Experiment 1: FOUR BIT BCD ADDER

Mokashi Harish Mahesh, Roll Number- 210070053

August 27, 2022

1 Overview of the experiment

Aim:- To be able to write design of Four Bit BCD Adder on Quartus and to understand RTL simulation, DUT, Testbench work and Xenon board simulation of Four Bit BCD Adder.

Procedure:-

1. Made a external interface and internal design constituting total design of Four Bit BCD Adder using 2 FOUR BIT RIPPLE ADDER-SUBTRACTOR, 3 OR gates, 1 AND gate.
2. Documented a VHDL code of experiment on quartus.
3. Performed DUT test and ran RTL Simulation of experiment in ModelSim Altera Window using Quartus.
4. Performed experiment on Xenon board by configuring it with VHDL file on PC with USB port

This report contains details of my experimental setup of assignment, sketch of code of experiment design, truth table of experiment, observations which include pictures of RTL design view, RTL simulation, and conclusions.

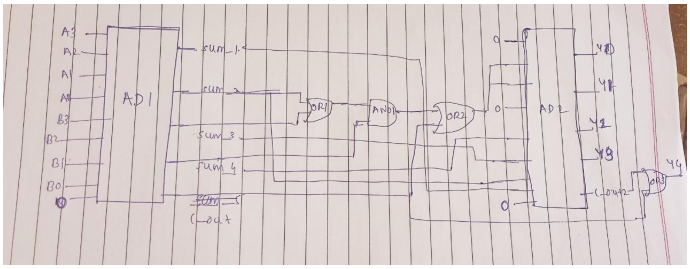

Figure 1: Design of experimental setup

2 Experimental setup of the assignment

2.1 Design documentation and design code

Design shown consists of Four Bit BCD Adder which has eight inputs A3, A2, A1, A0, B3, B2, B1, B0 and five outputs Y4, Y3, Y2, Y1, Y0. Internal structure includes 2 FOUR BIT RIPPLE ADDER-SUBTRACTORs named **AD1, AD2**, 3 OR gates named **OR1, OR2 and OR3** and 1 AND gate named **AND1**. The port connections of these gates to each other are indicated by signals **sum_1, sum_2, sum_3, sum_4, c_out, y_1, y_2, y_3, c_out2**.

In sketch of code given Four Bit BCD Adder is described using its entity(external interface) architecture(internal functionality). Entity named BCD_ADDER is shown with its inputs A3, A2, A1, A0, B3, B2, B1, B0 and outputs Y4, Y3, Y2, Y1, Y0 is shown in code in port lists. It has architecture of name Struct. Names of signals and instantiation of 2 FOUR BIT RIPPLE ADDER-SUBTRACTOR, 3 OR gates, 1 AND gate along with their port map from input via signals to output are shown in architecture part of code. Date type of input, output and all signals is *std_logic*. The code given below designs RTL view of experiment and sets up its functionality.

After RTL simulation of experiment Board test is done in order to test working of board which is followed by Pin planning of inputs, outputs to switches and LED on Xenon Board is done. Inputs A3, A2, A1, A0, B3, B2, B1, B0 are respectively matched to switches SW1, SW2, SW3, SW4, SW5, SW6, SW7, SW8 and outputs Y4, Y3, Y2, Y1, Y0 to first 5 LED's via pin no. in quartus. Then svf file of experiment is generated and it is loaded into FPGA board using UrJTAG. Then testcases were verified on by checking turning on input switches for input bit 1 and corresponding output LED's glow for

output bit 1.

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

library work;
use work.Gates.all;

entity FULL_ADDER is
port(A, B, C: in std_logic; SUM, C_OUT: out std_logic);
end entity FULL_ADDER;

architecture Design of FULL_ADDER is
signal s1, s2, s3, s4, s5, s6, s7: std_logic;
begin
N1: NAND_2 port map (A => A, B => B, Y => s1);
N2: NAND_2 port map (A => A, B => s1, Y => s2);
N3: NAND_2 port map (A => s1, B => B, Y => s3);
N4: NAND_2 port map (A => s2, B => s3, Y => s4);
N5: NAND_2 port map (A => s4, B => C, Y => s5);
N6: NAND_2 port map (A => s4, B => s5, Y => s6);
N7: NAND_2 port map (A => s5, B => C, Y => s7);
N8: NAND_2 port map (A => s6, B => s7, Y => SUM);
N9: NAND_2 port map (A => s5, B => s1, Y => C_OUT);

end architecture Design;

library ieee;
use ieee.std_logic_1164.all;

library work;
use work.Gates.all;

entity RIPPLE_ADDER_SUBTRACTOR is
port(A3, A2, A1, A0, B3, B2, B1, B0, M: in std_logic; C_OUT, S3,
S2, S1, S0: out std_logic);
end entity RIPPLE_ADDER_SUBTRACTOR;
```

```
architecture Struct of RIPPLE_ADDER_SUBTRACTOR is
component FULL_ADDER is
port(A, B, C: in std_logic; SUM, C_OUT: out std_logic);
end component FULL_ADDER;

signal sum1, sum2, sum3, sum4, c0, c1, c2: std_logic;
begin
XOR1: XOR_2 port map (A => M, B => B0, Y => sum1);
XOR2: XOR_2 port map (A => M, B => B1, Y => sum2);
XOR3: XOR_2 port map (A => M, B => B2, Y => sum3);
XOR4: XOR_2 port map (A => M, B => B3, Y => sum4);
FA1: FULL_ADDER port map (A => A0, B => sum1, C => M, SUM => S0, C_OUT => c0);
FA2: FULL_ADDER port map (A => A1, B => sum2, C => c0, SUM => S1, C_OUT => C1);
FA3: FULL_ADDER port map (A => A2, B => sum3, C => c1, SUM => S2, C_OUT => c2);
FA4: FULL_ADDER port map (A => A3, B => sum4, C => c2, SUM => S3, C_OUT => C_OUT);

end architecture Struct;


library ieee;
use ieee.std_logic_1164.all;

library work;
use work.Gates.all;

entity BCD_ADDER is
port(A3, A2, A1, A0, B3, B2, B1, B0: in std_logic; Y4,
Y3, Y2, Y1, Y0: out std_logic);
end entity BCD_ADDER;

architecture Struct of BCD_ADDER is
component RIPPLE_ADDER_SUBTRACTOR is
port(A3, A2, A1, A0, B3, B2, B1, B0, M: in std_logic; C_OUT,
S3, S2, S1, S0: out std_logic);
end component RIPPLE_ADDER_SUBTRACTOR;
signal sum_1, sum_2, sum_3, sum_4, c_out, y_1, y_2, y_3, c_out2: std_logic;
begin
AD1: RIPPLE_ADDER_SUBTRACTOR port map (A3 => A3, A2 => A2, A1 => A1, A0 => A0,
```

```
B3 => B3, B2 => B2, B1 => B1, B0 => B0, M => '0',
C_OUT => c_out, S3 => sum_4, S2 => sum_3, S1 => sum_2, S0 => sum_1);
OR1: OR_2 port map (A => sum_2, B => sum_3, Y => y_1);
AND1: AND_2 port map (A => y_1, B => sum_4, Y => y_2);
OR2: OR_2 port map (A => y_2, B => c_out, Y => y_3);
AD2: RIPPLE_ADDER_SUBTRACTOR port map (A3 => sum_4, A2 => sum_3,
A1 => sum_2, A0 => sum_1, B3 => '0', B2 => y_3, B1 => y_3, B0 => '0', M => '0',
C_OUT => c_out2, S3 => Y3, S2 => Y2, S1 => Y1, S0 => Y0);
OR3: OR_2 port map (A => c_out, B => c_out2, Y => Y4);

end architecture Struct;
```

3 Observations

A3	A2	A1	A0	B3	B2	B1	B0	Y4	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	1	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	1	0	0
0	0	0	0	0	1	0	1	0	0	1	0	1
0	0	0	0	0	1	1	0	0	0	1	1	0
0	0	0	0	0	1	1	1	0	0	1	1	1

Table 1: Some of the testcases

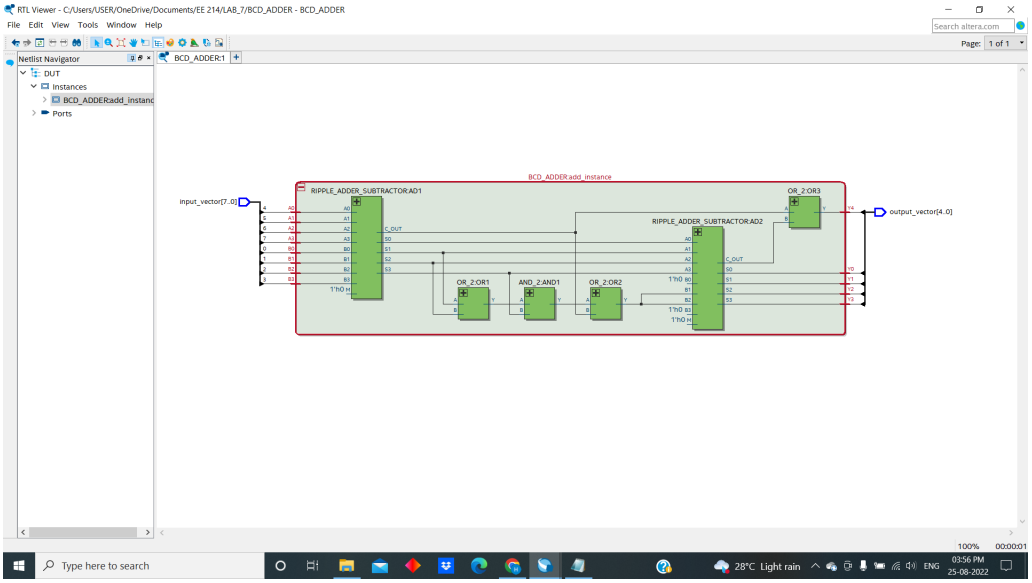

Figure 2: RTL view of experiment

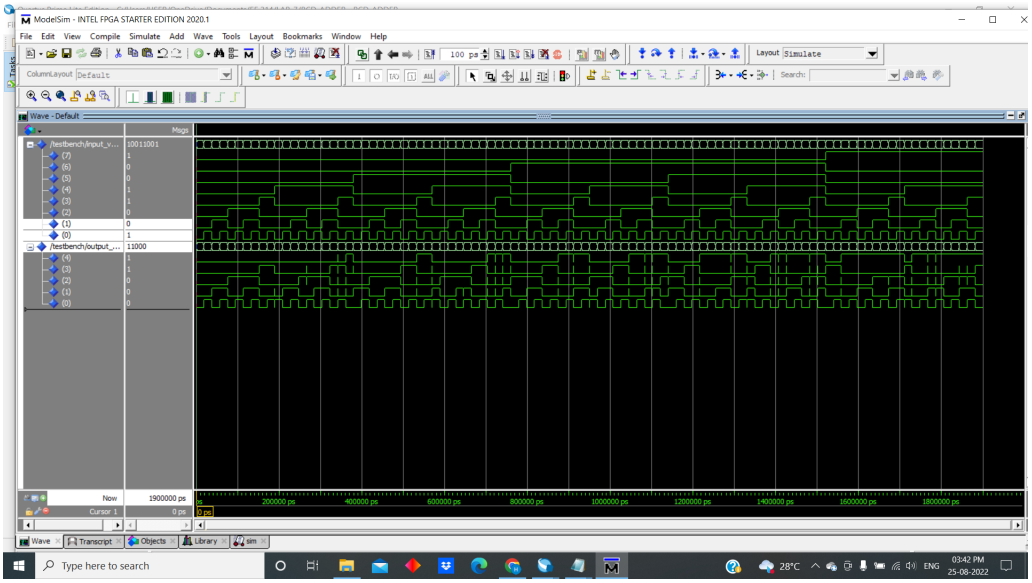

Figure 3: RTL simulation of experiment

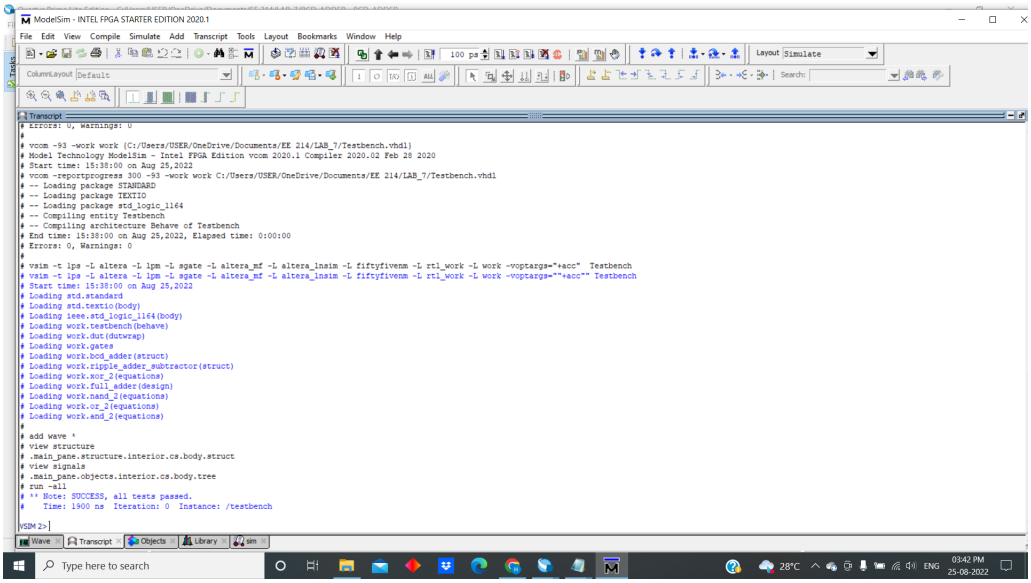

Figure 4: Success message of transcript

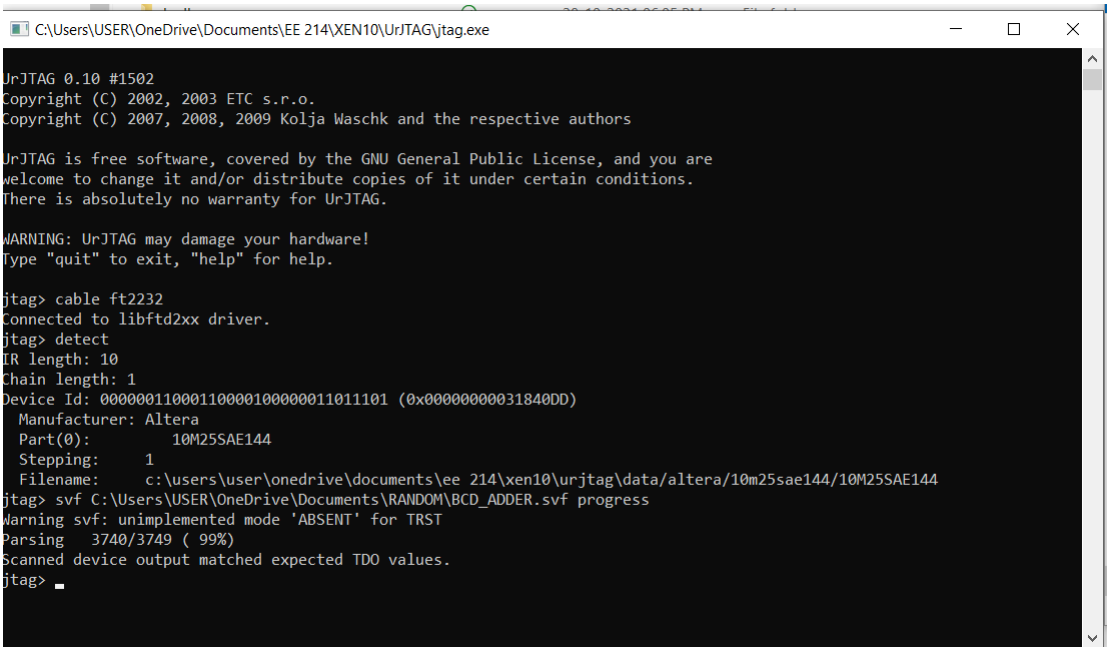

Figure 5: UrJTAG success window