Dynamic Carry Look-Ahead Adder

Department of Electrical and Computer Engineering
Birzeit University

Raghad Jamhour[1], *Layal Hajji*[2], *Yasmin Al Shawarah*[3]

1220212@*student.birzeit.edu*[1], 1220871@*student.birzeit.edu*[2], 1220848@*student.birzeit.edu*[3]

Abstract

This project presents a dynamic 1x1, 4x4, and 8x8 Carry Look-Ahead Adder (CLA), implemented using dynamic XOR and NAND gates. The 8x8 CLA is extended through modular integration. The design leverages dynamic logic, utilizing pre-charge and evaluation phases to achieve high speed while maintaining scalability.

Despite higher power due to switching activity, the modular approach optimizes complexity, balancing power and performance. This dynamic CLA design offers significant speed improvements over traditional ripple-carry adders, making it ideal for high-performance applications.

I. Introduction

A. Carry Look-Ahead Adder

The Carry Look-Ahead Adder (CLA) is a widely-used technique in digital circuits, primarily designed to speed up the process of addition by overcoming the limitations of traditional ripple-carry adders. The CLA reduces the dependency on the sequential propagation of carry bits, significantly improving the addition time, especially for larger bit-widths. This efficiency makes the CLA a fundamental component in high-performance processors and digital systems. In the context of digital logic, dynamic circuits offer additional speed advantages over static designs. By exploiting the pre-charge and evaluation phases, dynamic logic reduces critical path delays, enabling faster operation, albeit at the cost of increased power consumption. Despite these trade-offs, dynamic logic remains an excellent option for high-speed applications where performance is the primary concern.

B. Motivation

As digital systems demand faster processing speeds, traditional ripple-carry adders (RCAs) become limiting due to their sequential carry propagation. While carry look-ahead adders (CLAs) reduce delay, static logic implementations still struggle with speed in high-performance applications. Dynamic logic, offering faster switching speeds, can address this issue but comes with trade-offs in power consumption. This work uses 300nm CMOS technology to design a 4-bit CLA with dynamic logic, optimizing speed while balancing area and power, providing a scalable solution for high-speed arithmetic operations in modern systems.

II. Background Information

The Carry Look-Ahead Adder (CLA) is a key component in high-performance digital systems, designed to address the delay limitations of Ripple-Carry Adders (RCAs). Traditional RCAs propagate carry bits sequentially, resulting in delays that scale linearly with the number of bits. The CLA overcomes this by computing carry bits in parallel using generate and propagate logic, significantly reducing the delay. This parallel approach makes CLAs highly suitable for modern processors and applications requiring fast arithmetic operations.

II.I. Adder Algorithms

A. Ripple-Carry Adder (RCA)

The RCA is the simplest adder architecture, where each stage computes the sum and passes the carry sequentially to the next stage. While this results in minimal area and power consumption, the sequential carry propagation causes a delay proportional to the bit width. Compared to our dynamic CLA, RCAs exhibit significantly slower performance, especially as the number of bits increases, due to their inherently linear delay [1].

B. Carry Skip Adder (CSA)

The CSA reduces delay by skipping over blocks of bits when the carry does not propagate through them. This approach balances speed and power consumption better than RCAs but lacks the high-speed optimization of CLAs. Compared to our dynamic CLA, CSAs are slower and less efficient for high-performance applications [1].

C. Carry Select Adder (CSeA)

The CSeA improves speed by precomputing potential sums for carry-in values of 0 and 1, selecting the appropriate result when the actual carry is determined. While this reduces delay, it comes at the cost of increased area due to duplicate hardware. Our dynamic CLA avoids this hardware duplication and achieves higher speeds with optimized logic design [1].

D. Carry Increment Adder (CIA)

The CIA divides the adder into blocks where partial results are computed and adjusted based on the carry-in value. This design strikes a balance between speed and area. However, our dynamic CLA outperforms CIAs in terms of delay reduction by exploiting dynamic logic and parallel processing of carry signals [1].

E. Carry Save Adder (CSA)

Primarily used in multi-operand addition, the CSA computes partial sums and carries without propagating the carry, which makes it faster for certain applications like multiplication. However, the design is not well-suited for general-purpose addition where final carry propagation is still needed. Our dynamic CLA provides a more balanced and efficient solution for general-purpose addition [1].

F. Carry Bypass Adder (CBA)

The CBA combines ripple and skip adder principles, bypassing carry propagation for specific blocks when the carry does not propagate. While it offers a trade-off between speed and complexity, its performance is limited compared to our dynamic CLA, which ensures parallel carry computation for all stages [1].

III. Comparison with Dynamic CLA

The implemented dynamic CLA offers significant improvements over other adder algorithms, including static CLAs and alternatives like ripple-carry and carry-select adders. While traditional static CLAs are power-efficient, they suffer from slower performance due to slower gate switching and larger area. Dynamic CLA overcomes these issues by using dynamic XOR and NAND gates, resulting in faster switching speeds and reduced delays and area.

This design provides superior speed and scalability compared to other static designs. While static adders may offer better power efficiency, dynamic CLA strikes a better balance between speed and performance, making it the ideal choice for high-speed, high-performance applications.

IV. Design and implementation

The design of the dynamic Carry Look-Ahead Adder (CLA) utilizes dynamic logic principles, implementing the carry generation and sum calculation using dynamic XOR, NAND gates, and inverters to achieve high-speed performance and scalability. The design addresses the delays associated with static CLAs, using dynamic logic to reduce carry propagation time and improve overall computation speed.

For the carry signal and propagate function, we used dynamic NAND gates and NMOS transistors along with an inverter to compute the carry generate (G) and carry propagate (P) signals. The carry generate (G) signal is generated when both inputs A and B are high, ensuring a carry will propagate to the next bit:

Carry Generate (G): $G = A \cdot B$

The propagate function (P) is calculated as the XOR of the two inputs, A and B. The propagate signal controls whether the carry from the previous stage will affect the current sum calculation:

Carry Propagate (P): $P = A \oplus B$

Using these propagate and generate functions, the carry-out for each bit is calculated dynamically through a NAND-based logic:

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

This dynamic approach to carry calculation removes the sequential carry propagation delay found in ripple carry adders, allowing for faster addition, especially with higher bit-widths.

For the sum calculation, we implemented the XOR operation between the propagate signal (P) and the carry-in (C), producing the sum for each bit:

$S = P \oplus C_{in}$

The XOR gates in our design are implemented using dynamic NMOS logic, allowing faster switching times by utilizing the pre-charge and evaluation phases. This dynamic XOR gate structure performs significantly faster than traditional static XOR gates due to reduced delay in evaluating the sum.

A key feature of the design is the clocking mechanism that synchronizes the pre-charge and evaluation phases of the dynamic gates. We employed PMOS transistors for the clocking system, enabling a shared clock to manage the gates operation. During the pre-charge phase, the PMOS transistors ensure that the outputs are reset to a known state, while during the evaluation phase, the transistors turn off, and the logic gates evaluate the inputs to compute the carry and sum outputs.

The design was implemented using 300nm CMOS technology, which optimizes both the NMOS and PMOS transistors to minimize delays and improve the switching speed. By utilizing dynamic logic and minimizing the critical path, the design achieves much higher speeds than static designs.

The adder's scalability is enhanced by modularity. While the base design implements a 4x4 CLA, the architecture can easily be extended to larger bit-widths, such as an 8x8 CLA, through the modular integration of additional 4-bit blocks. This modular approach enables the efficient handling of operations with larger bit-widths, ensuring that the design can accommodate high-performance requirements without compromising speed. By adding more blocks, the CLA can be extended even further to support 16-bit, 32-bit, or more bit-wide adders, depending on the application.

This flexibility allows the design to maintain performance across various scales, making it suitable for a wide range of high-speed applications, from simple arithmetic operations to complex processing tasks requiring larger bit-widths.

A. *Dynamic NAND Implementation*

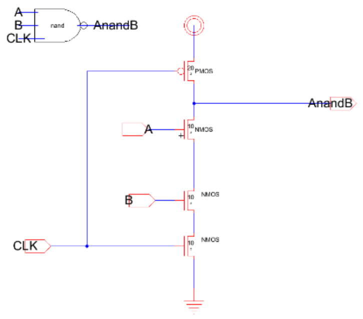


Fig. 1. NAND gate schematic

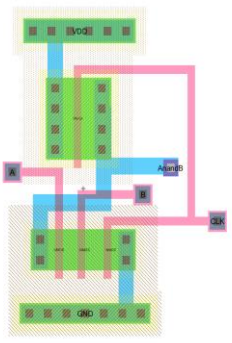


Fig. 2. NAND gate layout

B. *Dynamic XOR Implementation*

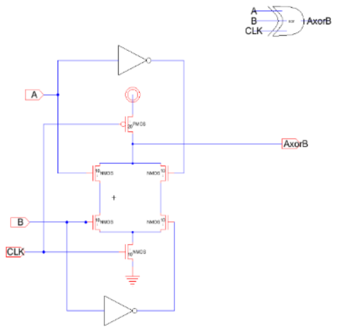


Fig. 3. XOR gate schematic

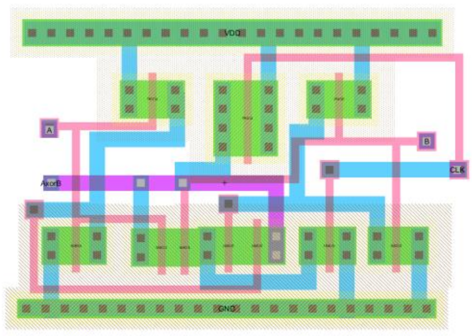


Fig. 4. XOR gate layout

C. *1x1 Carry Look-Ahead Add*

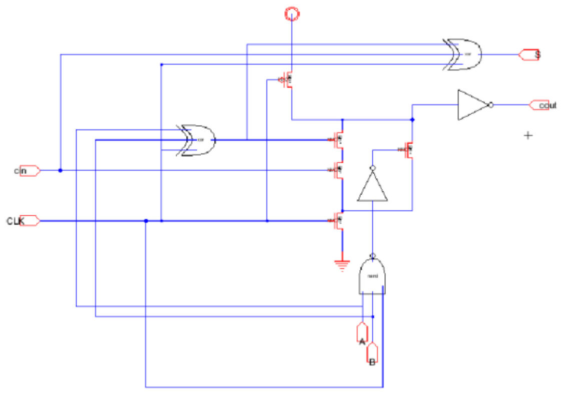


Fig. 5. 1x1 CLA schematic

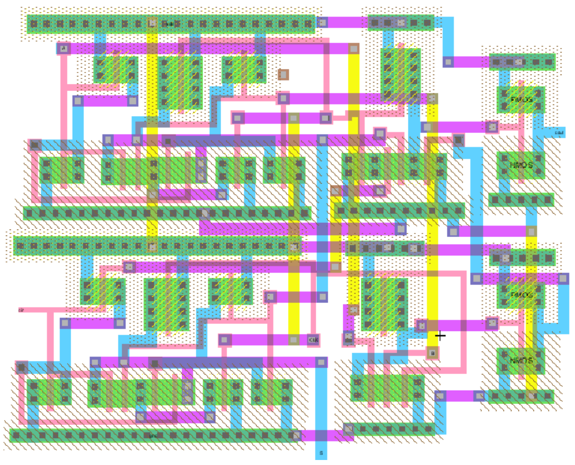


Fig. 6. 1x1 CLA layout

D. *4X4 Carry Look-Ahead Adder*

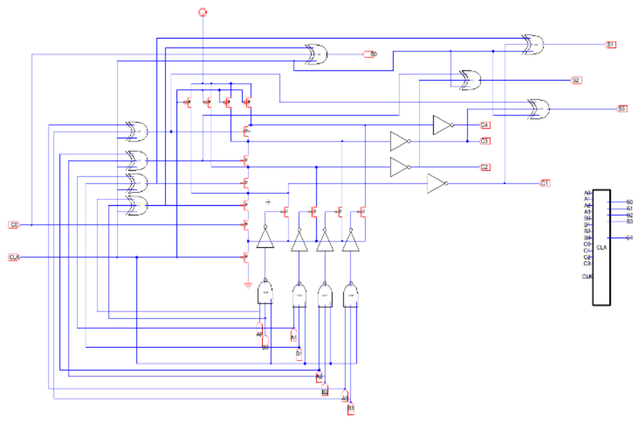


Fig. 7. 4x4 CLA schematic

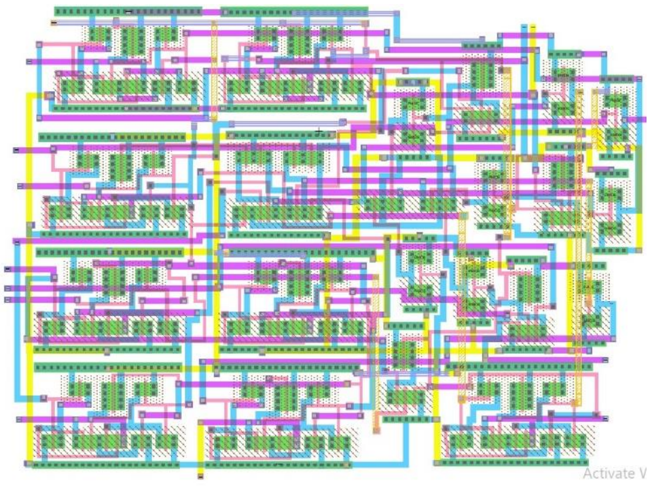


Fig. 8. 4x4 CLA layout

E. *8X8 Carry Look-Ahead Add*

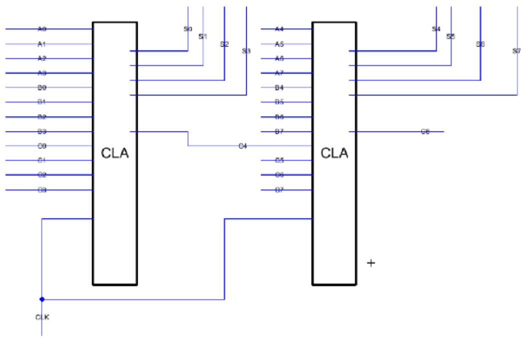


Fig. 9. 8x8 CLA schematic

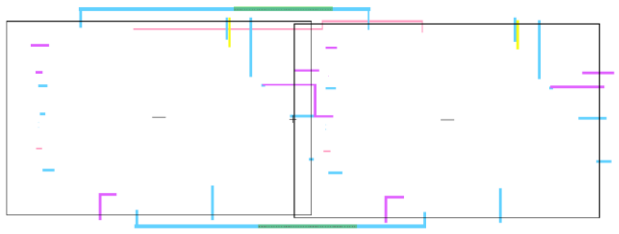


Fig. 10. 8x8 CLA layout

V. RESULTS

A. Results for 1x1 CLA

When A = 0, B = 1 and C_{in} = 1

Sum = 0 when clock is high and C_{out} = 1

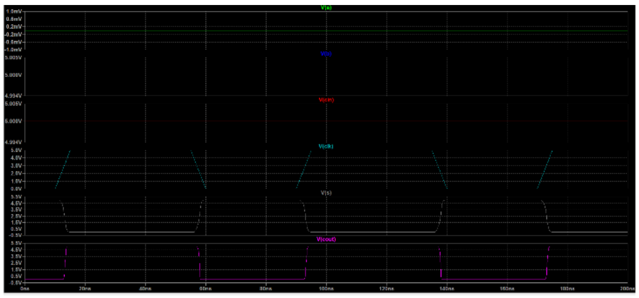


Fig. 11. Results for the schematic of 1x1 CLA

When A = 1, B = 1 and C_{in} = 1

Sum = 1 when clock is high and C_{out} = 1

The glitch in the waveform happens because of small delays in the circuit when signals are changing and due to delays in switching transistors. This is normal and expected in digital circuits. It doesn't cause any problems as long as the output settles to the correct value before it's used, which is the case here.

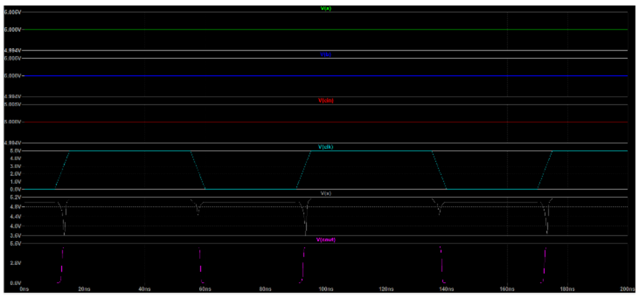


Fig. 12. Results for the layout of 1x1 CLA

B. Results for 4x4 CLA

When A = 15, B =15 and C_{in} = 1

Sum = 15 when clock is high and C_{out} = 1 which is 31 in binary

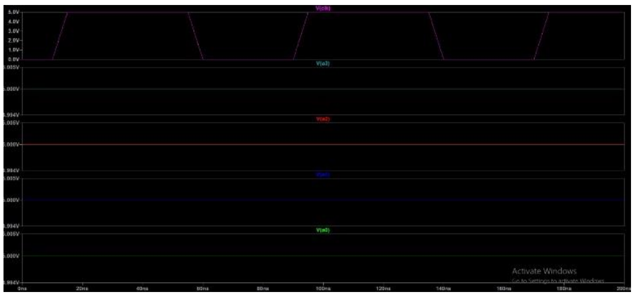


Fig. 13. simulation of the first input

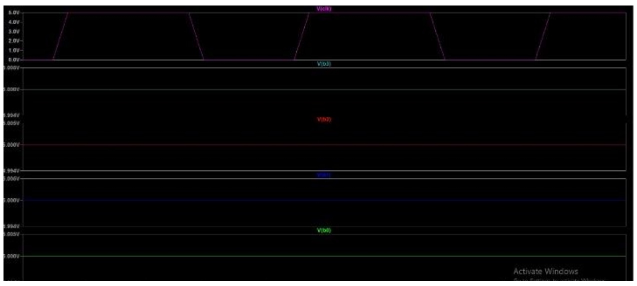
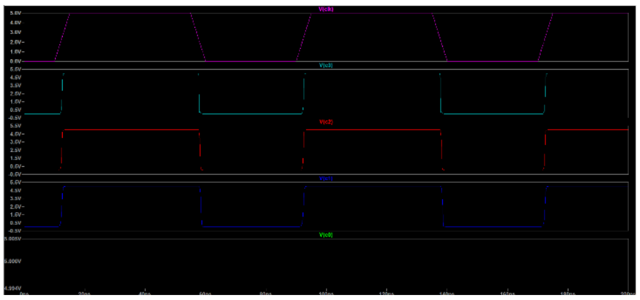


Fig. 14. simulation of the second input



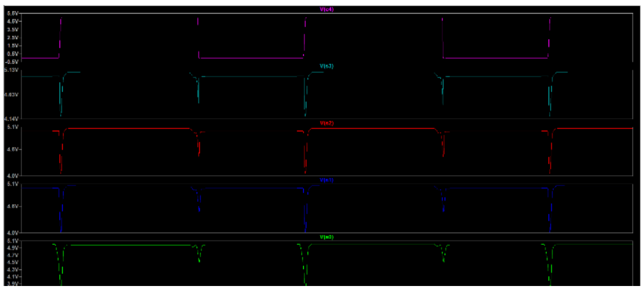


Fig. 15. Results for the schematic of 4x4 CLA

When A = 10, B =5 and C_{in} = 1

Sum = 0 when clock is high and C_{out} = 1 which is 16 in binary

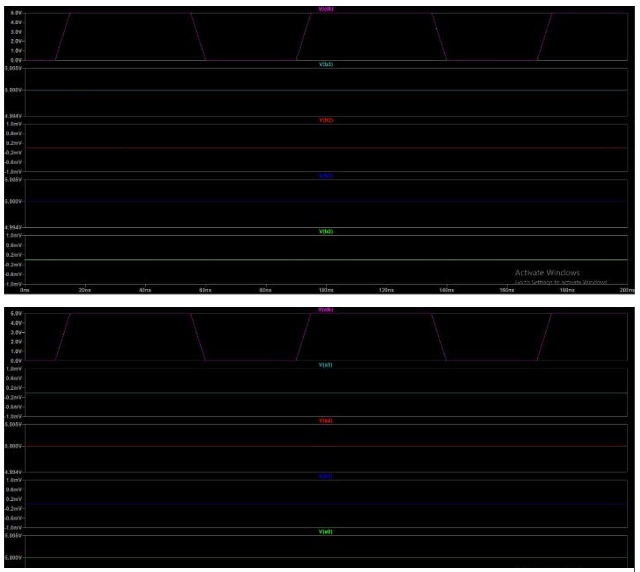
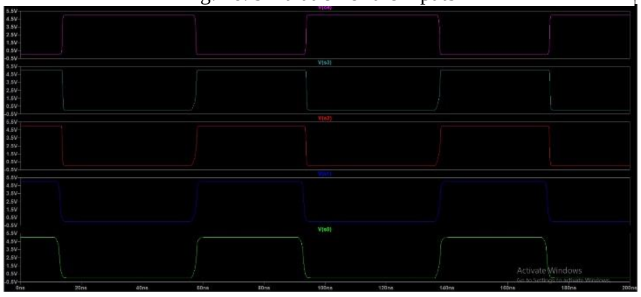


Fig. 16. simulation of the inputs



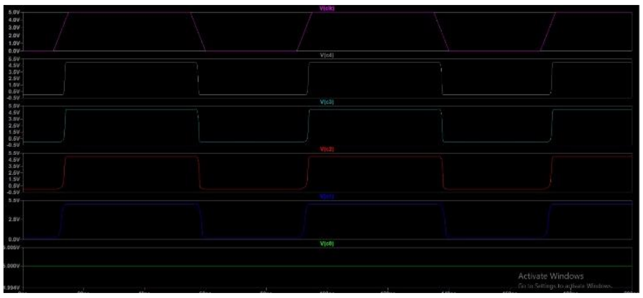
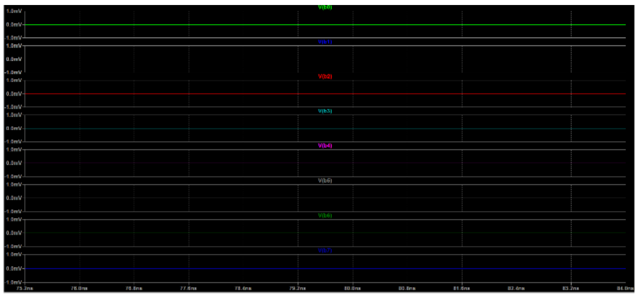


Fig. 17. Results of the layout for 4x4 CLA

C. Results for 8x8 CLA

When A = 255, B =0 and C_{in} = 1

Sum = 0 when clock is high and C_{out} = 1 which is 256 in binary



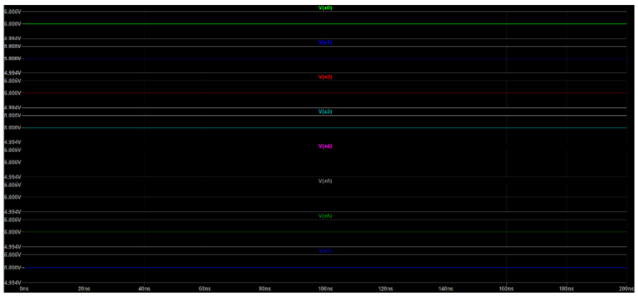
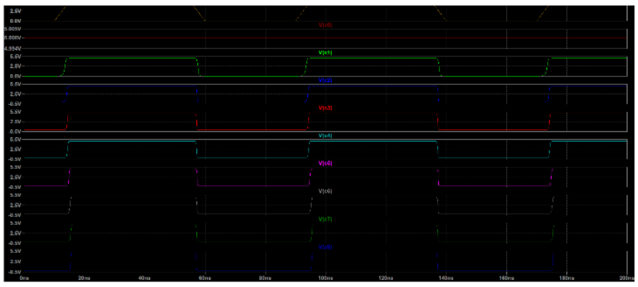


Fig. 18. simulation of inputs



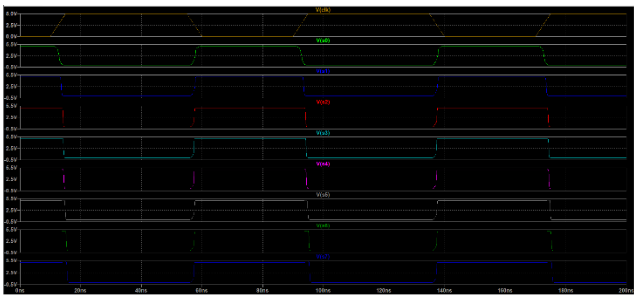
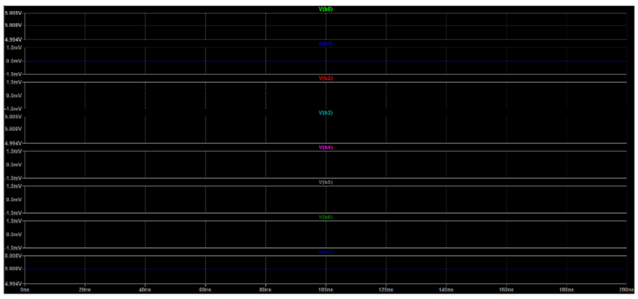


Fig. 19. Results of the layout of 8x8 CLA

When A = 247, B =137 and C_{in} = 1

Sum = 129 when clock is high and C_{out} = 1 which is 385 in binary



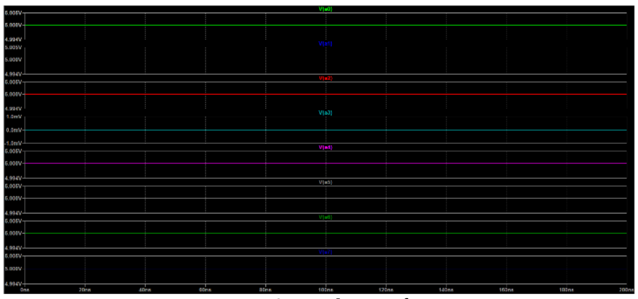
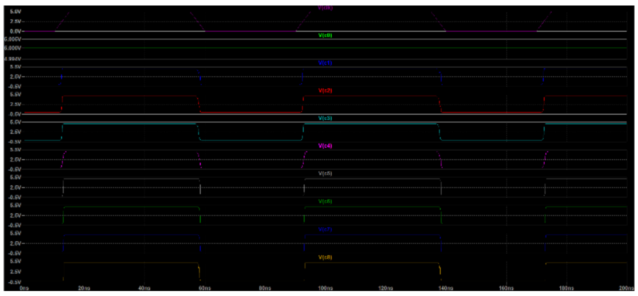


Fig. 20. simulation of inputs



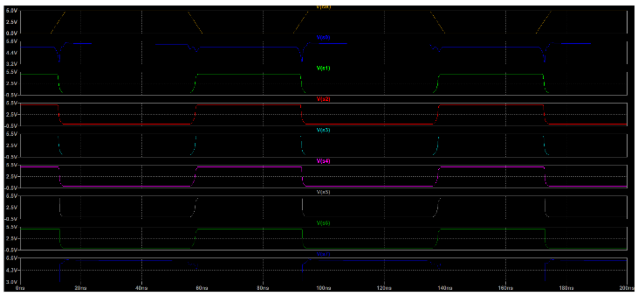


Fig. 21. Results of the schematic of 8x8 CLA

Our design successfully passed all the tests, demonstrating high accuracy and reliability across all parameters. values align with the expected results, confirming the correctness of the design. The design meets the required specifications and exhibits optimal functionality, ensuring its robustness and suitability for practical applications.

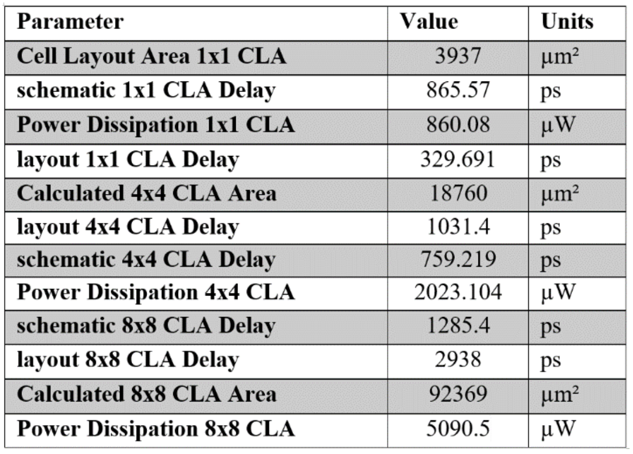
VI. Area, Power, and Delay Calculations

Area was calculated by multiplying each of the width and height by the scale used in the design (300nm).

The delay was calculated by measuring the clock-to-output delay, with cursors placed at the 50% point of the output signal's transition. This allows for accurate timing measurement by capturing the time difference between the clock signal and the output's response.

Power was determined by calculating the highest output voltage and leakage current, both of which contribute to the overall power consumption. The highest voltage output indicates the maximum voltage swing, while the leakage current represents the idle power dissipation in the circuit, which is higher in dynamic logic.

The table below shows the calculated values.

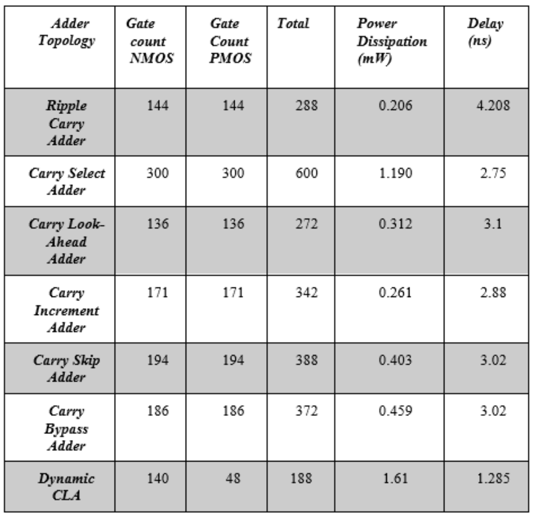


Parameter	Value	Units
Cell Layout Area 1x1 CLA	3937	µm²
schematic 1x1 CLA Delay	865.57	ps
Power Dissipation 1x1 CLA	860.08	µW
layout 1x1 CLA Delay	329.691	ps
Calculated 4x4 CLA Area	18760	µm²
layout 4x4 CLA Delay	1031.4	ps
schematic 4x4 CLA Delay	759.219	ps
Power Dissipation 4x4 CLA	2023.104	µW
schematic 8x8 CLA Delay	1285.4	ps
layout 8x8 CLA Delay	2938	ps
Calculated 8x8 CLA Area	92369	µm²
Power Dissipation 8x8 CLA	5090.5	µW

Table 1. Calculated Values

VII. Performance Comparison of other 8x8 bit Adders

Compared to the reference paper that used 0.12µm 6-metal layer CMOS technology with a static CLA, we improved the design by using a 300nm 5-metal layer CMOS process. Not only reducing the area but also implementing dynamic logic for all gates, leading to faster calculations and reduced delays. The use of dynamic logic enhances performance and ensures more efficient operation compared to static designs. The table below shows the calculations demonstrating the improvements in area, number of devices, delay, and overall efficiency. While the power consumption is higher, this is expected for dynamic logic and it represents a trade-off we made in choosing to prioritize faster performance.



Adder Topology	*Gate count NMOS*	*Gate Count PMOS*	*Total*	*Power Dissipation (mW)*	*Delay (ns)*
Ripple Carry Adder	144	144	288	0.206	4.208
Carry Select Adder	300	300	600	1.190	2.75
Carry Look-Ahead Adder	136	136	272	0.312	3.1
Carry Increment Adder	171	171	342	0.261	2.88
Carry Skip Adder	194	194	388	0.403	3.02
Carry Bypass Adder	186	186	372	0.459	3.02
Dynamic CLA	140	48	188	1.61	1.285

Table 2. Comparison results

VIII. Conclusion

In conclusion, the dynamic Carry Look-Ahead Adder (CLA) design successfully addressed the limitations of static CLA implementations by utilizing dynamic logic for faster carry generation and sum calculation. By leveraging dynamic XOR and NAND gates, we achieved significant speed improvements with lesser area, making our design more efficient compared to traditional static adders and even other dynamic implementations. The modular scalability of the design also allows for easy expansion to handle larger bit-widths. For future improvements, further optimization in power consumption and integrating advanced techniques like pipeline stages or further clocking innovations could enhance both speed and efficiency.

IX. References

[1] K. Singh and S. Sharma, "Area, delay and power comparison of adder topologies," International Journal of VLSI Design & Communication Systems (VLSICS), vol. 3, no. 1, pp. 153-160, Mar. 2012. [Online]. Available: https://aircconline.com/vlsics/V3N1/3112vlsics13.pdf