University of Toronto
ECE532 Digital Hardware
Module m06: Using ISE

Version for EDK 8.2.02i and ISE 8.2.03i as of January 5, 2007

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 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 8.2.03i

Preparation

The documentation for ISE 8.2i is available online and can be either viewed directly in the browser or downloaded for offline reading. The tools delt 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 Multimedia board, read through the section on User Input and Output in the MicroBlaze and Multimedia Development Board User Guide. You will be using the pushbuttons, User Input dip switches, and User LEDs in this lab.
- 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 Multimedia board:

   - The Device Family should be Virtex2.
   - The Device should be XC2V2000.
   - The Package should be FF896.
   - The Speed Grade should be -4.

   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.

3. If you are using the Multimedia board:

   (a) Unzip the `m06.zip` file in your project directory to get the files for this module.

   (b) Select Project → Add Copy of Source. Select the three Verilog files and the UCF file you just unzipped and add them to the project. Leave the default options selected in the dialog that pops up.

   (c) The `mem_init.coe` file was created using Microsoft Excel and the Memory Editor as described in Xilinx Answer Record #11744. Copy it into your `learn_ise` project directory. Select Project → New Source. Select IP (CoreGen & Architecture Wizard). Call it `led_lookup_ip`. Make sure the Add to Project checkbox is checked. Click Next.

   (d) The Select IP dialog box will pop up. Expand Memories & Storage Elements → RAMs & ROMs. Select Single Port Block Memory v6.2. Click Next. Click Finish.

   (e) A CoreGen GUI will pop up. Enter `led_lookup_ip` as the Component Name. Select Read Only as the Port Configuration. Enter a Width of 2 and a 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 panel. Leave all other options as their default values. Click Next. Click Next. In the Initial Contents panel, 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>" 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.

   (f) 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. (Naturally, you could come up with that yourself, but copying and pasting from the template saves typing.)

(g) 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 scaling. 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.

(h) The Select IP dialog box pops up. Expand FPGA Features and Design → Clocking → Virtex-II Pro, Virtex-II, Spartan-3. Choose Single DCM v8.2i. Click Next. Click Finish. The Xilinx Clocking Wizard pops up. Enter 27 MHz as the Input Clock Frequency, since 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 Next, then Finish.

If you are using the XUPV2P board:

(a) 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;
```

(b) Select Project → New Source. Select IP (CoreGen & Architecture Wizard). Call it `example_memory`. Make sure the Add to Project checkbox is checked. Click Next.

(c) The Select IP dialog box will pop up. Expand Memories & Storage Elements → RAMs & ROMs. Select Single Port Block Memory v6.2. Click Next. Click Finish.

(d) A CoreGen GUI will pop up. Leave `example_memory` as the Component Name. Select Read Only as the Port Configuration. Enter a Width of 4 and a Depth of 16 for Memory Size. This will create a ROM that has 16 4-bit words. Click Next. Let CoreGen Optimize for Area in the Primitive Selection panel. Leave all other options as their default values. Click Next. Click Next. In the Initial Contents panel, check the Load Init File checkbox. Click on Load File... and browse to and select the COE file that you created. Click Generate. 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.

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

(f) 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.

(g) 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: `addr` with {`sw_3,sw_2,sw_1,sw_0`}, `clk` with `system_clock`, `din` with `4'h0`, `dout` with {`led_3,led_2,led_1,led_0`}, and `we` with `1'b0`.

(h) Select Project → New Source. Select Implementation Constraints File. Call it `example_verilog`. Make sure the Add to Project checkbox is checked. Click Next. Select example_verilog as the source with which the UCF file should be associated. Click Next and Finish.

(i) 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.

(j) Save and close all the open design files.

4. 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. This will generate an instantiation template to instantiate the module represented by that file.

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

   If you are using the project files for the Multimedia board, you should get errors. In the Transcript panel, 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 one option is 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; the offending line is indicated by a yellow triangle. Scroll up in the file to find the problem and fix it. Save `led_lookup.v`. Re-run synthesis.

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

   If you are using the project files for the Multimedia board, you should get an error. The first error in the Transcript panel will tell you what file to open. Make the required modification and save the file. (*Hint: to open the UCF file in the ISE text editor, select* `learn_ise.ucf` *from the Sources panel and double-click on User Constraints → Edit Constraints (Text) in the Processes panel.*) Re-run Implement Design.

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

8. When the programming file has been generated, go to the Windows program folder where the Project Navigator shortcut is located. Make sure your board is connected correctly, turned on, and with the User Input switches on (*i.e.*, up). Open the Accessories folder and launch iMPACT. Select create a new project (.ipf) and click OK and then Finish.

- If you are using the Multimedia board, bypass the first Assign New Configuration dialog and select `learn_ise.bit` from your project directory in the second.
- If you are using the Multimedia board, bypass the first two Assign New Configuration dialogs and select `example_verilog.bit` from your project directory in the second.
- If you get a warning saying "Startup Clock has been changed to JtagClk", just click OK.

9. Click on the xc2v2000 device (Multimedia board) or xc2vp30 device (XUPV2P board) so that it turns green. Select 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.

    - If you are using the Multimedia board, you can then enter different combinations on the push-buttons 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 ten pushbuttons, why do only the two pushbuttons nearest the Enter key make any difference? Which of the two User Input switches is the RESET switch?
    - If you are using the XUPV2P board, 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
Module m09: Using and Modelling OPB Interfaces