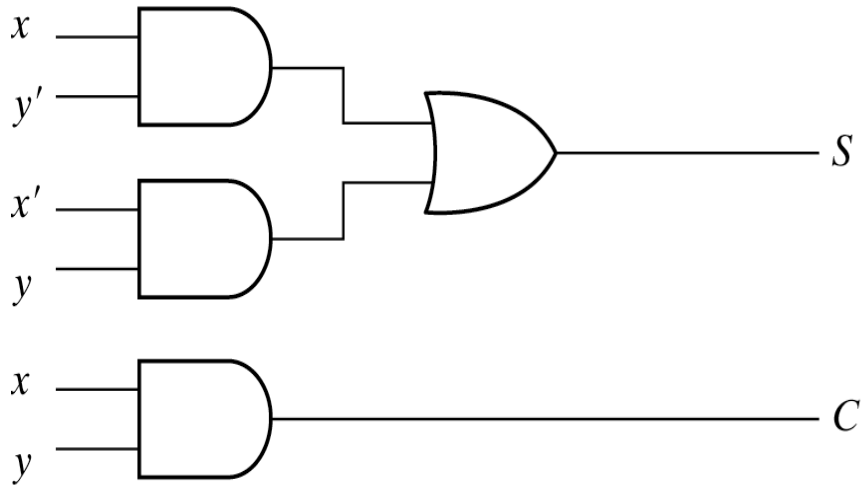
Combinational circuits

Binary Adder

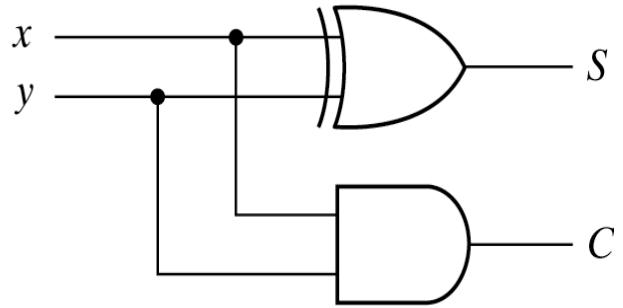
Table 4-3
Half Adder

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



(a) $S = xy' + x'y$
$C = xy$



(b) $S = x \oplus y$
$C = xy$

Fig. 4-5 Implementation of Half-Adder

Binary Adder

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

S = x'y'z + x'yz' + xy'z' + xyz

C = xy + xz + yz

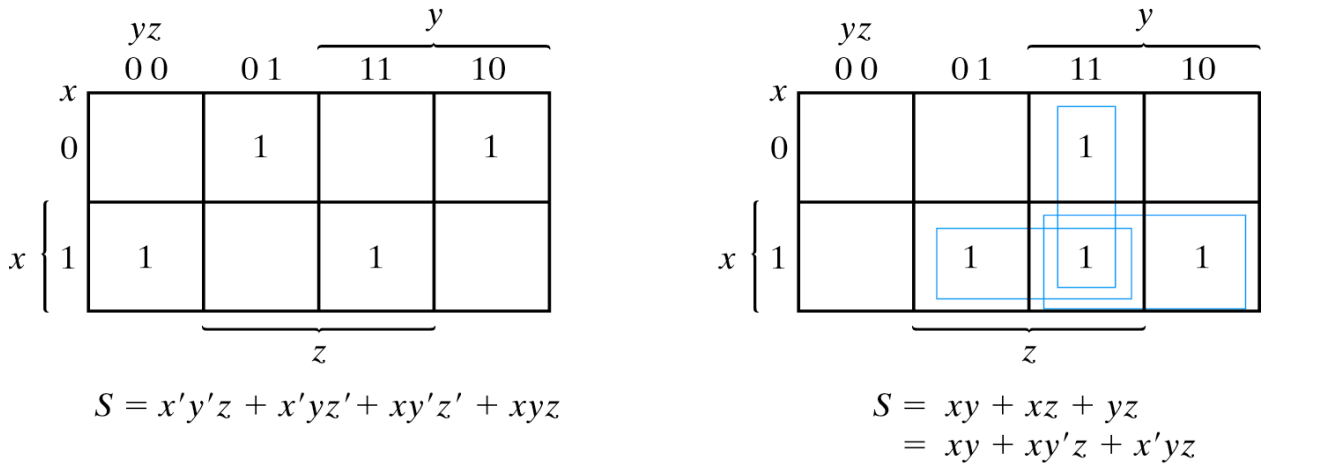


Fig. 4-6 Maps for Full Adder

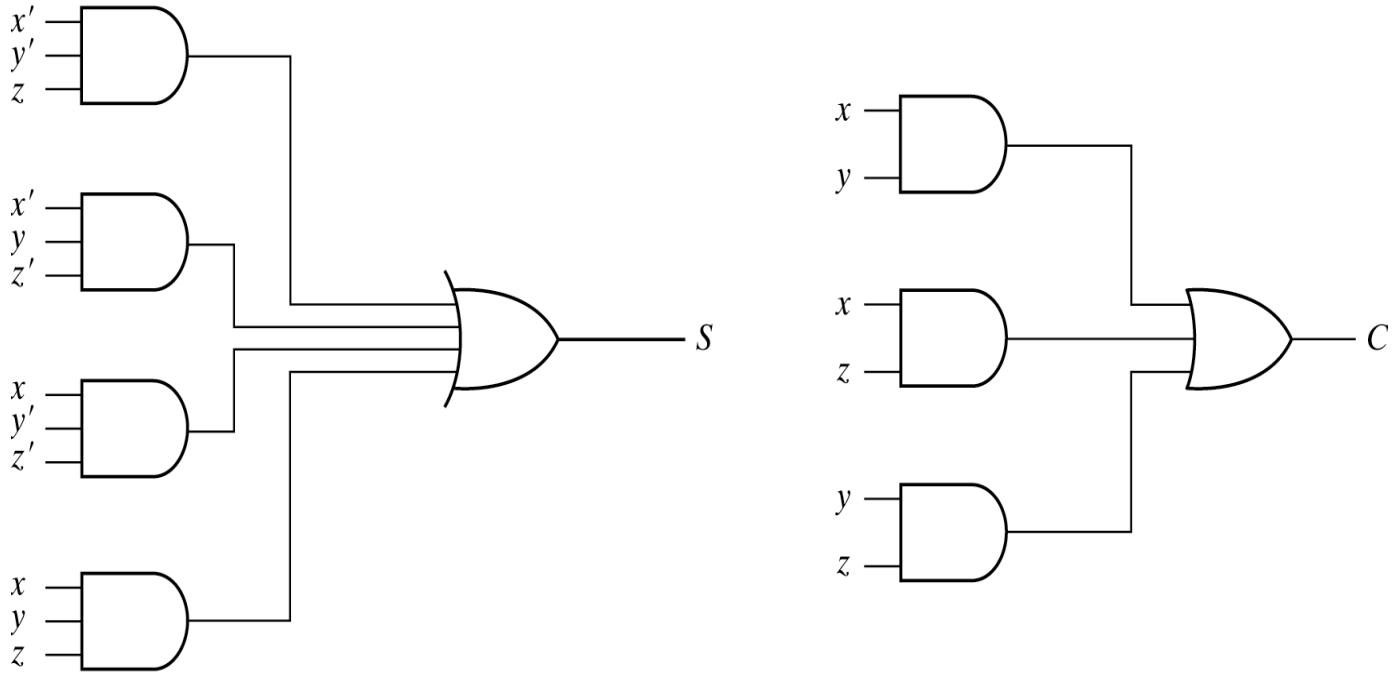


Fig. 4-7 Implementation of Full Adder in Sum of Products

Binary Adder

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

S = x'y'z + x'yz' + xy'z' + xyz
C = xy'z + x'yz + xy

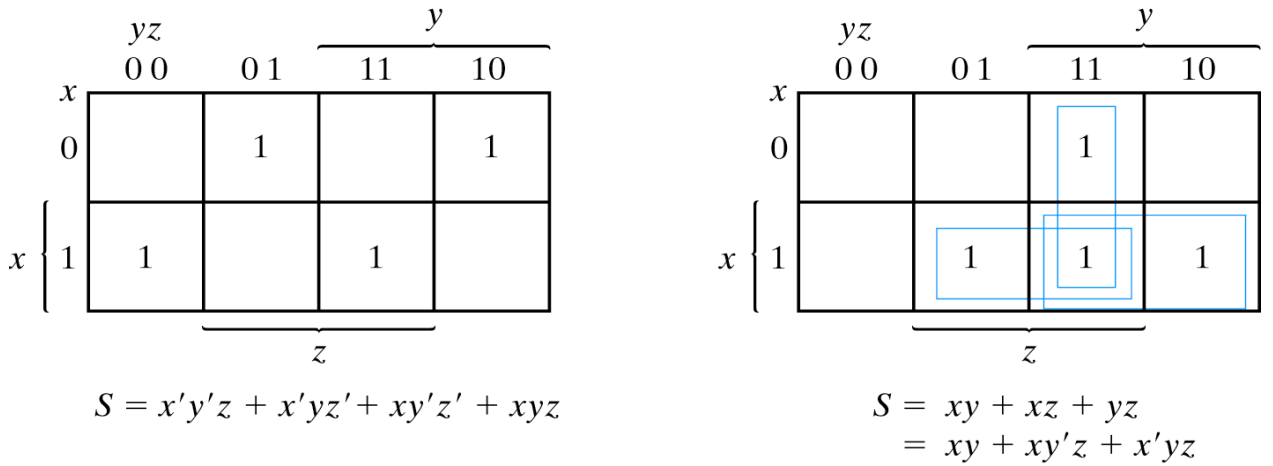


Fig. 4-6 Maps for Full Adder

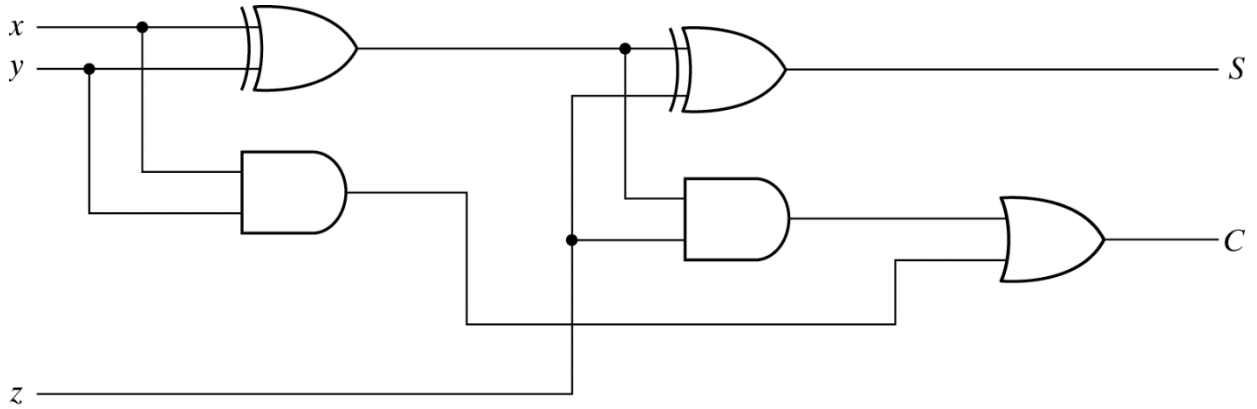


Fig. 4-8 Implementation of Full Adder with Two Half Adders and an OR Gate

Parallel Adder

Ripple Carry Adder

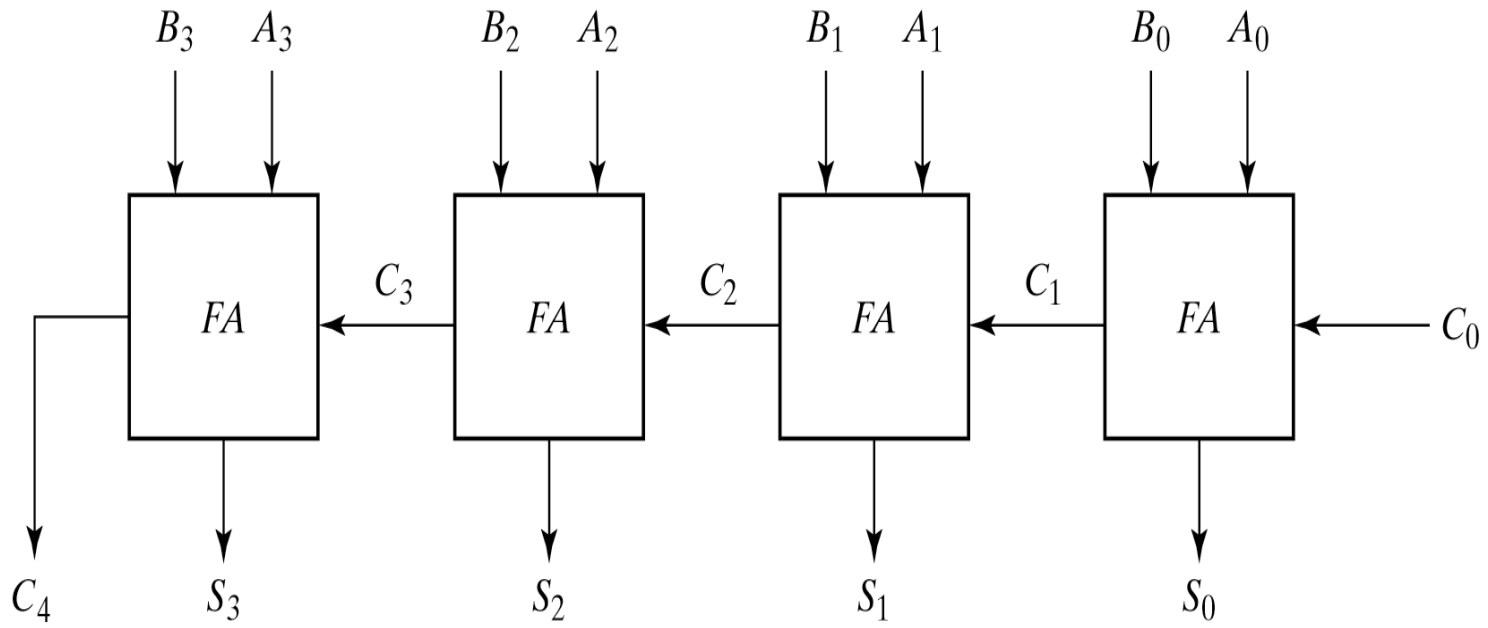


Fig. 4-9 4-Bit Adder

Carry Look-ahead Generator

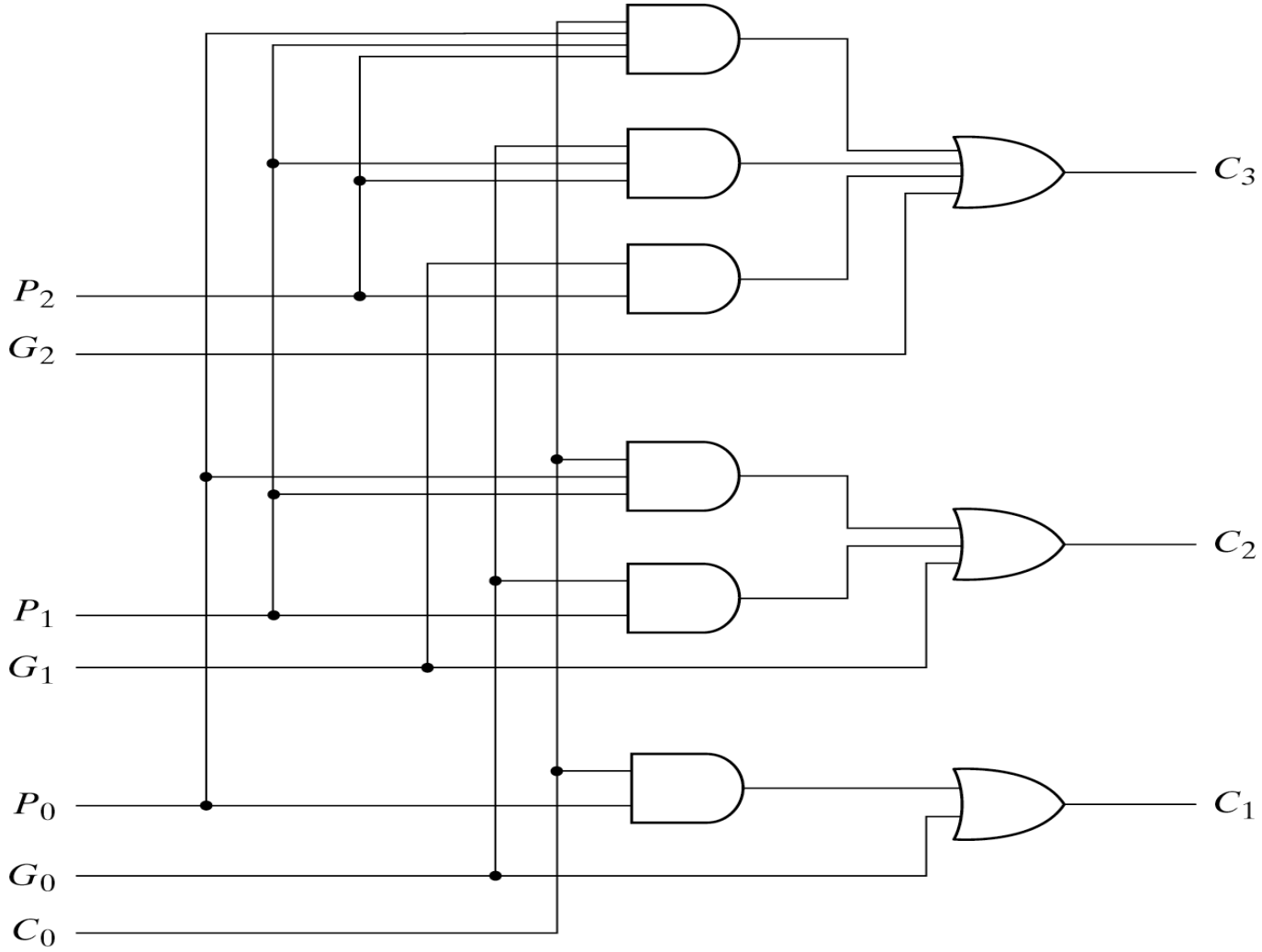


Fig. 4-11 Logic Diagram of Carry Lookahead Generator

Magnitude comparator

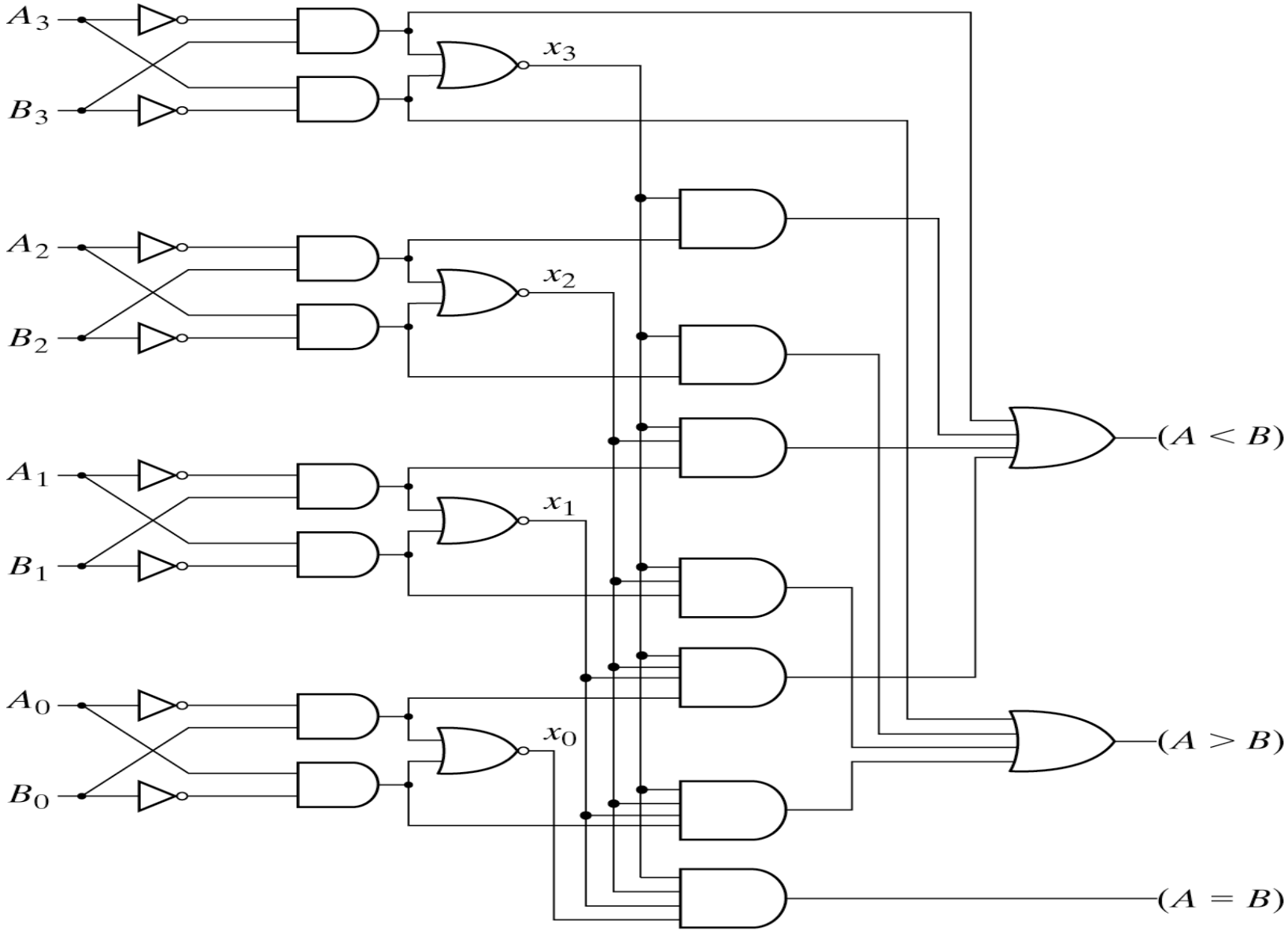


Fig. 4-17 4-Bit Magnitude Comparator

Decoders

- The decoder is called n-to-m-line decoder, where $m \leq 2^n$.
- the decoder is also used in conjunction with other code converters such as a BCD-to-seven-segment decoder.
- 3-to-8 line decoder: For each possible input combination, there are seven outputs that are equal to 0 and only one that is equal to 1.

Implementation and truth table

Table 4-6
Truth Table of a 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

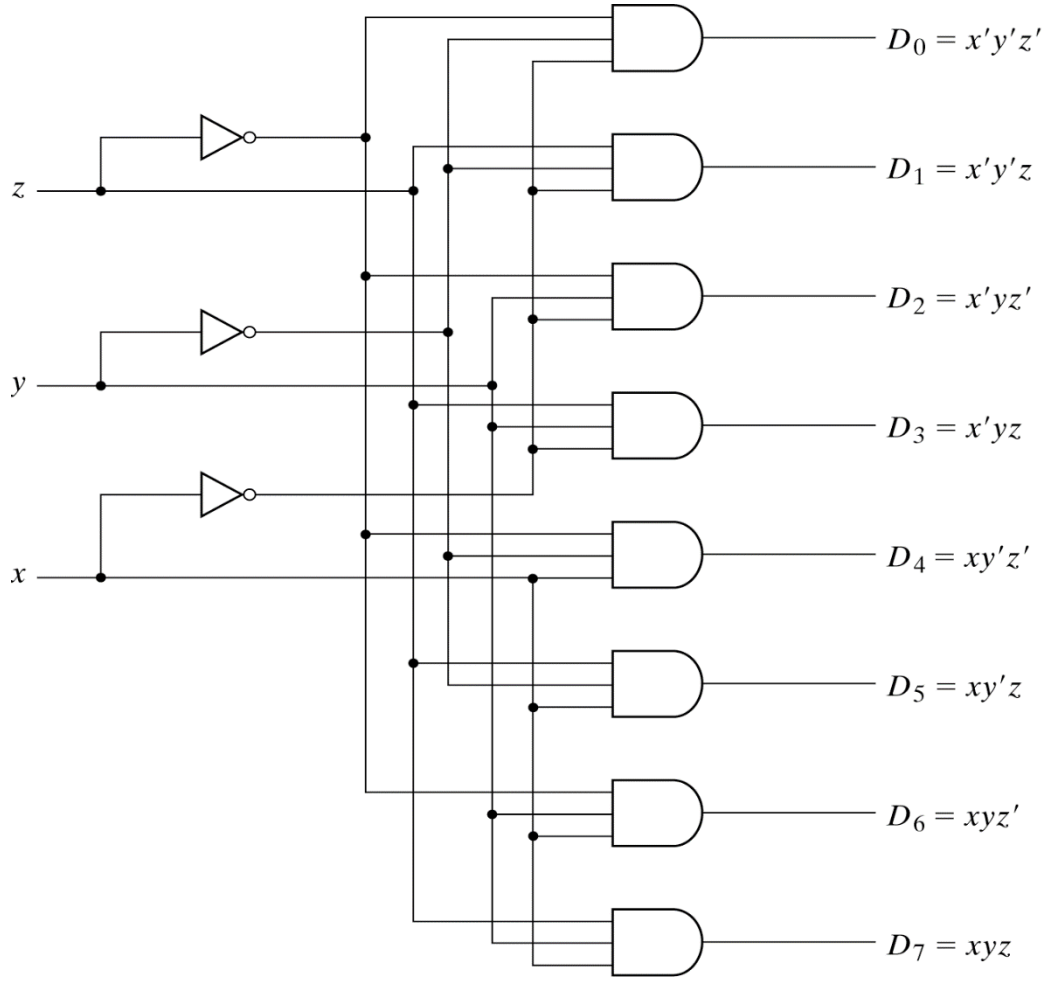


Fig. 4-18 3-to-8-Line Decoder

Encoder

Table 4-7
Truth Table of Octal-to-Binary Encoder

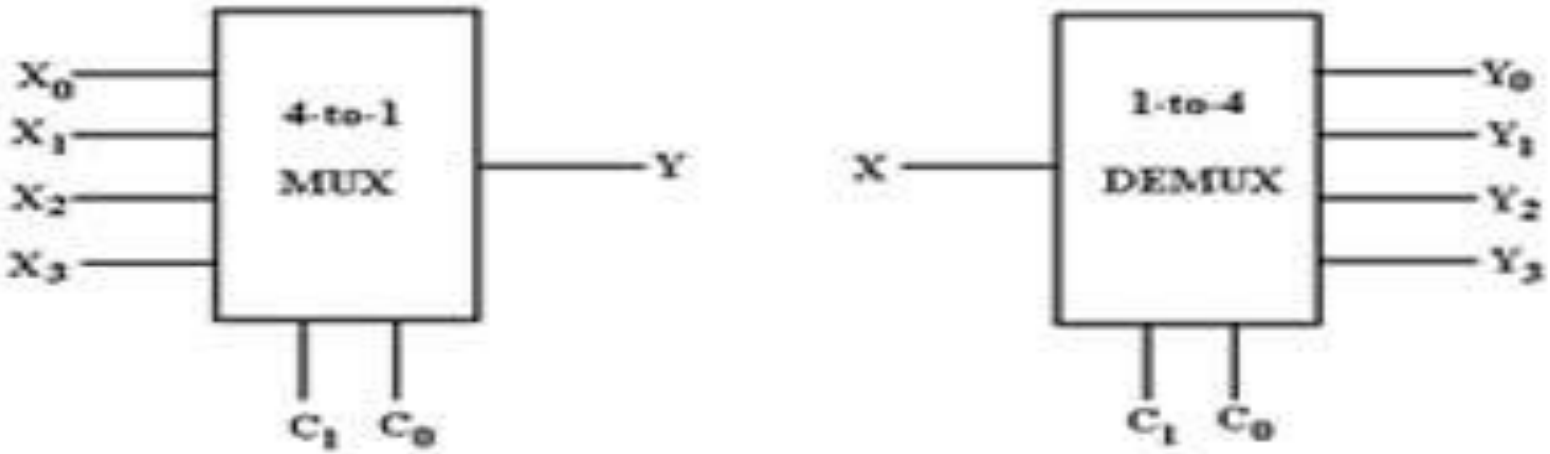
Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$z = D_1 + D_3 + D_5 + D_7$

$y = D_2 + D_3 + D_6 + D_7$

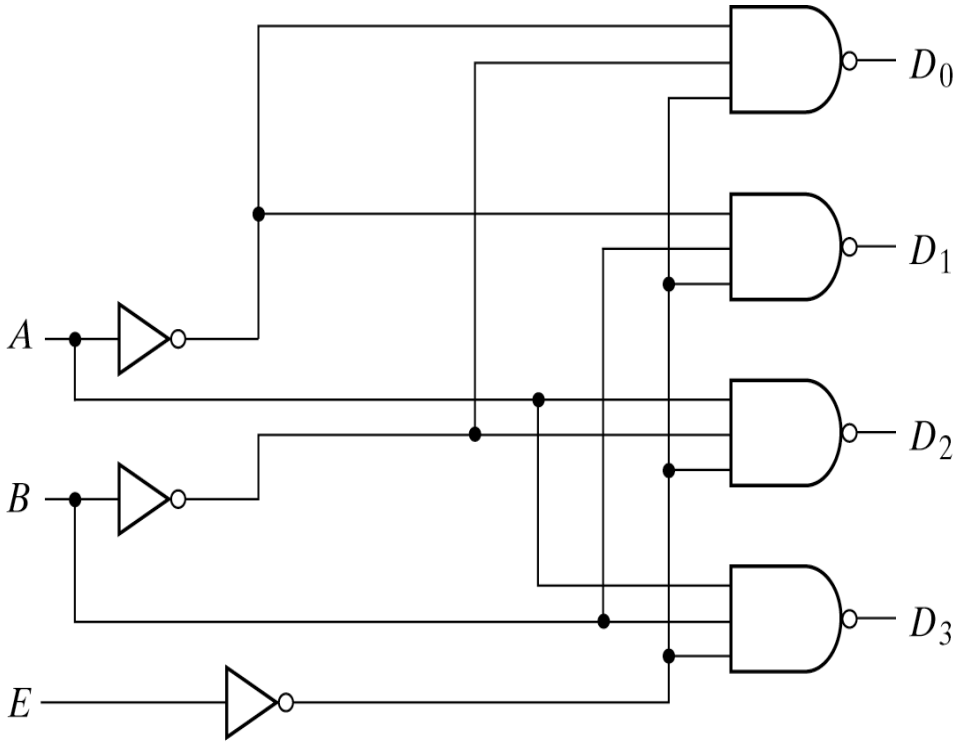
$x = D_4 + D_5 + D_6 + D_7$

Multiplexer and Demultiplexer



Decoder with enable input

A demultiplexer is a circuit that receives information on a single line and transmits this information on one of 2^n possible output lines.



(a) Logic diagram

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

Fig. 4-19 2-to-4-Line Decoder with Enable Input

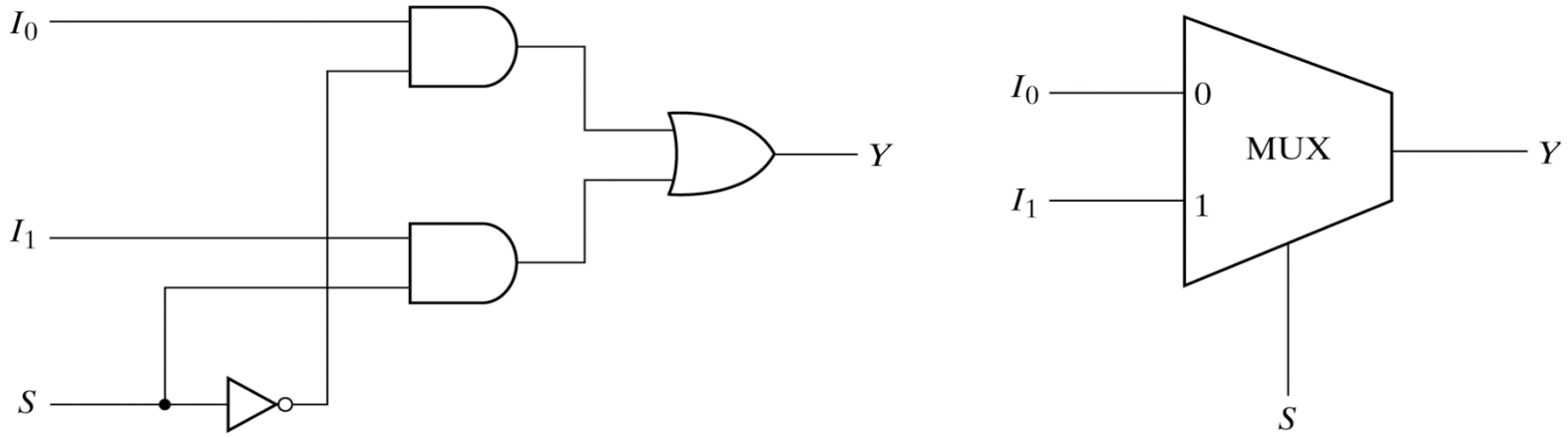
Multiplexers

S = 0, Y = I_0
S = 1, Y = I_1

Truth Table→

S	Y
0	I_0
1	I_1

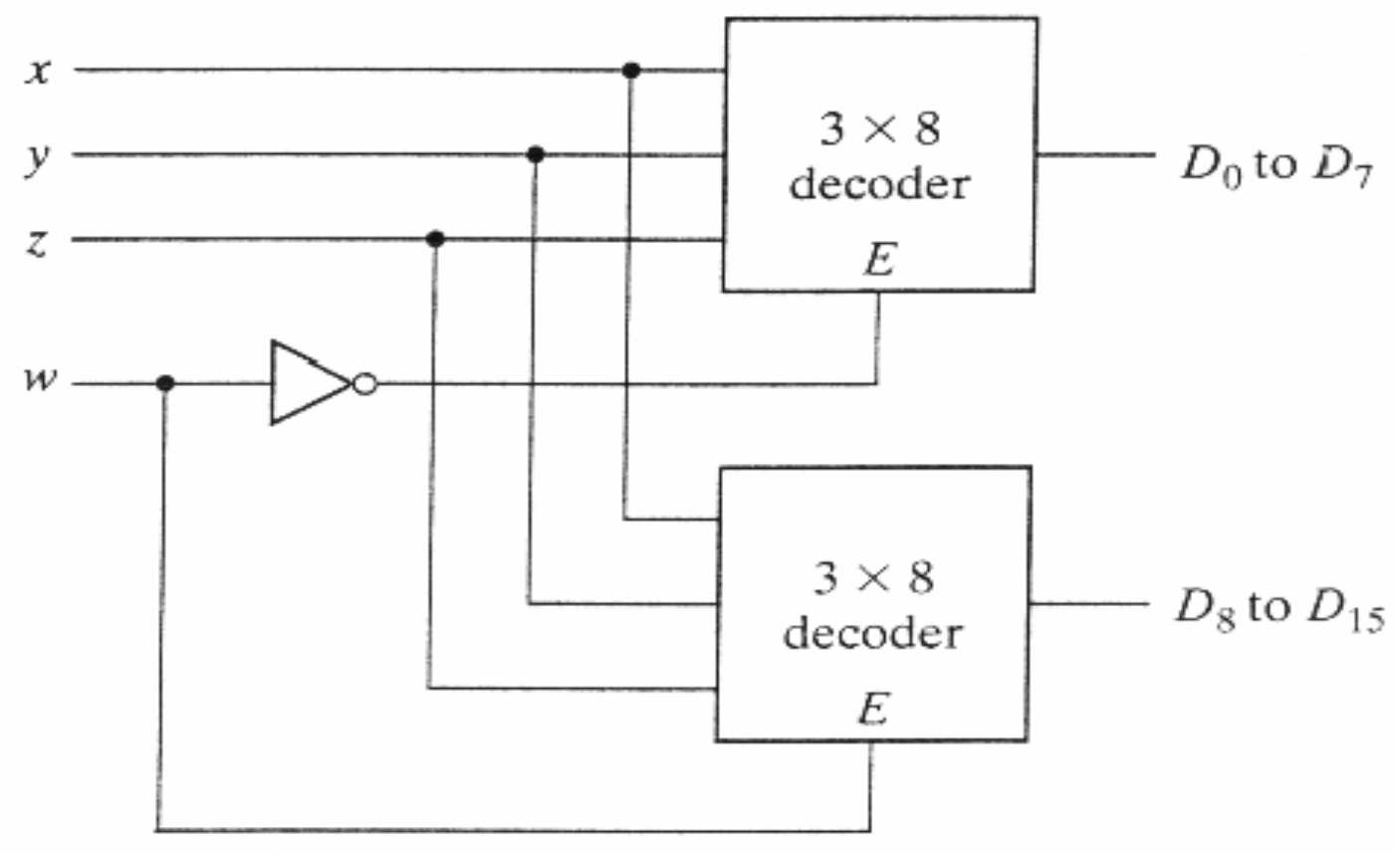
$Y = S'I_0 + SI_1$



(a) Logic diagram (b) Block diagram

Fig. 4-24 2-to-1-Line Multiplexer

Constructing larger decoders



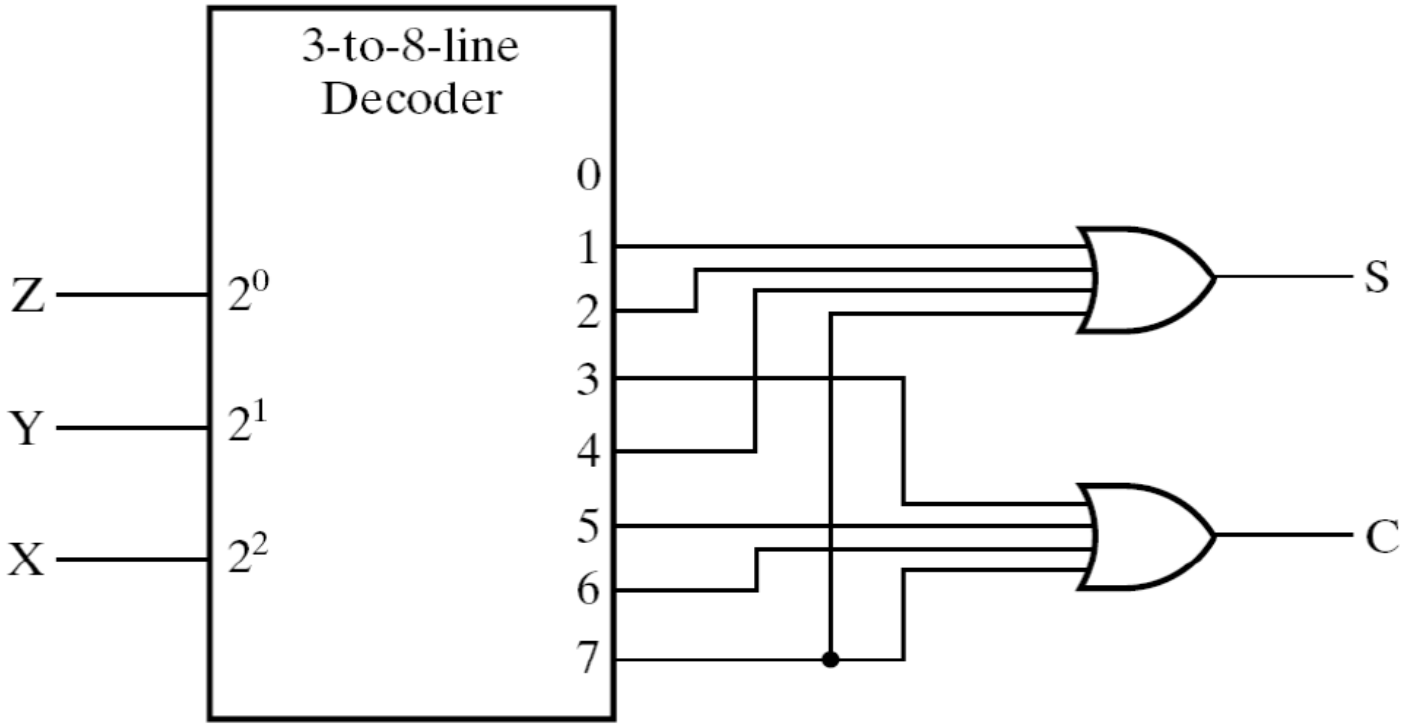
Implementing combinational logic functions using decoders

Truth Table for 1-bit Binary 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

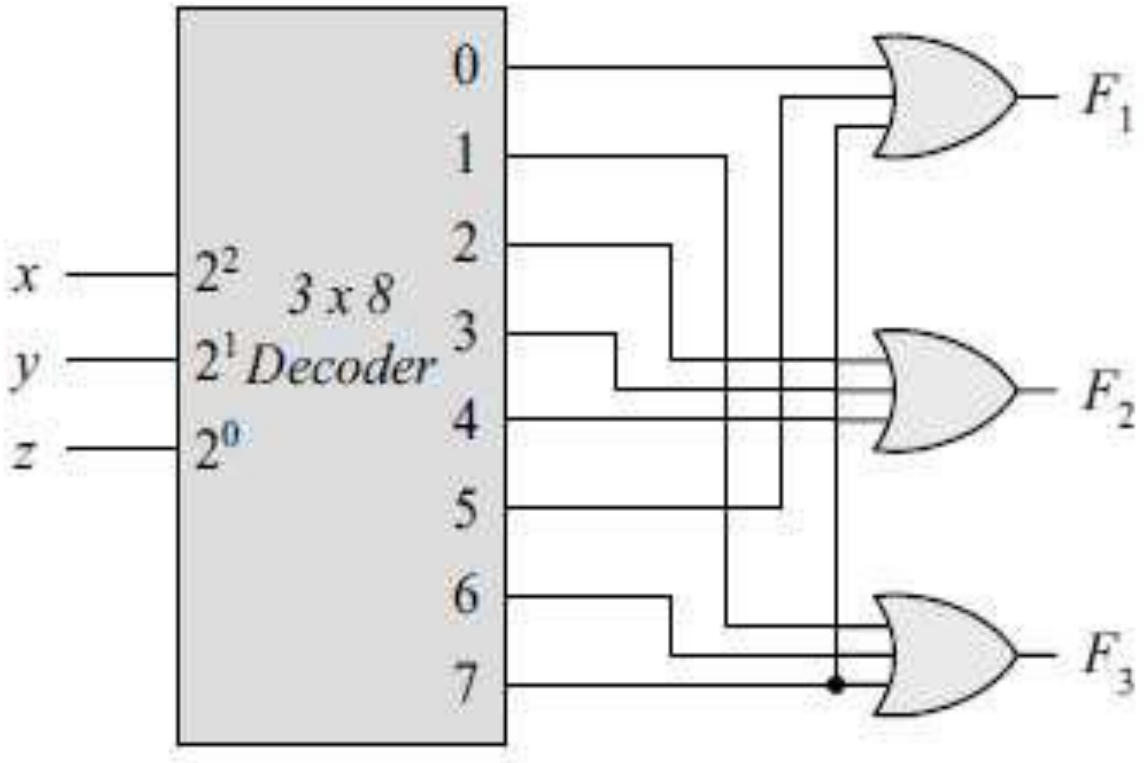
$$S(X,Y,Z) = \sum m(1,2,4,7)$$

$$C(X,Y,Z) = \sum m(3,5,6,7)$$



Implementing combinational logic functions using decoders

F 1 = x'y'z' + xz = Σ(0,5,7)
F 2 = xy'z' + x'y = Σ(2,3,4)
F 3 = x'y'z + xy = Σ(1,6,7)



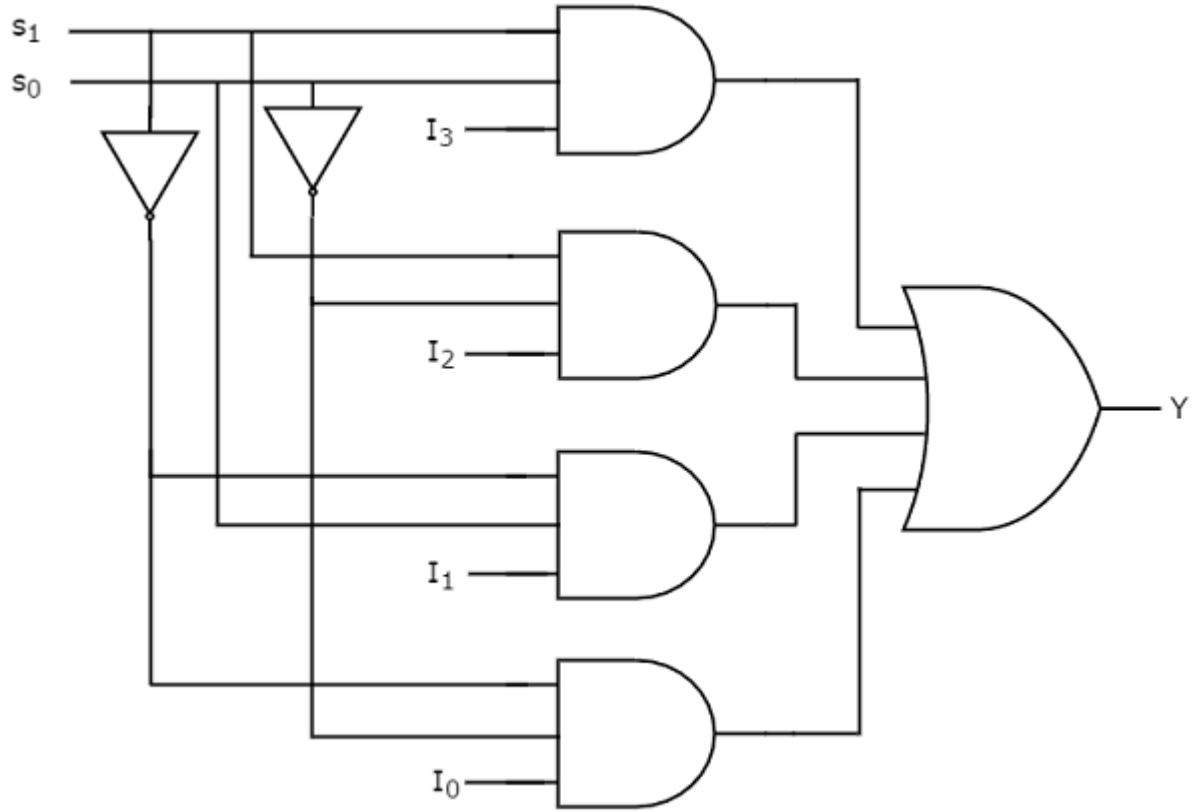
Implementing combinational logic functions using decoders

$F_1 = (y' + x)z$

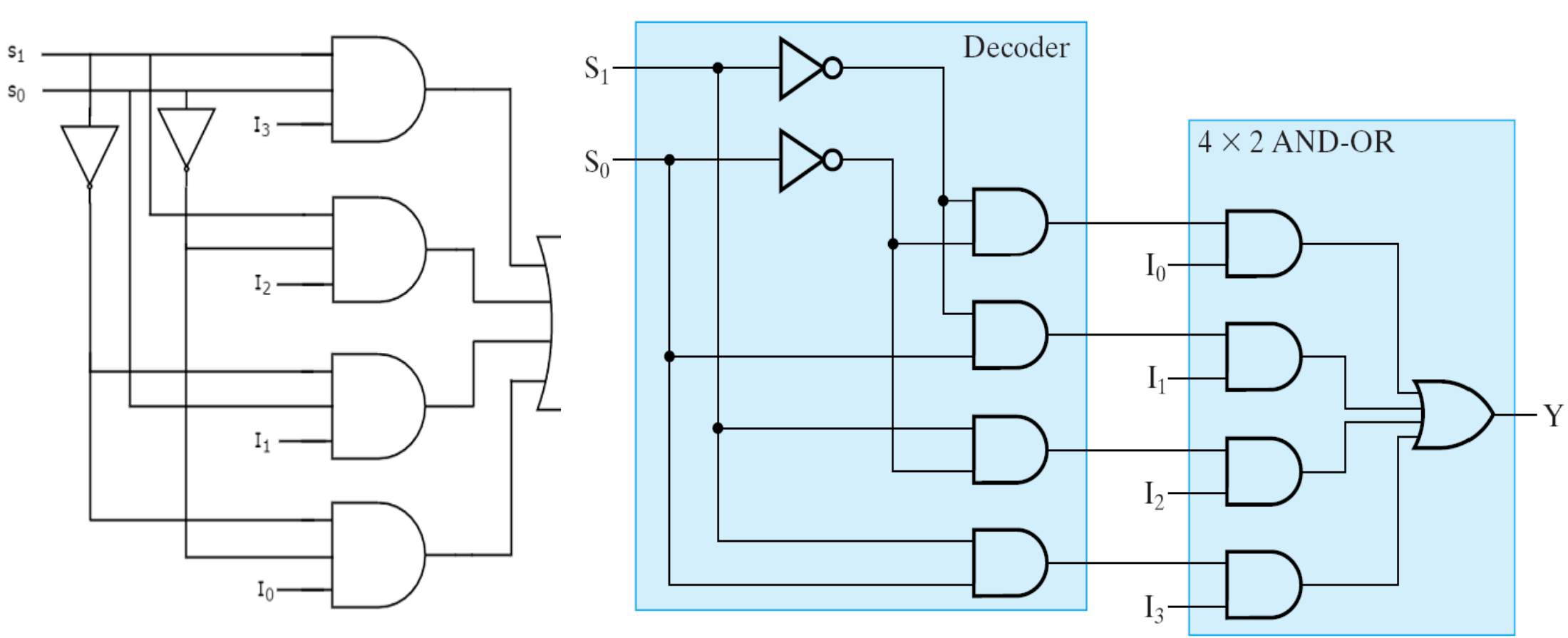
$F_2 = y'z' + xy' + yz'$

$F_3 = (x' + y)z$

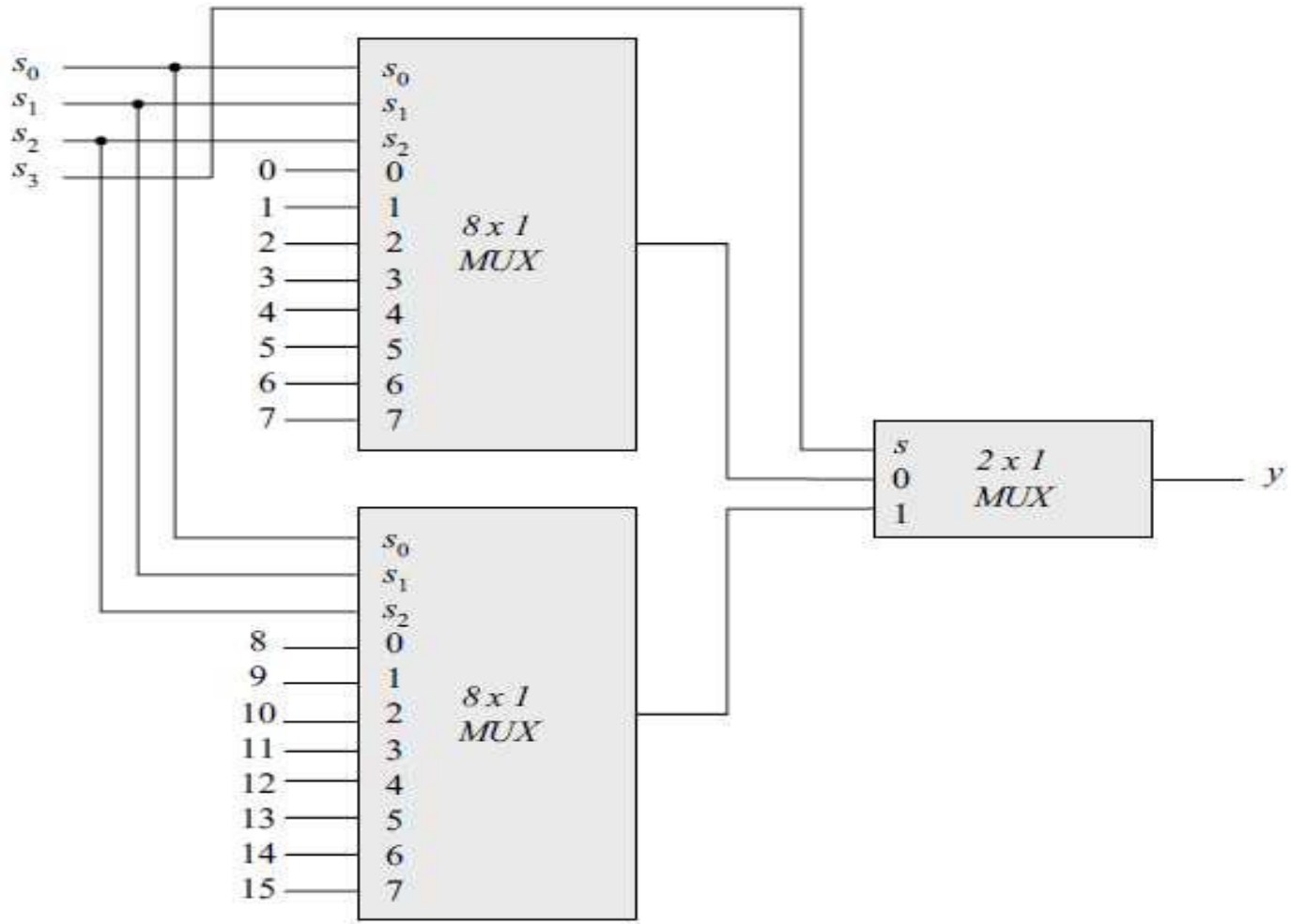
4 to 1 multiplexer



4 to 1 multiplexer



Construction of larger multiplexers



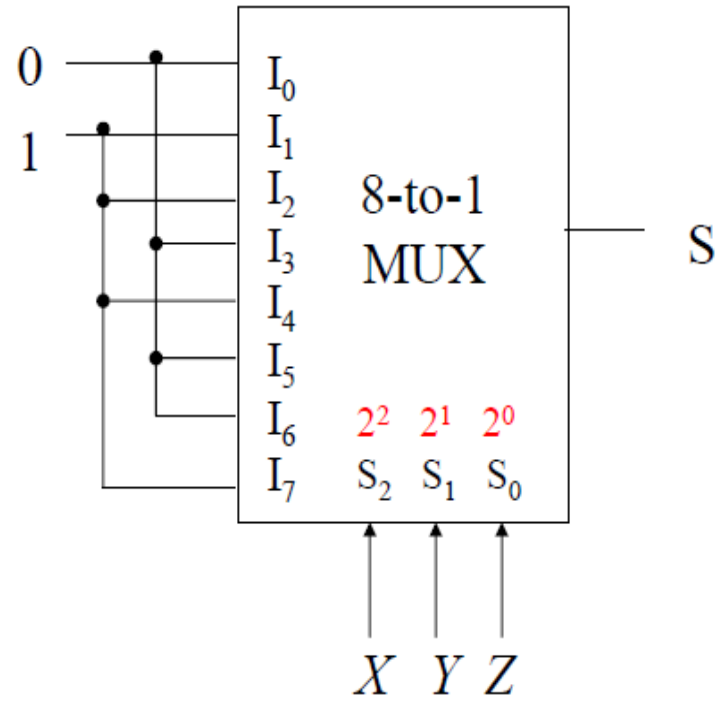
MUX implementation of a binary adder bit

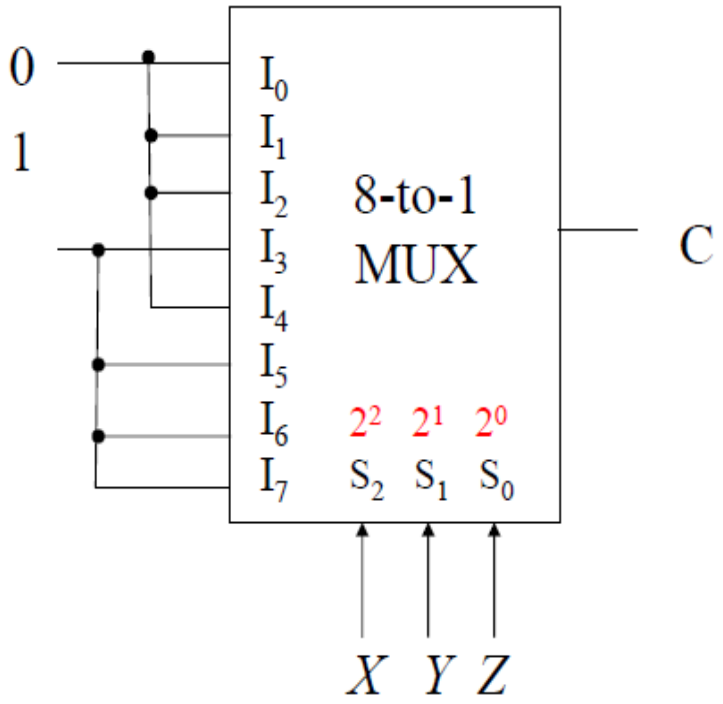
Truth Table for 1-bit Binary Adder

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

$$S(X,Y,Z) = \sum m(1,2,4,7)$$

$$C(X,Y,Z) = \sum m(3,5,6,7)$$

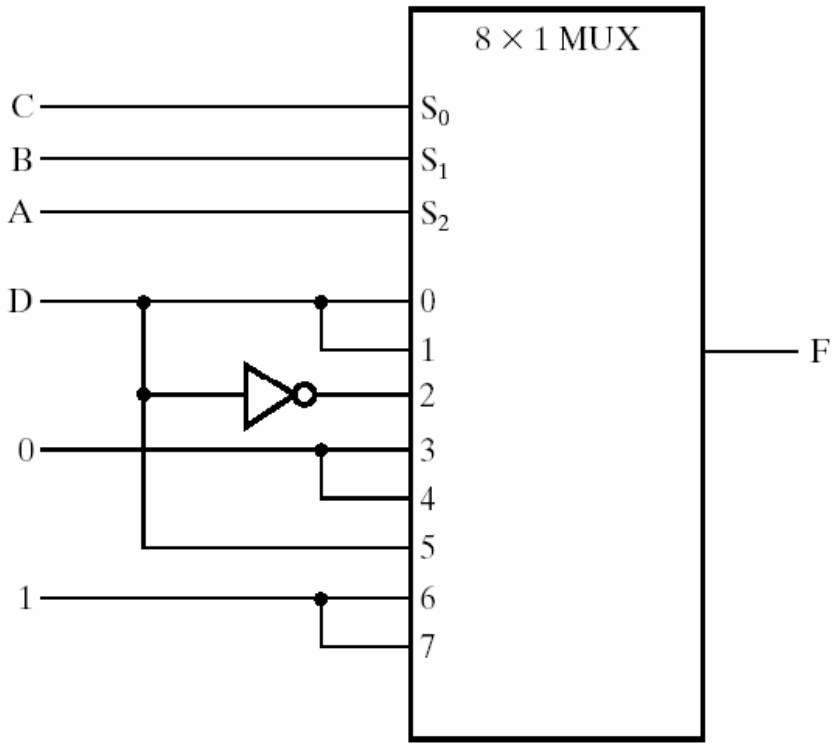




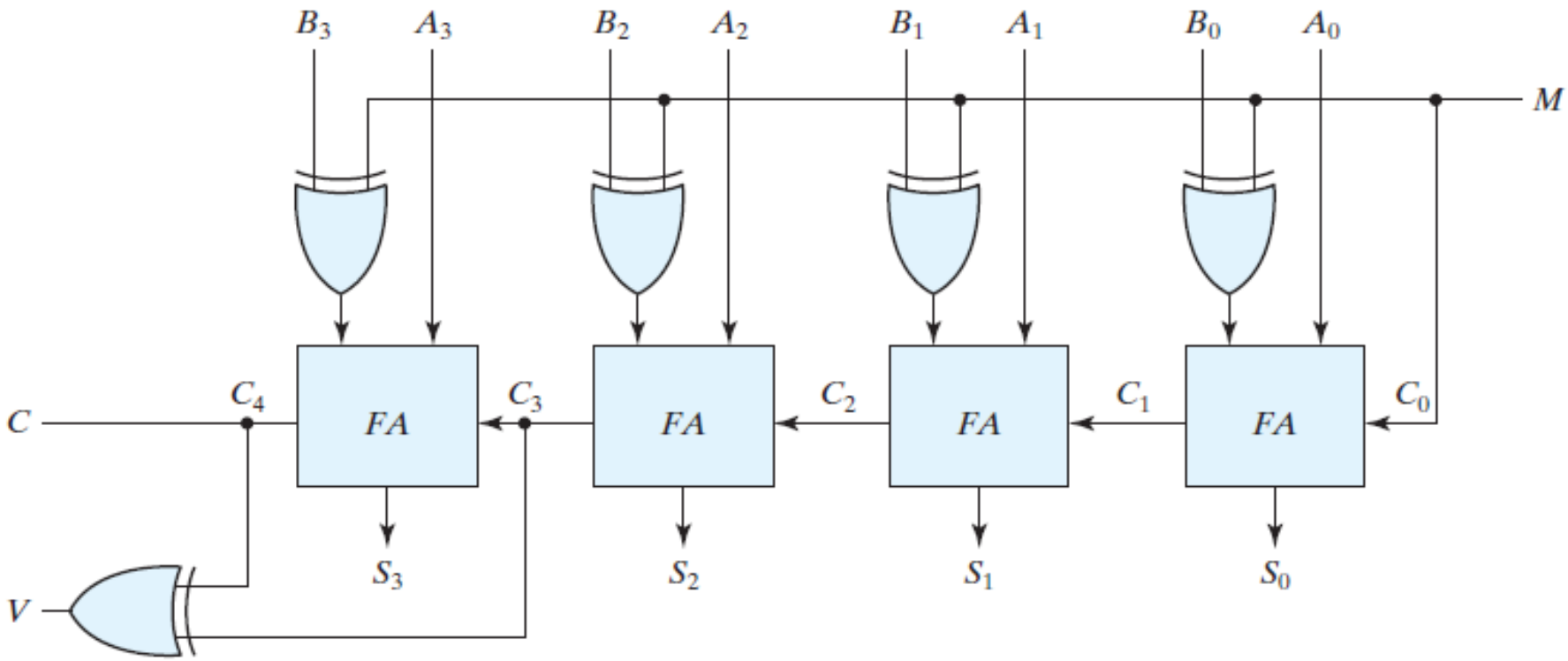
Implementing combinational logic functions using multiplexer

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

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 = $\overline{D}$
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	



4-bit Adder-Subtractor



Priority Encoder