APPROXIMATE COMPUTING FOR ENERGY EFFICIENT ARITHMETIC UNITS

Mini Project

By

Johan Rinu Jacob & Manal Ahmed

Roll.no. 231EC127 & 231EC135

Department of Electronics and Communication Engineering

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA (NITK)

SURATHKAL, MANGALORE- 575 025

May 6, 2025

Certificate

This is to certify that the project titled “Approximate Computing for Energy Efficient Arithmetic Units” submitted by Johan Rinu Jacob(231EC127) and Manal Ahmed (231EC135) in the record of the work carried out under mini-project, is accepted as the Mini-project submission in fulfillment of the requirements for the award of credits.

Dr. Sushil Kumar Pandey

Guide. Assistant Professor

Department of ECE, NITK Surathkal-575025

(Signature with Date and Seal)

Declaration

We hereby declare that the unit project titled "Approximate Multipliers for Energy Efficient Computing" which is to be authorized by the National Institute of Technology Karnataka, Surathkal in fulfillment of the requirements for the award of credits. This is the foundation report of the unit project work carried out by Manal Ahmed Matheen and Johan Rinu Jacob under the guidance of Dr. Sushil Kumar Pandey.

The project represents our independent efforts and is free from any form of plagiarism. The materials referred to in this work have been duly acknowledged and cited in the reference section.

We affirm that the findings and outcomes of this project are based on our research and experiments and have not been submitted elsewhere for academic evaluation.

Manal Ahmed Matheen, Johan Rinu Jacob

Roll. No: 231EC135, 231EC127
Department of Electronics and Communication Engineering
National Institute of Technology Karnataka, Surathkal
Date: May 2025

TABLE OF CONTENTS

Abstract

In modern digital systems, power consumption is a critical design constraint, especially in applications like image processing, machine learning, and sensor networks where slight inaccuracies are tolerable. Approximate computing emerges as a promising solution by relaxing exact computation requirements to achieve significant gains in energy efficiency, area, and speed. This project focuses on the design and analysis of an 8-bit approximate multiplier optimized for low-power and high-speed performance. By strategically introducing approximations in the multiplier architecture, we aim to reduce switching activity and logic complexity, thus improving energy efficiency while maintaining acceptable accuracy levels. The design will be evaluated based on power, delay, area, and error metrics.

1. Introduction

The growing demand for energy-efficient hardware accelerators has pushed researchers to explore non-traditional design methodologies. One such method is approximate computing, which leverages the error-tolerant nature of many applications to trade off computational accuracy for gains in energy, area, and speed. Multipliers, being one of the most power-hungry arithmetic units, are prime candidates for approximation.

In this project, we target the design of an 8-bit multiplier using approximate computing principles. The focus is on reducing power consumption and area while preserving acceptable output quality. Applications such as multimedia processing, machine learning inference, and sensor data analytics can benefit from such designs where perfect accuracy is not always essential. The study explores various approximation techniques and evaluates their impact on key design metrics.

2. Objective and problem statement

In digital systems, multipliers are key components that contribute significantly to the overall power consumption and area of arithmetic logic units. Traditional multipliers are designed for exact computation, which, while accurate, results in high energy usage and longer processing times. In contrast, approximate multipliers can significantly reduce power and area requirements by trading off some computational accuracy—making them ideal for error-tolerant applications such as image processing, neural networks, and multimedia.

This project aims to design and implement both an exact and an approximate 8-bit multiplier using Verilog, and to compare their outputs in real time. The core objective is to calculate the percentage error between the outputs of the exact and approximate multipliers for a given input, and display this approx output on a 7-segment display. This practical comparison will help visualize the trade-off between accuracy and efficiency in approximate computing.

The system will be implemented on FPGA hardware, where both multipliers will be given the same 8-bit inputs. The outputs will be analyzed to compute the error percentage, providing valuable insight into the feasibility of approximate arithmetic units for low-power and real-time applications.

3. Literature Review

Approximate computing has gained significant attention in recent years as a method to reduce power consumption and circuit complexity in digital systems. This is particularly relevant for arithmetic units like multipliers, which are among the most power-hungry components in processors. The core idea is to introduce controlled imprecision in the computation process to achieve energy efficiency, lower delay, and reduced silicon area, especially in error-resilient applications such as image processing, neural networks, and sensor data analysis.

Researchers have proposed several approximate multiplier architectures based on different techniques:

- Truncation-based approaches, where the least significant bits of partial products are ignored to simplify logic (Kulkarni et al., 2011).
- Segmented multipliers, which divide inputs into sections and apply approximation only in selected regions.
- Compressor-tree and carry-save based approximate multipliers, which use simplified adder logic to reduce critical path delay and gate count.

Some works have also focused on hybrid designs, combining accurate and approximate parts to maintain a balance between accuracy and energy savings. For example, Gupta et al. (2013) introduced multipliers that compute higher-order bits accurately while approximating lower-order bits.

Importantly, various error metrics such as Mean Error Distance (MED), Mean Relative Error (MRE), and Error Rate (ER) have been introduced to quantify the trade-off between accuracy and efficiency.

In recent studies, FPGA-based implementations of approximate multipliers have been used to experimentally validate power and area reductions. Moreover, visual feedback using displays or LEDs has been proposed to help understand accuracy trade-offs in real time—an idea your project extends by displaying error percentage on a 7-segment display.

4. Methodology

(i) Hardware Requirements

1. FPGA Development Board
   - Example: Xilinx Spartan-6, Xilinx Artix-7, or Intel Cyclone IV/V
   - Must include: I/O pins, clock source, and 7-segment display or provision to connect external one.

2. 7-Segment Display
    - Onboard or external (Common cathode/anode depending on board compatibility)
    - Used to display the approx output of the approximate multiplier.
3. Power Supply / USB Power Cable
    - To power the FPGA board (via USB or external adapter depending on board).
4. JTAG Programmer / USB Cable
    - For downloading the Verilog bitstream (e.g., *Xilinx Platform Cable USB or onboard USB-JTAG*).
5. Switches/Buttons (optional)
    - To provide manual input for testing different 8-bit values.
6. LEDs (optional)
    - For visual output/debugging (e.g., to show multiplier outputs or control signals).
7. Personal Computer (PC/Laptop)
    - To write Verilog code, simulate, synthesize, and program the FPGA.

(ii) Software Requirements

1. Xilinx Vivado Design Suite
    - Version: Preferably the latest version compatible with your FPGA board (e.g., Xilinx Spartan-6, Artix-7, etc.)
    - Purpose:
        - Write, simulate, and synthesize Verilog code for the exact and approximate 8-bit multipliers.
        - Generate bitstream files for FPGA programming.
        - Implement the design onto the FPGA board.
        - Manage I/O constraints and perform timing analysis.

2. Vivado Simulator (ISim)
   - Purpose:
     - Simulate Verilog code to verify functionality before hardware implementation.
     - Test and debug the exact and approximate multiplier designs.
     - Simulate the error percentage calculation logic.
3. Xilinx Virtual Cable (JTAG)
   - Purpose:
     - For hardware programming, using a USB-to-JTAG programmer (e.g., Xilinx Platform Cable USB).
     - Download the bitstream to the FPGA and program the FPGA device.
4. Xilinx SDK (optional, if you need embedded processing)
   - Purpose:
     - For designing and deploying software applications that may interact with the FPGA hardware (if needed for your project).

(iii) Working principle

The project operates on the principle of approximate computing, which reduces hardware complexity and power consumption by allowing minor errors in computation — a trade-off that is acceptable in many real-world applications.

In this system, two multipliers are implemented:

1. Exact 8-bit Multiplier: Uses full-precision logic (e.g., array or Wallace tree multiplier) to calculate the exact product of two 8-bit numbers.
2. Approximate 8-bit Multiplier: Uses simplified logic (e.g., truncated partial products, simplified adders, or omission of carry propagation) to compute an imprecise but faster and more energy-efficient result.

The system follows these steps:

1. Inputs: Two 8-bit binary numbers are provided as inputs—manually using switches, or programmatically through test input vectors.
2. Parallel Multiplication: Both exact and approximate multipliers process the same input values in parallel.

3. Output Comparison:
   - The system computes the difference between the two outputs (exact and approximate).
   - It then calculates the error percentage, typically using:

     Error (%)= (|Pexact−Papprox|Pexact) × 100

If the exact output is zero, a zero error or a predefined error value is used to avoid division by zero.

4. Display Logic:
   - We turn on the switches in the FPGA based on the two inputs
   - The display then shows the approximate output, helping users visually observe the impact of approximation.

(iv) Verilog Design Explanation

The entire system is modularly designed in Verilog, consisting of the following core components:

1. Exact 8-bit Multiplier Module

- Implements full-precision multiplication (e.g., array multiplier or shift-and-add).
- Takes two 8-bit inputs (A and B) and produces a 16-bit product P_exact.
- Fully accurate and uses standard logic gates (AND, full-adders).

2. Approximate 8-bit Multiplier Module

- Designed to reduce logic complexity and power usage.
- Example techniques used:
  - Truncation: Ignoring the lower 2–4 bits of the partial product.
  - Simplified adders: Replacing full adders with half adders or logic-only expressions.
  - Omitting carry chains in lower bits.
- Takes the same inputs A and B and produces an approximate 16-bit output P_approx.

3. Error Percentage Calculator Module

- Computes the absolute error:

  $E = |P\_exact - P\_approx|$

- Then computes the error percentage using approximate division logic or lookup tables (as full division in hardware is complex).
- Handles special cases like divide-by-zero by displaying zero or a max error flag.

4. 7-Segment Display Controller

- Takes the two inputs and the switches are turned on based on them
- Displays the approximate output.

5. Top-Level Module

- Connects all submodules:
  - Routes inputs A and B to both multipliers.
  - Feeds the outputs to the error calculator.
  - Sends the calculated error to the display unit.
- Maps I/O pins to switches (inputs), LEDs (optional debug), and 7-segment display.

Simulation

- Performed using Vivado Simulator (XSIM).
- Testbench provides various A and B input combinations.
- Simulates and verifies:
  - Correct operation of exact and approximate multipliers.
  - Correct calculation and formatting of error.
  - Proper display on 7-segment outputs.

(v) Pin Configuration and XDC File

In Vivado, the XDC (Xilinx Design Constraints) file is used to assign physical FPGA pins to your input and output ports defined in Verilog. This ensures that the signals from your Verilog modules (e.g., 8-bit inputs, outputs, 7-segment display) are properly mapped to the actual hardware components on your FPGA board.

For our project, the key pin mappings include:

- Clock input (e.g., 100 MHz clock)
- Switches or buttons for feeding input values to the multipliers
- 7-segment display pins to show the approximate output
- Optional LEDs for output observation or debugging

The XDC file specifies:

- Which FPGA pins connect to each signal
- The I/O standard (typically LVCMOS33)
- Any timing constraints (like clock frequency)

(vi) Flowchart

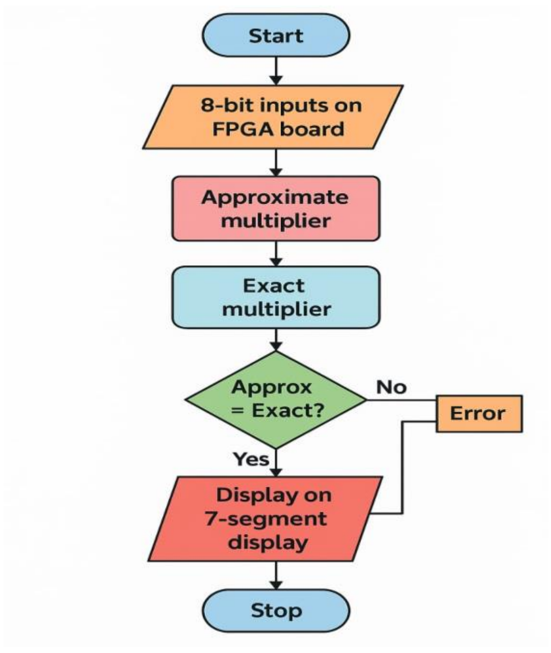


Fig:0 - Flowchart

5. Seven-Segment display

To display the approximate multiplication result on a 7-segment display, a series of steps are followed. First, the last two bits of both input values a and b are truncated to reduce hardware complexity and improve energy efficiency. The truncated inputs are then multiplied to produce a 14-bit approximate product. This result is divided into four 4-bit segments (nibbles), each representing a hexadecimal digit to be displayed. Since the 7-segment display typically shares segment control lines among all digits, multiplexing is used to cycle through the digits—activating one digit at a time in quick succession to create the illusion of a continuous display. Each nibble is decoded into its corresponding 7-segment pattern, which determines which segments (a–g) are illuminated for that digit. The appropriate digit is activated using the digit enable signals (an[3:0]). The segment control logic is adjusted based on whether the display is common cathode or common anode. This entire process is repeated rapidly to ensure all digits appear lit simultaneously and maintain a stable visual output.

6. Simulation Results

(i) Software results

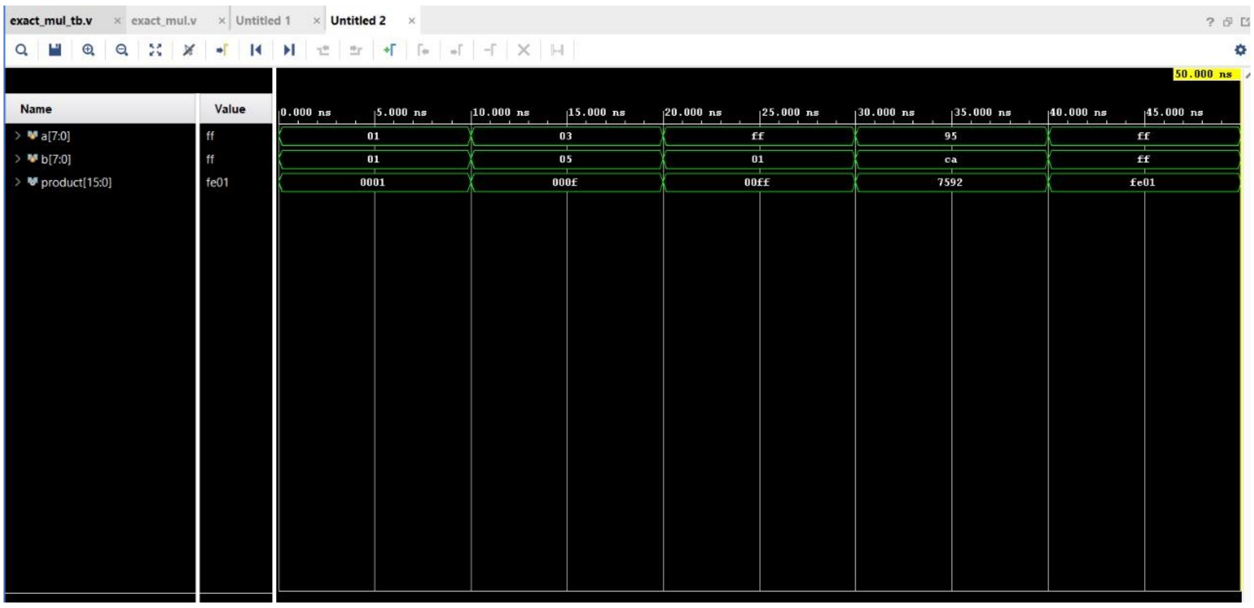


Fig:1-Exact multiplier waveform

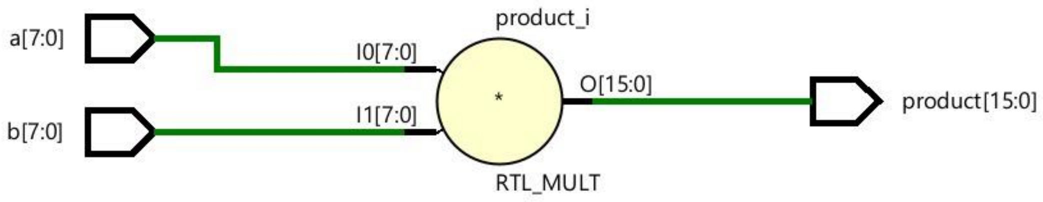
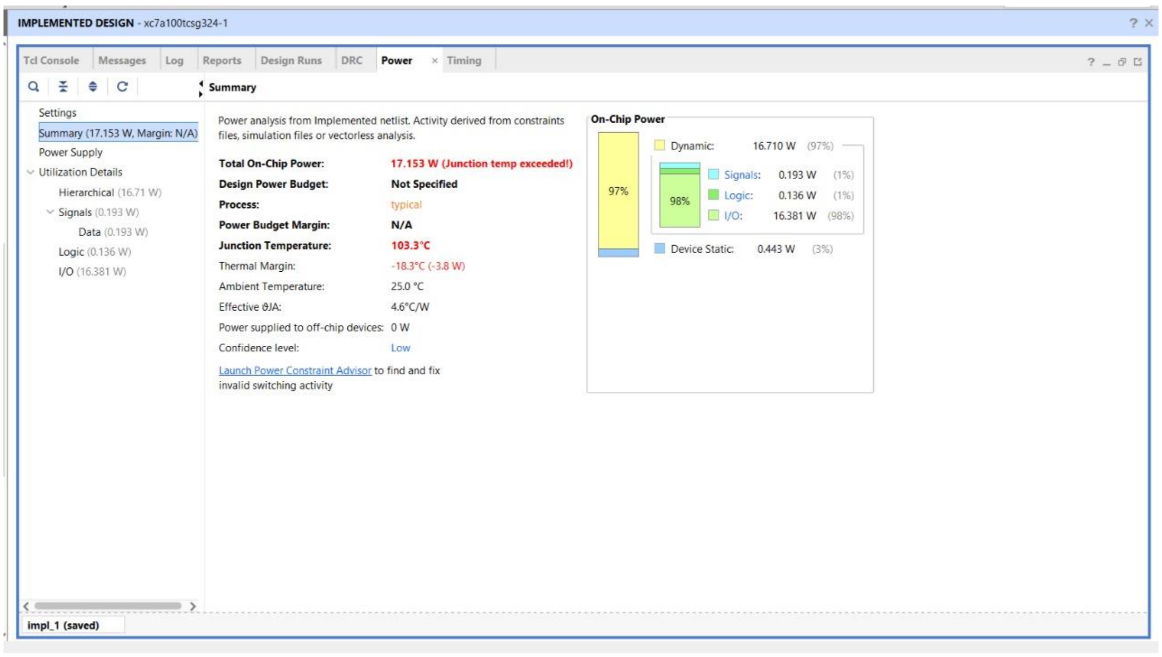


Fig:2-Exact multiplier schematic



Settings
Summary (17.153 W, Margin: N/A)
Power Supply
Utilization Details
Hierarchical (16.71 W)
Signals (0.193 W)
Data (0.193 W)
Logic (0.136 W)
I/O (16.381 W)

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**17.153 W (Junction temp exceeded!)**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**103.3°C**
Thermal Margin:	-18.3°C (-3.8 W)
Ambient Temperature:	25.0 °C
Effective θJA:	4.6°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

97%

98%

Dynamic:	16.710 W	(97%)	
	Signals:	0.193 W	(1%)
	Logic:	0.136 W	(1%)
	I/O:	16.381 W	(98%)
Device Static:	0.443 W	(3%)	

Fig:3-Exact multiplier power analysis

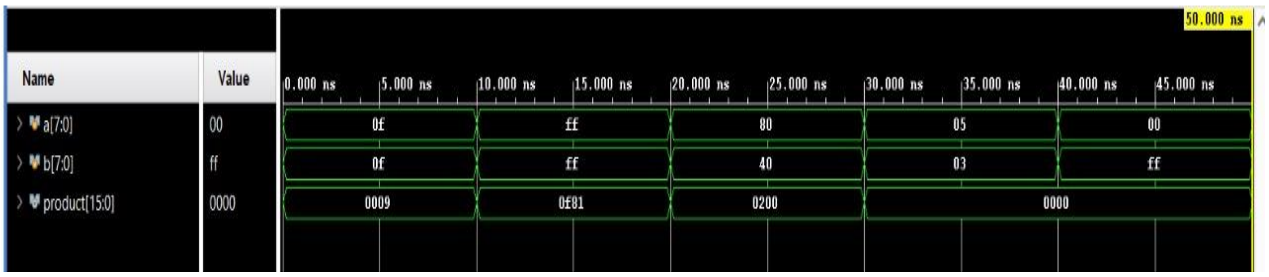


Fig:4-Approx multiplier waveform

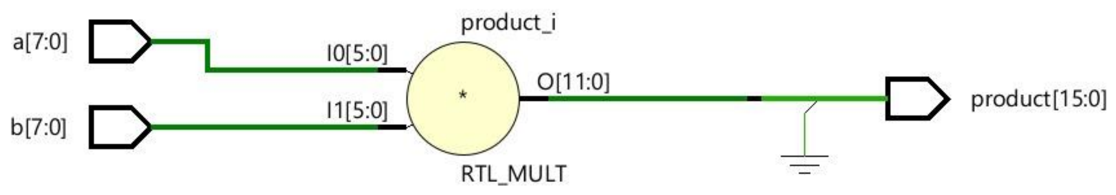
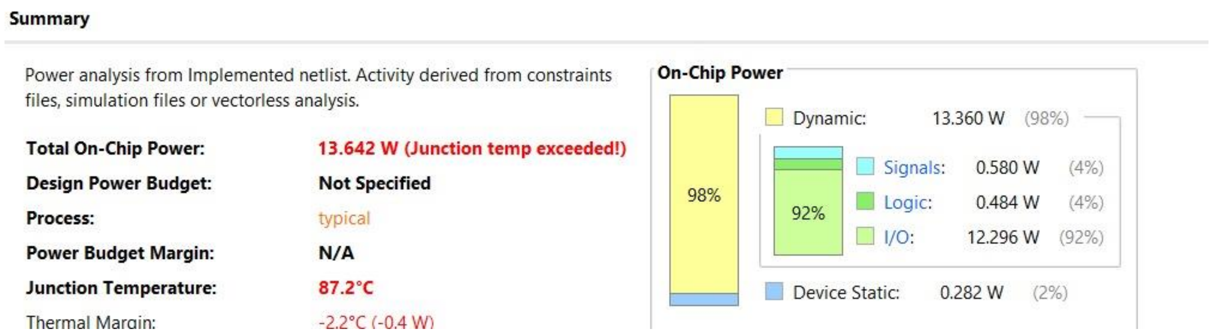


Fig:5-Approx multiplier schematic



Summary

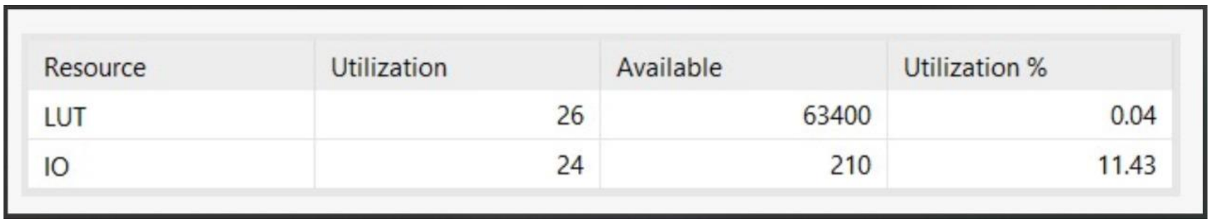
Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**13.642 W (Junction temp exceeded!)**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**87.2°C**
Thermal Margin:	-2.2°C (-0.4 W)

On-Chip Power

Dynamic:		13.360 W	(98%)
	Signals:	0.580 W	(4%)
	Logic:	0.484 W	(4%)
	I/O:	12.296 W	(92%)
Device Static:		0.282 W	(2%)

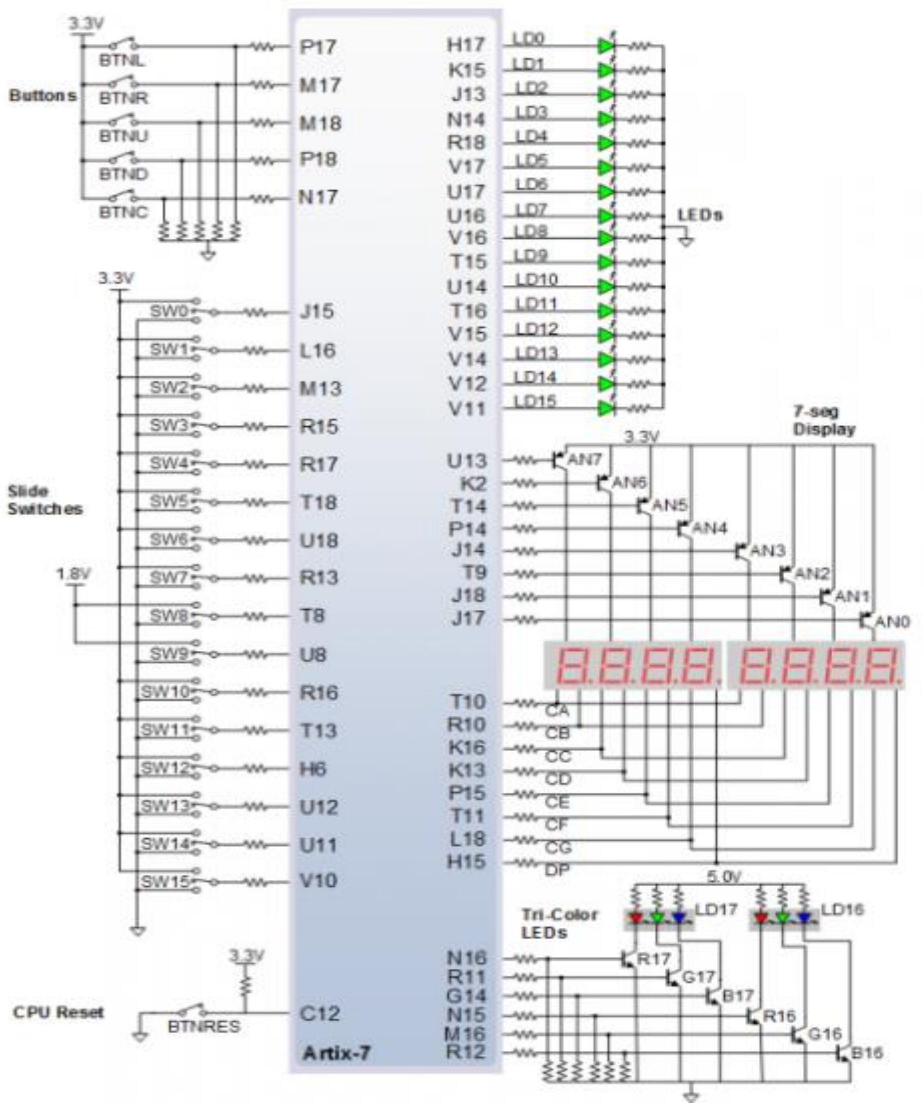
Fig:6-Approx multiplier power analysis



Resource	Utilization	Available	Utilization %
LUT	26	63400	0.04
IO	24	210	11.43

Fig:7-LUT Utilization by Approximate Multiplier

(ii) Hardware implementation results



<u>Fig:8</u> – Pin Diagram Interface of Nexys 4

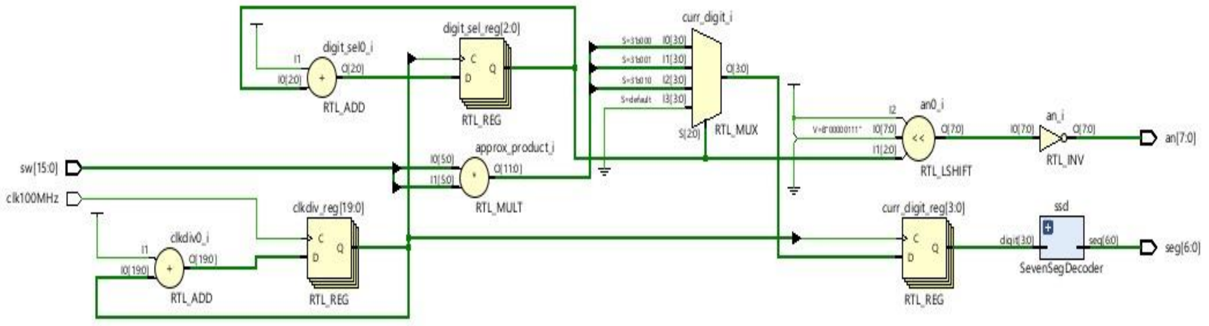


Fig:9 - Block Diagram Implementation of our Approximate Multiplier for implementation in FPGA

(Output Display in Anode – SevenSegment Display)

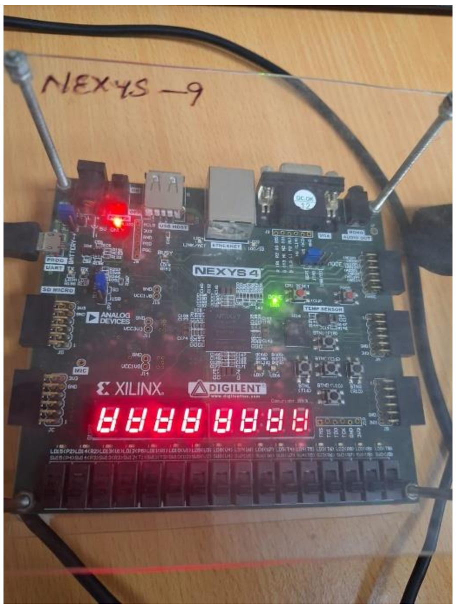


Fig:10 - Initial display

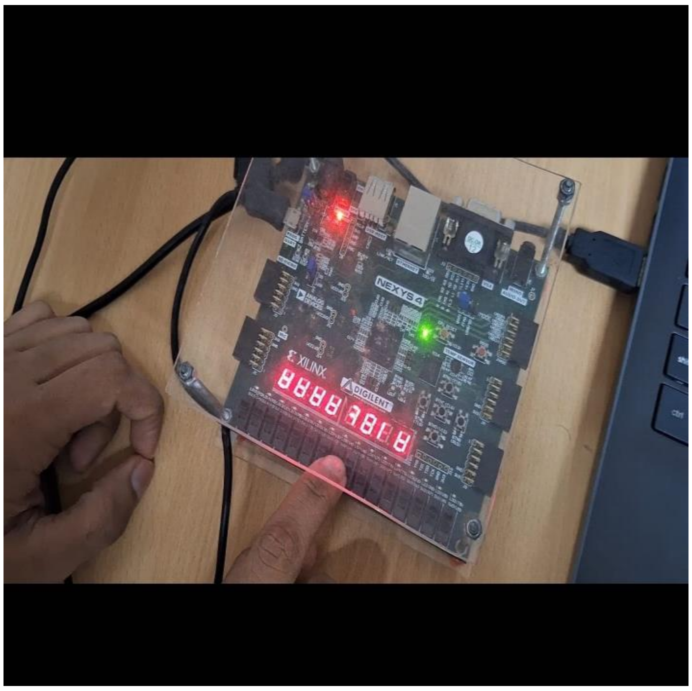


Fig:11-Final output display

7. Advantages of FPGA design

FPGAs offer several advantages that make them highly suitable for implementing approximate multipliers. Firstly, their ability to perform **parallel processing** enables efficient execution of multiple arithmetic operations simultaneously, which is ideal for multiplier-based computations. Secondly, they provide **hardware-level customization**, allowing designers to optimize both exact and approximate multiplier architectures down to the gate level for improved performance and energy savings. With **real-time processing** capabilities, FPGAs support low-latency operation, which is essential for immediate error computation and display in embedded applications. Their **reconfigurability** also allows for rapid design changes, making them an excellent platform for prototyping and evaluating various approximation techniques. Moreover, the ability to implement **energy-efficient custom architectures** is a key benefit, offering advantages over general-purpose processors. FPGAs also enable **concurrent verification**, where simulation and hardware testing can occur side-by-side to ensure functional correctness. Finally, FPGAs are **cost-effective** for development and small-scale production, providing a more economical alternative to ASICs for custom arithmetic unit design.

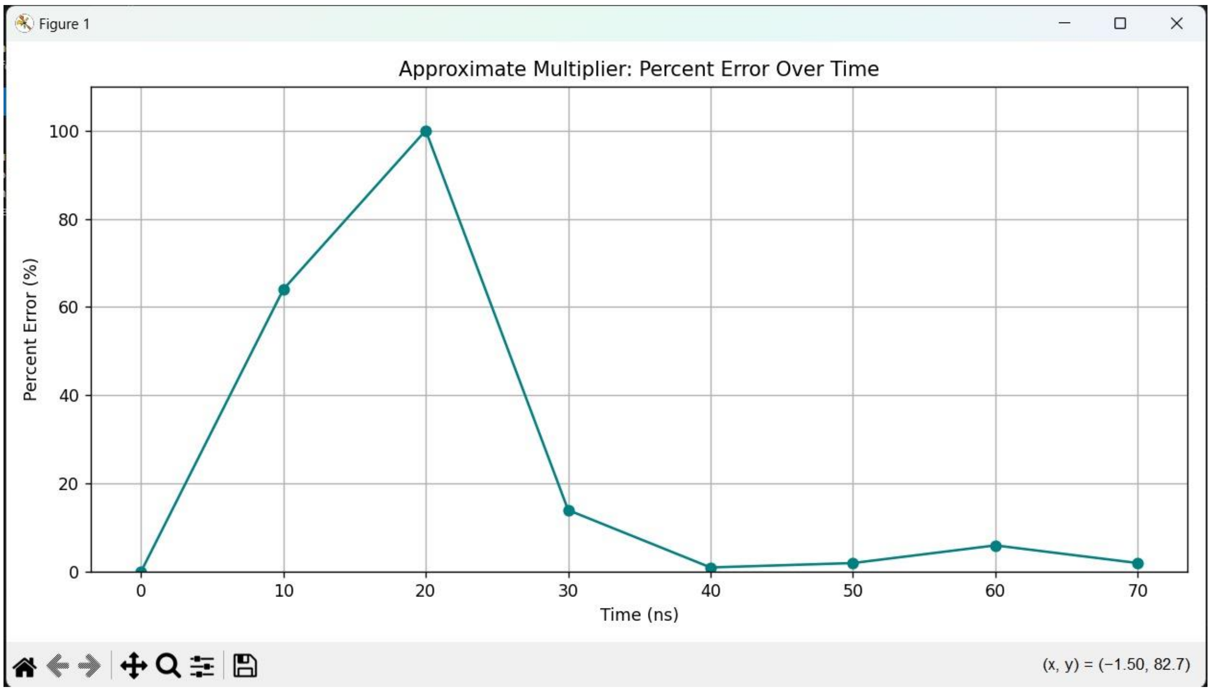


Fig:12- The percentage error decreases over time as we increase the computational burden

8. Challenges faced

Complex Error Calculation in Hardware:
Implementing real-time error percentage computation on an FPGA presents a significant challenge, primarily due to limited native support for floating-point arithmetic. Division operations required for percentage calculations are especially resource-intensive and complex to realize efficiently in hardware.

7-Segment Display Control:
Controlling a 7-segment display to show multi-digit approximate output adds complexity to the design. Digit multiplexing, segment decoding, and precise timing control are required to ensure stable and readable output, all of which demand careful coordination within the FPGA logic.

Resource Constraints:
FPGAs have finite logic resources, and implementing both exact and approximate multipliers along with comparison logic may strain these resources—especially on smaller or entry-level FPGA boards. Optimizing utilization is crucial to avoid synthesis failures or performance bottlenecks.

Debugging Hardware Logic:
Unlike software, debugging on FPGA involves limited visibility into internal signals. Identifying logic errors or timing issues often requires the use of additional tools such as the Integrated Logic Analyzer (ILA), which must be configured and integrated carefully.

Clock Management:
Generating appropriately divided clock signals for display multiplexing and timing control without interfering with the core multiplier logic is non-trivial. In our design, we use a 100 MHz clock signal connected to the FPGA via pin 'E3'. Managing this clock and deriving slower signals reliably is essential to prevent display flicker or synchronization issues.

Approximate Logic Design:
Designing an effective approximate multiplier involves making trade-offs between error tolerance, power consumption, logic area, and speed. Selecting the right approximation method depends on the application requirements and must be carefully evaluated to maintain an acceptable balance.

Timing Violations and Constraints:
Ensuring the design meets all timing requirements is especially challenging in multi-module systems. High-speed paths and sequential dependencies can lead to setup or hold time violations if not handled properly during synthesis and implementation.

Pin Mapping and XDC Constraints:
Proper pin assignment is critical for correct hardware operation. Configuring the XDC (constraints) file to map inputs and outputs like switches, LEDs, and 7-segment displays requires precision, as incorrect mapping can lead to unexpected behavior or failed programming**.**

9. Applications

Low-Power Embedded Systems:
Approximate multipliers are particularly useful in low-power embedded systems such as wearables, IoT nodes, and wireless sensor networks. In these devices, energy efficiency often takes precedence over exact numerical accuracy. By allowing small computational errors, approximate computing helps extend battery life and reduce overall power consumption without significantly affecting functionality.

Image and Video Processing:
In applications like image enhancement, compression, and video filtering, multipliers are widely used in operations such as convolution and discrete cosine transforms (DCT). These tasks can tolerate a degree of imprecision, making approximate multipliers an excellent fit for real-time image and video processing where speed and power efficiency are critical.

Machine Learning and AI Accelerators:
Machine learning models, especially neural networks during inference, are inherently tolerant to minor errors in computation. Approximate multipliers can be used to accelerate inference tasks by reducing latency and energy usage, all while maintaining an acceptable level of model accuracy. This makes them ideal for deployment in energy-constrained AI accelerators.

Digital Signal Processing (DSP):
DSP applications like audio processing, radar systems, and wireless communication often involve intensive mathematical operations. In many cases, slight approximation in multiplication can yield significant performance improvements, enabling faster processing with minimal perceptible loss in output quality.

Edge Computing Devices:
Devices operating at the edge of a network, such as smart cameras, edge AI chips, and embedded controllers, face strict constraints on power and thermal budgets. Using approximate multipliers helps reduce the computational load and energy demand, which is crucial for maintaining real-time operation and responsiveness.

Consumer Electronics:
Smartphones, digital televisions, gaming consoles, and other consumer electronics benefit from approximate computing by achieving better energy efficiency. Approximate multipliers reduce power consumption and heat generation, contributing to longer battery life and more compact thermal designs in handheld or portable devices.

Robotics and Autonomous Systems:
In robotics, drones, and autonomous vehicles, real-time control and data processing are essential. Slight errors in arithmetic operations are often tolerable if they enable faster and more energy-efficient decision-making. Approximate multipliers support these requirements by improving computational efficiency without compromising control stability.

10. Conclusion

This project demonstrated the potential of approximate computing in energy-efficient design for FPGA-based arithmetic units. By implementing both exact and approximate 8-bit multipliers, we were able to evaluate the trade-offs between computational accuracy and energy consumption. The approximate multiplier, while sacrificing some precision, provides significant advantages in terms of power efficiency and processing speed, making it ideal for resource-constrained applications.

We achieved real-time error calculation, with the error percentage displayed on a 7-segment display, offering insights into the trade-offs made between accuracy and performance. The findings highlight the viability of approximate computing for systems where speed and energy efficiency are prioritized over absolute precision, such as in IoT devices, embedded systems, and edge computing.

In future work, this approach can be extended to more complex arithmetic units and optimized for larger-scale systems. The ability to balance performance, power, and accuracy will continue to play a crucial role in the evolution of low-power hardware design.

11. Future scope

The concept of approximate computing in energy-efficient designs offers a wide range of possibilities for further development and optimization. Some key areas for future work include:

1. Extension to Larger Multipliers
   - The current design focuses on an 8-bit multiplier. Extending this to 16-bit or 32-bit multipliers would improve its applicability in more complex arithmetic units, such as those used in image processing, cryptography, and AI inference engines.
2. Optimization of Approximation Techniques
   - Further research can explore different approximation strategies (e.g., stochastic computing, reduced precision techniques) to achieve better trade-offs between error, speed, and power consumption. This could lead to more efficient designs in low-power devices.
3. Hardware-Software Co-design
   - Integrating approximate hardware designs with software-level algorithms (such as machine learning models) could further enhance performance. Hybrid approaches that combine approximate hardware with adaptive software can help improve accuracy when needed.
4. Multi-core and Parallel Systems
   - FPGAs with multiple cores or parallel units could be used to deploy several approximate multipliers in a multi-threaded architecture, improving throughput while maintaining low energy consumption.
5. Dynamic Approximation
   - Implementing dynamic approximation where the level of approximation can be adjusted based on the system load or real-time requirements could provide optimal energy savings in diverse operating conditions.
6. Error-Resilient Applications
   - Researching error-resilient algorithms that can tolerate higher levels of approximation, such as in audio compression, neural networks, or real-time video processing, will help expand the range of applications for approximate computing.

7. Advanced Power Optimization
    - Further investigation into power gating and clock gating techniques could help reduce power consumption even further, making approximate computing viable for ultra-low-power applications, such as wearable devices and medical implants.
8. Integration with Emerging Technologies
    - The future of approximate computing could also include its integration into quantum computing, neuromorphic computing, or memristor-based systems, where approximate results are acceptable and energy efficiency is paramount.

Note On Technology Node :

Truncating 8×8-bit multipliers to 6×6 bits on Artix-7 FPGAs (28nm CMOS) enhances performance and efficiency by reducing critical path delays, enabling higher clock frequencies through simplified logic. Dynamic power consumption decreases due to fewer toggling gates and reduced fanout, while the smaller logic footprint minimizes leakage, maintaining power efficiency despite 28nm process characteristics. The design consumes fewer LUTs and routing resources, freeing space for parallel modules or pipelining, and shorter interconnects simplify placement/routing, improving timing closure in Vivado. Tighter timing margins ease integration with multi-clock domains and output peripherals like 7-segment displays, while the 12-bit truncated result simplifies display multiplexing. These optimizations also enhance scalability for multi-unit designs or hybrid exact/approximate arithmetic systems.

12. References

Becher, J. Echavarria, D. Ziener, and J. Teich, “Approximate Adder Structures on FPGAs,” Dept. of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany. [Online]. Available: https://ris.utwente.nl/ws/portalfiles/portal/28605262/paper.pdf

G. Shobana, R. Chithiraimuthu, and A. Adhithyavel, “Performance analysis and implementation of approximate multipliers on Spartan 6 FPGA,” *Int. J. Health Sci.*, vol. 6, no. S1, pp. 10633–10652, 2022. [Online]. Available: https://www.researchgate.net/publication/360747922_Performance_analysis_and_implementation_of_approximate_multipliers_on_spartan_6_FPGA

M.A. van Loo, *van Loo MA EEMSC*, M.Sc. thesis, Computer Architecture for Embedded Systems (CAES), Faculty of EEMCS, Univ. of Twente, Enschede, The Netherlands, supervised by Dr. ir. N. Alachiotis, Dr. ir. S.G.A. Gillani, and Dr. ir. A.B.J. Kokkeler. [Online]. Available: https://essay.utwente.nl/93756/1/van%20Loo_MA_EEMSC.pdf

Digilent Inc., *Nexys4 DDR FPGA Board Reference Manual*, Rev. C, Sep. 11, 2014. [Online]. Available: https://digilent.com/reference/programmable-logic/nexys-4-ddr/reference-manual