Computer Organization

COMP2120

Qi Zhao

January 17, 2024

Digital Logic

Boolean Algebra

- Proposed by George Boole in 1854
- Variables: the values of the variables are the truth values true (1) and false (0)
- A Boolean function with k input variables

$$f : \{0, 1\}^k \longrightarrow \{0, 1\} \tag{1}$$

- Operations: **AND** $(A \cdot B)$, **OR** $(A + B)$, **NOT** $(\overline{A})$

Boolean Algebra

- Proposed by George Boole in 1854
- Variables: the values of the variables are the truth values true (1) and false (0)
- A Boolean function with k input variables

$$f : \{0, 1\}^k \longrightarrow \{0, 1\} \tag{1}$$

- Operations: **AND** $(A \cdot B)$, **OR** $(A + B)$, **NOT** $(\overline{A})$
- Truth table

A	**NOT** A
0	1
1	0

A	B	A **AND** B	A **OR** B
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Boolean Algebra

- **NAND** and **NOR** : the complement (NOT) of AND and OR, $\overline{A \cdot B}, \overline{A + B}$

Boolean Algebra

- **NAND** and **NOR** : the complement (NOT) of AND and OR, $\overline{A \cdot B}, \overline{A + B}$
- **XOR**: exactly one of the operands has the value 1, $A \oplus B$

Boolean Algebra

- **NAND** and **NOR** : the complement (NOT) of AND and OR, $\overline{A \cdot B}$, $\overline{A + B}$
- **XOR**: exactly one of the operands has the value 1, $A \oplus B$
- Boolean Algebra Laws

Boolean Algebra

- **NAND** and **NOR** : the complement (NOT) of AND and OR, $\overline{A \cdot B}, \overline{A + B}$
- **XOR**: exactly one of the operands has the value 1, $A \oplus B$
- Boolean Algebra Laws
    1. Commutative Law

$$AB = BA, \ A + B = B + A$$

Boolean Algebra

- **NAND** and **NOR** : the complement (NOT) of AND and OR, $\overline{A \cdot B}, \overline{A+B}$
- **XOR**: exactly one of the operands has the value 1, $A \oplus B$
- Boolean Algebra Laws
  1. Commutative Law
  $$AB = BA,\ A + B = B + A$$
  2. Identity elements
  $$1 \cdot A = A, 0 + A = A$$

Boolean Algebra

- **NAND** and **NOR** : the complement (NOT) of AND and OR, $\overline{A \cdot B}, \overline{A + B}$
- **XOR**: exactly one of the operands has the value 1, $A \oplus B$
- Boolean Algebra Laws
  1. Commutative Law
  $$AB = BA, \; A + B = B + A$$
  2. Identity elements
  $$1 \cdot A = A, 0 + A = A$$
  3. Null law:
  $$0 \cdot A = 0, 1 + A = 1$$

Boolean Algebra

- **NAND** and **NOR** : the complement (NOT) of AND and OR, $\overline{A \cdot B}, \overline{A + B}$
- **XOR**: exactly one of the operands has the value 1, $A \oplus B$
- Boolean Algebra Laws
  1. Commutative Law
  $$AB = BA,\ A + B = B + A$$
  2. Identity elements
  $$1 \cdot A = A, 0 + A = A$$
  3. Null law:
  $$0 \cdot A = 0, 1 + A = 1$$
  4. Idempotent law:
  $$A \cdot A = A, A + A = A$$

Boolean Algebra

- **NAND** and **NOR** : the complement (NOT) of AND and OR, $\overline{A \cdot B}, \overline{A + B}$
- **XOR**: exactly one of the operands has the value 1, $A \oplus B$
- Boolean Algebra Laws
  1. Commutative Law
$$AB = BA,\ A + B = B + A$$
  2. Identity elements
$$1 \cdot A = A, 0 + A = A$$
  3. Null law:
$$0 \cdot A = 0, 1 + A = 1$$
  4. Idempotent law:
$$A \cdot A = A, A + A = A$$
  5. Inverse law:
$$A \cdot \overline{A} = 0, A + \overline{A} = 1$$

Boolean Algebra

6. Associative law

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C,$$
$$A + (B + C) = (A + B) + C$$

Boolean Algebra

6. Associative law

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C,$$
$$A + (B + C) = (A + B) + C$$

7. Distributive Law

$$A(B + C) = AB + AC$$
$$A + (BC) = (A + B)(A + C)$$

Boolean Algebra

6. Associative law

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C,$$
$$A + (B + C) = (A + B) + C$$

7. Distributive Law

$$A(B + C) = AB + AC$$
$$A + (BC) = (A + B)(A + C)$$

Proof

$$\begin{aligned} RHS &= (A + B)(A + C) \\ &= A \cdot A + A \cdot C + B \cdot A + B \cdot C \end{aligned}$$
$$if\ A = 0,\ RHS = 0 + B \cdot C$$
$$if\ A = 1,\ RHS = 1 + B + C + B \cdot C = 1 + B \cdot C$$

Boolean Algebra

8. DeMorgan's Theorem

$$\overline{A \cdot B} = \overline{A} + \overline{B}, \; \overline{A + B} = \overline{A} \cdot \overline{B}$$

Boolean Algebra

8. DeMorgan's Theorem

$$\overline{A \cdot B} = \overline{A} + \overline{B}, \ \overline{A + B} = \overline{A} \cdot \overline{B}$$

- Represent Boolean function via Venn diagrams

Boolean	Sets
$A \cdot B$	$A \cap B$, intersection
$A + B$	$A \cup B$, union

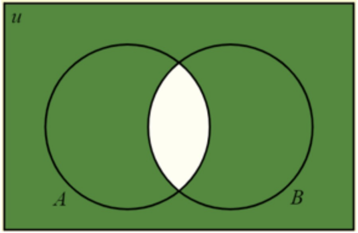
Boolean Algebra

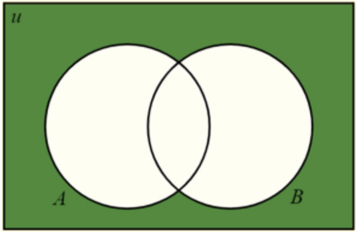
8. DeMorgan's Theorem

$$\overline{A \cdot B} = \overline{A} + \overline{B}, \ \overline{A + B} = \overline{A} \cdot \overline{B}$$

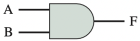
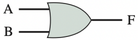
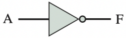
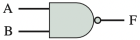
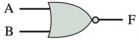
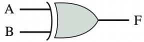
- Represent Boolean function via Venn diagrams

Boolean	Sets
$A \cdot B$	$A \cap B$, intersection
$A + B$	$A \cup B$, union

- 



Gates

Name	Graphical Symbol	Algebraic Function	Truth Table
AND		$F = A \bullet B$ or $F = AB$	A B | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = A + B$	A B | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
NOT		$F = \overline{A}$ or $F = A'$	A | F 0 | 1 1 | 0
NAND		$F = \overline{AB}$	A B | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = \overline{A + B}$	A B | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
XOR		$F = A \oplus B$	A B | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0

Functionally complete sets

- **AND**, **OR**, **NOT**
- **AND**, **NOT**: DeMorgan's theorem

$$A + B = \overline{\overline{A} \cdot \overline{B}}$$

- **OR**, **NOT**, think about it!
- **NAND**
- **NOR**

three basic functions can be reduced to **NAND** or **NOR**. Why??

Functionally complete sets

- **AND**, **OR**, **NOT** can be implemented by **NAND** or **NOR**

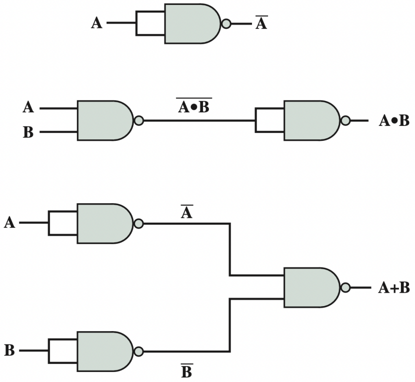


Figure 11.2 Some Uses of NAND Gates

- DeMorgan's law, $A + B = \overline{\overline{A} \cdot \overline{B}}$

Functionally complete sets

- DeMorgan's law, $A \cdot B = \overline{\overline{A} + \overline{B}}$.

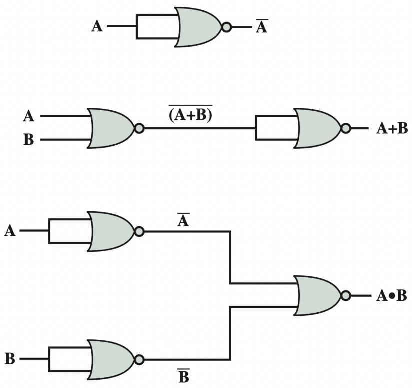


Figure 11.3 Some Uses of NOR Gates

Implementation of Boolean function

Table 11.3 A Boolean Function of Three Variables

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

(b) Boolean Operators Extended to More than Two Inputs (A, B, . . .)

Operation	Expression	Output = 1 if
AND	$A \cdot B \cdot \ldots$	All of the set {A, B, ...} are 1.
OR	$A + B + \ldots$	Any of the set {A, B, ...} are 1.
NAND	$\overline{A \cdot B \cdot \ldots}$	Any of the set {A, B, ...} are 0.
NOR	$\overline{A + B + \ldots}$	All of the set {A, B, ...} are 0.
XOR	$A \oplus B \oplus \ldots$	The set {A, B, ...} contains an odd number of ones.

- Sum of products (SOP): $F = XXX + XXX + XXX + \ldots$
  any of input combinations that produce 1 is true.
- Product of sums (POS): $F = XXX \cdot XXX \cdot XXX \ldots$

Implementation of Boolean function

- Sum of products (SOP): $F = XXX + XXX + XXX + \ldots$
- **Minterm**: the AND (product) of terms consists of exactly one instance of each literal, e.g. ABC.
- For the $OR(+)$ of minterms, all 0's will have no effect on the final outcome (because $0 + x = x$, for any x)
- 1's for minterms cause the output to be 1 ($1 + x = 1$, for any x).
- By collecting all minterms with $F = 1$, any logical expression can be expressed as a sum of minterms (sum-of-products)

Implementation of Boolean function

Table 11.3 A Boolean Function of Three Variables

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

- Minterms with $F = 1$: 3rd, 4th, 7th rows
- If the value of some variable is 1, take the variable without complementing it, e.g., 3rd row, B
- If the value of some variable is 0, take the variable by complementing it, e.g., 3rd row, $\overline{A}$, $\overline{C}$.
- $F = \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C}$

Implementation of Boolean function

- Product of sums (POS): non of the input combinations that produce 0 (the row with $F = 0$) is true

$$F = \overline{(\overline{A}\,\overline{B}\,\overline{C})} \cdot \overline{(\overline{A}\,\overline{B}\,C)} \cdot \overline{(A\,\overline{B}\,\overline{C})} \cdot \overline{(A\,\overline{B}\,C)} \cdot \overline{(A\,B\,C)}$$

Generalized DeMorgan's theorem:

$$\overline{(A \cdot B \cdot C)} = \overline{A} + \overline{B} + \overline{C}$$

- Simplified

$$F = (A + B + C) \cdot (A + B + \overline{C}) \cdot (\overline{A} + B + C) \cdot (\overline{A} + B + \overline{C}) \cdot (\overline{A} + \overline{B} + \overline{C})$$

- We can implement any logic function by sum-of-product implementation or POS.

Graphical Symbol of the implementation

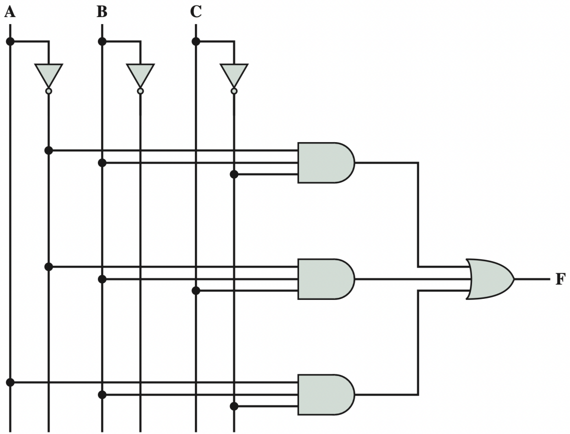


Figure 11.4 Sum-of-Products Implementation of Table 11.3

Graphical Symbol of the implementation

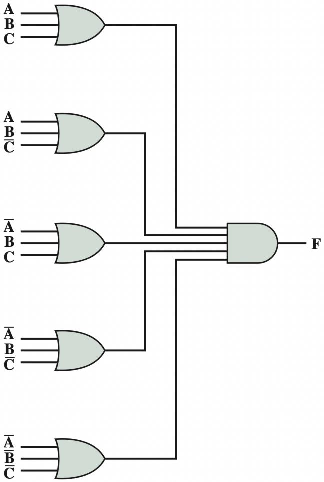


Figure 11.5 Product-of-Sums Implementation of Table 11.3

Simplification

- Sum of products (SOP) $F = \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C}$

Simplification

- Sum of products (SOP) $F = \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C}$
- Can we do something better? Yes,

Simplification

- Sum of products (SOP) $F = \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C}$
- Can we do something better? Yes,
- Idempotent law: $A + A = A$
- Inverse law: $A + \overline{A} = 1$
- Distributive Law: $A(B + C) = AB + AC$

Simplification

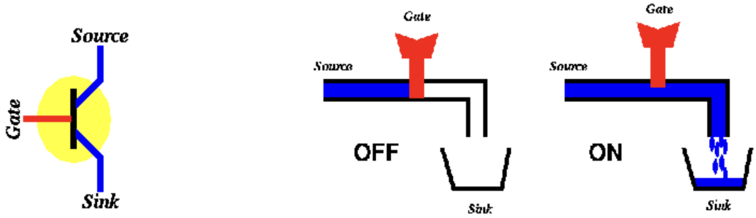
- Sum of products (SOP) $F = \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C}$
- Can we do something better? Yes,
- Idempotent law: $A + A = A$
- Inverse law: $A + \overline{A} = 1$
- Distributive Law: $A(B + C) = AB + AC$
- $\overline{A}B\overline{C} + \overline{A}BC = \overline{A}B(\overline{C} + C) = \overline{A}B$
  $\overline{A}B\overline{C} + AB\overline{C} = (\overline{A} + A)B\overline{C} = B\overline{C}$

Simplification

- Sum of products (SOP) $F = \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C}$
- Can we do something better? Yes,
- Idempotent law: $A + A = A$
- Inverse law: $A + \overline{A} = 1$
- Distributive Law: $A(B + C) = AB + AC$
- $\overline{A}B\overline{C} + \overline{A}BC = \overline{A}B(\overline{C} + C) = \overline{A}B$
  $\overline{A}B\overline{C} + AB\overline{C} = (\overline{A} + A)B\overline{C} = B\overline{C}$
-

$$
\begin{aligned}
F &= \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C} \\
&= (\overline{A}B\overline{C} + \overline{A}BC) + (\overline{A}B\overline{C} + AB\overline{C}) \\
&= \overline{A}B + B\overline{C} = B(\overline{A} + \overline{C})
\end{aligned}
$$

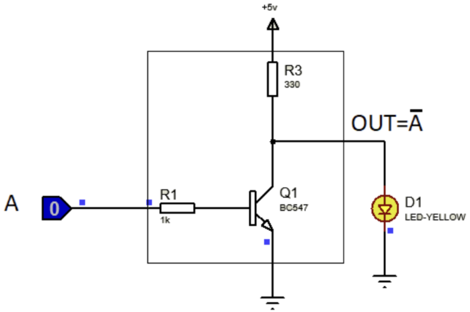
Logic gate implementation (optional)

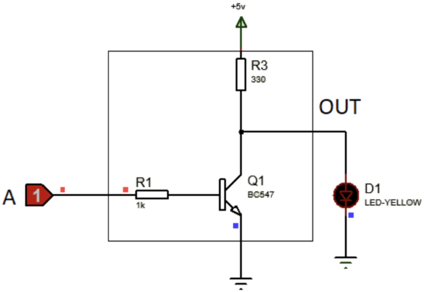


- Logic Gates are implemented using transistors.
- Transistor can be treated as switches, which turns on/off according to the value of input. If input is 1, the transistor is ON, otherwise it is OFF.

Logic gate implementation (optional)

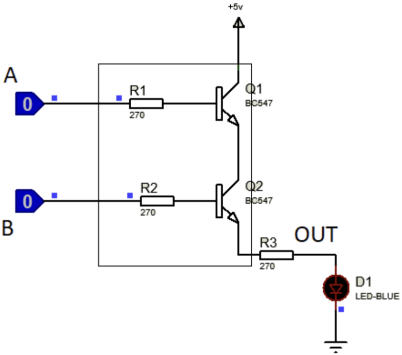
- NOT gate
- An additional resistor: protect transistors

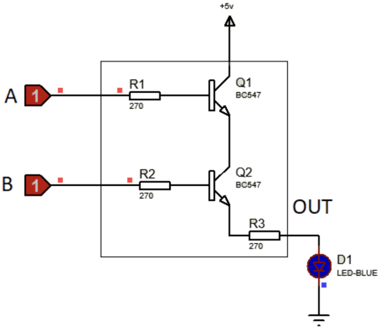




Logic gate implementation

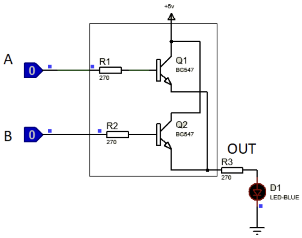
- AND gate

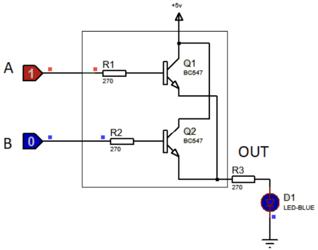


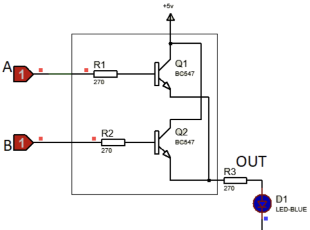


Logic gate implementation

- OR gate







Multiplexers (Optional)

- The multiplexer connects multiple inputs to a single output. One of the inputs is selected to be passed to the output.

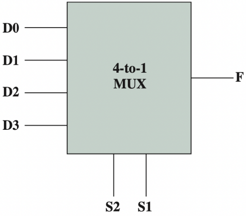


Figure 11.12 4-to-1 Multiplexer Representation

Table 11.7 4-to-1 Multiplexer Truth Table

S2	S1	F
0	0	D0
0	1	D1
1	0	D2
1	1	D3

Flip-Flops (Optional)

- The flip-flop is a bistable device. It exists in one of two states and, in the absence of input, remains in that state. Thus, the flip-flop can function as a 1-bit memory.
- The flip-flop has two outputs, which are always the complements of each other. These are generally labeled Q and $\overline{Q}$.

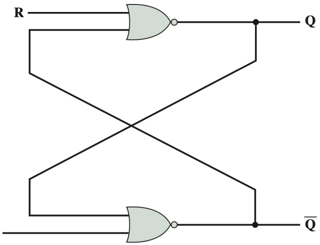


Figure 11.22 The S–R Latch Implemented with NOR Gates

Summary

- Basic operations of Boolean algebra
- All functions on binary numbers can be treated as logic functions.
- The logic function can then be implemented by SOP or POS
- The logic function can then be represented as either NAND or NOR function.
- How to use transistors to implement logic functions