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 ISE
- synthesize, simulate, implement designs, generate bitstreams and configure FPGA
- write truth tables of all MSI elements
- simplify combinatorial circuits by karnaugh maps.

References:

1. Nexys2 Reference Manual
2. Spartan-3E 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

Creating MSI Library

Create a new ISE project with the settings below:

- Family: **Spartan 3E**
- Device: **XC3S500E**
- Package: **FG320**
- Speed: **-4**
- Synthesis Tool: **XST (VHDL\Verilog)**
- Simulator: **ISim (VHDL\Verilog)**
- Preferred Language: **Verilog**

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

Decoder (75 Minutes)

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 (**"Nexys2_500General.ucf"**)to your project.
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. Synthesize your design and add the following informations to your report:
    - 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?

11. Implement your design with the processes, translate, map and Place and Route (PAR), respectively. Then, find the **greatest delay** of the decoder, after the PAR process, and add it to your report.
12. Run **FPGA Editor**, and add the layout view of the design to your report.
13. Add your design a constraint satisfying that the greatest pad to pad delay of the circuit will be **10ns**.
14. Find the **greatest delay** of the decoder, after the PAR process. Add the comparison of the results after adding the time constraint and results for without constraint to your report.
15. Run **FPGA Editor**, and add the layout view of the design to your report. Furthermore, explain the difference of the two layout(with and without constraint) in your report.
16. Generate programming file, and program your FPGA. Observe your design on the FPGA.

Priority Encoder (60 Minutes)

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. Synthesize your design and add the following informations to your report:
   - RTL and Technology schematic.
   - Difference between the RTL and Technology schematic, regarding to counts of the components(LUTs,gates).
9. Implement your design.
10. Run **FPGA Editor**, and add the layout view of the design to your report.
11. Generate programming file, and program your FPGA. Observe your design on the FPGA.

Multiplexer (45 Minutes)

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. Synthesize your design and add the following informations to your report:
   - RTL and Technology schematic.
   - Find the component except the LUTs in technology schematic.
5. Implement your design.
6. Run **FPGA Editor**, and add the layout view of the design to your report. Find the component providing the MUX behaviour in the slice, and specify in your report.
7. Generate programming file, and program your FPGA. Observe your design on the FPGA.

Demultiplexer (45 Minutes)

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. Synthesize your design and add the following informations to your report:
   - RTL and Technology schematic.
   - Explain the structure given in the Technology schematic.
4. Implement your design with the processes, translate, map and Place and Route (PAR), respectively.
5. Run **FPGA Editor**, and add the layout view of the design to your report. Analyze the internal view of the output led[0]. What is the difference between the outputs of the previous desings(DECODER,ENCODER,MUX) and the current design(DEMUX)?
6. Generate programming file, and program your FPGA. Observe your design on the FPGA.

Experiment Report

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

- Screenshots and results asked above
- Your comments on results
- Write verilog code of the ENCODER and MUX using **always** and **case** structure. Add the comparison of the results (timing,space,etc.) for the designs given in the experiment and results for the behavioural designs(always and case).
- Definition of **Minimal Set**.

Reports are to be submitted before next experiment. Submission includes reports and project files.