An 8-bit ALU with Comparable Study of Multiple Adder Architectures and Extended Operations: FPGA Implementation and Performance Analysis

Deepshikha Chatterjee, Electronics and Communication Engineering
Guru Nanak Institute of Technology, Kolkata

***Abstract*—The design and FPGA implementation of an 8-bit Arithmetic Logic Unit (ALU) utilizing the Carry Look-Ahead Adder (CLA), Carry Select Adder (CSA), and Ripple Carry Adder (RCA) architectures are described in this project. Arithmetic operations like addition, subtraction, increment, and decrement are supported by the ALU. Every module was synthesized using Xilinx Vivado and implemented in Verilog HDL. Post-synthesis reports were used to analyze area and combinational delay. According to the results, FPGA carry-chain optimization causes RCA and CLA to perform similarly, while CSA shows a marginally lower delay for the 8-bit design.**

***Index Terms*—Verilog HDL, FPGA, RCA, CLA, CSA, ALU**

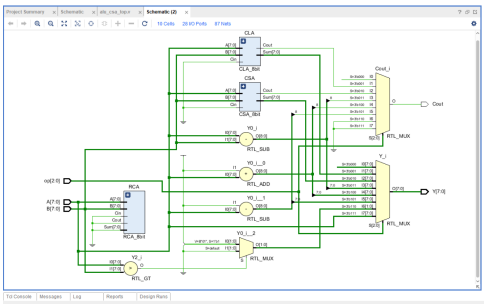

Fig. 1. ALU Schematic Diagram

I. Introduction

Arithmetic operations are essential in digital systems, and the Arithmetic Logic Unit (ALU) is a key component in processor design. The performance of an ALU relies heavily on the adder architecture chosen for arithmetic calculations. Different adder architectures provide different balances between hardware complexity and speed. This project focuses on designing and analyzing the performance of an 8-bit ALU implemented on FPGA. It emphasizes comparing multiple adder architectures under the same synthesis conditions.

II. ALU Architecture

The Arithmetic Logic Unit (ALU) in this project is an 8-bit circuit that carries out various arithmetic and logical operations on two 8-bit input values. The design is modular and adjustable, which allows for the integration and testing of different adder types. The ALU takes two 8-bit inputs, A and B, and produces an 8-bit result, Y, along with a carry-out signal, Cout.

The ALU features three adder types: Ripple Carry Adder (RCA), Carry Look-Ahead Adder (CLA), and Carry Select Adder (CSA). Each adder calculates the addition result for the same inputs. An operation select signal, op, determines which adder output or arithmetic operation is chosen at the ALU's output.

For addition, the ALU has three modes that correspond to the RCA, CLA, and CSA designs. It also supports other operations, including subtraction, increment, decrement, comparison, and pass-through. The ALU performs subtraction using two's complement arithmetic, while increment and decrement operations use simple addition and subtraction with fixed values.

To allow for clear performance testing, each adder type is contained in its own dedicated wrapper module. This keeps the synthesis and timing features of each adder independent, avoiding interference from any additional selection or control logic. The modular design enhances clarity, reusability, and scalability, making it easier to expand to larger bit widths or include more operations.

III. Adder Architectures

A. Ripple Carry Adder(RCA)

The Ripple Carry Adder passes the carry from the least significant bit to the most significant bit one step at a time. It has a straightforward design and low hardware complexity, but it experiences higher delays because the carry moves in a sequence.

$$S_i = A_i \oplus B_i \oplus C_i \tag{1}$$

$$C_{i+1} = A_i B_i + (A_i \oplus B_i) C_i \tag{2}$$

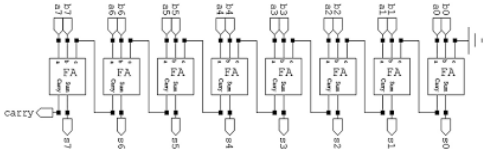

Fig. 2. Ripple Carry Adder

B. Carry Look-Ahead Adder(CLA)

The Carry Look-Ahead Adder speeds up calculations by computing carry signals ahead of time with generate and propagate logic. This cuts down on carry propagation delay, but it does increase the complexity of the logic.

$$G_i = A_i \cdot B_i \tag{3}$$

$$P_i = A_i \oplus B_i \tag{4}$$

$$C_{i+1} = G_i + P_i C_i \tag{5}$$

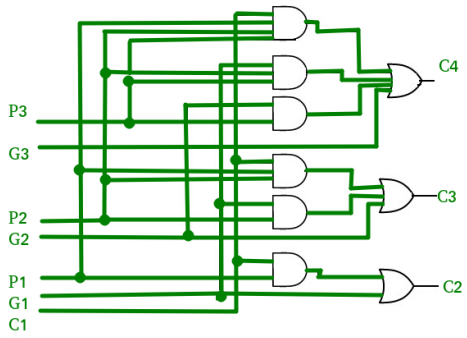

Fig. 3. Carry Look-Ahead Adder

C. Carry Select Adder(CSA)

The Carry Select Adder computes addition results in parallel for several carry-in values. It uses multiplexers to select the right result. This design balances speed and area cost.

$$S_i^{(0)} = A_i \oplus B_i \oplus 0, \quad C_{i+1}^{(0)} = (A_i \cdot B_i) + (A_i \oplus B_i) \cdot 0 \tag{6}$$

$$S_i^{(1)} = A_i \oplus B_i \oplus 1, \quad C_{i+1}^{(1)} = (A_i \cdot B_i) + (A_i \oplus B_i) \cdot 1 \tag{7}$$

$$S_i = \begin{cases} S_i^{(0)}, & \text{if } C_{\text{in}} = 0 \\ S_i^{(1)}, & \text{if } C_{\text{in}} = 1 \end{cases} \tag{8}$$

$$C_{i+1} = \begin{cases} C_{i+1}^{(0)}, & \text{if } C_{\text{in}} = 0 \\ C_{i+1}^{(1)}, & \text{if } C_{\text{in}} = 1 \end{cases} \tag{9}$$

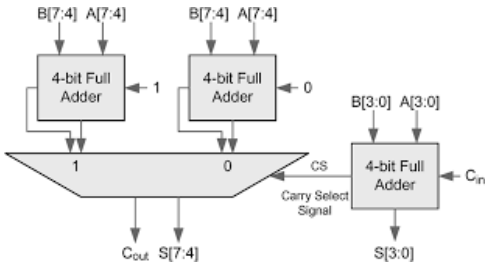

Fig. 4. Carry Select Adder

IV. Methodology

To ensure a fair area–delay comparison, three wrapper-based ALU configurations were synthesized. Each configuration instantiated only one adder architecture (RCA, CLA, or CSA), while all other datapath logic and synthesis constraints remained identical. The designs were synthesized using Xilinx Vivado targeting the same FPGA device. Area was evaluated using Look-Up Table (LUT) utilization, and delay was measured using the critical path delay reported in the timing summary.

V. Implementation Details

- HDL Language: Verilog
- Design Type: Fully combinational
- Tool: Xilinx Vivado
- Target Device: Xilinx FPGA
- Comparison Setup: Wrapper-based synthesis

VI. Area & Critical Path Delay & Power Analysis

TABLE I
Compare of adders using LUT Count, CP Delay & Power

Adder Architecture	LUT Count	CP Delay (ns)	Power (W)
RCA	48	11.202	5.288
CLA	49	10.873	5.288
CSA	4	10.169	0.554

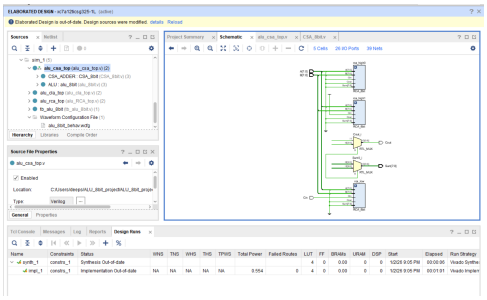

Fig. 5. Carry Select Adder Schematic & LUT Analysis

VII. Discussion

The 8-bit ALU was built using three types of adders: Ripple Carry Adder (RCA), Carry Look-Ahead Adder (CLA), and Carry Select Adder (CSA). It was assessed based on area (LUTs), critical path delay (ns), and power consumption (W). These metrics help to understand the trade-offs between speed, resource use, and energy efficiency in FPGA-based digital design.

A. Area (LUTs) Comparison

The synthesis results show that RCA uses 48 LUTs, CLA uses 49 LUTs, and CSA needs only 4 LUTs. RCA and CLA consume a similar number of LUTs because both use combinational logic for carry propagation, with CLA slightly higher due to extra logic for generating carries in parallel. CSA is highly optimized for this 8-bit design. Its pre-computation and selection method greatly cuts down on the combinational logic needed, leading to a large decrease in LUT usage.

B. Critical Path Delay

The critical path delay analysis shows that CSA is the fastest, with a delay of 10.169 ns. CLA follows at 10.873 ns, and RCA takes 11.202 ns. RCA has the highest delay because its carry has to propagate one bit at a time through all 8 bits, which increases the path length. CLA improves this by generating carry signals at the same time, slightly reducing the propagation time. CSA has the shortest delay by precomputing sums for both possible carry-in conditions and selecting the right result. This reduces the carry propagation in the critical path.

C. Power Consumption

Power analysis shows a big difference: RCA and CLA each consume 5.288 W, while CSA uses only 0.554 W. The high power of RCA and CLA comes from more switching activity during sequential carry propagation and more combinational logic. CSA, with its lower LUT usage and precomputed paths, cuts down on both switching and static power, making it very energy-efficient.

VIII. ALU Simulation Details

Simulation of the 8-bit ALU was conducted with a Verilog testbench to confirm that it works correctly with all supported operations and adder designs. The ALU was tested using three adder types: Ripple Carry Adder (RCA), Carry Look-Ahead Adder (CLA), and Carry Select Adder (CSA). It also included various arithmetic operations such as subtraction, increment, decrement, comparison, and pass-through.

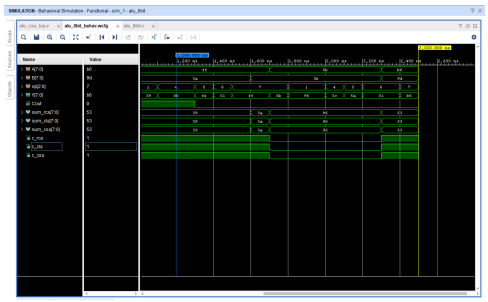

Fig. 6. ALU Simulation

IX. Limitations

- Design is limited to an 8-bit ALU. Scalability to higher bit widths, like 16-bit and 32-bit, is not examined.
- The ALU is purely combinational. There is no pipelining or sequential logic, and throughput improvement is not investigated.
- Results come from a single FPGA device and synthesis setup. Performance may differ on other devices.
- Low-power techniques, such as clock gating and operand isolation, are not used.
- The study focuses only on basic arithmetic and logic operations. More complex functions are not included.

X. Future Work

- Extend the ALU design to higher bit-widths, such as 16-bit and 32-bit, for wider applications.
- Implement pipelining to improve throughput and clocked operation performance.
- Explore hybrid adder architectures that combine RCA, CLA, and CSA for better area-delay-power trade-offs.
- Integrate low-power techniques like clock gating, operand isolation, or dynamic voltage scaling to lower power consumption.
- Add more arithmetic and logical operations, including multiplication, division, and bitwise shifts.
- Compare performance across different FPGA families to generalize results.

XI. Conclusion

An 8-bit ALU using RCA, CLA, and CSA adders was designed, simulated, and implemented on an FPGA. The performance analysis shows clear differences among the architectures. RCA and CLA have similar area utilization, with 48 and 49 LUTs, and critical path delays of 11.202 ns and 10.873 ns, respectively. In contrast, CSA demonstrates significant improvements, using only 4 LUTs and achieving a reduced delay of 10.169 ns.

Regarding power consumption, both RCA and CLA consume 5.288 W each, while CSA only consumes 0.554 W, which is about an 89.5% reduction in power. CSA also offers approximately a 9% lower critical path delay compared to RCA, emphasizing its efficiency in speed, area, and power.

Overall, CSA is the best choice for small-bit FPGA ALUs, as it provides high performance and low resource use. RCA and CLA, though simpler and more traditional, show higher area and power usage with slightly longer delays. This study highlights the need for careful adder selection based on FPGA resources and performance needs, and it lays the groundwork for scaling up to larger bit-width ALUs or pipelined designs in the future.

Acknowledgments

I want to sincerely thank my faculty mentor and the Department of Electronics and Communication Engineering for their ongoing guidance and support during this project. I appreciate the help I received in grasping FPGA design concepts and Verilog HDL implementation.

References

[1] S. M. Azhaan Mashir, S. K. Jha, M. Dixit and A. Maurya, *Design and Implementation of ALU: A Comparative Study of Full Adder and MUX-Based Architectures.* Boca Raton, FL, USA, Taylor & Francis Group, 2025.

[2] Tiwari, K.S., Kadam, R.S., Dudhedia, M.A., Pansare, J.R., Khedkar, S.P., Gawande, *S.H.: Reversible logic gates and applications—a low power solution to VLSI chips.* Math. Model. Eng. Probl. 11(3) (2024)

[3] H. Singaravelan and S. Ravi,*64bit Hybrid Adder for ALU Design Applications.* Vellore, India: Int. J. Innov. Technol. Explor. Eng., vol. 9, no. 8, pp. 694–698, Jun. 2020

[4] Shubham Sarkar, Sujan Sarkar, Jishan Mehedi, *Comparison of Various Adders and their VLSI Implementation*, IEEE International Conference on Computer Communication and Informatics (ICCCI), 2018

[5] Prashant Gurjar, Rashmi Solanki, Pooja Kansliwal and Mahendra Vucha, *VLSI Implementation of Adders for High Speed ALU. New York, USA: International Journal of Computer Applications*, vol. 29, no. 10, pp. 11–15, Sep. 2011.

[6] Anjali Arora and Vandana Niranjan, *A new 16-bit high speed and variable stage carry skip adder*, 3rd IEEE International Conference on "Computational Intelligence and Communication Technology" (IEEE-CICT 2017),

[7] Morgenshtein, A. Fish, I. A. Wanger, *Gate-Diffusion Input (GDI): A power efficient method for digital combinational circuits*, IEEE Trans. on Very Large Scale Integration (VLSI), vol. 10, No. 5, pp. 566-581, 2002.

[8] Vijaylaxmi C Kalal , Ravikumar K. and Chaitrali V. Pawar,*Novel Low Power Logic Gates using Sleepy Techniques*, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering Vol. 4, No. 1, pp 165 -72,January 2015.