Contents

ASIC Logic

1. Overview

This lab gives a brief introduction to digital IP authoring techniques. The ability of writing hardware description language is essential in this lab. In this lab, you will design your own IP with hardware description language. Next, the design will be checked by lint tools and coverage verification tool. Finally, the design is to be synthesized and downloaded to FPGA on ARM Logic Module for verification.

2. Background Information

2.1. Coverage-driven Verification

Generally speaking, a coverage-driven verification methodology makes the verification flow more complete and efficient, and coverage report gives us a sense of the good and the bad of our HDL design and test bench.

The coverage-driven verification can be performed using several coverage metrics. A simple example of these metrics is the code coverage. By investigating the code coverage helps the designer find untested or redundant code in early stage of development and the quality of the stimuli can be measured. Therefore coverage gives the information that you need to know when you are ready for RTL sign-off. With a high coverage score, you can have more confidence that the code, in passing, works correctly.

3. Instructions

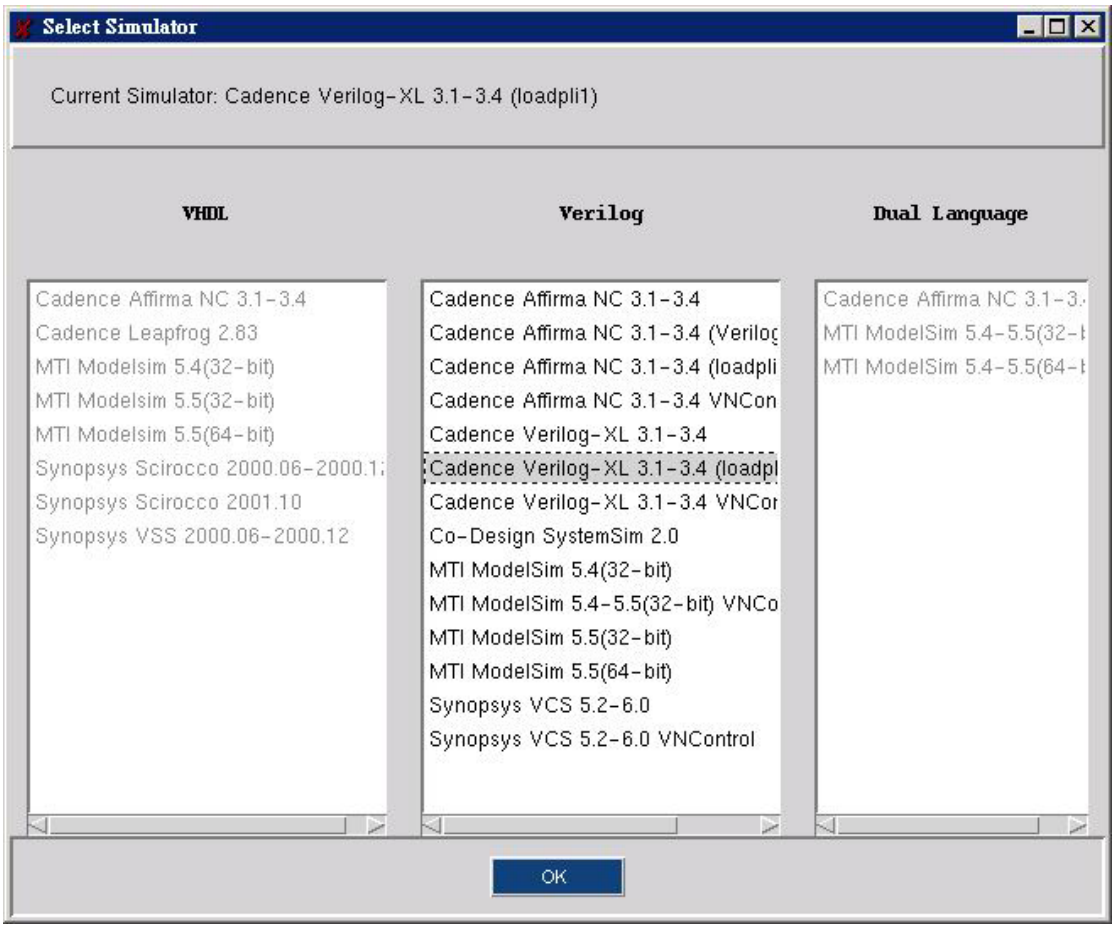
3.1. Coverage-driven Verification

1. In the directory where you want to work, start VN-Cover with the command:

```
% vn &
```

2. ***Click on the "Dynamic Verification VN-Property DX/VN-Cover"*** button in the main flow diagram to invoke VN-Cover. The flow diagram will now show the **VN-Cover flow**.

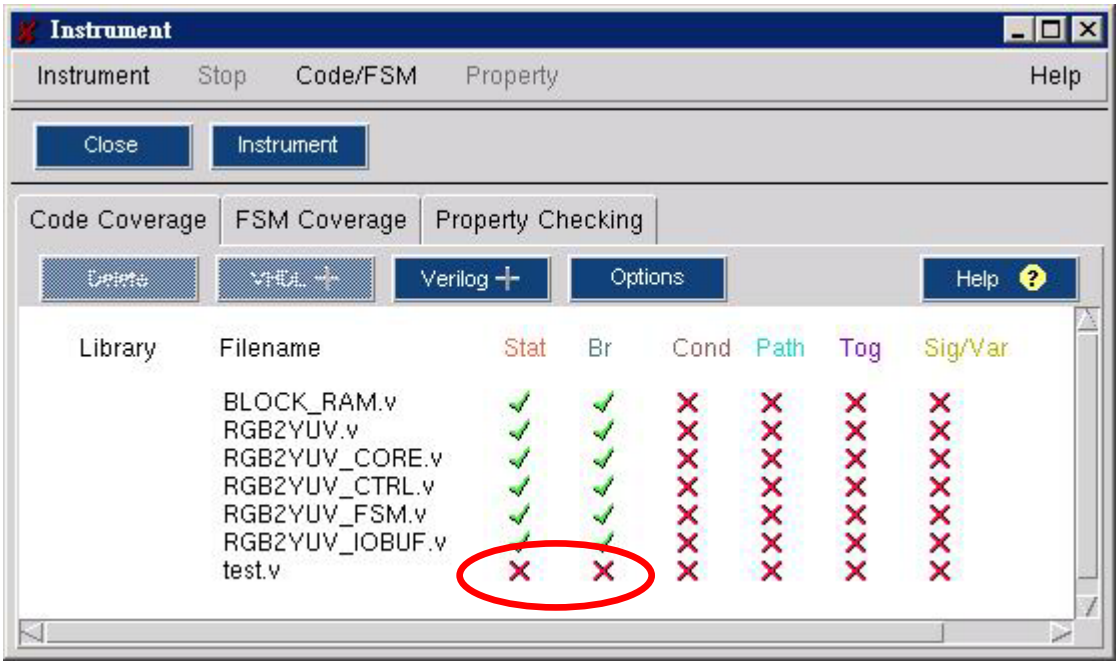
3. In the VN-Cover flow diagram, ***select "Set Simulator".*** In the **Select Simulator** window, ***select "Cadence Verilog-XL 3.1-3.4 (loadpli1)"*** as your simulator for coverage verification and ***click "OK"***. You have to make sure this simulator is properly configured on your system and supports Verification Navigator's PLI plug-ins.

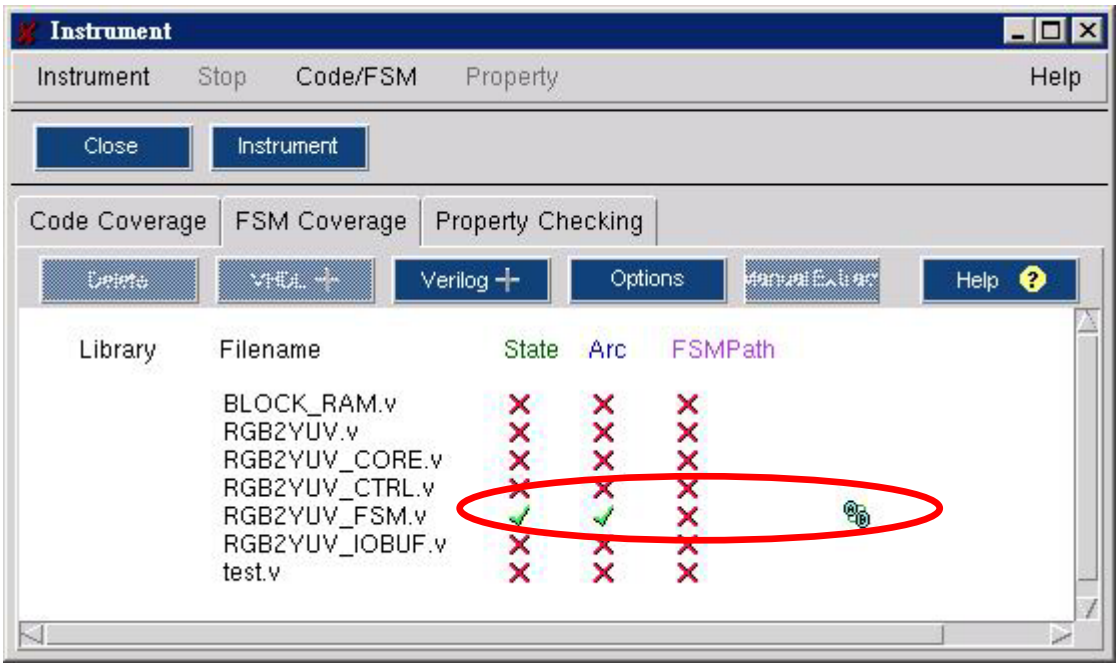


4. Now, select the files to be verified. In the flow diagram, ***select "VN-Cover/VN-Property DX Options"***. A window entitled **"Instrument"** pops up.

5. In the Instrument window, ***click the "Verilog+"*** button, and then select the verilog files you want to verify. Note that you have to include your test bench in the file list since the coverage verification is a dynamic verification that requires stimulus. You can load the file named "***test.f***" which automatically loads all necessary files.

6. ***Disable all verification*** on ***"test.v"***, which we are not interested. Do this action for both the **"Code Coverage"** tab and the **"FSM Coverage"** tab. In addition, ***disable all verification on non-FSM modules*** in the **"FSM Coverage"** tab.





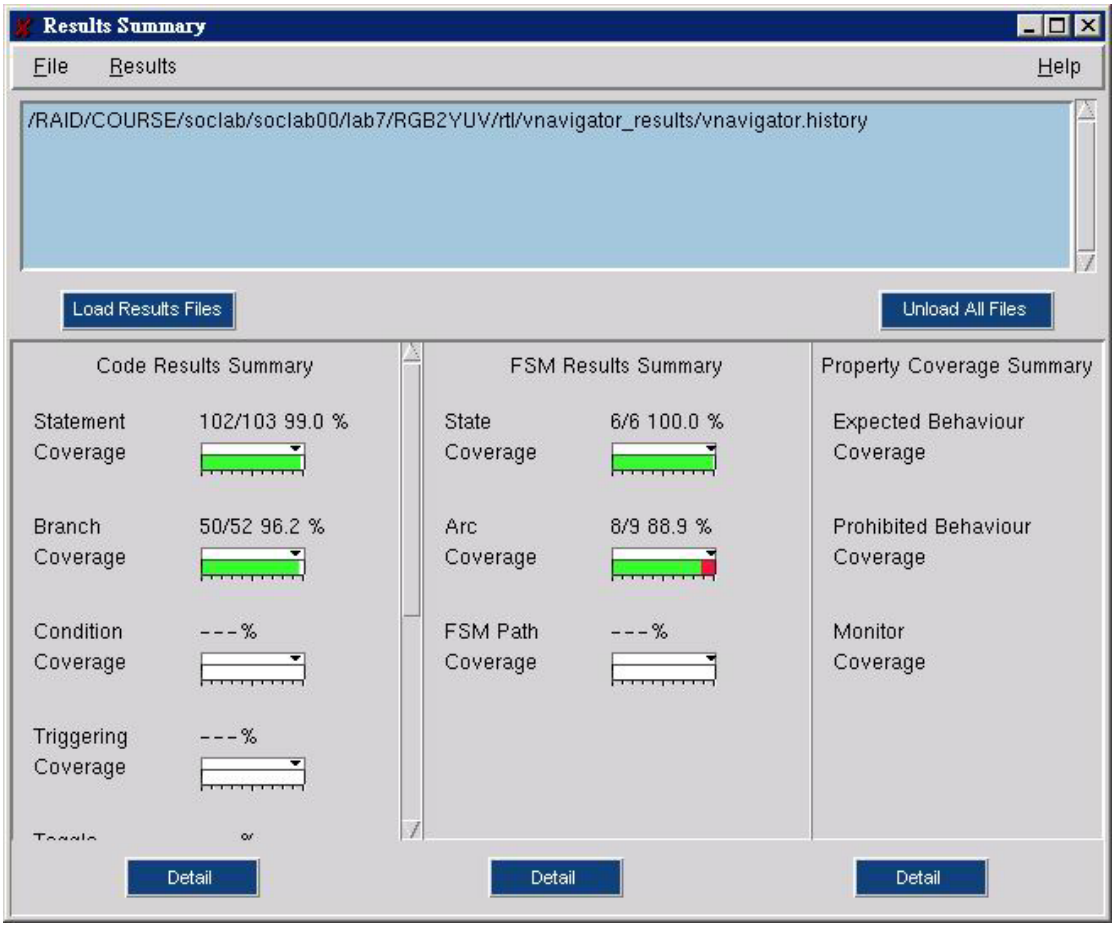
7. Now, ***click on the "Instrument"*** button. VN adds some extra commands to your verilog files which will generate the coverage results. A dialog box requesting the file name of command file will pop up. Use the **default name** is ok.

8. Next, ***click the "Simulate"*** button in the flow diagram. In the **"Simulate"** window, select the ***"Add"*** button on the top, and select the command file you just created in the previous step.

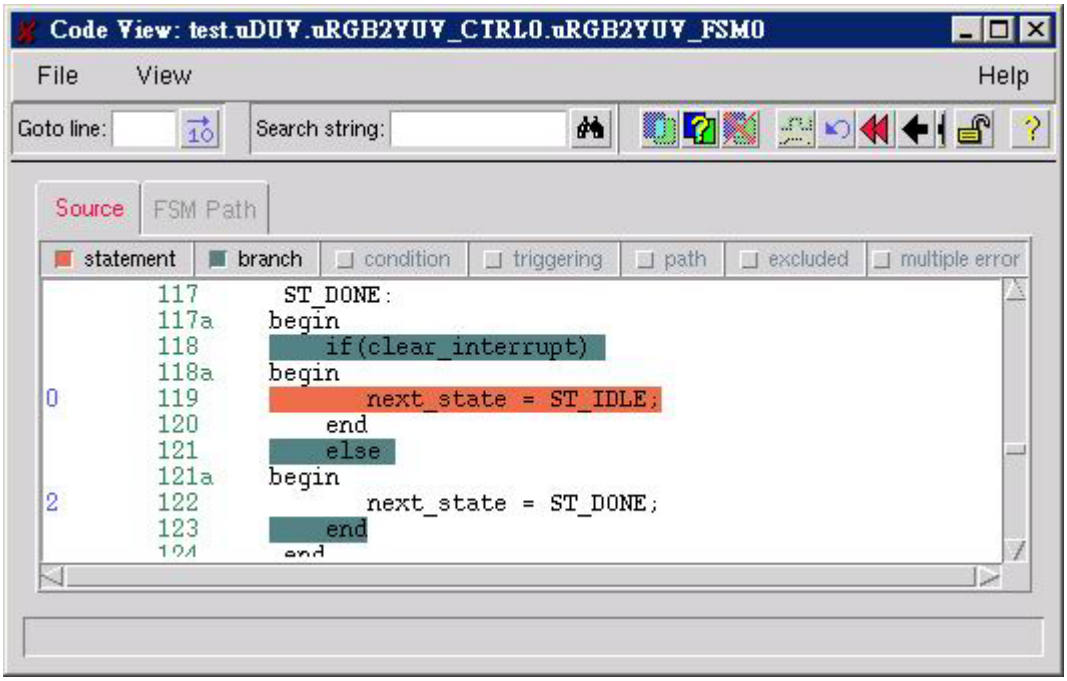
9. ***Click the "Simulate"*** button to generate coverage verification results.

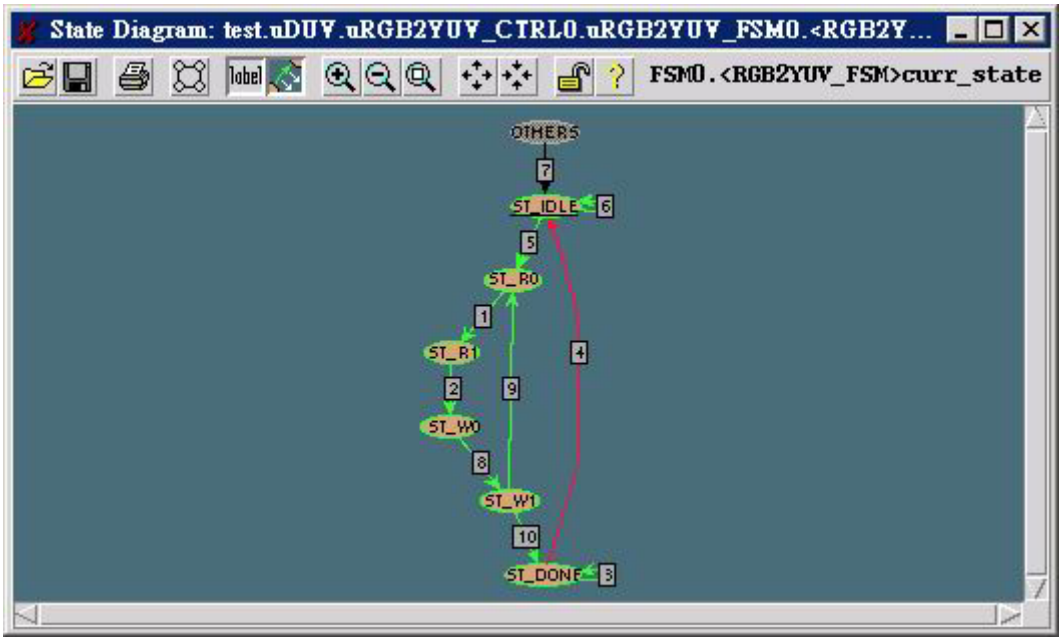
10. ***Click the "Results"*** button in the flow diagram. ***Click "Load Result Files"*** and load the **vnavigator_results/vnavigator.index" file**.

11. You will see the coverage verification summary of the design.



12. ***Click the "Detail"*** button to find out those uncovered statements or FSM states.

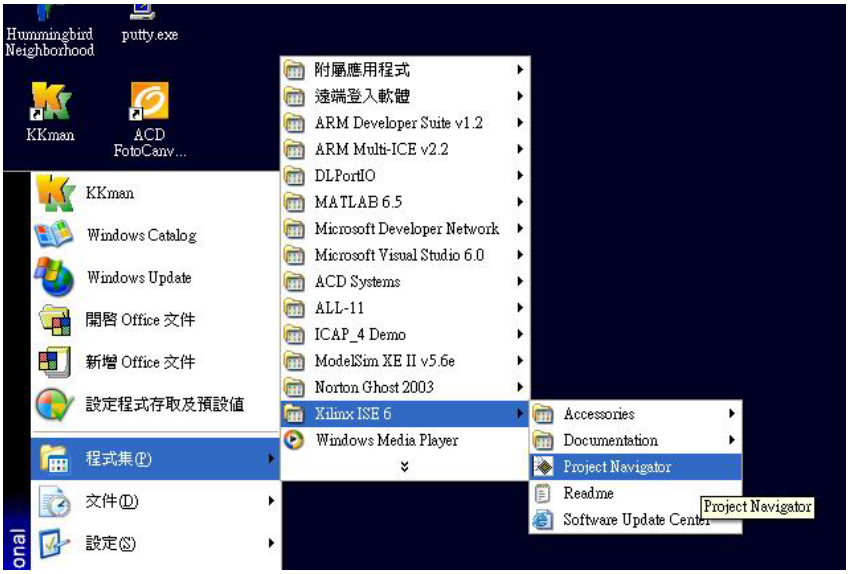




13. Refine your test vectors. Try to increase the coverage results.

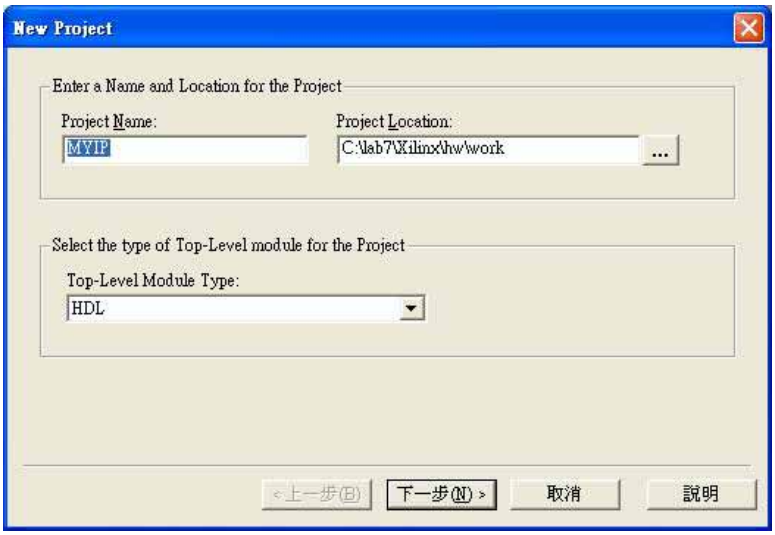
3.2. Design Implementation with Xilinx ISE

1. Launch Xilinx ISE 6.1i



2. ***Select "File → New Project"*** from menu. In the dialog, enter the following options.

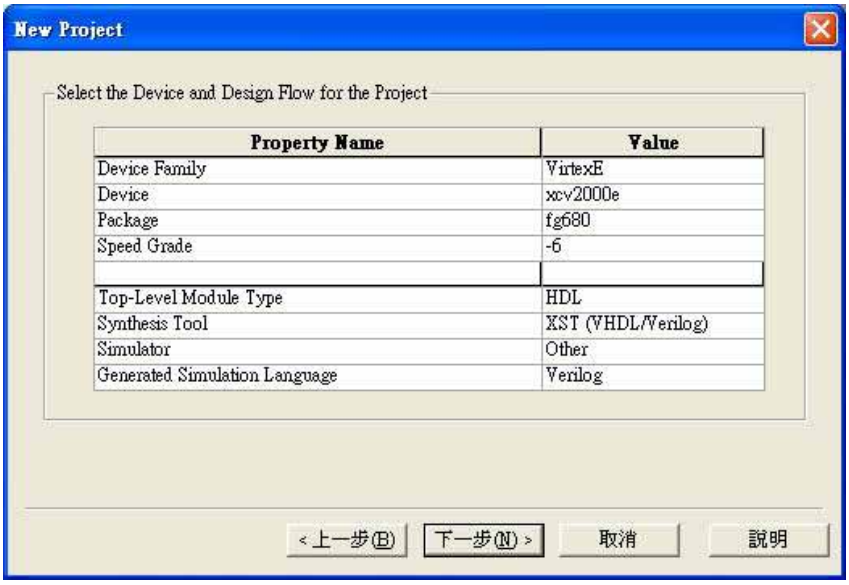
- Project Name: ***MYIP***
- Project Location: ***C:\lab7\Xilinx\hw\work***
- Top-Level Module Type: ***HDL***



Click "Next Step", and enter the following options.

- Device Family: ***VertexE***
- Device: ***xcv2000e***
- Package: ***fg680***
- Speed Grade: ***-6***

- Top-Level Module Type: ***HDL***
- Synthesis Tool: ***XST (VHDL/Verilog)***
- Simulator: ***Other***
- Generated Simulation Language: ***Verilog***



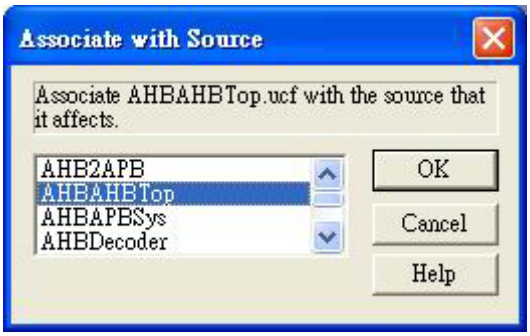
Click "Next Step" twice, we will skip the step adding new source. Instead, we add existing sources.

Click "Add Source", and add the following source files.

In ***C:\lab7\Xilinx\hw\verilog***:
"AHB2APB.v""
"AHBAHBTop.v"
"AHBAPBSys.v"
"AHBDecoder.v"
"AHBDefaultSlave.v"
"AHBMuxS2M.v"
"AHBZBTRAM.v"
"APBIntcon.v"
"APBRegs.v"

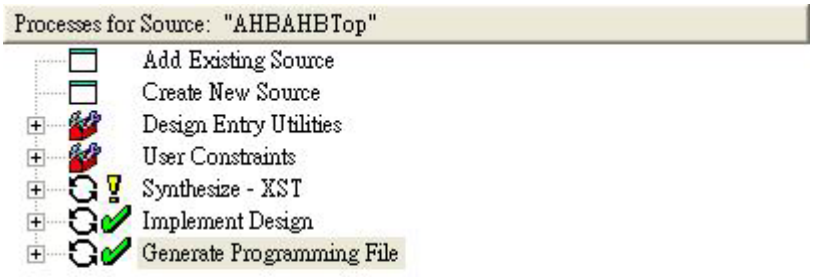
Finally, ***click "Finish"*** to create the project.

3. We want to constraint the implementation. So, ***select "Project → Add Source"*** from menu. And add the constraint ***C:\lab7\Xilinx\hw\work\AHBAHBTop.ucf*** as source file. You have to associate this file to a module. Select the top module: ***AHBAHBTop*** to be associated.



4. In the **"Module View"**, select the top module ***"AHBAHBTop"***, and then run the processes for the module in the following order:

- Synthesis -XST
- Implementation Design
- Generate Programming File



5. ***Download the generated bitstream*** and use the test software in ***C:\lab7\sw*** to test the design.

4. Exercises

1. Try to improve the coverage of RGB2YUV design. You must achieve 100% code coverage and 100% state coverage and 100% arc coverage.

5. Reference

1. http://twins.ee.nctu.edu.tw/courses/ip_core_02/index.html
2. http://twins.ee.nctu.edu.tw/courses/ip_core_01/index.html
   http://www.arm.com/ 4. Integrator ASIC Platform [DUI_0098B_AP_UG]
3. System Memory Map [DUI_0098B_AP_UG 4.1]
4. Counter/Timer [DUI_0098B_AP_UG 3.7, 4.6]
5. Interrupt [DUI_0098B_AP_UG 3.6, 4.8]