INTRODUCTION TO SEQUENTIAL LOGIC

Dr. Suneth Pathirana
B.Sc.(UWU-CST)(Hons), M.Sc.(AI)(Moratuwa), MIEEE, Ph.D.(MSU)

Senior Lecturer
Department of Computer Science & Informatics
Uva Wellassa University of Sri Lanka

suneth@uwu.ac.lk

Combinational Logic Vs Sequential Logic

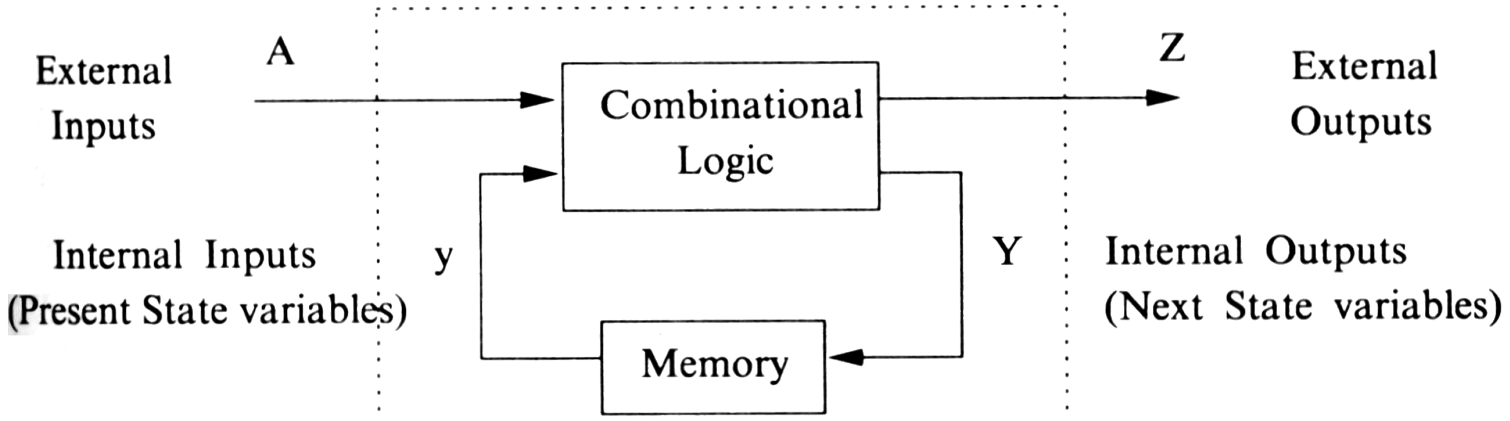
- Combinational logic refers to circuits whose **output is a function of the present value of the inputs only**
- As soon as inputs are changed, the information about the previous inputs is lost, that is, combinational logic circuits have **no memory**

Combinational Logic Vs Sequential Logic

- Sequential logic circuits are those whose **outputs are also dependent upon past inputs**, and hence outputs. In other words the output of a sequential circuit may depend upon its previous outputs and so in effect has some form of "**memory**"
- The mathematical model of a sequential circuit is usually referred to as a sequential machine

General Form of a Sequential Circuit

- Sequential circuits are basically combinational circuits with the additional properties of storage (to remember past inputs) and feedback
- Note that the output Z is a function of inputs x and internal states y

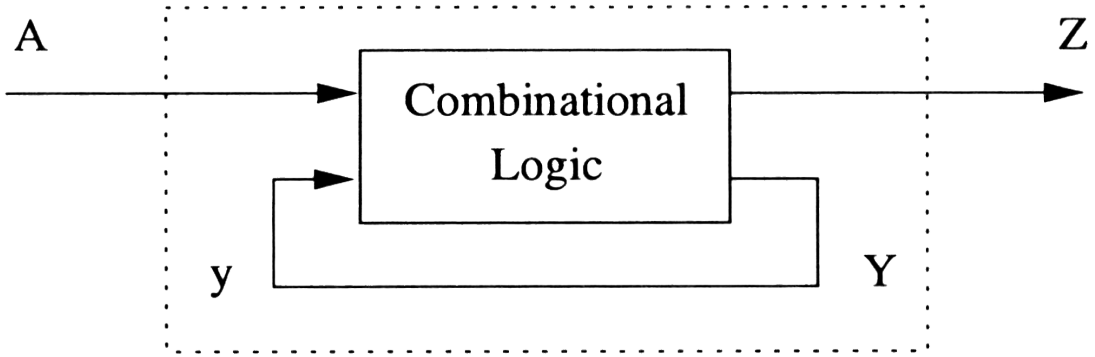


Classification Of Sequential Circuits

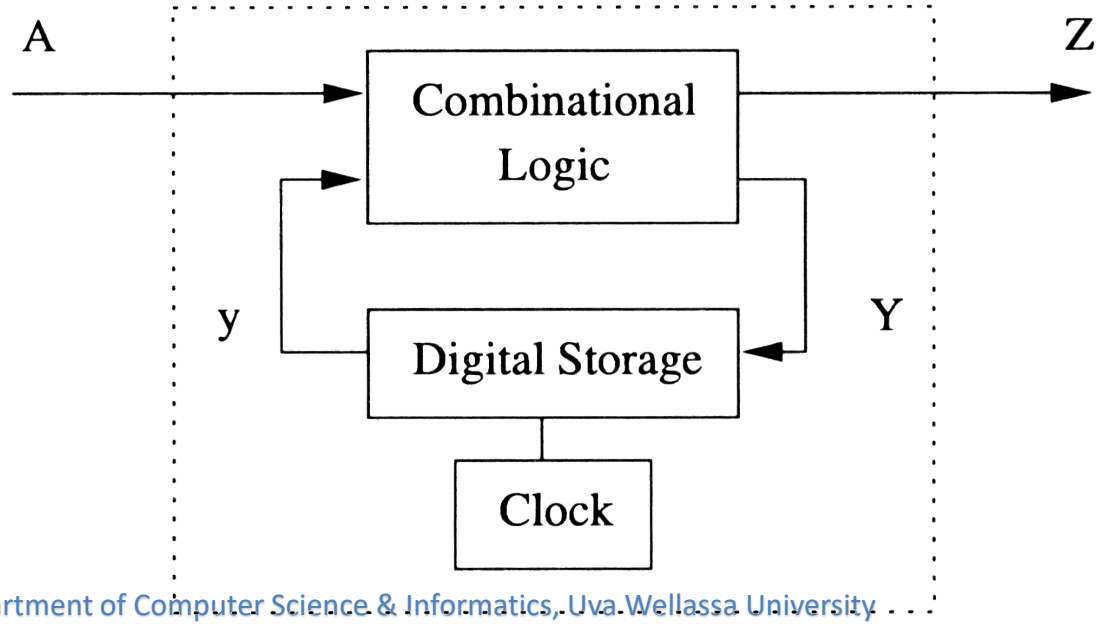
- Sequential circuits fall into two classes:
  - synchronous and
  - asynchronous

Classification Of Sequential Circuits

Asynchronous



Synchronous



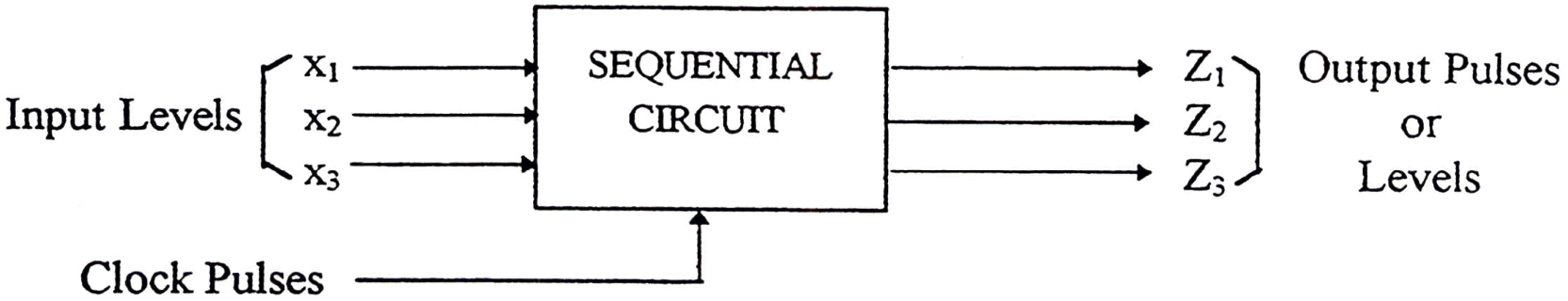
Synchronous circuits

- In synchronous circuits the input are pulses (or levels and pulses) with certain restrictions on pulse width and circuit propagation delay
- Therefore synchronous circuits can be divided into **clocked sequential** circuits and **uncklocked or pulsed** sequential circuits

Synchronous circuits

- In a clocked sequential circuit which has **flip-flops** or, in some instances, gated **latches**, for its memory elements there is a (synchronizing) periodic clock connected to the clock inputs of all the memory elements of the circuit, to synchronize all internal changes of state
- Hence the operation of the entire circuit is controlled and synchronized by the periodic pulses of the clock

Synchronous circuits



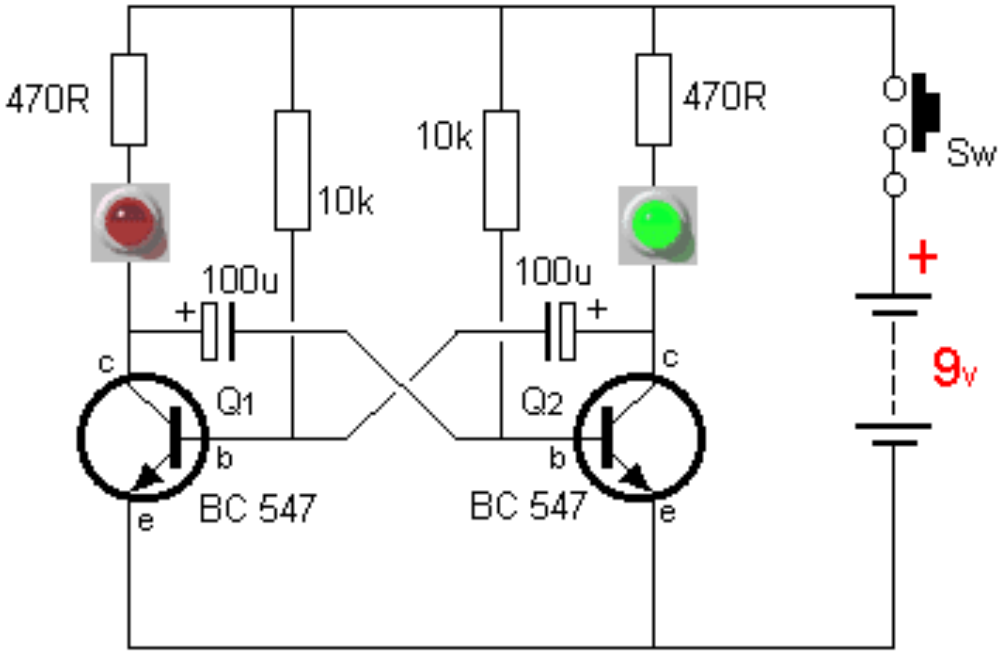
Clocked Sequential Circuit

Asynchronous circuits

- The circuit is considered to be asynchronous if it **does not employ a periodic clock signal** to synchronize its internal changes of state
- Therefore the state changes occur in direct response to signal changes on primary (data) input lines, and different memory elements can change state at different times
- In asynchronous sequential circuits the inputs are levels and there are no clock pulses; the inputs events drive the circuit

Latch

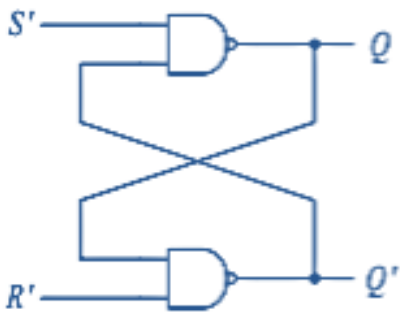
- A latch is a circuit that has two stable states and can be used to store state information



THE FLIP FLOP CIRCUIT IN ACTION

SR Latch

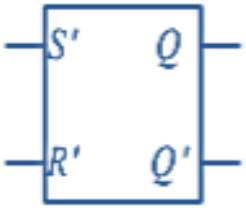
- The bistable element is able to remember or store one bit of information



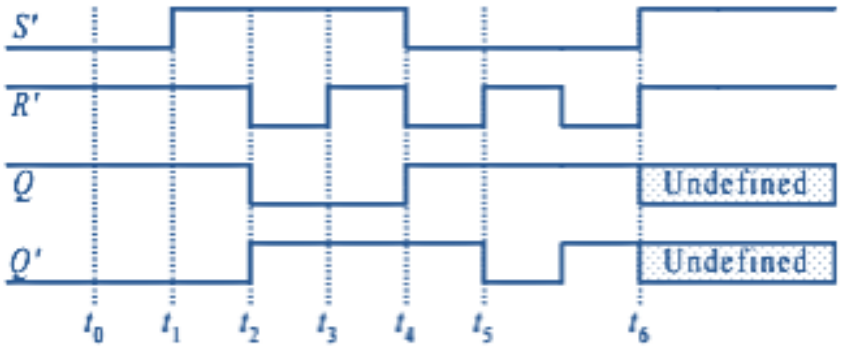
(a)

S	R	Q	Q_{next}	Q_{next}'
0	0	×	1	1
0	1	×	1	0
1	0	×	0	1
1	1	0	0	1
1	1	1	1	0

(b)



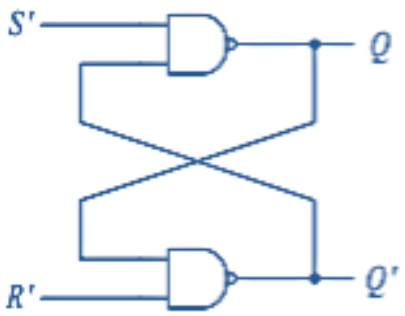
(c)



(d)

SR Latch

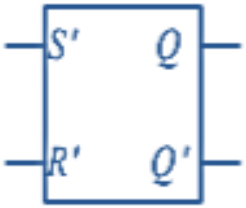
- The bistable element is able to remember or store one bit of information



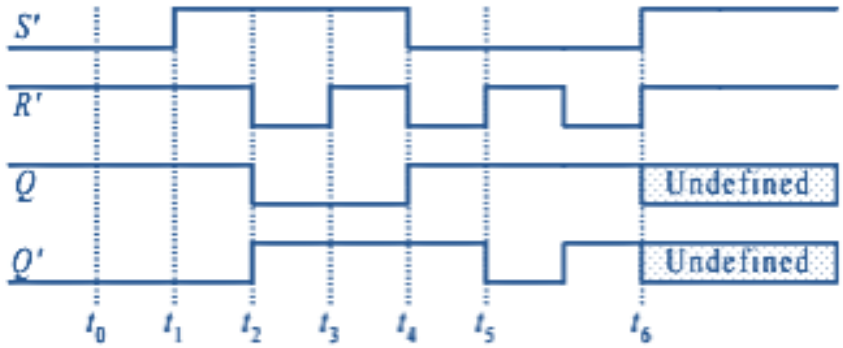
(a)

S	R	S'	R'	Q	Q_{next}	Q_{next}'
0	0	1	1	×	1	1
0	1	1	0	×	1	0
1	0	0	1	×	0	1
1	1	0	0	0	0	1
1	1	0	0	1	1	0

(b)



(c)

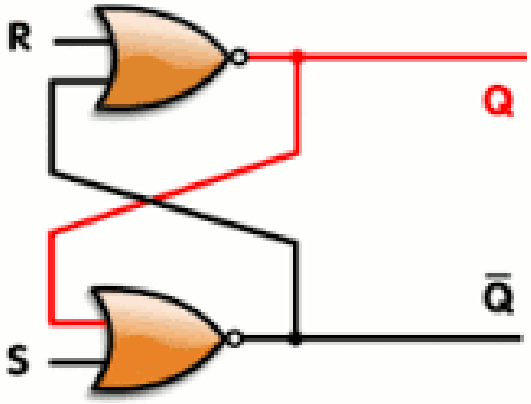


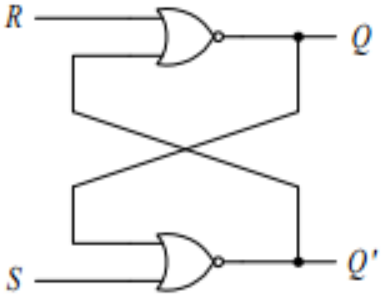
(d)

SR Latch

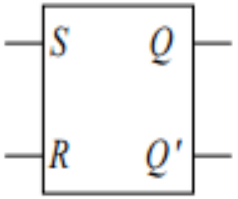
- In order to change the information bit, we need to add inputs to the circuit
- The simplest way to add inputs is to replace the two inverters with two NAND gates
- This circuit is called a SR latch
- In addition to the two outputs Q and Q', there are two inputs S‘ and R' for set and reset respectively
- Following the convention, the prime in S and R denotes that these inputs are active low
- **The SR latch can be in one of two states: a set state when Q = 1, or a reset state when Q = 0**

SR latch circuit using NOR gates



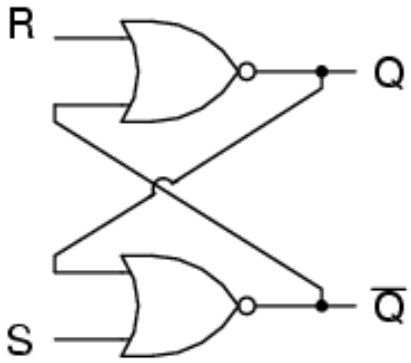


S	R	Q	Q_{next}	Q_{next}'
0	0	0	0	1
0	0	1	1	0
0	1	×	0	1
1	0	×	1	0
1	1	×	0	0



Latch

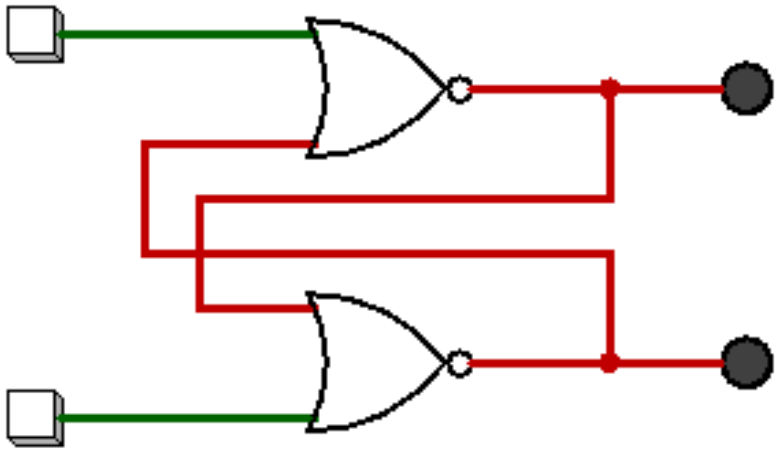
- **Active-high** circuit: Both inputs are normally tied to ground (LOW), and the latch is triggered by a momentary HIGH signal on either of the inputs
- Active-low circuit: Both inputs are normally HIGH, and the latch is triggered by a momentary LOW signal on either input



S	R	Q	$\overline{Q}$
0	0	latch	latch
0	1	0	1
1	0	1	0
1	1	0	0

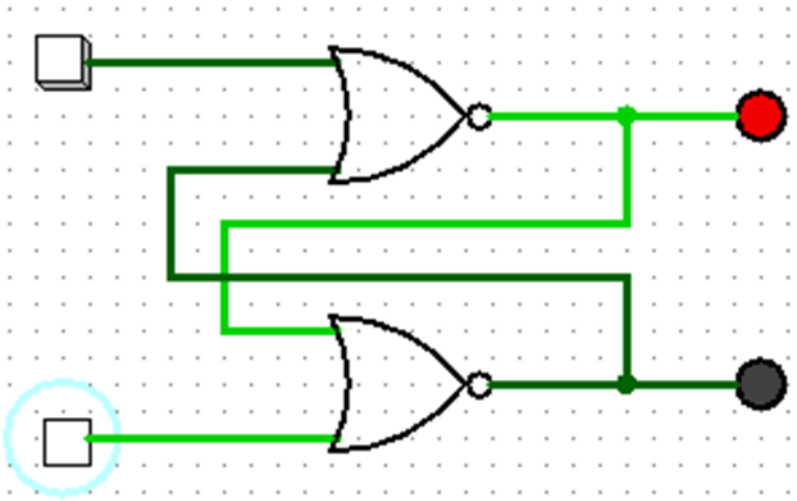
Latch

- The output of the active-high latch stays HIGH until the RESET input goes HIGH
- Then, the output returns to LOW and will go HIGH again only when the SET input is triggered once more



Latch

- The output of the active-high latch stays HIGH until the RESET input goes HIGH
- Then, the output returns to LOW and will go HIGH again only when the SET input is triggered once more



Latch

- In other words, the latch remembers that the SET input has been activated
- If the SET input goes HIGH for even a moment, the output goes HIGH and stays HIGH, even after the SET input returns to LOW. The output returns to LOW only when the RESET input goes HIGH

Latch

- On the other hand, in an active-low latch the inputs are normally held at HIGH
- When the SET input momentarily goes LOW, the output goes HIGH. The output then stays HIGH until the RESET input momentarily goes LOW

Latch

- Note that most latch circuits actually have a second output that is simply the first output inverted
- In other words, whenever the first output is HIGH, the second output is LOW, and vice versa. These outputs are usually referred to as Q and Q-bar with the latter notated as follows

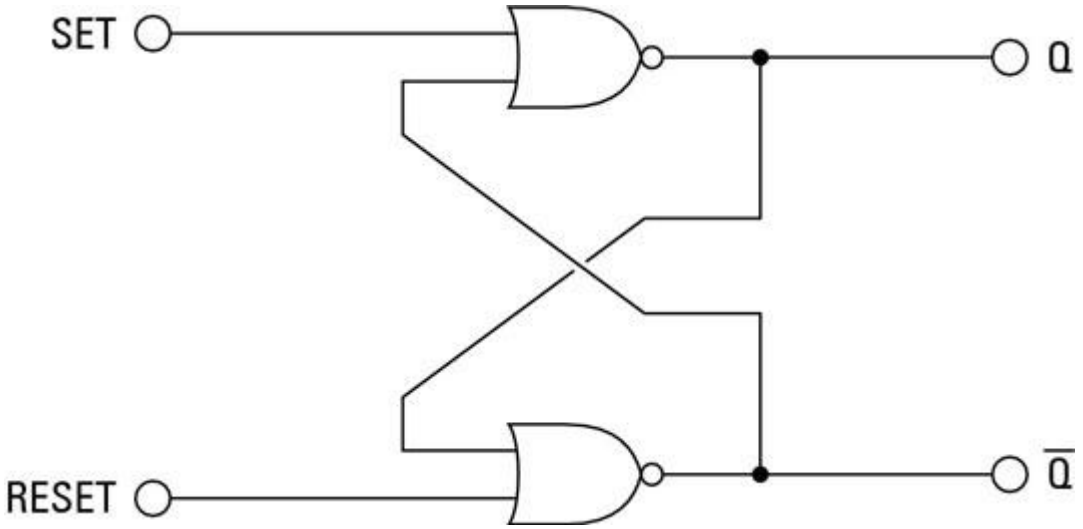
$\overline{Q}$

Latch

- The notation is usually pronounced either “bar Q” or “Q bar,” though some people pronounce it “not Q.”
- The horizontal bar symbol over a label is a common logical shorthand for inversion. That is, Q-bar is the inverse of Q. If Q is HIGH, Q-bar is LOW, and if Q is LOW, Q-bar is HIGH
- You can easily create an active-high latch from a pair of NOR gates
- (The output of a NOR gate is HIGH if both inputs are LOW; otherwise, the output is LOW.)
- In this circuit, the SET input is connected to one of the inputs of the first NOR gate, and the RESET input is connected to one of the inputs of the second NOR gate

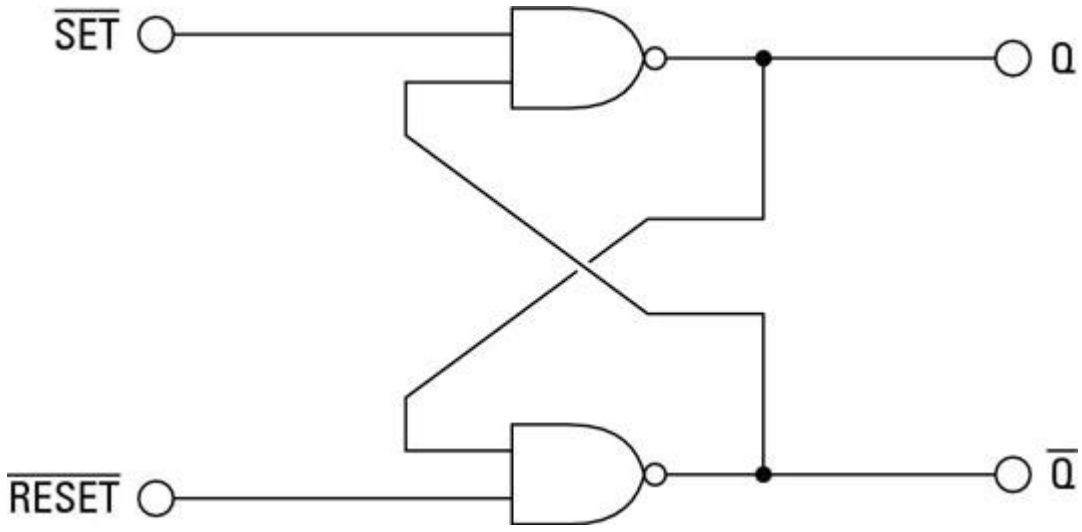
Latch

- The trick of the latch circuit is that the output of the NOR gates are cross-connected to the remaining NOR gate inputs
- In other words, the output from the first NOR gate is connected to one of the inputs of the second NOR gate, and the output from the second NOR gate is connected to one of the inputs of the first NOR gate

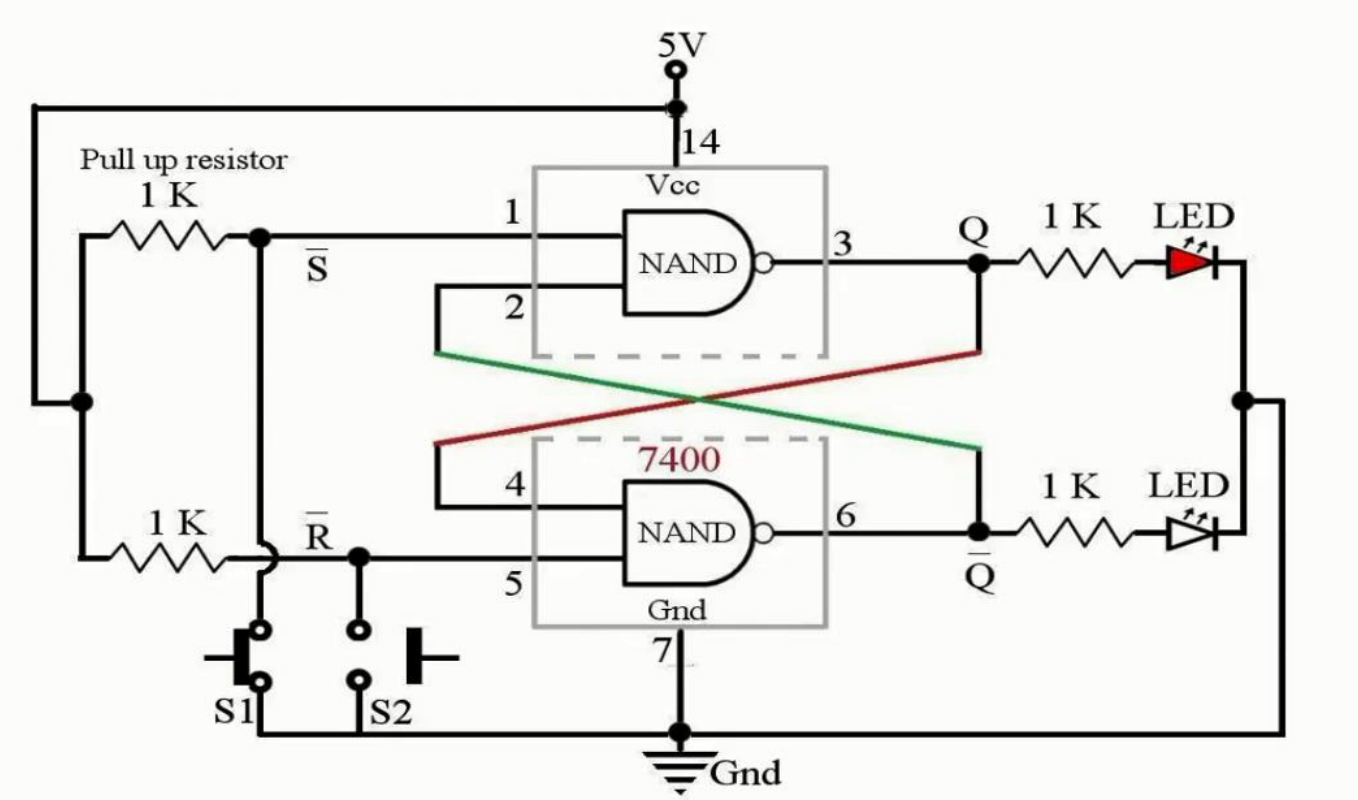


Latch

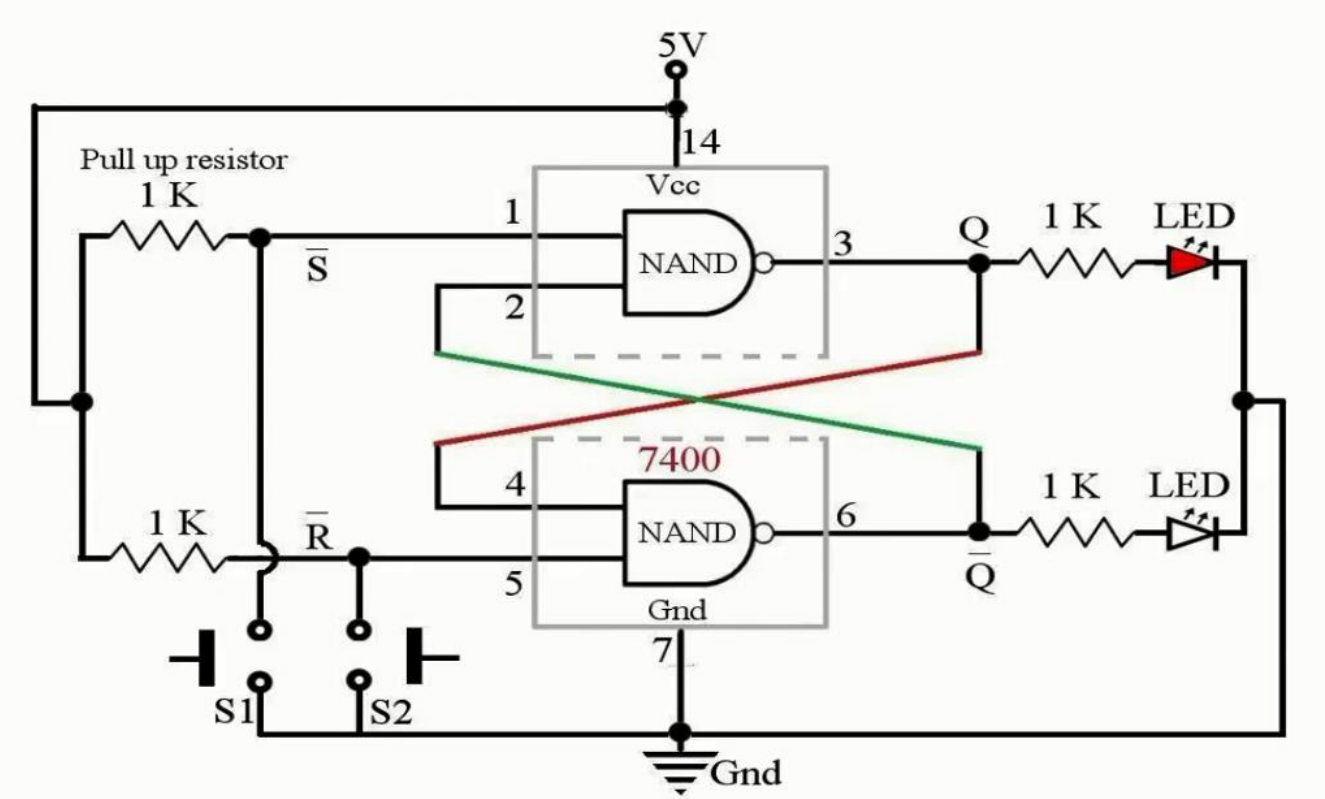
- **Active-low** latch
- The only difference between this schematic and the one shown previously is that the active-low latch uses NAND gates instead of NOR gates
- Notice also in this diagram that the inputs are referred to as SET-bar and RESET-bar rather than SET and RESET, which indicates that the inputs are active-low



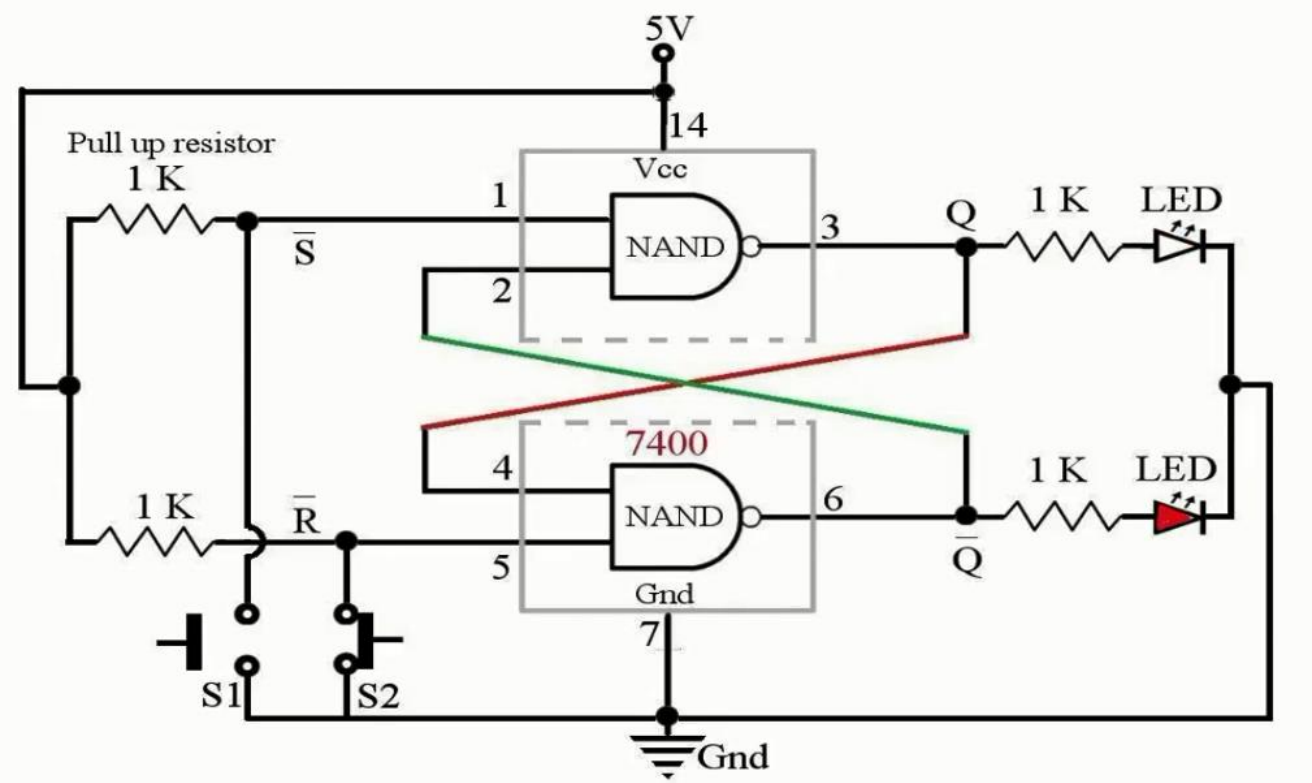
How a Latch Works



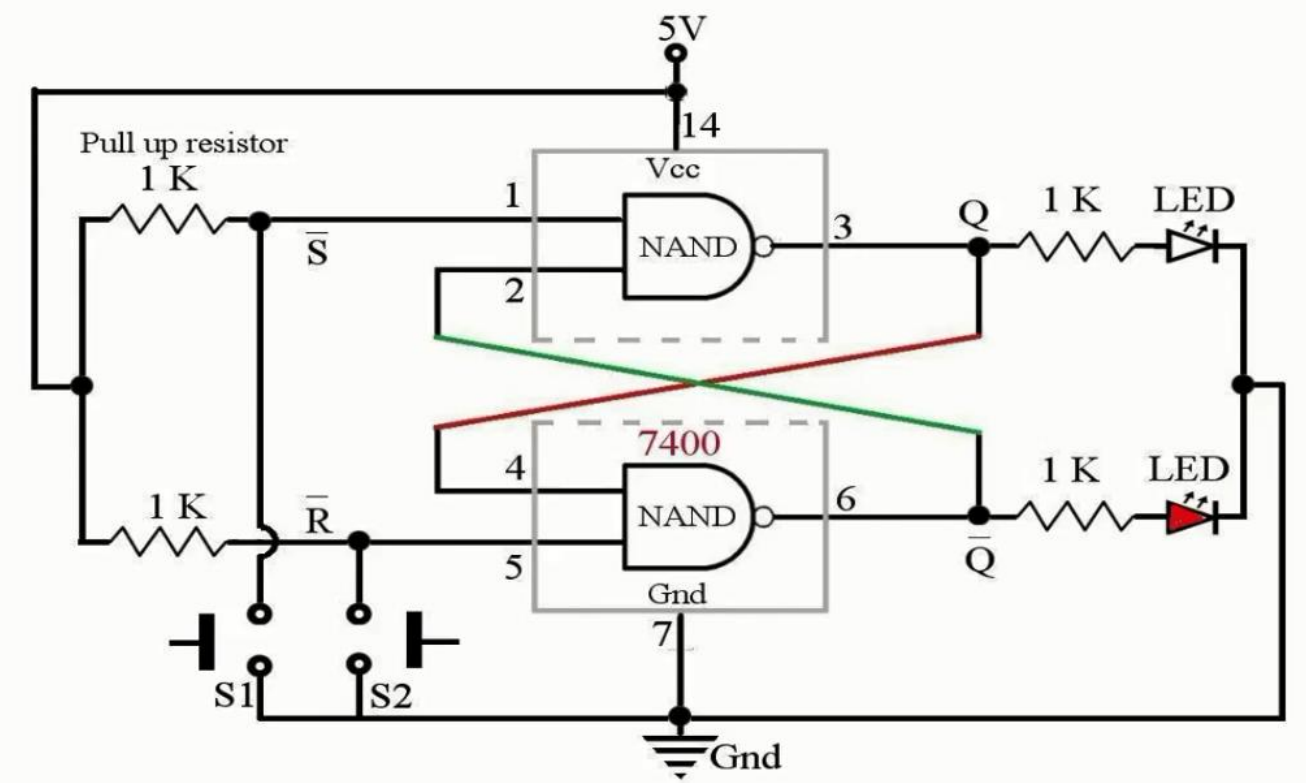
How a Latch Works



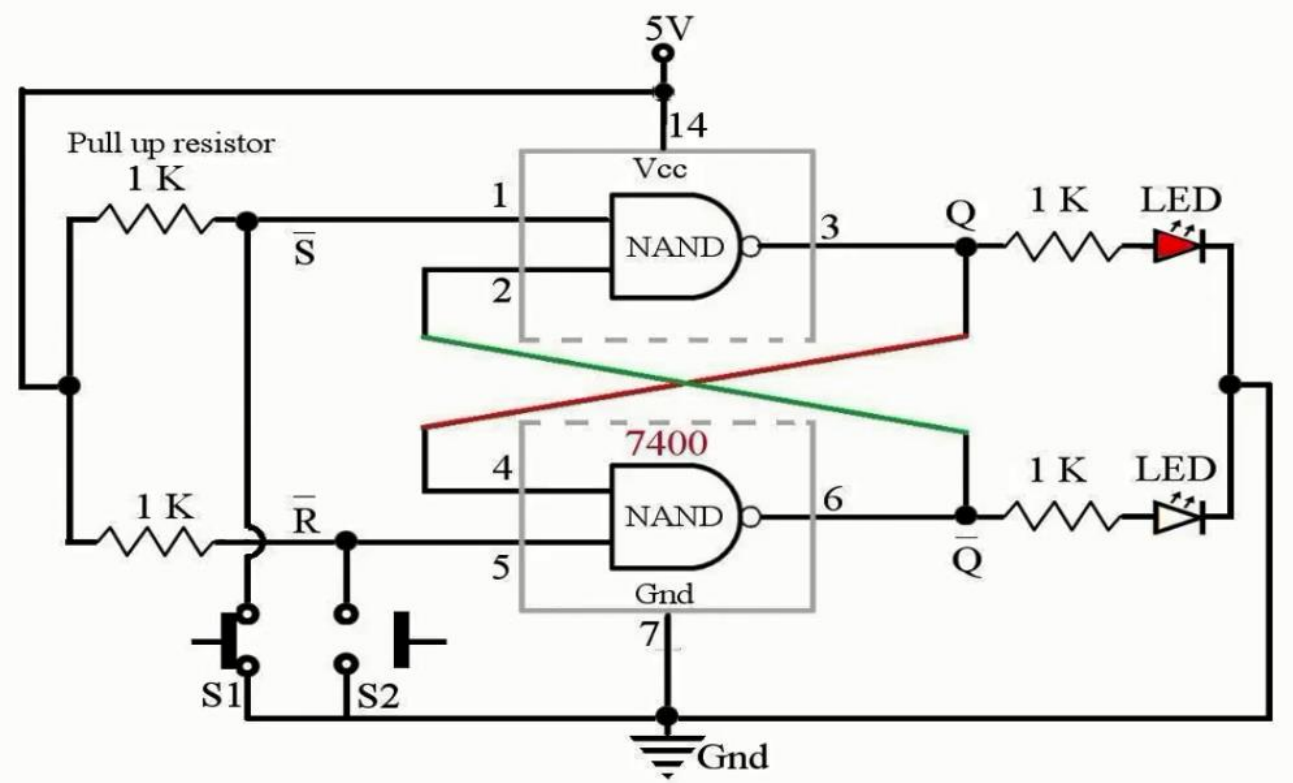
How a Latch Works



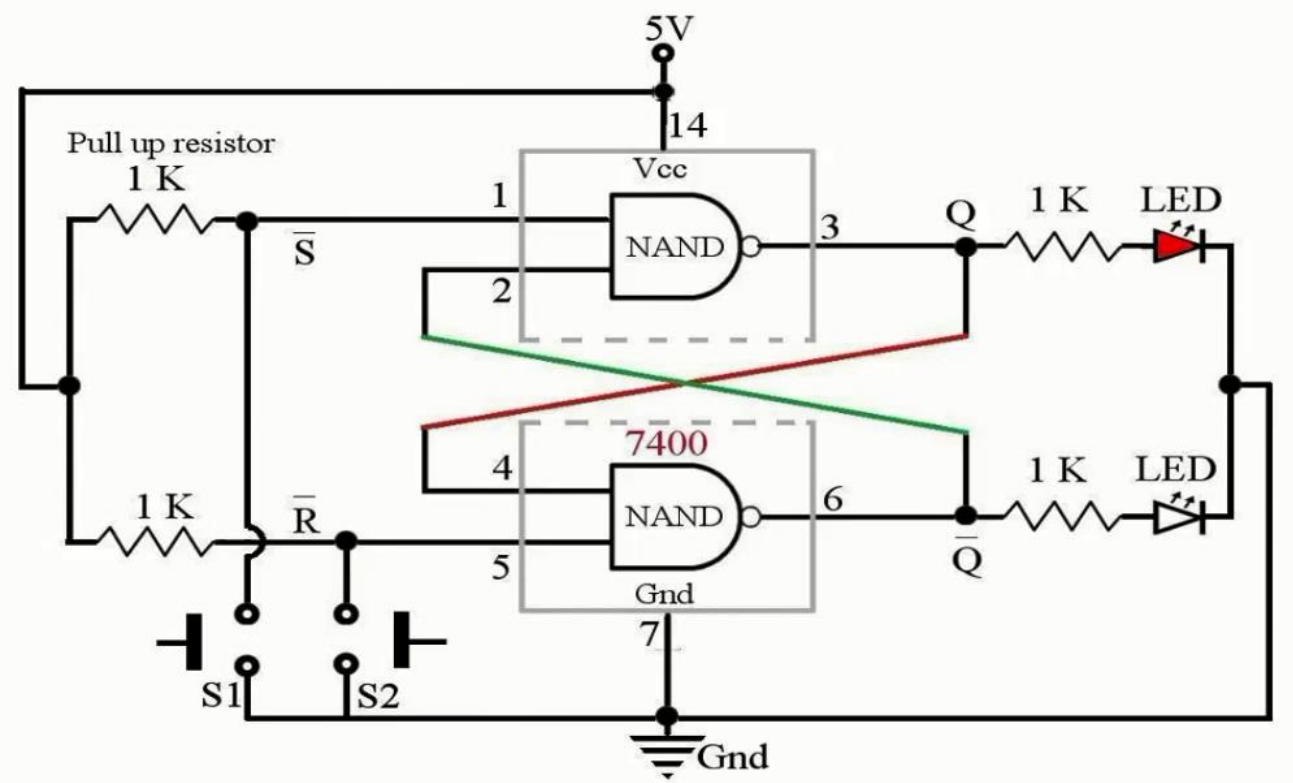
How a Latch Works



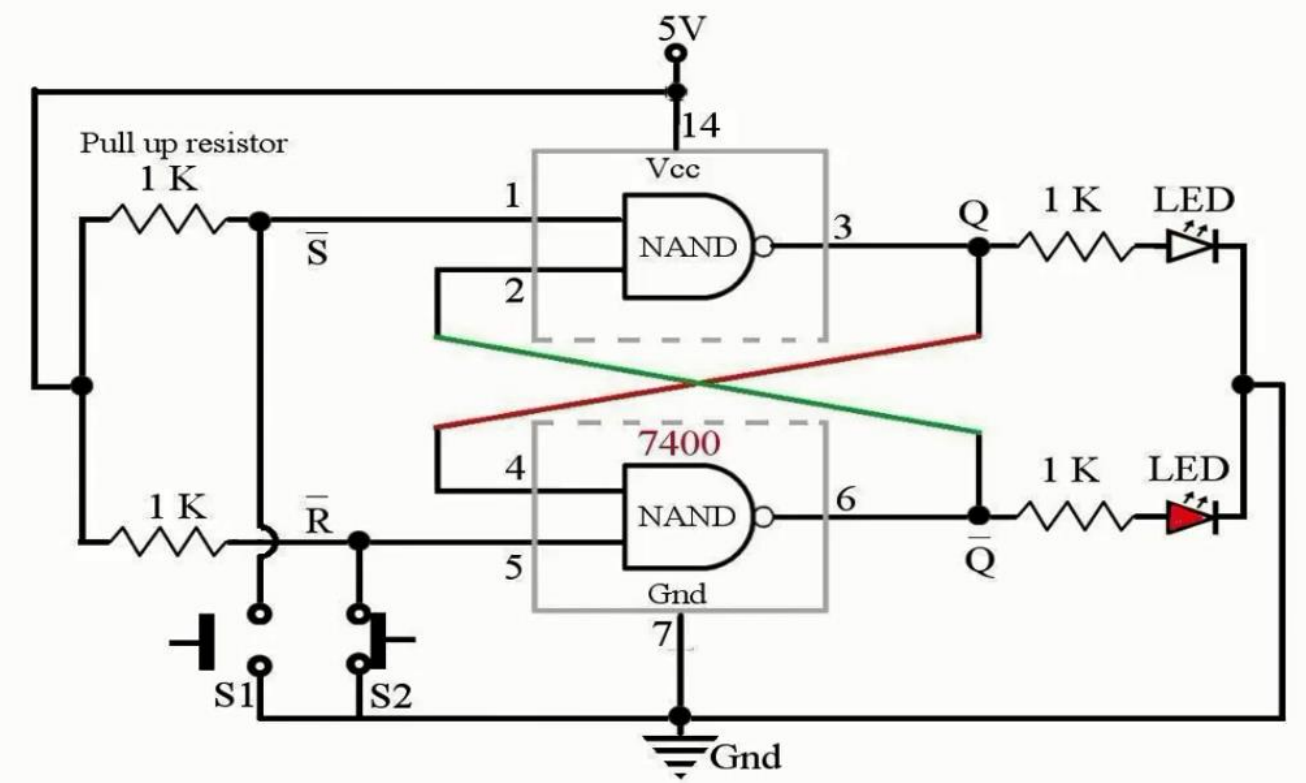
How a Latch Works



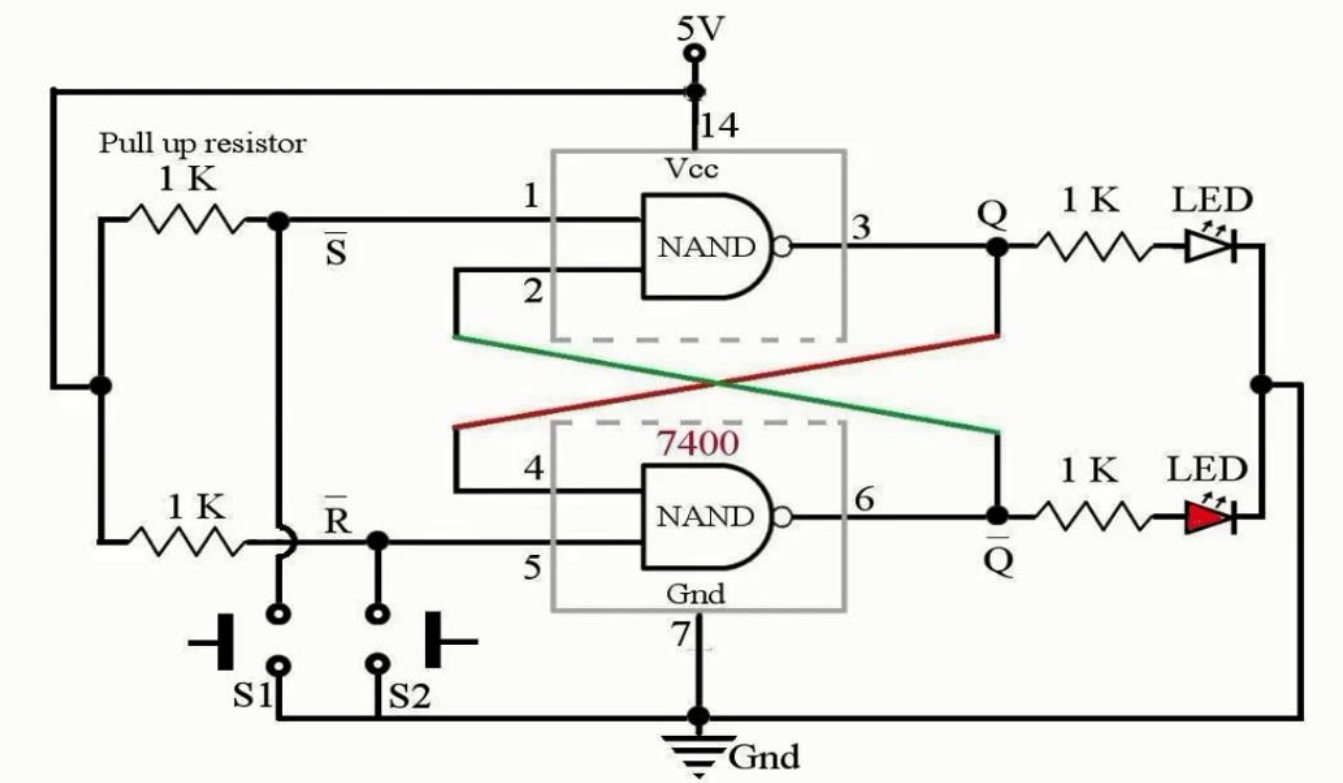
How a Latch Works



How a Latch Works

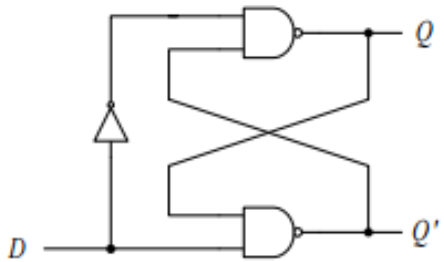


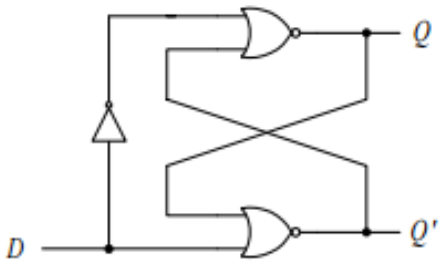
How a Latch Works



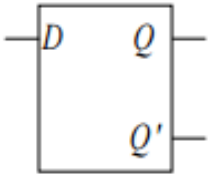
D Latch

D	Q	Q_{next}	Q_{next}'
0	×	0	1
1	×	1	0



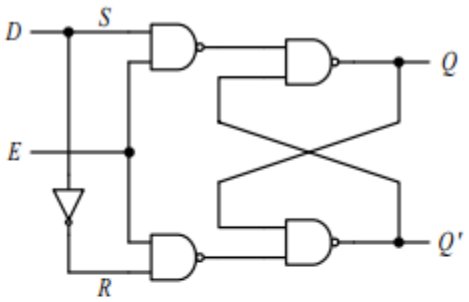


(c)



- The disadvantage with the SR latch is that we need to ensure that the two inputs, S and R, are never de-asserted at the same time
- This situation is prevented in the D latch by adding an inverter between the original S and R inputs and replacing them with just one input D (for data)

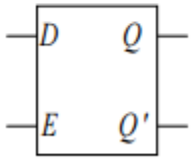
D Latch with Enable



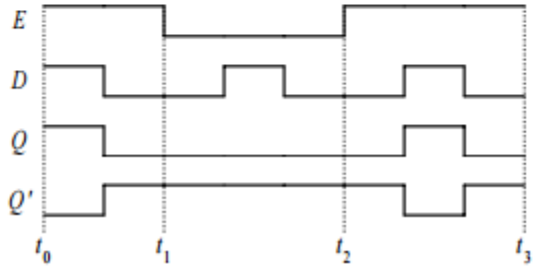
(a)

E	D	Q	Q_{next}	Q_{next}'
0	×	0	0	1
0	×	1	1	0
1	0	×	0	1
1	1	×	1	0

(b)



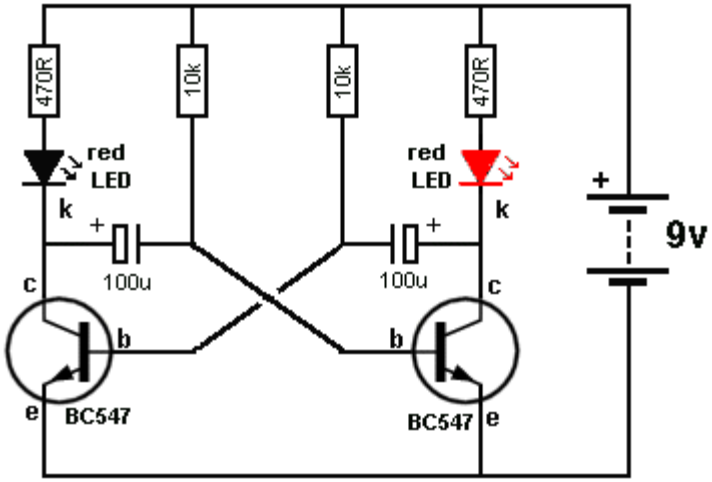
(c)



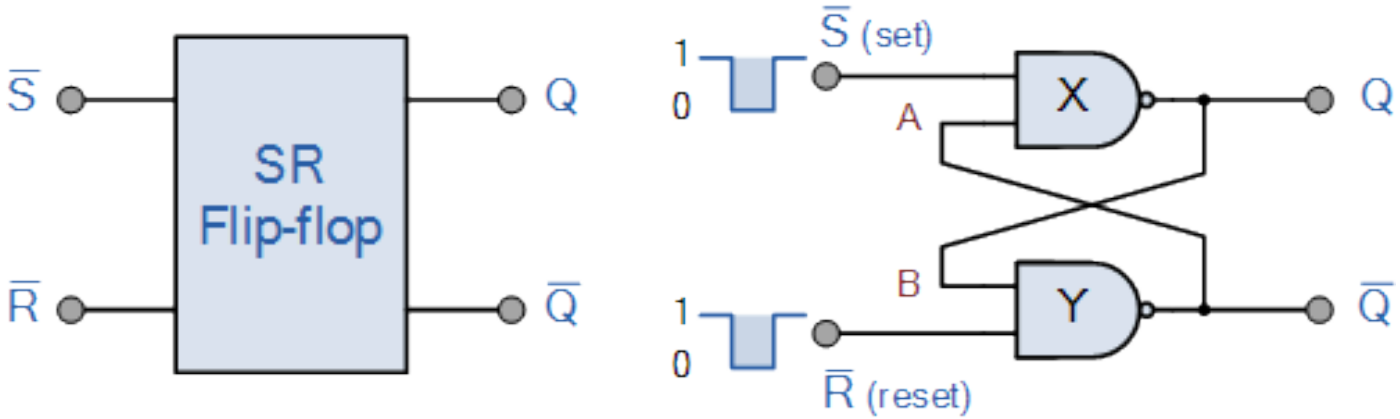
- Notice that the placement of the inverter with respect to the Q output is such that the Q output value follows the D input
- This feature is useful because, whereas the SR latch is useful for setting or resetting a flag on a given condition, the D latch is useful for simply storing a bit of information that is presented on a line

Flip-Flop

- A flip-flop is a bistable multivibrator
- The circuit can be made to change state by signals applied to one or more control inputs and will have one or two outputs



The Basic SR Flip-flop



Symbol Circuit

- 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 $\overline{Q}$ (not-Q)

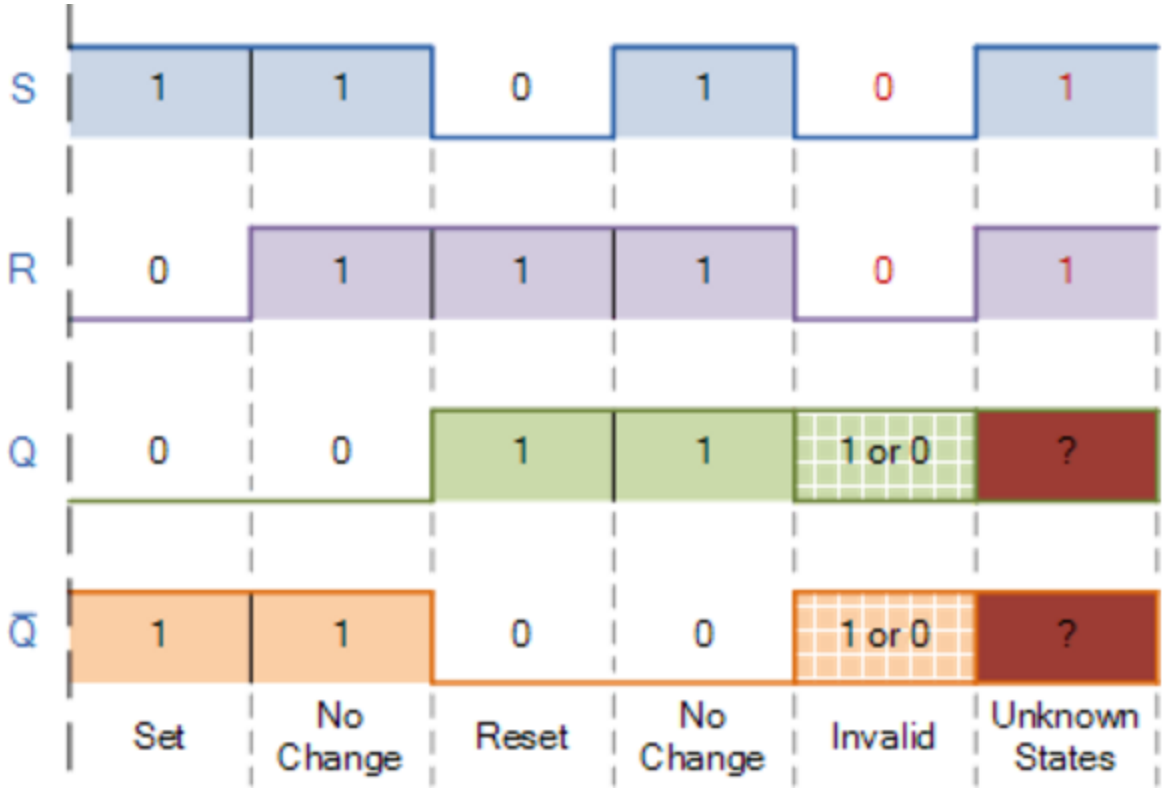
The Basic SR Flip-flop

- Truth Table for this Set-Reset Function

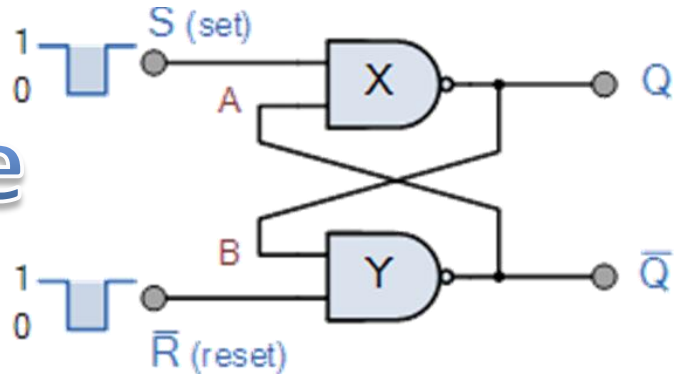
State	S	R	Q	$\overline{Q}$	Description
Set	1	0	0	1	Set $\overline{Q}$ » 1
	1	1	0	1	no change
Reset	0	1	1	0	Reset $\overline{Q}$ » 0
	1	1	1	0	no change
Invalid	0	0	1	1	Invalid Condition

The Basic SR Flip-flop

- S-R Flip-flop Switching Diagram

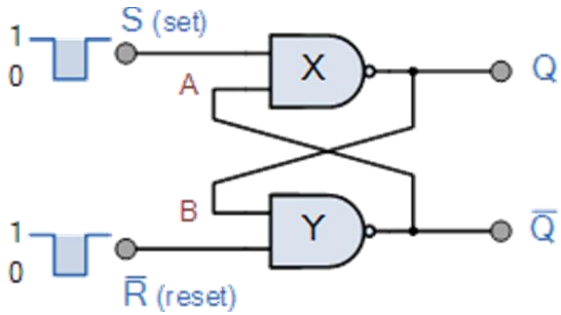


The Set State



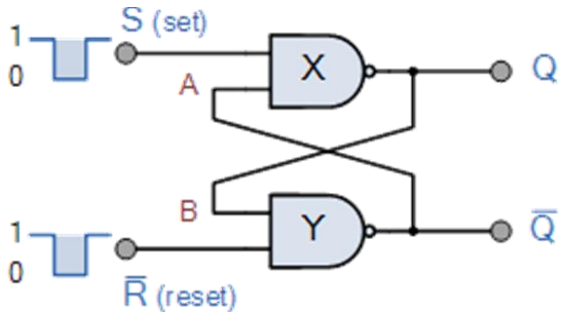
- Consider the circuit
- If the input R is at logic level "0" (R = 0) and input S is at logic level "1" (S = 1), the NAND gate *Y* has at least one of its inputs at logic "0" therefore, its output $\overline{Q}$ must be at a logic level "1"
  - NAND Gate principles
- Output $\overline{Q}$ is also fed 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"

The Set State



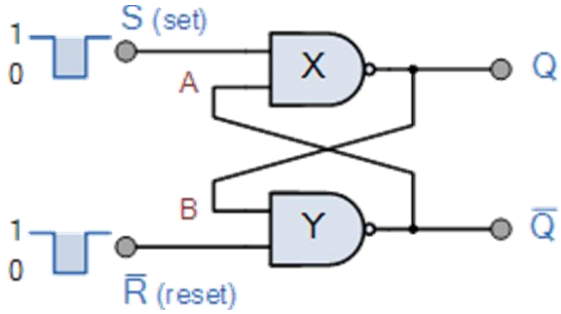
- If the reset input R changes state, and goes HIGH to logic "1" with S remaining HIGH also at logic level "1", NAND gate *Y* inputs are now R = "1" and B = "0"
- Since one of its inputs is still at logic level "0" the output at $\overline{Q}$ still remains HIGH at logic level "1" and there is no change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

Reset State



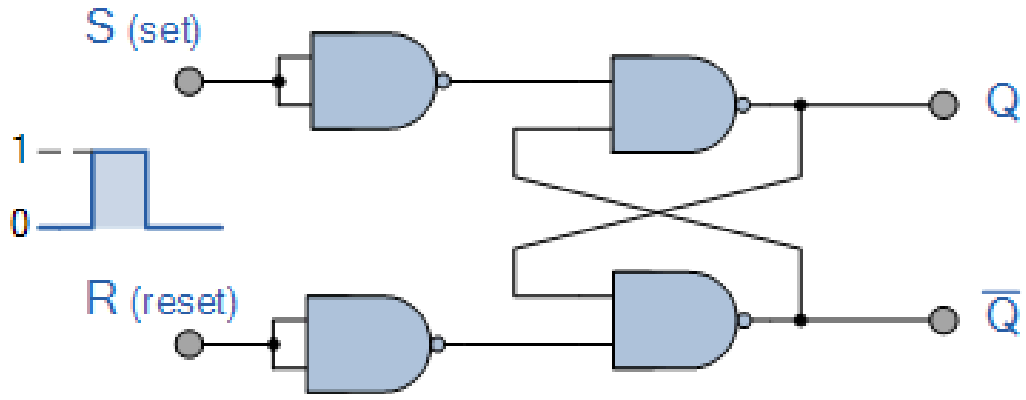
- In this second stable state, $\overline{Q}$ is at logic level "0", (not Q = "0") its inverse output at Q is at logic level "1", (Q = "1"), and is given by R = "1" and S = "0"
- As gate *X* has one of its inputs at logic "0" its output Q must equal logic level "1"
- Output Q is fed back to input "B", so both inputs to NAND gate *Y* are at logic "1", therefore, $\overline{Q}$ = "0"

Reset State



- If the set input, S now changes state to logic “1” with input R remaining at logic “1”, output $\overline{Q}$ 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

Positive NAND Gate SR Flip-flop

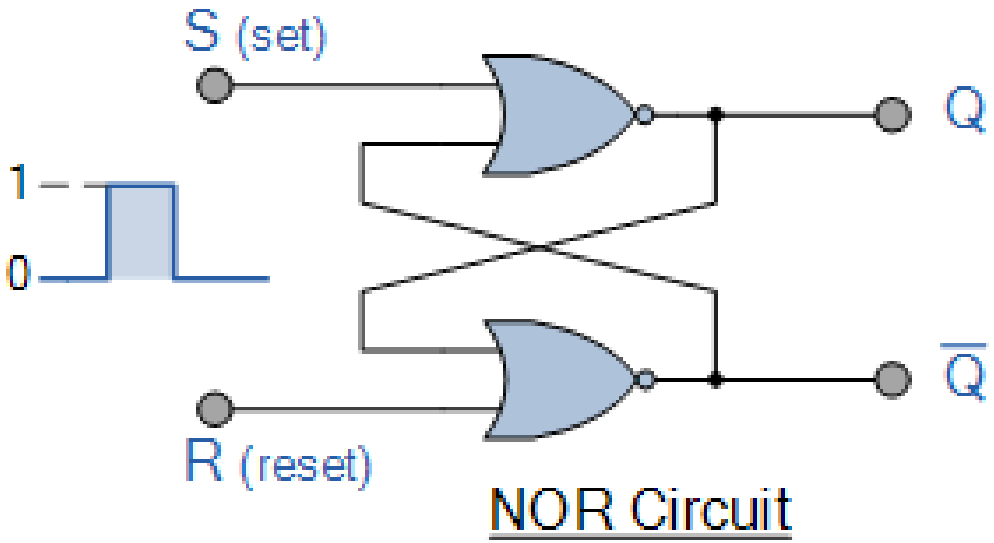


NAND Circuit

S	R	Q	$\overline{Q}$
0	0	No change	
0	1	0	1
1	0	1	0
1	1	X (Invalid)	X

- As well as using NAND gates, it is also possible to construct simple one-bit SR Flip-flops using two cross-coupled NOR gates connected in the same configuration
- The circuit will work in a similar way to the previous NAND gate circuit, except that the inputs are active HIGH and the invalid condition exists when both its inputs are at logic level “1”

The NOR Gate SR Flip-flop



NOR Circuit

S	R	Q	$\overline{Q}$
0	0	No change	
0	1	1	0
1	0	0	1
1	1	X (Invalid)	X

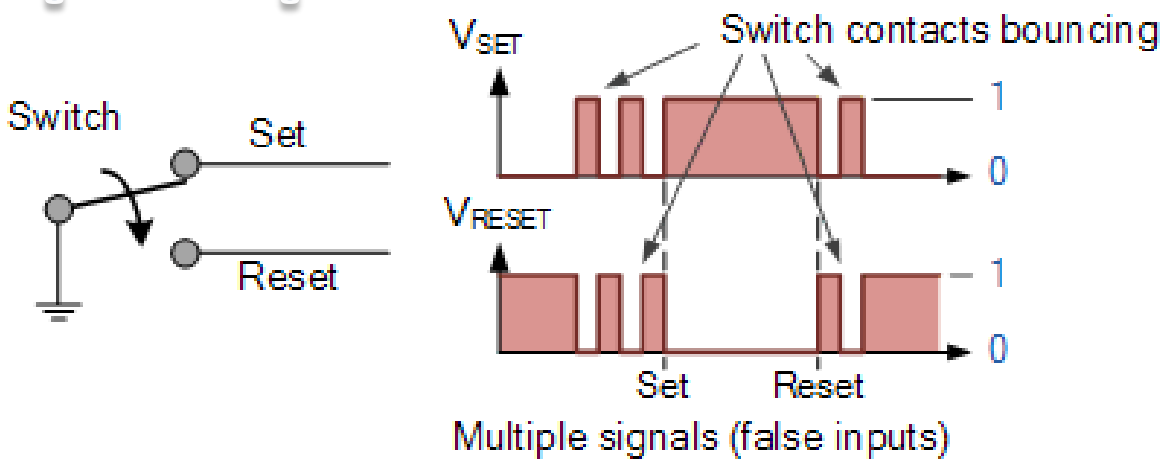
Switch Debounce

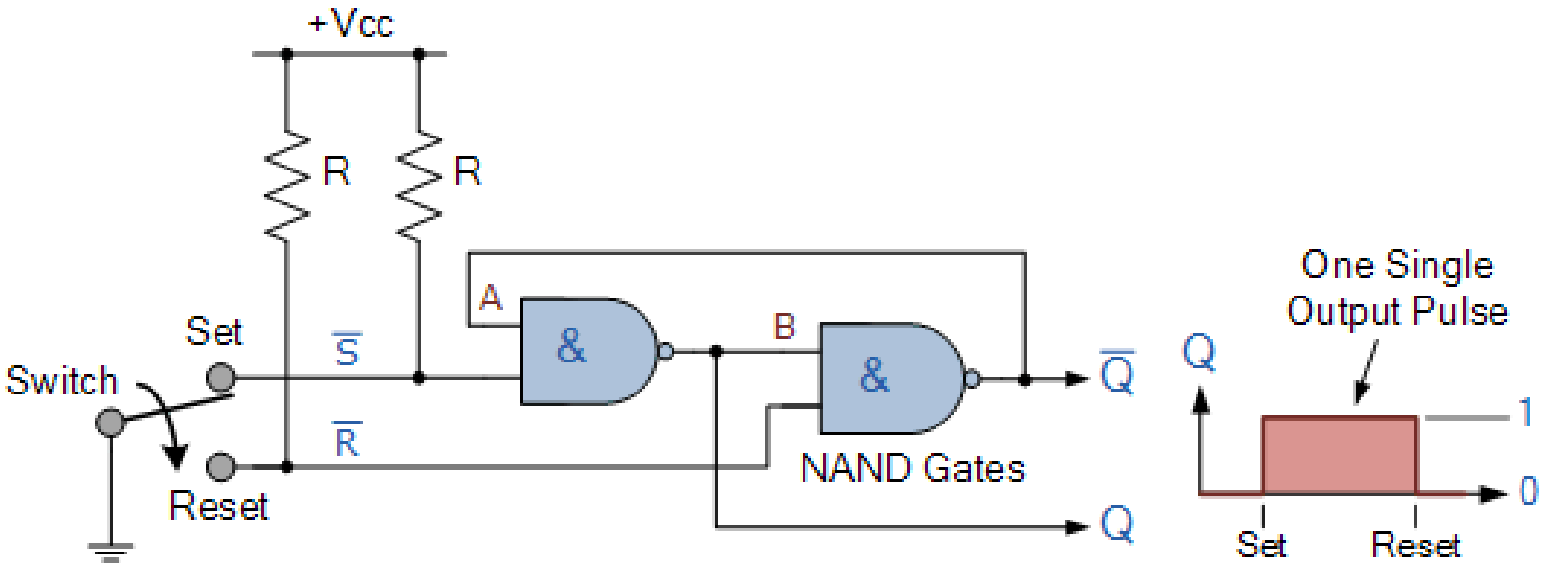
- Edge-triggered flip-flops require a nice clean signal transition, and one practical use of this type of set-reset circuit is as a latch used to help eliminate mechanical switch “bounce”
- As its name implies, switch bounce occurs when the contacts of any mechanically operated switch, push-button or keypad are operated and the internal switch contacts do not fully close cleanly, but bounce together first before closing (or opening) when the switch is pressed

Switch Debounce

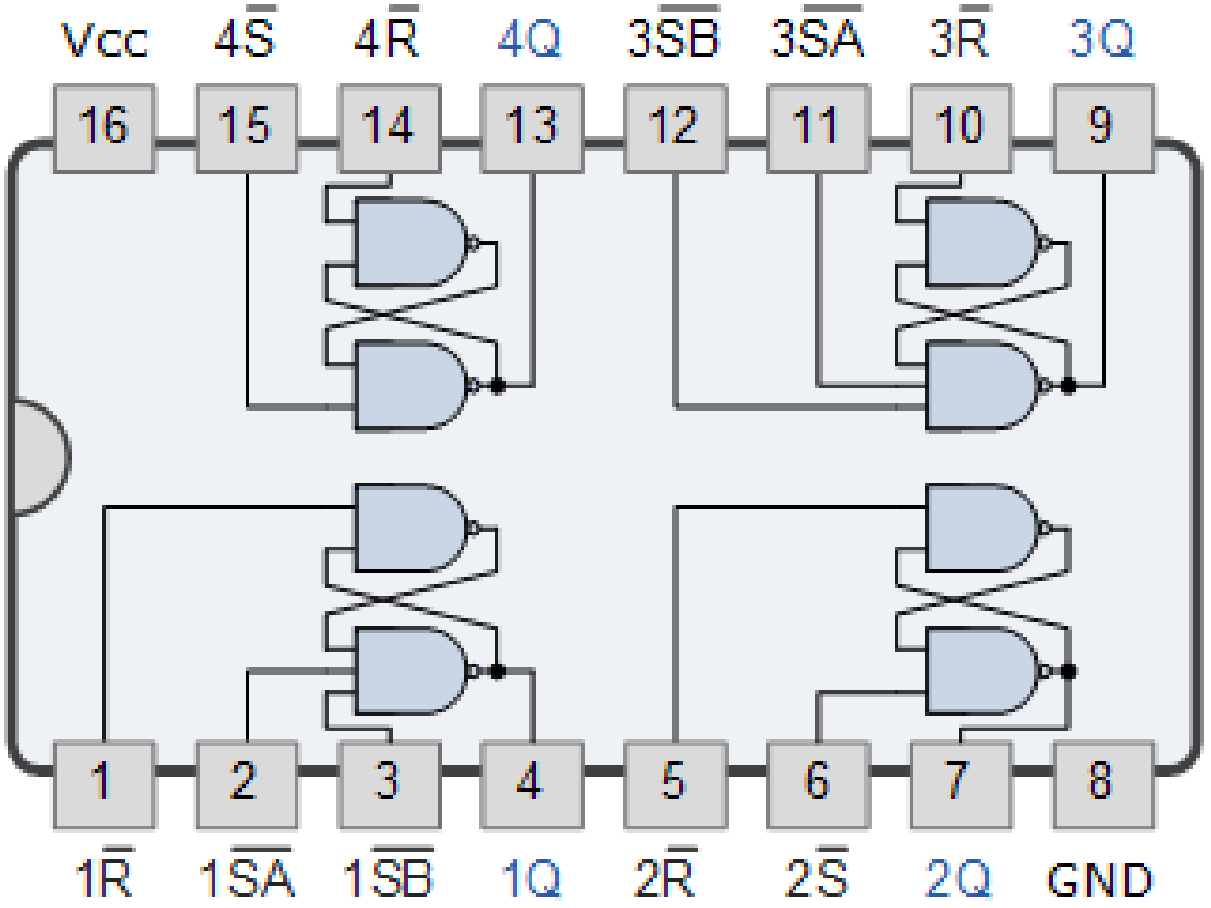
- This gives rise to a series of individual pulses which can be as long as tens of milliseconds that an electronic system or circuit such as a digital counter may see as a series of logic pulses instead of one long single pulse and behave incorrectly
- For example, during this bounce period the output voltage can fluctuate wildly and may register multiple input counts instead of one single count
- Then set-reset SR Flip-flops or Bistable Latch circuits can be used to eliminate this kind of problem and this is demonstrated below

SR Flip Flop Switch Debounce Circui

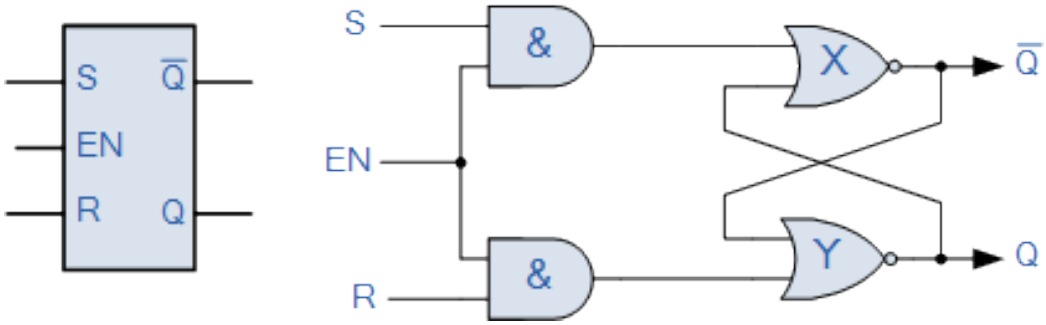




Quad SR Bistable Latch 74LS279



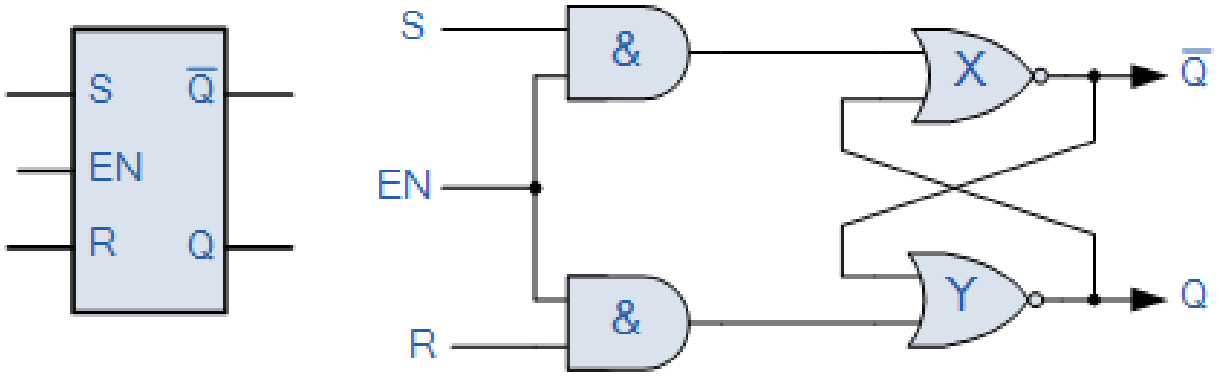
Gated or Clocked SR Flip-Flop



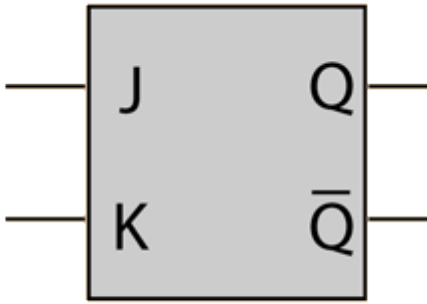
- It is sometimes desirable in sequential logic circuits to have a bistable SR flip-flop that only changes state when certain conditions are met regardless of the condition of either the Set or the Reset inputs
- By connecting a 2-input AND gate in series with each input terminal of the SR Flip-flop a Gated SR Flip-flop can be created

Gated or Clocked SR Flip-Flop

- This extra conditional input is called an "Enable" input and is given the prefix of "EN"
- The addition of this input means that the output at Q only changes state when it is HIGH and can therefore be used as a clock (CLK) input making it level-sensitive



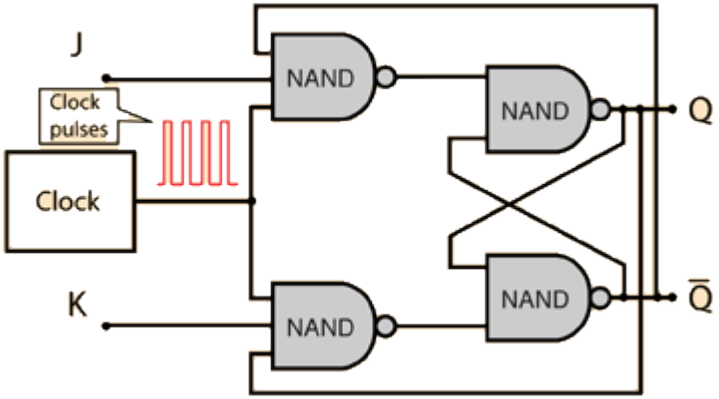
JK Flip Flop



J-K Type

- The J-K flip-flop is the most versatile of the basic flip-flops
- It has the input- following character of the clocked D flip-flop but has two inputs, traditionally labeled J and K
- If J and K are different then the output Q takes the value of J at the next clock edge

J-K Flip-Flop Structure



Truth Table

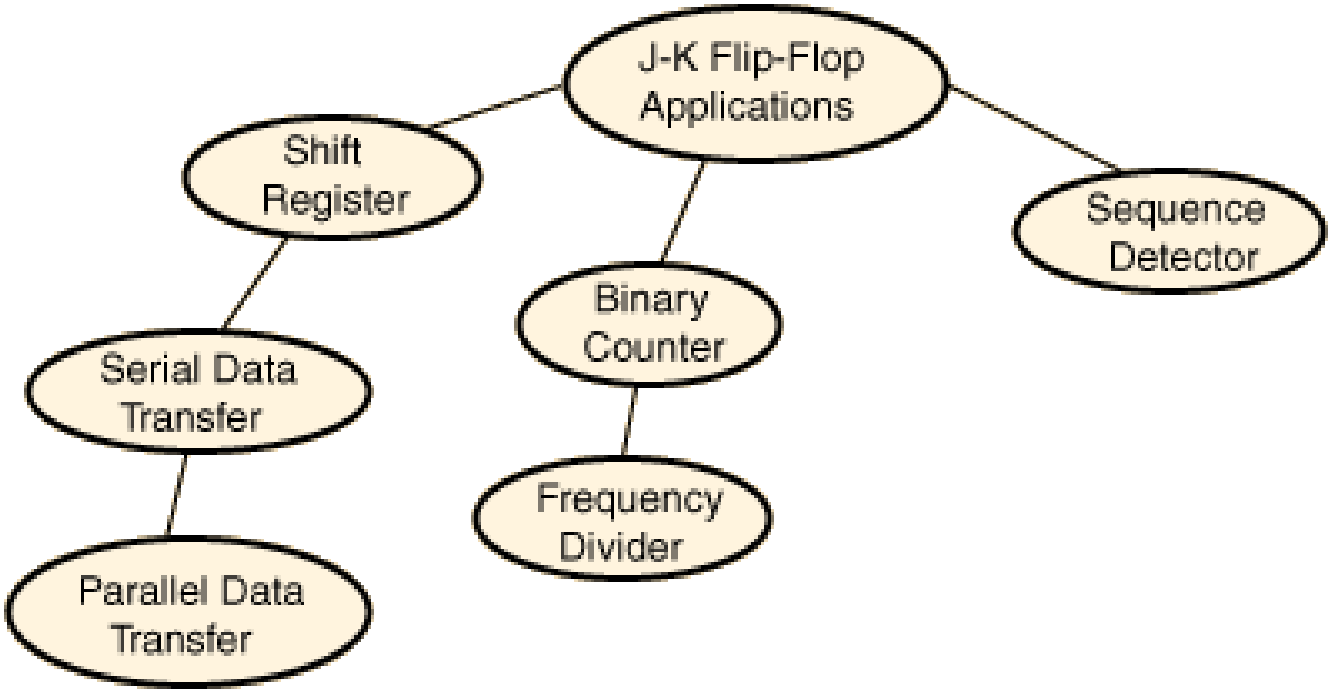
J	K	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q}_0$ (toggles)

- Note that the outputs feed back to the enabling NAND gates. This is what gives the toggling action when J=K=1

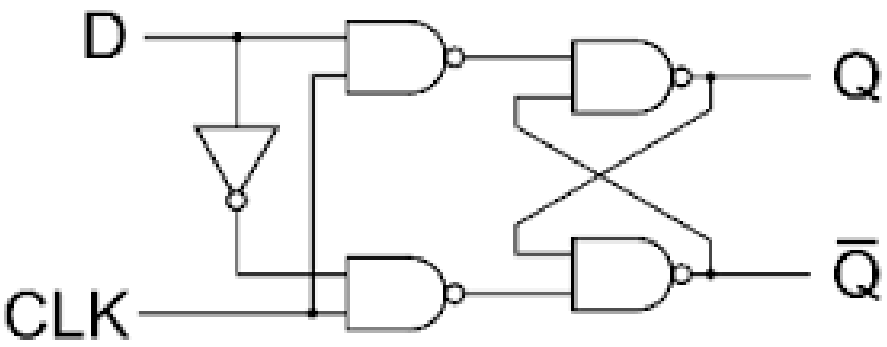
J-K Flip-Flop

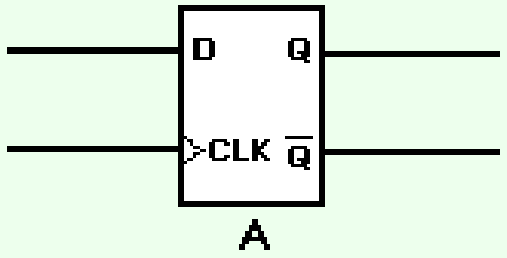
- If J and K are both low then no change occurs
- If J and K are both high at the clock edge then the output will toggle from one state to the other
- It can perform the functions of the set/reset flip-flop and has the advantage that there are no ambiguous states
- It can also act as a T flip-flop to accomplish toggling action if J and K are tied together
- This toggle application finds extensive use in binary counters

Applications of JK Flip-Flop



A D Flip-Flop with Clock

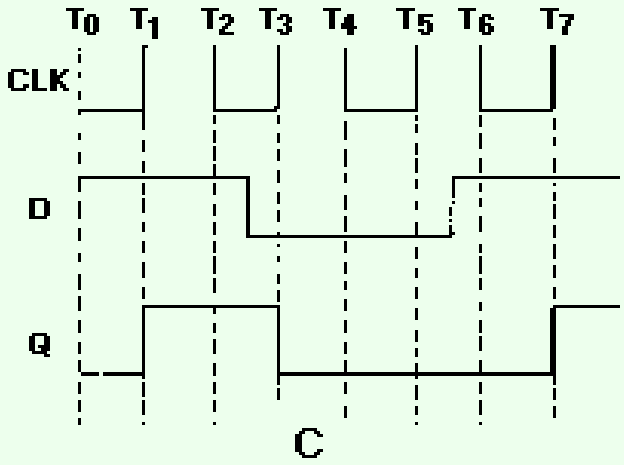




A

CLK	D	Q	
0	1	0	START
↑	1	1	STORE 1
0	0	Q	NO CHARGE
↑	0	0	STORE 0

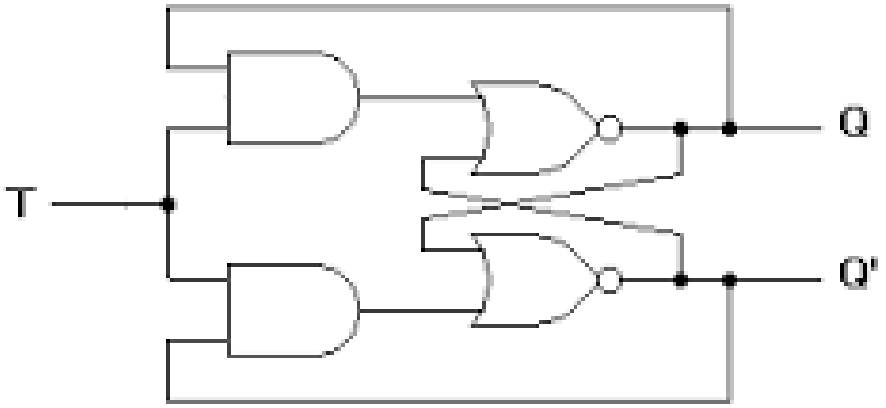
B

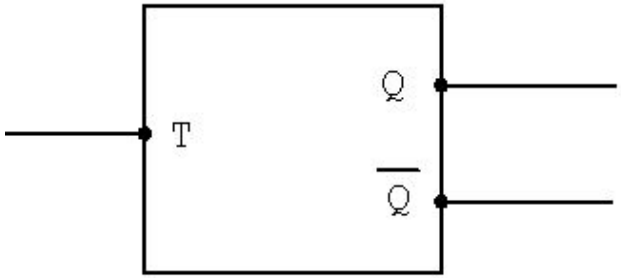


C

T (Toggle) Flip-Flop

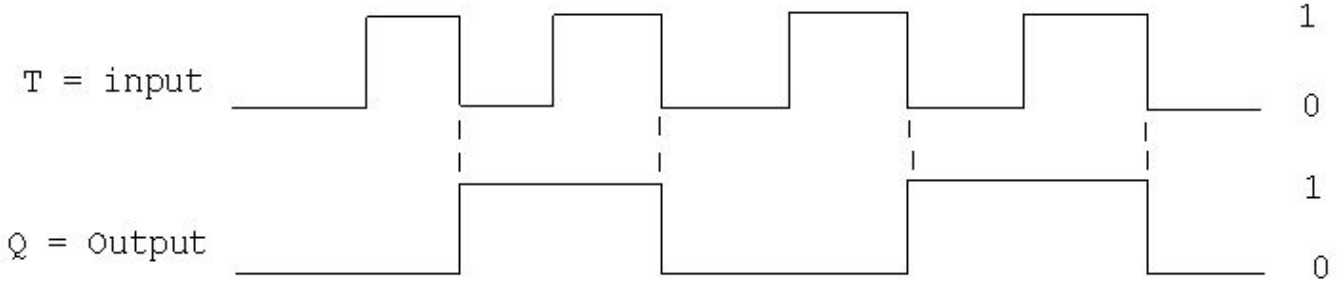
- Without Clock input



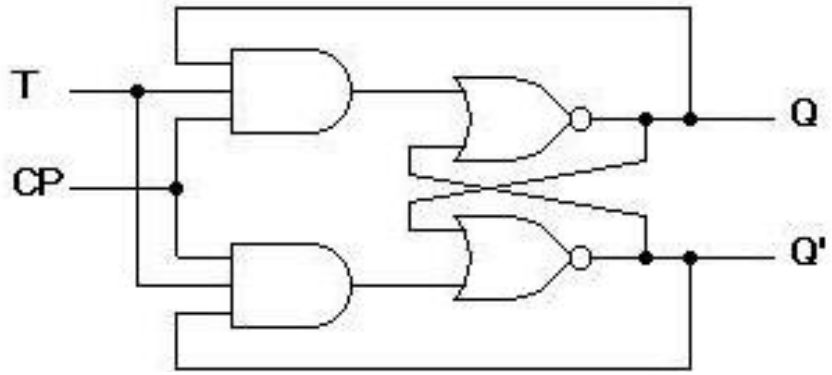


CLOCK	T	Q	Q′
↓	0	1	0
↓	1	0	0
↓	0	1	0
↓	1	0	1

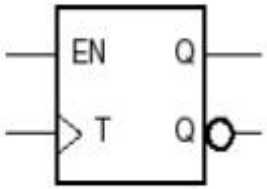
Timing Diagram

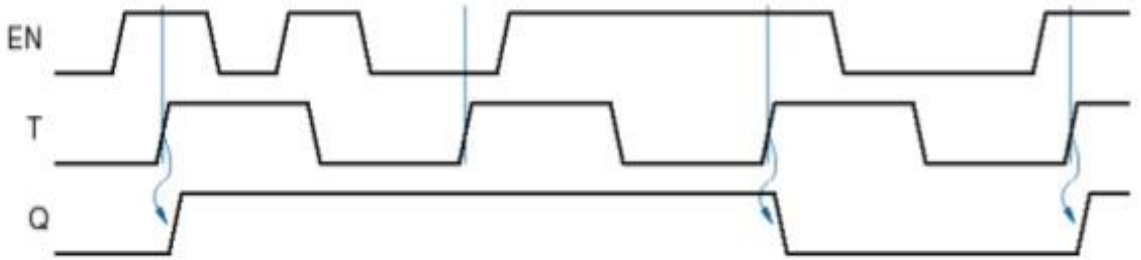


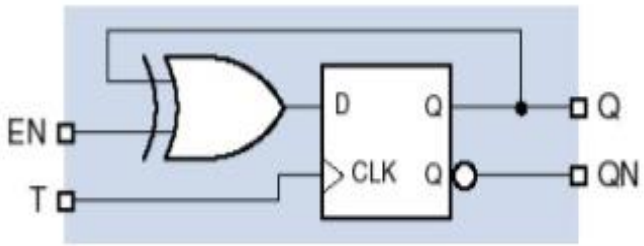
T (Toggle) Flip-Flop : Clocked

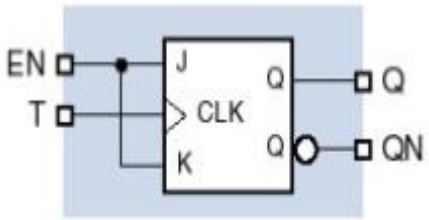


- Important for counters









Latch Vs. Flip-Flop

- A latch is an example of a bistable multivibrator, that is, a device with exactly two stable states
- These states are high-output and low-output
- A latch has a feedback path, so information can be retained by the device
- Therefore latches can be memory devices, and can store one bit of data for as long as the device is powered
- As the name suggests, latches are used to "latch onto" information and hold in place
- Latches are very similar to flip-flops, but are not synchronous devices, and do not operate on clock edges as flip-flops do

Latch Vs. Flip-Flop

- A flip-flop is a device very like a latch in that it is a bistable mutivibrator, having two states and a feedback path that allows it to store a bit of information
- The difference between a latch and a flip-flop is that a **latch is asynchronous, and the outputs can change as soon as the inputs do** (or at least after a small propagation delay)
- A flip-flop, on the other hand, is **edge-triggered and only changes state when a control signal goes from high to low or low to high**
- This distinction is relatively recent and is not formal, with many authorities still referring to flip-flops as latches and vice versa, but it is a helpful distinction to make for the sake of clarity