Chapter1: Digital Systems and Binary Numbers

Lecture6- Study ASCII Code, Error Detection and Correction Codes, and Binary Logic

Engr. Arshad Nazir, Asst Prof
Dept of Electrical Engineering
SEECS

Objectives

- Study ASCII Code
- Error Detection and Correction Codes
- Binary Logic

ASCII Character Code

- The American Standard Code for Information Interchange (ASCII) uses seven bits to code 128 characters.
- The ASCII code contains 94 graphic characters that are printable and 34 nonprinting characters used for various control functions. The graphic characters consist of the 26 uppercase letters (A through Z), the 26 lowercase letters (a through z), the 10 numerals (0 through 9), and 32 special characters, such as %, *, and $.
- The 34 control characters are designated in the ASCII table with abbreviated names. They are listed again in the table with their function names. The control characters are used for routing data and arranging the printed text into a prescribed format.
  - The following ASCII chart allows you to specify the characters in decimal representation by concatenating the column headings to the row headings. For example, the character 5 is represented in binary as 0110101

ASCII Table

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	NULL	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

ASCII Table (Cont …

Control Characters

NULL	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

- Error-Detecting uses an eighth bit (added to 7-bit ASCII character) to indicate parity.
  - A parity bit is an extra bit that is set to 0 or 1 as needed to make the total number of 1's either even or odd.
  - In an odd-parity code, the parity bit is specified so that the total number of ones is odd.
  - In an even-parity code, the parity bit is specified so that the total number of ones is even.
  - It detects one, three or any odd combination of errors but even combination of errors is undetected.


```
1 1 0 0 0 0 1 1
↑
Added even parity bit
```

```
0 1 0 0 0 0 1 1
↑
Added odd parity bit
```

Parity Code Example

- Concatenate a parity bit to the ASCII code for the characters 0, X, and = to produce both odd-parity and even-parity codes.

Character	ASCII	Odd-Parity ASCII	Even-Parity ASCII
0	0110000	10110000	00110000
X	1011000	01011000	11011000
=	0111100	10111100	00111100

Your Turn

- *Express your complete registration number in ASCII code using:*

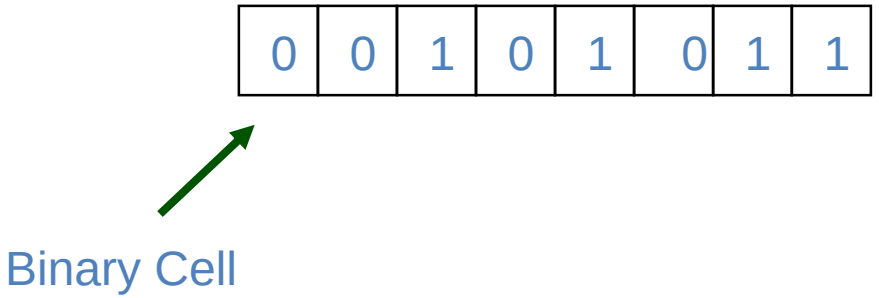
➢ Even parity

➢ Odd parity

➢ MSB set as zero.

Binary Storage

- Binary storage represents the storage mechanisms for binary data stored in a computer.
  - A binary cell is a device that possesses two stable states, and it can store a single state value (0 or 1). It stores single bit of data. examples: flip-flop circuits, ferrite cores, capacitor
  - A register is a group of binary cells. n cells allows the register to store n bits and thus 2^n possible states.
    - The type of information (BCD, ASCII, etc.) stored in a register has to be agreed upon by the users of the register.



Register Transfer

- A register transfer operation involves the transfer of binary information from one set of binary registers into other set of binary registers.
- The capture and storage of information requires:
  - An input register to store the key inputs from the keyboard
  - A processor register to store the data when processed by the CPU
  - A memory register in the memory unit to store the values

Register Transfer

- Data input at keyboard
- Shifted into place
- Stored in memory

➢ NOTE: Data input in ASCII

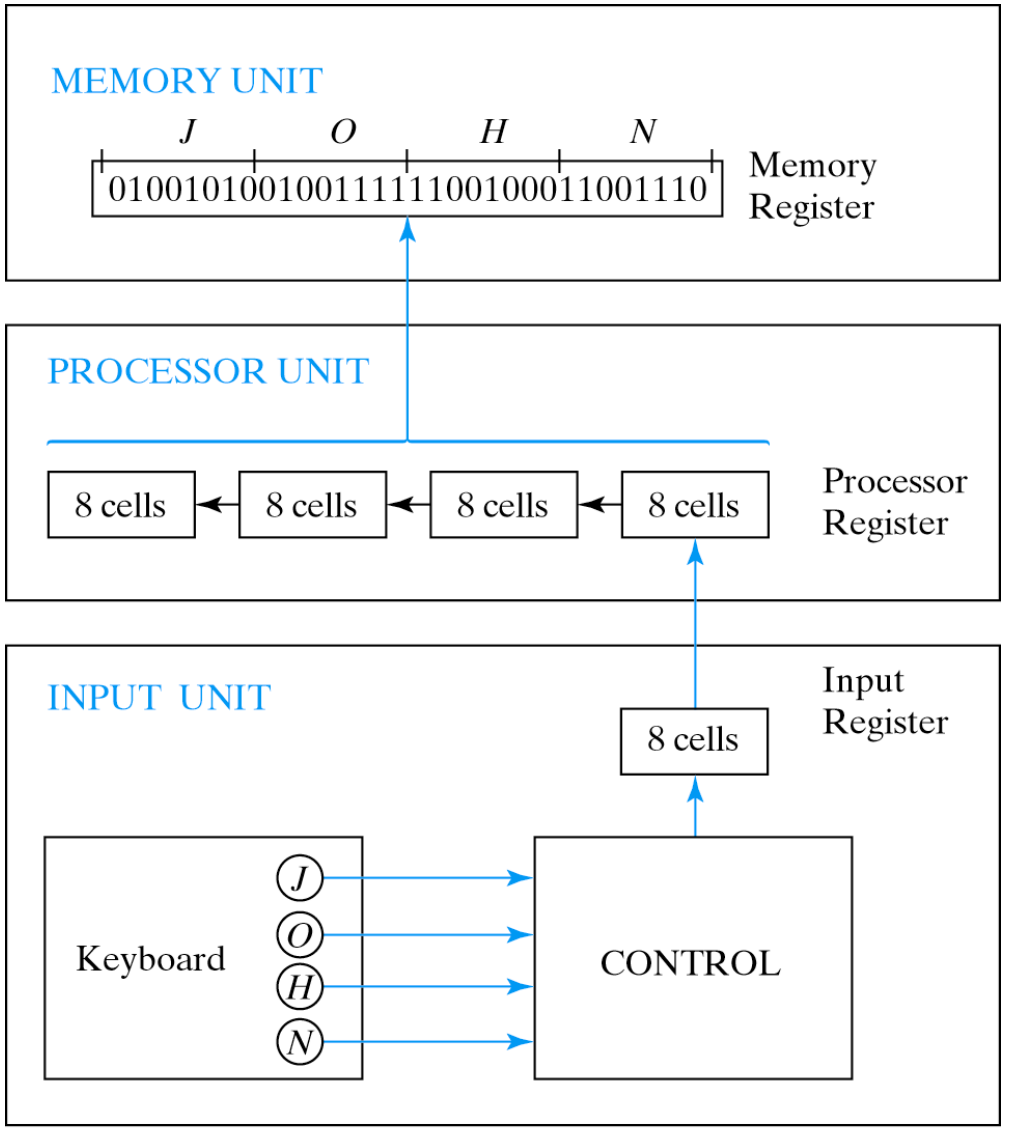


Fig. 1-1 Transfer of information with registers

Binary Information Processing

- The actual processing of binary information in a computer is completed by digital logic circuits which have been implemented to serve a specific purpose (i.e. addition).
  - The registers are accessed (read and write) when they are needed to complete an operation. For example we need two register sets to store two values to be added and a register set to store the result of the sum.
    - Furthermore, we need three registers in both the memory unit and in the processor.

Example Binary Information Processing

- We need processing
- We need storage
- We need communication

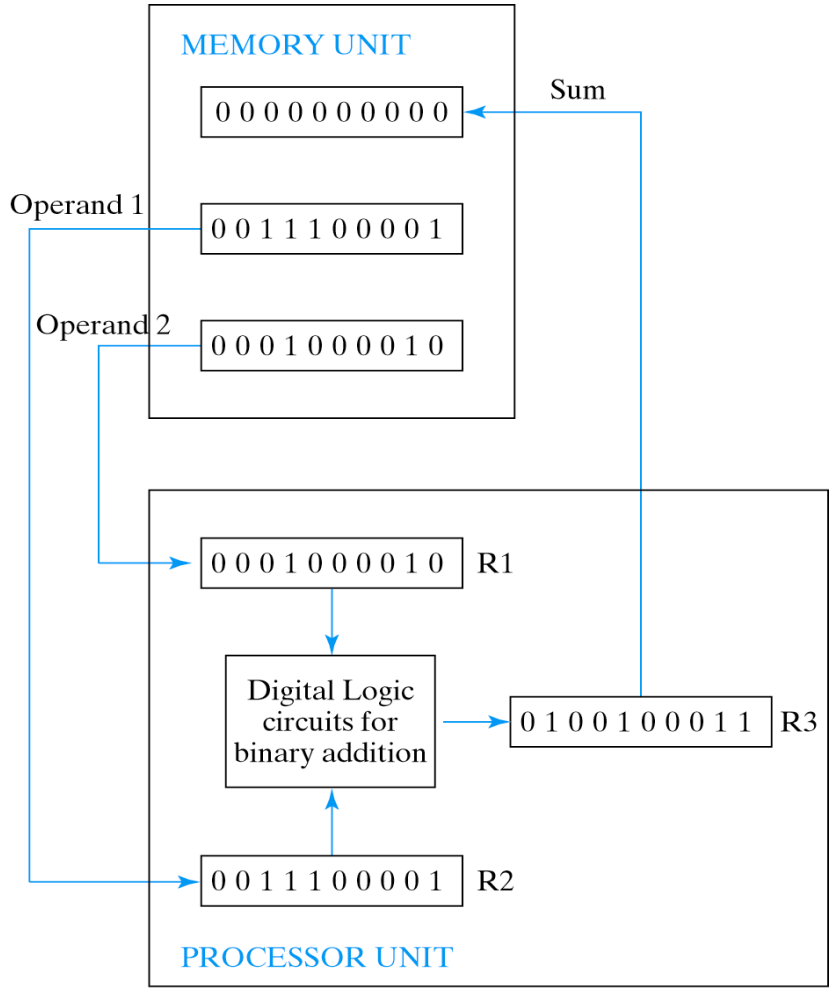


Fig. 1-2 Example of binary information processing

Binary Logic

- Binary logic consists of binary variables and logical operations.
  - The variables are designated by letters of the alphabet (A, B, C, x, y, z, etc.).
  - There are three basic logical operations:
    - AND
    - OR
    - NOT

Logical Operations

- AND is represented by a dot or the absence of an operator.
  - $x \cdot y = z$ or $xy = z$
  - Read as "x and y is equal to z"
  - Means that z=1 if and only if x=1 and y=1
- OR is represented by a plus sign.
  - $x+y = z$
  - Read as "x or y is equal to z"
  - Means that z=1 if x=1 or y=1 or both x=1 and y=1
- NOT is represented by a prime or an over bar.
  - $x' = z$ or $\bar{x} = z$
  - Read as "not x is equal to z"
  - Means that if x=1 then z=0 or if x=0 then z=1

Truth Tables

- Since each binary variable consists of value of 0 or 1, each combination of values for the variables involved in a binary operation has a specific result value.
- A truth table is a method of visualizing all possible combinations of the input values and the respective output values that occur due to the operation on the specified combination.

AND

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

OR

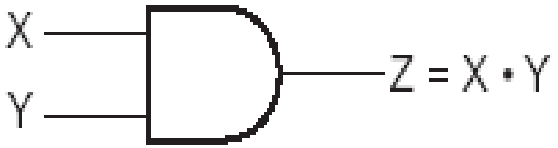
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

NOT

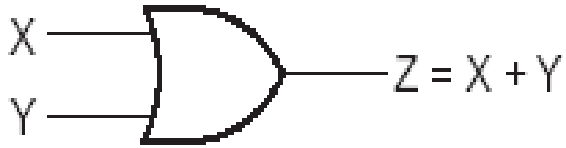
A	A′
0	1
1	0

Logic Gates

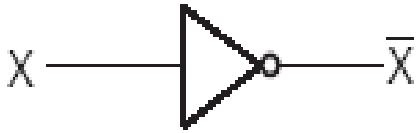
- Logic gates are electronic circuits that operate on one or more input signals to produce an output signal.
  - The state (high-low, on-off) of electricity on a line represents each of the two states for binary representation (1 or 0).
- Logic Gate Symbols



AND gate



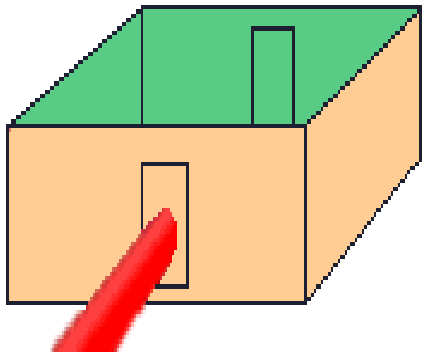
OR gate



NOT gate or inverter

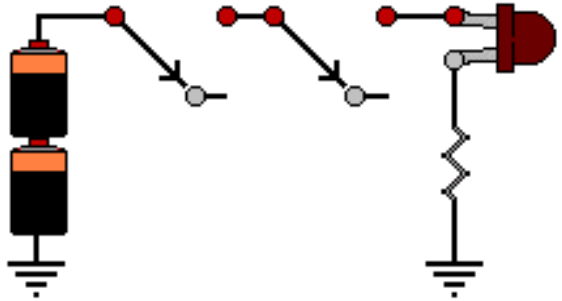
AND Logic Function

- Using door
  - Both doors are opened to pass the light



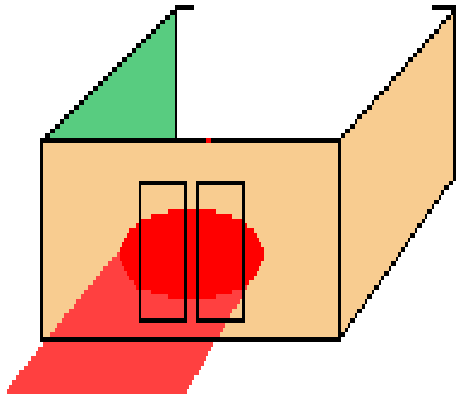
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

- Using Switches
  - Switches are input and LED is output
  - Both witches closed (ON) to give output



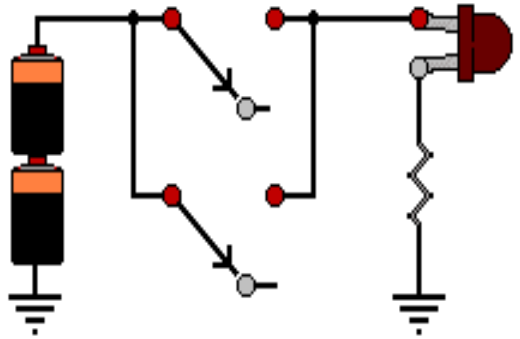
OR Logic Function

- Using door
  - Any one or both doors are opened to pass the light



A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

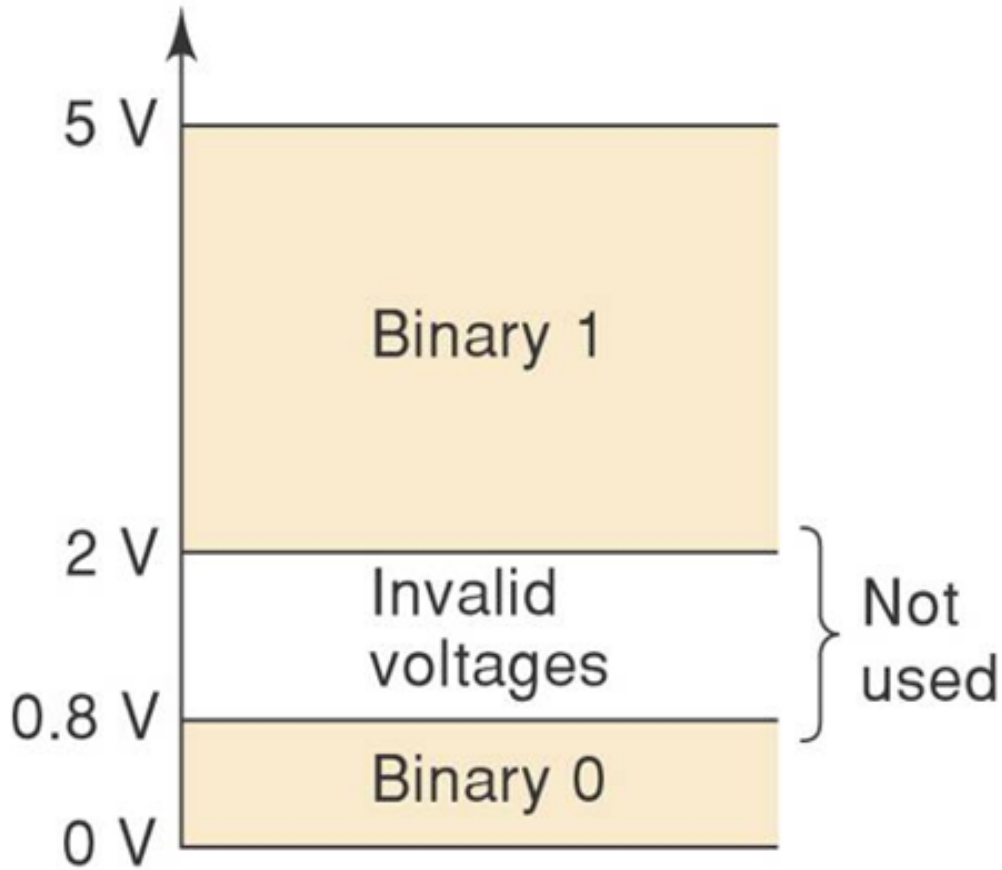
- Using Switches
  - Switches are input and LED is output
  - Any one switch or both closed to give output



Typical representation of the two states of a digital signal.

A *higher* range of voltages represent a valid 1 and a *lower* range of voltages represent a valid 0.

HIGH and LOW are often used to describe the states of a digital system—instead of "1" and "0"



Timing Diagram

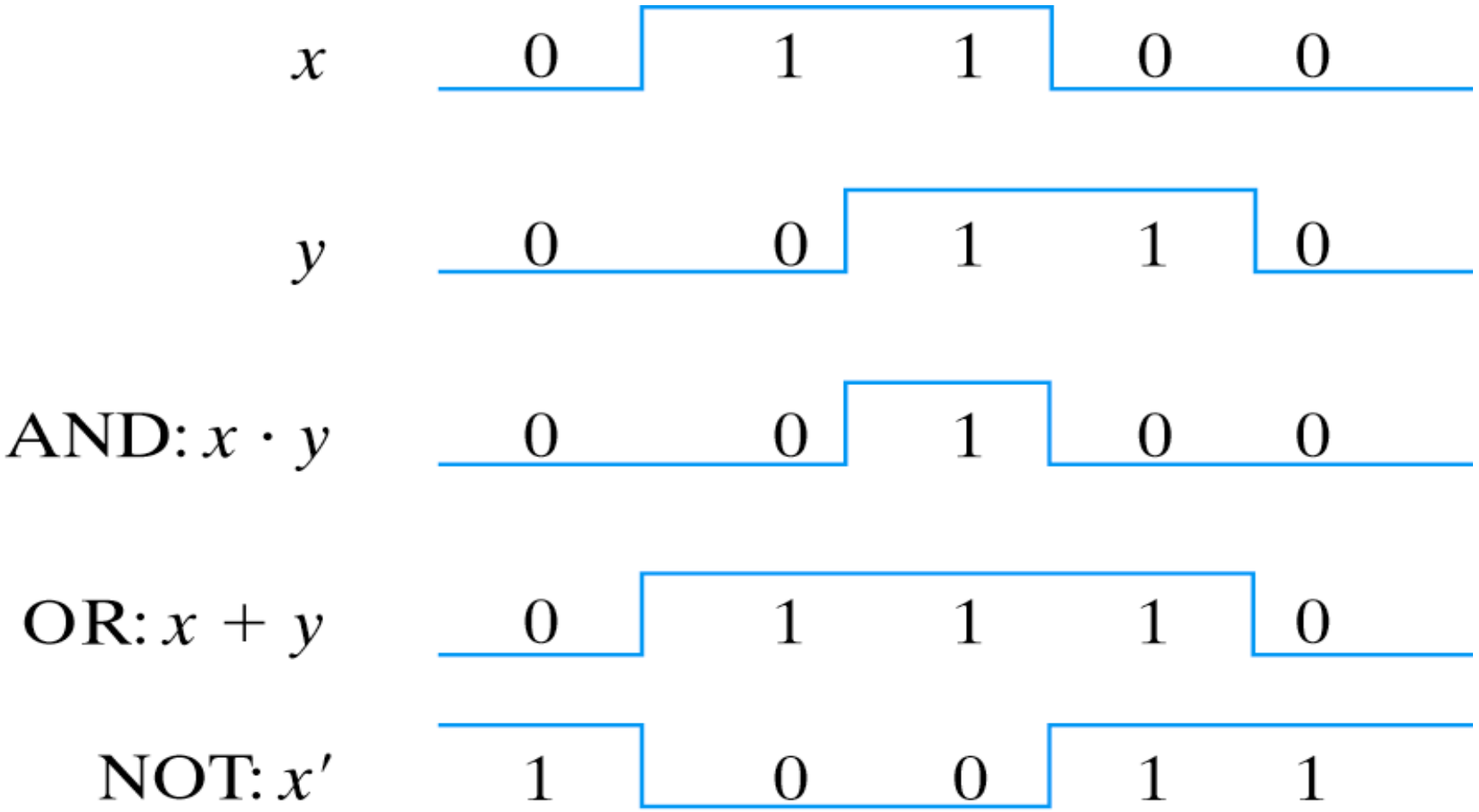
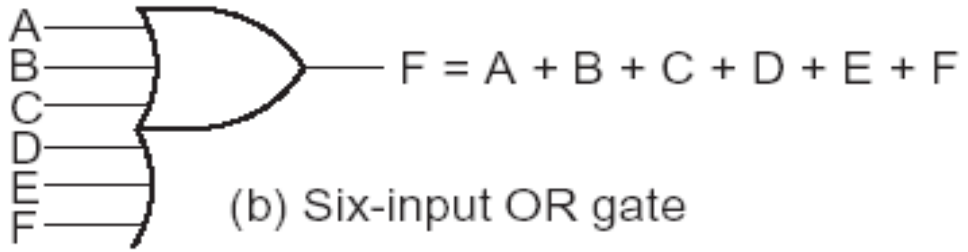


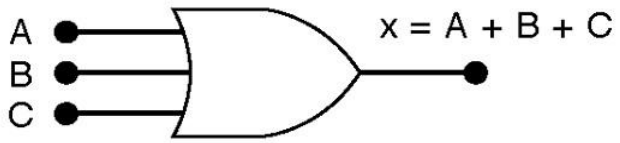
Fig. 1-5 Input-output signals for gates

Multi-Input Circuits

(a) Three-input AND gate



(b) Six-input OR gate



A	B	C	x = A + B + C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

The End