

A P P N O T E S SM

Designing with Altera's NIOSII Embedded Processor in HDL Designer

By: Roy Clement
Last Modified: 2 November 2006



©Copyright Mentor Graphics Corporation 1995-2006. All rights reserved.

This document contains information that is proprietary to Mentor Graphics Corporation. The original recipient of this document may duplicate this document in whole or in part for internal business purposes only, provided that this entire notice appears in all copies. In duplicating any part of this document, the recipient agrees to make every reasonable effort to prevent the unauthorized use and distribution of the proprietary information.

Trademarks that appear in Mentor Graphics product publications that are not owned by Mentor Graphics are trademarks of their respective owners.

This application note contains step-by-step instructions for creating and importing NIOSII embedded processor sub-systems into HDL Designer. This includes instructions on setting up the HDL Designer environment for simulation, synthesis and interfacing to device "place and route" for FPGA designs containing embedded processor sub-systems.

Quartus, QuartusII, SOPC Builder, Cyclone II and NIOSII are all register trademarks of the Altera Corporation.

Contents

[Introduction](#)

[Setting up the Altera Specific Libraries \(simulation\) in HDL Designer](#)

[Creating the HDL Designer Library for the NIOSII Sub-system design](#)

[Creating the NIOSII Sub-system](#)

[ModelSim/QuartaSim Simulation](#)

[Adding the NIOSII Sub-system to the rest of the design](#)

[Synthesizing the complete design with QIS](#)

[Synthesizing the complete design with Precision](#)

[Running Quartus Place and Route](#)

[Appendix A - Typical QIS synthesis script](#)

Introduction

This application note will go through the initial steps required to setup the HDL Designer environment to include systems developed with Altera's SOPC Builder (part of the QuartusII environment). The application note refers specifically to designs developed with HDL Designer 2005.3/2006.1 and Quartus 6.0.

The example and screen shots are based upon the NIOSII Cyclone II embedded processor design example used in Altera's "**NIOSII Hardware development tutorial**". However the methodology is valid for all Altera families which support the NIOSII processor. The Altera tutorial can be downloaded from their web page.

A basic working knowledge on the part of the reader is assumed for QuartusII, the SOPC Builder and HDL Designer. Only the processes and operations specific to the SOPC Builder flow in HDL Designer are covered. The application note also assumes that QuartusII 6.0, the SOPC Builder and HDL Designer have been installed and correctly configured to use the required simulator and synthesis tools. The system variable QUARTUS_ROOTDIR must be correctly set to point at the root location for Quartus.

Setting up the Altera Specific Libraries (simulation) in HDL Designer

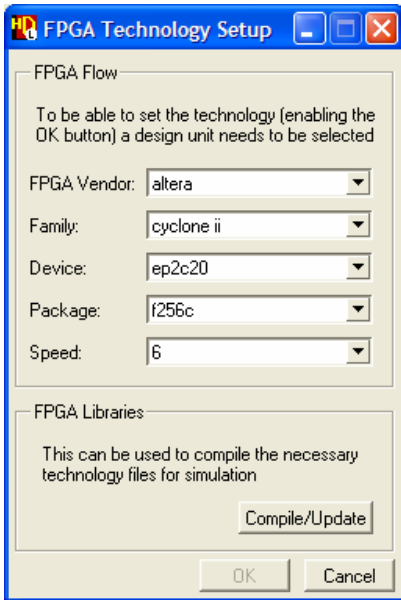
A number of general and technology specific libraries are supplied by Altera primarily to support simulation. This section contains a step by step description of how to map and compile these libraries within the HDL Designer environment. The compiled libraries can be in a central location shared by all users or specific to your project, more on this later. Additional libraries specific to the SOPC Builder flow are also required, these are managed by the SOPC Builder plugin, again more on this later.

Step by Step Process for setting up the simulation libraries:

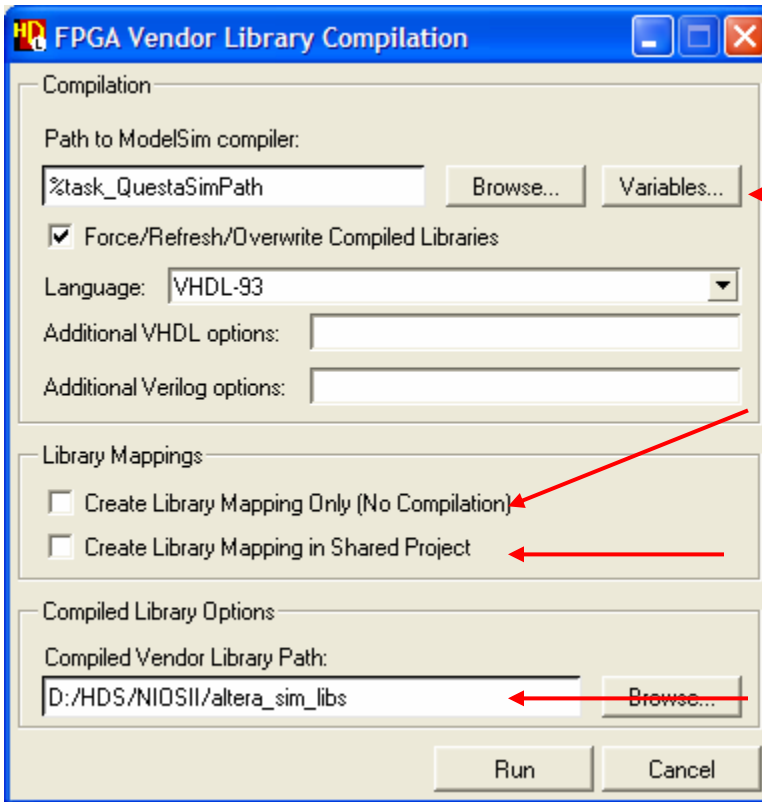
In the "Tasks" tab select the "FPGA Technology Setup" task.

1. Enter the details as required for the chosen technology by:

First selecting the "FPGA vendor", subsequent selections for "Family", "Device", "Package" and "Speed grade" are context sensitive to the previous selection. The "FPGA Vendor" and "Family" options both affect the libraries selected by the plugin for compilation. Any changes to these two options may require the plugin to be re-run. The "Device", "Package" and "Speed grade" do not effect the library selection and can be changed later before synthesis.



2. The next step is to setup the HDL Designer library mapping and compile options for simulation, access the related setup dialog box by pressing the “Compile/Update” button.



Reflects setting for default simulator (HDL Designer internal variable). Use variable button to change compiler.

Tick this option to create an HDL Designer library mapping only.

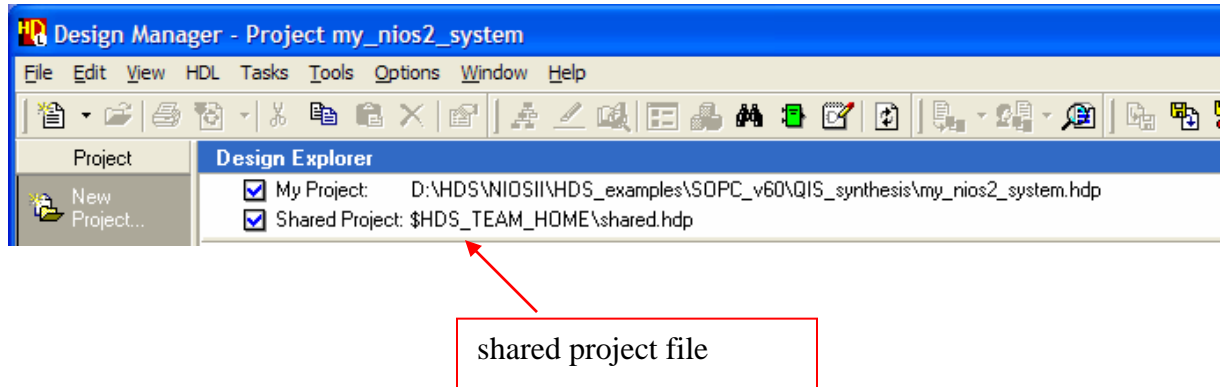
Tick this option to create the HDL Designer library mapping in a shared project (shared.hdp) file.

location of compiled libraries

3. Enter the required options in the “FPGA Vendor Library Compilation” dialog box.

3.1 **“Create Library Mapping Only”** – Tick this option if the library has already been compiled, in which case the plugin will create only the appropriate library mappings in the HDL Designer project (no actual compilation). The directory containing the compiled library must exist in the location pointed to by the “Compiled Vendor Library Path”.

3.2 “**Create Library Mapping in Shared Project**” – Tick this option if you wish to create the library mappings in the shared project file. In a team based environment or where other users need to use the same libraries, the mappings can be shared through a common, shared hdp file. The path to this share project file is detailed in the “Project” tab, as shown below.

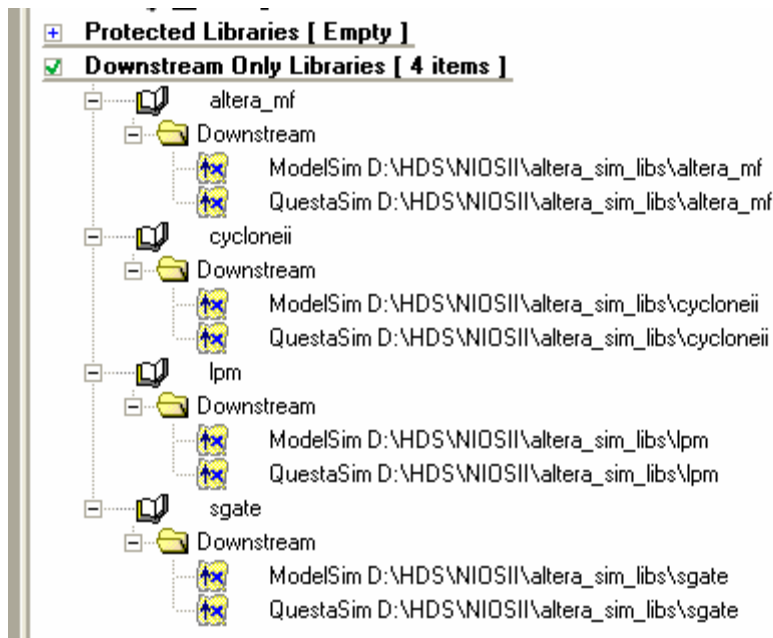


3.3 “**Compiled Vendor Library Path**” – Enter the path for the root directory of the compiled libraries. This location will be used by the plugin to create the HDL Designer library mappings and, if required, compile the libraries. The source data for these libraries is automatically located by the plugin through the system variable \$QUARTUS_ROOTDIR.

4. Press “Run” to apply the options.

Typical HDL DESIGNER Library Mappings for the Altera Tutorial using Cyclone II

These libraries are mapped as “downstream only” as they are used only for simulation.



Creating the HDL Designer Library for NIOSII Sub-system design

The exact library structure will differ from design to design but the recommendation is that the NIOSII embedded processor sub-system is maintained in a separate HDL Designer library. The sub-system design is not particularly portable because of hard links generated in the source code by the SOPC Builder (see [ModelSim Simulation](#)). However, maintaining the sub-system in a unique library helps with design understanding and clarification. The top level and the rest of the FPGA design can use as many different libraries as the designer requires.

If the HDL designer library for the NIOSII sub-system does not exist, the next step is to create it. If you have not already done so, go to the HDL DESIGNER project tab and press the “New Library” icon and follow the instructions for creating a new library. At this stage the library must be created as a “regular” type.

Creating the NIOSII Sub-system

The NIOSII sub-system is created using Altera’s SOPC Builder software. HDL Designer has a specific plugin for running the SOPC Builder. This plugin will also import the generated HDL code into the designated HDL Designer library. The design can then be simulated and synthesized as a self contained system or as part of a much larger design. The plugin will also manage the compilation and mapping of any additional Altera supplied libraries specific to the SOPC flow.

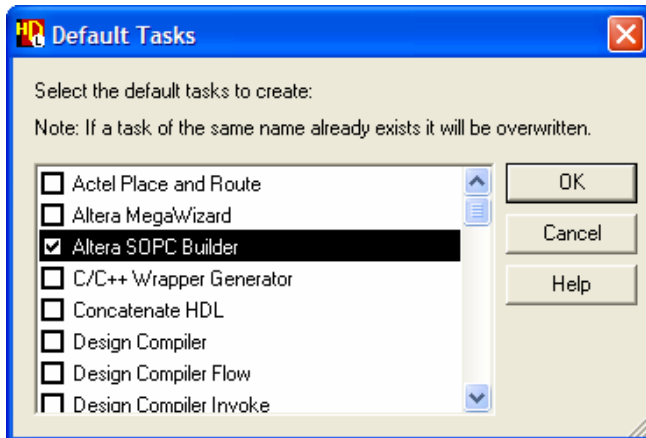
Step by step process for creating the SOPC design:

1. Invoke the “Altera SOPC Builder” plugin.

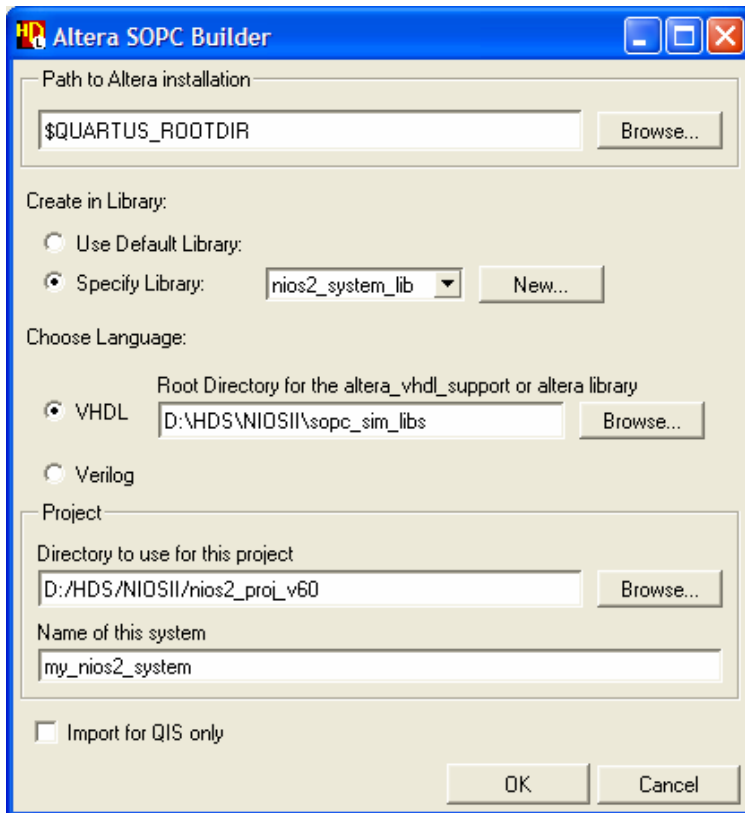
If the task is not visible in the “Tasks” tab, use the following steps to enable it.

- 1.1 In the “Tasks” tab, click the Right Hand Mouse (RHM) Key to access the popup menu.
- 1.2 Select the “Supplied Tasks” option.
- 1.3 In the tasks selection box scroll to find the “Altera SOPC Builder” task.
- 1.4 Tick to enable and press OK to apply the selection.

You may need to press the F5 key in the “Tasks” tab to refresh the UI in order to make “Altera SOPC Builder” plugin visible.



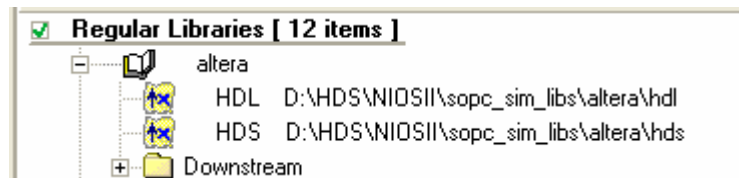
2. Enter the required options in the “Altera SOPC Builder” plugin dialog box.



2.2 **“Create in Library”** – Select either the “Use Default Library” or “Specify Library” option. The “Specify Library” option is preferred. Using the default library can cause problems later. For example if the default library is change by another activity and this plugin is run again, the design may be imported into the wrong library if the default option is left unchanged.

2.3 **“Root Directory for the altera_vhdl_support or altera library”** – These libraries are supplied by Altera and referenced by the NIOSII design. The path specifies the root directory for the compiled Modelsim directory. This can be the same directory as specified for the general and technology libraries or an entirely different location. The example uses a different directory simply to identify that the library is compiled by a different task. The source data for either library is automatically located by the plugin through the system variable \$QUARTUS_ROOTDIR. Typically the design will only require one of these libraries. See the example below of the HDL Designer library mapping created by the plugin.

The library is created as a regular library as it may need to be included in the files used for synthesis.



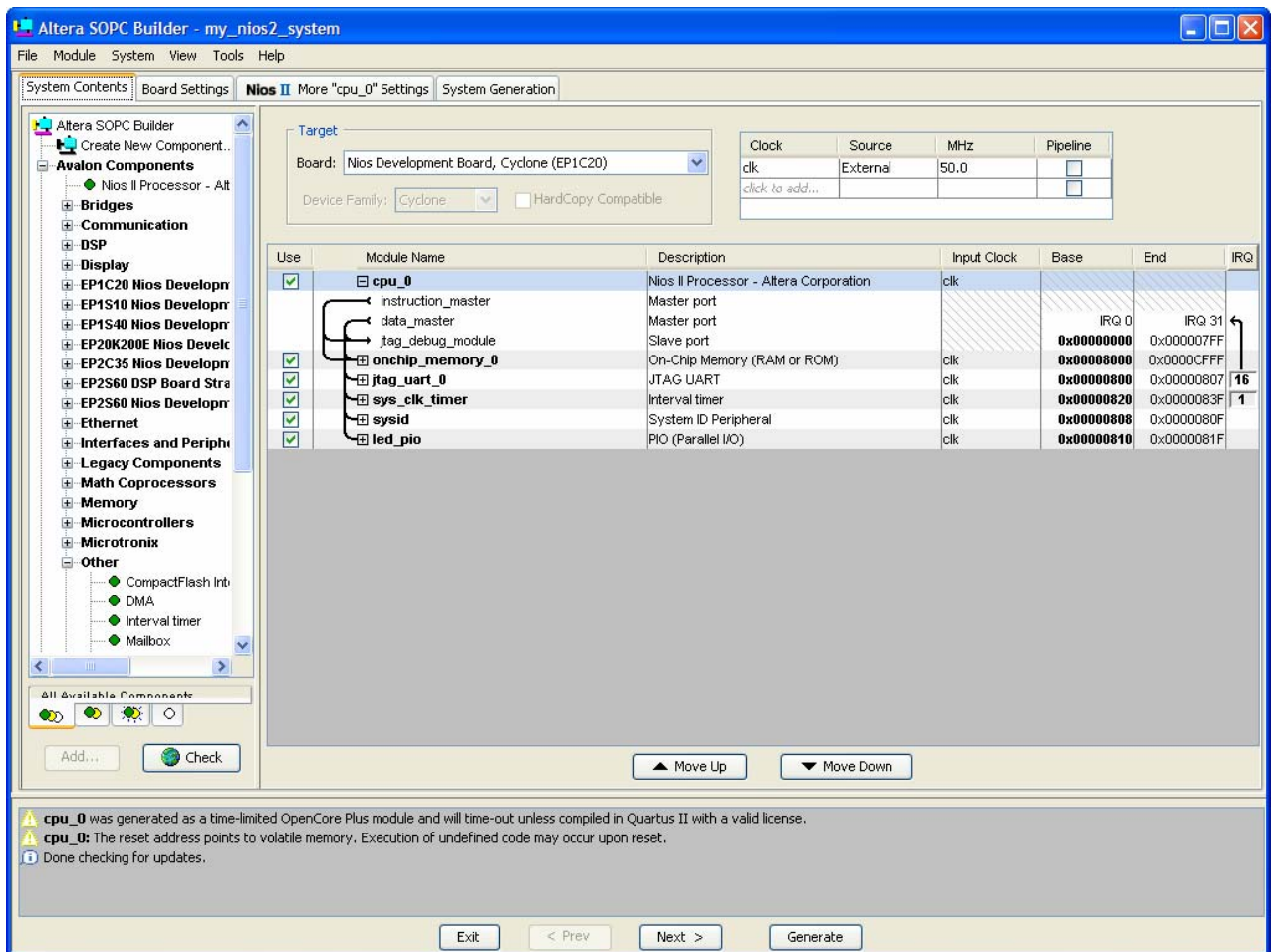
2.4 **“Name of this system”** – This is used to define the top level name of the NIOSII sub-system.

2.5 “Import for QIS only” – The NIOSII system will almost certainly contain encrypted cores. These are not synthesizable by Precision. A warning is issued if the import process detects an encrypted core. Ticking this option suppresses the warning message. This option does not affect simulation.

3. Press “OK” to invoke the SOPC Builder.

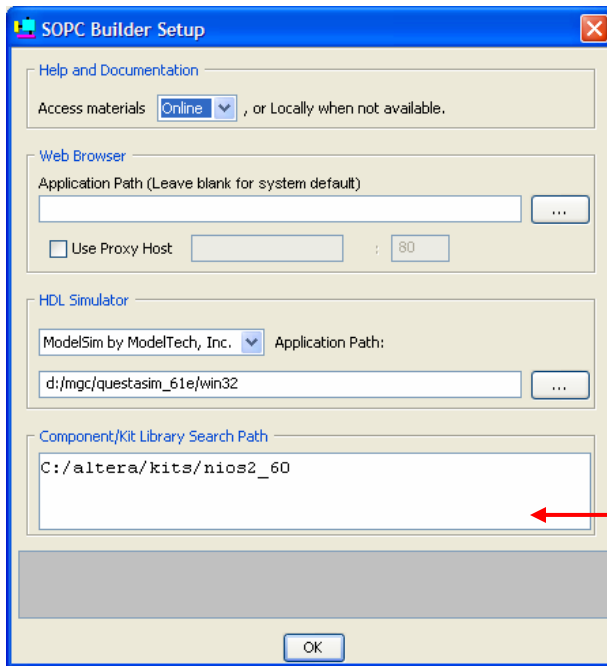
4. Create the NIOSII system in the SOPC Builder.

Add components to the NIOSII system as required, the view below shows Altera’s tutorial example. Please refer to the Altera documentation for details of the NIOSII system specifications and a full description of how to add and configure components.



5. If you experience warnings in the transcript window in regard to missing objects. It may be necessary to add the NIOSII kit to the library search path in the SOPC Builder setup.

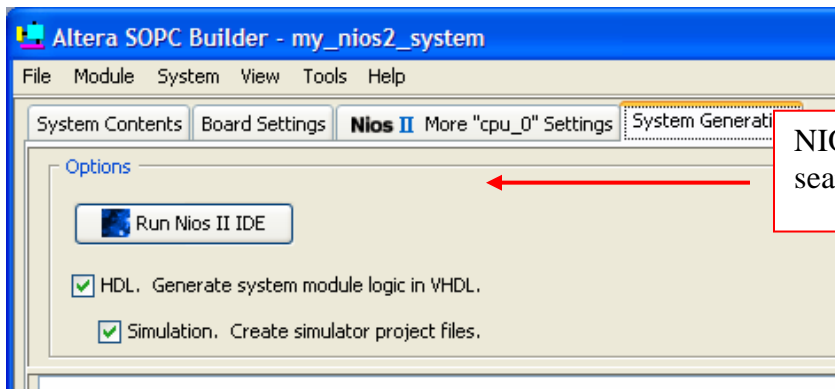
This is located under the menu “File >> SOPC Builder Setup”.



path to root directory of NIOSII kit

6. When you have finished adding components to the system, go to the “system generation” tab. Select the generation options as required. The default is to generate the source code and files for synthesis and simulation. The simulation option will create any hex or dat files that maybe need to support simulation. Typically these are placed in the SOPC project sub-directory <design_name>_sim. The HDL code will have hard coded paths referencing these files.

Generate will also create .hex files for the on-chip memory. These are placed in the NIOSII project root directory and typically have a relative path in the HDL source code. More on both these files later.

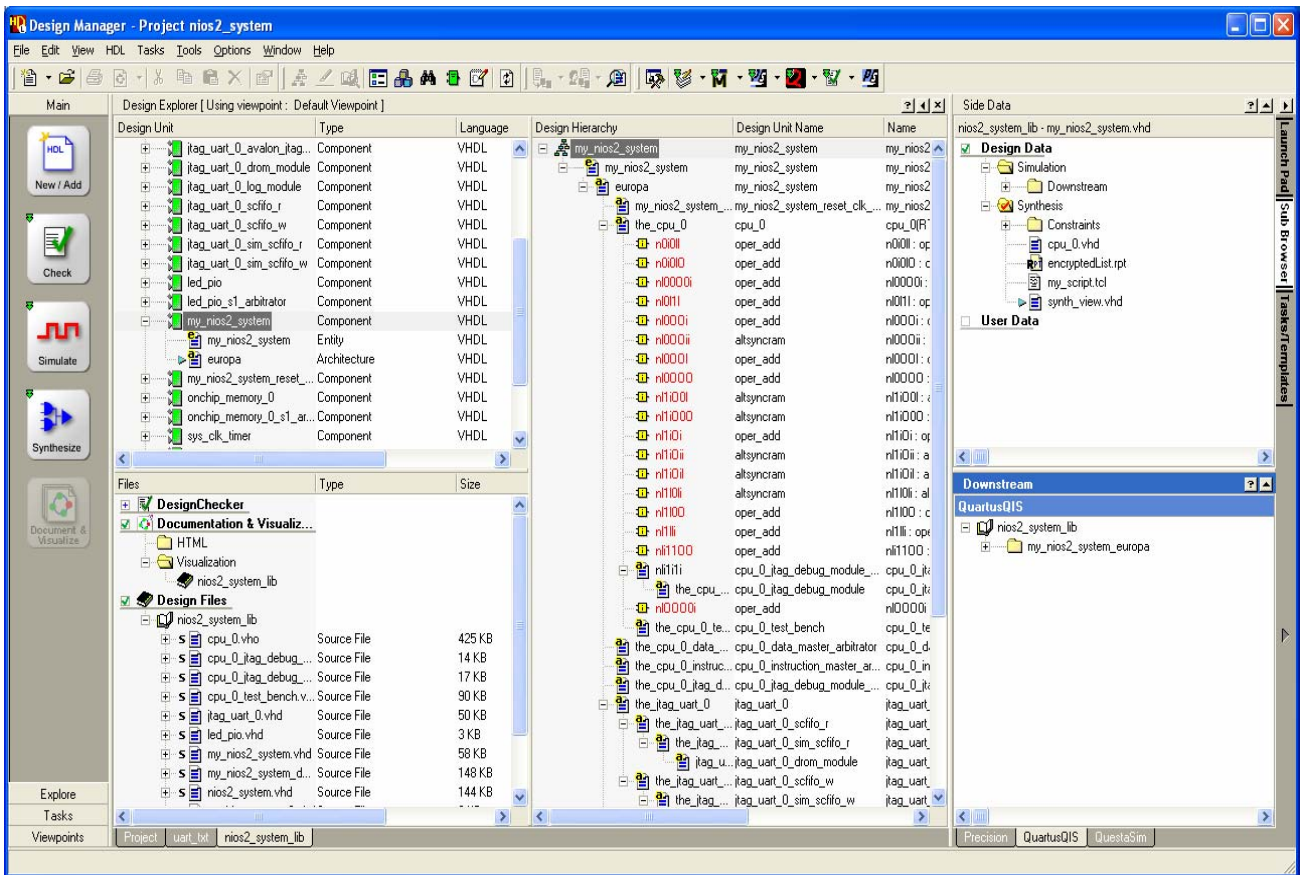


NIOSII kits search path

7. Press the “Generate” button.

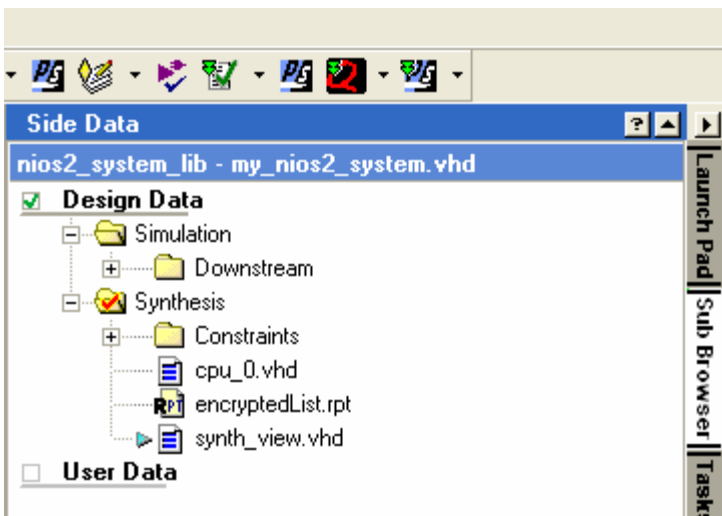
8. Wait for the generation to complete, then exit the SOPC Builder by selecting “File >> Exit”

The SOPC Builder plugin will automatically import the design and any associated encrypted cores (for synthesis) into the specified HDL Designer library. See below the HDL Designer UI representing a typical design.



The view above shows the cyclone II tutorial example. The top level component “My_nios2_system” has been selected in the Design Explorer UI with the show hierarchy option expanded to reveal the full hierarchy. The Design Hierarchy column has highlighted in red some apparently missing components. These are logic functions defined in the downstream “sgate” simulation library. The Design Hierarchy view does not currently display downstream libraries. The “Side Data” browser shows the files that have been imported to support synthesis. An expanded view is shown below.

The folder containing the files appertaining to synthesis is shown below. All the encrypted files are listed in the file, **encryptedList.rpt**. The **synth_view.vhd** is created by the SOPC plugin and contains the completed NIOSII sub-system design. This file is used by the synthesis plugins, Precision and QIS. The **cpu_0.vhd** is the encrypted cpu core (only readable by QIS).



ModelSim/QuestaSim Simulation

The SOPC Builder can create a simple testbench for the NIOSII sub-system. The testbench is created when the “Simulation - Create simulator project files” option is enabled (default) in the “SOPC Builder”. The testbench provides an initial reset function along with a free running clock. This very simple testbench can be used to help verify the NIOSII system integrity before merging it with the rest of the design.

Depending upon the exact NIOSII sub-system configuration there are a number of hex and dat files that the simulator may need to access. Access to these files generally relies on hard-coded paths specified in the source code. Providing the SOPC project is not moved the paths will remain valid.

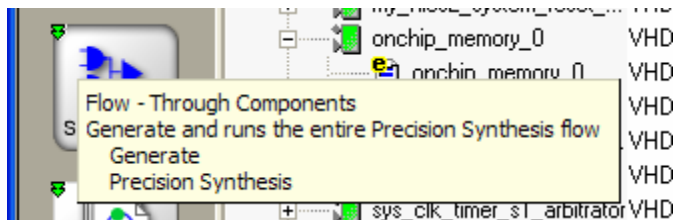
However you should be aware that the hex and dat files imported to the simulation/sidedata area are copies not the originals used by simulation. The next release of the SOPC builder plugin will import these files using the reference mechanism to point at the original source files. If viewing the originals with HDL Designer is an important factor, the originals can be re-imported manually as a group in HDL Designer 2006.1 or individually in 2005.3.

Step by step process to compile the design for simulation:

1. Select either the testbench (if required) or the SOPC sub-system component in the Design Explorer view.

2. Press the main “Simulation” icon.

The simulation task will automatically create a downstream library mapping for Modelsim and compile the complete design. If it does not compile the complete design, check the hierarchy option on the “Compilation” icon. It should be set to compile through components. You can check which option is active simply by placing the mouse cursor over the icon. The option can be change by selecting the icon and pressing the LHM button.



Adding the NIOSII Sub-system to the rest of the design

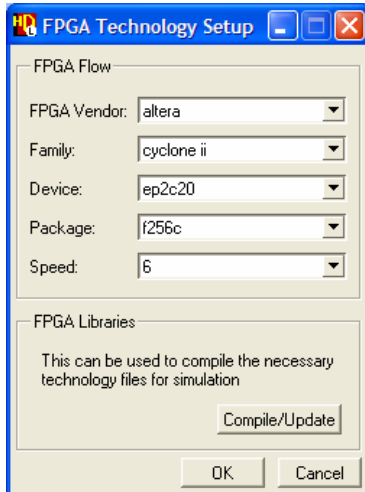
There are several ways the NIOSII system can be incorporated into the rest of your FPGA design. The recommended method is to create an overall top level structural view and instantiate the top level of the NIOSII sub-system in this view. The top component can have more than one level of hierarchy and can be described using any of the editors that support structure - text, Block Diagram or IBD. It is also recommended that the top level component is created in a different library to that of the NIOSII system. This will help with design re-use.

Synthesizing the complete design with QIS

The plugin for the Quartus Integrated Synthesis (**QIS**) can be setup to either run synthesis directly or create a batch file for running synthesis at a later date.

Step by step description of using QIS to synthesise the complete design:

1. Select the top level component for the overall design.
2. The “Device”, “Package” and “Speed grade” are all taken from the settings defined in the “FPGA Technology Setup” task. Before invoking the QuartusII Synthesis task, run the “FPGA Technology Setup” task again to check the options are correctly set. Do not press the “Compile/Update” button unless you have changed either the “FPGA Vendor” or “Family” settings as this may cause the libraries to be re-compiled un-necessarily.



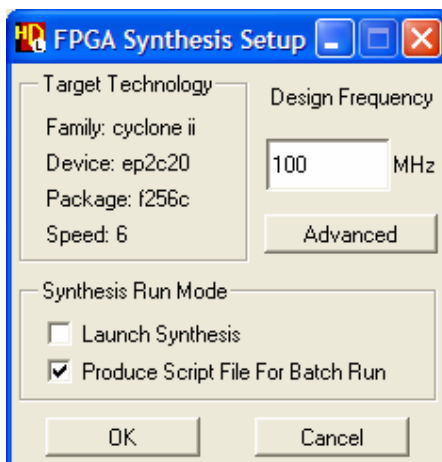
Warning: You do not need to recompile these libraries unless the “FPGA Vendor” or “Family” options have changed

3. Run the “QuartusII Synthesis” (QIS) plugin. This can be done directly by pressing the main “Synthesis” icon or by selecting the appropriate task in the “Tasks” tab.

If you want to use the main “Synthesis” icon, QIS must be set as the default synthesis tool.

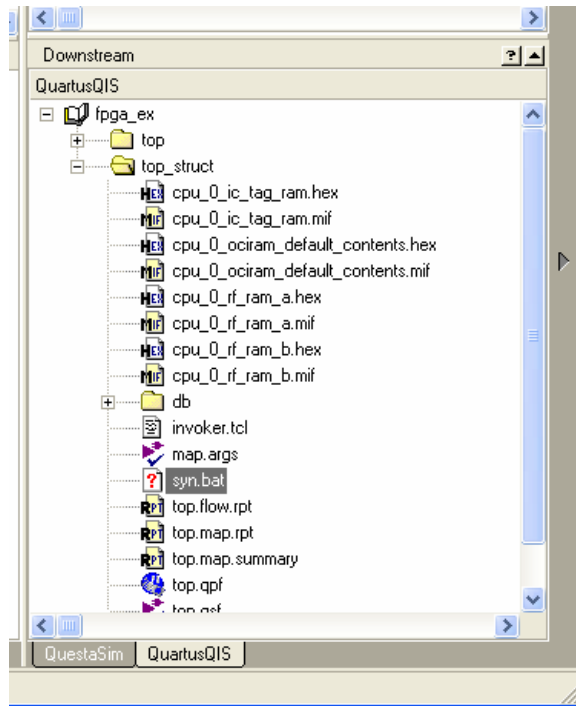
- 3.1 This can be done by using the “Setup assistant” which can be activated through the “Help >> Setup Assistant” option on the main toolbar.

The figure below shows the option in the “Quartus Synthesis” plugin to create only the batch file.



The synthesis plugin for QIS automatically creates a Quartus script for creating a project and running synthesis. The script lists all the required HDL files including the encrypted cores, packages and MIF files. See Appendix A for an example of this file.

The results of the QIS synthesis can be viewed in the “Downstream” browser under the “QuartusQIS” tab after QIS has completed.

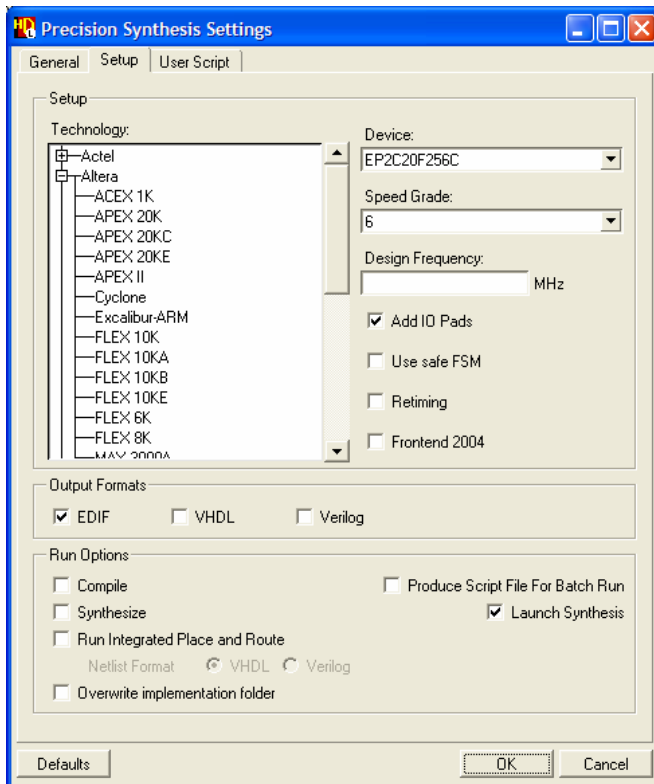


Synthesizing the complete design with Precision

The NIOSII core is supplied by Altera as an encrypted core and as such can not be synthesized by Precision. However it is possible to synthesize the rest of the design in Precision and merge the results with the encrypted core in Quartus.

Step by step description of the synthesis process using Precision:

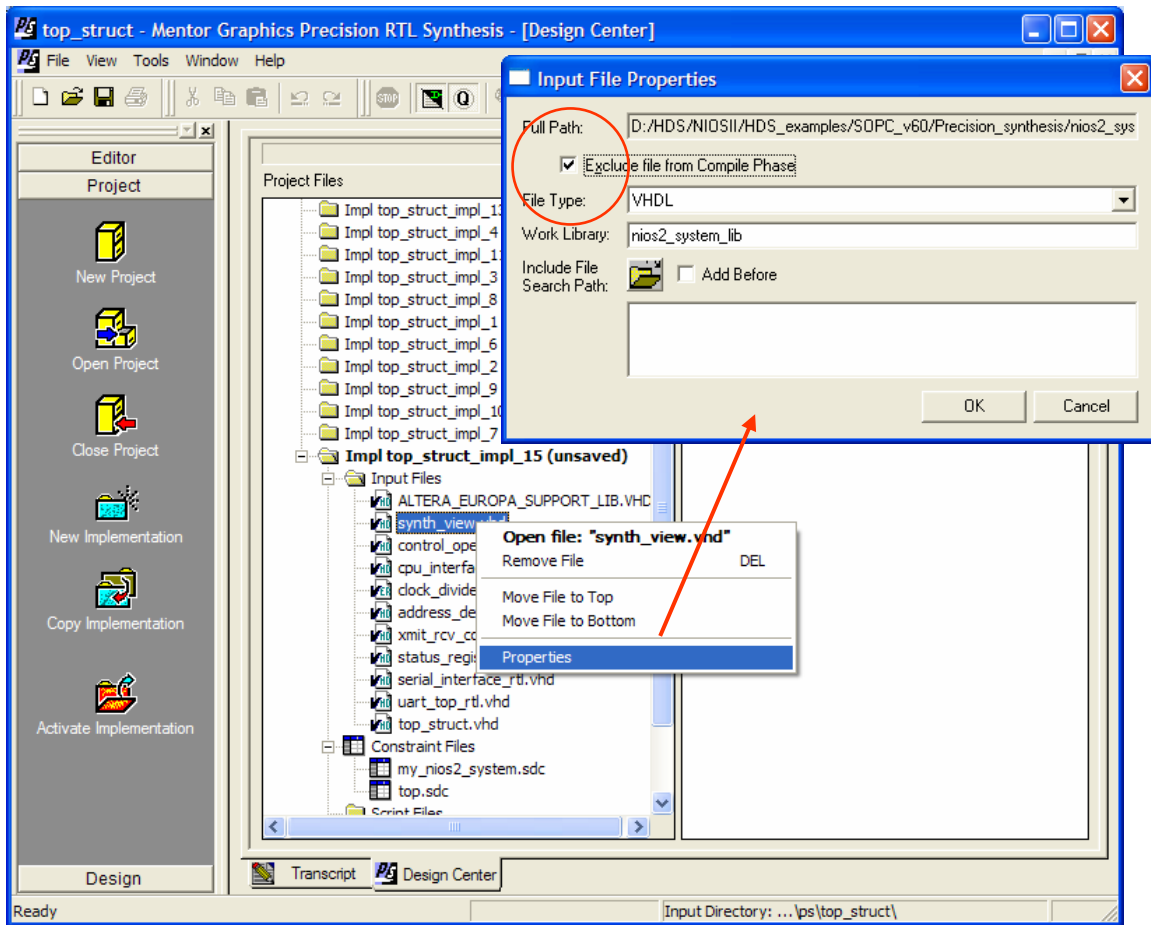
1. In the Design Explorer view select the top level component of the overall design.
2. Press the main “Synthesis” icon (assumes Precision is set as the default synthesis tool). Alternatively go to the “Tasks” tab and run the “Precision Flow” task.
3. The design environment within Precision must be manipulated to exclude the NIOSII sub-system before running synthesis. Ensure both compile and synthesis options are both deselected on the “Precision Synthesis Settings” setup box as shown below.
4. Select the required options for “Technology” (family), “Device” and “Speed Grade”, complete the other options as required.
5. Press “OK” to start Precision.



6. The easiest way to “black box” the cpu core within Precision is to exclude the whole NIOSII sub-system. This also removes the requirement to define any additional Altera libraries that are specific to the NIOSII sub-system. In the Precision input file list, select the file `synth_view.vhd` and use the RHM button to select the properties option. See the example below.

7. In the “Input files Properties” pop-up box, tick the “Exclude file from Compile Phase” option.

8. Press “OK”.

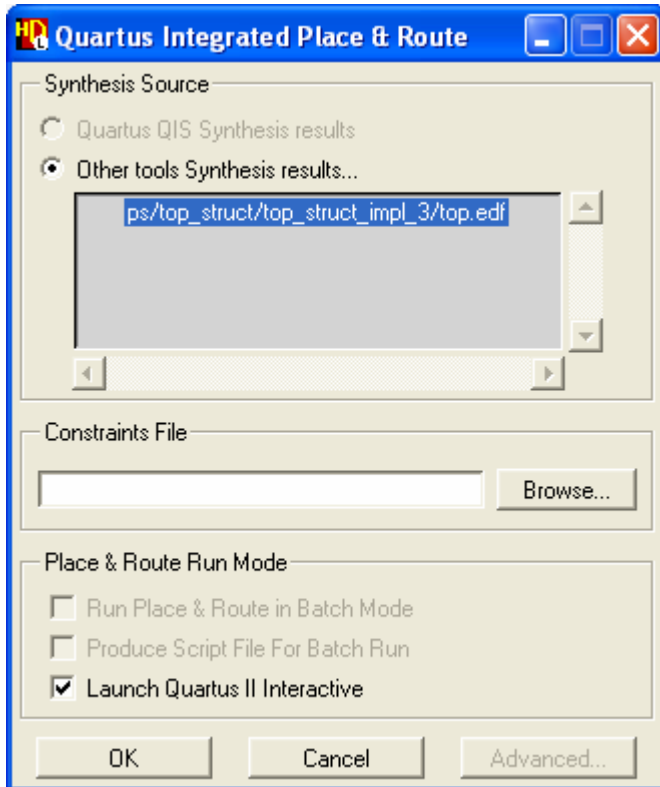


9. Still within the Precision UI, Go to the “Design” tab.
10. Set any other options that may be required by pressing the “Design Setup” icon.
11. Run “Compile” (press icon).
12. Run “Synthesis” (press icon).
13. After the synthesis has completed, Exit Precision through the menu: File >> Exit.
14. Select the “save and Exit” option.
15. Return to HDL Designer.

Running Quartus Place and Route

1. Ensure the top-level design object is still selected in the Design Explorer UI.
2. In the “Tasks” tab run the “Quartus Place and Route” plugin.
3. Any previous synthesis results should be detected by the Plugin
However, you do need to ensure that the synthesis options of the “Quartus Integrated Place and Route” setup match the actual tool used. The options are either QIS or Other (i.e. Precision).

If this is the first time Place and Route has been run, make sure you select the interactive mode as you will need to add some additional files to the Quartus Project.

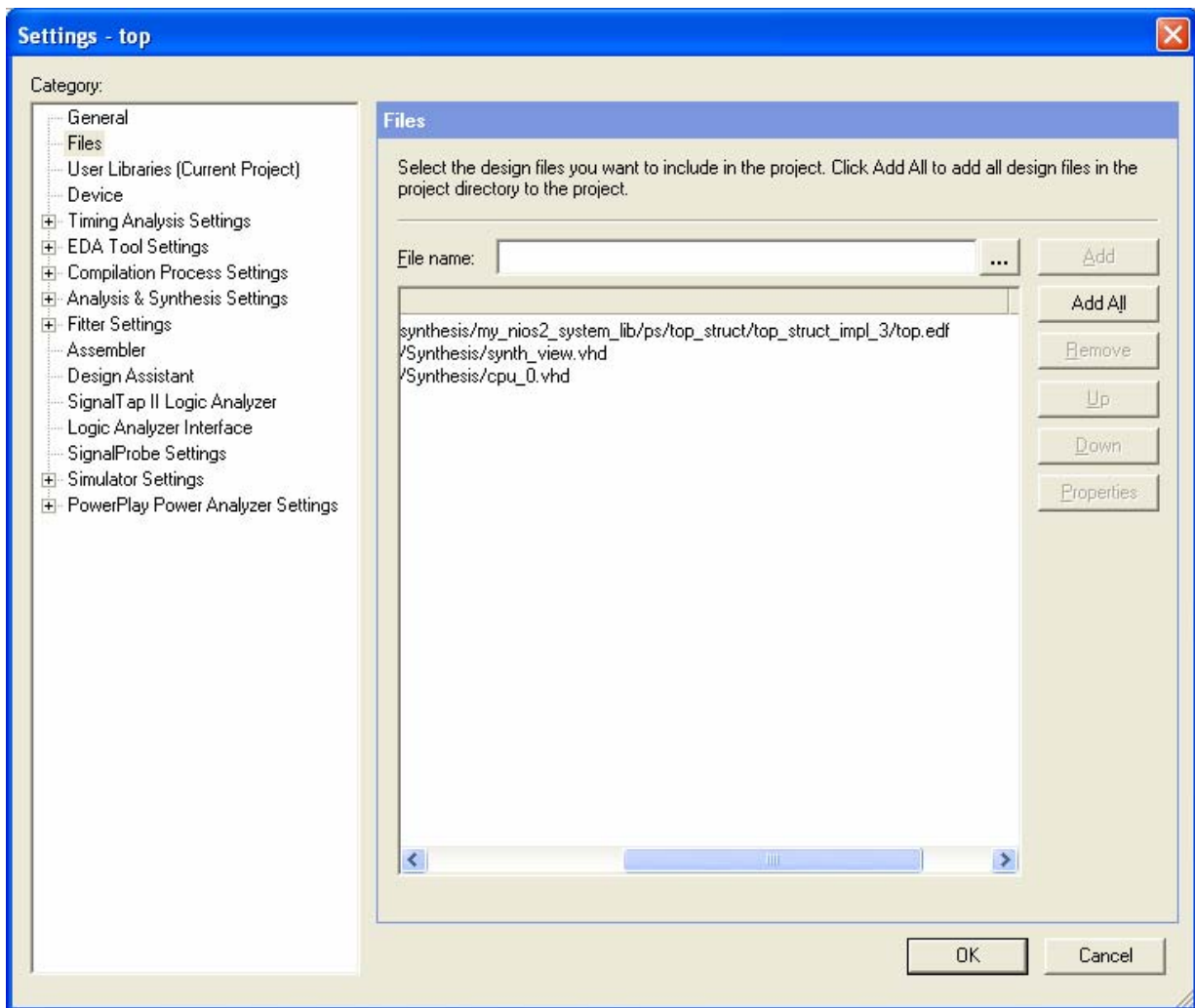


4. After Quartus has invoked, select the menu: Project >> Add/remove files in Project”.

Add both the synth_view.vhd and cpu_0.vhd files to the Quartus project setup as shown below
Use the “Up” and “Down” options to ensure the correct order of files, also shown below.

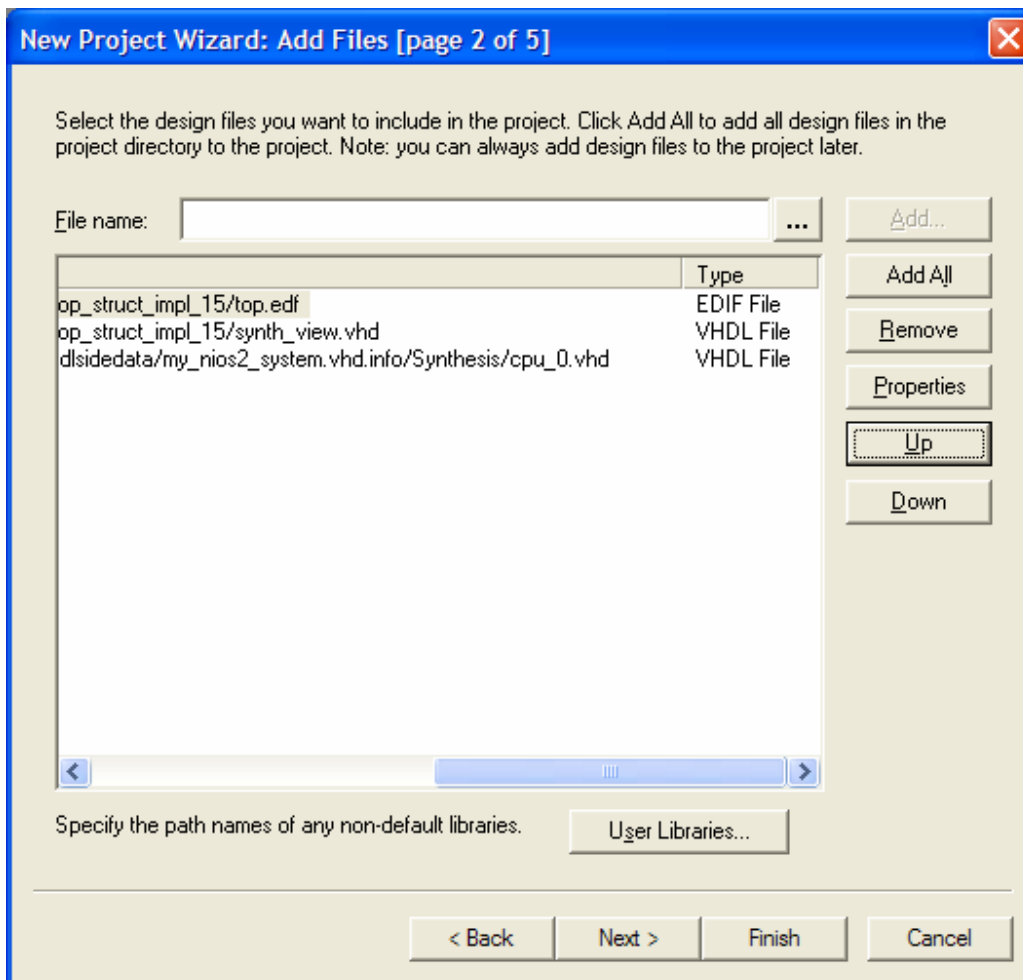
Both these files can be found in the “sidedata” area for the NIOSII sub-system, typically:

```
../hds/.hdlsidedata/my_nios2_system.vhd.info/synthesis/synth_view.vhd  
../hds/.hdlsidedata/my_nios2_system.vhd.info/synthesis/cpu_0.vhd
```



4a. Alternatively if you are creating your own Quartus Project, run the Project Wizard to create a new project which includes both the EDIF file from Precision and the NOSII system generated by the SOPC Builder.

The example below shows the Quartus "New Project Wizard", setup to include the top level EDIF file, the synth_view.vhd file which is the complete NIOSII sub-system (excluding the cpu core) and the NIOSII cpu encrypted core, cpu_0.vhd. You will recall the synth_view.vhd file was generated by HDL Designers' SOPC Builder plugin.



3. Set any other design specific options in the Quartus project.
4. Select the top level component in the "Project Navigator" UI and compile the design. This can be started through the "Processing >> Start Compilation" menu.
5. After Quartus has completed the Place and Route process, exit and return to HDL Designer.
6. The results of Place and Route can be view in the "QuartusQIS" downstream folder within the "Sub Browser" tab.

Appendix A – Typical QIS synthesis script

```
package require ::quartus::project
set need_to_close_project 0
set make_assignments 1
# Check that the right project is open
if {[is_project_open]} {
    if {[string compare $quartus(project) "top"]} {
        puts "Project top is not open"
        set make_assignments 0
    }
} else {
    # Only open if not already open
    if {[project_exists top]} {
        project_open -revision top top
    } else {
        project_new -revision top top
    }
    set need_to_close_project 1
}

# Make assignments
if {$make_assignments} {
    set_global_assignment -name VHDL_FILE "D:/HDL
DESIGNER/NIOSII/sopc_sim_libs/altera/hdl/ALTERA_EUROPA_SUPPORT_LIB.VHD"
    set_global_assignment -name VHDL_FILE "D:/HDL DESIGNER/NIOSII/HDL
DESIGNER_examples/SOPC_v60/Precision_synthesis/nios2_system_lib/HDL
Designer/.hdl-sidedata/my_nios2_system.vhd.info/Synthesis/synth_view.vhd"
    set_global_assignment -name VHDL_FILE "D:/HDL DESIGNER/NIOSII/HDL
DESIGNER_examples/SOPC_v60/Precision_synthesis/nios2_system_lib/HDL
Designer/.hdl-sidedata/my_nios2_system.vhd.info/Synthesis/cpu_0.vhd"
    set_global_assignment -name VHDL_FILE "D:/HDL DESIGNER/NIOSII/HDL
DESIGNER_examples/SOPC_v60/Precision_synthesis/fpga_ex/hdl/top_struct.vhd"
    set_global_assignment -name MIF_FILE "cpu_0_ic_tag_ram.mif"
    set_global_assignment -name MIF_FILE "cpu_0_ociram_default_contents.mif"
    set_global_assignment -name MIF_FILE "cpu_0_rf_ram_a.mif"
    set_global_assignment -name MIF_FILE "cpu_0_rf_ram_b.mif"
    set_global_assignment -name HEX_FILE "cpu_0_ic_tag_ram.hex"
    set_global_assignment -name HEX_FILE "cpu_0_ociram_default_contents.hex"
    set_global_assignment -name HEX_FILE "cpu_0_rf_ram_a.hex"
    set_global_assignment -name HEX_FILE "cpu_0_rf_ram_b.hex"
    set_global_assignment -name COMPILER_SETTINGS top
    set_global_assignment -name SIMULATOR_SETTINGS top
    set_global_assignment -name SOFTWARE_SETTINGS top
    set_global_assignment -name FMAX_REQUIREMENT 100MHz
    set_global_assignment -name FAMILY "cyclone ii"
    set_global_assignment -name TOP_LEVEL_ENTITY top
    set_global_assignment -name DEVICE ep2c20f256c6
    set_global_assignment -name USE_COMPILER_SETTINGS top

    # Commit assignments
    export_assignments

    # Close project
    if {$need_to_close_project} {
        project_close
    }
}
}
```