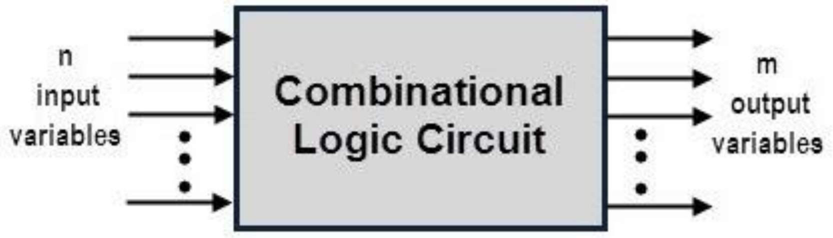
Combinational Circuit Design

Combinational Circuit



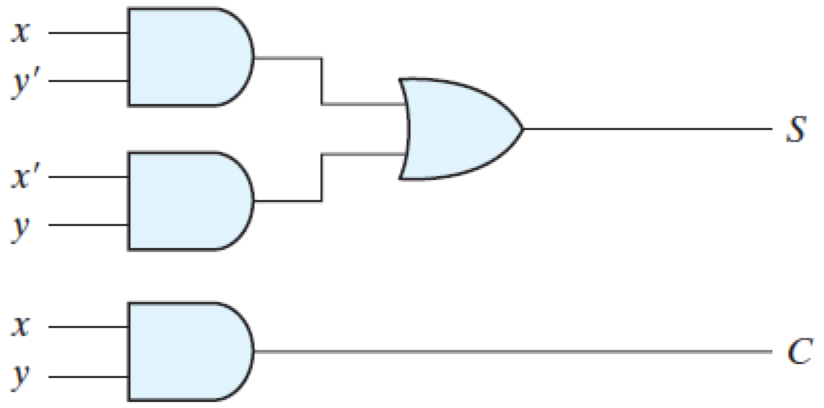
Design Procedure

1. From the specifications of the circuit, determine the required number of inputs and outputs and assign a symbol to each.
2. Derive the truth table that defines the required relationship between inputs and outputs.
3. Obtain the simplified Boolean functions for each output as a function of the input variables.
4. Draw the logic diagram and verify the correctness of the design

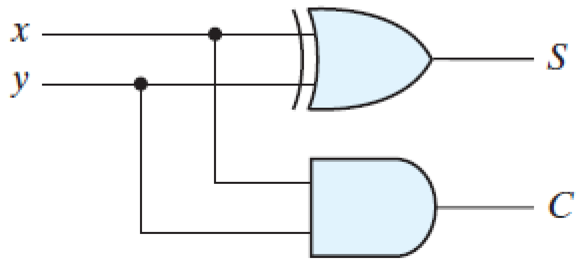
Binary Adder

- Half Adder

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

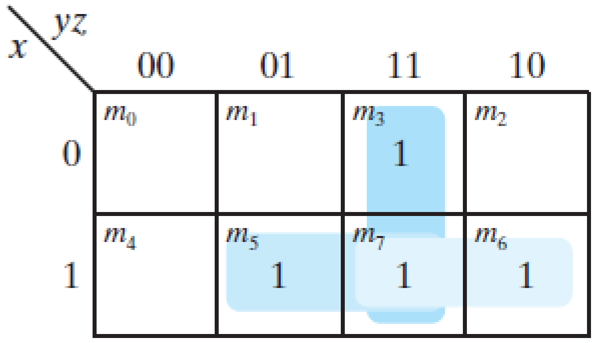


$$S = x'y + xy'$$
$$C = xy$$

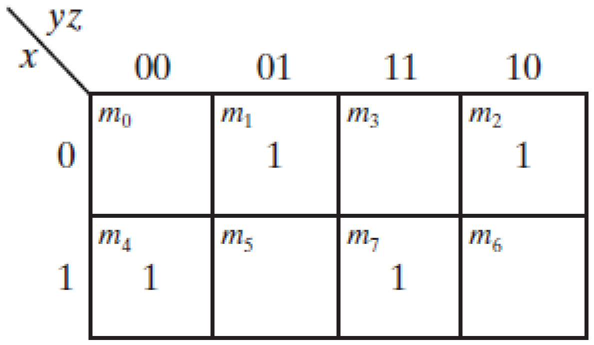


Full Adder

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

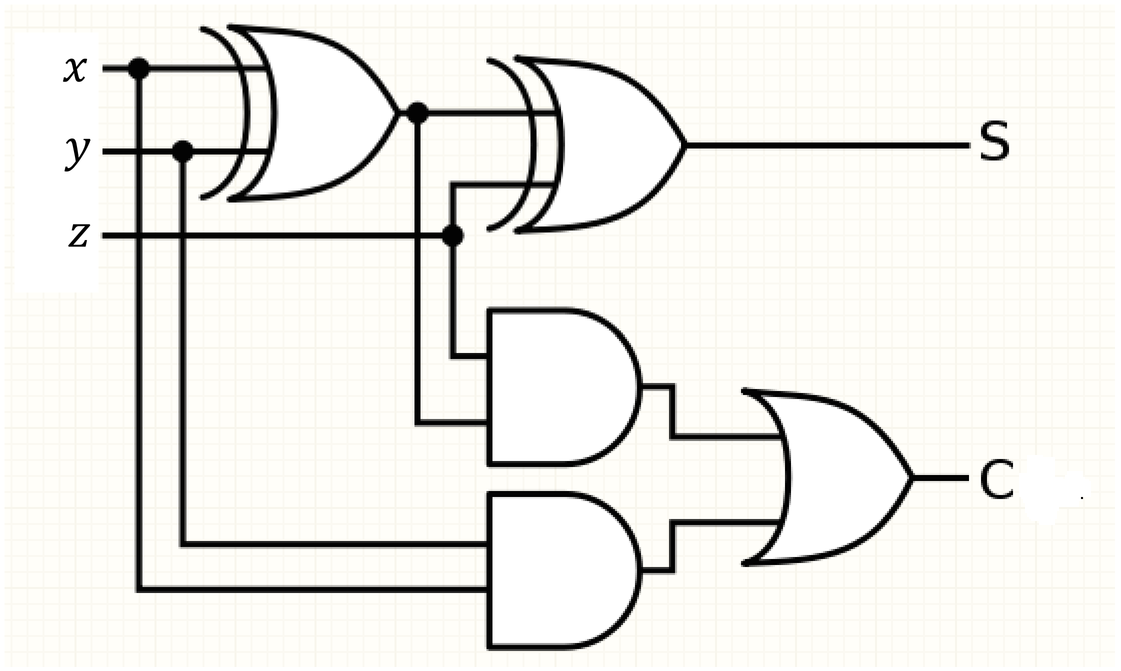


$C = xy + xz + yz$

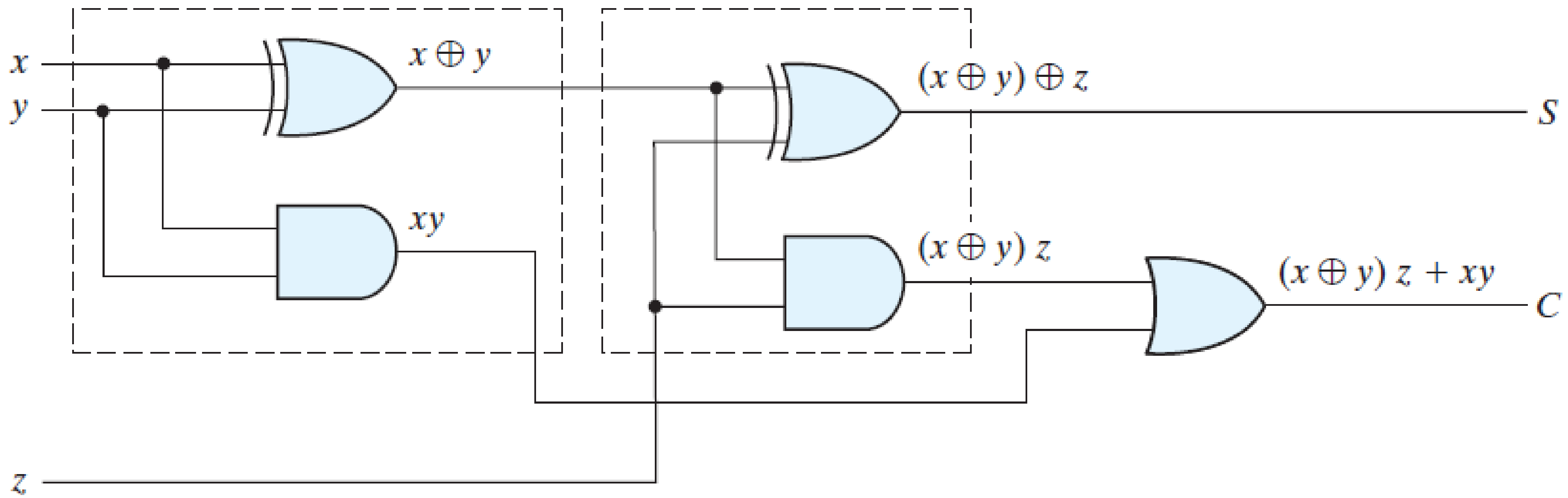


$$\begin{aligned} S &= z \oplus (x \oplus y) \\ &= z'(xy' + x'y) + z(xy' + x'y)' \\ &= z'(xy' + x'y) + z(xy + x'y') \\ &= xy'z' + x'yz' + xyz + x'y'z \end{aligned}$$

Full Adder



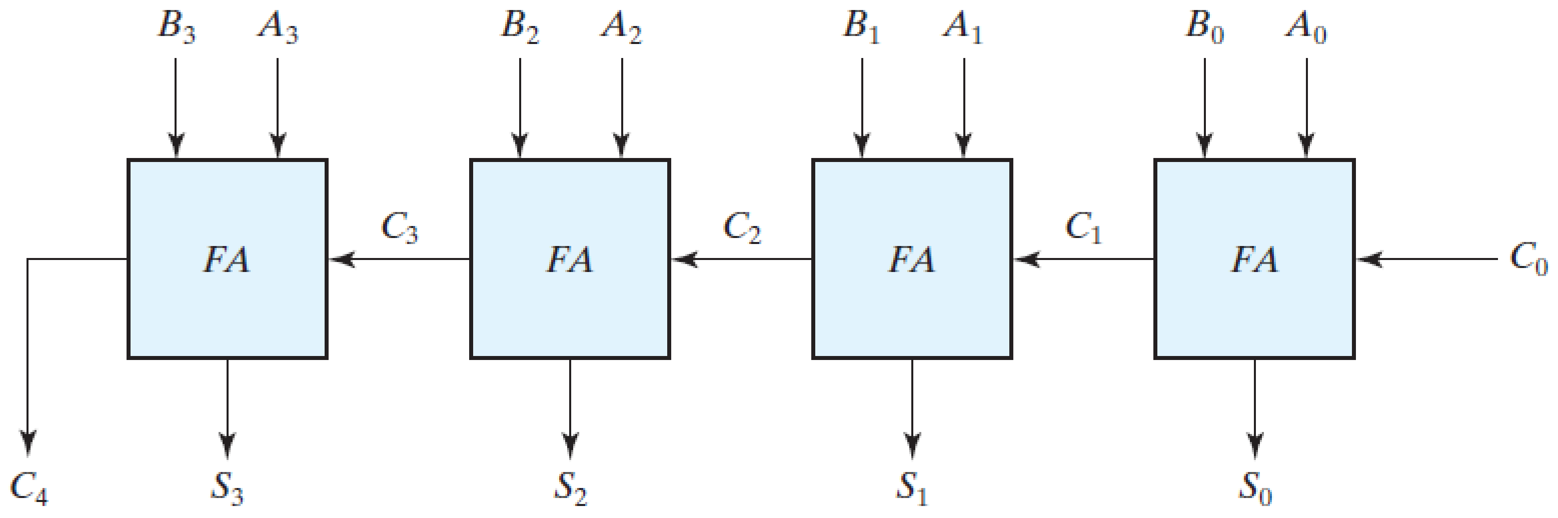
Full Adder using two Half Adder



Binary Adder

1011+0011

Subscript *i*:	3	2	1	0	
Input carry	0	1	1	0	C_i
Augend	1	0	1	1	A_i
Addend	0	0	1	1	B_i
Sum	1	1	1	0	S_i
Output carry	0	0	1	1	C_{i+1}



Propagation Delay

Carry Look ahead Adder (CLA)

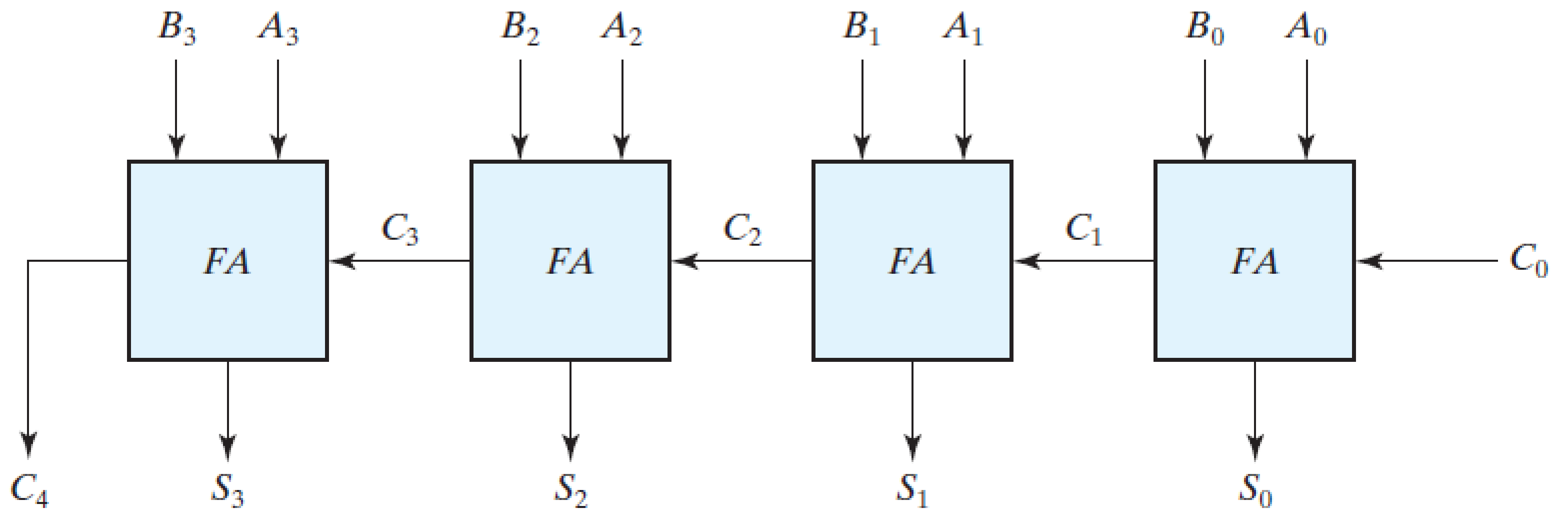
$$A \; - \; A_3\, A_2\, A_1\, A_0$$
$$B \; - \; B_3\, B_2\, B_1\, B_0$$
$$C_4 \;\; S_3\, S_2\, S_1\, S_0$$

$$C_o = AB + (A \oplus B)C_i$$

$$C_o = G + PC_i$$

A	**B**	**C_i**	**C_o**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

CLA



$$C_i = G_{i-1} + P_{i-1}C_{i-1}$$

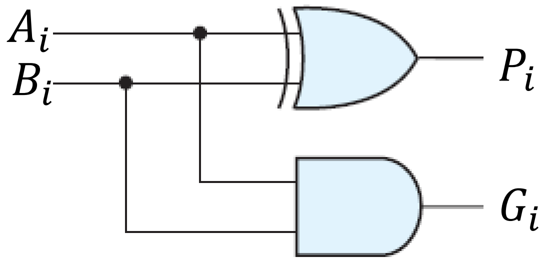
$$C_1 = G_0 + P_0C_0$$

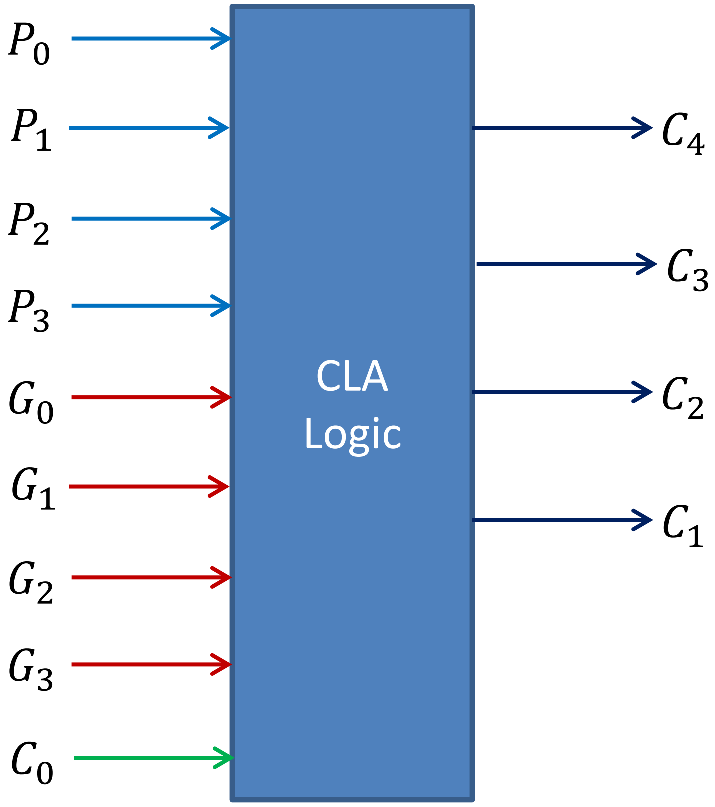
$$C_2 = G_1 + P_1C_1 = G_1 + P_1(G_0 + P_0C_0) = G_1 + P_1G_0 + P_0P_1C_0$$

$$C_3 = G_2 + P_2C_2 = G_2 + P_2(G_1 + P_1G_0 + P_0P_1C_0) = G_2 + P_2G_1 + P_1P_2G_0 + P_0P_1P_2C_0)$$

$$\begin{aligned} C_4 = G_3 + P_3C_3 &= G_3 + P_3(G_2 + P_2G_1 + P_1P_2G_0 + P_0P_1P_2C_0) \\ &= G_3 + P_3G_2 + P_2P_3G_1 + P_1P_2P_3G_0 + P_0P_1P_2P_3C_0) \end{aligned}$$

CLA Logic

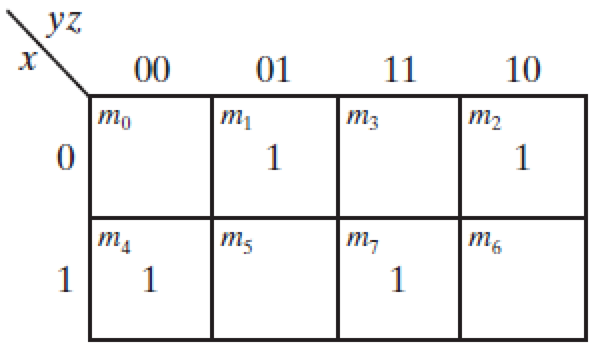




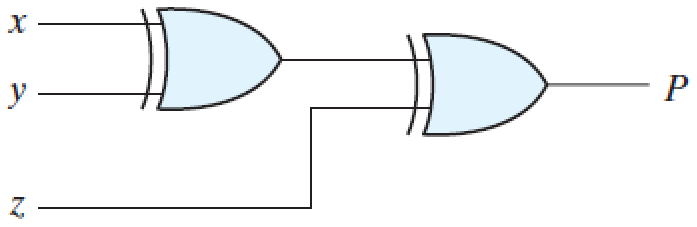
Parity Generation and Checking

Even parity Generator

Three-Bit Message			Parity Bit
x	***y***	***z***	***P***
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



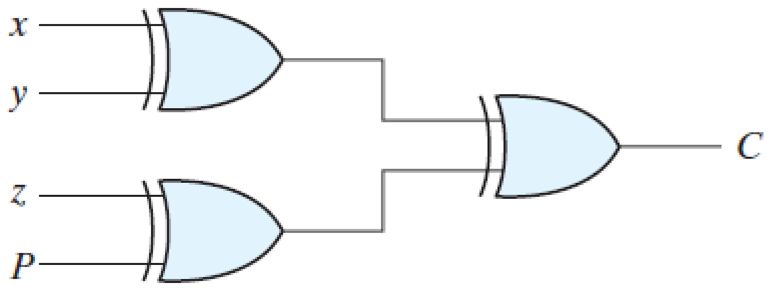
$$P = x \oplus y \oplus z$$



Parity Checking

Four Bits Received				Parity Error Check
x	*y*	*z*	*P*	*C*
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

$$C = x \oplus y \oplus z \oplus P$$



Code Converters

- Convenience
- Security reasons
- Hardware reasons

Examples

- ASCII code
- BCD (Binary coded decimal)
- Excess 3 code
- Gray code

BCD to Excess 3 Converter

b_3	b_2	b_1	b_0	e_3	e_2	e_1	e_0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

$$e_3 = b_3 + b_2 b_1 + b_2 b_0$$

$$e_2 = \bar{b}_2 b_1 + \bar{b}_2 b_0 + b_2 \bar{b}_1 \bar{b}_0$$

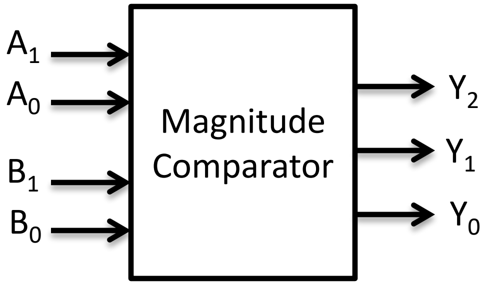
$$e_1 = \bar{b}_1 \bar{b}_0 + b_1 b_0 = b_1 \odot b_0$$

$$e_0 = \bar{b}_0$$

Binary to Gray code

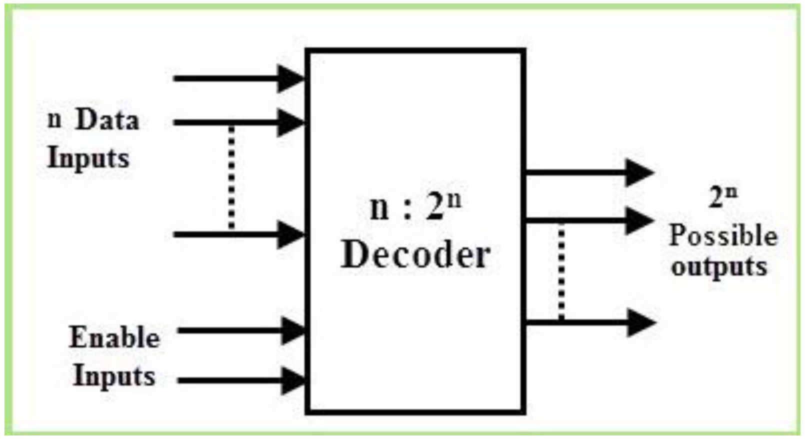
Binary				Gray Code			
b_3	b_2	b_1	b_0	g_3	g_2	g_1	g_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Magnitude Comparator



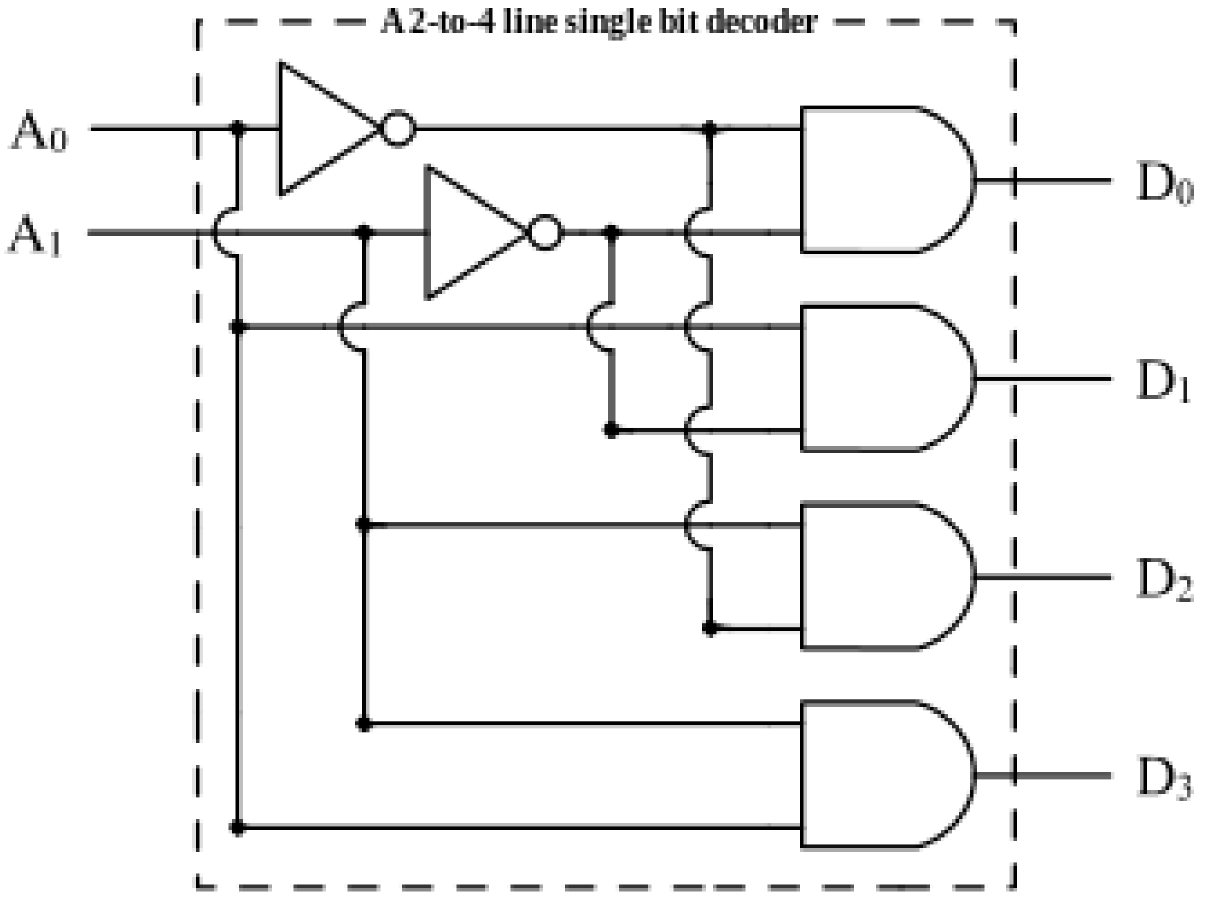
Inputs				Outputs		
A_1	A_0	B_1	B_0	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

Decoder



Eg. 3 to 8 line decoder

2 to 4 line Decoder



Truth Table

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

Minterm Equations

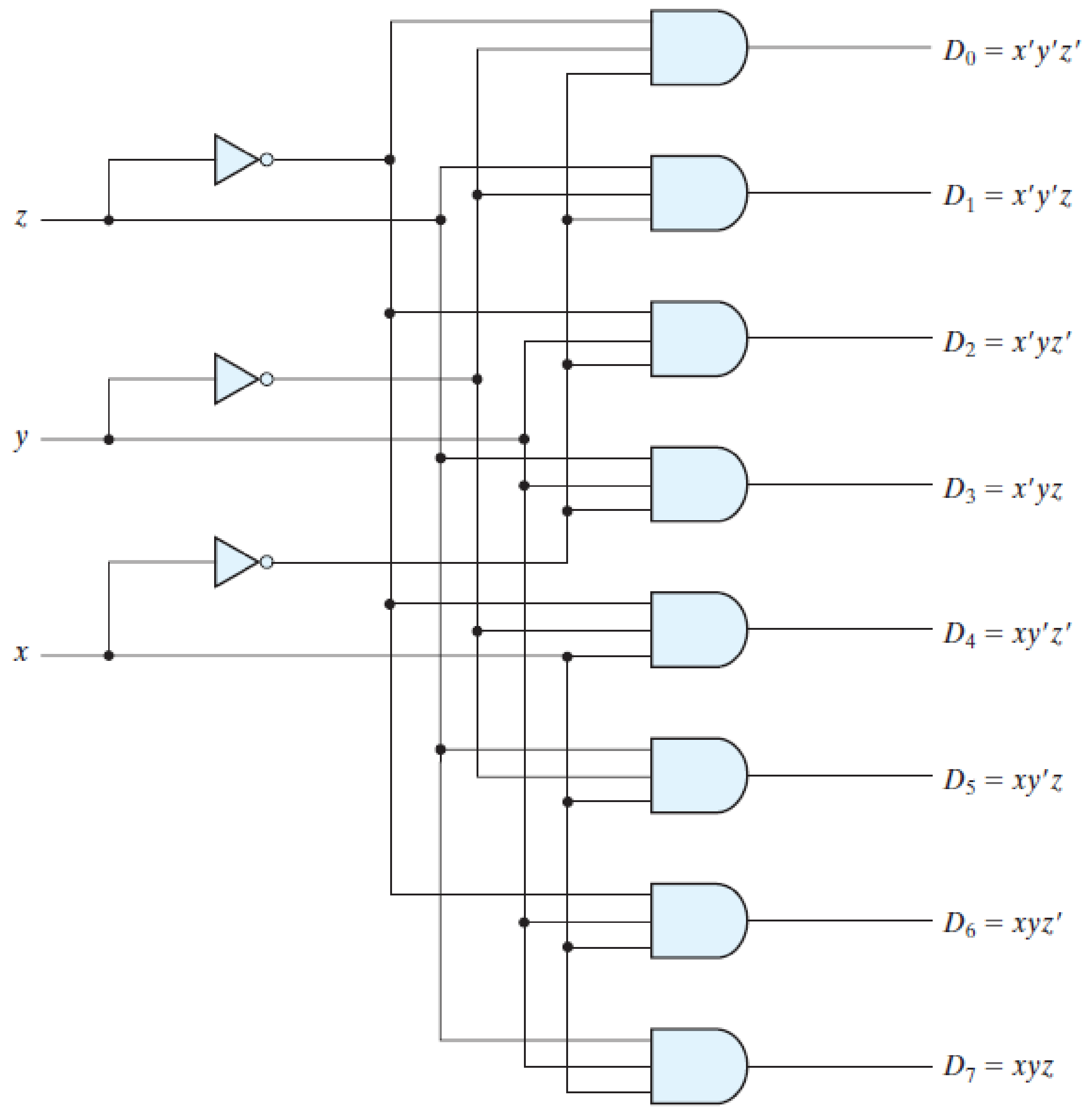
$$D_0 = \overline{A_1} \cdot \overline{A_0}$$

$$D_1 = \overline{A_1} \cdot A_0$$

$$D_2 = A_1 \cdot \overline{A_0}$$

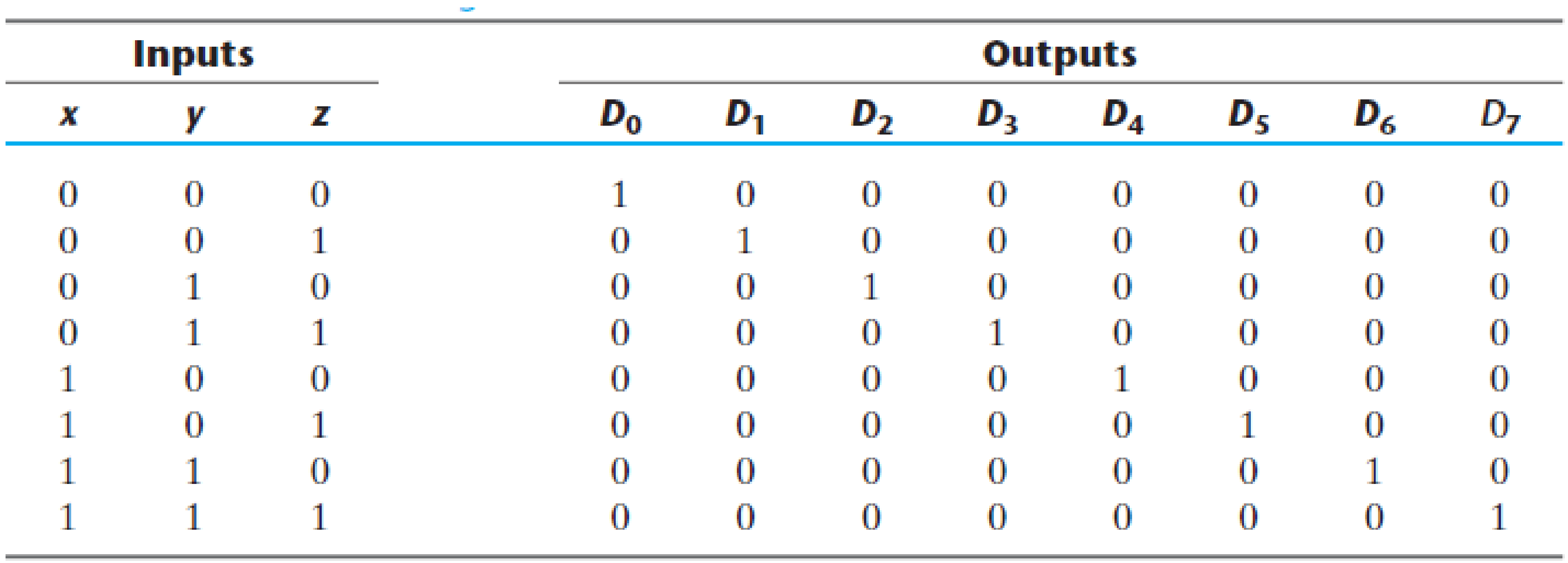
$$D_3 = A_1 \cdot A_0$$

3 to 8 line Decoder



3 to 8 line Decoder

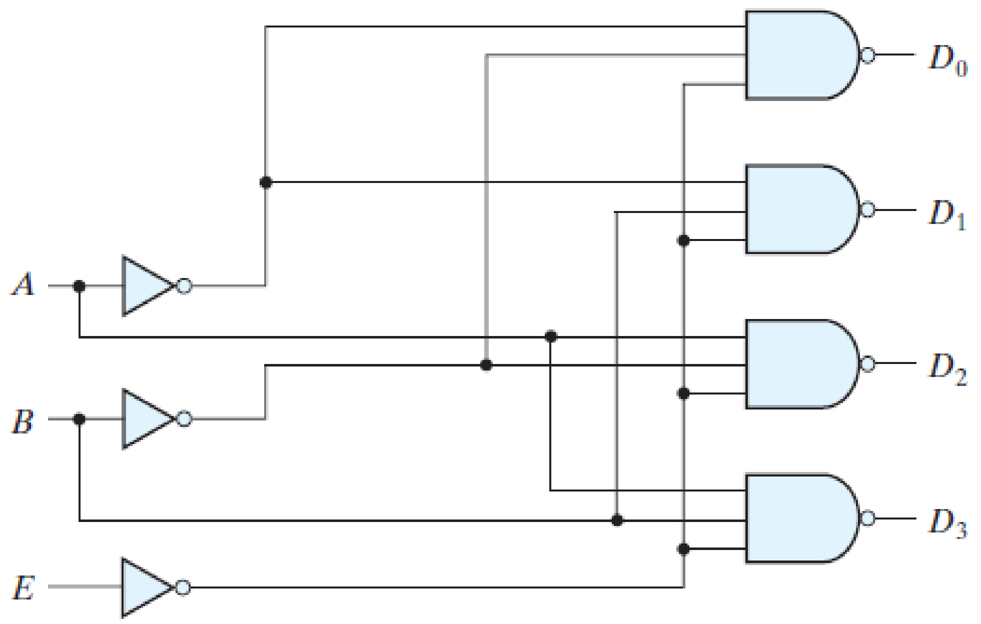
Truth Table



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

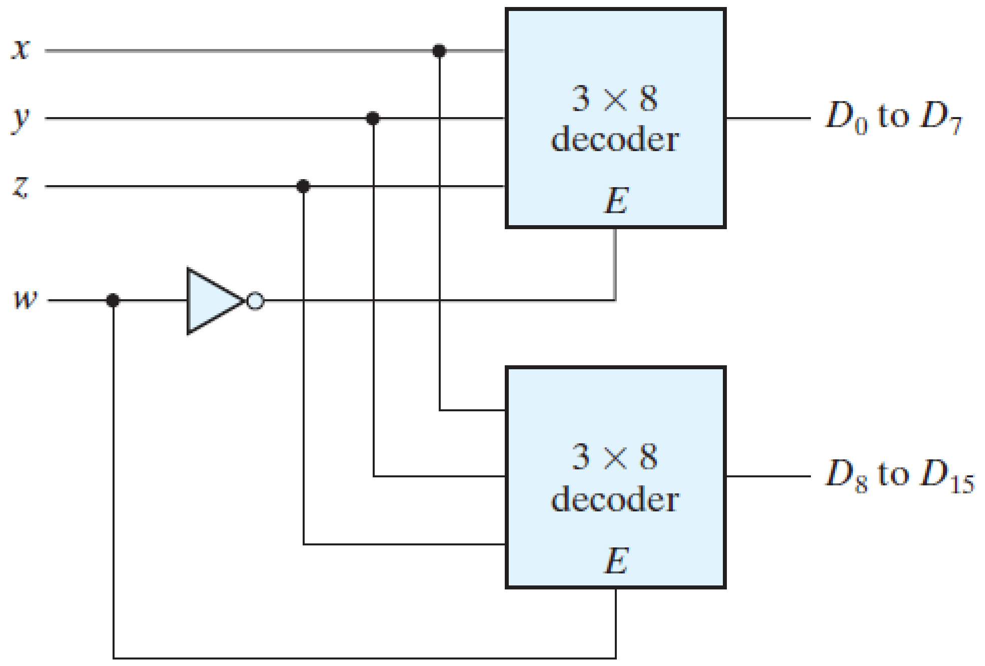
Decoder with Enable

- NAND gate based



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

Connecting Decoders



When $w = 0$, top decoder is enabled, bottom decoder is disabled.
Generates output for input 0000 to 0111

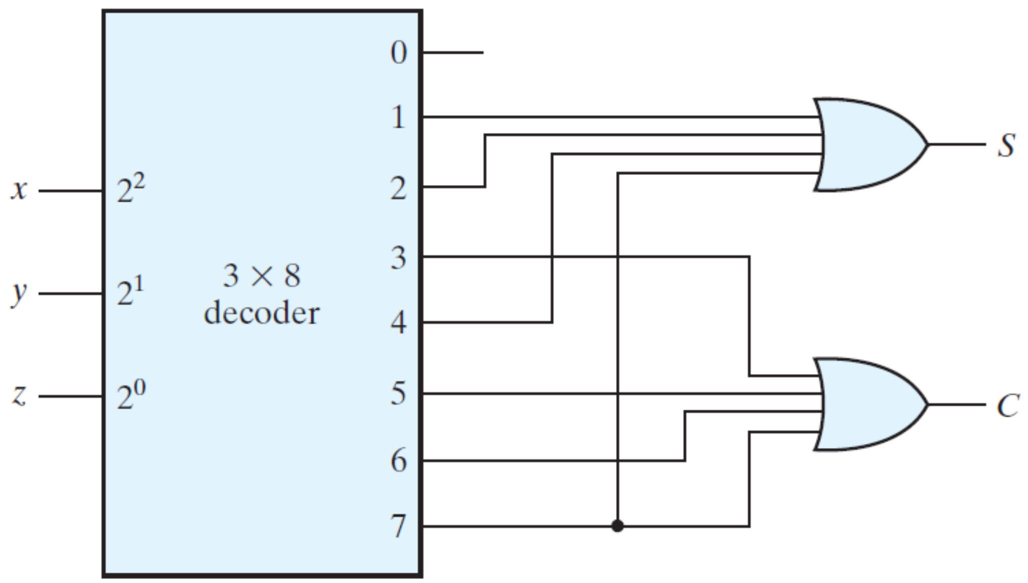
When $w = 1$, top decoder is disabled, bottom decoder is enabled.
Generates output for input 1000 to 1111

Combinational Logic Implementation using Decoder

- Full Adder using 3 x 8 Decoder

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

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

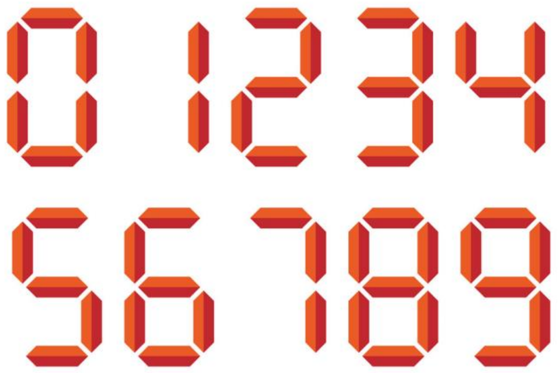
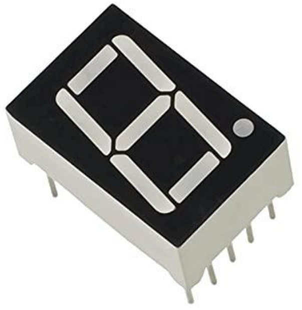


BCD to 7 segment Decoder

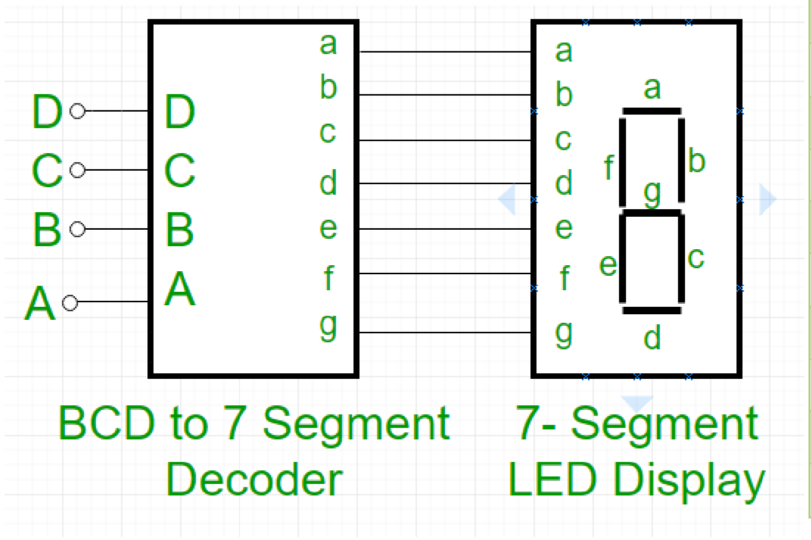
- BCD – Binary coded Decimal

Decimal	Binay (BCD) 8 4 2 1
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

10 – 00010000
99 – 10011001
100 - 000100000000

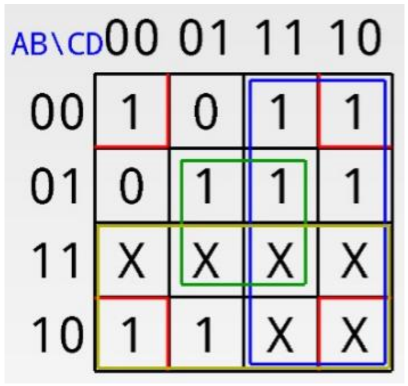



BCD to 7 segment Decoder



A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

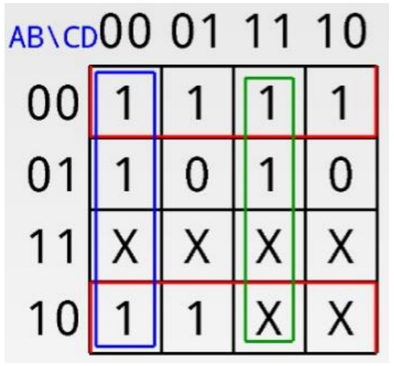
K Map for 'a'



AB\CD	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

$$a = A + C + AD + \bar{A}\bar{D}$$

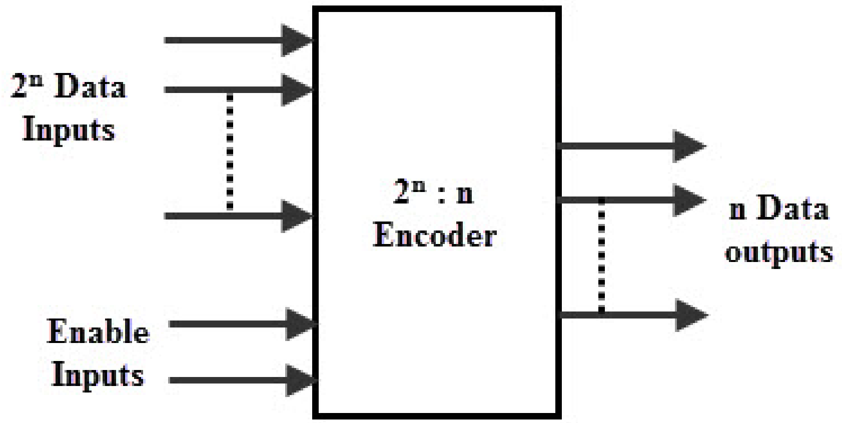
K Map for 'b'



AB\CD	00	01	11	10
00	1	1	1	1
01	1	0	1	0
11	X	X	X	X
10	1	1	X	X

$$b = \bar{B} + \bar{C}\bar{D} + CD$$

Encoders



Eg. 8 to 3 line encoder

8 to 3 Line Encoder (Octal to Binary Encoder)

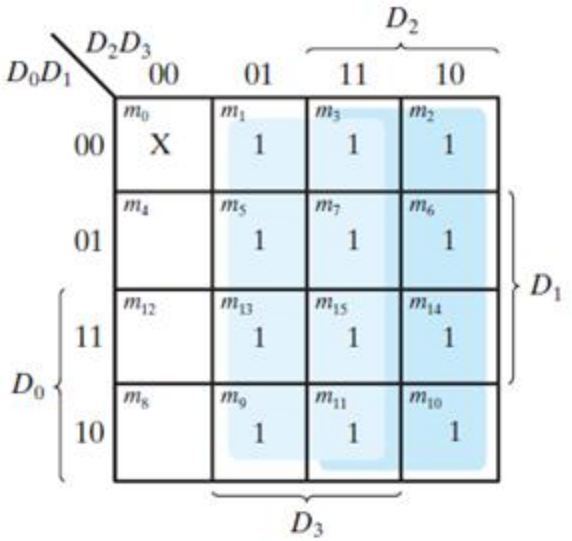
Truth Table of an Octal-to-Binary Encoder

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

$$z = D_1 + D_3 + D_5 + D_7$$
$$y = D_2 + D_3 + D_6 + D_7$$
$$x = D_4 + D_5 + D_6 + D_7$$

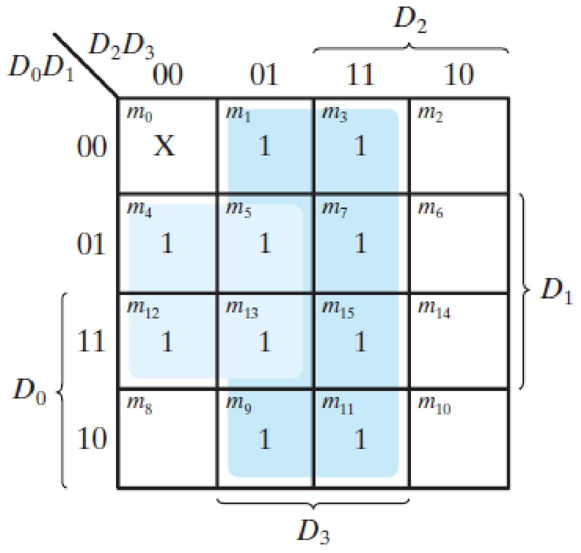
Priority Encoder

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



D_2D_3
D_0D_1
00 01 11 10
m_0 X, m_1 1, m_3 1, m_2 1
m_4, m_5 1, m_7 1, m_6 1
m_{12}, m_{13} 1, m_{15} 1, m_{14} 1
m_8, m_9 1, m_{11} 1, m_{10} 1
D_2
D_1
D_0
D_3

$x = D_2 + D_3$

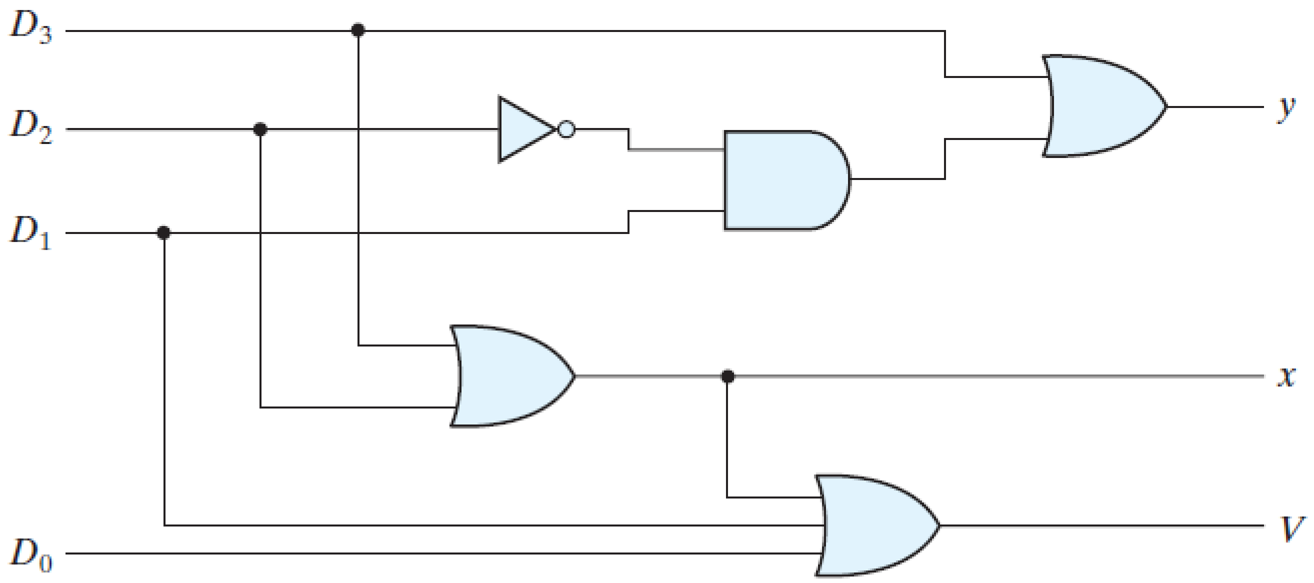


D_2D_3
D_0D_1
00 01 11 10
m_0 X, m_1 1, m_3 1, m_2
m_4 1, m_5 1, m_7 1, m_6
m_{12} 1, m_{13} 1, m_{15} 1, m_{14}
m_8, m_9 1, m_{11} 1, m_{10}
D_2
D_1
D_0
D_3

$y = D_3 + D_1D'_2$

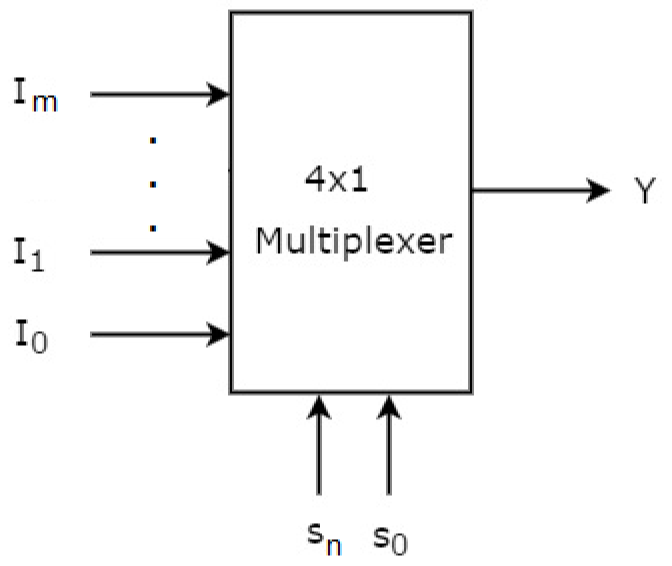
$$x = D_2 + D_3$$
$$y = D_3 + D_1 D'_2$$
$$V = D_0 + D_1 + D_2 + D_3$$

4 to 2 line priority Encoder

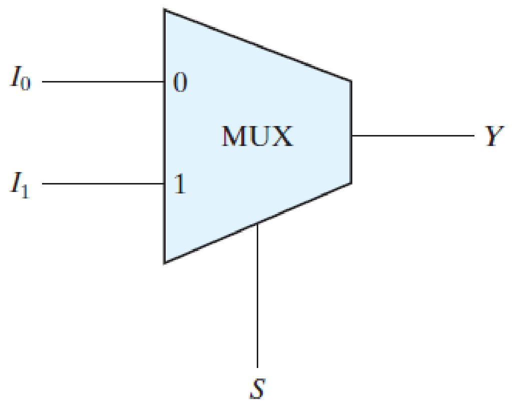


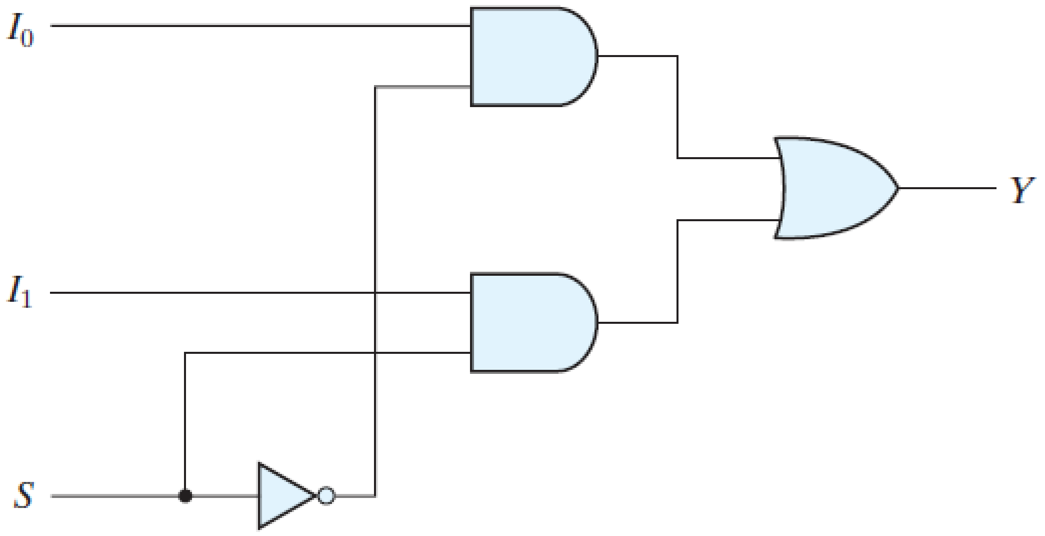
Multiplexers

- selects binary information from one of many input lines and directs it to a single output line



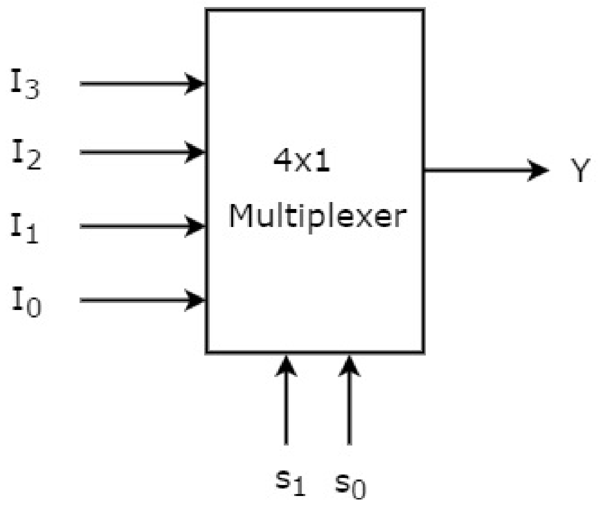
2 X 1 Multiplexer



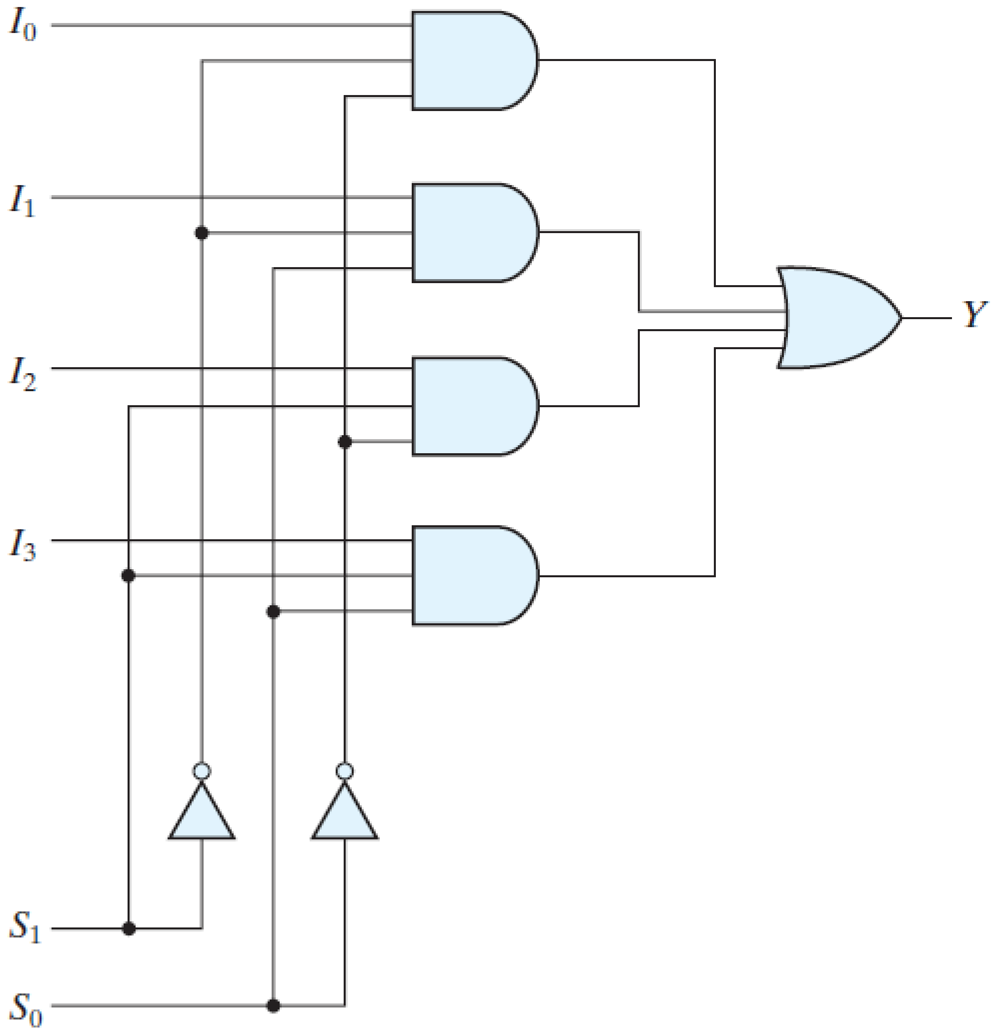


S	Y
0	I_0
1	I_1

4 X 1 Multiplexer



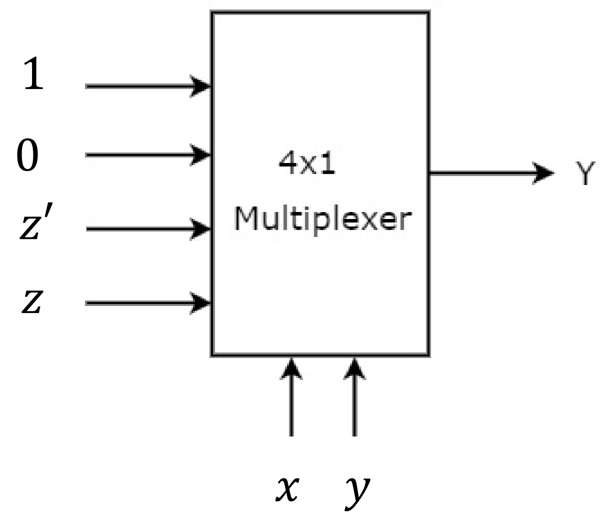
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



Boolean Function Implementation

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

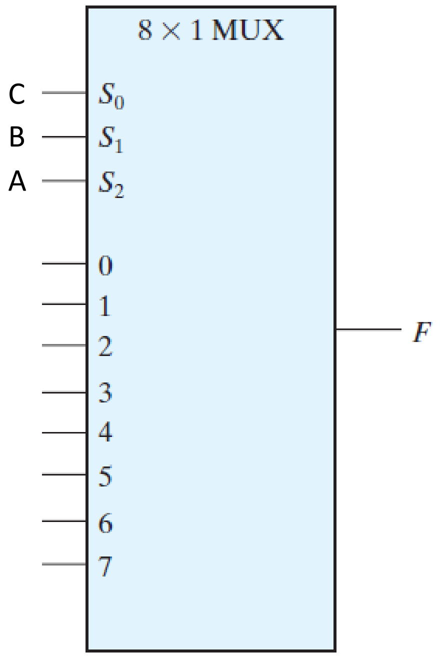
x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



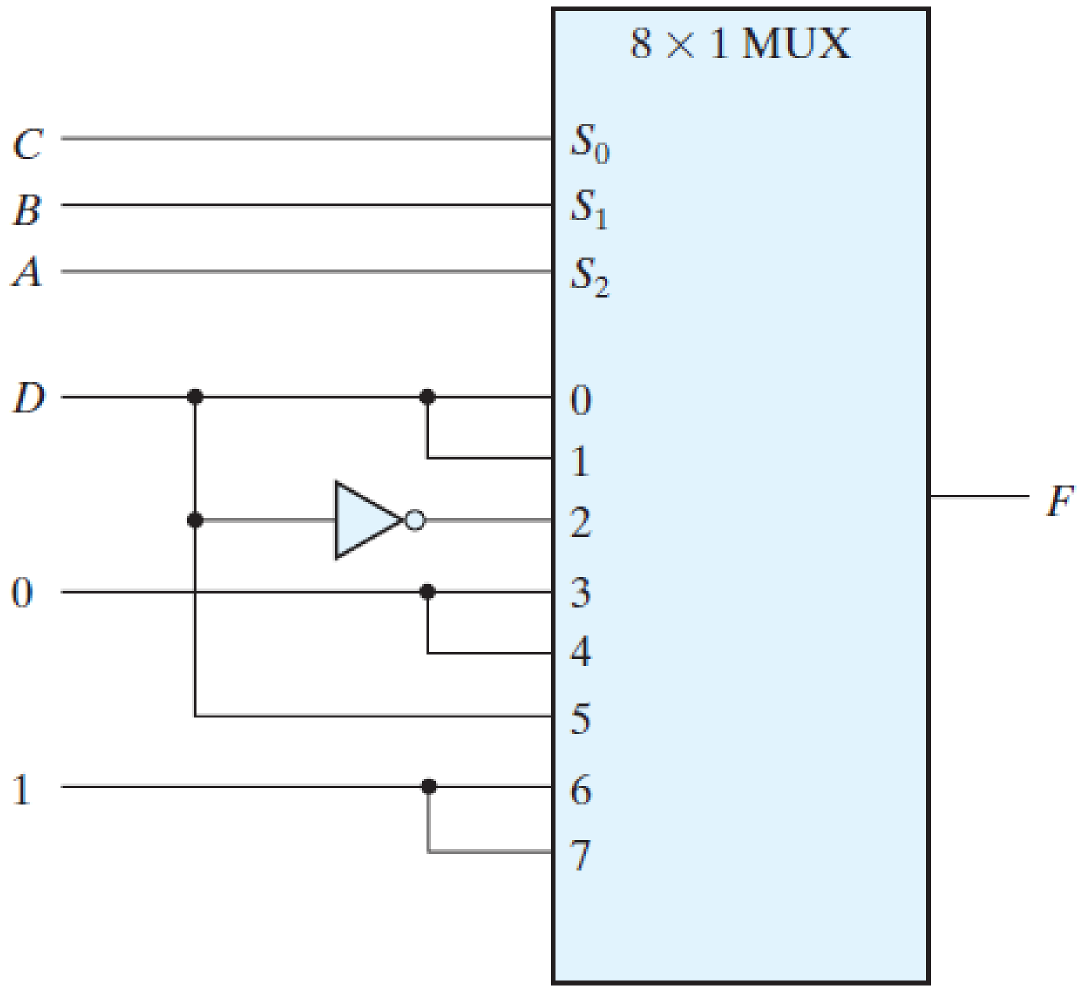
xy \ z	0	1	
00	0	1	$I_0 = z$
01	1	0	$I_1 = z'$
11	1	1	$I_3 = 1$
10	0	0	$I_2 = 0$

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

AB \ CD	00	01	11	10
00	m_0	m_1 1	m_3 1	m_2
01	m_4 1	m_5	m_7	m_6
11	m_{12} 1	m_{13} 1	m_{15} 1	m_{14} 1
10	m_8	m_9	m_{11} 1	m_{10}

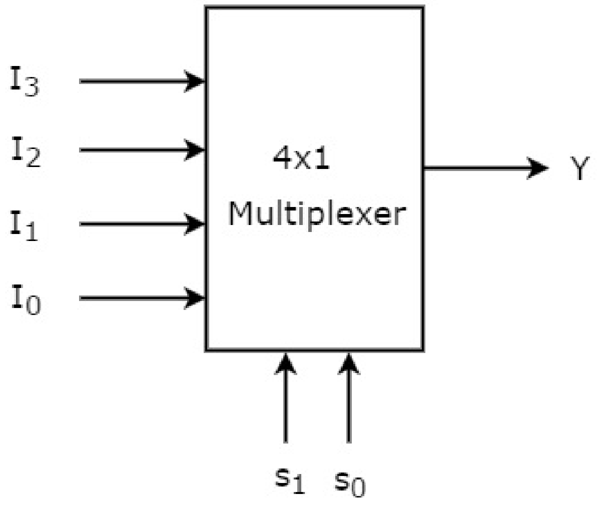


A	B	C	D	F	
0	0	0	0	0	$F = D$
0	0	0	1	1	
0	0	1	0	0	$F = D$
0	0	1	1	1	
0	1	0	0	1	$F = D'$
0	1	0	1	0	
0	1	1	0	0	$F = 0$
0	1	1	1	0	
1	0	0	0	0	$F = 0$
1	0	0	1	0	
1	0	1	0	0	$F = D$
1	0	1	1	1	
1	1	0	0	1	$F = 1$
1	1	0	1	1	
1	1	1	0	1	$F = 1$
1	1	1	1	1	

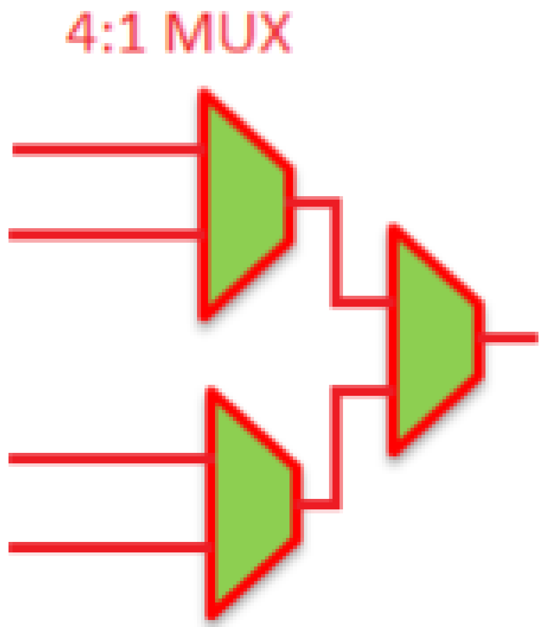

8 × 1 MUX
C
B
A
S_0
S_1
S_2
D
0
1
0
1
2
3
4
5
6
7
F

$$F(w,x,y,z) = \sum(1,4,5,7,9,12,13)$$

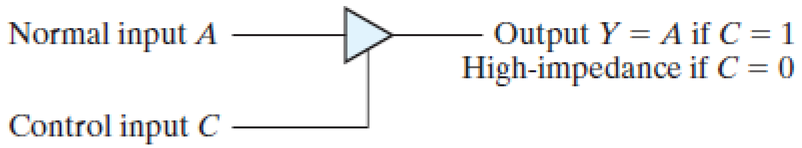
$wx \backslash yz$	00	01	11	10
00	m_0	m_1 1	m_3	m_2
01	m_4 1	m_5 1	m_7 1	m_6
11	m_{12} 1	m_{13} 1	m_{15}	m_{14}
10	m_8	m_9 1	m_{11}	m_{10}

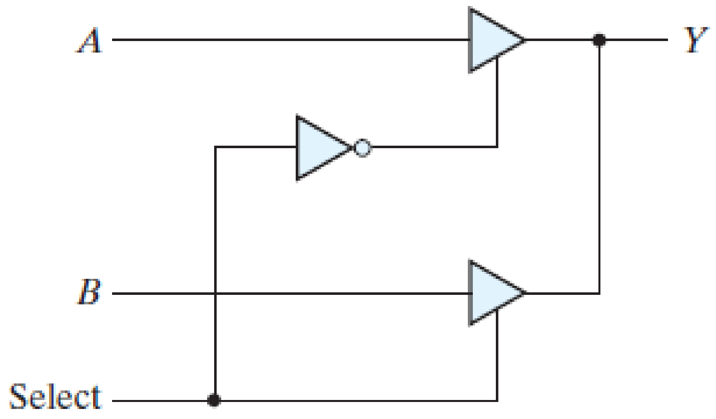


Mux Trees



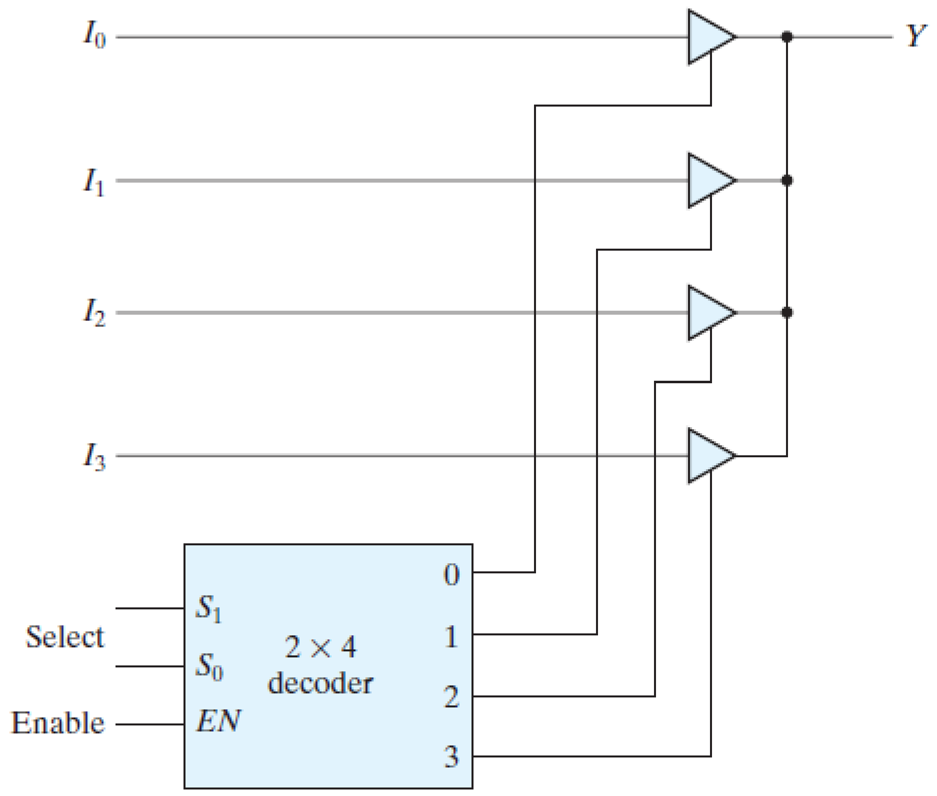
Three State Gates





2X1 Mux using Three state gates

Three State Gates



4X1 Mux using Three state gates