ISTANBUL TECHNICAL UNIVERSITY

Digital System Design Applications

Experiment 2
MSI COMPONENTS

Preliminary

Students should know MSI elements needed for combinatorial circuits. MSI elements can be listed as following: Decoders, Encoders, Priority Encoders, Multiplexers. For the experiment, students should also know simplifying methods for Combinatorial Circuits like Karnaugh Maps.

Objectives

- Create a library which contains MSI Components using SSI library generated in the previous experiment.
- Implement the MSI elements on the FPGA board using LEDs and 7-Segment Display.

Requirements

Students are expected to be able to

- define hardwares with Verilog
- create project on Vivado
- synthesize, simulate, implement designs, generate bitstreams and configure FPGA
- write truth tables of all MSI elements
- simplify combinatorial circuits by karnaugh maps.

Experiment Report Checklist

Each student is going to prepare his/her own report. Reports should include:

1. **DECODER**
   - DECODER Verilog code, testbench code, and behavioral simulation wave screenshots.
   - RTL and Technology schematic of DECODER. How many LUTs are there in the Technology schematic? What are the logic statements of the LUTs? How do these logic statements implement the decoding operation.
   - **Greatest delay** of implemented DECODER design.
2. **PRIORITY ENCODER**
   - Truth table of Priority encoder and simplified logic statements of outputs OUT[0], OUT[1] and E using Karnaugh Map
   - Hand drawn schematic of priority encoder.
   - ENCODER Verilog code, testbench code, and behavioral simulation wave screenshots.
   - RTL and Technology schematic of ENCODER. Add the difference between the RTL and Technology schematic of ENCODER regarding to counts of the components (LUTs, gates).
   - Verilog code of the ENCODER using **always** and **case** structure. Add the comparison of the implementation results (timing,space,etc.) for both structural and behavioral designs.
3. **MULTIPLEXER**
   - MULTIPLEXER Verilog code, testbench code and behavioral simulation wave screenshots.
   - RTL and Technology schematic of MUX. Find the component except the LUTs in the technology schematic.
   - Verilog code of the MUX using **always** and **case** structure. Add the comparison of the implementation results (timing,space,etc.) for both structural and behavioral designs.
4. **DEMULTIPLEXER**
   - DEMULTIPLEXER Verilog code, testbench code and behavioral simulation wave screenshots.
   - RTL and Technology schematic of DEMUX. Explain the structure given in the Technology schematic.
5. Schematics of expanded top module.
6. Edited constraint file code (only used lines)
7. **Your comments on the results**
8. Definition of **Minimal Set**

- **Projects and reports are to be done INDIVIDUALLY. High amounts of points will be deducted from similar works.**
- **Reports must be written in a proper manner. Divide your text to sections and subsections if needed, label your figures and connect your sections with proper explanations of your works. Reports filled with imprecisely placed tables and figures, with no verbal explanations in workflow, will not fare well.**
- **Check homeworks section in Ninova for submission dates. There will be two different submissions for project archive folder and a PDF report file.**

Creating MSI Library

Create a new Vivado project with the settings below:

- Download and extract this Zip file from the link below
  https://reference.digilentinc.com/reference/software/vivado/board-files
- Copy the files that in this directory "vivado-board-master>new>boards-files"
- Paste the files to C:Xilinx>Vivado>2018.3>data>boards>board-files
- Restart Vivado
- You are now ready to use a Vivado project for the Digilent Nexys 4, Nexys 4 DDR, Zybo, Zedboard and Basys 3 FPGA Boards.
- You can select an FPGA board on which you want to study.

Add a new Verilog module named **MSI_Library** to your project. Clear all lines in your module.

Decoder

1. Write the truth table of a **4x16** decoder and add it your report.
2. Write down a module which is called **DECODER** into your **MSI_Library.v** file. This module is going to have 4-bit input **IN** and 16-bit output **OUT**. This module should behave like **DECODER**. Obtain this behaviour using only **case structures** and the truth table.
3. Create a new file named **top_module.v** and set this file as top module.
4. Add the inputs to your top module given below
   8-bit **sw**
   4-bit **btn**.
5. Add the output to your top module given below
   8-bit **led**
   7-bit **seg**
   1-bit **dp**
   4-bit **an**.
6. Add the constraint file (**"Nexys4DDR.xdc"**)to your project. You will need to edit this file according to your design.

7. Instantiate the **DECODER** module with the name **decoder1** in the **top module**.
8. Connect the least significant 4-bits of input **sw** to the input **IN** of the **decoder1**. Connect the outputs **dp**,**seg** and **led** to the output **OUT** of the **decoder1**, respectively.
most significant(dp) → last significant(led)
9. Connect least significant bit of the **an** to logic 0 and other digits to logic 1.
10. Create testbench for your design then make **behavioral simulation**.
11. Synthesize your design and add the following informations to your report:
    - Verilog code, testbench code and behavioral simulation wave screenshots.
    - RTL and Technology schematic.
    - How many LUTs are there in the Technology schematic? What is the logic statement of the LUTs? How does the logic statement implement the decoding operation?
12. Implement your design. Then, find the **greatest delay** of the decoder and add it to your report.
13. Edit the constraint file according to top module design then generate programming file (generate bitstream), and program your FPGA. Observe your design on the FPGA.

Priority Encoder

1. Write the truth table of the priority encoder, and add to your report.
2. Simplify the logic statement of the OUT[0], OUT[1] and E using Karnaugh Map and add to your report.
3. Draw the schematic of the reduced statements with hand, and add to your report.
4. Write down another module which is called **ENCODER** into your **MSI_Library.v** file. This module is going to have 4-bit input **IN**, 2-bit output **OUT** and 1-bit output **E**.
5. Write the structural code of the schematics obtained from reduced logic statements using SSI library.
6. Replace the **DECODER** in the top module with the **ENCODER**.
7. Connect the least significant 4-bits of input **sw** to the input **IN** of the ENCODER. Connect the least significant 2-bit of the output **led** to output **OUT** and the most significant bit of the output **led** to the output **E** of the ENCODER.
8. Create testbench for your design then make **behavioral simulation**.
9. Synthesize your design and add the following information to your report:
    - Verilog code, testbench code and behavioral simulation wave screenshots.
    - RTL and Technology schematic.
    - Difference between the RTL and Technology schematic, regarding to counts of the components(LUTs,gates).
10. Implement your design.

11. Write Verilog code of the ENCODER using **always** and **case** structure.
12. Implement behavioral design.
13. Add the comparison of the implementation results (timing,space,etc.) for both structural and behavioral designs.
14. Generate programming file (generate bitstream), and program your FPGA. Observe your design on the FPGA.

Multiplexer

1. Write down another module which is called **MUX** into your **MSI_Library.v** file. This module is going to have 4-bit input **D**, 2-bit input **S**, and 1-bit output **O**. This module should behave like an **MULTIPLEXER**. Obtain this behaviour writing the structural code (assign and logic operators) of the schematics given in Brown & Vranesic's book, page 99.
2. Replace the **ENCODER** in the top module with the **MUX**.
3. Connect the least significant 4-bits of input **sw** to the input **D** of the MUX. Connect the least significant 2-bit of the input **btn** to input **S** of the MUX. Connect least significant bit of the **led** to the ouput **O** of the MUX.
4. Create testbench for your design then make **behavioral simulation**.
5. Synthesize your design and add the following informations to your report:
   - Verilog code, testbench code and behavioral simulation wave screenshots.
   - RTL and Technology schematic.
   - Find the component except the LUTs in technology schematic.
6. Implement your design.
7. Write Verilog code of the MUX using **always** and **case** structure.
8. Implement behavioral design.
9. Add the comparison of the implementation results (timing,space,etc.) for both structural and behavioral designs.
10. Generate programming file (generate bitstream), and program your FPGA. Observe your design on the FPGA.

Demultiplexer

1. Write down another module which is called **DEMUX** into your **MSI_Library.v** file. This module is going to have 1-bit input **D**, 2-bit input **S** and 4-bit output **O**. This module should behave like an **DEMULTIPLEXER**. Obtain this behaviour using **NOT**, **AND** and **TRI** gates in the SSI library.
2. Connect the least significant bit of input **sw** to the input **D** of the DEMUX. Connect the least significant 2-bit of the input **btn** to input **S** of the DEMUX. Connect least significant 4-bit of the **led** to the ouput **O** of the DEMUX.

3. Create testbench for your design then make **behavioral simulation**.
4. Synthesize your design and add the following information to your report:
   - Verilog code, testbench code and behavioral simulation wave screenshots.
   - RTL and Technology schematic.
   - Explain the structure given in the Technology schematic.
5. Implement your design.
6. Generate programming file (generate bitstream), and program your FPGA. Observe your design on the FPGA.

References:

1. Nexys4DDR Reference Manual
2. Artix-7 Libraries Guide for HDL Designs
3. Constraints Guide
4. Brown&Vrasenic, "Fundamentals of Digital Logic with Verilog Design", McGraw-Hill, p.297-319, 149-201