UNIVERSITY OF TEHRAN

Electrical and Computer Engineering Department

Digital Logic Design, ECE 367 / Digital Systems I, ECE 894

Spring 1399-1400

Homework 5

Logic Minimization, Logic Synthesis, Basic RTL Combinational Parts

Name:	**Date:**
Username:	

1. Function *f(a,b,c,d)* is given below. You are to synthesize this function using Yosys and compare the results with your own hand-synthesis.
    a. Write a Verilog description of this function using an **assign** statement. Use the function's SSOP (no minimization). Using Yosys and using the library provided to you for the synthesis target (note that this library only contains NOT, NOR, and NAND), synthesize your Verilog description to a netlist of the library components.
    b. Take the netlist output of Yosys and show its corresponding circuit diagram. Using Boolean algebra, verify that this netlist or the circuit correctly implements the SSOP of function *f(a,b,c,d)*.
    c. Starting from the SSOP expression of function *f* (as shown below) and using K-Maps, generate the minimal realization of this function using NAND gates.
    d. Compare the circuit you extracted from Yosys synthesis (Part b) with your hand-synthesis of this function (Part c) in terms of number of gates and circuit delay.

$$f(a,b,c,d) = \sum_m (2, 3, 4, 6, 7, 8, 9, 12, 13)$$

2. In this problem, you are to synthesize an adder using Yosys.
    a. Write a Verilog description of a full-adder using the following assignment:

```
assign {co,sum} = a+b+ci;
```

    Using Yosys and using the library provided to you for the synthesis target, synthesize your Verilog description to a netlist of the library components.
    b. Take the netlist output of Yosys and show its corresponding circuit diagram. Show that the output circuit is indeed functions as a full-adder.
    c. Write a description for a 4-bit adder in Verilog. Synthesize this description using Yosys. Write the number of cells and gates used as the result of synthesis. Compare these with the same numbers reposted for the full-adder of Part b.
    d. Based on Part c, you should be able to know the structure used for the implementation of the 4-bit adder. Show a block diagram of the synthesized circuit.

3. You can do this problem using Karnaugh maps. Cubical representations can be driven from K-Maps and need not necessarily be done by a tabular representation. Function *f (a,b,c,d)* shown below is a four-variable function.
    a. Show the complete list of Prime Implicants of this function in the cubical form.
    b. Show the complete list of Essential Prime Implicants in the cubical form.

c. Show the minimal two-level logic realization of this function in Boolean form. Show all alternatives if the implementation of the function is not unique.

$$f(a,b,c,d) = \sum_m (2, 3, 4, 6, 8, 9, 11, 12, 13), d(7, 15)$$

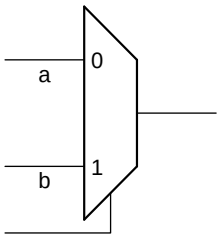
4. Function $f(a,b,c,d)$ shown below is a four-variable function.
   a. Using QM tabular minimization method, show the complete list of Prime Implicants of this function in the cubical form.
   b. Show the complete list of Essential Prime Implicants in the cubical form.
   c. Show the minimal two-level logic realization of this function in Boolean form. Show all alternatives if the implementation of the function if it is not unique.

$$f(a,b,c,d) = \sum_m (2, 3, 4, 6, 7, 8, 9, 12, 13)$$

5. Show SystemVerilog description for a 3-to-8 binary decoder with active high outputs, and an active low enable input (EN). Use nesting of SystemVerilog condition operator, i.e., ?:.

6. Implement function shown with as few 2-to-1 multiplexers as possible.

$$f(a, b, c, d) = \Sigma_m(0, 1, 2, 3, 4, 6, 8, 9, 13, 15), d(5, 11, 14)$$



7. Using multiplexers for random logic realization.
   a. Write SystemVerilog code for a 4-to-1 multiplexer with an active high and an active low enable input.
   b. Use the module of Part A and as few extra gates as possible, to create a module for a 16-to-1 multiplexer.
   c. In a SystemVerilog module, use the module of Part B to implement the 5-variable function given below.

$$f(a, b, c, d, e) = \Sigma_m(0, 1, 2, 3, 4, 6, 8, 9, 13, 15, 17, 18, 20, 22, 24, 27, 29, 30, 31)$$

8. Show the design of a cascadable comparator. The final outputs are to come from the most-significant bits of the comparator.
   a. Write SystemVerilog code for a 1-bit magnitude comparator with *a, b, l, e, g* inputs and *lt, eq,* and *gt* outputs.
   b. In a SystemVerilog module wire eight of comparators of Part a to build an 8-bit comparator. Assign constants to the right-most *l, e,* and *g* inputs as appropriate.