Requirements

1. Design the function F using the VHDL behavior model of an 8-to-1 multiplexer.
2. Simulate and test the VHDL design using a VHDL test bench.
3. Capture the waveform and verify the output with the truth table.
4. Implement the function F on the Nexys 4 FPGA board with switches SW0 (D), SW1 (C), SW2 (B), and SW3 (A) as inputs and LED LD0 (F) as output.

Learning Objectives

The general learning objectives of this lab were to design the function F using an 8-to-1 multiplexer and to implement it on the Nexys 4 FPGA board using the VHDL design feature of Xilinx ISE design studio.

$$F(A,B,C,D) = \sum m(1,3,5,8,10,11,15)$$

General Steps

The general steps needed to complete this lab were to design an 8-to-1 multiplexer using a VHDL behavioral model and then to modify that VHDL file to include the function F. Another step needed to do the lab was to derive the truth table for the function F. The next step is to create the VHDL test bench and simulate the design with it. Compare the waveform outputs to the truth table. Finish synthesizing the design after fixing any errors. Lastly, map the inputs to the Nexys 4 FPGA board using a user constraint file and test the design.

Detailed Steps

Some detailed steps to complete this lab were . . .

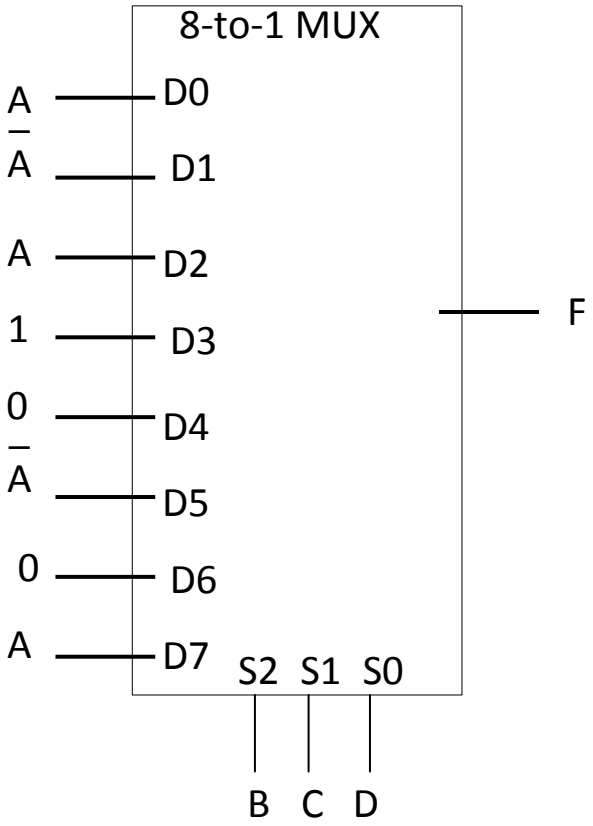
1. Write the truth table from the function F given.

ABCD	F	
0000	0	$F = A$
0001	1	$F = \bar{A}$
0010	0	$F = A$
0011	1	$F = 1$
0100	0	$F = 0$
0101	1	$F = \bar{A}$
0110	0	$F = 0$
0111	0	$F = A$
1000	1	$F = A$
1001	0	$F = \bar{A}$
1010	1	$F = A$
1011	1	$F = 1$
1100	0	$F = 0$
1101	0	$F = \bar{A}$
1110	0	$F = 0$
1111	1	$F = A$

2. Design the 8-to-1 multiplexer.

S_2 S_1 S_0	F

000	D_0
001	D_1
010	D_2
011	D_3
100	D_4
101	D_5
110	D_6
111	D_7



3. Write an 8-to-1 multiplexer in VHDL.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY mux8to1 IS
        PORT( D : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                        S : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
                        F : OUT STD_LOGIC );
END mux8to1;

ARCHITECHTURE behavioral of mux8to1 IS
```

```
BEGIN
	PROCESS(D, S)
	BEGIN
		CASE S IS
			WHEN "000" => F <= D(0);
			WHEN "001" => F <= D(1);
			WHEN "010" => F <= D(2);
			WHEN "011" => F <= D(3);
			WHEN "100" => F <= D(4);
			WHEN "101" => F <= D(5);
			WHEN "110" => F <= D(6);
			WHEN "111" => F <= D(7);
		END CASE;
	END PROCESS;
END behavioral;
```

4. Modify the 8-to-1 multiplexer to incorporate the function F.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity mux8to1 is
   Port ( A : in STD_LOGIC;
	S : in  STD_LOGIC_VECTOR (2 downto 0);
       F : out  STD_LOGIC);
end mux8to1;

architecture Behavioral of mux8to1 is

begin
	process(A, S)
	begin
		case S is
			WHEN "000" => F <= A;
			WHEN "001" => F <= (NOT A);
			WHEN "010" => F <= A;
			WHEN "011" => F <= '1';
			WHEN "100" => F <= '0';
			WHEN "101" => F <= (NOT A);
			WHEN "110" => F <= '0';
			WHEN "111" => F <= A;
			WHEN OTHERS => F <= '0';
		end case;
	end process;



end Behavioral;
```

5. Write a VHDL test bench to test the design.

```
A <= '0'; S(2) <= '0'; S(1) <= '0'; S(0) <= '0';
wait for 10 ns;

A <= '0'; S(2) <= '0'; S(1) <= '0'; S(0) <= '1';
wait for 10 ns;

A <= '0'; S(2) <= '0'; S(1) <= '1'; S(0) <= '0';
wait for 10 ns;

A <= '0'; S(2) <= '0'; S(1) <= '1'; S(0) <= '1';
wait for 10 ns;

A <= '0'; S(2) <= '1'; S(1) <= '0'; S(0) <= '0';
wait for 10 ns;

A <= '0'; S(2) <= '1'; S(1) <= '0'; S(0) <= '1';
wait for 10 ns;

A <= '0'; S(2) <= '1'; S(1) <= '1'; S(0) <= '0';
wait for 10 ns;

A <= '0'; S(2) <= '1'; S(1) <= '1'; S(0) <= '1';
wait for 10 ns;

A <= '1'; S(2) <= '0'; S(1) <= '0'; S(0) <= '0';
wait for 10 ns;

A <= '1'; S(2) <= '0'; S(1) <= '0'; S(0) <= '1';
wait for 10 ns;

A <= '1'; S(2) <= '0'; S(1) <= '1'; S(0) <= '0';
wait for 10 ns;

A <= '1'; S(2) <= '0'; S(1) <= '1'; S(0) <= '1';
wait for 10 ns;

A <= '1'; S(2) <= '1'; S(1) <= '0'; S(0) <= '0';
wait for 10 ns;

A <= '1'; S(2) <= '1'; S(1) <= '0'; S(0) <= '1';
wait for 10 ns;

A <= '1'; S(2) <= '1'; S(1) <= '1'; S(0) <= '0';
wait for 10 ns;

A <= '1'; S(2) <= '1'; S(1) <= '1'; S(0) <= '1';
```

wait for 10 ns;

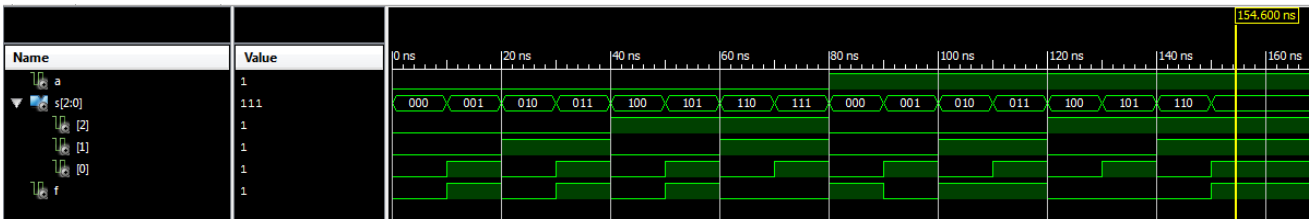
6. Simulate the design with the test bench.
7. Capture the waveform and verify the waveform output by comparing the output with the truth table.
8. Synthesize the design after fixing any errors.
9. Add a new source: a constraints file.
10. Map the inputs A, B (S2), C (S1), and D (S0) to switches SW0, SW1, SW2, and SW3 respectively and output F to LED LD0 on the user constraint file.

```
NET "A"        LOC=J15 | IOSTANDARD=LVCMOS33; #IO_L24N_T3_RS0_15
NET "S<2>"         LOC=L16 | IOSTANDARD=LVCMOS33; #IO_L3N_T0_DQS_EMCCLK_14
NET "S<1>"         LOC=M13 | IOSTANDARD=LVCMOS33; #IO_L6N_T0_D08_VREF_14
NET "S<0>"         LOC=R15 | IOSTANDARD=LVCMOS33; #IO_L13N_T2_MRCC_14
NET "F"        LOC=H17 | IOSTANDARD=LVCMOS33; #IO_L18P_T2_A24_15
```

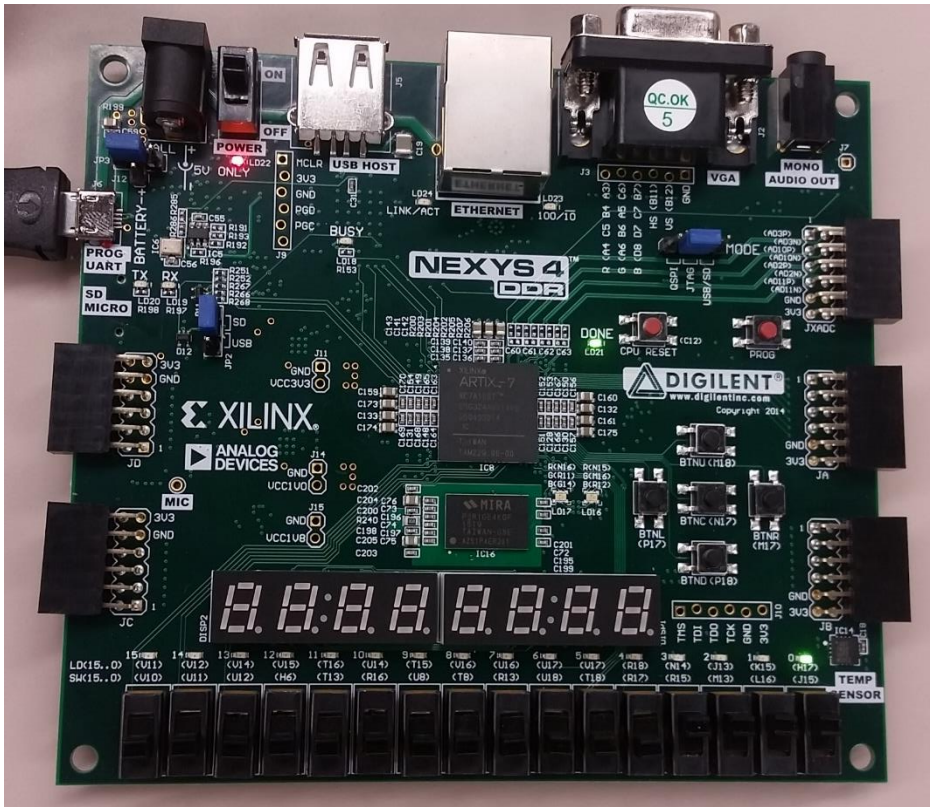
11. Synthesize the design at the top module after completing the user constraint file.
12. Generate the programming file.
13. Plug in the Nexys 4 board and click configure target device.
14. Double click on boundary scan and then click on initialize chain to detect your FPGA.
15. Select the .bit file to be configured.
16. Select No to attaching an SPI of BPI PROM.
17. Click ok to configure the .bit file.
18. Double click on program to configure the FPGA.
19. After configuration succeeds, test the board.

Observations

During the lab, I noticed that the test bench file generated for the VHDL module was different than the one generated for a schematic module. I had to comment out the parts that were asking for a clock definition for the test bench to run. After simulating the design, I observed the following waveform and compared it to the truth table for the function F.



After completing this simulation, I configured the Nexys 4 board and tested the switches to see the LED output.



Summary

In this lab, I learned how to create a VHDL module. Then I learned how to create and run a test bench for a VHDL module. I also learned how to map inputs and outputs to an FPGA board with a user constraint file. Then I learned to configure the Nexys 4 board. I learned how to test my VHDL module by moving the switches and observing the LED output.