Tutorial servomotor (write)

Aim of the project/tools

Create a tutorial based on a known development environment and target platform.
We use:

- Altera Quartus (version 18.1 lite) EDA tool (link to download: https://www.intel.ca/content/www/ca/en/software/programmable/quartus-prime/download.html)
- Intel Altera SoC Qsys Platform designer. Intel Altera SoC Qsys Development Environment allows the creation of hardware/software development. It is a combination of a processor (software) and a reconfigurable circuit (templates/examples: https://moodle.umons.ac.be/mod/resource/view.php?id=189534)
- DE1/DE0 nano SoC development kits to interact with the peripheral (servo) through a hardware driver.
- https://www.intel.com/content/www/us/en/software-kit/665987/intel-quartus-prime-standard-edition-design-software-version-18-1-for-windows.html (software tools to compile C program)

The SoC will be developed on the DE0 nano SOC Atlas board provide by Terasic. This board Uses the FPGA Cyclone V and the board is revision C0. When you install Quartus, you must select the device Cyclone V. ModelSim is needed for simulations. It is available when Quartus is installed. You can get the reference design, the template that shows the pins and peripherals available on the board, that will help you to easily connect your hardware or software: Then, download Atlas-SoC Kit Resources Reference Designs and Community Support:
https://www.terasic.com.tw/cgibin/page/archive.pl?Language=English&CategoryNo=205&No=941&PartNo=4#soc

Monitor is also needed to develop software for the processor. We will use the Linux OS which is installed on the SD card. You will need the software tools to compile your C program using the ARM tools. For that, download and install: https://www.intel.com/content/www/us/en/software-kit/665987/intel-quartus-prime-standard-edition-design-software-version-18-1-for-windows.html

Quartus is the tool to develop your hardware. It includes the synthesis, simulation and implementation on an FPGA. The tool has also a link to Platform Designer, that will help you to develop the Platform, the SoC (System-on-Chip) including the ARM processor (PS the Programmable System) and the FPGA (PL the Programmable Logic).

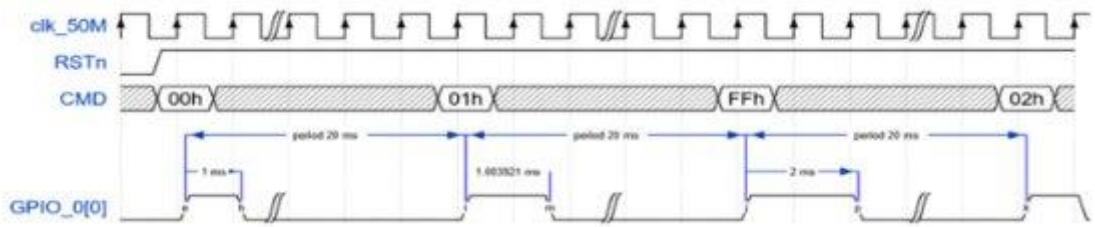
Programming language

VHDL is a hardware description language for representing the behaviour and architecture of a digital electronic system. Its full name is VHSIC Hardware Description Language.

The hardware driver is in VHDL.

Servo protocol: context

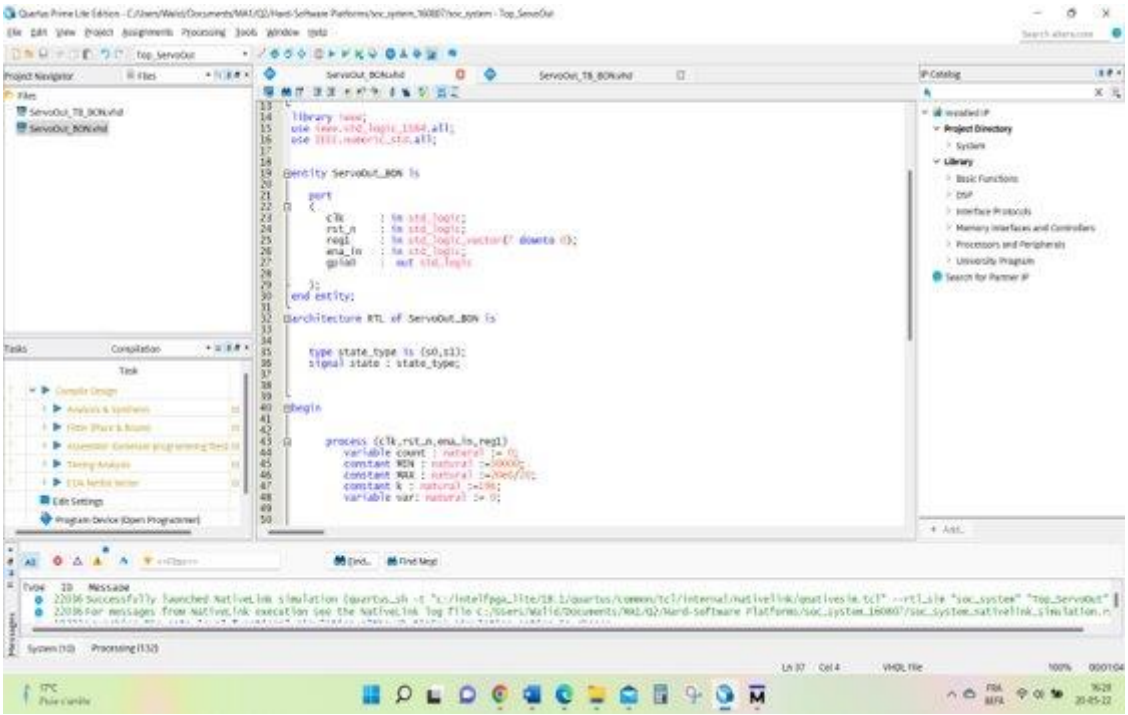
There is a Clock of 50MHz and RSTn reset low, CMD command duration range 0 to 255 (0 to 1ms) 8-bit data input register and a GPIO output pin.

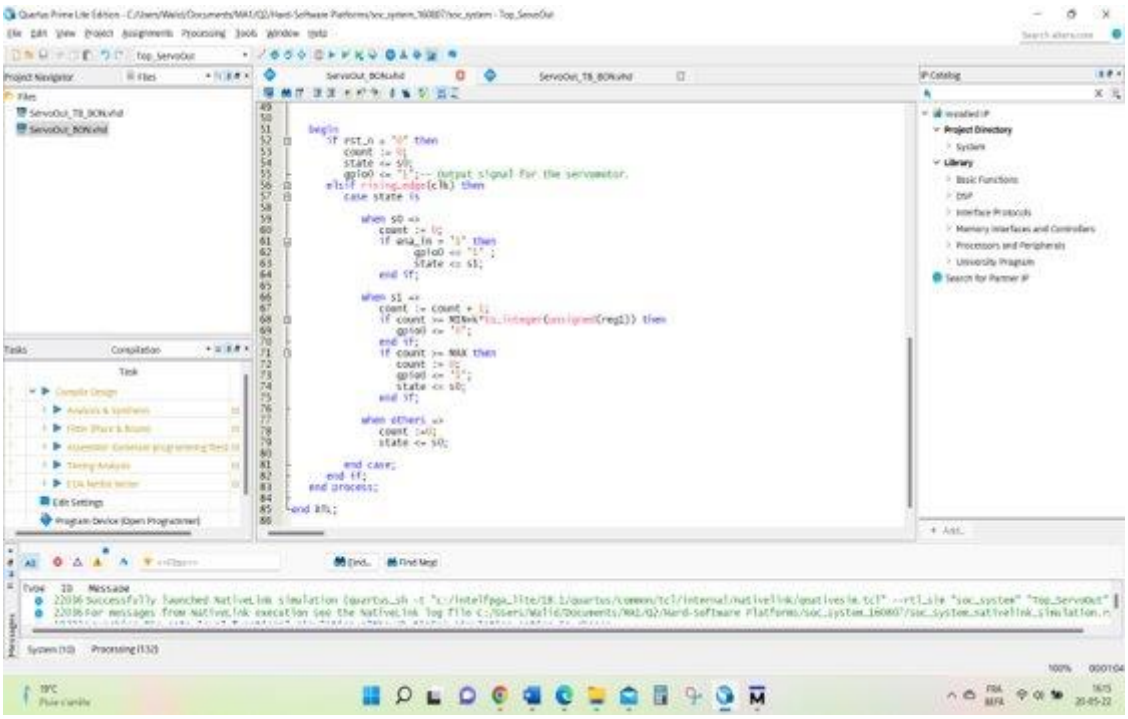


Hardware

Create driver

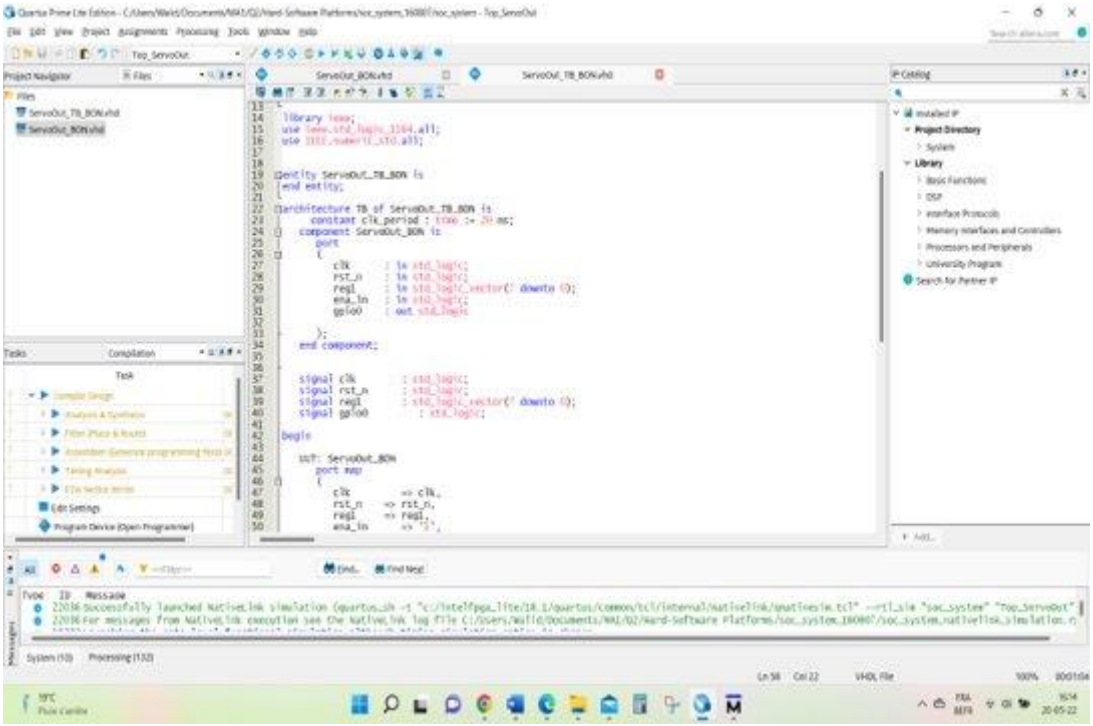
Create a new VHDL file (Driver), create the entity, architecture and ports. We need two 7-bit I/O ports (to connect to the HPS PIO external connection registers of the processor), a clock and reset signals. We will drive the GPIOs of the board.

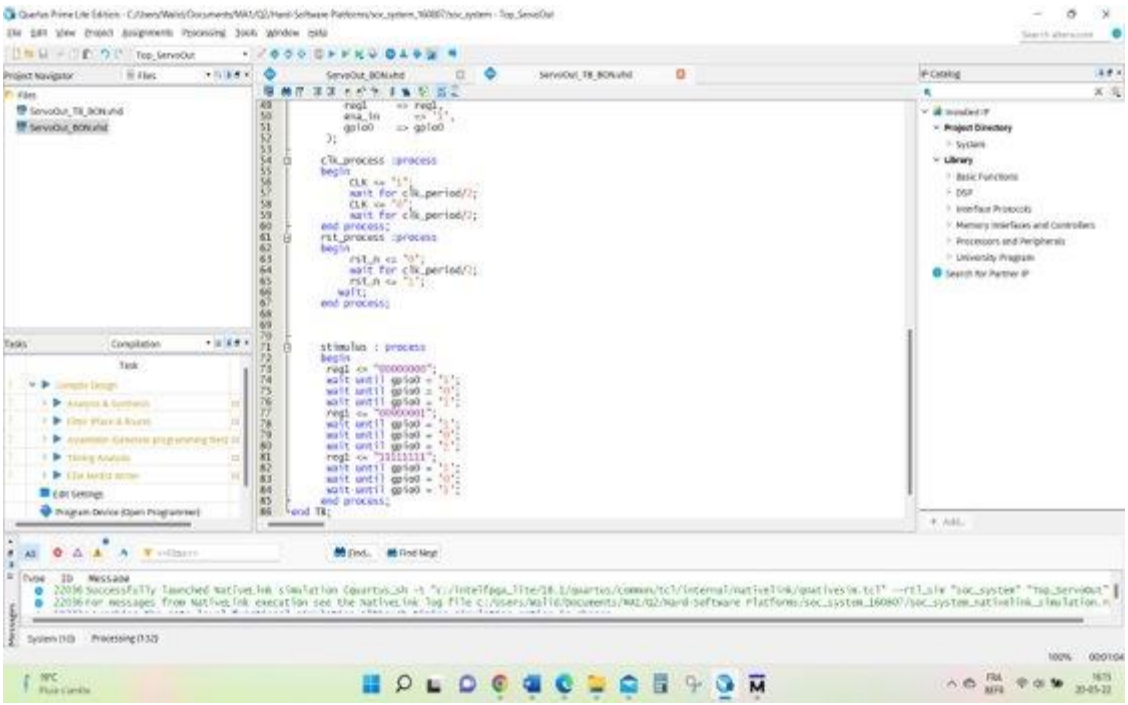




Create the tesbench

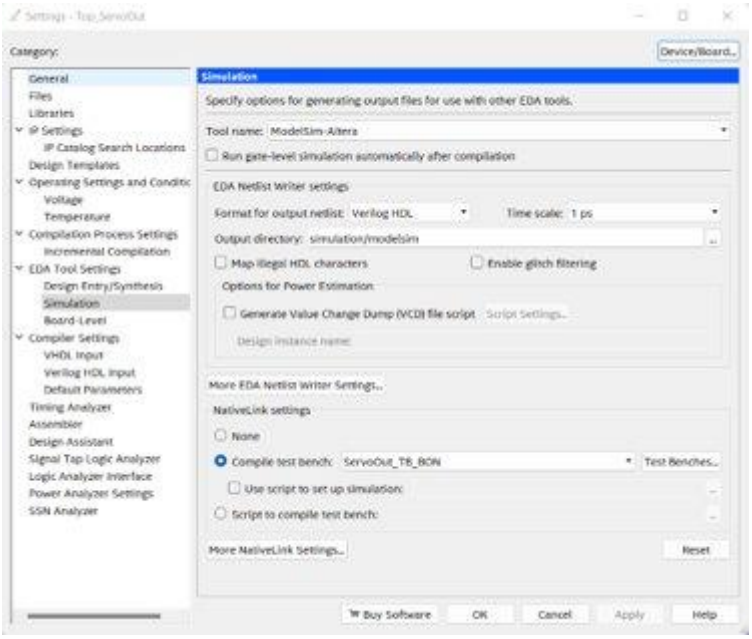
The testbench is used to check that our VHDL code is correct and does what we want. Create new VHDL file (Driver_TB). To be considered as a Testbench, just change the property: right click on the Project Navigator > Drive_TB>Properties> VHDL Test bench. Create the input signals with a default value, the clock and reset signals





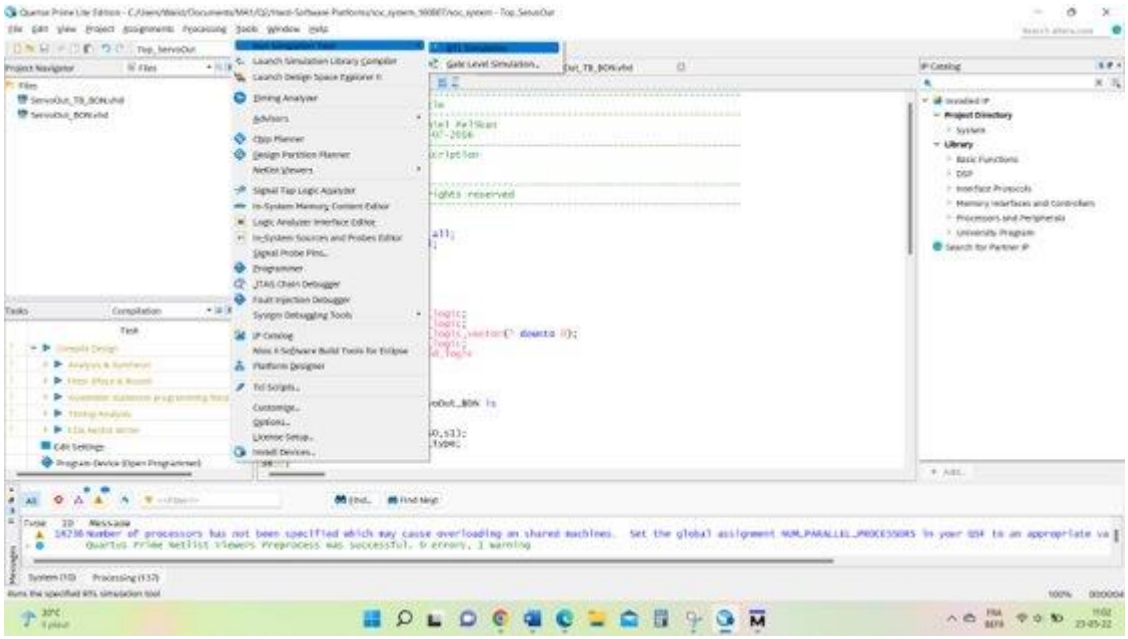
Simulation

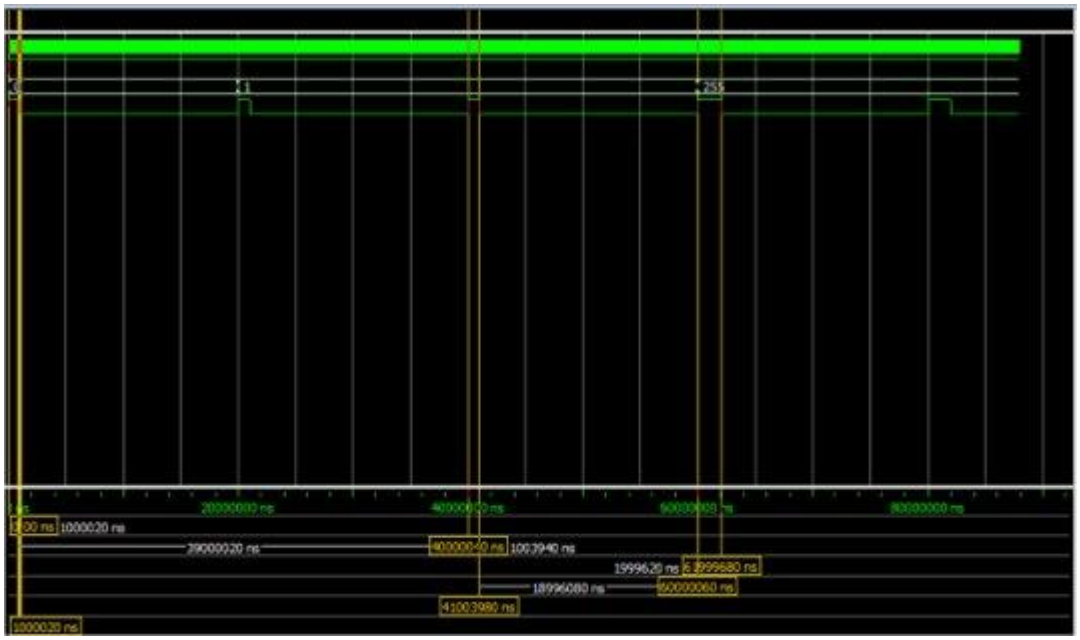
Now, you can test the Driver using the Testbench > Assignments> Settings and add the test bench you created as the file to be used for simulation.

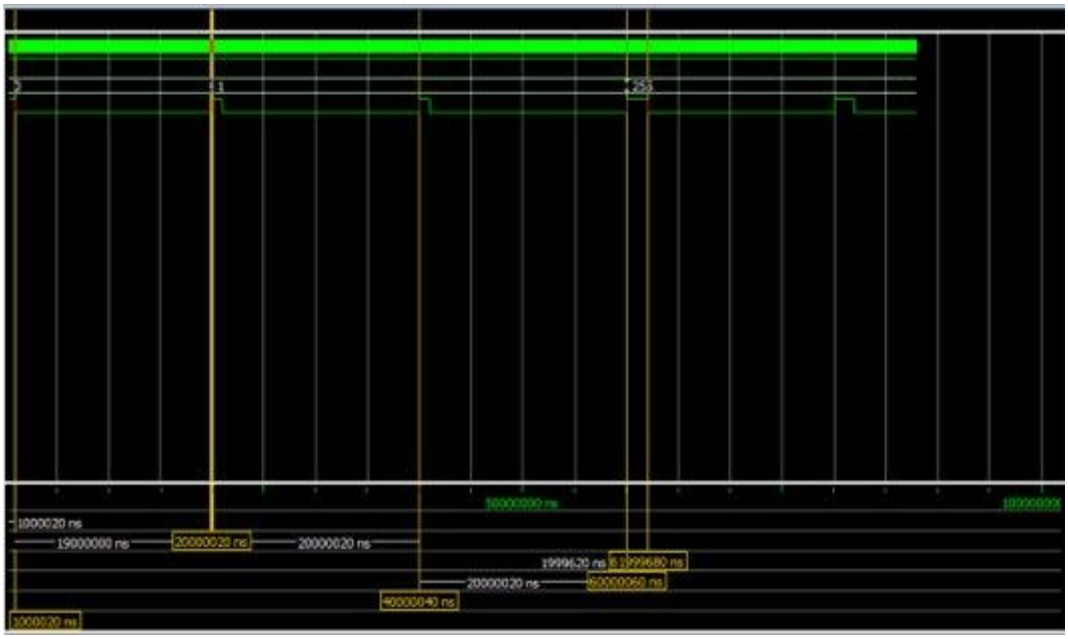


Run the simulation to see if your module is working correctly. With ModelSim, you could see the expected results (1ms for the first (0), 2ms for the last (255) and finally the 20ms period

for the clock). We go from 1 to 2 ms because it is the limit given in the datasheet of the servomotor (we may use the cursors).





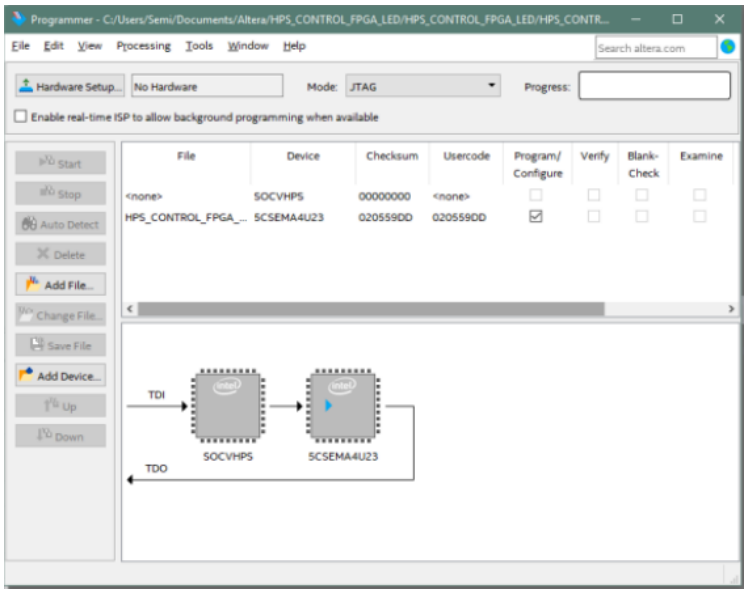


Software

Program the FPGA

First you need to connect your FPGA to the alimentation and the USB blaster (for FPGA programming) to the USB port of your PC:

To program the FPGA just click on Tools/Programmer, when the project is open in Quartus. This will open the tool to program the FPGA:6



In case, you don't see the FPGA (Hardware setup), click on Hardware Setup and you should be able to see it and add it.

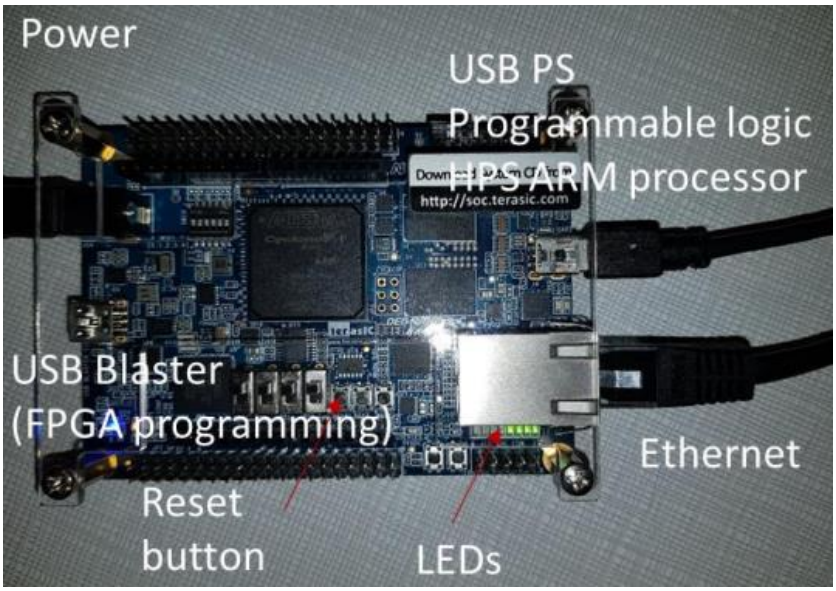
To program the FPGA you need to open the programming file **soc_system.sof**. (click on Add File and select the file)

To program the FPGA, now just click the Start button.

If the programming is correct you will see the message 100% successful.

Program the Processor HPS PS ARM

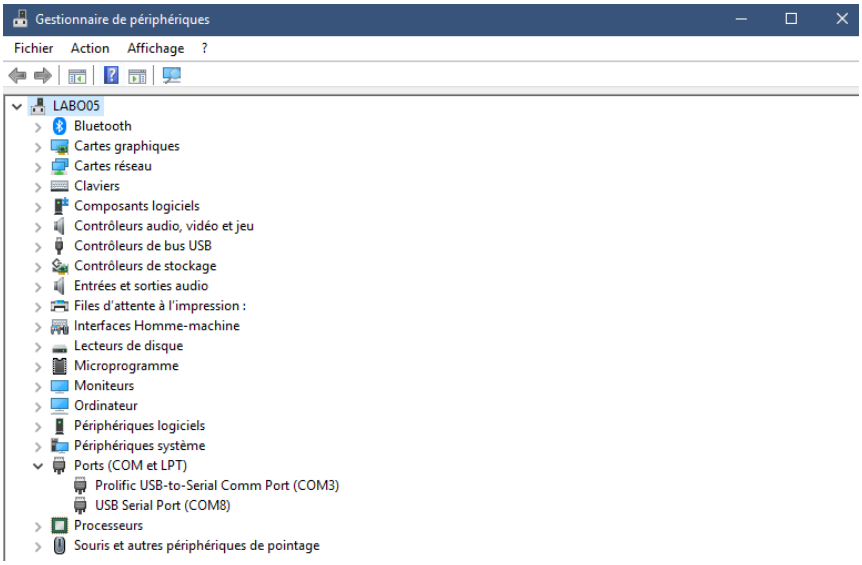
If you have only one USB cable, just unplug the USB Blaster and connect it to the HPS PS USB port (do not unplug the Power cable):



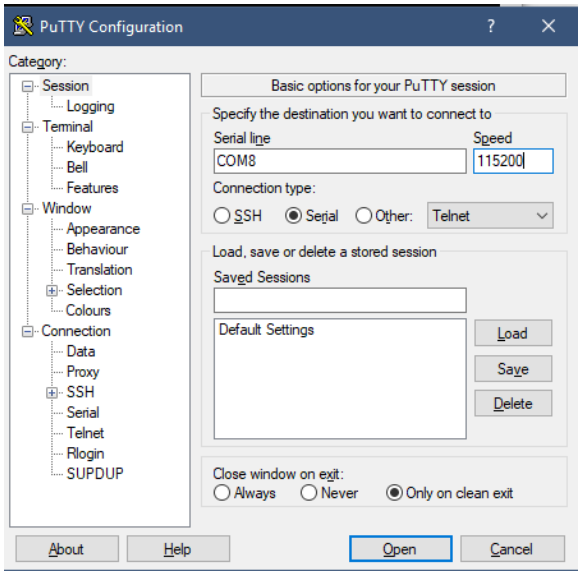
To program the processor you need to transfer your application program to your home directory in linux. The Linux OS is already installed on the SD card of the Board. To login you can do it using a USB

port connected to your PC (Serial Terminal using Putty) or the Ethernet cable (SSH Terminal using Putty). In the second case you also need the IP address. For that, just connect another USB port (at the opposite site of your board) to your PC. Then run Putty using the COM and 115200 baudrate.

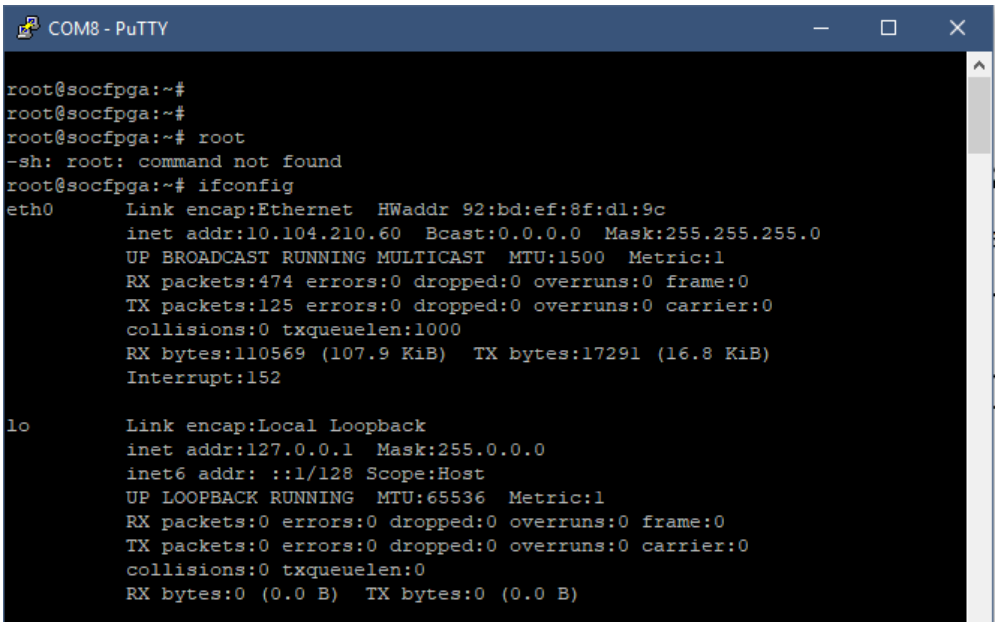
To see the port just start Gestionaire de Peripheriques (on windows) and see the ports (see the port name available COMXXX):



Then you can start Putty, Serial, Port name COMXXX, and baudrate 115200, and click Open.



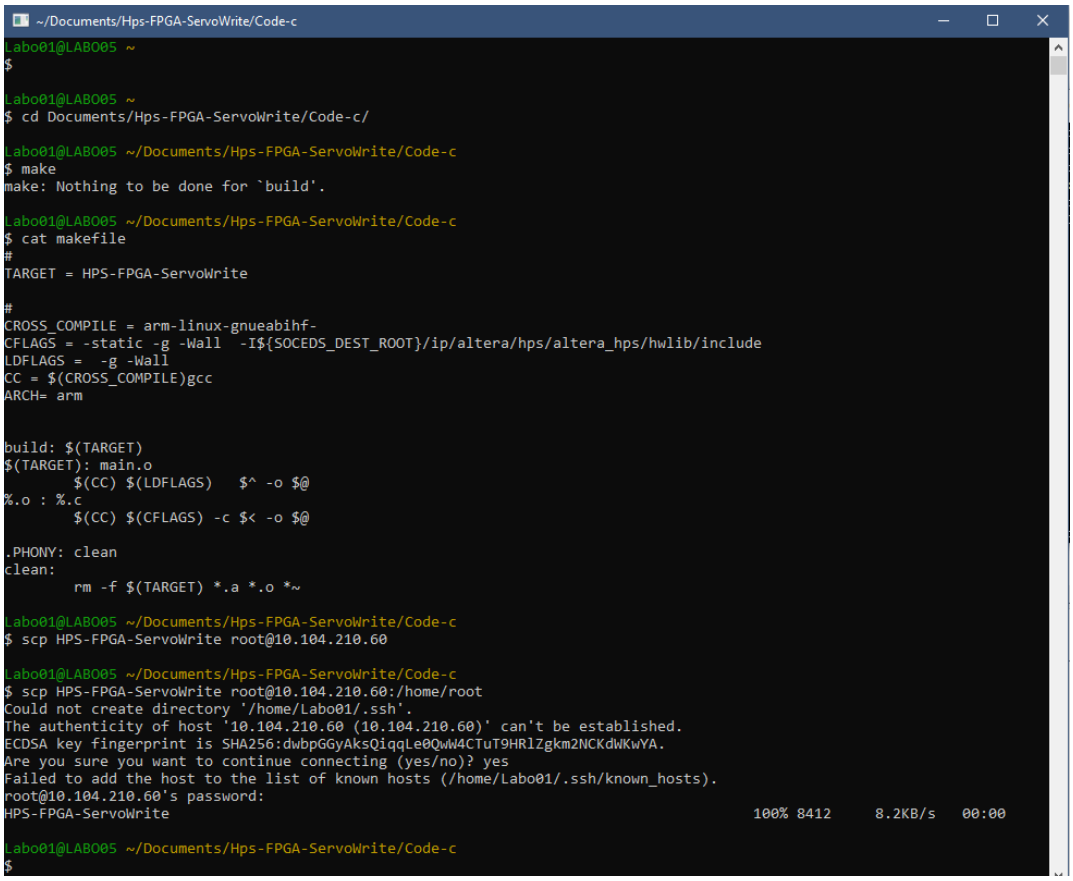
Type ENTER several times on the terminal to see the login. If the login doesn't appear, then press the reset button on the board and you will see the Linux reset then the login. In our case, we will login as root (password terasic, but it is not needed in Serial mode).



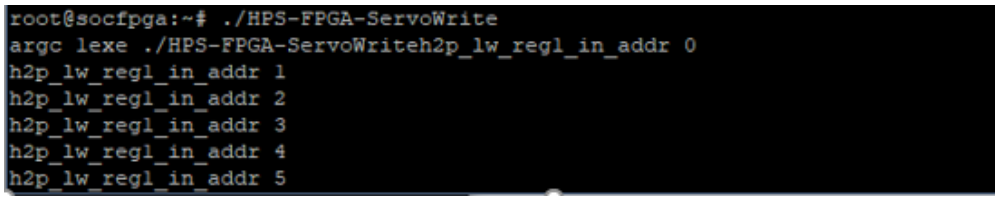
As you have the Ethernet cable connected to the board, you can now get the IP address by typing ifconfig. With that IP address, you can create an SSH connection login to transfer you application program to that address.

In the case that your application program is not available (check the files using ls), we will need to copy the executable using ssh copy. For that we will use the **SoC EDS shell**, that is used to compile your application program, but also to transfer the executable to your board. Just start EDS:

- Access the directory where the code is.
- Type **make** to see if you need to compile the program (sometimes the source code was modified and the executable must be updated). If the program is already updated, make will show the message Nothing to be done. Otherwise, the make will execute and compile the application program for the ARM processor.
- When done, the executable will be ready to be sent to the board. The generated program is names **HPS-FPGA-ServoWrite**, (you can see the name by typing **cat Makefile** and checking the target executable.
- Now that you know the IP address of your Linux and you have the executable, just copy the executable to your account on the board by using the command **scp 'nom_d_executable' root@'adresse_ip':/home/root**. In our case, **scp HPS-FPGA-ServoWrite root@10.104.210.60:/home/root.**



Once transferred, you just execute the application program. Just type the name of the application program in Putty.



The platform

Open platform designer to see the platform components, peripherals and I/O pins used. When the tool opens, you need to select the soc_system.qsys file, that was created for this project. A software register can be created by adding an external connection. That is a connection to the processor bus that is external. The external connection is connected to the Avalon bus as slave (the processors HPS is the master connected to be Avalon bus). We need registers to write values to the peripherals. To create another register just click on the + button and select PIO parallel processor peripheral. Click the Add button to configure the PIO.

To make it simple, just leave it as 8 bits output and click finish. Note the default address that will be used for your application program. Once the PIO created, you must add it to the Avalon bus master port available on the processor by click on the s1 Avalon slave line.

Go back to the System Contents and add the clock and reset signals. Just click on the unconnected message and the tool will propose clk_0 to be connected to. You must also add the reset signal. For

that, just click on the left side of the reset pin of your PIO.
Now you must Generate HDL and check for errors.
Pay attention to the Output directory path for the soc_system that it is preferable to be create in the same project directory you are using.