Chapter 1: Introduction

Computer organization is concerned with the way the hardware components operate and the way they are connected together to form the computer system. *Computer architecture* is concerned with the structure and behavior of the computer as seen by the user. It includes the information formats, the instruction set, and techniques for addressing memory.

Computer design is concerned with the determination of what hardware should be used and how the parts should be connected. This aspect of computer hardware is sometimes referred to as *computer implementation*.

DIGITAL COMPUTERS:

The digital computer is a digital system that performs various computational tasks. The word *digital* implies that the information in the computer is represented by variables that take a limited number of discrete values.

The decimal digits 0, 1, 2, . . . , 9, for example, provide 10 discrete values.

Digital computers use the binary number system, which has two digits: 0 and 1. A binary digit is called a *bit*. Information is represented in digital computers in groups of bits.

bits are used to develop complete sets of instructions for performing various types of computations.

In contrast to the common decimal numbers that employ the base 10 system, binary numbers use a base 2 system with two digits: 0 and 1. The decimal equivalent of a binary number can be found by expanding it into a power series with a base of 2. For example, the binary number 1001011 represents a quantity that can be converted to a decimal number by multiplying each bit by the base 2 raised to an integer power as follows:

$$1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 75$$

this same group of seven bits represents the letter K

A computer system is sometimes subdivided into two functional entities:

The hardware of the computer consists of all the electronic components and electromechanical devices that comprise the physical entity of the device.

Computer software consists of the instructions and data that the computer manipulates to perform various data-processing tasks.

A sequence of instructions for the computer is called a *program*. The data that are manipulated by the program constitute the *data base*.

The hardware of a computer is been divided into three parts:

1) Central Processing Unit (CPU)
2) Memory (RAM)
3) Input – Output Process(IOP)

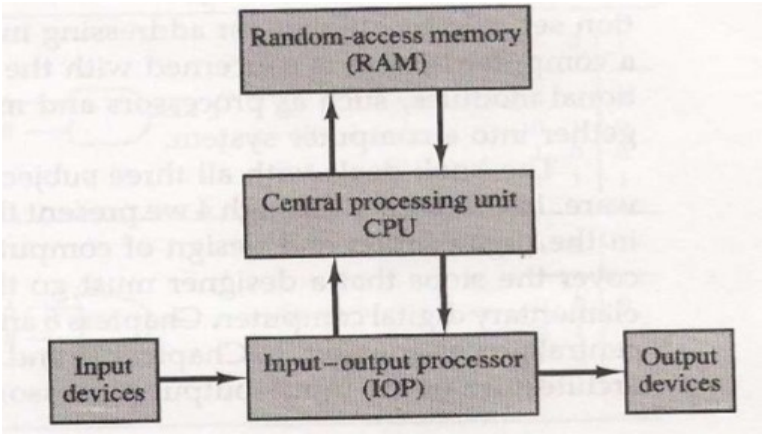


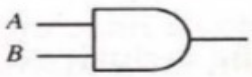
Figure 1: Block Diagram of Digital Computer

The central processing unit (CPU) contains an arithmetic and logic unit for manipulating data, a number of registers for storing data, and control circuits for fetching and executing instructions. The memory of a computer contains storage for instructions and data. It is called a random-access memory (RAM) because the CPU can access any location in memory at random and retrieve the binary information within a fixed interval of time. The input and output processor (IOP) contains electronic circuits for communicating and controlling the transfer of information between the computer and the outside world. The input and output devices connected to the computer include keyboards, printers, terminals, magnetic disk drives, and other communication devices.

Logic Gates:

Binary information is represented in digital computers by physical quantities called *signals* such as voltages The two states represent a binary variable that can be equal to 1 or 0 such as a signal of 3 volts to represent binary 1 and 0.5 volt to represent binary 0.

Binary logic deals with binary variables and with operations that assume a logical meaning. The manipulation of bi·nary information is done by logic circuits called *gates*. The input–output relationship of the binary variables for each gate can be represented in tabular form by a *truth table*.

Name	*Graphic symbol*	*Algebraic function*	*Truth table*
AND		$x = A \cdot B$ or $x = AB$	A B | x 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1

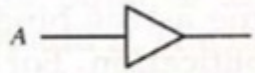
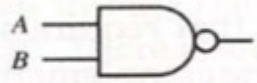
Name	Graphic symbol	Algebraic function	Truth table
OR		$x = A + B$	A B | x 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
Inverter		$x = A'$	A | x 0 | 1 1 | 0
Buffer		$x = A$	A | x 0 | 0 1 | 1
NAND		$x = (AB)'$	A B | x 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$x = (A + B)'$	A B | x 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
Exclusive-OR (XOR)		$x = A \oplus B$ or $x = A'B + AB'$	A B | x 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Exclusive-NOR or equivalence		$x = (A \oplus B)'$ or $x = A'B' + AB$	A B | x 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

Figure 2: Digital Logic Gates with Truth Table

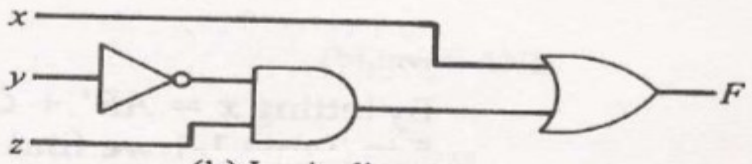
Boolean Function:

Boolean algebra is an algebra that deals with binary variables and logic operations. The variables are designated by letters such as A, B, x, and y. The three basic logic operations are AND, OR, and complement. A Boolean function can be expressed algebraically with binary variables, the logic operation symbols, parentheses, and equal sign. For a given value of the variables, the Boolean function can be either 1 or 0. for example, the Boolean function $F = x + y'z$ The function F is equal to 1 if x is 1 or if both y' and z are equal to 1; F is equal to 0 otherwise. F is equal to 1 if $x = 1$ or if $yz = 01$. it is equal to 0 for all other combinations

A Boolean function can be transformed from an algebraic expression into a logic diagram composed of AND, OR, and inverter gates. The logic diagram for F is shown in Fig 3

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(a) Truth table



(b) Logic diagram

Figure 3 Truth table and logic diagram for $F = x + y'z$.

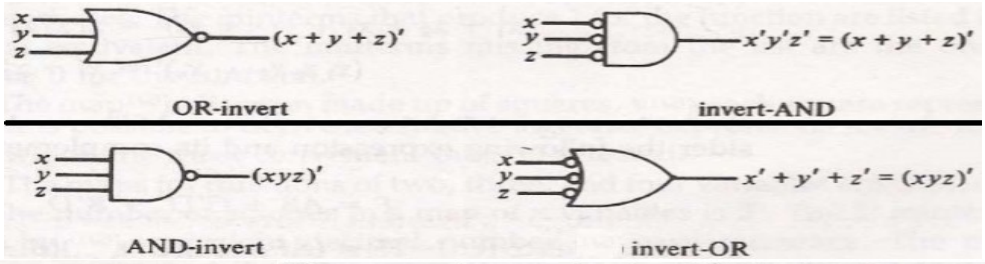
The purpose of Boolean algebra is to facilitate the analysis and design of digital circuits. It provides a convenient tool to:

1. Express in algebraic form a truth table relationship between binary variables.
2. Express in algebraic form the input–output relationship of logic diagrams.
3. Find simpler circuits for the same function.

Table 1 lists the most basic identities of Boolean algebra. All the identities in the table can be proven by means of truth tables. Identities 15 and 16 are called DeMorgan's theorems

TABLE 1 Basic Identities of Boolean Algebra

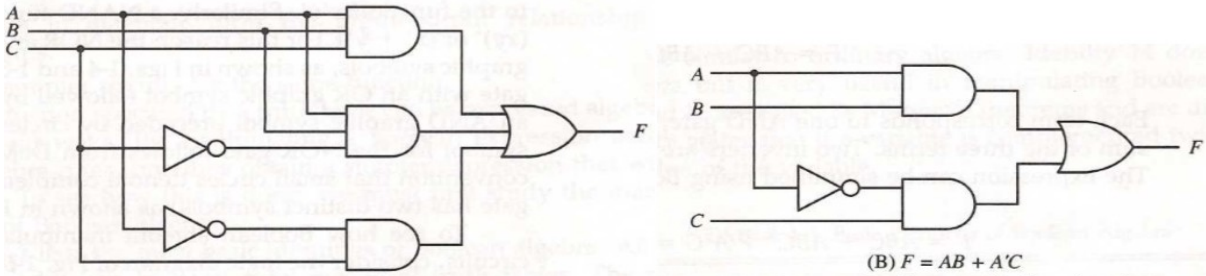
(1) $x + 0 = x$	(2) $x \cdot 0 = 0$
(3) $x + 1 = 1$	(4) $x \cdot 1 = x$
(5) $x + x = x$	(6) $x \cdot x = x$
(7) $x + x' = 1$	(8) $x \cdot x' = 0$
(9) $x + y = y + x$	(10) $xy = yx$
(11) $x + (y + z) = (x + y) + z$	(12) $x(yz) = (xy)z$
(13) $x(y + z) = xy + xz$	(14) $x + yx = (x + y)(x + z)$
(15) $(x + y)' = x'y'$	(16) $(xy)' = x' + y'$
(17) $(x')' = x$	



$F = ABC + ABC' + A'C$ The expression can be simplified using Boolean algebra

$F = ABC + ABC' + A'C = AB(C + C') + A'C \quad = AB + A'C$

Note that $(C + C)' = 1$ by identity 7 and $AB \cdot 1 = AB$ by identity 4



(a) $F = ABC + ABC' + A'C$

(B) $F = AB + A'C$

Two logic diagrams for the same Boolean function.

Complement of a Function The complement of a function F when expressed in algebraic form, the complement of the function can be derived by means of DeMorgan's theorem expressed as follows:

$$(x_1 + x_2 + x_3 + \cdots + x_n)' = x_1' x_2' x_3' \cdots x_n'$$

$$(x_1 x_2 x_3 \cdots x_n)' = x_1' + x_2' + x_3' + \cdots + x_n'$$

Map Specifications:

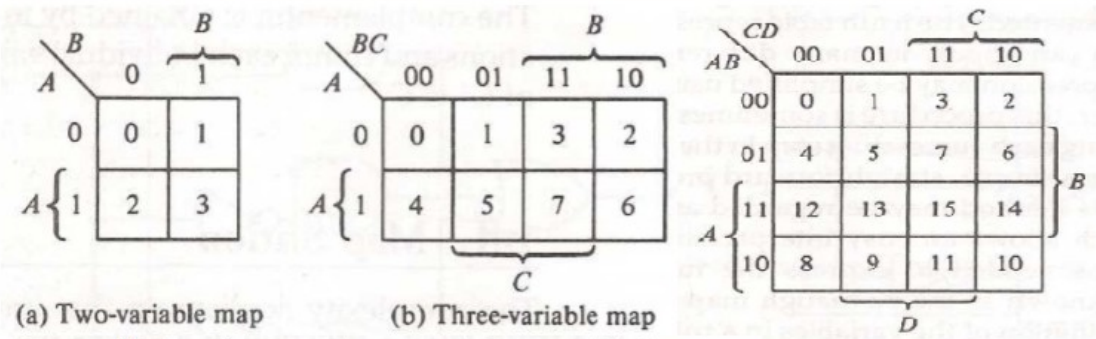
The complexity of the logic diagram that implements a Boolean function is related directly to the complexity of the algebraic expression from which the function is implemented. The truth table representation of a function is unique,

The expression may be simplified using basic relation of Boolean algebra but it is difficult to predict the successive steps. There is a method known as K- map or Karnaugh map which is simple and straightforward. It is pictorial representation of truth table which allows selecting minimum number of terms needed.

Each combination of the variables in a truth table is called a minterm. the truth table of Fig -3 contains eight minterms. in a truth table a function of n variables will have 2^n minterms, the truth table of Fig. 3 can be expressed as follows:

$F(x, y, z) = \sum (1, 4, 5, 6, 7)$ The information contained in a truth table may be expressed in compact form by listing the decimal equivalent of those minterms that produce a 1 for the function. The symbol $\sum$ stands for the sum of the minterms

The maps for functions of two, three, and four variables are shown in Fig. 4



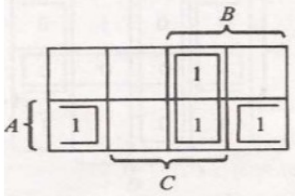
(a) Two-variable map (b) Three-variable map (c) Four-variable map

Figure 4 Maps for two-, three-, and four-variable functions.

The minterm represented by a square is determined from the binary assignments of the variables along the left and top edges in the map. Minterms of adjacent squares in the map are identical except for one variable, which appears complemented in one square and uncomplemented in the adjacent square. According to this definition of adjacency the squares at the extreme ends of the same horizontal row are also to be considered adjacent. The same applies to the top and bottom squares of a column. the four corner squares of a map must also be adjacent. A Boolean function represented by a truth table is plotted into the map by inserting 1's in those squares where the function is 1. The squares containing 1's are combined in groups of adjacent squares.

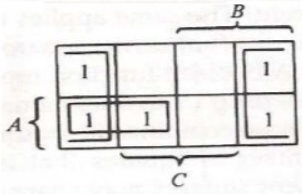
The squares containing 1's are combined in groups of adjacent squares. These groups must contain a number of squares that is an integral power of 2. Groups of combined adjacent squares may share one or more squares with one or more groups. Each group of squares represents an algebraic term, and the OR of those terms gives the simplified algebraic expression for the function.

Map for $F(A, B, C) = \Sigma(3,4,6,7)$.



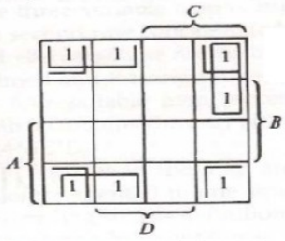
simplified algebraic expression
$F = BC + AC'$

Map for $F(A, B, C) = \Sigma(0,2,4,5,6)$.



The simplified function is
$F = C' + AB'$

Map for $F(A, B, C, D) = \Sigma(0,1,2,6,8,9,10)$.



The simplified function is
$F = B'D' + B'C' + A'CD'$

The preceding examples were expressed in sum-of-products form. The product terms are AND terms and the sum denotes the ORing of these terms

Products of Sum:

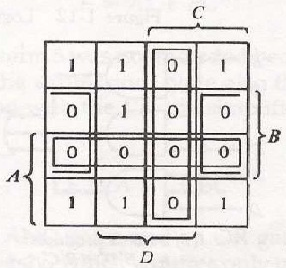
sums are OR terms and the product denotes the ANDing of these terms. With a minor modification, a product-of-sums form can be obtained from a map.

The squares not marked by 1 represent the minterms that produce 0 for the function. If we mark the empty squares with 0's and combine them into groups of adjacent squares, we obtain the complement of the function, F'. Taking the complement of F' produces an expression for F in product-of-sums form.

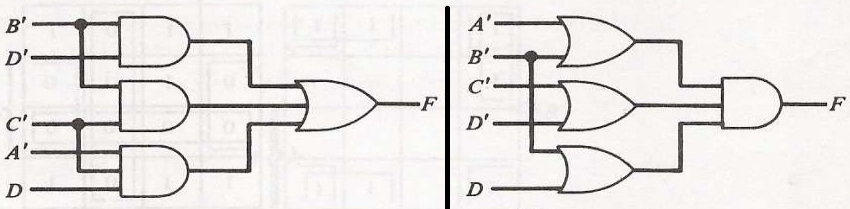
If the squares marked with 0's are combined, as shown in the diagram, we obtain the simplified complemented function: $F' = AB + CD + BD'$

Taking the complement of F', we obtain the simplified function in product-of-sums form: $F = (A' + B')(C' + D')(B' + D)$

Map for $F(A, B, C, D) = \Sigma(0,1,2,5,8,9,10)$.

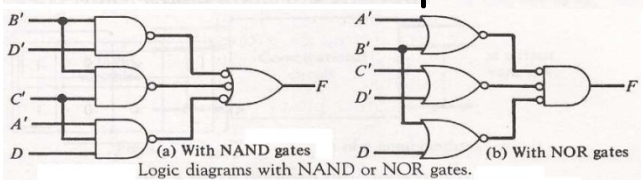


Logic diagrams with AND and OR gates.



(a) Sum of products: $F = B'D' + B'C' + A'C'D$

(b) Product of sums: $F = (A' + B')(C' + D')(B' + D)$

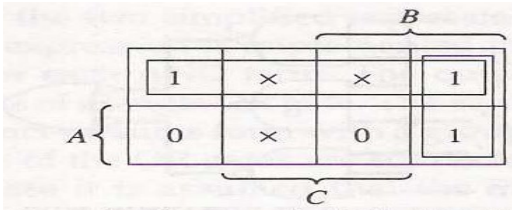


Logic diagrams with NAND or NOR gates.

Dont care condition:

Minterms that may produce either 0 or 1 for the function are said to be don't-care conditions and are marked with an × in the map. These don't-care conditions can be used to provide further simplification of the algebraic expression.

When choosing adjacent squares for the function in the map, the ×'s may be assumed to be either 0 or 1, whichever gives the simplest expression. In addition, an × need not be used at all if it does not contribute to the simplification of the function.consider the following Boolean function together with the don't-care minterms: $F(A, B, C) = \Sigma(0, 2, 6)$ | $d(A, B, C) = \Sigma(1, 3, 5)$



Example of map with don't-care conditions.

The 1's and ×'s are combined
The simplified expression is

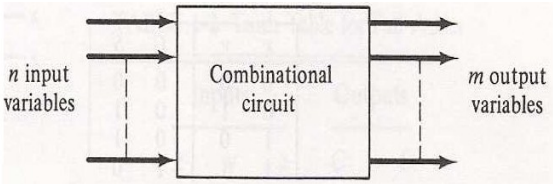
$$F = A' + BC'$$

Note that don't-care minterm 5 was not included because it does not contribute to the simplification of the expression. represents the Boolean function

$$F(A, B, C) = \sum (0, 1, 2, 3, 6)$$

Combinational Circuits:

A combinational circuit is a connected arrangement of logic gates with a set of inputs and outputs. At any given time, the binary values of the outputs are a function of the binary combination of the inputs. A block diagram of a combinational circuit is



Block diagram of a combinational circuit.

A combinational circuit can be described by a truth table showing the binary relationship between the n input variables and the m output variables. The truth table lists the corresponding output binary values for each of the 2^n input combinations. A combinational circuit can also be specified with m Boolean functions, one for each output variable. Each output function is expressed in terms of the n input variables.

The design of combinational circuits starts from the verbal outline of the problem and ends in a logic circuit diagram. The procedure involves the following steps:

1. The problem is stated.
2. The input and output variables are assigned letter symbols.
3. The truth table that defines the relationship between inputs and outputs is derived.
4. The simplified Boolean functions for each output are obtained.
5. The logic diagram is drawn.

Half-Adder

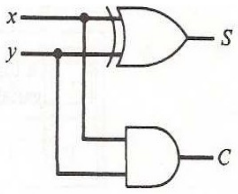
The most basic digital arithmetic circuit is the addition of two binary digits. A combinational circuit that performs the arithmetic addition of two bits is called a half-adder. One that performs the addition of three bits (two significant bits and a previous carry) is called a full-adder.

The input variables of a half-adder are called the augend and addend bits. The output variables the sum and carry. It is necessary to specify two output variables because the sum of 1 + 1 is binary 10, which has two digits.

Half-adder.

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

(a) Truth table



(b) Logic diagram

$S = x'y + xy' = x \oplus y$

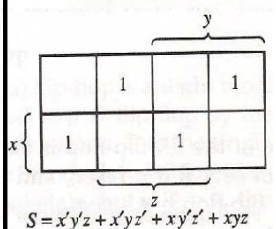
The Boolean functions for the two outputs can be obtained directly from the truth table: $C = xy$

Full-Adder

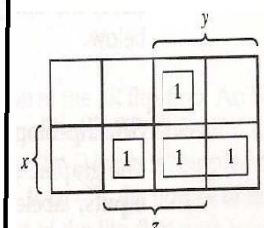
A full-adder is a combinational circuit that forms the arithmetic sum of three input bits. It consists of three inputs and two outputs. Two of the input variables, denoted by x and y, represent the two significant bits to be added. The third input, z, represents the carry from the previous lower significant position. Boolean expressions for the full-adder:

$S = x \oplus y \oplus z$ $\quad C = xy + (x \oplus y)z$

Maps for full-adder.



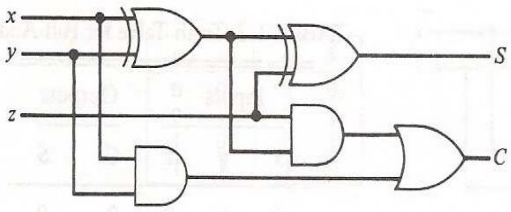
$S = x'y'z + x'yz' + xy'z' + xyz$
$= x \oplus y \oplus z$



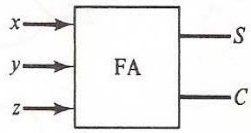
$C = xy + xz + yz$
$= xy + (x'y + xy')z$

TABLE Truth Table for Full-Adder

Inputs			Outputs	
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



(a) Logic diagram Full-adder circuit.



(b) Block diagram

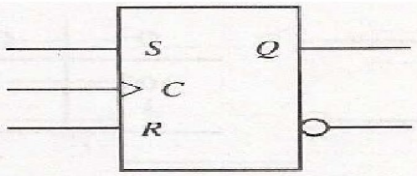
Although every digital system is likely to have a combinational circuit, most systems encountered in practice also include storage elements, which require that the system be described in terms of sequential circuits. The most common type of sequential circuit is the synchronous type. Synchronous sequential circuits employ signals that affect the storage elements only at discrete instants of time. Synchronization is achieved by a timing device called a clock pulse generator that produces a periodic train of *clock pulses*. The clock pulses are distributed throughout the system in such a way that storage elements are affected only with the arrival of the synchronization pulse. Clocked synchronous sequential circuits are the type most frequently encountered in practice.

Flip- Flop

The storage elements employed in clocked sequential circuits are called flip-flops. A flip-flop is a binary cell capable of storing one bit of information. It has two outputs, one for the normal value and one for the complement value of the bit stored in it.

SR Flip-Flop

The graphic symbol of the *SR* flip-flop . It has three inputs, labeled *S* (for set), *R* (for reset), and *C* (for clock). It has an output *Q* and sometimes the flip-flop has a complemented output, which is indicated with a small circle at the other output terminal. There is an arrowhead-shaped symbol in front of the letter *C* to designate a *dynamic input*. The dynamic indicator symbol denotes the fact that the flip-flop responds to a positive transition (from 0 to 1) of the input clock signal.



S	*R*	*Q* (*t* + 1)	
0	0	*Q* (*t*)	No change
0	1	0	Clear to 0
1	0	1	Set to 1
1	1	?	Indeterminate

(a) Graphic symbol (b) Characteristic table

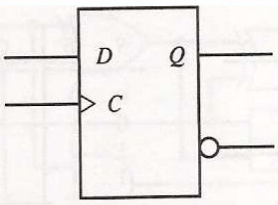
SR flip-flop.

Q(*t*) is the binary state of the *Q* output *present state*

Q(*t* + 1) is the binary state of the *Q* output after the occurrence of a clock transition *next state*

D Flip-Flop

The *D* (data) flip-flop is a slight modification of the *SR* flip-flop. An *SR* flip-flop is converted to a *D* flip-flop by inserting an inverter between *S* and *R* and assigning the symbol *D* to the single input. The *D* input is sampled during the occurrence of a clock transition from 0 to 1. If *D* = 1, the output of the flip-flop goes to the 1 state, but if *D* = 0, the output of the flip-flop goes to the 0 state.



D	*Q* (*t* + 1)	
0	0	Clear to 0
1	1	Set to 1

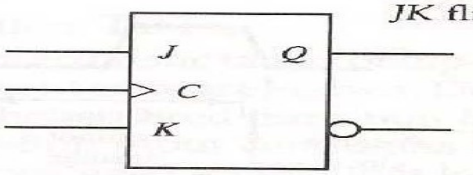
(a) Graphic symbol (b) Characteristic table

D flip-flop.

The "no change" condition can be accomplished either by disabling the clock signal or by feeding the output back into the input, so that clock pulses keep the state of the flip-flop unchanged.

JK Flip-Flop

A *JK* flip-flop is a refinement of the *SR* flip-flop in that the indeterminate condition of the *SR* type is defined in the *JK* type. Inputs *J* and *K* behave like inputs *S* and *R* to set and clear the flip-flop, respectively. When inputs *J* and *K* are both equal to 1, a clock transition switches the outputs of the flip-flop to their complement state.



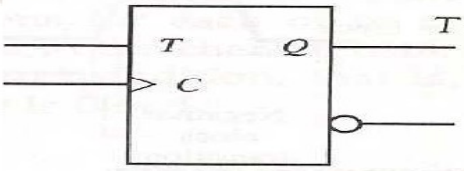
J	*K*	*Q* (*t* + 1)	
0	0	*Q* (*t*)	No change
0	1	0	Clear to 0
1	0	1	Set to 1
1	1	*Q*′ (*t*)	Complement

(a) Graphic symbol (b) Characteristic table

JK flip-flop.

T Flip-Flop

Another type of flip-flop found in textbooks is the *T* (toggle) flip-flop. This flip-flop, is obtained from a *JK* type when inputs *J* and *K* are connected to provide a single input designated by *T*.



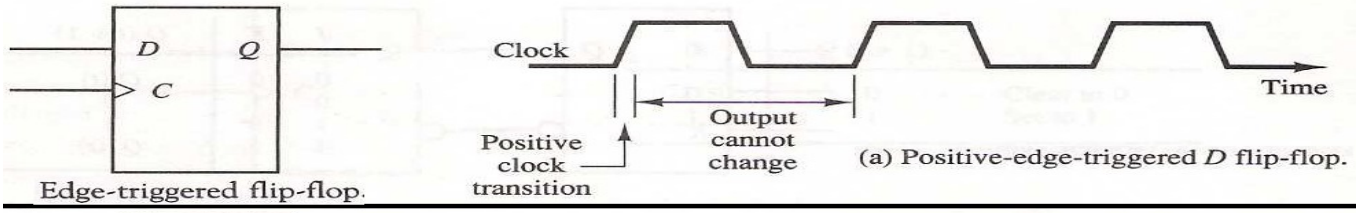
T	*Q* (*t* + 1)	
0	*Q* (*t*)	No change
1	*Q*′ (*t*)	Complement

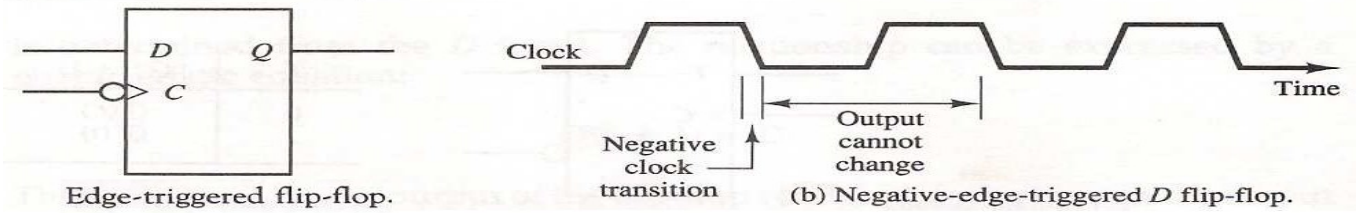
(a) Graphic symbol (b) Characteristic table

T flip-flop.

Edge-Triggered Flip-Flops

The most common type of flip-flop used to synchronize the state change during a clock pulse transition is the edge-triggered flip-flop. In this type of flip-flop, output transitions occur at a specific level of the clock pulse. When the pulse input level exceeds this threshold level, the inputs are locked out so that the flip-flop is unresponsive to further changes in inputs until the clock pulse returns to 0 and another pulse occurs. Some edge-triggered flip-flops cause a transition on the rising edge of the clock signal (positive-edge transition), and others cause a transition on the falling edge (negative-edge transition).





master-slave flip-flop

Another type of flip-flop used in some systems is the master-slave flip-flop. This type of circuit consists of two flip-flops. The first is the master, which responds to the positive level of the clock, and the second is the slave, which responds to the negative level of the clock. The result is that the output changes during the 1-to-0 transition of the clock signal.

Excitation Tables

The characteristic tables of flip-flops specify the next state when the inputs and the present state are known. During the design of sequential circuits we usually know the required transition from present state to next state and wish to find the flip-flop input conditions that will cause the required transition. For this reason we need a table that lists the required input combinations for a given change of state. Such a table is called a flip-flop excitation table.

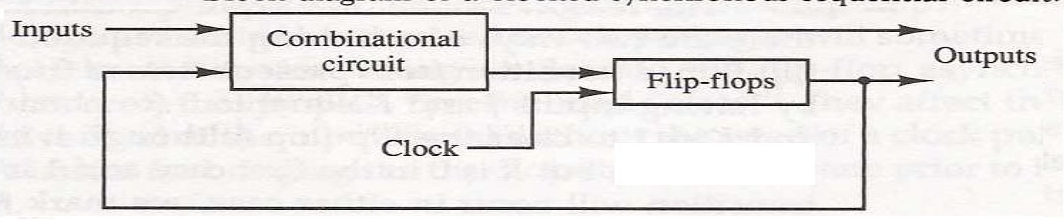
TABLE Excitation Table for Four Flip-Flops

SR flip-flop				*D* flip-flop			*JK* flip-flop				*T* flip-flop		
Q(*t*)	*Q*(*t* + 1)	*S*	*R*	*Q*(*t*)	*Q*(*t* + 1)	*D*	*Q*(*t*)	*Q*(*t* + 1)	*J*	*K*	*Q*(*t*)	*Q*(*t* + 1)	*T*
0	0	0	×	0	0	0	0	0	0	×	0	0	0
0	1	1	0	0	1	1	0	1	1	×	0	1	1
1	0	0	1	1	0	0	1	0	×	1	1	0	1
1	1	×	0	1	1	1	1	1	×	0	1	1	0

Sequential Circuits

A sequential circuit is an interconnection of flip-flops and gates. The gates by themselves constitute a combinational circuit, but when included with the flip-flops, the overall circuit is classified as a sequential circuit.
This process demonstrates
that the external outputs of a sequential circuit are functions of both external inputs and the present state of the flip-flops. Moreover, the next state of flip-flops is also a function of their present state and external inputs. Thus a sequential circuit is specified by a time sequence of external inputs, external outputs, and internal flip-flop binary states.

Block diagram of a clocked synchronous sequential circuit.



Flip-Flop Input Equations

An example of a sequential circuit is shown in Fig. It has one input x - variable x, one output variable y, and two clocked D flip-flops. The AND gates, OR gates, and inverter form the combinational logic part of the circuit. The interconnections among the gates in the combinational circuit can be specified by a set of Boolean expressions. The part of the combinational circuit that generates the inputs to flip-flops are described by a set of Boolean expressions called flip-flop input equations. we write the first input equation as

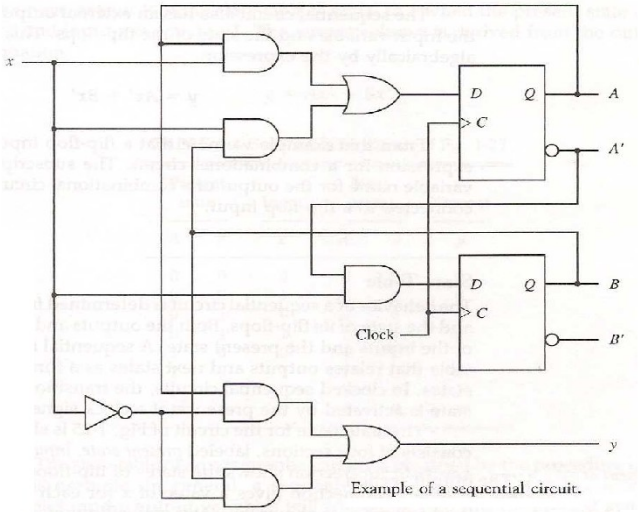
$D_A = Ax + Bx$ second input equation $D_B = A'x$ This output can be specified by the expression $y = Ax' + Bx'$

State Table

The behavior of a sequential circuit is determined from the inputs, the outputs, and the state of its flip-flops. Both the outputs and the next state are a function of the inputs and the present state. A sequential circuit is specified by a state table The state table for the circuit consists of four sections, labeled *present state*, *input*, *next state*, and *output*.

State Diagram

The information available in a state table can be represented graphically in a state diagram. In this type of diagram, a state is represented by a circle, and the transition between states is indicated by directed lines connecting the circles. The binary number inside each circle
identifies the state of the flip-flops. The directed lines are labeled with two binary numbers separated by a slash. The input value during the present state is labeled first and the number after the slash gives the output during the present state.

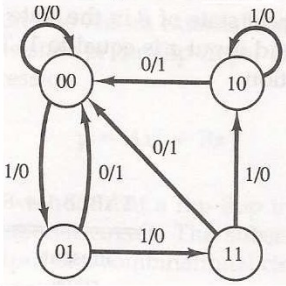


Example of a sequential circuit.

State Table for Circuit

Present state		Input	Next state		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

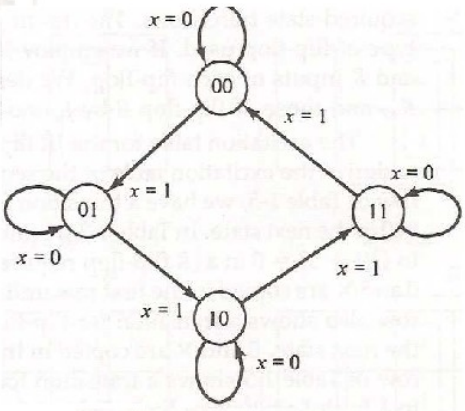
State diagrams of circuit.



Design Example

We wish to design a clocked sequential circuit that goes through a sequence of repeated binary states 00, 01, 10, and 11 when an external input x is equal to 1. The state of the circuit remains unchanged when $x = 0$. This type

The binary counter needs two flip-flops to represent the two bits.

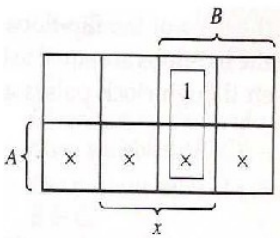


State diagram for binary counter.

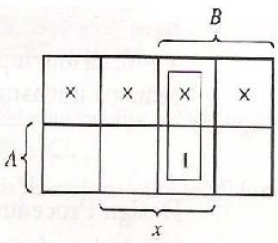
$$J_A = Bx \qquad K_A = Bx$$
$$J_B = x \qquad K_B = x$$

Excitation Table for Binary Counter

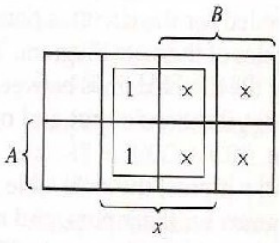
Present state		Input	Next state		Flip-flop inputs			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	×	0	×
0	0	1	0	1	0	×	1	×
0	1	0	0	1	0	×	×	0
0	1	1	1	0	1	×	×	1
1	0	0	1	0	×	0	0	×
1	0	1	1	1	×	0	1	×
1	1	0	1	1	×	0	×	0
1	1	1	0	0	×	1	×	1



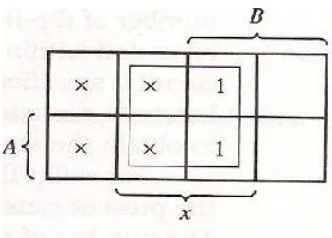
$J_A = Bx$



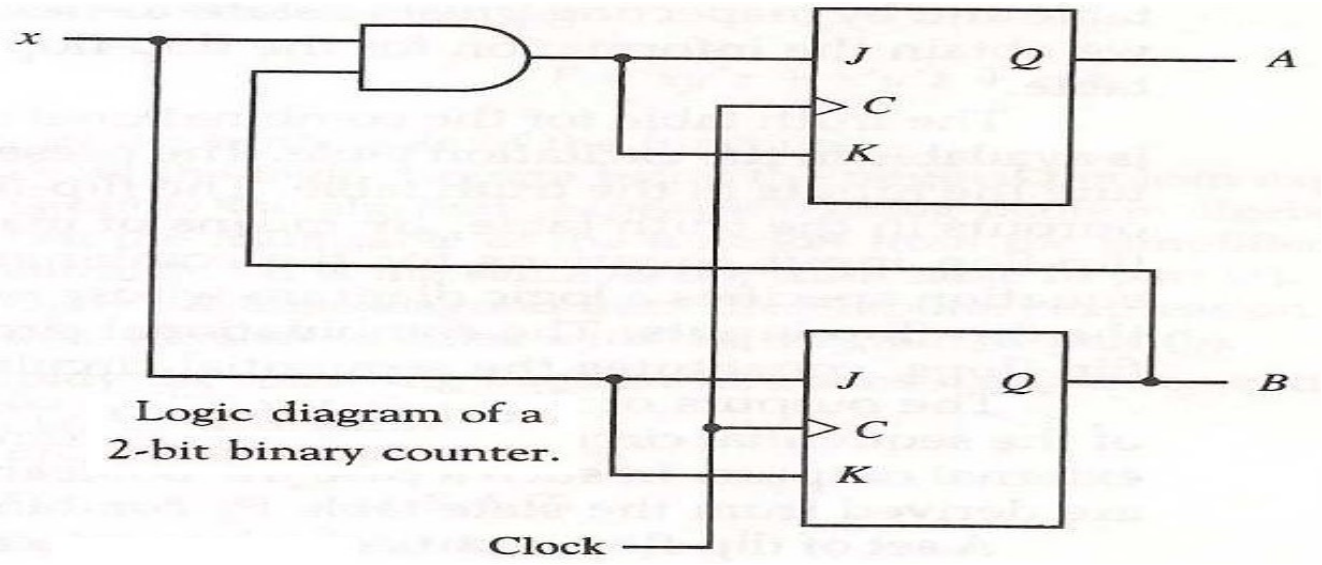
$K_A = Bx$



$J_B = x$



$K_B = x$



Logic diagram of a
2-bit binary counter.