Function Realization Using Multiplexer

Recap:

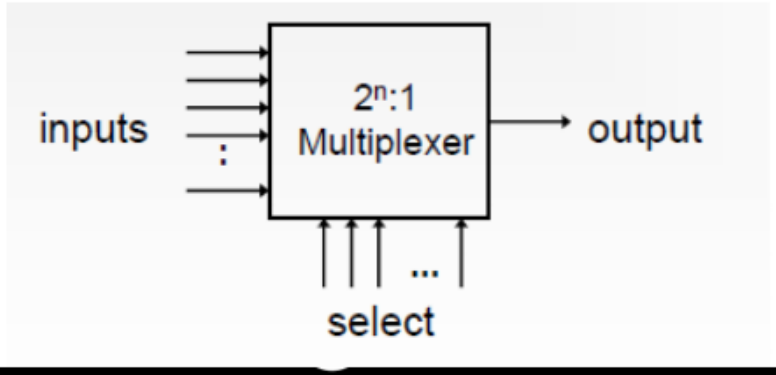
Mux

Multiplexor (MUX)

A multiplexer is a device which has

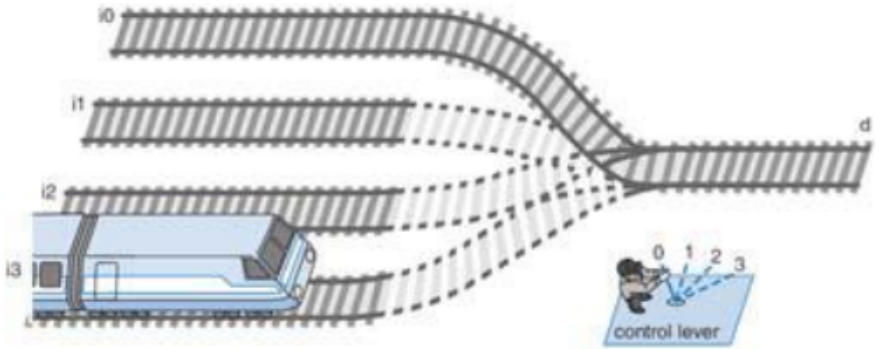
- a number of *inputlines*
- a number of *selectionlines*
- one *outputline*

- It steers one of 2^n inputs to a single output line, using *n* selection lines. Also known as a data selector.



Multiplexor (Mux)

- Mux: Another popular combinational building block
  - Routes one of its N data inputs to its one output, based on binary value of select inputs
    - 4 input mux needs 2 select inputs to indicate which input to route through
    - 8 input mux needs 3 select inputs
    - N inputs → $\log_2(N)$ selects
  - Like a railyard switch

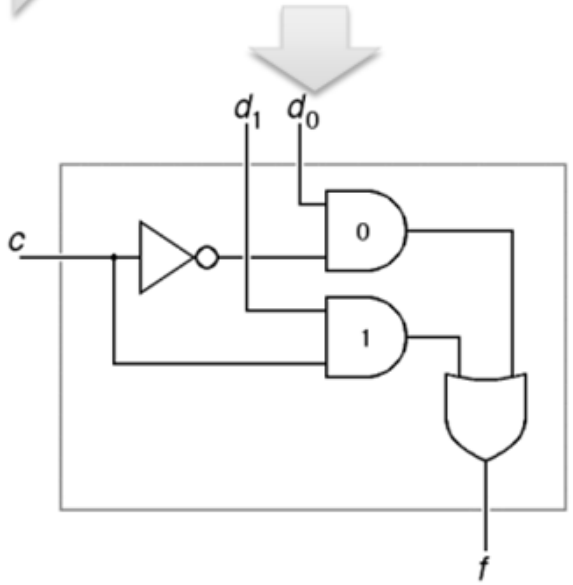


A 2-input multiplexor

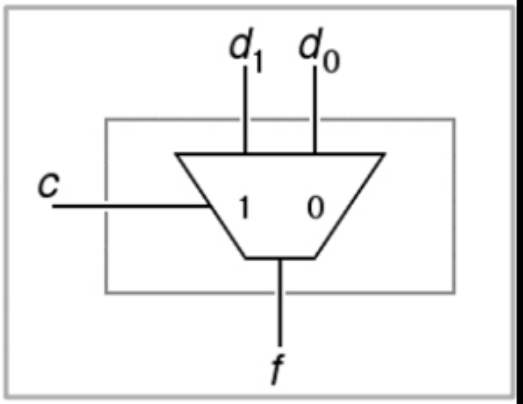
- Truth table for a multiplexor with 2 data inputs d_0 and d_1 and one control input c is as follows:

inputs			f
c	d_0	d_1	
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$$f = (d_0 \bar{c}) + (d_1 c)$$



logic circuit involving only 4 gates



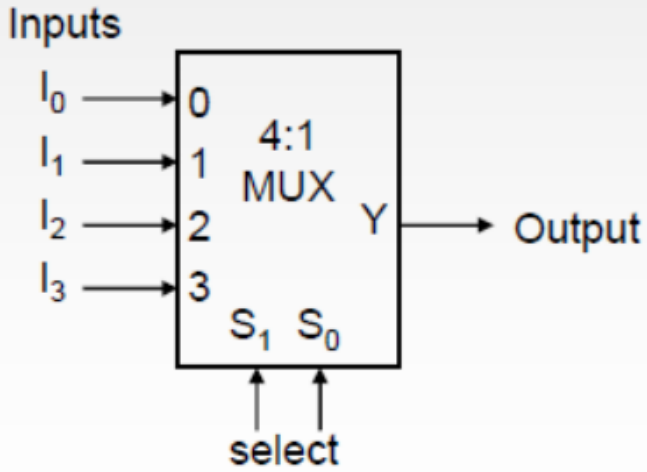
schematic symbol

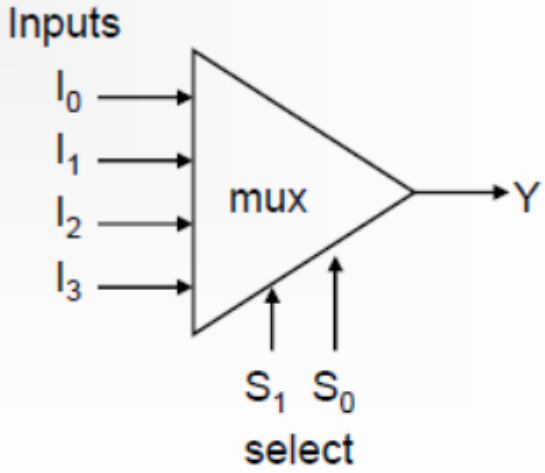
4-input multiplexor

- Truth table for a 4-to-1 multiplexer:

I_0	I_1	I_2	I_3	S_1	S_0	Y
d_0	d_1	d_2	d_3	0	0	d_0
d_0	d_1	d_2	d_3	0	1	d_1
d_0	d_1	d_2	d_3	1	0	d_2
d_0	d_1	d_2	d_3	1	1	d_3

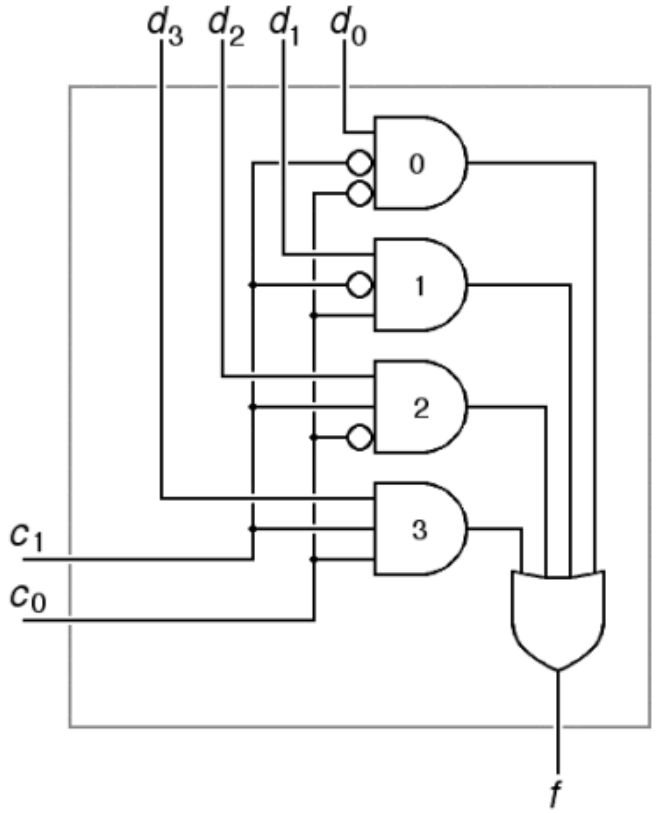
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

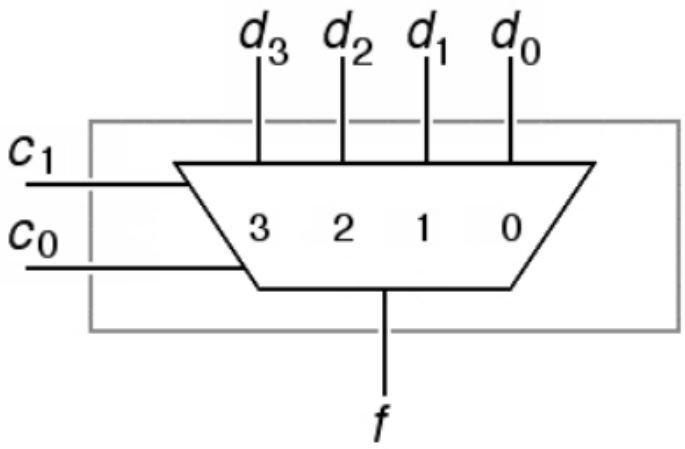




Gate-level design for a 4-input multiplexor

$$f = (d_0\, c'_1\, c'_0) + (d_1\, c'_1\, c_0) + (d_2\, c_1\, c'_0) + (d_3\, c_1\, c_0)$$





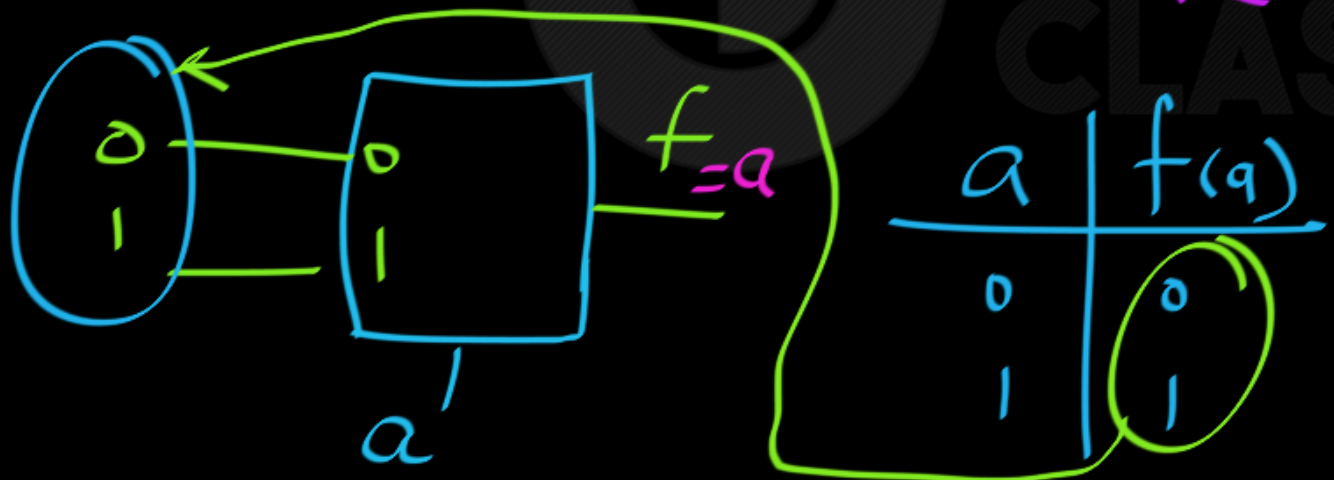
schematic symbol

Next Topic:

Single $2^n * 1$ Mux is enough for any(every) n-var function.

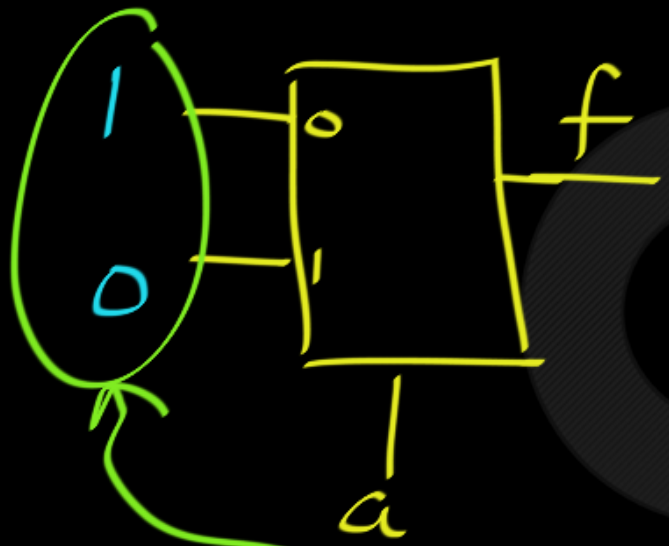
$f(a)$ ⇒ one 2 x 1 mux is Enough.
without any Additional Hardware.

$f(a) = a$ ⇒ one variable ⇒ on select line of 2x1 mux



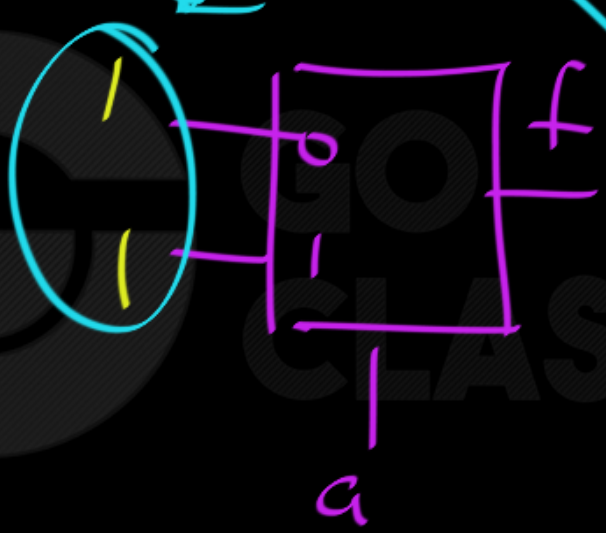
a	f(a)
0	0
1	1

$f(a) = \overline{a}$



a	f
0	1
1	0

$f(a) = 1$



a	f
0	1
1	1

$f(a,b)$ ⇒ one 4x1 mux is Enough.

||

2 Var ⇒ on select lines of 4x1 mux

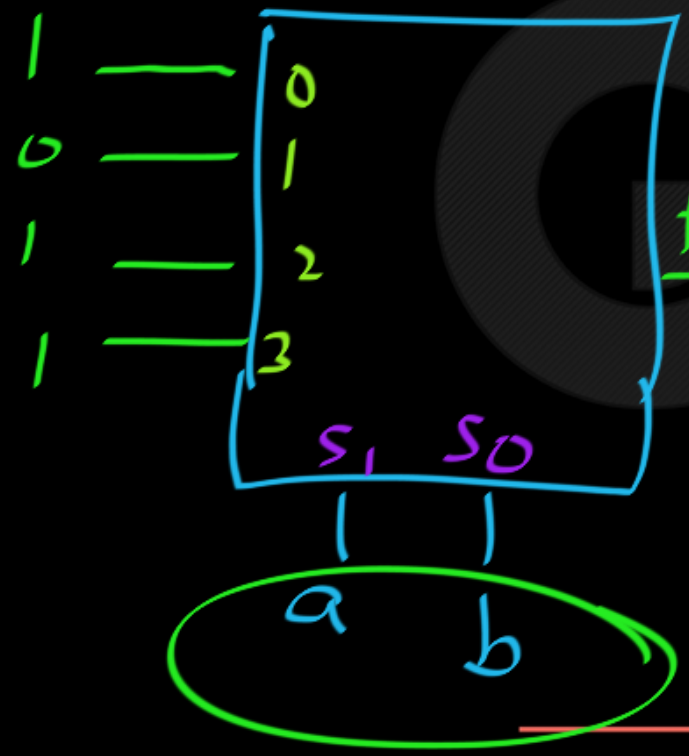
No Additional Hardware.

2^2 select line

Data input : Put the Truth Table.

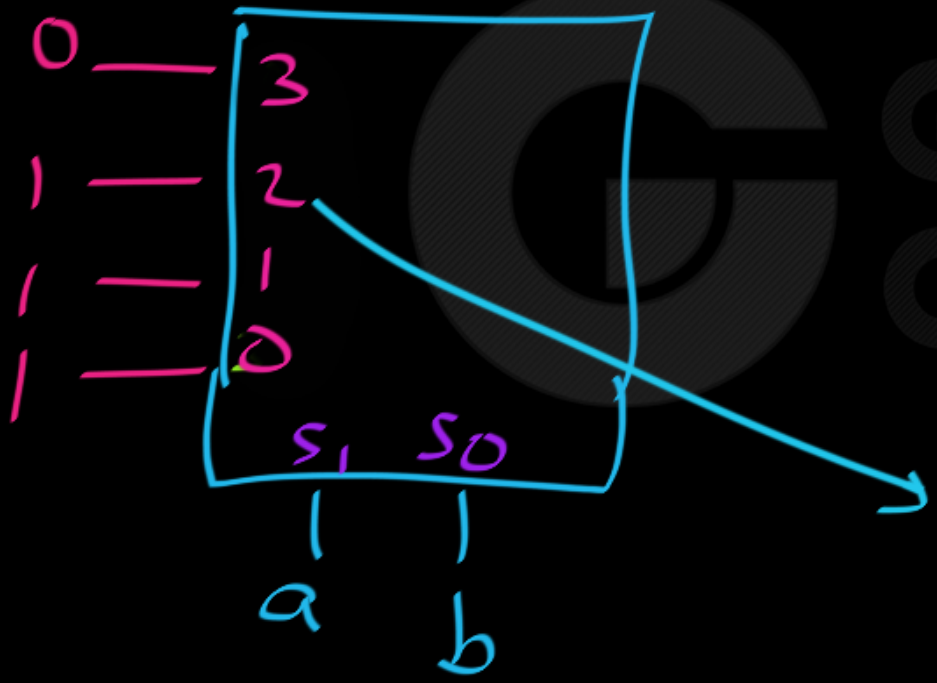
$f(a,b) = a + \overline{b}$

a	b	f(a,b)
0	0	1
0	1	0
1	0	1
1	1	1



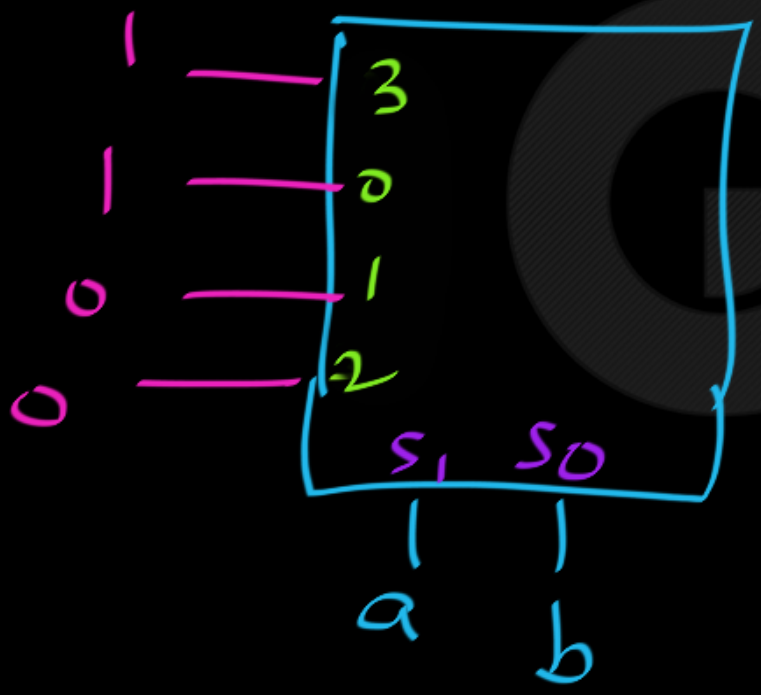
$f(a,b) = \overline{a\,b}$

	a	b	f(a,b)
0	0	0	1
1	0	1	1
2	1	0	1
3	1	1	0



means $a = 1, b = 0$

$f(a,b) = a \odot b$



	a	b	f(a,b)
0	0	0	1
1	0	1	0
2	1	0	0
3	1	1	1

$f(a,b,c)$ $\Rightarrow$ one 8-1 mux is Enough.
without Additional Hardware

3 Variables $\Rightarrow$ on 3 select lines

Truth Table $\Rightarrow$ on Data inputs.

$f(a_1, a_2, \cdots, a_n)$ $\Rightarrow$ one $2^h - 1$ mux is Enough.

without Additional Hardware

n Variables $\Rightarrow$ on h select lines

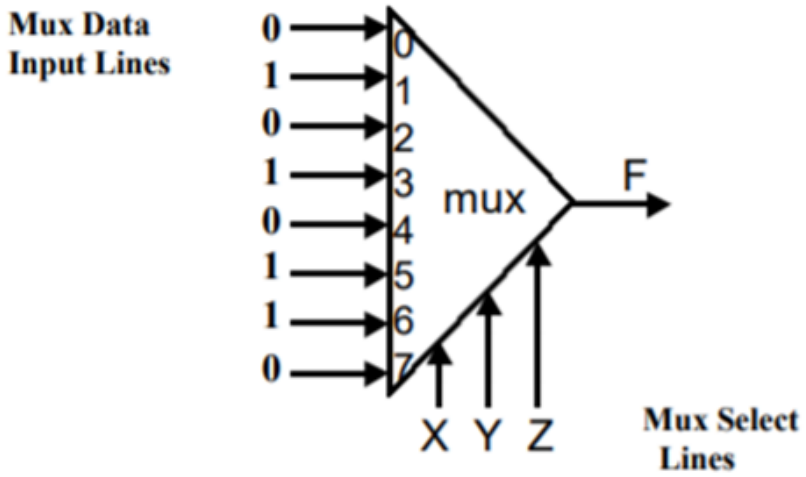
Truth Table $\Rightarrow$ on Data inputs.

Implementing n-variable Functions Using 2^n-to-1 Multiplexers

- Any n-variable logic function, in canonical sum-of-minterms form can be implemented using a single 2^n-to-1 multiplexer:
  - The n input variables are connected to the mux select lines.
  - For each mux data input line I_i ($0 \leq i \leq 2^n - 1$):
    - Connect 1 to mux input line I_i if i is a minterm of the function.
    - Otherwise, connect 0 to mux input line I_i (because i is not a minterm of the function thus the selected input should be 0).

Example: 3-variable Function Using 8-to-1 mux

- Implement the function $F(X,Y,Z) = \Sigma(1,3,5,6)$ using an 8-to-1 mux.
  - Connect the input variables X, Y, Z to mux select lines.
  - Mux data input lines 1, 3, 5, 6 that correspond to function minterms are connected to 1.
  - The remaining mux data input lines 0, 2, 4, 7 are connected to 0.



Next Topic:

Simple

↑

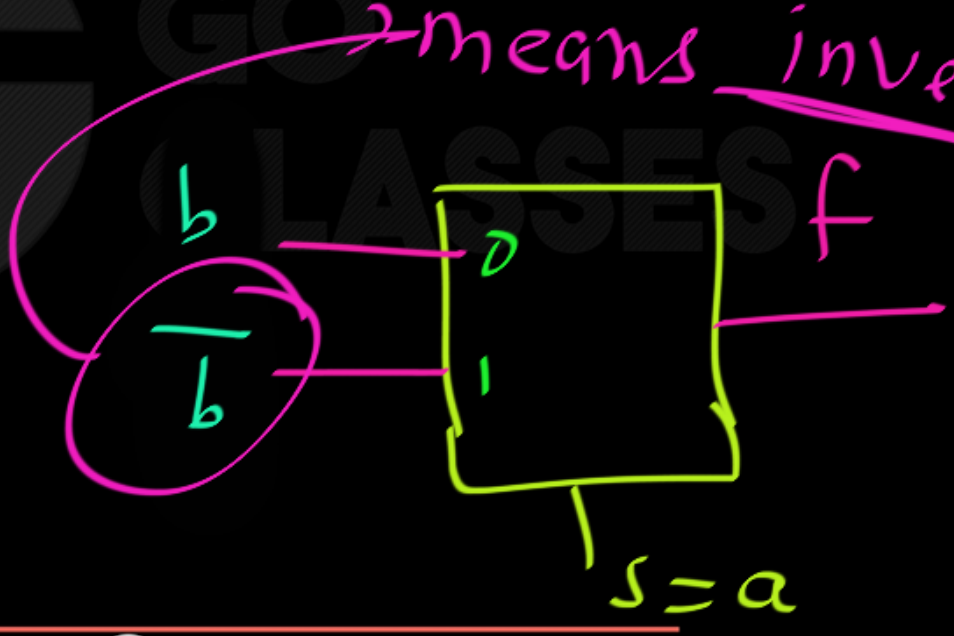
$2^{n-1} * 1$ Mux is NOT enough for every n-var function.

$f(a,b)$ $\Rightarrow$ Simple $2*1$ mux is NOT enough.

We need Additional Hardware.

$f(a,b) = a \oplus b$

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



$f(a,b) = a \oplus b$

Simple 2-1 mux NOT enough.

(2-1 mux) + (1 NOT gate)

$f(a,b)$ $\Rightarrow$ Simple $2*1$ mux is NOT enough.

We need Additional Hardware.

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

$\bar{a} + ab$

Q: No 2-var function can be implemented using single 2-1 mux without Additional HW ?

Q: No 2-var can be implemented using single 2-1 mux without Additional HW ? ⇒ false

Some 2-var functions can be implemented.

Eg: $f(a,b) = \overline{a} + b$

Q: Some 2-var function can be implemented using single 2-1 mux without Additional HW?

Q: Some 2-var can be implemented using single 2-1 mux without Additional HW ?

True

Q: Every 2-Var ⊥ function can be implemented using single 2-1 mux without Additional HW ?

Q: Every 2-var can be implemented using single 2-1 mux without Additional HW ?

false

Q:

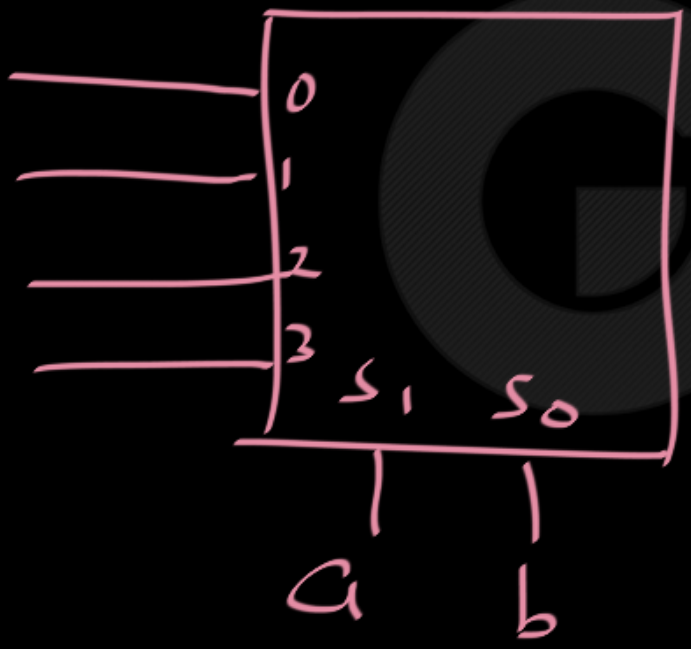
2-Var function can be implemented using single 2-1 mux without Additional HW ?

⇒ false

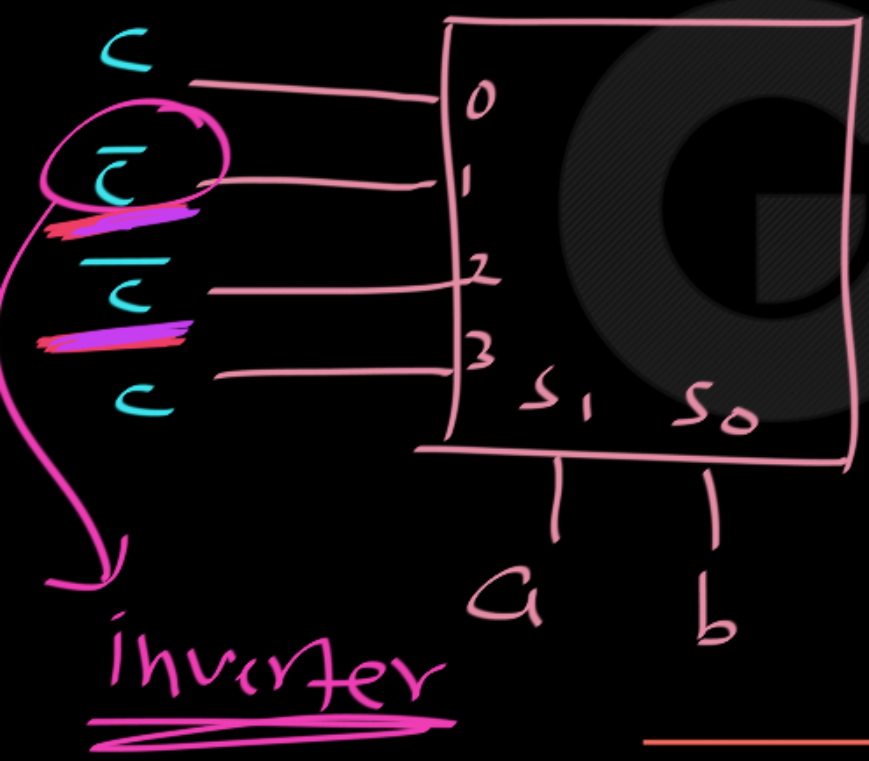
$f(a,b,c)$ $\Rightarrow$ one 4-1 mux Not enough.

$\Rightarrow$ (one 4-1 mux + one NOT Gate is Enough.)

$$f(a,b,c) = a \oplus b \oplus c$$

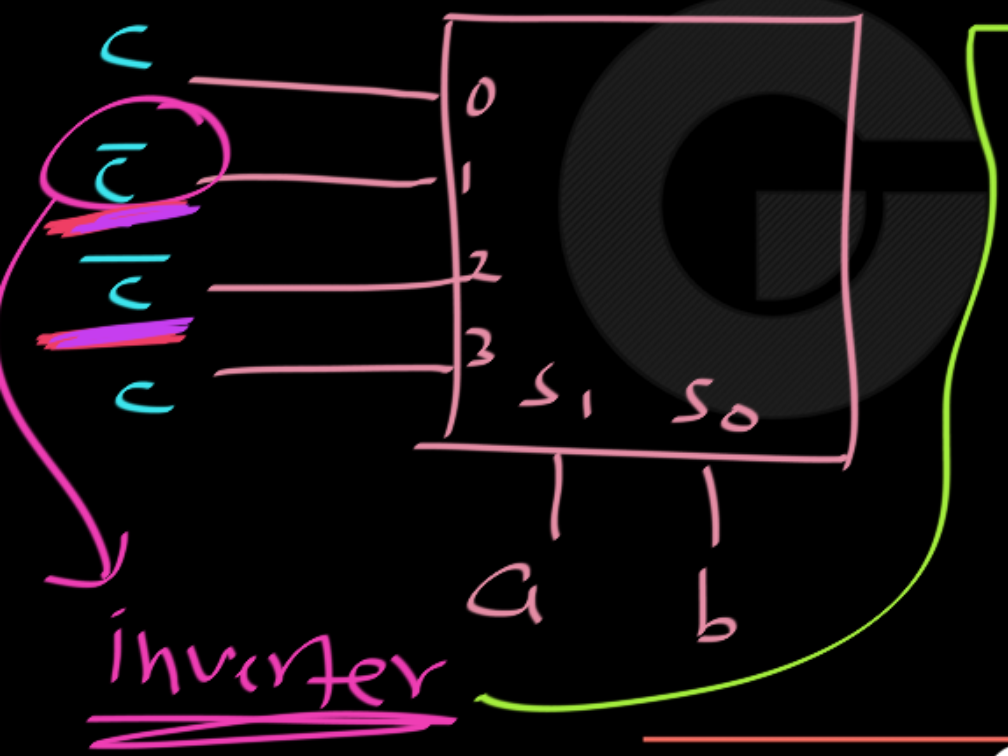


$f(a,b,c) = a \oplus b \oplus c$



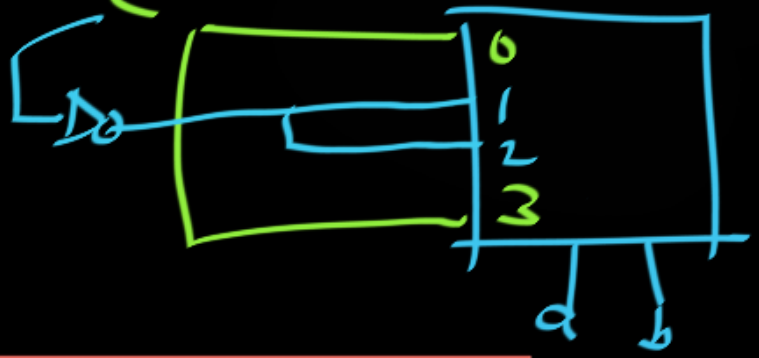
a	b	f(a,b,c)
0	0	c
0	1	c̄
1	0	c̄
1	1	c

$f(a,b,c) = a \oplus b \oplus c \Rightarrow$ 1 NOT gate + 1 4-1 mux



→ Only 1- inverter Needed

NOT 2 inverters.



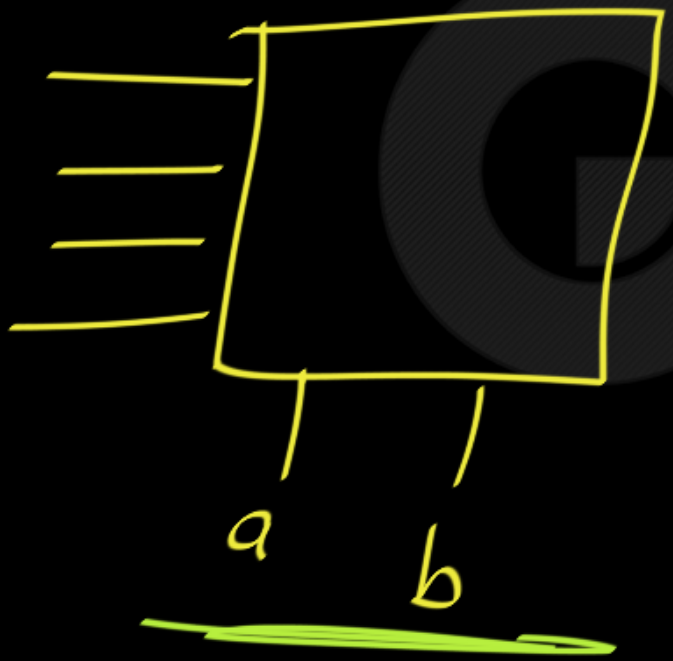
$0 \oplus 0 \oplus C = 0 \oplus C = C$

0

$0 \oplus 1 \oplus C = 1 \oplus C = \overline{C}$

1

$f(a,b,c) =$ 1 4x1 mux



a	b	f(a,b,c)
0	0	
0	1	
1	0	
1	1	

3-variable function :

1. One 8x1 mux
2. One 4x1 mux + one inverter
3. one 2x1 mux + Additional HW (OR, AND, NOT)

n -variable function :

1. One $2^n \times 1$ mux

2. One $2^{n-1} \times 1$ mux + one inverter

3. One $2^{n-2} \times 1$ mux + Additional HW (OR, AND, NOT)

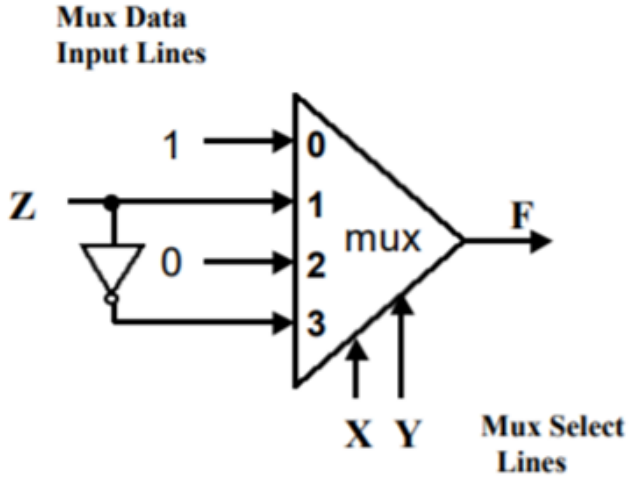
Implementing n-variable Functions Using 2^{n-1}-to-1 Multiplexers

- Any n-variable logic function can be implemented using a smaller 2^{n-1}-to-1 multiplexer and a single inverter (e.g 4-to-1 mux to implement 3 variable functions) as follows:
  - Express function in canonical sum-of-minterms form.
  - Choose n-1 variables as inputs to mux select lines.
  - Construct the truth table for the function, but grouping inputs by selection line values (i.e select lines as most significant inputs).
  - Determine multiplexer input line i values by comparing the remaining input variable and the function F for the corresponding selection lines value i:
    - Four possible mux input line i values:
      - Connect to 0 if the function is 0 for both values of remaining variable.
      - Connect to 1 if the function is 1 for both values of remaining variable.
      - Connect to remaining variable if function is equal to the remaining variable.
      - Connect to the inverted remaining variable if the function is equal to the remaining variable inverted.

Example: 3-variable Function Using 4-to-1 mux

- **Implement the function $F(X,Y,Z) = \Sigma(0,1,3,6)$ using a single 4-to-1 mux and an inverter.**
  - **We choose the two most significant inputs X, Y as mux select lines.**
  - **Construct truth table:**

Select Lines Value i	Select Lines X	Y	Z	F	Mux Input i
0	0	0	0	1	1
	0	0	1	1	
1	0	1	0	0	Z
	0	1	1	1	
2	1	0	0	0	0
	1	0	1	0	
3	1	1	0	1	Z'
	1	1	1	0	



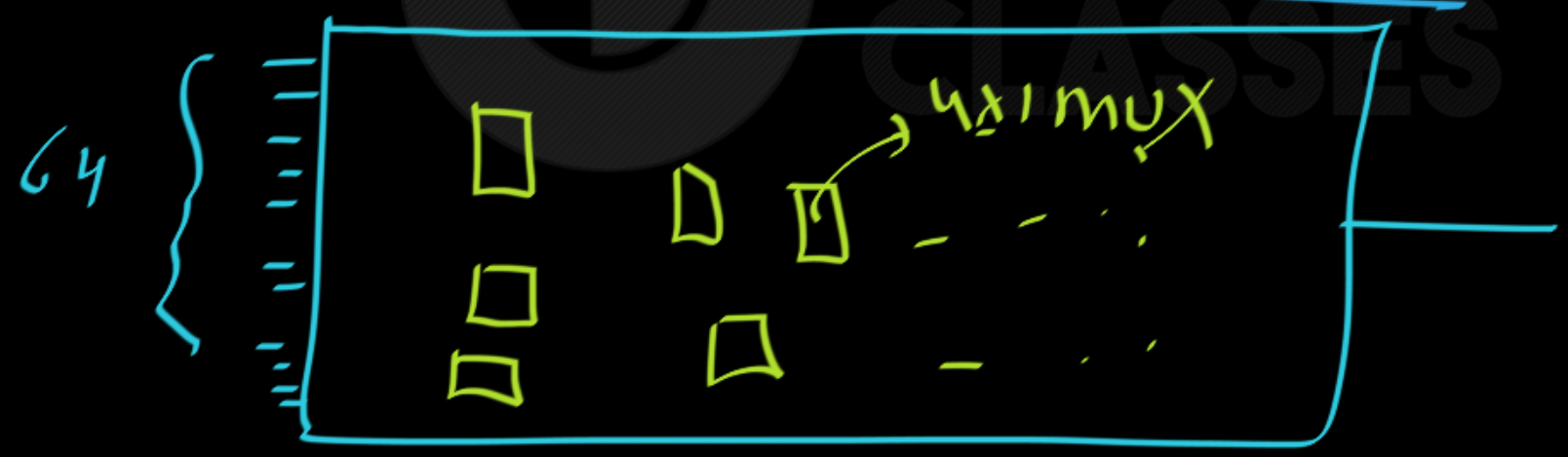
- **We Determine multiplexer input line i values by comparing the remaining input variable Z and the function F for the corresponding selection lines value i:**
  - **when XY=00 the function F=1 (for both Z=0, Z=1) thus mux input0 = 1**
  - **when XY=01 the function F=Z thus mux input1 = Z**
  - **when XY=10 the function F=0 (for both Z=0, Z=1) thus mux input2 = 0**
  - **when XY=11 the function F=Z' thus mux input3 = Z'**

Next Topic:

Building Larger Mux From Smaller Mux

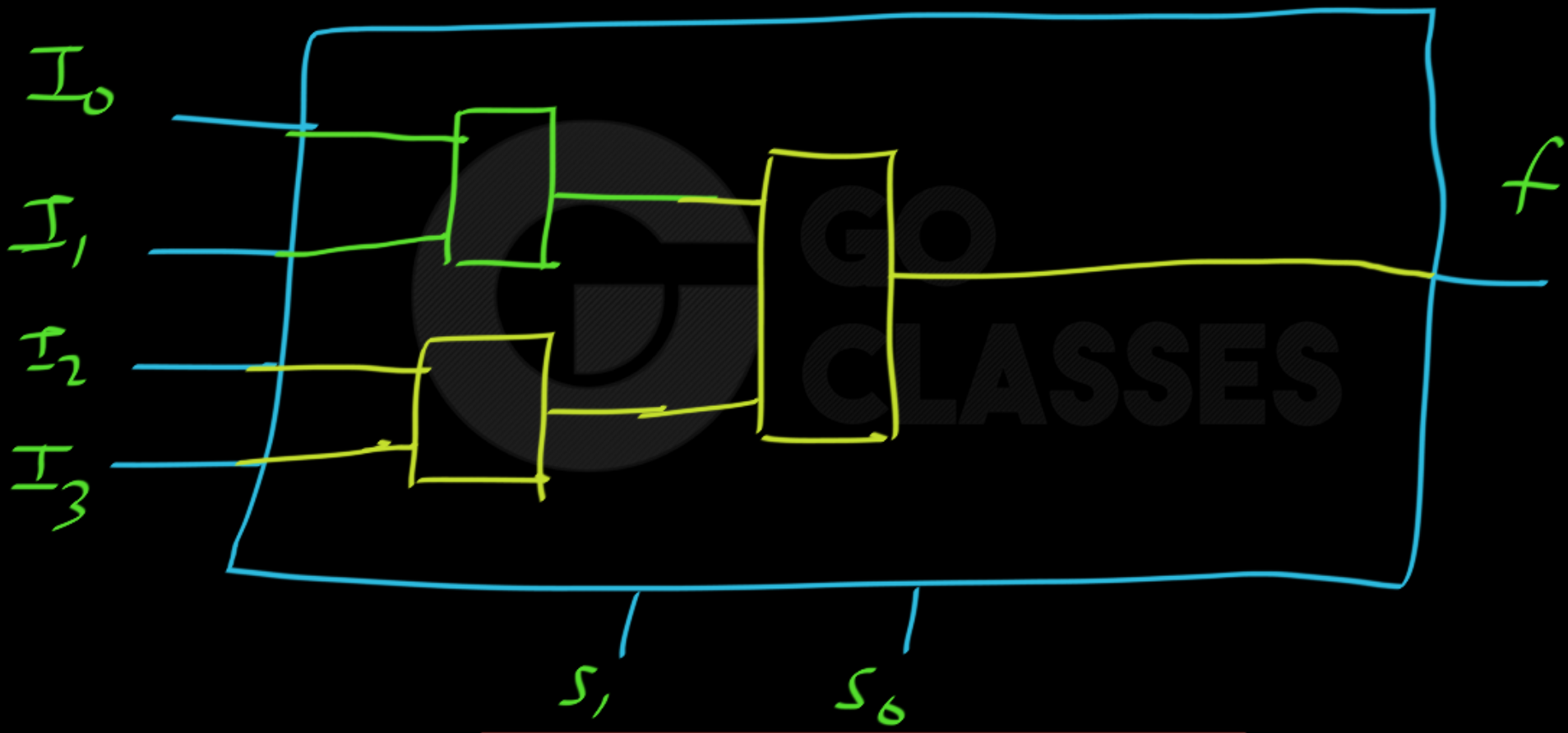
4 x 1 mux ← Manufactures Company

Your Requirement: 64 x 1 mux

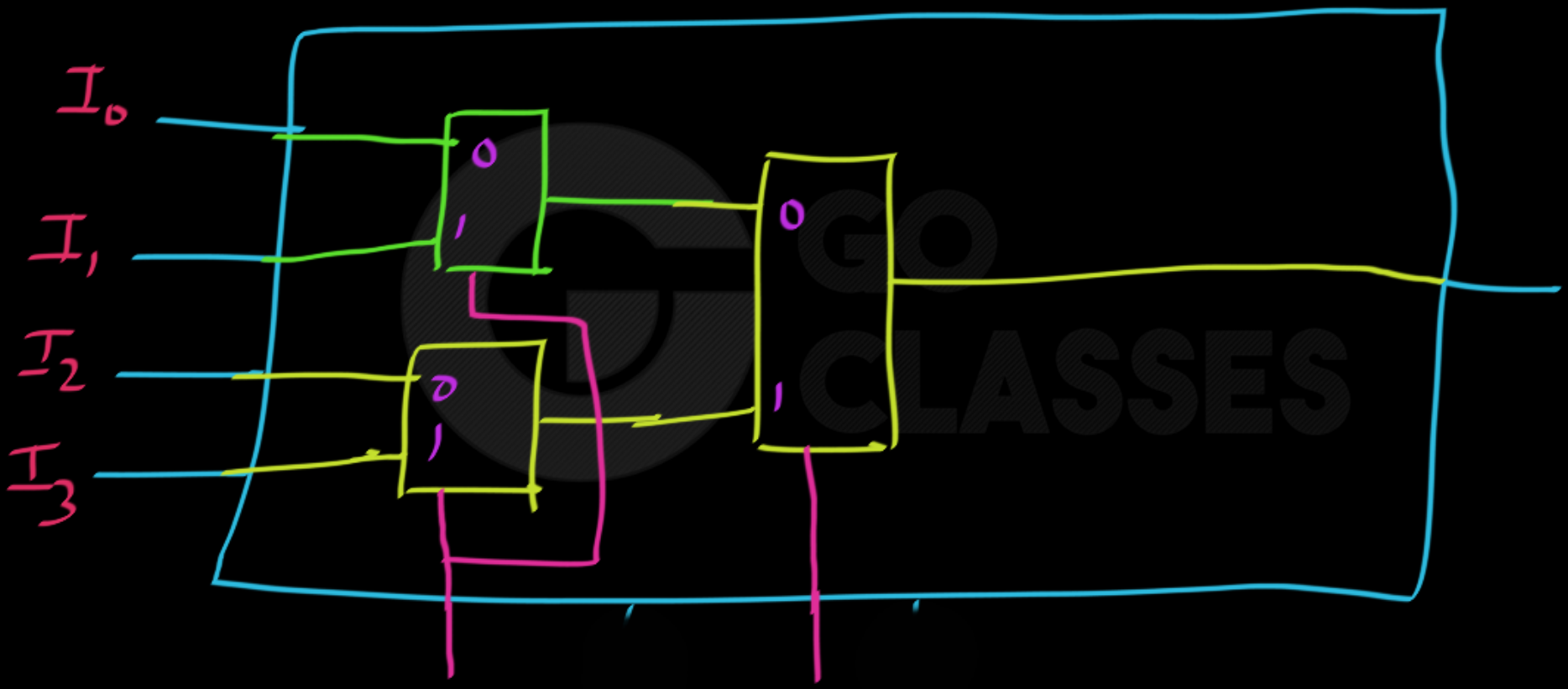


4x1 mux using 2x1 muxes:

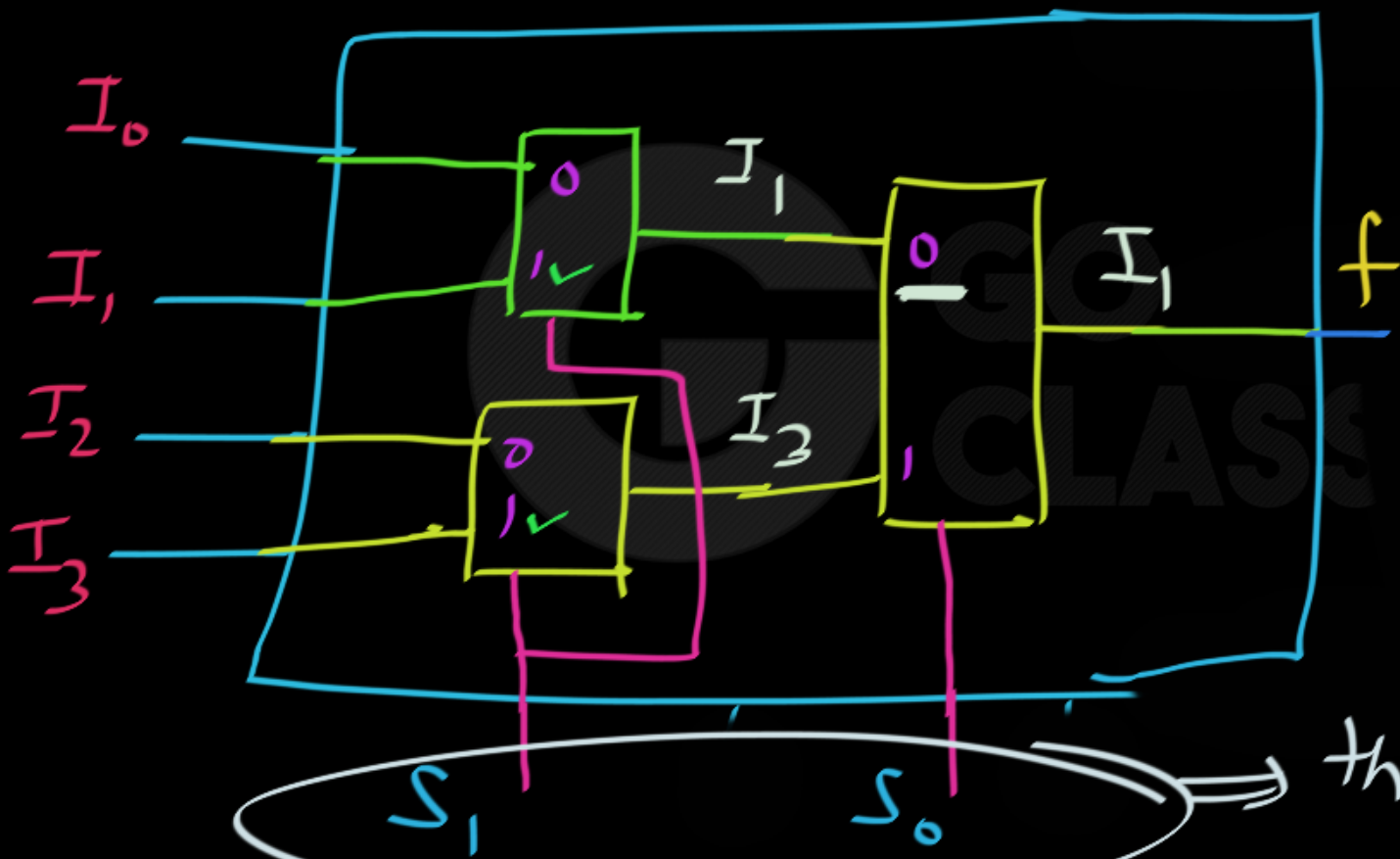
4x1 mux using 2x1 muxes:



4x1 mux using 2x1 muxes:



4x1 mux using 2x1 muxes:



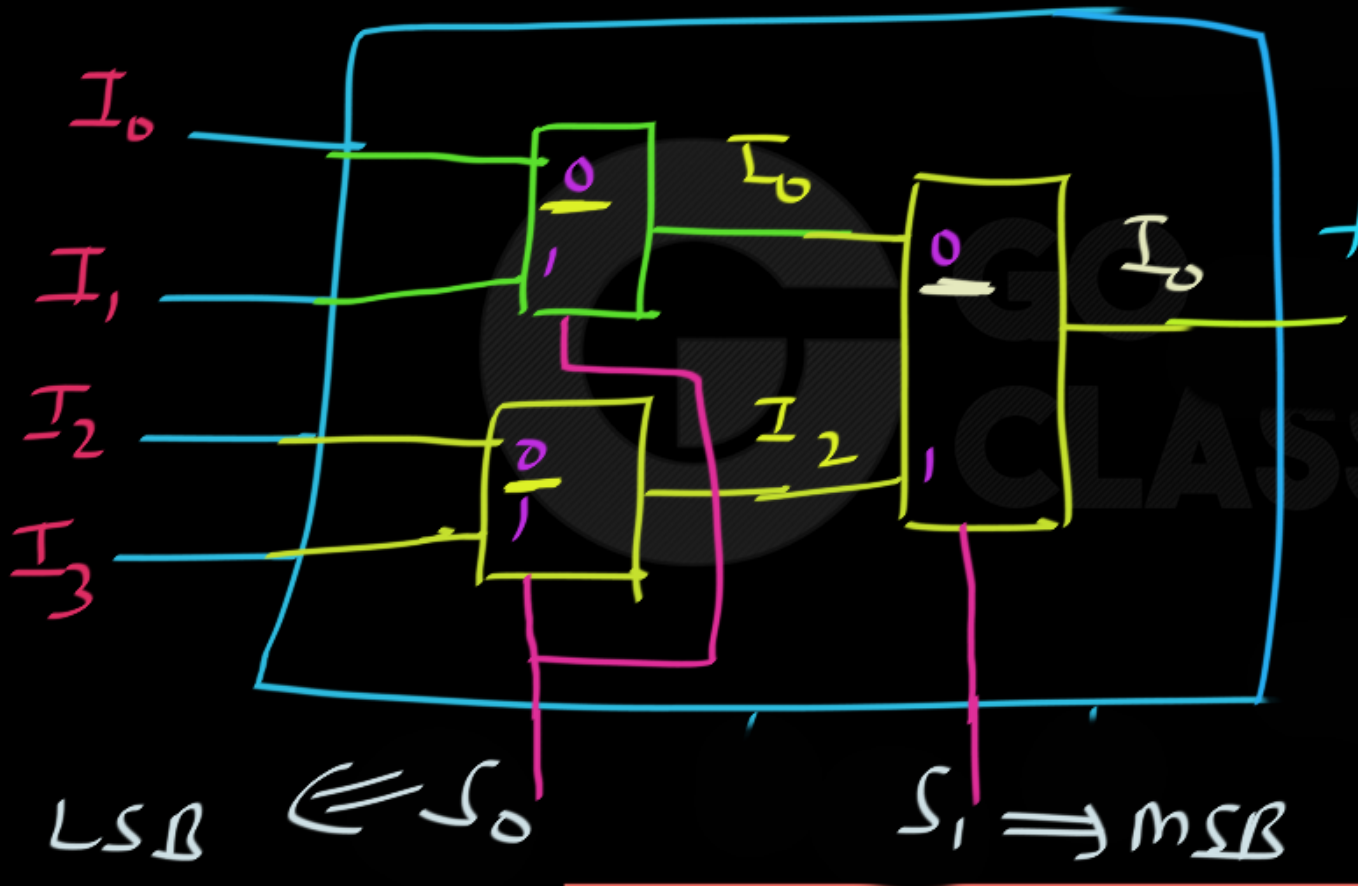
$S_1 = 1, S_0 = 0$

Desired $f = I_2$

Actual $f = I_1$

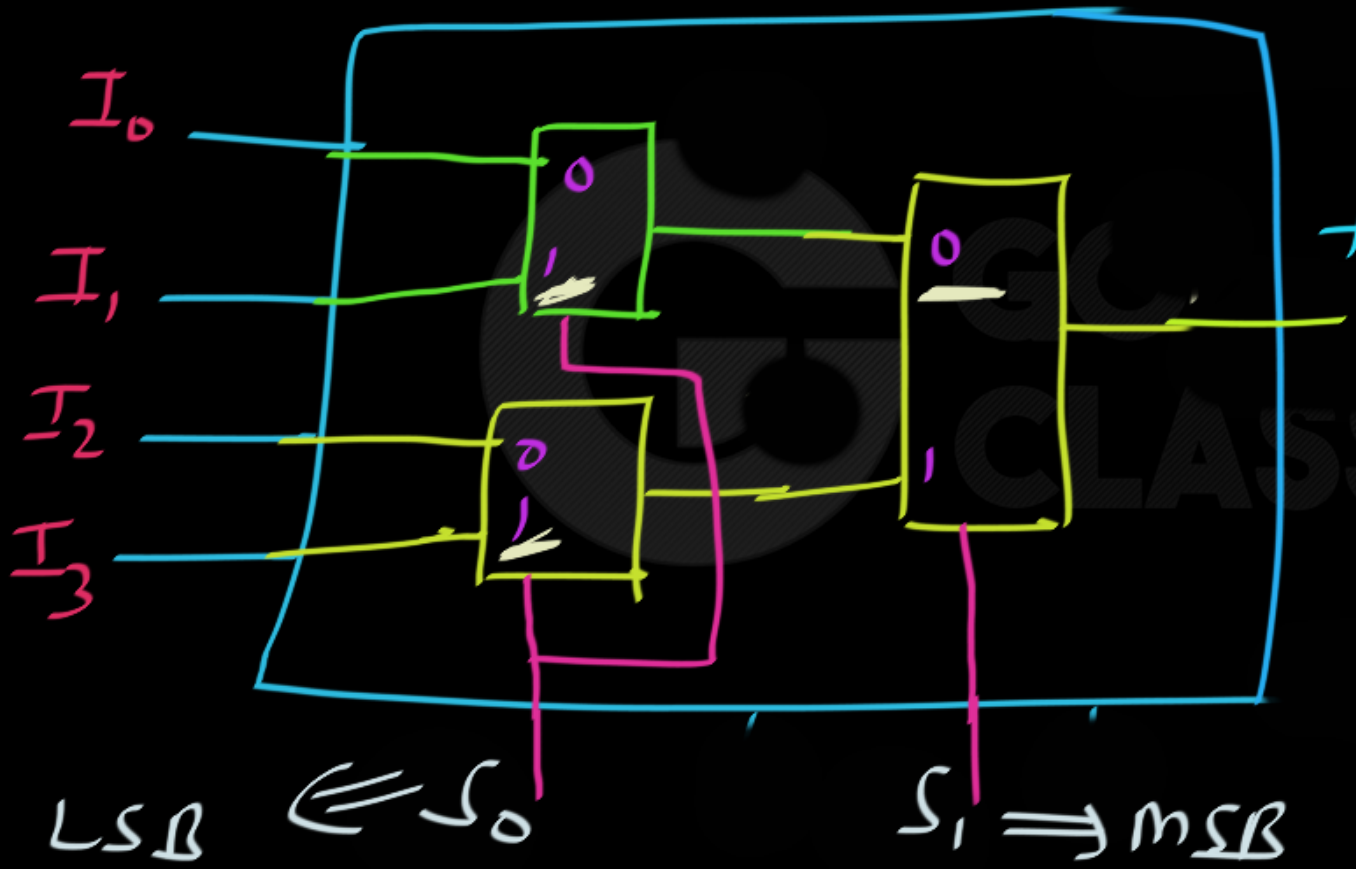
this setting is Not Correct.

4 x 1 mux using 2 x 1 muxes :



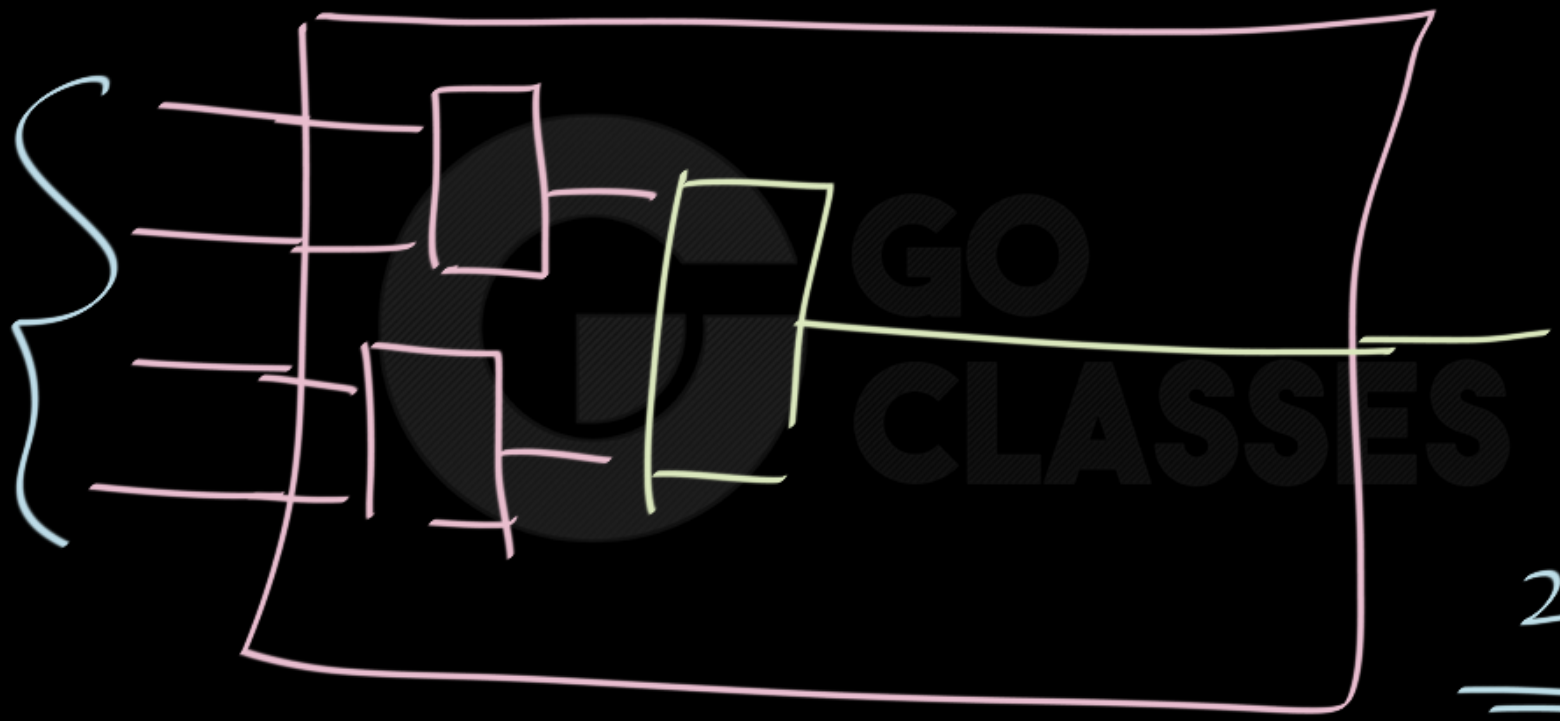
S_1	S_0	Desired	Actual
0	0	I_0	I_0

4 x 1 mux using 2 x 1 muxes :



S_1	S_0	Desired	Actual
0	0	I_0	I_0
0	1	I_1	I_1
1	0	I_2	I_2
1	1	I_3	I_3

4x1 mux using 2x1 mux :

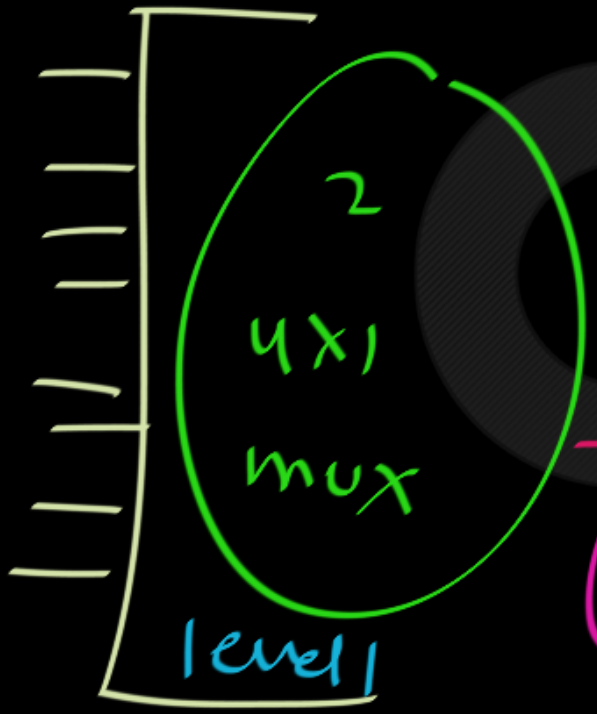


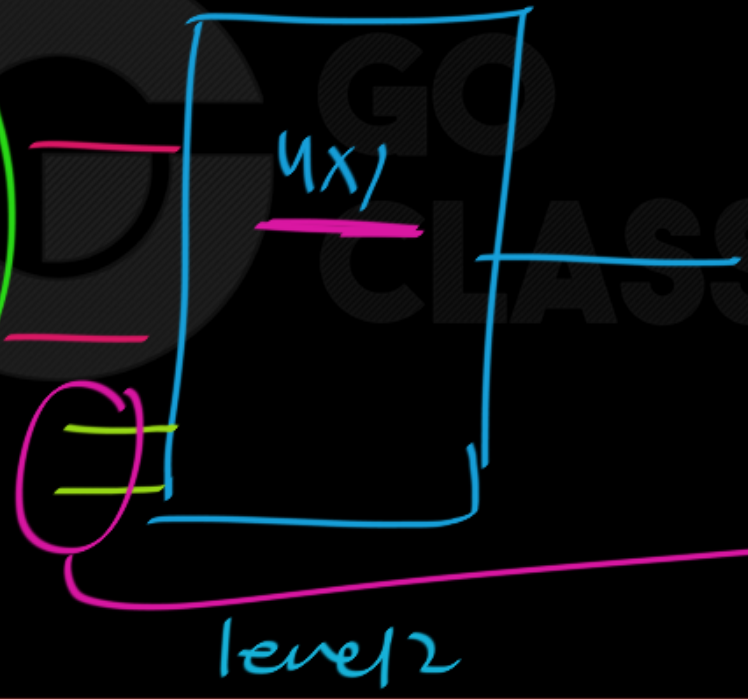
3 ↳ 2-1 mux ↳

8x1 mux using 2x1 mux:

$\#2\times1 \text{ mux} = 4+2+1 = 7$

8x1 mux using 4x1 mux :





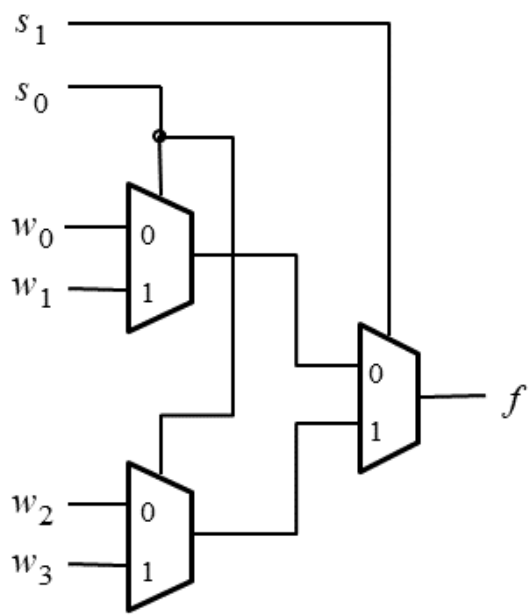
1 4x1 mux

4 input lines

#4x1 mux = 3

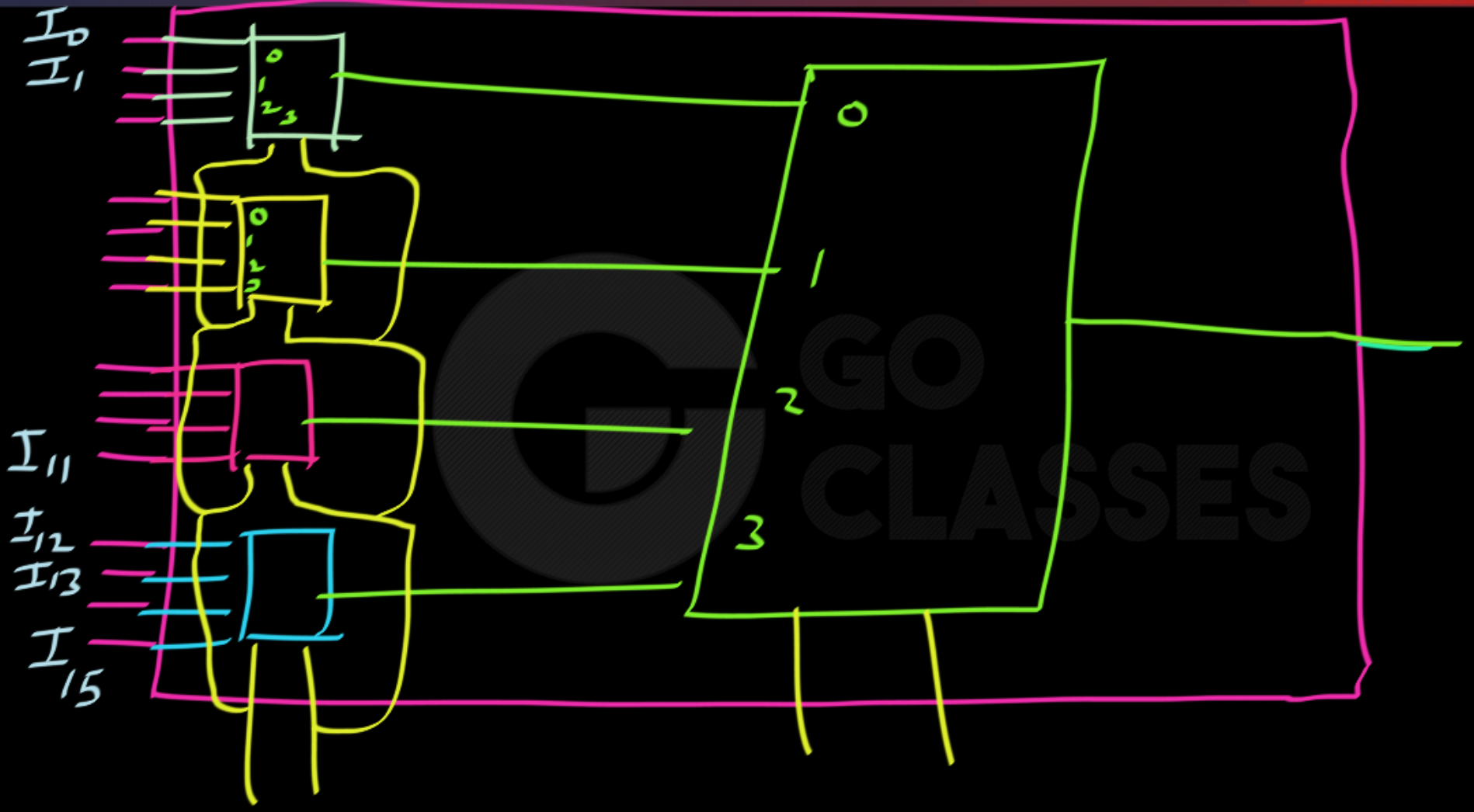
unused

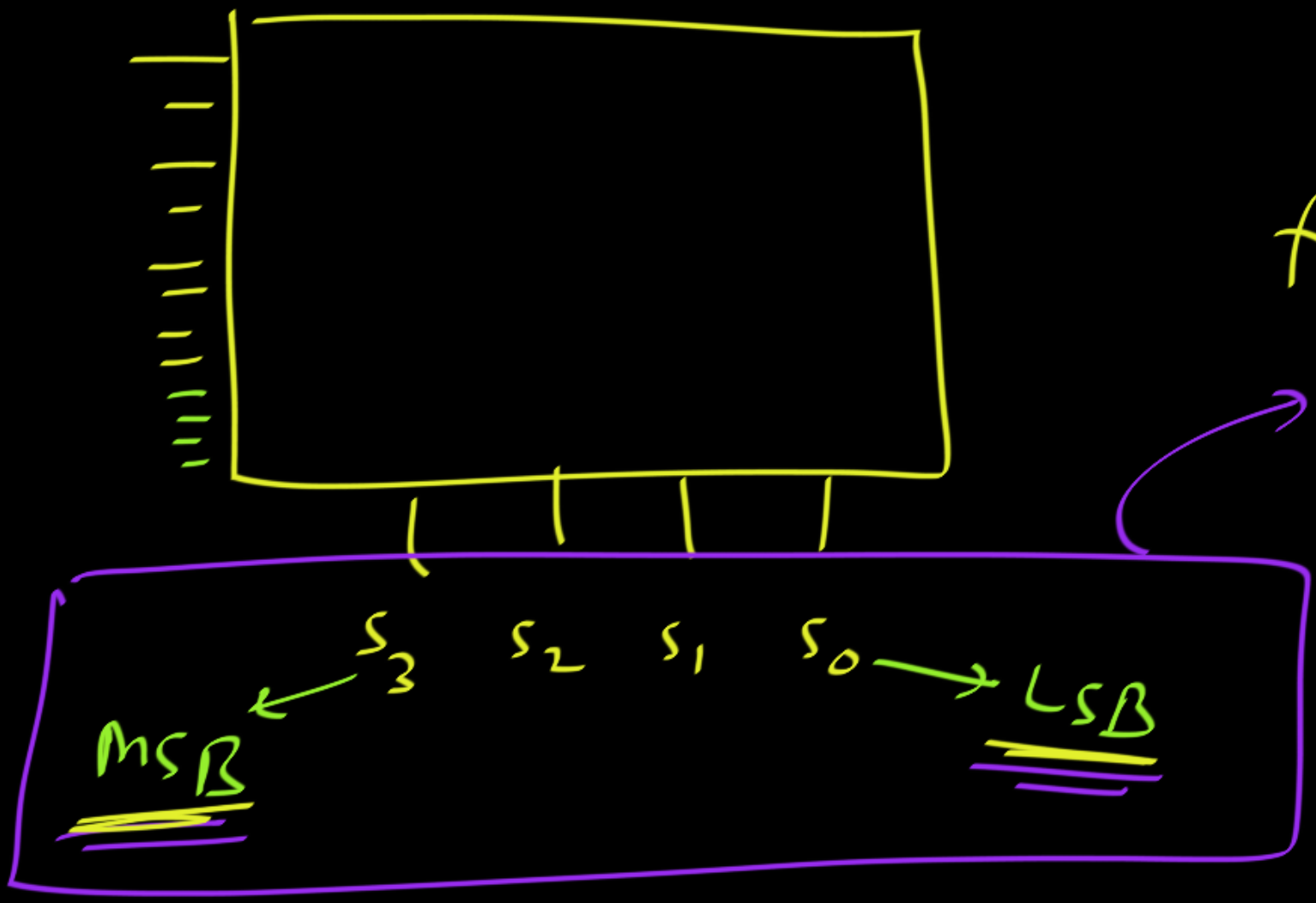
Using three 2-to-1 multiplexers to build one 4-to-1 multiplexer

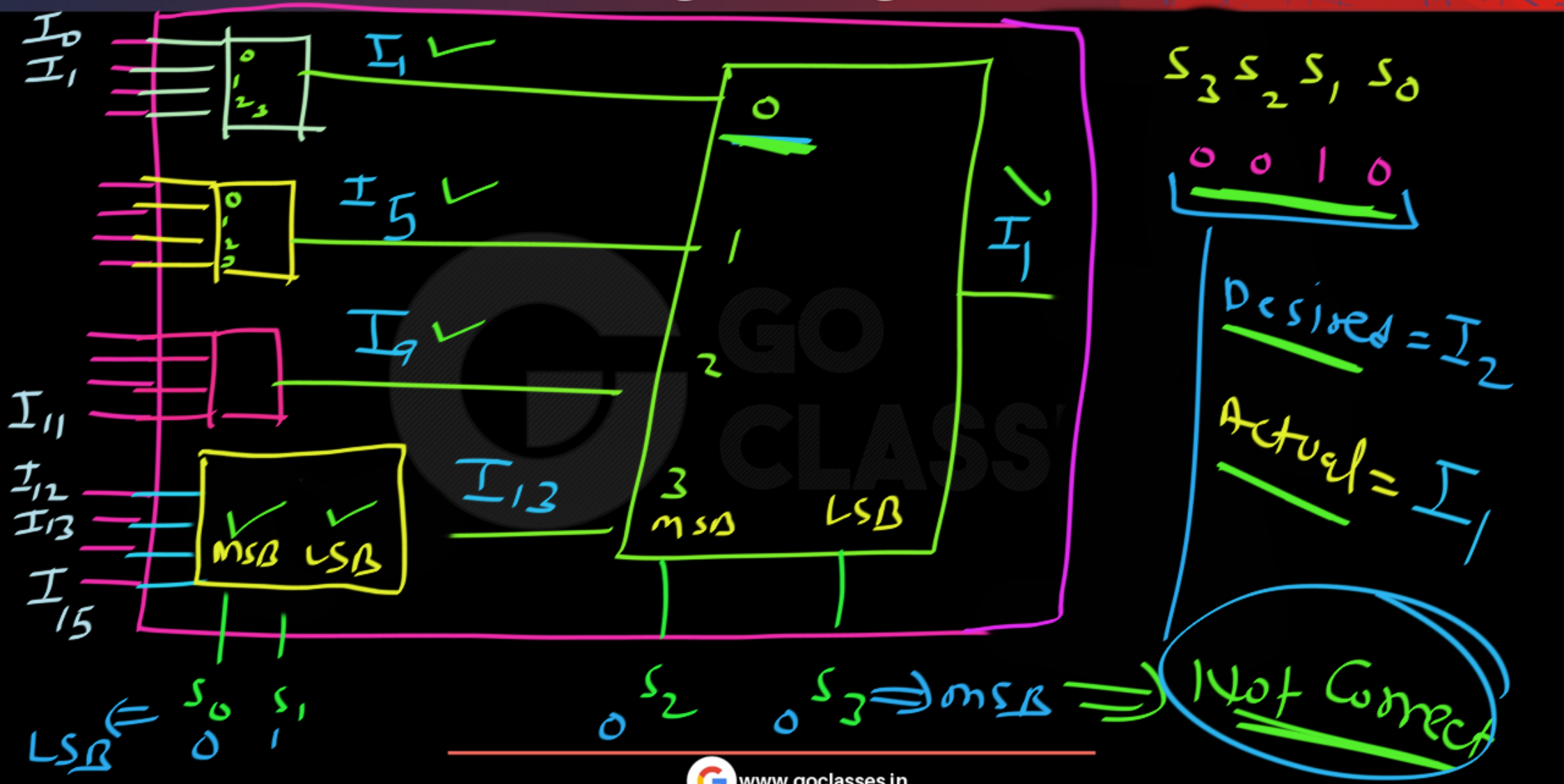


How to Handle Select lines ?

16 x 1 mux using 4x1 mux

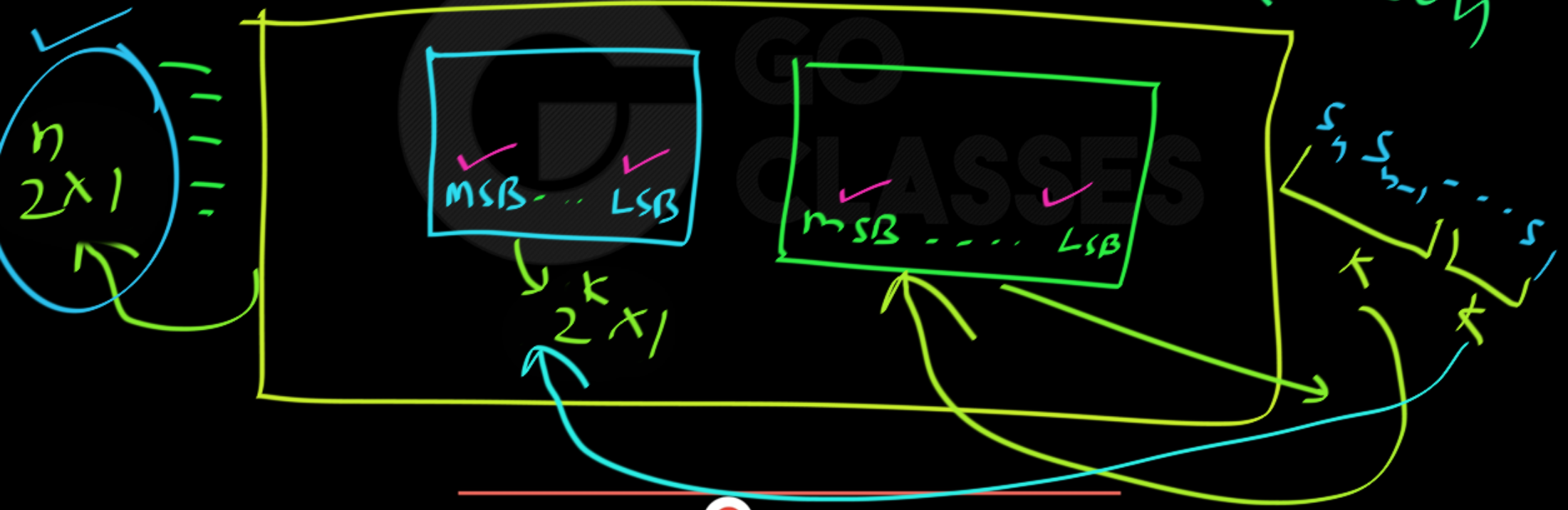

I_0
I_1
I_{11}
I_{12}
I_{13}
I_{15}
0
1
2
3


from 16x1 mux point of view.
S_3
S_2
S_1
S_0
MSB
LSB


I_0
I_1
I_5
I_9
I_{11}
I_{12}
I_{13}
I_{15}
I_{13}
MSB
LSB
I_1
I_5
I_9
I_{13}
I_1
MSB
LSB
S_0
S_1
S_2
S_3
LSB
MSB
S_3 S_2 S_1 S_0
0 0 1 0
Desired = I_2
Actual = I_1
Not Correct


I_0
I_1
I_{11}
I_{12}
I_{13}
I_{15}
0
1
2
3
MSB
LSB
S_1
S_0
S_3
S_2
S_3 S_2 S_1 S_0
from 16x1 Point of view,

By Default ; ⇒ If everything is in standard order that we have seen



By Default ; ⇒ If everything is in standard order that we have seen

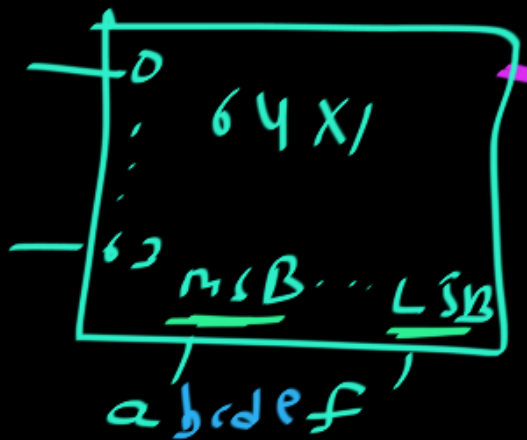
Ex: 64 X 1 mux using 4 x 1 mux

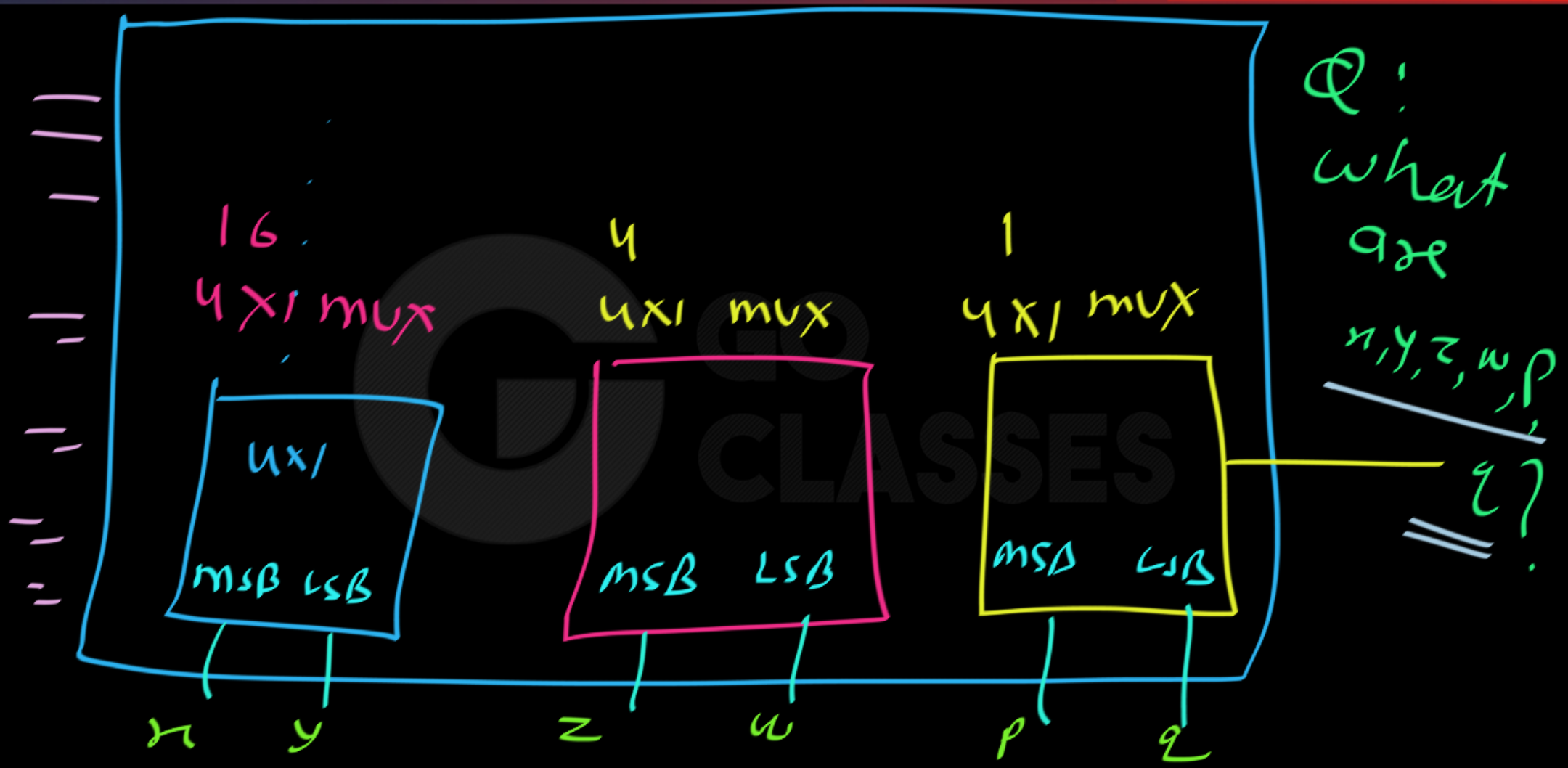
||

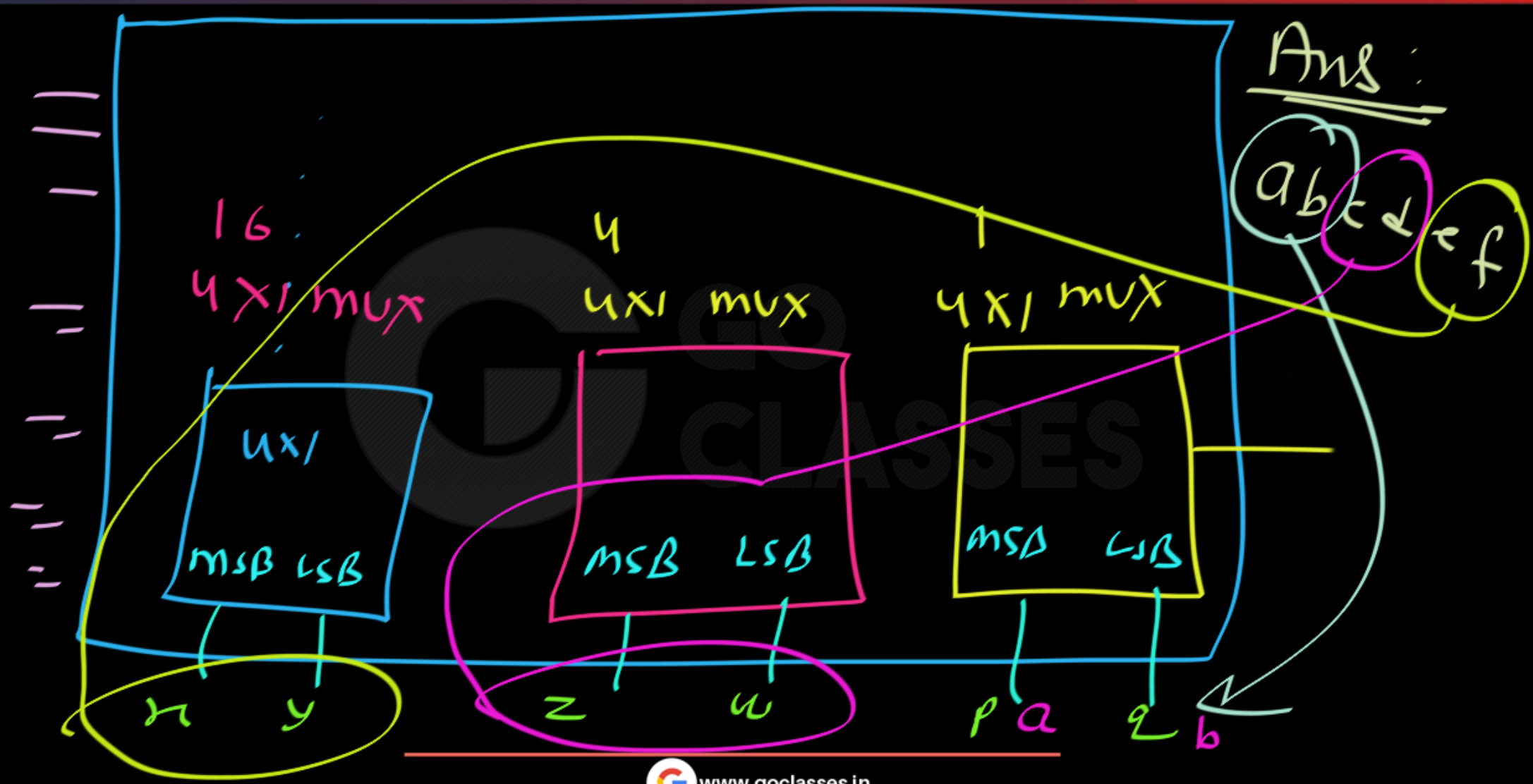
6 select lines : a b c d e f

MSB

LSB of 64 X 1 mux




16
4x1 mux
4
4x1 mux
1
4x1 mux
4x1
MSB LSB
MSB LSB
MSB LSB
x y z w p q
Q: what are x,y,z,w,p,q ??


16
4x1 mux
4
4x1 mux
1
4x1 mux
4x1
MSB LSB
MSB LSB
MSB LSB
x y
z w
p q
Ans:
a b c d e f
a
b

Ans: $p = a$ $\quad r = e$

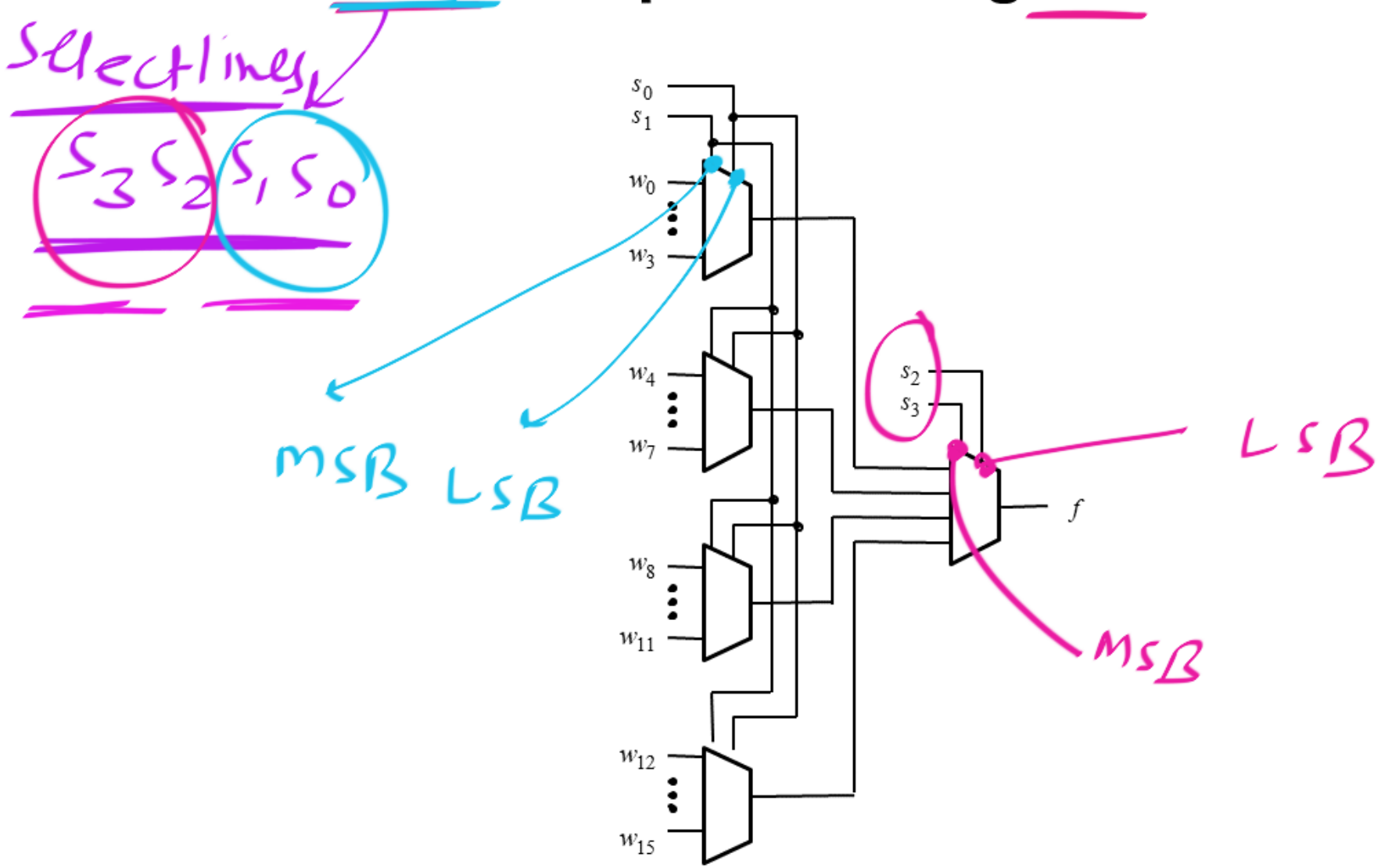
$q = b$ $\quad y = f$

$z = c$

$w = d$

$x, y, z, w, p, q \equiv e\ f\ c\ d\ a\ b$

16-1 Multiplexer using 4-1 Mux



16-1 Multiplexer using 4-1 Mux