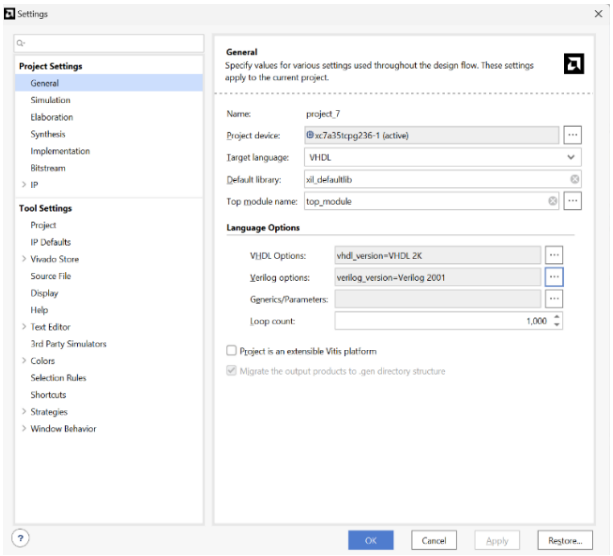
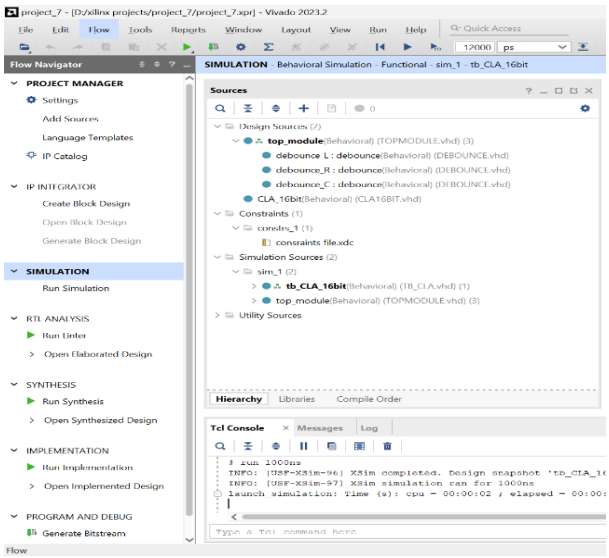
FPGA LAB ASSIGNMENT

Design and Implementation of a 16-Bit Fast Adder/Carry Look-Ahead Adder Using VHDL on Basys3 FPGA

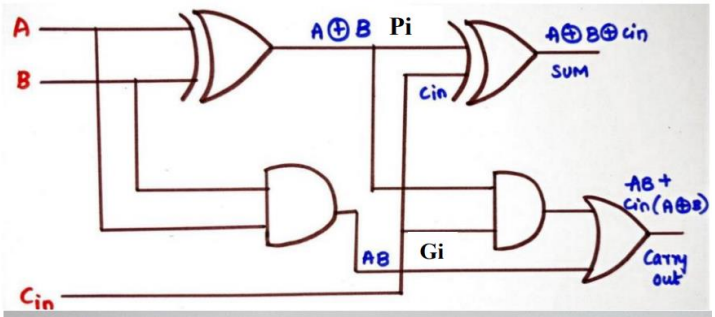
- **IMPLEMENTATION OF 16-BIT CLA:**





I have selected the board **basys3** in order to perform the implementation and I have created design **code** for 16-bit CLA and **test bench** for simulation and added **top module**, **debounce**, **constraints file**.

- **Design of 1-bit CLA:**



Generate term → $G_i = A_i B_i$ → ①

Propagate term → $P_i = \overline{A_i} B_i + A_i \overline{B_i} = A_i \oplus B_i$ → ②

* Adder has 2 outputs Sum and Carry-out

$Sum = A \oplus B \oplus cin$

$Carry = AB + cin(A \oplus B)$

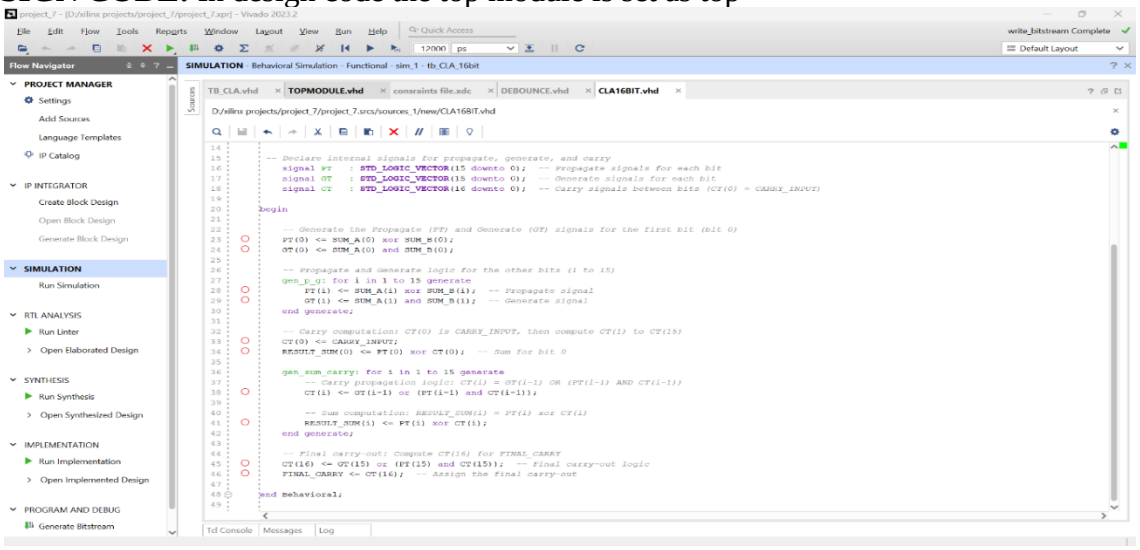
In terms of (i) P_i

$Sum_i = A_i \oplus B_i \oplus C_{i-1} = P_i \oplus C_{i-1}$ → ③

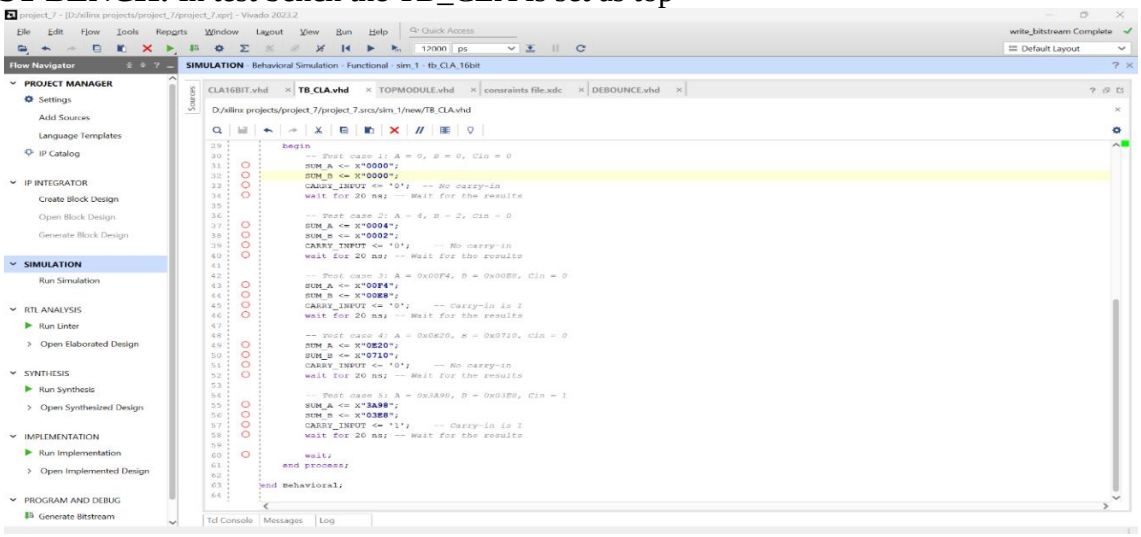
$C_i = A_i B_i + C_{i-1}(A_i \oplus B_i) = G_i + C_{i-1} P_i$ → ④

($A_i B_i$ = G_i; $A_i \oplus B_i$ = P_i)

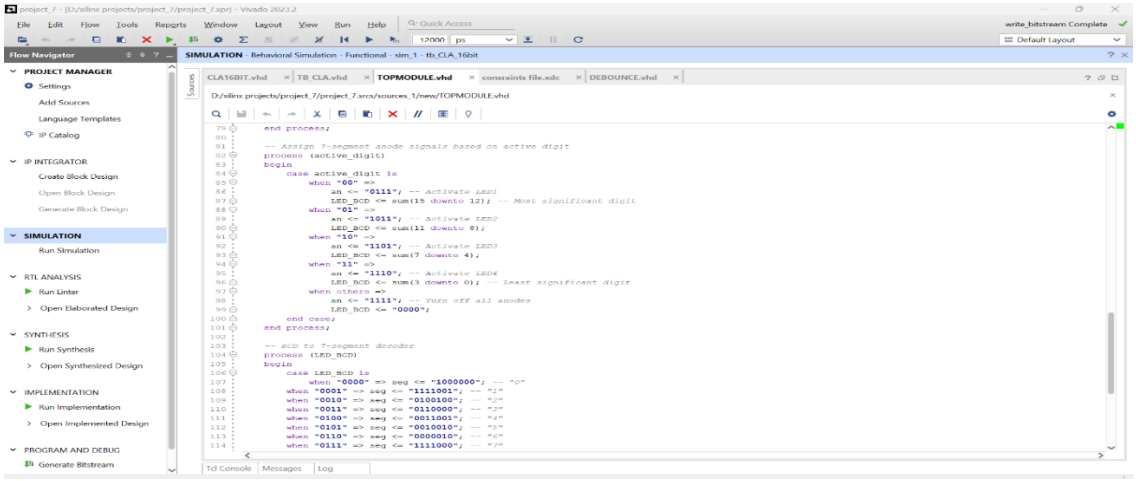
DESIGN CODE: In design code the top module is set as top



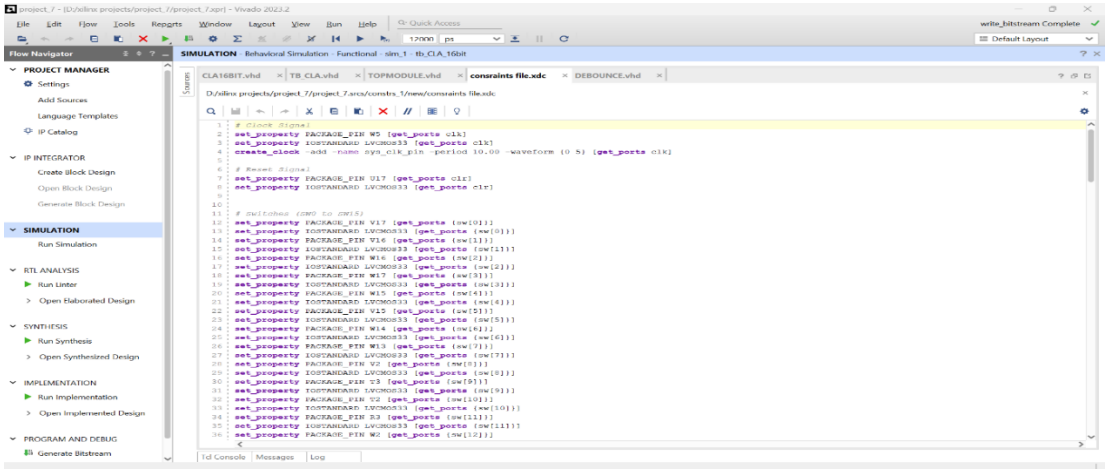
TEST BENCH: In test bench the TB_CLA is set as top



TOPMODULE:

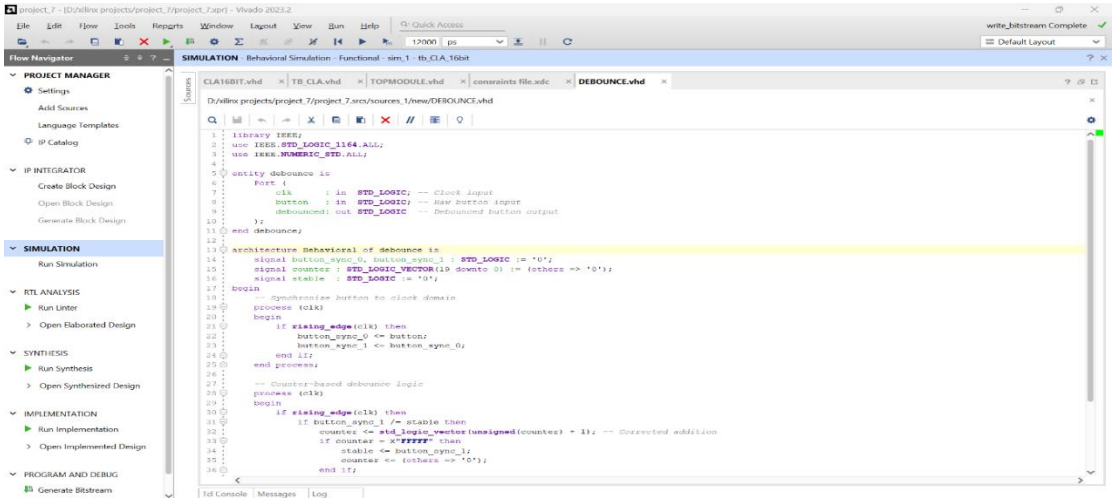


CONSTRAINTS FILE: Under constraints file I added constraints file



```
# Clock Signal
set_property PACKAGE_PIN W5 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

# Reset Signal
set_property PACKAGE_PIN U17 [get_ports clr]
set_property IOSTANDARD LVCMOS33 [get_ports clr]


# Switches (sw0 to sw15)
set_property PACKAGE_PIN V17 [get_ports {sw[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]}]
set_property PACKAGE_PIN V16 [get_ports {sw[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]}]
set_property PACKAGE_PIN W16 [get_ports {sw[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[2]}]
set_property PACKAGE_PIN W17 [get_ports {sw[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[3]}]
set_property PACKAGE_PIN W15 [get_ports {sw[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[4]}]
set_property PACKAGE_PIN V15 [get_ports {sw[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[5]}]
set_property PACKAGE_PIN W14 [get_ports {sw[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[6]}]
set_property PACKAGE_PIN W13 [get_ports {sw[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[7]}]
set_property PACKAGE_PIN V2 [get_ports {sw[8]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[8]}]
set_property PACKAGE_PIN T3 [get_ports {sw[9]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[9]}]
set_property PACKAGE_PIN T2 [get_ports {sw[10]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[10]}]
set_property PACKAGE_PIN R3 [get_ports {sw[11]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[11]}]
set_property PACKAGE_PIN W2 [get_ports {sw[12]}]
```

DEBOUNCE:

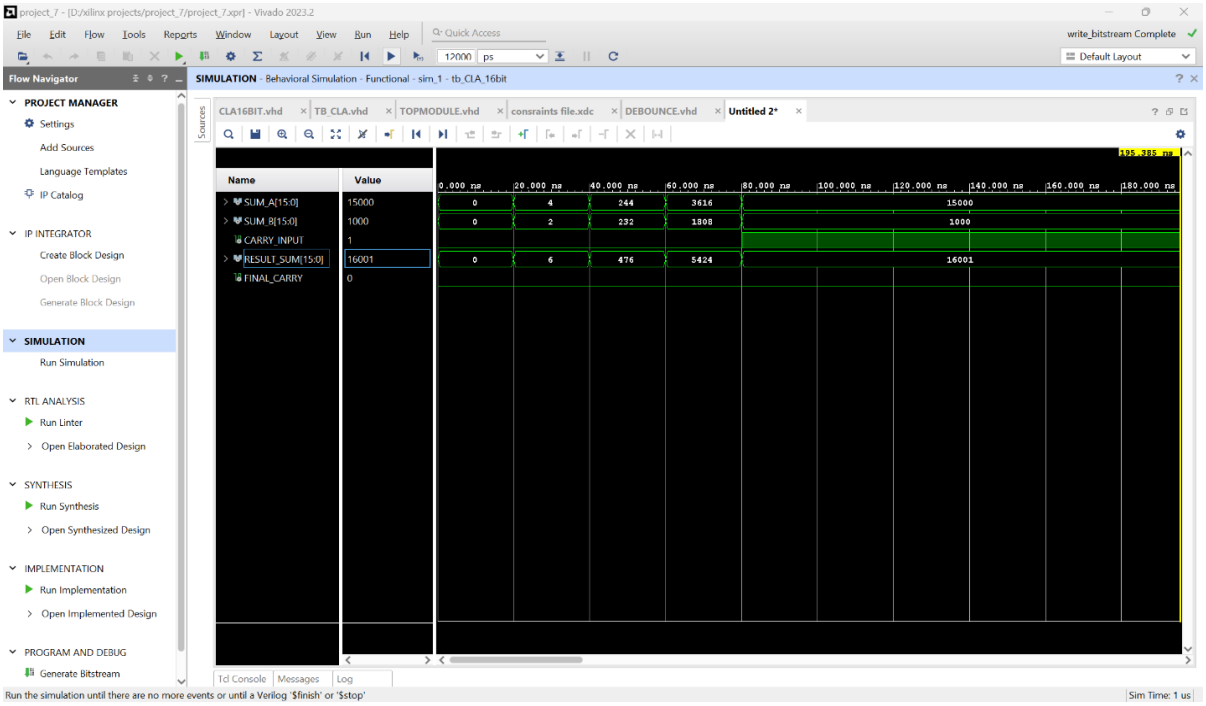


```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity debounce is
    Port (
        clk     : in  STD_LOGIC; -- Clock input
        button  : in  STD_LOGIC; -- Raw button input
        debounced: out STD_LOGIC  -- Debounced button output
    );
end debounce;

architecture Behavioral of debounce is
    signal button_sync_0, button_sync_1 : STD_LOGIC := '0';
    signal counter : STD_LOGIC_VECTOR(19 downto 0) := (others => '0');
    signal stable  : STD_LOGIC := '0';
begin
    -- Synchronise button to clock domain
    process (clk)
    begin
        if rising_edge(clk) then
            button_sync_0 <= button;
            button_sync_1 <= button_sync_0;
        end if;
    end process;

    -- Counter-based debounce logic
    process (clk)
    begin
        if rising_edge(clk) then
            if button_sync_1 /= stable then
                counter <= std_logic_vector(unsigned(counter) + 1); -- Corrected addition
                if counter = x"FFFFF" then
                    stable <= button_sync_1;
                    counter <= (others => '0');
                end if;
```

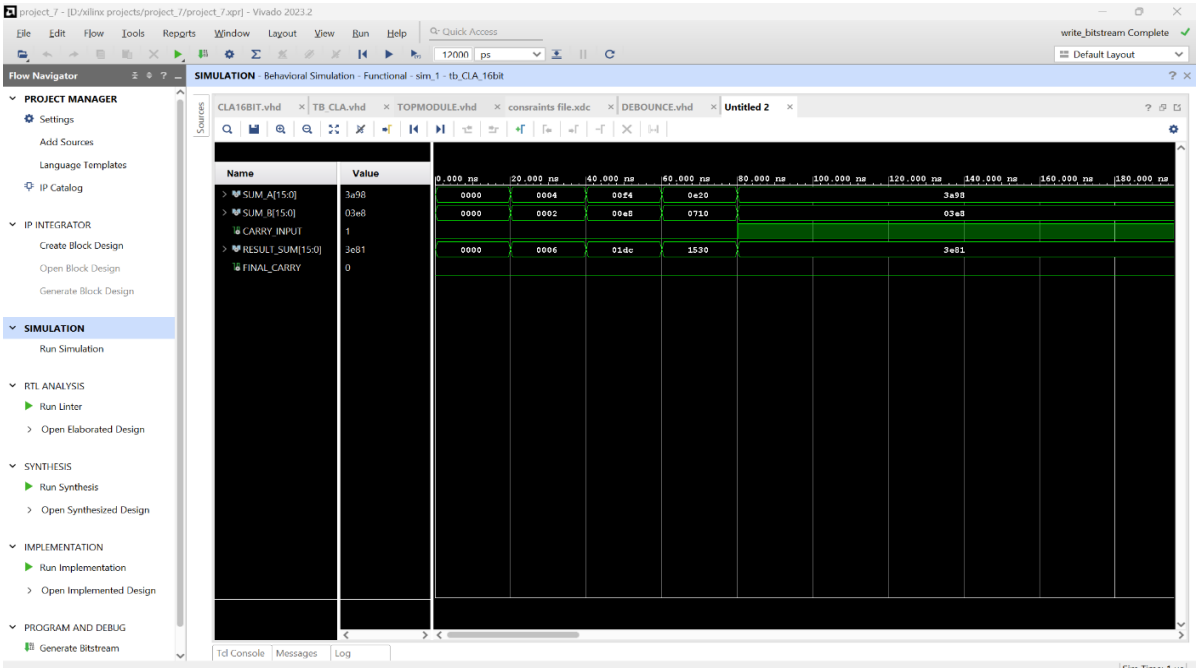
SIMULATION WAVEFORM:

The test cases mentioned in the test bench are shown in the simulation waveform. After Run simulation the waveform displays.

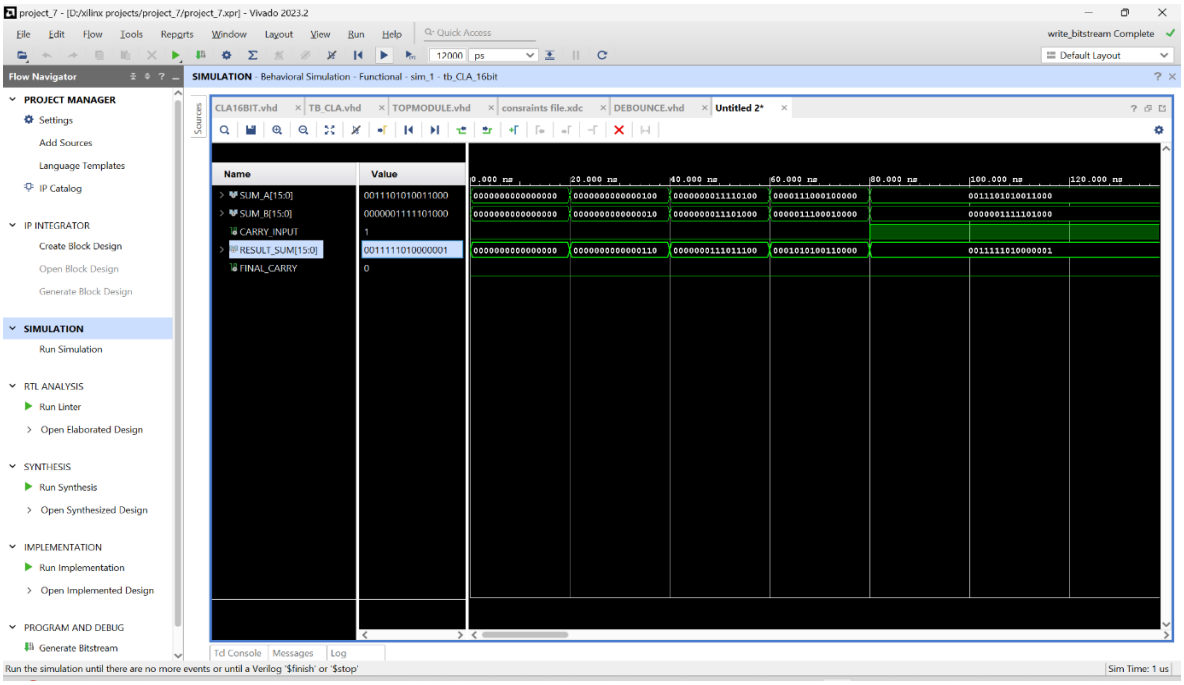
UNSIGNED DECIMAL:



HEXADECIMAL:

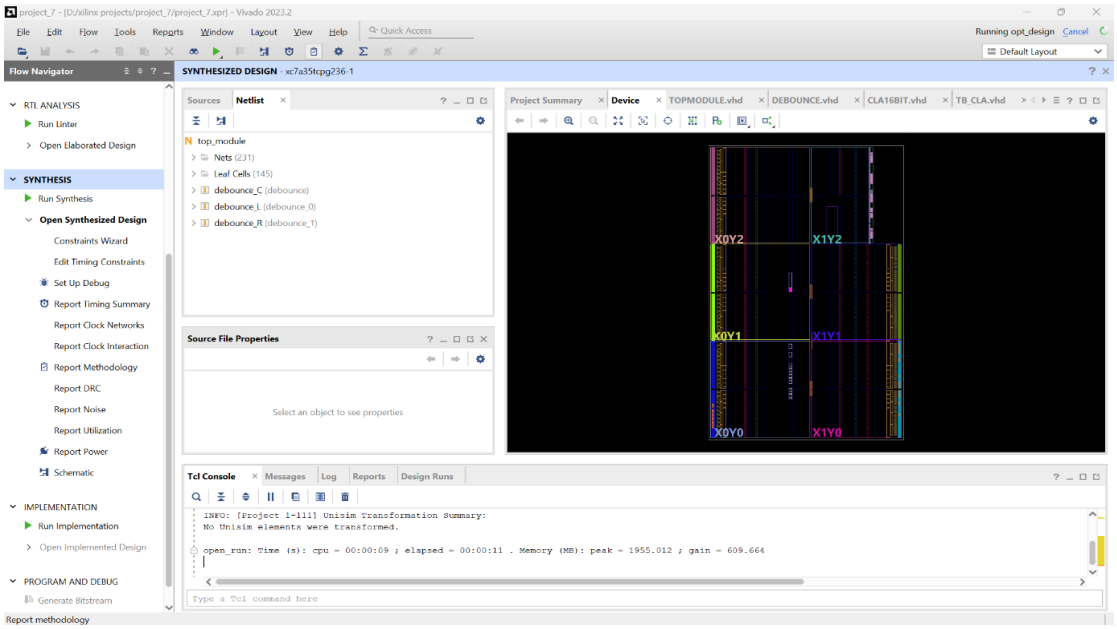


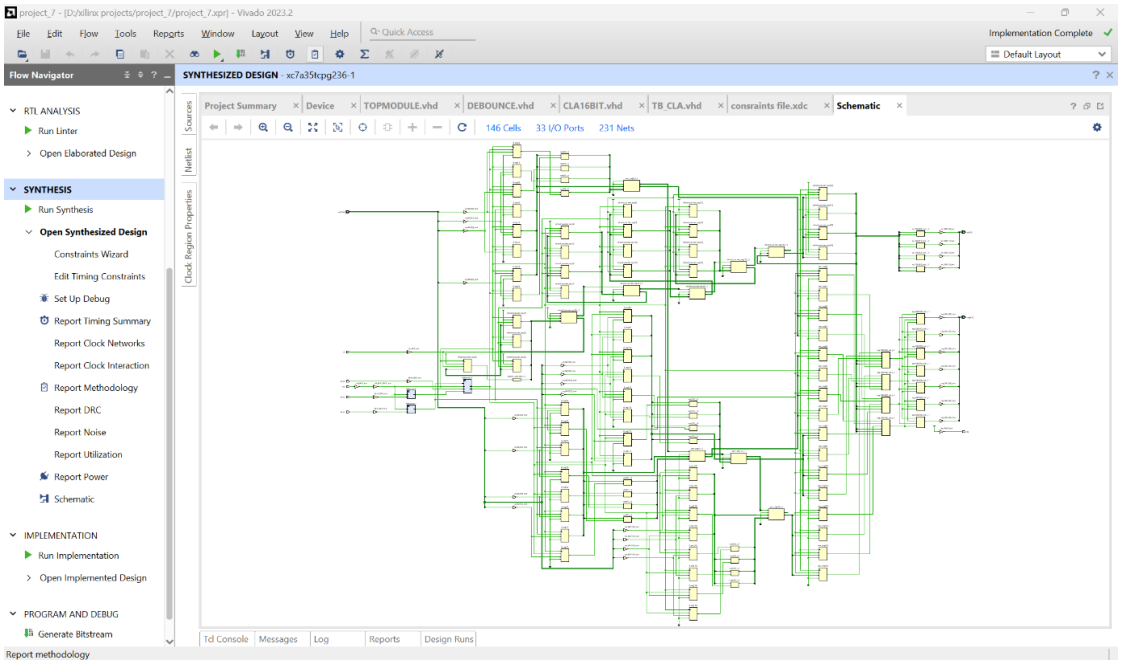
BINARY:


project_7 - [D:/xilinx projects/project_7/project_7.xpr] - Vivado 2023.2
File Edit Flow Tools Reports Window Layout View Run Help
write_bitstream Complete
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Run Linter
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
Generate Bitstream
SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_CLA_16bit
CLA16BIT.vhd
TB_CLA.vhd
TOPMODULE.vhd
consraints file.xdc
DEBOUNCE.vhd
Untitled 2*
Name
Value
SUM_A[15:0]
0011101010011000
SUM_B[15:0]
0000001111101000
CARRY_INPUT
1
RESULT_SUM[15:0]
0011111010000001
FINAL_CARRY
0
0.000 ns
20.000 ns
40.000 ns
60.000 ns
80.000 ns
100.000 ns
120.000 ns
Tcl Console
Messages
Log
Run the simulation until there are no more events or until a Verilog '$finish' or '$stop'
Sim Time: 1 us

SYNTHESIS:

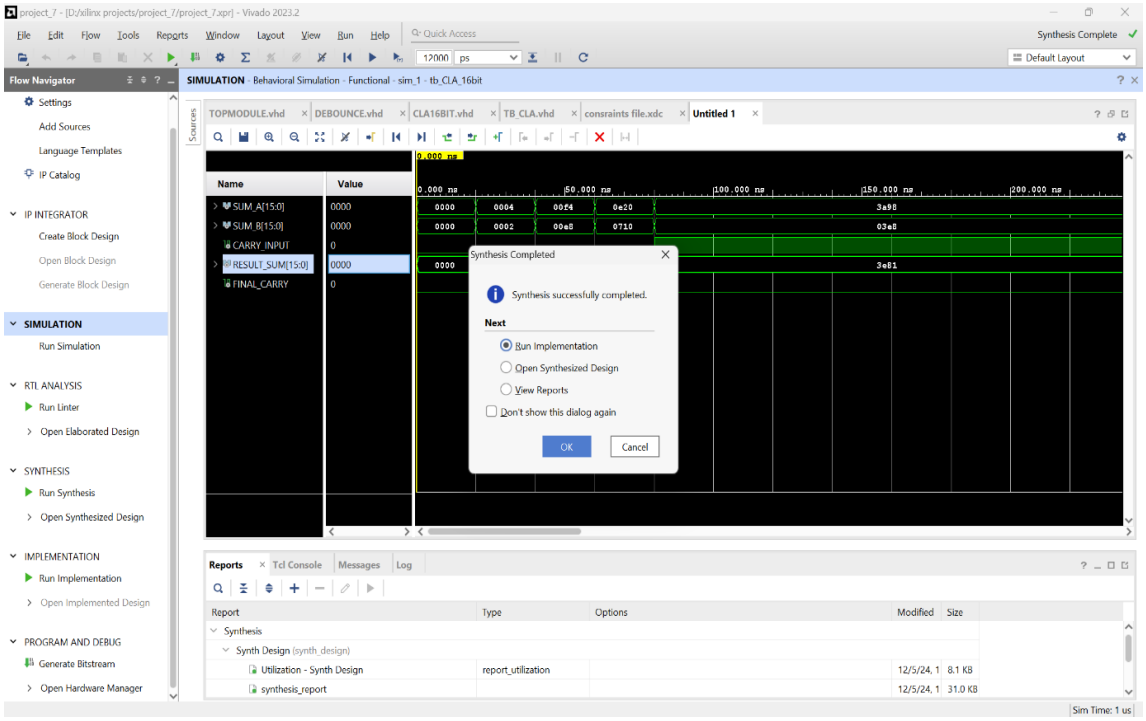
After simulation, I have done **Run synthesis** in order to obtain the synthesized image.

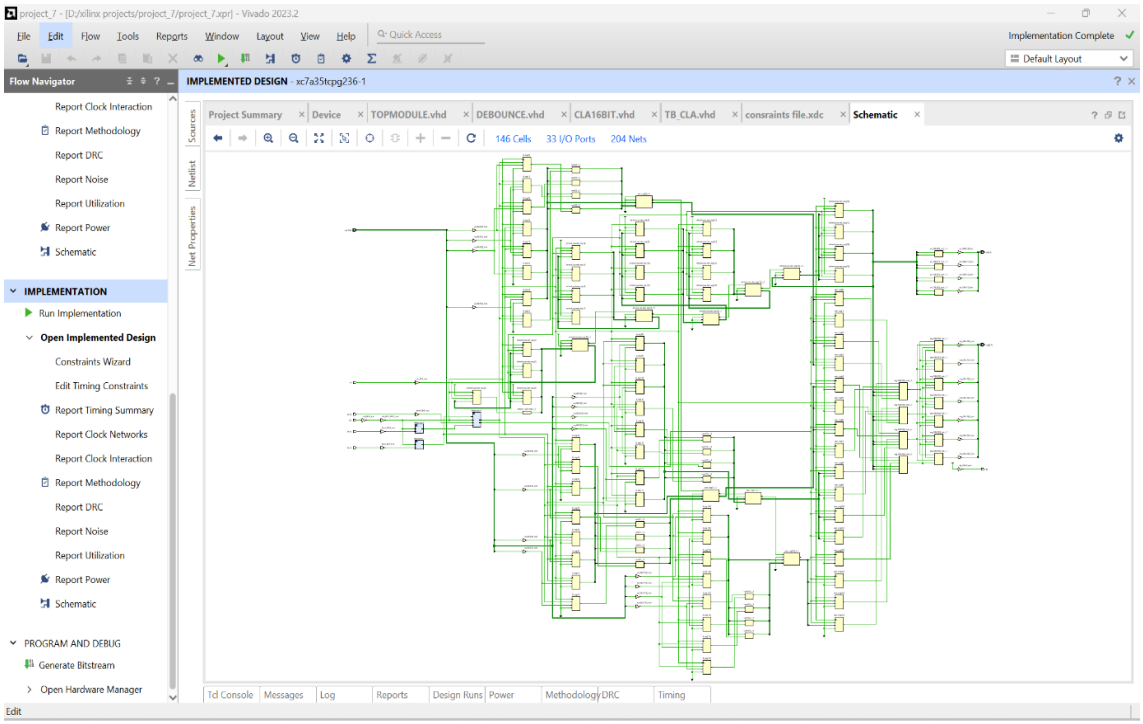




IMPLEMENTATION:

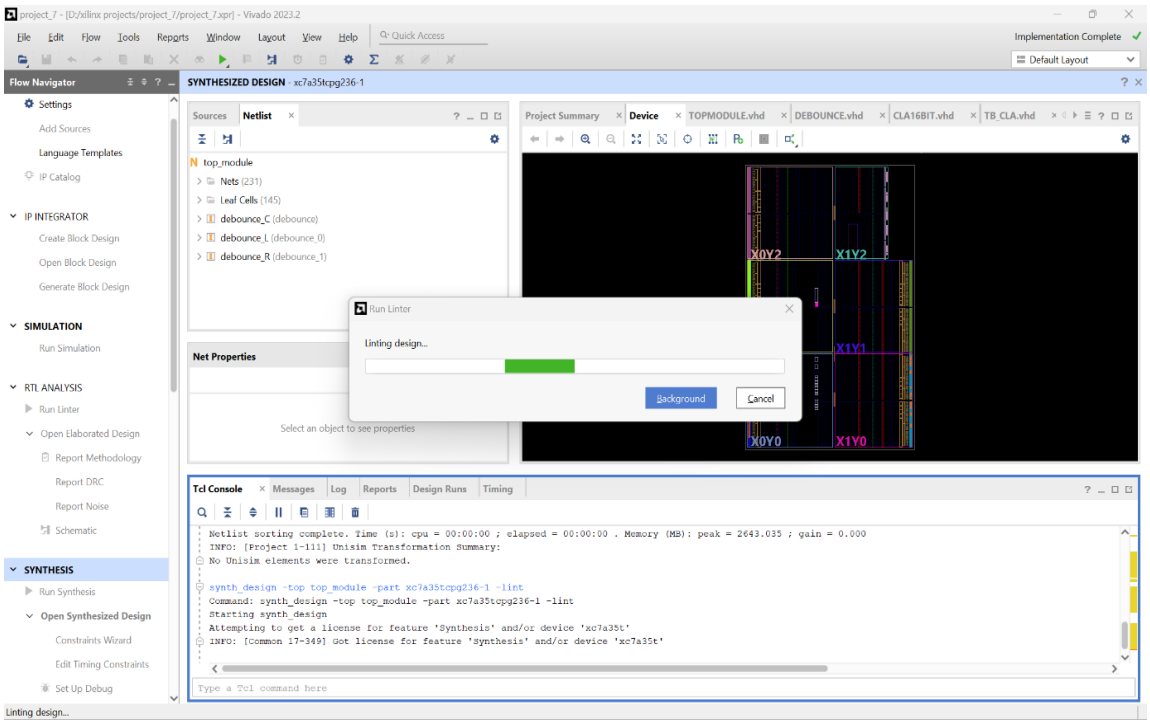
After synthesis, I have done **Run Implementation**.

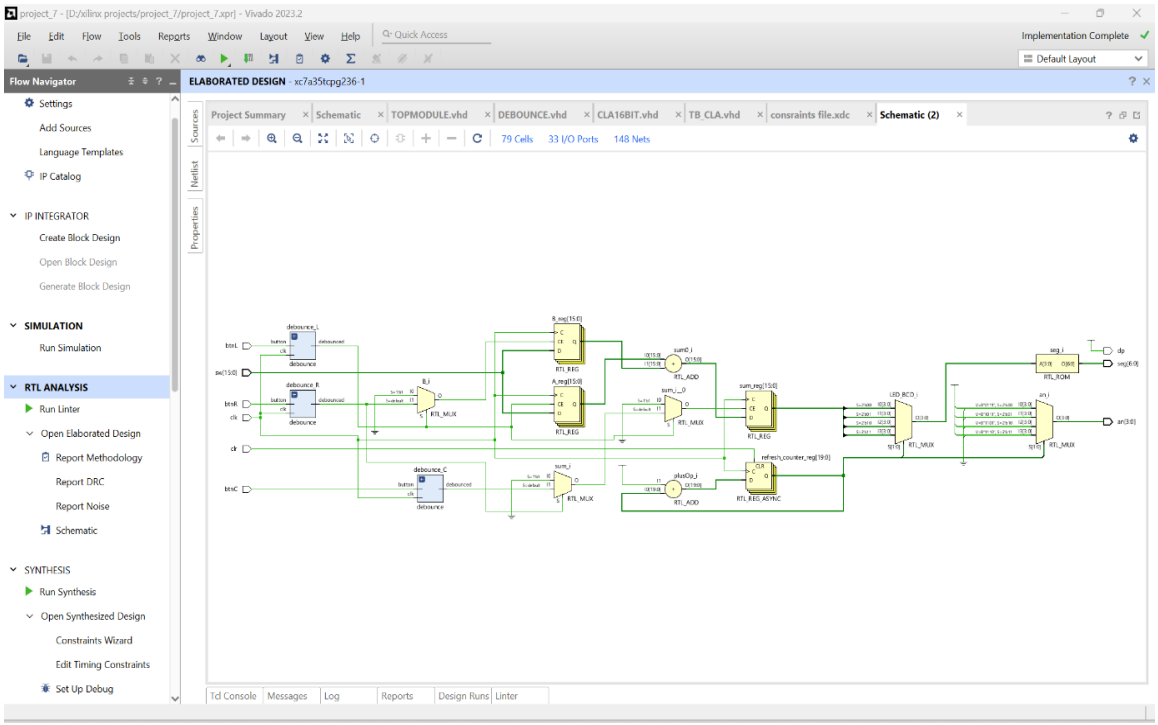




RTL ANALYSIS:

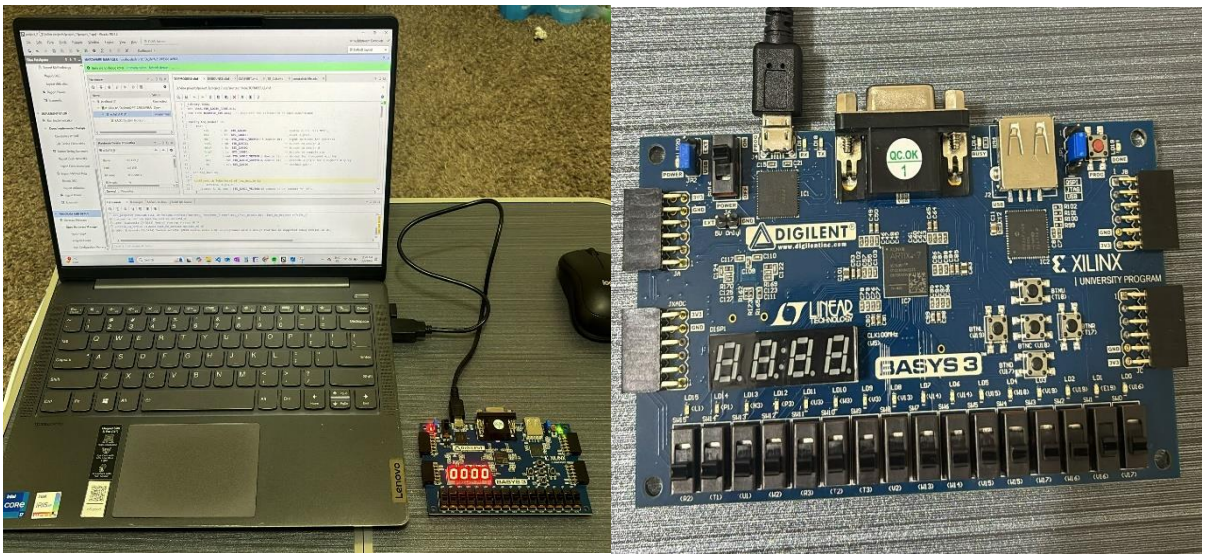
In order to get RTL Design, I have done Run Linter then Schematic to obtain the RTL schematic.





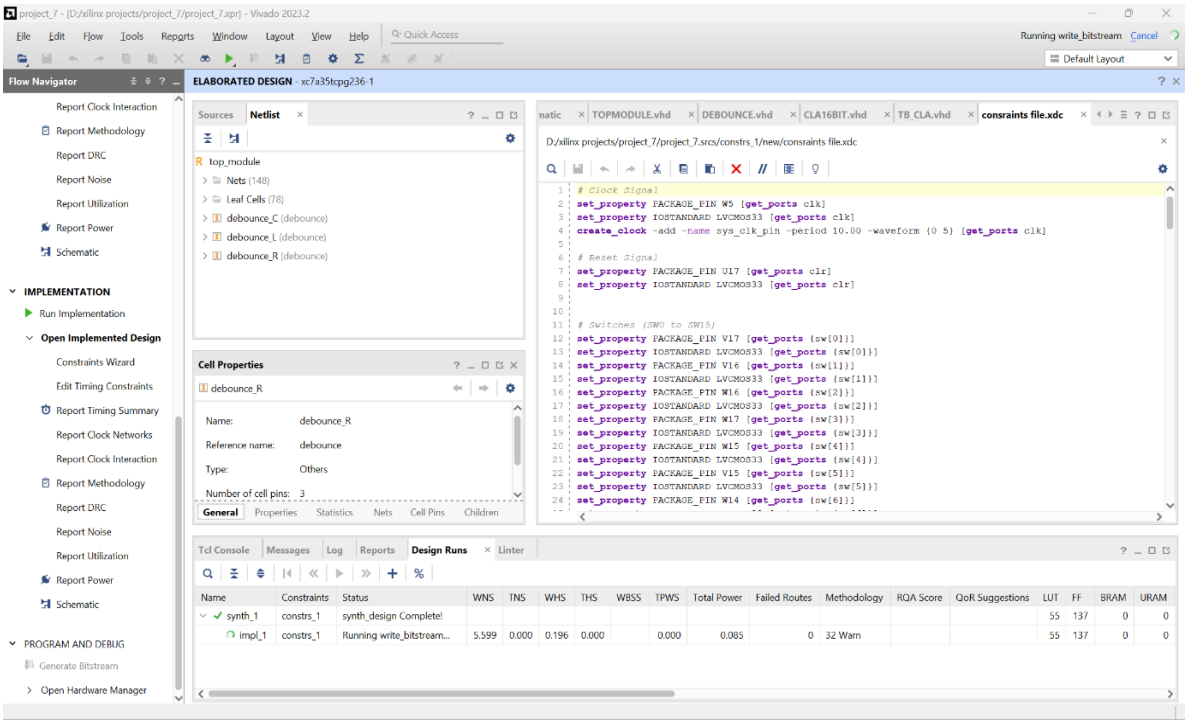
EXPERIMENTAL SETUP:

The below is the experimental setup for implementation.

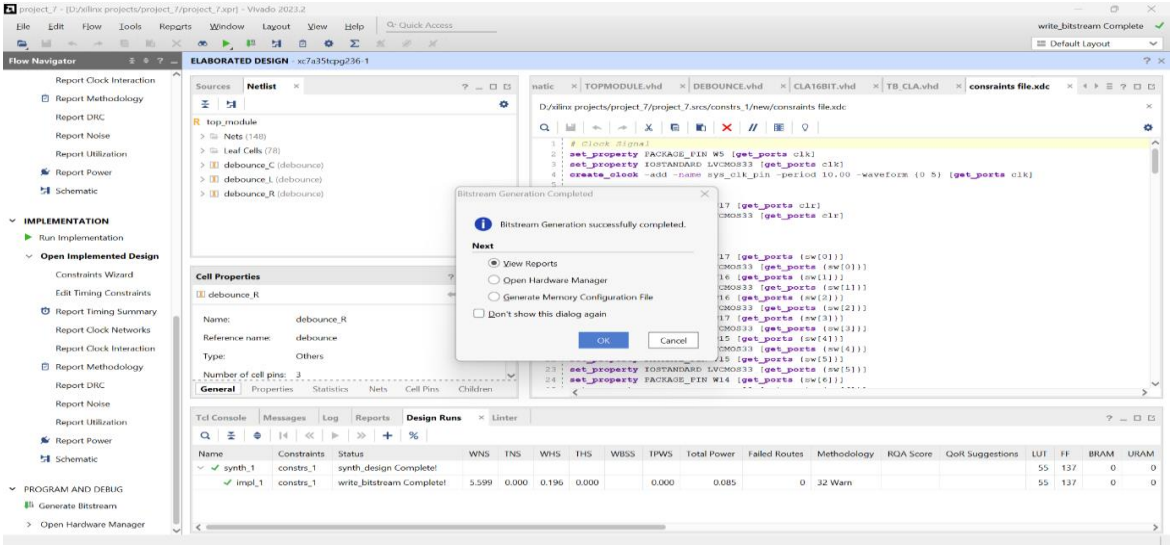


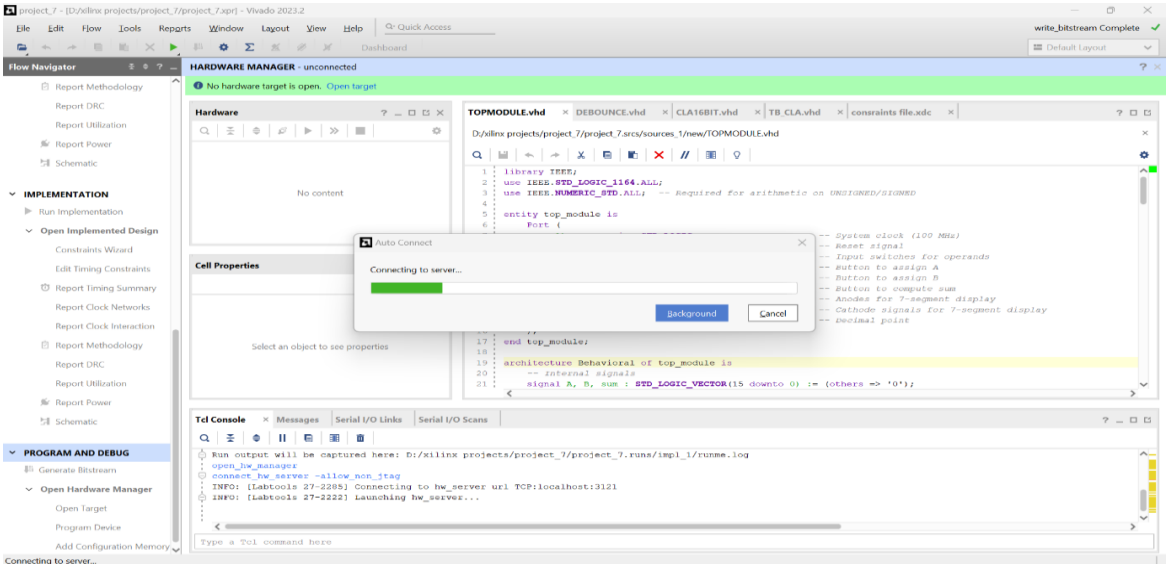
BITSTREAM:

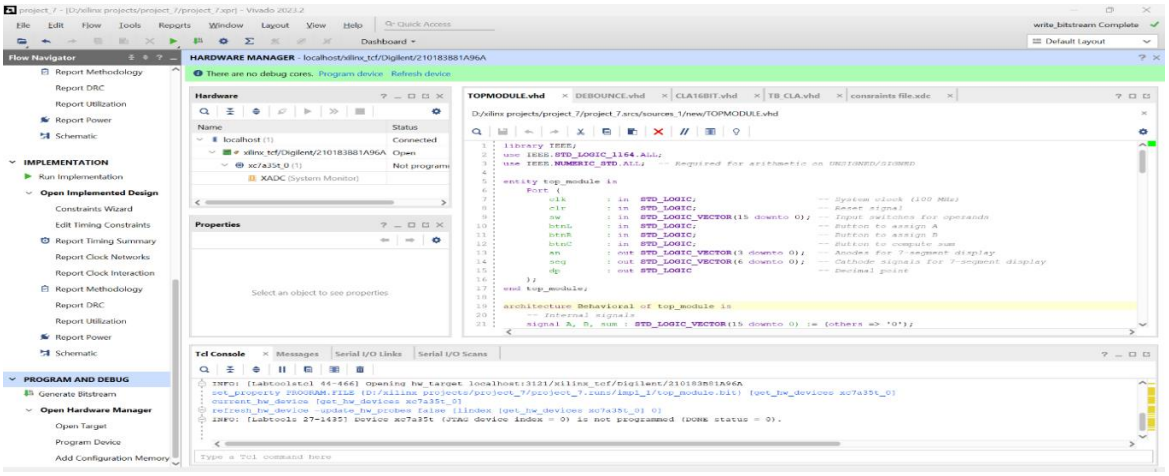
After connecting the basys3 board to laptop and after simulation, I have clicked on Generate bitstream.



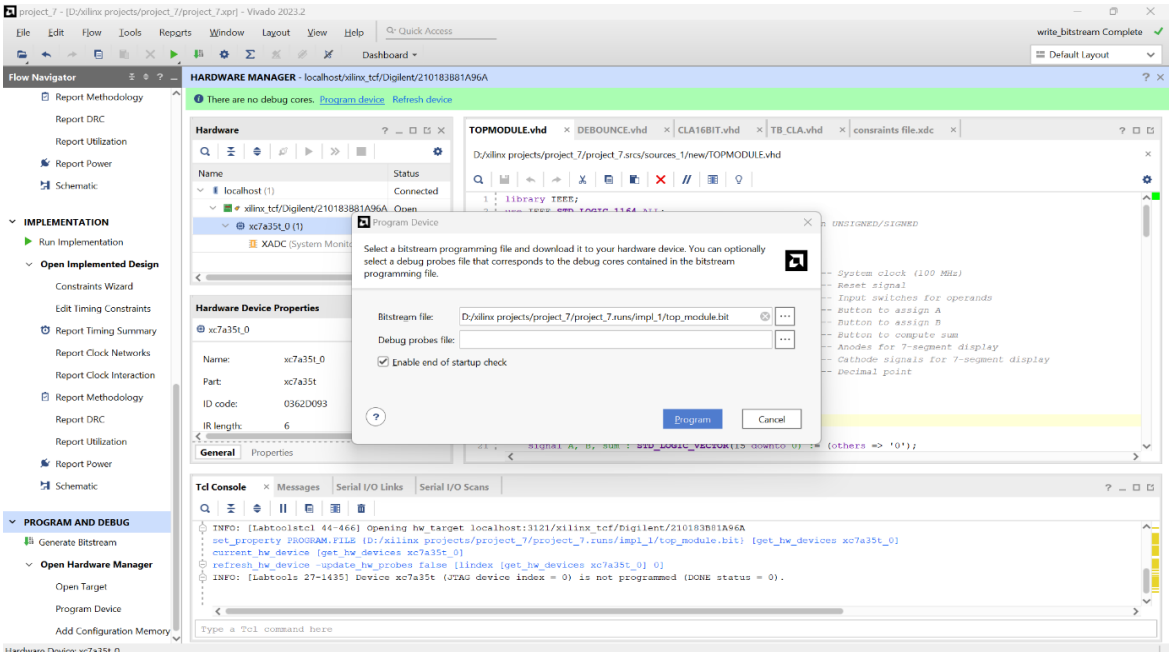
After generating the bitstream, I clicked on open target and then program device.





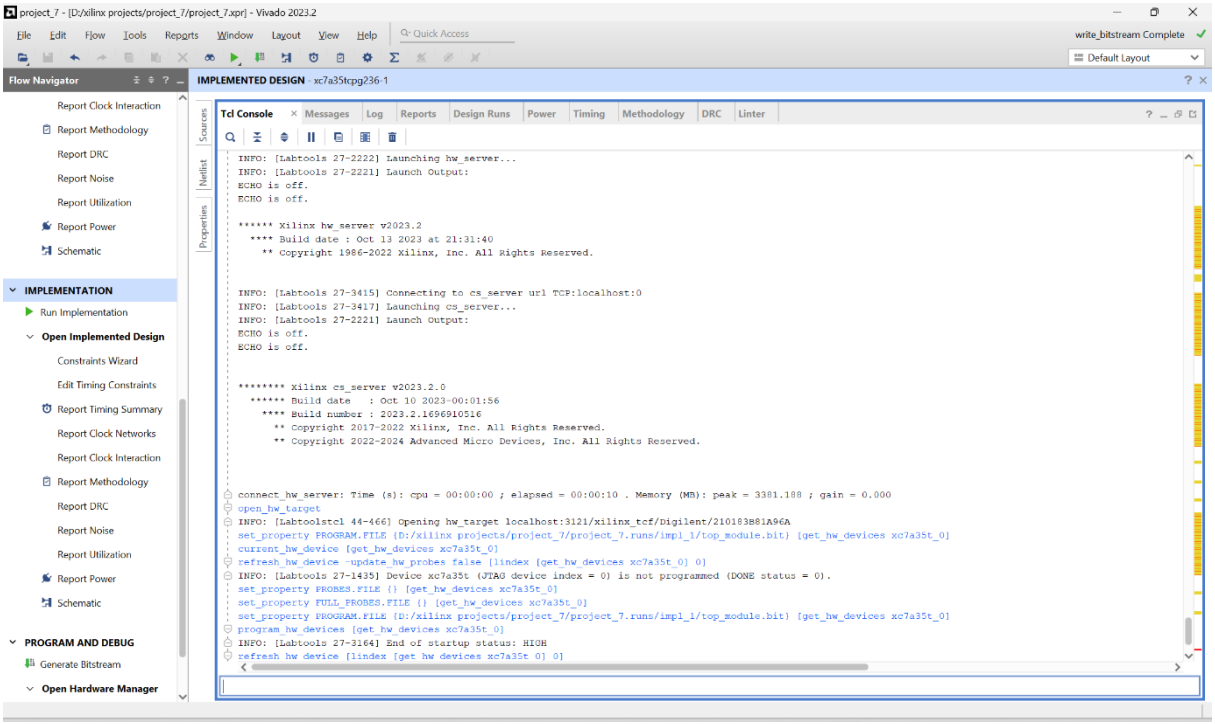


After successfully generating the bitstream and device is programmed, the test cases are given and the output is displayed on 7-segment display.

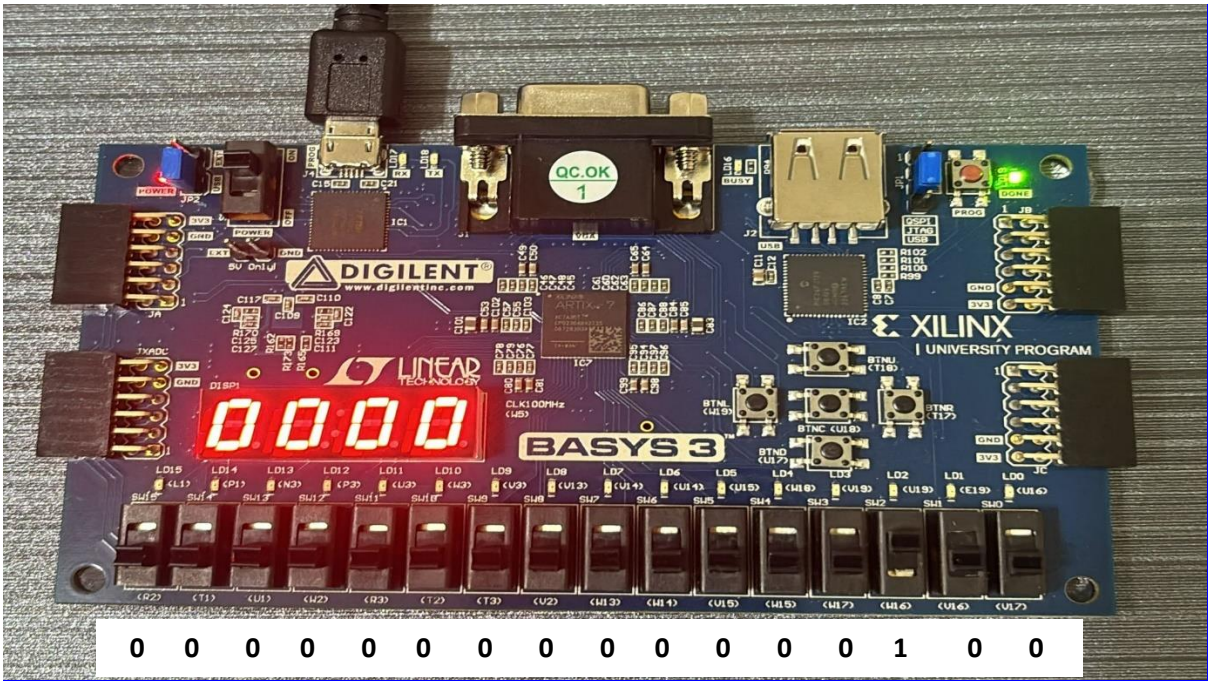


TCL CONSOLE:

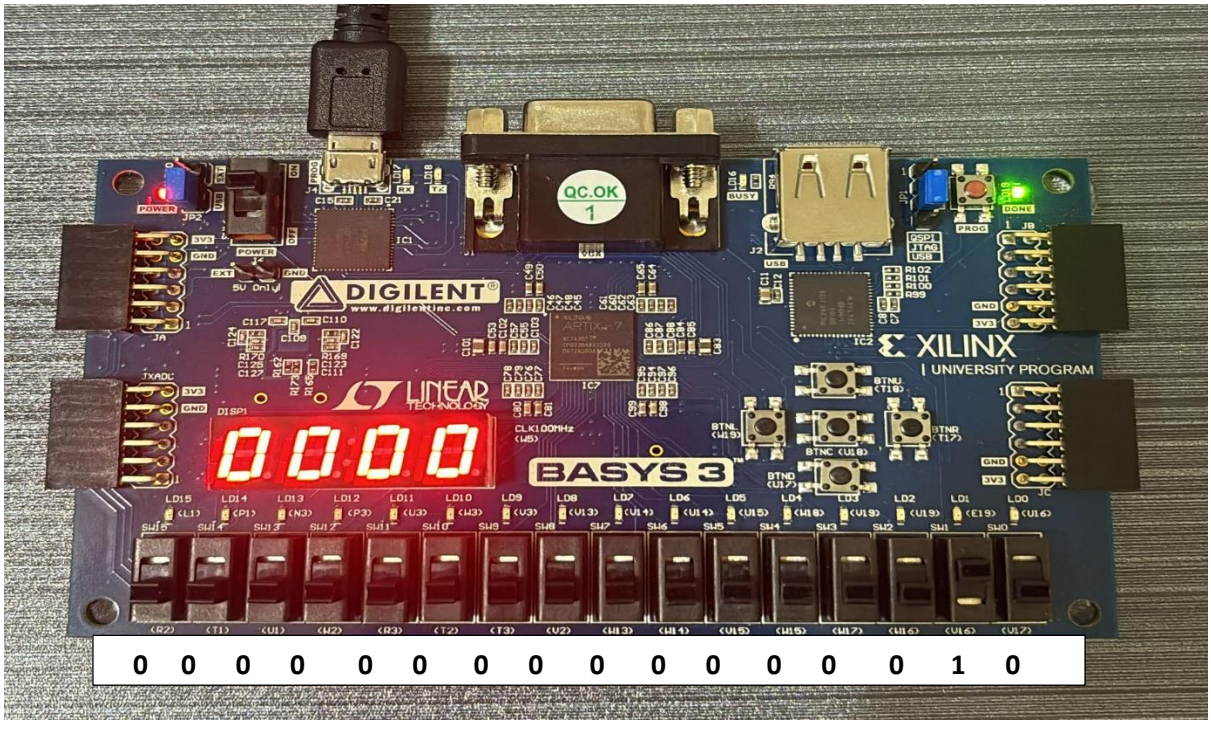
The below is a TCL Console window after connecting the board, generating bitstream and running the testcases.



TESTCASE-1:

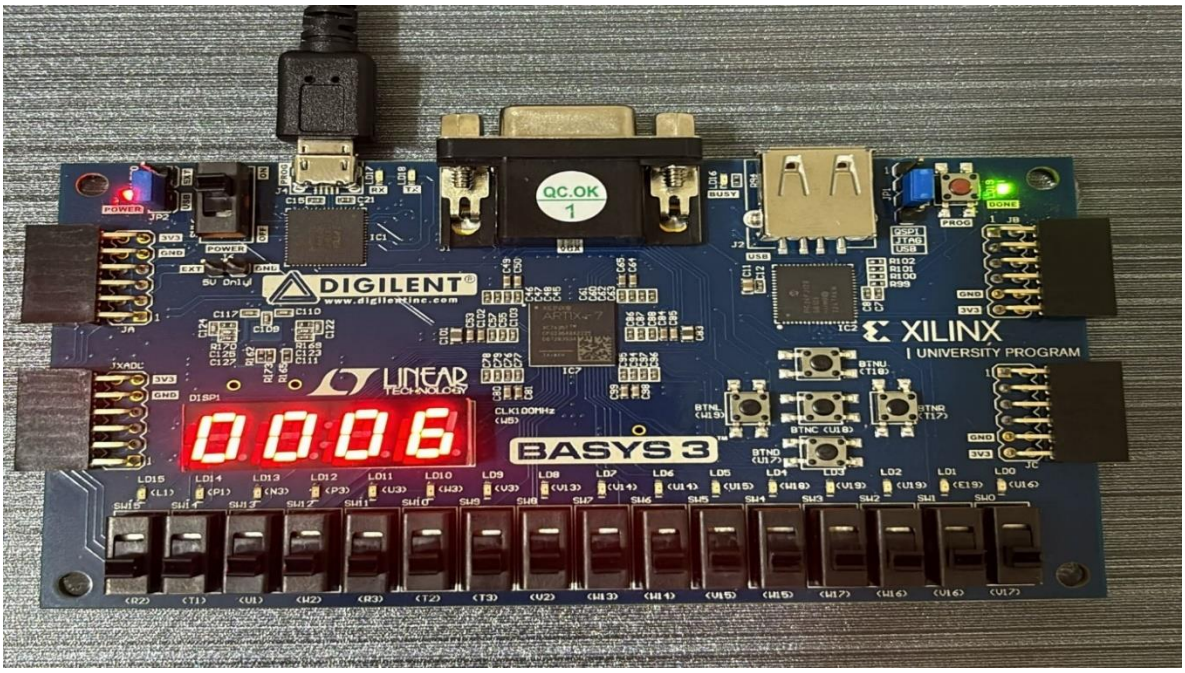


SUM_A = 0X0004



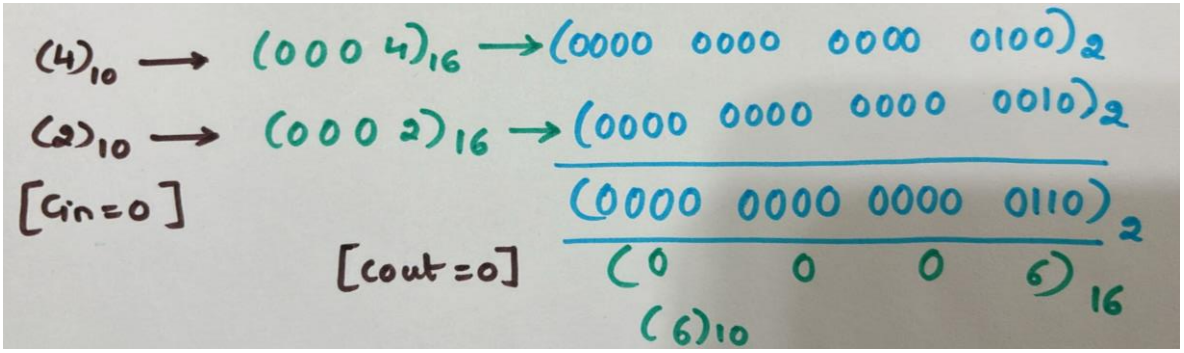
SUM_B = 0X0002

CARRY_INPUT = 0



RESULT_SUM = 0x0006

FINAL_CARRY = 0



Result

Hex value:
0004 + 0002 = **6**

Decimal value:
4 + 2 = **6**

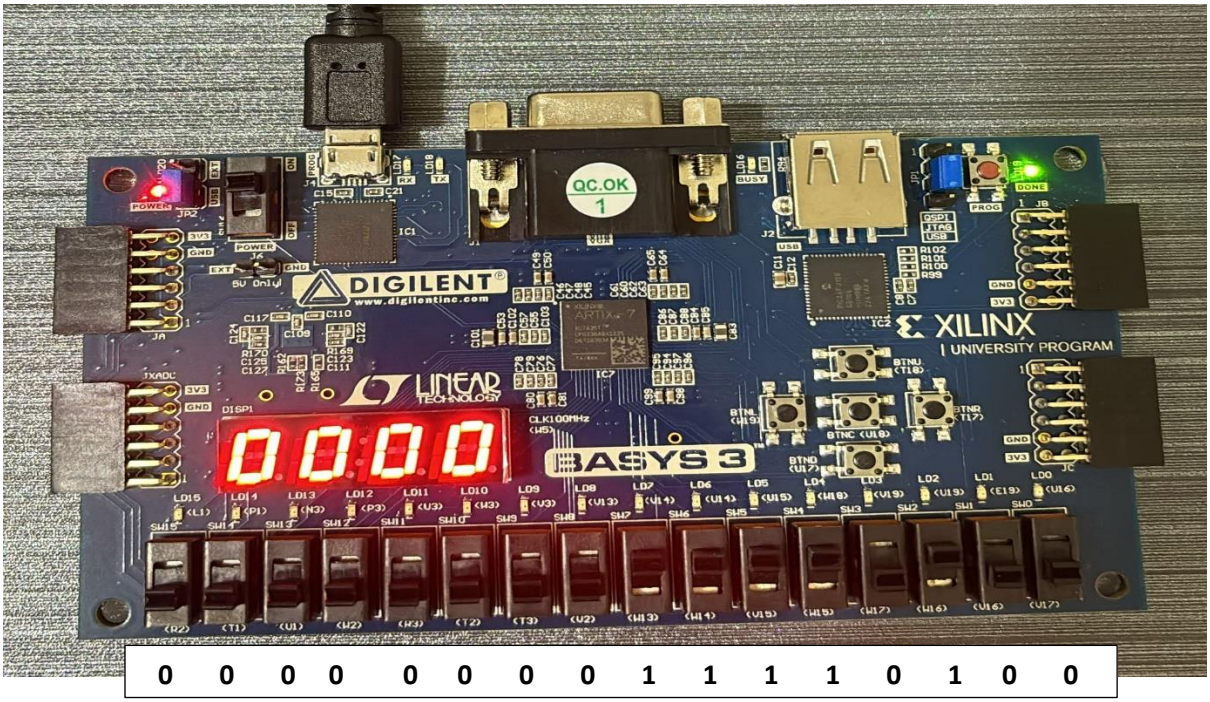
0004 + 0002 = ?

Calculate Clear

DEMONSTARTION VIDEO:

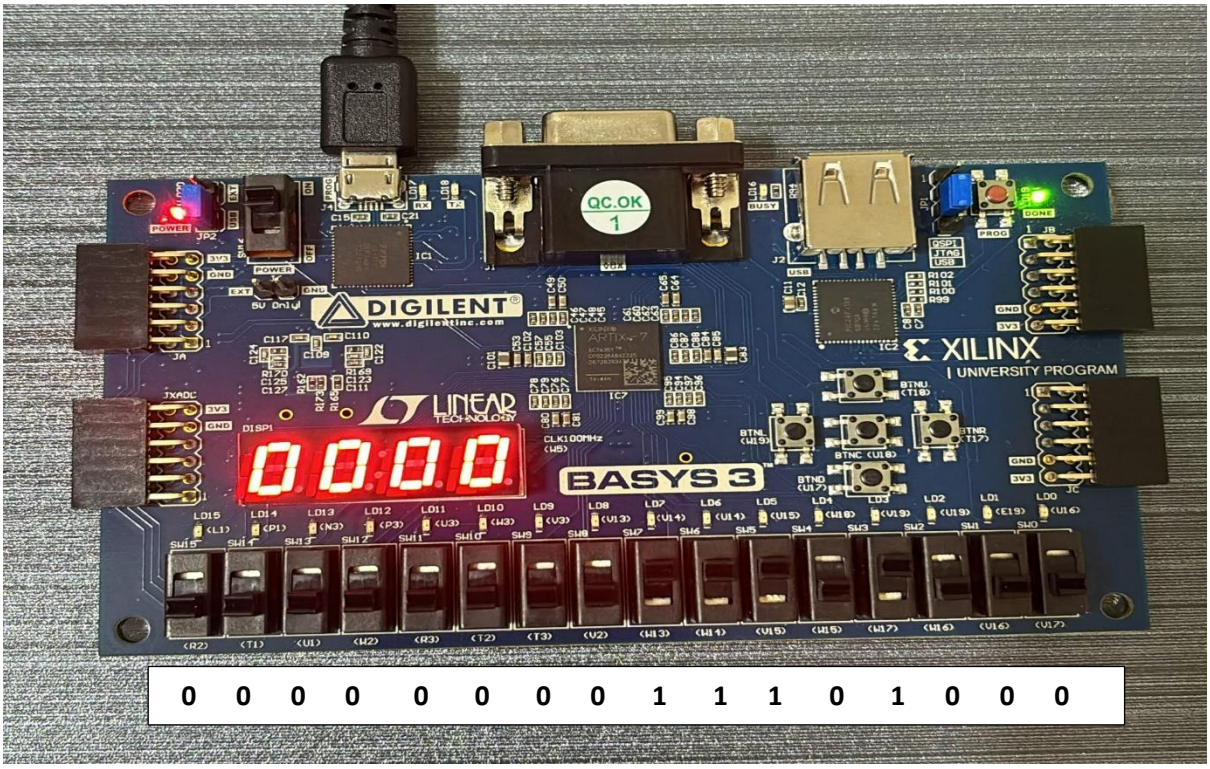
https://drive.google.com/file/d/1RDyDGR4T66bj5breZdT2_fs-lnqmbVVG/view?usp=drive_link

TESTCASE-2:



0	0	0	0	0	0	0	0	1	1	1	1	0	1	0	0

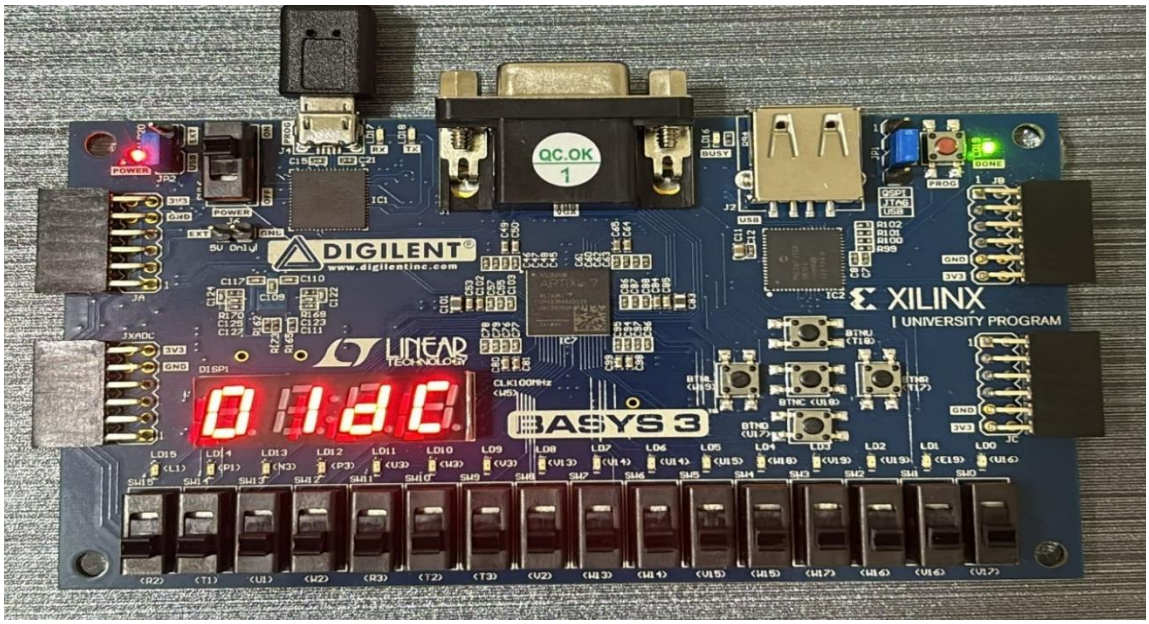
SUM_A = 0X00f4



0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0

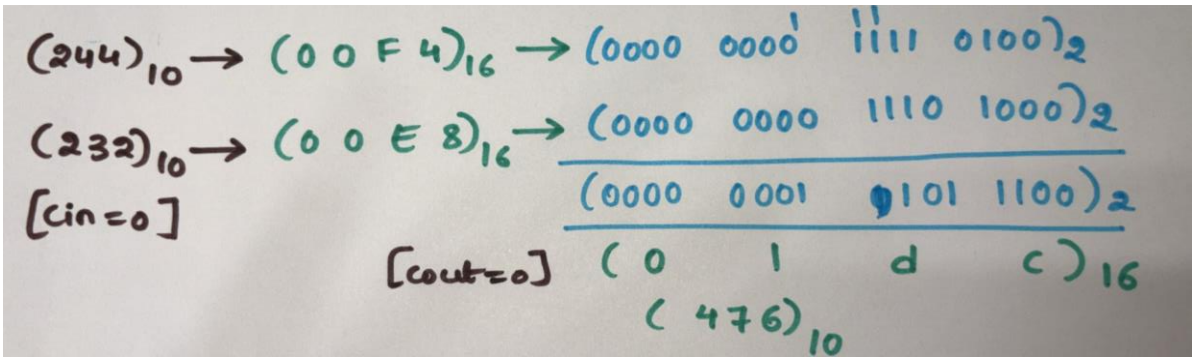
SUM_B = 0X00e8

CARRY_INPUT = 0



RESULT_SUM = 0x01dc

FINAL_CARRY = 0



Result

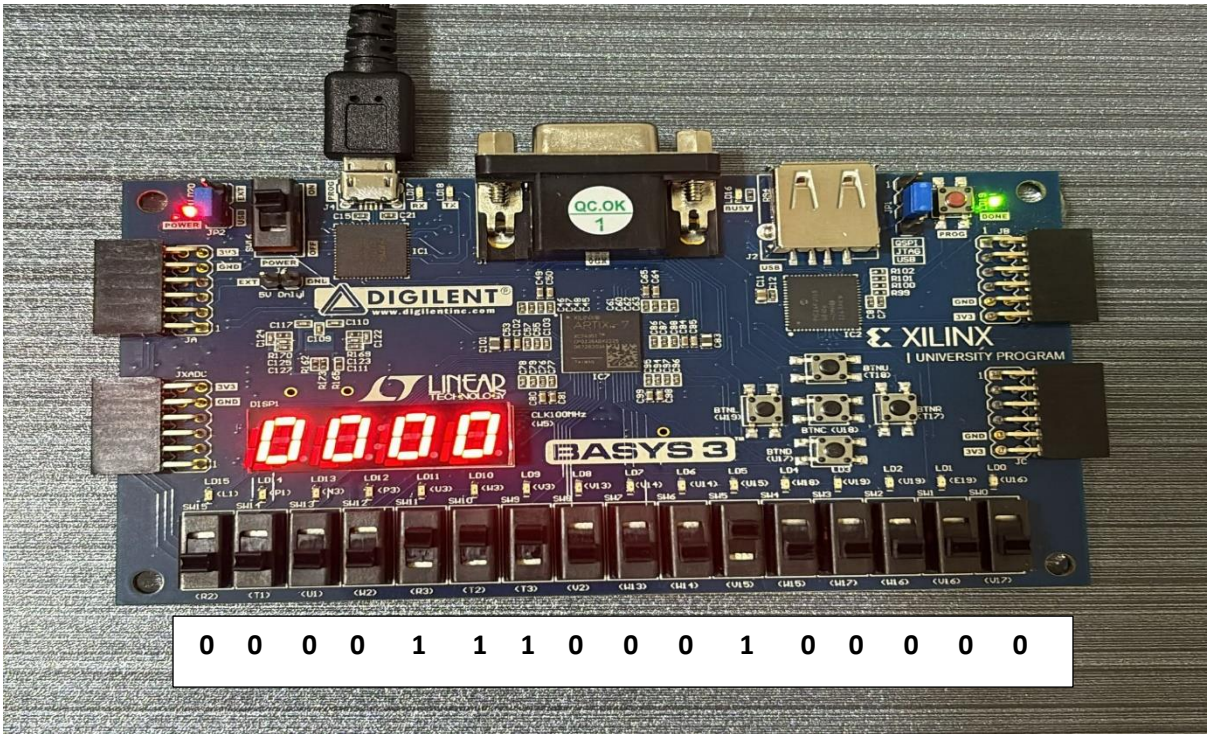
Hex value:
00F4 + 00E8 = **1DC**

Decimal value:
244 + 232 = **476**

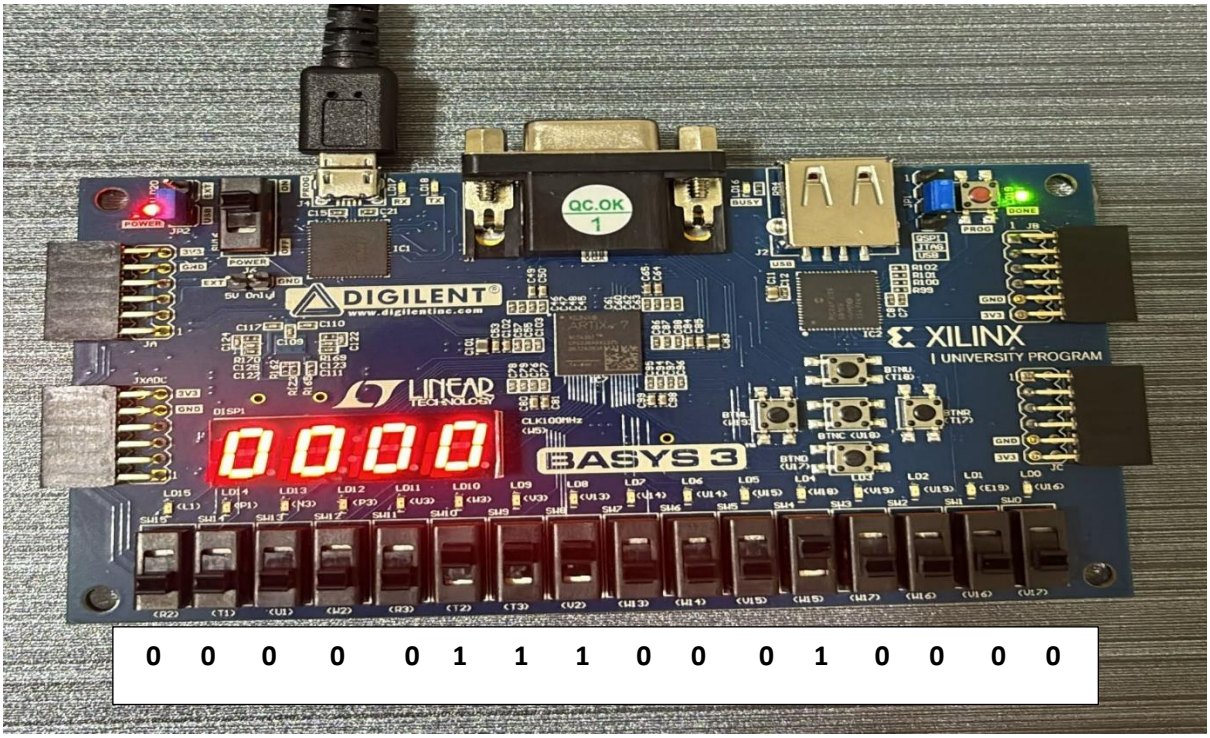
00F4 + 00E8 = ?

Calculate Clear

TESTCASE-3:

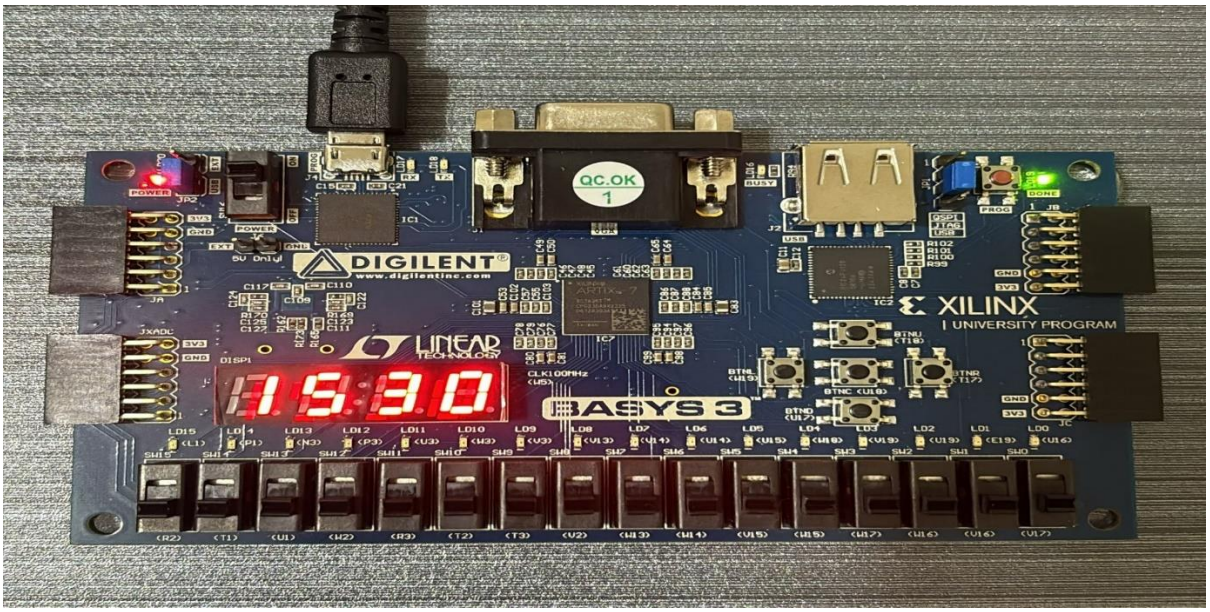


SUM_A = 0X0e20



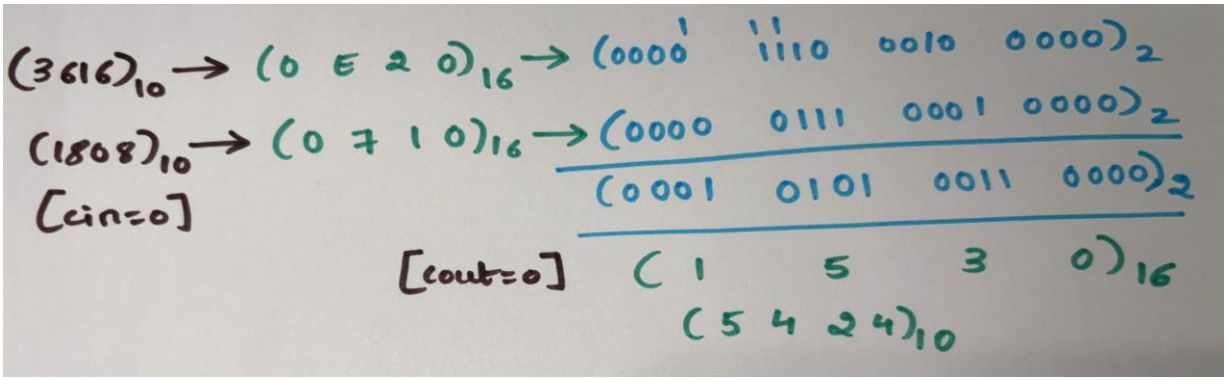
SUM_B = 0X0710

CARRY_INPUT = 0



RESULT_SUM = 0x1530

FINAL_CARRY = 0



Result

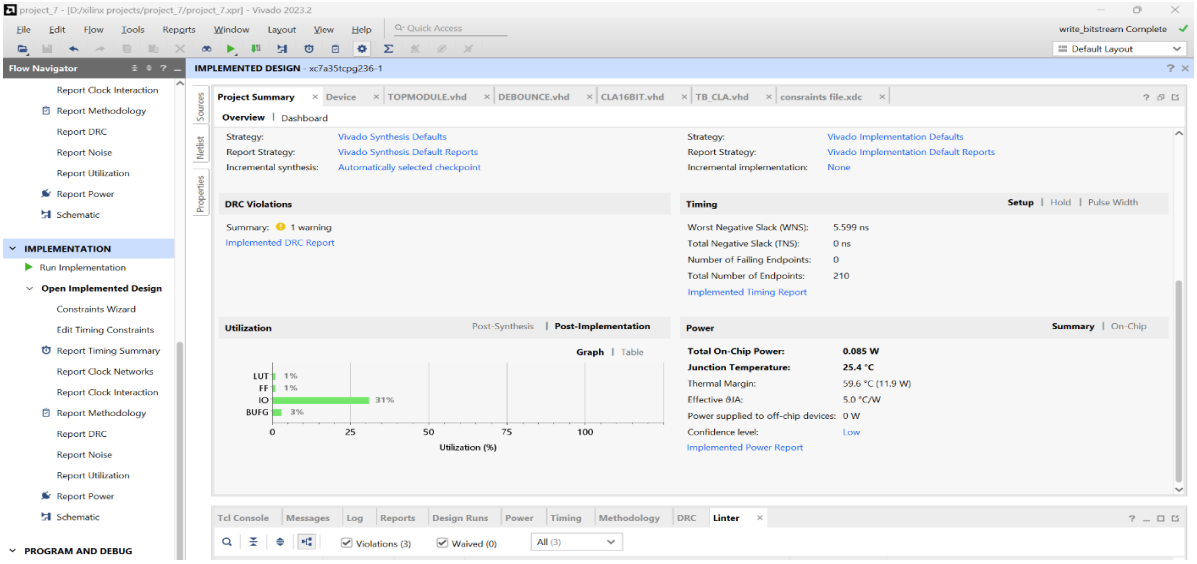
Hex value:
0E20 + 0710 = **1530**

Decimal value:
3616 + 1808 = **5424**

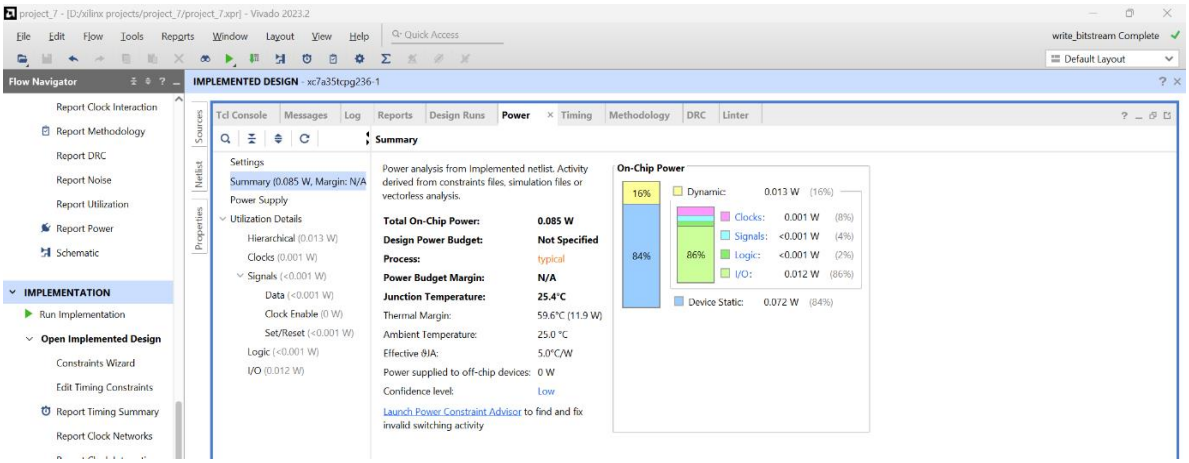
0E20 + 0710 = ?

Calculate Clear

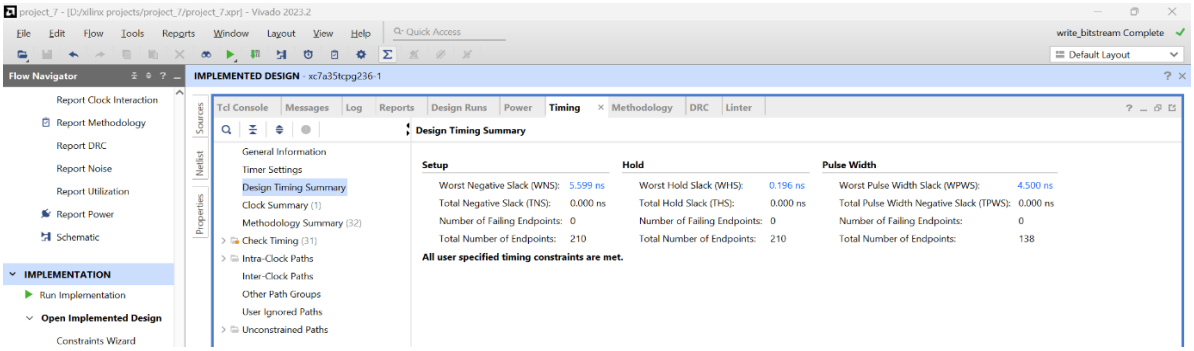
SUMMARY:



POWER ANALYSIS:



TIMING ANALYSIS:



POST-IMPLEMENTATION:

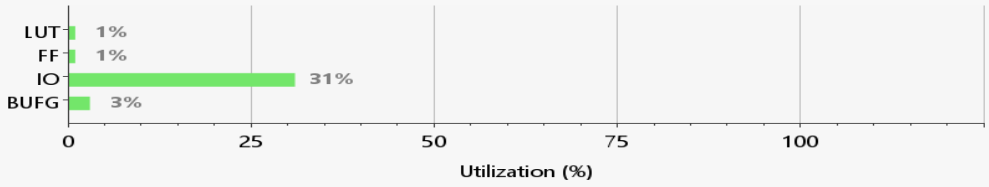
Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	55	20800	0.26
FF	137	41600	0.33
IO	33	106	31.13
BUFG	1	32	3.13

Utilization Post-Synthesis | **Post-Implementation**

Graph | Table



POST-SYNTHESIS:

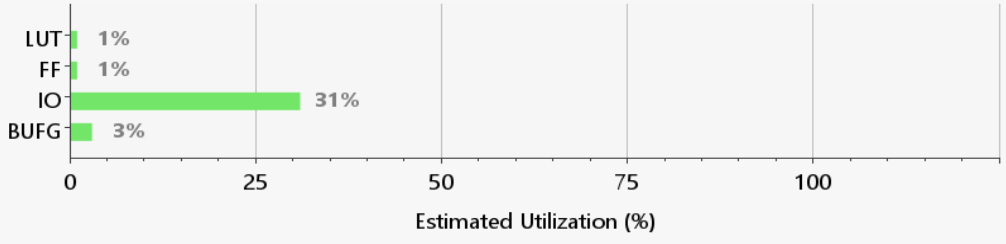
Utilization **Post-Synthesis** | Post-Implementation

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	55	20800	0.26
FF	137	41600	0.33
IO	33	106	31.13
BUFG	1	32	3.13

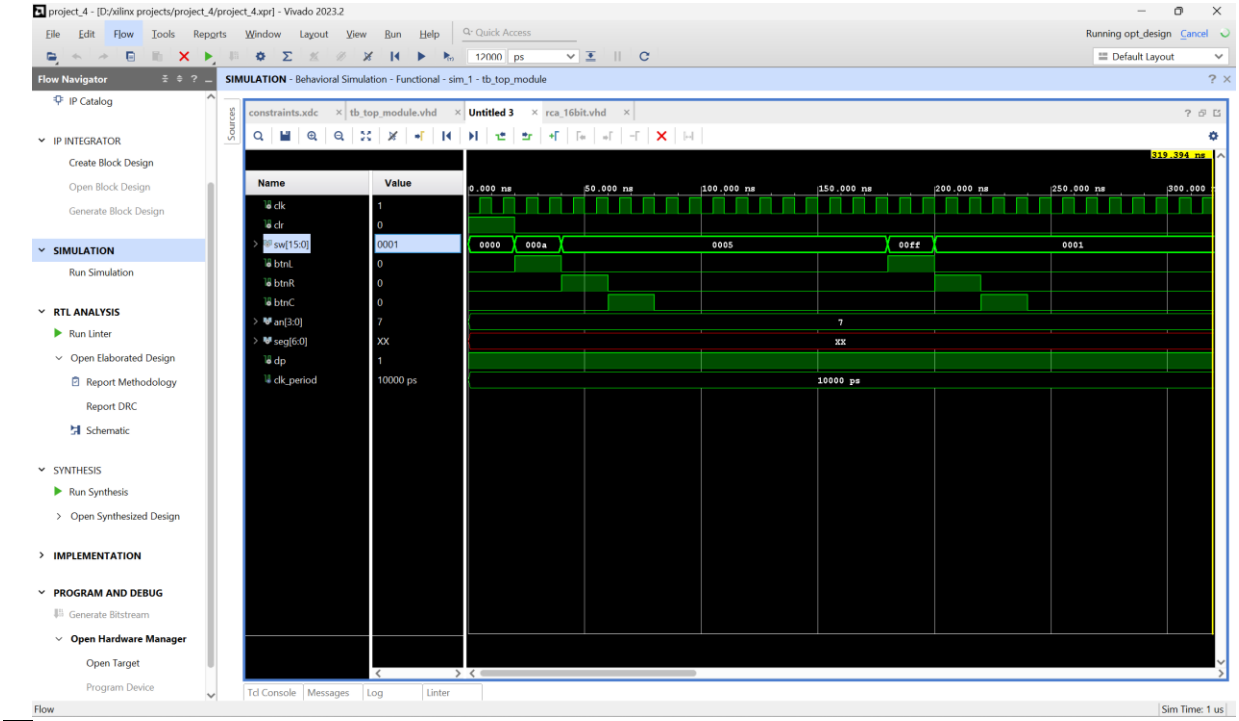
Utilization **Post-Synthesis** | Post-Implementation

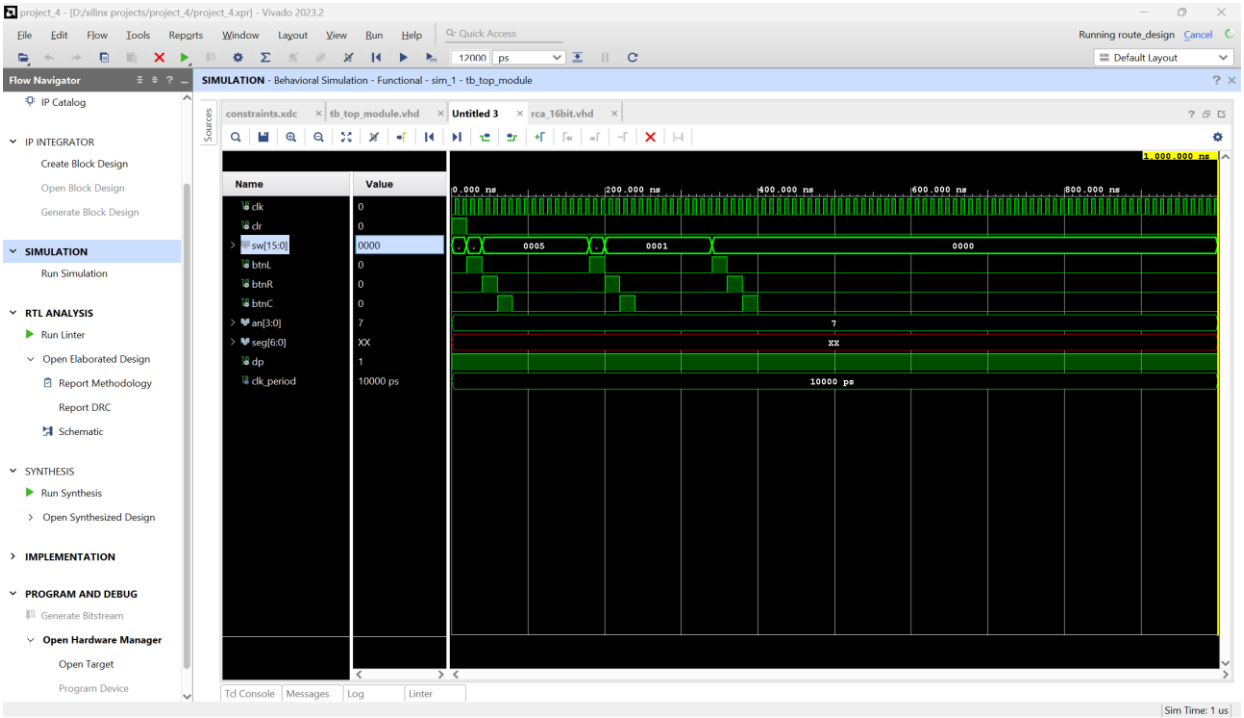
Graph | Table



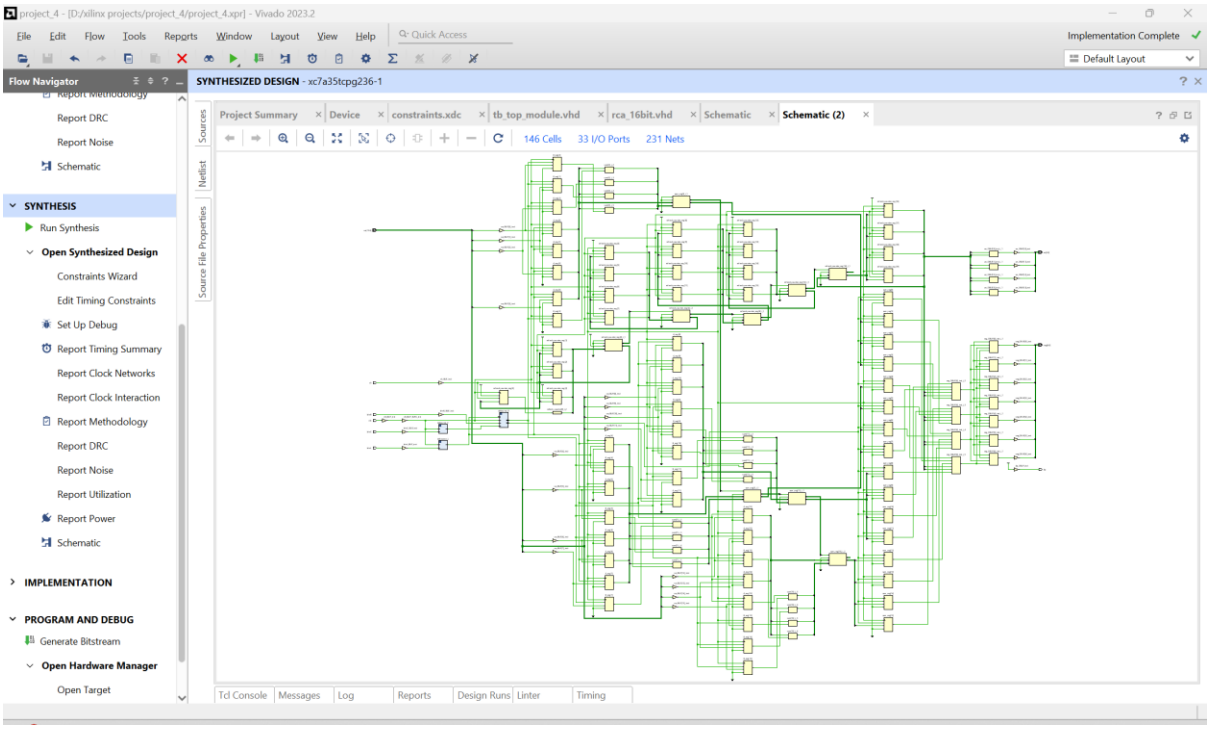
RCA IMPLEMENTATION:

SIMULATION:

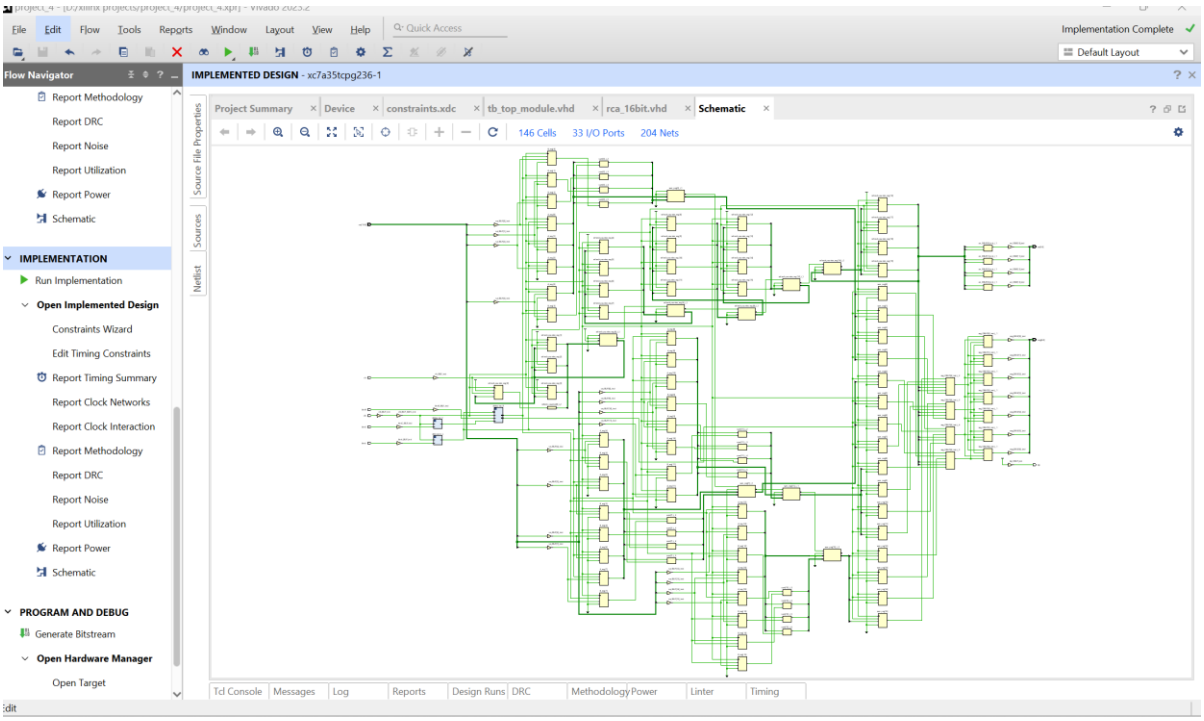




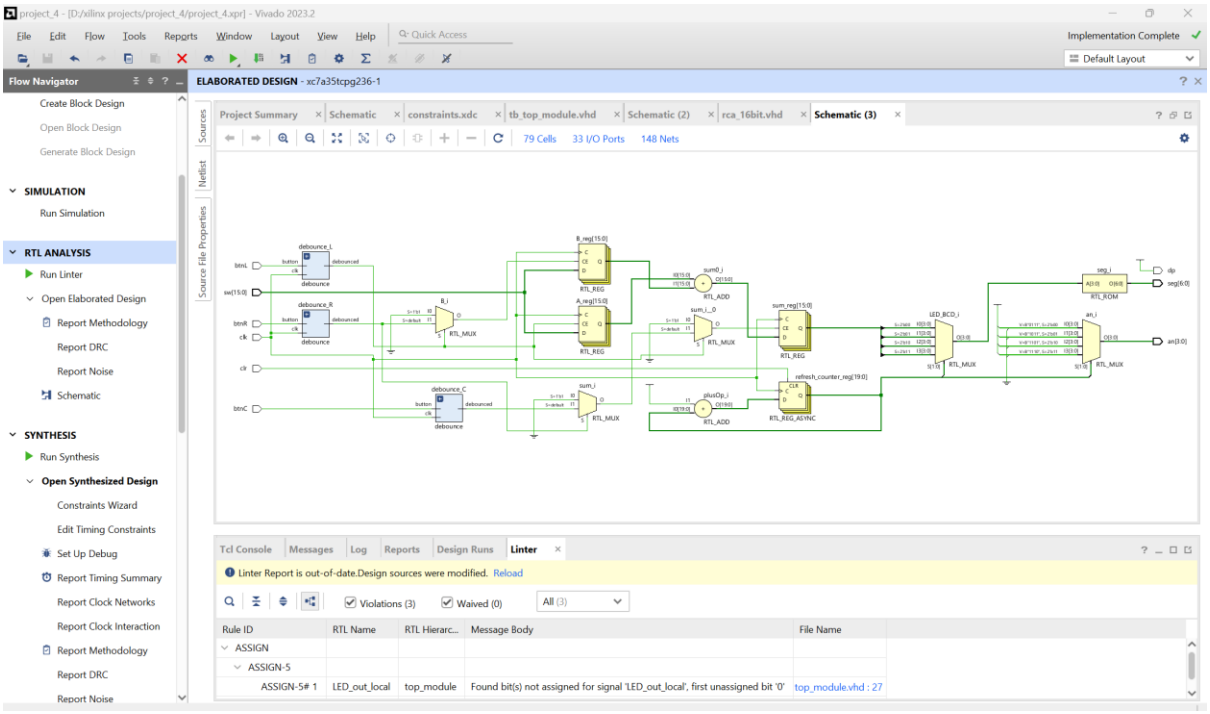
SYNTHESIS:



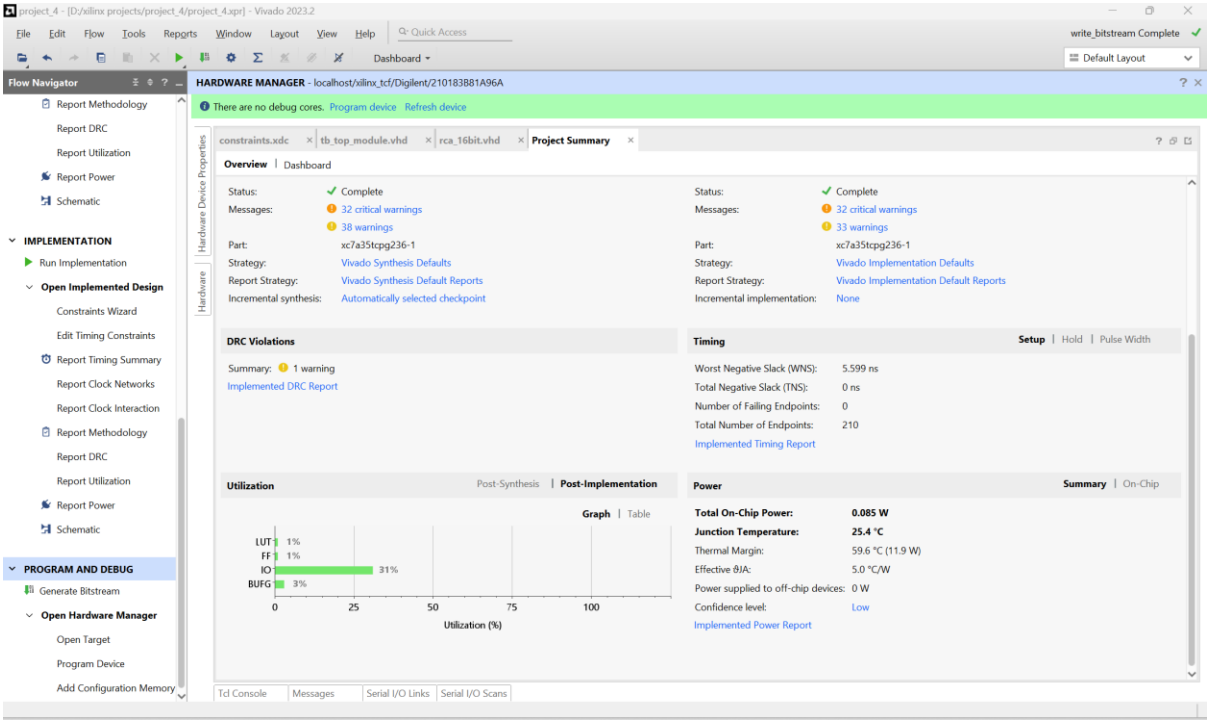
IMPLEMENTATION:



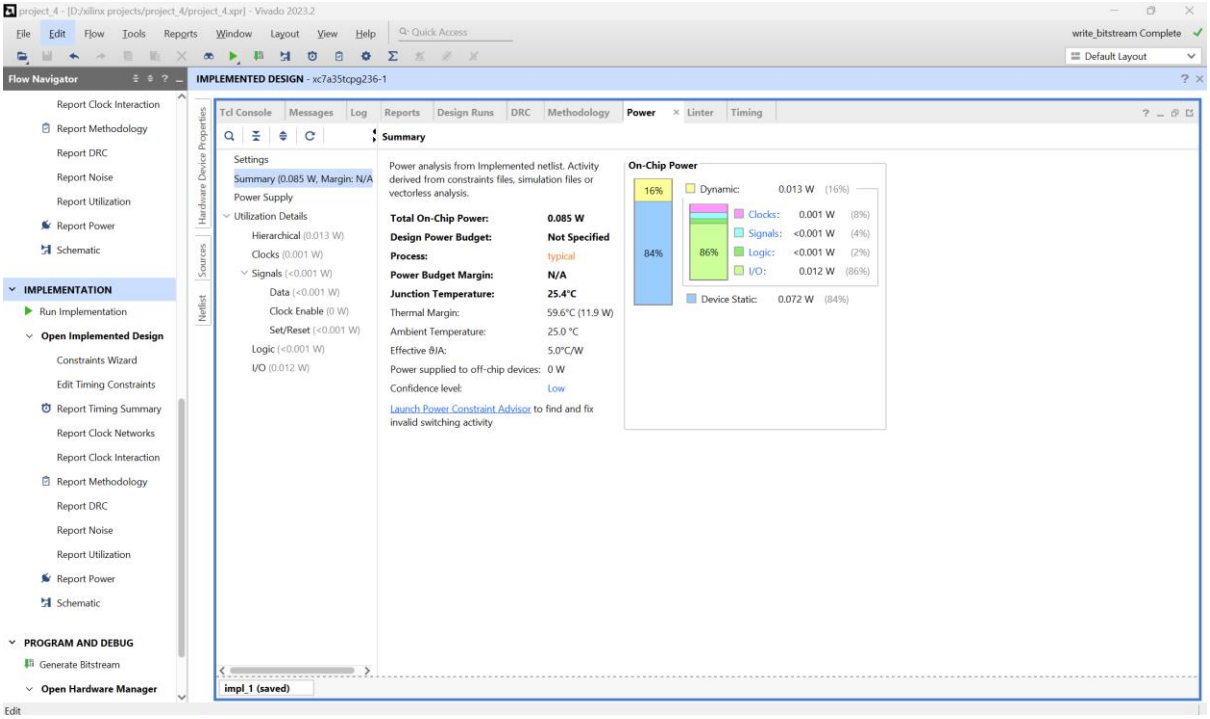
RTL ANALYSIS:



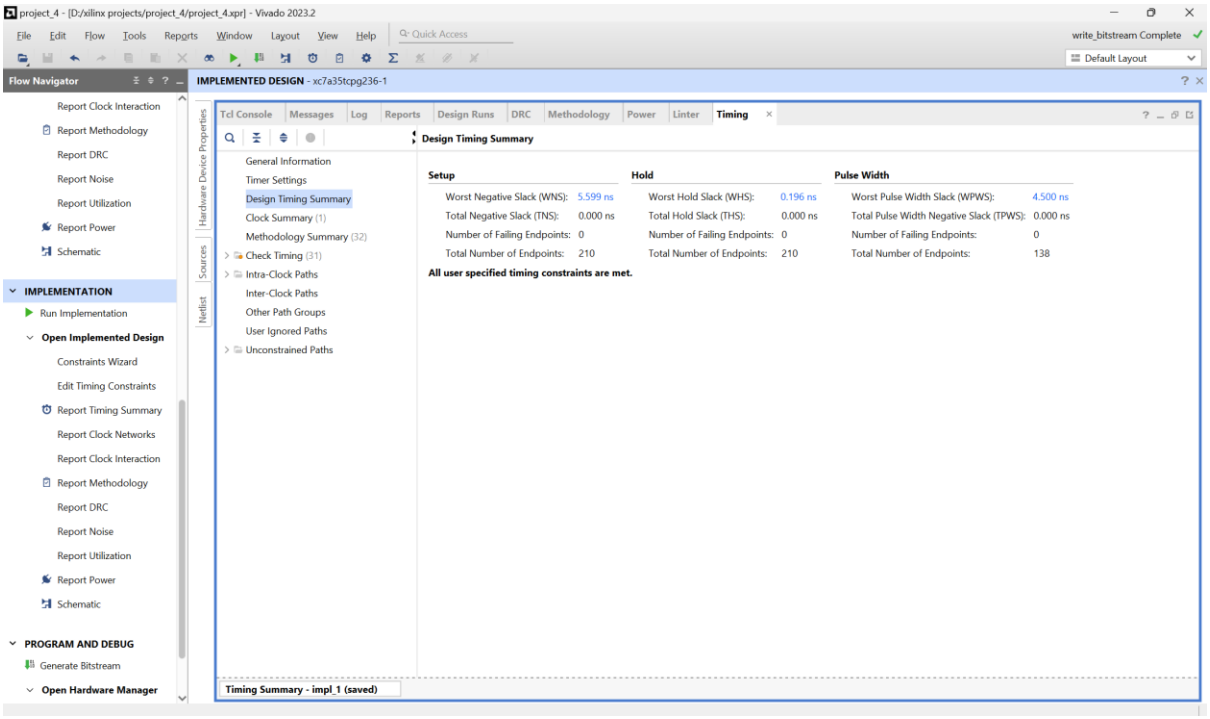
SUMMARY:



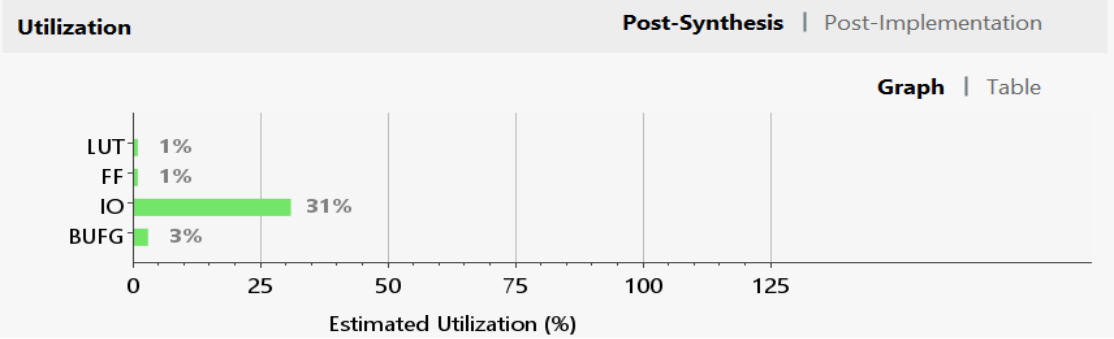
POWER ANALYSIS:



TIMING ANALYSIS:



POST-SYNTHESIS:

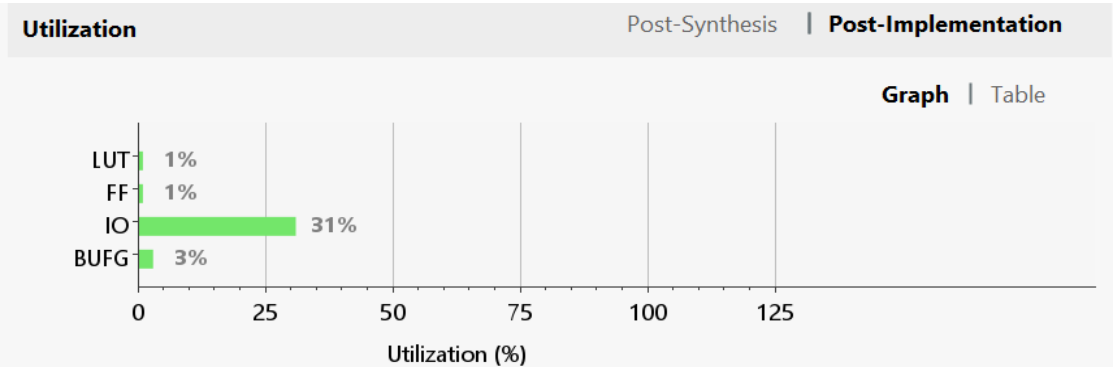


Utilization — **Post-Synthesis** | Post-Implementation

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	55	20800	0.26
FF	137	41600	0.33
IO	33	106	31.13
BUFG	1	32	3.13

POST-IMPLEMENTATION:

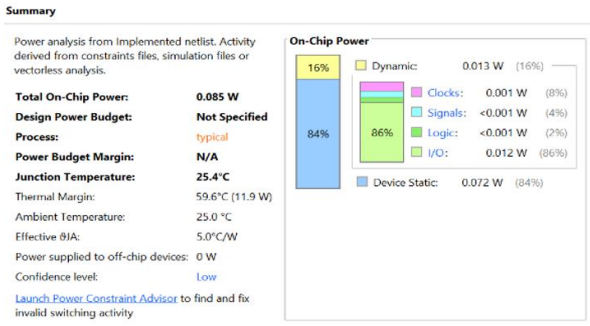
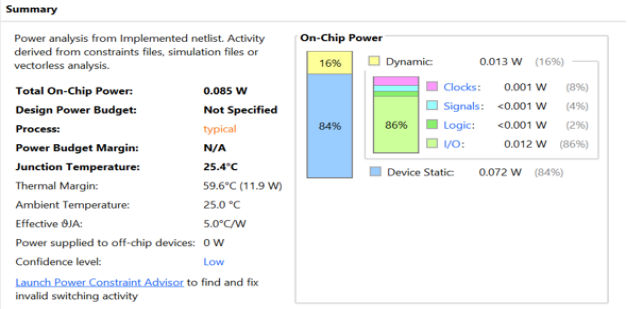
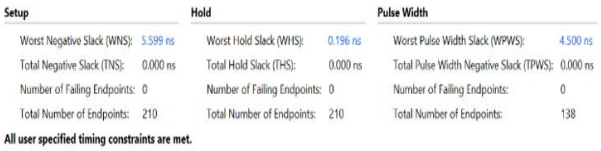
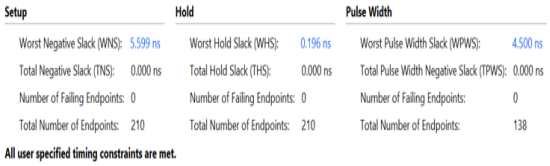
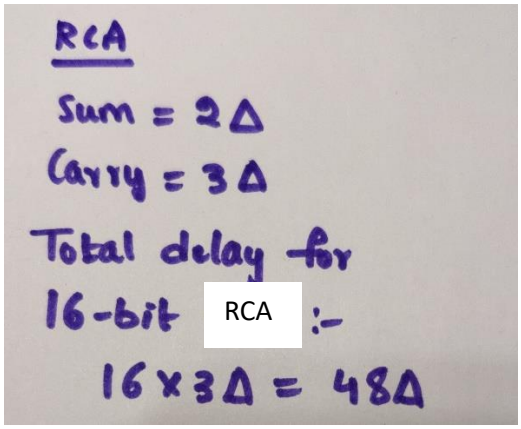
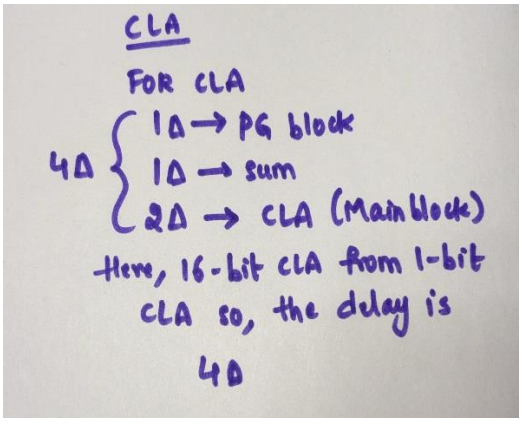


Utilization — Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	55	20800	0.26
FF	137	41600	0.33
IO	33	106	31.13
BUFG	1	32	3.13

COMPARISION:

RIPPLE-CARRY ADDER	CARRY-LOOKAHEAD ADDER
POWER:  Summary Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis. Total On-Chip Power: 0.085 W Design Power Budget: Not Specified Process: typical Power Budget Margin: N/A Junction Temperature: 25.4°C Thermal Margin: 59.6°C (11.9 W) Ambient Temperature: 25.0 °C Effective θJA: 5.0°C/W Power supplied to off-chip devices: 0 W Confidence level: Low Launch Power Constraint Advisor to find and fix invalid switching activity On-Chip Power 16% 84% Dynamic: 0.013 W (16%) 86% Clocks: 0.001 W (8%) Signals: <0.001 W (4%) Logic: <0.001 W (2%) I/O: 0.012 W (86%) Device Static: 0.072 W (84%) The total on-chip power for 16-bit RCA (Ripple Carry Adder) is recorded as 0.085 W after debugging on the Basys3 board.	**POWER:**  Summary Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis. Total On-Chip Power: 0.085 W Design Power Budget: Not Specified Process: typical Power Budget Margin: N/A Junction Temperature: 25.4°C Thermal Margin: 59.6°C (11.9 W) Ambient Temperature: 25.0 °C Effective θJA: 5.0°C/W Power supplied to off-chip devices: 0 W Confidence level: Low Launch Power Constraint Advisor to find and fix invalid switching activity On-Chip Power 16% 84% Dynamic: 0.013 W (16%) 86% Clocks: 0.001 W (8%) Signals: <0.001 W (4%) Logic: <0.001 W (2%) I/O: 0.012 W (86%) Device Static: 0.072 W (84%) The total on-chip power for 16-bit CLA (Carry Lookahead Adder) is recorded as 0.085 W after debugging on the Basys3 board.
TIMING:  Setup Worst Negative Slack (WNS): 5.599 ns Total Negative Slack (TNS): 0.000 ns Number of Failing Endpoints: 0 Total Number of Endpoints: 210 Hold Worst Hold Slack (WHS): 0.196 ns Total Hold Slack (THS): 0.000 ns Number of Failing Endpoints: 0 Total Number of Endpoints: 210 Pulse Width Worst Pulse Width Slack (WPWS): 4.500 ns Total Pulse Width Negative Slack (TPWS): 0.000 ns Number of Failing Endpoints: 0 Total Number of Endpoints: 138 All user specified timing constraints are met. The setup time for the 16-bit RCA is 5.599 ns The hold time for the 16-bit RCA is 0.196 ns The pulse width for the 16-bit RCA is 4.500 ns	**TIMING:**  Setup Worst Negative Slack (WNS): 5.599 ns Total Negative Slack (TNS): 0.000 ns Number of Failing Endpoints: 0 Total Number of Endpoints: 210 Hold Worst Hold Slack (WHS): 0.196 ns Total Hold Slack (THS): 0.000 ns Number of Failing Endpoints: 0 Total Number of Endpoints: 210 Pulse Width Worst Pulse Width Slack (WPWS): 4.500 ns Total Pulse Width Negative Slack (TPWS): 0.000 ns Number of Failing Endpoints: Total Number of Endpoints: 138 All user specified timing constraints are met. The setup time for the 16-bit CLA is 5.599 ns The hold time for the 16-bit CLA is 0.196 ns The pulse width for the 16-bit CLA is 4.500 ns
DELAY:  RCA Sum = 2Δ Carry = 3Δ Total delay for 16-bit RCA :- 16 x 3Δ = 48Δ	**DELAY:**  CLA FOR CLA 4Δ { 1Δ → PG block 1Δ → sum 2Δ → CLA (Main block) Here, 16-bit CLA from 1-bit CLA so, the delay is 4Δ

The delay for the 16-bit RCA is 48Δ.	The delay for the 16-bit CLA is 4Δ.
AREA: RCA consists of n full adders, each using basic logic gates (AND, OR, XOR). The area required grows directly with the number of bits, resulting in a linear increase in area.	**AREA:** CLA is more complex because it requires additional logic for generating carry signals (G and P). The area grows faster than RCA, with a linear increase for the full adder array and a more than linear increase for the carry generation logic, making it more area-intensive for large bit widths.

CONCLUSION:

When comparing the **16-bit Ripple Carry Adder (RCA)** and the **16-bit Carry Lookahead Adder (CLA)**, both have similar timing characteristics, including **setup time**, **hold time**, and **pulse width**, meaning they operate under the same basic clocking conditions. However, the key difference is in their **delay** performance. The **CLA** significantly outperforms the **RCA**, with a much lower delay **4Δ** compared to **48Δ** delay of the **RCA**. This is due to the **CLA's** more efficient carry propagation, which enables faster computation by reducing the carry delay, whereas the **RCA** experiences delays as the carry must propagate sequentially through each bit.

In terms of **power consumption**, both the **CLA** and the **RCA** have the same total on-chip power consumption of **0.085 W**. Despite the CLA's faster operation, its power usage remains equivalent to the RCA. This indicates that the CLA's enhanced speed comes without a significant increase in power demand.

Overall, while the **RCA** is simpler and easier to implement, the **CLA** provides a notable performance advantage in terms of speed, making it more suitable for applications requiring faster arithmetic operations. The **RCA**, though slower, is more straightforward but consumes the same amount of power, which may make it more appealing for low-complexity designs.