Lab Report 5

Ashish Prasad, 21D180009

September 6, 2022

1 Overview of the experiment

The purpose of this experiment was to make a universal shifter circuit, which can perform a logical right shift or left shift on 8-bit input by the specified number of bits.

2 Experiment setup

2.1 Digital Design for Universal shifter

The Pen-paper design for the universal shifter is shown as follows:

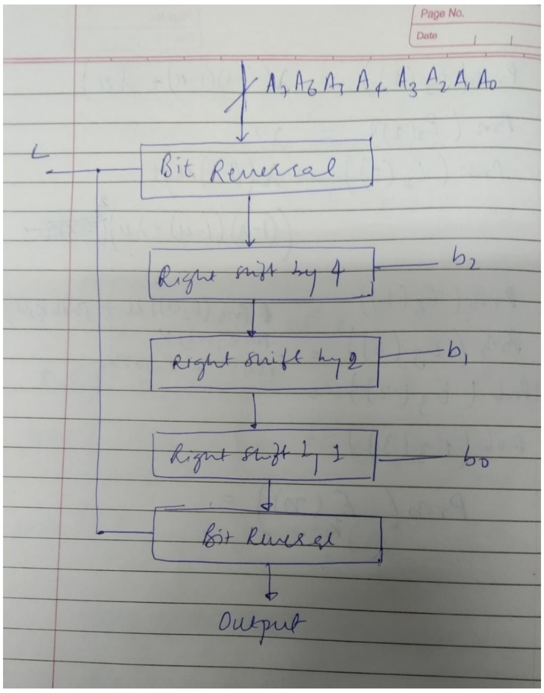


Figure 1: Design

The code for the same Universal shifter design is shown :

2.1.1 Entity: MUX

```
entity MUX  is
   port (A0, A1, S: in std_logic; Y: out std_logic);
end entity MUX;
```

2.1.2 Entity Declaration: Shifter

```
entity shifter is
 port (A: in std_logic_vector(7 downto 0);
       L: in std_logic;
        B_1: in std_logic_vector(2 downto 0);
 output: out std_logic_vector(7 downto 0));
end entity;
```

2.1.3 Architecture Declaration

```
architecture pqrs of shifter is
component bit_reversal is
 port (a: in std_logic_vector(7 downto 0);
       X: in std_logic;
        b: out std_logic_vector(7 downto 0));
end component bit_reversal;

component shift_4 is
 port (a: in std_logic_vector(7 downto 0);
       X: in std_logic;
        b: out std_logic_vector(7 downto 0));
end component shift_4;

component shift_2 is
 port (a: in std_logic_vector(7 downto 0);
       X: in std_logic;
        b: out std_logic_vector(7 downto 0));
end component shift_2;

component shift_1 is
 port (a: in std_logic_vector(7 downto 0);
       X: in std_logic;
        b: out std_logic_vector(7 downto 0));
end component shift_1;

signal s1,s2,s3,s4 : std_logic_vector(7 downto 0);
------------------
begin
l1: bit_reversal port map(a =>A, X => L, b=> s1);
```

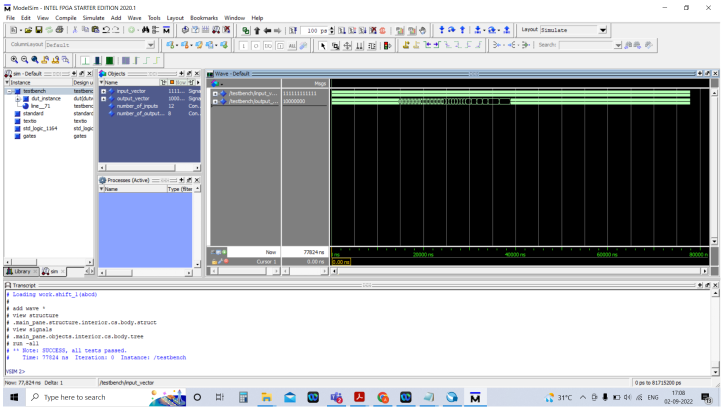
```
l2: shift_4 port map(a =>s1, X => B_1(2), b => s2);
l3: shift_2 port map(a =>s2, X => B_1(1), b => s3);
l4: shift_1 port map(a =>s3, X => B_1(0), b => s4);
l5: bit_reversal port map(a =>s4, X => L, b => output);
end architecture pqrs;
```

3 Observations

3.1 Digital Design

The Design passed all the test cases as given in TRACEFILE the RTL Simulation for the same is shown below.

3.2 RTL Simulation



```
# Loading work.shift_1(abcd)
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# ** Note: SUCCESS, all tests passed.
#    Time: 77824 ns  Iteration: 0  Instance: /testbench

VSIM 2>
```

Now: 77,824 ns Delta: 1 /testbench/input_vector

Figure 2: RTL Simulation

3.3 RTL View

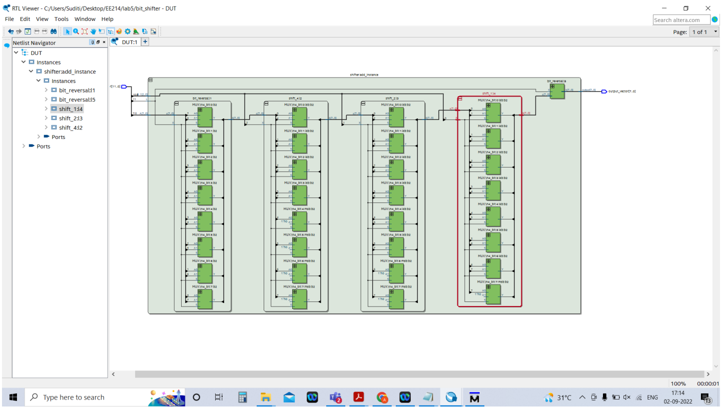
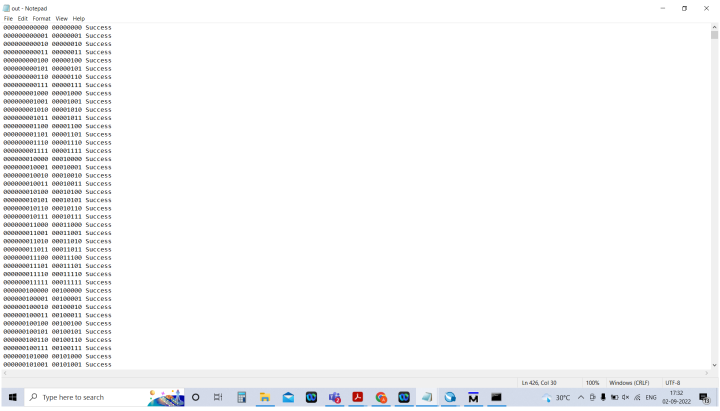


Figure 3: RTL View

3.4 Final Observation



```
000000000000 00000000 Success
000000000001 00000001 Success
000000000010 00000010 Success
000000000011 00000011 Success
000000000100 00000100 Success
000000000101 00000101 Success
000000000110 00000110 Success
000000000111 00000111 Success
000000001000 00001000 Success
000000001001 00001001 Success
000000001010 00001010 Success
000000001011 00001011 Success
000000001100 00001100 Success
000000001101 00001101 Success
000000001110 00001110 Success
000000001111 00001111 Success
000000010000 00010000 Success
000000010001 00010001 Success
000000010010 00010010 Success
000000010011 00010011 Success
000000010100 00010100 Success
000000010101 00010101 Success
000000010110 00010110 Success
000000010111 00010111 Success
000000011000 00011000 Success
000000011001 00011001 Success
000000011010 00011010 Success
000000011011 00011011 Success
000000011100 00011100 Success
000000011101 00011101 Success
000000011110 00011110 Success
000000011111 00011111 Success
000000100000 00100000 Success
000000100001 00100001 Success
000000100010 00100010 Success
000000100011 00100011 Success
000000100100 00100100 Success
000000100101 00100101 Success
000000100110 00100110 Success
000000100111 00100111 Success
000000101000 00101000 Success
000000101001 00101001 Success
```