ECE 223
Digital Circuits and Systems

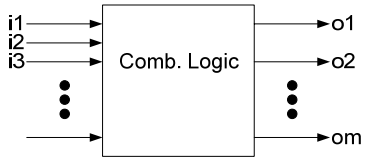
Combinational Logic

M. Sachdev,
Dept. of Electrical & Computer Engineering
University of Waterloo

1

Combinational Circuits

- consists of logic gates with outputs that are determined entirely by the present value of the inputs (no memory)
  - Combinatorial circuits might be 2-level logic (SOP,POS) or multi-level



- Two important procedure
  - **Analysis** – Given circuit schematic, explain its behavior
  - **Design** – Given the specifications, build it

2

Analysis Procedure

1. Label all logic gate outputs and primary inputs
2. Starting from primary inputs, represent outputs in terms of their input variables
3. Repeat 2, till you reach output
4. Represent primary output(s) in terms of primary inputs

$F_2 = AB + AC + BC$

$T_1 = A + B + C$

$T_2 = ABC$

$T_3 = F_2'T_1$

$F_1 = T_2 + T_3$

Simplify the F_1

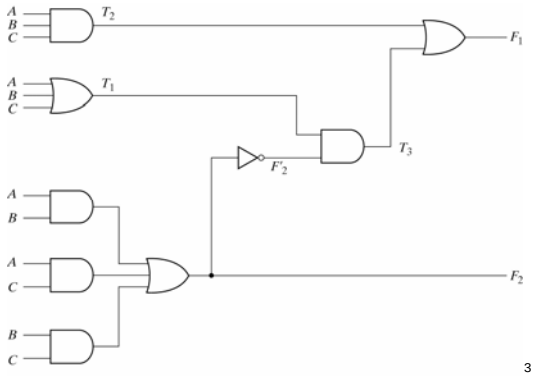


Fig. 4-2 Logic Diagram for Analysis Example

3

Design Procedure

1. From Specifications, determine the required number of inputs and outputs
2. Assign a variable to each input and output
3. Derive a truth Table that defines the required relationship between inputs and outputs
4. Perform logic minimization
5. Draw the logic diagram

4

Design Procedure - Example

- Given input bits in BCD, design a circuit to convert inputs to Excess-3 code outputs

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

5

Example – K maps

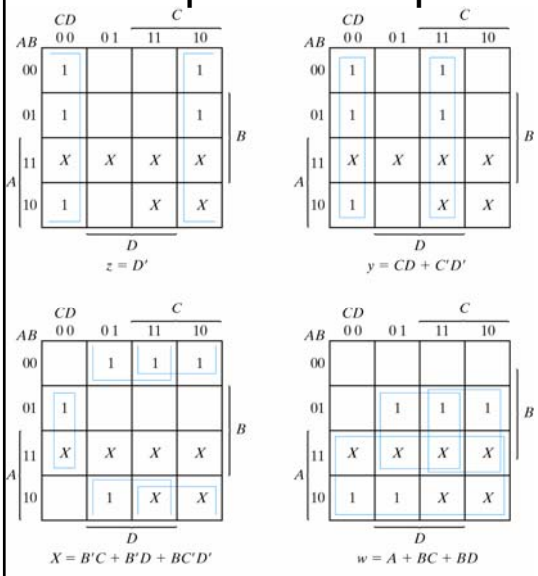


Fig. 4-3 Maps for BCD to Excess-3 Code Converter

z = D'

y = CD +C'D'

= CD + (C +D)'

x = B'C +B'D +BC'D'

= B'(C +D) +B(C +D)'

w = A +BC +BD

= A +B(C+D)

6

Example – Schematic

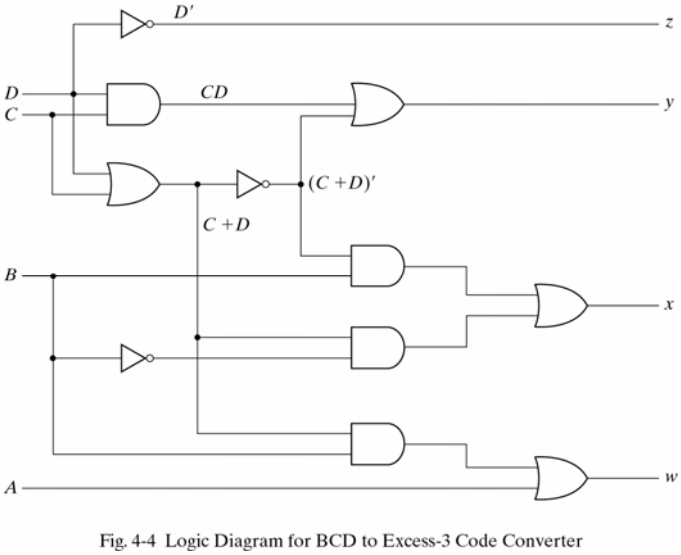


Fig. 4-4 Logic Diagram for BCD to Excess-3 Code Converter

7

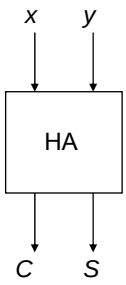
Binary Adder

- Adder is an important component in many logic circuits
- Half Adder

  $(CS)_2$ = x plus y

  S = x'y +xy' = x⊕y

  C = xy



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

8

Full Adder

$(CS)_2$ = x plus y plus z

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

9

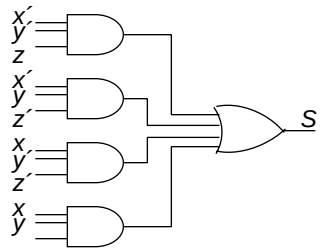
Full Adder – K Maps & Schematics

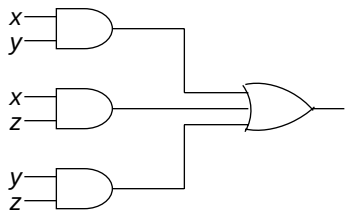
S

x \ yz	00	01	11	10
0	0	1	0	1
1	1	0	1	0

C

x \ yz	00	01	11	10
0	0	0	1	0
1	0	1	1	1



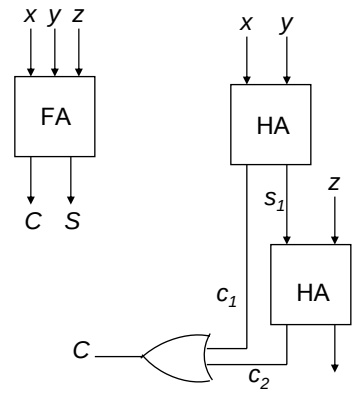


S = x'y'z +x'yz' + xy'z' + xyz = x⊕y⊕z

C = xy +xz +yz

10

Implementation with 2 Half Adders

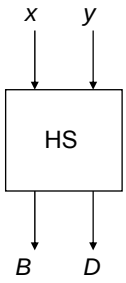


11

Half Subtractor

- Produces x -y
  - D – difference
  - B – Borrow

x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0



D = x'y +xy' = S of half adder

B = x'y

12

Full Subtractor

(x -y) –z; where z represents a borrow

x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

D = x'y'z +x'yz' + xy'z' + xyz = x⊕y⊕z = S of full adder

B = x'y +x'z +yz (same as C of full adder except x is inverted

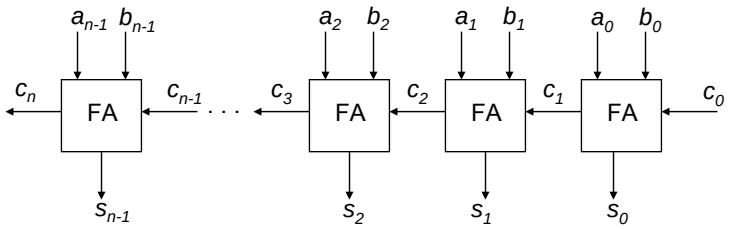
13

Parallel Adder

- Often, we must add 2 n bit numbers and carry
- Classical approach
  - (2n + 1) inputs and (n+1) outputs
  - Design a (n+1) output, 2-level (e.g. SOP) design
- Problems
  - Too many gates
  - Fan-in is too large
  - Not practical for n>3
- Use iterative circuit, reduced complexity

14

Parallel Adder – 4 Bit Adder

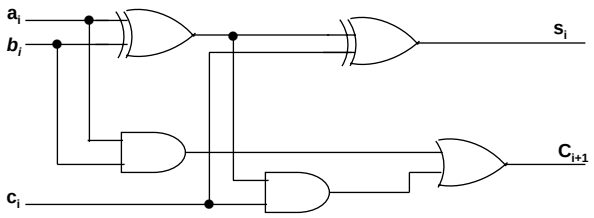


- For n = 4
  - Inputs 9
  - Outputs 5
  - Power, Gnd 2
- A 4 bit adder is accommodated in 16 pin package (MSI)

15

Parallel Adder - Issues

- Fewer gates than 2-level implementation, however significantly slower → Rippling effect of carry
- For an n bit adder
  - Propagation delay = (2xAverage gate delay)x(number of bits)
  - = 2nxd, where d = Average gate delay



16

Carry Lookahead Logic

- Define:
  - Carry Propagate

    $P_i = a_i \oplus b_i$ for $I = 0, \text{---- } n\text{-}1$
  - Carry Generate

    $G_i = a_i\, b_i$
  - Now

    $s_i = a_i \oplus b_i \oplus c_i$

    $= P_i \oplus c_i$

    $c_{i+1} = a_i\, b_i + a_i\, c_i + b_i\, c_i$

    $= a_i\, b_i + a_i\, b_i' c_i + a_i' b_i\, c_i$

    $= G_i + c_i\, P_i$

17

Carry Lookahead Logic

- Observation
  - All carries can be generated simultaneously

    $c_2 = G_1 + P_1 c_1$

    $c_3 = G_2 + P_2 c_2 = G_2 + P_2 G_1 + P_2 P_1 c_1$

    $c_4 = G_3 + P_3 c_3 = G_3 + P_3 G_2 + P_3 P_2\, G_1 + P_3\, P_2 P_1 c_1$
- Delay
  - P, G → 2 gate delay
  - c → 2 gate delay (2-level circuit)
  - S → XOR (2-level circuit)
- 6 gate delays; independent of n
  - Practically, n is limited by # of connections and gate loading

18

Carry Lookahead Logic

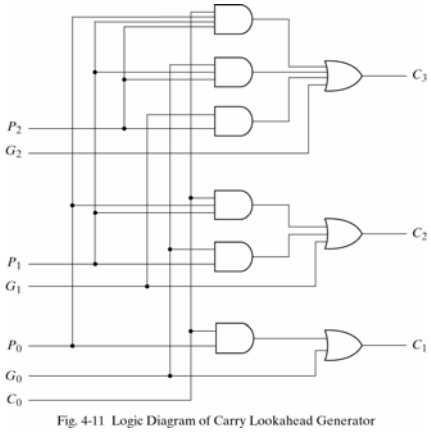


Fig. 4-11 Logic Diagram of Carry Lookahead Generator

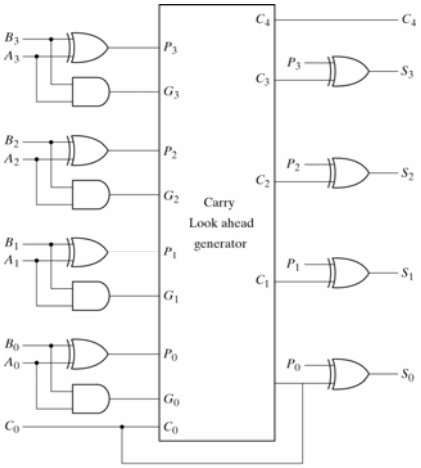


Fig. 4-12 4-Bit Adder with Carry Lookahead

19

4 Bit Adder Subtractor

- Desirable to use same hardware for both operations
  - 2's complement addition for subtraction
- Overflow
  - Addition/subtraction of 2 n bit numbers may produce n+1 bit output
  - Problem if fixed # of bits (n) are allocated for input and output
  - Must detect its occurrence

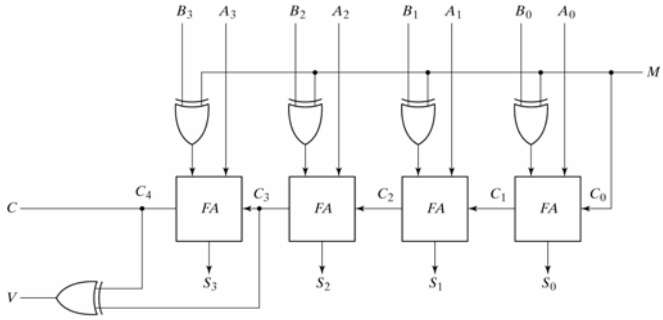


Fig. 4-13 4-Bit Adder Subtractor

20

BCD Adder

Binary Sum					BCD Sum					#
k	Z8	Z3	Z2	Z1	C	S8	S4	S2	S1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	1	1
0	0	0	1	0	0	1	0	1	0	2
0	0	0	1	1	0	1	0	1	1	3
0	0	1	0	0	0	1	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

21

Binary Sum					BCD Sum					#
k	Z8	Z3	Z2	Z1	C	S8	S4	S2	S1	
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

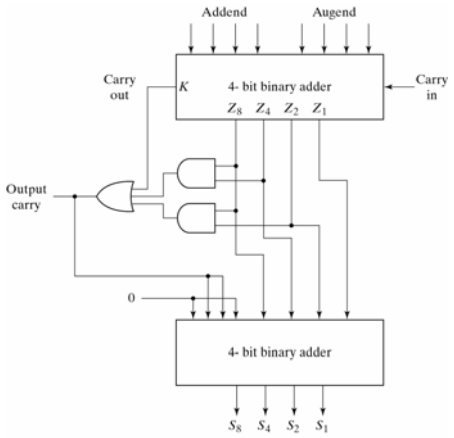
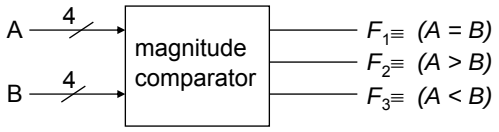


Fig. 4-14 Block Diagram of a BCD Adder

22

Magnitude Comparator



- Classical approach
  - 3 outputs and 2n inputs
  - Large logic complexity, if n >3
- Alternative Approach
- 2-step process

23

Magnitude Comparator

- 2-step process
  1. Define

     $x_i \equiv (A_i \odot B_i) = A_i B_i + A_i' B_i'$, $0 < i < 3$

  2. $F_1 = (A = B) = x_3 x_2 x_1 x_0$

$$F_2 = (A > B) = (A_3 > B_3) + (A_3 = B_3) \bullet (A_2 > B_2) + (A_3 = B_3) \bullet (A_2 = B_2) \bullet (A_1 > B_1) + (A_3 = B_3) \bullet (A_2 = B_2) \bullet (A_1 = B_1) \bullet (A_0 > B_0)$$
$$= A_3 B'_3 + x_3 A_2 B'_2 + x_3 x_2 A_1 B'_1 + x_3 x_2 x_1 A_0 B'_0$$

$$F_3 = A'_3 B_3 + x_3 A'_2 B_2 + x_3 x_2 A'_1 B_1 + x_3 x_2 x_1 A'_0 B_0$$
$$= (F_1 + F_2)'$$

24

Magnitude Comparator

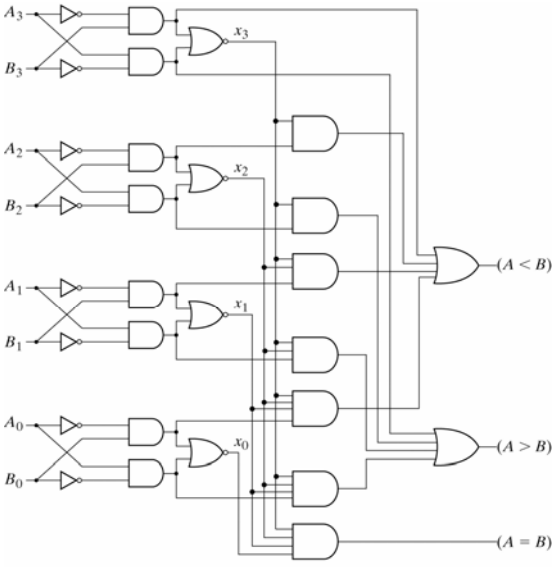


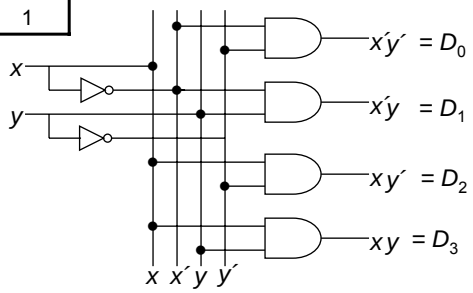
Fig. 4-17 4-Bit Magnitude Comparator

25

Decoders

- Code of n bits can represent 2^n decode outputs
  - One output is true at a time

x	y	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	0	0	1	0	0
0	1	0	0	1	0
0	1	0	0	0	1

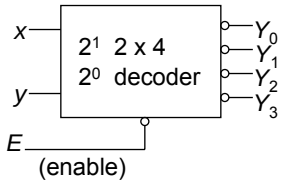


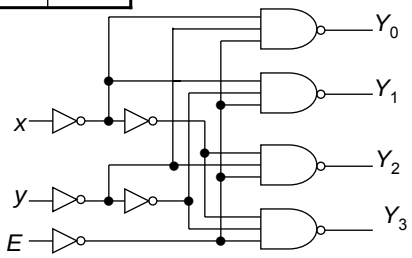
26

Decoders with Enable

- If E = 0 → all outputs are disabled
  - Outputs are active low

E	x	y	Y_0	Y_1	Y_2	Y_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	0	1	0	1	1
0	0	1	1	1	0	1
0	0	1	1	1	1	0





27

Larger Decoders

- Larger decoders are built with smaller ones

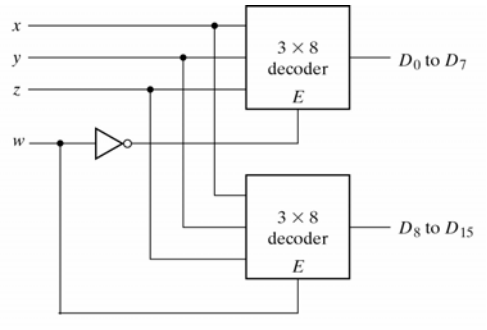


Fig. 4-20 4 × 16 Decoder Constructed with Two 3 × 8 Decoders

- Special decoders (other than n to 2^n)
  - BCD to 7 segment display

28

Function implementation using Decoders

- Adder S = ∑(1,2,4,7); C = ∑(3,5,6,7)

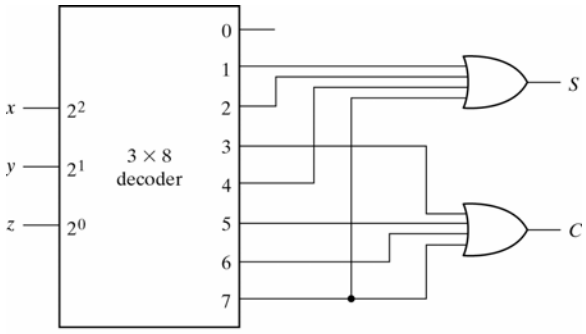


Fig. 4-21 Implementation of a Full Adder with a Decoder

29

Encoders

- Does reverse operation to decoder
- Constraint – only one input is active at a time
- Example, Octal to Binary Encoder

Z = D1 +D3 +D5 +D7; y = D2 +D3 +D6 +D7; x = D4 +D5 +D6 +D7

Inputs								Outputs		
D0	D1	D2	D3	D4	D5	D6	D7	x	y	z
1	0	0	0	0	0	1	0	0	0	0
0	1	0	0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	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	1	0	0	0	0	0	1	1	1	1

30

Priority Encoders

- Encoder with priority function
  - Multiple inputs may be true simultaneously
  - Higher priority input gets the precedence

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	0	1	1	1

- Valid bit, V, indicates two inputs are true at the same time
  - Higher priority input gets the precedence

31

Priority Encoders - Circuit

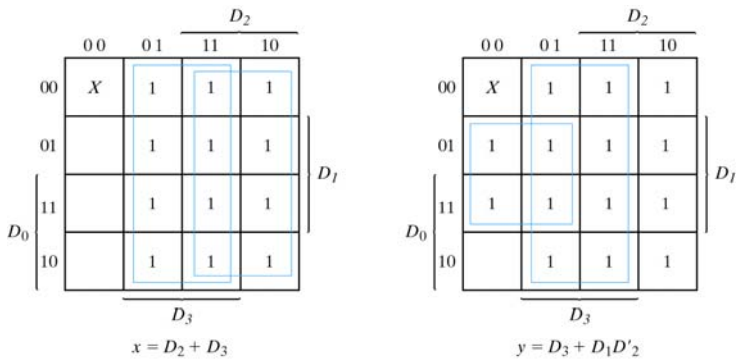


Fig. 4-22 Maps for a Priority Encoder

x = $D_2 + D_3$

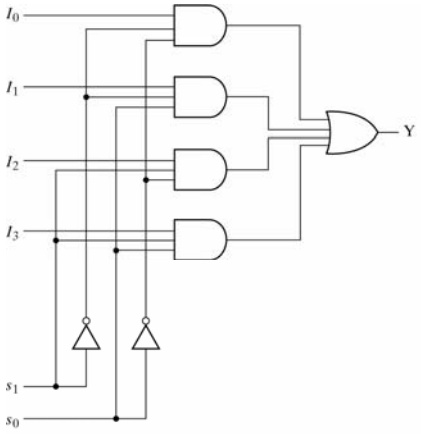
y = $D_3 + D_1D_2'$

V = $D_0 + D_1 + D_2 + D_3$

32

Multiplexers

- A Multiplexer has n inputs, one output, and S (2^s = n) select (control) inputs



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

(b) Function table

Passes (selects) an input depending on S

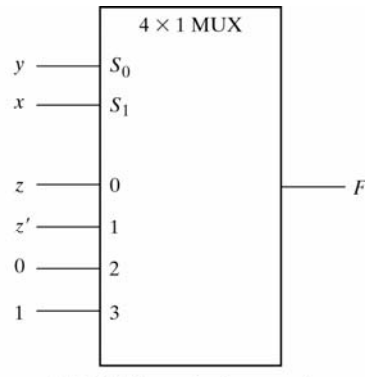
Fig. 4-25 4-to-1-Line Multiplexer

Function Implementation with Multiplexers

- Let 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	

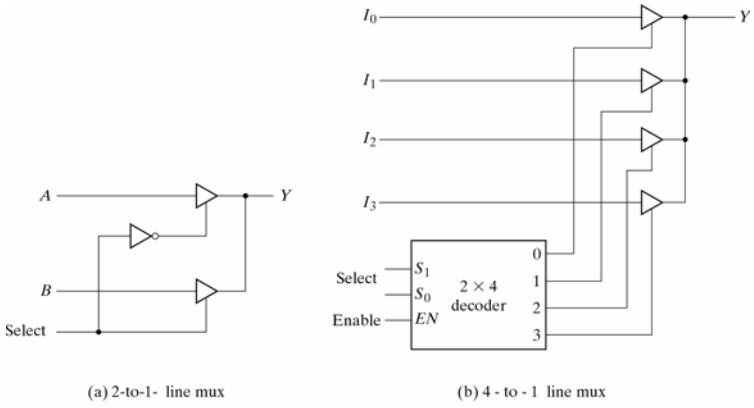
(a) Truth table



(b) Multiplexer implementation

Fig. 4-27 Implementing a Boolean Function with a Multiplexer

Multiplexers with Tri-stated Gates



(a) 2-to-1- line mux (b) 4 - to - 1 line mux

Fig. 4-30 Multiplexers with Three-State Gates

35

Book Sections – Combination Circuits

- Material is covered in Sections 4.1 – 4.5, 4.7 – 4.10

36