

Name (in CAPITAL letters)

Entry Number

Kerberos ID

Department of Computer Science and Engineering, IIT Delhi

Digital Logic and System Design (COL 215)

I Semester 2025-26, Mid-semester Exam, 16 Sep 2025, 8:00 AM to 10:00 AM

Maximum Marks: 80

Please write your answers ONLY IN THE SPACE BELOW THE QUESTIONS.

Use reverse side of paper for rough work.

1. [5 + 5 = 10 Marks] Consider a Gray code sequence of n -bit numbers.
 (a) Suppose we interchange two adjacent (consecutive) numbers. Is the resulting sequence still in Gray code? Always? Sometimes? Never? Justify. [Assume that the first and last numbers are adjacent.]

Never. If we interchange two numbers at positions i and $i+1$, the number of bits changing between numbers at positions $i-1$ and i becomes 2. Example:

$i-1$: 000
 i : 001
 $i+1$: 011

If we interchange numbers at i and $i+1$, then the new sequence is:

$i-1$: 000
 i : 011
 $i+1$: 001

2 bits are changing between 000 and 011

- (b) Suppose we interchange two adjacent bit positions of every number. For example, the 2nd bit and 3rd bit of every number are interchanged. Is the resulting sequence still in Gray code? Always? Sometimes? Never? Justify.

Yes, always. *Hamming distance* between two numbers (number of bits changing in respective positions) does not change if we interchange the bits at two positions, since the interchange happens in both numbers. For example:

$i-1$: 000
 i : 001

If we interchange bits at positions 2 and 3, the new sequence is:

$i-1$: 000
 i : 010

One only bit is changing.

2. **[5 + 5 = 10 Marks]** Consider a 4-variable K-map with 0/1/x in its cells.
(a) How many distinct 4-variable K-maps are possible? Why? [*Distinct* refers to the specification – patterns of 0/1/x, not the final Boolean function.]

Each cell can have 3 possibilities: 0/1/x

We have $2^4 = 16$ cells

Number of maps = $3 \times 3 \times \dots \times 3$ (16 times) = 3^{16}

- (b) Considering that every 'x' must be ultimately resolved to a '0' or '1', how many distinct 4-variable K-maps are possible after this resolution is done? Why?

Each resolved cell now has 2 possibilities: 0/1

We have $2^4 = 16$ cells

Number of maps = 2^{16}

3. **[5 + 5 = 10 Marks]** A country where base r is used for representing numbers, decided to ban the digit x , where $0 < x < r$.
(a) Will people in this country still be able to write all numbers, in spite of the ban? Why?

No. Gaps will remain on the number line. E.g., the number x itself cannot be represented.

- (b) Suppose the President of the country decides to make a compromise, and permits people to use bases other than r . Would this compromise permit people to write all numbers in spite of the ban on x ? Does this depend on the chosen base? Justify.

If the new base $1 < R < x$, then we don't need x , as x can be represented using multiple digits of the smaller base, and all other numbers can also be represented. For example, if $x = 7$ and $R = 5$, we have no problem, since 7 can be represented as 12 in base 5, etc.

If the new base $R > x$, then some numbers (such as the number x itself) cannot be represented without using the digit x .

4. [10 Marks] In the K-map minimisation process, is it possible to end up with two different Boolean expressions that are both optimal? ~~If yes, explain why; if no, give a counter example.~~ If yes, give an example; if no, explain why.

Yes, it is possible. In the K-map below, the two regions grouped by **red** dashes need to be retained, but of the two regions grouped by **black** dashed lines, only one needs to be retained in an optimal expression.

ab \ cd	00	01	10	11
00	0	0	0	0
01	0	1	1	0
11	0	1	0	0
10	1	1	0	0

5. [10 Marks] In an n -bit adder with inputs A , B , C_{in} and outputs S and C_{out} , the carry-in (C_{in}) is propagated all the way to carry-out (C_{out}). What condition must the inputs A and B satisfy? Why?

We need to have: $A \oplus B = 111\dots 1$

If $A_i \oplus B_i = 1$, then carry-in is propagated to carry-out

Same condition as: $P_i = 1$ for all i

6. **[10 Marks]** In the comparator implementation (which generates all functions $=$, $<$, and $>$), why did we use the AND gates and inverters at the first stage instead of XOR/XNOR gates?

Intermediate signals of the XOR function (such as AB' and $A'B$) are used in the $>$ and $<$ computation. These signals would not be available if we used an XOR/XNOR gate, which has only one output.

7. **[10 Marks]** Suppose we need to multiply an n -bit number with a known CONSTANT integer between 0 and 15 (such as 12). Show how to implement this multiplication efficiently in hardware. You are allowed to use adders but *not allowed to use multipliers* in your implementation.

Consider the binary representation of the number (Decimal 12 is Binary 1100). For every 1 in the representation of the constant, we need to first generate M shifted by the position corresponding to the 1 (which is done by just appending the right number of 0's to M), and then add all such numbers. For 12 (binary 1100), we need to perform:

M appended by 000 + M appended by 00

which is the same as $8M + 4M$. No multipliers are needed. Since we know the constant, we know how many 0's to append for each 1, and how many adders we need.

8. **[10 Marks]** We studied two different ways of generating the carry-out at the i -th bit position of an adder:

$$C_{i+1} = x_i y_i + y_i C_i + x_i C_i$$

$$C_{i+1} = G_i + P_i C_i$$

Which implementation is better? Justify. [*Limit yourself to analysing only these Boolean expressions; ignore the rest of the adder.*]

G_i and P_i have to be generated first. We have:

$$G_i = x_i y_i$$

$$P_i = x_i \oplus y_i = x_i y_i' + x_i' y_i$$

$$C_{i+1} = G_i + P_i C_i = x_i y_i + (x_i y_i' + x_i' y_i) C_i = x_i y_i + x_i y_i' C_i + x_i' y_i C_i$$

The first version is better, since second version has more literals.