
International Institute of Information Technology
Hyderabad

# VLSI Design Project

*Course Code: M25EC2*

**November - 25**

**Ravi Kumar Sahu**

Student ID: 2024102024

# Contents

**Abstract**

**The Carry Look-Ahead (CLA) Adder is designed and implemented using 180nm CMOS technology, reducing propagation delay through hierarchical blocks. Validated using NGSPICE simulations and MAGIC layouts, the design is verified post-layout. A Verilog-based structural description enables FPGA implementation for hardware validation. Key performance metrics, including delay, maximum frequency, and power efficiency, demonstrate the CLA adder's high-speed and reliable operation. Keywords—CLA, NGSPICE, MAGIC, Delay, Verilog, FPGA**

# I Introduction

The Carry Look-Ahead Adder (CLA) is an improved version of the Ripple Carry Adder (RPA). The primary disadvantage of the RPA is the carry propagation delay, where the most significant bit must wait for the carry signal to ripple through all previous stages. In this 5-bit design, the input bits are defined as $A_i$ and $B_i$ (where $i = 0$ to 4) along with an initial carry $C_0$. The CLA solves the delay problem by pre-computing carries in parallel rather than waiting for the sequential ripple effect.

Sequential dependency is effectively removed by computing the Propagate ($P_i$) and Generate ($G_i$) signals for each bit independently. The fundamental logic for these signals is defined as:

$$P_i = A_i \oplus B_i \tag{1}$$

$$G_i = A_i \cdot B_i \tag{2}$$

The sum bits ($S_i$) are subsequently calculated as:

$$S_i = P_i \oplus C_i \tag{3}$$

Positive edge-triggered flip-flops are used to provide stable inputs and capture the outputs. If input bits are available before the rising edge of the clock, the output is computed and latched at the next rising edge.

# II Concepts CLA

The carry for any stage $i + 1$ is derived from the generate and propagate terms of the previous stage according to the equation $C_{i+1} = G_i + (P_i \cdot C_i)$.

For the implemented 5-Bit CLA, the carry equations are expanded to compute $C_1$ through $C_5$ in parallel. This architecture removes the wait time associated with ripple adders. The expanded boolean equations are:

$$
\begin{aligned}
C_1 &= G_0 + P_0 \cdot C_0 \\
C_2 &= G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_0 \\
C_3 &= G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0 \\
C_4 &= G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 \\
&\quad + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_0 \\
C_5 &= G_4 + P_4 \cdot G_3 + P_4 \cdot P_3 \cdot G_2 + P_4 \cdot P_3 \cdot P_2 \cdot G_1 \\
&\quad + P_4 \cdot P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_4 \cdot P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_0
\end{aligned}
$$

As demonstrated by the equation for $C_5$, the hardware complexity and fan-in requirements increase with the bit-width. The final term requires a 6-input AND gate and the summation requires a 6-input OR gate.

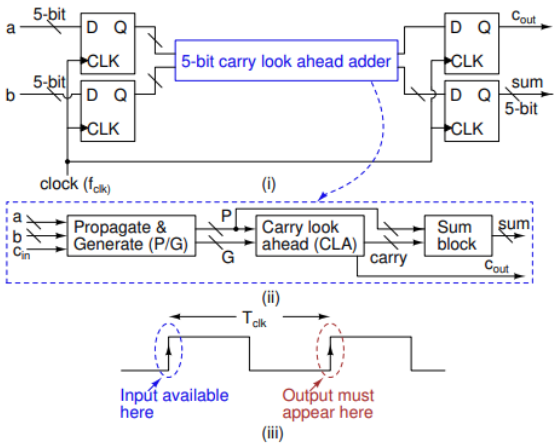


**Figure 1**

Figure 1: figure-noName

# III Logic Gate Requirements

To implement the 5-Bit CLA logic based on the derived equations, the following specific logic gates are required:

- **NOT Gate**
- **XOR Gate:** 2-input (for Sum generation).
- **AND Gates:** 2, 3, 4, 5, and 6-input configurations.
- **OR Gates:** 2, 3, 4, 5, and 6-input configurations.

The inclusion of 6-input AND and OR gates is a specific requirement for the 5-bit design to handle the $C_5$ carry generation term, which involves propagating the carry logic through the five preceding stages (0 to 4).

All logic gates have been designed using the Complementary CMOS logic style. The input and output D flip-flops are designed using True-Single-Phase-Clock (TSPC) topology. The overall design utilizes a 3-stage structure where the $1^{st}$ and $3^{rd}$ stages are flip-flops for signal stability, and the $2^{nd}$ stage is the combinational CLA block.

# IV Topology and Sizing

The circuit is implemented using 180nm technology with a minimum feature size ($\lambda$) of $0.09\mu m$. The sizing strategy aims to balance rise and fall times by compensating for the lower mobility of PMOS carriers compared to NMOS carriers.

## IV.i Logic Gate Sizing

- **NOT Gate:** Standard sizing is used with $W_N = 10\lambda$ and $W_P = 20\lambda$.
- **AND Gates:** Implemented by cascading a NAND gate with an inverter. The NMOS widths in the NAND stack are multiplied by the number of inputs ($N$) to maintain equivalent pull-down strength. For the 5-bit design, this extends up to a 6-input NAND where $W_N = 6 \times W_{ref}$.
- **OR Gates:** Implemented by cascading a NOR gate with an inverter. The PMOS widths in the NOR stack are multiplied by the number of inputs ($N$) to maintain equivalent pull-up strength. For the 6-input case, $W_P = 6 \times W_{ref}$.

## IV.ii Flip-Flop Topology

The D flip-flops utilize the TSPC (True-Single-Phase-Clock) topology. This dynamic logic style is chosen to reduce the transistor count and clock load compared to static CMOS flip-flops.

# V Timing Analysis Theory

The performance of the 5-bit CLA is evaluated based on the following timing parameters:

- **Setup Time ($T_{setup}$):** The minimum time interval before the clock's active edge during which the data input must remain stable. This is determined by changing the D input just before the rising edge of the clock.
- **Hold Time ($T_{hold}$):** The minimum time interval after the clock's active edge during which the data input must remain stable. For the TSPC flip-flop topology used, the theoretical hold time is approximately 0 ps.
- **Clock-to-Q Delay ($T_{cq}$):** The time delay between the active edge of the clock and the valid output appearing at Q.
- **Propagation Delay ($T_{pd}$):** The time required for the combinational logic (CLA block) to produce a valid sum and carry output after the inputs change. This is measured by identifying the maximum rise and fall delays for the worst-case input transitions (e.g., propagating a carry from bit 0 to bit 5).

# VI PreLayout Analysis

## VI.i Inverter

```
  Measurements for Transient Analysis
tpcq_inv            =  1.765958e-11 targ=  5.032660e-09 trig=  5.015000e-09
```

Figure 2: Propagation delay analysis for the CMOS inverter.

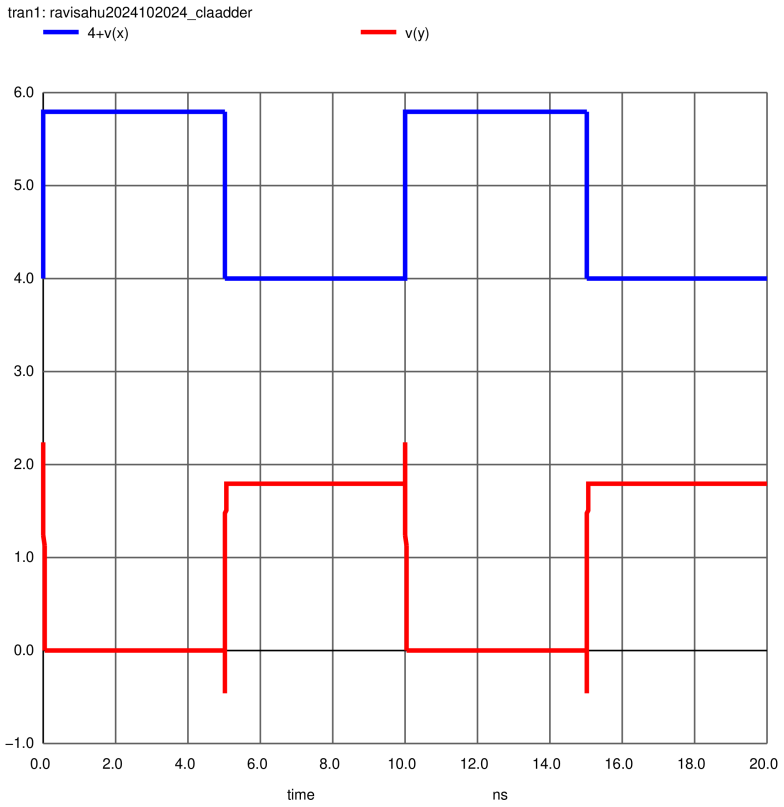


Figure 3: Voltage transfer characteristics (VTC) and transient plot of the inverter.

### VI.i.1 Delay Analysis

The Delay of this Inverter is **17.66ps**

## VI.ii DFF

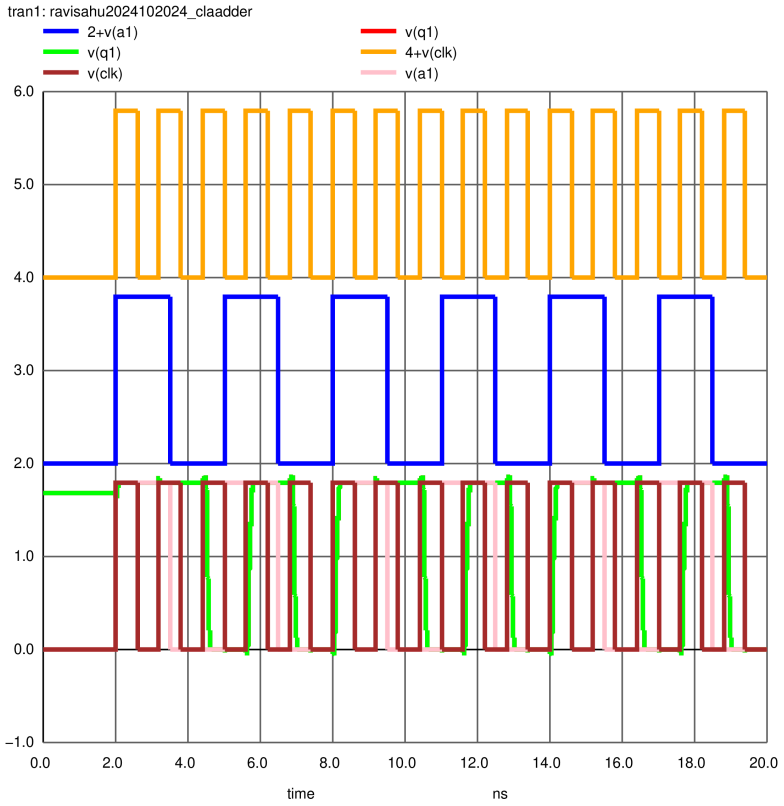


Figure 4: Transient response plot of the D-Flip Flop.

```
  Measurements for Transient Analysis

tpcq_lh            =  2.495996e-09 targ=  5.696496e-09 trig=  3.200500e-09
tpcq_hl            =  1.446968e-10 targ=  4.545197e-09 trig=  4.400500e-09
avg_tpcq           =  1.32035e-09
```

Figure 5: Analysis of the propagation delay for the D-Flip Flop.

### VI.ii.1 Dealy Analysis

| Measurement | Value (ps) |
|---|---|
| tpcq_lh (LH) | **2495.99ps** |
| tpcq_hl (HL) | **144.70ps** |
| avg_tpcq (average) | **1320ps** |

## VI.iii Propagator and Generator

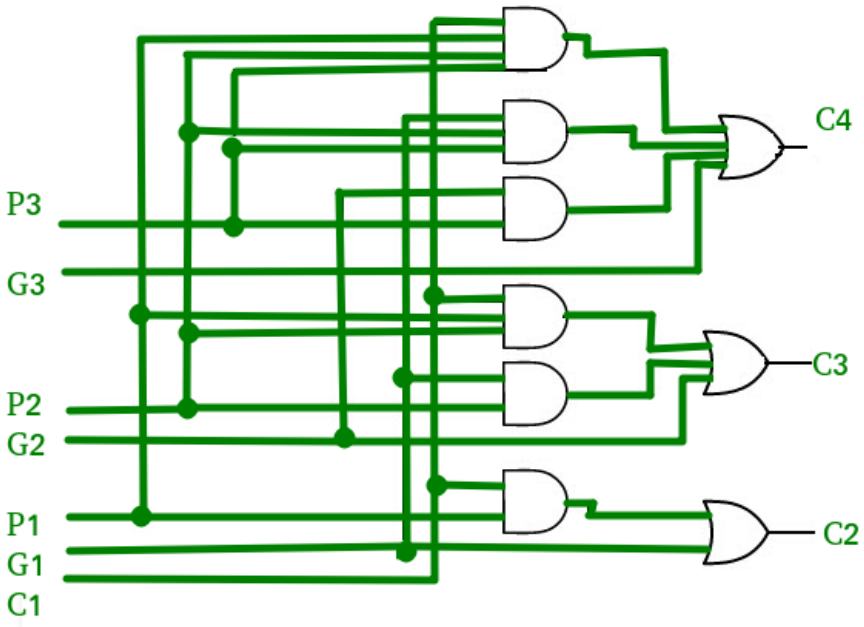


Figure 6: reference diagram

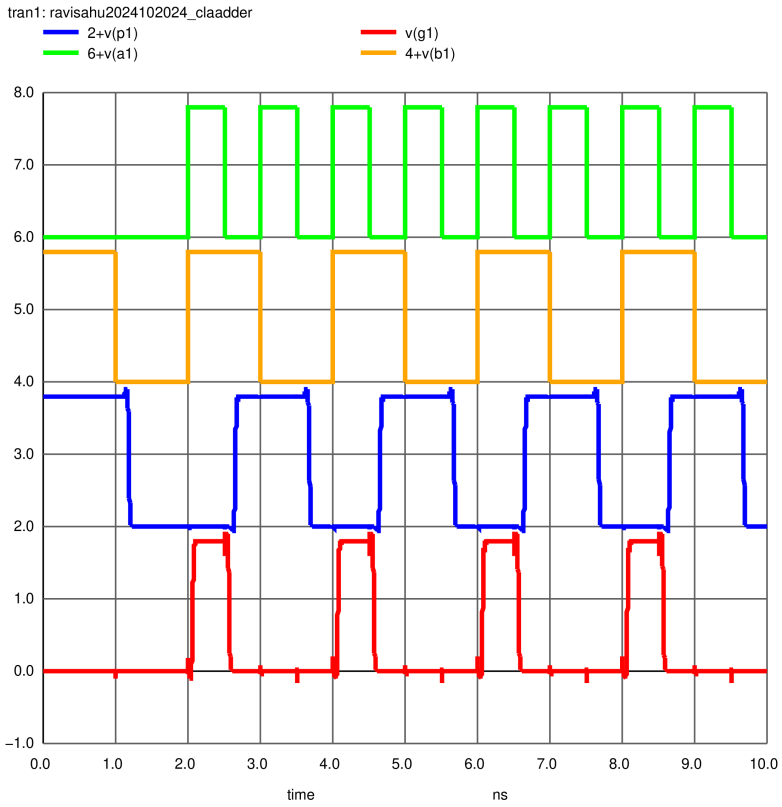


Figure 7: Propagate and Generate signal outputs for Bit 1.

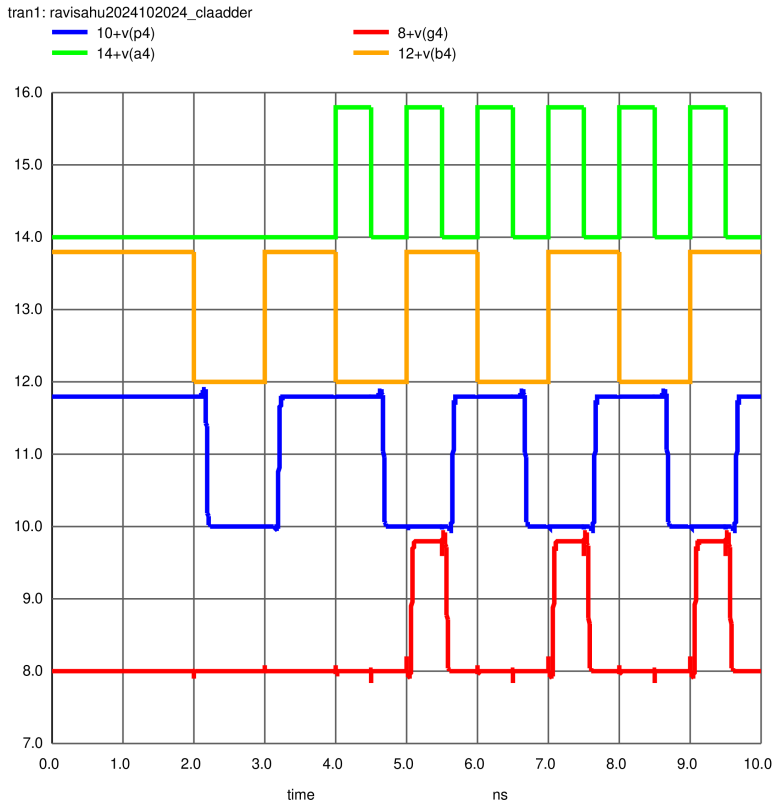


Figure 8: Propagate and Generate signal outputs for Bit 4.

| **Measurement** | **Value (ps)** |
|---|---|
| prop_delay | **154.03ps** |
| gen_delay | **65.13ps** |

## VI.iv CLA Block

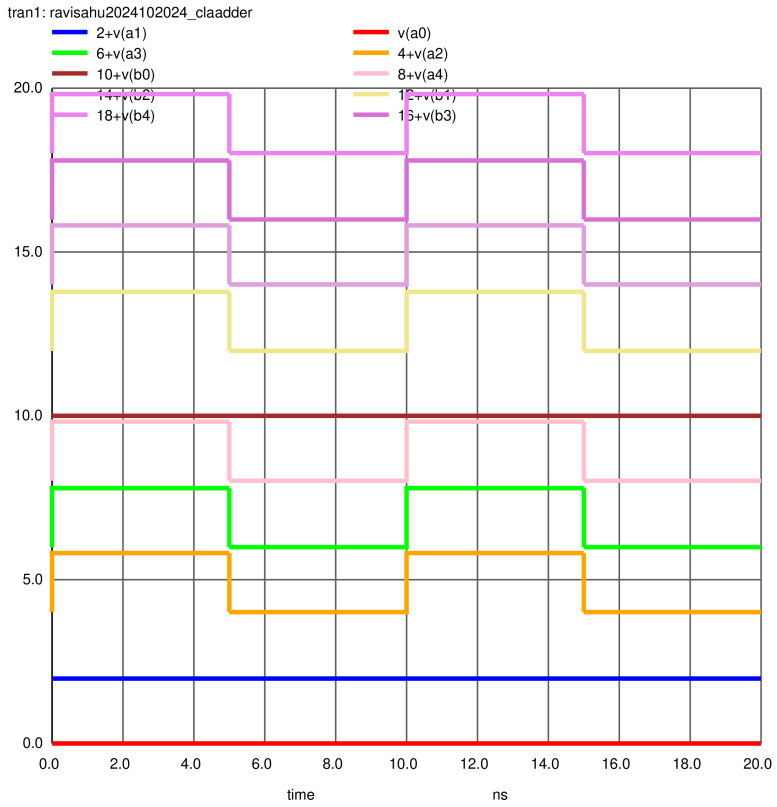


Figure 9: Input signals entering the Carry Look-Ahead (CLA) block.

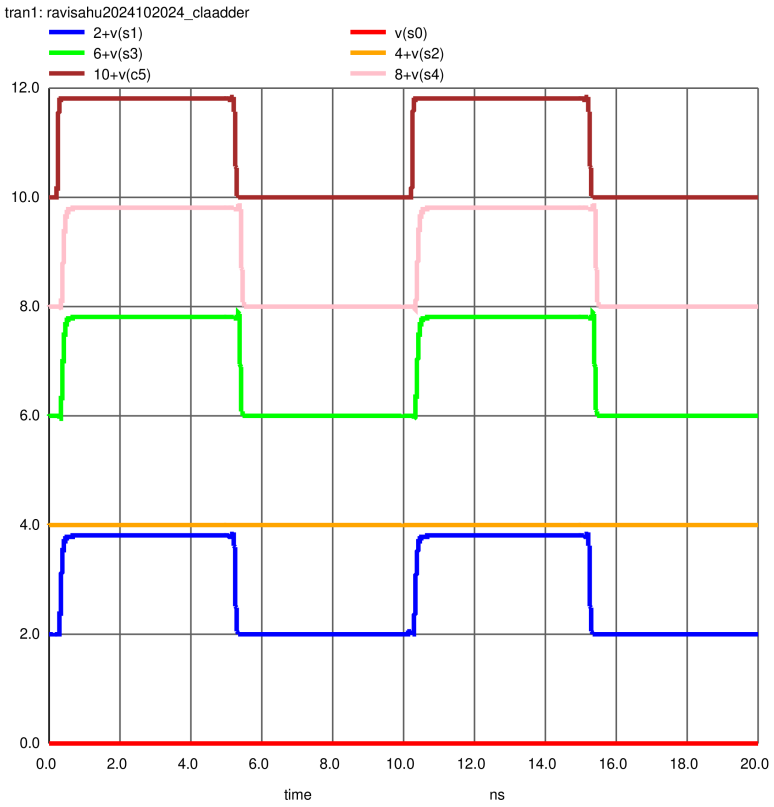


Figure 10: Output signals generated by the Carry Look-Ahead (CLA) block.

| **Measurement** | **Value (ps)** |
|---|---|
| tpcq_cla | **401.32ps** |

## VI.v Final Adder

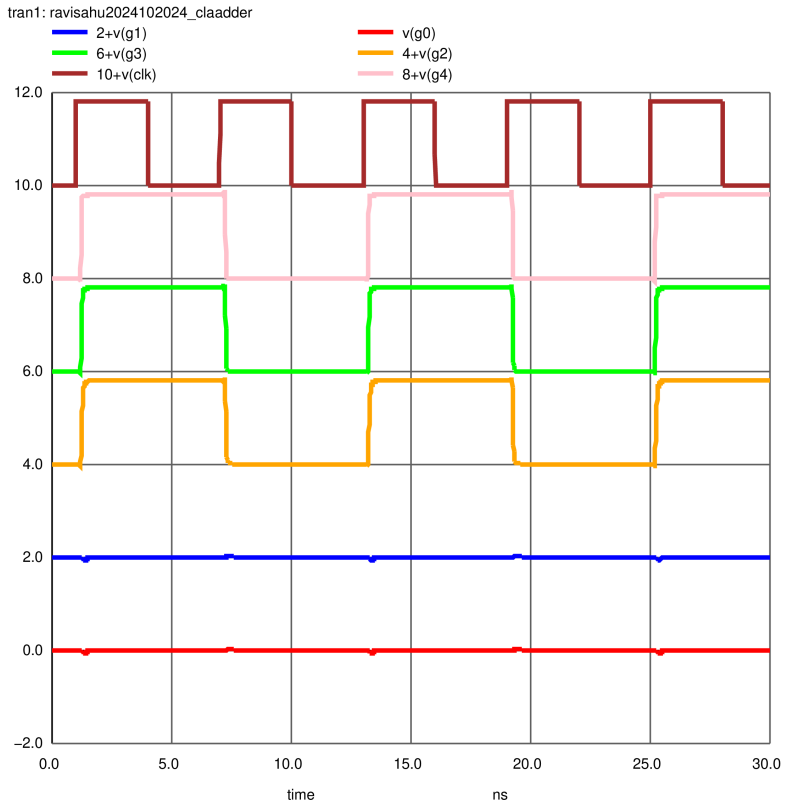


Figure 11: Waveform of the final generate signals for the adder circuit.

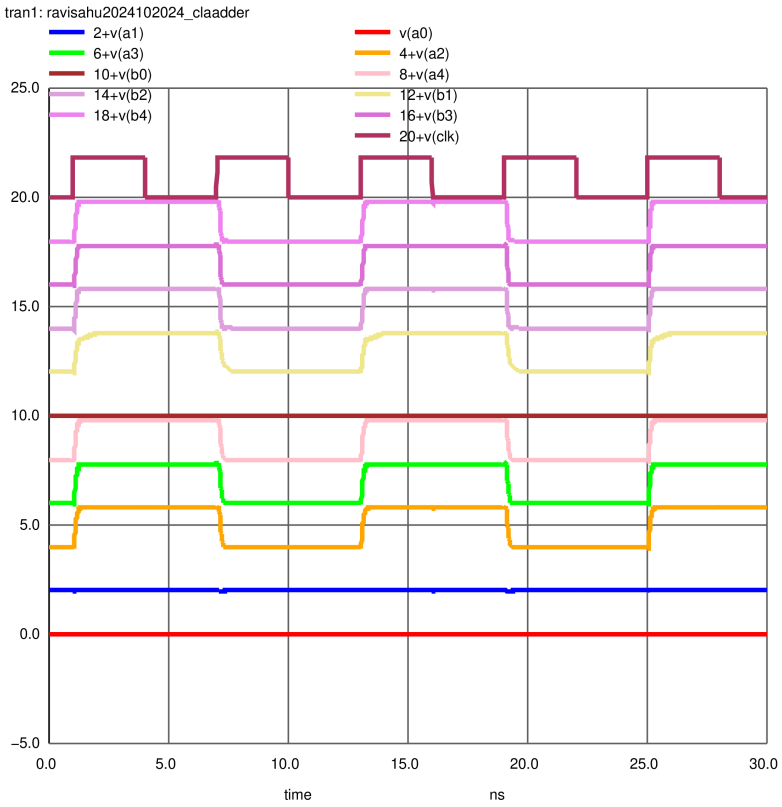


Figure 12: Outputs of the input D-Flip Flops feeding into the adder logic.

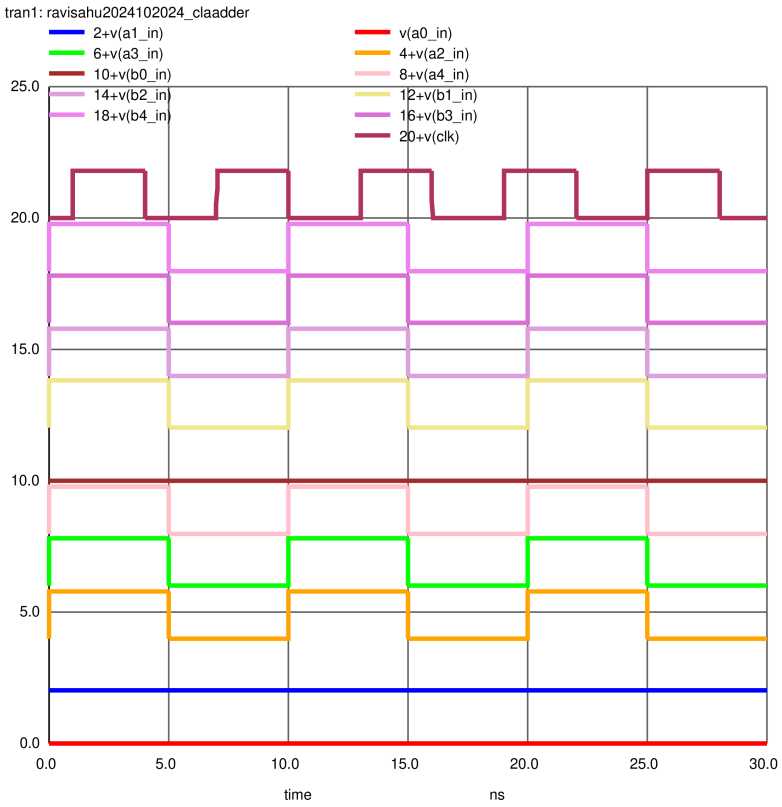


Figure 13: Initial input signals applied to the adder testbench.

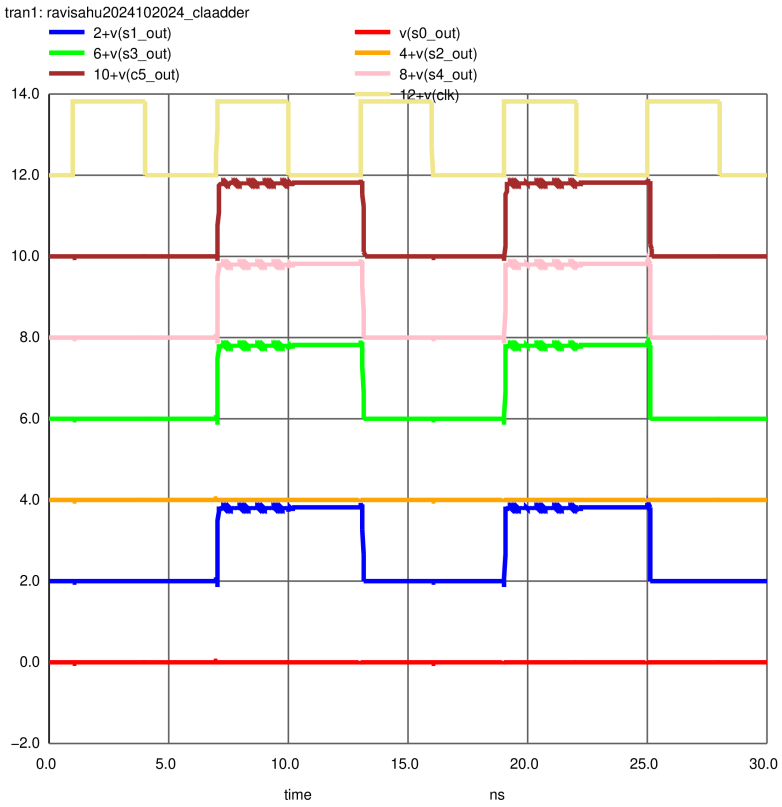


Figure 14: Captured output signals from the final D-Flip Flops.

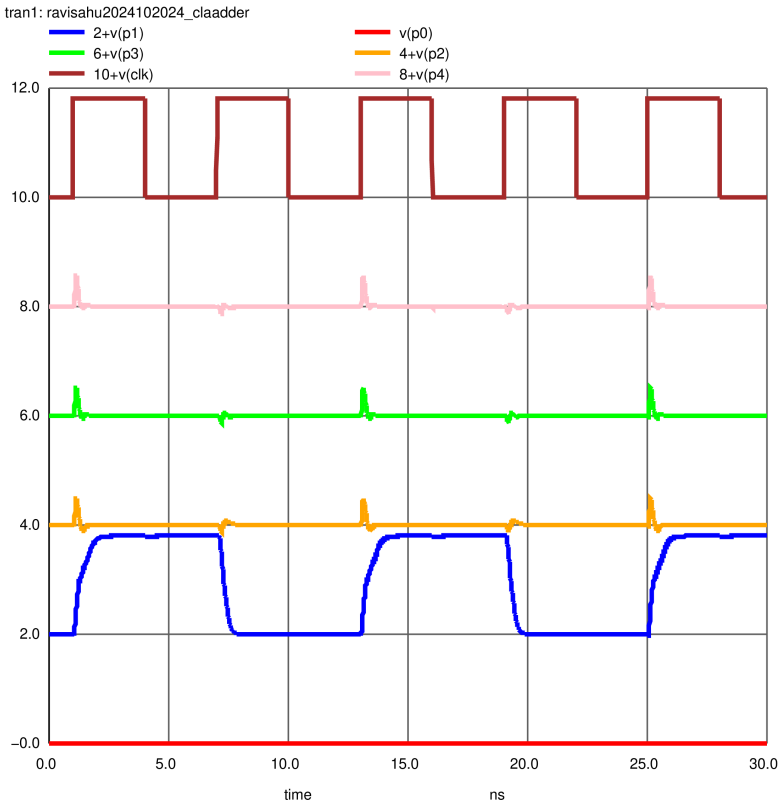


Figure 15: Waveform of the propagate signals within the adder logic.

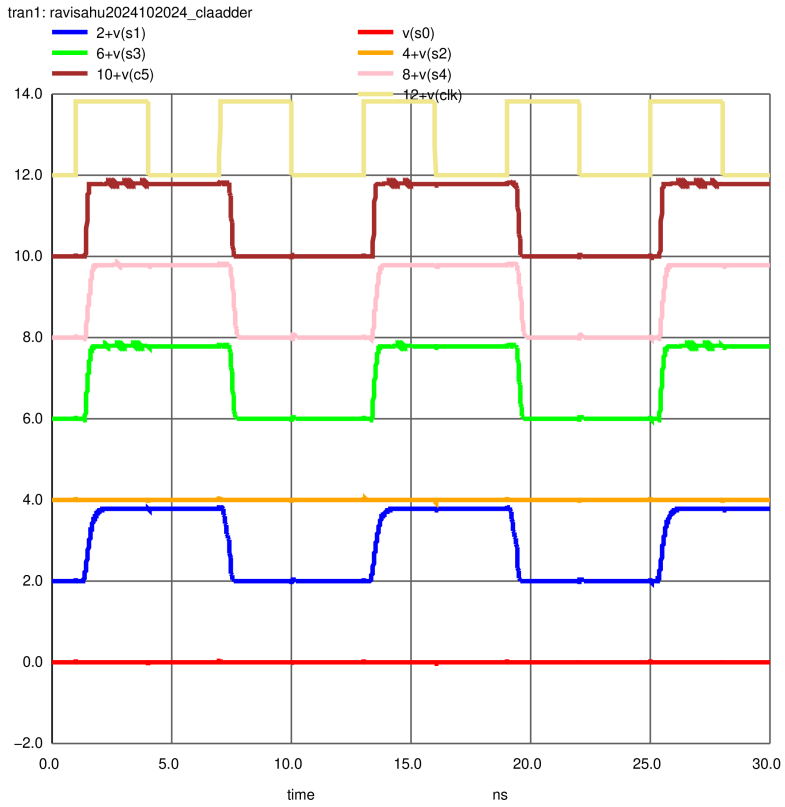


Figure 16: Final Sum and Carry outputs of the adder.

| **Measurement** | **Value (ps)** |
|---|---|
| delay_dff_in | **96.30ps** |
| delay_cla | **432.99ps** |
| delay_dff_out | **47.46ps** |
| total_delay | **576.75ps** |

| **Stage** | **Delay (ps)** | **Percentage of Total** |
|---|---|---|
| Stage 1 - Input DFF | **96.30ps** | 16.7% |
| Stage 2 - CLA Core | **432.99ps** | 75.1% |
| Stage 3 - Output DFF | **47.46ps** | 8.2% |
| **Total Path Delay** | **576.75ps** | 100% |

## VI.vi Summary PreLayout

| Circuit | Key Delay | Value (ps) | Value (ns) |
|---|---|---|---|
| Inverter | tpcq_inv | **17.66** | 0.01766 |
| D Flip-Flop | tpcq_lh (rise) | **2495.99** | 2.496 |
| D Flip-Flop | tpcq_hl (fall) | **144.70** | 0.1447 |
| D Flip-Flop | avg_tpcq | **1320.00** | 1.320 |
| CLA Block | tpcq_cla | **401.32** | 0.40132 |
| Full Adder (Input) | delay_dff_in | **96.30** | 0.0963 |
| Full Adder (Core) | delay_cla | **432.99** | 0.4330 |
| Full Adder (Output) | delay_dff_out | **47.46** | 0.0475 |
| Full Adder (Total) | total_delay | **576.75** | 0.5768 |
| Prop/Gen (Prop) | prop_delay | **154.03** | 0.15403 |
| Prop/Gen (Gen) | gen_delay | **65.13** | 0.06513 |

# VII Post Layout Work

## VII.i Magic Layout

In the design of a 5-bit Carry Lookahead Adder (CLA) using the magic layout methodology, the approach builds upon modular propagate and generate cells that are consistently reused across different bit-widths (2-, 3-, 4-, 5-, and 6-bit). Each bit cell computes the propagate ($P_i = A_i \oplus B_i$) and generate ($G_i = A_i \cdot B_i$) signals, which are then combined to form the carry equations. The carry logic follows the standard lookahead formulation:

$$C_{i+1} = G_i + P_i \cdot C_i,$$

with higher-order carries expanded hierarchically, e.g.,

$$C_2 = G_1 + P_1 G_0 + P_1 P_0 C_0.$$

By verifying the base propagate/generate unit once, the same layout can be replicated and tiled to construct larger CLAs without redesigning the logic. For the 5-bit case, five identical propagate/generate cells are placed in sequence, with careful alignment of diffusion regions and metal layers to minimize parasitic capacitances and interconnect delay. This modularity ensures reproducibility and scalability: the same generator and propagator templates are reused across different bit-width designs, yielding a consistent and optimized carry path suitable for implementation.

VI. STICK DIAGRAMS

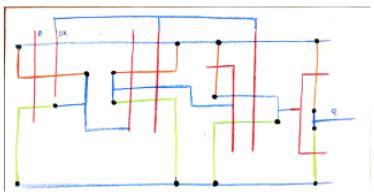

Fig. 21. D Flip-Flop

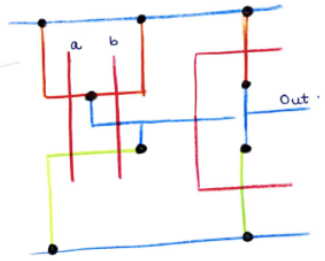


Fig. 22. AND Gate (Generate)

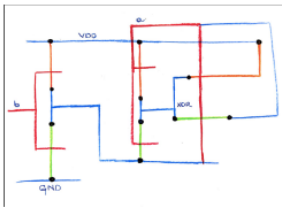


Fig. 23. XOR Gate (Propagate)


Fig. 24. Inverter

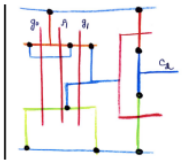

Fig. 25. CARRY 2

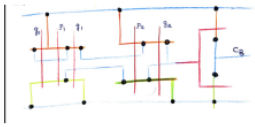

Fig. 26. CARRY 3

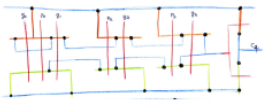

Fig. 27. CARRY 4

## VII.ii Magic Layout

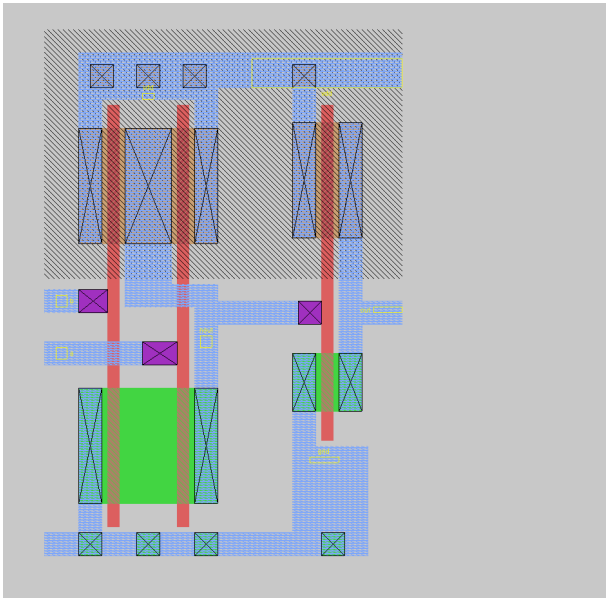

Figure 17: Magic layout for and2

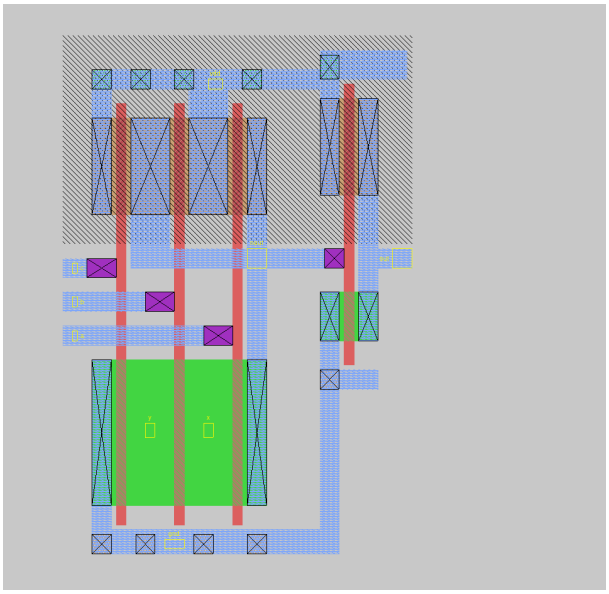

Figure 18: Magic layout for and3

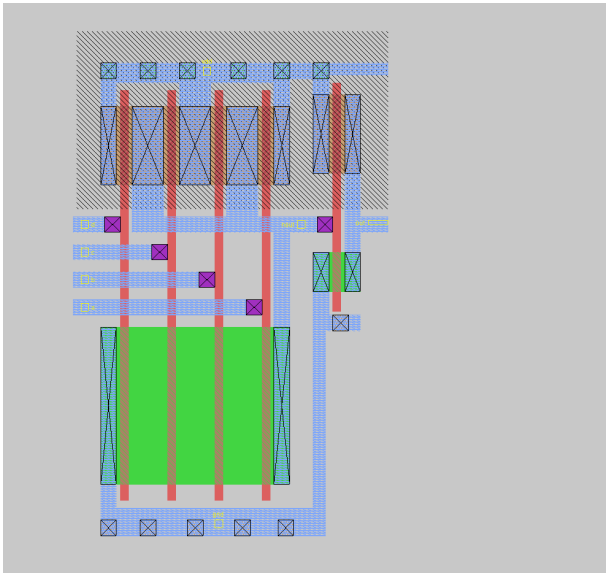

Figure 19: Magic layout for and4

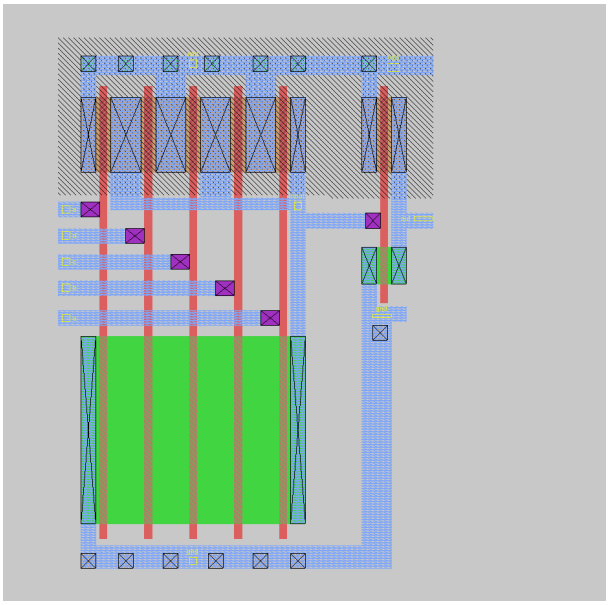

Figure 20: Magic layout for and5

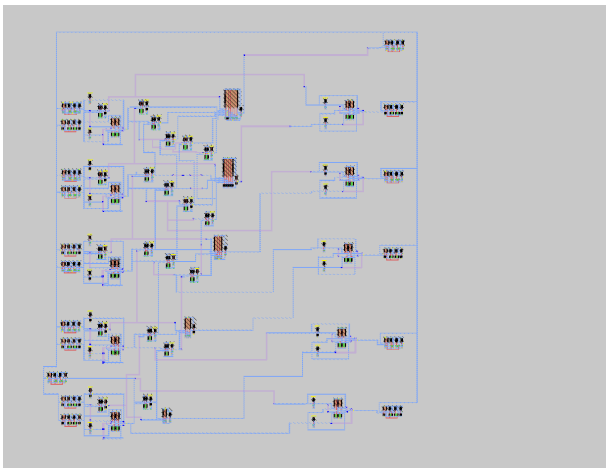

Figure 21: Magic layout for claFinal

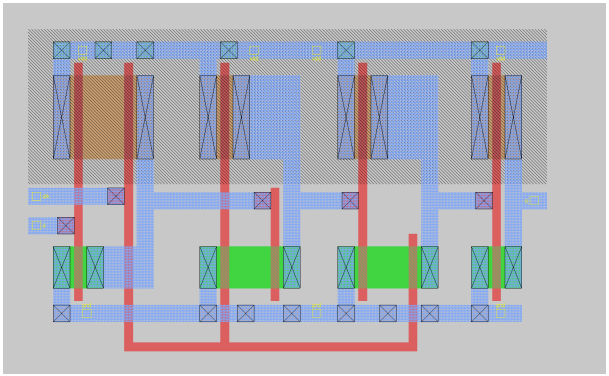

Figure 22: Magic layout for dff.png

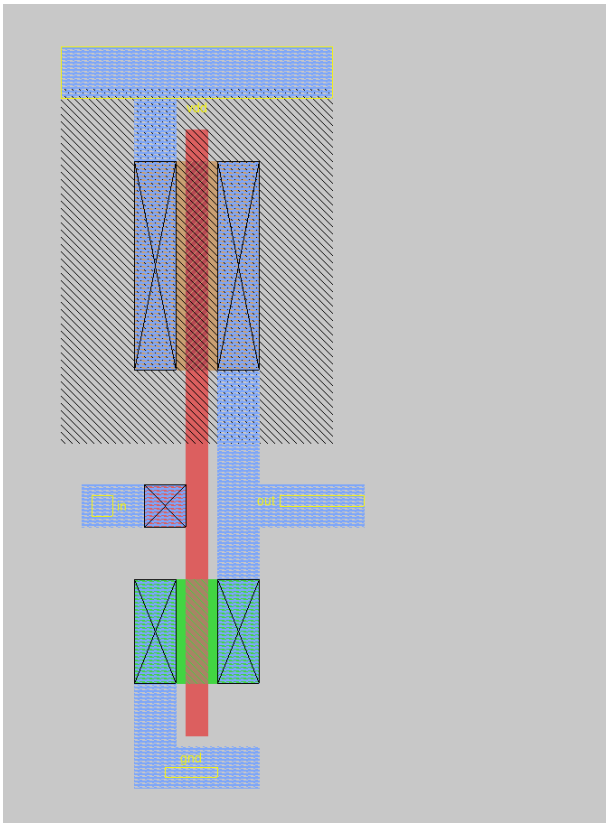

Figure 23: Magic layout for inv

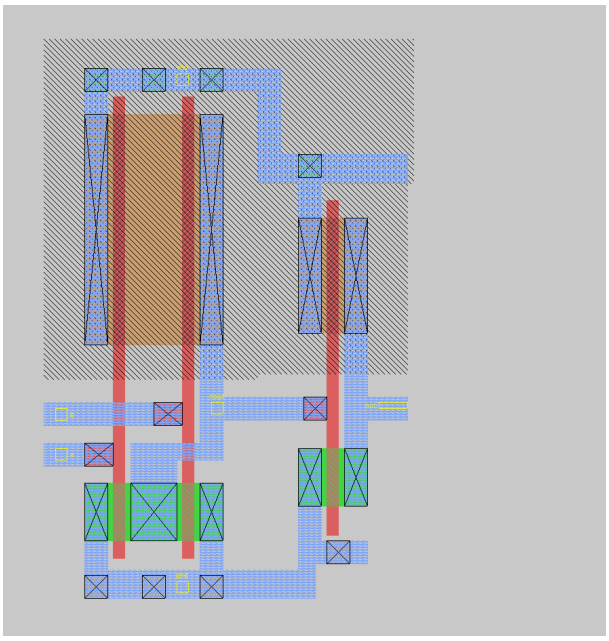

Figure 24: Magic layout for or2

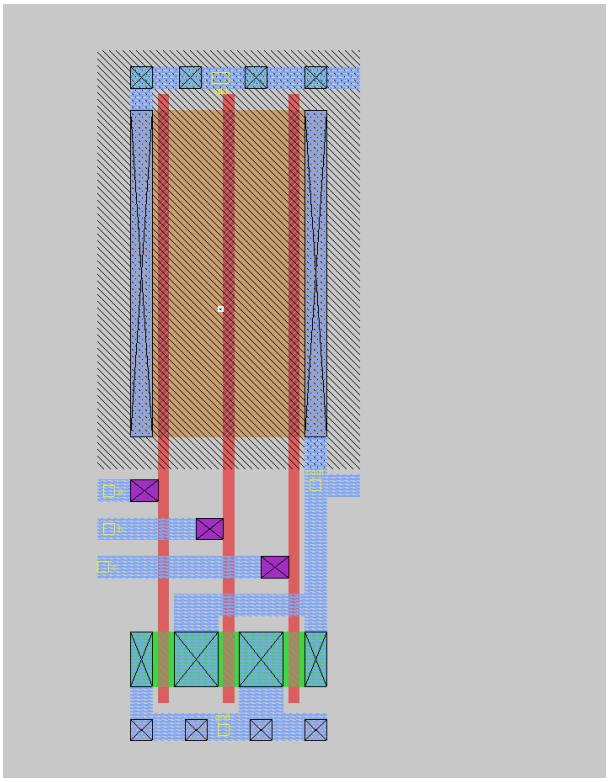

Figure 25: Magic layout for or3

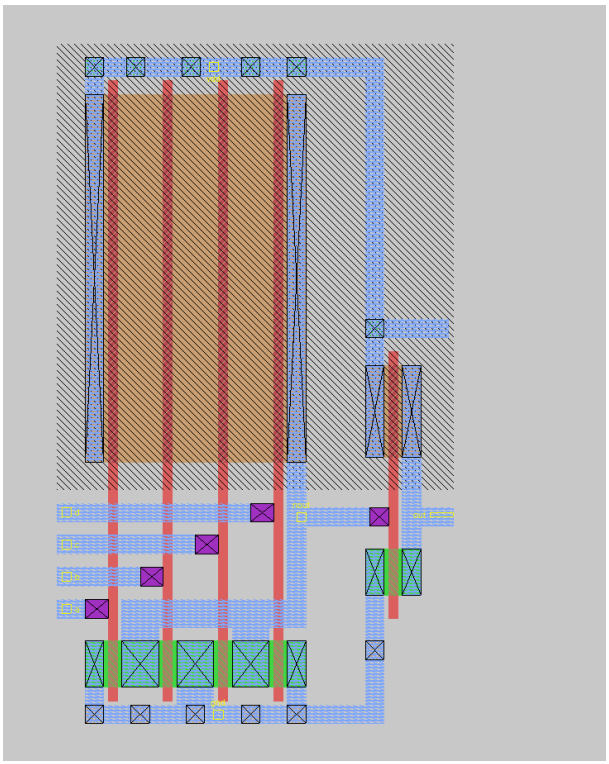

Figure 26: Magic layout for or4

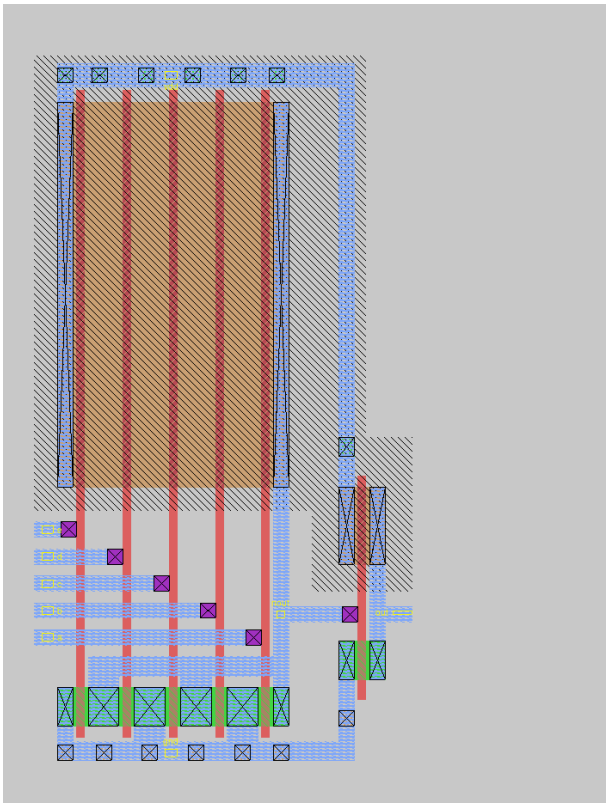

Figure 27: Magic layout for or5

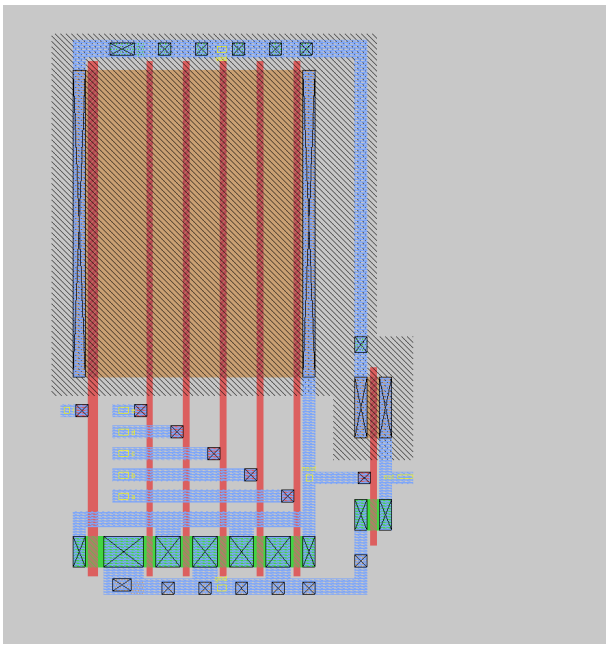

Figure 28: Magic layout for or6

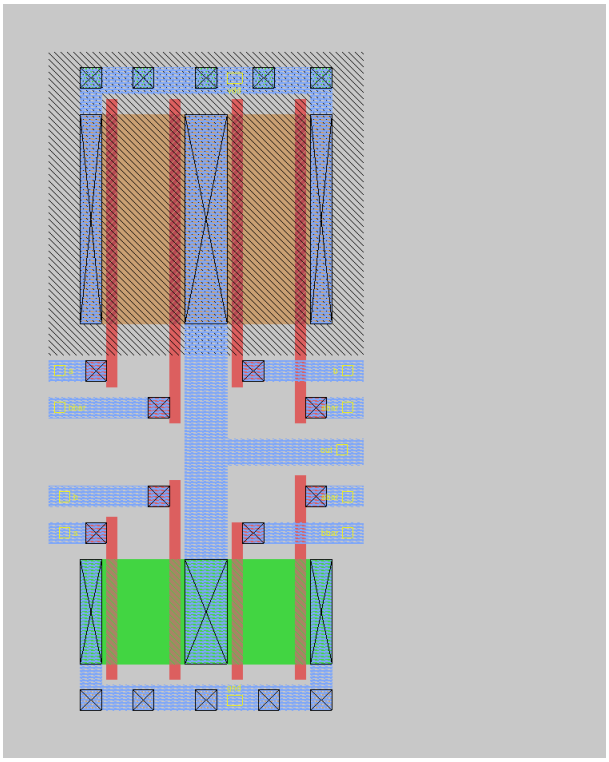

Figure 29: Magic layout for XorPost

## VII.iii Post Layout Simulation

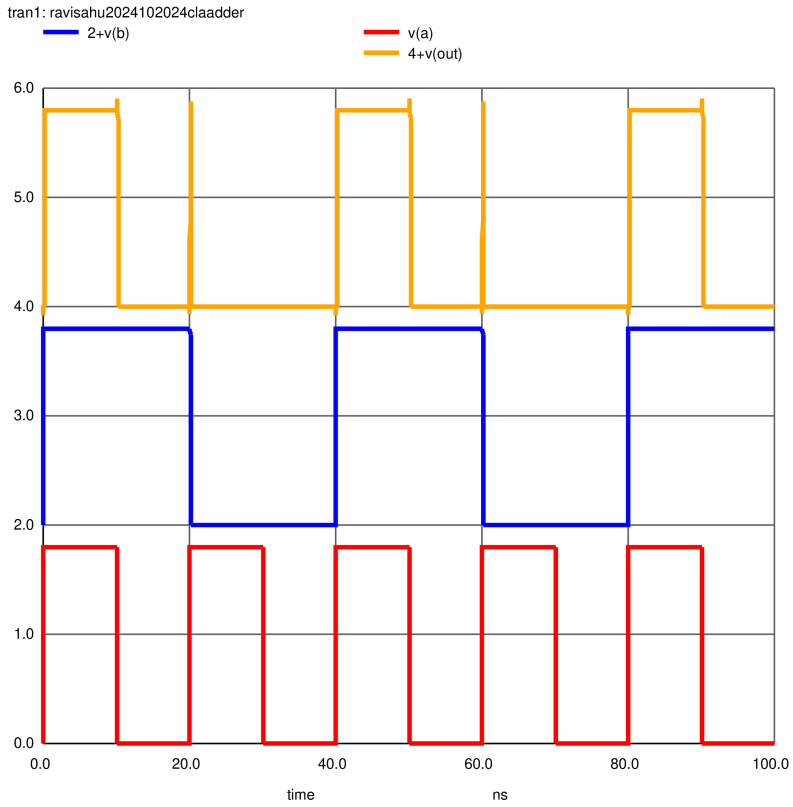


Figure 30: Post-layout simulation waveform for the 2-input AND gate.

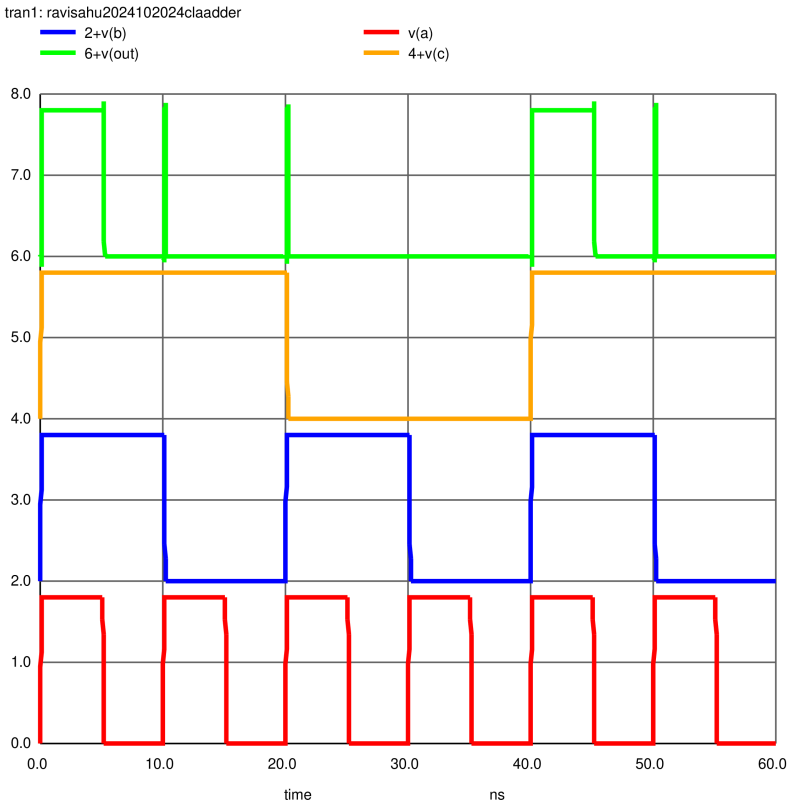


Figure 31: Post-layout simulation waveform for the 3-input AND gate.

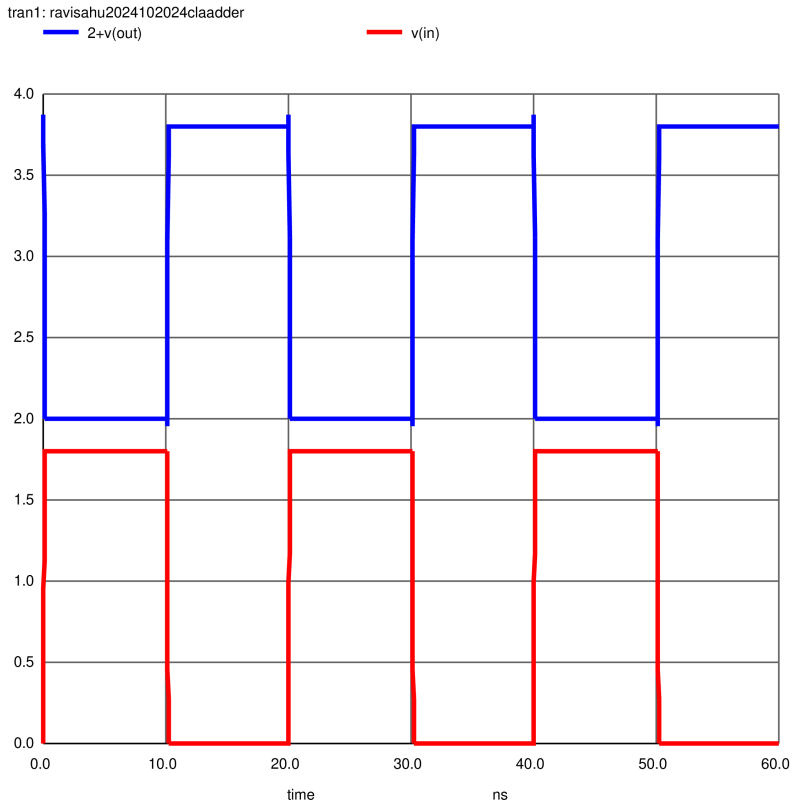


Figure 32: Post-layout simulation waveform for the Inverter (INV).

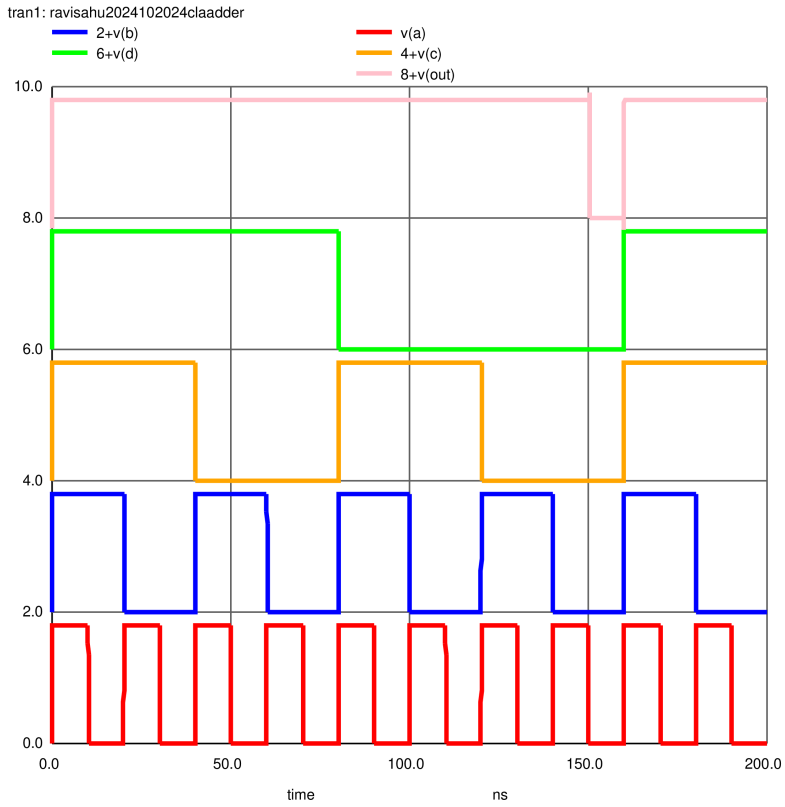


Figure 33: Post-layout simulation waveform for the 4-input OR gate.

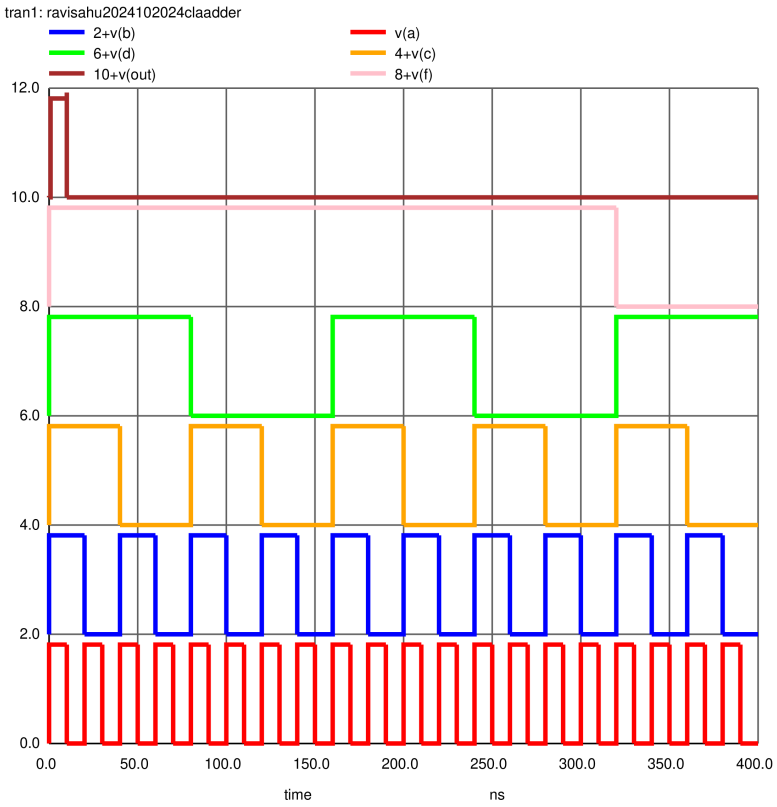


Figure 34: Post-layout simulation waveform for the 6-input OR gate.

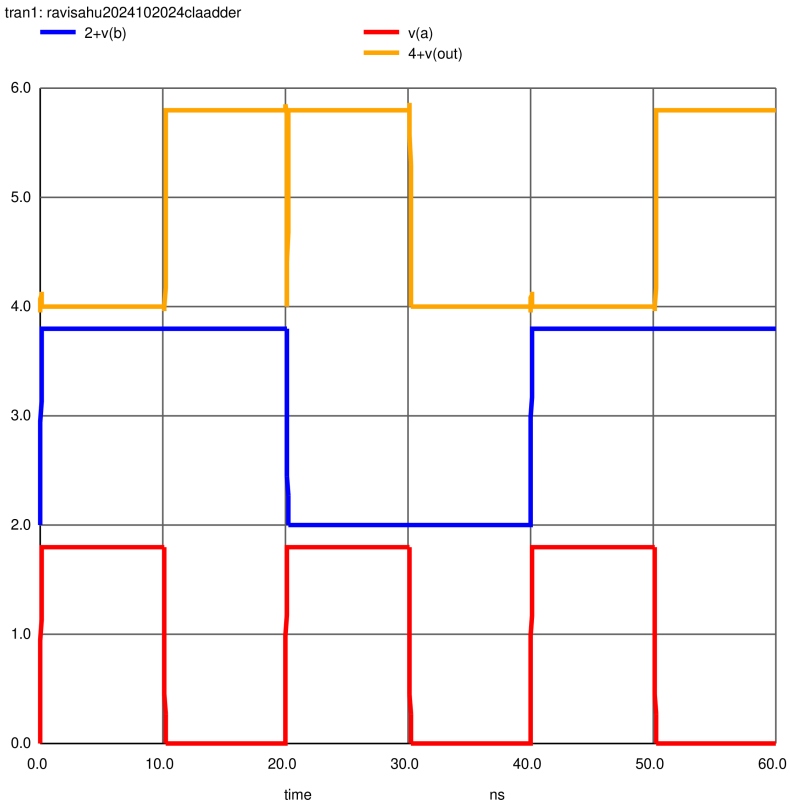


Figure 35: Post-layout simulation waveform for the XOR gate.

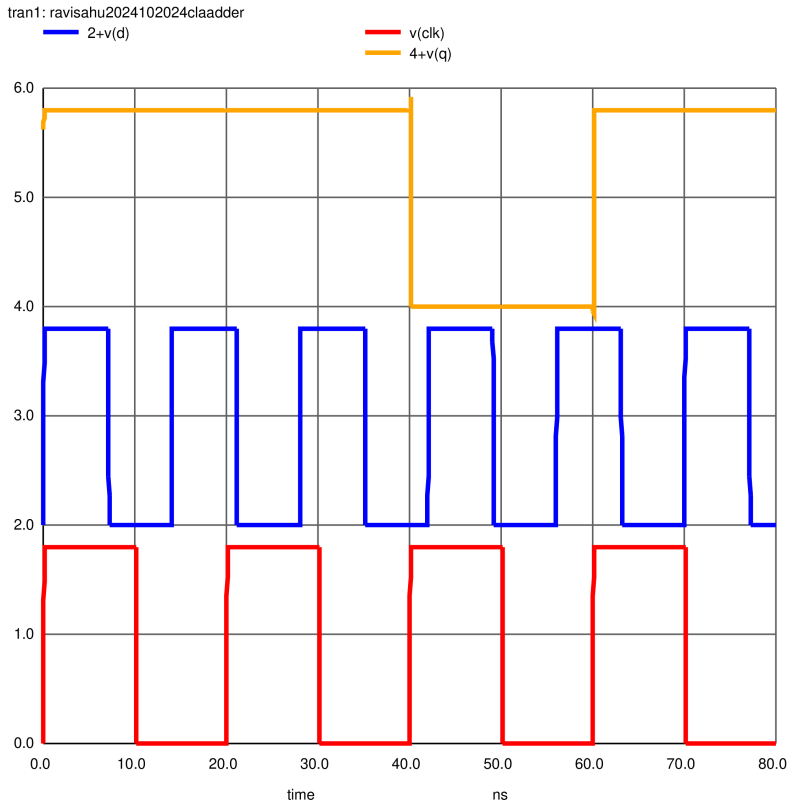


Figure 36: Post-layout simulation waveform for the D Flip-Flop (DFF).

```
Node                                    Voltage
----                                    -------
vdd                                         1.8
clk                                           0
d                                             0
q                                       1.67518
vd#branch                                     0
vclk#branch                                   0
vdd#branch                          -4.94458e-05

 Reference value :  7.12850e-08
No. of Data Rows : 8116

  Measurements for Transient Analysis

tpcq_lh             =  6.006806e-08 targ=  6.011806e-08 trig=  5.000000e-11
tpcq_hl             =  2.010624e-08 targ=  4.015624e-08 trig=  2.005000e-08
avg_tpcq            =  4.00872e-08
hold_time           =  -2.000000e-08 targ=  5.000000e-11 trig=  2.005000e-08

Warning: Missing charsets in String to FontSet conversion
```

Figure 37: Hold Time

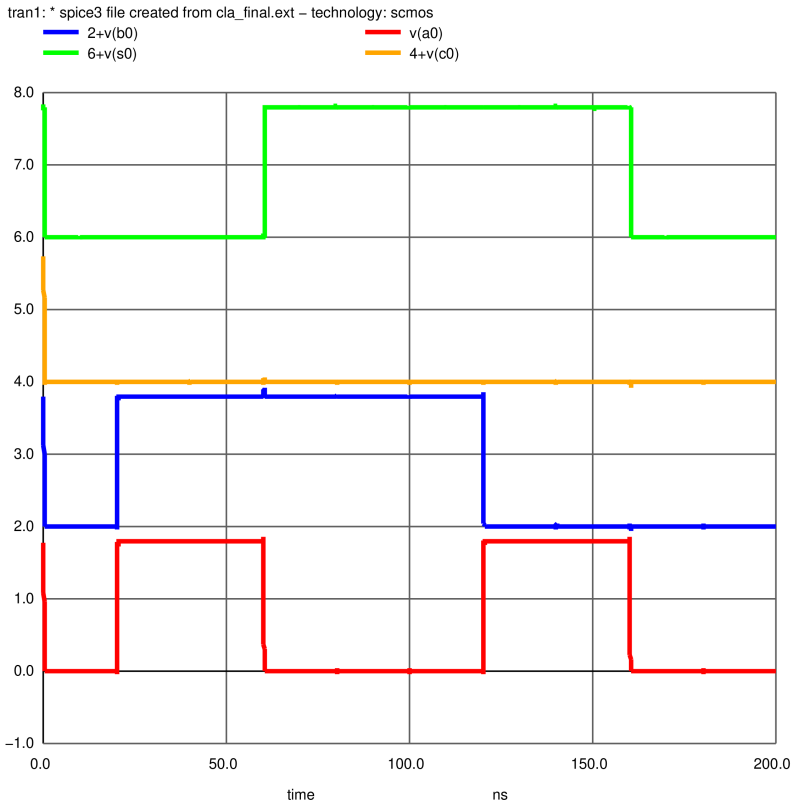


Figure 38: Final output waveform for the 0th bit of the Carry Look-Ahead (CLA) adder.

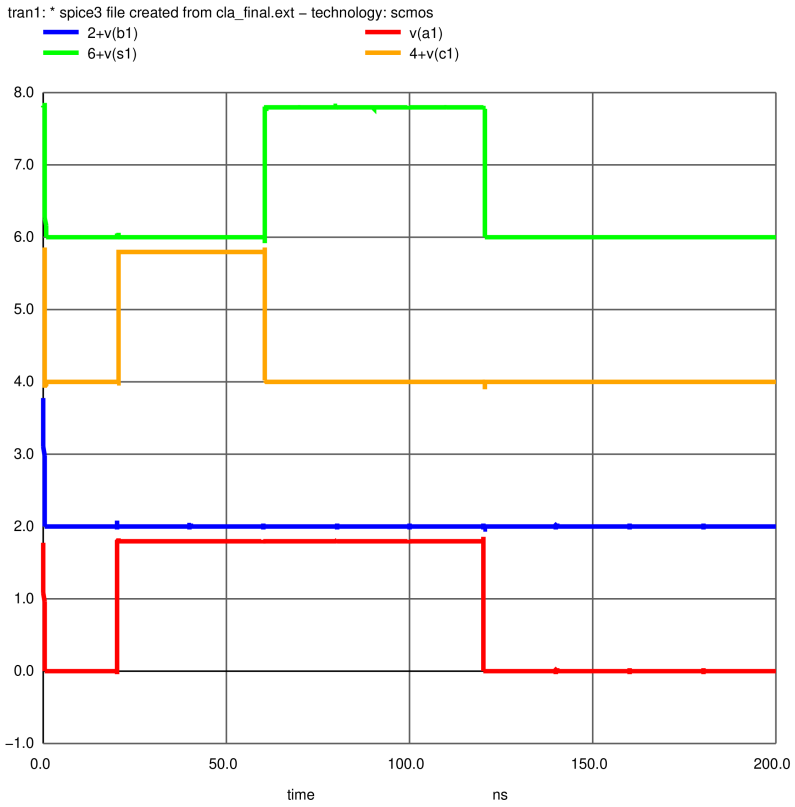


Figure 39: Final output waveform for the 1st bit of the Carry Look-Ahead (CLA) adder.

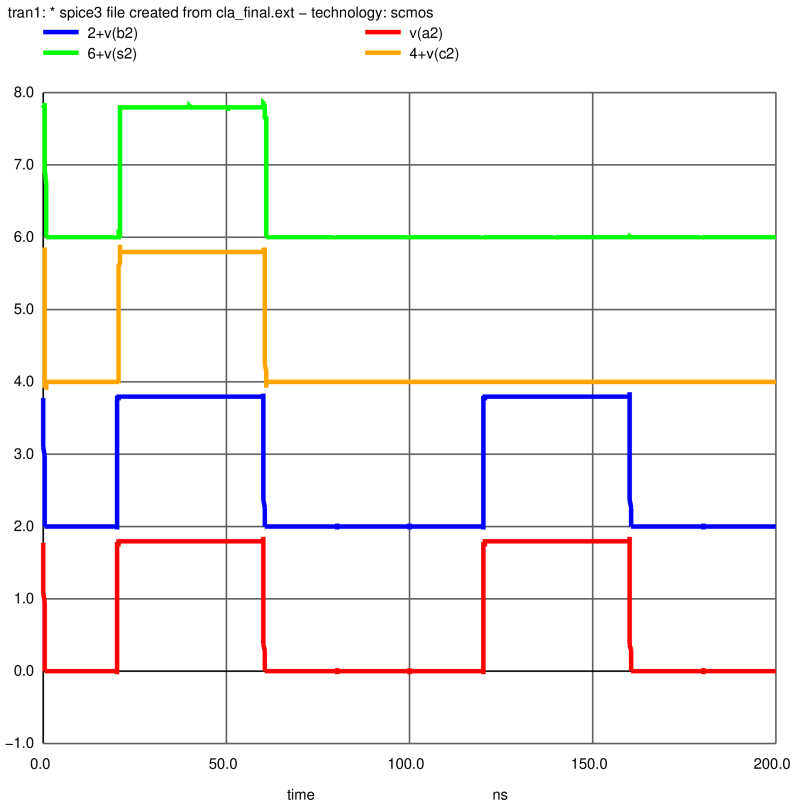


Figure 40: Final output waveform for the 2nd bit of the Carry Look-Ahead (CLA) adder.

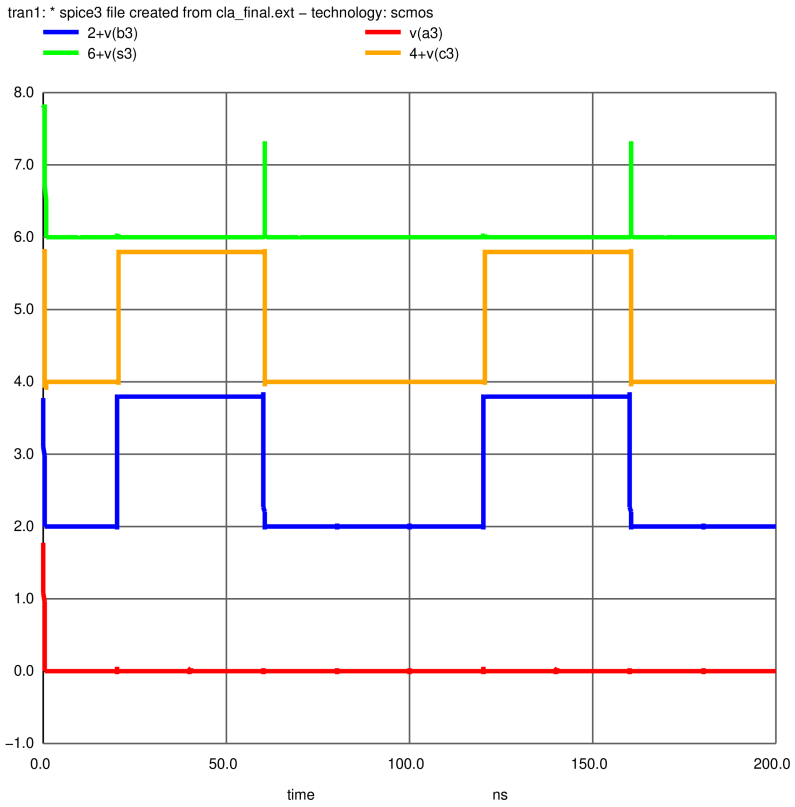


Figure 41: Final output waveform for the 3rd bit of the Carry Look-Ahead (CLA) adder.

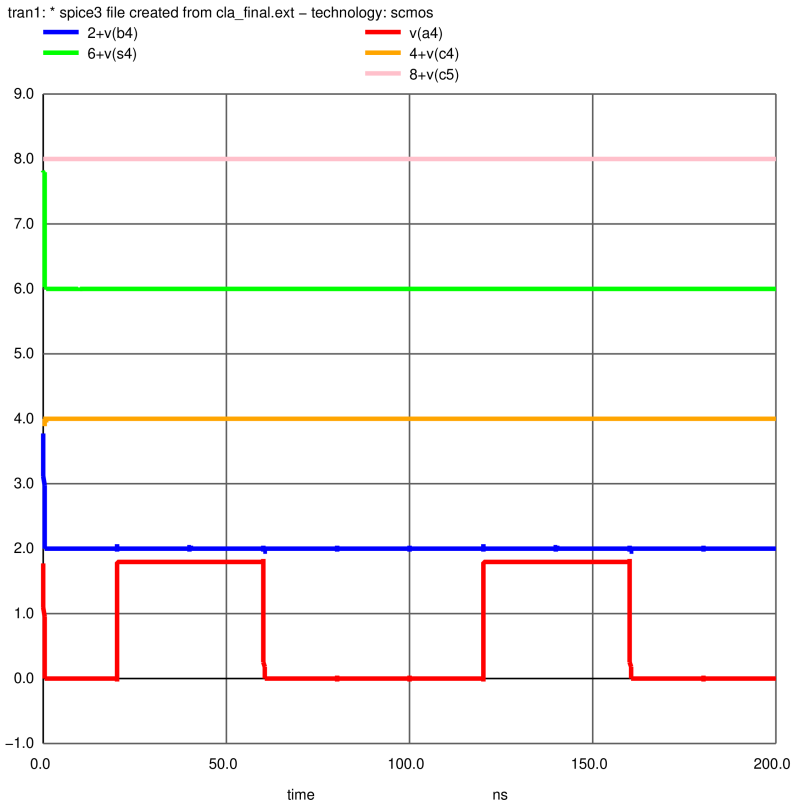


Figure 42: Final output waveform for the 4th bit of the Carry Look-Ahead (CLA) adder.

### VII.iii.1 Critical Path Delay A=11111 , B=00001

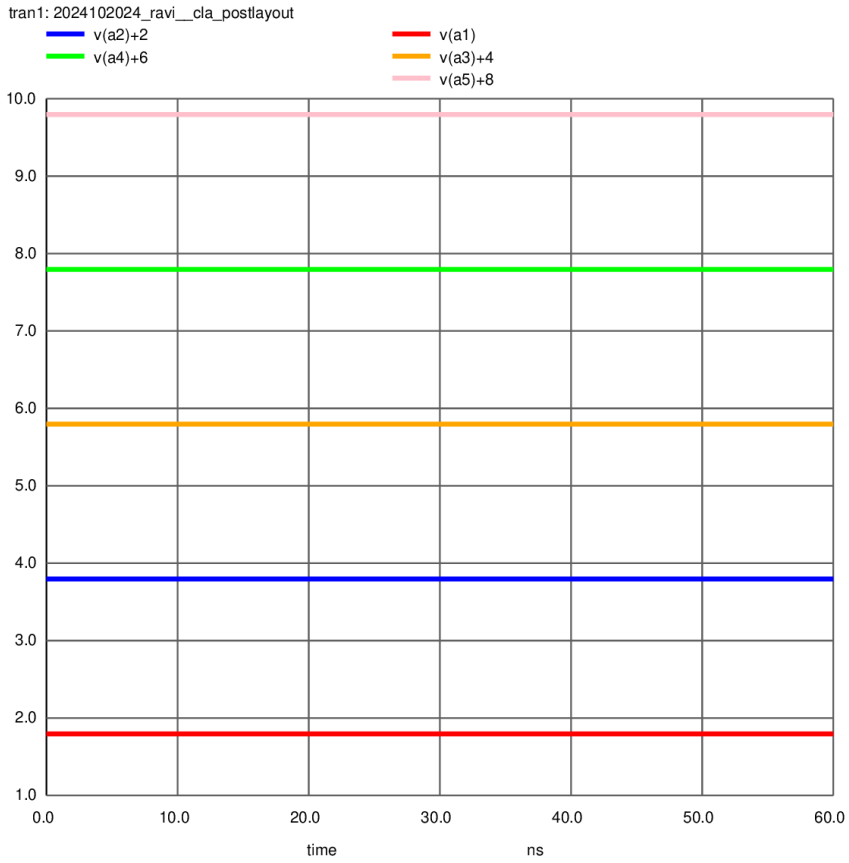


Figure 43: input A

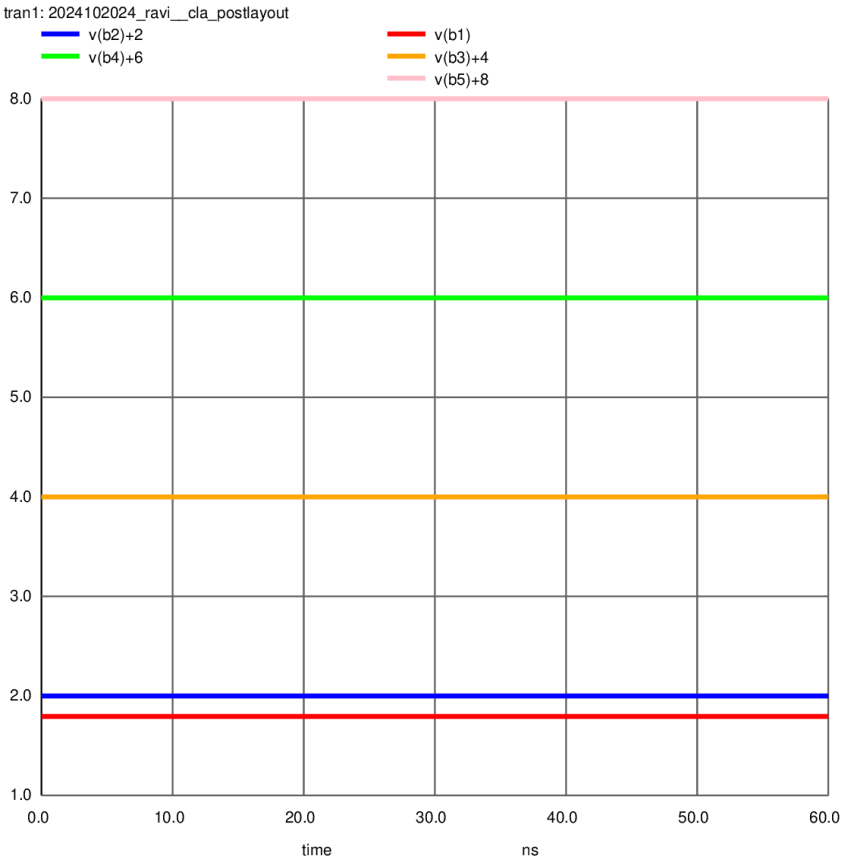


Figure 44: input B

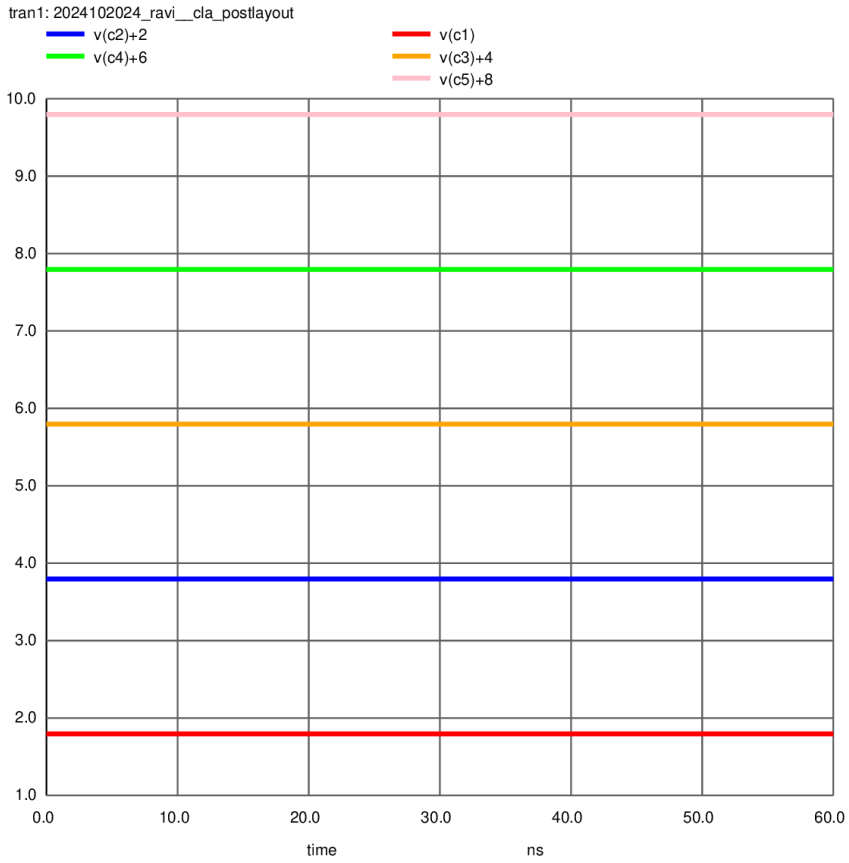


Figure 45: Carry Out (C)

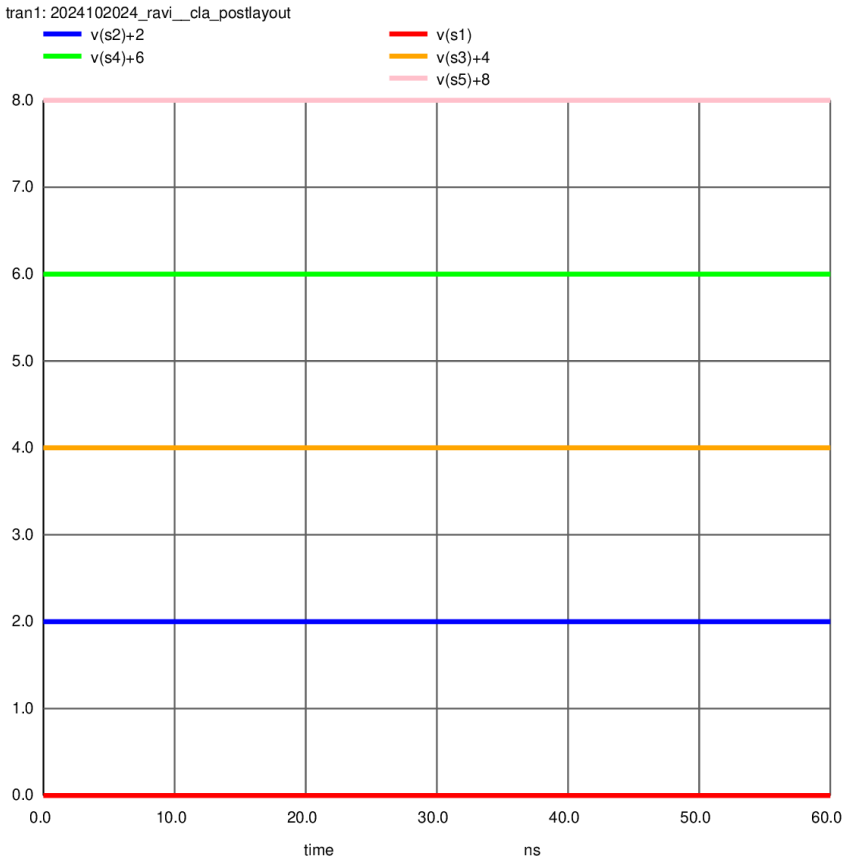


Figure 46: Sum Out (S)

## VII.iv Delay analysis

| XOR GATE | | | | |
|---|---|---|---|---|
| **Input** | **Pin** | **Rise** | **Fall** | **Average** |
| Input A | a | 10159.43 ns | 0.0452 ps | 5102.33 ns |
| Input B | b | 10159.43 ns | -19954.78 ps | -4897.67 ns |

| D FLIP-FLOP [Clock-to-Q measurements] | | | | |
|---|---|---|---|---|
| **Signal** | **Pin** | **Rise (ps)** | **Fall (ps)** | **Average (ps)** |
| Clock | clk | 60067.06 | 20106.24 | 40087.15 |
| SetupTime | $t_{setup}$ | **2 ns** | | |

| **CLA [Sum outputs S0-S3, Carry output C5]** | | | | |
|---|---|---|---|---|
| **Output** | **Pin** | **Rise (ps)** | **Fall (ps)** | **Average (ps)** |
| Sum Bit 0 (S0) | s0 | 60456.31 | -19700.34 | 20377.98 |
| Sum Bit 1 (S1) | s1 | 60516.97 | -19490.64 | 20513.16 |
| Sum Bit 2 (S2) | s2 | 20797.46 | -19453.23 | 672.11 |
| Sum Bit 3 (S3) | s3 | 60474.01 | -19464.47 | 20504.76 |
| Sum Bit 4 (S4) | s4 | **60500.00** | -19531.97 | **20484.02** |
| Carry Out (C5) | c5 | **60500.00** | **-19500.00** | **20500.00** |

# VIII Verilog

```
❯ iverilog -o cla_sim cla.v cla_tb.v && vvp cla_sim
VCD info: dumpfile dump_cla_adder.vcd opened for output.
Time=0 a_in=00000 b_in=00000 cin=0 sum=xxxxx cout=x
Time=7 a_in=00000 b_in=00000 cin=0 sum=xxxxx cout=0
Time=21 a_in=00000 b_in=00000 cin=0 sum=00000 cout=0
Time=105 a_in=00001 b_in=00001 cin=0 sum=00000 cout=0
Time=119 a_in=00001 b_in=00001 cin=0 sum=00010 cout=0
Time=217 a_in=01111 b_in=00001 cin=0 sum=00010 cout=0
Time=231 a_in=01111 b_in=00001 cin=0 sum=10000 cout=0
Time=329 a_in=10101 b_in=01010 cin=0 sum=10000 cout=0
Time=343 a_in=10101 b_in=01010 cin=0 sum=11111 cout=0
Time=441 a_in=10000 b_in=01111 cin=0 sum=11111 cout=0
Time=553 a_in=11111 b_in=11111 cin=0 sum=11111 cout=1
Time=567 a_in=11111 b_in=11111 cin=0 sum=11110 cout=1
cla_tb.v:58: $finish called at 753 (1s)
```

Figure 47: Verilog Output

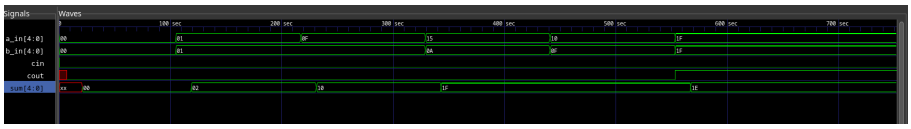


Figure 48: WaveForm

# IX RTL Simulation

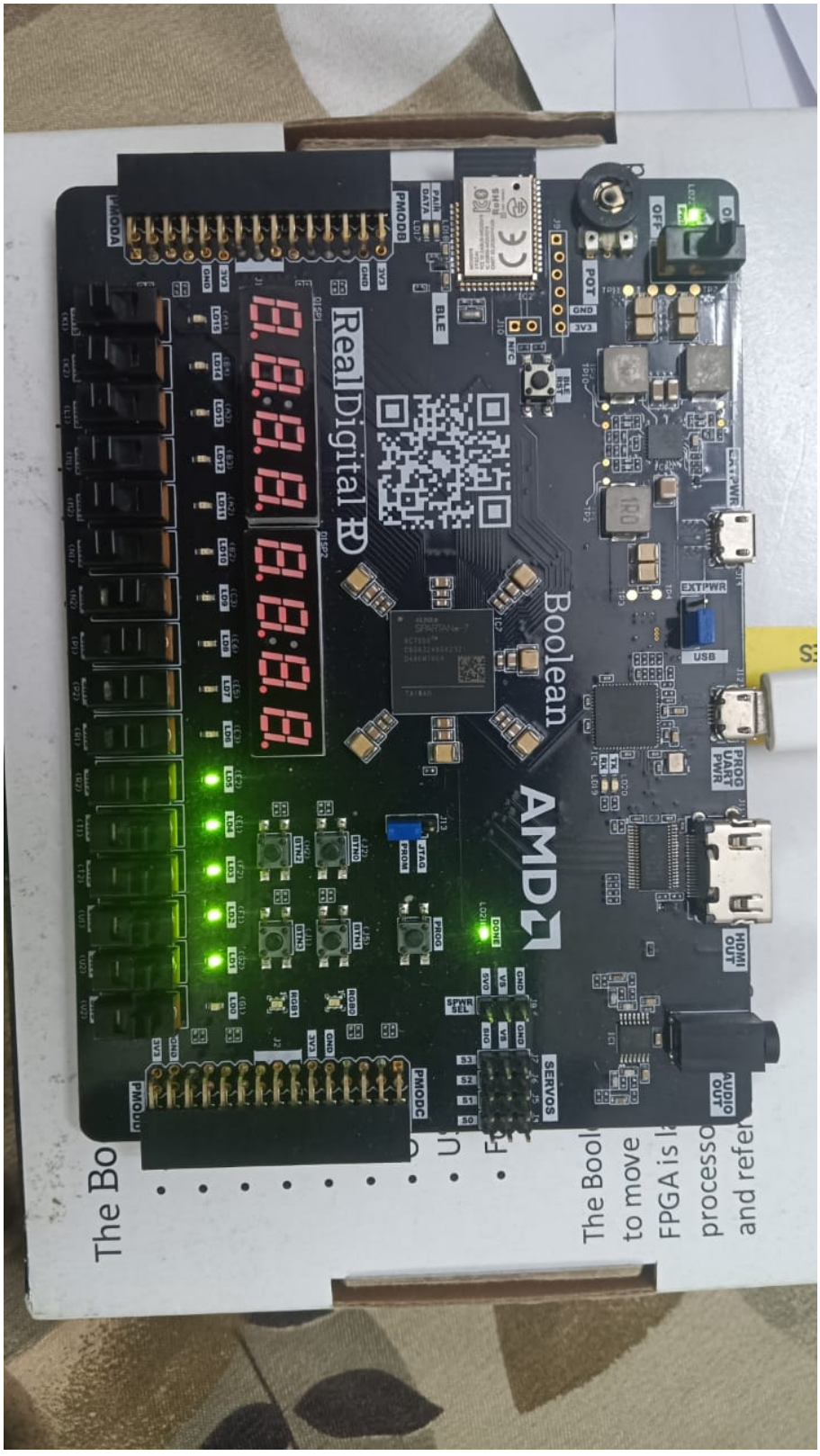


Figure 49: Add 11111+11111

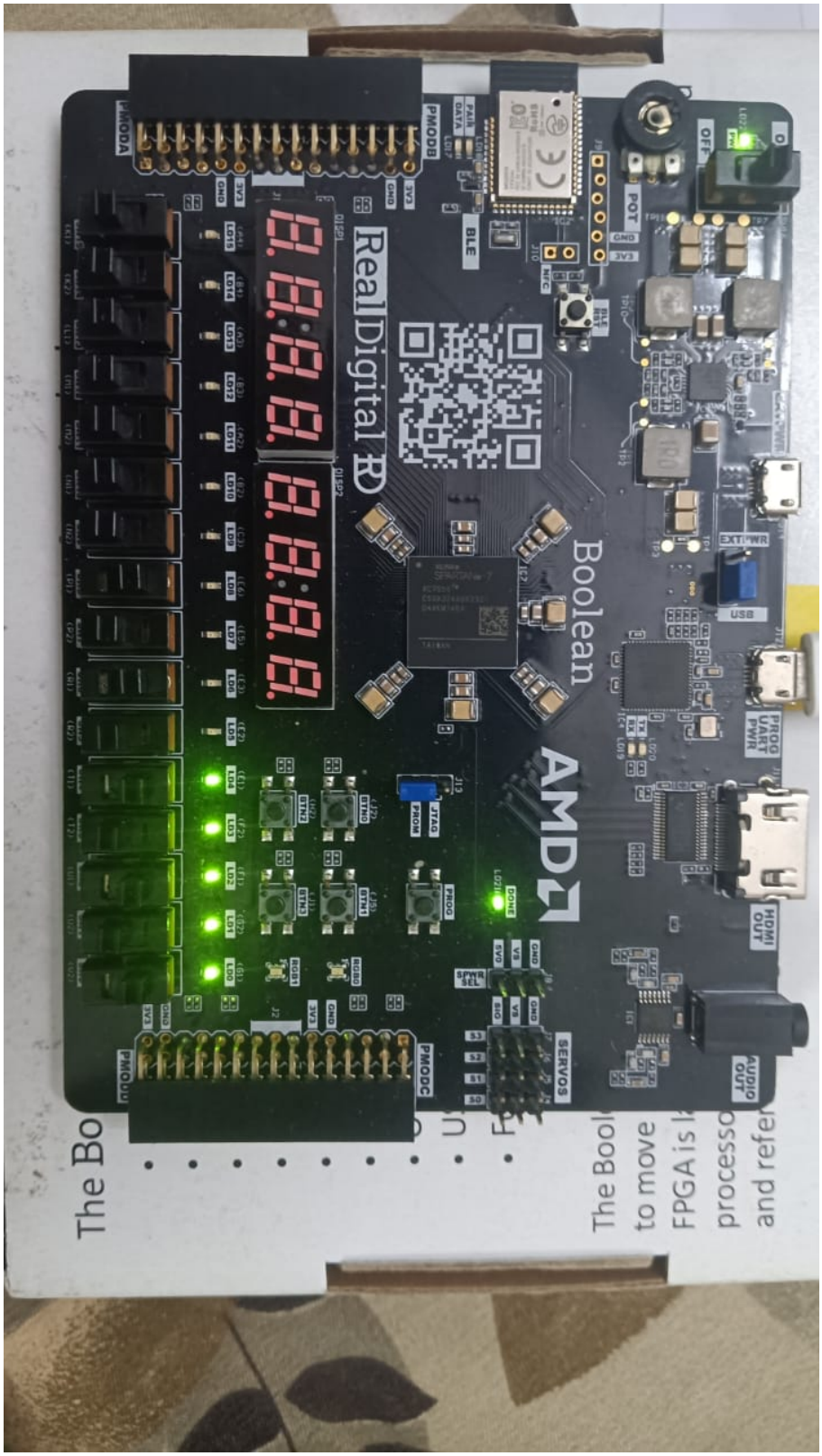


Figure 50: Add 10101+01010


Figure 51: Add 11000+11100