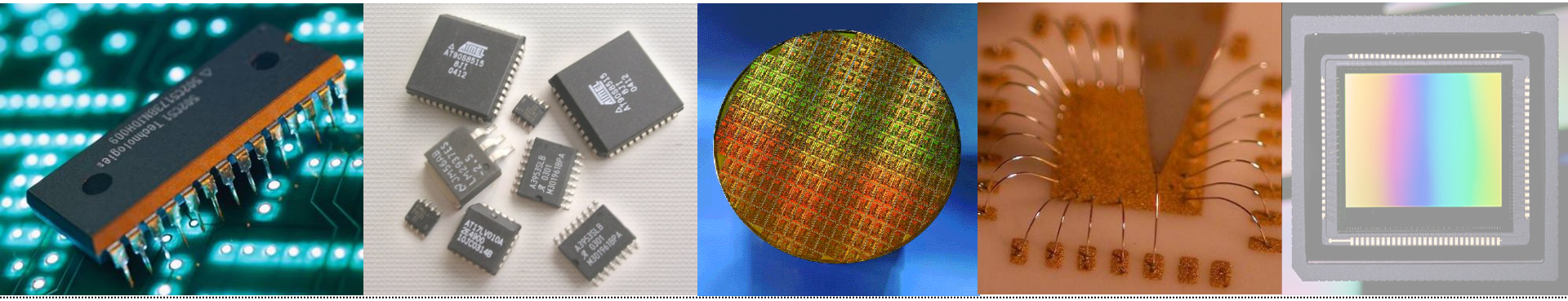


Ch 4. Combinational logic

4.2 Combinational circuits

- Outputs are determined from the present inputs
- Consist of input/output variables and logic gates

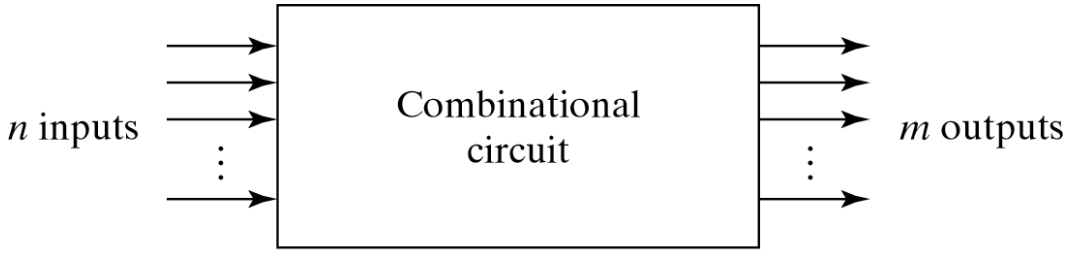


Fig. 4-1 Block Diagram of Combinational Circuit

4.3 Analysis procedure

- To determine the function of circuit

- Analysis procedure
  - Make sure the circuit is combinational or sequential
  - Obtain the output Boolean functions or the truth table

4.3 Analysis procedure

- Boolean function
  - Label all gate outputs
  - Make output functions at each level
  - Substitute final outputs to input variables
- Truth table
  - Put the input variables to binary numbers
  - Determine the output value at each gate
  - Obtain truth table

4.3 Analysis procedure

Table 4-1
Truth Table for the Logic Diagram of Fig. 4-2

A	B	C	F_2	F_2	T_1	T_2	T_3	F_1
0	0	0	0	1	0	0	0	0
0	0	1	0	1	1	0	1	1
0	1	0	0	1	1	0	1	1
0	1	1	1	0	1	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	1	0	1	0	0	0
1	1	0	1	0	1	0	0	0
1	1	1	1	0	1	1	0	1

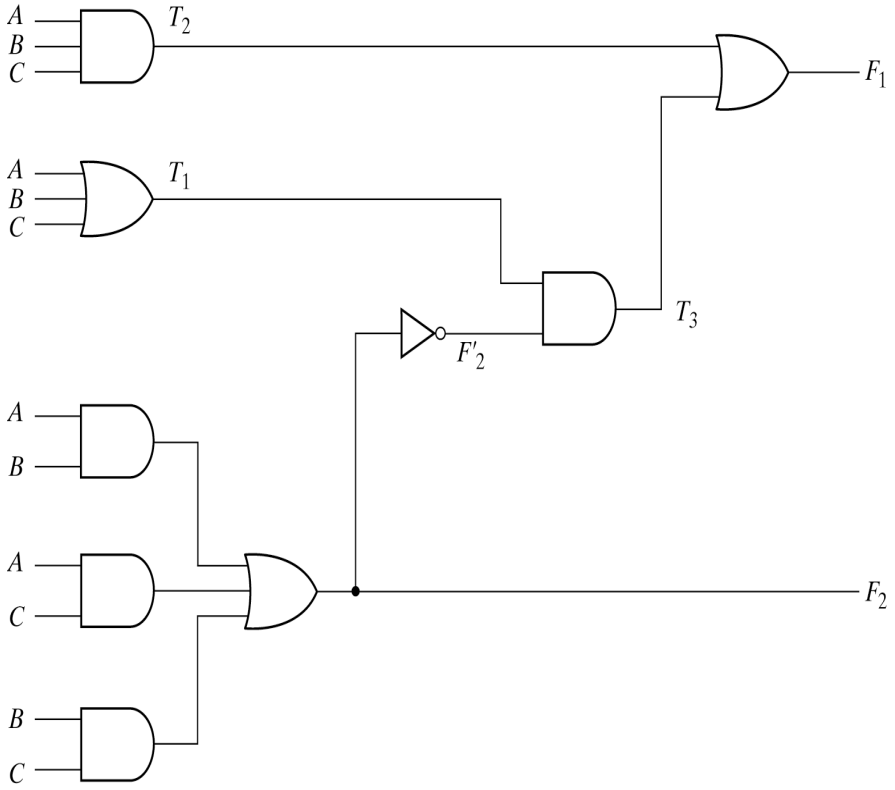


Fig. 4-2 Logic Diagram for Analysis Example

4.4 Design procedure

- Procedure to design
  - Determine the required number of input and output fro m specification
  - Assign a letter symbol to each input/output
  - Derive the truth table
  - Obtain the simplified Boolean functions
  - Draw the logic diagram and verify design correctness

4.5 Binary adder-subtractor

- Binary adder
  - Half adder : performs the addition of 2-bits(x+y)
  - Full adder : performs the addition of 3-bits(x+y +z)
  - Two half adder can be employed to a full adder

- Realization of Binary adder-subtractor
  - Half adder
  - Full adder
  - Cascade of n-full adder
  - Providing a complementing circuit

4.5 Binary adder-subtractor - Half Adder

- Sum of 2 binary inputs
  1. Determine # of input & outputs
     - 2-bit input, 2-bit output

- Input : X(augend), Y(addend) 2. assign variables
  Output : S(sum), C(carry)

  3. draw truth table 4. simplify logic if possible

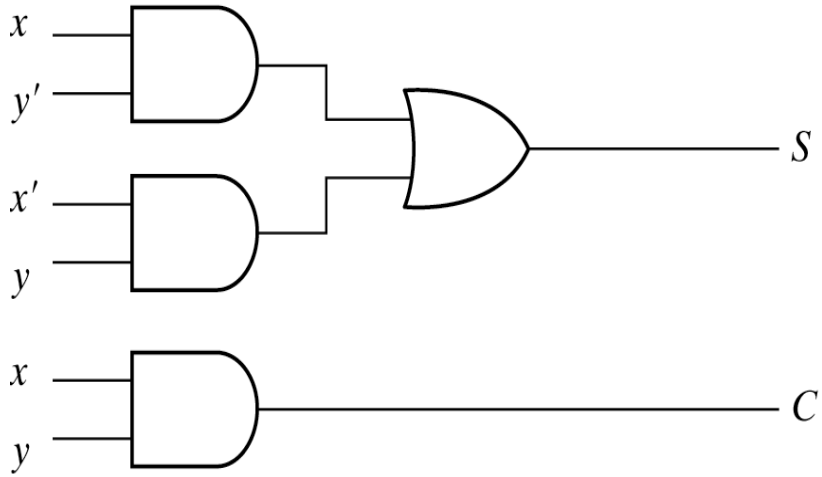
Table 4-3
Half Adder

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

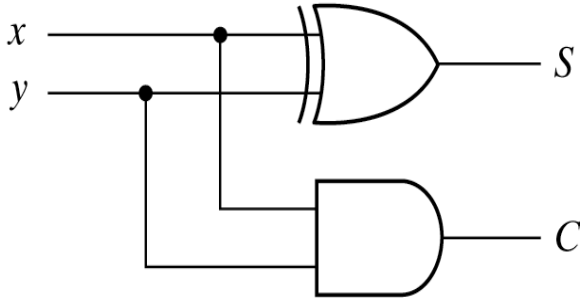
$S = xy' + x'y$

$C = xy$

4.5 Binary adder-subtractor - Half Adder



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



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

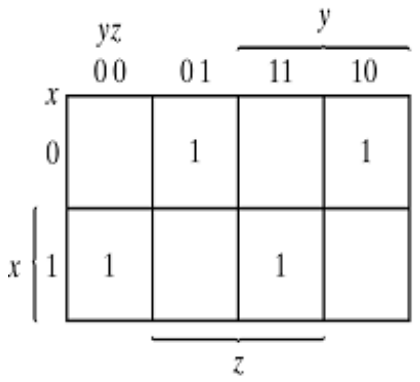
Fig. 4-5 Implementation of Half-Adder

4.5 Binary adder-subtractor - Full adder

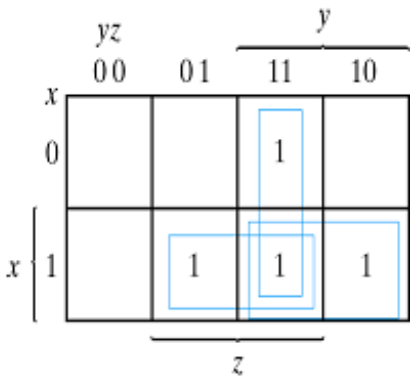
- Sum of 3 binary inputs
- Input : X,Y(2 significant bits),Z(1 carry bit)
- Output : S(sum),C(carry)

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



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



$C = xy + xz + yz$
$= xy + xy'z + x'yz$

Fig. 4-6 Maps for Full Adder

4.5 Binary adder-subtractor - Full adder

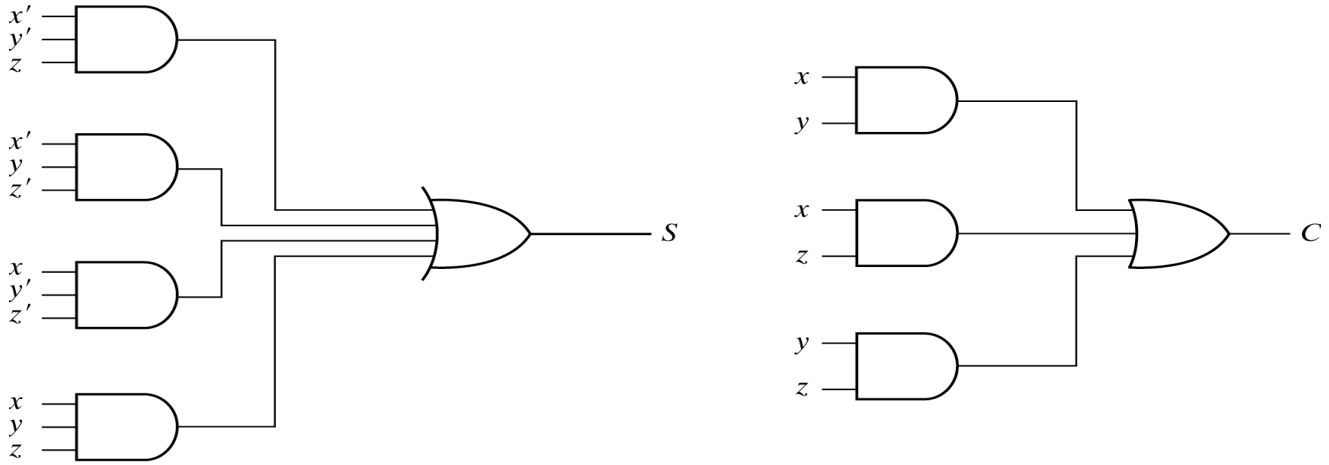


Fig. 4-7 Implementation of Full Adder in Sum of Products

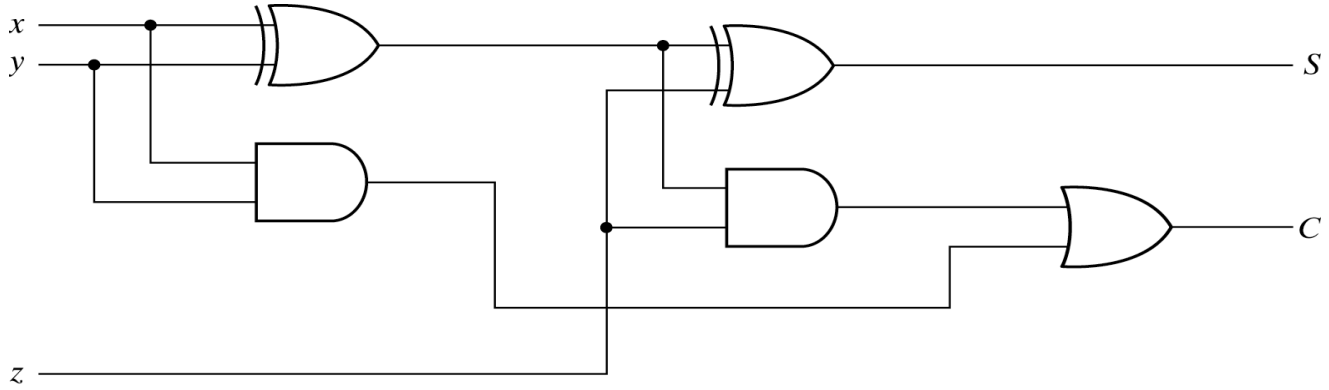


Fig. 4-8 Implementation of Full Adder with Two Half Adders and an OR Gate

4.5 Binary adder-subtractor - Binary adder

- Sum of two n-bit binary numbers
  - 4-bit adder

A=1011, B=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}

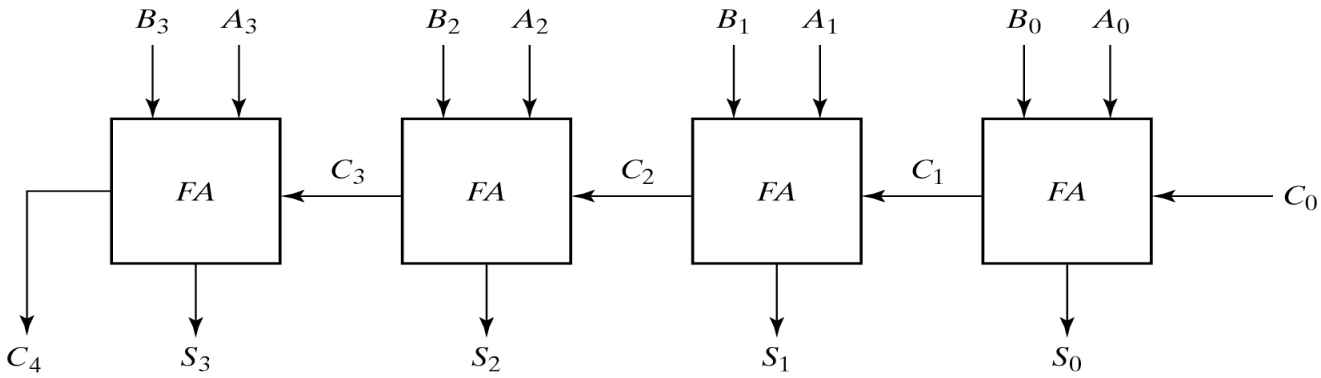


Fig. 4-9 4-Bit Adder

4.5 Binary adder-subtractor - Carry propagation

- Rising of delay time(carry delay)
- One solution is ***carry lookahead***
- All carry is a function of P_i,G_i and C_0

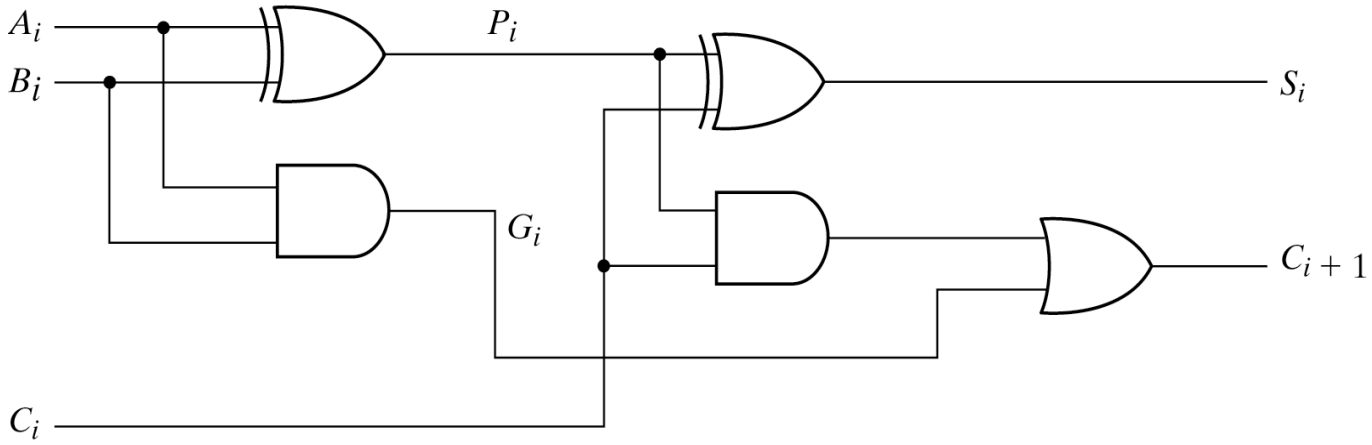


Fig. 4-10 Full Adder with P and G Shown

$$C_{i+1}=G_i+P_iC_i$$
$$S_i = P_i \oplus C_i$$

4.5 Binary adder-subtractor - Carry propagation

- Carry lookahead generator

C_0 = input carry

$C_1 = G_0 + P_0C_0$

$C_2 = G_1 + P_1C_1 = G_1 + P_1(G_0 + P_0C_0) = G_1 + P_1G_0 + P_1P_0C_0$

$C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$

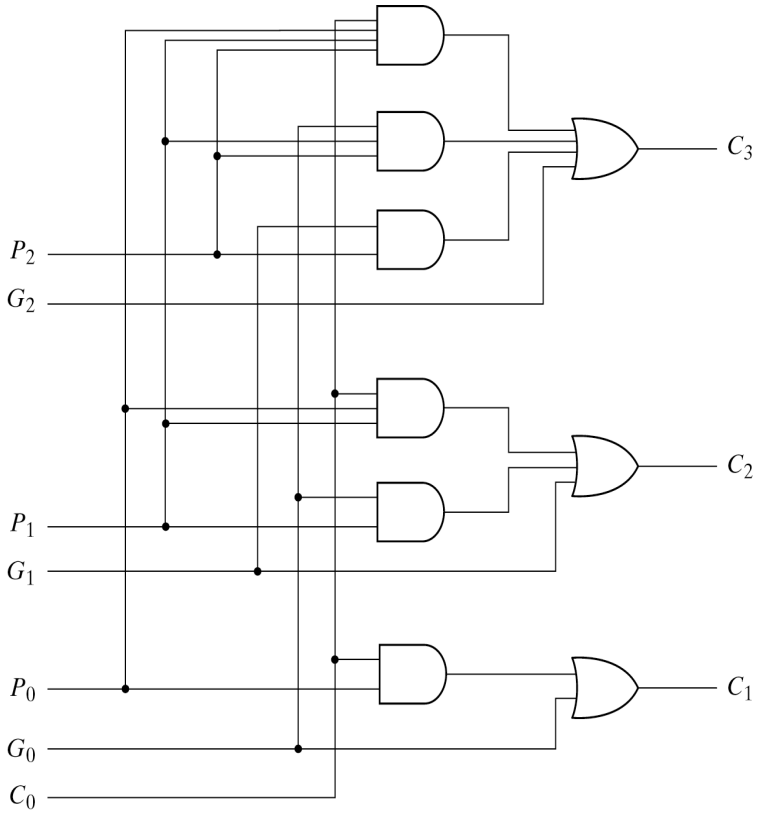


Fig. 4-11 Logic Diagram of Carry Lookahead Generator

4.5 Binary adder-subtractor - Carry propagation

- 4-bit adder with carry lookahead

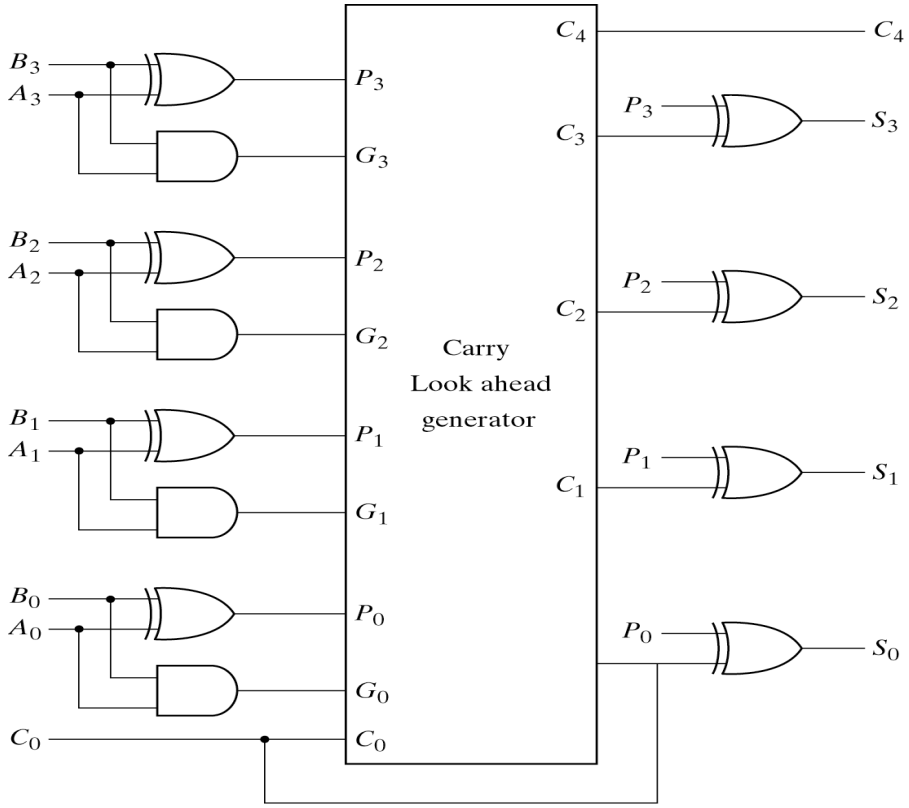


Fig. 4-12 4-Bit Adder with Carry Lookahead

4.5 Binary adder-subtractor - Binary subtractor

- A-B equals A+(2'complement of B)
- When M=0(act as adder) M=1(subtractor)

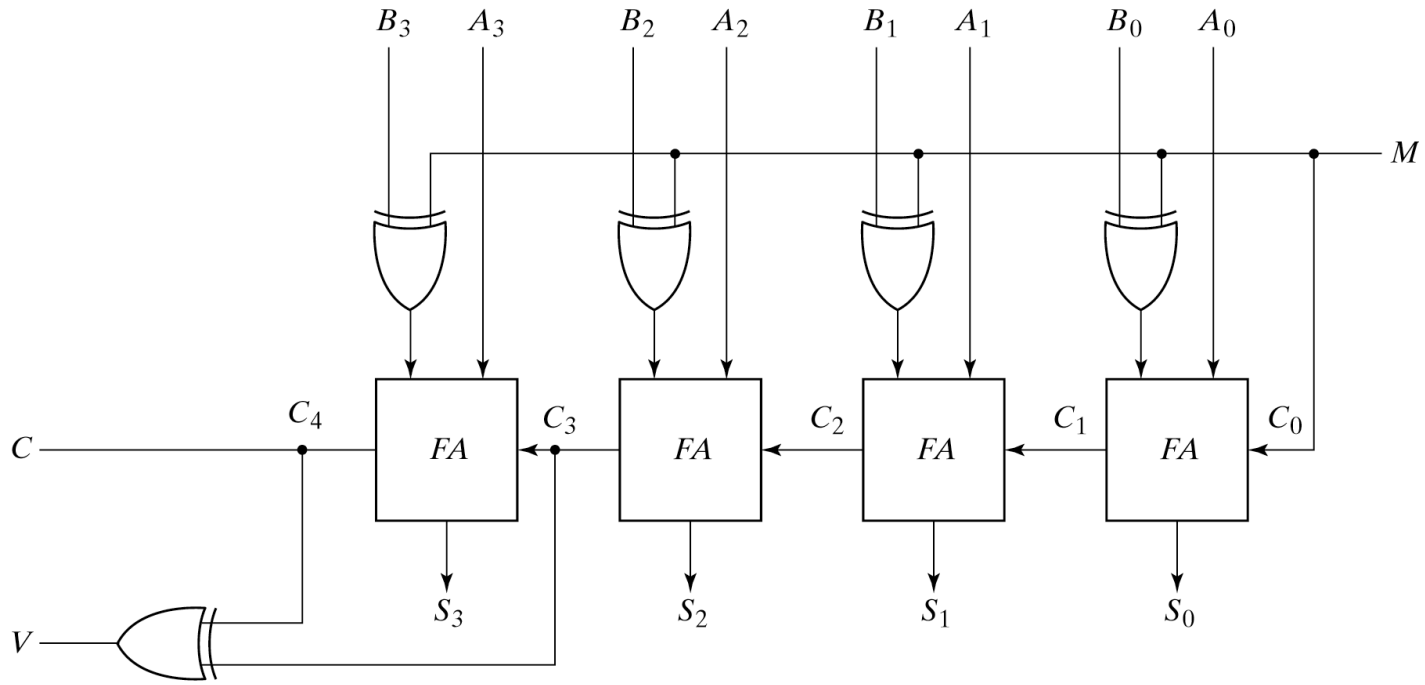


Fig. 4-13 4-Bit Adder Subtractor

4.5 Binary adder-subtractor - Overflow

- Sum of *n* digit number occupies *n*+1digit
- Occurs when two numbers are same sign

(examples of overflow)

carries:	0	1		carries:	1	0	
+70		0	1000110	−70		1	0111010
+80		0	1010000	−80		1	0110000
+150		1	0010110	−150		0	1101010

4.6 Decimal adder

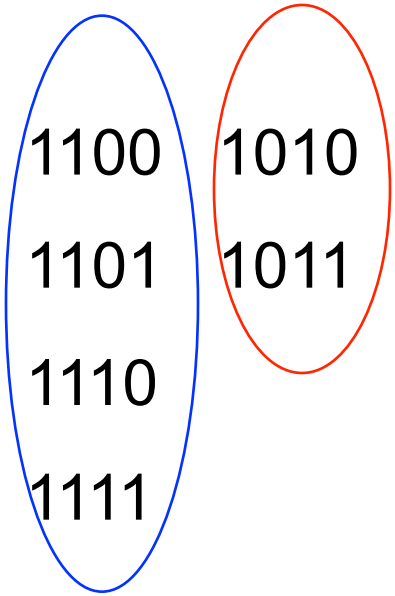
- Calculate binary and represent decimal in binary coded f orm
- Decimal adder for the BCD code

Table 4.5
Derivation of BCD Adder

Binary Sum					BCD Sum					Decimal
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

4.6 Decimal adder - BCD Adder

- BCD digit output of 2-B CD digit sum
- Carry arise if output 101 0~1111
- $C=K+Z_8Z_4+Z_8Z_2$



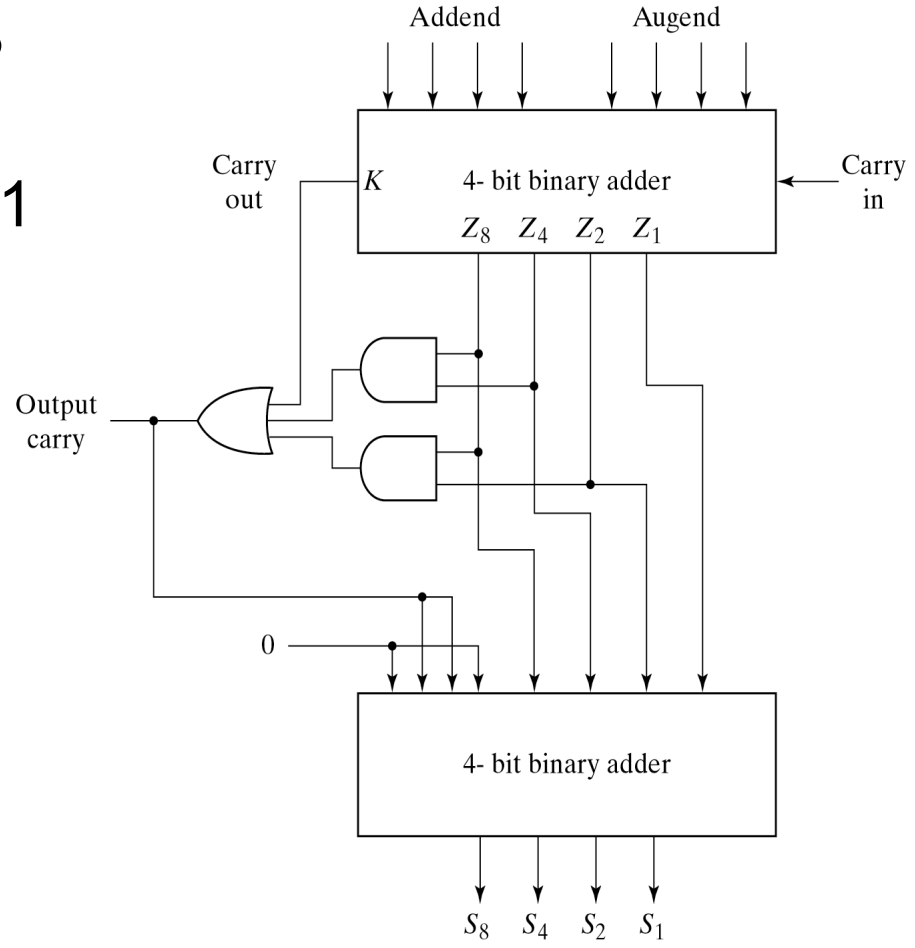


Fig. 4-14 Block Diagram of a BCD Adder

4.7 Binary multiplier

- 2bit x 2bit = 4bit(max)

$$
\begin{array}{cccc}
 & & B_1 & B_0 \\
 & & A_1 & A_0 \\
\hline
 & & A_0B_1 & A_0B_0 \\
 & A_1B_1 & A_1B_0 & \\
\hline
C_3 & C_2 & C_1 & C_0
\end{array}
$$

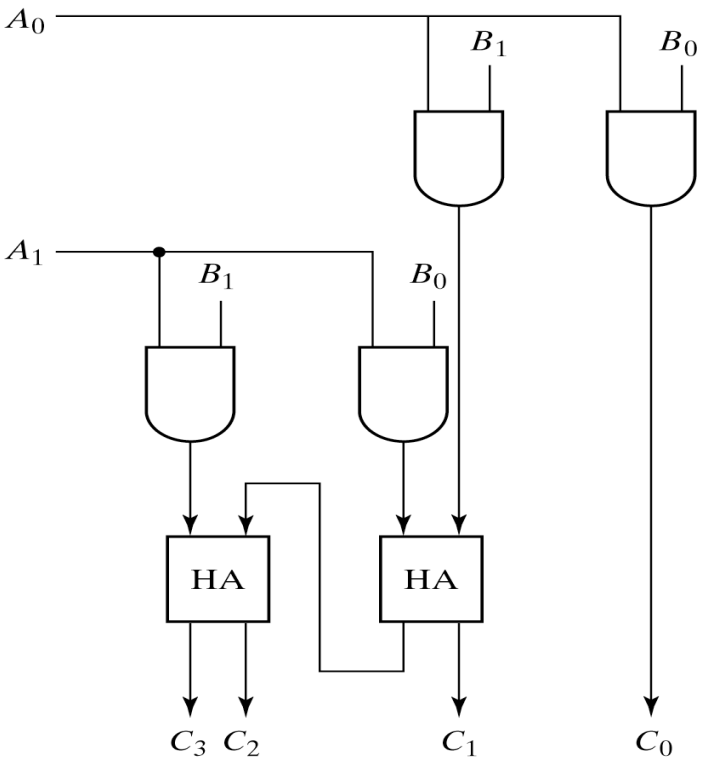


Fig. 4-15 2-Bit by 2-Bit Binary Multiplier

4.7 Binary multiplier

- (K-bit) x (J-bit)
  - (K x J) AND gates,

  (J-1) K-bit adder needed

$$\begin{array}{ll} & B_3B_2B_1B_0 \\ \times & A_2A_1A_0 \\ \hline \end{array}$$

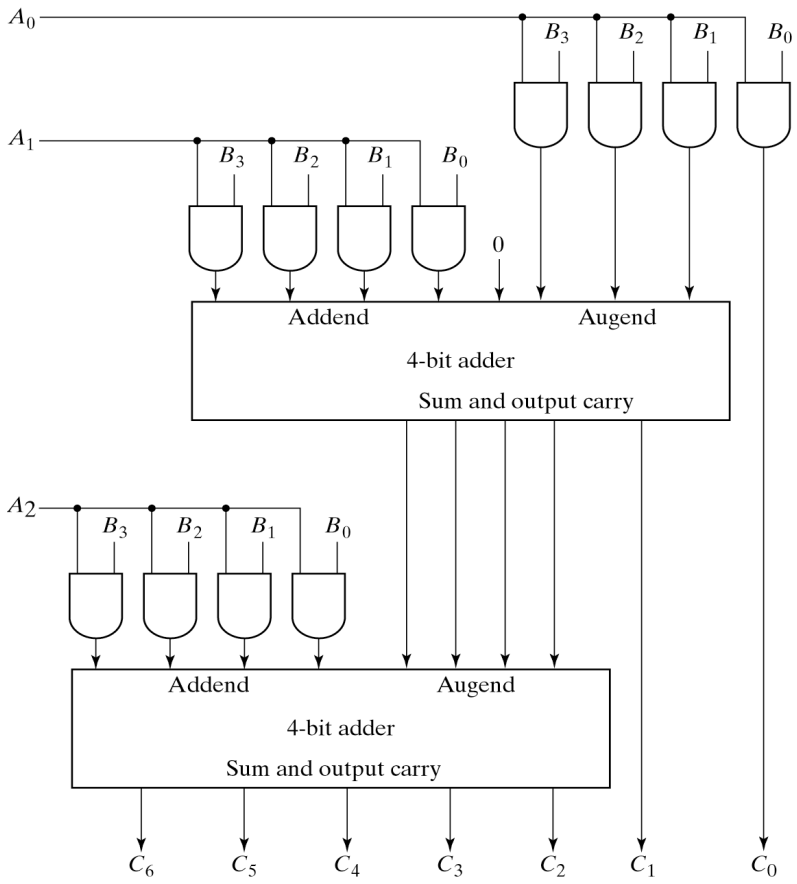


Fig. 4-16 4-Bit by 3-Bit Binary Multiplier

4.8 Magnitude comparator

- X_i=1only if the pair of bits A_i and B_i are equal
  - $X_i=A_iB_i+A_i'B_i'$
- $(A=B)=x_3x_2x_1x_0$
- $(A>B)=A_3B_3'+x_3A_2B_2'+x_3x_2A_1B_1'+x_3x_2x_1A_0B_0'$
- $(A<B)=A_3'B_3+x_3A_2'B_2+x_3x_2A_1'B_1+x_3x_2x_1A_0'B_0$

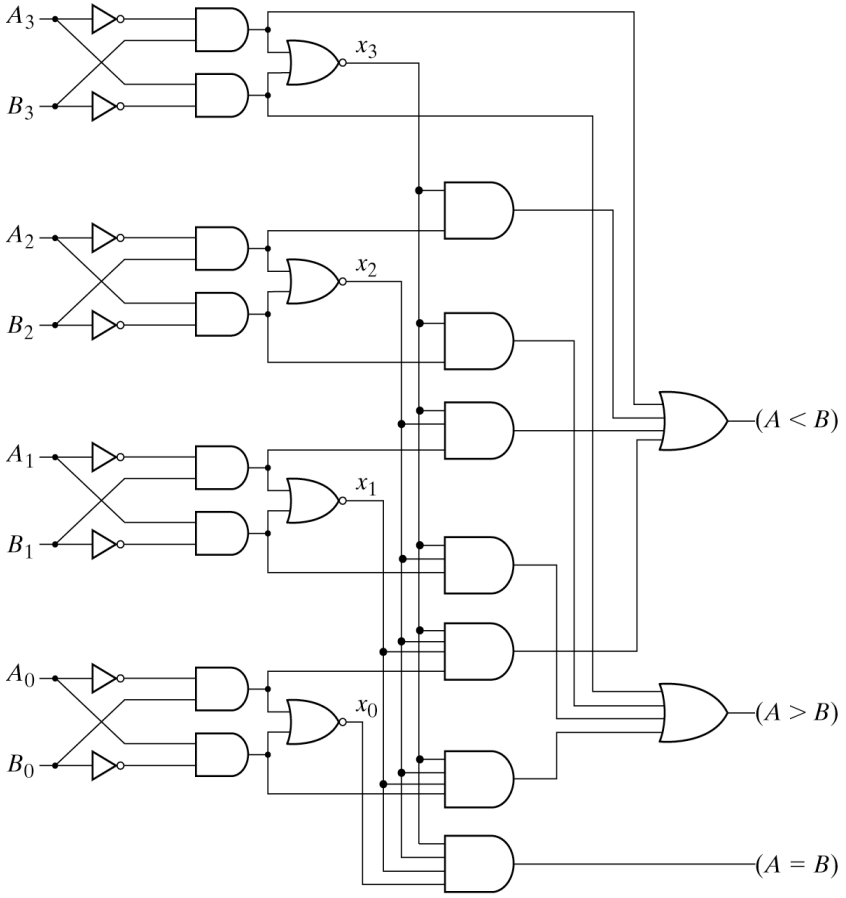


Fig. 4-17 4-Bit Magnitude Comparator

4.9 Decoders

- Generate the 2^n(or less) minterms of n input variables
  - Eg)3 to 8 line decoder

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

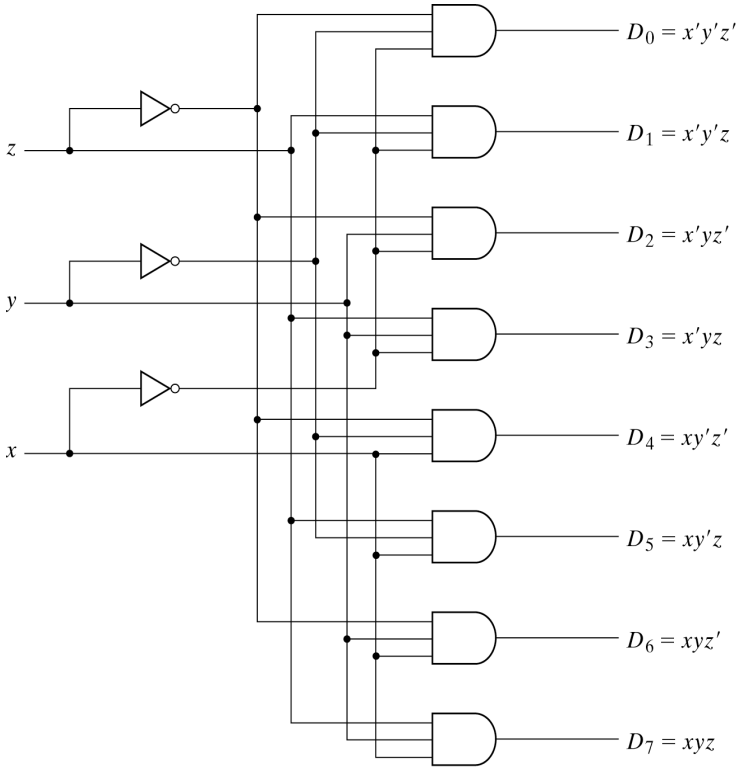
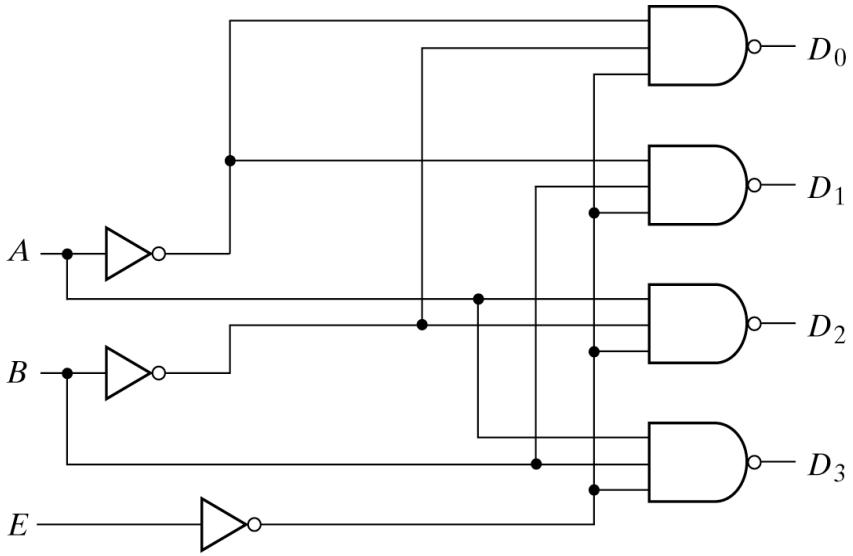


Fig. 4-18 3-to-8-Line Decoder

4.9 Decoders

- 2 to 4 line decoder with Enable input
  - Control circuit operation by E



(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

NAND gate implementation
negative logic is used

4.9 Decoders

- Decoders with enable inputs can be a larger decoder circ uit

Eg)4x16 decoder by t wo 3x8 decoders

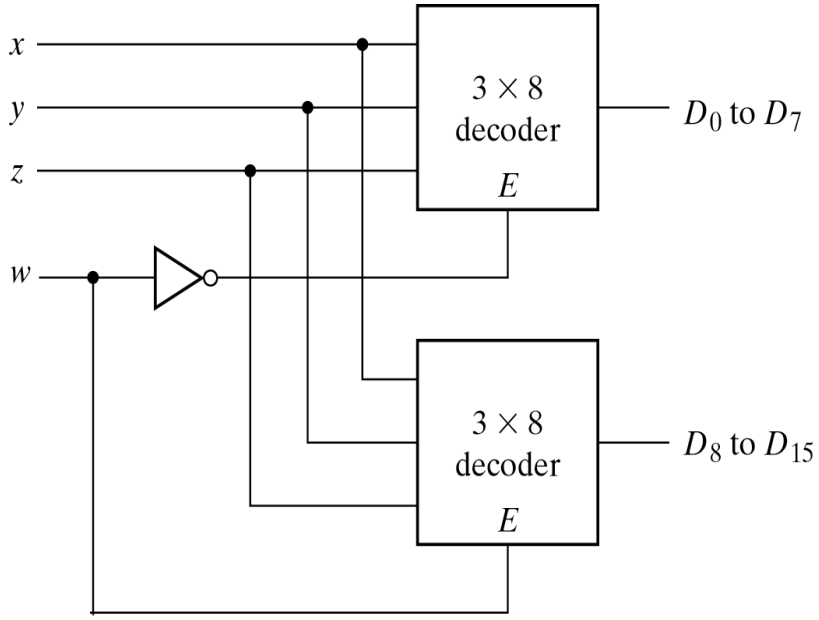


Fig. 4-20 4 × 16 Decoder Constructed with Two 3 × 8 Decoders

positive logic
E=1 enables decoder

- Combinational logic implementation
  - Any combinational circuit can be implemented with lin e decoder and OR gates
  - Eg)full 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

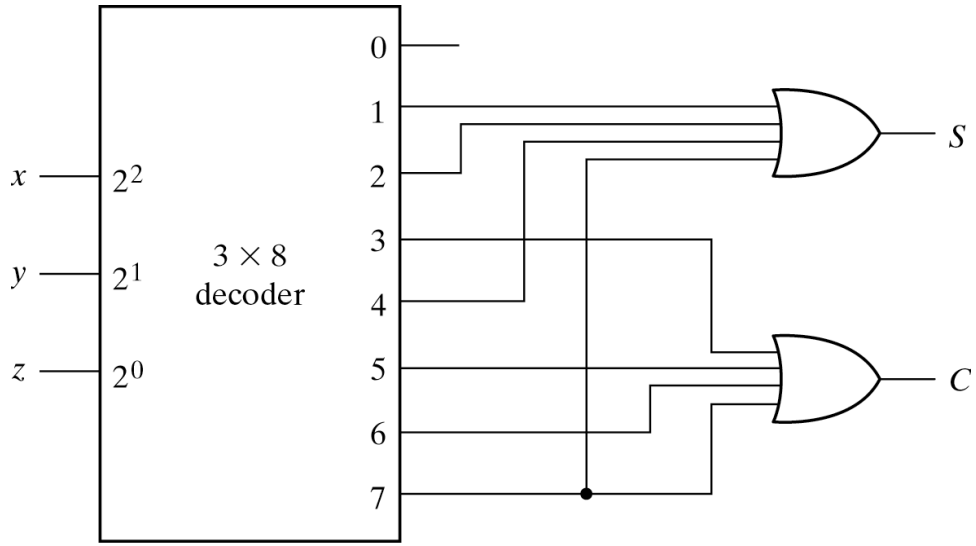


Fig. 4-21 Implementation of a Full Adder with a Decoder

4.10 Encoders

- Inverse operation of a decoder
- Generate n outputs of 2^n input values
  - Eg) octal to binary 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

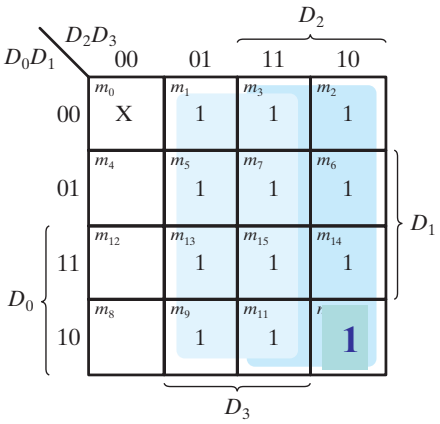
4.10 Encoders - Priority encoder

- Problem happens two or more inputs equal to 1 at the sa me time
- Give a priority function to circuit

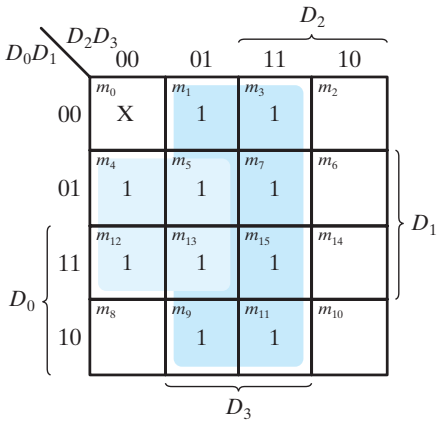
Table 4.8
Truth Table of a Priority Encoder

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

(x100 means 0100,1100)



$x = D_2 + D_3$



$y = D_3 + D_1D'_2$

FIGURE 4.22
Maps for a priority encoder

4.11 Multiplexers

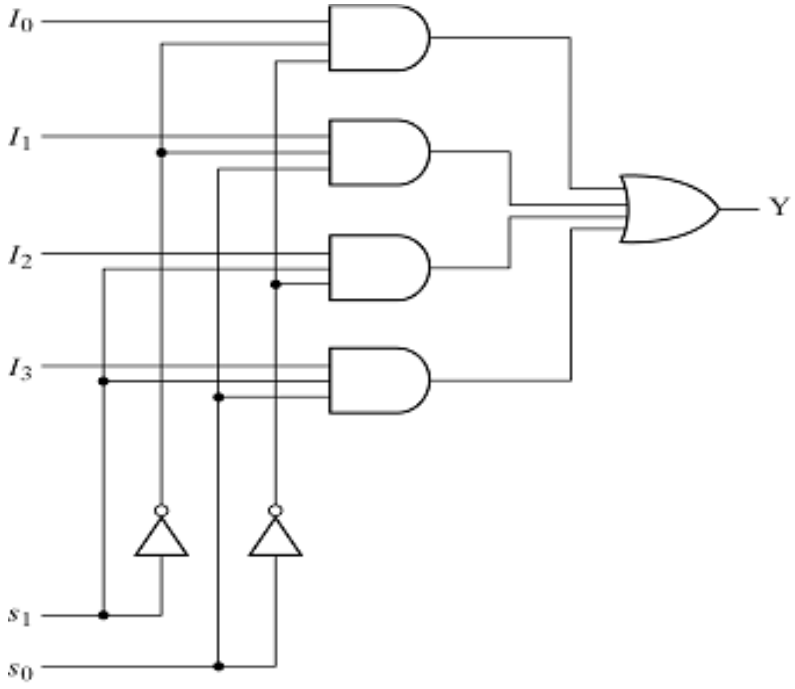
- Select a binary information from many input lines
- Selection is controlled by a set of selection lines
- 2^n input lines have *n* selection lines

4.11 Multiplexers

- 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

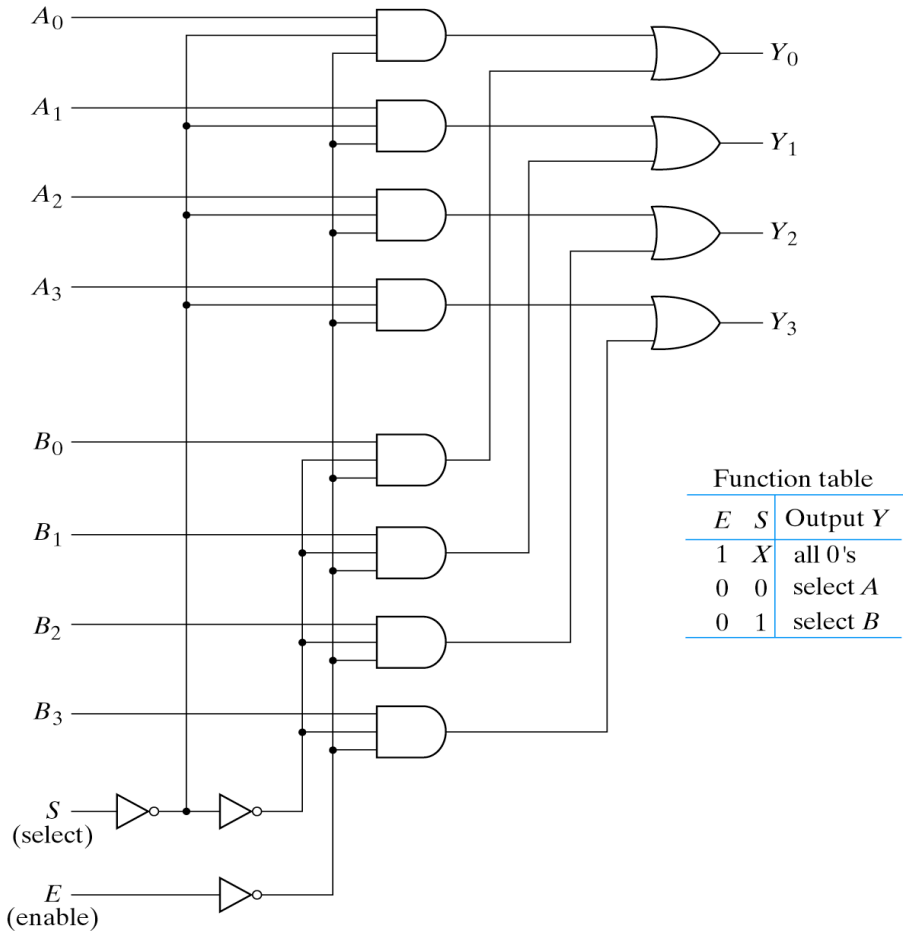
(b) Function table



(a) Logic diagram

4.11 Multiplexers

- Quadruple 2 to 1 line multiplexer



Function table

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

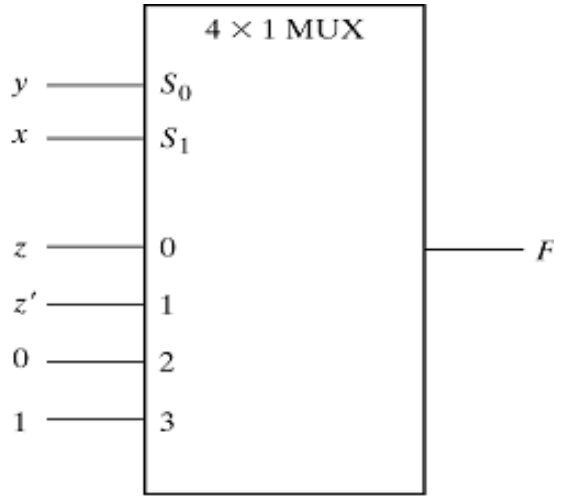
Fig. 4-26 Quadruple 2-to-1-Line Multiplexer

4.11 Multiplexers - Boolean function implementation

- Boolean function implementation
  - Minterms of function are generated in a MUX
  - n input variables, n-1 selection input

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

F=xy+yz′+x′y′z

4.11 Multiplexers - Three-state gates

- Three-state gates
  - Logic 1, 0 and *high-impedance*
  - *High-impedance* behaves like an open circuit

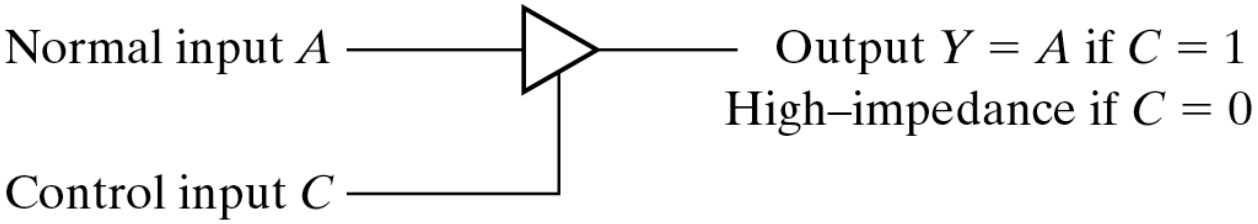


Fig. 4-29 Graphic Symbol for a Three-State Buffer

4.11 Multiplexers

- Multiplexers with three-state gates

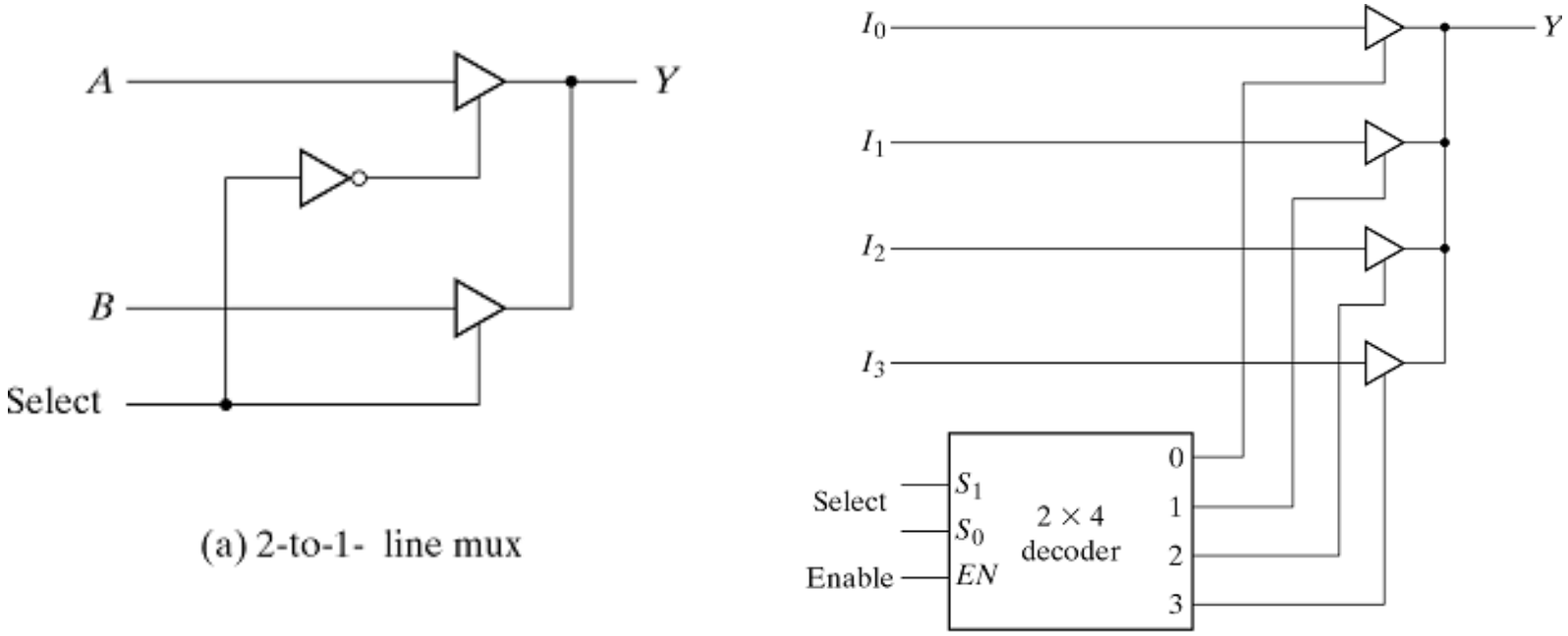


Fig. 4-30 Multiplexers with Three-State Gates