THE Ultimate Starter's Setup Guide

The first step is to download the provided project directory template from Moodle: http://moodle.epfl.ch/mod/folder/view.php?id=911830. Use 7zip (or any other compression software) to decompress the archive. **We highly encourage you to read the provided README carefully before continuing.** Then, rename the folder to *nios_introduction*.

Creating the Quartus project

1. Go to **File->New Project Wizard…**
    a. Choose *<project dir>/hw/quartus* as the working directory.
    b. Use *nios_introduction* as the project name.
    c. Click **Finish**.
2. We are using the DE1-SoC board. So we are going to execute a script to configure the FPGA family and the pin assignment.
    a. Go to **Tools->Tcl Scripts**…
    b. Select *pin_assignment_DE1_SoC.tcl*.
    c. Click **Run** and wait (Quartus might freeze for a while).
3. Then, we need to add the top-level entity of the project so Quartus can have a starting point to compile the design.
    a. Go to **Project->Add/Remove Files in Project…**
    b. Click on the button labelled **…**
    c. Select *<project dir>/hw/hdl/DE1_SoC_top_level.vhd* and click **Open**.
    d. Click **Add**. Note that the added path is relative from the Quartus project directory, i.e. *../hdl/DE1_SoC_top_level.vhd*.
    e. Click **Ok.**
    f. In the **Project navigator**, switch to the **Files** view.
    g. Right click on *../hdl/DE1_SoC_top_level.vhd* and click on **Set as Top-Level Entity**.

Creating a Qsys system

We will create a full system including a processor and memory capable of executing software that we are going to write in C.

1. Launch Qsys by going to **Tools->Qsys**.
2. In Qsys, go to **File->Save as** and save the Qsys system as *soc_system.qsys* under *<project dir>/hw/quartus.*
3. Use the **IP Catalog** to add the following components:
    a. A "Nios II Processor" to act as our main processor.
        i. Click **Finish** on the configuration view that just opened.
    b. An "On-Chip Memory (RAM or ROM)" to act as our main memory.
        i. Choose its **Total memory size** to be 256k and hit the tab key on your keyboard to autocomplete.
        ii. Click **Finish** on the configuration view.
    c. A "JTAG UART" to see the results of the standard output on your computer screen.
        i. Click **Finish** on the configuration view that just opened.
    d. A "PIO (Parallel I/O)" to toggle LEDs on the board.

i. Use a **Width** of 10 bits because the DE1-SoC has 10 leds.
ii. Select **Output** as the direction.
iii. Click **Finish** on the configuration view.

e. Now it's time to connect the system components together. Here are a few tips to do that correctly:
   i. Go to **System->Create Global Reset Network** to stop thinking about manually connecting the reset *everywhere*.
   ii. Connect the *clk* interface of the *clk_0* component to the *Clock input* interface of all the other components.
   iii. Connect the instruction bus (*instruction_master)* of the CPU to the on-chip memory.
   iv. Connect the data bus (*data_master)* of the CPU to all the other components.
   v. Connect the Interrupt Sender interface of the JTAG UART to the Interrupt Receiver interface of the CPU.

f. We need to export the *external_connection* of *pio_0* by double-clicking in the **Export** column and pushing ↵ on your keyboard. Recall: exporting a signal means making it available outside the Qsys system to route it on the board.

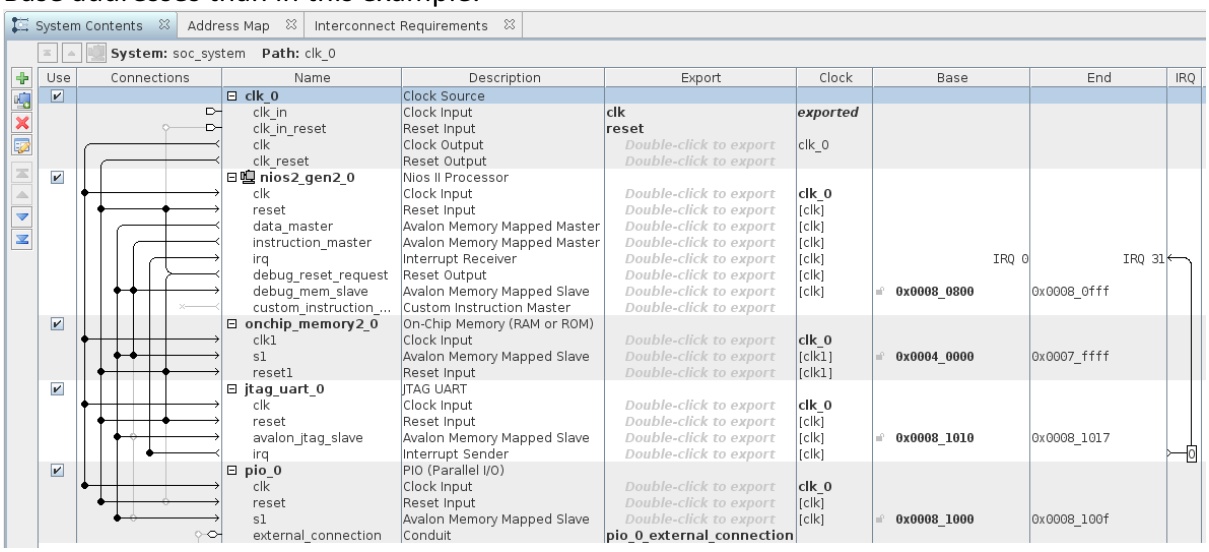
g. Double-click on the *nios2_gen2_0* component to open up its configuration view.
   i. Under the **Vectors** tab, select the on-chip memory to be its **Reset vector memory** and **Exception vector memory**.
      1. The **Reset vector offset** is the offset from the base address of the on-chip memory to which the CPU jumps upon reset.
      2. Similarly, the **Exception vector offset** is the offset to which the CPU jumps upon receiving hardware exceptions or traps.
      3. You can now close the configuration view.

h. Go to **System->Assign Base Addresses** to properly configure the address range of each component and avoid conflicts.

i. Similarly, go to **System->Assign Interrupt Numbers**.

j. Go to **File->Save**.

4. Your system should now look like the following – don't worry if you have different Base addresses than in this example:



5. You can now close Qsys by clicking on **Finish**. Do **not** generate the system when asked.
6. Use the same procedure you used to add the top-level VHDL file to the project to, this time, add the *<project dir>/hw/quartus/soc_system.qsys* file to the project.
7. You should now update your top-level to include your Qsys system.
    a. In a text editor, open *<project dir>/hw/quartus/soc_system/soc_system_inst.vhd* to get the VHDL component declaration and instantiation template of the Qsys system.
    b. Add the component declaration to your top-level architecture.
    c. Add the component instantiation and map the port as follows:
        i. The clock should be routed to CLOCK_50, the clock input pin of the FPGA/SoC device.
        ii. Use the button KEY_N(0) as the reset signal of your design.
        iii. Route the PIO to LEDR, the pin connected to the LEDs on the board.
    d. Then, in the entity, comment all the ports you are not using, i.e. everything except the 3 mentioned above. Be careful with your semi-colons!
8. Go to **Processing->Start Compilation** and grab a coffee/tea or even a pineapple juice, we are open-minded.
9. Once the compilation is finished, plug in your FPGA board and go to **Tools->Programmer.**
    a. Click **Auto-detect** and select **5CSEMA5**.
    b. Right-click on the beautiful picture of Altera chip labelled **5CSEMA5** and select **Change File…**
    c. Select *<project dir>/hw/quartus/output_files/nios_introduction.sof.*
    d. Enable the "Program/Configure" checkbox for device **5CSEMA5F31**.
    e. Press **Start**.

It's software party time!

1. It's now time to launch our beloved software IDE, the *Nios II SBT*.
    a. Launch the **Nios II Command Shell** from the Start menu of your Windows machine.
    b. Use the following command "eclipse-nios2 &" to launch the IDE.
2. Go to **File->New->Nios II Application and BSP from Template**.
3. Select *<project dir>/hw/quartus/sco_system.sopcinfo* as **SOPC Information File name**.
4. Name your software project *nios_introduction*.
5. We invite your to uncheck the **Use default location** checkbox and choose *<project dir>/sw/nios/application*. We encourage this practice to properly separate software from hardware design files.
6. Choose **Hello World** as the **Project template**.
7. Click **Finish.**
8. You can now write/compile/run your software and have a lot of fun with those LEDs. Recall: you will want to include the *io.h* and *system.h* file to get some useful macros.

For more information, you can check the DE1-SoC tutorial we provide in the *SoC-FPGA Design Guide* we provide on Moodle:
http://moodle.epfl.ch/pluginfile.php/1606624/mod_resource/content/6/SoC-FPGA%20Design%20Guide%201.05.pdf.