"FPGA-Programming" Exercise Sheet I:

Switches, LEDs 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 $SW[9] - SW[0]$ on the DE1-SoC board as inputs to the circuit. We will use the LEDs and 7-segment displays as output devices.

As target hardware always choose FPGA chip Cyclone V SoC 5CSEMA5F31C6

1) *Simple switch to LED assignment*

The DE1-SoC board features following switches and LEDs:

Number of Switches	Name of Switches	Number of LEDs	Name of LEDs
10	$SW[9]$ - $SW[0]$	10	$LEDR[9]$ - $LEDR[0]$

Write a simple VHDL program which assigns the 10 switches to the 10 red LEDs, in a way that switch $SW[0]$ is assigned to $LEDR[0]$, switch $SW[1]$ to $LEDR[1]$ and so on. Perform following steps:

i) Generate a new Quartus project for the circuit, named **e_my_first_fpga**. Create a new VHDL file, named **e_my_first_fpga.vhd**.

ii) Create a VHDL entity and describe the architecture of the circuit, named **a_my_first_fpga_1**.

iii) Perform an analysis and synthesis step by heading to *Processing -> Start -> Start Analysis & Synthesis* or by pressing the appropriate button.

iv) Goto *Assignments -> Pin Planner* or press the appropriate button. In the Pin Planner assign the necessary pins. Information to perform the correct assignments can be found in the document DE1-SoC_User_manual.pdf provided on iLearn.

v) Compile the circuit (*Processing -> Start Compilation*). Analyze it with Quartus **RTL Viewer** (*Tools -> Netlist Viewers -> RTL Viewer*), and **Chip Planner** (*Tools -> Chip Planner*) in order to become familiar with these tools.

vi) Download the compiled circuit (*Tools -> Programmer or appropriate button press*) and test the functionality. Information about how to program the device can be found in the document DE1-SoC_User_manual.pdf under section *Configure the FPGA in JTAG Mode* starting at page 14.

vii) Change the functionality of the circuit such that the switches will be assigned to the LEDs in reversed order, so $SW[9]$ to $LEDR[0]$, $SW[8]$ to $LEDR[1]$, and so on. Use a second architecture description named **a_my_first_fpga_2** and the VHDL syntax of a **configuration** block named **c_my_first_fpga**. Compile the circuit, analyze it, download it and test the functionality. Also try using internal signals.

2) *Four bit wide* 2*-to-*1 *multiplexer*

Following figure shows a simple circuit, which implements a 2-to-1 multiplexer with address input s. If $s = 0$ than the output m will take the value of input x, otherwise the output m will take the value of input y.

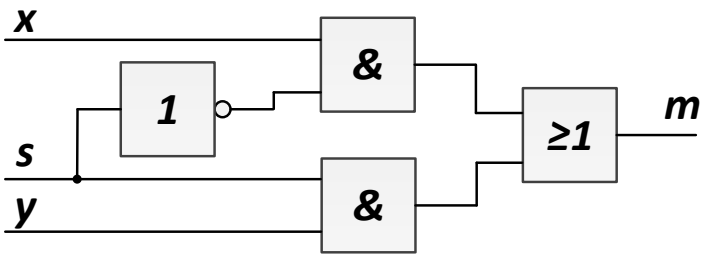

Figure 2.1: Circuit of a 2-to-1 multiplexer

Based on this circuit, following abstract truth table results:

s	m
0	x
1	y

Following figure illustrates the symbol of the circuit:

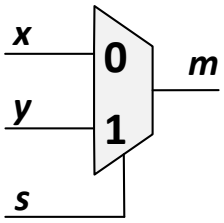

Figure 2.2: Symbol of a 2-to-1 multiplexer

i) Derive the complete truth table for the given circuit and describe the logic function for output m.

In the following a VHDL entity should be generated, which describes the circuit illustrated in the figure below.

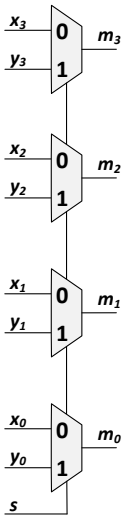

Figure 2.3: Circuit of a 4-bit wide 2-to-1 multiplexer

The circuit has two 4-bit wide inputs X and Y and produces a 4-bit wide output M. If the address input $s = 0$ than $M = X$, otherwise $M = Y$. The symbol of the 4-bit wide 2-to1 multiplexer is given below:

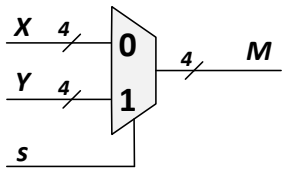

Figure 2.4: Symbol of a 4-bit wide 2-to-1 multiplexer

Perform following steps:

ii) Generate a new Quartus project for the circuit, named **e_my_4bit2to1mux**.

iii) Create a VHDL entity and describe the circuit. Use switch $SW[9]$ as address input s, switches $SW[3-0]$ as input vector X, switches $SW[7-4]$ as input vector Y. Furthermore the value of address input s should be displayed on $LEDR[9]$, the output vector M should be displayed on $LEDR[3-0]$. Unused LEDs should be assigned with a constant value of 0.

iv) Import the necessary pin assignments.

v) Compile the project, download the compiled circuit and test the functionality.

Furthermore inspect the circuit with **RTL Viewer** and **Chip Planner**.

3) *Two bit wide* 3*-to-*1 *multiplexer*

One part of previous task was the description of a 2-to-1 multiplexer. Following figure shows, how this circuit can be used for the description of a 3-to-1 multiplexer. A 3-to-1 multiplexer has three data inputs u, v and w, which means that a two bit wide address signal $s1$ $s0$ is needed.

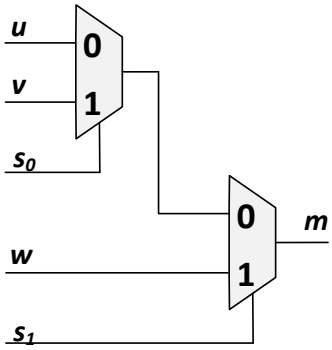

Figure 3.1: Circuit of a 3-to-1 multiplexer

Based on this circuit, following abstract truth table results:

s1	s0	m
0	0	u
0	1	v
1	0	w
1	1	w

Following figure illustrates the symbol of the circuit:

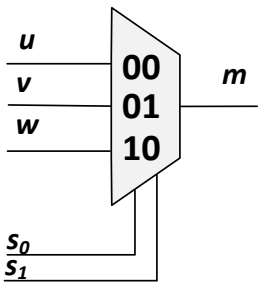

Figure 3.2: Symbol of a 3-to-1 multiplexer

i) Derive the complete truth table for the given circuit and describe the logic function for output m.

In the following a VHDL entity should be generated, which describes the circuit illustrated in the figure below.

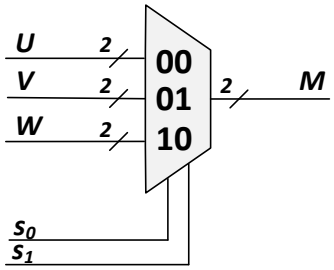

Figure 3.3: Symbol of a 2-bit wide 3-to-1 multiplexer

The circuit has three 2-bit wide inputs U, V and W and produces a 2-bit wide output M. Perform follwing steps:

ii) Generate a new Quartus project for the circuit, named **e_my_2bit3to1mux**.

iii) Create a VHDL entity and describe the circuit. Use switches $SW[9-8]$ as address input s, switches $SW[5-0]$ for the input vectors of U, V and W. The output vector M should be displayed on $LEDR[1-0]$. Unused LEDs should be assigned with a constant value of 0.

iv) Import the necessary pin assignments.

v) Compile the project, download the compiled circuit and test the functionality.

Furthermore inspect the circuit with **RTL Viewer** and **Chip Planner**.

4) *Simple 7-segment decoder*

Goal of this task is the description of a simple 7-segment decoder. The character to display depends on a 2-bit input vector. The decoder produces 7 output signals, which are used to shows a character on a 7-segment display. Following figure clarifies the circuit to be described with the inputs $c1$ $c0$.

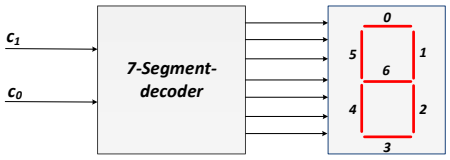

Figure 4.1: Simple 2-input 7-segment decoder

The table below states the characters, which should be displayed for different choices of $c1$ and $c0$, including a *blank* character:

$c1$	$c0$	character
0	0	d
0	1	E
1	0	1
1	1	

i) Derive the truth table for each of the individual segment of the 7-segment display and describe the logic function for each segment.

The seven segments of the display will be identified by the indices 0 up to 6. The individual segments are **low-active**, which means that a segment is illuminated if it will be driven by a logic 0. In the following a VHDL entity should be described, which implements the derived logic functions. Only use simple VHDL assignments based on Boolean expressions here. Perform following steps:

ii) Generate a new Quartus project for the circuit named **e_my_7segdec**.

iii) Create a VHDL entity for the 7-segment decoder. Connect the inputs $c1$ $c0$ with the switches $SW[1-0]$, and the outputs of the decoder with the 7-segment display $HEX0$. The individual segments of the display are named $HEX0[0]$ to $HEX0[6]$. Use following port description:

```
HEX0: out std_logic_vector(0 to 6);
```

iv) Import the necessary pin assignments.

v) Compile the project, download the compiled circuit and test the functionality.

Furthermore inspect the circuit with **RTL Viewer** and **Chip Planner**.

5) *Multiplexers and 7-segment displays*

Consider the circuit of following figure:

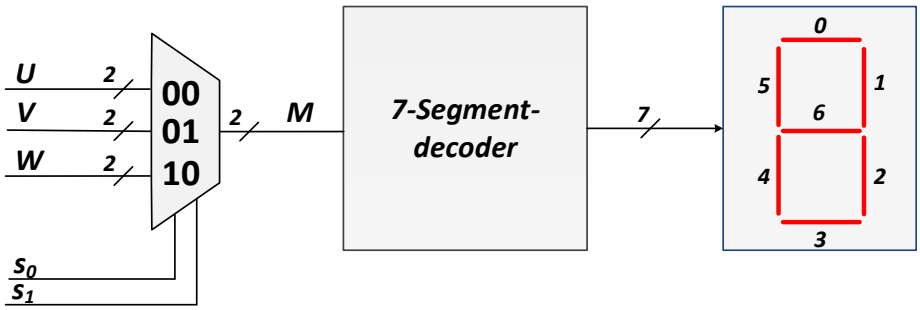

Figure 5.1: Simple controlled 7-segment decoder

The circuit uses a 2-bit wide 3-to-1 multiplexer in order to enable the selection of three characters, which should be displayed on the 7-segment display. The 7-segment decoder of part 4) permits displaying the characters d, E, 1 as well as the *blank* character.

The cipher codes should be set with switches $SW[9-8]$ as given in the table below. This simple circuit can be quickly generated using the descriptions of part 3) and part 4).

$SW[9]$	$SW[8]$	character
0	0	d
0	1	E
1	0	1

Goal of this task is to describe a circuit, which enables to drive a total of three 7-segment displays instead only one. It should be utilized to *rotate* the word $dE1$ in a circular fashion, as given in the table below:

$SW[9]$	$SW[8]$	$HEX2$	$HEX1$	$HEX0$
0	0	d	E	1
0	1	E	1	d
1	0	1	d	E

For the realization, three instances of the sub-level entities (**e_2bit3to1mux** and **e_7segdec**) will be needed, which have to be declared as **component** in the top-level entity and instantiated. Perform the following steps:

i) Generate a new Quartus project for the circuit, named **e_my_mux7seg**.

ii) Create a VHDL top-level entity for the circuit. Connect the switches $SW[9-8]$ with the address inputs of the three instances of the 2-bit wide 3-to-1 multiplexer. Furthermore connect the switches $SW[5-0]$ with the inputs of the three 7-segment decoder in a way to achieve the desired functionality. Show the position of the switches on the LEDs and connect the outputs of the three instances of the 7-segment decoder with the 7-segment displays $HEX2$, $HEX1$ and $HEX0$.

iii) Import the necessary pin assignments.

iv) Compile the project, download the compiled circuit and test the functionality.

Furthermore inspect the circuit with **RTL Viewer** and **Chip Planner**.

6) *Rotating letters*

Goal of this task is to extend the circuit of part 5) in a way that all six 7-segment displays will be driven. The circuit must be capable of displaying the word $dE1$ on three 7-segment displays while the remaining show the *blank* character. The letters of the word will be determined by three 2-bit wide inputs as in part 5). Furthermore the possibility to rotate the word from right-to-left should be implemented, as given in the table below.

$SW[9]$	$SW[8]$	$SW[7]$	$HEX5$	$HEX4$	$HEX3$	$HEX2$	$HEX1$	$HEX0$
0	0	0				d	E	1
0	0	1			d	E	1	
0	1	0		d	E	1		
0	1	1	d	E	1			
1	0	0	E	1				d
1	0	1	1				d	E

For the realization a VHDL entity named **e_2bit6to1mux** must be written, which describes a 2-bit wide 6-to-1 multiplexer.

i) Determine the truth table for this *6-to-1* multiplexer and derive the logic functions.

In total six instances of **e_2bit6to1mux** and **e_7segdec** will be needed, which have to be declared as **component** in the top-level entity and instantiated. Perform the following steps:

ii) Generate a new Quartus project for the circuit named **e_my_rotword**.

iii) Create a VHDL top-level entity for the circuit. Connect the switches $SW[9-7]$ with the address inputs of the six instances of the 2-bit wide 6-to-1 multiplexer. Furthermore connect the switches $SW[5-0]$ with the inputs of the six 7-segment decoder in a way to achieve the desired functionality. Show the position of the switches on the LEDs and connect the outputs of the six instances of the 7-segment decoder with the 7-segment displays $HEX5$ to $HEX0$.

iv) Import the necessary pin assignments.

v) Compile the project, download the compiled circuit and test the functionality.

Furthermore inspect the circuit with **RTL Viewer** and **Chip Planner**.