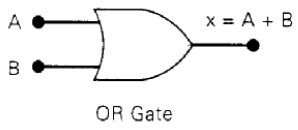
BOOLEAN THEOREMS

1	$x \cdot 0 = 0$	2.	$x \cdot 1 = x$	3.	$x \cdot x = x$
4.	$x \cdot \bar{x} = 0$	5.	$x + 0 = x$	6.	$x + 1 = 1$
7	$x + x = x$	8.	$x + \bar{x} = 1$	9.	$x + y = y + x$
10.	$x \cdot y = y \cdot x$	11.	$x + (y + z) = (x + y) + z = x + y + z$	12.	$x(yz) = (xy)z = xyz$
13a.	$x(y + z) = xy + xz$	13b.	$(w + x)(y + z) = wy + xy + wz + xz$	14.	$x + xy = x$
15a.	$x + \bar{x}y = x + y$	15b.	$\bar{x} + xy = \bar{x} + y$	16.	$\overline{x + y} = \bar{x}\,\bar{y}$
17.	$\overline{xy} = \bar{x} + \bar{y}$				

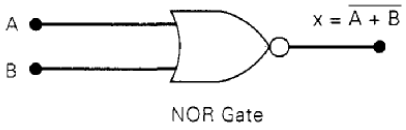
LOGIC GATE TRUTH TABLES

A	B	OR $A + B$	NOR $\overline{A + B}$	AND $A \cdot B$	NAND $\overline{A \cdot B}$	XOR $A \oplus B$	XNOR $\overline{A \oplus B}$
0	0	0	1	0	1	0	1
0	1	1	0	0	1	1	0
1	0	1	0	0	1	1	0
1	1	1	0	1	0	0	1

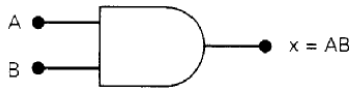
LOGIC GATE SYMBOLS



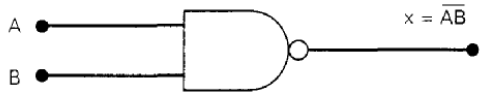
OR Gate



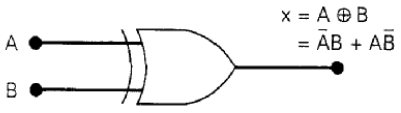
NOR Gate



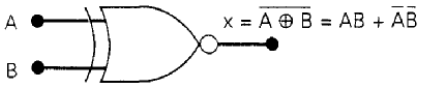
AND Gate



NAND Gate



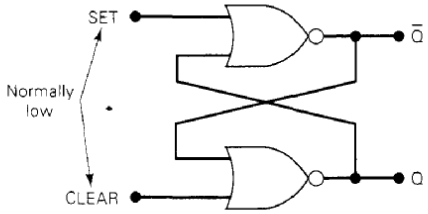
XOR

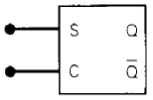


XNOR

FLIP-FLOPS

NOR Latch

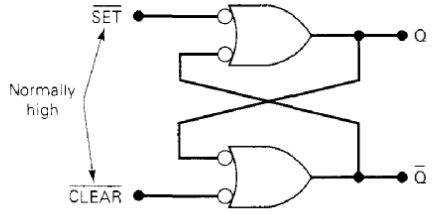


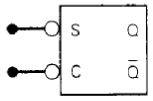


(Alternate symbol)

S	C	Q
0	0	No change
1	0	Q = 1
0	1	Q = 0
1	1	Invalid

NAND Latch

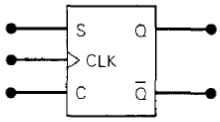




(Alternate symbol)

S	C	Q
0	0	Invalid
1	0	Q = 0
0	1	Q = 1
1	1	No change

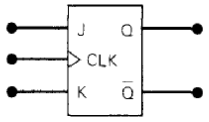
Clocked S-C



S	C	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	Ambiguous

↓ of CLK has no effect on Q

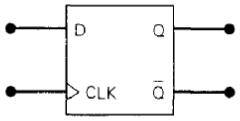
Clocked J-K



J	K	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q_0}$ (toggles)

↓ of CLK has no effect on Q

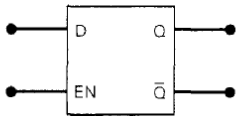
Clocked D



D	CLK	Q
0	↑	0
1	↑	1

↓ of CLK has no effect on Q

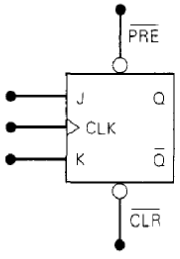
D Latch



EN	D	Q*
0	X	No change
1	0	0
1	1	1

*Q follows D input while EN is HIGH

Asynchronous Inputs



$\overline{PRE}$	$\overline{CLR}$	Q*
1	1	No effect; FF can respond to J, K and CLK
1	0	Q = 0 independent of J, K, CLK
0	1	Q = 1 independent of J, K, CLK
0	0	Ambiguous (not used)

*CLK can be in any state

accordance with ***uniform clocking pulses***. In other words the present input and present output can be measured prior to the active edge of a clocking pulse. In asynchronous cases the behavior of the sequential circuit depends on the change in the input signal logic levels. These circuits tend to become less stable and thus are sometimes avoided. The synchronous sequential logic circuits, on the other hand, are more stable and are widely used. The memory element in sequential circuits, called ***flip-flop***, will be reviewed next.

V-5.1. FLIP-FLOPS

These units permanently store binary information until certain control (or input) signals (at most three) are applied to them causing a change in their status. The way that these control signals activate these units (flip-flops) is the reason for their differences. Actually each flip-flop has two outputs that are generally the complement of each other and thus the main source of difference among various flip-flops are the ways that these input signals are implemented. These signals are either *static* or *dynamic* in the following sense: static signals refer to the cases in which the states of flip-flop are defined by just the logic values of the control signals; on the other hand, the dynamic signals refer to both logic values of the control signals and their timing, of course. This dynamic situation is also referred to as *clocking* or *triggering*. One final comment regarding the triggering is that in order to have a stable operation we must have a delay time for our flip-flop which takes longer than the clock pulse duration. In the following, and for the sake of refreshing the reader's memory, we will briefly introduce these various flip-flops with the understanding that these notations are not unique, neither are some of their truth tables. One may look at these truth table as starting points to synthesize the corresponding flip-flops with various logic gates.

V-5.1A. Basic R-S Flip-Flop

This is the simplest flip-flop and is known as a *direct-coupled* R-S flip-flop as well as R-S *latch*. Its symbol, truth table, and one possible logic diagram are shown in Fig. 22. The *NOR* gate implementation of this flip-flop is also possible. In this figure UC means *unchanged* and ND means *not defined* (a forbidden condition that must normally be avoided). These elements are also considered as asynchronous sequential logic circuits.

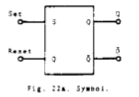


Fig. 22a. Symbol.

R	S	Q
1	1	ND
1	0	1
0	1	0
0	0	UC

Fig. 22b. Truth Table

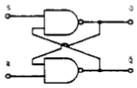
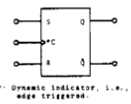


Fig. 22c. A Logic Diagram for R-S Flip-Flop *NAND* Gate Implementation

V-5.1B. Synchronous or Clocked R-S Flip-Flop

The previous basic flip-flop can be made synchronous or clocked as shown in Fig. 23. Here n refers to instant t_n and $Q_{n+1} = Q_n$ means UC conditions for Q_{n+1} which is the state after the occurrence of clock pulse. If the clock (*enable*) signal were inverted and the flip-flop reponds to the falling edge of the clock pulse, then we call this situation a *negative* clock flip-flop. If the flip-flop responds to the rising edge of the clock then we call this case a *positive* clock flip-flop. If there were small circles at a given input (S, R or C), then that represents negative-edge triggering, otherwise positive-edge triggering is assumed.



*: Dynamic indicator, i.e., edge triggered.

Fig. 23a. Symbol.

R_n	S_n	Q_{n+1}
0	0	Q_n
0	1	1
1	0	0
1	1	ND

Fig. 23b. Truth Table

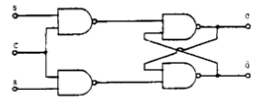


Fig. 23c. A Logic Diagram for Synchronous R-S Flip-Flop *NAND* Gate Implementation (positive-edge triggered)

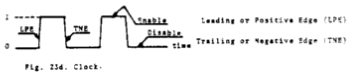
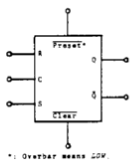


Fig. 23d. Clock.

To make flip-flop state independent of clock two new DC control inputs called *preset* and *clear* may be used. These are used when we are initializing the circuit. These input signals are applied to the last level of the logic circuits. The symbol and truth table for this type flip-flop are shown in Fig. 24.

The general properties of the above R-S flip-flops can be summerized as follows.

If S is *HIGH*, Q=1.
If R is *HIGH*, Q=0.
If $\overline{\text{Preset}}$ is *LOW*, Q=1.
If $\overline{\text{Clear}}$ is *LOW*, Q=0.



*: Overbar means *LOW*.

Fig. 24a. Symbol.

Synchronous

R_n	S_n	Q_{n+1}
0	0	Q_n
0	1	1
1	0	0
1	1	ND

Asynchronous *

Clear	Preset	Q^*
0	1	0
1	0	1

*: Independent of time

Fig. 24b. Truth Table.

V-5.1C. D-Type Flip-Flop

This is an improved version of the R-S flip-flop that has no UC state and uses only one input signal in addition to the clock pulse. The symbol, truth table and one possible logic design of this type flip-flop is shown in Fig. 25. This type of flip-flop is also known as a gated D-latch.

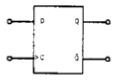


Fig. 25a. Symbol.

D_n	Q_{n+1}
0	0
1	1

Fig. 25b. Truth Table.

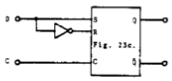


Fig. 25c.

Fig. 25c. A Logic Diagram for Implementation of D-Type Flip-Flop.

V-5.1D. J-K Flip-Flop

This further improves the R-S flip-flop in which the UC state is defined by the J-K inputs (K is for clear). The symbol, truth table and one possible logic design of this flip-flop is shown in Fig. 26.

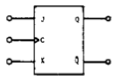


Fig. 26a. Symbol.

J_n	K_n	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\bar{Q}_n$

Fig. 26b. Truth Table.

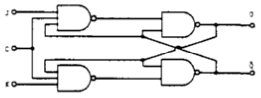


Fig. 26c. Clock J-K Flip-Flop.

V-5.1E. Toggle Flip-Flop

The R-S or J-K flip-flops can be modified to make the so-called Toggle flip-flop. In this case the state of flip-flop changes on each positive-edge clock pulse. When T is *HIGH* the flip-flop is toggling if the clock is at positive edge and when T is *LOW* the output remains unchanged. The symbol, truth table and a logic diagram for this type flip-flop is shown in Fig. 27. As this figure shows this type flip-flop can be made from those that are available in the market. In general, toggle means $Q_{n+1} = \bar{Q}_n$.

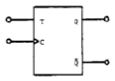


Fig. 27a. Symbol.

T_n	Q_{n+1}
0	0
1	1

Fig. 27b. Truth Table.

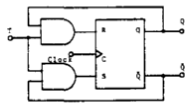


Fig. 27c. A Logic Diagram.

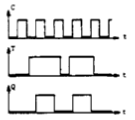


Fig. 27d. Waveforms.

V-5.1F. Master-Slave Flip-Flop

We can improve the performance of a flip-flop regarding timing and output by building this type flip-flop. The master-slave (or M-S) flip-flop is made by cascading two flip-flops of the previous types such that the input clock triggers the master flip-flop and its complement triggers the slave flip-flop. It is possible to synchronize all clock pulse inputs such that the binary contents of one flip-flop be transferred to the next one and *vice-versa* during the same clocking pulse interval.

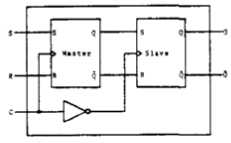


Fig. 28. A Logic Diagram of an M-S Flip-Flop

V-5.1G. Edge-Triggered Flip-Flop

The main aspect of this type of flip-flop is that the input data do not affect the output once the transition occurs and the input data reach the threshold value. There are a variety of different configurations for this type of flip-flop in most textbooks.

V-5.2. BASIC APPLICATIONS OF FLIP-FLOPS

In this section we will briefly mention a few general applications of flip-flops in building some widely used sequential logic circuits.