

Name: Rijurekha Sen

Roll No:

(COL 2015/COL215)

Sep 16, 2026

Mid-term

Duration: 120 minutes

(50 marks)

Note: Be concise in your writing. You can use rough sheets for calculations. But you cannot submit any additional sheet for grading on Gradescope. So make sure you are certain when you write something (after rough work, or use a dark pencil).

If you cheat or any electronic device is found on you, you will get an F in this course. More importantly, constant copying from LLMs and friends will make your brain cells non-functional. Think, and protect your brain cells.

1. Boolean Algebra: Boolean function manipulation, minimization

[Total: 5 marks]

- (a) For the following function, derive an expression containing only NAND operations. Show the essential steps.

$$F = (a + b)c + d$$

[2 marks]

- (b) A student makes the following claims about grouping 1s in a Karnaugh map:

- Every 1 in a K-map must belong to exactly one group.
- The number of cells in a valid K-map group must be a power of two; therefore, a valid group cannot contain exactly three or six cells.

For each claim, state whether it is correct or incorrect and briefly justify your answer.

[2 marks]

- (c) A student argues that

$$X + Y = X + Z \implies Y = Z,$$

because X can be cancelled from both sides.

Is this reasoning valid in Boolean algebra? Explain.

[1 mark]

(a) $F = (a + b)c + d$

NAND gate can implement NOT $\bar{x} = x \text{ NAND } x$

Using De Morgan's laws $a + b = \overline{\bar{a} \cdot \bar{b}}$

$$= (\overline{a \text{ NAND } a}) \text{ NAND } (\overline{b \text{ NAND } b})$$
$$= A$$
$$F = Ac + d = \overline{\overline{Ac} \cdot \bar{d}} \quad (\text{again using De Morgan})$$
$$= (A \text{ NAND } c) \text{ NAND } (d \text{ NAND } d)$$
$$= (((a \text{ NAND } a) \text{ NAND } (b \text{ NAND } b)) \text{ NAND } c) \text{ NAND } (d \text{ NAND } d)$$

(b) (i) Incorrect, Every 1 must be covered in at least one group, but a cell may belong to more than 1 group.

(ii) Correct. A valid rectangular K-map group contains 1, 2, 4, 8, 16 ... cells. These sizes arise because each grouping eliminates some number of boolean variables. A group of size 2^q eliminates q variables.

(c) The reasoning is not valid. Boolean OR does not obey ordinary cancellation.

Eg. let $X = 1$ $Y = 0$ $Z = 1$

$$X + Y = 1 + 0 = 1$$

$$X + Z = 1 + 1 = 1$$

$$X + Y = X + Z \text{ but } Y \neq Z$$

When $X = 1$, both sides are 1, irrespective of the values of Y and Z . Hence equality of the two expressions does not imply $Y = Z$.

Name: _____

Roll No: _____

2. Counting Boolean Functions with stated characteristic: k -Stable Boolean Functions [Total: 10 marks]

Let $n \geq 2$, and $f: \{0,1\}^n \rightarrow \{0,1\}$ be a Boolean function. For each assignment $\mathbf{a} = (x_2, \dots, x_n) \in \{0,1\}^{n-1}$, consider the pair of input vectors $(0, \mathbf{a})$ and $(1, \mathbf{a})$. For an integer k satisfying $0 \leq k \leq 2^{n-1}$, the function f is said to be k -stable in its first variable if $f(0, \mathbf{a}) = f(1, \mathbf{a})$ for exactly k assignments of $\mathbf{a}$, and $f(0, \mathbf{a}) \neq f(1, \mathbf{a})$ for every remaining assignment of $\mathbf{a}$.

- How many pairs of the form $\{(0, \mathbf{a}), (1, \mathbf{a})\}$ are there? Explain your answer. [2 marks]
- Suppose that a specified set of exactly k pairs is required to be stable, and every remaining pair is required to be non-stable. For each of the following types of pairs, determine the number of possible ordered output pairs $(f(0, \mathbf{a}), f(1, \mathbf{a}))$ to be (i) a stable pair, (ii) a non-stable pair. Hence determine the number of Boolean functions corresponding to this fixed selection of exactly k stable pairs. [3 marks]
- In how many ways can the set of exactly k stable pairs be selected? Hence derive a closed-form expression for the number $N_k(n)$ of n -variable Boolean functions that are k -stable in their first variable. [2 marks]
- Using your answer from part (c), show that $\sum_{k=0}^{2^{n-1}} N_k(n) = 2^{2^n}$, which is the total number of Boolean functions of n variables. [3 marks]

(a) $\{(0, \mathbf{a}), (1, \mathbf{a})\}$ is the form of the pair where each assignment for $\mathbf{a} = (x_2, x_3, \dots, x_n)$ determines exactly one pair. Number of pairs = possible assignments of $\mathbf{a}$
 $= 2^{n-1}$ (as there are $n-1$ variables in $\mathbf{a}$, each of which can independently take the values 0 or 1)

Let $A = 2^{n-1}$

(b) For a stable pair, the outputs must be equal.

Possible outputs for stable pairs are

$$(f(0, \mathbf{a}), f(1, \mathbf{a})) = (0, 0) \text{ or } (1, 1)$$

For a non-stable pair, the outputs must be different.

Possible outputs for non-stable pairs are

$$(f(0, \mathbf{a}), f(1, \mathbf{a})) = (0, 1) \text{ or } (1, 0)$$

For a specified " k " stable pairs, there are 2^k choices for stable pair outputs and 2^{A-k} choices

for non-stale pair outputs. Number of functions for this particular selection of $k = 2^k \cdot 2^{A-k}$
 $= 2^A$
 $= 2^{2^{n-1}}$

(c) there are $A = 2^{n-1}$ input pairs. Exactly k of them need to be stale. Number of ways to make this selection $\binom{A}{k} = \binom{2^{n-1}}{k}$

For each selection, part (b) shows there are 2^A possible functions.

Total number of n variable boolean functions, that are k -stale in their first variable $= \binom{2^{n-1}}{k} 2^A$
 $= \binom{2^{n-1}}{k} 2^{2^{n-1}}$

(d) Summing over all k $\sum_{k=0}^{2^{n-1}} \binom{2^{n-1}}{k} 2^{2^{n-1}}$

$$= 2^{2^{n-1}} \sum_{k=0}^{2^{n-1}} \binom{2^{n-1}}{k} = 2^A \left[\sum_{k=0}^A \binom{A}{k} \right]$$

Binomial theorem $\rightarrow (a+b)^n = \sum_{r=0}^n \binom{n}{r} a^{n-r} b^r$

$$\sum_{k=0}^{2^{n-1}} N_k(n) = 2^A \cdot 2^A = 2^{2A} = 2^{2 \cdot 2^{n-1}} = 2^{2^n}$$

3. Combinational Circuit Design: Carry-Skip Adder**[Total: 15 marks]**

A ripple-carry adder is slow because a carry may have to pass through several consecutive full adders. A *carry-skip adder* reduces this delay by allowing a carry to bypass a block of full adders. For bit position i , define the generate and propagate signals as $G_i = A_i B_i$ and $P_i = A_i \oplus B_i$, so that $C_{i+1} = G_i + P_i C_i$ and $S_i = P_i \oplus C_i$. Here, $+$ denotes Boolean OR and multiplication denotes Boolean AND.

For a block containing bits j through k , define its group-propagate signal as $GP_{j:k} = P_j P_{j+1} \cdots P_k$.

If $GP_{j:k} = 1$, every bit in the block propagates its input carry. The carry entering the block can therefore bypass the ripple-carry logic. A 2-to-1 multiplexer selects:

$$\begin{cases} \text{the block input carry,} & GP_{j:k} = 1, \\ \text{the ripple-carry output of the block,} & GP_{j:k} = 0. \end{cases}$$

Assume that all A_i , B_i , and the initial carry C_0 are available at time 0. Use the following delays:

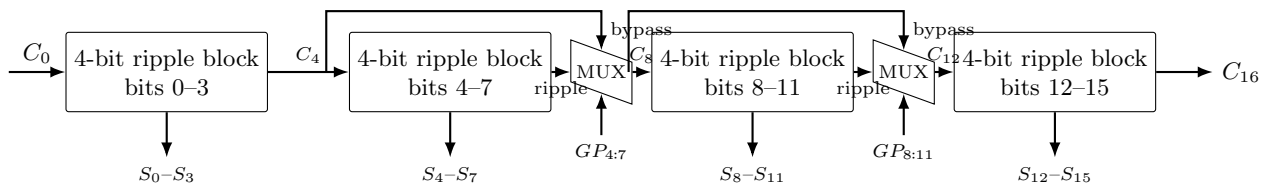
- producing P_i and G_i from A_i and B_i : 1 time unit;
- producing C_{i+1} once P_i , G_i , and C_i are valid: 1 time unit;
- producing S_i once P_i and C_i are valid: 2 time units;
- producing a group-propagate signal once the corresponding P_i signals are valid: 1 time unit; and
- propagation through a 2-to-1 multiplexer: 2 time units.

A multiplexer output becomes valid 2 time units after both its select input and its selected data input are valid. The delay of the adder is the time by which all sum outputs and the final carry-out are valid.

(a) Explain why $GP_{j:k} = 1$ guarantees that the carry entering the block can safely bypass the block. Your explanation must consider both possible values of the block input carry. **[3 marks]**

(b) Consider the following 16-bit carry-skip adder. It is divided into four equal blocks.

The first and final blocks do not have bypass multiplexers. Each of the two intermediate blocks has a bypass multiplexer.



Consider a path in which a carry is generated at bit position 0, propagates through bit positions 1–3 of the first block, bypasses the two intermediate blocks, and then propagates through bit positions 12–14 of the final block to produce S_{15} .

- Determine when C_4 becomes valid at the output of the first block.
- Determine when the carry reaches the input of the final block, that is, when C_{12} becomes valid.
- Determine when S_{15} becomes valid.

[6 marks]

(c) An n -bit carry-skip adder is divided into m equal blocks of b bits each, where $n = mb$.

As in part (b), the first and final blocks do not have bypass multiplexers. A bypass multiplexer is placed after each of the $m - 2$ intermediate blocks.

Consider a path that propagates through the first block, bypasses all intermediate blocks, and propagates through the final block to produce the most significant sum bit.

- Derive the delay of this path in terms of b and m . Clearly identify the delay contributed by the first block, the $m - 2$ multiplexers, and the final block.
- Determine the delay of this path in a 32-bit carry-skip adder for each of the following block sizes: $b = 4$ and $b = 8$.

[4 marks]

(d) A carry-lookahead adder and a carry-skip adder both reduce carry-propagation delay. Briefly explain the fundamental difference between the two techniques.

[2 marks]

(a) For a block containing bit positions j through k ,
 $G_{P_{j:k}} = P_j P_{j+1} \dots P_k$. If $G_{P_{j:k}} = 1$, then
 $P_j = P_{j+1} = \dots = P_k = 1$. Since $P_i = A_i \oplus B_i$, $P_i = 1$
means A_i and B_i are different. Therefore $C_i = A_i \cdot B_i = 0$
The carry equation becomes

$$C_{i+1} = G_i + P_i C_i = 0 + 1 \cdot C_i = C_i$$

Thus every bit position passes its input carry unchanged to the next position. Applying this through the entire block gives $C_{k+1} = C_j$

If $C_j = 0$, every bit propagates 0 and $C_{k+1} = 0$

If $C_j = 1$, every bit propagates 1 and $C_{k+1} = 1$

So it is safe for the multiplier to bypass the ripple carry output and directly select the block input carry.

→ low latency / more area

(d) Carry look ahead adder computes carry signals directly from generate and propagate expressions. It combines these signals from additional logic, so that it does not have to wait for the carry to ripple through every bit.

Carry skip adder retains ripple carry blocks. It speeds up a carry only when all bits in a block propagate the

Carry. In that case, a MUX allows the carry to bypass the block. $\rightarrow$ more latency / less area

Name: _____

Roll No: _____

(b) All P_i, G_i become valid at $t=1$.

(i) At bit position 0, P_0, G_0 are valid at $t=1$, while C_0 is available at $t=0$. The carry logic requires one additional time unit. $C_1: t=2$. The carry then ripples through bits 1, 2, 3. $C_2: t=3$, $C_3: t=4$, $C_4: t=5$

(ii) The group propagate signal $G_{P_{4:7}}$ gets computed one time unit after the relevant P_i signals become valid. Since all P_i signals become valid at $t=1$, $G_{P_{4:7}}$ becomes valid at $t=2$.

The selected data input C_4 becomes valid at $t=5$. The bypass MUX output occurs 2 time units after both the select signal and selected data are valid.

$$C_8: \max(2, 5) + 2 = 7.$$

Similarly $G_{P_{8:11}}$ is already valid at $t=2$. The next bypass MUX receives C_8 at $t=7$, so

$$C_{12}: \max(2, 7) + 2 = 9$$

(iii) Starting with C_{12} at $t=9$, the carry propagates through bits 12, 13, 14. $C_{13}: t=10$, $C_{14}: t=11$, $C_{15}: t=12$. The sum output $S_{15} = P_{15} \oplus C_{15}$. P_{15} is valid at $t=1$, C_{15} becomes valid at $t=12$. Sum logic takes 2 more time

units. $\boxed{S_{15} : 12 + 2 = 14}$

Name: _____

Roll No: _____

(c) Let the n bit adder have m blocks, each with b bits.

$$n = mb.$$

Delay in terms of b and m .

First block $P_i G_i$ signals take 1 time unit. Producing carry through b bit positions takes b time units. Carry at output of first block valid at $\boxed{1 + b}$

Intermediate bypass multiplexers There are $(m-2)$ intermediate blocks and hence $(m-2)$ bypass multiplexers. Each MUX contributes 2 time units. Their total delay is $2(m-2)$. Group propagate signals are ready early at $t = 2$. Therefore they do not add further delay on this path, once the carry reaches the MUXes.

Final block The carry reaches the final block at the least significant bit. To produce the most significant sum, it must propagate through the first $b-1$ bit positions of the final block, taking $(b-1)$ time. Sum logic takes 2 more time units. Final block delay is $b-1 + 2 = b+1$

$$\begin{aligned} T = \text{Total path delay} &= (b+1) + 2(m-2) + (b+1) \\ &= 2b + 2m - 2 = \boxed{2b + 2\frac{n}{b} - 2} \end{aligned}$$

$\boxed{n=32} \quad b=4 \quad T = 8 + \frac{64}{4} - 2 = 22$

$$b=8 \quad T = 16 + \frac{64}{8} - 2 = 22$$

Name: _____

Roll No: _____

4. **Binary representation and encoding: A five-bit encoding of four commands**

[Total: 20 marks]

A communication system must transmit one of four commands,

$$\{\text{IDLE, READ, WRITE, STOP}\}.$$

Each command is represented by a five-bit binary codeword. During transmission, at most one bit may change: a 0 may become 1, or a 1 may become 0.

Consider the binary matrix

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

For a five-bit word $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5)$, the product $H\mathbf{x}^T$ is calculated using ordinary matrix multiplication, except that all additions are evaluated modulo 2. Thus, $0 + 0 = 0$, $0 + 1 = 1$, and $1 + 1 = 0 \pmod{2}$. Addition modulo 2 is equivalent to XOR and is denoted by $\oplus$. The matrix H is called a *parity-check matrix*. A five-bit word $\mathbf{x}$ is a *valid codeword* if $H\mathbf{x}^T = \mathbf{0}$, where $\mathbf{0}$ denotes the three-bit all-zero column vector.

- (a) Expand $H\mathbf{x}^T = \mathbf{0}$ and write the three XOR conditions that $x_1, x_2, \dots, x_5$ must satisfy for $\mathbf{x}$ to be a valid codeword. [2 marks]

- (b) If a valid codeword $\mathbf{x}$ is transmitted and bits change during transmission, the received word can be written as $\mathbf{r} = \mathbf{x} \oplus \mathbf{e}$. The *error vector* $\mathbf{e}$ contains 1 in every position where a bit changed and 0 elsewhere. The *syndrome* of a received word is $\mathbf{s} = H\mathbf{r}^T$. You may use

$$H(\mathbf{a} \oplus \mathbf{b})^T = H\mathbf{a}^T \oplus H\mathbf{b}^T.$$

- i. Suppose $\mathbf{x}$ is any valid codeword and $\mathbf{r} = \mathbf{x} \oplus \mathbf{e}$. Using the definition of a valid codeword, show that $\mathbf{s} = H\mathbf{e}^T$. [2 marks]
 - ii. Let $\mathbf{e}_i$ contain 1 in position i and 0 elsewhere. Using the result of part (i), show that a single-bit error in position i produces syndrome H_i , where H_i denotes column i of H . [2 marks]
 - iii. The received word is $\mathbf{r} = 10111$. It is known that a valid codeword was transmitted and that at most one bit changed. Calculate the syndrome $H\mathbf{r}^T$, use it to identify and correct the erroneous bit, and state the transmitted codeword. [2 marks]
- (c) In H , is it necessary for every column to be nonzero if every single-bit error must be detected? Is it necessary for the columns to be distinct if the position of every single-bit error must be identified uniquely? Verify that the given H satisfies both properties, and explain whether these properties are together sufficient to identify and correct any single-bit error. [6 marks]
- (d) Let $\mathbf{c}_1$ and $\mathbf{c}_2$ be two distinct valid codewords, and define their *difference vector* as $\mathbf{z} = \mathbf{c}_1 \oplus \mathbf{c}_2$. Thus, $\mathbf{z}$ contains 1 exactly in those positions where the two codewords differ. Using the fact that both codewords are valid, show that $H\mathbf{z}^T = \mathbf{0}$. Then use the column properties established in part(c), to explain why $\mathbf{z}$ cannot have exactly one entry equal to 1 or exactly two entries equal to 1. [4 marks]
- (e) Can a four-bit encoding of four commands uniquely correct every possible single-bit error? (*Hint*: Define the *one-error set* of a codeword as the set containing the codeword itself and all words obtained by changing exactly one of its bits. How many words are in the one-error set of a four-bit codeword?) [2 marks]

$$(a) \quad H\mathbf{x}^T = \mathbf{0} \quad H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} x_4 \oplus x_5 \\ x_2 \oplus x_3 \\ x_1 \oplus x_3 \oplus x_5 \end{bmatrix}$$

$$x_4 \oplus x_5 = 0$$

$$x_2 \oplus x_3 = 0$$

$$x_1 \oplus x_3 \oplus x_5 = 0$$

(b)(i) Received codeword is $r = x \oplus e$. Receiver calculates syndrome $s = Hr^T$

$$= H(x \oplus e)^T$$

$$= Hx^T \oplus He^T$$

Since x is a valid keyword $Hx^T = 0 \Rightarrow s = He^T$.

b(ii) For a single bit error in position i , $e = e_i$, where e_i contains a 1 in position i and 0 elsewhere. From part(i) $s = He^T = He_i^T$. Multiplying a matrix by e_i^T selects the i th column of the matrix. Therefore $s = H_i$. Thus the syndrome of a single bit error at position i , is exactly column i of H .

b(iii) $x = (1, 0, 1, 1, 1)$

syndrome $s = Hr^T = \begin{bmatrix} r_4 \oplus r_5 \\ r_2 \oplus r_3 \\ r_1 \oplus r_3 \oplus r_5 \end{bmatrix}$

$$= \begin{bmatrix} 1 \oplus 1 \\ 0 \oplus 1 \\ 1 \oplus 1 \oplus 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = H_3$$

Therefore error occurred in bit 3. Corrected (10011)

(c) Yes, every column must be non zero if every single bit error is to be detected. A single bit error at position i has syndrome $s = H_i$. If $H_i = 0$, then an error in position i produces the 0 syndrome, which is indistinguishable from the syndrome of no error. Therefore that error will not be detected.

Yes, the columns must be distinct if the position of every single bit error is to be identified uniquely. If $H_i = H_j$ for $i \neq j$, then an error in position i and an error in position j produce the same syndrome. The receiver would know an error has occurred (non zero syndrome) but will not know the position.

Columns of H are 001, 010, 011, 100, 101 - all non zero and distinct.

Under the stated assumption that at most 1 bit changes $\rightarrow$ syndrome 000 means no error, every possible single bit error produces a non-zero syndrome. As columns are distinct, syndrome identifies exactly

one bit position. The given H can detect and correct every single bit error.

Name: _____

Roll No: _____

(d) c_1 and c_2 are valid codewords. $z = c_1 \oplus c_2$. As both codewords are valid, $Hc_1^T = 0$ and $Hc_2^T = 0$. Using linearity $H_z^T = H(c_1 \oplus c_2)^T = Hc_1^T \oplus Hc_2^T = 0 \oplus 0 = 0$
 $\Rightarrow \boxed{H_z^T = 0}$

Suppose z has a 1 only in position i . Then $z = e_i$. $H_z^T = H_i$, But each H_i is nonzero, contradicts $H_z^T = 0$. Therefore z cannot have weight 1.

Suppose z has 1s in positions i and j . Then $H_z^T = H_i \oplus H_j$. For this to be 0, H_i has to be equal to H_j , which is false. Therefore z cannot have weight 2.

(e) For a 4 bit codeword, its one error set contains the original codeword itself and four words obtained by flipping exactly one of its 4 bits. Thus each error set contains $1 + 4 = 5$ four bit words.

To uniquely correct every single bit error, the one-error sets associated with any correct word need to be disjoint. Else same received word could have arisen from two different

transmitted words.

With 4 commands, required number of distinct received words $= 4 \times 5 = 20$

But there are only $2^4 = 16$ words with 4 bits.

As $20 > 16$, a 4-bit encoding cannot uniquely correct every single bit error.