وزارة التعليم العالي والبحث العلمي

جامعة الانبار

كلية علوم الحاسوب وتكنولوجيا المعلومات

قسم علوم الحاسبات

Department	علوم الحاسبات	القسم:
Subject Name:	**Logic Design**	أسم المادة :
Year of Study:	**2025-2024**	السنة الدراسية:
Course:	الكورس الاول	الكورس:
Title and No of lecture:	**Lecture 10:COMBINATIONAL LOGIC Design**	عنوان ورقم المحاضرة:
Instructor Name:	د. مصطفى معد حمدي	أسم التدريسي:

LECTURE TEN
COMBINATIONAL LOGIC DESIGN

Objectives:

1. **Design procedure.**
2. **Fundamental circuits.**

1. Design procedure

- Design procedure has five steps:
  - *Specification.*
  - *Formulation.*
  - *Optimization.*
  - *Technology mapping.*
  - *Verification.*

- **Specification:**
  The design of a combinational circuit starts with the specification of the problem:
  Write a specification for the given circuit (text or HDL description (hardware description language)) with symbols for inputs and outputs.
- **Formulation:**
  Derive the truth table or initial Boolean expressions (that define the required relationships between inputs and outputs).
  Formulation converts the specification into forms that can be optimized (truth table or Boolean expression).
- **Optimization:**
  - ➢ Any available methods to minimize the logic:
    - ✓ **Algebraic manipulation.**
    - ✓ **K-map method.**
    - ✓ **Computer-based program.**

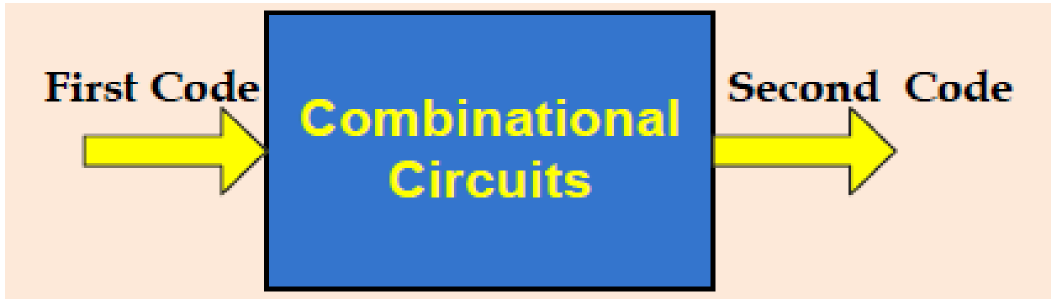
      Then, we can use:
    - ✓ **Two-level optimization or multiple-level optimization to get less cost (use NAND and NOR gate technologies).**
- **Technology mapping (implementation):**
  Transform the logic diagram to new logic diagram with available implementation.
- **Verification:**
  Verify the correctness of the final design.

2. Fundamental circuits

- These blocks are useful for designing large digital system, for example:
  - **Code converters.**
  - **Adders.**
  - **Multiplexers.**
  - **Decoders.**
  - **Encoders and so on.**

Code converters

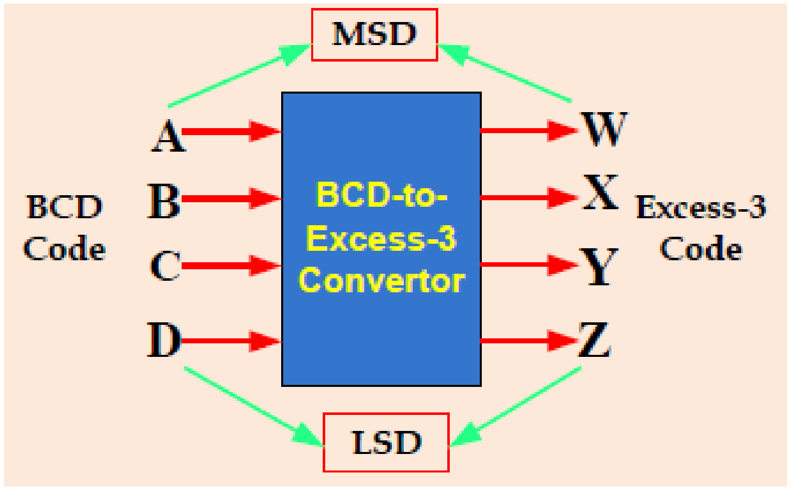
- Translate information from one binary code to another.
  - **BCD to Excess-3 converter.**
  - **BCD to seven-segment code converter.**
  - **BCD to Gray code converter.**



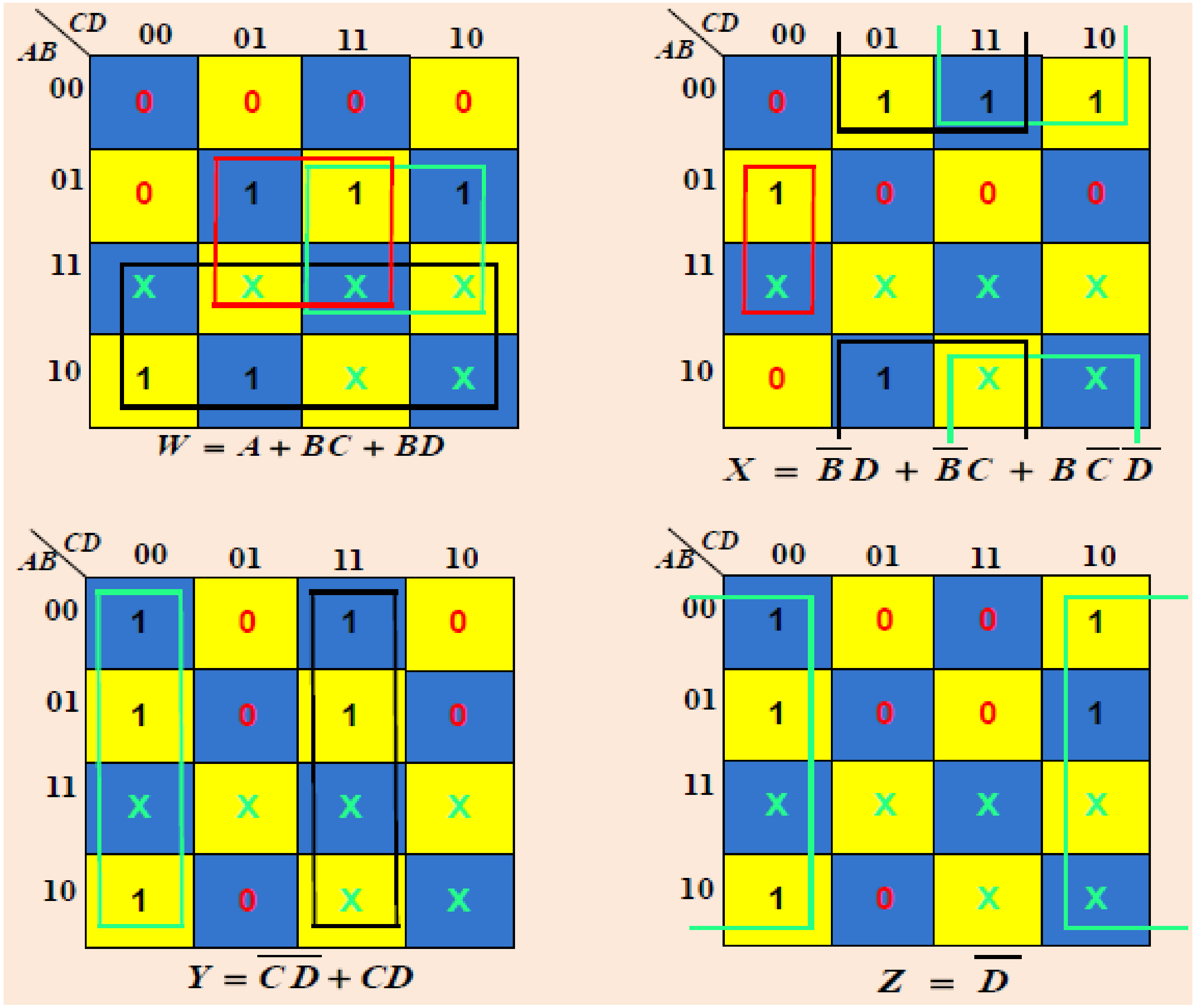
Example 1: Design of a BCD-to-Excess-3 code converter

- **Specification**: the excess-3 code for a decimal digit is *binary combination corresponding to the* ***decimal digit plus 3***.
- **Formulation**: the excess-3 code is easily obtained from BCD code by *adding binary* ***0011*** *to it*. The truth table relating the input and output values is the following:

Decimal digit	Input BCD code				Output Excess-3			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0
10	1	0	1	0	X	X	X	X
11	1	0	1	1	X	X	X	X
12	1	1	0	0	X	X	X	X
13	1	1	0	1	X	X	X	X
14	1	1	1	0	X	X	X	X
15	1	1	1	1	X	X	X	X



- **Optimization:**
  K-map for the outputs (four outputs) are shown, they are plotted to obtain simplified sum-of-products Boolean expressions for the outputs.



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

$$W = A + BC + BD$$

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

$$X = \overline{B}D + \overline{B}C + B\overline{C}\,\overline{D}$$

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

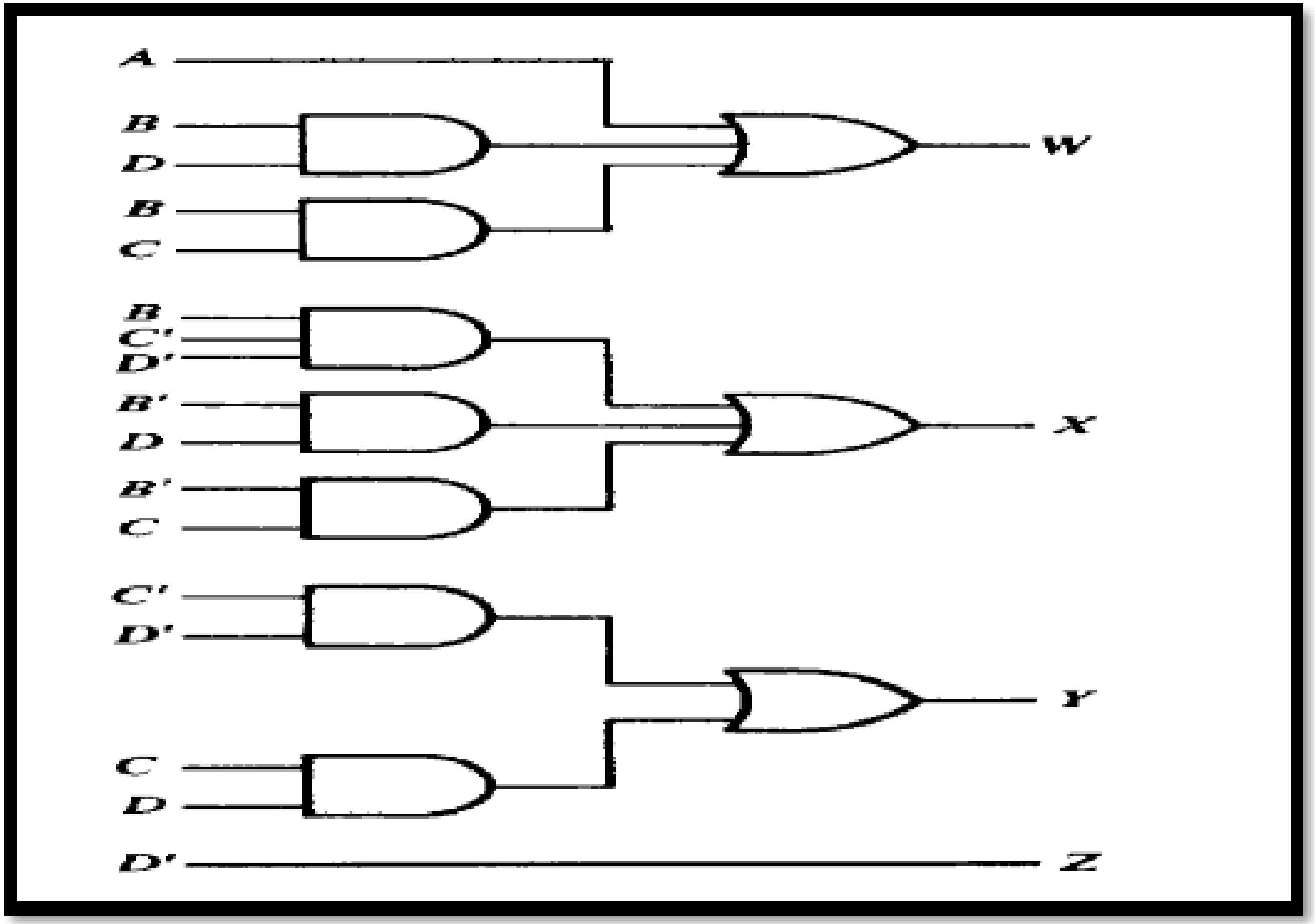
$$Y = \overline{C\,D} + CD$$

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

$$Z = \overline{D}$$

- ✓ The six don't care minterms, 10 through 15 are marked as ***X***.
- ✓ ***Two-level optimization*** (***AND-OR***) logic diagram for the circuit can be obtained directly from the Boolean expressions derived from the maps.

"Input gate cost = 26 including inverters"



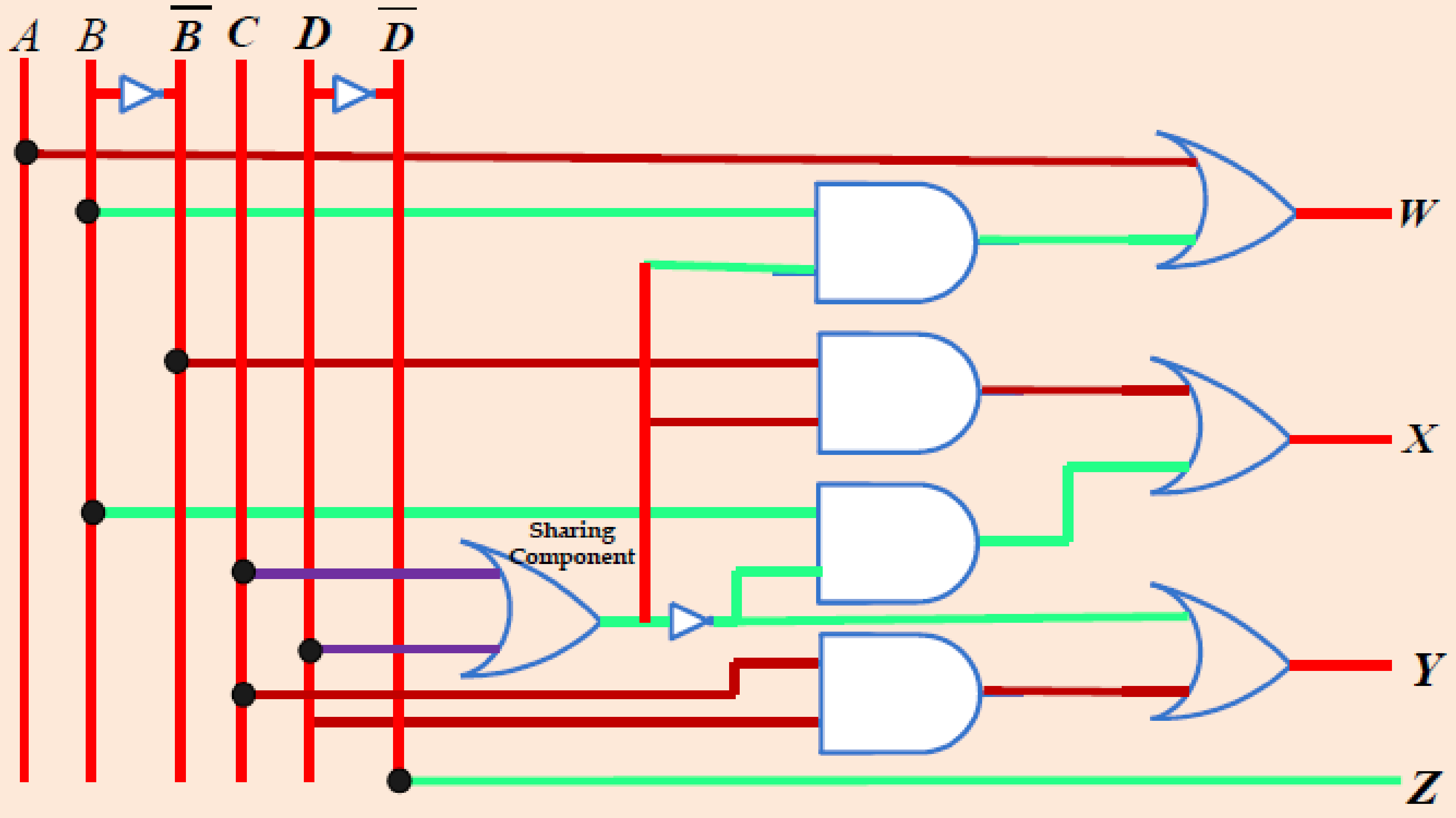
- ✓ We can reduce the input gate cost using ***multiple-level optimization*** as a second optimization step.
    - In this step, we consider the ***sharing sub expressions*** between the four output expressions.
    - ***Sharing expression***:

$$\boldsymbol{T_1 = C + D}$$
$$\boldsymbol{W = A + BC + BD = A + BT_1}$$
$$\boldsymbol{X = \overline{B}C + \overline{B}D + B\overline{C}\overline{D} = \overline{B}T_1 + B\overline{T_1}}$$
$$\boldsymbol{Y = CD + \overline{T_1}}$$
$$\boldsymbol{Z = \overline{D}}$$

    - The manipulation allows to reduce the gate input cost from *26* to *19*.

- **Technology mapping:**

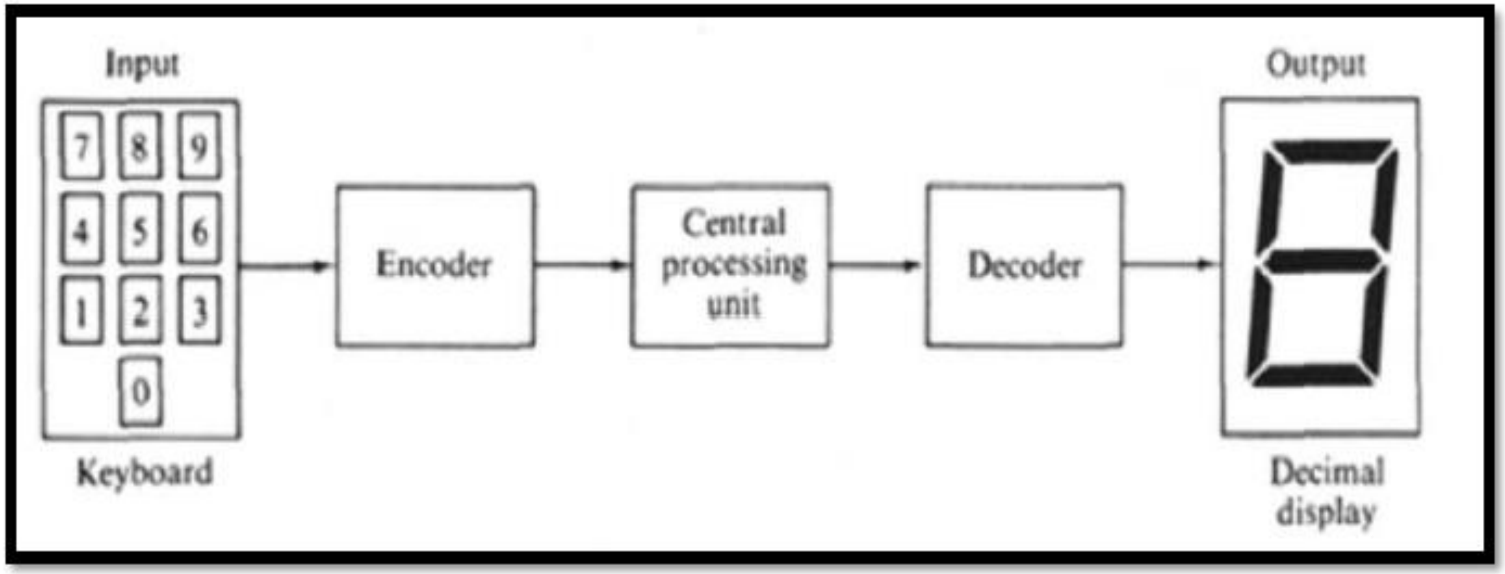
The logic diagram is the following:

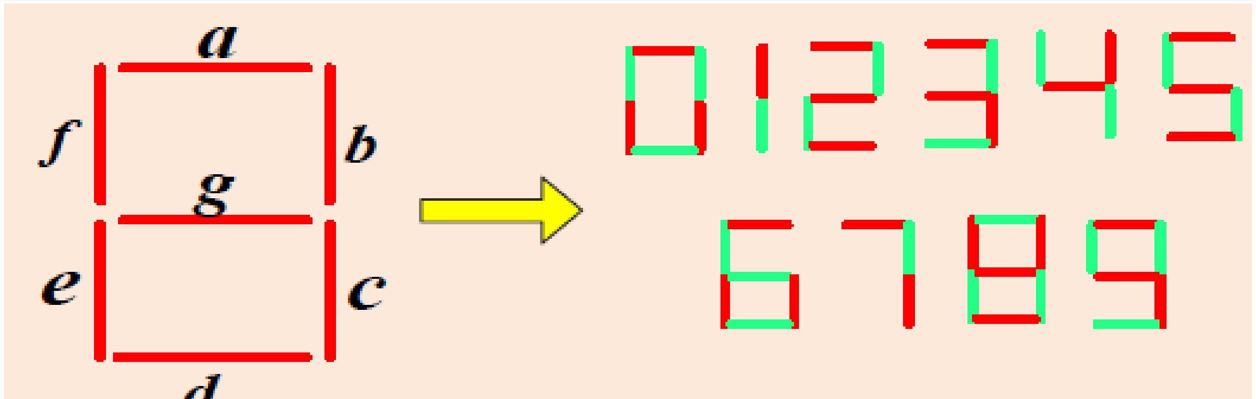


Logic Diagram for BCD-to-Excess-3 Converter

Example 2: Design of a BCD-to-seven-segment Decoder

- **Specification:**
  - ✓ **BCD-to-seven segment decoder** is a combinational circuit that
    - Accepts a decimal digit in ***BCD*** and generates the appropriate output of the decoder: ***(a, b, c, d, e, f, g)*** segments.
    - selects the corresponding segments in the ***LED*** display (***light-emitting diodes***) as shown in figure:





Seven-Segment Display (LED)

- **Formulation:**

➢ The truth table for BCD-to-seven segment decoder is the following:

BCD Input				Seven-Segment Outputs						
A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
All other inputs				0	0	0	0	0	0	0

- We must draw for each output Karnaugh map and minimize all maps.
- The simplified outputs:

$$a = \bar{A}C + \bar{A}BD + \bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}$$
$$b = \bar{A}\bar{B} + \bar{A}\bar{C}\bar{D} + \bar{A}CD + A\bar{B}\bar{C}$$
$$c = \bar{A}B + \bar{A}D + \bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}$$
$$d = \bar{A}C\bar{D} + \bar{A}\bar{B}C + \bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C} + \bar{A}B\bar{C}D$$
$$e = \bar{A}C\bar{D} + \bar{B}\bar{C}\bar{D}$$
$$f = \bar{A}B\bar{C} + \bar{A}\bar{C}\bar{D} + \bar{A}B\bar{D} + A\bar{B}\bar{C}$$
$$g = \bar{A}C\bar{D} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C}$$

✓ ***Two-level implementation:***

27 AND gates and 7 OR gates

- ***Multiple-level implementation:***

14 AND gates

Using sharing terms: $A\bar{B}\bar{C}$, $\bar{B}\bar{C}\bar{D}$ and so on

Example 3: Binary-to-Gray Converter

1. Truth tables for outputs: Gray Code

Decimal number	Binary input				Gray outputs			
	B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

2. K-maps:

$B_1B_0 \backslash B_3B_2$	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$$G_0 = B_0\overline{B_1} + \overline{B_0}B_1 = B_0 \oplus B_1$$

$B_1B_0 \backslash B_3B_2$	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	1	0	0	1
10	1	0	0	1

$$G_1 = B_1\overline{B_2} + \overline{B_1}B_2 = B_1 \oplus B_2$$

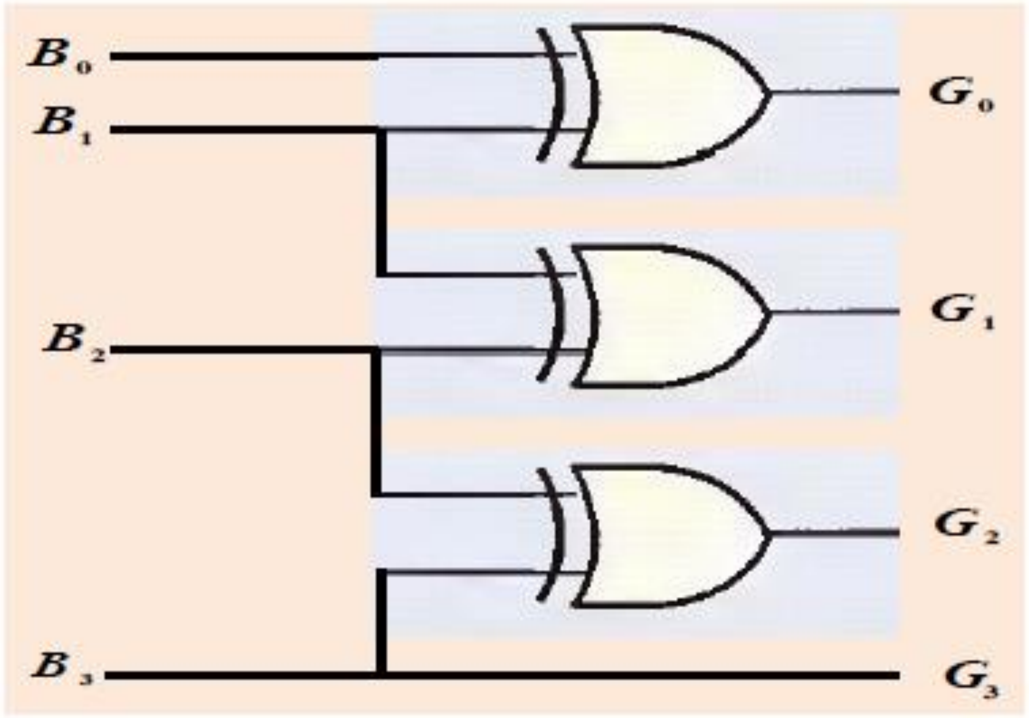
$B_1B_0 \backslash B_3B_2$	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$$G_2 = B_2\overline{B_3} + \overline{B_2}B_3 = B_2 \oplus B_3$$

$B_1B_0 \backslash B_3B_2$	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

$$G_3 = B_3$$

3. Logic Diagram:

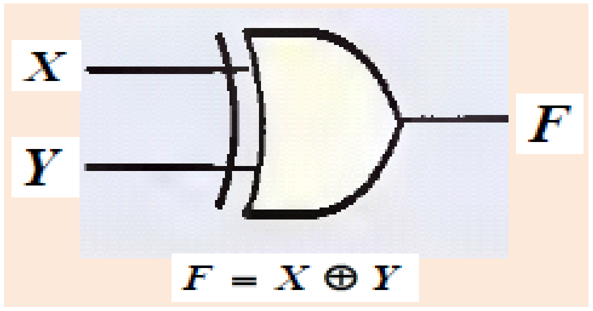


Logic diagram for binary-to-gray converter

- ***Exclusive-OR-operator** (**XOR Gate**).*

$$F = X\overline{Y} + \overline{X}Y = X \oplus Y$$

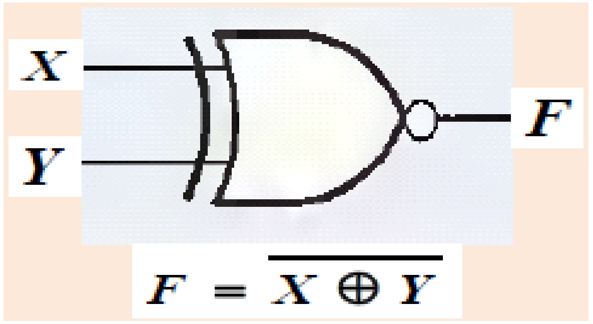
Truth Table		
Inputs		**Output**
X	Y	$F = X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0



- ***Exclusive-NOR-operator** (**XNOR Gate**).*

$$F = XY + \overline{X}\,\overline{Y} = \overline{X \oplus Y}$$

Truth Table		
Inputs		**Output**
X	Y	$F = \overline{X \oplus Y}$
0	0	1
0	1	0
1	0	0
1	1	1


X
Y
F
F = $\overline{X \oplus Y}$