



# **MULTIPLEXER / MUX / DATA SELECTOR**

# MULTIPLEXERS

- A multiplexer is a combinational circuit that selects one of many input lines ( $2^n$ ) and directs it to its single output line.
- There are  $n$  selection lines whose bit combinations determine which input is selected.
- A multiplexer is also called a data selector, since it selects one of many inputs and steers the binary information to the output line.
- The size of a multiplexer is specified by the number  $2^n$  of its input lines and the single output line. It is then implied that it also contains  $n$  selection lines.
- Example: 2X1 MUX, 4X1 MUX, 8X1 MUX, 16X1 MUX etc.



# MULTIPLEXERS

## 4-to-1 MUX

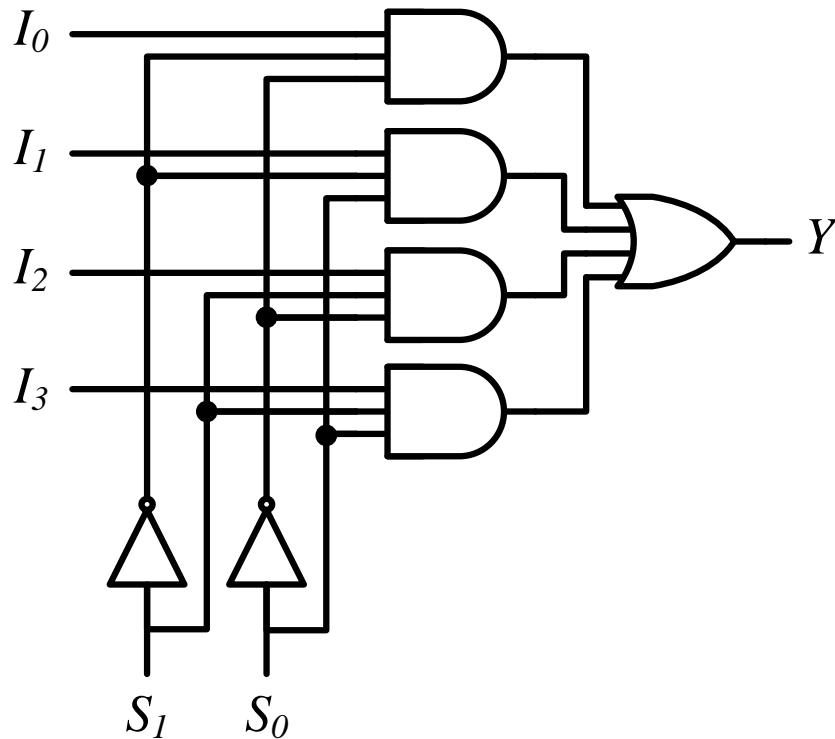


Fig: Logic diagram of a 4X1 MUX

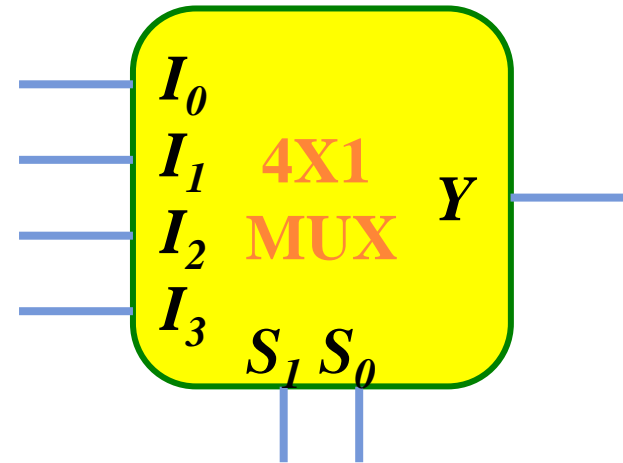


Fig: Block diagram of a 4X1 MUX

Function Table of a 4X1 MUX:

| $S_1$ | $S_0$ | $Y$   |
|-------|-------|-------|
| 0     | 0     | $I_0$ |
| 0     | 1     | $I_1$ |
| 1     | 0     | $I_2$ |
| 1     | 1     | $I_3$ |

Output Equation:

$$Y = S_1'S_0'I_0 + S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3$$

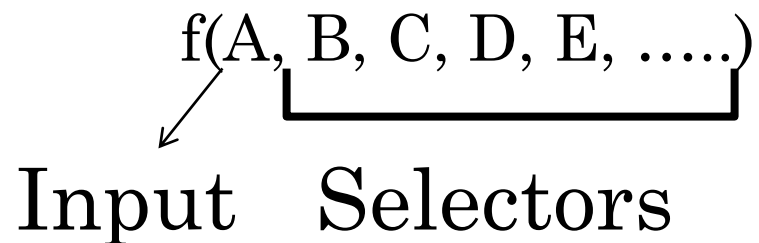
# USES OF MUX

- It is used for connecting two or more sources to a single destination among computer units.
- It is useful for constructing a common bus system etc.



# FUNCTION IMPLEMENTATION USING MUX

- $(n+1)$  variable function can be implemented with  $2^n \times 1$  MUX
- Simplify the function in sum of minterms form
- Among  $(n+1)$  variables,  $n$  variables are used as selector and one variable is connected with input lines



# Procedure 1



# IMPLEMENTATION USING MULTIPLEXERS:

## PROCEDURE 1

$$F(A, B, C) = \sum(1, 3, 5, 6)$$

Steps:

1. Choose the selector variables.  
Let's choose,
  - B, C as selector  $S_1$  and  $S_0$
  - A as input line
2. In the first row, list the name of the input lines of the multiplexers horizontally
3. In the second row, list the minterms where A is complemented
4. In the third row, list the minterms where A is uncomplemented




# IMPLEMENTATION USING MULTIPLEXERS:

## PROCEDURE 1

$$F(A, B, C) = \sum(1, 3, 5, 6)$$

Steps:

5. Circle the minterms for which the function outputs 1
  6. Fourth row presents the multiplexer inputs
    - If the two minterms in a column are not circled, apply 0 to the corresponding multiplexer input
    - If the two minterms in a column are circled, apply 1 to the corresponding multiplexer input
    - If the bottom minterm is circled and the top is not circled, apply A to the corresponding multiplexer input
    - If the top minterm is circled and the bottom is not circled, apply A' to the corresponding multiplexer input
- 



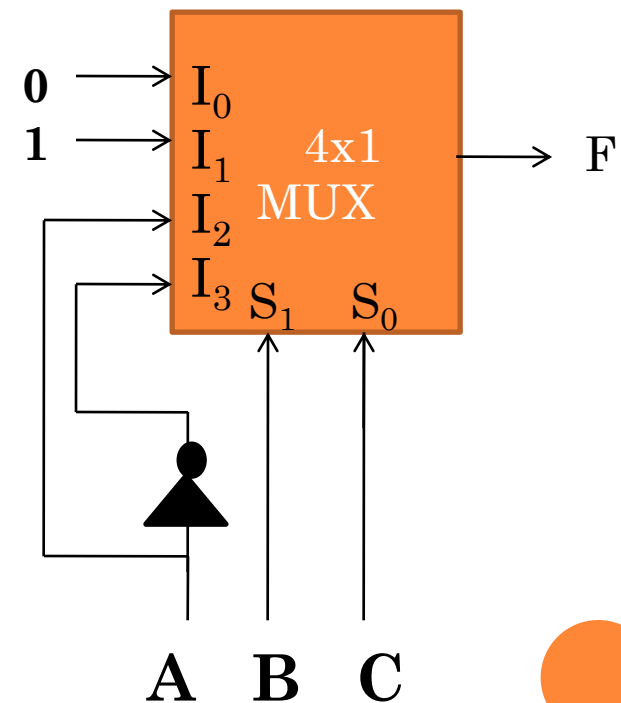
# IMPLEMENTATION USING MULTIPLEXERS:

## PROCEDURE 1

○  $F(A, B, C) = \sum(1, 3, 5, 6)$

### Implementation Table:

| MUX input line | $I_0$ | $I_1$ | $I_2$ | $I_3$ |
|----------------|-------|-------|-------|-------|
| $A'$           | 0     | ①     | 2     | ③     |
| $A$            | 4     | ⑤     | ⑥     | 7     |
| Input values   | 0     | 1     | $A$   | $A'$  |



# IMPLEMENTATION USING MULTIPLEXERS:

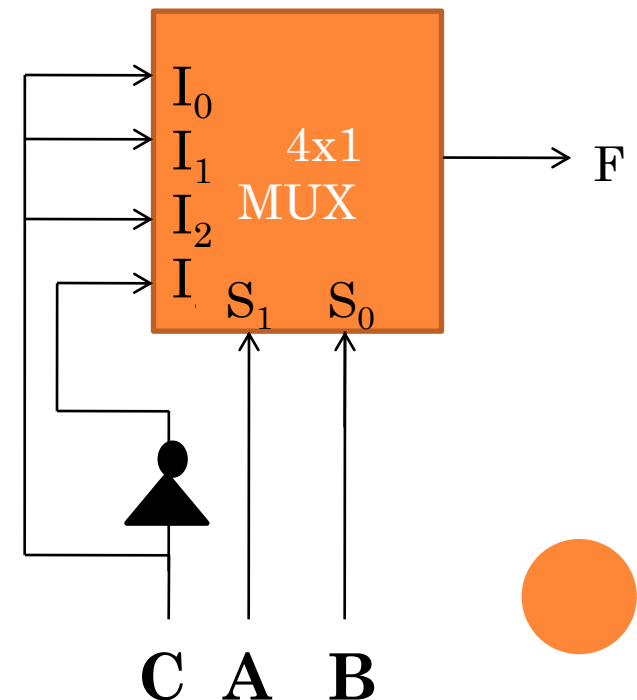
## PROCEDURE 1

- $F(A, B, C) = \sum(1, 3, 5, 6)$

What if A, B are the selectors and C goes to input line?

**Implementation Table:**

| MUX input line | I <sub>0</sub> | I <sub>1</sub> | I <sub>2</sub> | I <sub>3</sub> |
|----------------|----------------|----------------|----------------|----------------|
| C'             | 0              | 2              | 4              | ⑥              |
| C              | ①              | ③              | ⑤              | 7              |
| Input values   | C              | C              | C              | C'             |



# Procedure 2



# IMPLEMENTATION USING MULTIPLEXERS:

## PROCEDURE 2

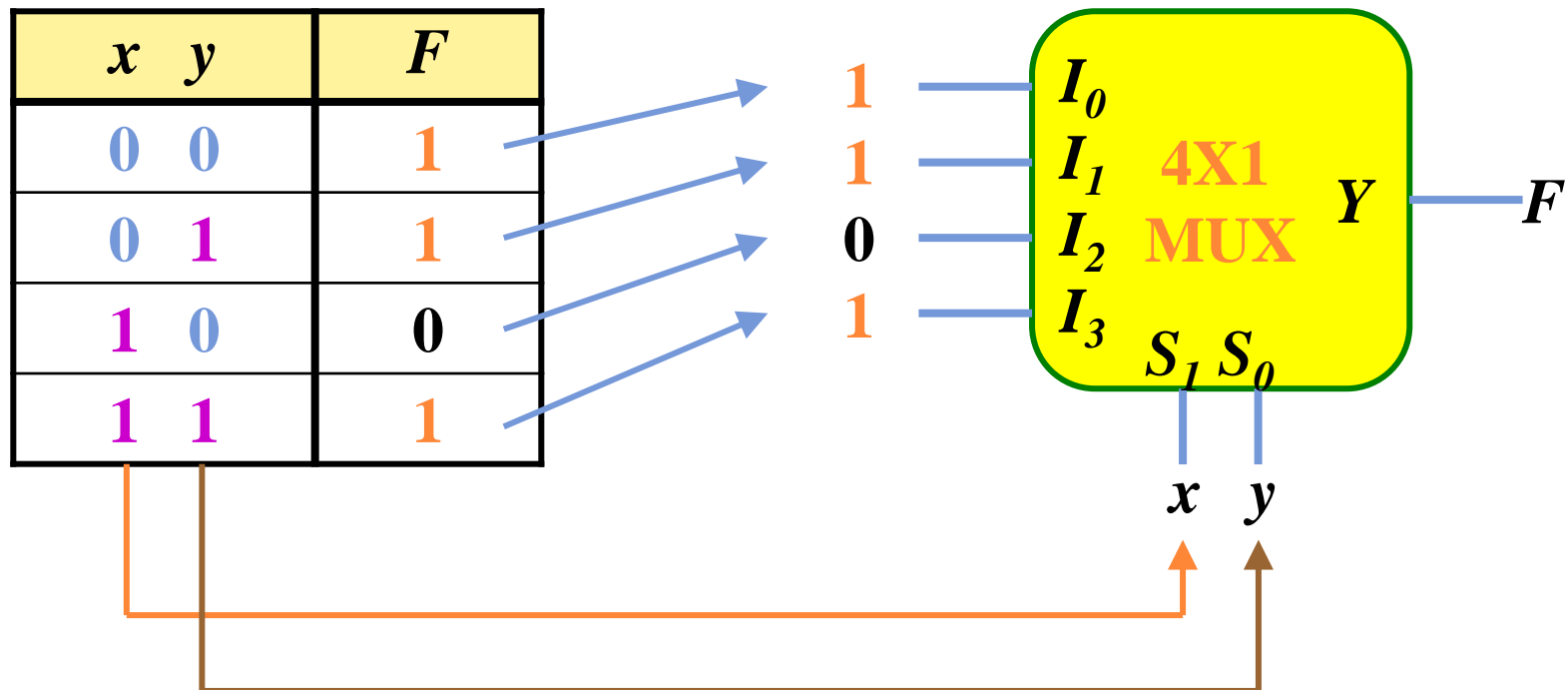
- Steps:

1. Complete the truth table from the SOP.
2. The first  $n - 1$  variables in the table are applied to the selection inputs of the multiplexer.
3. For each combination of the selection variables, we evaluate the output as a function of the last variable.
4. Apply these values to the data input in proper order.



# IMPLEMENTATION USING MULTIPLEXERS: PROCEDURE 2

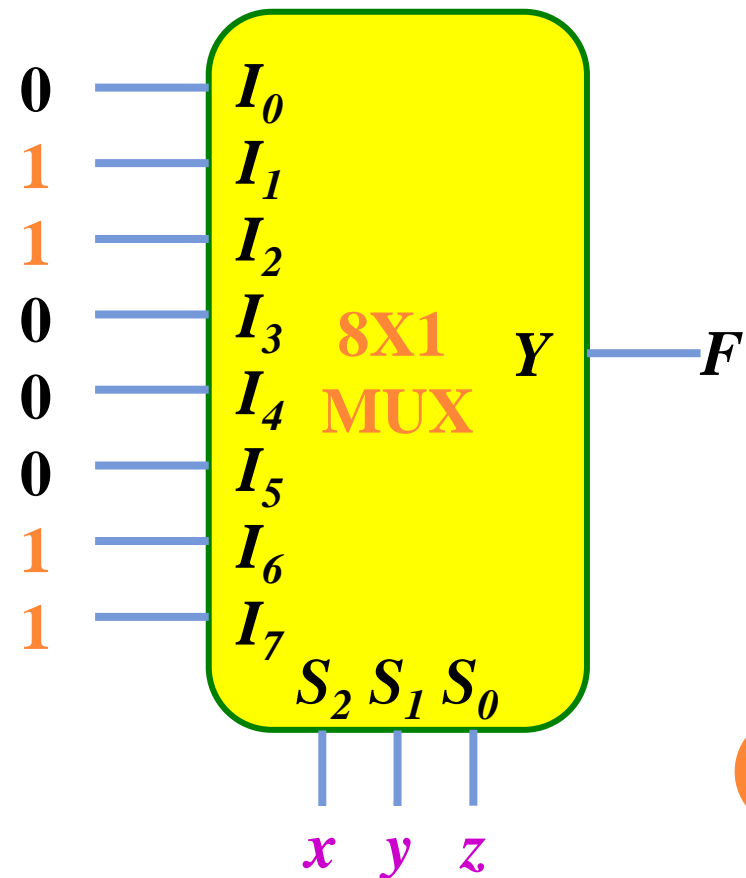
- Example  $F(x, y) = \sum(0, 1, 3)$



# IMPLEMENTATION USING MULTIPLEXERS: PROCEDURE 2

- Example  $F(x, y, z) = \sum(1, 2, 6, 7)$

| $x$ | $y$ | $z$ | $F$ |
|-----|-----|-----|-----|
| 0   | 0   | 0   | 0   |
| 0   | 0   | 1   | 1   |
| 0   | 1   | 0   | 1   |
| 0   | 1   | 1   | 0   |
| 1   | 0   | 0   | 0   |
| 1   | 0   | 1   | 0   |
| 1   | 1   | 0   | 1   |
| 1   | 1   | 1   | 1   |



# IMPLEMENTATION USING MULTIPLEXERS: PROCEDURE 2

○  $F(x, y, z) = \sum(1, 2, 6, 7)$

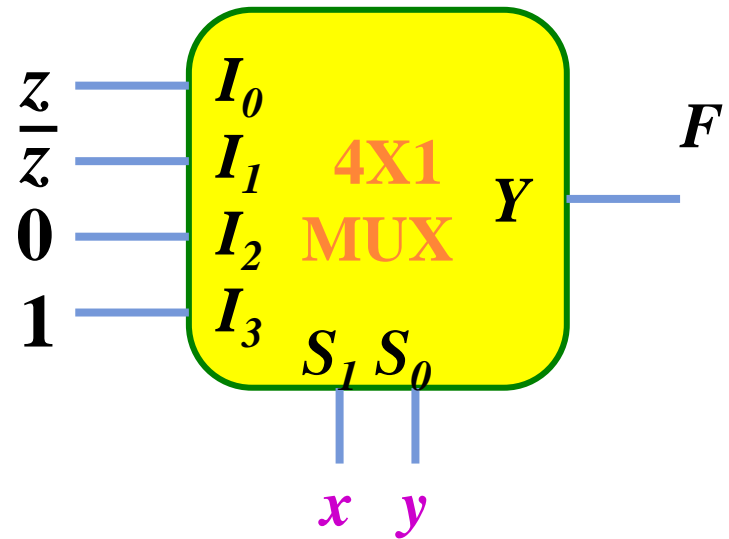
| $x$ | $y$ | $z$ | $F$ |
|-----|-----|-----|-----|
| 0   | 0   | 0   | 0   |
| 0   | 0   | 1   | 1   |
| 0   | 1   | 0   | 1   |
| 0   | 1   | 1   | 0   |
| 1   | 0   | 0   | 0   |
| 1   | 0   | 1   | 0   |
| 1   | 1   | 0   | 1   |
| 1   | 1   | 1   | 1   |

$F = z$

$F = \bar{z}$

$F = 0$

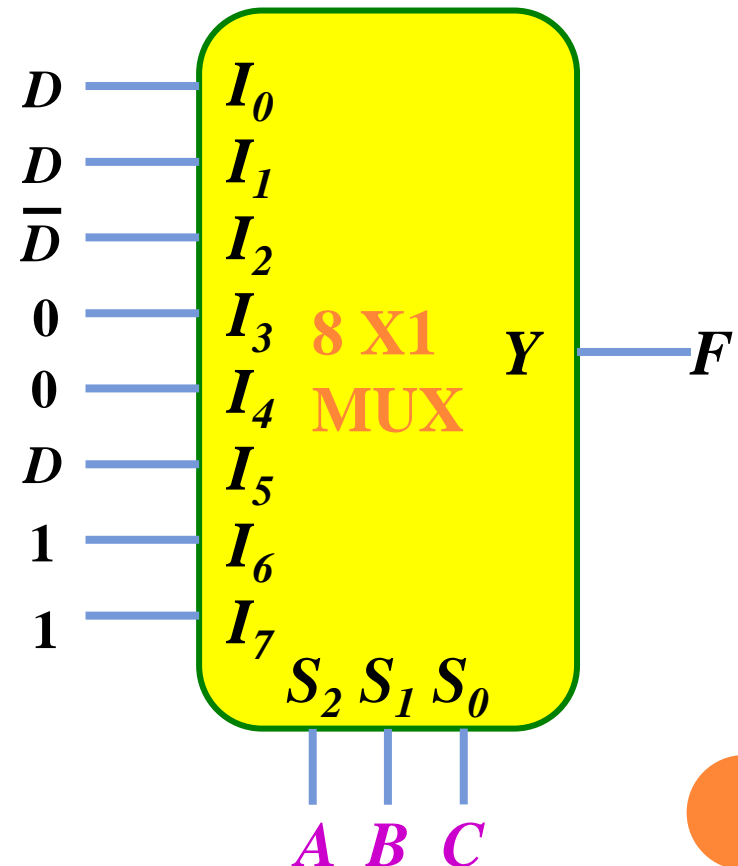
$F = z + z' = 1$



# IMPLEMENTATION USING MULTIPLEXERS: PROCEDURE 2

○  $F(A, B, C, D) = \sum(1, 3, 4, 11, 12, 13, 14, 15)$

| A | B | C | D | F |                 |
|---|---|---|---|---|-----------------|
| 0 | 0 | 0 | 0 | 0 | } $F = D$       |
| 0 | 0 | 0 | 1 | 1 |                 |
| 0 | 0 | 1 | 0 | 0 | } $F = D$       |
| 0 | 0 | 1 | 1 | 1 |                 |
| 0 | 1 | 0 | 0 | 1 | } $F = \bar{D}$ |
| 0 | 1 | 0 | 1 | 0 |                 |
| 0 | 1 | 1 | 0 | 0 | } $F = 0$       |
| 0 | 1 | 1 | 1 | 0 |                 |
| 1 | 0 | 0 | 0 | 0 | } $F = 0$       |
| 1 | 0 | 0 | 1 | 0 |                 |
| 1 | 0 | 1 | 0 | 0 | } $F = D$       |
| 1 | 0 | 1 | 1 | 1 |                 |
| 1 | 1 | 0 | 0 | 1 | } $F = 1$       |
| 1 | 1 | 0 | 1 | 1 |                 |
| 1 | 1 | 1 | 0 | 1 | } $F = 1$       |
| 1 | 1 | 1 | 1 | 1 |                 |





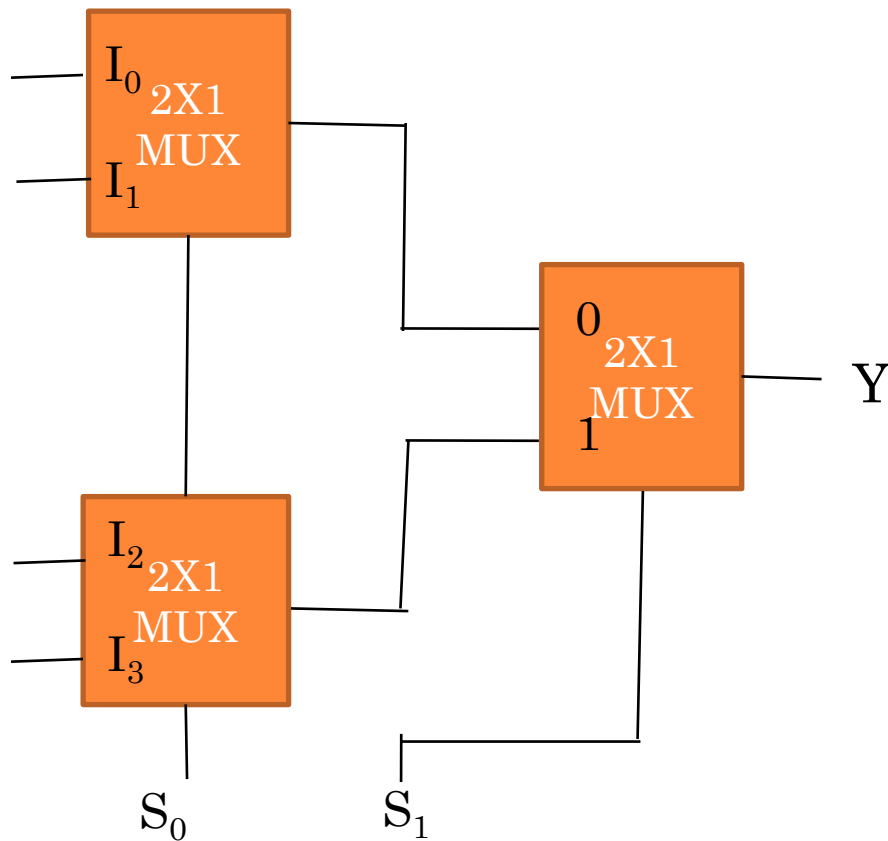
## PROCEDURE 1 VS PROCEDURE 2

- Among the function variables, if the first or some middle variable other than the last one is to be used in input line then procedure 1 is preferable.



# MULTIPLEXER EXPANSION

## 4-TO-1 MUX USING 2-TO-1 MUX

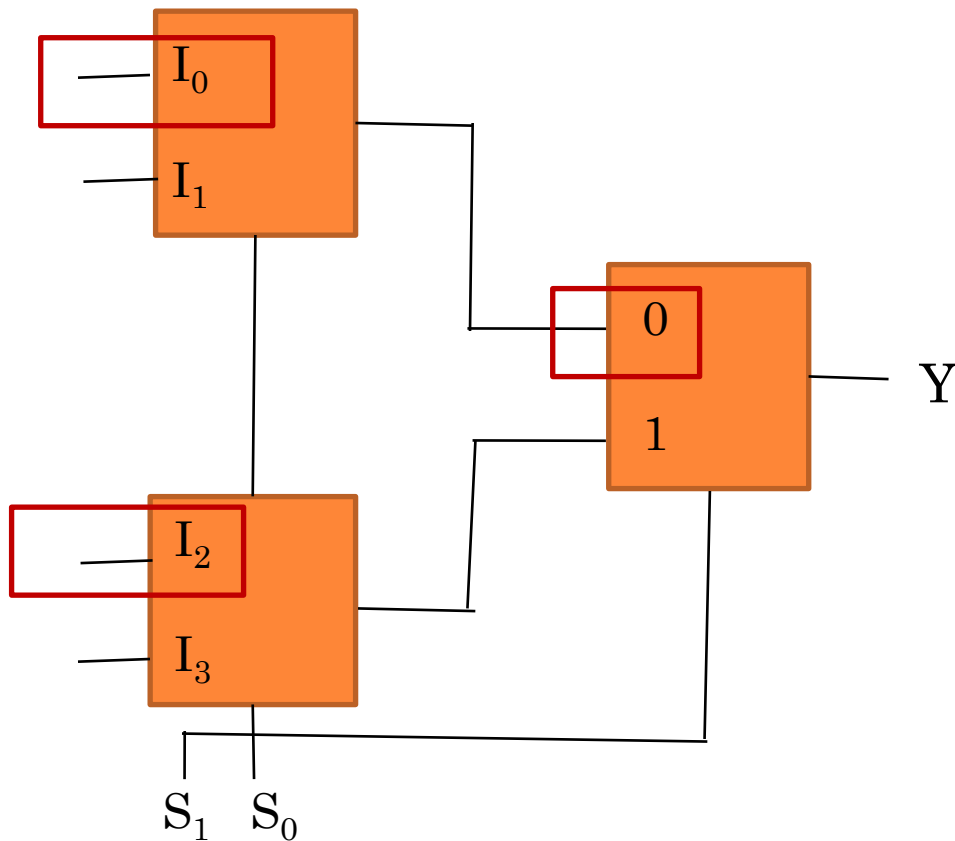


| $S_1$ | $S_0$ | $Y$   |
|-------|-------|-------|
| 0     | 0     | $I_0$ |
| 0     | 1     | $I_1$ |
| 1     | 0     | $I_2$ |
| 1     | 1     | $I_3$ |



# MULTIPLEXER EXPANSION

## 4-TO-1 MUX USING 2-TO-1 MUX

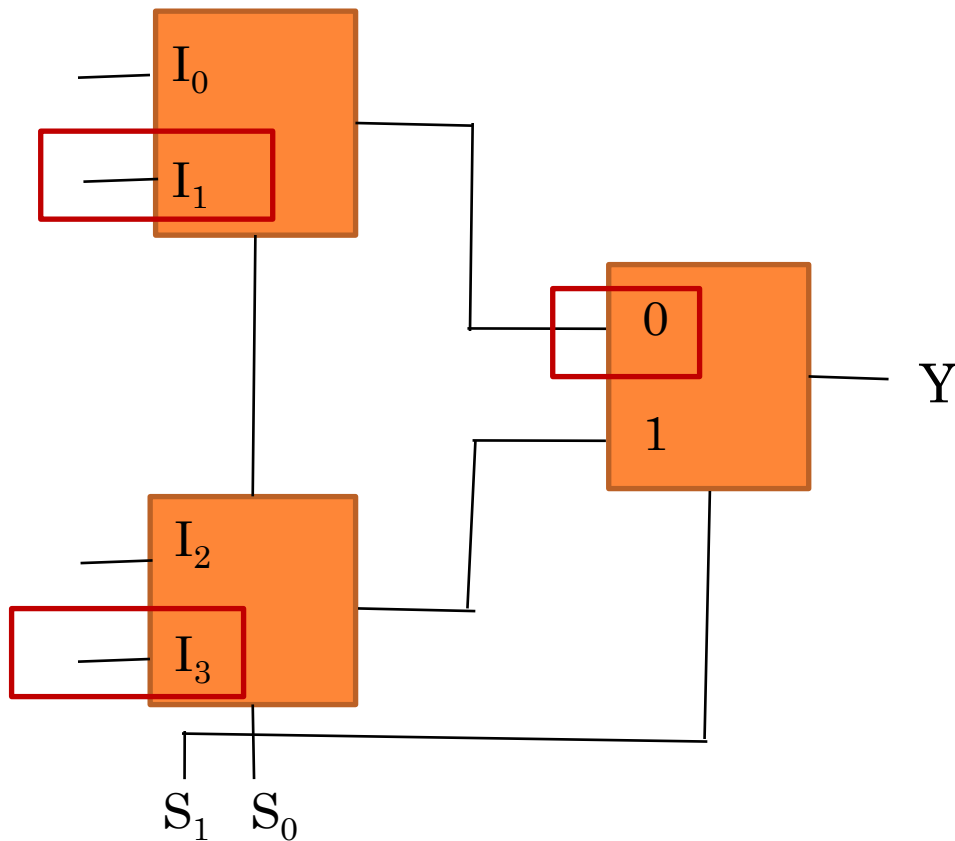


| $S_1$ | $S_0$ | $Y$   |
|-------|-------|-------|
| 0     | 0     | $I_0$ |
| 0     | 1     | $I_1$ |
| 1     | 0     | $I_2$ |
| 1     | 1     | $I_3$ |



# MULTIPLEXER EXPANSION

## 4-TO-1 MUX USING 2-TO-1 MUX

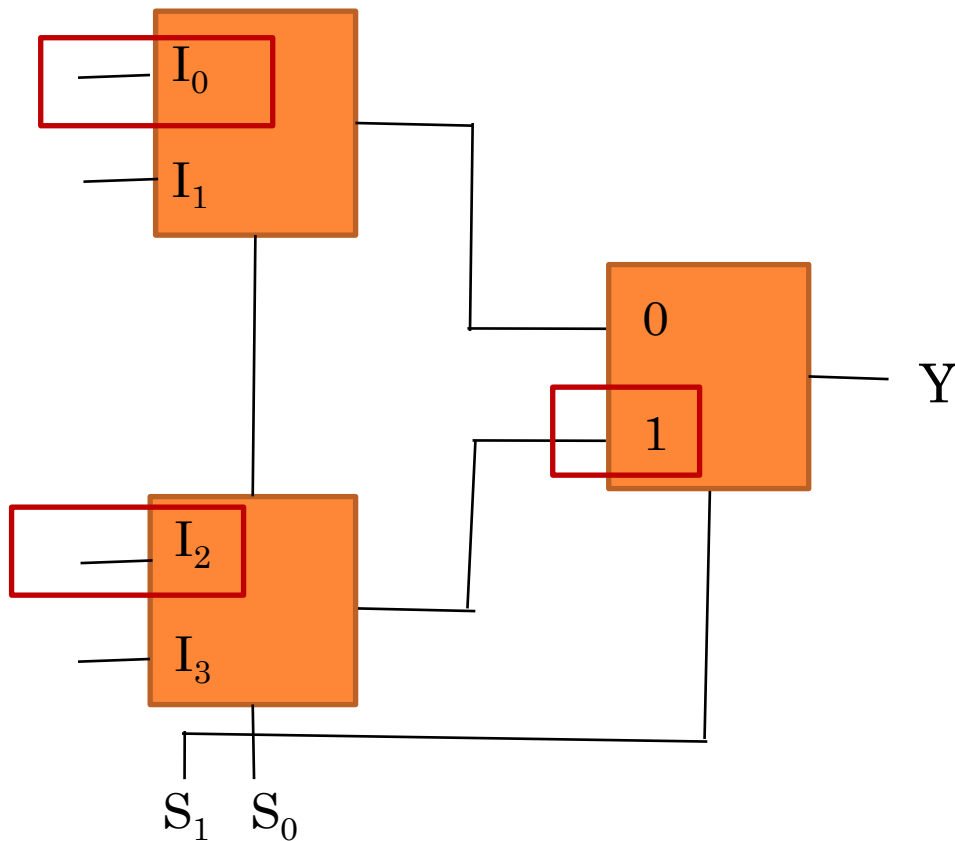


| $S_1$ | $S_0$ | $Y$   |
|-------|-------|-------|
| 0     | 0     | $I_0$ |
| 0     | 1     | $I_1$ |
| 1     | 0     | $I_2$ |
| 1     | 1     | $I_3$ |



# MULTIPLEXER EXPANSION

## 4-TO-1 MUX USING 2-TO-1 MUX

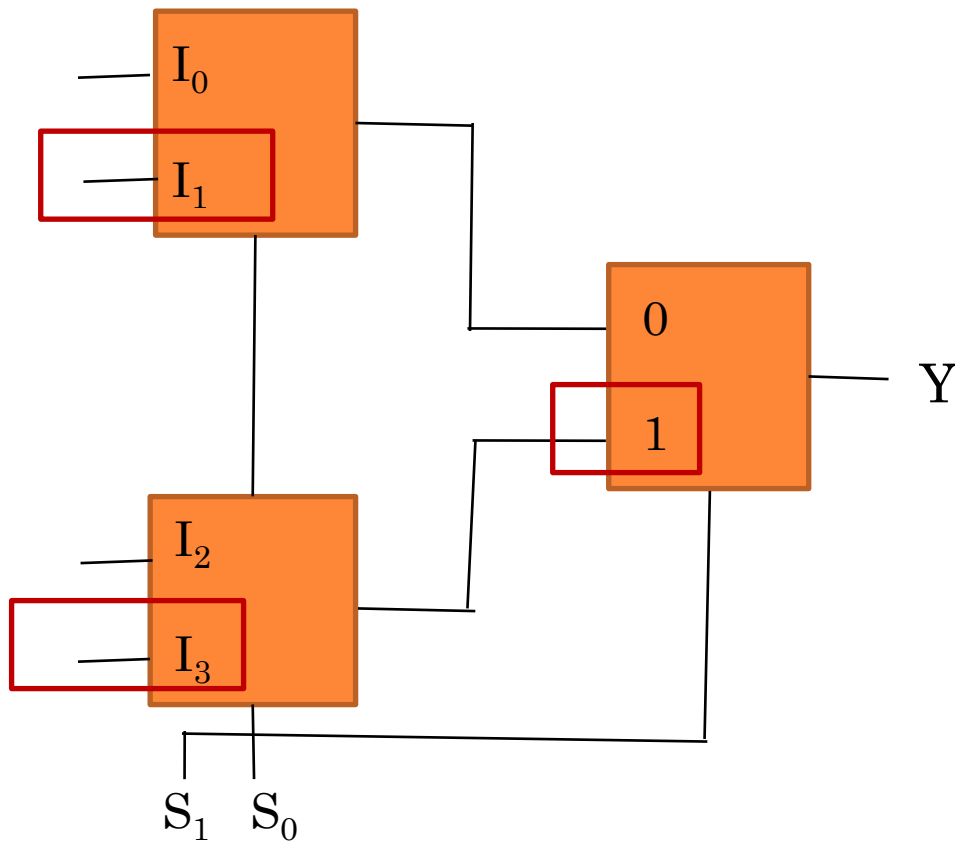


| $S_1$ | $S_0$ | Y     |
|-------|-------|-------|
| 0     | 0     | $I_0$ |
| 0     | 1     | $I_1$ |
| 1     | 0     | $I_2$ |
| 1     | 1     | $I_3$ |



# MULTIPLEXER EXPANSION

## 4-TO-1 MUX USING 2-TO-1 MUX

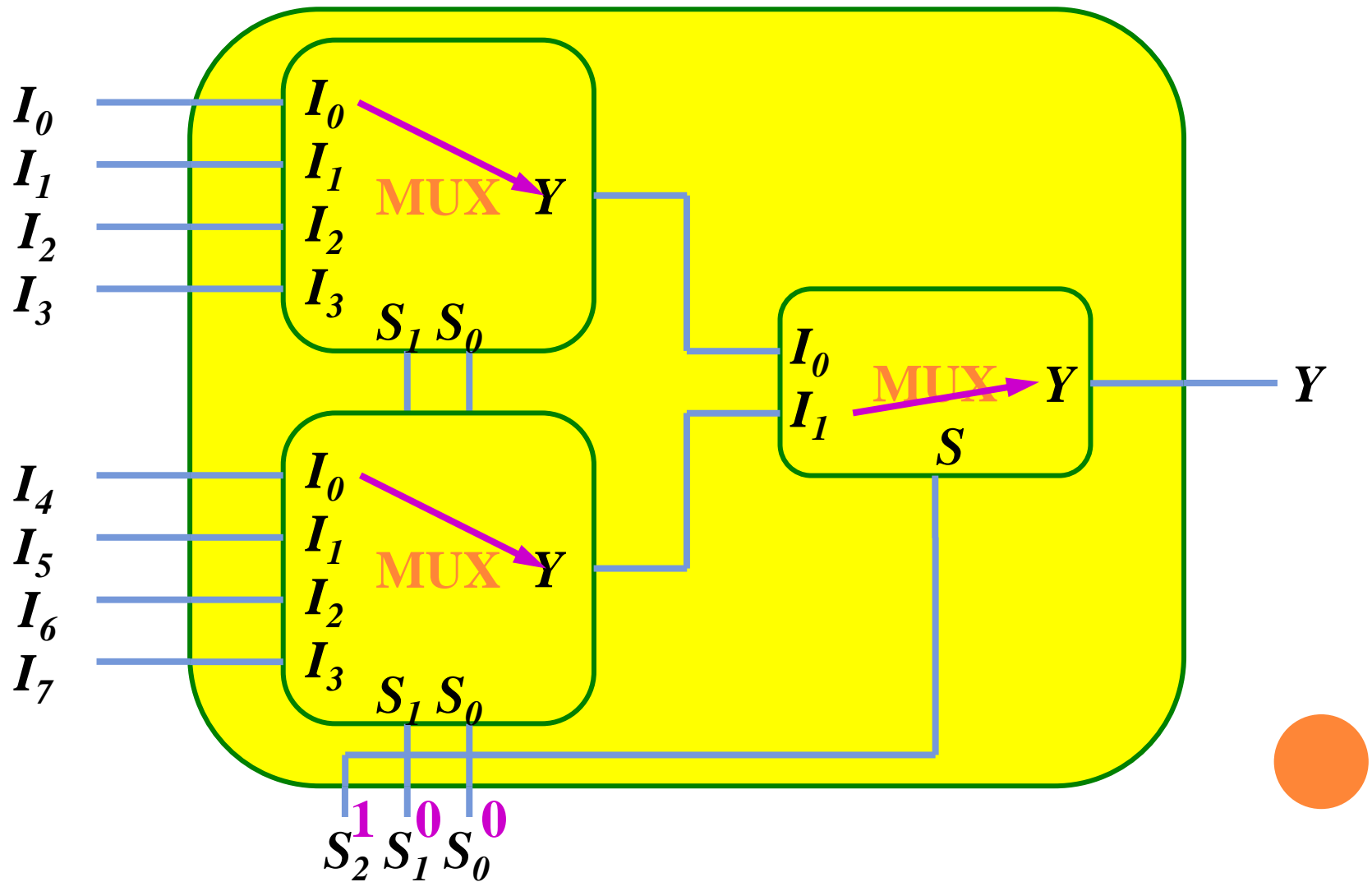


| $S_1$ | $S_0$ | $Y$   |
|-------|-------|-------|
| 0     | 0     | $I_0$ |
| 0     | 1     | $I_1$ |
| 1     | 0     | $I_2$ |
| 1     | 1     | $I_3$ |



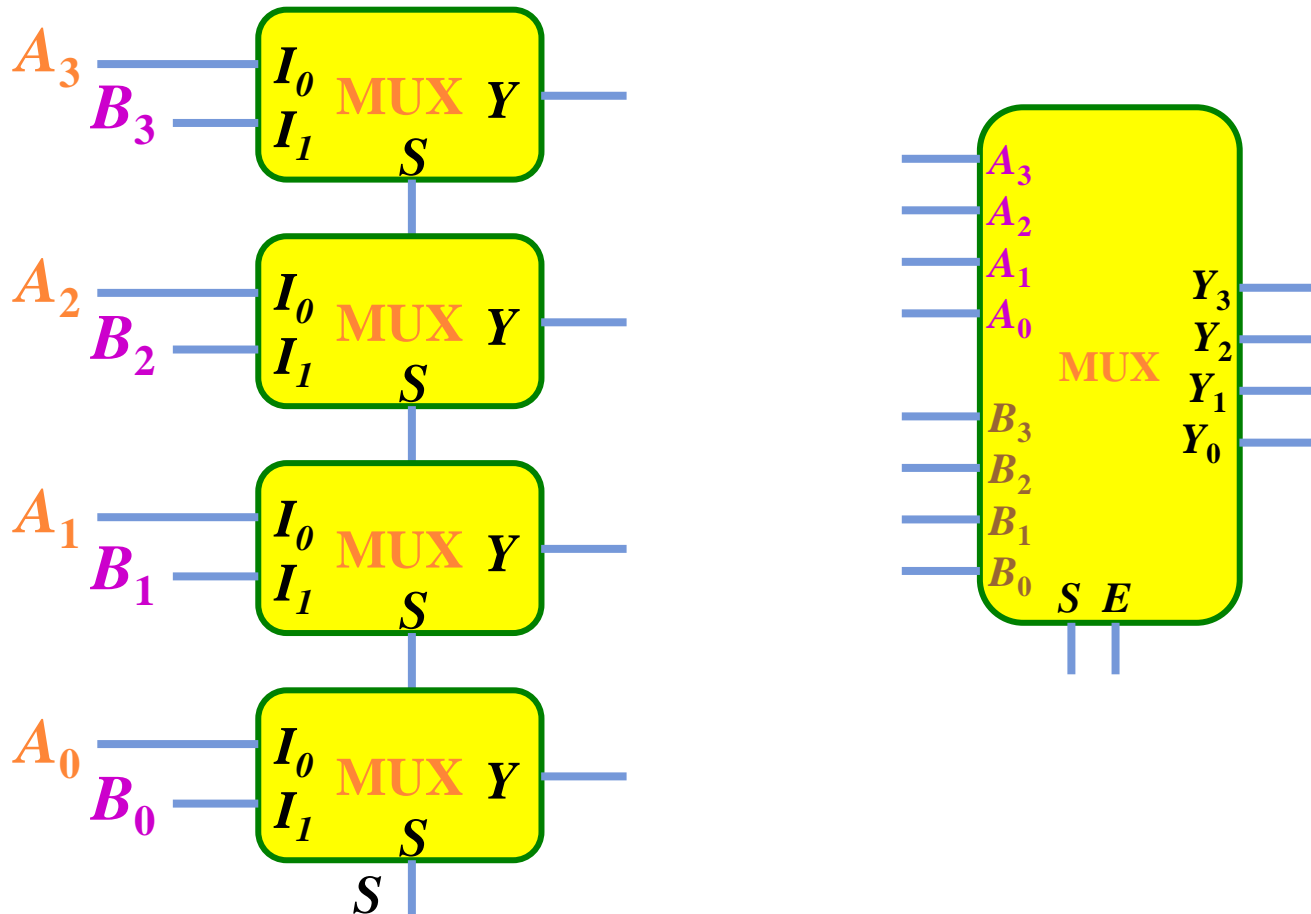
# MULTIPLEXER EXPANSION [SELF STUDY]

## 8-TO-1 MUX USING DUAL 4-TO-1 MUX



# MULTIPLEXERS

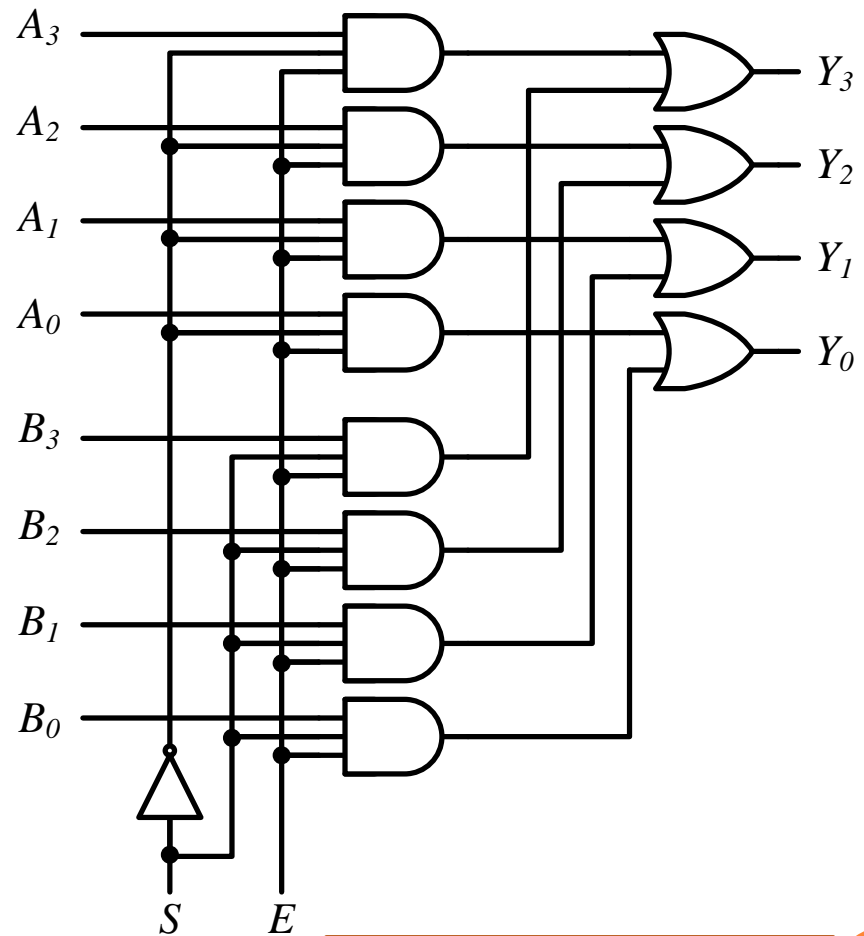
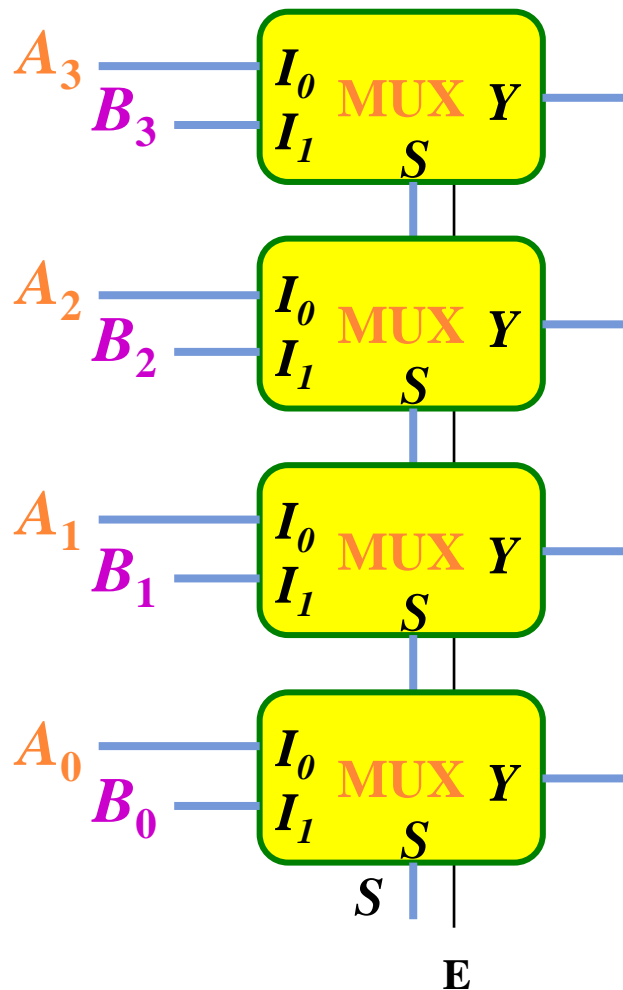
- Quad 2-to-1 MUX:  
Four 2x1 MUX can be used simultaneously





# MULTIPLEXERS

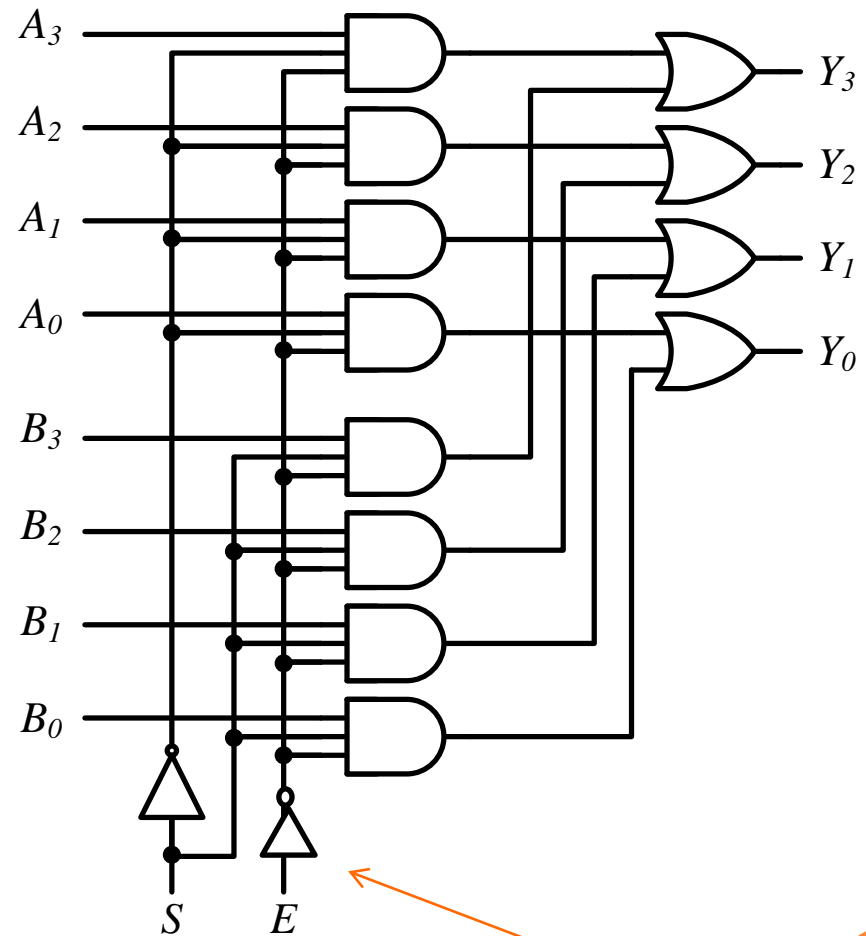
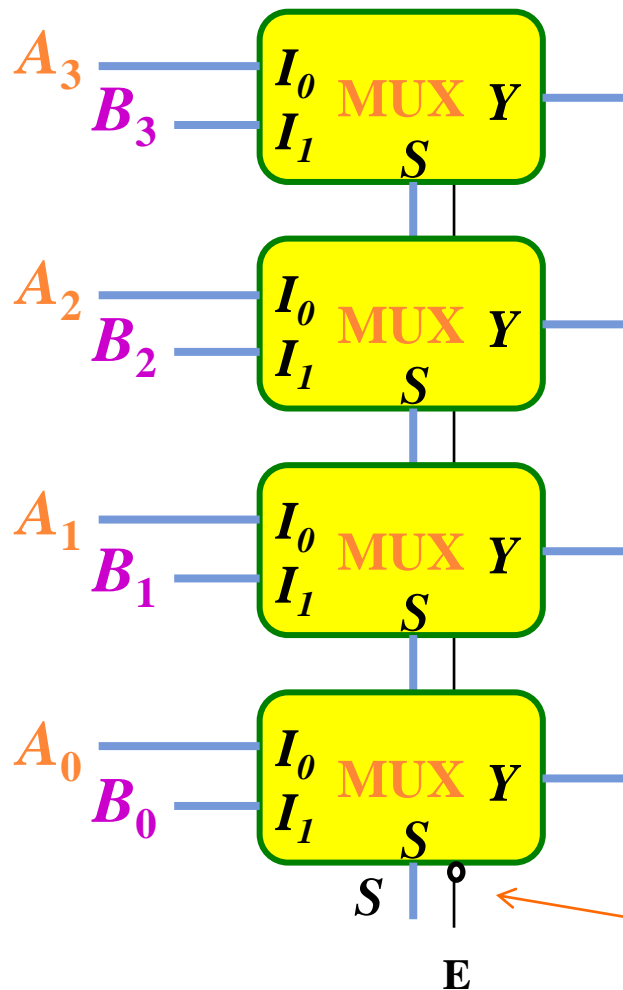
## Quad 2-to-1 MUX



Active **High** Enable:  
The output is  
enabled when  $E=1$

# MULTIPLEXERS

## Quad 2-to-1 MUX



Active **Low** Enable:  
The output is  
enabled when  $E=0$