Combinational Logic Design

objectives :

1) Design Procedure
2) fundamental Circuits

1) Design Procedure

Design procedure has five steps :

1- specification 2- formulation 3) optimization
4- technology mapping 5- 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

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 .

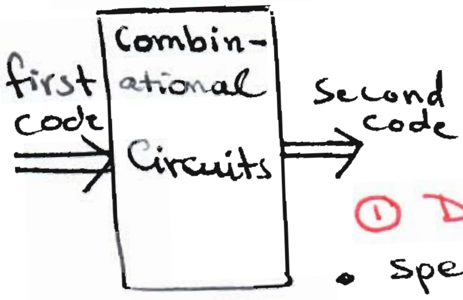
2 fundamental Circuits:

these blocks are useful for desinging large digital system, for example

- Code converters
- Adders
- Multiplexors
- Decoders
- Encoders and so on

• Code Converters:

Translate information from one binary code to another.



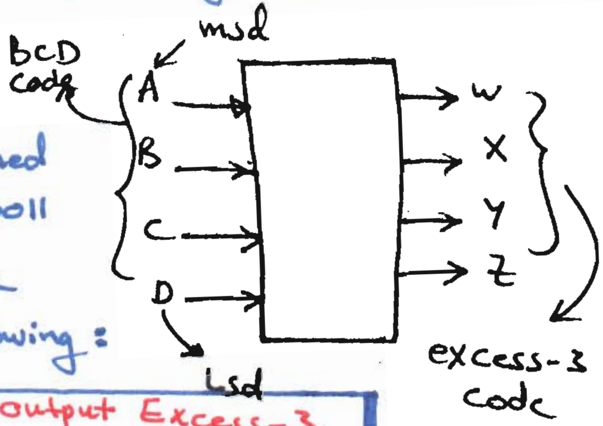
- BCD to Excess-3 Converter
- BCD To seven-segment code converter
- BCD To Gray code Converter.

1 Design of a BCD-to-Excess-3 Code converter

• specification: The excess-3 code for a decimal digit is the binary combination corresponding to the decimal digit plus $\underline{3}$.

• 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:

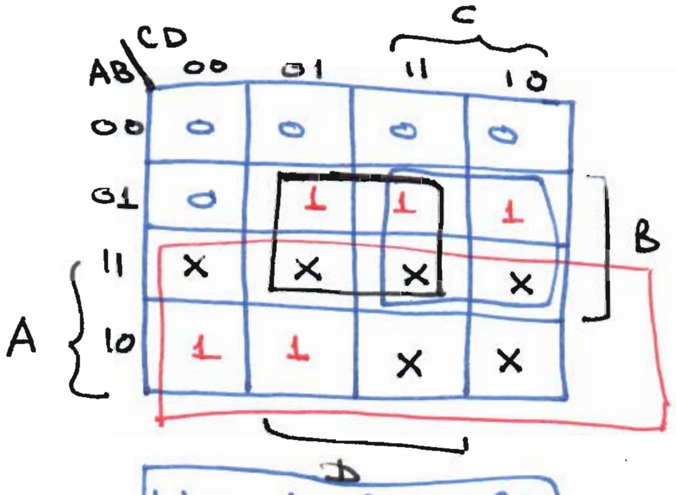


Decimal digit	Input (8421) code BCD code				output Excess-3			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0
10	1	0	1	0	X	X	X	X
11	1	0	1	1	X	X	X	X
12	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

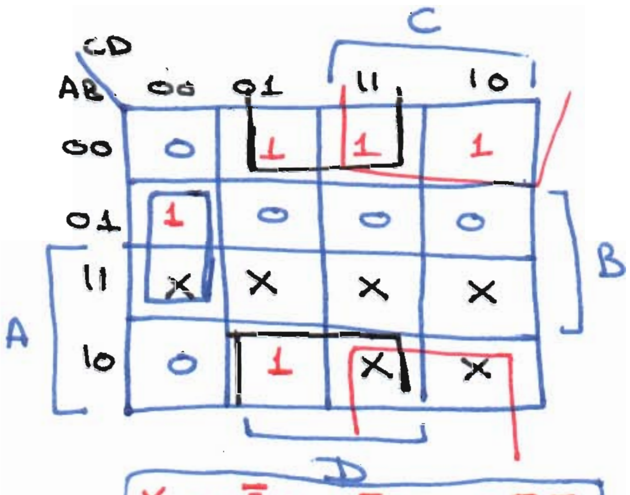
Don't care conditions.

- optimization :

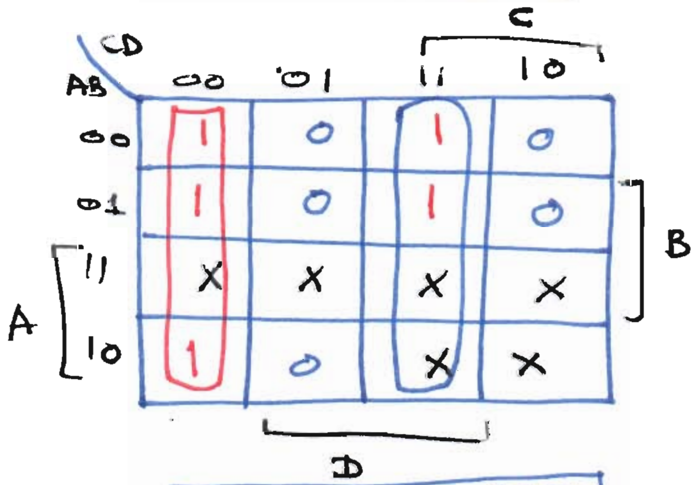
K-map for the outputs (four outputs) are shown, they are plotted to obtain simplified sum-of-products Boolean expressions for the outputs.



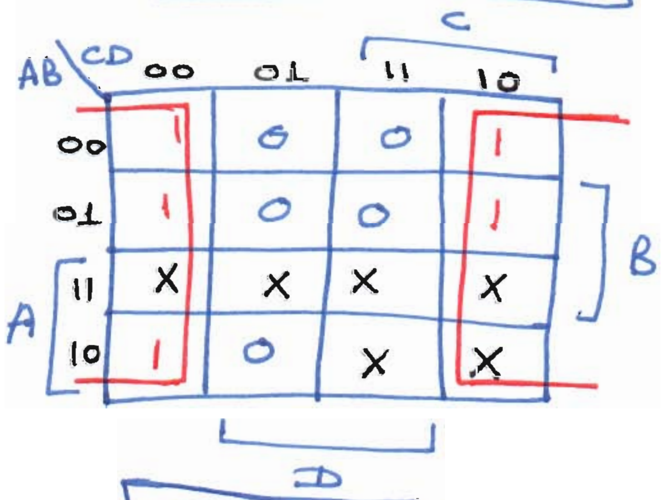
$\boxed{W = A + BC + BD}$



$\boxed{X = \bar{B}C + \bar{B}D + B\bar{C}\bar{D}}$



$\boxed{Y = CD + \bar{C}\bar{D}}$



$\boxed{Z = \bar{D}}$

- 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 subexpressions between the four output expressions.

- sharing expression:

$T_1 = C + D$

$W = A + BC + BD = A + BT_1$

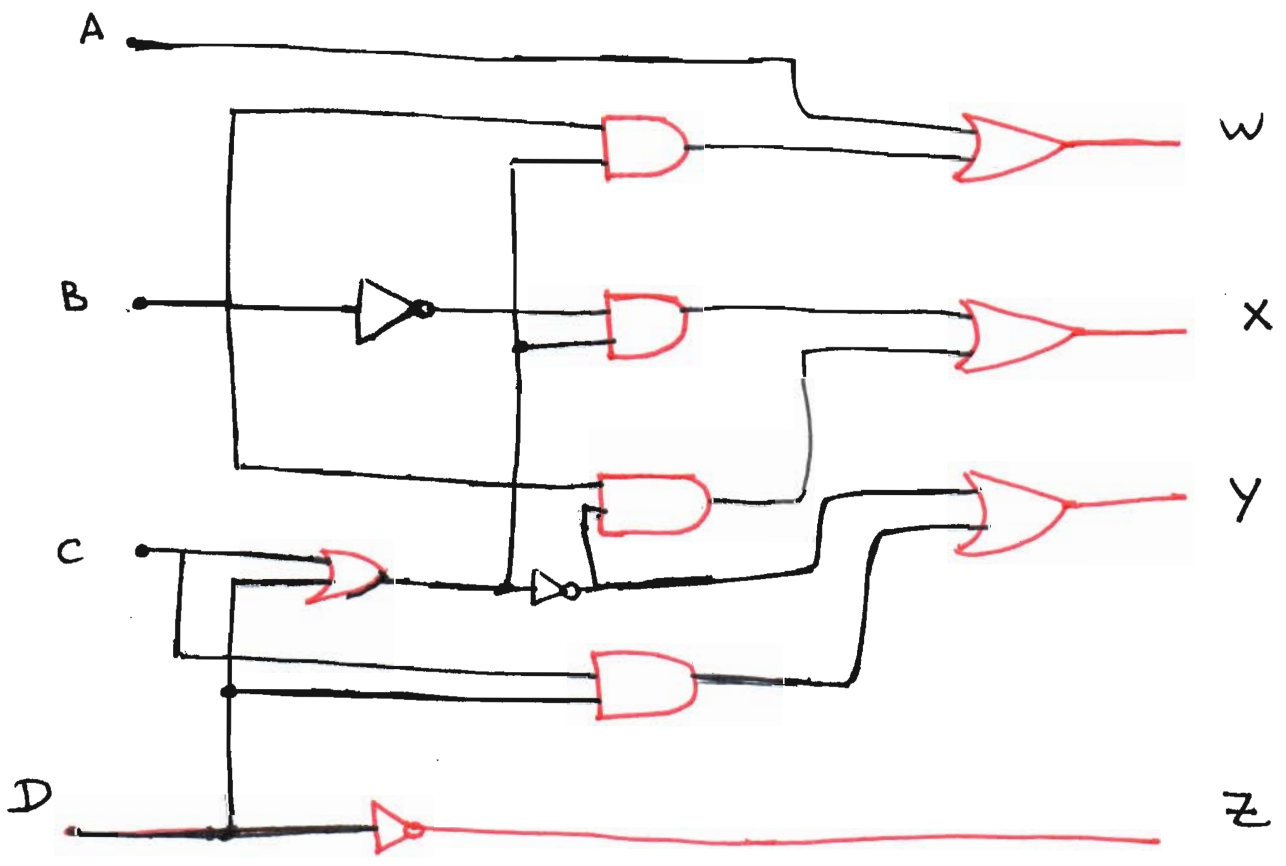
$X = \bar{B}C + \bar{B}D + B\bar{C}\bar{D} = \bar{B}T_1 + B\overline{T_1}$

$Y = CD + \overline{T_1}$

$Z = \bar{D}$

The manipulation allows to reduce the gate input cost from 26 to $\underline{\underline{19}}$.

The logic diagram is the following.

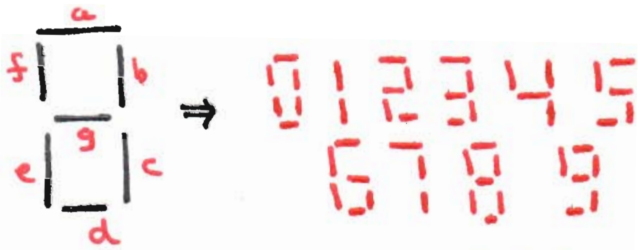


Logic Diagram for BCD-To-Excess-3 Converter

2. Design of a BCD-to-Seven-Segment Decoder

- **Specification:**

BCD-to-Seven Segment decoder is a combinational Circuit that accept a decimal digit in BCD and generates the appropriate outputs of the decoder (a, b, c, d, e, f, g) select the corresponding segments in the LED display (light-emitting diodes) as shown in figure:



Seven-Segment Display

- **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:

Better than Don't care conditions: (because of meaningless displays

$a = \bar{A}C + \bar{A}BD + \underline{\bar{B}\bar{C}\bar{D}} + \underline{A\bar{B}\bar{C}}$

$b = \bar{A}\bar{B} + \bar{A}\bar{C}\bar{D} + \bar{A}CD + \underline{A\bar{B}\bar{C}}$

$c = \bar{A}B + \bar{A}D + \underline{\bar{B}\bar{C}\bar{D}} + \underline{A\bar{B}\bar{C}}$

$d = \bar{A}C\bar{D} + \bar{A}\bar{B}C + \underline{\bar{B}\bar{C}\bar{D}} + \underline{A\bar{B}\bar{C}} + \bar{A}B\bar{C}D$

$e = \bar{A}C\bar{D} + \underline{\bar{B}\bar{C}\bar{D}}$

$f = \bar{A}B\bar{C} + \bar{A}\bar{C}\bar{D} + \bar{A}B\bar{D} + \underline{A\bar{B}\bar{C}}$

$g = \bar{A}C\bar{D} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + \underline{A\bar{B}\bar{C}}$

- Two-level implementation: ⇒ 27 AND gates and 7 OR gates
- multiple-level implem 14 AND gates using sharing terms: $A\bar{B}\bar{C}$, $\bar{B}\bar{C}\bar{D}$ and so on

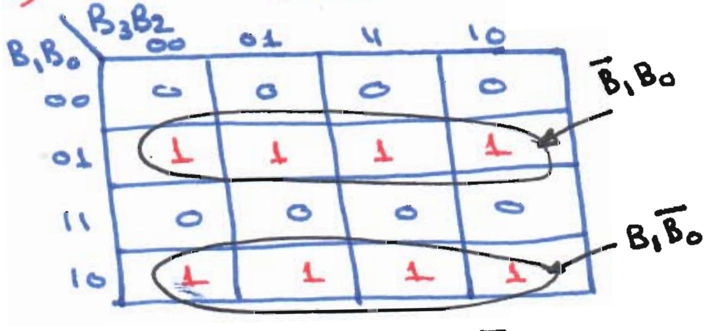
3. Binary-to-Gray converter.

1)

decimal number	Binary Input B_3	B_2	B_1	B_0	Gray outputs G_3	G_2	G_1	G_0
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

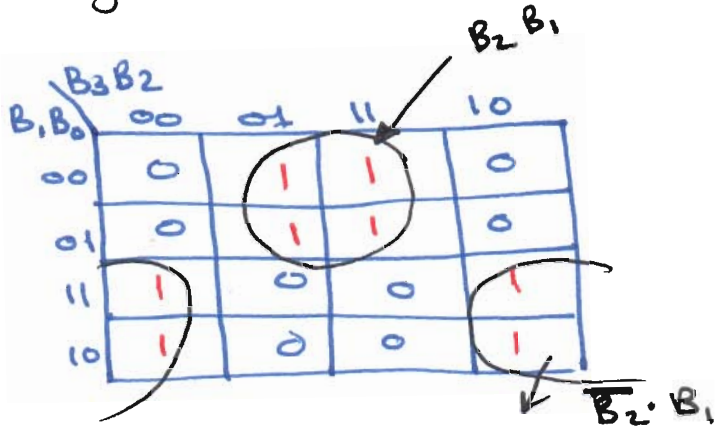
Truth tables ~~(~~ for outputs: Gray Code.

2) K-maps:



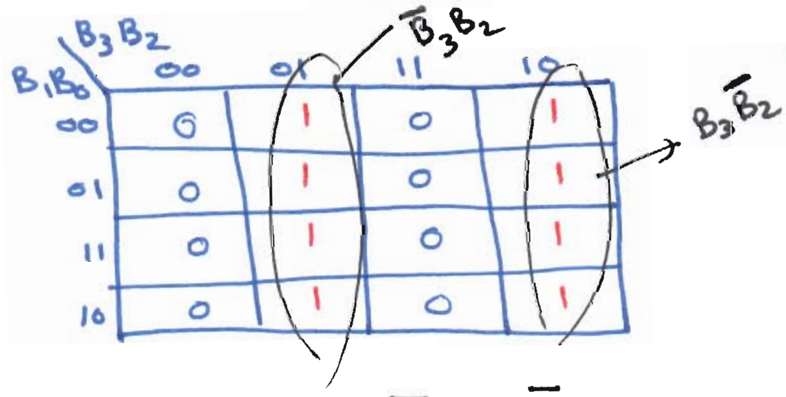
B_1B_0 \ B_3B_2	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$G_0 = B_0\overline{B}_1 + \overline{B}_0 B_1$
$= B_0 \oplus B_1$

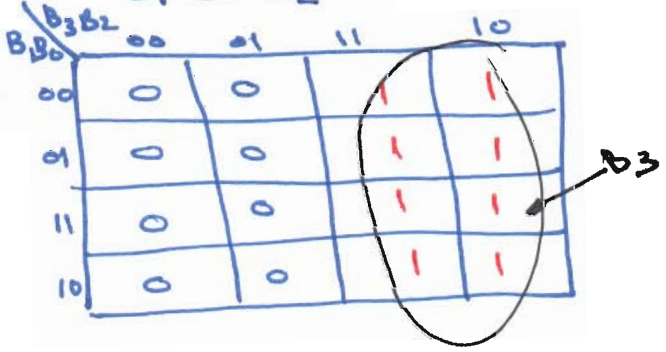


B_1B_0 \ B_3B_2	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	1	0	0	1
10	1	0	0	1

$G_1 = B_1\overline{B}_2 + \overline{B}_1 B_2 =$
$B_1 \oplus B_2$



B_1B_0 \ 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

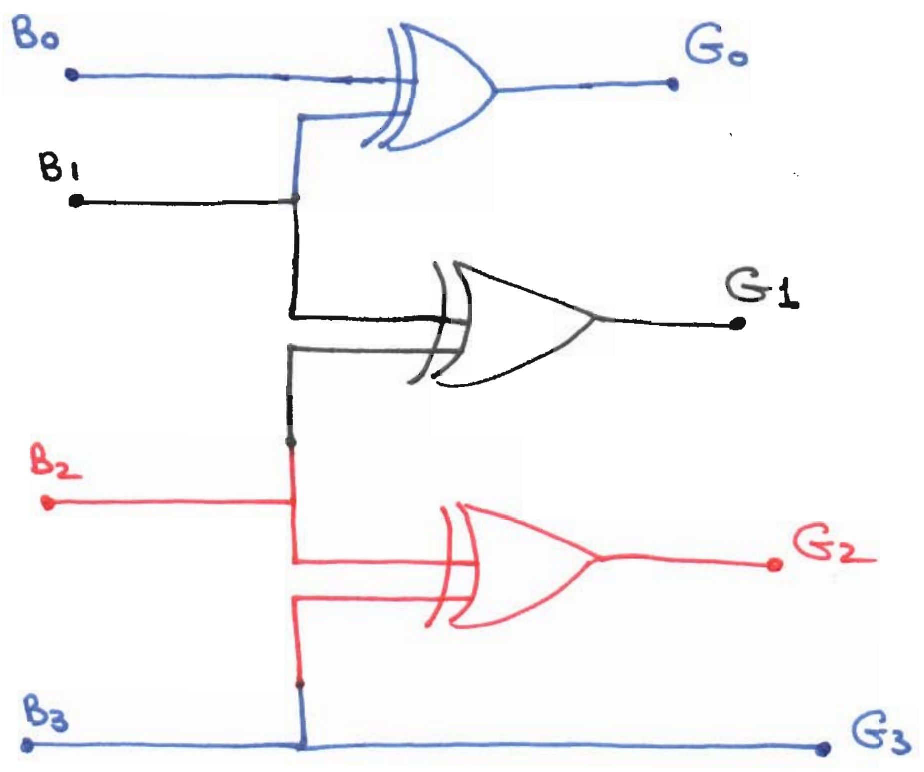


B_1B_0 \ B_3B_2	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

$G_2 = B_2\overline{B}_3 + \overline{B}_2 B_3 =$
$B_2 \oplus B_3$

$G_3 = B_3$

3) Logic Diagram :



" Logic Diagram for Binary-to-Gray Converter "

Important : $\oplus$ → called

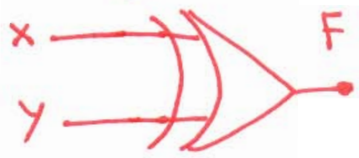
⊛ Exclusive-OR operator or Gate (XOR)

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

truth table

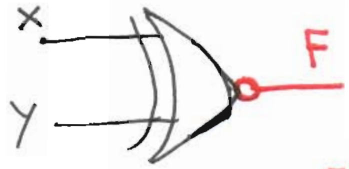
X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0

⇓



$F = X \oplus Y$

⊛ Exclusive - NOR operator or gate (XNOR gate)



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

X	Y	F
0	0	1
0	1	0
1	0	0
1	1	1