Design of a 32-bit Carry Look-Ahead Adder with Improved Speed and Power Efficiency

Your Name

*Department of Electrical and Computer Engineering Your University*

City, Country [email@example.com](mailto:email@example.com)

***Abstract*—The purpose of this paper is to present a carry look-ahead adder (CLA) design with improved speed and power efficiency. The Carry Look-Ahead Adder (CLA) is a fundamental circuit in digital systems and computers that performs arithmetic operations such as addition and subtraction. This project uses uniform-sized CLA units to reduce power consumption and in- crease speed, and uses this technique to evaluate the performance of a 32-bit adder. The number addition process is divided into multiple stages, where each CLA unit speeds up the calculations by executing the operations in parallel. This approach contributes significantly to improving the performance of the digital processor by reducing the time delay and increasing the efficiency of power consumption. Through this project, we seek to improve digital performance and reduce power consumption, which contributes to the development of more efficient digital systems in the future, especially in areas such as digital processors and high-performance computing devices.**

***Index Terms*—Carry Look-Ahead Adder (CLA), power efficiency, digital systems, 32-bit adder, MGDI, CMOS.**

1. Introduction

Due to rapid innovation and the expansion of technology, electronic circuits have become vital to many aspects of our daily lives. These circuits run the hidden engines that power many of the devices we use every day. As they become more reliant on them, new problems also arise in addition to the capabilities these chips offer. One of the most important issues of these is the **power efficiency**. One of the main goals in contemporary electronic circuit design is for engineers to strike a perfect balance between powerful performance and power consumption [2]. Another challenge is to make chips more efficient **in space**. As technology advances, it has become necessary to increase processing power in smaller spaces. This requires innovative solutions to organize small components on the chip surface, such as transistors, so that the space is used efficiently. Optimizing the distribution of components on the chip to minimize the space consumed is one of the main challenges that must be solved in the design of integrated circuits [4].

With this project, we created a carry look-ahead adder (CLA) that’s fast and power efficient. The adder is the heart of digital systems. It powers processors, embedded systems, and high-performance computers. A ripple carry adder (RCA) has propagation delays, which cause a bit to wait for the previous stage’s carry output, resulting in slower performance. The RCA has linear carry propagation, so it is not suitable for

high-speed applications despite being simple and area-saving [5]. In contrast to this, the CLA design reduces these delays by precalculating carry signals, allowing parallel computations to be performed and improving the performance of the device [1].

The key advantage of the CLA over the RCA is that it computes the carry signals in advance. An RCA propagates linearly from one stage to the next, resulting in a linear delay with the number of bits. In the case of an N-bit RCA, the total delay can be expressed as follows:

DelayRCA = *N × t*FA

where *t*FA is the delay of a single full adder stage .

In contrast, the CLA computes the carry signals for all stages in parallel using the following equations:

*Ck*+1 = *Gk* + *Pk · Ck*

where:

* *Ck*+1 is the carry output of the *k*-th stage,
* *Gk* is the **generate** signal (*Gk* = *Ak · Bk*),
* *Pk* is the **propagate** signal (*Pk* = *Ak ⊕ Bk*),
* *Ck* is the carry input to the *k*-th stage [1].

By precomputing the carry signals using these equations, the CLA reduces the critical path delay significantly. For example, in a 4-bit CLA, the carry signals can be computed as:

*C*1 = *G*0 + *P*0 *· C*0

*C*2 = *G*1 + *P*1 *· G*0 + *P*1 *· P*0 *· C*0

*C*3 = *G*2 + *P*2 *· G*1 + *P*2 *· P*1 *· G*0 + *P*2 *· P*1 *· P*0 *· C*0

*C*4 = *G*3+*P*3*·G*2+*P*3*·P*2*·G*1+*P*3*·P*2*·P*1*·G*0+*P*3*·P*2*·P*1*·P*0*·C*0

This parallel computation allows the CLA to achieve a much lower delay compared to the RCA, especially for larger bit widths [6].

Therefore, our project uses **uniformly sized CLAs** to build a 32-bit adder. In this modular approach, all the carry com- potations are parallelized so that the speed of the application can be increased while the number of duplicate operations is minimized to decrease the power consumption [1]. The result of this design is that we are able to overcome the limitations

of traditional adders, such as the RCA, and provide a much more efficient solution for modern digital systems as a result of eliminating the limitations of traditional adders [5].

1. *OR Gate*

The OR gate is one of the basic logic gates that performs the” OR” operation. Produces an output of 1 if at least one of the inputs is 1, and if all the inputs are zero, the output is zero. OR gates are used in many applications, such as control and comparison, and can support an unlimited number of inputs. We designed and implemented OR gates with different numbers of inputs. An OR gate with 2 inputs was built, another with 3 inputs, and an OR gate with 4 inputs. Here we shall discuss each one separately. We designed the gates using NMOS and PMOS transistors with the transistor sizes adjusted so that the NMOS was 10 and the PMOS was 20.

1) *2-input OR Gate:*

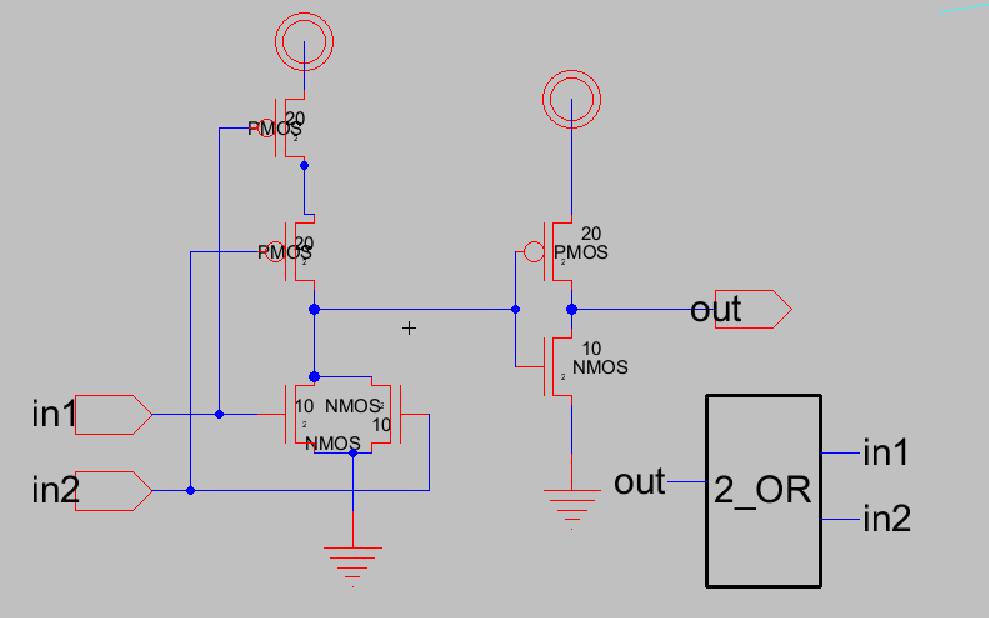


Fig. 1. Schematic of 2-input OR gate

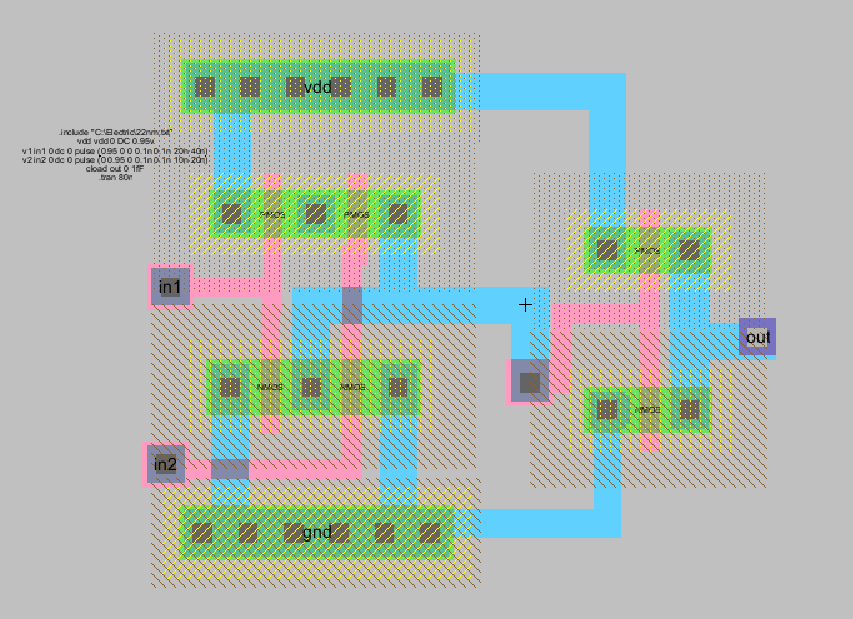


Fig. 2. Layout of 2-input OR gate

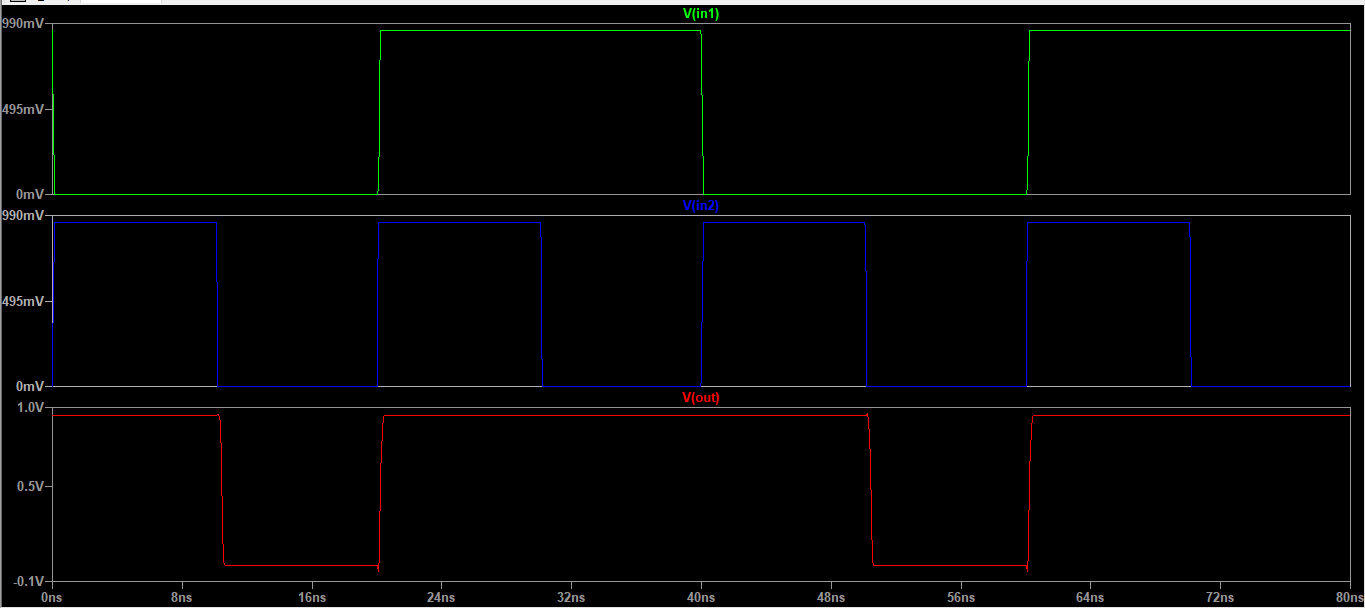


Fig. 3. Output of 2-input OR gate

2)*3-input OR Gate:*

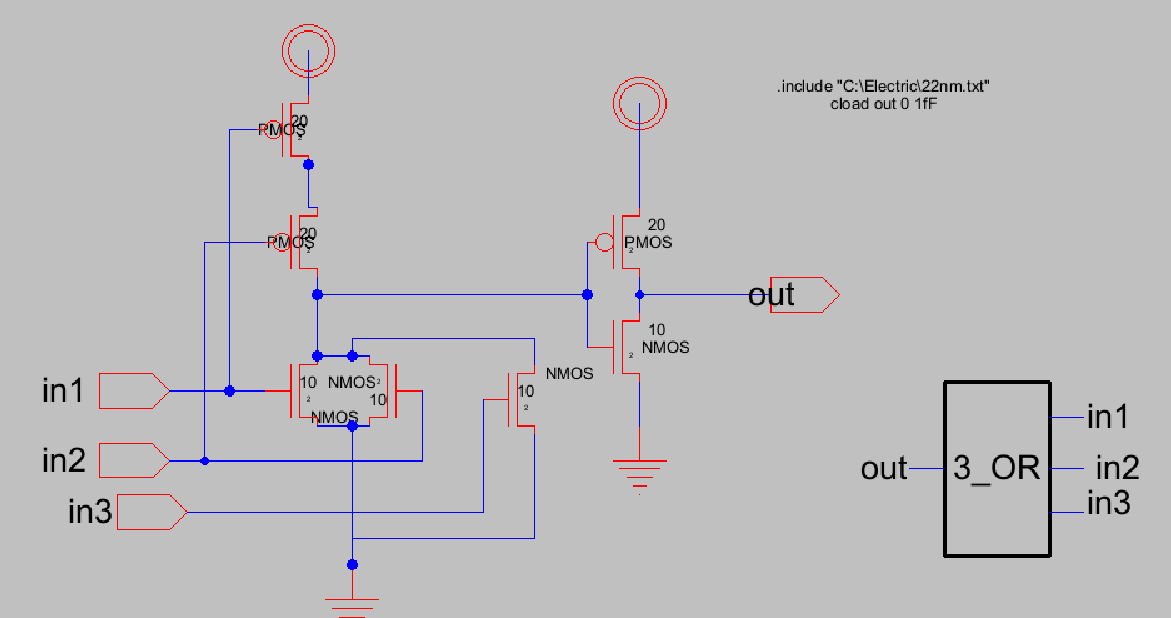


Fig. 4. Schematic of 3-input OR gate

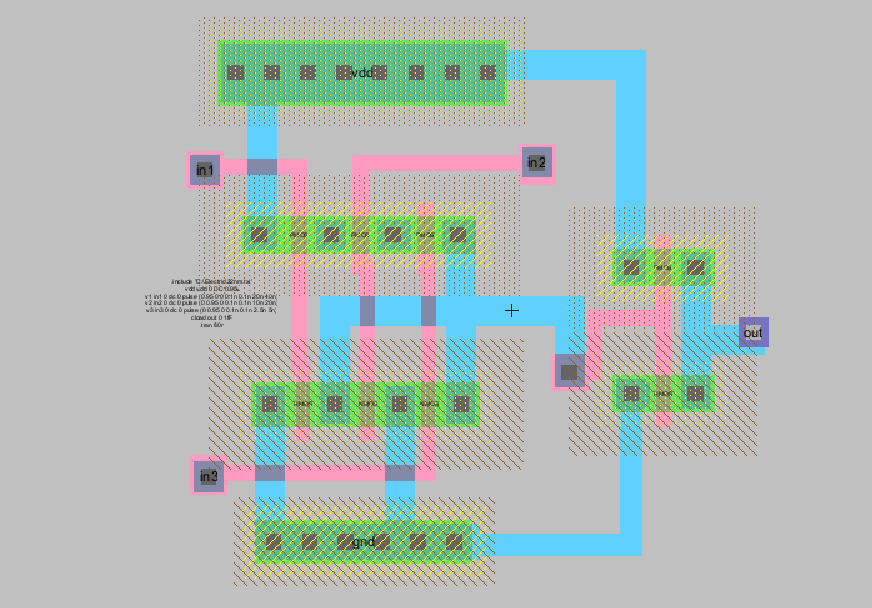


Fig. 5. Layout of 3-input OR gate

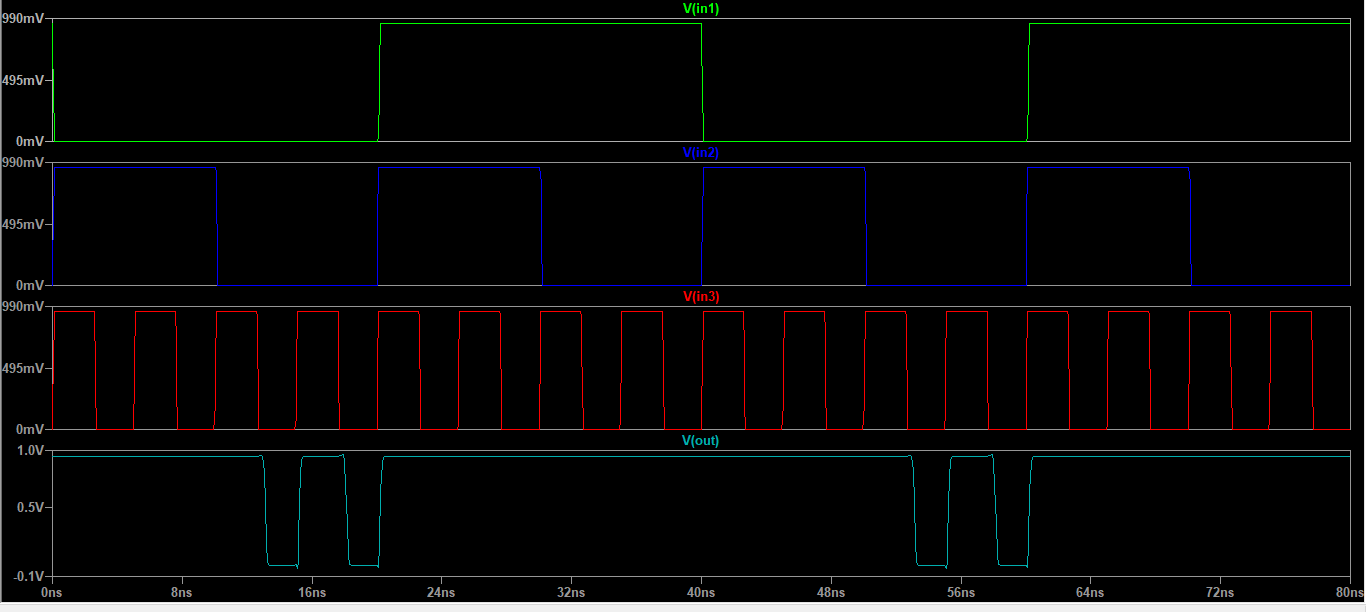


Fig. 6. Output of 3-input OR gate

1. *4-input OR Gate:*

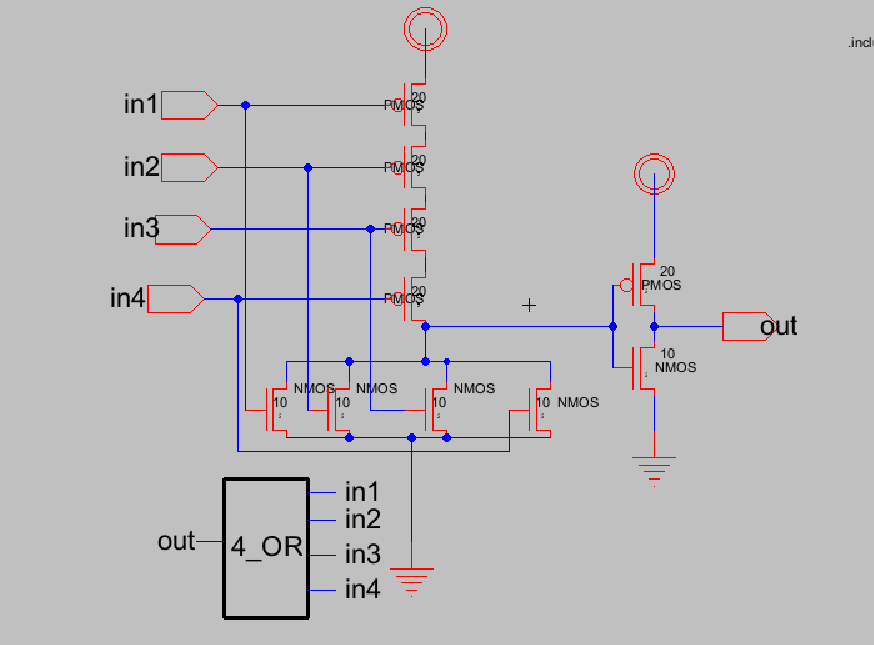
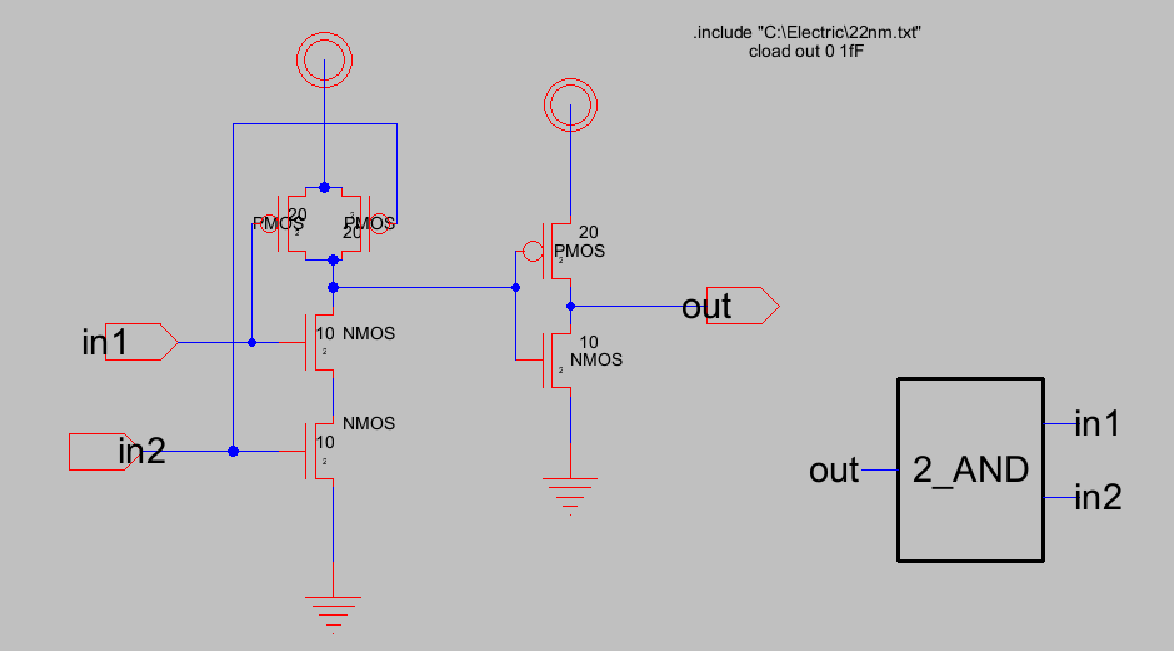


Fig. 7. Schematic of 4-input OR gate

3 inputs, and an OR gate with 4 inputs. Here we will discuss each one separately.

1. *2-input AND Gate:*



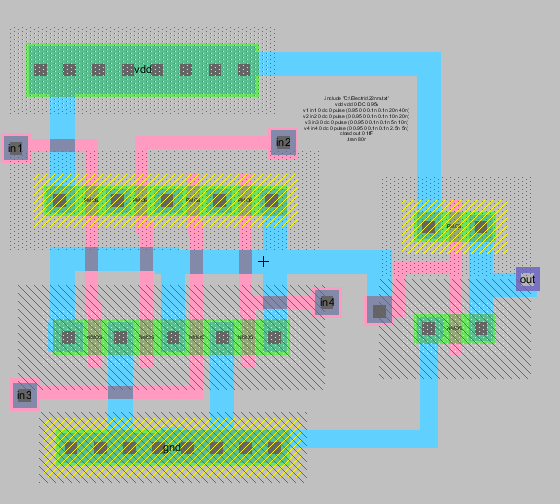


Fig. 8. Layout of 4-input OR gate

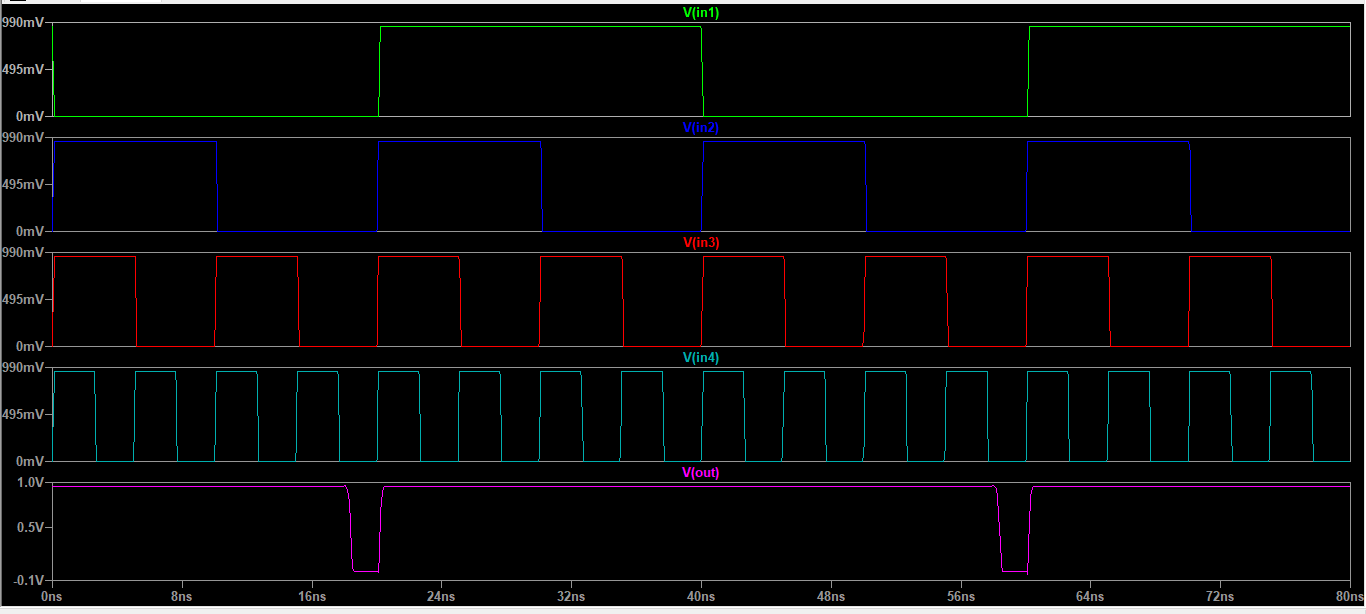


Fig. 9. Output of 4-input OR gate

1. *AND Gate*

Logic gates that implement the logical operation” AND” include AND gates. When all the inputs are 1, this gate will output 1, but if any of the inputs are zero, the output will be 0. In this project, AND gates are built using NMOS and PMOS transistors, where the size of the transistors is specified such that the size of NMOS is equal to 10, and the size of PMOS is equal to 20. The gate with 2 inputs was built, another with

Fig. 10. Schematic of 2-input AND gate

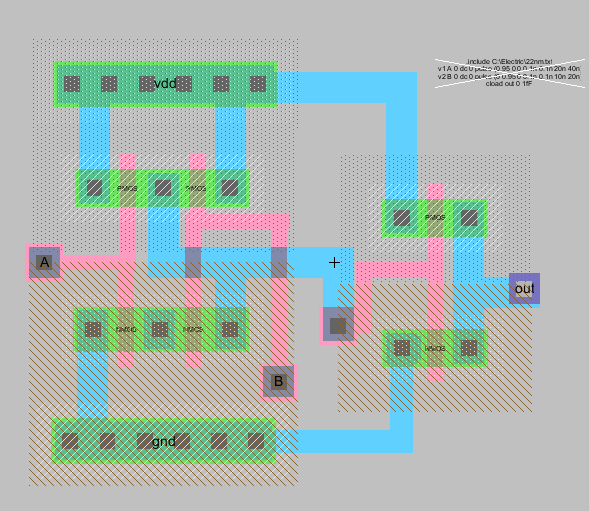


Fig. 11. Layout of 2-input AND gate

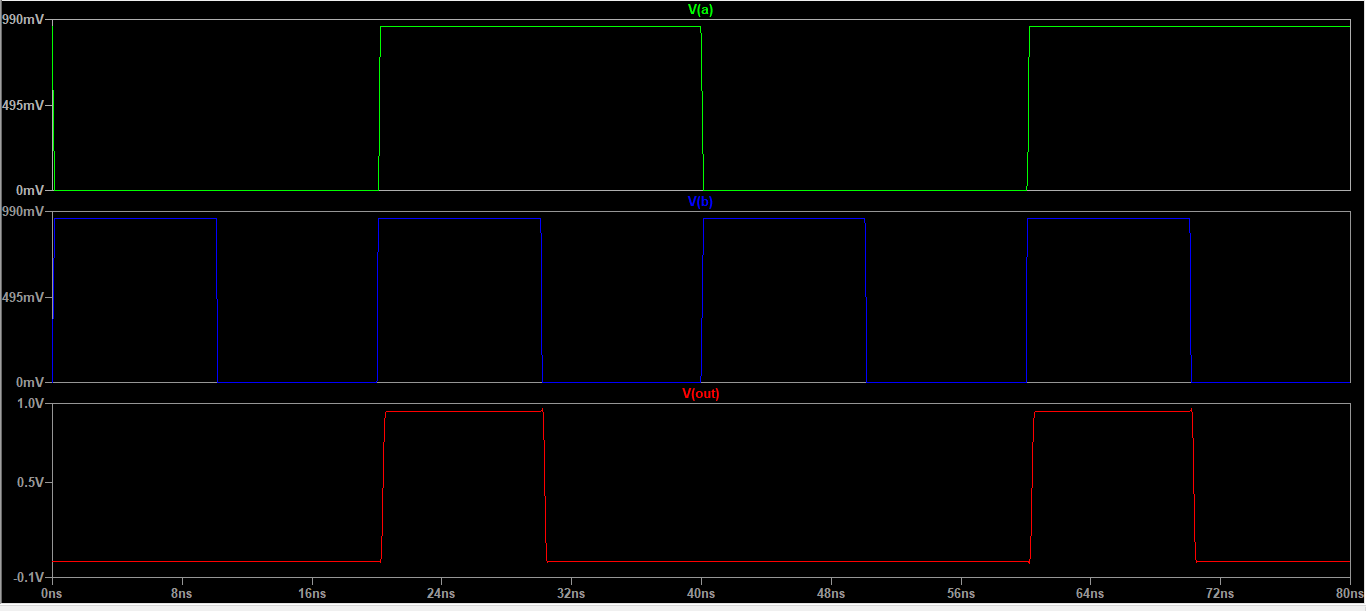


Fig. 12. Output of 2-input AND gate

*2)3-input AND Gate:*

*3)4-input AND Gate:*

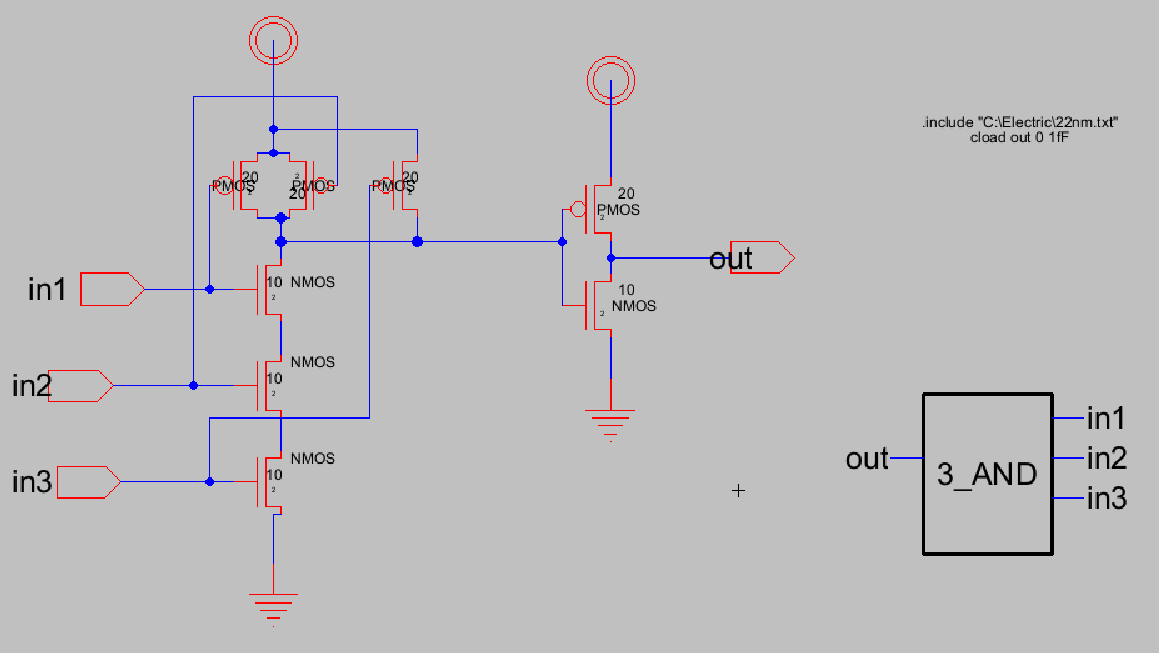
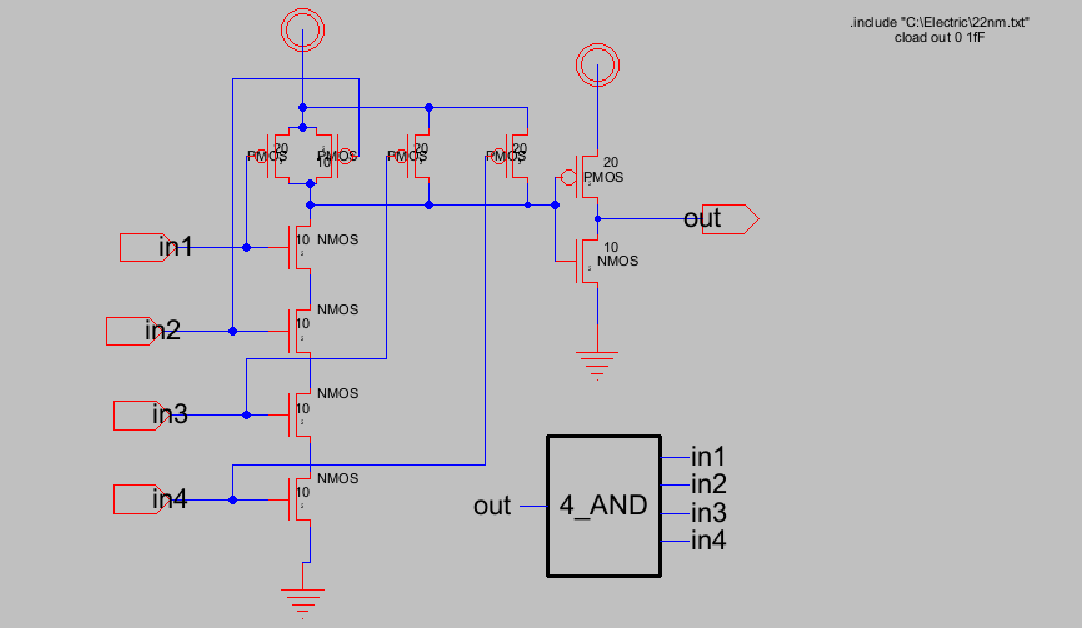
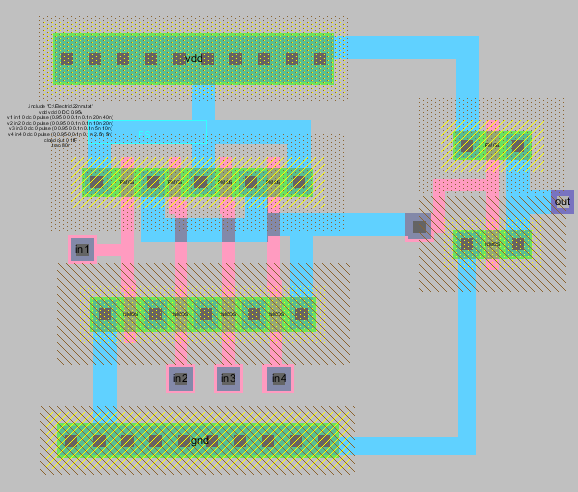
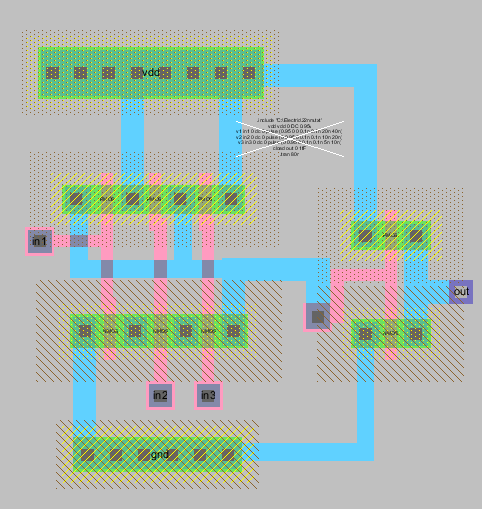
 

Fig. 13. Schematic of 3-input AND gate

Fig. 16. Schematic of 4-input AND gate



Fig. 14. Layout of 3-input AND gate

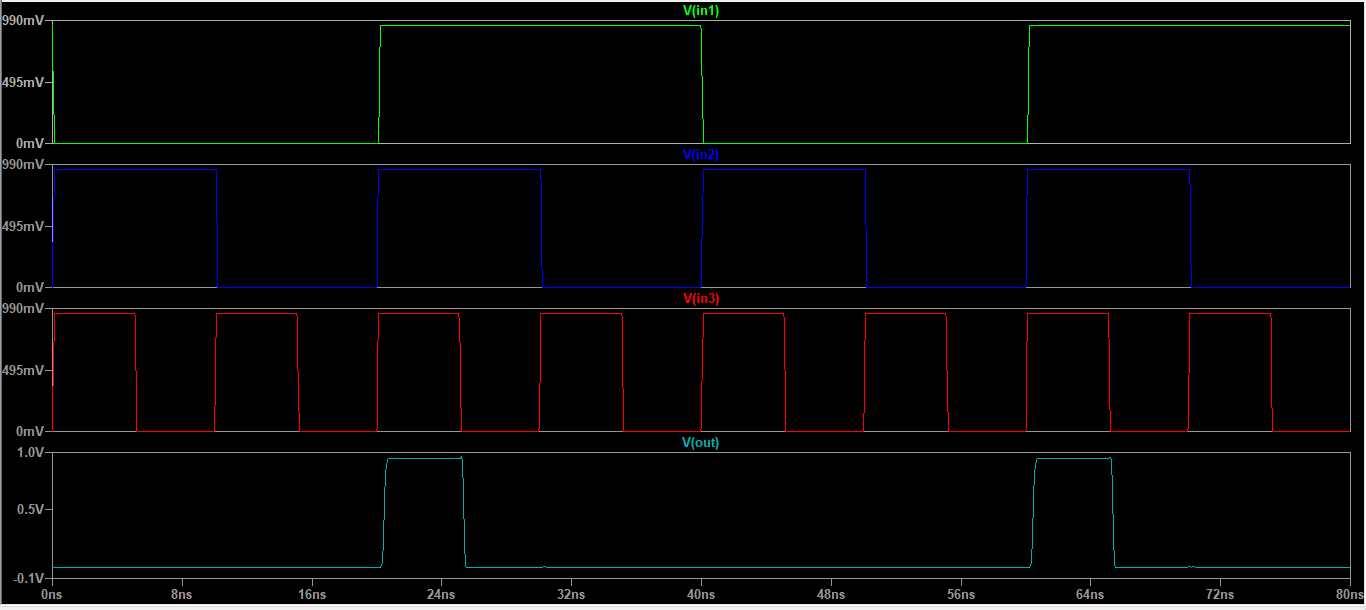


Fig. 15. Output of 3-input AND gate

Fig. 17. Layout of 4-input AND gate

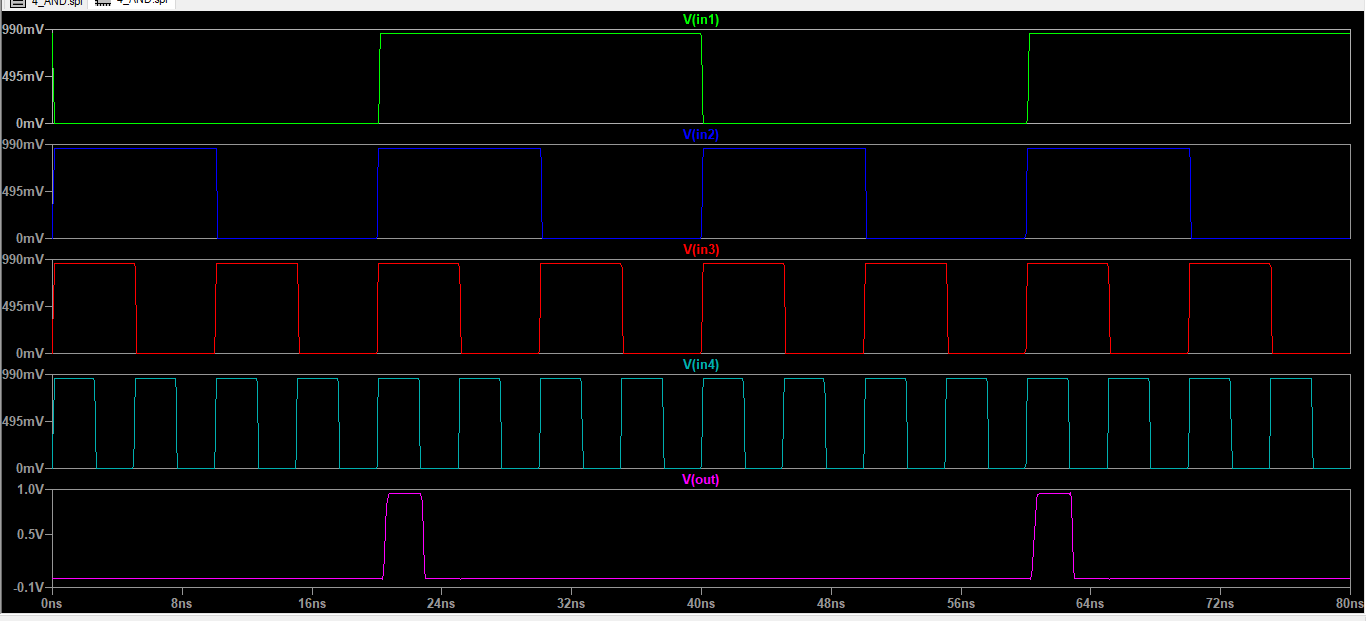


Fig. 18. Output of 4-input AND gate

1. *MGDI XOR Gates*

MGDI is an improved digital circuit design technique such as XOR. This technique is based on improving the traditional design of logic gates. It works by means of the traditional XOR gate used to perform the Exclusive OR operation, where the result is 1 if the inputs are different and the result is 0 if the inputs are equal. Usually, the XOR gate is designed using traditional CMOS transistors, but in MGDI technology, the design is modified to improve performance. One of the advantages of this technology is that it reduces power consumption as it uses fewer transistors compared to traditional designs, which leads to reduced power consumption. It also

increases speed; in addition to reducing space by using fewer transistors and other components, the space required on the chip can be reduced, allowing more functions to be integrated in the same space.

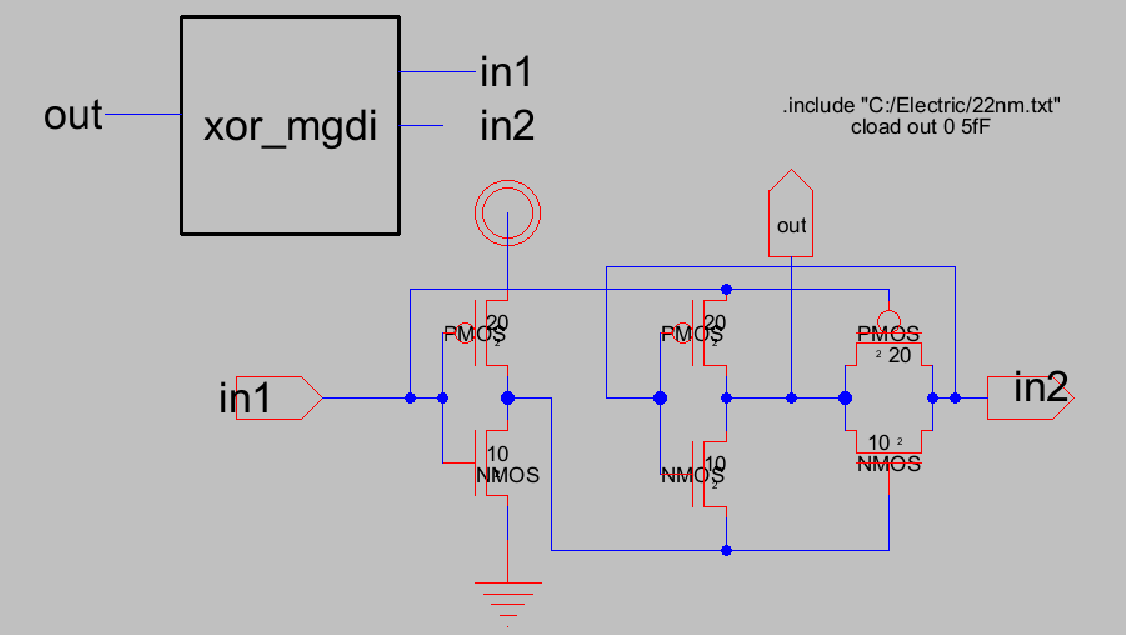


Fig. 19. Schematic of MGDI XOR gate

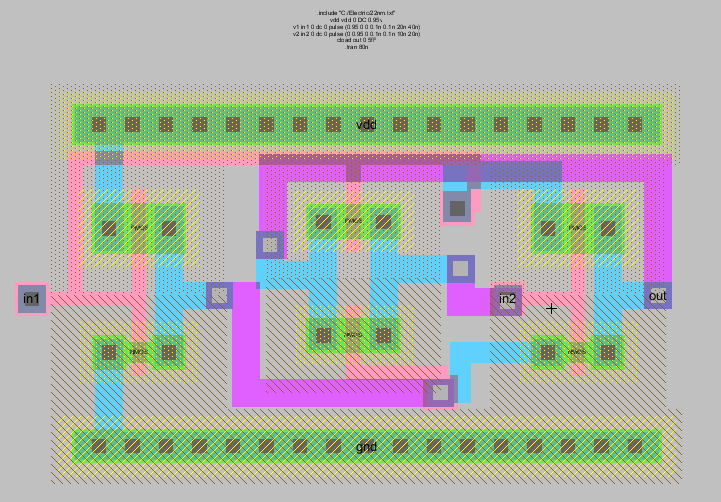


Fig. 20. Layout of MGDI XOR gate

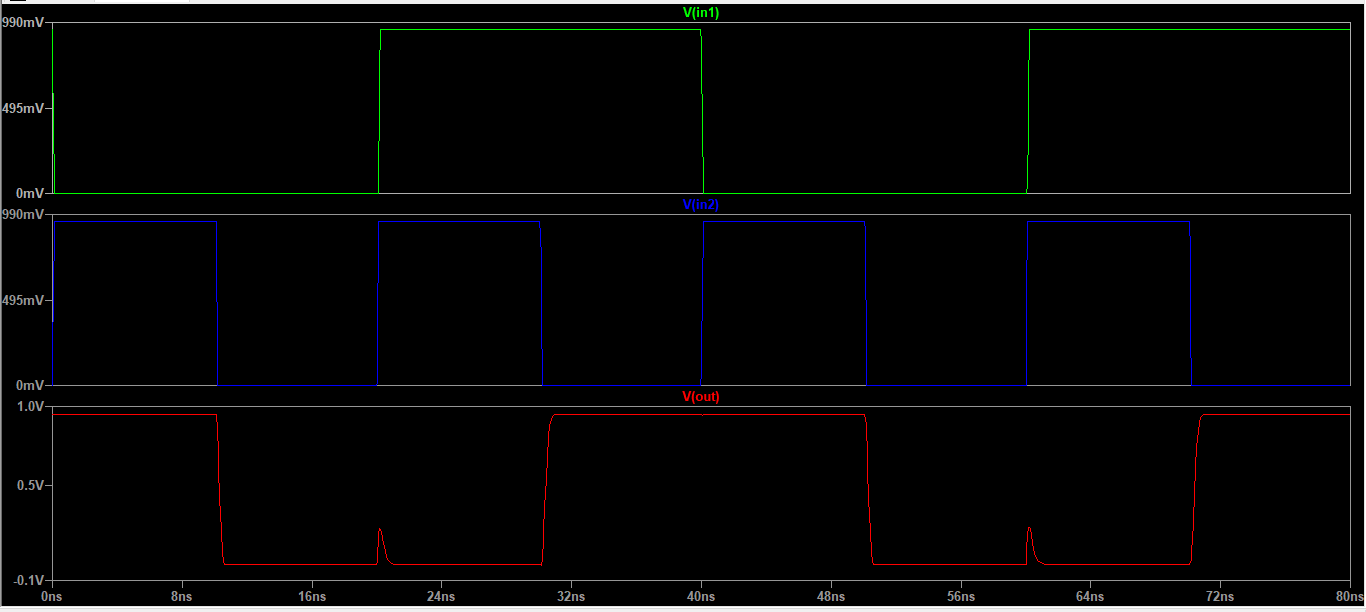


Fig. 21. Output of MGDI XOR gate

1. *4-bit Adder with Carry*

This design uses logic gates (XOR, AND, OR) to build a 4-bit adder, with an emphasis on the XOR gate (with MGDI technology) to improve efficiency. In this design, the work is split up between logic gates as needed for each step of addition, where the XOR gate (MGDI) calculates the partial sum S in each stage, and the AND gate (2-bits, 3-bits, 4-bits) calculates the partial carry based on the correlation. For OR (with the same number of AND gates), there is a summation of the signals resulting from the AND gates to reach the final output.

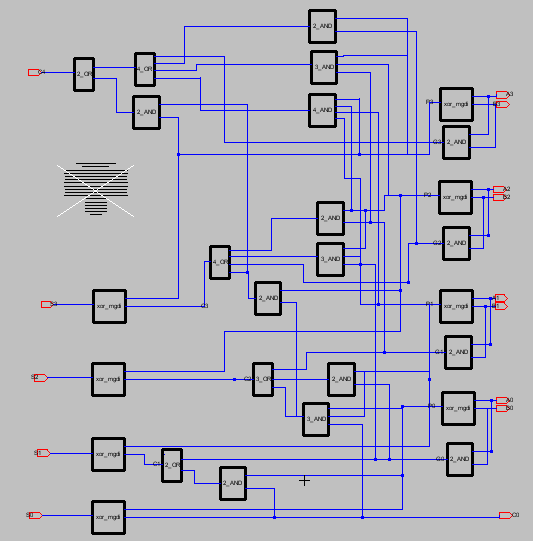


Fig. 22. Schematic of 4-bit adder with carry

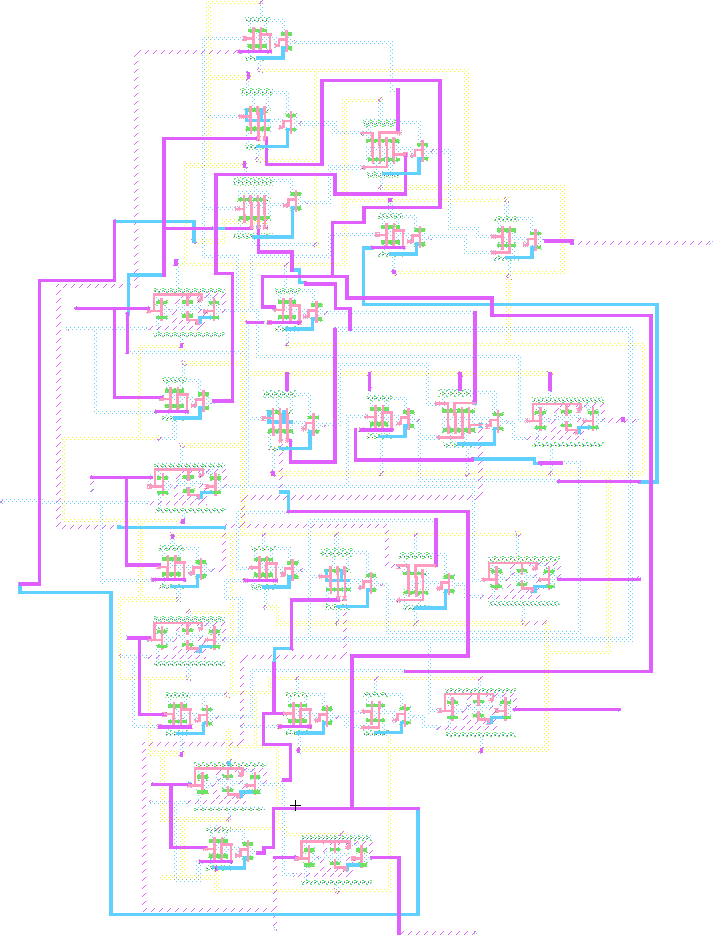


Fig. 23. Layout of 4-bit adder with carry

The equations for the 4-bit adder with carry are as follows:

*Sk* = (*Ak ⊕ Bk*) *⊕ Ck* (1)

*Ck*+1 = (*Ak · Bk*) + ((*Ak ⊕ Bk*) *· Ck*) (2)

To verify the validity of the system, we made manual calculations and compared them with the result we obtained from the simulation. Manual solution:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | A | B | S | C |
| 0  1  2  3  4 | 0  1  1  1 | 0  1  1  1 | 1  0  1  1 | 1  0  1  1  1 |

TABLE I

Manual solution for 4-bit adder with carry

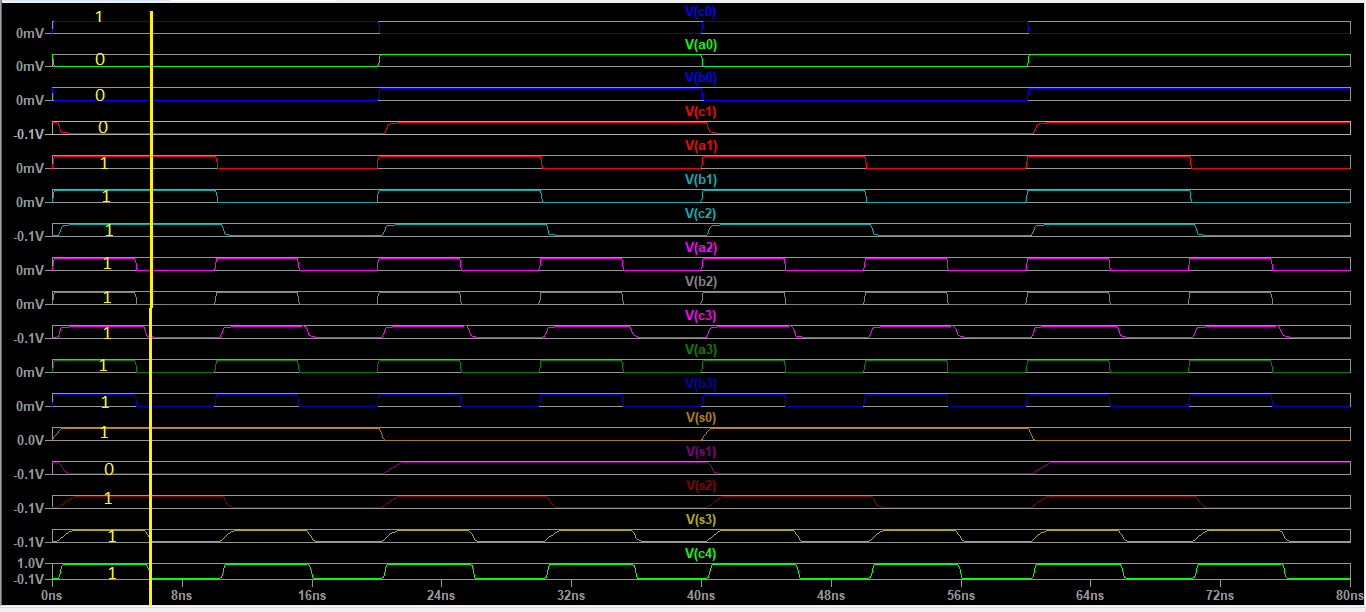
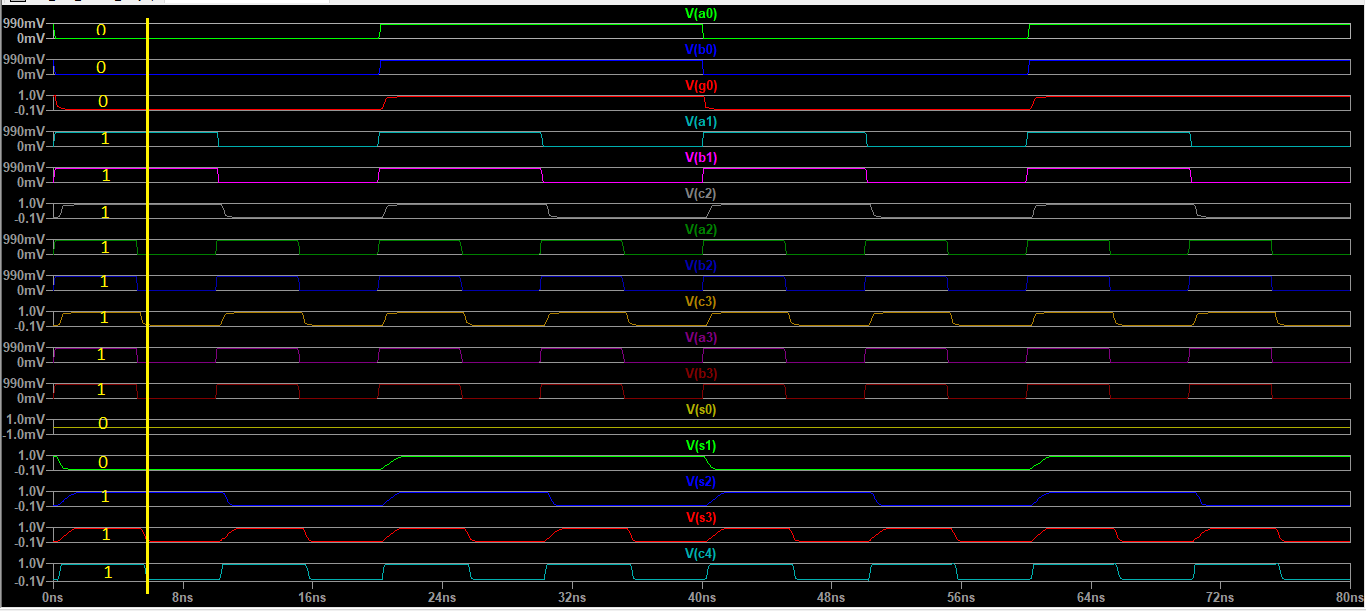
 

Fig. 24. Output simulation of 4-bit adder with carry

The accuracy of the digital design (using XOR, AND, and OR) was confirmed by the consistency of the results between the two methods.

1. *4-bit Adder without Carry*

Originally, it was designed as a 4-bit adder with carry, but here without taking into account the carry. By using logical operations such as AND, OR, and XOR, this circuit generates the desired result. Due to the absence of a carry, the goal is to simply add the corresponding bits and produce a result containing four bits, ignoring the carry resulting from arithmetic operations, therefore the result is stored in the added bits only.

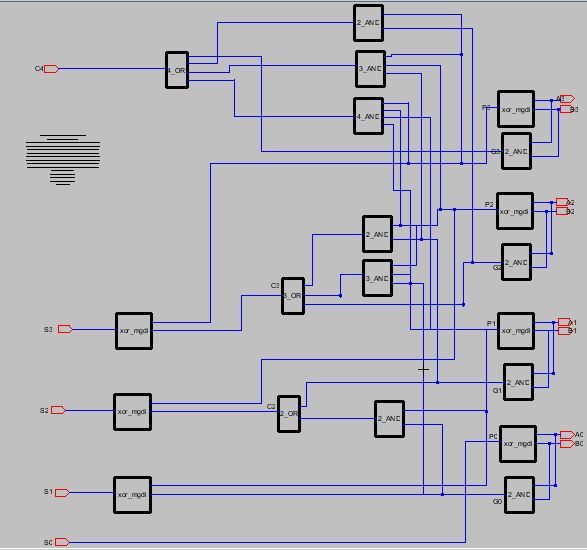


Fig. 25. Schematic of 4-bit adder without carry

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | A | B | S | C |
| 0  1  2  3  4 | 0  1  1  1 | 0  1  1  1 | 0  0  1  1 | 0  1  1  1 |

TABLE II

Manual solution for 4-bit adder without carry

Fig. 26. Output simulation of 4-bit adder without carry

We modified the design of a 4-bit adder circuit and implemented cascading, this step helped reduce the number of transistors needed to perform operations. With this technique, the performance was greatly improved by reducing the complexity of operations and increasing the efficiency of power consumption. Due to the reduced complexity in the design, timing and system performance were also improved, resulting in a faster and more efficient design. This saving in transistors contributed to improving the efficiency of the circuit in general.

1. *32-bit Adder*

The 32-bit adder consists of eight four-bit adders, with the first adder connected without a carry and the rest with one. The first adder processes only the first four bits of the number, whereas subsequent adders use the carry from the previous adder to process the rest. As a result of this design, the computing process can be improved by reducing the complexity of handling the carry in the first adder while at the same time increasing the accuracy of the calculations by connecting the carry between the adders. Also, using sequential carry between the adders helps to speed up the computational processes while maintaining accuracy, which contributes to reducing the time delay resulting from sequential calculations. This distribution in the operations contributes to improving the timing in general and reduces the number of circuits required for each operation, which increases the efficiency of energy consumption and enhances the performance of the circuit as a whole.

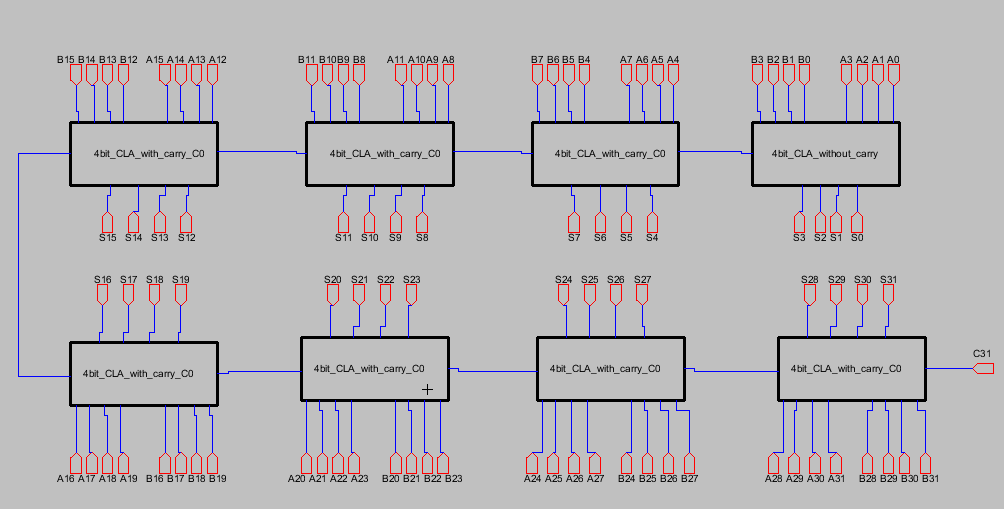
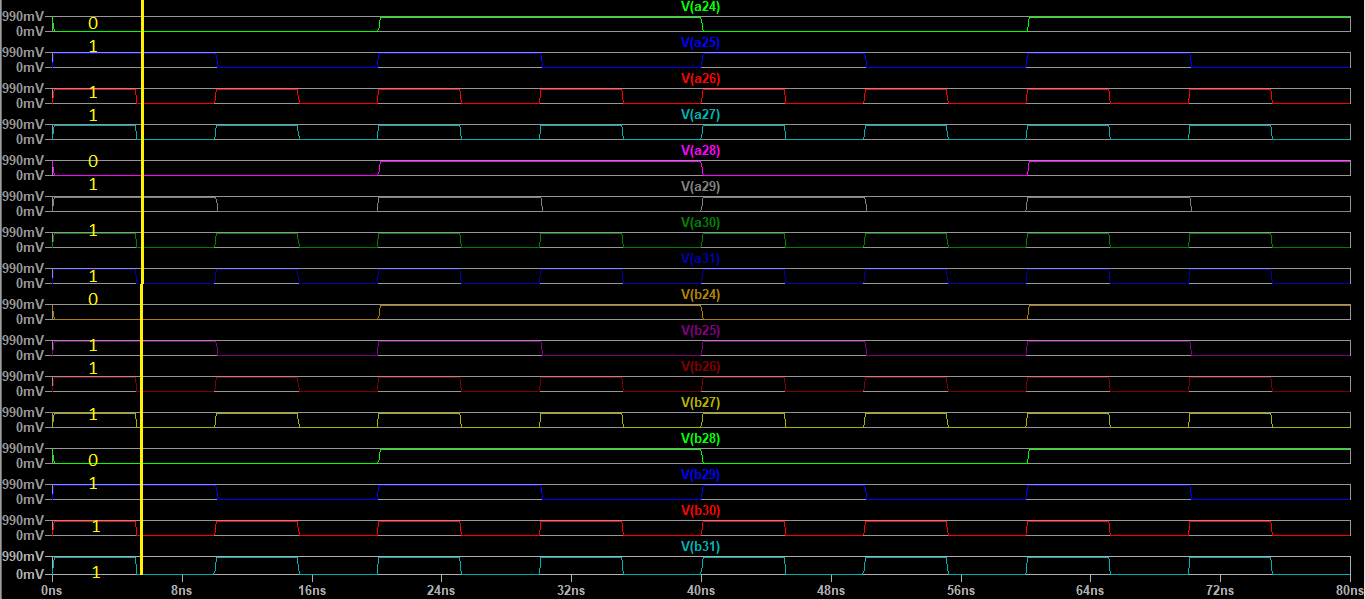
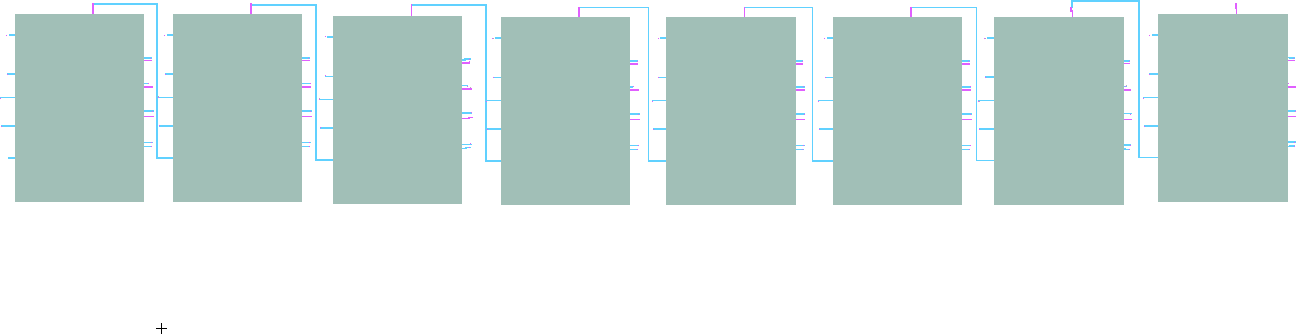


Fig. 27. Schematic of 32-bit adder



Fig. 28. Layout of 32-bit adder

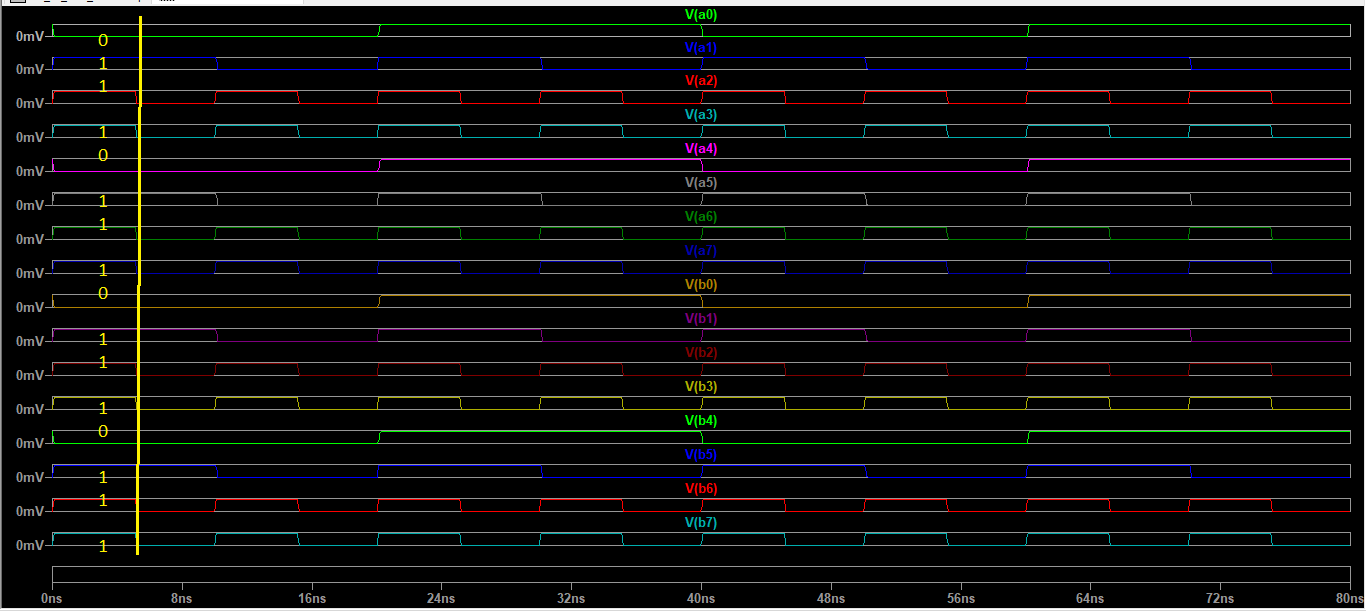


Fig. 29. Output of 32-bit adder (A0–B7)

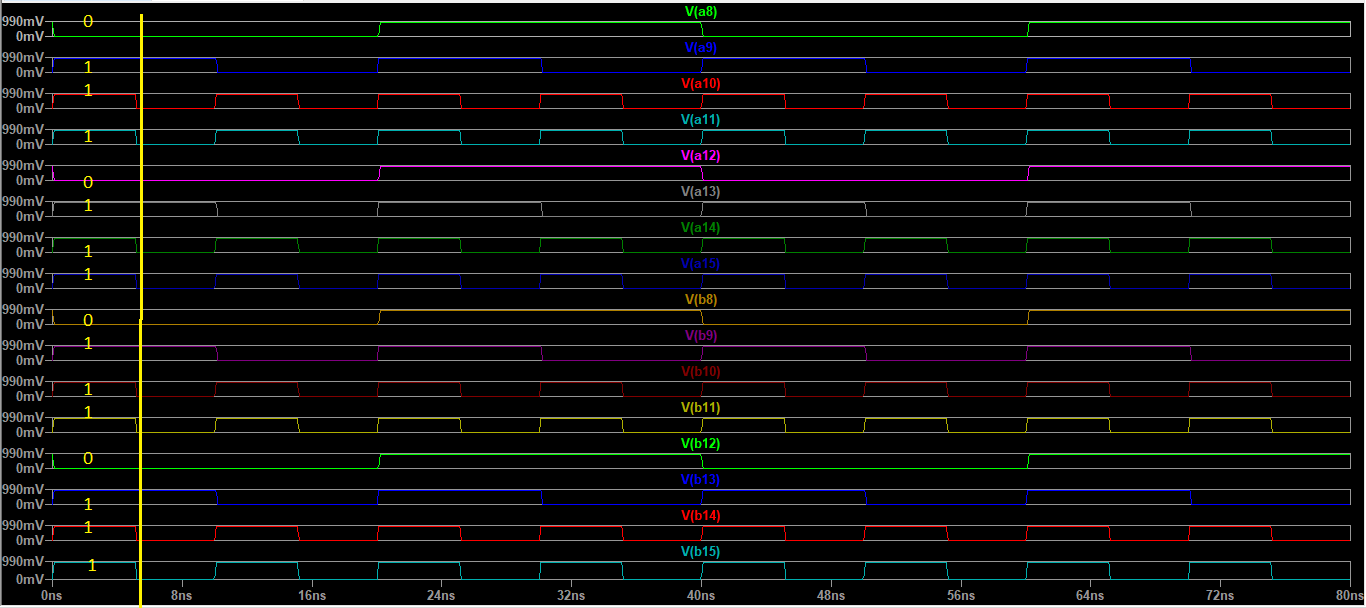


Fig. 30. Output of 32-bit adder (A8–B15)

Fig. 32. Output of 32-bit adder (A24–B31)

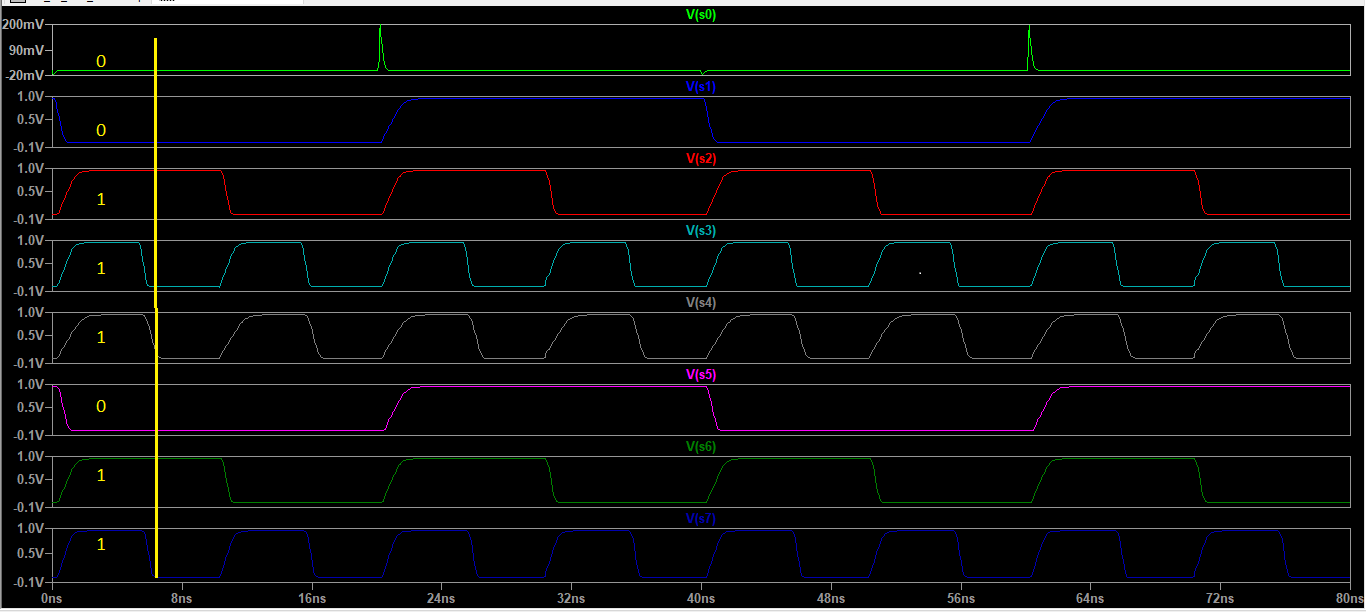


Fig. 33. Output of 32-bit adder (S0–S7)

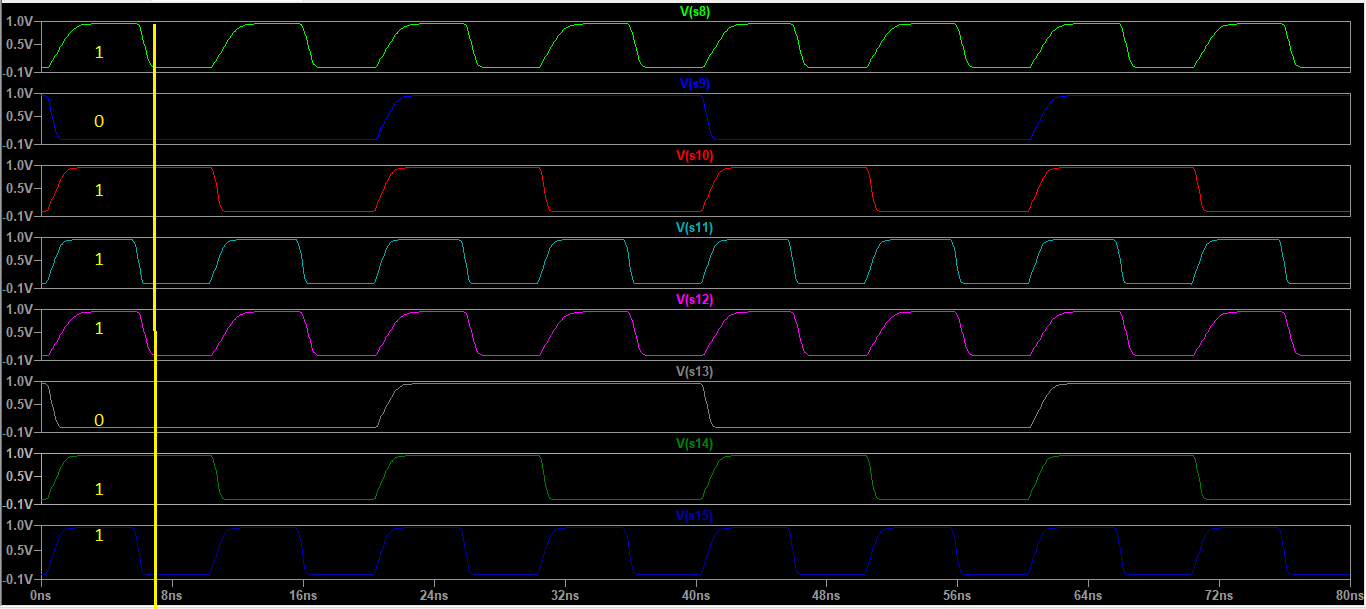


Fig. 34. Output of 32-bit adder (S8–S15)

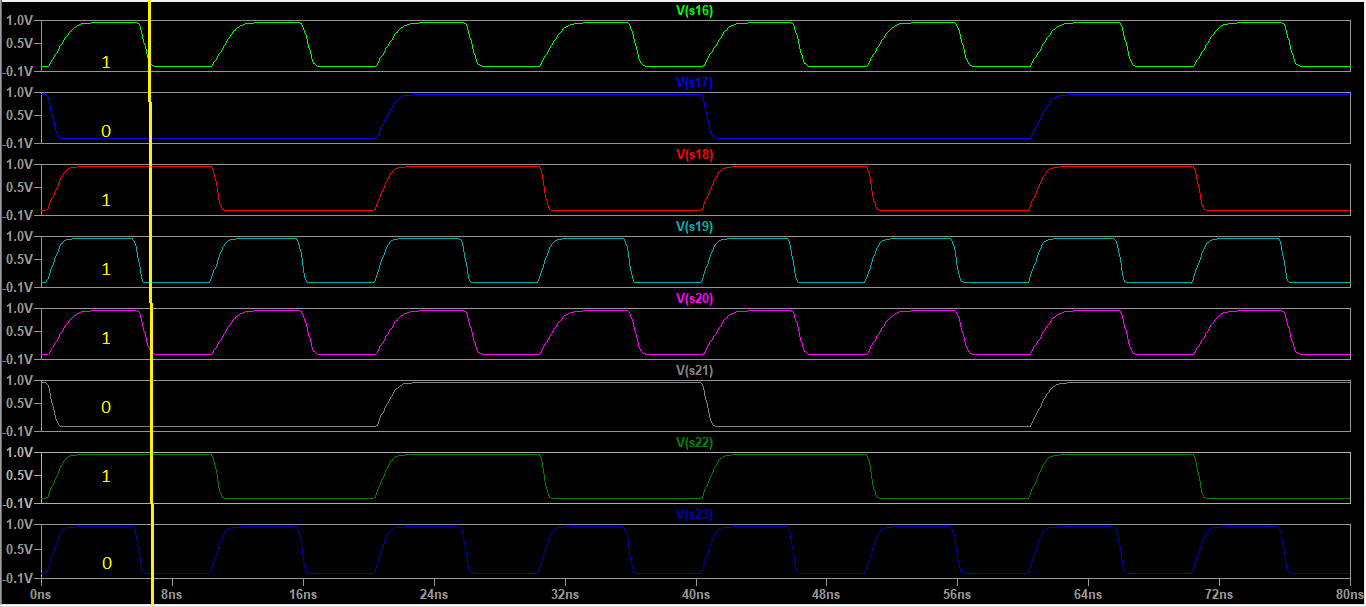


Fig. 35. Output of 32-bit adder (S16–S23)

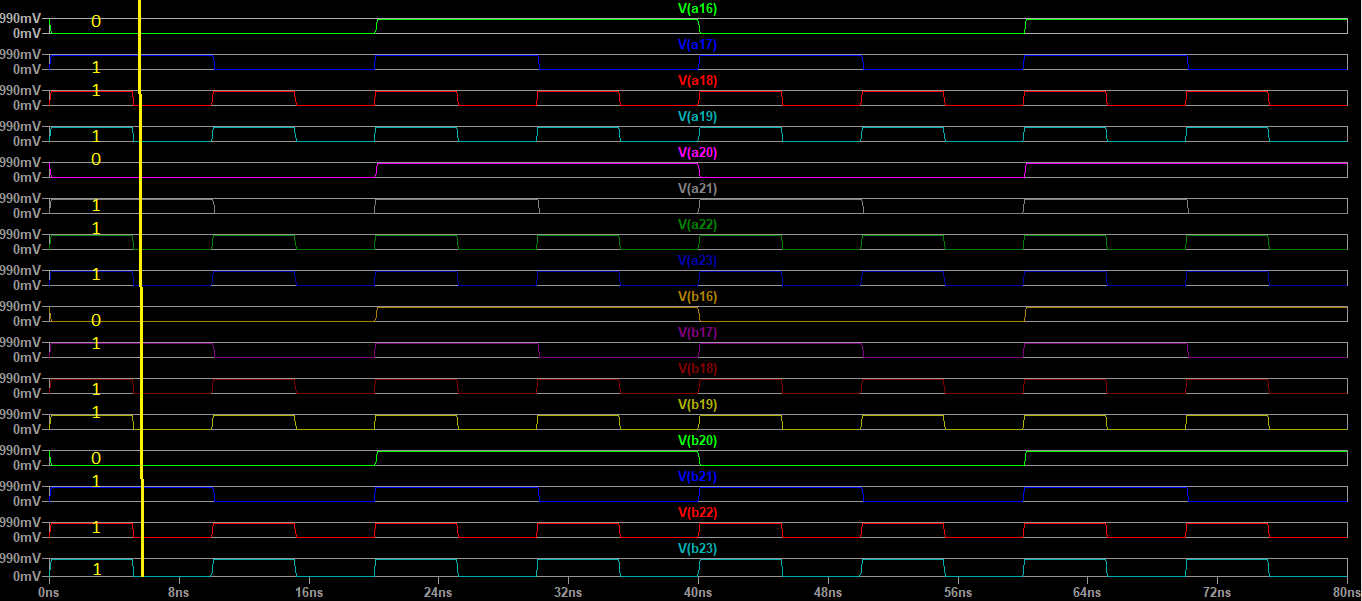
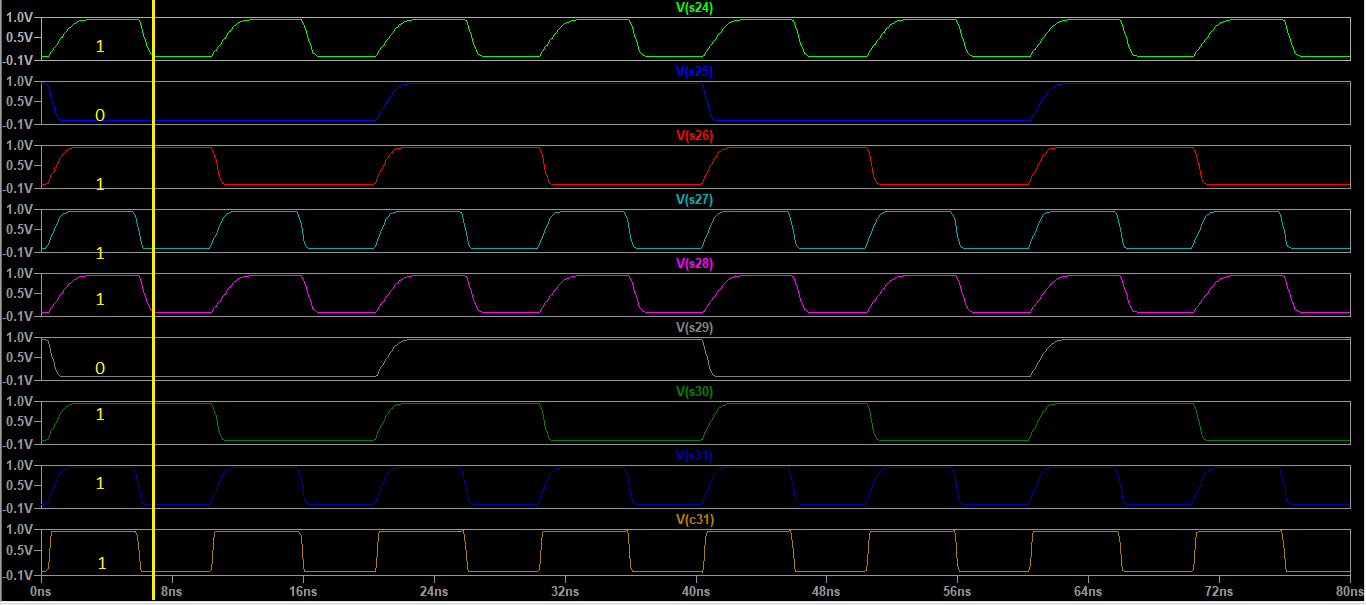
 

Fig. 31. Output of 32-bit adder (A16–B23) Fig. 36. Output of 32-bit adder (S24–S31+C31)

1. Area Estimation

To ensure efficient use of chip space and to achieve high performance and power efficiency, estimating the area of an adder component is an essential part of the design process. Using the CLA (Carry Look-Ahead) DESIGN ENHANCED Adder, we can analyze the power, delay, and area of various designs. In this section, we calculate the area of our adder circuit, which was designed using 22-nm technology.

1. *Transistor Details*

In this design, there are 728 NMOS transistors, each measuring 2L x 10W. Also, this design includes 704 PMOS transistors that measure 2L x 20W dimensions, which ensures that a good balance between performance and efficiency has been achieved.

1. *Area Calculation*

Given the 22nm technology:

* + Length *L* = 2 *×* 22 nm = 44 nm = 0*.*044 *µ*m
  + Width *W* = 10 *×* 22 nm = 220 nm = 0*.*220 *µ*m

The area of a single NMOS transistor is:

1. Comparison

|  |  |  |
| --- | --- | --- |
|  | CLA 8x4 28nm | CLA 8x4 22nm |
| Total Area  Total Delay | 505.96 *µ*m2  1.16 ns | 20.67 *µ*m2  0.44 ns ns |

TABLE III

Comparison between 28nm and 22nm CLA designs

Total Delay: decreased from 1.16 ns to 0.135 ns (88.4% improvement).

Total Area: Decreased from 505.96 µm² to 20.67 µm² (95.9% improvement).

1. Comparison with Existing Algorithms

Compared to traditional adders, this 32-bit Carry Look- Ahead Adder (CLA) is faster and more power efficient than traditional adders. In this section, we compare the proposed design with existing adder architectures, like the Ripple Carry Adder (RCA) and Carry Skip Adder (CSKA) [3].

1. *Speed Improvement*
   1. *Ripple Carry Adder (RCA):* The RCA has a linear carry propagation mechanism that creates significant propagation delays. By contrast, our CLA design reduces the total delay by 88.4%, from 1.16 ns (in 28nm technology) to 0.135 ns

Area

NMOS

= *L × W* = 0*.*044 *µ*m *×* 0*.*220 *µ*m = 0*.*00968 *µ*m2

(3)

(in 22nm technology). With parallel carry computations, we don’t need sequential propagation because we can parallelize

it across multiple stages [1].

For all NMOS transistors, the cumulative area is:

Total AreaNMOS = 728 *×* 0*.*00968 *µ*m2 = 7*.*047 *µ*m2 (4)

For PMOS transistors:

* Length *L* = 2 *×* 22 nm = 44 nm = 0*.*044 *µ*m
* Width *W* = 20 *×* 22 nm = 440 nm = 0*.*440 *µ*m

The area of a single PMOS transistor is:

AreaPMOS = *L × W* = 0*.*044 *µ*m *×* 0*.*440 *µ*m = 0*.*01936 *µ*m2

(5)

For all PMOS transistors, the cumulative area is:

Total AreaPMOS = 704 *×* 0*.*01936 *µ*m2 = 13*.*62 *µ*m2 (6)

The total area of the adder is:

Total Area = Total AreaNMOS + Total AreaPMOS

* 1. *Carry Skip Adder (CSKA):* While the CSKA improves speed by skipping carry propagation in certain groups of bits, it still relies on sequential carry propagation in some cases. Our CLA design outperforms the CSKA by precalculating carry signals for all bit positions, so it’s more consistent and faster [6].

1. *Power Efficiency*
   1. *Ripple Carry Adder (RCA):* RCA is popular for its low power consumption due to its simple design. However, it is not suitable for applications that require a high level of performance. The CLA design achieves a balance between speed and power efficiency by reducing redundant operations and improving transistor utilization [2].
   2. *Carry Skip Adder (CSKA):* The CSKA consumes more power due to its additional logic for skipping carries. Our CLA design uses fewer transistors and optimizes the carry calculation to reduce power consumption.

= 7*.*047 *µ*m2 + 13*.*62 *µ*m2

= 20*.*67 *µ*m2

1. Total Delay

(7)

1. *Area Efficiency*
   1. *Ripple Carry Adder (RCA):* The RCA has the smallest area footprint among traditional adders, but its slow speed limits its applicability. The proposed CLA design achieves a

The total delay is the sum of the bitwise addition delay and the carry propagation delay:

Total Delay = Bitwise Addition Delay + Carry Prop Delay

= 320 ps + 120 ps

= 440 ps

(8)

95.9% reduction in area compared to the conventional CLA in 28nm technology, making it more suitable for modern integrated circuits where space is a critical factor [4].

* 1. *Carry Skip Adder (CSKA):* The CSKA requires additional logic for carry skipping, which increases its area footprint. The our CLA design reduces area by using fewer transistors and optimizing the layout of the adder [5].

1. *Energy Efficiency*

CLA offers a 65.4% reduction in propagation delay com- pared to the Ripple Carry Adder (RCA), which is known for its low power consumption but suffers from significant propagation delays.

1. *Changes and Improvements*

Uniform-sized CLA Modules: The proposed design uses uniform-sized CLA modules to simplify the design and reduce power consumption. This approach allows better control over the carry computation process, leading to improved speed and efficiency [1].

MGDI (Modified Gate Diffusion Input) technology, requiring fewer transistors for XOR gates to save more power consumption and a small footprint. Larger 4-bit CLA units are cascaded in a hierarchical design to form the larger adder of size 32 bits. The modular nature of the adder allows us to scale this design and makes it easier for testing and optimization [6].

1. Conclusion

This paper introduced a novel architecture of CLA to achieve high-speed and low power consumption digital systems. We have designed a 32-bit adder that reduces as well the propagation delays and power consumption. By taking advantage of the uniform size of CLAs and the modular approach. The hierarchical structure of the unit, consisting of smaller 4-bit CLAs, allowed for parallel computing of the carry signals, which not only accelerated the addition process but also reduced the duplicate operations, resulting in better power efficiency.

XOR gates were optimized using techniques like MGDI (Modified Gate Diffusion Input), and so that contributed to the reduction in the number of transistors, hence the power consumption chip area, while keeping performance high. The 32- bit adder design presented better timing, lower cost, and higher energy efficiency, which makes it apt for high-performance computing applications and digital processors.

Detailed simulation and manual calculation results verified the precision and speed of the proposed design. The area estimation in 22nm technology confirmed the compactness of the design, with a total area of 20.67 µm², while the total delay was measured at 135 ps, showcasing the adder’s high-speed performance.

In conclusions, this paper concludes the necessity of modern and innovative design approaches for solving the issues in digital systems that are being used today, like power efficiency, fault tolerance speedup, and space optimization. These results together indicate that the proposed CLA design is scalable for future digital processors and high-performance computing systems, enabling much more advanced yet energy-efficient electronic systems.

1. References
2. Balasubramanian, P., Maskell, D. L. (2024). A new Carry Look-Ahead Adder architecture optimized for speed and energy. Electronics, 13(18), 3668. https://doi.org/10.3390/electronics13183668
3. CMOS VLSI Design: A Circuits and Systems perspective: Weste, Neil, Harris, David: 8601400041482: Amazon.com: Books. (n.d.). [https://www.amazon.com/CMOS-VLSI-Design-Circuits-](http://www.amazon.com/CMOS-VLSI-Design-Circuits-)

Perspective/dp/0321547748

1. Oklobdzija, V., Villeger, D. (1995). Improving multiplier design by using improved column compression tree and optimized final adder in CMOS technology. [https://www.semanticscholar.org/paper/Improving-](http://www.semanticscholar.org/paper/Improving-) multiplier-design-by-using-improved-tree-Oklobdzija- Villeger/d7372e2271082085a2a6404ec77741d7750e4e3a.
2. Weste, N., & Eshraghian, K. (1993). *Principles of CMOS VLSI Design: A Systems Perspective* (2nd ed.). Addison-Wesley.
3. Hennessy, J. L., & Patterson, D. A. (2017). *Computer Architecture: A Quantitative Approach* (6th ed.). Morgan Kaufmann.
4. Harris, D. (2003). A taxonomy of parallel prefix networks. In *Proc. IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 221-228. https://doi.org/10.1109/ICCD.2003.1240908.
5. Alioto, M., & Palumbo, G. (2005). *Model and Design of Bipolar and MOS Current-Mode Logic: CML, ECL, and SCL Digital Circuits*. Springer.