Digital Logic Design
Week No. 02

Muhammad Nouman Durrani

DECIMAL NUMBER SYSTEM

The decimal number system has ten digits.

The decimal number system has a base of 10.

EXAMPLE 2–1

Express the decimal number 47 as a sum of the values of each digit.

Solution The digit 4 has a weight of 10, which is 10^1, as indicated by its position. The digit 7 has a weight of 1, which is 10^0, as indicated by its position.

$$47 = (4 \times 10^1) + (7 \times 10^0)$$
$$= (4 \times 10) + (7 \times 1) = \mathbf{40 + 7}$$

Express the decimal number 568.23 as a sum of the values of each digit.

Solution The whole number digit 5 has a weight of 100, which is 10^2, the digit 6 has a weight of 10, which is 10^1, the digit 8 has a weight of 1, which is 10^0, the fractional digit 2 has a weight of 0.1, which is 10^{-1}, and the fractional digit 3 has a weight of 0.01, which is 10^{-2}.

$$\begin{aligned}568.23 &= (5 \times 10^2) + (6 \times 10^1) + (8 \times 10^0) + (2 \times 10^{-1}) + (3 \times 10^{-2}) \\ &= (5 \times 100) + (6 \times 10) + (8 \times 1) + (2 \times 0.1) + (3 \times 0.01) \\ &= \mathbf{500} + \mathbf{60} + \mathbf{8} + \mathbf{0.2} + \mathbf{0.03}\end{aligned}$$

BINARY NUMBER SYSTEM

The binary number system has two digits (bits).

The binary number system has a base of 2.

- With n bits we can count up to a number equal to 2^n - I
  Largest decimal number = 2^n - I

- For example, with five bits (n = 5) we can count from zero to thirty-one $2^5 - 1 = 32 - 1 = 31$

The weight structure of a binary number is

$$2^{n-1} . \; . \; . \; 2^3\, 2^2\, 2^1\, 2^0 \,.\, 2^{-1}\, 2^{-2} \; . \; . \; . \; 2^{-n}$$

Binary point

Convert the binary whole number 1101101 to decimal.

Solution Determine the weight of each bit that is a 1, and then find the sum of the weights to get the decimal number.

Weight:	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Binary number:	1	1	0	1	1	0	1

$$\begin{aligned} 1101101 &= 2^6 + 2^5 + 2^3 + 2^2 + 2^0 \\ &= 64 + 32 + 8 + 4 + 1 = \mathbf{109} \end{aligned}$$

Convert the fractional binary number 0.1011 to decimal.

Solution Determine the weight of each bit that is a 1, and then sum the weights to get the decimal fraction.

Weight:	2^{-1}	2^{-2}	2^{-3}	2^{-4}
Binary number:	0 . 1	0	1	1

$$\begin{aligned} 0.1011 &= 2^{-1} + 2^{-3} + 2^{-4} \\ &= 0.5 + 0.125 + 0.0625 = \mathbf{0.6875} \end{aligned}$$

Solution **(a)**

	Remainder
$\frac{19}{2} = 9$	1
$\frac{9}{2} = 4$	1
$\frac{4}{2} = 2$	0
$\frac{2}{2} = 1$	0
$\frac{1}{2} = 0$	1

1 0 0 1 1

MSB LSB

(b)

	Remainder
$\frac{45}{2} = 22$	1
$\frac{22}{2} = 11$	0
$\frac{11}{2} = 5$	1
$\frac{5}{2} = 2$	1
$\frac{2}{2} = 1$	0
$\frac{1}{2} = 0$	1

1 0 1 1 0 1

MSB LSB

MSB LSB

.0 1 0 1

	Carry
$0.3125 \times 2 = 0.\underbrace{625}$	0
$0.625 \times 2 = 1.\underbrace{25}$	1
$0.25 \times 2 = 0.\underbrace{50}$	0
$0.50 \times 2 = 1.\underbrace{00}$	1

Continue to the desired number of decimal places or stop when the fractional part is all zeros.

Binary Addition

The four basic rules for adding binary digits (bits) are as follows:

$0 + 0 = 0$	Sum of 0 with a carry of 0
$0 + 1 = 1$	Sum of 1 with a carry of 0
$1 + 0 = 1$	Sum of 1 with a carry of 0
$1 + 1 = 10$	Sum of 0 with a carry of 1

Add the following binary numbers:

(a) 11 + 11 **(b)** 100 + 10 **(c)** 111 + 11 **(d)** 110 + 100

Solution The equivalent decimal addition is also shown for reference.

(a) $\begin{array}{r} 11 \\ \underline{+11} \\ \mathbf{110} \end{array} \quad \begin{array}{r} 3 \\ \underline{+3} \\ 6 \end{array}$ **(b)** $\begin{array}{r} 100 \\ \underline{+10} \\ \mathbf{110} \end{array} \quad \begin{array}{r} 4 \\ \underline{+2} \\ 6 \end{array}$ **(c)** $\begin{array}{r} 111 \\ \underline{+11} \\ \mathbf{1010} \end{array} \quad \begin{array}{r} 7 \\ \underline{+3} \\ 10 \end{array}$ **(d)** $\begin{array}{r} 110 \\ \underline{+100} \\ \mathbf{1010} \end{array} \quad \begin{array}{r} 6 \\ \underline{+4} \\ 10 \end{array}$

Binary Subtraction

The four basic rules for subtracting bits are as follows:

$0 - 0 = 0$	
$1 - 1 = 0$	
$1 - 0 = 1$	
$10 - 1 = 1$	$0 - 1$ with a borrow of 1

Subtract 011 from 101.

Solution

$$\begin{array}{r} 101 \\ \underline{-011} \\ \mathbf{010} \end{array} \qquad \begin{array}{r} 5 \\ \underline{-3} \\ 2 \end{array}$$

Let's examine exactly what was done to subtract the two binary numbers since a borrow is required. Begin with the right column.

Left column:
When a 1 is borrowed,
a 0 is left, so $0 - 0 = 0$.

Middle column:
Borrow 1 from next column
to the left, making a 10 in
this column, then $10 - 1 = 1$.

$$\begin{array}{r} \overset{0}{\cancel{1}}{}^{1}01 \\ \underline{-0\,11} \\ 0\,10 \end{array}$$

Right column:
$1 - 1 = 0$

Binary Multiplication

The four basic rules for multiplying bits are as follows:

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

Perform the following binary multiplications:

(a) 11 × 11 **(b)** 101 × 111

Solution

(a)

$$\begin{array}{lr}
 & 11 \\
 & \underline{\times 11} \\
\text{Partial} & 11 \\
\text{products} & \underline{+11} \\
 & \mathbf{1001}
\end{array}
\qquad
\begin{array}{r}
3 \\
\underline{\times 3} \\
9
\end{array}$$

(b)

$$\begin{array}{lr}
 & 111 \\
 & \underline{\times 101} \\
 & 111 \\
\text{Partial products} & 000 \\
 & \underline{+111} \\
 & \mathbf{100011}
\end{array}
\qquad
\begin{array}{r}
7 \\
\underline{\times 5} \\
35
\end{array}$$

Perform the following binary divisions:

(a) 110 ÷ 11 **(b)** 110 ÷ 10

Solution

(a)

$$\begin{array}{r}
10 \\
11\overline{)110} \\
\underline{11} \\
000
\end{array}
\qquad
\begin{array}{r}
2 \\
3\overline{)6} \\
\underline{6} \\
0
\end{array}$$

(b)

$$\begin{array}{r}
11 \\
10\overline{)110} \\
\underline{10} \\
10 \\
\underline{10} \\
00
\end{array}
\qquad
\begin{array}{r}
3 \\
2\overline{)6} \\
\underline{6} \\
0
\end{array}$$

1'S AND 2'S COMPLEMENTS OF BINARY NUMBERS

Finding the 1's Complement

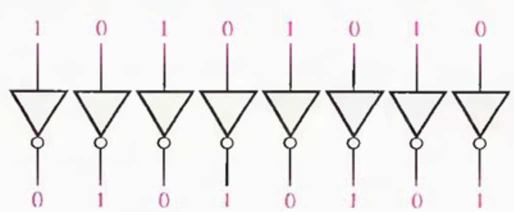
The 1's **complement** of a binary number is found by changing all 1s to 0s and all 0s to 1s, as illustrated below:

1 0 1 1 0 0 1 0	Binary number
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓	
0 1 0 0 1 1 0 1	1's complement

The simplest way to obtain the 1's complement of a binary number with a digital circuit is to use parallel inverters (NOT circuits), as shown in Figure 2–2 for an 8-bit binary number.

FIGURE 2–2

Example of inverters used to obtain the 1's complement of a binary number.



Finding the 2's Complement

Add 1 to the 1's complement to get the 2's complement.

The 2's complement of a binary number is found by adding 1 to the LSB of the 1's complement.

2's complement = (1's complement) + 1

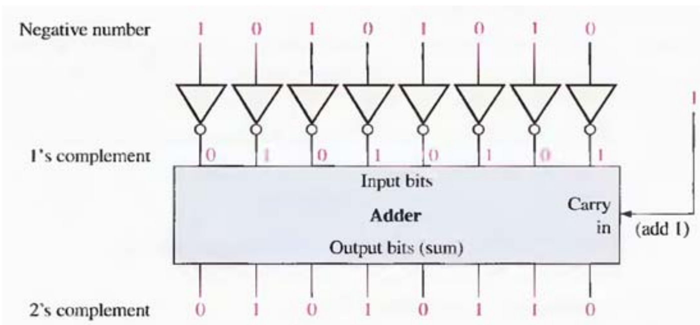
EXAMPLE 2–12

Find the 2's complement of 10110010.

Solution

10110010	Binary number
01001101	1's complement
+ 1	Add 1
01001110	2's complement

Related Problem Determine the 2's complement of 11001011.



SIGNED NUMBERS

The Sign Bit

The left-most bit in a signed binary number is the **sign bit,** which tells you whether the number is positive or negative.

A 0 sign bit indicates a positive number, and a 1 sign bit indicates a negative number.

Sign-Magnitude Form

When a signed binary number is represented in sign-magnitude, the left-most bit is the sign bit and the remaining bits are the magnitude bits.

00011001

Sign bit ↑ ↑ Magnitude bits

The decimal number −25 is expressed as

10011001

In the 1's complement form, a negative number is the 1's complement of the corresponding positive number.

In the 2's complement form, a negative number is the 2's complement of the corresponding positive number.

Express the decimal number −39 as an 8-bit number in the sign-magnitude, 1's complement, and 2's complement forms.

Solution First, write the 8-bit number for +39.

00100111

In the *sign-magnitude form*, −39 is produced by changing the sign bit to a 1 and leaving the magnitude bits as they are. The number is

10100111

In the *1's complement form*, −39 is produced by taking the 1's complement of +39 (00100111).

11011000

In the *2's complement form*, −39 is produced by taking the 2's complement of +39 (00100111) as follows:

11011000	1's complement
+ 1	
11011001	2's complement

Determine the decimal values of the signed binary numbers expressed in 1's complement:

(a) 00010111 **(b)** 11101000

Solution **(a)** The bits and their powers-of-two weights for the positive number are as follows:

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	1	0	1	1	1

Summing the weights where there are 1s,

$$16 + 4 + 2 + 1 = \mathbf{+23}$$

(b) The bits and their powers-of-two weights for the negative number are as follows. Notice that the negative sign bit has a weight of -2^7 or -128.

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	1	0	1	0	0	0

Summing the weights where there are 1s,

$$-128 + 64 + 32 + 8 = -24$$

Adding 1 to the result, the final decimal number is

$$-24 + 1 = \mathbf{-23}$$

Determine the decimal values of the signed binary numbers expressed in 2's complement:

(a) 01010110 **(b)** 10101010

Solution **(a)** The bits and their powers-of-two weights for the positive number are as follows:

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	1	0	1	1	0

Summing the weights where there are 1s,

$$64 + 16 + 4 + 2 = \mathbf{+86}$$

(b) The bits and their powers-of-two weights for the negative number are as follows. Notice that the negative sign bit has a weight of $-2^7 = -128$.

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	0	1	0	1	0

Summing the weights where there are 1s,

$$-128 + 32 + 8 + 2 = \mathbf{-86}$$

Floating-Point Numbers

A **floating-point number** (also known as a *real number*) consists of two parts plus a sign. The **mantissa** is the part of a floating-point number that represents the magnitude of the number. The **exponent** is the part of a floating-point number that represents the number of places that the decimal point (or binary point) is to be moved.

The floating-point number is written as : 0.2415068×10^9

For binary floating-point numbers, the format is defined by ANSI/IEEE Standard in three forms: single-precision, double-precision, and extended-precision.

Single-Precision Floating-Point Binary Numbers In the standard format for a single-precision binary number, the sign bit (S) is the left-most bit, the exponent (E) includes the next eight bits, and the mantissa or fractional part (F) includes the remaining 23 bits, as shown next.

32 bits

S	Exponent (E)	Mantissa (fraction, F)
1 bit	8 bits	23 bits

$$1011010010001 = 1.011010010001 \times 2^{12}$$

S	E	F
0	10001011	01101001000100000000000

Convert the decimal number 3.248×10^4 to a single-precision floating-point binary number.

Solution Convert the decimal number to binary.

$$3.248 \times 10^4 = 32480 = 111111011100000_2 = 1.11111011100000 \times 2^{14}$$

The MSB will not occupy a bit position because it is always a 1. Therefore, the mantissa is the fractional 23-bit binary number 11111011100000000000000 and the biased exponent is

$$14 + 127 = 141 = 10001101_2$$

The complete floating-point number is

0	**10001101**	**11111011100000000000000**

Add the signed numbers: 01000100, 00011011, 00001110, and 00010010.

Solution The equivalent decimal additions are given for reference.

68	01000100	
+ 27	+ 00011011	Add 1st two numbers
95	01011111	1st sum
+ 14	+ 00001110	Add 3rd number
109	01101101	2nd sum
+ 18	+ 00010010	Add 4th number
127	**01111111**	Final sum

Perform each of the following subtractions of the signed numbers:

(a) 00001000 − 00000011

(b) 00001100 − 11110111

(c) 11100111 − 00010011

(d) 10001000 − 11100010

Solution Like in other examples, the equivalent decimal subtractions are given for reference.

(a) In this case, $8 - 3 = 8 + (-3) = 5$.

	00001000	Minuend (+8)
	+ 11111101	2's complement of subtrahend (−3)
Discard carry →	1 **00000101**	Difference (+5)

(b) In this case, $12 - (-9) = 12 + 9 = 21$.

00001100	Minuend (+12)
+ 00001001	2's complement of subtrahend (+9)
00010101	Difference (+21)

(c) In this case, $-25 - (+19) = -25 + (-19) = -44$.

	11100111	Minuend (−25)
	+ 11101101	2's complement of subtrahend (−19)
Discard carry →	1 **11010100**	Difference (−44)

(d) In this case, $-120 - (-30) = -120 + 30 = -90$.

10001000	Minuend (−120)
+ 00011110	2's complement of subtrahend (+30)
10100110	Difference (−90)

HEXADECIMAL NUMBERS

- The hexadecimal number system has a base of sixteen; that is, it is composed of 16 numeric and alphabetic characters.

The hexadecimal number system consists of digits 0–9 and letters A–F.

DECIMAL	BINARY	HEXADECIMAL
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Convert the following binary numbers to hexadecimal:

(a) 1100101001010111 **(b)** 111111000101101001

Solution

(a) $\underbrace{1100}\underbrace{1010}\underbrace{0101}\underbrace{0111}$

$\downarrow\ \downarrow\ \downarrow\ \downarrow$

C A 5 7 $= \mathbf{CA57_{16}}$

(b) $\underbrace{0011}\underbrace{1111}\underbrace{0001}\underbrace{0110}\underbrace{1001}$

$\downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow$

3 F 1 6 9 $= \mathbf{3F169_{16}}$

Two zeros have been added in part (b) to complete a 4-bit group at the left.

Determine the binary numbers for the following hexadecimal numbers:

(a) $10A4_{16}$ **(b)** $CF8E_{16}$ **(c)** 9742_{16}

Solution

(a) 1 0 A 4

$\downarrow\ \downarrow\ \downarrow\ \downarrow$

1000010100100

(b) C F 8 E

$\downarrow\ \downarrow\ \downarrow\ \downarrow$

1100111110001110

(c) 9 7 4 2

$\downarrow\ \downarrow\ \downarrow\ \downarrow$

1001011101000010

In part (a), the MSB is understood to have three zeros preceding it, thus forming a 4-bit group.

Convert the following hexadecimal numbers to decimal:

(a) $1C_{16}$ **(b)** $A85_{16}$

Solution Remember, convert the hexadecimal number to binary first, then to decimal.

(a) 1 C

$\downarrow\ \downarrow$

$\overbrace{0001}\overbrace{1100} = 2^4 + 2^3 + 2^2 = 16 + 8 + 4 = \mathbf{28_{10}}$

(b) A 8 5

$\downarrow\ \downarrow\ \downarrow$

$\overbrace{1010}\overbrace{1000}\overbrace{0101} = 2^{11} + 2^9 + 2^7 + 2^2 + 2^0 = 2048 + 512 + 128 + 4 + 1 = \mathbf{2693_{10}}$

Convert the decimal number 650 to hexadecimal by repeated division by 16.

Solution

Hexadecimal remainder

$\frac{650}{16} = 40.625 \rightarrow 0.625 \times 16 = 10 =$ A

$\frac{40}{16} = 2.5 \longrightarrow 0.5 \times 16 = 8 =$ 8

$\frac{2}{16} = 0.125 \longrightarrow 0.125 \times 16 = 2 =$ 2

Stop when whole number quotient is zero.

2 8 A Hexadecimal number

MSD LSD

Add the following hexadecimal numbers:

(a) $23_{16} + 16_{16}$ **(b)** $58_{16} + 22_{16}$ **(c)** $2B_{16} + 84_{16}$ **(d)** $DF_{16} + AC_{16}$

Solution

(a) $\begin{array}{r} 23_{16} \\ +16_{16} \\ \hline \mathbf{39}_{16} \end{array}$ right column: $3_{16} + 6_{16} = 3_{10} + 6_{10} = 9_{10} = 9_{16}$; left column: $2_{16} + 1_{16} = 2_{10} + 1_{10} = 3_{10} = 3_{16}$

(b) $\begin{array}{r} 58_{16} \\ +22_{16} \\ \hline \mathbf{7A}_{16} \end{array}$ right column: $8_{16} + 2_{16} = 8_{10} + 2_{10} = 10_{10} = A_{16}$; left column: $5_{16} + 2_{16} = 5_{10} + 2_{10} = 7_{10} = 7_{16}$

(c) $\begin{array}{r} 2B_{16} \\ +84_{16} \\ \hline \mathbf{AF}_{16} \end{array}$ right column: $B_{16} + 4_{16} = 11_{10} + 4_{10} = 15_{10} = F_{16}$; left column: $2_{16} + 8_{16} = 2_{10} + 8_{10} = 10_{10} = A_{16}$

(d) $\begin{array}{r} DF_{16} \\ +AC_{16} \\ \hline \mathbf{18B}_{16} \end{array}$ right column: $F_{16} + C_{16} = 15_{10} + 12_{10} = 27_{10}$
$27_{10} - 16_{10} = 11_{10} = B_{16}$ with a 1 carry
left column: $D_{16} + A_{16} + 1_{16} = 13_{10} + 10_{10} + 1_{10} = 24_{10}$
$24_{10} - 16_{10} = 8_{10} = 8_{16}$ with a 1 carry

Subtract the following hexadecimal numbers:

(a) $84_{16} - 2A_{16}$ **(b)** $C3_{16} - 0B_{16}$

Solution

(a) $2A_{16} = 00101010$

2's complement of $2A_{16} = 11010110 = D6_{16}$ (using Method 1)

$$\begin{array}{r} 84_{16} \\ +D6_{16} \\ \hline \cancel{1}5A_{16} \end{array}$$

Add

Drop carry, as in 2's complement addition

The difference is $\mathbf{5A}_{16}$.

(b) $0B_{16} = 00001011$

2's complement of $0B_{16} = 11110101 = F5_{16}$ (using Method 1)

$$\begin{array}{r} C3_{16} \\ +F5_{16} \\ \hline \cancel{1}B8_{16} \end{array}$$

Add

Drop carry

The difference is $\mathbf{B8}_{16}$.

OCTAL NUMBERS

The **octal** number system is composed of eight digits, which are

0, 1, 2, 3, 4, 5, 6, 7

The octal number system has a base of 8.

Octal number: 2 3 7 4

$$\begin{aligned}2374_8 &= (2\times 8^3) + (3\times 8^2) + (7\times 8^1) + (4\times 8^0)\\ &= (2\times 512) + (3\times 64) + (7\times 8) + (4\times 1)\\ &= 1024 + 192 + 56 + 4 = 1276_{10}\end{aligned}$$

Remainder

$\frac{359}{8} = 44.875 \rightarrow 0.875 \times 8 = 7$

$\frac{44}{8} = 5.5 \rightarrow 0.5 \times 8 = 4$

$\frac{5}{8} = 0.625 \rightarrow 0.625 \times 8 = 5$

Stop when whole number quotient is zero

5 4 7 Octal number

MSD LSD

Convert each of the following octal numbers to binary:

(a) 13_8 **(b)** 25_8 **(c)** 140_8 **(d)** 7526_8

Solution

(a) 1 3 ↓ ↓ 001011

(b) 2 5 ↓ ↓ 010101

(c) 1 4 0 ↓ ↓ ↓ 001100000

(d) 7 5 2 6 ↓ ↓ ↓ ↓ 111101010110

Convert each of the following binary numbers to octal:

(a) 110101 **(b)** 101111001 **(c)** 100110011010 **(d)** 11010000100

Solution

(a) 110101 ↓ ↓ 6 5 = $\mathbf{65_8}$

(b) 101111001 ↓ ↓ ↓ 5 7 1 = $\mathbf{571_8}$

(c) 100110011010 ↓ ↓ ↓ ↓ 4 6 3 2 = $\mathbf{4632_8}$

(d) 011010000100 ↓ ↓ ↓ ↓ 3 2 0 4 = $\mathbf{3204_8}$

BCD

- Binary coded decimal means that each decimal digit, 0 through 9, is represented by a binary code of four bits.

DECIMAL DIGIT	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

◀ TABLE 2–5

Decimal/BCD conversion.

Convert each of the following decimal numbers to BCD:

(a) 35 (b) 98 (c) 170 (d) 2469

Solution

(a) $\overset{3}{\overbrace{0011}}\overset{5}{\overbrace{0101}}$ = 00110101

(b) $\overset{9}{\overbrace{1001}}\overset{8}{\overbrace{1000}}$ = 10011000

(c) $\overset{1}{\overbrace{0001}}\overset{7}{\overbrace{0111}}\overset{0}{\overbrace{0000}}$ = 000101110000

(d) $\overset{2}{\overbrace{0010}}\overset{4}{\overbrace{0100}}\overset{6}{\overbrace{0110}}\overset{9}{\overbrace{1001}}$ = 0010010001101001

Convert each of the following BCD codes to decimal:

(a) 10000110 (b) 001101010001 (c) 1001010001110000

Solution

(a) $\underbrace{1000}_{8}\underbrace{0110}_{6}$

(b) $\underbrace{0011}_{3}\underbrace{0101}_{5}\underbrace{0001}_{1}$

(c) $\underbrace{1001}_{9}\underbrace{0100}_{4}\underbrace{0111}_{7}\underbrace{0000}_{0}$

Add the following BCD numbers:

(a) 0011 + 0100 (b) 00100011 + 00010101

(c) 10000110 + 00010011 (d) 010001010000 + 010000010111

Solution The decimal number additions are shown for comparison.

(a)
$$\begin{array}{r} 0011 \\ +\,0100 \\ \hline \mathbf{0111} \end{array} \qquad \begin{array}{r} 3 \\ +\,4 \\ \hline 7 \end{array}$$

(b)
$$\begin{array}{rr} 0010 & 0011 \\ +\,0001 & 0101 \\ \hline \mathbf{0011} & \mathbf{1000} \end{array} \qquad \begin{array}{r} 23 \\ +\,15 \\ \hline 38 \end{array}$$

(c)
$$\begin{array}{rr} 1000 & 0110 \\ +\,0001 & 0011 \\ \hline \mathbf{1001} & \mathbf{1001} \end{array} \qquad \begin{array}{r} 86 \\ +\,13 \\ \hline 99 \end{array}$$

(d)
$$\begin{array}{rrr} 0100 & 0101 & 0000 \\ +\,0100 & 0001 & 0111 \\ \hline \mathbf{1000} & \mathbf{0110} & \mathbf{0111} \end{array} \qquad \begin{array}{r} 450 \\ +\,417 \\ \hline 867 \end{array}$$

Note that in each case the sum in any 4-bit column does not exceed 9, and the results are valid BCD numbers.

Add the following BCD numbers

(a) 1001 + 0100 (b) 1001 + 1001

(c) 00010110 + 00010101 (d) 01100111 + 01010011

Solution The decimal number additions are shown for comparison.

(a)		1001		9
		+ 0100		+4
		1101	Invalid BCD number (>9)	13
		+ 0110	Add 6	
	0001	**0011**	Valid BCD number	
	↓	↓		
	1	3		

(b)		1001		9
		+ 1001		+ 9
	1	0010	Invalid because of carry	18
		+ 0110	Add 6	
	0001	**1000**	Valid BCD number	
	↓	↓		
	1	8		

(c)	0001	0110		16
	+ 0001	0101		+ 15
	0010	1011	Right group is invalid (>9), left group is valid.	31
		+ 0110	Add 6 to invalid code. Add carry, 0001, to next group.	
	0011	**0001**	Valid BCD number	
	↓	↓		
	3	1		

(d)		0110	0111		67
		+ 0101	0011		+ 53
		1011	1010	Both groups are invalid (>9)	120
		+ 0110	+ 0110	Add 6 to both groups	
	0001	**0010**	**0000**	Valid BCD number	
	↓	↓	↓		
	1	2	0		

Gray Code

- It exhibits only a single bit change from one code word to the next in sequence.
- This property is important in many applications, such as shaft position encoders, where error susceptibility increases with the number of bit changes between adjacent numbers in a sequence.

Binary-to-Gray Code Conversion Conversion between binary code and Gray code is sometimes useful. The following rules explain how to convert from a binary number to a Gray code word:

1. The most significant bit (left-most) in the Gray code is the same as the corresponding MSB in the binary number.
2. Going from left to right, add each adjacent pair of binary code bits to get the next Gray code bit. Discard carries.

For example, the conversion of the binary number 10110 to Gray code is as follows:

$$\begin{array}{ccccccccccccccccc} 1 & - & + & \rightarrow & 0 & - & + & \rightarrow & 1 & - & + & \rightarrow & 1 & - & + & \rightarrow & 0 & \text{Binary} \\ \downarrow & & & & \downarrow & & & & \downarrow & & & & \downarrow & & & & \downarrow & \\ 1 & & & & 1 & & & & 1 & & & & 0 & & & & 1 & \text{Gray} \end{array}$$

The Gray code is 11101.

Gray-to-Binary Conversion To convert from Gray code to binary, use a similar method; however, there are some differences. The following rules apply:

1. The most significant bit (left-most) in the binary code is the same as the corresponding bit in the Gray code.
2. Add each binary code bit generated to the Gray code bit in the next adjacent position. Discard carries.

For example, the conversion of the Gray code word 11011 to binary is as follows:

$$\begin{array}{cccccccccc} 1 & & 1 & & 0 & & 1 & & 1 & \text{Gray} \\ \downarrow & + \nearrow & \downarrow & + \nearrow & \downarrow & + \nearrow & \downarrow & + \nearrow & \downarrow & \\ 1 & & 0 & & 0 & & 1 & & 0 & \text{Binary} \end{array}$$

The binary number is 10010.

Logic gate

- The term logic is applied to digital circuits used to implement logic functions.
- A circuit that performs a specified logic operation (AND, OR) is called a logic gate.

NOT

FIGURE 1–16

The NOT operation.

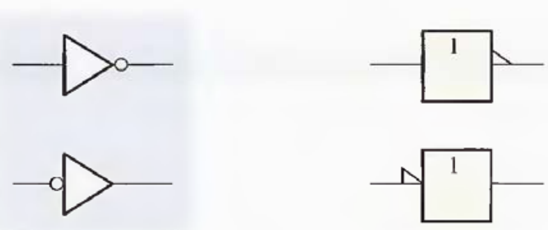
- The NOT operation changes one logic level to the opposite logic level.
- When the input is HIGH (1), the output is LOW (0). When the input is LOW, the output is HIGH.
- The NOT operation is implemented by a logic circuit known as an inverter.

NOT GATE

The inverter changes one logic level to the opposite level. In terms of bits, it changes a 1 to a 0 and a 0 to a 1.

FIGURE 3–1

Standard logic symbols for the inverter (ANSI/IEEE Std. 91-1984).



(a) Distinctive shape symbols with negation indicators

(b) Rectangular outline symbols with polarity indicators

TABLE 3–1

Inverter truth table.

INPUT	OUTPUT
LOW (0)	HIGH (1)
HIGH (1)	LOW (0)

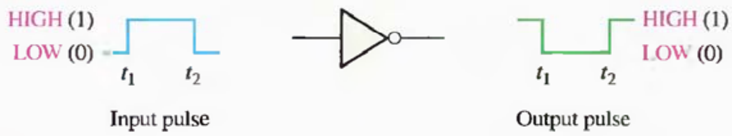


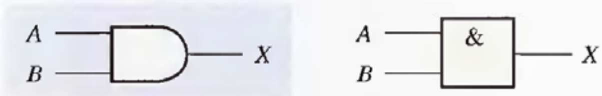
FIGURE 3–2

The operation of an inverter (NOT circuit) can be expressed as follows: If the input variable is called A and the output variable is called X, then

$$X = \overline{A}$$

AND GATE

For a 2-input AND gate, output X is HIGH only when inputs A and B are HIGH; X is LOW when either A or B is LOW, or when both A and B are LOW.



(a) Distinctive shape

(b) Rectangular outline with the

TABLE 3–2

Truth table for a 2-input AND gate.

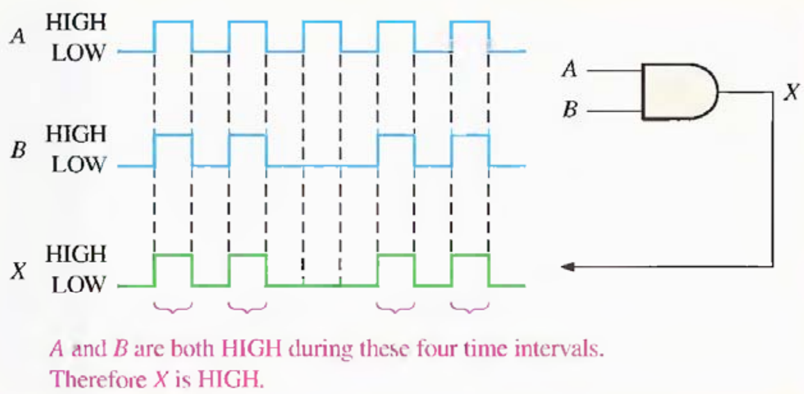
INPUTS		OUTPUT
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

1 = HIGH, 0 = LOW

Boolean expression is

$$X = AB$$

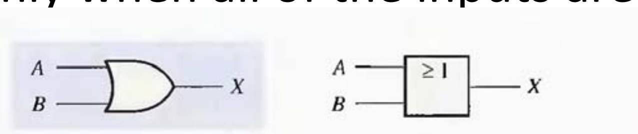
If two waveforms, A and B, are applied to the AND gate inputs as in Figure 3–11, what is the resulting output waveform?



▲ FIGURE 3–11

OR GATE

- An OR gate produces a HIGH on the output when any of the inputs is HIGH. The output is LOW only when all of the inputs are LOW.



(a) Distinctive shape

(b) Rectangular outline with the OR (≥ 1) qualifying symbol

INPUTS		OUTPUT
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

1 = HIGH, 0 = LOW

◄ TABLE 3–5

Truth table for a 2-input OR gate.

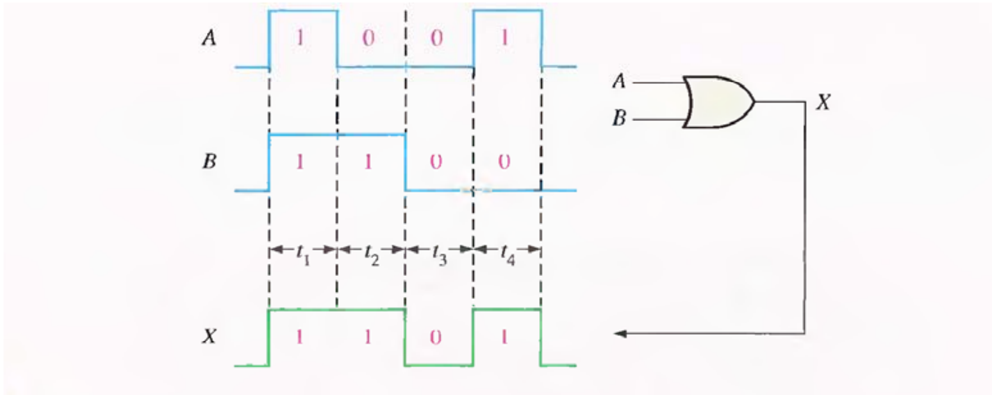
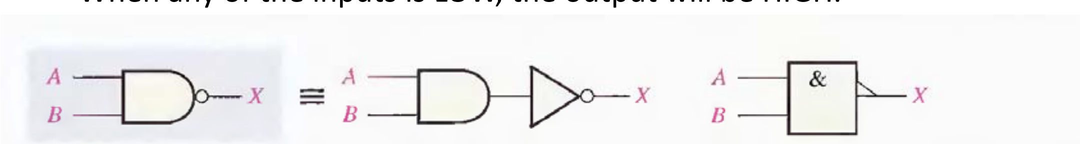


FIGURE 3–19

Example of OR gate operation with a timing diagram showing input and output time relationships.

NAND Gate

- A NAND gate produces a LOW output only when all the inputs are HIGH. When any of the inputs is LOW, the output will be HIGH.



(a) Distinctive shape, 2-input NAND gate and its NOT/AND equivalent

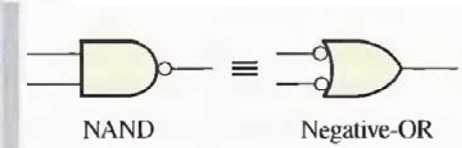
(b) Rectangular outline, 2-input NAND gate with polarity indicator

TABLE 3–7

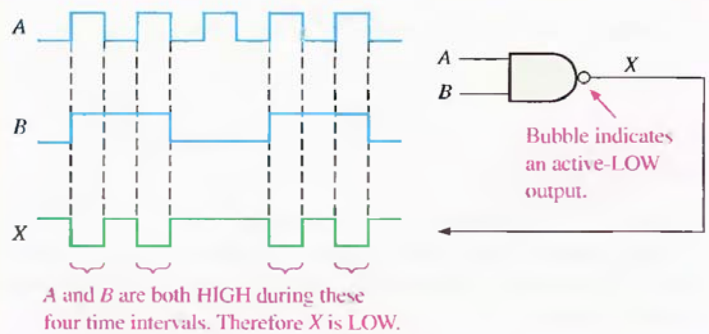
Truth table for a 2-input NAND gate.

INPUTS A	INPUTS B	OUTPUT X
0	0	1
0	1	1
1	0	1
1	1	0

1 = HIGH, 0 = LOW.



If the two waveforms A and B shown in Figure 3–27 are applied to the NAND gate inputs, determine the resulting output waveform.



Logic Expressions for a NAND Gate

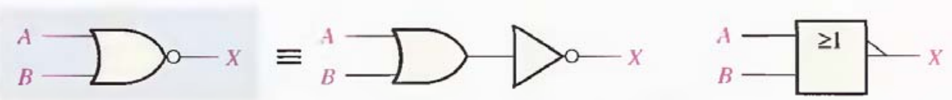
The Boolean expression for the output of a 2-input NAND gate is

$$X = \overline{AB}$$

A	B	$\overline{AB} = X$
0	0	$\overline{0 \cdot 0} = \overline{0} = 1$
0	1	$\overline{0 \cdot 1} = \overline{0} = 1$
1	0	$\overline{1 \cdot 0} = \overline{0} = 1$
1	1	$\overline{1 \cdot 1} = \overline{1} = 0$

NOR GATE

- A NOR gate produces a LOW output when any of its inputs is HIGH. Only when all of its inputs are LOW is the output HIGH.



(a) Distinctive shape, 2-input NOR gate and its NOT/OR equivalent

(b) Rectangular outline, 2-input NOR gate with polarity indicator

INPUTS		OUTPUT
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	0

1 = HIGH, 0 = LOW.

TABLE 3–9

Truth table for a 2-input NOR gate.

If the two waveforms shown in Figure 3–35 are applied to a NOR gate, what is the resulting output waveform?

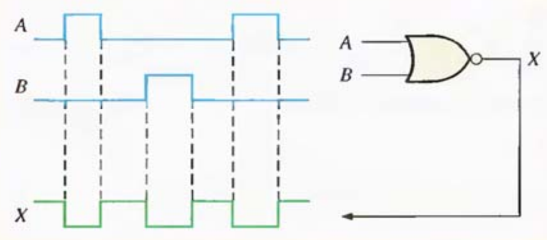


FIGURE 3–35

Solution Whenever any input of the NOR gate is HIGH, the output is LOW as shown by the output waveform X in the timing diagram.

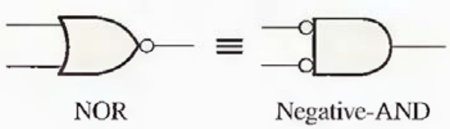


FIGURE 3–37

Standard symbols representing the two equivalent operations of a NOR gate.

Logic Expressions for a NOR Gate

The Boolean expression for the output of a 2-input NOR gate can be written as

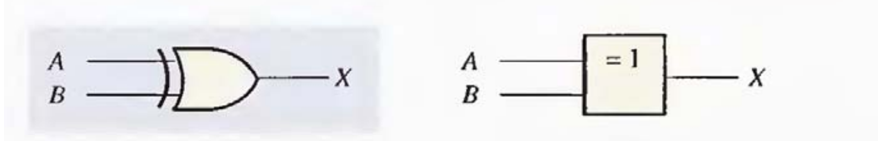
$$X = \overline{A + B}$$

TABLE 3–10

A	B	$\overline{A+B} = X$
0	0	$\overline{0+0} = \overline{0} = 1$
0	1	$\overline{0+1} = \overline{1} = 0$
1	0	$\overline{1+0} = \overline{1} = 0$
1	1	$\overline{1+1} = \overline{1} = 0$

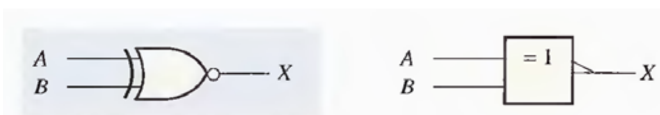
XOR GATE

- The output of an exclusive-OR gate is HIGH only when the two inputs are at opposite logic levels.



(a) Distinctive shape (b) Rectangular outline with the XOR

- When the two input logic levels are opposite. the output of the exclusive-NOR gate is LOW.



(a) Distinctive shape (b) Rectangular outline

TABLE 3–12

Truth table for an exclusive-NOR gate.

INPUTS		OUTPUT
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	1

FIGURE 3–46

Example of exclusive-OR gate operation with pulse waveform inputs.