h e p i a

Haute école du paysage, d'ingénierie
et d'architecture de Genève

A. Guerrieri

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences
Western Switzerland

Complete system setup

Introduction

For the second part of the course, we are going to focus on optimization techniques for the Sobel algorithm implementation.

In order to speed-up the development process and get focused into the optimization part, we provide you a working hardware-software system that you can download and test.

This short tutorial will show you how to setup the complete system to get it ready to run.

1 Download .zip files

Get the files *Hardware description file of the example system* and *Software required for the example system* from the Moodle's website. Unzip them and have a look at the contents.

2 Hardware system

As usual, we will start generating the bitstream file to program the FPGA. As we will use the same folder for the complete project, rename the folder where you unzipped the *hardware.zip* file to **mse_demo**. Create an empty folder named *software* inside (see Fig. 1).

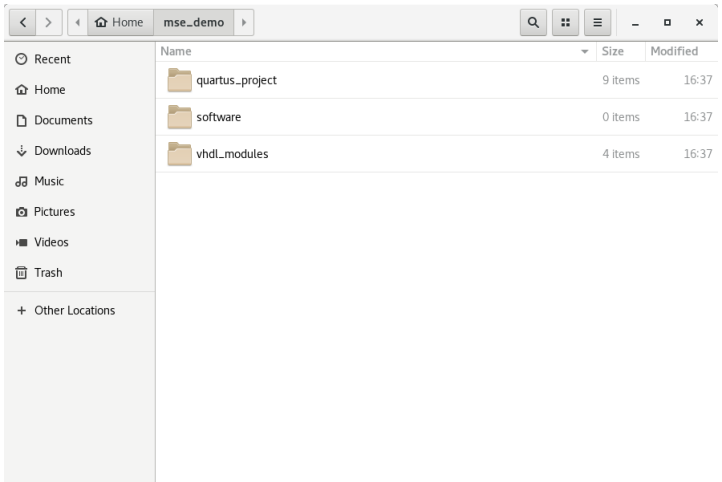

FIGURE 1 – *mse_demo* folder structure.

Start Quartus and select "Open project" in the "File" menu. Open the project file (*mse_demo.qpf*) found in *mse_demo/quartus_project/*

2.1 Part selection

Change the target device by opening "Assignments"→"Device" or right-click in the **device** in "Project Navigator" and selecting "Device". Then, choose the right part number (*EP4CE30F23C7*) as in Fig. 2 and click "OK".

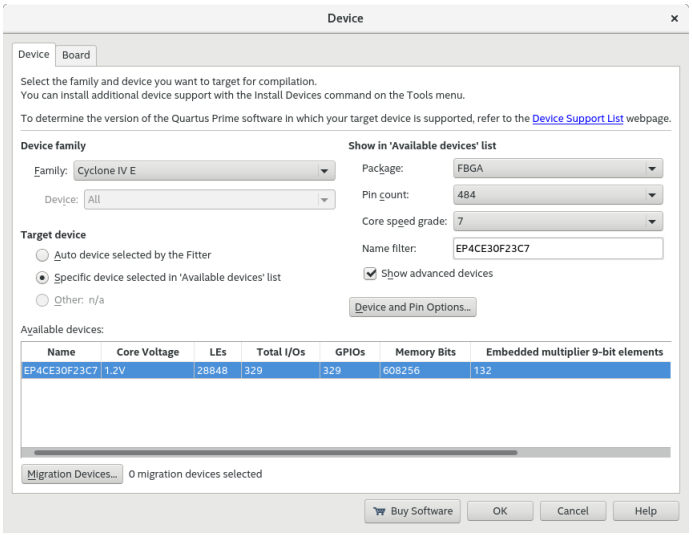

FIGURE 2 – Part number selection (*EP4CE30F23C7*).

2.2 Include files

Include the Qsys file in the project by opening "Assignments"→"Settings" or right-click in the **project** in "Project Navigator" and selecting "Settings". In the "Files" tab, click on the 3 dots (...) on the right side of "File name" and navigate to the file *base_system.qsys* (in *mse_demo/quartus_project/*). Add also the *constraints.sdc* file located in *mse_demo/quartus_project/*. Check that both files appear in the lower list (you may need to click "Add" to do so), click on "Apply" and then "OK".

Optional but recommended : In the "Settings" menu, Category : "Compilation Process Settings", you can select a different folder to save the output files : "Save project output files in specified directory". You can leave the default value *output_files* in the directory name. This may help you to find the bitstream file later on.

Now, open Qsys using "Tools"→"Platform designer" as usual. It will ask for the *base_system.qsys* file that we specified in the previous step. Have a look at the system setup and click on "Generate HDL" as in previous exercises. You can optionally select "VHDL" as language for the generated design files, if you want to.

2.3 Pin assignment

As we did in the previous exercises, we have to assign to the nets defined in our design, a physical pin in the device. To do so, we will run a Tcl script selecting "Tools"→"Tcl scripts". In the "Project" folder of the design, you will find the *pins.tcl* file to do the assignment. Select the script and click on "Run".

2.4 Bitstream generation and FPGA programming

Now, our hardware project is ready to be implemented, we can click on "Start compilation", as we have been doing so far, and wait for the bitstream generation to be completed.

Then, we can program our FPGA using the "Programmer", remembering not to close the pop-up window that opens when the device is programmed.

3 Software system

Open Eclipse as you have been doing in the previous exercises (either from Quartus "Tools"→"Nios II Software Build Tools for Eclipse" or from the menu : "Applications"→"Programming"→"Eclipse").

3.1 Application creation

Create a new application with "File"→"New"→"Nios II Application and BSP from Template". In the configuration window, select :

- SOPC file : *mse_demo/quartus_project/base_system.sopcinfo*
- Project name : *sobel*
- Project location : *mse_demo/software/sobel*
- Project template : *Blank Project*

You can double-check the configuration with Fig. 3.

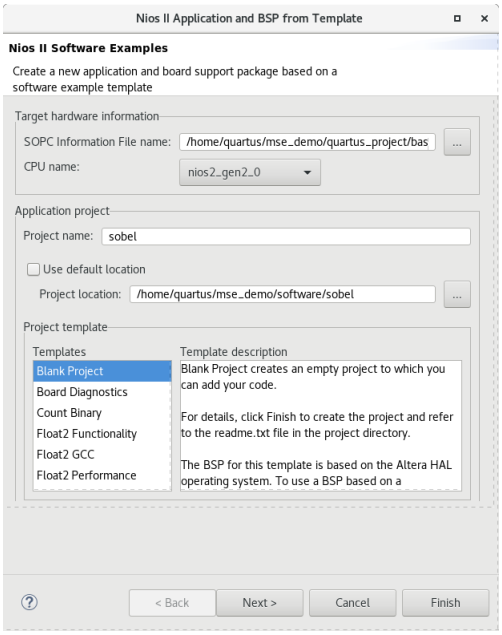

FIGURE 3 – Project configuration.

Click "Next", and, in the BSP configuration, leave the default values and click "Finish".

3.2 Project configuration

In the *mse_demo/software/sobel* folder, you will find the generated files of you software application. Unzip the contents of the *base_system.zip* file that you got from the Moodle website in this folder. You should find all the *.c* and *.h* files in *mse_demo/software/sobel*.

Get back to Eclipse and right-click on the "sobel" project. Select "Refresh". This will update the

Eclipse project with the new files. You can check that they have been correctly included in the hierarchy.

Right-click also in the *sobel_bsp* project and select "Nios II"→"BSP editor". In the "Main" tab, "Settings", "Common", "hal", select "Systimer" as "sys_clk_timer" and "ProfileTimer" as "timestamp_timer", as in Fig. 4. You can check the "Nios II Classic Software Developer's Handbook" if you want to know the details of the "sys_clk_timer" and "timestamp_timer". Then, click on "Generate" and "Exit".

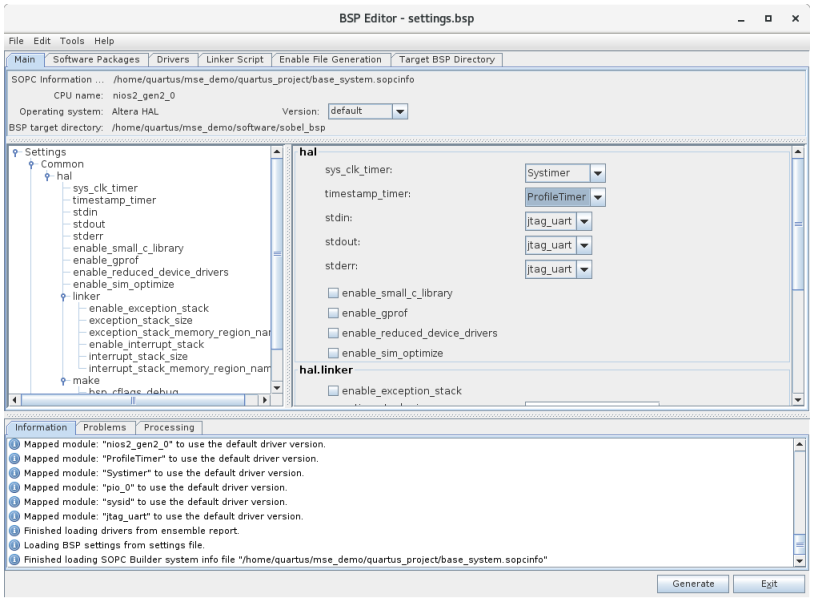

FIGURE 4 – BSP editor.

3.3 Run application and test

Now you can finally build the sobel application and run it in the board as you have done in previous exercises. If you have timestamp or System ID mismatch errors when trying to run the application, you can disable these checks in the "Target Connection" tab of the "Run Configurations" menu.

4 Code inspection

As your system is up and running, you can now check what you just did!

4.1 Reset button

If you look at the pinout definitions in the previous exercises and this one, you will realize which button is doing a system reset. How is it connected?

Try not to find out by trial and error, you will not learn much from that.

4.2 Mode selection

There are 5 modes of operation in the example. How does the user select them? Could you list the configurations with respect to the mode selector values?

The VGA output is also selectable by the user. How is it done ? When is the configuration checked and applied ?

4.3 Data flow

Check the code to see how the data circulates in the main loop within the different parts of the system. Could you make a rough list of VHDL modules and C functions (and files) with the data flow for each image ? Start with the simplest mode and then do the same with the one involving more processing elements.

What happens when the VGA is involved ?

5 Optional : Scripting, versioning

If you have time, you could investigate how to start automating the development process. For example : programming the bitstream in the FPGA, generating the BSP file, compiling the software and launching it in the board, solving JTAG connection problems, interfacing the Nios II console through JTAG, debugging, etc. Perhaps the "Nios II Command-Line Tools" document could give you some hints at this respect. **Note** : In the Virtual Machine, all Nios II related commands must be called using the *intelFPGA_lite* wrapper that is available in the *PATH*.

Using a version control system is also recommended. Once it is correctly set-up, you can easily get back to a certain version of your system and compare the differences with a newer version that may have bugs on it. It may save you time when trying to debug your system.

Note that these features are not substituting the procedure we have been using so far. They should be used as a complement to our development process.