Indian Institute of Technology (IIT-Bombay)

AUTUMN Semester, 2024
COMPUTER SCIENCE AND ENGINEERING

CS230: Digital Logic Design + Computer Architecture

Quiz I

Full Marks: 20

Time allowed: (0.75 + ϵ) hours

Roll Number: ______________________________

Name: ______________________________

Answer the questions in the spaces provided. If you run out of room for an answer, continue on the back of the page.

1. You are given three-input logic gates, each realizing the function $g(x, y, z) = x \oplus yz$. Use this function as a block to implement the following function

$$f = (a + b)c + ab'.$$

Show all your steps. (5 marks)

Solution: First, observe that $g(1, y, z) = (yz)'$, which is a NAND gate. Also, $g(1, 1, z) = z'$. Now, convert f to a NAND-based circuit.

$$f = (a + b)c + ab' = (((a + b)c)')' + ((ab')')' = (((a'b')'.c)'.(ab')')'.$$

Finally, replace each NAND gate with $g(1, y, z)$ and each complement with $g(1, 1, z)$.

2. (**Algebraic Normal Form**) In the class, we have discussed about two canonical forms, namely, SOP and POS. In this question, we introduce another one called Algebraic Normal Form (ANF). Informally, ANFs are expressions involving XOR and AND gates. For example, $y = x_1x_2 \oplus x_3$ is in ANF. More formally, a Boolean function $f(x_1, x_2, \cdots, x_n)$ in ANF form is written as:

$$f(x_1, x_2, \cdots, x_n) = c_0 \bigoplus_{1 \le i \le n} c_i x_i \bigoplus_{1 \le i,j \le n} c_{ij} x_i x_j \bigoplus \cdots \bigoplus c_{1 \cdots n} x_1 x_2 \cdots x_n,$$

where each coefficient $c_t \in \{0, 1\}$. It can be proven that every Boolean function of n variables can be uniquely represented in this form.

Convert the following Boolean function in ANF form:

$$f(x_0, x_1, x_2) = \sum m(0, 1, 5)$$

Hint: Let a minterm of 3 bits is represented as $m = (m_0, m_1, m_2)$. Define $x_i^{(m_i)}$ such that $x_i^{(m_i)} = 1$ if and only if $x_i = m_i$. Now define $x^m = \prod x_i^{(m_i)}$. This is a minterm for the ANF representation. (5 marks)

Solution: Let us write f as the sum of minterms as $f = m_0 + m_1 + m_5$. Observe that we can also write $f = m_0 \oplus m_1 \oplus m_5$. This is because only one minterm can be 1 for a given input to f. Now, we need to convert each minterm to an ANF expression. Let $m_i = (a_0, a_1, a_2)$ with each $a_j \in \{0, 1\}$. We can write $m_i = \prod_{j=0}^{2}(x_j \oplus a_j + 1)$. For our given function, therefore, we can write,

$$m_0 = x_0'x_1'x_2' = (x_0 \oplus 1)(x_1 \oplus 1)(x_2 \oplus 1)$$

. Similarly, we can represent m_1 and m_5. Finally, we can express $f = m_0 \oplus m_1 \oplus m_5$, which will be the ANF expression. The simplified solution is

$$f = 1 \oplus x_0 \oplus x_1 \oplus x_0x_1 \oplus x_0x_2 \oplus x_0x_1x_2$$

3. Find the simplest function $g(A, B, C, D)$ that will make the function $f = A'BC + (AC + B)D + g$ self dual. The *dual* of a Boolean function can be obtained by replacing all the ANDs with ORs and vice versa. The constant terms are replaced with their respective complements. Boolean function can be self-dual if the truth table contains the same number of 0's and 1's, and none of the minterm pairs are mutually exclusive (that is, no two minterms are complements to each other). (5 marks)

Solution: For this question, first, construct the K-Map for the already given minterms of f. The K-Map is presented in Figure 1. From the given minterms of f (except those from g), we can fill in the 1's in the K-Map. However, since the function is self-dual, we can also find out the locations of the 0's from the given 1's. This is because, for a self-dual function if $x_0x_1 \cdots x_n$ is a minterm, then $x_0 + x_1 + x_2 + \cdots x_n$ must also be a maxterm. Therefore, if we have a 1 in a cell, then we must have a 0 in a cell whose code is the complement to the 1-cell. With this observation, we can fill in 6 0's. Now, there are four cells which can be filled. By the property of self-dual, two of them must be filled in with 1's. For the given K-Map, we observe that either $(0001, 0011)$ or $(1110, 1100)$ must be 1. Therefore, we can have two possible minimal forms for f (i.e. two minimal g), given as:

$$f = A'BC + ACD + BD + A'D$$

and

$$f = AB + BD + A'BC + ACD$$

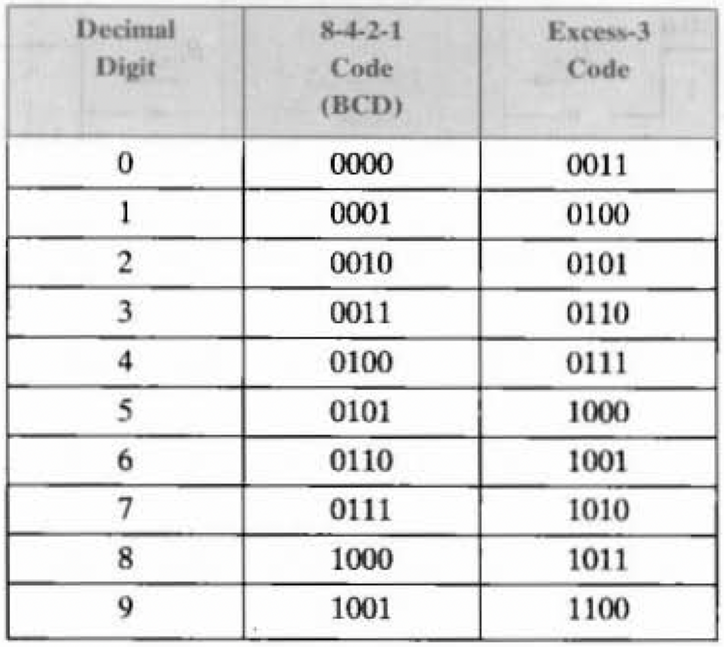
4. **(BCD to Excess-3 Code Converter)** In this question, we construct a **serial** BCD to Excess-3 code converter from serial input. In other words, the input will be provided, and the output will be generated one bit at a time. Excess-3 code can be generated by adding $(0011)_2$ with the BCD code. Therefore, you need to consider 4 bits of input at a time as a valid BCD code. The expected input and outputs are provided in Figure 5. Observe that the inputs will be provided from LSB to MSB, and the outputs will be generated in the same manner (more precisely, the LSB of the input will be processed first, and the LSB of the output will be generated first). So you have to consider each entry in the table from right to left while processing.

   (a) Derive the state machine and state table for this sequential circuit. Show all necessary steps for deriving and minimizing the machine. (3 marks)

   Hint 1: Your state machine will have at most 7 states. So, you may need to perform state minimization.

AB \ CD	00	01	11	10
00	0			0
01	0	1	1	1
11		1	1	
10	0	0	1	0

Figure 1: K-Map for question 3



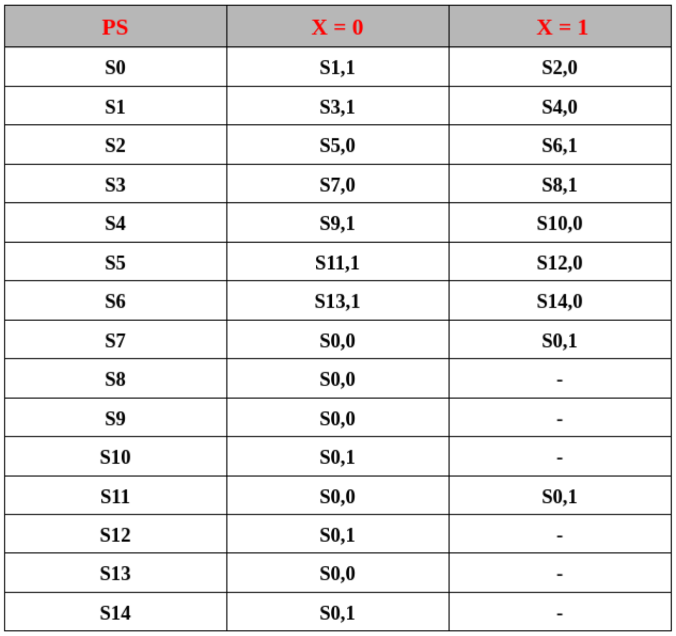
Decimal Digit	8-4-2-1 Code (BCD)	Excess-3 Code
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Figure 2: BCD to Excess-3 Code Converter

Hint 2: You may encounter incompletely specified/unspecified states and outputs. Just follow the simple convention: if a state S_i have some unspecified transitions (say the transition is defined for input $x = 0$ and not defined for $x = 1$), then you should consider the unspecified transition as a "don't care" and put appropriate state and output symbol there, so that you can minimize the machine as much as possible.

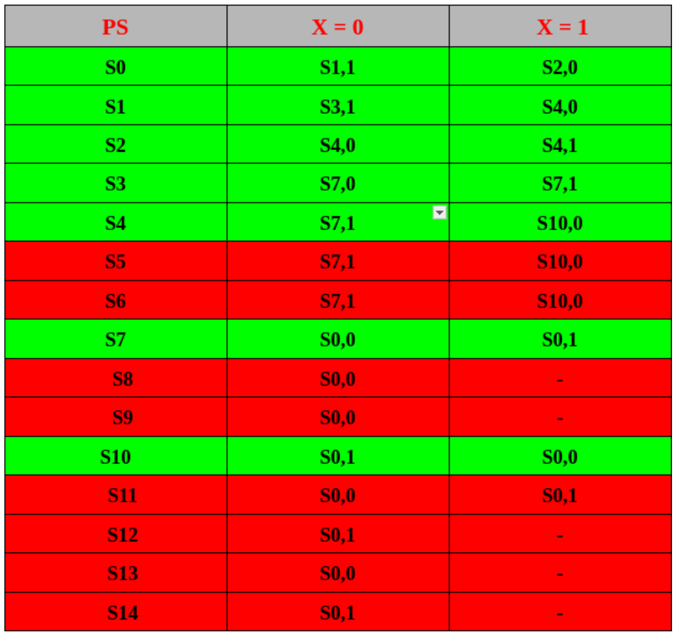
(b) Perform the state assignment using binary encoding and derive the state and output equations by solving K-Maps. Clearly show every step of your calculation. (2 marks)

Solution:



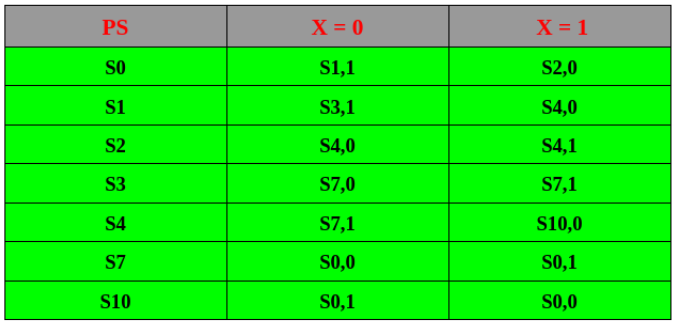
PS	X = 0	X = 1
S0	S1,1	S2,0
S1	S3,1	S4,0
S2	S5,0	S6,1
S3	S7,0	S8,1
S4	S9,1	S10,0
S5	S11,1	S12,0
S6	S13,1	S14,0
S7	S0,0	S0,1
S8	S0,0	-
S9	S0,0	-
S10	S0,1	-
S11	S0,0	S0,1
S12	S0,1	-
S13	S0,0	-
S14	S0,1	-

Figure 3: State Table



PS	X = 0	X = 1
S0	S1,1	S2,0
S1	S3,1	S4,0
S2	S4,0	S4,1
S3	S7,0	S7,1
S4	S7,1	S10,0
S5	S7,1	S10,0
S6	S7,1	S10,0
S7	S0,0	S0,1
S8	S0,0	-
S9	S0,0	-
S10	S0,1	S0,0
S11	S0,0	S0,1
S12	S0,1	-
S13	S0,0	-
S14	S0,1	-

Figure 4: Minimizing State Table



PS	X = 0	X = 1
S0	S1,1	S2,0
S1	S3,1	S4,0
S2	S4,0	S4,1
S3	S7,0	S7,1
S4	S7,1	S10,0
S7	S0,0	S0,1
S10	S0,1	S0,0

Figure 5: Minimized State Table