VLSI Implementation of Adders and Approximation Using ALU

Mr. Chethan B R, Divyashree G E, Harishree M, L V Charan, Mithun H R
Mr. Chethan B R, ECE & PES Institute of Technology and Management
Mrs. Divyashree G E, ECE & PES Institute of Technology and Management
Mrs. Harishree M, ECE & PES Institute of Technology and Management
Mr. L V Charan, ECE & PES Institute of Technology and Management
Mr. Mithun H R, ECE & PES Institute of Technology and Management

---***---

Abstract – This project presents the VLSI implementation of different adder architectures and the approximation of Arithmetic Logic Units (ALUs) to optimize speed, power, and area. Various adders such as Carry Save, Carry Skip, Carry Select, Kogge-Stone, Sklansky, and Carry Lookahead were designed and simulated using Cadence tool. Their performance was analyzed based on parameters like delay, power, and area utilization. Among them, the Carry Lookahead Adder showed the best overall efficiency with minimal delay and moderate power consumption. The project concludes that approximate computing and optimized VLSI designs can enhance system performance in energy-efficient and real-time applications.

Keywords: VLSI, ALU, adders, cadence tool, delay, power, area.

1. INTRODUCTION

Very Large-Scale Integration (VLSI) technology enables the integration of millions of transistors onto a single chip, allowing the design of compact, high-speed, and energy-efficient digital systems. In digital electronics, adders and the Arithmetic Logic Unit (ALU) play a crucial role in performing arithmetic and logical operations that form the foundation of processors. Efficient implementation of these components directly impacts the overall performance, speed, and power consumption of digital circuits. This project focuses on designing and analyzing various adder architectures using VLSI techniques to achieve optimal trade-offs between area, delay, and power. Additionally, it explores approximate ALU designs suitable for error-tolerant applications like image processing and machine learning, where reduced power and area are prioritized over absolute accuracy.

2. LITERATURE REVIEW

Modern VLSI adder research has focused on balancing latency, area, and power across a variety of architectures. Classical ripple-carry and carry-lookahead adders remain useful for small widths, but parallel-prefix (tree) adders — e.g., Kogge-Stone, Brent-Kung, Han-Carlson and sparse variants — dominate for wide, high-performance designs because they minimize critical-path depth at the cost of increased wiring and area; many comparative studies quantify these tradeoffs across CMOS design flows and technology nodes. Implementation papers typically evaluate adders at multiple abstraction levels (gate/cell, RTL-synthesis, and physical layout), showing that the "best" adder depends on target metrics: Kogge-Stone gives lowest delay, Brent-Kung and sparse trees lower area/wiring, and hybrid carry-select/skip styles can give attractive midpoint tradeoffs for medium speed/area constraints. Practical VLSI work also highlights routing congestion, fanout, and power-clock interaction as primary real-world constraints beyond pure

gate-delay analysis. Approximate computing — applied to ALUs and arithmetic units — has emerged as an effective way to trade correctness for energy, area and speed in error-tolerant applications (image/video processing, ML inference, sensor nodes). Surveys and recent papers categorize approximation approaches at multiple levels: algorithmic (reduced precision), architectural (approximate adders/multipliers, truncated or segmented arithmetic), circuit-level (inexact gates, transistor sizing, voltage over scaling), and system/runtime techniques that adapt approximation to QoS requirements. In the adder literature, families of approximate full-adder designs (e.g., truncated LSBs, hybrid error-reduction schemes, inexact logic primitives) have been proposed and evaluated; results consistently show substantial energy and area savings (often >20–50%) with bounded error metrics (mean error, worst-case error, and application QoR). Recent VLSI studies also stress that the most practical approximate ALUs are those that combine circuit-level simplifications with architecture-level error mitigation and evaluate designs with application-level error-impact metrics rather than gate-level error alone. Open research directions include adaptive approximation (dynamic fidelity tuning), better error metrics tied to end-applications, and layout-aware approximate designs that account for parasitics and variability.

3. METHODOLOGY

The project methodology involves systematic design, implementation, simulation, and analysis of different adder architectures using VLSI design techniques. The following steps were followed throughout the project:

1. Problem Identification and Objective Definition:

The first step was to identify the need for high-speed and low-power arithmetic circuits in modern VLSI systems. The objective was to design and compare various adder architectures to determine the most efficient one in terms of speed, area, and power.

2. Selection of Adders:

Various adder architectures such as Carry Save Adder (CSA), Carry Skip Adder (CSKA), Carry Select Adder (CSeA), Kogge-Stone Adder (KSA), Sklansky Adder, and Carry Lookahead Adder (CLA) were selected based on their performance characteristics.

2.a) EXPLANATION OF ADDERS

1. Carry Save Adder (CSA):

Idea: Adds multiple numbers at once without immediately calculating carries.
How it works: Saves (stores) the carry bits instead of propagating them immediately.
Use: Common in multipliers where several partial sums are added.
Advantage: Very fast for adding more than two numbers.
Disadvantage: Final carry needs to be resolved later using another adder.

2. Carry Skip Adder (CSkA):

Idea: Skips carry computation over certain blocks when not needed.
How it works: Divides bits into blocks; if all bits in a block can propagate a carry, the carry "skips" the block.
Advantage: Faster than Ripple Carry Adder.
Disadvantage: Slightly more complex hardware than ripple adder.

3. Carry Lookahead Adder (CLA):

Idea: Predicts carry bits in advance using logic equations.
How it works: Uses "generate" and "propagate" signals to compute carries in parallel.
Advantage: Much faster than ripple adder; reduces carry delay.
Disadvantage: Complex for large bit-widths (more hardware).

4. Kogge-Stone Adder (KSA):

Idea: A parallel prefix adder that computes carries in a tree structure.
How it works: Carries are calculated in multiple stages using generate/propagate signals.
Advantage: Very fast — one of the fastest adders.
Disadvantage: Requires a lot of wiring and hardware.

5. Sklansky Adder (SkA):

Idea: Another parallel prefix adder with fewer logic levels.
How it works: Groups bits hierarchically to compute carries quickly.
Advantage: Fast and has minimal logic depth.
Disadvantage: Has high fan-out (one signal drives many others), which may slow down in hardware

3. Design Phase (RTL Coding):

Each adder was designed at the Register Transfer Level (RTL) using Verilog HDL. Structural and behavioral modeling techniques were used to describe the logic and data flow of each architecture.

4. Simulation and Functional Verification:

The Verilog codes were simulated using tools to verify the correctness of the designs. Testbenches were created for each adder to check their functionality for different input combinations

5. Synthesis and Implementation:

The verified designs were synthesized using Cadence synthesis tools, converting the RTL design into gate-level netlists. Implementation on an FPGA platform was performed to analyze practical performance metrics.

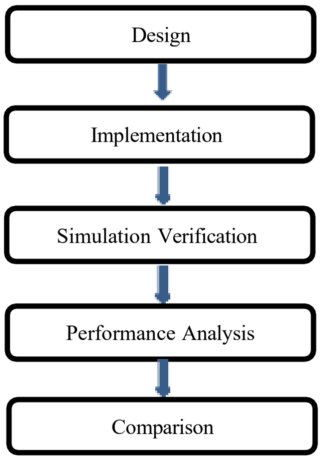
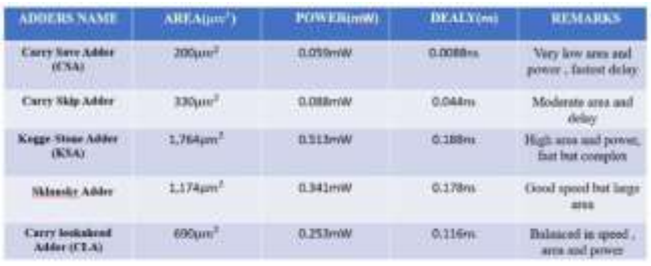


Fig -1: Block diagram

4. RESULTS:

4.a) COMPARITION OF (32-Bit) RESULTES:



ADDERS NAME	AREA(μm^2)	POWER(mW)	DEALY(ns)	REMARKS
Carry Save Adder (CSA)	200μm^2	0.059mW	0.0088ns	Very low area and power , fastest delay
Carry Skip Adder	330μm^2	0.088mW	0.044ns	Moderate area and delay
Kogge-Stone Adder (KSA)	1,764μm^2	0.513mW	0.188ns	High area and power, fast but complex
Sklansky Adder	1,174μm^2	0.341mW	0.178ns	Good speed but large area
Carry lookahead Adder (CLA)	690μm^2	0.253mW	0.116ns	Balanced in speed , area and power

Comparison of 32-bit Adders

In the 32-bit case, the impact of bit-width reduction changes the efficiency ranking slightly.

The Carry Save Adder (CSA) again achieves the lowest delay (0.0088 ns) and lowest power (0.059 mW) with minimal area (200 µm²), showing excellent performance in smaller bit designs.

The Carry Skip Adder maintains moderate area and delay, suitable for applications that prioritize simplicity.

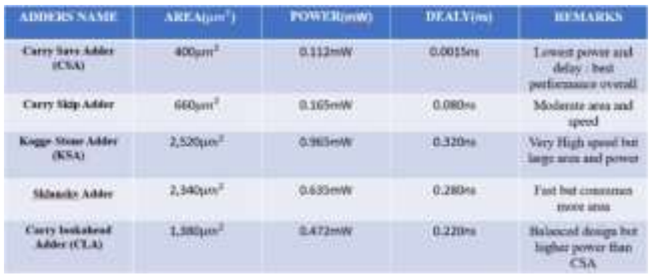
The Kogge-Stone Adder (KSA) offers very fast operation but suffers from high power (0.513 mW) and large area (1764 µm²) due to its complex interconnections.

The Sklansky Adder provides good speed but requires more area than simpler designs.

The Carry Lookahead Adder (CLA) achieves a balanced performance across area, power, and delay (0.116 ns), making it a practical choice for 32-bit implementations.

From this we can conclude that the **Carry Lookahead Adder (CLA)** is the best choice because it balances **speed, area,** and **power** efficiently.

4.b) **COMPARITION OF (64-Bit) RESULTES:**



ADDERS NAME	AREA(μm^2)	POWER(mW)	DEALY(ns)	REMARKS
Carry Save Adder (CSA)	400μm^2	0.112mW	0.0015ns	Lowest power and delay : best performance overall
Carry Skip Adder	660μm^2	0.165mW	0.080ns	Moderate area and speed
Kogge Stone Adder (KSA)	2,520μm^2	0.965mW	0.320ns	Very High speed but large area and power
Sklansky Adder	2,340μm^2	0.635mW	0.280ns	Fast but consumes more area
Carry lookahead Adder (CLA)	1,380μm^2	0.472mW	0.220ns	Balanced design but higher power than CSA

Comparison of 64-bit Adders

The 64-bit adder comparison clearly shows how design complexity affects performance.

The Carry Save Adder (CSA) demonstrates the lowest power (0.112 mW) and minimal delay (0.0015 ns) with the smallest area (400 µm²), making it the most efficient design overall in terms of speed, area, and power.

The Carry Skip Adder offers moderate performance with reasonable area and delay, serving as a balanced but less efficient option.

The Kogge-Stone Adder (KSA) achieves very high speed due to its parallel prefix structure but at the cost of large area (2520 µm²) and high power (0.965 mW).

The Sklansky Adder is also fast but consumes more area and power compared to the CSA.

The Carry Lookahead Adder (CLA) maintains a good balance between delay and power but is still less efficient than the CSA for 64-bit implementation.

Carry Save Adder (CSA) is the **most efficient adder** among the designs compared, offering the best balance of **speed, area, and power** efficiency.

4.c) **32-bit Carry Lookahead Adder (CLA) simulation:**

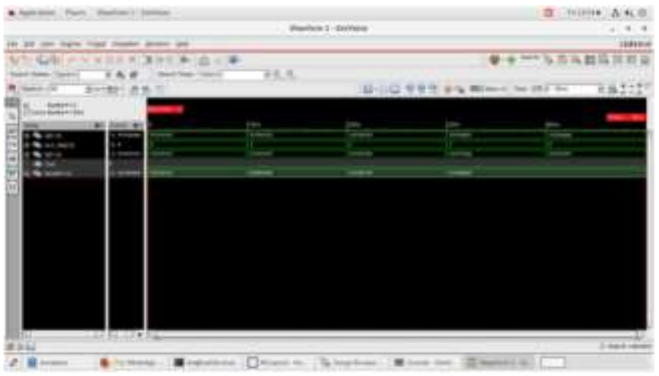


Fig -2(a): 32-bit (CLA) simulation

4.d) **64-bit Carry Save Adder (CSA) simulation:**

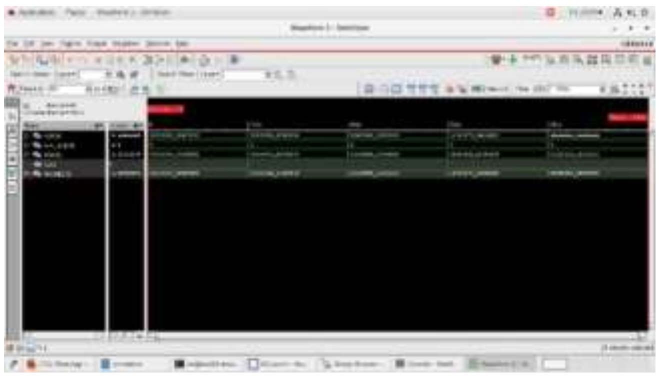
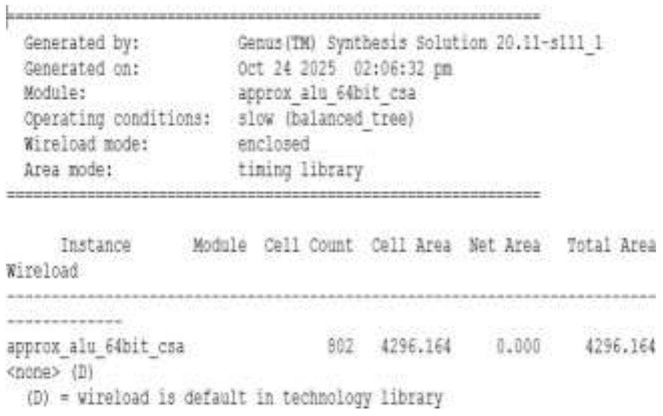


Fig -2(b): 64-bit (CSA) simulation

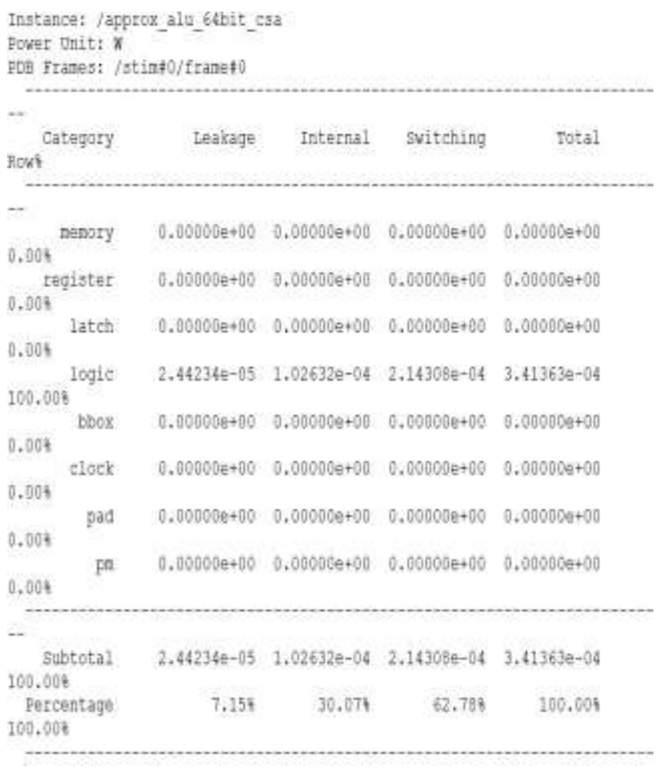
4.e) Synthesis Reports:

A) Report on AREA:



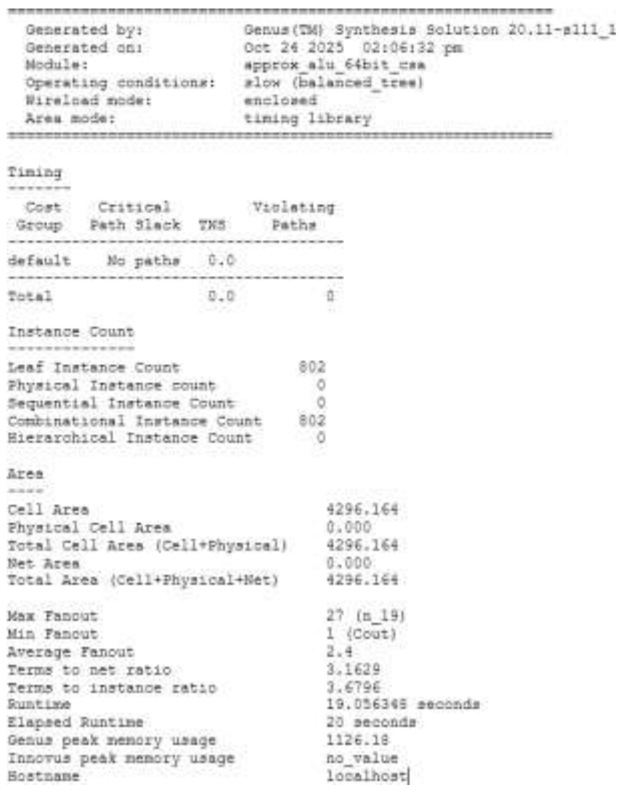
```
+===========================================================
  Generated by:          Genus(TM) Synthesis Solution 20.11-s111_1
  Generated on:          Oct 24 2025  02:06:32 pm
  Module:                approx_alu_64bit_csa
  Operating conditions:  slow (balanced_tree)
  Wireload mode:         enclosed
  Area mode:             timing library
============================================================

      Instance      Module  Cell Count  Cell Area  Net Area   Total Area
Wireload
-----------------------------------------------------------------------
--------------
approx_alu_64bit_csa              802   4296.164     0.000     4296.164
<none> (D)
  (D) = wireload is default in technology library
```

B) Report on POWER:



```
Instance: /approx_alu_64bit_csa
Power Unit: W
PDB Frames: /stim#0/frame#0
  ---------------------------------------------------------------------
--
    Category         Leakage     Internal    Switching        Total
Row%
  ---------------------------------------------------------------------
--
      memory     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00
0.00%
    register     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00
0.00%
       latch     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00
0.00%
       logic     2.44234e-05  1.02632e-04  2.14308e-04  3.41363e-04
100.00%
        bbox     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00
0.00%
       clock     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00
0.00%
         pad     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00
0.00%
          pm     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00
0.00%
  ---------------------------------------------------------------------
--
    Subtotal     2.44234e-05  1.02632e-04  2.14308e-04  3.41363e-04
100.00%
  Percentage           7.15%       30.07%       62.78%      100.00%
100.00%
  ---------------------------------------------------------------------
```

C) Report on DELAY:



```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 20.11-s111_1
  Generated on:           Oct 24 2025  02:06:32 pm
  Module:                 approx_alu_64bit_csa
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

Timing
-------
  Cost    Critical         Violating
 Group   Path Slack  TNS     Paths
-------------------------------------
default    No paths   0.0
-------------------------------------
Total                 0.0          0

Instance Count
--------------
Leaf Instance Count             802
Physical Instance count           0
Sequential Instance Count         0
Combinational Instance Count    802
Hierarchical Instance Count       0

Area
----
Cell Area                          4296.164
Physical Cell Area                 0.000
Total Cell Area (Cell+Physical)    4296.164
Net Area                           0.000
Total Area (Cell+Physical+Net)     4296.164

Max Fanout                         27 (n_19)
Min Fanout                         1 (Cout)
Average Fanout                     2.4
Terms to net ratio                 3.1629
Terms to instance ratio            3.6796
Runtime                            19.036348 seconds
Elapsed Runtime                    20 seconds
Genus peak memory usage            1126.18
Innovus peak memory usage          no_value
Hostname                           localhost
```

5. CONCLUSIONS

The study also explored approximation techniques in ALU design, which effectively reduced power consumption and circuit complexity with minimal loss in accuracy. Overall, the project achieved its objective of comparing and optimizing adder designs, contributing to the development of high-performance and low-power VLSI systems suitable for modern digital applications. In conclusion, the VLSI implementation of adders and their approximation using ALU plays a vital role in enhancing computational efficiency and reducing hardware complexity. By employing optimized adder architectures like carry look-ahead, carry skip, and parallel prefix adders, designers can achieve high-speed and low-power performance. Approximate adders further improve energy efficiency in error-tolerant applications such as image processing and machine learning. Overall, integrating these techniques into ALUs leads to faster, smaller, and more power-efficient VLSI systems suitable for modern digital applications. Moreover, VLSI-based design enables scalability and flexibility, allowing different adder architectures to be tailored for specific performance requirements. The trade-off between accuracy and efficiency in approximate computing provides a promising direction for low-power designs. Future advancements in nanotechnology and AI-driven optimization are expected to further enhance adder performance. Thus, the combination of precise and approximate adder designs in ALUs forms a strong foundation for next-generation high-performance and energy-efficient processors.

REFERENCES

1. Radha.N M, M Maheshwari, A. Abhinaya, “Feasible Implementation of ALU for next generation processors,"2024 Third International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT) | 979-8-3503-6908-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICEEICT61591.2024.10718541

2. Srilakshmi Kaza, Shymala yalagadda, “Ultra Low Power approximation arithmetic circuit,” 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT) | 979-8-3503-3509-5/23/$31.002023 IEEE |DOI:10.1109/ICCCNT56998.2023.1030

3. G. Surekha, Gajulan M Adesh, M Amidisette Pavan Kumar, “Design and Implementation of Arithmetic and ALU”, 2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC) | 978-1-6654-5630-2/23/$31.00 IEEE|DOI:10.1109/ICAAIC56838.2023.10140574

4. Balamanikandan A, Suresh K, “VLSI Implementation of Area-Error Optimized Compressor-Based Modified Wallace Tree Multiplier,” 2023 Second International Conference on Trends in Electrical, Electronics, and Computer Engineering (TEECCON)|979-8-3503-3994-9/23/$31.00 2023 (TEECCON) | 979-8-3503-3994-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/TEECCON59234.2023.10335812

5. Rounak Roy, Sudip Ghosh, Hafizur Rahaman “Implementation of Area Efficient Adders for Inexact computing,"2023 International Symposium on Devices, Circuits and Systems (ISDCS) | 979-8-3503-1504-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISDCS58735.2023.10153518

6. Rashi Pandey, “Implementation of Approximate Adders Circuit of Ladner Fischer Adder (16bit),” Fourth International Conference on Electronics, Communication and Aerospace Technology (ICECA-2020) IEEE Xplore Part Number: CFP20J88-ART; ISBN: 978-1-7281-6387-1