Fundamentals of Digital System Design

ECE/CS 3700
Spring 2018, Homework # 2
Assigned, Tuesday Feb 6, Due Date: Tuesday, Feb 13, 2018, by midnight in the HW locker.

1) **(FPGA Placement and Routing - 10 points)** Refer to Fig. B.39, in the textbook, given on page 768 Appendix B. 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 $f_1 = a + \overline{b}$ and $f_2 = \overline{a} + \overline{c}$. 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 your (in)convenience. Have fun!

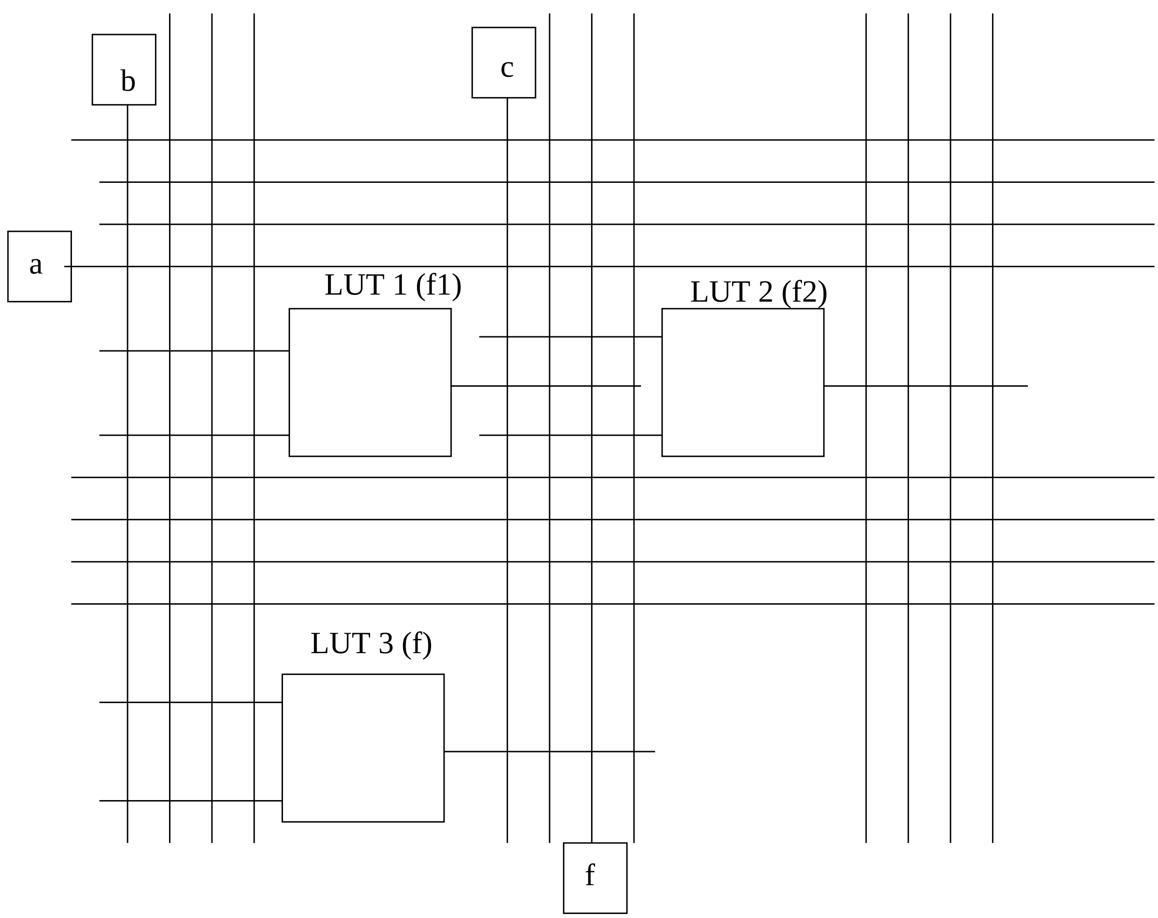


Fig. 1. Fill-up the truth-table entries in the look-up tables. Label the inputs and outputs of each LUT properly. Put a $\times$ mark to show an electrical connection between the vertical and horizontal metal wires.

2) **(MUX-based design - 10 points)** Given a Boolean function $F(a, b, c) = a \oplus b \oplus c$, implement a circuit using *only* multiplexor (MUX) gates and $0, 1$ (ground and V_{DD}) inputs. How many MUXes do you need? Draw a circuit schematic.

   If, in addition to MUXes, 0 and 1 inputs, NOT gates (inverters) are also allowed, can the number of MUXes be reduced in the implementation of F? If so, draw a simplified circuit, and provide the gate count of the number of MUXes and NOT gates needed.

3) **(Boolean Function Manipulation - 10 points)** Let $F(a, b, c) = a \cdot b + a \cdot c + b \cdot c$, then $F(\overline{a}, \overline{b}, \overline{c}) = \overline{a} \cdot \overline{b} + \overline{a} \cdot \overline{c} + \overline{b} \cdot \overline{c}$, where $\overline{a}$ denotes the complement of a. Now answer the following:
   - Is $\overline{F(a, b, c)} = F(\overline{a}, \overline{b}, \overline{c})$?
   - Now consider an *arbitrary* Boolean function $F(a, b, c)$. Is inverting the inputs of F equivalent to inverting the function itself? In other words, for an arbitrary Boolean function $F(a, b, c)$, is $\overline{F(a, b, c)} = F(\overline{a}, \overline{b}, \overline{c})$? If yes, prove it. Otherwise, give a counter-example.

4) **(K-Map minimization - 25 points)** For the following functions, whose on-set minterms are shown using the $sigma(\sum)$ notation, derive a minimum Sum-of-Product (SOP) form expression using Karnaugh maps (K-maps). Note that your final answer should be a sum-of-product form Boolean expression, derived using cube-covering on the K-maps.
   - $F(A, B, C, D) = \sum m(0, 2, 3, 5, 6, 7, 8, 10, 11, 14, 15)$
   - $F(A, B, C, D) = \sum m(1, 2, 3, 6, 7, 11)$
   - $F(A, B, C, D) = \sum m(2, 3, 5, 7, 10, 11, 13, 14, 15)$
   - $F(A, B, C, D, E) = \sum m(2, 5, 7, 8, 10, 13, 15, 17, 19, 21, 23, 24, 29, 31)$
   - $F(A, B, C, D, E) = \sum m(0, 4, 18, 19, 22, 23, 25, 29)$

5) **(K-maps with *don't cares* - 10 points)**. Derive minimum cost SOP forms for these functions.
   - $F(A, B, C, D) = \sum m(1, 3, 5, 7, 9) + \sum d(6, 8, 12, 13)$
   - $F(A, B, C, D) = \sum m(0, 2, 8, 9, 10, 15) + \sum d(1, 3, 6, 7)$

6) **(Another Min cost SOP - 15 points)** Consider the function $f(x_1, \ldots, x_4) = \sum m(0, 3, 4, 5, 7, 9, 11) + D(8, 12, 13, 14)$. Working on a K-map, first generate and list all the *prime implicants* of the function. Subsequently, from among these primes, identify the essential primes, and then derive a *minimum (literal) cost* SOP form Boolean expression. How many product-terms does the min-cost SOP form have? What is the total literal cost of the min-cost SOP?

7) **(Synthesis of a decomposed Boolean function by exploiting** ***don't care*** **conditions - 20 points)** Consider the Boolean function $F(a, b, c, d, e)$ whose K-map is shown below in Fig. 2 (i). Now suppose that a logic synthesis algorithm decomposes F as $F(a, b, c, d, e) = h(g_0(a, b, c), g_1(a, b, c), d, e)$, shown in Fig. 2, where the SOP representations are:

- Functions $g_0 = a'bc + ab'c + abc'$ and $g_1 = a'b'c + abc$
- Function $h = g_0'g_1'e' + g_0g_1'd' + g_0'g_1e$

You are asked to solve the following:

a) From the K-map of $F(a, b, c, d, e)$, identify a minimum SOP form representation of F in its *undecomposed* form in terms of the primary inputs $\{a, b, c, d, e\}$. What is the SOP literal cost of F?

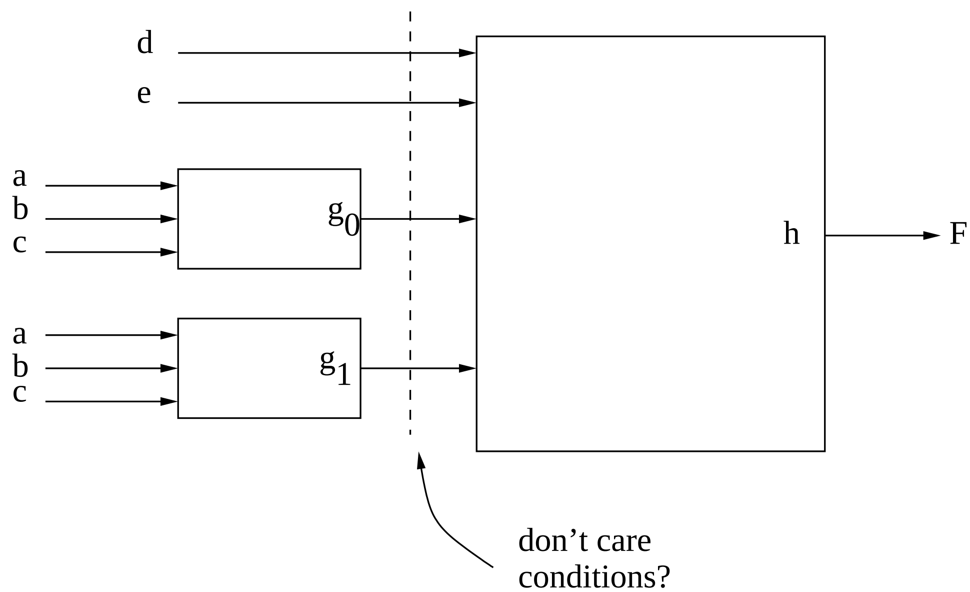
b) Now assume the a decomposition is applied as shown in Fig. 2. Minimize the SOP form of g_0, g_1, h. Are they already given in minimal form?

c) This decomposition creates don't care conditions at the input of the $h(g_0, g_1, d, e)$ block. Identify the don't care conditions at the input of h.

d) Using the don't care conditions, minimize the SOP form of h. What is the total SOP literal cost of g_0, g_1 and h. Do the don't care conditions result in further logic simplification with literal cost savings?

		a	0	0	0	0	1	1	1	1
		b	0	0	1	1	1	1	0	0
		c	0	1	1	0	0	1	1	0
de	00		1	0	1	1	1	0	1	1
	01		0	1	1	0	1	1	1	0
	11		0	1	0	0	0	1	0	0
	10		1	0	0	1	0	0	0	1

(i) F(a, b, c, d, e)



(ii) A decomposed implementation of F(a,b,c,d,e)

Fig. 2. Decomposition of $F(a, b, c, d, e) = h(g_0(a, b, c), g_1(a, b, c), d, e)$. Compute the don't cares at the input of the $h(g_0, g_1, d, e)$ block and simplify the SOP form of h.