WRIGHT STATE UNIVERSITY

EE4620L/EE6620L/CEG4324L/CEG6324L

DIGITAL INTEGRATED CIRCUIT DESIGN LAB

Project

By

Current, Logan

UID: U01019510

Email: Current.22@wright.edu

Submission date: 04/18/2024

"I have neither given nor received aid on this assignment, nor have I observed any violation of the Honor code"

Signature : 

Date : 04/18/2024

Table of Contents

List of Figures

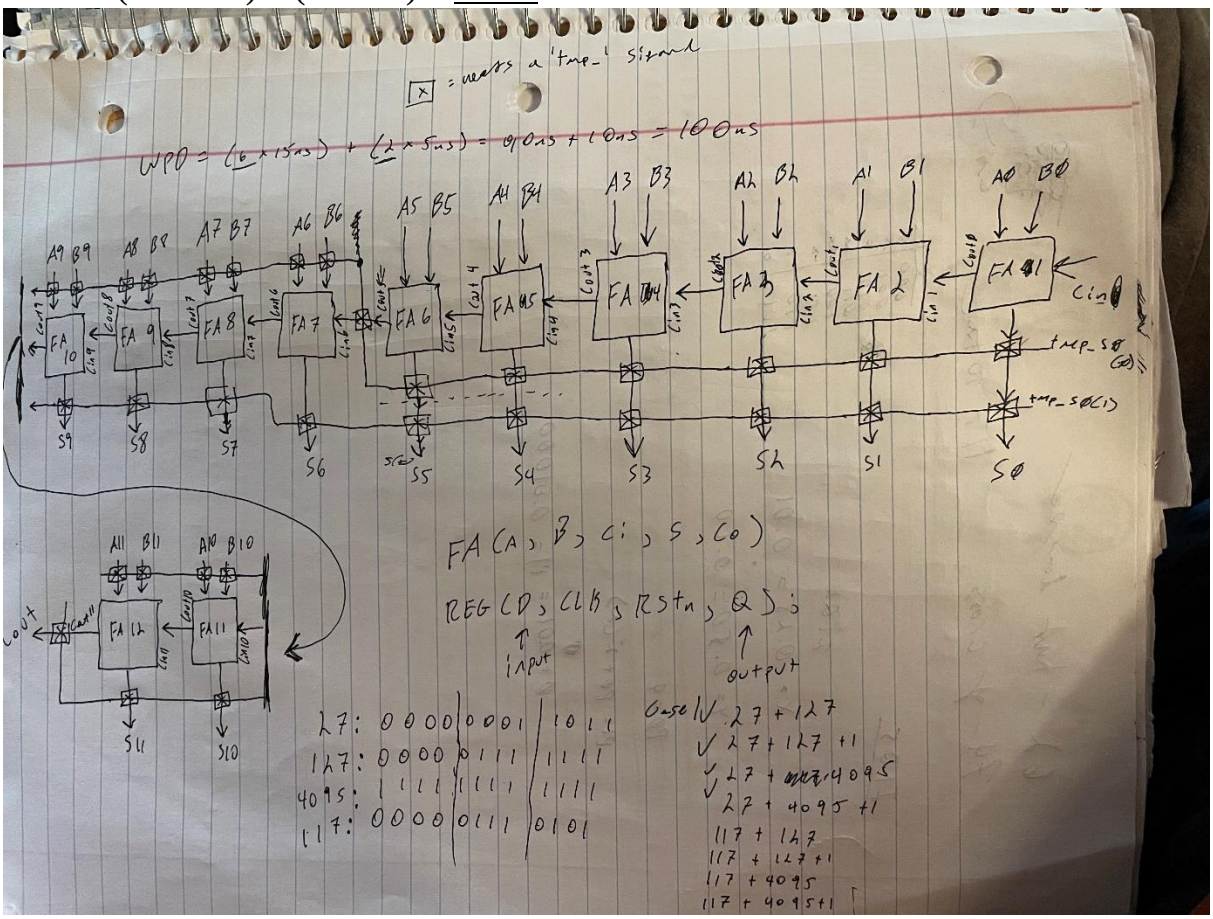
1. AIM / OBJECTIVE:

   The aim of this lab was to explore and design the concept of pipelining as well as implementing it in the Vivado environment.

2. 12 -bit pipeline Arch & WPD Calculation:

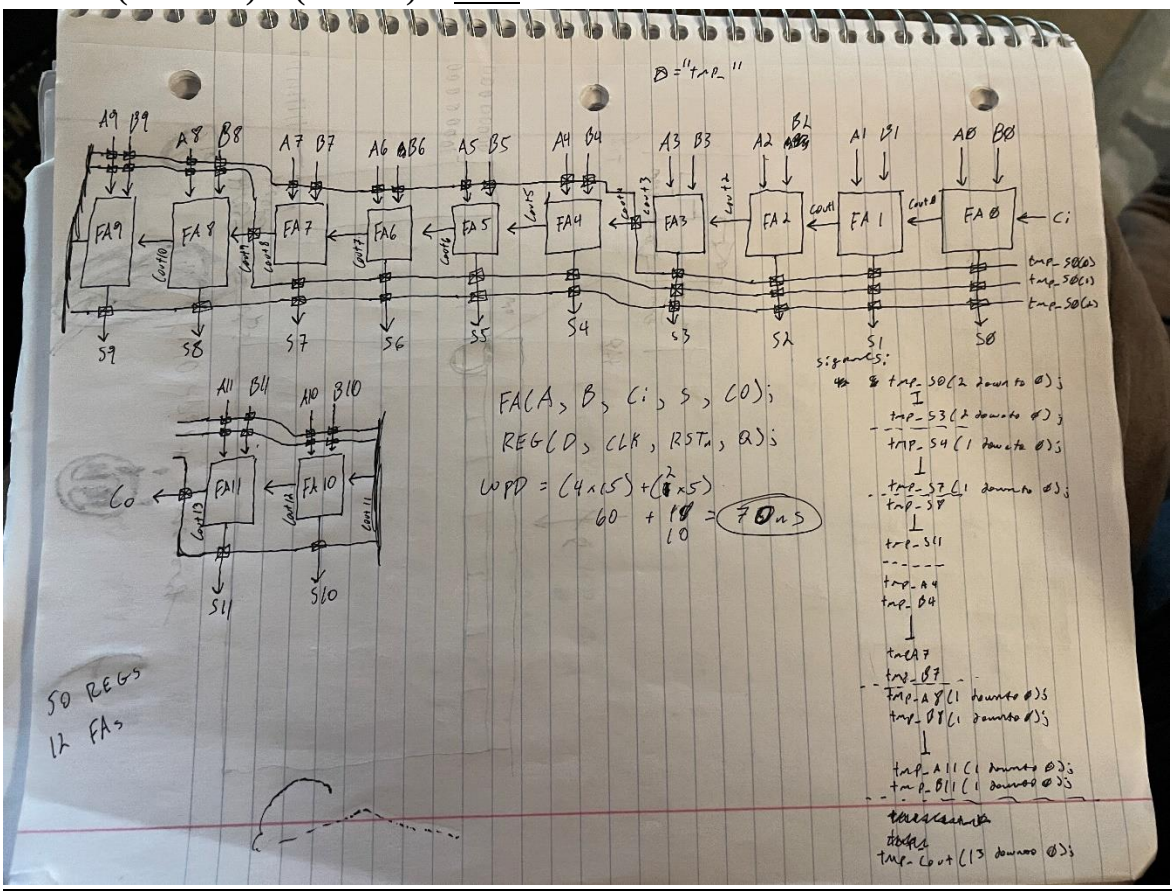
Case 1:

WPD = (6 * 15ns) + (2 * 5ns) = 100ns



Case 2:

WPD = (4 * 15ns) + (2 * 5ns) = 70ns



3. ## VHDL Code & TB Code:

VHDL Code:

Case 1 :

```
------------------------------------------------------------------------
------------
---REGISTER
library ieee;
use ieee.std_logic_1164.all;

entity REG is
    port(D, CLK, RSTn: in std_logic;
         Q: out std_logic);
```

```
end entity;

architecture behaviour of REG is

begin

process(CLK)
    begin
        if (rising_edge(CLK)) then
            if(RSTn = '1') then
                Q <= '0'
                  --pragma_sythesis_off
                  after 10ns
                  --pragma_sythesis_on
                  ;
               else
                Q <= D
                  --pragma_sythesis_off
                  after 10ns
                  --pragma_sythesis_on
                  ;
            end if;
        end if;
    end process;

end architecture;

--------------------------------------------------------------------------
------------
---FULL ADDER
library ieee;
use ieee.std_logic_1164.all;

entity FA is
    port (A, B, Ci: IN STD_LOGIC ;
          S, Co: OUT STD_LOGIC ) ;
end entity;

architecture behaviour of FA is
```

```
    begin
        S  <= A XOR B XOR Ci
        --pragma_sythesis_off
        after 10ns
        --pragma_sythesis_on
        ;
        Co <= (A AND B) OR (Ci AND A) OR (Ci AND B)
        --pragma_sythesis_off
        after 15ns
        --pragma_sythesis_on
        ;
end architecture;

------------------------------------------------------------------------
------------
---PIPELINED ADDER
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity pipelined_adder is
    generic(N :integer := 12); --4 bit pipelined adder is given here
as an example
    Port   (A  : in STD_LOGIC_VECTOR (N-1 downto 0);
            B  : in STD_LOGIC_VECTOR (N-1 downto 0);
            Ci : in STD_LOGIC;
            S  : out STD_LOGIC_VECTOR (N downto 0);
            CLK, RSTn : in STD_LOGIC );
end pipelined_adder;

architecture Behavioral of pipelined_adder is

--- component declaration

component FA
    PORT (A, B, Ci: IN STD_LOGIC ;
          S, Co: OUT STD_LOGIC ) ;
end component;

component REG
```

```
    port(D, CLK, RSTn: in std_logic;
         Q: out std_logic);
end component;

--- signal declaration example is given below.
--- You'll need to modify signals and their vector sizes for your
pipeline designs

signal Co       : std_logic;    --use this signal as your last carry
out signal

signal tmp_cout : std_logic_vector(12 downto 0);

signal tmp_S0   : std_logic_vector(1 downto 0);
signal tmp_S1   : std_logic_vector(1 downto 0);
signal tmp_S2   : std_logic_vector(1 downto 0);
signal tmp_S3   : std_logic_vector(1 downto 0);
signal tmp_S4   : std_logic_vector(1 downto 0);
signal tmp_S5   : std_logic_vector(1 downto 0);
signal tmp_S6   : std_logic;
signal tmp_S7   : std_logic;
signal tmp_S8   : std_logic;
signal tmp_S9   : std_logic;
signal tmp_S10   : std_logic;
signal tmp_S11   : std_logic;


signal tmp_A6 : std_logic;
signal tmp_B6 : std_logic;
signal tmp_A7 : std_logic;
signal tmp_B7 : std_logic;
signal tmp_A8 : std_logic;
signal tmp_B8 : std_logic;
signal tmp_A9 : std_logic;
signal tmp_B9 : std_logic;
signal tmp_A10 : std_logic;
signal tmp_B10 : std_logic;
signal tmp_A11 : std_logic;
signal tmp_B11 : std_logic;
```

```
--- add necessary signals here

begin

--- 1st Full Adder

FA_0    : FA port map(A(0), B(0), Ci, tmp_S0(0), tmp_cout(0));

reg_s0_0: REG port map(tmp_S0(0), CLK, RSTn, tmp_S0(1));
reg_s0_1: REG port map(tmp_S0(1), CLK, RSTn, S(0));

--- 2nd Full Adder

FA_1    : FA port map(A(1), B(1), tmp_cout(0), tmp_S1(0),
tmp_cout(1));

reg_s1_0: REG port map(tmp_S1(0), CLK, RSTn, tmp_S1(1));
reg_s1_1: REG port map(tmp_S1(1), CLK, RSTn, S(1));

---3rd Full Adder

FA_2: FA port map(A(2), B(2), tmp_cout(1), tmp_S2(0), tmp_cout(2));

reg_s2_0: REG port map(tmp_S2(0), CLK, RSTn, tmp_S2(1));
reg_s2_1: REG port map(tmp_S2(1), CLK, RSTn, S(2));

--- 4th Full Adder

FA_3: FA port map(A(3), B(3), tmp_cout(2), tmp_S3(0), tmp_cout(3));

reg_s3_0: REG port map(tmp_S3(0), CLK, RSTn, tmp_S3(1));
reg_s3_1: REG port map(tmp_S3(1), CLK, RSTn, S(3));

--- 5th Full Adder

FA_4: FA port map(A(4), B(4), tmp_cout(3), tmp_S4(0), tmp_cout(4));
```

```
reg_s4_0: REG port map(tmp_S4(0), CLK, RSTn, tmp_S4(1));
reg_s4_1: REG port map(tmp_S4(1), CLK, RSTn, S(4));

--- 6th Full Adder

FA_5: FA port map(A(5), B(5), tmp_cout(4), tmp_S5(0), tmp_cout(5));

reg_s5_0: REG port map(tmp_S5(0), CLK, RSTn, tmp_S5(1));
reg_s5_1: REG port map(tmp_S5(1), CLK, RSTn, S(5));

cout_reg_0: REG port map(tmp_cout(5), CLK, RSTn, tmp_cout(6));

--- 7th Full Adder

reg_tmp_A6: REG port map(A(6), CLK, RSTn, tmp_A6);
reg_tmp_B6: REG port map(B(6), CLK, RSTn, tmp_B6);

FA_6: FA port map(tmp_A6, tmp_B6, tmp_cout(6), tmp_S6, tmp_cout(7));

reg_s6: REG port map(tmp_S6, CLK, RSTn, S(6));

--- 8th Full Adder

reg_tmp_A7: REG port map(A(7), CLK, RSTn, tmp_A7);
reg_tmp_B7: REG port map(B(7), CLK, RSTn, tmp_B7);

FA_7: FA port map(tmp_A7, tmp_B7, tmp_cout(7), tmp_S7, tmp_cout(8));

reg_s7: REG port map(tmp_S7, CLK, RSTn, S(7));

--- 9th Full Adder

reg_tmp_A8: REG port map(A(8), CLK, RSTn, tmp_A8);
reg_tmp_B8: REG port map(B(8), CLK, RSTn, tmp_B8);

FA_8: FA port map(tmp_A8, tmp_B8, tmp_cout(8), tmp_S8, tmp_cout(9));

reg_s8: REG port map(tmp_S8, CLK, RSTn, S(8));
```

```
--- 10th Full Adder

reg_tmp_A9: REG port map(A(9), CLK, RSTn, tmp_A9);
reg_tmp_B9: REG port map(B(9), CLK, RSTn, tmp_B9);

FA_9: FA port map(tmp_A9, tmp_B9, tmp_cout(9), tmp_S9, tmp_cout(10));

reg_s9: REG port map(tmp_S9, CLK, RSTn, S(9));

--- 11th Full Adder

reg_tmp_A10: REG port map(A(10), CLK, RSTn, tmp_A10);
reg_tmp_B10: REG port map(B(10), CLK, RSTn, tmp_B10);

FA_10: FA port map(tmp_A10, tmp_B10, tmp_cout(10), tmp_S10,
tmp_cout(11));

reg_s10: REG port map(tmp_S10, CLK, RSTn, S(10));

--- 12th Full Adder

reg_tmp_A11: REG port map(A(11), CLK, RSTn, tmp_A11);
reg_tmp_B11: REG port map(B(11), CLK, RSTn, tmp_B11);

FA_11: FA port map(tmp_A11, tmp_B11, tmp_cout(11), tmp_S11,
tmp_cout(12));

reg_s11: REG port map(tmp_S11, CLK, RSTn, S(11));

cout_reg_1: REG port map(tmp_cout(12), CLK, RSTn, Co);

S(N)<=Co;

end Behavioral;
```

Case 2 :

```
------------------------------------------------------------------------
------------
---REGISTER
library ieee;
use ieee.std_logic_1164.all;

entity REG is
    port(D, CLK, RSTn: in std_logic;
         Q: out std_logic);
end entity;

architecture behaviour of REG is

begin

process(CLK)
    begin
        if (rising_edge(CLK)) then
            if(RSTn = '1') then
                Q <= '0'
                  --pragma_sythesis_off
                  after 10ns
                  --pragma_sythesis_on
                  ;
               else
                Q <= D
                  --pragma_sythesis_off
                  after 10ns
                  --pragma_sythesis_on
                  ;
            end if;
        end if;
    end process;

end architecture;

------------------------------------------------------------------------
------------
---FULL ADDER
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity FA is
    port (A, B, Ci: IN STD_LOGIC ;
          S, Co: OUT STD_LOGIC ) ;
end entity;

architecture behaviour of FA is
    begin
        S  <= A XOR B XOR Ci
        --pragma_sythesis_off
        after 10ns
        --pragma_sythesis_on
        ;
        Co <= (A AND B) OR (Ci AND A) OR (Ci AND B)
        --pragma_sythesis_off
        after 15ns
        --pragma_sythesis_on
        ;
end architecture;

------------------------------------------------------------------------
------------
---PIPELINED ADDER
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity pipelined_adder is
    generic(N :integer := 12); --4 bit pipelined adder is given here
as an example
    Port   (A  : in STD_LOGIC_VECTOR (N-1 downto 0);
            B  : in STD_LOGIC_VECTOR (N-1 downto 0);
            Ci : in STD_LOGIC;
            S  : out STD_LOGIC_VECTOR (N downto 0);
            CLK, RSTn : in STD_LOGIC );
end pipelined_adder;

architecture Behavioral of pipelined_adder is
```

```
--- component declaration

component FA
    PORT (A, B, Ci: IN STD_LOGIC ;
          S, Co: OUT STD_LOGIC ) ;
end component;

component REG
    port(D, CLK, RSTn: in std_logic;
         Q: out std_logic);
end component;

--- signal declaration example is given below.
--- You'll need to modify signals and their vector sizes for your
pipeline designs

signal Co       : std_logic;    --use this signal as your last carry
out signal

signal tmp_cout : std_logic_vector(13 downto 0);

signal tmp_S0   : std_logic_vector(2 downto 0);
signal tmp_S1   : std_logic_vector(2 downto 0);
signal tmp_S2   : std_logic_vector(2 downto 0);
signal tmp_S3   : std_logic_vector(2 downto 0);
signal tmp_S4   : std_logic_vector(1 downto 0);
signal tmp_S5   : std_logic_vector(1 downto 0);
signal tmp_S6   : std_logic_vector(1 downto 0);
signal tmp_S7   : std_logic_vector(1 downto 0);
signal tmp_S8   : std_logic;
signal tmp_S9   : std_logic;
signal tmp_S10   : std_logic;
signal tmp_S11   : std_logic;

signal tmp_A4 : std_logic;
signal tmp_B4 : std_logic;
signal tmp_A5 : std_logic;
signal tmp_B5 : std_logic;
```

```
signal tmp_A6 : std_logic;
signal tmp_B6 : std_logic;
signal tmp_A7 : std_logic;
signal tmp_B7 : std_logic;
signal tmp_A8 : std_logic_vector(1 downto 0);
signal tmp_B8 : std_logic_vector(1 downto 0);
signal tmp_A9 : std_logic_vector(1 downto 0);
signal tmp_B9 : std_logic_vector(1 downto 0);
signal tmp_A10 : std_logic_vector(1 downto 0);
signal tmp_B10 : std_logic_vector(1 downto 0);
signal tmp_A11 : std_logic_vector(1 downto 0);
signal tmp_B11 : std_logic_vector(1 downto 0);


--- add necessary signals here

begin

--- 1st Full Adder

FA_0    : FA port map(A(0), B(0), Ci, tmp_S0(0), tmp_cout(0));

reg_s0_0: REG port map(tmp_S0(0), CLK, RSTn, tmp_S0(1));
reg_s0_1: REG port map(tmp_S0(1), CLK, RSTn, tmp_S0(2));
reg_s0_2: REG port map(tmp_S0(2), CLK, RSTn, S(0));

--- 2nd Full Adder

FA_1    : FA port map(A(1), B(1), tmp_cout(0), tmp_S1(0),
tmp_cout(1));

reg_s1_0: REG port map(tmp_S1(0), CLK, RSTn, tmp_S1(1));
reg_s1_1: REG port map(tmp_S1(1), CLK, RSTn, tmp_S1(2));
reg_s1_2: REG port map(tmp_S1(2), CLK, RSTn, S(1));

---3rd Full Adder

FA_2: FA port map(A(2), B(2), tmp_cout(1), tmp_S2(0), tmp_cout(2));
```

```
reg_s2_0: REG port map(tmp_S2(0), CLK, RSTn, tmp_S2(1));
reg_s2_1: REG port map(tmp_S2(1), CLK, RSTn, tmp_S2(2));
reg_s2_2: REG port map(tmp_S2(2), CLK, RSTn, S(2));

--- 4th Full Adder

FA_3: FA port map(A(3), B(3), tmp_cout(2), tmp_S3(0), tmp_cout(3));

reg_s3_0: REG port map(tmp_S3(0), CLK, RSTn, tmp_S3(1));
reg_s3_1: REG port map(tmp_S3(1), CLK, RSTn, tmp_S3(2));
reg_s3_2: REG port map(tmp_S3(2), CLK, RSTn, S(3));

cout_reg_0: REG port map(tmp_cout(3), CLK, RSTn, tmp_cout(4));

--- 5th Full Adder

reg_tmp_A4: REG port map(A(4), CLK, RSTn, tmp_A4);
reg_tmp_B4: REG port map(B(4), CLK, RSTn, tmp_B4);

FA_4: FA port map(tmp_A4, tmp_B4, tmp_cout(4), tmp_S4(0),
tmp_cout(5));

reg_s4_0: REG port map(tmp_S4(0), CLK, RSTn, tmp_S4(1));
reg_s4_1: REG port map(tmp_S4(1), CLK, RSTn, S(4));

--- 6th Full Adder

reg_tmp_A5: REG port map(A(5), CLK, RSTn, tmp_A5);
reg_tmp_B5: REG port map(B(5), CLK, RSTn, tmp_B5);

FA_5: FA port map(tmp_A5, tmp_B5, tmp_cout(5), tmp_S5(0),
tmp_cout(6));

reg_s5_0: REG port map(tmp_S5(0), CLK, RSTn, tmp_S5(1));
reg_s5_1: REG port map(tmp_S5(1), CLK, RSTn, S(5));

--- 7th Full Adder

reg_tmp_A6: REG port map(A(6), CLK, RSTn, tmp_A6);
```

```
reg_tmp_B6: REG port map(B(6), CLK, RSTn, tmp_B6);

FA_6: FA port map(tmp_A6, tmp_B6, tmp_cout(6), tmp_S6(0),
tmp_cout(7));

reg_s6_0: REG port map(tmp_S6(0), CLK, RSTn, tmp_S6(1));
reg_s6_1: REG port map(tmp_S6(1), CLK, RSTn, S(6));

--- 8th Full Adder

reg_tmp_A7: REG port map(A(7), CLK, RSTn, tmp_A7);
reg_tmp_B7: REG port map(B(7), CLK, RSTn, tmp_B7);

FA_7: FA port map(tmp_A7, tmp_B7, tmp_cout(7), tmp_S7(0),
tmp_cout(8));

reg_s7_0: REG port map(tmp_S7(0), CLK, RSTn, tmp_S7(1));
reg_s7_1: REG port map(tmp_S7(1), CLK, RSTn, S(7));

cout_reg_1: REG port map(tmp_cout(8), CLK, RSTn, tmp_cout(9));

--- 9th Full Adder

reg_tmp_A8_0: REG port map(A(8), CLK, RSTn, tmp_A8(0));
reg_tmp_B8_0: REG port map(B(8), CLK, RSTn, tmp_B8(0));
reg_tmp_A8_1: REG port map(tmp_A8(0), CLK, RSTn, tmp_A8(1));
reg_tmp_B8_1: REG port map(tmp_B8(0), CLK, RSTn, tmp_B8(1));

FA_8: FA port map(tmp_A8(1), tmp_B8(1), tmp_cout(9), tmp_S8,
tmp_cout(10));

reg_s8: REG port map(tmp_S8, CLK, RSTn, S(8));

--- 10th Full Adder

reg_tmp_A9_0: REG port map(A(9), CLK, RSTn, tmp_A9(0));
reg_tmp_B9_0: REG port map(B(9), CLK, RSTn, tmp_B9(0));
reg_tmp_A9_1: REG port map(tmp_A9(0), CLK, RSTn, tmp_A9(1));
reg_tmp_B9_1: REG port map(tmp_B9(0), CLK, RSTn, tmp_B9(1));
```

```
FA_9: FA port map(tmp_A9(1), tmp_B9(1), tmp_cout(10), tmp_S9,
tmp_cout(11));

reg_s9: REG port map(tmp_S9, CLK, RSTn, S(9));
--- 11th Full Adder

reg_tmp_A10_0: REG port map(A(10), CLK, RSTn, tmp_A10(0));
reg_tmp_B10_0: REG port map(B(10), CLK, RSTn, tmp_B10(0));
reg_tmp_A10_1: REG port map(tmp_A10(0), CLK, RSTn, tmp_A10(1));
reg_tmp_B10_1: REG port map(tmp_B10(0), CLK, RSTn, tmp_B10(1));

FA_10: FA port map(tmp_A10(1), tmp_B10(1), tmp_cout(11), tmp_S10,
tmp_cout(12));

reg_s10: REG port map(tmp_S10, CLK, RSTn, S(10));

--- 12th Full Adder

reg_tmp_A11_0: REG port map(A(11), CLK, RSTn, tmp_A11(0));
reg_tmp_B11_0: REG port map(B(11), CLK, RSTn, tmp_B11(0));
reg_tmp_A11_1: REG port map(tmp_A11(0), CLK, RSTn, tmp_A11(1));
reg_tmp_B11_1: REG port map(tmp_B11(0), CLK, RSTn, tmp_B11(1));

FA_11: FA port map(tmp_A11(1), tmp_B11(1), tmp_cout(12), tmp_S11,
tmp_cout(13));

reg_s11: REG port map(tmp_S11, CLK, RSTn, S(11));

cout_reg_2: REG port map(tmp_cout(13), CLK, RSTn, Co);

S(N)<=Co;

end Behavioral;
```

TB Code:

```
----------------------------------------------------------------------------
------------
-- Company:
-- Engineer:
--
-- Create Date:
-- Design Name:
-- Module Name: tb_pipelined_adder - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------
------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_pipelined_adder is
    generic(N :integer := 12); --4 bit pipelined adder is given here
as an example
--  Port ( );
end tb_pipelined_adder;

architecture Behavioral of tb_pipelined_adder is
component pipelined_adder is
    generic(N :integer := 12); --4 bit pipelined adder (change
accordingly)
    Port ( A  : in STD_LOGIC_VECTOR (N-1 downto 0);
```

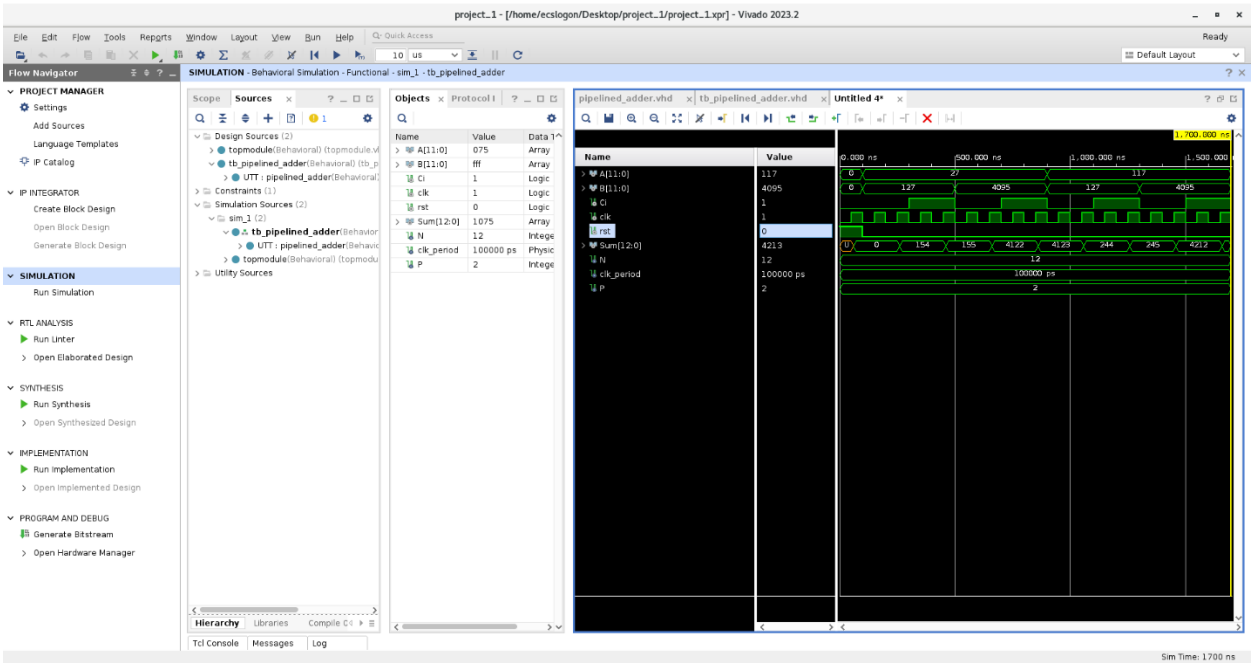
```
           B   : in STD_LOGIC_VECTOR (N-1 downto 0);
           Ci  : in STD_LOGIC;
           S   : out STD_LOGIC_VECTOR (N downto 0);
--           Co : out STD_LOGIC;
           CLK, RSTn : in STD_LOGIC );
end component;

   signal A,B : std_logic_vector (N-1 downto 0);
   signal Ci  : std_logic := '0';
   signal clk : std_logic := '0';
   signal rst : std_logic := '1';
   signal Sum : std_logic_vector (N downto 0);
--   signal Co  : std_logic;

constant clk_period : time := 70 ns;     -- Clock Frequency

constant  P: integer:= 3;                -- number of pipeline stages

begin

UTT: pipelined_adder generic map(N=>12) port map(A,B,Ci,Sum,clk,rst);
--4 bit pipelined adder (change accordingly)

clk_process :process
                begin
                    clk <= '0';
                    wait for clk_period/2;
                    clk <= '1';
                wait for clk_period/2;
             end process;

stim_proc: process
             begin

-------------12-bit ADDER INPUTS are given here as an example
                  rst <= '1';
                  A <= "000000000000";
                  B <= "000000000000";
                  Ci<= '0';
```

```
        wait for clk_period;

        rst <= '0';          --- 27+127
        A <= "000000011011";
        B <= "000001111111";
        Ci<= '0';
        wait for (P)*clk_period;

        A <= "000000011011"; -- 27+127+1
        B <= "000001111111";
        Ci<= '1';
        wait for (P)*clk_period;

        A <= "000000011011"; -- 27+4095
        B <= "111111111111";
        Ci<= '0';
        wait for (P)*clk_period;

        A <= "000000011011"; -- 27+4095+1
        B <= "111111111111";
        Ci<= '1';
        wait for (P)*clk_period;

        A <= "000001110101"; --117+127
        B <= "000001111111";
        Ci<= '0';
        wait for (P)*clk_period;

        A <= "000001110101"; --117+127+1
        B <= "000001111111";
        Ci<= '1';
        wait for (P)*clk_period;

        A <= "000001110101"; --117+4095
        B <= "111111111111";
        Ci<= '0';
        wait for (P)*clk_period;

        A <= "000001110101"; --117+4095+1
```

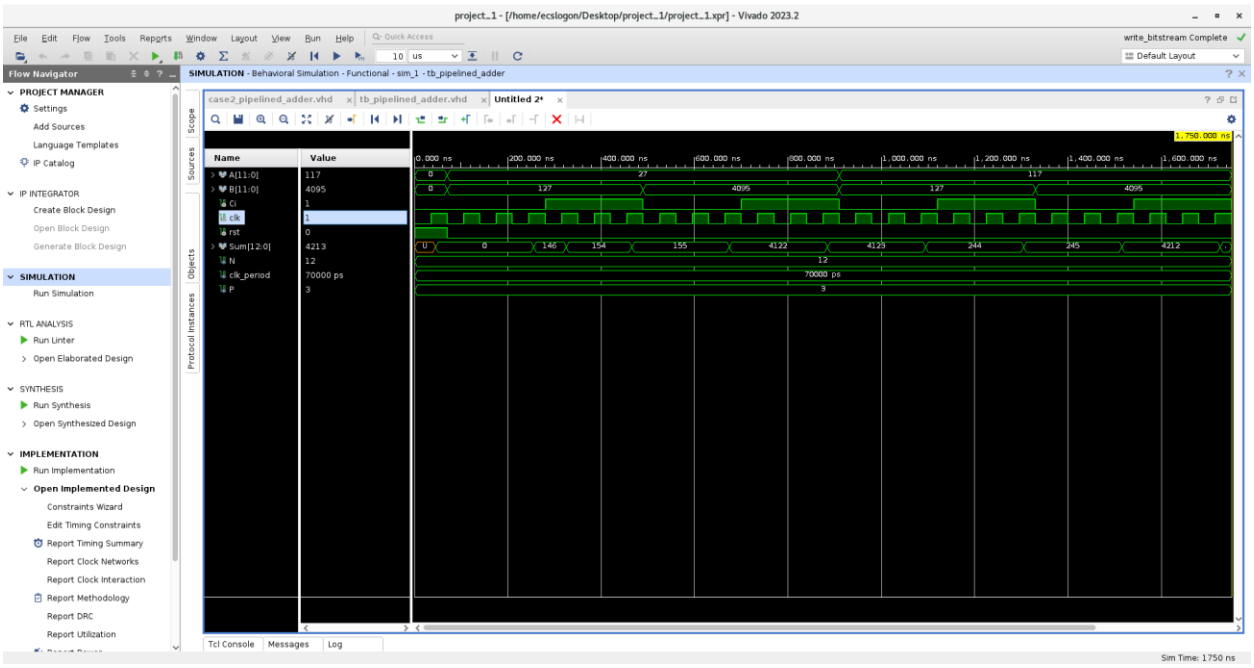
```
                    B <= "111111111111";
                    Ci<= '1';
                    wait for (P)*clk_period;

                std.env.finish;
----------------------------4-bit ADDER INPUT-----------------------------
-----
            end process;



end Behavioral;
```

4. Simulation Waveforms:

Case 1 :

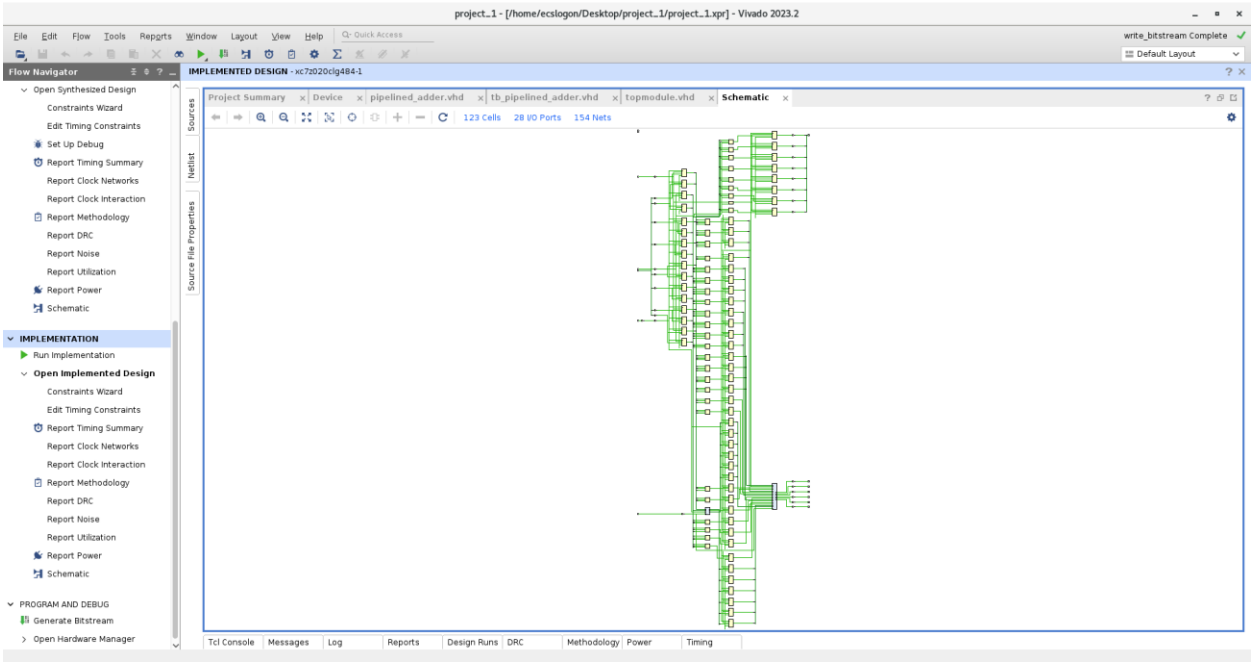


Case 2 :

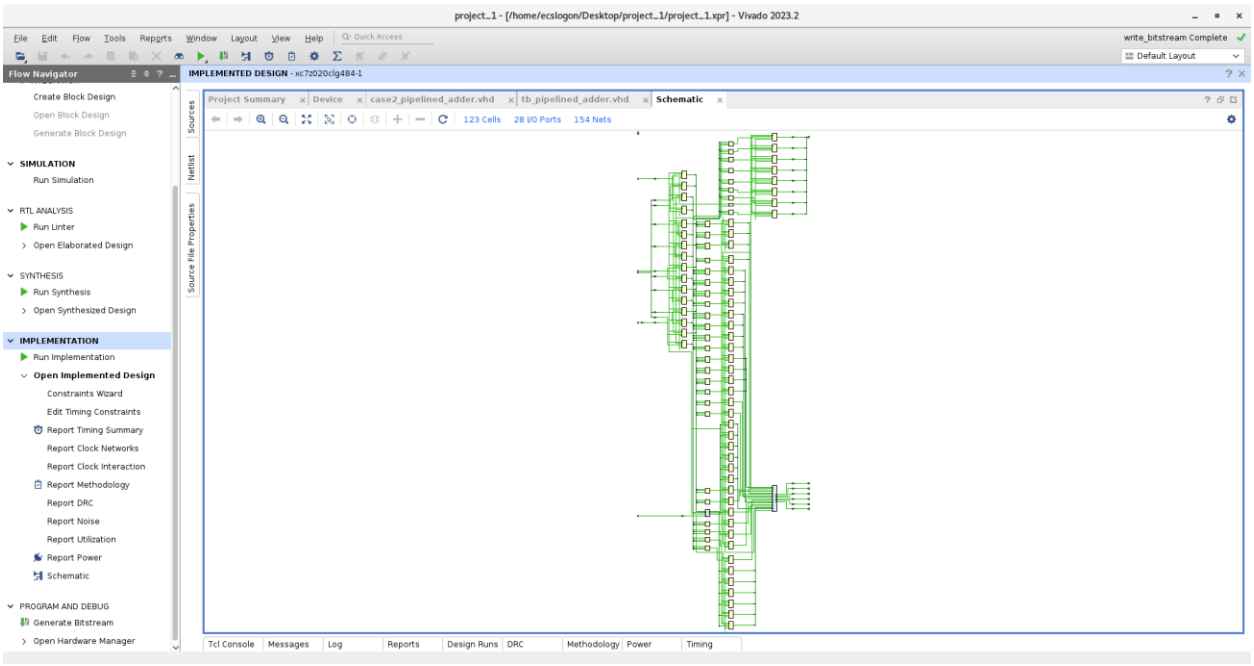


5. Post-synthesis Schematic Diagram:

Case 1:

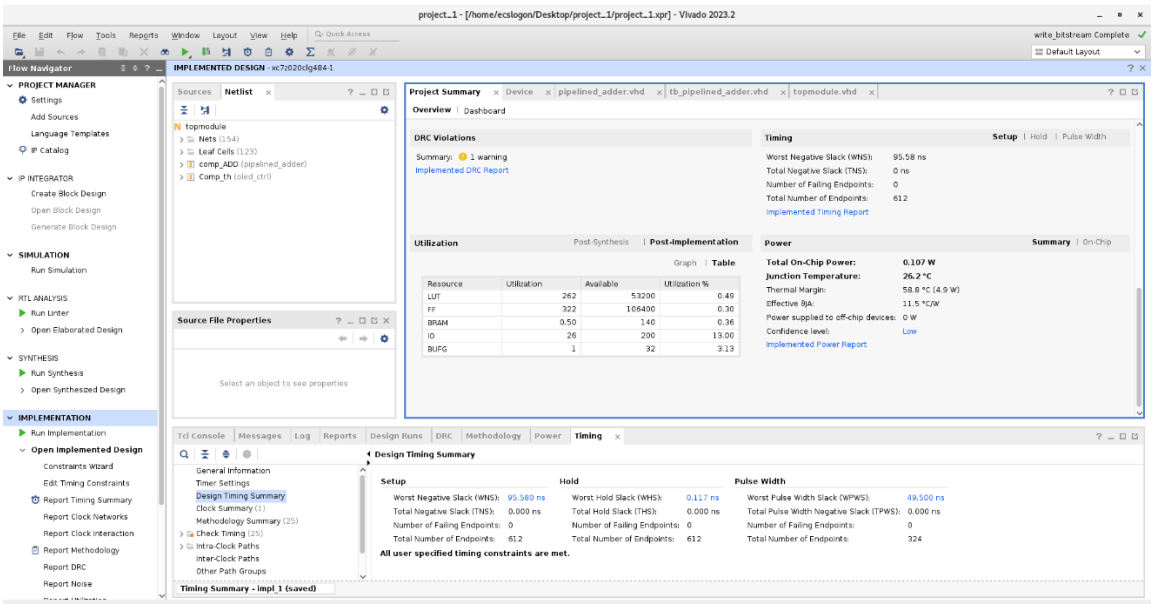


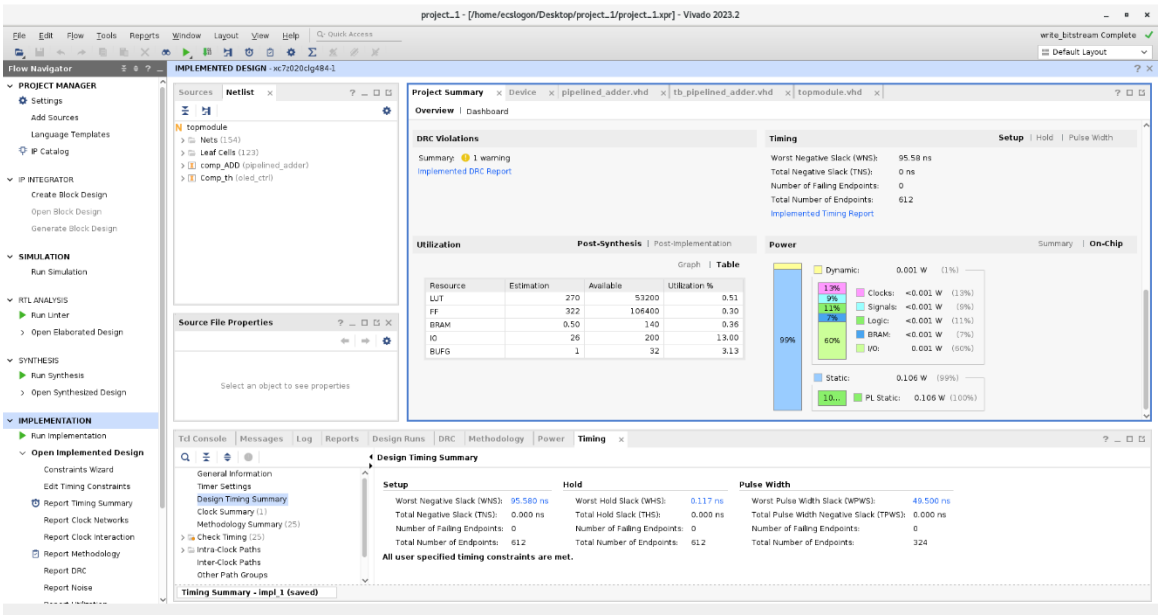
Case 2:


project_1 - [/home/ecslogon/Desktop/project_1/project_1.xpr] - Vivado 2023.2
File Edit Flow Tools Reports Window Layout View Help
write_bitstream Complete
Default Layout
Flow Navigator
IMPLEMENTED DESIGN - xc7z020clg484-1
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
Constraints Wizard
Edit Timing Constraints
Report Timing Summary
Report Clock Networks
Report Clock Interaction
Report Methodology
Report DRC
Report Noise
Report Utilization
Report Power
Schematic
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
Sources
Netlist
Source File Properties
Project Summary
Device
case2_pipelined_adder.vhd
tb_pipelined_adder.vhd
Schematic
123 Cells
28 I/O Ports
154 Nets
Tcl Console
Messages
Log
Reports
Design Runs
DRC
Methodology
Power
Timing

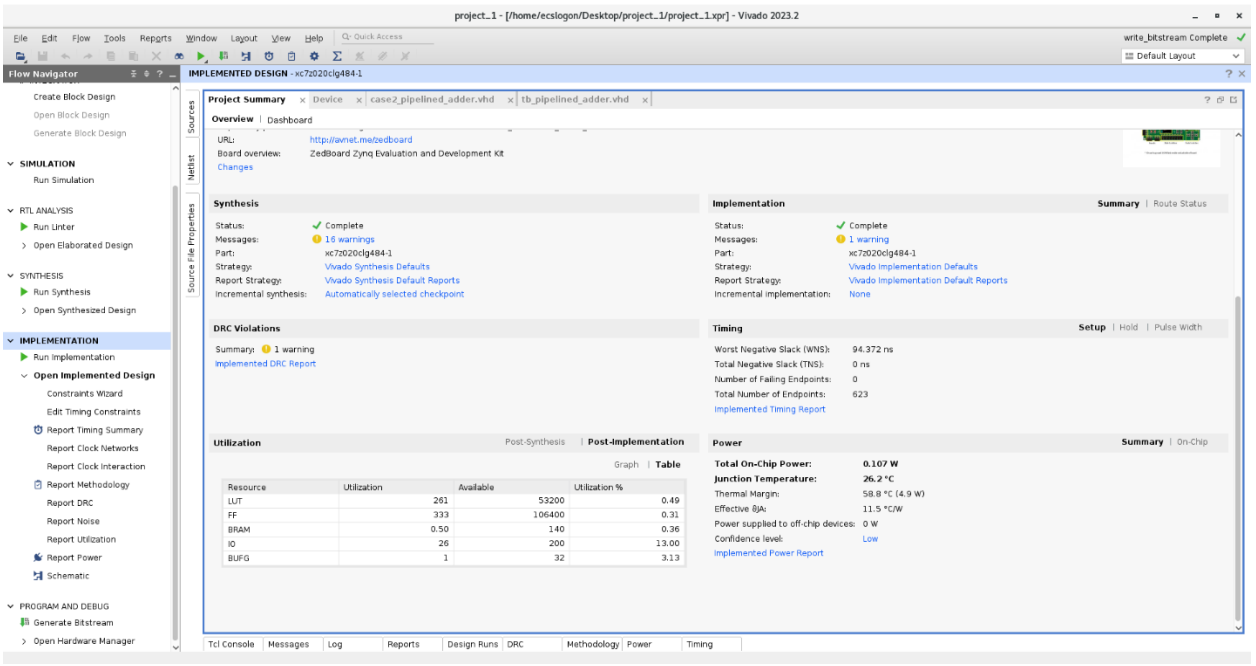
6. Hardware Report:

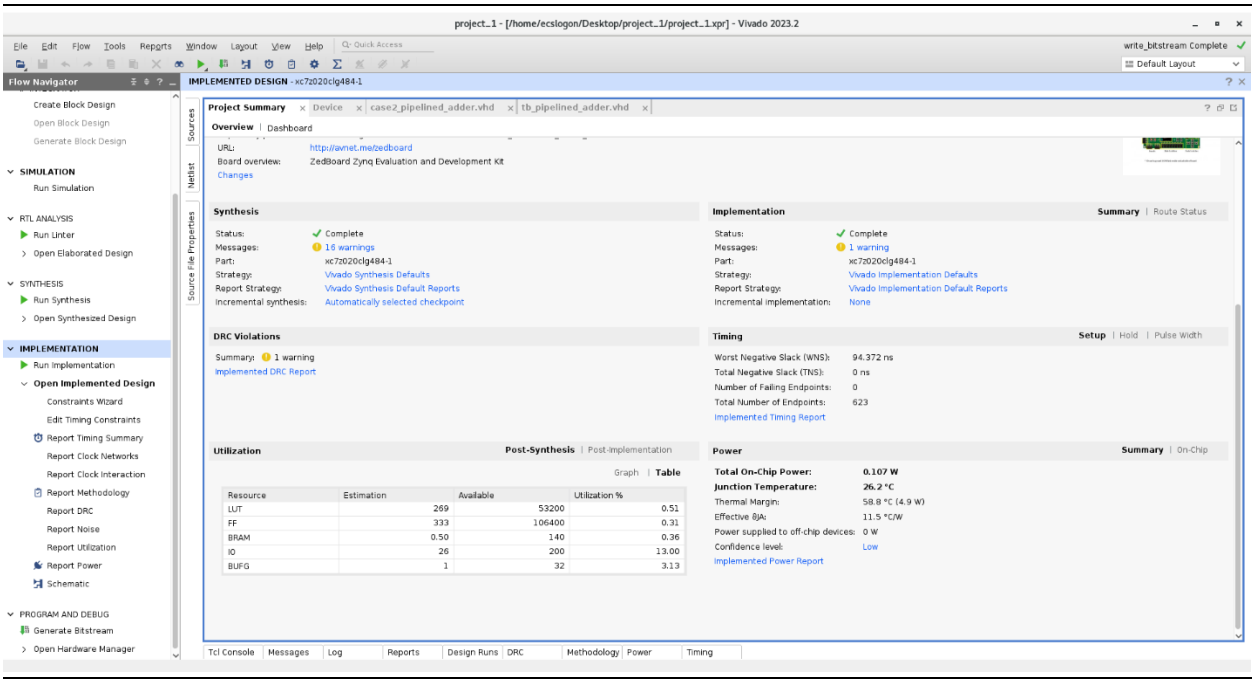
Case 1:





Case 2:


project_1 - [/home/ecslogon/Desktop/project_1/project_1.xpr] - Vivado 2023.2
File Edit Flow Tools Reports Window Layout View Help
write_bitstream Complete
Default Layout
Flow Navigator
IMPLEMENTED DESIGN - xc7z020clg484-1
Project Summary
Device
case2_pipelined_adder.vhd
tb_pipelined_adder.vhd
Overview
Dashboard
URL: http://avnet.me/zedboard
Board overview: ZedBoard Zynq Evaluation and Development Kit
Changes
Synthesis
Status: Complete
Messages: 16 warnings
Part: xc7z020clg484-1
Strategy: Vivado Synthesis Defaults
Report Strategy: Vivado Synthesis Default Reports
Incremental synthesis: Automatically selected checkpoint
Implementation
Summary | Route Status
Status: Complete
Messages: 1 warning
Part: xc7z020clg484-1
Strategy: Vivado Implementation Defaults
Report Strategy: Vivado Implementation Default Reports
Incremental implementation: None
DRC Violations
Summary: 1 warning
Implemented DRC Report
Timing
Setup | Hold | Pulse Width
Worst Negative Slack (WNS): 94.372 ns
Total Negative Slack (TNS): 0 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 623
Implemented Timing Report
Utilization
Post-Synthesis | Post-Implementation
Graph | Table
Resource Utilization Available Utilization %
LUT 261 53200 0.49
FF 333 106400 0.31
BRAM 0.50 140 0.36
IO 26 200 13.00
BUFG 1 32 3.13
Power
Summary | On-Chip
Total On-Chip Power: 0.107 W
Junction Temperature: 26.2 °C
Thermal Margin: 58.8 °C (4.9 W)
Effective θJA: 11.5 °C/W
Power supplied to off-chip devices: 0 W
Confidence level: Low
Implemented Power Report
SIMULATION
Run Simulation
RTL ANALYSIS
Run Linter
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
Constraints Wizard
Edit Timing Constraints
Report Timing Summary
Report Clock Networks
Report Clock Interaction
Report Methodology
Report DRC
Report Noise
Report Utilization
Report Power
Schematic
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
Tcl Console Messages Log Reports Design Runs DRC Methodology Power Timing


project_1 - [/home/ecslogon/Desktop/project_1/project_1.xpr] - Vivado 2023.2
File Edit Flow Tools Reports Window Layout View Help
write_bitstream Complete
Default Layout
Flow Navigator
IMPLEMENTED DESIGN - xc7z020clg484-1
Project Summary
Device
case2_pipelined_adder.vhd
tb_pipelined_adder.vhd
Overview
Dashboard
URL: http://avnet.me/zedboard
Board overview: ZedBoard Zynq Evaluation and Development Kit
Changes
Synthesis
Status: Complete
Messages: 16 warnings
Part: xc7z020clg484-1
Strategy: Vivado Synthesis Defaults
Report Strategy: Vivado Synthesis Default Reports
Incremental synthesis: Automatically selected checkpoint
Implementation
Summary | Route Status
Status: Complete
Messages: 1 warning
Part: xc7z020clg484-1
Strategy: Vivado Implementation Defaults
Report Strategy: Vivado Implementation Default Reports
Incremental implementation: None
DRC Violations
Summary: 1 warning
Implemented DRC Report
Timing
Setup | Hold | Pulse Width
Worst Negative Slack (WNS): 94.372 ns
Total Negative Slack (TNS): 0 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 623
Implemented Timing Report
Utilization
Post-Synthesis | Post-Implementation
Graph | Table
Resource Estimation Available Utilization %
LUT 269 53200 0.51
FF 333 106400 0.31
BRAM 0.50 140 0.36
IO 26 200 13.00
BUFG 1 32 3.13
Power
Summary | On-Chip
Total On-Chip Power: 0.107 W
Junction Temperature: 26.2 °C
Thermal Margin: 58.8 °C (4.9 W)
Effective θJA: 11.5 °C/W
Power supplied to off-chip devices: 0 W
Confidence level: Low
Implemented Power Report
SIMULATION
Run Simulation
RTL ANALYSIS
Run Linter
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
Constraints Wizard
Edit Timing Constraints
Report Timing Summary
Report Clock Networks
Report Clock Interaction
Report Methodology
Report DRC
Report Noise
Report Utilization
Report Power
Schematic
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
Tcl Console Messages Log Reports Design Runs DRC Methodology Power Timing

7. Compare Hardware & Clock Speed:

Hardware Comparison:

For both case 1 and case 2 they use over 320 FF, but in case 1 it uses 322 registers to make the 2-pipelining work and in case 2 it uses 333 registers in order to make it work. So, in these two cases, case 1 uses less hardware than case 2 uses in order to make the design.

Clock speed Comparison:

The speed than the time the first case takes is about 100ns to go through a full clock cycle, but in case 2 it uses less time at 70ns for a full clock cycle of the design.

8. Conclusion:

We learned in this project how to implement pipelining in the Vivado environment as well as comparing two cases of our pipelining. In the first case we designed a circuit using 2 pipelines for 12 1-bit full adders and in the second case we used 3 pipelines for the 12 1-bit full adders. Between the two cases here, case 1 ended up using less hardware than case 2 but it took more time to go though a full clock cycle than case 2 did. This showed us that 3 pipelines is faster than 2 pipelines for this particular project but it is more hardware intensive to design than using less pipelines as seen. Overall, this was a very fun and interesting project as it taught me how to implement pipelining in VHDL and in the Vivado environment as well as compare 2 different instances of pipelining and the effects of using more or less pipes in the same design.