Hw3a Lab Report Ongoing
Author: Ryan Oates

[Notation Key]

GateSymbolNotation = *[List of logic gate symbols used in this design]*
e.g., AND ($\wedge$), OR ($\vee$), NOT ($\neg$), etc., to represent gate operations in Boolean expressions.

WireConnectionSymbols = *[Notation for wires and connections in circuit diagrams]*
e.g., lines for connections, a dot for junction, labels/arrows for inputs and outputs.

LogicLevelsRepresentation = *[Definition of logic high and low levels]*
e.g., use '1' (HIGH/True) and '0' (LOW/False) to denote binary logic levels.

BooleanExpressionNotation = *[Format for writing Boolean formulas]*
e.g., use '+' for OR, $\cdot$ for AND, overline or '¡ for NOT ($A + B$ means $A \vee B$, AB or $A \cdot B$ means $A \wedge B$, $\neg A$ means NOT A).
Note: These notations are used throughout (in the truth table, K-map, expressions, and diagrams). See [**Glossary**] for definitions of terms and symbols.

Verification:
✓/× `verify_symbols_defined`: All necessary gate symbols and wire notations are defined above. (Comment: _____)
✓/× `verify_logic_levels`: Logic 0/1 representation is clear and used consistently. (Comment: _____)
✓/× `verify_notation_consistency`: Boolean expression notation is consistent across all sections. (Comment: _____)

[Possible Inputs]

InputVariables = *[List of input variable names]*
e.g., A, B, C (representing all independent inputs to the logic circuit).

InputDomain = *[All possible values or conditions for each input]*
e.g., each input $\in \{0, 1\}$ for binary logic (enumerate any constraints or don't-care conditions if applicable).
Note: These inputs define the domain for the [**Truth Table**].

Verification:
✓/× `verify_all_inputs_listed`: All input variables are identified and named. (Comment: _____)
✓/× `verify_domain_specified`: The range of input values (e.g., binary 0/1) is specified. (Comment: _____)

[Truth Table]

TruthTableInputs = *[Repeat input variables as column headers]*
TruthTableOutputs = *[Output variable names as columns]*
TruthTableFormat = *[Notation for table entries]*
e.g., 0/1 from **LogicLevelsRepresentation**.

An example for two inputs (A, B) and outputs (Sum, Carry):

A	B	Sum	Carry
0	0	0	1
0	1	1	1
1	0	1	1
1	1	0	0

Verification:
✓/× `verify_all_combinations`: Table includes all possible input combos (no missing rows). (Comment: _____)
✓/× `verify_output_values`: Output values reflect the intended logic. (Comment: _____)
✓/× `verify_table_format`: Table is formatted with headers and matches defined logic levels. (Comment: _____)

[K-Map Reduction]

KMapVariables: *[List the same input variables here]*
KMapLayout: *[2x2, 4x4, etc. with Gray code ordering]*
KMapFilling: *[Placement of 1s/0s based on the Truth Table]*
KMapGrouping: *[Groups of adjacent 1s to simplify expression]*
KMapSimplificationSteps: *[Step-by-step Boolean simplification]*
KMapResultExpression: *[Final simplified Boolean expression from the K-map]*

Example layout:

	$B = 0$	$B = 1$
$A = 0$	1	1
$A = 1$	1	0

Groups might yield $A'B' + A'B + AB'$.

Verification:
✓/× `verify_grouping_complete`: All 1-cells are grouped (possibly with powers-of-2). (Comment: _____)
✓/× `verify_minimal_expression`: Expression is minimal (no more reduction). (Comment: _____)
✓/× `verify_expression_correct`: K-map result matches original Truth Table. (Comment: _____)

(See also [SOP Reduction] and [Boolean Expression Reduction] to confirm equivalence.)

[Gate Operations]

Define the logic gates that implement the K-map result or other derived expression:

- **Gate1** = [Gate type, inputs → output]
  - e.g., $G1$: AND gate with inputs A and B, output X ($X = A \wedge B$).
- **Gate2** = [Gate type, inputs → output]
  - e.g., $G2$: NOT gate with input C, output Y ($Y = \neg C$).
- **Gate3** = [Gate type, inputs → output]
  - e.g., $G3$: OR gate with inputs X and Y, output F ($F = X + Y$).

GateOperationNotes: e.g., timing assumptions, gate delays, referencing **Notation Key** symbols, etc.

Verification:
✓/× `verify_gates_complete`: All parts of the simplified Boolean expression are present. (Comment: _____)
✓/× `verify_output_consistency`: Gate-level outputs match the expected Truth Table. (Comment: _____)
✓/× `verify_notation_adherence`: Gate symbols follow [**Notation Key**]. (Comment: _____)
✓/× `verify_timing_addressed`: Propagation delay or clocking details are noted if needed. (Comment: _____)

[Circuit Diagram Representation]

DiagramIllustration = *[Insert or describe the circuit diagram (ASCII or figure)]*
e.g.,

$$\begin{array}{l} A \longrightarrow [\text{NAND1}] \longrightarrow [\text{NAND3}] \longrightarrow [\text{OR}] \longrightarrow \text{Sum} \\ \qquad\quad \downarrow \\ B \longrightarrow [\text{NAND2}] \longrightarrow \ldots \\ \quad \ldots \end{array}$$

DiagramNotation = *[Gate shapes, lines, junctions, etc. per the Notation Key]*
DiagramLabels = *[Labels for inputs (A,B), outputs (Sum,Carry), and intermediate signals]*

Verification:
✓/× `verify_diagram_accuracy`: Matches gate-level design from [**Gate Operations**]. (Comment: _____)
✓/× `verify_label_consistency`: All signals in the diagram appear in the text/tables. (Comment: _____)
✓/× `verify_symbol_standard`: Symbols align with [**Notation Key**]. (Comment: _____)

[Full Signal Analysis]

Analyze the intermediate signals (nodes) and propagation:

- **NodeEquations**: e.g., $X = A \cdot B$, $Y = \neg C$, etc.
- **OutputEquation**: e.g., $F = X + Y$ or expanded in terms of inputs.
- **PropagationSteps**: Describe how signals flow from inputs to outputs.
- **TimingAnalysis**: If relevant, note worst-case delay (critical path).

Example:
`Node1 = ¬(A · B),  Node2 = ¬(A · B),  Node3 = ¬(Node1 · Node2),  ...`

Verification:
✓/× `verify_node_logic`: Intermediate node equations match the final expression. (Comment: _____)
✓/× `verify_propagation_correctness`: Tracking signals from input changes yields the correct final outputs. (Comment: _____)
✓/× `verify_timing_consistency`: Any delay or glitch concerns are addressed. (Comment: _____)

[SOP Reduction]

InitialSOP: e.g., list of minterms from the Truth Table.
ReductionSteps: Show step-by-step Boolean algebra.
ReducedSOP: Final minimized Sum-of-Products expression.

For example:

$$F = A\,B\,\neg C + A\,B\,C \quad \rightarrow \quad F = A\,B(\neg C + C) \quad \rightarrow \quad F = A\,B.$$

Verification:
✓/× `verify_all_minterms_used`: No missing or extra minterms. (Comment: _____)
✓/× `verify_algebraic_steps`: Each reduction step is valid. (Comment: _____)
✓/× `verify_sop_matches_kmap`: The final SOP equals the K-map simplified expression. (Comment: _____)

[Boolean Expression Reduction]

InitialExpression = [Full unsimplified or starting expression]
TargetExpression = [Intended simplified expression]
ReductionTechnique = [Method: algebraic, De Morgan's, consensus, etc.]
ReductionProof = [Verification that initial = final (truth table or symbolic)]
FinalBooleanExpression = [Form actually used in design]

Verification:
✓/× `verify_equivalence_proven`: Confirm the simplified expression is functionally identical. (Comment: _____)
✓/× `verify_expression_minimal`: No further reduction possible. (Comment: _____)
✓/× `verify_consistency_final`: Matches the implemented circuit and [**Truth Table**]. (Comment: _____)

[Direct Links]

- **NotationKey_related**: Used by all sections for consistent symbols.
- **PossibleInputs_related**: Feeds **Truth Table**.
- **TruthTable_related**: Basis for **K-Map Reduction** and **SOP Reduction**.
- **KMapReduction_related**: Simplifies the truth table outputs to minimal expression.
- **GateOperations_related**: Implements the simplified expression with physical gates.
- **CircuitDiagram_related**: Visual representation of Gate Operations and wiring.
- **FullSignalAnalysis_related**: Verifies dynamic signal flow.
- **SOPReduction_related**: Alternative method to match K-map result.
- **BooleanExprReduction_related**: Final check that everything is equivalent.
- **Glossary_related**: Definitions of terms and symbols.

[Glossary]

AND Gate = Logic gate that outputs 1 only if **all** its inputs are 1. (symbol: $\wedge$)
OR Gate = Logic gate that outputs 1 if **at least one** input is 1. (symbol: $\vee$)
NOT Gate = Logic gate that outputs the negation of its input. (symbol: $\neg$)
Logic 1 (HIGH) = True state or high voltage level.
Logic 0 (LOW) = False state or low voltage level.
Truth Table = A table enumerating all input combinations vs. output.
Karnaugh Map (K-Map) = Graphical method for simplifying Boolean expressions.
Sum-of-Products (SOP) = Boolean form with OR of AND terms (e.g., $AB + \neg A\,C$).
Boolean Expression = Algebraic form using logic operators.
Node (Intermediate) = Output of a gate not the final output (internal signal).
Propagation Delay = Time for signal change to travel gate to gate.
Verification Checklist = Steps to confirm correctness.
Cross-Referencing = Linking related sections or terms in the document.
✓ = Successful verification or pass.
× = Failed verification or requires correction.

[Simulation and Results]

SimulationTool = e.g., Tina, ModelSim, or Logisim.
TestDuration = e.g., 100 ns.
Stimulus = e.g., $A : 0 \to 1$ at 20 ns; $B : 0 \to 1$ at 40 ns.

WaveformConfiguration: *Colors, trace widths, background, etc.*
WaveformAnalysis: *Tabulate times vs. A, B, sum, carry, plus commentary.*
For instance:

Time	A	B	Sum	Carry	Analysis
$0-20$ ns	0	0	0	0	Initial
$20-40$ ns	1	0	1	0	A transitions
$40-60$ ns	1	1	0	1	B transitions, carry = 1
$\vdots$					

[Design History / Iterations]

Version 1: Basic design with potential redundancy
Version 2: Removed redundancy
Version 3: Corrected carry logic
Version 4: Final validated design

[Verification Checklist]

✓ `truth_table_verified`
✓ `timing_requirements_met`
✓ `carry_generation_correct`
✓ `waveform_visualization_optimized`