DIGITAL LOGIC DESIGN EE(109)

Chapter No:6
Function of combinational logic

TOPICS

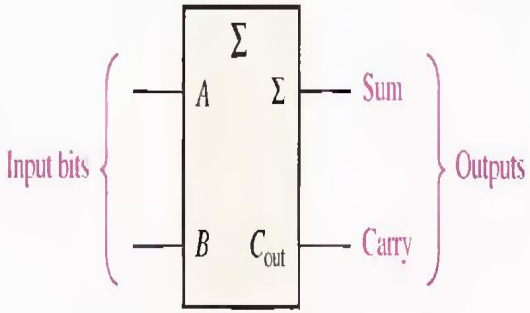
1. Half and Full Adders
2. Parallel Binary Adders
3. Comparators
4. Decoders
5. Encoders
6. Code Converters
7. Multiplexers (Data Selectors)
8. Demultiplexers

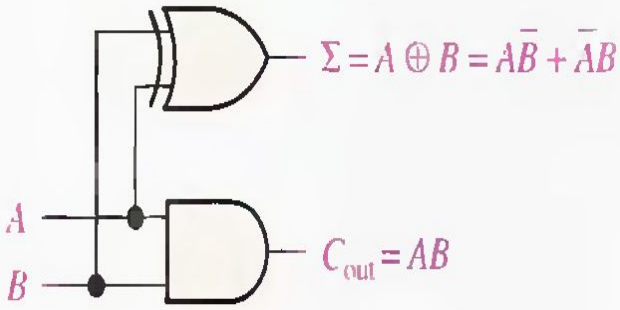
BASIC ADDERS

The Half-Adder

The operations are performed by a logic circuit called a **half-adder.**

The half-adder accepts two binary digits on its inputs and produces two binary digits on its outputs, a sum bit and a carry bit.





A	B	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Σ = sum

C_{out} = output carry

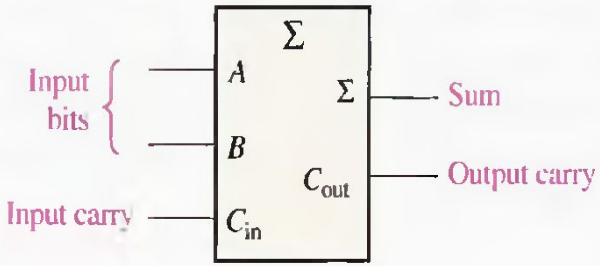
A and *B* = input variables (operands)

$$\Sigma = A \oplus B$$

$$C_{out} = AB$$

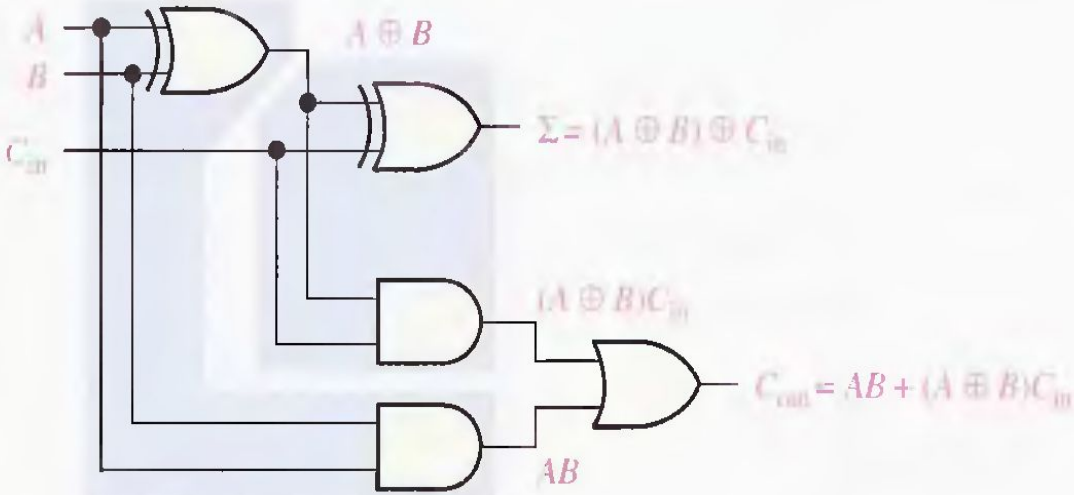
The Full-Adder

The full-adder accepts two input bits and an input carry and generates a sum output and an output carry.



$$\Sigma = (A \oplus B) \oplus C_{in}$$

$$C_{out} = AB + (A \oplus B)C_{in}$$



A	B	C_{in}	C_{out}	Σ
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

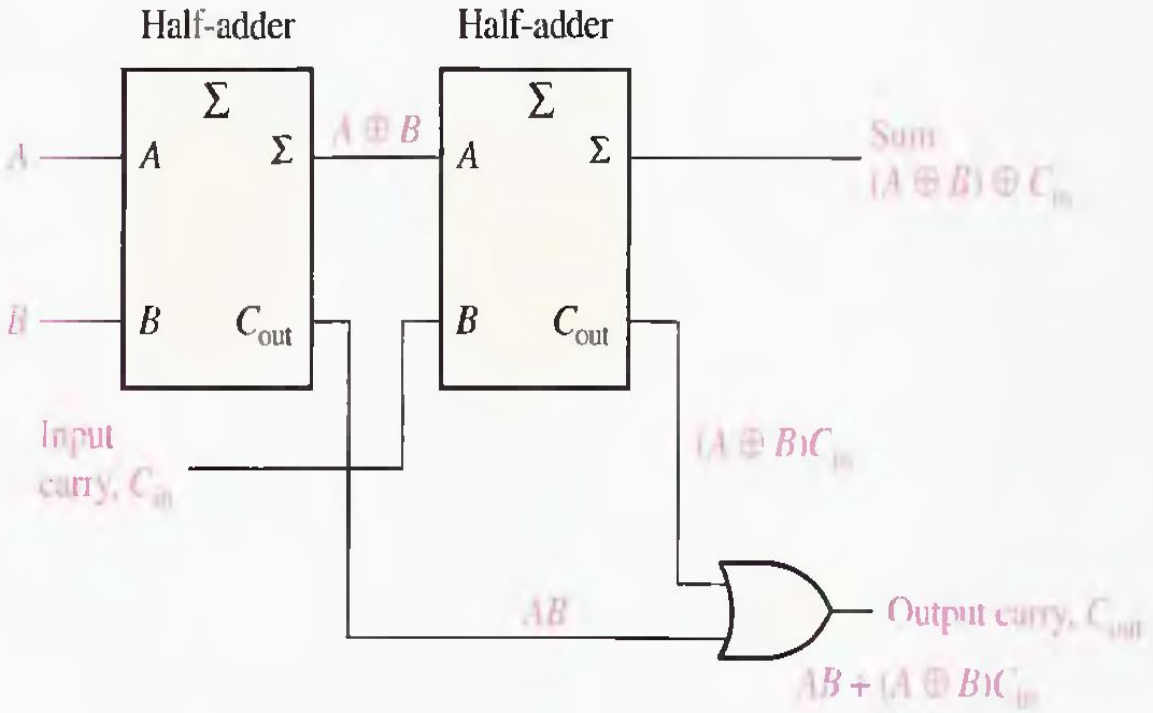
C_{in} = input carry, sometimes designated as *CI*

C_{out} = output carry, sometimes designated as *CO*

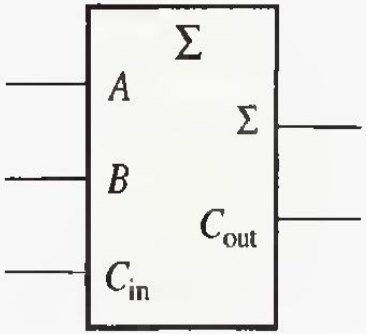
Σ = sum

A and *B* = input variables (operands)

The Full-Adder



(a) Arrangement of two half-adders to form a full-adder

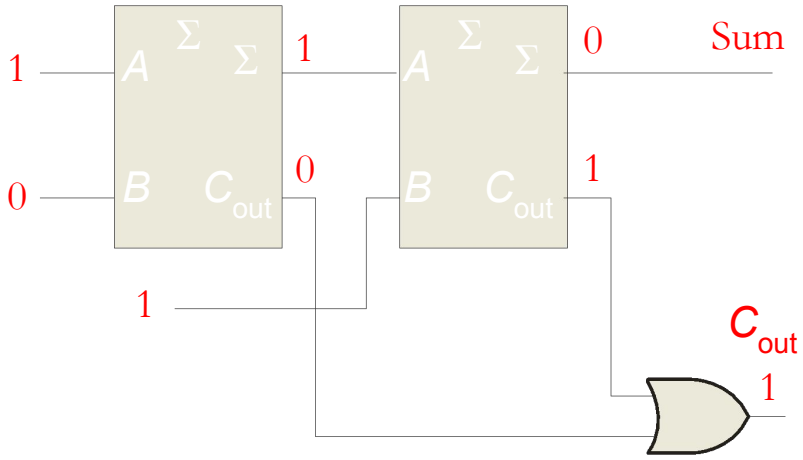


(b) Full-adder logic symbol

Full-Adder

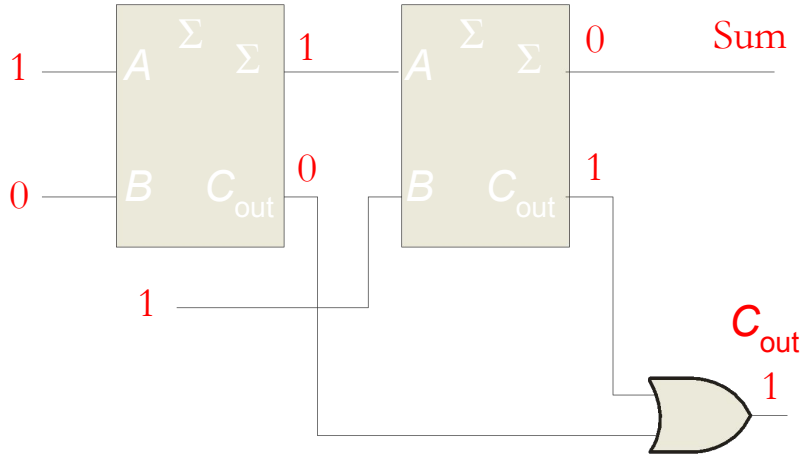
Example

Solution



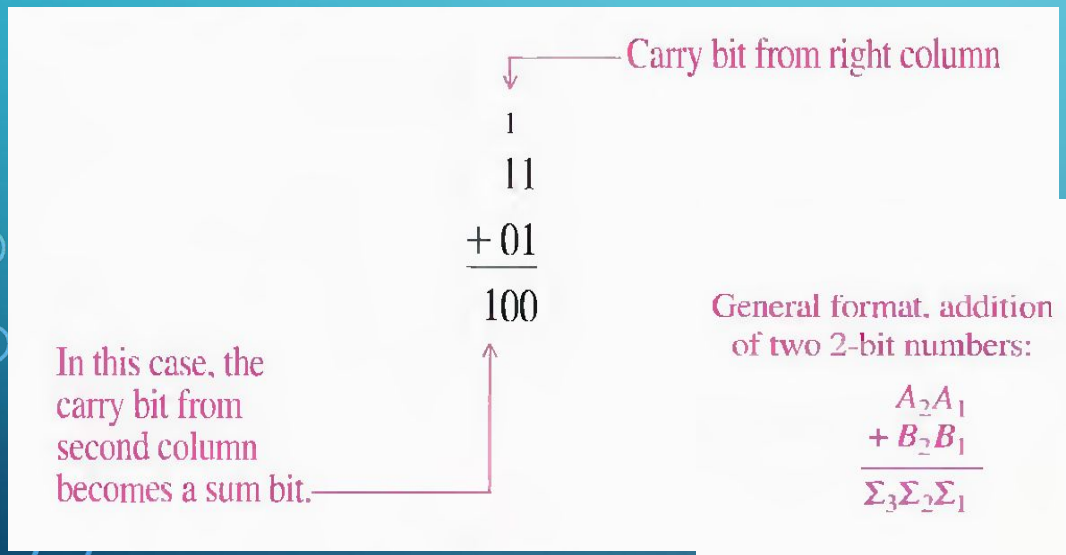
Full-Adder

Inputs			Outputs	
A	B	C_{in}	C_{out}	Σ
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



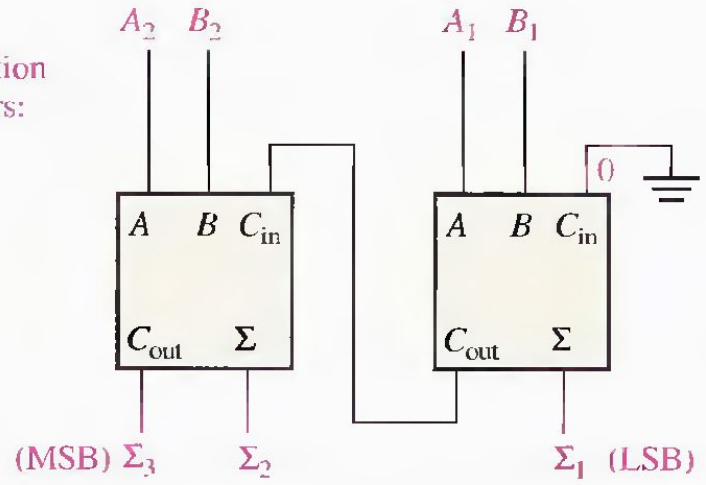
PARALLEL BINARY ADDERS

Two or more full-adders are connected to form parallel binary adders. A single full-adder is capable of adding two, I-bit numbers and an input carry. To add binary numbers with more than one bit, you must use additional full-adders.



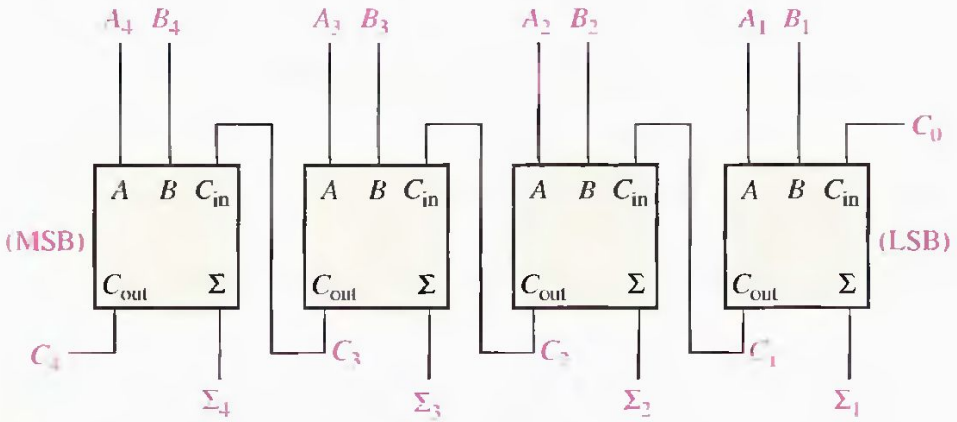
General format, addition of two 2-bit numbers:

$$\begin{array}{r} A_2A_1 \\ +\,B_2B_1 \\ \hline \Sigma_3\Sigma_2\Sigma_1 \end{array}$$

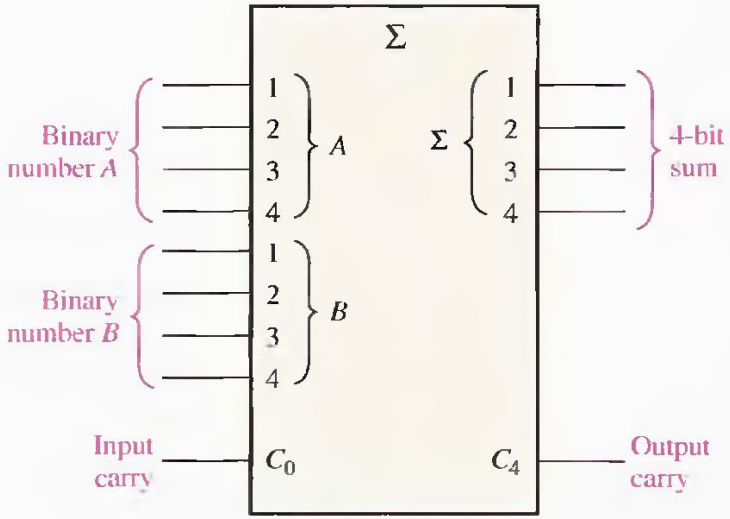


Four-Bit Parallel Adders

A group of four bits is called a nibble. A basic 4-bit parallel adder is implemented with four full-adder stages .



(a) Block diagram

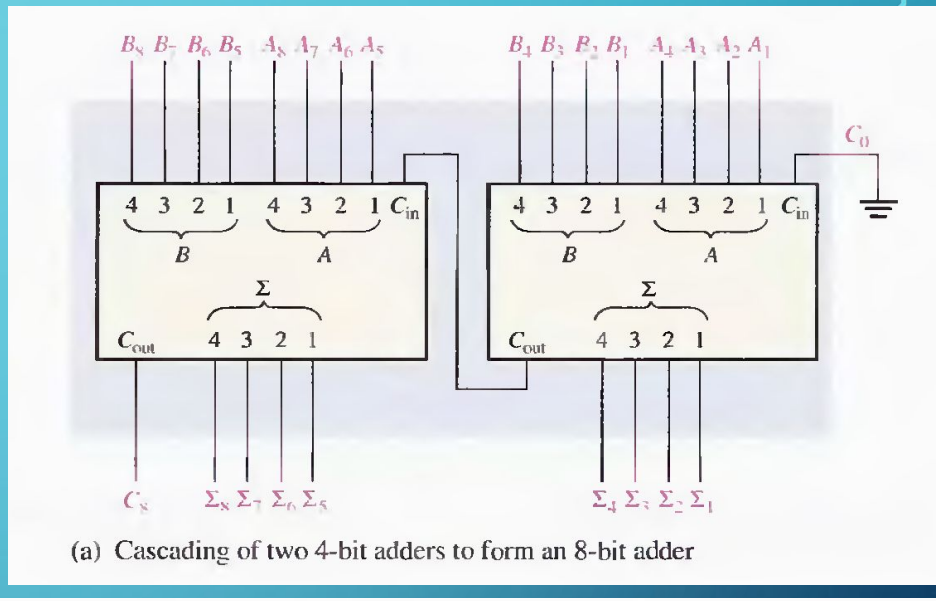


(b) Logic symbol

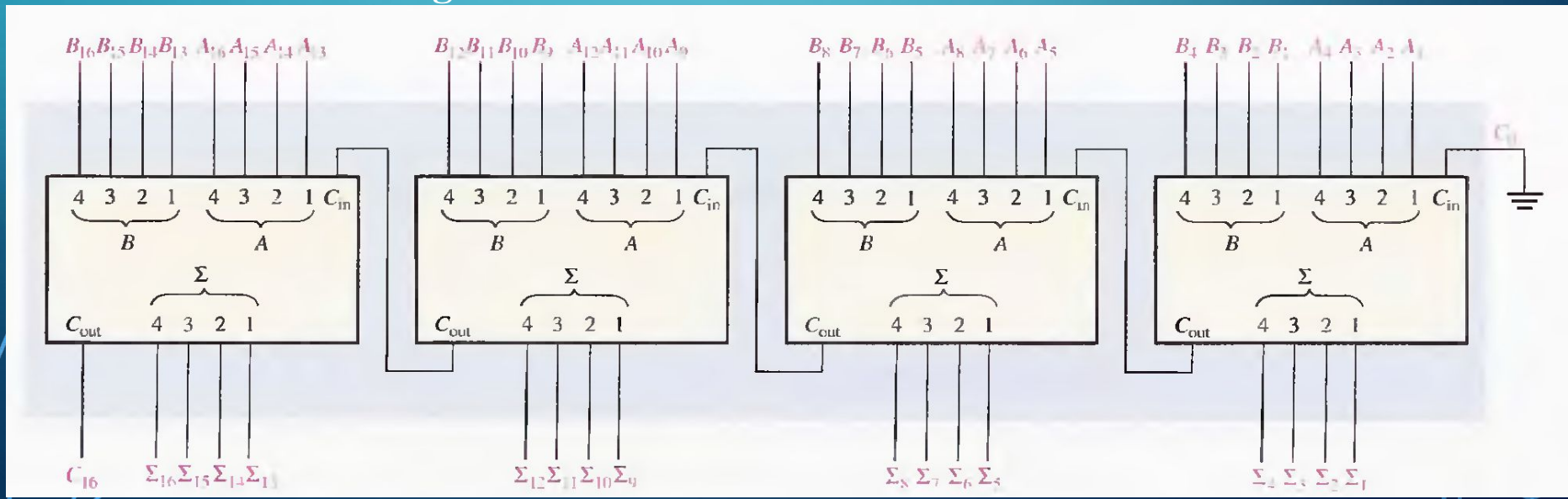
C_{n-1}	A_n	B_n	Σ_n	C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Adder Expansion

Cascading: The 4-bit parallel adder can be expanded to handle the addition of two 8-bit numbers by using two 4-bit adders. The carry input of the low-order adder (Co) is connected to ground because there is no carry into the least significant bit position, and the carry output of the low-order adder is connected to the carry input of the high-order adder, as shown in Figure . This process is known as ***cascading.***



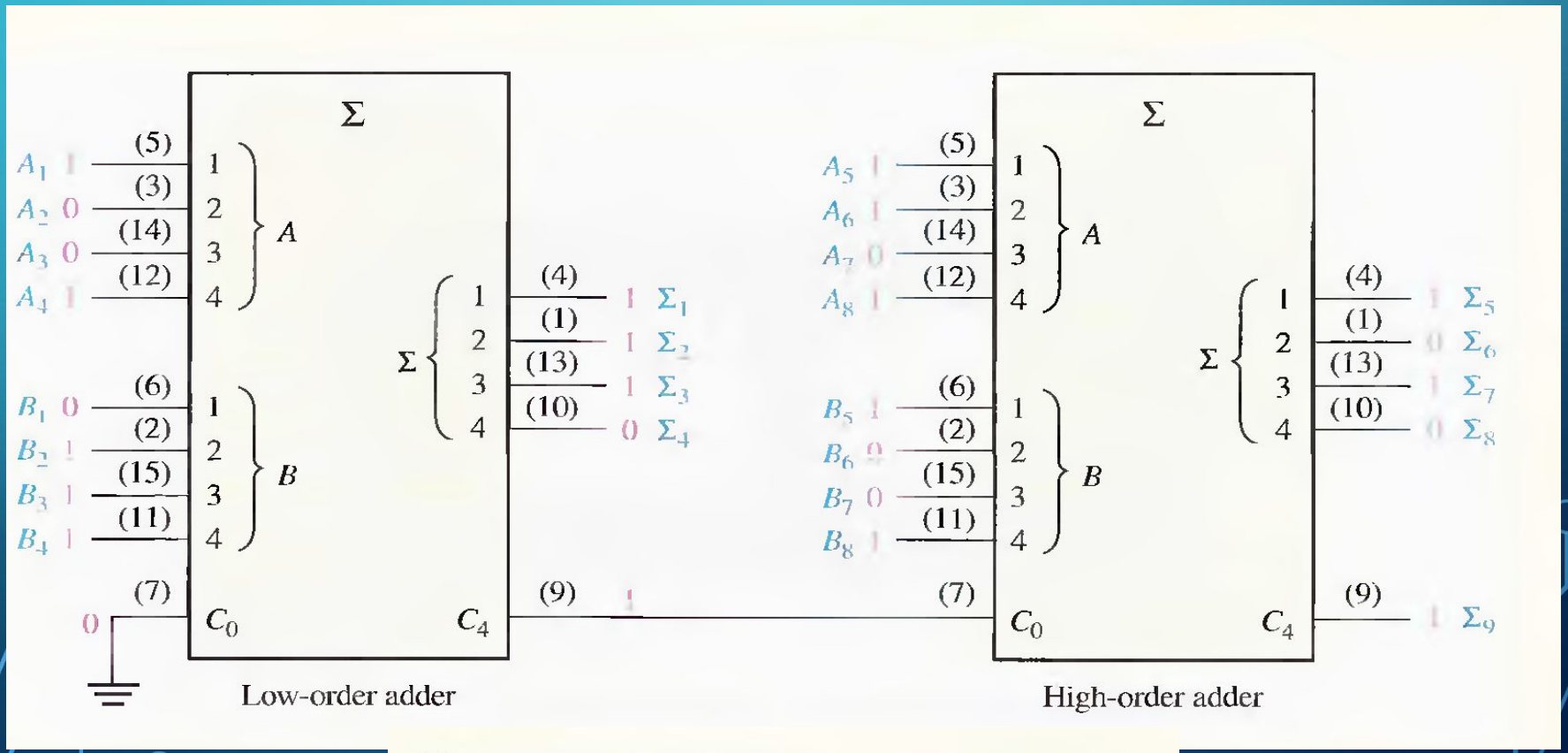
(a) Cascading of two 4-bit adders to form an 8-bit adder



EXAMPLE

Show how two 74LS283 adders can be connected to form an 8-bit parallel adder. Show output bits for the following 8-bit input numbers:

$$A_8A_7A_6A_5A_4A_3A_2A_1 = 10111001 \quad \text{and} \quad B_8B_7B_6B_5B_4B_3B_2B_1 = 10011110$$



$$\Sigma_9\Sigma_8\Sigma_7\Sigma_6\Sigma_5\Sigma_4\Sigma_3\Sigma_2\Sigma_1 = 101010111$$

Use the 4-bit parallel adder truth table (Table 6–3) to find the sum and output carry for the addition of the following two 4-bit numbers if the input carry (C_{n-1}) is 0:

$$A_4A_3A_2A_1 = 1100 \quad \text{and} \quad B_4B_3B_2B_1 = 1100$$

For $n = 1$: $A_1 = 0$, $B_1 = 0$, and $C_{n-1} = 0$. From the 1st row of the table,

$$\Sigma_1 = \mathbf{0} \quad \text{and} \quad C_1 = 0$$

For $n = 2$: $A_2 = 0$, $B_2 = 0$, and $C_{n-1} = 0$. From the 1st row of the table,

$$\Sigma_2 = \mathbf{0} \quad \text{and} \quad C_2 = 0$$

For $n = 3$: $A_3 = 1$, $B_3 = 1$, and $C_{n-1} = 0$. From the 4th row of the table,

$$\Sigma_3 = \mathbf{0} \quad \text{and} \quad C_3 = 1$$

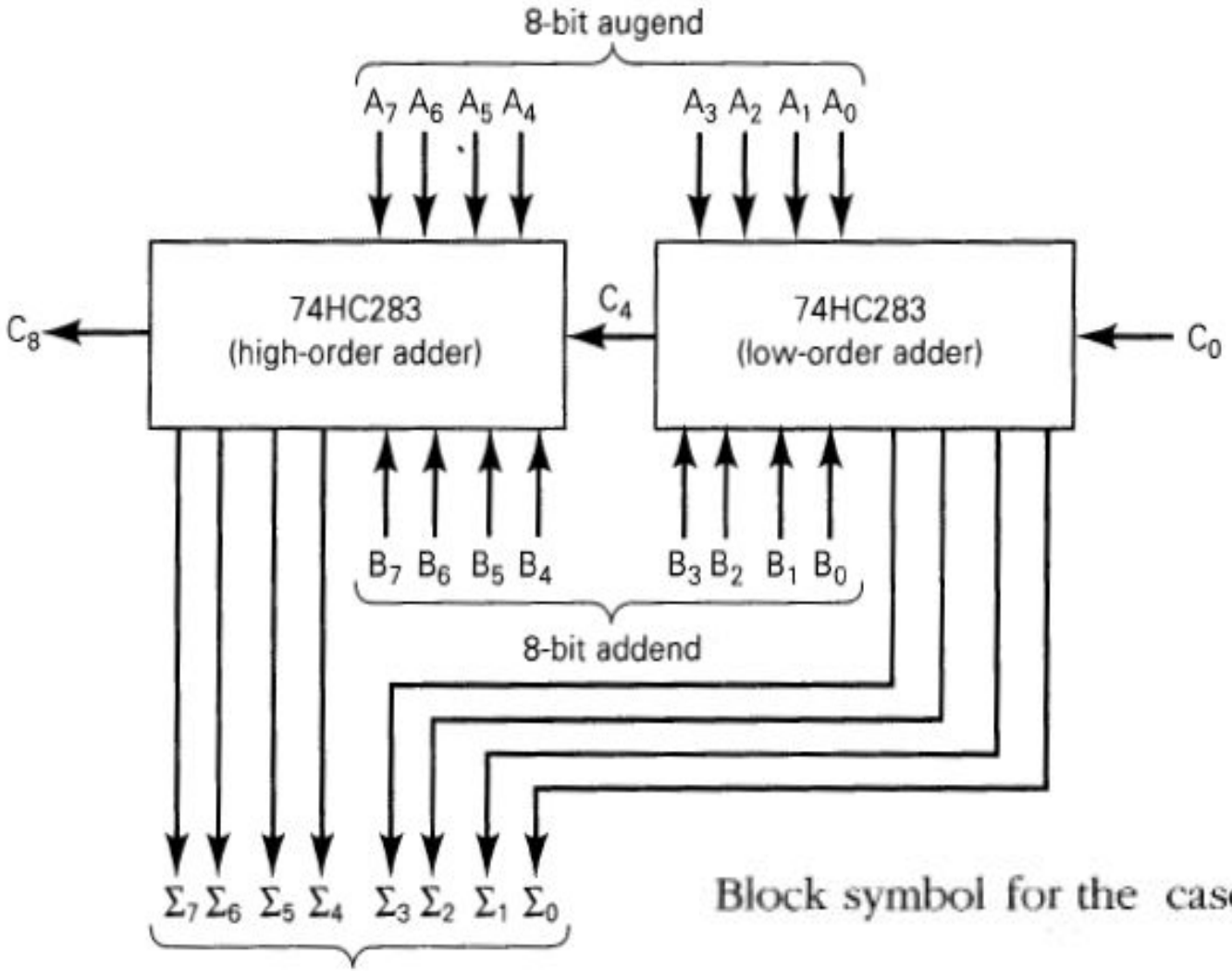
For $n = 4$: $A_4 = 1$, $B_4 = 1$, and $C_{n-1} = 1$. From the last row of the table,

$$\Sigma_4 = \mathbf{1} \quad \text{and} \quad C_4 = \mathbf{1}$$

C_4 becomes the output carry; the sum of 1100 and 1100 is 11000.

Cascading Parallel Adders

Two or more IC adders can be connected together (cascaded) to accomplish the addition of larger binary numbers. Figure shows two 74HC283 adders connected to add two 8-bit numbers $A_7\ A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$ and $B_7\ B_6\ B_5\ B_4\ B_3\ B_2\ B_1\ B_0$.



Block symbol for the cascading two 74HC283s.

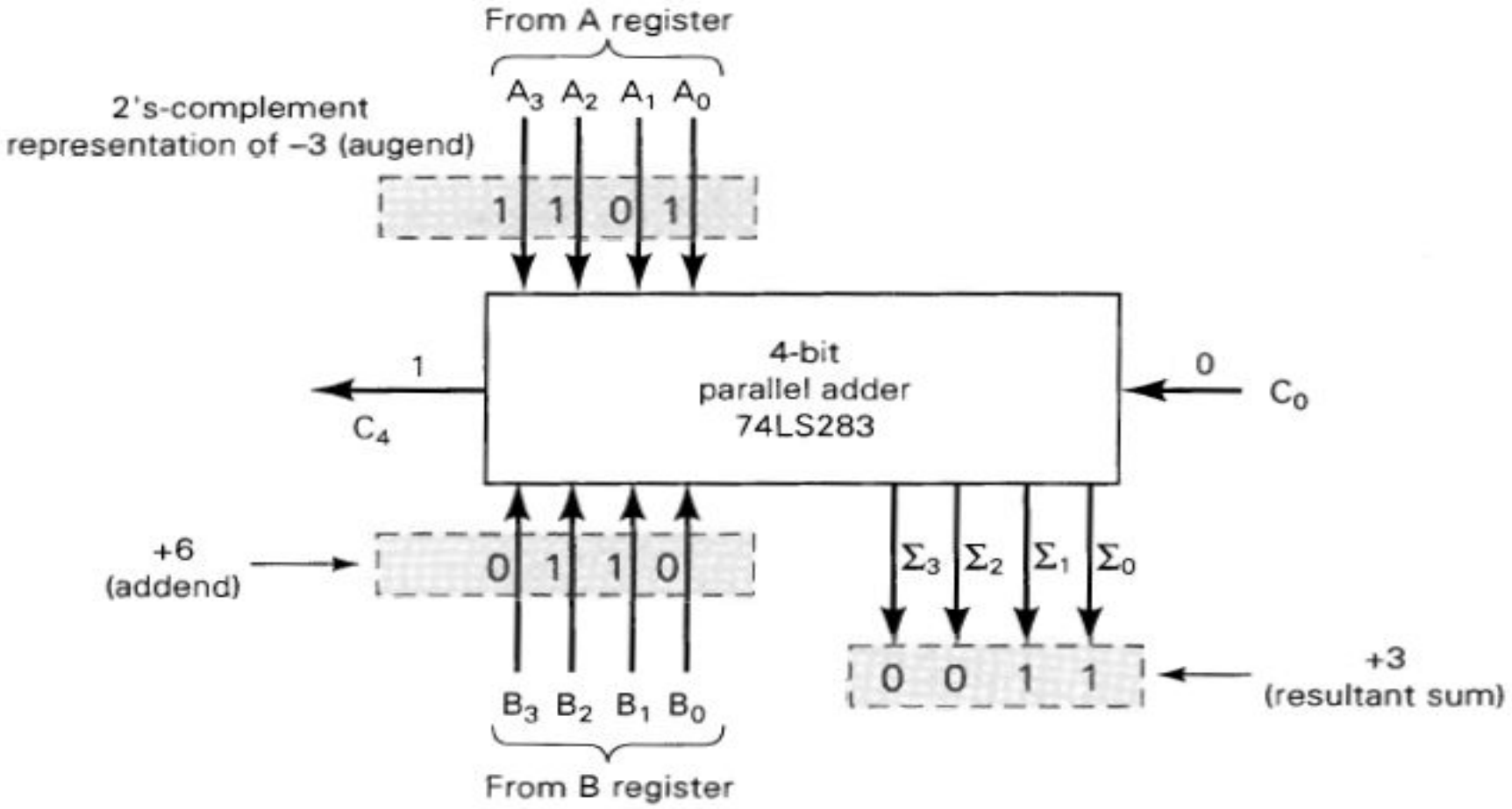
2's-COMPLEMENT SYSTEM

Addition

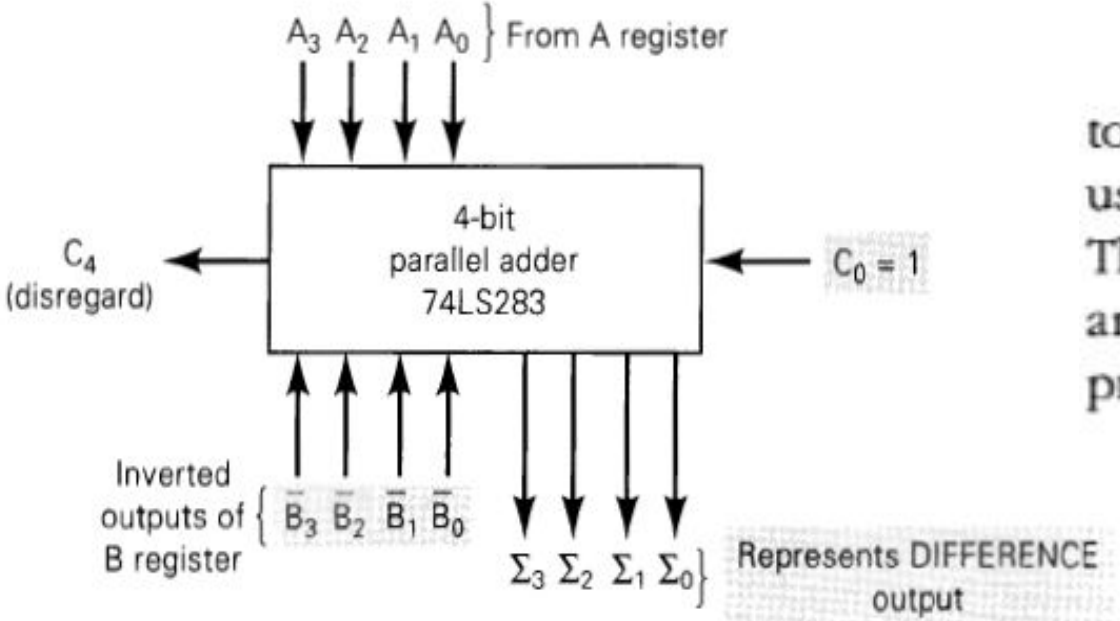
Positive and negative numbers, including the sign bits, can be added together in the basic parallel-adder circuit when the negative numbers are in 2's-complement form.

for the addition of −3 and +6. The −3 is represented in its 2's-complement form as 1101

Parallel adder used to add + and − numbers in 2's-complement system.



Subtraction

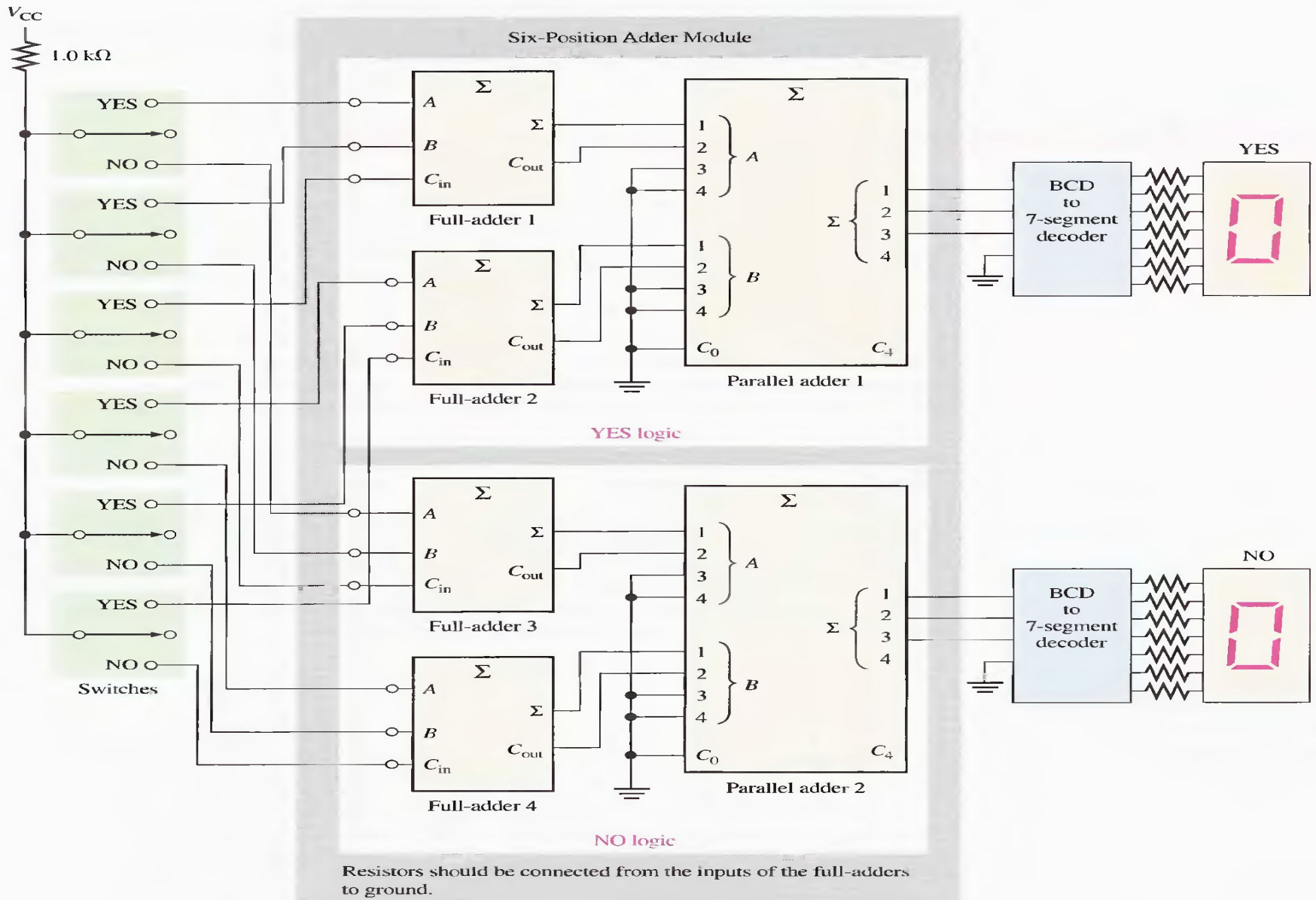


Parallel adder used to perform subtraction ($A - B$) using the 2's-complement system. The bits of the subtrahend (B) are inverted, and $C_0 = 1$ to produce the 2's complement.

1. +4 is stored in the A register as 0100.
2. +6 is stored in the B register as 0110.
3. The inverted outputs of the B-register FFs (1001) are fed to the adder.
4. The parallel-adder circuitry adds $[A] = 0100$ to $[\overline{B}] = 1001$ along with a carry, $C_0 = 1$, into the LSB. The operation is shown below.

$$\begin{array}{rl} 1 & \leftarrow C_0 \\ 0100 & \leftarrow [A] \\ + \ \underline{1001} & \leftarrow [\overline{B}] \\ 1110 & \leftarrow [\Sigma] = [A] - [B] \end{array}$$

In its simplest form, the system includes a switch for "yes" or "no" selection at each position in the assembly and a digital display for the number of yes votes and one for the number of no votes.

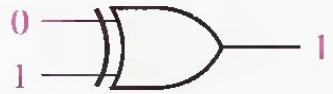


COMPARATOR

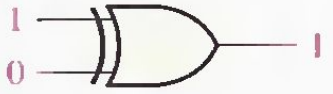
The basic function of a comparator is to compare the magnitudes of two binary quantities to determine the relationship of those quantities.

Basic comparator operation.

The input bits are equal.



The input bits are not equal.

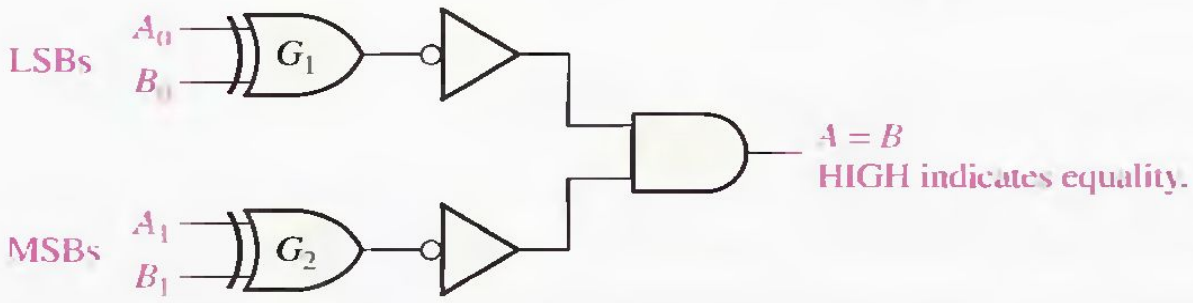


The input bits are not equal.

The input bits are equal.

In order to compare binary numbers containing two bits

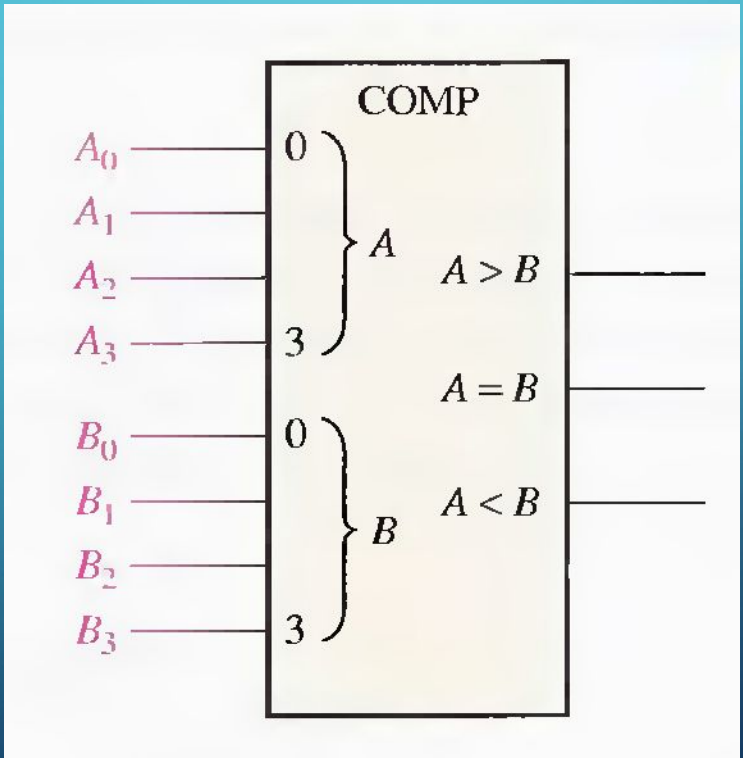
Equality



General format: Binary number $A \rightarrow A_1A_0$
Binary number $B \rightarrow B_1B_0$

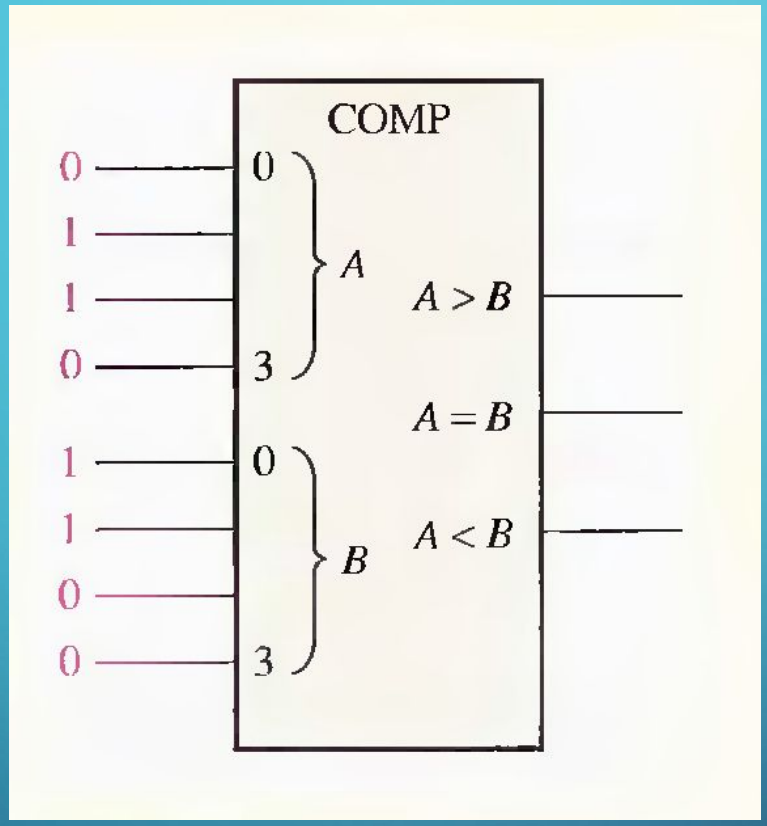
Inequality

1. If $A_3 = 1$ and $B_3 = 0$, number A is greater than number B.
2. If $A_3 = 0$ and $B_3 = 1$, number A is less than number B.
3. If $A_3 = B_3$, then you must examine the next lower bit position for an inequality.



The highest-order indication must take precedence.

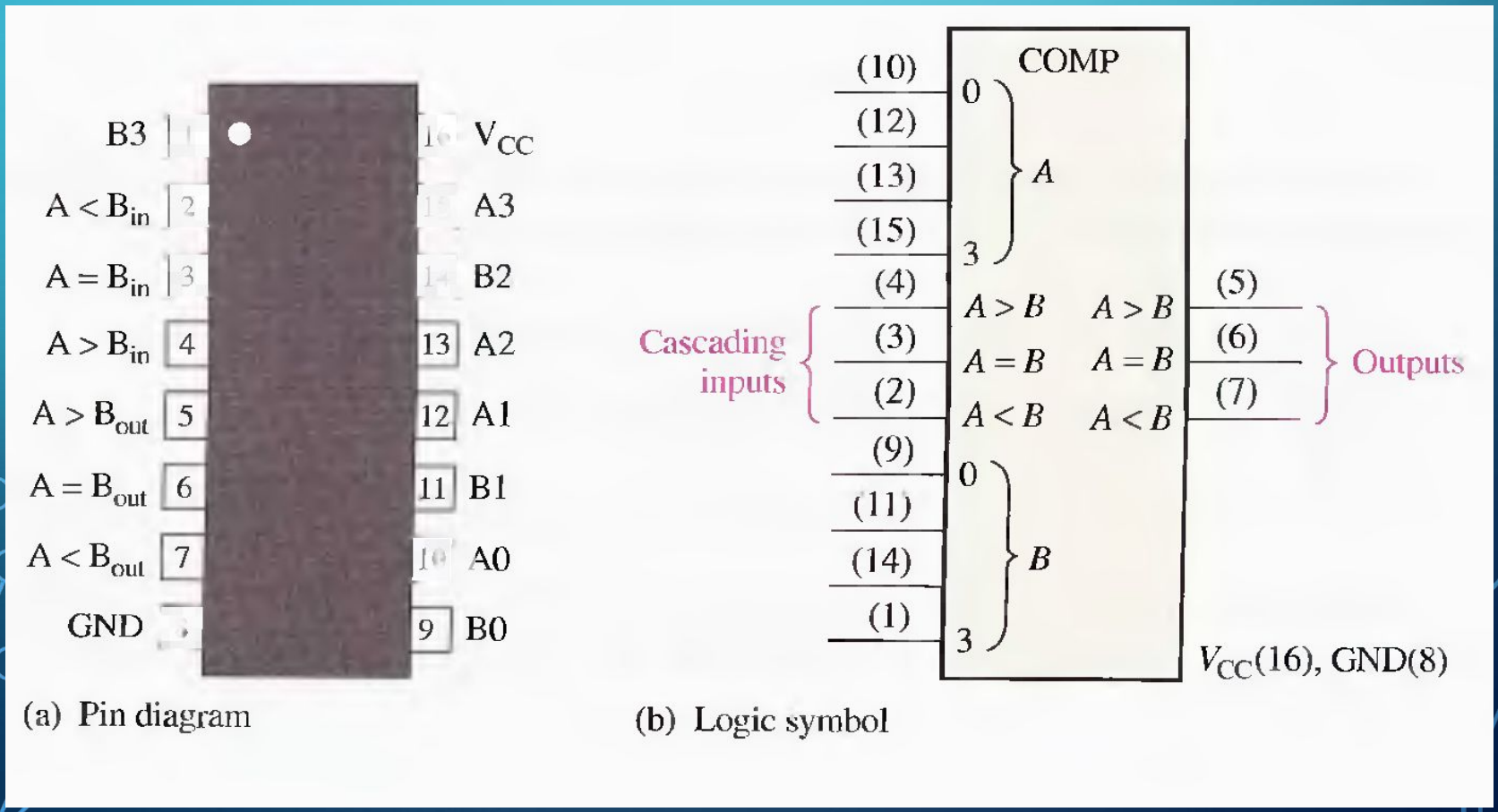
Determine the A = B, A > B, and A < B outputs for the input numbers shown on the comparator in Figure .



The number on the *A* inputs is 0110 and the number on the *B* inputs is 0011. The ***A* > *B* output is HIGH and the other outputs are LOW.**

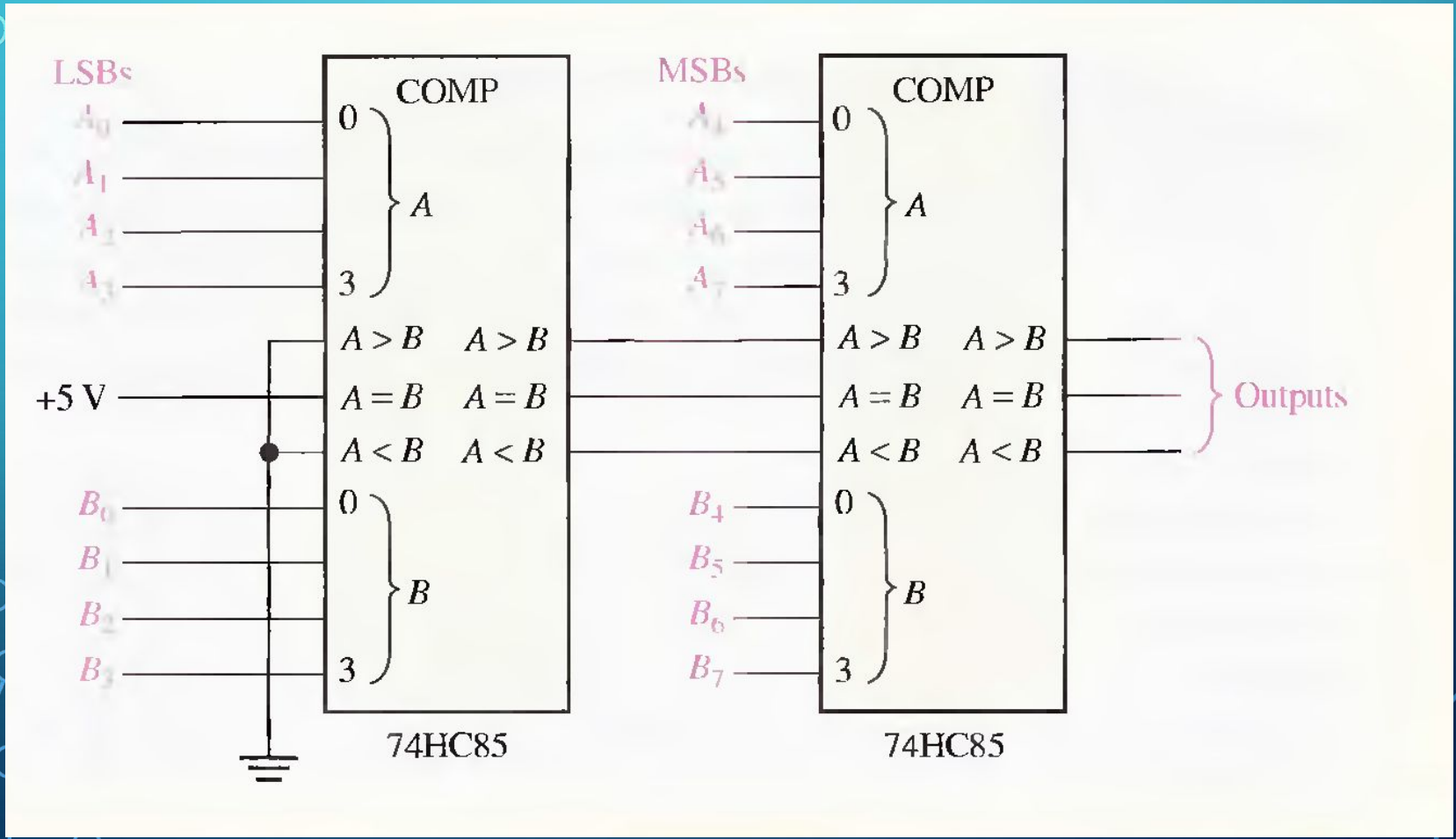
THE 74HC85 4-BIT MAGNITUDE COMPARATOR

The 74HC85 is a comparator that is also available in other IC families. The pin diagram and logic symbol are shown in Figure 0-24. Notice that this device has all the inputs and outputs of the generalized comparator previously discussed and, in addition, has three cascading inputs: A < B, A = B, A > B. These inputs allow several comparators to be cascaded for comparison of any number of bits greater than four.



Compare two 8-bit numbers:

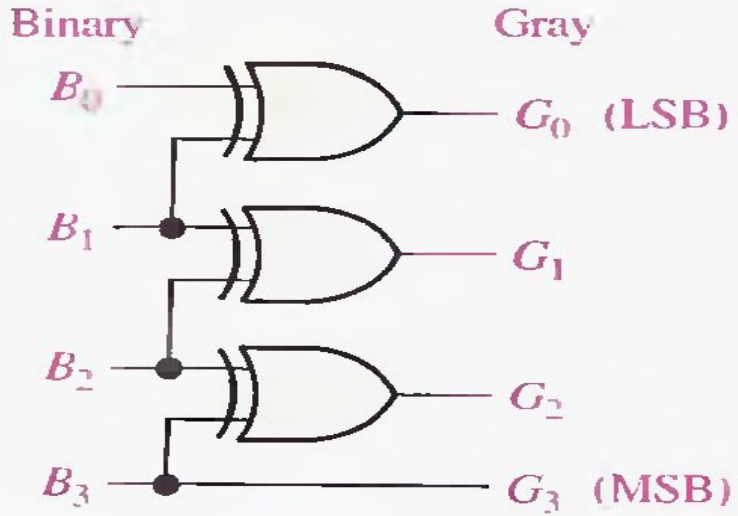
Two 74HC85s are required to compare two 8-bit numbers. They are connected as shown in Figure in a cascaded arrangement.



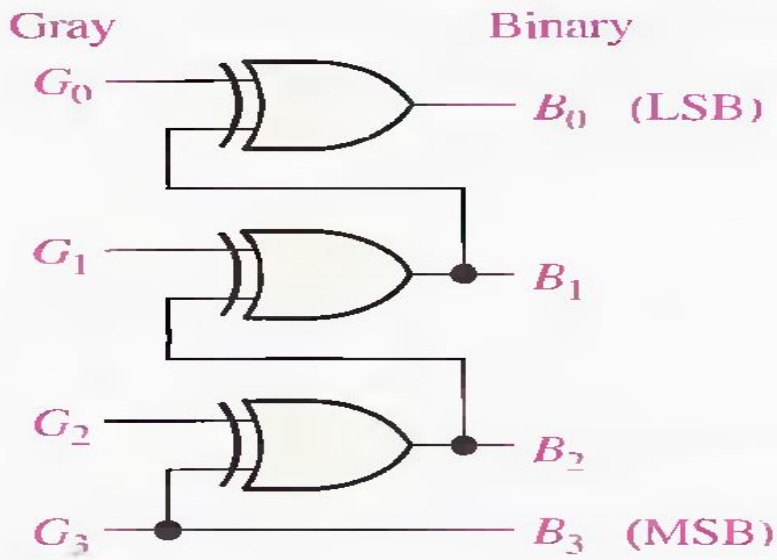
Binary-to-Gray and Gray-to-Binary Conversion

There are various code converters that change one code to another. Two examples are the four bit binary-to-Gray converter and the Gray-to-binary converter.

Four-bit binary-to-Gray conversion logic.

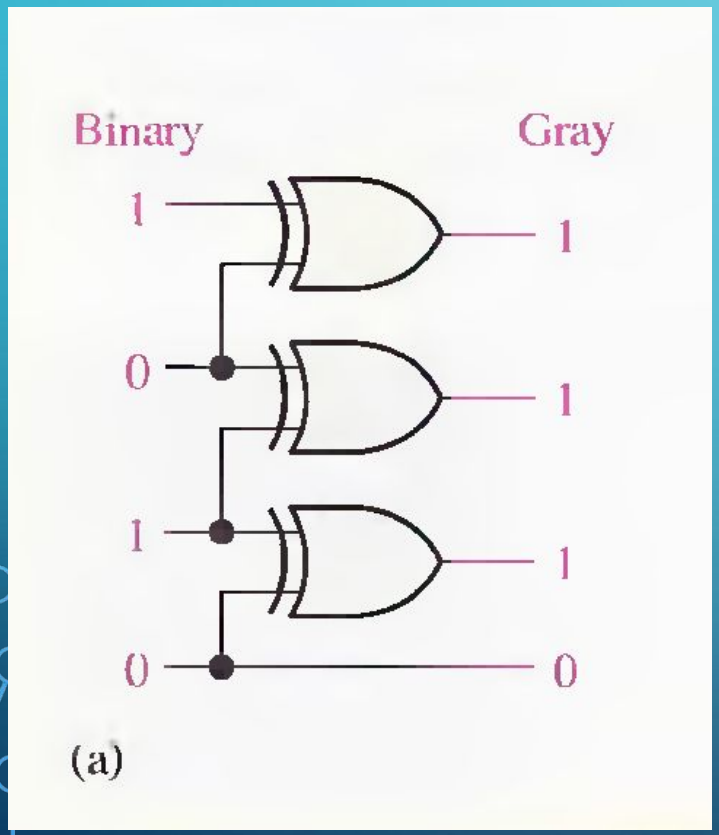


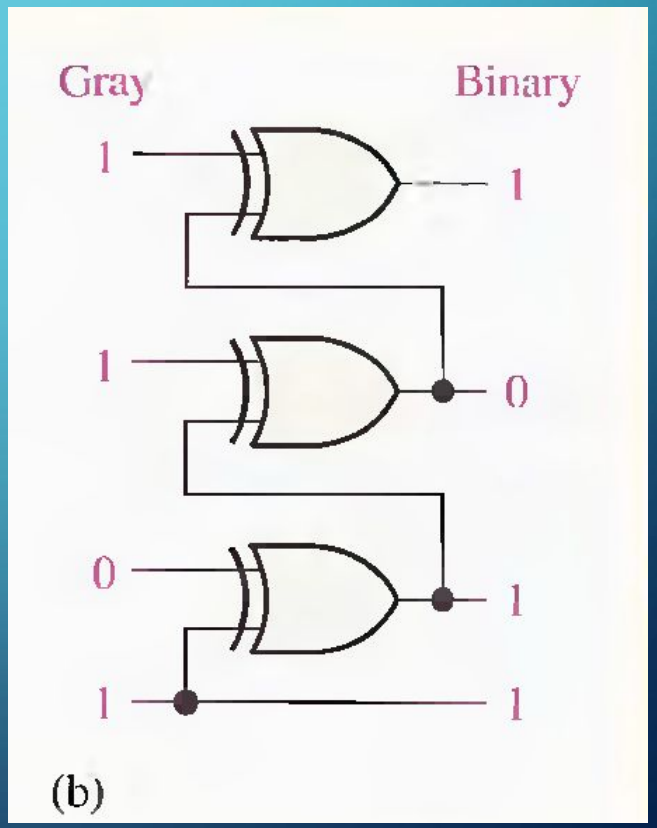
Four-bit Gray-to-binary conversion logic.



(a) Convert the binary number 0101 to Gray code with exclusive-OR gates.

(b) Convert the Gray code 1011 to binary with exclusive-OR gates.

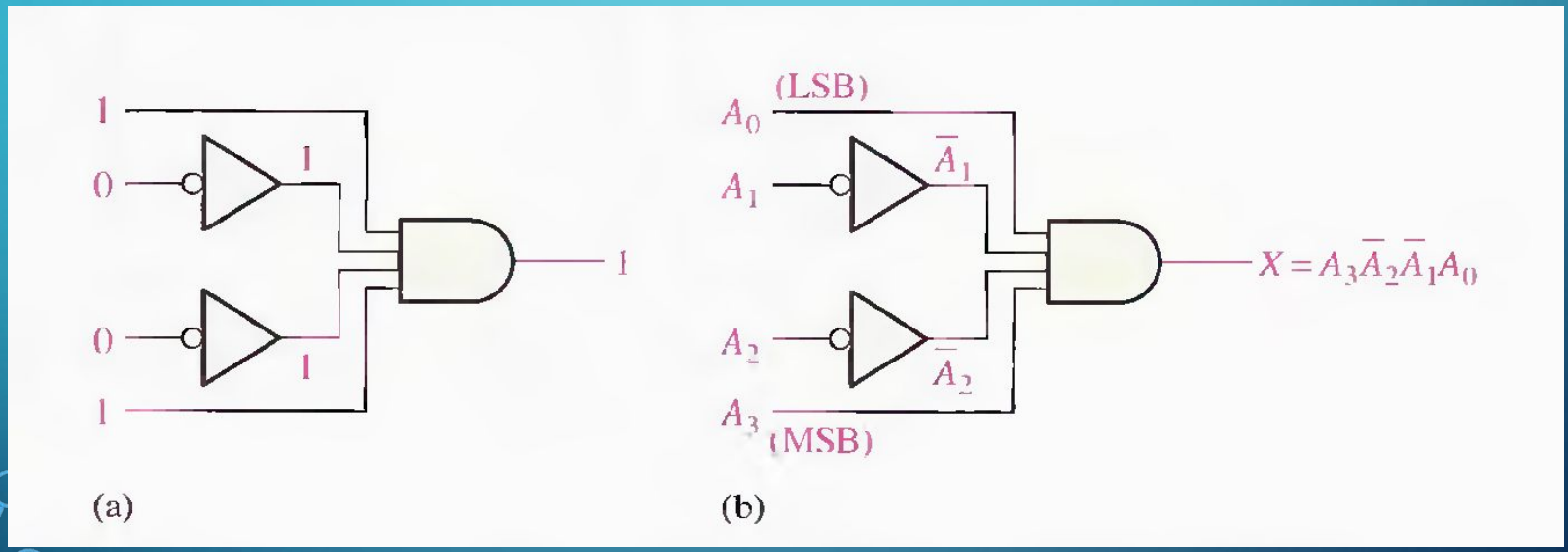




DECODERS

A decoder is a digital circuit that detects the presence of a specified combination of bits (code) on its inputs and indicates the presence of that code by a specified output level.

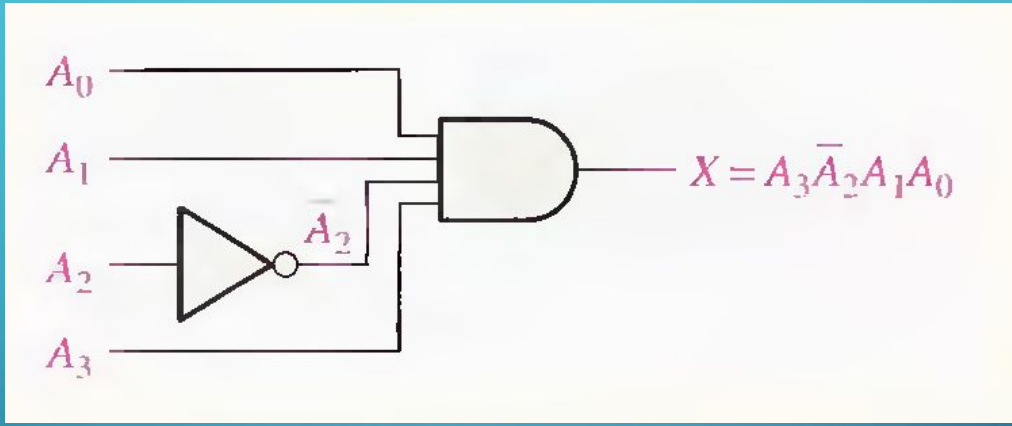
Suppose you need to determine when a binary 1001 occurs on the inputs of a digital circuit.



If a NAND gate is used in place of the AND gate in Figure , a LOW output will indicate the presence of the proper binary code, which is 100 I in this case.

Example

Determine the logic required to decode the binary number 1011 by producing a HIGH level on the output.



Develop the logic required to detect the binary code 10010 and produce an active-LOW output.

The 4-Bit Decoder

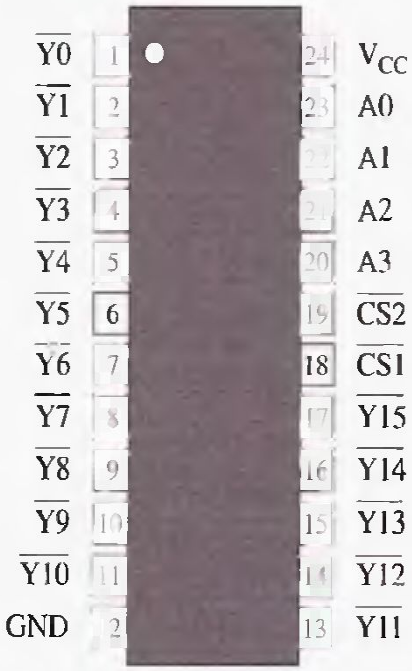
In order to decode all possible combinations of four bits, sixteen decoding gates are required (2^4 = 16). This type of decoder is commonly called either a 4-1ine-to-16-1ine decoder because there are four inputs and sixteen outputs .

Decoding functions and truth table for a 4-line-to-16-line (1-of-16) decoder with active-LOW outputs.

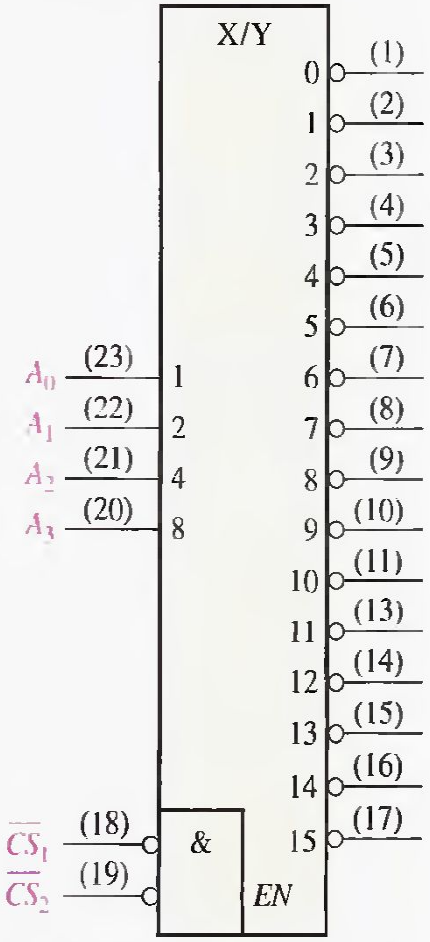
DECIMAL DIGIT	BINARY INPUTS A_3	A_2	A_1	A_0	DECODING FUNCTION	OUTPUTS 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	1	0	$\overline{A}_3\overline{A}_2A_1\overline{A}_0$	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0	1	1	$\overline{A}_3\overline{A}_2A_1A_0$	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
4	0	1	0	0	$\overline{A}_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
5	0	1	0	1	$\overline{A}_3A_2\overline{A}_1A_0$	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
6	0	1	1	0	$\overline{A}_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
7	0	1	1	1	$\overline{A}_3A_2A_1A_0$	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
8	1	0	0	0	$A_3\overline{A}_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
9	1	0	0	1	$A_3\overline{A}_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
10	1	0	1	0	$A_3\overline{A}_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
11	1	0	1	1	$A_3\overline{A}_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
12	1	1	0	0	$A_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
13	1	1	0	1	$A_3A_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
14	1	1	1	0	$A_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
15	1	1	1	1	$A_3A_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

THE 74HC154 1-OF-16 DECODER

The 74HCl54 is a good example of an IC decoder. There is an enable function (EN) provided on this device, which is implemented with a NOR gate used as a negative-AND. A LOW level on each chip select input, CS1' and CS 2' , is required in order to make the enable gate output (EN) HIGH. The enable gate output is connected to an input of each NAND gate in the decoder, so it must be HIGH for the NAND gates to be enabled. If the enable gate is not activated by a LOW on both inputs, then all sixteen decoder outputs (Y) will be HIGH regardless of the states of the four input variables.



(a) Pin diagram

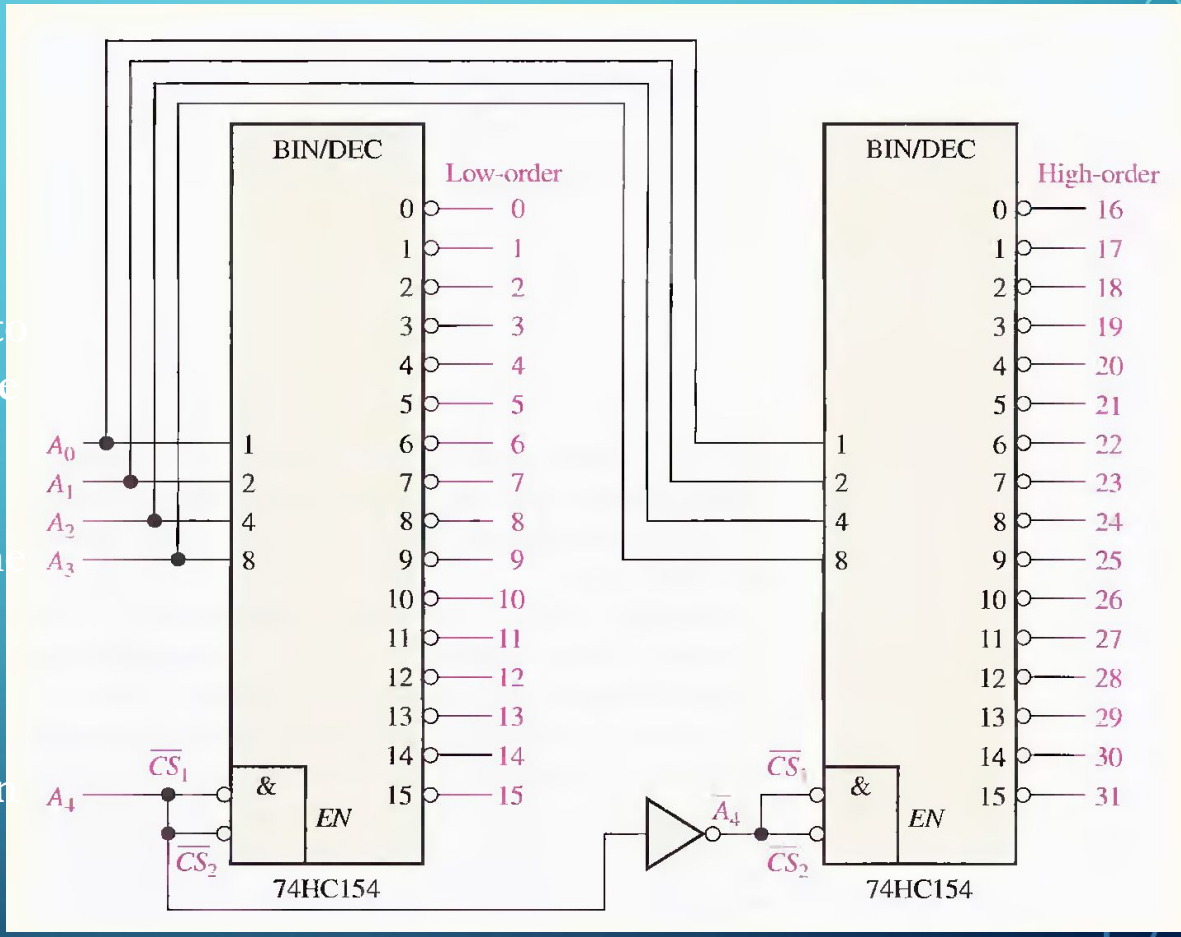


(b) Logic symbol

EXAMPLE

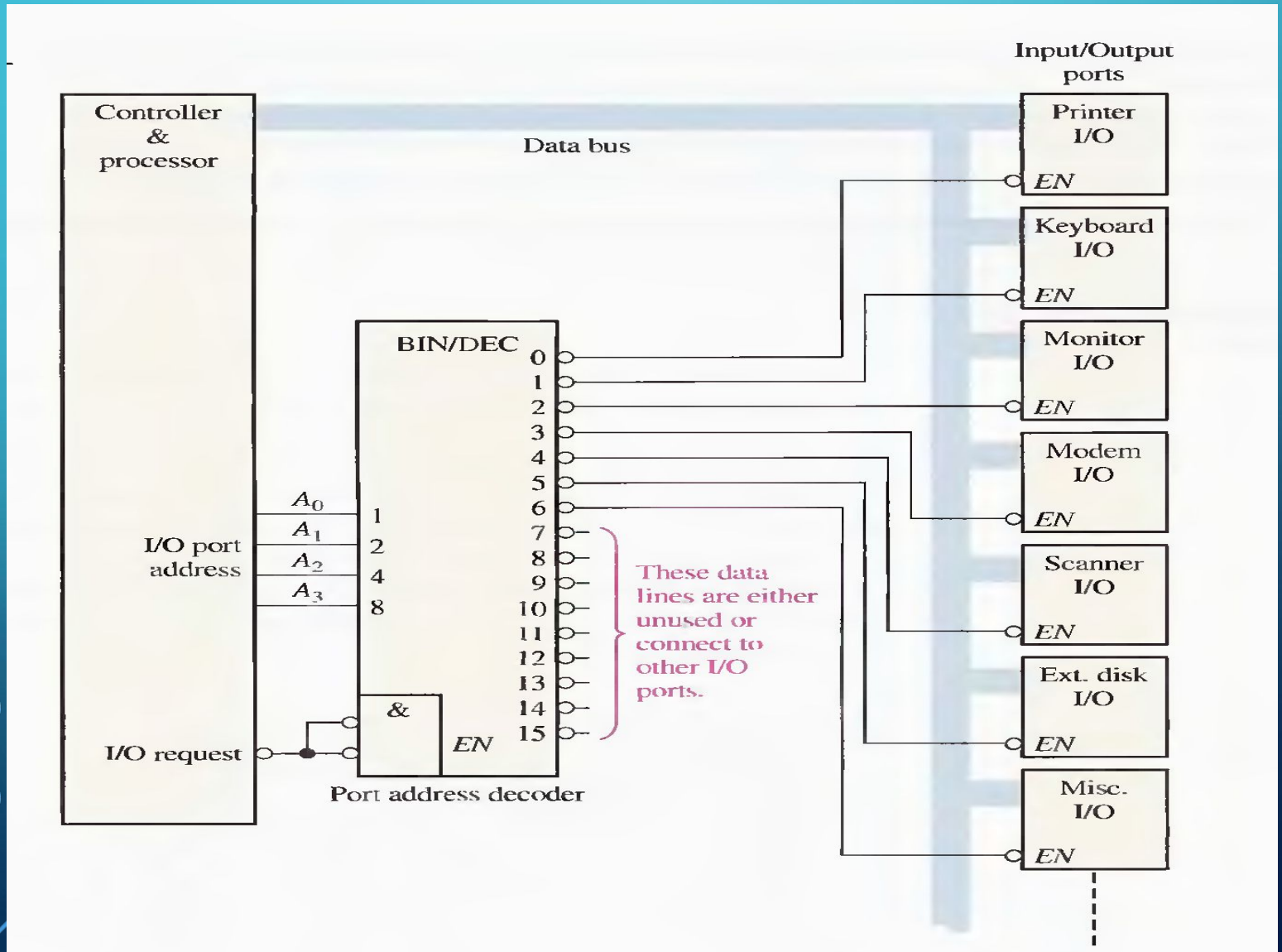
A certain application requires that a 5-bit number be decoded. Use 74HC154 decoders to implement the logic. The binary number is represented by the format $A_4A_3A_2A_1A_0$.

Since the 74HC154 can handle only four bits, two decoders must be use d to decode five bits. The fifth bit, A 4 , is connected to the chip select inputs, CSl' and CS 2' , of one decoder, and A4 is connected to the CS l'and CS 2' inputs of the other decoder, as shown in Figure . When the decimal number is 15 or less, A4 = 0, the low-order decoder is enabled, and the high-order decoder is disabled. When the decimal number is greater than 15, A4 = I so A4' = 0, the high-order decoder is enabled. and the low -order decoder is disabled.



An Application

A simplified computer I/O port system with a port address decoder with only four address lines shown.



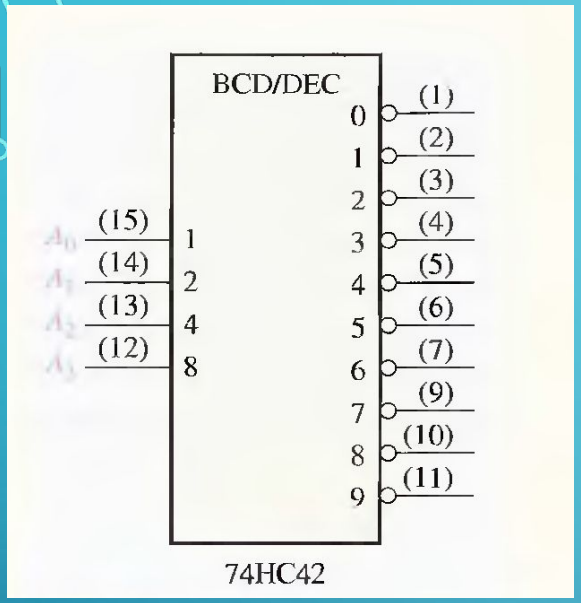
The BCD-to-Decimal Decoder

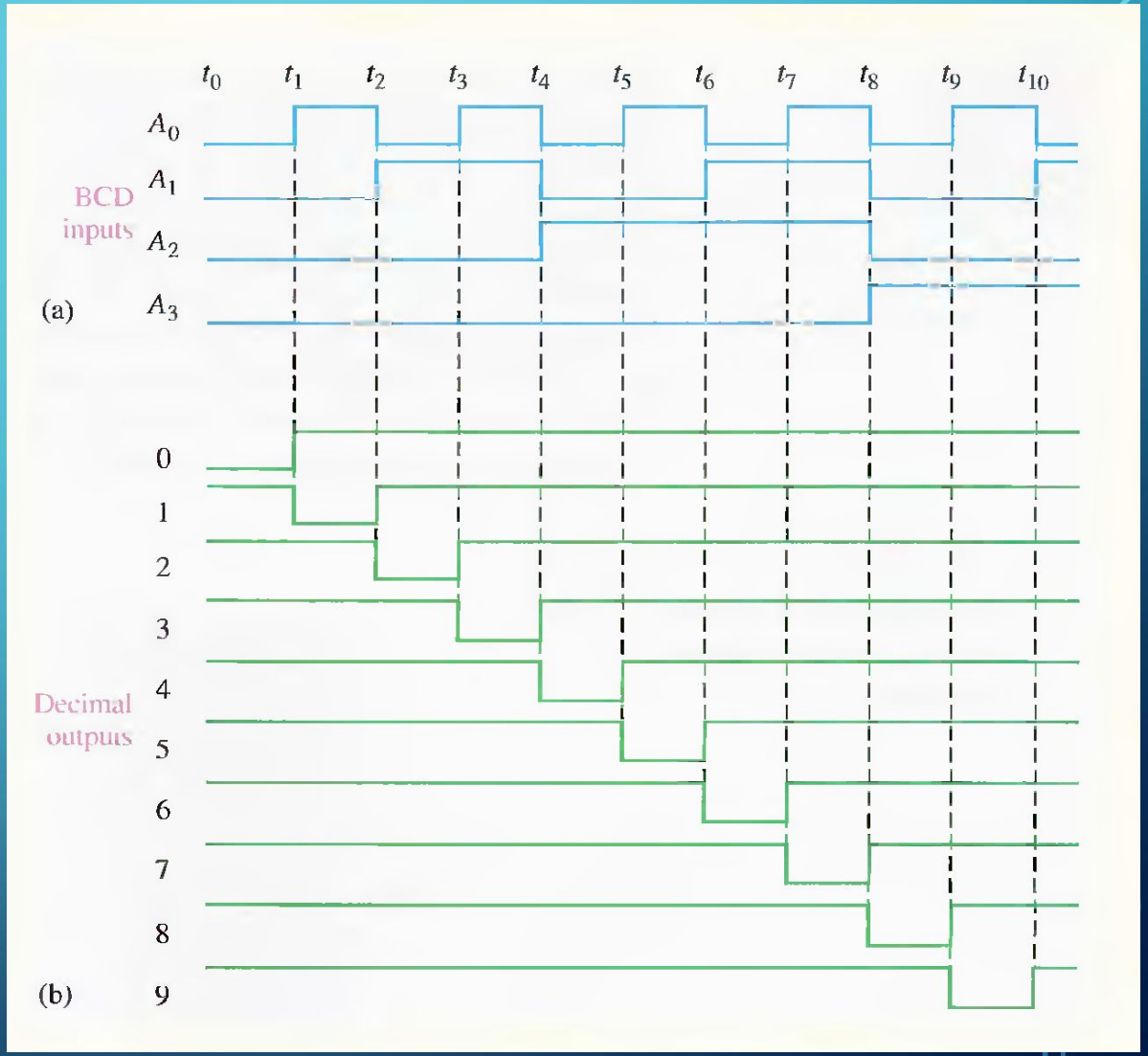
The BCD-to-decimal decoder converts each BCD code (8421 code) into one of ten possible decimal digit indications. It is frequently referred as a *4-line-to-10-line decoder* or a *1-of-10 decoder.*

BCD decoding functions.

DECIMAL DIGIT	BCD CODE A_3	A_2	A_1	A_0	DECODING FUNCTION
0	0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$
1	0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$
2	0	0	1	0	$\overline{A}_3\overline{A}_2A_1\overline{A}_0$
3	0	0	1	1	$\overline{A}_3\overline{A}_2A_1A_0$
4	0	1	0	0	$\overline{A}_3A_2\overline{A}_1\overline{A}_0$
5	0	1	0	1	$\overline{A}_3A_2\overline{A}_1A_0$
6	0	1	1	0	$\overline{A}_3A_2A_1\overline{A}_0$
7	0	1	1	1	$\overline{A}_3A_2A_1A_0$
8	1	0	0	0	$A_3\overline{A}_2\overline{A}_1\overline{A}_0$
9	1	0	0	1	$A_3\overline{A}_2\overline{A}_1A_0$

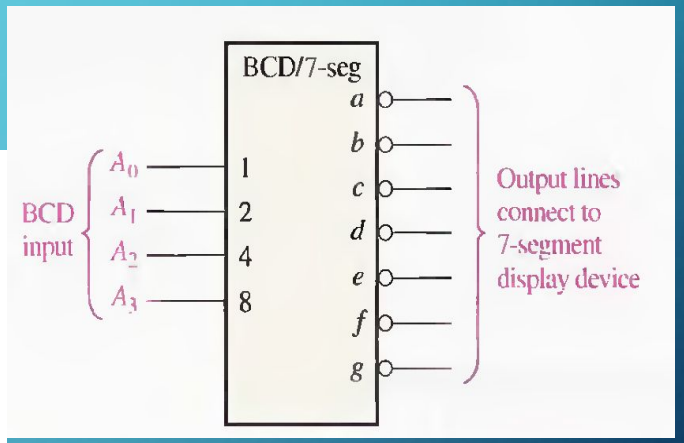
The 74HC42 is an integrated circuit BCD-to-decimal decoder.





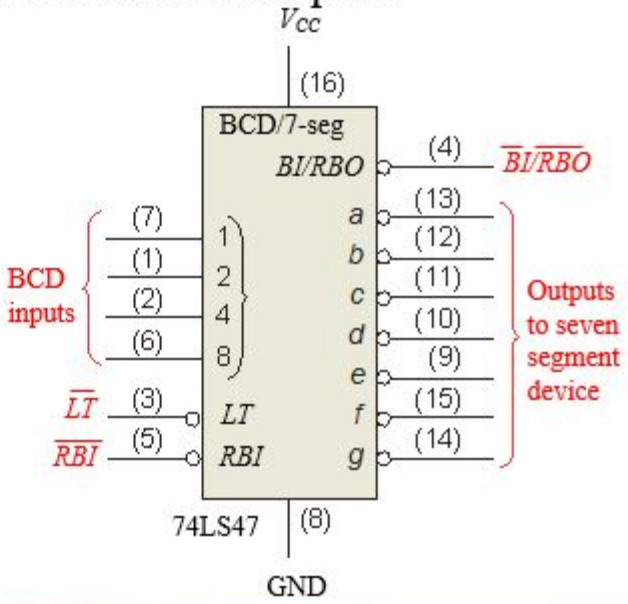
The BCD-to-7-Segment Decoder

The BCD-to-7-segment decoder accepts the BCD code on its inputs and provides outputs to drive 7-segment display devices to produce a decimal readout. The logic diagram for a basic 7-segment decoder is shown in Figure 6–34.



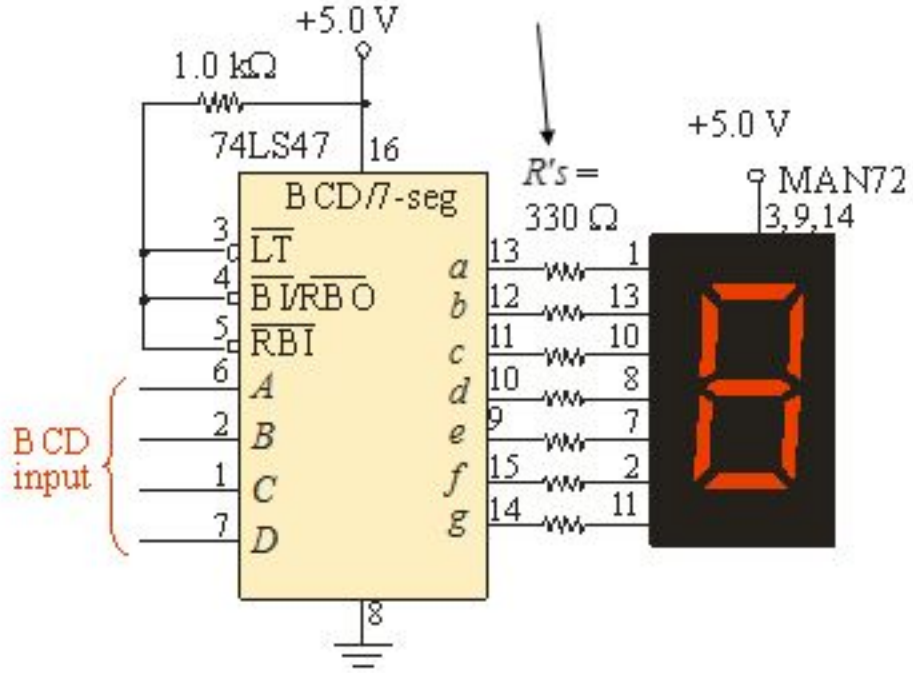
Another useful decoder is the 74LS47. This is a BCD-to-seven segment display with active LOW outputs.

The *a*-*g* outputs are designed for much higher current than most devices (hence the word driver in the name).

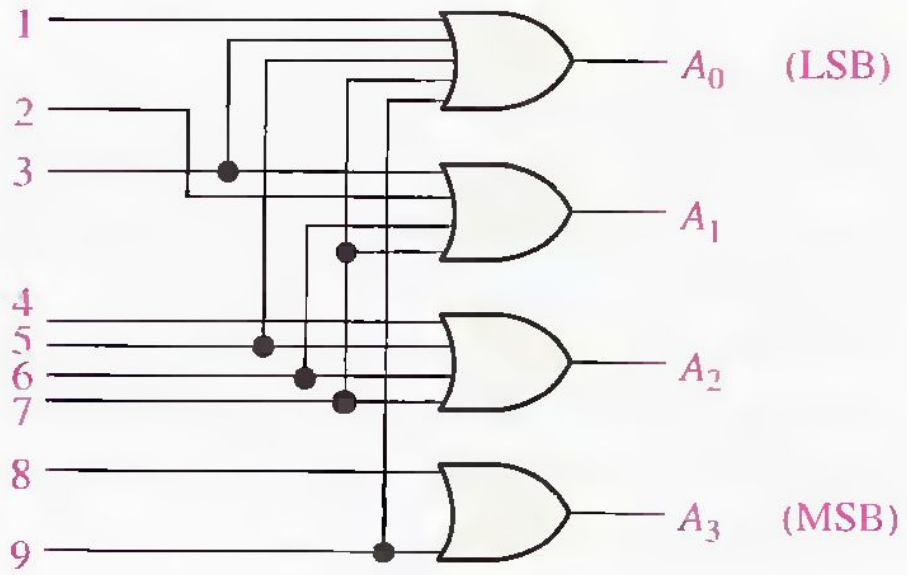


BCD Decoder/Driver

Here the 7447A is an connected to an LED seven segment display. Notice the current limiting resistors, required to prevent overdriving the LED display.



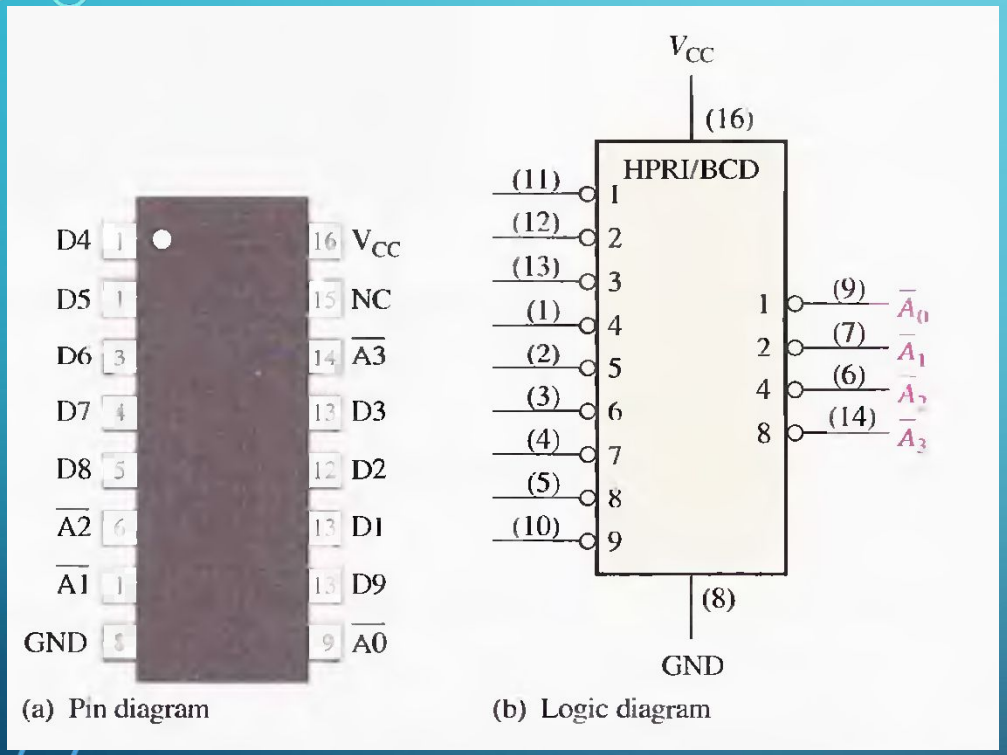
ENCODERS



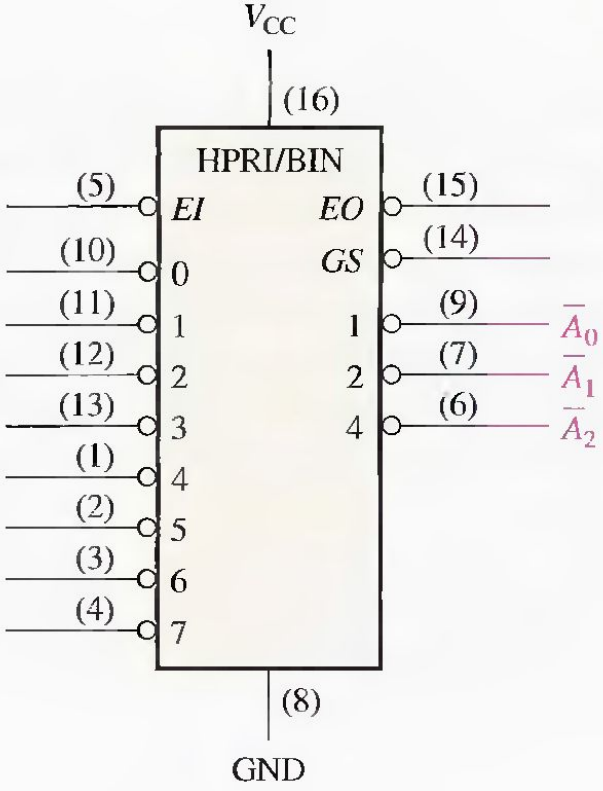
Basic logic diagram of a decimal-to-BCD encoder. A 0-digit input is not needed because the BCD outputs are all LOW when there are no HIGH inputs.

The Decimal-to-BCD Priority Encoder This type of encoder performs the same basic encoding function as previously discussed. A **priority encoder** also offers additional flexibility in that it can be used in applications that require priority detection. The priority function means that the encoder will produce a BCD output corresponding to the *highest-order decimal digit* input that is active and will ignore any other lower-order active inputs. For instance, if the 6 and the 3 inputs are both active, the BCD output is 0110 (which represents decimal 6).

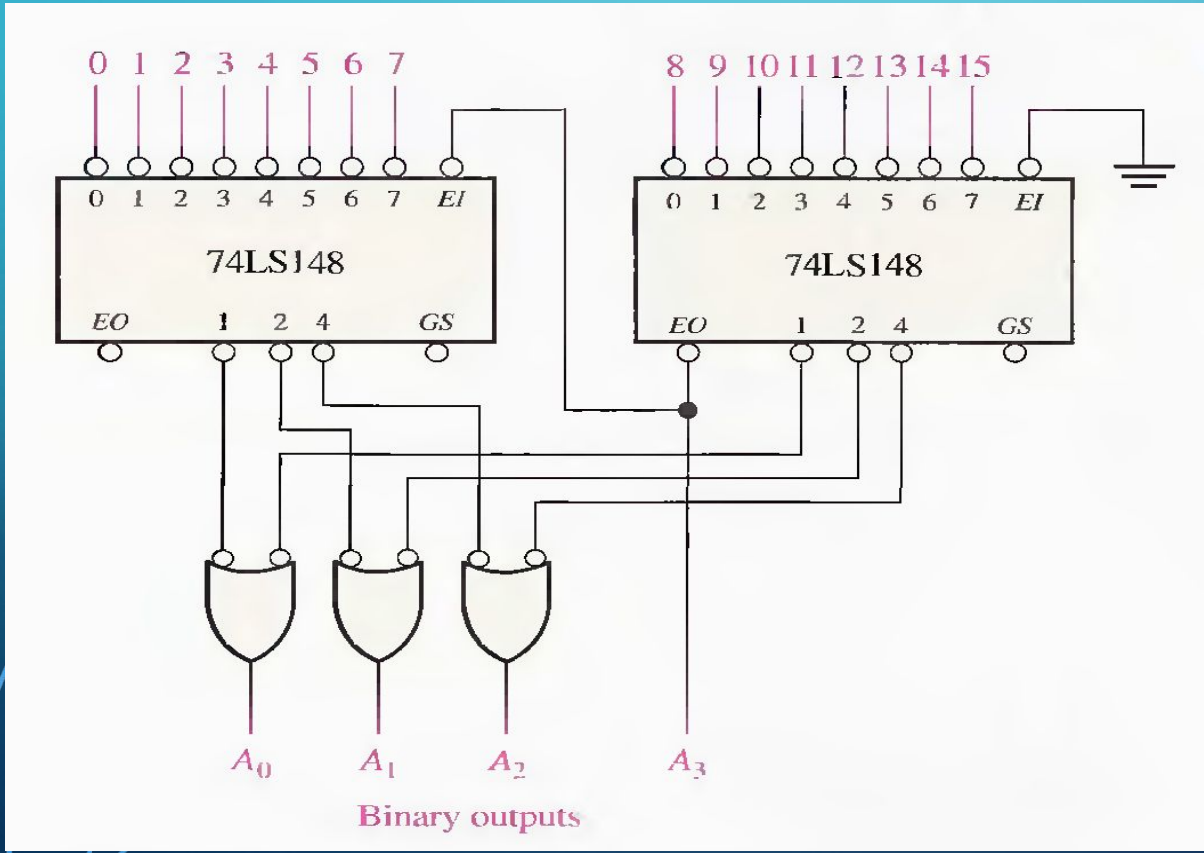
THE 74HC147 DECIMAL-TO-BCD ENCODER



Logic symbol for the 74LS148 8-line-to-3-line encoder.



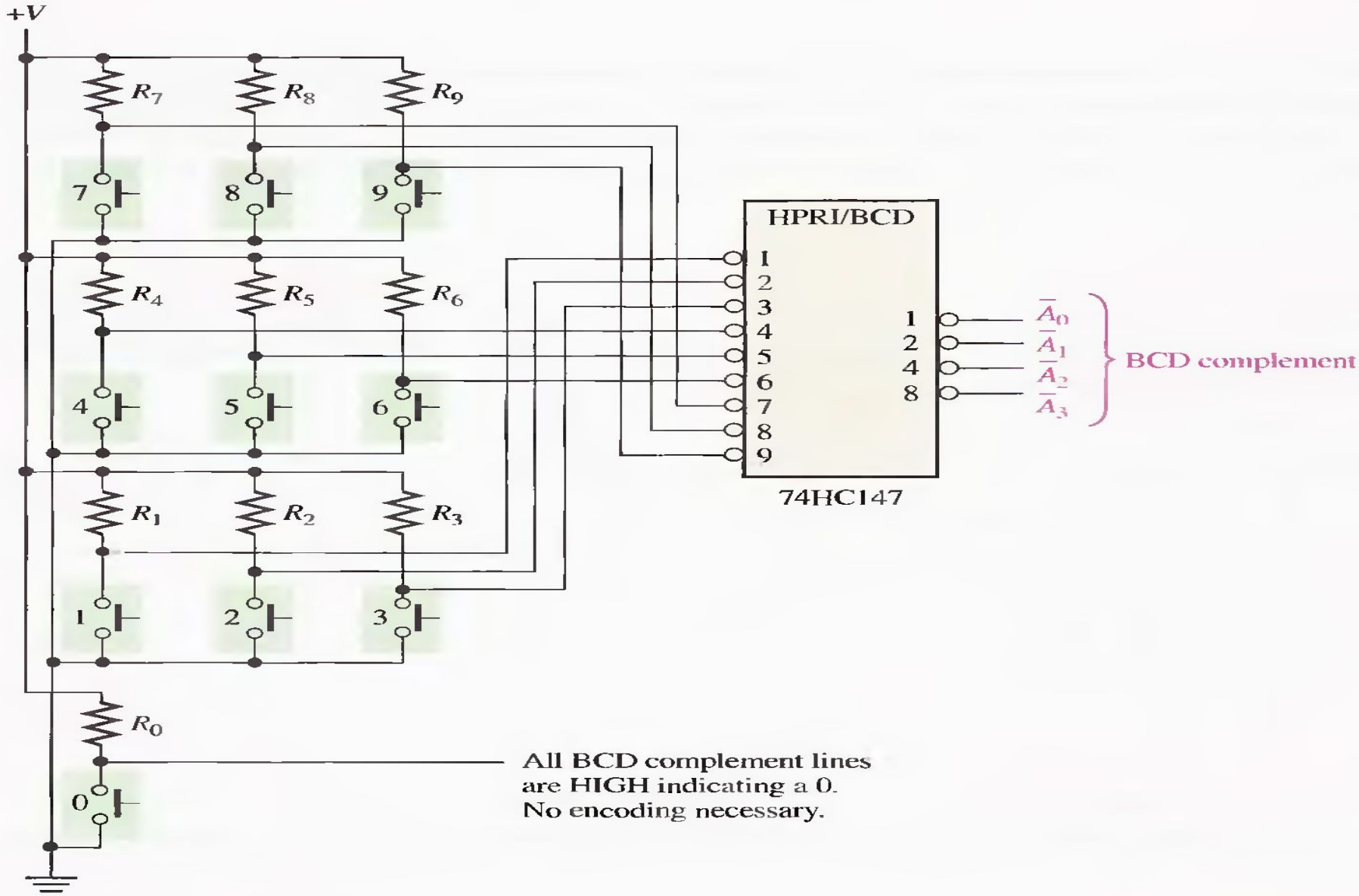
The 74LS148 can be expanded to a 16-1ine-to-4-line encoder by connecting the EO of the higher-order encoder to the EI of the lower-order encoder and negative-ORing the corresponding binary outputs as shown in Figure . The EO is used as the fourth and MSB. This particular configuration produces active-HIGH outputs for the 4-bit binary number.



A 16-line-to-4 line encoder using 74LS148s and external logic.

An Application

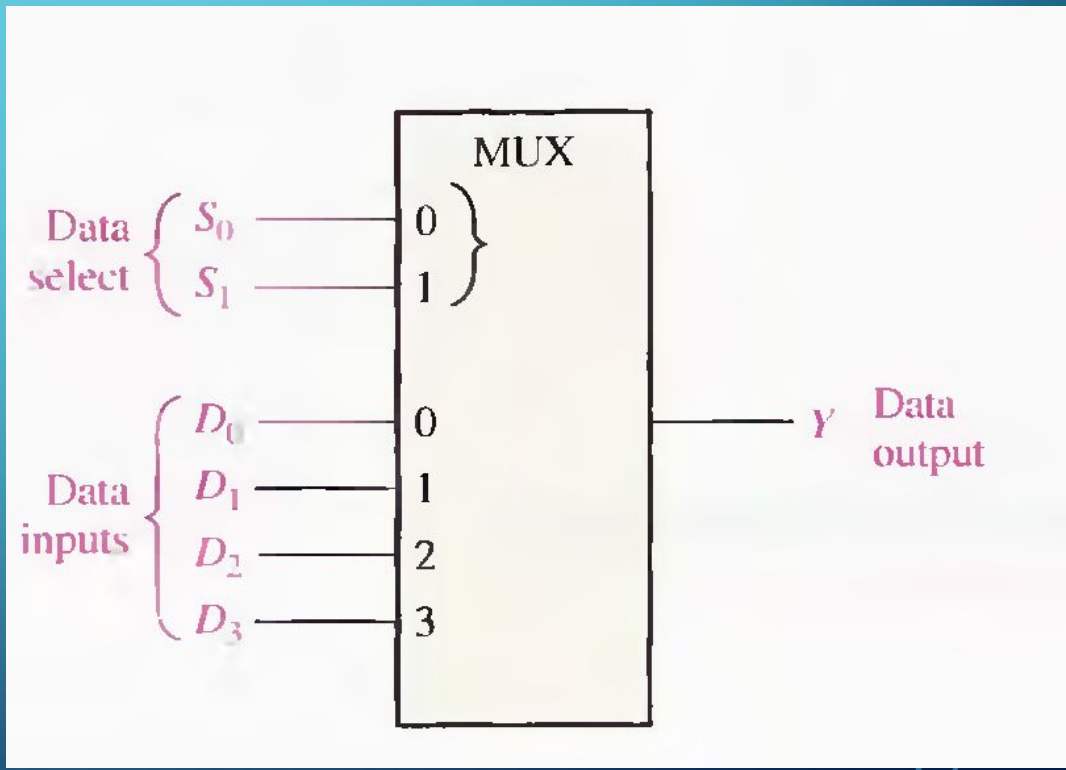
A classic application example is a keyboard encoder.



MULTIPLEXERS (DATA SELECTORS)

A **multiplexer (MUX)** is a device that allows digital information from several sources to be routed onto a single line for transmission over that line to a common destination. The basic multiplexer has several data-input lines and a single output line. It also has data-select inputs, which permit digital data on any one of the inputs to be switched to the output line. Multiplexers are also known as data selectors.

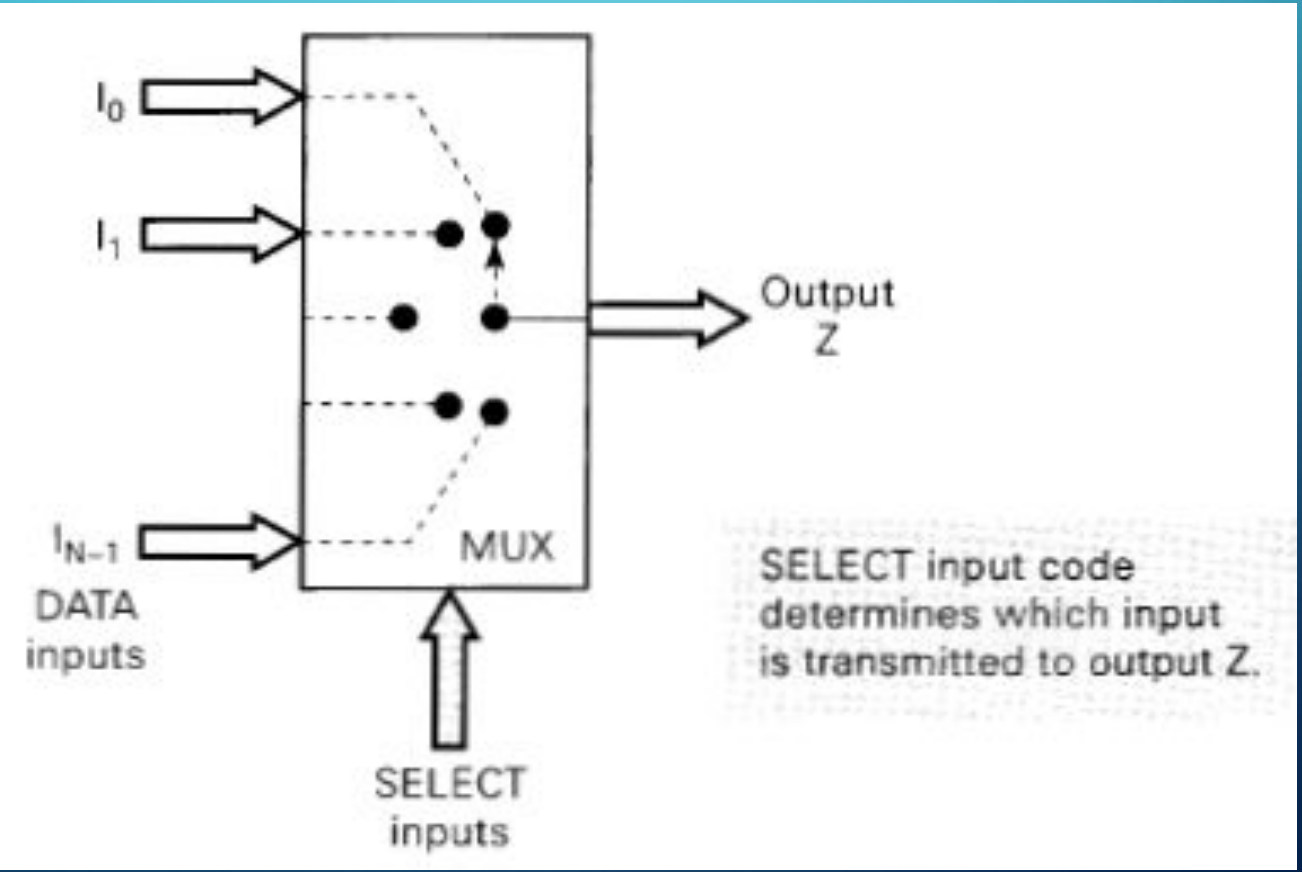
Logic symbol for a 1-of-4 data selector/multiplexer.



DATA-SELECT INPUTS S_1	S_0	INPUT SELECTED
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

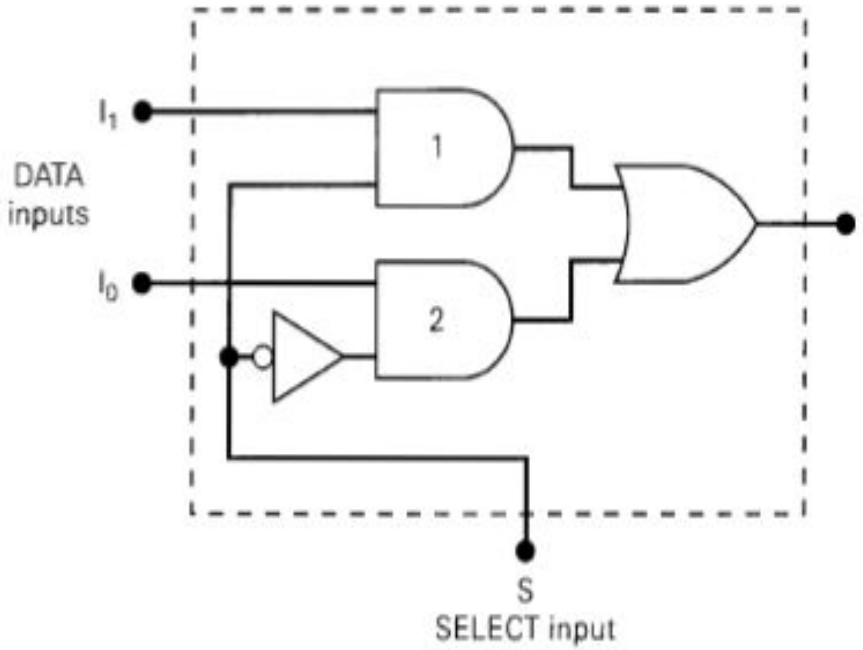
MULTIPLEXERS (DATA SELECTORS)

A **multiplexer (MUX)** is a device that allows digital information from several sources to be routed onto a single line for transmission over that line to a common destination. The basic multiplexer has several data-input lines and a single output line. It also has data-select inputs, which permit digital data on any one of the inputs to be switched to the output line. Multiplexers are also known as data selectors.



Basic Two-Input Multiplexer

$$Z = I_0\bar{S} + I_1S$$



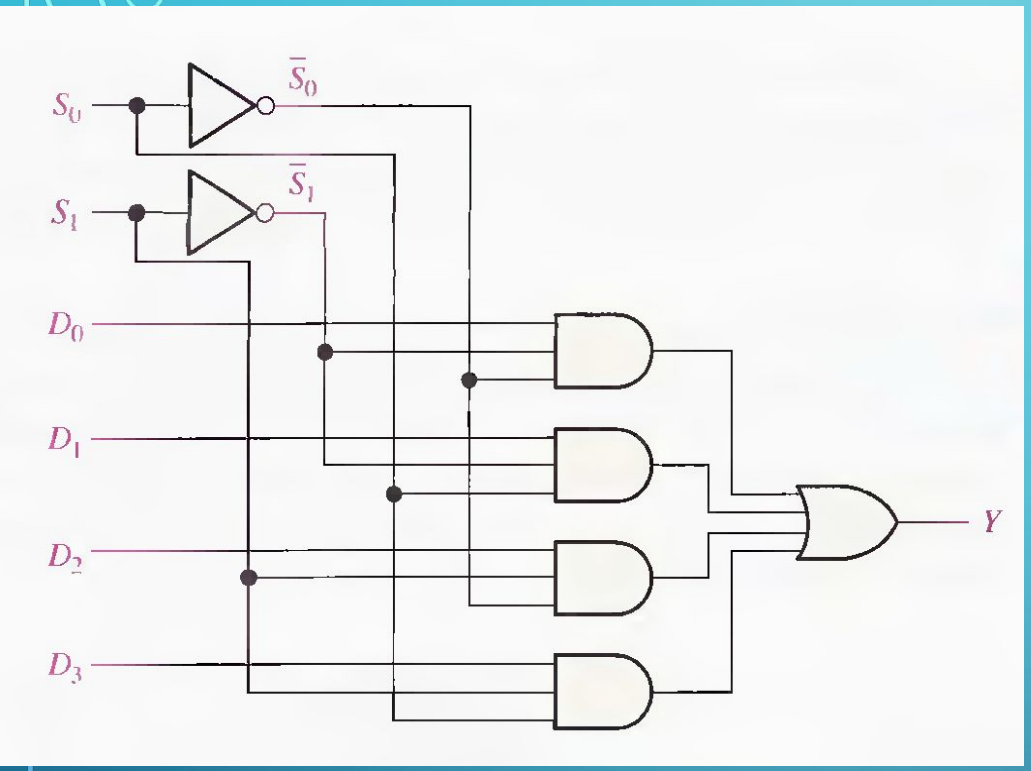
With $S = 0$, this expression becomes

$$Z = I_0 \cdot 1 + I_1 \cdot 0 \quad \text{(gate 2 enabled)}$$
$$= I_0$$

With $S = 1$, the expression becomes

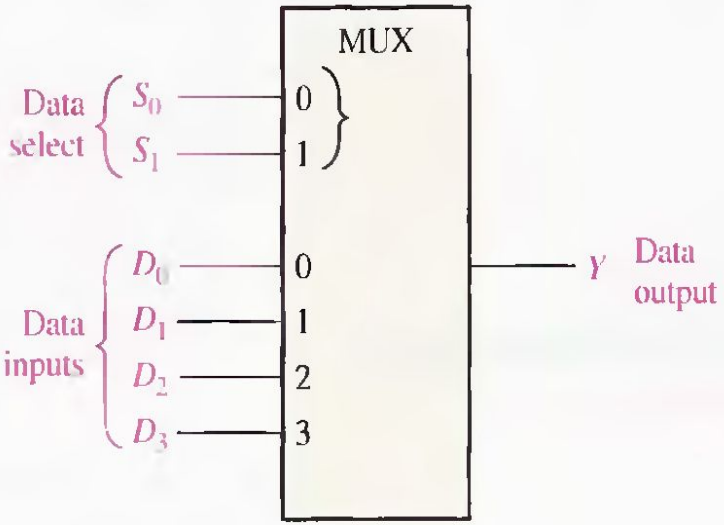
$$Z = I_0 \cdot 0 + I_1 \cdot 1 = I_1 \quad \text{(gate 1 enabled)}$$

Logic symbol for a 1-of-4 data selector/multiplexer.



DATA-SELECT INPUTS S_1	S_0	INPUT SELECTED
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

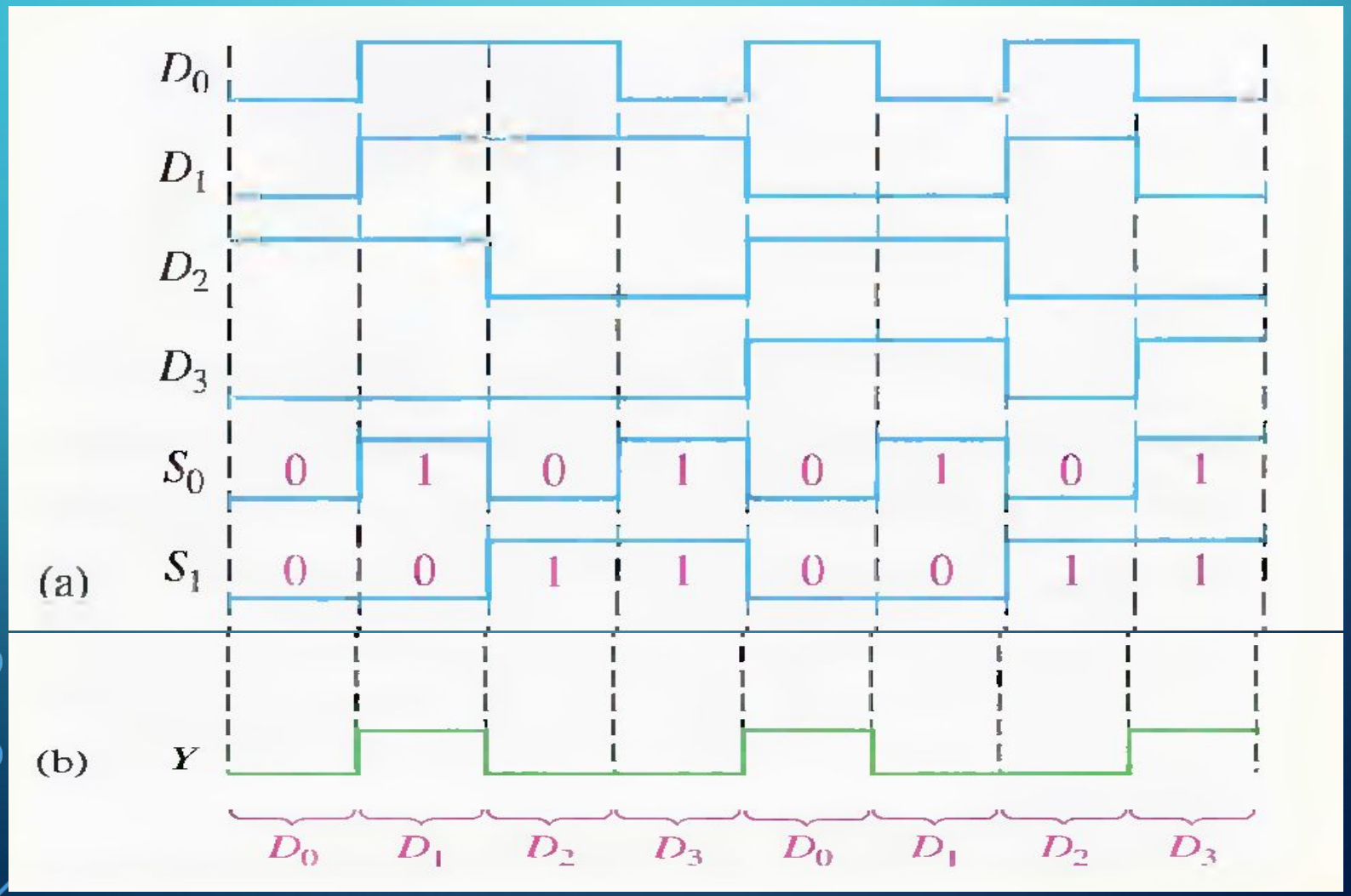
Figure -2



$$Y = D_0\overline{S}_1\overline{S}_0 + D_1\overline{S}_1S_0 + D_2S_1\overline{S}_0 + D_3S_1S_0$$

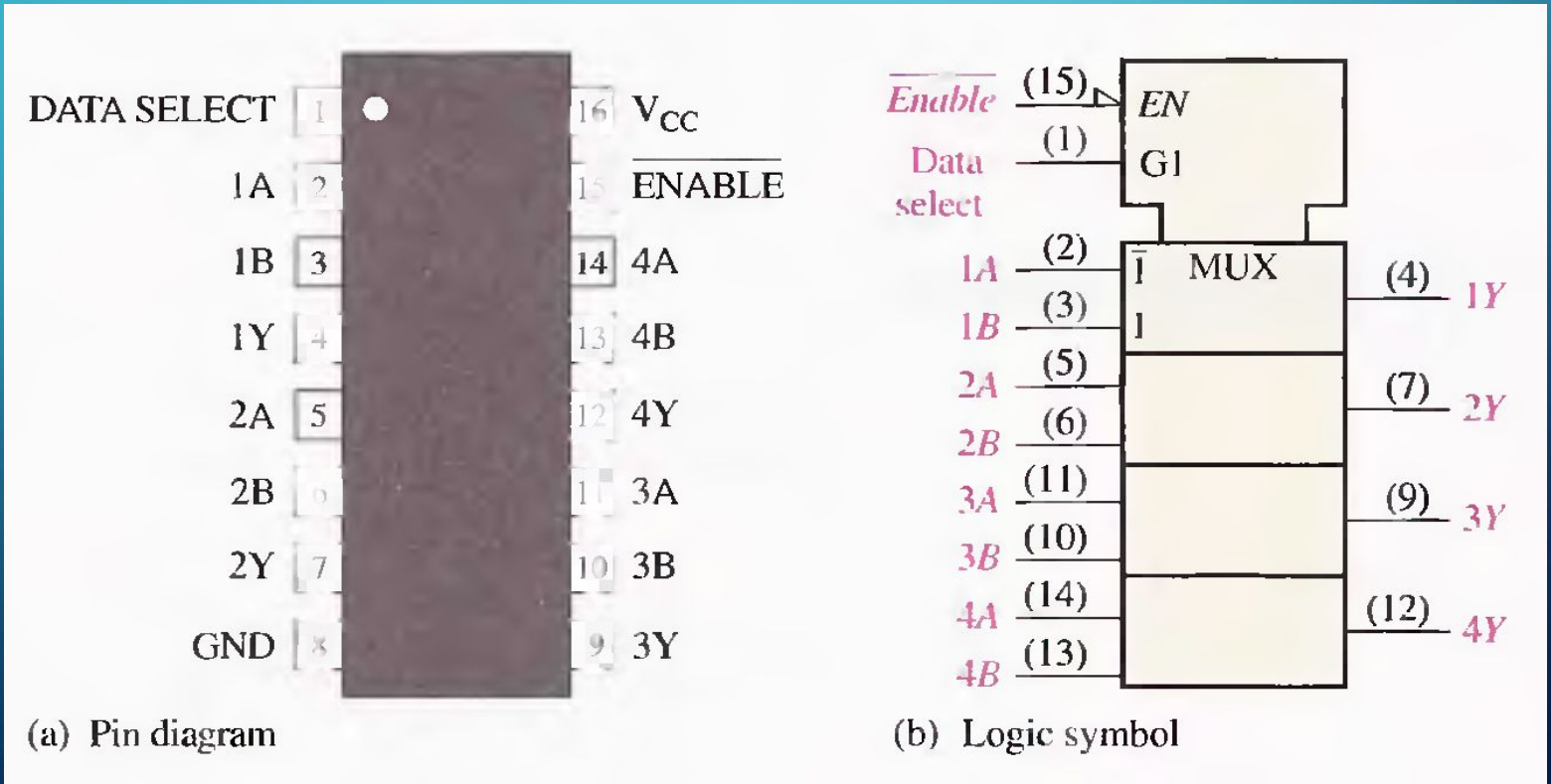
Example:

The data-input and data-select waveforms in Figure are applied to the multiplexer in Figure -2 . Determine the output waveform in relation to the inputs.



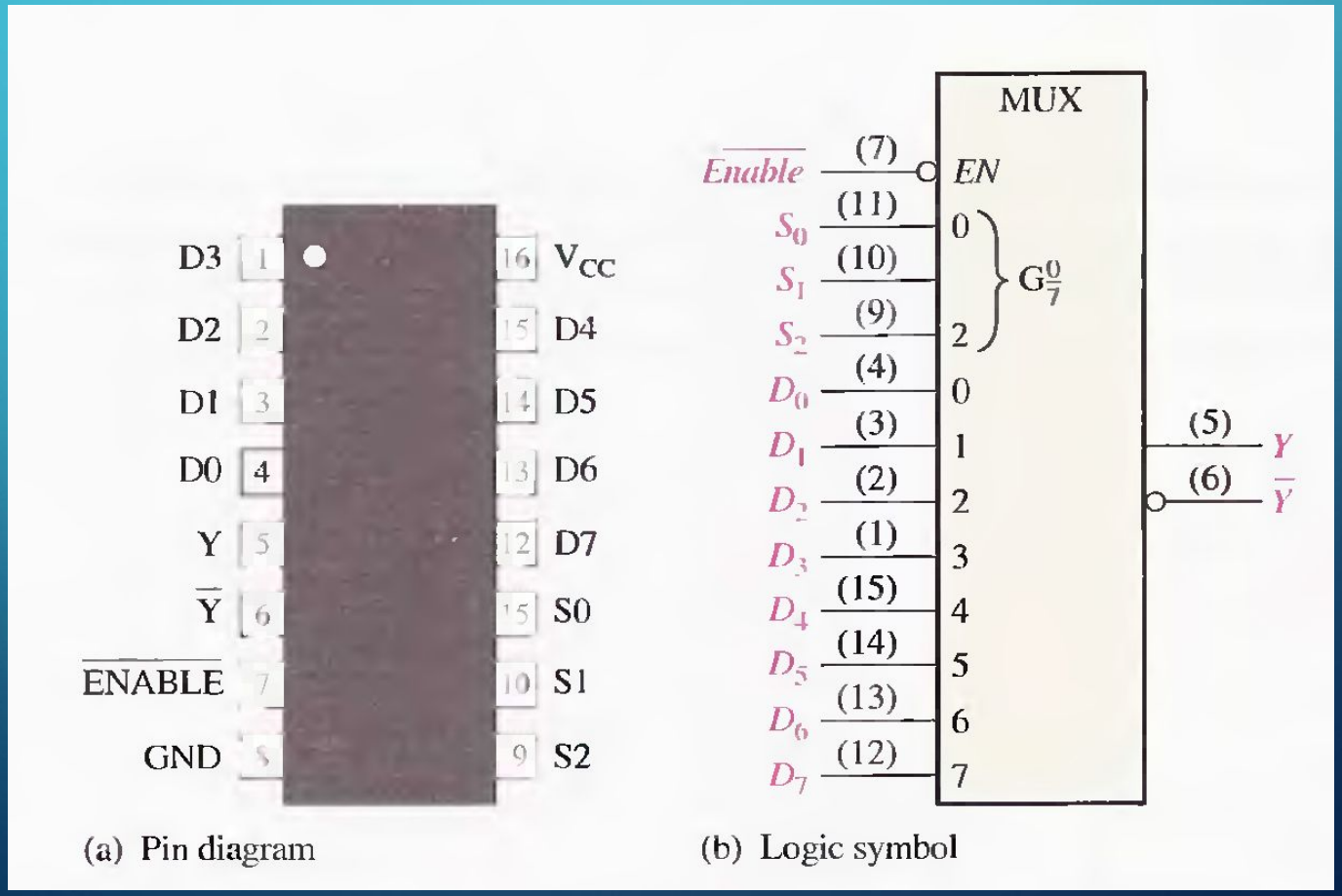
THE 74HC157 QUAD 2-INPUT DATA SELECTOR/MULTIPLEXER

The 74HC157 consists of four separate 2-input multiplexers. Each of the four multiplexers shares a common data-select line and a common Enable. Because there are only two inputs to be selected in each multiplexer, a single data-select input is sufficient. A LOW on the Enable input allows the selected input data to pass through to the output. A HIGH on the Enable input prevents data from going through to the output; that is, it disables the multiplexers. This device may be available in other CMOS or TTL families.



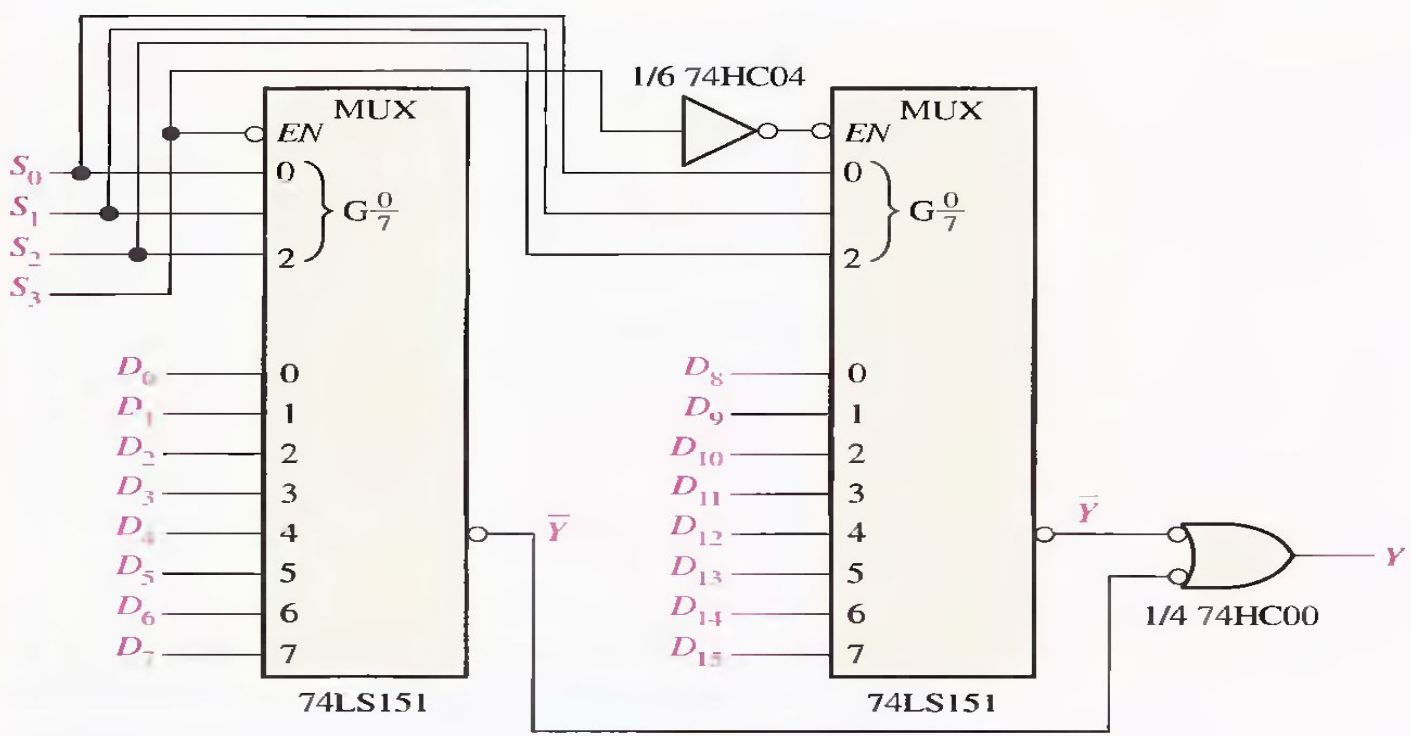
THE 74LS151 8-INPUT DATA SELECTOR/MULTIPLEXER

The 74LS151 has eight data inputs (D_0–D_7) and, therefore, three data-select or address input lines (S_0–S_2). Three bits are required to select any one of the eight data inputs ($2^3 = 8$). A LOW on the *Enable* input allows the selected input data to pass through to the output.



Use 74LS151s and any other logic necessary to multiplex 16 data lines onto a single data-output line.

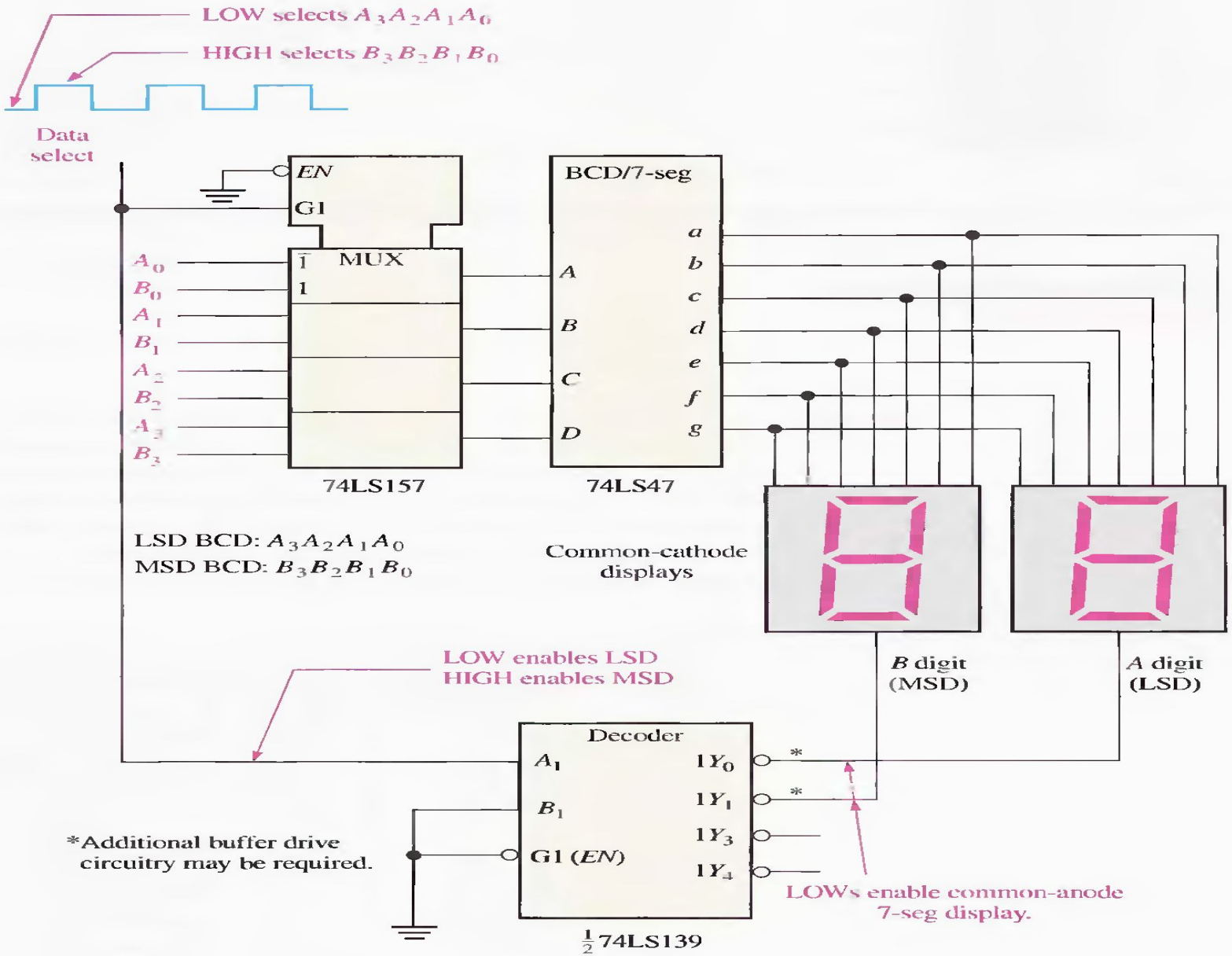
An implementation of this system is shown in Figure 6–51. Four bits are required to select one of 16 data inputs ($2^4 = 16$). In this application the $\overline{Enable}$ input is used as the most significant data-select bit. When the MSB in the data-select code is LOW, the left 74LS151 is enabled, and one of the data inputs (D_0 through D_7) is selected by the other three data-select bits. When the data-select MSB is HIGH, the right 74LS151 is enabled, and one of the data inputs (D_8 through D_{15}) is selected. The selected input data are then passed through to the negative-OR gate and onto the single output line.



Determine the codes on the select inputs required to select each of the following data inputs: D_0, D_4, D_8, and D_{13}.

Applications

A 7-Segment Display Multiplexer



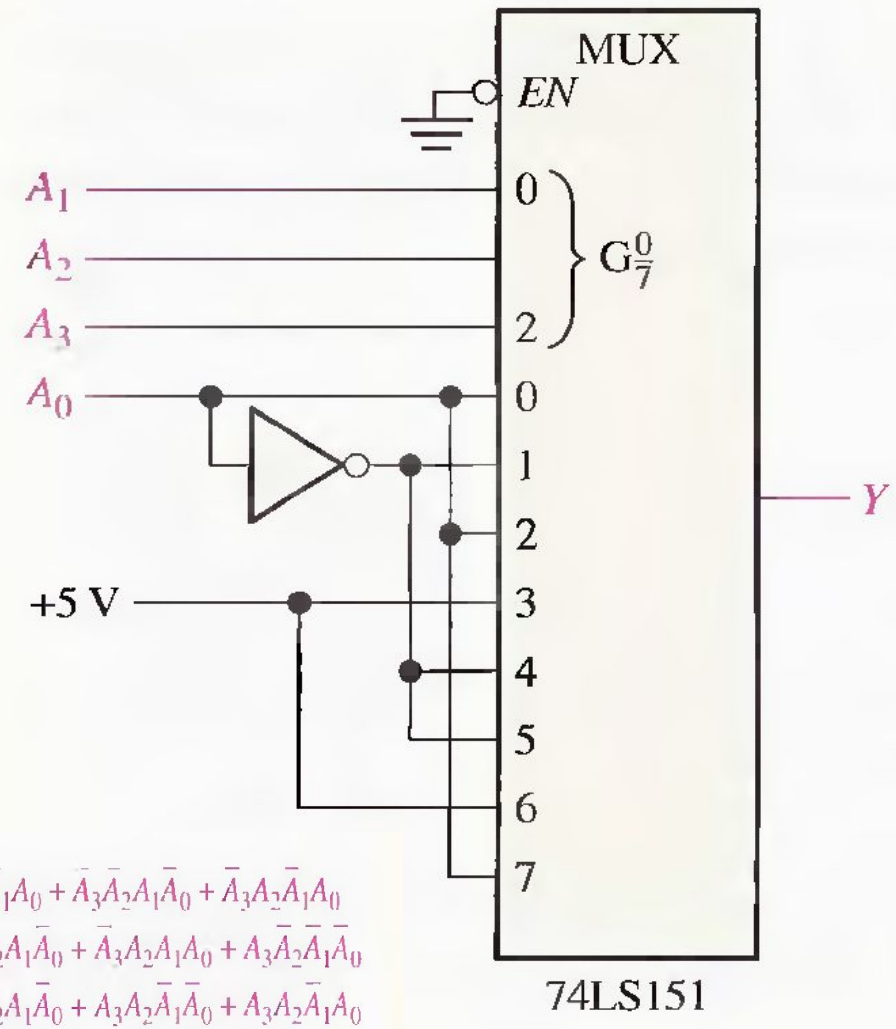
A 4-variable truth table has sixteen combinations of input variables.

A 4-variable truth table has sixteen combinations of input variables. When an 8-bit data selector is used, each input is selected twice: the first time when A_0 is 0 and the second time when A_0 is 1. With this in mind, the following rules can be applied (Y is the output, and A_0 is the least significant bit):

1. If $Y = 0$ both times a given data input is selected by a certain combination of the input variables, $A_3A_2A_1$, connect that data input to ground (0).
2. If $Y = 1$ both times a given data input is selected by a certain combination of the input variables, $A_3A_2A_1$, connect the data input to $+V$ (1).
3. If Y is different the two times a given data input is selected by a certain combination of the input variables, $A_3A_2A_1$, and if $Y = A_0$, connect that data input to A_0.
4. If Y is different the two times a given data input is selected by a certain combination of the input variables, $A_3A_2A_1$, and if $Y = \overline{A}_0$, connect that data input to $\overline{A}_0$.

Implement the logic function in Table 6–10 by using a 74LS151 8-input data selector/multiplexer. Compare this method with a discrete logic gate implementation.

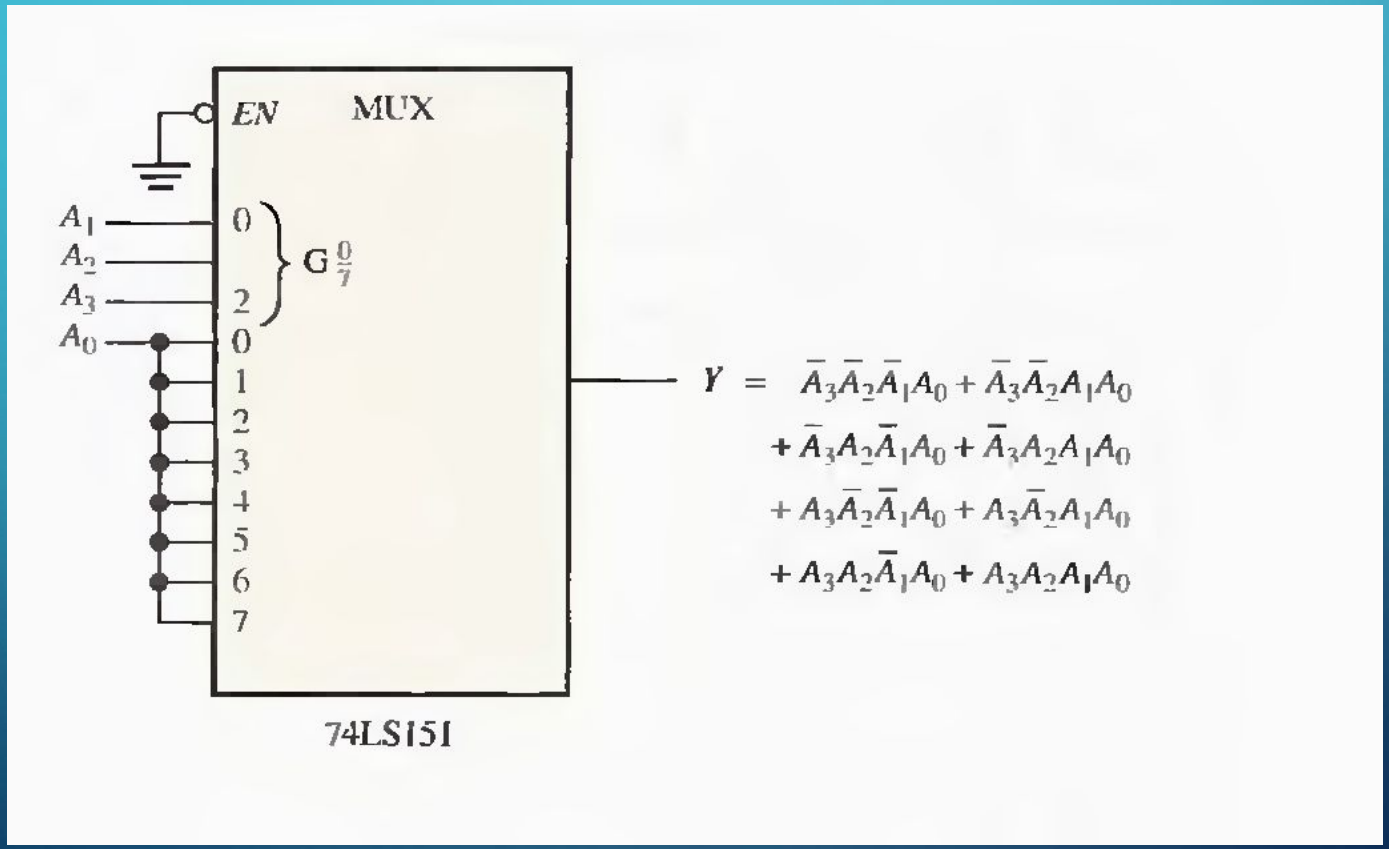
DECIMAL DIGIT	INPUTS A_3	A_2	A_1	A_0	OUTPUT Y
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1



$$Y = \bar{A}_3\bar{A}_2\bar{A}_1A_0 + \bar{A}_3\bar{A}_2A_1\bar{A}_0 + \bar{A}_3A_2\bar{A}_1A_0 + \bar{A}_3A_2A_1\bar{A}_0 + \bar{A}_3A_2A_1A_0 + A_3\bar{A}_2\bar{A}_1\bar{A}_0 + A_3\bar{A}_2A_1\bar{A}_0 + A_3A_2\bar{A}_1\bar{A}_0 + A_3A_2\bar{A}_1A_0 + A_3A_2A_1A_0$$

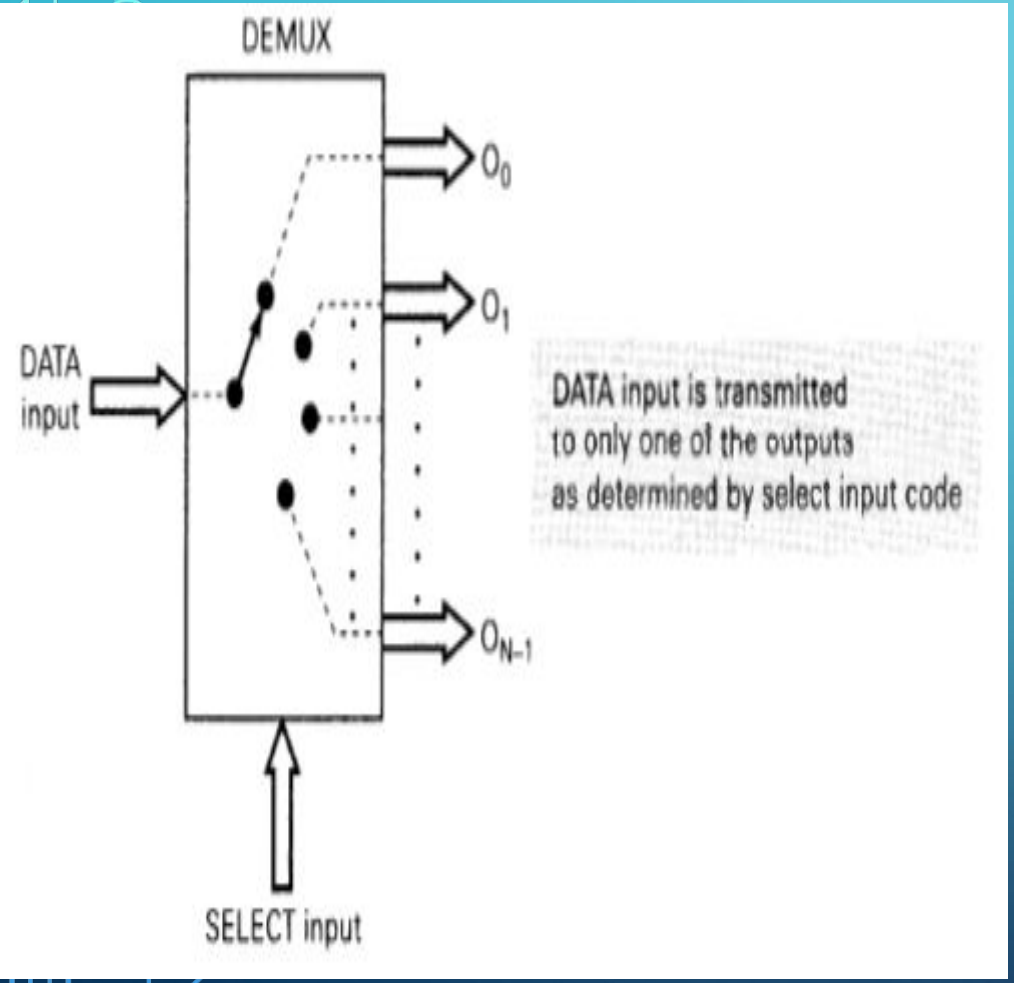
Example

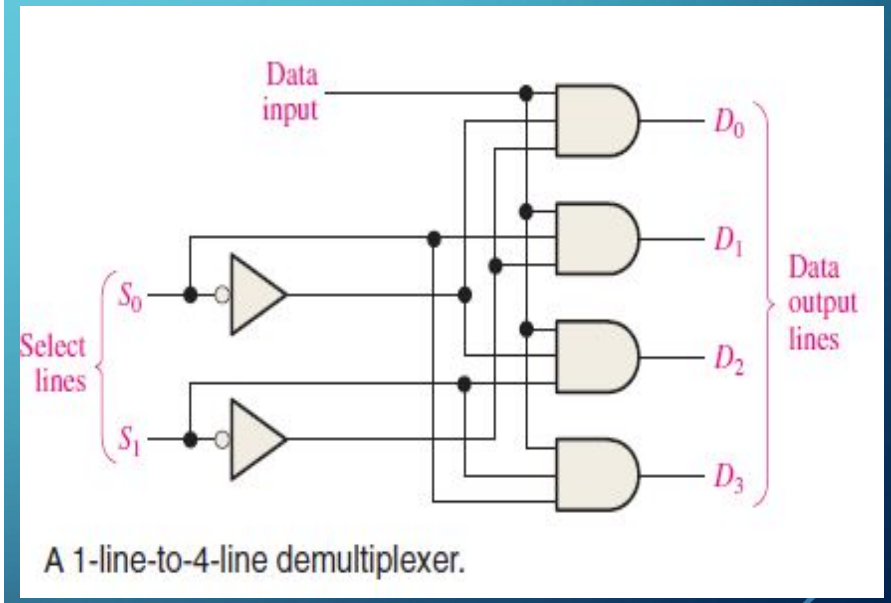
In Table 6–10, if $Y = 0$ when the inputs are all zeros and is alternately a 1 and a 0 for the remaining rows in the table, use a 74LS151 to implement the resulting logic function.



DEMULTIPLEXERS

A **demultiplexer (DEMUX)** basically reverses the multiplexing function. It takes digital information from one line and distributes it to a given number of output lines.



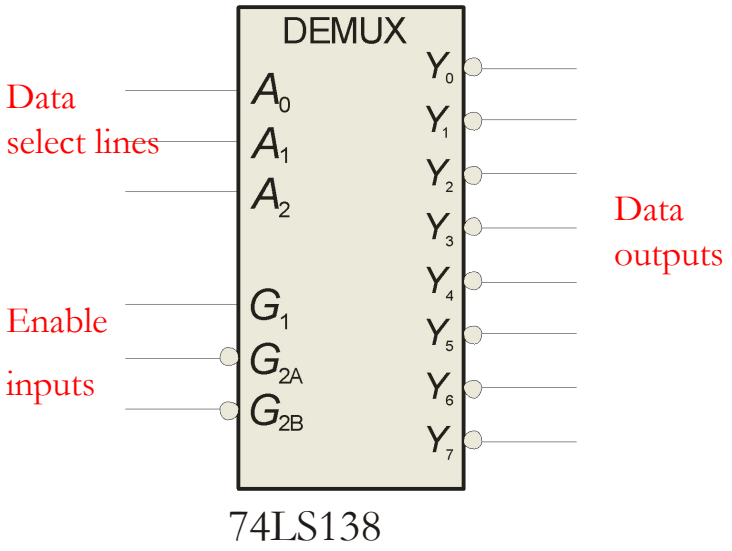


A 1-line-to-4-line demultiplexer.

Demultiplexers

A demultiplexer (DEMUX) performs the opposite function from a MUX. It switches data from one input line to two or more data lines depending on the select inputs.

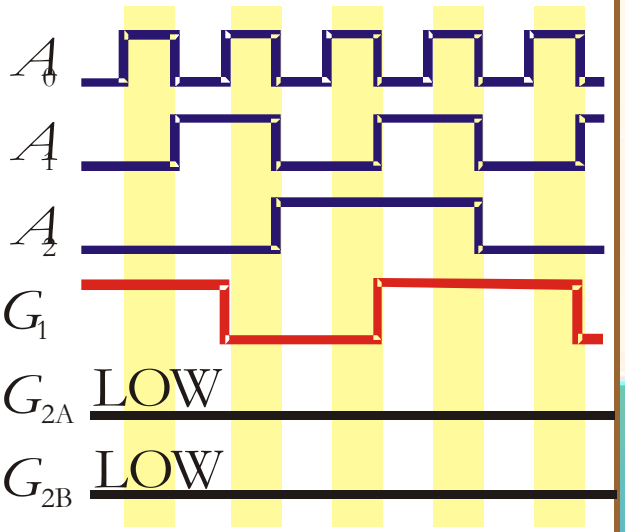
The 74LS138 was introduced previously as a decoder but can also serve as a DEMUX. When connected as a DEMUX, data is applied to one of the enable inputs, and routed to the selected output line depending on the select variables. Note that the outputs are active-LOW as illustrated in the following example…



Demultiplexers

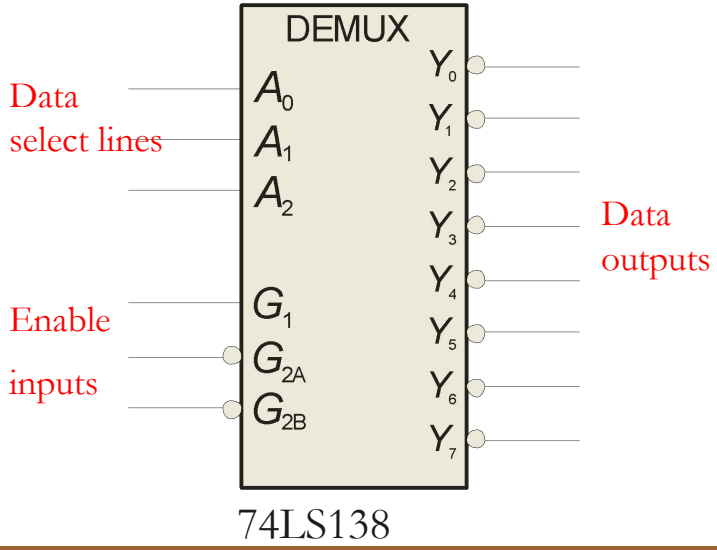
Example

Determine the outputs, given the inputs shown.

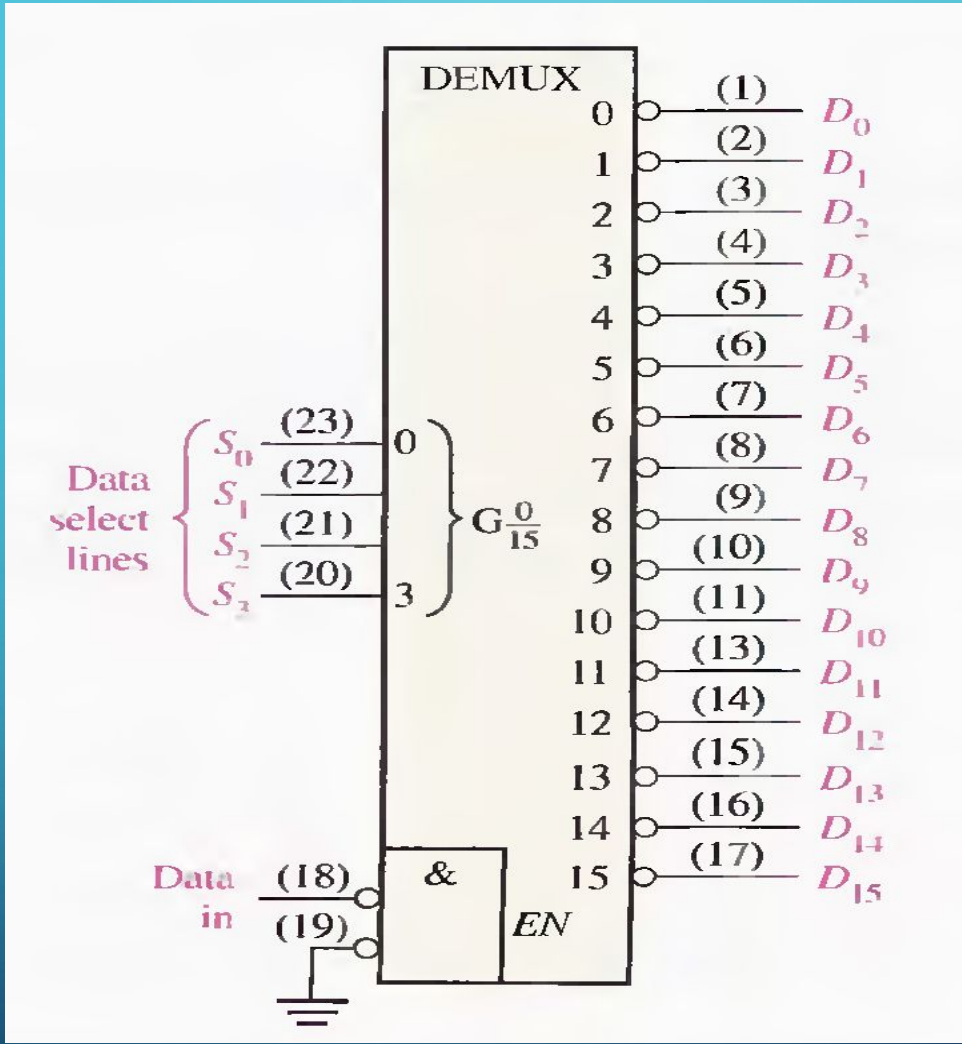


Solution

The output logic is opposite to the input because of the active-LOW convention. (Red shows the selected line).



THE 74HC154 DEMULTIPLEXER

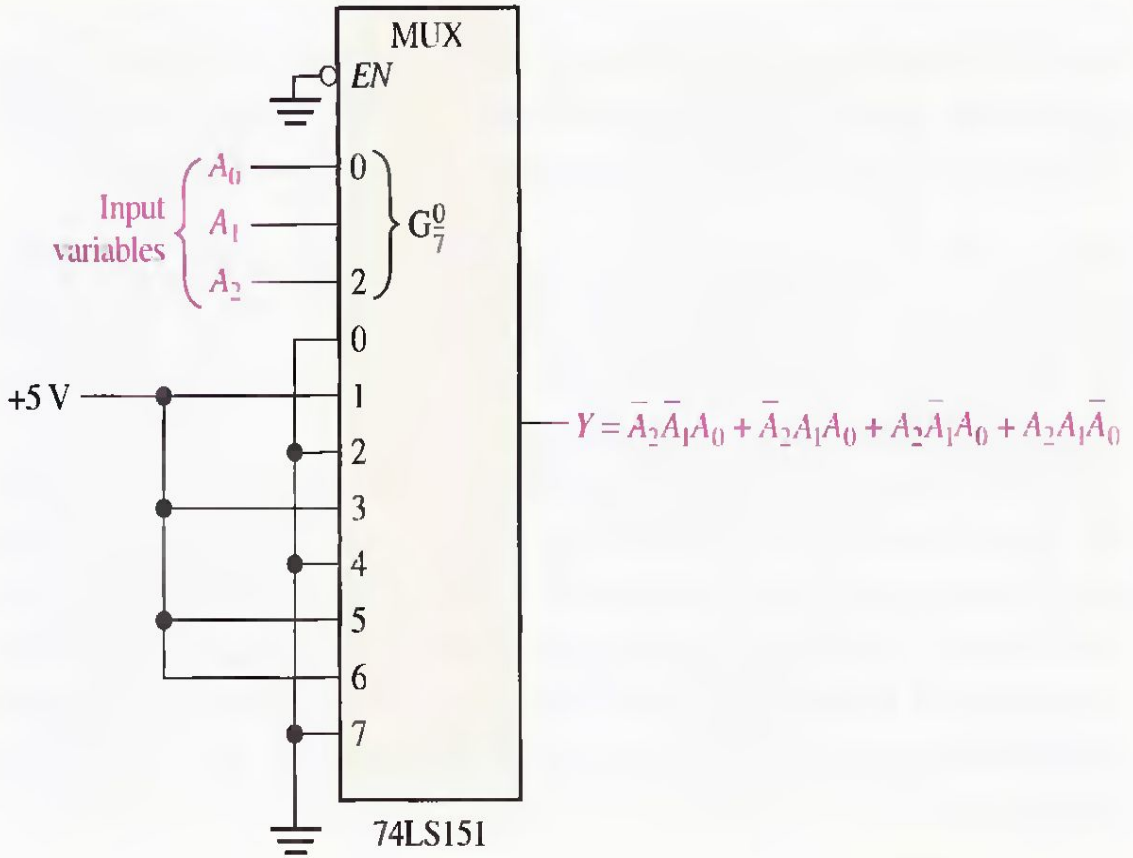


EXAMPLE

Implement the logic function specified in Table 6–9 by using a 74LS151 8-input data selector/multiplexer. Compare this method with a discrete logic gate implementation.

TABLE 6-9

INPUTS			OUTPUT
A_2	A_1	A_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0



Use the 74LS151 to implement the following expression:

$$Y = \overline{A}_2\overline{A}_1\overline{A}_0 + A_2\overline{A}_1\overline{A}_0 + \overline{A}_2A_1\overline{A}_0$$