ECE327 Digital System Design
Lab 1: Design of basic gates and I/O

Lab Overview: 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 $SW_{9\text{-}0}$ and push-buttons $KEY_{3\text{-}0}$ on the DE1-SoC board as inputs to the circuit. We will use light emitting diodes (LEDs) and 7-segment displays as output devices. The requirements for this lab consist of completing the QUARTUS II designs, demoing the designs, and completing the laboratory report (including VHDL source files and simulation results). For more information regarding pins and I/O for the DE1-SoC board (in these and all labs), please refer to the user manual available online and on the blackboard site.

Background:

The DE1-series board provides 10 toggle switches, called $SW_{9\text{-}0}$, that can be used as inputs to a circuit, and 10 red lights, called $LEDR_{9\text{-}0}$, that can be used to display output values. Since there are 10 switches and lights, it is convenient to represent them as arrays (STD_LOGIC_VECTOR) in the VHDL code.

The DE1-series board has hardwired connections between its FPGA chip and the switches and lights. To use $SW_{9\text{-}0}$ and $LEDR_{9\text{-}0}$ it is necessary to include in your Quartus II project the correct pin assignments, which are given in the *DE1-series User Manual*. For example, on the DE1-SoC board, SW_0 is connected to the FPGA pin *AB12* and $LEDR_0$ is connected to pin *V16*. A good way to make the required pin assignments is to import into the Quartus II software the file called *DE1_SoC.qsf* for the DE1-SoC board, which is provided on blackboard and in the University Program section of Altera's web site. The procedure for making pin assignments is described in the tutorial Quartus II Introduction using VHDL Design, which is also available from Altera.

It is important to realize that the pin assignments in the .qsf file are useful only if the pin names given in the file are exactly the same as the port names used in your VHDL entity. The file uses the names *SW[0] : : : SW[9]* and *LEDR[0] : : : LEDR[9]* for the switches and lights (note that the Quartus II software uses [ ] square brackets for array elements, while the VHDL syntax uses ( ) round brackets).

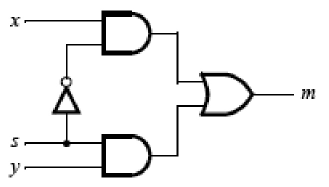
When creating your project in Quartus II, select the Cyclone V 5CSEMA5F31C6N device.

Part I

Generate a D flip-flop that latches the input value on the rising or falling edge of a clock pulse. Use *KEY[0]* to send a pulse to the flip-flop, use *SW[0]* as the input, and send the output value to *LEDR[0]*.

Part II

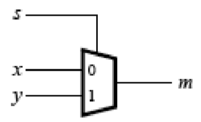
Figure 4a below shows a sum-of-products circuit that implements a 2-to-1 *multiplexer* with a select input s. If $s = 0$ the multiplexer's output m is equal to the input x, and if $s = 1$ the output is equal to y. Part b of the figure gives a truth table for this multiplexer, and part c shows its circuit symbol.



a) Circuit

s	m
0	x
1	y

b) Truth table



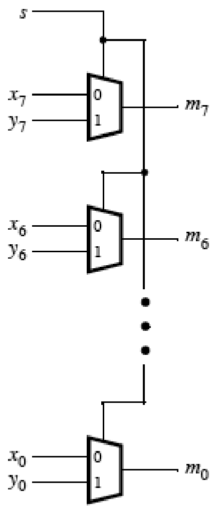
c) Symbol

Figure 4. A 2-to-1 multiplexer.

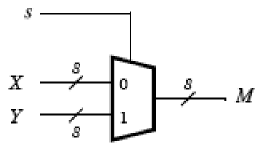
The multiplexer can be described by the following *Data Flow* VHDL statement:

m <= (NOT (s) AND x) OR (s AND y);

1. Write a VHDL entity that describes the circuit given in Figure 5a (an eight-bit wide 2-to-1 multiplexer). 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*. It has the circuit symbol shown in Figure 5b, in which *X*, *Y*, and *M* are depicted as eight-bit wires (vectors).
2. Simulate with a test bench.
3. Synthesize, Place & Route, download the bitstream for your design onto the DE1 board, and test the operation.



a) Circuit



b) Symbol

Figure 5. A eight-bit wide 2-to-1 multiplexer.

Part III

1. Modify your design to create a circuit where the output *m* is selected from five 3-bit inputs *u*, *v*, *w*, *x*, and *y*.
2. Simulate with a test bench.

3. Synthesize, Place & Route, download the bitstream for your design onto the DE1 board, and test the operation.

Part IV

Figure 8 shows a *7-segment decoder* module that has the three-bit input $c_2c_1c_0$. This decoder produces seven outputs that are used to display a character on a 7-segment display. Table 1 below lists the characters that should be displayed for each valuation of $c_2c_1c_0$ (plus the 'blank' character, which is selected for codes 101 - 111).

The seven segments in the display are identified by the indices 0 to 6 shown in Figure 8. Each segment is illuminated by driving it to the logic value 0.

1. Write a VHDL entity/architecture to activate each of the seven segments. Use only simple VHDL assignment statements in your code to specify each logic function. Connect the outputs of the decoder to the *HEX0* display on the DE2-series board. The segments in this display are called $HEX0_0$, $HEX0_1$, : : :, $HEX0_6$, corresponding to Figure 8. You should declare the 7-bit port:

   HEX0 : OUT STD_LOGIC_VECTOR(0 TO 6);

2. Simulate with a test bench.
3. Synthesize, Place & Route, download the bitstream for your design onto the DE2 board and test the operation.

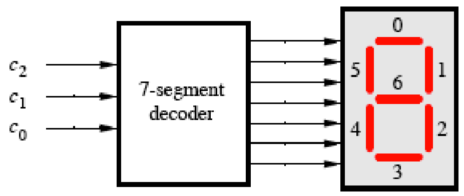


Figure 8. A 7-segment decoder.

Table 1.

$c_2c_1c_0$	Character
000	H
001	E
010	L
011	O
100	
101	
110	
111	

Part V

Consider the circuit shown in Figure 9. It uses a three-bit wide 5-to-1 multiplexer (Part III) to enable the selection of five characters that are displayed on a 7-segment display. Using the 7-segment decoder from Part IV this circuit can display any of the characters H, E, L, O and 'blank'. The character codes are set according to Table 1 by using the switches SW_{2-0}, are latched with KEY_{3-0}, and a specific character is selected for display by setting the switches SW_{9-7}.

1. Use circuits from Parts I, II and III as subcircuits in this code to drive five 7-segment displays rather than just one. The purpose of your circuit is to display any word on the five displays that is composed of the

characters in Table 1, and be able to rotate this word in a circular fashion across the displays when the switches $SW_{9\text{-}7}$ are toggled. As an example, if the displayed word is “HELLO”, then your circuit should produce the output patterns illustrated in Table 2. You should be able to modify this circuit to display “TIGER”

2. Simulate with a test bench.
3. Synthesize, Place & Route, download the bitstream for your design onto the DE1 board, and test the operation.

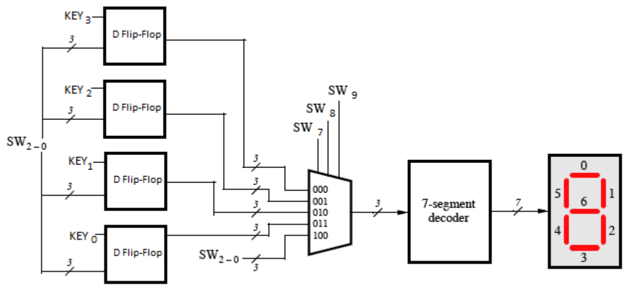


Figure 9. A circuit that can select and display one of five characters.

Table 2.

SW₉SW₈SW₇	**Character Pattern**				
000	H	E	L	L	O
001	O	H	E	L	L
010	L	O	H	E	L
011	L	L	O	H	E
100	E	L	L	O	H

Rubric	
Report - Proper format - All sections included - Valid images where applicable - Proper grammar, punctuation, and spelling	**50%** - 10% - 30% - 5% - 5%
Demo - Live Demonstration o Includes working code and answering questions from the TA - Comments o Thoughtful comments, not English translations of code	**40%** - 35% - 5%
Proper Assign Server Code Submission	**10%**