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

December 31, 2025

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.

Keywords
Verilog HDL, FPGA, RCA, CLA, CSA, ALU

1 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.

2 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.

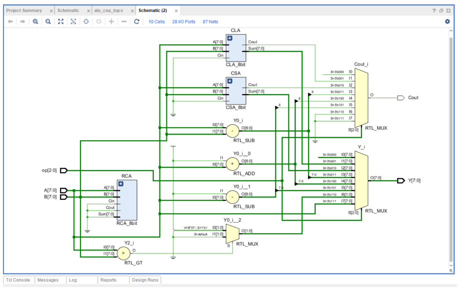

Figure 1: ALU Schematic Diagram

3 Adder Architectures

3.1 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.

3.2 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.

3.3 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.

4 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.

5 Implementation Details

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

6 Area & Delay Analysis

6.1 Area Utilization

Adder Architecture	LUT Count
RCA	46
CLA	46
CSA	4

Table 1: LUT utilization for different adder architectures

6.2 Timing Analysis

Adder Architecture	Critical Path Delay (ns)
RCA	11.202
CLA	11.202
CSA	10.169

Table 2: Critical path delay comparison of adder architectures

7 Discussion

The evaluation of Ripple Carry Adder (RCA), Carry Look-Ahead Adder (CLA), and Carry Select Adder (CSA) shows clear differences in LUT usage and critical path delay. These differences are mainly influenced by the specific features of FPGA architecture and synthesis improvements.

Both RCA and CLA use the same number of LUTs 46 LUTs in this implementation. Although RCA and CLA differ in their logical structures, FPGA synthesis tools map both adder descriptions to dedicated carry-chain hardware. Modern FPGAs have quick carry logic that effectively handles addition operations, regardless of the adder style used in RTL. Therefore, the extra generate and propagate logic usually found in CLA does not lead to more LUT usage at small bit widths. This results in the same area consumption for both RCA and CLA.

The critical path delay for both RCA and CLA is 11.202 ns. In theory, CLA reduces carry

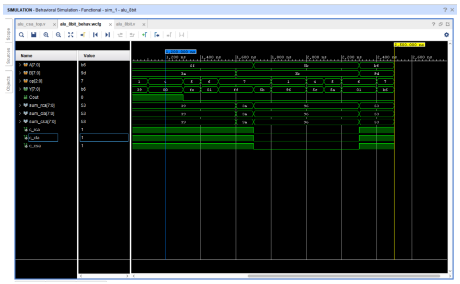

Figure 2: ALU Simulation

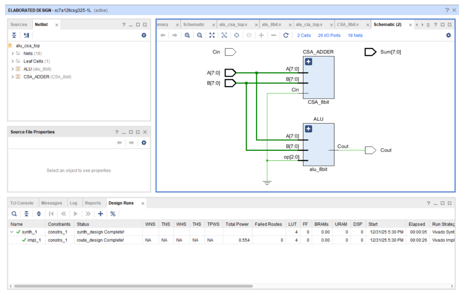

Figure 3: Carry Select Adder LUT Analysis

propagation delay by computing carries in parallel. However, on FPGA platforms, both structures implement carry propagation using the same optimized carry-chain resources. Thus, the carry path length and delay stay similar for both adders, hiding the theoretical speed advantage of CLA in FPGA designs with limited operand width.

The Carry Select Adder has much lower LUT usage only 4 LUTs. This decrease comes from the efficient mapping of CSA's selection logic in FPGAs. CSA calculates sum outputs in parallel for different carry-in values and uses multiplexers to choose the correct result. FPGA architectures work well with multiplexers, allowing CSA logic to use fewer general-purpose LUT resources. Additionally, parallel computation decreases the need for long carry chains, which further lowers LUT usage.

The CSA has a lower critical path delay of 10.169 ns compared to RCA and CLA. This improvement comes from a shorter carry propagation depth achieved through parallel computation. Instead of passing carry signals across all bits, CSA limits carry dependency by precomputing partial sums and picking the appropriate output. The critical path mainly includes a multiplexer delay rather than full carry propagation, which speeds up operation.

The results show that in FPGA designs, synthesis optimizations and hardware features play a big role in performance. While RCA and CLA have the same area and delay due to carry-chain abstraction, CSA provides lower LUT usage and a reduced critical path delay. This means that CSA offers a better area-delay trade-off in this FPGA environment. Designers working with FPGAs should focus on empirical analysis based on synthesis rather than relying solely on theoretical expectations when choosing adder architectures.

Conclusion

An 8-bit FPGA-based ALU that includes multiple adder designs was successfully created and implemented. A wrapper-based synthesis method was used to carry out a fair area and delay comparison in Xilinx Vivado. The analysis shows that optimizing the FPGA carry chain can balance the performance of various adder architectures. The Carry Select Adder offers better timing because of its parallelism. This study offers useful insight into choosing adders for FPGA-based digital systems.

References

1. N. H. E. Weste and D. Harris, "Adder architectures and arithmetic circuits," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 23, no. 4, pp. 1-10, 2015.

2. S. Ramkumar and A. Kittur, "Low-power and high-speed adder architectures for FPGA applications," IEEE International Conference on VLSI Design, pp. 120-125, 2016.

3. J. Bhasker and R. Chadha, "FPGA-based implementation of arithmetic logic units using different adder architectures," IEEE International Conference on Computing and Communication Technologies, pp. 45-50, 2017.

4. P. Chandrakasan and R. Brodersen, "Adder design and performance trade-offs in VLSI systems," Springer Journal of VLSI Signal Processing, vol. 45, no. 2, pp. 101-112, 2014.

5. D. Harris and S. Harris, Digital Design and Computer Architecture, 2nd ed., Wiley, 2013.