Lecture 7

Design of Embedded Hardware and Firmware

Make the system work

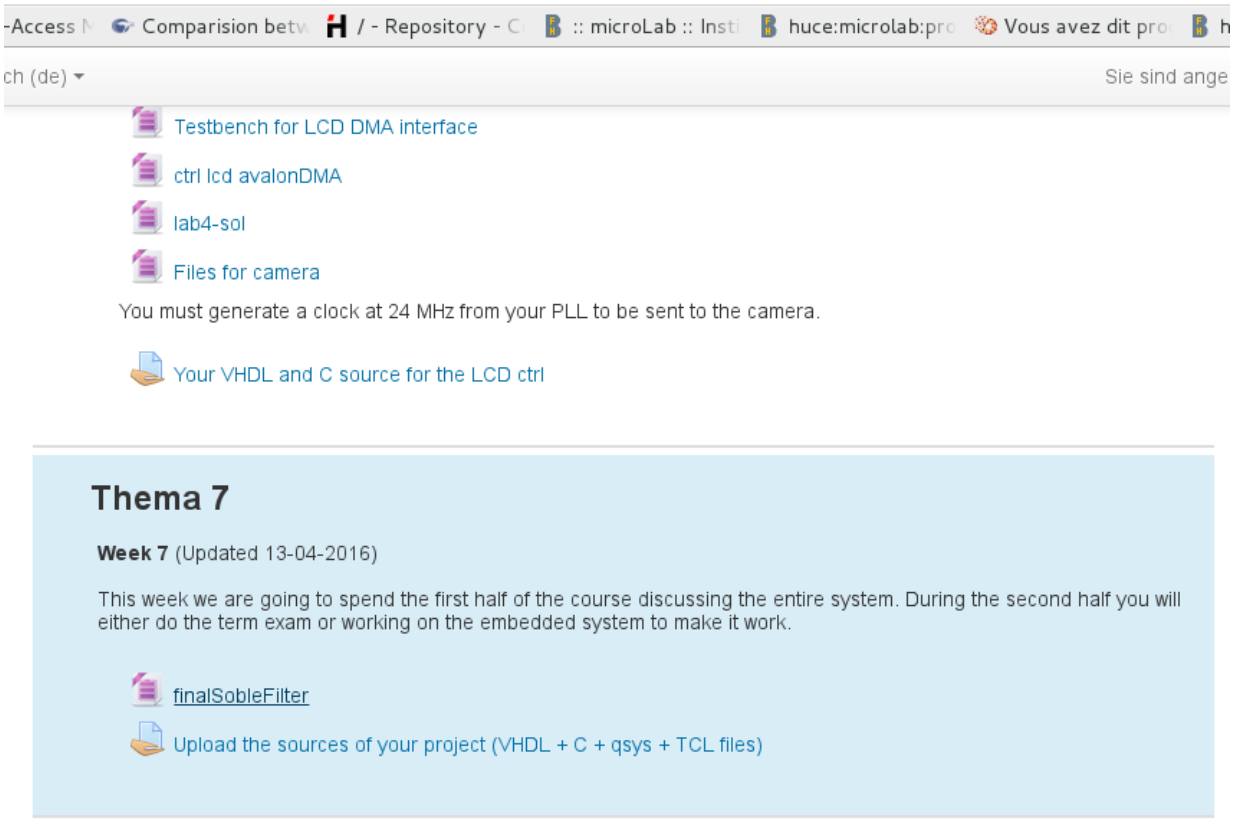
TSM_EmbHardw
Apr. 13, 2016

Prof. A. Habegger
Bern University of Applied Sciences

The Goals

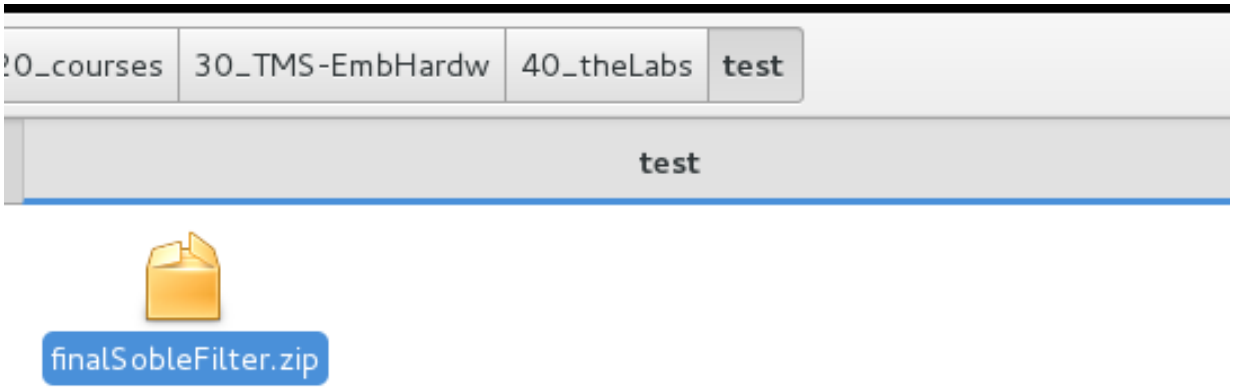
- Everyone knows the steps invoked to make the system work
- The SoPC works on every board
- You test the entire system including the firmware
- Among of different VHDL background, it is clear how we extend the embedded system
  - Memory Map
  - Avalon MM Slave and MM Master
  - Interfacing the cores from within FW
  - Using the user IP API

Get the Files from Moodle



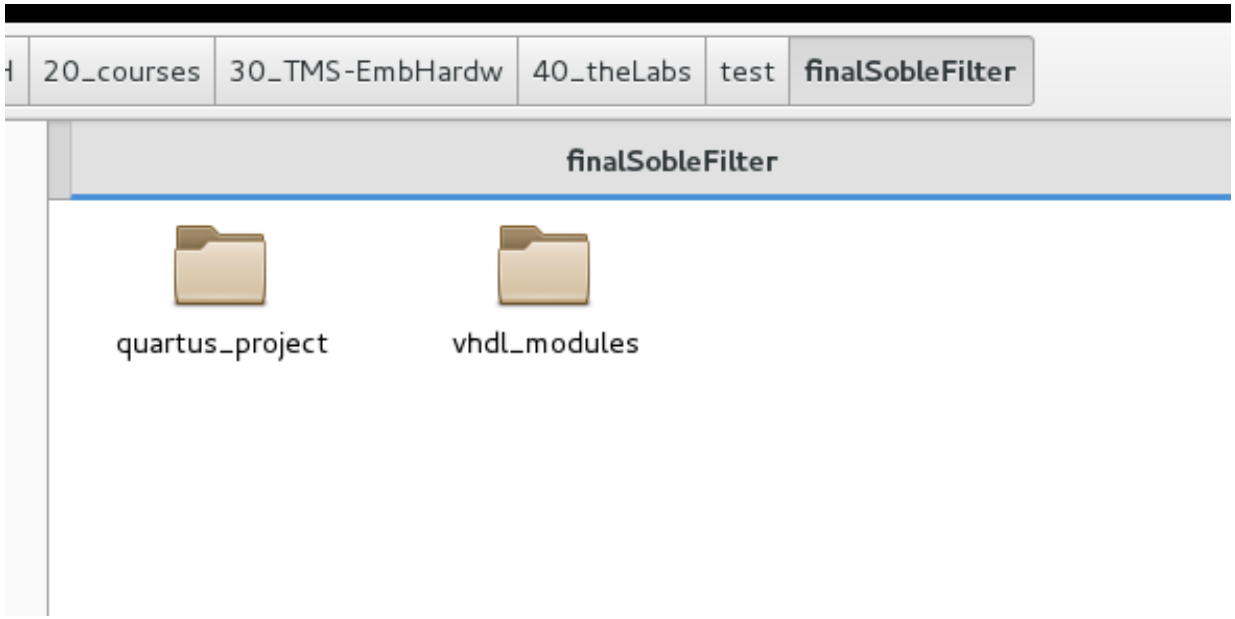
- Login to Moodle and download the ZIP file. (*finalSobelFilter*)
- Save the ZIP somewhere on yours system disk

Unpack the archive



- Navigate to the location where the files are stored
- Unpack the ZIP archive

Unpack the archive



- Navigate to the location where the files are stored
- Unpack the ZIP archive
- The unpacked archive will look as shown in figure above

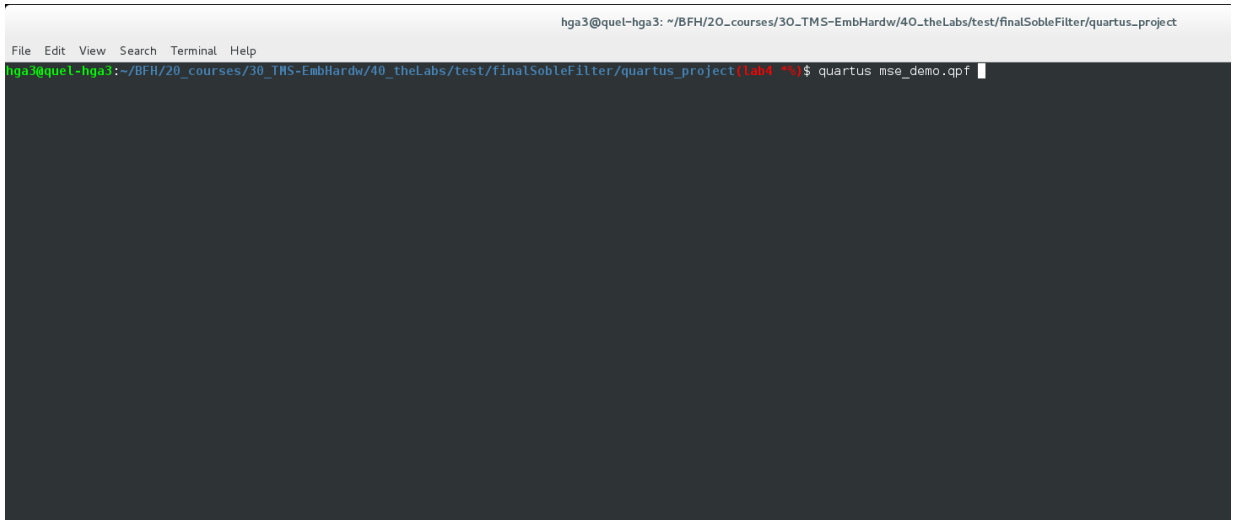
Browse the Folders

- Open a terminal and switch to the location where the files are (`cd $LOCATION`).
- The following files are in the sub-folder *vhdl_modules* and *quartus_project*, respectively

```
└── vhdl_modules
    ├── camera_controller
    │   ├── cam_avalon_slave_behavior.vhdl
    │   ├── cam_avalon_slave_entity.vhdl
    │   ├── cam_dma_behaviour.vhdl
    │   ├── cam_dma_ctrl_behaviour.vhdl
    │   ├── cam_dma_ctrl_entity.vhdl
    │   ├── cam_dma_entity.vhdl
    │   ├── frame_interpreter_behavior.vhdl
    │   ├── frame_interpreter_entity.vhdl
    │   ├── pixel_interface_behavior.vhdl
    │   ├── pixel_interface_entity.vhdl
    │   ├── synchroflop_behavior.vhdl
    │   └── synchroflop_entity.vhdl
    ├── i2c_core
    │   ├── i2c_autodetect_behavior.vhdl
    │   ├── i2c_autodetect_entity.vhdl
    │   ├── i2c_avalon_slave_behavior.vhdl
    │   ├── i2c_avalon_slave_entity.vhdl
    │   ├── i2c_cntrl_behavior.vhdl
    │   ├── i2c_cntrl_entity.vhdl
    │   ├── i2c_data_behavior.vhdl
    │   ├── i2c_data_entity.vhdl
    │   ├── i2c_start_stop_behavior.vhdl
    │   └── i2c_start_stop_entity.vhdl
    ├── lcd_controller
    │   ├── dma_controller_behavior.vhdl
    │   ├── dma_controller_entity.vhdl
    │   ├── lcd_avalon_slave_behavior.vhdl
    │   ├── lcd_avalon_slave_entity.vhdl
    │   ├── lcd_dma_behavior.vhdl
    │   ├── lcd_dma_entity.vhdl
    │   ├── pixel_formatter_behavior.vhdl
    │   ├── pixel_formatter_entity.vhdl
    │   ├── send_receive_if_behavior.vhdl
    │   └── send_receive_if_entity.vhdl
    └── vga_controller
        ├── delay_line_behavior.vhdl
```

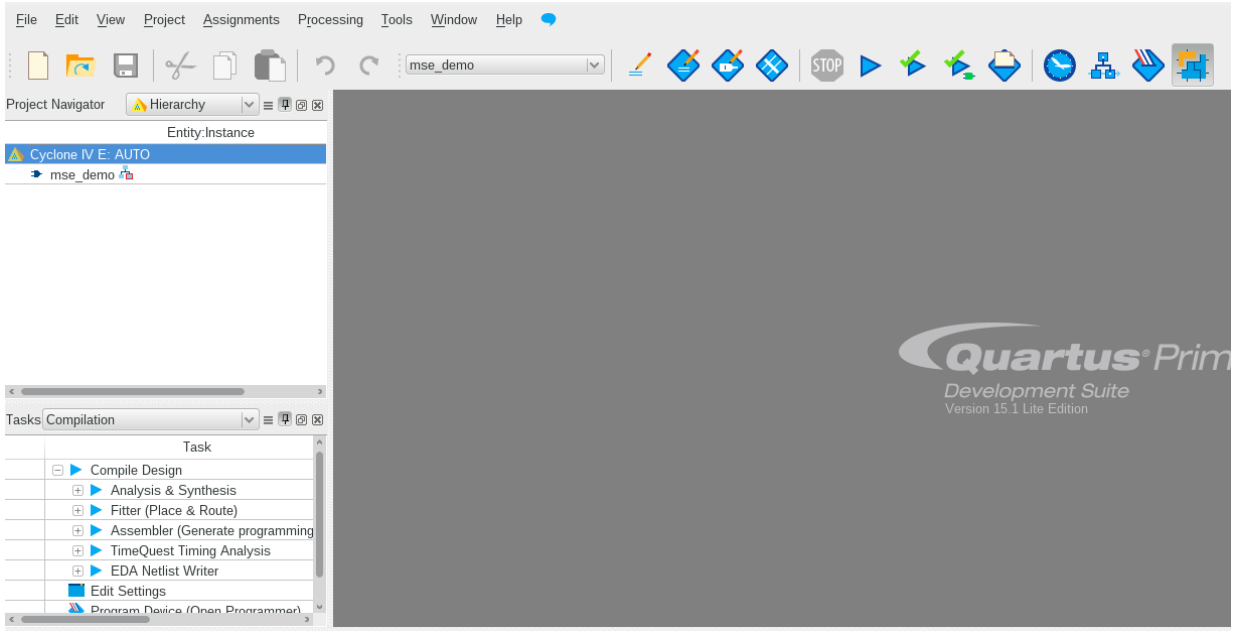
```
└── quartus_project
    ├── base_system.qsys
    ├── base_system.sopcinfo
    ├── cam_dma_hw.tcl
    ├── i2c_master_hw.tcl
    ├── lcd_dma_hw.tcl
    ├── mse_demo.bdf
    ├── mse_demo.qpf
    ├── pins.tcl
    └── software
        ├── sobel
        │   ├── create-this-app
        │   ├── Makefile
        │   ├── readme.txt
        │   └── src
        │       ├── camera.c
        │       ├── camera.h
        │       ├── dipswitch.c
        │       ├── dipswitch.h
        │       ├── grayscale.c
        │       ├── grayscale.h
        │       ├── i2c.c
        │       ├── i2c.h
        │       ├── lcd_simple.c
        │       ├── lcd_simple.h
        │       ├── main.c
        │       ├── mt9d112.h
        │       ├── sobel.c
        │       ├── sobel.h
        │       ├── vga.c
        │       └── vga.h
        └── sobel_bsp
            ├── create-this-bsp
            ├── Makefile
            └── settings.bsp
```

Run Quartus



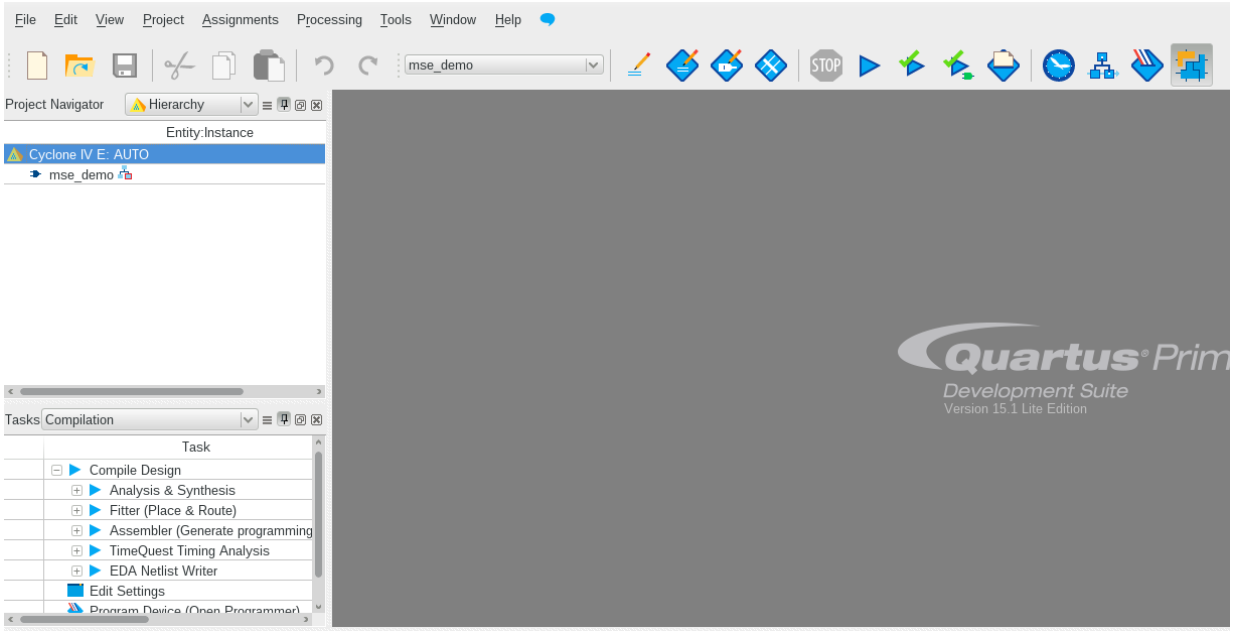
- Run quartus. To do so run the command *altera_shell* first
- If the command is not available run the file *nios2_command_shell.sh*, which is located in altera installation folder ↪ nios2eds
- Run quartus with the argument $PROJ_NAME.qps, where $PRO_NAME is the projects name.
- More details are shown in figure above.

Run Quartus



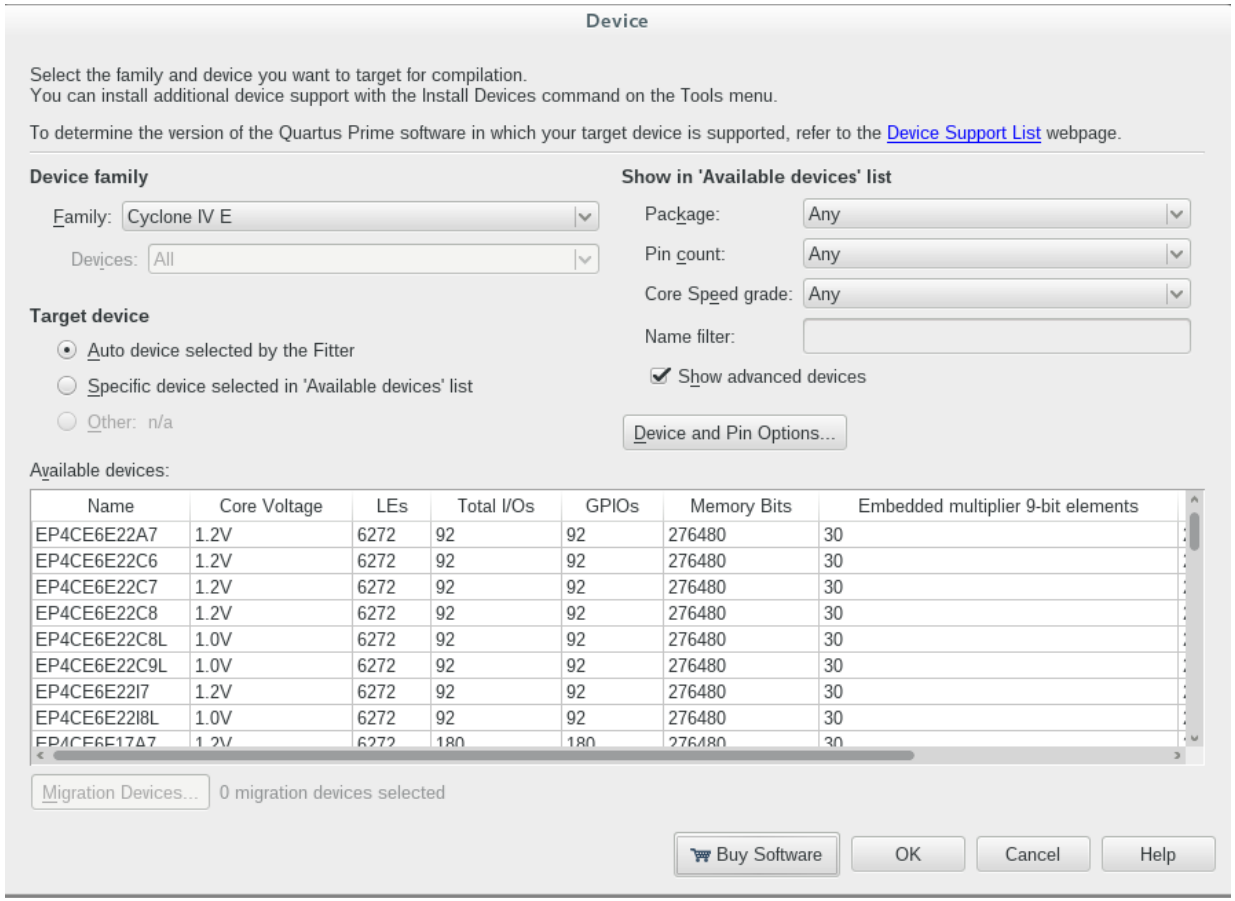
- Run quartus. To do so run the command *altera_shell* first
- If the command is not available run the file *nios2_command_shell.sh*, which is located in altera installation folder ↪ nios2eds
- Run quartus with the argument $PROJ_NAME.qps, where $PRO_NAME is the projects name.
- More details are shown in figure above.

Apply Correct Device Settings



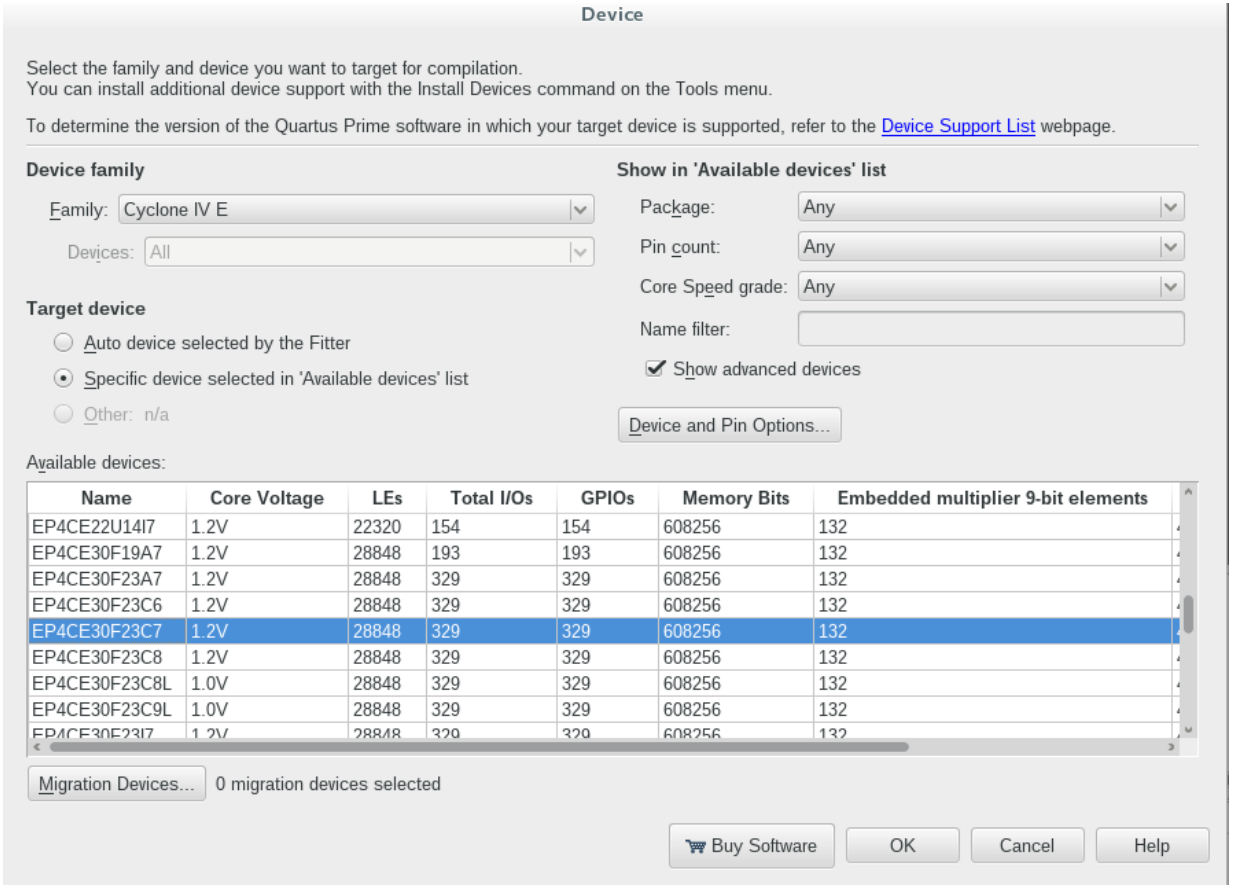
- Apply correct decive settings.
- Right click on the instance and tick settings

Apply Correct Device Settings



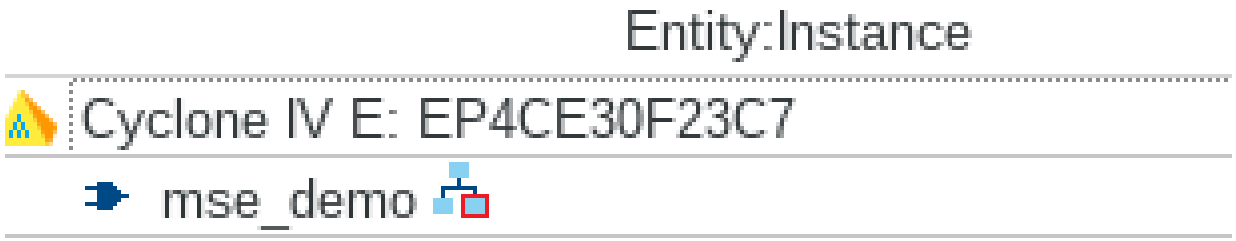
- Apply correct decive settings.
- Right click on the instance and tick settings
- A overview of all available device configurations is listed

Apply Correct Device Settings



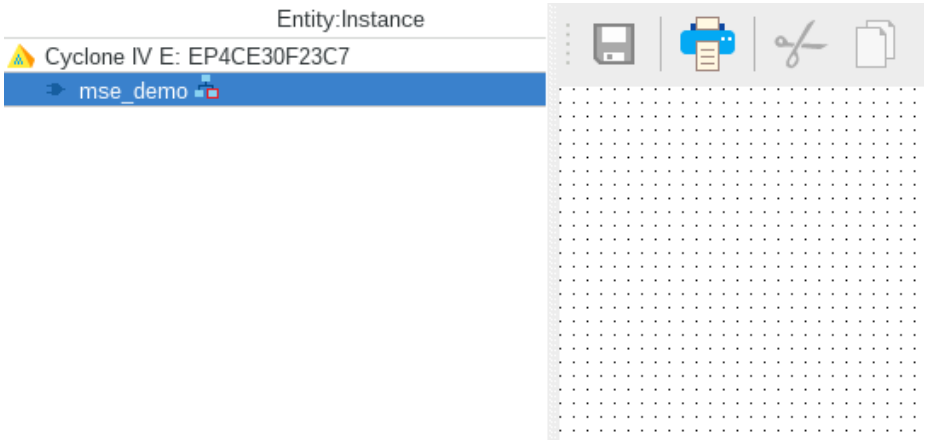
- Apply correct decive settings.
- Right click on the instance and tick settings
- A overview of all available device configurations is listed
- Select the device mounted on the board (EP4CE30F237)
- Confirm the settings by click on OK button

Apply Correct Device Settings



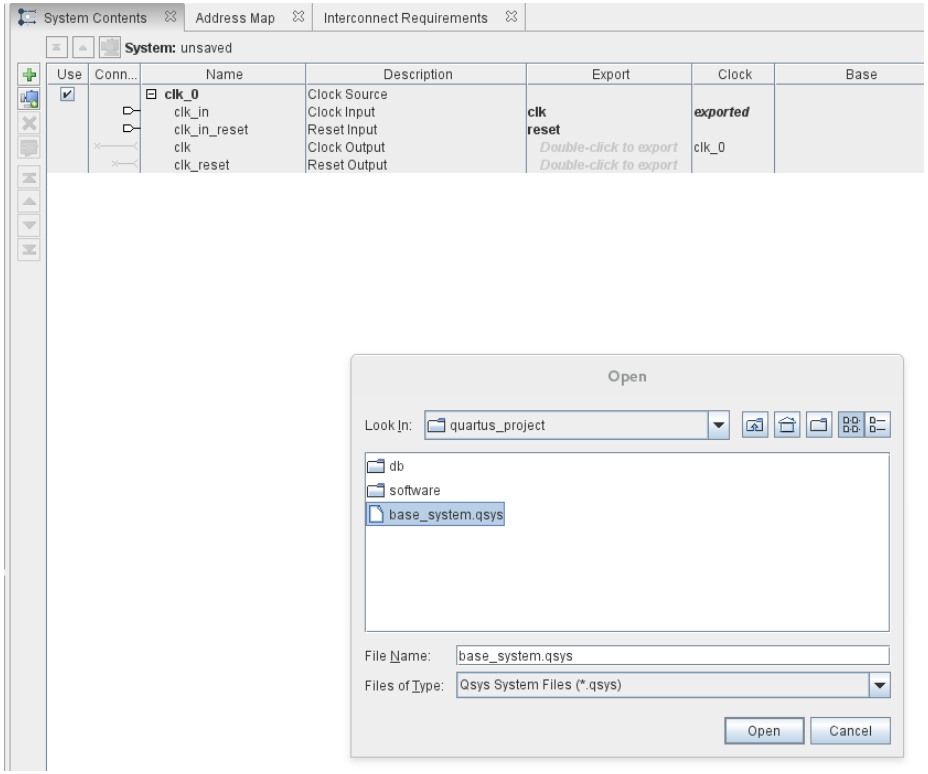
- Check if correct device setting was applied
- Click on the project named “mse_demo” to open

Apply Correct Device Settings



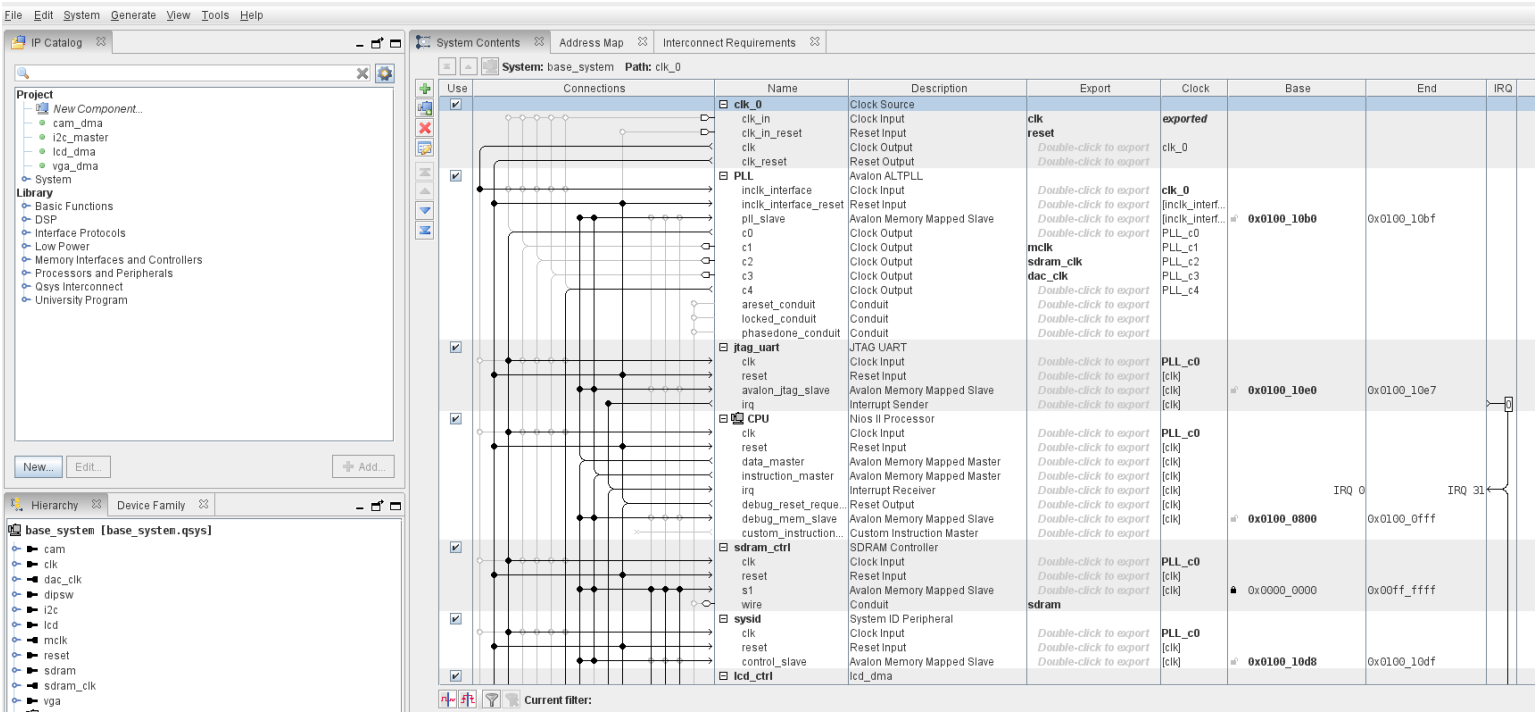
- The top system block is shown in figure above
- Pins are not assigned yet
- Open QSys
- Select the SoPC configuration file (base_system.qsys)

Apply Correct Device Settings



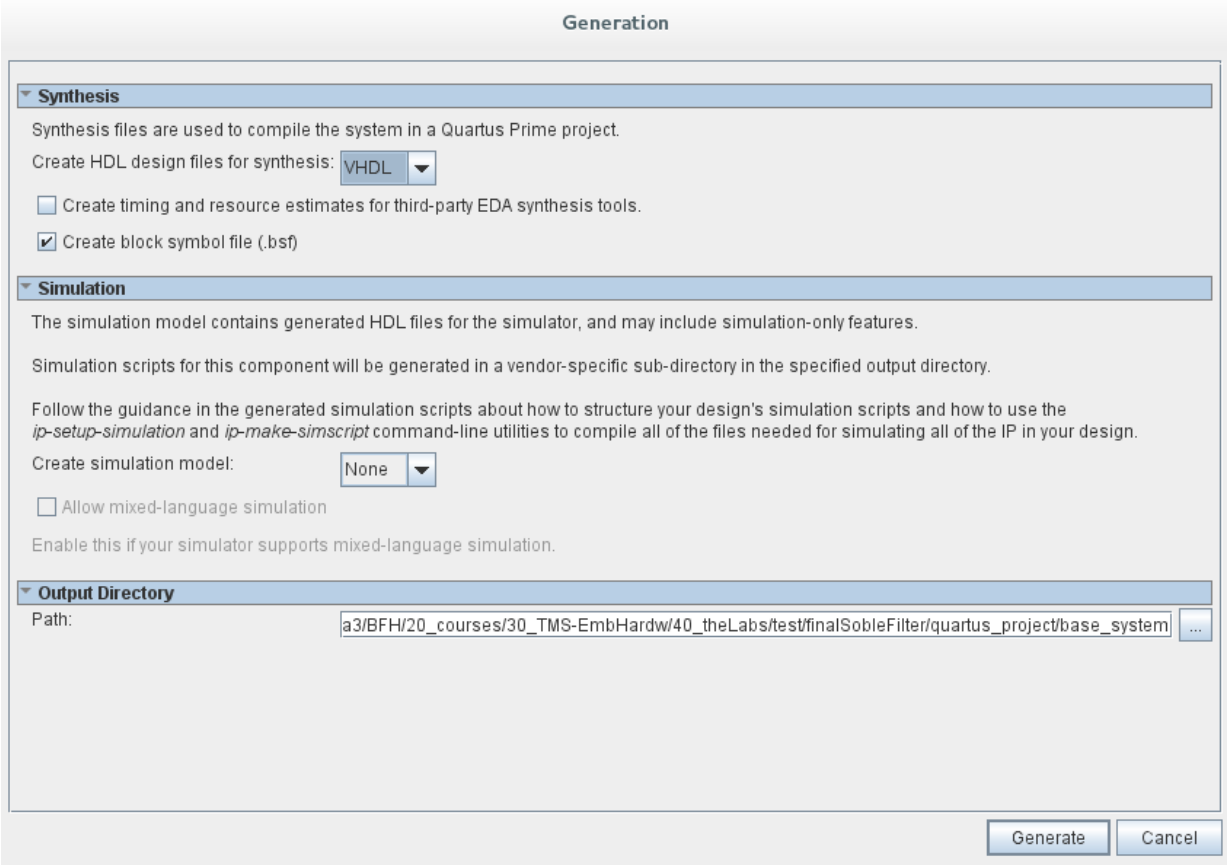
- The top system block is shown in figure above
- Pins are not assigned yet
- Open QSys
- Select the SoPC configuration file (base_system.qsys)

Build the SoPC



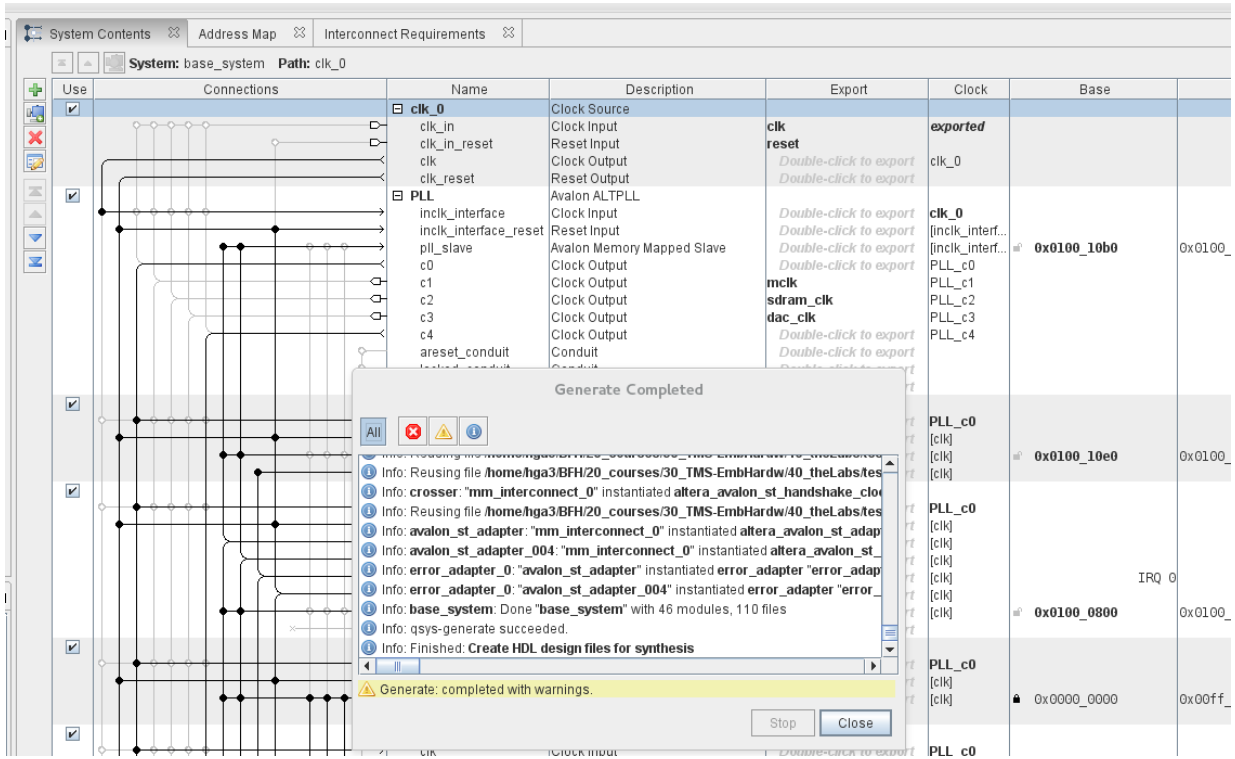
- Compare it loaded system looks like the one shown in figure
- Click on the button `Generate HDL...`
- Click `Generate`
- Wait...
- Switch back to Quartus

Build the SoPC



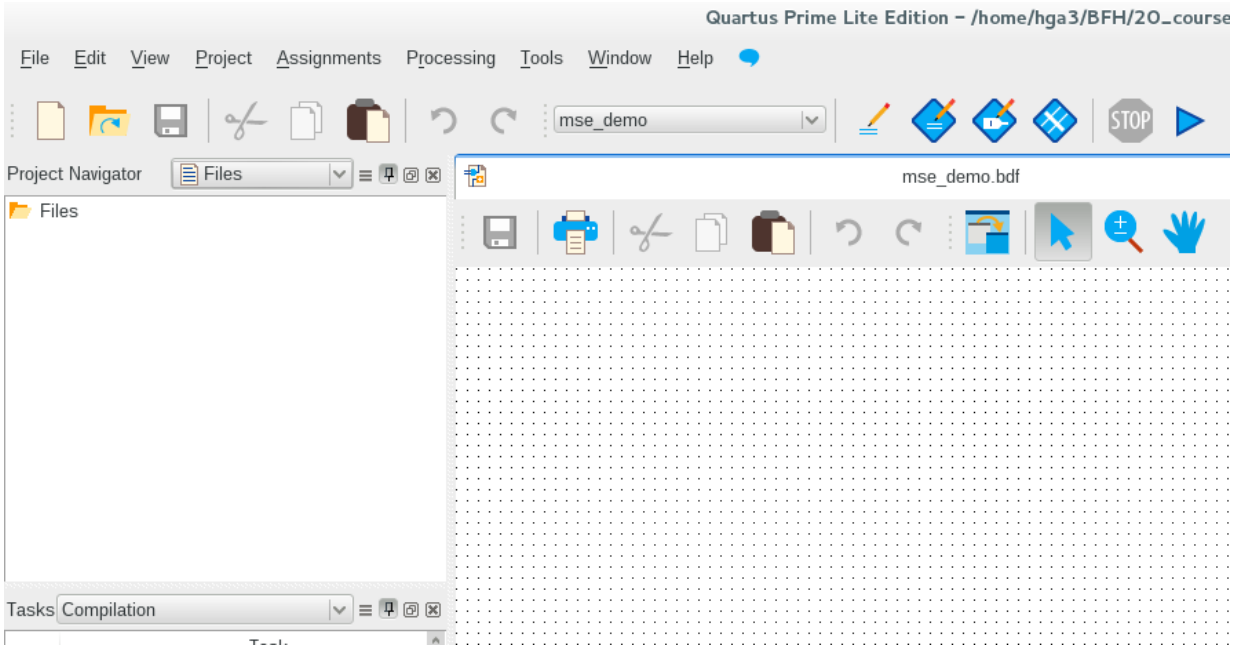
- Compare it loaded system looks like the one shown in figure
- Click on the button `Generate HDL...`
- Click `Generate`
- Wait...
- Switch back to Quartus

Build the SoPC



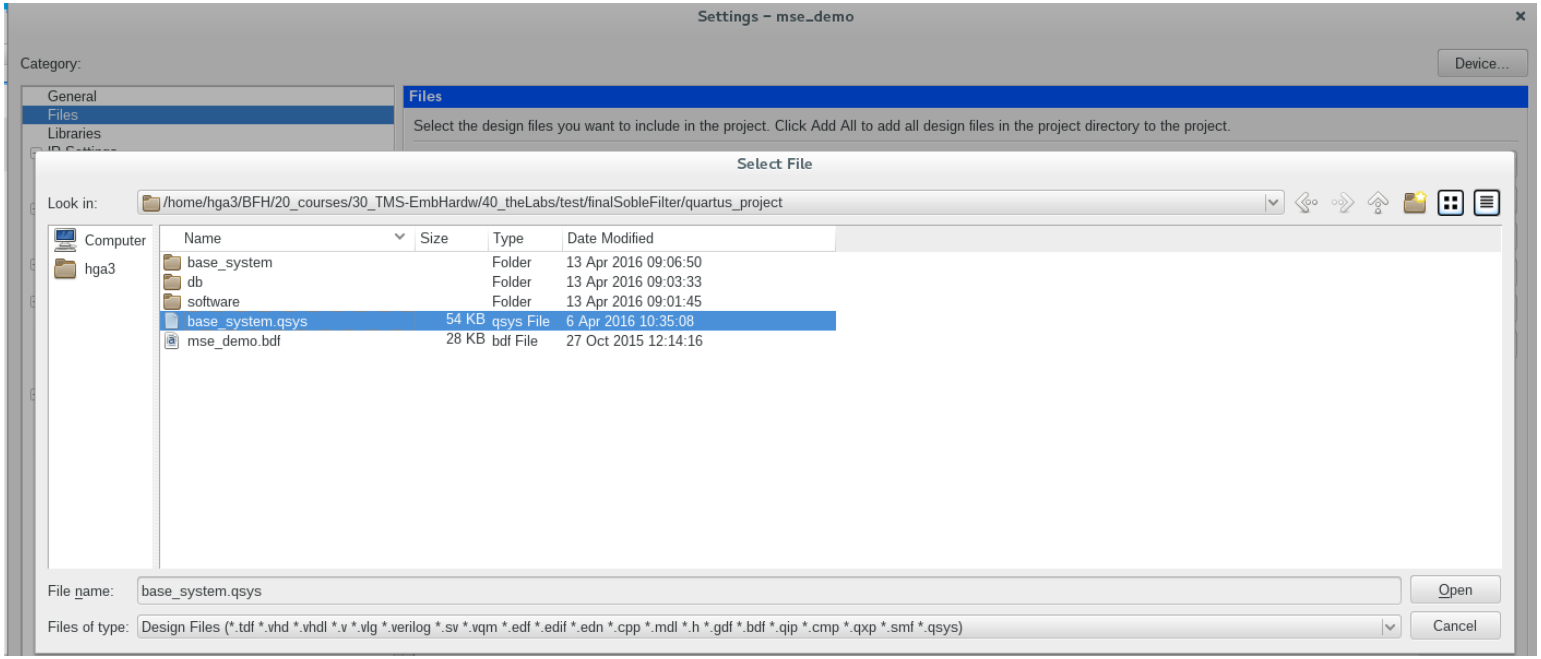
- Compare it loaded system looks like the one shown in figure
- Click on the button `Generate HDL...`
- Click `Generate`
- Wait...
- Switch back to Quartus

Add the Files



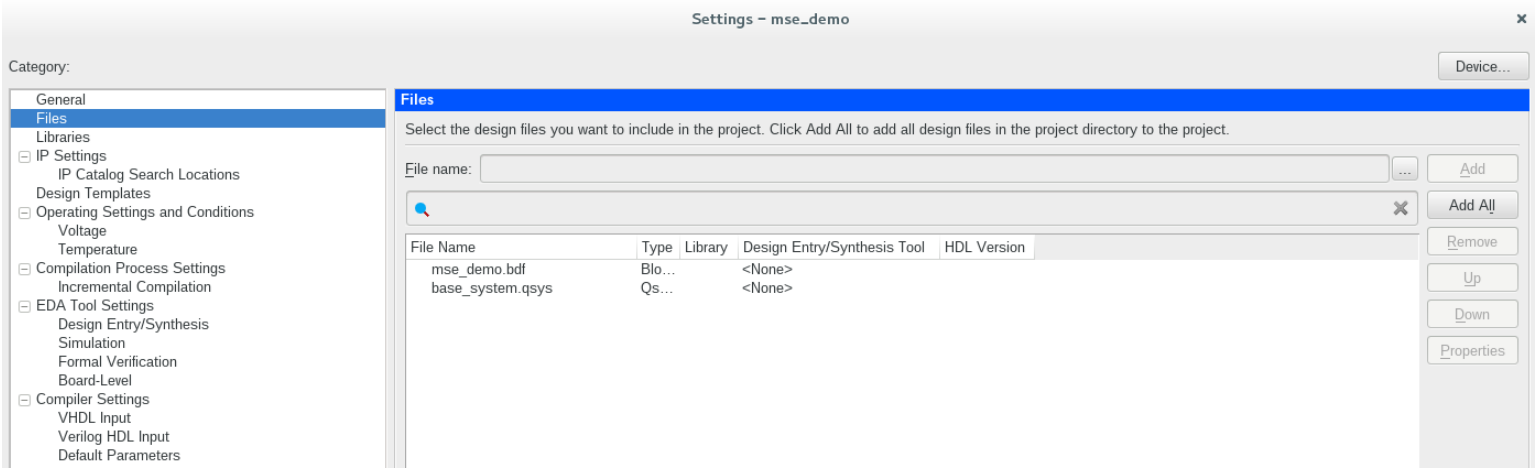
- Switch to `Files` in project navigator
- Add additional files (right click and select `add...`)
- Select the ***.qsys** and **mse_demo.bdf** files
- Add both files (one after the other...)
-

Add the Files



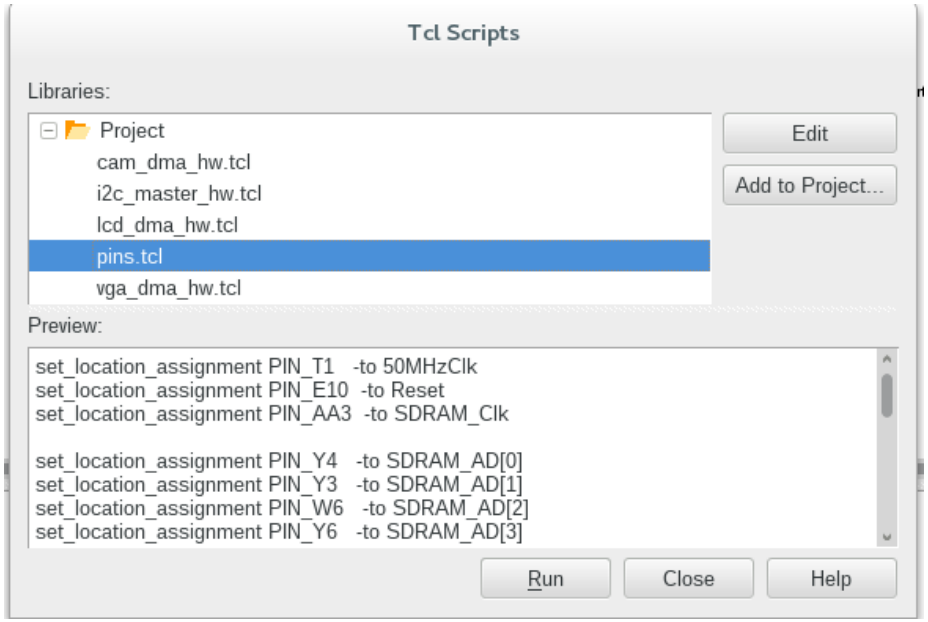
- Switch to `Files` in project navigator
- Add additional files (right click and select `add...`)
- Select the ***.qsys** and **mse_demo.bdf** files
- Add both files (one after the other...)
-

Add the Files



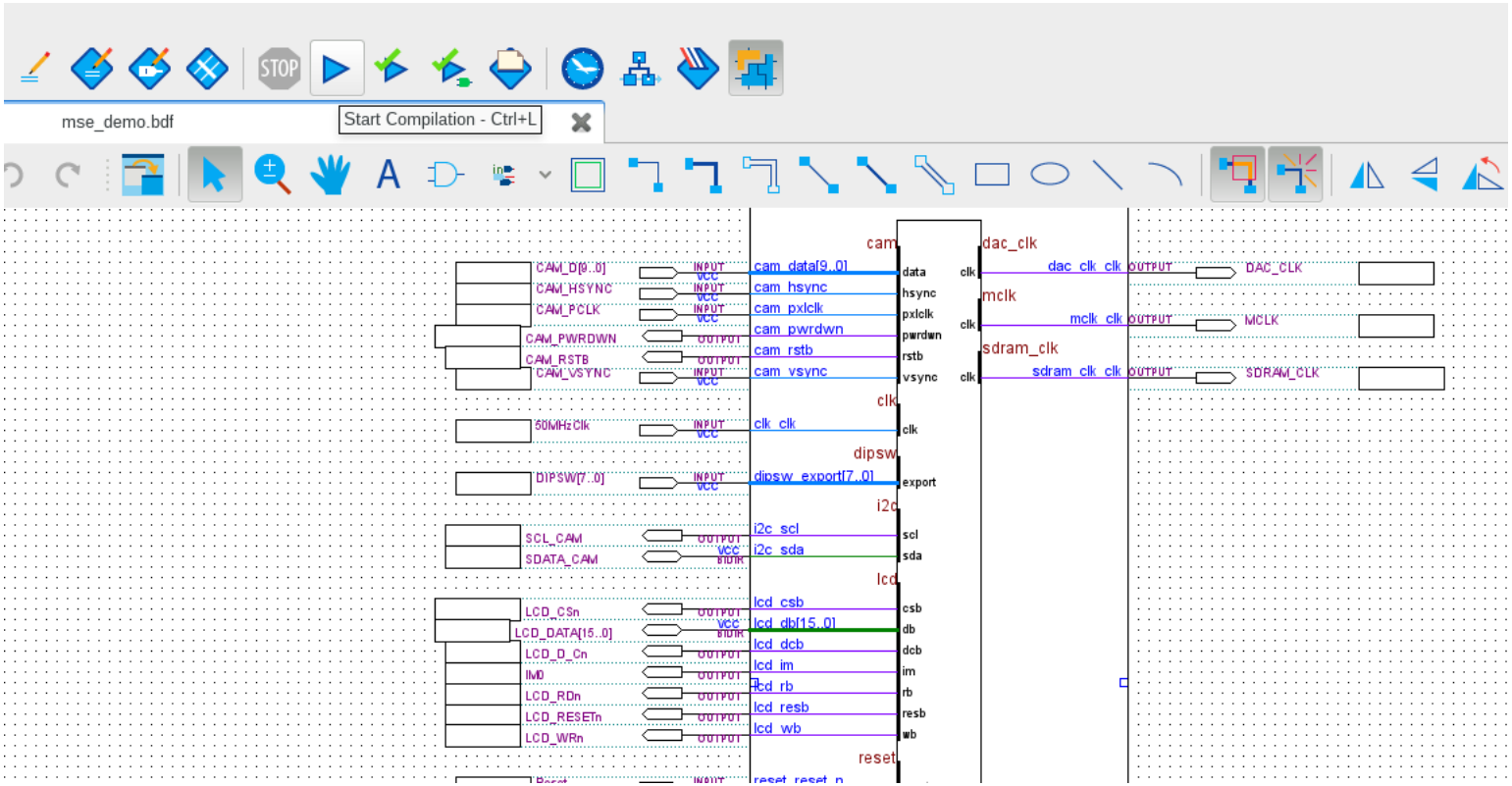
- Switch to `Files` in project navigator
- Add additional files (right click and select `add...`)
- Select the ***.qsys** and **mse_demo.bdf** files
- Add both files (one after the other...)
-

TCL Scripting



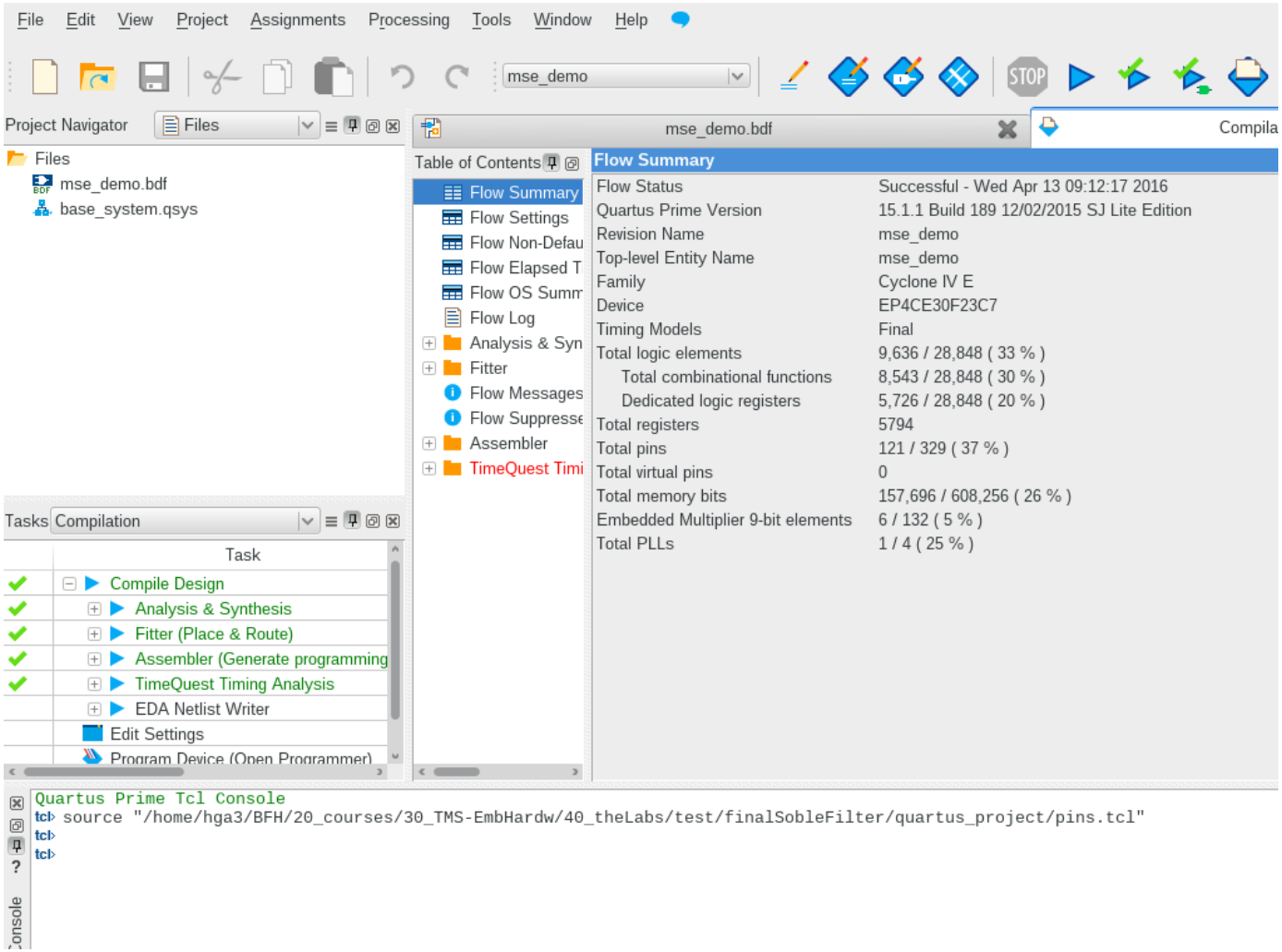
- Run the TCL script `pins.tcl` for pin assignment
- The file is located in the project folder
- Check if all pins have been assigned
- Your system should look like the one in figure above

TCL Scripting



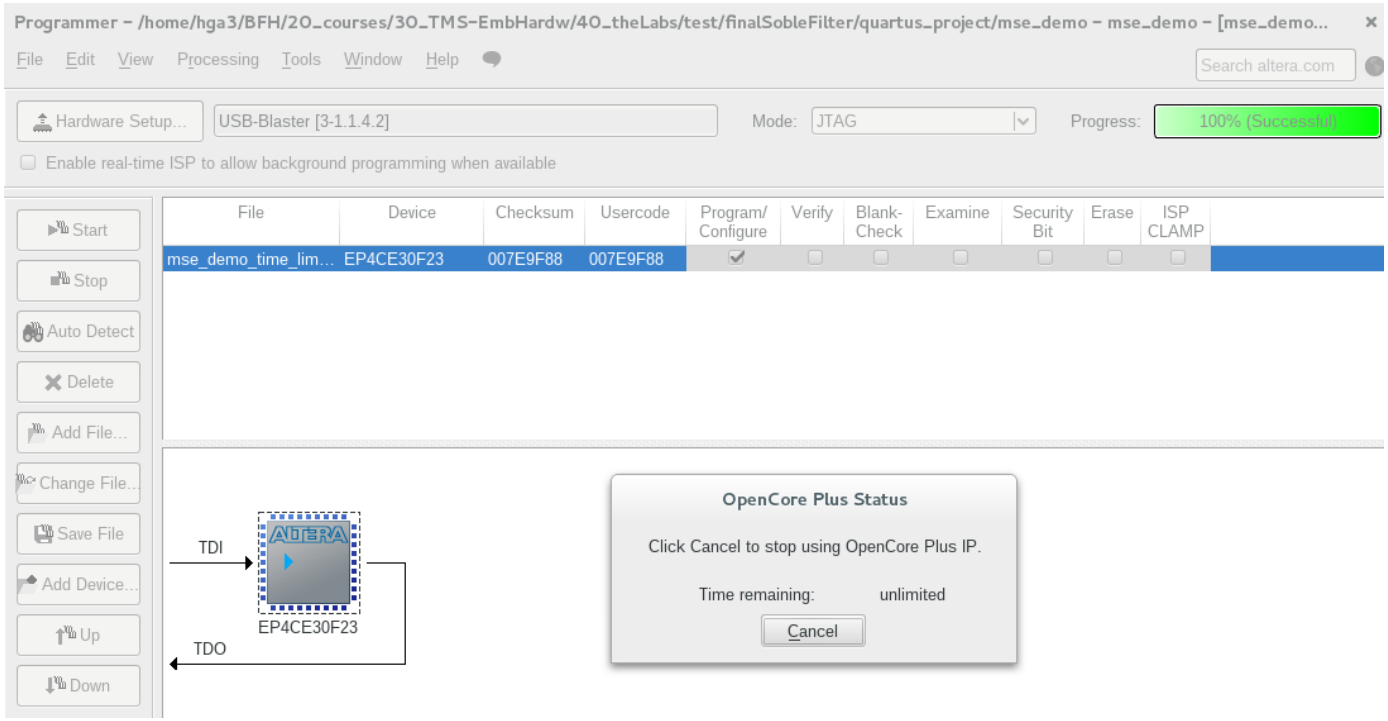
- Run the TCL script `pins.tcl` for pin assignment
- The file is located in the project folder
- Check if all pins have been assigned
- Your system should look like the one in figure above

Build the System



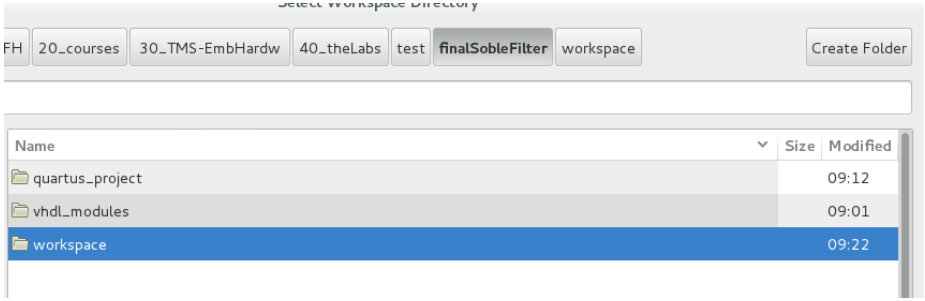
- Run the “Compile Design” process (click on the blue triangle)
- After successful compilation program the board
- Run the quartus programmer and select the target to program as well as the configuration file

Build the System



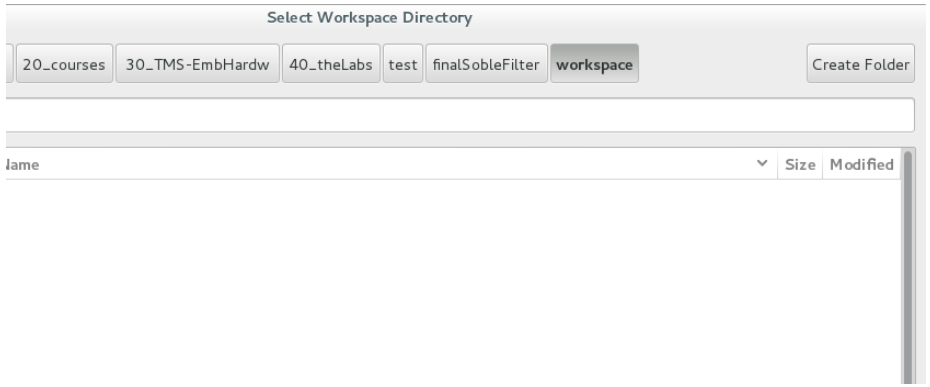
- Run the “Compile Design” process (click on the blue triangle)
- After successful compilation program the board
- Run the quartus programmer and select the target to program as well as the configuration file
- The configuration is time limited make sure you **not** click on cancel

Start SDK and select Workspace



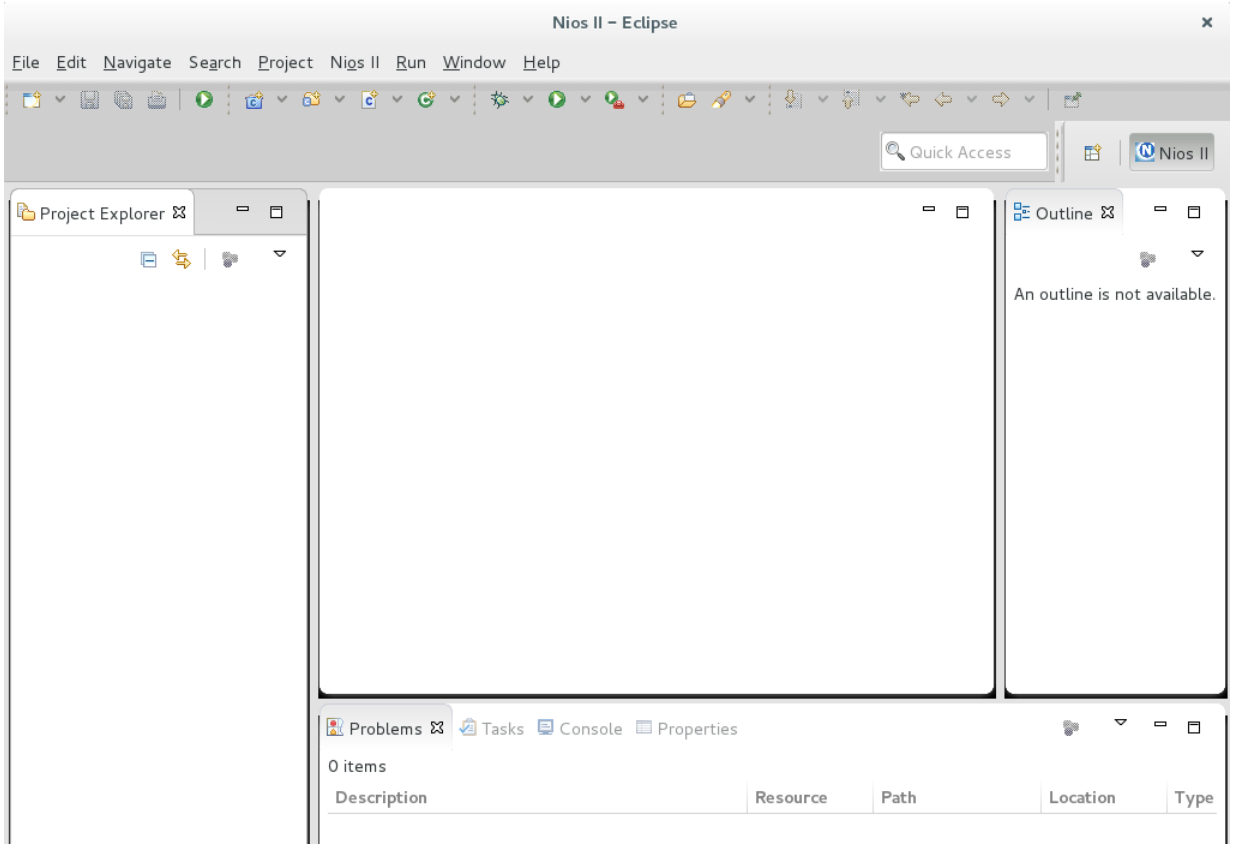
- Start Nios II SDK
- Select a workspace (to do so create new folder $\hookrightarrow$ create it in the finalSobelFilter directory)
- Choose the created folder as current workspace

Start SDK and select Workspace



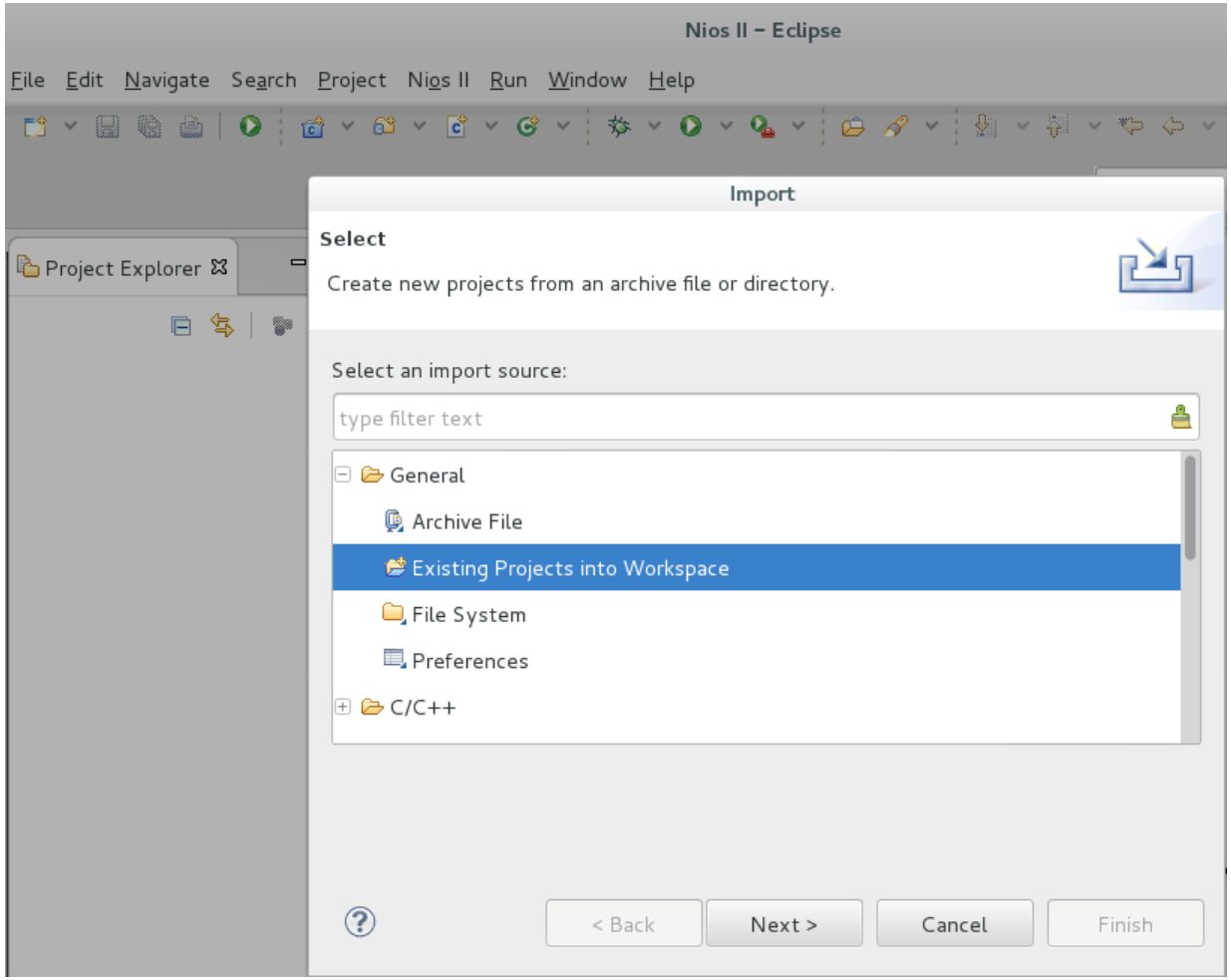
- Start Nios II SDK
- Select a workspace (to do so create new folder $\hookrightarrow$ create it in the finalSobelFilter directory)
- Choose the created folder as current workspace

Import Firmware



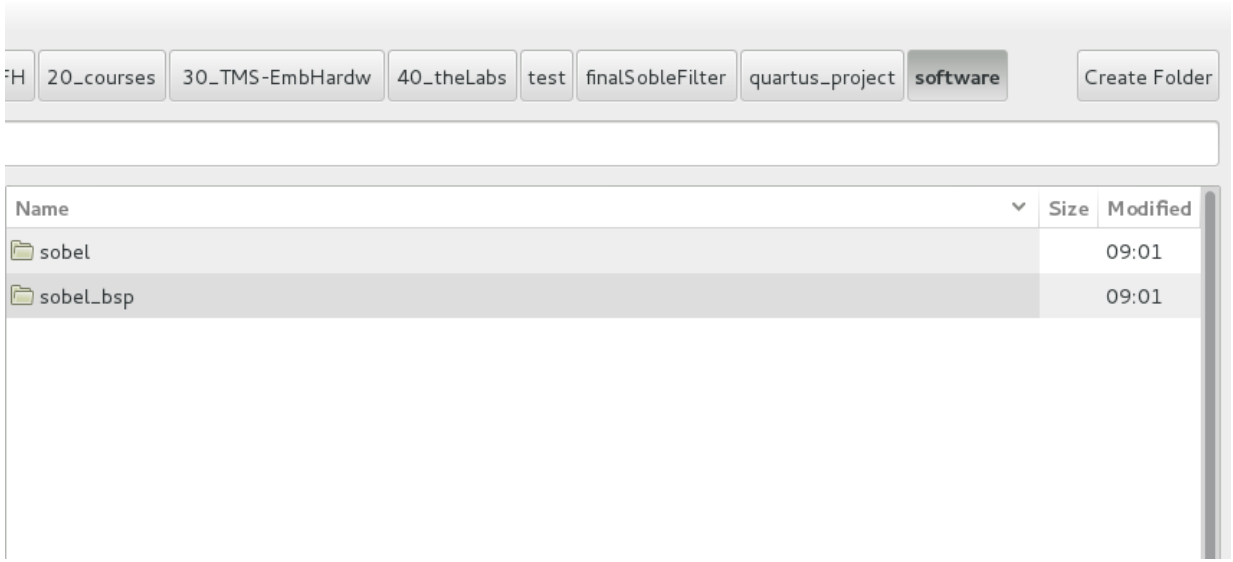
- Right click in project explorer
- Select `Existing Projects`
- Select the software folder located in quartus_project folder

Import Firmware



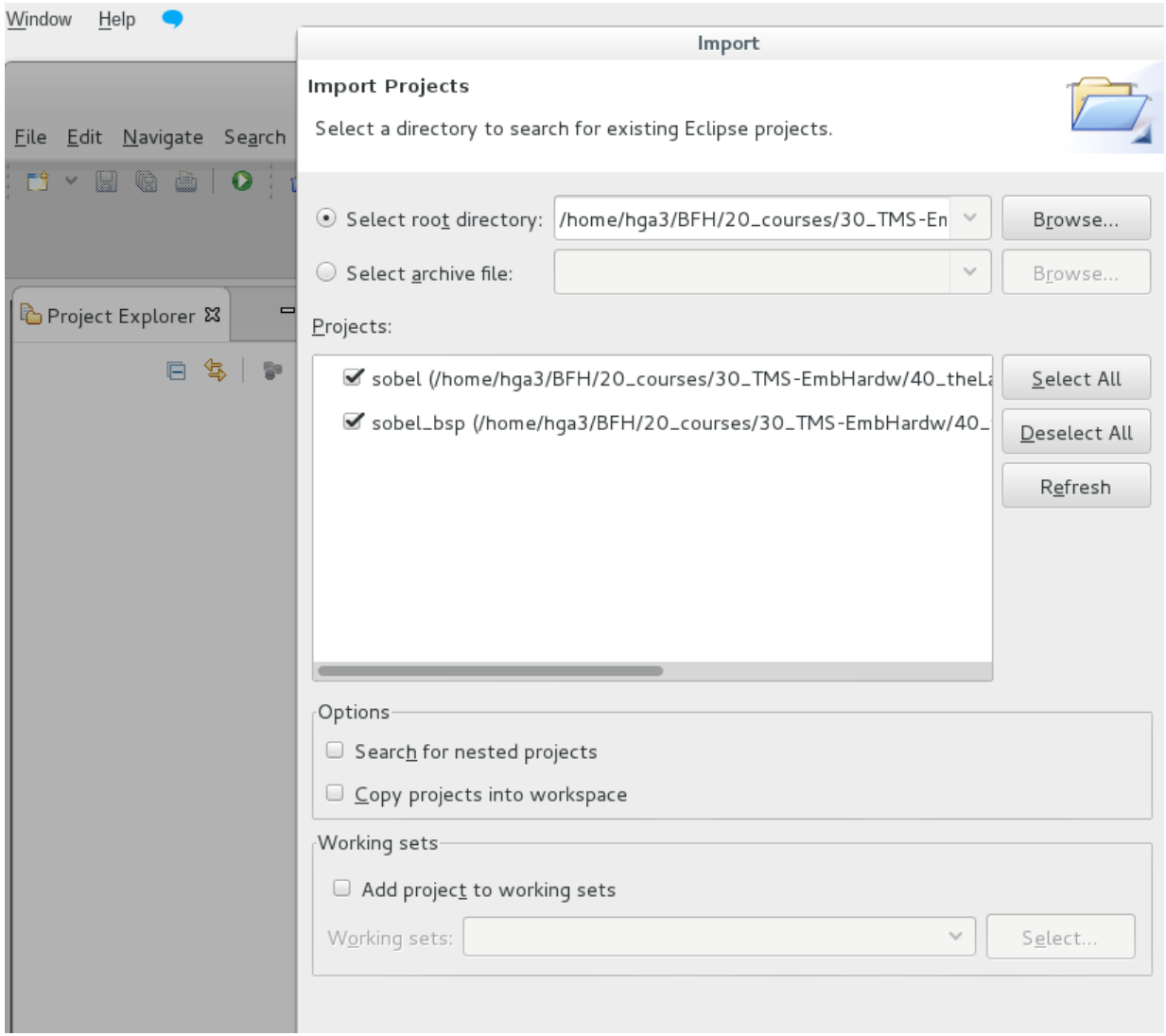
- Right click in project explorer
- Select `Existing Projects`
- Select the software folder located in quartus_project folder

Import Firmware



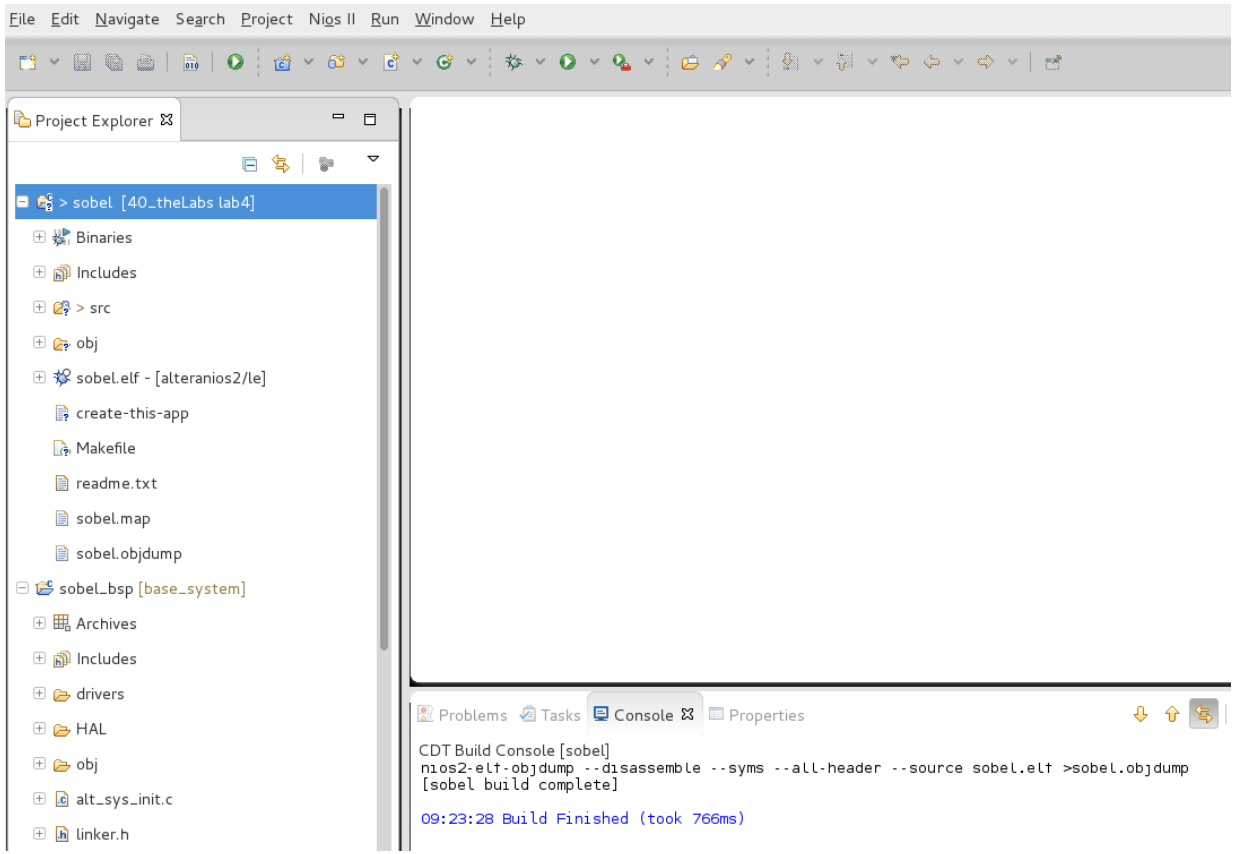
- Right click in project explorer
- Select `Existing Projects`
- Select the software folder located in quartus_project folder

Import Firmware



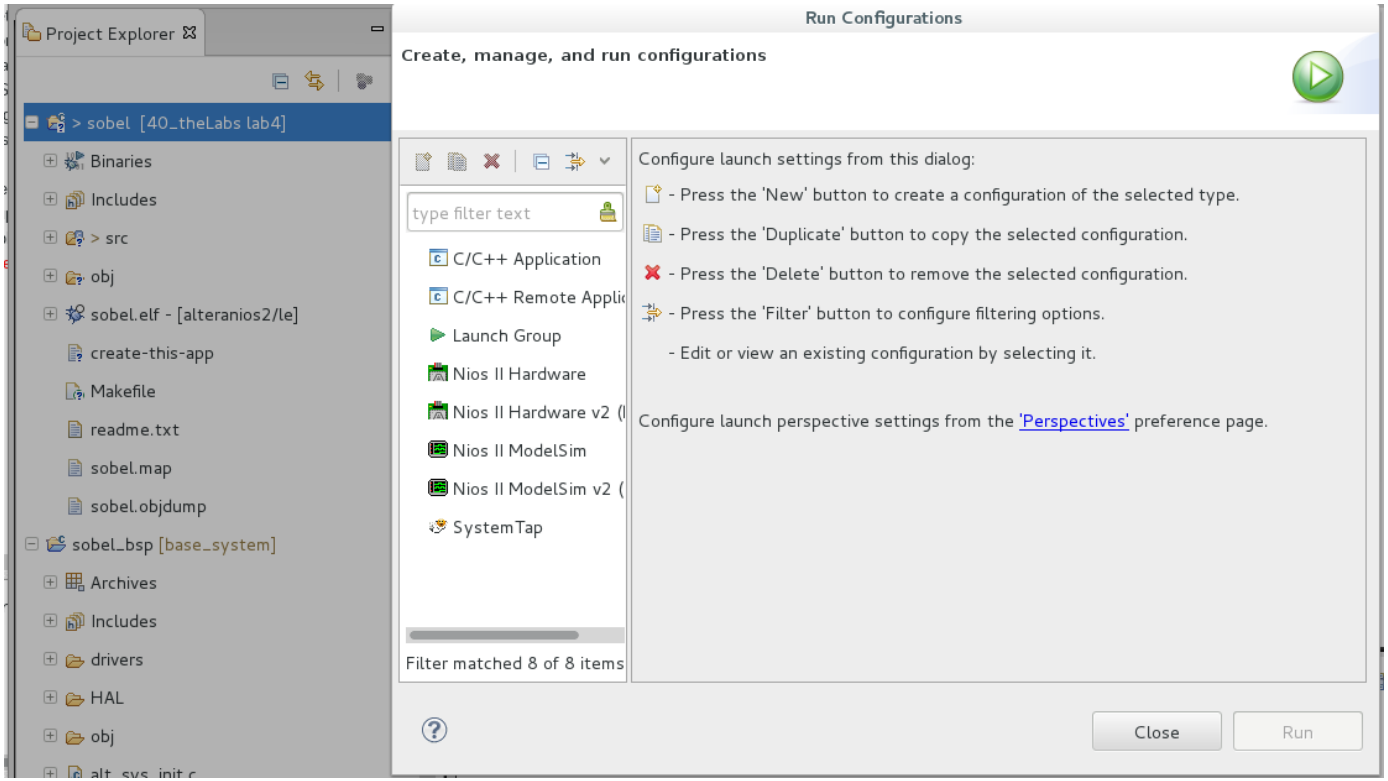
- Select both the sobel firmware and the corresponding BSP
- Rebuild the BSP first
- Rebuild BSP project and afterwards the sobel project

Import Firmware



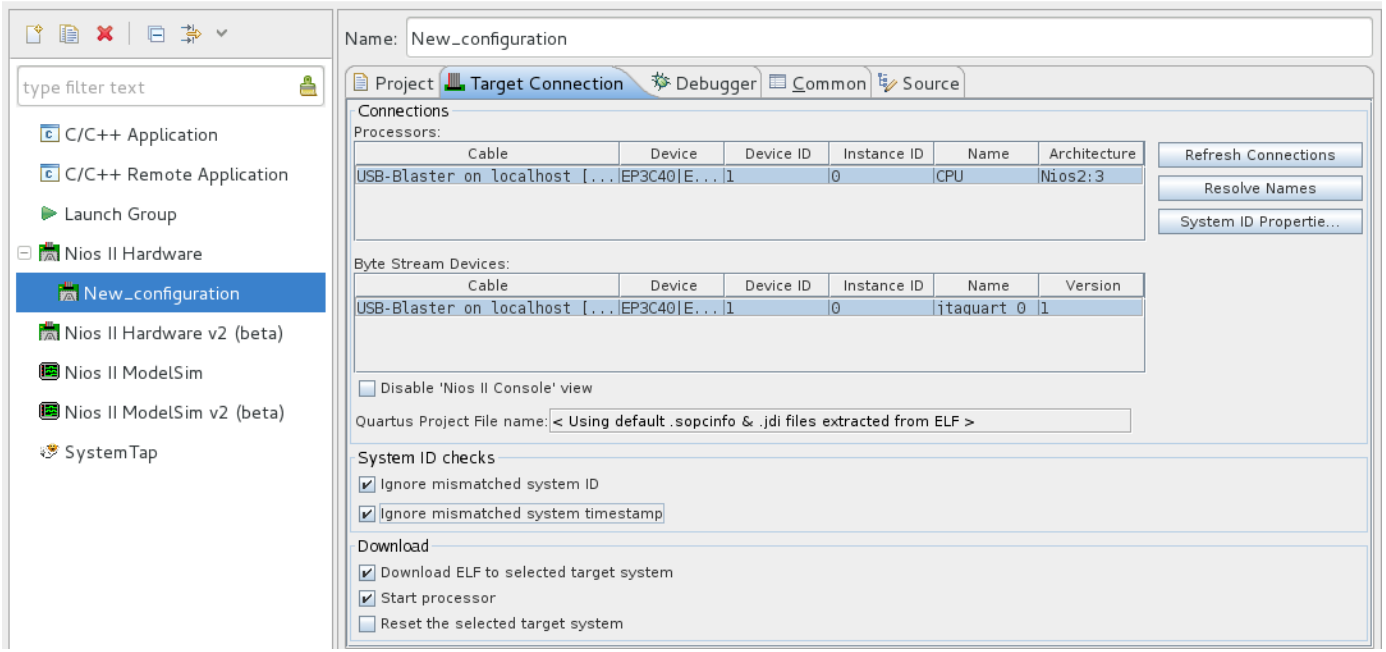
- Select both the sobel firmware and the corresponding BSP
- Rebuild the BSP first
- Rebuild BSP project and afterwards the sobel project

Download Firmware



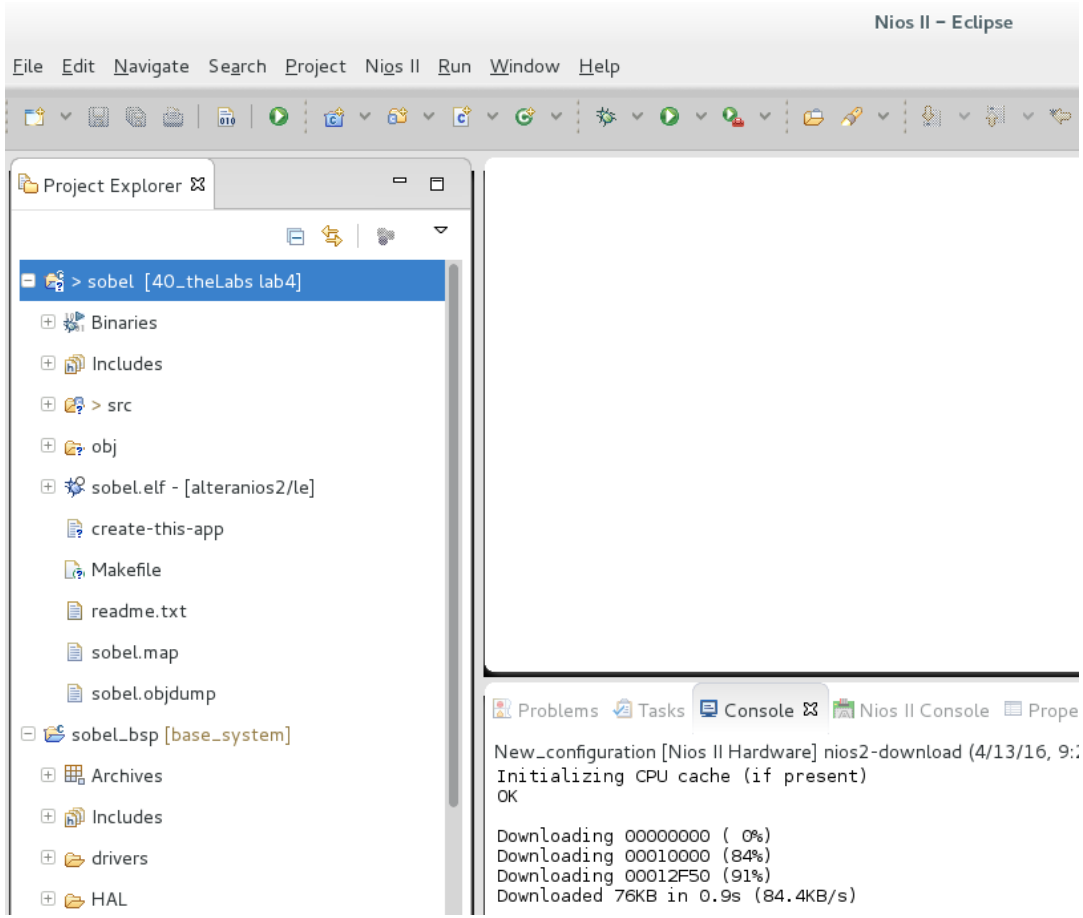
- Create new “run Configuration”
- Choose “Nios II Hardware”
- Tick both check-boxes for System ID checks (to deactivate the checks)
- Click on “Refresh connection”
- Click on “Apply” first and second on “Run”

Download Firmware



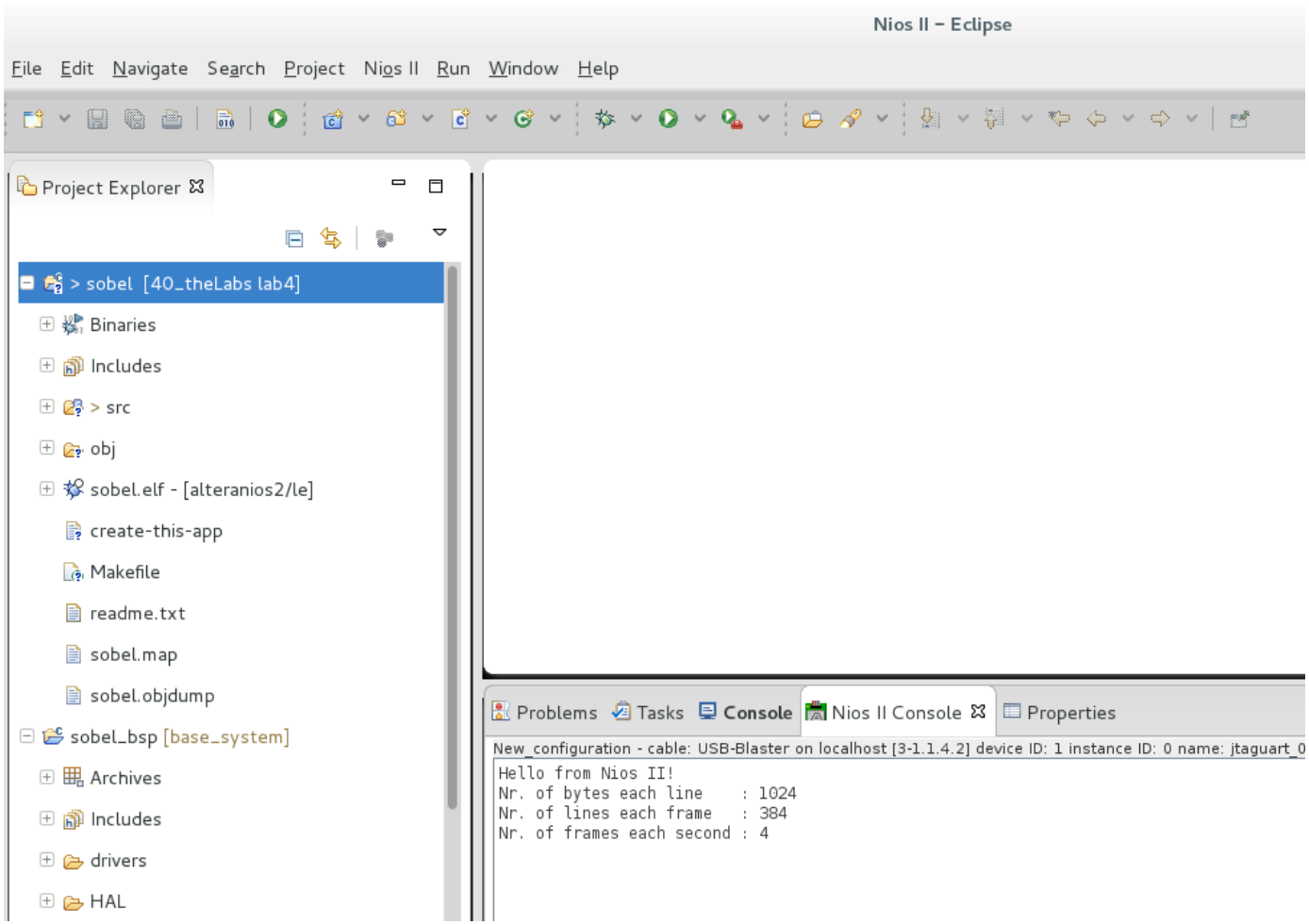
- Create new “run Configuration”
- Choose “Nios II Hardware”
- Tick both check-boxes for System ID checks (to deactivate the checks)
- Click on “Refresh connection”
- Click on “Apply” first and second on “Run”

Download Firmware



- Create new “run Configuration”
- Choose “Nios II Hardware”
- Tick both check-boxes for System ID checks (to deactivate the checks)
- Click on “Refresh connection”
- Click on “Apply” first and second on “Run”

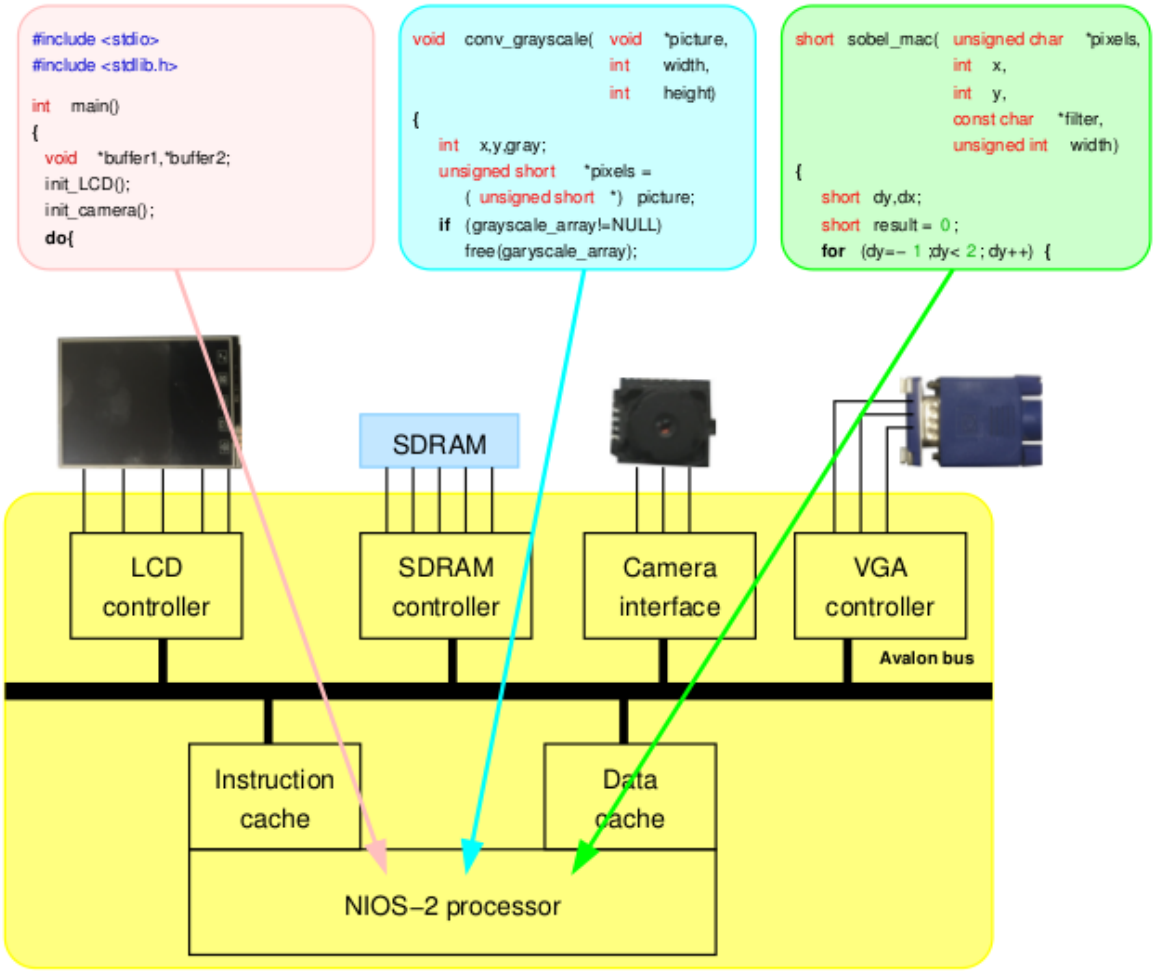
Test on Device



- The Firmware sends a hello message like the one shown in figure back
- Test the function as explained next

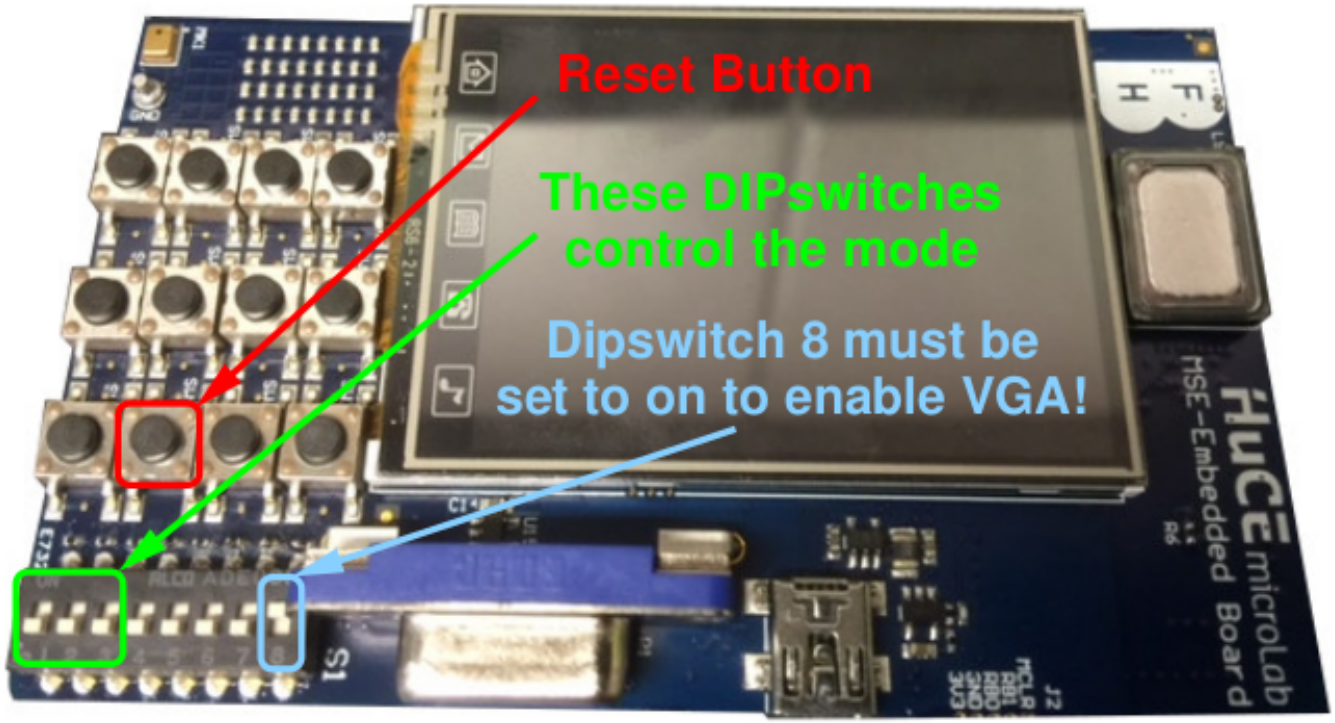
Test on Device

Block Diagram



- Current system on the development board
- A VGA IP was added for demo on an external screen
- The SOBEL filtering is done on software
- Some parts of SOBEL are computation intense

Switch Modes



SW1	SW2	SW3	Function
off	off	off	Direct camera ↪ LCD
on	off	off	Grayscale
off	on	off	Sobel-X
on	on	off	Sobel-Y
X	X	on	Sobel complete