

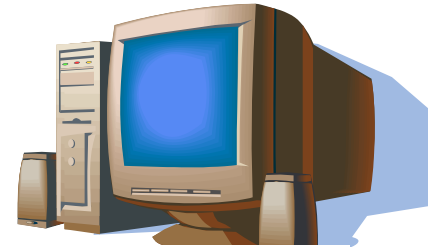
Digital Systems and Binary Numbers

What you will I Learn in this Course?

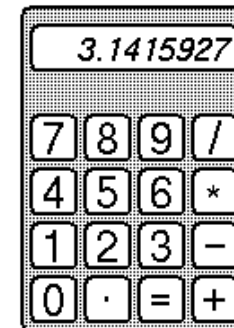
- ❖ Towards the end of this course, you should be able to:
 - ✧ Carry out arithmetic computation in various number systems
 - ✧ Apply rules of Boolean algebra to simplify Boolean expressions
 - ✧ Translate truth tables into equivalent Boolean expressions and logic gate implementations and vice versa
 - ✧ Design efficient combinational and sequential logic circuit implementations from functional description of digital systems
 - ✧ Use software tools to simulate and verify the operation of logic circuits

1.1 Digital Systems

❖ Digital Computer



❖ Handheld Calculator



❖ Digital Watch

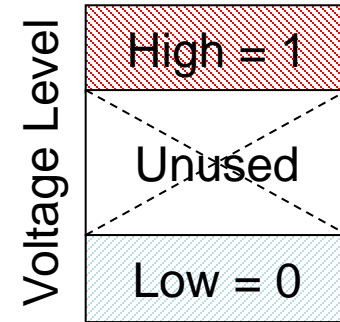


Is it Worth the Effort?

- ❖ Absolutely!
- ❖ Digital circuits are employed in the design of:
 - ✧ Digital computers
 - ✧ Data communication
 - ✧ Digital phones
 - ✧ Digital cameras
 - ✧ Digital TVs, etc.
- ❖ This course provides the fundamental concepts and the basic tools for the design of digital circuits and systems

How do Computers Represent Digits?

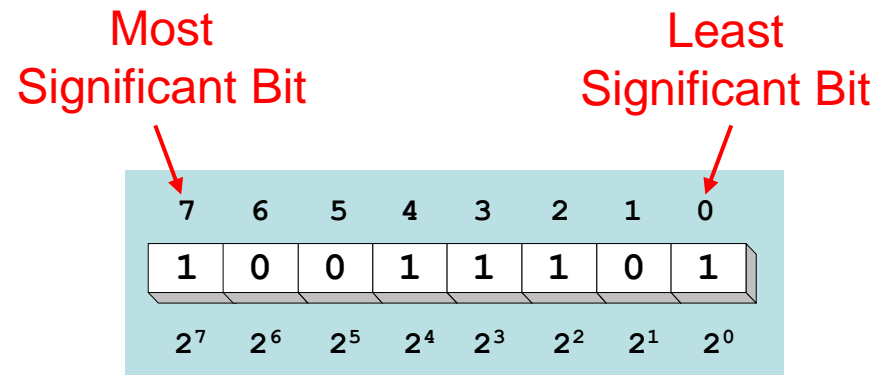
- ❖ Binary digits (0 and 1) are the simplest to represent
- ❖ Using electric voltage
 - ✧ Used in processors and digital circuits
 - ✧ High voltage = 1, Low voltage = 0
- ❖ Using electric charge
 - ✧ Used in memory cells
 - ✧ Charged memory cell = 1, discharged memory cell = 0
- ❖ Using magnetic field
 - ✧ Used in magnetic disks, magnetic polarity indicates 1 or 0
- ❖ Using light
 - ✧ Used in optical disks, optical lens can sense the light or not



Binary Numbers

- ❖ Each binary digit (called a bit) is either 1 or 0
- ❖ Bits have no inherent meaning, they can represent ...

- ✧ Unsigned and signed integers
- ✧ Fractions
- ✧ Characters
- ✧ Images, sound, etc.



❖ Bit Numbering

- ✧ Least significant bit (LSB) is rightmost (bit 0)
- ✧ Most significant bit (MSB) is leftmost (bit 7 in an 8-bit number)

Decimal Value of Binary Numbers

- ❖ Each bit represents a power of 2
- ❖ Every binary number is a sum of powers of 2
- ❖ Decimal Value = $(d_{n-1} \times 2^{n-1}) + \dots + (d_1 \times 2^1) + (d_0 \times 2^0)$
- ❖ Binary $(10011101)_2 = 2^7 + 2^4 + 2^3 + 2^2 + 1 = 157$

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

Some common
powers of 2



2^n	Decimal Value	2^n	Decimal Value
2^0	1	2^8	256
2^1	2	2^9	512
2^2	4	2^{10}	1024
2^3	8	2^{11}	2048
2^4	16	2^{12}	4096
2^5	32	2^{13}	8192
2^6	64	2^{14}	16384
2^7	128	2^{15}	32768

Positional Number Systems

Different Representations of Natural Numbers

XXVII Roman numerals (not positional)

27 Radix-10 or **decimal** number (positional)

11011_2 Radix-2 or **binary** number (also positional)

Fixed-radix positional representation with n digits

Number N in radix $r = (d_{n-1}d_{n-2} \dots d_1d_0)_r$

$$N_r \text{ Value} = d_{n-1} \times r^{n-1} + d_{n-2} \times r^{n-2} + \dots + d_1 \times r + d_0$$

$$\text{Examples: } (11011)_2 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2 + 1 = 27$$

$$(2107)_8 = 2 \times 8^3 + 1 \times 8^2 + 0 \times 8 + 7 = 1095$$

Convert Decimal to Binary

- ❖ Repeatedly divide the decimal integer by 2
- ❖ Each remainder is a binary digit in the translated value
- ❖ Example: Convert 37_{10} to Binary

Division	Quotient	Remainder
$37 / 2$	18	1
$18 / 2$	9	0
$9 / 2$	4	1
$4 / 2$	2	0
$2 / 2$	1	0
$1 / 2$	0	1

← least significant bit

$$37 = (100101)_2$$

← most significant bit

← stop when quotient is zero

Decimal to Binary Conversion

- ❖ $N = (d_{n-1} \times 2^{n-1}) + \dots + (d_1 \times 2^1) + (d_0 \times 2^0)$
- ❖ Dividing N by 2 we first obtain
 - ✧ $\text{Quotient}_1 = (d_{n-1} \times 2^{n-2}) + \dots + (d_2 \times 2) + d_1$
 - ✧ $\text{Remainder}_1 = d_0$
 - ✧ Therefore, first remainder is least significant bit of binary number
- ❖ Dividing first quotient by 2 we first obtain
 - ✧ $\text{Quotient}_2 = (d_{n-1} \times 2^{n-3}) + \dots + (d_3 \times 2) + d_2$
 - ✧ $\text{Remainder}_2 = d_1$
- ❖ Repeat dividing quotient by 2
 - ✧ Stop when new quotient is equal to zero
 - ✧ Remainders are the bits from least to most significant bit

Popular Number Systems

❖ Binary Number System: Radix = 2

- ✧ Only two digit values: 0 and 1
- ✧ Numbers are represented as 0s and 1s

❖ Octal Number System: Radix = 8

- ✧ Eight digit values: 0, 1, 2, ..., 7

❖ Decimal Number System: Radix = 10

- ✧ Ten digit values: 0, 1, 2, ..., 9

❖ Hexadecimal Number Systems: Radix = 16

- ✧ Sixteen digit values: 0, 1, 2, ..., 9, A, B, ..., F
- ✧ A = 10, B = 11, ..., F = 15

❖ Octal and Hexadecimal numbers can be converted easily to Binary and vice versa

Octal and Hexadecimal Numbers

- ❖ Octal = Radix 8
- ❖ Only eight digits: 0 to 7
- ❖ Digits 8 and 9 not used
- ❖ Hexadecimal = Radix 16
- ❖ 16 digits: 0 to 9, A to F
- ❖ A=10, B=11, ..., F=15
- ❖ First 16 decimal values (0 to 15) and their values in binary, octal and hex.

Memorize table

Decimal Radix 10	Binary Radix 2	Octal Radix 8	Hex Radix 16
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Binary, Octal, and Hexadecimal

- ❖ Binary, Octal, and Hexadecimal are related:
Radix 16 = 2^4 and Radix 8 = 2^3
- ❖ Hexadecimal digit = 4 bits and Octal digit = 3 bits
- ❖ Starting from least-significant bit, group each 4 bits into a hex digit or each 3 bits into an octal digit
- ❖ Example: Convert 32-bit number into octal and hex

3	5	3	0	5	5	2	3	6	2	4	Octal																			
1	1	1	0	1	0	1	1	0	0	0	1	0	1	1	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	32-bit binary
E	B	1	6	A	7	9	4	Hexadecimal																						

Converting Octal & Hex to Decimal

❖ Octal to Decimal: $N_8 = (d_{n-1} \times 8^{n-1}) + \dots + (d_1 \times 8) + d_0$

❖ Hex to Decimal: $N_{16} = (d_{n-1} \times 16^{n-1}) + \dots + (d_1 \times 16) + d_0$

❖ Examples:

$$(7204)_8 = (7 \times 8^3) + (2 \times 8^2) + (0 \times 8) + 4 = 3716$$

$$(3BA4)_{16} = (3 \times 16^3) + (11 \times 16^2) + (10 \times 16) + 4 = 15268$$

Converting Decimal to Hexadecimal

- ❖ Repeatedly divide the decimal integer by 16
- ❖ Each remainder is a hex digit in the translated value
- ❖ Example: convert 422 to hexadecimal

Division	Quotient	Remainder
422 / 16	26	6
26 / 16	1	A
1 / 16	0	1

← least significant digit

← most significant digit

$$422 = (1A6)_{16}$$

← stop when
quotient is zero

- ❖ To convert decimal to octal divide by 8 instead of 16

Important Properties

- ❖ How many possible digits can we have in Radix r ?

r digits: 0 to $r - 1$

- ❖ What is the result of adding 1 to the largest digit in Radix r ?

Since digit r is not represented, result is $(10)_r$ in Radix r

Examples: $1_2 + 1 = (10)_2$ $7_8 + 1 = (10)_8$

$9_{10} + 1 = (10)_{10}$ $F_{16} + 1 = (10)_{16}$

- ❖ What is the largest value using 3 digits in Radix r ?

In binary: $(111)_2 = 2^3 - 1$

In octal: $(777)_8 = 8^3 - 1$

In decimal: $(999)_{10} = 10^3 - 1$

In Radix r :

largest value = $r^3 - 1$

Important Properties - cont'd

❖ How many possible values can be represented ...

Using n binary digits?

2^n values: 0 to $2^n - 1$

Using n octal digits

8^n values: 0 to $8^n - 1$

Using n decimal digits?

10^n values: 0 to $10^n - 1$

Using n hexadecimal digits

16^n values: 0 to $16^n - 1$

Using n digits in Radix r ?

r^n values: 0 to $r^n - 1$

Representing Fractions

❖ A number N_r in *radix* r can also have a fraction part:

$$N_r = \underbrace{d_{n-1} d_{n-2} \dots d_1 d_0}_{\text{Integer Part}} \cdot \underbrace{d_{-1} d_{-2} \dots d_{-m+1} d_{-m}}_{\text{Fraction Part}} \quad 0 \leq d_i < r$$

Radix Point

❖ The number N_r represents the value:

$$N_r = d_{n-1} \times r^{n-1} + \dots + d_1 \times r + d_0 + \quad \text{(Integer Part)}$$
$$d_{-1} \times r^{-1} + d_{-2} \times r^{-2} \dots + d_{-m} \times r^{-m} \quad \text{(Fraction Part)}$$

$$N_r = \sum_{i=0}^{i=n-1} d_i \times r^i + \sum_{j=-m}^{j=-1} d_j \times r^j$$

Examples of Numbers with Fractions

$$\diamond (2409.87)_{10} = 2 \times 10^3 + 4 \times 10^2 + 9 + 8 \times 10^{-1} + 7 \times 10^{-2}$$

$$\diamond (1101.1001)_2 = 2^3 + 2^2 + 2^0 + 2^{-1} + 2^{-4} = 13.5625$$

$$\diamond (703.64)_8 = 7 \times 8^2 + 3 + 6 \times 8^{-1} + 4 \times 8^{-2} = 451.8125$$

$$\diamond (A1F.8)_{16} = 10 \times 16^2 + 16 + 15 + 8 \times 16^{-1} = 2591.5$$

$$\diamond (423.1)_5 = 4 \times 5^2 + 2 \times 5 + 3 + 5^{-1} = 113.2$$

$$\diamond (263.5)_6$$

Digit 6 is NOT allowed in radix 6

Converting Decimal Fraction to Binary

- ❖ Convert $N = 0.6875$ to Radix 2
- ❖ Solution: **Multiply** N by 2 repeatedly & collect integer bits

Multiplication	New Fraction	Bit	
$0.6875 \times 2 = 1.375$	0.375	1	→ First fraction bit
$0.375 \times 2 = 0.75$	0.75	0	
$0.75 \times 2 = 1.5$	0.5	1	
$0.5 \times 2 = 1.0$	0.0	1	→ Last fraction bit

- ❖ Stop when new fraction = 0.0, or when enough fraction bits are obtained
- ❖ Therefore, $N = 0.6875 = (0.1011)_2$
- ❖ Check $(0.1011)_2 = 2^{-1} + 2^{-3} + 2^{-4} = 0.6875$

Converting Fraction to any Radix r

- ❖ To convert fraction N to any radix r

$$N_r = (0.d_{-1} d_{-2} \dots d_{-m})_r = d_{-1} \times r^{-1} + d_{-2} \times r^{-2} \dots + d_{-m} \times r^{-m}$$

- ❖ Multiply N by r to obtain d_{-1}

$$N_r \times r = \textcolor{red}{d_{-1}} + d_{-2} \times r^{-1} \dots + d_{-m} \times r^{-m+1}$$

- ❖ The integer part is the digit $\textcolor{red}{d_{-1}}$ in radix r

- ❖ The new fraction is $d_{-2} \times r^{-1} \dots + d_{-m} \times r^{-m+1}$

- ❖ Repeat multiplying the new fractions by r to obtain $\textcolor{red}{d_{-2}} \textcolor{red}{d_{-3}} \dots$

- ❖ Stop when new fraction becomes 0.0 or enough fraction digits are obtained

More Conversion Examples

- ❖ Convert $N = 139.6875$ to Octal (Radix 8)
- ❖ Solution: $N = 139 + 0.6875$ (split integer from fraction)
- ❖ The integer and fraction parts are converted separately

Division	Quotient	Remainder
$139 / 8$	17	3
$17 / 8$	2	1
$2 / 8$	0	2

Multiplication	New Fraction	Digit
$0.6875 \times 8 = 5.5$	0.5	5
$0.5 \times 8 = 4.0$	0.0	4

- ❖ Therefore, $139 = (213)_8$ and $0.6875 = (0.54)_8$
- ❖ Now, join the integer and fraction parts with radix point
 $N = 139.6875 = (213.54)_8$

Conversion Procedure to Radix r

- ❖ To convert decimal number N (with fraction) to radix r
- ❖ Convert the Integer Part
 - ✧ Repeatedly divide the integer part of number N by the radix r and **save the remainders**. The integer digits in radix r are the remainders in **reverse order** of their computation. If radix $r > 10$, then convert all remainders > 10 to digits A, B, ... etc.
- ❖ Convert the Fractional Part
 - ✧ Repeatedly multiply the fraction of N by the radix r and **save the integer digits** that result. The fraction digits in radix r are the integer digits in **order of their computation**. If the radix $r > 10$, then convert all digits > 10 to A, B, ... etc.
- ❖ Join the result together with the radix point

Simplified Conversions

- ❖ Converting fractions between Binary, Octal, and Hexadecimal can be simplified
- ❖ Starting at the radix pointing, the integer part is converted from right to left and the fractional part is converted from left to right
- ❖ Group 4 bits into a hex digit or 3 bits into an octal digit

← integer: right to left — — fraction: left to right →

7		2		6		1		3		.	2		4		7		4		5		2		Octal										
1	1	1	0	1	0	1	1	0	0	0	1	0	1	1	.	0	1	0	1	0	0	1	1	1	1	0	0	1	0	1	0	1	Binary
7		5		8		B		.	5		3		C		A		8		Hexadecimal														

- ❖ Use binary to convert between octal and hexadecimal

Important Properties of Fractions

- ❖ How many fractional values exist with m fraction bits?

2^m fractions, because each fraction bit can be 0 or 1

- ❖ What is the largest fraction value if m bits are used?

Largest fraction value = $2^{-1} + 2^{-2} + \dots + 2^{-m} = 1 - 2^{-m}$

Because if you add 2^{-m} to largest fraction you obtain 1

- ❖ In general, what is the largest fraction value if m fraction digits are used in radix r ?

Largest fraction value = $r^{-1} + r^{-2} + \dots + r^{-m} = 1 - r^{-m}$

For decimal, largest fraction value = $1 - 10^{-m}$

For hexadecimal, largest fraction value = $1 - 16^{-m}$

Complements of Numbers

- ❖ Complements are used for simplifying the subtraction operation and for easy manipulation of certain logical rules and events
- ❖ Two types of complements for each *base- r* system:
 - radix complements (r 's complements)
 - diminished radix complements ($(r - 1)$'s complements)
- ❖ Diminished radix complement
 - Given a number N in base r having n digits, the $(r-1)$'s complement of N is defined as $(r^n - 1) - N$

Diminished Radix Complements

❖ For decimal number, $r = 10$, $r-1=9$, $n=6$

- 9's complement of 546700 = $999999 - 546700 = 453299$
- 9's complement of 012398 = $999999 - 012398 = 987601$

❖ For binary number, $r = 2$, $r-1 = 1$, $n=7$

- 1's complement of 1011000 = $1111111 - 1011000 = 0100111$
- 1's complement of 0101101 = $1111111 - 0101101 = 1010010$

Radix Complements

- ❖ The r 's complement of an n -digit number N is defined as
 - $(r^n - N, \text{ for } N \neq 0 \text{ and } 0 \text{ for } N = 0)$
- ❖ Examples:
 - 1) 10's complement of 546700 = $1000000 - 546700 = 453300$
 - 2) 10's complement of 012398 = $1000000 - 012398 = 987602$
 - 3) 2's complement of 1011000 = $10000000 - 1011000 = 0101000$
 - 4) 2's complement of 0101101 = $10000000 - 0101101 = 1010011$
- ❖ The 2's complement can be derived by **1's complement + 1**
- ❖ The complement of the complement restores the number to its original value
- ❖ If there is a radix point, the radix point **is temporarily removed during the process, and restored in the same position afterwards**

Subtraction using 10's complement

- ❖ For subtracting two numbers using 10's complement, we first have to find the 10's complement of the subtrahend, and then we will add this complement value with the minuend. There are two possible cases when we subtract the numbers using 10's complement.
- ❖ **Case 1: When the subtrahend is smaller than the minuend.**
 - ✧ For subtracting the smaller number from the larger number using 10's complement, we will find the 10's complement of the subtrahend and then we will add this complement value with the minuend. By adding both these values, the result will come in the formation of carry. We ignore this carry and the remaining digits will be the answer.
- ❖ **Case 2: When the subtrahend is greater than the minuend.**
 - ✧ In this case, when we add the complement value and the minuend, the result will not come in the formation of carry. This indicates that the number is negative and for finding the final result, we need to find the 10's complement of the result obtained by adding complement value of subtrahend and minuend.

Examples

❖ **E.g.** using 10's comp do $72532 - 3250$

72532

+ 96750 \rightarrow 10's comp of 3250

169282

Answer = 69282

❖ **E.g.** Using 10's comp do $3250 - 72532$

03250

+ 27468 \rightarrow 10's comp of 72532

30718 \rightarrow no end carry

Answer = $-(10's \text{ comp of } 30718) = -69282$

Subtraction using 9's complement

- ❖ For subtracting two numbers using 9's complement, we first have to find the 9's complement of the subtrahend and then we will add this complement value with the minuend. There are two possible cases when we subtract the numbers using 9's complement.
- ❖ **Case 1: When the subtrahend is smaller than the minuend.**
 - ✧ For subtracting the smaller number from the larger number using 9's complement, we will find the 9's complement of the subtrahend, and then we will add this complement value with the minuend. By adding both these values, the result will come in the formation of carry. At last, we will add this carry to the result obtained previously.
- ❖ **Case 2: When the subtrahend is greater than the minuend.**
 - ✧ In this case, when we add the complement value and the minuend, the result will not come in the formation of carry. This indicates that the number is negative, and for finding the final result, we need to find the 9's complement of the result.

Examples

❖ Example using 9's complement:

- do $72532 - 3250$

72532

+ 96749 \rightarrow 9's comp of 3250

169281

+ 1 \rightarrow end around carry

69282

- do $3250 - 72532$

03250

+ 27467 \rightarrow 9's comp of 72532

30717 $\rightarrow -(9's \text{ comp of } 30717) = -69282$

2's Complement Subtraction

13-6

$$\begin{array}{r} 00001101 \\ -00000110 \\ \hline 00000111 \end{array}$$

2' compl. of 6 : 11111010

$$\begin{array}{r} 00001101 \\ + 11111010 \\ \hline \underline{1} 00000111 \quad (\text{discard } 2^8) \end{array}$$

6-13

2' compl. of 13: 11110011

$$\begin{array}{r} 00000110 \\ + 11110011 \\ \hline 11111001 \quad (2' \text{ compl. of } 7) \end{array}$$

1's complement subtraction

(i) 110101 – 100101

Solution:

1's complement of 10011 is 011010. Hence

Minued -		1	1	0	1	0	1
1's complement of subtrahend -		0	1	1	0	1	0
Carry over -	1	0	0	1	1	1	1
						1	
							0
							1
							0
							0
							0
							0

(ii) 101011 – 111001

Solution:

1's complement of 111001 is 000110. Hence

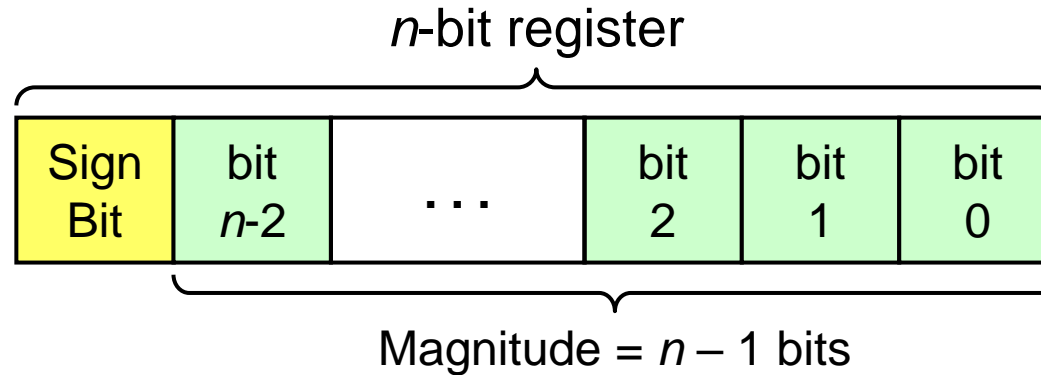
Minued -		1	0	1	0	1	1
1's complement -		0	0	0	1	1	0

Hence the difference is – 1 1 1 0

Signed Numbers

- ❖ Several ways to represent a signed number
 - ✧ Sign-Magnitude
 - ✧ 1's complement
 - ✧ 2's complement
- ❖ Divide the range of values into 2 equal parts
 - ✧ First part corresponds to the positive numbers (≥ 0)
 - ✧ Second part correspond to the negative numbers (< 0)
- ❖ The 2's complement representation is widely used
 - ✧ Has many advantages over other representations

Sign-Magnitude Representation



- ❖ Independent representation of the sign and magnitude
- ❖ Leftmost bit is the sign bit: 0 is positive and 1 is negative
- ❖ Using n bits, largest represented magnitude = $2^{n-1} - 1$

Sign-magnitude
representation of +45
using 8-bit register

0	0	1	0	1	1	0	1
---	---	---	---	---	---	---	---

Sign-magnitude
representation of -45
using 8-bit register

1	0	1	0	1	1	0	1
---	---	---	---	---	---	---	---

Properties of Sign-Magnitude

- ❖ Two representations for zero: +0 and -0
- ❖ Symmetric range of represented values:
 - For n-bit register, range is from $-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$
 - For example using 8-bit register, range is -127 to +127
- ❖ Hard to implement addition and subtraction
 - ✧ Sign and magnitude parts have to be processed independently
 - ✧ Sign bit should be examined to determine addition or subtraction
 - Addition is converted into subtraction when adding numbers of different signs
 - ✧ Need a different circuit to perform addition and subtraction
 - Increases the cost of the logic circuit

2's Complement Representation

- ❖ Almost all computers today use 2's complement to represent signed integers

- ❖ A simple definition for 2's complement:

Given a binary number N

The 2's complement of $N = 1$'s complement of $N + 1$

- ❖ Example: 2's complement of $(01101001)_2 =$

$$(10010110)_2 + 1 = (10010111)_2$$

- ❖ If N consists of n bits then

$$\text{2's complement of } N = 2^n - N$$

Computing the 2's Complement

starting value	$00100100_2 = +36$
step1: reverse the bits (1's complement)	11011011_2
step 2: add 1 to the value from step 1	$+ \quad \quad 1_2$
sum = 2's complement representation	$11011100_2 = -36$

2's complement of 11011100_2 (-36) = $00100011_2 + 1 = 00100100_2 = +36$

The 2's complement of the 2's complement of N is equal to N

Another way to obtain the 2's complement:

Start at the least significant 1

Leave all the 0s to its right unchanged

Complement all the bits to its left

Binary Value

= $00100\boxed{1}00$ least significant 1

2's Complement

= $11011\boxed{1}00$

Unsigned and Signed Value

❖ Positive numbers

✧ Signed value = Unsigned value

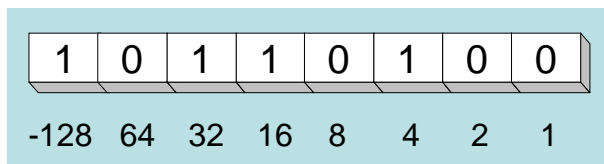
❖ Negative numbers

✧ Signed value = Unsigned value – 2^n

✧ n = number of bits

❖ Negative weight for MSB

✧ Another way to obtain the signed value is to assign a negative weight to most-significant bit




$$= -128 + 32 + 16 + 4 = -76$$

8-bit Binary value	Unsigned value	Signed value
00000000	0	0
00000001	1	+1
00000010	2	+2
...
01111110	126	+126
01111111	127	+127
10000000	128	-128
10000001	129	-127
...
11111110	254	-2
11111111	255	-1

Properties of the 2's Complement

- ❖ The 2's complement of N is the negative of N
- ❖ The sum of N and 2's complement of N must be zero

The final carry is ignored

- ❖ Consider the 8-bit number $N = 00101100_2 = +44$
 $-44 = 2\text{'s complement of } N = 11010100_2$
 $00101100_2 + 11010100_2 = \mathbf{1\ 00000000_2}$ (8-bit sum is 0)
 Ignore final carry

- ❖ In general: Sum of $N + 2\text{'s complement of } N = 2^n$
where 2^n is the final carry (1 followed by n 0's)
- ❖ There is only one zero: 2's complement of $0 = 0$

Ranges of Unsigned/Signed Integers

- ❖ For n -bit unsigned integers: Range is 0 to $(2^n - 1)$
- ❖ For n -bit signed integers: Range is -2^{n-1} to $(2^{n-1} - 1)$
- ❖ Positive range: 0 to $(2^{n-1} - 1)$
- ❖ Negative range: -2^{n-1} to -1

Storage Size	Unsigned Range	Signed Range
8 bits (byte)	0 to $(2^8 - 1) = 255$	$-2^7 = -128$ to $(2^7 - 1) = +127$
16 bits	0 to $(2^{16} - 1) = 65,535$	$-2^{15} = -32,768$ to $(2^{15} - 1) = +32,767$
32 bits	0 to $(2^{32} - 1) =$ 4,294,967,295	$-2^{31} = -2,147,483,648$ to $(2^{31} - 1) = +2,147,483,647$
64 bits	0 to $(2^{64} - 1) =$ 18,446,744,073,709,551,615	$-2^{63} = -9,223,372,036,854,775,808$ to $(2^{63} - 1) = +9,223,372,036,854,775,807$

Two's Complement Special Cases

❖ Case 1

- ❖ $0 = 00000000$
- ❖ Bitwise not 11111111
- ❖ Add 1 to LSB $+1$
- ❖ Result $1\ 00000000$
- ❖ Overflow is ignored, so:
- ❖ $-0 = 0 \checkmark$

❖ $-128 = 10000000$

- ❖ bitwise not 01111111
- ❖ Add 1 to LSB $+1$
- ❖ Result 10000000
- ❖ Monitor MSB (sign bit)
- ❖ It should change during negation

Table 1-3: Signed Binary Numbers

Table 1.3
Signed Binary Numbers

Decimal	Signed-2's Complement	Signed-1's Complement	Signed Magnitude
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
−0	—	1111	1000
−1	1111	1110	1001
−2	1110	1101	1010
−3	1101	1100	1011
−4	1100	1011	1100
−5	1011	1010	1101
−6	1010	1001	1110
−7	1001	1000	1111
−8	1000	—	—

Arithmetic Addition

- The addition of two signed binary numbers with negative numbers represented in signed-2's-complement form is obtained from the addition of the two numbers, including their sign bits. A carry out of the sign-bit position is discarded
- In order to obtain a correct answer, we must ensure that the result has a sufficient number of bits to accommodate the sum
- If we start with two n -bit numbers and the sum occupies $n + 1$ bits, we say that an **overflow** occurs

Binary Addition

- ❖ Start with the least significant bit (rightmost bit)
- ❖ Add each pair of bits
- ❖ Include the carry in the addition, if present

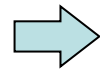
carry		1	1	1	1		
	0	0	1	1	0	1	1
	0	0	0	1	1	1	0
+	<hr/>						
	0	1	0	1	0	0	1
bit position:	7	6	5	4	3	2	1

(54)
(29)
(83)

Binary Subtraction

- ❖ When subtracting $A - B$, convert B to its 2's complement
- ❖ Add A to $(-B)$

$$\begin{array}{r} \text{borrow:} \quad 1 \quad 1 \quad \quad 1 \\ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \\ - \\ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \\ \hline 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \end{array}$$



$$\begin{array}{r} \text{carry:} \quad 1 \quad 1 \quad \quad 1 \quad 1 \\ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \\ + \\ \color{red}{1 \ 1 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0} \text{ (2's complement)} \\ \hline 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \text{ (same result)} \end{array}$$

- ❖ Final carry is ignored, because
 - ✧ Negative number is sign-extended with 1's
 - ✧ You can imagine infinite 1's to the left of a negative number
 - ✧ Adding the carry to the extended 1's produces extended zeros

Carry and Overflow

❖ Carry is important when ...

- ✧ Adding or subtracting **unsigned integers**
- ✧ Indicates that the **unsigned sum** is out of range
- ✧ Either < 0 or $>$ maximum unsigned n -bit value

❖ Overflow is important when ...

- ✧ Adding or subtracting **signed integers**
- ✧ Indicates that the **signed sum** is out of range

❖ Overflow occurs when

- ✧ Adding two positive numbers and the sum is negative
- ✧ Adding two negative numbers and the sum is positive
- ✧ Can happen because of the fixed number of sum bits

Carry and Overflow Examples

- ❖ We can have carry without overflow and vice-versa
- ❖ Four cases are possible (Examples are 8-bit numbers)

				1					
	0	0	0	0	1	1	1	1	15
+	0	0	0	0	1	0	0	0	8
<hr/>									
	0	0	0	1	0	1	1	1	23
Carry = 0 Overflow = 0									

1	1	1	1	1					
	0	0	0	0	1	1	1	1	15
+	1	1	1	1	1	0	0	0	248 (-8)
<hr/>									
	0	0	0	0	0	1	1	1	7
Carry = 1 Overflow = 0									

				1					
	0	1	0	0	1	1	1	1	79
+	0	1	0	0	0	0	0	0	64
<hr/>									
	1	0	0	0	1	1	1	1	143 (-113)
Carry = 0 Overflow = 1									

1				1		1			
	1	1	0	1	1	0	1	0	218 (-38)
+	1	0	0	1	1	1	0	1	157 (-99)
<hr/>									
	0	1	1	1	0	1	1	1	119
Carry = 1 Overflow = 1									

Addition of Numbers in Twos Complement Representation

$$\begin{array}{r} 1001 = -7 \\ +0101 = 5 \\ \hline 1110 = -2 \end{array}$$

(a) $(-7) + (+5)$

$$\begin{array}{r} 1100 = -4 \\ +0100 = 4 \\ \hline 10000 = 0 \end{array}$$

(b) $(-4) + (+4)$

$$\begin{array}{r} 0011 = 3 \\ +0100 = 4 \\ \hline 0111 = 7 \end{array}$$

(c) $(+3) + (+4)$

$$\begin{array}{r} 1100 = -4 \\ +1111 = -1 \\ \hline 11011 = -5 \end{array}$$

(d) $(-4) + (-1)$

$$\begin{array}{r} 0101 = 5 \\ +0100 = 4 \\ \hline 1001 = \text{overflow} \end{array}$$

(e) $(+5) + (+4)$

$$\begin{array}{r} 1001 = -7 \\ +1010 = -6 \\ \hline 10011 = \text{overflow} \end{array}$$

(f) $(-7) + (-6)$

Subtraction of Numbers in Twos Complement Representation (M - S)

$\begin{array}{r} 0010 = 2 \\ +1001 = -7 \\ \hline 1011 = -5 \end{array}$ <p>(a) M = 2 = 0010 S = 7 = 0111 -S = 1001</p>	$\begin{array}{r} 0101 = 5 \\ +1110 = -2 \\ \hline 10011 = 3 \end{array}$ <p>(b) M = 5 = 0101 S = 2 = 0010 -S = 1110</p>
$\begin{array}{r} 1011 = -5 \\ +1110 = -2 \\ \hline 11001 = -7 \end{array}$ <p>(c) M = -5 = 1011 S = 2 = 0010 -S = 1110</p>	$\begin{array}{r} 0101 = 5 \\ +0010 = 2 \\ \hline 0111 = 7 \end{array}$ <p>(d) M = 5 = 0101 S = -2 = 1110 -S = 0010</p>
$\begin{array}{r} 0111 = 7 \\ +0111 = 7 \\ \hline 1110 = \text{Overflow} \end{array}$ <p>(e) M = 7 = 0111 S = -7 = 1001 -S = 0111</p>	$\begin{array}{r} 1010 = -6 \\ +1100 = -4 \\ \hline 10110 = \text{Overflow} \end{array}$ <p>(f) M = -6 = 1010 S = 4 = 0100 -S = 1100</p>

Binary Codes

- ❖ How to represent characters, colors, etc?
- ❖ Define the set of all **represented elements**
- ❖ Assign a unique binary code to each element of the set
- ❖ Given n bits, a **binary code** is a mapping from the set of elements to a subset of the 2^n binary numbers
- ❖ Coding Numeric Data (example: coding decimal digits)
 - ✧ Coding must simplify common arithmetic operations
 - ✧ Tight relation to binary numbers
- ❖ Coding Non-Numeric Data (example: coding colors)
 - ✧ More flexible codes since arithmetic operations are not applied

Example of Coding Non-Numeric Data

- ❖ Suppose we want to code 7 colors of the rainbow
- ❖ As a minimum, we need 3 bits to define 7 unique values
- ❖ 3 bits define 8 possible combinations
- ❖ Only 7 combinations are needed
- ❖ Code 111 is not used
- ❖ Other assignments are also possible

Color	3-bit code
Red	000
Orange	001
Yellow	010
Green	011
Blue	100
Indigo	101
Violet	110

Minimum Number of Bits Required

- ❖ Given a set of M elements to be represented by a binary code, the **minimum number of bits**, n , should satisfy:

$$2^{(n-1)} < M \leq 2^n$$

$n = \lceil \log_2 M \rceil$ where $\lceil x \rceil$, called the **ceiling function**, is the integer greater than or equal to x

- ❖ How many bits are required to represent 10 decimal digits with a binary code?
- ❖ **Answer:** $\lceil \log_2 10 \rceil = 4$ bits can represent 10 decimal digits

Decimal Codes

- ❖ Binary number system is most natural for computers
- ❖ But people are used to the decimal number system
- ❖ Must convert decimal numbers to binary, do arithmetic on binary numbers, then convert back to decimal
- ❖ To simplify conversions, decimal codes can be used
- ❖ Define a binary code for each decimal digit
- ❖ Since 10 decimal digits exist, a 4-bit code is used
- ❖ But a 4-bit code gives 16 unique combinations
- ❖ 10 combinations are used and 6 will be unused

Binary Coded Decimal (BCD)

- ❖ Simplest binary code for decimal digits
- ❖ Only encodes ten digits from 0 to 9
- ❖ BCD is a **weighted code**
- ❖ The weights are 8,4,2,1
- ❖ Same weights as a binary number
- ❖ There are **six invalid code words**
1010, 1011, 1100, 1101, 1110, 1111
- ❖ Example on BCD coding:
 $13 \Leftrightarrow (0001\ 0011)_{\text{BCD}}$

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
Unused	1010
	...
	1111

Warning: Conversion or Coding?

- ❖ Do **NOT** mix up **conversion** of a decimal number to a binary number with **coding** a decimal number with a binary code
- ❖ $13_{10} = (1101)_2$ This is **conversion**
- ❖ $13 \Leftrightarrow (0001\ 0011)_{\text{BCD}}$ This is **coding**
- ❖ In general, coding requires more bits than conversion
- ❖ A number with n decimal digits is coded with $4n$ bits in BCD

BCD Arithmetic

- Given a BCD code, we use binary arithmetic to add the digits:

8	1000	Eight
<u>+5</u>	<u>+0101</u>	Plus 5
13	1101	is 13 (> 9)

- Note that the result is **MORE THAN 9**, so must be represented by two digits!

- To correct the digit, subtract 10 by adding 6 modulo 16.

8	1000	Eight
<u>+5</u>	<u>+0101</u>	Plus 5
13	1101	is 13 (> 9)
	<u>+0110</u>	so add 6
carry = 1	0011	leaving 3 + cy
0001	0011	Final answer (two digits)

BCD Addition Example

❖ Add 2905_{BCD} to 1897_{BCD} showing carries and digit corrections.

			1	1	1	
1897_{BCD}		0001	1000	1001	0111	
2905_{BCD}	+	<u>0010</u>	<u>1001</u>	<u>0000</u>	<u>0101</u>	
		0100	10010	1010	1100	
		<u>0000</u>	<u>0110</u>	<u>0110</u>	<u>0110</u>	
		0100	1000	0000	0010	
		4	8	0	2	

Gray Code

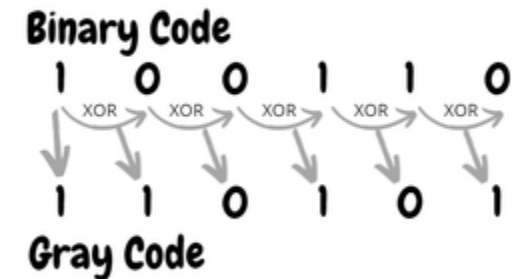
- ❖ The reflected binary code or Gray code is an ordering of the binary numeral system such that two successive values differ in only one bit (binary digit).
- ❖ Gray codes are very useful in the normal sequence of binary numbers generated by the hardware that may cause an error or ambiguity during the transition from one number to the next.

Decimal	Gray	Binary
00	0000	0000
01	0001	0001
02	0011	0010
03	0010	0011
04	0110	0100
05	0111	0101
06	0101	0110
07	0100	0111
08	1100	1000
09	1101	1001
10	1111	1010
11	1110	1011
12	1010	1100
13	1011	1101
14	1001	1110
15	1000	1111

Conversion

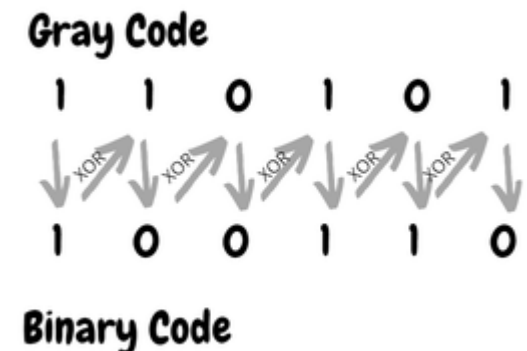
❖ From Binary to Gray:

- ❖ The most significant bit (MSB) of the Gray code is always equal to the MSB of the given Binary code.
- ❖ Other bits of the output Gray code can be obtained by **XORing** binary code bit at the index and previous index.



❖ From Gray to Binary:

- ❖ The Most Significant Bit (MSB) of the binary code is always equal to the MSB of the given binary number.
- ❖ Other bits of the output binary code can be obtained by checking gray code bit at that index. If current gray code bit is 0, then copy previous binary code bit, else copy invert of previous binary code bit.



Other Decimal Codes

- ❖ BCD, 5421, 2421, and 8 4 -2 -1 are **weighted codes**
- ❖ Excess-3 is not a weighted code
- ❖ 2421, 8 4 -2 -1, and Excess-3 are **self complementary codes**

Decimal	BCD 8421	5421 code	2421 code	8 4 -2 -1 code	Excess-3 code
0	0000	0000	0000	0000	0011
1	0001	0001	0001	0111	0100
2	0010	0010	0010	0110	0101
3	0011	0011	0011	0101	0110
4	0100	0100	0100	0100	0111
5	0101	1000	1011	1011	1000
6	0110	1001	1100	1010	1001
7	0111	1010	1101	1001	1010
8	1000	1011	1110	1000	1011
9	1001	1100	1111	1111	1100
Unused

Character Codes

❖ Character sets

- ✧ Standard ASCII: 7-bit character codes (0 – 127)
- ✧ Extended ASCII: 8-bit character codes (0 – 255)
- ✧ Unicode: 16-bit character codes (0 – 65,535)
- ✧ Unicode standard represents a universal character set
 - Defines codes for characters used in all major languages
 - Each character is encoded as 16 bits
- ✧ UTF-8: variable-length encoding used in HTML
 - Encodes all Unicode characters
 - Uses 1 byte for ASCII, but multiple bytes for other characters

❖ Null-terminated String

- ✧ Array of characters followed by a NULL character

Printable ASCII Codes

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	space	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

❖ Examples:

- ✧ ASCII code for space character = 20 (hex) = 32 (decimal)
- ✧ ASCII code for 'L' = 4C (hex) = 76 (decimal)
- ✧ ASCII code for 'a' = 61 (hex) = 97 (decimal)

Control Characters

- ❖ The first 32 characters of ASCII table are used for control
- ❖ Control character codes = 00 to 1F (hexadecimal)
 - ✧ Not shown in previous slide
- ❖ Examples of Control Characters
 - ✧ Character 0 is the **NULL** character \Rightarrow used to terminate a string
 - ✧ Character 9 is the **Horizontal Tab (HT)** character
 - ✧ Character 0A (hex) = 10 (decimal) is the **Line Feed (LF)**
 - ✧ Character 0D (hex) = 13 (decimal) is the **Carriage Return (CR)**
 - ✧ The LF and CR characters are used together
 - They advance the cursor to the beginning of next line
- ❖ One control character appears at end of ASCII table
 - ✧ Character 7F (hex) is the **Delete (DEL)** character

Binary Logic

- ❖ Deals with binary variables that take one of two discrete values
- ❖ Values of variables are called by a variety of very different names
 - ✧ high or low based on voltage representations in electronic circuits
 - ✧ true or false based on their usage to represent logic states
 - ✧ one (1) or zero (0) based on their values in Boolean algebra
 - ✧ open or closed based on its operation in gate logic
 - ✧ on or off based on its operation in switching logic
 - ✧ asserted or de-asserted based on its effect in digital systems

Basic Operations - AND

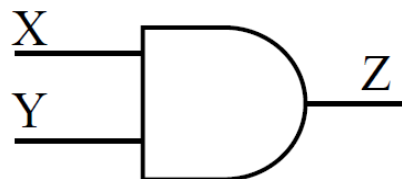
- Another Symbol is ".", e.g.

$$Z = X \text{ AND } Y \text{ or}$$

$$Z = X.Y \text{ or even}$$

$$Z = XY$$

- X and Y are inputs, Z is an output
- Z is equal to 1 if and only if $X = 1$ and $Y = 1$; $Z = 0$ otherwise (similar to the multiplication operation)
- Truth Table:
- Graphical symbol:



X	Y	Z=XY
0	0	0
0	1	0
1	0	0
1	1	1

Basic Operations - OR

- Another Symbol is "+", e.g.

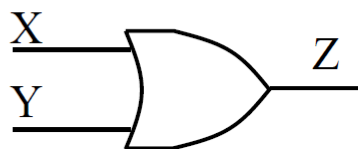
$$Z = X \text{ OR } Y \text{ or}$$

$$Z = X + Y$$

- X and Y are inputs, Z is an output
- Z is equal to 0 if and only if $X = 0$ and $Y = 0$; $Z = 1$ otherwise (similar to the addition operation)

- Truth Table:

- Graphical symbol:



X	Y	Z=X+Y
0	0	0
0	1	1
1	0	1
1	1	1

Basic Operations - NOT

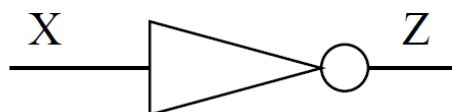
- Another Symbol is " $\overline{}$ ", e.g.

$$Z = \overline{X} \text{ or } Z = X'$$

- X is the input, Z is an output
- Z is equal to 0 if $X = 1$; $Z = 1$ otherwise
- Sometimes referred to as the complement or invert operation
- Truth Table:

X	Z=X'
0	1
1	0

- Graphical symbol:

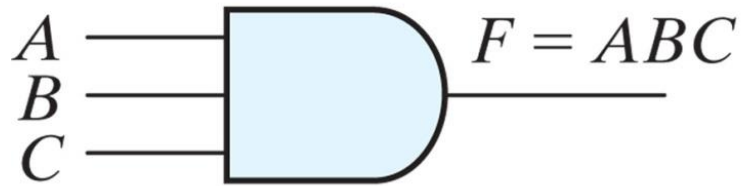


Two Input Gates – Timing Diagram

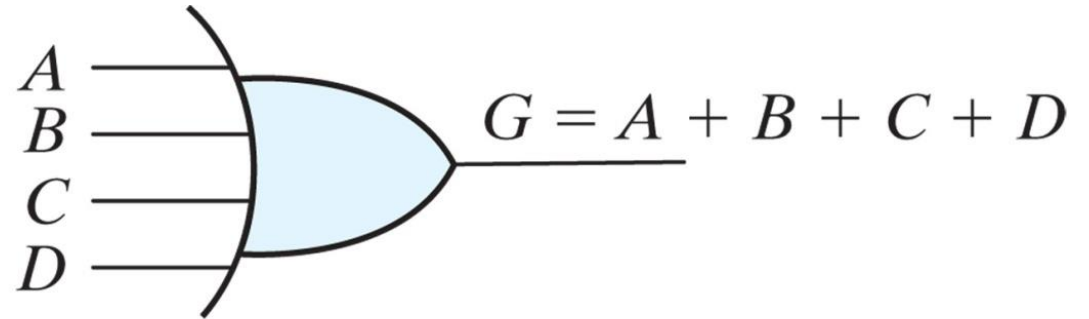


Copyright ©2013 Pearson Education, publishing as Prentice Hall

Gates with multiple inputs



(a) Three-input AND gate



(b) Four-input OR gate

Copyright ©2013 Pearson Education, publishing as Prentice Hall