Adder

3. Ripple Carry Adder

- Delay of a full adder (If implemented using Basic gates) :

– Assume that the delay of all basic gates (AND, OR, NAND, NOR, NOT) is δ.

– Delay for Carry = 2δ

– Delay for Sum = 3δ

(AND-OR delay plus one inverter delay)

- Delay of a full adder (If implemented using Basic gates) :
  - Assume that the delay of all basic gates (AND, OR, NAND, NOR, NOT) is δ.
  - Delay for Carry = 2δ
  - Delay for Sum = 3δ

(AND-OR delay plus one inverter delay)

Assume AND, OR, NOT are Basic gates.

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

2δ

for sum

3δ

In FA (If implemented using only Basic gates)

Delay for Sum : 3δ (NOT gate → AND gate → OR gate)

" " Carry : 2δ (AND → OR)

Delay of FA : max (3δ, 2δ) = 3δ

HA : Can add two bits.

FA : " " 3 bits.

We want binary number addition.

0 1 1

+ 1 0 1

Cannot be done using one FA.

Parallel Binary Adder Design :

A binary adder is a digital circuit that produces the arithmetic sum of two binary numbers. It can be constructed with full adders connected in cascade, with the output carry from each full adder connected to the input carry of the next full adder in the chain.

- We shall look at the various designs of n-bit parallel adder.

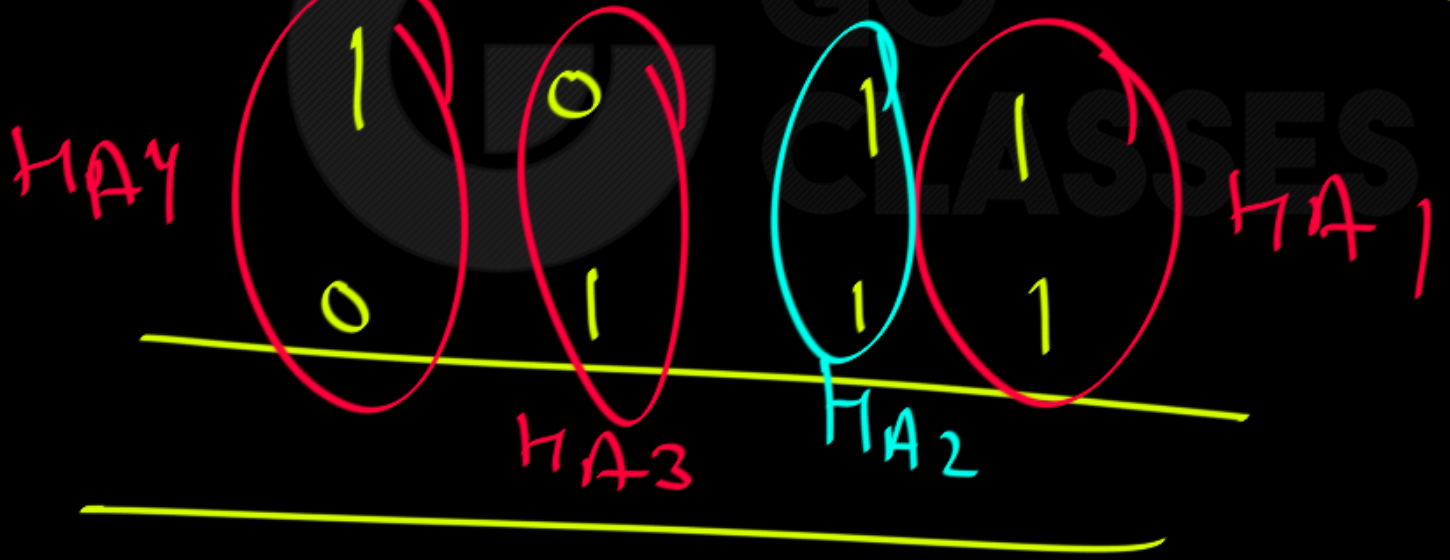
a) Ripple carry adder
b) Carry look-ahead adder

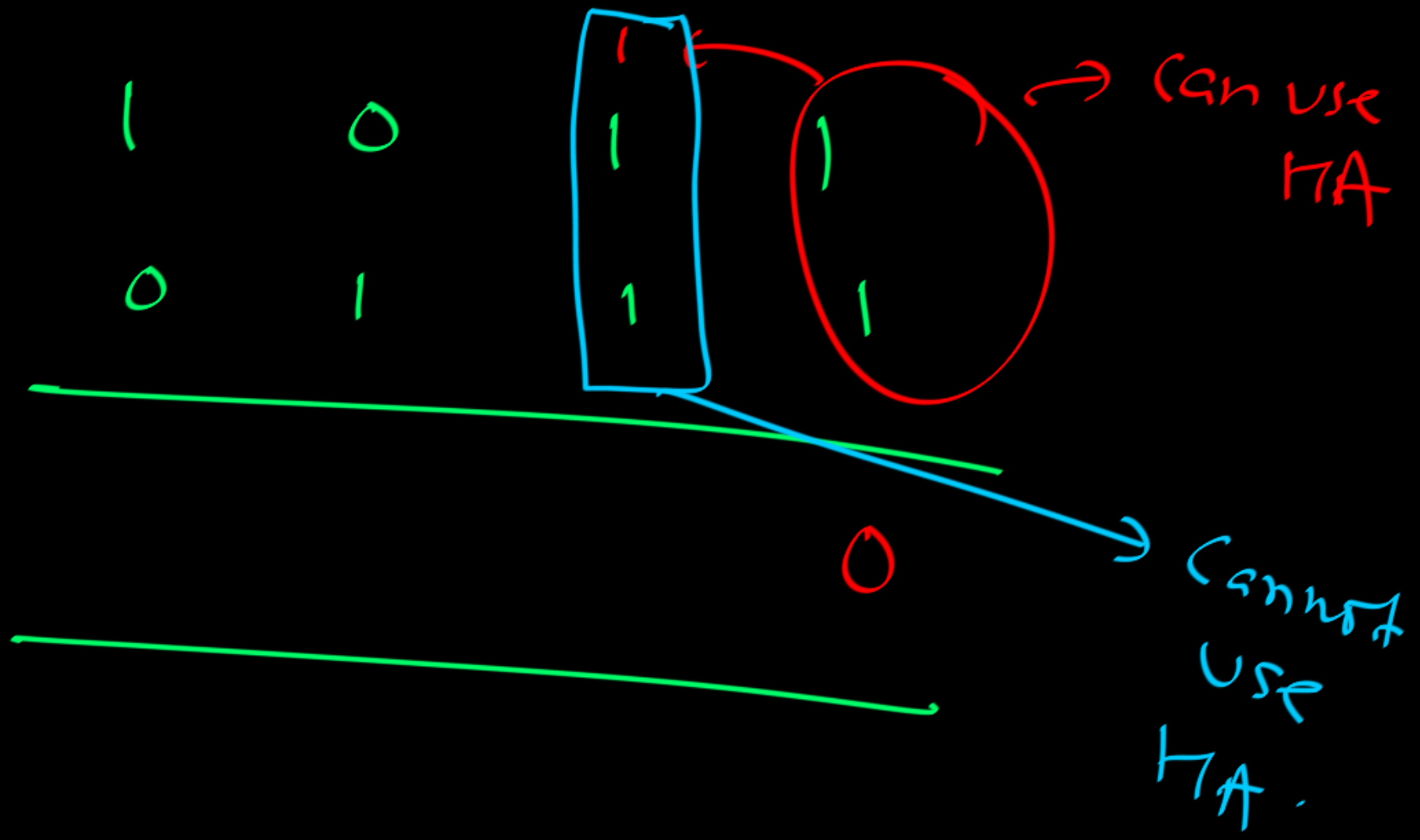
Ripple Carry Adder

- A binary adder is a digital circuit that produces the ***arithmetic sum of two binary numbers***.
- A binary adder can be constructed with ***full adders connected in cascade*** with the output carry form each full adder connected to the input carry of the next full adder in the chain.
- The ***four-bit adder*** is a typical example of a ***standard component*** .It can be used in many application involving arithmetic operations.

Q: 1011 + 0111

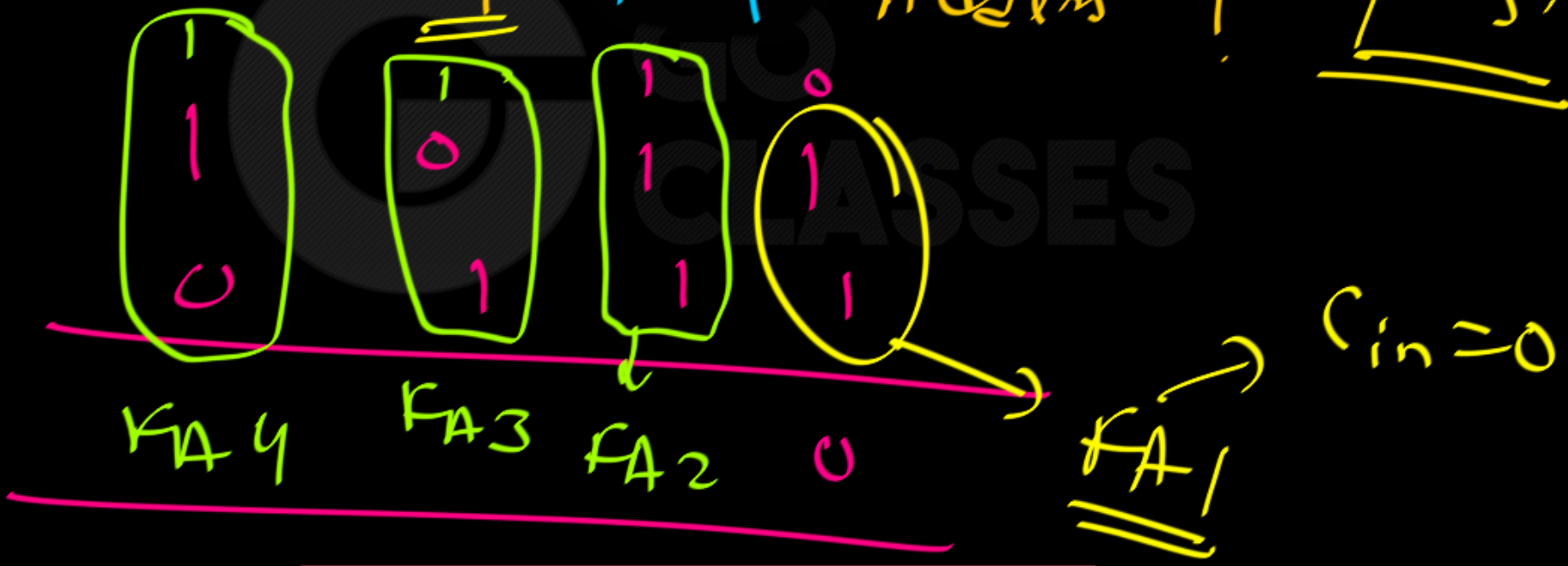
Can I use 4 Half Adders ? ⇒ No




1
0
0
1
1
1
1
1
1
0
Can use HA
Cannot use HA.

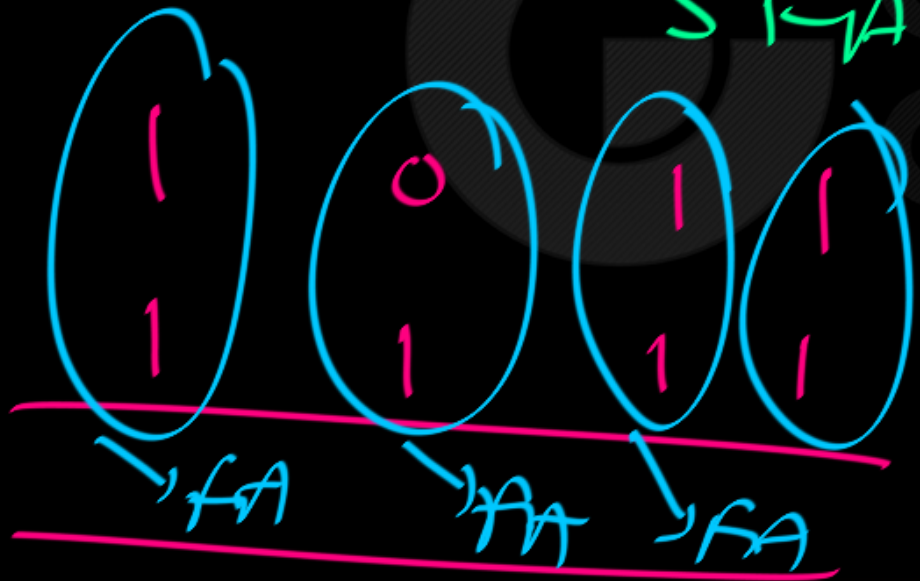
Q: 1011 + 0111

Can I use 4 full Adders ? Yes.



Q: A, B ⇒ 4 bit numbers;

A + B : Can we use 1 HA; 3 FA ?

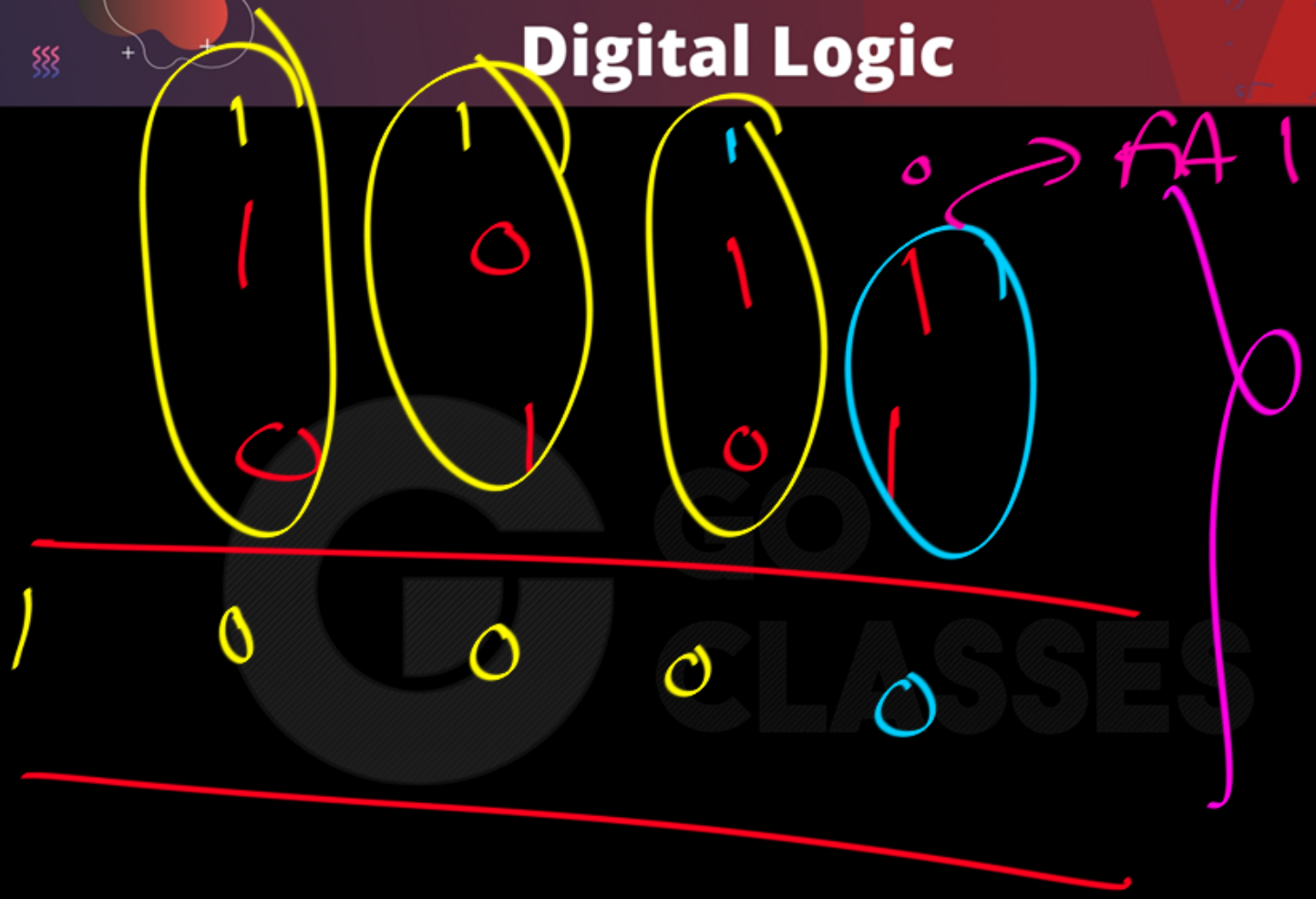


Yes.

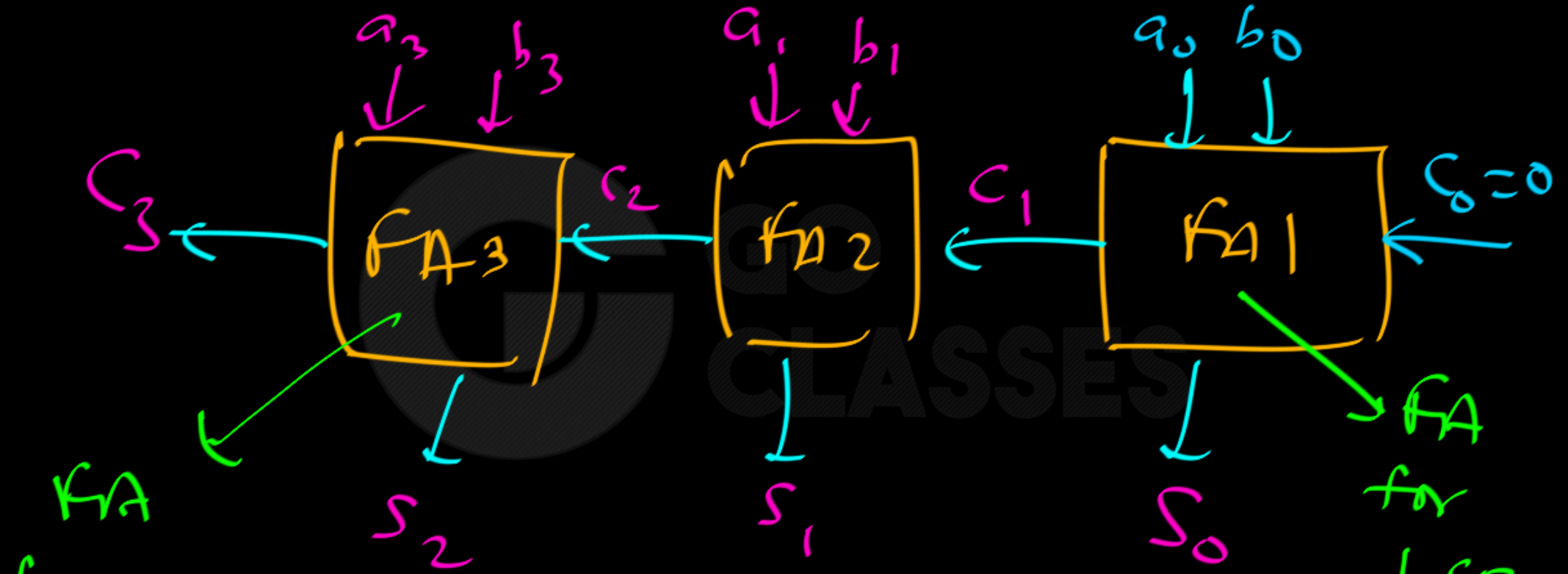
We can use HA in LSB position; But we must have FA in others.

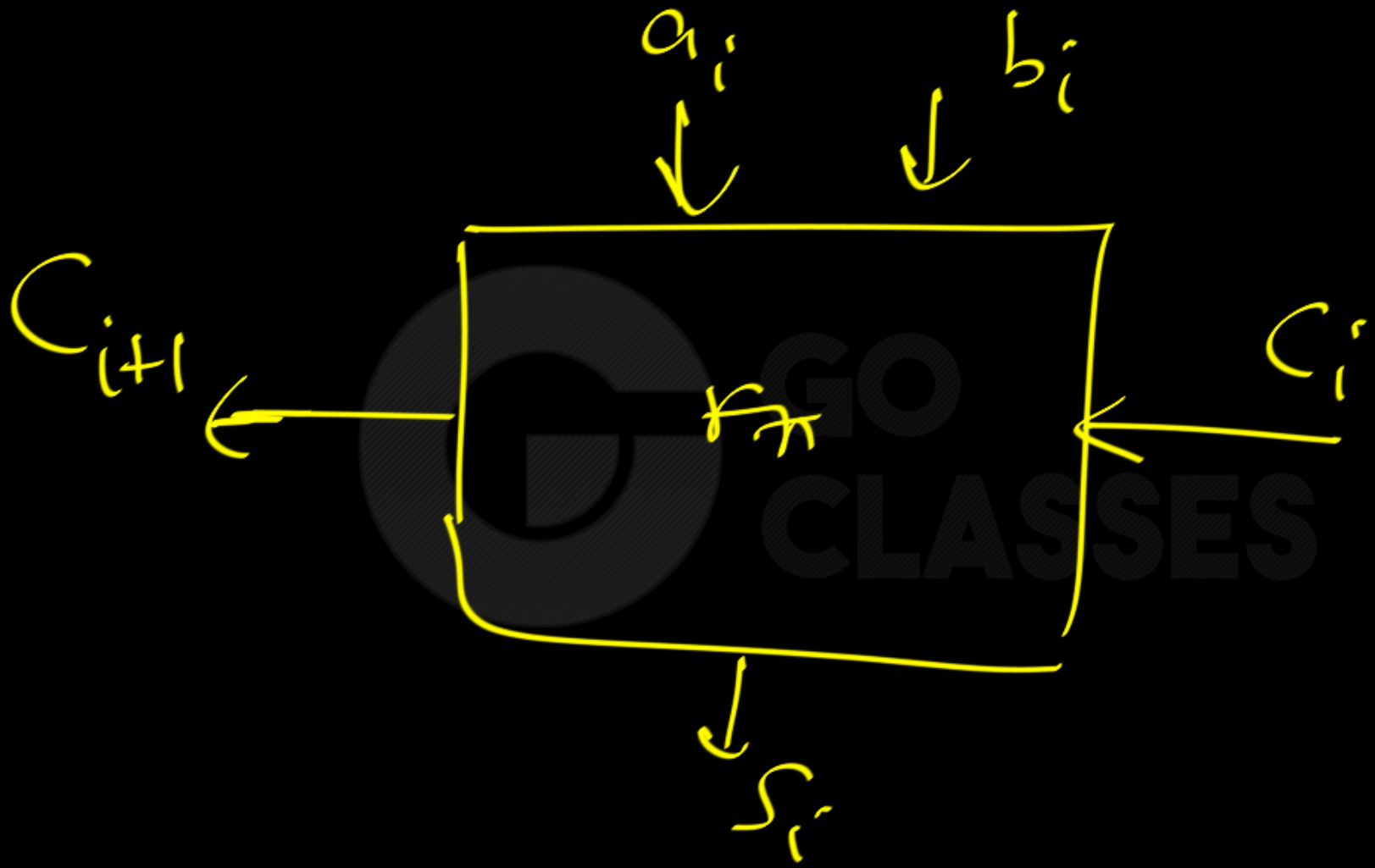
Ripple Carry Adder:

→ Circuit that Resembles our Hand Computation of binary Addition.


1 1 0 1
1 0 1 1
0 1 0 1
0 FA 1
1 0 0 0 0

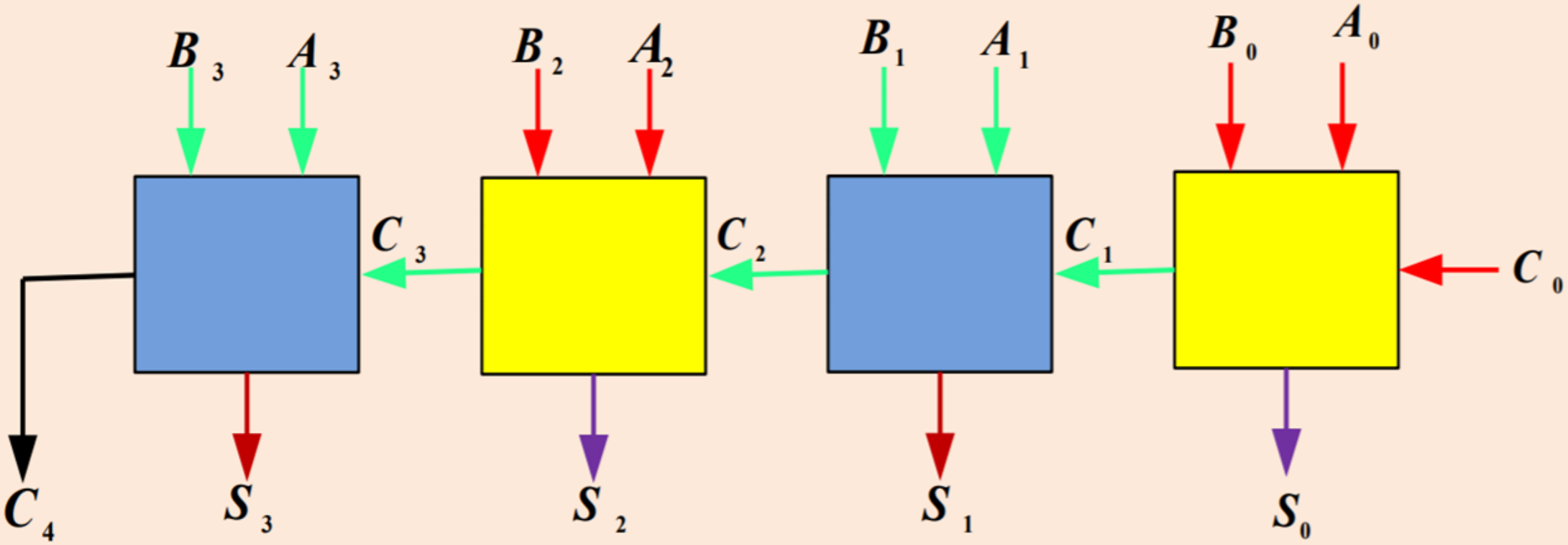
3 bit Addition:- $a_2 a_1 a_0 + b_2 b_1 b_0$




a_i
b_i
c_{i+1}
f_n
c_i
S_i

Addition of n-bit numbers requires a chain of n full adders OR a chain of one-half adder and n-1 full adders.

In the former case, the input carry to the least significant position is fixed at 0.



Four-bit Adder (Ripple Carry Adder)

- The input carry to the adder is C_0 and it ripples through the full adders to the output carry C_4.
- n-bit binary adder requires n full adders.

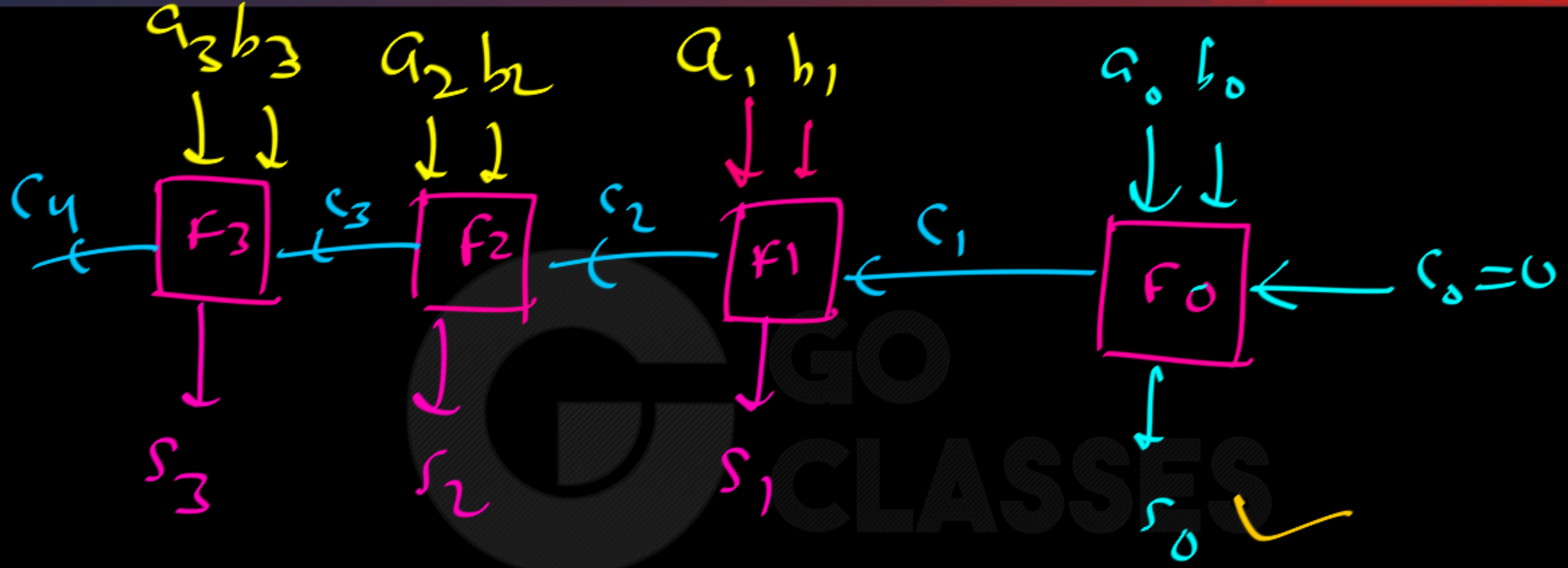
Example:

$A + B$ $(A = 1011)$ and $(B = 0011)$

Subscript i	3	2	1	0	
Input Carry	0	1	1	0	C_i
A	1	0	1	1	A_i
+					
B	0	0	1	1	B_i
Sum	1	1	1	0	S_i
Output Carry	0	0	1	1	C_{i+1}

$C_0 = 0$

The bits are added with full adders, starting from the least significant position (subscript 0), to form the sum bit and carry bit. The input carry C_0 in the least significant position must be 0. The value of C_{i+1} in a given significant position is the output carry of the full adder. This value is transferred into the input carry of the full adder that adds the bits one higher significant position to the left. The sum bits are thus generated starting from the rightmost position and are available as soon as the corresponding previous carry bit is generated. All the carries must be generated for the correct sum bits to appear at the outputs.



initially; S_i are invalid (intuitively; these are unavailable)

Initially ; we have ;

$a_3 a_2 a_1 a_0$; $b_3 b_2 b_1 b_0$; C_0

ADD

initially; c_1 to c_4 ⇒ invalid (intuitively unavailable)

s_4, c_4 depend on c_3; c_3 depends on c_2; c_2 depends on c_1; c_1 depends on c_0

initially; C_1 to C_4, S_0 to S_3 } invalid

They will become valid when Carry has enough time to propagate through the circuit (Full Adders)

Carry Propagation

- The addition of $A + B$ binary numbers in ***parallel*** implies that all the bits of A and B are available for computation at the same time.
- As in any combinational circuit, the signal must ***propagate*** through the gates before the correct output sum is available.
- The output will not be correct unless the signals are given enough time to propagate through the gates connected form the input to the output.
- The longest ***propagation delay time*** in an adder is the time it takes the carry to propagate through the full adders.

Carry Propagation

The addition of two binary numbers in parallel implies that all the bits of the augend and addend are available for computation at the same time. As in any combinational circuit, the signal must propagate through the gates before the correct output sum is available in the output terminals. The total propagation time is equal to the propagation delay of a typical gate, times the number of gate levels in the circuit. The longest propagation delay time in an adder is the time it takes the carry to propagate through the full adders. Since each bit of the sum output depends on the value of the input carry, the value of S_i at any given stage in the adder will be in its steady-state final value only after the input carry to that stage has been propagated. In this regard, consider output S_3 in Fig. 4.9. Inputs A_3 and B_3 are available as soon as input signals are applied to the adder. However, input carry C_3 does not settle to its final value until C_2 is available from the previous stage. Similarly, C_2 has to wait for C_1 and so on down to C_0. Thus, only after the carry propagates and ripples through all stages will the last output S_3 and carry C_4 settle to their final correct value.

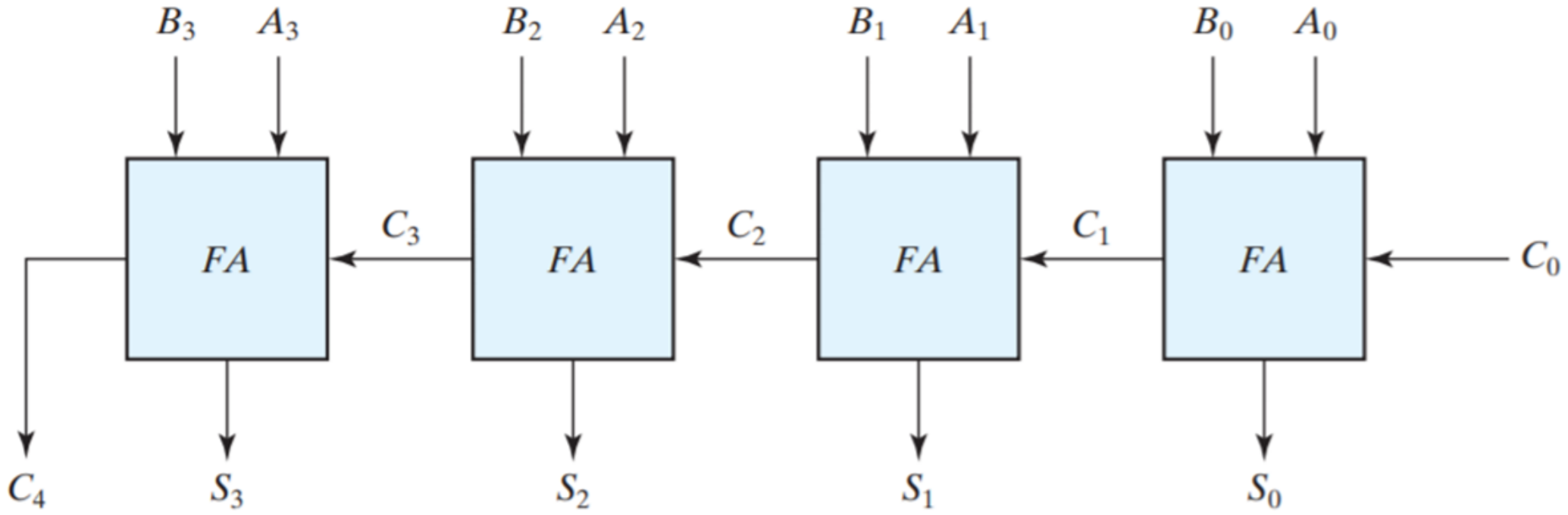
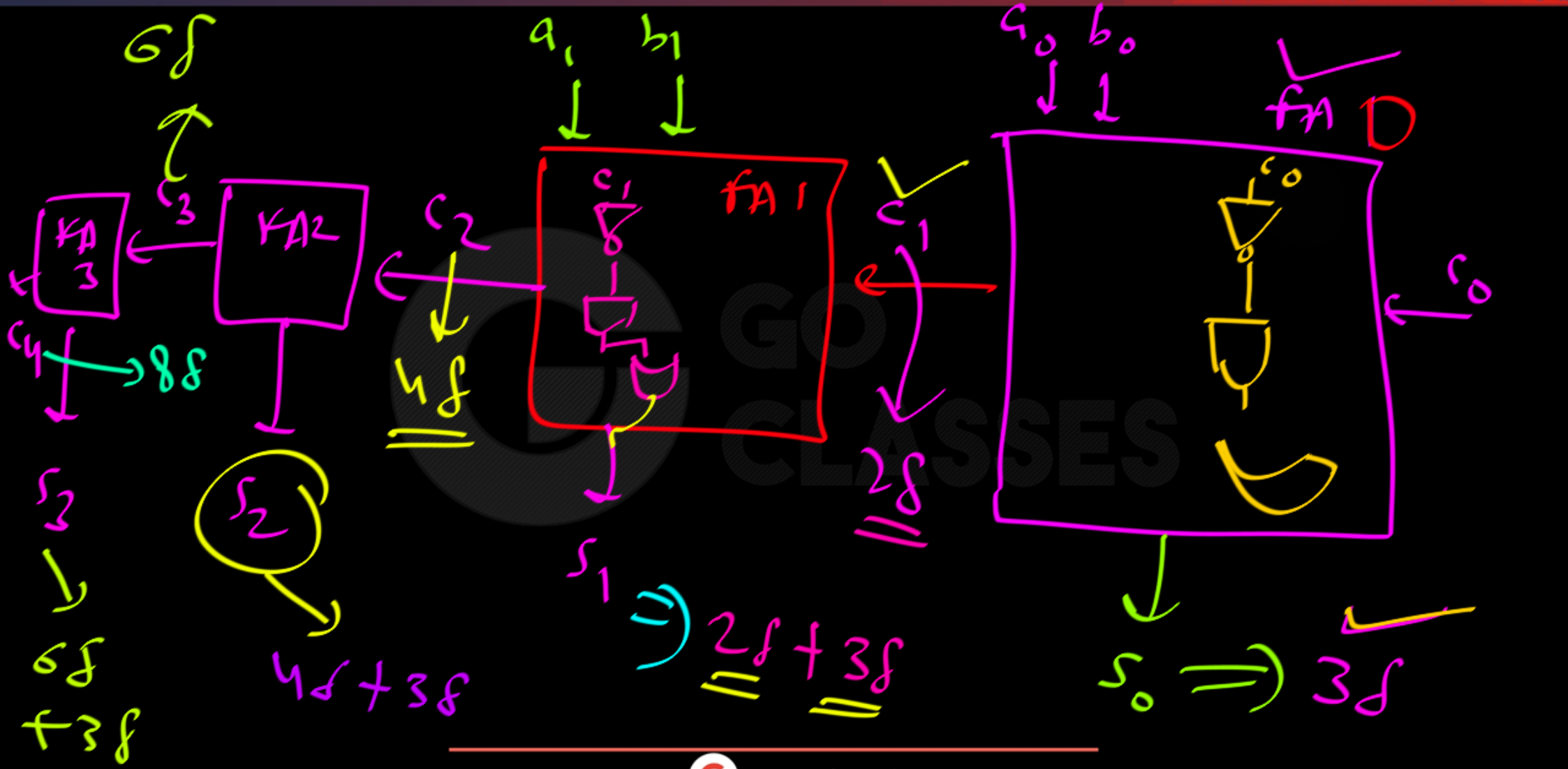


FIGURE 4.9
Four-bit adder

- The signal form the carry input C_i to the output carry C_{i+1} propagates through an ***AND*** gate and an ***OR*** gate, which equals ***2 gate levels***.
  - If there are ***4*** full adders in the binary adder, the output carry C_4 would have ***2×4=8 gate levels***, form C_0 to C_4
  - For an n-bit adder, $2n$ gate levels for the carry to propagate form input to output are required.

Q: Assume every FA is created using basic gates; then what is the Delay of Ripple carry adder of 4 bit addition;

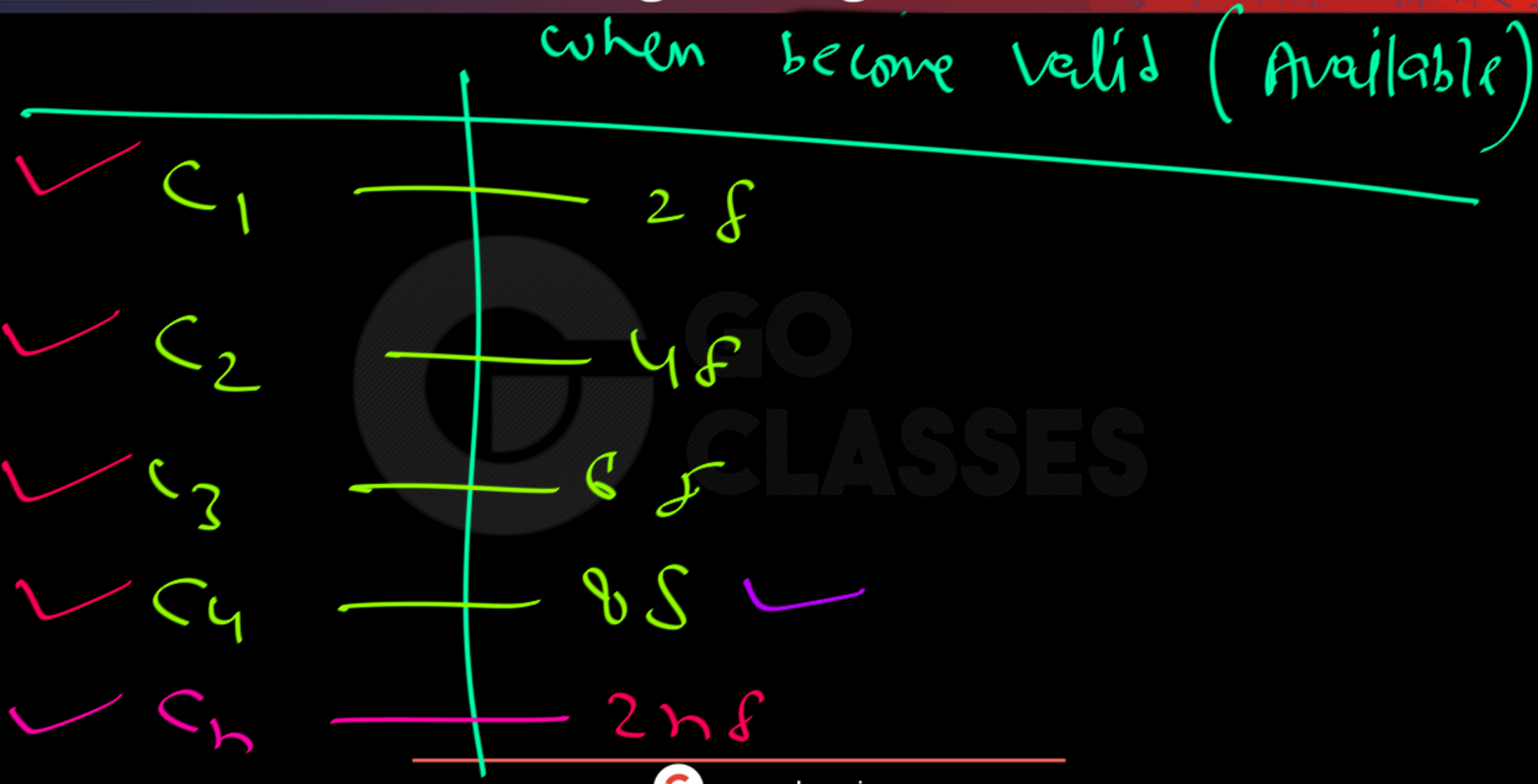
If every basic gate has Delay δ.

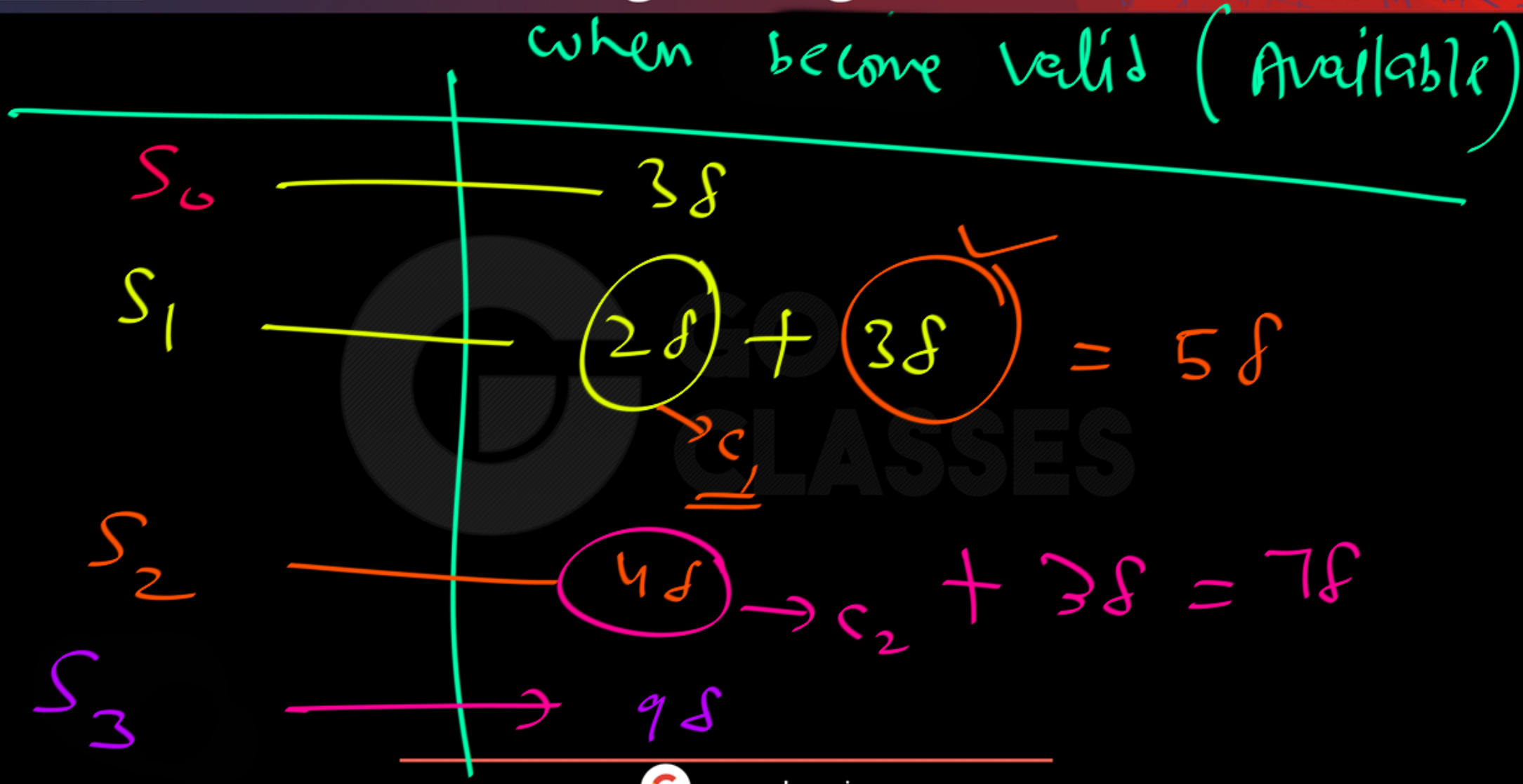

FA 3
FA2
FA 1
FA 0
c_3
c_2
c_1
c_0
a_1
b_1
a_0
b_0
S_3
S_2
S_1
S_0
$6\delta + 3\delta$
$4\delta + 3\delta$
$2\delta + 3\delta$
3δ
8δ
4δ
2δ

At $t=0$; All S_i; C_1 to C_4 invalid (Not Available)

At $t=2\delta$; $C_1 \Rightarrow$ valid

At $t=3\delta$; $S_0 \Rightarrow$ valid


when become valid (Available)
C1
2f
C2
4f
C3
6f
C4
8f
Cn
2nf


when become valid (Available)
S_0
3δ
S_1
2δ + 3δ = 5δ
c_1
S_2
4δ → c_2 + 3δ = 7δ
S_3
9δ

last sum : $S_{n-1} = (2n-2)\delta + 3\delta$

$= C_{n-1}$

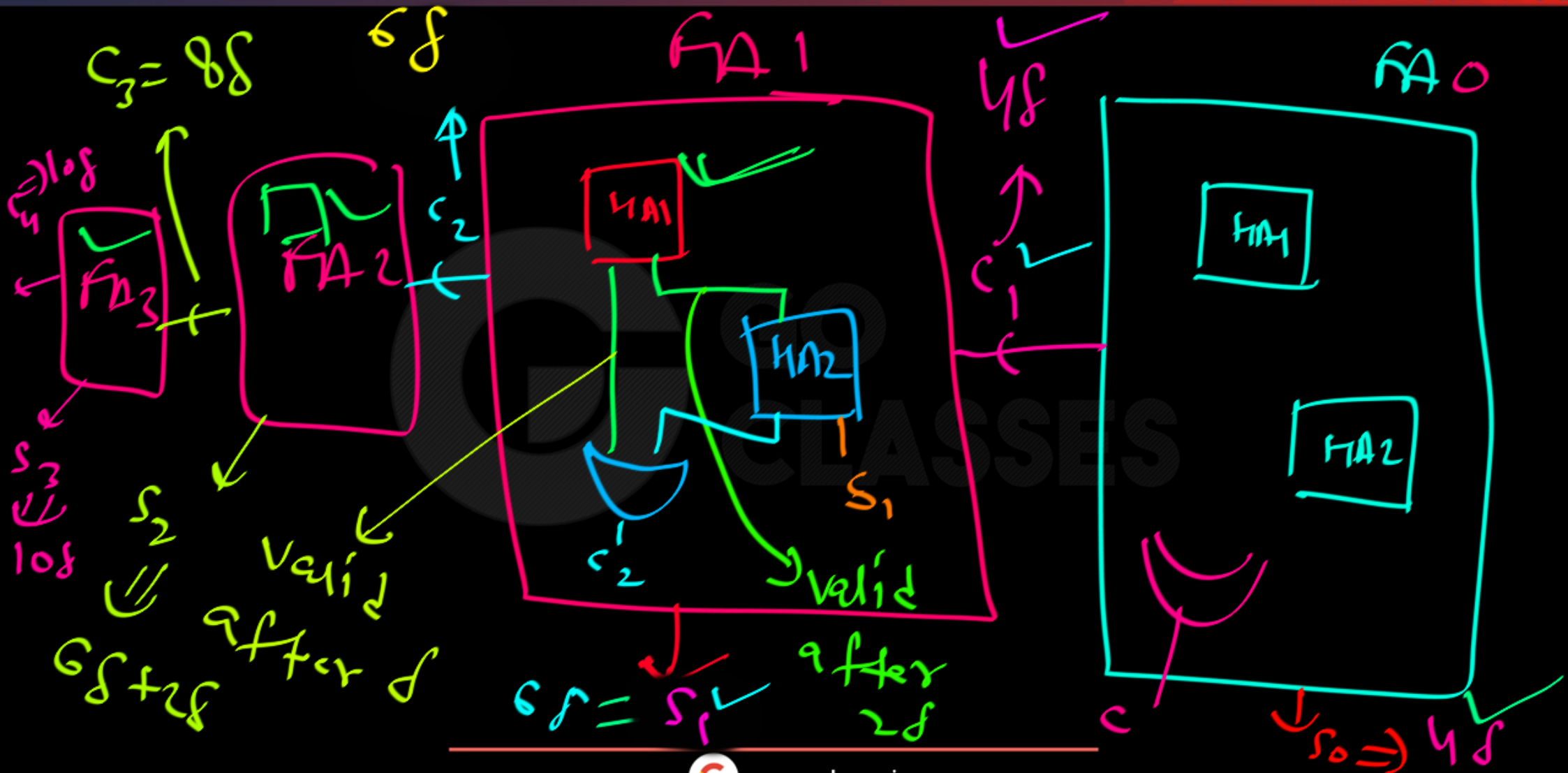
$= 2n\delta + \delta$

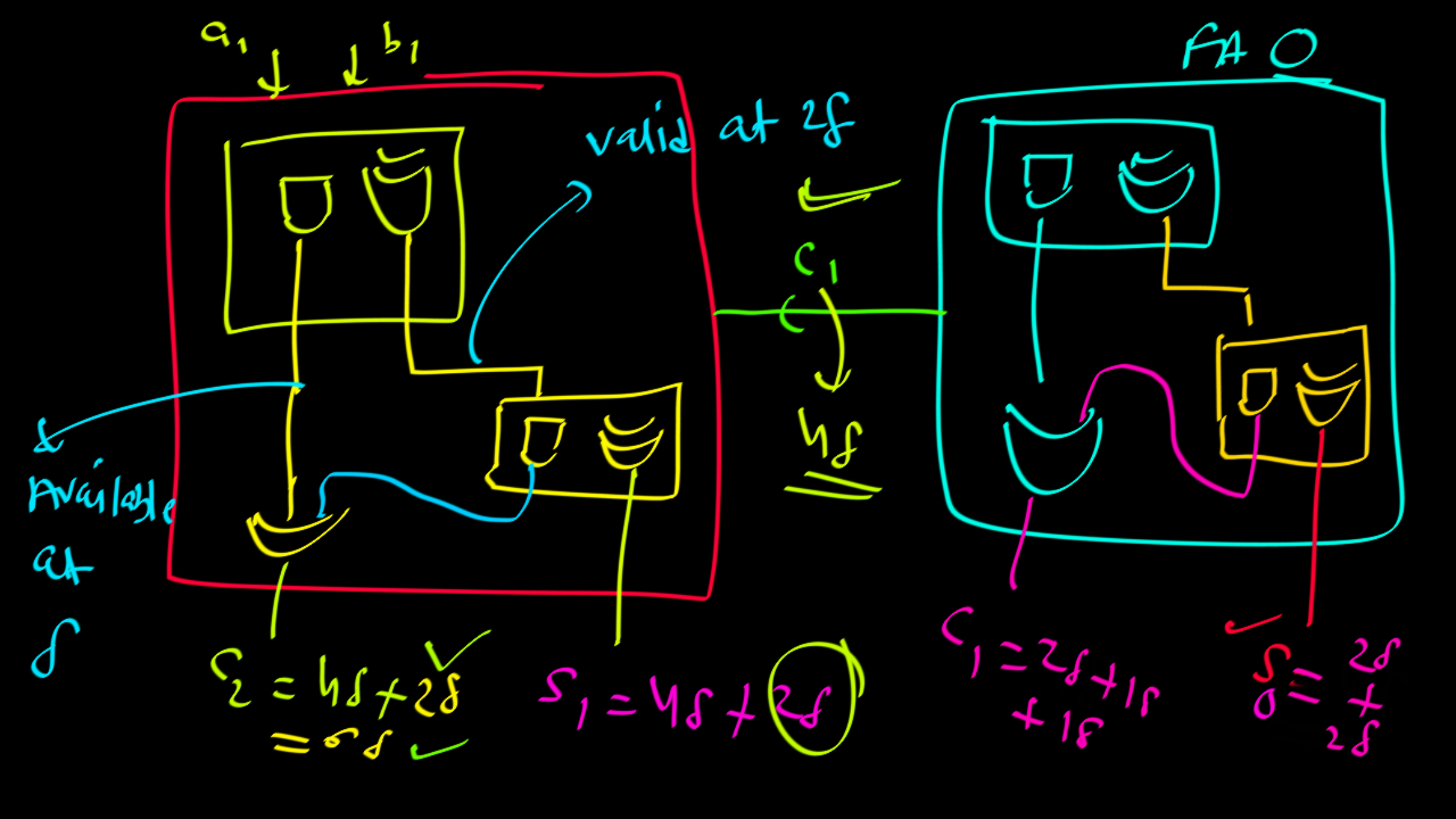
$= (2n+1)\delta$

Q: FA is implemented using 2 HA, 1 OR gate ; AND, OR, NOT ⇒ Delay 8

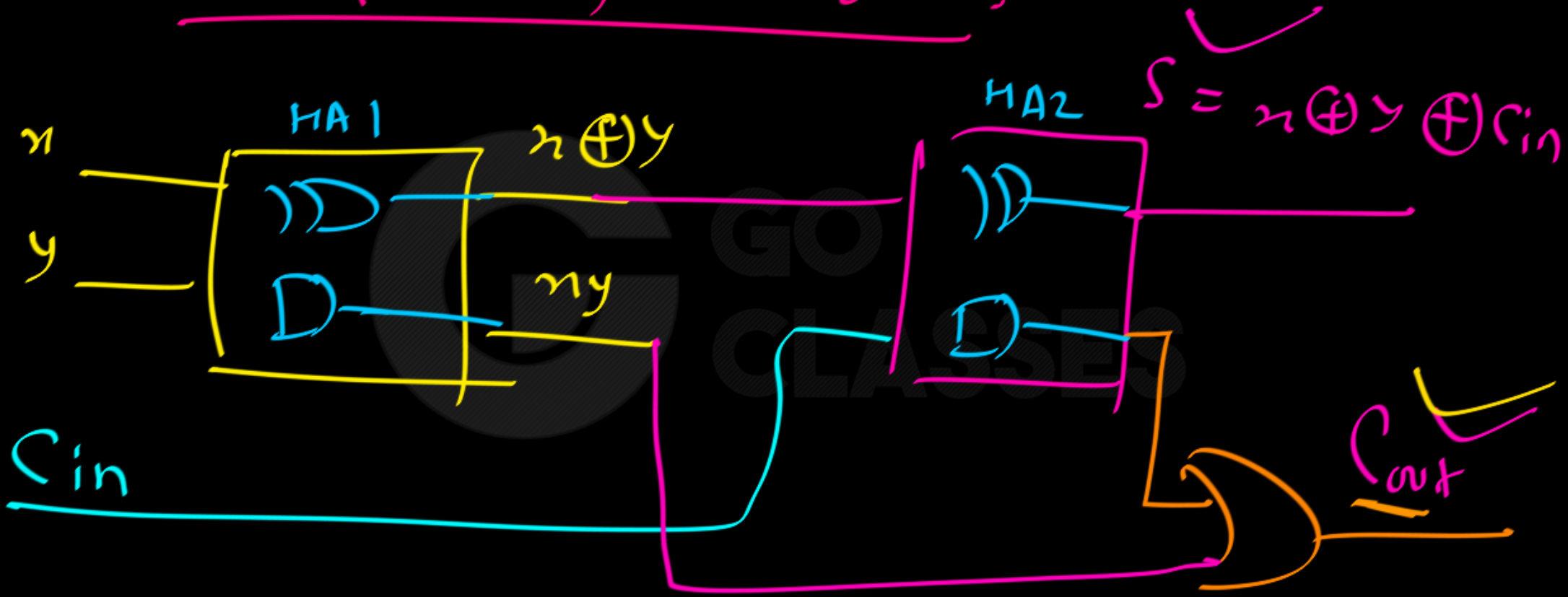
let T_{XOR} = Delay 2δ

then Delay of ripple carry Adder ?

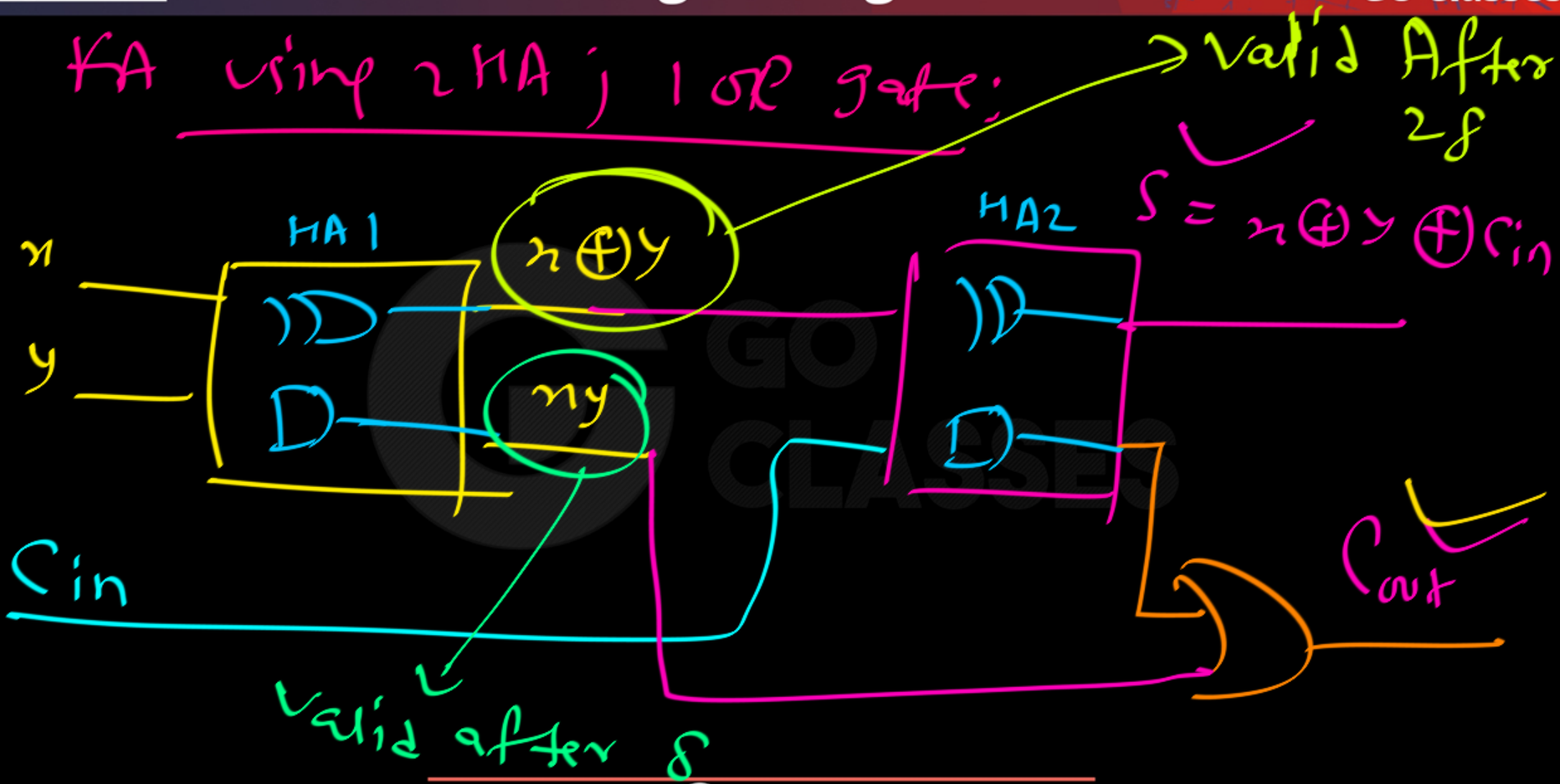

FA 1
FA 0
FA3
FA2
HA1
HA2
HA1
HA2
Valid
Valid
after 2δ
after δ
www.goclasses.in


a_1
b_1
FA 0
valid at 2δ
c_1
4δ
&
Available
at
δ
$c_2 = 4\delta + 2\delta$
$= 6\delta$
$s_1 = 4\delta + 2\delta$
$c_1 = 2\delta + 1\delta$
$+ 1\delta$
$s_0 = 2\delta$
$+ 2\delta$

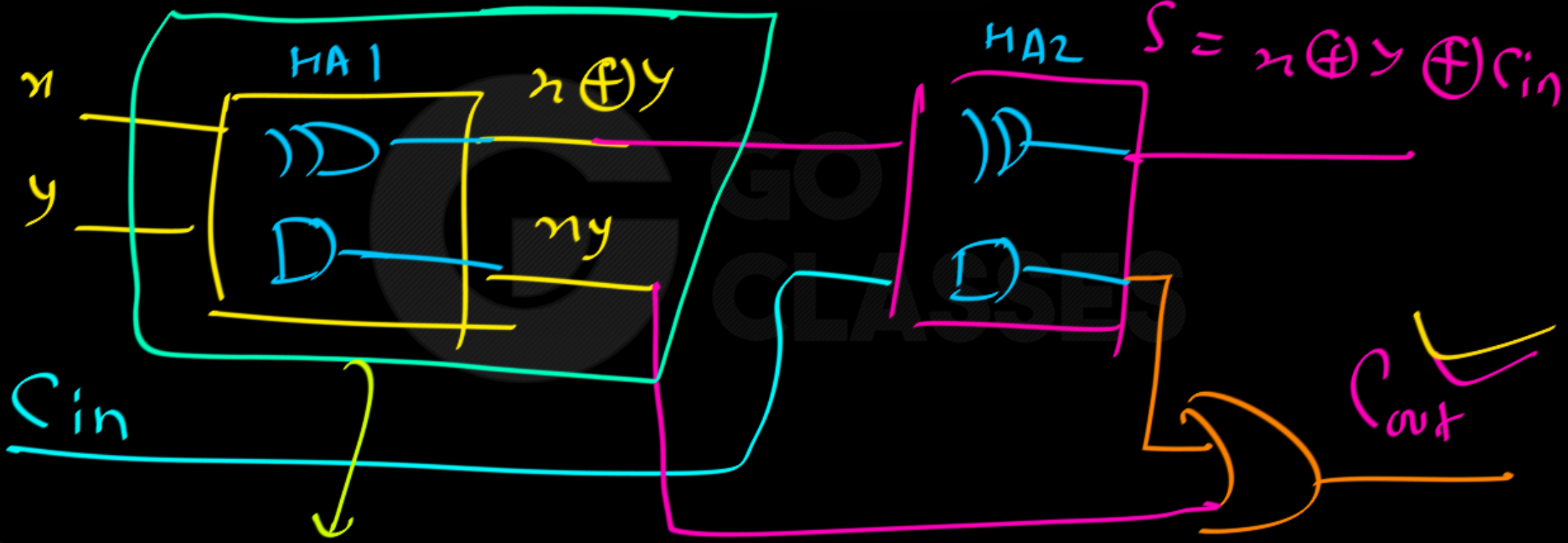
FA using 2 HA & 1 OR gate:



FA using 2 HA & 1 OR gate:

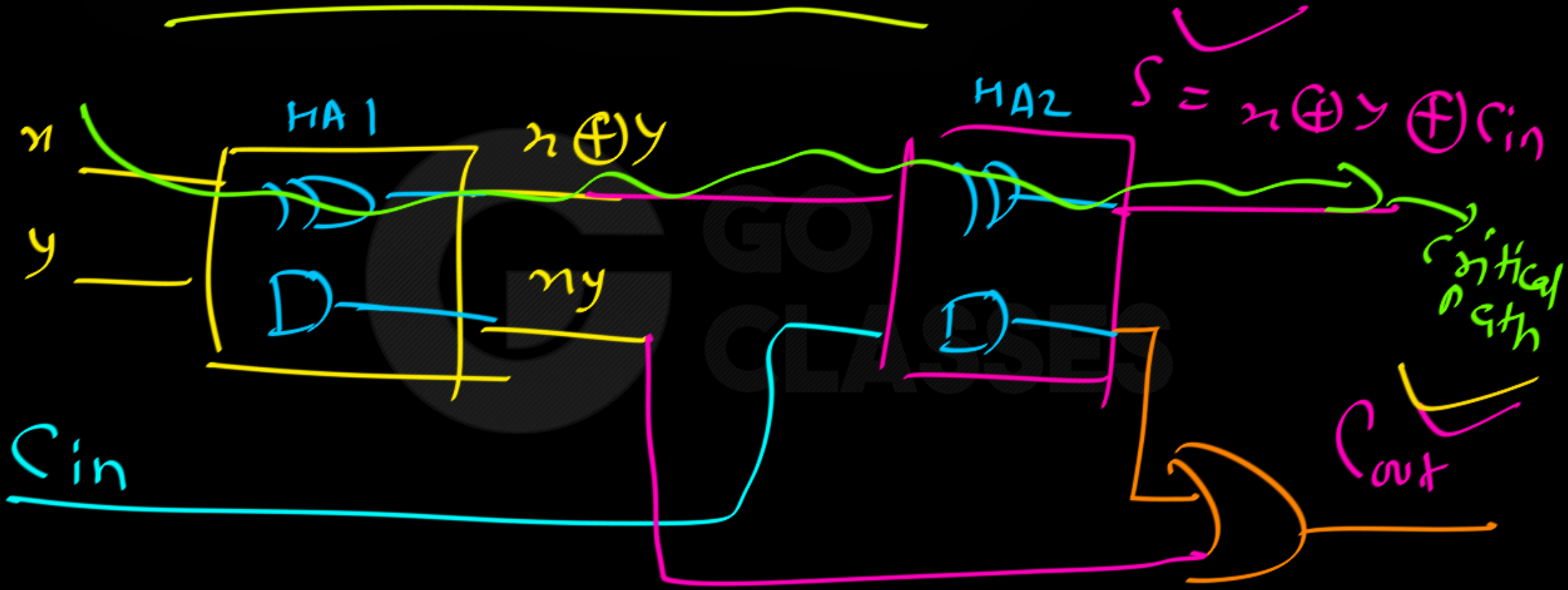


Which part depends on previous carry C_{in}?



HA1 ⇒ This part is independent of C_{in}.

Critical Path for S:

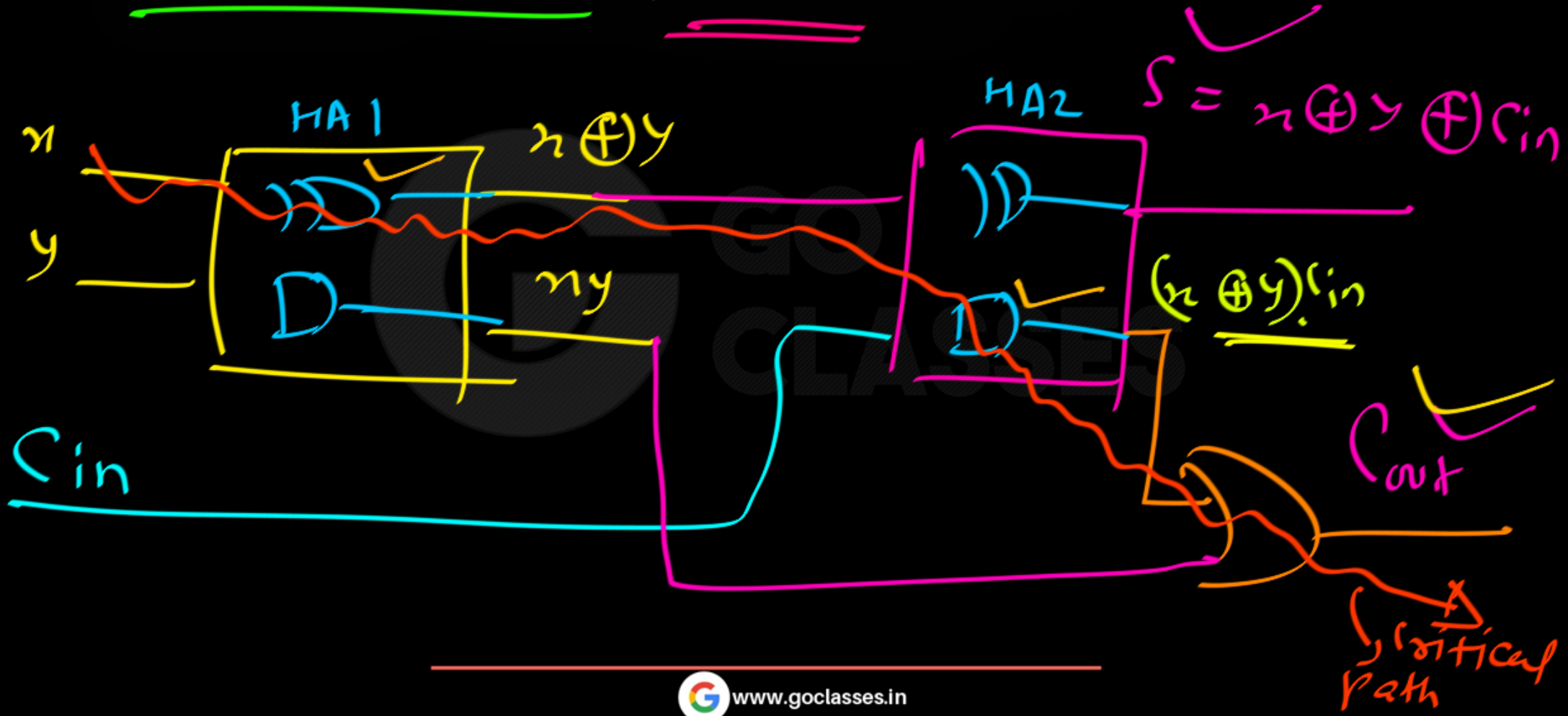


Delay for S ⇒ Critical path of S
(path of maximum Delay)

⇒ EXOR → EXOR

⇒ 4f

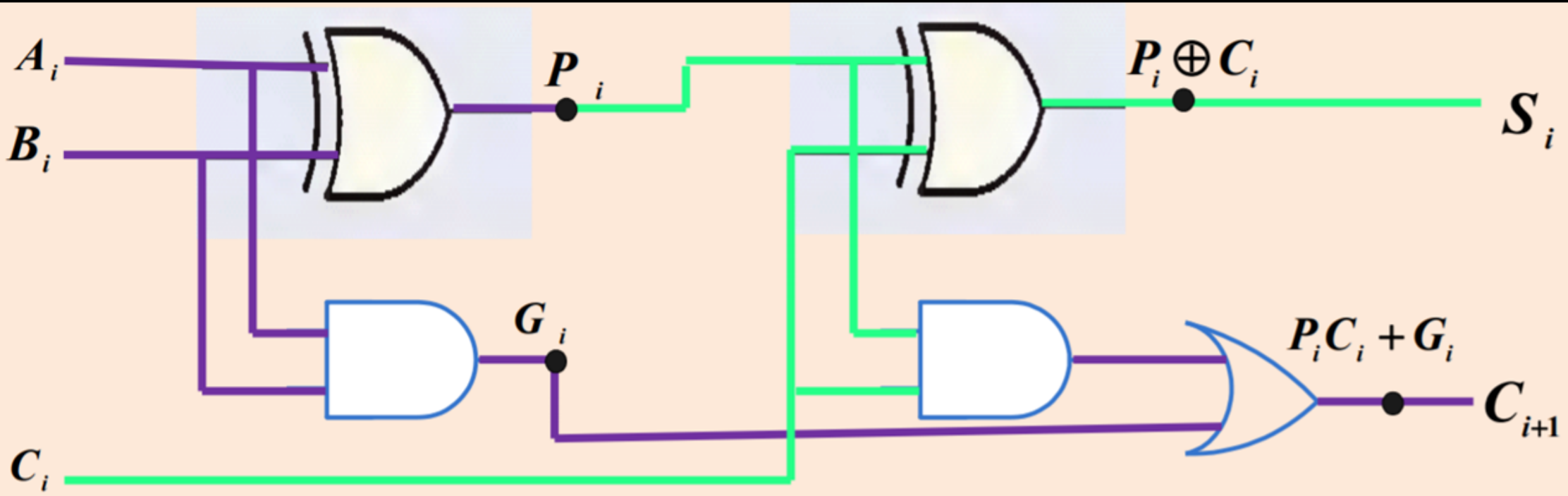
Critical path of Cout :



Delay for C_{out} = Critical Path for C_{out}

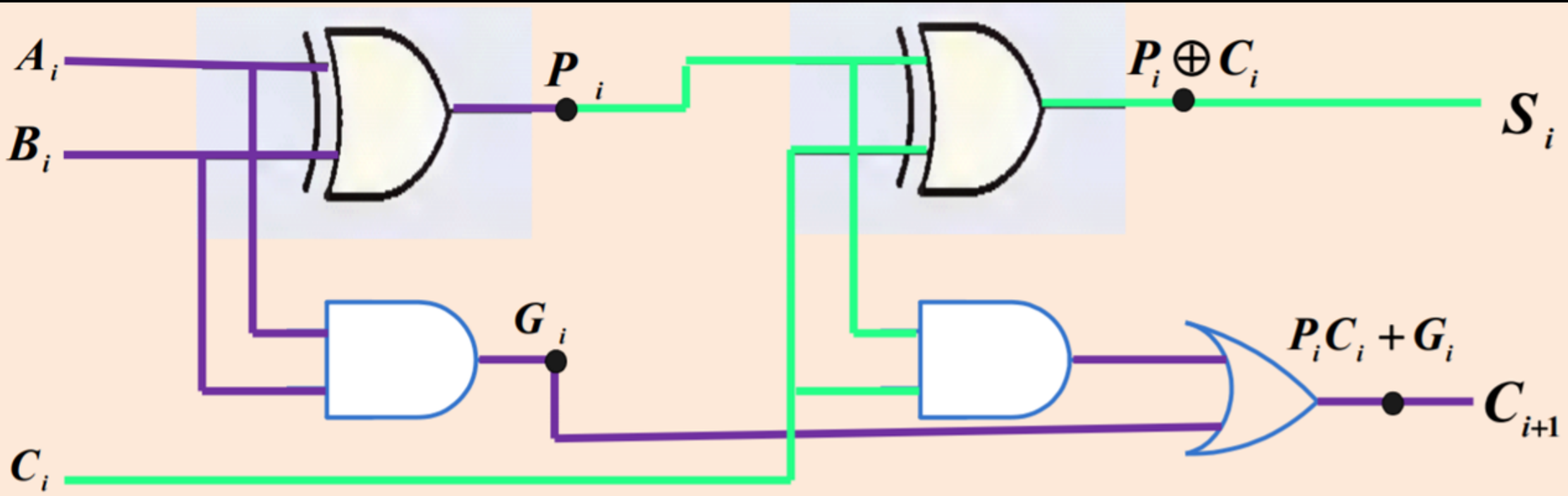
Ex-OR → AND → OR

$= 2\delta + \delta + \delta = 4\delta$



Full Adder with P and G

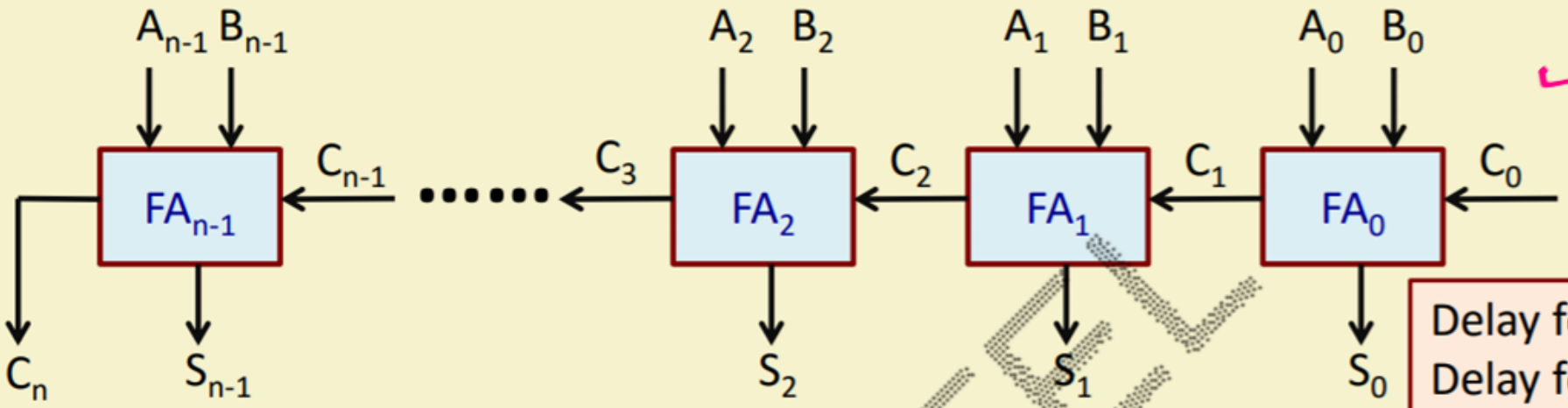
The number of gate levels for the carry propagation can be found from the circuit of the full adder. The circuit is redrawn with different labels in Fig. 4.10 for convenience. The input and output variables use the subscript i to denote a typical stage of the adder. The signals at P_i and G_i settle to their steady-state values after they propagate through their respective gates. These two signals are common to all half adders and depend on only the input augend and addend bits. The signal from the input carry C_i to the output carry C_{i+1} propagates through an AND gate and an OR gate, which constitute two gate levels. If there are four full adders in the adder, the output carry C_4 would have $2 \times 4 = 8$ gate levels from C_0 to C_4. For an n-bit adder, there are $2n$ gate levels for the carry to propagate from input to output.



Full Adder with P and G

NOTE :

Although the adder—or, for that matter, any combinational circuit—will always have some value at its output terminals, the outputs will not be correct unless the signals are given enough time to propagate through the gates connected from the inputs to the outputs.



Using only Basic Gates.

Delay for C_1	$= 2\delta$
Delay for C_2	$= 4\delta$
Delay for C_{n-1}	$= 2(n-1)\delta$
Delay for C_n	$= 2n\delta$

Two numbers:	$A_{n-1}...A_2A_1A_0$ and $B_{n-1}...B_2B_1B_0$
Input carry:	C_0
Sum:	$S_{n-1}...S_2S_1S_0$
Output carry:	C_n

Delay for S_0	$= 3\delta$
Delay for S_1	$= 2\delta + 3\delta = 5\delta$
Delay for S_2	$= 4\delta + 3\delta = 7\delta$
Delay for S_{n-1}	$= 2(n-1)\delta + 3\delta = (2n+1)\ \delta$

Delay is proportional to n

- Main drawback of ripple-carry adder: (RCA) :

– The total delay is proportional to the number of bits n.

– Performance degradation for large values of n.

– The main bottleneck is the carry, which propagates sequentially from one stage to the next.

- How this can be overcome?

– Generate all the carry bits in parallel before the actual addition starts.

– After the initial delay, all the additions can be done in parallel.

Why RCA is slow?

Because S_3 Depends on C_3

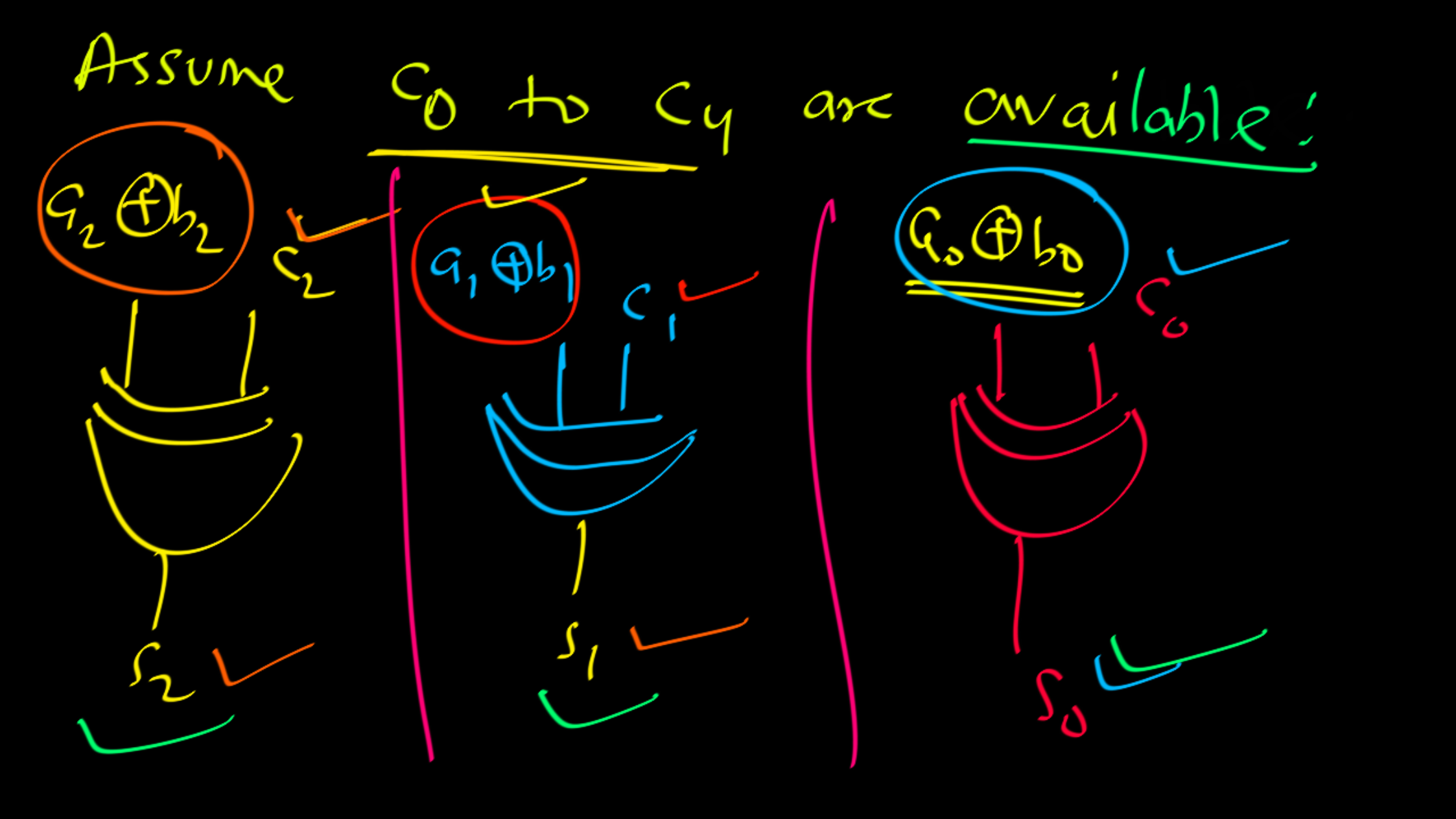
C_3 Depends on C_2

C_2 Depends on C_1

C_1 Depends on C_0

How to make our Adder fast?

Calculate Carry of each adder Directly. initially; first Calculate Carry of each adder: Now all carry are available.


Assume
c_0 to c_4 are
available!
$a_2 \oplus b_2$
c_2
s_2
$a_1 \oplus b_1$
c_1
s_1
$a_0 \oplus b_0$
c_0
s_0

Carry Look-ahead Adder (CLA)

Carry Look-ahead Adder (CLA)

- The propagation delay of an n-bit ripple carry order has been seen to be proportional to n.

– Due to the rippling effect of carry sequentially from one stage to the next.

- One possible way to speedup the addition.

– Generate the carry signals for the various stages in parallel.

– Time complexity reduces from O(n) to O(1).

– Hardware complexity increases rapidly with n.