UMONS
Université de Mons

Faculté Polytechnique

Interface MAX7219 ledMatrix SPI

Projet 2021

Readme file completed

Arthur LEFEBVRE, Julien BERTIEAUX

POLYTECH MONS

SEMI Service Electronique et Microélectronique FPMs

Daniel BINON , *Assistant*
Sous la direction de Monsieur le Professeur Carlos Alberto VALDERRAMA SAKUYAMA

Année académique 2020-2021

pôle hainuyer

Sommaire

Chapitre 1

Getting started with Quartus

1.1 Project creation

In this first part, we will see together how to understand the Quatus program in order to use the documentation found on Github. You have to work sequentially in the following way : First, start the Quartus program and wait for the welcome screen to appear. Second, create a project in which to include the Github files. To do this, go to : File - New Project Wizard. In the project creation window, make sure that the addressing is correct, that the processed files are in VHDL and that the simulation program is ModelSim Altera. In the project creation menu, it is also possible to include existing VHDL files. You must use the tool to include Github files, specifying the nature of the file (i.e. whether they are testbenches or not). In order to avoid a problem with the "Top-Level Entity" it is necessary to name the project by the name of the top-level entity.

1.2 Verify the driver

In order to be sure that the project can be operational. It is interesting to test the driver in advance. That is why we have provided a testbench to prove that the system is functional.

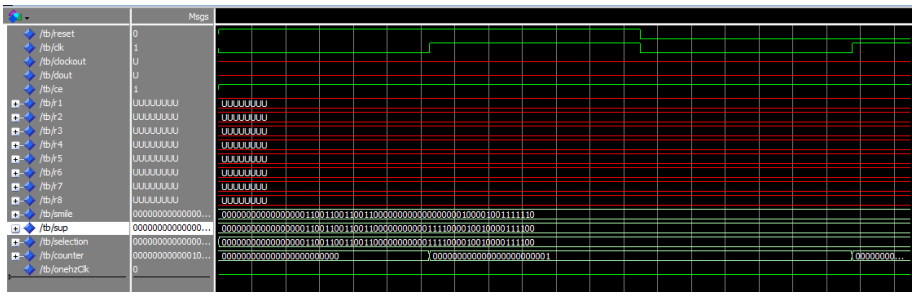


FIGURE 1.1 – Result of the testbench

First, we used a testbench using only a clocked counter to make the driver as simple as possible. This simulation was then run through Altera ModelSim. In the figure 1.2, we can see that at each rising edge of the clock, a counter is incremented. By passing through a counter, we can create a secondary clock which we have used for convenience at the rate of one second. We can also use the counter to adapt the clock from 500MHz to 400kHz to be compatible

with the FPGA.

In the figure 1.2, a happy smiley and a surprised smiley are alternately displayed on the interface. First of all, we have defined a half-period time of 1250 ns (i.e. a frequency of 400kHz) and 64-bit signals for the smileys. Then, in the process, we define the clock and the reset. At each rising edge of the clock, the counter is incremented until it reaches 400000. This gives a clock signal of 1Hz. The smileys then alternate every second with this second clock. Note that the counter is a little oversized as 19 bits are needed and sufficient to count up to 400000.

```
process
   begin
      reset <= '1';
      wait for 2500 ns;
      reset <= '0';
      wait;
end process;
-- "Clock Pattern"
process
   begin
    clk <= '0' ;
    wait for halfperiod;
    clk <= '1' ;
    wait for halfperiod;
end process;

process(clk) begin

   if( rising_edge(clk) ) then

      if( counter >= 400000 ) then
         counter <= std_logic_vector(to_unsigned(0, 26));
         onehzclk <= not onehzclk;
      else
         counter <= counter + 1;
      end if;

   end if;

end process;

process(onehzclk) begin

   if( onehzclk = '1' ) then
      selection <= smile;
   else
      selection <= sup;
   end if;

end process;
```

FIGURE 1.2 – Testbench process

To run the testbench, follow these steps. First, compile the script to check that there are no syntax errors. To do this, click on the blue "play" button in the Quartus toolbar. Secondly, run a simulation by clicking on : Tool - Run Simulation Tool - RTL Simulation. Normally, the Altera ModelSim program should start. Thirdly, in ModelSim Altera, a new associated project must be created. To do this, click on : File - Create New Project. On the menu that just appeared, you must add the files you want to test by clicking on "Add existing files". Here we will test the testbench. Once the project is created, you can launch a compilation in : Compile - Compile All. After compiling successfully, you have to run the simulation with : Simulate - Start Simulation. In the window that has appeared, we select the entity to be tested

in the "Work" area. Finally, add the waveform of the data by selecting : Add - Wave.
In order to clarify the procedure, you will find in the following figures the different steps to follow to start the simulation.

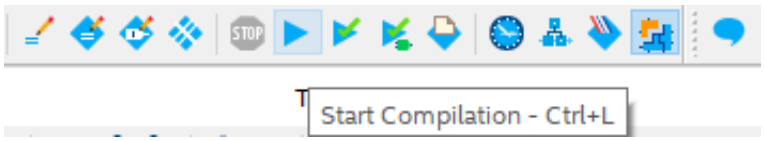


FIGURE 1.3 – Step 1 - Compile all the files in Quartus

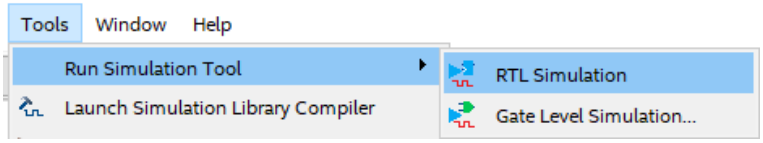


FIGURE 1.4 – Step 2 - Launch the simulation tool ModelSim Altera

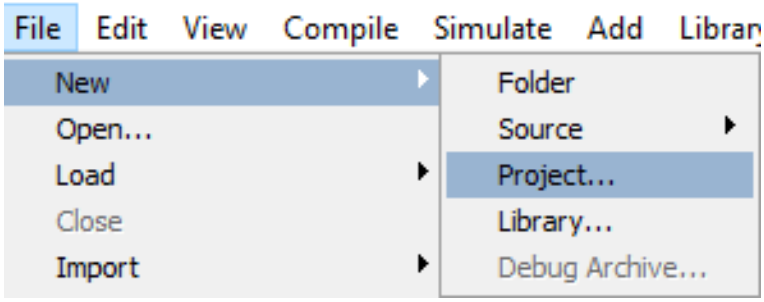


FIGURE 1.5 – Step 3 - Create a new project in ModelSim Altera

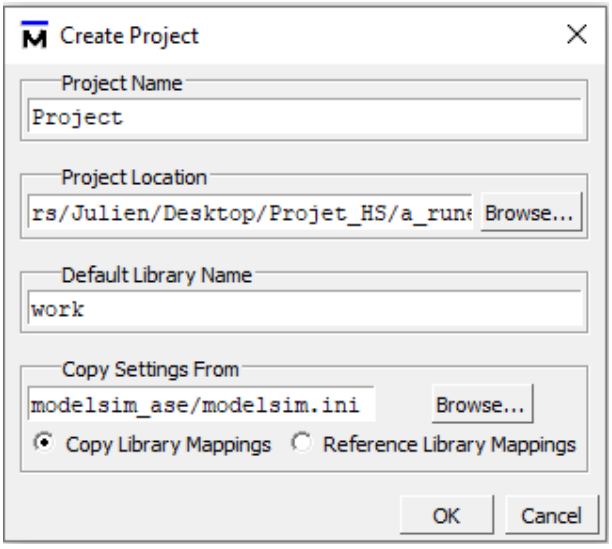


FIGURE 1.6 – Step 4 - Name the new project in ModelSim Altera

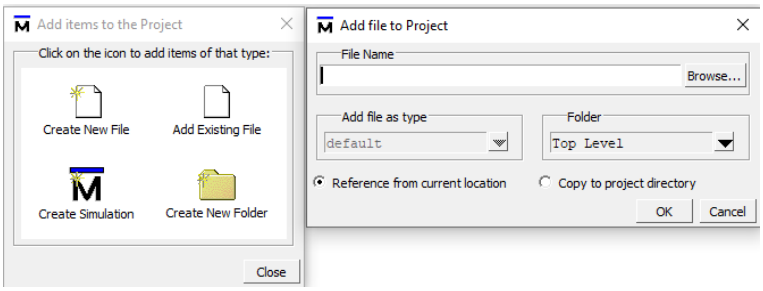


FIGURE 1.7 – Step 5 - Add existing files to the new project in ModelSim Altera

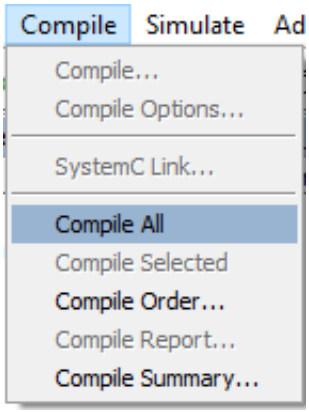


FIGURE 1.8 – Step 6 - Compile all files in ModelSim Altera

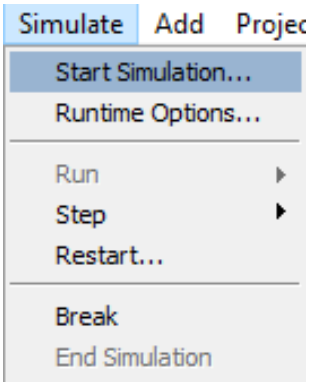


FIGURE 1.9 – Step 7 - Start the simulation in ModelSim Altera

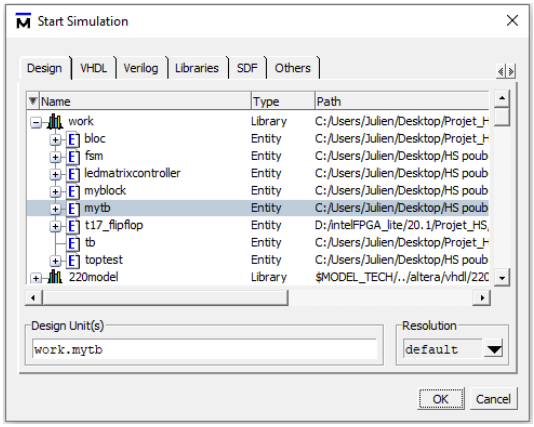


FIGURE 1.10 – Step 8 - Select the right entity for the simulation in ModelSim Altera

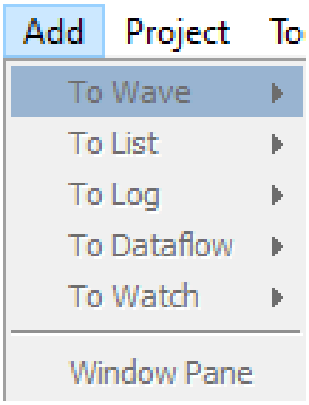


FIGURE 1.11 – Step 9 - Add waves to the simulation in ModelSim Altera

Chapitre 2

Hardware

2.1 DE0 Nano SoC

In order to fulfill this project, we were led to use the DE0 Nano SoC board shown on figure 2.1. In order to have a better understanding on why we used the DE0 nano here is a list of the different advantages and drawbacks we can encounter when using it we got from [1] :

- Advantages
  - The size. It's slightly smaller than a Raspberry Pi and around the size of a credit card.
  - It's portable.
  - It's all inclusive. Despite there being plenty of IO on the board you don't need to connect it to anything to get a lot out of it. The USB programmer is built in and it runs from USB power. No additional hardware is required.
  - It's packed with features. It has a rich array of peripherals on board as well as a large number of IO pins accessible by headers.
  - It's sort of got a case. Not really fully enclosed, but the big slab of acrylic that comes screwed to the top does offer some protection. More than most dev modules, which usually just come as a bare PCB.
- Drawbacks
  - Their thirst for power. Microcontrollers offer far better power consumption, something to consider if your application is battery powered.
  - The learning curve for FPGA development is also considered to be steeper, especially as there is not so much community support as with Raspberry Pi or Arduino development.
  - There are additional skills required for developing FPGA applications, such as understanding how to set up timing correctly and debugging with simulation.
  - The cost consideration. Although getting started with electronics is generally quite reasonable in terms of cost, FPGAs are definitely on the higher end of the price spectrum.

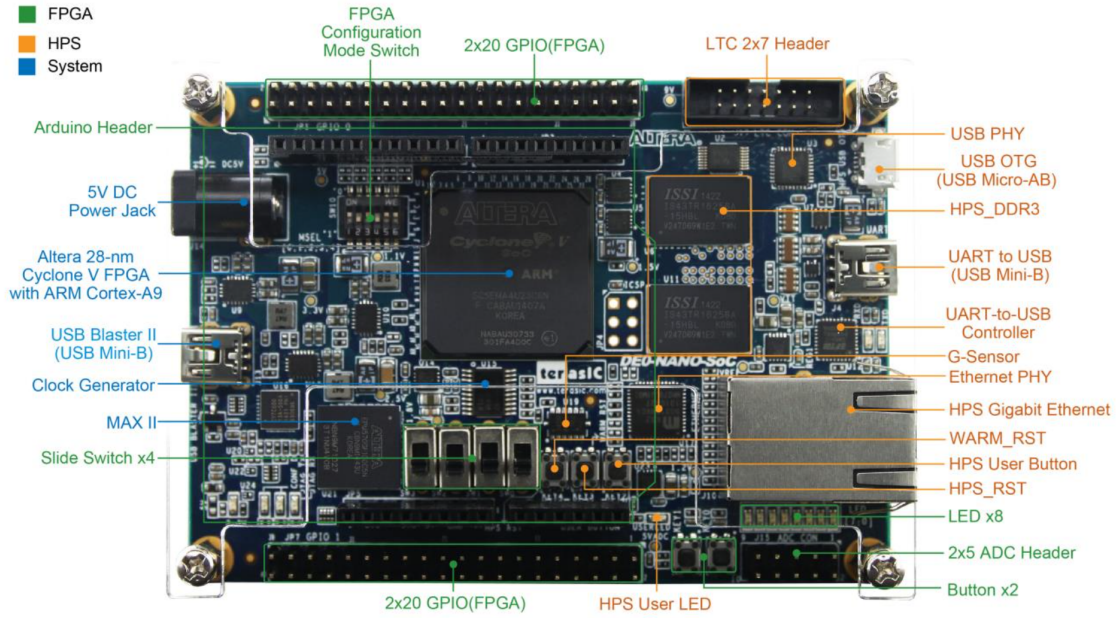


FIGURE 2.1 – DE0-Nano-SoC Board

The board has two 40-pin expansion headers. Each header has 36 user pins connected directly to the Cyclone V SoC FPGA. It also comes with DC +5V (VCC5), DC +3.3V (VCC3P3), and two GND pins. Figure 2.2 and 2.3 shows the I/O distribution of the GPIO connector [2].

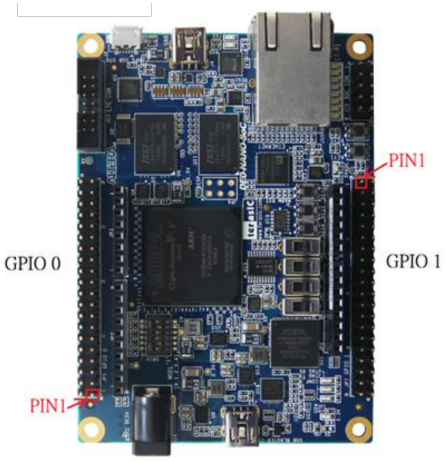


FIGURE 2.2 – DE0-Nano-SoC Board focus on the GPIO connector

GPIO 0 (JP1)

PIN_V12	GPIO_0[0]	1	2	GPIO_0[1]	PIN_AF7	PIN_Y15	GPIO_1[0]	1	2	GPIO_1[1]	PIN_AG28
PIN_W12	GPIO_0[2]	3	4	GPIO_0[3]	PIN_AF8	PIN_AA15	GPIO_1[2]	3	4	GPIO_1[3]	PIN_AH27
PIN_Y8	GPIO_0[4]	5	6	GPIO_0[5]	PIN_AB4	PIN_AG26	GPIO_1[4]	5	6	GPIO_1[5]	PIN_AH24
PIN_W8	GPIO_0[6]	7	8	GPIO_0[7]	PIN_Y4	PIN_AF23	GPIO_1[6]	7	8	GPIO_1[7]	PIN_AE22
PIN_Y5	GPIO_0[8]	9	10	GPIO_0[9]	PIN_U11	PIN_AF21	GPIO_1[8]	9	10	GPIO_1[9]	PIN_AG20
	5V	11	12	GND			5V	11	12	GND	
PIN_T8	GPIO_0[10]	13	14	GPIO_0[11]	PIN_T12	PIN_AG19	GPIO_1[10]	13	14	GPIO_1[11]	PIN_AF20
PIN_AH5	GPIO_0[12]	15	16	GPIO_0[13]	PIN_AH6	PIN_AC23	GPIO_1[12]	15	16	GPIO_1[13]	PIN_AG18
PIN_AH4	GPIO_0[14]	17	18	GPIO_0[15]	PIN_AG5	PIN_AH26	GPIO_1[14]	17	18	GPIO_1[15]	PIN_AA19
PIN_AH3	GPIO_0[16]	19	20	GPIO_0[17]	PIN_AH2	PIN_AG24	GPIO_1[16]	19	20	GPIO_1[17]	PIN_AF25
PIN_AF4	GPIO_0[18]	21	22	GPIO_0[19]	PIN_AG6	PIN_AH23	GPIO_1[18]	21	22	GPIO_1[19]	PIN_AG23
PIN_AF5	GPIO_0[20]	23	24	GPIO_0[21]	PIN_AE4	PIN_AE19	GPIO_1[20]	23	24	GPIO_1[21]	PIN_AF18
PIN_T13	GPIO_0[22]	25	26	GPIO_0[23]	PIN_T11	PIN_AD19	GPIO_1[22]	25	26	GPIO_1[23]	PIN_AE20
PIN_AE7	GPIO_0[24]	27	28	GPIO_0[25]	PIN_AF6	PIN_AE24	GPIO_1[24]	27	28	GPIO_1[25]	PIN_AD20
	3.3V	29	30	GND			3.3V	29	30	GND	
PIN_AF9	GPIO_0[26]	31	32	GPIO_0[27]	PIN_AE8	PIN_AF22	GPIO_1[26]	31	32	GPIO_1[27]	PIN_AH22
PIN_AD10	GPIO_0[28]	33	34	GPIO_0[29]	PIN_AE9	PIN_AH19	GPIO_1[28]	33	34	GPIO_1[29]	PIN_AH21
PIN_AD11	GPIO_0[30]	35	36	GPIO_0[31]	PIN_AF10	PIN_AG21	GPIO_1[30]	35	36	GPIO_1[31]	PIN_AH18
PIN_AD12	GPIO_0[32]	37	38	GPIO_0[33]	PIN_AE11	PIN_AD23	GPIO_1[32]	37	38	GPIO_1[33]	PIN_AE23
PIN_AF11	GPIO_0[34]	39	40	GPIO_0[35]	PIN_AE12	PIN_AA18	GPIO_1[34]	39	40	GPIO_1[35]	PIN_AC22

GPIO 1 (JP7)

FIGURE 2.3 – I/O distribution of the GPIO connector

2.2 Ledmatrix

In this project we are going to design an 8x8 LED matrix display. An 8x8 LED matrix contains 64 LEDs (Light Emitting Diodes) which are arranged in the form of a matrix, hence the name LED matrix. These matrixes can be made by circuiting 64 LEDs, however that process is time consuming. Now a day they are available in compact forms as shown in below image. These compact modules are available in different sizes and many colors. The cost of module is same as cost of 64 LEDs, so for hobbyists this is easiest to work on. The bare LED matrix has 16 pin outs with 8 common positive and another 8 common negative. Because we cannot spare 16 PINS on the DE0 we'll use, we need to connect this matrix to a driver chip. This driver chip along with matrix comes as a set which is shown on figure 2.4 [3].

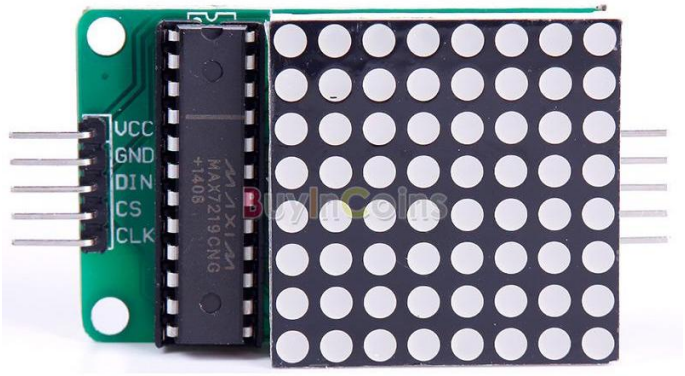


FIGURE 2.4 – LED matrix module

In order to connect it to the DE0, we can have a look at the input and output of the ledmatrix on figure 2.5 [4].

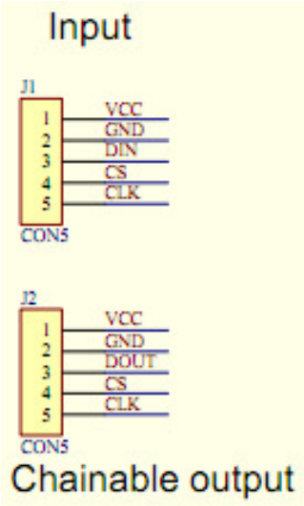
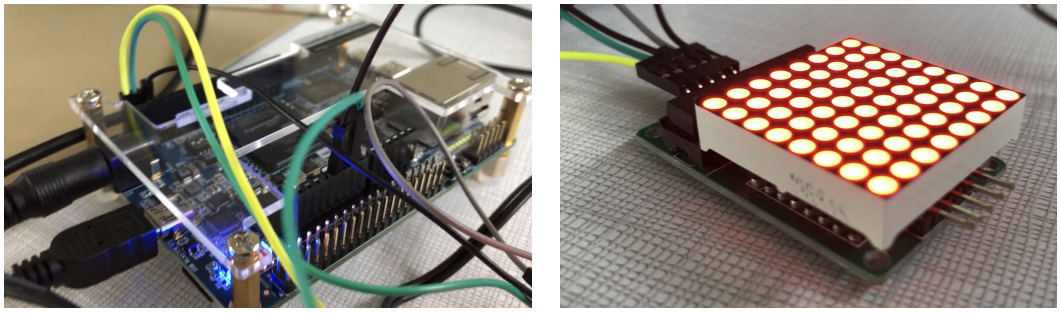


FIGURE 2.5 – Input and Output of Ledmatrix

2.3 DE0 nano SoC golden and Interface MAX7219 ledMatrix SPI connected together

Now that we've looked at both the DE0 nano and the ledMatrix, we need to connect them together. The connections between DE0 nano and LED matrix module are listed bellow and can be seen on figure 2.6 :

- VCC of the led module - +5V
- GND of the led module - GND
- CLK of the led module - GPIO_0[0]
- CS of the led module - GPIO_0[2]
- DIN of the led module - GPIO_0[4]



(a) DE0 nano SoC golden (b) Ledmatrix

FIGURE 2.6 – DE0 nano SoC golden and Interface MAX7219 ledMatrix SPI connected together

Bibliographie

[1] Siytek. *TERASIC DE0-NANO REVIEW : BEST BEGINNER FPGA BOARD UNDER $100 ?* Consulté le 05 Juin 2021.
URL : https://siytek.com/de0-nano-review/#Why_is_parallel_processing_so_awesome.

[2] Altera University Program. *DE0-Nano-SoC User Manual*. August 31, 2017.
URL : http://www.terasic.com.tw/attachment/archive/941/DE0-Nano-SoC_User_manual_rev.C1.pdf.

[3] Anonymous. *8x8 LED Matrix using Arduino*.

[4] Inc LinkSprite technologies. *2770157 LED Matrix Kit. User Guide*.