Experiment 5: Universal Shifter

Debasish Panda 21D070021

September 6, 2022

1 Overview of the experiment

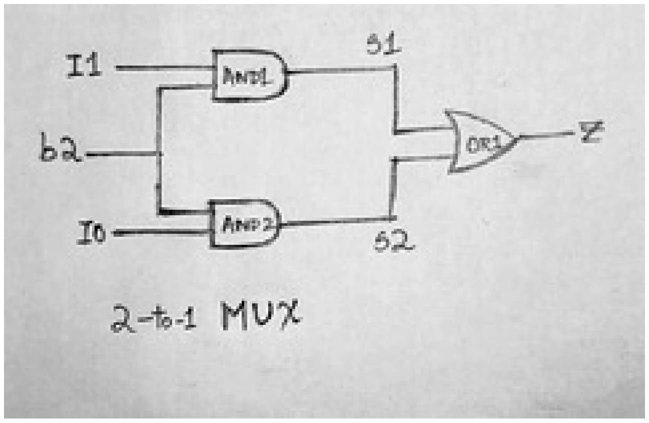
The purpose of the experiment is to design a Universal Shifter using fundamental logic gates and 2-to-1 MUXes as well as describe them in VHDL using Structural Modelling. Following a successful RTL simulation, we have to demonstrate the same using Scanchain.
In order to perform the experiment, first the Bit Reversal unit of the Universal Shifter is designed using MUXes. Following this, the Right Shift units are also designed using MUXes. A rough sketch of the digital circuit is drawn first. The appropriate code for Structural Modelling of these components is written using VHDL on Quartus Prime software. Next, we have to demonstrate the Bit Reversal Unit practically by using the Xen10 FPGA board and Scanchain. In order to run the VHDL code on the board, we also need to determine the input and output pins correctly as per the specifications of the circuit board.
In this report first we discuss the setup required for the experiment and the approach used by me to solve the assignment. Attached alongwith are the pen-and-paper diagram and picture showing the simulation output of the design.

2 Experimental setup

First we need to design the Universal Shifter using logic gates and MUXes. As per the specifications, if L = 0, the output will be right shifted version of the input and if L = 1, the output will be left shifted version of the input.

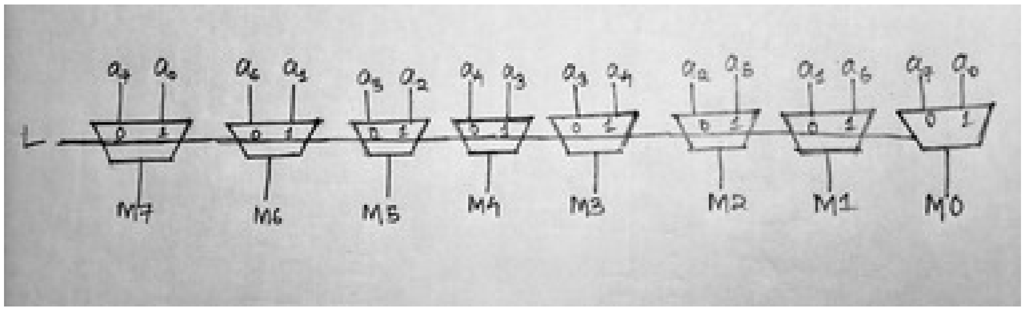
This is ensured by the two Bit Reversal units at the beginning and the end. We have already been provided with the 4-bit right shifter which we can use to design the 2-bit and 1-bit right shifters. These shifters are controlled by bits b2, b1, and b0 respectively.

The pen-and-paper diagram of the MUX unit is:



2-to-1 Multiplexer

The pen-and-paper diagram of the Bit Reversal unit is:



Bit Reversal Unit

Now that we have the required circuit, we need to assign various component instances, architecture and signals as per VHDL syntax. The required code for the same is discussed in the next section.

2.1 Design Documentation and VHDL Code

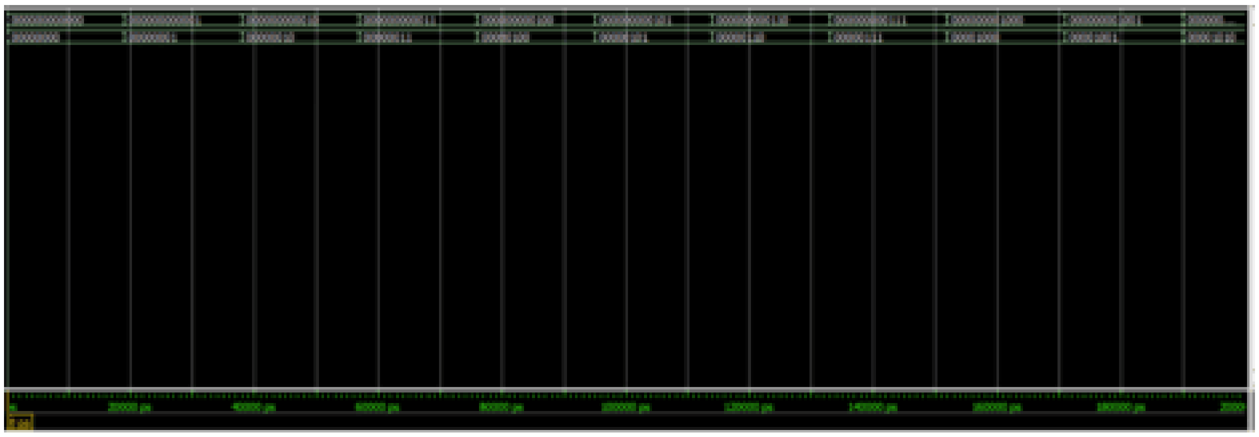
The VHDL code for designing the entity-architecture pairs associated with the digital circuit is as follows:

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

entity MUX is
 port (A: in std_logic_vector(1 downto 0); S: in std_logic_vector(0 downto 0); OUT
end entity MUX;

architecture Struct of MUX is
 signal S_BAR, S0, S1: std_logic;
begin
  AND1: AND_2 port map (A => A(0), B => S_BAR, Y => S0);
  AND2: AND_2 port map (A => A(1), B => S(0), Y => S1);
  INV1: INVERTER port map (A => S(0), Y => S_BAR);
  OR1: OR_2 port map (A => S0, B => S1, Y => OUTPUT);
end Struct;

--------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
library work;
use work.Gates.all;

entity Universal_Shifter is
 port (A: in std_logic_vector(7 downto 0); B: in std_logic_vector(2 downto 0); L:
end entity Universal_Shifter;

architecture Complex of Universal_Shifter is
 signal Q: std_logic_vector(7 downto 0); signal R: std_logic_vector(7 downto 0); s

   component MUX is
port (A: in std_logic_vector(1 downto 0); S: in std_logic_vector(0 downto 0); OUTP
end component MUX;

   for all: MUX
   use entity work.MUX(Struct);

begin
 mux_1: for i in 0 to 7 generate
 MUX1: MUX port map (A(0) => A(i), A(1) => A(7-i), S(0) => L(0), OUTPUT => Q(i));
 end generate;

 n4_bit : for i in 0 to 7 generate
 lsb: if i < 4 generate
 MUX2: MUX port map(A(0) => Q(i), A(1) => Q(i+4), S(0) => B(2), OUTPUT => R(i));
 end generate lsb;
 msb: if i > 3 generate
 MUX2: MUX port map(A(0) => Q(i), A(1) => '0', S(0) => B(2), OUTPUT => R(i));
 end generate msb;
 end generate;

 n2_bit : for i in 0 to 7 generate
 lsb: if i < 6 generate
 MUX3: MUX port map(A(0) => R(i), A(1) => R(i+2), S(0) => B(1), OUTPUT => T(i));
 end generate lsb;
 msb: if i > 5 generate
 MUX3: MUX port map(A(0) => R(i), A(1) => '0', S(0) => B(1), OUTPUT => T(i));
 end generate msb;
 end generate;

 n1_bit : for i in 0 to 7 generate
 lsb: if i < 7 generate
 MUX4: MUX port map(A(0) => T(i), A(1) => T(i+1), S(0) => B(0), OUTPUT => U(i));
 end generate lsb;
 msb: if i > 6 generate
 MUX4: MUX port map(A(0) => T(i), A(1) => '0', S(0) => B(0), OUTPUT => U(i));
 end generate msb;
 end generate;

 mux_2: for i in 0 to 7 generate
 MUX5: MUX port map (A(0) => U(i), A(1) => U(7-i), S(0) => L(0), OUTPUT => Y(i));
 end generate;

end Complex;
```

A new addition to this lab's VHDL code is the *std_logic_vector* class, which is used to easily define a large number of similar circuit elements by representing them in the form of an array.

3 Observations

The experiment was conducted successfully, and the expected output was obtained through the RTL Simulation. The Scanchain also gave the desired result, with all test cases passing successfully. The expected output of the Universal Shifter is shown in the form of the final RTL Simulation result of the design.



RTL Simulation of Universal Shifter

Following this, we have to do the pin planning correctly and convert our VHDL file to a .svf file, which is executable by the Xen10 board and is also needed for running the Scanchain program.

4 References

[1] EE214 Github page

[2] Overleaf LaTeX tutorial for beginners

[3] Introduction to Xen10 Board