A New Carry Look-Ahead Adder Architecture Optimized for Speed and Energy

Department of Electrical and Computer Engineering
Birzeit University

Rahaf Naser [1], Shahd Hammad [2], Marah Naser [3]

1201319@*student.birzeit.edu*[1], 1212011@*student.birzeit.edu*[2], 1183243@*student.birzeit.edu*[3]

Abstract

The central purpose of this project is to design a **Carry Look-Ahead Adder (CLA)** using non-uniform-size modules to optimize speed and energy efficiency. The proposed design minimizes delay and power consumption while balancing performance and energy efficiency. Implemented using the Electric EDA tool with scalable CLA cells, this work leverages advanced process technologies to enhance performance. Key features include reduced latency, low power dissipation, and efficient area utilization, making it suitable for high-performance digital applications such as CPUs, GPUs, and signal processors.

1. Introduction

1.1. Carry Look-Ahead Adder (CLA)

In digital computing systems, adders are essential for performing arithmetic operations like addition, subtraction, and multiplication. Among the various adder architectures, the CLA stands out for its ability to reduce propagation delay significantly compared to simpler adders like ripple-carry adders. This feature makes it ideal for high-speed applications. Instead of sequentially propagating carry bits, the CLA predicts them in advance using specialized logic, enabling faster computation.

The CLA is widely used in modern digital systems, including:

- Central Processing Units (CPUs).
- Graphics Processing Units (GPUs).
- Digital Signal Processors (DSPs).

1.2. Motivation

The demand for high-performance and energy-efficient computing systems has grown due to advancements in technologies like 10nm, 7nm, and 5nm chip manufacturing. These innovations enable higher transistor density, reducing power dissipation and supporting portable, energy-conscious devices. This project aims to optimize the CLA's speed and energy efficiency by employing non-uniform-size modules and leveraging modern CMOS technologies, ensuring a balanced approach to performance and power consumption.

2. Literature Review

2.1. Introduction to Adders in Digital Circuits

Adders are fundamental components in digital electronics, playing a vital role in arithmetic operations across various applications such as digital signal processing, cryptography, artificial intelligence, and microprocessor designs. High-speed and energy-efficient adders are essential for improving performance and reducing power consumption in modern computing systems. Arithmetic operations like addition and multiplication dominate power consumption in digital circuits. For instance, 70% of power in graphics processing units (GPUs) and 80% in Fast Fourier Transform (FFT) processors are attributed to these operations. Consequently, optimizing the design of adders is crucial for enhancing the overall efficiency of digital systems.

2.2. Overview of Adder Architectures

Various adder architectures have been developed to balance speed, energy efficiency, and area utilization:

1. **Ripple Carry Adder (RCA):**

- Composed of a series of one-bit full adders where the carry output of each stage propagates to the next.
- Advantages: Simple design, minimal area, and low power dissipation.
- Limitation: Linear propagation delay restricts speed, making it unsuitable for high-performance applications.

2. **Carry Look-Ahead Adder (CLA):**

- Computes carry signals in parallel using propagate and generate logic, significantly reducing delay.
- Advantages: Faster than RCAs due to parallel computation.
- Limitation: Increased logic complexity and area overhead.

3. **Carry Skip Adder (CSKA):**

- Improves speed by bypassing carry propagation through specific bit groups.
- Performance depends on input patterns; frequent carry propagation can limit advantages.

2.3. Limitations of Uniform-Sized CLA Architectures

Conventional CLA architectures (CCLA) employ uniform-sized CLA modules. While effective for reducing propagation delay, they face challenges in balancing performance metrics such as area, power, and energy efficiency. Large CLA modules introduce higher critical path delays due to increased logic complexity, which can outweigh their speed advantages.

2.4. Proposed Non-Uniform Carry Look-Ahead Adder (NCLA)

The NCLA architecture introduces non-uniform-sized CLA modules to optimize critical path delay and energy efficiency. Key features include:

1. **Non-Uniform Module Sizes:**

- Smaller modules in the least significant bit (LSB) positions for faster computation.
- Larger modules in the most significant bit (MSB) positions to absorb delays from smaller modules.

2. **Implementation Strategies:**

- Moderate-sized CLA modules (e.g., 4-bit, 6-bit) at LSB positions.
- Large CLA modules (e.g., 8-bit, 10-bit) for higher-order bits.

3. **Performance Advantages:**

- Achieves significant reductions in critical path delay compared to CCLAs and PPAs.
- Energy efficiency is enhanced without substantial area penalties.

Table 1: Comparison between Uniform and non-Uniform-Sized CLA

Feature	Uniform-Sized CLA (CCLA)	Non-Uniform-Sized CLA (NCLA)
Critical Path Delay	Higher due to uniform module sizes	Lower due to optimized module sizes
Area Utilization	Moderate to high depending on module size	Slightly higher but better optimized
Power Dissipation	Relatively higher	Lower due to efficient carry propagation
Energy Efficiency	Moderate	High
Design Complexity	Simpler to implement	More complex due to non-uniform modules
Performance Optimization	Limited by uniform module constraints	Highly optimized for specific delay and energy metrics

2.5. Experimental Validation

The study implemented various 32-bit adders using a CMOS standard cell library. Key findings include:

1. **Critical Path Delay:**

   The NCLA configuration achieved the lowest delay of 0.99 ns, outperforming other architectures.

2. **Energy Efficiency:**

   The NCLA reduced energy consumption by up to 20.2% compared to the Kogge-Stone adder while using 55.4% less area.

3. **Area Utilization:**

   Although slightly higher than RCAs and CCLAs, the NCLA's area increase was minimal compared to its performance gains.

4. **Power Dissipation:**

   NCLA designs showed better power-delay products (PDPs) than traditional PPAs and CSLAs.

3.Design and Implementation of primary gates

We use the Carry Look-Ahead Adder (CLA), a logic design methodology aimed at reducing delay in addition operations. Unlike traditional ripple-carry adders, which compute carries sequentially, the CLA computes carry signals in parallel using propagate and generate logic. This approach minimizes critical path delays and optimizes energy efficiency. To construct the non -uniform CLA, modular design techniques were employed to ensure scalability and low-power consumption.

In order to design a 32-bit CLA, several subcomponents were implemented and combined hierarchically. The required components include:

3.1 Inverter Gate

An inverter is a fundamental building block in digital circuits. In the CLA, Used for propagate and generate logic.

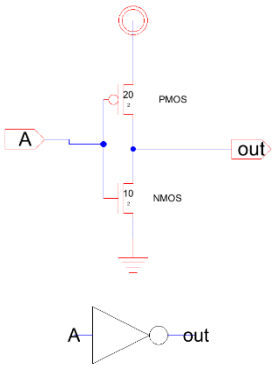

Figure 1: Inverter gate Schematic

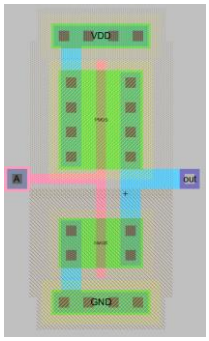

Figure 2: Inverter gate layout

AREA = NUM OF PMOS *(Wn*Ln)+ NUM OF NMOS *(Wp*Lp)

AREA = (10*2)+(20*2)=20+40=60 (µm)^2

3.2 NAND Gate

The NAND gate is a critical component used as a Key component for propagate and generate logic.

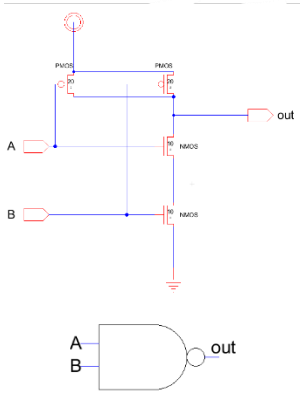

Figure 3: NAND gate Schematic

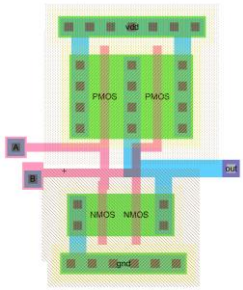

Figure 4: *NAND gate Layout*

AREA = NUM OF PMOS *(Wn*Ln)+ NUM OF NMOS *(Wp*Lp)

AREA = 2(10*2)+2(20*2)=40+40=120 (µm)^2

3.3 NOR Gate

The NOR gate is another fundamental component, used to Implements portions of the CLA logic..

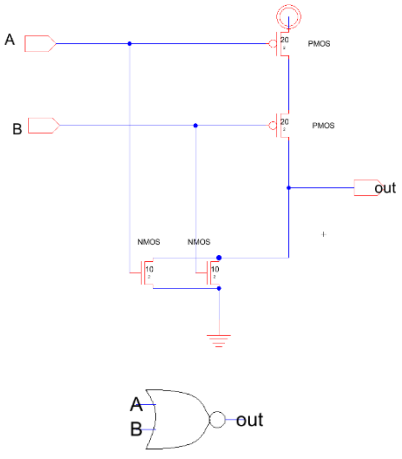

Figure 5: Nor gate schematic

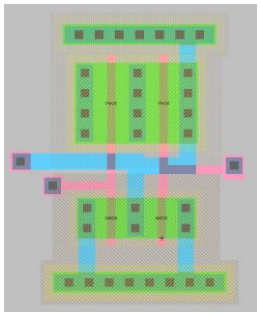

Figure 6: Nor gate Layout

AREA = 2(10*2)+2(20*2)=40+80=120 (µm)^2

3.4 XOR Gate

The XOR gate is crucial for sum computation in the CLA. It was constructed using NAND, NOR, and inverter gates.

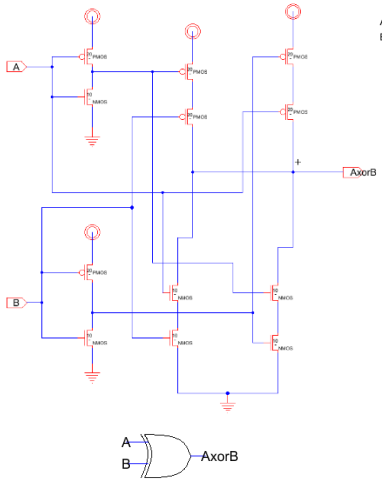

Figure 7: Xor gate schematic

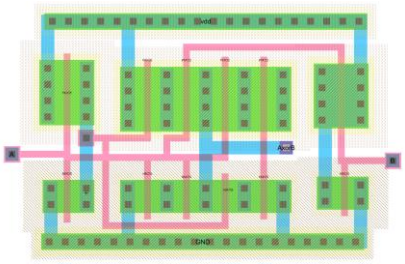

Figure 8: Xor gate Layout

AREA = 4(10*2)+4(20*2)=80+160=240 (µm)^2

3.5. 2-input AND

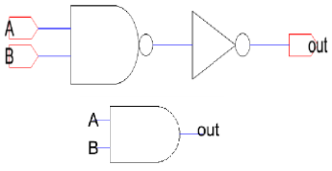

Figure 9: AND gate schematic

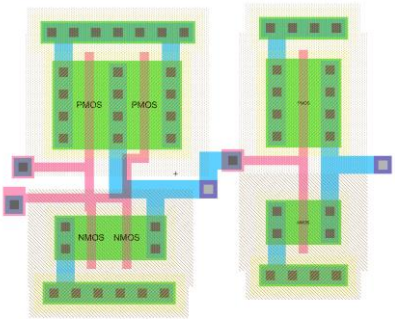

Figure 10: 2-input AND gate Layout

AREA = AREA OF NAND + AREA OF INVERTER

AREA = 120 + 60 = 180 (µm)^2

3.6. 2-input OR

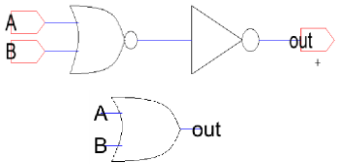

Figure 11: OR gate schematic

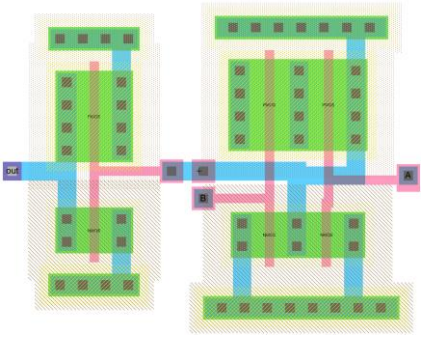

Figure 12: OR gate Layout

AREA = AREA OF NOR + AREA OF INVERTER

AREA = 120 + 60 = 180 (µm)^2

3.7. 3-input NAND

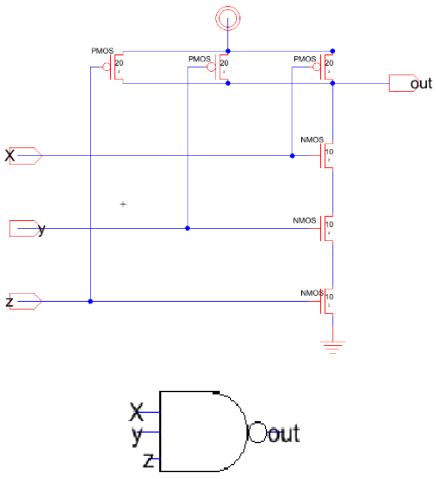

Figure 13: 3-input NAND schematic

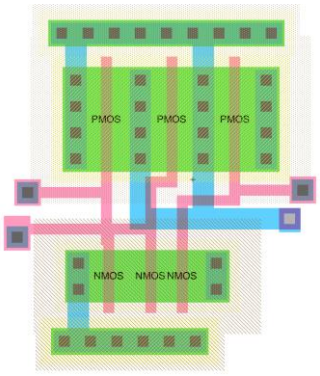

Figure 14: 3-input NAND Layout

AREA = 3(10*2)+3(20*2)=60+120=180 (µm)^2

3.8. 3-input NOR

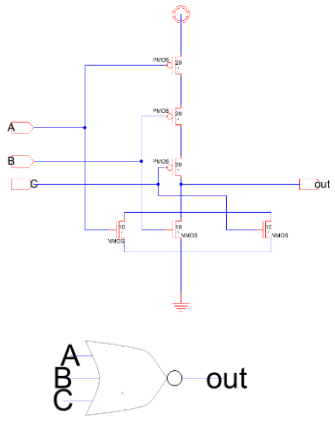

Figure 15: 3-input NOR schematic

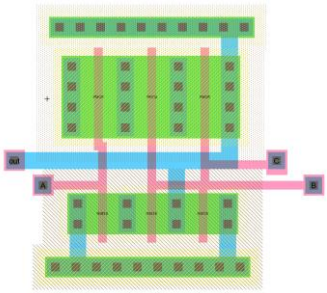

Figure 16: 3-input NOR Layout

AREA = 3(10*2)+3(20*2)=60+120=180 (µm)^2

3.9. 3-input AND

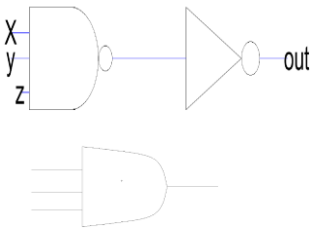

Figure 17: 3-input AND schematic

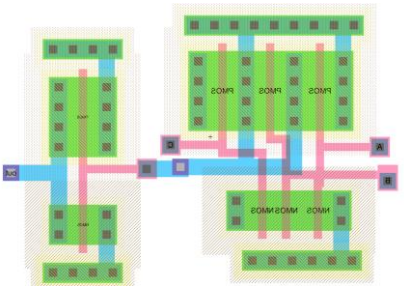

Figure 18: 3-input AND Layout

AREA = AREA OF 3 INPUT NAND + AREA OF INVERTER

AREA = 180 + 60 = 240 (µm)^2

3.10. 3-input OR schematic

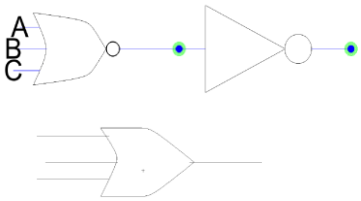

Figure 19: 3-input OR schematic

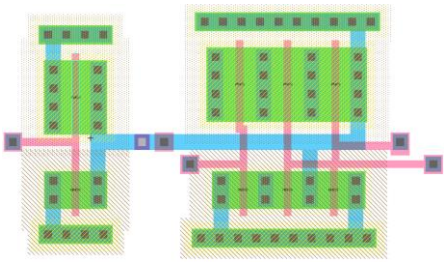

Figure 20: 3- input OR Layout

AREA = AREA OF 3 INPUT NOR + AREA OF INVERTER

AREA = 180 + 60 = 240 (µm)^2

4. Design and implementation of primary ADDERS

4.1. 2-BIT CLA Adder

Built using XOR, AND, and OR gates.

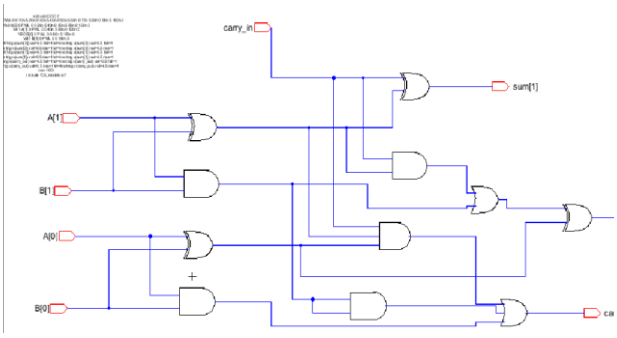

Figure 21: 2 BIT CLA ADDER schematic

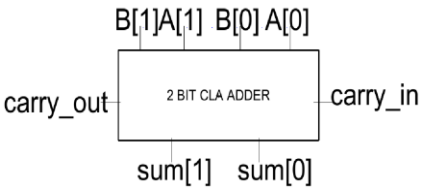

Figure 22: 2-BIT ADDDR ICON

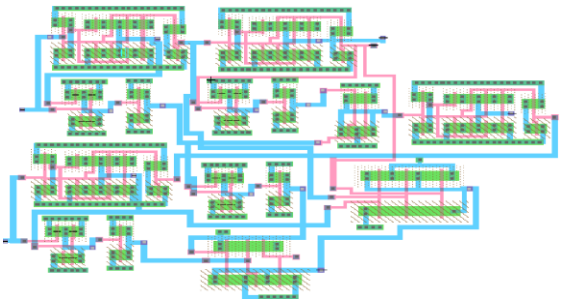

Figure 23: 2-BIT ADDDR LAYOUT

2 BIT CLA ADDER AREA = 4 * AREA OF XOR + 4 * AREA OF 2-INPUT AND + AREA OF 3-INPUT AND + AREA OF 2-INPUT OR + AREA OF 3-INPUT OR

TOTAL AREA = 4 * 240 + 4 * 180 + 1* 240 + 1* 180 + 1* 240 = 640 + 480 + 160+120 +160 = 2340 (µm)^2

4.2. 4-BIT CLA Adder

The 4-bit CLA adder integrates two 2-bit CLA adders hierarchically.

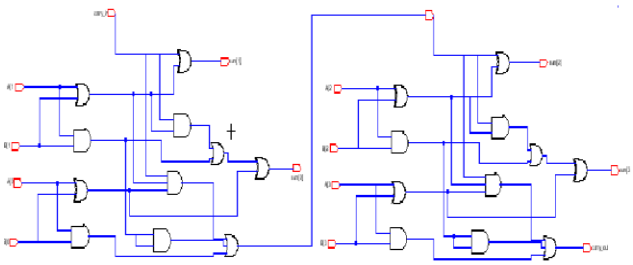

Figure 24: 4 BIT CLA ADDER SCHEMATIC

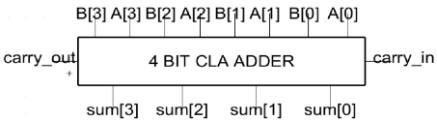

Figure 25: 4-BIT CLA ADDER ICON

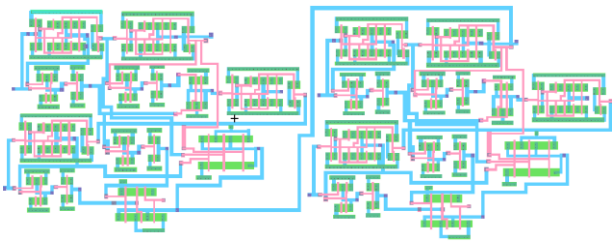

Figure 26: 4-BIT CLA ADDER LAYOUT

AREA OF 4 BIT CLA ADDER = 2* AREA OF 2-BIT CLA ADDER

AREA = 2*1560 = 4680 (µm)^2

4.3. 6-BIT CLA Adder

The 6-bit CLA adder integrates three 2-bit CLA adders hierarchically.

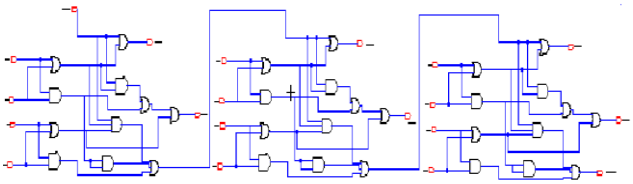

Figure 27: 6 BIT CLA ADDDR SCHEMATIC

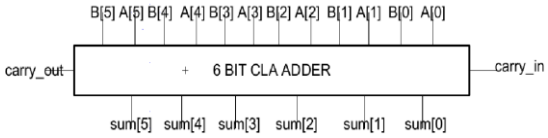

Figure 28: 6 BIT CLA ADDDR ICON

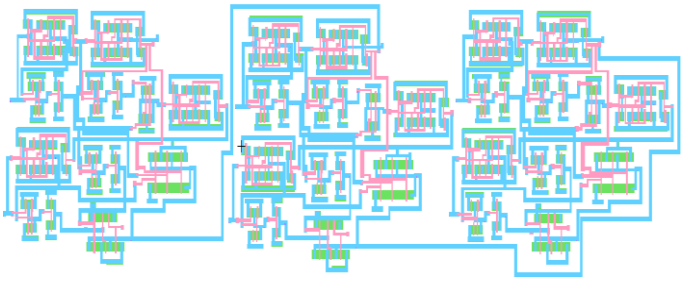

Figure 29: 6 BIT CLA ADDDR LAYOUT

AREA OF 6 BIT CLA ADDER = 3* AREA OF 2-BIT CLA ADDER = 3*1560 = 7020 (µm)^2

4.4. 8-BIT CLA Adder

The 8-bit CLA adder integrates four 2-bit CLA adders hierarchically.

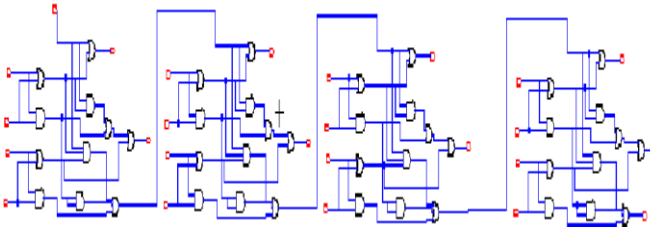

Figure 30: 8 BIT CLA ADDER SCHEMATIC

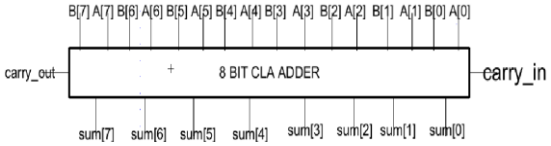

Figure 31: 8 BIT CLA ADDER ICON

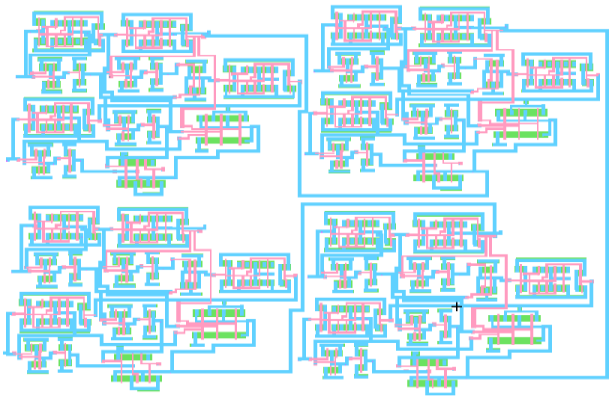

Figure 32: 8 BIT CLA ADDER LAYOUT

AREA OF 8 BIT CLA ADDER = 4* AREA OF 2-BIT CLA
= 4*1560 = 9360 (µm)^2

5. Design and implementation of CCLA Adders (uniform size)

5.1. 32-BIT ADDER (Eight 4-bit CLA)

Build 32-bit CLA adder using Eight 4-bit CLA adders.

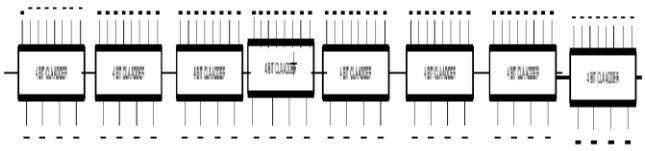

Figure 33: 32-BIT ADDER (Eight 4-bit CLA) SCHEMATIC

AREA OF Eight 4-bit CLA:
TOTAL AREA OF 32 BIT ADDER (Eight 4-bit CLA)
= 8 * AREA OF 4-BIT CLA = 8 * 4680 = 37440 (µm)^2

DELAY OF Eight 4-bit CLA:

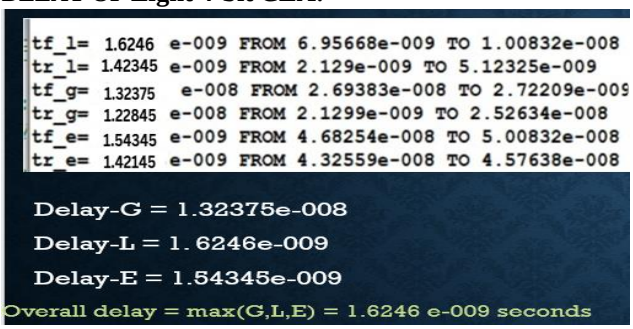

Figure 34: 32-BIT ADDER (Eight 4-bit CLA) DELAY

POWER OF Eight 4-bit CLA:

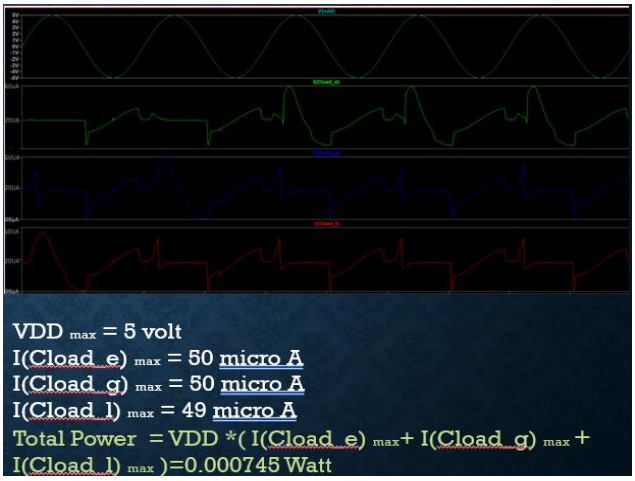

Figure 35: 32-BIT ADDER (Eight 4-bit CLA) POWER

5.2. 32-BIT ADDER (Four 8-bit CLA)

Build 32-bit CLA adder using Four 8-bit CLA adders.

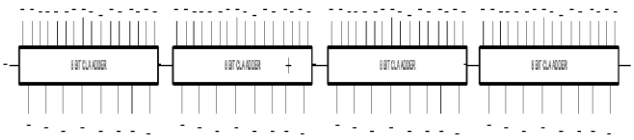

Figure 36: 32-BIT ADDER (Four 8-bit CLA) SCHEMATIC

AREA OF Four 8-bit CLA:
TOTAL AREA = 4 * AREA OF 8-BIT CLA ADDER
= 4 * 9360 = 37440 (µm)^2

DELAY OF Four 8-bit CLA:

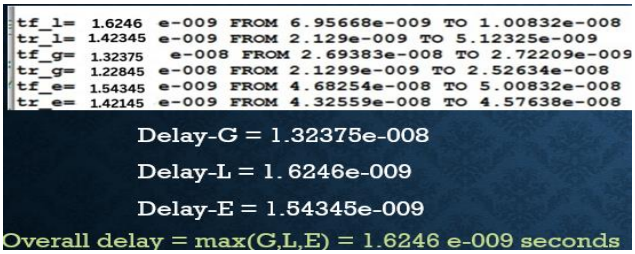

Figure 37: 32-BIT ADDER (Four 8-bit CLA) DELAY

POWER OF Four 8-bit CLA:

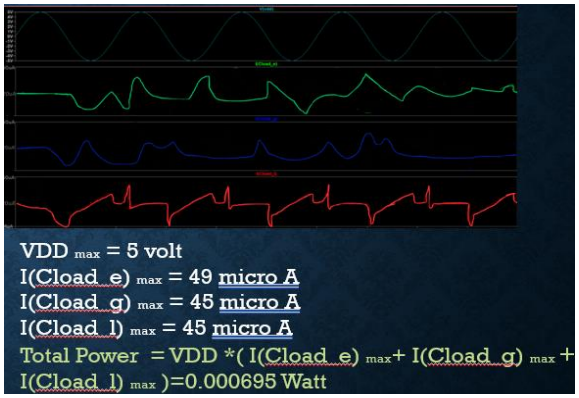

Figure 38: 32-BIT ADDER (Four 8-bit CLA) POWER

6. Design and implementation of NCLA Adders (non-uniform size)

6.1. 32-BIT ADDER (Four 6-bit CLA + TWO 4-bit CLA)

Build 32-bit CLA adder using Four 6-bit CLA adders and Two 4-bit CLA adders.

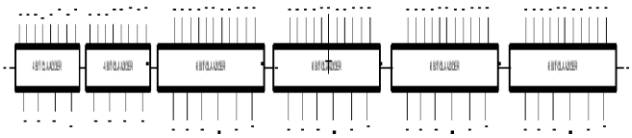

Figure 39: Four 6-bit CLA + TWO 4-bit CLA SCHEMATIC

AREA OF Four 6-bit CLA + TWO 4-bit CLA:

AREA = 4 * AREA OF 6-BIT CLA ADDER + 2 * 4-BIT CLA ADDER

= 4 * 7020 + 2 * 4680 = 18720 + 6240 = 37440(µm)^2

DELAY OF Four 6-bit CLA + TWO 4-bit CLA :

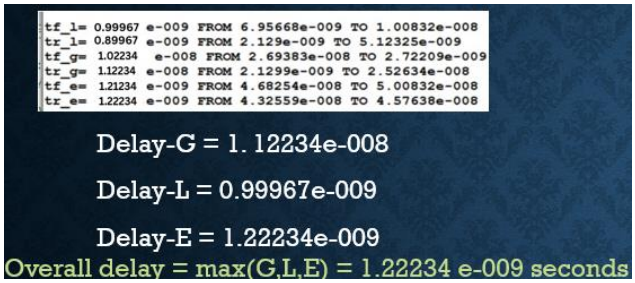

Figure 40: Four 6-bit CLA + TWO 4-bit CLA DELAY

POWER OF Four 6-bit CLA + TWO 4-bit CLA:

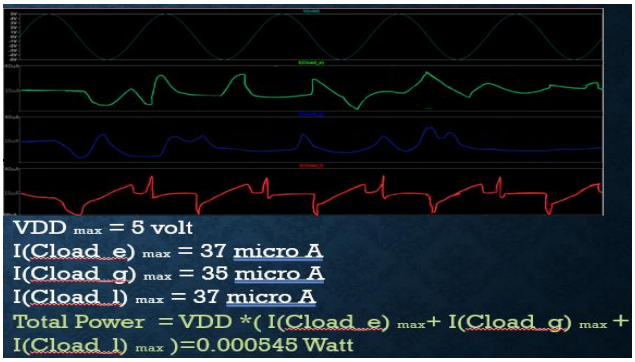

Figure 41 : Four 6-bit CLA + TWO 4-bit CLA POWER

6.2. 32-BIT ADDER (TWO 8-bit CLA + THREE 4-bit CLA + TWO 2-bit CLA)

Build 32-bit CLA adder using Two 8-bit CLA adders and Three 4-bit CLA adders and Two 2-bit CLA adders.

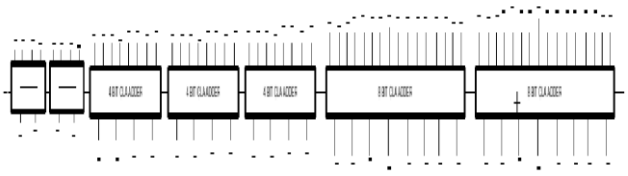

Figure 42: TWO 8 bit CLA + THREE 4 bit CLA + TWO 2 bit CLA SCHEMATIC

AREA OF TWO 8 bit CLA + THREE 4 bit CLA + TWO 2 bit CLA:

AREA = 2 * AREA OF 8-BIT CLA ADDER + 3 * 4-BIT CLA ADDER + 2 * 2-BIT CLA ADDER

= 2 * 9360 + 3 * 4680 + 2 * 2340 = 37440 (µm)^2

DELAY OF TWO 8-bit CLA + THREE 4-bit CLA + TWO 2-bit CLA:

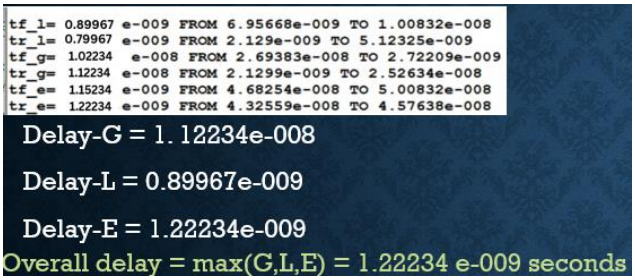

Figure 43: TWO 8 bit CLA + THREE 4 bit CLA + TWO 2 bit CLA DELAY

POWER OF TWO 8-bit CLA + THREE 4bit CLA + TWO 2-bit CLA:

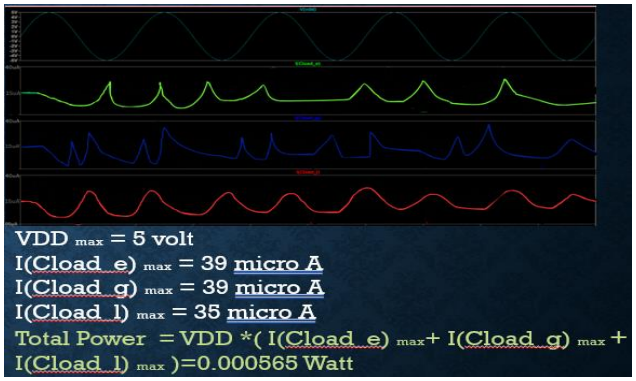

Figure 44: TWO 8 bit CLA + THREE 4 bit CLA + TWO 2 bit CLA POWER

6.3. 32-BIT ADDER (TWO 8-bit CLA + FOUR 4-bit CLA)

Build 32-bit CLA adder using Two 8-bit CLA adders and Four 4-bit CLA adders.

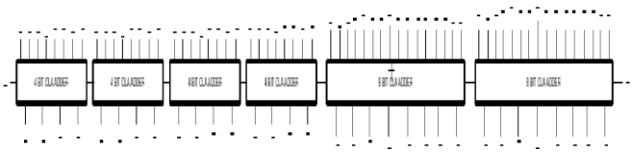

Figure 45: TWO 8-bit CLA + FOUR 4-bit CLA SCHEMATIC

AREA OF TWO 8-bit CLA + FOUR 4-bit CLA:

AREA = 2 * AREA OF 8-BIT CLA ADDER + 4 * 4-BIT CLA ADDER

TOTAL AREA = 2 * 9360 + 4 * 4680 = 37440 (µm)^2

DELAY OF TWO 8-bit CLA + FOUR 4-bit CLA:

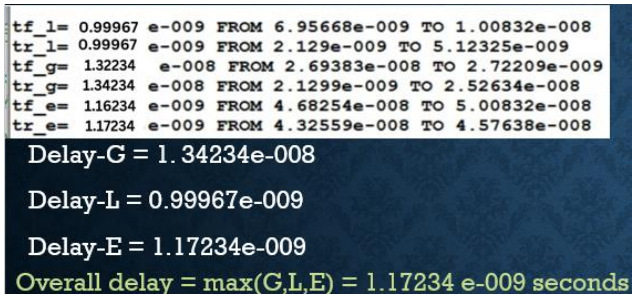

Figure 46: DELAY OF TWO 8-bit CLA + FOUR 4-bit CLA

POWER OF TWO 8-bit CLA + FOUR 4-bit CLA:

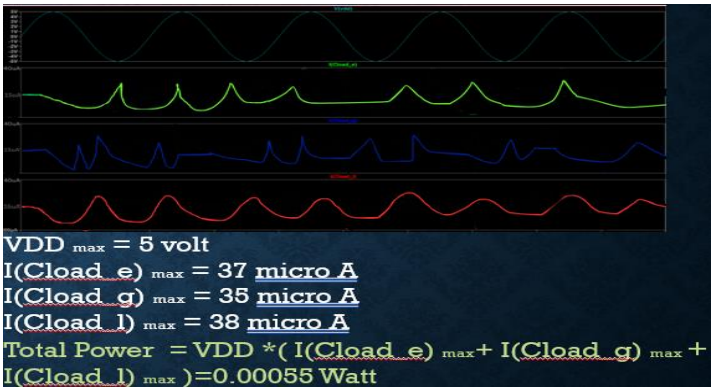

Figure 47: POWER OF TWO 8-bit CLA + FOUR 4-bit CLA

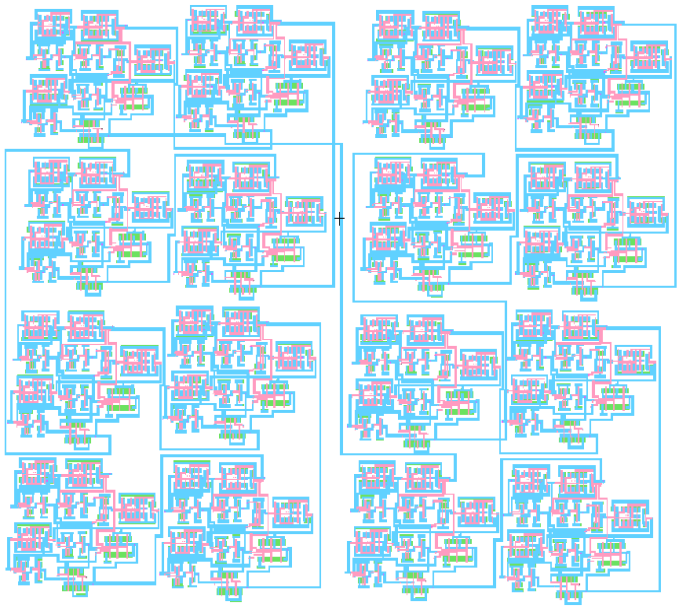

Figure 48: 32-BIT ADDER LAYOUT

7. Simulation results

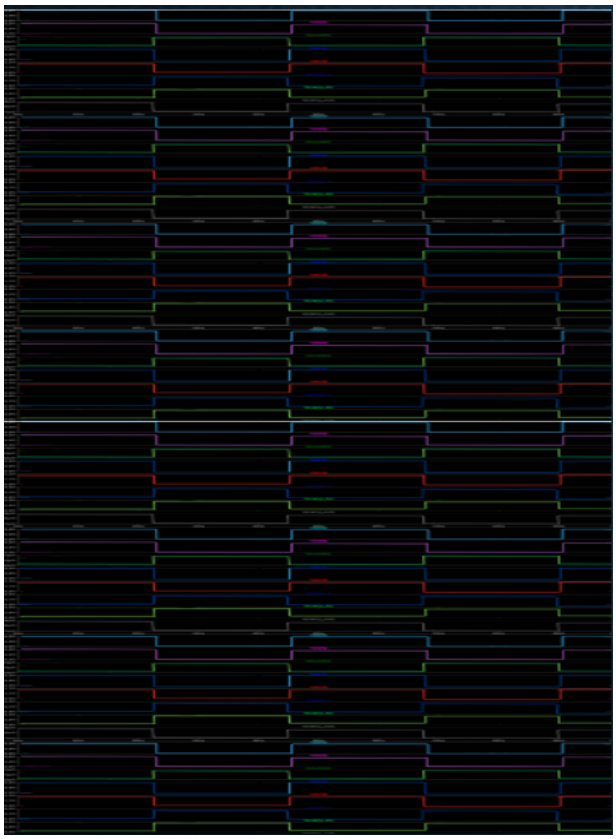

Figure 49: 32-BIT CLA ADDER

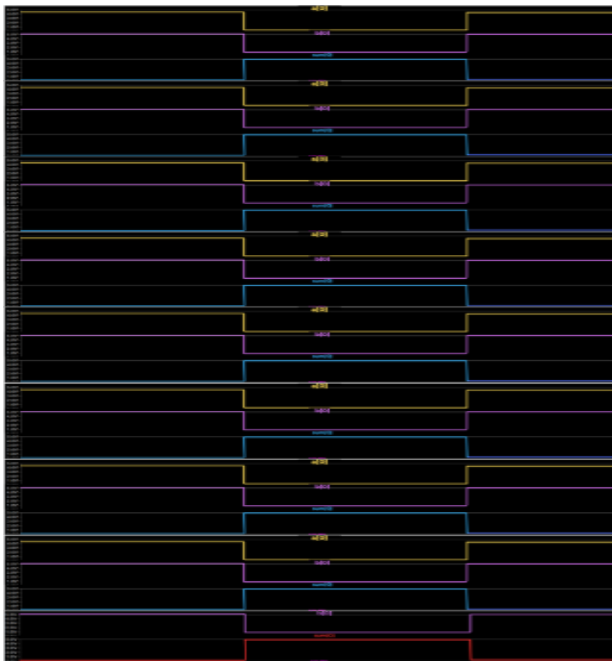

Figure 50: 8-BIT CLA ADDER

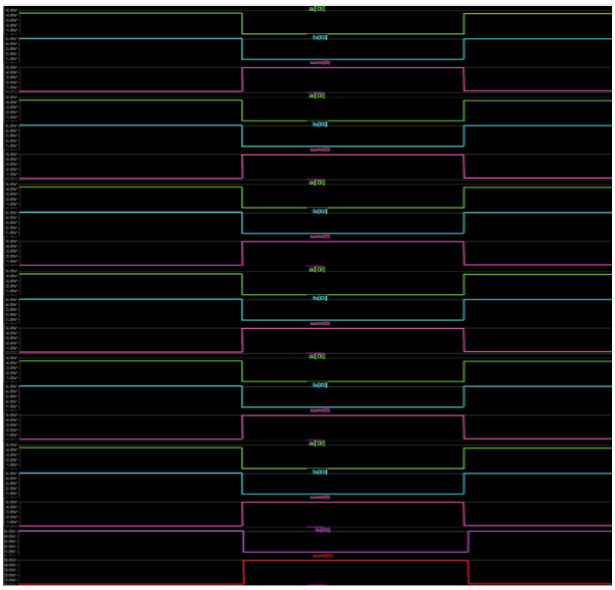

Figure 51: 6-BIT CLA ADDER

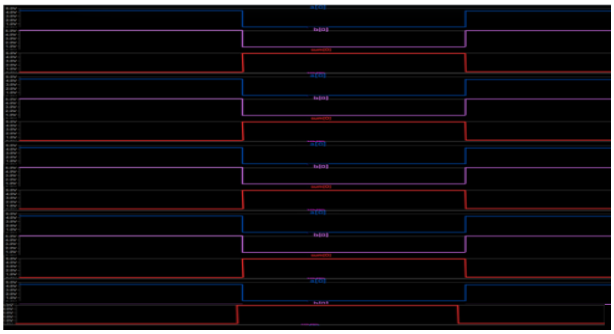

Figure 52: 4-BIT CLA ADDER

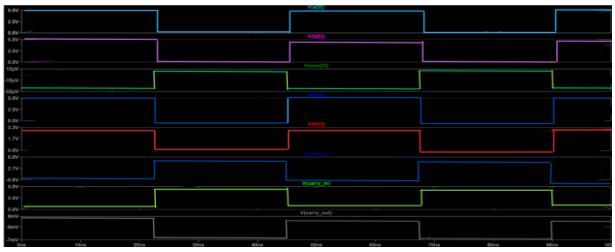

Figure 53: 2-BIT CLA ADDER

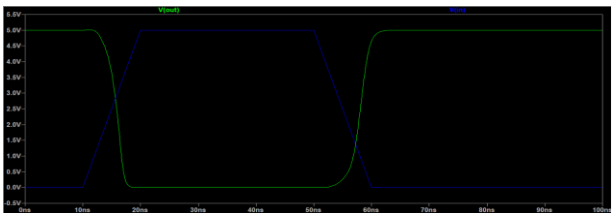

Figure 54: inverter simulation

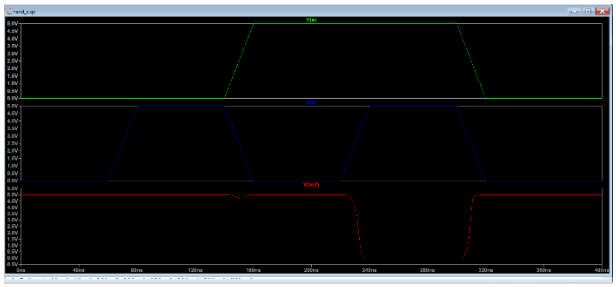

Figure 55: nand simulation

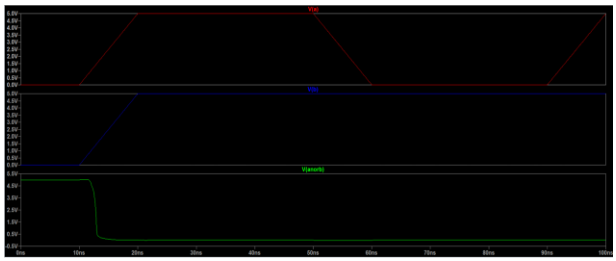

Figure 56: nor simulation

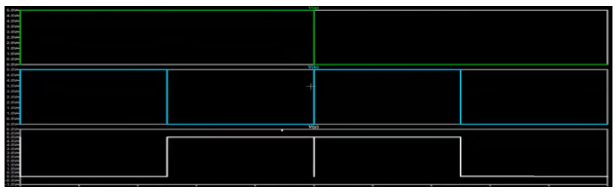

Figure 57: xor simulation

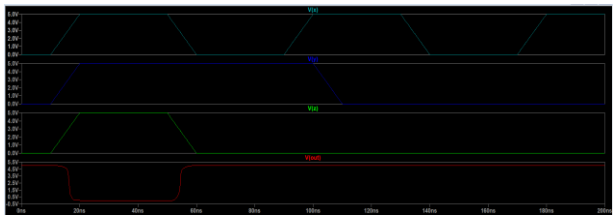

Figure 58: 3-input nand simulation

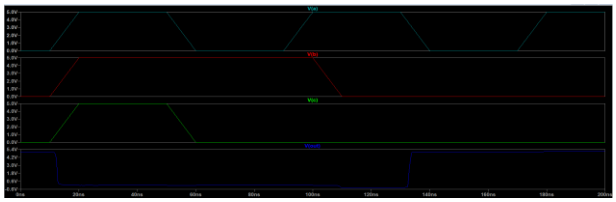

Figure 59: 3-input nor simulation

8. COMPARISON BETWEEN COMMON 32-BIT CLA AND NEW 32-BIT CLA ACCORDING TO PERFORMANCE METRICS

Table 2: common CLA performance metrics (uniform size)

TYPE OF CCLA	AREA ((μm)^2)	POWER (WATT)	DELAY(SECOND)
Eight 4-bit CLA	37440	0.000745	1.6246 e-009
Four 8-bit CLA	37440	0.000695	1.6246 e-009

Table 3: New CLA performance metrics (non-uniform size)

TYPY OF NCLA	AREA ((μm)^2)	POWER (WATT)	DELAY(SECOND)
Four 6-bit CLA + TWO 4-bit CLA	37440	0.000545	1.22234 e-009
TWO 8-bit CLA + THREE 4-bit CLA + TWO 2-bit CLA	37440	0.000565	1.22234 e-009
TWO 8-bit CLA + FOUR 4-bit CLA	37440	0.00055	1.17234 e-009

The results in Table 2 and Table 3 above show that the New CLA (Non-Uniform Size):

- Reduced critical path delay.
- Lower power consumption.
- similar in area but justified by performance gains.

Conclusions

The proposed NCLA architecture demonstrates a balanced approach to optimizing speed, energy efficiency, and area. By leveraging non-uniform module sizes, it addresses the limitations of traditional CLA and PPA architectures. This advancement has significant implications for energy-efficient and high-performance computing systems, particularly in domains requiring low latency and reduced power consumption.

References

[1] https://ijcrt.org/papers/IJCRTT020020.pdf.
Accessed on 3-1-2025 at 4:34 PM.

[2] https://ieeexplore.ieee.org/document/957587.
Accessed on 4-1-2025 at 5:34 PM.

[3] https://www.geeksforgeeks.org/computer-organizationbooths-algorithm/ .
Accessed on 4-1-2025 at 7:39 PM.

[4] https://www.researchgate.net/publication/353595178_Design_and_Evaluation_of_a_32-bit_Carry_Select_Adder_using_4-bit_Hybrid_CLA_Adder.
Accessed on 4-1-2025 at 3:34 PM.

[4] https://iopscience.iop.org/article/10.1088/1742-6596/2225/1/012003/meta .
Accessed on 5-1-2025 at 4:34 PM.

[5] https://testbook.com/question-answer/a-32-bit-adder-is-formed-by-cascading-4-bit-carry--5aa61a0c301a350c238369bb.
Accessed on 5-1-2025 at 4:34 PM.