Homework 2: Synthesis

58 marks

To be done individually and submitted by 24 Sep, 11:59pm

In synthesis, we generate a logic circuit from a behavioral description. The behavioral description is in the form of Verilog code. This behavioral code is often called “RTL” (for Register Transfer Level), but this should strictly be used only if the behavioral code is for sequential logic.

The logic circuit is generated using logic gates from a specified standard cell library. This logic circuit, also called a gate-level netlist, is also in Verilog format.

The inputs for synthesis are:

(i) The design, in the form of behavioral Verilog code and timing requirements (i.e. “constraints”)

(ii) The standard cell library in the form of Liberty view and, possibly, LEF view

(A) sel_sense (24 marks)

“sel_sense” is a simple combinational circuit with two inputs and one output. A plain-English behavioral description of the logic is as follows:

1. The circuit has one output, B, and two inputs, A and S
2. The output B is equal to the input A if S = 0 or is equal to the inverse of input A if S = 1.

Submissions:

(i) Write a simple behavioral Verilog code for sel_sense. You can use the conditional or ternary operator, “?”.

(ii) Draw a schematic of a circuit that implements sel_sense. Your schematic must involve gates like NAND2, NOR2, MUX, INVERTER, XOR, NAND, etc. Choose whichever gates you like to implement the logic. Do not make a truth-table; draw a schematic based only on your understanding of the logic. Use standard symbols for the gates in your schematic. (2 marks)

(iii) Now write a truth table for the circuit. Based on your truth-table, if you would like to make an alternative schematic for the circuit, please do so, and you can submit that schematic as well. (2 marks)

(iv) Synthesize sel_sense with the SkyWater HS and MS libraries. Use the ss, 150C, 1.6V corner. Specify an output load cap equal to the input pin cap of an X2 buffer, and a max delay from input to output of 50 ps before the optimization step. Submit the resulting netlists, and your synthesis script. (6 marks)

(v) Put a “don’t use” on all XOR and XNOR gates, resynthesize, and submit the resulting netlists for both HS and MS libraries. (6 marks)

(vi) For the HS and MS netlists with no “don’t use” directives, plot A->B delay as a function of load capacitance at B. Let the capacitance vary from 0.005 to 0.1 pF. For each cap value, report timing before and after doing

an incremental optimization (-incr option). Create plots for S = 0 and for S = 1. Let transition time at A be 0.04 ns. Note that for timing to be reported from A to B, you will need to specify a max delay constraint for that path. (8 marks)

(B) Buffers in Standard Cell Libraries (6 marks)

From the "slow corner" dotlibs of the standard cell libraries in Cadence RAK, and Nangate_15nm_OCL, fill out the table below for the **buffer of drive strength 2:** [The number of rows in the table is only for the purpose of indication]

Cell Name	Name of Input Pin	Capacitance of Input Pin (pF)

Hints and notes:

(i) Using the "grep" command with the "-A" option will be one easy way to get the information for the above table

(ii) Libraries often have many types of buffers. For example, there may be special "clock buffers" for use in distributing clock, or tri-sate buffers. In the table above, provide information only for "ordinary" drive-strength-two buffer. You will need to guess which one that is from the cell name, or, if you are fortunate, there may be a cell description as a comment in the dotlib.

(iii) Sometimes different capacitance values are listed in the dotlib for rise and fall transitions; you need to report the canonical capacitance, i.e., the "average" capacitance that is not dependent on type of transition. For example:

```
pin (I) {
        direction          : input;
        related_power_pin  : "VDD";
        related_ground_pin : "VSS";
        capacitance        : 0.839939;   <-- Put this value in the
        fall_capacitance   : 0.841850;   <-- Don't put this value in
                                             the table
```

(C) mux_comparator (28 marks)

Submissions:

(i) Write behavioral Verilog code for a combinational logic function called "mux_comparator" that compares two three-bit numbers x and y, and outputs whichever one of the two numbers is numerically less. Output should be called z. Verify it simulates as expected (4 marks)

(ii) Specify the load capacitance at all outputs to be equal to the input pin capacitance of a drive-strength=2 buffer from the standard cell library, using the table you generated in Section B. Specify the transition time at all inputs to be 100 ps. Set a "max delay" constraint of 250 ps from all

inputs to all outputs. Put these specifications in an SDC file called “mux_comparator_<library_name>.sdc”.

(iii) Synthesize this function with the libraries Cadence RAK and Nangate_15nm_OCL. Submit the three netlists that result. (8 marks)

(iv) In each case, identify the critical path from input to output, listing the max_delay from input to output. List the number of standard cells in the critical path. (6 marks)

(v) In each case, identify the min_delay path and tabulate the min delay. List the number of standard cells in the min_delay path. (6 marks)

(vi) Change the three-bit numbers x and y to eight-bit numbers. Resynthesize, and report the number of standard cells in the critical (max_delay) path for just the Cadence RAK PDK. (4 marks)