From Concept to Bitfile: Step-By-Step Instructions
(Spartan-3 version)

Edward R. Doering
ECE Department
Rose-Hulman Institute of Technology

Summary: This document describes the step-by-step process to create, simulate, and implement a Xilinx FPGA-based design.

Required Software: (versions as of 01 Mar 2008)

1. Xilinx ISE (or Xilinx ISE WebPACK) version 9.2
2. Cadence NC-Simulator (NCLaunch) version 5.4
3. Verilog Module Builder (vmb) version 1.0 at `http://www.rose-hulman.edu/PLD_Oasis` "Software" section
4. LabVIEW Run-Time Engine version 8.5 (required to run 'vmb')
5. UCF Generator for Digilent Boards at `http://www.rose-hulman.edu/PLD_Oasis` "Software" section
6. Digilent Adept suite version 1.8

Software Installation Instructions: Installation instructions for all required software packages are available at the "Software" section of PLD Oasis.

Step 1 of 7 — Create your conceptual design:

1. Draw the I/O (input/output) symbol of your overall circuit, i.e., draw a rectangle with input pins on the left side and output pins on the right side, and write a name for each signal pin.
2. Draw the high-level (register-level or gate-level) diagram of your design. This diagram will help you visualize the hardware when you write your circuit descriptions.
3. Determine the assertion levels (active high or active low) of your I/O signals. Consult the UCF Generator spreadsheet for the Spartan-3 to learn about the external circuit devices that connect to the FPGA.

Step 2 of 7 — Create a new Xilinx ISE project:

1. Start Xilinx ISE Project Navigator, and create a new project by selecting "File | New Project..." (note that Xilinx opens the most recent project on startup, so you want to begin from scratch here).
2. Select a single word for the name of your project, and select the directory for your project files. Choose a directory that is NOT in the `C:\Xilinx` folder tree (you do not want to run the risk of corrupting your software installation) and whose name contains no spaces. Ensure that "HDL" is selected for the top-level source type, and click "Next".
3. Enter the following information:
   - Product Category = All

- Family = Spartan3
- Device = XC3S200 // XC3S500 for Nexys boards
- Package = FT256
- Speed Grade = -4
- Synthesis Tool = XST (VHDL/Verilog)
- Simulator = ISE Simulator (VHDL/Verilog)
- Generated Simulation Language = Verilog

Leave the remaining check boxes at their default values, click the "Next" button three times, and click "Finish."

Step 3 of 7 — Enter your Verilog circuit description and testbench:

1. Start Verilog Module Builder (vbm).
2. Activate the "Context Help" window by typing Ctrl+H, and then hover your cursor over each `vmb` input device to learn more.
3. Enter the module name. The testbench module uses this name with `_TB` appended.
4. Enter the module description and other documentation in the lower left corner. The description can be arbitrary text.
5. Enter the input signal names as well as the generated signal names. Attach a generated signal to an output port by prepending an asterisk to the name.
6. Enter the testbench input simulation code.

   NOTE: Watch the "code preview" windows as you type to see the Verilog code you are creating. "vmb" does not parse or interpret your code, but watching the preview window can help you reduce typing mistakes.

7. Click "Save" or press F5, navigate to the Xilinx project folder you created in the preview step, and save the `vmb` source file. The Verilog `.v` file is simultaneously created, and always updates whenever you save your `.vmb` source file.
8. Leave the `vmb` application open for future use.
9. Go back to Xilinx Project Navigator and identify the "Processes" panel on the left side. Double-click "Add Existing Source" and select the `.v` file created by `vmb`. Click "OK."

Step 4a of 7 — Perform functional verification (simulation) in Xilinx ISE: Skip to Step 4b if you prefer to use the Cadence NC-Sim simulator.

1. Identify the "Sources" panel above the "Processes" panel. Change the "Sources for" entry to "Behavioral Simulation."
2. Single-click the testbench module (the module name ending with `_TB`).
3. Expand the hierarchy of "Xilinx ISE Simulator" in the "Processes" window.
4. Double-click "Check Syntax." If any errors or warnings appear, return to `vmb`, make corrections, click "Save" or press F5 to update the `.v` file, and try "Check Syntax" again.

   IMPORTANT: Do *not* edit the Verilog .v file in Xilinx ISE directly! Any changes you make to the file will be lost the next time you save from "vmb." You may, however, find the syntax color coding in Xilinx ISE to be helpful as a debugging aid.

5. Double-click "Simulate Behavioral Model" to launch the simulator. The `.v` file will open with a yellow cursor parked at the `$stop` line in the testbench module. Select the "Simulation" tab to show the waveforms. Right-click in the waveform window and choose "Zoom Full View" to ensure that you see the complete time range you specified in your testbench.

6. Iterate on your design as follows:

   (a) Modify the `vmb` source file, then click "Save" or press F5 to update the Verilog `.v` file.

   (b) Return to Xilinx ISE, select the "Processes" tab (lower left), right-click on "Simulate Behavioral Model," and select "Rerun All." Click "Yes" twice, then click the "Simulation" tab to view the updated waveforms.

Step 4b of 7 — Perform functional verification (simulation) in Cadence NC-Sim: Go back to Step 4a if you prefer to use the Xilinx ISE simulator.

1. Start the HDL simulator by running Cadence NCLaunch (you need to be connected to the RHIT network now).

2. Switch to the "multiple step" mode by selecting "File | Switch to Multiple Step." If the menu only has the "Switch to Single Step" entry, then you are already in the multiple step mode.

3. Use the same design folder as the Xilinx project folder you created in Step 2: Select "File | Set Design Directory" and either key in the folder name directly in the top line, or click the "..." button to navigate to the folder. Also click the "Create cds.lib File" button, then click "OK." You should see your two Verilog modules appear in the NCLaunch directory browswer panel.

4. Click and drag the cursor over the `.v` file, then click the "VLOG" button to compile the files.

5. Expand the "worklib" hierarchy in the right-side panel, single-click your testbench module, then click the "Elaborator" button (icon looks like a paper clip on a sheaf of papers).

6. Expand the "Snapshots" hierarchy, select the testbench module, then click the "Launch Simulator" button (icon looks like a paper airplane). Two new windows will open at this point.

7. Go to the "Design Browser" window, select the testbench module, then click the "Send to Waveform" icon in the upper right corner. The waveform window will appear. Press the "Run Simulation" button (looks like the "play" button on a CD player) to see the waveforms for your circuit (click the "zoom full" button (icon has an "equals" sign) in the upper right corner to fit the entire simulation time into the window).

8. Begin the simulation process by single-clicking your testbench module in the "Design" panel (on the right side) then click the paper airplane icon to launch the NCVerilog program.

9. Iterate on your design as follows:

   (a) Modify the `vmb` source file, then click "Save" or press F5 to update the Verilog `.v` file.

   (b) Re-run the simulation by selecting the SimVision application, then select "Simulation | Reinvoke Simulator," then click the "play" button in the simulator.

10. The waveform display may be customized and saved:

    - Right-click on the signal name in the waveform window; more specifically, right-click in the "Cursor" column of the signal name. The "Radix" option can be used to select binary, decimal, hexadecimal, and other display modes for buses (bit vectors)
    - Trace order can be adjusted by middle-clicking on a signal name, dragging to the new location, and releasing

- Additional signal traces at any level in the hierarchy can be added: select the module of interest from the "Design Browser" window, select the desired signal(s), then click "Send to Waveform" button.
- When you have invested a significant effort getting the waveform traces positioned and configured, you can save the configuration to a script file which can be run later to re-build your display. Select the waveform display window, the select "File — Save Command Script" and either accept the default filename or choose your own (note that the default file folder is "My Documents" rather than your design folder). In the future you can retrieve the display by selecting "File — Source Command Script."

IMPORTANT: Do *not* proceed until you are 100% confident that your design performs according to specifications! Debugging is much easier and faster in the simulator. Moreover, you will be following the same design verification method practiced by engineers in industry.

NOTE: It is quite easy to write a Verilog circuit description that simulates successfully, and yet is incompatible in some way with the hardware synthesis or implementation steps to follow 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 deal only with a relatively small *subset* of the language.

The bottom line: You need to learn how to write Verilog HDL using *synthesizable* constructs.

Step 5 of 7 — Synthesize your design to a bitstream file:

1. Return to Xilinx ISE, which should still be open.
2. Change "Sources for" to "Synthesis/Implementation," and then single-click your top-level module in the "Sources" window in the upper left corner. Identify the "Processes" window in the lower left corner, and expand the "User Constraints" hierarchy button by clicking the plus sign. Double-click the "Edit Constraints" process, and say "Yes" to the prompt to create a UCF file.

   IMPORTANT: You *must* select the top-level module *first* before you create the constraints file, because it is possible to associate a UCF with submodules.

3. Use the "UCF Generators for Digilent Boards" (Spartan-3 version) to create the user constraints file, or UCF (refer to the "Required Software" section at the beginning of this document for details). The UCF generator is an Excel spreadsheet that translates your Verilog port names into FPGA pin assignments. Instructions for using the spreadsheet are included at the top of the spreadsheet. Copy all of the text from the yellow box in the UCF generator to the editor in ISE, then save the file in ISE.

   IMPORTANT: Copy and paste *all* of the available text in the yellow box. The "CONFIG PROHIBIT" lines will help prevent accidental improper connections that could permanently damage the FPGA!

   An explanation is helpful here: Xilinx ISE knows nothing about external devices connected to the FPGA. In the absence of any constraints, ISE will randomly assign your Verilog ports to FPGA pins. The random assignment will obviously not match the devices that are pre-wired to the FPGA (by virtue of the FPGA being installed in the Digilent Spartan-3 board). Problems arise when your design assigns a signal output to a pin that is already connected to a signal generating device such as a switch or pushbutton. When the signals conflict, the resulting short circuit may permanently damage the FPGA, rendering the entire board non-functional. Therefore, it is *your* responsibility to ensure that ISE places your Verilog ports at the appropriate pin locations, and this is the purpose of the UCF file.

The UCF generator spreadsheet helps you to make correct pin assignments. However, if you later add an additional output port to your module, but then forget to make the associated pin assignment in the UCF file, the "CONFIG PROHIBIT" line will cause the place-and-route tool to fail, drawing your attention to the problem. Thus, it is vital that you include *all* of the "CONFIG PROHIBIT" constraints, otherwise the new output may get assigned to a signal generating device.

NOTE: Consider saving the UCF generator spreadsheet to your project folder so that updates and changes can easily be made later on.

4. Start the hardware synthesizer by first selecting your top-level module in the upper left "Sources" panel, then double-click the "Synthesize" process on the lower left "Processes" panel.

   NOTE 1: A green checkmark on a process means all is well; a yellow exclamation point means one or more warnings were generated, and a red X means a fatal error. You can find the warning and error messages by selecting the appropriate tabs at the very bottom of the Project Navigator window. You can also find the error messages in the report file associated with each process in the lower left "Processes" window (you need to expand the hierarchy to find the reports).

   NOTE 2: Ignoring yellow warnings in synthesis will almost always lead to fatal errors in the implementation steps to follow, so find and correct the problem(s) now. Always begin by attacking the *first* problem mentioned in the warning/error list, since many (or possibly all) of the subsequent messages may stem from the first problem.

   NOTE 3: Once you have corrected a problem, rerun by right-clicking the "Synthesize" process and selecting "Rerun," or simply double-click the "Synthesize" process again.

   NOTE 4: Expand the "Synthesize" process to find the synthesis report (includes the device utilization summary, so you can find out how much of the FPGA resources were used; the device utilization summary is also available by selecting the "Design Summary" tab (it has a green sigma character) at the bottom of the editor window).

   The "RTL Schematic" viewer, also available under the "Synthesize" process, gives you a schematic view of the hardware that was synthesized from your Verilog description, which can be very helpful when troubleshooting your design. That is, you can find out what hardware was actually synthesized from your description, which might be very different from what you thought your were describing!

5. Start the implementation tools by double-clicking the "Implement Design" process on the lower left "Processes" panel.

   When finished, verify that your pin assignment is correct by expanding the "Implement Design" process and then the "Place & Route" process, then double-click "Pad Report." Look for your Verilog port names in the second column, and the pin numbers in the first column.

   IMPORTANT: If your UCF file does not exist, or is not associated properly with your top-level module, then you will end up with a random pin assignment. At best this will appear as very strange behavior on the Spartan-3 board, and at worst can cause permanent damage to the board! If you encounter totally unexpected behavior with your hardware, *immediately* remove power from the board, then check the pad report.

6. Right-click on the "Generate Programming File" process, select "Properties," select the "Startup Options" tab, and specify the "JTAG Clock" option for the "FPGA Start-Up Clock" (you only need to do this step one time for a given project).

7. Create the bitstream (`.bit`) file by double-clicking the "Generate Programming File" process.

   NOTE: In general you can run all previous processes by double-clicking the desired final process. For example, if you make a change to the `.v` file, the implementation data will be cleared (all the checkmarks vanish), and double-clicking the "Generate Programming File" process will cause the synthesis and implementation processes to run automatically.

   If you need to re-generate all project files from scratch, perhaps because you suspect something is "fishy" about your results, select the "Project" pulldown menu at the top of the main window and choose "Cleanup Project Files."

Step 6 of 7 — Download the bitstream file to the FPGA:

1. Begin with power disconnected to the Spartan-3 board.
2. (Optional step) You can self-test the board to ensure that it is working properly: Ensure that all three blue shorting blocks on jumper J8 are connected, then connect power to the Spartan-3. If factory-loaded bitfile has not been changed, you will see the word "PASS" on the display, and the discrete LEDs will blink a pattern for about 10 seconds, after which the display will scroll the phrase "Spartan-3 Starter Board." Once you are finished, disconnect power to the board.

(optional) To test the Nexys board, follow the instructions located on the box.

3. Remove the outer two shorting blocks (M0 on the left, M2 on the right) and replace them so that the top pin is exposed, that is, move the shorting blocks down one pin. Do NOT simply remove the shorting blocks and toss them in the box or elsewhere! The shorting blocks are easy to lose, and replacing them will be an unnecessary hassle!

4. Connect the black shrink-wrapped connector of the JTAG-USB cable to the Spartan-3 board. Be very careful to orient the connector according to the printed signal names, since reversing the connection will damage both the board and the cable!

   NOTE: The black end of the JTAG-USB cable contains electronics that can get quite warm to the touch — this is considered normal.

5. Connect the USB-side of the JTAG-USB cable to your computer, then re-apply power to the board. You should see the green "POWER" LED active, and the eight red discrete LEDs active.

6. Start the Digilent "ExPort" application, which is part of the Digilent Adept suite.

7. Click the "Initialize Chain" button to establish communications between your computer and the Spartan-3 board. You should see a "scan chain" diagram appear showing the FPGA and the ROM. If all is well, skip the next step.

8. Click the "Configure" button in the upper left corner, select "Delete," then select "Add Module." Choose the USB tab. You should see something similar to "DCabUsb" followed by a number in the "Connected Modules" panel. If you do not, then re-install the driver for your JTAG-USB cable. Single-click the connected module, click "Add" and then "Done." Click "Save" and then "Close." Now repeat the previous step to connect to the Spartan-3 board.

9. Troubleshooting step only: Click the "Browse" button next to the FPGA symbol, navigate to the project folder that contains your bitstream file, then double-click the bitstream file. Notice that your file shows up next to the FPGA, indicating that you have selected a file to

download to the FPGA. You can add additional bitstream files using the "Add File" button in the lower left, and then select your desired file with the pull-down menu next to the FPGA. This is useful when you need to try several bitstream files in quick succession.

10. Click the checkbox next to the ROM to remove it from the scan chain.

11. Click "Program Chain" or press F4. You will see a progress indicator bar, and then a confirmation message if the bitstream file downloaded correctly. Immediately after download is complete, the green "DONE" LED will light on the Spartan-3 board, and your digital system is ready for testing.

12. When you need to make changes to your Verilog module, switch to `vmb`, edit and save, switch to Xilinx ISE and double-click "Generate Programming File" to create a new bitstream file, then switch back to ExPort and press F4 download the new bitstream file to the Spartan-3 board. Repeatedly downloading bitstream files may occasionally get the Spartan-3 board confused. If it acts funny, press the "PROG" button above the power LED to reset the board, then download your bitstream file again.

Step 7 of 7 — Test your design:

1. Operate the input devices and observe the behavior of the output devices. If everything works OK, then celebrate!

2. If the board seems dead, or acts very strangely, ***immediately*** *remove power to the board!* Your pin assignments are probably missing or otherwise incorrect. Without a UCF file the implementation tools will make a random pin assignment. Go back to the beginning of Step 5 to make a correct pin assignment, and be sure to check the pad report to verify the correct assignment of your signal ports.

3. Consider making a simple file (such as an inverter connected between a switch and an LED) that can be used as a sanity check on your board. Additional board tester `.bit` files are available at PLD Oasis.

Additional tips:

- If you need to change the target device once you have implemented the project (e.g., take your design from a Spartan family device to a Virtex device), right-click on the target device line in the "Sources" panel, select "Properties," enter your new information, and then re-implement your design by double-clicking the "Implement" process.

- You can remove files from your project by selecting the desired file from the "Sources" panel, right-clicking, and choosing "Remove." Note that your file does not get deleted from the filesystem, but simply becomes unknown to the Xilinx Project Navigator.

- If something seems "not quite right" about the behavior of the software tool, try selecting "Project | Cleanup Project Files" to start from the beginning.

- You can learn the resource usage of your design such as slice count and IO block count by opening the "Map Report" located in the process "Implement Design | Map | Map Report."

- In Xilinx ISE you can see a schematic representation of the hardware synthesized from your Verilog description by looking at the "RTL Schematic" viewer under the "Synthesize" process. Enable the "Keep Hierarchy" option to preserve the module boundaries of more complex designs; do this by right-clicking on the "Synthesize" process, select "Properties," select the "Synthesis Options" tab, then set "Keep Hierarchy" to "Yes."

- In NC-Simulator you can see a hardware block diagram of your Verilog description. Select your top-level module in the "Design Browser," then select "Send to Schematic."

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