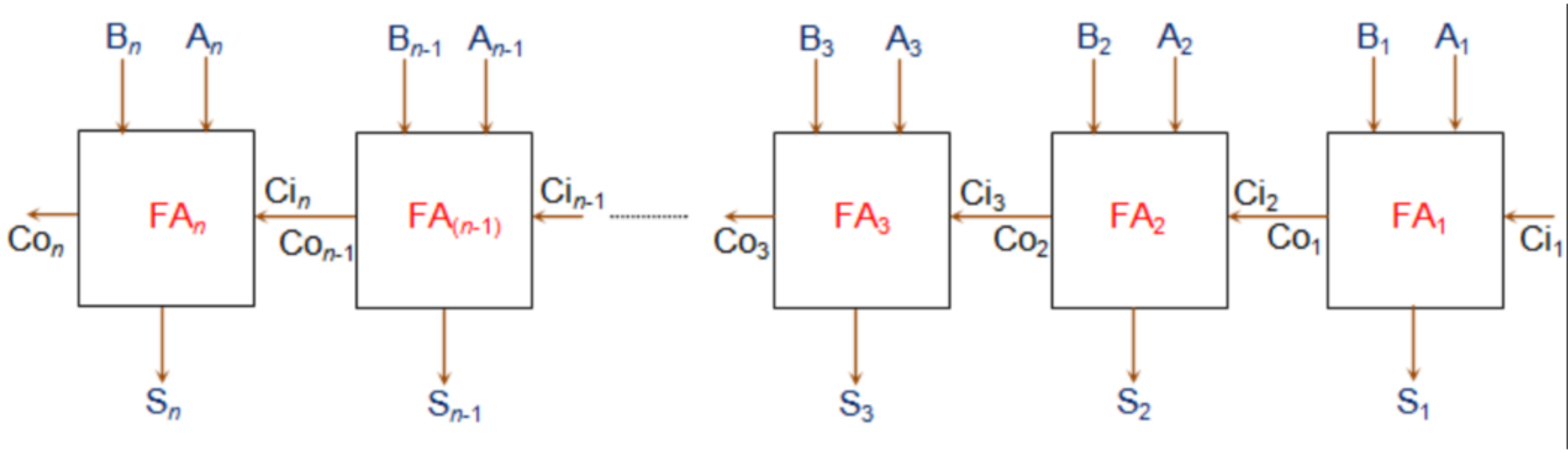
Parallel Adder Circuit:



'It is a chain of full adders Connected Parallely where the C_{in} of a adder is the C_{out} of Previous adder.'

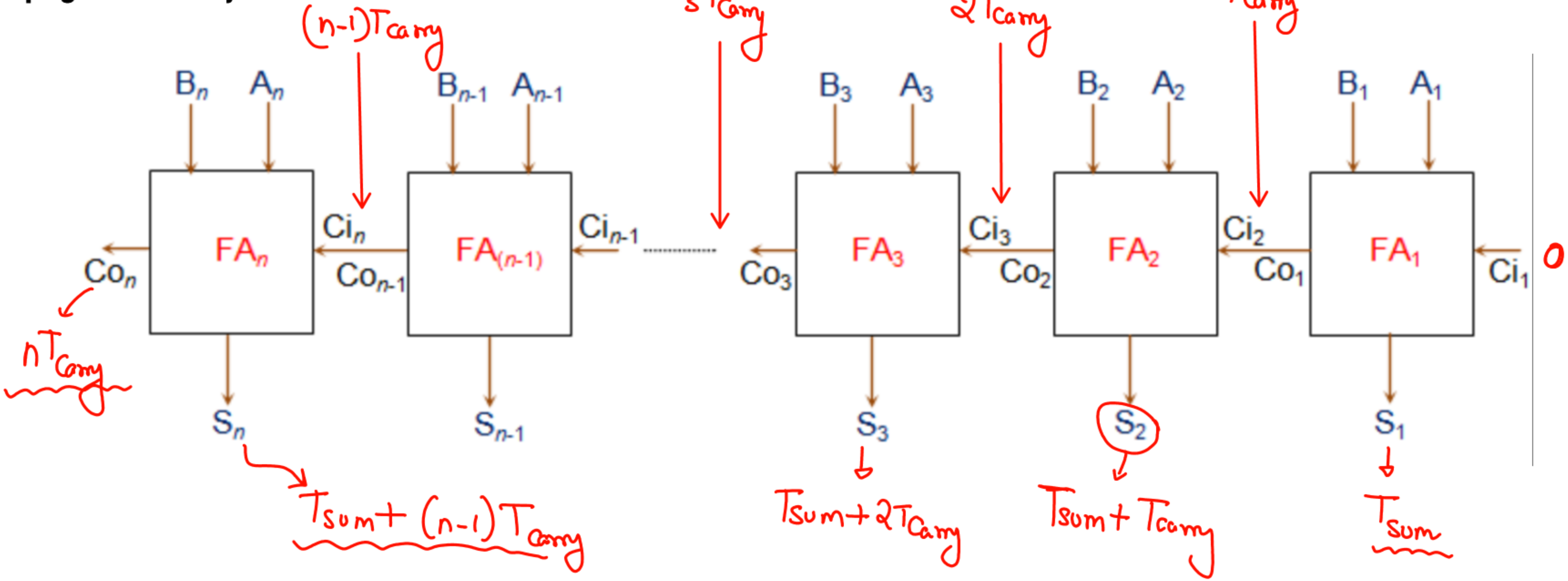
* It is also Called as ripple Carry adder.

	$C_{in}=1$	$C_{in}=1$	$C_{in}=0$		LSB
$A =$	1	0	1	0	$C_{in}=0$
$B =$	+ 1	1	1	0	
	Sum = 1	Sum = 0	Sum = 0	Sum = 0	
MSB	$C_{out}=1$	$C_{out}=1$	$C_{out}=1$	$C_{out}=0$	
	generate				

$(1\ 1\ 0\ 0\ 0)_2$

C_{out}

Propagation Delay in Parallel Adder



Suppose time taken by a fulladder
for Sum = T_{sum}
for Carry$_{out}$ = T_{carry}

$$\boxed{\begin{aligned} S_n &= T_{sum} + (n-1)T_{carry} \\ Co_n &= nT_{carry} \end{aligned}} \rightarrow \text{max}$$

Parallel Subtractor using full adder

A Parallel subtractor is a digital Circuit Capable to find the diff. b/w 2 or more than two bits.

$A - B \rightarrow \boxed{A + \overline{B} + 1}$

$= A + 2\text{'s Complement of } B$

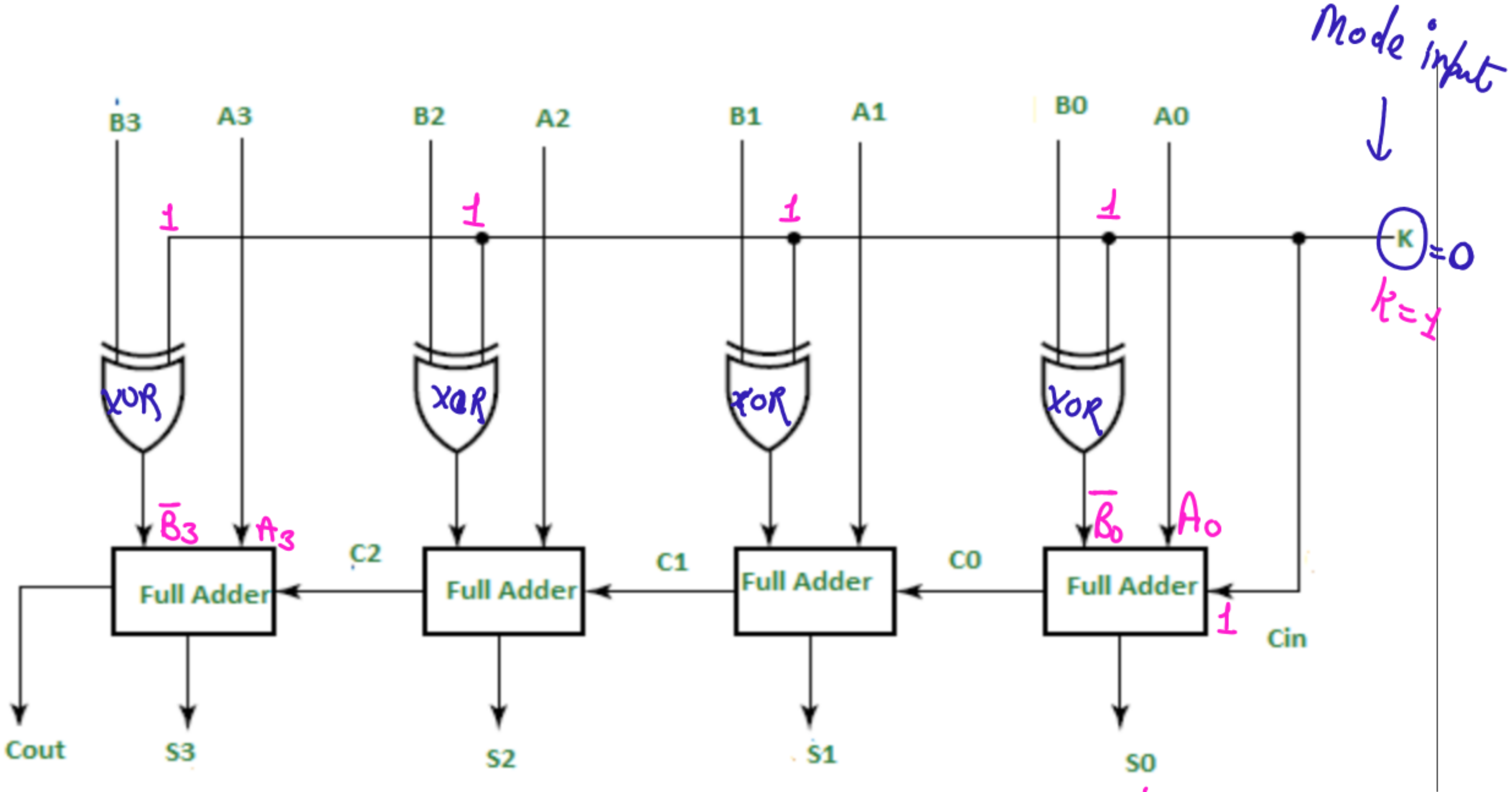
$= \underbrace{A + [1\text{'s Complement of } B] + 1}_{3 \text{ bits}}$

XOR Property

$B \oplus 0 = B$

$B \oplus 1 = \overline{B}$

Inverter



$S = A_0 + \overline{B_0} + 1$

If Mode Input (K) = 0 then it works as Parallel Adder.

Propagation delay in full adder:

if $A = B = 1$

$T_{x1} \Rightarrow$ No delay

$Y = T_{A2}$ but if $C_{in} = 0$, then No delay for sum

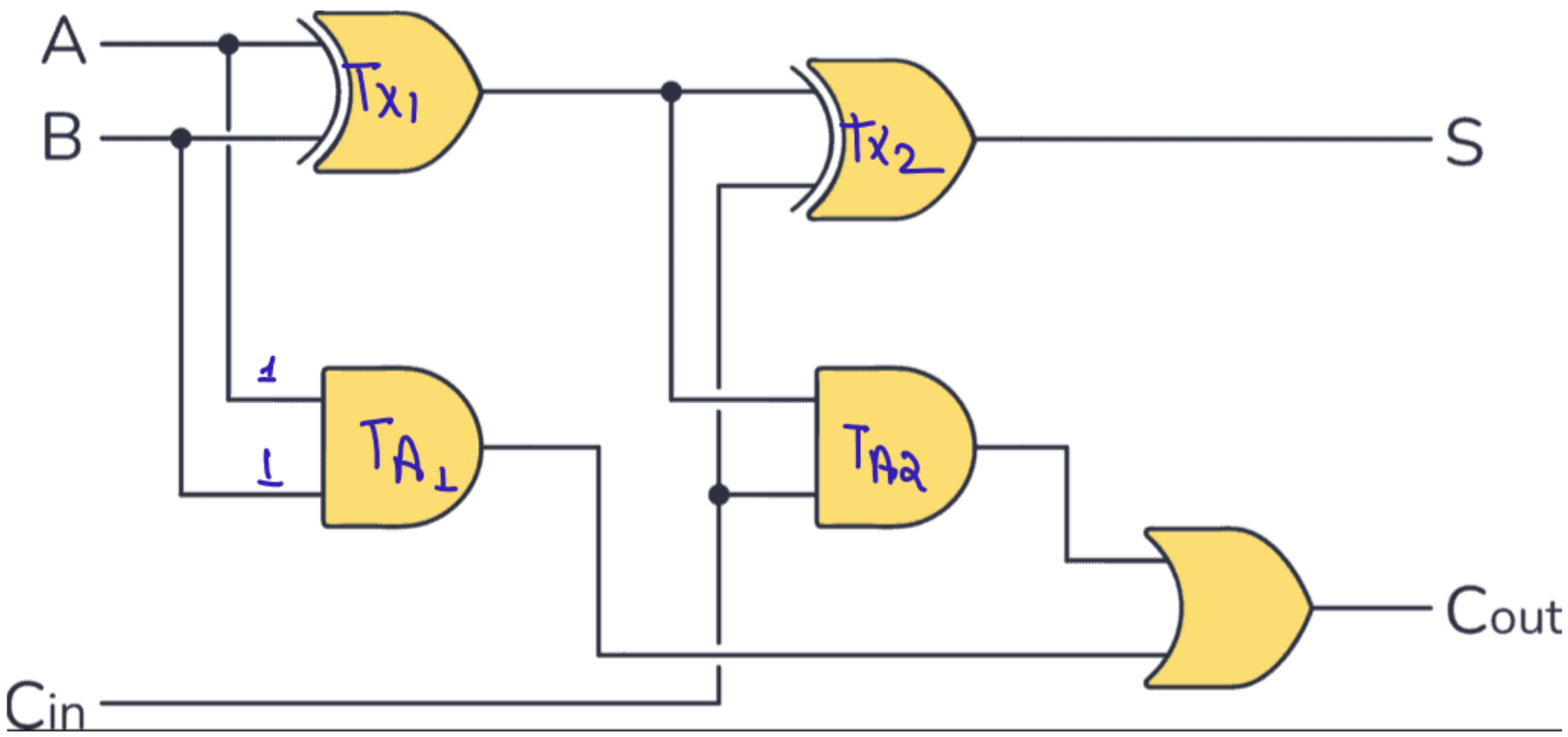
sum depends on Carry

else $C_{in} = 1$, then $\boxed{T_{Sum} = T_{x_2}}$

if $C_{in} = 0$, $z = 0$ (No delay)

$C_{in} = 1$, $z = 0$ (No delay)

$\boxed{C_{out} = T_{A1} + T_{OR}}$



if $A = 0$ & $B = 1$ OR $A = 1$ & $B = 0$

$y =$ No delay, if $C_{in} = 1$

$\Rightarrow$ Sum $= 0$ (no delay) $\longrightarrow$ $\boxed{C_{out} = T_{x1} + T_{A2} + T_{OR}}$

if $C_{in} = 0$

$\Rightarrow$ Sum $= [T_{x1} + T_{x2}]$

Overall delay:

$$\boxed{Sum = T_{X1} + T_{X2} + T_{C_{in}}}$$

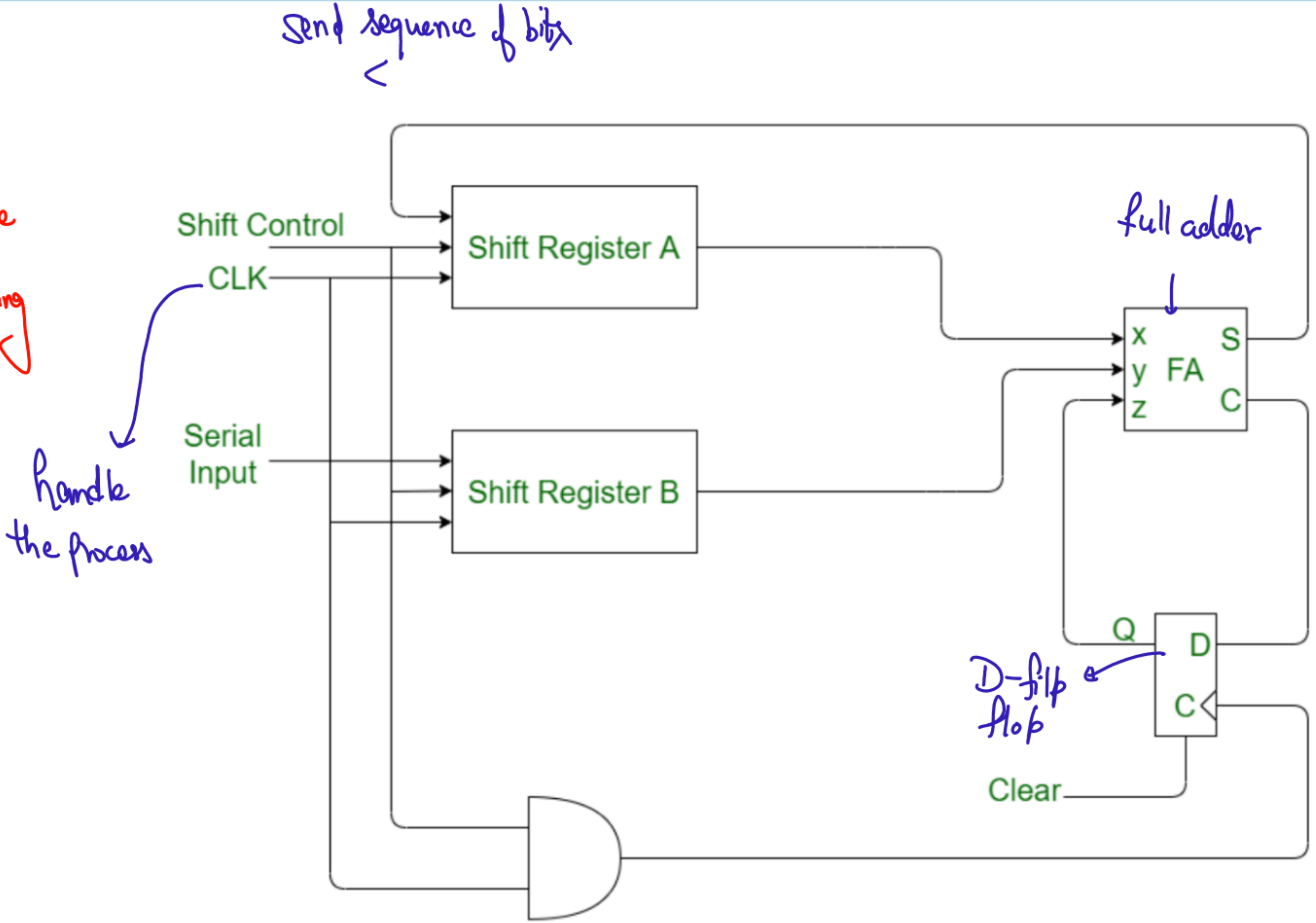
If inputs are identical then we will not add the delay of Carry input

$$\boxed{C_{out} = T_{X1} + T_{A2} + T_{OR}}$$

Serial Adder

* That adds upto 'n' bits by using single full adder by performing operation for 'n' times.

→ It's slower than Parallel adder



Code Converter

↳ A Code Converter is a circuit that Convert the Binary Code to other Codes.

Eg

BCD → XS-3

Excess-3 → BCD

BCD → Gray Code

BCD → 7-Segment Code.

Code Comparator

↓

Compare the I/P Values & gives the O/P.

Magnitude Comparator

↳ It is a digital circuit that compares two values
↓ compares → Binary (values)

- → less than
- ⇒ greater than
- ⇒ equals to

I/P

* here A & B are not only a single bit
It can be multiple bit number

Eg

A = 0	A = 10	A = 101
B = 1	B = 01	B = 111
Single bit	two digit	3 bit number

1 bit Magnitude Comparator

A & B holds Single Bit

	A	B	A<B	A=B	A>B
0	0	0	0	1 $\bar{A}\bar{B}$	0
1	0	1	1 $\bar{A}B$	0	0
2	1	0	0	0	1 $A\bar{B}$
3	1	1	0	1 AB	0

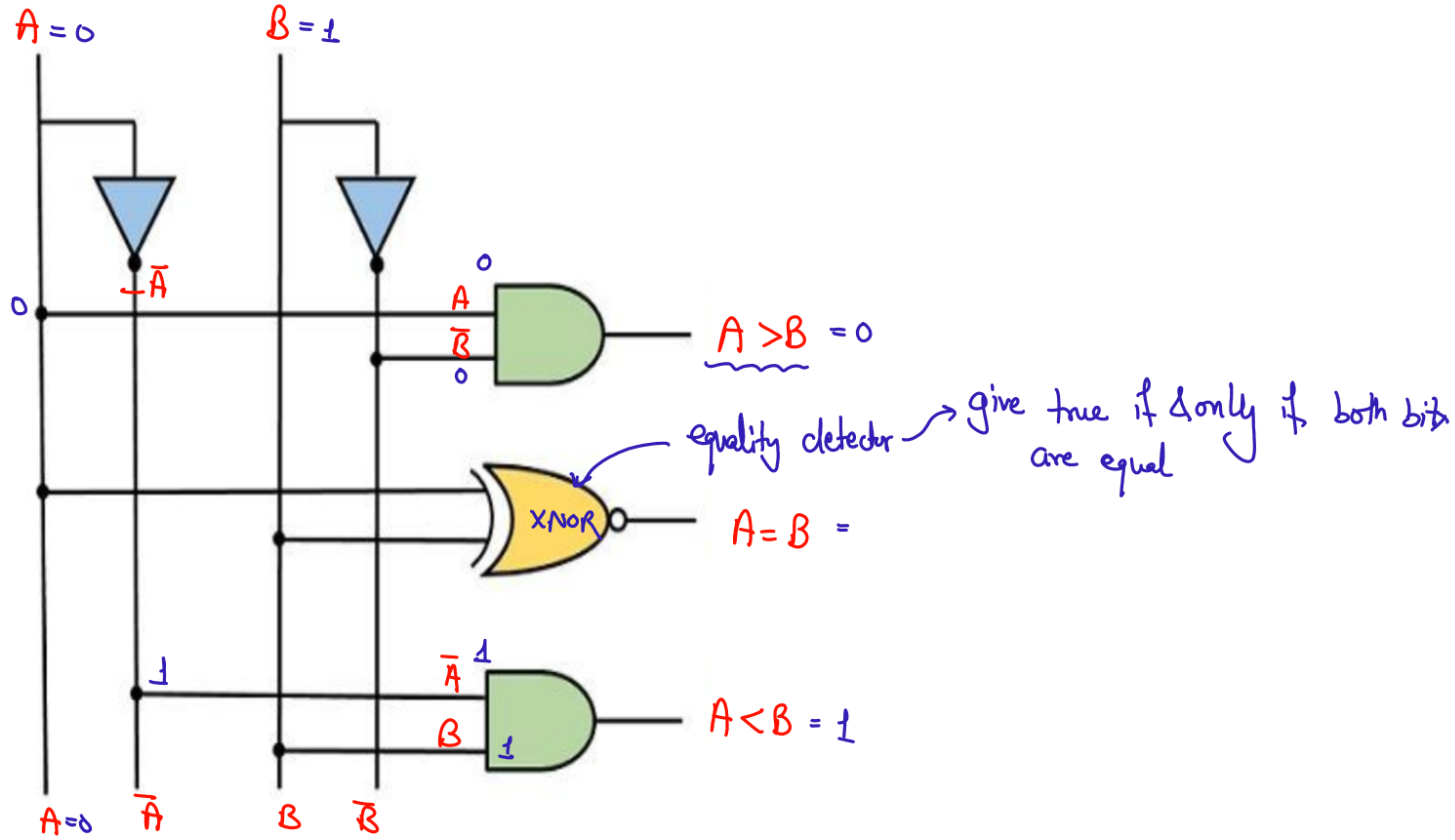
Expression

$A<B : \bar{A}B$

$A=B : AB+\bar{A}\bar{B} \rightarrow A \odot B$

$A>B : A\bar{B}$

1 bit Magnitude Comparator



2 bit magnitude comparator

Compares two bit numbers

$A = A_1A_0$ $B = B_1B_0$

MSB LSB MSB LSB

INPUT				OUTPUT		
A1	A0	B1	B0	A<B	A=B	A>B
0	0	0	0	0	1 m_0	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1 m_5	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1 m_{10}	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1 m_{15}	0

A>B

A1A0 \ B1B0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

$$\boxed{A_1\bar{B}_1 + (A_1 \odot B_1)\cdot A_0\bar{B}_0}$$

A =B

A1A0 \ B1B0	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

$$\boxed{(A_1 \odot B_1)\cdot(A_0 \odot B_0)}$$

A <B

A1A0 \ B1B0	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

$$\boxed{\bar{A}_1 B_1 + (A_1 \odot B_1)\cdot \bar{A}_0 B_0}$$

* Use these Expressions to make logic circuit

$A = B$

$A = \quad A_1 \quad A_0$

$B = \quad B_1 \quad B_0$

$(A_1 \odot B_1) \cdot (A_0 \odot B_0)$ → XNOR

↓

A=B then Y=1

if $A_1 = B_1$ & $A_0 = B_0$ the we can say $A=B$

$A > B$

$A = \quad A_1 \quad A_0$

$B = \quad B_1 \quad B_0$

$A_1 \overline{B_1} + (A_1 \odot B_1) \cdot A_0 \overline{B_0}$

Check for MSB if $A>B$ $O/P = 1$

if A_1 is not greater than B_1 then, Compare of equality

If equal then we Check for $A_0 > B_0$

$A < B$

$A = \quad A_1 \quad A_0$

$B = \quad B_1 \quad B_0$

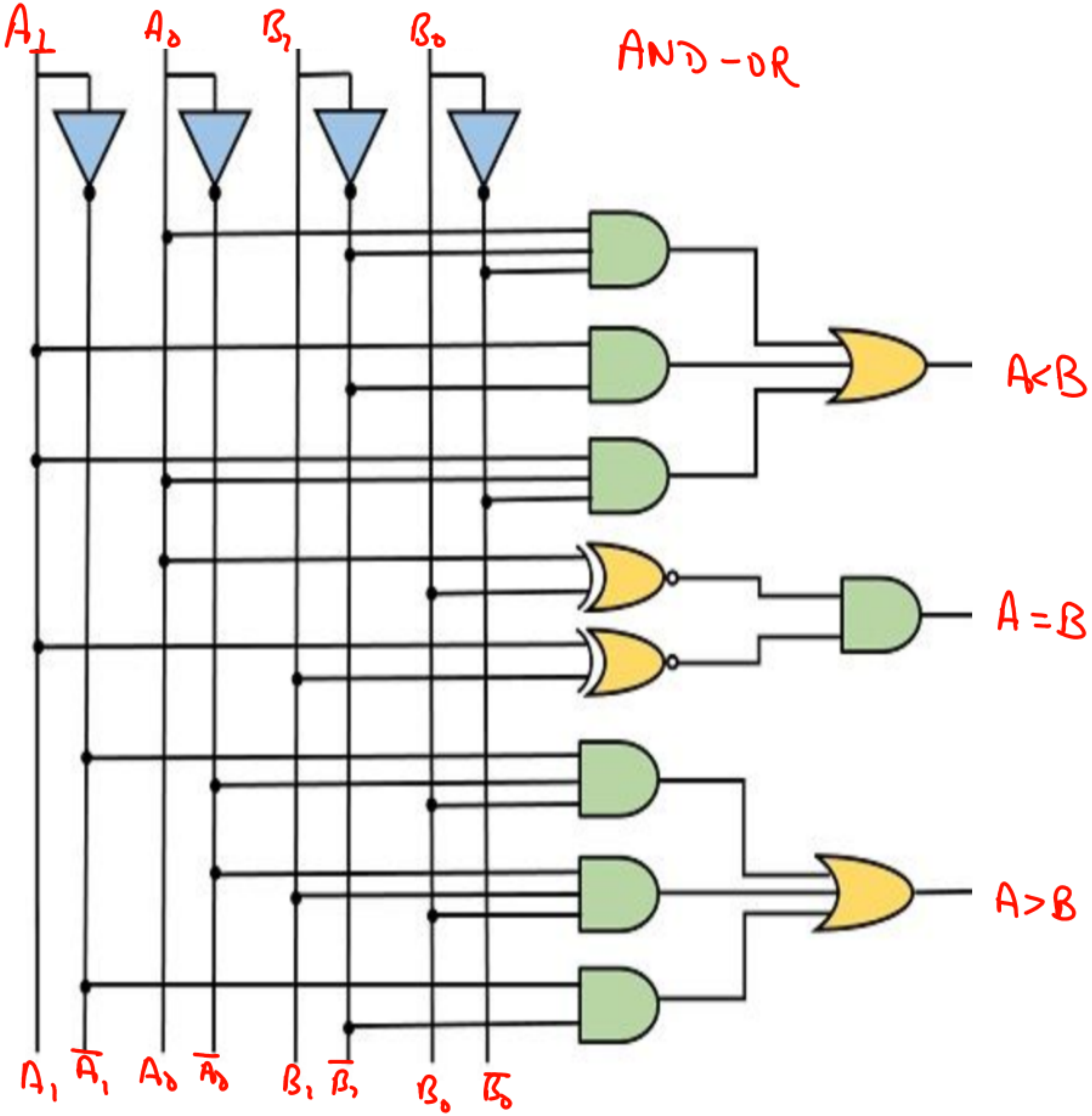
$\overline{A_1} B_1 + (A_1 \odot B_1) \overline{A_0} B_0$

$A_1 < B_1$ the return True (1)

If they are equal then $O/P = 1$

Return True if $A<B$ $O/P = 1$

2 bits magnitude comparator circuit



By Solving K-Maps we will get Basic AND-OR Logic

A=B → Replaced AND-OR Combination with XNOR gates

BCD to XS-3 Converter

BCD to XS-3 Converter (also called BCD to Excess-3 code converter) is a combinational circuit that takes a 4-bit Binary Coded Decimal (BCD) input and produces the corresponding Excess-3 code as output.

Excess-3 code is a non-weighted code used to express decimal numbers. It is derived by adding 3 (i.e., 0011_2) to the BCD number.

	MSB			LSB	MSB			LSB
	BCD Code				Excess-3 Code			
	B_3	B_2	B_1	B_0	X_3	X_2	X_1	X_0
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

$$X_3 = \sum m(5,6,7,8,9)$$

$$X_2 = \sum m(1,2,3,4,9)$$

$$X_1 = \sum m(0,3,4,7,8)$$

$$X_0 = \sum m(0,2,4,6,8)$$

$$+ d(10,11,12,13,14,15)$$

X3

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	(0)	(1)	(3)	(2)
01	(4)	1 (5)	1 (7)	1 (6)
11	X (12)	X (13)	X (15)	X (14)
10	1 (8)	1 (9)	X (11)	X (10)

B_2B_1

B_2B_0

B_3

$$\boxed{X_3 = B_3 + B_2B_1 + B_2B_0}$$

X2

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	(0)	1 (1)	1 (3)	1 (2)
01	1 (4)	(5)	(7)	(6)
11	X (12)	X (13)	X (15)	X (14)
10	(8)	1 (9)	X (11)	X (10)

$\bar{B}_2 B_1$

$B_2\bar{B}_1\bar{B}_0$

$B_0\bar{B}_2$

$$\boxed{X_2 = B_2\bar{B}_1\bar{B}_0 + B_0\bar{B}_2 + B_1\bar{B}_2}$$

X1

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	1 (0)	(1)	1 (3)	(2)
01	1 (4)	(5)	1 (7)	(6)
11	X (12)	X (13)	X (15)	X (14)
10	1 (8)	(9)	X (11)	X (10)

$\bar{B}_1\bar{B}_0$

B_1B_0

$$X_1 = \bar{B}_1\bar{B}_0 + B_1B_0$$
$$= B_1 \odot B_0$$

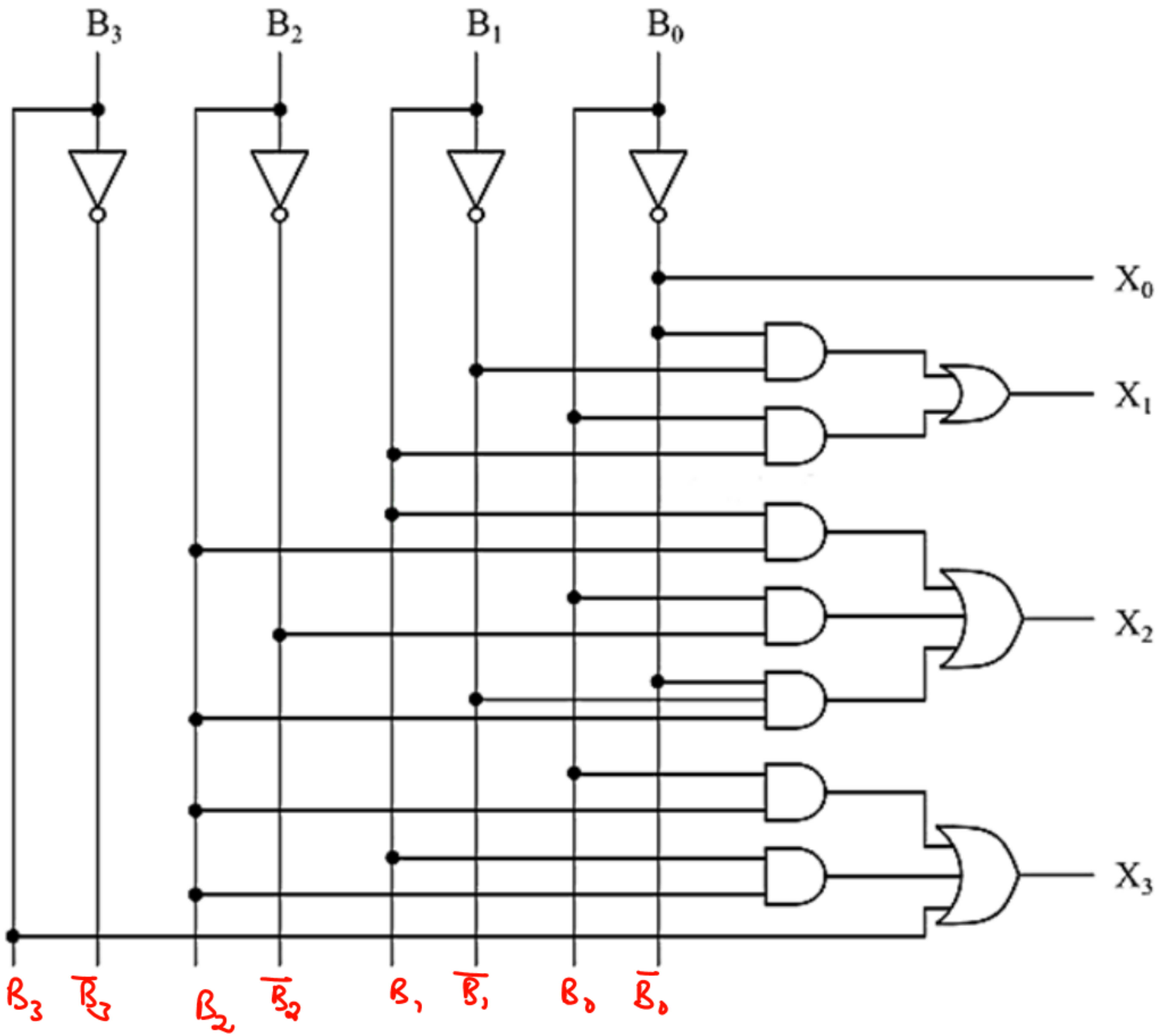
Xo

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	1 (0)	(1)	(3)	1 (2)
01	1 (4)	(5)	(7)	1 (6)
11	X (12)	X (13)	X (15)	X (14)
10	1 (8)	(9)	X (11)	X (10)

B_0

$$\boxed{X_0 = B_0}$$

BCD to XS-3 Code Converter



$X_0 \rightarrow B_0$

X_1: $\bar{B}_1 \bar{B}_0 + B_1 B_0$

X_2: $B_2 \bar{B}_1 \bar{B}_0 + B_0 \bar{B}_2 + B_1 \bar{B}$

X_3: $B_3 + B_2 B_1 + B_2 B_0$

Binary to Gray Code

In Gray Code only one bit changes at a time.
In BCD, only 10 combinations can be made using 4 bits. (0 to 9).
For each Gray code output is D3, D2, D1 and D0.

	Binary Code				Gray Code			
	B3	B2	B1	B0	D3	D2	D1	D0
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

don't care

$$D_3 = \Sigma m(8,9) + d(10,11,12,13,14,15)$$

$$D_2 = \Sigma m(4,5,6,7,8,9) + d(10,11)$$

$$D_1 = \Sigma m(2,3,4,5) + d(10,11,12,13)$$

$$D_0 = \Sigma m(1,2,5,6,9) + d(10,13,14)$$

Binary to Gray Code

D3

B_3B_2 \ B_1B_0	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	X 12	X 13	X 15	X 14
10	1 8	1 9	X 11	X 10

B_3

$\boxed{D_3 = B_3}$

D2

B_3B_2 \ B_1B_0	00	01	11	10
00	0	1	3	2
01	1 4	1 5	1 7	1 6
11	12	13	15	14
10	1 8	1 9	X 11	X 10

$\bar{B}_3B_2$

$B_3\bar{B}_2$

$D_2 = \bar{B}_3B_2 + B_3\bar{B}_2$

$\boxed{D_2 = B_3 \oplus B_2}$

D1

B_3B_2 \ B_1B_0	00	01	11	10
00	0	1	1 3	1 2
01	1 4	1 5	7	6
11	X 12	X 13	15	14
10	8	9	X 11	X 10

$B_2\bar{B}_1$

$B_1\bar{B}_2$

$D_1 = B_2\bar{B}_1 + \bar{B}_2B_1$

$D_1 = B_2 \oplus B_1$

D0

B_3B_2 \ B_1B_0	00	01	11	10
00	0	1 1	3	1 2
01	4	1 5	7	1 6
11	12	X 13	15	X 14
10	8	1 9	11	X 10

$\bar{B}_1B_0$

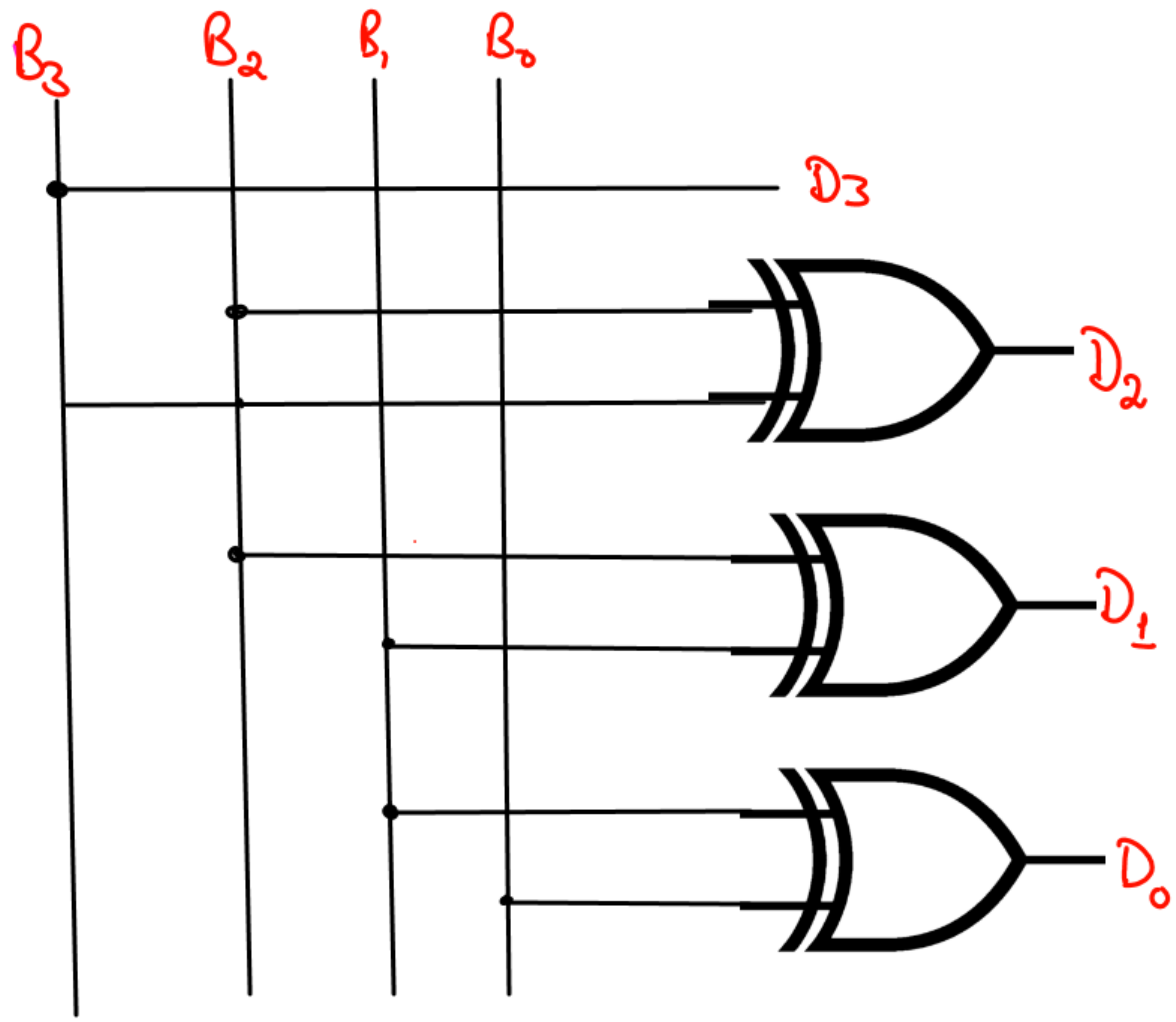
$B_1\bar{B}_0$

$D_0 = \bar{B}_1B_0 + B_1\bar{B}_0$

$\boxed{D_0 = B_1 \oplus B_0}$

* XoR gate is used to Convert the straight Binary code to gray code

Binary to Gray Code → It will also work for 4 Bit Binary to gray Code Converter Not only BCD.



Parity Circuit

A circuit that is used to check that the number of 1's are even or odd.
--> It is used for error detection.
--> It can detect only one bit error.

Parity Generator

↓

It accepts n-1 bits
and add 1 extra bit (at MSB)
Such that total number of
1's in the bit stream are even/odd

Parity Checker

↓

Check that the data
sent by the sender is
Correct or NOT
↓
even parity / odd parity

Even Parity Generator

If number of 1's are even then Y=0 and Number of 1's are odd the Y = 1

3 Bit data

A	B	C	Y	
0	0	0	0	
0	0	1	1	$\rightarrow m_1$
0	1	0	1	$\rightarrow m_2$
0	1	1	0	
1	0	0	1	$\rightarrow m_4$
1	0	1	0	
1	1	0	0	
1	1	1	1	$- m_7$

$$\boxed{f = \sum m(1,2,4,7)}$$

$$= A \oplus B \oplus C$$

$$f = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

$$\bar{B}(\bar{A}C + A\bar{C}) + B(\bar{A}\bar{C} + AC)$$

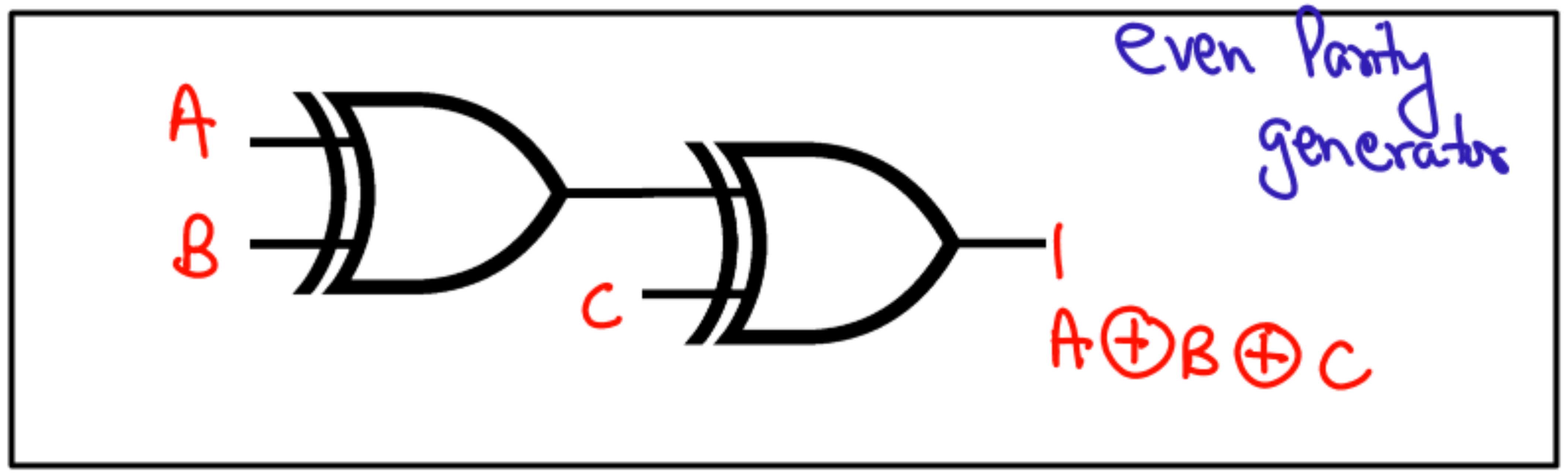
$$\bar{B}(A \oplus C) + B(A \odot C)$$

$$\bar{B}(\underbrace{A \oplus C}_{x}) + B(\underbrace{\overline{A \oplus C}}_{\bar{x}})$$

$$= \bar{B}x + B\bar{x} = B \oplus x$$

$$f = B \oplus A \oplus C$$

$$\boxed{f = A \oplus B \oplus C}$$



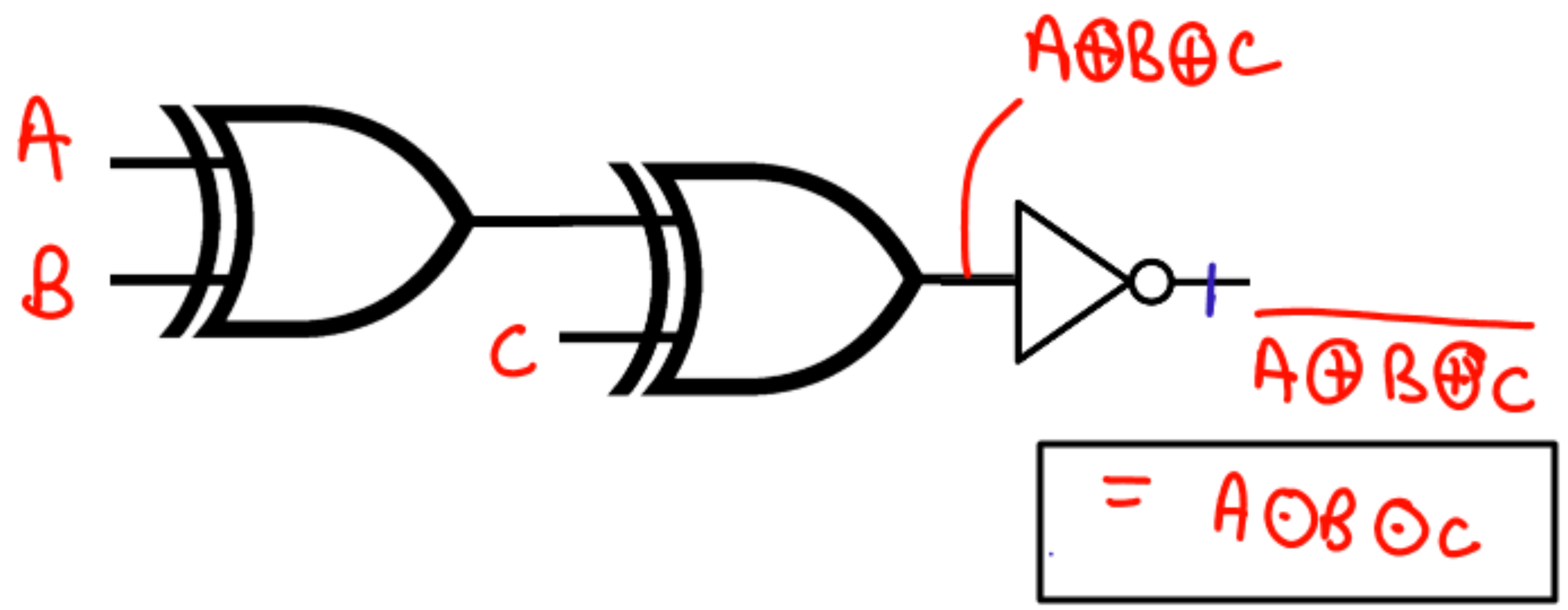
Odd Parity Generator

If number of 1's are odd then Y=0 and Number of 1's are even the Y = 1

$$f = \Sigma m(0,3,5,6)$$

for even Parity, $A \oplus B \oplus C$ then for odd parity the function will be $\overline{A \oplus B \oplus C}$

A	B	C	Y	
0	0	0	1	$\rightarrow m_0$
0	0	1	0	
0	1	0	0	
0	1	1	1	$\rightarrow m_3$
1	0	0	0	
1	0	1	1	$\rightarrow m_5$
1	1	0	1	$\rightarrow m_6$
1	1	1	0	



Parity Check

A	B	C	P_{even}	P_{odd}
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

$$P_{even} = \sum m(0, 3, 5, 6) \rightarrow \text{Kmap}$$

$$P_{odd} = \sum m(1, 2, 4, 7) \rightarrow \text{Kmap}$$

Take 4 Bits for Parity Check
$\downarrow$
4-Variable Kmap