**Unit- 5 Sequential Circuit Design**

**Introduction of flip flops**

Flip-flops and latches are fundamental building blocks of [digital electronics](https://en.wikipedia.org/wiki/Digital_electronics) systems used in computers, communications, and many other types of systems. In [electronics](https://en.wikipedia.org/wiki/Electronics), a **flip-flop** or **latch** is a [circuit](https://en.wikipedia.org/wiki/Electronic_circuit) that has two stable states and can be used to store state information. Flip-flops can be divided into common types: the **SR** ("set-reset"), **D** ("data" or "delay"[), **T** ("toggle"), and **JK** .

Flip-flops can be either simple (transparent or opaque or asynchronous) or [clocked](https://en.wikipedia.org/wiki/Clock_signal) (synchronous or edge-triggered). Although the term flip-flop has historically referred generically to both simple and clocked circuits, in modern usage it is common to reserve the term *flip-flop* exclusively for clocked circuits; the simple ones are commonly called *latches*.

**Difference between latches and flip flops**

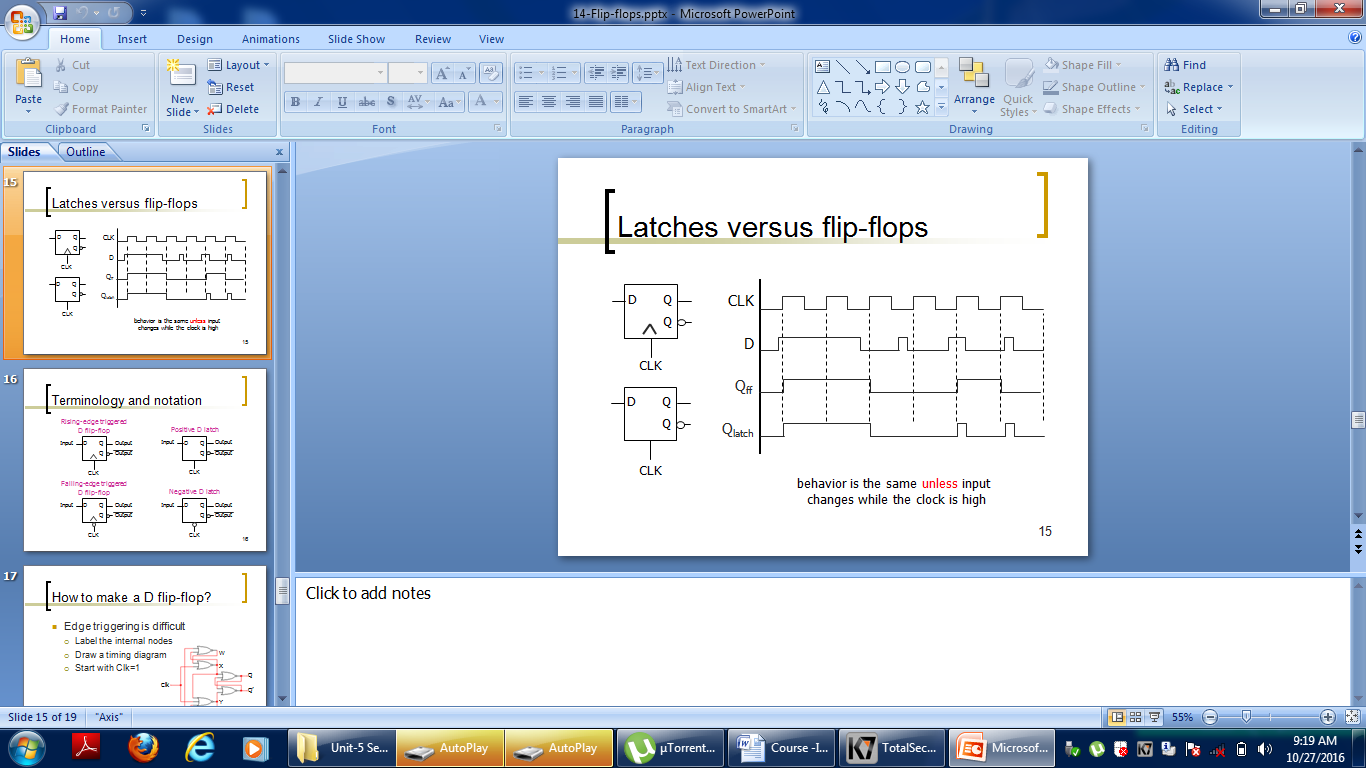
A flip-flop is a one-bit memory. As long as power is available, the flip-flop retains the bit. However, its output (stored bit) can be changed by a clock input. The basic type of flip-flop is called a *latch* which uses a level-sensitive clock. Flip-flops are designed using the latches. The most common latch is the SR (Set-Reset) latch. A flip-flop is a latch with an edge-trigerred clock input. Synchronous sequential circuits are designed using flip-flops whereas asynchronous sequential circuits are designed using latches.

*Latches* and *flip-flops* are the basic elements for storing information. One latch or flip-flop can store one bit of information. The main difference between latches and flip-flops is that for latches, their outputs are constantly affected by their inputs as long as the enable signal is asserted. In other words, when they are enabled, their content changes immediately when their inputs change. Flip-flops, on the other hand, have their content change only either at the rising or falling edge of the enable signal. This enable signal is usually the controlling clock signal. After the rising or falling edge of the clock, the flip-flop content remains constant even if the input changes. There are basically four main types of latches and flip-flops: SR, D, JK, and T. The major differences in these flip-flop types are the number of inputs they have and how they change state. For each type, there are also different variations that enhance their operations. In this chapter, we will look at the operations of the various latches and flip-flops.

Using this terminology, a latch is level-sensitive, whereas a flip-flop is edge-sensitive. That is, when a latch is enabled it becomes transparent, while a flip flop's output only changes on a single type (positive going or negative going) of clock edge.

Thus, a *flip-flop* is a special type of gated latch. The difference between a flip-flop and a gated latch is that in a flip-flop, the inputs aren’t enabled merely by the presence of a HIGH signal on the CLOCK input. Instead, the inputs are enabled by the *transition* of the CLOCK input. Thus, at the moment that the clock input transitions from low to high, the inputs are briefly enabled. Once the clock stabilizes at the HIGH setting, the output state of the flip-flop is latched until the next clock pulse.

Flip-flops are often said to be *edge-triggered* because it’s the edge of the clock signal that triggers the flip-flop. When used in clock-driven computer circuits, edge-triggering is an important characteristic because it helps circuit designers maintain better control over the timing in circuits that contain hundreds or perhaps thousands of flip-flops.



## Types of flip –flops

There are basically four main types of flip-flops: SR, D, JK, and T. The major differences in these flip-flop types are in the number of inputs they have and how they change state. Each type can have different variations such as active high or low inputs, whether they change state at the rising or falling edge of the clock signal, and whether they have asynchronous inputs or not.

## SR Flip-Flop

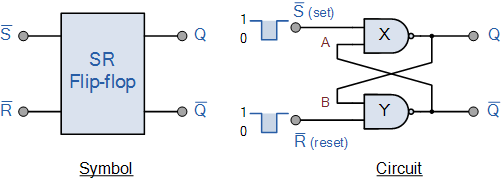
The **SR flip-flop** is also known SR Latch. It is one of the most basic sequential logic circuits. This simple flip-flop is basically a one-bit memory bistable device that has two inputs, (1) which will “SET” the device (meaning the output = “1”), and is labelled S and (2)another which will “RESET” the device (meaning the output = “0”), labelled R.

The SR description stands for “Set-Reset”. The reset input resets the flip-flop back to its original state with an output Q that will be either at a logic level “1” or logic “0” depending upon this set/reset condition.

A basic NAND gate SR flip-flop circuit provides feedback from both of its outputs back to its opposing inputs and is commonly used in memory circuits to store a single data bit. Then the SR flip-flop actually has three inputs, Set, Reset and its current output Q relating to its current state or history. The term “Flip-flop” relates to the actual operation of the device, as it can be “flipped” into one logic Set state or “flopped” back into the opposing logic Reset state.

## The NAND Gate SR Flip-Flop

The simplest way to make any basic single bit set-reset SR flip-flop is to connect together a pair of cross-coupled 2-input NAND gates as shown, to form a Set-Reset Bistable also known as an active LOW SR NAND Gate Latch, so that there is feedback from each output to one of the other NAND gate inputs. This device consists of two inputs, one called the Set, S and the other called the Reset, R with two corresponding outputs Q and its inverse or complement Q (not-Q) as shown below.The Basic SR Flip-flop



### The Set State

If the input R is at logic level “0” (R = 0) and input S is at logic level “1” (S = 1), the NAND gate Y  has at least one of its inputs at logic “0” therefore, its output Q must be at a logic level “1” (NAND Gate principles). Output Q is also fed back to input “A” and so both inputs to NAND gate X are at logic level “1”, and therefore its output Q must be at logic level “0”.

Again NAND gate principals. If the reset input R changes state, and goes HIGH to logic “1” with S remaining HIGH also at logic level “1”, NAND gate Y inputs are now R = “1” and B = “0”. Since one of its inputs is still at logic level “0” the output at Q still remains HIGH at logic level “1” and there is no change of state. Therefore, the flip-flop circuit is said to be “Latched” or “Set” with Q = “1” and Q = “0”.

### Reset State

In this second stable state, Q is at logic level “0”, (not Q = “0”) its inverse output at Q is at logic level “1”, (Q = “1”), and is given by R = “1” and S = “0”. As gate X has one of its inputs at logic “0” its output Q must equal logic level “1” (again NAND gate principles). Output Q is fed back to input “B”, so both inputs to NAND gate Y are at logic “1”, therefore, Q = “0”.

If the set input, S now changes state to logic “1” with input R remaining at logic “1”, output Q still remains LOW at logic level “0” and there is no change of state. Therefore, the flip-flop circuits “Reset” state has also been latched and we can define this “set/reset” action in the following truth table.

### Truth Table for this Set-Reset Function

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | State | S | R | Q | Q | Description | | Set | 1 | 0 | 0 | 1 | Set Q = 1 | | 1 | 1 | 0 | 1 | no change | | Reset | 0 | 1 | 1 | 0 | Reset Q = 0 | | 1 | 1 | 1 | 0 | no change | | Invalid | 0 | 0 | 1 | 1 | Invalid Condition | |

It can be seen that when both inputs S = “1” and R = “1” the outputs Q and Q can be at either logic level “1” or “0”, depending upon the state of the inputs S or R BEFORE this input condition existed. Therefore the condition of S = R = “1” does not change the state of the outputs Q and Q.

However, the input state of S = “0” and R = “0” is an undesirable or invalid condition and must be avoided. The condition of S = R = “0” causes both outputs Q and Q to be HIGH together at logic level “1” when we would normally want Q to be the inverse of Q. The result is that the flip-flop looses control of Q and Q, and if the two inputs are now switched “HIGH” again after this condition to logic “1”, the flip-flop becomes unstable and switches to an unknown data state based upon the unbalance as shown in the following switching diagram.

1. **JK Flip-flops:**

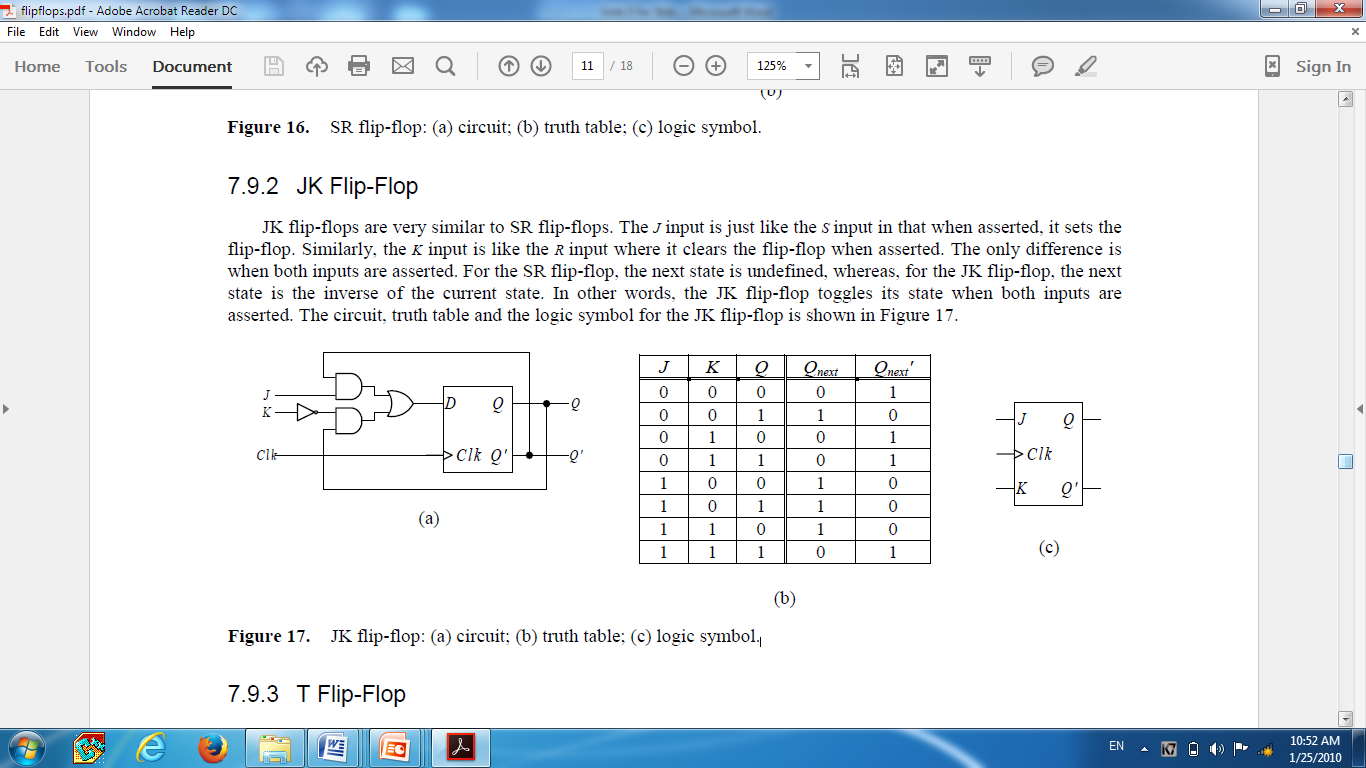
JK flip-flops are very similar to SR flip-flops. The JK Flip-flop is a variation or modification of the SR Flip-flop. The disadvantage of the SR flip-flop is that both inputs shouldn't be HIGH when the clock is triggered. This is considered an invalid input condition, and the resulting output isn't predictable if this condition occurs.

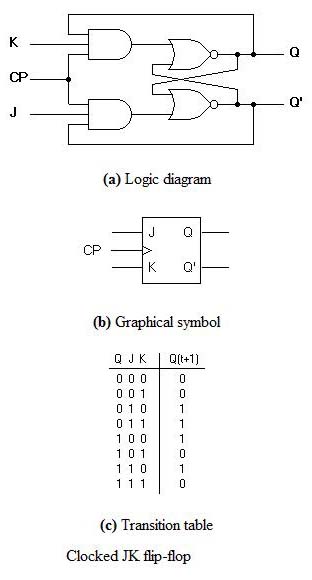
The main difference between a JK flip-flop and an SR flip-flop is that **in the JK flip-flop, both inputs can be HIGH**. When both the J and K inputs are HIGH, the Q output is *toggled*, which means that the output alternates between HIGH and LOW. Thereby the invalid condition which occurs in the SR flip-flop is eliminated.

The *J* input is just like the *S* input in that when asserted, it sets the flip-flop. Similarly, the *K* input is like the *R* input where it clears the flip-flop when asserted. The only difference is when both inputs are asserted. For the SR flip-flop, the next state is undefined, whereas, for the JK flip-flop, the next state is the inverse of the current state. In other words, the JK flip-flop toggles its state when both inputs are asserted. The circuit, truth table and the logic symbol for the JK flip-flop is shown in the figure.

The inputs J and K are same as the S and R inputs of the S-R flip flop. The letter J stands for SET and the letter K stands for CLEAR (RESET). When both the inputs J and K have a HIGH state, the flip-flop switch to the complement state. So, for a value of Q = 1, it switches to Q=0 and for a value of Q = 0, it switches to Q=1.

**Study the following figure:**

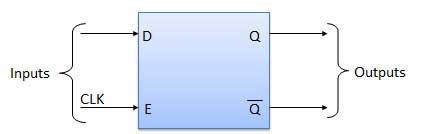


Also study another diagram: 

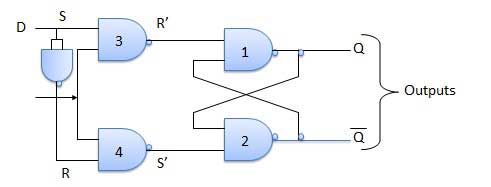
**Delay Flip Flop / D Flip Flop**

Delay Flip Flop or D Flip Flop is the simple gated S-R latch with a NAND **inverter connected between S and R inputs**. It has only one input. The input data is appearing at the output after some time. Due to this data delay between i/p and o/p, it is called delay flip flop. S and R will be the complements of each other due to NAND inverter. Hence S = R = 0 or S = R = 1, these input condition will never appear. This problem is avoid by SR = 00 and SR = 1conditions.

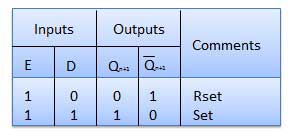
**Block diagram**



**Circuit diagram**



Truth Table



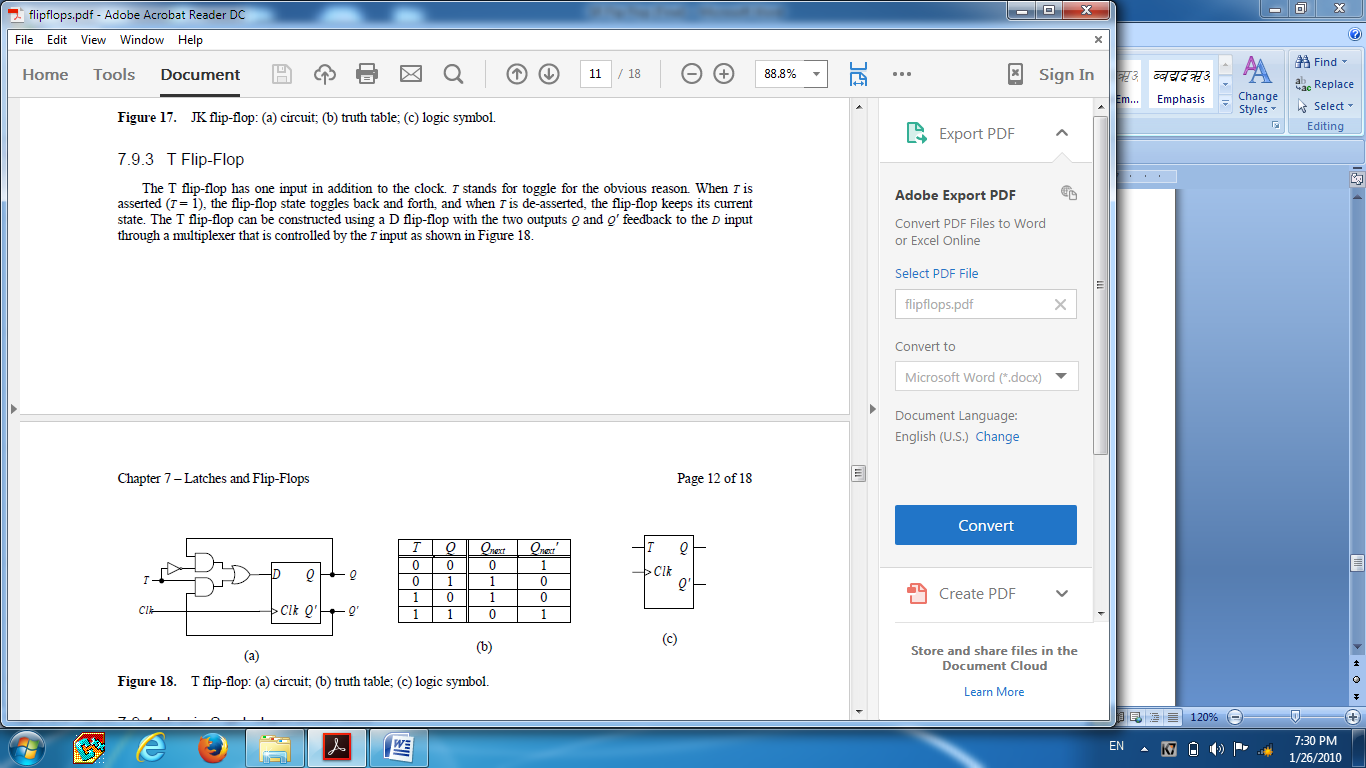
**Operation**

|  |  |  |
| --- | --- | --- |
| **S.N.** | **Condition** | **Operation** |
| 1 | **E = 0** | Latch is disabled. Hence no change in output. |
| 2 | **E = 1 and D = 0** | If E = 1 and D = 0 then S = 0 and R = 1. Hence irrespective of the present state, the next state is Qn+1 = 0 and Qn+1 bar = 1. This is the reset condition. |
| 3 | **E = 1 and D = 1** | If E = 1 and D = 1, then S = 1 and R = 0. This will set the latch and Qn+1 = 1 and Qn+1 bar = 0 irrespective of the present state. |

**c) T-flip-flop/ Toggle flip-flop**

Toggle flip flop is basically a JK flip flop with J and K terminals permanently connected together. The T (Toggle) flip-flop complements its output when the clock input is applied with *T* = 1; the output remains **unchanged** when *T* = 0. The name “toggle” is based on the fact that the T flip-flop toggles or complements its output when the clock input is 1 with *T* = 1.

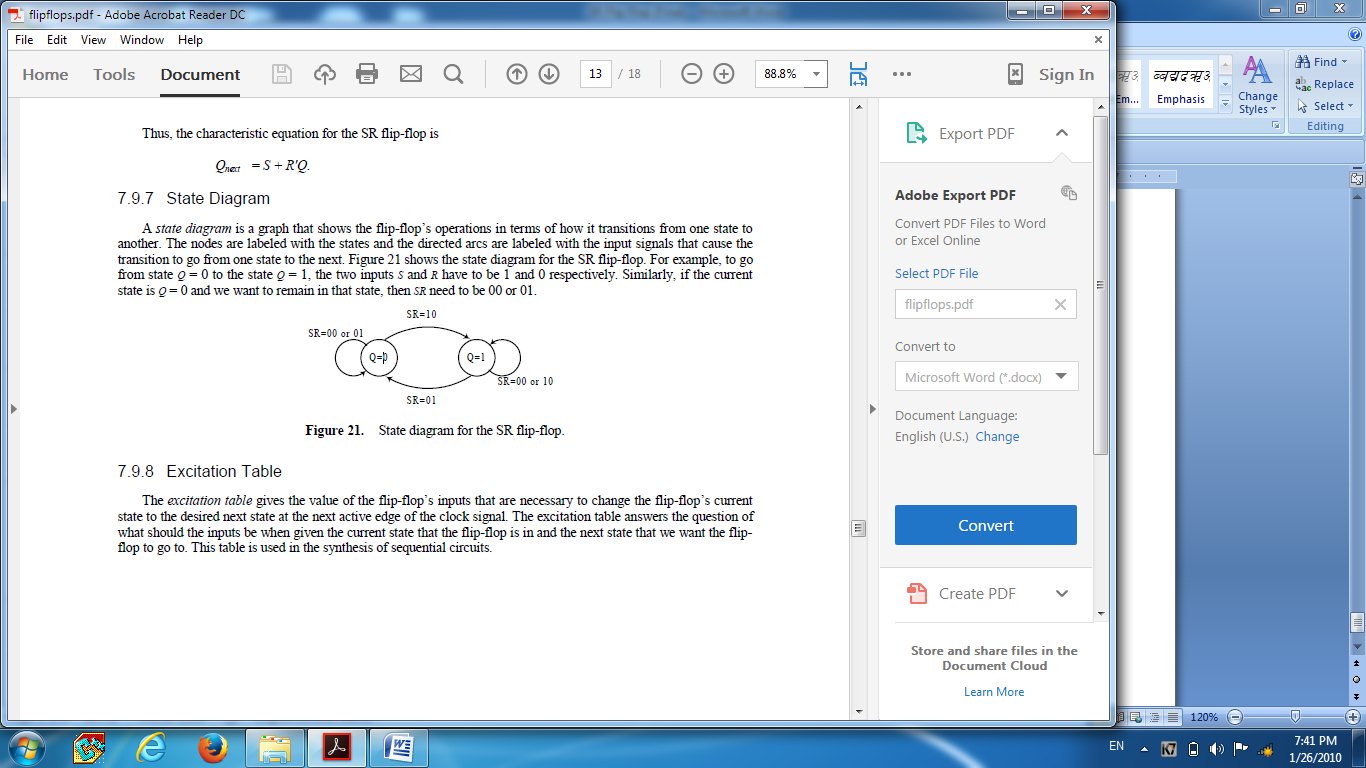
The T flip-flop has one input in addition to the clock. *T* stands for toggle. When *T* is asserted (*T* = 1), the flip-flop state toggles back and forth, and when *T* is de-asserted, the flip-flop keeps its current state. The T flip-flop can be constructed using a D flip-flop with the two outputs *Q* and *Q'* feedback to the *D* input through a multiplexer that is controlled by the *T* input as shown in the figure.



1. Circuit (b) Truth table (c) Logic Symbol

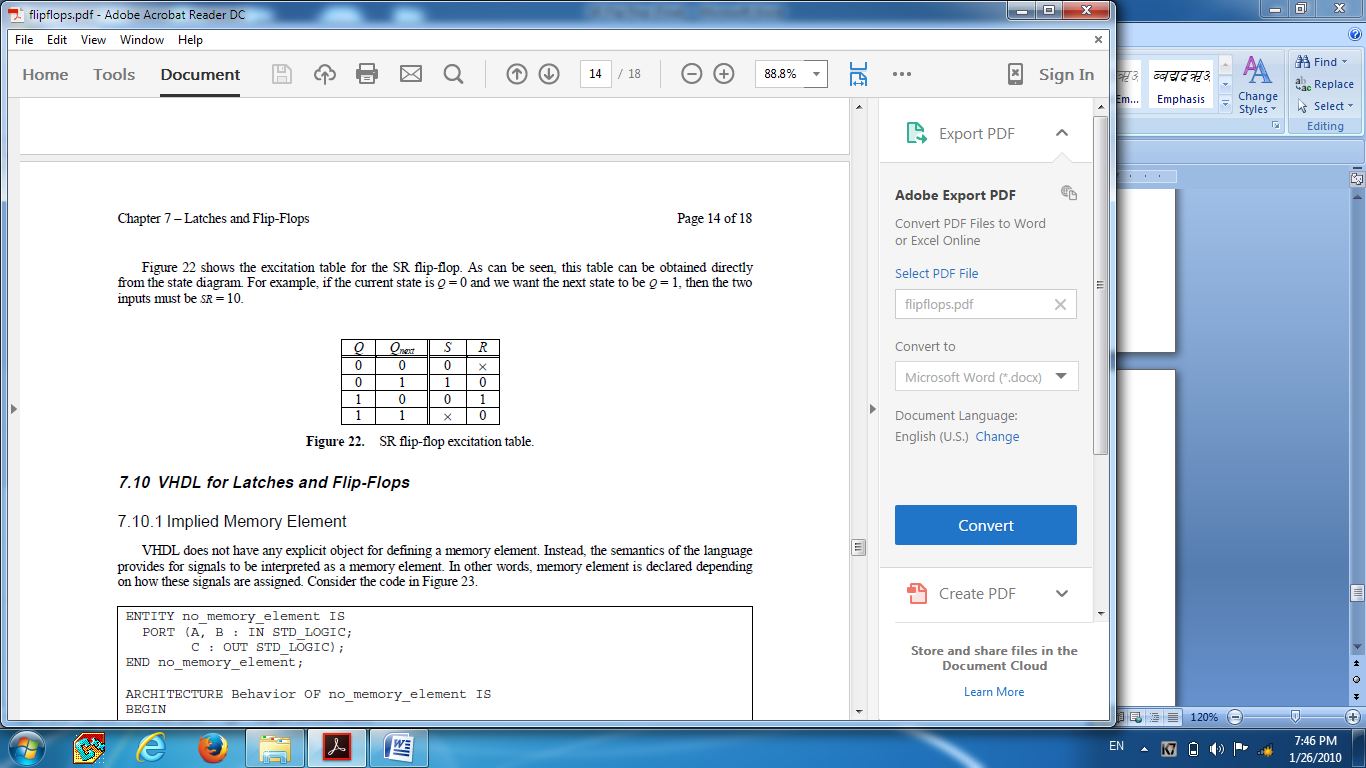
**State Diagram**

A *state diagram* is a graph that shows the flip-flop’s operations in terms of how it transitions from one state to another. The nodes are labeled with the states and the directed arcs are labeled with the input signals that cause the transition to go from one state to the next. Figure 21 shows the state diagram for the SR flip-flop. For example, to go from state *Q* = 0 to the state *Q* = 1, the two inputs *S* and *R* have to be 1 and 0 respectively. Similarly, if the current state is *Q* = 0 and we want to remain in that state, then *SR* need to be 00 or 01.

 State diagram for SR flip-flop

**Excitation Table**

The *excitation table* gives the value of the flip-flop’s inputs that are necessary to change the flip-flop’s current state to the desired next state at the next active edge of the clock signal. The excitation table answers the question of what should the inputs be when given the current state that the flip-flop is in and the next state that we want the flip-flop to go to. This table is used in the synthesis of sequential circuits. The following figure shows the excitation table for the SR flip-flop. This table can be obtained directly from the state diagram. For example, if the current state is *Q* = 0 and we want the next state to be *Q* = 1, then the two inputs must be *SR* = 10.



**SR flip-flop excitation table**