數位電路設計

Digital Circuit Design

莊仁輝

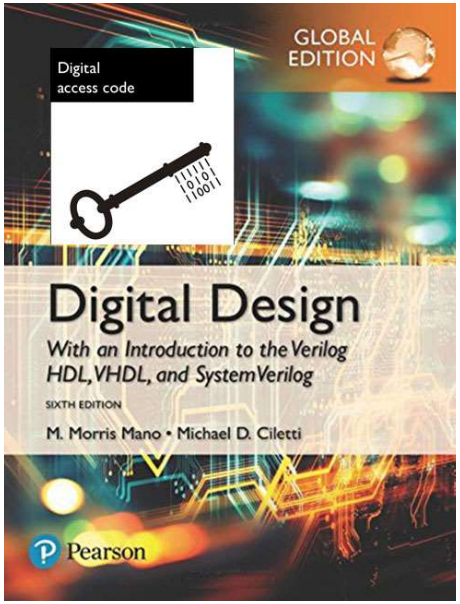
陽明交通大學資訊工程系

Digital Circuit Design

- Instructor: Jen-Hui Chuang
  - E-mail: jchuang@cs.nycu.edu.tw
  - Office: EC239
- Class time & classroom: W34F7-EC016
- Office hours: W5
- Web Page: (New) E3

Digital Circuit Design

- Textbook
  - M. M. Mano and M. D. Ciletti, “Digital Design," 6th Ed., Pearson Prentice Hall, 2019



- Grade
  - Homework/Quiz: 30%
  - Mid-terms: 35%
  - Final: 35%

Digital Circuit Design

1. Digital Systems and Binary Numbers
2. Boolean Algebra and Logic Gates
3. Gate-Level Minimization
4. Combinational Logic
5. Synchronous Sequential Logic
6. Registers and Counters
7. Memory and Programmable Logic

Chapter 1

Digital Systems and Binary Numbers

Outline

- 1.1 Digital Systems
- 1.2 Binary Numbers
- 1.3 Number-Base Conversions
- 1.4 Octal and Hexadecimal Numbers
- 1.5 Complements of Numbers
- 1.6 Signed Binary Numbers
- 1.7 Binary Codes
- 1.8 Binary Storage and Registers
- 1.9 Binary Logic

1.1 Digital Systems

- Digital age
- Digital computers
  - general purposes
  - many scientific, industrial, commercial applications
- Digital systems
  - digital telephone / TV / camera
  - digital versatile disk (DVD)
  - Smartphone
  - Robotic/UAV/AGV
  - XR/MR/AR/VR

1.1 Digital Systems

- Reasons of using digital circuits
  - Programmable
  - Low cost
  - High speed
  - Reliable
  - Small size
  - Low power

1.2 Binary Numbers

- Decimal number

$$\ldots a_5 a_4 a_3 a_2 a_1 a_0 . a_{-1} a_{-2} a_{-3} \ldots$$

Decimal point

$$10^5 a_5 + 10^4 a_4 + 10^3 a_3 + 10^2 a_2 + 10^1 a_1 + 10^0 a_0 + 10^{-1} a_{-1} + 10^{-2} a_{-2} + 10^{-3} a_{-3}$$

Ex. $7,329 = 7 \times 10^3 + 3 \times 10^2 + 2 \times 10^1 + 9 \times 10^0$

- General form of base-*r* system

$$a_n \cdot r^n + a_{n-1} \cdot r^{n-1} + \cdots + a_2 \cdot r^2 + a_1 \cdot r^1 + a_0 + a_{-1} \cdot r^{-1} + a_{-2} \cdot r^{-2} + \cdots + a_{-m} \cdot r^{-m}$$

Coefficient: $0 \le a_j \le r - 1$

1.2 Binary Numbers

- Ex. Base-2 number

$$(11010.11)_2 = (26.75)_{10}$$
$$= 1\times 2^4 + 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 0\times 2^0 + 1\times 2^{-1} + 1\times 2^{-2}$$

- Ex. Base-8 number

$$(127.4)_8$$
$$= 1\times 8^2 + 2\times 8^1 + 7\times 8^0 + 4\times 8^{-1} = (87.5)_{10}$$

- Ex. Base-16 number

$$(B65F)_{16} = 11\times 16^3 + 6\times 16^2 + 5\times 16^1 + 15\times 16^0 = (46,687)_{10}$$

1.2 Binary Numbers

- Ex. Conversion from base-2 number to decimal

$$(110101)_2 = 32 + 16 + 4 + 1 = (53)_{10}$$

Table 1.1
Powers of Two

n	2^n	n	2^n	n	2^n
0	1	8	256	16	65,536
1	2	9	512	17	131,072
2	4	10	1,024	18	262,144
3	8	11	2,048	19	524,288
4	16	12	4,096	20	1,048,576
5	32	13	8,192	21	2,097,152
6	64	14	16,384	22	4,194,304
7	128	15	32,768	23	8,388,608

1.2 Binary Numbers

- Special powers of 2
  - 2^{10} (1024) is referred to as K (Kilo)
  - 2^{20} is referred to as M (Mega)
  - 2^{30} is referred to as G (Giga)
  - 2^{40} is referred to as T (Tera)
  - 2^{50} is referred to as P (Peta)

1.2 Binary Numbers

- Arithmetic operations with numbers in base *r* follow the same rules as decimal numbers
  - Addition

Augend:	101101
Addend:	+100111
Sum:	1010100

  - Subtraction

Minuend:	101101
Subtrahend:	−100111
Difference:	000110

  - Multiplication

Multiplicand	**1011**
Multiplier	**× 101**
Partial Products	**1011**
	0000 -
	1011 - -
Product	**110111**

1.3 Number-Base Conversions

Name	Radix	Digits
Binary	2	0,1
Octal	8	0,1,2,3,4,5,6,7
Decimal	10	0,1,2,3,4,5,6,7,8,9
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

- The six letters (in addition to the 10 integers) in hexadecimal represent: 10, 11, 12, 13, 14, and 15, respectively

1.3 Number-Base Conversions

Ex. 1.1 Convert decimal 41 to binary

divide by 2

Integer	Remainder
41	
20	1
10	0
5	0
2	1
1	0
0	1

$$(41)_{10} = (a_5 a_4 a_3 a_2 a_1 a_0)_2 = (101001)_2$$

1.3 Number-Base Conversions

Ex. 1.2 Convert decimal 153 to octal.

divide by 8

Integer	Remainder
153	
19	1
2	3
0	2

$= (231)_8$

1.3 Number-Base Conversions

Ex. 1.3 Convert $(0.6875)_{10}$ to binary

	Integer		Fraction	Coefficient
$0.6875 \times 2 =$	1	+	0.3750	$a_{-1} = 1$
$0.3750 \times 2 =$	0	+	0.7500	$a_{-2} = 0$
$0.7500 \times 2 =$	1	+	0.5000	$a_{-3} = 1$
$0.5000 \times 2 =$	1	+	0.0000	$a_{-4} = 1$

$$\boxed{(0.6875)_{10} = (0.a_{-1}a_{-2}a_{-3}a_{-4})_2 = (0.1011)_2}$$

- To convert a decimal fraction to a number expressed in base *r*, a similar procedure is used

1.3 Number-Base Conversions

Ex. 1.4 Convert $(0.513)_{10}$ to octal

$$\boxed{\begin{aligned} 0.513 \times 8 &= 4.104 \\ 0.104 \times 8 &= 0.832 \\ 0.832 \times 8 &= 6.656 \\ 0.656 \times 8 &= 5.248 \\ 0.248 \times 8 &= 1.984 \\ 0.984 \times 8 &= 7.872 \end{aligned}} \Rightarrow \boxed{(0.513)_{10} = (0.406517\ldots)_8}$$

- From Examples 1.1 and 1.3: $(41.6875)_{10} = (101001.1011)_2$
- From Examples 1.2 and 1.4: $(153.\underline{513})_{10} = (231.\underline{406517\ldots})_8$

1.4 Octal and Hexadecimal Numbers

Table 1.2
Numbers with Different Bases

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	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

1.4 Octal and Hexadecimal Numbers

- Conversion from binary to octal can be done by positioning the binary number into groups of three digits each, starting from the binary point and proceeding to the left and to the right

(10	110	001	101	011	·	111	100	000	110) $_2$	= $(26153.7406)_8$
2	6	1	5	3		7	4	0	6	

- Conversion from binary to hexadecimal is similar, except that the binary number is divided into groups of four digits:

(10	1100	0110	1011	·	1111	0010) $_2$	= $(2C6B.F2)_{16}$
2	C	6	B		F	2	

1.4 Octal and Hexadecimal Numbers

- Conversion from octal or hexadecimal to binary is done by reversing the preceding procedure

$(673.124)_8 = (110$	111	011	$\cdot$	001	010	$100)_2$
6	7	3		1	2	4

$(306.D)_{16} = (0011$	0000	0110	$\cdot$	$1101)_2$
3	0	6		D

1.5 Complements of Numbers

- Diminished Radix Complement
  - Example:

The 9's complement of 546700 is 999999 – 546700 = 453299.

The 9's complement of 012398 is 999999 – 012398 = 987601.

  - Example:

The 1's complement of 1011000 is 0100111

The 1's complement of 0101101 is 1010010

1.5 Complements of Numbers

- Radix Complement
  - Example:

The 10's complement of 012398 is 987602
The 10's complement of 246700 is 753300

  - Example:

The 2's complement of 1101100 is 0010100
The 2's complement of 0110111 is 1001001

1.5 Complements of Numbers

- Subtraction with Complements
  - The subtraction of two *n*-digit unsigned numbers $M - N$ in base r can be done as follows:

1. Add the minuend M to the r's complement of the subtrahend N. Mathematically, $M + (r^n - N) = M - N + r^n$.
2. If $M \geqq N$, the sum will produce and end carry r^n, which can be discarded; what is left is the result $M - N$.
3. If $M < N$, the sum does not produce an end carry and is equal to $r^n - (N - M)$, which is the r's complement of $(N - M)$. To obtain the answer in a familiar form, take the r's complement of the sum and place a negative sign in front.

1.5 Complements of Numbers

Ex. 1.5 Using 10's complement, subtract 72532 – 3250

$M =$	72532
10's complement of $N =$	$+96750$
Sum $=$	169282
Discard end carry $10^5 =$	-100000
Answer $=$	69282

$\leftarrow 100000 - 3250$

1.5 Complements of Numbers

Ex. 1.6 Using 10's complement, subtract 3250 – 72532

	M =	03250
10's complement of	N =	+ 27468
	Sum =	30718

← 100000 – 72532

no end carry!

Therefore, the answer is – (10's complement of 30718) = – 69282

30718 – 100000

1.5 Complements of Numbers

Ex. 1.7 Using 2's complement to perform (a) X – Y and (b) Y– X, for X = 1010100 and Y = 1000011

$$
\begin{array}{lrr}
\text{(a)} & X = & 1010100 \\
 & \text{2's complement of } Y = & \underline{+0111101} \\
 & \text{Sum} = & 10010001 \\
 & \text{Discard end carry } 2^7 = & \underline{-10000000} \\
 & \text{Answer. } X - Y = & 0010001
\end{array}
$$

$\leftarrow$ 10000000 – 1000011

$$
\begin{array}{lrr}
\text{(b)} & Y = & 1000011 \\
 & \text{2's complement of } X = & +\,\underline{0101100} \\
 & \text{Sum} = & 1101111
\end{array}
$$

$\leftarrow$ 10000000 – 1010100

110111 – 10000000

no end carry!

Therefore, the answer is – (2's complement of 1101111) = – 0010001

1.5 Complements of Numbers

Ex. 1.8 Using 1's complement, repeat **Ex. 1.7**

(a) $X - Y = 1010100 - 1000011$

$X =$	1010100
1's complement of $Y =$	+ 0111100
Sum =	10010000
End-around carry =	+ 1
Answer. $X - Y =$	0010001

$\leftarrow 1111111 - 1000011$

$\leftarrow 10010000 - 1111111$

(b) $Y - X = 1000011 - 1010100$

$Y =$	1000011
1's complement of $X =$	+ 0101011
Sum =	1101110

$\leftarrow 1111111 - 1010100$

$1101110 - 1111111$

no end carry!

Therefore, the answer is – (1's complement of 1101110) = – 0010001

1.6 Signed Binary Numbers

- To represent negative integers, we need a notation for negative values
- It is customary to represent the sign with a bit placed in the leftmost position of the number
- The convention is to make the sign bit 0 for positive and 1 for negative
- Example:

Signed-magnitude representation:	10001001
Signed-1's-complement representation:	11110110
Signed-2's-complement representation:	11110111

1.6 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	—	—

1.6 Signed Binary Numbers

- Arithmetic Addition
  - The addition of two numbers in the signed-magnitude
    - same signs: add the two magnitudes, use the common sign
    - different signs: subtract the smaller magnitude from the larger one, use the sign of the larger magnitude.

    (normally not used in computer arithmetic)
  - The addition of two signed binary numbers:
    - represent negative numbers in signed-2's-complement form
    - add the two numbers, including their sign bits.
    - a carry out of the sign-bit position is discarded.

    (will only deal with this representation …)

1.6 Signed Binary Numbers

- Example:

$+\ 6$	00000110	-6	11111010
$+13$	00001101	$+13$	00001101
$+19$	00010011	$+7$	00000111
$+\ 6$	00000110	-6	11111010
-13	11110011	-13	11110011
-7	11111001	-19	11101101

1.6 Signed Binary Numbers

- Arithmetic Subtraction
  - Take the 2's complement of the subtrahend (including sign bit) and add it to the minuend (including sign bit)

$$(\pm A) - (+B) = (\pm A) + (-B)$$
$$(\pm A) - (-B) = (\pm A) + (+B)$$

  - Example

(–6) – (–13) ⟹ (11111010 – 11110011)

⟹ (11111010 + 00001101)

⟹ 00000111 (+7)

  - A carry out of sign-bit position is discarded

1.6 Signed Binary Numbers

- Example (overflow & underflow):

$+6$	00110	-6	11010
$\underline{+13}$	$\underline{01101}$	$\underline{+13}$	$\underline{01101}$
$+19$	10011	$+7$	00111
$+6$	00110	-6	11010
$\underline{-13}$	$\underline{10011}$	$\underline{-13}$	$\underline{10011}$
-7	11001	-19	01101

1.7 Binary Codes

- BCD Code

Table 1.4
Binary-Coded Decimal (BCD)

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

1.7 Binary Codes

- Example:
  - Decimal 185 and its corresponding values in BCD and binary:

$$(185)_{10} = (\underbrace{0001}_{1}\ \underbrace{1000}_{8}\ \underbrace{0101}_{5})_{BCD} = (10111001)_2$$

  - BCD Addition

$$\begin{array}{rrrrrrl}
4 & 0100 & 4 & 0100 & 8 & 1000 & \\
\underline{+5} & \underline{+0101} & \underline{+8} & \underline{+1000} & \underline{+9} & \underline{+1001} & \\
9 & 1001 & 12 & 1100 & 17 & 10001 & \leftarrow \text{incorrect} \\
 & \text{illegal} & & \underline{+0110} & & \underline{+0110} & \leftarrow +6 \\
 & & & 1\underbrace{0010}_{2} & & 1\underbrace{0111}_{7} &
\end{array}$$

1.7 Binary Codes

- Example:
  - Consider the addition of 184 + 576 = 760 in BCD:

BCD	1	1		
	0001	1000	0100	184
	+ 0101	0111	0110	+576
Binary sum	0111	10000	1010	
Add 6		0110	0110	
BCD sum	0111	0110	0000	760

- Decimal Arithmetic (using 10's complement)

(+375) + (–240) →

0	375
+9	760
0	135

1.7 Binary Codes

- Other Decimal Codes

Table 1.5
Four Different Binary Codes for the Decimal Digits

Decimal Digit	BCD 8421	2421	Excess-3	$\overbrace{8, 4, -2, -1}^{+/-\text{ weights}}$
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
	1010	0101	0000	0001
Unused	1011	0110	0001	0010
bit	1100	0111	0010	0011
combi-	1101	1000	1101	1100
nations	1110	1001	1110	1101
	1111	1010	1111	1110

weighted Self-complimenting

1.7 Binary Codes

- Gray Code

0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

$$g_{m-1} = a_{m-1}$$

$$g_i = a_i \oplus a_{i+1}$$

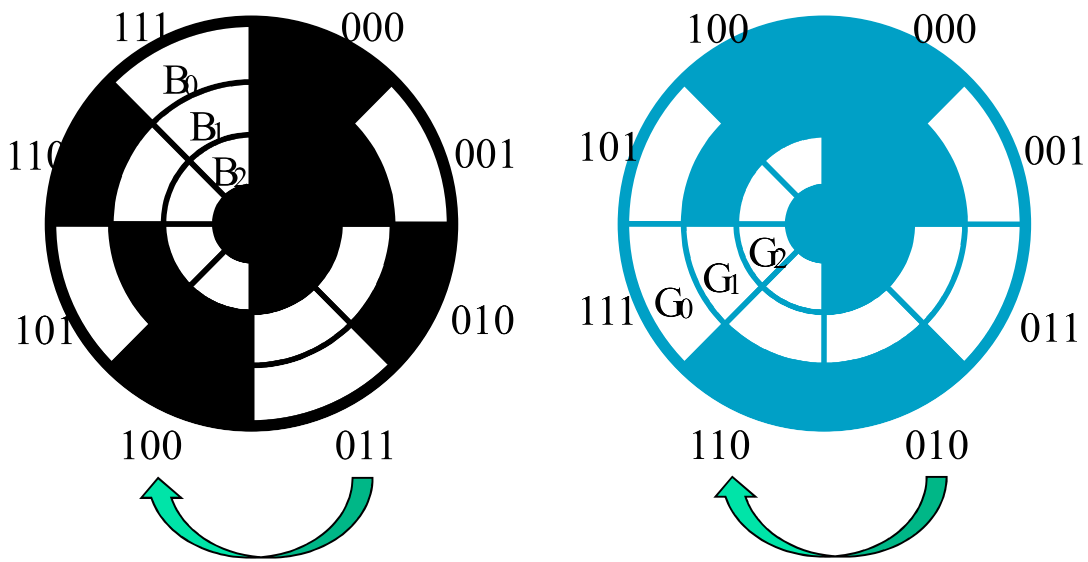
$$(0 \leq i \leq m - 2)$$

Table 1.6
Gray Code

Gray Code	Decimal Equivalent
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15

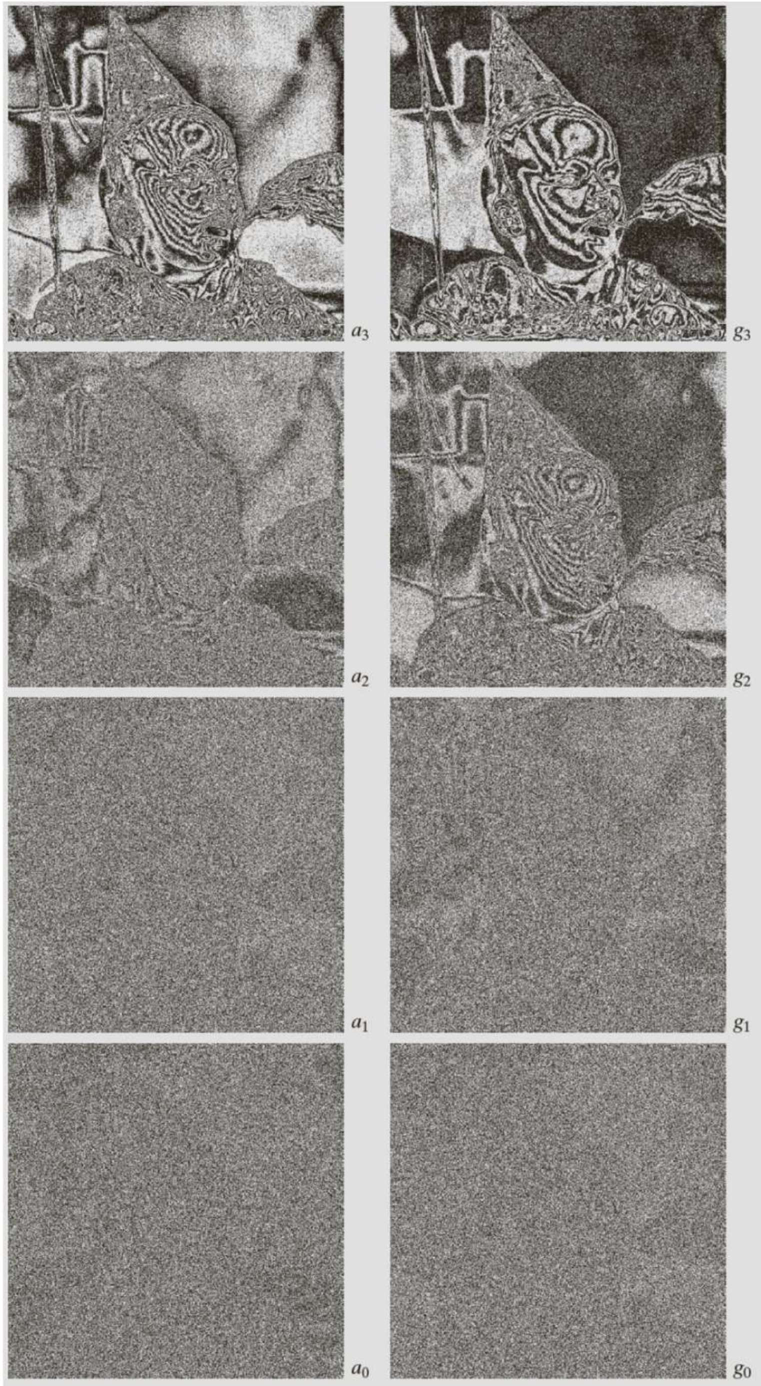
1.7 Binary Codes

- Why Gray Code?
  - One-bit change → no error/ambiguity produced
  - Example: Optical Shaft Encoder



  - In image compression: save about one bit …


All bits
$a_{7,8}$
a_6
g_6
a_5
g_5
a_4
g_4


a_3
g_3
a_2
g_2
a_1
g_1
a_0
g_0

1.7 Binary Codes

- ASCII Character Code

Table 1.7
American Standard Code for Information Interchange (ASCII)

	$b_7b_6b_5$							
$b_4b_3b_2b_1$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	–	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

1.7 Binary Codes

- ASCII Character Code

Control characters

NUL	Null	DLE	Data-link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Negative acknowledge
ACK	Acknowledge	SYN	Synchronous idle
BEL	Bell	ETB	End-of-transmission block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SO	Shift out	RS	Record separator
SI	Shift in	US	Unit separator
SP	Space	DEL	Delete

1.7 Binary Codes

- ASCII Character Code
  - American Standard Code for Information Interchange
  - A popular code used to represent information sent as character-based data.
  - It uses 7-bits to represent:
    - 94 Graphic printing characters.
    - 34 Non-printing characters
      - a. for text format (e.g. BS = Backspace, CR = carriage return)
      - b. for record marking and flow control (e.g. STX and ETX start and end text areas)

1.7 Binary Codes

- Error-Detecting Code
  - To detect errors in data communication and processing, an eighth bit is sometimes added to the ASCII character to indicate its parity
  - A parity bit is an extra bit included with a message to make the total number of 1's either even or odd
- Example:
  - Consider the following two characters and their even and odd parity:

	With even parity	With odd parity
ASCII A = 1000001	01000001	11000001
ASCII T = 1010100	11010100	01010100

1.7 Binary Codes

- Conversion or Coding?
  - 1310 = 11012 (conversion)
  - 13 ⇔ 0001|0011 (coding)

1.9 Binary Logic

Table 1.8
Truth Tables of Logical Operations

AND

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

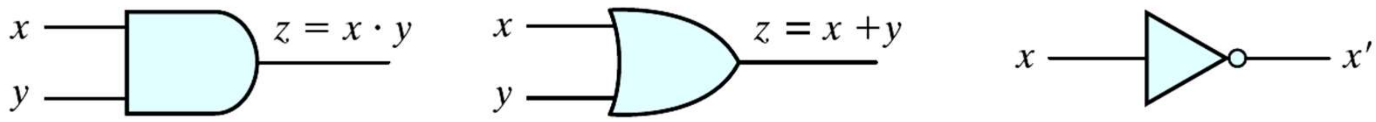
OR

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT

x	x'
0	1
1	0

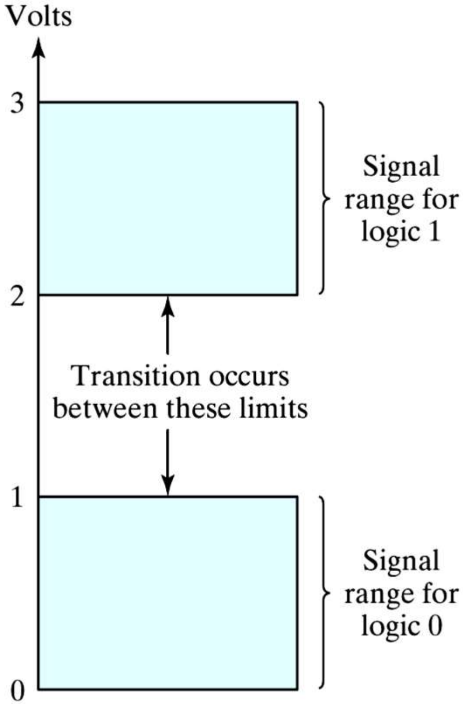
- Logic Gates (Graphic Symbols)



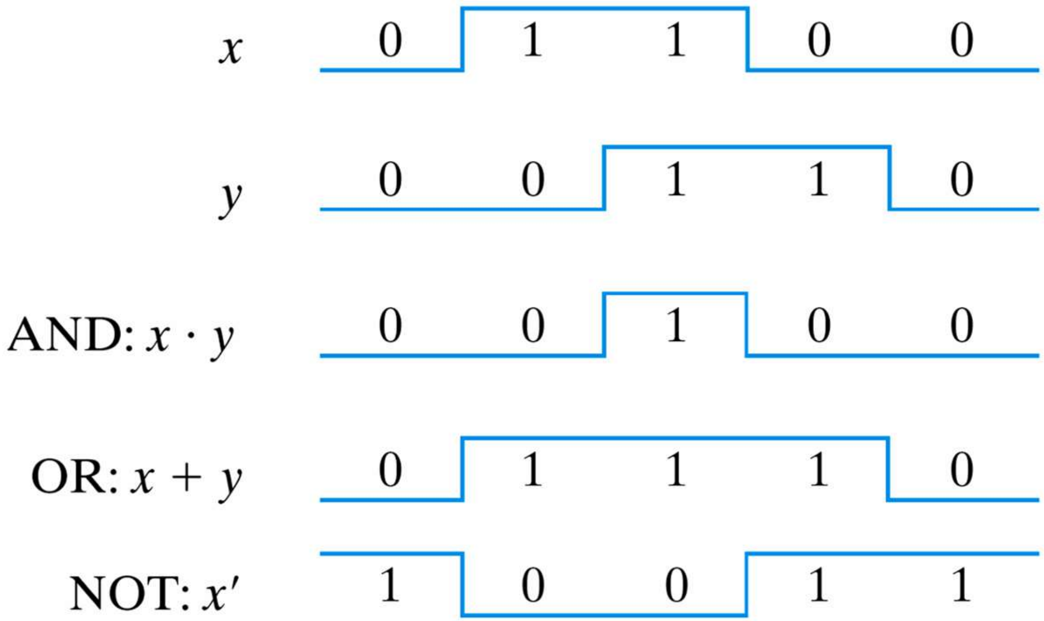
(a) Two-input AND gate (b) Two-input OR gate (c) NOT gate or inverter

1.9 Binary Logic

- Example of binary signals



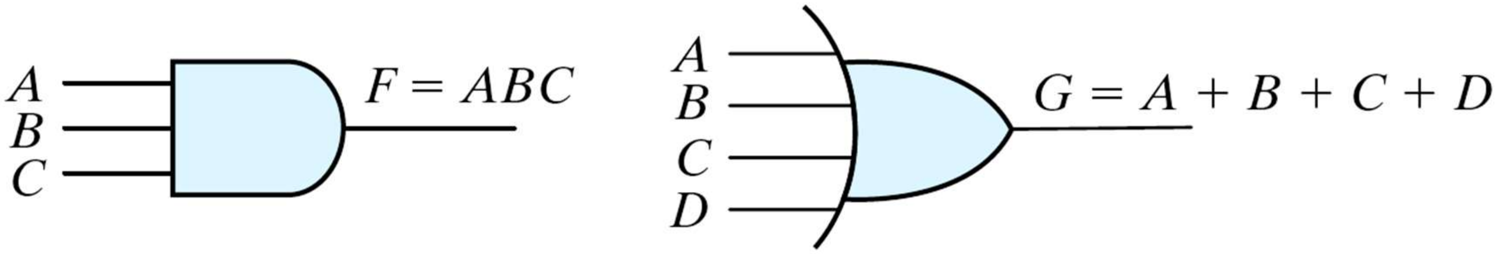
discrete value



discrete value/time

1.9 Binary Logic

- Gates with multiple inputs



(a) Three-input AND gate (b) Four-input OR gate