Queen's University
Department of Electrical and Computer Engineering
ELEC 271 Digital Systems
Fall 2018

Independent Learning Minilab on Logic Optimization with Karnaugh Maps and Flip-Flop Behavior

Objectives

This independent exercise for *ELEC 271 Digital Systems* provides the opportunity for students to:

- optimize logic functions with Karnaugh maps for sum-of-products/product-of-sums forms, and
- explore the behavior of a flip-flop through waveform simulation.

The Thanksgiving holiday on October 8, 2018, affects two of the six laboratory sections for Lab 3. To mitigate the loss of in-lab activity from the holiday, this exercise provides affected students with independent activity related to logic optimization and flip-flops. This activity is intended as comparable preparation for the in-class quiz as other students who will complete the full Lab 3 exercise.

Students in the other sections that are not affected by the holiday may also benefit from pursuing this independent exercise as additional practice, either before or after completing their scheduled session for Lab 3.

Preparation

- In a file entitled `lab3.vhd`, prepare the following template for later modification.

```
library ieee;
use ieee.std_logic_1164.all;

entity lab3 is
  port (?, ?, ?  : in  std_logic;
        fsp, fps : out std_logic);
end entity;

architecture logic of lab3 is

begin

-- define the simplified sum-of-products function
fsp <=  ???????;

-- define the simplified product-of-sums function
fps <=  ???????;

end architecture;
```

- In a file entitled `tb_lab3.vhd`, prepare contents that are identical to the testbench template that was provided in Lab 2, except for the modifications that are described below.
    - Replace every occurrence of `lab2` with `lab3`.
    - Replace every occurrence of `tb_lab2` with `tb_lab3`.
    - For the `component` declaration, replace the single `f` with the pair `fsp, fps`.
    - For the `signal` definition, replace the single `f` with the pair `fsp, fps`.
    - For the `assert` statement, replace `f` with `fsp` and also change the text of the error message to `"SOP output did not match"` to make it more informative.
    - Copy all of the lines of the updated `assert` statement and paste the copy immediately below the original. In the copy, replace `fsp` with `fps` and also change the text of the error message to `"POS output did not match"` to make it more informative.

- In a file entitled `flipflop.vhd`, prepare a VHDL description for a `flipflop` entity definition and corresponding architecture body as described below.
    - Introduce four input ports for the entity: `clk`, `reset_n`, `load_en`, and `d`.
    - Introduce one output port for the entity: `q`.
    - For the architecture, use lecture material to guide you in preparing a VHDL process that has correct `if` statement syntax (including appropriate `if` nesting) to handle the `clk`, `reset_n`, `load_en`, and `d` inputs for generating a correct `q` output.
    - Ensure that the sensitivity list for the process has the appropriate subset of input signals (not all of the inputs) in accordance with conventions for use of this VHDL feature.

- Prepare a template file `signal_values.txt` identical to the one described in Lab 2. Review the discussion about this template file as a reminder of how it will be used for simulation purposes.

- Review the relevant course material on logic optimization with Karnaugh maps.
- Review the relevant course material on D-type flip-flops and their descriptions in VHDL.

Part 1: Logic Optimization in Sum-of-Products and Products-of-Sums Forms

- Consider the logic function $f(x, y, z) = \Sigma(m_1, m_4, m_5, m_6)$ in sum-of-products form.
- *On your worksheet*, record the preceding function in the appropriate space. Use a Karnaugh map to obtain the simplified sum-of-products form of the function.
- *On your worksheet*, write the corresponding the product-of-sums form, i.e., $f(x, y, z) = \Pi(...)$. Use another Karnaugh map to obtain the simplified product-of-sums form of the function.
- On your network storage, create a folder `Z:\My Files\ELEC271\LAB3`. (If you have already completed or will be completing the full Lab 3 in a scheduled session, then use a different folder such as `Z:\My Files\ELEC271\MINILAB` to keep the projects separate. Project labels and project/VHDL files can have the same names as long as they are located in separate folders.)
- Place files `lab3.vhd`, `tb_lab3.vhd`, and `signal_values.txt` in the above folder.
- Start Altera Quartus II, and create a new project labelled `lab3` in the folder you created above.
- At the appropriate screen of the New Project Wizard, include `lab3.vhd` into the project.
- As done in Lab 1 and Lab 2, use the New Project Wizard to complete the setup for the correct Cyclone III chip on the Altera DE0 board. *Also select ModelSim-Altera and VHDL for simulation*.
- Even though the hardware will not be used for this exercise, use *Assignments → Device…* to change settings for *unused pins* and *voltage* as a way of remembering to do so for other labs.
- In the Quartus II main menubar, select *File → Open...* and select your `lab3.vhd` file for editing.
- Complete the entity section for the given input variable names. Then, in the architecture section, type your optimized sum-of-products and product-of-sums expressions for outputs `fsp` and `fps`. *Use parentheses in your logic expressions to ensure your desired order of operations.*
- From the main menubar, select *File → Save* to save the modified file.
- Select *Processing → Start Compilation* from the main menubar in order to synthesize the circuit.
- In Quartus II, select *Tools → Netlist Viewers → **RTL** Viewer* to examine the initial schematic.
- Then, select *Tools → Netlist Viewers → Technology Map Viewer (Post-**Mapping**)* to examine the result of processing by the software for comparison with your optimized expressions. Just click on the boxes labelled LOGIC_CELL_COMB. The inputs and outputs are pins, and they will have buffers as symbols. Our interest is the logic function implementation, not the pins.
- Finally, select *Tools → Chip Planner* to view the placement of the logic functions by Quartus II. Use the zoom features to closely inspect the clusters in which the functions have been placed, and also double-click (with the arrow cursor) within the clusters to examine the logic elements. How many lookup tables are used, and is that the result what would be expected for this chip?

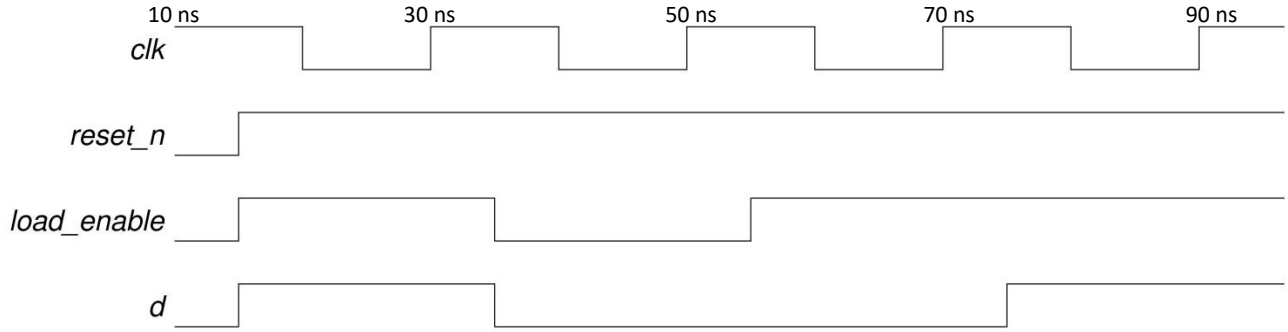
Part 2: Testbench Simulation using the ModelSim-Altera Software Tool

- In the Quartus II menubar, select *File → Open...* to edit your `tb_lab3.vhd` file. The Quartus text editor is used here for convenience; this file will *not* be synthesized by Quartus for an FPGA.
- In the testbench VHDL file, replace all '?' characters with the specific names of the three inputs. Note the following order for the `in_vector` assignments: bit 2 should be the *first* input variable, bit 1 should be the *second* variable, and bit 0 should be the *third* variable.
- Select *File → Save* to save the changes to `tb_lab3.vhd`.
- Select *File → Open...* to edit your `signal_values.txt` template with '?' for output values.
- Using the minterm/maxterm information on your worksheet, change '?' characters to either 0 or 1.
- Select *File → Save* to save the changes to `signal_values.txt`.
- Using Microsoft Windows Explorer, navigate to your project folder.

- Right-mouse-click on the icon for the modified `signal_values.txt` file and select *Copy*.
- Find the `simulation` folder icon in the Windows Explorer view and double-click on it.
- You should now see the `modelsim` folder icon. Double-click on it to enter this folder.
- With the Windows Explorer view now showing the `simulation\modelsim` folder contents, right-mouse-click on a clear area of the `modelsim` folder view, then select *Paste*. Ensure that a copy of your modified `signal_values.txt` file has indeed been pasted in the `modelsim` folder.
- In the Quartus II menubar, select *Tools → Run Simulation Tool → **RTL** Simulation*.
- *Refer to the instructions in Part 2 of Lab 2 for the commands to type in the Transcript window*.
- Also refer to the instructions in Part 2 of Lab 2 for how to manipulate the waveform window to view the simulation results. The waveforms for the two optimized output signals should match each other and they should reflect the expected output values for the original function specification.
- Furthermore, the text displayed in the *Transcript* window should indicate that there are no errors.
- If the results are incorrect, review your logic optimization, check the VHDL code, and try again.

Part 3: Inferred D Flip-Flop Specification in VHDL with Waveform Simulation

- Consider the diagram below.



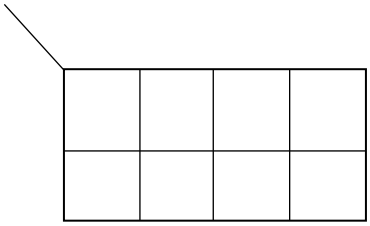
- *On your worksheet*, draw the waveforms given to you (using lightly-drawn vertical lines to on each rising edge provides helpful positioning information for a correct diagram).
- *On your worksheet*, draw your prediction for how the *q* output will behave for the given waveforms.
- Place your prepared file `flipflop.vhd` in the folder you created for this exercise.
- In Quartus II, create a new project labelled `flipflop` in the same folder that you created for this exercise. It is acceptable to have two projects in the same folder, provided that appropriate care is taken to prevent conflicts/confusion with any shared files. Thus, when a prompt appears asking "Do you want to select a *different* directory?" it is safe to answer "no" in this case.
- At the appropriate screen of the New Project Wizard, include `flipflop.vhd` into the project.
- Perform all of the other usual project-preparation steps (even though no hardware will be used).
- Select *Processing → Start Compilation* from the main menubar in order to synthesize the circuit.
- Select *Tools → Chip Planner* to view how Quartus chose to place of the inferred D flip-flop and the additional logic for the load-enable capability (i.e., the multiplexer for the D input) within the chip.
- Select *File → New...* and select University Program VWF file from the list of available file types.
- When the input waveform editor window appears, first use *Edit → Set End Time...* to reduce the duration of simulation time to 90 ns. Also, use *Edit → Grid Size...* to set to grid size to 5 ns.
- Then, add the signals in the order listed on your worksheet. Consult the procedure in the description for Lab 1 for using the Node Finder to include the signals in the correct order.
- To create a clock waveform, consult the Lab 1 description (the counter circuit simulation). For the

clock period, use 20 ns. With the End Time setting above, this will give a total of four clock cycles to match the template diagram that is provided on the worksheet for this exercise.

- Draw the remaining input waveforms so that they appear the same as the given waveforms with signal transitions within the appropriate high/low half of the appropriate clock cycle.
- Save the waveform file with a meaningful name (e.g., `test_flipflop.vwf`).
- Perform a ***functional*** simulation.
- When the output window appears with the results of the simulation, compare your prediction for the behavior of the *q* signal with the simulated behavior. Make certain that your input waveforms are correct, and if your prediction is not quite the same as the simulated behavior, make certain that you understand *why* so that you will be able to make a correct prediction on a quiz/exam.

Minilab Worksheet

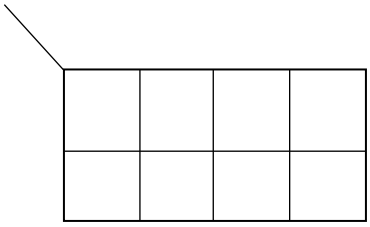
Given canonical sum-of-products function: *f*(, ,) =



Optimized sum-of-products expression:

_____

Equivalent canonical product-of-sums function: *f*(, ,) =



Optimized product-of-sums expression:

_____

Waveform diagrams for prediction of simulation output for inferred D flip-flop: