PLD Design Flow: FPGA, Verilog

Summary

This document describes the step-by-step process to create, simulate, and implement a Xilinx FPGA-based design using the Cadence/Orcad design suite for simulation and the Xilinx tools for implementation.

Required Software

- Xilinx Web Pack FPGA implementation software
- Cadence NC-Simulator

Required Hardware

- Spartan2E FPGA Development Board (checked out from the instrument room)

Boldface type indicates comments that are particuarly critical to the success of your project and/or prevents the destruction of your device.

What to Do:	How To Do It:
Create a Conceptual Design	
Draw I/O block diagram of your circuit (i.e. rectangle with pins sticking out and a name for each pin/signal)	Use a piece of paper and pen or pencil to do this! You may need to clarify some details using a manner that the design can easily be changed before jumping in with the computer tools.
Draw a diagram of your circuit components and their interconnections	See above. The diagram will help you visualize the hardware before writing the HDL.
Enter Your Design:	
Set up design directory	Create a new folder for your design files.
Run the Cadence NC-Simulator software	Run Programs | Cadence Design Systems | Affirma Design and Verification | NCLaunch
Change your working directory	File | Set Design Directory Change the design directory to the folder you created above (only the first time your run the program) create a cds.lib file

What to Do:	How To Do It:
Create a new Verilog file for the circuit description	Select “File | EditNewFile” If you are not already there, navigate to where you want the file to be saved and enter a file name (use the .v extension). Click on the “Save” button. This will bring up notepad. Begin writing your Verilog code. (possible bug depending on your set-up: if you click OK on the pop-up window, and notepad closes, reopen the file by selecting the file in the directory window and clicking on the “Browsers | Edit the currently selected file” button on the Toolbar.)
Enter your Verilog circuit description	Enter text as needed. Save your file (use Ctrl-S or push the save-to-disk icon). <u>Example Verilog circuit description:</u> *module Demo (a,b,c);* *input a;* *output b;* *output [2:0] c;* *assign b = ~a;* *assign c = {a,a,a};* *endmodule*

What to Do:	How To Do It:
Compile the circuit description to check for syntax errors.	Select the file under the working directory (in the left-hand side of the window). Click on the “Tools | Launch **Verilog** Complier” button on the Toolbar. The compiled circuit should appear under the “worklib” directory in the right-hand side of the window. You may need to expand the “worklib” directory to see it. If it does not appear, refresh the screen. Error messages will show up in red in the “nclaunch>” field at the bottom of the window. **All errors MUST be corrected before proceeding.**
Verify Your Design Using Simulation	
Write a testbench file instantiating your Design	Create a testbench file (Select “File | EditNewFile” then reopen using the “Browsers | Edit the currently selected file” button on the Toolbar.). Write the testbench file and save. Compile the testbench file to check for syntax errors (“Tools | Launch **Verilog** Complier” button on the Toolbar.)
Elaborate the circuit description to turn the Verilog description into a netlist that can be simulated.	Expand the “worklib” directory on the right-hand side of the window. Select the testbench filename, and the “Tools | Launch Elaborator” button on the Toolbar should activate. Click on the “Tools | Launch Elaborator” button on the Toolbar. Error messages will show up in red in the “nclaunch>” field at the bottom of the window. **All errors MUST be corrected before proceeding.**

What to Do:	How To Do It:
Simulate your design	Open the “Snapshots” directory on the right-hand side of the window. Select the testbench file and the “Tools | Launch Simulator” button on the Toolbar should activate. Press the “Tools | Launch Simulator” button and the simulator window should appear. Press the “Play” button to run the simulation. Binary results will be shown in the text field at the bottom of the window.

What to Do:	How To Do It:
View waveforms of your simulation.	Note: You must have the two following lines in your testbench in order to obtain waveforms: *$shm_open("waves.shm");* *$shm_probe("AC");* To view waveforms, choose the waveforms that you want to diplay from the testbench file displayed at the top of the window. Multiple signals can be selected by holding the ctrl key down. The waveforms will appear in the order that you select them. Click on the “Waveform View” button on the tool bar. This will bring up the waveform viewer. All of your signal names should appear. If you have already run your simulation, to get the waveforms to appear, you must rerun your simulation. Go to “File | Reset Simulation” to rerun the simulation. The waveforms should appear. Alternatively, click on the “Tools | Display waveforms” button on the toolbar in the simulator window (waveforms icon). A waveforms window will appear. Select “Windows | NewDesignBrowser”. A new window will appear. Select the testbench file. All of the inputs, outputs, and internal nodes/variables will appear in the right-hand side of the window. Select the signal that you want to observe. Multiple signals can be selected by holding the ctrl key down. The waveforms will appear in the order that you select them. Press the waveform icon on the Tool Bar. The waveforms should appear in the waveform window. Signals may be removed or grouped into busses by right clicking on the signal name.

What to Do:	How To Do It:
Iterate until your design is correct	If the simulation is not correct, you must go back to your .v file and modify to correct the behavioral problem. The modified files must be recompiled and the testbench file must be relaunched. Do not reinvoke another simulator/waveform viewer. All of you settings will be retained if you go to the currently open simulator window and select "File | Reinvoke Simulation" to resimulate using your new file. You may resimulate an unchanged file by selecting "File | Reset Simulation". You may single step through a simulation by using the small "Play" buttons. This sometimes help you in debugging your circuit description. Leave all of your windows open! If you resimulate, your windows will automatically update.
Do NOT proceed until your design simulates correctly according to your design specs!!	
Convert Your Design Files to Produce a bitstream File:	
Make a new directory for your project.	It is recommended that you also create a new folder for your project, since the software will generate many files for your single project. You cannot create a new directory inside the Xilinx package, so this must be done before running it using your windows browser of choice (Windows Explorer, etc.) Do not have any spaces or special characters in your project name or folder name.
Start Xilinx Web Pack	In Windows, do Start | Programs | XilinxISE5 | Project Navagator

What to Do:	How To Do It:
Start a new project in Xilinx	Select “File | New Project” (Note that Xilinx will open the most recently-used project, so you want to begin from scratch here). Browse to and select the directory you just created for your project. Then, enter a name for your project. Specify the device type by entering the following fields: Device Family = Spartan2E Device = XC2S200E Package = pq208 Speed = -7 Design Flow = XSTVerilog
Add your Verilog description to the project	Select “Project | Add Source...” in the Project Navigator window. Navigate to the folder that contains “circuitname.v”. Do NOT add the testbench file. NOTE: It is possible to write a Verilog description that simulates successfully, and yet is incompatible in some way with the hardware synthesis or implementation steps in Xilinx ISE. This is not necessarily a shortcoming of ISE, but rather is due to the fact that Verilog is inherently a simulation and modeling language, and hardware synthesis tools only deal with a subset of the language. The bottom line: you need to learn how to write synthesizable descriptions.

What to Do:	How To Do It:
Generate a partial implementation to automatically assign the pins	Select your top-level module. Double-click the “Synthesize” process in the second window on the left. NOTE: A green checkmark on a process means all is well; a yellow exclamation point means one or more warnings were generated; a red X means a fatal error. You can find the error messages in the bottom scrolling window at the bottom of the Project Navigator. If you have errors, you can edit your Verilog files in the Xilinx ISE environment. Double-click the .v file in the “Sources” panel to open it in the editor. Synthesis will pick up any syntax or synthesis errors, but major modifications of the program may result in incorrect operation. You need to resimulate to catch these errors.
Implement the design	Select your top-level module. Double-click the “Implement Design” process in the second window on the left.
Generate a ucf file	Expand the “Implement Design | Place and Route | Back-annotate Pin Locations” directories in the process window. Double click on “View Locked Pin Constraints” to view the pins report. Save the file as a “user constraints” file (filename.ucf)
Modify the ucf file	Xilinx randomly chooses pins to connect your inputs and outputs. These pin assignments have nothing to with the I/O board or the breadboard. You must find the proper pin assignments and modify the ucf file to reflect the proper pin assignment. A file that will determine the proper pin assignments can be found on the Lab Resources web page.
Add the ucf file to your project	Select “Project | Add Source | filename.ucf

What to Do:	How To Do It:
Reimplement the design	Select the top level file. Double-click the “Implement Design” process again.
Generate the bitstream	Double-click the “Generate Programming File” process.
Download the bitstream file to Spartan board:	
Connect the board	Ensure that the Spartan board is connected to power. Ensure that the Spartan board is connected by the parallel cable to the computer’s parallel port. Ensure that all desired peripheral boards are connected. Connect the I/O board to either connectors A/B or E/F. Connect the bread board to connectors C/D. **BE CAREFUL when removing the I/O board and breadboard board so that you don’t bend the pins!!** Using a penny to gradually pry one side and then the other will help prevent this from happening. The “JAG” switch must be set to “JTAG” to program the board. It must be set to “port” to use the parallel port from the computer.
Configure the device	Expand the “Generate Programming File” directory in the processes window. Double click on the “Configure Device” process. Step through the series of windows that appear: choose “configure device” next window, choose “boundary scan mode” next window, choose “automatic connect ...” a window should tell you that one device is connected at the “Assign New Configuration File” prompt, choose the filename.bit file

What to Do:	How To Do It:
Dowload the .BIT file	Right click the Xilinx device and choose “Program” If you update your verilog files, you must either close the “Configure Device” window and reopen it or right click on the Xilinx device and choose “New Configuration”
Test Your Design	
Apply external stimulus	Test and debug your circuit on the DBB1 breadboard before you connect it to the D2E system board. **Ensure that any waveforms you apply to the circuit swing from 0 to 3.3 or 5 volts BEFORE applying them to your circuit. Otherwise, you may destroy the device!** Use 400- to 500-ohm resistors instead of wire to establish the connections between your analog circuit and the connector socket leading to the FPGA Do not use the red power bus on the DBB1 breadboard for any purpose, unless your circuit uses a +3.3 VDC supply and does not require a large amount of current. Be sure the use the black ground bus for all of your ground connections to prevent ground loops.

To Obtain Resource Information About Your Design in the Xilinx Software

- You can learn the resource usage of your design such as CLB count (configurable logic blocks) by opening the “Map Report” located in the “Processes” hierarchy under “Implement Design | Map | Map Report”.

- You can see how your design was routed on the chip itself by looking in the “Processes” hierarchy under “Implement Design | Place & Route | View/Edit Placed Design”. Just look, don’t actually change the design!!