VLSI PROJECT(4 BIT CLA Adder)

Manikya Pant
2023112024

I. Introduction

This is the project report for the VLSI project of a 4bit adder circuit, it includes the simulation , layout and FPGA part of it

II. Design of the circuit

To implement the 4bit CLA adder we used the standard approach used to make a 4-bit CLA adder, using the Carry propagate and Generate method.
The carry equations for a Carry-Lookahead Adder (CLA) are as follows:

$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 C_1$$

Substituting C_1:

$$C_2 = G_1 + P_1(G_0 + P_0 C_0)$$

$$C_3 = G_2 + P_2 C_2$$

Substituting C_2:

$$C_3 = G_2 + P_2[G_1 + P_1(G_0 + P_0 C_0)]$$

$$C_4 = G_3 + P_3 C_3$$

Substituting C_3:

$$C_4 = G_3 + P_3[G_2 + P_2[G_1 + P_1(G_0 + P_0 C_0)]]$$

This recursive pattern can be generalized for any n-bit CLA adder. To make the D flip flops we used the tspc style, so that our flip flops are fast.
We have used static CMOS implementation for most of the gates, and for some gates we used pass transistor bases styling.

In order to reduce the delay, we have used 2 input gates in place of multiple input gates in all the places wherever required.

We have used the standard CMOS static sizing,so that the delay is minimized and the other unwanted effects don't bother

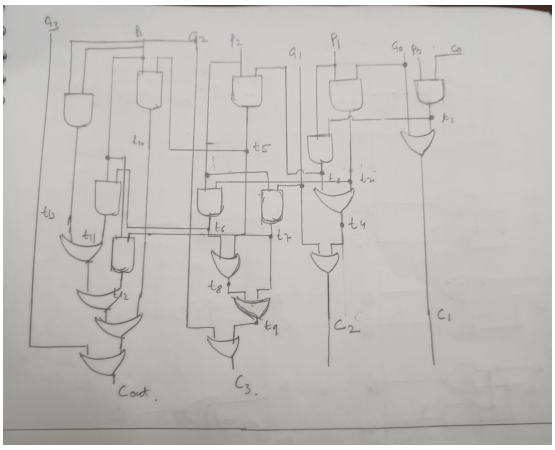

Fig. 1. The carry generator part of the circuit

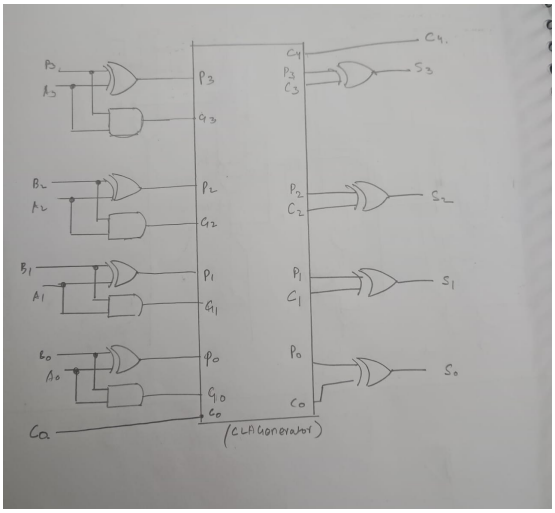

Fig. 2. The cla adder without the flip flops

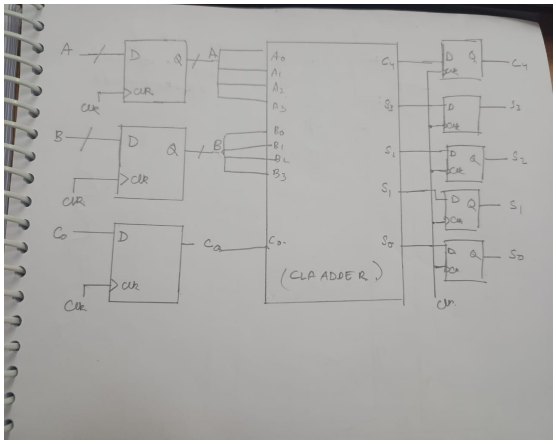

Fig. 3. General layout of the circuit

III. Design topolgy

We have defined and used $W_n = 10 * LAMBDA$ and $W_p = 2 * W_n$, where $LAMBDA = 0.09\mu m$.

The sizing of the following gates are now described in terms of W_n and W_p

The length of each MOSFET used is $2 * LAMBDA$

A. Inverter

The inverter we have used all the place has the $\frac{W_p}{W_n}$ ratio=2 and W_n and W_p are the same as described above.

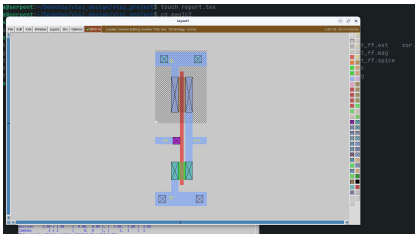

Fig. 4. Topology of inverter

B. NAND gate

The nand gate is also implemented in static CMOS style and the sizing of the pull up network is $2 * W_n$ and the sizing of the pull down network is $2 * W_n$.

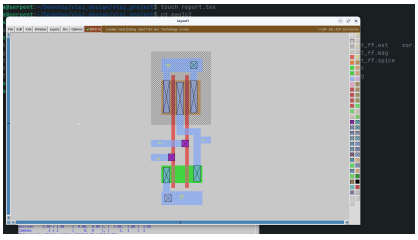

Fig. 5. Topology of nand

C. XOR gate

Xor gate is implemented in PTL style. The sizing of the PMOS and NMOS are the same as the ones mentioned above

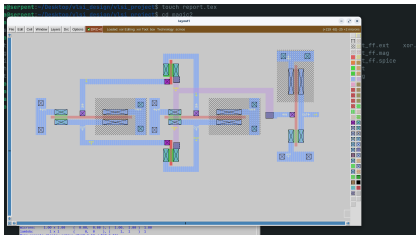

Fig. 6. Topology of xor

D. AND gate

And gate is made by connecting the above inverter to the output of NAND gate.

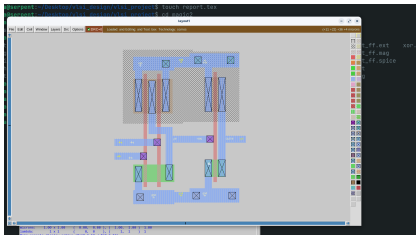

Fig. 7. Topology of and

E. NOR gate

NOR gate made using static CMOS implementation. The pull up network has a size $4 * Wn$ and the pull down network has a size of W_n

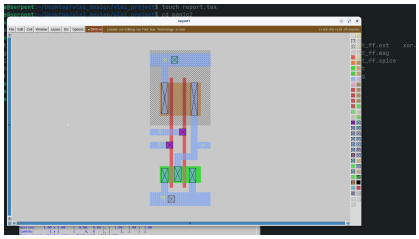

Fig. 8. Topology of NOR

F. OR gate

Or gate is made by connecting the inverter to the output of the OR gate

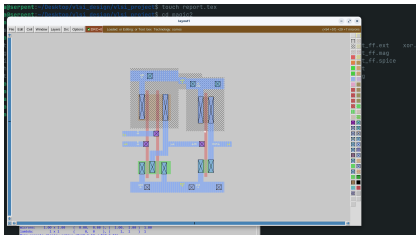

Fig. 9. Topology of or gate

G. D flip flop

tspc logic style is used to make all the flip flops and the NMOS used has the size W_n and the PMOS has size W_p

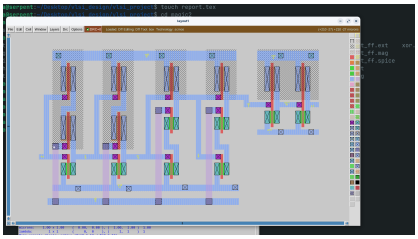

Fig. 10. Topology of D Flip flop in tspc style

IV. Delays of D flip flop

The clock to Q delay of the D flip flop was found out to be 0.2ns

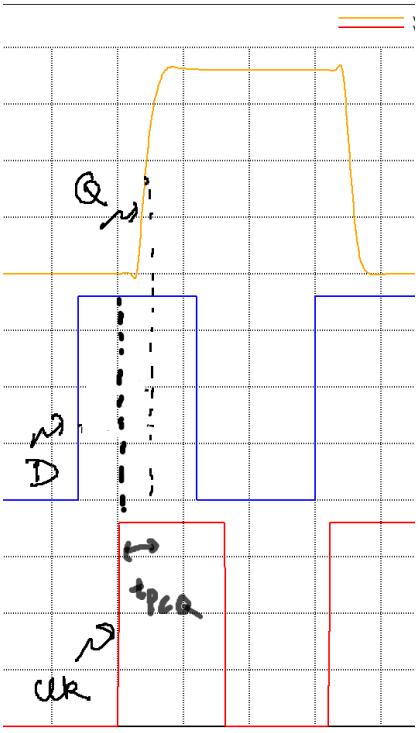

Fig. 11. The clock to Q delay of the D flip flop, in prelayout

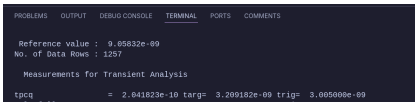

Fig. 12. The clock to Q delay of the D flip flop, in prelayout

$T_{setup} = 0.17ns$, which is nearly same as Tcpq delay of tspc flip flop.

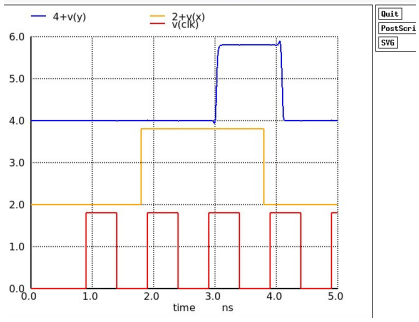

Fig. 13. Setup time of the flip flop

T_{hold} of tspc=0ns.

V. Stick diagram

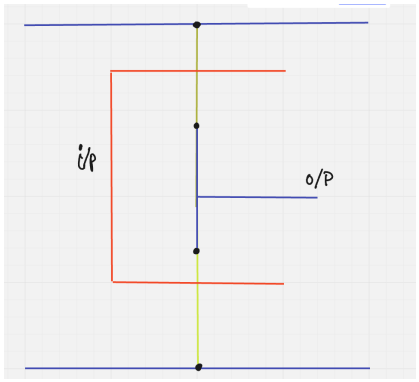

Fig. 14. stick diagram of inverter

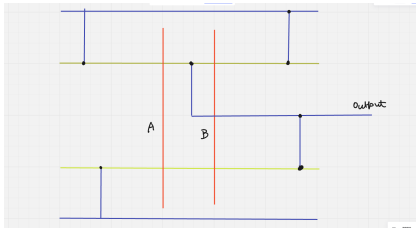

Fig. 15. stick diagram of nand

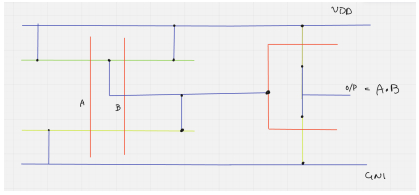

Fig. 16. stick diagram of and

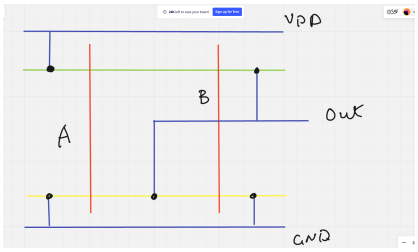

Fig. 17. stick diagram of nor

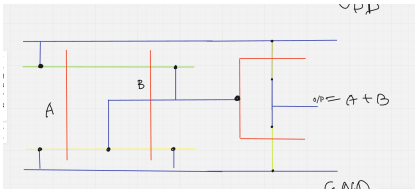

Fig. 18. stick diagram of or

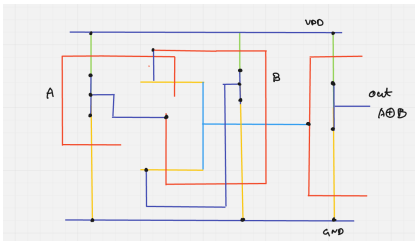

Fig. 19. stick diagram of xor

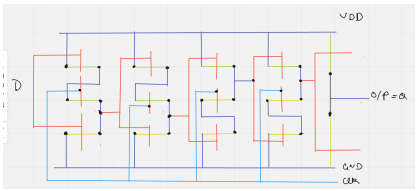

Fig. 20. stick diagram of tspc dff

VI. Delay of the circuit

The minimum time period of the clock is less than or equal to $tpcq + tcla + tsetup$.

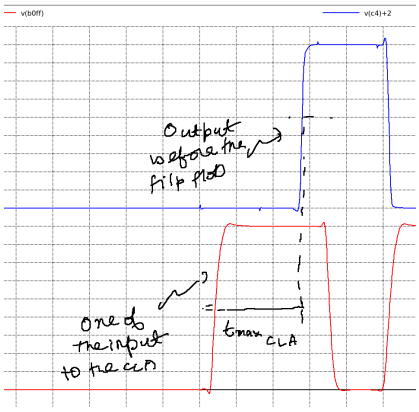

Fig. 21. delay of the cla for the worst case

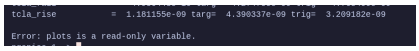

Fig. 22. delay of the cla for the worst case

The delay of the cla adder for the worst case input is 1.18ns.

The minimum time period of our circuit prelayout was found to be 1.5ns, or a clock frequency of 666MHz. The minimum time period in which our circuit works correctly comes out to be 1.6ns.

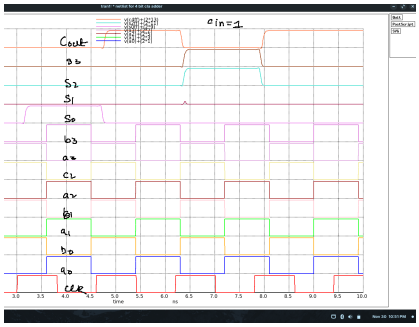

Fig. 23. Output of the circuit for the worst case

VII. Floor plan of the circuit

The circuit is cascade of the various blocks, involved in the circuit, input flip flops, adder module and then output flip flops.

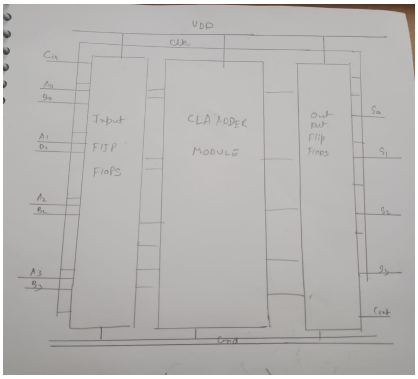

Fig. 24. Floor plan of the circuit

The final magic layout of the whole circuit.

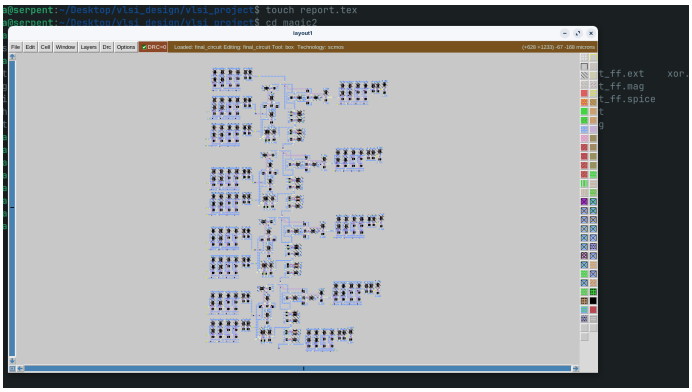

Fig. 25. Layout of full circuit

VIII. Post layout simulation

The post layout simulation was performed and the output was correctly observed, although the delay in the output or the time period of the clock was increased.

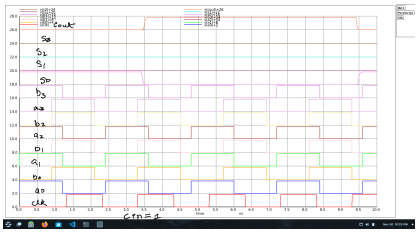

Fig. 26. Post layout output of the circuit

IX. Post layout delay

The post layout delay of the circuit has increase as compared to the prelayout delays. The increase in delay is mostly accounted due to the increase in parasitics.

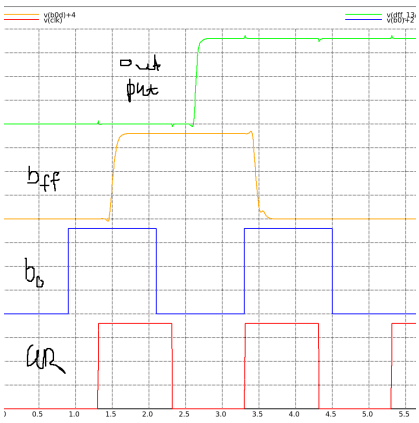

Fig. 27. Post layout delay of the circuit

The above figure shows the tcla of the adder after layout.

Fig. 28. Post layout delay of the circuit

Quantity	Pre-Layout	Post-Layout
Tpcq	0.204ns	0.185ns
Tcla(max)	1.08ns	1.13ns
Tclk(min)	1.5ns	1.6ns

TABLE I
COMPARISON OF PRE-LAYOUT AND POST-LAYOUT RESULTS

X. Verilog implementaion

The following are the outputs of the structural implementations of the verilog for the cla circuit.

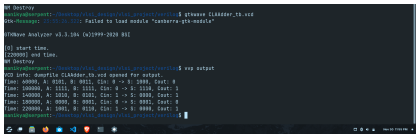

Fig. 29. vvp output for some inputs

The gtkwave plots for the same set of inputs.

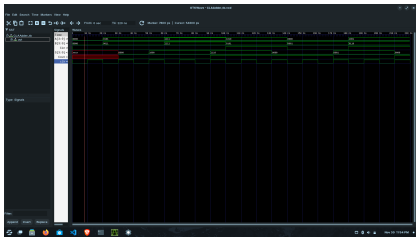

Fig. 30. vvp output for some inputs

XI. FPGA IMPLEMENTAION

The same verilog code was synthesized and run on the FPGA. The following are the outputs observed for the inputs, and the videos are in the drive link attached. Click here
The given input was A=7(0111) and B=15(1111) and the output was observes as 22(10110) with cout=1 and s=0110

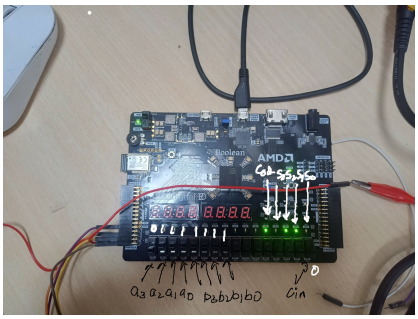

Fig. 31. Output of the fpga on the led pins

XII. FPGA IMPLEMENTAION ON OSCILLOSCOPE

The link for video of implementation: Click here
For the FPGA implementation on oscilloscope we gave the cin=1 and A=1111 and B=1111 and got the desired result of S=1111 and cout=1.

Fig. 32. all the oscilloscopes

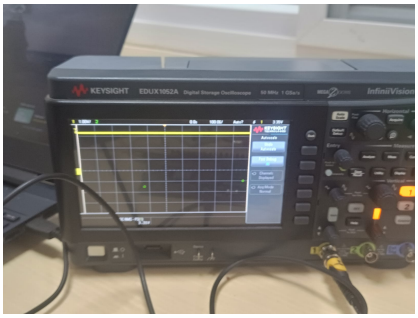

Fig. 33. cout

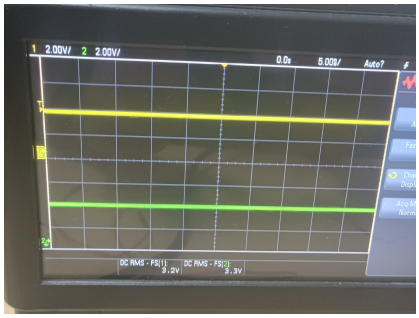

Fig. 34. S3 and S2 bits

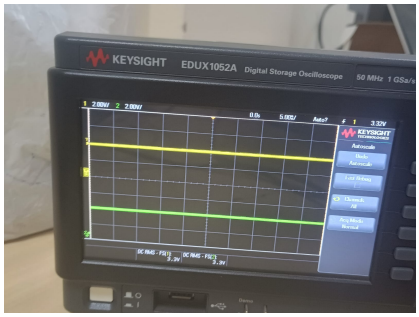

Fig. 35. S1 and S0 bits

REFERENCES

[1] M.Morris Mano, Digital Design
[2] Jan M Rabaey, Digital Integrated Circuits
[3] Class notes