CARRY LOOK AHEAD ADDER

Radheshyam Modampuri
International Institute of Information Technology, Hyderabad
radheshyam.modampuri@research.iiit.ac.in
2023102032

***Abstract*—This paper presents an overview of the design, implementation, and analysis of a Carry Look Ahead Adder. The study focuses on the efficiency of the adder in terms of speed and resource utilization compared to conventional adders. The implementation demonstrates reduced propagation delay and increased performance in arithmetic computations. This work serves as a foundational step toward optimized VLSI-based arithmetic unit designs.**

I. Introduction

The introduction will describe the motivation, objectives, and key aspects of the Carry Look Ahead Adder project. Further subsections will elaborate on the background, challenges, and the design methodology adopted during the project lifecycle.

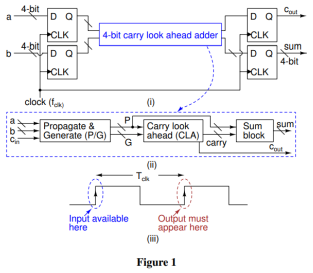

Fig. 1. Overall circuit

II. Design Methodology

A. Overview of CLA Adder Architecture

The CLA adder computes carry signals in parallel, reducing delay compared to ripple-carry adders. The design consists of three main modules: the Propagate and Generate (P/G) module, the Carry Look-Ahead (CLA) module, and the Sum Block. D flip-flops are used to synchronize the inputs and outputs.

B. Propagate and Generate Module

The propagate ($p_i = a_i \oplus b_i$) and generate ($g_i = a_i \cdot b_i$) signals for each bit are calculated here. These signals form the basis for the CLA module, which computes carries for each bit.

$$p_i = a_i \oplus b_i, \tag{1}$$

$$g_i = a_i \cdot b_i. \tag{2}$$

$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 C_1$$

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:

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

C. Carry Look-Ahead Module

The CLA module uses p_i, g_i, and carry-in values to calculate carry-out signals in parallel. The carry-out for each bit is computed as:

$$c(i+1) = g_i + (p_i \cdot c_i) \tag{3}$$

a_i	b_i	$p_i = a_i \oplus b_i$	$g_i = a_i \wedge b_i$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

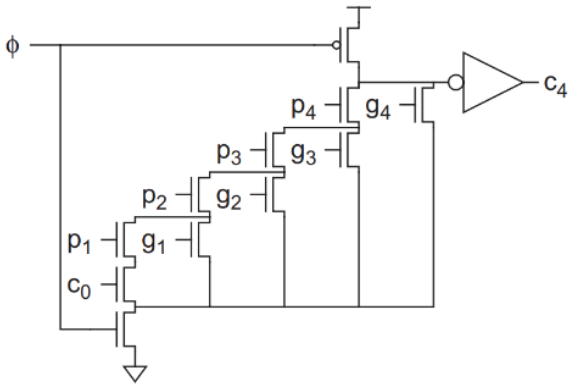

Fig. 2. C4

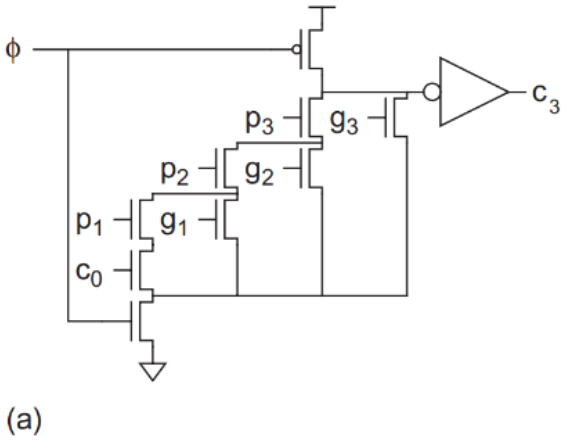

Fig. 3. C3

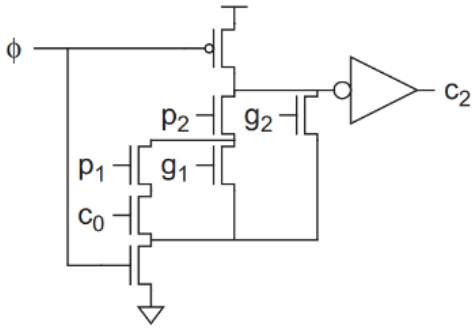

Fig. 4. C2

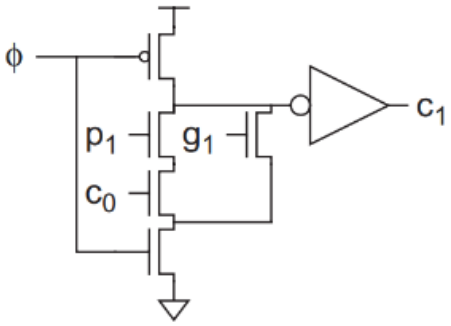

Fig. 5. C1

This design significantly reduces the delay encountered in conventional adders.

$$c_1 = g_1 + p_1 c_0$$
$$c_2 = g_2 + p_2(g_1 + p_1 c_0)$$
$$c_3 = g_3 + p_3(g_2 + p_2(g_1 + p_1 c_0))$$
$$c_4 = g_4 + p_4(g_3 + p_3(g_2 + p_2(g_1 + p_1 c_0)))$$

Fig. 6. recurtion

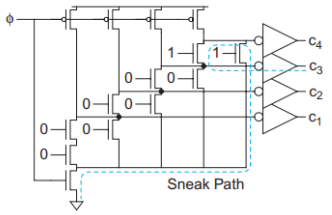

FIGURE 9.45 Sneak path

Fig. 7. sneaky path XOR preferd over OR gate

D. Sum Block

The sum for each bit position is calculated using:

$$s_i = p_i \oplus c_i \tag{4}$$

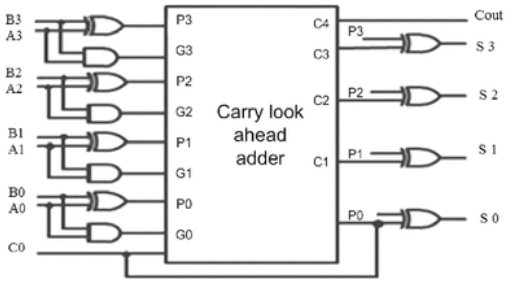

Fig. 8. Over all sum block

III. Design Topology

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 $2 * LAMBDA$

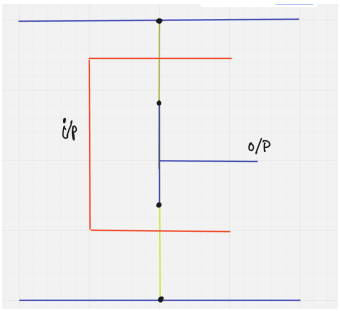

Fig. 10. Stick diagram

A. Inverter

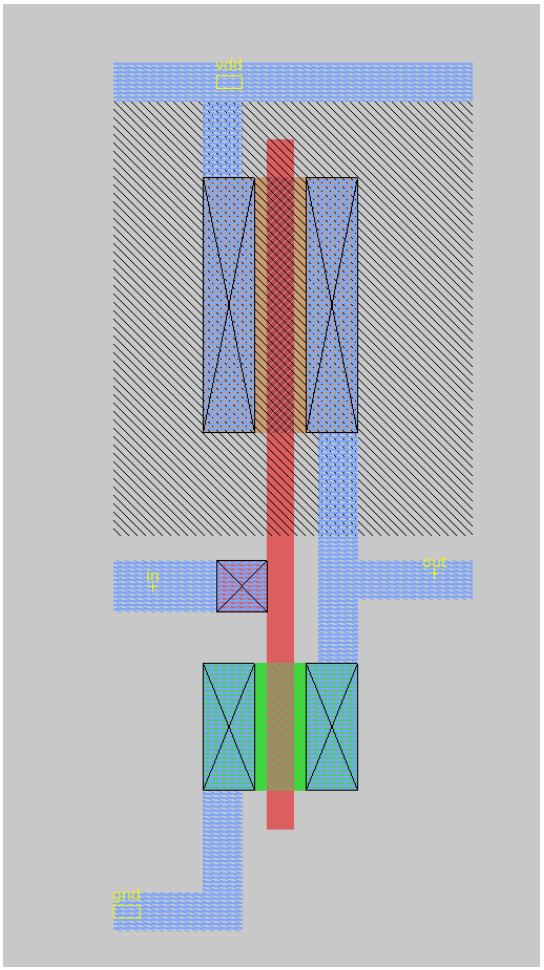

Fig. 9. inverter

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.

B. And

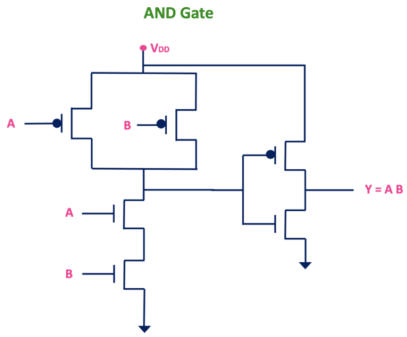

Fig. 11. And Gate CMOS

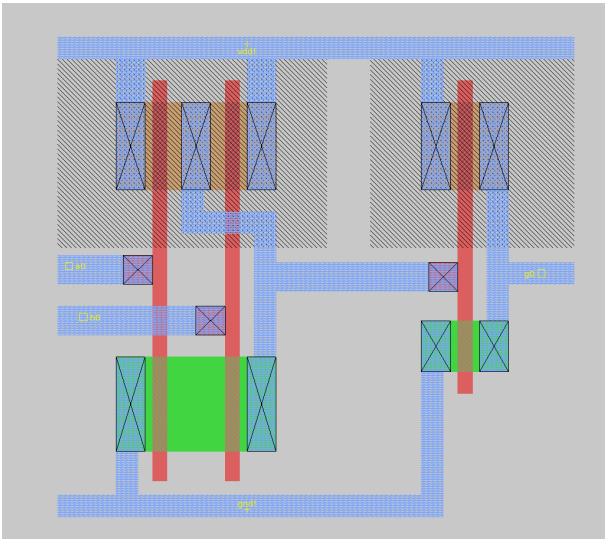

Fig. 12. Magic And

I used And gate with cmos static logic i.e A nand gate connected to inverter.The ratio of nand gate $\frac{W_p}{W_n}$ is 1.The inverter has same size as above.

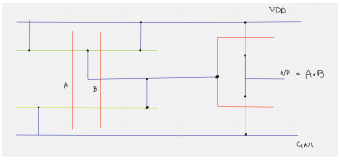

Fig. 13. Stick diagram

C. Xor

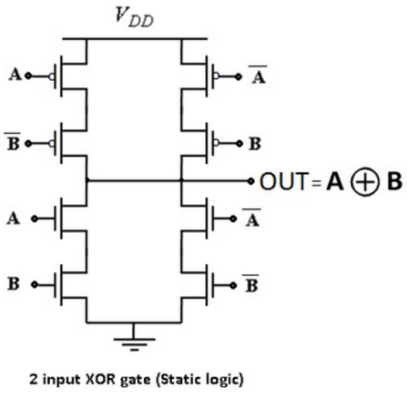

Fig. 14. Xor Gate CMOS

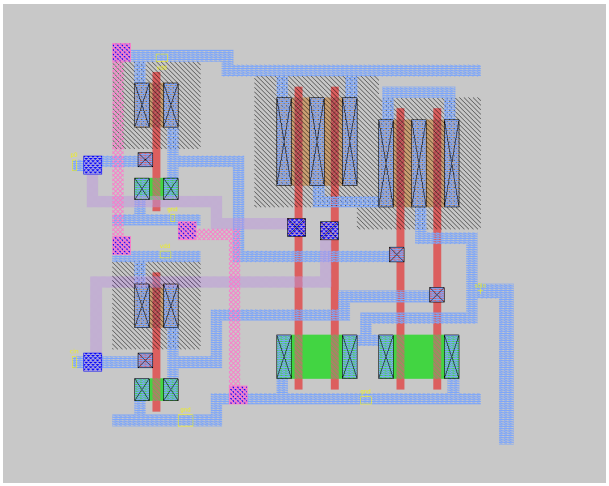

Fig. 15. Magic

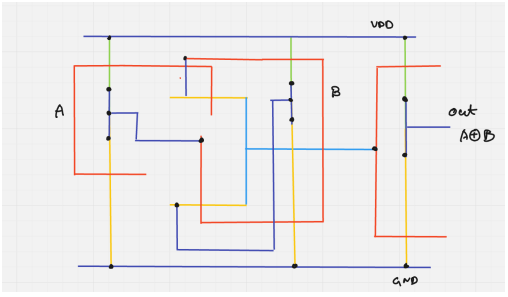

Fig. 16. Stick diagram

I used Xor of Cmos static logic.Sizings are $\frac{W_p}{W_n}$ is 1 to match with inverter delay.

D. D flipflop

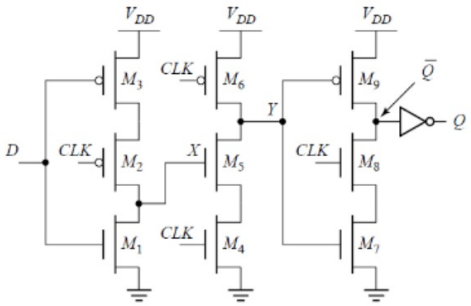

Fig. 17. TSPC FF

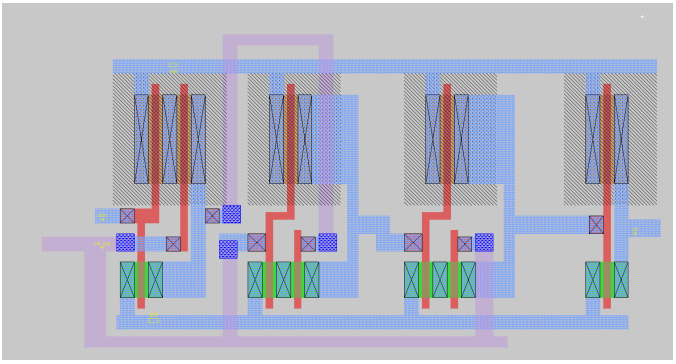

Fig. 18. Magic

- The D Flip-Flop is implemented using True Single-Phase Clocking (TSPC) logic, which dynamically stores and transfers data with reduced transistor count and a single clock phase.
- This design uses 11 MOSFETs instead of the traditional 14 by optimizing the logic as follows:
  - The design combines pull-up and pull-down paths efficiently, using fewer transistors while maintaining functionality.
  - A clocked NMOS network is used for data capture, and the stored data is dynamically transferred to the output through successive stages.
- The sizing of transistors:
  - NMOS and PMOS transistor widths are chosen to ensure proper rise and fall times. Typical sizing:
    - NMOS transistors: $W_n = W$.
    - PMOS transistors: $W_p = 2W$.
- Advantages of this TSPC implementation:
  - Reduced power consumption due to fewer transistors and dynamic operation.
  - Simplified clocking scheme with a single-phase clock, reducing clock skew issues.
  - Compact design with fewer MOSFETs, saving area.

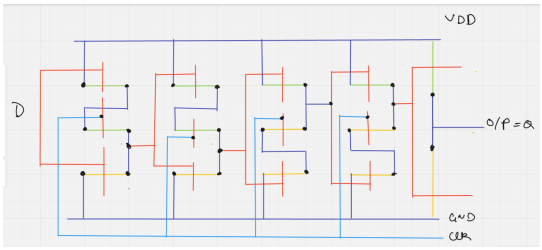

Fig. 19. Stick Diagram

- Functionality

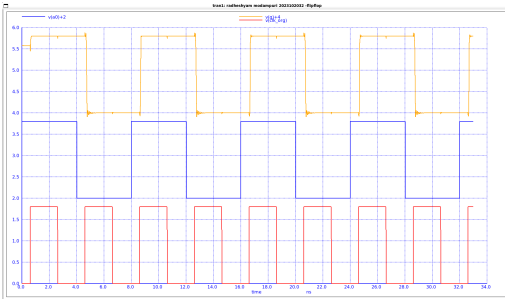

Fig. 20. Dff prelayout

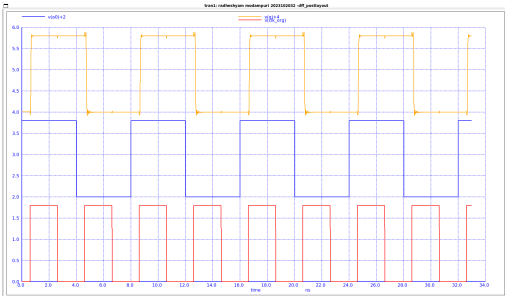

Fig. 21. Dff postLaout

- Delay of D Flip flop

```
tsetup          =  7.100000e-11 targ=  1.805000e-09 trig=  1.734000e-09
```

Fig. 22. Tsetup prelayout

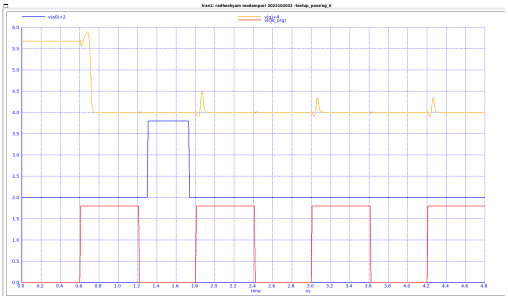

Fig. 23. Tsetup prelayout

```
thold           =  2.230000e-11 targ=  1.827300e-09 trig=  1.805000e-09
```

Fig. 24. Thold prelayout

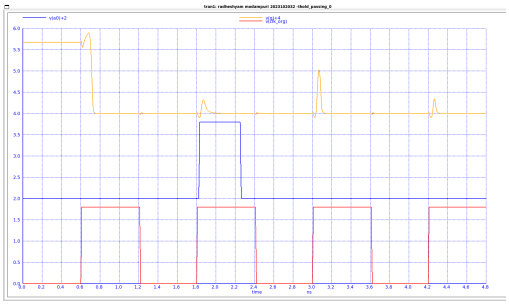

Fig. 25. Thold prelayout

```
tpcq_0_to_1     =  6.275301e-11 targ=  1.867753e-09 trig=  1.805000e-09
```

Fig. 26. Tpcq prelayout

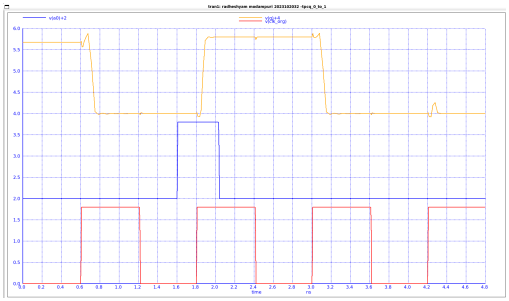

Fig. 27. Tpcq prelayout

```
tsetup          =  5.300000e-11 targ=  1.805000e-09 trig=  1.752000e-09
```

Fig. 28. Tsetup postlayout

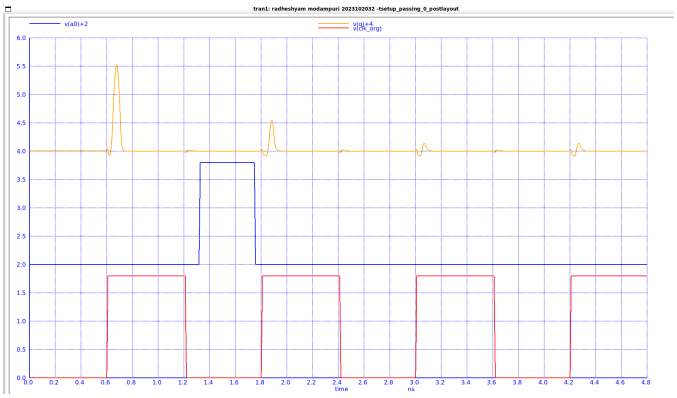

Fig. 29. Tsetup postlayout

```
thold                =  2.290000e-11 targ=  1.827900e-09 trig=  1.805000e-09
```

Fig. 30. Thold postlayout

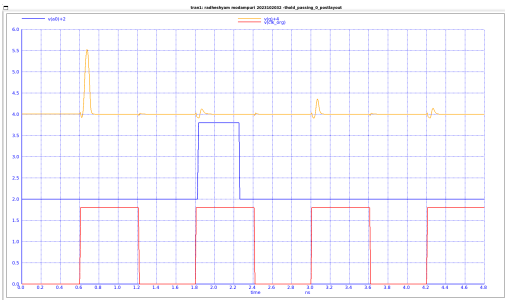

Fig. 31. Thold postlayout

```
tpcq_0_to_1          =  7.660507e-11 targ=  1.881605e-09 trig=  1.805000e-09
```

Fig. 32. Tpcq postlayout

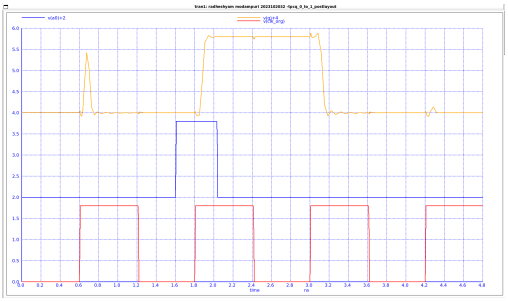

Fig. 33. Tpcq postlayout

E. Cla Block

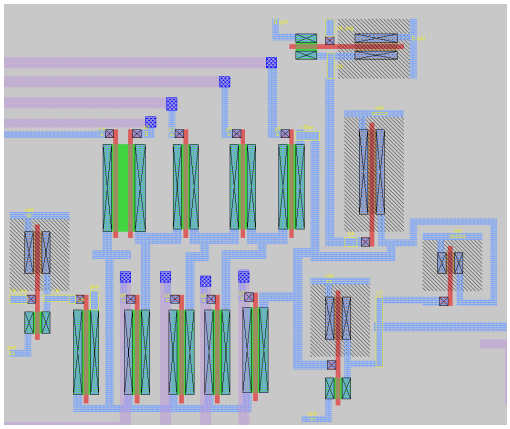

Fig. 34. C3

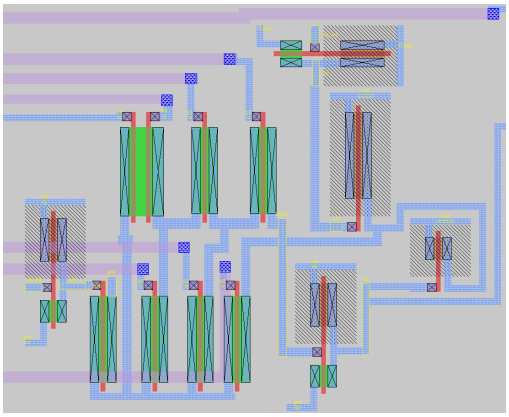

Fig. 35. C2

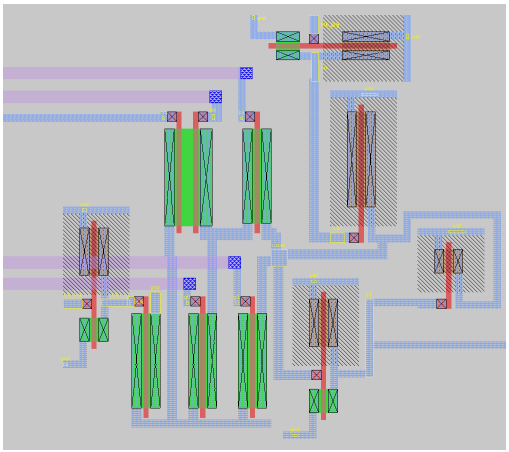

Fig. 36. C1

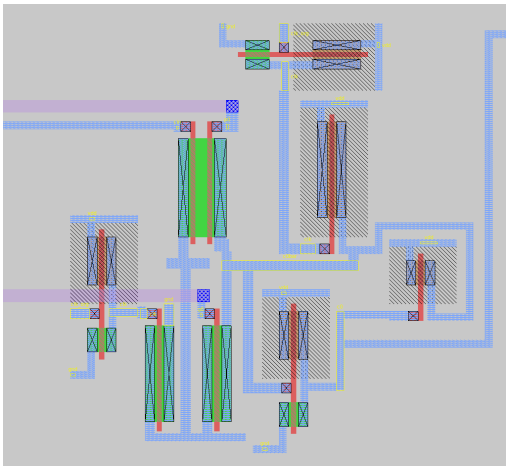

Fig. 37. C0

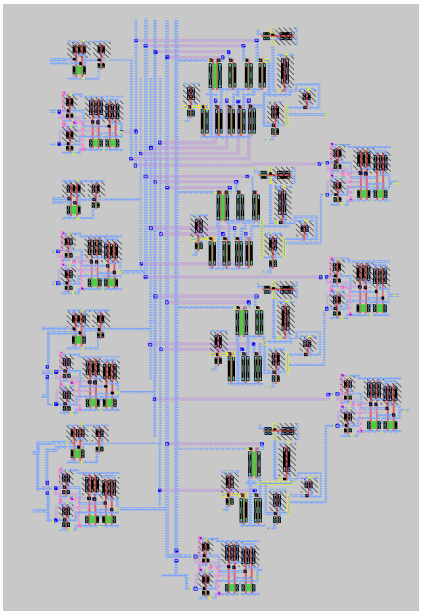

Fig. 38. CLA routed

Advantages

- High-speed operation due to parallel carry computation and domino logic.
- Reduced gate count compared to static CMOS designs.
- Efficient area utilization in high-performance applications.
- Lower power consumption in short or less frequently used paths.
- Scalable design for larger bit-widths.

Disadvantages

- High power consumption due to precharge and evaluation phases.
- Complex design requiring careful clocking and timing considerations.
- Susceptibility to noise due to dynamic nodes.
- Clock synchronization issues, prone to clock skew.

```
tpd_max              =  3.681120e-10 targ=  2.044311e-08 trig=  2.007500e-08
```

Fig. 39. Tpd max prelayout

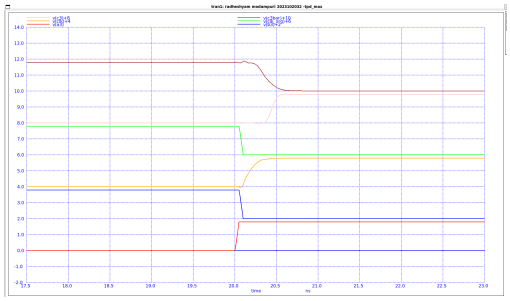

Fig. 40. Tpd max prelayout

```
tpd_max              =  2.978544e-10 targ=  2.037285e-08 trig=  2.007500e-08
```

Fig. 41. Tpd max postlayout

Fig. 42. Tpd max postlayout

IV. Overall Circuit

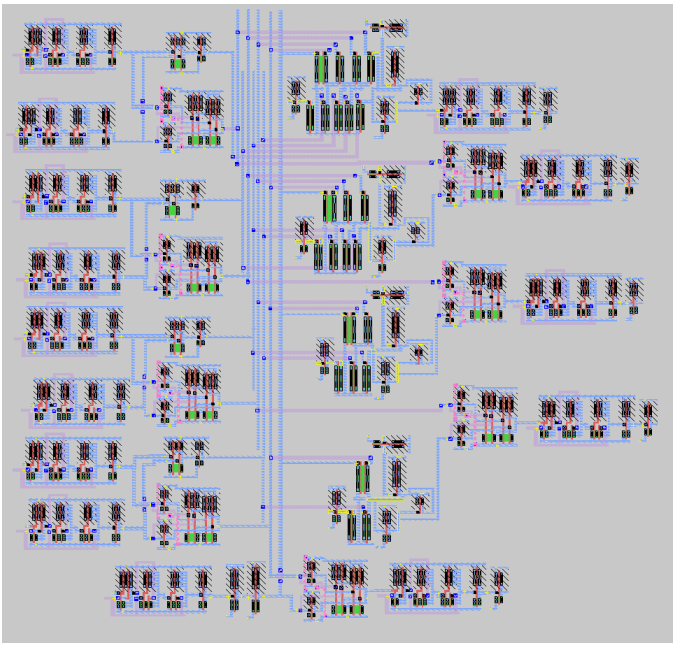

Fig. 43. over all circuit

V. Overall Functionality

```
.param Ton=10n
.param Tperiod={2*Ton}


V1 a0in 0 pulse(0 1.8 0 10p 10p {2*Ton} {4*Ton})
V2 a1in 0 pulse(0 1.8 0 10p 10p {3*Ton} {6*Ton})
V3 a2in 0 pulse(0 1.8 0 10p 10p {4*Ton} {8*Ton})
V4 a3in 0 pulse(0 1.8 0 10p 10p {5*Ton} {10*Ton})
V5 b0in 0  pulse(0 1.8 0 10p 10p {6*Ton} {12*Ton})
V6 b1in 0  pulse(0 1.8 0 10p 10p {7*Ton} {14*Ton})
V7 b2in 0  pulse(0 1.8 0 10p 10p {8*Ton} {16*Ton})
V8 b3in 0  pulse(0 1.8 0 10p 10p {9*Ton} {18*Ton})
V9 cinin 0 0

V_clk_org clk_org 0 pulse(0 1.8 {0.3*Ton} 10p 10p {Ton} {Tperiod})
```

Fig. 44. Inputs given

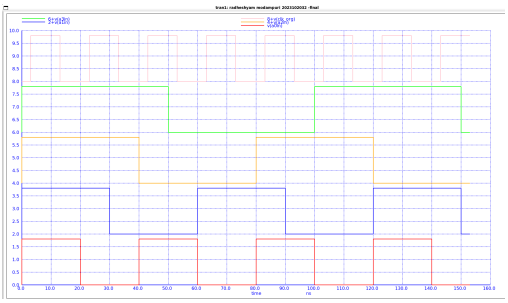

Fig. 45. Ai inputs

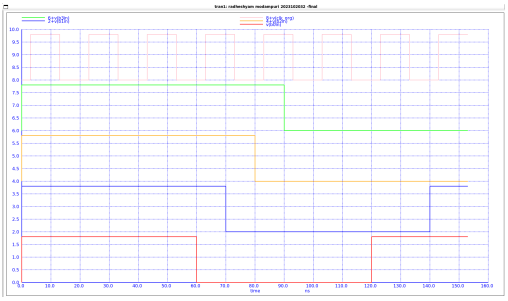

Fig. 46. Bi inputs

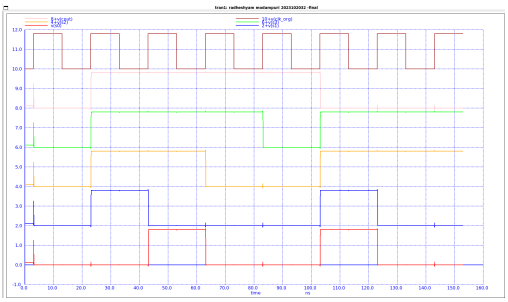

Fig. 47. Si prelayout

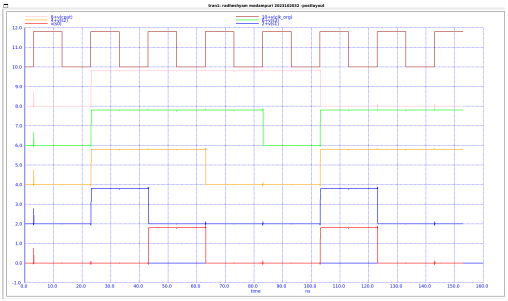

Fig. 48. Si postlayout

Over all 7 test cases

VI. Frequncy Max

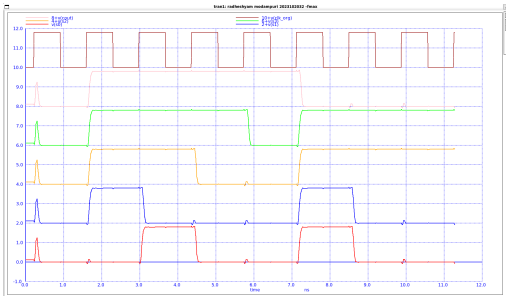

Fig. 49. Tclkmin=1.38ns

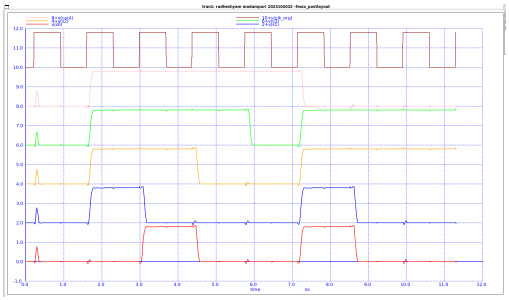

Fig. 50. Tclkmin=1.384ns

VII. Floor plan of four bit adder

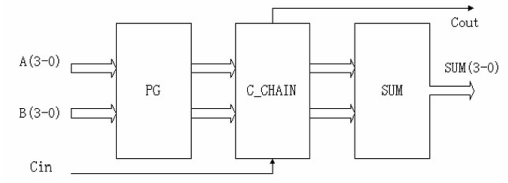

Fig. 51. floorplaning

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

VIII. Horizontital and vertical pitch

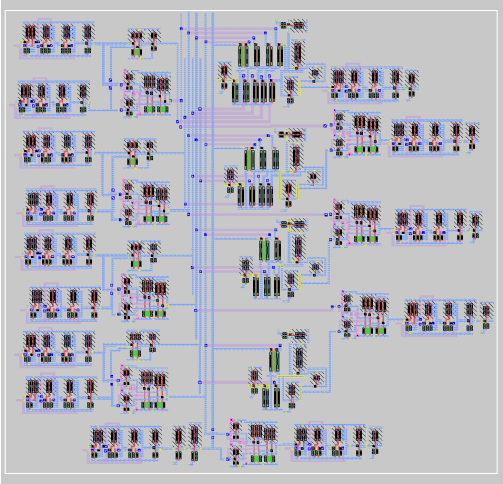

Fig. 52. Box selected

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

microns:   95.13 x 89.37   ( 143.01, -7.74 ), ( 238.14,  81.63)  8501.77
lambda:     1057 x 993     (  1589, -86   ), (  2646,  907  )  1049601
```

Fig. 53. horizontal and vertical pitch

IX. Pre and post layout comparision

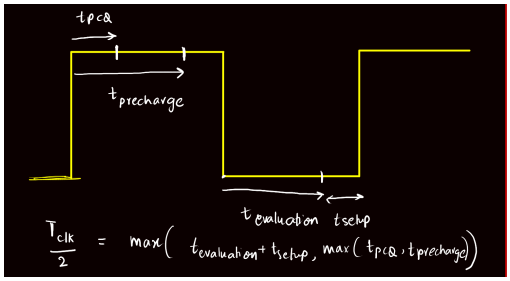

Fig. 54. Timeing diagram

Category	Parameter	Pre	Post
And Gate	$t_{\text{pd max}}$	$0.057ns$	$0.061ns$
Xor Gate	$t_{\text{pd max}}$	$0.06ns$	$0.0415ns$
D Flip-Flop	t_{setup}	$0.1ns$	$0.058ns$
	t_{hold}	$27.8ps$	$24ps$
	t_{pcq} $(0 \rightarrow 1)$	$68.09ps$	$76.6ps$
	t_{pcq} $(1 \rightarrow 0)$	$0.13ns$	$0.126ns$
Sum Block without Xor	$t_{\text{pd max}}$	$0.4399ns$	$0.4535ns$
Minimum Clock Period	T_{clk}	$1.1998ns$	$1.09ns$
Clock Frequency	T_{freq}	$833MHz$	$917MHz$

TABLE I
Pre and Post Timing Parameters

X. Verilog

A. outputs of verilog

```
a =  4, b =  1,cin= 1, Sum (c) =  x
a =  3, b = 13,cin= 1, Sum (c) =  6
a =  5, b =  2,cin= 1, Sum (c) = 17
a = 13, b =  6,cin= 1, Sum (c) =  8
a = 13, b = 12,cin= 1, Sum (c) = 20
a =  6, b =  5,cin= 0, Sum (c) = 26
a =  5, b =  7,cin= 0, Sum (c) = 11
a = 15, b =  2,cin= 0, Sum (c) = 12
a =  8, b =  5,cin= 0, Sum (c) = 17
a = 13, b = 13,cin= 1, Sum (c) = 13
[Done] exit with code=0 in 0.076 seconds
```

Fig. 55. veriolog output

Input given 13+8+1

Fig. 56. Gcd wave form

XI. FPGA

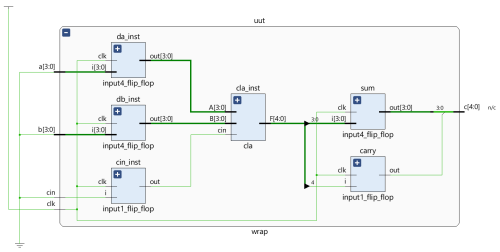

Fig. 57. simulation skematic

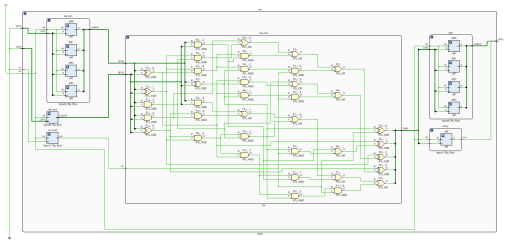

Fig. 58. Simulation skematic 2

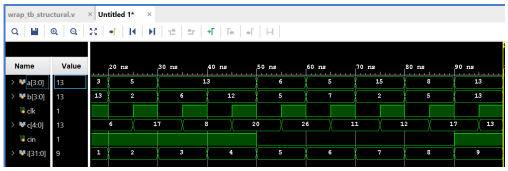

Fig. 59. Simulation

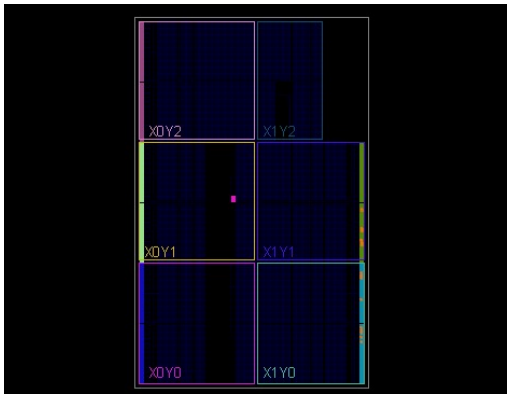

Fig. 60. FPGA using muxes

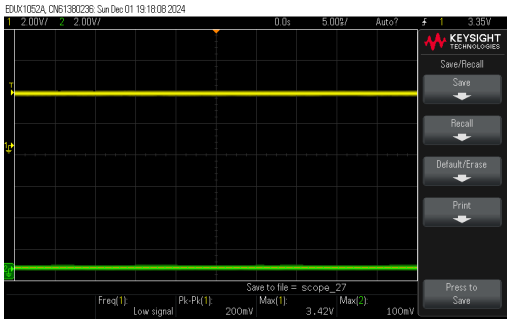

Fig. 61. cout s4

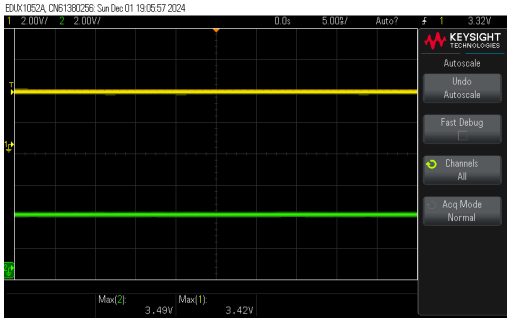

Fig. 62. s3 s2

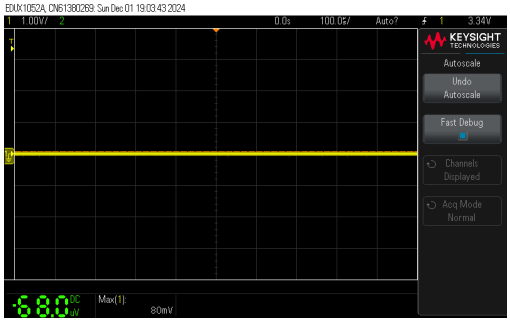

Fig. 63. s0

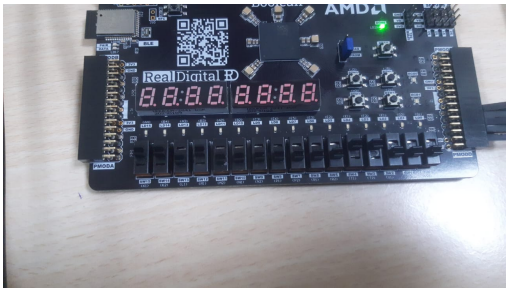

Fig. 64. inputs in FPGA

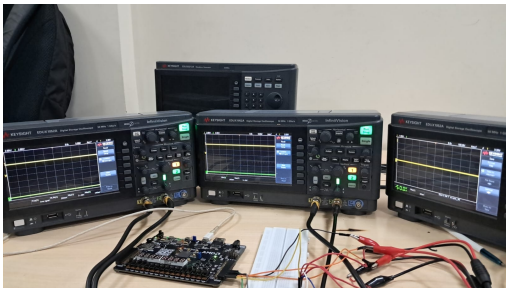

Fig. 65. cout s3 s2 s1

- $A = 1101, B = 1000, C_{\text{in}} = 1$: The sum is 0110 and $C_{\text{out}} = 1$, as the result exceeds 4 bits.
- $A = 1010, B = 0101, C_{\text{in}} = 1$: The sum is 0000 and $C_{\text{out}} = 1$.

XII. Conclusion

This project demonstrated the effectiveness of the Carry Look-Ahead Adder design in achieving high-speed addition in a 4-bit architecture. The use of CMOS technology in MAGIC layout and FPGA implementation confirmed the adder's performance. Future work could include extending the design for larger bit-widths or optimizing power consumption.

References

[1] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Design Perspective*, 2nd ed. Prentice Hall, 2002.
[2] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. Addison-Wesley, 2010.
[3] M. Mano, *Digital Logic and Computer Design*. Prentice Hall, 1979.

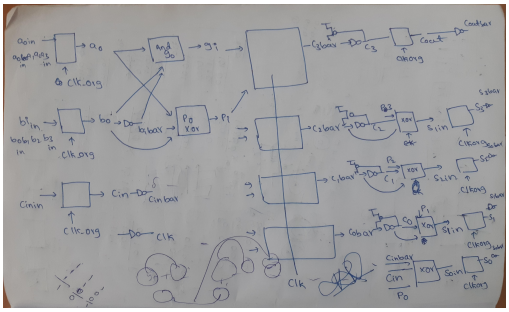

Fig. 66. nodes used in code

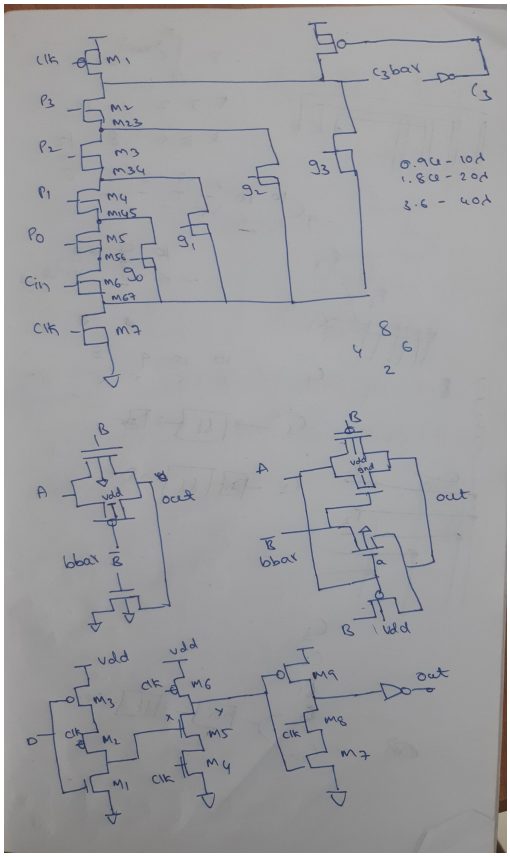

Fig. 67. nodes used