VLSI PROJECT - 4-Bit CLA Adder

Vansh Agarwal
2023102043
vansh.agarwal@students.iiit.ac.in

***Abstract*—The Carry Lookahead Adder (CLA) improves arithmetic operation speed by minimising carry propagation delay through propagate and generate logic. This paper details the CLA's design, focusing on parallel carry computation and its advantages over ripple-carry adders in speed and scalability. Implementation results demonstrate its efficiency in high-speed processors and digital systems, making it a key component for modern computing applications.**

INTRODUCTION

A 4-bit Carry Look-Ahead (CLA) Adder adds two 4-bit numbers and produces a 4-bit sum along with a 5th-bit carry. Unlike a conventional 4-bit Ripple Carry Adder, which calculates each carry sequentially, the CLA Adder computes all carries simultaneously, thereby reducing propagation delay.

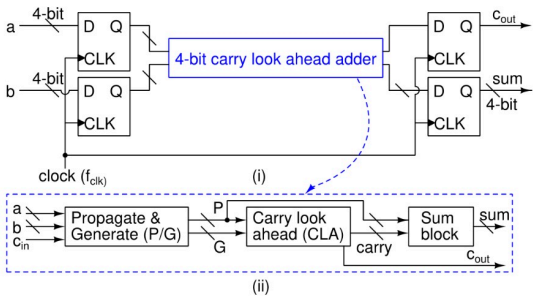


Fig. 1: Various circuit blocks for our Adder

1. BASIC WORKING AND EQUATIONS FOR THE CLA ADDER

To create the three blocks shown in Fig. 1 part (ii), we first generate the p_i(propagate) and g_i (generate) signals for the Carry Lookahead Adder (CLA) block. These are defined as:

$$p_i = a_i \oplus b_i$$
$$g_i = a_i \cdot b_i$$

Next, the carries in the CLA block are calculated using the following logic:

$$c_{i+1} = p_i \cdot c_i + g_i$$

From this, all carry values c_i can be derived from the propagate (p_i) and generate (g_i) signals:

$$c_1 = (p_0 \cdot c_0) + g_0$$
$$c_2 = (p_1 \cdot p_0 \cdot c_0) + (p_1 \cdot g_0) + g_1$$
$$c_3 = (p_2 \cdot p_1 \cdot p_0 \cdot c_0) + (p_2 \cdot p_1 \cdot g_0) + (p_2 \cdot g_1) + g_2$$
$$c_4 = (p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_0) + (p_3 \cdot p_2 \cdot p_1 \cdot g_0) + (p_3 \cdot p_2 \cdot g_1) + (p_3 \cdot g_2) + g_3$$

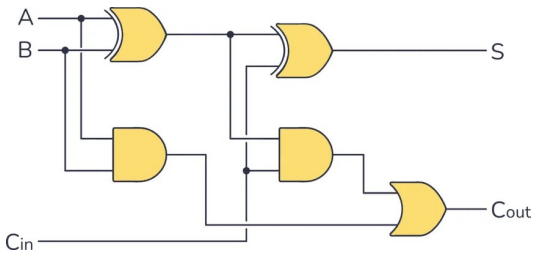


Fig. 2: Full Adder Circuit

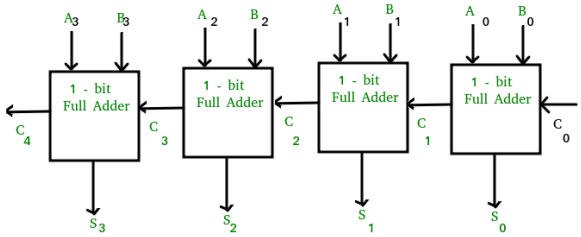


Fig. 3: Circuit implementation

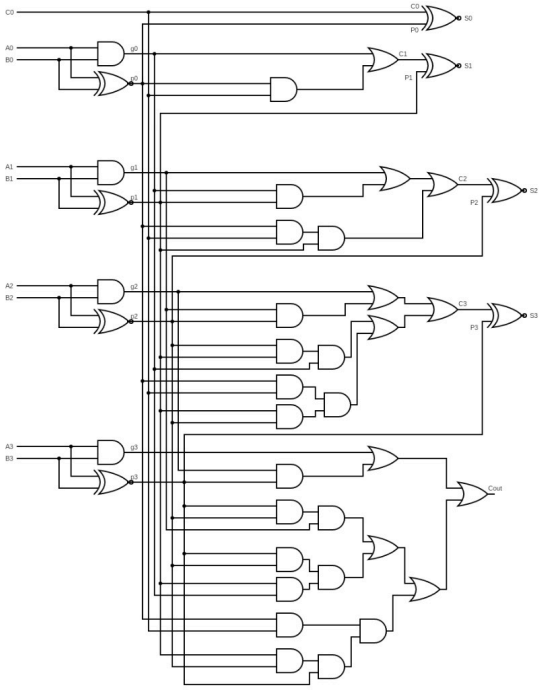


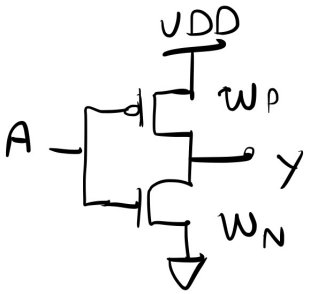
Fig. 4: Full circuit for CLA adder

2. DESIGN TOPOLOGY AND SIZING

A. CLA Adder :

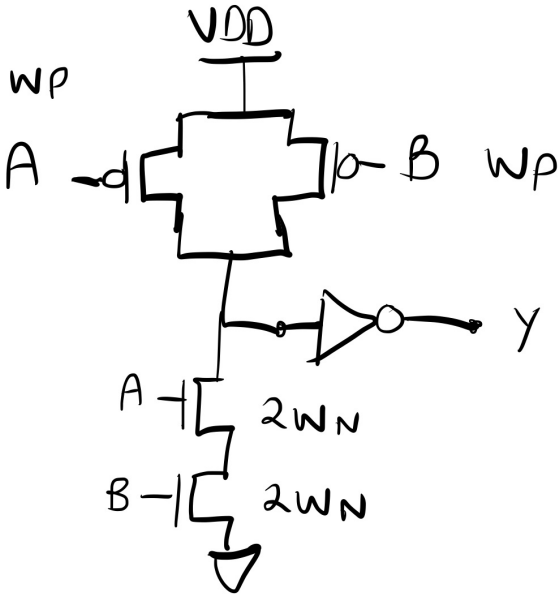
For XOR gate , PTL(Pass Transistor Logic) Topology was implemented , and CMOS Topology for inverters , AND & OR gates.

NOT GATE



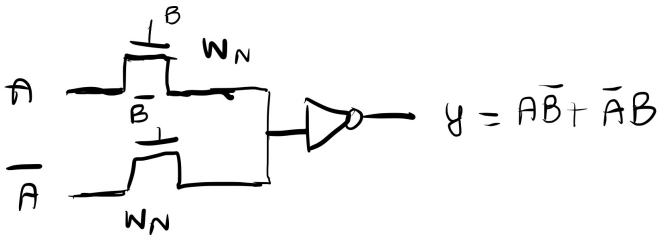
$W_P = 2W_N = 3.6\mu$

AND GATE



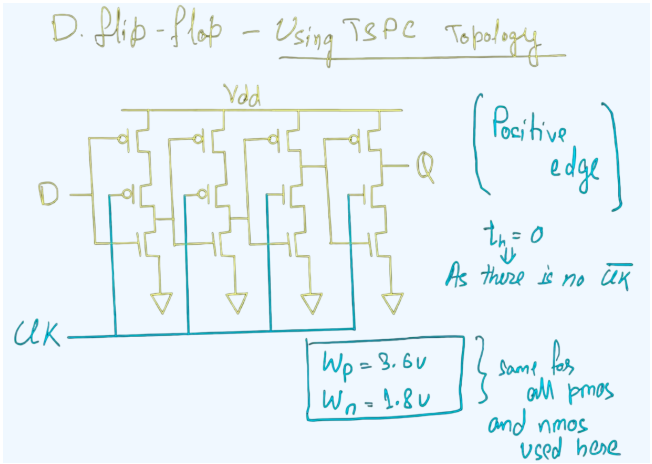
$W_P = 2W_N = 3.6\mu$

XOR GATE

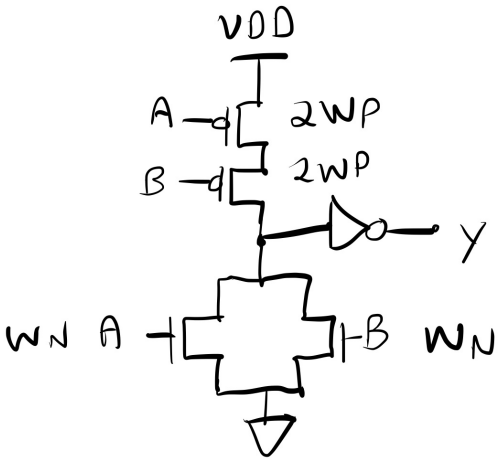


$W_N = 1.8\mu$

B. D-FLIPFLOP



OR GATE



$W_P = 2W_N = 3.6\mu$

3. PRE-LAYOUT SIMULATIONS IN NGSPICE

A. CLA Adder :

For CLA Adder, we take input as a= 1001 and b= 1001 at t=5ns. The output is as follows.

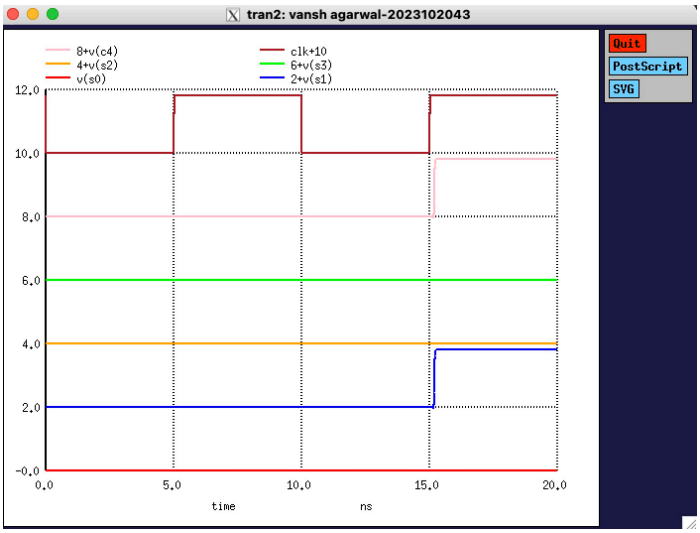


Fig. 5: Final output of the circuit

The output is s= 10110 which is correct for give input.

1001 + 1001 = 10010

The system operates as expected. To analyze propagation delays, we assign a= 1001 and b= 1001 and Cin=1 , even though Cin is usually 0 in typical adder applications. This results in an output of s = 11111, allowing us to examine the delays in generating each sum bit and the final carry.

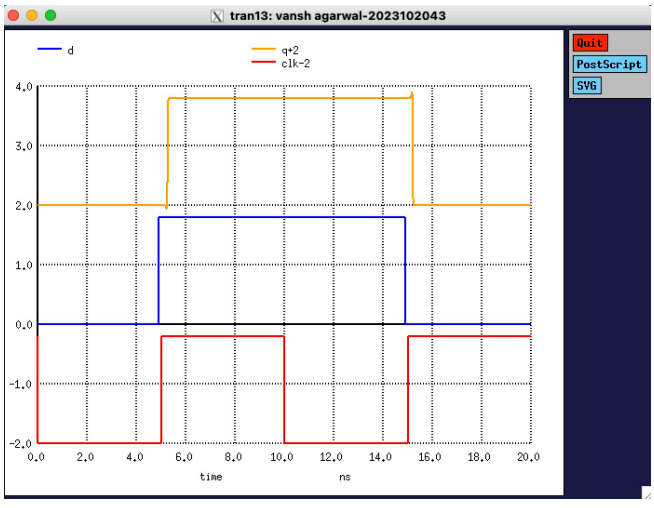
```
Measurements for Transient Analysis

tpd_s0          =  1.132424e-10 targ=  3.118242e-09 trig=  3.005000e-09
tpd_s1          =  2.368528e-10 targ=  3.241853e-09 trig=  3.005000e-09
tpd_s2          =  5.539943e-10 targ=  3.558994e-09 trig=  3.005000e-09
tpd_s3          =  6.645967e-10 targ=  3.669597e-09 trig=  3.005000e-09
tpd_carry       =  6.077316e-10 targ=  3.612732e-09 trig=  3.005000e-09
```

From this we get t_{pdmin} = 113.2424 ps and t_{pdmax} = 607.7316 ps

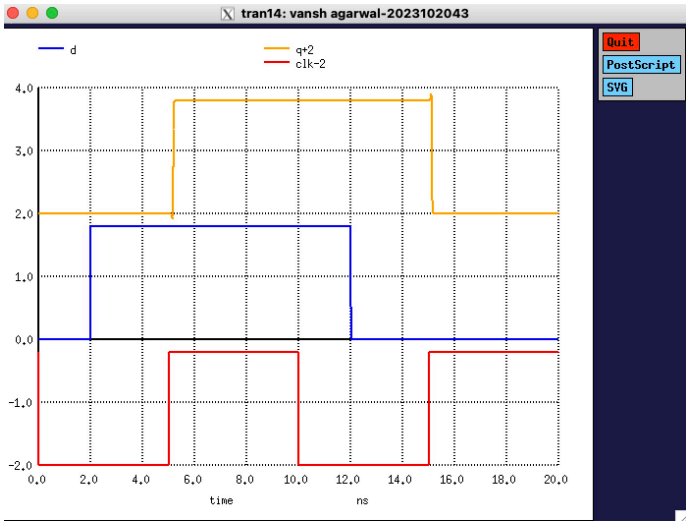
B. D-flipflop :

For the D-flipflop (+ve edged), we get the following simulation :

Simulation for t_{pcqmax} :



Simulation for t_{pcqmin} :



It can be observed that Q is being updated at the +ve rising edge , hence verifying that the D-flipflop is working correctly .

4. SETUP TIME, HOLD TIME AND PROPAGATION DELAY FROM CLK TO Q IN FLIPFLOP

We implemented a D-Flip-Flop using the TSPC topology. Since the design does not utilise a complement of the clock signal, the hold time (t_{hold}) is effectively zero. For setup time (t_{setup}), we determined through trial and error that it is 0.11 ns. If the input is applied less than t_{setup}before the next positive clock edge, the input either fails to register or causes output corruption.
The propagation delay from the clock to Q (t_{pcq}) is illustrated in the figure above, which provides the minimum and maximum delay values. These values are summarised as follows:

a) t_{pcqmax} :

```
Measurements for Transient Analysis

tpcq_r          =  2.498728e-10 targ=  5.264873e-09 trig=  5.015000e-09
tpcq_f          =  1.890267e-10 targ=  1.520403e-08 trig=  1.501500e-08
tpcq            =  2.19450e-10
```

b) t_{pcqmin} :

```
Measurements for Transient Analysis

tpcq_r          =  1.672032e-10 targ=  5.182203e-09 trig=  5.015000e-09
tpcq_f          =  1.163537e-10 targ=  1.513135e-08 trig=  1.501500e-08
tpcq            =  1.41778e-10
```

We get t_{pcqmin} = 0.141ns and t_{pcqmax}= 0.219ns.

5. STICK DIAGRAMS FOR ALL UNIQUE GATES AND FLIPFLOP

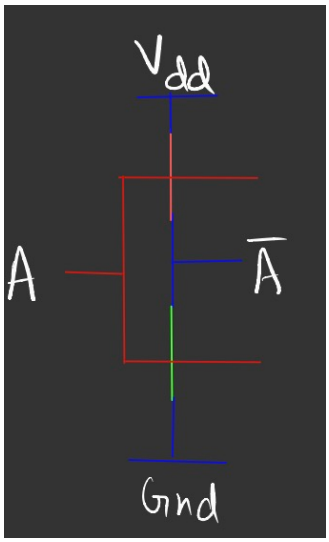


Fig. 11 : Stick Diagram for NOT gate

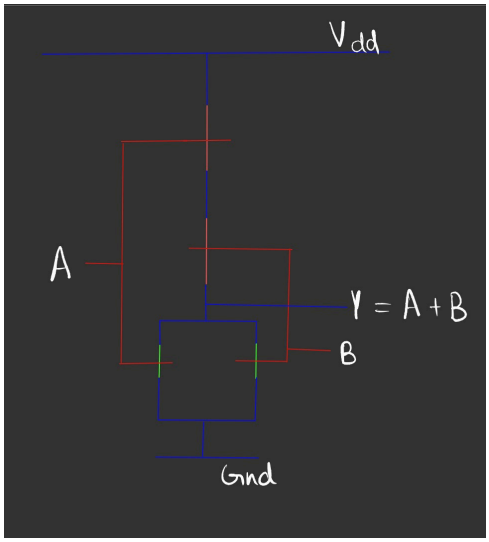


Fig. 14 : Stick Diagram for OR gate

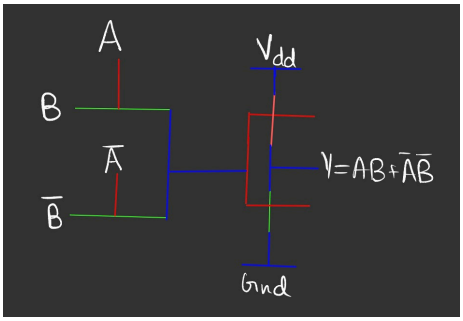


Fig. 12 : Stick Diagram for XOR gate

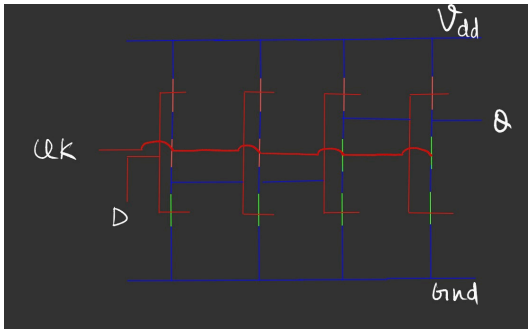


Fig. 15: Stick Diagram for D-FLIPFLOP

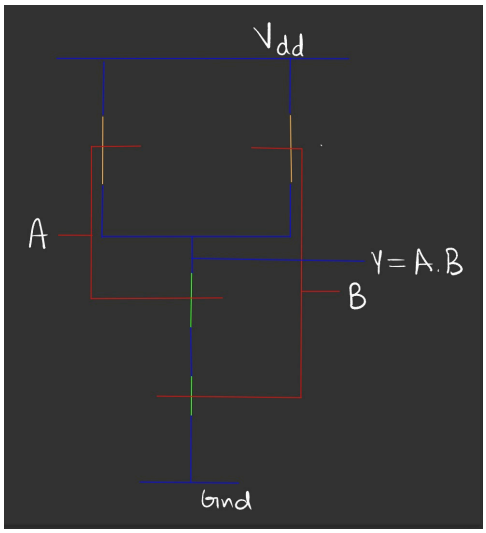


Fig. 13 : Stick Diagram for AND gate

6. POST LAYOUT SIMULATIONS

A. D-FLIPFLOP :

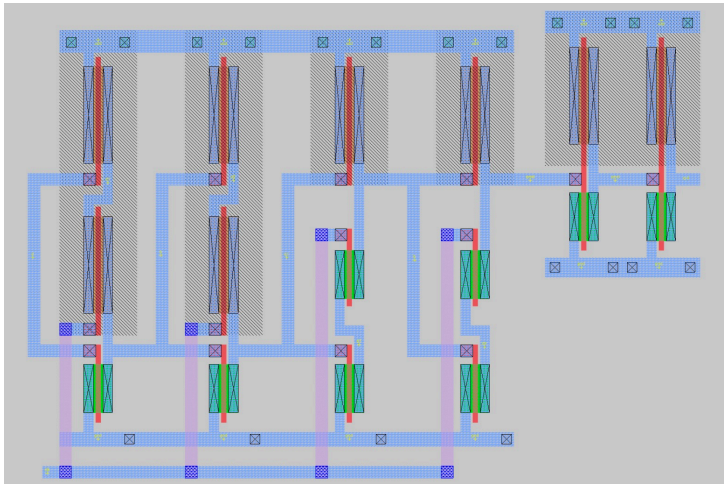


Fig. 16: Magic layout for D-FLIPFLOP

On performing post-Layout simulations on the circuit made on MAGIC software :

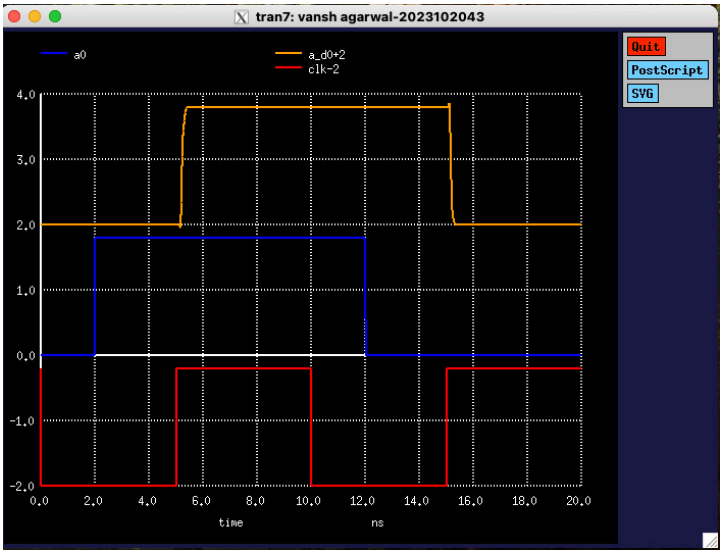


Fig. 17:Post-Layout simulation for $t_{pcq_{min}}$:

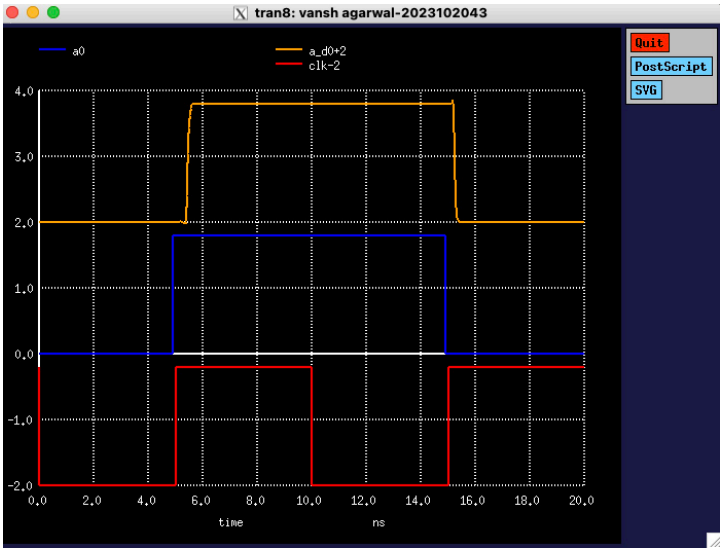


Fig. 18:Post-Layout simulation for $t_{pcq_{max}}$:

a) $t_{pcq_{min}}$:

```
   Measurements for Transient Analysis

tpcq_r              =  2.078833e-10 targ=  5.222883e-09 trig=  5.015000e-09
tpcq_f              =  1.584898e-10 targ=  1.517349e-08 trig=  1.501500e-08
tpcq                =  1.83187e-10
```

b) $t_{pcq_{max}}$:

```
   Measurements for Transient Analysis

tpcq_r              =  4.316071e-10 targ=  5.446607e-09 trig=  5.015000e-09
tpcq_f              =  2.274975e-10 targ=  1.524250e-08 trig=  1.501500e-08
tpcq                =  3.29552e-10
```

Comparison with Pre-Layout :

Parameter	Pre-Layout	Post-Layout
$t_{pcq_{min}}$	0.136 ns	0.183 ns
$t_{pcq_{max}}$	0.207 ns	0.329 ns

B. CLA ADDER :

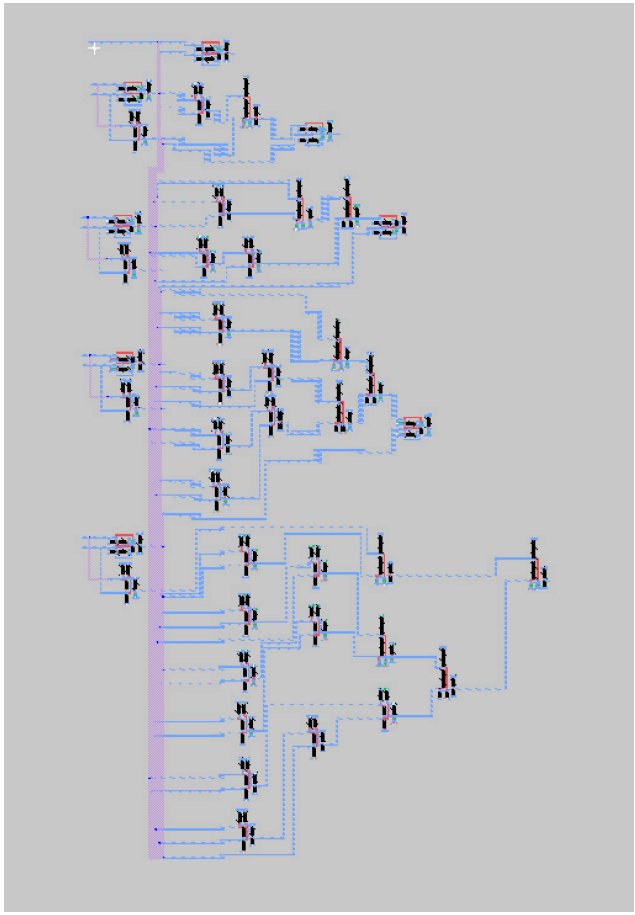
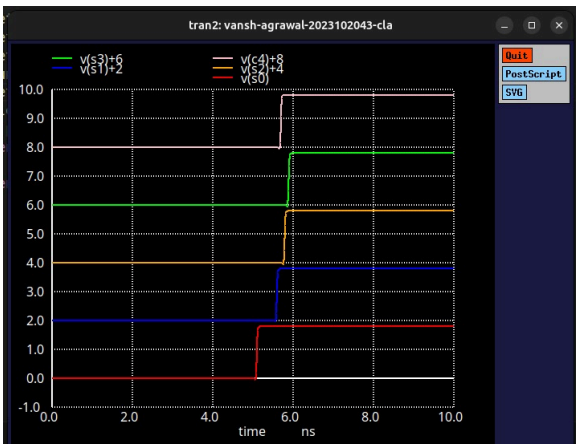


Fig. 19: Magic Layout for CLA Adder

We create the circuit shown in Fig. 4 using the MAGIC layout and conduct post-layout simulations. Using input a=1111 and b=1111 , Cin=1 at t=5ns ,we obtain the following results:

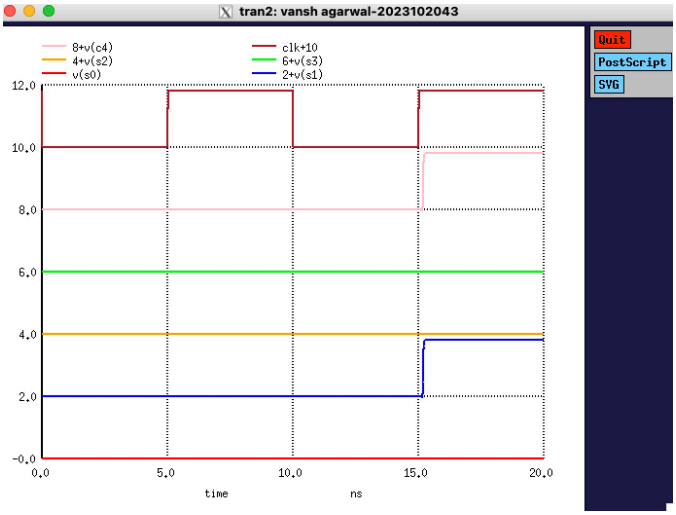


```
  Measurements for Transient Analysis

t_worst             =  7.921090e-10 targ=  5.879638e-09 trig=  5.087529e-09
```

7. SIMULATIONS OF FINAL CIRCUIT IN NGSPICE

We combine the CLA-Adder with D-FlipFlops to give input at positive edge and get output as second positive edge. We give inputs a= 1001 and b= 1001 at first positive edge .

The output is :



From our pre-layout simulations, the worst-case delay(t_{pdmax})is found to be 0.86 ns. This corresponds to a maximum clock frequency of 833.33 MHz or a minimum clock period of 1.2 ns. These results align well with our theoretical calculations.

$$t_{\text{clk\_min}} = t_{\text{setup}} + t_{\text{pd\_max}} + t_{\text{pcq\_max}}$$

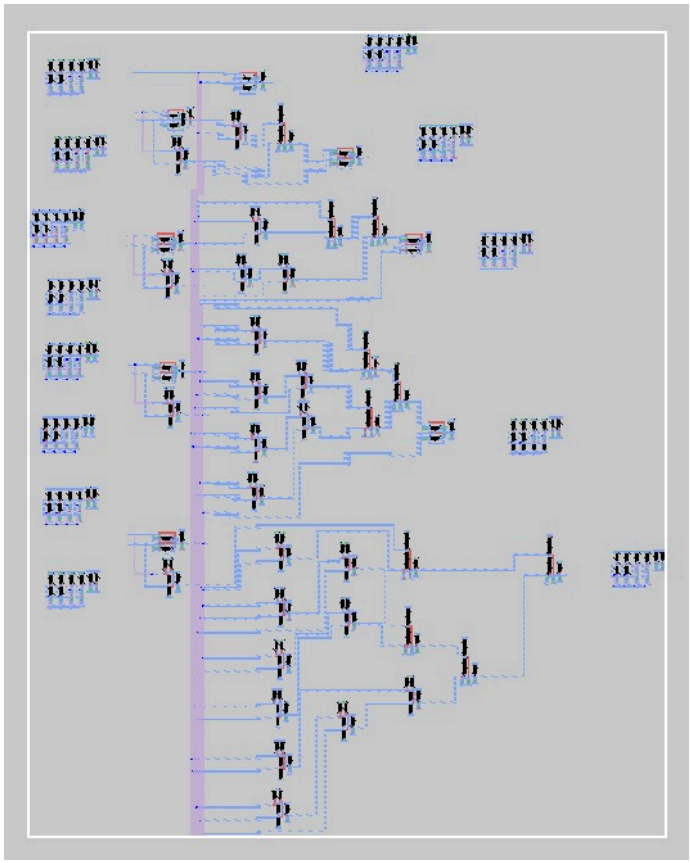
$$t_{\text{clk\_min}} = 0.11ns + 0.607ns + 0.219ns = 0.94ns$$

Hence at f_{max} =1063 MHZ, our circuit works perfectly.

8. FLOOR PLAN AND MAGIC LAYOUT FOR FINAL CIRCUIT

The complete floor plan has a horizontal pitch of 294.48 µm and a vertical pitch of 372.06 µm

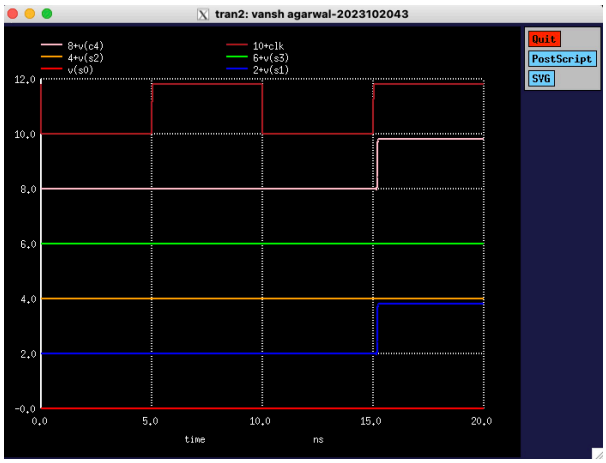
MAGIC Layout of the final circuit:



9. POST LAYOUT SIMULATIONS OF FINAL CIRCUIT

We again give a = 1001 and b = 1001 as input at first +ve edge.

The output is :



$$t_{\text{clk\_min}} = t_{\text{setup}} + t_{\text{pd\_max}} + t_{\text{pcq\_max}}$$

$$t_{\text{clk\_min}} = 0.11ns + 0.792ns + 0.329ns = 1.23ns$$

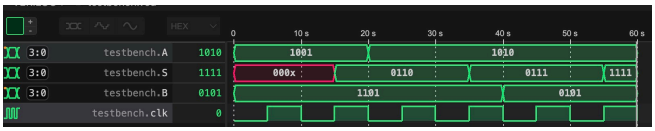
f_{max} =813 MHZ

10. VERILOG SIMULATIONS AND WAVEFORMS

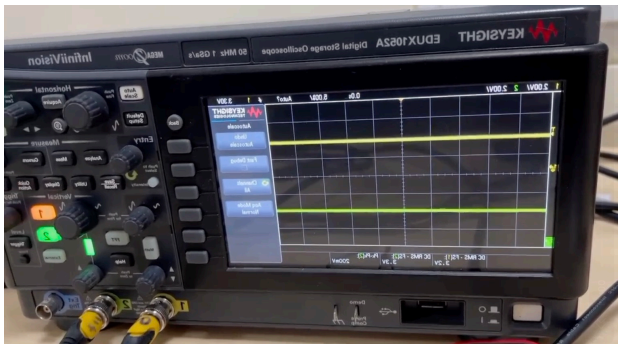
Now, we create the final CLA Circuit in verilog. We give three inputs, a = 1001, b = 1101 and a = 1010, b = 1101 and a=1010 , b=0101 .
We get the following output:

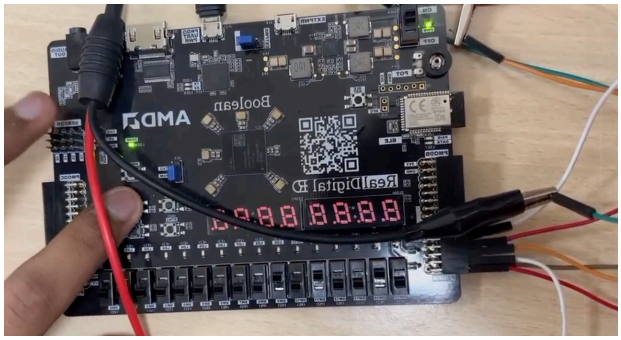
```
VCD info: dumpfile testbench.vcd opened for output.
Time=0 A=1001 B=1101 C0=0 S=xxxx C4=x
Time=15 A=1001 B=1101 C0=0 S=0110 C4=1
Time=20 A=1010 B=1101 C0=0 S=0110 C4=1
Time=35 A=1010 B=1101 C0=0 S=0111 C4=1
Time=40 A=1010 B=0101 C0=0 S=0111 C4=1
Time=55 A=1010 B=0101 C0=0 S=1111 C4=0
testbench.v:36: $finish called at 60 (1s)
[Done] exit with code=0 in 1.932 seconds
```

As we can see, when we give the input, we get the output in the next cycle. We get sum=10110 and sum=01111 and sum= 11110 after the next input has been given, that is, in the next +ve edge of the clock. In GTKWave, we get the following waveforms:



Hardware implementation :





LINK FOR THE VIDEOS

11. CONCLUSION

Thus, we successfully designed a CLA Adder that takes inputs at one positive clock edge and produces outputs at the subsequent positive clock edge. This ensures the CLA Adder has sufficient time to compute the sum bits.

12. REFERENCES

[1] CMOS VLSI Design (fourth edition) by Weste and Harris.
[2] Digital Logic and Computer Design by Morris Mano.
[3] Verilog HDL - Samir Palnitkar
[4] Internet - Google