Department of Electronics and Telecommunications
Bachelor degree in Electronic Engineering

Digital Systems Electronics

LAB 2: Switches, Decoders, Numbers and Displays.

Due date: April 02, 2024
Delivery date: March ??, 2024

Group: 19

Contributions:

- Valtorta Alexander James
- Tedesco Angelo
- Urgu Sara

The members of the group listed above declare under their own responsibility that no part of this document has been copied from other documents and that the associated code is original and has been developed expressly for the assigned project.

Contents

1 Introduction

1.1 Aim of this project

The purpose of this laboratory is to learn how to connect previously existing units and other logic circuits, using the switches SW9-0 on the DE1-SoC board and control a 7-segment display with a specific decoder in order to multiplex the 7-segment display output and implement a binary to decimal converter and a binary-to-BCD converter.

1.2 Content and structure of the document

The report is divided into four sections, each for every exercise of the laboratory assignment. Moreover, each section contains the key parts in order to design and implement each circuit:
- Design Entry, which describes the circuit to be implemented on the FPGA device, in particular it is indicated the overall architecture of the circuit, by reporting the name of the source file or files and the key role of each of them.
- Functional Simulation, which describes the testbench and the approach used to assess the completed design.
- Synthesis, which shows the results of Modelsim functional simulation and that of the timing simulation.

2 Controlling a 7-segment display

2.1 Design entry

The 7-segment decoder drives seven output signals that control a 7-segment display, thus, in order to implement the logic functions needed to activate each of the seven-segment displays, according to the mapping of the table provided in the assignment, a truth table is required.
In particular, since each segment is lit when driven to the logic value '0', given all the possible input values for *c2 c1 c0*, the output value for each segment needs to be set to '0' or '1' depending on whether the segment has to be lit or not, respectively, so that the letter corresponding to the input is formed.

$c_2\ c_1\ c_0$	H_6	H_5	H_4	H_3	H_2	H_1	H_0	
000	0	0	0	1	0	0	1	H
001	0	0	0	0	1	1	0	E
010	1	0	0	0	1	1	1	L
011	1	0	0	0	1	1	1	L
100	1	0	0	0	0	0	0	O
others	1	1	1	1	1	1	1	blank

Once the truth table is formed, a Karnaugh map for each segment is necessary, so that it is possible to obtain the Boolean expression to specify each logic function.

$H_6 = c_2 + c_1$ $\quad H_5 = c_2c_0 + c_2c_1$ $\quad H_4 = c_2c_0 + c_2c_1$ $\quad H_3 = \overline{(c_2c_1c_0)} + c_2c_0 + c_2c_1$

$H_2 = c_1 + c_0$ $\quad H_1 = c_1 + c_0$ $\quad H_0 = \overline{(c_2c_0)} + c_2c_0 + c_1$

Thus, once declared the input and output values as 3-bit and 7-bit arrays respectively, by means of a dataflow structure in the architecture it is possible to assign to each output the logic function. File

d7seg_decoder.vhd

Lastly, it is necessary to map the inputs and outputs values previously defined in *d7seg_decoder.vhd* to those assigned to the DE1 board, in particular connect the *c2 c1 c0* inputs to switches SW2-0, and the outputs of the decoder to the HEX0 display. *d7seg_decoder_map.vhd* In order to do so, it is only a matter of defining the decoder design as a component in the structured architecture and then simply associating the signals of the DE1 (declared in the entity) to the inputs and outputs of the decoder.

2.2 Functional simulation

With the testbench [*tb_7seg_decoder.vhd*] it is possible to check the functionalities of the completed design. In particular after the typical empty entity the behavioural architecture need to contain in the declarative part the component, defined in *d7seg_decoder.vhd* and the internal signals for the simulation (*C_t* and *H_t*). Then, in the descriptive part, once the internal signals are properly mapped to the components' ones, the simulation is performed by associating each possible input value to the input signal waiting a short interval of time in between.

2.3 Synthesis

Once the circuit is successfully synthesized by Quartus it is possible to run a timing simulation (or post-synthesis simulation), as well as the Modelsim functional simulation. Generally it provides the circuit's actual performances in terms of speed and propagation delay of clock frequency; however for this design it was possible to observe little to no delay in the simulation.

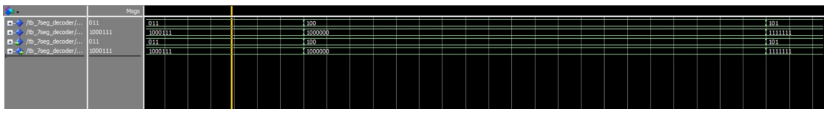


Figure 1: Modelsim functional simulation

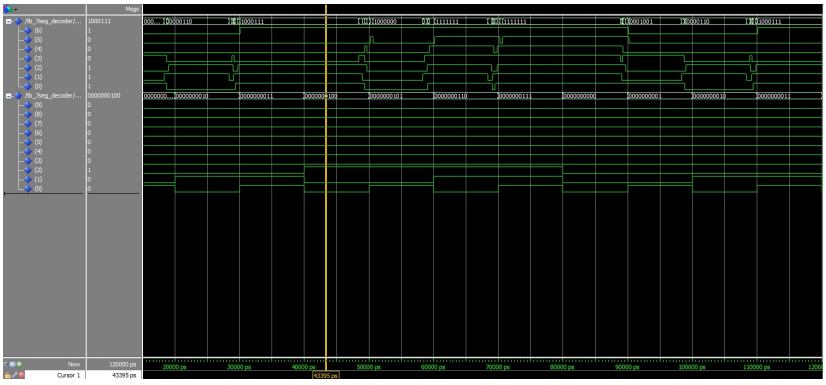


Figure 2: Timing simulation

3 Multiplexing the 7-segment display output

3.1 Design entry

Following the outline provided in the assignment it is possible to define a file [*d7seg_out.vhd*] that manages all the components and interconnects them in order to obtain the desired output, thus displaying the four words and properly shifting their letters.

The file was obtained following thoroughly the structure provided in the assignment, but choosing a structural approach of the architecture instead of the behavioural one.

Each component is defined in a dedicated vhdl file, following thoroughly the structure provided in the assignment:
- The multiplexer, whose output correspond to the input of the shifter, is defined in the *mux_5_1_15bit.vhd* file. It consists of a process that consider all different cases of the input *sel* and for each it assigns to the output *output* the 15 bits to create the word, with every 3 bit being associated to a letter, according to the following arrangement:

000 : H
001 : E
010 : L
011 : O
100 : C
101 : P
110 : F
111 : blank
so that, for example, the word *HELLO* can be obtained with 000001010010011

- The combinational shifter, whose output corresponds to the input of the 7-segment display, is defined in *shifter.vhd*. The file takes as input the 15-bit output of the multiplexer in order to shift the bits and show the letters in a different order: since each letter is represented by three bits, the shifter shifts three bits at a time to the left. Thus, in the descriptive part of the architecture for every input related to the commutation of the switches (*sel*), the output *output* is associated to the input from the multiplexer (*input*), according to the letters that need to shift. For example:
To display the word without shifting:

```
when "000" => output<=input;
```

To start the word with the "last three letters" (the last twelve bits, from 11 to 0) and shift the "first letter" (the first three bits, from 14 to 12) to the last position:

```
when "001" => output<=(input(11 downto 0)&input(14 downto 12));
```

to manage the errors: when the sequence of bits of *sel* does no correspond to any shifting command all segments of the display will be blank:

```
when others => output<="111111111111111";
```

- The decoder for 7-segment display is defined in the *decoder7.vhd*. It consists of a process that consider all different cases of the input and for each of them it assigns to the output *Display* the 7 bits to create the letter, with every bit being associated to the corresponding segment of the display, careful that each segment is lit when driven to the logic value '0'. Thus:

```
when "000" => Display <= "1001000";
when "001" => Display <= "0110000";
when "010" => Display <= "1110001";
when "011" => Display <= "0000001";
when "100" => Display <= "0110001";
when "101" => Display <= "0011000";
when "110" => Display <= "0111000";
when others => Display <= "1111111";
```

to obtain H,E,L,O,C,P,F and blank respectively.

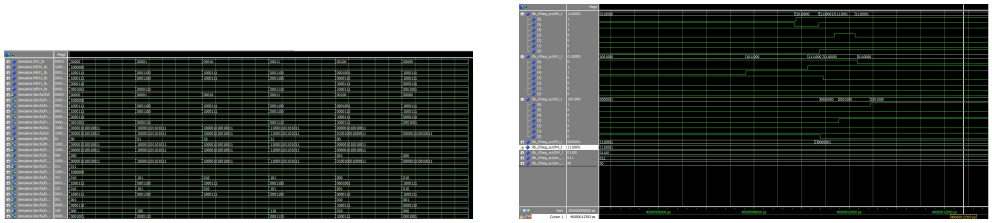
The circuit designed in the first assignement could have been used instead, but compared to the dataflow structure with Boolean expression previuosly used, a behavioural approach is more straightforward and readable.

3.2 Functional simulation

With the testbench [*tb_d7seg_out.vhd*] it is possible to check the functionalities of the completed design. In particular after the typical empty entity the behavioural architecture needs to contain, in the declarative part, the component, defined in *d7seg_out.vhd* and the internal signals for the simulation (*SW_t*, *H0_t*, *H1_t H2_t*, *H3_t*, *H4_t*, *sw_word*, *sw_shift*).
Then, in the descriptive part, once the internal signals are properly mapped to the components' ones, the simulation is performed by considering for each word all the possible shifting of the letters and waiting a short amount of time in between each commutation

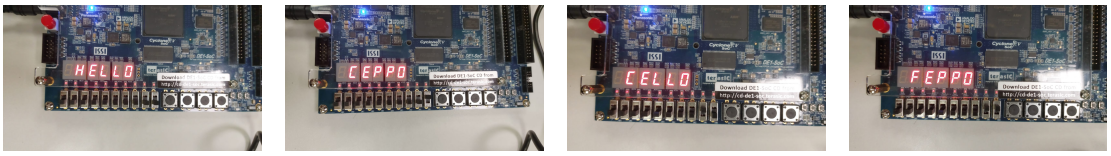
3.3 Synthesis

Once the circuit is successfully synthesized by Quartus it is possible to run a timing simulation, as well as the Modelsim functional simulation, and than observe the results.



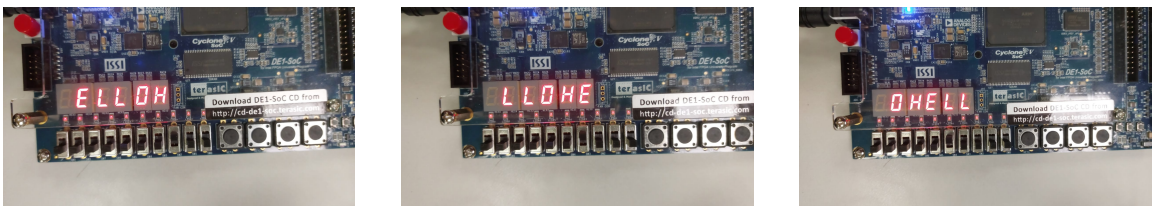
(a) Modelsim functional simulation (b) Timing simulation

Figure 3: Simulations



(a) HELLO (b) CEPPO (c) CELLO (d) FEPPO

Figure 4: Words



(a) First letter shifting (b) First two letter shifting (c) Last four letter shifting

Figure 5: Examples of rotating word

4 Binary to Decimal converter

4.1 Design entry

In order to design a circuit that converts a four-bit binary number *V = v3 v2 v1 v0* into its equivalent two-digit decimal number *D = d1d0* it is first important to define all the allocated blocks:

- The comparator is defined in *comparator.vhd*. After declaring the input *A* as 4-bit unsigned, in order to properly consider the 4-bit binary number *V = v3 v2 v1 v0*, the output as a single-bit signal and considering the suitable libraries it is possible to manage the behavior of the comparator in the architecture by means of a process. In particular a variable *b* is declared to represent the number '9' so that it is possible to check whether the input *A* is greater than 9 or not in order to associate to the output *Q* the value '1' or '0' respectively.

- The circuit A (defined in *circ_a.vhd*) manages the last 3 bits of the binary number (*v2 v1 v0*): it consider the last three bits of the six cases when the whole 4-bit binary number is greater than 9 and converts them into the correspondent binary number of the second digit of the equivalent to the 4-bit binary number.

- The four one-bit wide 2-to-1 multiplexer are defined in *mux_2_1_1bit.vhd* (it was first implemented in **LAB1**). Depending on the output from the comparator, so i.f the number is greater than 9 or not, the output of the multiplexers will be equal to the original bit or to the bit assigned from the circuit A for the last three bits (*v2 v1 v0*) and equal to '0' for the first bit (*v3*).

- The *converter.vhd* is used to properly connect all the components previously defined by mapping their inputs and outputs to that defined for the converter in the entity (*v, m, z*) and the internal signals defined in the architecture (*Q1, Y1, V1*). I particular the input *v* of the converter is connected to one input of each multiplexer (*x*) and that of the comparator (*A*); the internal signal *Y1* is connected to second input of three multiplexers *y* (for the first multiplexer it is associated to the logic value '0') and to the output of the circuit A *Y*;the internal signals *Q1* is connected to the output of the comparator *Q*, to the selection bit *s* of the multiplexers, and connected the one of the outputs of the converter (*z*); the other output of the converter (vector *m*) is connected to each output of the multiplexers *m*; the internal signal *V1* is connected to the input of the circuit A (*B*). In the architecture, the conversion from *unsigned* to *std_logic_vector* of the input *v* also takes place.

- The circuit B focuses on the implementation of the first digit *d1* of the decimal number. It is defined in *circ_b.vhd* by means of a process where if the enable input is equal to the logic value '1' (so if the number is greater than 9) it consider the cases when the input vector *D* corresponds to a number greater than 9 and assign to the output *H* (which will be connected to the 7-segment display) "1001111" in order to lit only the segments needed to display the number '1' (in any other case the display will be blank since it is associated "1111111"); on the other hand if the enable input is not equal to '1' (so if the number is less than 9) the display will show '0' since it is associated "0000001" to the output *H*.

- The decoder is used to display the second digit *d0* of the decimal number, so it consider all the cases when the 4-bit input *C* corresponds to a decimal number from '0' to '9' and associate to the output *Display* the right combination in order to lit the correspondent number in the display. It also manages possible errors by associating to *Display* "1111111", which will result in a blank display.

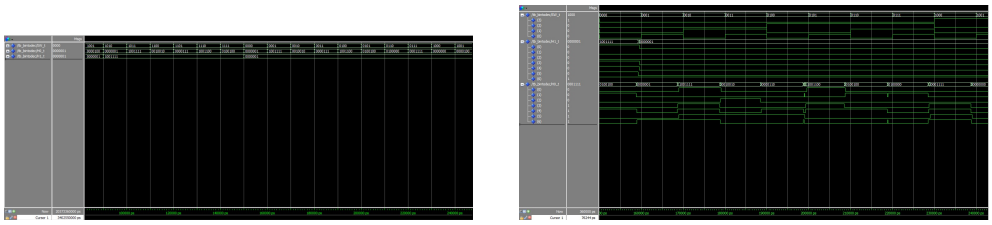
- Lastly, in the *binarytodecimal.vhd* file the mapping of the inputs and outputs values of the converter, the circuit B and the decoder to those assigned to the DE1 board takes place. That is after defining in the entity the switches *SW* in input and the *HEX0, HEX1* displays in output, and in the architecture the components and the internal signals *Z1, M1*.

4.2 Functional simulation

With the testbench [*tb_biotodec.vhd*] it is possible to check the functionalities of the completed design. In particular after the typical empty entity, in the declarative part of the behavioural architecture is defined the component (specified in *binarytodecimal.vhd*), and the internal signals for the simulation (*SW_t*, *H0_t* and *H1_t*.
Then, in the descriptive part, once the internal signals are properly mapped to the components' ones, the simulation is performed by considering all the possible switching combinations and waiting a short amount of time in between.

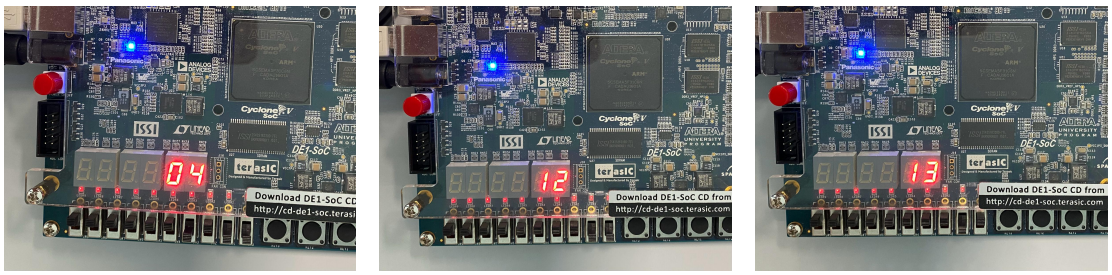
4.3 Synthesis

Once the circuit is successfully synthesized by Quartus it is possible to run a timing simulation, as well as the Modelsim functional simulation, and than observe the results.



(a) Modelsim functional simulation (b) Timing simulation

Figure 6: Simulations



(a) (b) (c)

Figure 7: Examples of results

5 Binary-to-BCD converter

5.1 Design entry

in order to design a combinational circuit that converts a 6-bit binary number into a 2-digit decimal number represented in the BCD system it is necessary to define a suitable decoder and map its inputs and output to those assigned to the DE1 board.

The decoder is defined in *decoder_bcd.vhd* by means of a process that consider all the possible cases for a 3-bit input (C) and associates to the 7-bit output ($Display$) the combination of lit segment that corresponds to the equivalent decimal number (considering that the number that can be displayed are from '0' to '7'). In case of errors it associate a "blank display" to the output (thus the combination "1111111").

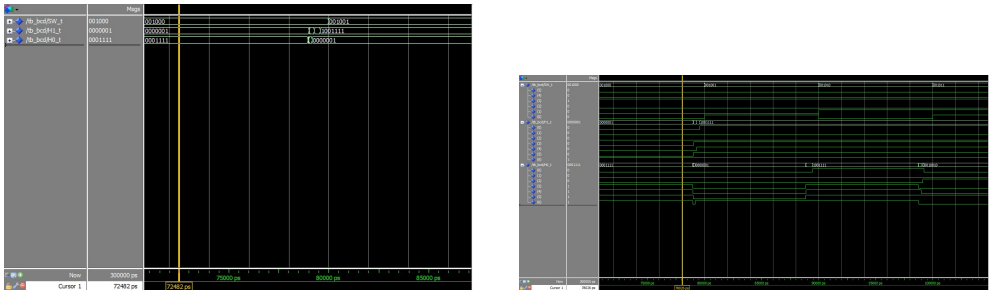
In the *bcd_converter* file takes place the mapping of the input C and output $Display$ of the decoder (defined as a component) to the switches *SW5-0* and the 7-segments displays *HEXO, HEX1* of the board (formerly defined in the entity), considering that the decimal numbers to be displayed are two so the vector associated to the switches needs to be split into two.

5.2 Functional simulation

With the testbench [*tb_bcd.vhd*] it is possible to check the functionalities of the completed design. In particular after the typical empty entity, in the declarative part of the behavioural architecture is defined the component, (specified in *bcd_converter.vhd*), and the internal signals for the simulation (*SW_t*, *H0_t* and *H1_t*.
Then, in the descriptive part, once the internal signals are properly mapped to the components' ones, the simulation is performed by considering all the possible switching combinations and waiting a short amount of time in between.

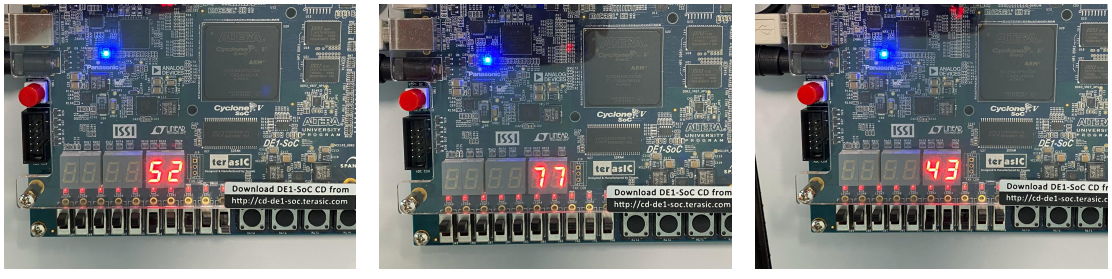
5.3 Synthesis

Once the circuit is successfully synthesized by Quartus it is possible to run a timing simulation, as well as the Modelsim functional simulation, and then observe the results.



(a) Modelsim functional simulation (b) Timing simulation

Figure 8: Simulations



(a) (b) (c)

Figure 9: Examples of results