MEC520 디지털 공학

Combinational Logic

Jee-Hwan Ryu

School of Mechanical Engineering
Korea University of Technology and Education

Combinational circuits

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

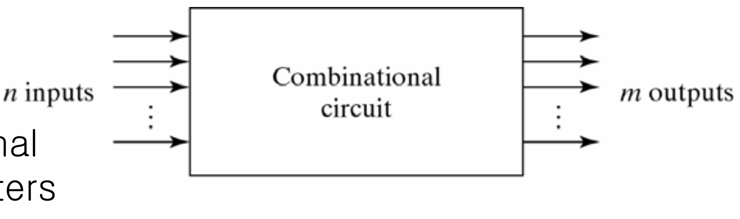


Fig. 4-1 Block Diagram of Combinational Circuit

- Sequential Circuits
  - Outputs are determined from the present inputs and the state of the storage elements
  - The state of the storage elements is a function of previous inputs
  - Depends on present and past inputs

Analysis procedure

- To determine the function from a given circuit diagram
- Analysis procedure
  - Make sure the circuit is combinational or sequential
    - No Feedback and memory elements
  - Obtain the output Boolean functions or the truth table

Obtain Procedure-Boolean Function

- Boolean function from a logic diagram
  - Label all gate outputs with arbitrary symbols
  - Make output functions at each level
  - Substitute final outputs to input variables

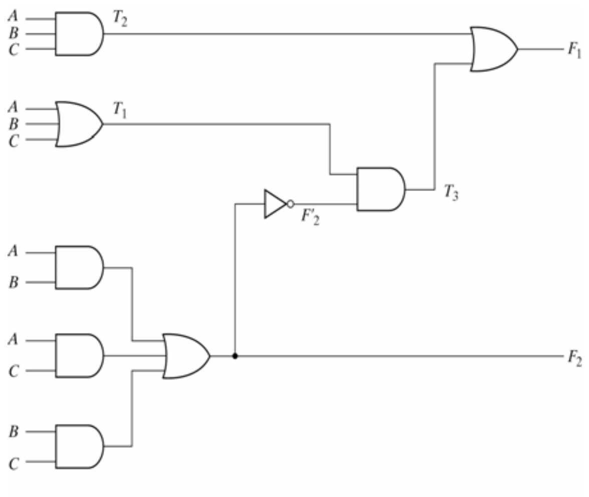


Fig. 4-2 Logic Diagram for Analysis Example

Obtain Procedure-Truth Table

- Truth table from a logic diagram
  - Put the input variables to binary numbers
  - Determine the output value at each gate
  - Obtain truth table

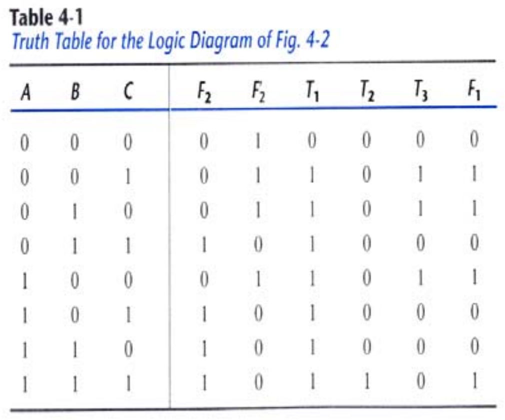


Table 4-1
Truth Table for the Logic Diagram of Fig. 4-2

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

Example

4-2 Obtain the simplified Boolean expressions for output F and G in terms of the input variables in the circuit of Fig. P4-2.

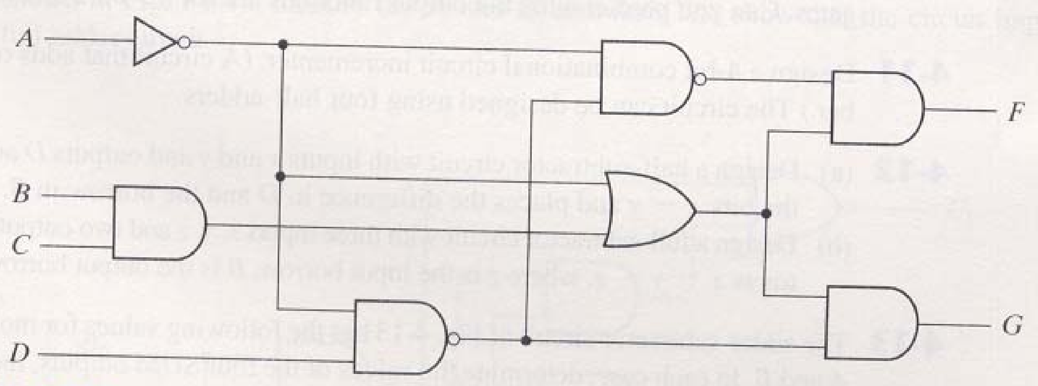


FIGURE P4-2

Design Procedure

- Procedure to design a combinational circuit
  1. Determine the required number of input and output from specification
  2. Assign a symbol to each input/output
  3. Derive the truth table from the required relationship
  4. Obtain the simplified Boolean functions
  5. Draw the logic diagram and verify design correctness

Code conversion example

- BCD to excess-3 code converter
  - Excess-3 code : decimal digit+3
- Design procedure

  1)Determine inputs/outputs

  Inputs : A,B,C,D (0000～1001)

  Outputs : W,X,Y,Z (0011～1100)

Code conversion example

2) Derive truth table

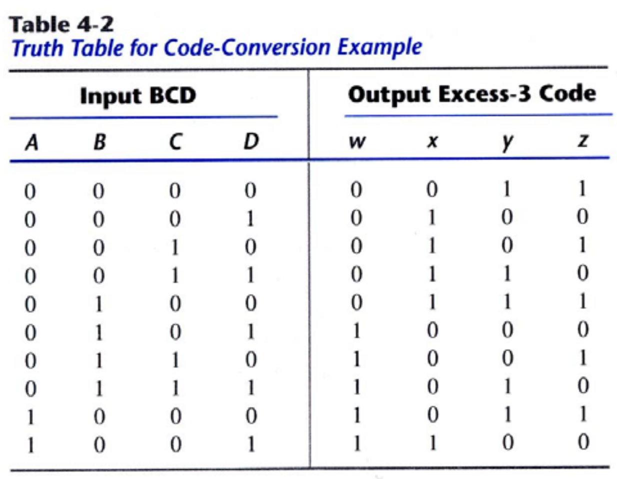


Table 4-2
Truth Table for Code-Conversion Example

Input BCD				Output Excess-3 Code			
A	*B*	*C*	*D*	*w*	*x*	*y*	*z*
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

Code conversion example

3) Obtain simplified Boolean functions

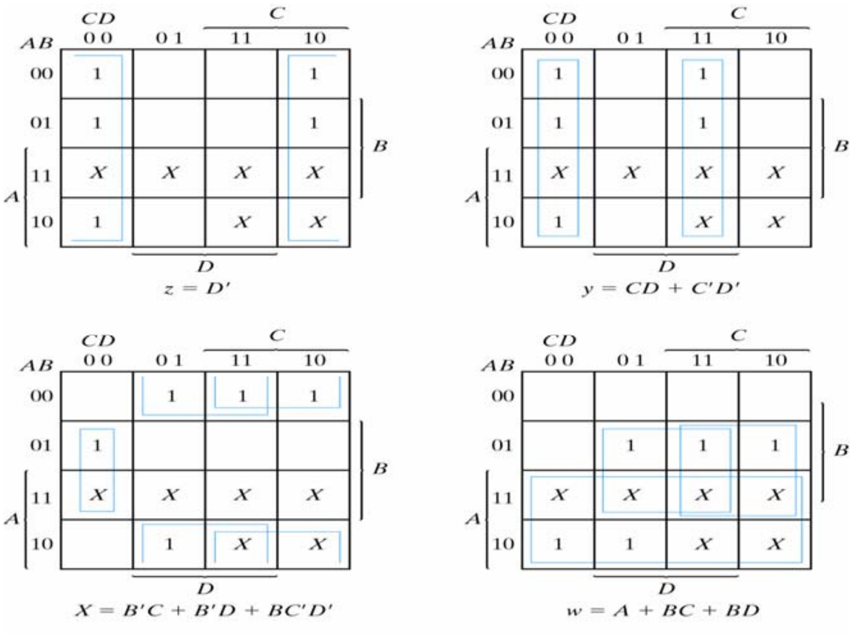


Fig. 4-3 Maps for BCD to Excess-3 Code Converter

Code conversion example

4) Draw the logic diagram

$$z = D'$$
$$y = CD + C'D' = CD + (C + D)'$$
$$x = B'C + B'D + BC'D' = B'(C + D) + BC'D'$$
$$= B'(C + D) + B(C + D)'$$
$$w = A + BC + BD = A + B(C + D)$$

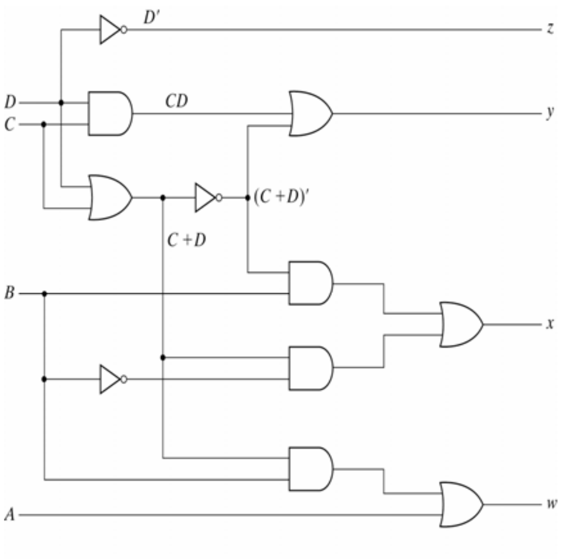


Fig. 4-4 Logic Diagram for BCD to Excess-3 Code Converter

Example

- Design a combinational circuit with three inputs and one output. The output is 1 when the binary value of the inputs is less than 3. The output is 0 otherwise.

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

Half Adder

- Sum of 2 binary inputs
- Input : X(augend), Y(addend)
- Output : S(sum), C(carry)

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

$S=xy'+x'y$

$C=xy$

Half Adder

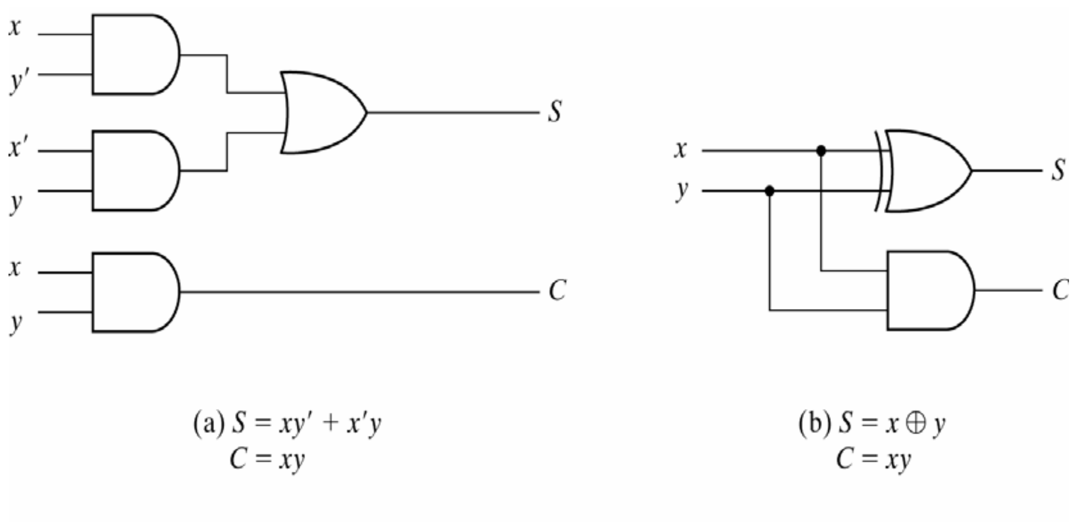


Fig. 4-5 Implementation of Half-Adder

Full adder

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

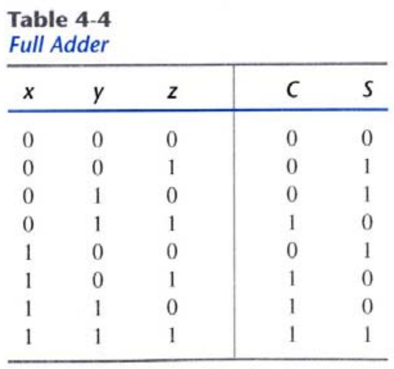


Table 4-4
Full Adder

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

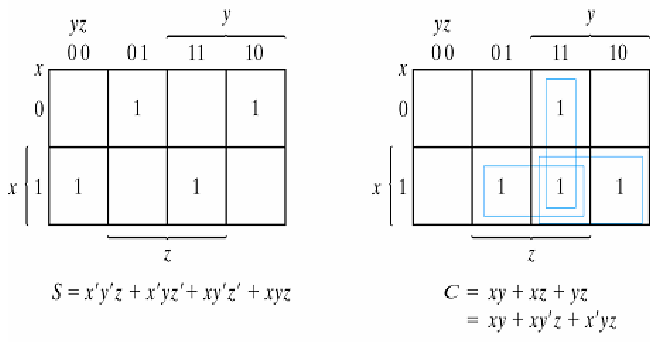


Fig. 4-6 Maps for Full Adder

Full Adder

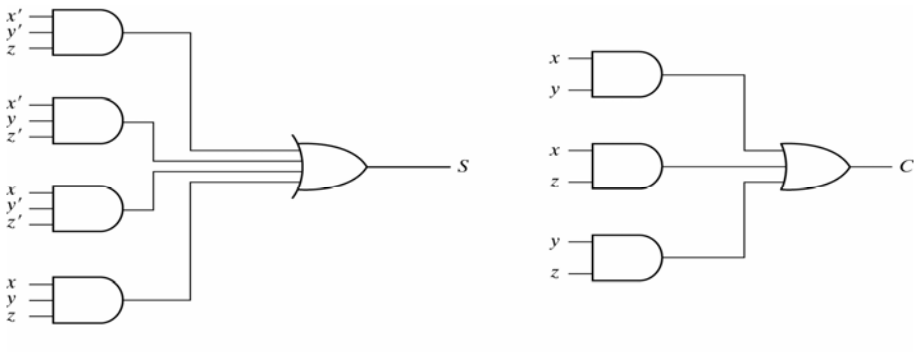


Fig. 4-7 Implementation of Full Adder in Sum of Products

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

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

Full Adder with Two Half Adders and an OR

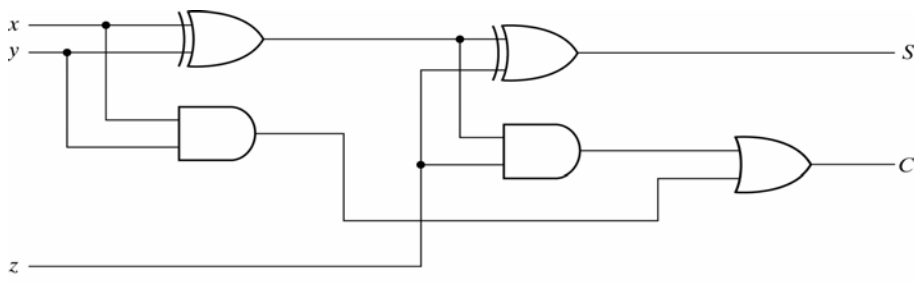
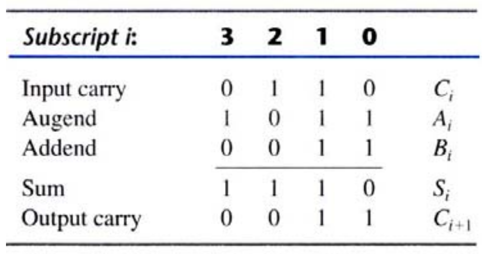


Fig. 4-8 Implementation of Full Adder with Two Half Adders and an OR Gate

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}

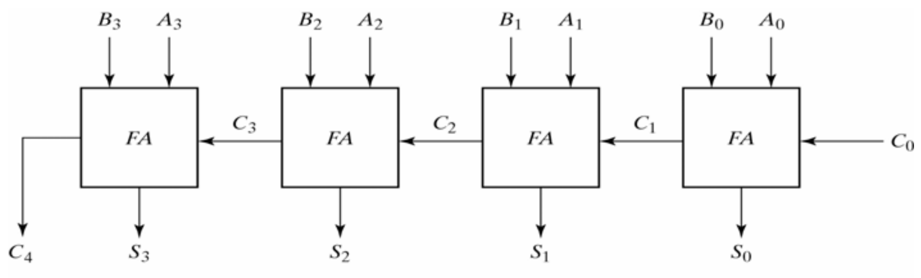
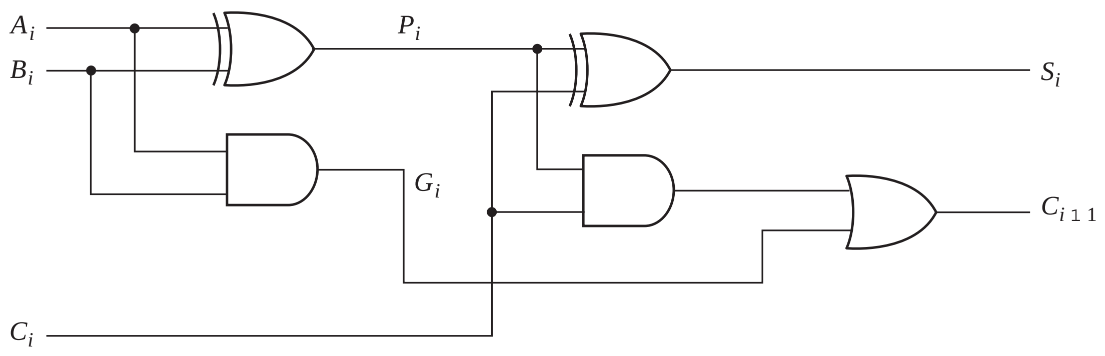


Fig. 4-9 4-Bit Adder

Binary Adder



Binary Subtractor

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

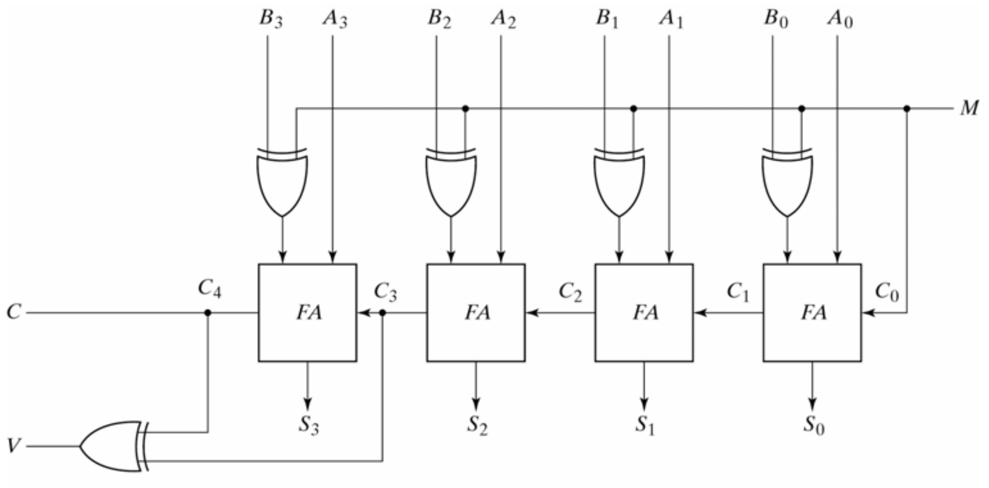


Fig. 4-13 4-Bit Adder Subtractor

Overflow

- Sum of n digit number occupies $n+1$ digit
- Always occurs when two numbers are same sign

(examples of overflow)

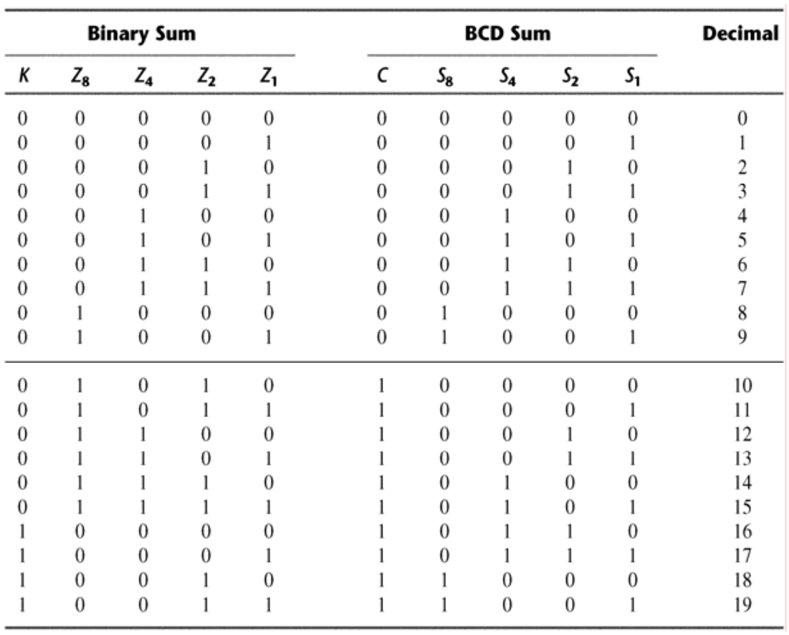
carries:	0 1		carries:	1 0	
+70	0	1000110	−70	1	0111010
+80	0	1010000	−80	1	0110000
+150	1	0010110	−150	0	1101010

Decimal Adder

- Calculate binary and represent decimal in binary coded form
- 9 inputs and 5 outputs
  - 4 bits for each decimal numbers
  - input and output carry
- Wide variety of decimal adder circuit depending on the code
- In this Chapter, decimal adder for the BCD code

Binary and BCD Sum

9(addend)+9(augend)+1(carry)=19 (Maximum)



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

BCD Sum=Binary Sum

BCD Sum
=Binary Sum+0110

BCD Adder

- BCD digit output of 2-BCD digit sum
- Correction is needed
  - K=1
  - 1010~1111
- C=K + Z_8Z_4 + Z_8Z_2

1100
1101
1110
1111

1010
1011

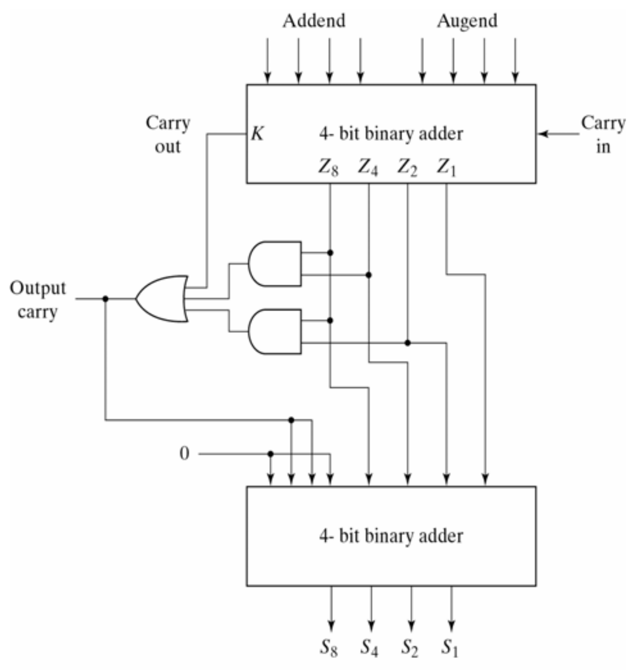


Fig. 4-14 Block Diagram of a BCD Adder

Binary Multiplier

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

		B_1	B_0
		A_1	A_0
		A_0B_1	A_0B_0
	A_1B_1	A_1B_0	
C_3	C_2	C_1	C_0

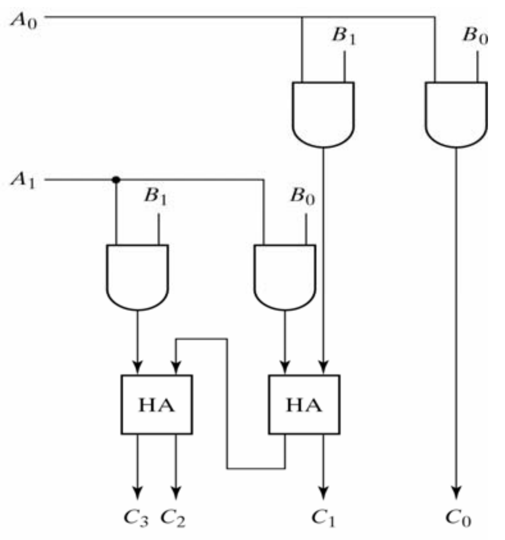


Fig. 4-15 2-Bit by 2-Bit Binary Multiplier

Binary Multiplier

- (K-bit) x (J-bit)
  - (K x J) AND gates,
  - (J-1) K-bit adder needed

$$
\begin{array}{r}
B_3B_2B_1B_0 \\
\times \quad A_2A_1A_0 \\
\hline
\end{array}
$$

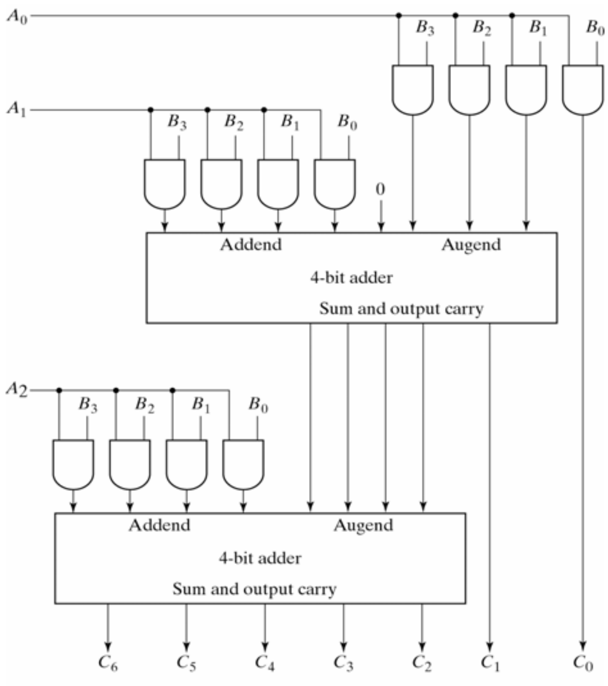


Fig. 4-16 4-Bit by 3-Bit Binary Multiplier

Magnitude Comparator

$$A = A_3A_2A_1A_0$$

$$B = B_3B_2B_1B_0$$

$$x_i = A_iB_i + A_i'B_i' \qquad for \quad i = 0,1,2,3$$

1 only if the pair of bits in i are equal

$$(A = B) = x_3x_2x_1x_0$$

$$(A > B) = A_3B_3' + x_3A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0'$$

$$(A < B) = A_3'B_3 + x_3A_2'B_2 + x_3x_2A_1'B_1 + x_3x_2x_1A_0'B_0$$

$$A = 101$$

$$B = 110$$

Compare from the most significant bit

Magnitude Comparator

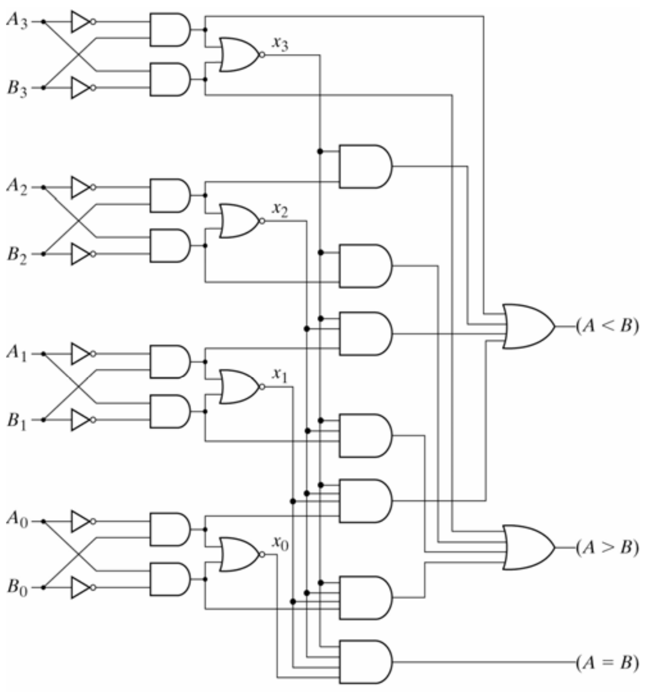


Fig. 4-17 4-Bit Magnitude Comparator

Decoders

- A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output.
- 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

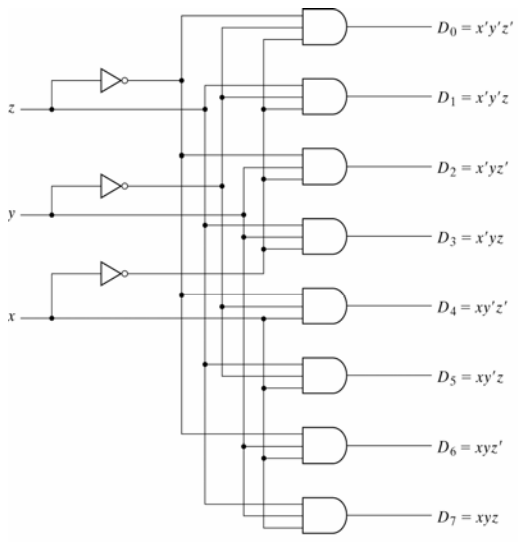
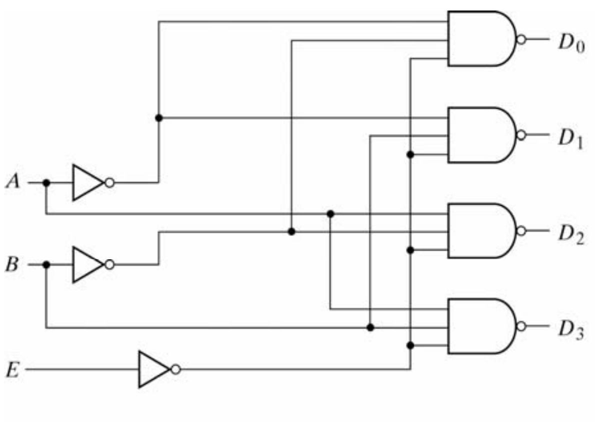


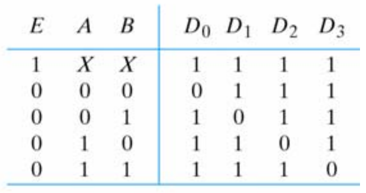
Fig. 4-18 3-to-8-Line Decoder

2-to-4-Line Decoder With Enable Input

- Operates with
  - complemented outputs
  - complemented enable input



(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

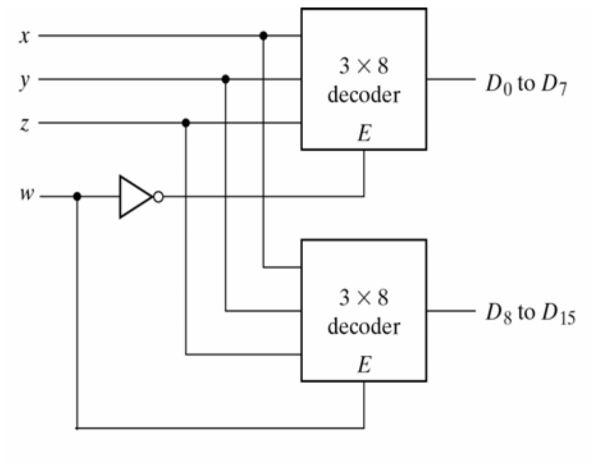
Fig. 4-19 2-to-4-Line Decoder with Enable Input

Decoder With Enable Input=Demultiplexer

- Demultiplexer
  - A circuit that receive information from a single line and directs it to one of 2^n possible output lines
- 2-to-4-line decoder with enable input = 1-to-4-line demultiplexer
  - E is taken as a data input line
  - A and B are taken as the selection inputs

Decoders with enable inputs can be a larger decoder circuit

- 4x16 decoder by two 3x8 decoders



$w = 0$
0000 ~ 0111

$w = 1$
1000 ~ 1111

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

Combinational Logic Implementation with Decoder

- Any combinational circuit can be implemented with line decoder and OR gates
  - Example) full adder

$$S(x, y, z) = \sum(1,2,4,7)$$
$$C(x, y, z) = \sum(3,5,6,7)$$

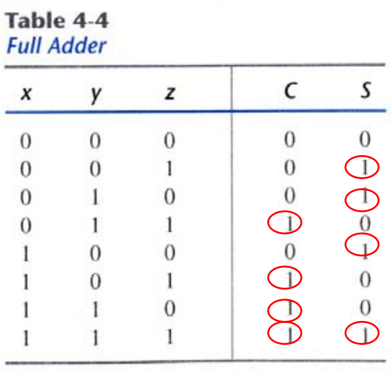


Table 4-4
Full Adder

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

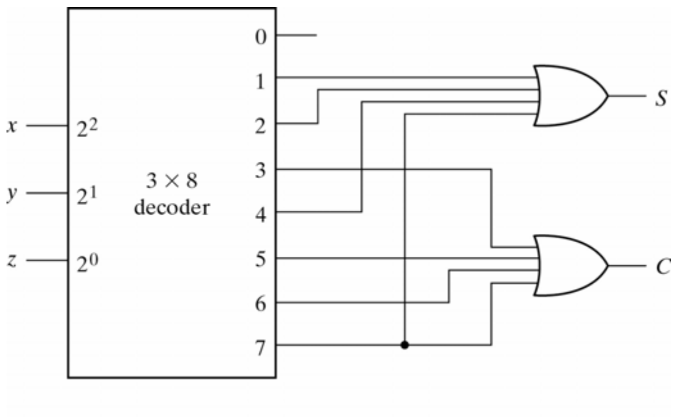


Fig. 4-21 Implementation of a Full Adder with a Decoder

Example

- A combinational circuit is defined by the following three Boolean functions:

  $F1 = x'y'z' + xz$

  $F2 = xy'z' + x'y$

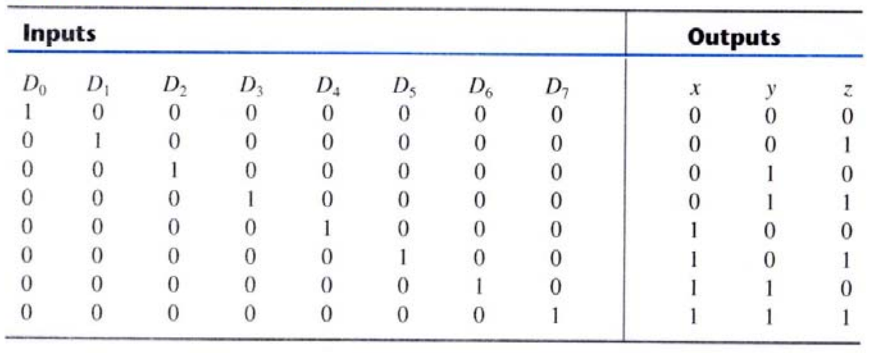
  $F3 = x'y'z + xy$

  Design the circuit with a decoder and external gates.

Encoders

- Inverse operation of a decoder
- Generate n outputs of 2^n input values
  - Ex) 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

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

Only one input can be active at any given time

Priority Encoder

- Problem happens two or more inputs equal to 1 at the same time
- Give a priority function to circuit
- V is valid bit, 1 when one or more inputs are 1, then inspect

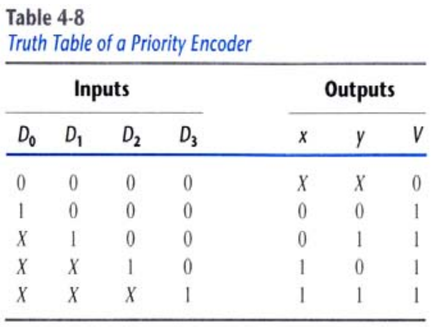


Table 4-8
Truth Table of a Priority Encoder

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

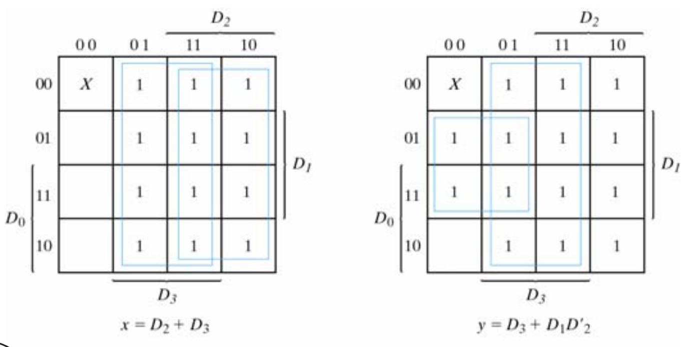
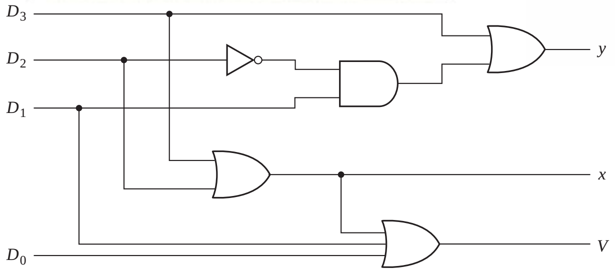


Fig. 4-22 Maps for a Priority Encoder



(x100 means 0100, 1100)

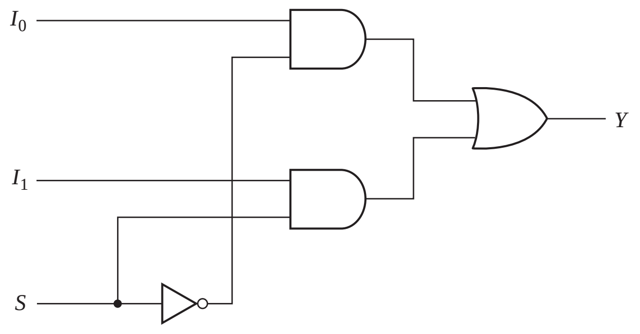
Example

- Design a 4-input priority encoder with inputs as in Table 4-8, but with input D0 having the highest priority and input D3 the lowest priority

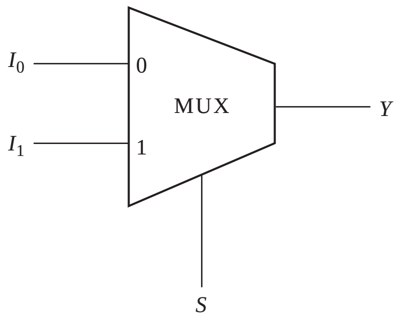
Multiplexers

- Select a binary information from many input lines
- Directs it to a single output line
- Selection is controlled by a set of selection lines
- 2^n input lines have n selection lines

2-to-1-line Multiplexer

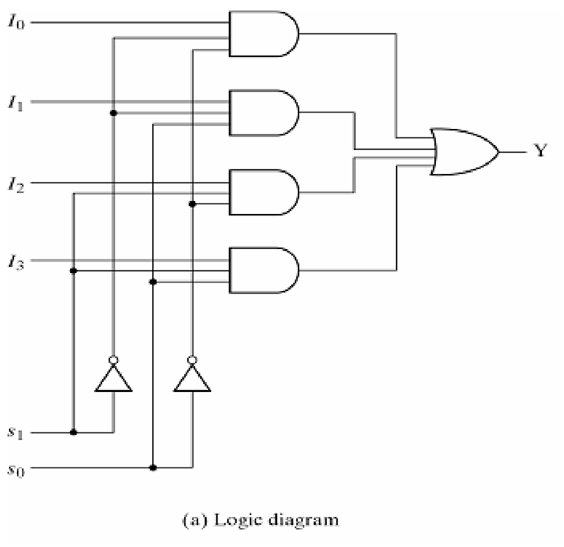


(a) Logic diagram



(b) Block diagram

4-to-1-Line Multiplexer

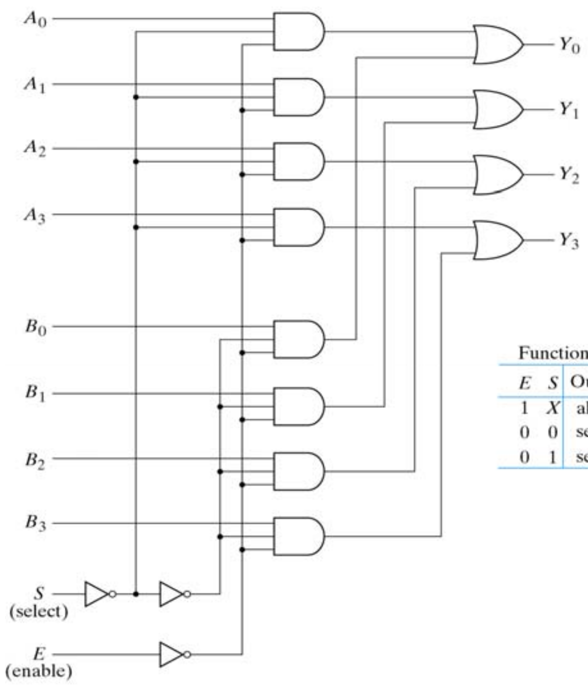


(a) Logic diagram

s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(b) Function table

Quadruple 2-to-1-Line Multiplexer



Function table

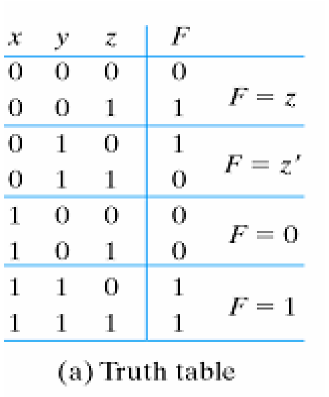
E	S	Output Y
1	X	all 0's
0	0	select A
0	1	select B

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

Boolean Function Implementation with MUX

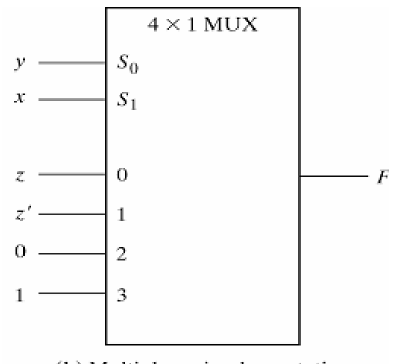
- MUX is essentially decoder includes OR gate
- Minterms of function are generated in a MUX
- *n* input variables
  - First *n-1* variables -> input of MUX
  - Remaining variable -> data inputs

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



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	

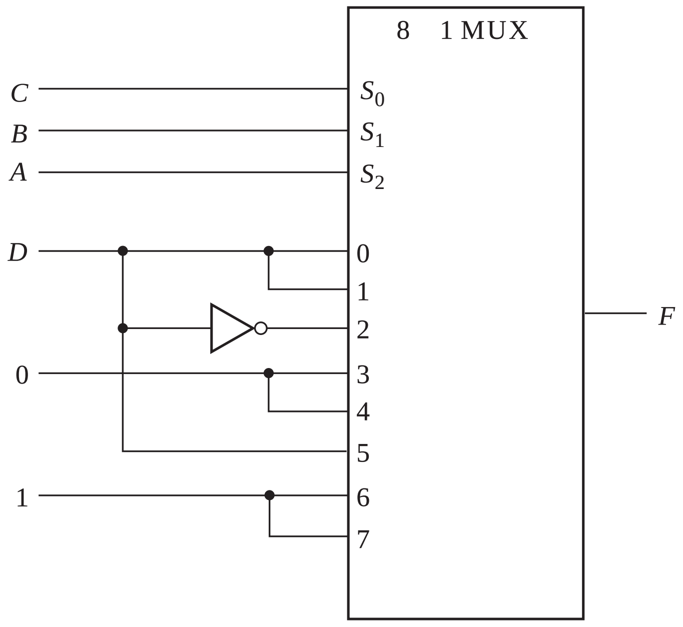
(a) Truth table



(b) Multiplexer implementation

Boolean Function Implementation with MUX

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	



Three-state Gates

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

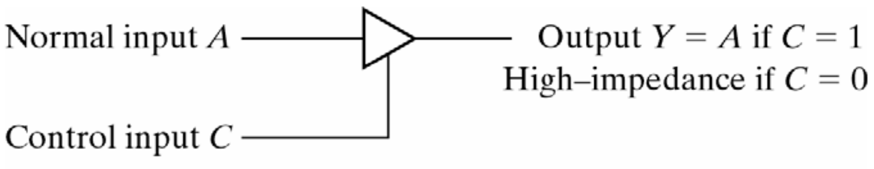
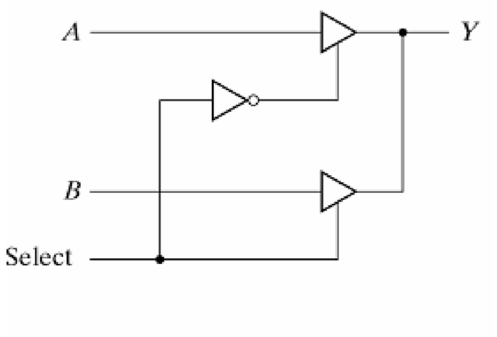


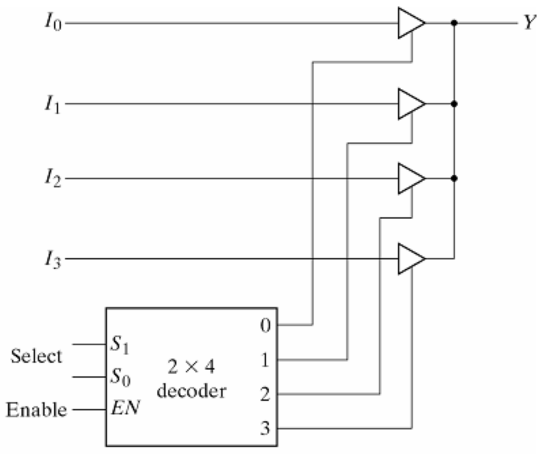
Fig. 4-29 Graphic Symbol for a Three-State Buffer

MUX with Three-state Gates

- Multiplexers can be constructed with three-state gates



(a) 2-to-1- line mux



(b) 4 - to - 1 line mux

Fig. 4-30 Multiplexers with Three-State Gates