Vietnam National University HCMC

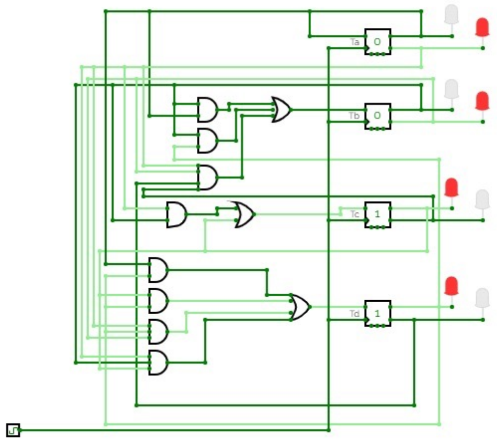
International University

School of Electrical Engineering

EE053IU

Digital Logic Design

Lecture 6: Functions of Combinational Logic



INSTRUCTOR: Dr. Vuong Quoc Bao

1. Half and Full Adders

The Half-Adder

- Recall the basic rules for binary addition.

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

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.

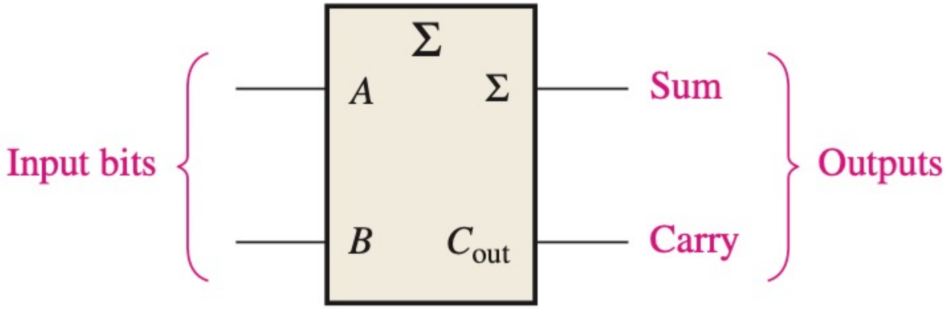


FIGURE 6–1 Logic symbol for a half-adder. Open file F06-01 to verify operation. *A Multisim tutorial is available on the website.*

Half-Adder Logic

- The expressions can be derived for the sum and the output carry as functions of the inputs.
- Notice that the output carry (C_{out}) is a 1 only when both A and B are 1s; therefore, C_{out} can be expressed as the AND of the input variables.

$$C_{out} = AB$$

- The sum can therefore be expressed as the exclusive-OR of the input variables.

$$\Sigma = A \oplus B$$

TABLE 6–1

Half-adder truth table.

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)

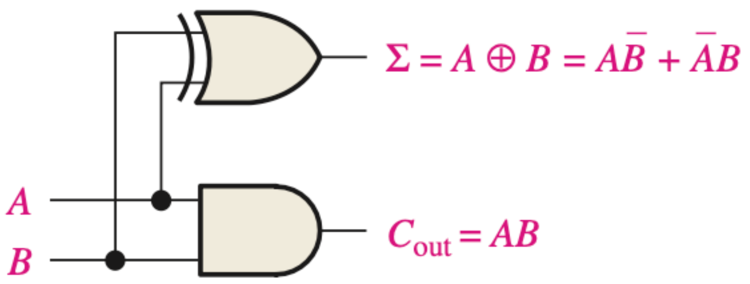


FIGURE 6–2 Half-adder logic diagram.

The Full-Adder

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

- The basic difference between a full-adder and a half-adder is that the full-adder accepts an input carry.

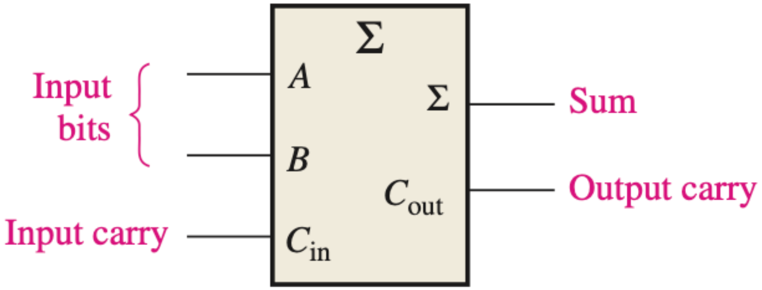


FIGURE 6–3 Logic symbol for a full-adder. Open file F06-03 to verify operation.

TABLE 6–2

Full-adder truth table.

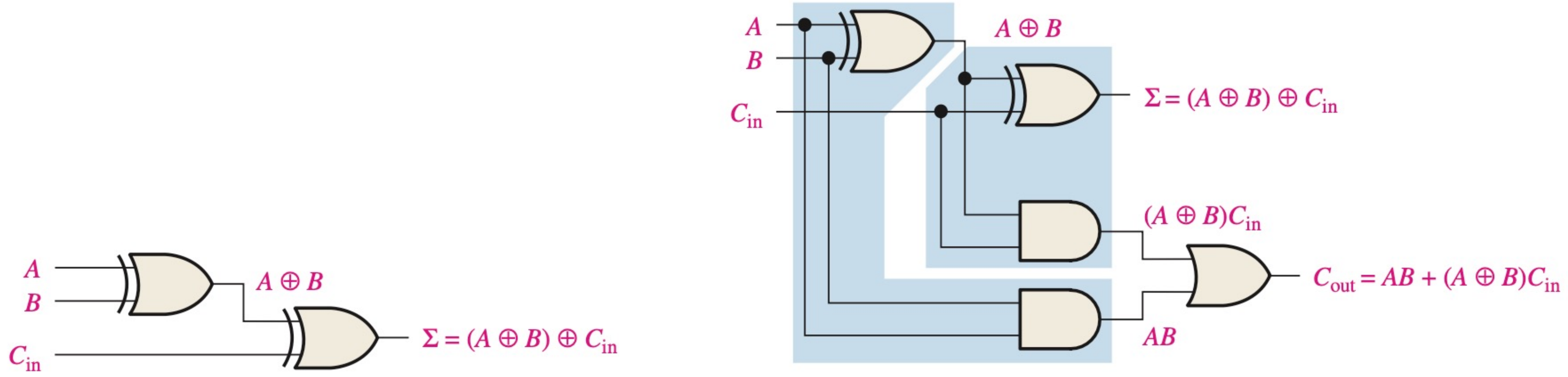
A	B	C_{in}	C_{out}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

C_{in} = input carry, sometimes designated as CI
C_{out} = output carry, sometimes designated as CO
Σ = sum
A and B = input variables (operands)

Full-Adder Logic

The full-adder must add the two input bits and the input carry. From the half-adder you know that the sum of the input bits A and B is the exclusive-OR of those two variables, $A \oplus B$. For the input carry (C_{in}) to be added to the input bits, it must be exclusive-ORed with $A \oplus B$, yielding the equation for the sum output of the full-adder.

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

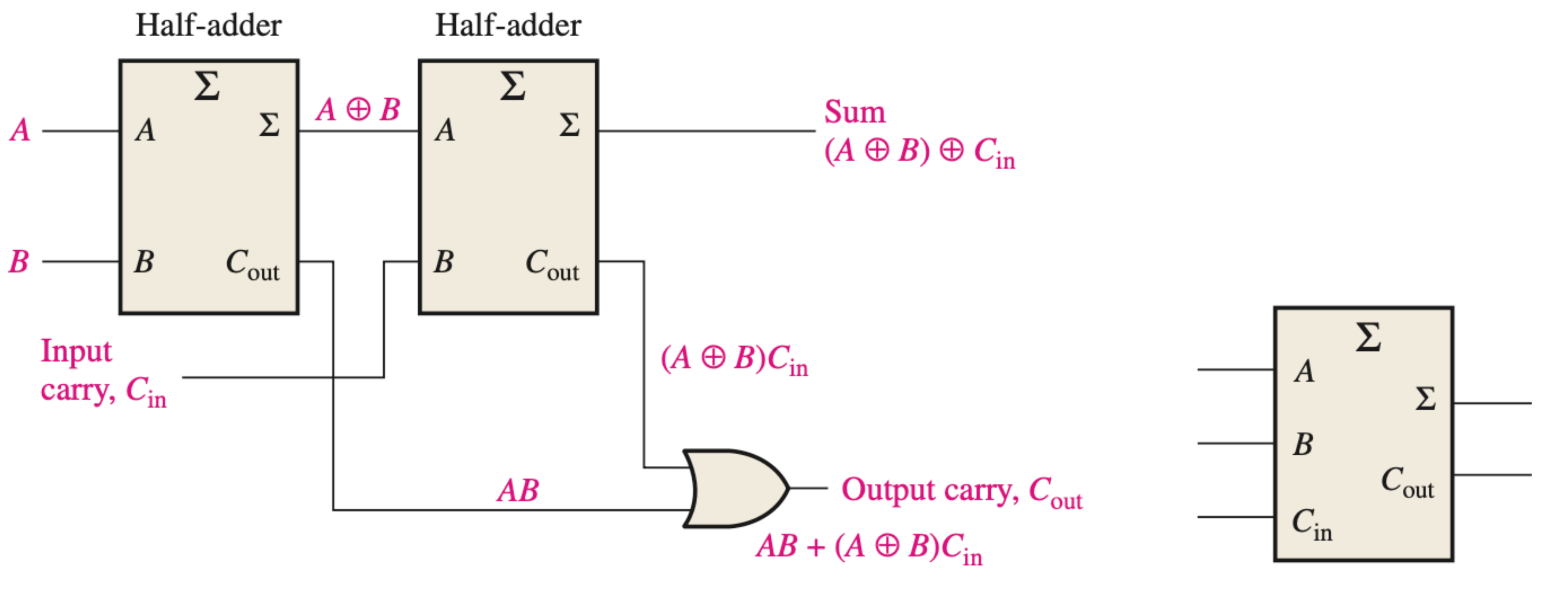


(a) Logic required to form the sum of three bits

(b) Complete logic circuit for a full-adder (each half-adder is enclosed by a shaded area)

FIGURE 6–4 Full-adder logic.

Full-Adder Logic



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

(b) Full-adder logic symbol

FIGURE 6–5 Full-adder implemented with half-adders.

EXAMPLE 6–1

For each of the three full-adders in Figure 6–6, determine the outputs for the inputs shown.

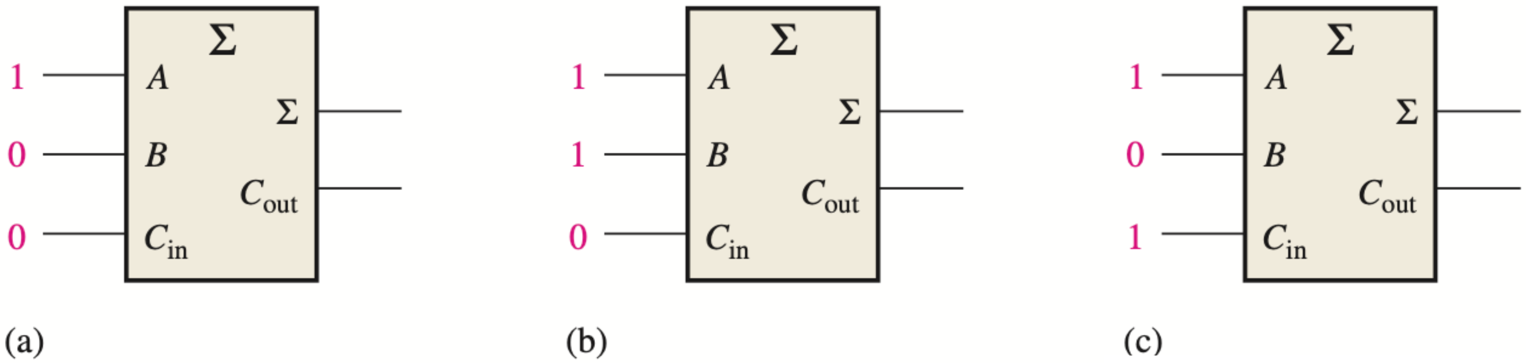


FIGURE 6–6

Solution

(a) The input bits are $A = 1$, $B = 0$, and $C_{in} = 0$.

$$1 + 0 + 0 = 1 \text{ with no carry}$$

Therefore, $\Sigma = \mathbf{1}$ and $C_{out} = \mathbf{0}$.

(b) The input bits are $A = 1$, $B = 1$, and $C_{in} = 0$.

$$1 + 1 + 0 = 0 \text{ with a carry of } 1$$

Therefore, $\Sigma = \mathbf{0}$ and $C_{out} = \mathbf{1}$.

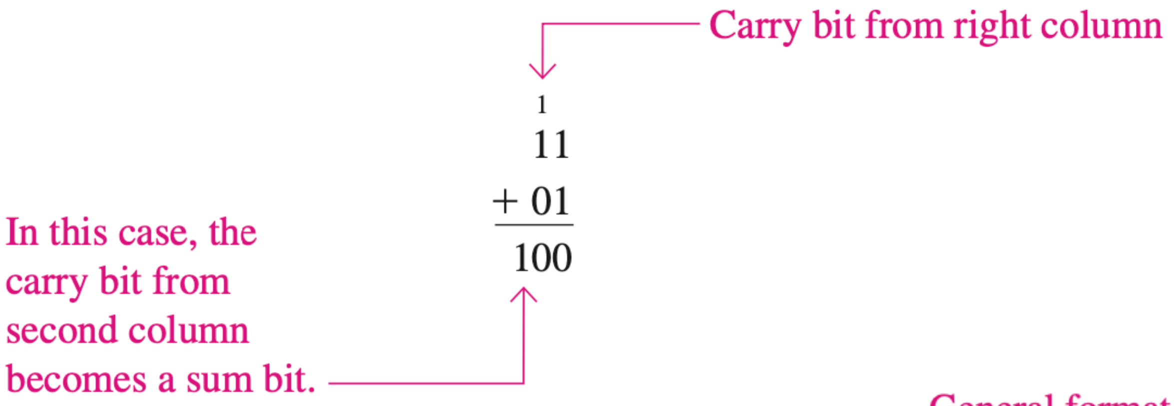
(c) The input bits are $A = 1$, $B = 0$, and $C_{in} = 1$.

$$1 + 0 + 1 = 0 \text{ with a carry of } 1$$

Therefore, $\Sigma = \mathbf{0}$ and $C_{out} = \mathbf{1}$.

2. Parallel Binary Adders

- When one binary number is added to another, each column generates a sum bit and a 1 or 0 carry bit to the next column to the left. as illustrated here with 2-bit numbers.



- To add two binary numbers, a full-adder (FA) is required for each bit in the numbers.

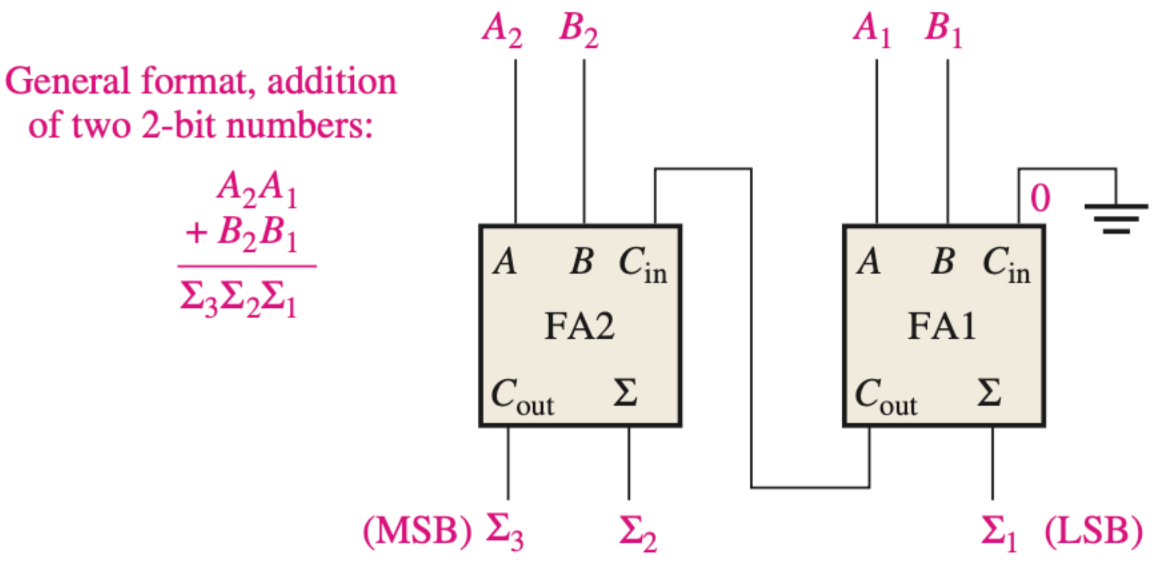


FIGURE 6–7 Block diagram of a basic 2-bit parallel adder using two full-adders.

EXAMPLE 6–2

Determine the sum generated by the 3-bit parallel adder in Figure 6–8 and show the intermediate carries when the binary numbers 101 and 011 are being added.

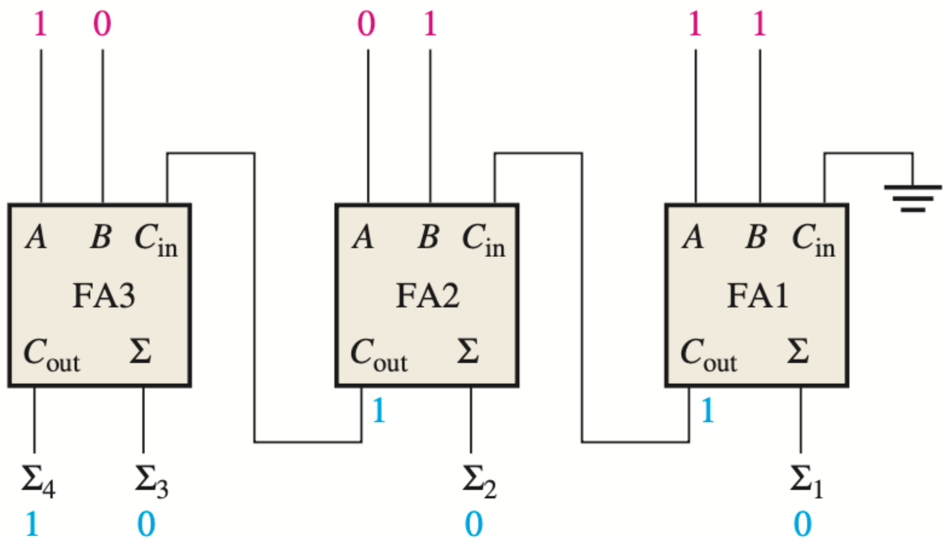
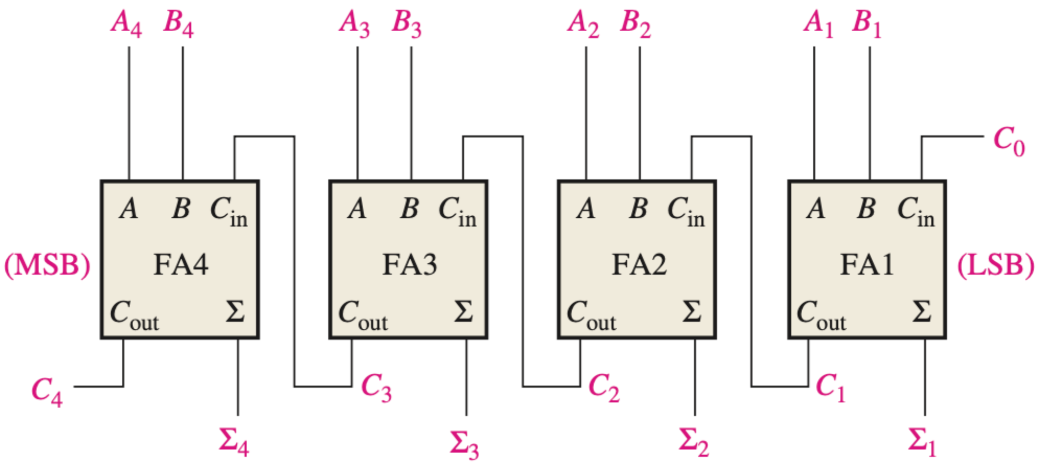


FIGURE 6–8

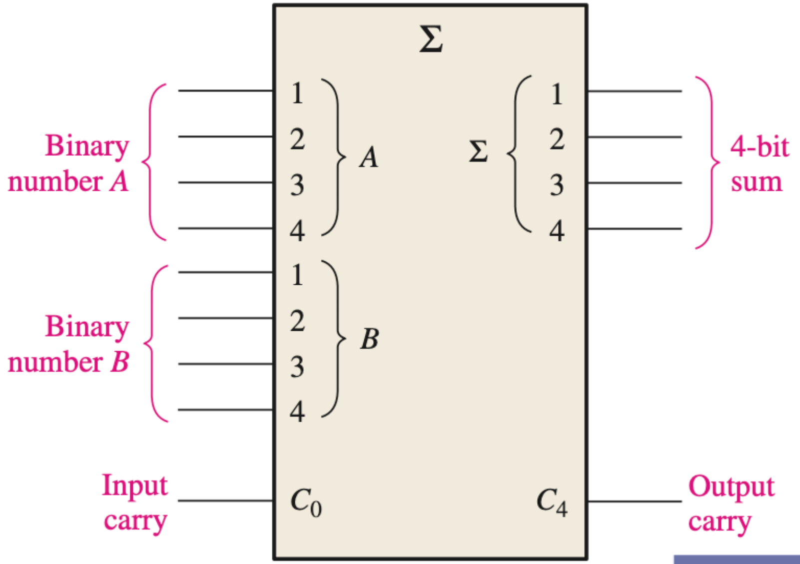
Solution

The LSBs of the two numbers are added in the right-most full-adder. The sum bits and the intermediate carries are indicated in blue in Figure 6–8.

Four-Bit Parallel Adders



(a) Block diagram



(b) Logic symbol

FIGURE 6–9 A 4-bit parallel adder.

Truth Table for a 4-Bit Parallel Adder

TABLE 6–3

Truth table for each stage of a 4-bit parallel adder.

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

Adders can be expanded to handle more bits by cascading.

EXAMPLE 6–3

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:

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

Solution

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,

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

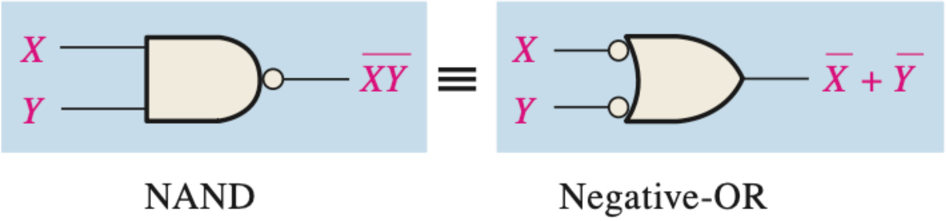
C_4 becomes the output carry; the sum of 1100 and 1100 is 11000.

3. Ripple Carry and Look-Ahead Carry Adders

DeMorgan's first theorem is stated as follows:

- The complement of a product of variables is equal to the sum of the complements of the variables. Or
- The complement of two or more ANDed variables is equivalent to the OR of the complements of the individual variables.
- The formula for expressing this theorem for two variables is:

$$\overline{XY} = \overline{X} + \overline{Y}$$



Inputs		Output	
X	Y	$\overline{XY}$	$\overline{X} + \overline{Y}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

The Ripple Carry Adder

A **ripple carry** adder is one in which the carry output of each full-adder is connected to the carry input of the next higher-order stage (a stage is one full-adder). The sum and the output carry of any stage cannot be produced until the input carry occurs; this causes a time delay in the addition process

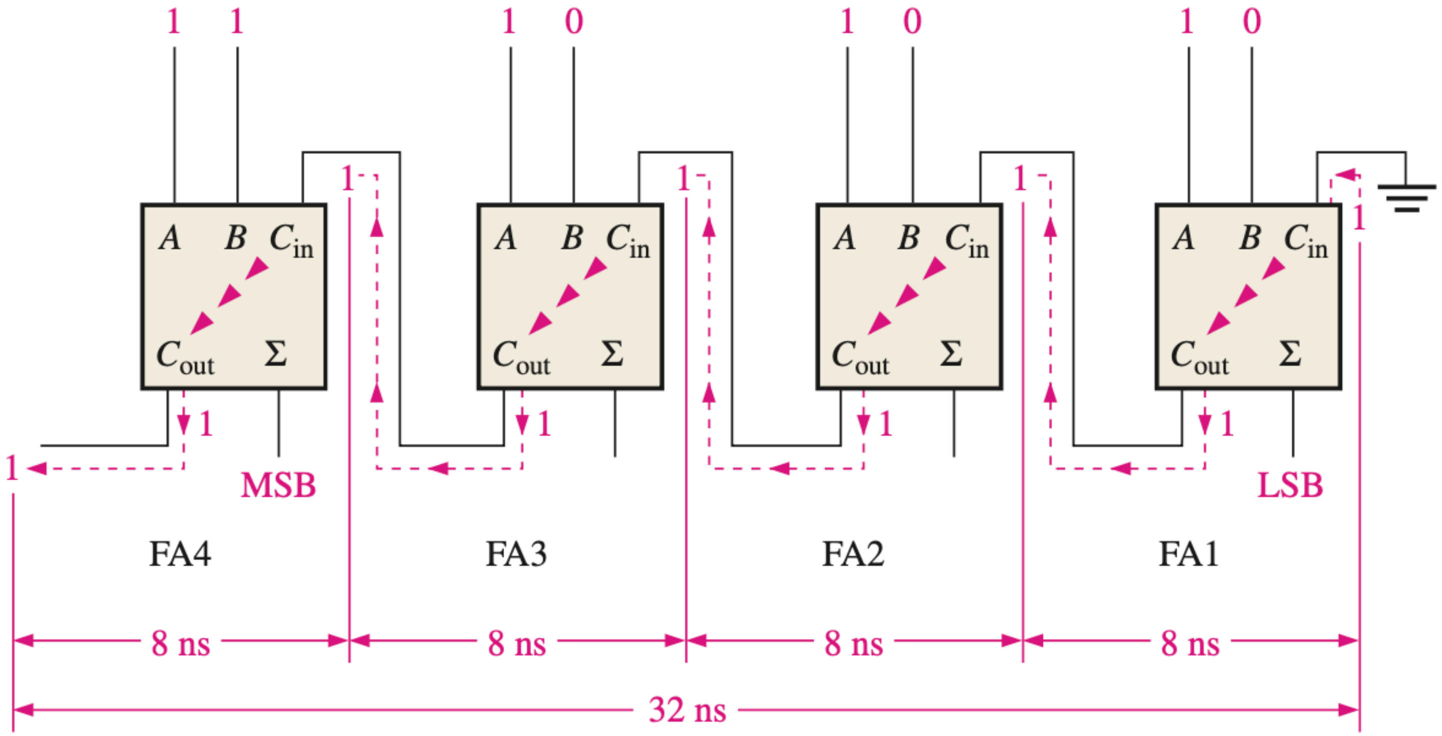


FIGURE 6–14 A 4-bit parallel ripple carry adder showing "worst-case" carry propagation delays.

The Look-Ahead Carry Adder

Carry generation occurs when an output carry is produced (generated) internally by the full-adder. A carry is generated only when both input bits are 1s. The generated carry, C_g, is expressed as the AND function of the two input bits, A and B.

$$C_g = AB$$

Carry propagation occurs when the input carry is rippled to become the output carry. An input carry may be propagated by the full-adder when either or both of the input bits are 1s. The propagated carry, C_p, is expressed as the OR function of the input bits.

$$C_p = A + B$$

The output carry of a full-adder can be expressed in terms of both the generated carry (C_g) and the propagated carry (C_p). The output carry (C_{out}) is a 1 if the generated carry is a 1 OR if the propagated carry is a 1 AND the input carry (C_{in}) is a 1.

$$C_{out} = C_g + C_p C_{in}$$

The Look-Ahead Carry Adder

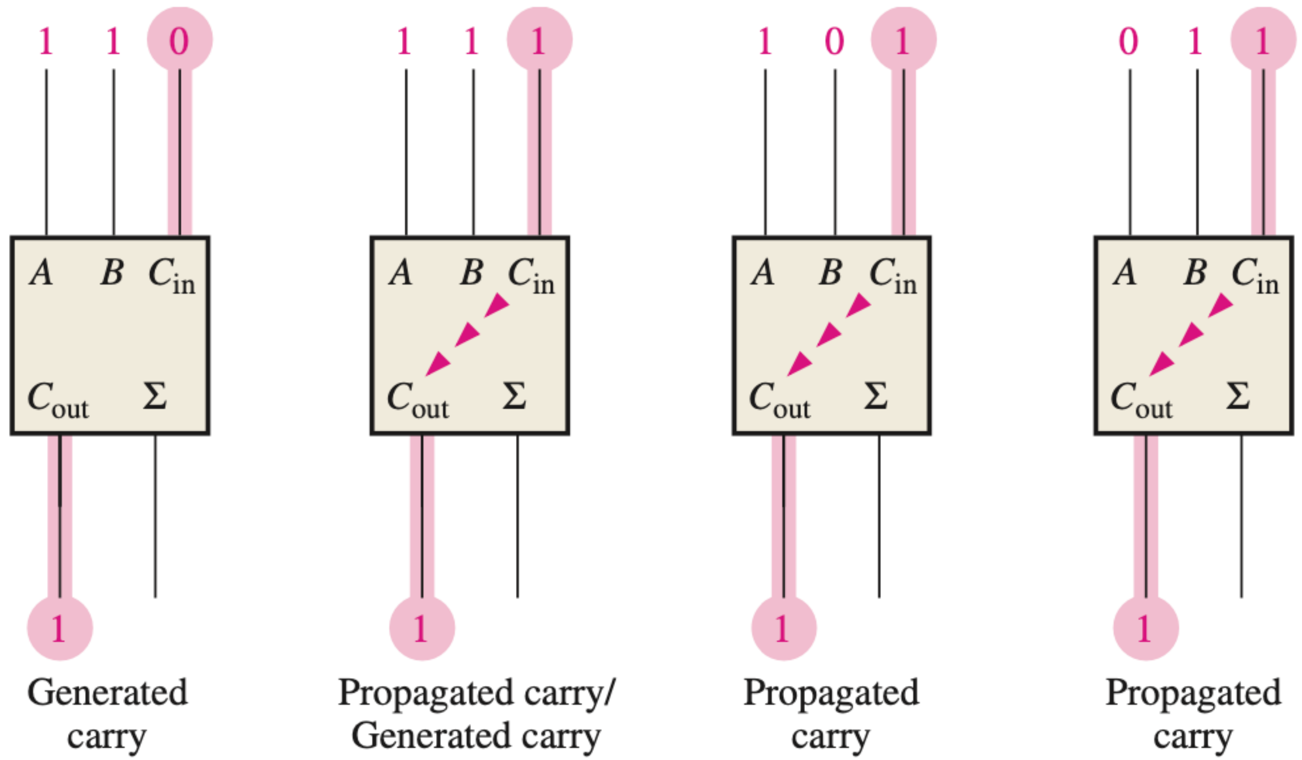


FIGURE 6–15 Illustration of conditions for carry generation and carry propagation.

The Look-Ahead Carry Adder

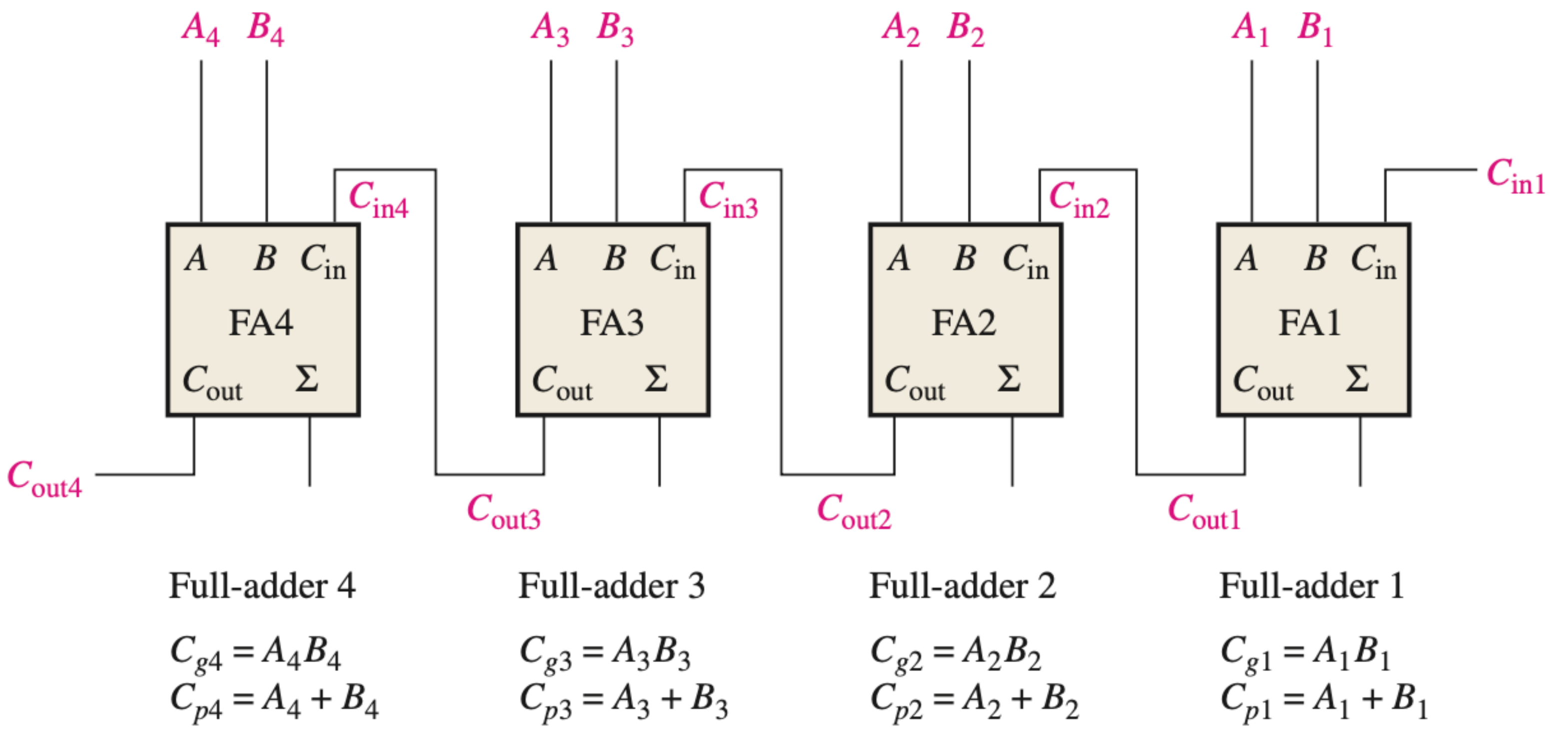


FIGURE 6–16 Carry generation and carry propagation in terms of the input bits to a 4-bit adder.

The Look-Ahead Carry Adder

Full-adder 1:

$$C_{\text{out1}} = C_{g1} + C_{p1}C_{\text{in1}}$$

Full-adder 2:

$$\begin{aligned} C_{\text{in2}} &= C_{\text{out1}} \\ C_{\text{out2}} &= C_{g2} + C_{p2}C_{\text{in2}} = C_{g2} + C_{p2}C_{\text{out1}} = C_{g2} + C_{p2}(C_{g1} + C_{p1}C_{\text{in1}}) \\ &= C_{g2} + C_{p2}C_{g1} + C_{p2}C_{p1}C_{\text{in1}} \end{aligned}$$

Full-adder 3:

$$\begin{aligned} C_{\text{in3}} &= C_{\text{out2}} \\ C_{\text{out3}} &= C_{g3} + C_{p3}C_{\text{in3}} = C_{g3} + C_{p3}C_{\text{out2}} = C_{g3} + C_{p3}(C_{g2} + C_{p2}C_{g1} + C_{p2}C_{p1}C_{\text{in1}}) \\ &= C_{g3} + C_{p3}C_{g2} + C_{p3}C_{p2}C_{g1} + C_{p3}C_{p2}C_{p1}C_{\text{in1}} \end{aligned}$$

Full-adder 4:

$$\begin{aligned} C_{\text{in4}} &= C_{\text{out3}} \\ C_{\text{out4}} &= C_{g4} + C_{p4}C_{\text{in4}} = C_{g4} + C_{p4}C_{\text{out3}} \\ &= C_{g4} + C_{p4}(C_{g3} + C_{p3}C_{g2} + C_{p3}C_{p2}C_{g1} + C_{p3}C_{p2}C_{p1}C_{\text{in1}}) \\ &= C_{g4} + C_{p4}C_{g3} + C_{p4}C_{p3}C_{g2} + C_{p4}C_{p3}C_{p2}C_{g1} + C_{p4}C_{p3}C_{p2}C_{p1}C_{\text{in1}} \end{aligned}$$

The Look-Ahead Carry Adder

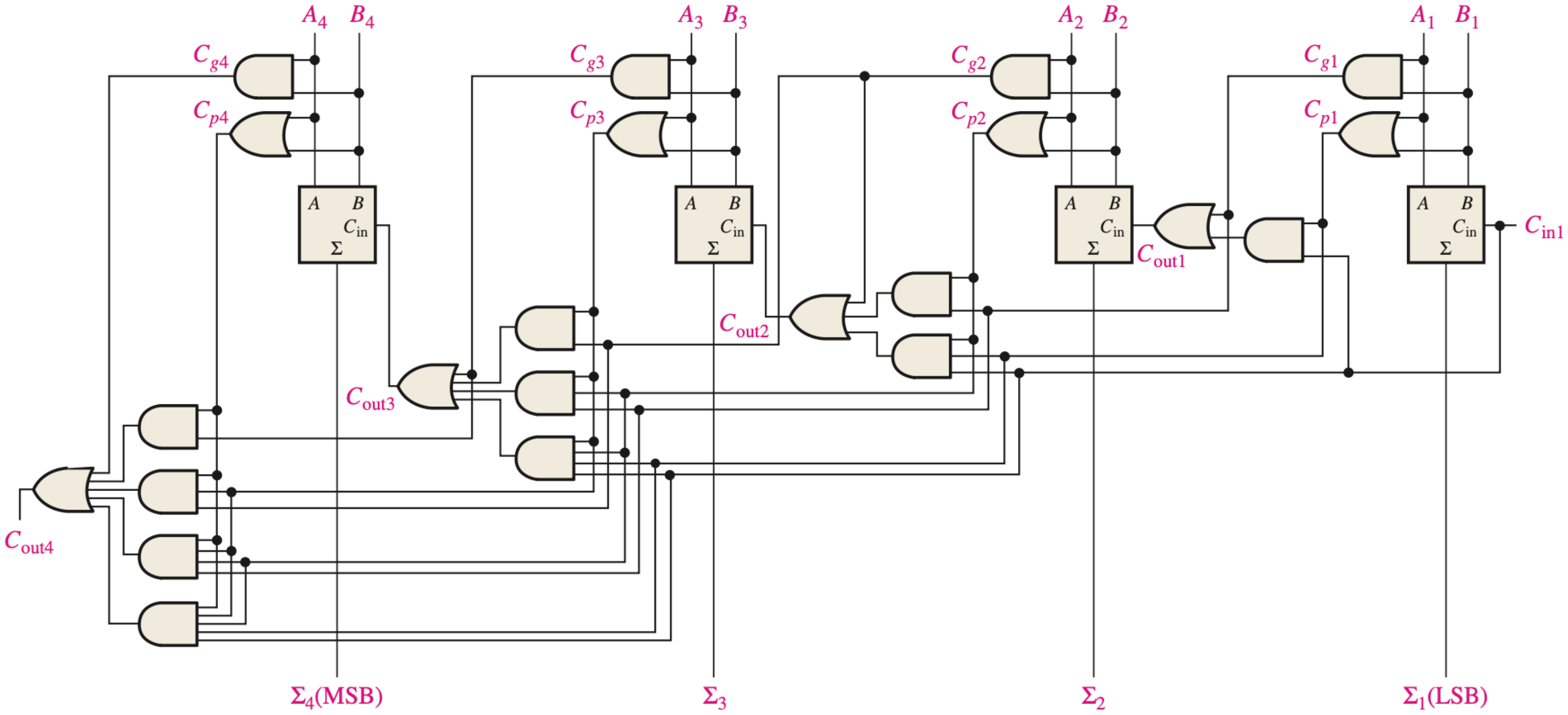


FIGURE 6–17 Logic diagram for a 4-stage look-ahead carry adder.

4. Comparators

Equality

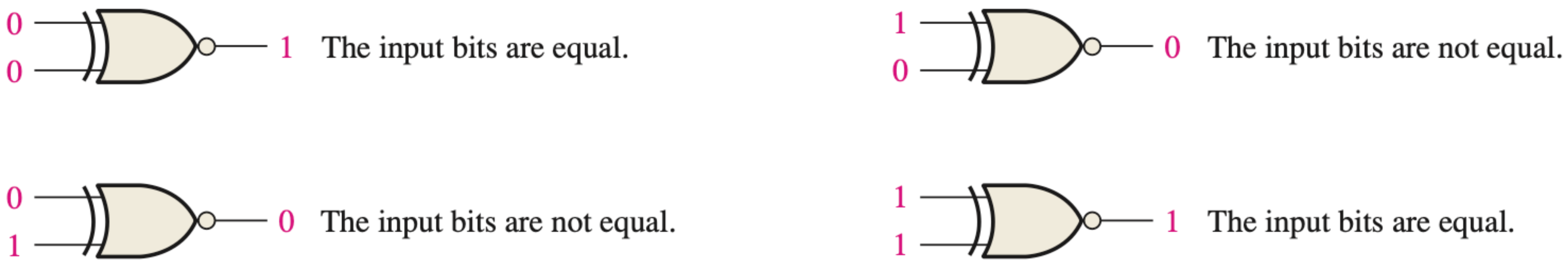


FIGURE 6–18 Basic comparator operation.

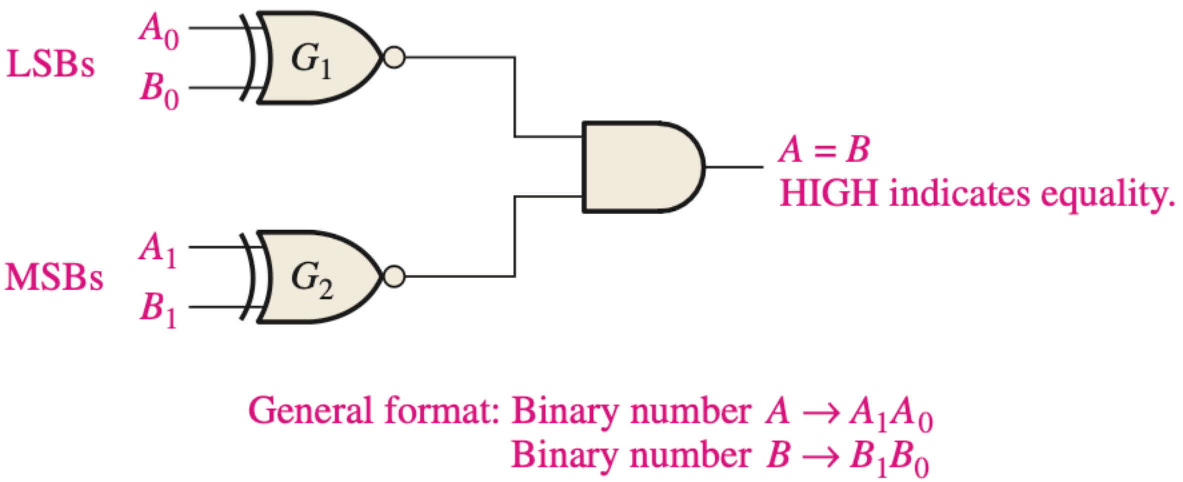


FIGURE 6–19 Logic diagram for equality comparison of two 2-bit numbers.

EXAMPLE 6–5

Apply each of the following sets of binary numbers to the comparator inputs in Figure 6–20, and determine the output by following the logic levels through the circuit.

(a) 10 and 10 **(b)** 11 and 10

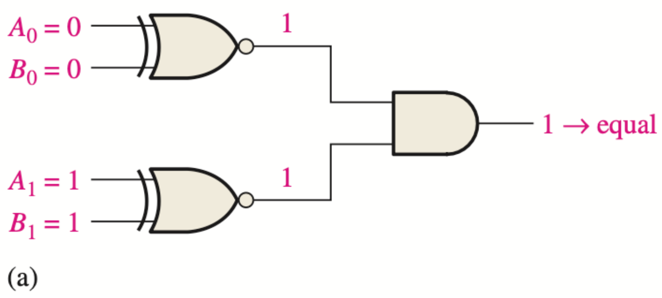


FIGURE 6–20

Inequality

To determine an inequality of binary numbers A and B, you first examine the highest- order bit in each number. The following conditions are possible:

- If $A_3 = 1$ and $B_3 = 0$, number A is greater than number B.
- If $A_3 = 0$ and $B_3 = 1$, number A is less than number B.
- If $A_3 = B3$, then you must examine the next lower bit position for an inequality.

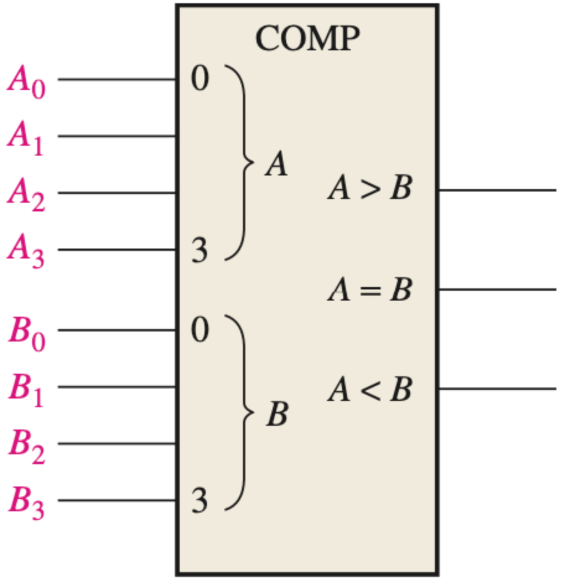


FIGURE 6–21 Logic symbol for a 4-bit comparator with inequality indication.

EXAMPLE 6–6

Determine the $A = B$, $A > B$, and $A < B$ outputs for the input numbers shown on the comparator in Figure 6–22.

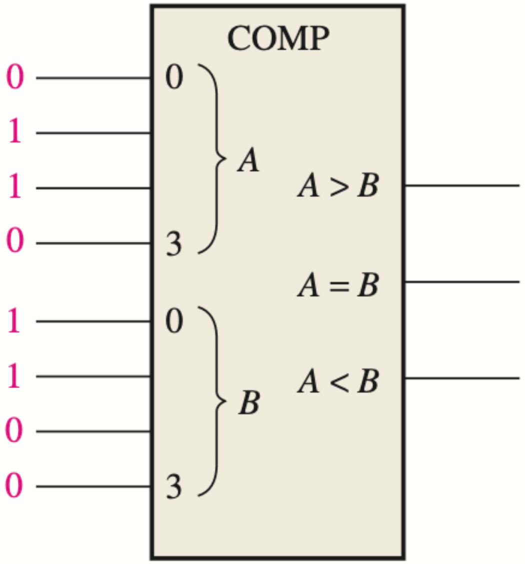


FIGURE 6–22

Solution

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**.

Related Problem

What are the comparator outputs when $A_3A_2A_1A_0 = 1001$ and $B_3B_2B_1B_0 = 1010$?

5. 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.

The Basic Binary Decoder

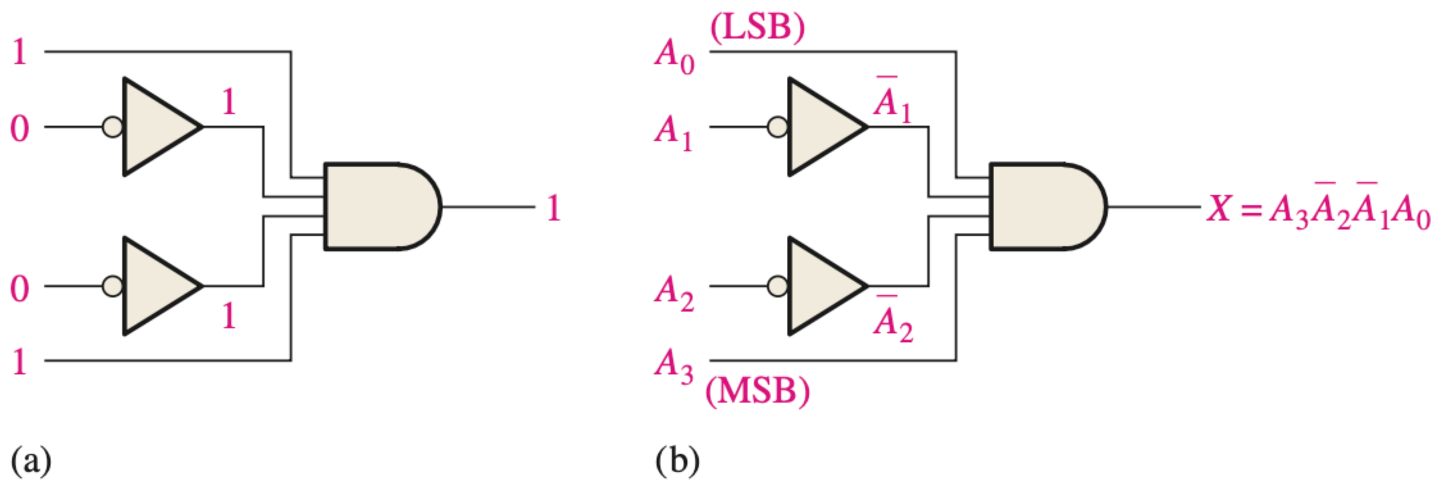


FIGURE 6–26 Decoding logic for the binary code 1001 with an active-HIGH output.

In the representation of a binary number or other weighted code in this book, the LSB is the right-most bit in a horizontal arrangement and the topmost bit in a vertical arrangement, unless specified otherwise.

EXAMPLE 6–8

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

Solution

The decoding function can be formed by complementing only the variables that appear as 0 in the desired binary number, as follows:

$$X = A_3\overline{A}_2A_1A_0 \qquad (1011)$$

This function can be implemented by connecting the true (uncomplemented) variables A_0, A_1, and A_3 directly to the inputs of an AND gate, and inverting the variable A_2 before applying it to the AND gate input. The decoding logic is shown in Figure 6–27.

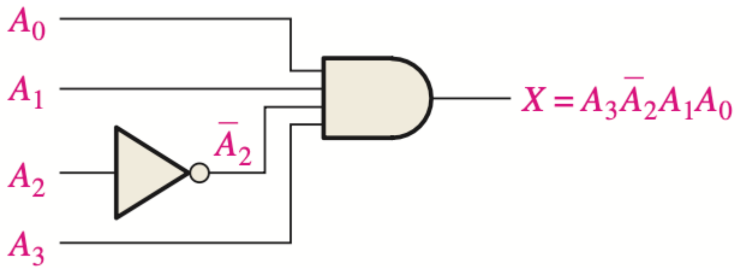


FIGURE 6–27 Decoding logic for producing a HIGH output when 1011 is on the inputs.

Related Problem

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-line-to-16-line decoder* because there are four inputs and sixteen outputs or a *1-of-16 decoder* because for any given code on the inputs, one of the sixteen outputs is activated.

TABLE 6–4

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

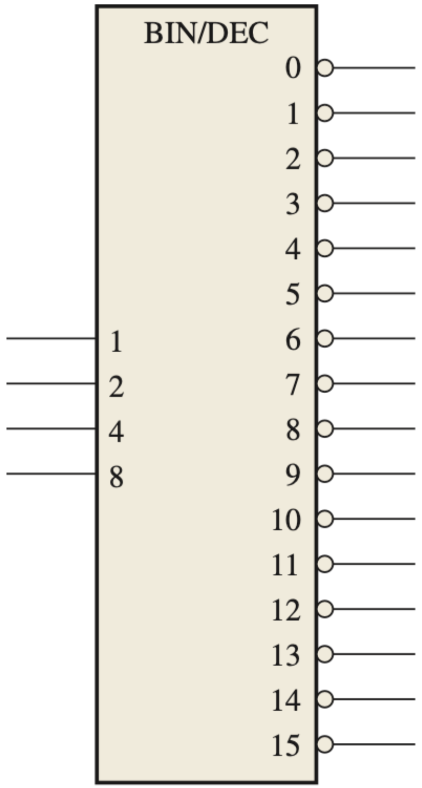


FIGURE 6–28 Logic symbol for a 4-line-to-16-line (1-of-16) decoder.

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.
- Each of these decoding functions is implemented with NAND gates to provide active-LOW outputs. If an active-HIGH output is required, AND gates are used for decoding.

TABLE 6–5

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	$A_3\overline{A}_2\overline{A}_1A_0$

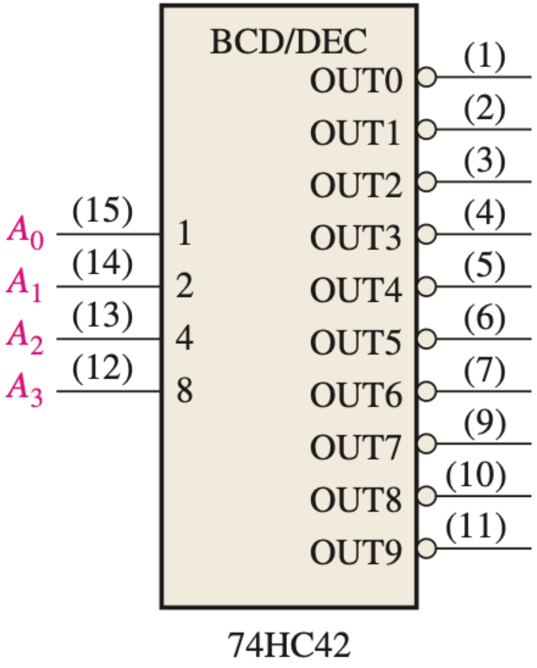


FIGURE 6–31 The 74HC42 BCD-to-decimal decoder.

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.

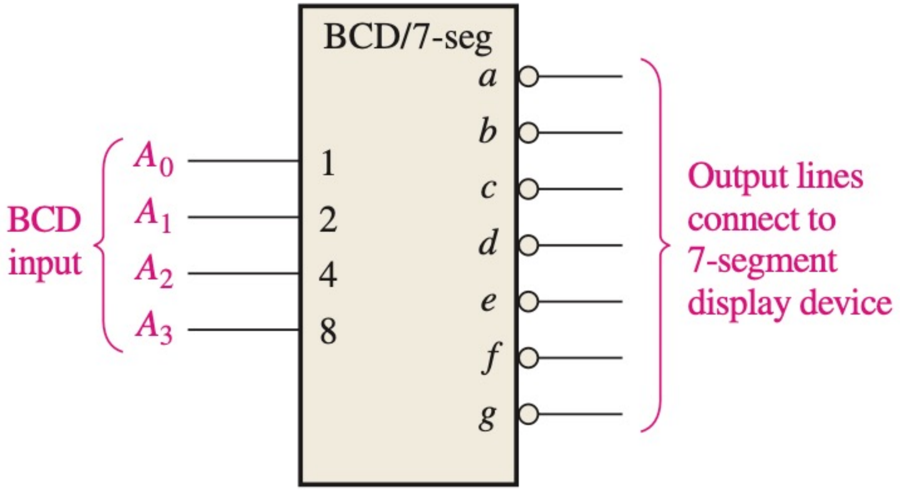


FIGURE 6–33 Logic symbol for a BCD-to-7-segment decoder/driver with active-LOW outputs. Open file F06-33 to verify operation.

6. Encoders

An encoder is a combinational logic circuit that essentially performs a “reverse” decoder function.

- An encoder accepts an active level on one of its inputs representing a digit, such as a decimal or octal digit, and converts it to a coded output, such as BCD or binary.

- Encoders can also be devised to encode various symbols and alphabetic characters.

The Decimal-to-BCD Encoder

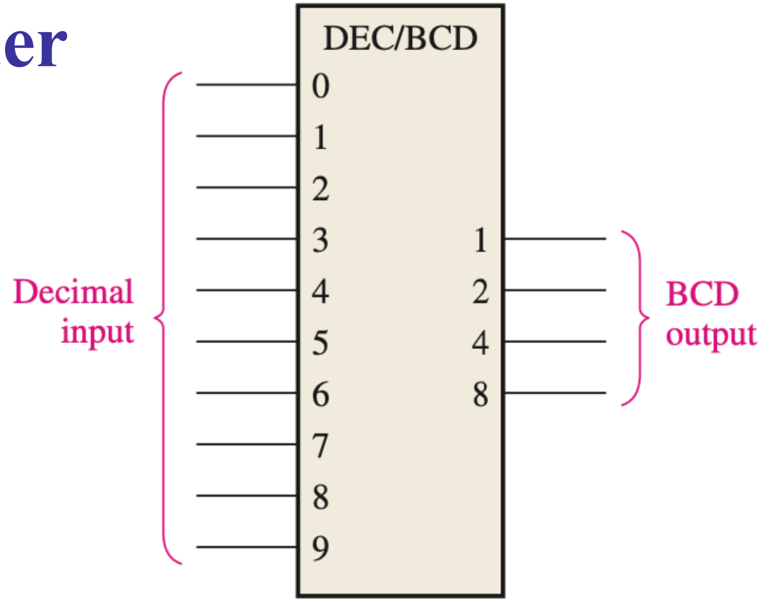


FIGURE 6–36 Logic symbol for a decimal-to-BCD encoder.

The Decimal-to-BCD Encoder

TABLE 6–6

Decimal Digit	BCD Code A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

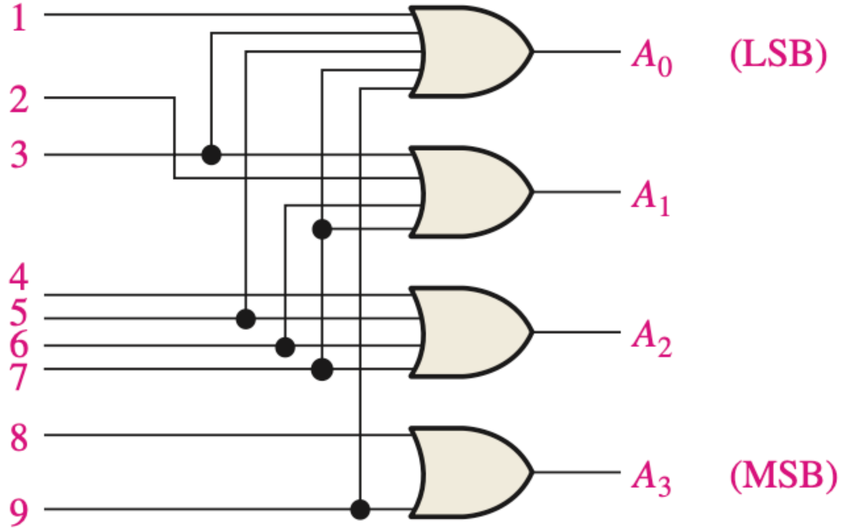


FIGURE 6–37 Basic logic diagram of a decimal-to-BCD encoder. A 0-digit input is not needed because the BCD outputs are all LOW when there are no HIGH inputs.

7. Code Converters

BCD-to-Binary Conversion

One method of BCD-to-binary code conversion uses adder circuits. The basic conversion process is as follows:

- The value, or weight, of each bit in the BCD number is represented by a binary number.
- All of the binary representations of the weights of bits that are 1s in the BCD number are added.
- The result of this addition is the binary equivalent of the BCD number.

The binary numbers representing the weights of the BCD bits are summed to produce the total binary number.

BCD-to-Binary Conversion

- The left-most 4-bit group represents 80, and the right-most 4-bit group represents 7.
- That is, the left-most group has a weight of 10, and the right-most group has a weight of 1.
- Within each group, the binary weight of each bit is as follows:

	Tens Digit				Units Digit			
Weight:	80	40	20	10	8	4	2	1
Bit designation:	B_3	B_2	B_1	B_0	A_3	A_2	A_1	A_0

TABLE 6–7

Binary representations of BCD bit weights.

		(MSB)		Binary Representation				(LSB)
BCD Bit	**BCD Weight**	**64**	**32**	**16**	**8**	**4**	**2**	**1**
A_0	1	0	0	0	0	0	0	1
A_1	2	0	0	0	0	0	1	0
A_2	4	0	0	0	0	1	0	0
A_3	8	0	0	0	1	0	0	0
B_0	10	0	0	0	1	0	1	0
B_1	20	0	0	1	0	1	0	0
B_2	40	0	1	0	1	0	0	0
B_3	80	1	0	1	0	0	0	0

EXAMPLE 6–12

Convert the BCD numbers 00100111 (decimal 27) and 10011000 (decimal 98) to binary.

Solution

Write the binary representations of the weights of all 1s appearing in the numbers, and then add them together.

80	40	20	10	8	4	2	1
0	0	1	0	0	1	1	1

0000001	1
0000010	2
0000100	4
+ 0010100	20
0011011	Binary number for decimal 27

80	40	20	10	8	4	2	1
1	0	0	1	1	0	0	0

0001000	8
0001010	10
+ 1010000	80
1100010	Binary number for decimal 98

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

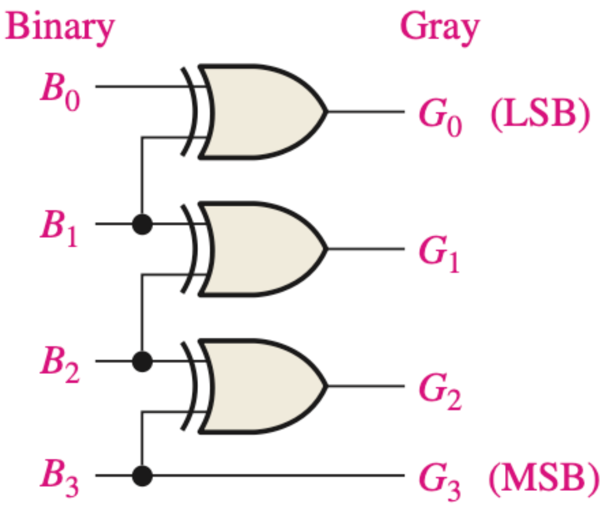


FIGURE 6–40 Four-bit binary-to-Gray conversion logic. Open file F06-40 to verify operation.

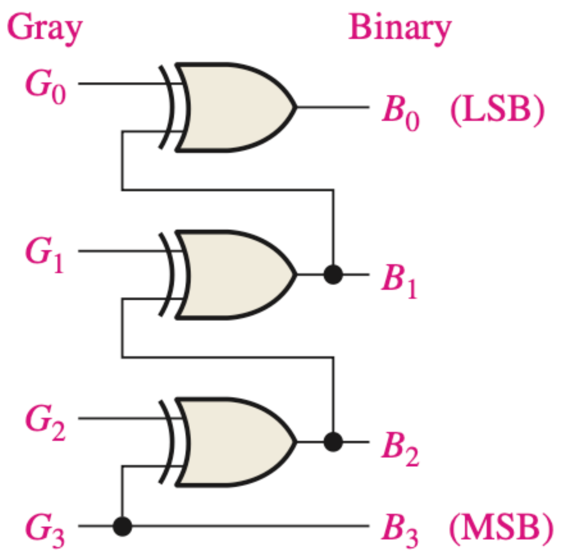


FIGURE 6–41 Four-bit Gray-to-binary conversion logic. Open file F06-41 to verify operation.

EXAMPLE 6–13

(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.

Solution

(a) 0101_2 is 0111 Gray. See Figure 6–42(a).

(b) 1011 Gray is 1101_2. See Figure 6–42(b).

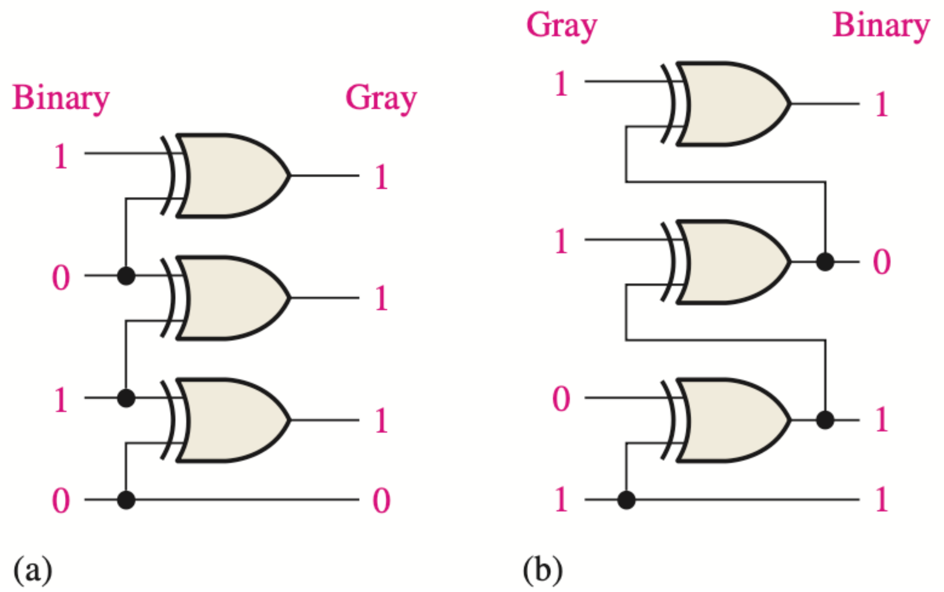


FIGURE 6–42

8. 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..

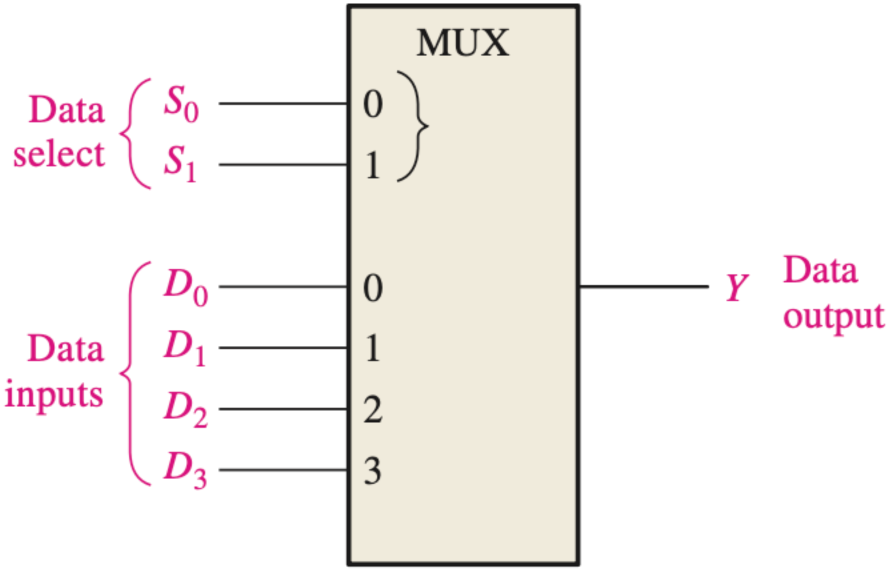


FIGURE 6–43 Logic symbol for a 1-of-4 data selector/multiplexer.

TABLE 6–8

Data selection for a 1-of-4-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

- Now let’s look at the logic circuitry required to perform this multiplexing operation. The data output is equal to the state of the selected data input.

The data output is equal to D_0 only if $S_1 = 0$ and $S_0 = 0$: $Y = D_0\overline{S}_1\overline{S}_0$.

The data output is equal to D_1 only if $S_1 = 0$ and $S_0 = 1$: $Y = D_1\overline{S}_1S_0$.

The data output is equal to D_2 only if $S_1 = 1$ and $S_0 = 0$: $Y = D_2S_1\overline{S}_0$.

The data output is equal to D_3 only if $S_1 = 1$ and $S_0 = 1$: $Y = D_3S_1S_0$.

When these terms are ORed, the total expression for the data output is

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

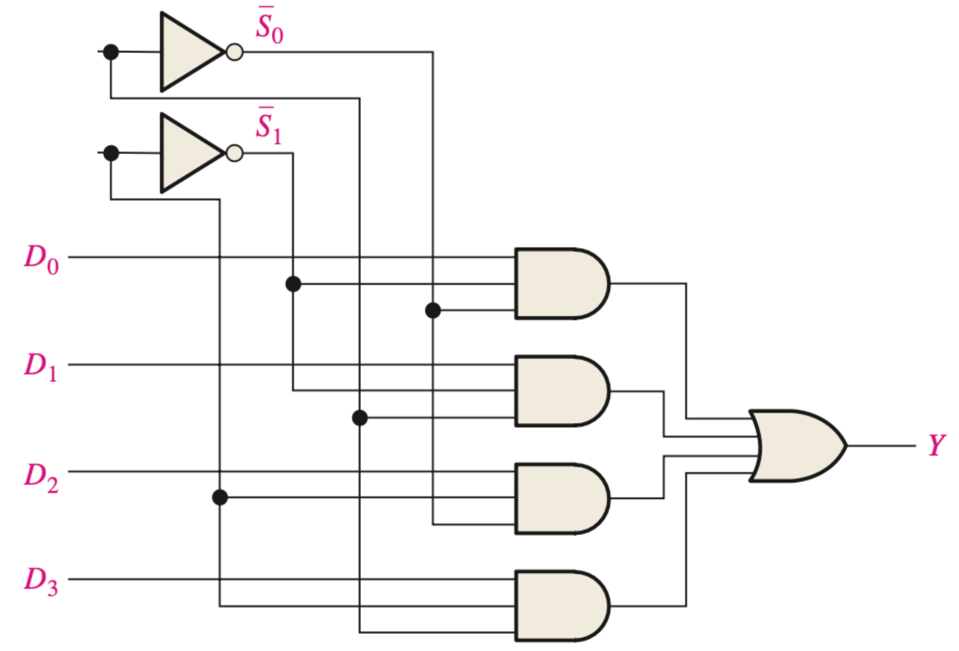
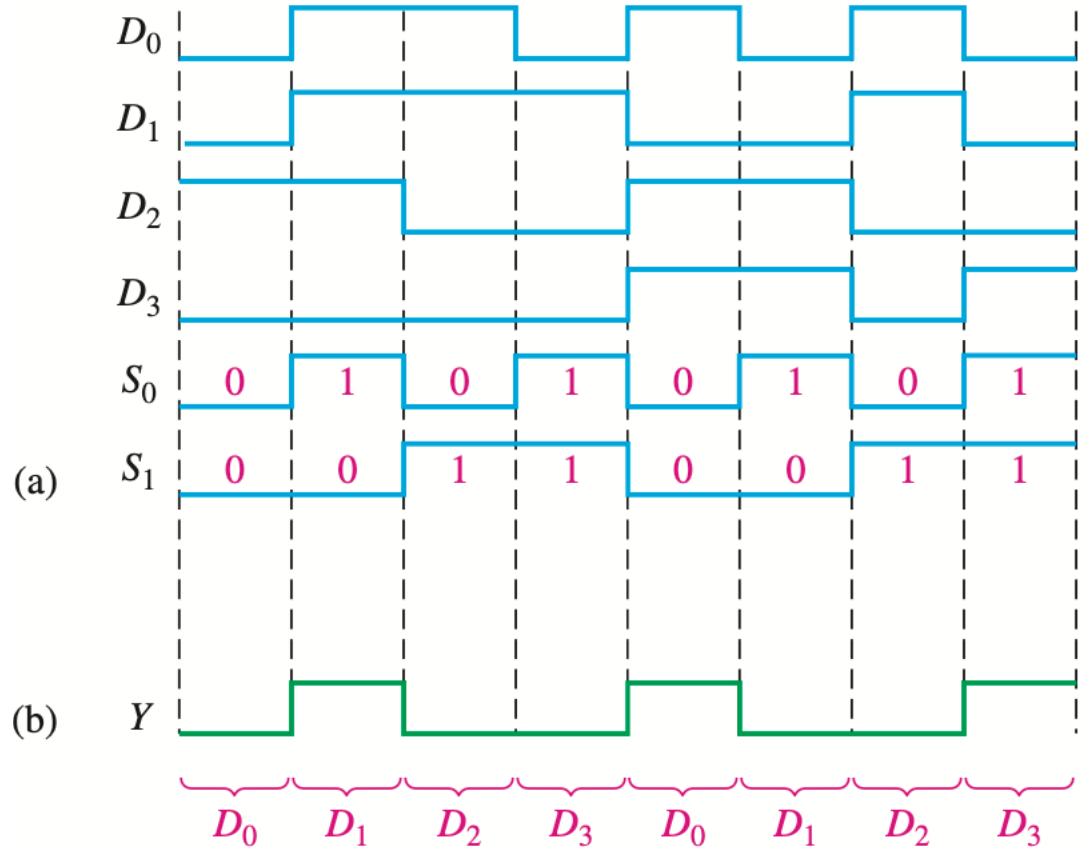


FIGURE 6–44 Logic diagram for a 4-input multiplexer. Open file F06-44 to verify operation.

EXAMPLE 6–14

The data-input and data-select waveforms in Figure 6–45(a) are applied to the multiplexer in Figure 6–44. Determine the output waveform in relation to the inputs.

FIGURE 6–45



Solution

The binary state of the data-select inputs during each interval determines which data input is selected. Notice that the data-select inputs go through a repetitive binary sequence 00, 01, 10, 11, 00, 01, 10, 11, and so on. The resulting output waveform is shown in Figure 6–45(b).

9. 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.
- For this reason, the demultiplexer is also known as a data distributor. .

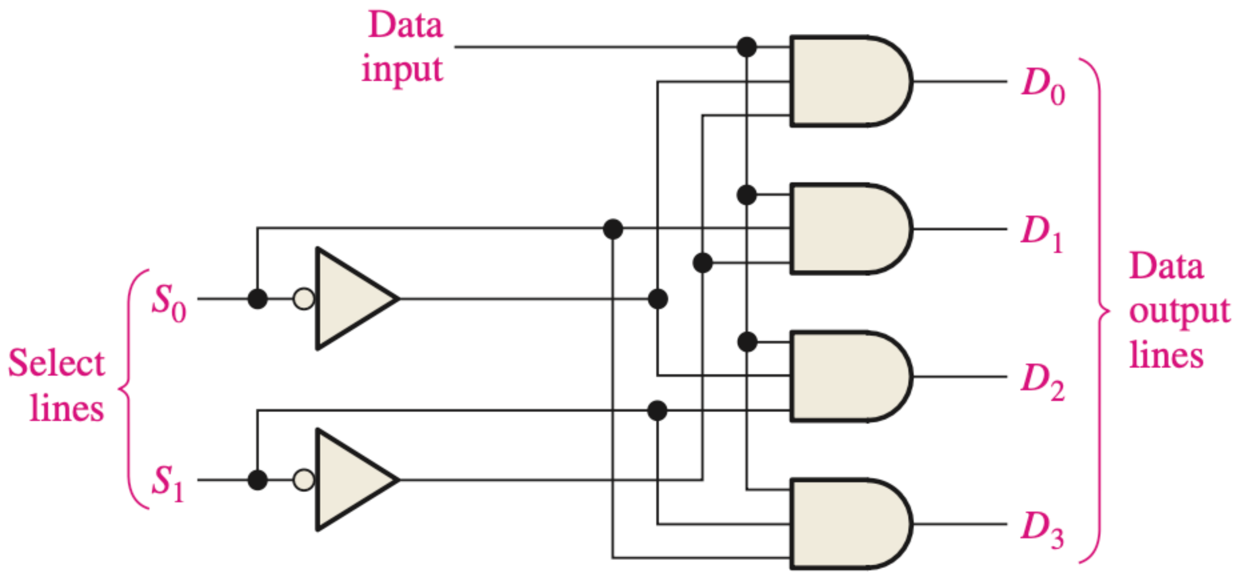


FIGURE 6–52 A 1-line-to-4-line demultiplexer.

EXAMPLE 6–18

The serial data-input waveform (Data in) and data-select inputs (S_0 and S_1) are shown in Figure 6–53. Determine the data-output waveforms on D_0 through D_3 for the demultiplexer in Figure 6–52.

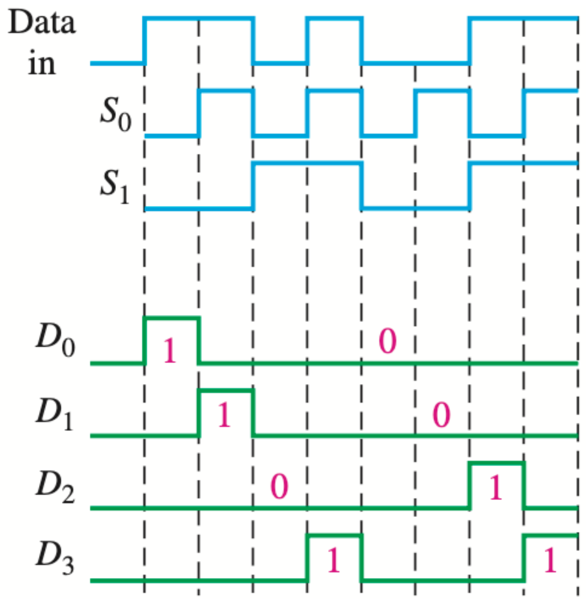


FIGURE 6–53

Solution

Notice that the select lines go through a binary sequence so that each successive input bit is routed to D_0, D_1, D_2, and D_3 in sequence, as shown by the output waveforms in Figure 6–53.

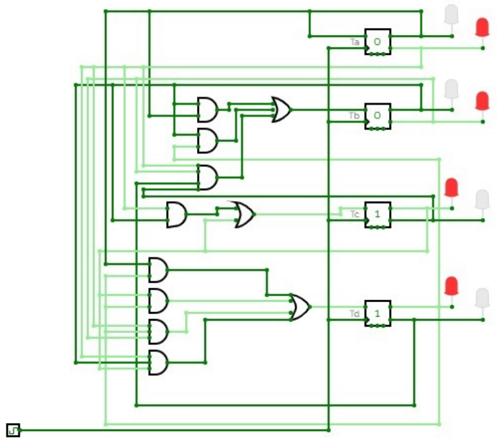
Vietnam National University HCMC

International University

School of
Electrical Engineering

THE END

Lecture 6: Functions of Combinational Logic



INSTRUCTOR: Dr. Vuong Quoc Bao