AMRITA VISHWA VIDYAPEETHAM

AHEAD Online

# Logic Gates

Ms. Prathibha Prakash
Department of Computer Science
Amrita Vishwa Vidyapeetham

# Objective

- Basics of Logic Gates

# Logic gate

- Building blocks of digital electronic circuits.
- Boolean functions are implemented in digital circuits using these **logic gates**.
- Most logic gates have 2 inputs and 1 output.
- At any time, every terminal will be in one of the two binary conditions LOW (FALSE; 0;0V) or HIGH (TRUE;1;+5V).
- The function of each logic gate will be represented by Boolean expression

# Logic gate

- Logic gates are classified into:

  Basic gates : OR, AND, NOT

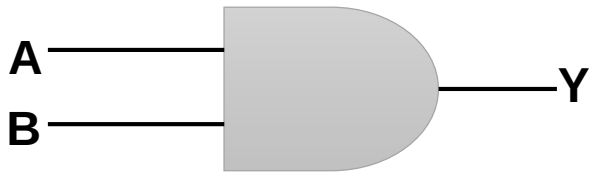
  Universal gate : NAND, NOR

  Special purpose gates: EX-OR, EX-NOR

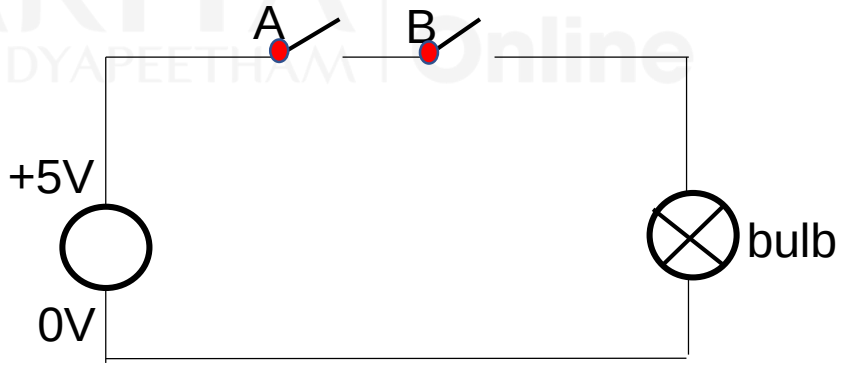
- Truth table explains all the outputs of a logic circuit for all possible inputs to that circuits.

# AND gate

- The realization of logical AND operator .
- Logic expression, **Y =A.B** or **AB**
- The circuit will give high output (1) if both inputs are high otherwise, the output is low




Logic symbol



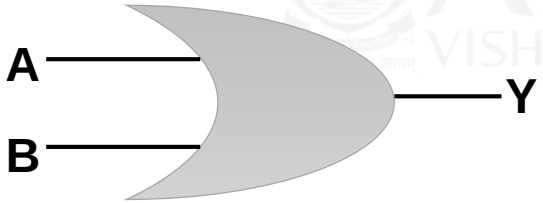

Switching circuit

| inputs | | output |
|---|---|---|
| **A** | **B** | **Y** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| **1** | **1** | **1** |

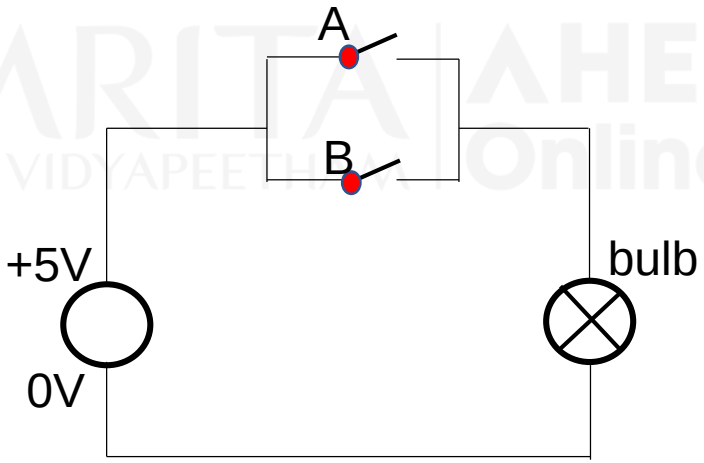
Truth table

# OR Gate

- Logic expression, **Y=A+B**
- The circuit will give high output if any one input is high otherwise the output is low.




Logic symbol



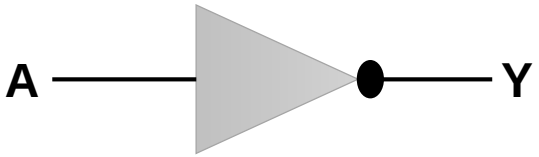

Switching circuit

| inputs | | output |
|---|---|---|
| A | B | Y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

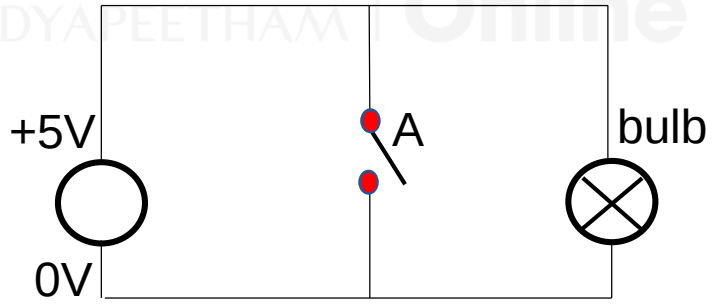
Truth table

# NOT Gate

- Single input single output gate.
- Also called Inverter because it inverts the given binary input.
- Logic expression, **$Y=A' =\overline{A}$**




Logic symbol




Switching circuit

| input | output |
|---|---|
| A | Y |
| 0 | 1 |
| 1 | 0 |

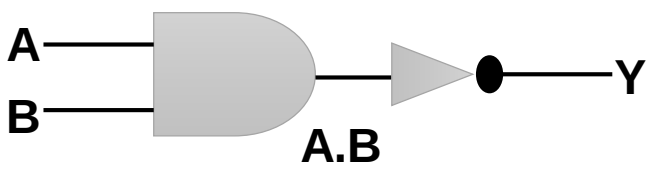
Truth table

# NAND Gate

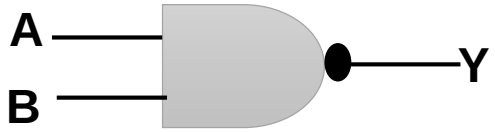
- NAND is a combination of AND gate and NOT gate
  **NOT + AND = NAND**

- The circuit will give high (1) output if any one input is low (0) otherwise, the output is low (0).

- Logical expression, $Y = \overline{A.B}$




or




| inputs | | output |
|---|---|---|
| A | B | $Y=\overline{A.B}$ |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

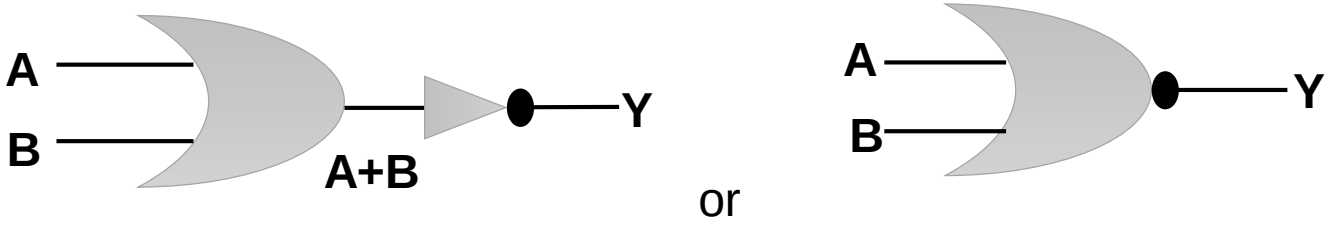
Truth table

# NOR Gate

- NOR is a combination of OR gate and NOT gate.
  **N**OT + **OR** = **NOR**

- The circuit will give high (1) output if both inputs is low (0) otherwise the output is high (1).

- Logical expression, $Y = \overline{A+B}$

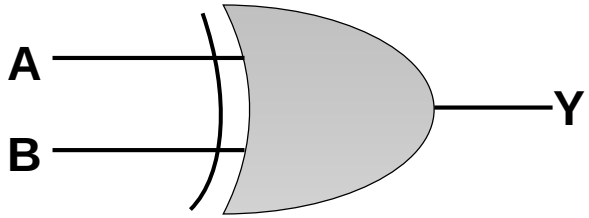



| inputs | | output |
|---|---|---|
| A | B | $Y=\overline{A+B}$ |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Truth table

# XOR Gate

- Exclusive-OR gate
- The output is high if the circuit input has odd number of 1's otherwise the output is low i.e., with even number of 1's.
- Logical expression **Y= A ⊕B = A'B+AB'** (SOP)



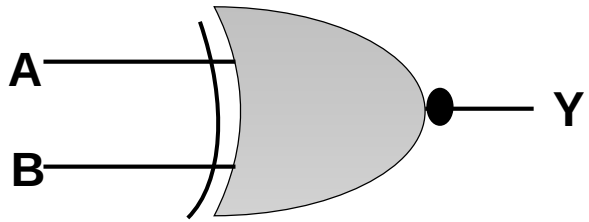

| inputs | | output |
|---|---|---|
| **A** | **B** | **Y** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Truth table

# X-NOR Gate

- Exclusive NOR gate or equivalence.
- The output is high if the circuit input has even number of 1's otherwise the output is low.
- Logical expression **Y = A ⊙ B = A'B'+AB**

Complement of XOR

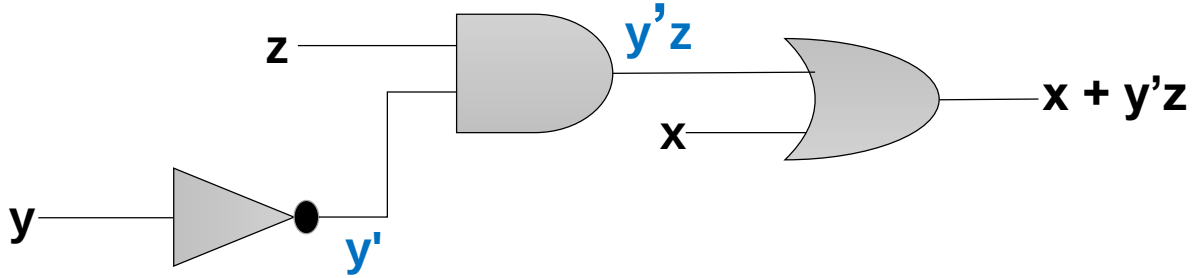



| inputs | | output |
|---|---|---|
| A | B | Y |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Truth table

# Realizing Logic circuit from function

- Boolean Function, **F = x + y'z**
  input variable : x, y, z
- Truth table
  3 input variables = $2^3$ combinations
- Logic circuit
  literals – 3 (x, y', z) and 2 terms

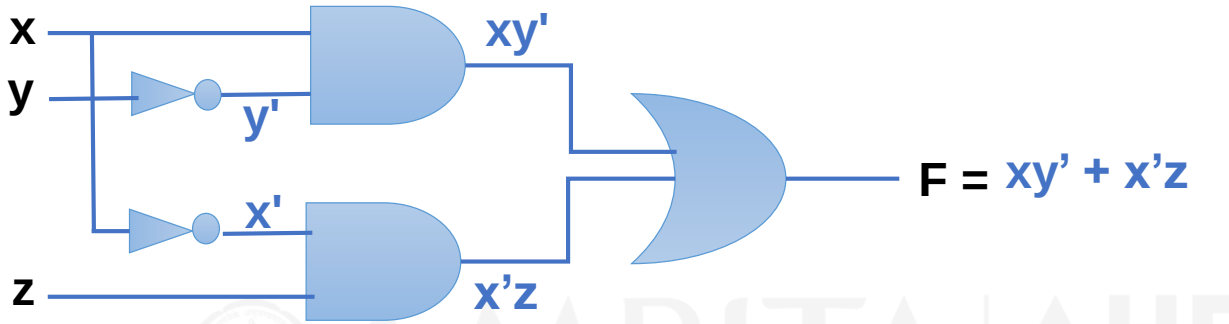


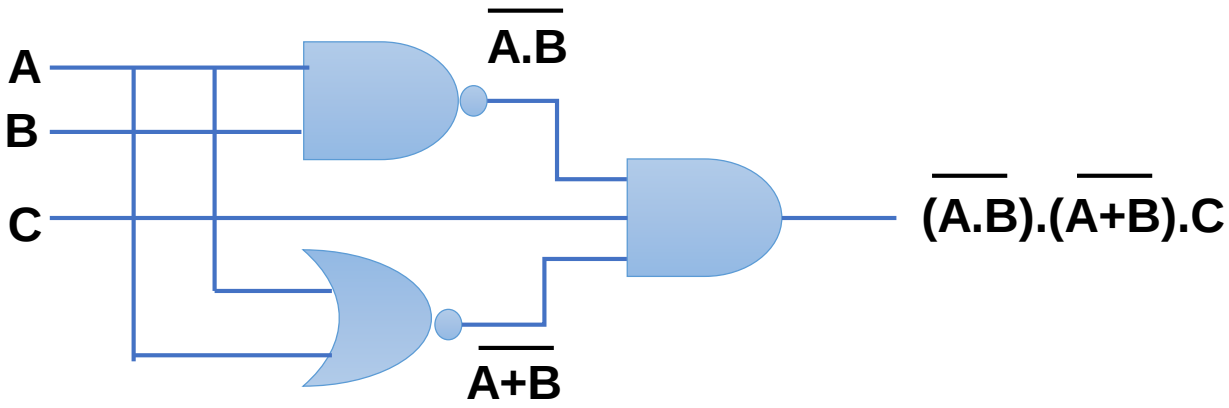

| inputs | | | output |
|---|---|---|---|
| x | y | z | F=x+y'z |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

→ x = 0,y =1,z = 0
F = 0 + 1'.0
= 0 + 0.0
= 0 + 0
= 0

→ x = 1,y = 1, z = 0
F = 1 + 1'. 0
= 1 + 0.0
= 1 + 0
= 1

# Realizing function from Logic circuit



# Summary

- Basics of Logic gates

# Reference


- M Morris Mano - Computer System Architecture - PHI - Third Edition
- Gideon Langholz, Abraha& Joe L Mott - Digital Logic Design - World Scientific Publishing Co Ltd
- Thomas C Bartee - Digital Computer Fundamentals - Tata Mc Graw Hill - Sixth Edition

AMRITA VISHWA VIDYAPEETHAM

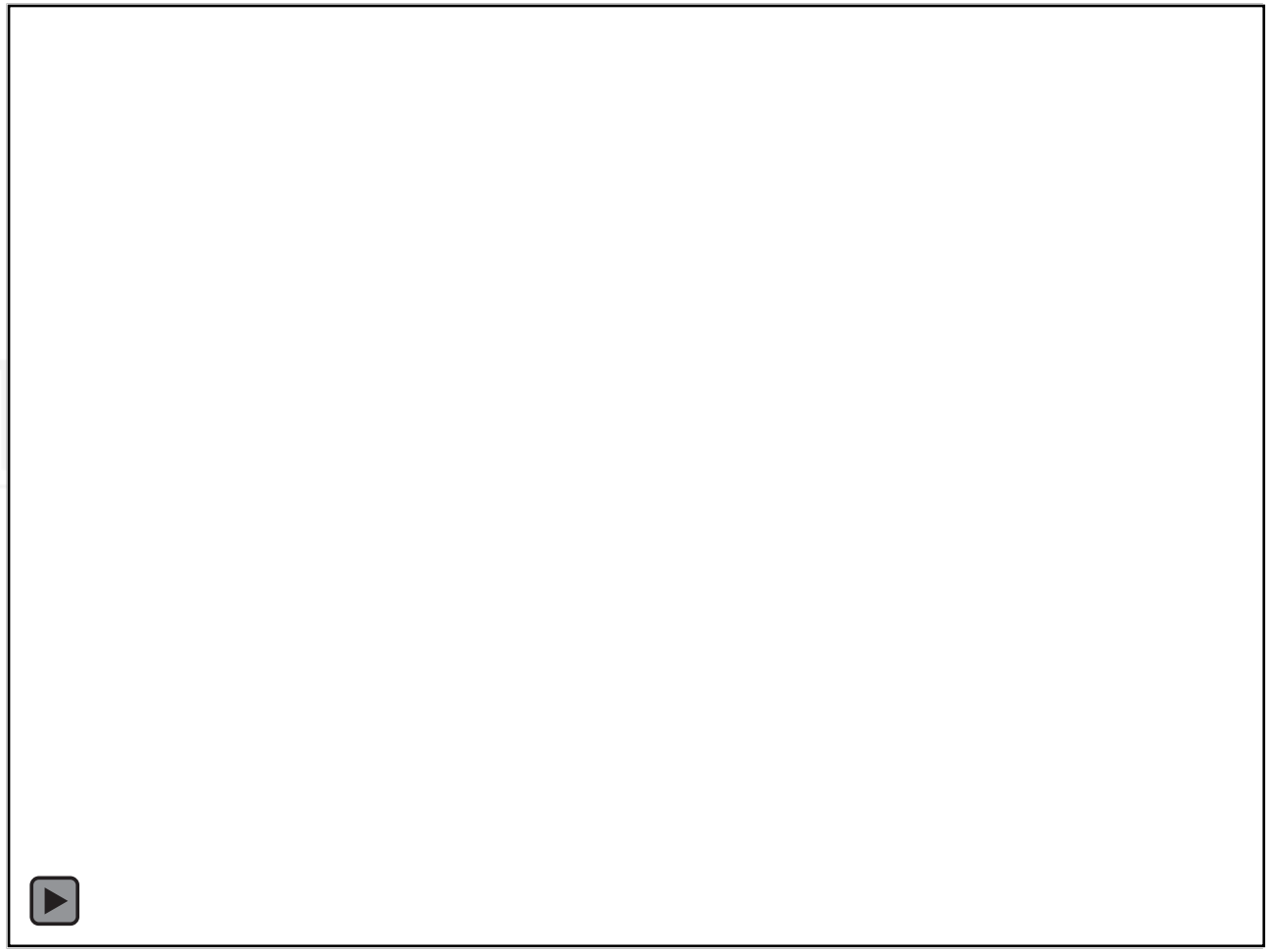
AHEAD Online

# Universal Logic Gate

Ms. Prathibha Prakash
Department of Computer Science
Amrita Vishwa Vidyapeetham

# Objective

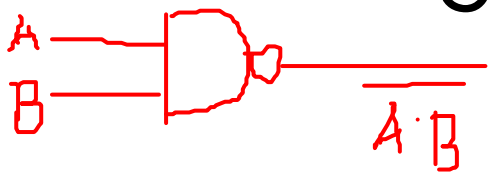
- Universal Logic gates
- NAND gate
- NOR gate

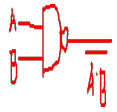


# Introduction

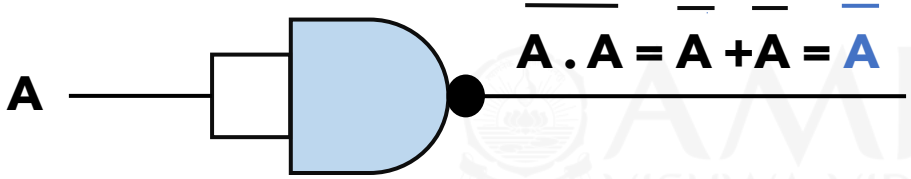
- **NAND** & **NOR** gate are known as the **universal gates**.
- These gates alone sufficient to implement any Boolean expression and are inexpensive.
- Basic logic gates (AND, OR and NOT) are logically complete.
- Sufficient to show that AND, OR and NOT can be implemented with NAND and NOR gate.

# Universal NAND gate

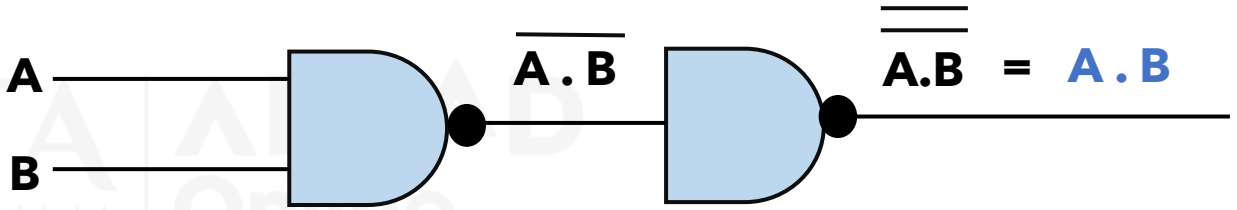


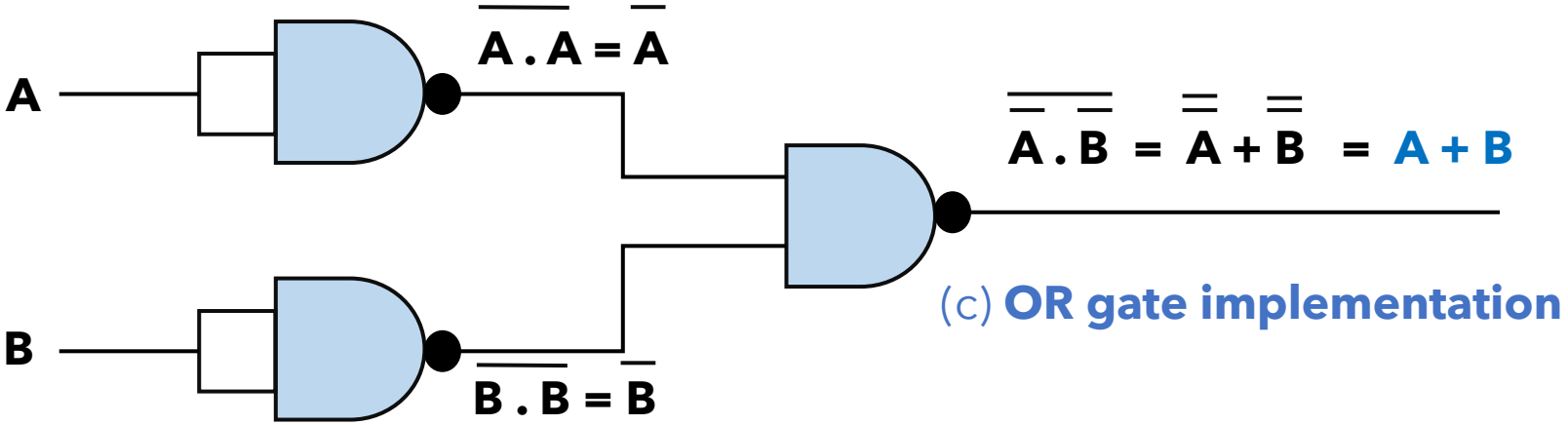
- We can implement NOT, AND and OR gates by combining one or more NAND gates.




(a) **NOT gate implementation**



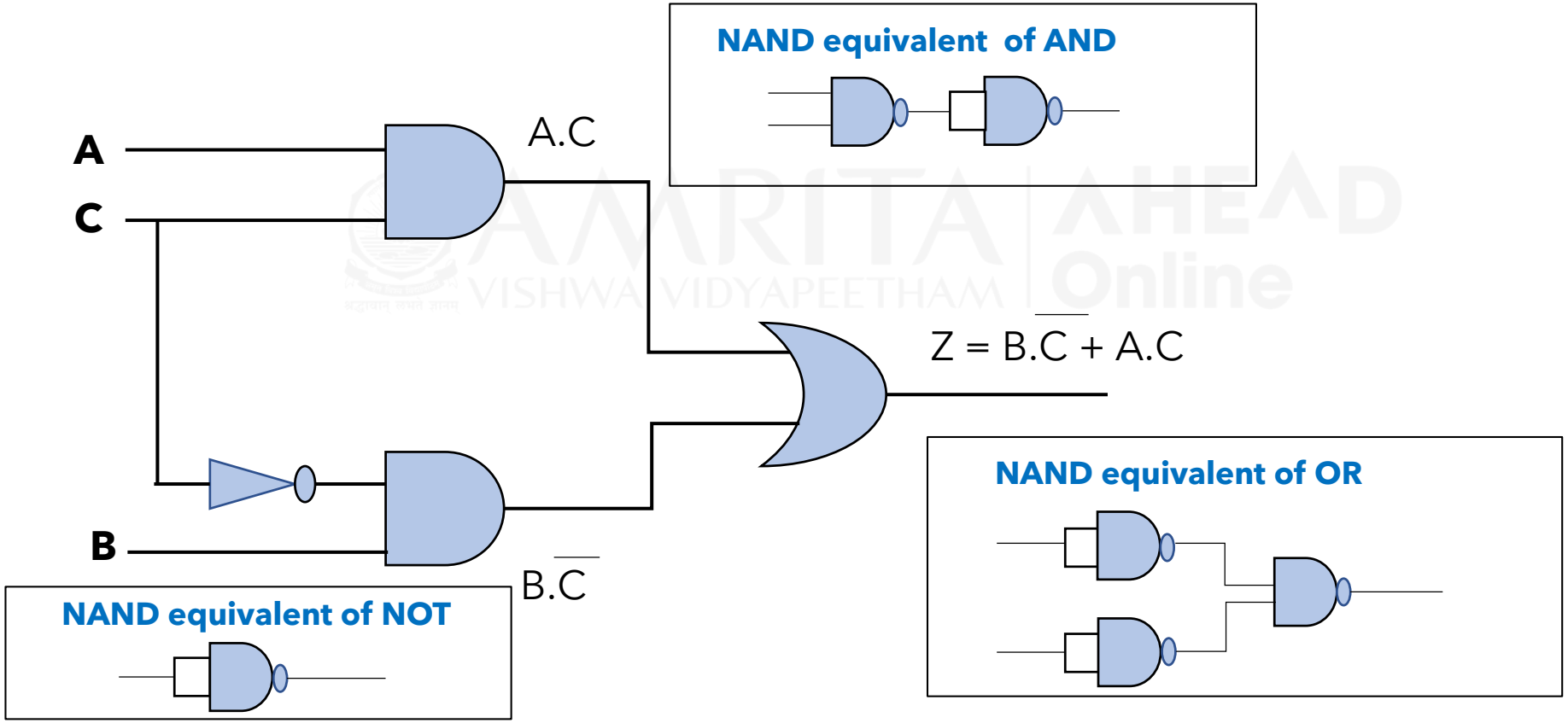

(b) **AND gate implementation**



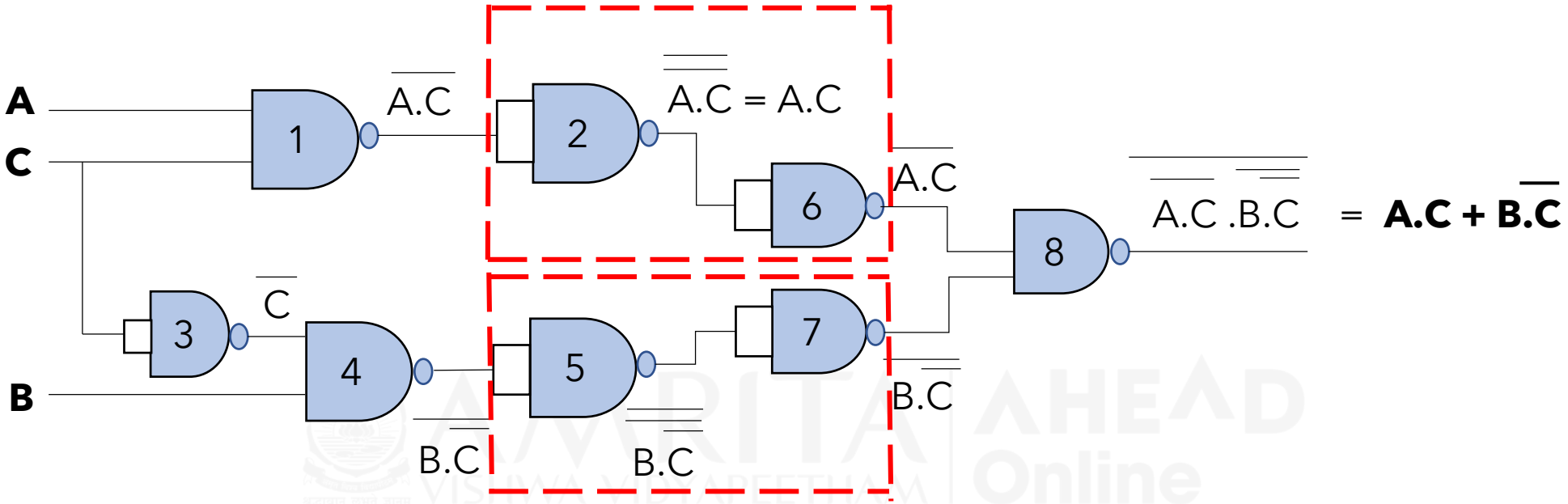

(c) **OR gate implementation**

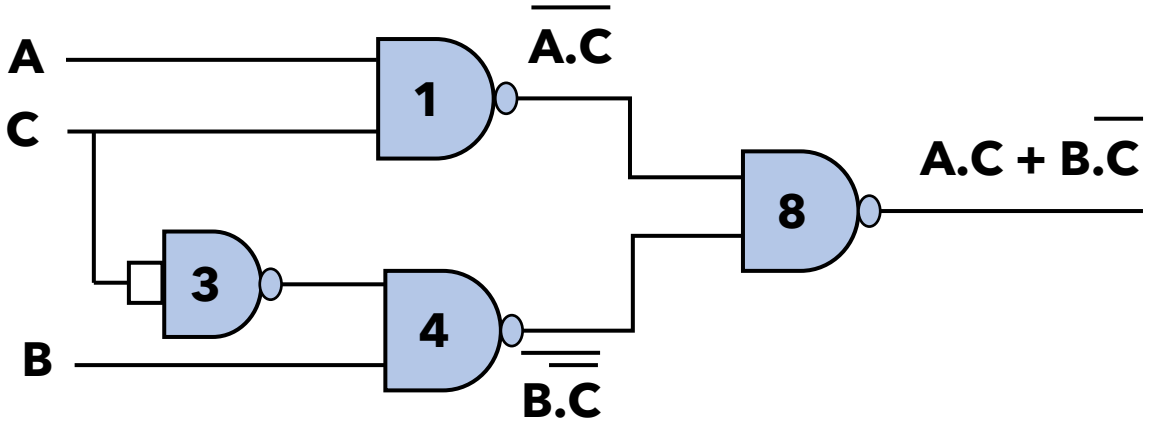
# NAND Logic circuit

- Design NAND logic circuit for the Boolean expression $Z = B\overline{C} + AC$

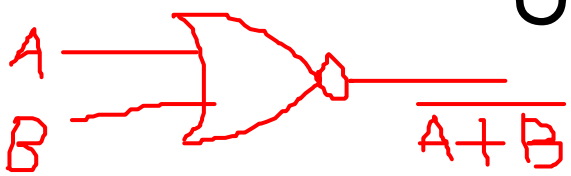


# NAND logic circuit

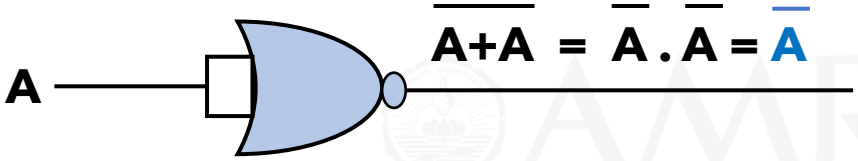


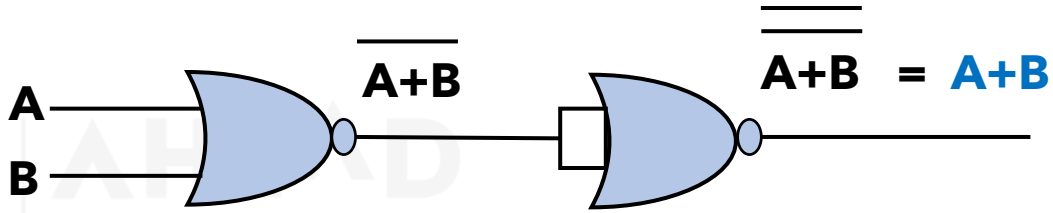
# Universal NOR gate



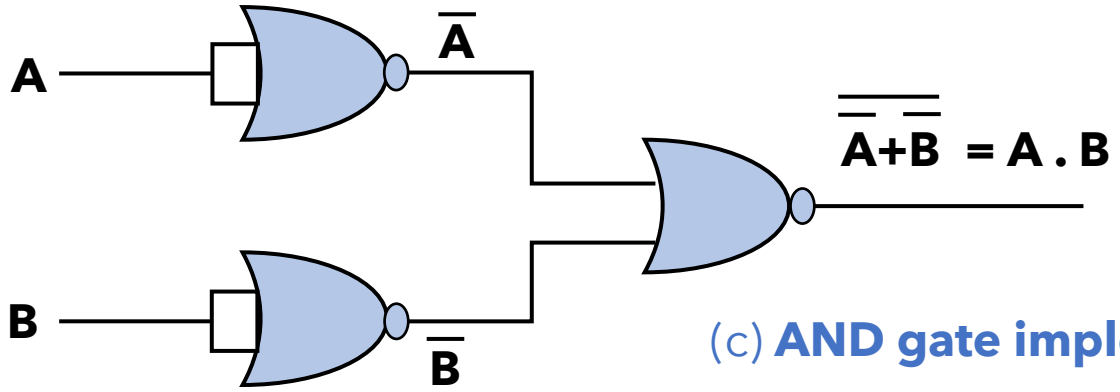

- Implement NOT, AND and NOR gates by combining one or more NOR gates.




(a) **NOT gate implementation**



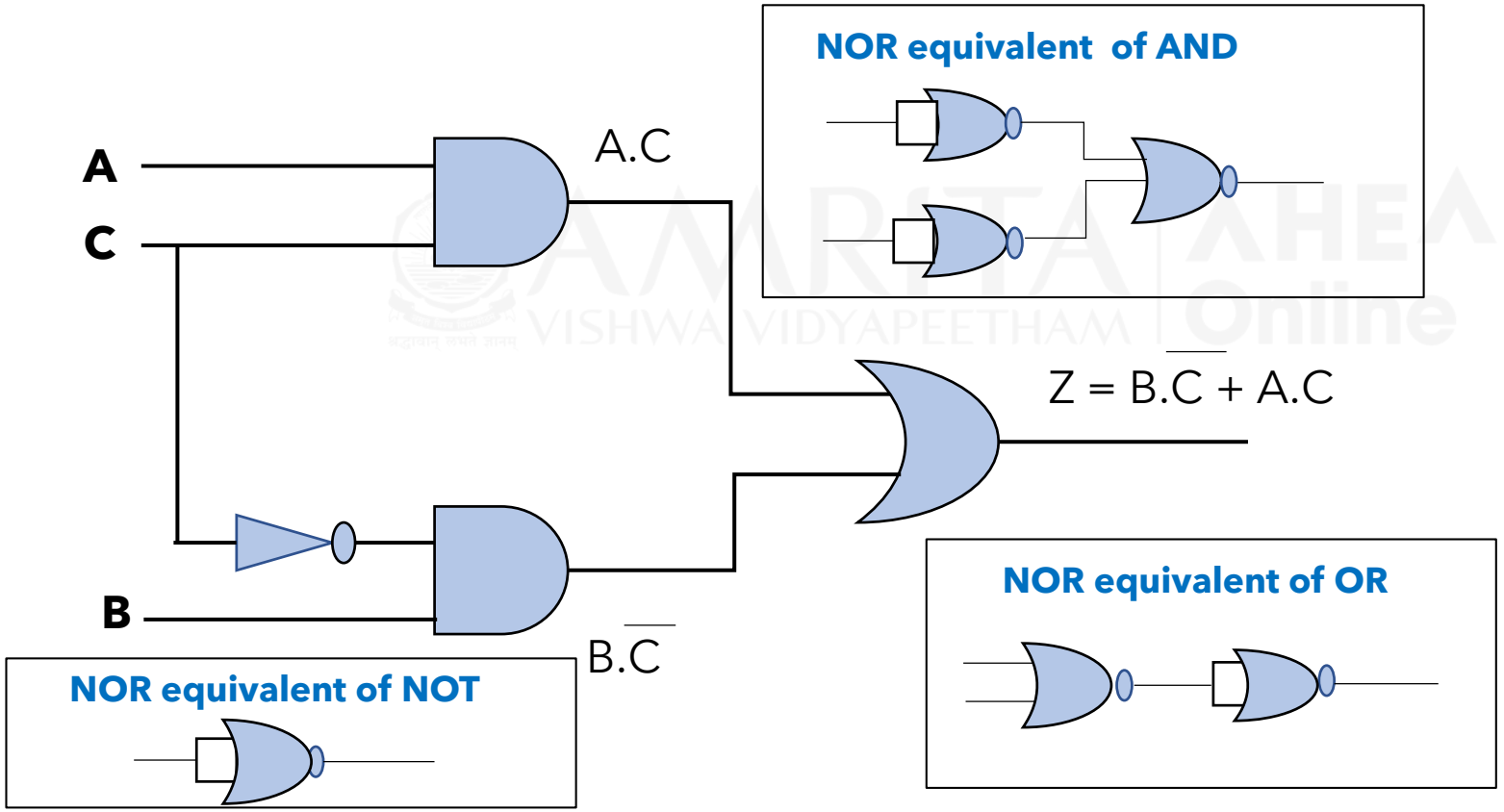

(b) **OR gate implementation**



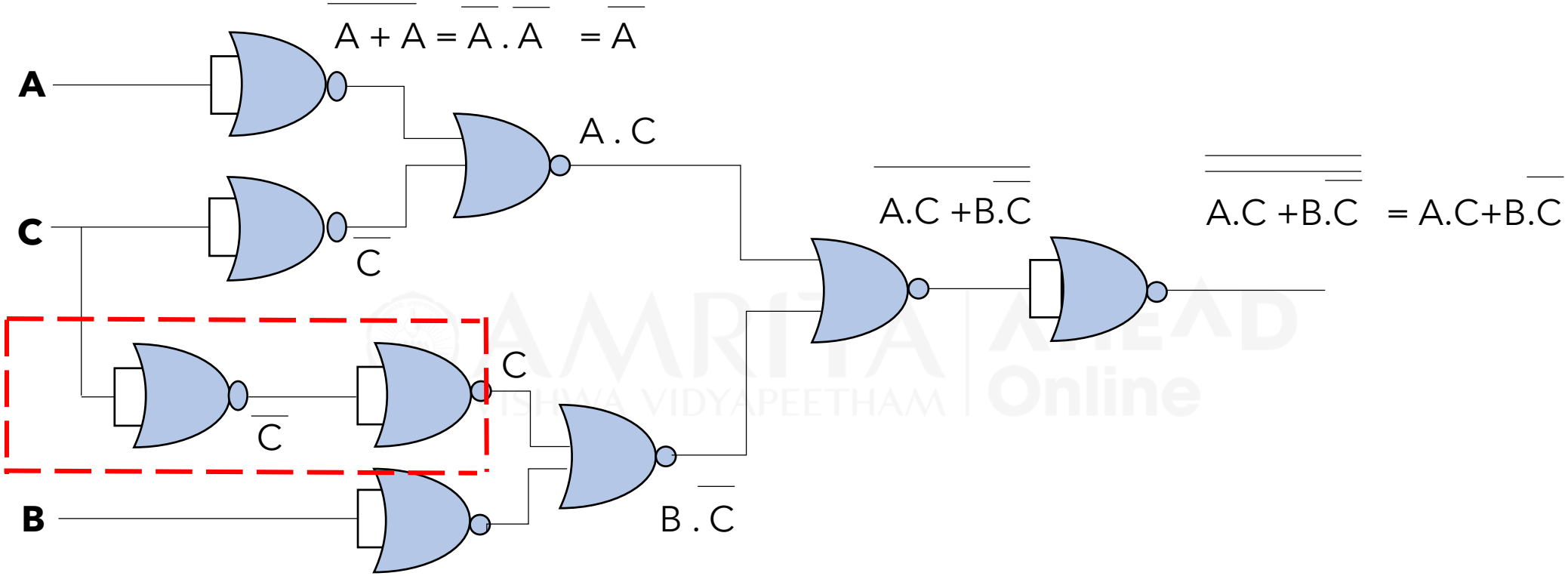

(c) **AND gate implementation**

# NOR Logic circuit

- Design NOR logic circuit for the Boolean expression $Z = B\overline{C} + AC$

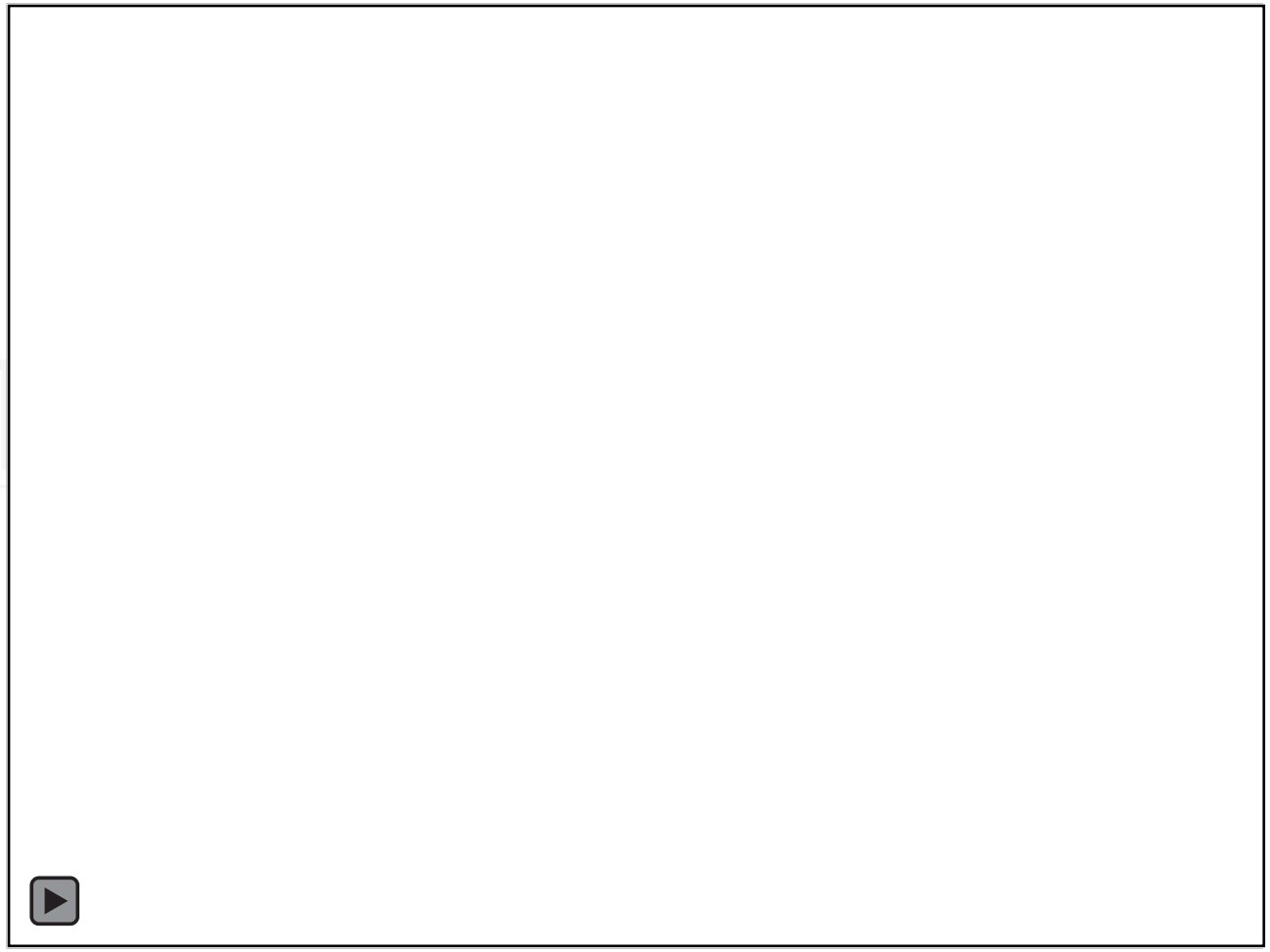


# NOR Logic circuit



# Summary

- Universal Logic gates



# Reference


- M Morris Mano - Computer System Architecture - PHI - Third Edition
- Gideon Langholz, Abraha& Joe L Mott - Digital Logic Design - World Scientific Publishing Co Ltd
- Thomas C Bartee - Digital Computer Fundamentals - Tata Mc Graw Hill - Sixth Edition


# Canonical & Standard Forms

Ms. Prathibha Prakash
Department of Computer Science
Amrita Vishwa Vidyapeetham

# Objective

Representation of Boolean Function

- Standard form
- Canonical Form

# Introduction

- Boolean functions can be expressed in **canonical** and **standard forms**

- In the Boolean expression binary variable will be either in normal form (x) or in its complement form (x').

F(A,B,C,D) is a function :

- **Product term** : Logical ANDed among the literals.
  e.g., AB, A'B, ABC'

- **Sum term** : Logical ORed among the literals.
  e.g., A+B, A'+B, A+B+C'

# Minterm : $m_j$

- **Minterm /Standard product** : Product term containing all the variables in normal or as complements.
  e.g.,F(A,B,C) : ABC' , ~~AB~~
- For n- variable function have $2^n$ possible minterms.

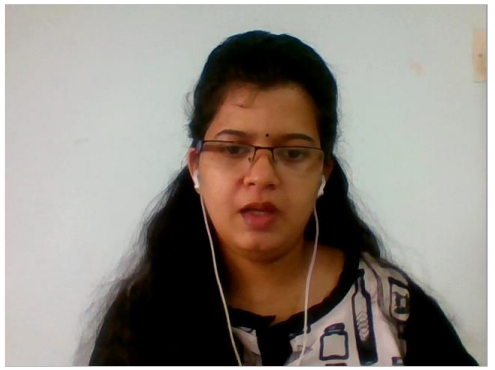
- While obtaining minterm, each variable is complemented if the corresponding bit of binary number is 0 and uncomplemented if 1.

| x | y | minterm | symbol |
|---|---|---|---|
| 0 | 0 | x'y' | $m_0$ |
| 0 | 1 | x'y | $m_1$ |
| 1 | 0 | xy' | $m_2$ |
| 1 | 1 | xy | $m_3$ |

**A minterm equals 1 at exactly one input combination and is equal to 0 otherwise.**
**e.g., x'y' = 1 only when x = 0, y = 0**

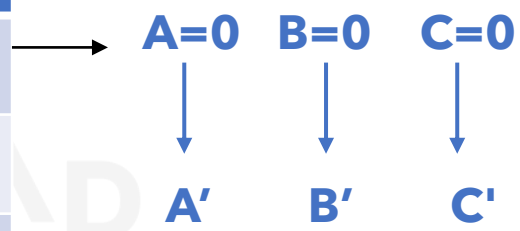
**Minterms with 2 binary variables x and y**
**2 variable - $2^2$ = 4 minterms**
**x is primed if x=0 ; x is unprimed if x=1**

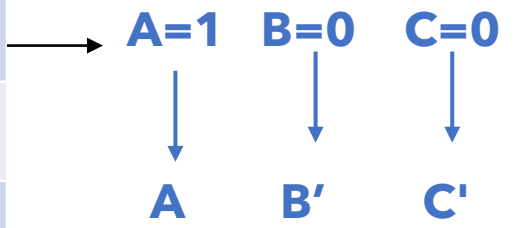
# Minterms



- 3 variable function with $2^3 = 8$ possible minterms

| A | B | C | minterm | symbol |
|---|---|---|---|---|
| 0 | 0 | 0 | A′B′C' | $m_0$ |
| 0 | 0 | 1 | A′B′C | $m_1$ |
| 0 | 1 | 0 | A′BC' | $m_2$ |
| 0 | 1 | 1 | A′BC | $m_3$ |
| 1 | 0 | 0 | AB′C' | $m_4$ |
| 1 | 0 | 1 | AB′C | $m_5$ |
| 1 | 1 | 0 | ABC' | $m_6$ |
| 1 | 1 | 1 | ABC | $m_7$ |



# Expressing functions as Sum of **Minterms**

- Boolean function can be expressed algebraically from the truth table
  - ➢ Select the minterms that produces a 1 in the function.
  - ➢ Take OR of those selected minterms.

i.e., sum of all minterms

$$\mathbf{f} = x'y'z + xy'z' + xyz$$

$$= m_1 + m_4 + m_7$$

$$= \Sigma(1,4,7)$$

| x | y | z | f | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 1 | **1** | x'y'z ← |
| 0 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 0 | |
| 1 | 0 | 0 | **1** | xy'z' ← |
| 1 | 0 | 1 | 0 | |
| 1 | 1 | 0 | 0 | |
| 1 | 1 | 1 | **1** | xyz ← |

# Expressing functions as Sum of **Minterms**

- Any function can be expressed by **ORing all minterms ($m_i$)** corresponding to input combinations (i) at which the function has a value of 1.

- Resulting expression is referred to as SUM of minterms and is expressed as F = Σ(2, 4, 5, 7) where Σ indicates ORing of the indicated minterms.

- F = Σ(2, 4, 5, 7) = (m2 + m4 + m5 + m7)

# Summary

- Canonical forms : Sum of Minterms

# Reference


- M Morris Mano - Computer System Architecture - PHI - Third Edition
- Gideon Langholz, Abraha& Joe L Mott - Digital Logic Design - World Scientific Publishing Co Ltd
- Thomas C Bartee - Digital Computer Fundamentals - Tata Mc Graw Hill - Sixth Edition

AMRITA VISHWA VIDYAPEETHAM

AHEAD Online

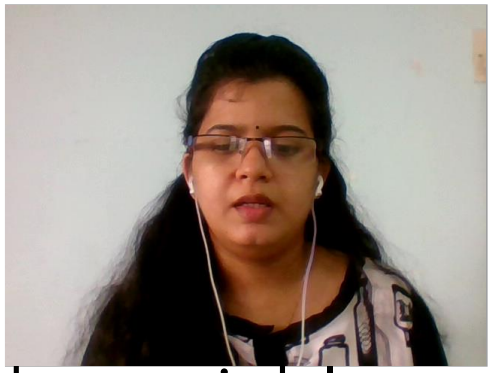
# Canonical & Standard form-II

Ms. Prathibha Prakash
Department of Computer Science
Amrita Vishwa Vidyapeetham

# Objective

- Canonical form and standard form using maxterms

# Maxterm : $M_j$



- **Maxterm /Standard sums** : Sum term containing all the variables or their complements.

  **e.g., F(A,B,C) : A+B+C′ , A′+C**

- For n – variables can have $2^n$ possible maxterms.

- For maxterm, each variable is primed if the corresponding bit of binary number is 1 and unprimed if 0

| x | y | maxterm | symbol |
|---|---|---|---|
| 0 | 0 | x+y | $M_0$ |
| 0 | 1 | x+y' | $M_1$ |
| 1 | 0 | x′+y | $M_2$ |
| 1 | 1 | x′+y' | $M_3$ |

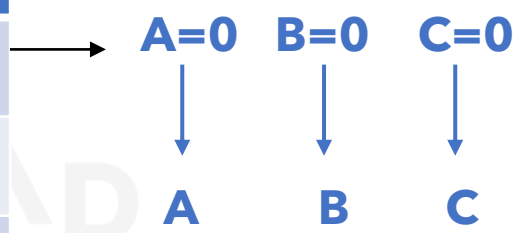
**A maxterm equals 0 at exactly one input combination and is equal to 1 otherwise.**
**e.g., x+y′ = 0 only when x = 0, y = 1**

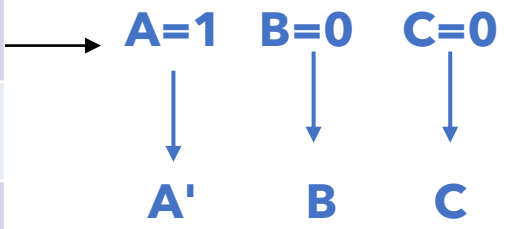
**Maxterms with 2 binary variables x and y**
**2 variable - 2^2 = 4 maxterms**
**x is primed if x=1 ; x is unprimed if x=0**

# Maxterms

- 3 variable function with $2^3 = 8$ possible maxterms

| A | B | C | maxterm | symbol |
|---|---|---|---|---|
| 0 | 0 | 0 | A+B+C | $M_0$ |
| 0 | 0 | 1 | A+B+C' | $M_1$ |
| 0 | 1 | 0 | A+B′+C | $M_2$ |
| 0 | 1 | 1 | A+B′+C' | $M_3$ |
| 1 | 0 | 0 | A′+B+C | $M_4$ |
| 1 | 0 | 1 | A′+B+C' | $M_5$ |
| 1 | 1 | 0 | A′+B′+C | $M_6$ |
| 1 | 1 | 1 | A′+B′+C' | $M_7$ |



# Expressing functions as Product of **Maxterms**

- Select maxterms that produces a 0 in the function
- Take AND of those selected maxterm

i.e., product of all maxterms

**f** = (x+y+z)(x+y′+z)(x+y′+z′)(x′+y+z′)(x′+y′+z)

$= M_0.M_2.M_3.M_5.M_6$

$= \Pi(0, 2, 3, 5, 6)$

| | x | y | z | f |
|---|---|---|---|---|
| x+y+z ← | **0** | **0** | **0** | **0** |
| | 0 | 0 | 1 | 1 |
| x+y′+z ← | **0** | **1** | **0** | **0** |
| x+y′+z' ← | **0** | **1** | **1** | **0** |
| | 1 | 0 | 0 | 1 |
| x′+y+z' ← | **1** | **0** | **1** | **0** |
| x′+y′+z ← | **1** | **1** | **0** | **0** |
| | 1 | 1 | 1 | 1 |

# Maxterm from Minterm

- Minterm and Maxterm are complement to each other.
- Complement of minterm = maxterm

| x | y | z | F | F′ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

Example : $F(x, y, z) = \Sigma (1, 4, 5, 6, 7)$

$F'(x, y, z) = \Sigma(0, 2, 3) = (m0 + m2 + m3)$

Take complement of F′ using De-morgan's law,

$F(x, y, z) = (m0+m2+m3)' = (m0'.m2'.m3') = (M0.M2.M3) = \Pi(0, 2, 3)$

# Canonical & Standard Form

- The sum of minterms and product of maxterms are called the **canonical forms**.

- In canonical form, each minterm or maxterm contain all variables in complemented or uncomplemented form.

  e.g., F(A, B, C) = ABC + A'B'C'

- **Standard form** : simplified versions of canonical forms

  The terms that form the function contain one, two, or more literals each.

  e.g., F(A, B, C) = (A+ B+ C)(A'+B')

# Canonical V/s Standard Form

| Canonical Form | Standard Form |
|---|---|
| Representation that helps to describe the Boolean functions | A simplified version of Canonical form |
| Divides into Canonical SoP (Sum of minterms) and Canonical PoS (Product of maxterms) | Divides into Standard SOP and Standard POS |
| More complex | Simple |
| E.g. F(a,b,c) = a'b'c + ab'c + abc | E.g. F(a,b,c) = a + b'c |

# Summary

- Canonical and Standard forms : POS

# Reference


- M Morris Mano - Computer System Architecture - PHI - Third Edition
- Gideon Langholz, Abraha& Joe L Mott - Digital Logic Design - World Scientific Publishing Co Ltd
- Thomas C Bartee - Digital Computer Fundamentals - Tata Mc Graw Hill - Sixth Edition

AMRITA VISHWA VIDYAPEETHAM

AHEAD Online

# Karnaugh map

Ms. Prathibha Prakash
Department of Computer Science
Amrita Vishwa Vidyapeetham

# Objective

- Systematic method of simplifying Boolean Expression – **Karnaugh map**

# Introduction

- Pictorial representation of writing Boolean expression.
- The map method modified by Karnaugh is called **K-map** or **Karnaugh map**.
- Objective of K-map is the minimization of Boolean function with a simple procedure.
- K-map : Diagram made up of squares ; n-variable $2^n$ squares

  Each square represents one minterm of the function

  Visual diagram of a function in standard form(SOP or POS)

# K-maps

- K-maps use 2-diamensional tables to simplify the Boolean expression.

| A \ B | 0 | 1 |
|---|---|---|
| 0 | $m_{00}$ | $m_{01}$ |
| 1 | $m_{10}$ | $m_{11}$ |

**2-variable map**

2 variable = f(A,B)

$2^2$minterm = 4 cell map

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | $m_0$ | $m_1$ | $m_3$ | $m_2$ |
| 1 | $m_4$ | $m_5$ | $m_7$ | $m_6$ |

**3-variable map**

3-variable = f(A,B,C)

$2^3$ minterm = 8 cell map

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | $m_0$ | $m_1$ | $m_3$ | $m_2$ |
| 01 | $m_4$ | $m_5$ | $m_7$ | $m_6$ |
| 11 | $m_{12}$ | $m_{13}$ | $m_{15}$ | $m_{14}$ |
| 10 | $m_8$ | $m_9$ | $m_{11}$ | $m_{10}$ |

**4-variable map**

**Adjacent cells differ by just one single bit**

# K- map Simplification

- First fill the appropriate cells with the value based on the output variable.
- Group the maximum number of 1's (SOP) and 0's (POS).
- Group need to be in power of 2 and done in decreasing order.
  If 8 cell K-map, try grouping for 8 ($=2^3$), then for 4 ($=2^2$), next for 2 ($=2^1$) and last consider single bit.
- Later each group is expressed in terms of common input variable in the rows and column.

# Grouping Rules

- Group the cells with largest number of consecutive 1's and no 0's .

correct

| | | | |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |

Incorrect

| | | | |
|---|---|---|---|
| 0 | 1 | 1 | 0 |

- Grouping must be done in decreasing order and the number of 1's in the group must be a power of 2.

incorrect

| | | | |
|---|---|---|---|
| 0 | 1 | 1 | 1 |

incorrect

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |

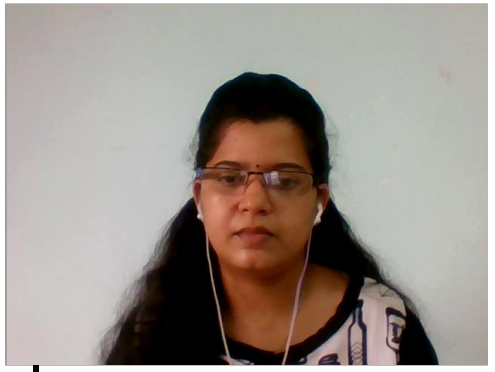
correct

| | | | |
|---|---|---|---|
| 0 | 1 | 1 | 1 |

correct

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |

# Grouping Rules



- Focus is on increasing the size of the group, so same elements can be repeated in multiple groups.

incorrect

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 |

correct

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 |

- Diagonal grouping is not permitted and elements around the edges can be grouped (circular property).

incorrect

| | |
|---|---|
| 1 | 0 |
| 0 | 1 |

correct

| | | | |
|---|---|---|---|
| 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 |

# Summary

- Introduction to K-map with the simplification procedures

# Reference


- M Morris Mano - Computer System Architecture - PHI - Third Edition
- Gideon Langholz, Abraha& Joe L Mott - Digital Logic Design - World Scientific Publishing Co Ltd
- Thomas C Bartee - Digital Computer Fundamentals - Tata Mc Graw Hill - Sixth Edition

AMRITA VISHWA VIDYAPEETHAM

AHEAD Online

# K-map simplification

Ms. Prathibha Prakash
Department of Computer Science
Amrita Vishwa Vidyapeetham

# Objective

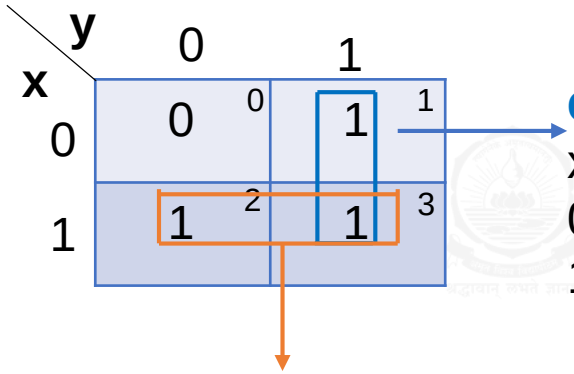
- K-map simplification examples

# K-map Simplification

- F(x,y) = x'y + xy' + xy (In SOP x' = 0; x =1)

01 10 11

**With this expression we need 3 AND gates and 1 OR gate**




**Group1**

x y

0 1 — y

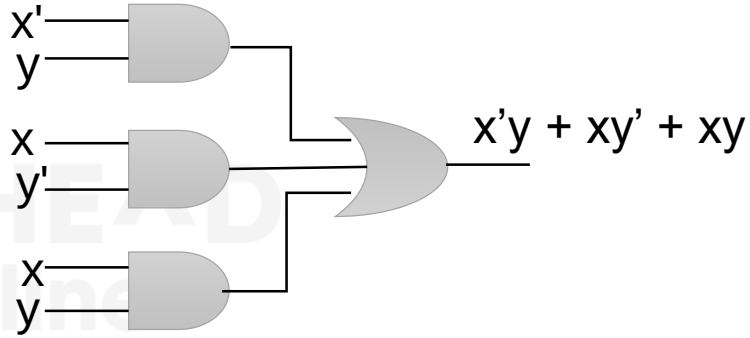
1 1 — Common term is y

**Group2**

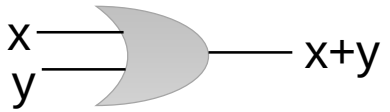
x y

1 0 — x

1 1 — Common term is x




Simplified expression, F(x, y) = x+y




**Simplified expression need only 1 OR gate**

# Example(SOP)

- Y (A,B,C,D) = ∑(0, 2, 5, 7, 8, 10, 13, 15)

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 (0) | 0 (1) | 0 (3) | 1 (2) |
| 01 | 0 (4) | 1 (5) | 1 (7) | 0 (6) |
| 11 | 0 (12) | 1 (13) | 1 (15) | 0 (14) |
| 10 | 1 (8) | 0 (9) | 0 (11) | 1 (10) |

Y = A'B'C'D' + A'B'CD' + A'BC'D + A'BCD + AB'C'D' + AB'CD' + ABC'D + ABCD

**Group1**

| A | **B** | C | **D** | |
|---|---|---|---|---|
| 0 | **1** | 0 | **1** | } **BD** |
| 0 | **1** | 1 | **1** | |
| 1 | **1** | 0 | **1** | |
| 1 | **1** | 1 | **1** | |

**Group2**

| A | **B** | C | **D** | |
|---|---|---|---|---|
| 0 | **0** | 0 | **0** | } **B'D'** |
| 0 | **0** | 1 | **0** | |
| 1 | **0** | 0 | **0** | |
| 1 | **0** | 1 | **0** | |

- Simplified expression Y(A,B,C,D) = BD + B'D'

# Example(POS)

- F(A,B,C,D) = ∑(0,1,2,5,8,9,10)
  Given SOP but for simplification using POS group the0's

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 0 | 1 |
| 01 | 0 | 1 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 1 |

**Group1**
A B **C D**
0 0 **1 1**
0 1 **1 1**
1 1 **1 1**
1 0 **1 1**
→ **C'+D'**

**Group2**
**A B** C D
**1 1** 0 0
**1 1** 0 1
**1 1** 1 1
**1 1** 1 0
→ **A'+B'**

**Group3**
A **B** C **D**
0 **1** 0 **0**
1 **1** 0 **0**
0 **1** 1 **0**
1 **1** 1 **0**
→ **B'+D**

- Simplified expression F(A,B,C,D) = (C'+D')(A'+B')(B'+D)

# Example(Don't care)

- Functions can have unspecified output for some input combination, we don't care the value of those unspecified terms.

➢ F(A,B,C,D) = ∑ (1,3,5,7,9) + d(6,12,13)

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 1 | 0 |
| 01 | 0 | 1 | 1 | X |
| 11 | X | X | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 |

**Group1**

| A | B | C | D | |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | A'D |
| 0 | 0 | 1 | 1 | |
| 0 | 1 | 0 | 1 | |
| 0 | 1 | 1 | 1 | |

**Group2**

| A | B | C | D | |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | C'D |
| 0 | 1 | 0 | 1 | |
| 1 | 1 | 0 | 1 | |
| 1 | 0 | 0 | 1 | |

**F(A,B,C,D) = A'D+C'D**

Without don't care
F(A,B,C,D) = A'D+B'C'D

# Summary

- Explained how to perform K-map simplification with examples

# Reference


- M Morris Mano - Computer System Architecture - PHI - Third Edition
- Gideon Langholz, Abraha& Joe L Mott - Digital Logic Design - World Scientific Publishing Co Ltd
- Thomas C Bartee - Digital Computer Fundamentals - Tata Mc Graw Hill - Sixth Edition