Fastest Carry Look-Ahead Adder Design for Enhanced Computational Efficiency

Vedant Pahariya
ECD IIIT-H
2023112012
vedant.pahariya@research.iiit.ac.in

***Abstract*—This paper proposes the design for the fastest CLA. It mentions about all the blocks and topology used in the design. It also mentions about the simulation results and the comparison with existing designs. The design is implemented in 180nm technology. The design is compared with the existing designs and it is found that the proposed design is the fastest among all.**

***Index Terms*—Carry Look-Ahead Adder, CLA, VLSI, Computational Efficiency, 180nm Technology, Digital Design, High-Speed Arithmetic**

I. Introduction

The Carry Look-Ahead Adder (CLA) is a digital circuit that is used to add two binary numbers. It is a basic unit component for all arithmetic processes which goes in processor. Making it faster can enhance the computational efficiency of whole system. The CLA is a fast adder because it can generate the carry signals for all the bits in parallel. The CLA is faster than the Ripple Carry Adder (RCA) because the RCA generates the carry signals sequentially, faster than the Carry Select Adder (CSA) because the CSA generates the carry signals for a group of bits in parallel, faster than the Carry Skip Adder (CSKA) because the CSKA generates the carry signals for a group of bits in parallel, faster than the Carry Increment Adder (CIA) because the CIA generates the carry signals for a group of bits in parallel.

II. Carry Look-Ahead Adder Design

CLA design consists of three blocks: the Propagate & Generate block, Carry Look Ahead (CLA) block and Sum block. The Propagate & Generate block gives carry generate (G_i) and propagate generate (P_i) signal for every i^{th} bit of inputs. The output of this block is used in CLA block to generate Carry C_i which is then utilised in the Sum block to get the Sum output. The basic CLA design is based on the following equations:

$$G_i = A_i \cdot B_i \tag{1}$$

$$P_i = A_i \oplus B_i \tag{2}$$

$$C_i = G_i + P_i \cdot C_{i-1} \tag{3}$$

where G_i is the Generate signal, P_i is the Propagate signal, C_i is the Carry signal, A_i is the i^{th} bit of the first number, B_i is the i^{th} bit of the second number, and C_{i-1} is the $(i-1)^{th}$ Carry signal.

Here, we are focusing on the 4-bit CLA design. Therefore, using recursive approach for carry in above equations, we can write the equations for the 4-bit CLA design as follows:

$$C_0 = G_0 + P_0 \cdot C_{in} \tag{4}$$

$$\begin{aligned} C_1 &= G_1 + P_1 \cdot C_0 \\ &= G_1 + P_1 \cdot (G_0 + P_0 \cdot C_{in}) \\ &= G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_{in} \end{aligned} \tag{5}$$

$$\begin{aligned} C_2 &= G_2 + P_2 \cdot C_1 \\ &= G_2 + P_2 \cdot (G_1 + P_1 \cdot (G_0 + P_0 \cdot C_{in})) \\ &= G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_{in} \end{aligned} \tag{6}$$

$$\begin{aligned} C_3 &= G_3 + P_3 \cdot C_2 \\ &= G_3 + P_3 \cdot (G_2 + P_2 \cdot (G_1 + P_1 \cdot (G_0 + P_0 \cdot 0))) \\ &= G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 \end{aligned} \tag{7}$$

where C_i, G_i, and P_i are the Carry, Generate, and Propagate signals for the i^{th} bit, respectively. C_{in} is the input Carry signal.

A. Improving the Basic CLA Design

The above basic CLA design can be improved extensively by making the following changes:

1) Use of OR gate instead of XOR gate for Propagate block: XOR gate can be replaced by OR gate which can be further reduced to the NOR logic for the Propagate block to reduce the number of gates in the design. The Propagate block can be designed as follows:

$$P_i = A_i + B_i \tag{8}$$

To demonstrate that both XOR and OR gates give the same results for the Propagate block, we can see in the following truth table that results are same for both XOR and OR gates except when both inputs are 1.

TABLE I
Truth Table for XOR and OR Gates

A_i	B_i	$A_i \oplus B_i$ (XOR)	$A_i + B_i$ (OR)
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1

In case of carry generation, when both inputs are High (1), that is both $A_i = 1$ and $B_i = 1$:

$$\begin{aligned} G_i &= A_i \cdot B_i = 1 \cdot 1 = 1 \\ C_i &= G_i + P_i \cdot C_{i-1} \\ &= 1 + P_i \cdot C_{i-1} \\ &= 1 \text{ (regardless of } P_i \text{ and } C_{i-1}) \end{aligned}$$

Therefore, when both inputs are 1, the carry signal is always generated ($C_i = 1$) regardless of the value of the propagate signal (P_i) or the previous carry (C_{i-1}).
As shown above, for the purpose of carry propagation in the CLA design, the OR gate can be used in place of XOR to reduce the number of gates.

2) Modification of the equations for the Carry signals to reduce the number of gates: The equations for the Carry signals for basic CLA use AND and OR gates which inherently includes two extra transistors for each gate becuase of the presence of the inverter. The number of gates can be reduced by modifying the logic such that it uses NAND and NOR gates in place of AND and OR wherever possible. Starting with simplification of C_1, we know that:

$$C_1 = G_0 + P_0 \cdot C_0$$

Here, we can convert AND gate between P_0, C_0 to NOR gate as shown below:

$$C_1 = G_0 + \overline{\left(P_0' + \overline{C_0}\right)}$$

Further, we can convert OR gate to NAND gate as shown below:

$$C_1 = \overline{G_0' \left(P_0' + \overline{C_0}\right)}$$

Similarly, for the other carry signals, the modified equations can be written as follows:

$$C_1 = \overline{G_0' \left(P_0' + \overline{C_0}\right)} \tag{9}$$

$$C_2 = \overline{G_1' \left(P_1' + G_0'\right)} + \left(\overline{P_1' + P_0'}\right) C_0 \tag{10}$$

$$C_3 = \overline{G_2' \left(P_2' + G_1'\right)} + \left(\overline{P_2' + P_1'}\right) \overline{G_0' \left(P_0' + \overline{C_0}\right)} \tag{11}$$

$$G'_{\text{out}} = \overline{G_3' \left(P_3' + G_2'\right)} + \left(\overline{P_3' + P_2'}\right) \overline{G_1' \left(P_1' + G_0'\right)} \tag{12}$$

$$P'_{\text{out}} = \overline{\left(P_3' + P_2'\right)\left(P_1' + P_0'\right)} \tag{13}$$

$$C_4 = \overline{G'_{\text{out}} \left(P'_{\text{out}} + \overline{C_0}\right)} \tag{14}$$

where G_i' and P_i' are the modified Generate and Propagate signals for the i^{th} bit, respectively.

B. Topology and Sizing of Each Block

Based on the equations and logic functions derived above, the topology of the 4-bit CLA design can be implemented as shown in the figure below. The design is implemented in 180nm technology. The sizing of the transistors in each block is done considering the speed of the design. I have followed the CMOS logic for all the logic gates except XOR. For XOR, I used Pass Transistor Logic because it uses less no. of transistors. Here, I implemented the method of logical effort for sizing each block as following:

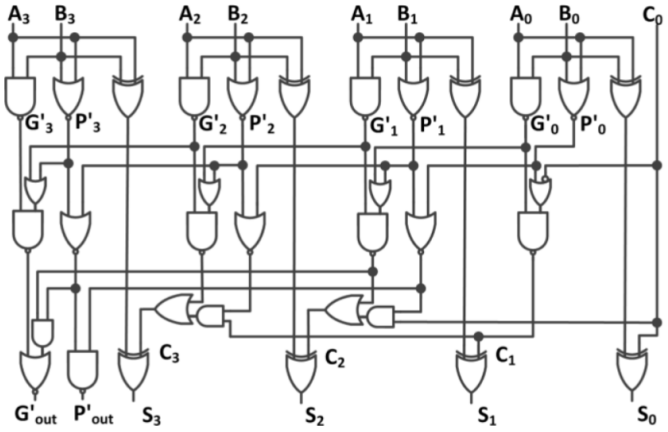

Fig. 1. 4-bit Carry Look-Ahead Adder Design

1) Propagate & Generate Block: This block includes three gates for each carry bit: One NOR gate for the Propagate signal, one NAND gates for the Generate signal and one XOR gate for the Sum signal.

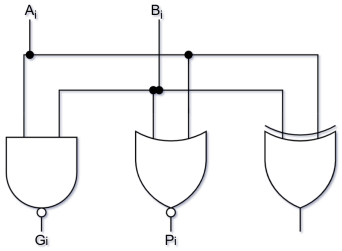

Fig. 2. Propagate & Generate Block

Accoroding to the method of logical effort, the size of these gates is larger i.e. just double the proceedings gates of the intermediate block of CLA. So, the table below shows the sizing of gates in this block:

TABLE II
SIZING OF GATES IN PROPAGATE & GENERATE BLOCK

Gate	Width (W_N/W_P) (μm)	Length (μm)	Number of Gates
NOR	3.6 / 7.2	0.18	4
NAND	3.6 / 3.6	0.18	4
XOR	0.36 / 0.72	0.18	4

2) Carry Look Ahead Block: This block again further divided into two more subblocks namely intermediate block and AndOr block. The intermediate and the AndOr block includes three gates and two gates respectively for each carry bit. The intermediate block includes a NOR gate, NAND and OR gate and the AndOr block includes one AND gate and one OR gate.

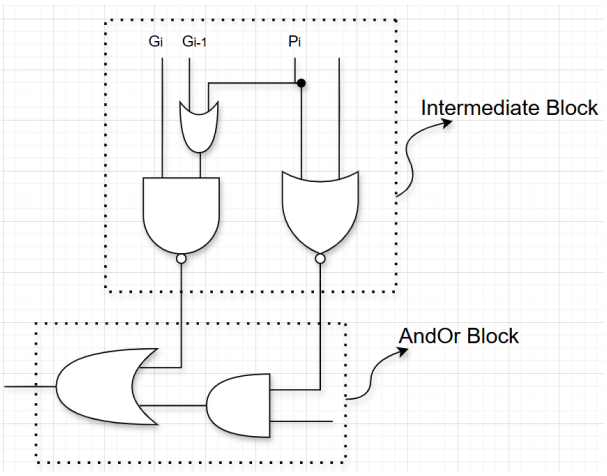

Fig. 3. Carry Look Ahead Block for Single Carry Bit

The sizing of gates in the intermediate and AndOr block is done as per the method of logical effort. As a result, also mentioned in the previous block that this block sizing is half of the propagate & generate block sizing. The OR gate is implemented using the NOR gate with an inverter at the output. Similarly for the AND gate, the NAND gate is used with an inverter at the output. The table below shows the sizing of gates in these blocks:

TABLE III
SIZING OF GATES IN INTERMEDIATE AND ANDOR BLOCKS

Block	Gate	Width (W_N/W_P) (μm)	Length (μm)
Intermediate	NOR	0.9 / 1.8	0.18
	NAND	1.8 / 1.8	0.18
	OR	0.9 / 1.8	0.18
AndOr	AND	1.8 / 1.8	0.18
	OR	0.9 / 1.8	0.18
Inverter	-	0.9 / 1.8	0.18

3) Sum Block: This block includes single XOR gate for the Sum signal which is implemented using Pass Transistor Logic. The sizing of this is set to be least possible just enough to drive the W-sized inverter and satisfy the DRC constraints because it is the last block in the design and don't have to drive any other block.

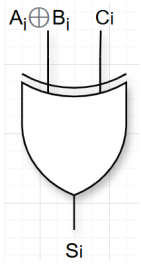

Fig. 4. Sum Block

TABLE IV
SIZING OF GATES IN SUM BLOCK

Gate	Width (W_N/W_P) (μm)	Length (μm)	Number of Gates
XOR	0.36 / 0.72	0.18	6

4) D-Flip Flop: It is implemented using the True Single Phase Clocked (TSPC) technology. It includes total no. of 12 transistors in CMOS logic style. The sizing of this is done by equating the resistances with the minimum W-sized inverter. There are two parts in the postive edge trigerred TSPC D-Flip Flop: Low-Level Latch and High-Level Latch.

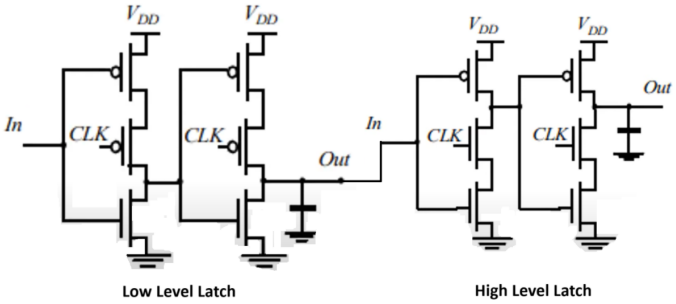

Fig. 5. TSPC D-Flip Flop

TABLE V
SIZING OF TRANSISTORS IN TSPC D-FLIP FLOP

Block	Transistor	Width (μm)	Length (μm)
Low-Level Latch	PMOS	3.6	0.18
	NMOS	0.9	0.18
High-Level Latch	PMOS	1.8	0.18
	NMOS	1.8	0.18

III. DELAY CHARACTERISTICS OF D-FLIP FLOP (PRELAYOUT)

The TSPC D Flip-Flop was simulated using NGSPICE with a clock period of 2ns and input period of 4ns. Both clock and input signals were given sharp rise/fall times of 50ps to analyze the flip-flop's delay characteristics.

A. Clock to Q Delay

Upon simulating, we observe the following waveforms and terminal outputs:

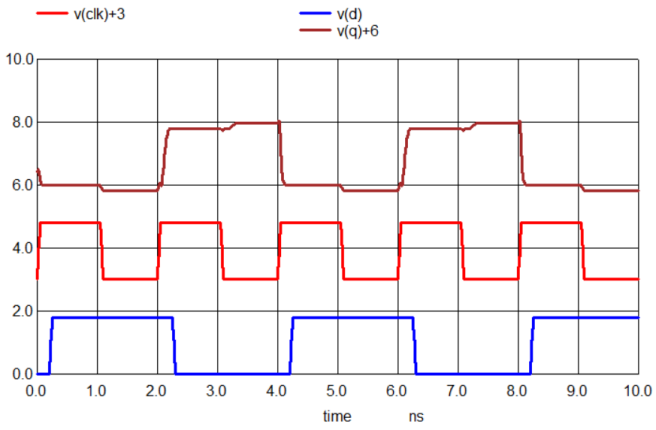

Fig. 6. Waveforms Output of D-Flip Flop

```
  Measurements for Transient Analysis

rise_delay          =  9.157922e-11 targ=  2.116579e-09 trig=  2.025000e-09
fall_delay          =  3.076888e-11 targ=  4.055769e-09 trig=  4.025000e-09
propagation_delay   =  6.11740e-11
```

Fig. 7. Terminal Output for Delay

The simulation gives the following timing parameters:

- Rise Delay: 91.58ps - Time taken for the output to rise after the triggering clock edge
- Fall Delay: 30.77ps - Time taken for the output to fall after the triggering clock edge
- Average Propagation Delay: 61.17ps - Mean of rise and fall delays

The asymmetry between rise and fall delays (ratio $\approx$ 3:1) can be attributed to:

- Different sizing of PMOS/NMOS transistors in the low-level and high-level latches
- Inherent mobility difference between PMOS and NMOS devices
- Cascaded structure of the latches affecting signal propagation paths

B. Setup Time

Setup time (t_{setup}) is the minimum time before the active clock edge during which the data input (D) must remain stable. For TSPC D flip-flop, setup time is equal to the propagation delay of the Low-Level Latch because when clock is low, the Low-Level Latch is active and as it rises to high, the correct stable value of D must be present at the output of the Low-Level Latch for ensuring correct operation. This stable value of D appears at the output of the Low-Level Latch after the propagation delay of the Low-Level Latch.

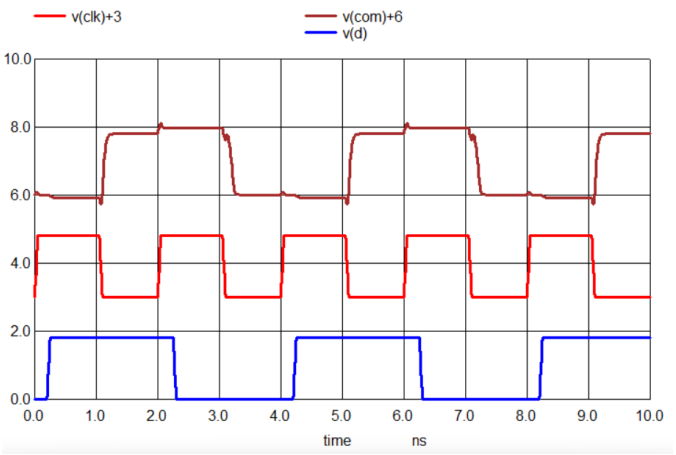

Fig. 8. Waveforms Output of Low-Level Latch

```
  Measurements for Transient Analysis

rise_delay          =  4.883258e-11 targ=  1.123833e-09 trig=  1.075000e-09
fall_delay          =  1.264689e-10 targ=  3.201469e-09 trig=  3.075000e-09
propagation_delay   =  8.76507e-11
```

Fig. 9. Terminal Output for Setup Time

C. Hold Time

Hold time (t_{hold}) is the minimum time after the active clock edge during which the data input must remain stable. For TSPC D flip-flop, hold time is zero because the High-Level Latch is active when the clock is high and the output of the High-Level Latch is not dependent on the input D when the clock is high. Therefore, the input D can change immediately after the active clock edge without affecting the output Q.

TABLE VI
TIMING CHARACTERISTICS OF TSPC D-FLIP FLOP

Parameter	Value (ps)
Rise Delay	91.58
Fall Delay	30.77
Propagation Delay	61.17
Setup Time	87.65
Hold Time	0

IV. STICK DIAGRAM OF UNIQUE GATES

A. Inverter

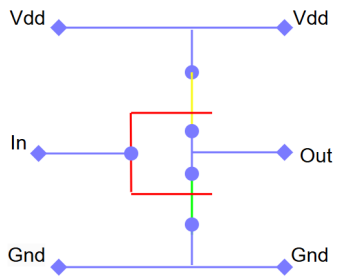

Fig. 10. Inverter Stick Diagram

B. NOR Gate

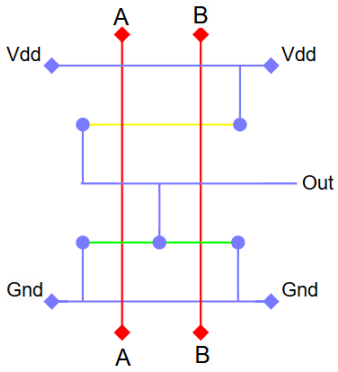

Fig. 11. NOR Gate Stick Diagram

C. NAND Gate

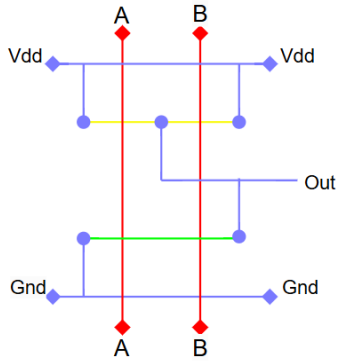

Fig. 12. NAND Gate Stick Diagram

D. XOR Gate

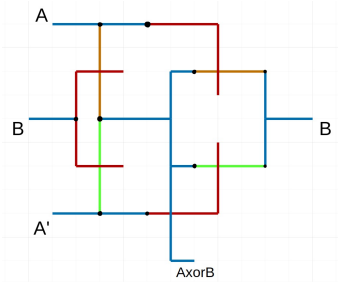

Fig. 13. XOR Gate Stick Diagram

V. Prelayout Simulation Results & Analysis of CLA

The 4-bit Carry Look-ahead Adder was simulated using NGSPICE with the following test conditions:

A. Verification of functionality

1) Input Specifications:

- Supply Voltage (VDD): 1.8V
- Input A (4-bit):
  - A0: Pulse starting at 10ns, Period = 40ns
  - A1: Pulse starting at 15ns, Period = 50ns
  - A2: Pulse starting at 20ns, Period = 60ns
  - A3: Pulse starting at 25ns, Period = 70ns
  - Rise/Fall times: 50ps
- Input B (4-bit):
  - B0: Pulse starting at 12ns, Period = 44ns
  - B1: Pulse starting at 18ns, Period = 56ns
  - B2: Pulse starting at 24ns, Period = 68ns
  - B3: Pulse starting at 30ns, Period = 80ns
  - Rise/Fall times: 50ps
- Carry Input (Cin): Pulse starting at 36ns, Period = 88ns
- Clock (CLK): Period = 8ns

2) Output Analysis: The simulation produces the following outputs:

- Sum outputs (S0-S3): Generated through XOR combinations of inputs and internal carries
- Final Carry (C4): Represents the overflow from the 4-bit addition

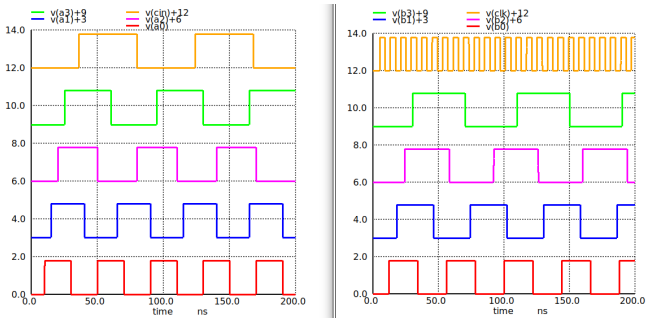

Fig. 14. Input Waveform

Above plots show the input waveforms given to the 4-bit CLA. The left plot shows the A bits along with C_{in} and right one shows the B bits along with the clock.

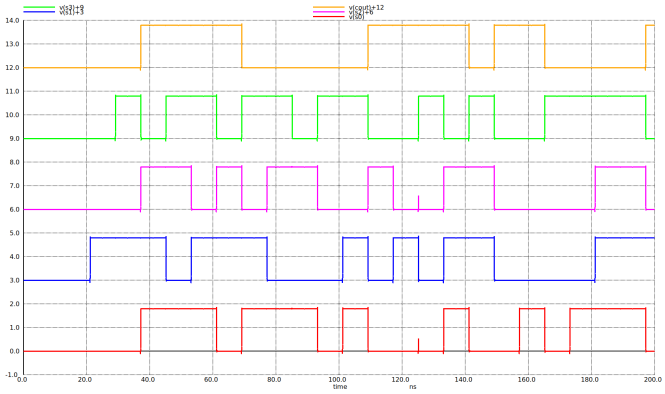

Fig. 15. Output Waveform

Above plot shows the output waveforms which includes the four Sum bits and the final Carry bit.

The waveforms demonstrate correct operation of the CLA with proper carry propagation and sum generation across all bit

positions. The progressive delays in input signals help verify the adder's functionality under varying timing conditions.

B. Worst Case Delay of CLA Adder Block

Worst case delay is the maximum time taken for the output to stabilize after the input changes. This happens in the path which has highest resistance or very large number of gates. In the 4-bit CLA design proposed above, the worst case delay is observed in the carry propagation path from B_0 through C3 to S3 because it is the longest path with total 9 gates in midway. For considering the worst case delay, the S3 bit must change when B0 bit changes. So, here we take the input signals as follows:

- $A_0 = A_1 = A_2 = A_3 = 0, B_1 = B_2 = 1, B_3 = 0$
- B_0: Pulse starting at 25ns, Period = 70ns
- C_{in} = 1
- Clock (CLK): Pulse starting at 0ns, Period = 8ns

With the above inputs, we get the simulation results for the worst case delay as follows:

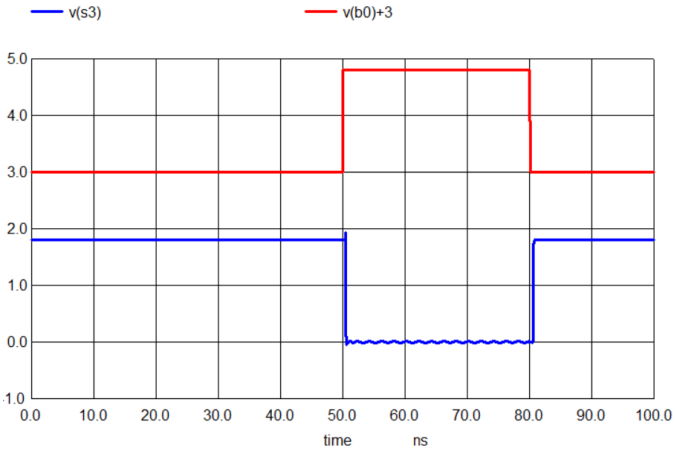

Fig. 16. WaveForms for the Worst Case Delay

```
Measurements for Transient Analysis

rise_delay          =  4.621690e-10 targ=  5.048717e-08 trig=  5.002500e-08
fall_delay          =  4.961677e-10 targ=  8.057117e-08 trig=  8.007500e-08
tpd3                =  4.79168e-10
```

Fig. 17. Worst Case Delay Terminal Output

The simulation results show that the worst case delay for the 4-bit CLA design is 479.16ps. This delay is observed in the carry propagation path from B_0 through C3 to S3.

C. Maximum Clock Speed

1) Theoritical Calculation: The maximum clock speed of the 4-bit CLA design can be calculated using the following inequality:

$$t_{CQ_1} + t_{pd} + t_{su} \leq T_{clk} \tag{15}$$

$$f_{max} = \frac{1}{t_{CQ_1} + t_{pd} + t_{su}} \tag{16}$$

where t_{CQ_1} is the clock-to-Q delay of the first flip-flop, t_{pd} is the worst-case propagation delay of the CLA adder block, and t_{su} is the setup time of the flip-flop.

Given the values we calculated earlier:

- $t_{CQ_1} = 61.17$ ps
- $t_{pd} = 479.16$ ps
- $t_{su} = 87.65$ ps

The maximum clock frequency is:

$$f_{max} = \frac{1}{61.17 \text{ ps} + 479.16 \text{ ps} + 87.65 \text{ ps}} \approx 1.59 \text{ GHz} \tag{17}$$

2) Found using Simulation: Using the Ngspice Simulation, I found that the maximum clock frequency of the 4-bit CLA design is 1.04 GHz which is less than the theoritical value. This difference can be attributed to the following factors:

- Parasitic Capacitances: The simulation does not consider the parasitic capacitances which can affect the delay and hence the maximum clock frequency.
- Wire Delays: The simulation does not consider the wire delays which can affect the delay and hence the maximum clock frequency.

VI. Delay Characteristics of D-Flip Flop (Postlayout)

The TSPC D Flip-Flop was simulated using NGSPICE with a clock period of 2ns and input period of 4ns. Both clock and input signals were given sharp rise/fall times of 50ps to analyze the flip-flop's delay characteristics.

A. Clock to Q Delay

Upon simulating, we observe the following waveforms and terminal outputs:

The simulation gives the following timing parameters:

- Rise Delay: 92.21ps - Time taken for the output to rise after the triggering clock edge
- Fall Delay: 24.09ps - Time taken for the output to fall after the triggering clock edge
- Average Propagation Delay: 53.15ps - Mean of rise and fall delays
- Setup Time: 67.43ps - Minimum time before the active clock edge during which the data input must remain stable
- Hold Time: 0ps - Minimum time after the active clock edge during which the data input must remain stable

VII. Post-Layout Simulation Results & Analysis of CLA

The 4-bit CLA desgign layout was implemented in 180nm technology in magic as shown in the figure below. The layout was then extracted and simulated using NGSPICE with the same test conditions as the pre-layout simulation.

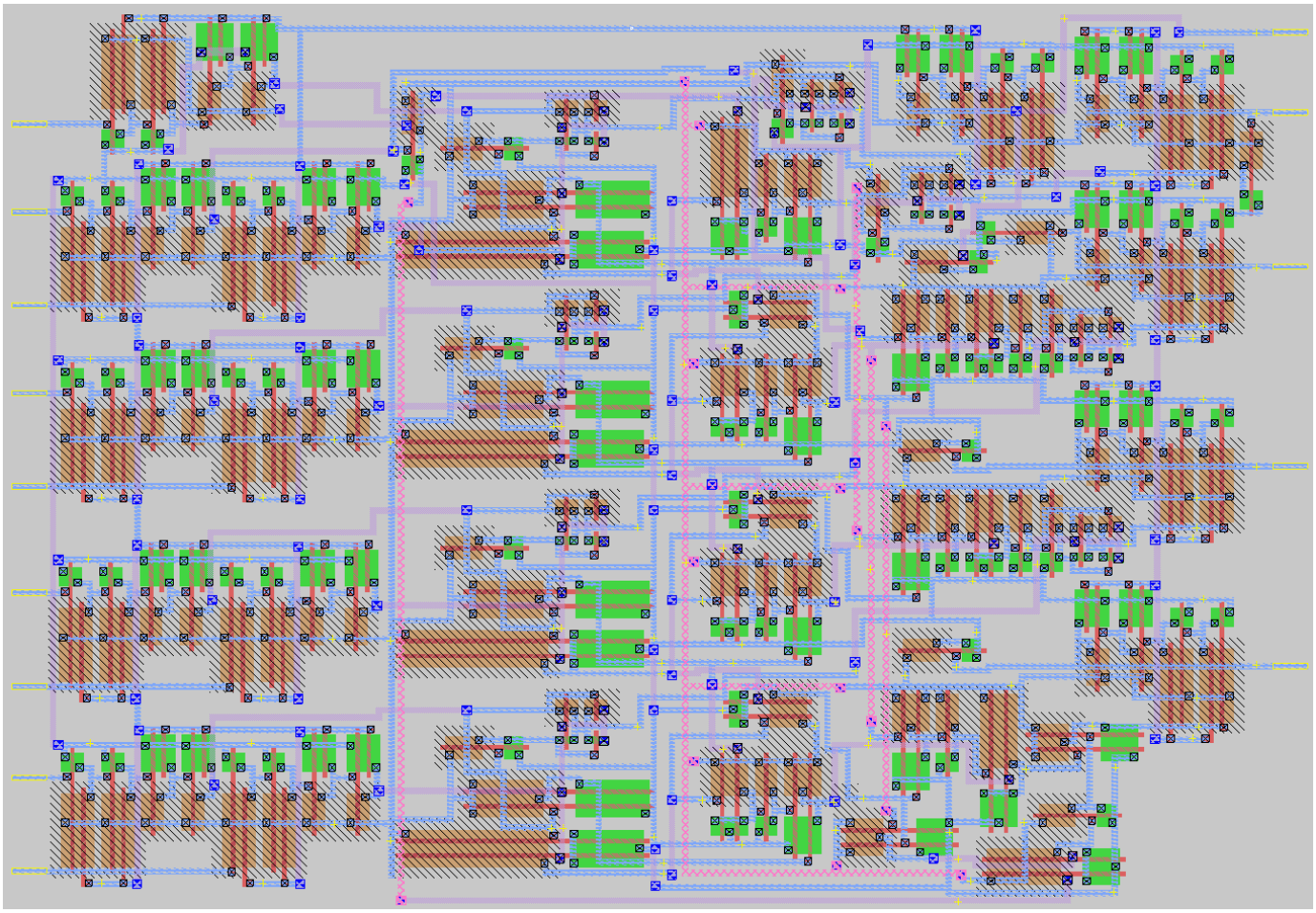

Fig. 18. Magic Layout of 4-bit CLA Design

A. *Verification of functionality*

The input and output specifications are same as mentioned previously in the prelayout simulation. The post-layout simulation produces the following outputs:

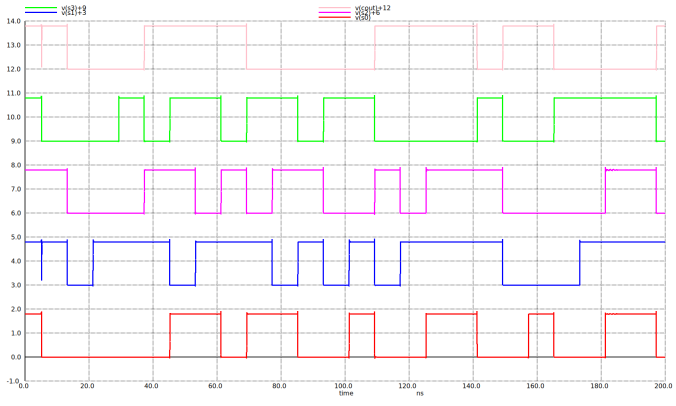

Fig. 19. Input Waveform

The intial outputs are different from the prelayout simulation because of the initial conditions of the flip-flops. After few clock cycles, the outputs stabilize and matches exactly with the prelayout simulation hence the correct operation of the CLA is verified.

B. *Worst Case Delay of CLA Adder Block*

The worst case delay of the post-layout 4-bit CLA design is calculated using the same inputs as in the pre-layout simulation. The simulation results for the worst case delay are as follows:

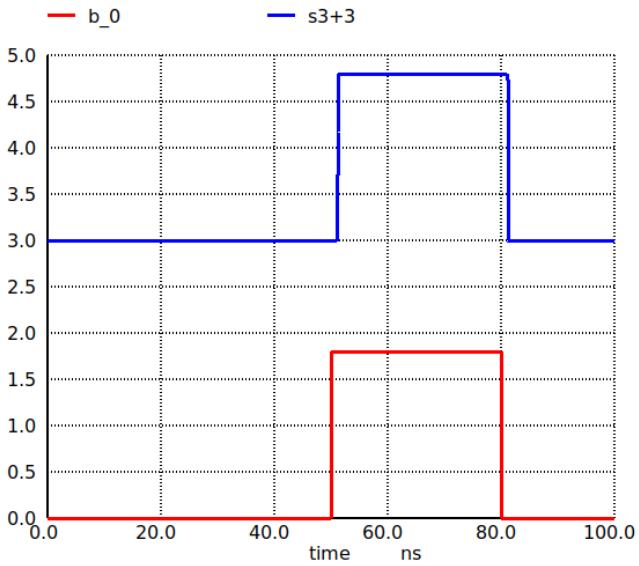

Fig. 20. WaveForms for the Worst Case Delay

```
   Measurements for Transient Analysis

rise_delay          =  1.093636e-09 targ=  5.111864e-08 trig=  5.002500e-08
fall_delay          =  1.131605e-09 targ=  8.120661e-08 trig=  8.007500e-08
propagation_delay   =  1.11262e-09
```

Fig. 21. Worst Case Delay Terminal Output

The simulation results show that the worst case delay for the post-layout 4-bit CLA design is 1112.62ps. This delay is observed in the carry propagation path from B_0 through C3 to S3.

C. Maximum Clock Speed

The maximum clock speed of the post-layout 4-bit CLA design is calculated using the same method as in the pre-layout simulation. Given the values we calculated earlier:

- $t_{CQ_1} = 53.15$ ps
- $t_{pd} = 1112.62$ ps
- $t_{su} = 67.43$ ps

The maximum clock frequency is:

$$f_{max} = \frac{1}{53.15 \text{ ps} + 1112.62 \text{ ps} + 67.43 \text{ ps}} \approx 0.81 \text{ GHz} \tag{18}$$

TABLE VII
COMPARISON OF PRE AND POST LAYOUT SIMULATION RESULTS

Parameter	Pre-Layout	Post-Layout
Worst Case Delay (ps)	479.16	1112.62
Maximum Clock Frequency (GHz)	1.59	0.81

VIII. FLOOR PLAN FOR COMPLETE CIRCUIT LAYOUT

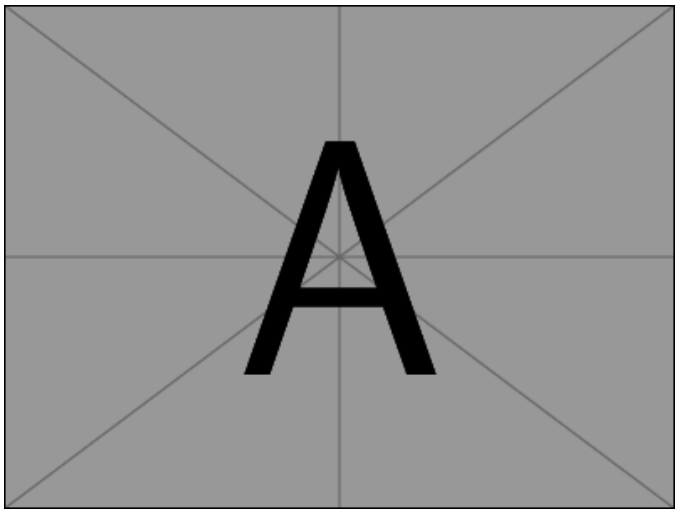

Fig. 22. Floor Plan for Complete Circuit Layout

```
.layout1> b
Root cell box:
          width x height  (   llx,  lly  ), (   urx,  ury  )  area (units^2)

microns:   61.56 x 43.83   (-30.51, -42.66), ( 31.05,  1.17 )  2698.18
lambda:      684 x 487     (  -339, -474  ), (   345,  13   )  333108
```

Fig. 23. Horizontal and Vertical pitch

From the above figure data, the area of the complete circuit layout is calculated as follows:

$$\begin{aligned} \text{Area} &= \text{Horizontal Pitch} \times \text{Vertical Pitch} \\ &= 61.56\mu m \times 43.83\mu m \\ &= 2698.18\mu m^2 \end{aligned}$$

IX. VERILOG HDL SIMULATION RESULTS

This GTKWave plot illustrates the simulation of a 4-bit Carry-Lookahead Adder (CLA). The inputs are:
- $A[3:0]$: A 4-bit binary number. - $B[3:0]$: Another 4-bit binary number. - Cin: The carry-in input.

The outputs are:
- $S[3:0]$: The 4-bit sum. - Cout: The carry-out signal.

For example, at $t = 0$ ns:

$$A = 8\,(1000_2),\ B = 7\,(0111_2),\ \text{Cin} = 1.$$

The resulting sum is:

$$S = A + B + \text{Cin} = 8 + 7 + 1 = 16\,(10000_2).$$

This gives:

$$S[3:0] = 0 \quad \text{(lower 4 bits)}, \quad \text{Cout} = 1 \quad \text{(carry-out)}.$$

As the simulation progresses, changes in A and B update S and Cout efficiently, demonstrating the CLA's ability to compute the carry quickly without ripple propagation.

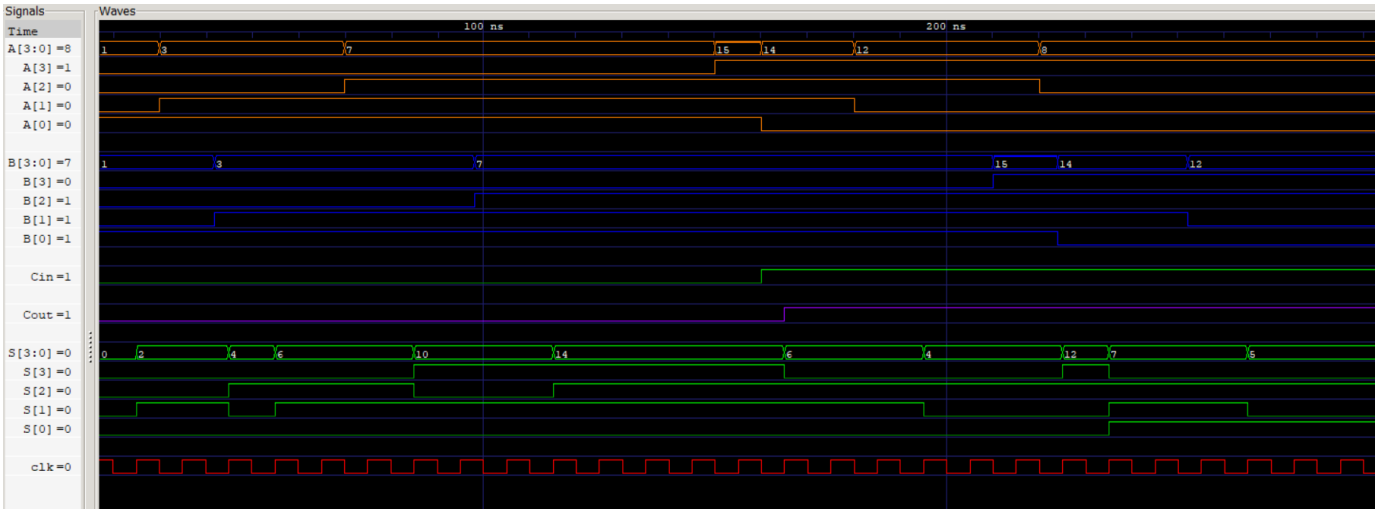

Fig. 24. Verilog HDL Simulation Waveforms

X. Implementaion on FPGA board and oscilloscope

The images below show the output of FPGA board burned with the verilog code of 4-bit CLA design plotted in oscilloscope.

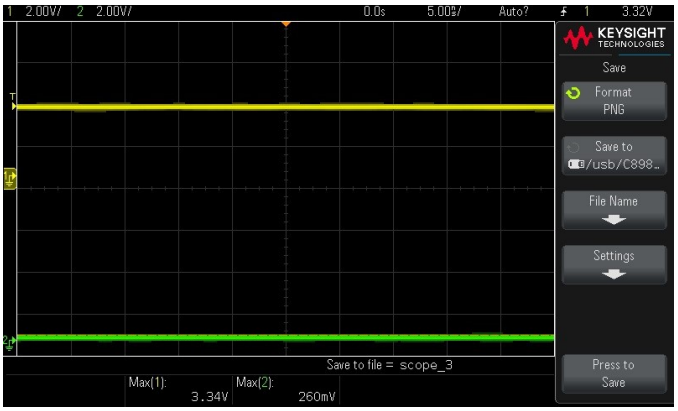

Fig. 25. Oscilloscope Output1 of 4-bit CLA

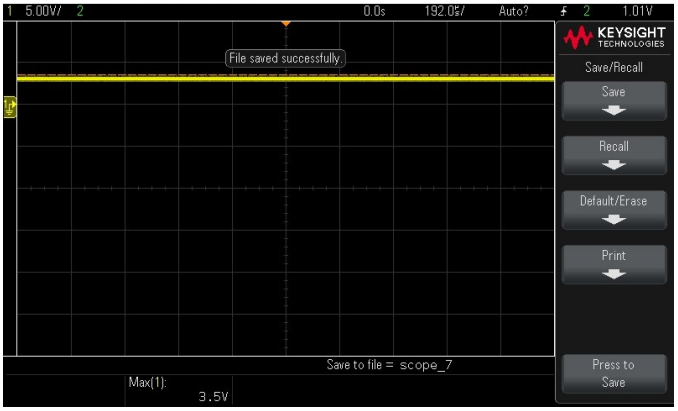

Fig. 26. Oscilloscope Output2 of 4-bit CLA

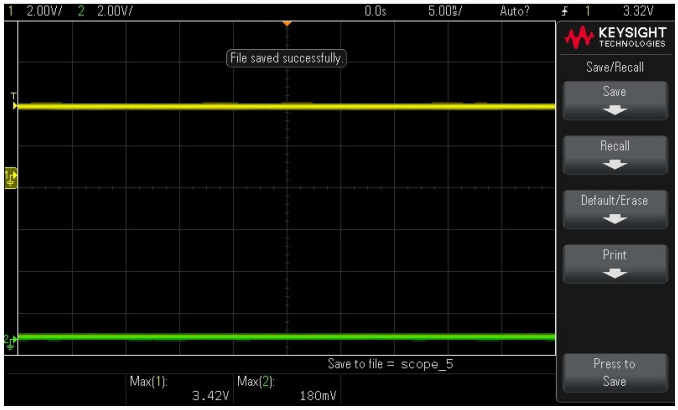

Fig. 27. Oscilloscope Output3 of 4-bit CLA

References

[1] Author(s), "A Novel Implementation of 4-Bit Carry Look-Ahead Adder," Certified by IEEE PDFeXpress, 2017.