Binary Number				Gray Code			
a	b	c	d	w	x	y	z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

1. Design a combinational circuit that generates a 4-bit Gray-code of a 4-bit binary number.
(see the Truth Table on the right hand.)

(a) Express the output functions in sum-of-minterms form.
(b) Use K-maps to simplify each output function and compare with the (a).
(c) Use Boolean algebra to simplify each output function to a minimum number of literals.
(d) Implement the output functions with only NOR gates from the simplified expression.
(*Find Boolean expressions for implementing with NAND gates and Draw the logic diagram.)

2. Design a combinational circuit that compares two 3-bit numbers (A and B) to check if A is equal to B (F_1 (A = B)), or if A is greater than B (F_2 (A > B)), or if A is less than B (F_3 (A < B)).

(a) Obtain the SOP Table for only F1 =1. (*No required of the whole Truth Table.
(b) Obtain the Boolean functions F_1, F_2, and F_3 .
(c) Implement the function F_3 with only NAND gates. (*Find Boolean expression for implementing with NOR gates and Draw the logic diagram.)

1. Design a combinational circuit that generates a 4-bit Gray-code of a 4-bit binary number.

(a) Express the output functions in **sum-of-minterms form**.

$$\begin{cases} A = \sum(8,9,10,11,12,13,14,15) \\ B = \sum(4,5,6,7,8,9,10,11) \\ C = \sum(2,3,4,5,10,11,12,13) \\ D = \sum(1,2,5,6,9,10,13,14) \end{cases}$$

	Binary Number				Gray Code			
	W	X	Y	Z	A	B	C	D
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

1. Design a combinational circuit that generates a 4-bit Gray-code of a 4-bit binary number.

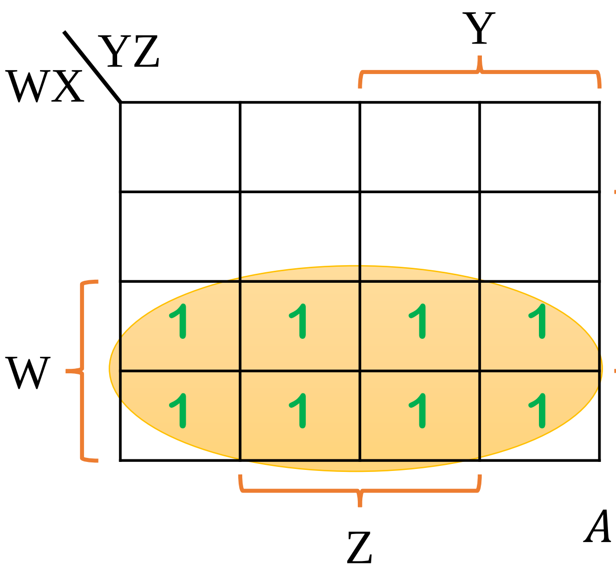
(b) Use K-map to **simplify each output function and compare them** with the answers from (a).

$$A = \sum(8,9,10,11,12,13,14,15)$$

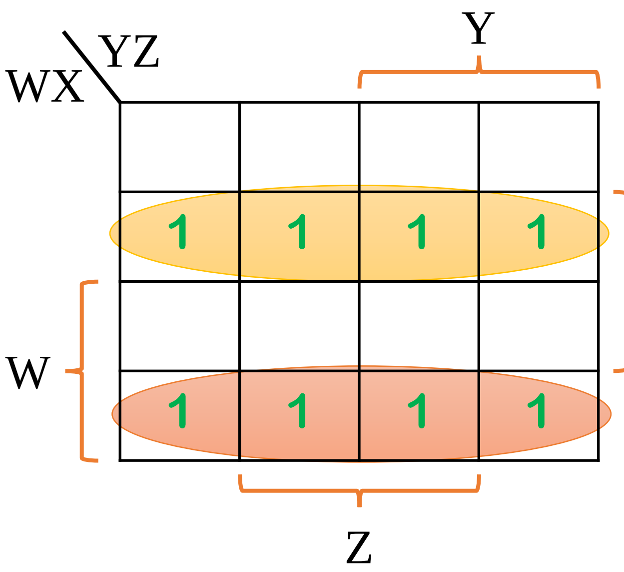
$$B = \sum(4,5,6,7,8,9,10,11)$$

$$C = \sum(2,3,4,5,10,11,12,13)$$

$$D = \sum(1,2,5,6,9,10,13,14)$$

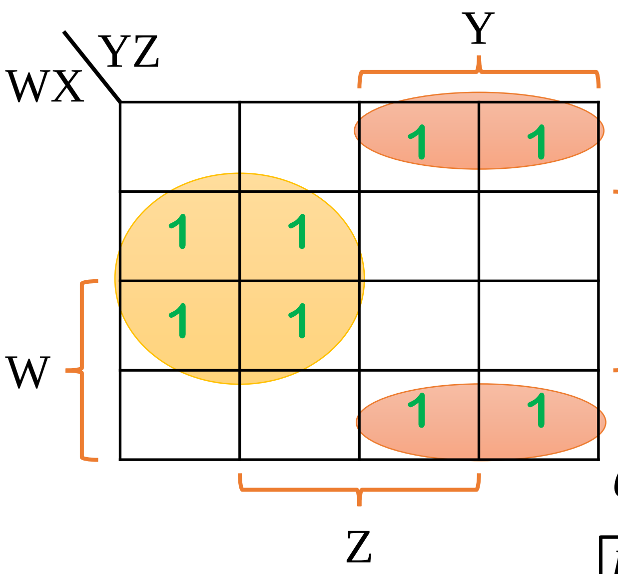


$A = W$



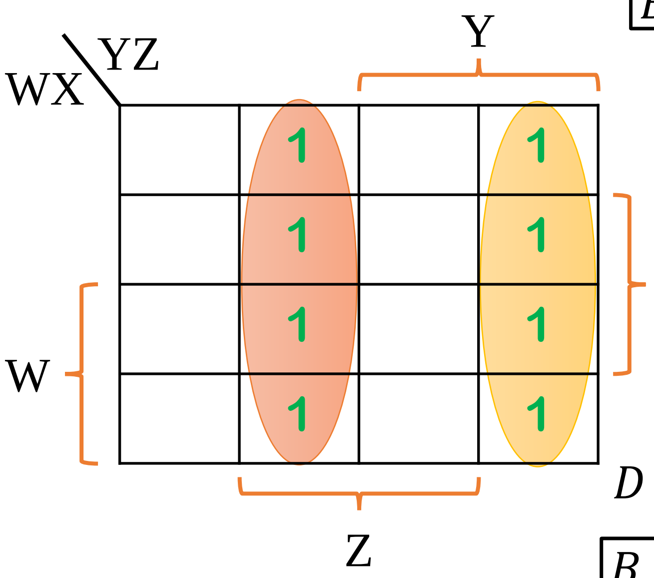
$B = W'X + WX'$

$\boxed{B = W \oplus X}$



$C = X'Y + XY'$

$\boxed{B = X \oplus Y}$



$D = Y'Z + YZ'$

$\boxed{B = Y \oplus Z}$

*** Indicators are important!**

*** Either one is okay!**

1. Design a combinational circuit that generates a 4-bit Gray-code of a 4-bit binary number.

(c) Use Boolean algebra to **simplify each output function** to a minimum number of literals.

$$(XOR\ Functiom)$$

$$A = WX'Y'Z' + WX'Y'Z + WX'YZ' + WX'YZ + WXY'Z' + WXY'Z + WXYZ' + WXYZ$$

$$= W(X'Y'Z' + X'Y'Z + X'YZ' + X'YZ + XY'Z' + XY'Z + XYZ' + XYZ)$$

$$= W$$

$$B = W'XY'Z' + W'XY'Z + W'XYZ' + W'XYZ + WX'Y'Z' + WX'Y'Z + WX'YZ' + WX'YZ$$

$$= W'XY'(Z'+Z) + W'XY(Z' + Z) + WX'Y'(Z' + Z) + WX'Y(Z' + Z)$$

or

$$= W'X(Y'Z' + Y'Z + YZ' + YZ) + WX'(Y'Z' + Y'Z + YZ' + YZ)$$

$$= W'X + WX$$

$$A = \sum(8,9,10,11,12,13,14,15)$$

$$B = \sum(4,5,6,7,8,9,10,11)$$

$$C = \sum(2,3,4,5,10,11,12,13)$$

$$D = \sum(1,2,5,6,9,10,13,14)$$

Same way as B:

$$C = X'Y + XY'$$

$$D = Y'Z + YZ'$$

1. Design a combinational circuit that generates a 4-bit Gray-code of a 4-bit binary number.

(d) **Implement the output functions with only NOR gates** from the simplified Boolean expressions.

$$A = W$$

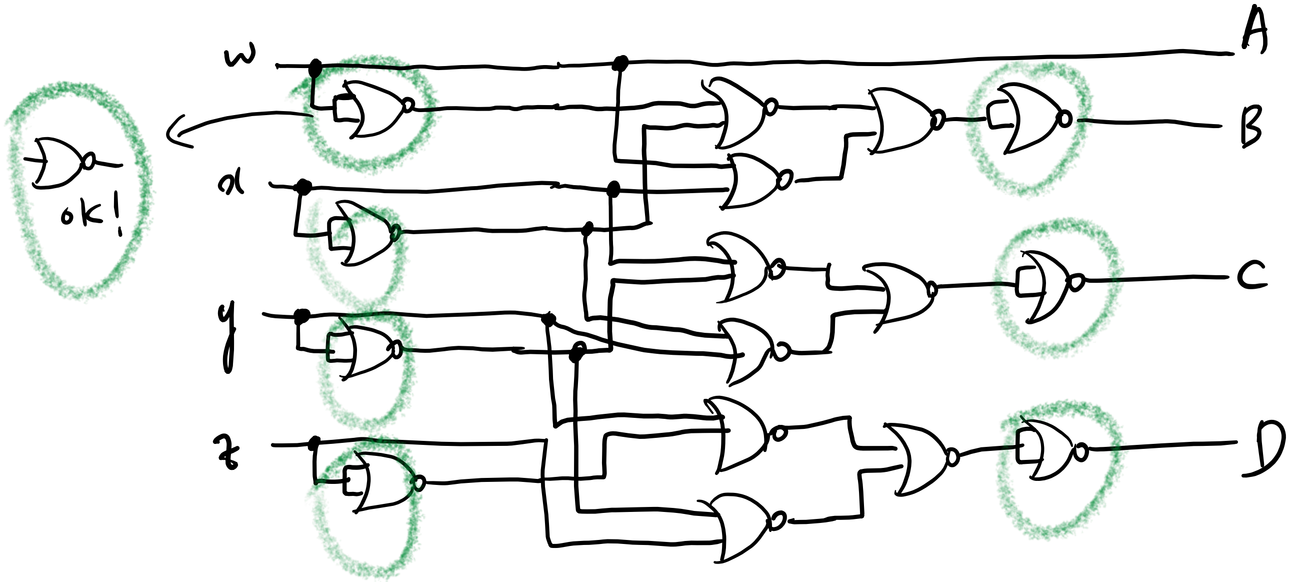
$$B = W'X + WX' \xrightarrow{Then} B' = (W'X)'.(WX')' = (W + X').(W' + X)$$

$$\xrightarrow{Then} (B')' = (W + X')' + (W' + X)'$$

$$\xrightarrow{Then} (B')' = \left[[(W + X')' + (W' + X)']'\right]'$$

$$C = X'Y + XY' = \left[[(X + Y')' + (X' + Y)']'\right]'$$

$$D = Y'Z + YZ' = \left[[(Y + Z')' + (Y' + Z)']'\right]'$$



*** W' , X' , Y' , Z' inputs are needed to implement with a single input NOR gate.**

2. Design a combinational circuit that compares **two 3-bit numbers (A and B)** to **check if** A is equal to B (**F_1 (A = B)**), or if A is greater than B (**F_2 (A > B)**), or if B is greater than A (**F_3 (A < B)**).

(a) Obtain the Truth Table for only F_1 =1. (* No required of the whole Truth Table.)

A_2	A_1	A_0	B_2	B_1	B_0	$F(A=B)$	
0	0	0	0	0	0	1	m_0
0	0	1	0	0	1	1	m_9
0	1	0	0	1	0	1	m_{18}
0	1	1	0	1	1	1	m_{27}
1	0	0	1	0	0	1	m_{36}
1	0	1	1	0	1	1	m_{45}
1	1	0	1	1	0	1	m_{54}
1	1	1	1	1	1	1	m_{63}

2. Design a combinational circuit that compares **two 3-bit numbers (A and B)** to **check if** A is equal to B (**F_1 (A = B)**), or if A is greater than B (**F_2 (A > B)**), or if B is greater than A (**F_3 (A < B)**).

(b) Obtain the **Boolean functions** F_1, F_2, and F_3.

$$A \qquad\qquad B$$

$$A_2\ A_1\ A_0 \qquad B_2\ B_1\ B_0$$

$$F_1 = (A_2'B_2' + A_2B_2).(A_1'B_1' + A_1B_1).(A_0'B_0' + A_0B_0)$$

$$F_2 = A_2B_2' + (A_2'B_2' + A_2B_2).A_1B_1' + (A_2'B_2' + A_2B_2).(A_1'B_1' + A_1B_1).A_0B_0'$$

$$F_3 = A_2'B_2 + (A_2'B_2' + A_2B_2).A_1'B_1 + (A_2'B_2' + A_2B_2).(A_1'B_1' + A_1B_1).A_0'B_0$$

$You\ can\ also\ define\ x_i\ as\ x_i = A_i'B_i' + A_iB_i\ \ So$:

$$F_1 = x_2.x_1.x_0$$

$$F_2 = A_2B_2' + x_2.A_1B_1' + x_2.x_1.A_0B_0'$$

$$F_3 = A_2'B_2 + x_2.A_1'B_1 + x_2.x_1.A_0'B_0$$

2. Design a combinational circuit that compares **two 3-bit numbers (A and B)** to **check if** A is equal to B (**F_1 (A = B)**), or if A is greater than B (**F_2 (A > B)**), or if B is greater than A (**F_3 (A < B)**).

(c) **Implement** the function F_3 with **only NAND gates**.

$$x_2 = A_2'B_2' + A_2B_2$$

$$x_1 = A_1'B_1' + A_1B_1$$

$$F_3 = A_2'B_2 + x_2.A_1'B_1 + x_2.x_1.\ A_0'B_0$$

$$(F_3)' = (A_2'B_2 + x_2.A_1'B_1 + x_2.x_1.\ A_0'B_0)'$$

$$= (A_2'B_2)'.(x_2.A_1'B_1)'.(x_2.x_1.\ A_0'B_0)'$$

$$((F_3)')' = F_3$$

$$= [(A_2'B_2)'.(x_2.A_1'B_1)'.(x_2.x_1.\ A_0'B_0)'\ ]'$$

$$*\ \ x_2' = (A_2'B_2' + A_2B_2)' = (A_2'B_2')'.(A_2B_2)'$$

$$((x_2)')' = x_2 = [(A_2'B_2')'.(A_2B_2)']'$$

Similarly: $x_1 = [(A_1'B_1')'.(A_1B_1)']'$