ISTANBUL TECHNICAL UNIVERSITY

Digital System Design Applications

Experiment III
VARIOUS IMPLEMENTATIONS OF BOOLE FUNCTIONS

In this experiment, different implementation methods of combinational circuits will be covered. Students will realize a Boolean function given in truth table format, then simplify it using Boolean reduction methods. Full gate based, decoder based and multiplexer based realizations will be done in HDL code. Designs will be verified using testbenches and real-time FPGA executions. Routing, timing and placement constraints for a target device will also be introduced.

Objectives

- Learning different realization methods for combinational circuits
- Examining how Xilinx tools hanlde the synthesis of combinational circuits
- Gaining basic knowledge about XDC based routing, timing and placement constraints
- Using testbenches to perform proper simulations

Requirements

Students are expected to know;

- The basic concept of combinational circuits and Boolean algebra,
- Boolean expression simplification methods.
- How a decoder and a multiplexer operates.

Experiment Report Checklist

1. **Realization with SSI Library**
   - Step-by-step Karnaugh Map simplification of the truth table. Boolean expressions of all outputs in terms of inputs
   - Reformatted output expressions and final gate level circuit schematic (Draw it by hand or with a computer program).
   - Behavioral simulation console output
   - Include the following items related to **the design with NO TIMING AND LOC CONSTRAINTS:**
     - RTL schematic and Post-Synthesis technology schematic
     - Pad to pad path delays of the implemented design
     - Resource usage table of the implemented design
   - Include the following items related to **the TIMING CONSTRAINED design WITHOUT LOC constraints:**
     - Pad to pad path delays of the implemented design
     - Resource usage table of the implemented design
   - Include the following items related to **the LOC CONSTRAINED design WITHOUT timing constraints:**
     - Pad to pad path delays of the implemented design
     - Zoomed device layout of your design (only include the places that logic elements are placed - they will be highlighted on the device layout)
   - Include the following items related to **the TIMING AND LOC CONSTRAINED desgin:**
     - Pad to pad path delays of the implemented design
     - Post-implementation timing simulation waveform output
   - Comparison of all four designs above, in terms of path delays and placement differences
   - Give answers to all questions that are asked within experiment steps, in their related parts within the report text.
2. **Realization with Decoder**
   - 4-to-16 decoder representation of all four-variable minterms
   - Sum of product forms of the outputs f0,f1,f2 and f3
   - Behavioral simulation waveform output.
   - RTL schematic and Post-Synthesis technology schematic
   - Resource usage table of the implemented design
   - Pad to pad path delays of the implemented design
   - Zoomed device layout of your design (only include the places that logic elements are placed - they will be highlighted on the device layout)
   - Comparison of resource usage of Decoder based implementation and SSI based implementation
   - Pad to pad path delays of the implemented design, with timing constraint

- Give answers to all questions that are asked within experiment steps, in their related parts within the report text.

3. **Realization with MUX**
   - Circuit schematic of MUX based realization of the truth table (draw by hand or use a computer tool)
   - Behavioral simulation waveform output
   - RTL schematic and Post-Synthesis technology schematic
   - Resource usage table of the implemented design
   - Pad to pad path delays of the implemented design
   - Zoomed device layout of your design (only include the places that logic elements are placed - they will be highlighted on the device layout)
   - Comparison of resource usage of MUX based, decoder based and SSI based implementations
   - Pad to pad path delays of the implemented design, with timing constraint
   - Give answers to all questions that are asked within experiment steps, in their related parts within the report text.

4. **Miscellaneous Questions/Works**
   - What are the differences between **Behavioral Simulation, Post-Synthesis Functional Simulation, Post-Implementation Functional Simulation and Post-Implementation Timing Simulation**.
   - Consider that the truth table inputs *{a,b}* and *{c,d}* represent two 2-bit numbers. According to this hint, what arithmetic operation does the truth table that you used in this experiment perform?
   - Consider the three realizations you've done in this experiment. Evaluate them in terms of following traits:
     - Design difficulty
     - Coding difficulty
     - LUT usage
     - Path delays

- **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 arhive folder and a PDF report file.**

Implementation of Combinational Circuits with SSI and MSI Libraries

Starting Up

- Create a new Vivado project (see Experiment Sheet 1). Add the SSI and MSI library source files you've written in experiments 1 and 2 (SSI_Library.v and MSI_Library.v) into your project. In the "Add or Create Design Sources" window, make sure that "Copy sources into project" option is selected. Then, create a new Verilog file named **three_different_methods.v** in your project. Clear all of the automatically generated template code in this file to make it blank. This file will hold three different modules, each realizing a given Boolean function in a specific way. **The Boolean function in question can be found at Ninova, under Class Files, at "Experiment Files\Experiment_3\truth_table.png"**. Note that the function has four outputs (f_3, f_2, f_1, f_0) and four inputs (*a, b, c,* and *d*).

Realization with SSI Library

1. At the first step, each four output are needed to be derived as Boolean expressions, in terms of given inputs. Achieve this by applying Karnaugh Maps simplification on the given function.

2. Try to simplify your results by coverting to **multi-level** logic (Hint: try to use XOR operations if needed). Note that your library consist of two-input gates, so get your expressions to an according form. Add your reformatted results and your final hand-drawn gate-level circuit schematic to your report.

3. Create a module named **with_SSI.v** in your **three_different_methods.v** file.

4. The module should have four 1-bit inputs named *a, b, c, d* and four 1-bit outputs named *f3, f2, f1, f0.*, in order to be consistent with the given truth table.

5. Set **"with_SSI"** as the top module. Implement your final two-input gate based circuit in this module, by using the elements of your SSI library.

6. Download the testbench file provided in Ninova Class Files, under "Experiment Files\Experiment_3\experiment3_tb.v". Add it to your project as a simulation source.

7. Perform a behavioral simulation using this testbench. You should see the generated outputs that are relevant to your inputs at the bottom of the screen (the simulator console). Show that these outputs are consistent with the given truth table.

8. Return to your HDL design. **Synthesize** your circuit, obtain **RTL** and post-synthesis **Technology** schematics and add them to the your report.

9. Create a constraints file in your project, using Nexys 4 DDR master constraint file as a basis. In this file, connect your a, b, c, d inputs to switches SW3, SW2, SW1, SW0 respectively. Also, connect your outputs f3, f2, f1, f0 to LED outputs LED3, LED2, LED1, LED0 respectively.

10. Implement your design. Once it's done, obtain **Pad to Pad** delays related to your circuit, by running "Report Timing" command (remember to mark "Report Datasheet" checkbox), under Reports menu. Add it to your report.

11. Add a **set_max_delay** constraint to the .xdc file to try limiting the maximum pad to pad delay of your design to be 9 nanoseconds. Re-implement the desing and run "Report Timing" command again. Did the constrained implementation achieve it's goal? What differences are seen between constrained and non-constrained device overviews? Obtain the resource usage (number of used LUTs,I/O's etc.) of your design.

12. Comment out the "set_max_delay" constraint from your .xdc file. Try to spread your design across logic slices named "SLICE_X12Y67","SLICE_X12Y66","SLICE_X13Y67" and "SLICE_X14Y64"; using LOC constraints. Obtain your LUT cell names from your netlist, to be used with "get_cells" attribute of LOC constraints. Implement your constrainted design and observe the device layout (Click "Open Implemented Design" command on the left side, if the "Device" tab does not appear in your working space.) to see where your design is physically placed. Add the zoomed device layout (slice names must be clearly seen) to your report. Get Pad to Pad delays of this implementation.

13. Keep the LOC constraints in your .xdc file, and uncomment the "set_max_delay" constraint to enable it again. Implement the design and observe the Pad to Pad delays of this implementation. Observe if the tool managed to met all of the design constraints you specified. Consider all four implementations you've done so far (no constraints, time constraint only, location constraint only, and both time and location constrained). and their related Pad to Pad path delays. Which one gave the best results in terms of combinational delay? Does placement and routing affect combinational delay? Based on your results, write down your deductions in your report.

14. Perform a **Post-Implementation Timing Simulation** on your latest implemented design, by right-clicking to "Run simulation" command under Flow Navigator. Use the same testbench again (experiment3_tb.v). What are the distinctive differences between post-Implementation timing simulation and behavioral simulation?

15. Generate **.BIT file** and program your FPGA. Show that your circuit works as expected, by trying all possible input combinations with switches and observing the LEDs.

Synthesizing with Decoder

1. Open your Verilog source file, **three_different_methods.v**, and define a new module named **with_decoder** (do not delete "with_SSI" module, just create a second one). It's input and output names and properties are going to be same with "with_SSI" module. This module will hold the DECODER based realization of the same truth table that's used in previous step.

2. Draw a 4-to-16 decoder schematic on paper and show how all possible **four-variable minterms** can be represented on this. Include your drawing to your report.

3. Express the logical functions f0,f1,f2 and f3 in **sum of products** form, using the same truth table. Add it to your report.

4. Based on your DECODER minterm setup and f0,f1,f2 and f3 expressions, code your circuit using your own MSI_library element, DECODER; and SSI_library element, OR. The design should only include one DECODER, OR gates and wires.

5. From the source tab, set "with_decoder" module as your top module. Then, change the instance named uut (as in Unit Under Test)" in your current testbench file **experiment3_tb.v** with **with_decoder** (should be with_SSI from before) and perform a behav-

ioral simulation. Verify that your circuit gives the correct results for all possible inputs. Add the waveform to your report.

6. Comment out all timing and location constraints in your .xdc file (only keep I/O constraints). Implement the design, add RTL schematic, technology schematic and pad to pad timing information to your report.

7. Go into the "Device" overview of your FPGA. Zoom in to the design layout and observe the placed design. Are there any differences in used primitive types?

8. Obtain the resource usage (number of used LUTs, I/Os etc.) of your design. Compare the resource usage of DECODER based design and previous SSI based design. Comment on the resource usage differences.

9. Implement the circuit again using 6ns max delay constraint. Add pad to pad timing information to your report.

10. Generate BIT file and program your FPGA. Show that your circuit is working correctly.

Synthesizing with MUX

1. Open your Verilog source file, **three_different_methods.v**, and define a new module named **with_MUX** (do not delete previously defined modules from this file, just create a third one). It's input-output names and properties are going to be same with "with_SSI" and "with_decoder" modules. This module will hold the MUX based realization of the same truth table that's used in previous steps.

2. Set "with_MUX" as the top module.

3. You will use four multiplexers for generating f3, f2, f1, f0 outputs. For this purpose, connect **a, c** inputs to select bits of multiplexers, where "a" is the most significant bit and "c" is the least.

4. Using the setup explained in the previous step, draw the MUX based circuit schematic that realizes the given truth table, on paper. Add it to your report.

5. Express this hand-drawn schematic as structural HDL code, by using your SSI and MSI libraries.

6. Open your testbench file, **experiment3_tb.v**, and change the instance named "uut" with **with_MUX** (should be with_decoder from before) and perform a behavioral simulation. Verify that your circuit gives the correct results for all possible inputs. Add the waveform to your report.

7. Comment out all timing and location constraints in your .xdc file (only keep I/O constraints). Implement the design, add RTL schematic, technology schematic and pad to pad timing information to your report.

8. Go into the "Device" overview of your FPGA. Zoom in to the design layout and observe the placed design. Are there any differences in used primitive types?

9. Obtain the resource usage (number of used LUTs, I/Os etc.) of your design. Compare the resource usage of MUX based design with DECODER based design and SSI based design. Comment on the resource usage differences.

10. Implement the circuit again using 6ns max delay constraint. Add pad to pad timing information to your report.
11. Generate BIT file and program your FPGA. Show that your circuit is working correctly.

References:

1. Xilinx ISim User Guide (UG660)
2. Xilinx Vivado Design Suite Tutorial: Using Constraints (UG945)
3. Vivado Design Suite Properties Reference Guide (UG912)