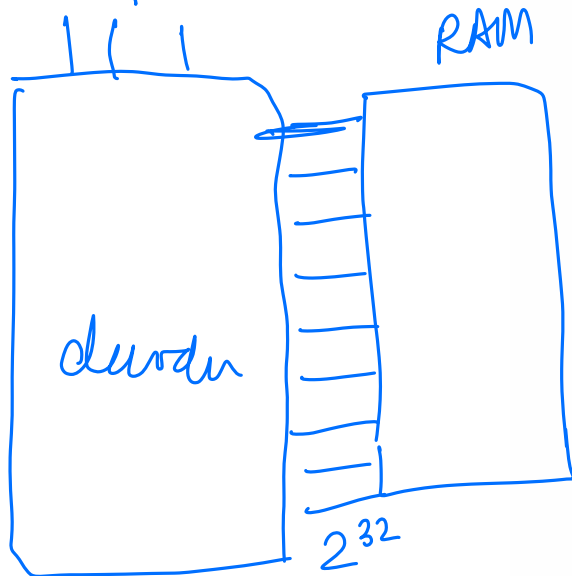
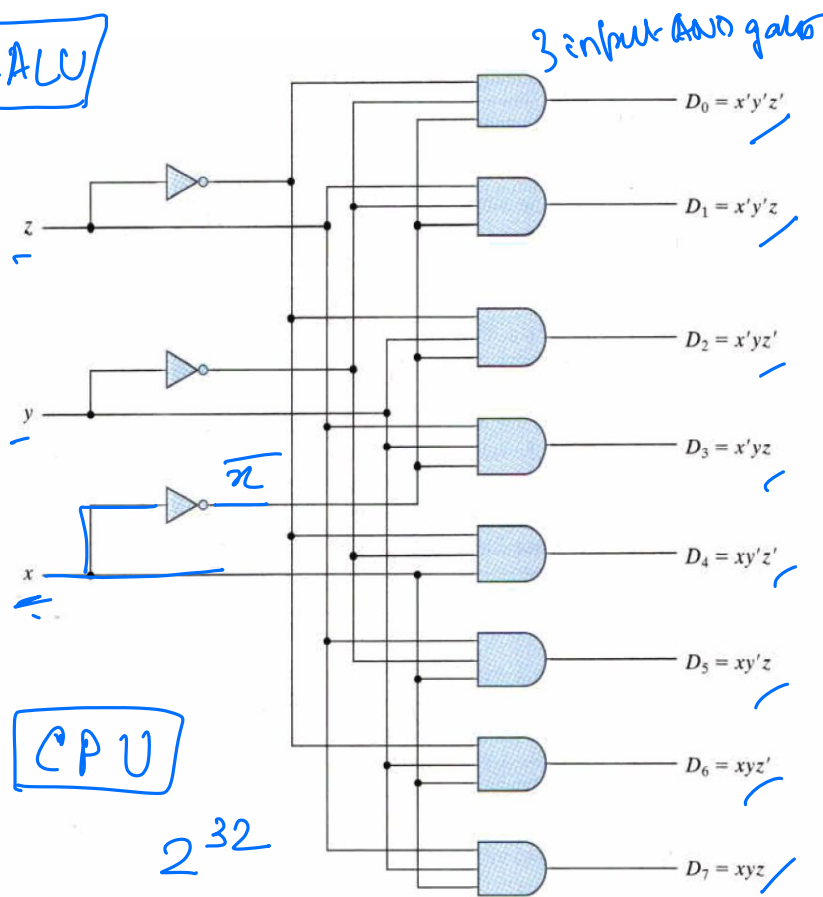


# Decoder

input:  $n$  bits  
output:  $2^n$



64bit ALU



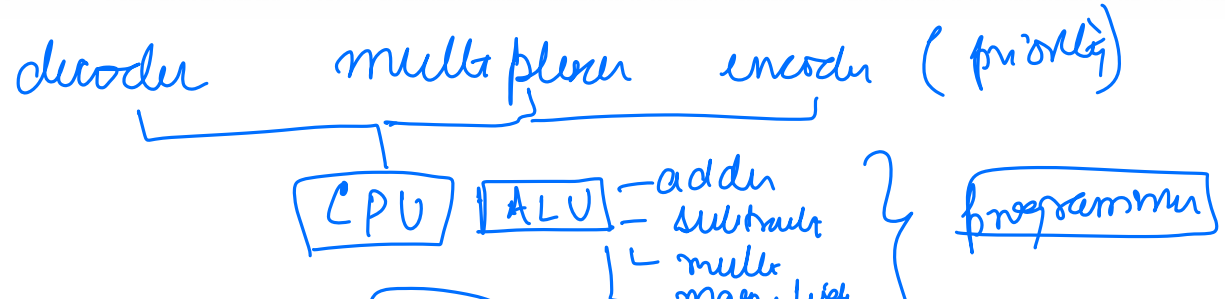
Truth Table of a Three-to-Eight-Line Decoder

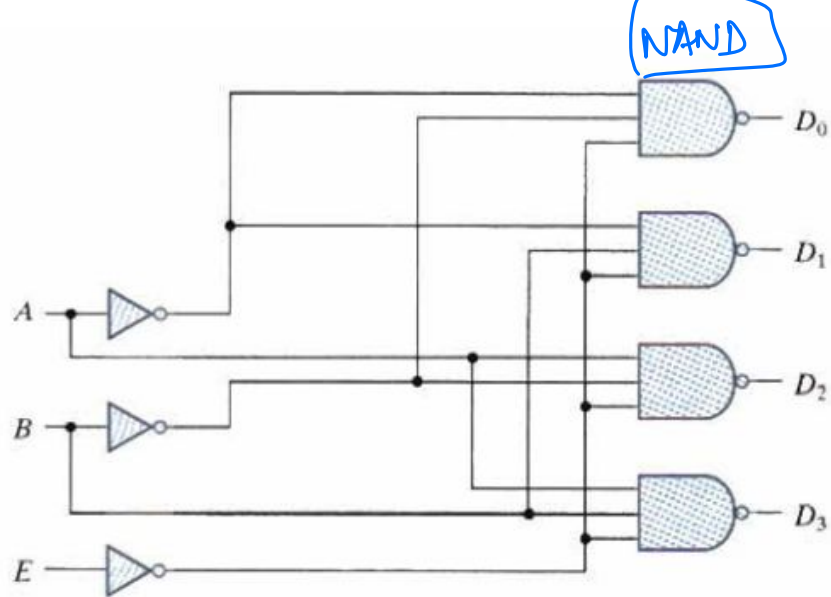
no address / → always have address line order

0  
1  
2  
1  
2  
1  
2  
1  
2  
1  
2

| Inputs <sup>3</sup> |   |   | Outputs <sup>2<sup>3</sup></sup> |                |                |                |                |                |                |                |
|---------------------|---|---|----------------------------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| x                   | y | z | D <sub>0</sub>                   | D <sub>1</sub> | D <sub>2</sub> | D <sub>3</sub> | D <sub>4</sub> | D <sub>5</sub> | D <sub>6</sub> | D <sub>7</sub> |
| 0                   | 0 | 0 | 1                                | 0              | 0              | 0              | 0              | 0              | 0              | 0              |
| 0                   | 0 | 1 | 0                                | 1              | 0              | 0              | 0              | 0              | 0              | 0              |
| 0                   | 1 | 0 | 0                                | 0              | 1              | 0              | 0              | 0              | 0              | 0              |
| 0                   | 1 | 1 | 0                                | 0              | 0              | 1              | 0              | 0              | 0              | 0              |
| 1                   | 0 | 0 | 0                                | 0              | 0              | 0              | 1              | 0              | 0              | 0              |
| 1                   | 0 | 1 | 0                                | 0              | 0              | 0              | 0              | 1              | 0              | 0              |
| 1                   | 1 | 0 | 0                                | 0              | 0              | 0              | 0              | 0              | 1              | 0              |
| 1                   | 1 | 1 | 0                                | 0              | 0              | 0              | 0              | 0              | 0              | 1              |

m<sub>0</sub>  
m<sub>1</sub>





(a) Logic diagram

**FIGURE 4.19**  
Two-to-four-line decoder with enable input

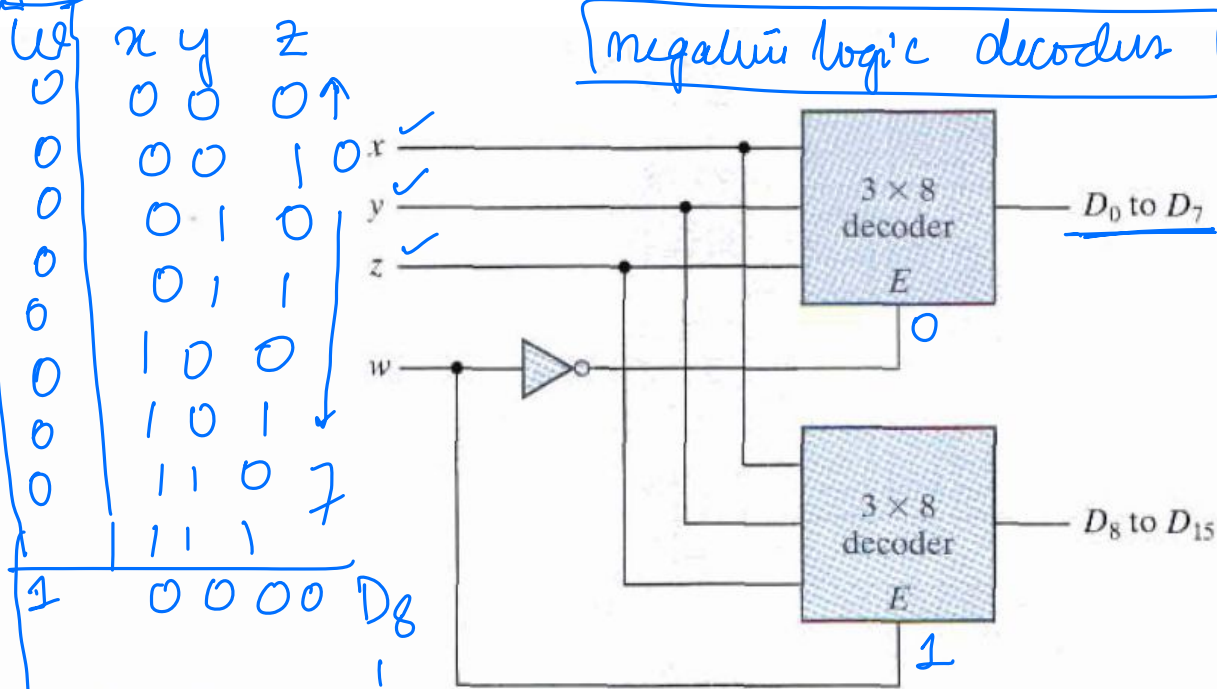
enable

| $E$ | $A$ | $B$ | $D_0$ | $D_1$ | $D_2$ | $D_3$ |
|-----|-----|-----|-------|-------|-------|-------|
| 1   | X   | X   | 1     | 1     | 1     | 1     |
| 0   | 0   | 0   | 0     | 1     | 1     | 1     |
| 0   | 0   | 1   | 1     | 0     | 1     | 1     |
| 0   | 1   | 0   | 1     | 1     | 0     | 1     |
| 0   | 1   | 1   | 1     | 1     | 1     | 0     |

(b) Truth table

Enable

## Negative logic decoder

**FIGURE 4.20**

4 × 16 decoder constructed with two 3 × 8 decoders

Dis -

Any function

Sum of products

Any truth

table

↓

Some Boolean  
function

↓

can use decoder

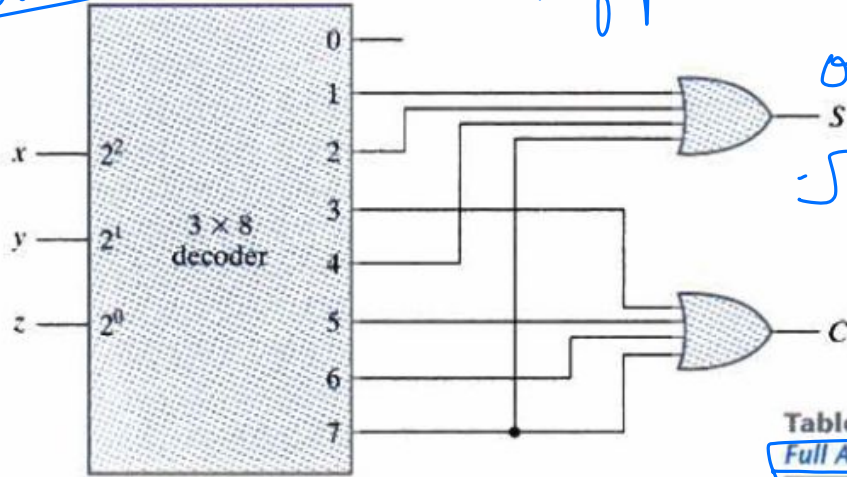


**FIGURE 4.21**  
Implementation of a full adder with a decoder

$$S(x, y, z) = \Sigma(1, 2, 4, 7)$$

$$C(x, y, z) = \Sigma(3, 5, 6, 7)$$

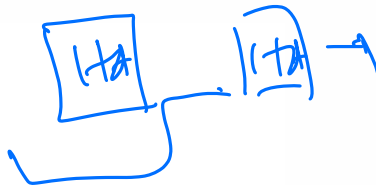
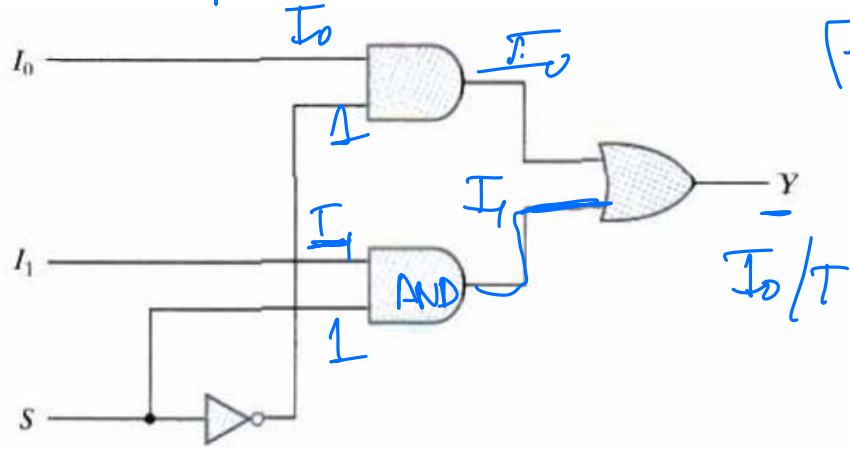


**Table 4.4**  
**Full Adder**

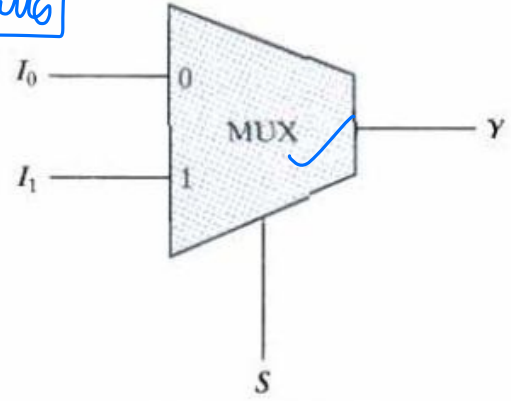
| x | y | z | C | S |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

Multiplex



(a) Logic diagram

inputs



(b) Block diagram

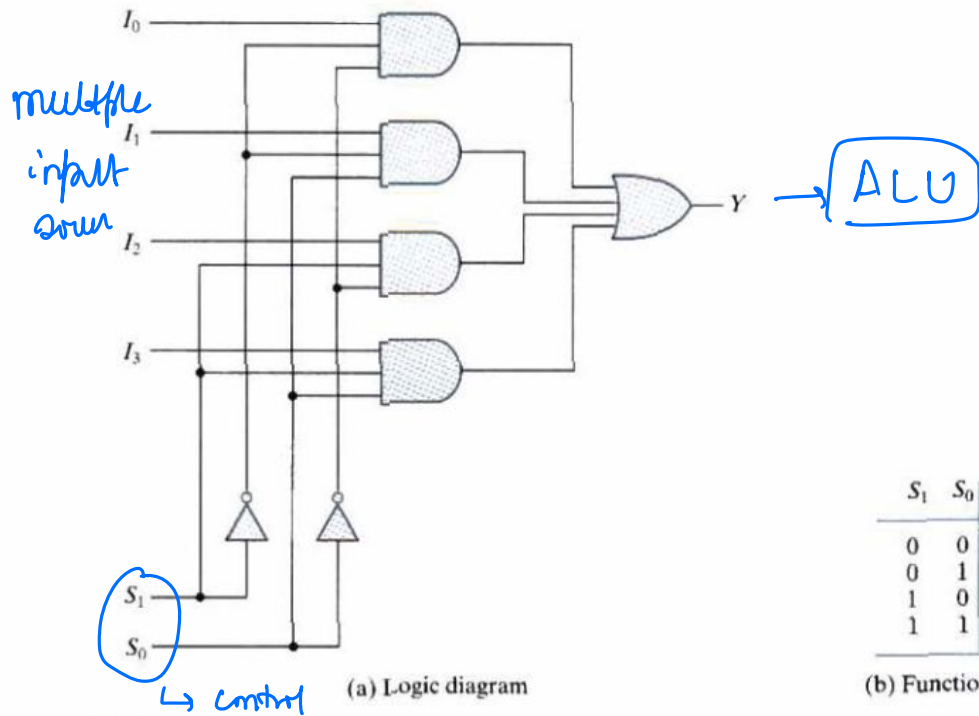
**FIGURE 4.24**

Two-to-one-line multiplexer

library of  
modules

$2^n$  inputs  
 $n$  select lines

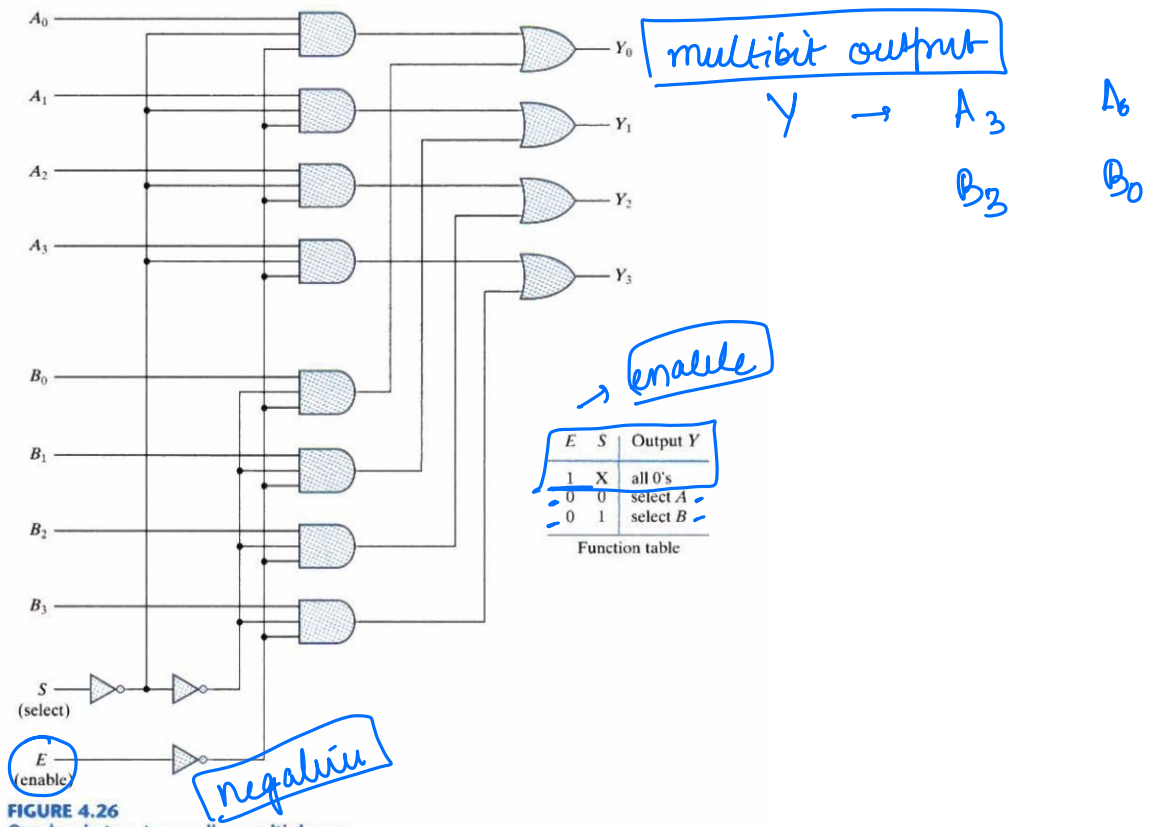
one output



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

(b) Function table

**FIGURE 4.25**  
Four-to-one-line multiplexer



| $E$ | $S$ | Output $Y$ |
|-----|-----|------------|
| 1   | X   | all 0's    |
| 0   | 0   | select $A$ |
| 0   | 1   | select $B$ |

**FIGURE 4.26**  
Quaduple two-to-one-line multiplexer

$$F(x, y, z) = \Sigma(1, 2, 6, 7)$$

$n$  inputs for a function

$(n-1)$  as select signals

Truth table  
select signals

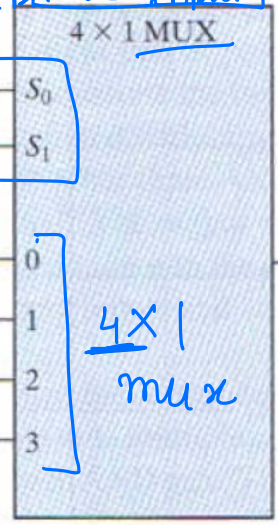
| x | y | z | F |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

4 minterms  
0/1

last bit as input of MUX

Inputs

y —  $S_0$   
x —  $S_1$



1 output

$n$  bits input

$\frac{2^n}{\text{minterms}}$

$2^{2^n}$  functions

$(n-1)$  bits  
1 bus

(a) Truth table

(b) Multiplexer implementation

FIGURE 4.27

Implementing a Boolean function with a multiplexer

last bit  $\rightarrow$  toggles between condition

Linux

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

select inputs -

| 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 | $F = D'$ |
| 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 |          |

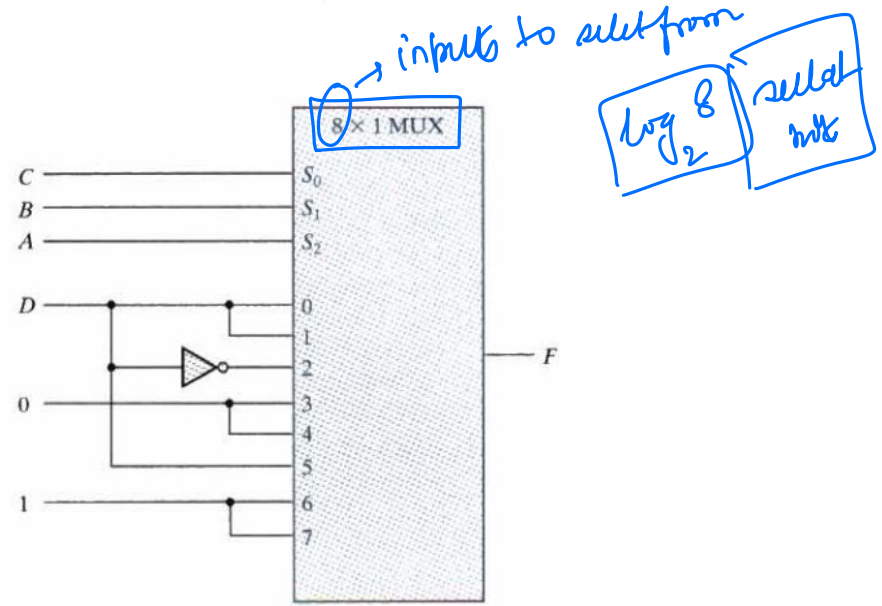


**FIGURE 4.28**  
Implementing a four-input function with a multiplexer

CPU

Table 4.7

Truth Table of an Octal-to-Binary Encoder

| Inputs |       |       |       |       |       |       |       | Outputs |     |     |
|--------|-------|-------|-------|-------|-------|-------|-------|---------|-----|-----|
| $D_0$  | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ | $x$     | $y$ | $z$ |
| 1      | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0       | 0   | 0   |
| 0      | 1     | 0     | 0     | 0     | 0     | 0     | 0     | 0       | 0   | 1   |
| 0      | 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0       | 1   | 0   |
| 0      | 0     | 0     | 1     | 0     | 0     | 0     | 0     | 0       | 1   | 1   |
| 0      | 0     | 0     | 0     | 1     | 0     | 0     | 0     | 1       | 0   | 0   |
| 0      | 0     | 0     | 0     | 0     | 1     | 0     | 0     | 1       | 0   | 1   |
| 0      | 0     | 0     | 0     | 0     | 0     | 1     | 0     | 1       | 1   | 0   |
| 0      | 0     | 0     | 0     | 0     | 0     | 0     | 1     | 1       | 1   | 1   |

$$z = D_1 + D_3 + D_5 + D_7$$

$$y = D_2 + D_3 + D_6 + D_7$$

$$x = D_4 + D_5 + D_6 + D_7$$

encoder

Priority encoder

interrupt

mux

high at same

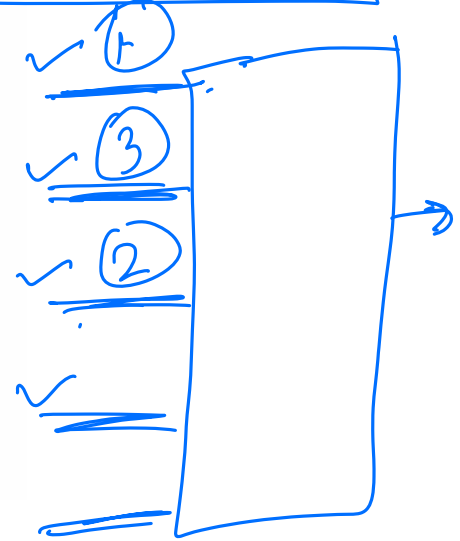


**Table 4.8**  
Truth Table of a Priority Encoder

24 rows

| Inputs |       |       |       | Outputs |     |     |
|--------|-------|-------|-------|---------|-----|-----|
| $D_0$  | $D_1$ | $D_2$ | $D_3$ | $x$     | $y$ | $V$ |
| 0      | 0     | 0     | 0     | X       | X   | 0   |
| 1      | 0     | 0     | 0     | 0       | 0   | 1   |
| X      | 1     | 0     | 0     | 0       | 1   | 1   |
| X      | X     | 1     | 0     | 1       | 0   | 1   |
| X      | X     | X     | 1     | 1       | 1   | 1   |

validity  
multiple input  
invalid by  
 $D_0 - D_7$   
can be active

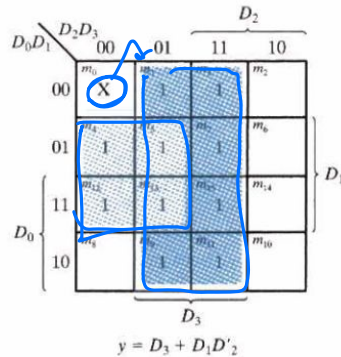
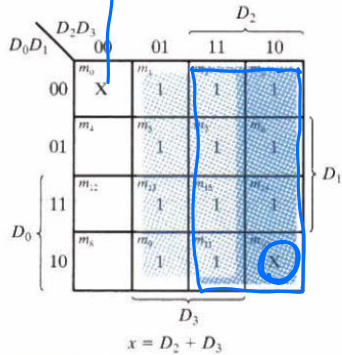


**FIGURE 4.22**  
Maps for a priority encoder

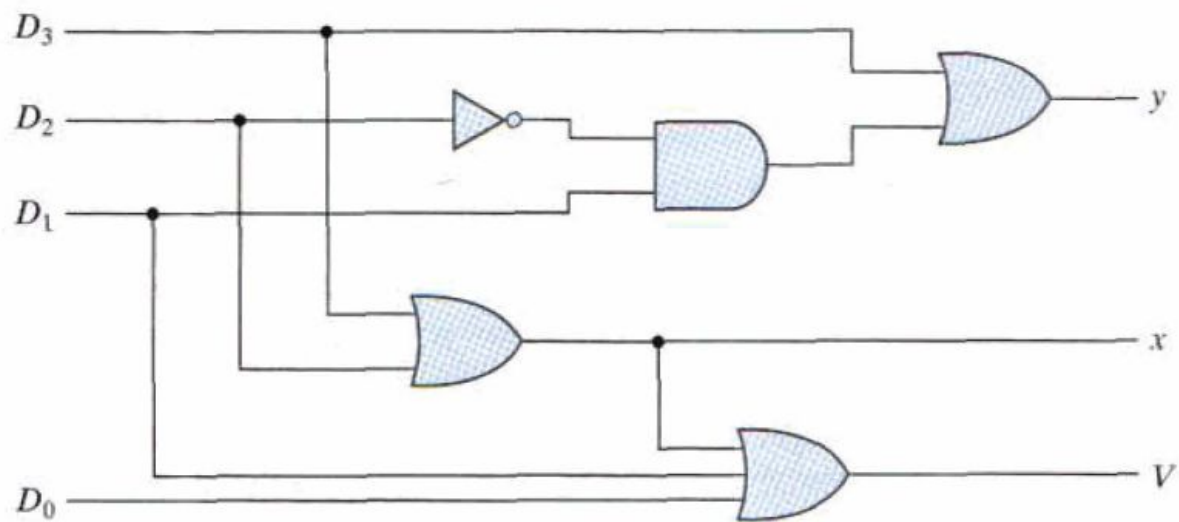


**FIGURE 4.23**  
Four-input priority encoder