VLSI Course Projrct
4-Bit CLA Adder

Snehil Sanjog
2023102051
snehil.sanjog@students.iiit.ac.in

Introduction

4 Bit Carry Look Ahead (CLA) Adder adds two 4 bit numbers and gives the final 4 bit sum along with the 5th bit carry. However unlike normal 4 bit Ripple Carry Adder which calculates each carry one at a time, CLA Adder calculates all the carry at the same time hence reducing the propagation delay.

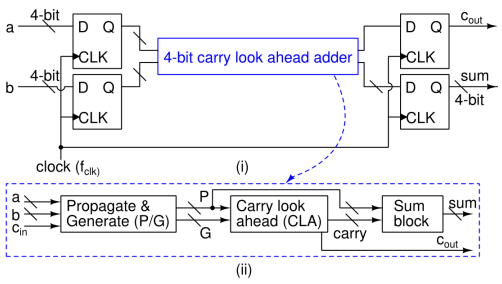


Fig. 1: Various circuit blocks for our Adder

I. Basic structure for the CLA Adder

In Fig. 1 part (ii), we need to create the three blocks. First we generate the p_i and g_i (Propagate and Generate) for CLA block.

$$p_i = a_i \oplus b_i$$

$$g_i = a_i \cdot b_i$$

Then we calculate all carries in CLA block by following logic,

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

From this, we can calculate all carries $c_i{}'s$ from the propagates and generates ($p_i{}'s$ and $g_i{}'s$).

$$c_1 = (p_0 \cdot c_0) + g_0$$

$$c_2 = (p_1 \cdot p_0 \cdot c_0) + (p_1 \cdot g_0) + g_1$$

$$c_3 = (p_2 \cdot p_1 \cdot p_0 \cdot c_0) + (p_2 \cdot p_1 \cdot g_0) + (p_2 \cdot g_1) + g_2$$

$$c_4 = (p_3{\cdot}p_2{\cdot}p_1{\cdot}p_0{\cdot}c_0){+}(p_3{\cdot}p_2{\cdot}p_1{\cdot}g_0){+}(p_3{\cdot}p_2{\cdot}g_1){+}(p_3{\cdot}g_2){+}g_3$$

Finally we can get the final sum by following logic,

$$S_i = p_i \oplus c_i$$

From this we can get circuit digram shown in Fig. 2. For multiple input AND & OR gates, we have used multiple gates insteas of one gate with multiple inputs.

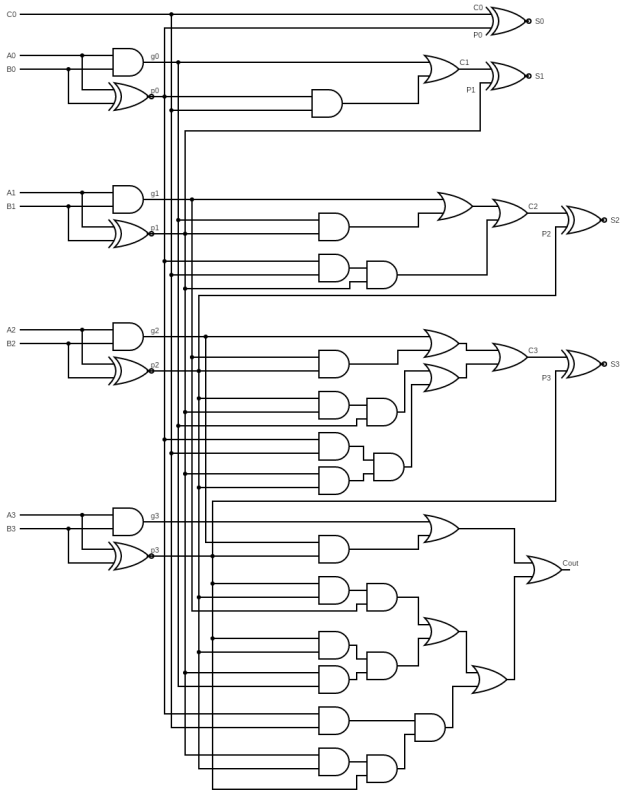


Fig. 2: Circuit for CLA Adder

II. Design topology and sizing

A. CLA Adder

For XOR, AND & OR gates, we use PTL(Pass Transistor Logic) Topology and CMOS Topology for Inverters. Their circuit diagram along with sizes are given in figures below.

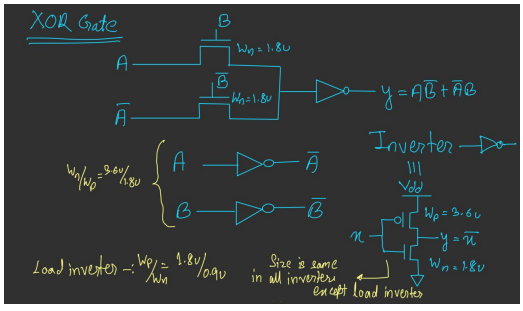


Fig. 3: Circuit for XOR gate and Inverter

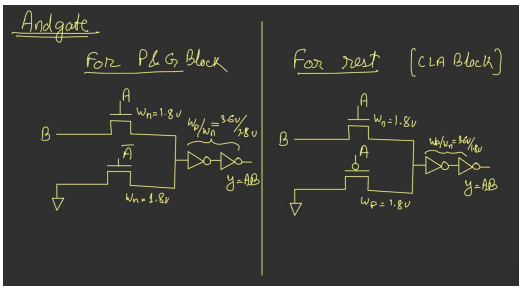


Fig. 4: Circuit for AND gate

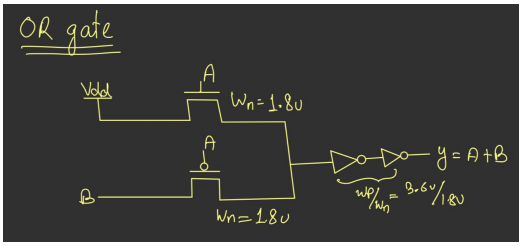


Fig. 5: Circuit for OR gate

The Sizes are same for all the gates in CLA Adder.

B. D-Flipflop

For D flipflop, we use TSPC (True Single Phase Clock) topology. Given below is the circuit for flipflop along with its size.

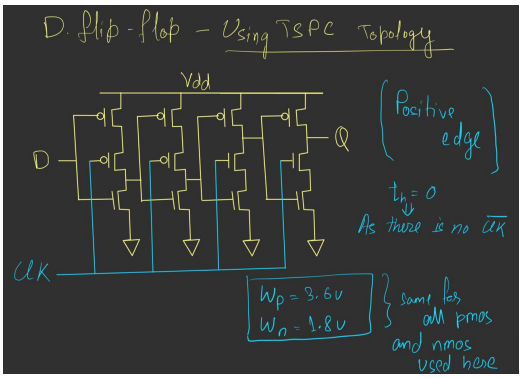


Fig. 6: Circuit for D FlipFlop with its sizes

III. Pre-Layout Simulations in Ngspice

A. CLA Adder

For CLA Adder, we take input as $a = 1001$ and $b = 1101$ at t=5ns. The output is as follows.

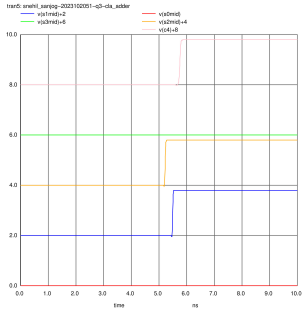


Fig. 7: Final Output for given inputs

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

$$1001 + 1101 = 10110$$

Hence is working properly. As for Propagation delays, we give $a = 1111$, $b = 1111$ as well as $c_{in} = 1$ which is usually 0 for nornal adder applications. From this we get output $s = 11111$. Here, we see the delays of all sum and final carry unit:

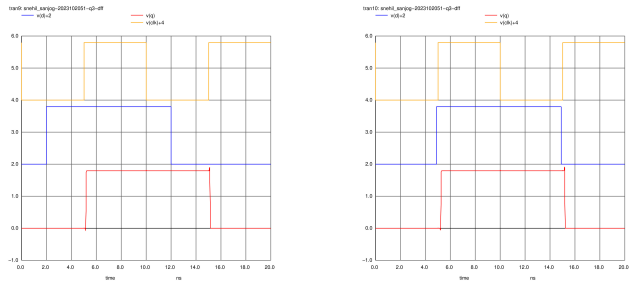
```
tpd_s0          =  4.499356e-11 targ=  5.049994e-09 trig=  5.005000e-09
tpd_s1          =  4.884426e-10 targ=  5.493443e-09 trig=  5.005000e-09
tpd_s2          =  8.601897e-10 targ=  5.865190e-09 trig=  5.005000e-09
tpd_s3          =  8.515328e-10 targ=  5.856533e-09 trig=  5.005000e-09
tpd_carry       =  7.508256e-10 targ=  5.755826e-09 trig=  5.005000e-09
```

Fig. 8: t_{pd} for all S_i and C_{out}

From this we get $t_{pd_{min}} = 44.9ps$ and $t_{pd_{max}} = 0.86ns$

B. D-Flipflop

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



(a) Simulation for min t_{pcq} (b) Simulation for max t_{pcq}

Fig. 9: Ngspice Simulation for D-FlipFlop

As we can see, Q is updated to D at +ve edge only. Hence its working properly.

IV. Setup time, hold time and propagation delay from clk to Q in FlipFlop

We have used TSPC topology for our D-FlipFlop. Since there is no use of $\overline{clk}$, we have zero hold time ($t_{hold} = 0$).

For setup time, we did trial and error and found it to be $t_{setup} = 0.11ns$. Any input given less than t_{setup} before next +ve edge, either input is not registered or output is being corrupted.

For propagation delay from clk to Q (t_{pcq}), we can see the minimum and maximum t_{pcq} in Fig. 9. The values are as follows:

```
tpcq_r          =  1.603094e-10 targ=  5.175309e-09 trig=  5.015000e-09
tpcq_f          =  1.133761e-10 targ=  1.512838e-08 trig=  1.501500e-08
tpcq            =  1.36843e-10
```

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

```
tpcq_r          =  2.335070e-10 targ=  5.248507e-09 trig=  5.015000e-09
tpcq_f          =  1.819591e-10 targ=  1.519696e-08 trig=  1.501500e-08
tpcq            =  2.07733e-10
```

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

Fig. 10: t_{pcq} for D-Flipflop

We get $t_{pcq_{min}} = 0.136ns$ and $t_{pcq_{max}} = 0.207ns$.

V. Stick Diagrams for all unique gates and flipflops

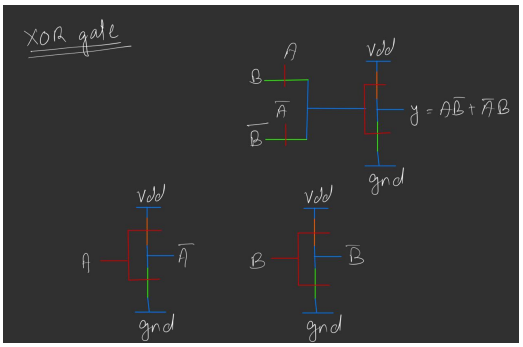


Fig. 11: Stick Diagram for XOR and Inverter

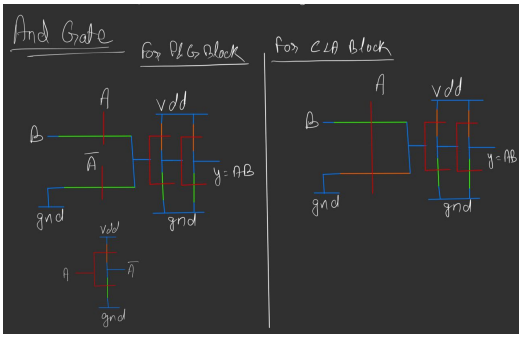


Fig. 12: Stick Diagram for AND

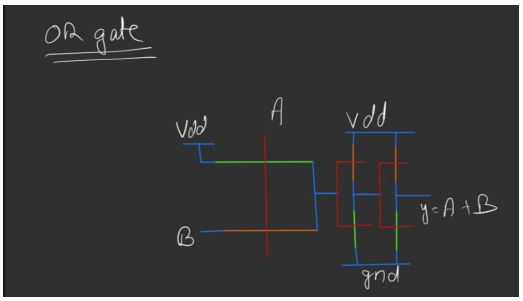


Fig. 13: Stick Diagram for OR

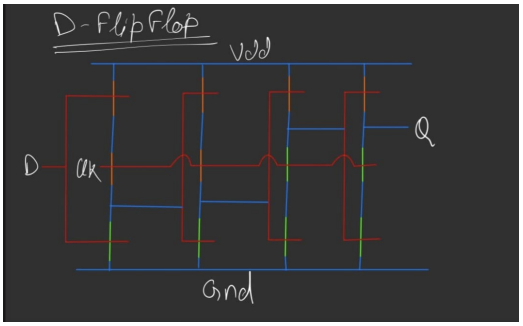


Fig. 14: Stick Diagram for D-FlipFlop

VI. Post Layout Simulations

A. D-FlipFlop

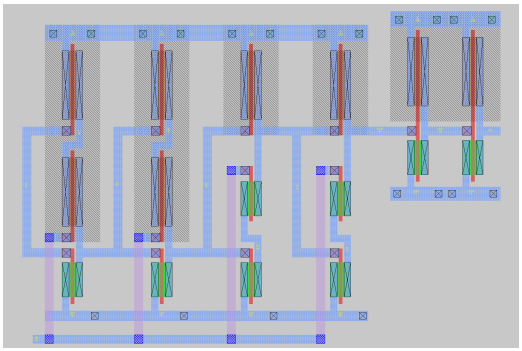
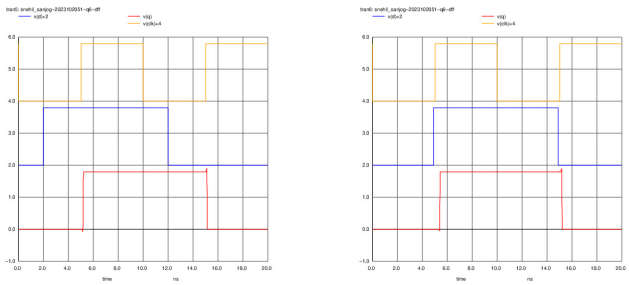


Fig. 15: Magic Layout for D-FlipFlop

Here we make the circuit given in Fig. 6 in MAGIC Layout and perform post-Layout simulations.



(a) Simulation for min t_{pcq} (b) Simulation for max t_{pcq}

Fig. 16: Post-Layout Simulations for D-FlipFlop

Here we can see that we get similar output as pre-layout simulations. However, we have slightly different t_{pcq}.

```
tpcq_r           =  1.690262e-10 targ=  5.184026e-09 trig=  5.015000e-09
tpcq_f           =  1.188576e-10 targ=  1.513386e-08 trig=  1.501500e-08
tpcq             =  1.43942e-10
```

(a) $t_{pcq_{min}}$ for Post-Layout

```
tpcq_r           =  3.889382e-10 targ=  5.403938e-09 trig=  5.015000e-09
tpcq_f           =  1.873369e-10 targ=  1.520234e-08 trig=  1.501500e-08
tpcq             =  2.88138e-10
```

(b) $t_{pcq_{max}}$ for Post-Layout

Fig. 17: t_{pcq} for D-Flipflop

	Pre-Layout	Post-Layout
$t_{pcq_{min}}$	0.136ns	0.144ns
$t_{pcq_{max}}$	0.207ns	0.288ns

TABLE I: Comparision between pre and post layout simulations

B. CLA-Adder

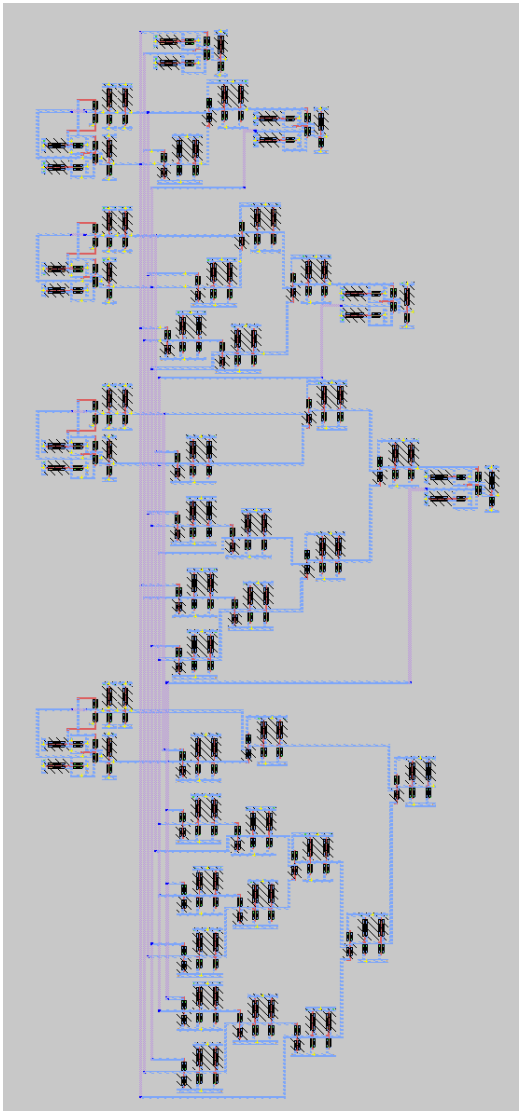


Fig. 18: Magic Layout for CLA-Adder

Here we make the circuit given in Fig. 2 in MAGIC Layout and perform post-Layout simulations. We again perform simulation with input $a = 1001$ and $b = 1101$ at t=5ns and get the following result:

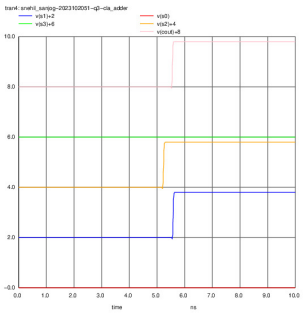


Fig. 19: Final output for post-layout

As we can see, with some delay, we get the final output $s = 10110$ which is same as pre-layout simulations. By giving a and b as 1111 as well as $c_{in} = 1$ for $s = 1111$, we get

```
tpd_s0              =  7.954170e-11 targ=  5.084542e-09 trig=  5.005000e-09
tpd_s1              =  5.824552e-10 targ=  5.587455e-09 trig=  5.005000e-09
tpd_s2              =  6.905608e-10 targ=  5.695561e-09 trig=  5.005000e-09
tpd_s3              =  6.639990e-10 targ=  5.668999e-09 trig=  5.005000e-09
tpd_carry           =  5.585690e-10 targ=  5.563569e-09 trig=  5.005000e-09
```

Fig. 20: Post-layout delays for all $s_i's$ and final carry

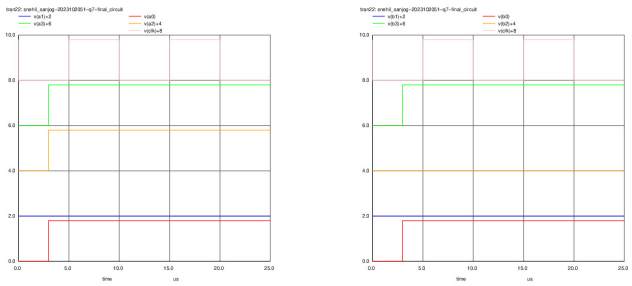
Now comparing delays with pre-layout simulations, we get:

	Pre-Layout	Post-Layout
$t_{pd_{min}}$	44.9ps	79.54ps
$t_{pd_{max}}$	0.86ns	0.690ns

TABLE II: Comparision between pre and post layout simulations

VII. 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 thprevious inputs $a = 1101$ and $b = 1001$ at first positive edge as shown:



(a) $a = 1101$ (b) $b = 1001$

Fig. 21: Inputs for our final adder

We get the following output:

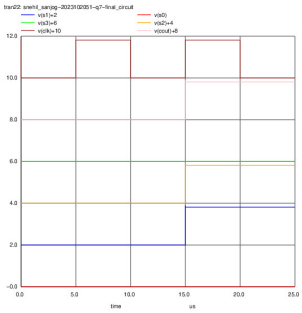


Fig. 22: Final output at second +ve edge

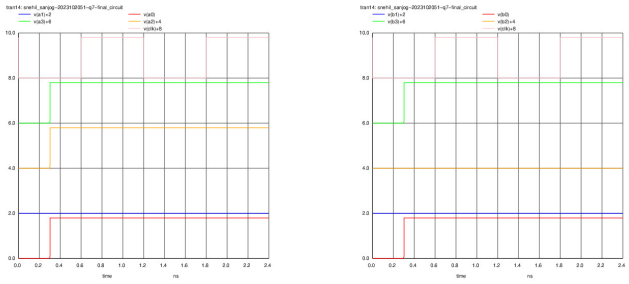
As we can see, we get final output as 10110 at the second +ve edge of the clock. This means that our final circuit is working perfectly fine.

Our worst case delay as seen in pre-layout simulations is $t_{pd_{max}} = 0.86ns$ and for maximum clock frequency at which the circuit works perfectly fine is 833.33MHz or minimum time period of 1.2ns. We can verify this with our theoritical calculations that is

$$t_{clk_{min}} = t_{setup} + t_{pd_{max}} + t_{pcq_{max}}$$

$$t_{clk_{min}} = 0.11ns + 0.86ns + 0.207ns = 1.177ns$$

Now t=1.2ns is very close to 1.177ns. Hence at $f_{max} = 833.33MHZ$, our circuit works perfectly. We can see this in simulations as well.



(a) $a = 1101$ (b) $b = 1001$

Fig. 23: Inputs for our final adder at maximum clock frequency

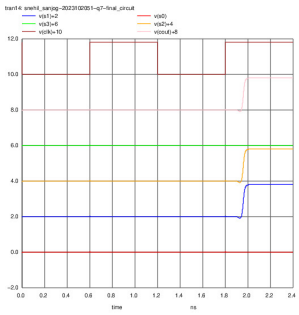


Fig. 24: Final output at second +ve edge for maximum clock frequency

With less than 0.2ns as t_{pcq}, we can see the correct output at second +ve edge.

VIII. Floor Plan and Magic Layout for Final Circuit

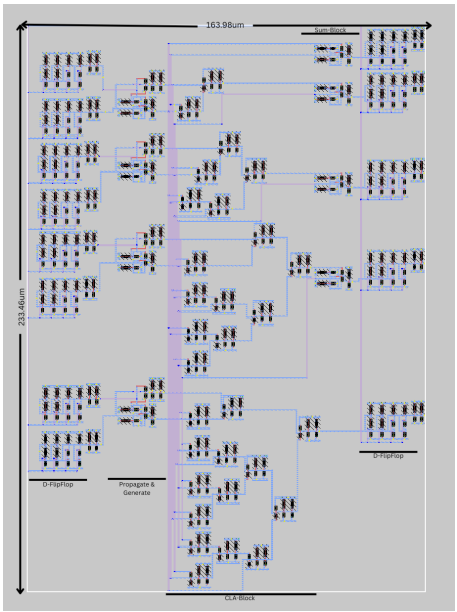
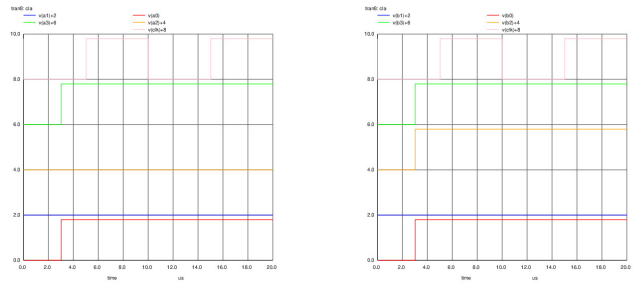


Fig. 25: Complete Floor Plan for Final Circuit

This is the complete floor plan of final circuit with Horizontal Pitch $163.98\mu m$, Vertical Pitch $233.46\mu m$ and total area $38232.77\mu m^2$.

IX. Post Layout Simulations of Final Circuit

We again give $a = 1001$ and $b = 1101$ as input at first +ve edge.



(a) $a = 1001$ (b) $b = 1101$

Fig. 26: Inputs for our final circuit

We get the output as follows:

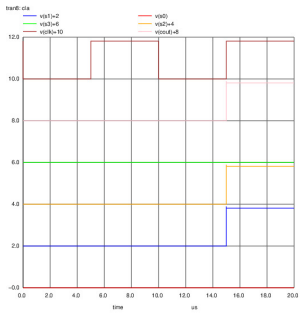
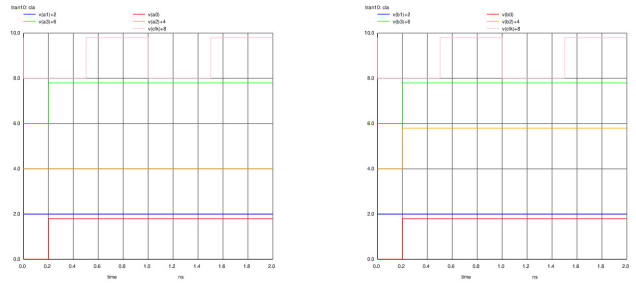


Fig. 27: Final output at second +ve edge

As we can see, we get output at second +ve edge as s=10110. So it matches with its pre-layout simulation. Now for maximum clock frequency, we need to find minimum clock time period. We have $t_{pd_{max}} = 0.690ns$ for CLA-Adder, $t_{pcq_{max}} = 0.288ns$ and $t_{setup} = 0.11ns$

$$t_{clk_{min}} = 0.11 + 0.690ns + 0.288ns = 1.088ns$$



(a) $a = 1001$ (b) $b = 1101$

Fig. 28: Inputs for our final circuit at maximum clock frequency

We get the output as follows:

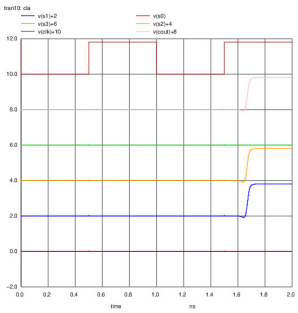


Fig. 29: Final output at maximum clock frequency

We get proper output at $t_{clk} = 1ns$ which is nearly as same as our theoritical minimum value. So, $f_{max} = 1000MHz$ is the maximum clock frequency.

	Pre-Layout	Post-Layout
$t_{pd_{max}}$	0.86ns	0.690ns
$t_{clk_{min}}$	1.2ns	1ns
$f_{clk_{max}}$	833.33MHz	1000MHz

TABLE III: Comparision between pre and post layout simulations

X. Verilog Simulations and Waveforms

Now, we create the final CLA Circuit in verilog. We give two inputs, a = 1001, b = 1101 and a = 1010, b = 01101. We get the following output:

```
[Running] tb.v
VCD info: dumpfile tb.vcd opened for output.
a=1001 b=1101    cout=x s=xxxx
a=1010 b=0101    cout=1 s=0110
a=0000 b=0000    cout=0 s=1111
a=0000 b=0000    cout=0 s=0000
tb.v:27: $finish called at 43 (1s)
[Done] exit with code=0 in 0.038 seconds
```

Fig. 30: Verilog Output for the above inputs

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

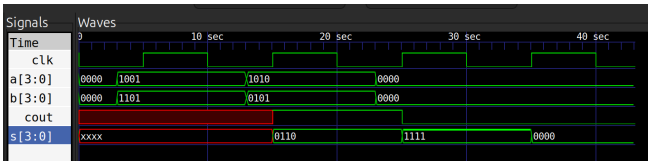


Fig. 31: GTK waveforms for the given inputs

XI. FPGA and Hardware Simulations

Now, we give inputs in FPGA board and get the output as shown:

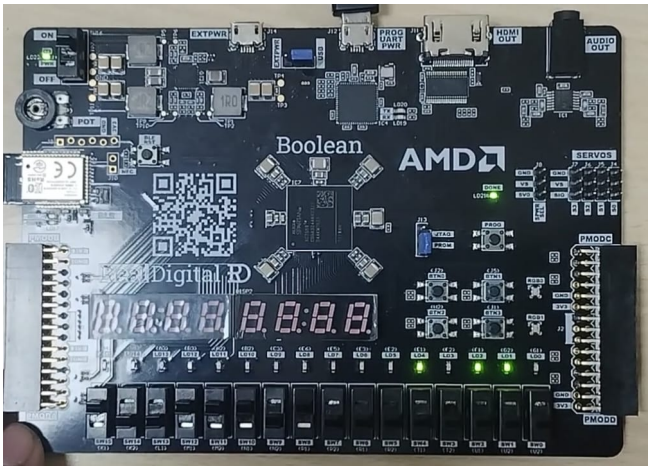


Fig. 32: Output on FPGA Board

For a complete video, here is the link: https://drive.google.com/file/d/1b7-uLKQ62djXwmcypvaVRPbVqIZsADK0/view?usp=drivesdk
We then give the input a=1001 and b=1101 and finally get the output S = 10110 in the oscilloscope:

Fig. 33: Output in Oscilloscope

Individual Oscilloscope Readings:

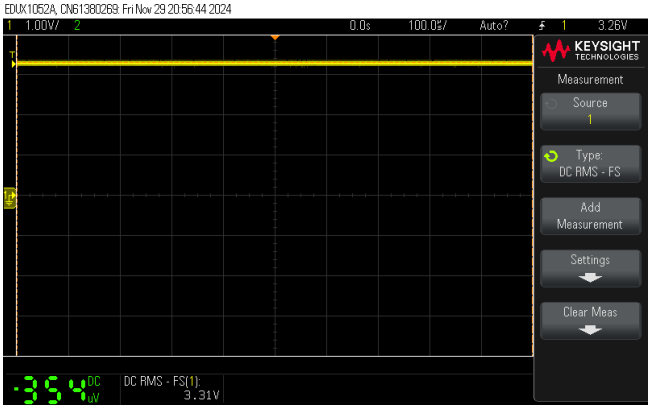


Fig. 34: c_{out}

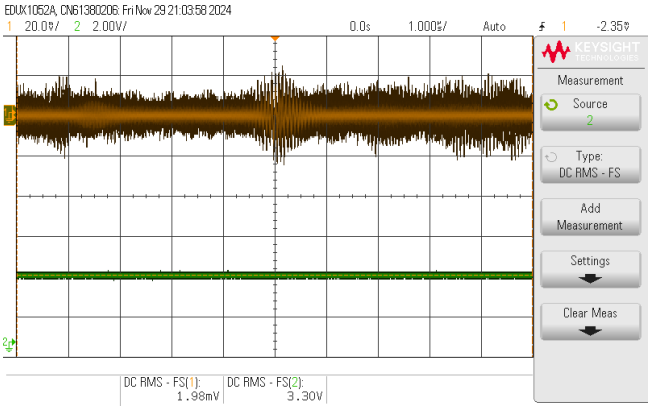


Fig. 35: s_3 and s_2

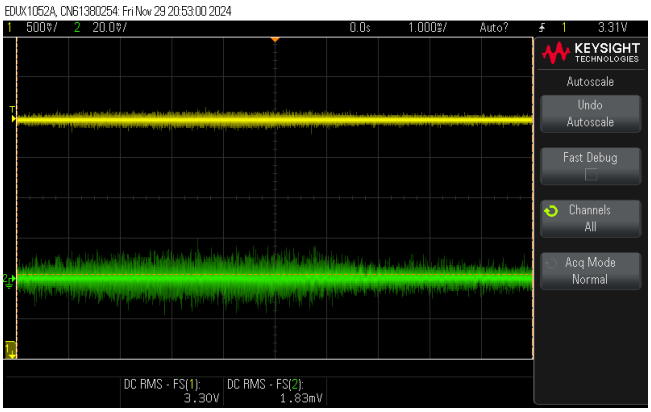


Fig. 36: s_1 and s_0

As we can see, we are getting 3.3V for HIGH (bit 1) and some noise (in mV) for LOW (bit 0). Here we also are getting the sum as 10110.

XII. Conclusion

So, like this, we are able to design a CLA Adder which takes input at one +ve edge and gives output at next +ve edge clock. This makes sure that the CLA adder has enough time to calculate the sum bits.

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