Indian Institute of Technology (IIT-Bombay)

AUTUMN Semester, 2025
COMPUTER SCIENCE AND ENGINEERING

CS230: Digital Logic Design + Computer Architecture

Quiz I

Full Marks: 15

Time allowed: 0.5 hours

1. **Easy:** Choose the correct answer(s) – one or multiple can be correct: An n-bit Balanced Boolean Function is a function with n-bit inputs and 1-bit output which has an equal number of 0's and 1's in the "output" column of its truth table. Which of the following is (are) true for such an n-bit Balanced Boolean Function?

   1. The K-map will have many don't cares.
   2. The K-map will have 2^{n-1} 0's.
   3. The complement of such a function is also a balanced function.
   4. The K-map will have 2^{n-2} 0's.

   Give **proper justification** to your answer(s). No marks will be given without **proper justification**. (2 marks)

   Solution: 2,3. There are a total 2^n rows in the truth table, and, therefore, same number of cells in the K-map. Half of them (2^{n-1}) are 0's and the rest are 1's. Therefore, 2 is correct. Also, the complement of a function changes the 0's to 1's (in the output column) and vice versa. So 3 is also correct.

   Grading Policy: 0.5 for choosing each correct option + 0.5 for each correct justification. Also, if you consider that there are don't cares and, based on that argue 2 as conditionally true, you shall get marks.

   Siddesh+Neel

2. **Medium:** You have to make a $2^n : 1$ MUX (A multiplexer with 2^n inputs and 1 output) using only $2 : 1$ MUXes. The number of $2 : 1$ MUXes needed is (**only one answer is correct**):

   1. $n - 1$, $2 : 1$ MUXes will be needed .
   2. 2^{n-1}, $2 : 1$ MUXes will be needed.
   3. 2^n, $2 : 1$ MUXes will be needed.
   4. $2^n - 1$, $2 : 1$ MUXes will be needed.

Give **clear and proper justification** to your answer(s). No marks will be given without **proper justification**. (3 marks)

Solution: 4. First, observe that a $2^n : 1$ MUX will be a binary tree of $2:1$ MUXes (see an example of $8:1$ MUX from Tutorial 1). There will be $\log_2(2^n) = n$ layers in this tree. The first layer (input) will have 2^{n-1} $2:1$ MUXes. The second layer will have 2^{n-2} $2:1$ MUXes, and so on. Finally, the nth layer (output) will have only one $2:1$ MUX. Therefore, the total number of $2:1$ MUXes needed is $1+2+2^2+\cdots+2^{n-1} = 2^n - 1$.

Grading Policy: 1 for choosing correct option. 2 for clear and proper justification.

Aritra+Shivam

3. **Self-Correcting Counter (Easy):** A *self-correcting counter* is a sequential counter designed so that:

- It follows a desired sequence of states (for example, a custom binary sequence).
- If due to noise, power-up uncertainty, or fault, the counter enters an unused/invalid state (a state which is **not** a part of the binary sequence of the counter), the logic guarantees that on the very next clock cycle, the counter will automatically transition into a valid state of the sequence.

Construct a counter for the following binary sequence using T flip-flops: $000 \rightarrow 010 \rightarrow 111 \rightarrow 100 \rightarrow 011$. However, the counter must be self-correcting, that is, from an illegal state it should always come back to the 000 state in the next clock cycle. Also, from the last valid state 011, the counter must come back to 000. The counter is a 3-bit counter.

(a) Construct the state-table clearly showing the present states and next states. (2 marks)

(b) Derive the equations for the T-flip-flop inputs by clearly showing and solving K-maps. Your expression should only have NOT, 2-input AND and 2-input ORs. The expressions should be minimal in terms of gate count. (3 marks)

Solution: The state-transition table is shown in Table 3. There are 3-bit present states $Q_2\ Q_1\ Q_0$ and 3-bit next states $Q_2^+\ Q_1^+\ Q_0^+$. Each illegal state transits to 000 in the next state. The T-flip-flop inputs can be derived as $T_i = Q_i \oplus Q_i^+$ with $0 \le i \le 2$. The (three) K-maps will have inputs $(Q_2\ Q_1\ Q_0)$ and outputs as T_2, T_1, and T_0. The final equations are as follows:

$$T_0 = Q_0 + Q_1.\overline{Q_2} + Q_2.\overline{Q_1}$$

$$T_2 = Q_1.\overline{Q_0} + Q_2.\overline{Q_1}$$

$$T_1 = \overline{Q_1}.\overline{Q_0} + Q_1.Q_0 + Q_2.\overline{Q_0}$$

(Alternative)

$$T_1 = \overline{Q_1}.\overline{Q_0} + Q_1.Q_0 + Q_2.Q_1$$

Grading Policy: (a) Deduct 1 mark if upto one entry in the table is wrong. Else, either give 0 or 2. b) Each K-map 0.5 + each expression 0.5.
Shoaib+Jayshree

Present State (Q2 Q1 Q0)	**Next State ($Q2^+$ $Q1^+$ $Q0^+$)**	**T2**	**T1**	**T0**
000	010	0	1	0
001	000	0	0	1
010	111	1	0	1
011	000	0	1	1
100	011	1	1	1
101	000	1	0	1
110	000	1	1	0
111	100	0	1	1

4. **Majority Function (a bit tricky)** A 3-input *majority function/majority gate* is a function which evaluates to 1(0) only if at least two of the 3 input bits are 1(0).
   (a) Derive the (minimized) logic expression for the majority function/gate with inputs (x_1, x_2, x_3). (1 mark)
   (b) Prove/disprove with proper logic: Majority gate/function is a universal gate. "Proving" means: you have to show that all the basic gates (AND, OR, NOT) or at least one of the universal gates (NAND, NOR) **can** be constructed with a majority gate or a combination of majority gates. Disproving" means: you have to show that **at least one of the basic gates** (AND, OR, NOT) or **none of the universal gates** (NAND, NOR) can be constructed using a majority gate, or a combination of such majority gates. (4 marks)

Solution: The truth table for 3-input majority gate is given in Table 1. Solving the K-map gives: $\mathsf{Majority}(x_1, x_2, x_3) = x_1x_2 + x_2x_3 + x_1x_3$.

Observe that you **cannot** construct a NOT gate using majority gates. To understand why, first consider a single majority gate $\mathsf{Maj}(x_1, x_2, x_3)$. Without loss of generality, let us consider that we want to invert the input x_1. Observe that, if we keep (x_2, x_3) fixed to some value (00, or 01, or 10, or 11), and change x_1 from $0 \to 1$, the $\mathsf{Maj}(x_1, x_2, x_3)$ output always changes from $0 \to 1$. In short, for x_1 going from $0 \to 1$, a transition of $1 \to 0$ is never possible. Next consider a network of such majority gates in which x_1 appears in

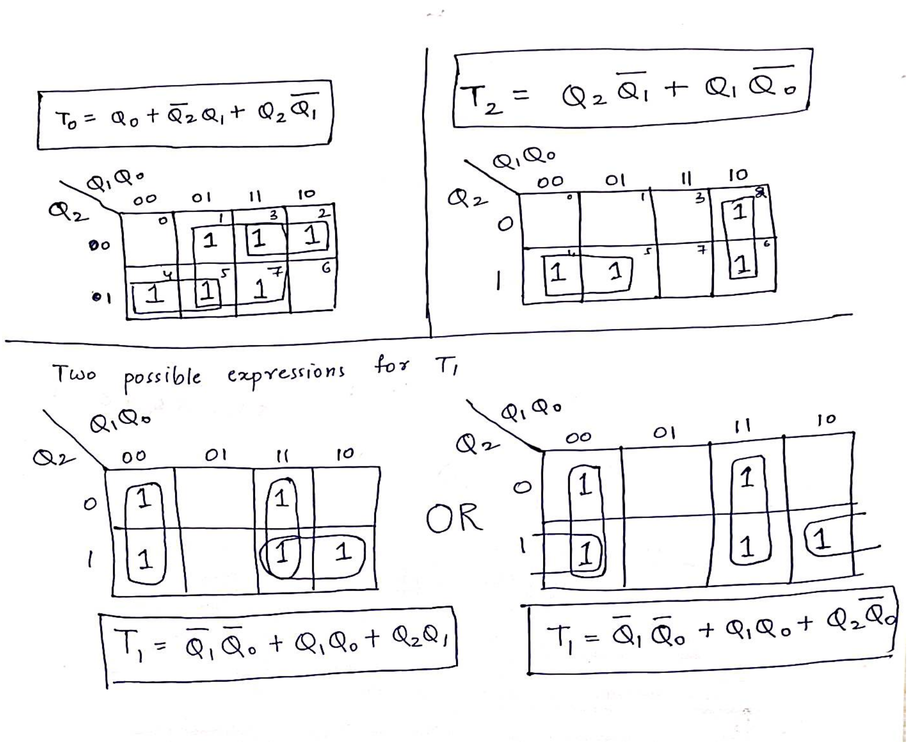
$T_0 = Q_0 + \overline{Q_2} Q_1 + Q_2 \overline{Q_1}$
$T_2 = Q_2 \overline{Q_1} + Q_1 \overline{Q_0}$
Two possible expressions for T_1
OR
$T_1 = \overline{Q_1}\,\overline{Q_0} + Q_1 Q_0 + Q_2 Q_1$
$T_1 = \overline{Q_1}\,\overline{Q_0} + Q_1 Q_0 + Q_2 \overline{Q_0}$

Figure 1: Question 3 (K-maps)

many gates as inputs. Since none of the gate outputs can transit $1 \to 0$ for an input transition of $x_1 : 0 \to 1$, it is impossible to invert x_1, even using such a network of majority gates. Therefore, it is not possible to implement a NOT gate, and a majority gate is not universal.

Grading Policy: (a) Give 1 only if the correct expression is derived. Else 0. b) Give 2 if they have identified that making NOT is hard. Give 2 marks if they somehow identify that inversion is not possible with majority gate (even if they go for NAND or NOR). Give 4, only if they can properly justify as given in the answer key.

Enanko+Mrityunjay+Iqbal

x_1	x_2	x_3	$\mathsf{Maj}(x_1, x_2, x_3)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table 1: Truth table of 3-input majority function

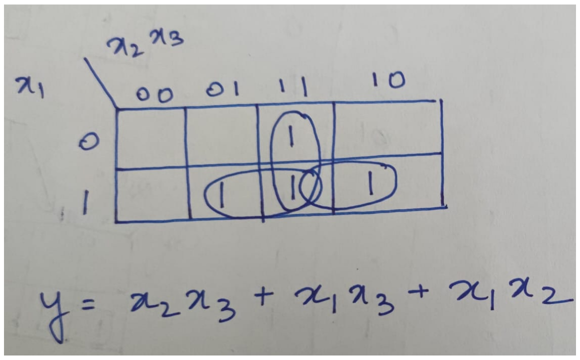

Figure 2: Question 4 (K-maps)