Laboratory Exercise 1

Switches, Lights, and Multiplexers

The purpose of this exercise is to learn how to connect simple input and output devices to an FPGA chip and implement a circuit that uses these devices. We will use the switches SW_{15-0} on the NEXYs 4 DDR board as inputs to the circuit. We will use light emitting diodes (LEDs) as output device.

Part I

The NEXYs 4 DDR board provides 16 toggle switches, called SW_{15-0}, that can be used as inputs to a circuit, and 16 red lights, called LD_{15-0}, that can be used to display output values. Figure 1 shows a simple Verilog module that uses these switches and shows their states on the LEDs. Since there are 16 switches and lights it is convenient to represent them as vectors in the Verilog code, as shown. We have used a single assignment statement for all 16 *LD* outputs, which is equivalent to the individual assignments

```
assign LED[15] = SW[15];
assign LEDR[15] = SW[15];
…
assign LED[0] = SW[0];
```

The NEXYs 4 DDR board has hardwired connections between its FPGA chip and the switches and lights. To use SW_{15-0} and LD_{15-0} it is necessary to include in your vivado project the correct pin assignments, which are given in the NEXYs 4 DDR *User Manual*.

It is important to realize that the pin assignments in the NEXYs 4 DDR *pin assignments.csv* file are useful only if the pin names given in the file are exactly the same as the port names used in your Verilog module. The file uses the names *SW*[0] … *SW*[17] and *LED*[0] … *LED*[17] for the switches and lights, which is the reason we used these names in Figure 1.

```
// Simple module that connects the SW switches to the LEDR lights
module  part1 (SW, LED);
    input [15:0] SW;          // toggle switches
    output [15:0] LED;        // red LEDs

    assign LED = SW;
endmodule
```

Figure 1. Verilog code that uses NEXYs 4 DDR board switches and lights.

Perform the following steps to implement a circuit corresponding to the code in Figure 1 on the NEXYs 4 DDR board.

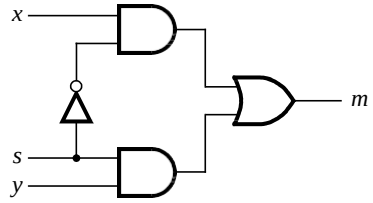
1. Create a new project for your circuit in Vivado. Select **XC7a100tcsg324-1** as the target chip, which is the FPGA chip on the NEXYs 4 DDR board.
2. Create a Verilog module for the code in Figure 1 and include it in your project.

3. Include in your project the required pin assignments for the NEXYs 4 DDR board, as discussed above. Compile the project.

4. Download the compiled circuit into the FPGA chip. Test the functionality of the circuit by toggling the switches and observing the LEDs.

Part II

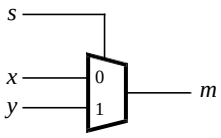
Figure 2*a* shows a sum-of-products circuit that implements a 2-to-1 *multiplexer* with a select input *s*. If $s = 0$ the multiplexer's output *m* is equal to the input *x*, and if $s = 1$ the output is equal to *y*. Part *b* of the figure gives a truth table for this multiplexer, and part *c* shows its circuit symbol.



a) Circuit

s	*m*
0	*x*
1	*y*

b) Truth table



c) Symbol

Figure 2. A 2-to-1 multiplexer.

The multiplexer can be described by the following Verilog statement:

```
assign m = (~s& x) | (s & y);
```

You are to write a Verilog module that includes eight assignment statements like the one shown above to describe the circuit given in Figure 3*a*. This circuit has two eight-bit inputs, X and Y, and produces the eight-bit output M. If $s = 0$ then $M = X$, while if $s = 1$ then $M = Y$. We refer to this circuit as an eight-bit wide 2-to-1 multiplexer. It has the circuit symbol shown in Figure 3*b*, in which X, Y, and M are depicted as eight-bit wires. Perform the steps shown below.

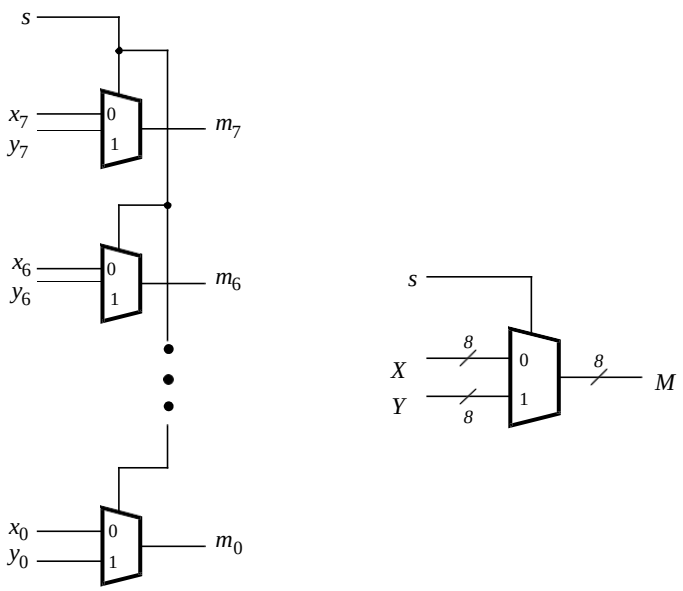


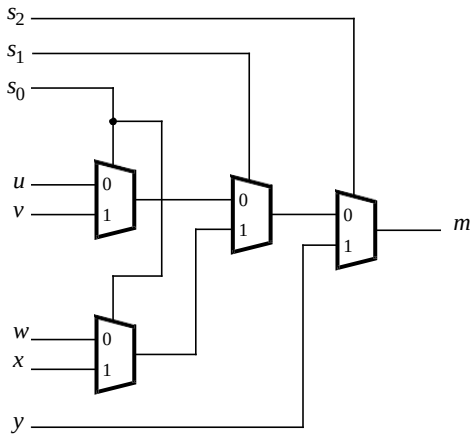
Figure 3. An eight-bit wide 2-to-1 multiplexer.

1. Create a new project for your circuit in vivado.
2. Include your Verilog file for the eight-bit wide 2-to-1 multiplexer in your project. Use switch SW_{15} on the NEXYs 4 DDR board as the s input, switches SW_{7-0} as the X input and SW_{15-8} as the Y input. Connect the *SW* switches to the red lights *LEDR and* connect the output M to the green lights $LEDG_{7-0}$.
3. Include in your project the required pin assignments for the NEXYs 4 DDR board. As discussed in Part I, these assignments ensure that the input ports of your Verilog code will use the pins on the FPGA that are connected to the *SW* switches, and the output ports of your Verilog code will use the FPGA pins connected to the *LEDR* and *LEDG* lights.
4. Compile the project.
5. Download the compiled circuit into the FPGA chip. Test the functionality of the eight-bit wide 2-to-1 multiplexer by toggling the switches and observing the LEDs.

Part III

In Figure 2 we showed a 2-to-1 multiplexer that selects between the two inputs *x* and *y*. For this part consider a circuit in which the output *m* has to be selected from five inputs u, v, w, x, and y. Part a of Figure 4 shows how we can build the required 5-to-1 multiplexer by using four 2-to-1 multiplexers. The circuit uses a 3-bit select input $s_2s_1s_0$ and implements the truth table shown in Figure 4b. A circuit symbol for this multiplexer is given in part c of the figure.

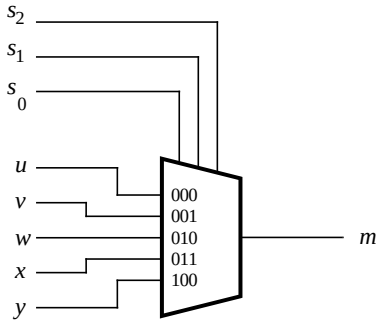
Recall from Figure 3 that an eight-bit wide 2-to-1 multiplexer can be built by using eight instances of a 2-to-1 multiplexer. Figure 5 applies this concept to define a three-bit wide 5-to-1 multiplexer. It contains three instances of the circuit in Figure 4a.



a) Circuit

s_2	s_1	s_0	m
0	0	0	u
0	0	1	v
0	1	0	w
0	1	1	x
1	0	0	y
1	0	1	y
1	1	0	y
1	1	1	y

b) Truth table



c) Symbol

Figure 4. A 5-to-1 multiplexer.

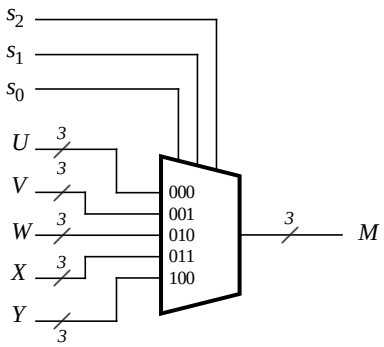


Figure 5. A three-bit wide 5-to-1 multiplexer.

Perform the following steps to implement the three-bit wide 5-to-1 multiplexer.

1. Create a new project for your circuit in vivado.
2. Create a Verilog module for the three-bit wide 5-to-1 multiplexer. Connect its select inputs to switches SW_{15-13}, and use the remaining 14 switches SW_{13-0} to provide the five 3-bit inputs U to Y. Connect the SW switches to the red lights $LEDR$ and connect the output M to the green lights $LEDG_{2-0}$.
3. Include in your project the required pin assignments for the NEXYs 4 DDR board. Compile the project.
4. Download the compiled circuit into the FPGA chip. Test the functionality of the three-bit wide 5-to-1 multiplexer by toggling the switches and observing the LEDs. Ensure that each of the inputs U to Y can be properly selected as the output M.

Part IV

A 4-to-2 priority encoder is a combinational circuit that takes 4 input signals and outputs a 2-bit binary code (Out_Y [1] and Out_Y[0]) corresponding to the highest-priority active input (In_A). If no inputs are active, the output defaults to zero, or a valid signal can be included to indicate the presence of valid input. The priority is assigned in descending order from In_A[3] (highest priority) to In_A[0] (lowest priority). This means:

- If In_A[3] is 1, it overrides all other inputs.
- If In_A[3] is 0 and In_A[2] is 1, it takes precedence over In_A[1] and In_A[0], and so on.

Write a Verilog code for 4-to-2 priority encoder using if/else statement.

In_A[3]	In_A[2]	In_A[1]	In_A[0]	Out_Y[1]	Out_Y[0]
1	X	X	X	1	1
0	1	X	X	1	0
0	0	1	X	0	1
0	0	0	1	0	0

Truth Table of 4_to_2 priority encoder

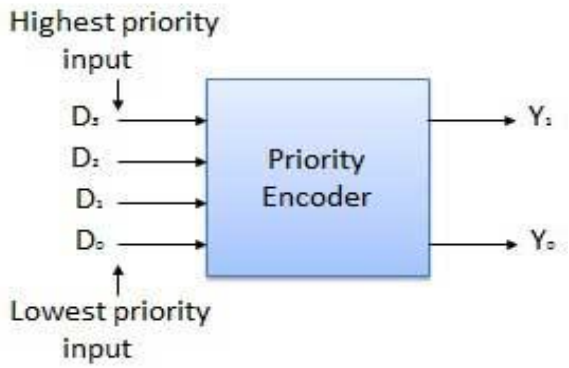


Figure 6. 4_to_2 priority encoder

Perform the following steps:

1. Create a new project for your circuit in vivado.

1. Create a Verilog module for the 4_to_2 priority encoder. Connect the In_A[3]…. In_A[0] inputs to switches SW_{3-0}, and connect the outputs of the encoder to the *leds* on the NEXYs 4 DDR board.

   output [1:0] Out_y;

   in your Verilog code so that the names of these outputs match the corresponding names in the NEXYs 4 DDR *User Manual* and the NEXYs 4 DDR *pin assignments.csv* file.

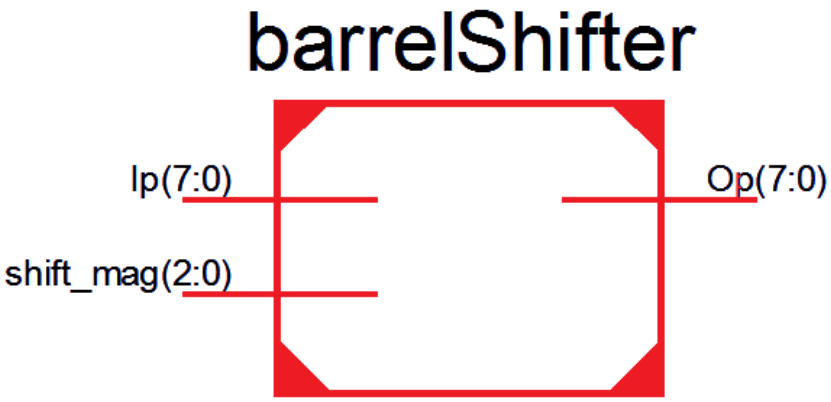
2. After making the required NEXYs 4 DDR board pin assignments, compile the project.

3. Download the compiled circuit into the FPGA chip. Test the functionality of the circuit by toggling the SW_{3-0} switches and observing the leds on the board.

Part V

A **barrel shifter** is a digital circuit that can **shift** a **data word** by a specified number of bits in **a one clock cycle**. It can be implemented as a **sequence of multiplexers** and in such an implementation the output of one **mux** is connected to the input of the next **mux** in a way that depends on the shift distance.

Implement an 8-bit Right-Shift Barrel Shifter in Verilog. The purpose of this task is to design and implement an 8-bit right-shift barrel shifter using a **case statement in Verilog**. The barrel shifter will take an 8-bit input data signal and shift it to the right based on a 3-bit control signal amt, which determines the number of positions to shift.



Perform the following steps:

1. Create a new project for your circuit in Vivado and select as the target chip the **XC7a100tcsg324-1.**

2. Include your Verilog module in the Vivado project. The 8-bit input data is connected to the switches (SW[7:0]) on the Nexys DDR4 FPGA board. Connect the 3-bit shift amount to (SW[8:10]) The shifted output is displayed using the LEDs.

3. Include the required pin assignments for the NEXYs 4 DDR board for all switches, and LEDs. Compile the project.

4. Download the compiled circuit into the FPGA chip. Test the functionality of the circuit by switching.