Lecture 3

信息科学与技术学院
School of Information Science and Technology

Combinational Logic Circuits

Haoyu Wang
ShanghaiTech University

信息科学与技术学院
School of Information Science and Technology

Outline

- Logic Circuits Simplification
- Combinational Logic Circuits Design
- Karnaugh Map
- Exclusive-OR & Exclusive-NOR Circuits
- Enable/Disable Circuits

Outline

- Logic Circuits Simplification
- Combinational Logic Circuits Design
- Karnaugh Map
- Exclusive-OR & Exclusive-NOR Circuits
- Enable/Disable Circuits

Sum-of-Products Form

- **Sum-of-products (SOP)** :
  - » Multiple **AND** terms **OR**ed together

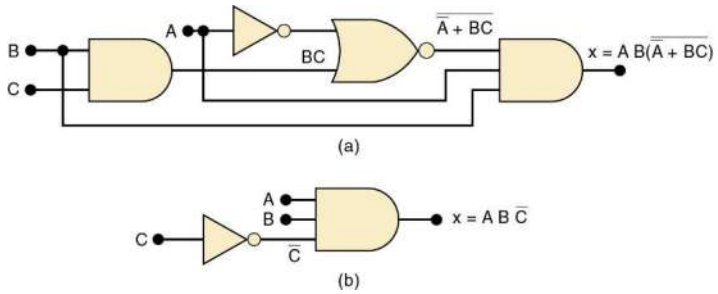
1. $ABC + \overline{A}B\overline{C}$
2. $AB + \overline{A}B\overline{C} + \overline{C}\,\overline{D} + D$
3. $\overline{A}B + C\overline{D} + EF + GK + H\overline{L}$

Products-of-Sums Form

- **Product-of-sums (POS)** expression:
  - » Multiple **OR** terms (sums) **AND**ed together.

1. $(A + \overline{B} + C)(A + C)$
2. $(A + \overline{B})(\overline{C} + D)F$
3. $(A + C)(B + \overline{D})(\overline{B} + C)(A + \overline{D} + \overline{E})$

Simplifying Logic Circuits



- (a) and (b) are equivalent!
  - » Circuit (b) is clearly less complex.

Algebraic Simplification

- (a)=> (b) logic circuit simplification
- Circuit simplification techniques
  - » Boolean algebra (algebraic simplification)
  - » Karnaugh map

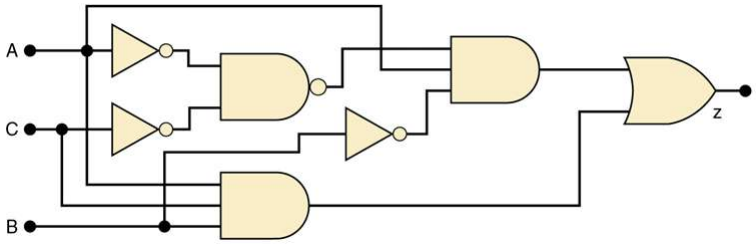
信息科学与技术学院
School of Information Science and Technology

Algebraic Simplification

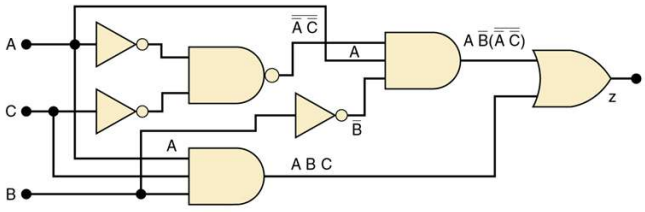
- 1. Place the expression in SOP form by applying DeMorgan's theorems and multiplying terms.
- 2. Check the SOP form for common factors
  - » Factoring where possible should eliminate one or more terms.

Example

Simplify the logic circuit shown.



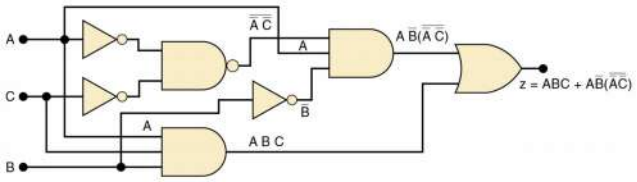
Step 1



Determine the expression for the output:

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

Step 2

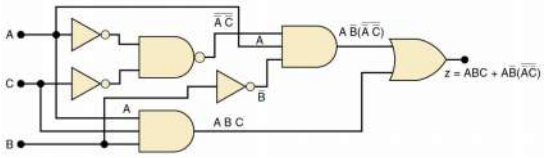


$$z = ABC + A\overline{B} \cdot (\overline{\overline{A}\ \overline{C}})$$

Convert the expression to SOP form by applying DeMorgan's theorems and multiplying terms

$$\begin{aligned} z &= ABC + A\overline{B}(\overline{\overline{A}} + \overline{\overline{C}}) && \text{[theorem (17)]} \\ &= ABC + A\overline{B}(A + C) && \text{[cancel double inversions]} \\ &= ABC + A\overline{B}A + A\overline{B}C && \text{[multiply out]} \\ &= ABC + A\overline{B} + A\overline{B}C && [A \cdot A = A] \end{aligned}$$

Step 3



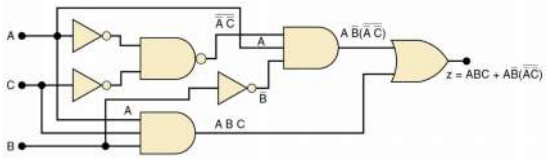
$$z = ABC + A\overline{B} + A\overline{B}C$$

Factoring

$$z = AC(B + \overline{B}) + A\overline{B}$$

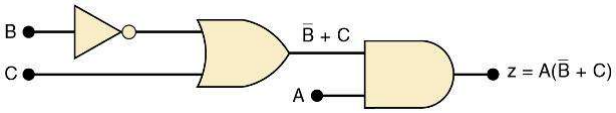
$$\begin{aligned} z &= AC(1) + A\overline{B} \\ &= AC + A\overline{B} \end{aligned}$$

Step 4



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

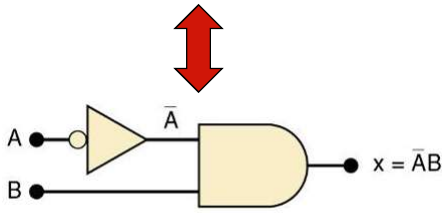
Draw the simplified logic circuit



Outline

- Logic Circuits Simplification
- Combinational Logic Circuits Design
- Karnaugh Map
- Exclusive-OR & Exclusive-NOR Circuits
- Enable/Disable Circuits

Description: circuit that produces a 1 output only for the A = 0, B = 1 condition.



- Question: How to implement a combinational digital circuit from the literal description?

Example

- Design a logic circuit
  - » The circuit has 3 inputs, A, B, and C, and 1 output, X.
  - » X will be HIGH only when a majority of the inputs are HIGH

Step 1

- Set up the truth table

Problem: Design a logic circuit that has 3 inputs, A, B, and C, and whose output will be HIGH only when a majority of the inputs are HIGH.

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Step 2

- Write the **AND** term for each case where the output is a 1.

A	B	C	X	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	→ A′BC
1	0	0	0	
1	0	1	1	→ AB′C
1	1	0	1	→ ABC′
1	1	1	1	→ ABC

Step 3

- Write the sum-of-products expression for the output

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$X=A'BC+AB'C+ABC'+ABC$

$A'BC$

$AB'C$

ABC'

ABC

Step 4

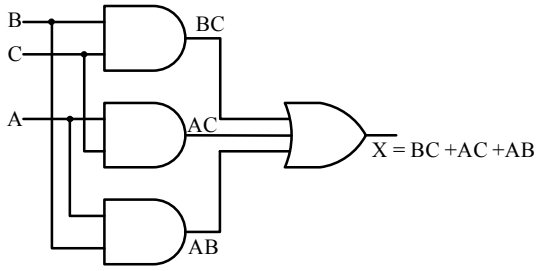
- Simplify the output expression
  - » Recall Algebraic Simplification

$$X=A'BC+AB'C+ABC'+ABC$$
$$=A'BC+ABC+AB'C+ABC+ABC'+ABC$$
$$X=BC(A'+A)+AC(B'+B)+AB(C'+C)$$
$$X=BC+AC+AB$$

Step 5

- Implement the circuit for the final expression

X =BC+AC+AB



Since the expression is in SOP form, the circuit is a group of **AND** gates, working into a single **OR** gate,

Generalized Procedures

» 1. Interpret the problem and set up its truth table.
» 2. Write the **AND** (product) term for each case where output = 1.
» 3. Combine the terms in SOP form.
» 4. Simplify the output expression if possible.
» 5. Implement the circuit for the final, simplified expression.

Outline

- Logic Circuits Simplification
- Combinational Logic Circuits Design
- Karnaugh Map
- Exclusive-OR & Exclusive-NOR Circuits
- Enable/Disable Circuits

信息科学与技术学院
School of Information Science and Technology

What is Karnaugh Map?

- A graphical method of simplifying logic equations or truth tables
- Also called a K map (卡诺图)

Example: 2-variable K map

A	B	X
0	0	1 → $\bar{A}\bar{B}$
0	1	0
1	0	0
1	1	1 → AB

$$x = \bar{A}\bar{B} + AB$$

	$\bar{B}$	B
$\bar{A}$	1	0
A	0	1

Example: 4-variable K map

A	B	C	D	X
0	0	0	0	0
0	0	0	1	1 → $\bar{A}\bar{B}\bar{C}D$
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1 → $\bar{A}B\bar{C}D$
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1 → $AB\bar{C}D$
1	1	1	0	0
1	1	1	1	1 → $ABCD$

$$X = \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}D + AB\bar{C}D + ABCD$$

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	0	0
$\bar{A}B$	0	1	0	0
AB	0	1	1	0
$A\bar{B}$	0	0	0	0

-Adjacent K map square differ in only one variable both horizontally and vertically.
-A SOP expression can be obtained by **OR**ing all squares that contain a 1.

Looping

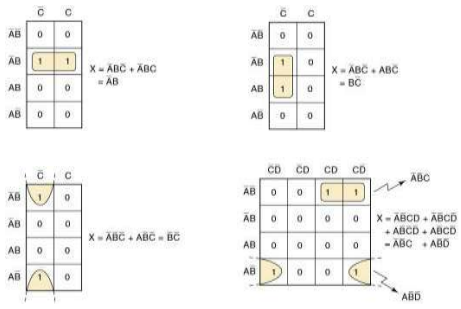
- Looping 1s in adjacent groups of 2, 4, or 8 will result in further simplification.

	C′	C
A′B′	0	0
A′B	1	1
AB	0	0
AB′	0	0

$X = A'BC' + A'BC = A'B$

Pairs

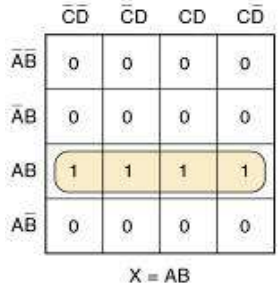
- Pairs: Looping 1s in adjacent groups of 2 elements



Note: Looping a pair of adjacent 1s eliminates **1** variable that appears in complemented and uncomplemented form

Quads

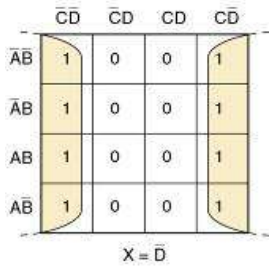
- Quads: Looping 1s in adjacent groups of 4 elements



Note: Looping a quad of adjacent 1s eliminates 2 variable that appears in complemented and uncomplemented form

Octets

- Octets: Looping 1s in adjacent groups of 8 elements



Note: Looping a octet of adjacent 1s eliminates 3 variable that appears in complemented and uncomplemented form

- Note:
  - » 1. When the largest possible groups have been looped, only the common terms are placed in the final expression.
  - » 2. Looping may also be wrapped between **top**, **bottom**, and **sides**.

Example: Step 1

- Construct the K map from the truth table
- Place 1s as indicated in the truth table.

X	C'D'	C'D	CD	CD'
A'B'	0	0	0	1
A'B	0	1	1	0
AB	0	1	1	0
AB'	0	0	1	0

Step 2

- Loop 1s that are not adjacent to any other 1s.

X	C'D'	C'D	CD	CD'
A'B'	0	0	0	1
A'B	0	1	1	0
AB	0	1	1	0
AB'	0	0	1	0

→ A'B'CD'

信息科学与技术学院
School of Information Science and Technology

Step 3

- Loop 1s that are in pairs.

X	C'D'	C'D	CD	CD'
A'B'	0	0	0	1
A'B	0	1	1	0
AB	0	1	1	0
AB'	0	0	1	0

→ A'B'CD'

→ ACD

Step 4

- Loop 1s in octets even if they have already been looped

X	C'D'	C'D	CD	CD'
A'B'	0	0	0	1
A'B	0	1	1	0
AB	0	1	1	0
AB'	0	0	1	0

A'B'CD'

ACD

There are no octets

Step 5

- Loop quads that have one or more 1s not already looped.

X	C'D'	C'D	CD	CD'
A'B'	0	0	0	1
A'B	0	1	1	0
AB	0	1	1	0
AB'	0	0	1	0

A'B'CD'

BD

ACD

Step 6

- Loop any pairs necessary to include 1st not already looped.

X	C'D'	C'D	CD	CD'
A'B'	0	0	0	1
A'B	0	1	1	0
AB	0	1	1	0
AB'	0	0	1	0

A'B'CD'

BD

ACD

All 1s have already been looped

Step 7

- Form the **OR** sum of terms generated by each loop.

X	C'D'	C'D	CD	CD'
A'B'	0	0	0	1
A'B	0	1	1	0
AB	0	1	1	0
AB'	0	0	1	0

A'B'CD'

BD

ACD

X = A'B'CD' + ACD + BD

Generalized Procedures

- 1. Construct the K map, place 1s as indicated in the truth table.
- 2. Loop 1s that are not adjacent to any other 1s.
- 3. Loop 1s that are in pairs.
- 4. Loop 1s in octets even if they have already been looped.
- 5. Loop quads that have one or more 1s not already looped.
- 6. Loop any pairs necessary to include 1st not already looped.
- 7. Form the **OR** sum of terms generated by each loop.

信息科学与技术学院
School of Information Science and Technology

X	C'D'	C'D	CD	CD'
A'B'	0	0	0	1
A'B	0	1	1	0
AB	0	1	1	0
AB'	0	0	1	0

- Note:
  - 1. When a variable appears in both complemented and uncomplemented form within a loop, that variable is eliminated from the expression.
  - 2. Variables that are the same for all squares of the loop must appear in the final expression.
  - 3. K map theoretically can be used for any number of input variables—practically limited to 5 or 6 variables.

Don't-Care Conditions

- *x* represents the don't care condition
  - » Its free to make the output for any don't care condition either a 0 or a 1 to produce the simplest output expression.

A	B	C	Z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	x
1	0	0	x
1	0	1	1
1	1	0	1
1	1	1	1

	C'	C
A'B'	0	0
A'B	0	x
AB	1	1
AB'	x	1

	C'	C
A'B'	0	0
A'B	0	0
AB	1	1
AB'	1	1

z = A

Outline

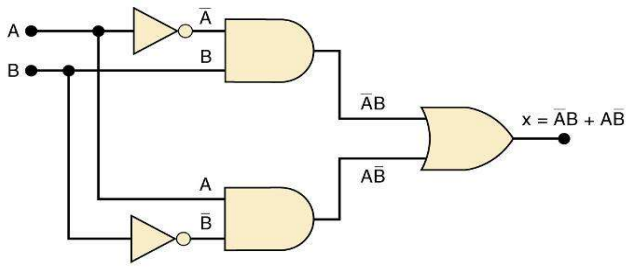
- Logic Circuits Simplification
- Combinational Logic Circuits Design
- Karnaugh Map
- Exclusive-OR & Exclusive-NOR Circuits
- Enable/Disable Circuits

Exclusive OR and Exclusive NOR Circuits

- The exclusive **OR** (**XOR**) produces a HIGH output whenever the two inputs are at *opposite* levels.

Exclusive OR (XOR)

Exclusive **OR** circuit and truth table.

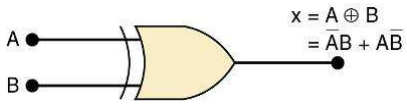


A	B	x
0	0	0
0	1	1
1	0	1
1	1	0

Output expression: $x = \overline{A}B + A\overline{B}$

This circuit produces a HIGH output whenever the two inputs are at opposite levels.

Traditional **XOR** gate symbol.



An **XOR** gate has only *two* inputs, combined so that $x = \overline{A}B + A\overline{B}$.

A shorthand way indicate the **XOR** output expression is: $x = A \oplus B$.

…where the symbol ⊕ represents the **XOR** gate operation.

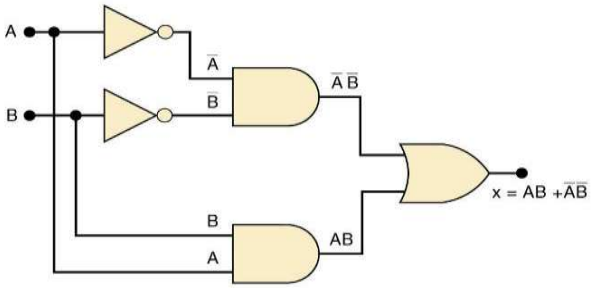
Output is **HIGH** only when the two inputs are at different levels.

信息科学与技术学院
School of Information Science and Technology

Exclusive NOR (XNOR)

- The exclusive **NOR** (**XOR**) produces a HIGH output whenever the two inputs are at the *same* level.
  - » **XOR** and **XNOR** outputs are opposite.

Exclusive **NOR** circuit and truth table.

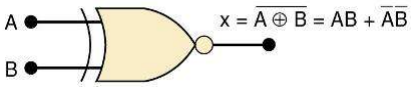


A	B	x
0	0	1
0	1	0
1	0	0
1	1	1

Output expression: $x = AB + \overline{A}\,\overline{B}$

XNOR produces a HIGH output whenever the two inputs are at the SAME levels.

Traditional **XNOR** gate symbol.



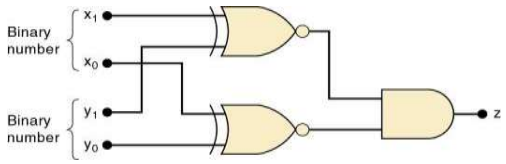
An **XNOR** gate has only *two* inputs, combined so that $x = AB + \overline{A}\,\overline{B}$.

A shorthand way indicate the **XOR** output expression is: $x = \overline{A \oplus B}$.

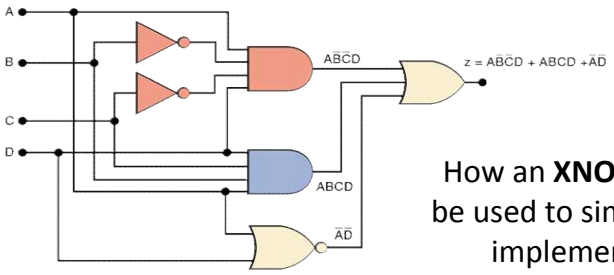
XNOR represents inverse of the **XOR** operation.

Output is HIGH only when the two inputs are at the same level.

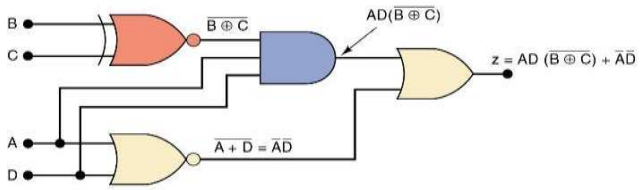
Truth table and circuit for detecting equality of two-bit binary numbers.



x_1	x_0	y_1	y_0	z (Output)
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



How an **XNOR** gate may be used to simplify circuit implementation.

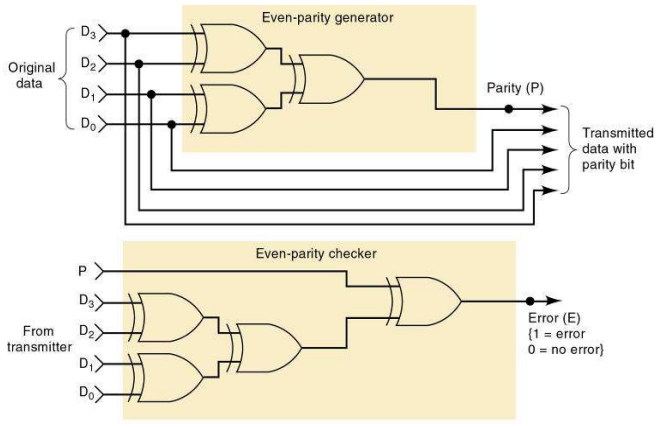


Even Parity

- When using even parity, the parity bit is set to 1 if the number of ones in a given set of bits (not including the parity bit) is odd, making the entire set of bits (including the parity bit even) even
- Example of even parity is
  - » 00000000
  - » 11010001

Parity Generator and Checker

XOR and **XNOR** gates are useful in circuits for parity generation and checking.



Outline

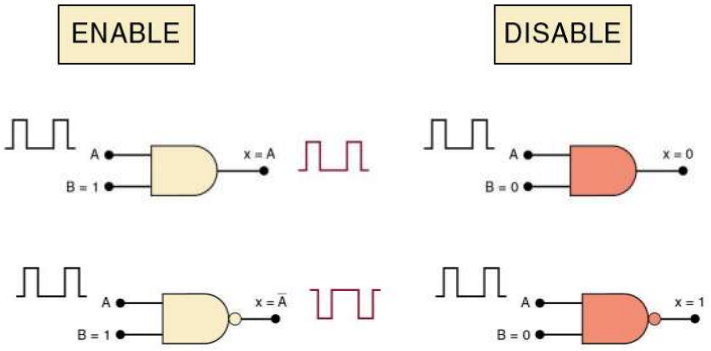
- Logic Circuits Simplification
- Combinational Logic Circuits Design
- Karnaugh Map
- Exclusive-OR & Exclusive-NOR Circuits
- Enable/Disable Circuits

Enable/Disable Circuits

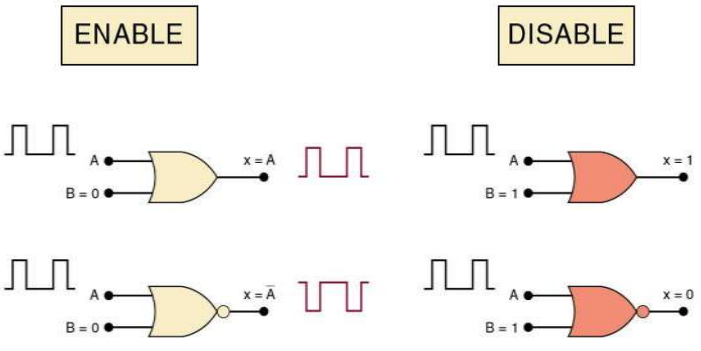
- A circuit is *enabled* when it *allows* the passage of an input signal to the output.
- A circuit is *disabled* when it *prevents* the passage of an input signal to the output.

Note: Situations requiring enable/disable circuits occur frequently in digital circuit design.

AND

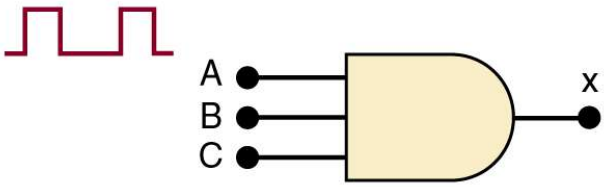

ENABLE
DISABLE
A
B = 1
x = A
A
B = 0
x = 0
A
B = 1
x = Ā
A
B = 0
x = 1

OR


ENABLE
DISABLE
A
B = 0
x = A
A
B = 1
x = 1
A
B = 0
x = Ā
A
B = 1
x = 0

Example 1

Input signal: **A** Control inputs: **B &C** Output**: X**



-Question: How can A be transmitted to X?

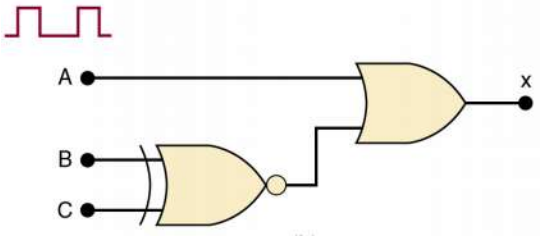
-Allows A to pass to X only when control inputs *B* and *C* are both HIGH.

-Otherwise, output will stay LOW.

信息科学与技术学院
School of Information Science and Technology

Example 2

Input Signal: **A** Control inputs: **B &C** Output: **X**



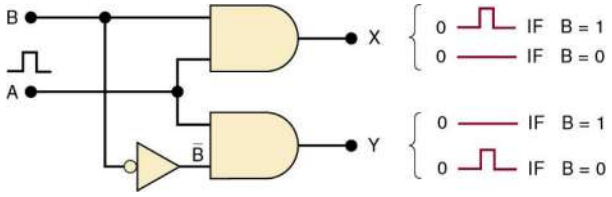
-Question: How can A be transmitted to X?

-Allow A to pass to output only when one, but *not* both control inputs are HIGH.

- Otherwise, output will stay HIGH.

Example 3

Input signal: **A** Control input: **B** Output: **X&Y**



-Question: How can A be transmitted to X?

-When B = 1, output X will follow input A, and output Y will be 0.
-When B = 0, output X will be 0, and output Y will follow input A.