

## Magnitude Comparator

---

A *magnitude comparator* is a combinational circuit that compares two numbers,  $A$  and  $B$ , and then determines their relative magnitudes.

$$A > B$$

$$A = B$$

$$A < B$$

Algorithm Consider two numbers,  $A$  and  $B$ , with four digits each:

$$A = A_3A_2A_1A_0$$

$$B = B_3B_2B_1B_0$$

$$x_i = 1 \text{ if } A = B = 0 \text{ or } A = B = 1$$

$$x_i = A_iB_i + A_i'B_i' \quad \text{for } i = 0, 1, 2, 3 \text{ XNOR}$$

For equality to exist, all  $x_i$  variables must be equal to 1:

$$(A = B) = x_3x_2x_1x_0 \rightarrow \text{AND operation}$$

## Magnitude Comparator

---

To determine if  $A$  is greater than or less than  $B$ , we inspect the relative magnitudes of significant digits.

If the two digits are equal, we compare the next lower significant pair of digits. The comparison continues until a pair of unequal digits is reached.

The sequential comparison can be expressed by:

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

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

Compare:  $A = 1010$  and  $B = 0101 \rightarrow (A > B) = 1$

$A = 0101$  and  $B = 1010 \rightarrow (A < B) = 1$

## 4-bit Magnitude Comparator

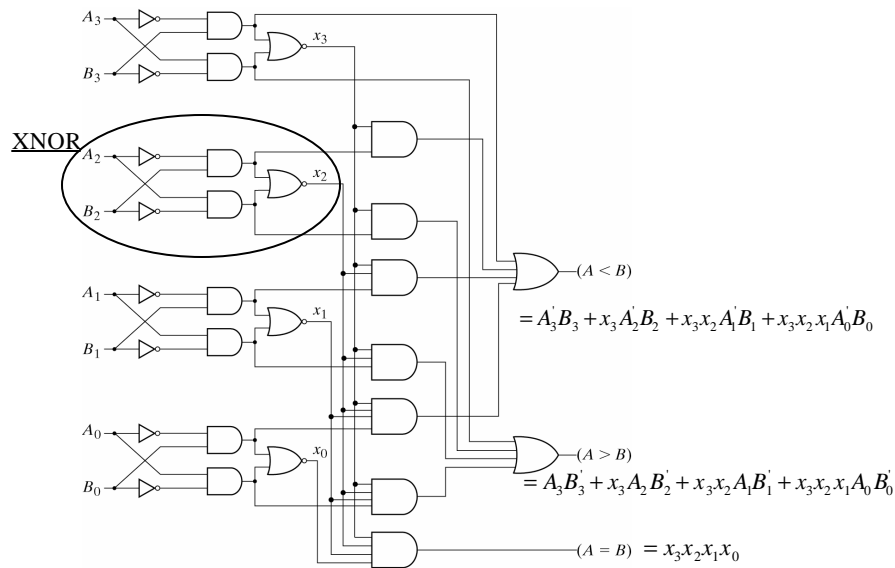


Fig. 4-17 4-Bit Magnitude Comparator

## DECODERS

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

### Some Applications:

- Microprocessor memory system: selecting different banks of memory.
- Microprocessor I/O: Selecting different devices.
- Memory: Decoding memory addresses (e.g. in ROM).
- In our lab... decoding the binary input to activate the LED segments so that the decimal number can be displayed.

## 3-to-8-line DECODER

Binary Inputs			Outputs								
			D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	D <sub>5</sub>	D <sub>6</sub>	D <sub>7</sub>	
0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	0	1

If the input corresponds to minterm  $m_i$  then the decoder output  $D_i$  will be the single asserted output.

## 3-to-8-line DECODER

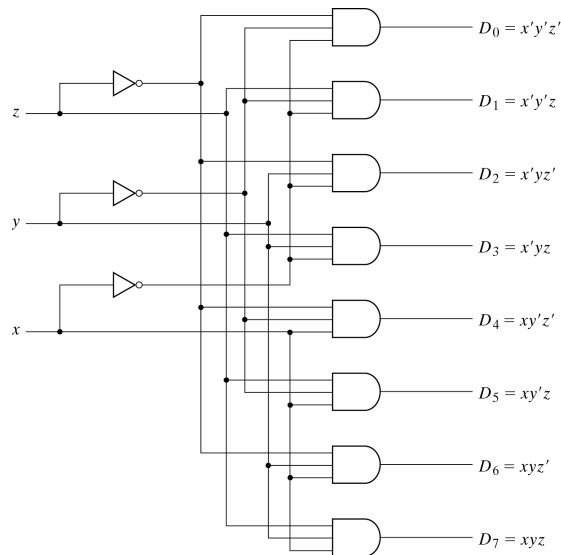


Fig. 4-18 3-to-8-Line Decoder

## 2-to-4-line DECODER with Enable

The decoder is enabled when  $E = 0$ . The output whose value = 0 represents the minterm is selected by inputs  $A$  and  $B$ .

The decoder is inactive when  $E = 1 \rightarrow D_0 \dots D_3 = 1$

A Decoder with enable input is called a decoder/demultiplexer. Demultiplexer receives information from a single line and directs it to the output lines.

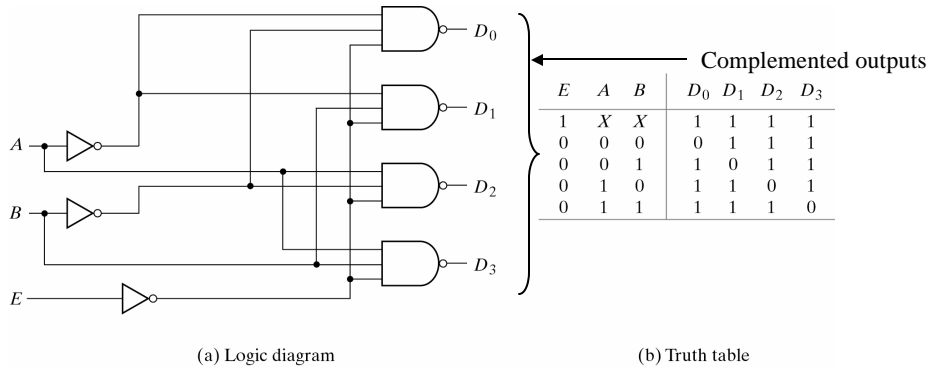


Fig. 4-19 2-to-4-Line Decoder with Enable Input

## A 4 x 16 DECODER

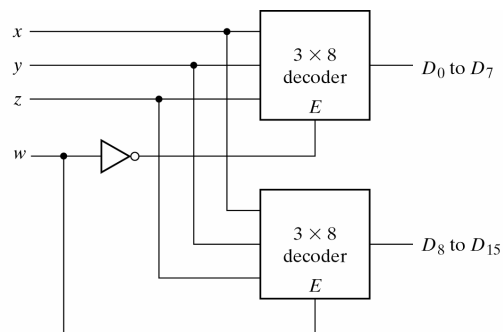


Fig. 4-20 4 x 16 Decoder Constructed with Two 3 x 8 Decoders

- When  $w = 0$ , the top decoder is enabled and the bottom is disabled. Top decoder generates 8 minterms 0000 to 0111, while the bottom decoder outputs are 0's.
- When  $w = 1$ , the top decoder is disabled and the bottom is enabled. Bottom decoder generates 8 minterms 1000 to 1111, while the top decoder outputs are 0's.

## Full-Adder using Decoder

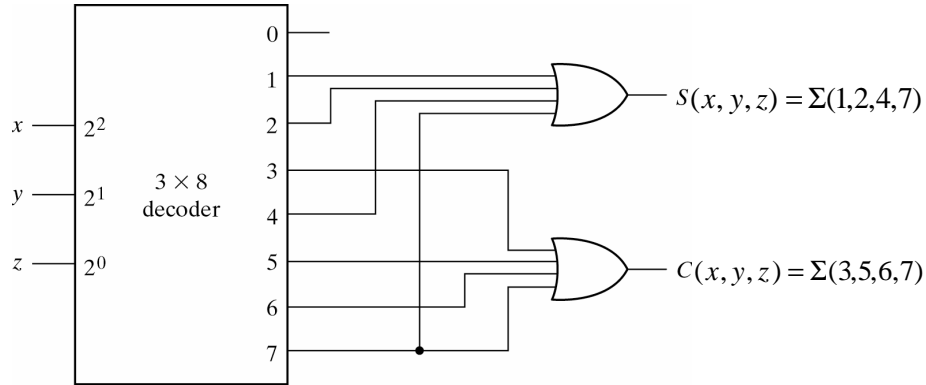


Fig. 4-21 Implementation of a Full Adder with a Decoder

## MULTIPLEXERS/DATA SELECTORS

A multiplexer is a combinational circuit that selects one of many input lines ( $2^n$ ) and steers it to its single output line. There are ( $2^n$ ) and  $n$  selection lines whose bit combinations determine which input is selected.

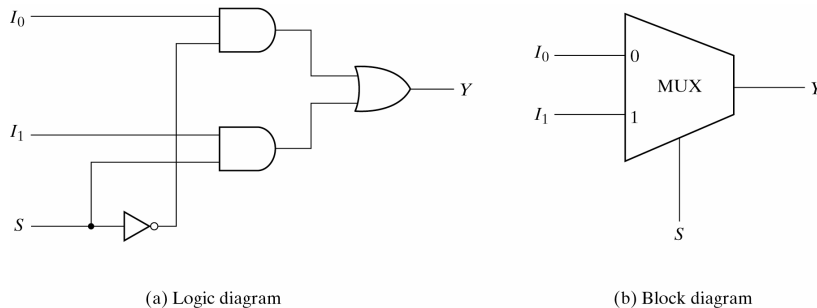


Fig. 4-24 2-to-1-Line Multiplexer

## 4-to-1 LINE MULTIPLEXER DESIGN

In general, a  $2^n$ -to-1-line multiplexer is constructed from an  $n$ -to- $2^n$  decoder by adding to it  $2^n$  lines, one to each AND gate.

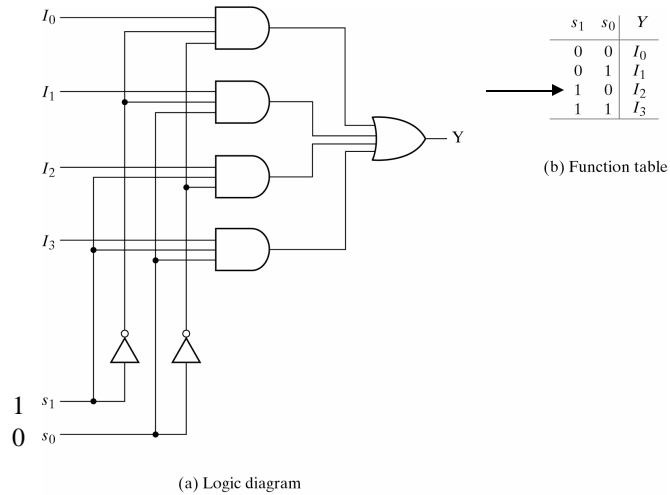
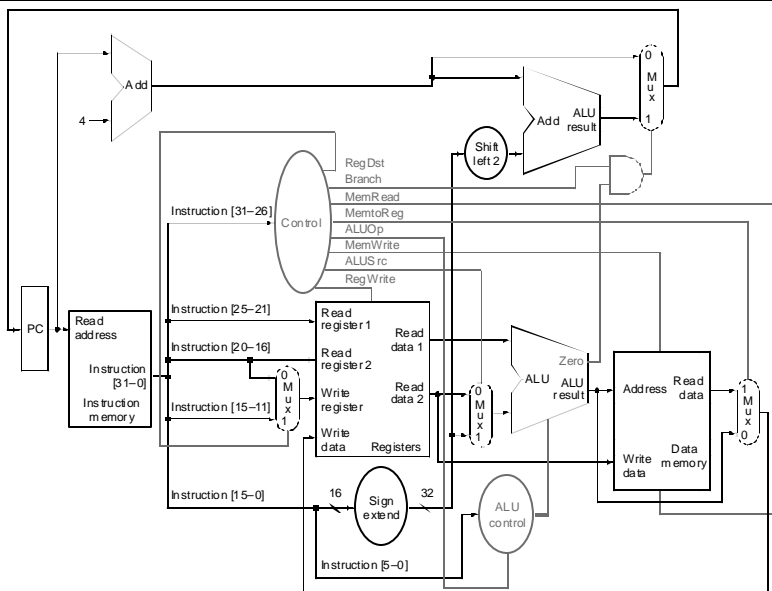


Fig. 4-25 4-to-1-Line Multiplexer

## Simple Datapath



## QUADRUPLE 4-to-1LINE MULTIPLEXER

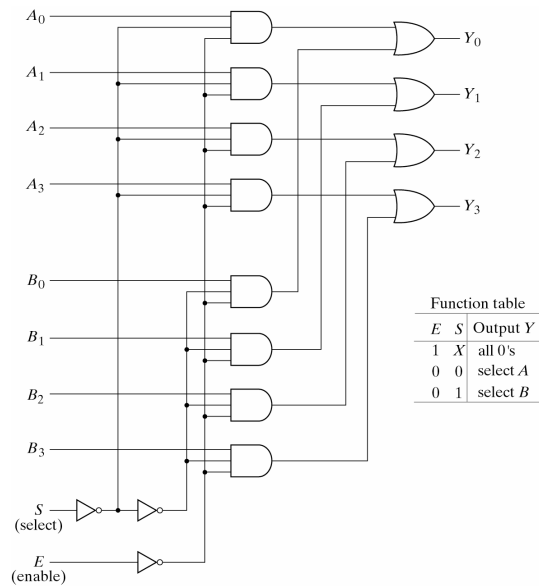


Fig. 4-26 Quadruple 2-to-1-Line Multiplexer

## Function implementation using multiplexers

Function with  $n$  variables and multiplexer with  $n - 1$  selection

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

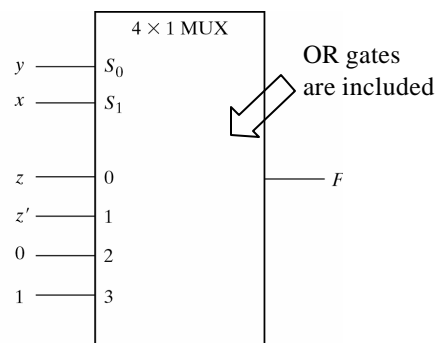
Input variables  $x, y$ : Selection lines,  $S_1$  and  $S_0$

Variable  $z$ : Data line 0

Data lines 1, 2, 3:  $z', 0, 1$

<i>x</i>	<i>y</i>	<i>z</i>	<i>F</i>
0	0	0	0
0	0	1	$F = z$
0	1	0	1
0	1	1	$F = z'$
1	0	0	0
1	0	1	$F = 0$
1	1	0	1
1	1	1	$F = 1$

(a) Truth table



(b) Multiplexer implementation

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

## Function implementation using 4x1 multiplexer

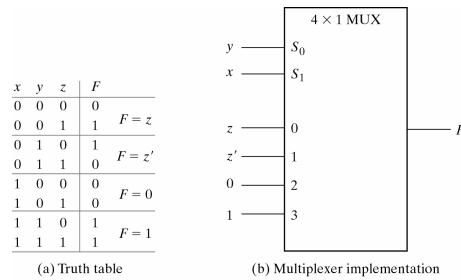


Fig. 4-27 Implementing a Boolean Function with a Multiplexer

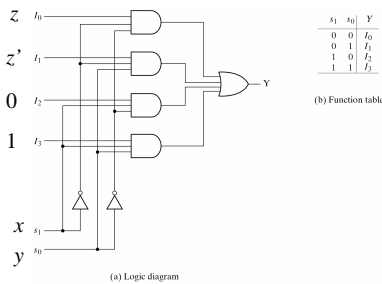


Fig. 4-25 4-to-1-Line Multiplexer

## Function implementation using 8x1 multiplexer

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

1. Complete the truth table from the SOP.
2. The first  $n - 1$  variables in the table are applied to the selection inputs of the multiplexer.
3. For each combination of the selection variables, we evaluate the output as a function of the last variable.
4. Apply these values to the data input in proper order.

## Function implementation using 8x1 MUX

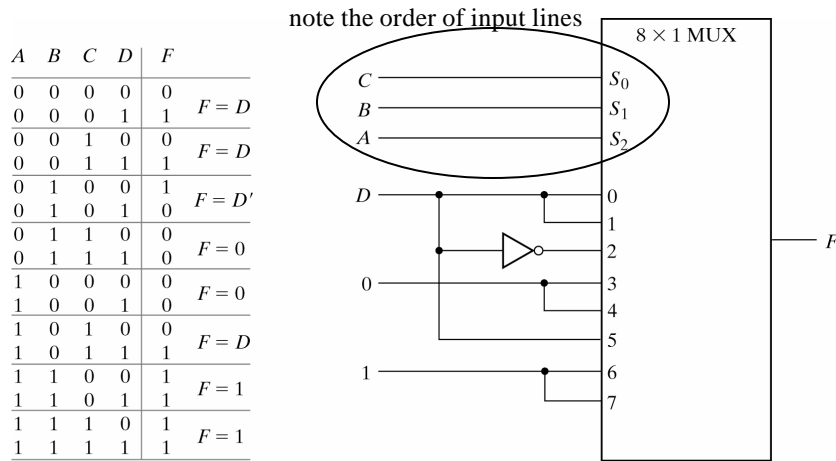


Fig. 4-28 Implementing a 4-Input Function with a Multiplexer

## Three State Gates

A three-state gate is a digital circuit that exhibits three states: 0, 1 and a high-impedance (high  $z$  state). The high impedance state behaves as an open circuit.

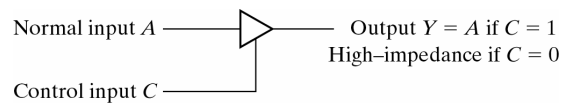


Fig. 4-29 Graphic Symbol for a Three-State Buffer

Because of this feature (high  $z$  state), a large number of three-state gate outputs can be connected to form a common line without endangering load effects.

## Multiplexers with Three State Gates

Note that the two output connections cannot be done with other gates.

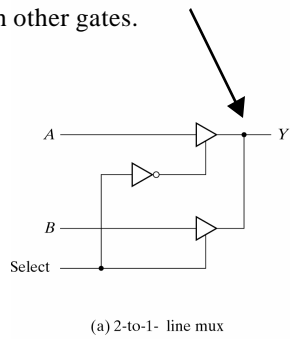


Fig. 4-30 Multiplexers with Three-State Gates