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 increase 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.

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 of these issues is **power efficiency**. One of the main goals in contemporary electronic circuit design is for engineers to strike the perfect balance between powerful performance and power consumption. Another challenge is making chips more **space-efficient**. 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.

To meet and overcome these challenges, in this project we have created a **Carry Look-Ahead Adder (CLA)** that improves both speed and power efficiency. Adders are fundamental building blocks of digital systems, responsible for the computations essential to processors, embedded systems, and high-performance computing. However, traditional adders, such as **Ripple Carry Adders (RCA)**, suffer from **propagation delays**, where each bit must wait for the previous stage's carry output, resulting in slower performance and higher power consumption. The RCA, while simple and area-efficient, is limited by its linear carry propagation, which makes it unsuitable for high-speed applications. In contrast, the proposed CLA design reduces these delays by precomputing carry signals, enabling parallel computation and significantly improving speed.

Therefore, our project takes advantage of **uniform-sized CLAs** to build a 32-bit adder. With this modular approach, the carry computations are parallelized to improve speed while minimizing duplicate operations to reduce power consumption. This design not only addresses the limitations of traditional adders like the RCA but also provides a more efficient solution for modern digital systems.

In this paper, we will work on creating adders that are not only faster but also smaller and more energy-efficient. Much of an adder's performance depends on its design, which has a direct impact on things like speed, power consumption, and how efficiently the chip uses its space.

Combining smaller adders in a hierarchical fashion is an efficient way to build large adders. For example, a 32-bit adder can be created by connecting eight 4-bit adders. The first 4-bit adder operates without a primary load, and its output load is passed to the next adder. This process continues through the remaining units to ensure that all the bits are added correctly. Because each adder must wait for the load from the previous one, this introduces delays. To avoid this, **Carry Look-Ahead (CLA)** methods are used to anticipate the load values in advance and reduce the delays.

When using smaller 4-bit units, this hierarchical design becomes easier to implement, test, and optimize for both speed and power efficiency. In modern integrated circuits, where power and area efficiency are key factors, this modular approach is particularly useful.

EXISTING SYSTEM AND SIMULATION COMPARISON :

We designed a system consisting of many small circuits. In this section, we will talk about the working principle of each small circuit and how it will be improved in terms of power, area, memory, and number of transistors.

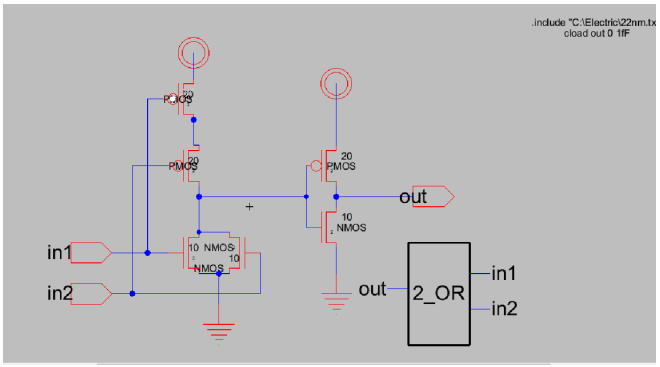
1. OR GATE :

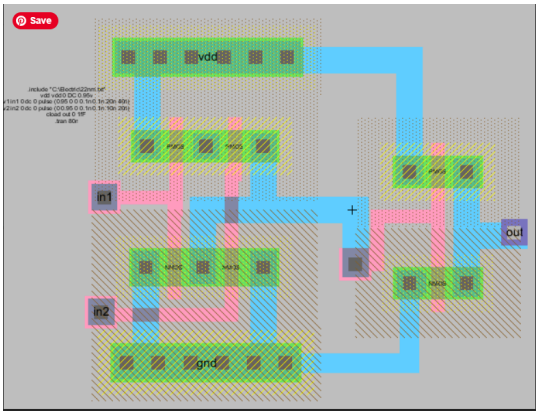
   OR gate is one of the basic logic gates that performs the "OR" operation. It produces an output of 1 if at least one of the inputs is 1, and if all the inputs are zero, the output is zero. OR gates are used in many applications such as control and comparison, and can support an unlimited number of inputs.We 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 will discuss each one separately.

   We designed the gates using NMOS and PMOS transistors with the transistor sizes adjusted so that the NMOS was 10 Ω and the PMOS was 20 Ω.

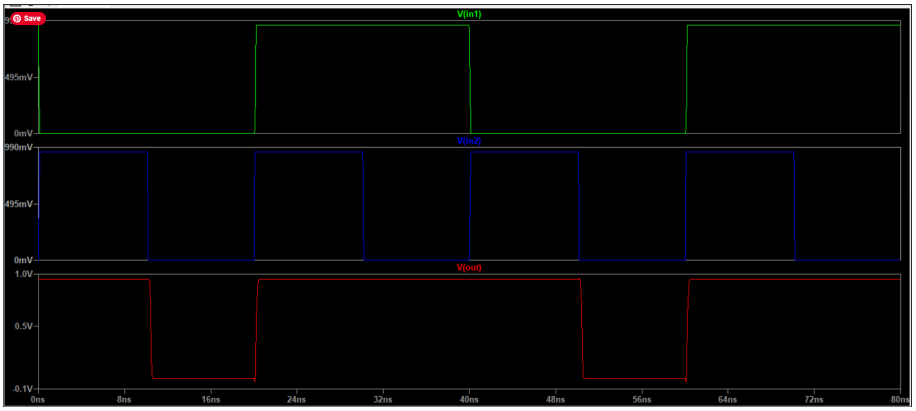
   1.1. 2-input OR gate:

   schematic & layout :



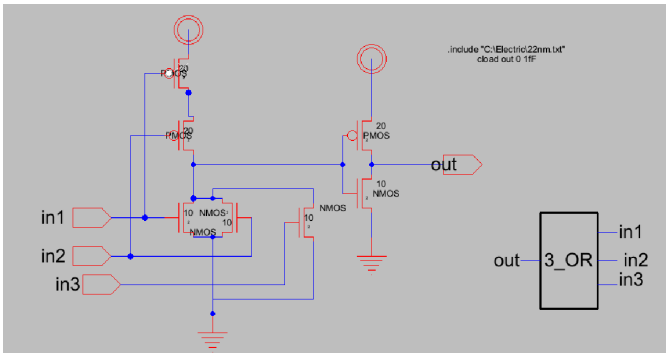


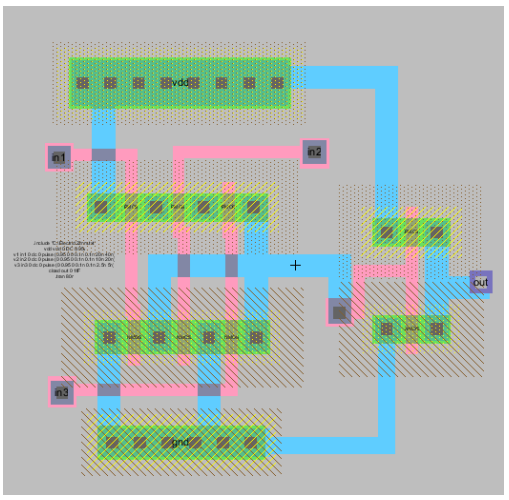
output:



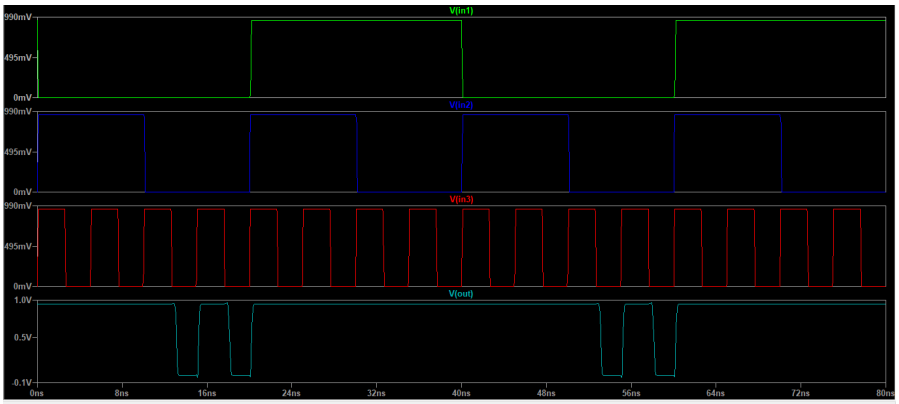
1.2. 3-input OR gate :

schematic & layout :



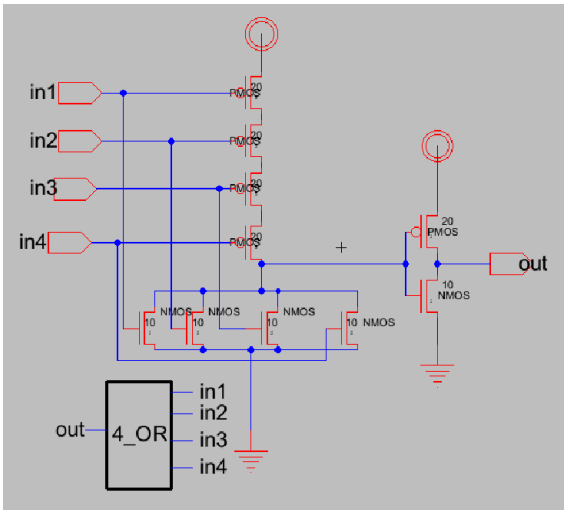


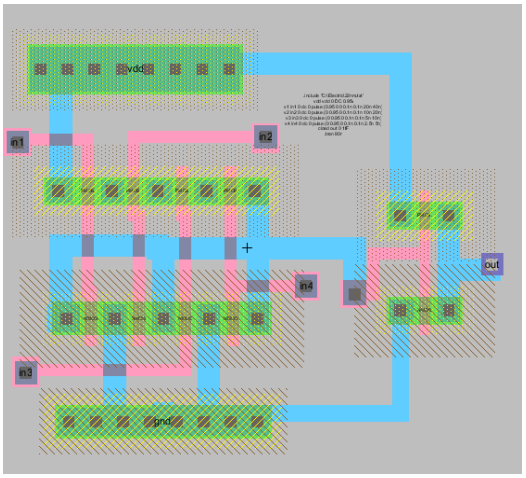
output:



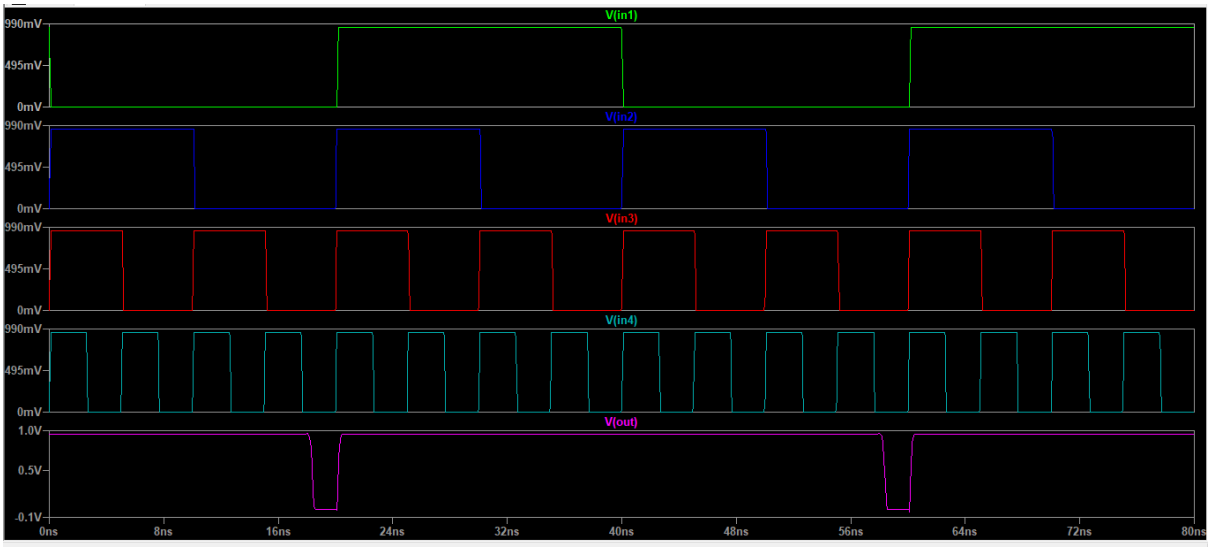
1.2. 4-input OR gate :

schematic & layout :





output:



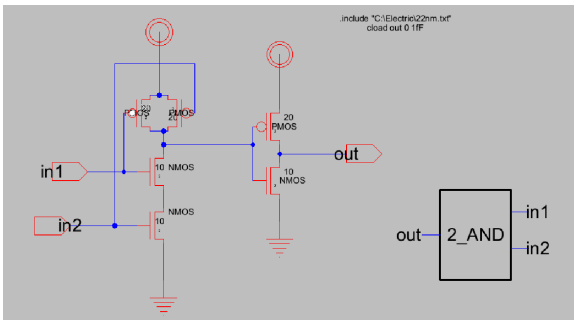
2.And Gate :

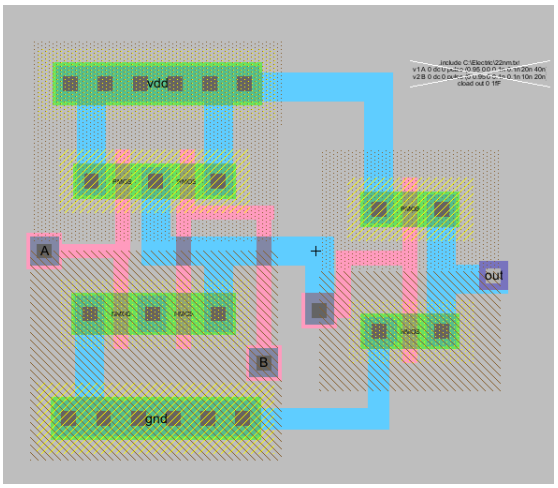
Logic gates that implement the logical operation "AND" include AND gates. When all the inputs are 1, this gate will output 1, but if any of the inputs are zero, the output will be 0. In this project, AND gates are built using NMOS and PMOS transistors, where the size of the transistors is specified such 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 3 inputs, and an OR gate with 4 inputs. Here we will discuss each one separately.

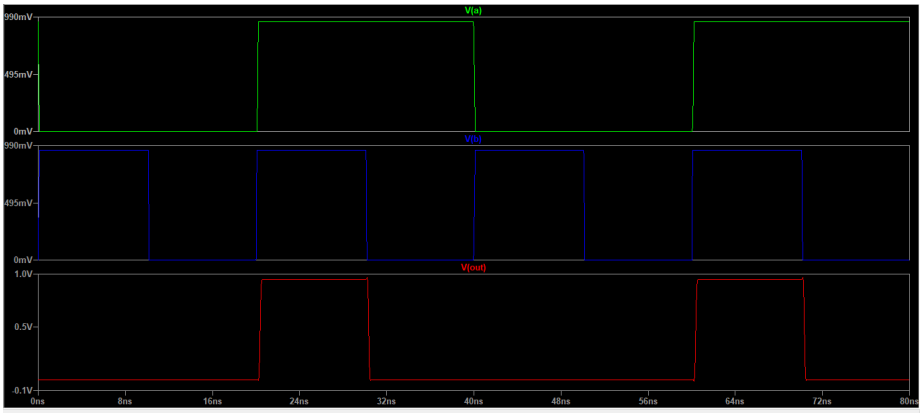
2.1. 2-input And gate:

schematic & layout :

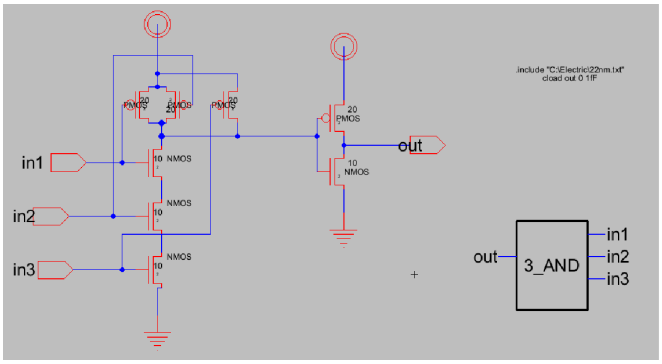


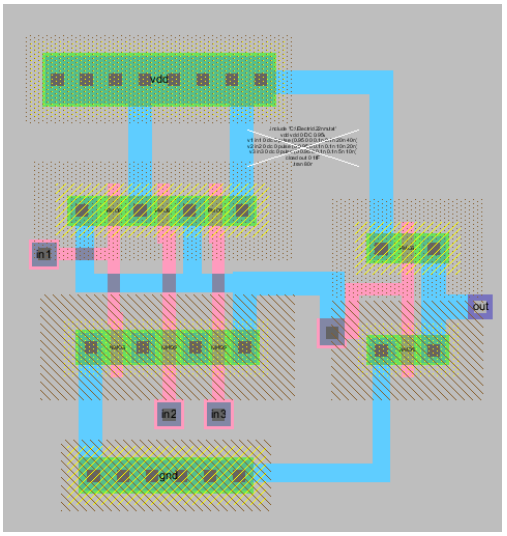


output:

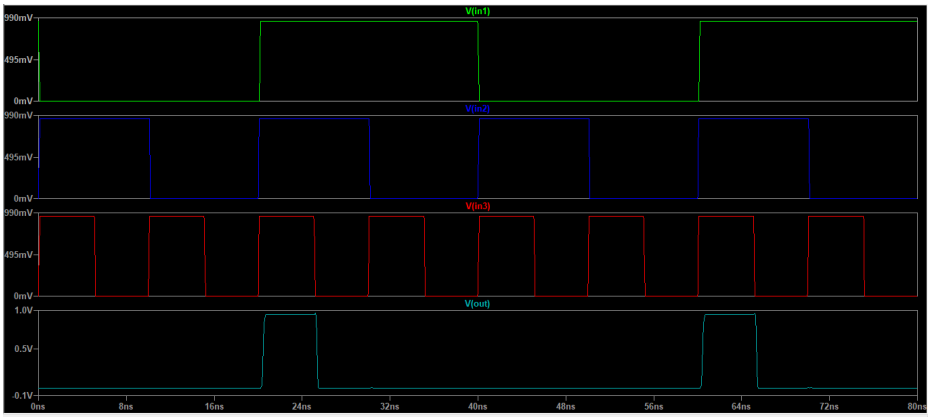


2.2. 3-input AND gate:
schematic & layout :

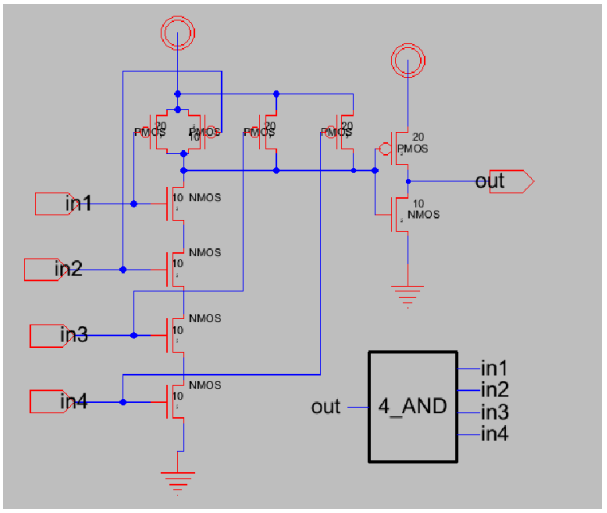


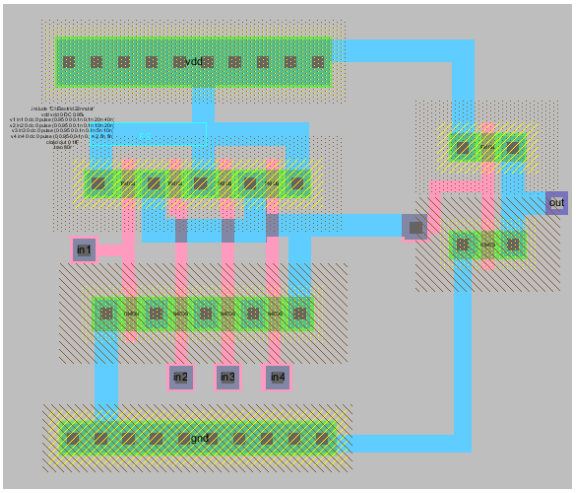


output:

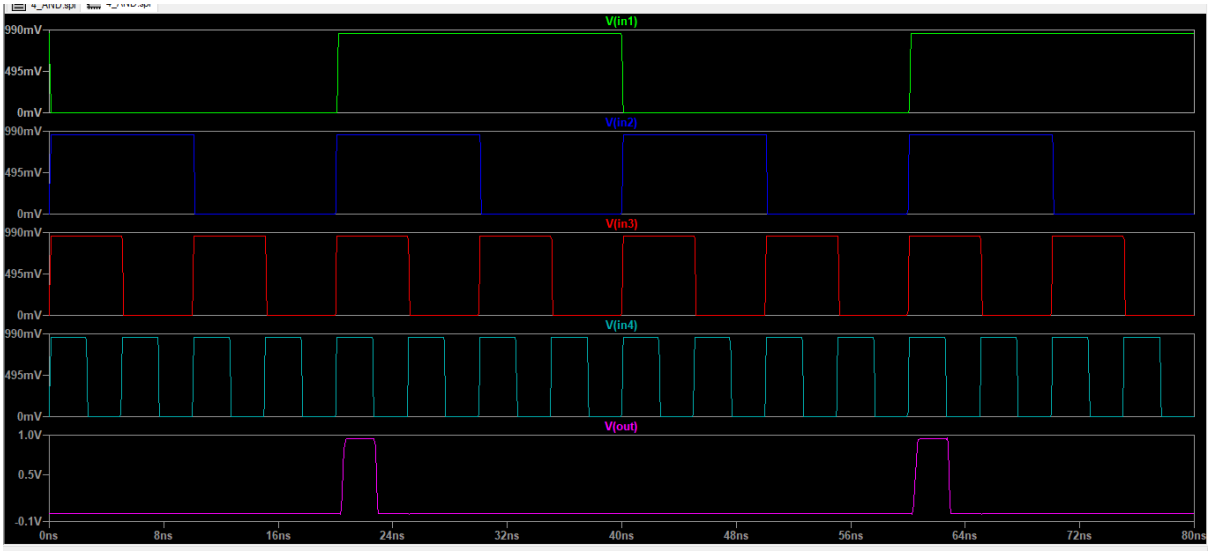


2.3. 4-input AND gate:
schematic & layout :


PMOS
NMOS
in1
in2
in3
in4
out
4_AND

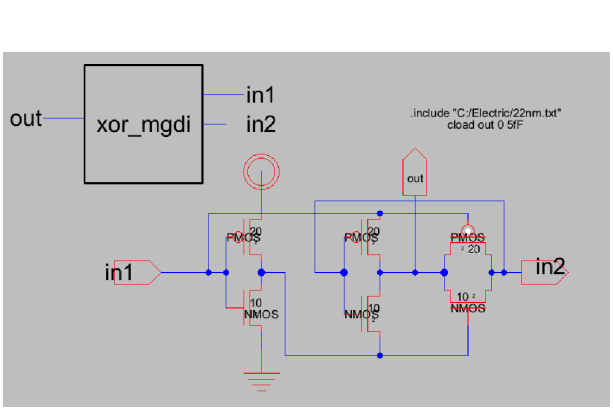

vdd
gnd
out
in1
in2
in3
in4

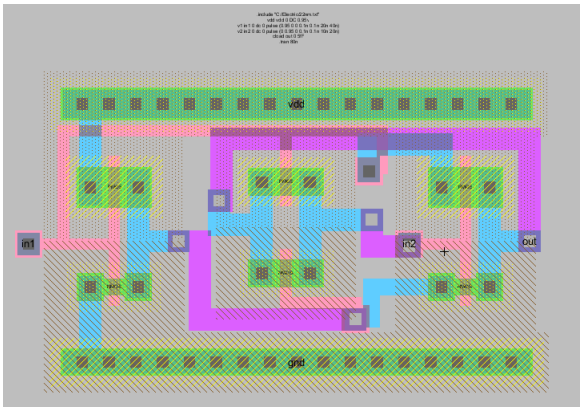
output:

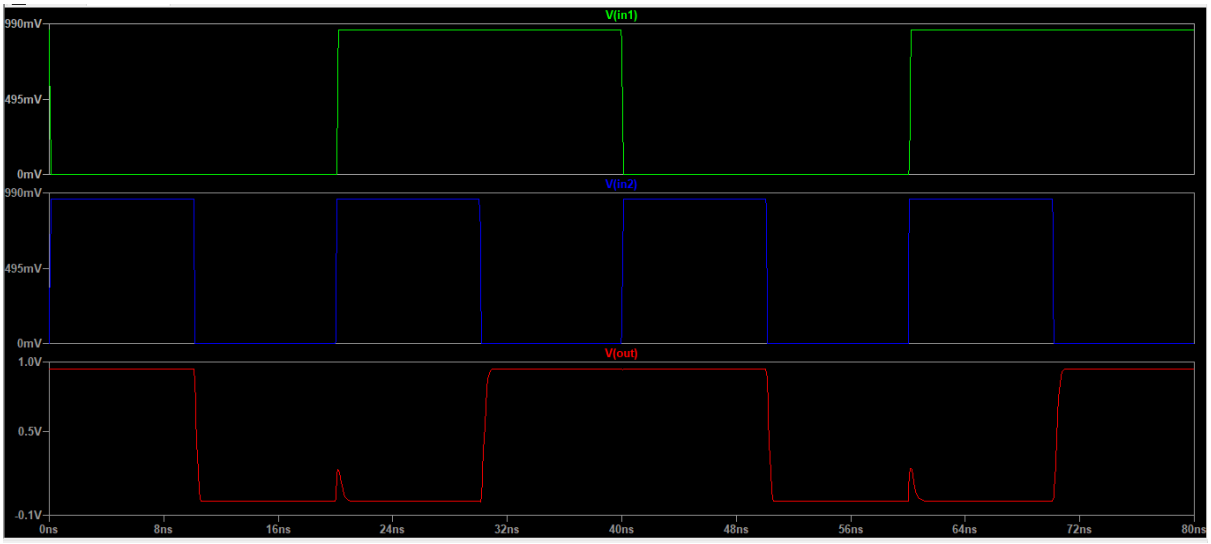

V(in1)
V(in2)
V(in3)
V(in4)
V(out)
990mV
495mV
0mV
1.0V
0.5V
-0.1V
0ns
8ns
16ns
24ns
32ns
40ns
48ns
56ns
64ns
72ns
80ns

3- 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 to reduce 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.



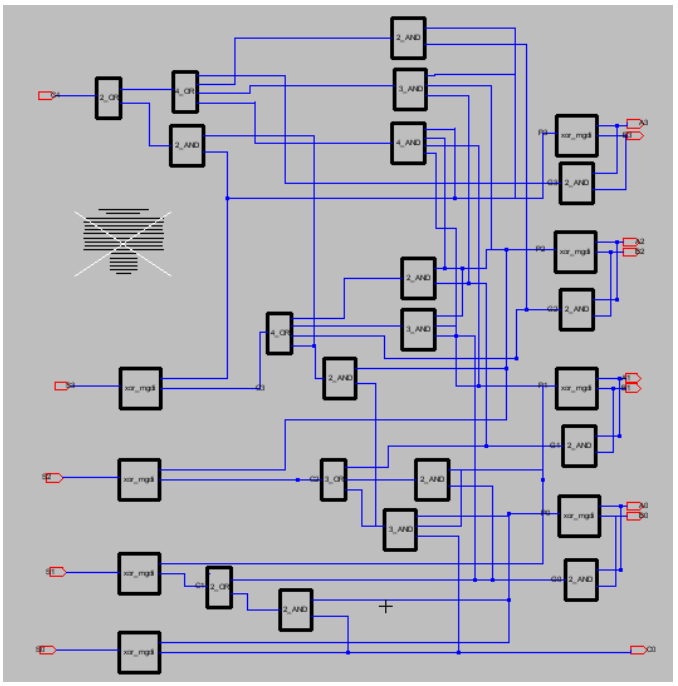




4. 4-bit adder with carry :

In this design, we build a 4-bit adder using a set of logic gates (XOR, AND, OR), focusing on using the XOR gate (with MGDI technology) to build the process more efficiently. The design is based on distributing the work between logic gates as needed for each step of the addition, where we use the XOR gate (MGDI) to calculate the partial sum S in each stage, while the AND (2-bit, 3-bit, 4-bit) is used to calculate the partial carry (Partial Carry) in each stage, based on the correlation of the inputs. As for OR (with the same number of AND

gates): the signals resulting from the AND gates are summed to generate the final output carry C out.

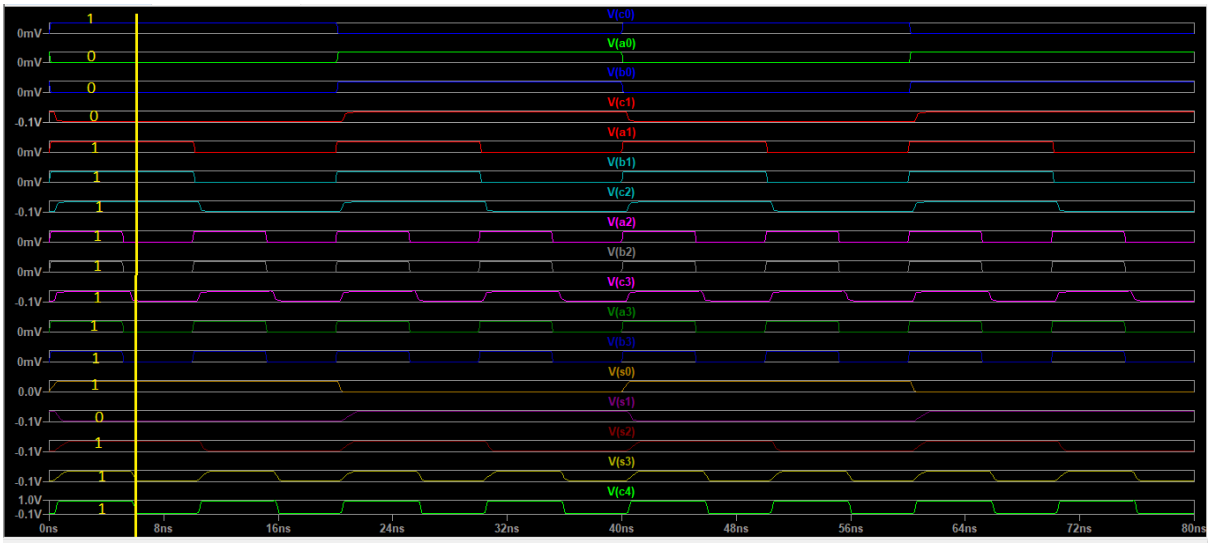


Equations:
$Sk=(Ak \oplus Bk) \oplus Ck$
$Ck+1=(Ak \cdot Bk)+((Ak \oplus Bk) \cdot Ck)$

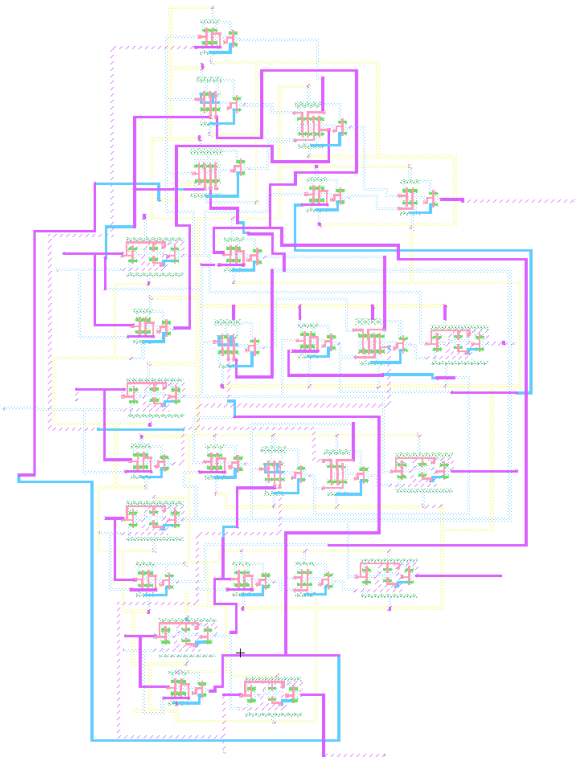
To verify the validity of the system, we made manual calculations and compared them with the result we obtained from the simulation.
Manual solution:

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

output simulation :

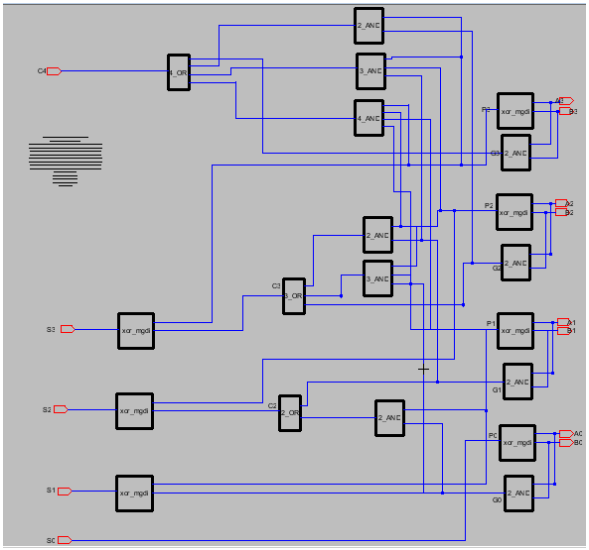


The accuracy of the digital design (using XOR, AND, and OR) was confirmed by the consistency of the results between the two methods.



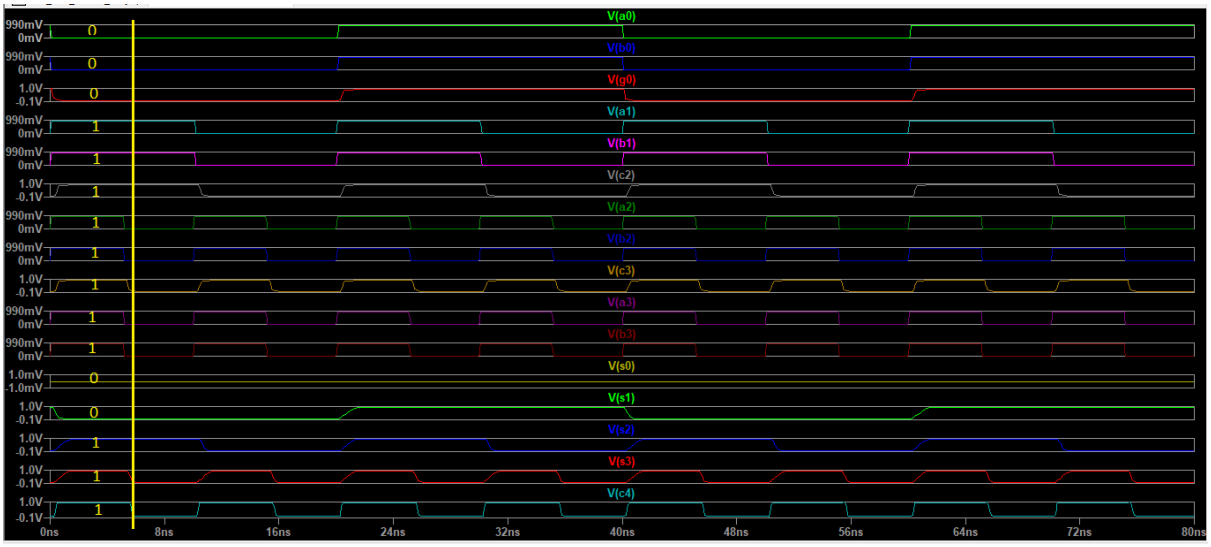
5. 4- bit adder without carry :

We designed it as a 4-bit adder with carry but here without taking the carry into account. This circuit relies on using basic logical operations like AND, OR, and XOR to generate the result. Since without a carry, the goal is just to add the corresponding bits and produce a result containing 4 bits, ignoring the carry resulting from the arithmetic operations, so the result is stored in the added bits only.



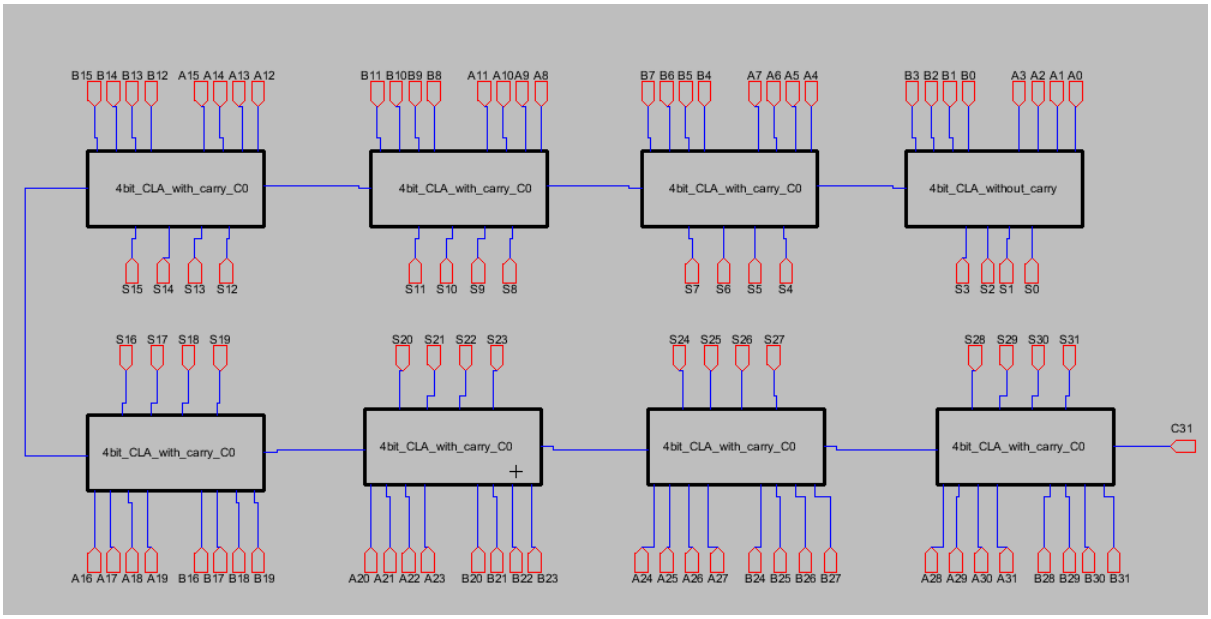
Manual solution:

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



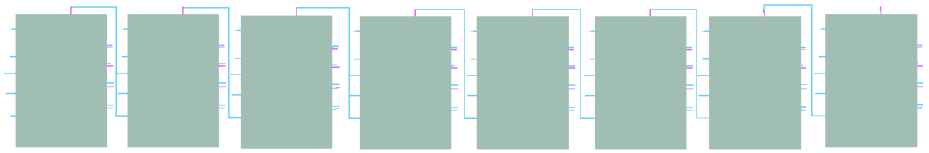
We modified the design of a 4-bi adder circuit and implemented cascading, this step helped reduce the number of transistors needed to perform operations. With this technique, the performance was greatly improved by reducing the complexity of operations and increasing the efficiency of power consumption. 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

32-bit adder:



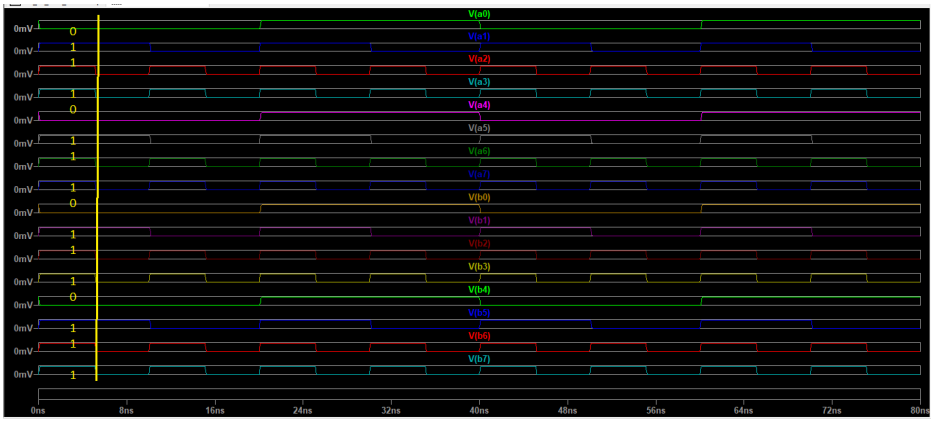
The 32-bit adder consists of eight four-bit adders, with the first adder connected without a carry and the rest with one. The first adder processes only the first four bits of the number, whereas subsequent adders use the carry from the previous adder to process the rest. As a result of this design, 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

32-bit layout:

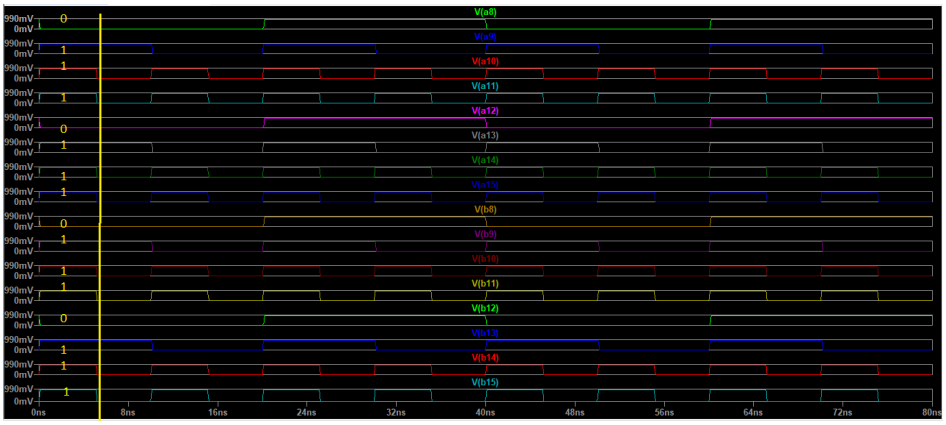


32-bit adder output:

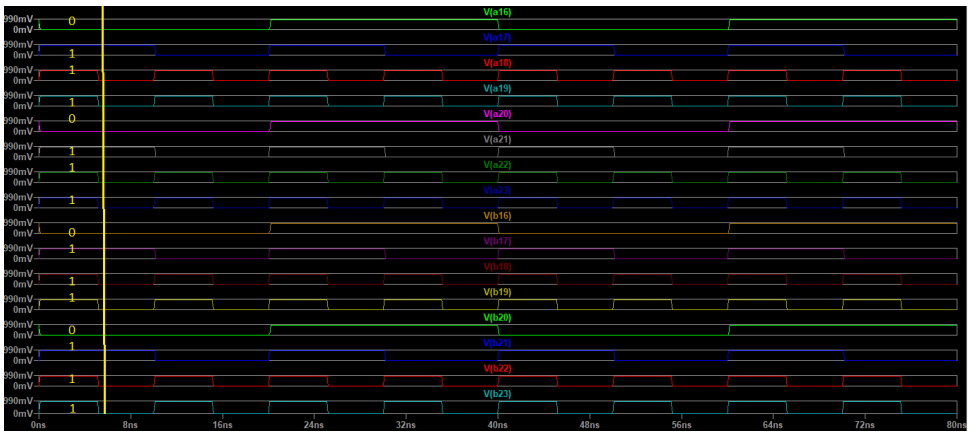
A0--B7



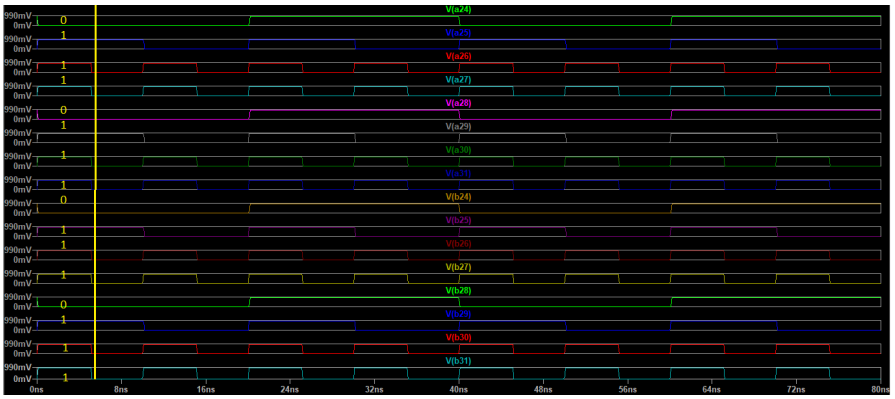
A8–B15



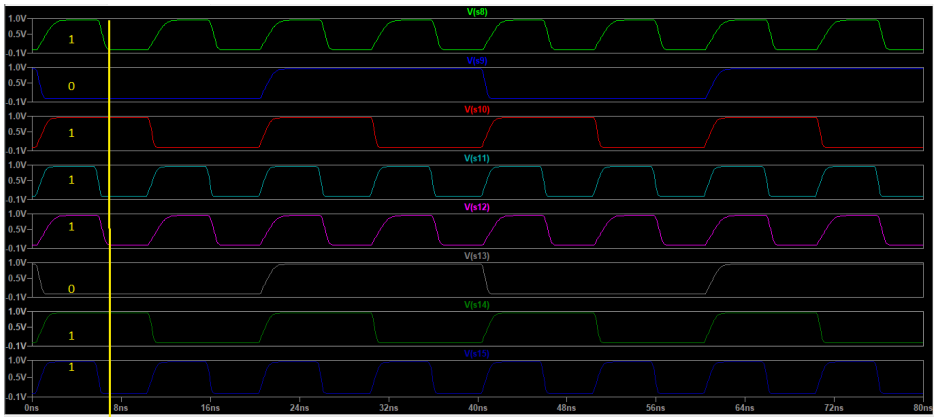
A16–B23



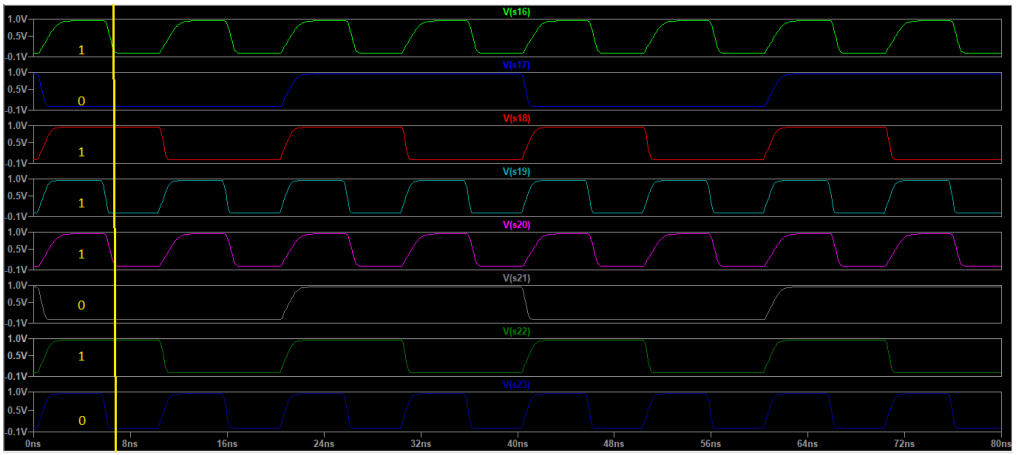
A24–B31



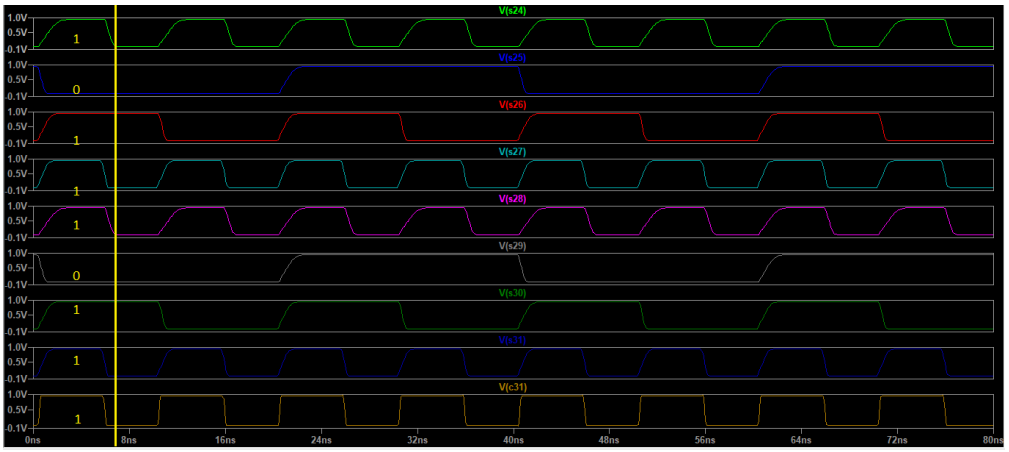
S8–S15



S16–S23


V(s16)
V(s17)
V(s18)
V(s19)
V(s20)
V(s21)
V(s22)
V(s23)
1.0V
0.5V
-0.1V
0ns
8ns
16ns
24ns
32ns
40ns
48ns
56ns
64ns
72ns
80ns

S24–S31+C31


V(s24)
V(s25)
V(s26)
V(s27)
V(s28)
V(s29)
V(s30)
V(s31)
V(c31)
1.0V
0.5V
-0.1V
0ns
8ns
16ns
24ns
32ns
40ns
48ns
56ns
64ns
72ns
80ns

Refering to the CLA DESIGN ENHANCED Adder we can see how the
power, delay and area for the various design
Area:
Area Estimation of Adder Design in 22nm Technology
As part of the design process, estimating the area of an Adder component is crucial to ensuring efficient use of chip space and achieving high performance and power efficiency. We calculate our adder circuit's area, which was created using 22nm technology, in this section.

Transistor Details:

In this design, there are 728 NMOS transistors each measuring 2L x 10W. Additionally, the design contains 704 PMOS transistors with dimensions of 2L x 20W per transistor, ensuring a good balance between performance and efficiency.

2. Area Calculation:

a. NMOS Transistors: Given the 22nm technology:
Length L = 2 x 22 mm = 44 mm = 0.044 um

Width W = 10 x 22 mm = 220 mm = 0.220 um

The area of a single NMOS transistor is:

Area NMos = L x W = 0.044 um x 0.2200 um = 0.00968 um^2

For all NMOS transistors, the cumulative area is:

Total Area NMos = 728 x 0.00968 um^2 = 7.047 um^2

b. PMOS Transistors: Given the 22nm technology:
Length L = 2 x 22 mm = 44 mm = 0.044 um

Width W =20 x 22 mm =440 mm = 0.440 um

The area of a single PMOS transistor is:

Area PMos = L x W = 0.044 um x 0.440 um = 0.01936 um^2

For all PMOS transistors, the cumulative area is:

Total Area PMos = 704 x0.01936 um^2 = 13.62 um^2

Total Area = Total Area NMOS + Total Area PMOS

Total Area = 7.047 um^2 + 13.622 um^2 = 20.67 um^2

Total Delay:

- The total delay is the sum of the bitwise addition delay and the carry propagation delay:
- Total Delay=Bitwise Addition Delay+Carry Propagation Delay
- Total Delay=15ps+120ps=135ps

Comparison :

	CLA 8x4 28nn	CLA 8x4 22nm
Total Area	1.16 ns	0.135 ns
Total Delay	505.96 μm²	20.67 μm²

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).

Comparison with Existing Algorithms

This 32-bit Carry Look-Ahead Adder (CLA) shows significant improvements in both speed and power efficiency over traditional adders. Below, we compare the proposed design with existing adder architectures, such as the Ripple Carry Adder (RCA) and Carry Skip Adder (CSKA)

Speed Improvement:

RCA (Ripple Carry Adder): The RCA has a linear carry propagation mechanism that creates significant propagation delays. In contrast, the our CLA design reduces the total delay by 88.4%, from 1.16 ns (in 28nm technology) to 0.135 ns (in 22nm technology). Through parallelizing carry computations across multiple stages, we're able to eliminate the need for sequential carry propagation.

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.

Power Efficiency:

RCA: The RCA is known for its low power consumption due to its simple design. Although it's slow, it's not good for high-performance applications. By minimizing redundant operations and optimizing transistor usage, our CLA design achieves a balance between speed and power efficiency.

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

Area Efficiency:

RCA: The RCA has the smallest area footprint among traditional adders, but its slow speed limits its applicability. The proposed CLA design achieves a 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.

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.

Energy Efficiency:

The proposed CLA design achieves a 20.2% reduction in energy consumption compared to the Kogge-Stone Adder (KSA), which is known for its high speed but also high power consumption. By optimizing PDP, a key metric for digital circuit efficiency, we can improve energy efficiency.

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

The proposed design simplifies the design and reduces power consumption by using uniform-sized CLA modules. By controlling the carry computation process better, speed and efficiency are improved.
The Modified Gate Diffusion Input (MGDI) technology reduces the number of transistors required for XOR gates, resulting in lower power consumption and a smaller footprint.
In the 32-bit adder, smaller 4-bit CLA units are cascaded to form the larger adder through a hierarchical design. The modular design improves scalability and makes the adder easier to test and optimize. efficiency.

Conclusion:

In this project, we presented an enhanced design of a Carry Look-Ahead Adder (CLA) aimed at improving speed and power efficiency in digital systems. By utilizing uniform-sized CLA units and a modular approach, we successfully designed a 32-bit adder that significantly reduces propagation delays and power consumption. The hierarchical structure of the adder, composed of smaller 4-bit CLA units, allowed for parallel computation of carry signals, which not only accelerated the addition process but also minimized redundant operations, leading to better power efficiency.

The use of advanced techniques such as MGDI (Modified Gate Diffusion Input) for XOR gates further contributed to reducing transistor count, power consumption, and chip area while maintaining high performance. The 32-bit adder design demonstrated improved timing, reduced complexity, and enhanced energy efficiency, making it suitable for applications in high-performance computing and digital processors.

Through detailed simulations and manual calculations, we validated the accuracy and efficiency of the proposed design. The area estimation in 22nm technology confirmed the compactness of the design, with a total area of 20.67 μm^2, while the total delay was measured at 135ps, showcasing the adder's high-speed performance.

In conclusion, this project highlights the importance of innovative design techniques in addressing the challenges of modern digital systems, such as power efficiency,

speed, and space optimization. The proposed CLA design offers a scalable and efficient solution for future digital processors and high-performance computing devices, paving the way for more advanced and energy-efficient electronic systems.