

Exercise walk-through

4.3

Binary Applications

GAC Mathematics

1

The method

Method — Key knowledge: logical truth and bit patterns need explicit conventions

Boolean algebra 布尔代数 uses true and false with AND, OR and NOT. Here OR is inclusive: true when either or both inputs are true. A **truth table** 真值表 evaluates every input combination; a two-input table has four rows. **Logic gates** 逻辑门 implement stated operations in a circuit.

Use brackets to make expression grouping explicit. De Morgan's identities are $\neg(A \wedge B) = (\neg A) \vee (\neg B)$ and $\neg(A \vee B) = (\neg A) \wedge (\neg B)$. Bitwise operations apply separately to each matching bit position. This sheet's bit patterns are unsigned unless stated otherwise, and NOT in a pattern inverts only the specified width.

Method — Worked example: an OR output is inverted by NOT

Known: $Y = \neg(A \vee B)$. Inclusive OR is false only when both inputs are false; NOT reverses that result. In row order $(A, B) = (0, 0), (0, 1), (1, 0), (1, 1)$, the output column is 1, 0, 0, 0. De Morgan gives equivalent expression $(\neg A) \wedge (\neg B)$.



Exercises 2.1-2.3

2.1 For A true and B false, find A AND B and A OR B, using inclusive OR. [2]

2.2 Invert the four-bit pattern 1010 using bitwise NOT, and find its unsigned decimal value. [2]

2.3 List the outputs of $A \wedge \neg B$ in row order (0,0), (0,1), (1,0), (1,1). State which input condition makes the output true. [3]

Exercises 2.1-2.3

2.1 For A true and B false, find A AND B and A OR B, using inclusive OR. [2]

2.1

A AND B is false because both inputs are not true. A OR B is true because at least one input is true.

2.2 Invert the four-bit pattern 1010 using bitwise NOT, and find its unsigned decimal value. [2]

2.2

The inverted four-bit pattern is 0101. Unsigned value is 5. The width remains four bits.

2.3 List the outputs of $A \wedge \neg B$ in row order (0,0), (0,1), (1,0), (1,1). State which input condition makes the output true. [3]

2.3

In order the outputs are 0, 0, 1, 0. Applying NOT before AND earns M1; correct output column earns A1. Output is true exactly when A is true and B false.

Exercises 2.4·3.1

2.4 Use De Morgan to rewrite $\neg(A \wedge B)$. Evaluate both forms at A true and B true. [3]

3 Exam-style questions

3.1 For four-bit patterns $X = 1100$ and $Y = 1010$, calculate (a) bitwise AND, (b) bitwise inclusive OR, and (c) bitwise NOT of X. Give each result also as an unsigned decimal value. [6]

Exercises 2.4·3.1

2.4 Use De Morgan to rewrite $\neg(A \wedge B)$. Evaluate both forms at A true and B true. [3]

3 Exam-style questions

2.4

Equivalent form $(\neg A) \vee (\neg B)$. Original form is NOT true, hence false. Rewritten form is false OR false, also false. A single row check alone is not a proof of full equivalence.

3.1 For four-bit patterns $X = 1100$ and $Y = 1010$, calculate (a) bitwise AND, (b) bitwise inclusive OR, and (c) bitwise NOT of X. Give each result also as an unsigned decimal value. [6]

3.1

AND gives 1000. Decimal 8. OR gives 1110. Decimal 14. Four-bit NOT of X gives 0011. Decimal 3. NOT does not create an unrestricted or signed value here.

Exercise 3.2

3.2 A simplified access rule is $Y = (A \wedge B) \vee (A \wedge \neg B)$, where A and B are Boolean inputs. (a) Construct its complete truth table with intermediate columns. (b) Simplify the rule and justify equivalence. (c) Explain whether this mathematical simplification establishes that a real access system is secure. [8]

Exercise 3.2

3.2 A simplified access rule is $Y = (A \wedge B) \vee (A \wedge \neg B)$, where A and B are Boolean inputs. (a) Construct its complete truth table with intermediate columns. (b) Simplify the rule and justify equivalence. (c) Explain whether this mathematical simplification establishes that a real access system is secure. [8]

3.2

(a) Use row order 00, 01, 10, 11. Column NOT B is 1, 0, 1, 0. Column A AND B is 0, 0, 0, 1. Column A AND NOT B is 0, 0, 1, 0. Final Y is 0, 0, 1, 1.

(b) Factor A using distributivity.

$$Y = A \wedge (B \vee \neg B)$$

The bracket is always true, so $Y = A$. This agrees with the A column in every truth-table row, proving Boolean equivalence.

(c) The truth table tests this abstract rule only; actual authentication, implementation and policy are not established by its simplification.