PROJECT REPORT ON

IMPLEMENTATION OF 4-Bit ALU

A report submitted for the partial fulfilment of the requirements of the course

ECL-312 : CMOS DESIGN

Submission date : 11TH NOVERMBER 2024

Submitted by : Divy Dwivedi(BT22ECI023)

Under the guidance of

DR. PARITOSH PESHWE

Department of Electronics and Communication Engineering

भारतीय सूचना प्रौद्योगिकी संस्थान, नागपुर

Indian Institute of Information Technology, Nagpur

INTRODUCTION

An Arithmetic Logic Unit (ALU) is a fundamental component of digital computing, responsible for performing arithmetic and logical operations on binary numbers. In this project, a 4-bit ALU is designed using CMOS technology, encompassing essential functions like addition, subtraction, NAND, XOR, and a multiplexer for selecting the desired operation.

- **OBJECTIVE :**

  The objective is to design a 4-bit ALU using CMOS logic to achieve efficient computation with minimal power consumption and area. This project aims to simulate each logical unit to ensure accurate and reliable operation under various conditions.

- **Circuit Components**

  The design consists of several sub-circuits:

1. **Inverter**
2. **XOR Gate**
3. **AND Gate**
4. **OR Gate**
5. **Full Adder**
6. **Full Subtractor**
7. **NAND Gate**
8. **Multiplexer**

Each component is designed to minimize the transistor count, reducing power consumption and area.

Working Principle

1. Inverter

Operation: An inverter (NOT gate) takes a single binary input and flips its value, outputting 1 when the input is 0 and vice versa. This operation is implemented using CMOS technology with one nMOS and one pMOS transistor in complementary configuration. When the input is high, the nMOS conducts, pulling the output low; when the input is low, the pMOS conducts, pushing the output high.

Role in ALU: Inverters are used within the ALU to produce the complement of signals. They are essential in generating negated inputs for operations like subtraction, where a two's complement approach may be used, and in control logic for certain operations.

```
.subckt inv 1 2 3
m1 3 1 0 0 nmod w=100u l=10u
m2 3 1 2 2 pmod w=100u l=10u
.ends
```

2. XOR Gate

Operation: An XOR gate outputs 1 if its two inputs are different and 0 if they are the same. The XOR function is implemented using CMOS with a combination of nMOS and pMOS transistors arranged to produce the desired output based on the logical states of the inputs.

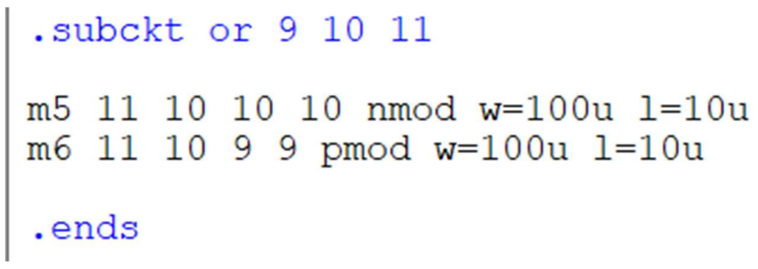
Role in ALU: XOR gates are primarily used in addition and subtraction operations. In full adders, they calculate the sum bit by comparing two input bits and, if applicable, the carry bit. XOR gates are also used for logical XOR operations between two binary numbers within the ALU.

```
.subckt xor 4 5 6 2
xv4_inv 4 2 7 inv
xv5_inv 5 2 8 inv

m3 6 4 8 8 nmod w=100u l=10u
m4 6 7 5 5 nmod  w=100u l=10u

.ends
```

3. AND Gate

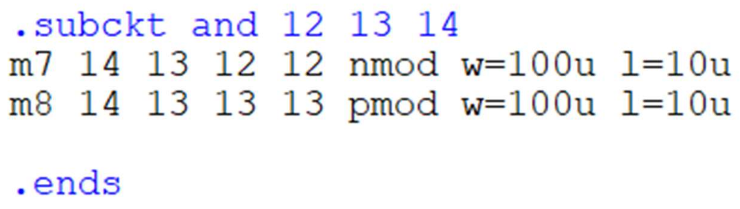
Operation: The AND gate outputs a high signal (1) only if both inputs are high. CMOS AND gates use pairs of nMOS and pMOS transistors in a series configuration, where both transistors must be conductive to produce a high output.

Role in ALU: The AND gate is essential in performing logical AND operations on two binary inputs. It's used within the ALU for bitwise ANDing of two 4-bit numbers, as well as internally in complex circuits like full adders and full subtractors.



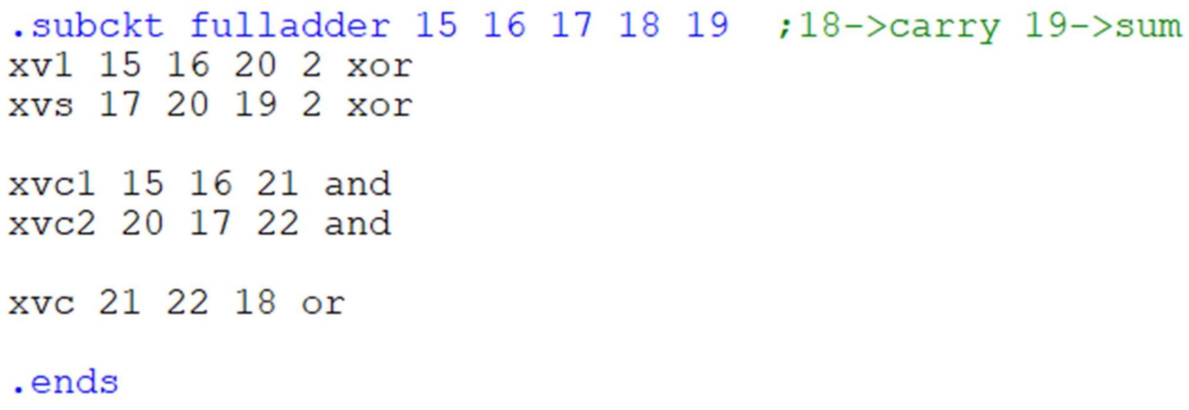
```
.subckt or 9 10 11

m5 11 10 10 10 nmod w=100u l=10u
m6 11 10 9 9 pmod w=100u l=10u

.ends
```

4. OR Gate
Operation: An OR gate outputs a high signal (1) if either or both of its inputs are high. In CMOS, the OR gate can be constructed with nMOS and pMOS transistors in a parallel configuration, where any high input activates the output.
Role in ALU: OR gates in the ALU are used for bitwise OR operations. They can also be used in specific control circuits within the ALU to pass data through or enable specific pathways based on control signals.



```
.subckt and 12 13 14
m7 14 13 12 12 nmod w=100u l=10u
m8 14 13 13 13 pmod w=100u l=10u

.ends
```

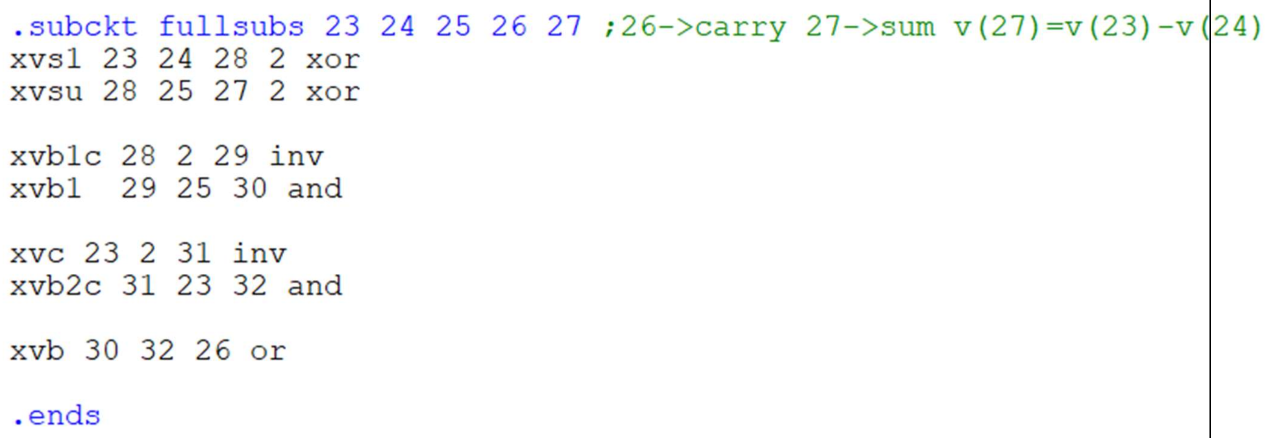
5. Full Adder
Operation: A full adder takes three inputs—two operands and a carry-in—and produces two outputs: a sum and a carry-out. The circuit involves two XOR gates for sum calculation, along with additional AND and OR gates to determine the carry-out based on the input combinations.
Role in ALU: Full adders are the core of the addition function in the ALU. They enable multi-bit addition by connecting the carry-out of one adder to the carry-in of the next bit, allowing the ALU to add 4-bit numbers sequentially. This chain of full adders produces a 4-bit sum and a final carry-out bit.



```
.subckt fulladder 15 16 17 18 19  ;18->carry 19->sum
xv1 15 16 20 2 xor
xvs 17 20 19 2 xor

xvc1 15 16 21 and
xvc2 20 17 22 and

xvc 21 22 18 or

.ends
```

6. Full Subtractor
Operation: A full subtractor is similar to a full adder but instead calculates the difference and borrow-out. It takes three inputs—two operands and a borrow-in—and produces a difference output and a borrow-out. XOR gates are used to determine the difference, and additional AND/OR gates calculate the borrow-out.

Role in ALU: The full subtractor is fundamental to subtraction operations in the ALU. Like the adder, full subtractors are chained together to compute the difference between two 4-bit numbers. This setup generates a 4-bit result with a borrow-out, effectively implementing binary subtraction.

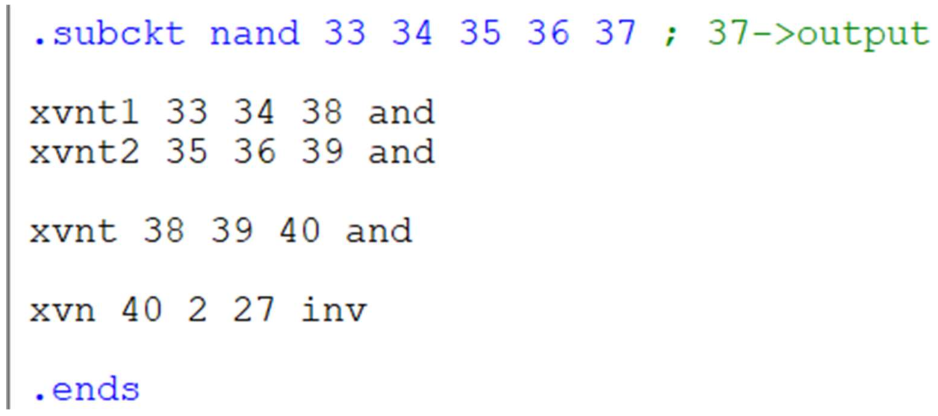


```
.subckt fullsubs 23 24 25 26 27 ;26->carry 27->sum v(27)=v(23)-v(24)
xvs1 23 24 28 2 xor
xvsu 28 25 27 2 xor

xvb1c 28 2 29 inv
xvb1  29 25 30 and

xvc 23 2 31 inv
xvb2c 31 23 32 and

xvb 30 32 26 or

.ends
```

7. NAND Gate

Operation: A NAND gate produces a low output only if all its inputs are high; otherwise, the output is high. CMOS NAND gates use both nMOS and pMOS transistors, arranged to create this behavior by turning on the output only when both inputs are low.

Role in ALU: The NAND gate performs a logical NAND operation on two binary numbers. It’s included as a fundamental logical operation within the ALU, expanding the range of basic operations beyond AND and OR. NAND is a universal gate, meaning it can be combined to create any other logical function if necessary.



```
.subckt nand 33 34 35 36 37 ; 37->output

xvnt1 33 34 38 and
xvnt2 35 36 39 and

xvnt 38 39 40 and

xvn 40 2 27 inv

.ends
```

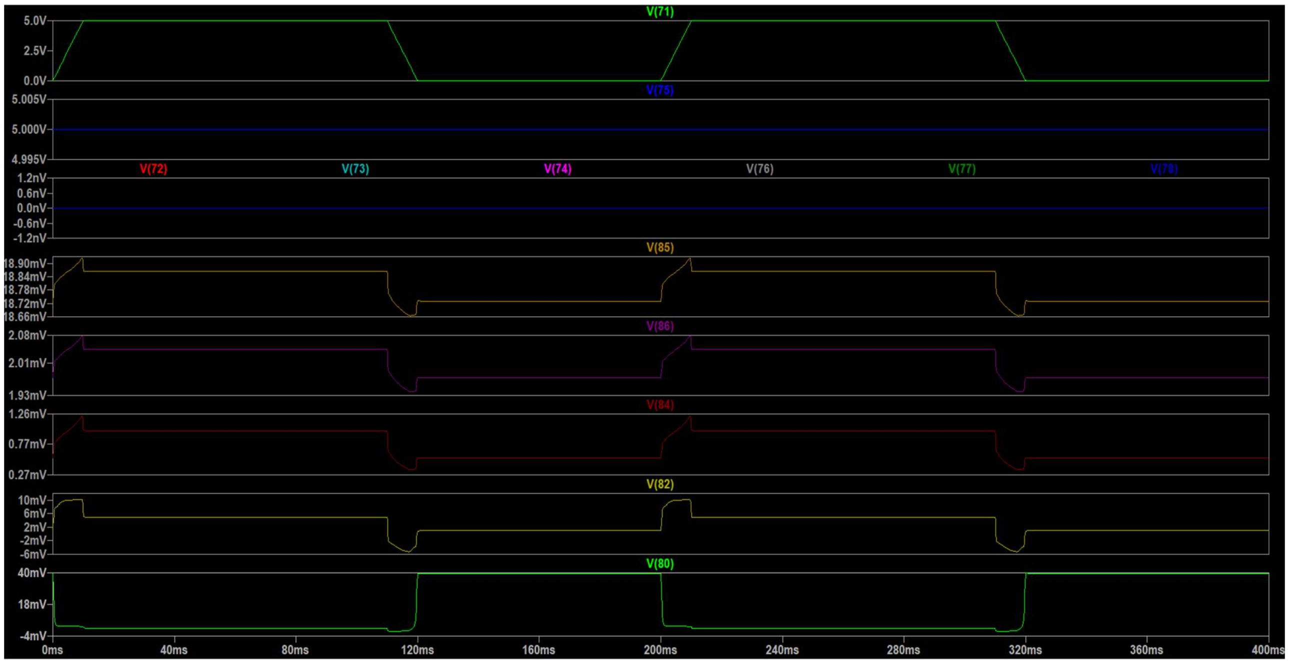
8. Multiplexer

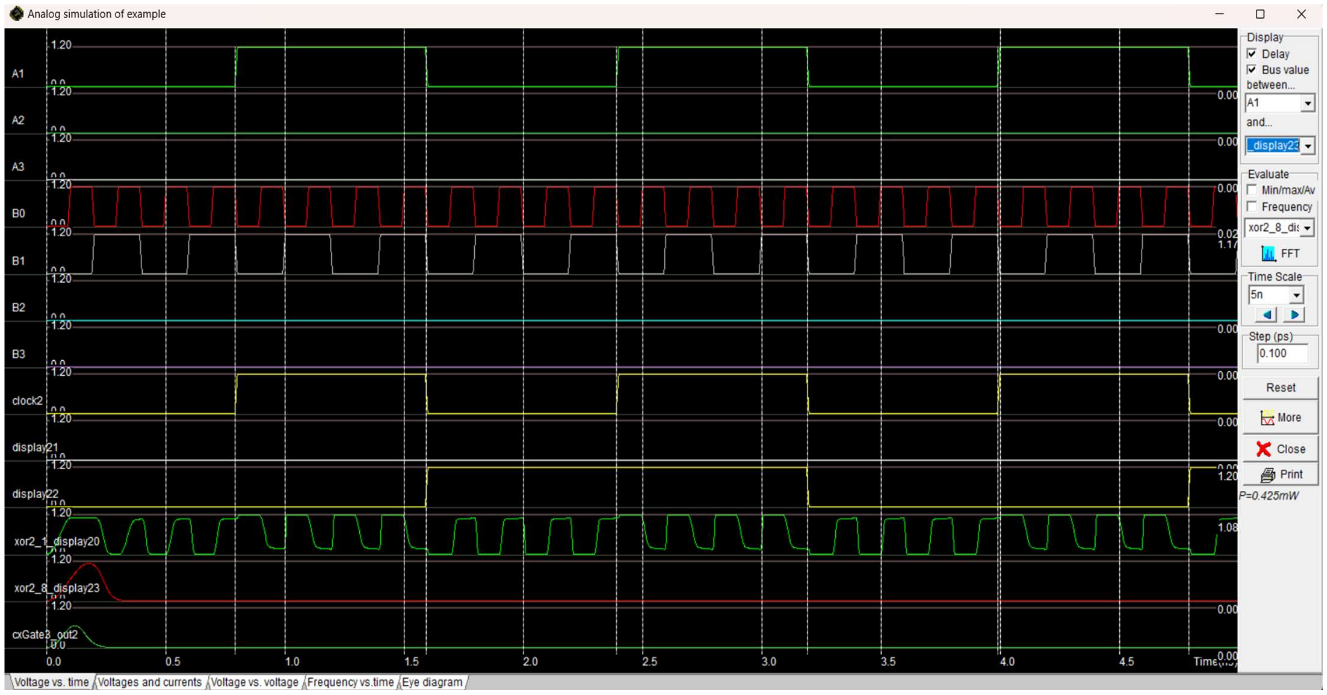
Operation: A multiplexer selects one of several input lines and forwards it to a single output line based on the values of its select lines. In a 4-bit ALU, the multiplexer takes outputs from different operation circuits (e.g., addition, subtraction, NAND, XOR) and selects one based on the control input.

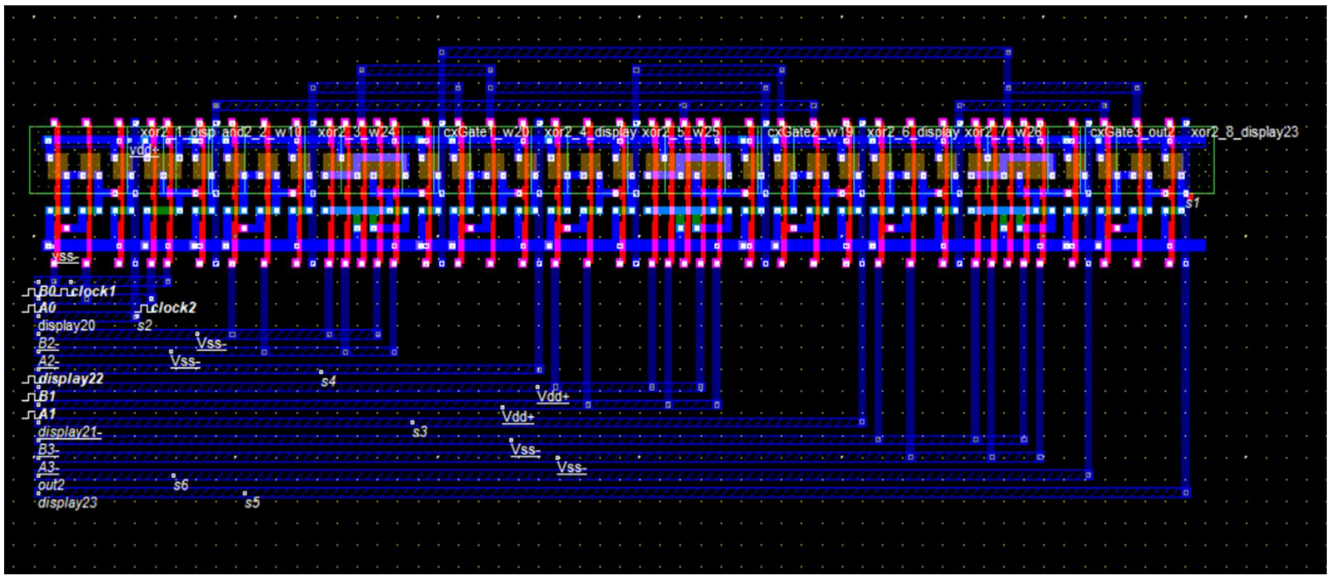
Role in ALU: The multiplexer serves as the ALU's operation selector, enabling the ALU to output the result of the desired operation. By setting the select lines, it determines whether the ALU performs addition, subtraction, NAND, or XOR on the inputs, effectively switching between different functionalities.

```
.subckt mux 48 49 50 51 52 53 54 ;52,53->select line, 54->output

xvac 52 2 55 inv
xvbc 53 2 56 inv

xvmt1 55 56 57 and ; 00 sum 01 subs 10 nand 11 xor
xvmt2 48 57 58 and

xvmt3 55 53 59 and
xvmt4 59 49 60 and

xvmt5 52 56 61 and
xvmt6 61 50 62 and

xvmt7 52 53 63 and
xvmt8 63 51 64 and

xvmt 58 60 65 or
xvmtt 62 64 66 or

xm 65 66 54 or
.ends
```

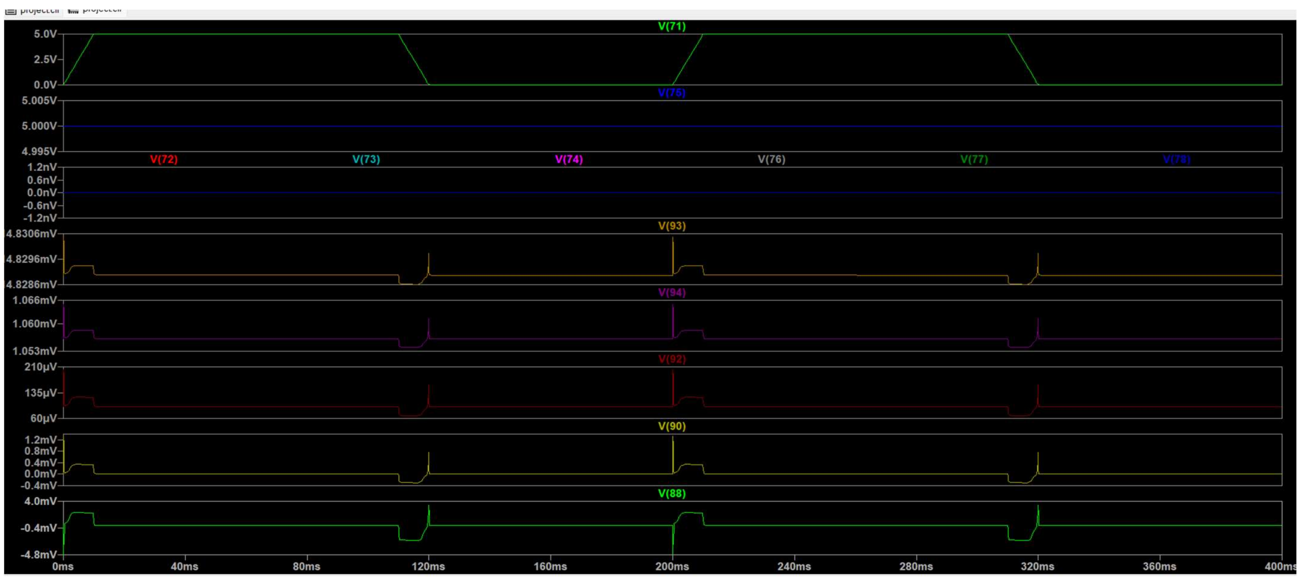
- **Operations in ALU:**
  Now, we will see voltage v/s Time analysis of circuit in LTSpice and in Microwind and Layout design of the all the 4 operation.
  1. **4-Bit Adder:**

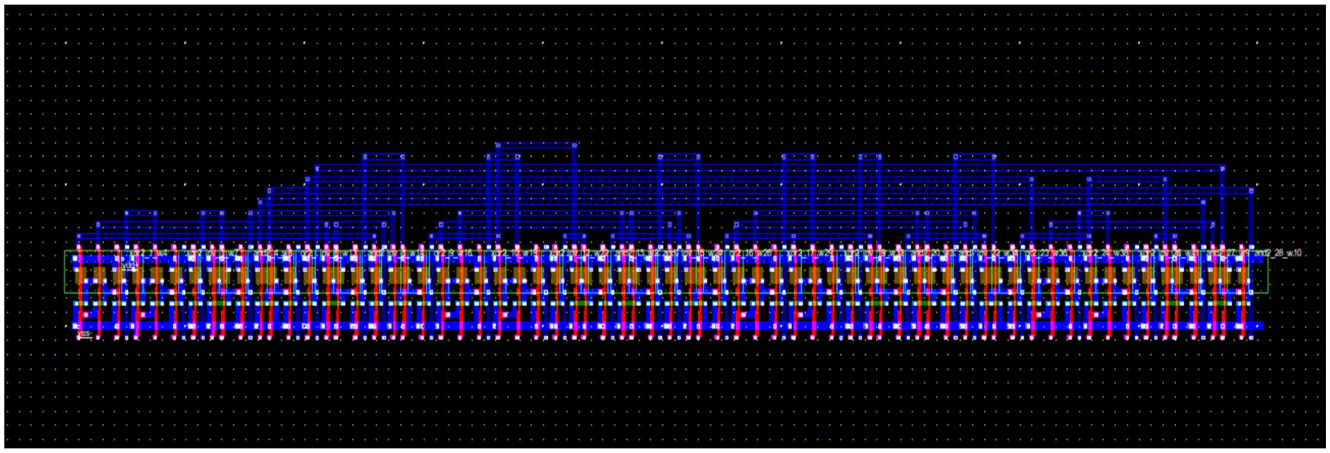



Analog simulation of example
A1
A2
A3
B0
B1
B2
B3
clock2
display21
display22
xor2_1_display20
xor2_8_display23
cxGate3_out2
Time(ns)
Display
Delay
Bus value between...
A1
and...
_display23
Evaluate
Min/max/Av
Frequency
xor2_8_dis
FFT
Time Scale
5n
Step (ps)
0.100
Reset
More
Close
Print
P=0.425mW
Voltage vs. time
Voltages and currents
Voltage vs. voltage
Frequency vs.time
Eye diagram

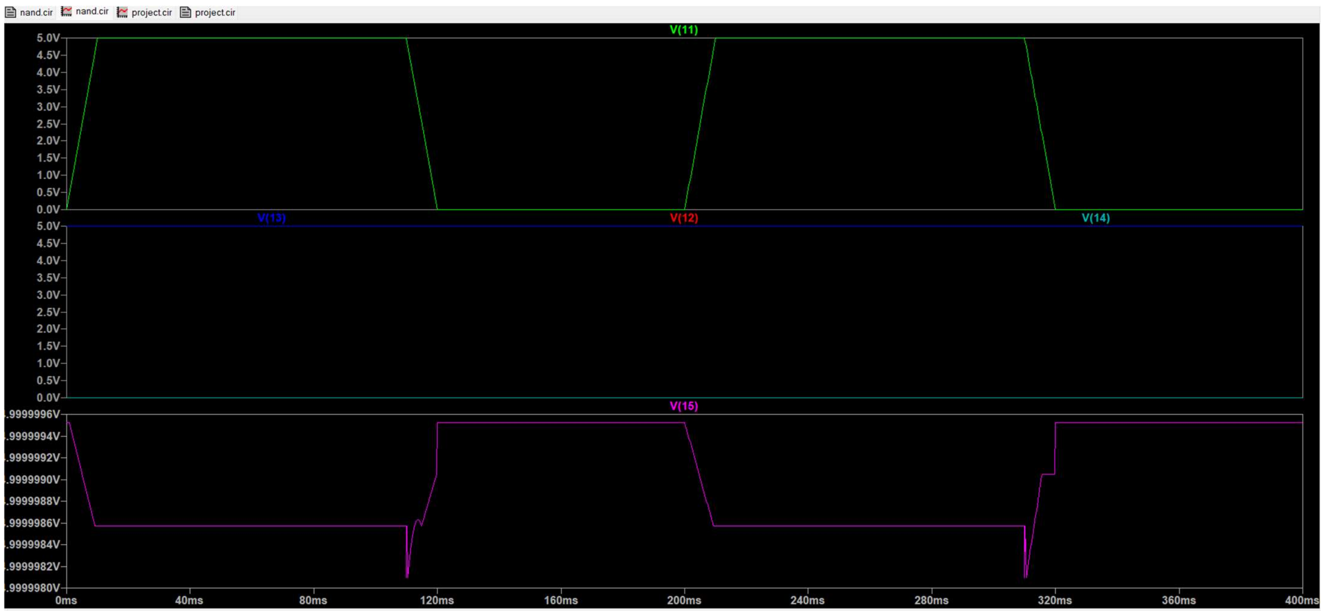

xor2_1_disp
and2_2_w10
xor2_3_w24
cxGate1_w20
xor2_4_display
xor2_5_w25
cxGate2_w19
xor2_6_display
xor2_7_w26
cxGate3_out2
xor2_8_display23
vdd+
vss-
s1
B0
clock1
A0
clock2
display20
s2
B2-
Vss-
A2-
display22
s4
B1
Vdd+
A1
display21-
s3
B3-
A3-
out2
s6
display23
s5

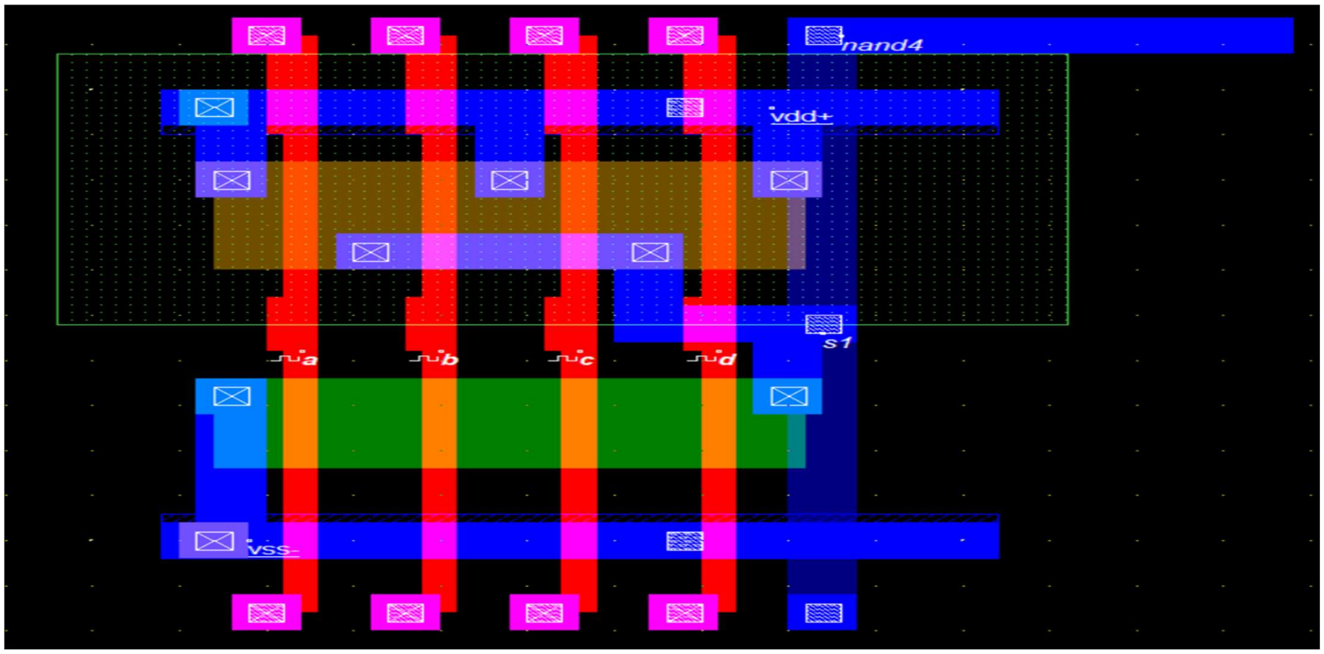
2. 4-Bit Substractor:



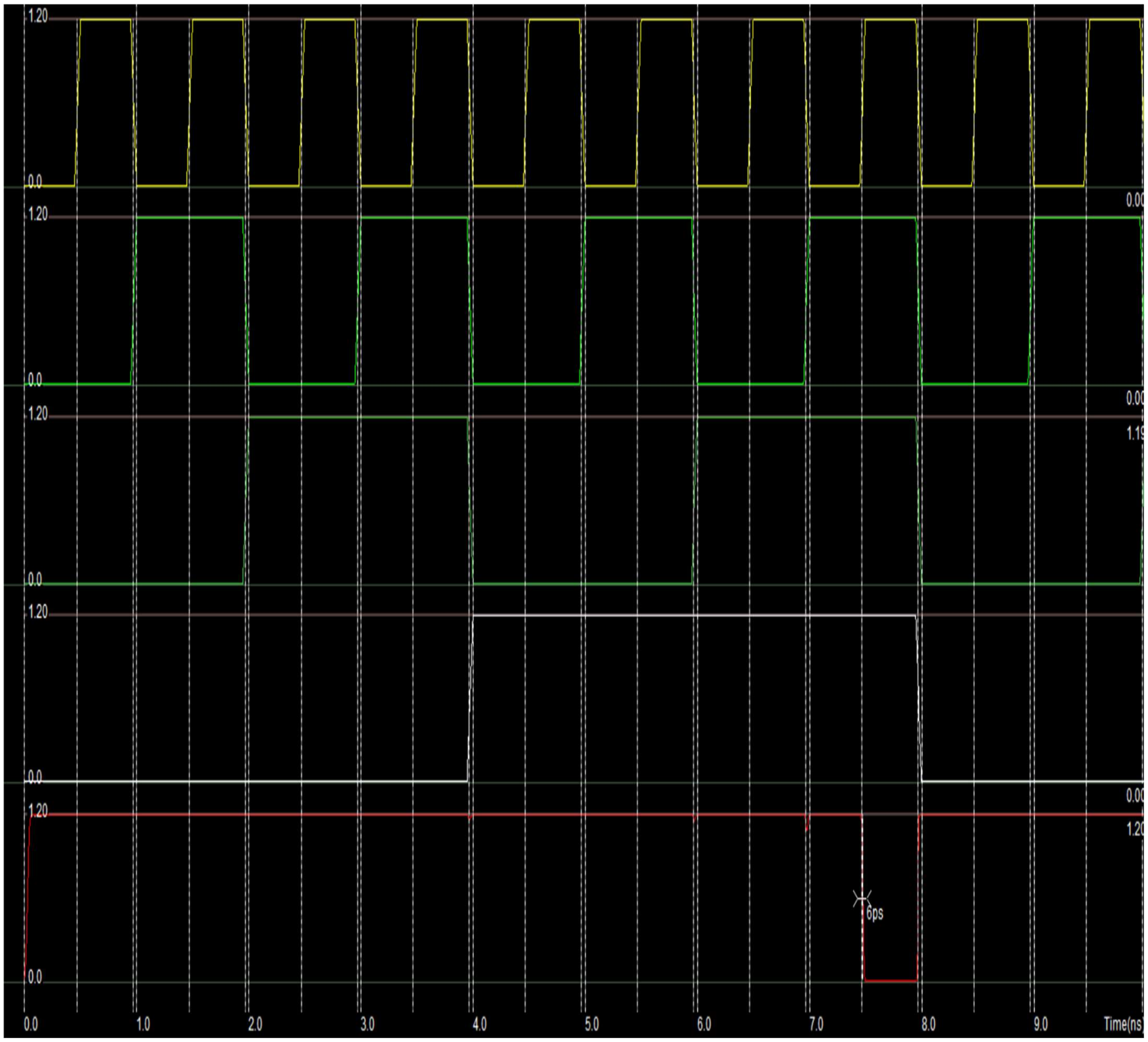


3. 4-Bit Nand:



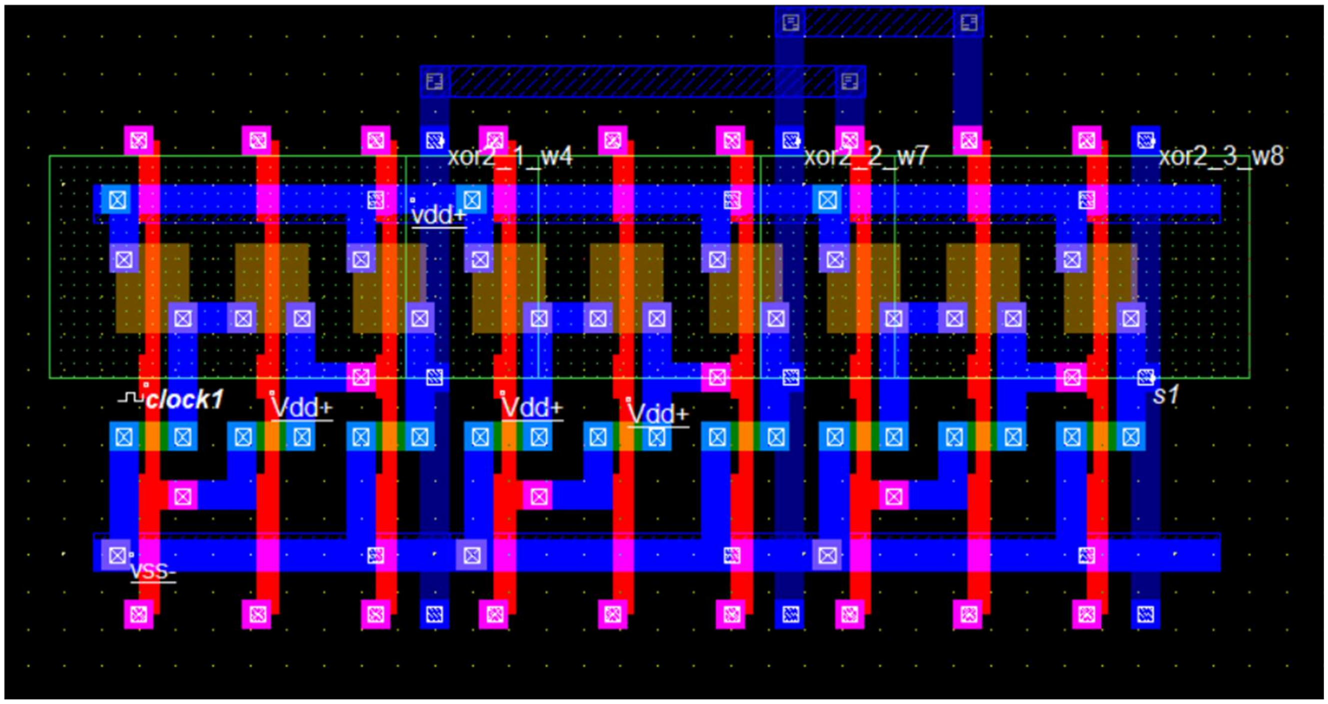


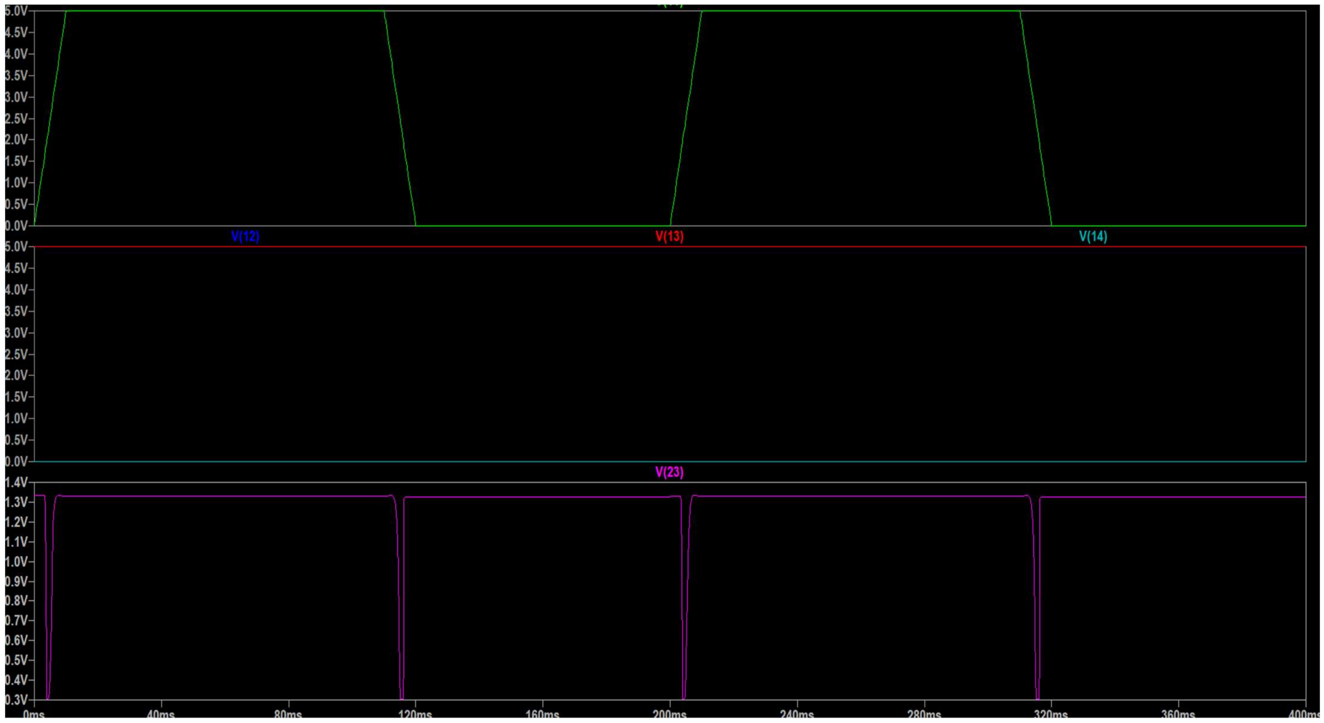
nand4
vdd+
s1
a
b
c
d
vss-

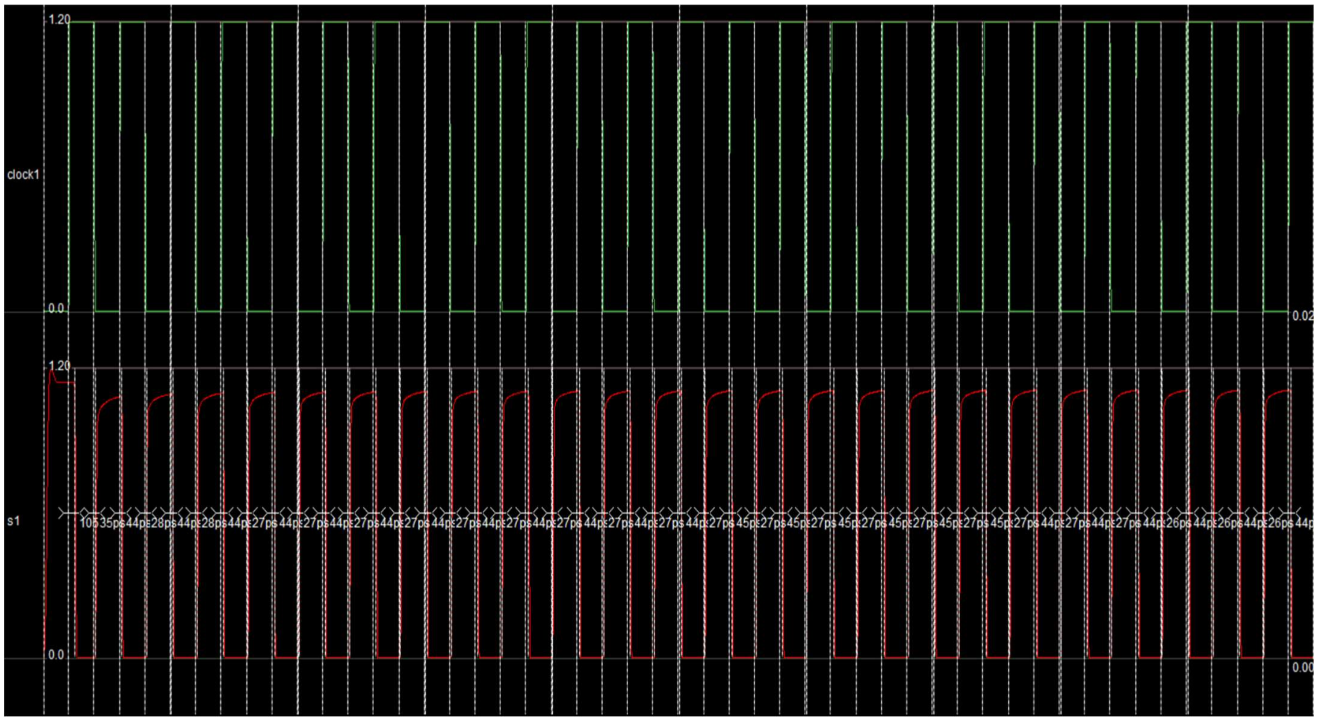


1.20
0.0
0.00
1.19
1.20
6ps
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
Time(ns

4. 4-Bit Xor:

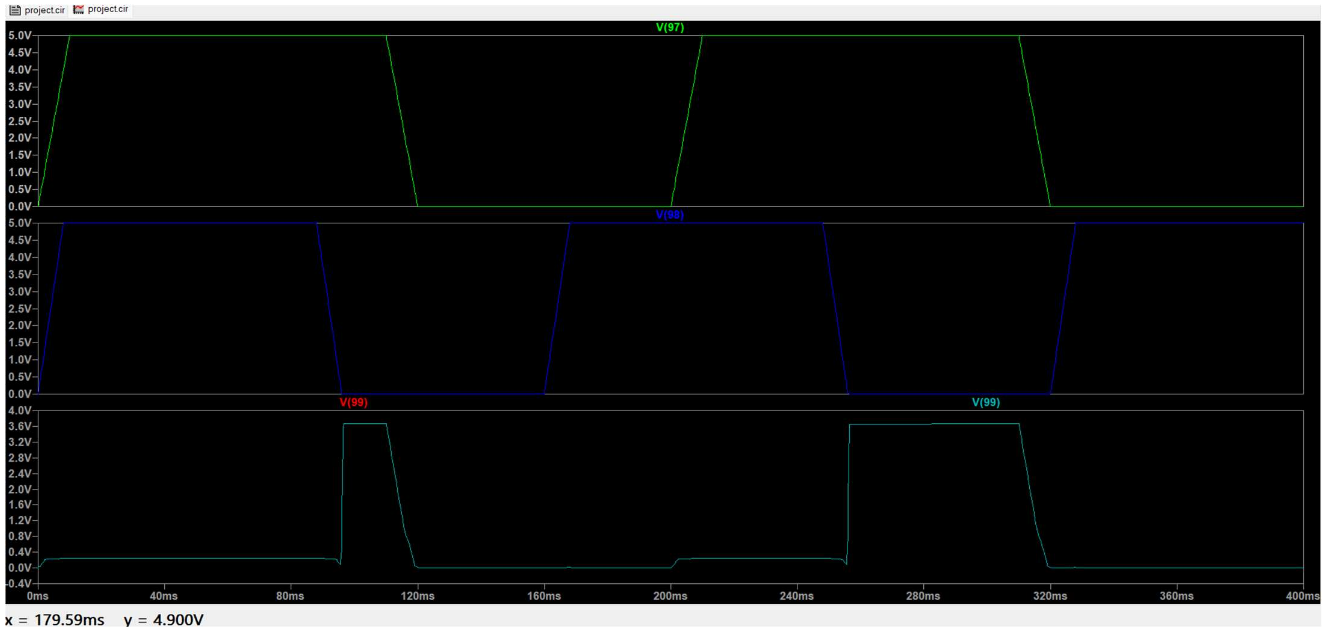

xor2_1_w4
xor2_2_w7
xor2_3_w8
vdd+
clock1
Vdd+
Vdd+
Vdd+
s1
vss-


5.0V
4.5V
4.0V
3.5V
3.0V
2.5V
2.0V
1.5V
1.0V
0.5V
0.0V
V(12)
V(13)
V(14)
V(23)
1.4V
1.3V
1.2V
1.1V
1.0V
0.9V
0.8V
0.7V
0.6V
0.5V
0.4V
0.3V
0ms
40ms
80ms
120ms
160ms
200ms
240ms
280ms
320ms
360ms
400ms


1.20
clock1
0.0
0.02
1.20
s1
105 35ps 44ps 28ps 44ps 28ps 44ps 27ps 44ps 27ps 44ps 27ps 44ps 27ps 44ps 27ps 44ps 27ps 44ps 27ps 44ps 27ps 44ps 27ps 44ps 27ps 45ps 27ps 45ps 27ps 45ps 27ps 45ps 27ps 45ps 27ps 45ps 27ps 44ps 27ps 44ps 27ps 44ps 26ps 44ps 26ps 44ps 26ps 44p
0.0
0.00

4-Bit Alu:

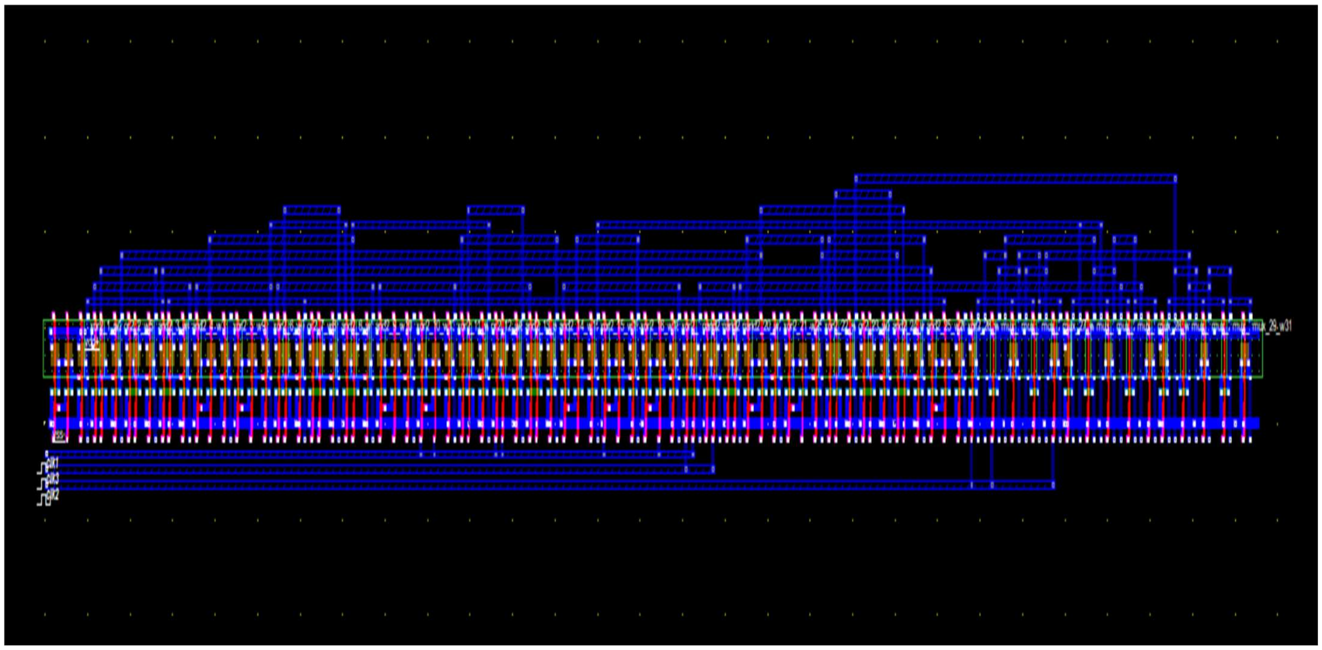
Voltage v/s Time analysis of 4-Bit Alu

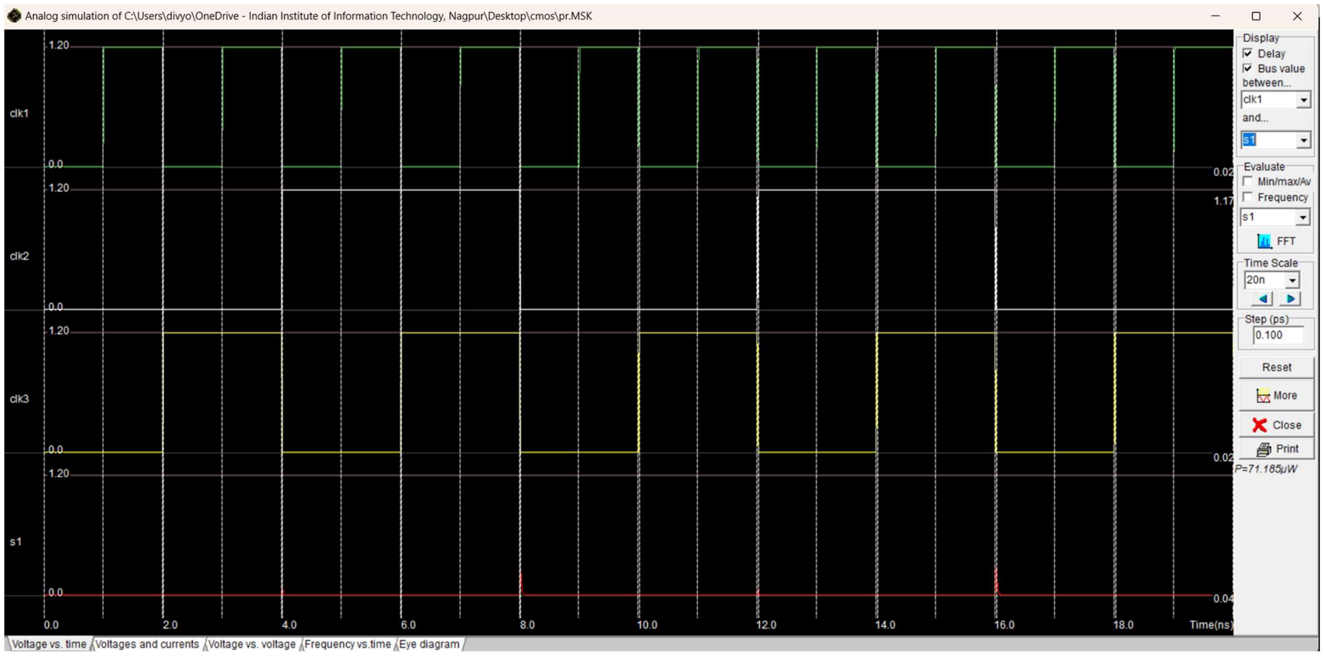


Here, v(97) and v(98) are the select lines. V(99) is the output

Logic level of select lines	Output
00	Cout of Adder
01	Bout of Substractor
10	Nand
11	Xor

Layout:





References:

- https://www.instructables.com/4-Bit-Binary-Adder-1/
- https://www.researchgate.net/figure/Sub-adder-4-bits-layout_fig2_271891339
- https://www.iosrjournals.org/iosr-jece/papers/Vol.%209%20Issue%203/Version-5/G09353643.pdf