4-Bit Carry Lookahead Adder

Pragnya T
ECE
International Institute of Information and Technology
Hyderabad
pragnya.tatiparthi@students.iiit.ac.in

Abstract

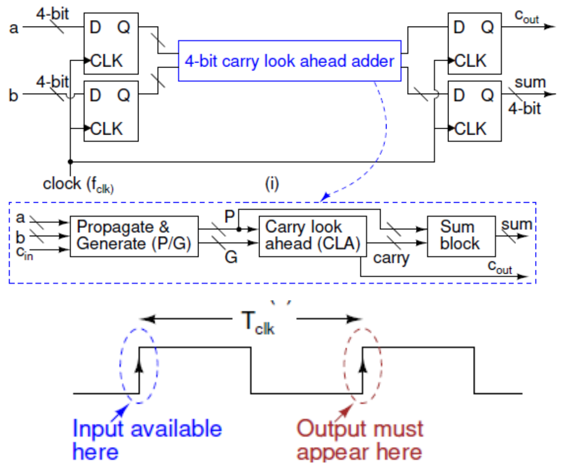
Carry look-ahead adder is a fast adder. In this project we're trying to optimize the CLA further. The main modification being the replacement of the XOR gate by a NOR gate. Now, the logic gates have less fan-in and fan-out and signal throughs one less MOS transistor in critical path. All gates are built with CMOS logic gates ant the input and output flipflops are implemented with TSPC logic. Simulation results show that the proposed architecture has advantages over conventional circuits in speed, area and power consumption.

Keywords—CLA, CMOS, NGSpice, Magic

I. Introduction

There are various logic designs to implement addition such as ripple adder, carry select adder. In these circuits, the carry-sum of higher bits depend on the carry-out of lower significant bits and this causes long propagation delay. Carry look-ahead adder solves this problem. This method makes the carry-sum of each bit to be calculated by inputs directly. Our task is to optimize the CLA further in terms of delay, power consumption and area.

In our project we have input and output flipflops to regulate when the inputs and outputs become available.



II. Implementation of Carry look-ahead Adder

A. The principle of 4-bit CLA

Function of the CLA is to calculate the carry-in of each bit from inputs directly. For addition with two inputs, the external carry-in is denoted as C_0. Both inputs have 4 bits and they are represented as $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$ respectively. Then sum of this addition $S_3S_2S_1S_0$ can be calculated by Full adders. Thus $S_3S_2S_1S_0 = A_3A_2A_1A_0 \oplus B_3B_2B_1B_0 \oplus C_3C_2C_1C_0$, where C3 C2 and C1 are the carry-outs of previous bits.

Carry-Propagation (P_i) and Carry-Generate (G_i) are obtained $P_i = A_i \oplus B_i$ and $G_i = A_i \,\&\, B_i = A_iB_i$

$$C_1 = G_0 + P_0C_0$$

$$C_2 = G_1 + P_1G_0 + P_1P_0C_0$$

$$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$$

$$G = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0$$

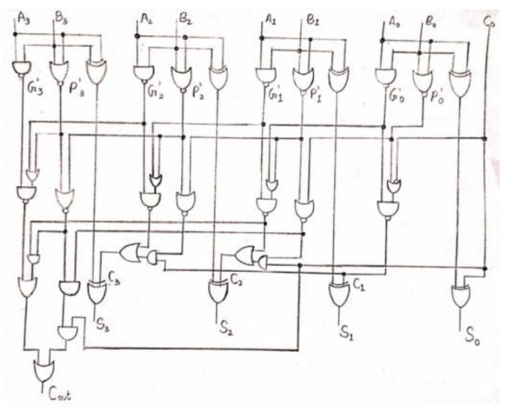
$$P = P_3 P_2 P_1P_0$$

B. *Proposed CLA Design*

In the original CLA multiple input AND and OR gates are required .But we try to modify the equations in such a way that only NAND and NOR gates are used so that we can avoid using extra transistors. Substituting G_0 and P_0 we get $C_1 = A_0 B_0 + (A_0 \oplus B_0)C_0$. C_1 is simplified to formula $C_1 = A_0 B_0 + (A_0+B_0)C_0$. We know that the values of A_0+B_0 and $A_0 \oplus B_0$ are different only when A_0, B_0 are both 1. In this situation, the final value of C_1 equals to '1' because $A_0\&B_0 = 1$.This tells us that we can replace XOR with NOT gate.

Using De Morgan's Laws we can get to the modified equations.

$$Pi' = (Ai+Bi)'$$
$$Gi' = (Ai.Bi)'$$
$$C1 = Cin$$
$$C2 = (G1' .(P1'+C1'))'$$
$$C3 = (G2' . (P2' + G1'))' + ((P2'+P1') . C1)$$
$$C4 = (G3'.(P3'+G2'))' + ((P3'+P2')' + (G1'.(P1'+C1'))')$$
$$Cout = (P4'+P3')'.(P2'+P1')'.C1 + (G4'.P4'+G3'))' + ((P4'+P3')' .(G2' .(P2'+G1'))')$$

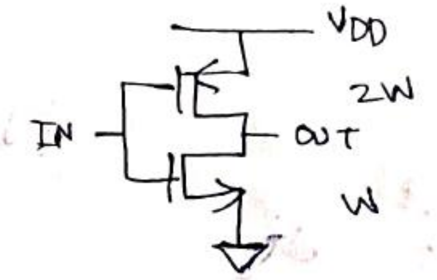


III. Design and Sizing

A. NOT GATE

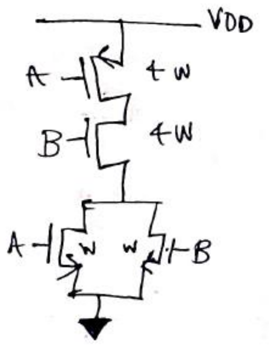
$W_P = 2W$

$W_N = W$



B. NOR GATE

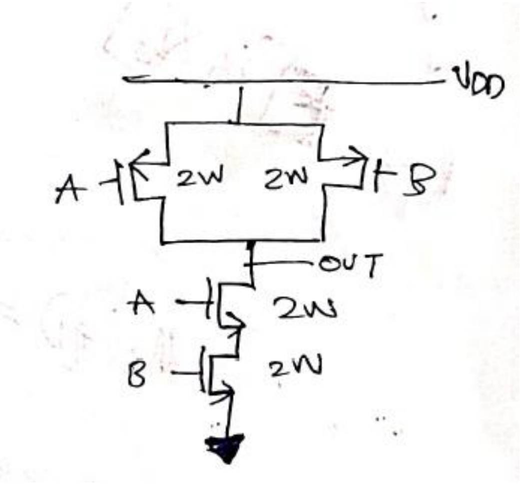
WP* = 4*W
WN* = *W



C. NAND GATE

$W_P = 2W$

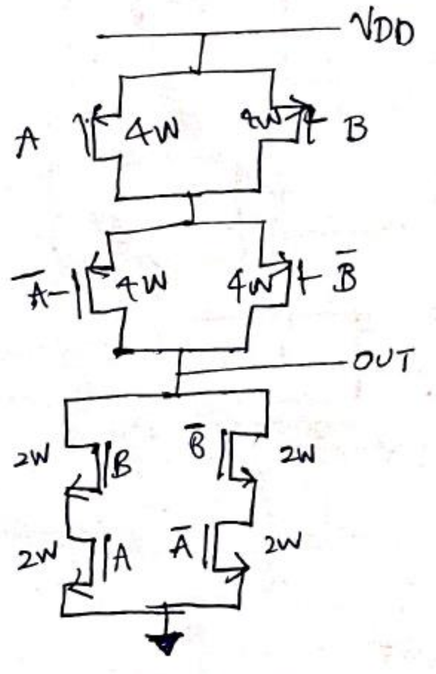
$W_N = 2W$



D.XOR GATE

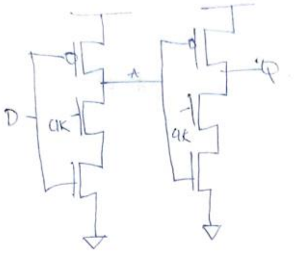
$W_P = 4W$

$W_N = 2W$



E. TSPC FLIPFLOP

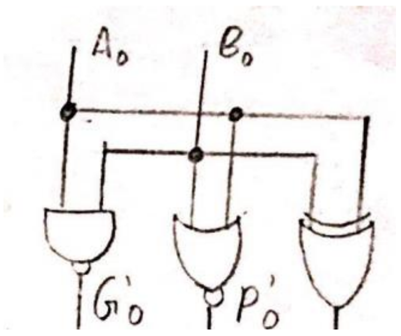
Flipflops have been implemented with TSPC logic to ensure we do not have to deal with different clocks. It makes it easier to implement the functionality



IV. ADDER MODULES

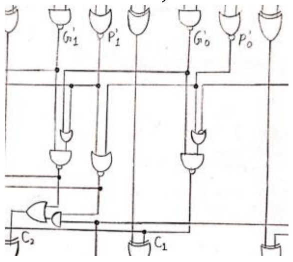
C. Propagate and Generate

PG module is used to generate Pi and Gi. It consists of one XOR,one NOR and one NAND.



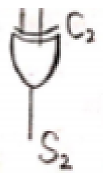
D. Carry Lookahead

It makes use of NOT, NAND and NOR gates



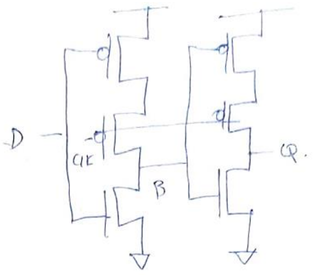
E. Sum Block

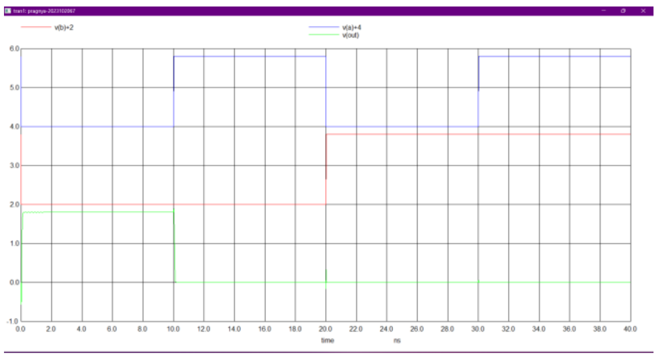
Consists of an XOR gate



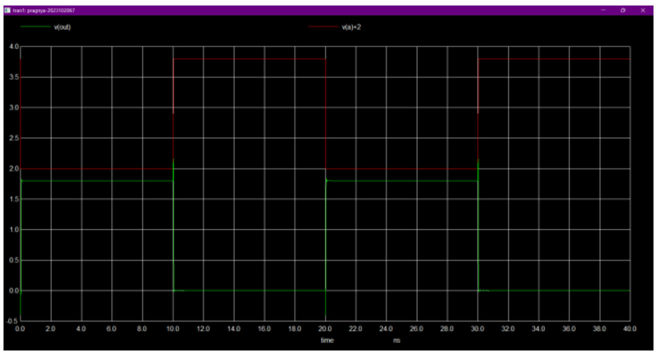
V. NGSPICE RESULT

NOR GATE

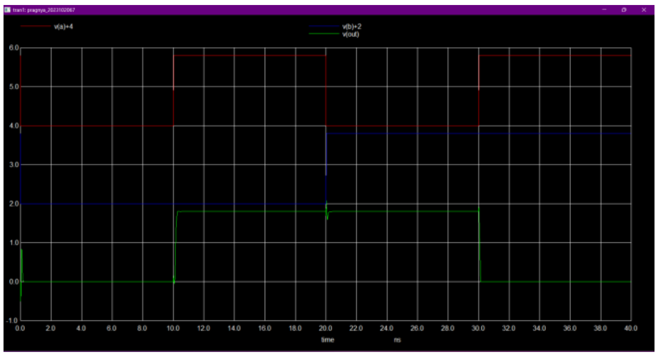




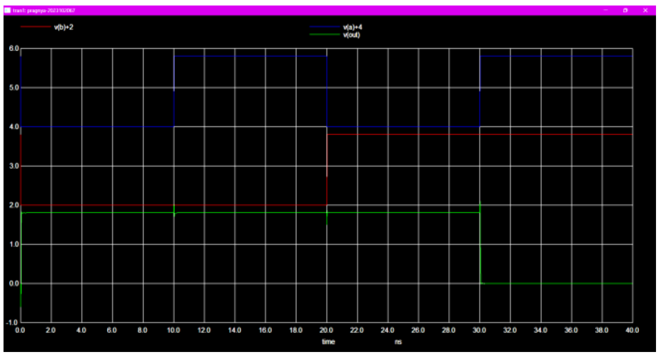
NOT GATE



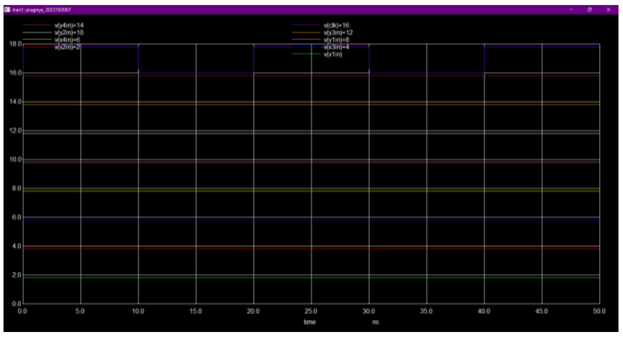
XOR GATE



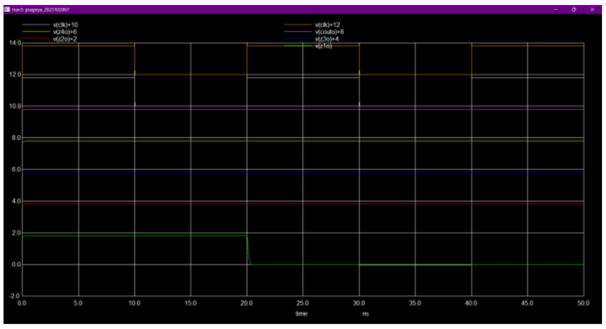
NAMD GATE



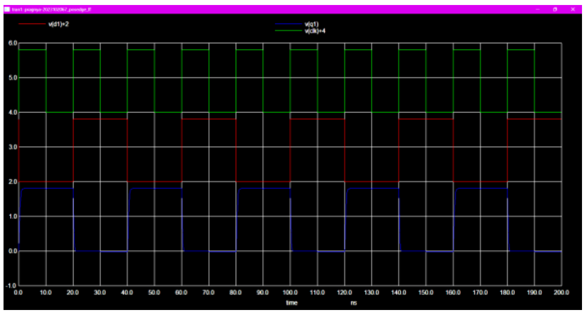
CLA INPUTS



CLA OP



FLIP FLOP



VI.DELAY CALCULATIONS FOR TSPC FLIPFLOP

In a TSPC there's never a direct path from Input to Output.So the hold time becomes 0.

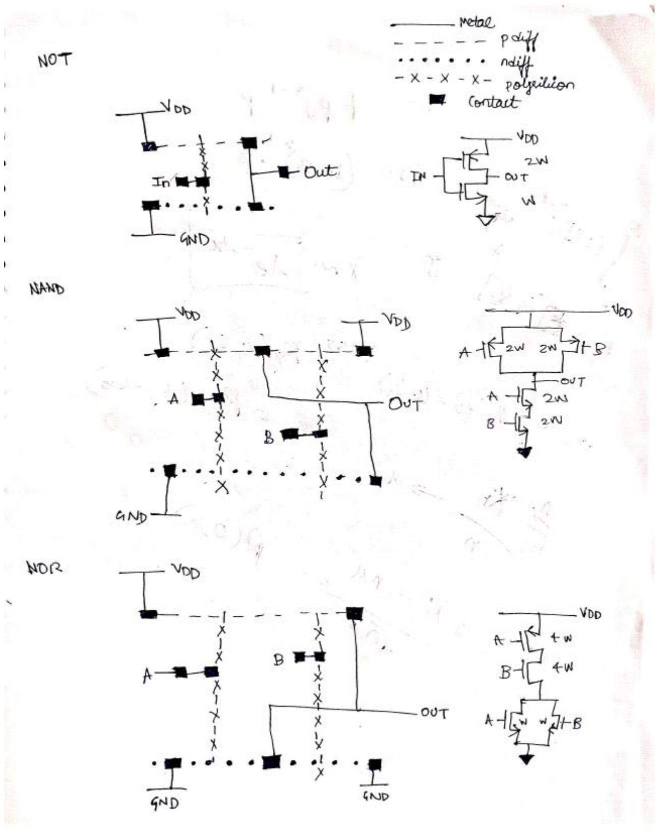
Pre-Layout

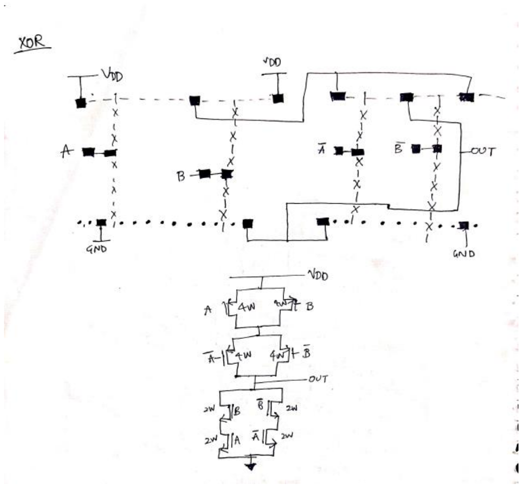
HOLD TIME – 0 sec
SETUP TIME – 214ps
Clock to Q - 4.11e-11

Post-Layout

HOLD TIME – 0 sec
SETUP TIME – 0.5ns
Clock to Q – 2.34e-9

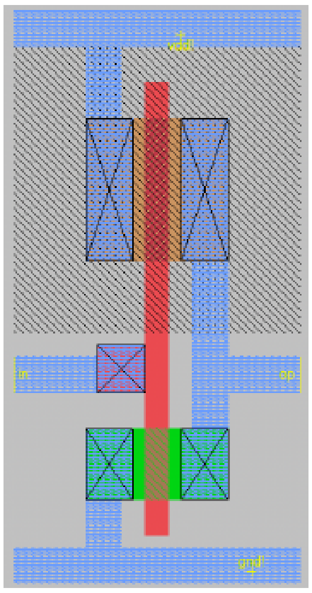
VII. STICK DIAGRAMS



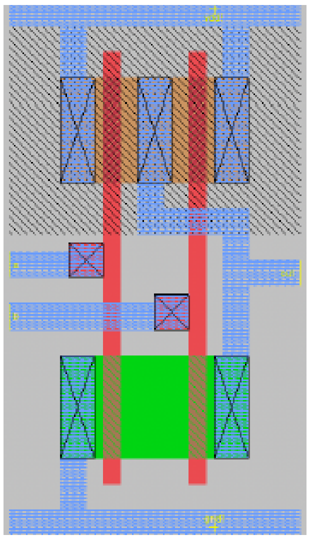


VII. MAGIC LAYOUTS

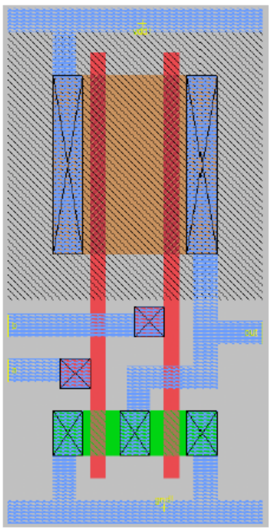
NOT GATE



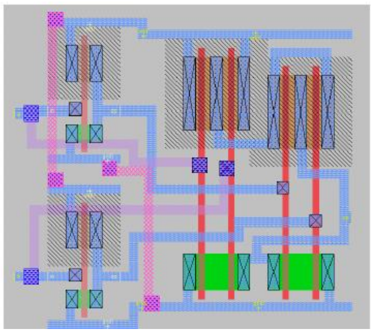
NOR GATE



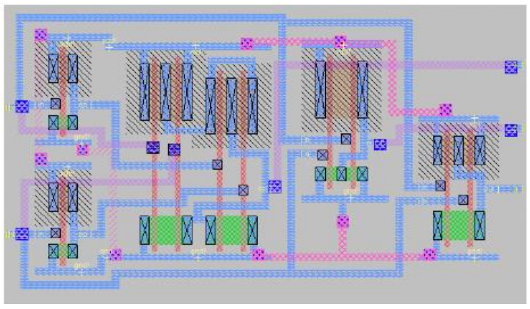
NAND GATE



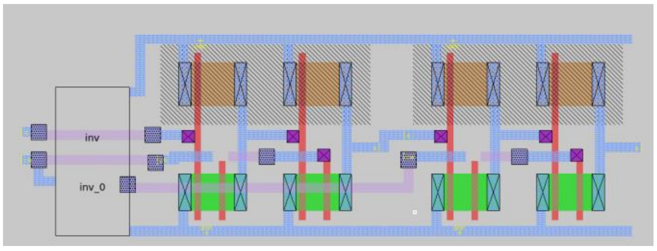
XOR GATE



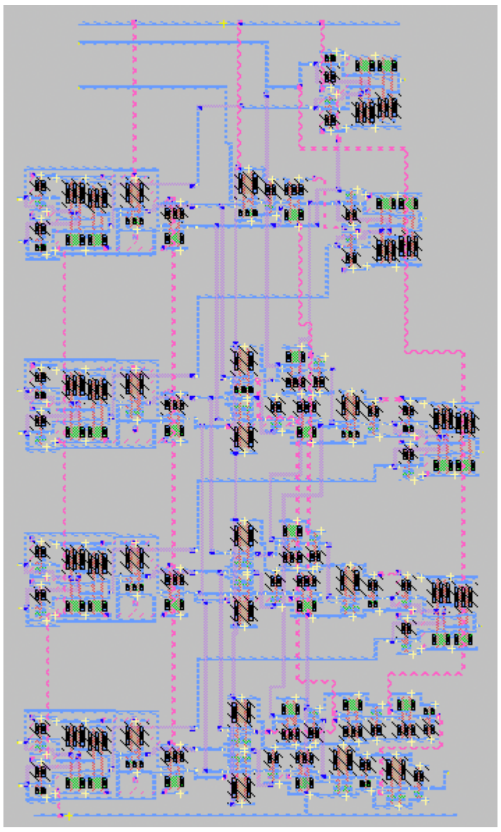
Pi and Gi



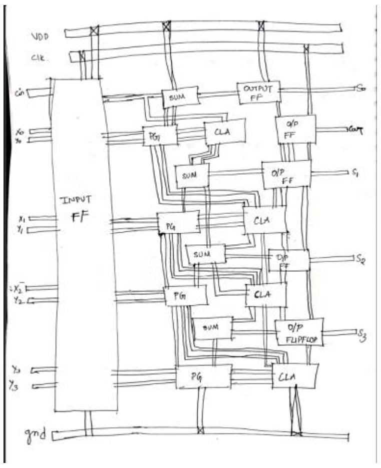
TSPC FLIPFLOP



FULL CRKT



IX. FLOOR PLAN



Horizontal and Vertical Pitches-

PG Generator-220 λ x 120 λ

CLA- 250λ x 150λ

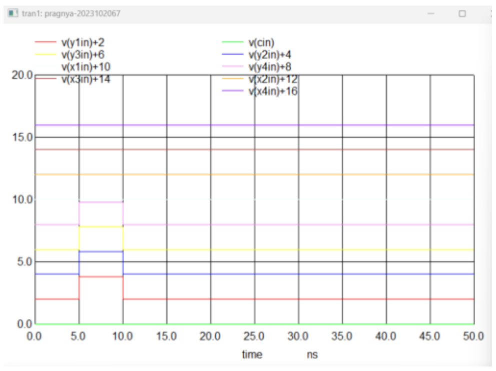
SUM Module- 125λ ✕ 100λ

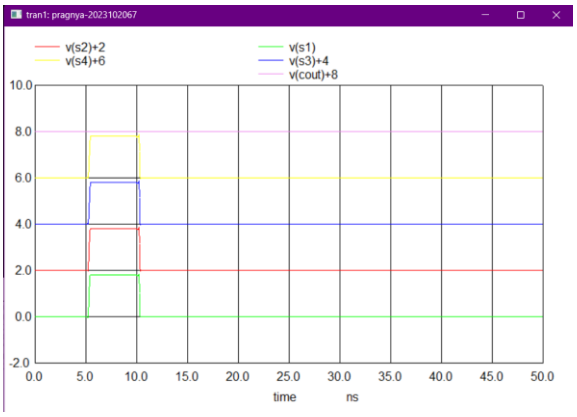
FLIPFLOP-201 x 66 (microns)

Total – 950 λ x 1400 λ

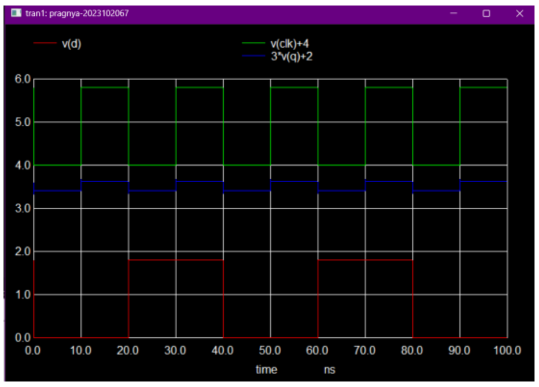
IX. POST LAYOUT

FULL ADDER

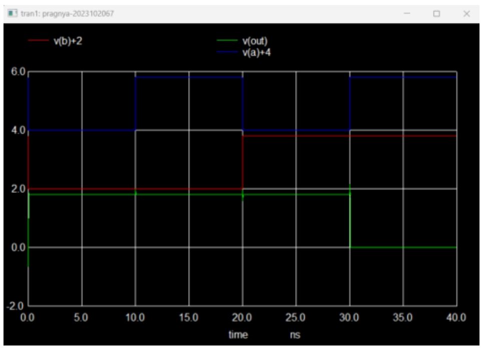




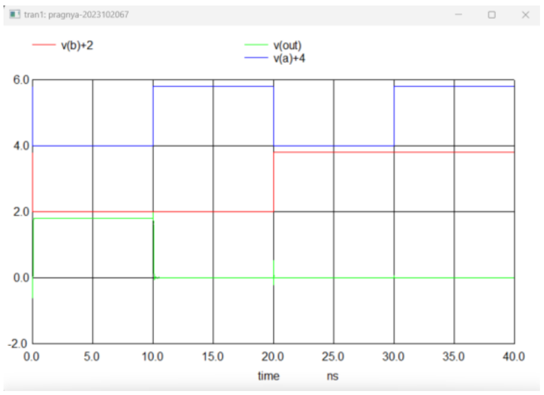
FLIPFLOP



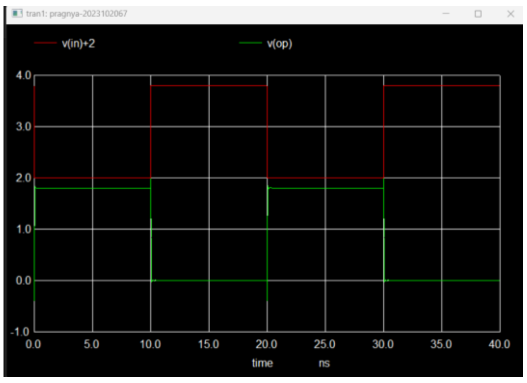
NAND



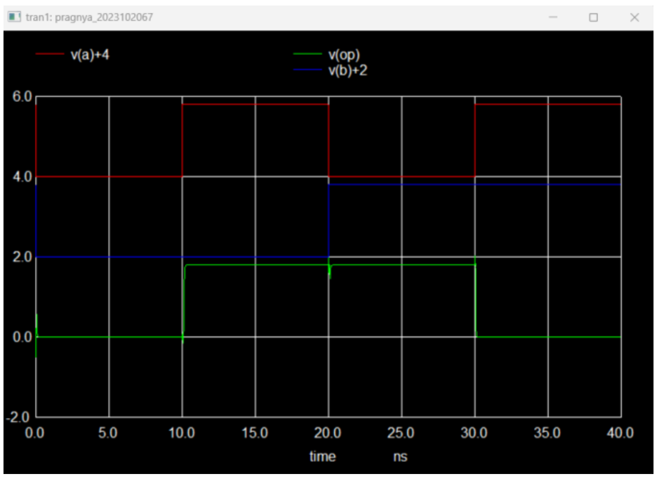
NOR



NOT



XOR



Worst case delay for adder – 1.98 ns

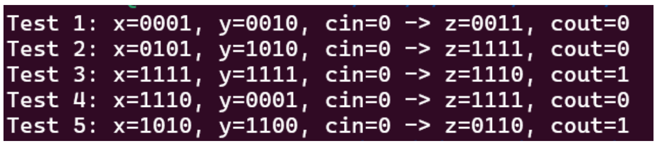
A	*B*	*Pre*	*Post*
0000	*0000*	*1.24*	*1.279*
1111	*0000*	*1.87*	*1.98*
0111	*0010*	*1.32*	*1.41*
0110	*0011*	*0.98*	*1.12*

Max clk frequency- 1 GHz

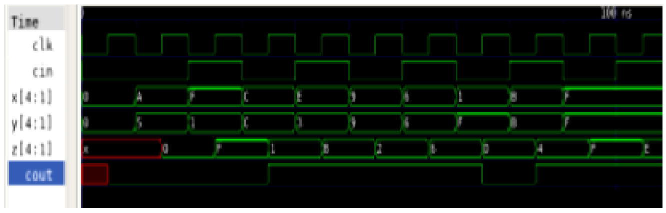
Total delay for Pre-Layout is 1.87 ns

Total delay for Pre-Layout is 1.98 ns

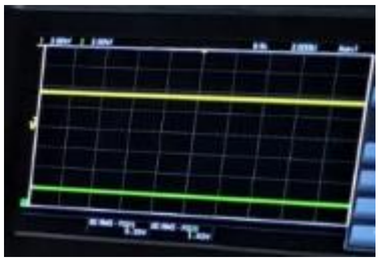
IX. VERILOG



```
Test 1: x=0001, y=0010, cin=0 -> z=0011, cout=0
Test 2: x=0101, y=1010, cin=0 -> z=1111, cout=0
Test 3: x=1111, y=1111, cin=0 -> z=1110, cout=1
Test 4: x=1110, y=0001, cin=0 -> z=1111, cout=0
Test 5: x=1010, y=1100, cin=0 -> z=0110, cout=1
```



IX. DSO OUTPUTS



DSO showing S0 S1 S2 S3 Cout for the input values

SUM - 11110

X-1111

Y-1111

DSO Setup

References

[1] *J. Miao and S. Li, "A novel implementation of 4-bit carry look-ahead adder," 2017 International Conference on Electron Devices and Solid-State Circuits*

[2] *Pai, Y. and Y. Chen. "The fastest carry lookahead adder."*

[3] *CMOS VLSI Design (fourth edition) by Weste and Harris.*

[4] *Fundamentals of Digital Logic with Verilog Design by Brown and Vranesic.*

[5] *Digital Logic and Computer Design by Morris Mano.*

[6] *Computer Architecture and Organization by John P. Hayes.*

[7] *Class Notes*