Problem 1: Karnaugh Maps

You'll analyze a 5-variable Boolean function using Karnaugh maps.

Part (a): Generate the Function

Compute a unique 5-variable Boolean function as follows:

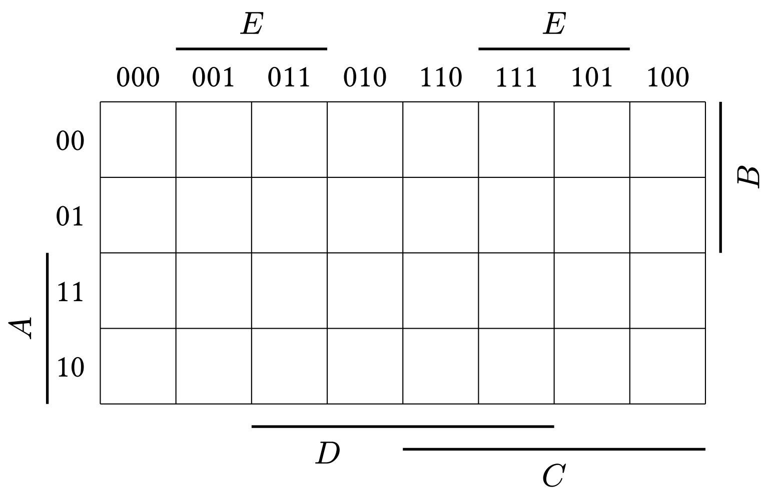
1. Hash the string `"DM Fall 2025 HW3"` (UTF-8, no quotes) using SHA-256 to get 256 bits.
2. Split the result into eight 32-bit blocks and XOR them together to get 32 bits.
3. XOR with the mask `0x71be8976` to obtain $w = (w_1 w_2 \cdots w_{32})_2$.

This 32-bit value w encodes the truth table of your function $f(A, B, C, D, E)$, where bit w_1 (MSB) gives $f(0, 0, 0, 0, 0)$ and bit w_{32} (LSB) gives $f(1, 1, 1, 1, 1)$.

Check: The hash ends with …`00010101`, and after XORing the blocks, the result starts with `0110`…

Part (b): Draw the K-Map

Construct a 5-variable Karnaugh map for your function using the template below:



Part (c): Find Minimal Forms

Using your K-map:

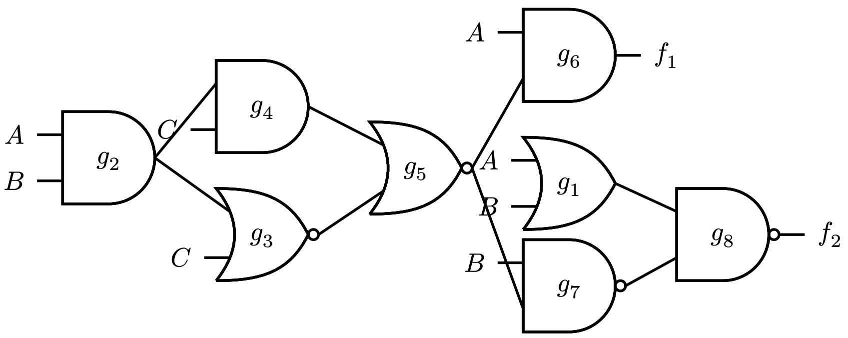
1. Find the minimal DNF for f.
2. Find the minimal CNF for f.
3. Count all prime implicants.
4. Suppose inputs where $A \wedge B \wedge C = 1$ are *don't-care* conditions.
   How does this affect your minimal DNF and CNF?

Part (d): Reflection

1. How much smaller is your minimal DNF compared to listing all minterms?
2. Compare the sizes of your minimal DNF vs minimal CNF. Which is smaller?
3. Why do K-maps become impractical beyond 5-6 variables?

Problem 2: Circuit Analysis

Given a combinational circuit with 3 inputs (A, B, C) and 2 outputs (f_1, f_2):



1. Compute the truth table for $f : \mathbb{B}^3 \to \mathbb{B}^2$ where $\langle A, B, C\rangle \mapsto \langle f_1, f_2\rangle$.
2. Express f_1 and f_2 in minimal DNF.
3. Identify redundant gates and simplify the circuit if possible.
4. What is the maximum propagation delay (in gate delays)?
5. Analyze the critical path: Which sequence of gates determines the overall delay?

Problem 3: Boolean Function Analysis

Analyze the following 4-variable functions using multiple representation methods.[1]

1. $f_1 = f^{(4)}_{47541}$
2. $f_2 = \sum m(1, 4, 5, 6, 8, 12, 13)$
3. $f_3 = f^{(4)}_{51011} \oplus f^{(4)}_{40389}$
4. $f_4 = A\overline{B}D + \overline{A}\,\overline{C}D + \overline{B}C\overline{D} + A\overline{C}D$

For each function:

- Draw the K-map.
- Find all prime implicants (Blake Canonical Form).
- Find the minimal DNF and minimal CNF.
- Construct the ANF using the K-map method, tabular method, or Pascal's triangle method. Use each ANF construction method at least once.

Note: WolframAlpha reverses the bit order for "n-th Boolean function of k variables". For $f^{(2)}_{10} = (1010)$, query "5th Boolean function of 2 vars" since $\mathrm{rev}(1010_2) = 0101_2 = 5$.

Problem 4: CNF Conversion

Converting Boolean formulae to CNF is fundamental for SAT solving and automated reasoning.

Part (a): Basic Conversions

Convert each formula to CNF:

1. $X \leftrightarrow (A \land B)$
2. $Z \leftrightarrow \bigvee_i C_i$
3. $D_1 \oplus D_2 \oplus \cdots \oplus D_n$
4. $\mathrm{majority}(X_1, X_2, X_3)$[2]
5. $R \to \left(S \to \left(T \to \bigwedge_i F_i\right)\right)$
6. $M \to \left(H \leftrightarrow \bigvee_i D_i\right)$

[1] Notation: $f^{(n)}_k$ is the k-th Boolean function of n variables, where k is the decimal value of the truth table with MSB = $f(0, \ldots, 0)$ and LSB = $f(1, \ldots, 1)$. For example, $f^{(2)}_{11} = (1011)$.

[2] Majority function returns 1 iff more than half of its inputs are 1.

Part (b): Tseitin Transformation

The Tseitin transformation converts any formula to equisatisfiable CNF by introducing auxiliary variables for subformulae.

1. Apply Tseitin to: $(A \vee B) \wedge (C \vee (D \wedge E))$
2. Prove equisatisfiability of your CNF with the original.
3. Compare the size (clauses and variables) with direct CNF conversion.

Part (c): Resource Allocation

A system has five resources $\{R_1, R_2, R_3, R_4, R_5\}$ and must satisfy:
- Either R_1 or both R_2 and R_3
- If R_1, then R_4 or R_5
- Not both R_2 and R_4
- At least two of $\{R_1, R_2, R_3\}$

1. Encode these constraints as Boolean formulae.
2. Convert to CNF.
3. Find all satisfying assignments.

Part (d): Optimization Analysis

1. For the formula in Part (b), compute the exact clause count for both Tseitin and direct CNF.
2. Explain the trade-off: Tseitin uses more variables but fewer/shorter clauses. When is each approach preferable?
3. For a formula of size n (number of connectives), what is the worst-case clause count for direct CNF vs Tseitin?

Problem 5: Functional Completeness

A set of Boolean functions is *functionally complete* if it can express any Boolean function.

Part (a): Apply Post's Criterion

Determine whether each system is functionally complete using Post's criterion:

1. $F_1 = \{\wedge, \vee, \neg\}$
2. $F_2 = \{f_{14}^{(2)}\}$
3. $F_3 = \{\rightarrow, \nrightarrow\}$[3]
4. $F_4 = \{1, \leftrightarrow, \wedge\}$

Part (b): Express Majority

For each complete basis from Part (a):

1. Express majority(A, B, C) using only that basis.
2. Draw the corresponding circuit.
3. Count the gates.

[3] $(A \nrightarrow B) \equiv (A \wedge \neg B)$

Problem 6: Zhegalkin Polynomials

The Zhegalkin basis $\{\oplus, \wedge, 1\}$ provides an algebraic view of Boolean functions over the field $\mathbb{F}_2$.

Part (a): Prove Completeness

Prove that $\{\oplus, \wedge, 1\}$ is functionally complete *without* using Post's criterion.

1. Express $\neg x$ using $\{\oplus, 1\}$.
2. Express $x \vee y$ using $\{\wedge, \oplus, 1\}$.
3. Explain why this establishes completeness.

Part (b): Polynomial Degree

The *degree* of a Zhegalkin polynomial is the size of the largest monomial.

1. Apply *Shannon decomposition* to $f(x_1, x_2, x_3) = x_1 x_2 \vee x_2 x_3 \vee x_1 x_3$ with respect to x_1. Verify: $f = x_1 f_{x_1=1} \oplus \overline{x}_1 f_{x_1=0}$ where $\overline{x}_1 = 1 \oplus x_1$.
2. Show how Shannon decomposition can be used to derive the ANF recursively.
3. Prove that every Boolean function has a unique Zhegalkin representation.
4. Find the Zhegalkin polynomial of $\text{majority}(x_1, x_2, x_3)$ using Shannon decomposition.

Part (c): Algebraic Degree in Cryptography

High algebraic degree is important for cryptographic security.

1. An S-box function $S : \mathbb{B}^3 \to \mathbb{B}$ has truth table $(0, 1, 1, 0, 1, 0, 0, 1)$. Find its ANF and degree.
2. Why do cryptographers prefer functions with high algebraic degree?
3. Construct a balanced 3-variable function of degree 3 with no linear terms.[4]

Problem 7: Gray Code Circuits

Gray code encodes integers so consecutive values differ in exactly one bit.

Example:
- $\texttt{0000}_2 \to \texttt{0000}_{\text{Gray}}$
- $\texttt{1001}_2 \to \texttt{1101}_{\text{Gray}}$
- $\texttt{1111}_2 \to \texttt{1000}_{\text{Gray}}$

Part (a): Truth Table

1. Build the complete 4-bit binary-to-Gray truth table.
2. Verify that consecutive binary numbers map to Gray codes differing by one bit.

Part (b): Circuit Design

1. Find minimal Boolean expressions for each g_i.
2. Design a binary-to-Gray circuit using only NAND and/or NOR gates.
3. Count the gates.

Part (c): Reverse Conversion

1. Derive the Gray-to-binary conversion formula.

[4] A function is *balanced* if it outputs 0 and 1 equally often.

2. Prove that composing "binary $\rightarrow$ Gray $\rightarrow$ binary" yields the identity function.
3. Design a Gray-to-binary circuit using only NAND gates.
4. Compare circuit complexity for both directions.

Problem 8: Arithmetic Circuits

Build fundamental arithmetic building blocks and combine them into more complex circuits.

Part (a): Half Subtractor

1. Derive Boolean expressions for the difference d and borrow b outputs of a half subtractor.
2. Construct the circuit using AND, OR, and NOT gates.
3. Verify with a truth table.

Part (b): Full Subtractor

1. Build a full subtractor using two half subtractors and NAND gates.
2. Draw the circuit.
3. Calculate propagation delay.

Part (c): Saturating Subtractor

Design a 4-bit saturating subtractor that computes $d = \max(0, x - y)$:

- If $x \geq y$: output $d = x - y$
- If $x < y$: output `0000` (saturate to zero)

1. Design the circuit using subtractors and basic gates.
2. Explain how your circuit detects $x < y$.
3. Test with: $5 - 3$, $3 - 5$, and $15 - 8$.

Part (d): 2-bit Comparator

Design a circuit comparing 2-bit integers $(x_1 x_0)_2$ and $(y_1 y_0)_2$.

1. Derive Boolean expressions for three outputs: $x > y$, $x = y$, and $x < y$.
2. Build the circuit using AND, OR, NOT gates.
3. Verify with test cases: $(3, 2)$, $(2, 2)$, $(1, 3)$.

Part (e): 2-bit Multiplier

Design a circuit computing $p = x \cdot y$ for 2-bit integers, giving a 4-bit result $(p_3 p_2 p_1 p_0)_2$.

1. Create the truth table.
2. Find minimal expressions for each p_i.
3. Draw the circuit.
4. Verify: $3 \times 2 = 6$, $3 \times 3 = 9$, $1 \times 1 = 1$.

Part (f): Analysis and Optimization

1. For the multiplier, identify any shared sub-expressions to reduce gate count.
2. Design a circuit that detects overflow in 2-bit addition (when sum requires more than 2 bits).
3. Compare the gate count of your multiplier with a repeated-addition approach.

Problem 9: Conditional Logic and BDDs

Part (a): If-Then-Else Function

The ternary function ITE : $\mathbb{B}^3 \to \mathbb{B}$ is defined as:

$$\text{ITE}(c, x, y) = \begin{cases} x \text{ if } c = 0 \\ y \text{ if } c = 1 \end{cases}$$

This is the Boolean equivalent of `c ? y : x`.

1. Express $\text{ITE}(c, x, y)$ using $\{\land, \lor, \neg\}$.
2. Is $\{\text{ITE}\}$ functionally complete? Identify which Post classes it belongs to.
3. Express $x \oplus y$ using only ITE.

Part (b): Binary Decision Diagrams

Construct a Reduced Ordered BDD (ROBDD) for each function using natural variable order $x_1 \prec x_2 \prec \cdots$:

1. $f_1(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$
2. $f_2(x_1, ..., x_5) = \text{majority}(x_1, ..., x_5)$
3. $f_3(x_1, ..., x_4) = \sum m(1, 2, 5, 12, 15)$
4. $f_4(x_1, ..., x_6) = x_1x_4 + x_2x_5 + x_3x_6$

Part (c): Variable Ordering

Find the variable ordering minimizing ROBDD size for each function in Part (b).

Part (d): Variable Ordering Impact

1. For $f_4(x_1, ..., x_6) = x_1x_4 + x_2x_5 + x_3x_6$, construct the ROBDD with natural ordering: $x_1 \prec x_2 \prec x_3 \prec x_4 \prec x_5 \prec x_6$.
2. Construct the ROBDD with interleaving ordering: $x_1 \prec x_4 \prec x_2 \prec x_5 \prec x_3 \prec x_6$.
3. Compare the BDD sizes. Prove that the second ordering yields a smaller BDD by counting nodes.
4. Explain why variable ordering matters more for some functions than others.

Submission Guidelines:
- Organize solutions clearly with problem numbers and parts.
- For circuit designs: Draw clear diagrams with labeled gates and signals.
- For proofs: State assumptions, show logical steps, and clearly mark conclusions.
- For truth tables: Use standard binary ordering (000, 001, 010, ..., 111).
- For K-maps: Show groupings clearly and indicate which groups form the minimal form.

Grading Rubric:
- Correctness of circuits and Boolean expressions: 40%
- Mathematical rigor and proof quality: 30%
- Clarity of diagrams and presentation: 20%
- Completeness and attention to detail: 10%