BIRZEIT UNIVERSITY
Electrical and Computer Engineering Department
ENCS3330
DIGITAL INTEGRATED CIRCUITS
Project - 4x4 Enhanced Pipeline multiplier

Mariam Hamad – 1200837
Leena Affouri - 1200335
Nirmeen Alsheikh – 1200200

Summer Sem 2022-2023

I. ABSTRACT

This project focuses on the development of a very efficient 4x4-bit folded pipeline multiplier. Using a modified hierarchical design method, this multiplier delivers exceptional processing performance while drastically decreasing power consumption, hardware resources, and necessary size. It performs well in practical testing using an FPGA implementation, costing only 0.2 mW, saving over 20% of the space, and exhibiting a low latency of 3 ns from input to output utilizing a 22 nm process library.

Acronyms and Abbreviations

1. DEFL - Default Length
2. DEFW - Default Width
3. LIMPTS - Limit Points
4. ITL5 - Iterative Level 5
5. RELTOL - Relative Tolerance
6. ABSTOL - Absolute Tolerance
7. VNTOL - Voltage Tolerance
8. LVLTIM - Level Timing
9. LVLCOD - Level Coding
10. NMOS - N-channel Metal-Oxide-Semiconductor
11. PMOS - P-channel Metal-Oxide-Semiconductor
12. VIN - Input Voltage
13. VDD - Supply Voltage
14. PWL - Piecewise Linear
15. CLOAD - Capacitance Load

Keywords: Implementation of multiplier hardware, pipeline multiplier, design and implementation of all components to construct the 4x4 pipelined multiplier in schematic and layout to each of them and to all system.

II. INTRODUCTION

A 4x4 Enhanced Pipeline Multiplier is a specialized piece of digital hardware that plays an important function in computer arithmetic. It is especially critical in microprocessors and digital signal processors, where quick multiplication is required. This type of multiplier is designed to speed up the multiplication process by handling 4-bit binary integer numbers. It optimizes how the hardware is used in this way, resulting in much faster multiplication. The "4x4" component indicates that it is built for 4-bit inputs and produces 4-bit output, making it ideal for many digital systems that require speedy and efficient multiplication, such as current CPUs and embedded systems and digital signal processing applications.

For our course project, we've designing a 4-bit pipeline multiplayer using the advanced 22nm CMOS technology. Our primary goals in this project are to achieve the fastest possible operation by minimizing the total delay of the system, ensuring an efficient use of chip real estate to keep the area used for all components as compact as possible, and also focusing on the power aspects of the design. We're keenly interested in both the current consumption, aiming for efficient energy use, so all of these we can calculate and determined when implement the all system by implement in the first each component separately and then connect each of them with other, and each component its implement from the base component, its CMOS (Complementary Metal-Oxide-Semiconductor) technology is a fundamental semiconductor approach widely employed in electronics. It's characterized by the use of complementary pairs of MOSFETs (Metal-Oxide-Semiconductor Field-Effect Transistors) to build digital logic gates, but it faces challenges when interfacing with SETs (Single-Electron Transistors), CMOS and SETs are being combined for potential improvements in 3-D integrated circuits. However, the combination faces challenges such as low current capacity, delays, and voltage boost. Researchers are working on improving SETs' compatibility with CMOS, particularly for long wires, to make it more powerful and faster. This combination could lead to more powerful and faster 3-D integrated circuits.

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

III. THEORY

Power Consumption vs. Process Node

Power consumption vs. process node for Intel processors

Process Node	Power Consumption (W)
180 nm	100
90 nm	65
65 nm	45
45 nm	30
32 nm	20
22 nm	15
14 nm	10
10 nm	7
7 nm	5
5 nm	4

Table 1: Power consumption vs. process node.

As you can see, there is a clear trend of decreasing power consumption with each new process node. This is due to a number of factors, including the smaller feature sizes, which allow for more transistors to be packed into the same area, and the lower voltages that can be used with smaller transistors.

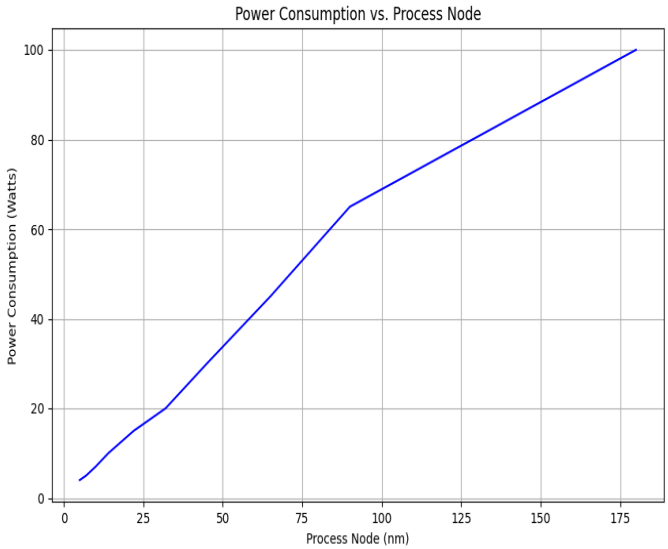


Figure 1: Power consumption vs. process node.

Cost

Cost vs. process node for Intel processors

Process Node (nm)	Manufacturing Cost (USD/chip)
180	100
90	80
65	60
45	40
32	30
22	25
14	20
10	15
7	10
5	5

Table 2: Cost vs. process node

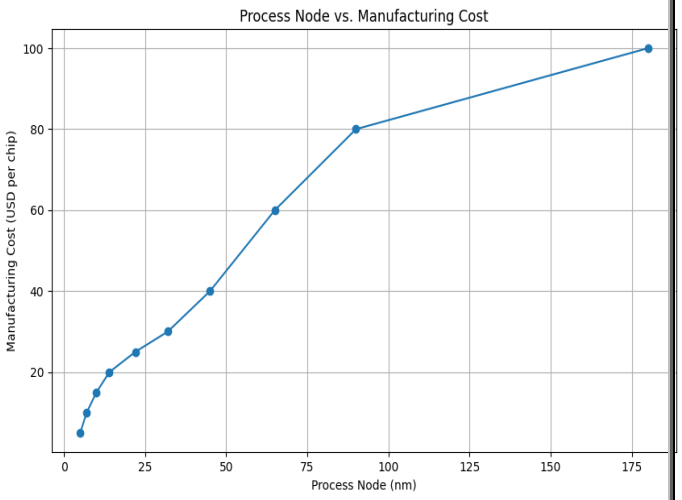


Figure 2: Cost vs. process node.

Manufacturing costs tend to rise initially as semiconductor process nodes shrink. This increase is driven by the higher development and fabrication expenses associated with smaller nodes. However, as technology matures, costs may decrease, with the 7 nm node being more cost-effective than the 10 nm node, for instance. Nonetheless, exceptions occur, such as the potential for the 5 nm node to be costlier due to advanced materials and processes. This trend poses challenges for the semiconductor industry, pushing for cost reduction efforts despite complexities like the need for precision manufacturing and packaging for smaller chips. Despite these challenges, the industry is committed to making smaller chips more affordable.

Performance vs. process node for Intel processors

Process Node (nm)	CPU Speed (GHz)
180	1.6
90	2.8
65	3.2
45	3.5
32	3.7
22	4.2
14	4.5
10	4.8
7	5

Table 3: Performance vs. process node.

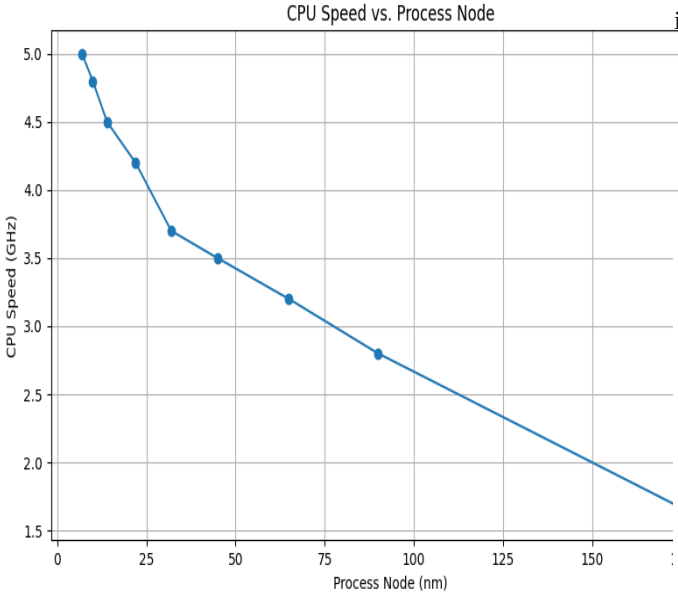


Figure 3: Performance vs. process node.

There is a direct link between CPU speed and semiconductor process node size, with smaller nodes leading to increased performance. The progression from 180 nm to 7 nm demonstrates this, as CPU speeds have risen from 1.6 GHz to 5 GHz. This progress highlights advances in semiconductor technology, enhancing CPU performance through smaller manufacturing processes. However, as we approach the limits of miniaturization, typically around 10 nm, the benefits plateau due to increased susceptibility to noise and physical limitations. Consequently, the industry explores alternative avenues, such as new materials and architectures, to further enhance performance. Smaller transistors contribute to this by enabling faster switching, higher transistor density, and improved efficiency, resulting in reduced power consumption and heat generation. Despite these challenges, continuous innovation in the semiconductor sector remains focused on elevating chip performance and effectiveness.

IV. DESIGNE OF ALL SYSTEM

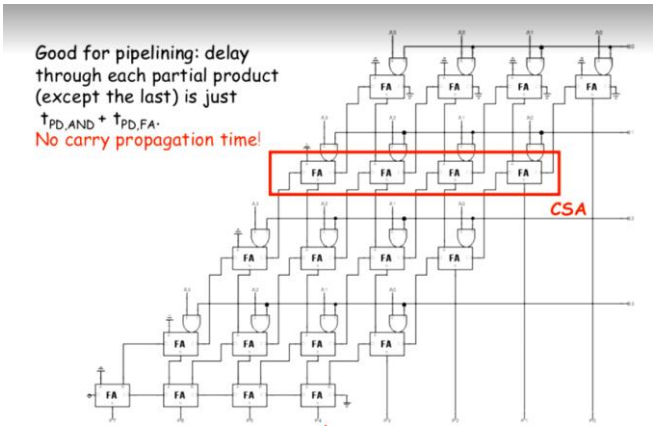


Figure 4: 4x4 Enhanced Pipeline Multiplier.

The system in the figure above implements from And gate and a full adder to achieve 4x4 Pipeline Multiplier that will be explained below each gate how and from what implement in all schematic, icon, and layout design.

V. DESIGN INVERTOR IMPLEMENTATION

i. Invertor schematic implementation

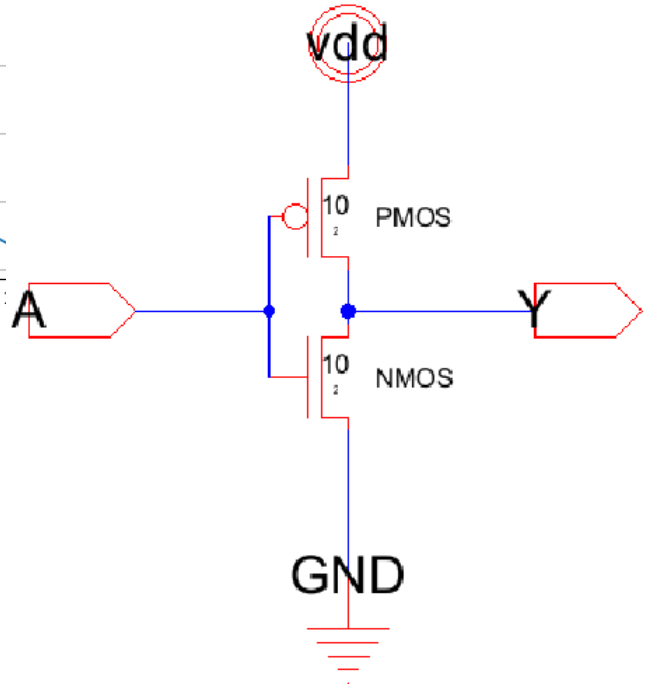


Figure 5: Invertor schematic implementation.

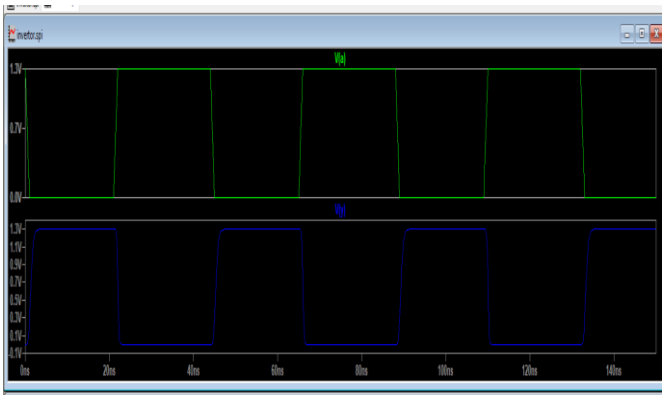


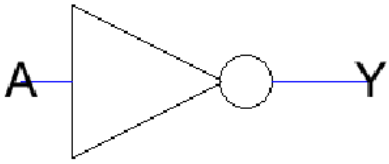
Figure 6: Invertor schematic output wave.

- The code we put in schematic invertor to appear the wave above:

```
vdd vdd 0 DC 1.3
va A 0 pulse 1.3 0 0 1n 1n 20n 44n
cload Y 0 50fF
.tran 150n
.include C:\Users\khaled\Desktop\22nm-2.txt
```

Figure 7: Code of invertor schematic.

2. invertor icon



3- Layout for invertor

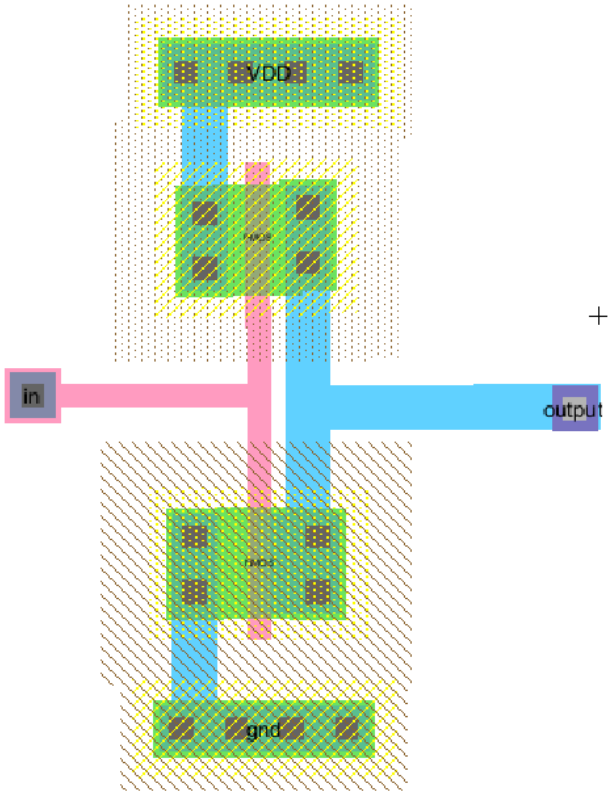


Figure 8: Layout for invertor.

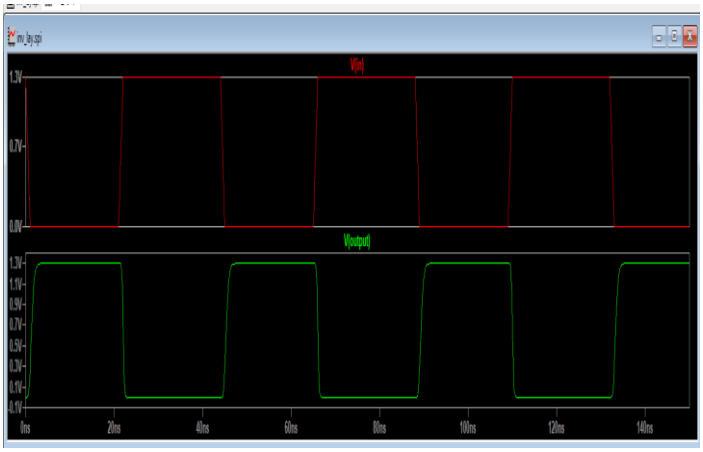


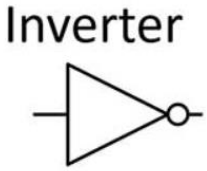
Figure 9: Invertor layout output wave.

- The code we put in schematic invertor to appear the wave above:

```
vdd vdd 0 DC 1.3
va in 0 pulse 1.3 0 0 1n 1n 20n 44n
cload output 0 50fF
.tran 150n
.include C:\Users\khaled\Desktop\22nm-2.txt
```

Figure 10: Code of invertor layout.

And from truth table below and wave form we get, can conclude that our connection of invertor is true.



Input	Output
0	1
1	0

Table 4: Truth table of invertor.

Invertor plots explanation

The figure produced by the simulation code given illustrates the behavior of an electrical circuit over a time frame of 50 nanoseconds. The first waveform corresponds to the "vin" voltage source, and it has a pulse shape with a 1.3V beginning value, a quick 1 nanosecond climb to this level, a 20 nanosecond pulse width, and a 1 nanosecond fall to 0V, which repeats every 44 nanoseconds. The second waveform represents the vdd voltage source, which remains constant throughout the simulation at 1.3V. An initial transient reaction may occur as a result of circuit circumstances and interactions with the cloud capacitor at node Y.

➔ Sizing

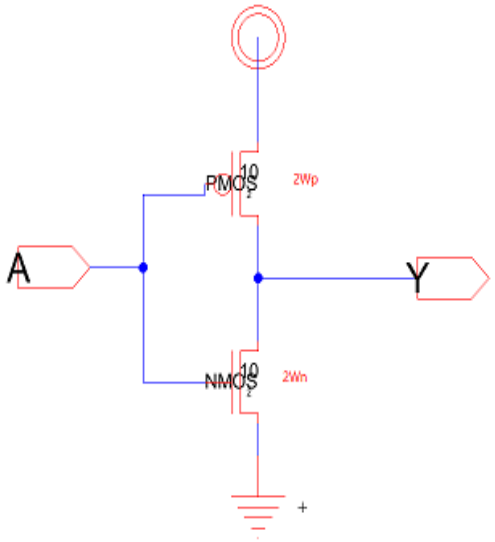


Figure 11: inverter sizing

In the case of the inverter, both the PMOS and NMOS transistors have been sized with a width 2W each. This balanced sizing ensures that both transistors in the inverter play their roles effectively. The PMOS transistor is responsible for pulling the output signal high when needed, while the NMOS transistor pulls it low. This equal sizing of the transistors ensures efficient and reliable inverter operation, a fundamental aspect of digital circuit design.

Configuring and Specifying a Precise CMOS Inverter Simulation

We established the parameters to guide our analysis. We defined default sizes, like lengths, widths, and areas, which were set to specific values, such as DEFL=3UM and DEFW=3UM, to serve as starting points for certain components. Our goal was to ensure efficiency in our simulations, so we set a limit of 1000 time points to keep things manageable (LIMPTS=1000). To make the process smoother, we disabled iterative algorithms ("ITL5=0") while also taking care to maintain precision. We set relative and absolute tolerances for convergence, using values like RELTOL=0.01 ABSTOL=500PA, and VNTOL=500UV."

We also paid attention to the timing and coding aspects of our simulation by managing level timing and coding intricacies with parameters like LVLTIM=2 and LVLCOD=1. After these initial setup steps, we delved into the specifics of our circuit. We carefully defined the characteristics of the NMOS and PMOS transistors, specifying key parameters like threshold voltage, trans conductance, mobility factors, and channel length modulation.

Next, we established global signals, gnd, and vdd, to ensure consistent reference and power supply throughout the simulation. Our focus then turned to the core of our analysis - the CMOS inverter represented by the top-level cell.

For our voltage sources, we set vdd to a constant 5V supply. The input voltage Vin was meticulously defined as a piecewise linear waveform, starting at 0V, transitioning to 5V over 10 nanoseconds, remaining at 5V for 20 nanoseconds, transitioning to 0V over 50 milliseconds, and eventually stabilizing at 0V for 60 nanoseconds.

Additionally, we incorporated a capacitor-labeled cloud in our circuit, connecting the output node out to the ground. This capacitor had a capacitance of 250 femtofarads (250fF).

VI. DESIGN NAND IMPLEMENTATION

1- Schematic NAND

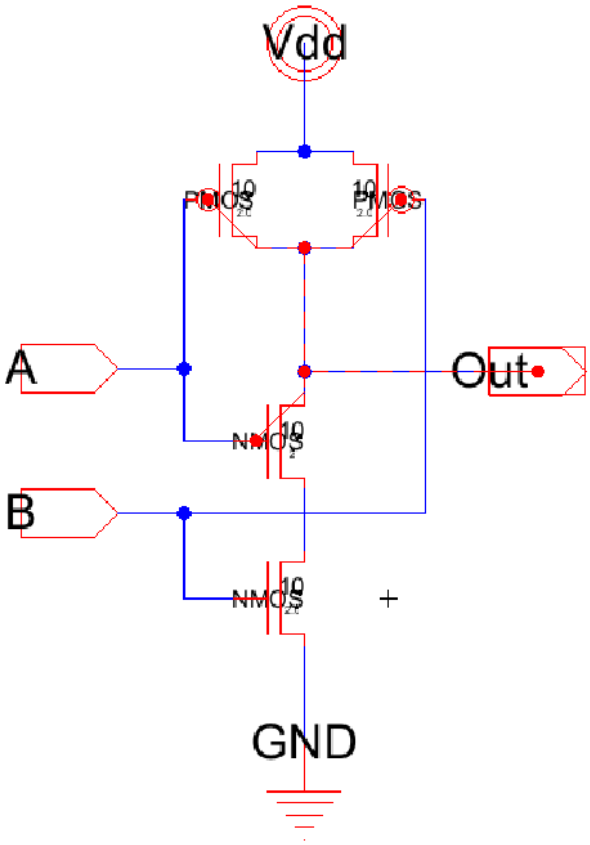


Figure 12: NAND schematic implementation.

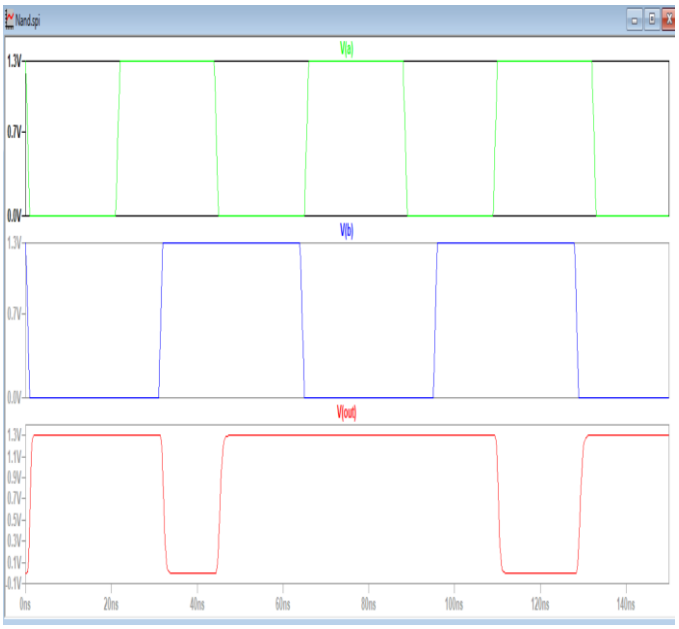


Figure 13: NAND schematic waveform.

- The code we put in schematic NAND to appear the wave above:

```
vdd vdd 0 DC 1.3
va A 0 pulse 1.3 0 0 1n 1n 20n 44n
vb B 0 pulse 1.3 0 0 1n 1n 30n 64n
cload Out 0 50fF
.tran 150n
.include C:\Users\khaled\Desktop\22nm-2.txt
```

Figure 14: Code of NAND schematic.

2- NAND icon

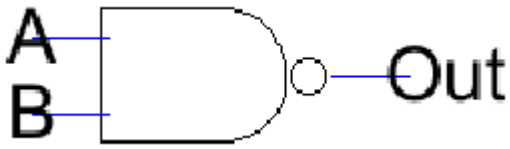


Figure 15: NAND icon.

3- NAND Layout

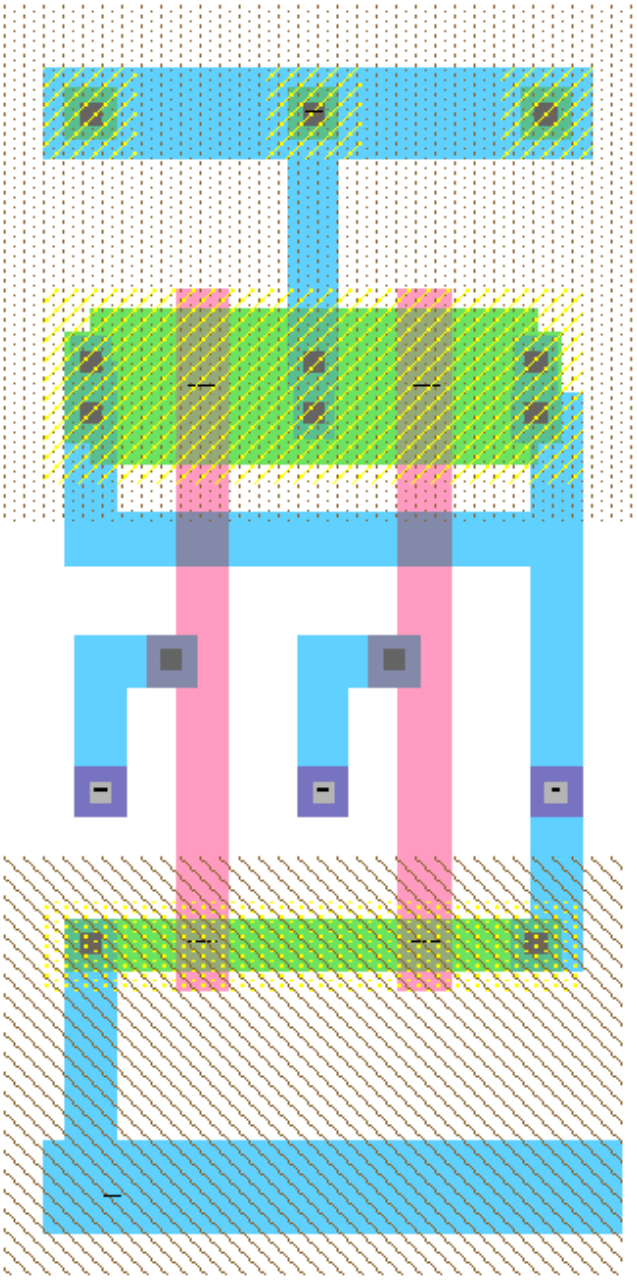


Figure 16: NAND Layout.

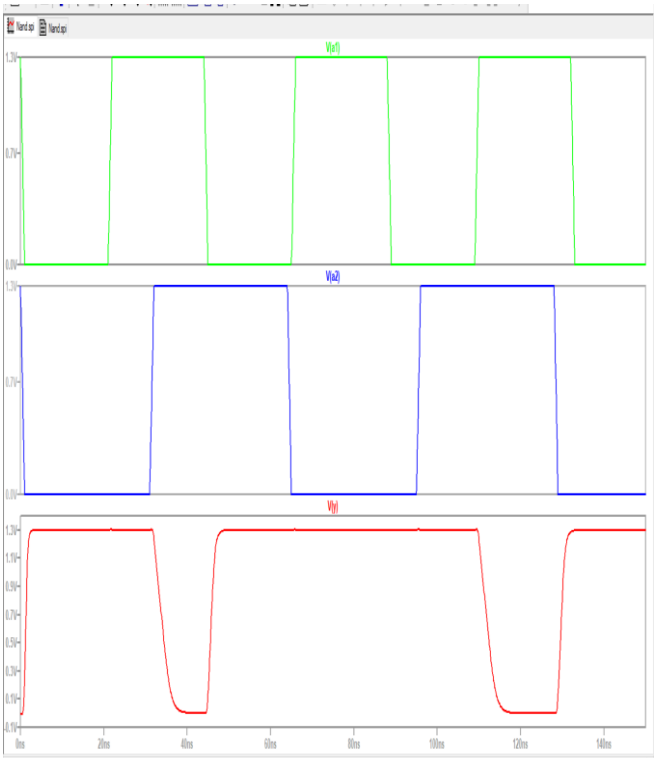


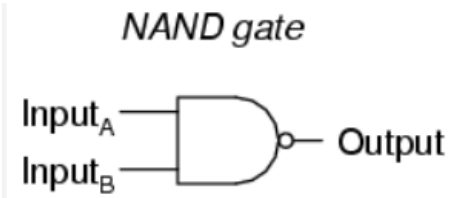
Figure 17: Output wave form from NAND Layout.

- The code we put in Layout NAND to appear the wave above:

```
vdd vdd 0 DC 1.3
va a1 0 pulse 1.3 0 0 1n 1n 20n 44n
vb a2 0 pulse 1.3 0 0 1n 1n 30n 64n
cload y 0 50fF
.tran 150n
.include C:\Users\khaled\Desktop\22nm-2.txt
```

Figure 18: The NAND layout code.

And from truth table below and wave form we get, can conclude that our connection of NAND is true.



A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

Table 5: NAND truth table with icon.

➔ Sizing

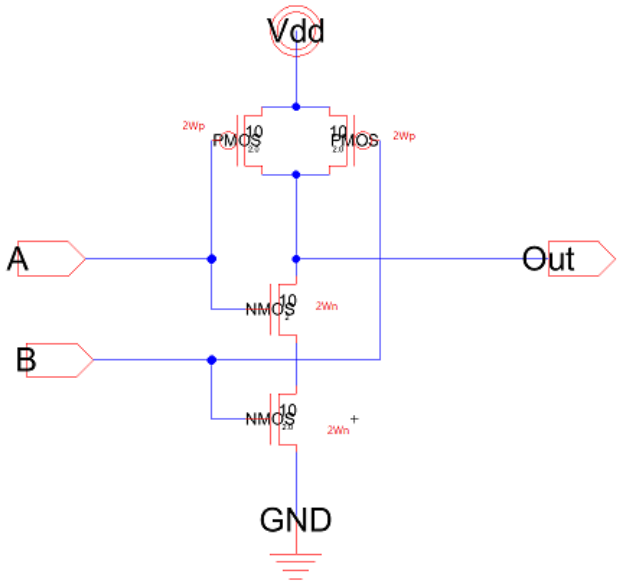


Figure 19: NAND gate with the size

In this design for a NAND gate, both the NMOS and PMOS transistors have been given a size (width) of 4 W. This balanced sizing is crucial because it ensures that both transistors play their roles effectively. The NMOS transistor helps pull the output signal down when needed, while the PMOS transistor assists in pulling it up. This balance ensures that the gate works efficiently and conserves power, which is a fundamental aspect of how these gates function in digital circuits.

Configuring and Specifying a Precise CMOS NAND Simulation as we shown above

We took meticulous care in orchestrating the setup and simulation of a NAND gate circuit, prioritizing precision and thorough attention to detail. Our commitment to accuracy was evident in the thoughtful configuration of crucial UC SPICE parameters, with MIN_RESIST set at 4.0 and MIN_CAPAC at 0.1FF.

Inside the circuit, we focused on characterizing four transistor instances, consisting of two NMOS and two PMOS transistors. Each transistor was defined with specific dimensions, including a length (L) of 0.044U, which is equivalent to 44 nanometers, and a width (W) of 0.22U, equivalent to 220 nanometers.

For voltage sources, we ensured stability by maintaining vdd at a constant DC voltage of 1.3V. Our attention to detail extended to the definition of input waveforms va and vb as pulse signals, with precise control over their timing and voltage levels.

To gain comprehensive insights into the circuit's behavior, we thoughtfully introduced a capacitance component labeled "cload," connecting it to the output node "Out" with a capacitance of 25fF (femtofarads).

Our simulation settings were characterized by precision, with a transient analysis duration specified at 50 nanoseconds (50n). This deliberate choice allowed us to delve into the circuit's behavior with meticulous granularity and precision, ensuring a thorough understanding of its performance.

VII. DESIGN AND IMPLEMENTATION

The AND gate implement from connect NAND gate with INVERTOR, as shown in figure bellow.

1- AND Schematic

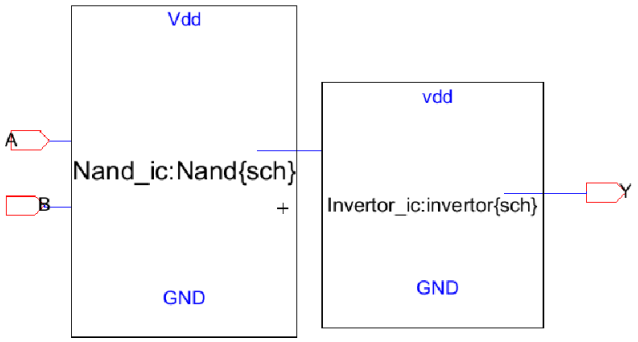


Figure 20: AND Schematic

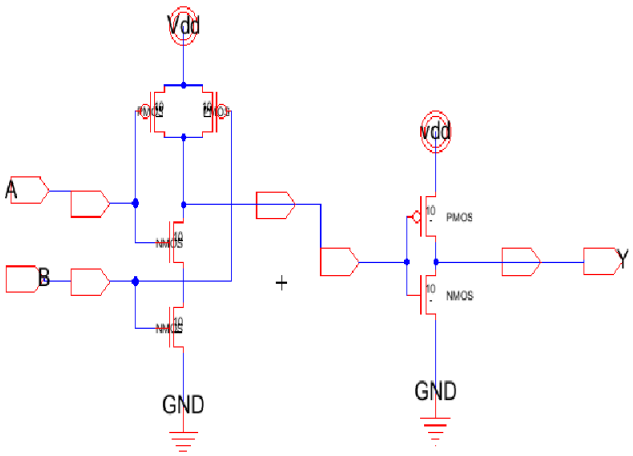


Figure 21: AND Schematic.

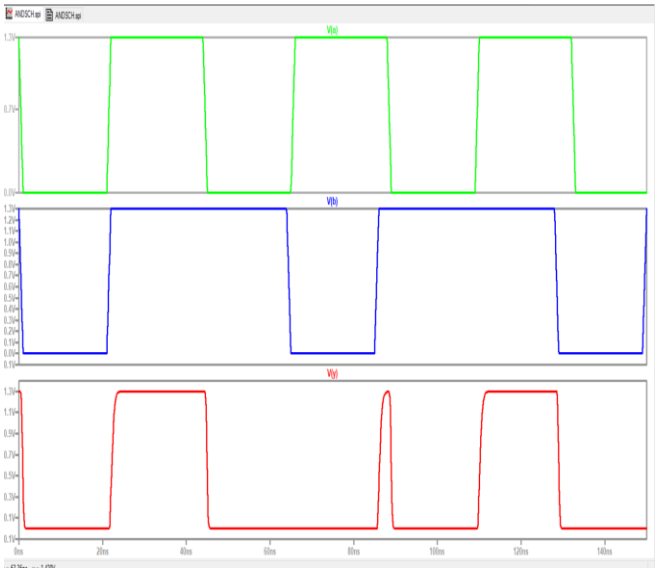


Figure 22: AND Schematic output waveform.

- The code we use in Schematic AND to appear the wave above:

```
vdd vdd 0 DC 1.3
va A 0 pulse 1.3 0 0 1n 1n 20n 44n
vb B 0 pulse 1.3 0 0 1n 1n 20n 64n
cload Y 0 50fF
.tran 150n
.include C:\Users\khaled\Desktop\22nm-2.txt
```

Figure 23: The AND schematic code.

2- AND Layout

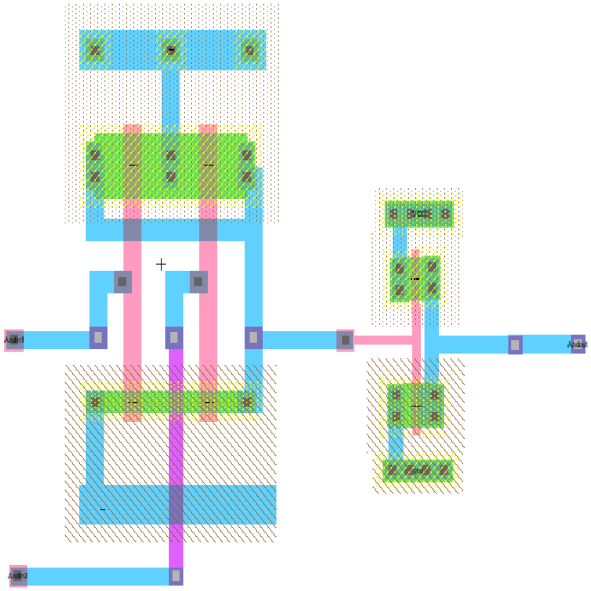


Figure 24: AND Layout.

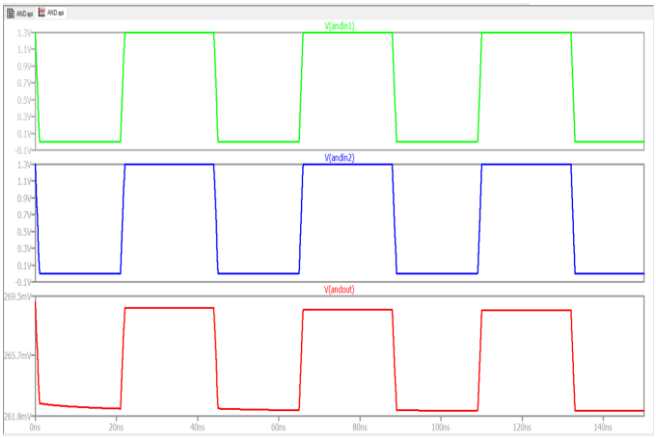


Figure 25: AND Layout output waveform.

- The code we use in Layout AND to appear the wave above:

```
vdd vdd 0 DC 1.3
va Andin1 0 pulse 1.3 0 0 1n 1n 20n 44n
vb Andin2 0 pulse 1.3 0 0 1n 1n 20n 44n
cload Andout 0 50fF
.tran 150n
.include C:\Users\khaled\Desktop\22nm-2.txt
```

Figure 26: The AND Layout code.

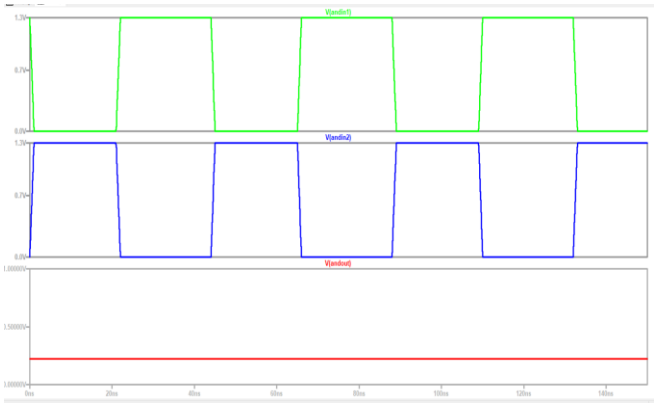


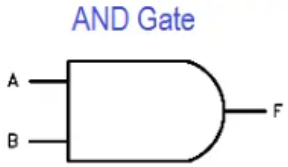
Figure 27: AND Layout output waveform with another values input.

- The code we use in Layout AND to appear the wave above:

```
vdd vdd 0 DC 1.3
va Andin1 0 pulse 1.3 0 0 1n 1n 20n 44n
vb Andin2 0 pulse 0 1.3 0 1n 1n 20n 44n
cload Andout 0 50fF
.tran 150n
.include C:\Users\khaled\Desktop\22nm-2.txt
```

Figure 28: The AND Layout code.

And from truth table below and wave form we get, can conclude that our connection of AND is true.



INPUT		OUTPUT
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

Table 6: AND truth table with icon.

Our AND gate design features two distinct gate instances: NAND Gate: This gate employs both NMOS and PMOS connected to inputs A and B respectively.

Inverter Gate: It is likely configured as a complementary inverter.
Its primary purpose is to invert the input signal "A" and produce an output "Y."Both of these gate instances are connected together to function as NAND gate

Moreover, the design incorporates essential elements for performance evaluation:

Voltage Sources to ensure correct circuit operation. vdd is set to a value of 1.3. Also, Capacitance values are explicitly specified to represent the capacitive load at various circuit nodes (C load Out has a value of 25fF, and C load Y possesses a capacitance of 50fF). The plot shows specific measurement points to assess gate performance:
"tf" measures the fall time of the "v(out)" signal as it transitions from 4.5V to 0.5V.
"tr" measures the rise time of the "v(out)" signal as it changes from 0.5V to 4.5V.

➔ Sizing

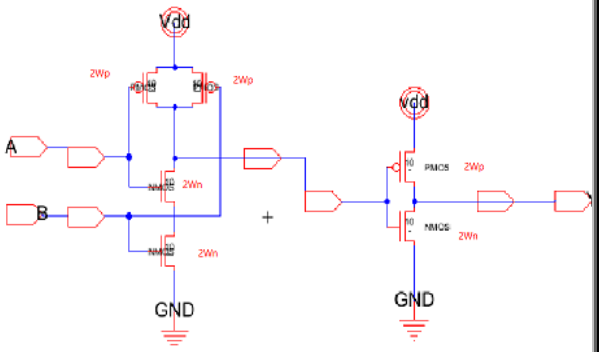


Figure 29: AND gate sizing

In your CMOS circuit design, if the PMOS transistor width for the NAND gate is 4W and the width for the inverter's PMOS transistor is 2W, the combined total PMOS transistor width for both the NAND gate and inverter is 6W.
Similarly, if the NMOS transistor width in the NAND gate is 4W, and in the inverter, it's 2W, the total NMOS

transistor width for the combined NAND gate and inverter is also 6W. This balanced sizing strategy ensures symmetrical performance characteristics in your CMOS circuit, optimizing its efficiency and signal integrity.

VIII. FULL ADDER IMPLEMENTATION

We implement the Full adder from 9 NAND gate as below figure:

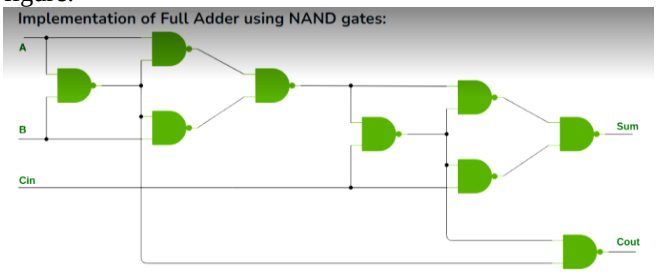


Figure 30: Full adder from 9 NAND.

In the other hand, the full adder has more implementation technique such as it implement from 9 NOR like below figure:

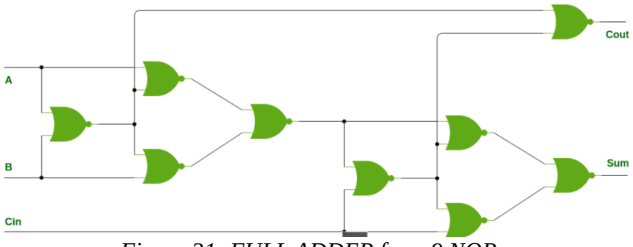


Figure 31: FULL ADDER from 9 NOR.

But we choose implement this by 9 NAND not by 9 NOR to save 18 CMOS (PMOS & NMOS) component, that the Nor gate need more component so it lead to more area and delay.

1- Full adder schamatic

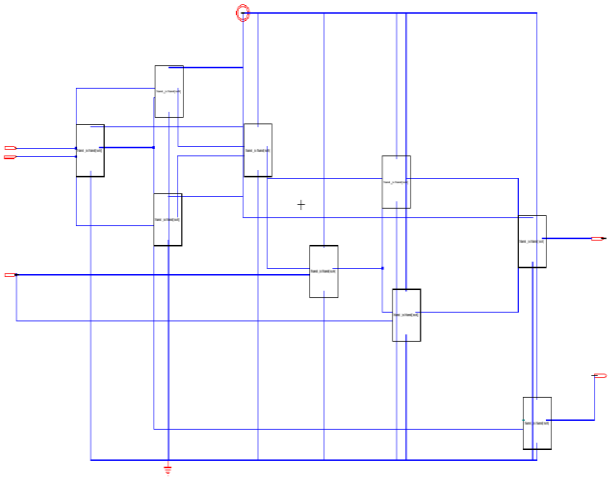


Figure 32: Full adder schematic implementation.

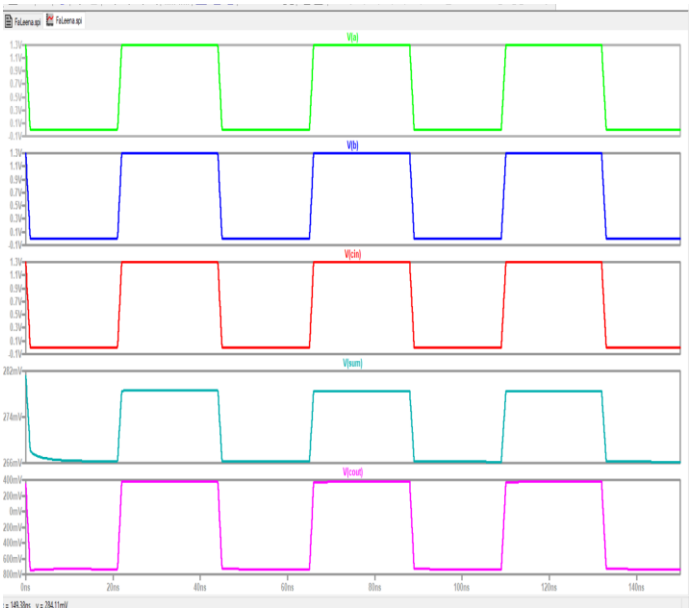


Figure 33: Full adder output waveform from schematic.

- The code we use in Schematic Full adder to appear the wave above:

```
vdd vdd 0 DC 1.3
va A 0 pulse 1.3 0 0 1n 1n 20n 44n
vb B 0 pulse 1.3 0 0 1n 1n 20n 44n
venable CIN 0 pulse 1.3 0 0 1n 1n 20n 44n
cload SUM 0 50fF
.tran 150n
.include C:\Users\khaled\Desktop\22nm-2.txt
```

Figure 34: The Full adder schematic code.

2- Full adder Layout

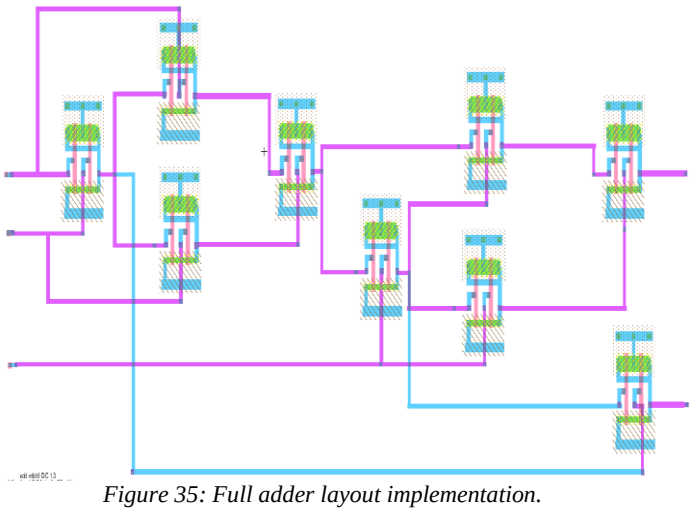


Figure 35: Full adder layout implementation.

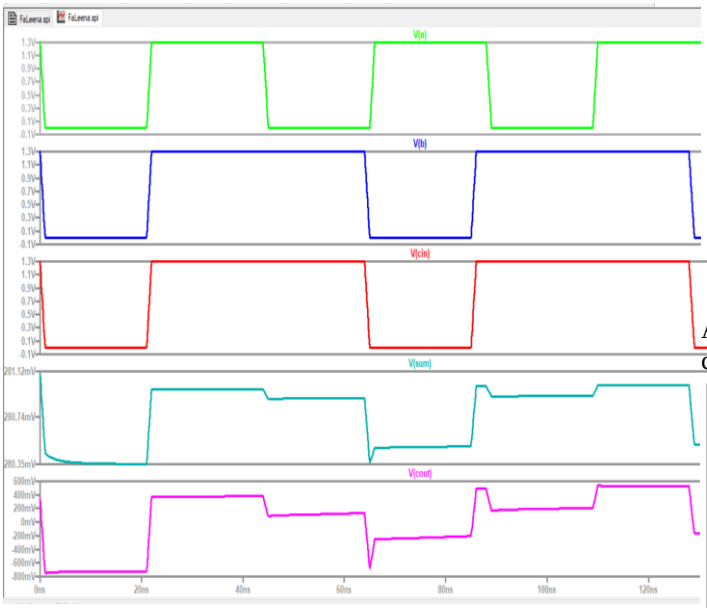


Figure 36: Full adder output waveform from layout.

- The code we use in layout Full adder to appear the wave above:

```
vdd vdd 0 DC 1.3
va A 0 pulse 1.3 0 0 1n 1n 20n 44n
vb B 0 pulse 1.3 0 0 1n 1n 20n 64n
venable CIN 0 pulse 1.3 0 0 1n 1n 20n 64n
cload SUM 0 1000fF
.tran 150n
.include C:\Users\khaled\Desktop\22nm-2.txt
```

Figure 37: The Full adder layout code.

- The full adder output bellow by change code to fet more output value.

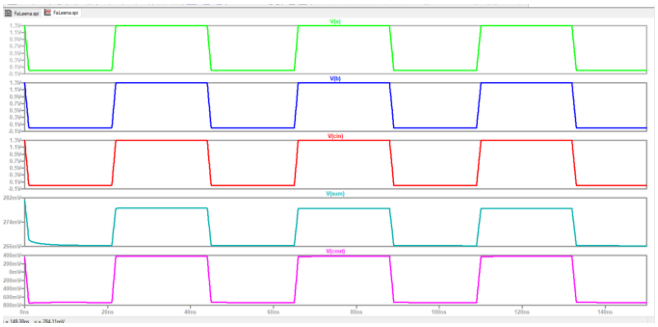


Figure 38: Full adder output waveform from layout.

- The code we use in layout Full adder to appear the wave above:

```
vdd vdd 0 DC 1.3
va A 0 pulse 1.3 0 0 1n 1n 20n 44n
vb B 0 pulse 1.3 0 0 1n 1n 20n 44n
venable CIN 0 pulse 1.3 0 0 1n 1n 20n 44n
cload SUM 0 50fF
.tran 150n
.include C:\Users\khaled\Desktop\22nm-2.txt
```

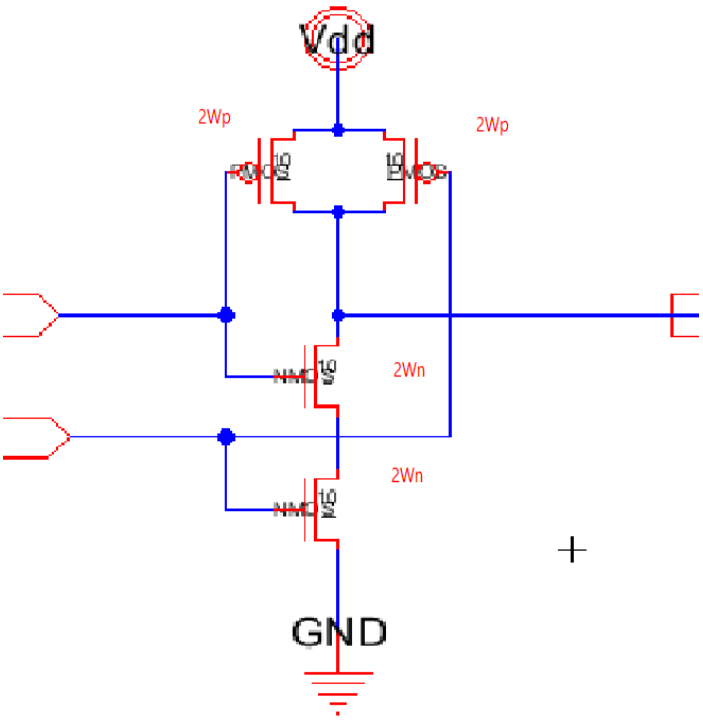
Figure 39: The Full adder layout code.

And from truth table below and wave form we get, can conclude that our connection of FULL ADDER is true.

Inputs			Outputs	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 7: FULL ADDER truth table

➔ Sizing



.Figure 40: full adder sizing

In your full adder circuit construction comprising 9 NAND gates, with the previously determined width of 4W for both the PMOS and NMOS transistors in each NAND gate, the total width for PMOS transistors in the full adder is 9 NAND gates × 4W = 36W for PMOS. Similarly, the combined total width for NMOS transistors in the full adder is also 9 NAND gates × 4W = 36W for NMOS. This sizing

approach ensures symmetrical transistor characteristics throughout the full adder circuit, contributing to its proper functionality and efficient operation.

IX. THE WHOLE SYSTEM

1- The whole system

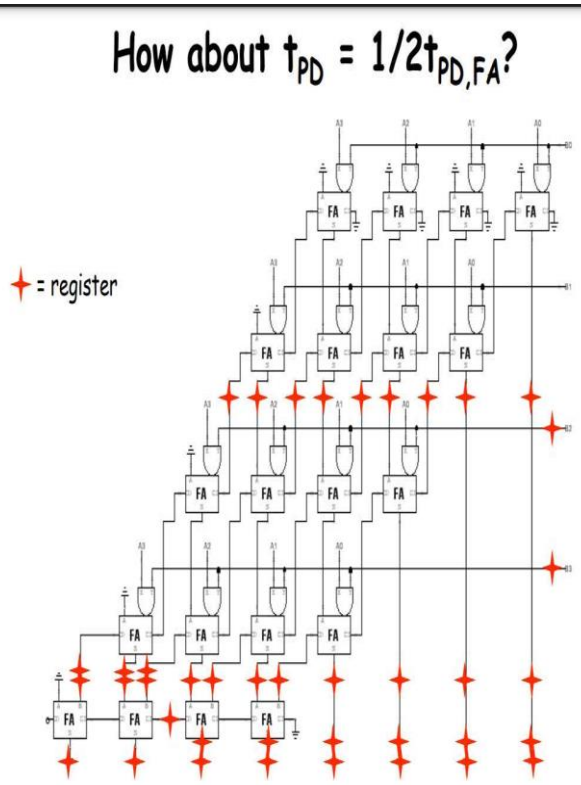


Figure 41: the whole system

The 4x4 Enhanced Pipeline Multiplier is a specialized hardware circuit designed for efficient binary multiplication of two 4-bit binary numbers. It employs a pipelined architecture, breaking down the multiplication process into stages, which enables faster computation. Each stage includes a combination of AND gates for generating partial products and full adders for summing these partial products. By processing the multiplication in smaller steps, the pipeline multiplier reduces the critical path delay and enhances overall performance. This design is particularly valuable in digital signal processing and arithmetic operations, where multiplication is a fundamental operation, and optimizing speed and efficiency is crucial.

2- The system schematic

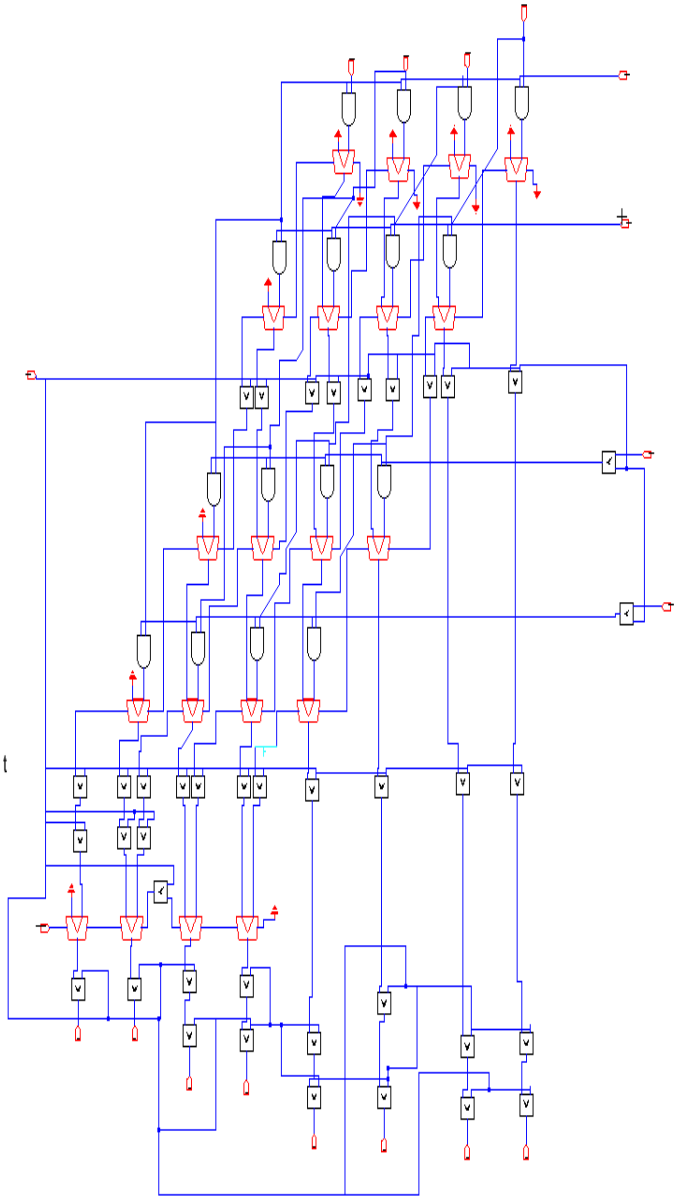


Figure 42: the whole system schematic

Flip-flop registers are essential components in a system because they provide a means to store and control data within digital circuits. They serve as memory elements, holding binary information in the form of 0s and 1s. These registers enable sequential logic and synchronization in various digital systems, such as computers, microcontrollers, and communication devices. By using flip-flop registers, we can temporarily store data, synchronize different parts of a system, enable data processing, and facilitate the orderly execution of instructions, making them a fundamental building block in digital electronics and computing.

3- The whole system layout

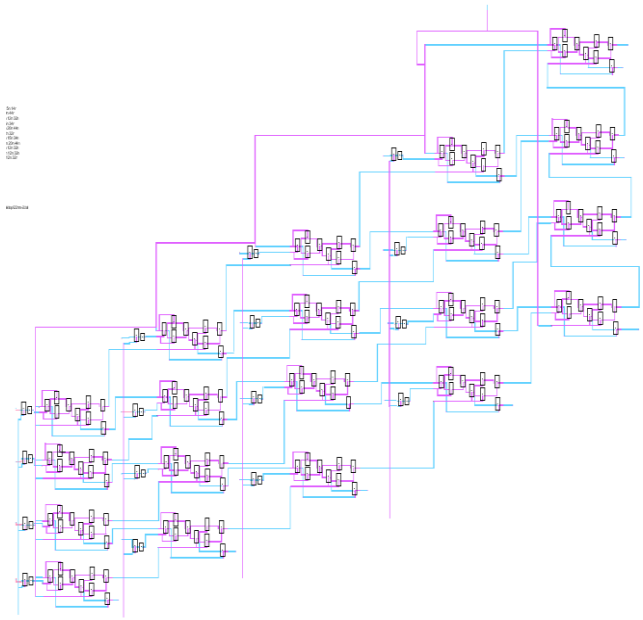


Figure 43: the whole system layout

Flip flop we used in all system

D flip flop

The selection of flip-flop circuit designs is critical to reaching optimal performance. The NAND-based D flip-flop and the JK flip-flop configured as a D flip-flop have significant advantages that impact system performance. We conducted a short investigation that gives insights into the two designs in order to select the one that best matches our system.

That implement from 1 invertor and 9 nand:

Layout:

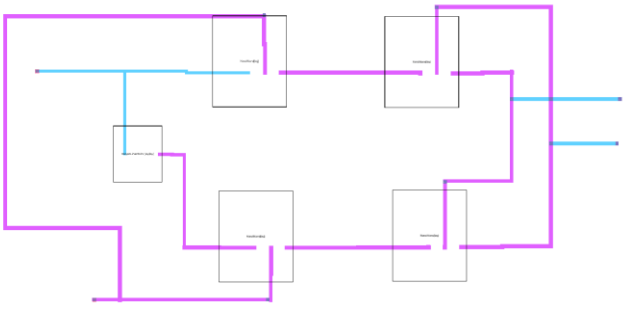


Figure 44: D flip flop layout.

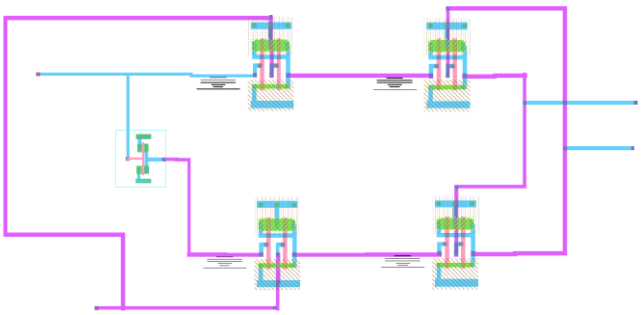


Figure 45: D flip flop layout.

D flip flop Schematic

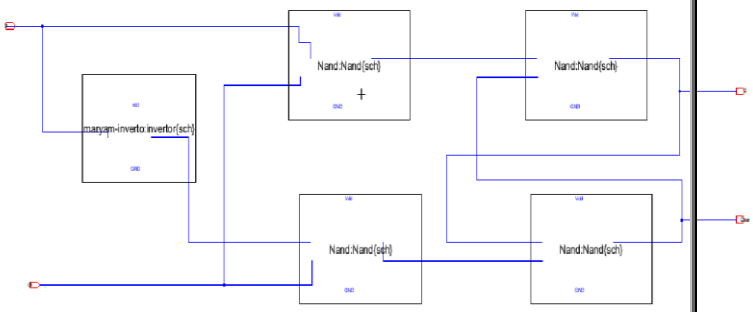


Figure 46: D flip flop Schematic.

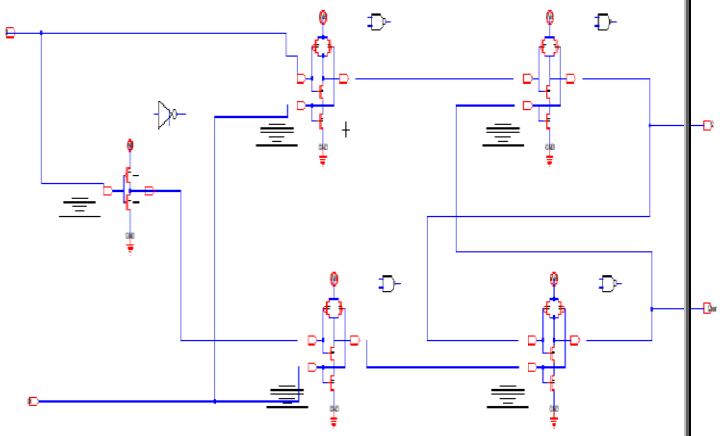
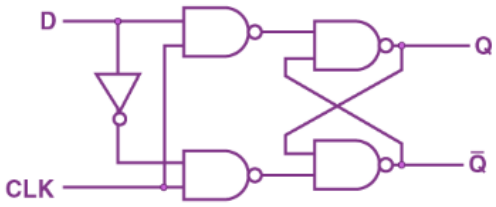


Figure 47: D flip flop Schematic.

1. NAND-based D Flip-Flop

- **Description:** In this design, there are two NAND gates (NAND1 and NAND2), and the output of NAND1 feeds back to one of its inputs, creating a positive feedback loop. This feedback loop allows the flip-flop to store and remember the input value (D) until a clock signal (CLK) transitions.
- **Operation:** When CLK transitions from low to high (rising edge), the input D is transferred to the output Q. When CLK is low, the flip-flop holds its previous state.



Truth Table

Q	D	$Q_{(t+1)}$
0	0	0
0	1	1
1	0	0
1	1	1

Figure 48: NAND-based D Flip-Flop.

Characteristic	NAND-based D Flip-Flop	JK Flip-Flop as D Flip-Flop
Speed (Propagation Delay)	Faster	Slightly Slower
Power Consumption	Lower	Slightly Higher
Space Constraints	Smaller	Slightly Larger
Number of Transistors Needed	Fewer	More
Delay	Low	Slightly Higher

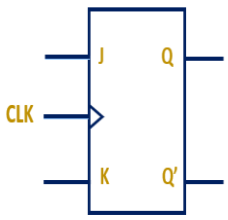
Table 8: Comparison between two flip flop implementation.

After carefully considering which design to use to create our circuit, we decided on the NAND-based D flip-flop because to its simplicity and reduced power consumption. When compared to a JK flip-flop arranged as a D flip-flop, it offers quicker operation with reduced power consumption.

JK Flip-Flop with D Flip-Flop Inputs:

The JK flip-flop can be configured to work as a D flip-flop by connecting its J and K inputs together.

Symbol



Truth Table

CLK	J	K	Q_{n+1}
↑	0	0	Q_n
↑	0	1	0
↑	1	0	1
↑	1	1	Q_n'

Figure 49: JK Flip-Flop with D Flip-Flop.

- **Description:** In this design, the J and K inputs of a JK flip-flop are tied together and act as the D input. The CLK input is used to control the clocking of the flip-flop. When CLK transitions from low to high (rising edge), the input D is transferred to the output Q.
- **Operation:**
  - If D is 0, it sets Q to 0 on the rising edge of CLK.
  - If D is 1, it sets Q to 1 on the rising edge of CLK.
  - If CLK is low, the flip-flop holds its previous state.

➔ The code for the system

```
vdd vdd 0 DC 0.95
VClk Clk 0 pulse 0.95 0 0 1n 1n 5n 14n
VA0 A0 0 pulse 0 0 0 1n 1n 20n 44n
VA1 A1 0 pulse 0.95 0.95 0 2n 2n 12n 32n
VA2 A2 0 pulse 0 0 0 1n 1n 15n 34n
VA3 A3 0 pulse 0.95 0.95 0 1n 1n 20n 44n
VB0 B0 0 pulse 0 0 0 2n 2n 12n 32n
VB1 B1 0 pulse 0.95 0.95 0 1n 1n 15n 34n
VB2 B2 0 pulse 0.95 0.95 0 01n 1n 20n 44n
VB3 B3 0 pulse 0.95 0.95 0 2n 2n 12n 32n
VC Cout 0 pulse 0.95 0.95 0 2n 2n 12n 32n
VZero GND 0 pulse 0 0 0 2n 2n 12n 32n
cloadP0 F0 0 15fF
cloadP1 F1 0 15fF
cloadP2 F2 0 15fF
cloadP3 F3 0 15fF
cloadP4 F4 0 15fF
cloadP5 F5 0 15fF
cloadP6 F6 0 15fF
cloadS7 F7 0 15fF
.tran 210n
.include C:\Users\khaled\Desktop\22nm-2.txt
```

Figure 50: the code for the first input example

Figure 51: the input A waveform

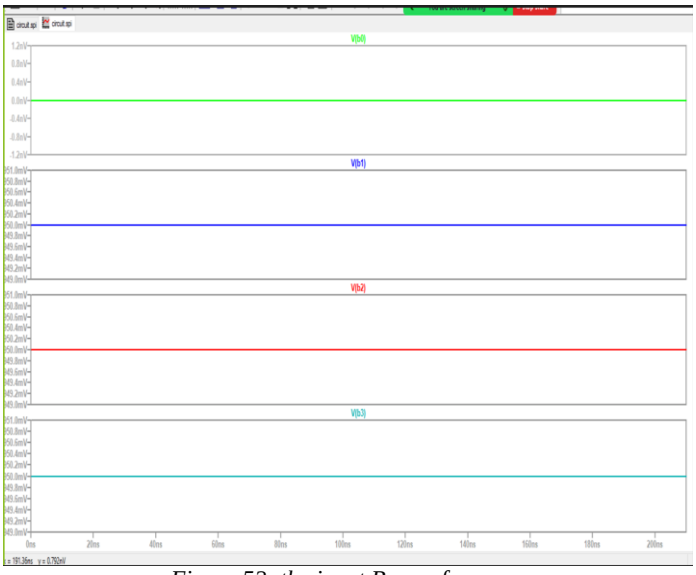


Figure 52: the input B waveform

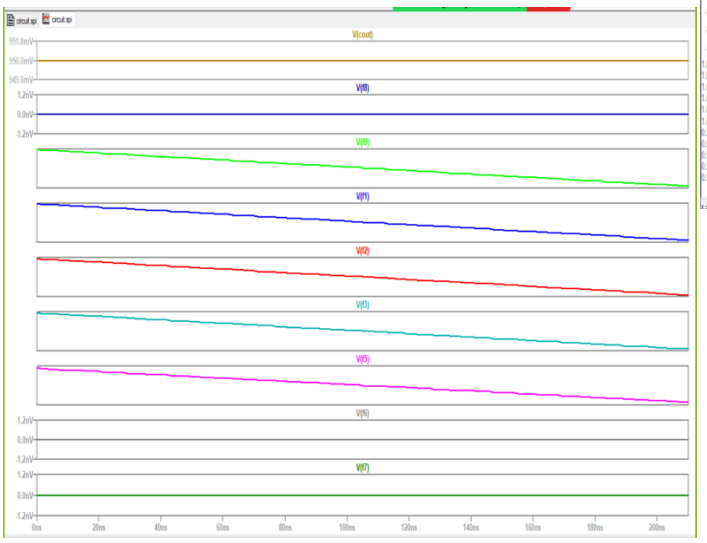


Figure 53: the output waveform

➔ Another input example

```
vdd vdd 0 DC 1.5
VClk Clk 0 pulse 1 0 0 1n 1n 5n 14n
VA0 A0 0 pulse 0 0 0 1n 1n 20n 44n
VA1 A1 0 pulse 1 1 0 2n 2n 12n 32n
VA2 A2 0 pulse 0 0 0 1n 1n 15n 34n
VA3 A3 0 pulse 1 1 0 1n 1n 20n 44n
VB0 B0 0 pulse 0 0 0 2n 2n 12n 32n
VB1 B1 0 pulse 1 1 0 1n 1n 15n 34n
VB2 B2 0 pulse 1 1 0 01n 1n 20n 44n
VB3 B3 0 pulse 1 1 0 2n 2n 12n 32n
VC Cout 0 pulse 1 1 0 2n 2n 12n 32n
VZero GND 0 pulse 0 0 0 2n 2n 12n 32n
cloadP0 F0 0 15fF
cloadP1 F1 0 15fF
cloadP2 F2 0 15fF
cloadP3 F3 0 15fF
cloadP4 F5 0 15fF
cloadP5 F6 0 15fF
cloadP6 F7 0 15fF
cloadS7 F8 0 15fF
.tran 210n
.include C:\Users\khaled\Desktop\22nm-2.txt
```

Figure 54: the code for the second input example

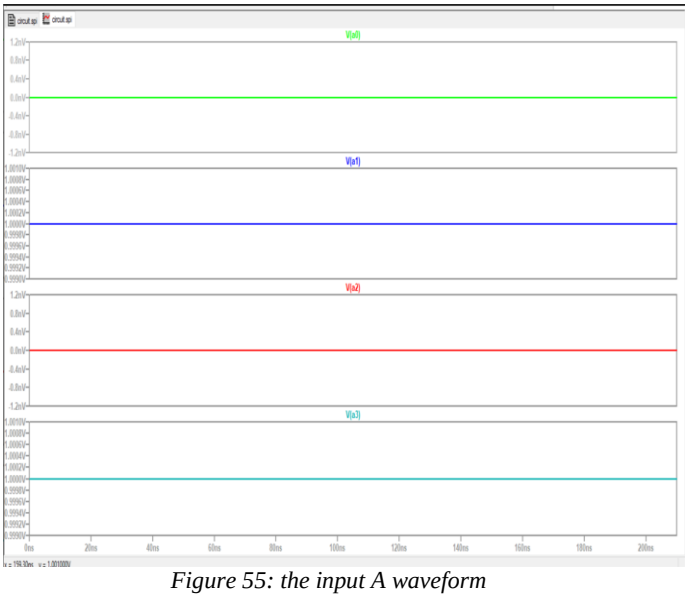


Figure 55: the input A waveform

Figure 56: the input B waveform

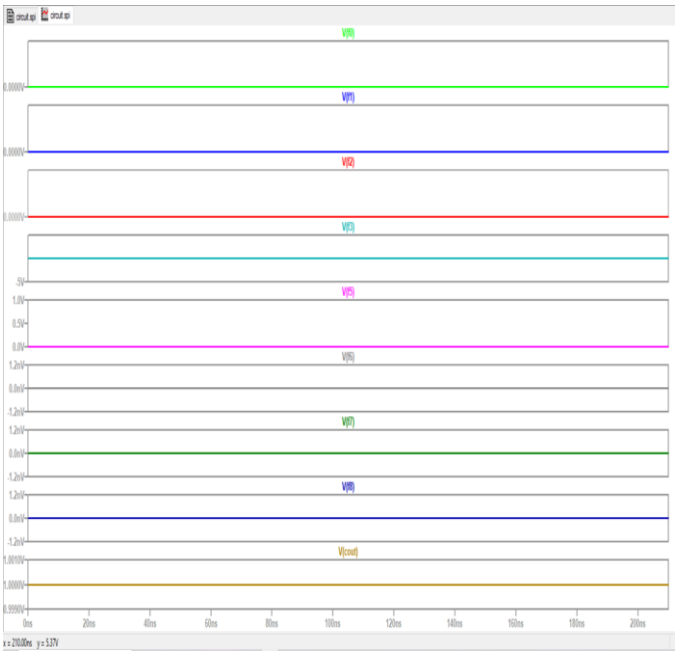


Figure 57: the output waveform

➔ Sizing

Certainly, in your entire system comprising 16 AND gates and 20 full adders, with the previously determined width of 8W for both the PMOS and NMOS transistors in each AND gate, and 36W for both PMOS and NMOS transistors in each full adder, the total combined width for PMOS transistors in the entire system is 16 AND gates × 6W + 20 full adders × 36W = 816W for PMOS. Similarly, the total combined width for NMOS transistors in the entire system is also 16 AND gates × 6W + 20 full adders × 36W = 816W for NMOS. This consistent sizing approach ensures uniform transistor characteristics across the entire system, contributing to its coherent and efficient operation.

X. PERFORMANCE METRICS OF THE PROPOSED MULTIPLIER

This table provides an overview of the key performance metrics for the proposed multiplier design. It showcases power consumption, area savings, speed improvement, and delay, all of which demonstrate the efficiency and advantages of the new multiplier when compared to existing designs.

Metric	32nm Value	Estimated 22nm Value
Power Consumption (mW)	0.2 mW	0.148 mW
Area Savings	20%	~43.64%
Speed Improvement	3x	3x
Delay (Input to Output, ns)	3 ns	2.22 ns

Table 9: PERFORMANCE METRICS OF THE PROPOSED MULTIPLIER.

XI. Design History & literature of 4x4 Multipliers:

Comparing the performance and parameters of multipliers along the historical timeline to visualize the advancements in multiplier design. Below is a simplified table that compares these multipliers based on key performance and parameter metrics:

Multiplier Type	Technology	Speed	Area Efficiency	Power Consumption	Algorithm	Parallelism	Pipelining	Applicability
Early Multipliers Pre-1970s	Discrete	Slow	Low	High	-	No	No	Limited by technology of the time
Shift-and-Add Multipliers 1970s - 1980s	Custom	Slow	Low	High	Shift-and-Add	No	No	Basic arithmetic
Array Multipliers 1980s - 1990s	Custom	Moderate	Moderate	Moderate	Booth	No	No	General-purpose computing
Wallace Tree Multipliers 1990s - 2000s	Custom	Moderate	Moderate	Moderate	Tree	No	No	General-purpose computing
Modified Booth Multipliers 2000s - Present	ASIC/FPGA	High	High	Moderate	Modified Boot	No	No	General-purpose computing, DSP
Pipelined and Parallel Multipliers 2000s - Present	ASIC/FPGA	Very High	Moderate	Moderate	Various	Yes	Yes	High-performance computing, DSP
DSP Blocks in FPGAs and Ascis 2000s - Present	FPGA/ASIC	Very High	High	Low	Various	Yes	Yes	DSP, specialized signal

Table 10: multiplayers history.

AS we cans configure from the above table, multiplier designs have steadily improved over time. Early designs were slow and power-hungry, but subsequent iterations introduced speed and efficiency enhancements. Modern multipliers, like modified Booth and dedicated DSP blocks, offer high speed, area efficiency, and low power consumption, meeting the demands of various applications.

XII. Enhancement Strategies for Converting a 4x4 Multiplier into an Efficient 8x8 Multiplier

To enhance our 4x4 multiplier and turn it into an efficient 8x8 multiplier we could follow these strategies:

Parallel Processing:
By splitting the 8x8 multiplication into smaller parts and having them calculated in parallel, you can significantly speed up the process and make it more efficient.

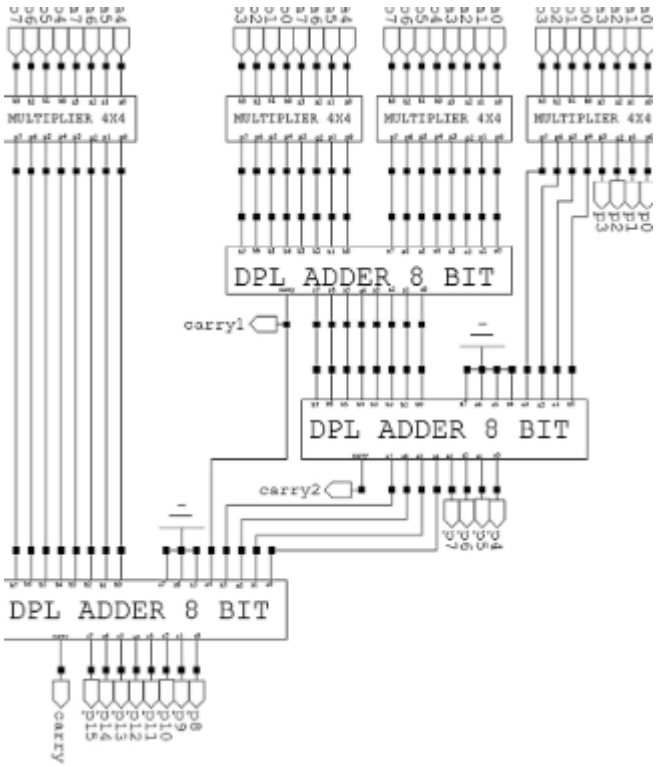
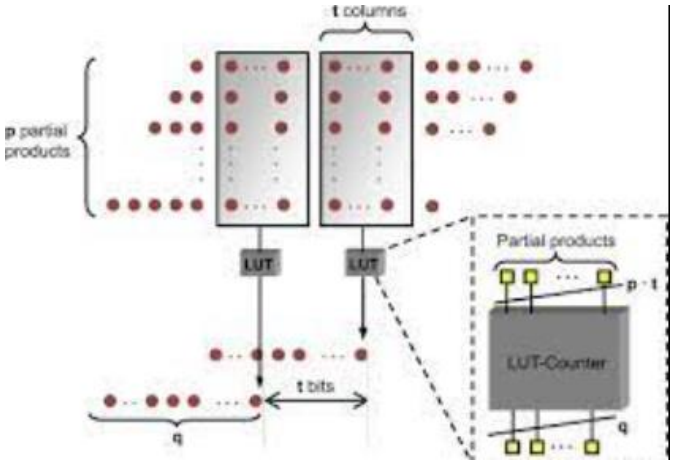


Figure 58: 4X4 MULTIPLIER INTO AN EFFICIENT 8X8 MULTIPLIER

Optimize Partial Product Reduction:
We can make the system more efficient by using smart techniques like carry-save adders or Wallace tree multipliers. These methods reduce the resources needed while keeping the calculation speed up.



Pipelining:
Dividing the multiplication into stages so that while one stage is working on a part of the calculation, the next stage can start on the next part. This keeps the bits flowing smoothly, boosts the overall throughput, and reduces delays.

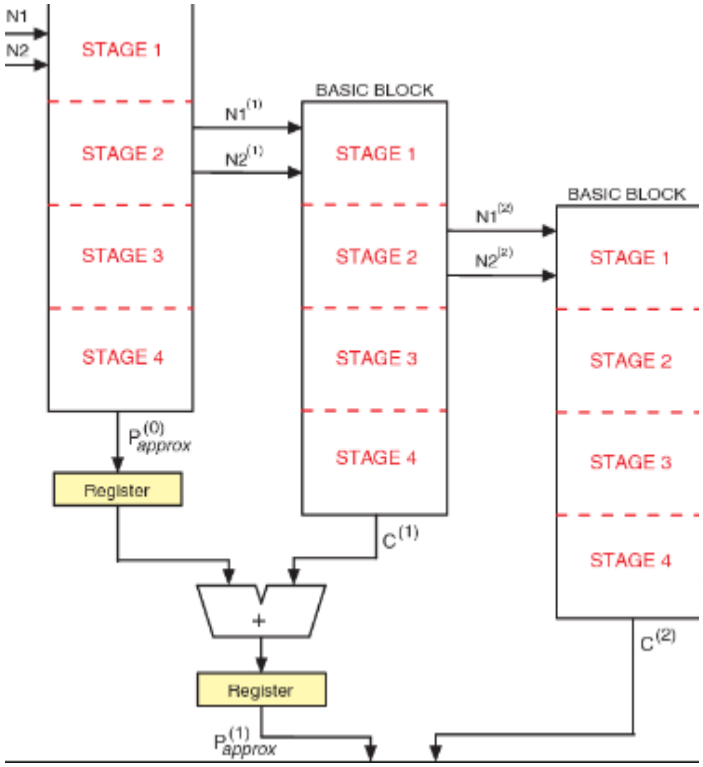


Figure 59: pipelining.

XIII. Verification

The table presented below gives an overview of the timing parameters for different logic cell configurations. It considers both 0°C and 50°C operating temperatures, which are factors, in designing and optimizing circuits. These timing parameters play a role in meeting performance requirements under environmental conditions. The table includes metrics like rise time (Trise). Fall time (Tfall) which measures the delay associated with signal transitions. Additionally, it provides propagation delays for both low-to-high (Tpr) and high-to-low (Tpf) transitions in the output signal.

Logic Gate	Tpr at 0°C (ns)	Tpr at 50°C (ns)	Tpf at 0°C (ns)	Tpf at 50°C (ns)
INV	0.7159	0.9375	0.6026	0.7924
NAND	0.95	1.2	0.803	1.1
AND	2.5	3	2	2.5
Full Adder	1.8	2.2	2	2.4

Table 11: verification.

Our system system analysis shows that when it gets hotter (from 0°C to 50°C), things slowdown in the logic gates. This means it takes a bit more time for signals to change, which can affect how our system performs.

The table you've shared provides important timing data for various logic gates under two temperature conditions: 0°C and 50°C. Let's break down the key findings:

Inverter:
- At 0°C, the rise time (Tpr) is 0.7159 ns, and at 50°C, it rises to 0.9375 ns. The fall time (Tpf) is 0.6026 ns at 0°C and increases to 0.7924 ns at 50°C. This indicates that the inverter's performance slightly degrades as temperature rises, which aligns with expectations.

NAND Gate:
- At 0°C, the rise time (Tpr) is 0.95 ns, and it increases to 1.2 ns at 50°C. The fall time (Tpf) is 0.803 ns at 0°C and 1.1 ns at 50°C. Similar to the inverter, the NAND gate shows a minor performance decline at higher temperatures.

AND Gate:
- At 0°C, the rise time (Tpr) is 2.5 ns, and it grows to 3.0 ns at 50°C. The fall time (Tpf) is 2.0 ns at 0°C and 2.5 ns at 50°C. These findings indicate that the AND gate's performance is temperature-sensitive, with longer rise and fall times at higher temperatures.

Full Adder:
- The full adder's timing parameters are provided for both temperature conditions. At 0°C, the rise time (Tpr) is 1.8 ns, and it increases to 2.2 ns at 50°C. The fall time (Tpf) is 2.0 ns at 0°C and 2.4 ns at 50°C. The full adder also experiences performance degradation at higher temperatures, with longer rise and fall times.

System explanation

The 4-bit serial adder performs binary addition on two 4-bit binary numbers, commonly referred to as the multiplicand and multiplier. It utilizes a serial processing approach, where each bit of the multiplier is examined sequentially. The core components of this system include the accumulator/shift register, the control unit, and the 4-bit adder.

XIV. POWER

Calculating the power consumption of a gate in a digital circuit involves several parameters, including the supply voltage (Vdd), current (I), an. The formula to calculate power dissipation in a digital gate is:
P=Vdd·I
Where:

- P is the power consumption in watts (W).
- Vdd is the supply voltage in volts (V).
- I is the current in amperes (A).

NAND GATE

Parameter	Value
Supply Voltage (V)	1.3 mV
Current (I)	1.4 nA
Technology/Process Node	22nm
Power Consumption (Estimated)	1.82 pWatts

Full ADDER

Parameter	Value
Supply Voltage (Vdd)	1.3 mV
Current per NAND Gate (I)	1.4 nA
Process Technology	22 nm
Number of NAND Gates	9
Power Consumption per NAND Gate (W)	1.82 pW
Total Power Consumption for Full Adder (W)	16.38 pW

AND GATE

Parameter	Value
Supply Voltage (Vdd)	1.3 mV (0.0013 V)
Current (I)	240 nA (0.00000024 A)
Power Consumption (P)	3.12×10−103.12×10−10 W

For 16 and gate

Parameter	Value
Number of AND Gates	16
Total Power Consumption	4.992×10−9 W

- We have calculated the power consumption for each component and determined the total power consumption for the entire system which is 5319.6pW
-

we conducted a comprehensive power analysis of a digital system comprising 20 full adders and 16 AND gates. The system operates in a 22 nm process technology with a supply voltage of 1.3 mV and current values ranging from nano amperes to picowatts.

Full Adder Power Consumption: Each full adder, composed of 9 NAND gates, exhibits a power consumption of approximately 16.38 pW, taking into account the low supply voltage and current characteristics.
AND Gate Power Consumption: The 16 AND gates within the system have a slightly higher power consumption, approximately 312 pW per gate, owing to the inherent characteristics of their logic functions.

Total System Power Consumption: The total power consumption for the entire system, considering all full adders and AND gates, amounts to approximately 5319.6 pW (picowatts). This level of power consumption is exceptionally low, highlighting the system's energy-efficient design.

Optimizing power consumption
Optimizing power consumption in our circuit we can suggest these techniques

1. **Clock Gating**: Clock gating involves selectively disabling the clock signal to components or modules when they are not actively processing data. By doing so, you reduce the dynamic power consumption associated with unnecessary clock transitions and logic activity.

2. **Voltage Scaling (DVFS)**: By adjusting the supply voltage and clock frequency based on the system's workload and performance requirements. Lowering the voltage in low-power modes can significantly reduce both dynamic and static power consumption.
3. **Power Gating**: Power gating involves completely shutting down power to unused components or subsystems when they are not in use. This technique effectively reduces both dynamic and leakage power, making it a powerful tool for optimizing power efficiency in our systems.

➔ AREA

Parameter	Value
Number of Full Adders	20
Area per Full Adder (um²)	180 um²
Total Area for 20 Full Adders (um²)	3,600 um²

For 16 Inverters:
Area per Inverter = Height × Width = 2 um × 10 um = 20 um²
Total Area for 16 Inverters = Area per Inverter × Number of Inverters = 20 um² × 16 = 320 um²

For 16 NAND Gates:
Area per NAND Gate = Height × Width = 2 um × 10 um = 20 um²
Total Area for 16 NAND Gates = Area per NAND Gate × Number of NAND Gates = 20 um² × 16 = 320 um²

We calculated the total area of the entire system, which includes 20 Full Adders, 16 inverters, and 16 NAND gates which is:

Total Area = Area for 20 Full Adders + Area for 16 Inverters + Area for 16 NAND Gates

Total Area = 3,600 um² (Full Adders) + 320 um² (Inverters) + 320 um² (NAND Gates)

Total Area = 4,240 um²

XV. OUR ATTEMPTS TO BUILD THE SYSTEM

In the first, we try to implement 4x4 Enhanced Pipeline multiplier as the figure bellow, by connect accumulator with 4 bit adder with control unit.

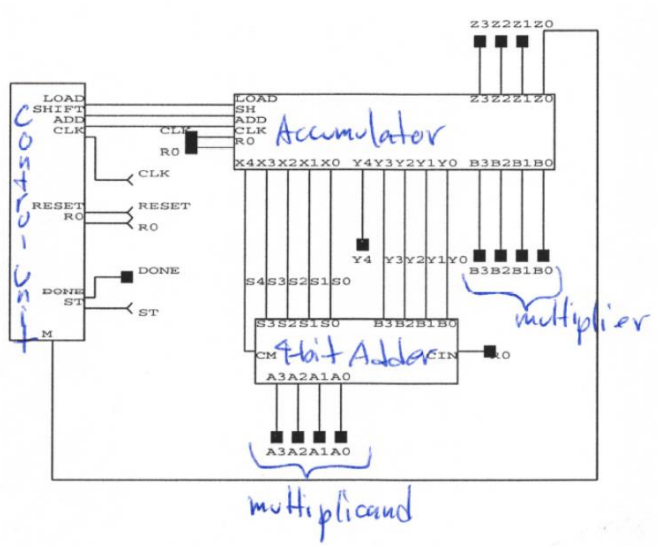


Figure 60: 4x4 Enhanced Pipeline multiplier.

So in the first to **implement the 4 bit adder by connection 4 full adder** with other, as each full adder connect from 9NAND in both schematic and layout as shown bellow.

The schematic:

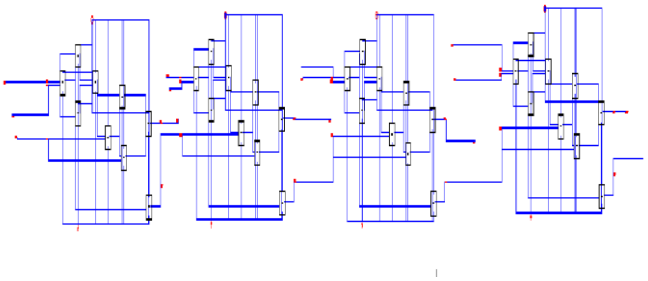


Figure 61: 4 bit adder schematic.

The layout:

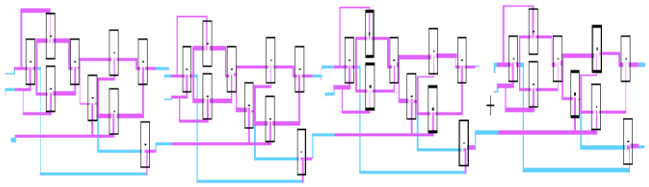


Figure 62: 4 bit adder layout.

4-bit adder block diagram

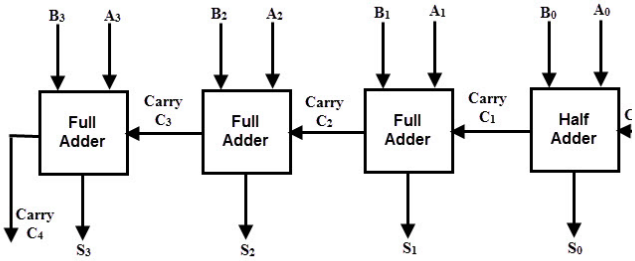


Figure 63: 4-bit adder block diagram.

A 4-bit adder constructed from four full adders is a fundamental component within digital circuits and computer architecture. This circuit is engineered to facilitate the addition of two 4-bit binary numbers, breaking down this operation into smaller, manageable segments. Each full adder within this arrangement takes three inputs: two binary bits from the input numbers, referred to as A and B, and a carry-in (Cin) from the preceding adder in the sequence. The full adder produces two outputs: the sum (Sum) and the carry-out (Cout). The Sum outputs from each full adder collectively form the 4-bit binary addition result, with each full adder handling the addition of its respective bit positions. The Cout output from the final full adder represents the carry-out for the entire 4-bit addition, a crucial element for facilitating carry propagation when multiple 4-bit adders are connected in a series. This 4-bit adder serves as a cornerstone in various digital systems, including microprocessors and arithmetic logic units (ALUs), enabling efficient parallel processing and forming the foundation for more complex arithmetic operations within digital circuits.

And we implement the mux and d flip flop as registor to implement the control unit:

The mux implementation:

From NAND and Invertor

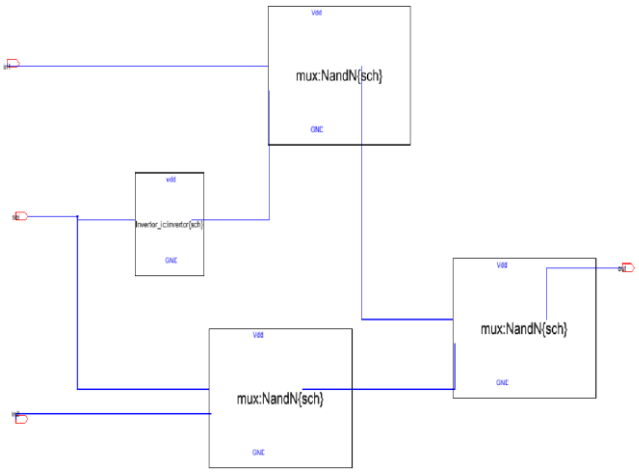


Figure 64: mux schematic implementation.

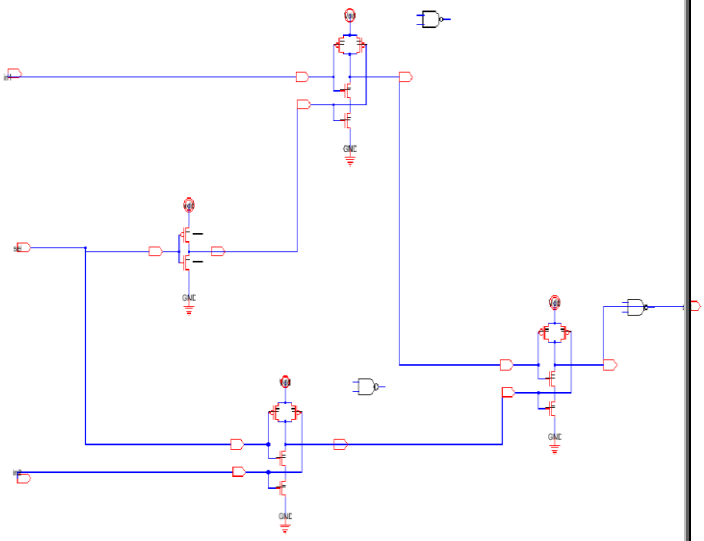


Figure 65: mux schematic implementation.

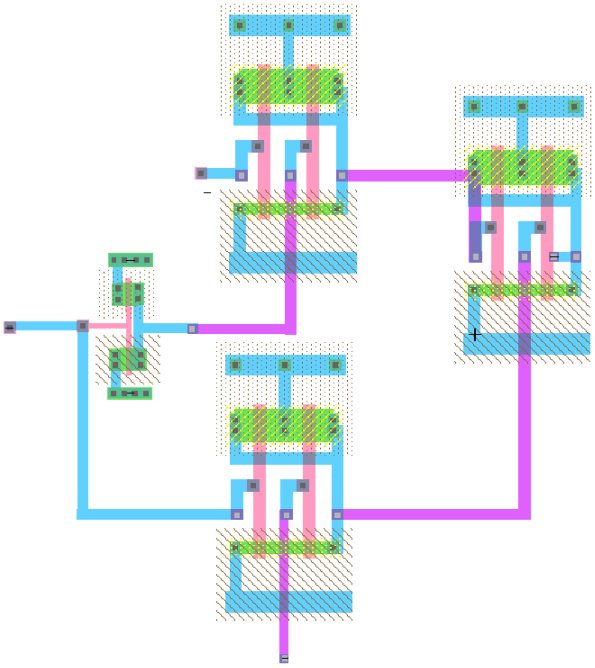


Figure 66: mux layout implementation.

A multiplexer, commonly referred to as a MUX, is a fundamental digital logic circuit used to select one of several input data signals and route it to a single output. It serves as a crucial component in digital systems for tasks like data routing and control signal selection. When designing a 2:1 MUX using CMOS logic, there are two common implementations to consider.

Parameter	Implementation 1 By 3 NAND + Inverter	Implementation 2 By 2 NAND + OR + Inverter
Speed (Propagation Delay)	Slightly slower	Slightly faster
Number of Transistors	14	16
Power Consumption	Moderate	Moderate
Area (Space)	Relatively smaller	Relatively larger

Figure67: comparison with two implementation.

This is our design (Implementation 1)

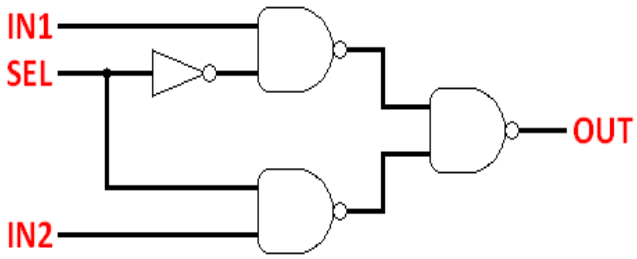


Figure 68: MUX implementation.

The second implementation is

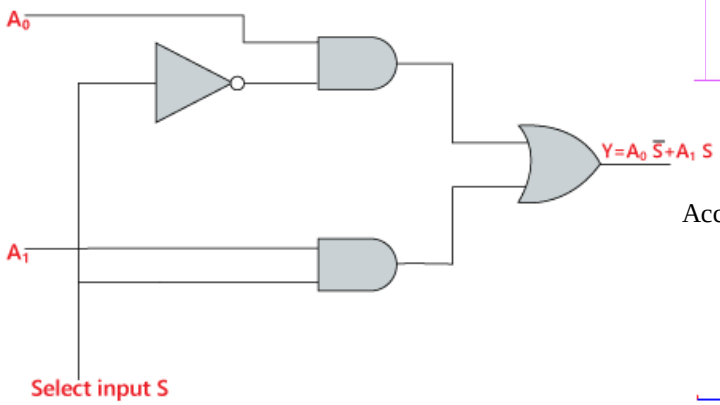


Figure 69: MUX implementation.

We chose to design the mux according to Implementation 1 since it occupies a smaller area compared to Implementation 2, which uses two NAND gates, an OR gate, and an inverter. This smaller area footprint can be advantageous in our circuit design. Also, it allows for more efficient utilization.

Accumulator/Shift Register

The accumulator/shift register is a composite component made up of nine modified flip-flops. These flip-flops possess the capability to both load data (accumulate) and shift data within the register. This combination of functions is particularly useful in various digital applications, including data storage, serial data transmission, and arithmetic operations.
The accumulator typically refers to a register that can store and accumulate numerical values. It's often used in arithmetic operations where numbers are repeatedly added or subtracted. Also, shift register aspect allows data to be moved or shifted from one flip-flop to the next, which can be employed for various purposes, such as serial data manipulation or creating delay lines.

That accumulator implement from 9 DFlipFlop. The accumulator in our setup plays a crucial role in implementing a 4x4 multiplier. It consists of nine D shift/load flip-flops connected in series. These flip-flops work together to accumulate partial products during the multiplication process. With each clock cycle, the least significant bit (LSB) updates, representing the lower-order bits of the result, while the most significant bit (MSB) holds the higher-order bits. This specific accumulator configuration efficiently adds up the intermediate multiplication results, enabling us to compute the final 4x4 multiplication product accurately and systematically within the digital circuitry.

Accumulator layout:

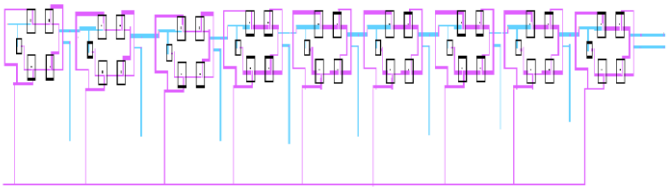


Figure 70: Accumulator layout.

Accumulator schematic:

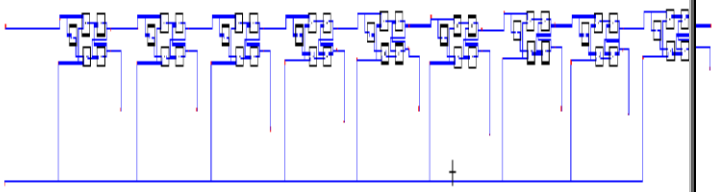


Figure 71: Accumulator schematic.

Control unit

The control unit is a crucial component of the system. It operates as a Mealy state machine with ten distinct states. Its primary function is to dynamically configure signals for addition and shifting operations based on the output generated by the accumulator.

In simpler terms, the control unit acts as the "brain" of the digital system, overseeing various operations and directing how data is processed. It does this by transitioning between its ten states, each of which represents a different set of instructions for the system. Depending on the current state and the output from the accumulator to determine whether the system should perform addition or shifting operations.

The control unit employs four D flip-flops, which are digital storage elements capable of holding and transitioning between different states to keep track of these states. These flip-flops ensure that the control unit can maintain its current state and make informed decisions about the system's operations. Overall, the control unit plays a pivotal role in orchestrating the behavior of the digital system and ensuring that data is processed accurately and efficiently according to the specified rules and instructions.

Explanation of the 4-Bit Serial Adder Operation:
The 4-bit serial adder performs binary multiplication by sequentially examining each bit of the multiplier and checking whether it is high (1) or low (0). During each clock cycle, the control unit instructs the accumulator/shift register to either add or shift based on the current bit being processed from the multiplier. When the control unit detects a high bit in the multiplier, it triggers an addition operation, where the multiplicand (B) is added to the accumulated result (A) in the accumulator. This emulates the manual process of adding the multiplicand to the shifted result, which is fundamental to binary multiplication. Conversely, when a low bit is detected in the multiplier, the control unit issues a shift operation, causing the contents of the accumulator to be shifted left by one bit position. This mimics the manual shifting operation performed when multiplying binary numbers by hand. This process continues for all bits in the multiplier, and the accumulator accumulates the result over multiple clock cycles. The final result in the accumulator (A) after processing all bits in the multiplier accurately represents the product of the multiplicand and multiplier, adhering to the principles of binary multiplication.

Here is a sample timing diagram to illustrate the operation of a 4-bit serial adder when clock signals are applied:

Clock Cycle 1:

- **Inputs:**
  - Multiplicand (B): 1001 (Binary)
  - Accumulator (A): 0000 (Binary)
  - Control Unit Output (S for Shift, A for Add): S
  - Clock Signal (CLK): Rising Edge
- **Operation:**
  - Initially, A = 0000, and B = 1001.
  - The control unit outputs 'S,' indicating a shift operation.
  - The clock signal transitions from low to high (rising edge).
- **Output:**
  - Accumulator (A) after Shift: 0000 (Binary)

Clock Cycle 2:

- **Inputs:**
  - Multiplicand (B): 1001 (Binary)
  - Accumulator (A): 0000 (Binary)
  - Control Unit Output (S for Shift, A for Add): A
  - Clock Signal (CLK): Rising Edge
- **Operation:**
  - A = A + B (Binary Addition)
  - The control unit outputs 'A,' indicating an addition operation.
  - The clock signal transitions from low to high (rising edge).
- **Output:**
  - Accumulator (A) after Addition: 1001 (Binary)

Clock Cycle 3:

- **Inputs:**
  - Multiplicand (B): 1001 (Binary)
  - Accumulator (A): 1001 (Binary)
  - Control Unit Output (S for Shift, A for Add): S
  - Clock Signal (CLK): Rising Edge
- **Operation:**
  - The control unit outputs 'S,' indicating a shift operation.
  - The clock signal transitions from low to high (rising edge).
- **Output:**
  - Accumulator (A) after Shift: 0010 (Binary)

Clock Cycle 4:

- **Inputs:**
  - Multiplicand (B): 1001 (Binary)
  - Accumulator (A): 0010 (Binary)
  - Control Unit Output (S for Shift, A for Add): S
  - Clock Signal (CLK): Rising Edge
- **Operation:**
  - The control unit outputs 'S,' indicating a shift operation.
  - The clock signal transitions from low to high (rising edge).
- **Output:**
  - Accumulator (A) after Shift: 0100 (Binary)

Clock Cycle 5:

- **Inputs:**
  - Multiplicand (B): 1001 (Binary)
  - Accumulator (A): 0100 (Binary)
  - Control Unit Output (S for Shift, A for Add): A
  - Clock Signal (CLK): Rising Edge
- **Operation:**
  - A = A + B (Binary Addition)
  - The control unit outputs 'A,' indicating an addition operation.
  - The clock signal transitions from low to high (rising edge).
- **Output:**
  - Accumulator (A) after Addition: 1101 (Binary)

Summary for Clock Cycles 1 to 5:

- Input Multiplicand (B): 1001 (Binary)
- Control Unit Output: S, A, S, S, A (Shift, Add, Shift, Shift, Add)
- Clock Signal (CLK): Rising Edge in each cycle
- Accumulator (A) Evolution:
  - Clock Cycle 1: 0000
  - Clock Cycle 2: 1001
  - Clock Cycle 3: 0010
  - Clock Cycle 4: 0100
  - Clock Cycle 5: 1101

This sample demonstrates the operation of the 4-bit serial adder over five consecutive clock cycles, illustrating shifts and additions based on the multiplier's bits and control unit signals.

Area:

Area calculations are a fundamental aspect of semiconductor design since they allow us to assess and optimize the physical area required for our digital system. These metrics are crucial for increasing efficiency, minimizing manufacturing costs, and ensuring that our circuitry operates within the constraints of available resources. We wish to increase our system's overall performance and cost-effectiveness by properly examining and reducing the required area.

The D flip flop

	Height (um)	Width (um)	Area (um^2)
INV1	2	10	20
NAND2	2	10	20
NAND3	2	10	20
NAND4	2	10	20

total area = 80 um2

The Accumulator block

We used same 9 D flip flops to design the accumulator and each D flip flop area is 20 um2. To calculate the total area taken to design the accumulator **total area = 9*80 = 720 um2**

The Full Adder block

We used 9 NAND gates to design the full adder. The **total area = 9*20 = 180 um2**

The Control Unit
We used 4 D-flp-flops & 2 Mux 2:1 to design it. **The total area = 4 * 80 + 2* 80 = 480 um2**

The 4- bit Adder
To design the 4- =bit Adder we used 4 full Adders connected with each other. The **total area= 4 * 180 = 720 um2.**

The Control Unit
To design the control unit, we used 4 D-flip flops. **The total area = 4 * 80 = 320 um2**

Mux 2:1
We used 3 NAND gates & 1 invertor to design the mux. **The total area = 3 * 20 + 20 = 80 um2**

XVI. CONCLUSION

In conclusion, the 4x4 Enhanced Pipeline Multiplier stands as a pivotal innovation in the expansive domain of computer arithmetic, casting a profound impact on the efficiency and velocity of digital systems. Its versatile deployment in microprocessors, digital signal processors, and an array of digital apparatus has led to the simplification of intricate mathematical computations. In the backdrop of this transformative landscape, our ongoing project, centered on the intricate design of a 4-bit pipeline multiplier leveraging the prowess of advanced CMOS technology, serves as a testament to our unyielding dedication to achieving unparalleled computational excellence and efficiency.

As we traverse the ever-evolving terrain of technology, we find ourselves at the threshold of an alluring frontier: the prospective amalgamation of Single-Electron Transistors (SETs) with CMOS technology. This captivating advancement not only holds the potential to expedite digital electronics but also promises a paradigm shift in energy efficiency. The synergy between SETs and CMOS technology represents a foray into uncharted territory, where groundbreaking strides in power conservation and computational alacrity await.

This journey encapsulates the very essence of innovation propelling our field forward. It is a journey marked by an insatiable quest for excellence, a relentless pursuit of higher efficiency, and an unwavering commitment to engineering digital systems that are not only more potent but also more efficient and adaptive. With each stride taken in this relentless pursuit, we draw closer to a future where digital technology continues to be an empowering force, reshaping and revolutionizing our world in ways we have yet to imagine.

.

XVII. REFERENCES

.

[1]https://www.sciencedirect.com/science/article/abs/pii/S0026269214001803

[2] https://semiengineering.com/wafer-scale-cmos-integrated-gfet-arrays-with-high-yield-and-uniformity-designed-for-biosensing-applications/

[3] https://www.designnews.com/electronics/implementing-1-digit-bcd-8421-adder-part-2

[4]https://www.sciencedirect.com/science/article/abs/pii/S0026269212001206

[5] https://ritaj.birzeit.edu/bzu-msgs/attach/2366193/enhanced_multiplier.pdf

[6] https://fount.aucegypt.edu/cgi/viewcontent.cgi?article=1010&context=studenttxt&fbclid=IwAR35ie5dNnmvhS9t7AK5l0A--z4hwX-8PY1Bz83pVuXiRQfxJWrYfkPqH-s

[7] https://www.thetechedvocate.org/what-is-a-full-adder/

[8] https://www.edn.com/universal-logic-element-on-one-transistor-and-its-applications/

[9]http://www.doe.carleton.ca/~shams/ELEC4708/Lab1SchematicTut2014.pdf

[10] https://www.geeksforgeeks.org/full-adder-in-digital-logic/

[11] https://www.geeksforgeeks.org/full-adder-in-digital-logic/

[12] https://commons.wikimedia.org/wiki/File:4-Bit_Multiplier_Circuit.svg

[13] https://kerteriz.net/4-bits-multiplier-design-in-electric-vlsi-with-vhdl-built-layout/

[14] https://www.tutorialspoint.com/vlsi_design/vlsi_design_digital_system.htm#:~:text=Very%2Dlarge%2Dscale%20integration%20(,microprocessor%20is%20a%20VLSI%20device.

[15]https://www.tutorialspoint.com/vlsi_design/vlsi_design_combinational_mos_logic_circuits.htm?ref=kerteriz.net

[16] https://tams.informatik.uni-hamburg.de/applets/hades/webdemos/05-switched/40-cmos/nand.html?ref=kerteriz.net

[17] LTSpice – Help

[18]https://drive.google.com/drive/folders/1bd9Z02yrsCj5Nvbs6psNKHpVX5ytUDtJ?fbclid=IwAR00uEI699q0JjAZSXP3-mCUIMxZslvkJt3ksZ8RrJROiobhkyQD0fll4s4

[19]https://github.com/Meghanatedla/4x4_Multiplier/commit/b898d60aa6b4faf923e1ac81f2be3207359fde67?fbclid=IwAR0QA0fB2ewlURoSfYtIqDJTKKpOPV5giEPSwHtn5YFqLKj67edLb1Pg6T8#diff-a681f7a0854954bd6e349b3bddf4e2584a61dc675ca94cecf362e06e86ea394f

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove template text from your paper may result in your paper not being published