Digital Systems Electronics Laboratory 02

Group 13

s295391 Giorgio Zoccatelli
s294422 Lorenzo Iemmulo
s295567 Vittorio Macripò

Due Date:	April 1, 2024
Delivery date:	March 31, 2024
Instructor:	Professor Guido Masera

Politecnico di Torino
Accademic Year 2023/24

Contents

1 Introduction **3**

2 Controlling a 7-segment display **3**
2.1 Design Entry 3
2.2 Functional Simulation 3
2.3 Synthesis 3

3 Multiplexing the 7-segment display output **4**
3.1 Design Entry 4
3.2 Functional Simulation 4
3.3 Synthesis 5

4 Binary to decimal converter **5**
4.1 Design Entry 5
4.2 Functional Simulation 6
4.3 Synthesis 6

5 Binary-to-BCD converter **7**
5.1 Design Entry 7
5.2 Functional Simulation 7
5.3 Synthesis 8

1 Introduction

The purpose of this laboratory activity is to implement both known elementary units already seen in the previous lab activity such as multiplexers and new units such as decoders, comparators or shifters and connect them not only to the switches but also to the 7-segment displays on the FPGA in order to obtain different types of decoders.

2 Controlling a 7-segment display

2.1 Design Entry

The aim of the first exercise is to get familiar with the 7-segment display by implementing a decoder that takes as an input a 3-bit array and gives as an output the display of one of the four letters forming the world "HELLO" according to a given truth table. Of course with a 3-bit array we can configure up to eight different configurations so half of the arrays will return a blank display. In order to obtain this result we implement a simple design called *lettere.vhd* that includes both the decoder and the pin connection to our display on the FPGA called HEX_0. Notice that the convention is to assign the logical value '1' if we want to turn off a segment while we assign the logical value '0' if we want to turn on a segment.

2.2 Functional Simulation

In this functional simulation we impelement a testbench trying to display the entire word "HELLO" one letter at a time. Once the world has been displayed we try one of the blank configurations that will turn off the display. The piece of code shown in Figure 1 refers to the

```
SW_tb <= "000"; -- H
wait for 20 ns;
SW_tb <= "001"; -- E
wait for 20 ns;
SW_tb <= "010"; -- L
wait for 20 ns;
SW_tb <= "010"; -- L
wait for 20 ns;
SW_tb <= "011"; -- O
wait for 20 ns;
SW_tb <= "100"; -- blank
wait;
```

Figure 1: 7-segment display testbench configurations

file *lettere_tb.vhd*.

2.3 Synthesis

We complete our exercise by adding the two files *lettere.vhd* and *lettere_tb.vhd* to a new ModelSim project in order to synthesize our project before downloading it on the FPGA. In Figue 2 we can see the transision between the letter "H" and the letter "E".

Figure 2: 7-segment display wave analysis

3 Multiplexing the 7-segment display output

3.1 Design Entry

In this exercise the aim is to implement a circuit that displays a word chosen between "HELLO", "CELLO", "CEPPO" and "FEPPO" eventually shifting their letters. We use in part previous implemented designs including them in a single top entity to describe the entire circuit. In particular we need these components:

- A 4-to-1 multiplexer used to choose one of the four world we want to display (for this purpose we allocate the switches $SW_{1\text{-}0}$).
- A shifter driven by the switches $SW_{4\text{-}2}$ used to decide if we want to shift the word.
- A 7-segment decoder to actually see the word displayed for which we need the displays $HEX_{4\text{-}1}$.

All these components correspond to the files *mux.vhd*, *shifter.vhd* and *display.vhd* which are then declared and connected in the top entity design called *mux_shifter_design*. We don't spend much words about the single designs because they're almost all a natural extension of the previous exercises, while we point out that in the top entity we declare two internal 15-bit wide signals *a1* and *a2* to connect the three designs: the signal *a1* will be the output of the multiplexer and the input of the shifter as well as the signal *a2* will be the output of the shfiter and the input of the display.

3.2 Functional Simulation

In the functional simulation *mux_shifter_decoder.vhd* we want to implement the testbench of four different words configurations in Figure 3. We implement four different words with four different shift to cover most of the possible configurations.

```
SW_tb <= "00100"; --Configurazione "ELLOH"
wait for 20 ns;
SW_tb <= "01001"; --Configurazione "PPOCE"
wait for 20 ns;
SW_tb <= "01110"; --Configurazione "LOCEL"
wait for 20 ns;
SW_tb <= "11011"; --Configurazione "OFEPP"
wait;
```

Figure 3: Multiplexing 7-segment display testbench configurations

3.3 Synthesis

In the final part of the simulation made on ModelSim, we upload as always the design files as well as the testbench and we look at the wave analyzer to check if the output is correct. In

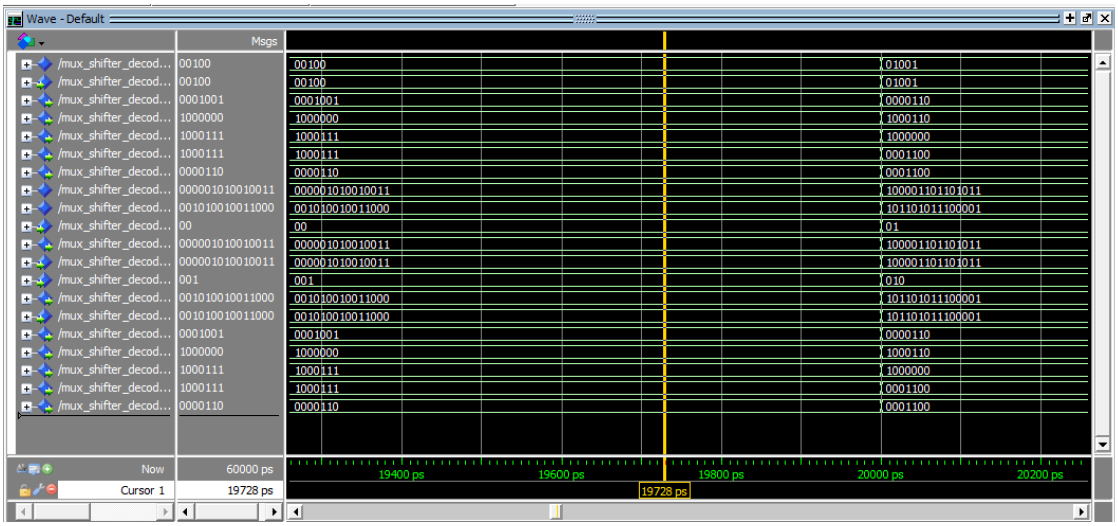

Figure 4: Multiplexing 7-segment display wave analysis

Figure 4 we can notice the transition between the word "ELLOH" and the word "PPOCE". We are now ready to download our design on the FPGA.

4 Binary to decimal converter

4.1 Design Entry

We start by defining the top-level entity, called *converter_display.vhd*. The architecture of this entity is composed of two components, corrisponding to the file *converter.vhd* and the *display.vhd*, which have different functions.
The converter takes as input the 4-bit number, expressed by the four switches $SW_{3\text{-}0}$ and converts it by defining z and m, respectively the first and second digit of the number expressed in decimal. By the fact the 4-bit input number has $2^4 - 1 = 15$ as maximum number in decimal unit, it is correct to represent z only as '0' or '1', so in order to compose the code, it is sufficient to only

control if the value of the number is more than 9 to have $z = 1$ or less to have $z = 0$ (see Figure 5). In the second part of the circuit the display takes as input z and m and gives us as output

```
case to_integer(SW) is
   when 0 to 9 =>
      z <= '0';
      m <= std_logic_vector(to_unsigned(to_integer(SW), 4));
   when 10 to 19 =>
      z <= '1';
      m <= std_logic_vector(to_unsigned(to_integer(SW) - 10, 4));
```

Figure 5: Binary to Decimal converter design description

respectively the HEX_1 and HEX_0 display. The way this works is exactly the same as for the first two exercizes.

4.2 Functional Simulation

In order to test the design, the testbench (called *converter_display_tb*) is created. Here we assign signals to SW_tb and we test the some of the possible configuration (the chosen ones can be seen in Figure 6)

```
begin
   SW_tb <= "1001"; -- Number 9
   wait for 20 ns;
   SW_tb <= "0100"; -- Number 4
   wait for 20 ns;
   SW_tb <= "1111"; -- Number 15
   wait for 20 ns;
   SW_tb <= "1101"; -- Number 13
   wait;
```

Figure 6: Binary to Decimal converter testbench configurations

4.3 Synthesis

Finally the signals used in the testbench are analysed using ModelSim. Following the procedure also used for the previous exercises, we can verify the correctness of the testbench and the signals, and we get the Figure 7, where in this case the configuration for the number 9 and the number 4 are verified.

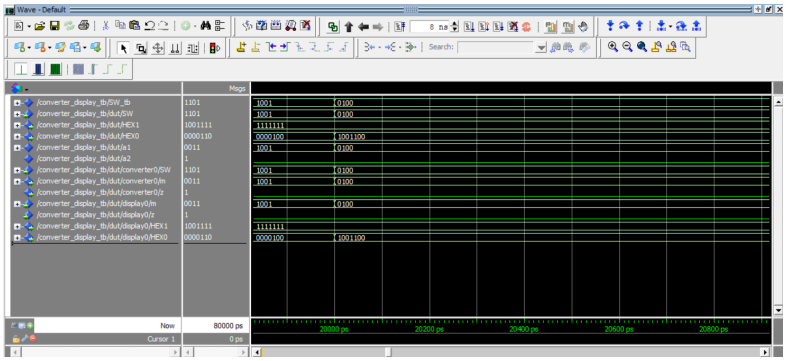

Figure 7: Binary to Decimal converter wave analysis

5 Binary-to-BCD converter

5.1 Design Entry

Similarly to the previous excercise, the description of a top-level design is created through the use of two different components:

- A binary-to-BCD converter
- A display

The first, described in *converter2.vhd*, has the task of converting the 6-bit binary number (input) in the respective decimal number (output) using the BCD (Binary Coded Decimal) representation that allows to obtain a decimal number from the combination of 4-bit binary representations.
Before realizing the actual design of the converter, we have to make some considerations.
A 6-bit binary number, has at most 2^6-1 representations, meaning that the maximum decimal value is 63. This also means that the first digit driven by the signal z that appears on the HEX_1 display varies from 0 to 6, while the second, that appears on the HEX_0 display driven by the signal m can vary from 0 up to 9.
To implement this, multiple comparators are used to divide different cases on the basis of the value of the tens digit represented by z and then determining the value of ones digit driven by m in the following way:

```
when 40 to 49 =>
        z <= "0100";
        m <= std_logic_vector(to_unsigned(to_integer(SW) - 40, 4));
```

Figure 8: Binary-to-BCD converter design description

For the description of the display design we use a natural extension of the code *display.vhd* used in excercise 2 that performs the same function (we call the updated design *display2.vhd*).

5.2 Functional Simulation

To test the correct functioning of the design, a testbech called *converter2_display_tb.vhd* is created, in which 6-bit test values are assigned to SW_tb. The following configurations are chosen to cover almost all possible eventualities.
As it can be seen from the Figure 9, the test signals also verify the functioning of the blank display, in addition to the most classic numerical configurations.

```
SW_tb <= "111111"; -- number 63
wait for 20 ns;

SW_tb <= "000000"; -- number 00 or blank
wait for 20 ns;

SW_tb <= "101111"; -- number 47
wait for 20 ns;

SW_tb <= "000010"; -- number 2
wait for 20 ns;

SW_tb <= "001011"; -- number 11
wait;
```

Figure 9: Binary-to-BCD converter testbench configurations

5.3 Synthesis

SW_tb signals are then analysed using ModelSim. The correct functioning of the implementation is verified by the evolution of the logical signals, as shown in Figure 10 below. The first output signal does in fact represent the number 63 by printing 6 on HEX1 and 3 on HEX0, while the second configuration is verified to be blank.
Every 20 ns a configuration change is made with a consequent change in the trend of the logical waves, but always in accordance with the expected results.

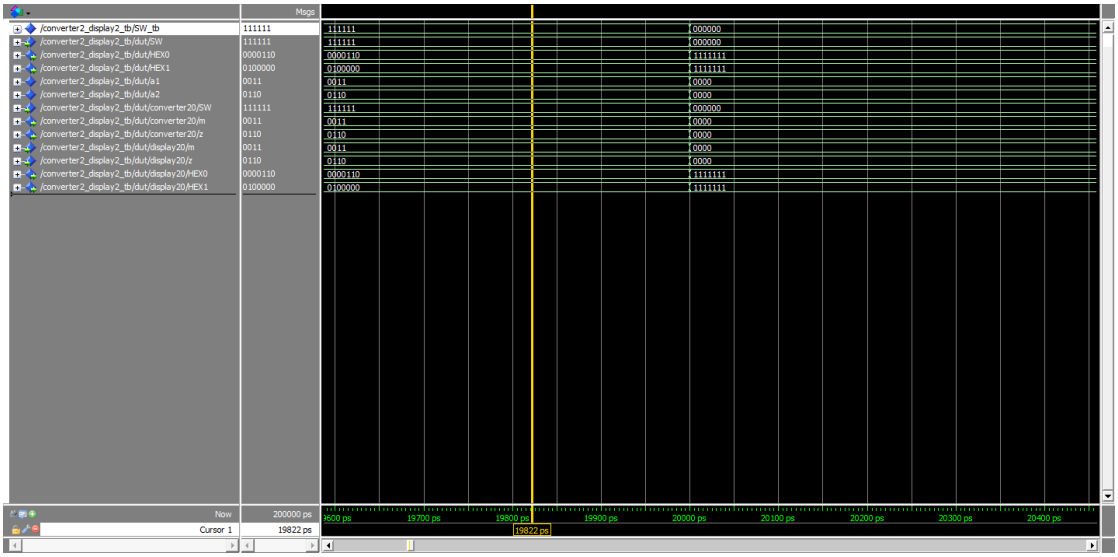

Figure 10: Binary-to-BCD converter wave analysis

In the same Figure we can also see the transition between the ”63” configuration and the ”blank” configuration reported in the previous section.