CSC 211

- LESSON 2 Magnitude Comparator

- Dr. Ronoh K. R.
- DEPARTMENT OF COMPUTER SCIENCE
- KIBABII UNIVERSITY

INTRODUCTION

- A circuit that compares two numbers, A and B, and determines their relative magnitudes.

- **1-bit Comparator**
  - For this case, it is simply the X-NOR function:

    - So the output $f = AB + A'B'$
    - In $A = B = 0$ or if $A = B = 1$
    - In addition to the equality relation, the outcome must indicate whether A > B, or A < B:

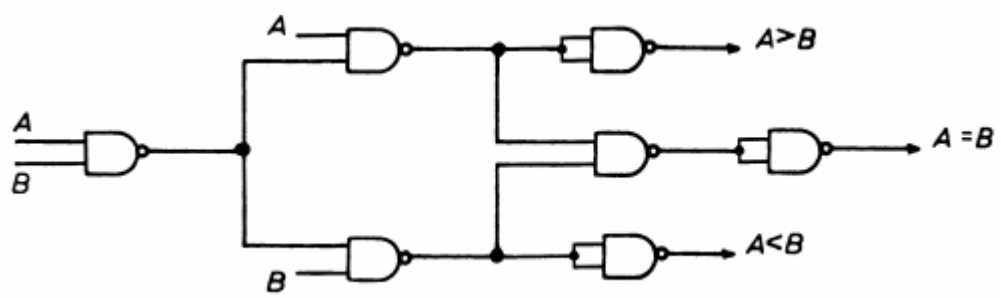
INTRODUCTION

From the table it is easy to show that:

$$A < B = A'B$$
$$A > B = AB'$$
$$A = B = A'B' + AB$$

A	B	$A < B$	$A = B$	$A > B$
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

with the following NAND-gate realization:



4-bit Comparator

- In this case an algorithm is required. Let the words be:

$$A = A_3 A_2 A_1 A_0$$

$$B = B_3 B_2 B_1 B_0$$

- The equality relation of each pair of bits can be expressed logically with the X-NOR function as:

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

- where $x_i = 1$ only if the pair of bits in position i are equal (i.e., if both are 1 or both are 0).

4-bit Comparator

- For the equality condition to exist, all xi variables must be equal to 1.
- This dictates an AND operation of all variables:

$$(A = B) = x_3 x_2 x_1 x_0$$

4-bit Comparator

- To determine whether A > B or A < B, examine the relative magnitudes of pairs of significant digits starting from the most significant position:
- If the two digits are equal, compare the next lower significant pair of digits. The comparison continues until a pair of unequal digits is found.
- If the corresponding digit of A is 1 and that of B is 0, conclude that A > B.

4-bit Comparator

- If the corresponding digit of A is 0 and that of B is 1, conclude that A<B.
- The sequential comparison can be expressed logically by the two Boolean functions:

$$(A > B) = A_3 B_3' + x_3 A_2 B_2' + x_3 x_2 A_1 B_1' + x_3 x_2 x_1 A_0 B_0'$$

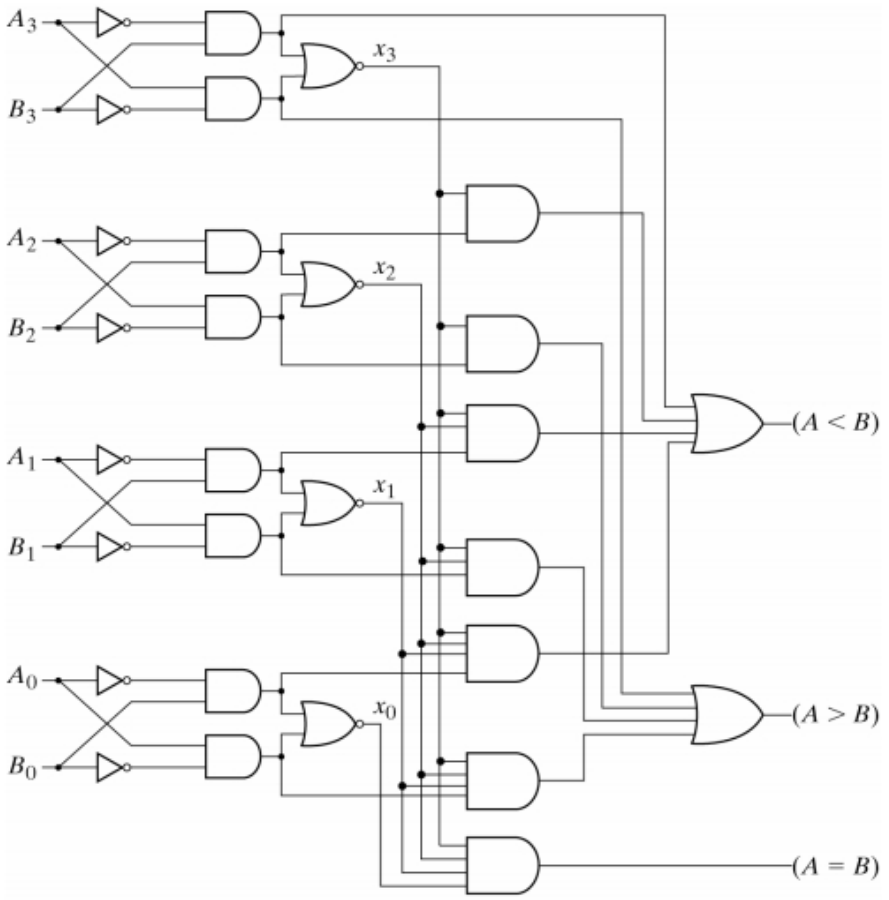
$$(A < B) = A_3' B_3 + x_3 A_2' B_2 + x_3 x_2 A_1' B_1 + x_3 x_2 x_1 A_0' B_0$$

4-bit Comparator

- The symbols (A > B) and (A < B) are binary outputs that are equal to 1 when A > B or A < B, respectively.

- Finally, the logic diagram of the 4-bit magnitude comparator is as follows:

4-bit Comparator



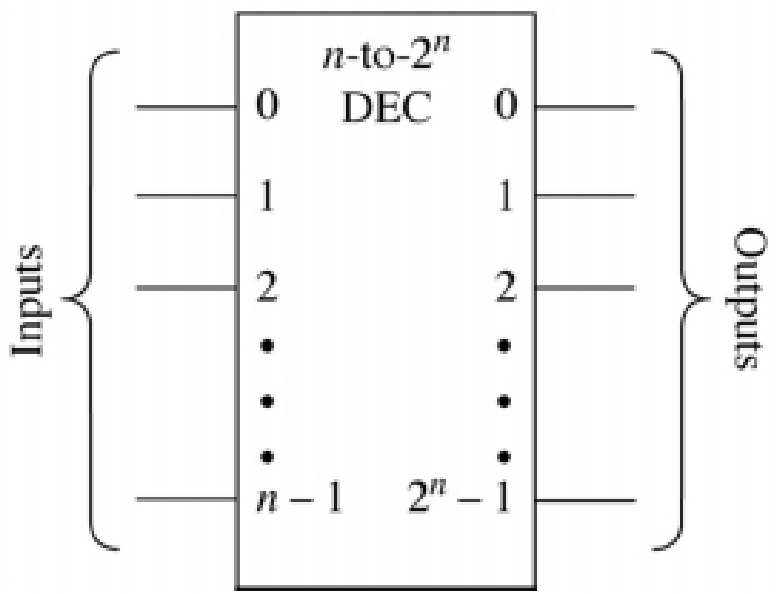
4-bit Comparator

- The four x outputs are generated with X-NOR circuits and applied to an AND gate to give the output binary variable (A = B).
- The other two outputs use the x variables to generate the Boolean functions shown before.

Decoders

- Often, digital information represented in dome binary form must be converted into some alternative digital form.
- This is achieved by a multiple-input, multiple output network referred to as a decoder.
- The most commonly used decoder is the n-to-2^n-line decoder:

Decoders



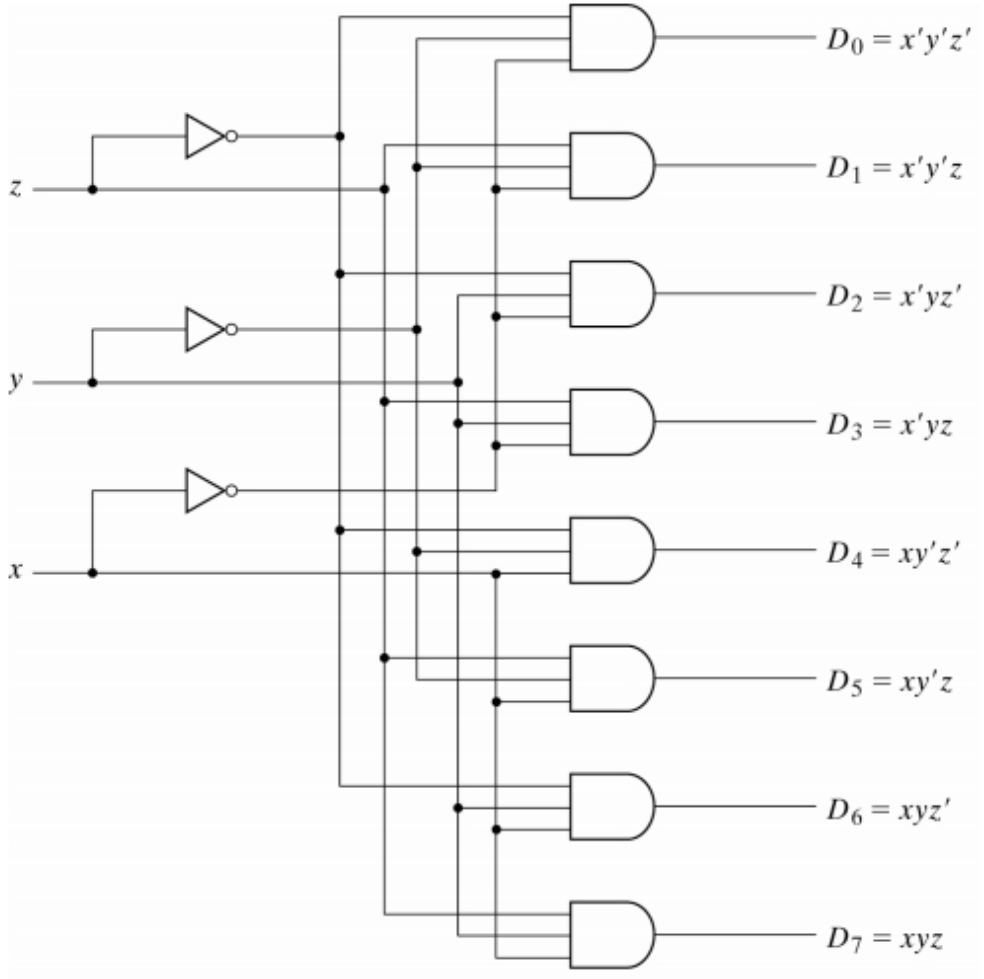
Decoders

- The structure of a such decoder is straightforward.
- Consider the 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

Decoders

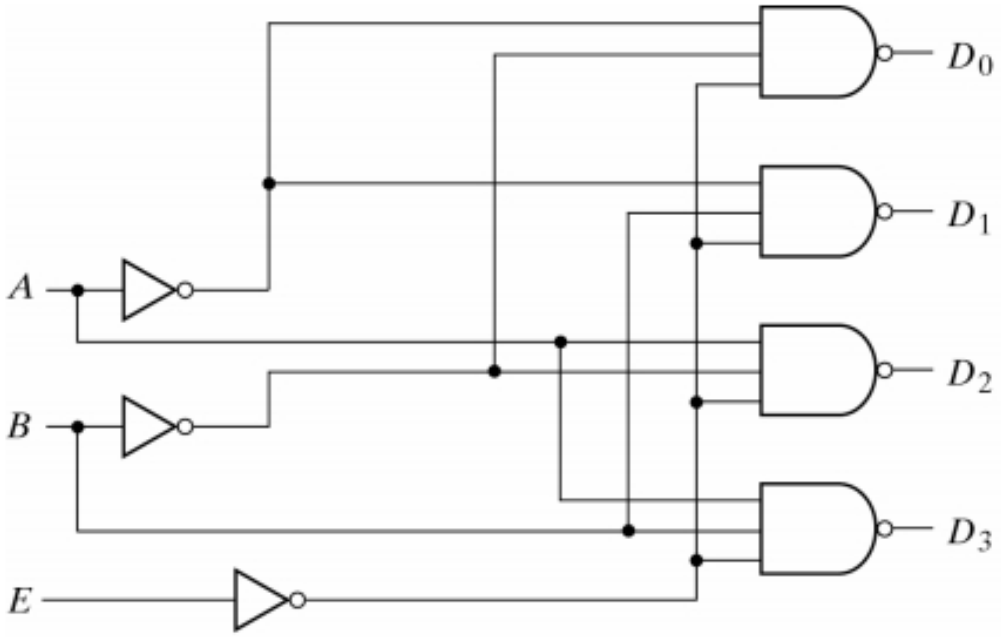
- This corresponds to the logic diagram shown below:



- A particular application for this decoder is binary-to-octal conversion.
- The input variables represent a binary number, and the outputs represent the eight digits in the octal number system.

Decoders with an Enable Input

- Some decoders include one or more enable inputs to control the circuit operation.
- The logic diagram and truth table of a 2-to-4-line decoder are shown below:



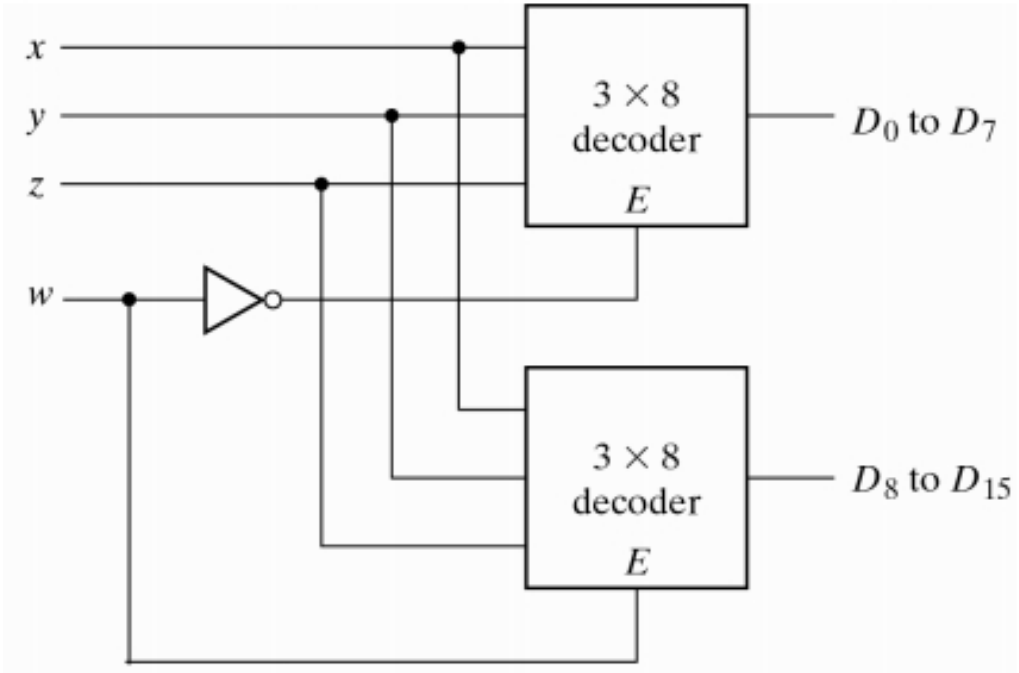
Decoders with an Enable Input

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

A decoder with enable input can function as a demultiplexer. The above decoder can function as a 4-to-1-line demultiplexer when E is taken as a data input line and A and B are taken as the selection inputs.

Decoders with an Enable Input

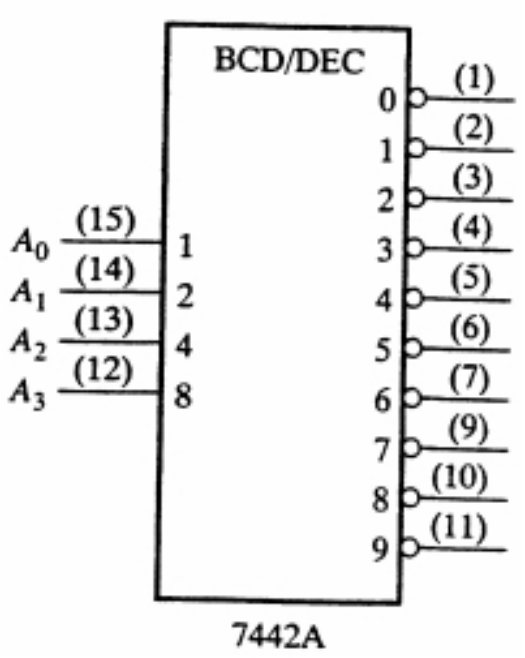
- Decoders with enable inputs can be connected together to form a larger decoder circuit.
- A 4-to-16-line decoder realized using two 3-to-8-line decoders is shown beside:

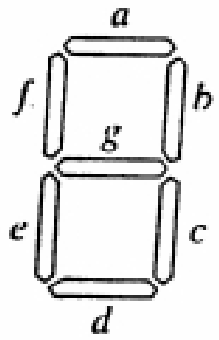


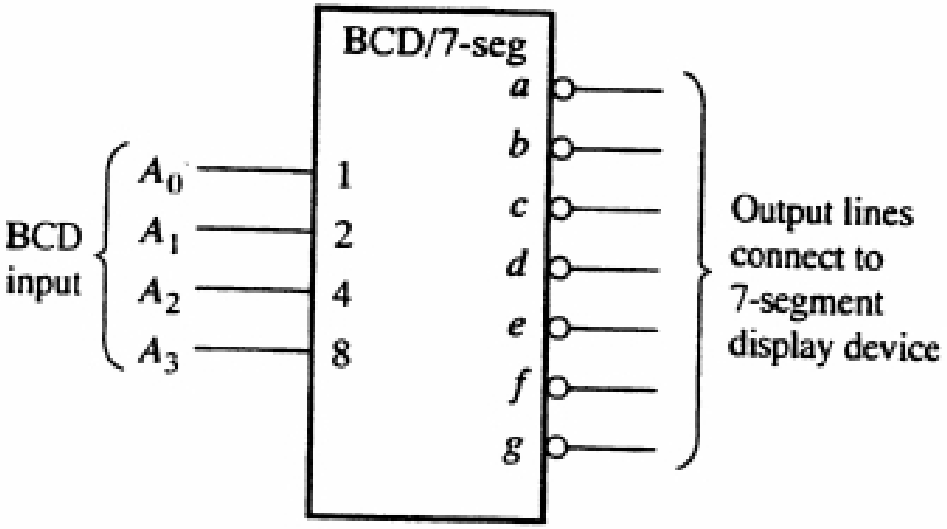
Decoders with an Enable Input

- When w = 0, the top decoder is enabled and the other is disabled.
- The bottom decoder outputs are all 0's, and the top eight outputs generate minterms 0000 to 0111.
- When w = 1, the enabled conditions are reversed; the bottom decoder generates minterms 1000 to 1111, while the outputs of the top decoder are all 0's.
- The n-to-2^n-line decoder is only one of several types of decoders. Function-specific decoders exist having fewer than 2^n outputs.
- Examples include the BCD-to-decimal decoder (7442A) and the BCD-to-7-segment decoder.

Decoders with an Enable Input







Combinational Logic Implementation

- An n-to-2^n-line decoder is a minterm generator.
- Recall that any Boolean function is describable by a sum-of-minterms. Thus, by using OR-gates in conjunction with an n-to-2^n-line decoder realizations of Boolean functions are possible.
- However, these realizations do not correspond to minimal sum-of-products.

Combinational Logic Implementation

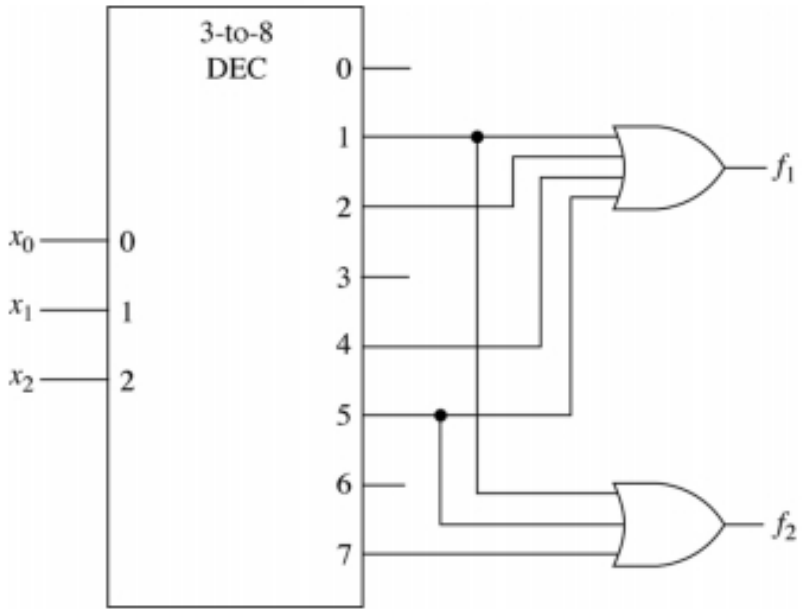
- Consider the pair of expressions:

$$f_1(x_2, x_1, x_0) = \sum(1, 2, 4, 5)$$

$$f_2(x_2, x_1, x_0) = \sum(1, 5, 7)$$

Combinational Logic Implementation

- Using a single 3-to-8-line decoder and two OR-gates, the following realization is obtained:



Combinational Logic Implementation

- When more than ½ the total number of minterms must be OR-ed, it is usually more economical to use NOR-gates rather than OR-gates to do the summing. Consider the pair of expressions:

$$f_1(x_2, x_1, x_0) = \sum (0, 1, 3, 4, 5, 6)$$

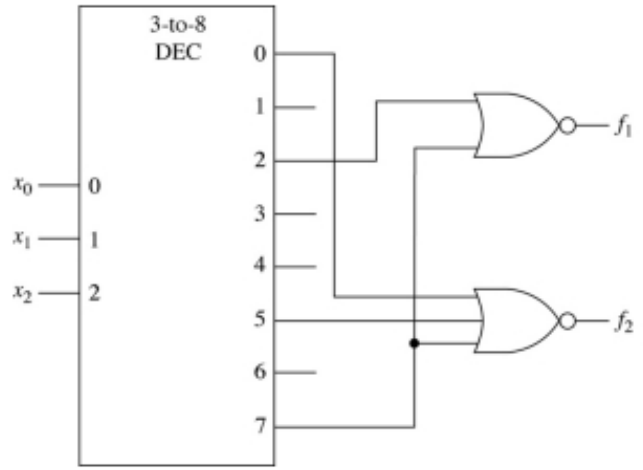
$$f_2(x_2, x_1, x_0) = \sum (1, 2, 3, 4, 6)$$

- These may be realized with a 3-to-8-line decoder and two OR-gates having a total of 11 terminals between them. However, a more efficient realization is to re-write the expressions as:

$$f_1''(x_2, x_1, x_0) = f_1'(x_2, x_1, x_0) = \overline{\sum(2, 7)}$$

$$f_2''(x_2, x_1, x_0) = f_2'(x_2, x_1, x_0) = \overline{\sum(0, 5, 7)}$$

- This corresponds to the realization shown below:



- A total of five gate-input terminals are needed.

Encoders

- Perform the inverse operation of decoders. An encoder has 2^n (or fewer) input lines and n output lines.
- The output lines generate the binary code corresponding to the input value.

- An example of an encoder is the octal-to-binary encoder whose truth table is as follows:

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

- The equations for the three outputs are:

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

- The encoder can be realized with three OR-gates.

Priority Encoder

- The encoder defined before has the limitation that only one input can be active at any given time.
- If two inputs are active simultaneously, the output produces an undefined combination.
- This is resolved by establishing an input priority function.

Priority Encoder

- The truth table of a four-input priority encoder is:

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

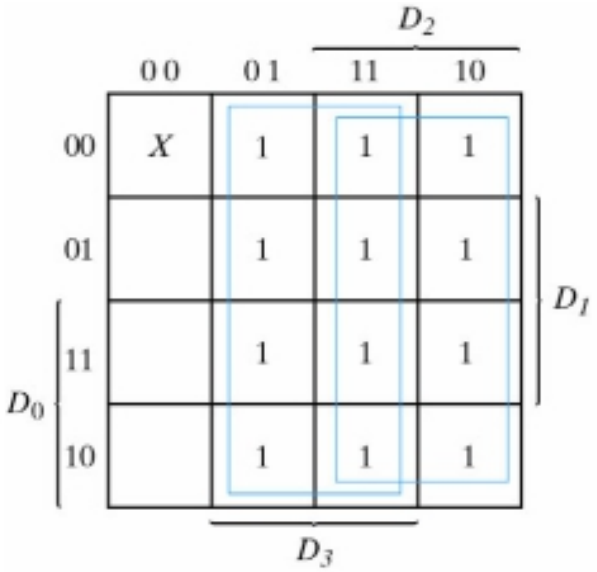
- In addition to the two outputs, x and y, the circuit has a third output V; this is a valid bit indicator and is set to 1 when one or more inputs are equal to 1.

Priority Encoder

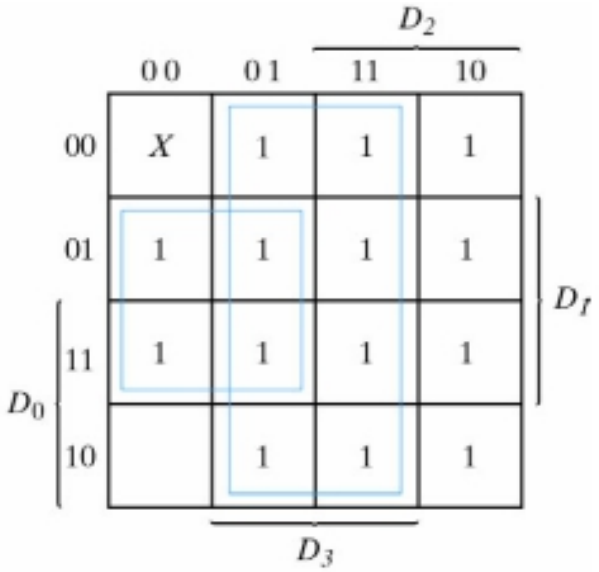
- X's in the output represent don't-care conditions.
- X's in the input columns are for representing the truth table in condensed form. Instead of listing all 16 minterms of four variables, the truth table uses an X to represent either 1 or 0.
- According to the table, D_3 has the highest priority followed by D_2 and D_1.

Priority Encoder

- The maps for simplifying outputs x and y are shown below:



$x = D_2 + D_3$



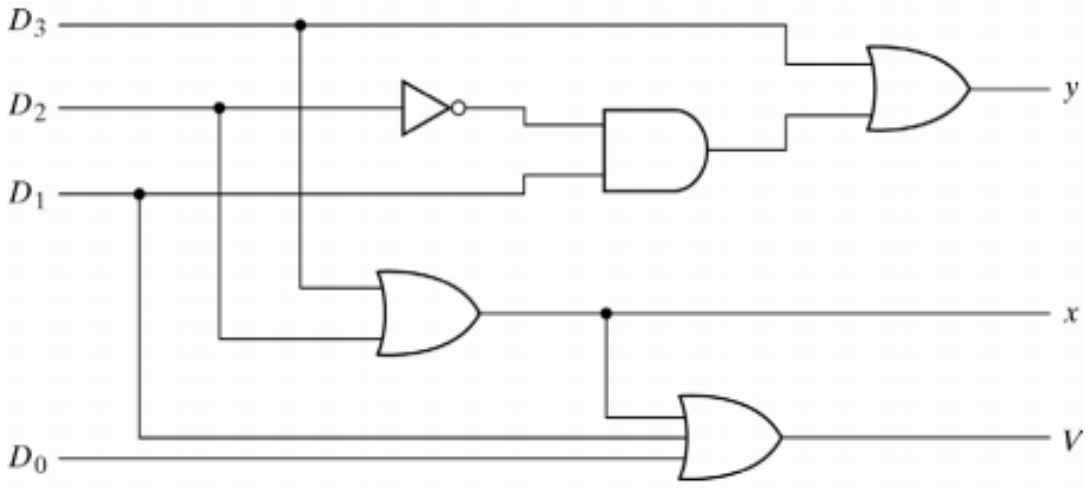
$y = D_3 + D_1D'_2$

Priority Encoder

- The condition for output V is an OR function of all the input variables:

$$V = D_0 + D_1 + D_2 + D_3$$

- The priority encoder is implemented as follows:

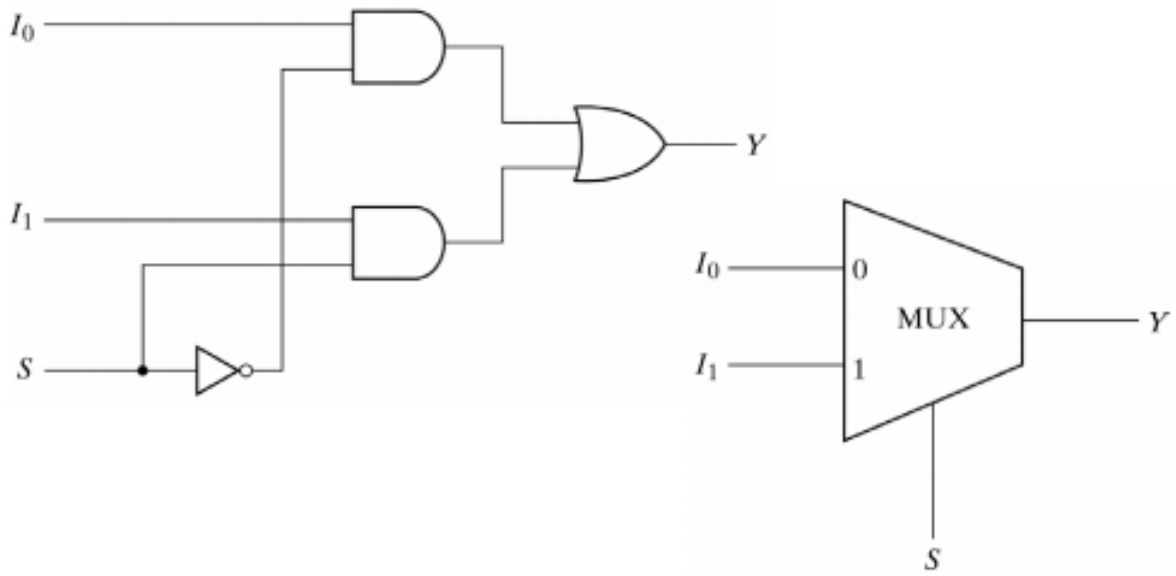


Multiplexers

- A multiplexer is a circuit that selects binary information from one of many input lines and directs it to a single output. Normally, there are 2^n input lines and n selection lines whose bit combination determine which input is selected.

Multiplexers

- The logic and block diagrams of a 2-to-1-line multiplexer are shown below:



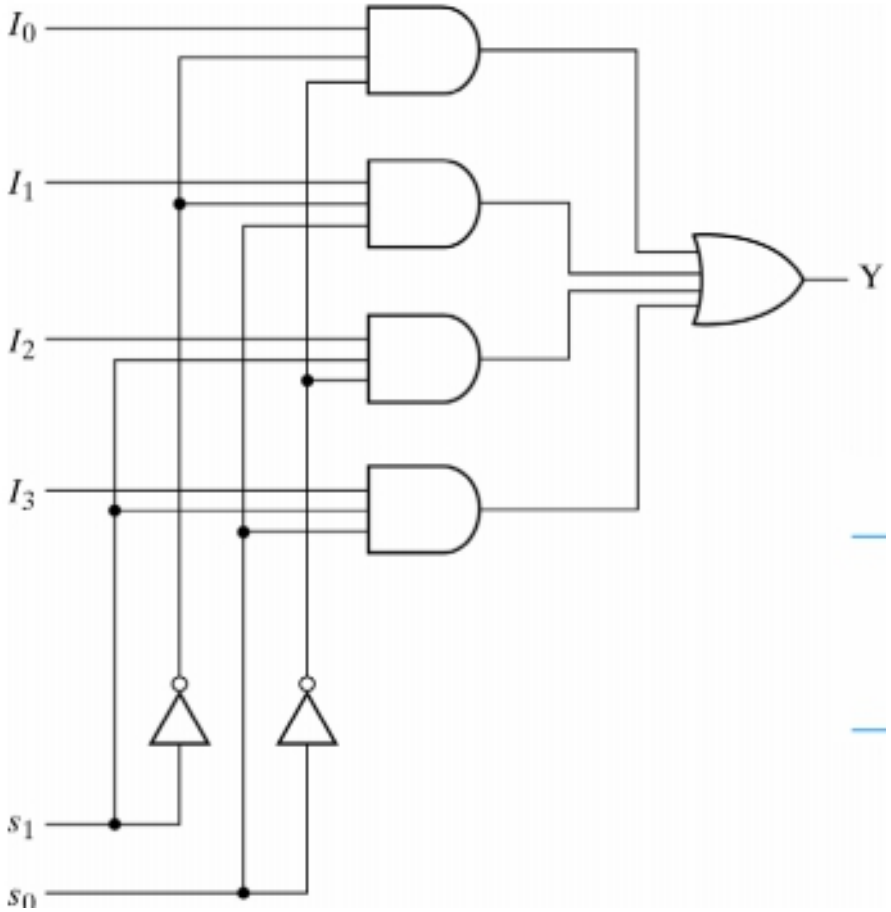
Multiplexers

- The circuit has two data input lines, I_1 and I_2, one output line Y, and one selection line S.
- When S = 1, the lower AND gate is enabled and I_1 has path to the output.
- This multiplexer acts like a switch that selects one of the two sources.

Multiplexers

- A 4-to-1-line multiplexer is shown below:

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



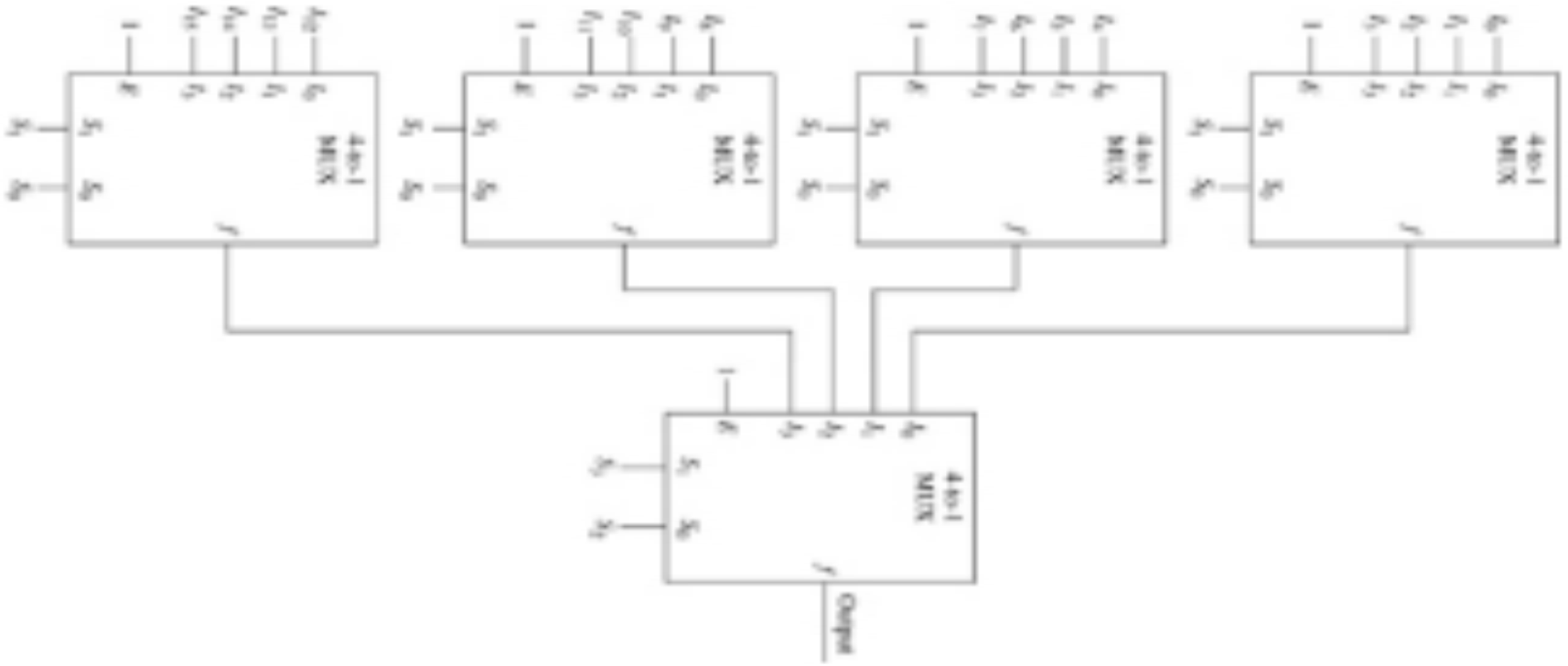
Multiplexers

- A multiplexer is also called a data selector, since it selects one of many inputs and steers the binary information to the output line.
- In general, a 2^n-to-1-line multiplexer is constructed from an n-to-2^n decoder by adding to it 2^n input lines, one to each AND gate. The outputs of the AND gates are applied to a single OR gate.
- As in decoders, multiplexers may have an enable input to control the operation of the unit.

Multiplexers

- By interconnecting several multiplexers in a treelike structure, it is possible to produce a larger multiplexer.
- For example, a 16-to-1 line multiplexer may be constructed using five 4-to-1-line multiplexers as follows:

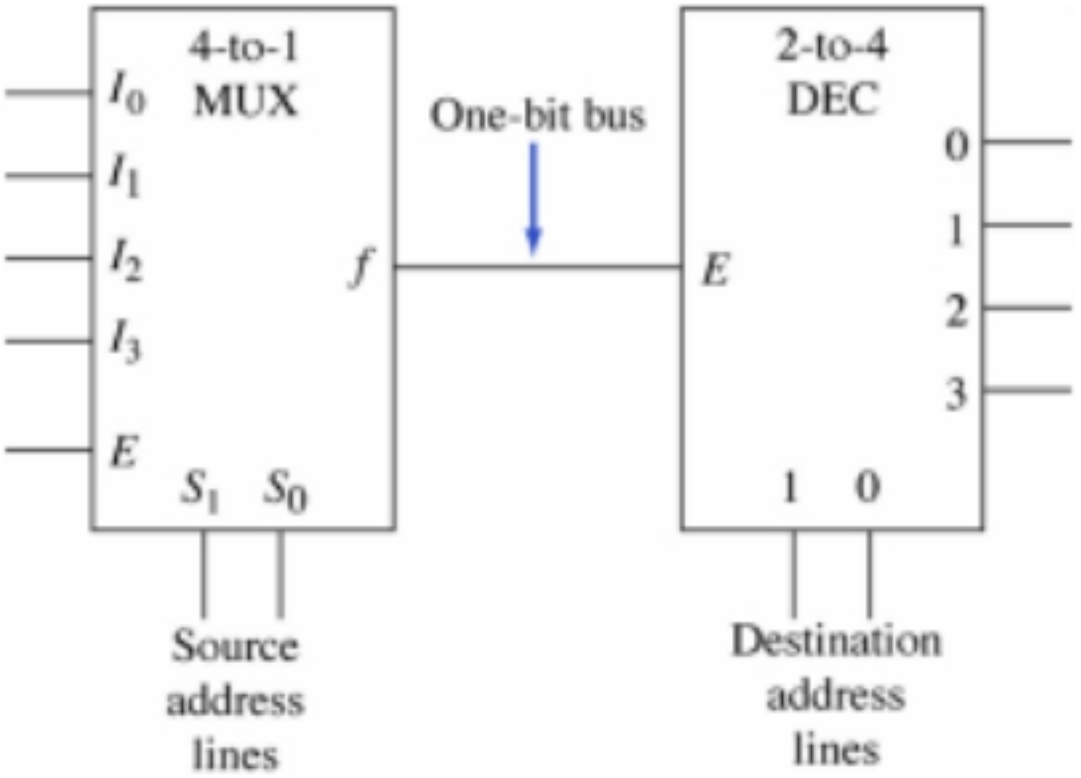
Multiplexers



MUX/DeMUX Transmission System

- One of the primary applications of multiplexers is to provide for the transmission of information from several sources over a single path.
- This process is known as multiplexing. E.g., the multiplexing of conversations on the telephone system.
- When a multiplexer is used in conjunction with a demultiplexer, an effective means is provided for connecting information from several source locations to several destination locations.
- This basic application is illustrated in the next slide:

MUX/DeMUX Transmission System



MUX/DeMUX Transmission System

- By using n of the structures shown above in parallel, an n-bit word from any of four source locations is transferred to the four destination locations.

Logic Design with Multiplexers

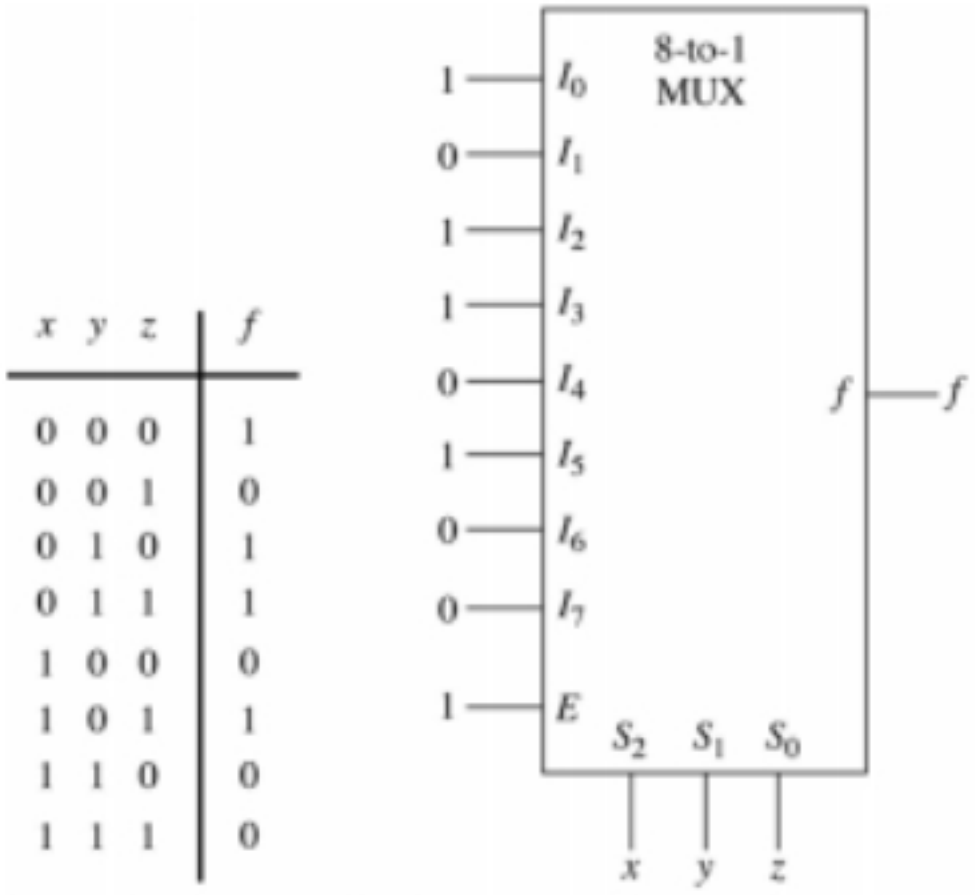
- Consider the Boolean function of three variables:

$$f(x, y, z) = \sum(0, 2, 3, 5)$$

- The function can be implemented with an 8-to-1-line multiplexer:

Logic Design with Multiplexers

x	y	z	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



Logic Design with Multiplexers

- The realization is obtained by placing x, y, and z on the S_2, S_1, and S_0 lines respectively, logic-1 on data input lines I_0, I_2, I_3, and I_5 and logic-0 on the remaining data input lines.
- Also the multiplexer must be enabled by setting E = 1.
- If at least one input variable of a Boolean function is assumed to be available in both its normal and complemented form, then any n-variable function can be realized with a 2^n–1-to-1-line multiplexer.
- For example, reconsider the previous function:

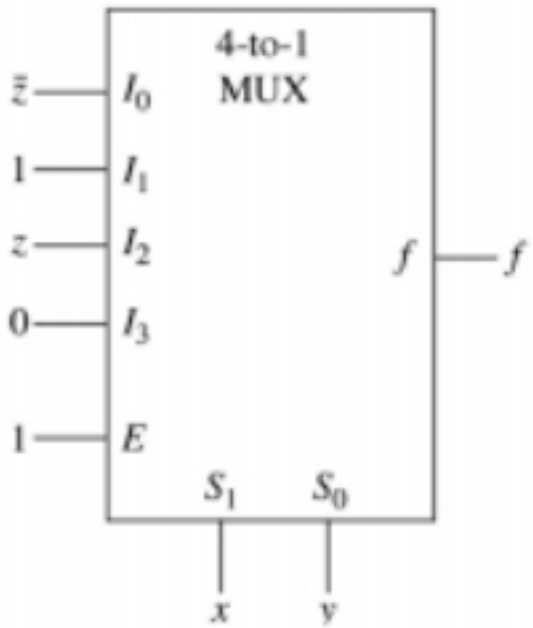
$$f(x, y, z) = \sum(0, 2, 3, 5)$$
$$= x'y'z' + x'yz' + x'yz + xy'z$$

Logic Design with Multiplexers

- Doing some simple factoring becomes:

$$f(x, y, z) = x'y'(z') + x'y(z' + z) + xy'(z)$$
$$= x'y'(z') + x'y(1) + xy'(z) + xy(0)$$

- which is realized using a 4-to-1-line multiplexer
- The last term, xy(0), was included to indicate what input must appear on the I_3 line to provide for the appropriate output when selected with x = y =1.



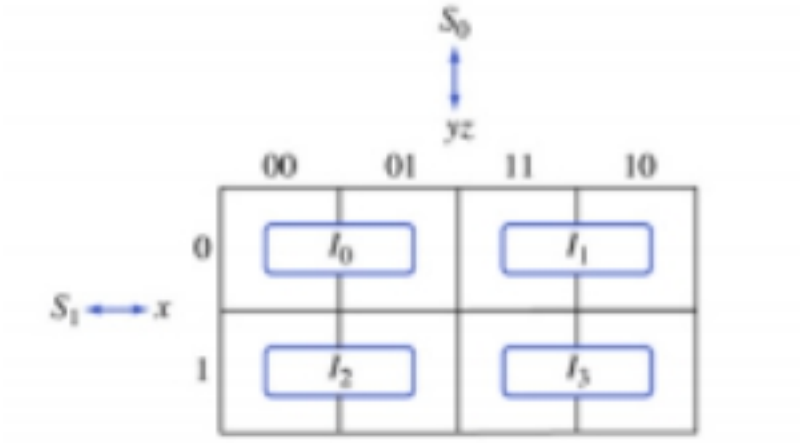
Logic Design with Multiplexers

- Karnaugh maps provide a convenient tool for obtaining multiplexer realizations.
- First it is necessary to establish which variables to assign to the select lines.
- Next the inputs for the Ii data lines are read directly from the map.
- To illustrate this, again consider the three-variable function:

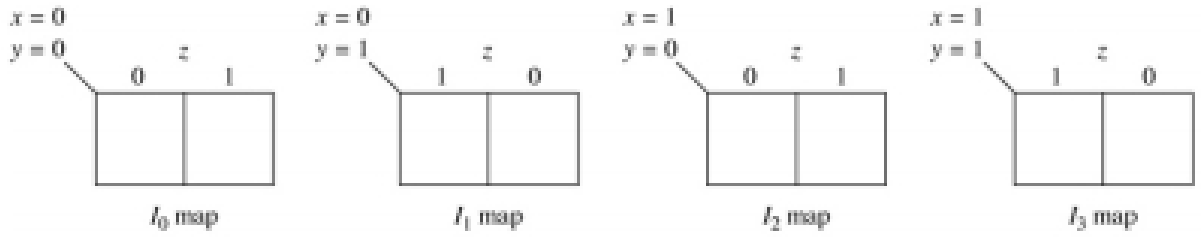
$$f(x, y, z) = \sum(0, 2, 3, 5)$$

Logic Design with Multiplexers

- Assume that x is placed on the S_1 line and y is placed on the S_0 line, the resulting map is:



- submaps:



Logic Design with Multiplexers

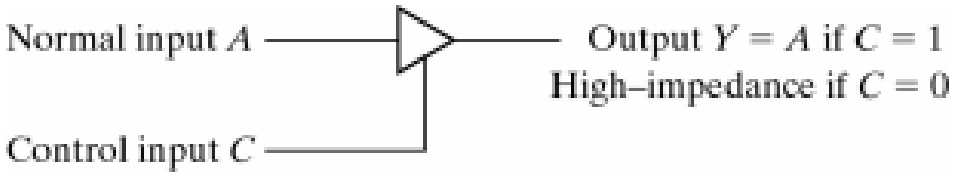
- Grouping the 1-cells, the expressions for the sub$_{\urcorner}$ functions may be written. That is, $I_0 = z'$, $I_1 = 1$, $I_2 = z$, and $I_3 = 0$.
- The logic realization is as before.

Three-State Gates

- A multiplexer can be constructed with three-state gates.
- A three-state gate is a digital circuit that exhibits three states.
- Two of the states are signals equivalent to logic 1 and 0.
- The third state is a high-impedance state which behaves like an open circuit.

Three-State Gates

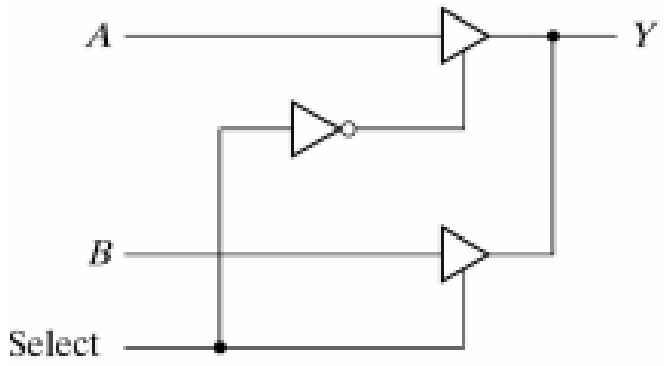
- The graphic symbol of a three-state buffer is:



- The presence of the high-impedance state allows the connection of a large number of three-state gate outputs to a common line without endangering loading effects.

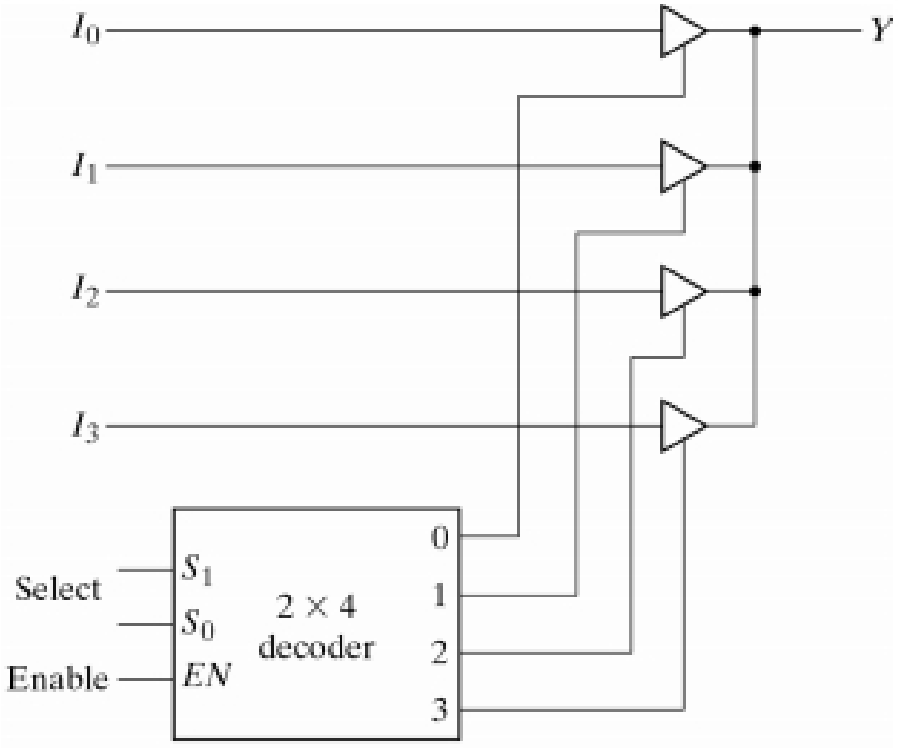
Three-State Gates

- The realization of a 2-to-1-line multiplexer with three-state buffers is shown below:



Three-State Gates

- and the construction of a 4-to-1-line multiplexer is shown below:



Three-State Gates

- The use of a decoder ensures that no more than one buffer control input is active at any given time.
- When the EN input of the decoder is 0, all of its four outputs are 0, and the bus line is in a high-impedance state.
- When EN is active, one of the buffers will be active depending on the binary value in S_1 and S_2 of the decoder.