Version for EDK 8.1i as of May 10 2006

Introduction

ISE is an integrated environment for developing your cores for the FPGA. The main GUI is Project Navigator and a number of other tools can be used in or launched from Project Navigator, such as CoreGen, HDL Bencher, and ModelSim. You will probably do the initial development and debugging of your cores using ISE before trying to connect them to a Microblaze in EDK (XPS).

Note that Xilinx tools dislike paths with spaces. Paths with spaces should be avoided like the plague when deciding where to install Xilinx tools, third party tools, and where to put your project files.

For additional information on using ISE, please refer to the Xilinx documentation. Some pointers to the documentation are given here.

Goals

- To gain a basic understanding of how to use ISE.
- To develop a simple core using ISE for use on the Xilinx Multimedia board.
- To use CoreGen IP in this design.
- To learn how to initialize memory.
- To use iMPACT to download the design to the board.
- To use the pushbuttons on the Multimedia board.

Requirements

Access to ISE 8.1i

Preparation

The documents linked here can be accessed online via the *HTML Collection* link:

`http://www.xilinx.com/support/sw_manual/xilinx8/index.htm`

and the pdf version is accessible via the following link:

`http://toolbox.xilinx.com/docsan/xilinx8/books/manuals.pdf`

available on the UofT EDK page under Xilinx Online Documentation. These tools are in the *Design Implementation* category.

1. Look through the link, *ISE Help*, in the Design Implementation tools list.

   `http://toolbox.xilinx.com/docsan/xilinx8/help/iseguide/iseguide.htm`

2. Skim through the *FPGA Design Flow Overview* to get a basic understanding of the parts of the design flow. The link to this is available on the *ISE Help* page described above:

   `http://toolbox.xilinx.com/docsan/xilinx8/help/iseguide/html/ise_fpga_design_flow_overview.htm`

3. Skim through the ISE Quick Start tutorial to get an idea of the additional capabilities of ISE. Access this through ISE Help menu in the ISE GUI or online at

   `http://toolbox.xilinx.com/docsan/xilinx8/books/data/docs/qst/qst0001_1.html`

4. In the Multimedia Board Users Guide, read the section on User Input and Output. You will be using the pushbuttons, User Input dip switches, and User LEDs in this lab.

Steps

1. Open Project Navigator.
2. Create a new project in a directory (without spaces!) by selecting File > New Project. When you specify a Project Name, a subdirectory for it will automatically appear in the Project Location box. In this document, we will call it "learn_ise". The top-level module type will be HDL. Click Next. The Device Family should be Virtex2. The Device should be xc2v2000. The package should be ff896. The speed grade should be -4. These correspond to the chip on the Multimedia board. The Top-Level Module Type should be HDL. The Synthesis Tool should be XST. The Simulator should be Other or ModelSim. The Generated Simulation Language should be Verilog. (This document assumes you are using either an UGSPARC PC or you have installed MXE for Verilog.) Click Next, Next, Next, Finish.
3. Unzip the m06.zip file in your project directory to get the files for this module.
4. Go to the menu Project > Add Copy of Source. Select the Verilog files you just unzipped and add them to the project. Choose Verilog Design File as the Source Type for each.
5. Go to the menu Project > Add Copy of Source. Select the learn_ise.ucf file you unzipped and add it to the project. Associate it with the learn_ise module.
6. The mem_init.coe file was created using Microsoft Excel and Memory Editor as described in the Xilinx Answer Record #11744. Go to the menu Project > New Source. Select IP (CoreGen & Architecture Wizard). Call it lookup_table. Make sure the Add to Project checkbox is checked. Click Next.
7. The Select Core Type dialog box will pop up. Expand Memories & Storage Elements. Expand RAMs & ROMs. Select Single Port Block Memory. Click Next. Click Finish.
8. A CoreGen GUI will pop up. Enter led_lookup_ip as the Component Name. Select Read Only as the Port Configuration. Enter Width of 2 and Depth of 1024 for Memory Size. This will create a ROM that has 1024 2-bit words. Click Next. Let CoreGen Optimize for Area in the Primitive Selection. Click Next. Click Next. In the Initial Contents box, check the Load Init File checkbox. Click on Load File and browse to and select the COE file that you copied. Click Generate. You should get the message "Successfully generated <led_lookup_ip> (Single Port Block Memory 6.30)".

   Note that you have just initialized a very small block of memory so it did not take very long. If you were to initialize something occupying > 50% of the on-chip block ram, expect that this could take upward of 10-15 minutes.
9. In your project directory you should now have a number of generated files for led_lookup_ip. The MIF file is the Memory Initialization File that can be used in behavioural simulations. The V and VHD files are for compiling at simulation time. The VEO and VHO files are the instantiation templates for the IP. (You could come up with that yourself, but copying and pasting from the template saves typing)
10. Next you will create a DCM for the module. The DCM in this module is mostly for demonstration purposes. You would really want to use a DCM when you need phase shifting or clock division. The feedback circuit works to line up or phase shift clock edges, as configured in CoreGen, and produces a "locked" signal when the DCM output has settled. You should NOT just put a clock through a T

flip-flop to generate another clock at half the frequency. That WILL cause timing headaches.

Create a DCM (Digital Clock Manager) IP in CoreGen by clicking on Project > New Source. Select IP (CoreGen & Architecture Wizard). Call it led_dcm. Make sure the Add to Project checkbox is checked. Click Next.

11. The Select Core Type dialog box pops up. Expand FPGA features & design, then Clocking. Choose Single DCM. Click Next. Click Finish. The Xilinx Clocking Wizard dialog box pops up. Enter 27 MHz as the Input Clock Frequency (This is one of the available clock frequencies on the Multimedia board). Ensure that RST, CLK0, CLKDV, and LOCKED are checked on the picture. Select 4 as the Divide By Value. Click Next. The next window is for Clock Buffers. One of the important aspects of the DCM is that all the DCM clock outputs get put through clock buffers. The default values should be fine. Click Finish.

    Another helpful tool for lazy typers: In the Project Navigator Sources for Project window, click on led_dcm. In the Processes for Source window, double click View HDL Instantiation Template. This will generate a Verilog instantiation template to instantiate the DCM.

12. In the Sources in Project window you will see a collapsible listing of the source file hierarchy for the project. Depending on which source you click on, the Processes for Source "<source file>" will change to correspond to that source file. This allows you to compile and debug at various levels in your design. Be sure that you have clicked on the source file you actually want before running one of the processes for it.

13. In the Sources in Project window, click on learn_ise. In the Processes for Source window, double click on Synthesize - XST. You should get errors. In the Console window, scroll up until you reach the first error from the top. It should be preceded by a little red WEB icon. Right click on the red WEB icon. Notice that you could Goto Answer Record. This links to the online Xilinx Answers Database and searches for any related answer records. Our problem is just a syntax error, so choose Goto Source. This will open the source file and indicates the offending line by a yellow triangle. Scroll up in the file to find the problem and fix it. Save led_lookup.v. Re-run synthesis.

14. When Synthesis completes successfully, double click on Implement Design in the Processes for Source window. You should get an error. The first error in the Console window will tell you what file to open. Make the required modification and save the file. Re-run Implement Design.

15. When Implementation completes successfully, double click on Generate Programming File in the Processes for Source window.

16. When the programming file has been generated, go to the Windows program folder where the Project Navigator shortcut is located. Open the Accessories folder. Open iMPACT. You will use iMPACT to download the .bit file to the board. Make sure your board is connected correctly, turned on, and with the User Input switches on.

17. From the Flow window, double click Boundary Scan. Right click on the screen and choose "Cable Auto Connect". Once the Transcript window shows that the connection is successfully done, right click on the screen and select "Add Xilinx Device"

18. Now you should get 2 devices in the boundary-scan chain shown on the screen.

19. Right click and select Assign New Configuration File for the XCCACE device; select Bypass. Then, do the same assigning new configuration file process for the XC2V2000 device but browse to your project directory and select learn_ise.bit. If you get a warning saying Startup Clock has been changed to JtagClk, click OK. If you want to repeat the initialization you just did (e.g. after making some modifications to your project), right click on the window and choose Initialize Chain.

20. Click on the xc2v2000 device so that it turns green. Go to the menu Operations > Program. DO NOT CHECK Verify! Click OK. Your .bit file will be downloaded to the board. When it is done you should get a message stating it is done and the DONE LED should light up on the board. If programming fails because the DONE LED could not be driven, repeat the programming attempt.

    You can then enter different combinations on the pushbuttons and hit Enter to get different combinations of flashing User LEDs. Even though there are mappings in the memory for an address affected by all 10 pushbuttons, why do only the 2 pushbuttons nearest the Enter key make any difference? Which of the 2 User Input switches is the RESET switch?

Look at Next

Module 7: ModelSim Simulation
Module 9: Using and Modelling OPB Interfaces