CS 288: Multiplexer and Demultiplexer

Dheerendra Singh Rathor
120050033
dheeru.rathor14@gmail.com

February 16, 2014

Aim:

Using Xilinix ISE Tools to simulate 8x1 Multiplexer and 1x8 Demultiplexer with
a. Concurrent Statements
b. Sequential Statements

Procedure:

The procedure for multiplexer and demultiplexer is given in the following subsections respectively

1 Multiplexer

Multiplexer is a electronic circuit (also know as mux) which have 8 input lines and 1 output line with 3 select lines.
For a given configuration of select lines, multiplexer assign output value as one of the input lines selected by select lines. For designing multiplex, first I included 8 input lines represent by i0 t0 i7, 3 select lines represented by s0 to s2 and a output line represented by output.
Here is the selection table.

S. No.	s0	s1	s2	selected input for output
1.	0	0	0	i0
2.	0	0	1	i1
3.	0	1	0	i2
4.	0	1	1	i3
5.	1	0	0	i4
6.	1	0	1	i5
7.	1	1	0	i6
8.	1	1	1	i7

This code is written in concurrent behaviour using when statements. Timing diagrams and code are included in coming sections.

2 Demultiplexer

Demultiplexer is a electronic circuit (also know ans demux) which have 1 input lines and 8 output line with 3 select lines. This do the reverse process of what multiplexer do.
For a given configuration of select lines, multiplexer assign one of the output line as the input lines selected by select lines. For designing multiplex, first I included 1 input lines represent by input, 1 select bus of size 3 (2 downto 0) represented by sel and 1 output bus of size 8 (7 downto 0) represented by output.
Here is the selection table for demultiplexer

S. No.	Sel Vector	Output Vector
1.	000	00000001
2.	001	00000010
3.	010	00000100
4.	011	00001000
5.	100	00010000
6.	101	00100000
7.	110	01000000
8.	111	10000000

This code is written in sequential behaviour using "case - when" statement. Timing diagram and code are included in coming sections.

Timing Diagrams

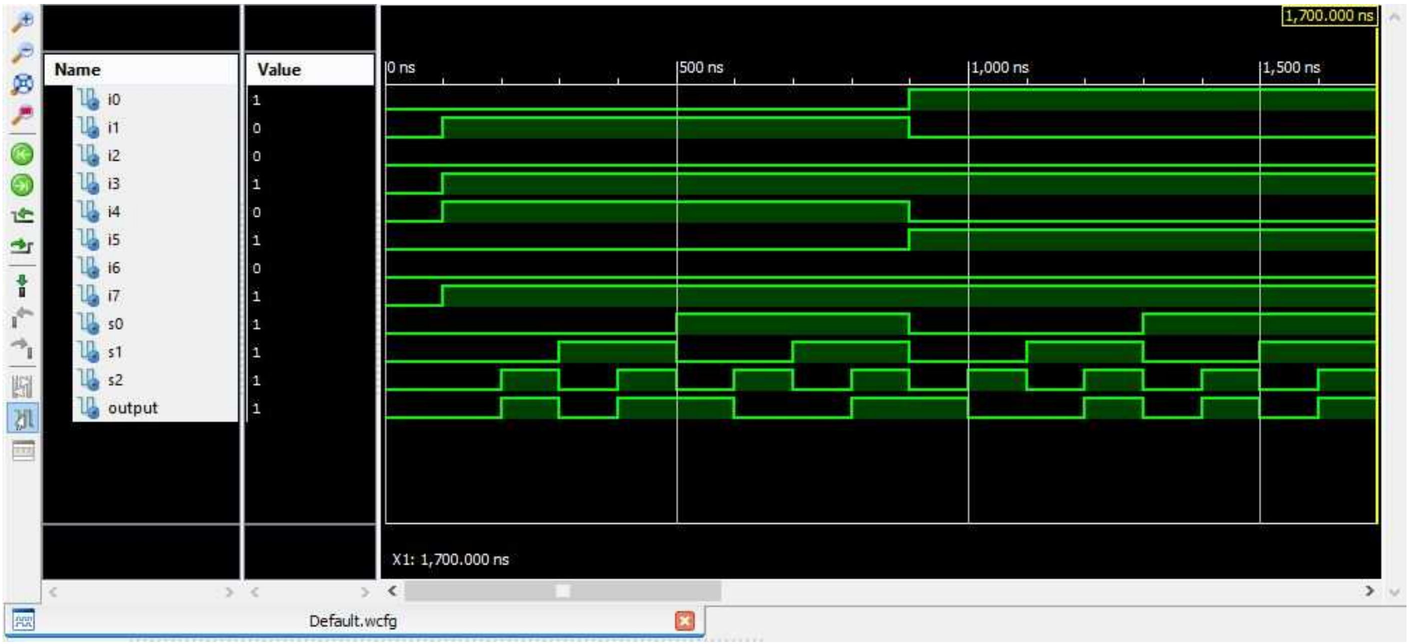

Figure 1: Timing Diagram for 8x1 multiplexer from 0ns to 1700 ns

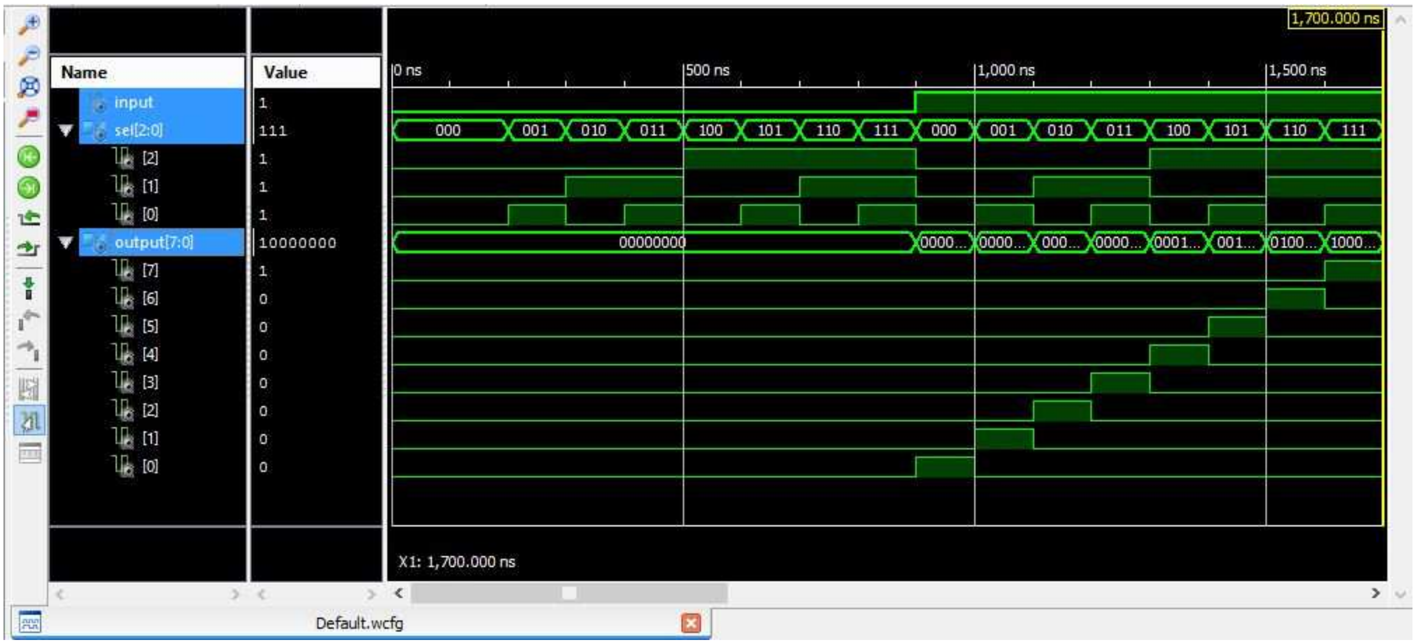

Figure 2: Timing Diagram for 1x8 demultiplexer from 0ns to 1700 ns

Codes

Here I have include working code for both multiplexer and demultiplexer

1 code for multiplexer

```
------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    14:28:18 02/11/2014
-- Design Name:
-- Module Name:    multiplexer - 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;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity multiplexer is
    Port ( i0 : in  STD_LOGIC;
           i1 : in  STD_LOGIC;
           i2 : in  STD_LOGIC;
           i3 : in  STD_LOGIC;
           i4 : in  STD_LOGIC;
           i5 : in  STD_LOGIC;
           i6 : in  STD_LOGIC;
           i7 : in  STD_LOGIC;
           s0 : in  STD_LOGIC;
           s1 : in  STD_LOGIC;
           s2 : in  STD_LOGIC;
           output : out  STD_LOGIC);
end multiplexer;

architecture Behavioral of multiplexer is

begin
output <= i0 when s0 = ’0’ and s1 = ’0’ and s2 = ’0’ else
          i1 when s0 = ’0’ and s1 = ’0’ and s2 = ’1’ else
          i2 when s0 = ’0’ and s1 = ’1’ and s2 = ’0’ else
          i3 when s0 = ’0’ and s1 = ’1’ and s2 = ’1’ else
          i4 when s0 = ’1’ and s1 = ’0’ and s2 = ’0’ else
          i5 when s0 = ’1’ and s1 = ’0’ and s2 = ’1’ else
          i6 when s0 = ’1’ and s1 = ’1’ and s2 = ’0’ else
          i7 when s0 = ’1’ and s1 = ’1’ and s2 = ’1’ ;
end Behavioral;
```

2 Code for Demultiplexer

```
-----------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    15:32:55 02/11/2014
-- Design Name:
-- Module Name:    demux - 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;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity demux is
    Port ( input : in  STD_LOGIC;
           sel : in  STD_LOGIC_VECTOR (2 downto 0);
           output : out  STD_LOGIC_VECTOR (7 downto 0));
end demux;

architecture Behavioral of demux is
begin process (input, sel)
begin
output <= "00000000";
        case sel is
                when "000" =>
```

```
                        output(0) <= input;
                when "001" =>
                        output(1) <= input;
                when "010" =>
                        output(2) <= input;
                when "011" =>
                        output(3) <= input;
                when "100" =>
                        output(4) <= input;
                when "101" =>
                        output(5) <= input;
                when "110" =>
                        output(6) <= input;
                when "111" =>
                        output(7) <= input;
                when others =>
                        null;
        end case;
end process;
end Behavioral;
```

Inference

In this assignment I infered the followings:

1. Working of 8x1 multiplexer and 1x8 demultiplexers
2. Wrinting VHDL code in behavioral style rather then structural style
3. Working with Logic Vectors in VHDL
4. Use of select lines with multplexers and demultiplexer
5. Use of select lines as decoders