VLSI Course Project

M.P. Samartha
Roll no : 2023102038
m.samartha@students.iiit.ac.in

International Institute of Information Technology, Hyderabad

Contents

I Proposed structure for the CLA **2**

II Design Details **3**

IV Timing Constraints of FF **3**

V Stick Diagrams **4**

VI Magic layout of building blocks **6**

VIIComplete circuit integration on NG-Spice **7**

VIIIFloorplan for Magic layout **8**

IX Complete circuit Magic layout **9**

X Reporting Delay and Max. frequency of CLA **10**

XI Icarus Verilog HDL **10**

XIIFPGA **10**

XIIIAcknowledgment **11**

References **11**

***Abstract*—We design a Carry Lookahead Adder (CLA) from scratch to the final hardware implementation. The CLA is a massive improvement over the conventional Ripple Carry Adder (RCA), because this calculate the carry in for each bit using just the P_i and G_i signals rather than relying on the propagated carry through multiple stages. In chip design where 32, 64 and even 128 bit adders are required, CLA makes a major difference. Here we first discuss the approach, simulate the CLA in NG-Spice, layout the complete CLA in Magic software, write Icarus Verilog HDL. The for the hardware implementation, we burn the verilog onto the Field Programmable Gate Array (FPGA), and verify the functionality on the oscilloscope. The aim of the project is to optimise the delay of the CLA, decrease the power consumption, and also optimise the area of the circuit on the layout.**

I. Proposed structure for the CLA

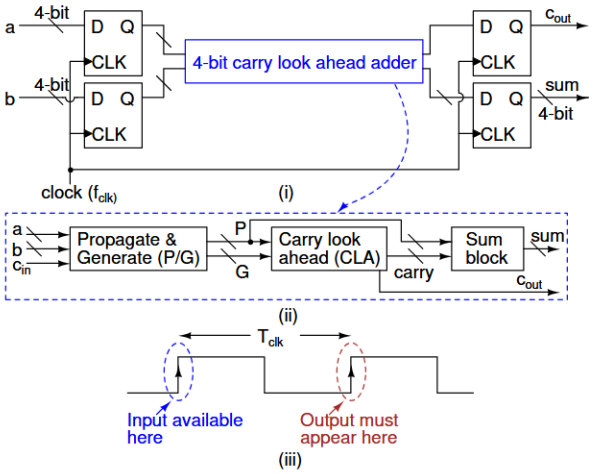

Fig. 1: Given Structure of the CLA

Above is the given high level block diagram of the CLA. Here we have 9 Flipflops, being used to load the inputs and outputs at the clock edge. The CLA itself is containing 3 sub-structures viz. P/G block, CLA (carry generate) block and the Sum block. The question also asks use to generate the output at the clock edge following that where input is loaded onto the CLA.

The following are the conventional equations for generating carry-in for each bit as a function of P_i, G_i and C_0 and finally final Carry out for the next stage of the adders.

$$C_1 = G_0 + P_0C_0$$
$$C_2 = G_1 + P_1G_0 + P_1P_0C_0$$
$$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$$
$$G = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0$$
$$P = P_3P_2P_1P_0$$
$$C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$$

We see from the above function that, to realize these functions in the circuit, we need complex gates such as 5 input AND and OR gates. This would increase the delay in the circuit as delay is proportional to the logical efforts of the gates and more the inputs, higher is the logical effort. To optimize this, we equivalently write the above equations in the following form. We clearly see the improvement in the implementation. This is because now we no longer need any gate of more than 2 inputs.

$$C_1 = \overline{G'_0\ (P'_0 + \overline{C_0})} \tag{1}$$
$$C_2 = \overline{G'_1\ (P'_1 + G'_0)} + \overline{(P'_1 + P'_0C_0)} \tag{2}$$
$$C_3 = \overline{G'_2\,(P'_2 + G'_1)} + \left(\overline{P'_2 + P'_1}\,\overline{G'_0\ (P'_0 + C'_0)}\right) \tag{3}$$
$$G'_{\text{out}} = \overline{\overline{G'_3\ (P'_3 + G'_2)} + \overline{(P'_3 + P'_2)}\,\overline{G'_1\ (P'_1 + G'_0)}} \tag{4}$$
$$P'_{\text{out}} = \overline{\overline{P'_3 + P'_2}\,\overline{P'_1 + P'_0}} \tag{5}$$
$$C_{out} = \overline{G'_{\text{out}}\,(P'_{\text{out}} + C'_0)} \tag{6}$$

Implementing the above equations, we get the following circuit diagram, 2. Here A_i, B_i, C_0 are the inputs and S_i are the outputs. Here $C_{out}$is calculate from the G'_{out} and $P'out$ as seen in the equations above. **We see that because of the nature of the modified equations, we have effectively replaced an AND gate with a NAND gate and an XOR gate with a NOR gate.** Thus we have improved over gate complexity and hence the layout area is minimized. This is the advantage that this implementation provides.

As seen in the circuit below, the P/G block is the NAND and NOR gates used on the inputs which produce G'_i and P'_i. These are then applied to the CLA block whose equations are as above. The CLA block then produces the carry in for all the 4 bits. Then using these carry in, the final sum is calculated in the SUM block. This block consists of 8 XOR gates, 2 for each sum output.

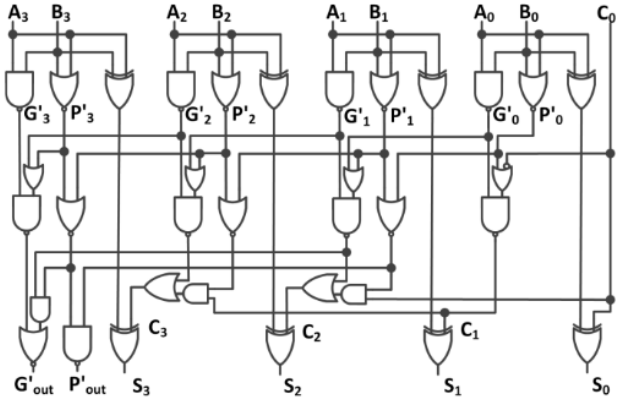

Fig. 2: Circuit diagram of CLA

II. Design Details

Here we have used mixed topology involving both CMOS and PTL for implementing the gates as seen in 2. For implementing D-Flipflop we use a TSPC flipflop because of its superior properties like 0 hold time which will be explained later in appropriate sections. The following is the circuit used for TSPC flipflop. Here for sizing, we define widths of the minimum sized inverter as follows : $\mathbf{W_n = 10\lambda, W_p = 2 * W_n = 20\lambda}$. We then try to match every gate to this minimum sized inverter resistance, as we usually do in the gates design to have equal rise and fall times. Here 10λ is chosen as it provides a reasonable driving capability and consumes not too much area also in the layout.

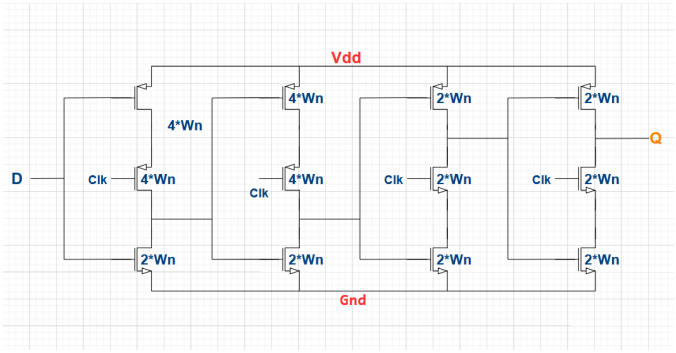

Fig. 3: Circuit Diagram of Positive triggered TSPC Flipflop

The basic logic gates (NOT, NAND, NOR, AND, OR) are implemented as usual in the CMOS topology, whereas PTL topology is used for implementing XOR gate due to its benefits as will be discussed later. Following is the sizing of the basic gates.

Gates	W_n	W_p
NOT	10λ	20λ
NAND	20λ	20λ
NOR	10λ	40λ
AND	20λ	20λ
OR	10λ	40λ

Following is the implementation of the XOR gate. Here we see that this is better the conventional XOR gate as this uses just 6 transistors compared to the 8 transistors in the CMOS topology.

IV. Timing Constraints of FF

Now we find the setup time, hold time and t_{CQ} of the TSPC flipflop. First we verify the functionality of the TSPC FF. Above is the plot of clk, input and output signal of the TSPC FF. Appropriate rise/fall times for the clk and pulse widths are chosen. From the plot, we can clearly see the action of the flipflop. The input is loaded onto the FF at the positive rising edge.

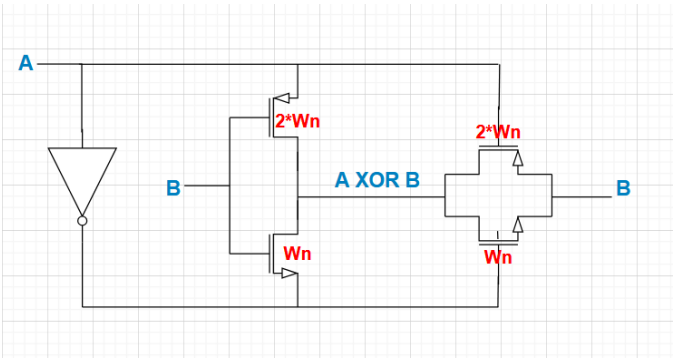

Fig. 4: Implementation of XOR gate

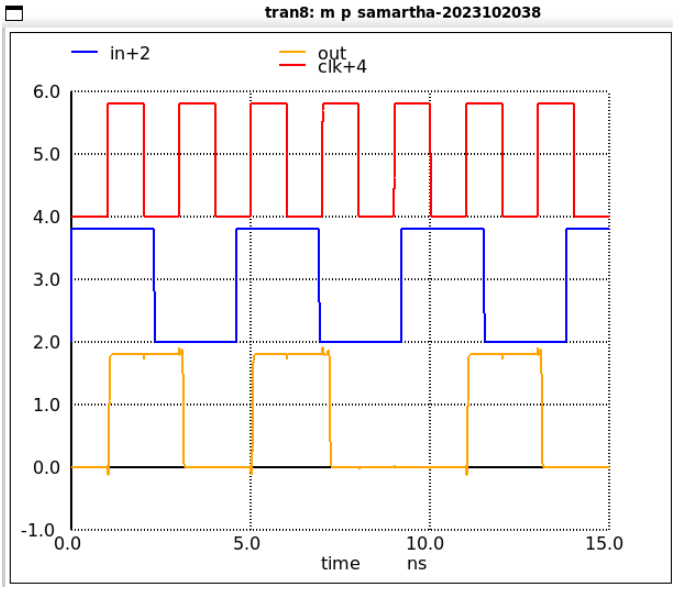

Fig. 5: TSPC FF simulation

Now to measure Setup time, we see that the it is exactly equal to the t_{CQ} of the negative edge triggered TSPC latch. This is because if input is changed anytime before this value before the clock edge, the negative latch has not enough time to charge/discharge the input node to the positive edge triggered latch, so reliable value is not passed through the FF.

For hold time analysis, we know that the hold time for TSPC FF is ≈ 0. This is a consequence of how the FF is built using two different edge triggered latches. When input changes at the positive edge of the clock, the input to the positive latch would not be disturbed, because the negative latch is turned off. This is one of the benefits of using TSPC FF for implementing D-Flipflop.

For t_{CQ}, we juse measure the 50%-50% delay between clock and rising edge of the output. Here all the delays are measured using 50%-50% measure. From the above plot we

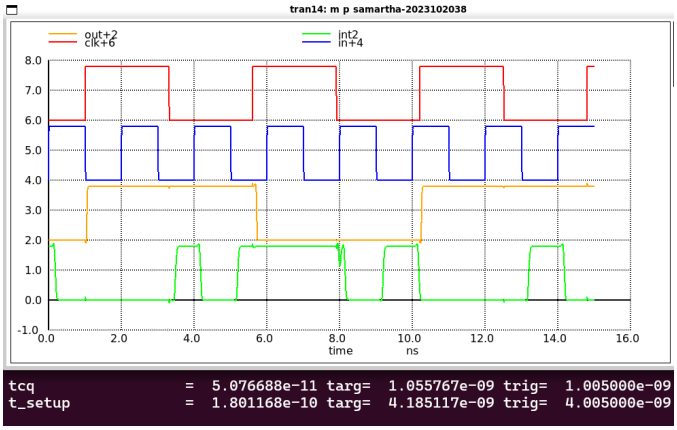

Fig. 6: Plot for t_{CQ} and setup time

have the following

$$\begin{aligned} \textbf{setup time} &= \textbf{180.117 ps} \\ \textbf{hold time} &= \textbf{0 ps} \\ \mathbf{t_{CQ}} &= \mathbf{50.767} \end{aligned}$$

V. Stick Diagrams

The following are the stick diagrams of Inverter, NAND, NOR, AND, OR, XOR gates and TSPC Filpflop

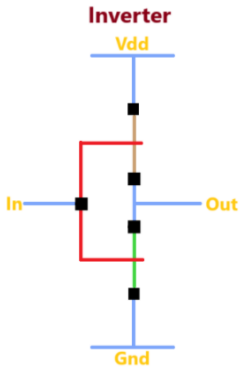

Fig. 7: Stick diagram for inverter

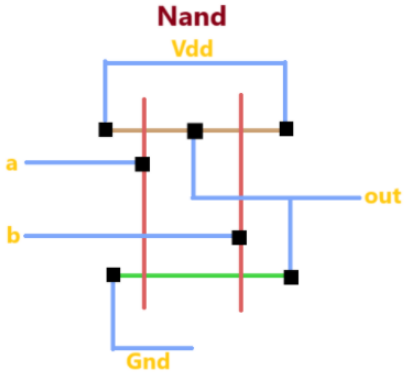

Fig. 8: Stick diagram for NAND gate

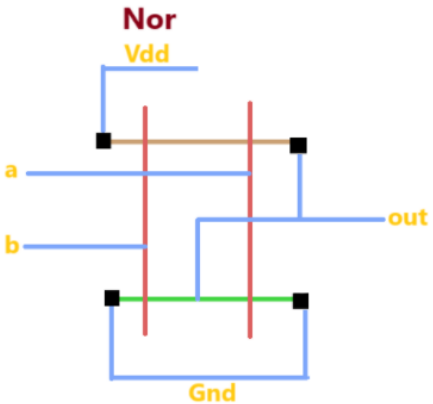

Fig. 9: Stick diagram for NOR gate

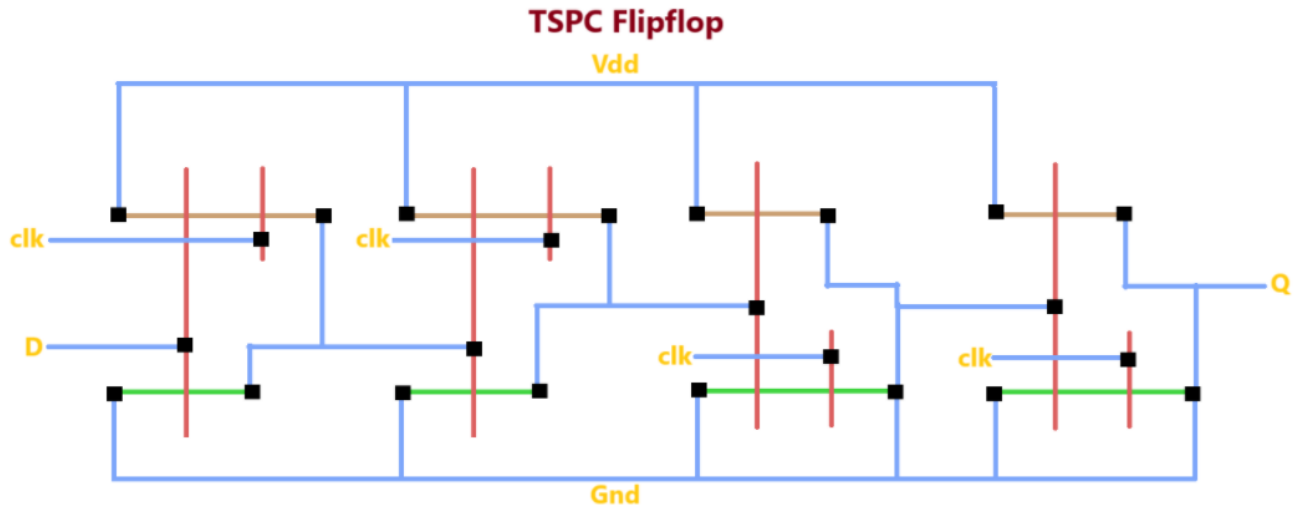

Fig. 10: Stick diagram for TSPC flipflop

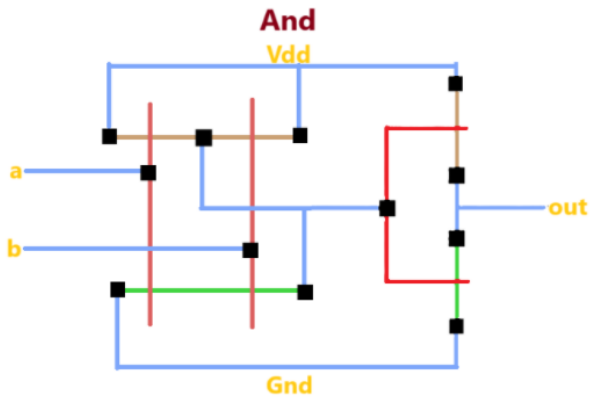

Fig. 11: Stick diagram for AND gate

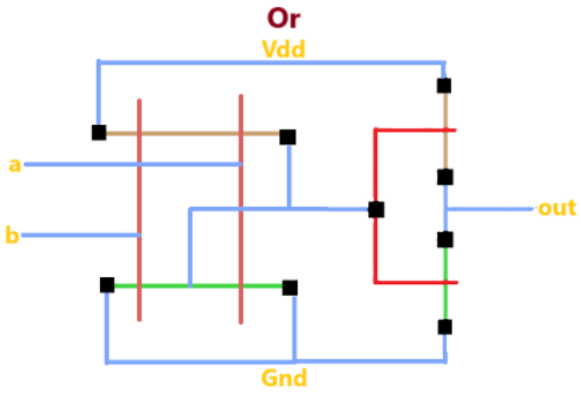

Fig. 12: Stick diagram for OR gate

.

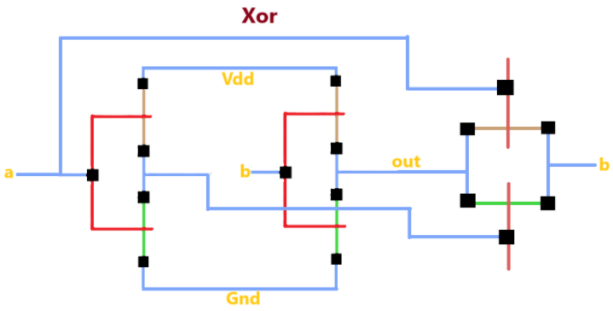

Fig. 13: Stick diagram for XOR gate

VI. Magic layout of building blocks

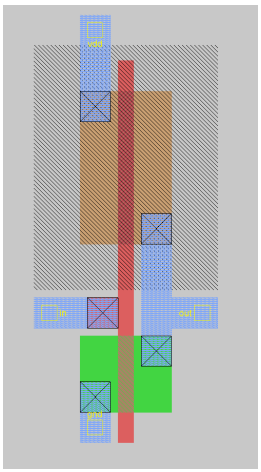

Fig. 14: NOT gate

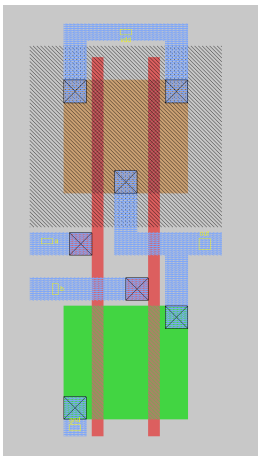

Fig. 15: NAND gate

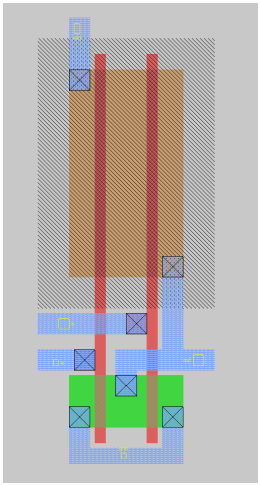

Fig. 16: NOR gate

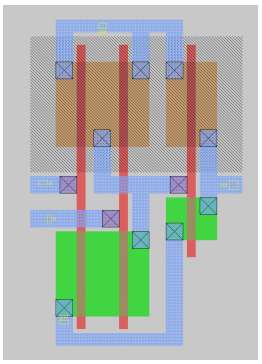

Fig. 17: AND gate

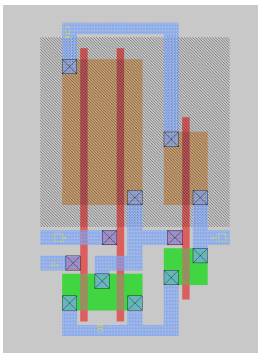

Fig. 18: OR gate

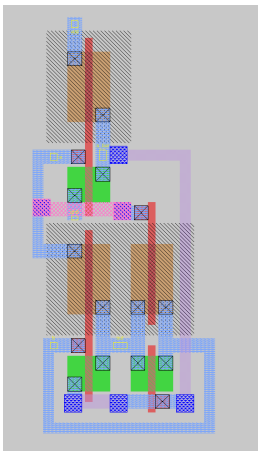

Fig. 19: XOR gate

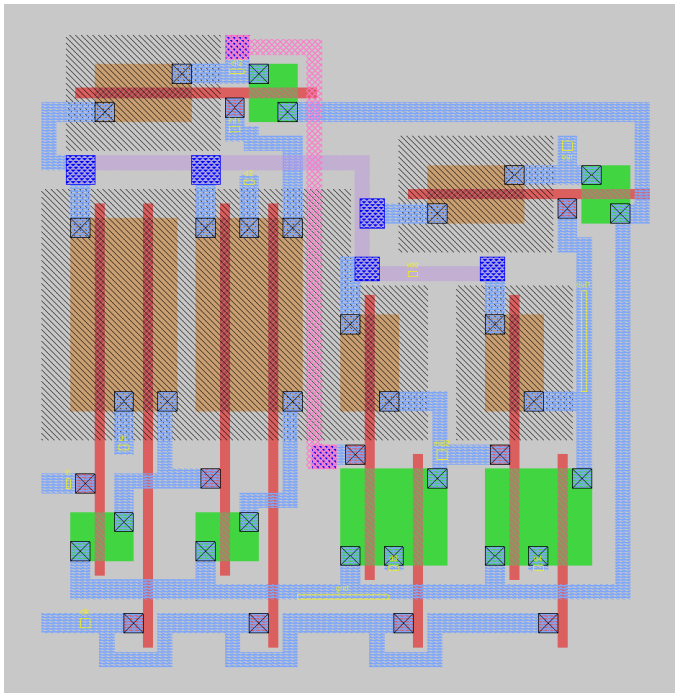

Fig. 20: Magic layout for TSPC flipflop

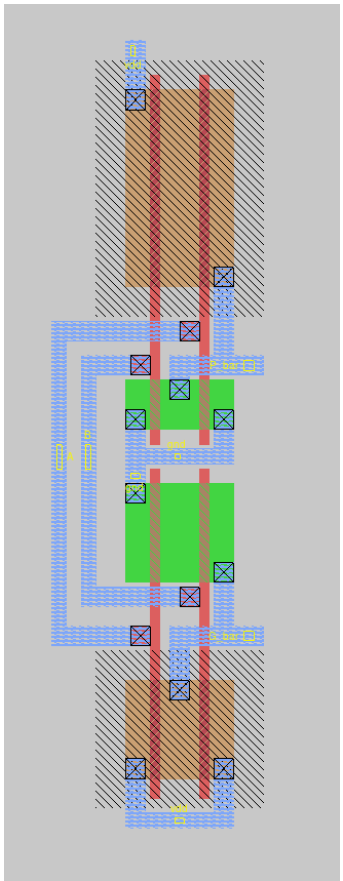

Fig. 21: Magic layout for P/G block

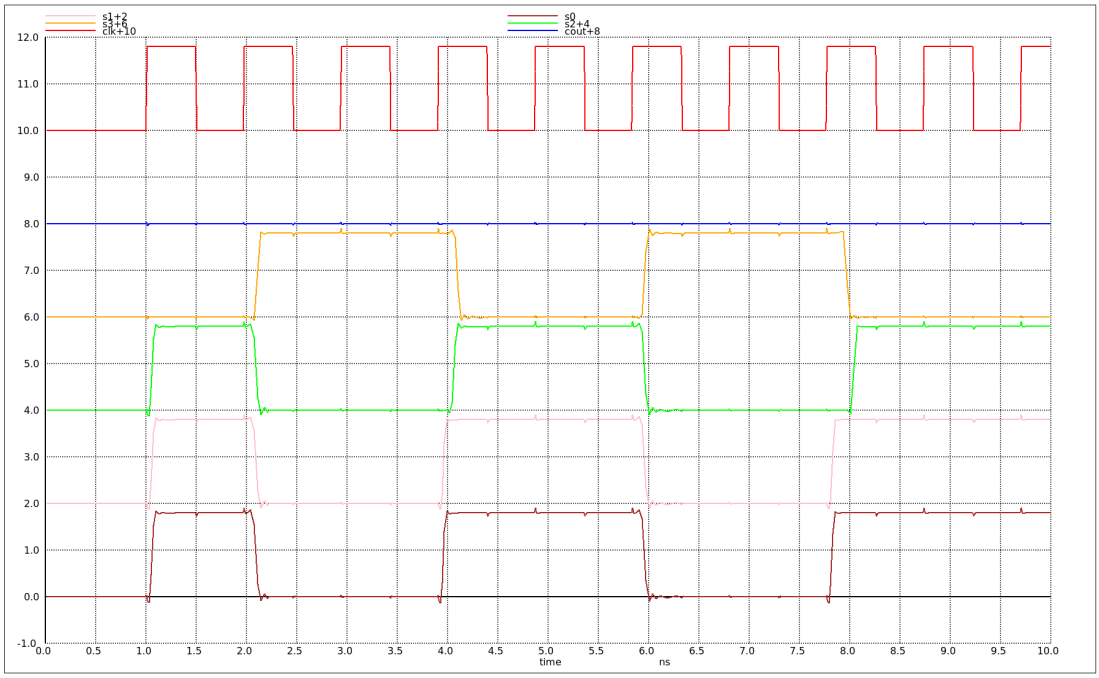

Fig. 22: Final output waveforms

VII. Complete circuit integration on NG-Spice

Above is the plot for the final waveforms. The first signal on the top is the clock signal, then the 5 signals are C_{out}, S_3, S_2, S_1 and S_0 from top to bottom. Here the following inputs are used for the testing.

```
.param time_period = 1/1.035G
.param t_on = time_period/2

VA3 A3 gnd 0
VA2 A2 gnd 0
VA1 A1 gnd 0
VA0 A0 gnd 0

VB3 B3 gnd 0
VB2 B2 gnd 1.8
VB1 B1 gnd 1.8
VB0 B0 gnd pulse 0 1.8 0 10ps 10ps 2.2ns 4.4ns
```

Fig. 23: Input signals used for the plot

For these input combination, it can be verfied that the the above plot gives the right output. **These inputs are chose because these give the maximum possible delay for our circuit, since the signal has to travel the longest path.** This can be verified from 2. The above analysis also has been done for the maximum frequency possible where out circuit works properly. Thus any higher frequency gives non-reliable outputs. Thus the maximum frequency of operation is :

$$\mathbf{f_{max} = 1.035GHz}$$

The following is the output on the command window for Propagation delay of the circuit, in the worst case. Thus any other input combination will definitely give lesses propagation delay. Also we output the max current drawn through Vdd.

```
Measurements for Transient Analysis
tpd_rise            =  6.001206e-10 targ=  1.740733e-09 trig=  1.140612e-09
tpd_fall            =  6.044603e-10 targ=  3.739981e-09 trig=  3.135521e-09
tpd_ave             =  6.02290e-10
peak_current        =  2.761811e-02 at=  8.739469e-09
```

Fig. 24: Propagation delay and peak current

As seen in the current plot, and from the peak current printed on Command window we have the following :

$$\mathbf{P_{max} = 27.6m * 1.8 \approx 50mW}$$

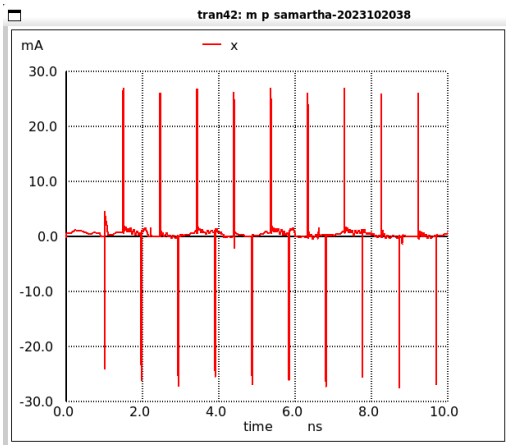

Fig. 25: Current drawn through Vdd

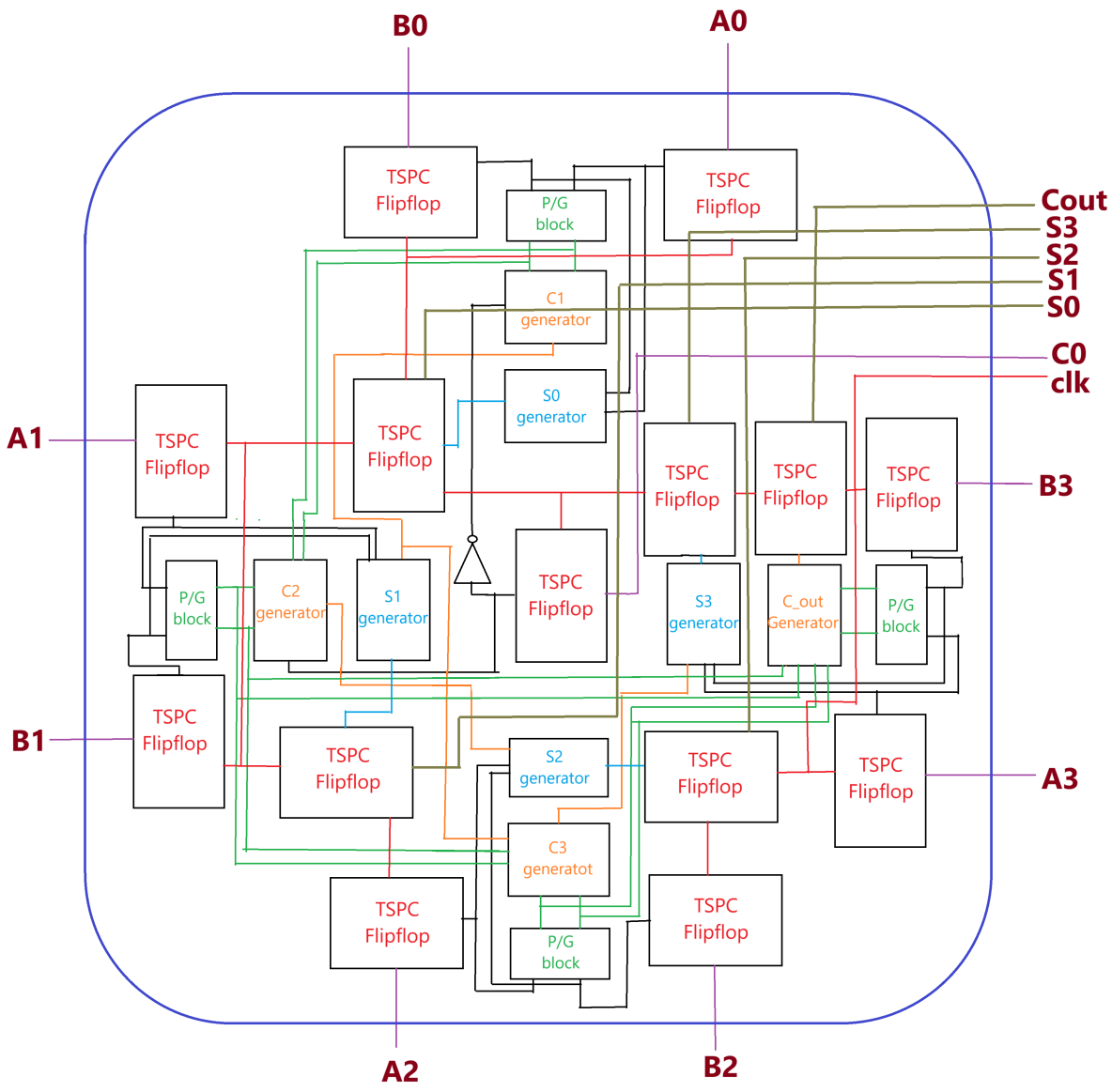

Fig. 26: Floorplan for the Magic layout

VIII. Floorplan for Magic layout

The following is the floorplan for the magic layout. The figure contains various blocks such as P/G block, Carry generator, sum generator and TSPC flipflops. The inputs and outputs are shown on the top-right. Now, we list out the pitches for regular structures seen in the floorplan above. Here since the carry generator and sum generator are implemented in a dynamic fashion to reduce the layout area, it is not meaningful to talk about its pitches. It is also slightly different for each bit in out design.

Structure	Hor. Pitch	Vertical Pitch
TSPC FF	126λ	131λ
P/G block	43λ	160λ
XOR gate	52λ	119λ

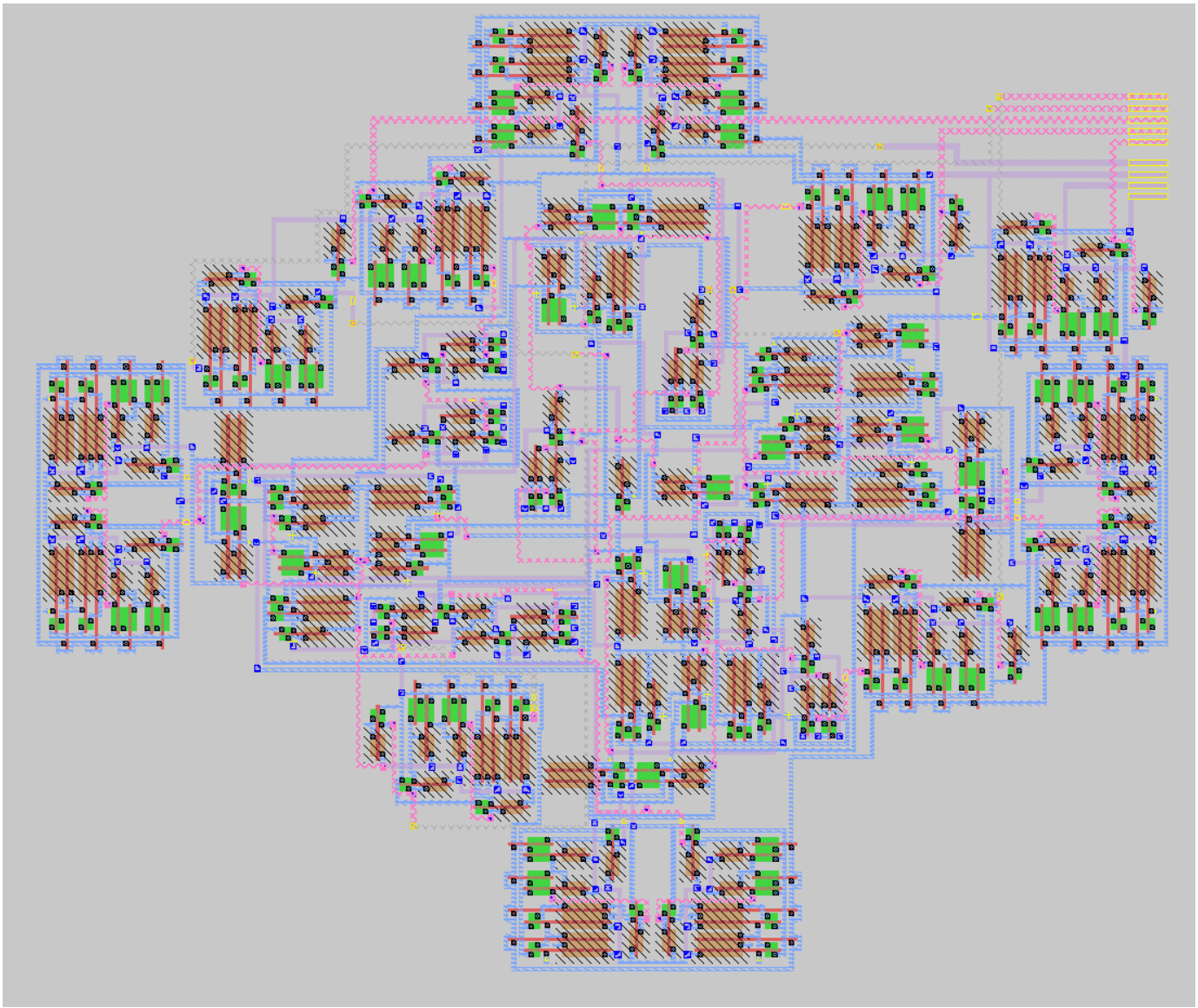

Fig. 27: Complete Magic layout

IX. Complete circuit Magic layout

Above is the complete layout of the circuit. As seen in the floorplan 26, we can somewhat identify similar structure/motifs in the above layout. The complete layout is of dimension $\mathbf{1021}\ \lambda\ \mathbf{x}\ \mathbf{862}\ \lambda$. Although the area covered would be much lesser dues to the extra space towards the corners.

```
  Measurements for Transient Analysis

tpd_rise            =  6.406136e-10 targ=  1.778701e-09 trig=  1.138088e-09
tpd_fall            =  5.688357e-10 targ=  3.698677e-09 trig=  3.129841e-09
tpd_ave             =  6.04725e-10
peak_current        =  1.868153e-02 at=  2.933077e-09
```

Fig. 28: Postlayout output on the command window

We also note that postlayout circuit works fine for the maximum frequency of the clock which was determined pre-layout. Also as seen in the image above, the $\mathbf{t_{PD}(max) \approx 605ps}$.

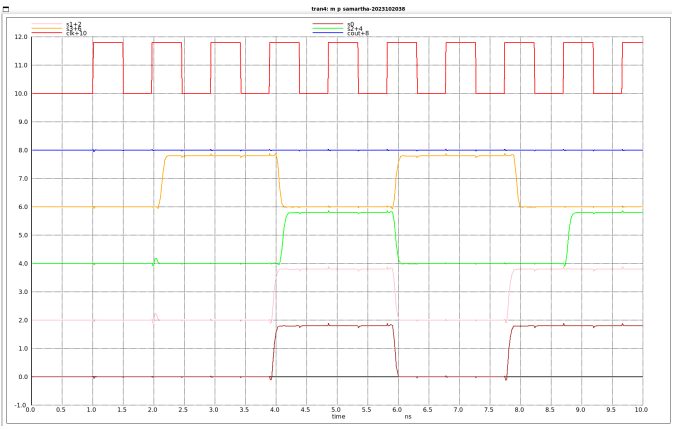

Fig. 29: Postlayout waveforms

Now comparing the pre-layout and post-layout values, we have the following :

Property	Pre-layout	Post-layout
Max. Freq. of operation	1.035 GHz	1.035 GHz
Max. t_{PD}	602 ps	605 ps
Peak current	27.6 mA	18.69 mA

X. Reporting Delay and Max. frequency of CLA

As discussed already above, we have the following value for worst case delay, which is analysed for the longest path in the circuit giving appropriate inputs. The frequency of operation is determined by increasing the frequency of clock until the CLA don't give reliable outputs.

$$\frac{\mathbf{f_{max} = 1.035GHz}}{\mathbf{t_{PD,max} = 605ps}}$$

XI. Icarus Verilog HDL

Now we write the HDL code to realize the CLA. We follow a structural description of the circuit. To test the circuit, we take an example of **A = 13, B = 7, C0 = 1**. Thus we expect an output of **S = 5, Cout = 1**. As seen in the following, we see that we are indeed getting right outputs.

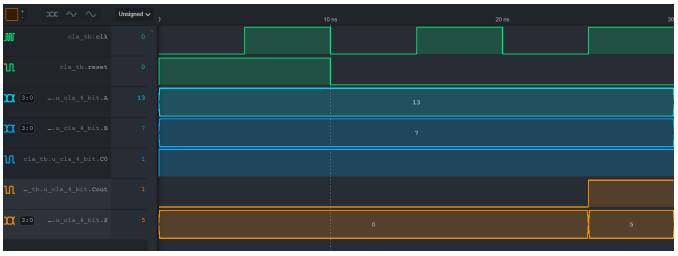

Fig. 30: Verilog Waveforms

XII. FPGA

Now we implement the HDL on actual hardware using FPGA. The following is the constraint scheme used for the PMOD pins to output onto the oscilloscope.

```
# PMOD Output
set_property -dict {PACKAGE_PIN B13 IOSTANDARD LVCMOS33} [get_ports {S[0]}]
set_property -dict {PACKAGE_PIN A14 IOSTANDARD LVCMOS33} [get_ports {S[1]}]
set_property -dict {PACKAGE_PIN B15 IOSTANDARD LVCMOS33} [get_ports {S[2]}]
set_property -dict {PACKAGE_PIN A15 IOSTANDARD LVCMOS33} [get_ports {S[3]}]
set_property -dict {PACKAGE_PIN B14 IOSTANDARD LVCMOS33} [get_ports {Cout}]

```

Fig. 31: Constraints used for PMOD pins

For this, we take an example of **A = 13, B = 12, C0 = 1**. Thus we expect **Cout =1, S = 1010**. As seen from the plots below, we are indeed getting the right outputs. Here the signals are $C_{out}, S_3, S_2, S_1, S_0$ from top to bottom. Here the Max value of 3.3V for the signal indicate a binary of 1 and a few tens of millivolts caused due to noise, can be considered as a binary 0.

Fig. 32: Output observed on oscilloscope

XIII. Acknowledgment

Foremost I would like to thank the professor, Dr. Abhishek Shrivastava for their invaluable guidance, encouragement, and support throughout the course of this project. I would also like to thank the TAs who were readily available for clearing any doubts, this made the project much easier. Finally, I would also like to thank my peers, whose valuable discussions and insightful ideas greatly contributed to the successful completion of this project.

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 (EDSSC), Hsinchu, Taiwan, 2017, pp. 1-2, doi: 10.1109/EDSSC.2017.8126457. keywords: Logic gates;Adders;Power demand;Mathematical model;Acceleration;MOSFET;CLA;CMOS;HSpice and logic gate, J. Miao and S. Li, "A novel implementation of 4-bit carry look-ahead adder," , Hsinchu, Taiwan, 2017, pp. 1-2, doi: 10.1109/EDSSC.2017.8126457. keywords: Logic gates;Adders;Power demand;Mathematical model;Acceleration;MOSFET;CLA;CMOS;HSpice and logic gate

[2] 4E, CMOS VLSI Design A cicuits and systems perspective by Neil H.E. Weste, David Money Harris

[3] Ihsen Alhouni, Hamzeh Ahangari, Ozcan Ozturk, NS-SRAM: Neighborhood Solidarity SRAM for Reliability Enhancement of SRAM Memories

[4] J. Yuan and C. Svensson, "High-speed CMOS circuit technique," IEEE J. Solid-State Circuits, vol. 24, pp. 62–70, Feb. 1989.