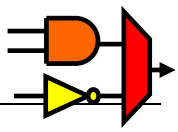



POLITECNICO
DI TORINO

# – Elettronica dei Sistemi Digitali –
# Lab#2

# Switches, Decoders, Numbers and Displays

The purpose of this laboratory session is to start learning how to connect previously existing units and other ad-hoc logic circuits. We will use the switches $SW_{17-0}$ on the DE2 board, but we will try to control a 7-segments display with a specific decoder. In this lab you cannot use any conditional construct like *CASE*, *SELECT*, *WHEN* and *IF*. **Please note that some parts are optional and will be evaluated separately.**

**Contents:**

1. Controlling a 7-segments display
2. Multiplexing the 7-segments display output
3. Binary to decimal converter
4. Binary to BCD converter (optional)

**Abbreviations and acronyms:**

IC – Integrated Circuit
LED – Light Emitting Diode
MUX – Multiplexer
VHDL – Very high speed integrated circuits Hardware Description Language
[VHDL cookbook: http://www.onlinefreeebooks.net/engineering-ebooks/electrical-engineering/the-vhdl-cookbook-pdf.html]

***USE EXACTLY THE SAME I/O PINS SPECIFIED IN THIS DOCUMENT.***
***YOU ARE NOT ALLOWED TO USE "IF", "SELECT", "WHEN" OR "CASE" STATEMENTS IN ALL THE EXERCISES.***

## 1 - Controlling a 7-segments display

Figure 1 shows a *7-segment* decoder module with the three input bits $c_2\,c_1\,c_0$. This decoder drives seven outputs that are used to display an alphanumeric character on a 7-segment display. Table 1 lists the characters that should be displayed for each value of $c_2\,c_1\,c_0$. In order to keep the design simple, only four characters are included in the table (plus the '`blank`' character, which is selected by the four codes $100 - 111$).

The seven segments in the display are identified by the indexes 0 to 6 shown in figure. Each segment is lit when driven to the logic value '0'. You need to write a VHDL entity that implements the logic functions needed to activate each of the seven segments. Use only simple VHDL assignments and write simple Boolean expressions in your code to specify each logic function.

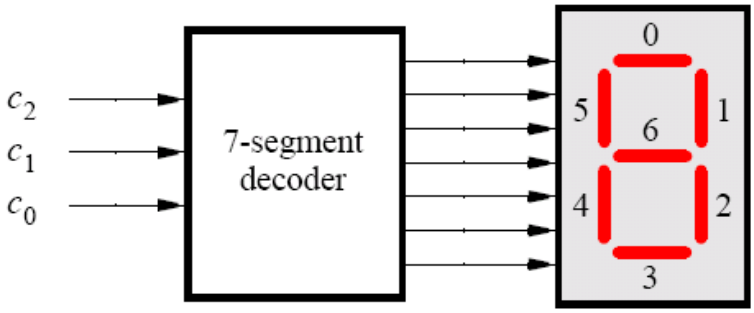



Figure 1 - A 7-segment decoder.

| $c_2 c_1 c_0$ | Character |
|---|---|
| 000 | H |
| 001 | E |
| 010 | L |
| 011 | O |
| 100 | |
| 101 | |
| 110 | |
| 111 | |

Table 1 - Character codes.

In particular, you need to do the following steps:

1. **Create a new Quartus II project** for your circuit.
2. **Create a VHDL entity** for the 7-segment decoder. Connect the $c_2$ $c_1$ $c_0$ inputs to switches $SW_{2-0}$, and connect the outputs of the decoder to the *HEX0* display on the DE2 board. The segments in this display are called $HEX0_0$, $HEX0_1$, . . ., $HEX0_6$, corresponding to Figure 1. Remember to declare the 7-bit port

   ```
   HEX0 : OUT STD_LOGIC_VECTOR(0 TO 6);
   ```

   in your VHDL code in such a way that the names of the outputs match the corresponding names in the ***de2_pin_assignments.csv*** file.
3. After adding the required DE2 board **pin assignments**, compile the project.
4. **Download** the compiled circuit in the FPGA chip.
5. **Test** the functionality of the circuit by toggling the $SW_{2-0}$ switches and by looking at the 7-segment display.

***1. YOU NEED TO WRITE THE VHDL CODE OF THE "HELLO" DECODER.***
***2. YOU NEED TO TEST IT ON THE FPGA.***

# 2 – Multiplexing the 7-segments display output

Consider the circuit shown in Figure 2. It uses a three-bit wide 5-to-1 multiplexer to enable the selection of five characters that are displayed on a 7-segment display. By using the 7-segment decoder from section 1 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 $SW_{14-0}$, and a specific character is selected for display by setting the switches $SW_{17-15}$.
An outline of the VHDL code of the circuit is provided in Figure 3. You can use whatever name for the VHDL entities but use different *vhd* files for each different module (a file for each Entity-Architecture couple). Note that here we are re-using the circuits from previous Labs as sub-circuits of this more complex logic.

You need to extend the code in Figure 3 in such a way that it uses 5 7-segment displays rather than just one. You will need to use 5 instances of each of the sub-circuits. The purpose of your circuit is to display any word on the 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 $SW_{17-15}$ are toggled. For example, if the displayed word is HELLO, then your circuit should drive the output patterns as illustrated in Table 2.

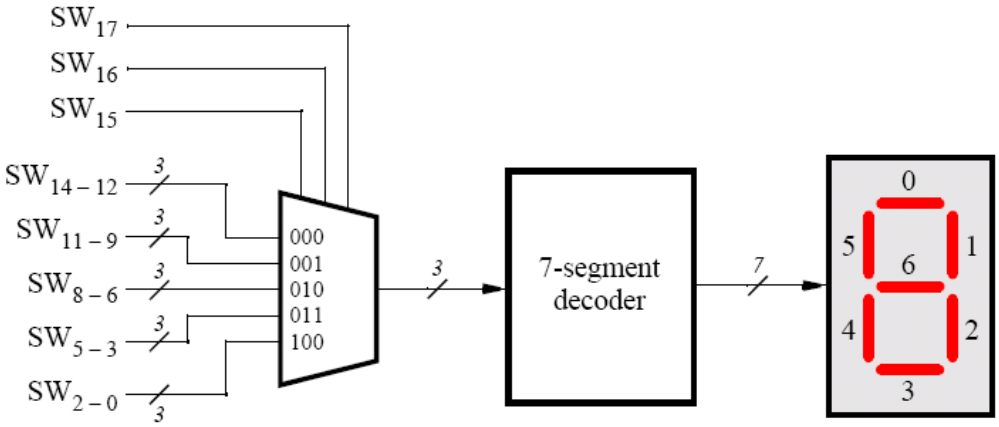



Figure 2 – 7-Segments output letter selection.

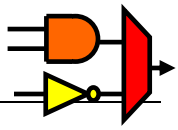
```
-- File 7segments_out.vhd

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY part2 IS
      PORT ( SW     : IN       STD_LOGIC_VECTOR(17 DOWNTO 0);
             HEX0   : OUT      STD_LOGIC_VECTOR(0 TO 6));
END part2;

ARCHITECTURE Behavior OF part2 IS

 COMPONENT mux_3bit_5to1
            PORT ( S, U,V, W, X,Y    : IN  STD_LOGIC_VECTOR(2 DOWNTO 0);
                   M                 : OUT STD_LOGIC_VECTOR(2 DOWNTO 0));
 END COMPONENT;
 COMPONENT char_7seg
            PORT ( C                 : IN  STD_LOGIC_VECTOR(2 DOWNTO 0);
                   Display           : OUT       STD_LOGIC_VECTOR(0 TO 6));
 END COMPONENT;
 SIGNAL M : STD_LOGIC_VECTOR(2 DOWNTO 0);
```



```
BEGIN
  M0: mux_3bit_5to1 PORT MAP (SW(17 DOWNTO 15), SW(14 DOWNTO 12), SW(11 DOWNTO 9),
                              SW(8 DOWNTO 6), SW(5 DOWNTO 3), SW(2 DOWNTO 0), M);
  H0: char_7seg PORT MAP (M, HEX0);
END Behavior;

-- File mux_3bit_5to1.vhd

LIBRARY ieee;
USE ieee.std_logic_1164.all;
- - implements a 3-bit wide 5-to-1 multiplexer

ENTITY mux_3bit_5to1 IS
      PORT ( S, U, V, W, X, Y       : IN  STD_LOGIC_VECTOR(2 DOWNTO 0);
             M                      : OUT STD_LOGIC_VECTOR(2 DOWNTO 0));
END mux_3bit_5to1;

ARCHITECTURE Behavior OF mux_3bit_5to1 IS
. . . code not shown
END Behavior;

-- File char 7seg.vhd

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY char_7seg IS
      PORT ( C          : IN  STD_LOGIC_VECTOR(2 DOWNTO 0);
             Display    : OUT STD_LOGIC_VECTOR(0 TO 6));
END char 7seg;

ARCHITECTURE Behavior OF char_7seg IS
. . . code not shown
END Behavior;
```

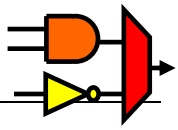
Figure 3 - VHDL code of the circuit in Figure 2.

| $SW_{17}$ $SW_{16}$ $SW_{15}$ | Character pattern | | | | |
|---|---|---|---|---|---|
| 000 | H | E | L | L | O |
| 001 | E | L | L | O | H |
| 010 | L | L | O | H | E |
| 011 | L | O | H | E | L |
| 100 | O | H | E | L | L |

Table 2 - Rotating the word HELLO on five 7-segments displays.

You need to do the following steps.

1. **Create a new Quartus II** project for your circuit.



2. **Include your VHDL top-level entity** in the Quartus II project as well as the other VHDL files. Connect the switches $SW_{17-15}$ to the select inputs of each of the five instances of the three-bit wide 5-to-1 multiplexers. Also connect $SW_{14-0}$ to each instance of the multiplexers as required to produce the patterns of characters shown in Table 2. Connect the outputs of the five multiplexers to the 7-segment displays *HEX4*, *HEX3*, *HEX2*, *HEX1*, and *HEX0*.
3. Include the required **pin assignments** for the DE2 board for all switches, LEDs, and 7-segment displays. Compile the project.
4. **Download** the compiled circuit in the FPGA chip.
5. **Test** the functionality of the circuit by setting the proper character codes on the switches $SW_{14-0}$ and then toggling $SW_{17-15}$ to observe the rotation of the characters.

***1. YOU NEED TO WRITE THE VHDL CODE OF THE ROTATING "HELLO" UNIT. HINT: TO SOLVE THE PROBLEM JUST TRY TO GUESS THE EXACT CONNECTIONS BETWEEN THE SWITCHES AND THE 5 MULTIPLEXERS. FOR THE MUXES YOU CAN USE THE CODE OF THE PREVIOUS LAB.***
***2. YOU NEED TO TEST IT ON THE FPGA.***

# 3 – Binary to Decimal converter

You need to design a circuit that converts a four-bit binary number $V = v_3\, v_2\, v_1\, v_0$ into its equivalent two-digit decimal number $D = d_1 d_0$. Table 3 shows the required output values. A partial design of this circuit is given in Figure 4. It includes a comparator that checks when the value of $V$ is bigger than 9, and uses its output to control the 7-segment displays. You need to complete the design of this circuit by creating a VHDL entity, which includes the comparator, some multiplexers and circuit *A* (do not include circuit *B* or the 7-segment decoder at this point). Your VHDL entity should have the four-bit input *v*, the four-bit output *m* and the output *z* (see Figure 5).

| Binary value | Decimal digits | |
|---|---|---|
| 0000 | 0 | 0 |
| 0001 | 0 | 1 |
| 0010 | 0 | 2 |
| ... | ... | ... |
| 1001 | 0 | 9 |
| 1010 | 1 | 0 |
| 1011 | 1 | 1 |
| 1100 | 1 | 2 |
| 1101 | 1 | 3 |
| 1110 | 1 | 4 |
| 1111 | 1 | 5 |

Table 3 - Binary-to-Decimal conversion.

Do the following steps:

1. **Make a Quartus II project** for your VHDL entity.
2. **Compile the circuit** and use functional simulation to verify the correct operation of your comparator, the multiplexers and circuit *A*.
3. **Comment your VHDL code** and **include circuit *B*** in Figure 4 **as well as the 7-segment decoder**. Here the idea is to have a separate entity for the BCD converter and to reuse the previous VHDL exercises for the upper level architecture. Change the inputs and outputs of your code in such a way that switches $SW_{3-0}$ on the DE2 board are used to represent the binary number *V*. Moreover, use the displays ***HEX1*** and ***HEX0*** to show the values of the decimal digits $d_1$ and $d_0$. Make sure you include the required pin assignments for the DE2 board in the project.

4. **Recompile the project**, and then download the circuit in the FPGA chip.
5. **Test your circuit** by setting all possible values of *V* and by looking at the 7-segments displays.

***1. YOU NEED TO WRITE THE VHDL CODE OF THE COMPLETE CIRCUIT IN FIGURE 4, CIRCUIT A, CIRCUIT B, MUXES… YOU CAN REUSE PART OF THE CODE OF THE PREVIOUS PARTS.***
***2. TEST THE CIRCUIT ON THE FPGA.***

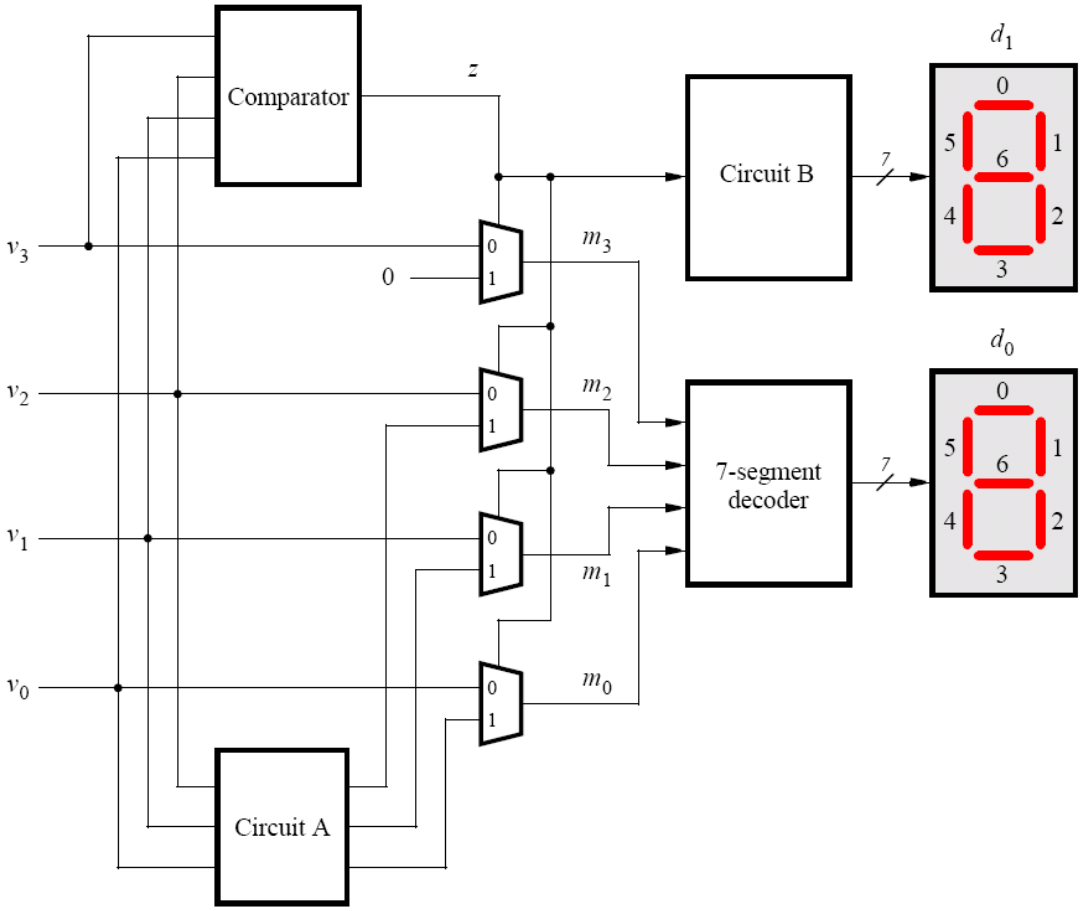


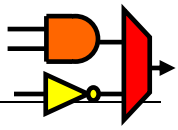

Figure 4 - Partial block scheme of the system that implements the binary-to-decimal conversion circuit.

Follows a VHDL template for the converter

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY converter IS
      PORT ( v     : IN      UNSIGNED(3 DOWNTO 0);
             m     : OUT     STD_LOGIC_VECTOR(3 DOWNTO 0);
             z     : OUT     STD_LOGIC);
END converter;
```

Figure 5 - VHDL template for the circuit in Figure 4.



# 4 – Binary-to-BCD Converter (optional)

In section 3 we discussed how it is possible to represent a binary number in a decimal convention by using 7-segments displays. Sometimes this conversion is mandatory because humans are used to refer to decimal representations instead of binary ones. As you can guess, in these circuits each decimal digit is represented by using four bits. This scheme is known as the *Binary Coded Decimal* (BCD) representation. As an example, the decimal value 59 is encoded in BCD form as 0101 1001.

Design a combinational circuit that converts a 6-bit binary number into a 2-digit decimal number represented in the BCD system. Use switches ***SW*$_{5-0}$** to input the binary number and the 7-segment displays ***HEX1*** and ***HEX0*** to display the decimal number. Implement your circuit on the DE2 board and demonstrate its functionality.

1. ***WRITE THE VHDL CODE OF THE CONVERTER. HINT: IF YOU WANT, YOU CAN USE THE `TO_UNSIGNED, TO_SIGNED` & `TO_INTEGER` FUNCTIONS. WHAT DIVISIONS SHOULD YOU DO TO CONVERT A BINARY TO A BCD? THINK ABOUT WHAT TO DO FOR A BASE 2 REPRESENTATION OF A DECIMAL NUMBER...***
2. ***YOU NEED TO TEST THE CIRCUIT ON THE FPGA.***