Lab 11

Part C – 64 Bit Two-Level Decoder

Description

- In this module, you will synthesize, implement, upload, and test a hardware description for a two-level decoder, encapsulated within a serial transceiver.
  - You will use the same program for serial communication as in Part B.

Procedure

Project Setup

1. Download `decoder_generic.vhd` from Canvas.
2. From here you have two options:
   1. **Create a new project** from scratch. To do this, follow Steps 1-8 from 11B-handout, but add `decoder_generic.vhd` instead of the three files listed for the 2LPE. Then, you can proceed to Step 6 of this handout.
      - Note that you will not need to enable VHDL 2008 on the 2LPE file (since it does not exist), but you must still do so for the hardware container.
      - This is recommended if you had issues with Part B.
   2. **Modify the project you created for Part B** by following the Steps 3-5 of this handout.
      - This will be much faster since you will not have to re-customize the Clocking Wizard IP.
3. Disable the encoder-specific files:
   - In the `PROJECT MANAGER` pane, in the `Sources` subpane, expand the `apex` top-level component, and then its `hw : hw_container` subcomponent.
     - Expand the `encoder : priority_encoder_64` component entry. Then, click on that entry so it becomes highlighted blue.
     - Hold `CTRL`, and click the first two subcomponents of the `encoder` component.
       - At this point the `encoder` component and its children should be highlighted.
       - The first and third refer to the same file, so clicking one highlights both.
   - Right-click on one of the selected files and click `Disable File`. Alternatively, press the `ALT` and `-` (minus) keys at the same time.

4. Add `decoder_generic.vhd` to the project:
   - In the toolbar, click `File`, then `Add Sources`. Alternatively, press `ALT` and `A` simultaneously.
   - If not selected already, check `Add or create design sources`, then click `Next`.
   - The rest of this wizard is identical to the corresponding part of the project creation wizard.
   - Click `Add Files`, then navigate to find `decoder_generic.vhd`. Once you have added it, click `Finish.`
5. Comment out the code for Part B and uncomment the code for Part C:
   - Open the hardware container file by double-clicking on the `hw : hw_container` entry in the `Sources` subpane of the `PROJECT MANAGER` pane.
   - There are two labeled sections you must comment out: lines 194-196, and lines 213-220.
   - Instead of doing this manually, use your mouse to select all rows of one section, then press `CTRL` and `/` at the same time. This will automatically comment each selected line.
   - There is one labeled section you must uncomment: lines 226-235.
   - Use `CTRL + /` in the same way to uncomment these lines. This shortcut toggles line comments on the selected lines, so it can be used in either way.
   - Save the file (`CTRL + S`, or hit the save icon at the top of the editor).

Hardware Review and Upload

6. Follow along with the instructor to briefly review the hardware.
7. You do not need to take a screenshot of the Elaborated Design for the decoder.
8. Run synthesis, implementation, and bitstream generation. Then, open the Hardware Manager.
   - Take a screenshot of the resource utilization (LUTs and FFs) to add to your report.
9. Before proceeding, plug the FPGA development board into your computer using the provided Micro-USB to USB-A cable.
10. At the top, in the green banner (or under the `Open Hardware Manager` dropdown) click `Open Target`, then `Auto Connect`.
11. Once the device is connected, select `Program Device` (in either of the locations where `Open Target` was previously).
    - See 06B-handout for troubleshooting steps, or ask the instructor or TA.
12. Click `Program` to upload the bitstream to the FPGA.

Hardware Testing and Analysis

13. Use the same process as Part B to connect to the FPGA with SerialTool.
    - The relevant section from 11B-handout is Steps 18-25. Steps 16-17 are to install and run the program.
    - If you disconnected the FPGA since Part B, you may need to scan for serial ports again.
14. For each input in the table below:
    - Type it into the text box in SerialTool and hit `SEND`.
    - Record whether the expected output corresponds to your actual result.
    - Press the center button (`reset`) to reset the transceiver.
    - Repeat.
    - (Note: see Steps 26 and 27 from 11B-handout for more details).

Input	Expected Output
`0000 0000 0000 0030`	`0001 0000 0000 0000`
`0000 0000 0000 0000`	`0000 0000 0000 0001`
`0000 0000 0000 003F`	`8000 0000 0000 0000`
`0000 0000 0000 0038`	`0100 0000 0000 0000`
`0000 0000 0000 002B`	`0000 0800 0000 0000`

15. Take a screenshot of the SerialTool window after testing all the inputs above, to attach to your lab report.
16. Notice how although the inputs you tested on the decoder are the same as the outputs of the priority encoder from Part B, the outputs are different.
    - What does this tell you about the priority encoder? Can you figure out what the original input to the priority encoder was based on the output of the decoder?
    - Could you do this with a standard encoder (that only accepts one high bit in the input)?

Deliverables

- Include as part of your **informal report**:
    - A screenshot of the FPGA resource utilization (LUTs, FFs) (Step 8)
    - A screenshot of SerialTool containing your testing results (Step 15)
    - A brief comment on whether your experimental results match the expected results.
    - Answers to handout questions (Step 16)

Outcomes

- Practice working with VHDL.
- Practice using Vivado for hardware synthesis and implementation.
- Practice programming and testing a hardware description on an FPGA development board.
- Understand how to use serial communication to test hardware descriptions.
- Learn how to use a serial communication software to send data to and receive data from an FPGA development board.
- Learn how to add an Intellectual Property (IP) Core to a Vivado project.
- Learn how to use an onboard MMCM to generate a faster clock from the base 100 MHz clock on the FPGA development board.