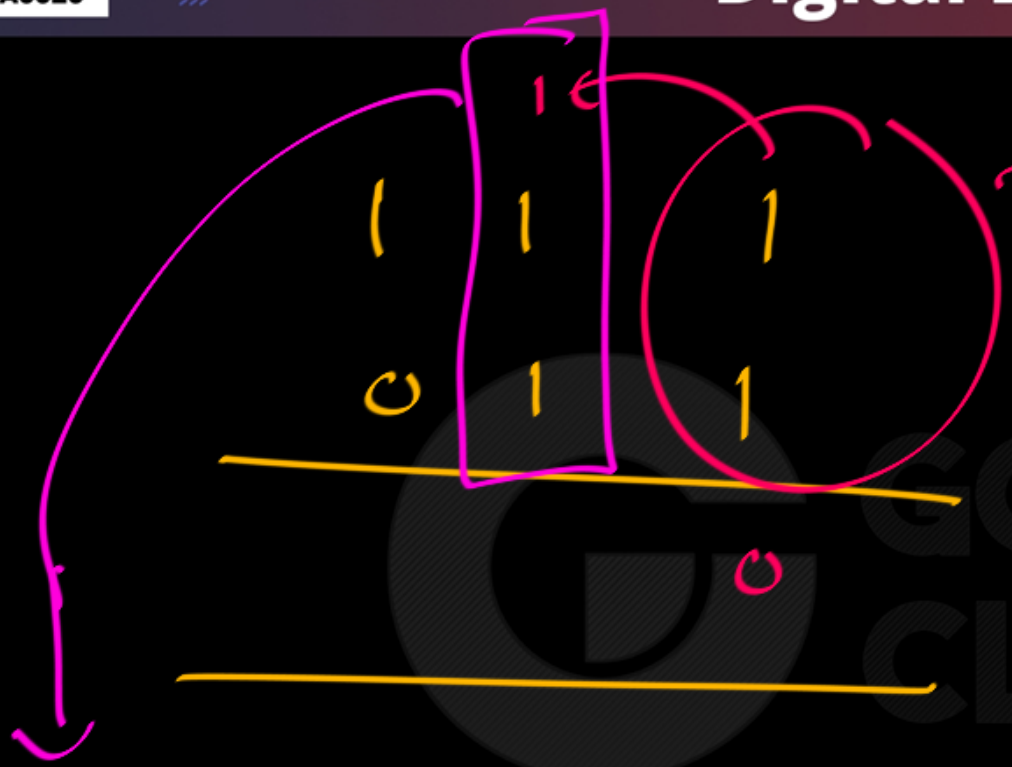
Adder

2. Full Adder

Addition of Multibit Unsigned Numbers

Analogy with addition in base 10

carry	0	1	1	0
		3	8	9
+		1	5	7
		5	4	6



Can use HA

Cannot use HA ; need full Adder.

Ckt for Add of 3 bits.

Analogy with addition in base 10

$$
\begin{array}{cccc}
 & c_3 & c_2 & c_1 & c_0 \\
 & & x_2 & x_1 & x_0 \\
+ & & y_2 & y_1 & y_0 \\
\hline
 & & s_2 & s_1 & s_0
\end{array}
$$

given these 3 inputs

compute these 2 outputs

Analogy with addition in base 10

$$
\begin{array}{cccc}
 & c_3 & c_2 & c_1 \quad c_0 \\
 & & x_2 & x_1 \quad x_0 \\
+ & & y_2 & y_1 \quad y_0 \\
\hline
 & & s_2 & s_1 \quad s_0
\end{array}
$$

Addition of Multi-bit Binary Numbers

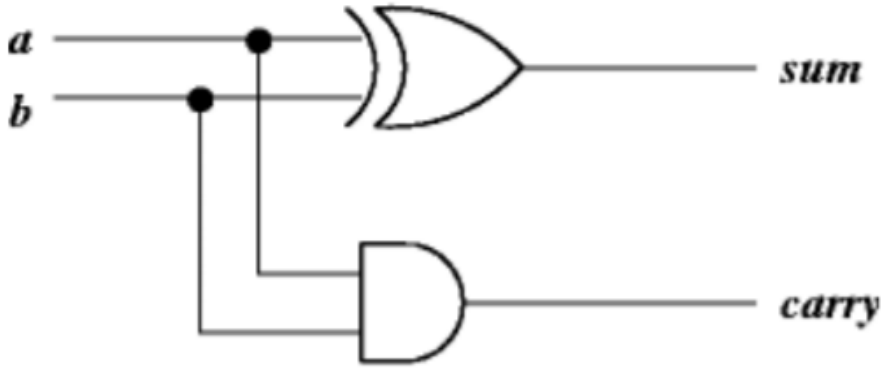
0 0 1 0 1 1 0 ←	Carry
0 1 0 1 0 1 1 ←	Number A
+ 0 0 0 1 0 0 1 ←	Number B
0 1 1 0 1 0 0 ←	Sum S

1 1 1 1 1 1 0 ←	Carry
0 1 1 1 1 1 1 ←	Number A
+ 0 0 0 0 0 0 1 ←	Number B
1 0 0 0 0 0 0 ←	Sum S

- At every bit position (stage), we require to add 3 bits:
  - 1 bit for number A
  - 1 bit for number B
  - 1 carry bit coming from the previous stage

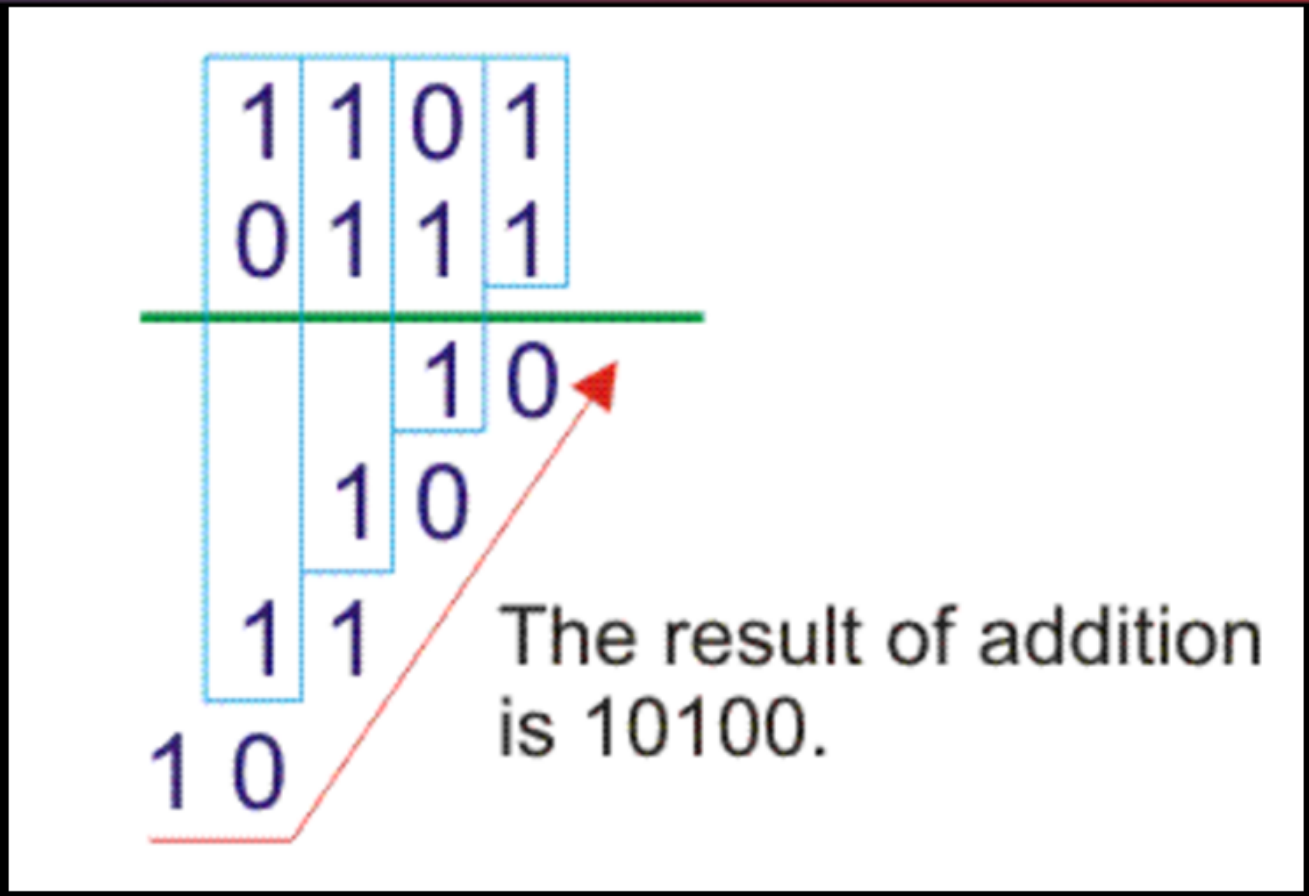
WE NEED A FULL ADDER

Half Adder



$$Sum = \bar{a}b + a\bar{b} = a \oplus b$$
$$Carry = ab$$

- Half adders cannot accept a carry input and hence it is not possible to cascade them to construct an n-bit binary adder.



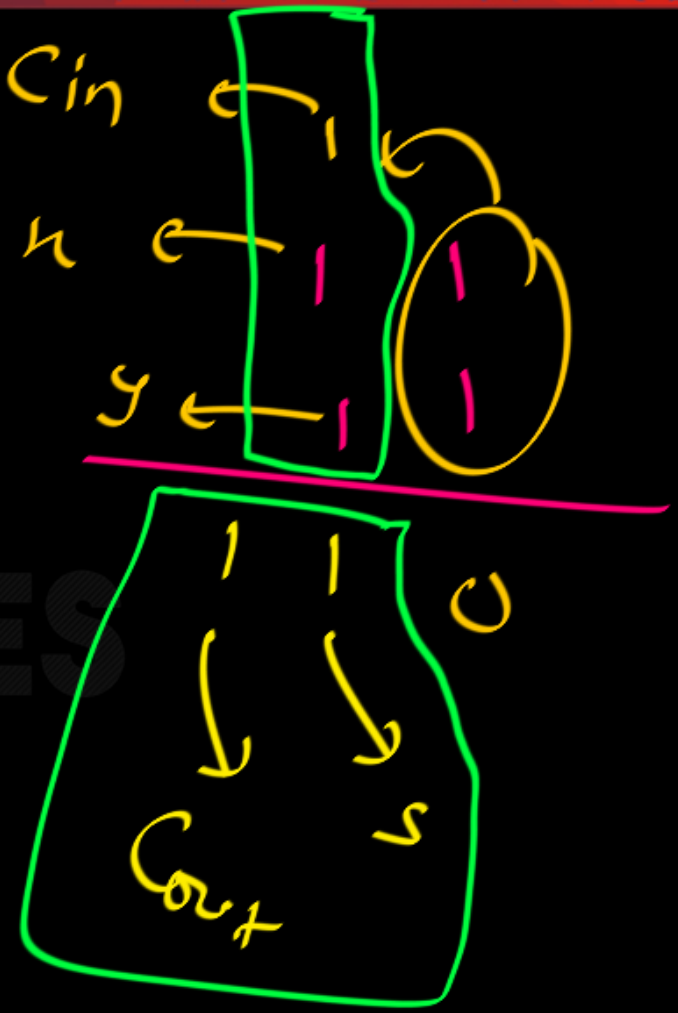
WE NEED A FULL ADDER

2. Full Adder

Full Adder is a combinational circuit that performs the addition of three bits (two significant bits and previous carry).

- It consists of ***three inputs and two outputs***, two inputs are the bits to be added, the third input represents the carry form the previous position.
- The full adder is usually a component in a cascade of adders, which add 8, 16, etc, binary numbers.

↳ carry from previous position



x	y	C_{in}	C_{out}	S
0	0	0	0	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	1	1

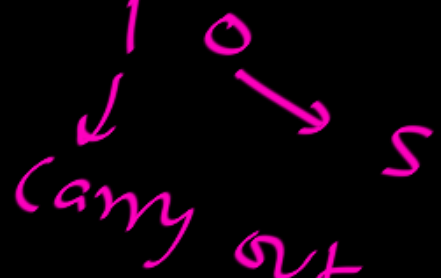
S is 1 when odd number of inputs are 1

C is 1 when at least two inputs are 1.

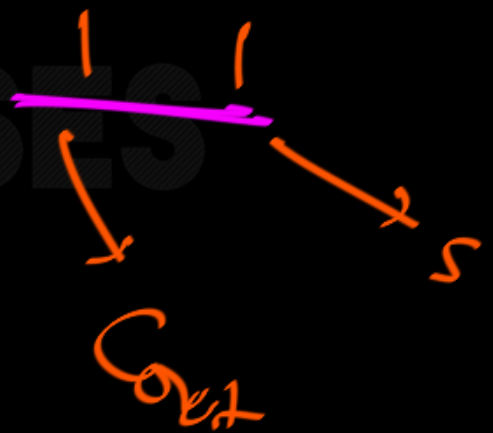
Decimal result

$0 + 1 + 1 = 2$

Binary result

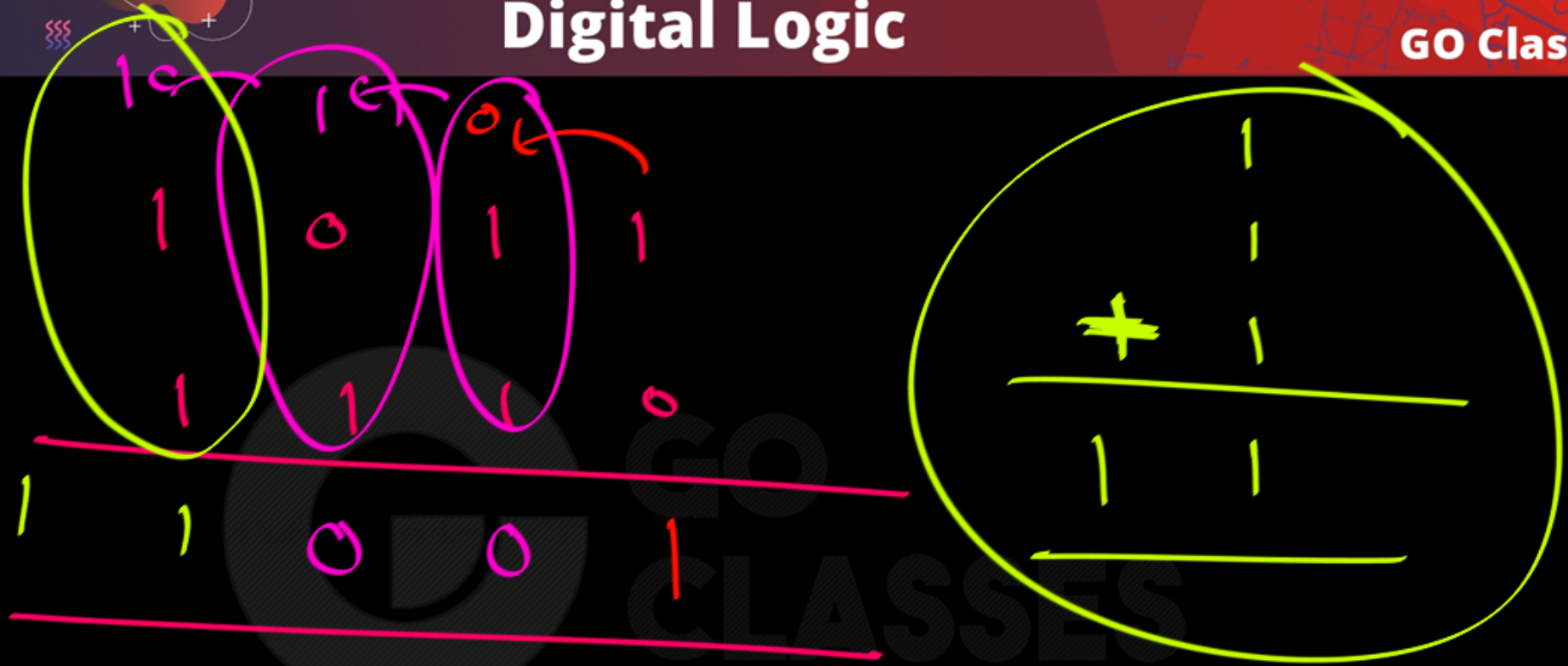
$=$ 

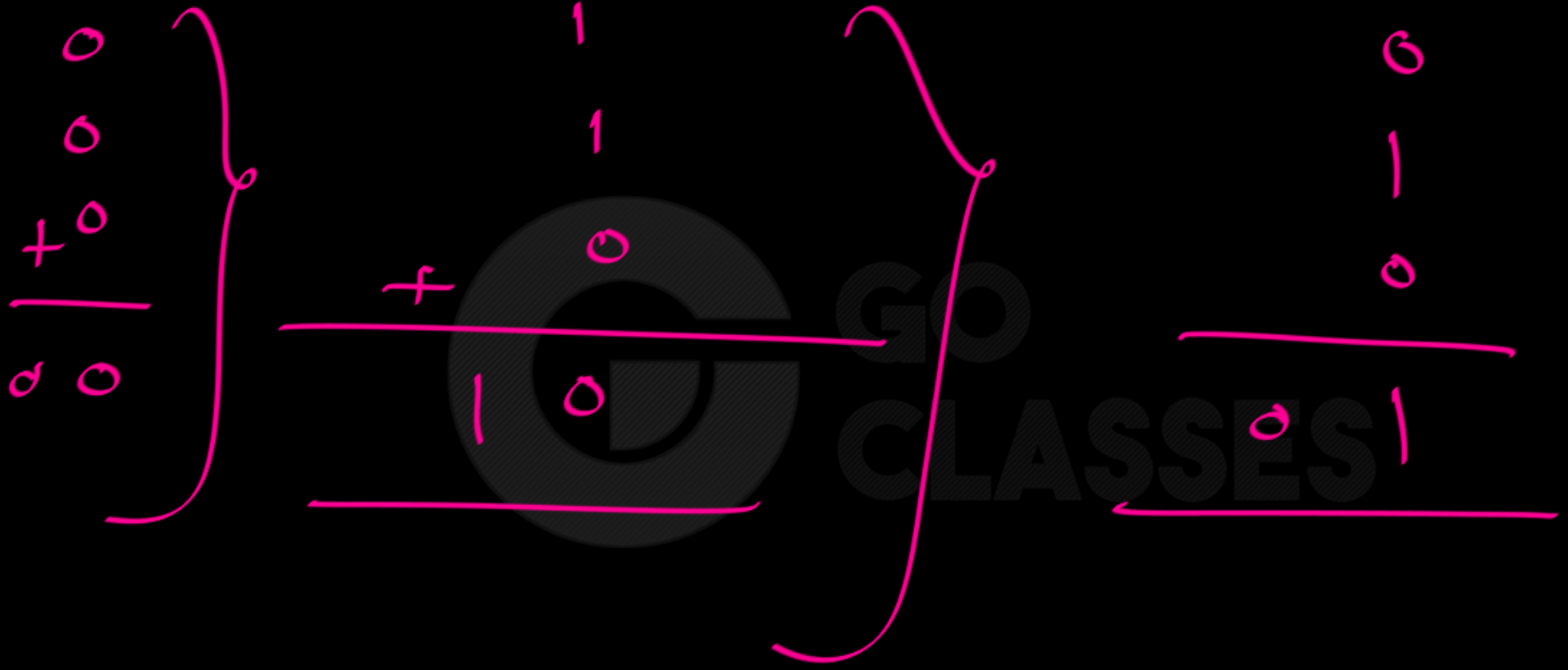
$1 + 1 + 1 = 3$

$=$ 

$1 + 1 + 0 = 2$

$=$ 


1 1 0
1 0 1 1
1 1 0 0
1
1
1 1


0
0
+0
0 0
1
1
+0
1 0
0
1
0
0 1

FULL ADDER :

Addition of n-bit binary numbers requires the use of a full adder, and the process of addition proceeds on a bit-by-bit basis, right to left, beginning with the least significant bit. After the least significant bit, addition at each position adds not only the respective bits of the words, but must also consider a possible carry bit from addition at the previous position.

A full adder is a combinational circuit that forms the arithmetic sum of three bits. It consists of three inputs and two outputs. Two of the input variables, denoted by x and y , represent the two significant bits to be added. The third input, z , represents the carry from the previous lower significant position. Two outputs are necessary because the arithmetic sum of three binary digits ranges in value from 0 to 3, and binary representation of 2 or 3 needs two bits. The two outputs are designated by the symbols S for sum and C for carry.

Inputs			Outputs	
X	Y	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Truth table for the full adder

- The S output is equal to 1 when only one input is equal to 1 or when all three inputs are equal to 1.
- The C_{out} output has a carry 1 if two or three inputs are equal to 1.
- The Karnaugh maps and the simplified expression are shown in the following figures:

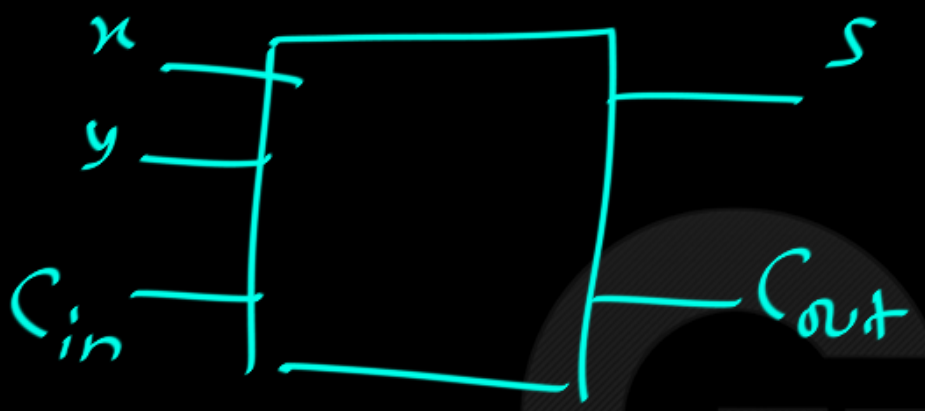
Note: EXOR is odd-Inputs detecting function. EXOR is 1 iff odd number of i/ps are 1.

X \ YC_{in}	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$S = \overline{X}\,\overline{Y}C_{in} + \overline{X}Y\overline{C_{in}} + X\overline{Y}\,\overline{C_{in}} + XYC_{in}$$

X \ YC_{in}	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$C_{out} = XY + XC_{in} + YC_{in}$$



$$S = x \oplus y \oplus C_{in}$$

$$C_{out} = xy + xC_{in} + yC_{in}$$

C_{out} is 1 when at least two inputs are 1.

$$S = \bar{x}\,\bar{y}\,C_{in} + x\,\bar{y}\,\bar{C_{in}} + \bar{x}\,y\,\bar{C_{in}} + xy\,C_{in}$$

S is 1 when odd no. of inputs are 1.

$$S = \bar{x}\,\bar{y}\,C_{in} + \bar{x}\,y\,\bar{C_{in}} + x\,\bar{y}\,\bar{C_{in}} + xy\,C_{in}$$

In FA:

for S, we have two equations:

$$S = x \oplus y \oplus C_{in}$$

$$S = \bar{x}\,\bar{y}\,C_{in} + \bar{x}\,y\,\bar{C_{in}} + x\,\bar{y}\,\bar{C_{in}} + xyC_{in}$$

for carry (C_{out}) we have reversal equations:

$$C = xy + xC_{in} + yC_{in}$$

$$C = xy + C_{in}(x+y)$$

$$C = xy + C_{in}(x \oplus y)$$

$$C = xy + C_{in}(x + y)$$

$$C = xy + C_{in}x + C_{in}y$$

$$C = xy \oplus C_{in}(x \oplus y)$$

$C = xy + C_{in}(x \oplus y)$

$C = xy \oplus C_{in}(x \oplus y)$

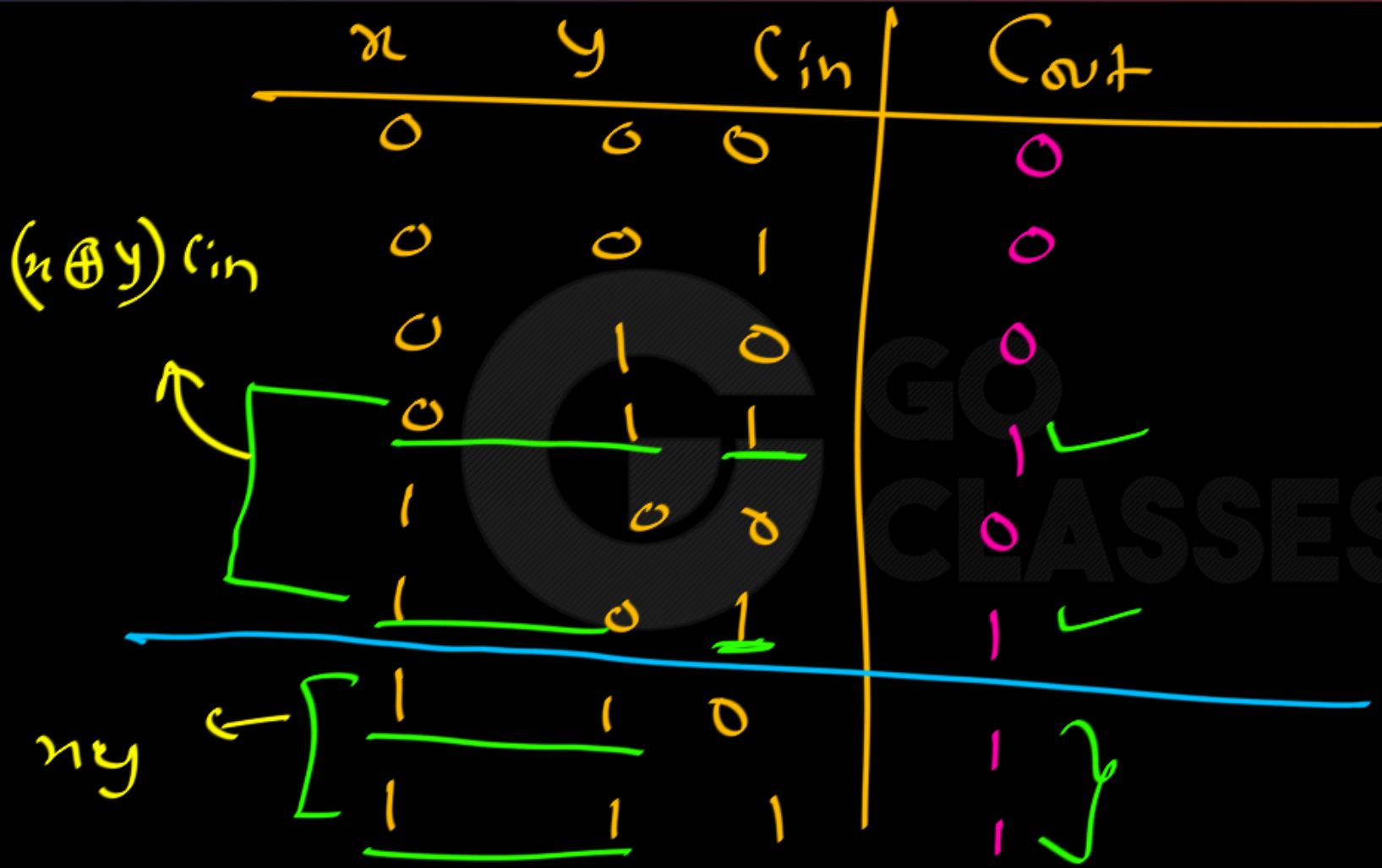
Same

$C = xy + C_{in}(x + y)$

$C = xy \oplus C_{in}(x + y)$

different

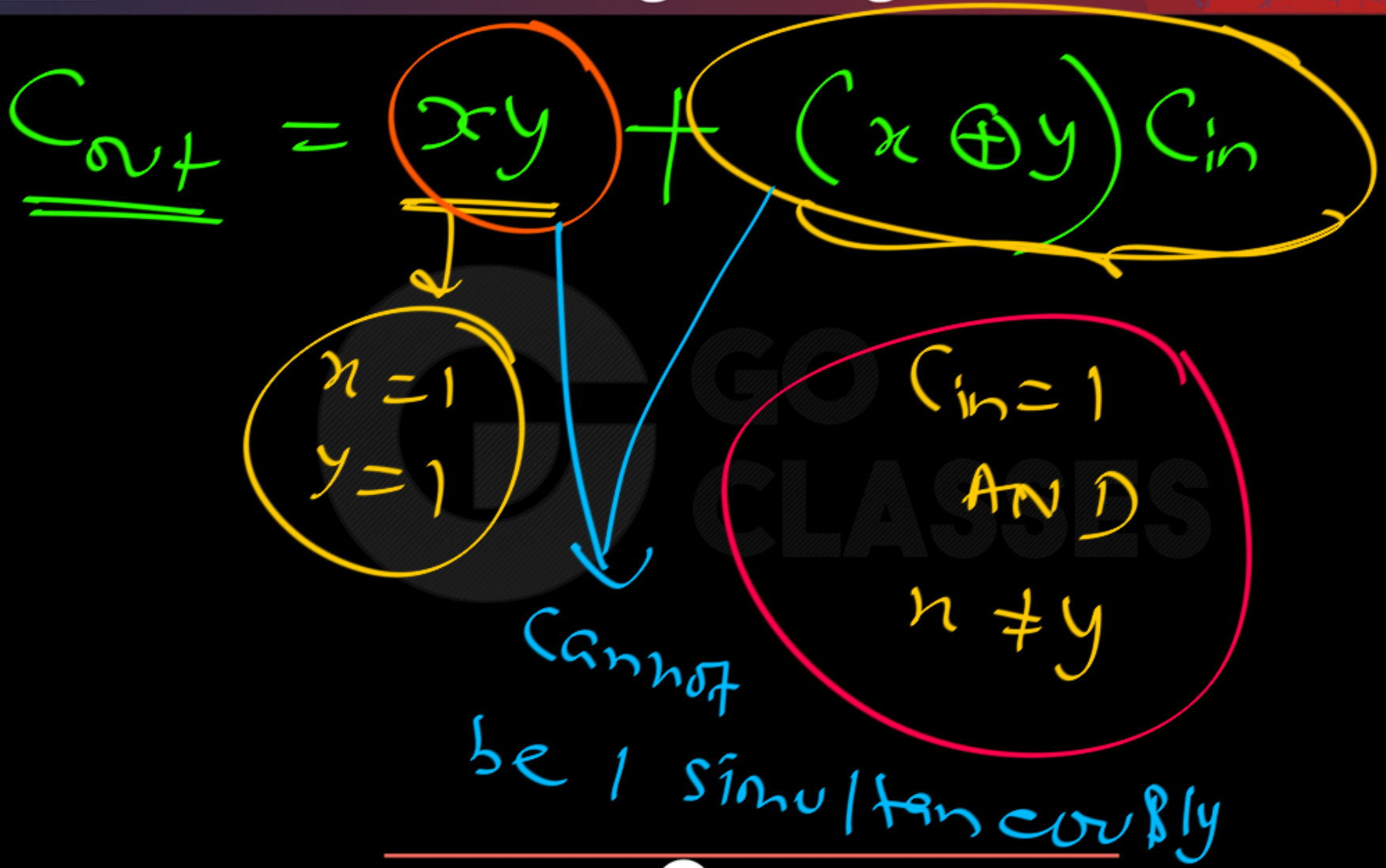
↳ wrong exp for Cout'


x
y
Cin
Cout
(x⊕y) Cin
xy
0 0 0 0
0 0 1 0
0 1 0 0
0 1 1 1
1 0 0 0
1 0 1 1
1 1 0 1
1 1 1 1

$$C_{out} = xy + (x \oplus y) C_{in}$$

$x = 1$
$y = 1$

$C_{in} = 1$
AND
$x \neq y$


$C_{out} = xy + (x \oplus y) C_{in}$
$x = 1$
$y = 1$
$C_{in} = 1$
AND
$x \neq y$
Cannot
be 1 simultaneously

Difference b/w + , ⊕ :

$a + b$; $a \oplus b$

the only difference is when $a = b = 1$

If it is NOT possible that a, b are simultaneously 1 then $a + b = a \oplus b$

Eg: $f = \underbrace{ab}_{\alpha} + \underbrace{a\bar{b}}_{\beta}$

Is it possible that α, β are 1 simultaneously?

No.

So $f = ab + a\bar{b} = ab \oplus a\bar{b}$

$ab + a\bar{b} = (ab) \oplus (a\bar{b})$

$\Downarrow$

$a(b + \bar{b})$

$a(b \oplus \bar{b})$

a

$a(1)$

$a = a(1)$

$a = a$

Distributive Prop:

$$\underbrace{a \cdot (b \oplus c)}_{LHS} = \underbrace{ab \oplus ac}_{RHS}$$

(AND is Dist. over Exor)

Prove:

$a = 0$

LHS: 0 = RHS: 0

$a = 1$

LHS: $b \oplus c$ = RHS: $b \oplus c$

a	b	f
0	0	1
0	1	1
1	0	0
1	1	1

CSOP:

$$f = \bar{a}\bar{b} + \bar{a}b + ab$$

$$g = (\bar{a}\bar{b}) \oplus (\bar{a}b) \oplus ab$$

same

$$f = \bar{a}\bar{b} + \bar{a}b + ab = \bar{a}\bar{b} + b$$

$$= \bar{a} + b$$

$$g = \bar{a}\bar{b} \oplus \left[\bar{a}b \oplus ab\right] = \bar{a}\bar{b} \oplus \left(b\right)$$

$$= \overline{\bar{a}\bar{b}}\, b + a\bar{b}\,\bar{b}$$

$$= (a+b)\,b + \bar{a}\bar{b} = b + \bar{a}$$

$$\left\{\begin{array}{l} S = \overline{X}\,\overline{Y}C_{in} + \overline{X}Y\overline{C_{in}} + X\overline{Y}\,\overline{C_{in}} + XYC_{in} \\ \\ C_{out=}\, XY + XC_{in} + YC_{in} \end{array}\right\} 1 \quad \textit{(Sum of products)}$$

$$\left\{\begin{array}{l} S = C_{in} \oplus (X \oplus Y) \\ C_{out=}\, C_{in} \cdot (X \oplus Y) + XY \end{array}\right\}$$

$$Sum = \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} + abc = a \oplus b \oplus c$$
$$C_{out} = ab + ac + bc = ab + c(a+b)$$

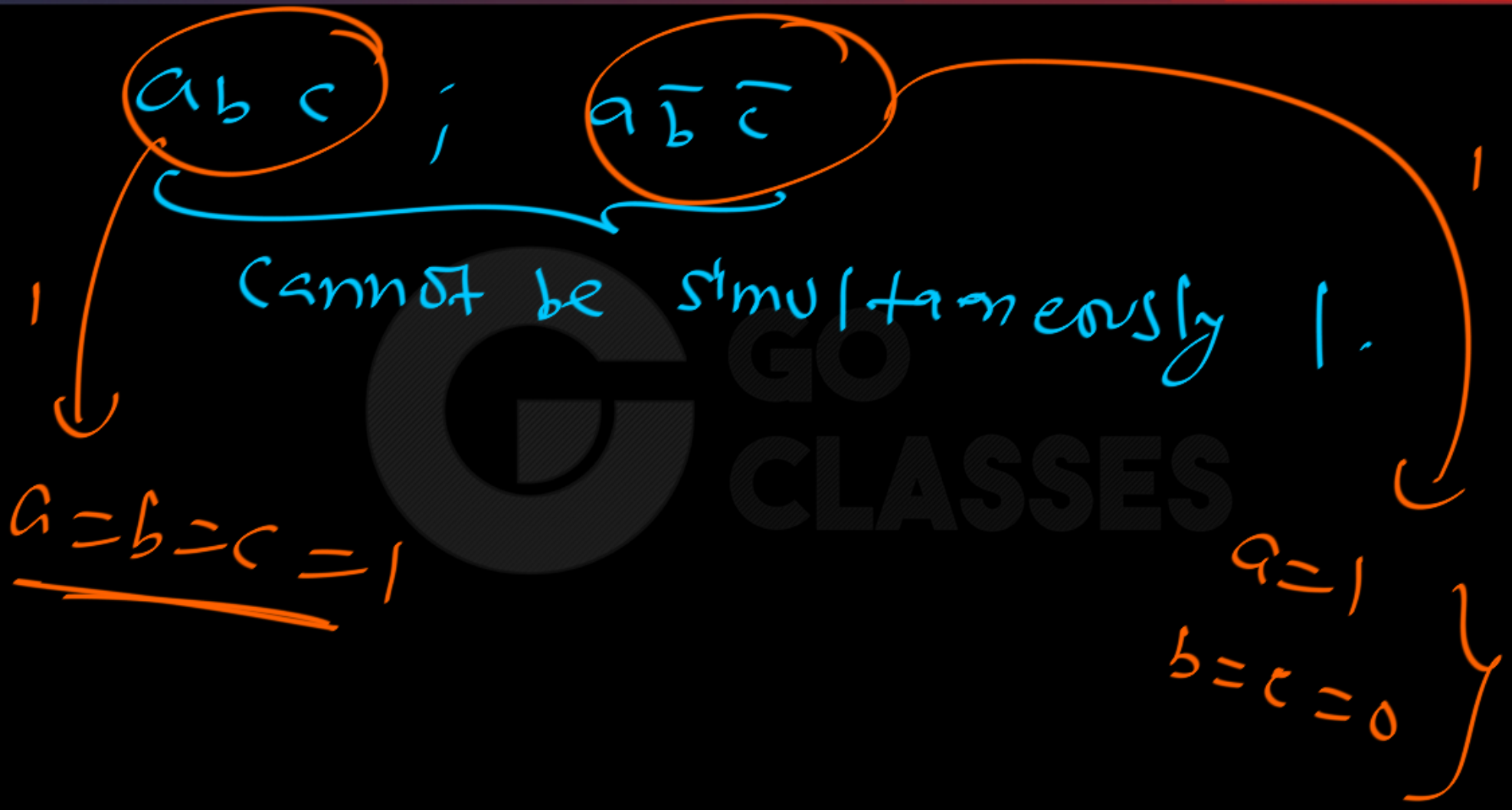
Note: If we know that it is Not possible that any two of α, β, γ are simultaneously 1 then

$$\alpha + \beta + \gamma = \alpha \oplus \beta \oplus \gamma$$

So

Note: Any function can be expressed as Exor of (true) minterms.

Why? because two minterms can NEVER be simultaneously 1.


a b c
;
a b̄ c̄
cannot be simultaneously 1.
1
1
a=b=c=1
a=1
b=c=0

The sum:

$$S = \bar{X}\,\bar{Y}C_{in} + \bar{X}Y\overline{C_{in}} + X\bar{Y}\,\overline{C_{in}} + XYC_{in}$$

$$= \overline{C_{in}}(\bar{X}Y + X\bar{Y}) + C_{in}(\bar{X}\,\bar{Y} + XY)$$

$$= \overline{C_{in}}(\bar{X}Y + X\bar{Y}) + C_{in}\overline{(\bar{X}Y + X\bar{Y})}$$

$$S = C_{in} \oplus (X \oplus Y)$$

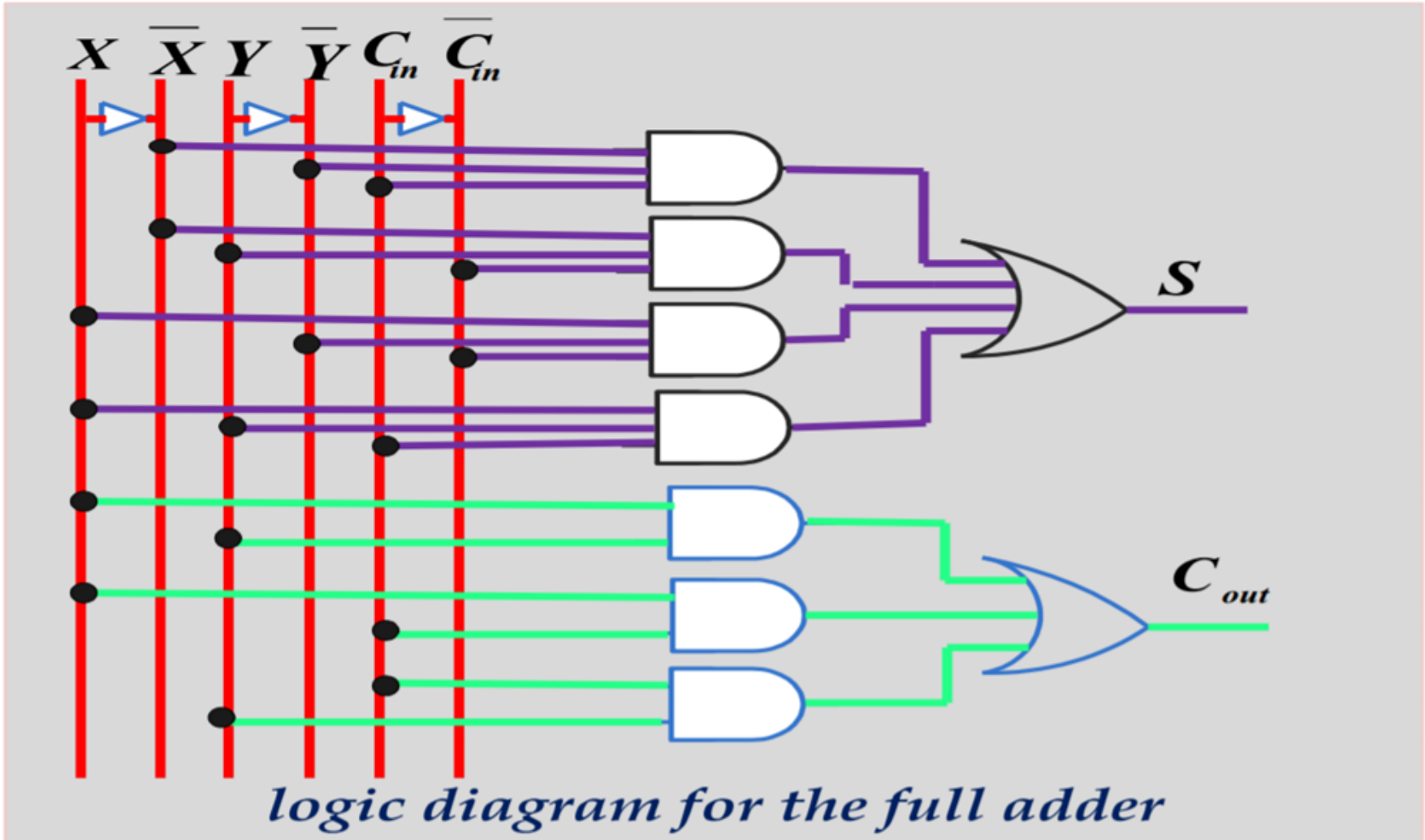
The carry output:

$$C_{out} = \bar{X}YC_{in} + X\bar{Y}C_{in} + XYC_{in} + XY\,\overline{C_{in}}$$

$$= C_{in}(\bar{X}Y + X\bar{Y}) + XY(C_{in} + \overline{C_{in}})$$

$$C_{out} = C_{in} \cdot (X \oplus Y) + XY$$

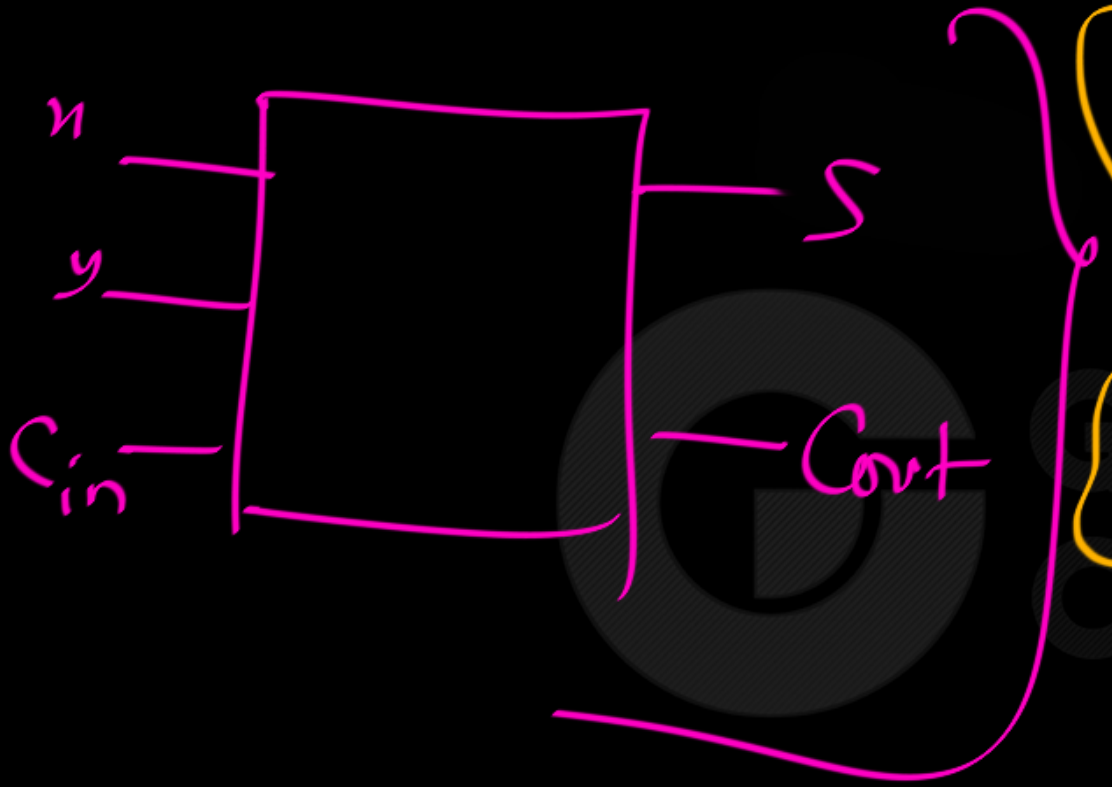
The ***logic diagrams*** for the full adder implemented in ***sum-of-products*** form are the following:



logic diagram for the full adder

- It can also be implemented using ***two half adders*** and ***one OR gate*** (using ***XOR*** gates).

$$\begin{Bmatrix} S = C_{in} \oplus (X \oplus Y) \\ C_{out} = C_{in} \cdot (X \oplus Y) + XY \end{Bmatrix}$$



$$S = \bar{x}\bar{y}C_{in} + \bar{x}y\bar{C_{in}} + x\bar{y}\bar{C_{in}} + xyC_{in}$$

$$C = xy + xC_{in} + yC_{in}$$

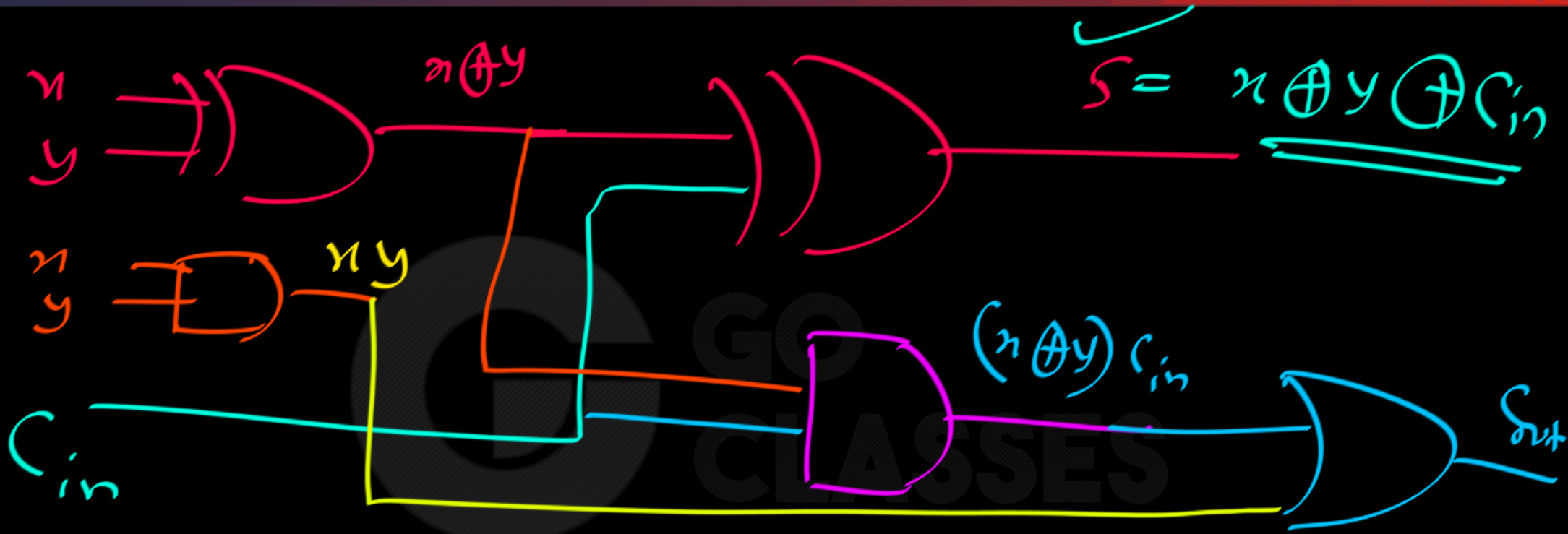
AND-OR implementation.

$S = x \oplus y \oplus C_{in}$

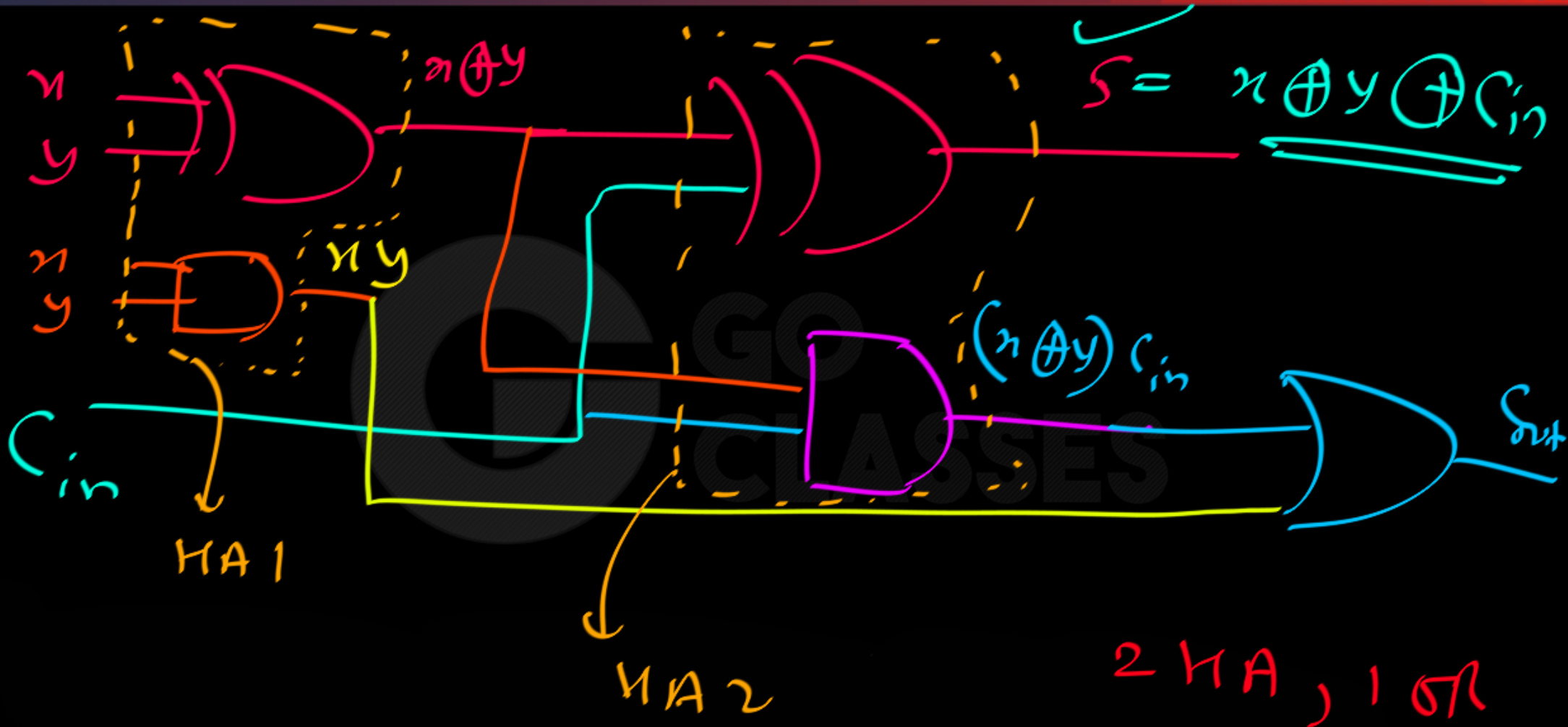
$C = xy + C_{in}(x \oplus y)$

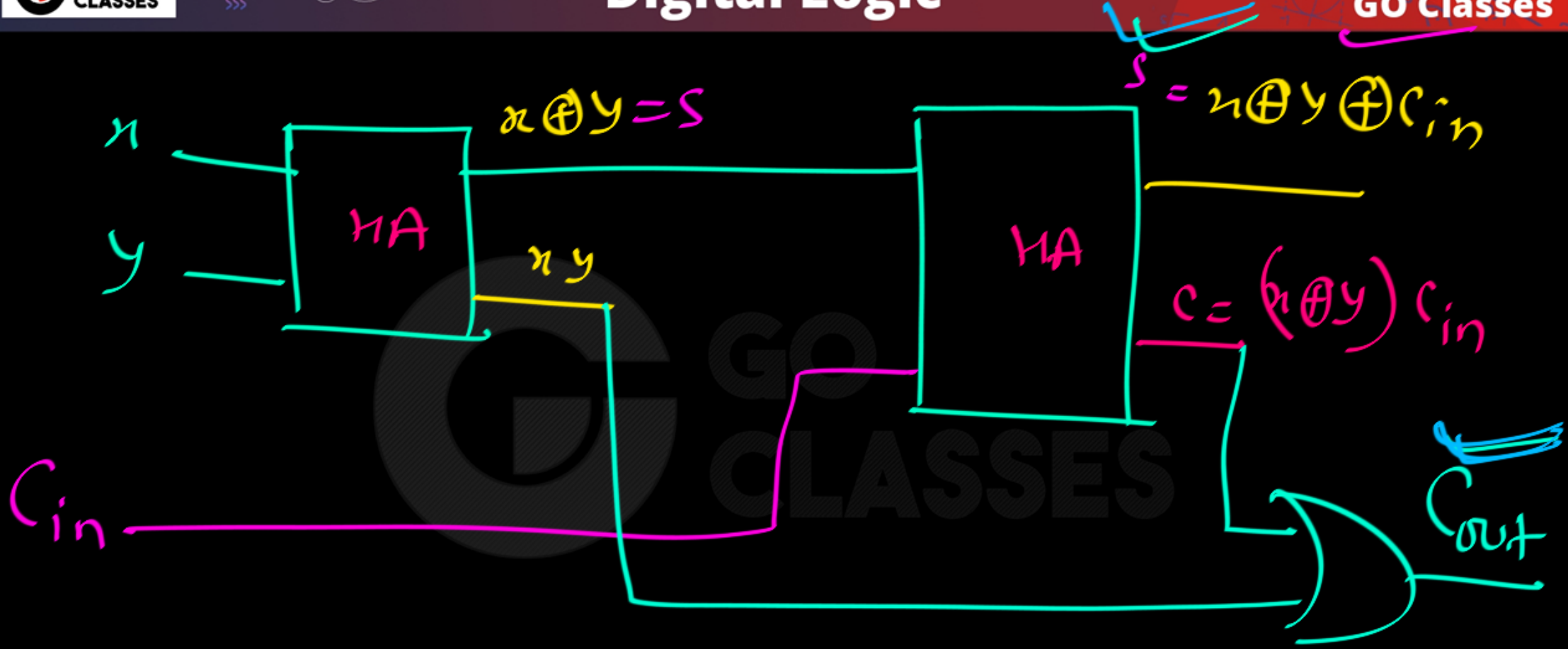
AND, EX-OR, OR implementation

Using HA implementation

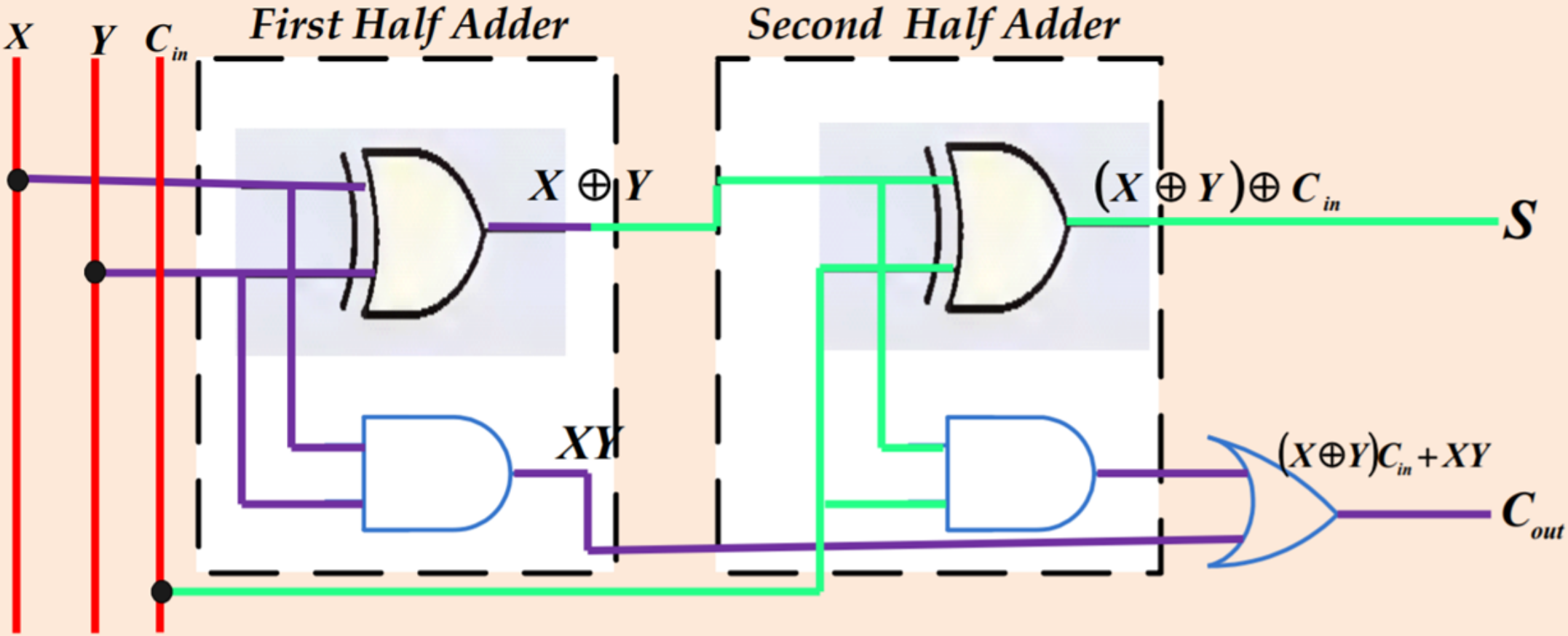


AND, OR, EXOR implementation


x
y
x ⊕ y
S = x ⊕ y ⊕ Cin
x
y
xy
(x ⊕ y) Cin
Cin
Cout
HA 1
HA 2
2 HA, 1 OR gate



FA using 2 HA, 1 OR gate



Implementation of Full Adder with two Half Adders and an OR gate

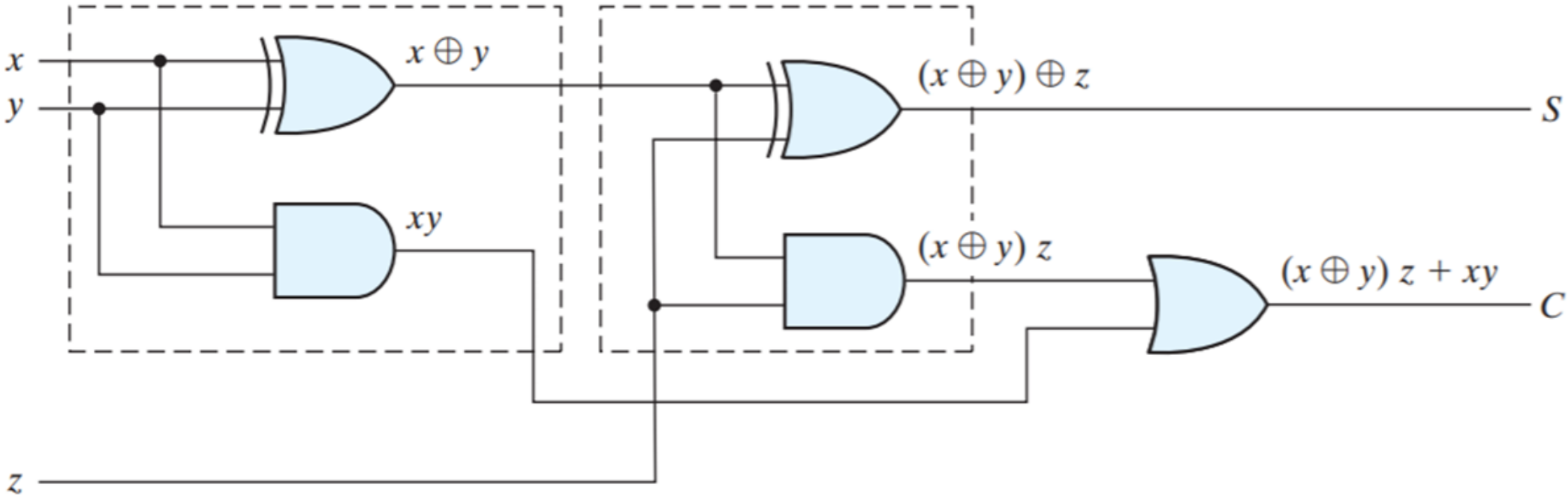
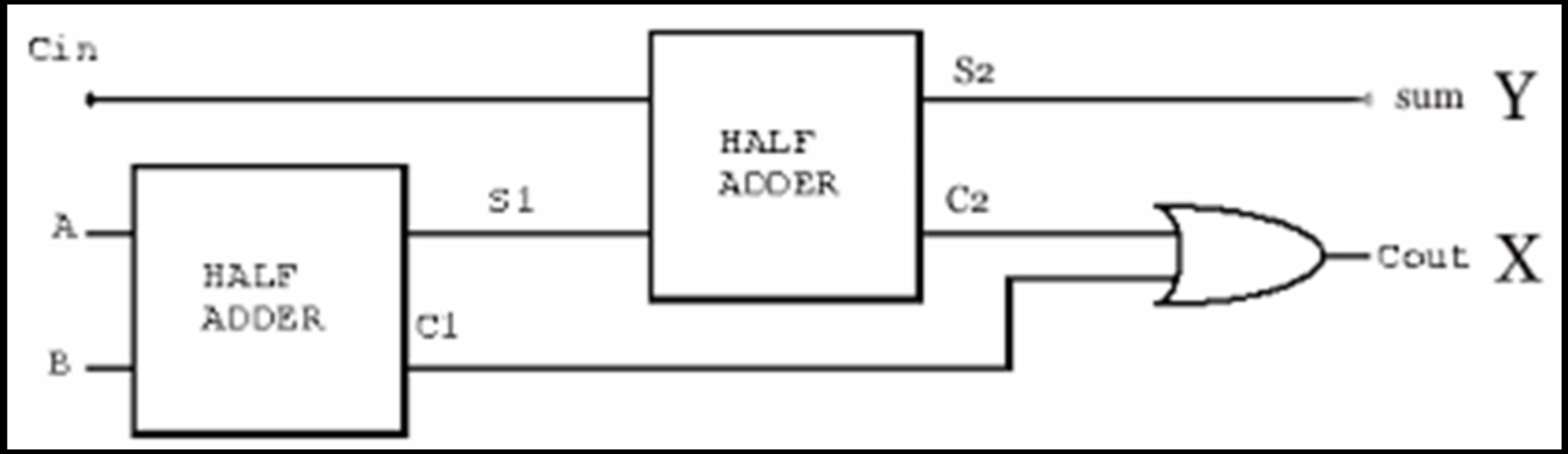


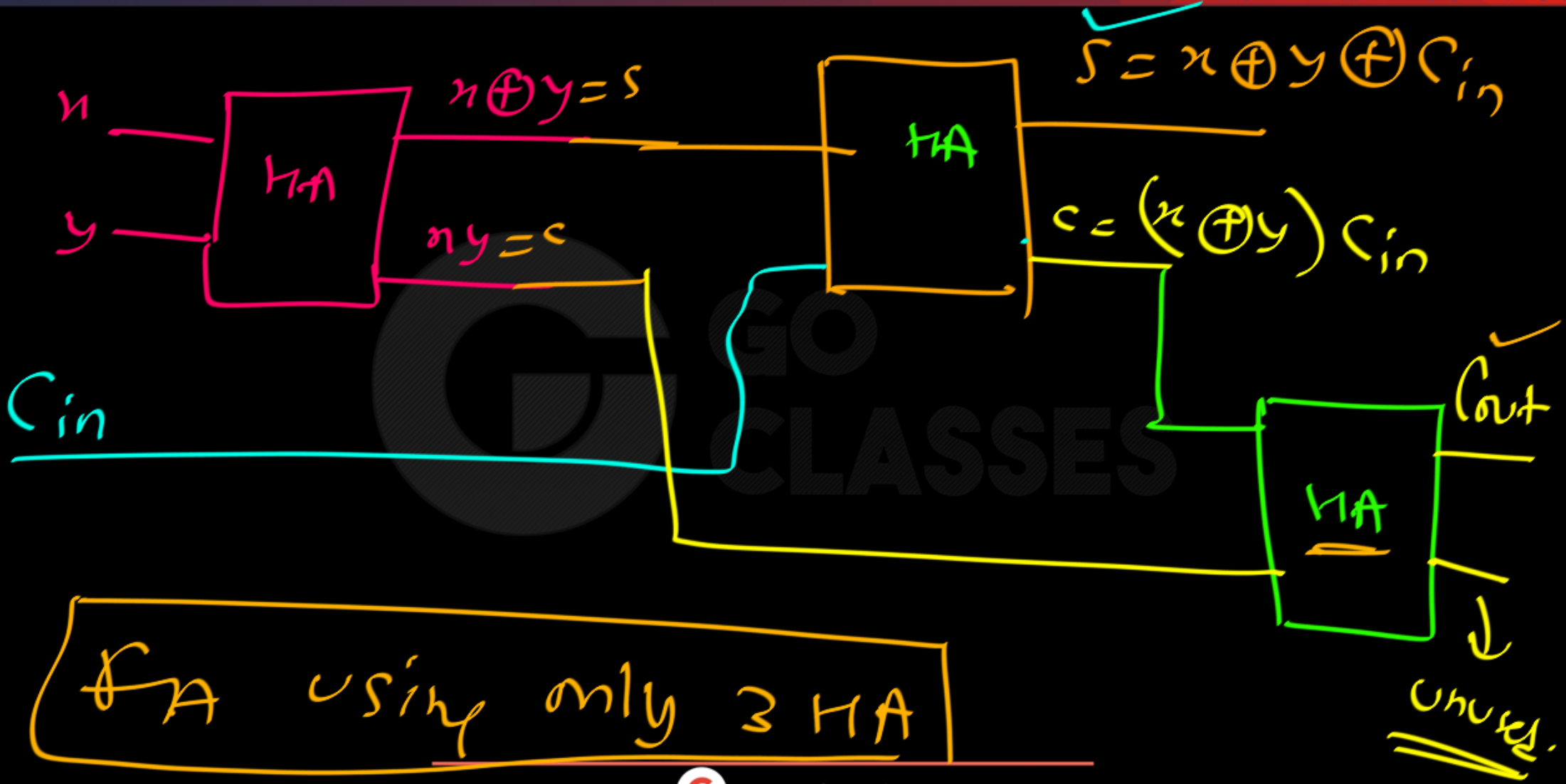
FIGURE 4.8
Implementation of full adder with two half adders and an OR gate


Cin
A
B
HALF ADDER
S1
C1
HALF ADDER
S2
C2
sum
Y
Cout
X

Q: FA using only Half Adders ;
How many HA we will need ?

FA = 2 HA, 1 OR gate

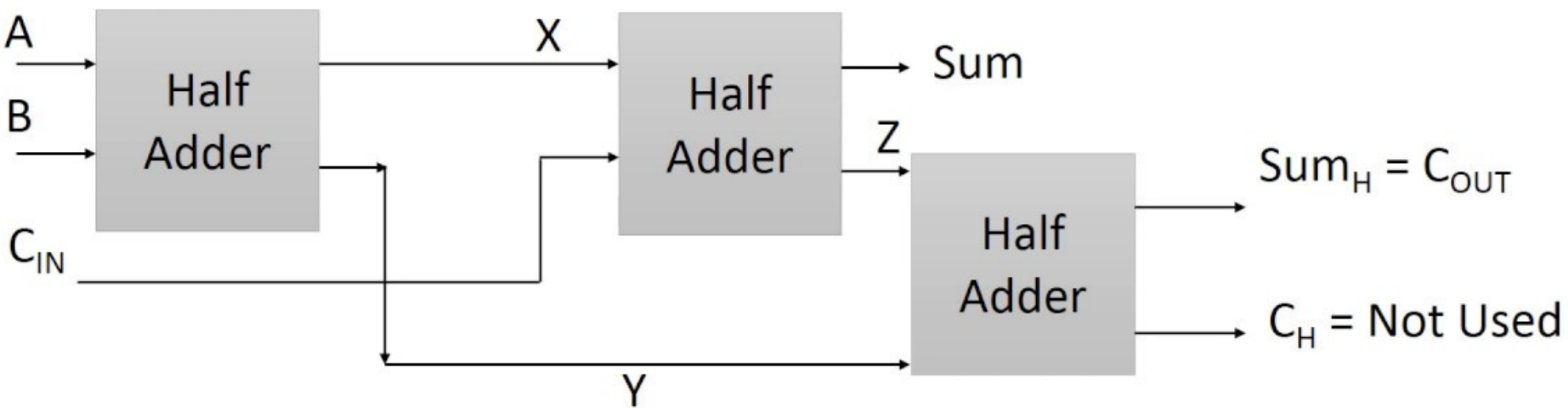
FA = 3 HA



$$C_{out} = \overline{xy} + C_{in}\overline{(x \oplus y)}$$

$$C_{out} = (xy) \oplus C_{in}(x \oplus y)$$

Using 1 HA $\Rightarrow$ we can Do "AND" "EXOR".


A
B
C_{IN}
Half Adder
X
Half Adder
Sum
Z
Half Adder
Y
$Sum_H = C_{OUT}$
C_H = Not Used

Note: In FA:

Equation for S:

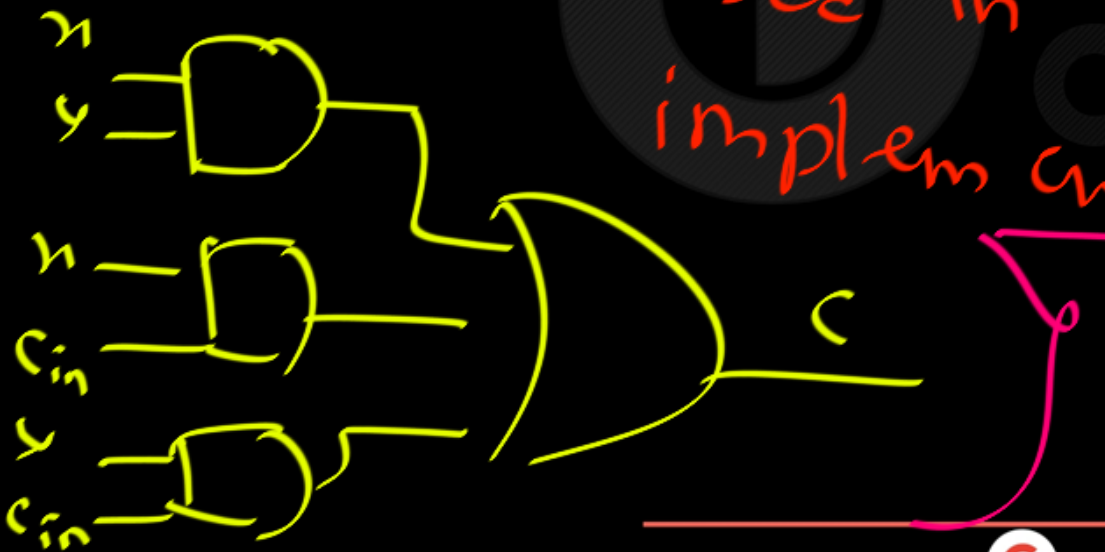
$$S = x \oplus y \oplus C_{in} = x \odot y \odot C_{in}$$

$$S = \bar{x}\bar{y}C_{in} + \bar{x}y\bar{C}_{in} + x\bar{y}\bar{C}_{in} + xyC_{in}$$

Equation for C_{out} :

$$C = xy + xC_{in} + yC_{in}$$

used in 2 level AND-OR implementation.



Equation for Cout:

$C = xy + (x \oplus y) C_{in}$

Very useful; used in the FA implementation

Using 2 HA, 1 OR gate.

Popular implementation

Equation for Cout:

$$C = \underbrace{(x\,y) \oplus [(x \oplus y)\, c_{in}]}$$

uses in FA implementation

using 3 HA. (using only HA)

$$C = xy + C_{in}(x+y)$$

Not much useful