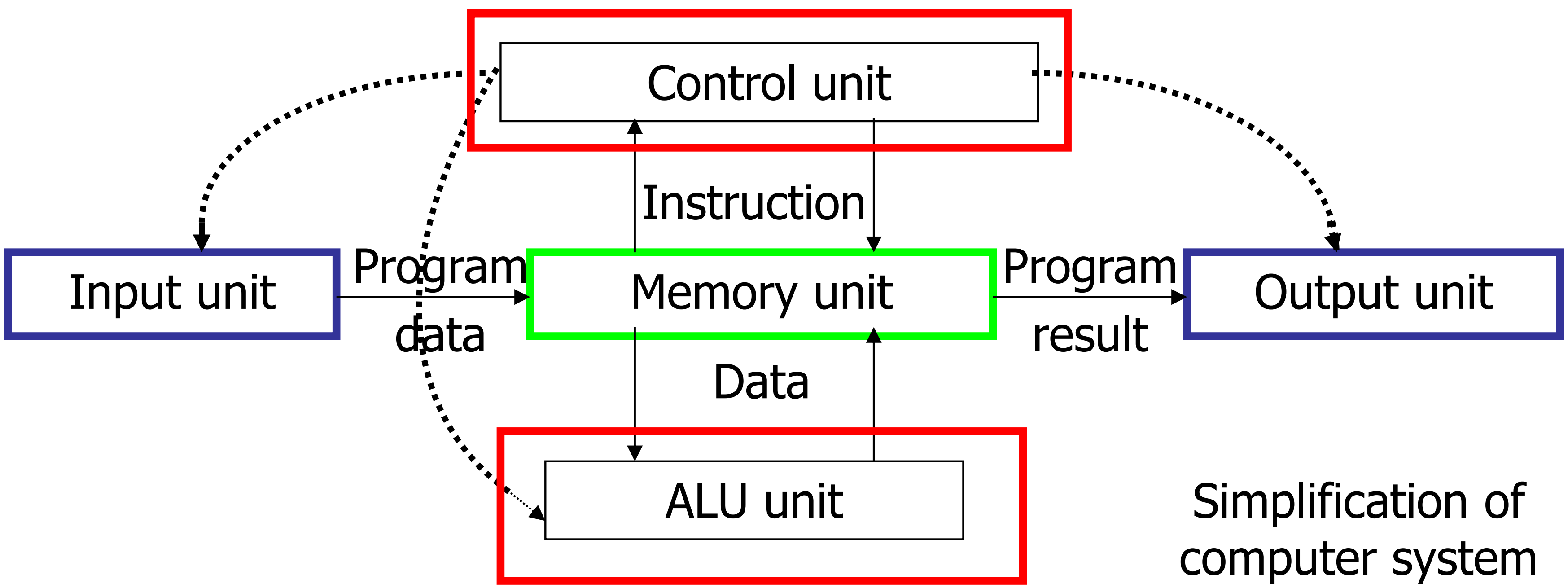
Chapter 1 Binary Systems
1-1. Digital Systems

- The General-purpose digital computer is the best-known example of a digital system.
- The major parts of a computer are a memory unit , a central processing unit, and input-output units.



Simplification of computer system

1-2. Binary Numbers

- A number with decimal point represented by a series of coefficients as follow:

  $$a_5a_4a_3a_2a_1a_0.a_{-1}a_{-2}a_{-3}$$

  the power of 10 by which the coefficient must be multiplied as following:

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

  the decimal number system is said to be of base, and the coefficients are multiplied by powers of 10.

Numbers convertion

- A number expressed in a base-r system has coefficients multiplied by powers of r

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

Coefficients a_j range in value from 0 to r-1.

Base-5 number:

$(4021.2)_5$

$= 4 \times 5^3 + 0 \times 5^2 + 2 \times 5^1 + 1 \times 5^0 + 2 \times 5^{-1} = (511.4)_{10}$

Others base-r number can be converted into decimal by this way.

Numbers conversion

Binary convert into decimal:

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

The number behind equal sign obtained as following table

Table 1-1
Powers of Two

n	2n	n	2n	n	2n
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

Other operations

- Examples of addition, subtraction, and multiplication of two binary numbers are as follows:

Augend: 101101	minuend: 101101	multiplicand: 101
Addend: +100111	subtrahend: -100111	multiplier: X101
Sum: 1010100	difference: 000110	101
	011001	000
		101
		product: 11001

Find 2's complement then add with minuend(section 1-5)

1-3. Number base conversions

- Ex1-1:Convert decimal 41 to binary

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

Answer=101001

The conversion from decimal integers to any base-r system is similar to the example, see the Ex1-2.

Number base conversions

- Ex1-3:Convert $(0.6875)_{10}$to octal

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

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

To convert a decimal fraction in base-r, a similar procedure is used. Combining the answer from Ex1-1 and Ex1-3

$$(41.6875)_{10} = (101001.1011)_2$$

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

$(10\ \ 110\ \ 001\ \ 101\ \ 011\ .\ 111\ \ 100\ \ 000\ \ 110)_2 = (26153.7460)_8$

$(10\ \ 1100\ \ 0110\ \ 1011\ .\ \ 1111\ \ 0010)_2 = (2C6B.F2)_{16}$

$(673.124)_8 = (110\ \ 111\ \ 011\ \ .\ \ 001\ \ 010\ \ 100)_2$

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

1-5. Complements

- Complements are used for simplifying the subtraction operation and for logical manipulation.
- There are two types of complements for each base-r system: the radix and the diminished radix complements.
- Binary numbers: 2's complement

  1's complement

  Decimal numbers: 10's complement

  9's complement

Diminished radix complement

- Given a number N in base-r having n digits, the (r-1)'s complement of N is defined as $(r^n-1)-N$.

Decimal numbers: 012398 have 6 digits and present below

$$(10^6 - 1) – 012398 = 999999 – 012398 = 987601$$

Binary numbers: $1011000=(88)_{10}$

$$(2^7 - 1) – 1011000 = 1111111 – 1011000 = 0100111(39)$$

shortcut(1<-->0) 0100111

Radix complement

- The r's complement of an n-digit number in base-r is defined as $r^n - N$, for N=0 and 0 for N=0.
- Compare with (r – 1)'s complement, the r's complement is (r – 1)'s + 1 since

  $r^n - N = [(r^n - 1) - N] + 1.$

Decimal number: 012398

$10^6 - 012398 = 987602 = 999999 - 012398 + 1$

Binary number: 1011000(88)

$2^7 - 1011000 = 0101000(38) = 1111111 - 1011000 + 1$

Leaving all least significant 0's and the first 1 unchanged, and others have complemented

Or 1011000

complemented unchanged

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. This performs $M + (r^n - N) = M - N + r^n$.
2. If $M \geq N$, sum will produce an end carry, r^n, which can be discarded; the result is $M - N$.
3. If $M < N$, the sum does not produce an end carry and is equal to $r^n - (N - M)$. Take the r's complement of the sum and place a negative sign in front.

Examples

Ex1-6: 3250 – 72532 using 10's complement

M = 03250

10's complement of N = + 27468

Sum= 30718 → no end carry

The answer is −(10's complement of 30718) = −69282

Ex1-7: X=1010100, Y=1000011 using 2'scomplement

(b) Y = 1000011

2's complement of X = +0101100

Sum = 1101111 → no end carry

The answer is Y−X =−(2's complement of 1101111)=−0010001

Examples

- We can also use (r – 1)'s complement, the sum is 1 less than the correct difference when an end carry occurs. Removing the end carry and adding 1 to the sum is referred to as an *end-around carry*.

Ex1-8: Repeat Ex1-7 using 1's complement

(a) X = 1010100

1's complement of Y = + 0111100

Sum = $\boxed{1}$0010000

End around carry = + 1

Answer : X – Y = 0010001

1-6. Signed binary numbers

- The convention is to make the sign bit 0 for positive and 1 for negative in Signed binary numbers.
- In signed binary, the leftmost bit represents the sign and the rest of the bits represent the number.
- In unsigned binary, the leftmost bit is the most significant bit(MSB).
- In 2's complement could represent one more than negative number because of no negative zero.

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	—	—

Identical

MSB is 1 to distinguish them from the positive numbers

Arithmetic of subtraction and addition

- We have discussed at section 1-5 and have following conclusion:

2's complement:

1. Have an end carry, discard then get answer
2. No end carry, find 2's complement and add minus sign front the answer

1's complement:

1. Have an end around carry, add this bit then get answer
2. No end around carry, find 1's complement and add minus sign front the answer

1-7. Binary codes
BCD code

- We are more accustomed to the decimal system, and is straight binary assignment as listed in Table1-4. this is called binary coded decimal(BCD).
- 1010~1111 are not used and have no meaning in BCD code.

Ex: $(185)_{10} = (1011001)_2$

$= (0001\ 1000\ 0101)_{BCD}$

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

BCD Addition

- When the binary sum is greater than or equal to 1010, the addition of 6 to the binary sum converts it to the correct digit and also produces a carry as required.

One digit addition:

1000	8
+1001	+9
10001	17
+0110	
1 0111	

two digits addition:

BCD carry	1	1			
	0001	1000	0100		184
	+0101	0111	0110		+576
Binary sum	0111	10000	1010	>9	
Add 6		0110	0110	+6	
BCD sum	0111	0110	0000		760

Other Decimal Codes

- The BCD,84-2-1, and the 2421 codes are examples of weighted codes.
- The 2421 and the excess-3 codes are examples of self-complementing codes.

Ex. $(395)_{10} = (0110\ 1100\ 1000)_{excess\text{-}3}$

9's complement self-complementing

$(604)_{10} = (1001\ 0011\ 0111)_{excess\text{-}3}$

it is obviously to know the self-complementing that the excess-3 code of 9's complement of 395 is complementing the excess-3 of 395 directly. So does the 2421 code.

Other Decimal Codes

Table1-5

Four Different Binary Codes for the Decimal Digits

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

2 → $8 \times \boxed{0} + 4 \times \boxed{1} + (-2) \times \boxed{1} + (-1) \times \boxed{0} = 2$

weight

7 → $\boxed{2} \times \boxed{1} + \boxed{4} \times \boxed{1} + \boxed{2} \times \boxed{0} + \boxed{1} \times \boxed{1} = 7$

Gray Code

- The advantage of the Gray code over the straight binary number sequence is that only one bit in the code group changes when one number to the next.

EX: from 7 to 8

Gray code changes from

0100 to 1100.

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

$(0\ 1\ 1\ 1)_2$

xor

$(01\ 0\ \ 0)_{Gray}$

xor

$(01\ 1\ 1)_2$

ASCII Character Code

- ASCII include seven bits, contain 94 graphic characters and 34 control functions as follows table.
- There are three types of control characters:

1. format effectors: control the layout of printing (BS, HT, CR…).
2. information separators: used to separate the data into divisions such as paragraphs and pages (RS, FS…).
3. communication-control characters: it is useful during the transmission of text between remote terminals (STX, ETX..).

ASCII Table

Table 1-7
American Standard Code for Information Interchange (ASCII)

	$b_7 b_6 b_5$							
$b_4 b_3 b_2 b_1$	**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

Control characters

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

Error Detecting Code

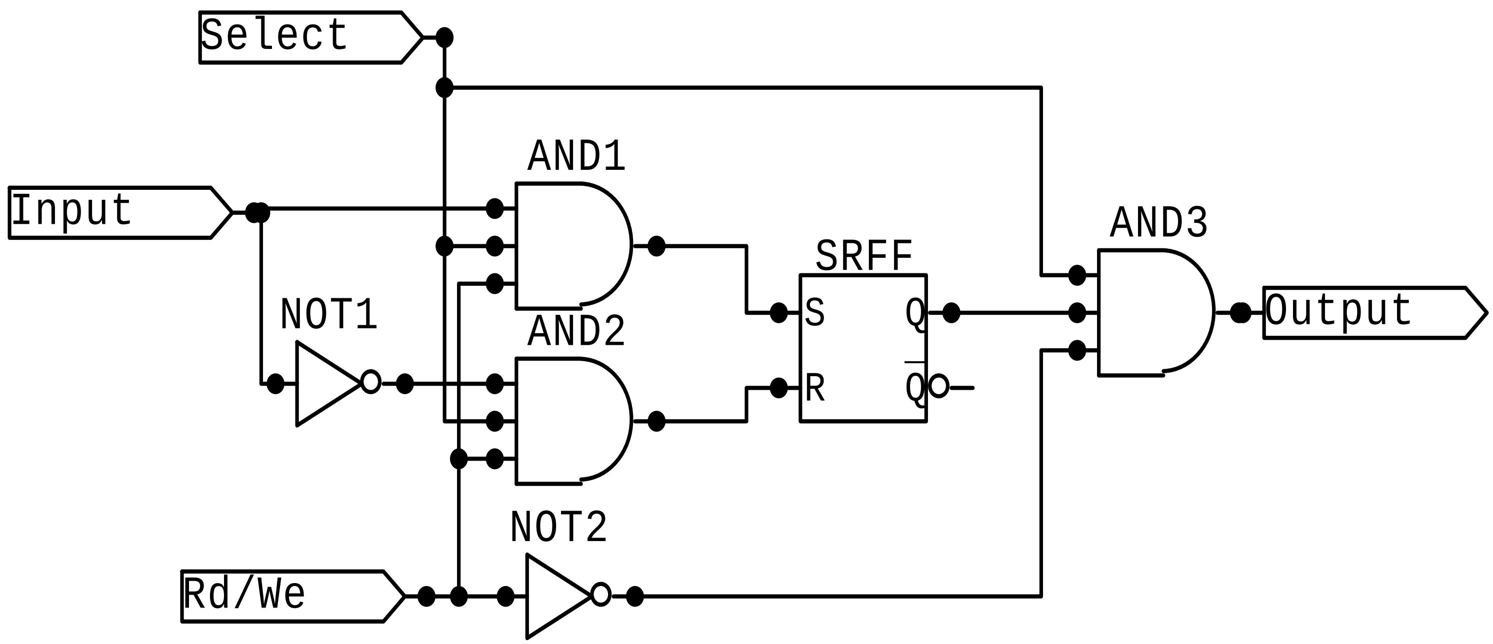
- An eighth bit is added to the ASCII character to indicate its parity. We have following even and odd parity:

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

- The even or odd parities can find out only odd combination of errors in each character, an even combination of errors is undetected. May be the hamming code can solve that in some bits range.

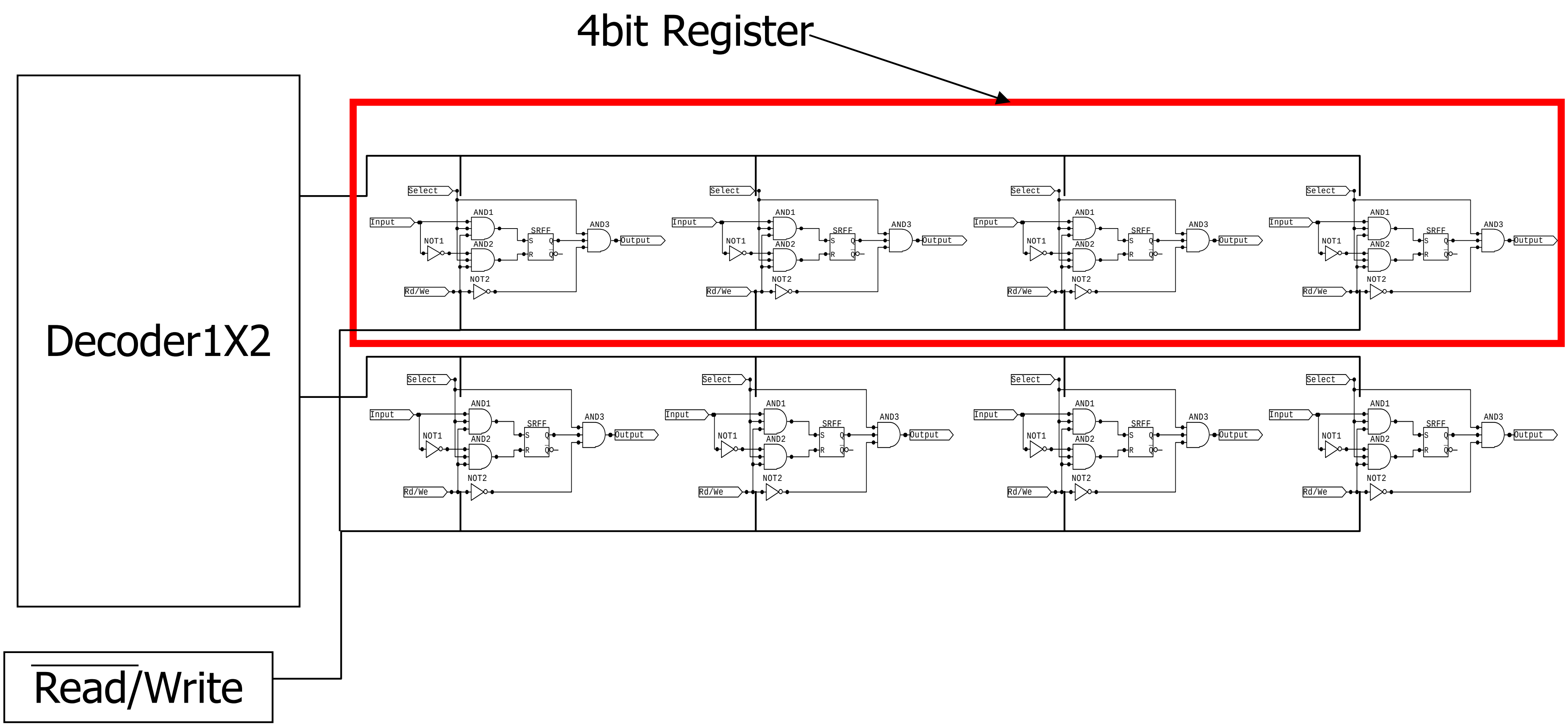
1-8. Binary storage and registers

- A binary cell is a device that processes two stable states and is capable of storing one bit of info, and a register is a group of binary cells.



Bit cell

Register



4bit Register
Decoder1X2
Read/Write
Select
Input
Rd/We
AND1
AND2
AND3
NOT1
NOT2
SRFF
Output

Register Transfer

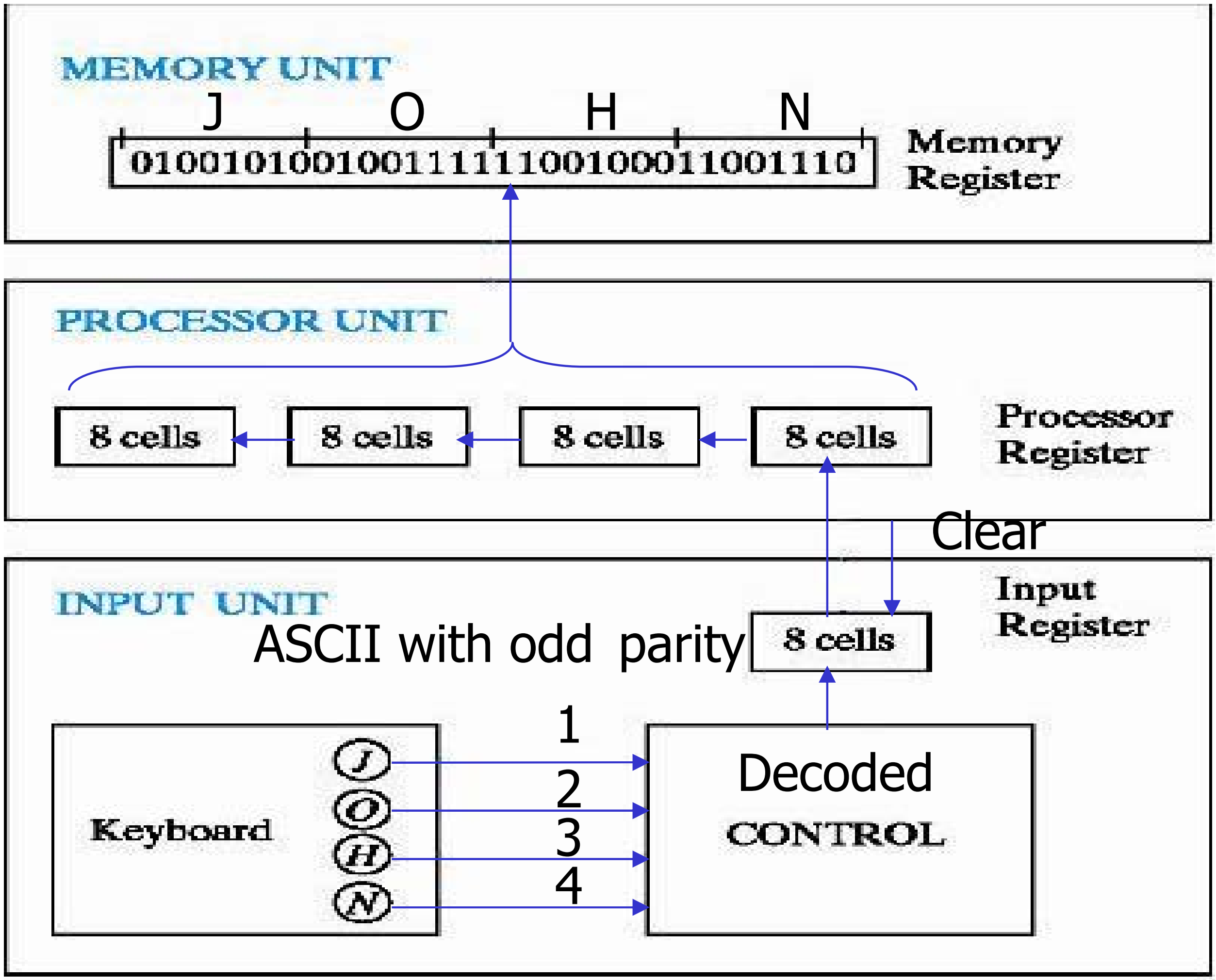


Fig. 1-1 Transfer of information with registers

Binary information processing

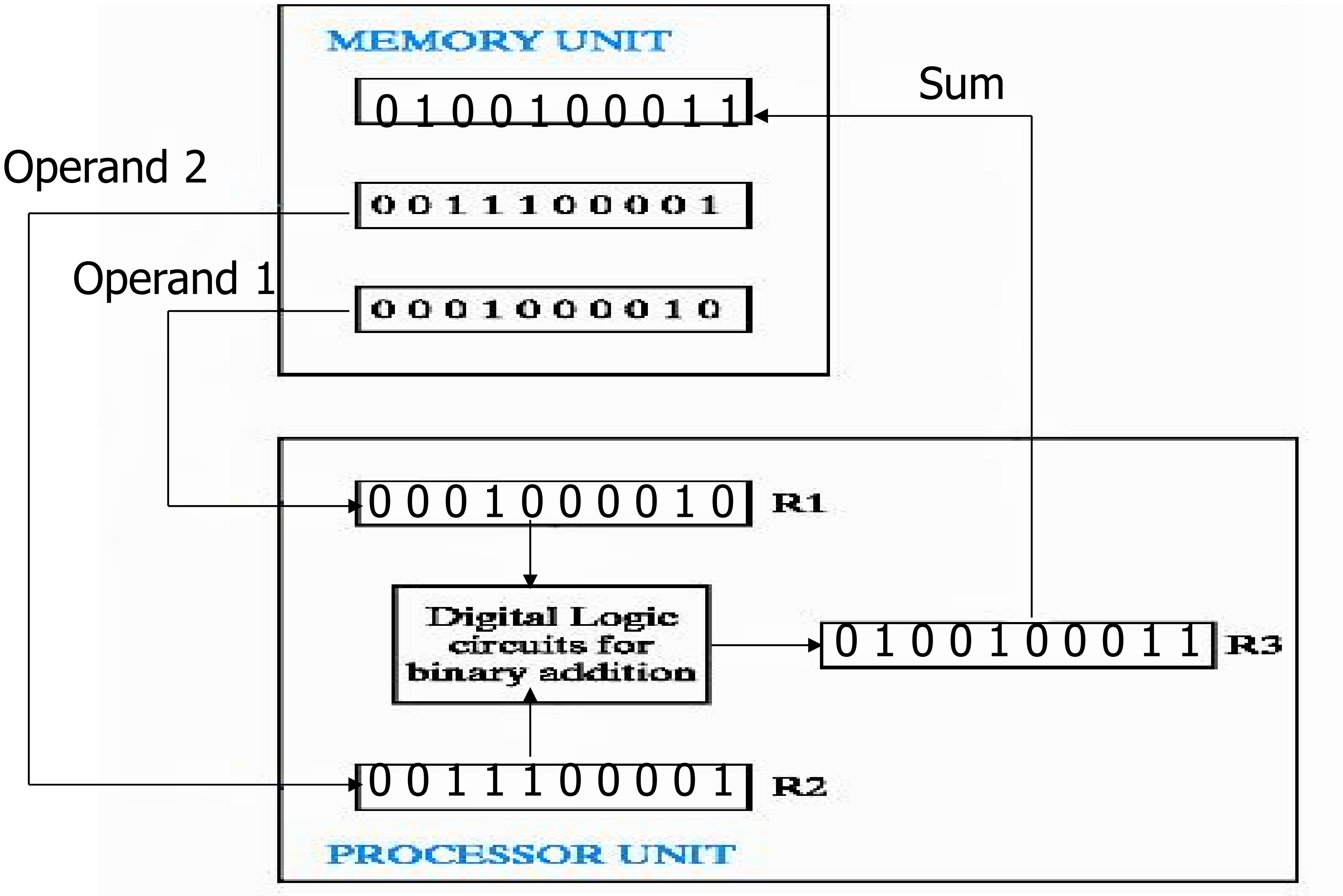


Fig. 1-2 Example of binary information processing

1-9. Binary Logic

- There are three basic logical operations:

1. AND: This operation is represented as follows

$$x \cdot y = z \text{ or } x\,y = z,$$

z=1 if and only if x=1 and y=1; otherwise z=0

2. OR: This operation is represented as follows

$$x + y = z$$

z=1 if x=1 or if y=1 or x=1 and y=1 ;otherwise z=0

3. NOT: This operation is represented as follows

$$x' = z \text{ or } \overline{x} = z$$

e.g. complement operation, changes a 1 to 0, 0 to 1

Similar and difference

- Binary logic resembles binary arithmetic, and the operations AND and OR have similarities to multiplication and addition, respectively.
- The symbols used for AND and OR are the same as those used for multiplication and addition.
- Binary logic should not be confused with binary arithmetic.

Binary arithmetic: $1 + 1 = 10_2 = (2)_{10}$

Binary logic: $1 + 1 = 1_2$

Logic Gates

- For each combination of the values of x and y, and output z, it may be listed in a compact form using truth tables.

Table 1-8
Truth Tables of Logical Operations

AND			OR			NOT	
x	y	$x \cdot y$	x	y	$x + y$	x	x'
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

Definition of the logic signals

- Logic gates are electronic circuits that operate on one or more input signals to produce an output signal.
- Signals such as voltages or currents, we define between some ranges as logic 1 or logic 0.

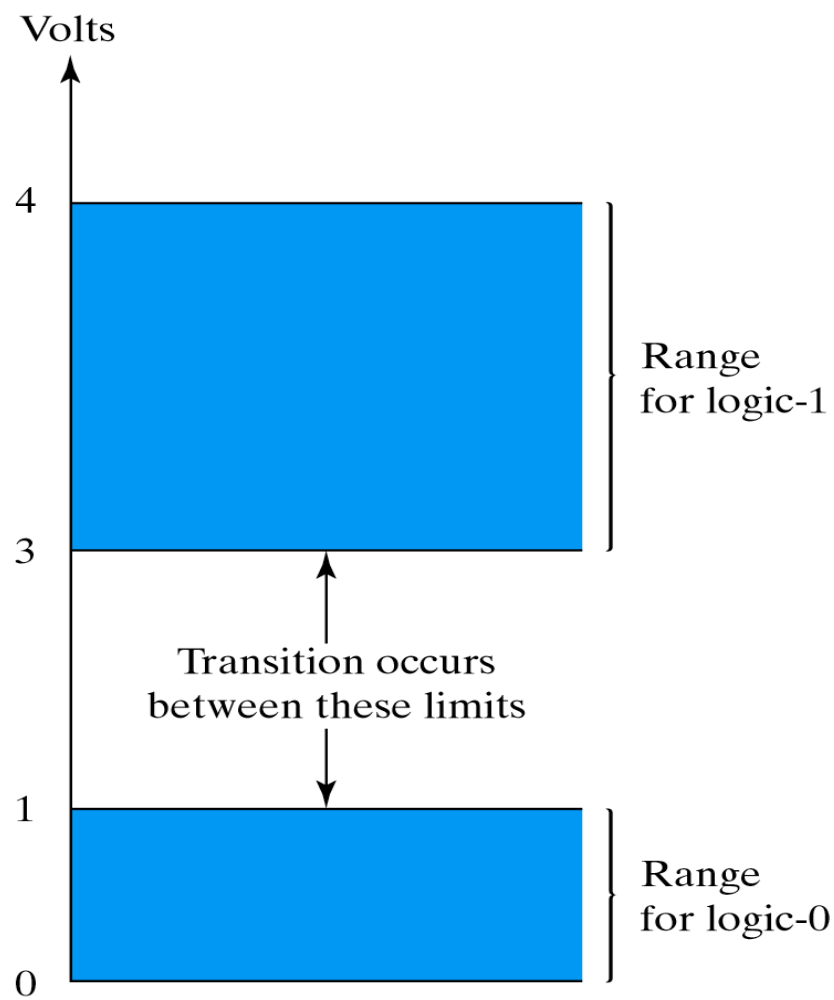
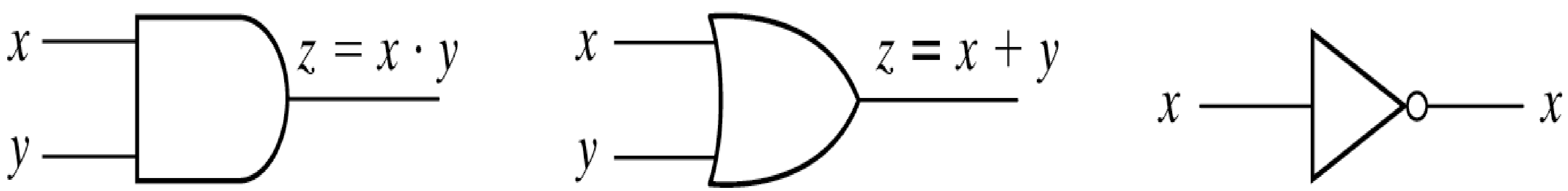


Fig. 1-3 Example of binary signals

The symbol of logic gates

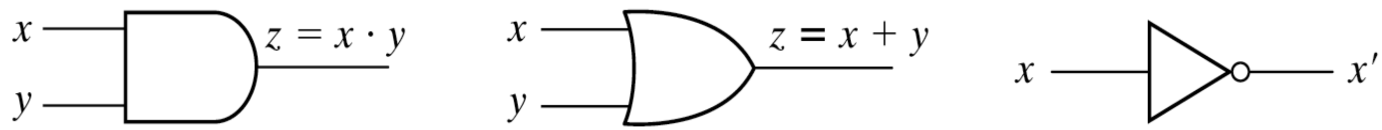
- The graphic symbols used to designate the three types of gates are shown below.



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

Fig. 1-4 Symbols for digital logic circuits

Timing diagrams



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

Fig. 1-4 Symbols for digital logic circuits

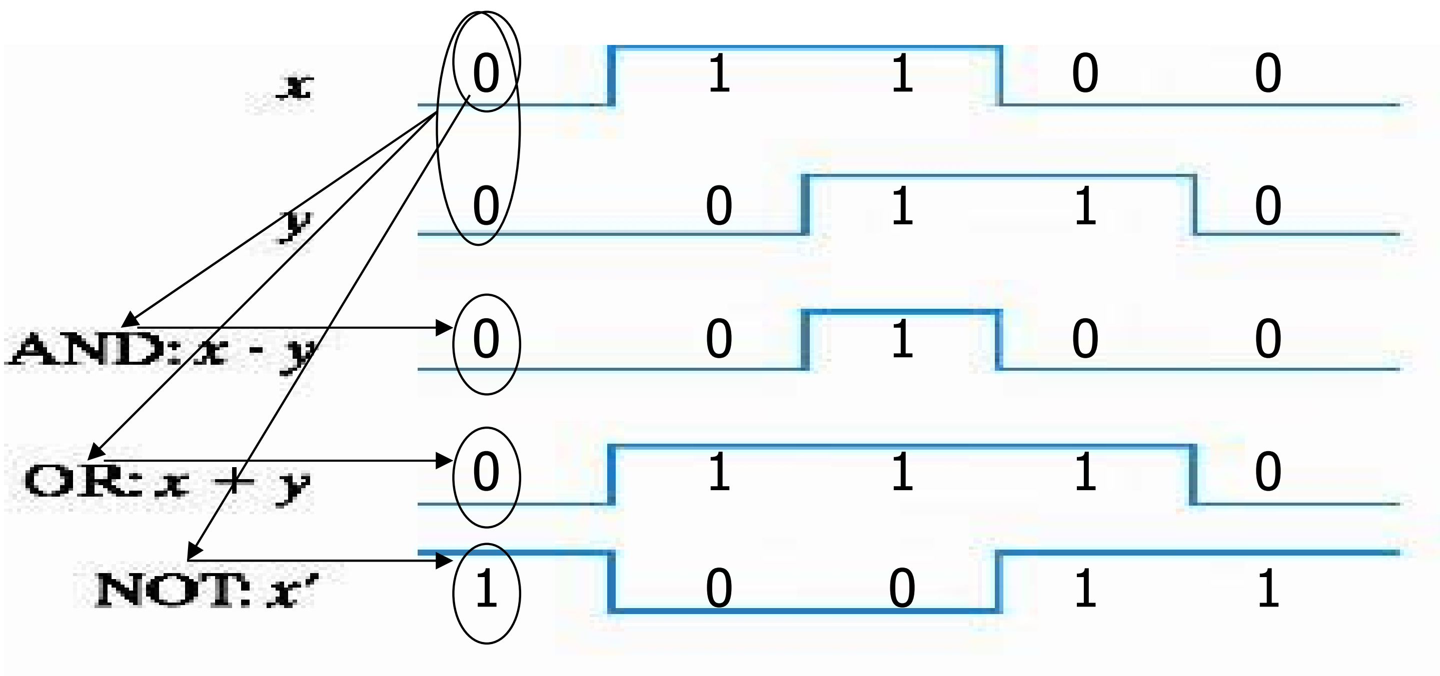
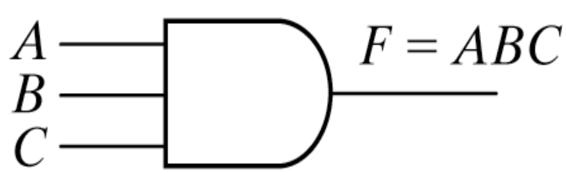


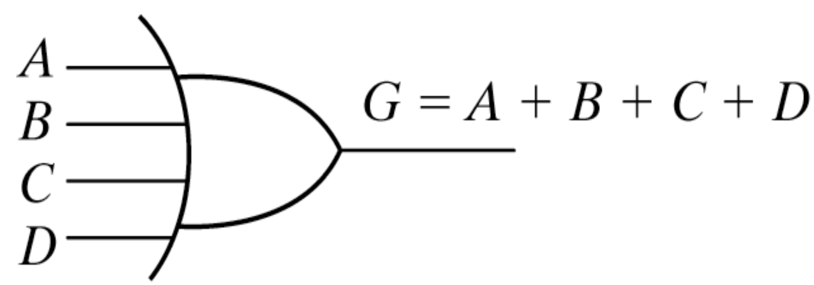
Fig. 1-5 Input-output signals for gates

Gates with multiple inputs

- AND and OR gates may have more than two inputs.
- Three –input AND gate responds with logic 1 output if all three inputs are logic 1. when any input is logic 0, output produces logic 0.
- OR gate characteristic have described before in this chapter.



(a) Three-input AND gate



(b) Four-input OR gate

Fig. 1-6 Gates with multiple inputs