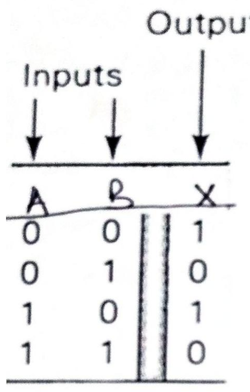
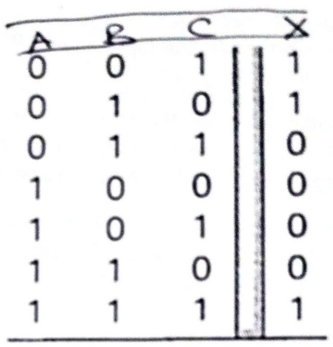
TRUTH TABLE

A truth table is a means for describing how a logic circuit's output depends on the logic levels present at the circuit's inputs. The Figure below illustrates a truth table for one type of two-input logic circuit. The table lists all possible combinations of logic levels present at inputs *A* and *B* along with the corresponding output level *x*. The first entry in the table shows that when *A* and *B* are both at the O level, the output *x* is at the 1 level or, equivalently, in the 1 state. The second entry shows that when input *B* is changed to the 1 state, so that *A = 0* and *B = 1,* the output *x* becomes a 0. In a similar way, the table shows what happens to the output state for any set of input conditions.


Output
Inputs

A	B	X
0	0	1
0	1	0
1	0	1
1	1	0

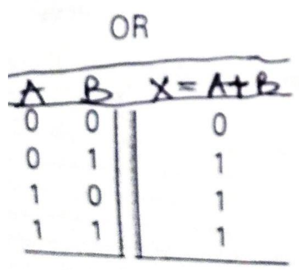


A	B	C	X
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

The OR operation is the first of the three basic Boolean operations to be learned. The truth table shows what happens when two logic inputs, *A* and *B*, are combined using the OR operation to produce the output *x.* The table shows that is a logic 1 for every combination of input levels where one or more inputs are 1, the only case where *x* is a 0 is when both inputs are 0.

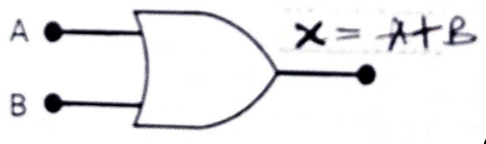
The Boolean expression for the OR operation is

$$x = A + B$$



OR

A	B	X=A+B
0	0	0
0	1	1
1	0	1
1	1	1



OR gate

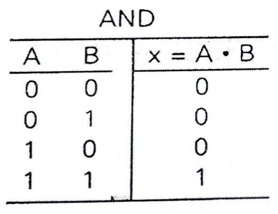
OR Gate

In digital circuitry an OR gate* is a circuit that has two or more inputs and whose output is equal to the OR combination of the inputs. The figure is the logic symbol for a two input OR gate. The inputs *A* and *B* are logic voltage levels, and the output x is a logic voltage level whose value is the result of the OR operations on A and B, that is, $x = A + B$. In other words, the OR gates operates in such a way that its output is HIGH (logic I) if either input *A or B* or both are at a logic 1 level. The OR gate output will be LOW (logic O) only if all its inputs are at logic O.

The AND operation is the second basic Boolean operation. The truth table shows what happens when two logic inputs, *A* and *B*, are combined using the AND operation to produce output x. The table shows that x is a logic 1 only when both *A* and *B* are at the logic 1 level. For any case where one of the inputs is 0, the output is 0.

The Boolean expression for the AND operation is

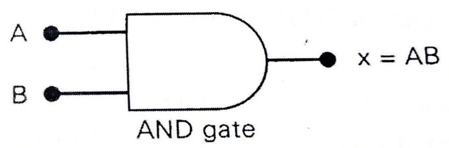
$$x = A . B$$



AND

A	B	x = A • B
0	0	0
0	1	0
1	0	0
1	1	1

(a)



(b)

AND Gate

The logic symbol for a two-input AND gate is shown in the figure. The AND gate output is equal to the AND product of the logic inputs; that is, = AB. In other words, the AND gate is a circuit that operates in such a way that its output is HIGH only when all its inputs are HIGH. For all other cases the AND gate output is LOW.

The NOT operation is unlike the OR and AND operations in that it can be performed on a single input variable. For example, if the valuable A is subjected to the NOT operation, the result x can be expressed as

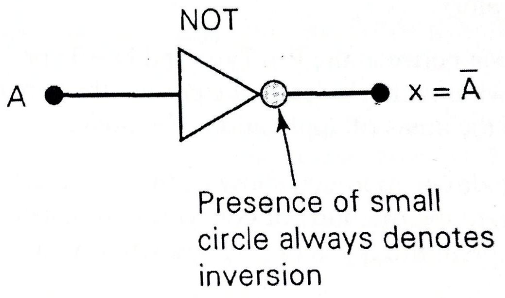
$$X = \bar{A}$$

Where the overbar represents the NOT operation. This expression is read as "x equals NOT A" or "x equals the inverse of A" or 'x equals the complement of A." Each of these is in common usage, and all indicate that the logic value of $x = \bar{A}$ is opposite to the logic value of A. The truth table in the figure clarifies this for the two cases A = O and A = 1. That is,

$\bar{1} = 0$ because NOT 1 is 0

NOT

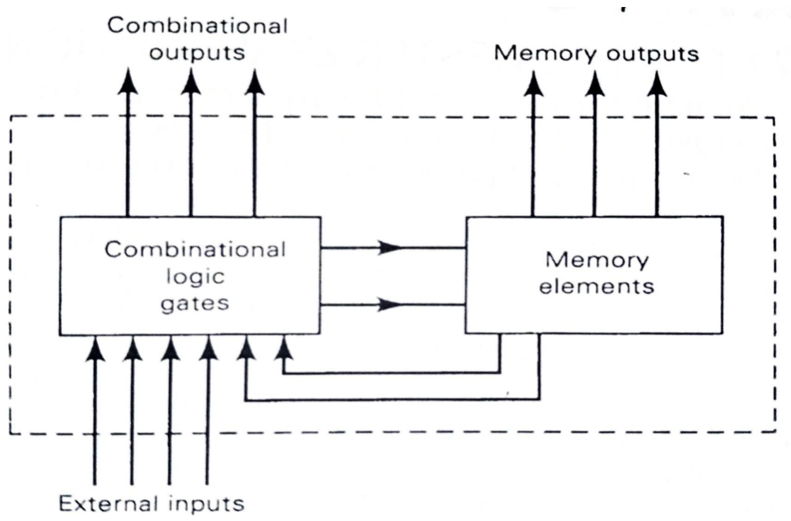
A	$X = \bar{A}$
0	1
1	0



NOT Circuit (INVERTER)

The second figure shows the symbol for a NOT circuit, which is more commonly called an INVERTER. This circuit always has only a single input, and its output logic level is always opposite to the logic level of this input. The INVERTER affects an input signal. It inverts (complements) the input signal at all points on the waveform so that whenever the input = 0, output = 1, and vice versa.

GENERAL DIGITAL SYSTEM



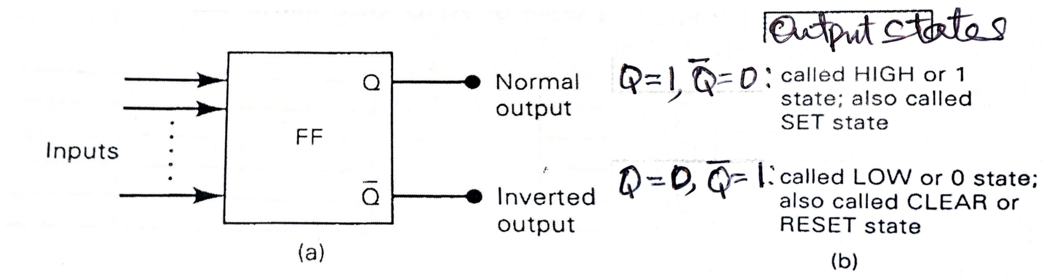
because combinational logic circuits have no memory. Most digital systems are made up of both combinational circuits and memory elements.

The figure shows a block diagram of a general digital system that combines logic signals from external inputs and from the outputs of the memory elements. The combinational circuit operates on these inputs to produce various outputs, some of which are used to determine the binary values to be stored in the memory elements. The outputs of some of the memory elements, in turn, go to the inputs of logic gates in the combinational circuits. This process indicates that the external outputs of a digital system are a function of both its external inputs and the information scored in its memory elements.

The most important memory element is the flip-flop, which is made up of an assembly of logic gates. Even though a logic gate, by itself, has no storage capability, several can be connected together in ways that permit information to be stored. Several different gate arrangements are used to produce these flip-flops (abbreviated FF).

The figure is the general type of symbol used for a flip-flop. It shows two outputs, labeled Q and Q, that are the inverse of each other. Q/Q are the most common designations used for a FF's outputs. From time to time, we will use other designations such as X/X and A/A convenience in identifying different FFs in a logic circuit.

The Q output is called the normal FF output, and Q is the inverted FF output. Whenever we refer to the state of a FF, we are referring to the state of its normal (Q) output; it is understood that its inverted output (Q) is in the opposite state.


Output states
Inputs
FF
Q
$\bar{Q}$
Normal output
Inverted output
$Q=1, \bar{Q}=0$: called HIGH or 1 state; also called SET state
$Q=0, \bar{Q}=1$: called LOW or 0 state; also called CLEAR or RESET state
(a)
(b)

For example, if we say that a FF is in the HIGH (1) state, we mean that Q = 1; if we say that a FF is in the LOW (0) state, we mean that Q = 0. Of course, the Q state will always be the inverse of Q.

The two possible operating states for a FF are summarized in Figure 5-2(b). Note that the HIGH or 1 state (Q = 1/Q = 0) is also referred to as the SET state. Whenever the inputs to a FF cause it to go to the Q = 1 state, we call this setting the FF; the FF has been set. In a similar way, the LOW or 0 state (Q = 0/Q = 1) is also referred to as the CLEAR or RESET state. Whenever the inputs to a FF cause it to go to the Q = 0 state, we call this clearing or resetting the FF; the FF has been cleared (reset). As we shall see, many FFs will have a SET input and/or a CLEAR (RESET) input that is used to drive the FF into a specific output state.

As the symbol in Figure 5-2 (a) implies, a FF can have one or more inputs. These inputs are used to cause the FF to switch back and forth ("flip-flop") between its possible output states. We will find out that most FF inputs need only to be momentarily activated (pulsed) in order to cause a change in the FF output state, and the output will remain in that new state even after the input pulse is over. This is the FF's memory characteristic.

The flip-flop is known by other names, including latch and bistable multivibrator. The term latch is used for certain types of flip-flops that we will describe. The term bistable multivibrator is the more technical name for a flip-flop, but it is too much of a mouthful to be used regularly.

Setting the Latch (FF)

Now let's investigate what happens when the SET input is momentarily pulsed LOW while CLEAR is kept HIGH. Figure shows what happened when Q = 0 prior to the occurrence of the pulse. As SET is pulsed LOW at time to1, Q will go HIGH, and this HIGH will force Q to go LOW so that NAND-1 now has two LOW inputs. Thus, when SET returns to the 1 state at t1, the NAND-1 output remains HIGH, which, in turn, keeps the NAND-2 output LOW.

Figure shows what happens when Q = 1 and Q = 0 prior to the application of the SET pulse. Since Q is already keeping the NAND-1 output HIGH, the LOW pulse at SET will not change anything. Thus, when SET returns HIGH, the latch outputs are still in the Q = 1, Q = 0 state.

We can summarize this figure by stating that a LOW pulse on the SET input will always cause the latch to end up in the Q = 1 state, This operation is called setting the latch or FF.

Digital systems can operate either asynchronously or synchronously. In asynchronous systems, the output of logic circuits can change state any time one or more of the inputs change. An asynchronous system is generally more difficult to design and troubleshoot than a synchronous system.

In synchronous systems, the exact times at which any output can change states are determined by a signal commonly called the clock. This clock signal is generally a rectangular pulse train or a square wave as shown in Figure 5-14. The clock signal is distributed to all parts of the system, and most (if not all) of the system outputs can change state only when the clock makes a transition. The transitions (also called edges) are pointed out in the. When the clock changes from a 0 to a 1, this is called the positive-going transition (PGT); when the Lo goes from 1 to 0, this is the negative-going transition (NGT). We will use the abbreviations PGT and NGT, since these terms appear so often throughout the text.

Most digital systems are principally synchronous (although there are always asynchronous parts), since synchronous circuits are easier to design and troubleshoot. They are easier to troubleshoot because the circuit outputs can change only at specific instants of time. In other words, almost everything is synchronized to the clock signal transitions.

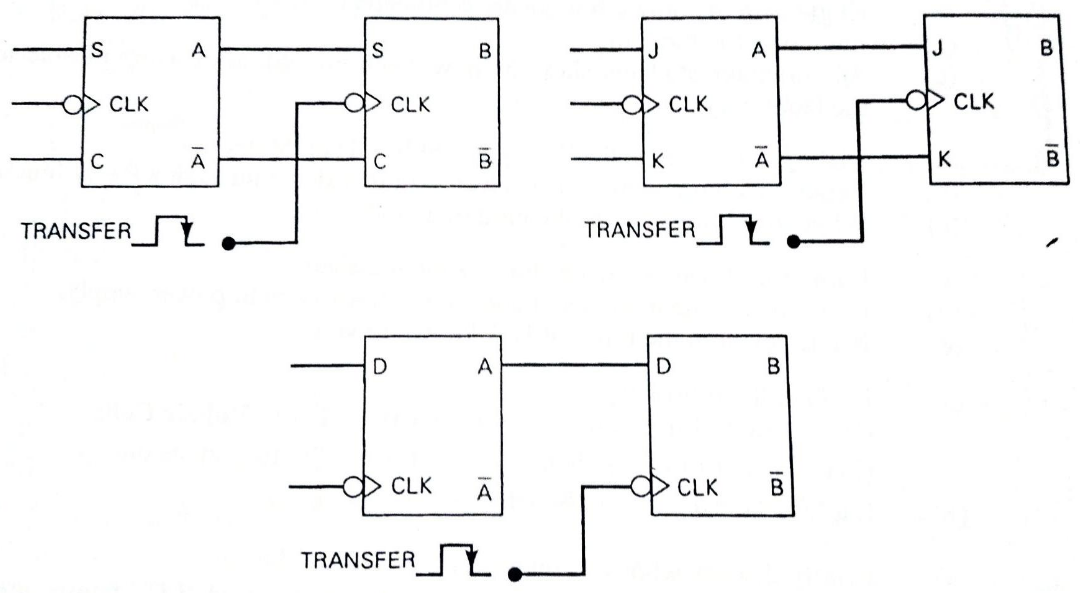
The synchronizing action of the clock signals is accomplished through the use of clocked flip-flops that are designed to change states on one or the other of the clock's transitions.

Edge-triggered (clocked) flip-flops are versatile devices that can be used in a wide variety of applications including counting, storing of binary data, transferring binary data from one location to another, and many more. Almost all of these applications utilize the FF's clocked operation. Many of them fall into the category of sequential circuits. A sequential circuit is one in which the output follow a predetermined sequence of states, with a new state occurring each time a clock pulse occurs. We will introduce some of the basic applications in the following sections, and we will expand on them in subsequent chapters.

By far the most common use of flip-flops is for the storage of data or information. The data may represent numerical values (e.g., binary numbers, BCD-coded decimal numbers) or any of a wide variety of types of data that have been encoded in binary. These data are generally stored in groups of FFs called registers.

The operation most often performed on data that are stored in a FF or a register is the data transfer operation. This involves the transfer of data from one FF or register to another. Figure 5-40 illustrates how data transfer can be accomplished between two FFs using clocked S-C, J-K, and flip-flops In each case, the logic value that is currently stored in FF A is transferred to FF B upon the NGT of the TRANSFER pulse. This, after this NGT, the B output will be the same as the A output.

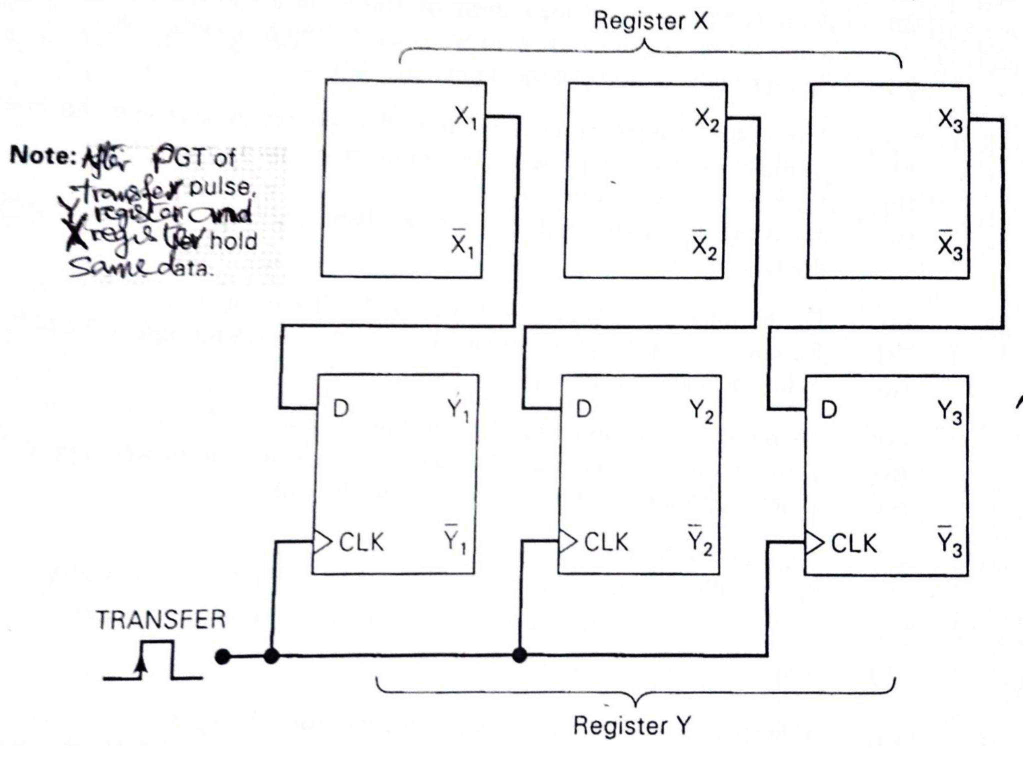
The transfer operations in Figure 5-40 are examples of synchronous transfer, since the synchronous control and CLK inputs are used to perform the transfer. A transfer operation can also be obtained using the asynchronous inputs of a FF. Figure 5-41 shows how an asynchronous transfer can be accomplished using the PRESET and CLEAR inputs of any type of FF. Here, the asynchronous inputs respond



to LOW levels. When the TRANSFER ENABLE line is held LOW, the two NAND outputs are kept HIGH, with no effect on the FF outputs. When the TRANSFER ENABLE line is more HIGH, one of the NAND outputs will go LOW, depending on the state of the A and A outputs. This LOW will either set or clear FF B to the same state as FF A. This asynchronous transfer is done independently of the synchronous and CLK inputs of the FF. Asynchronous transfer is also called jam transfer, because the data can be"jammed" into FF B even if it's synchronous inputs are active.

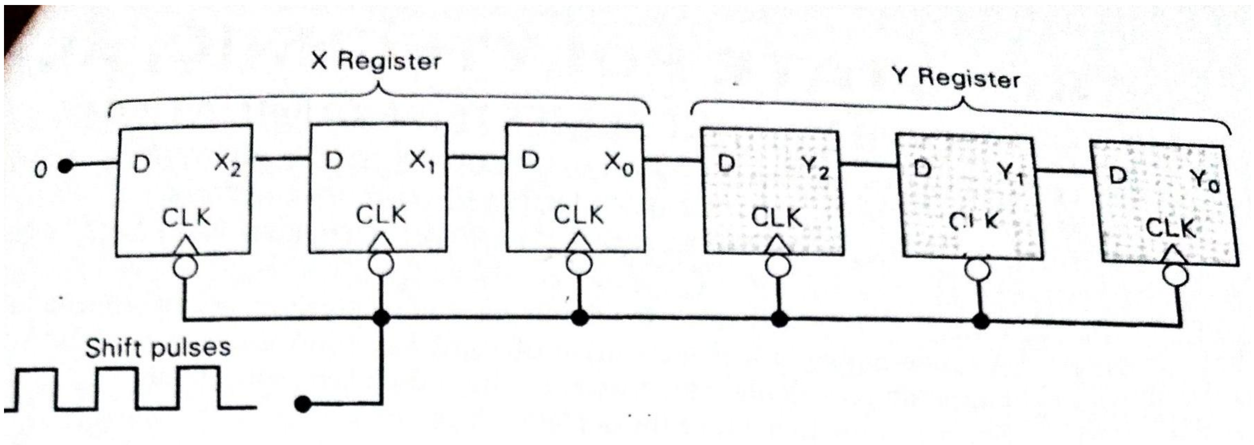
Parallel Data Transfer

Figure 5-42 illustrates data transfer from one register to another using D- type FFs. Register X consists of FFs X1, X2, and X3; register Y consists of FFs Y1, Y2, and Y3. Upon application of the PGT of the TRANSFER pulse, the level stored in X1 is transferred to Y1, Y2, and Y3 to Y3. The transfer of the contents of the X register into the Y register is a synchronous transfer. It is also referred to as a parallel transfer,



Since the contents of ....., are transferred simultaneously into .... If a serial transfer were performed, the contents of the X register would be transferred to the Y register one bit at a time. This will be examined in the next section.

Before we describe the serial data transfer operation, we must first examine the basic shift-register arrangement. A shift register is a group of FFs arranged so that the binary numbers stored in the FFs are shifted from one FF to the next for every clock pulse. You have undoubtedly seen shift registers in action in devices such as an electronic calculator, where the digits shown on the display shift over each time you key in a new digit. This is the same action taking place in a shift register.


X Register
Y Register
0
D
X2
D
X1
D
X0
D
Y2
D
Y1
D
Y0
CLK
CLK
CLK
CLK
CLK
CLK
Shift pulses