

# Fundamentals of Digital System Design

ECE/CS 3700

Spring 2009, Homework # 2

Due Date: Tu, Feb 17, 2009, 5pm.

**Note:** The following sets of questions correspond to chapters 3 and 4 from the textbook (and some other concepts that I covered in the class). The homework is due on Feb 17; please drop it in the proper ECE homework locker by 5pm. Show all your work. Should you have difficulty in understanding any of the questions, feel free to ask the TAs or the instructor. Good luck!

1. **(K-Map minimization - 25 points)** For the following functions, whose on-set minterms are shown using the  $\sigma(\Sigma)$  notation, derive a minimum Sum-of-Product form expression using Karnaugh maps. Note that your final answer should be a sum-of-product form Boolean expression, derived using cube-covering on the K-maps.

- $F(A, B, C, D) = \sum m(0, 2, 3, 5, 6, 7, 8, 10, 11, 14, 15)$
- $F(A, B, C, D) = \sum m(1, 2, 3, 6, 7, 11)$
- $F(A, B, C, D) = \sum m(2, 3, 5, 7, 10, 11, 13, 14, 15)$
- $F(A, B, C, D, E) = \sum m(2, 5, 7, 8, 10, 13, 15, 17, 19, 21, 23, 24, 29, 31)$
- $F(A, B, C, D, E) = \sum m(0, 4, 18, 19, 22, 23, 25, 29)$

2. **(K-maps with don't cares - 10 points).**

- $F(A, B, C, D) = \sum m(1, 3, 5, 7, 9) + \sum d(6, 8, 12, 13)$
- $F(A, B, C, D) = \sum m(0, 2, 8, 9, 10, 15) + \sum d(1, 3, 6, 7)$

3. **(Quine-McCluskey method of minimization - 20 points).** Solve problem 4.24 given in the textbook, pp 240. Approach the problem systematically. First compute the prime implicants. Then, setup the table covering problem properly and show all the reductions. Give the final *minimum cost cover* in SOP form.

4. **(Quine-McCluskey method of minimization - 25 points).** Solve problem 4.25. Be careful while performing the reductions on the table - consider the cost of the primes!

5. **(FPGA Placement and Routing - 10 points)** Refer to Fig. 3.39 in the textbook, given on page 113. The figure shows how Look-up Tables (LUTs) are interconnected by two layers of wiring: horizontal and vertical. The *blue coloured* cross-mark ( $\times$ ) indicates that a connection *has been made* between the horizontally and vertically drawn wires. Convince yourselves that  $f = f_1 + f_2 = x_1x_2 + x_2'x_3$ .

Based on the above concepts, you are asked to *program* an FPGA, whose LUTs and inter-connection wires are shown in Fig. 1. The function to be implemented is  $f = f_1 \cdot f_2$ , where  $f_1 = a + \bar{b}$  and  $f_2 = \bar{a} + \bar{c}$ . LUT 1 should

implement  $f_1$ . LUT 2 should implement  $f_2$  and LUT 3 should implement  $f_1 \cdot f_2$ . The horizontally and vertically placed interconnection wires are fabricated in different planes. In order to depict a connection between these wires at a cross-point, place a cross-mark ( $\times$ ). The inputs  $a, b, c$  and the output  $f$  have already been connected to the “input-output pads” for you (in)convenience. Have fun!

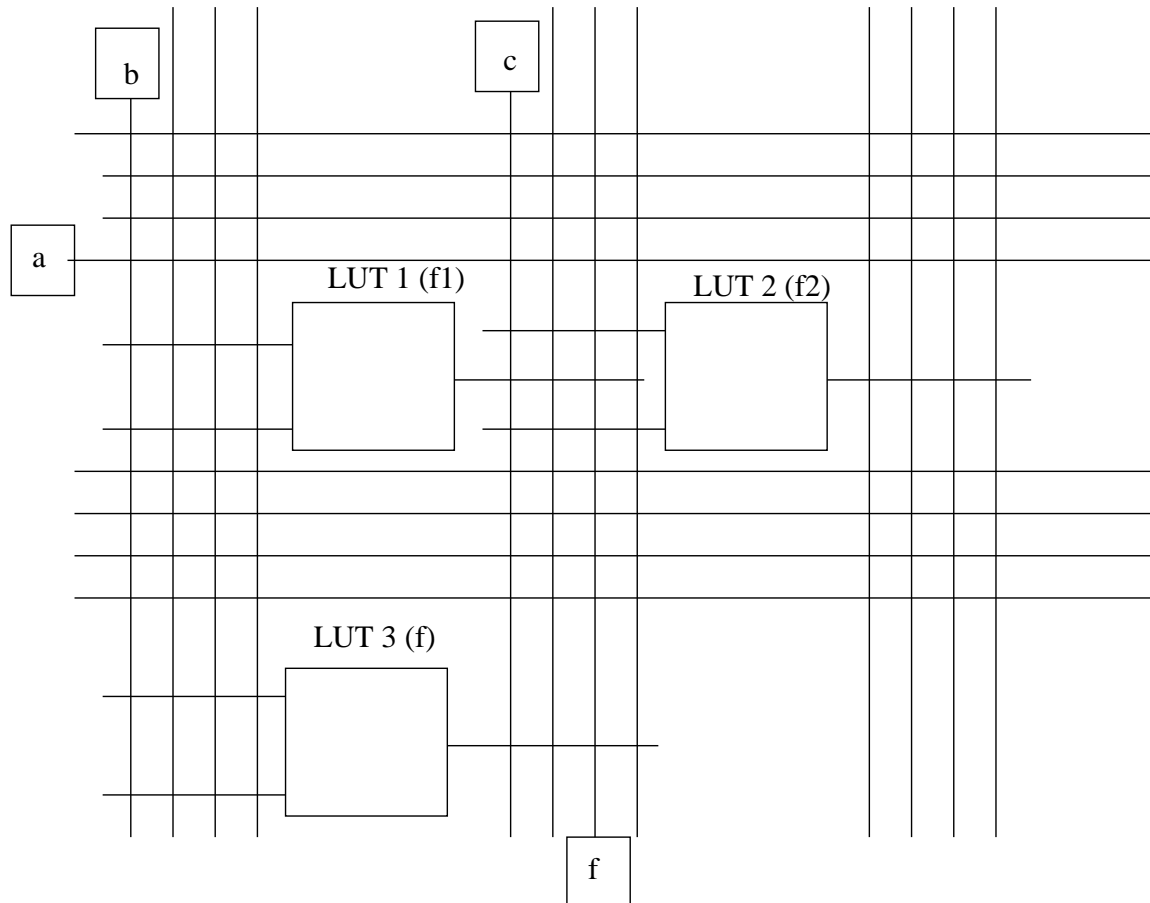


Fig. 1. Fill-up the truth-table entries in the look-up tables. Label the inputs and outputs of each LUT properly. Put a  $\times$  mark to show an electrical connection between the vertical and horizontal metal wires.

6. **(CMOS Logic Gate Design - 10 points)** Recall that CMOS logic gates consist of a pull-down and a pull-up network comprising of NMOS and PMOS transistors, respectively. Moreover, CMOS gates implement *inverting* logic. Keeping this in mind, design a CMOS gate that implements the following Boolean expressions. Note that you are asked to design just one logic gate implementing the entire Boolean function. Use as few transistors as possible. Make sure to label the input (gate of each transistor) variables accordingly. Furthermore, don't forget to show where the output of the logic gate is.

(a)  $\bar{A} + \bar{B} + \bar{C}$

(b)  $(\bar{A} + \bar{B}) \cdot (\bar{C} + \bar{D})$