Advance Computer Architecture Project

Implement Different Types of 64 bit Adders

Submitted To:
Prof. Kanchan Manna

Submitted By:
Tushar Ganesh Bhavsar (2024H1030116G)
Shreya Dokania (2024H1030124G)
Ayush Agarwal(2024H1030125G)

Birla Institute of Technology and Science, Pilani
K.K. Birla Goa Campus

Contents

Abstract

Efficient arithmetic operations are central to high-performance digital systems, with binary addition being a foundational component. This project presents the design, implementation, and analysis of four distinct adder architectures—Ripple Carry Adder (RCA), Carry Look-Ahead Adder (CLA), Carry Select Adder (CSA), and Carry Select Adder with Unequal Bit Groups (CSAU)—using Verilog. Each adder is architected with a modular design to enhance scalability and simplify testing. The RCA is realized by chaining multiple 4-bit adders built from 1-bit full adders, with the initial bit handled by a half adder. The CLA employs a hierarchical propagate-generate logic structure to reduce carry propagation delay. The CSA is constructed using duplicate adder blocks with multiplexers to select results based on carry-in, and the CSAU enhances this by using varying bit-width groups to further optimize performance. Simulation results validate the functionality of each design across diverse input conditions, providing a foundation for deeper performance comparisons in terms of latency, resource usage, and power consumption.

1 Introduction

Binary adders are critical building blocks in digital systems, particularly within arithmetic logic units (ALUs), where performance metrics such as speed, area, and power consumption are of paramount importance. This project aims to explore, implement, and evaluate four representative adder architectures to understand their trade-offs and suitability for different computational requirements.

The Ripple Carry Adder (RCA), known for its simplicity, serves as a baseline for comparison. It consists of cascaded 4-bit adders, each built from 1-bit full adders, with the first stage utilizing a half adder to eliminate the need for an external carry-in. In contrast, the Carry Look-Ahead Adder (CLA) enhances computational speed by using two-level propagate-generate logic, which significantly reduces carry propagation delay across blocks.

To address the latency bottleneck further, the Carry Select Adder (CSA) uses dual 4-bit adder blocks—one assuming a carry-in of 0 and another assuming 1—with a multiplexer selecting the appropriate output. The Carry Select Adder with Unequal Bit Groups (CSAU) refines this approach by dividing the input into variable-width blocks, aiming for a more balanced trade-off between speed and area.

These architectures are implemented using Verilog and tested under a range of input scenarios, including boundary conditions, to ensure functional correctness. The project emphasizes modular construction, thorough simulation, and comparative evaluation, laying the groundwork for further optimization and potential hardware deployment.

2 Adder Architecture Overview

This section describes the Verilog implementation of four different adder architectures: Ripple Carry Adder (RCA), Carry Look-Ahead Adder (CLA), Carry Select Adder (CSA), and Carry Select Adder with Unequal Bit Groups (CSAU). Each is modularly designed for flexibility, scalability, and testability.

2.1 Ripple Carry Adder (RCA)

The Ripple Carry Adder is implemented using sixteen 4-bit adders, where each 4-bit unit is composed of four 1-bit full adders. The first bit uses a half adder since the carry-in is not required externally. Carries are propagated sequentially from one stage to the next, resulting in a straightforward, albeit slower, architecture.

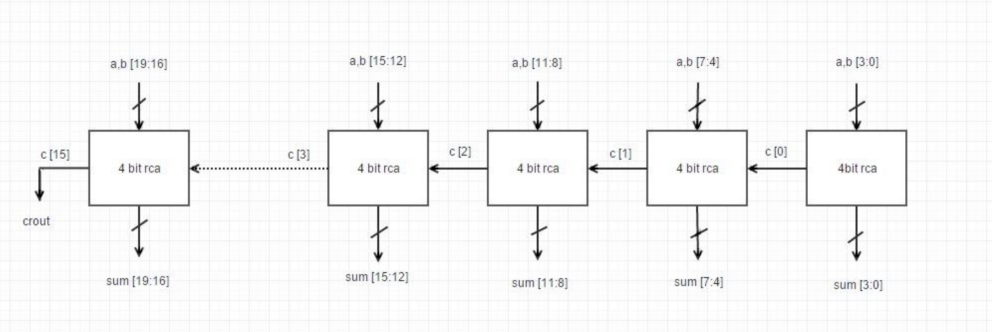


Figure 1: Ripple Carry Adder (RCA) Implementation

2.2 Carry Look-Ahead Adder (CLA)

To improve delay performance, the Carry Look-Ahead Adder utilizes two levels of propagate-generate (PG) logic. Each 16-bit adder is divided into 4-bit blocks, where local carries are computed in parallel using PG logic. The second-level PG generator aggregates and resolves carries from multiple blocks, enabling faster carry generation and reducing the dependency chain.

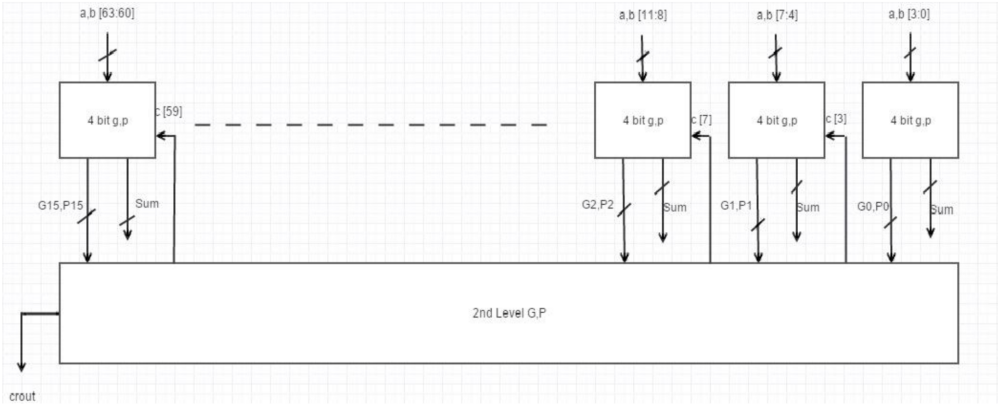
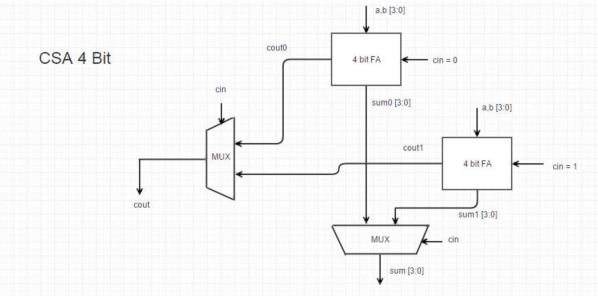


Figure 2: Carry Look-Ahead Adder (CLA) Implementation

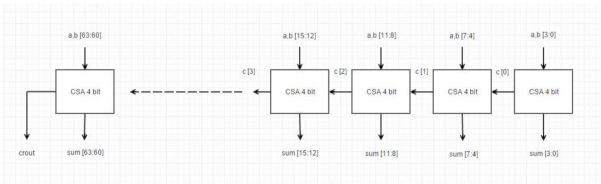
2.3 Carry Select Adder (CSA)

The Carry Select Adder is constructed using sixteen 4-bit ripple carry adders. Each block has two adder paths: one assumes carry-in as 0, and the other assumes carry-in as 1. A multiplexer follows each block to choose the correct result based on the actual carry-in, which

becomes the select line for the mux. The selected carry-out then drives the next 4-bit adder block.



(a) CSA Block Diagram - Part 1

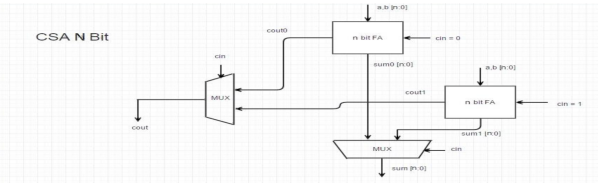


(b) CSA Block Diagram - Part 2

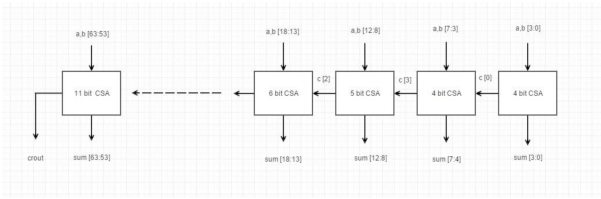
Figure 3: Carry Select Adder (CSA) Implementation

2.4 Carry Select Adder with Unequal Bit Groups (CSAU)

This enhanced version of the CSA is optimized by using groups of varying bit widths: 4-4-5-6-7-8-9-10-11. Smaller groups are used in the earlier stages to reduce the effect of carry propagation delay, while larger groups later on improve overall performance. This adaptive segmentation offers a more efficient trade-off between speed and complexity.



(a) CSAU Block Diagram - Part 1



(b) CSAU Block Diagram - Part 2

Figure 4: Carry Select Adder with Unequal Bit Groups (CSAU) Implementation

3 Implementation Details

All the resources are available at the following link: `Google Drive Folder (Click to View)`

To verify the correctness and robustness of the implemented adder architectures, six comprehensive test cases were designed. These test benches provide a variety of input conditions, covering normal operations, edge cases, and transitional scenarios involving reset conditions. Each test case uses a unique pair of hexadecimal inputs and validates the resulting sum across all four adder implementations: Ripple Carry Adder (RCA), Carry Look-Ahead Adder (CLA), Carry Select Adder with Equal Groups (CSA-EQG), and Carry Select Adder with Unequal Groups (CSA-UEQG).

The functionality of the reset signal was verified by introducing reset transitions during the input transitions in simulation. The goal was to ensure that each adder architecture responds correctly to reset conditions and resumes proper operation after reset is deasserted.

A summary of the selected test inputs and the expected output results is provided in Table **??**. The input combinations were chosen to test critical aspects of each adder, including:

- All-zero input for confirming proper gate behavior and output stability.
- Inputs designed to trigger maximum ripple propagation to evaluate delay in RCA.
- Random combinations to ensure the correctness of propagate-generate logic in CLA.
- Inputs for testing correctness and mux selection logic in CSA implementations.

Table 1: Six Selected Test Data

Test Case	op1 (hex)	op2 (hex)	RCA	CLA-2L	CSA-EQG	CSA-UEQG
1	ffff_ffff	0000_0000	0000_0000	0000_0000	0000_0000	0000_0000
	ffff_ffff	0000_0001	0000_0000	0000_0000	0000_0000	0000_0000
2	14ab_78ef	8dfd_eded	a2a9_66dd	a2a9_66dd	a2a9_66dd	a2a9_66dd
	d853_5c7d	cbed_8243	a440_dec0	a440_dec0	a440_dec0	a440_dec0
3	224f_9910	28ad_bc24	4afd_5534	4afd_5534	4afd_5534	4afd_5534
	23fa_39c6	bd8f_18c0	e18a_18c0	e18a_18c0	e18a_18c0	e18a_18c0
4	bd82_a234	66ff_abba	2482_4dee	2482_4dee	2482_4dee	2482_4dee
	b339_315c	239f_abcd	d6d8_dd29	d6d8_dd29	d6d8_dd29	d6d8_dd29
5	d2a9_b3c7	8822_2333	5acb_d6fa	5acb_d6fa	5acb_d6fa	5acb_d6fa
	72fa_7444	5575_5892	c86f_ccd6	c86f_ccd6	c86f_ccd6	c86f_ccd6
6	58ab_68fa	512c_438b	a9d7_ac82	a9d7_ac82	a9d7_ac82	a9d7_ac82
	259a_3333	4bba_2fab	7154_62de	7154_62de	7154_62de	7154_62de

4 Simulation and GTKWave Results

Each test case was simulated using the GTKWave waveform viewer. Screenshots were taken for every adder across all six test cases, totaling 24 visual outputs. These waveforms confirm that the sums produced by each architecture match the expected results and that the carry logic behaves as intended.

Below are the waveform results for each adder. Each figure includes subfigures representing each of the six test cases.

All GTKWave screenshots can also be accessed here: `Google Drive Folder (Click to View)`

4.1 Approximated Delay Based on Theory and Measured Delays

This section compares the theoretical delay calculations with the actual measured delay from synthesis for each of the four adder architectures.

4.1.1 Ripple Carry Adder (RCA)

For RCA, the theoretical delay is the sum of delays from all 63 carry ripples and the delay of calculating the last sum.

Approximated Delay:

$$t_{\text{add}} = t_{\text{setup}} + (n-1) \cdot t_{\text{carry}} + t_{\text{last\_sum}}$$

$$t_{\text{add}} = 0.31 + (64-1)(0.08 + 0.09) + (0.08 + 0.64 + 0.15 + 0.33) = 12.22\,\text{ns}$$

Measured Delay from Synthesis:

$$t_{\text{add}} = 14.08\,\text{ns}$$

4.1.2 Carry Look-Ahead Adder (CLA)

For a 64-bit 2-level CLA, the delay is approximated as 8 times the delay of a single generate/propagate operation.

Approximated Delay:

$$t_{\text{add}} = 8 \cdot T_g$$

$$t_{\text{add}} = 8 \cdot (0.11 + 0.18 + 0.08 + 0.10) = 3.76\,\text{ns}$$

Measured Delay from Synthesis:

$$t_{\text{add}} = 2.88\,\text{ns}$$

4.1.3 Carry Select Adder - Equal Groups (CSA-EQG)

For CSA-EQG, the delay includes 4-bit ripple delays, 16 mux delays, and delay for calculating the last sum.

Approximated Delay:

$$t_{\text{add}} = t_{\text{setup}} + 4 \cdot t_{\text{carry}} + \left(\frac{n}{4}\right) \cdot t_{\text{mux}} + t_{\text{extra\_last\_sum}}$$

$$t_{\text{add}} = 0.34 + 4(0.09 + 0.07) + 16(0.25) + 0.37 = 5.35\,\text{ns}$$

Measured Delay from Synthesis:

$$t_{\text{add}} = 5.37\,\text{ns}$$

4.1.4 Carry Select Adder - Unequal Groups (CSA-UEQG)

For CSA-UEQG, the delay is estimated from the first block's ripple delay, 8 mux delays, and the final sum delay.

Approximated Delay:

$$t_{\text{add}} = t_{\text{setup}} + t_{\text{carry}} + 8 \cdot t_{\text{mux}} + t_{\text{extra\_last\_sum}}$$

$$t_{\text{add}} = 0.34 + 0.17 + 8(0.27) + 0.31 = 2.98\,\text{ns}$$

Measured Delay from Synthesis:

$$t_{\text{add}} = 3.96\,\text{ns}$$

5 Performance Evaluation

This section evaluates the performance of the four adder architectures — Ripple Carry Adder (RCA), Carry Look-Ahead Adder (CLA-2L), Carry Select Adder with Equal Groups (CSA-EQG), and Carry Select Adder with Unequal Groups (CSA-UEQG). The evaluation is based on synthesis and timing analysis results obtained using industry-standard open-source tools.

The synthesis of each adder design was performed using **Yosys**, which generated the gate-level netlist and reported area and power consumption. The synthesized netlist was then passed to **OpenSTA** for static timing analysis, where timing slack and minimum clock period (latency) were evaluated.

The performance parameters considered include:

- **Latency (Clock Period)** – the minimum achievable clock period as determined by the timing analysis.
- **Time Slack** – the margin by which timing constraints are met or violated; positive slack indicates successful timing closure.
- **Power Consumption** – estimated during synthesis to assess energy efficiency.

The results for each architecture are presented in Tables IV.1 through IV.4. Each table lists multiple synthesis trials with varying timing constraints. The final row in each table typically represents the most optimized configuration, providing the best trade-off between timing and power. These optimized configurations were determined based on achieving zero or positive slack while minimizing power consumption.

Table IV.1: Synthesis Constraints and Results for Ripple Carry Adder (RCA)

Trial #	Clock (ns)	Time Slack (ns)	Power (mW)
1	10.00	-4.65	0.176
2	12.00	-2.65	0.150
3	14.00	-0.65	0.130
4	14.65	0.00	0.125
5	15.00	0.35	0.123

Table IV.2: Synthesis Constraints and Results for Carry Look-Ahead Adder (CLA-2L)

Trial #	Clock (ns)	Time Slack (ns)	Power (mW)
1	10.00	6.51	0.200
2	4.00	0.51	0.468
3	3.00	-0.49	0.617
4	3.50	0.01	0.532

Table IV.3: Synthesis Constraints and Results for Carry Select Adder - Equal Groups (CSA-EQG)

Trial #	Clock (ns)	Time Slack (ns)	Power (mW)
1	10.00	4.01	0.245
2	4.00	-1.99	0.58
3	6.00	0.01	0.394

Table IV.4: Synthesis Constraints and Results for Carry Select Adder - Unequal Groups (CSA-UEQG)

Trial #	Clock (ns)	Time Slack (ns)	Power (mW)
1	10.00	5.42	0.245
2	5.00	0.42	0.468
3	4.00	-0.58	0.579
4	4.60	0.02	0.507
5	4.59	0.01	0.508

6 Conclusion

This report provided a comprehensive analysis of four 64-bit adder architectures—Ripple Carry Adder (RCA), Carry Look-Ahead Adder (CLA-2L), Carry Select Adder with Equal Groups (CSA-EQG), and Carry Select Adder with Unequal Groups (CSA-UEQG). The evaluation was performed using Yosys for logic synthesis and OpenSTA for timing analysis. The designs were compared based on key performance metrics including clock period (latency), time slack, and power consumption. From the synthesis results, it was observed that each architecture offers unique trade-offs: RCA is simpler but slower, CLA offers lower latency at higher power, and CSA variants balance speed and power differently. The most optimized configurations were identified based on a combination of maximum positive slack and minimal power usage. These insights help in selecting an appropriate adder architecture depending on the performance and efficiency requirements of a given application.