
# 4-Bit Arithmetic Logic Unit (ALU) by CMOS.

**Team:**

Adham Ahmed Hammoda — 18100846

Ahmed Mohamed Fouad — 18101467

**Under the Supervision of**

Dr. Ahmed H.Madian

Eng. Alaa Osama

Submitted for ECEN-302/312 Analysis and Design of Digital Circuits Course Project

# Contents

## Figures

# Introduction

In this report we will implement a 4-bit Arithmetic Logic Unit (ALU) using two different types of adders: The Ripple Carry Adder and Carry Lookahead Adder. Furthermore, the performance of the two ALUs, regarding the number of transistors, worst-case propagation delay, and maximum power consumption, must be evaluated and summarized in a table of comparison. The ALUs will be designed using CMOS logic with the following specifications:

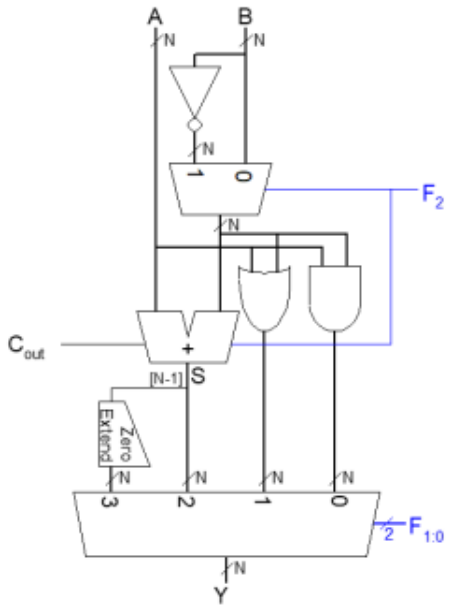
1) The supply voltage is 1.2V.

2) Use the minimum channel length that is allowed by the technology file.

3) The rise time and fall time of the ALU's inputs are100ps each.

4) Each logic gate must have symmetric propagation delays (TPHL=TPLH) under the worst-case scenarios. The propagation delay is the time between the point at which the input changes by 50% and the point at which the output changes by 50%.

5) The load capacitance of each output pin, Y(3:0), is 1pF.



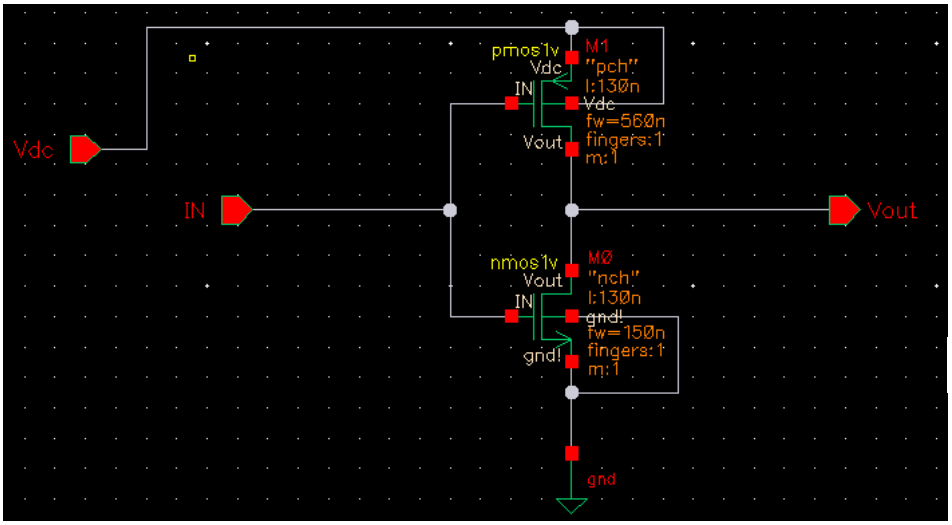

| $F_{2:0}$ | Function |
|---|---|
| 000 | A & B |
| 001 | A \| B |
| 010 | A + B |
| 011 | not used |
| 100 | A & ~B |
| 101 | A \| ~B |
| 110 | A - B |
| 111 | SLT |

*Figure 1 Functions of the ALU*

# Components of ALU

## Inverter Gate & 4-Inverter Gate

The inverter gate was implemented as follows (one PMOS along with one NMOS)



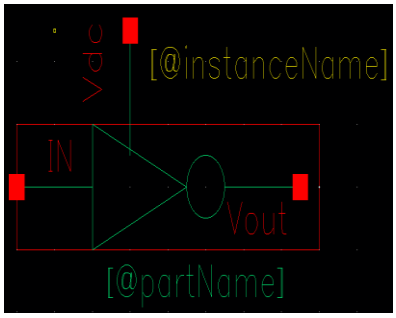
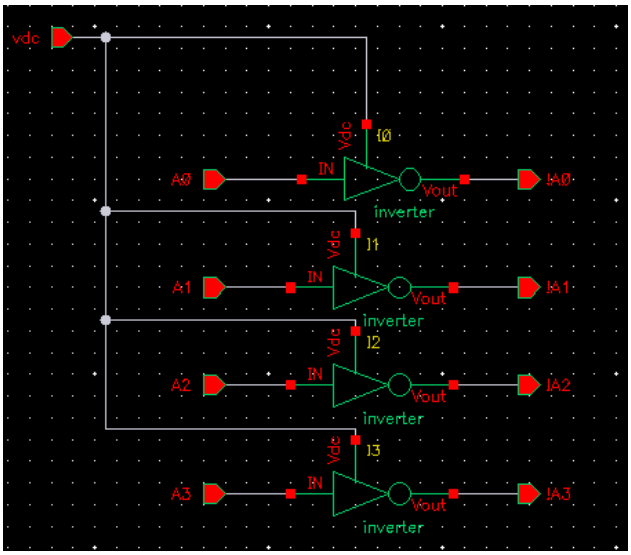




*Figure 2 Inverter Symbol*

*Figure 3 Inverter Circuit*

Then to make a 4-bit inverter we combined 4 inverters together.



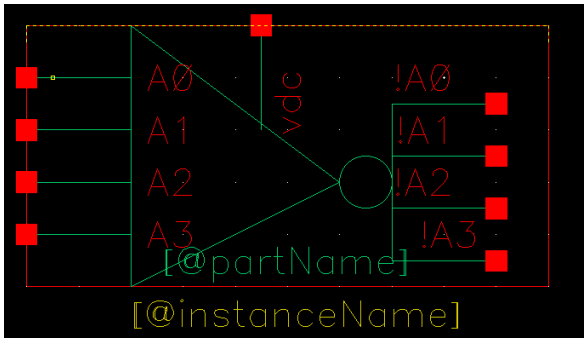




*Figure 4 4-inverter symbol*

*Figure 5 Four-inverter circuit*

We adjusted the sizing of the inverter to be w= 560n for PMOS and w=150n for NMOS.

## AND Gate & 4-AND Gate

The AND gate was implemented as follows (two parallel PMOS along with two series NMOS).

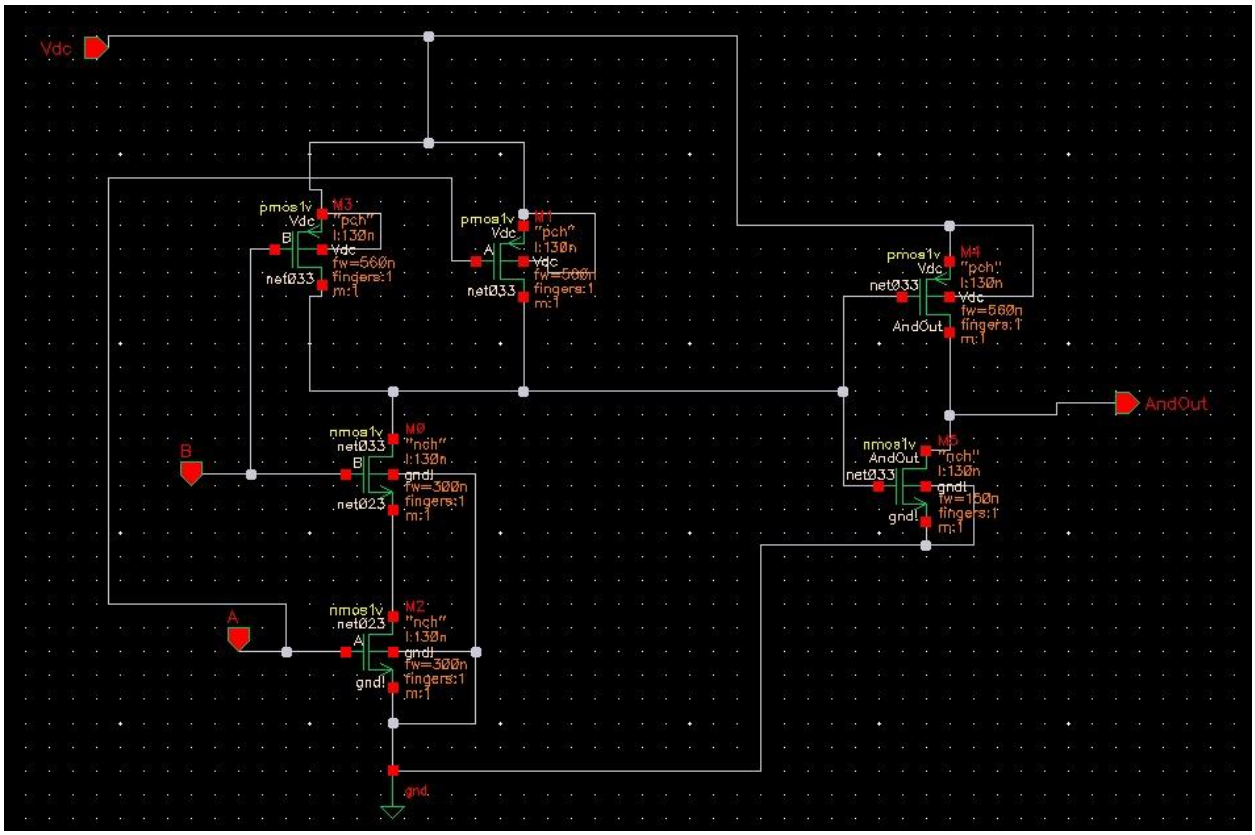



*Figure 6 A gate circuit*

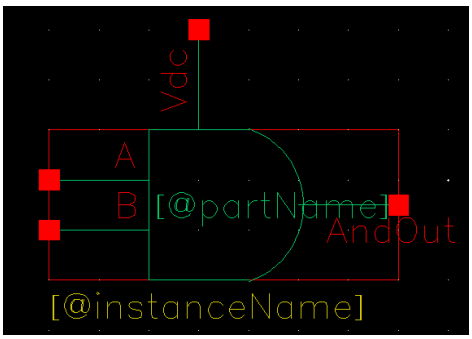



*Figure 7 AND gate symbol*

Then to design a 4-bit AND we combined 4 AND gates

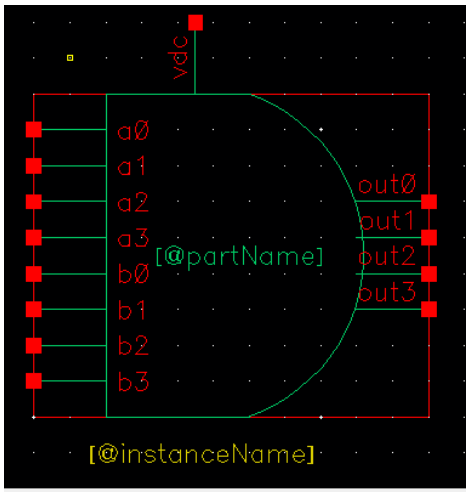



*Figure 8 Four-AND gate symbol*

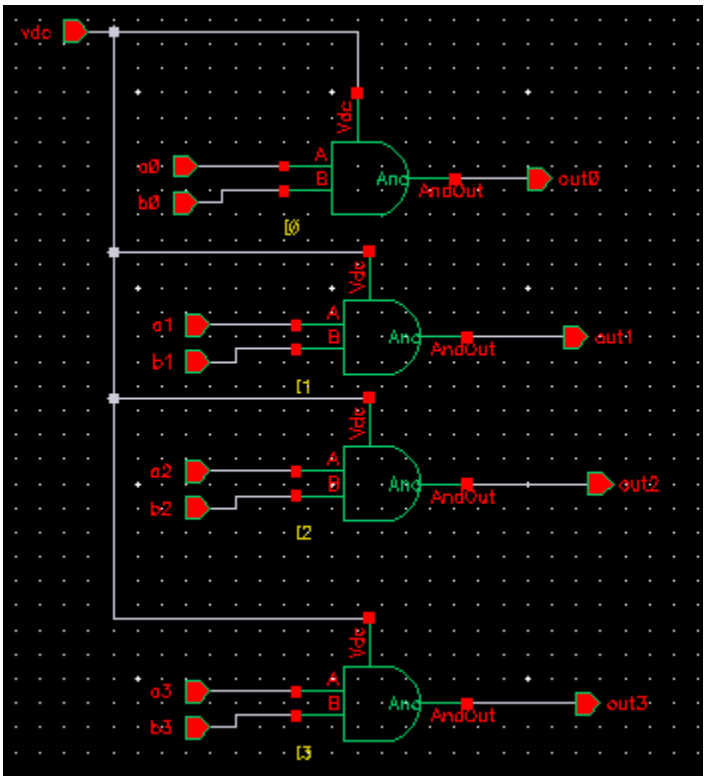



*Figure 9 Four AND gate circuit*

We adjusted the sizing of the AND gate to be w=560n for PMOS and w=300n for NMOS.

## OR Gate & 4-OR Gate

The OR gate was implemented as follows (two series PMOS along with two parallel NMOS)

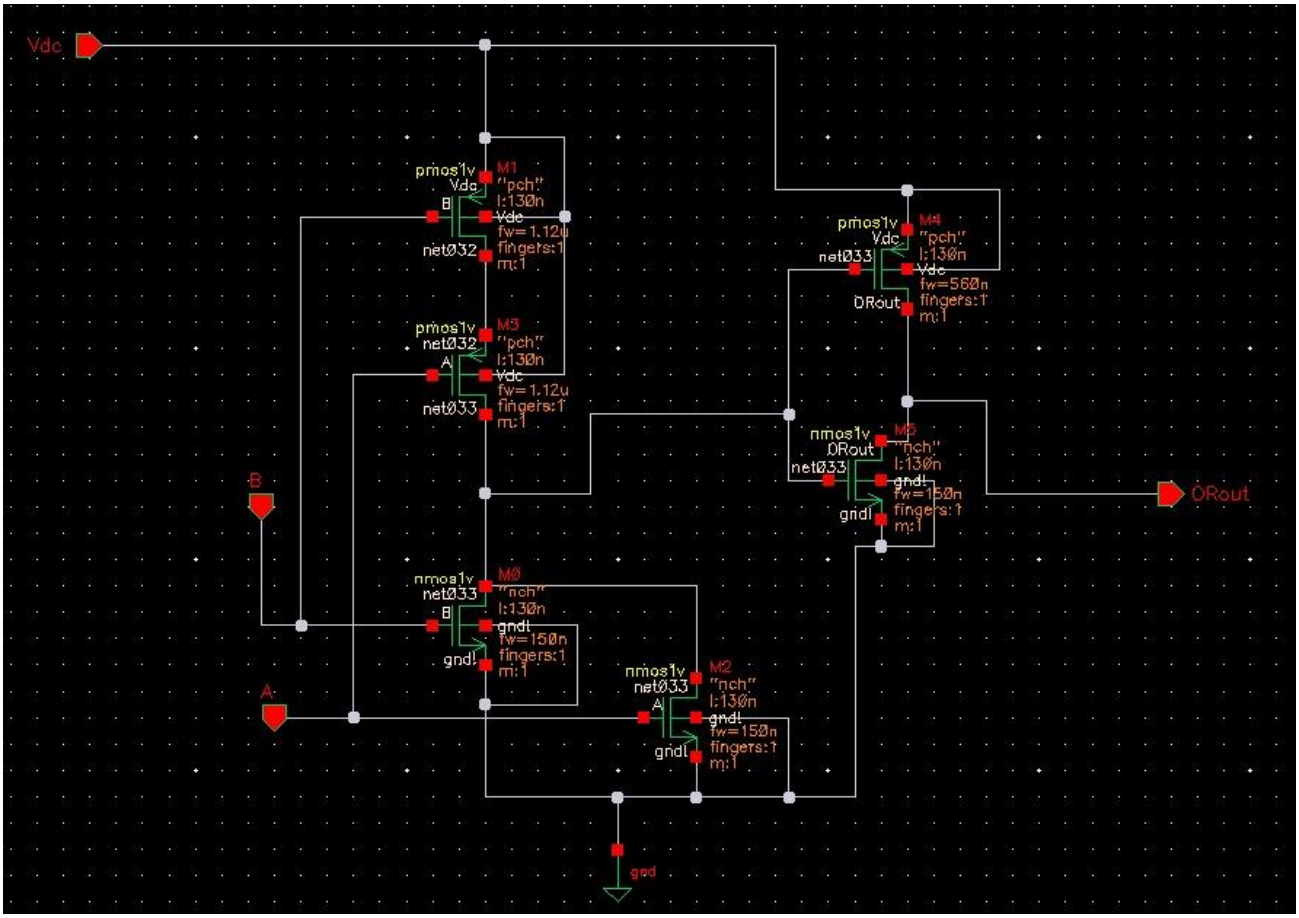



*Figure 10 OR Circuit*

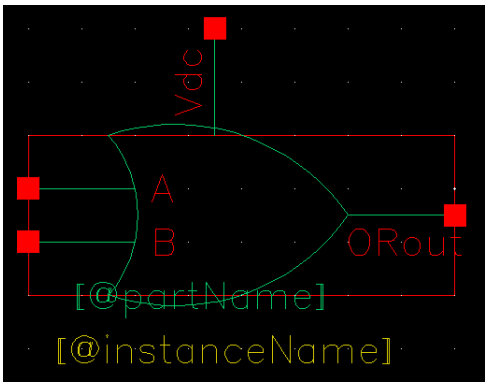



*Figure 11 OR Gate symbol*

Then to design a 4-bit OR we combined 4 OR gates

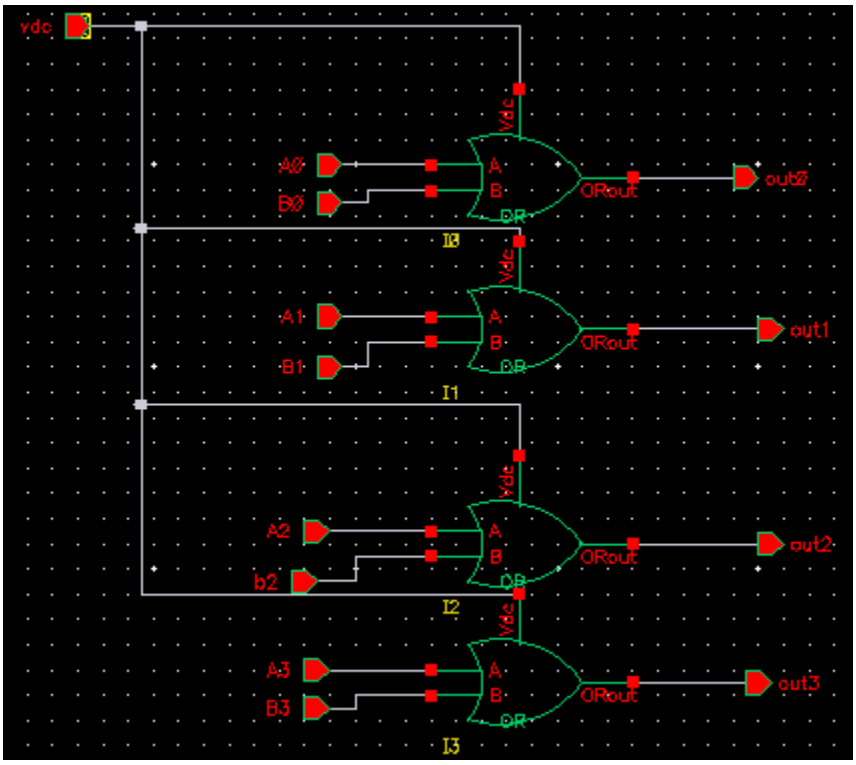



Figure 13 Four-OR circuit

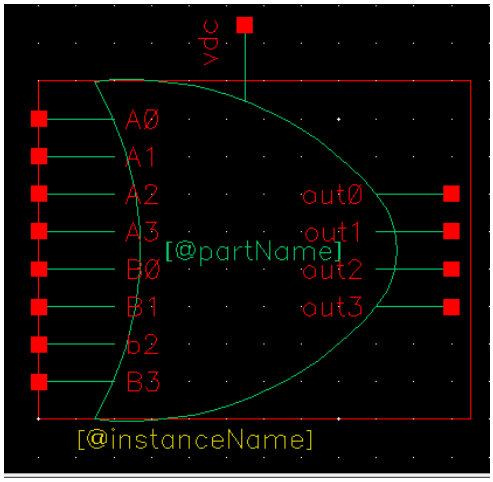



Figure 12 Four-OR Symbol

We adjusted the sizing of the gate to be w=1120n for PMOS and w=150n for NMOS.

## XOR Gate

The XOR Gate was implemented using two Inverter gates, two AND gates and one OR gate as shown below.

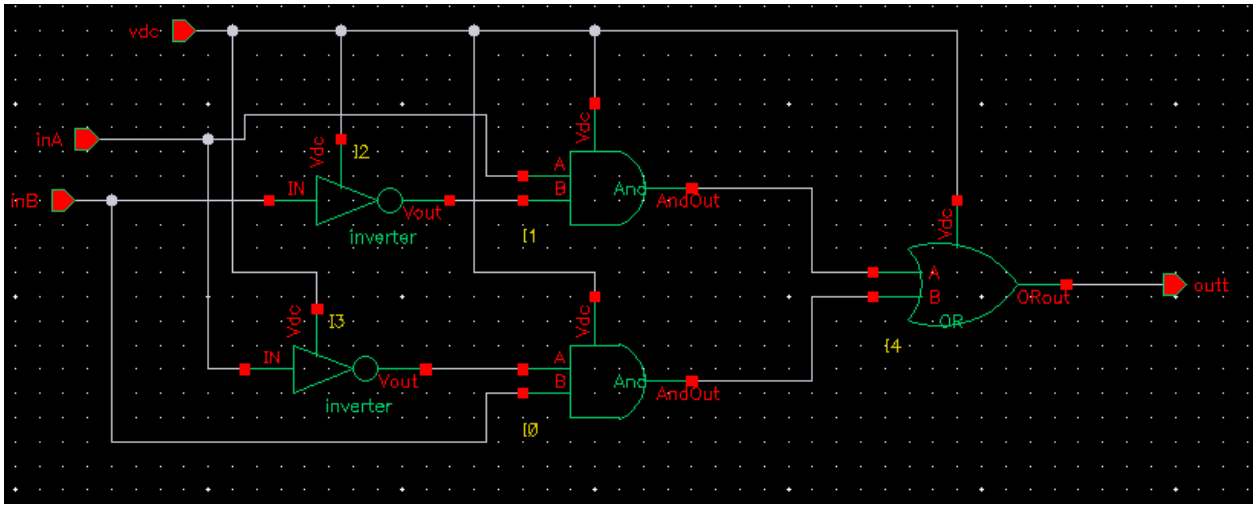



*Figure 14 XOR Circuit*

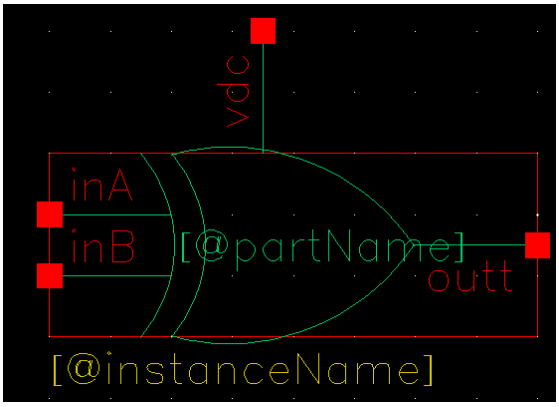



*Figure 15 XOR Symbol*

## 3-1 AND Gate, 4-1 AND Gate, 5-1 AND Gate

The 3-1 AND gate was designed as shown below (three parallel PMOS along with three series NMOS)

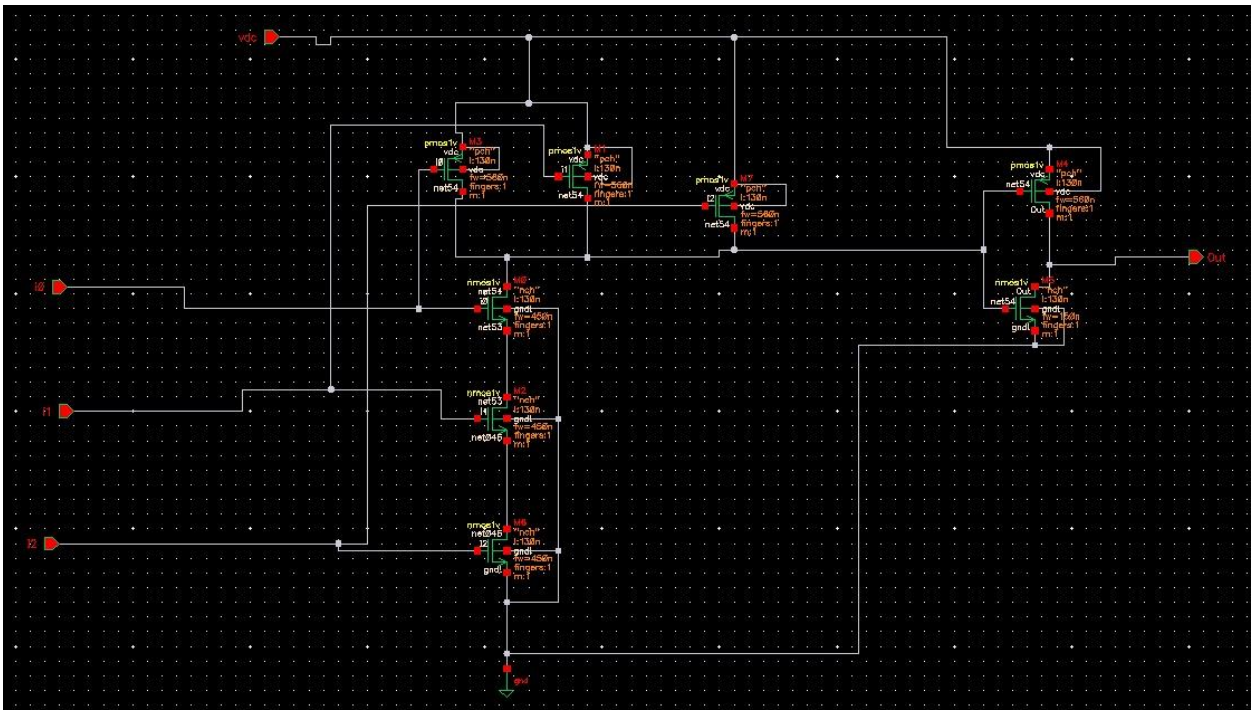


*Figure 16 3-1 AND circuit*

We adjusted the sizing of the gate to be

w=450n for NMOS and w=560n for PMOS

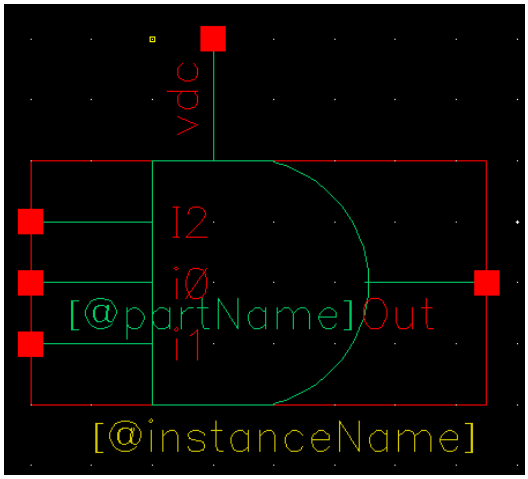



*Figure 17 4-AND symbol*

The 4-1 AND gate was designed as shown below (four parallel PMOS along with four series NMOS)

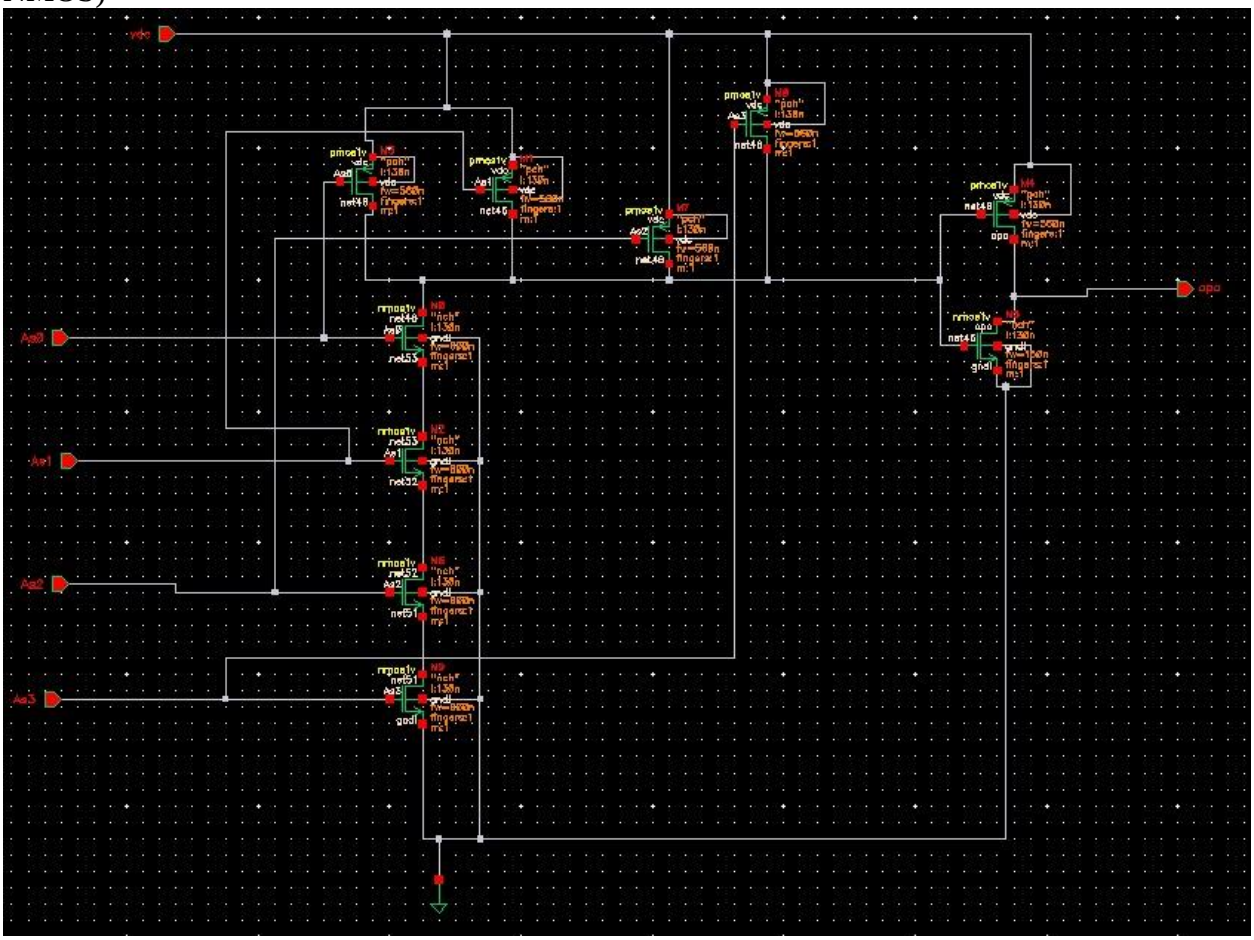


*Figure 18 4-AND circuit*

We adjusted the sizing of the gate to be

w=600n for NMOS and w=560n for PMOS

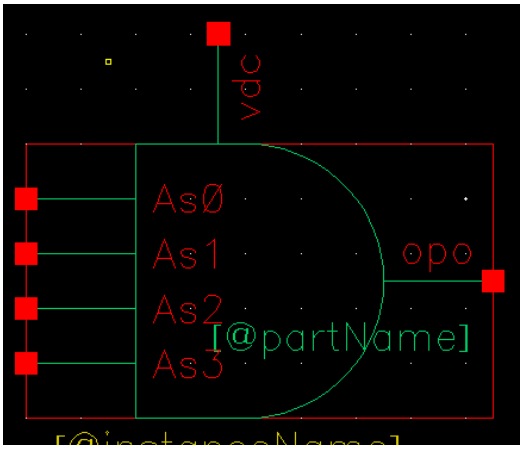



*Figure 19 4-AND symbol*

The 5-1 AND gate was designed as shown below (five parallel PMOS along with five series NMOS)

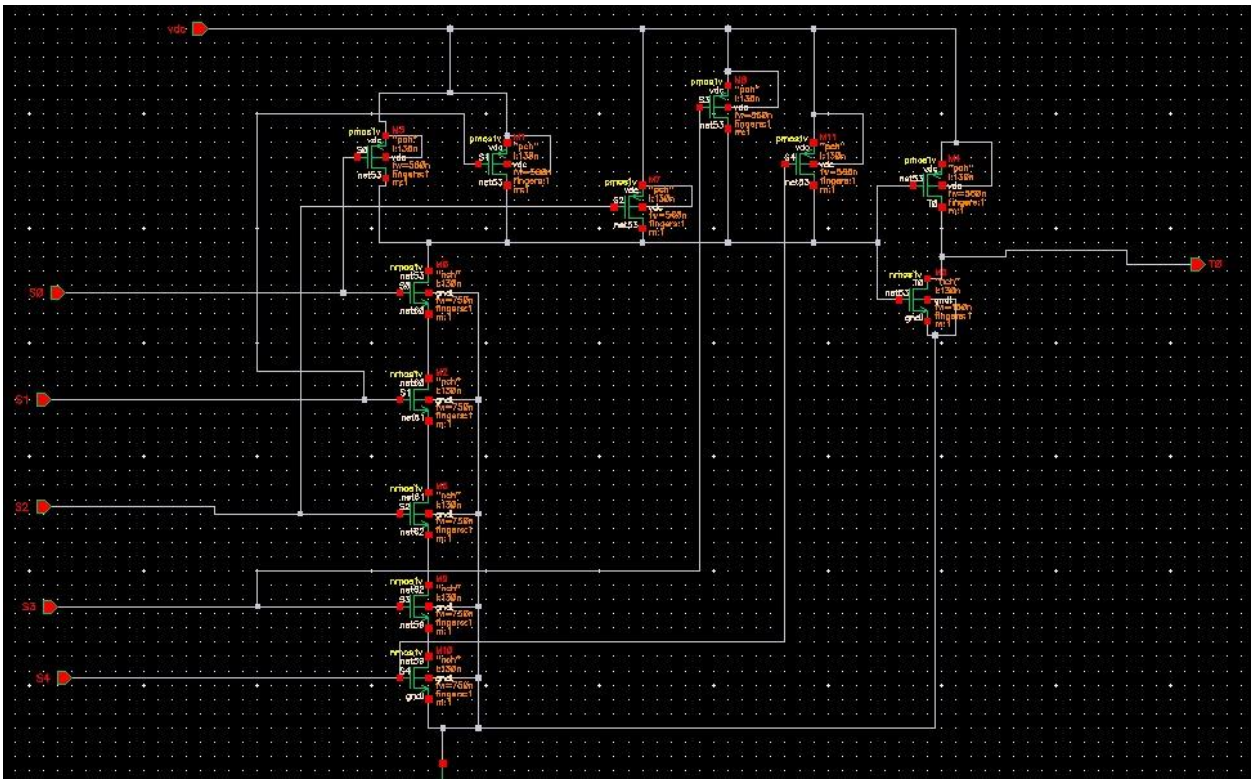


*Figure 20 5-1 AND gate circuit*

We adjusted the sizing of the gate to be

w=750n for NMOS and w=560n for PMOS

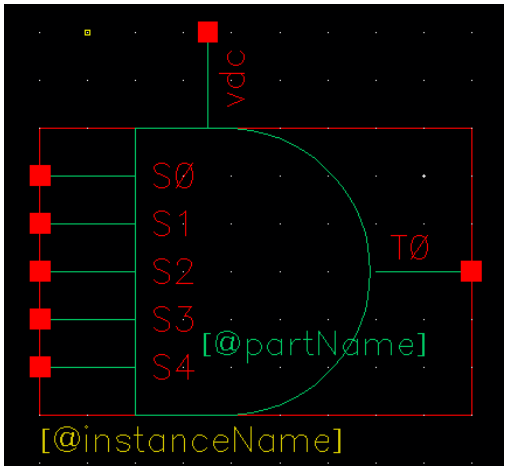



*Figure 21 5-1 AND symbol*

## 3-1 OR Gate, 4-1 OR Gate, 5-1 OR Gate

The 3-1 OR gate was implemented as follows (three series PMOS along with three parallel NMOS)

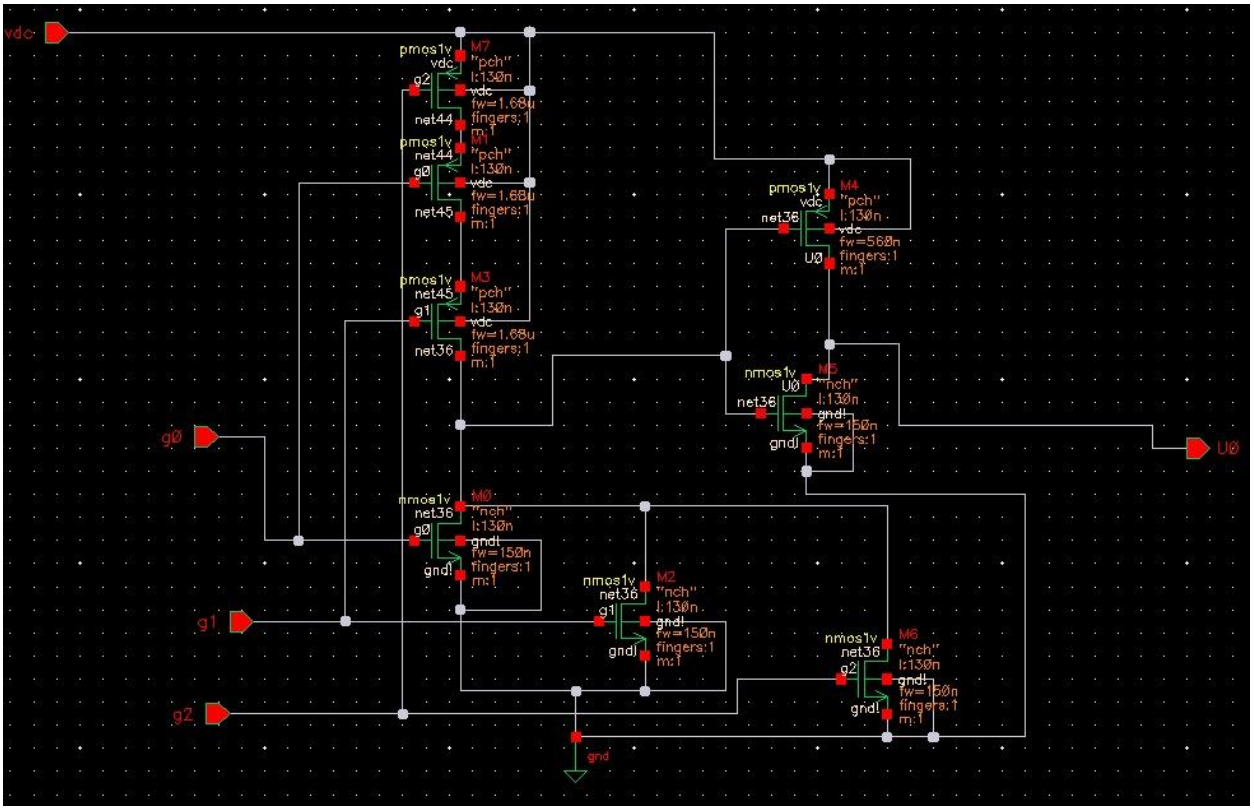



*Figure 22 3-1 OR Circuit*

We adjusted the sizing of the gate to be

w=150n for NMOS and w=1680n for PMOS

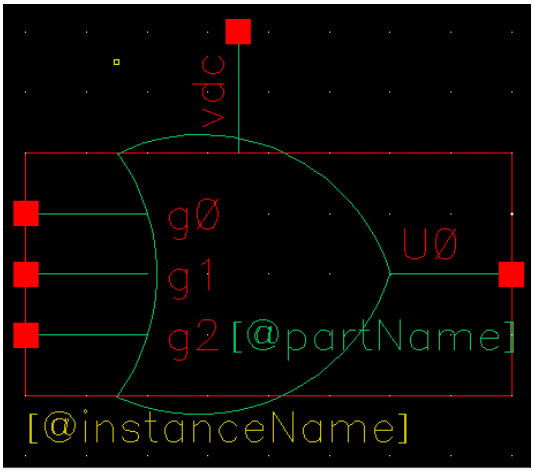



*Figure 23 3-1 OR gate symbol*

The 4-1 OR gate was implemented as follows (four series PMOS along with four parallel NMOS)

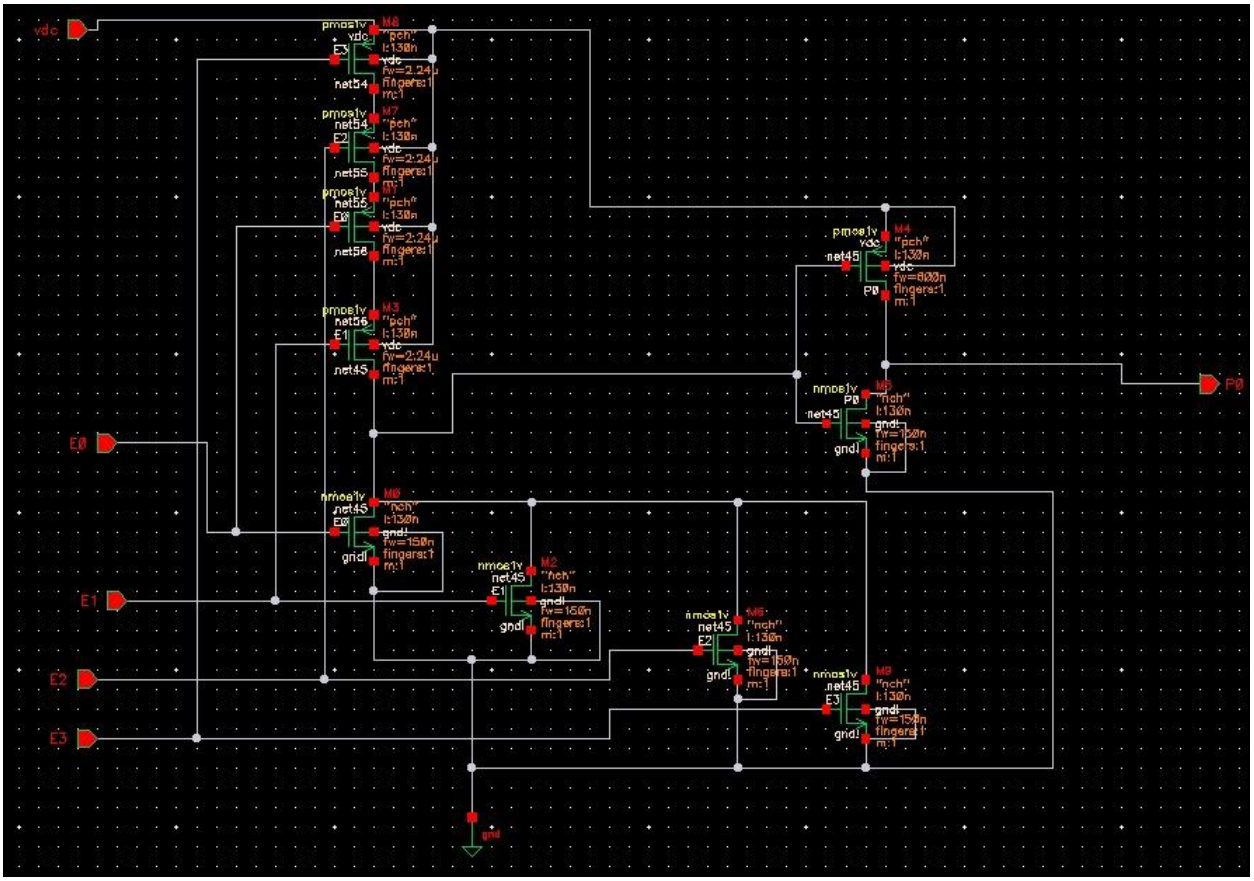



*Figure 24 4-1 OR Gate circuit*

We adjusted the sizing of the gate to be

w=150n for NMOS and w=2240n for PMOS

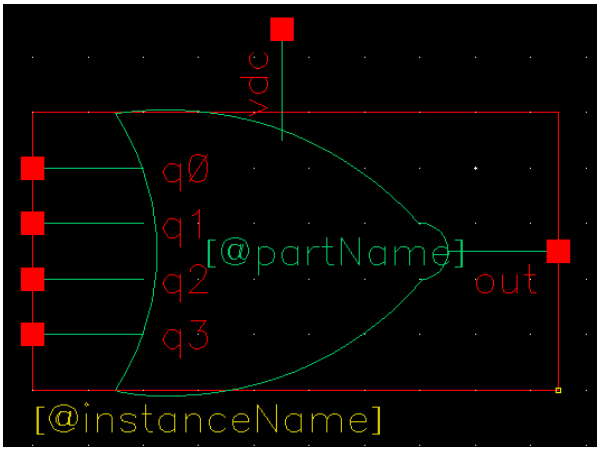



*Figure 25 4-1 OR symbol*

The 5-1 OR gate was implemented as follows (five series PMOS along with five parallel NMOS)

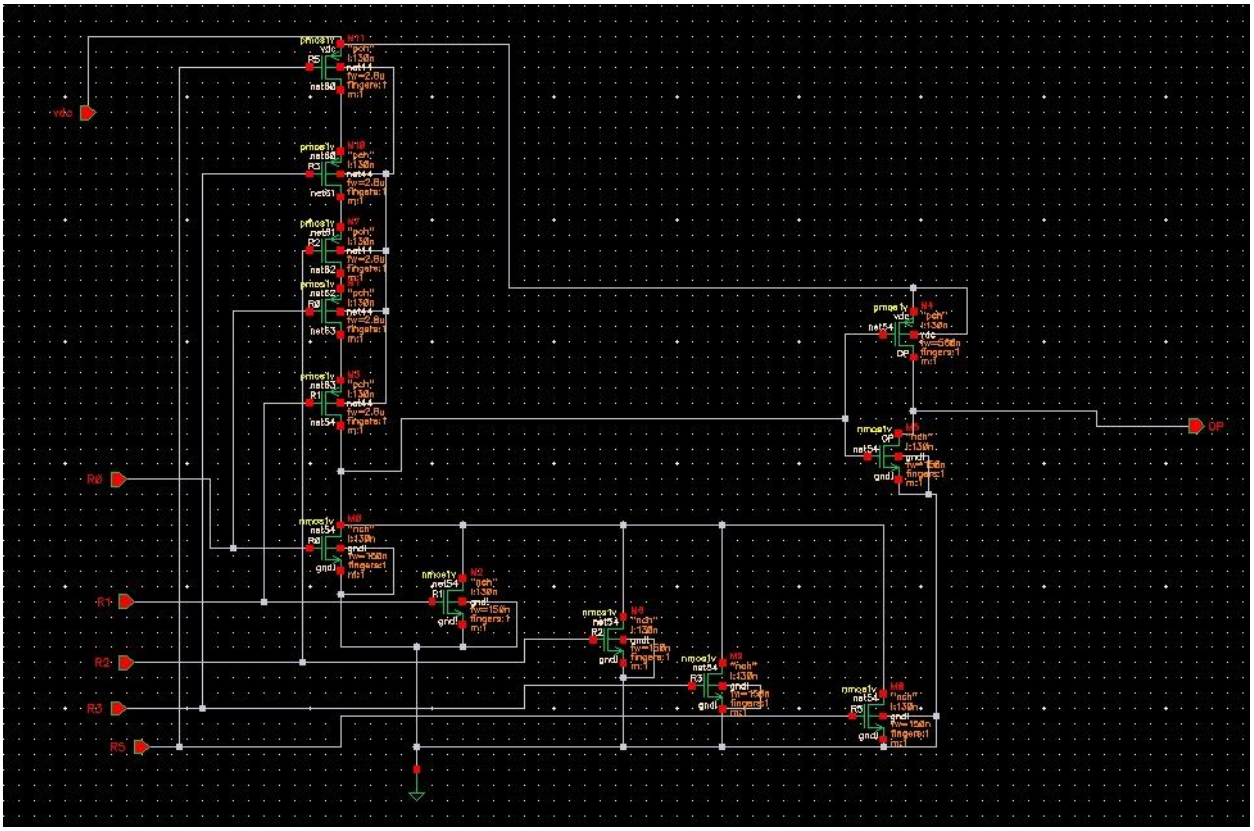


*Figure 26 5-1 OR circuit*

We adjusted the sizing of the gate to be

w=150n for NMOS and w=2800n for PMOS

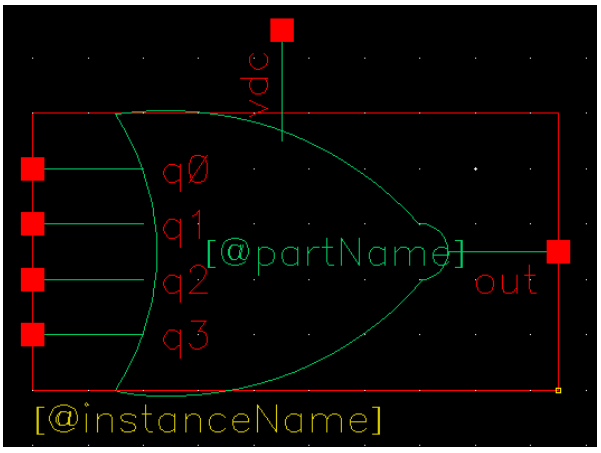



*Figure 27 5-1 OR gate*

## 2x1 MULTIPLEXER && 4 (2x1) Multiplexer

The 2x1 Multiplexer is implemented as follows ( One inverter gate, two AND gates, and one OR gate)

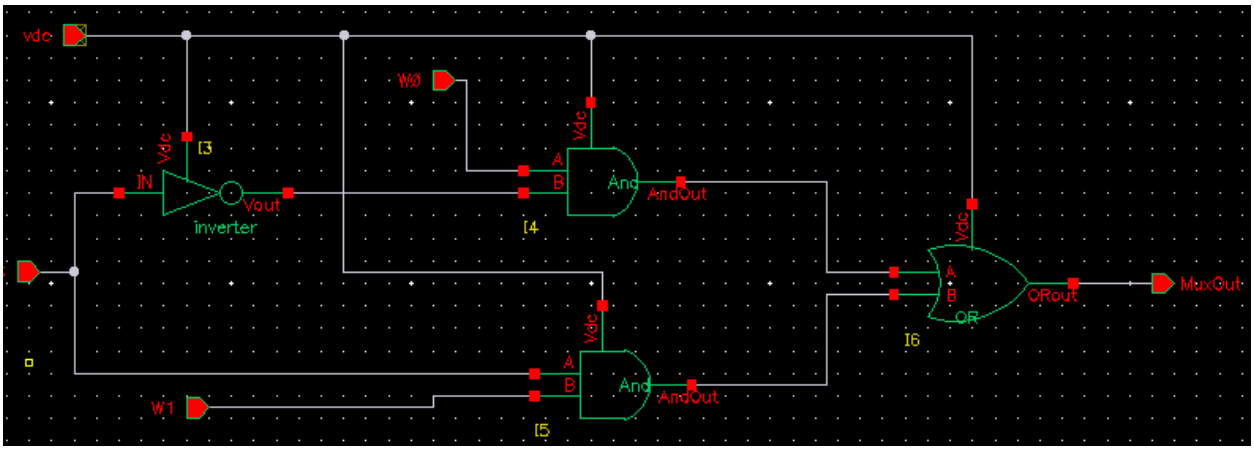



*Figure 28 2x1 Multiplexer Circuit*

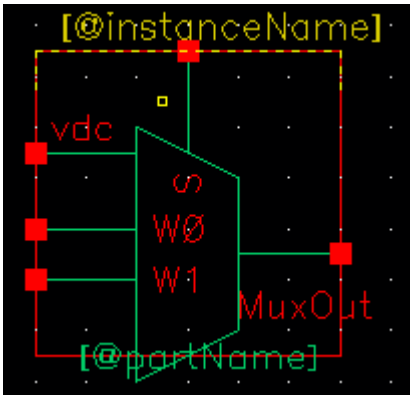



*Figure 29 2x1 Multiplexer symbol*

The 4 (2x1) Multiplexer was implemented using 4 2x1 Multiplexers.

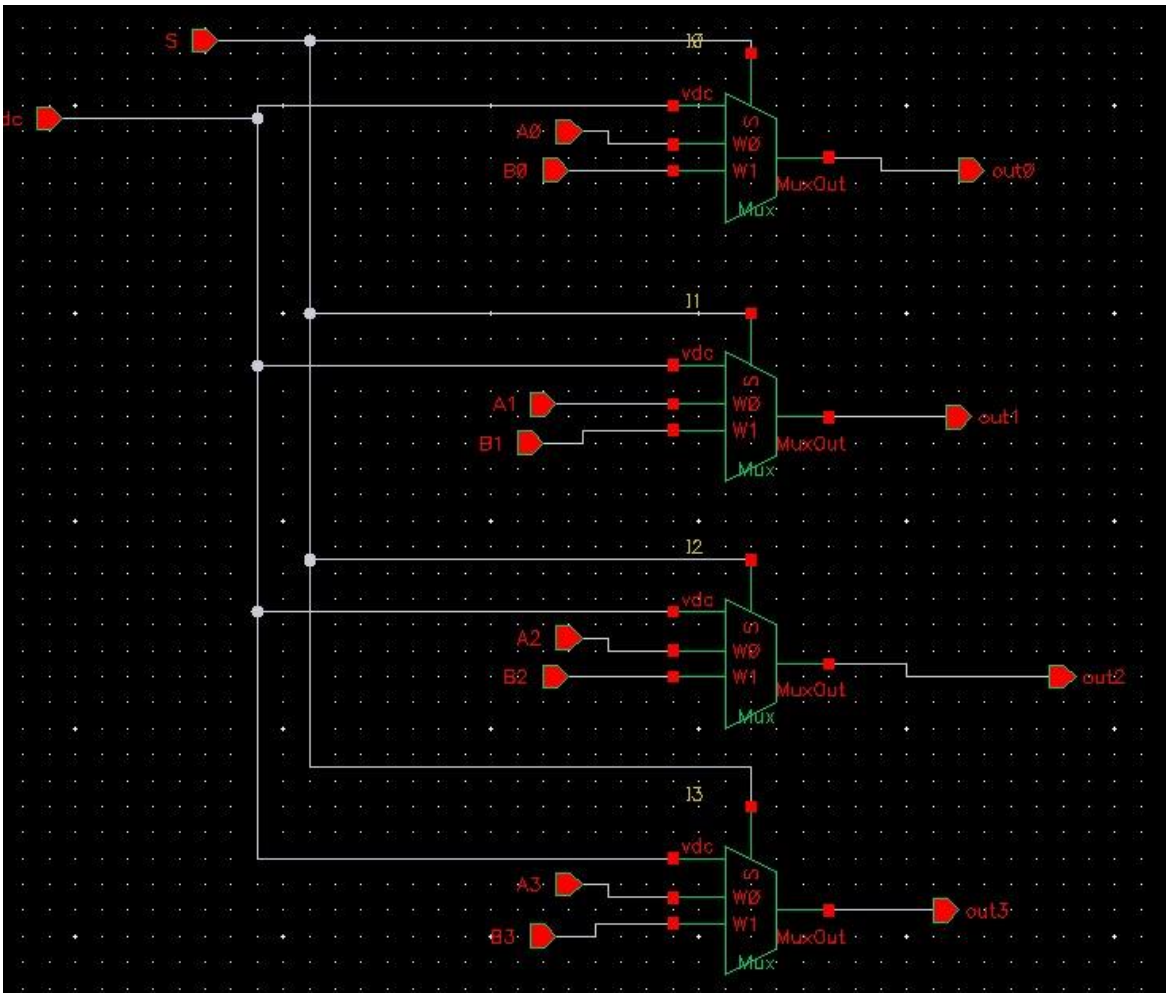



*Figure 30 4 2x1 Multiplexer circuit*

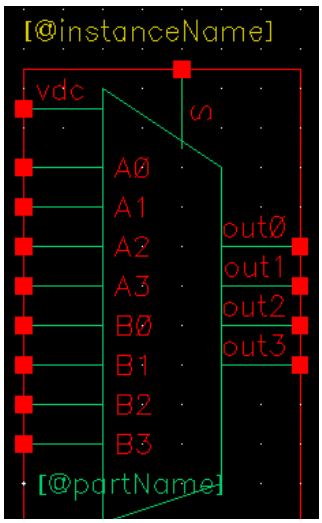



*Figure 31 4 2x1 Multiplexer symbol*

## 4x1 MULTIPLEXER

The 4x1 Multiplexer was implemented using two inverter gates, four 3-1 AND gates, one 4-1 OR gate

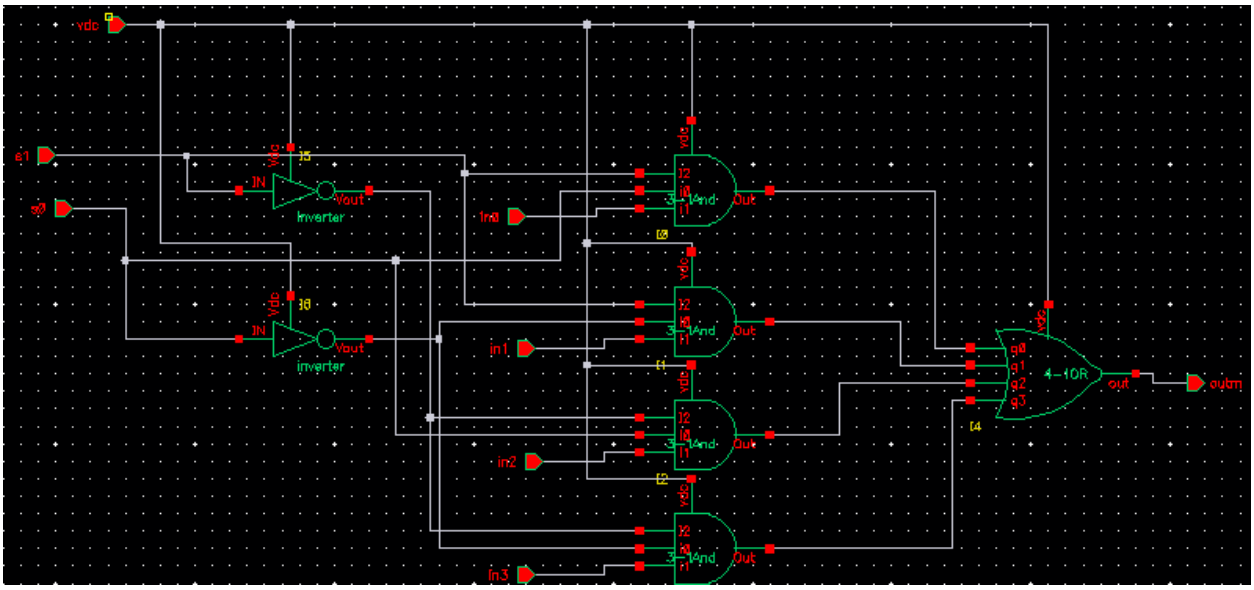



*Figure 33 4x1 Multiplexer Circuit*

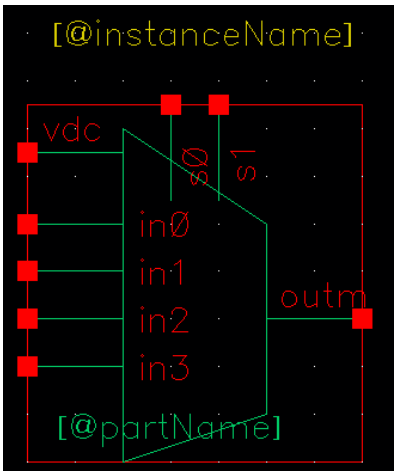



*Figure 32 4x1 Multiplexer symbol*

The 4 (4x1) Multiplexer was implemented using 4 4x1 Multiplexers.

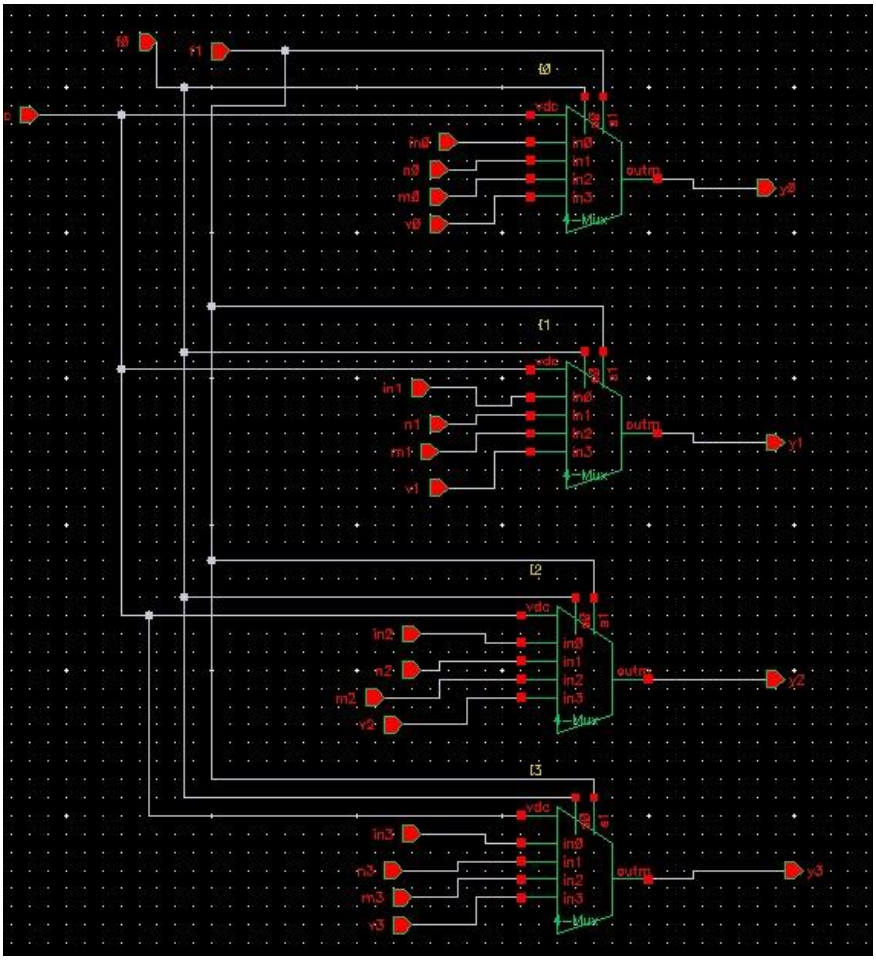


*Figure 35 4 4x1 Multiplexer circuit*

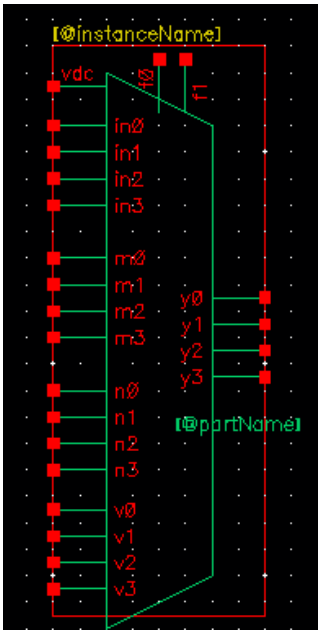



*Figure 34 4 4x1 Multiplexer symbol*

# CARRY LOOK AHEAD ADDER

The carry look ahead ADDER consists of three parts: P & G generator, Carry look ahead, and Adder.

## P & G Generator (Half Adder):

This block consists of half adders used for generating P and G terms of each bit and generate these terms needed for the CLA because the carry depends on it.

$$\text{Carry} = \quad AB + Cin\ (A \text{ XOR } B)$$

Where (A XOR B) is called carry propagate (known as P) because it propagates the Cin from previous stage to the next stage. And AB is called carry generate (known as G) because it can directly generate carry bit without any Cin.

P and G terms for a 4-bit number will be:

$$P_0 = (A_0 \text{ XOR } B_0) \ , G_0 = A_0B_0$$

$$P_1 = (A_1 \text{ XOR } B_1) \ , G_1 = A_1B_1$$

$$P_2 = (A_2 \text{ XOR } B_2) \ , G_2 = A_2B_2$$

$$P_3 = (A_3 \text{ XOR } B_3) \ , G_3 = A_3B_3$$

The P & G generator circuit is shown below:

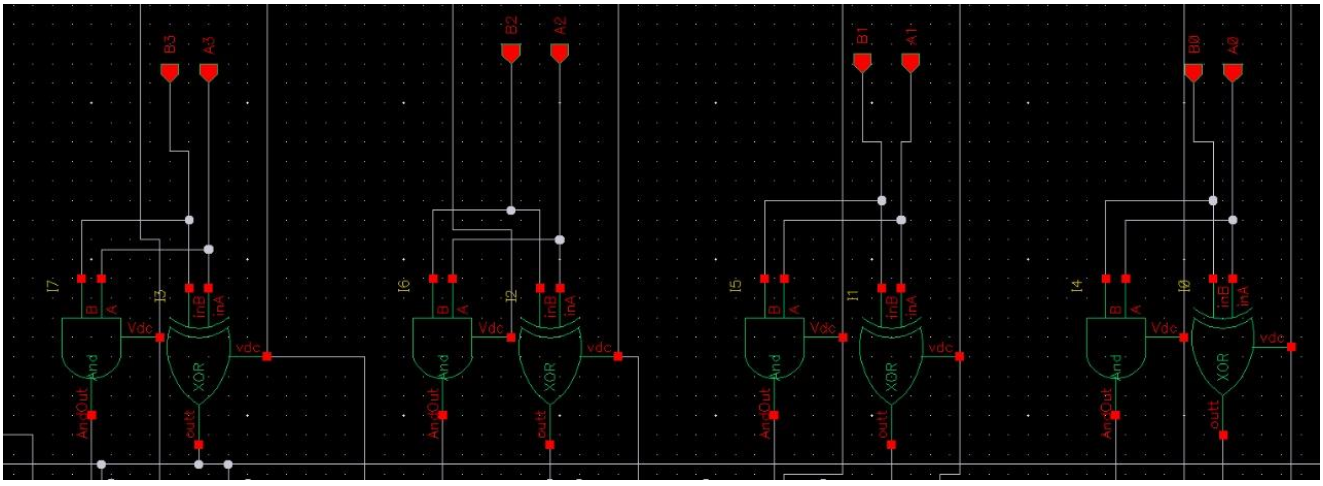



*Figure 36 P & G generator circuit*

## Carry Lookahead (CLA):

CLA block contains combinational circuit used to determine the carry bits. it takes Augend, Addend and carry in as inputs and produce the carry bits. Consider A, B two numbers being added both 4 bits wide and $\mathbf{C_0}$ as carry in. And $C_1$, $C_2$, $C_3$, $C_4$ are the output carry bits.

First we will derive $C_1$,$C_2$,$C_3$,$C_4$ using the carry equation from full adder.

Carry = AB + $C_{in}$ (A XOR B)

P = (A XOR B)

G = AB

Carry = G + $C_{in}$ P

So $C_1$, $C_2$, $C_3$, $C_4$ will become:

$C_1 = G_0 + C_0P_0$

$C_2 = G_1 + C_1P_1 = G_1 + (G_0 + C_0P_0)P_1 = G_1 + G_0P_1 + C_0P_0P_1$

$C_3 = G_2 + C_2P_2 = G_2 + (G_1 + G_0P_1 + C_0P_0P_1)P_2 = G_2 + G_1P_2 + G_0P_1P_2 + C_0P_0P_1P_2$

$C_4 = G_3 + C_3P_3 = G_3 + (G_2 + G_1P_2 + G_0P_1P_2 + C_0P_0P_1P_2)P_3 = G_3 + G_2P_3 + G_1P_2P_3 + G_0P_1P_2P_3 + C_0P_0P_1P_2P_3$

Thus, it shows that none of these carry bits depend on their previous stage carry bits. They only need A , B and $C_0$. This carry look ahead block will generate these carry bits and provide it to the adders block for addition. The circuit of the carry look ahead is shown below:

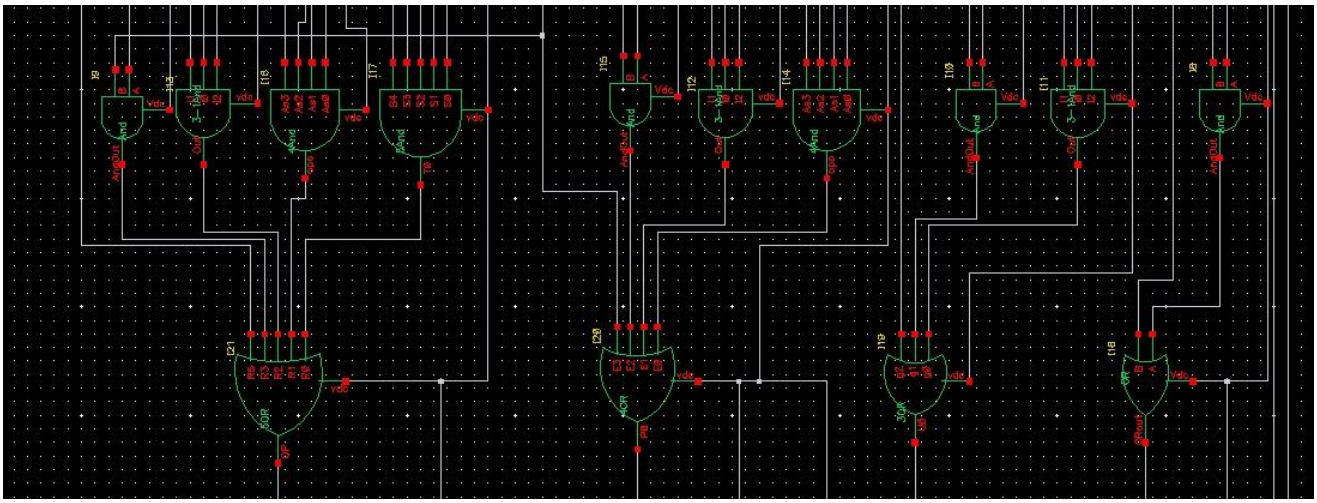


*Figure 37 Carry Look Ahead Circuit*

## Adder:

As we know that the carry bits will be provided by the CLA block, so we don't need Full adders to evaluate the carry bits. We only need the sum which is:

$$S = C_{in} XOR (A \ XOR \ B)$$

So the sum of each bit will be

$S_0 = C_0 \, XOR \, (A_0 \ XOR \ B_0) = C_0 \, XOR \, (P_0)$

$S_1 = C_1 \, XOR \, (A_1 \ XOR \ B_1) = C_1 \, XOR \, (P_1)$

$S_2 = C_2 \, XOR \, (A_2 \ XOR \ B_2) = C_2 \, XOR \, (P_2)$

$S_3 = C_3 \, XOR \, (A_3 \ XOR \ B_3) = C_3 \, XOR \, (P_3)$

And the last carry $C_4$ will be the $C_{out}$ of the **CLA full adder**.

$C_{out} = C_4$

The circuit of the Adder is shown below:

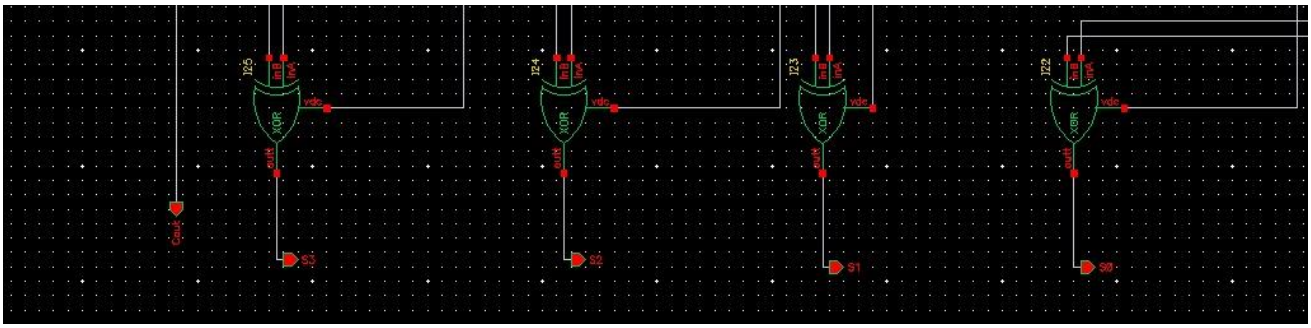



*Figure 38 Adder circuit*

The full circuit of the CLA adder is shown below:

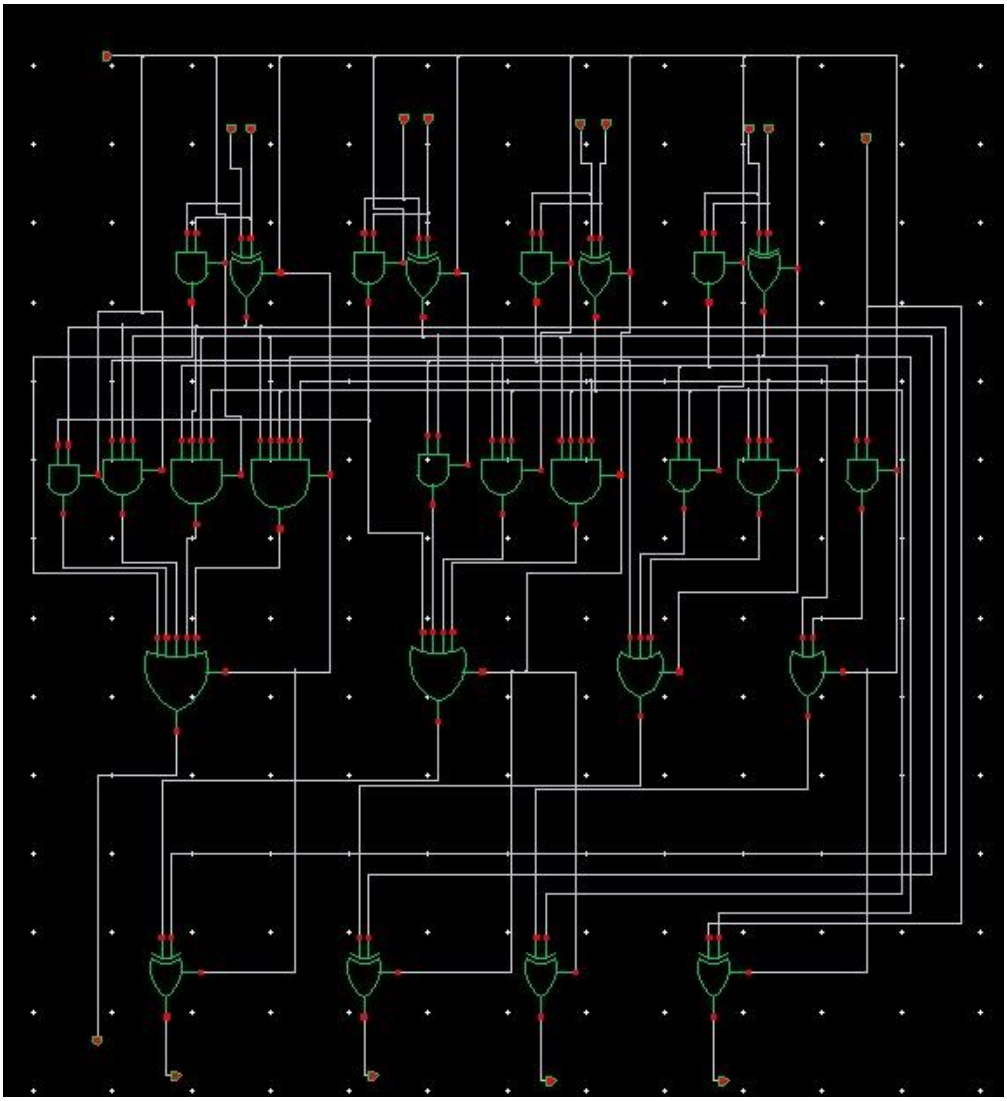


*Figure 39 Full circuit of CLA ADDER*

## RIPPLE ADDER

A Full adder was implemented by the follwing equations

$$S_i = A_i \oplus B_i \oplus C_i$$

$$C_i = C_{i-1}(A_i \oplus B_i) + A_i B_i$$

The circuit of the full adder is shown below:

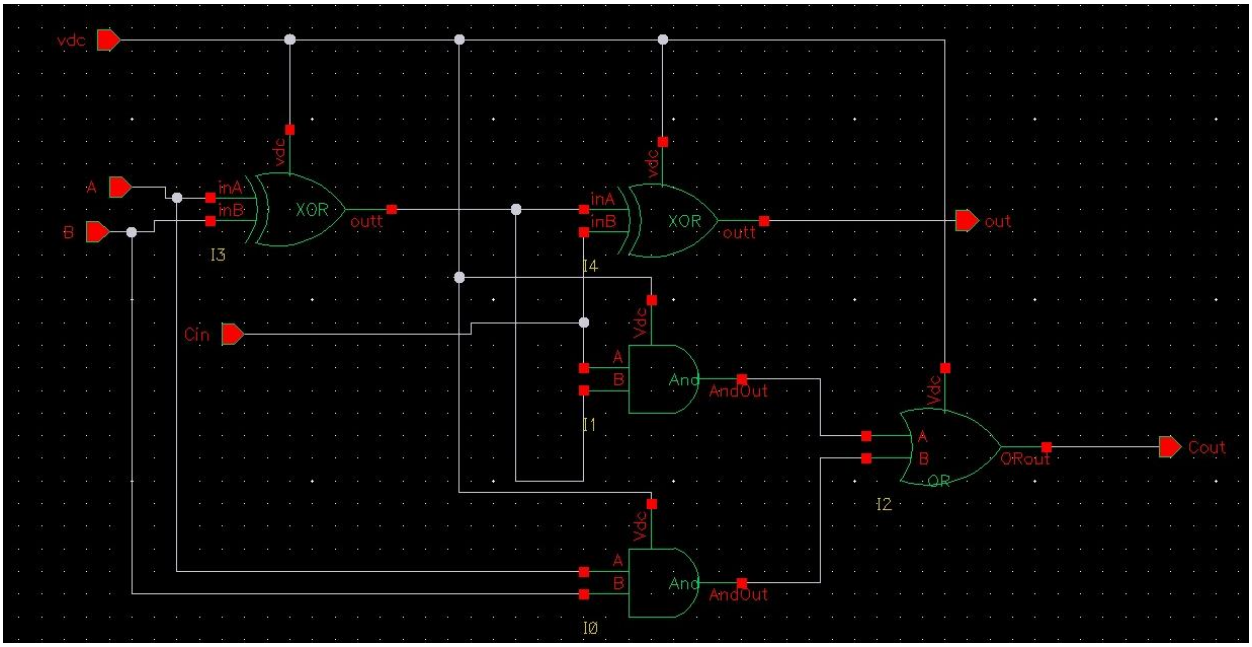



*Figure 40 Full adder circuit*

Then the Ripple adder was implemented by four full adders as shown below:

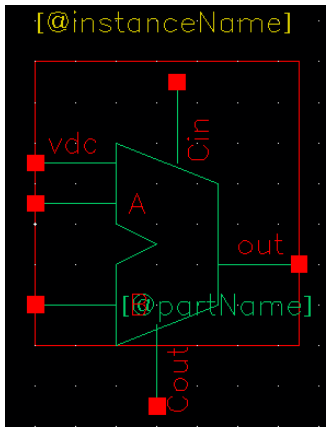



*Figure 41 Full adder symbol*

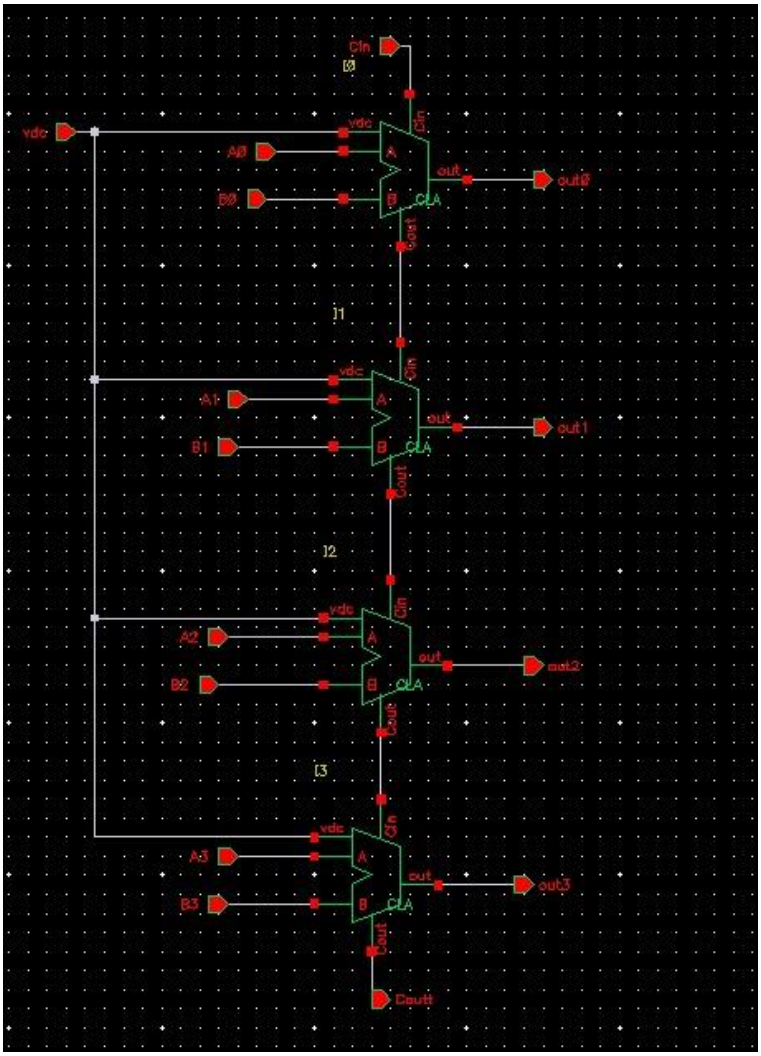



*Figure 42 4 Full adders connected ( Ripple adder)*

# Assembly

## ALU by carry lookahead adder

The assembly of the components of the ALU with CLA is shown below.

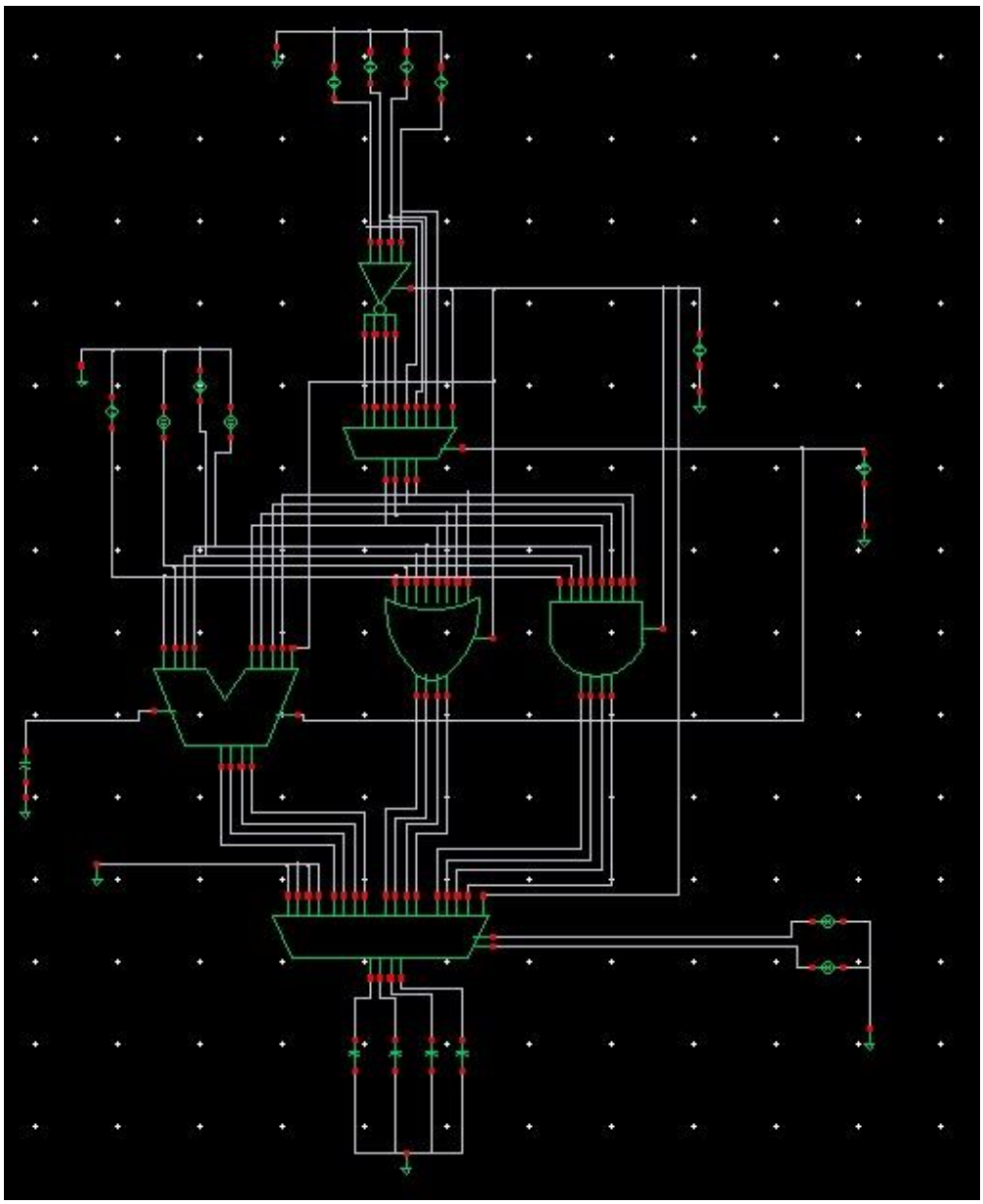


*Figure 43 Full ALU circuit with CLA*

## ALU by ripple adder

The assembly of the components of the ALU with ripple adder is shown below.

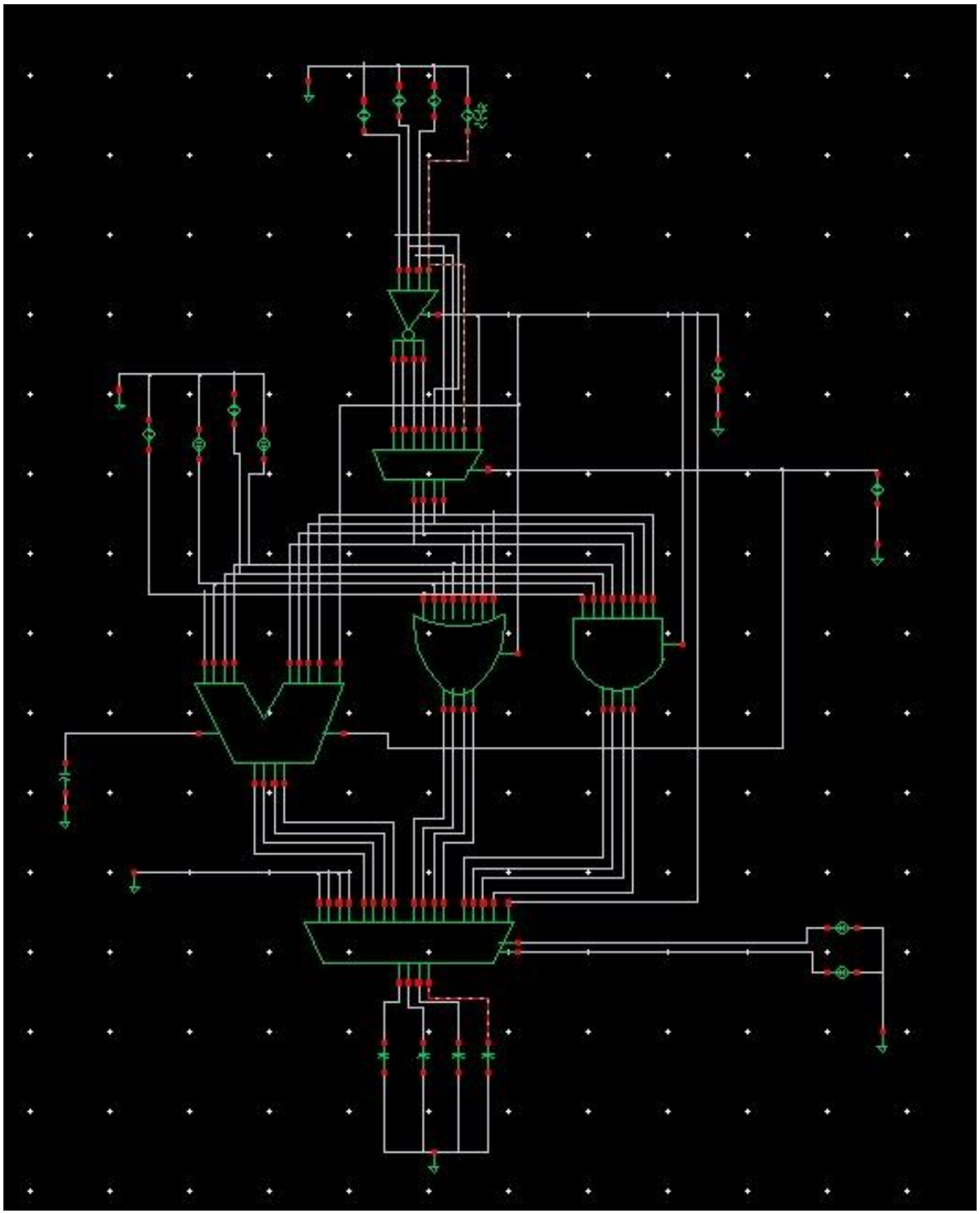


*Figure 44 Full ALU circuit with ripple adder*

# Testing

To test the performance of the ALU circuit we put A = 0111(7), B = 0101(5), the results of the functions is shown in the following table:

| F2:0 | Function | Result |
|---|---|---|
| 000 | & | 0101 |
| 001 | \| | 0111 |
| 010 | + | 1100 |
| 011 | ~used | 0000 |
| 100 | A & ~B | 0010 |
| 101 | A \| ~B | 1111 |
| 110 | A – B | 0010 |
| 111 | SLT | 0001 |

The following graph shows the results:

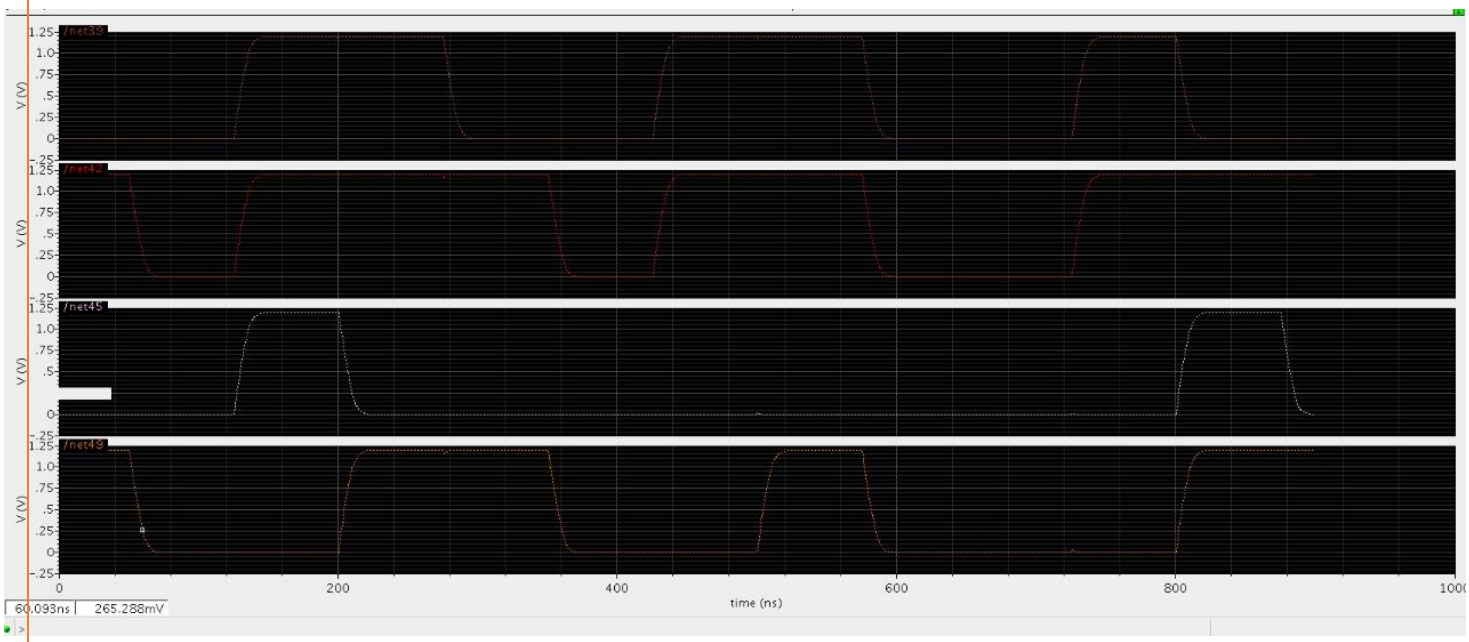



*Figure 45 Results of the functions graph*

## And Testing

| F2:0 | Function | Result |
|---|---|---|
| 000 | & | 0101 |

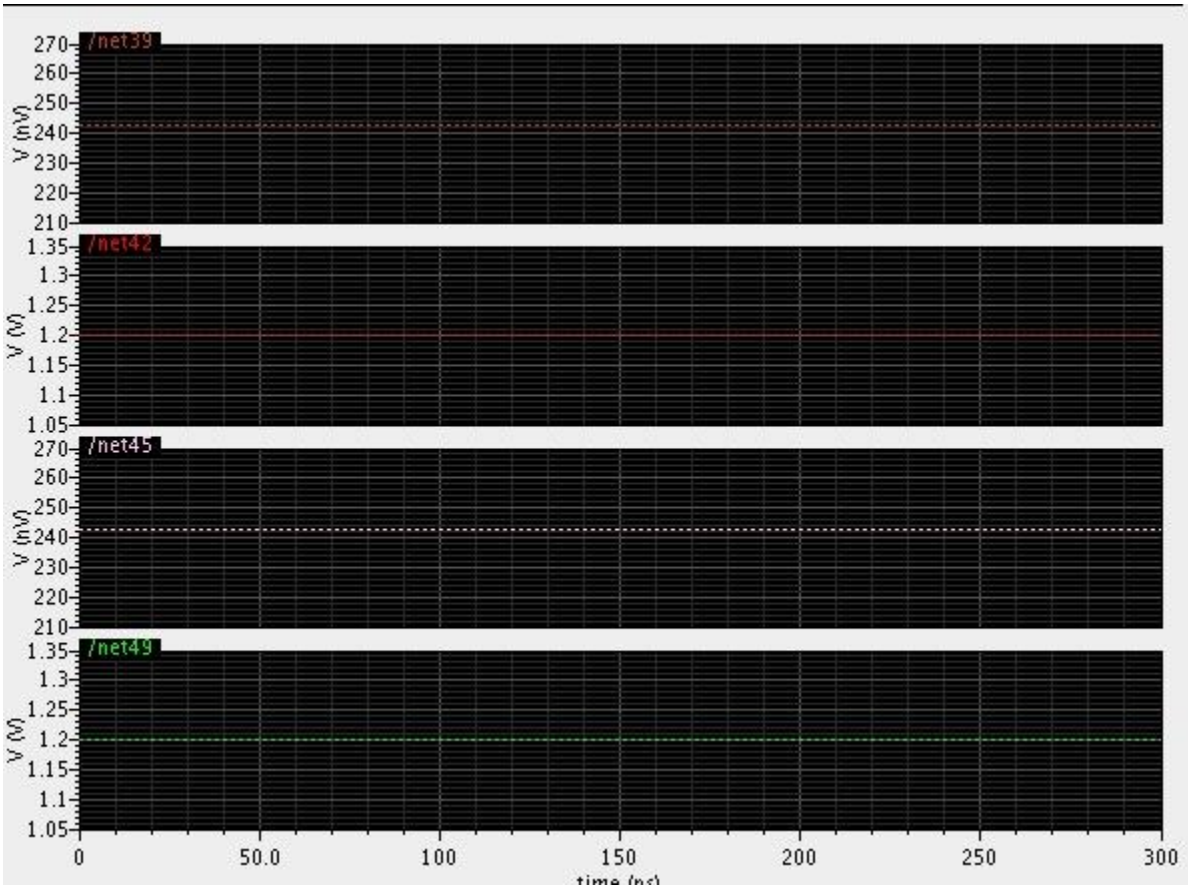



*Figure 46 A & B*

## A | B Testing

| F2:0 | Function | Result |
|---|---|---|
| 001 | \| | 0111 |

The result is shown in the graph below:

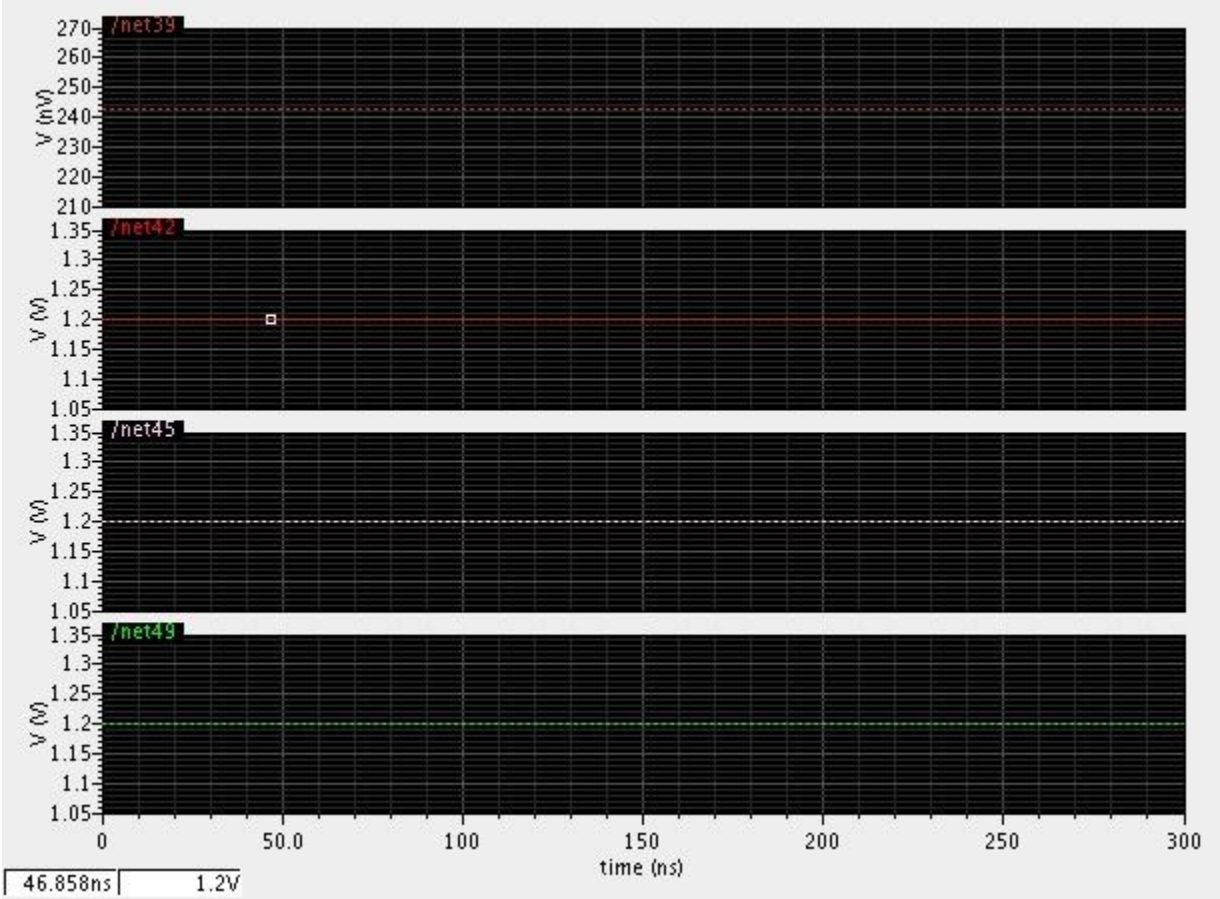



*Figure 47 A | B*

## A + B Testing

| F2:0 | Function | Result |
|---|---|---|
| 010 | + | 1100 |

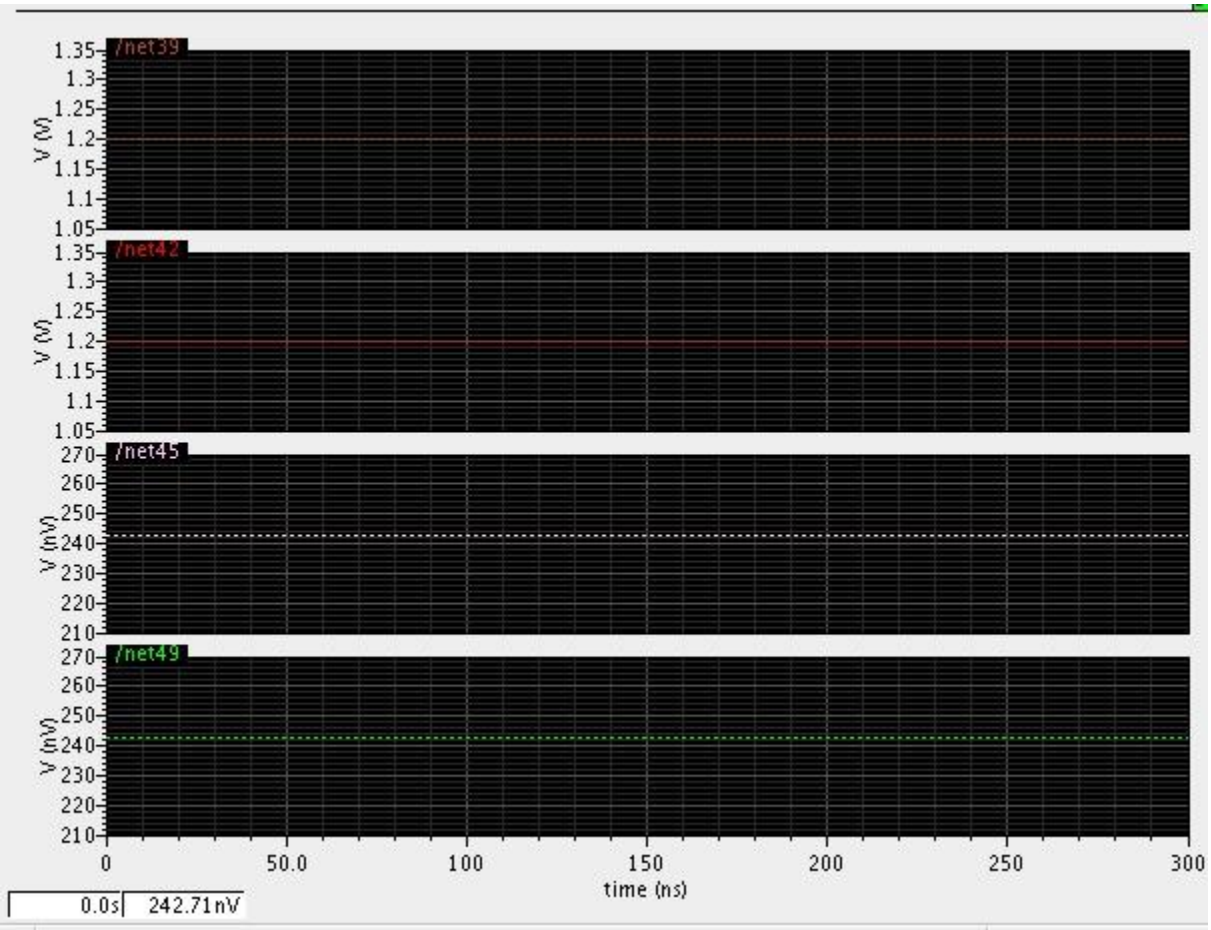



*Figure 48 A + B*

## Not Used

| F2:0 | Function | Result |
|---|---|---|
| 011 | ~used | 0000 |

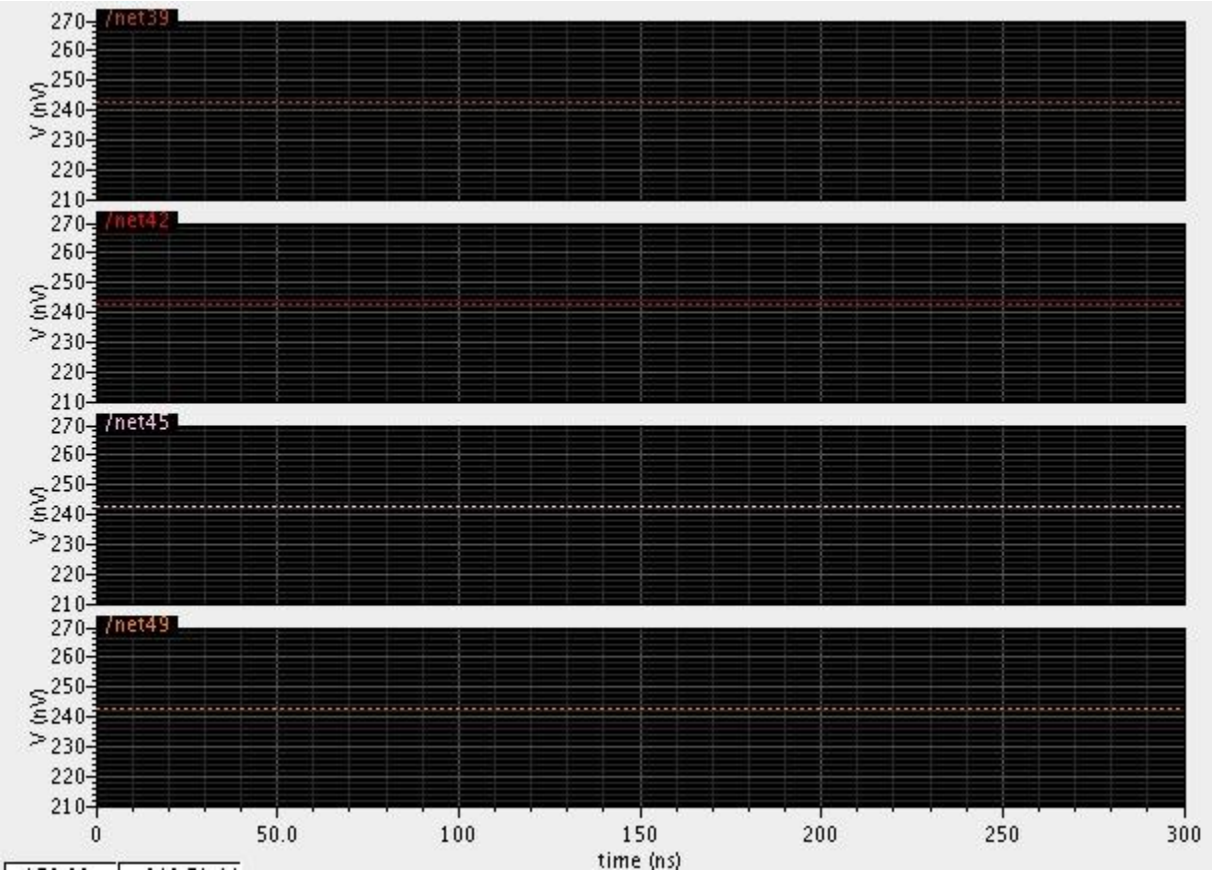



*Figure 49 Not Used*

## A & ~B

| F2:0 | Function | Result |
|---|---|---|
| 100 | A & ~B | 0010 |

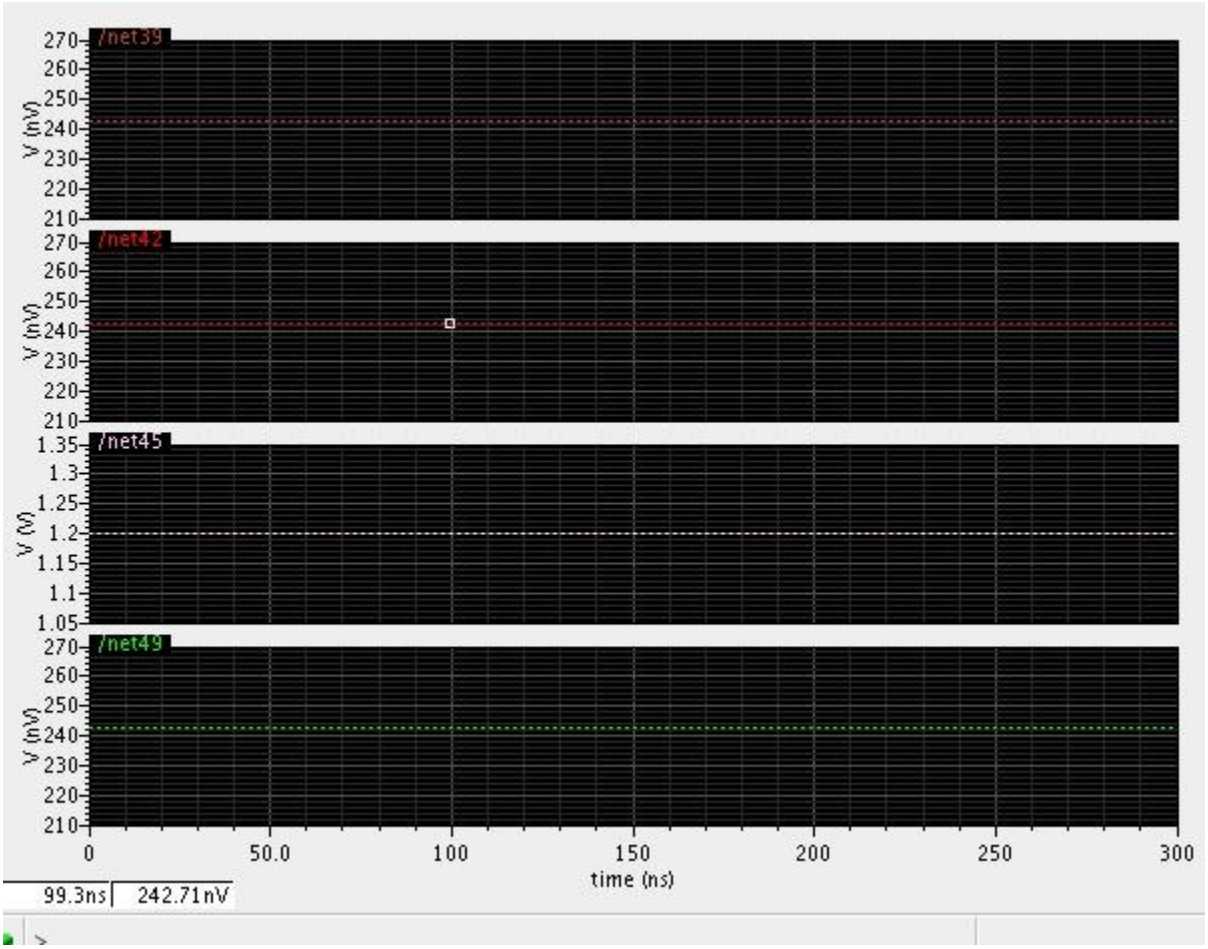



*Figure 50 A & ~B*

## A | ~B

| F2:0 | Function | Result |
|---|---|---|
| 100 | A \| ~B | 1111 |

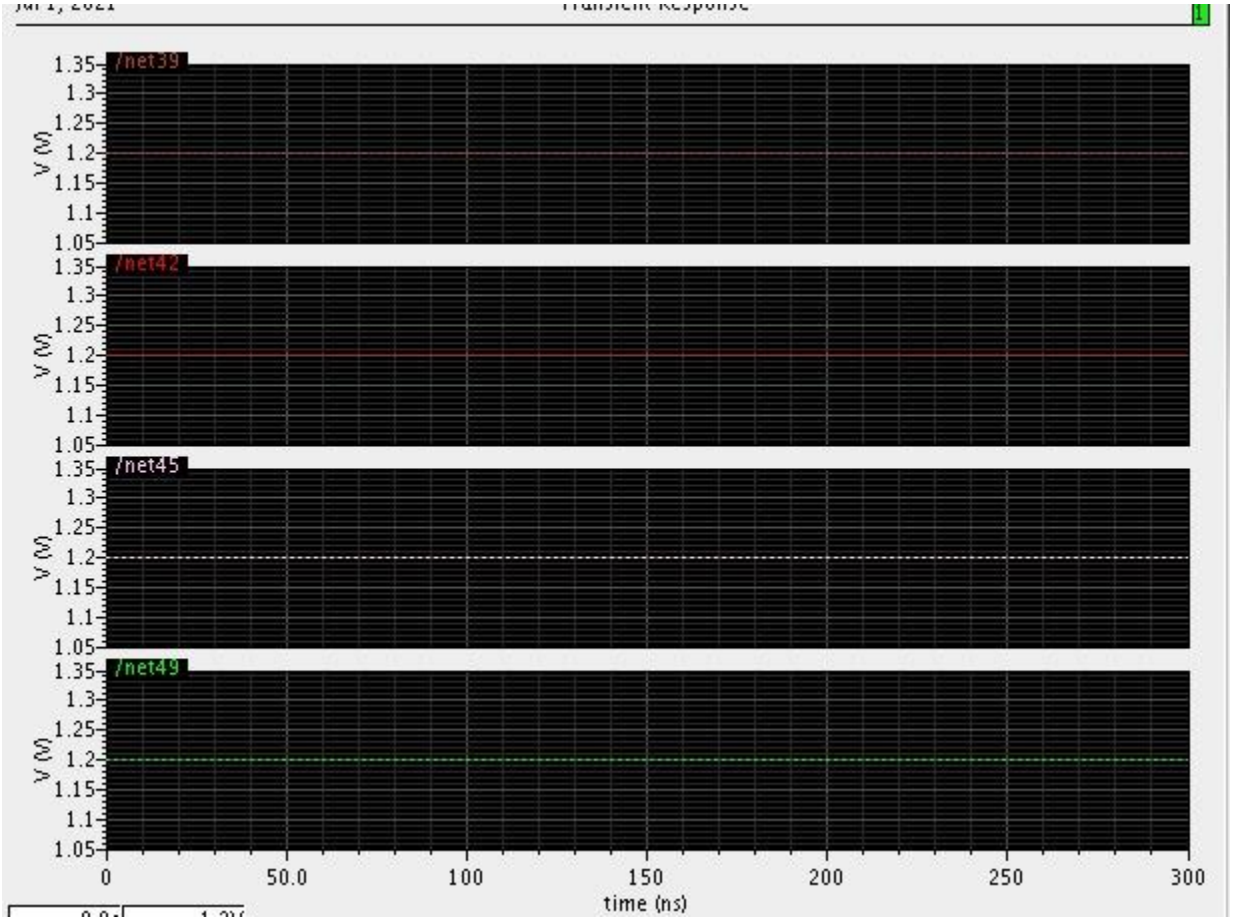



*Figure 51 A | ~B*

## A - B

| F2:0 | Function | Result |
|---|---|---|
| 110 | A – B | 0010 |

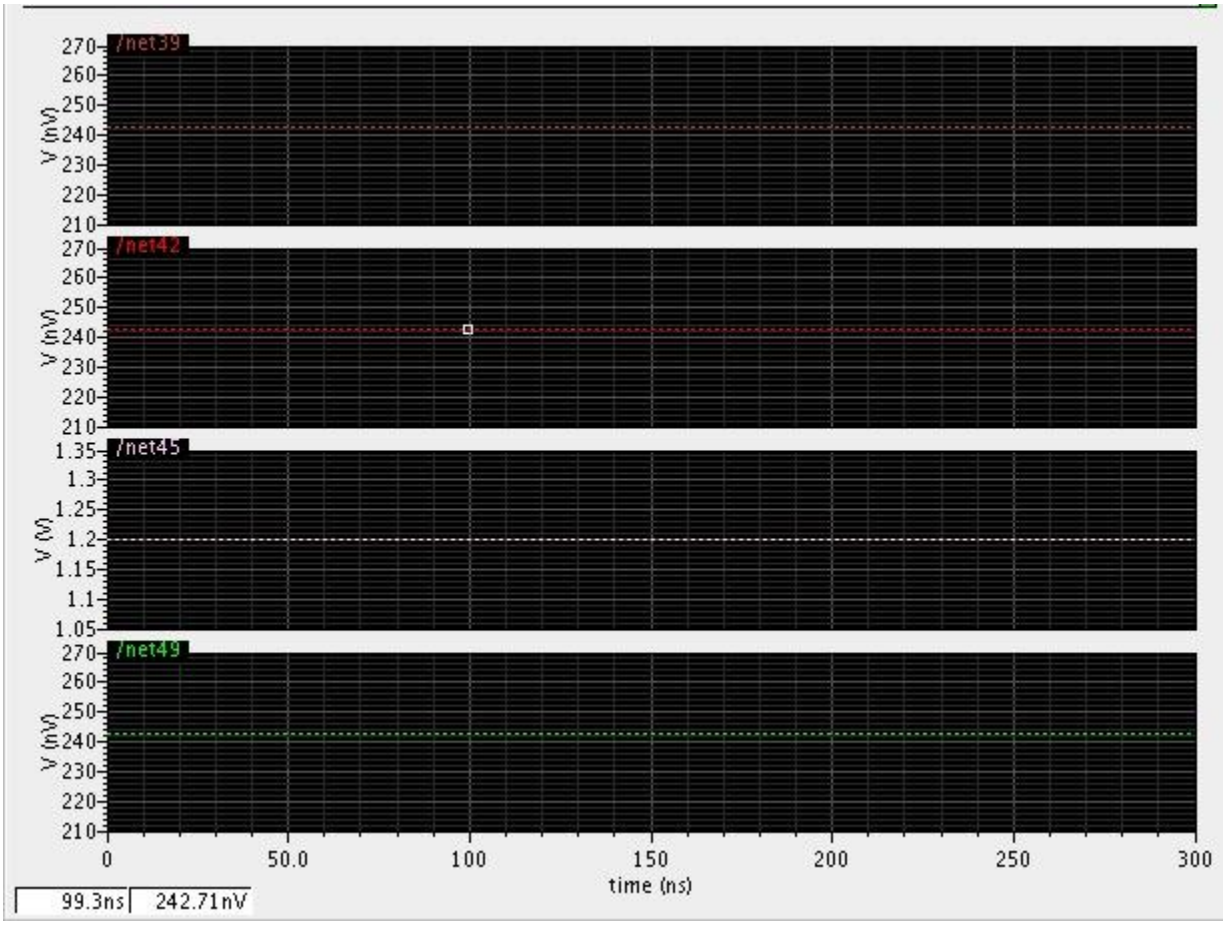



*Figure 52 A - B*

## SLT

| F2:0 | Function | Result |
|---|---|---|
| 111 | SLT | 0000 |

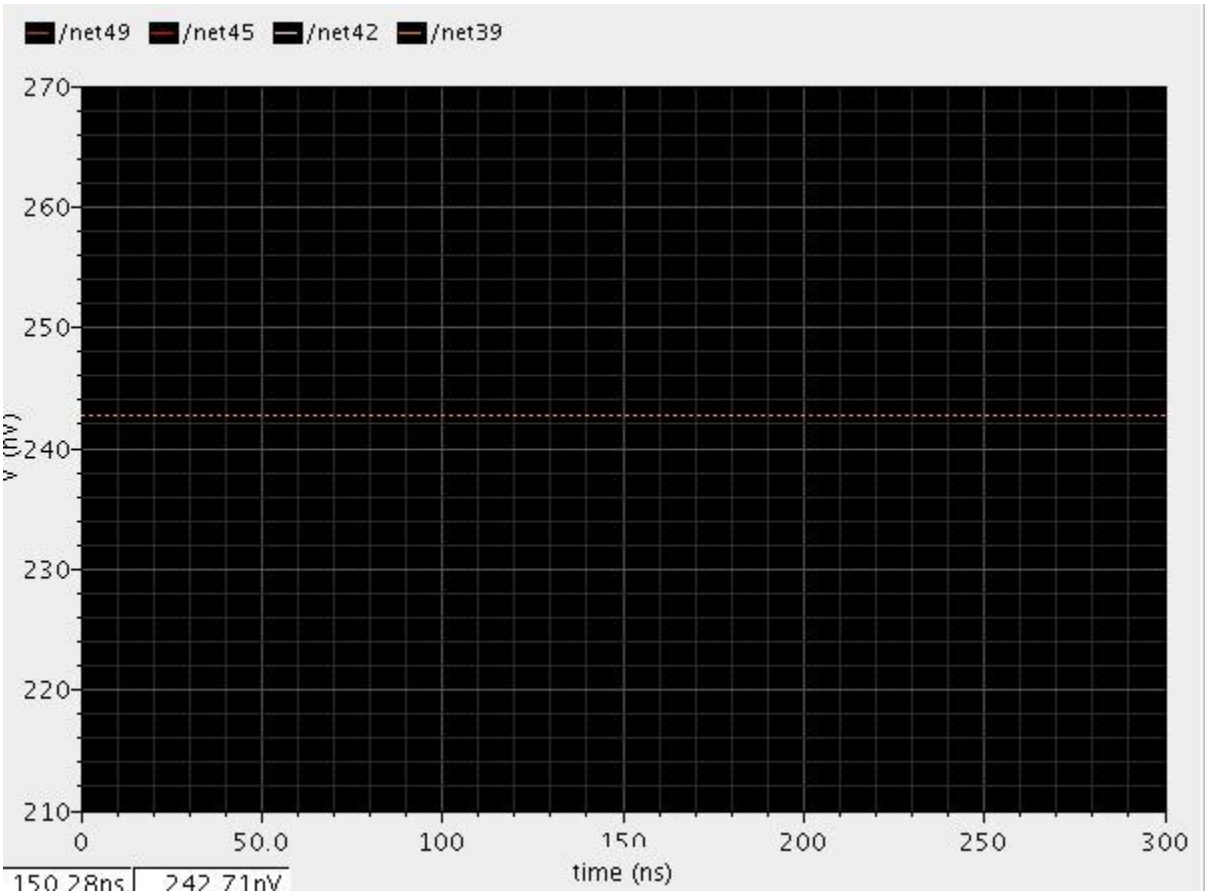



*Figure 53 SLT test*

# Results

## Propagation delay

The propagation delay was measured on the worst-case scenario; the worst-case here is the longest path of signals that is the addition part. Thus, the propagation delay of both ALUs' addition was measured.

### ALU with carry lookahead adder:

TPLH is 5.82ns and TPHL is 6.3ns as shown below.

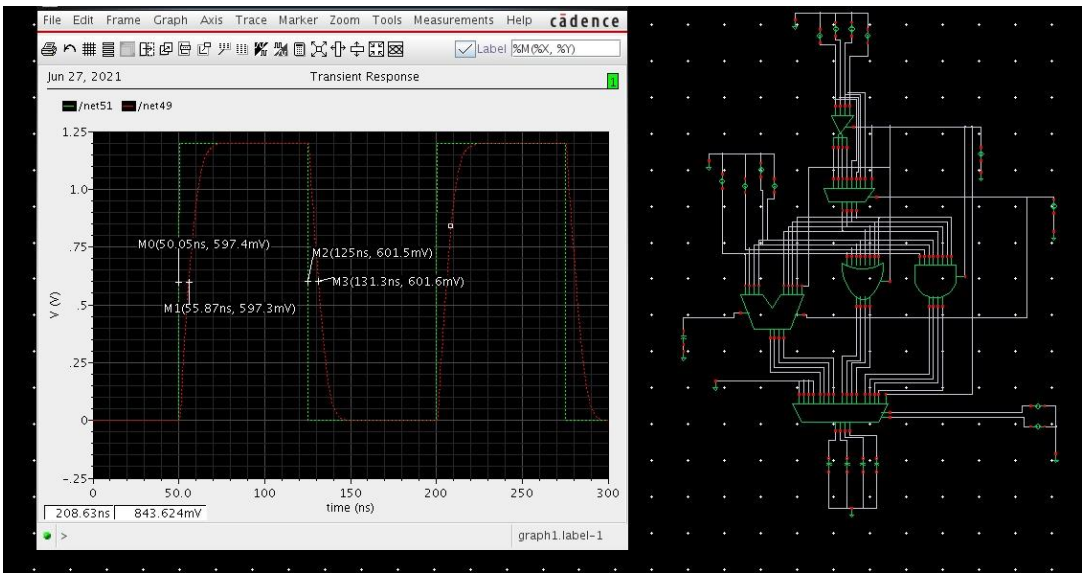



*Figure 54 CLA Propagation delay*

### ALU with ripple adder

TPLH is 5.83ns and TPHL is 6ns as shown below.

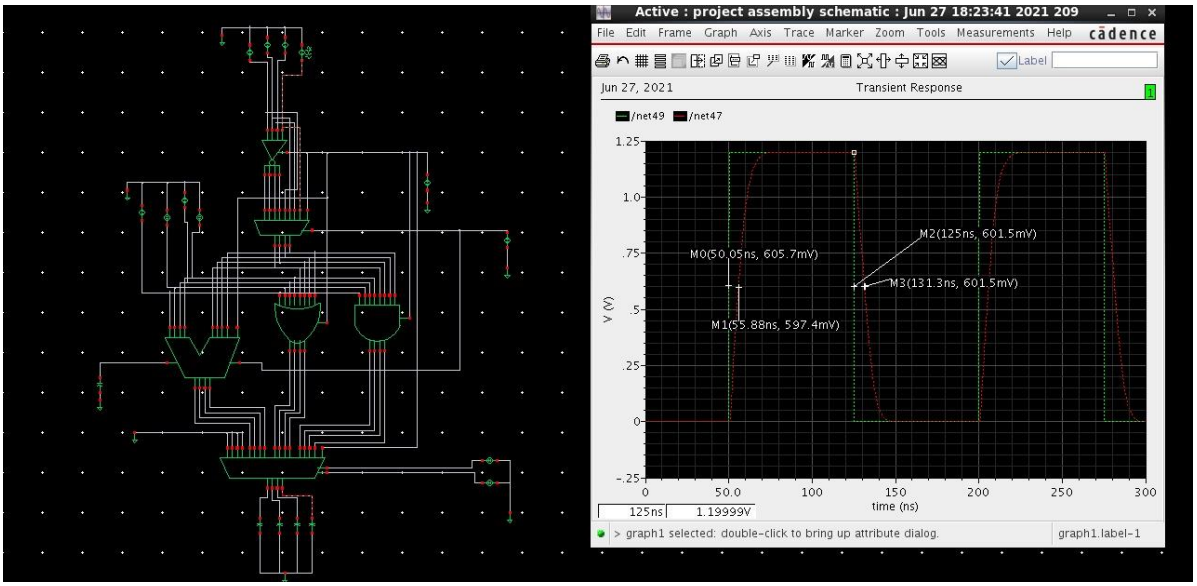



*Figure 55 Ripple Adder propagation delay*

## Maximum power consumption

### ALU by carry lookahead adder

The power consumption was measured by the simulation

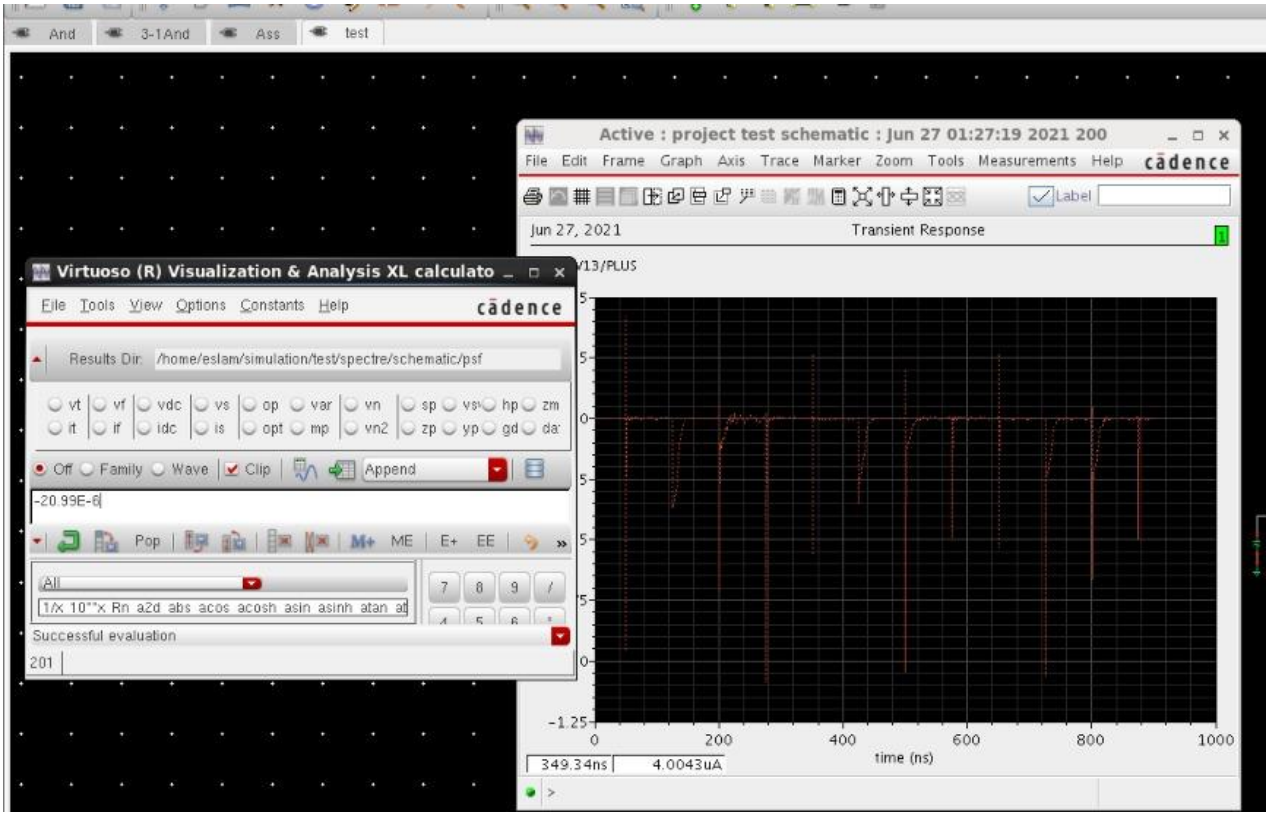



*Figure 56 ALU with CLA Power Consumption*

## ALU by ripple adder

The power consumption was measured by the simulation

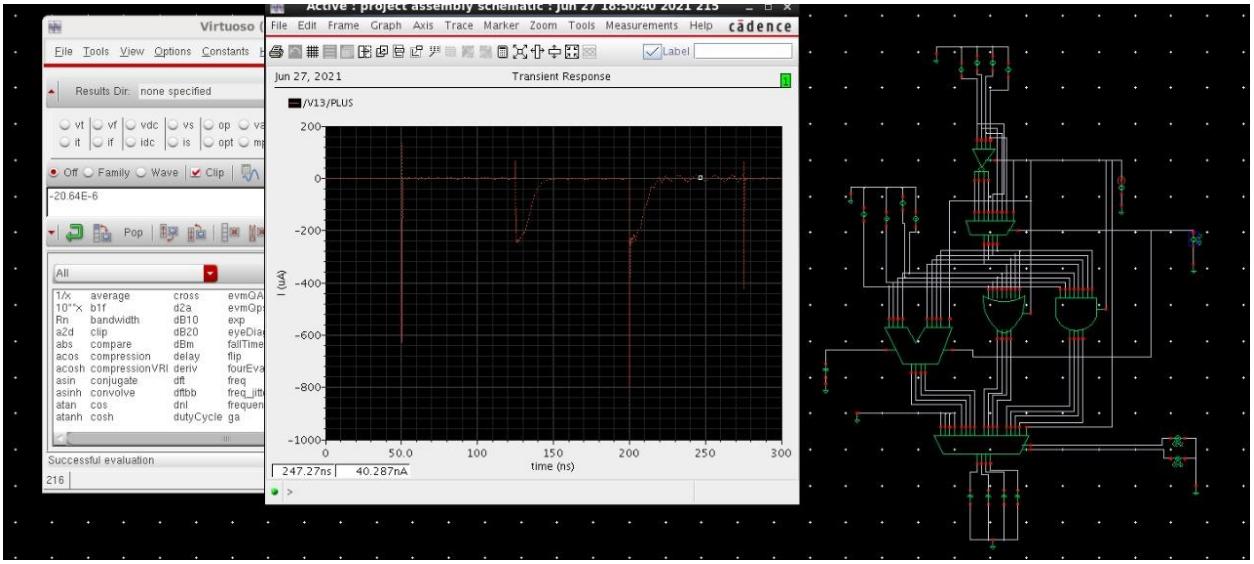



*Figure 57 ALU with ripple adder power consumption*

# Comparison and conclusion

| Aspect of comparison | Carry lookahead ALU | Ripple carry ALU |
|---|---|---|
| Number of used transistors | 636 | 568 |
| Power dissipation | $20.99\ \mu W$ | $20.64\mu W$ |
| Propagation delay | TPLH = 5.82ns<br>TPHL = 6.3ns | TPLH = 5.83ns<br>TPHL = 6ns |

The number of used transistors could have been larger than the current number (772 in case of CLA and 704 in case of ripple adder) if we used the multiple input one output combinational logic gates with designed logic gates instead of designing them as MOSFETs. The performance of the two adders is nearly the same because we are working on a scale of 4 bits, if the number increases the output will be different. Finally, the number of used transistors in CLA is more than that used with ripple adder, but it is not a big difference.