Digital Systems Electronics Laboratory 01

Group 13

s295391 Giorgio Zoccatelli
s294422 Lorenzo Iemmulo
s295567 Vittorio Macripò

Due Date:	March 25, 2024
Delivery date:	March 24, 2024
Instructor:	Professor Guido Masera

Politecnico di Torino
Accademic Year 2023/24

Contents

1 Introduction

The aim of this laboratory is to get familiar with the FPGA DE1-SoC by implementing three simple examples. In the first example we want to control the ten LEDs on the board by using ten switches while in the second and third examples we want to implement two different types of multiplexer: a four bit wide two-to-one multiplexer and a three bit wide five-to-one multiplexer.

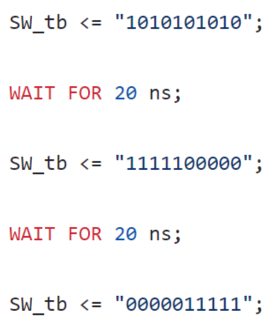
2 Controlling the LEDs

2.1 Design Entry

First of all we want do describe the steps used to reach the design entry reported in *led.vhd*. We have to open the software Quartus Prime, select **New Project Wizard**, choose the directory for the file and choose the file name which has to match the top entity name in the design file. Then we skip until the voice **Family, Design & Board Settings** and we select the device **5CSEMA5F31C6** which is the FPGA used on Altera's DE1-SoC. We can now proceede with the following path to open a new clean sheet where we can write the actual design entry: **File** → **New** → **VHDL File**. Another important step to succesfully compile the design entry is to select a proper pin assignemt based on the FPGA we have to use to correctly refer to the pin we want to use, so we have to go to **Assignments** → **Import Assignments** and we select the file reporting the pin assignment characterized by the *.qsf* extension (in our case the file *DE1_SoC.qsf* was given). After the completion of the design entry we can compile the file to find eventual errors following the path **Processing** → **Start Compilation**.

2.2 Functional Simulation

The Functional Simulation consists in a testbench of the device under test. Therefore we choose some input configurations to check in the next step if the design implemented works as we expect. Basically we have to create a new VHDL file following the exact steps of section 2.1. We can



```
SW_tb <= "1010101010";

WAIT FOR 20 ns;

SW_tb <= "1111100000";

WAIT FOR 20 ns;

SW_tb <= "0000011111";
```

Figure 1: Led testbench configurations

notice looking at Figure 1 that the three configurations should show these results:

1. Alternating lit LEDs
2. Half of the LEDs lit half off
3. Complementary configuration of configuration number 2

2.3 Synthesis

The last step of our simulation is to emulate the behavior of the configuration in our testbench by using ModelSim. After the opening of the software we will follow this path: **File** → **New** → **Project** then under the voice **Project Name** we will match the top entity name of our project and we will choose a new directory for the simulation under the voice **Project location** (we can also rename the **Default library name**). A new screen will show up asking us which files we want to import in our simulation. We will select the voice **Add existing file** two times to select both the design entry and the testbench (in this exercise we are refering to *led.vhd* and *led_tb.vhd*). It is then necessary to follow this path to assure that the design entry will be compiled before the testbench: **Compile** → **Compile Order** → **Auto Generate** (we can also manually select the wanted order).Now we can start the actual simulation by clicking on **Simulate**→ **Start Simulation** and under the directory of our Default library name we select the testbench. We can notice at a first glance that there are no items in the window panel we want to analyze so we have to go under the voice **Add to** → **Wave** → **All items in design** and then **Simulate** → **Run all**. We highlight in Figure 2 the transition between the first chosen

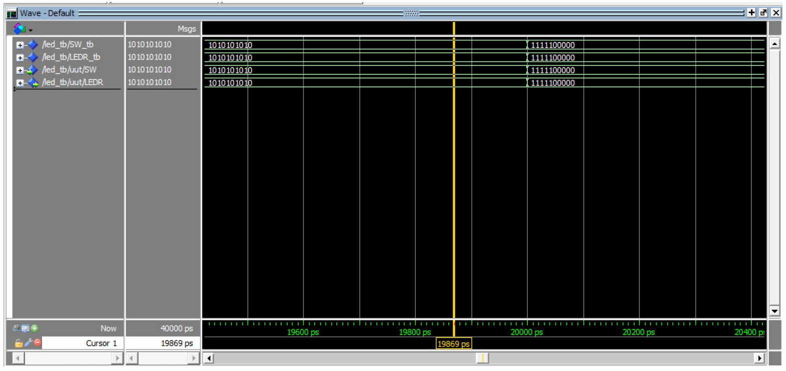


Figure 2: Led wave analysis

configuration and the second chosen configuration discussed in section 2.2.

3 2-to-1 Multiplexer

3.1 Design Entry

We create the design entry by following the same procedure seen in section 2.1.
With Quartus Prime opened, we open a new project and we name it *mux_2.vhd* (we make sure that it matches the entity name in the design file). We continue by selecting the device **5CSEMA5F31C6**, thus we create a VHDL file where we will be writing the actual design. In this project we implement a 4 bits wide 2-to-1 multiplexer, using the switches as the input (defined as a vector of dimension equal to 8) and the LEDs as the output (defined as a vector of dimension equal to 4). The configuration we are following is:

1. The $SW_{7\text{-}4}$ refers to the signal $\boldsymbol{x}$, which is one of the 4-bits wide input signal of the multiplexer.
2. The $SW_{3\text{-}0}$ refers to the signal $\boldsymbol{y}$, the other 4-bits wide input signal.

3. The SW$_8$ refers to the signal s which is the controlling signal of the multiplexer.

Now we continue by using the truth table and finish the design.
We remember to put in the project the pin assignment, then we are ready to compile the designe and see if there are any errors.

3.2 Functional Simulation

We proceed with the Functional Simulation, creating the testbench VHDL file named *mux_2_tb.vhd* where we check if the design implemented works ad expected. The configurations chosen are shown in Figure 3. They should work like this:

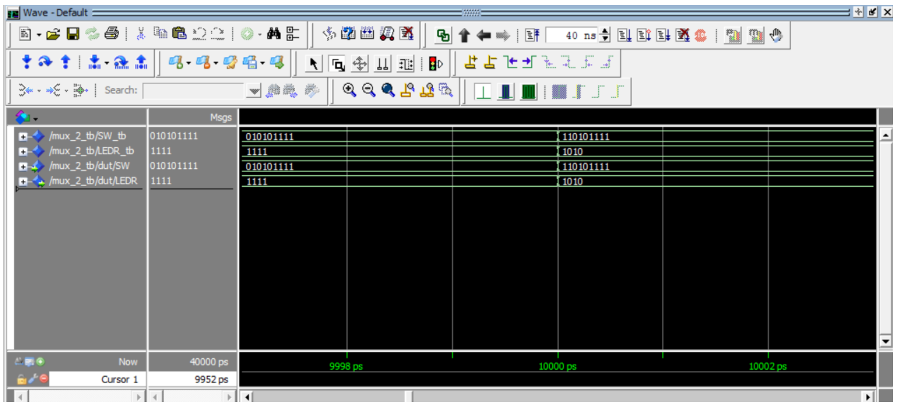
1. Multiplexer driven by 0: output based on SW$_{7\text{-}4}$
2. Multiplexer driven by 1: output based on SW $_{3\text{-}0}$
3. Multiplexer driven by 1: ouput based on SW$_{3\text{-}0}$

```
SW_tb <= "010101111";
WAIT FOR 10 ns;

SW_tb <= "110101111";
WAIT FOR 10 ns;

SW_tb <= "100110101";
WAIT FOR 10 ns;
```

Figure 3: Mux 2-to-1 testbench configurations

3.3 Synthesis

We continue by emulating the behavior of the testbench configuration using ModelSim as we did in section 2.3. Following the same steps, we end up obtaining the waveform shown below.



```
                       Msgs
/mux_2_tb/SW_tb        010101111    010101111   110101111
/mux_2_tb/LEDR_tb      1111         1111        1010
/mux_2_tb/dut/SW       010101111    010101111   110101111
/mux_2_tb/dut/LEDR     1111         1111        1010

Now                    40000 ps     9998 ps     10000 ps     10002 ps
Cursor 1               9952 ps
```

Figure 4: Mux 2-to-1 wave analysis

Here we are highlighting the transition between the first and the second configuration discussed in section 3.2.

4 5-to-1 Multiplexer

4.1 Design Entry

Following the same procedure as for the previous points and keeping the settings unchanged, a new project named *mux_5.vhd* is created. We proceed with the description of the design of a 3 bits wide 5-to-1 multiplexer, that transforms 5 input signals into 1 output signal using respectively SW and LEDR.
Following the indications of the delivery, a design is realized taking into account the values of the 3 selectors of the multiplexer and associating the respective value as indicated on the truth table. This procedure has to be implemented not before having made two clarifications as for the first multiplexer description.

1. With the signal $\boldsymbol{x}$ we are alluding to $SW_{2\text{-}0}$, while with $\boldsymbol{y}$ we are meaning $SW_{5\text{-}3}$.
2. The 3-bit signals $\boldsymbol{u}$, $\boldsymbol{v}$ and $\boldsymbol{w}$ are preassigned to three different given values.

At this point we can actually realize the design distinguishing the different cases indicated by the truth table. This is done by assigning to $SW_{8\text{-}6}$ certain values corresponding to the selectors and evaluating the output m.

4.2 Functional Simulation

The *mux_5_tb.vhd* file describes how the DUT (Device Under Test, in this case a multiplexer) behaves by assigning specific values to the SW_tb signal, whose first 3 values, corresponding to $SW_{8\text{-}6}$ (selectors s_0, s_1 and s_2), determine the output m. Different configurations of SW_tb are therefore implemented in order to cover all the possible cases indicated in the truth table.

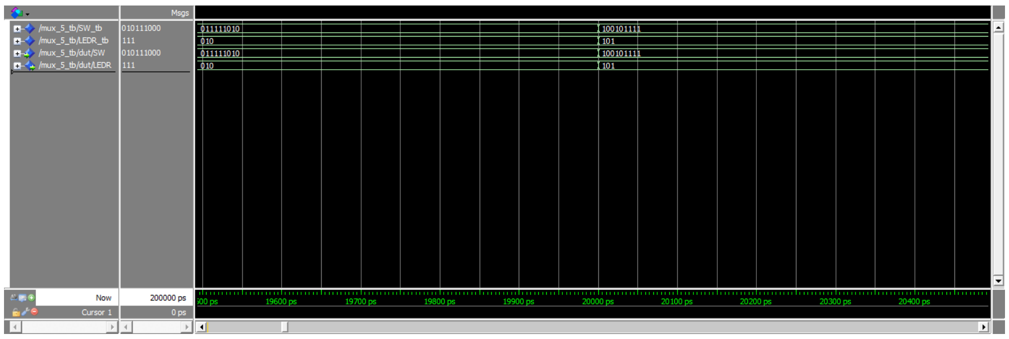
```vhdl
SW_tb <= "011111010";
WAIT FOR 20 ns;

SW_tb <= "100101111";
WAIT FOR 20 ns;

SW_tb <= "101001110";
WAIT FOR 20 ns;

SW_tb <= "110010110";
WAIT FOR 20 ns;

SW_tb <= "111011010";
WAIT FOR 20 ns;

SW_tb <= "000000101";
WAIT FOR 20 ns;

SW_tb <= "001010000";
WAIT FOR 20 ns;

SW_tb <= "010111000";
WAIT FOR 20 ns;
```

Figure 5: Mux 5-to-1 testbench configurations

With the increasing complexity of the design we decide to include more testbench configurations reported in Figure 5.

4.3 Synthesis

Once again we follow the instructions given in section 2.3. The correct functioning of the implementation is verified by the evolution of the logical signals, as shown in Figure 5 below.
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.



Msgs
/mux_5_tb/SW_tb 010111000 011111010 100101111
/mux_5_tb/LEDR_tb 111 010 101
/mux_5_tb/dut/SW 010111000 011111010 100101111
/mux_5_tb/dut/LEDR 111 010 101
Now 200000 ps
Cursor 1 0 ps
500 ps
19600 ps
19700 ps
19800 ps
19900 ps
20000 ps
20100 ps
20200 ps
20300 ps
20400 ps

Figure 6: Mux 5-to-1 wave analysis

In Figure 6 we are seeing the transition between the first and the second configuration reported in section 4.2.