Version for EDK 10.1.03 and ISE 10.1.03 as of January 7, 2009

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 **do not** work reliably if paths contain 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. Links to the online documentation are embedded in the PDF version of this document; most of the referenced documentation is also installed locally as part of ISE.

Goals

- To gain a basic understanding of how to use ISE.
- To develop a simple core using ISE for use on the XUPV2P.
- 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 XUPV2P.

Requirements

Access to ISE 10.1.03

Preparation

The documentation for ISE 10.1.03 can be found here and can be also viewed directly here. The tools dealt with in this module fall into the *Design Implementation* category.

- Take a quick look through the ISE Help documentation in the Design Implementation tools list.
- Skim through the FPGA Design Flow Overview to better understand the various tools and their interactions.
- Skim through the ISE Quick Start Tutorial to get an idea of the additional capabilities of ISE.
- If you are using the XUPV2P board, read through the Using the LEDs and Switches section of the Xilinx University Program Virtex-II Pro Development System Hardware Reference Manual. You will be using the 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.

   If you are using the XUPV2P board:

   - The Device Family should be Virtex2P.
   - The Device should be XC2VP30.
   - The Package should be FF896.
   - The Speed Grade should be -7.

   The Synthesis Tool should be XST and the Simulator should be ModelSim-SE Verilog (unless you have ModelSim-XE installed instead). Click Next, Next, Next, and Finish.

   If you are unfamiliar with the Verilog HDL language, dont worry. Verilog syntax is a cross between C and VHDL. It is beneficial to have exposure to both Verilog and VHDL as both languages are used in industry. is a reasonable site to visit to get an idea of how to use Verilog.

With the XUPV2P board:

1. Create a new text document in your project directory called `example_memory.coe` and paste the following into it:

```
memory_initialization_radix=2;
memory_initialization_vector=
0101,
1010,
0110,
1001,
1100,
0011,
1110,
0001,
0000,
1111,
1000,
0100,
0010,
0111,
1011,
1101;
```

2. Select Project → New Source. Select IP (CoreGen & Architecture Wizard). Call it `example_memory`. Make sure the Add to Project checkbox is checked. Click Next.
3. The Select IP dialog box will pop up. Expand Memories & Storage Elements → RAMs & ROMs. Select Block Memory Generator v2.8. Click Next. Click Finish.
4. A CoreGen GUI will pop up. Leave `example_memory` as the Component Name. Select Single Port RAM as the memory type and let CoreGen Optimize for Area in the Primitive Selection panel. Click Next. Enter a Width of 4 and a Depth of 16 for Memory Size. This will create a ROM that has 16 4-bit words.

Leave all other options as their default values. Click Next. In the Memory Initialization panel, check the Load Init File checkbox. Click on Load File... and browse to and select the COE file that you created. Click Next. Click Finish. You should get the message "Successfully generated <example_memory>" in the Transcript panel at the bottom of the Project Navigator window.

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 over 50% of the on-chip block RAM, generating the ROM could take upward of 10 minutes.

5. Select Project → New Source. Select Verilog Module. Call it `example_verilog`. Make sure the Add to Project checkbox is checked. Click Next.

6. Create input ports `system_clock`, `sw_0`, `sw_1`, `sw_2`, and `sw_3` and output ports `led_0`, `led_1`, `led_2`, and `led_3`. Click Next. Click Finish

7. Select File → Open... and open the `example_memory.veo` file that was automatically generated for you by CoreGen. Copy the instantiation template from this file to your newly-created `example_verilog.v`. Connect the ports of the memory as follows: `addra` with {`sw_3,sw_2,sw_1,sw_0`}, `clka` with `system_clock`, `dina` with `4'h0`, `douta` with {`led_3,led_2,led_1,led_0`}, and `wea` with `1'b0`.

   Note: In Verilog, a pair of curly braces are used to concatenate signals. So sw_3,sw_2,sw_1,sw_0 is a 4-bit vector formed by concatenating the four 1-bit sw_<#> signals.

   Note: In Verilog a simple component instationation uses the following syntax:

```
<name\_of\_component> <instance name> (
    .<component\_port\_0>(<signal\_0\_in\_instantiating\_file>)
    .<component\_port\_1>(<signal\_1\_in\_instantiating\_file>)
    .<component\_port\_2>(<signal\_2\_in\_instantiating\_file>)
);
```

8. Save your example_verolog.v file. Select Project → New Source. Select Implementation Constraints File. Call it `example_verilog`. Make sure the Add to Project checkbox is checked. Click Next. Click Finish.

9. Select example_verilog.ucf from the Sources panel, then double-click on User Constraints → Edit Constraints (Text) in the Processes panel. Copy the appropriate lines from the User Constraint Files (UCF) section Xilinx University Program Virtex-II Pro Development System Hardware Reference Manual into your new UCF file.

10. Save and close all the open design files.

11. 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 panel 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.

    Another helpful tool for lazy typers: In the Sources panel of the Project Navigator window, select a Verilog or VHDL source file. In the Processes panel, double click View HDL Instantiation Template under View Design Summary Tab. This will generate an instantiation template to instantiate the module represented by that file.

12. In the Sources panel, click on the top-level module. In the Processes panel, double click on Synthesize — XST.

13. When synthesis completes successfully, double click on Implement Design in the Processes panel.

14. When Implementation completes successfully, double click on Generate Programming File in the Processes panel.

15. When the programming file has been generated, turn the XUPV2P on.

16. Double-click on Configure Target Device → Manage Configuration Project (iMPACT) in the Processes Panel. A iMPACT dialog box will appear. Select Configure devices using Boundary- Scan (JTAG), then click Finish.

17. The iMPACT tool will open up in Project Navigator. This tool is used to download bitstreams to the FPGA. There will be a message about connecting to cable and configuring and then an Assign New Configuration File dialog box will appear. Click Bypass, Click Bypass for the next dialog box as well, then Select example_verilog.bit for the third dialog box. A long, narrow dialog box should now appear, click OK. A "Device Programming Properties" dialog box will appear, click OK.

18. Right-click the xc2vp30 FPGA image in the main pane. Choose Program from the menu and click OK in the dialog box that appears. If you get a warning saying "Startup Clock has been changed to JtagClk", just click OK. Check the console at the bottom to see if the download was successful.

19. You can fiddle with the user switches to get different configurations on the LEDs. Can you explain the results of a given switch configuration?

Look at Next

Module m07: ModelSim Simulation