Programme: DT021A
Laboratory Technical Report

	Name	Student Number
Author:	Talha Tallat	D18124645
Collaborator 1:		
Collaborator 2:		
Collaborator 3:		

Laboratory Number:	6
Semester Week Number:	13
Date:	10/05/2022

Submission Checklist and Declaration

To ensure that the focus of the assessment of your laboratory report can include the development of higher-order skills and competencies associated with a Level 8 qualification, please complete the checklist and declaration below.

I declare that the report that I am submitting:	Tick Boxes
• is my original work, with secondary sources acknowledged;	☑
• was proofread thoroughly for typographical errors;	☑
• contains citations with references formatted in the IEEE reference citation style.	☑
I understand that my work can be returned <u>uncorrected</u> if the criteria are unfulfilled.	

By submitting this report I declare that the above conditions are fully met.

1.0 Laboratory Aim

- Task:

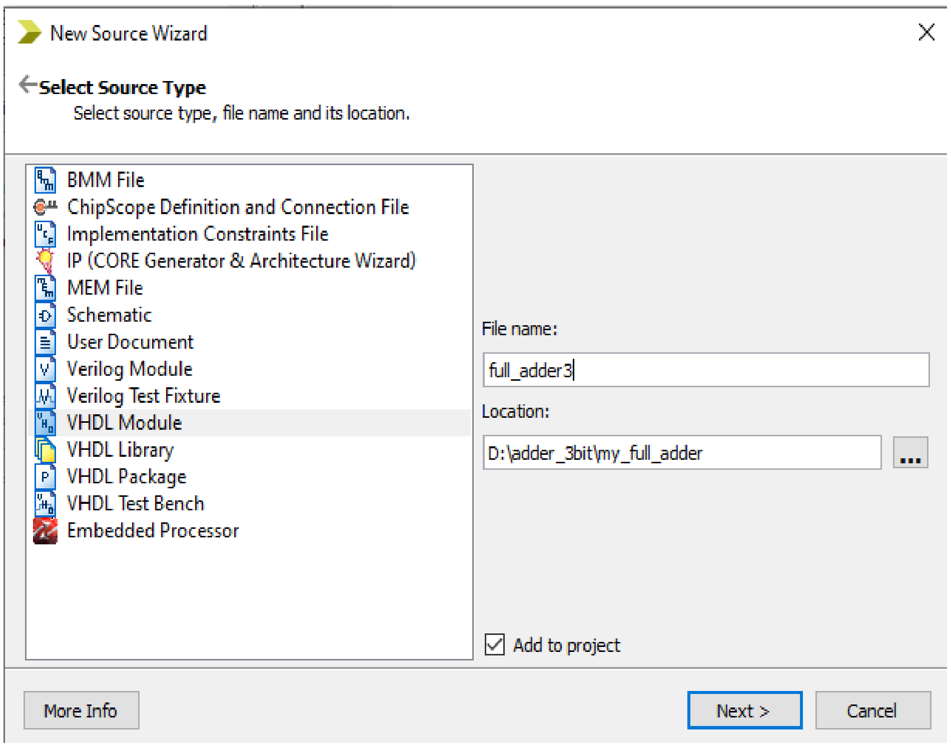
  The aim of this laboratory is to analyse structural logic circuits in the Xilinx Vivado software.

  In this lab, the task is to design an FPGA that adds two 3-bit unsigned numbers.

  - First writing a 1-bit full adder
  - Then using structural VHDL design to connect 1-bit adder to build a 2 input 3-bit adder having no carry output.
- Method:
  1. Implementing the circuit using the Xilinx Vivado design suite following the steps of the previous laboratory. Explain the code using remarks in your VHDL description.
  2. Simulating designs using an appropriate simulation strategy. Developing a testbench to carry out this simulation. Explaining code using remarks in VHDL description.
  3. Unable to test the circuit on the Basys 3 board as the board is not available.
  4. Analysing and documenting the slice logic of your circuit. And providing a brief overview of the results of the other nine sections of this report.
  5. Conclusions.

2.0 Laboratory Procedure

The lab started with the creation of the VHDL new Source file by selecting VHDL Module as indicated below in figure 1.


New Source Wizard

Select Source Type
Select source type, file name and its location.

File name: full_adder3
Location: D:\adder_3bit\my_full_adder
☑ Add to project

Figure 1 - Creating a new source file in VHDL

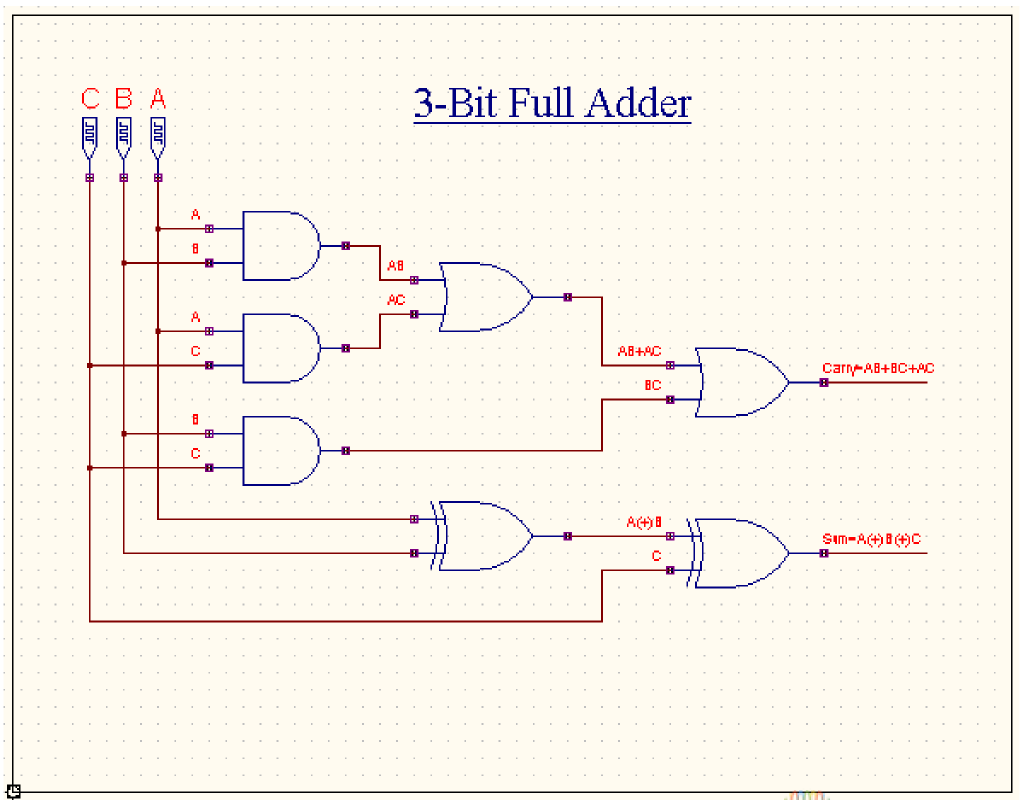


Figure 2 - 3-bit Full Adder [1]

Figure 2 shows the basic circuit for the 3-bit adder along with a simplified Boolean expression as shown below.

Truth Table

$$Sum = A\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}\overline{B}C + ABC$$

$$= A \oplus B \oplus C$$

Proof:

$$Sum = A \oplus B \oplus C$$

$$= \left(\overline{A}B + A\overline{B}\right) \oplus C$$

$$= \overline{\left(\overline{A}B + A\overline{B}\right)}C + \left(\overline{A}B + A\overline{B}\right)\overline{C}$$

$$= \left(\overline{\overline{A}B} \cdot \overline{A\overline{B}}\right)C + \left(\overline{A}B + A\overline{B}\right)\overline{C}$$

$$= \left[\left(A + \overline{B}\right)\left(\overline{A} + B\right)\right]C + \left(\overline{A}B + A\overline{B}\right)\overline{C}$$

$$= A\overline{A}C + ABC + \overline{A}\overline{B}C + B\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C}$$

$$= \underline{\underline{ABC + \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C}}}$$

- VHDL Code

  The code was created by using Xilinx ISE Software instead due to the crashes in the Vivado software and was ported on the Basys 3 development board on Artix-7 FPGA and applying signals to the board via the switches to confirm the functionality of the design. The code for one Adder circuit was written and then by using that code in components a three-bit adder circuit was formed.

- VHDL Code for full Adder

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity full_adder is
   Port ( A: in  STD_LOGIC;                          -----input and output declarations
         B: in  STD_LOGIC;
         Cin: in  STD_LOGIC;
         COUT: out  STD_LOGIC;
         SUM: out  STD_LOGIC);
end full_adder;
architecture Behavioral of full_adder is
begin
Sum <= A XOR B XOR Cin ;
Cout <= (A AND B) OR (Cin AND A) OR (Cin AND B) ;
end Behavioral;
```

- 3_bit full_adder VHDL code

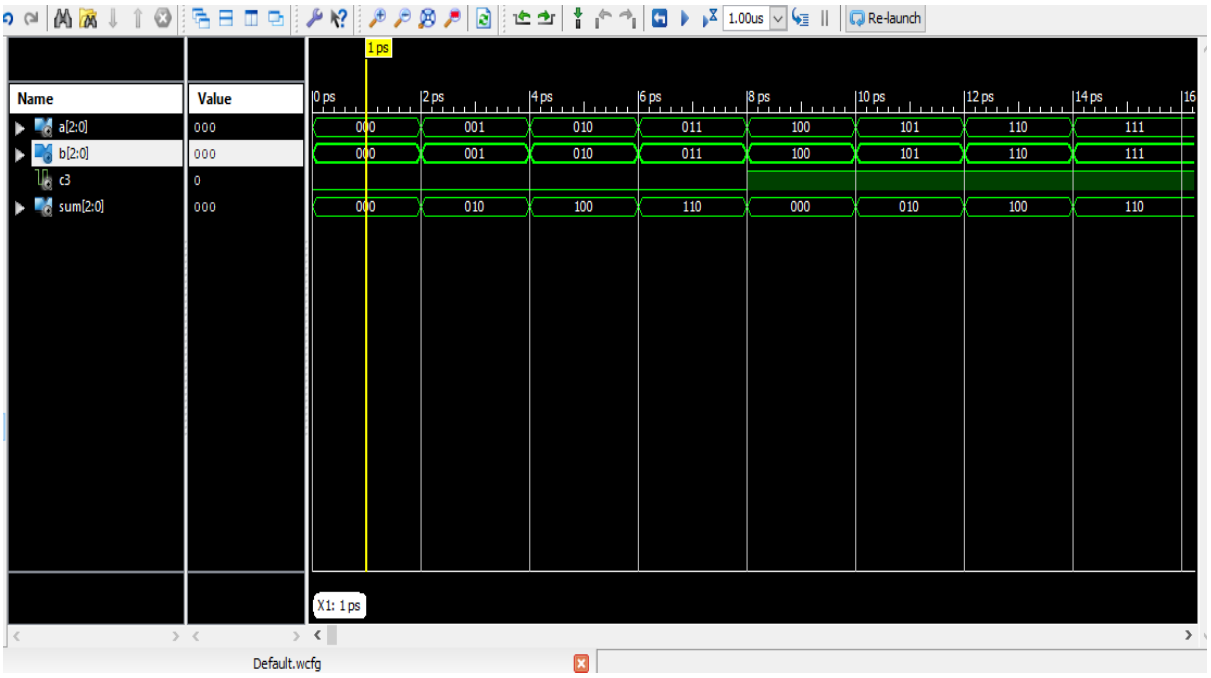
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity full_adder3 is
   Port ( A : in  STD_LOGIC_VECTOR (2 downto 0);----input vector of length 3 bits
         B : in  STD_LOGIC_VECTOR (2 downto 0);
         C3 : out  STD_LOGIC;
         Sum : OUT  STD_LOGIC_VECTOR (2 downto 0));     -----sum as output vector
                                                                  of length 3 bits
end full_adder3;
architecture Behavioral of full_adder3 is
SIGNAL C0,C1,C2 : STD_LOGIC;                         ----signal generation
component full_adder is            --------using component full_adder to add three bits
   Port ( A : in  STD_LOGIC;
         B : in  STD_LOGIC;
         Cin : in  STD_LOGIC;
         Cout : out  STD_LOGIC;
         Sum : out STD_LOGIC);
end component;
begin
FA1:full_adder Port map (A(0),B(0),'0',C1,SUM(0));        ------Calling full adder 1
FA2:full_adder Port map (A(1),B(1),C1,C2,SUM(1));         ----- Calling full adder2
FA3:full_adder Port map (A(2),B(2),C2,C3,SUM(2));         ---- Calling full adder3
end Behavioral;
```

- VHDL Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY testbench IS
END testbench;
ARCHITECTURE behavior OF testbench IS
   COMPONENT full_adder3
   PORT(
       A : IN  std_logic_vector(2 downto 0);
```

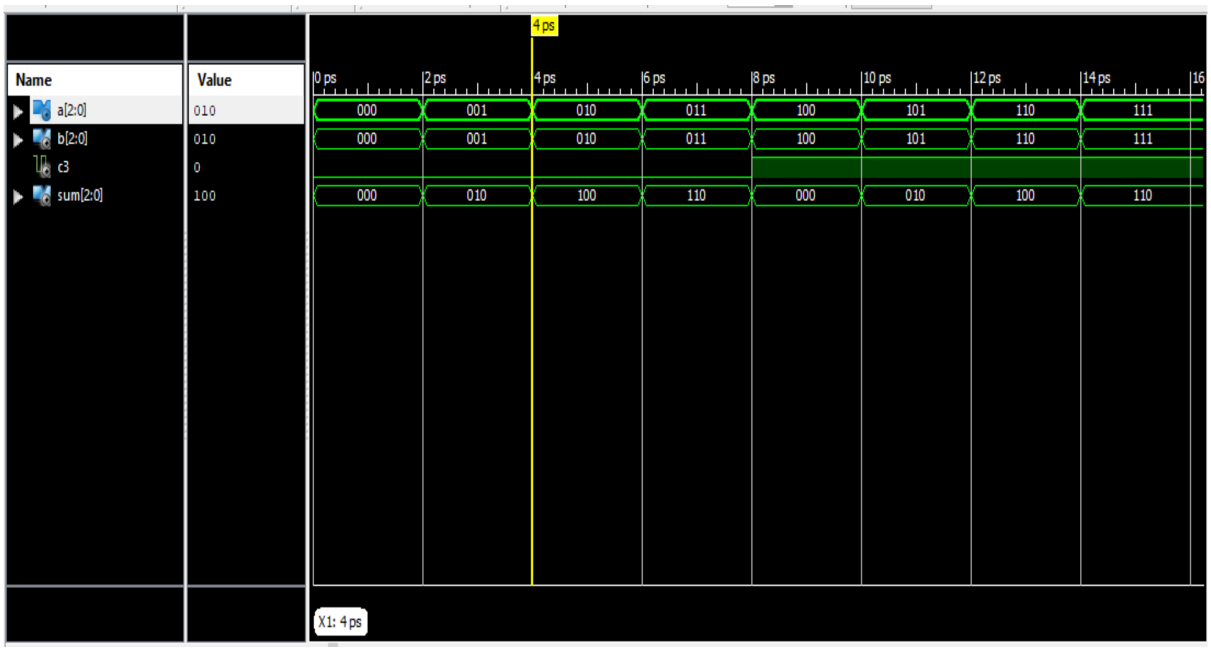
```
        B : IN  std_logic_vector(2 downto 0);
        C3 : OUT  std_logic;
        Sum : OUT  std_logic_vector(2 downto 0)
       );
    END COMPONENT;
                                                                        --Inputs
   signal A : std_logic_vector(2 downto 0) := (others => '0');
   signal B : std_logic_vector(2 downto 0) := (others => '0');
                                                                        --Outputs
   signal C3 : std_logic;
   signal Sum : std_logic_vector(2 downto 0);
 BEGIN
  the Unit Under Test (UUT)
   uut: full_adder3 PORT MAP (
          A => A,
          B => B,
          C3 => C3,
          Sum => Sum);
                                        -- Stimulus process
   stim_proc: process
   begin
                                        --------assigning value to a and b for 2 pico_seconds
            a<="000";
  b<="000";
  wait for 2 ps;
  a<="001";
  b<="001";
  wait for 2 ps;
  a<="010";
  b<="010";
  wait for 2 ps;
  a<="011";
  b<="011";
  wait for 2 ps;
  a<="100";
  b<="100";
  wait for 2 ps;
  a<="101";
  b<="101";
  wait for 2 ps;
  a<="110";
  b<="110";
  wait for 2 ps;
  a<="111";
  b<="111";
  wait for 2 ps;
   end process;
 END;
```

3.0 Results



```
Name      Value  0 ps   2 ps   4 ps   6 ps   8 ps   10 ps  12 ps  14 ps
a[2:0]    000    000    001    010    011    100    101    110    111
b[2:0]    000    000    001    010    011    100    101    110    111
c3        0
sum[2:0]  000    000    010    100    110    000    010    100    110
```

Figure 3 - Behavioural Simulation result 1



```
Name      Value  0 ps   2 ps   4 ps   6 ps   8 ps   10 ps  12 ps  14 ps
a[2:0]    010    000    001    010    011    100    101    110    111
b[2:0]    010    000    001    010    011    100    101    110    111
c3        0
sum[2:0]  100    000    010    100    110    000    010    100    110
```

Figure 4 - Behavioural Simulation result 2

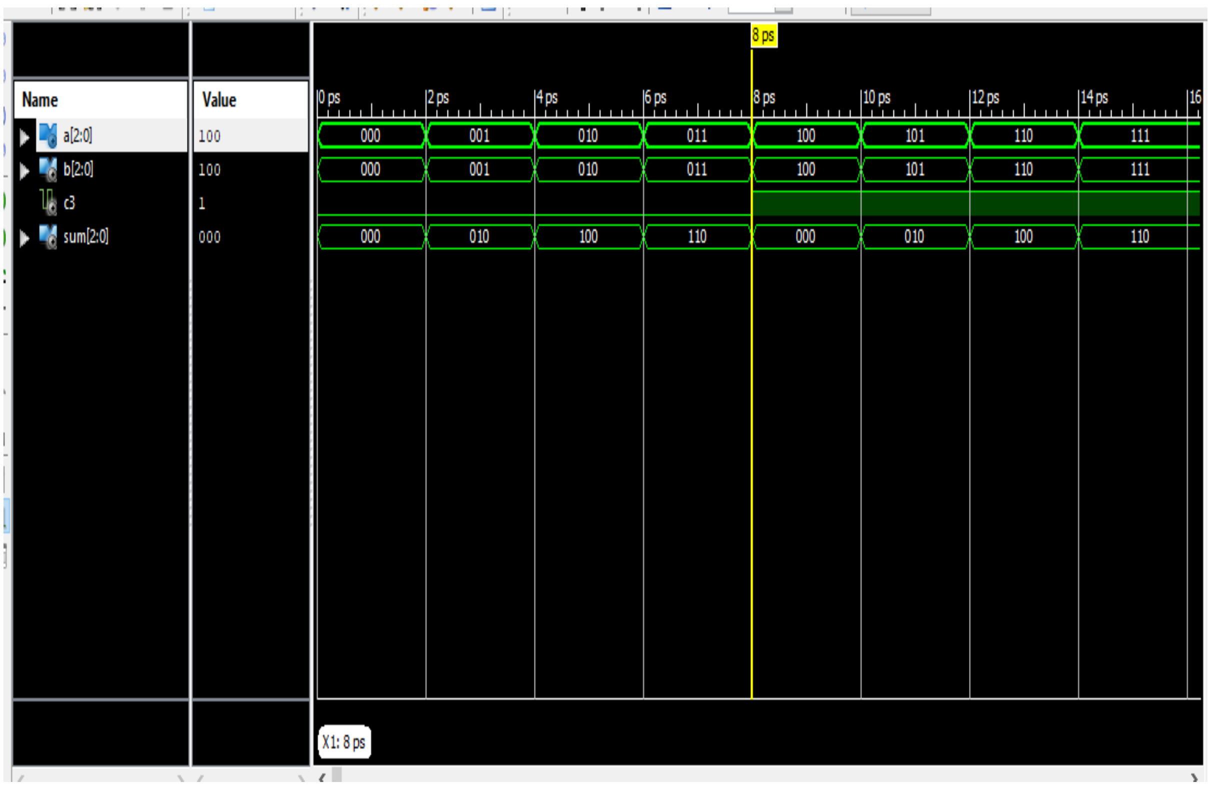


Figure 5 - Behavioural Simulation result 3

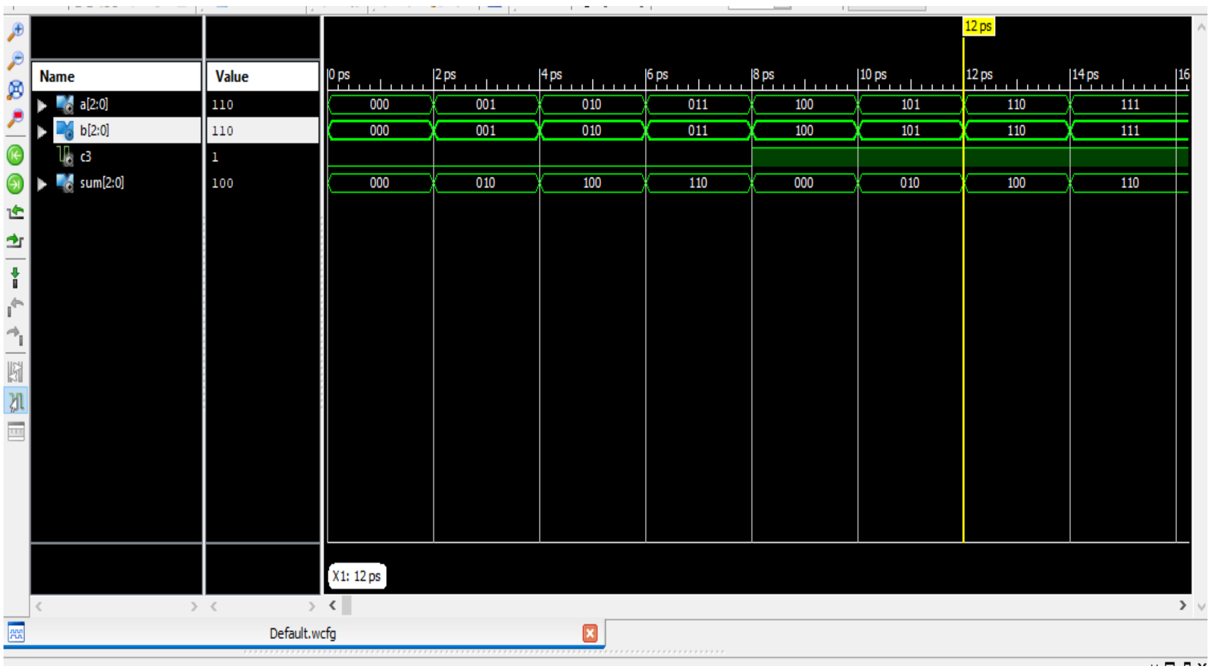


Figure 6 - Behavioural Simulation result 4

4.0 Analysis

The above results show the change in corresponding values and verify the functionality of the 3_bit full adder circuit. From the above simulations, the functionality of a three-bit full adder is verified in terms of the addition of three bits and generation of the carry bit.

- Figure 3 shows, at 1 Ps A=000, B=000, Carryout=0 and sum=000
- Figure 4 indicates, at 4 Ps A=010, B=010, Carryout=0 and sum=100
- Figure 5 indicates, at 8 Ps A=100, B=100, Carryout=1 and sum=100
- Figure 6 indicates, at 12 Ps A=110, B=110, Carryout=1 and sum=100

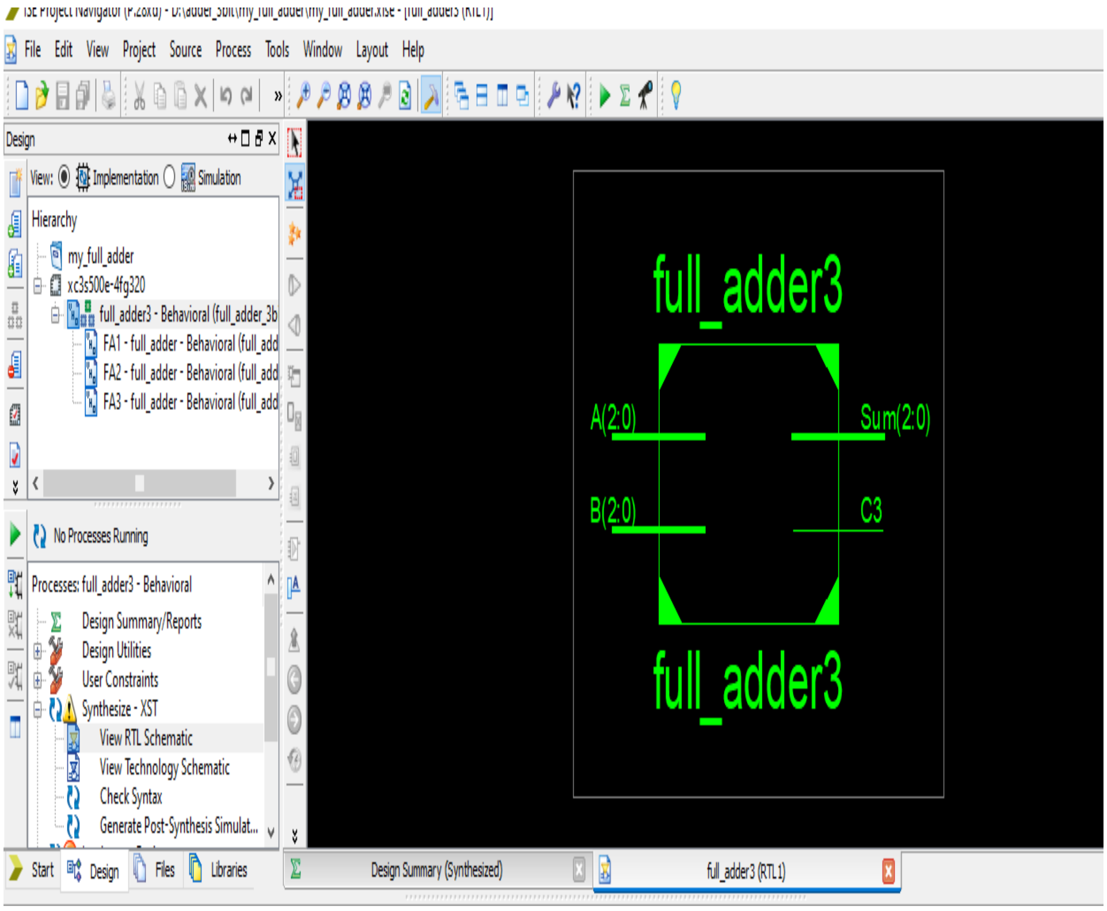

File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
my_full_adder
xc3s500e-4fg320
full_adder3 - Behavioral (full_adder_3b
FA1 - full_adder - Behavioral (full_add
FA2 - full_adder - Behavioral (full_add
FA3 - full_adder - Behavioral (full_add
No Processes Running
Processes: full_adder3 - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulat...
Start
Design
Files
Libraries
Design Summary (Synthesized)
full_adder3 (RTL1)
full_adder3
A(2:0)
B(2:0)
Sum(2:0)
C3
full_adder3

Figure 7 - RTL Schematic of the 3_bit full adder

Figure 7 shows the RTL Schematic of the 3_bit full adder, where A and B are the input vectors of the length of three bits and sum is the output with the length of three bits and the c3 is the output of a single bit.

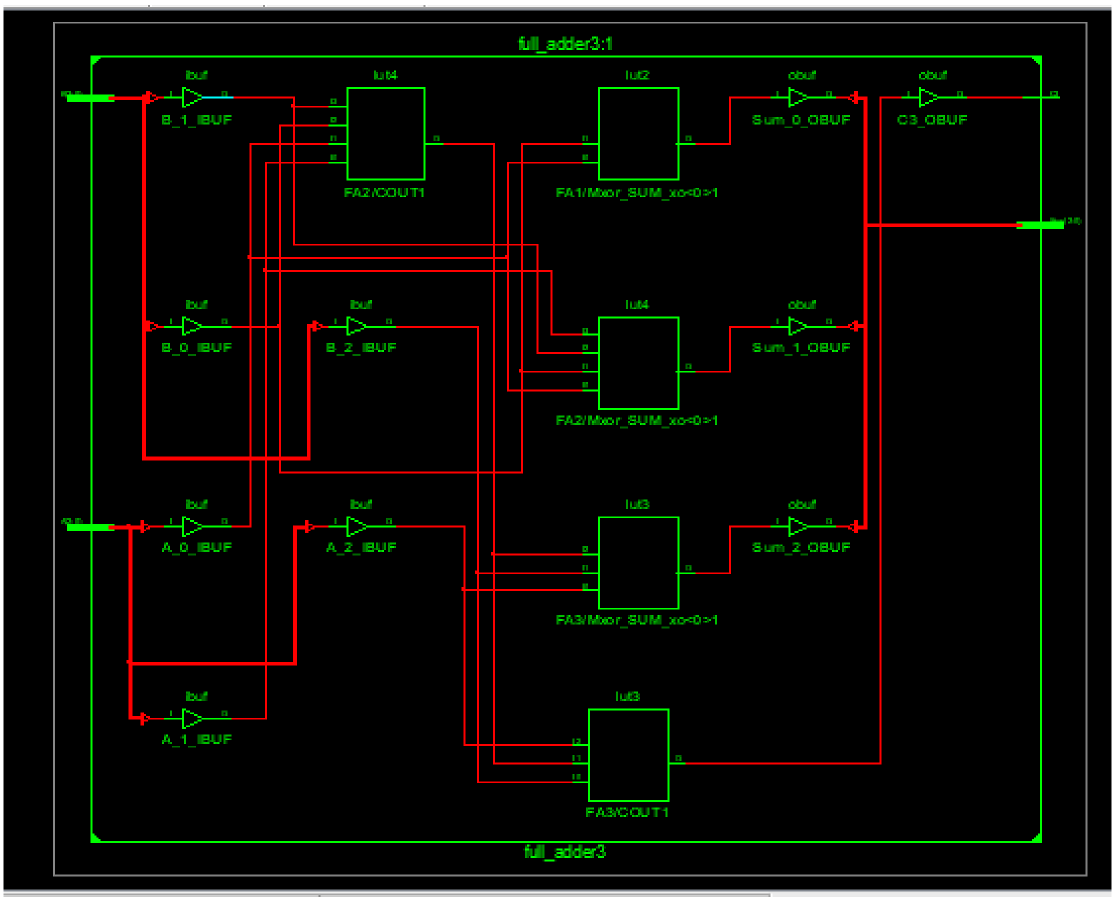
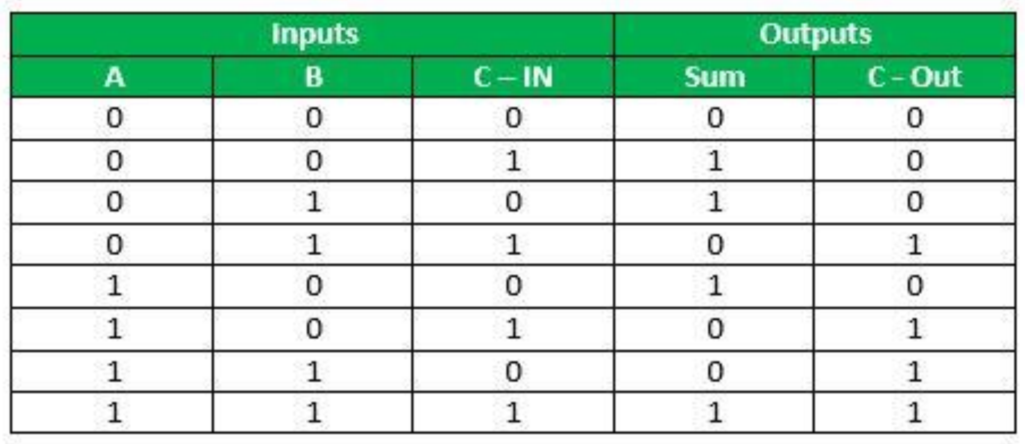


Figure 8 - Internal view of the technology used

Figure 8 shows the internal view of the technology used, which consists of four single-bit adders and buffers.

Table 1 - Input & Output results of the full adder circuit [2]



Inputs			Outputs	
A	B	C – IN	Sum	C - Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 1 shows the inputs and output results of the full adder circuit. The results show that the carryout is high only when any two of the corresponding outputs are high in the table.

5.0 Conclusions

This lab was conducted on adders and helped me to learn how to write the Verilog HDL code for a 3-bit adder and learned how to program the BASYS 3 board and simulate it in the Vivado software. My knowledge about the FPGAS has immensely increased.

- FPGA can execute a function faster than a CPU as FPGA can perform parallel processing at a faster rate.
- FPGA is reprogrammable even after the circuit has been designed and implemented meaning they are reprogrammable and reusable.
- No expensive tools are required to design or configure an FPGA chip.
- FPGAs are designed in a higher description language called HDL which is a modular programming code. Using HDL code, such as VHDL or Verilog makes the design process extremely fast and efficient. Thus, it has a faster time to market.
- FPGAs are ideal for real-time applications.

6.0 References

[1] "3bit full adder", Youspice.com, 2022. [Online]. Available: https://www.youspice.com/wp-content/uploads/2015/04/3bitfulladder.png. [Accessed: 09- May- 2022]

[2] "Google Image Result for https://media.geeksforgeeks.org/wp-content/uploads/2-41.jpg", Images.app.goo.gl, 2022. [Online]. Available: https://images.app.goo.gl/vRybWWaEryx1DowN9. [Accessed: 09- May- 2022]

[3] "Binary Adder", https://www.electronics-tutorials.ws/, 2022. [Online]. Available: https://www.electronics-tutorials.ws/combination/comb_7.html. [Accessed: 09- May- 2022]

Submission Checklist

1. The cover page is appropriately complete.	☑
2. The six sections in the body of the report are complete.	☑
3. The constructive feedback from a collaborator is addressed.	☑
4. A final spell-check is completed.	☑
5. A backup copy of the report is online.	☑