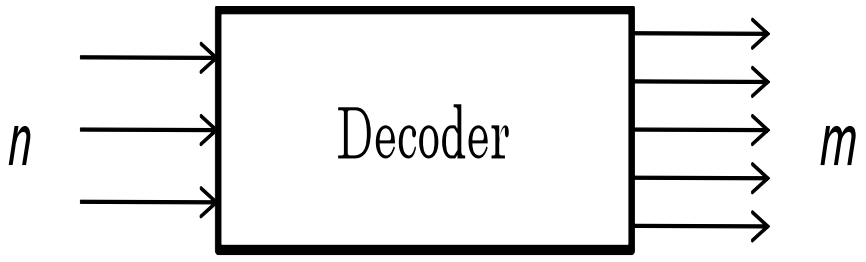
Lecture – 12

Outline

- Classical CL circuits
  - **Decoder**
  - **Encoder**
  - **Multiplexer**

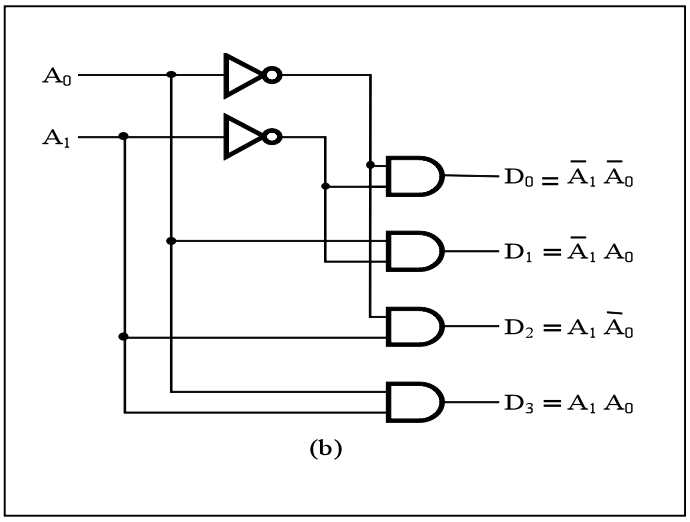
Decoder

Decoder – converts n-bit input lines to a maximum of 2^n unique output lines



n-to-m line decoders, where $n \leq m \leq 2^n$

- Function: generate 2^n (or fewer) minterms for the n input variables



- 2-to-4-Line Decoder

Each line D_i is a minterm m_i

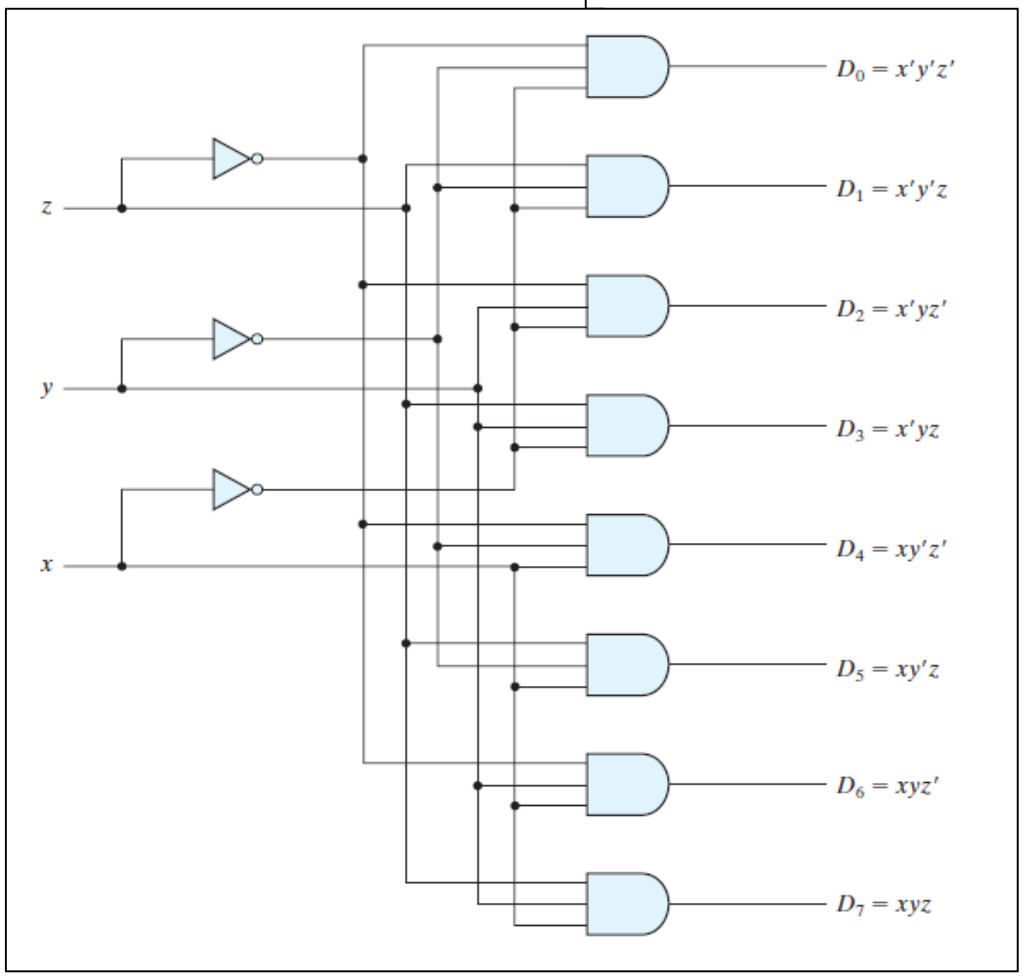
A_1	A_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

(a)

Decoder Examples

- 3-to-8-Line Decoder

Inputs			Outputs							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
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



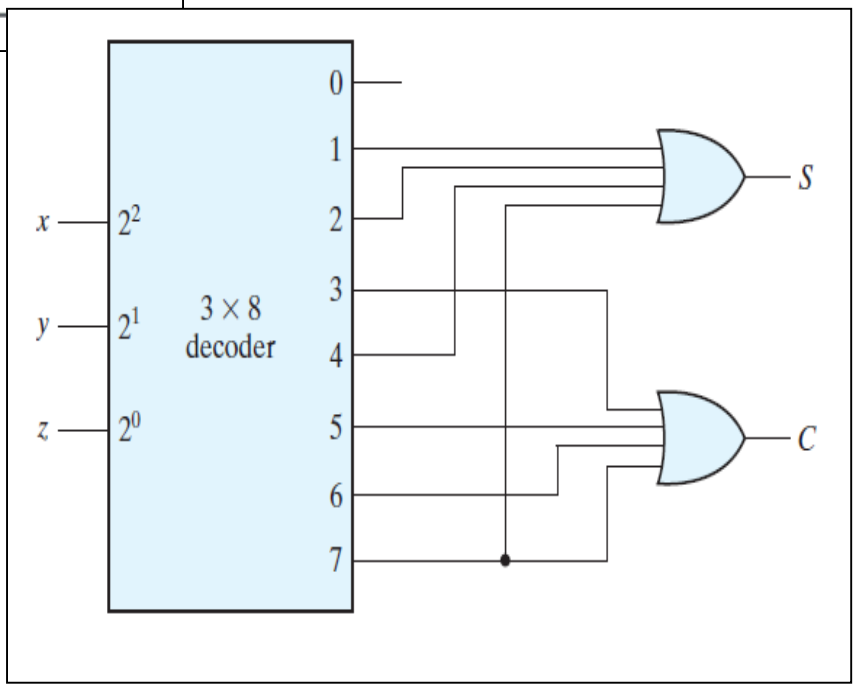
Application of Decoder

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

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

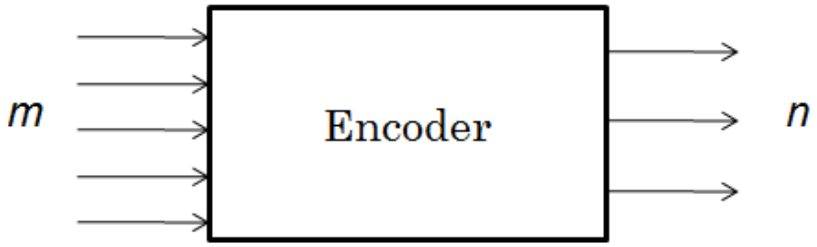
$C(x,y,z)=\sum(3,5,6,7)$



Encoder

- Encoder- the opposite of decoding – convert m-bit input line to a n-bit unique output line

  with $n \leq m \leq 2^n$



Encoder Example

- Inputs: 4 bits corresponding to decimal digits 0 through 3, (D_0, …, D_3)
- Outputs: 2 bits with binary codes
- Function: If input bit D_i is a 1, then the output (A_1, A_0) is the binary code for i

Input				Output	
D_3	D_2	D_1	D_0	A_1	A_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Input: 0 1 1 0 Output ?

Input: 0 0 0 0 Output ?

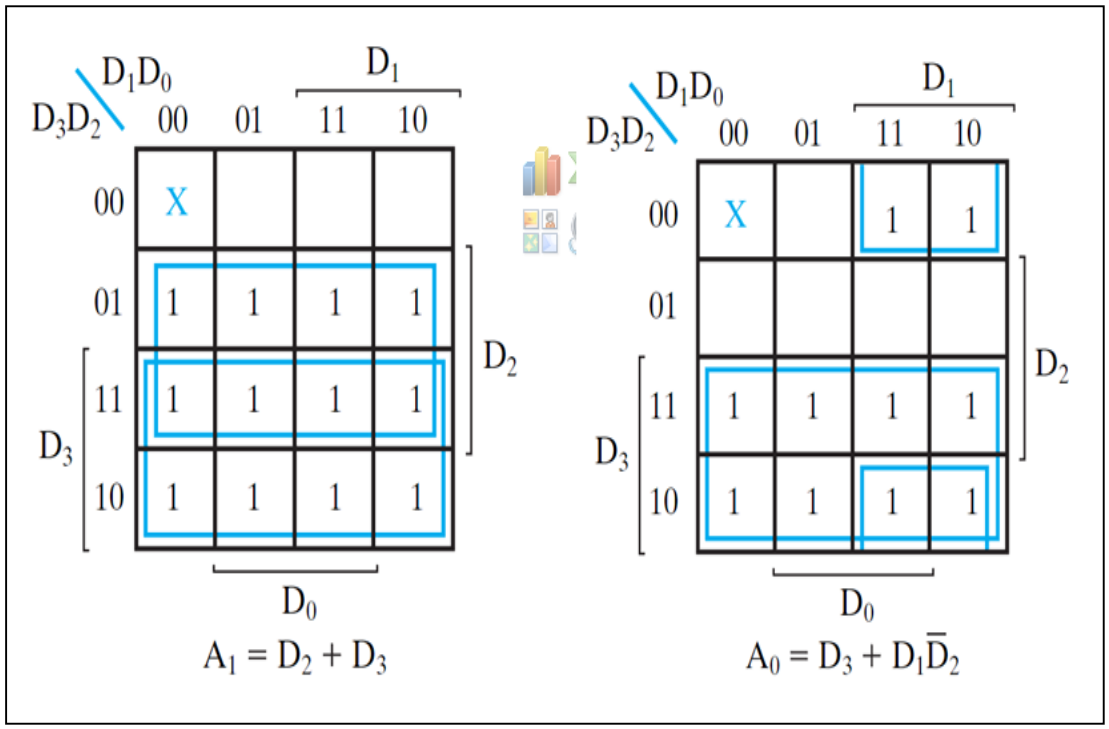
Priority Encoder

Among the 1s that appear, it selects the most significant input position (or the least significant input position) containing a 1 and responds with the corresponding binary code for that position.

Inputs				Outputs		
D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

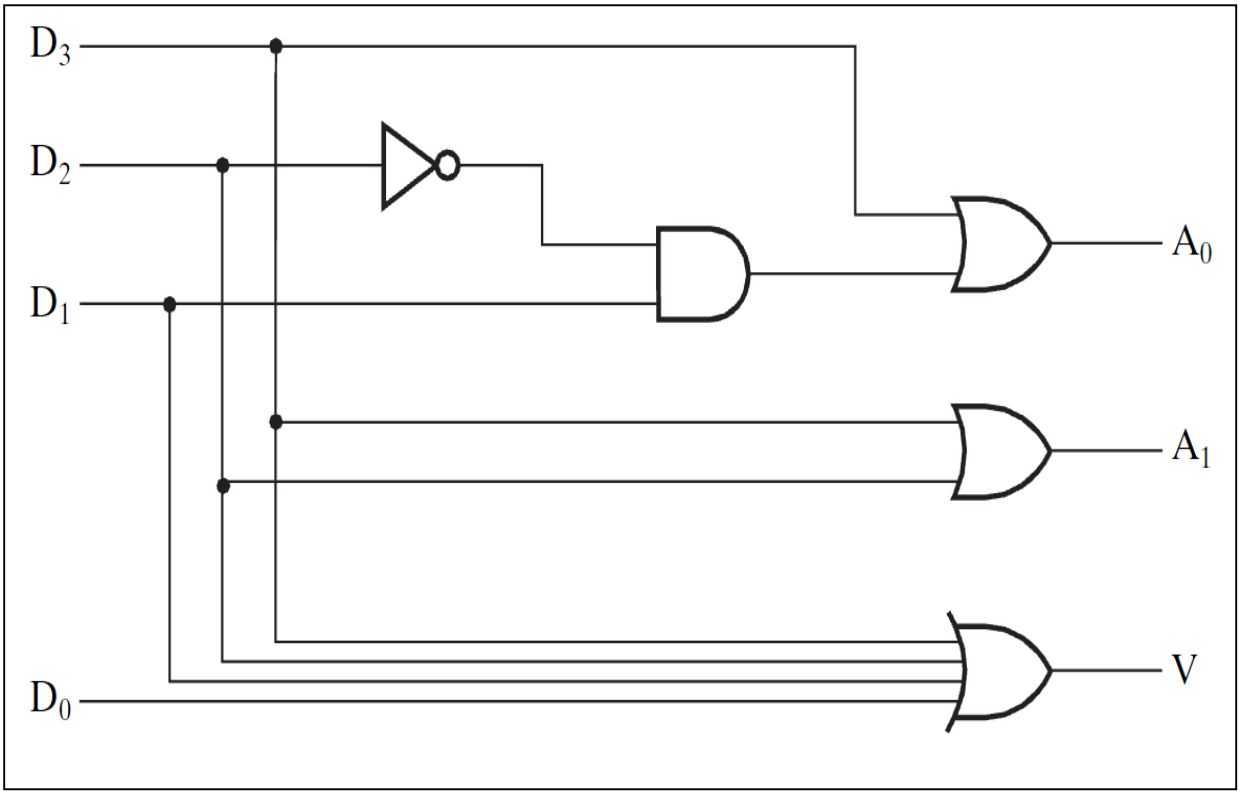
Priority Encoder

Inputs				Outputs		
D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1



Priority Encoder

$A_0=D_3+D_1D_2'$ $A_1=D_2+D_3$ $V=D_0+D_1+D_2+D_3$

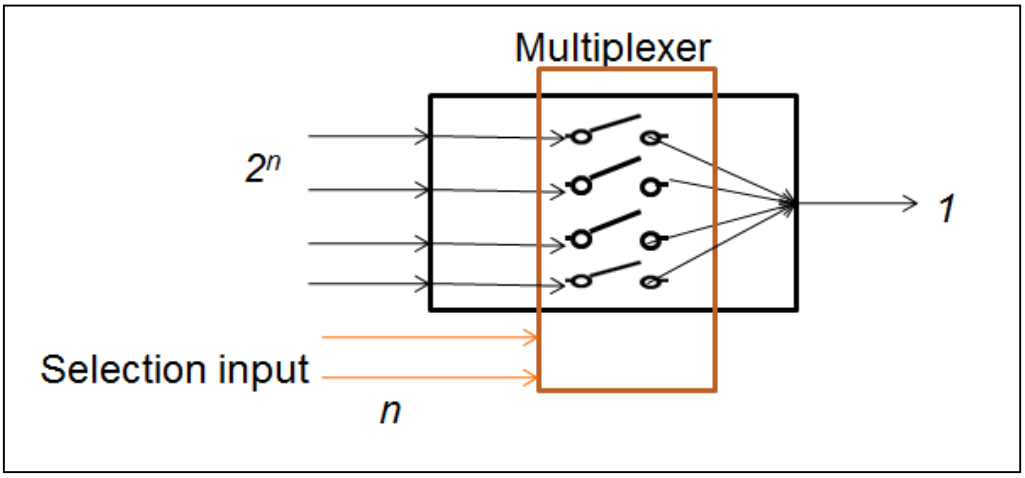


Outline

- Analysis of Combinational Logic (CL)
- Design of CL
- Classical CL circuits
  - Adder
  - Subtractor
  - Comparator
  - Decoder
  - Encoder
  - Multiplexer

Multiplexer

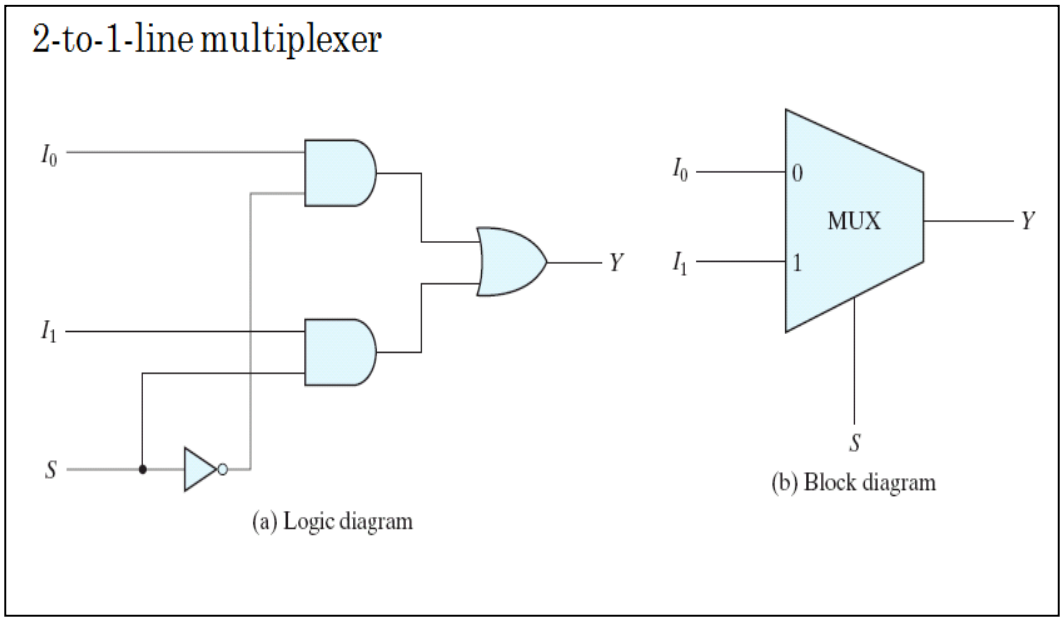
- A multiplexer selects binary information from **one** of many inputs and directs it to a **single** output line



- A typical multiplexer has n control inputs (S_{n_1}, … S_0) called *selection inputs*, 2^n information inputs ($I_{2^n\_1}$, … I_0), and one output Y

Multiplexer

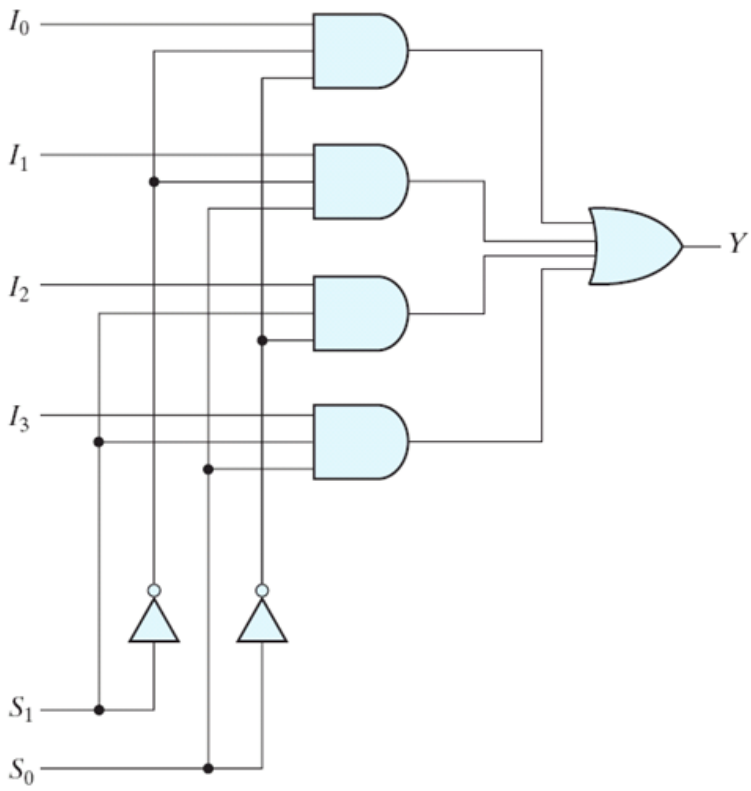
- 2-to-1-line multiplexer



Multiplexer

- 4-to-1-line multiplexer

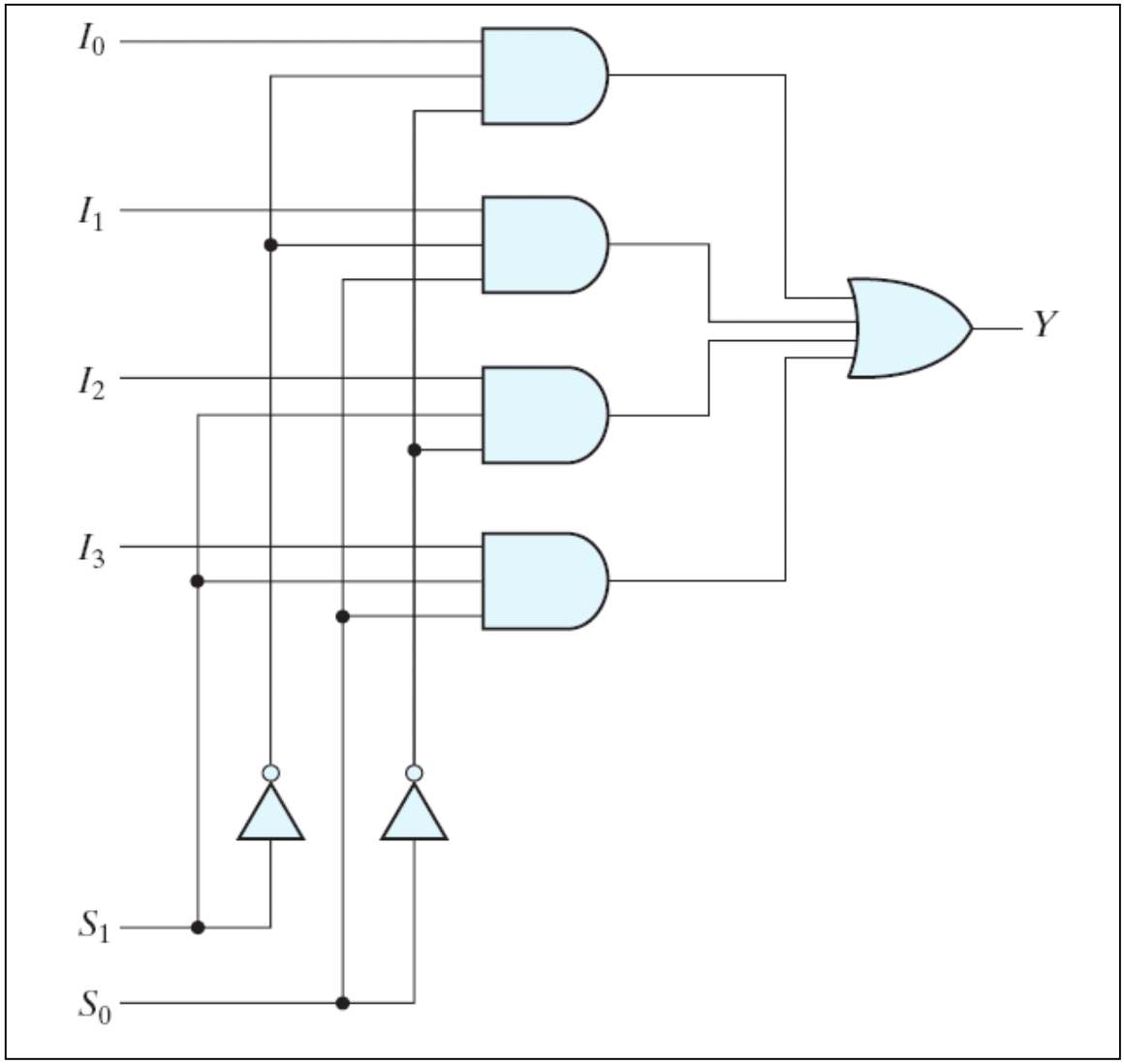
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



Multiplexer

In general, for an 2^n-to-1-line multiplexer:

- 2^n data input
- n selection input
- 1 output
- n inverter
- 2^n AND gates
- 1 OR gate



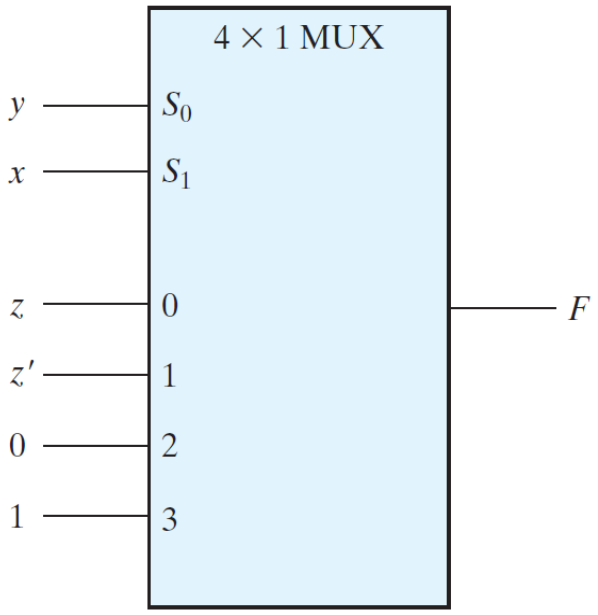
Application of Multiplexer

- Implement Boolean function

  F(x, y, z)=∑(1,2,6,7)

x	y	z	F	
0	0	0	0	$F = z$
0	0	1	1	
0	1	0	1	$F = z'$
0	1	1	0	
1	0	0	0	$F = 0$
1	0	1	0	
1	1	0	1	$F = 1$
1	1	1	1	

(a) Truth table

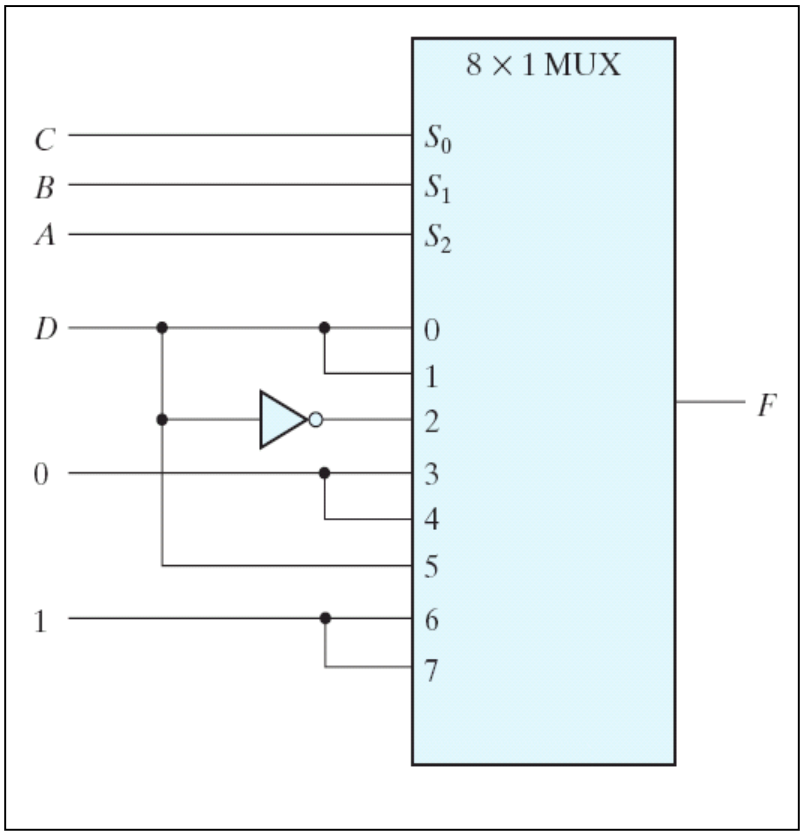


(b) Multiplexer implementation

Application of Multiplexer

F(A,B,C,D)=Π(0,2,5,6,7,8,9,10)

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	



Construct a 8×1 multiplexer with two 4×1 and one 2×1 multiplexers. ?