Sequential Logic I

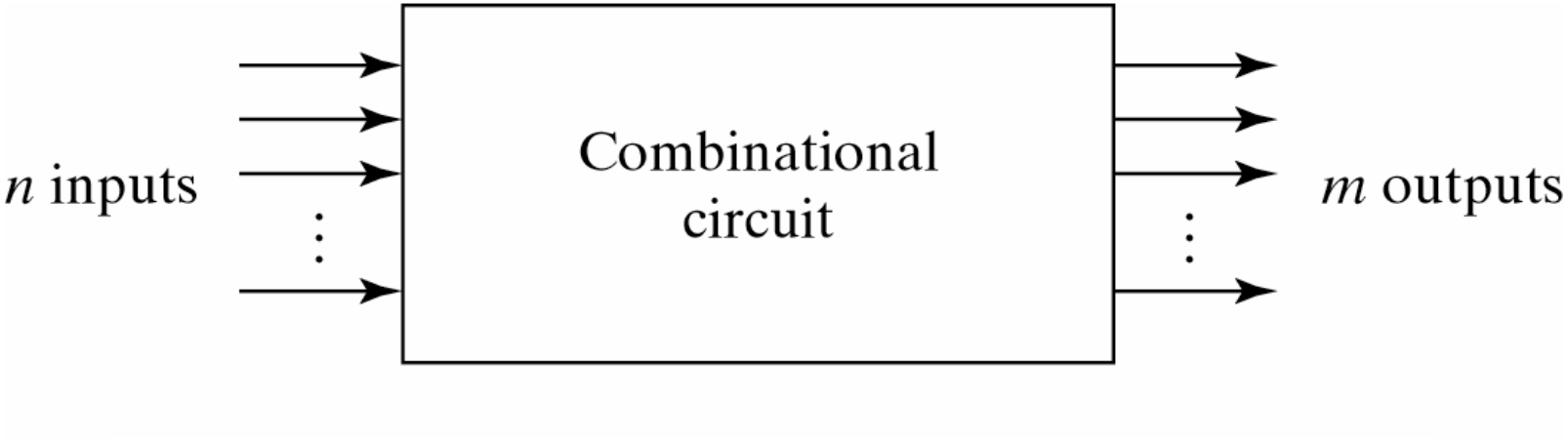
Latches and Flip-Flops

Outline

- Combinational → Sequential
- RS Latch
- Clocked RS Latch
- RS Flip Flops
- D Flip Flops
- JK Flip Flops
- T Flip Flops

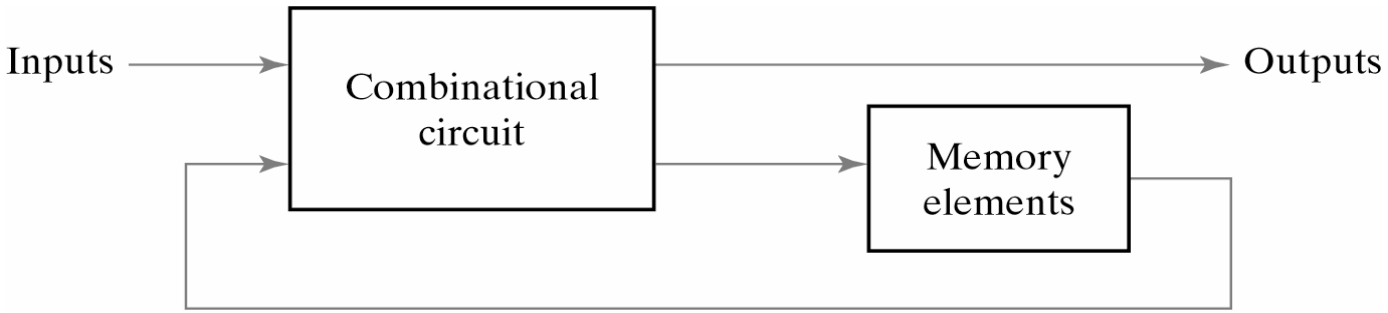
Combinational Logic

- Recall: A combinational logic circuit combines *n* logical inputs using AND, OR, NAND,… logic gates. The *m* outputs are described by *m* logical expressions.
- Combinational logic is regarded as **time-independent** logic. That is, the outputs are entirely dependent on the current inputs.



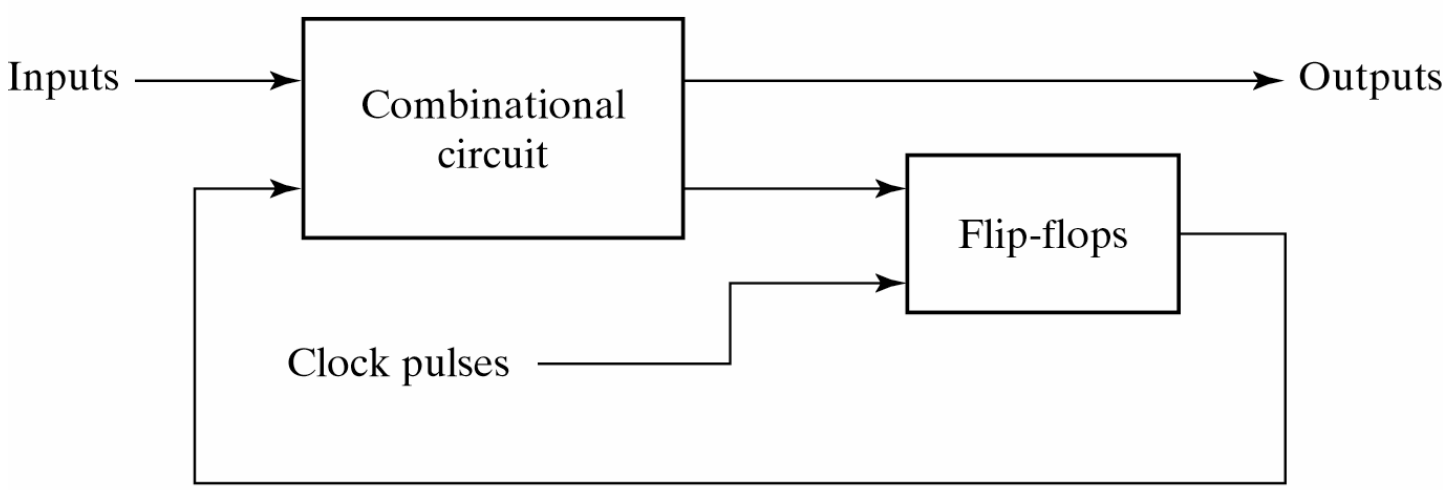
Sequential Logic

- In practice digital systems contain both combinational logic circuits and **memory storage elements**.
- The value of the outputs of those systems is determined by external inputs and the present state of the storage elements.
- This **time-dependent** digital logic is referred to as **sequential logic**.



Sequential Logic

- The memory elements in the sequential logic consist of binary storage elements. These binary storage elements are known as **flip-flops**, which are constructed from **Latches**.

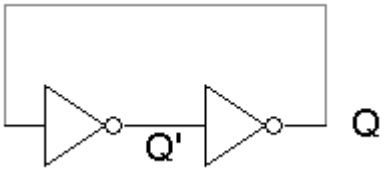


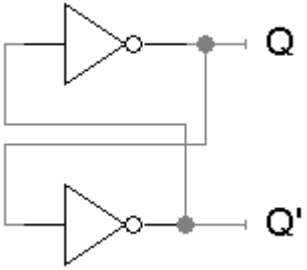
What exactly is memory?

- A memory should have at least three properties.
  1. It should be able to *hold* a value.
  2. You should be able to *read* the value that was saved.
  3. You should be able to *change* the value that was saved.
- We'll start with the simplest case, a one-bit memory.
  1. It should be able to hold a single bit, 0 or 1.
  2. You should be able to read the bit that was saved.
  3. You should be able to change the value. Since there's only a single bit, there are only two choices:
     - Set the bit to 1
     - Reset, or clear, the bit to 0.

The basic idea of Storage/Memory

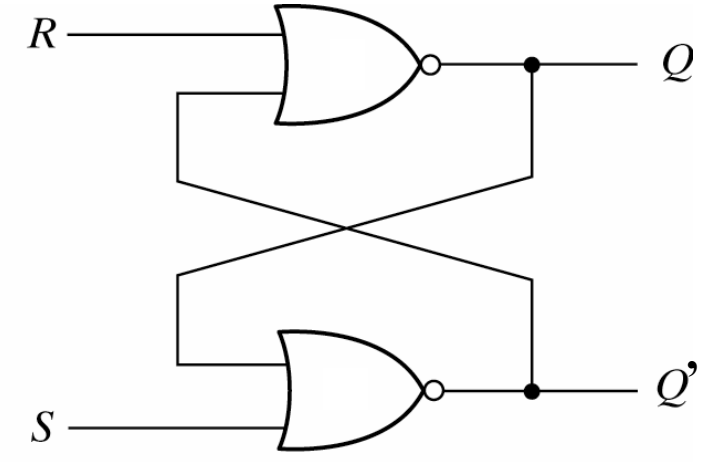
- How can a circuit “remember” anything? A simple idea is to make a loop, e.g., putting the circuit outputs into the inputs.
- Here is one initial attempt, shown with two equivalent layouts:





- Does this satisfy the properties of memory?
  - These circuits “remember” Q, because its value never changes. (Similarly, Q’ never changes either.)
  - We can also “read” Q, by attaching a probe or another circuit.
  - But we can’t *change* Q! There are no external inputs here, so we can’t control whether Q=1 or Q=0.

What about using NOR gates instead of inverters?



- Now, the two inputs S and R will let us control the outputs Q and Q'. This two-state circuit is called **"RS" (or "SR") Latch**.
- To figure out how Q and Q' change, we have to look at not only the inputs S and R, but also the *current* values of Q and Q':

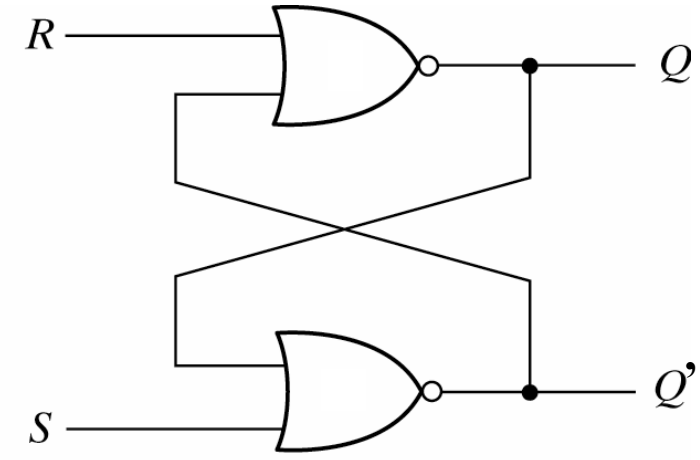
$$Q_{next} = (R + Q'_{current})'$$
$$Q'_{next} = (S + Q_{current})'$$

RS Latch

- What if S = 0 and R = 0?
- The equations on the right reduce to:

$$Q_{next} = (0 + Q'_{current})' = Q_{current}$$
$$Q'_{next} = (0 + Q_{current})' = Q'_{current}$$

- So when SR = 00, then $Q_{next} = Q_{current}$. Whatever value Q has, it keeps.
- This is exactly what we need to store, i.e., remember values in the latch.



$$Q_{next} = (R + Q'_{current})'$$
$$Q'_{next} = (S + Q_{current})'$$

RS Latch

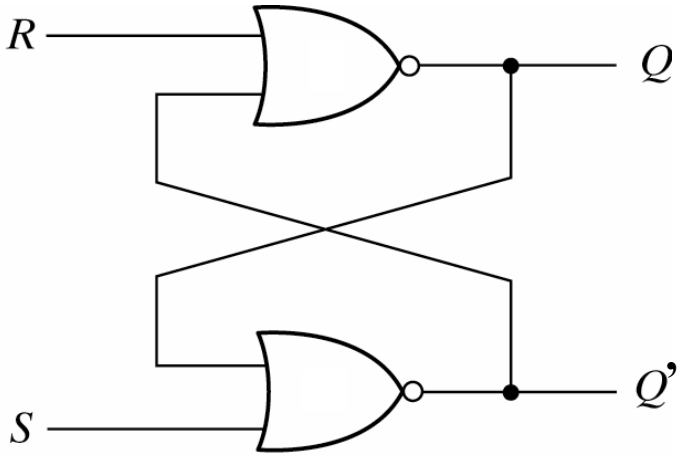
- What if S = 1 and R = 0?
- Since S = 1, Q'_{next} is 0, *regardless* of $Q_{current}$:

$$Q'_{next} = (1 + Q_{current})' = 0$$

- Then, this new value of Q' goes into the top NOR gate, along with R = 0.

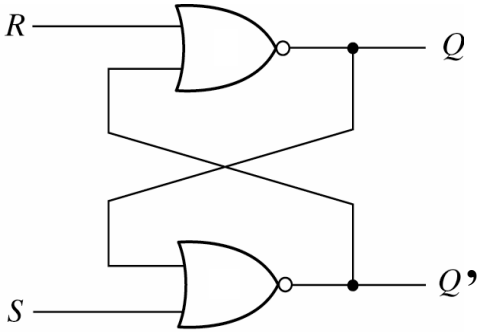
$$Q_{next} = (0 + 0)' = 1$$

- So when SR = 10, then $Q'_{next} = 0$ and $Q_{next} = 1$.
- This is how you set the latch to 1. The S input stands for "set."
- Notice that it can take up to two steps (two gate delays) from the time S becomes 1 to the time Q_{next} becomes 1.
- But once Q_{next} becomes 1, the outputs will stop changing. This is a stable state.



$$Q_{next} = (R + Q'_{current})'$$
$$Q'_{next} = (S + Q_{current})'$$

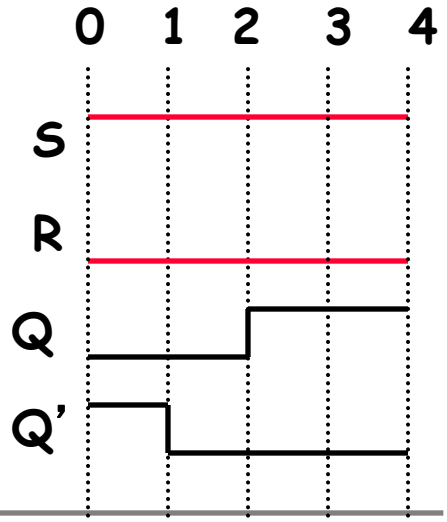
RS Latch



- Timing diagrams are especially useful in understanding how sequential circuits work.
- Here is a diagram which shows an example of how our latch outputs change with inputs SR=10.

$$Q_{next} = (R + Q'_{current})'$$
$$Q'_{next} = (S + Q_{current})'$$

0. Suppose that initially, Q = 0 and Q' = 1.
1. Since S=1, Q' will change from 1 to 0 after one NOR-gate delay (marked by vertical lines in the diagram for clarity).
2. This change in Q', along with R=0, causes Q to become 1 after another gate delay.
3. The latch then stabilizes until S or R change again.



RS Latch

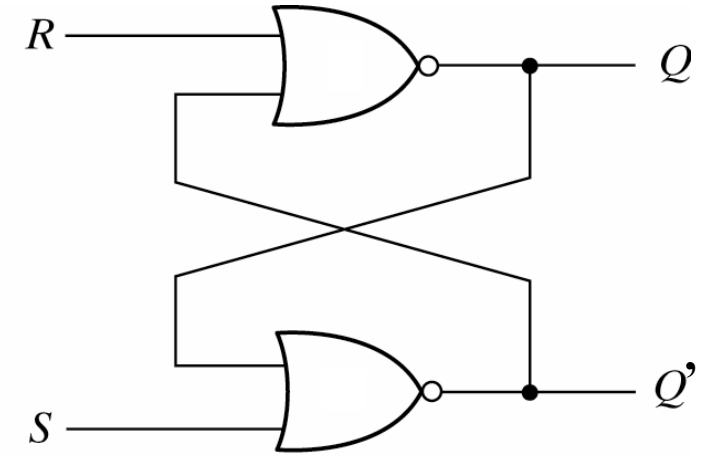
- What if S = 0 and R = 1?
- Since R = 1, Q_{next} is 0, *regardless* of $Q_{current}$:

$$Q_{next} = (1 + Q'_{current})' = 0$$

- Then, this new value of Q goes into the bottom NOR gate, where S = 0.

$$Q'_{next} = (0 + 0)' = 1$$

- So when SR = 01, then $Q_{next} = 0$ and $Q'_{next} = 1$.
- This is how you reset, or clear, the latch to 0. The R input stands for "reset."
- Again, it can take two gate delays before a change in R propagates to the output Q'_{next}.



$$Q_{next} = (R + Q'_{current})'$$
$$Q'_{next} = (S + Q_{current})'$$

RS Latch

- What about SR = 11?
- Both Q_{next} and Q'_{next} will become 0. This contradicts the assumption that Q and Q' are always complements.
- Another problem is what happens if we then make S = 0 and R = 0 together.

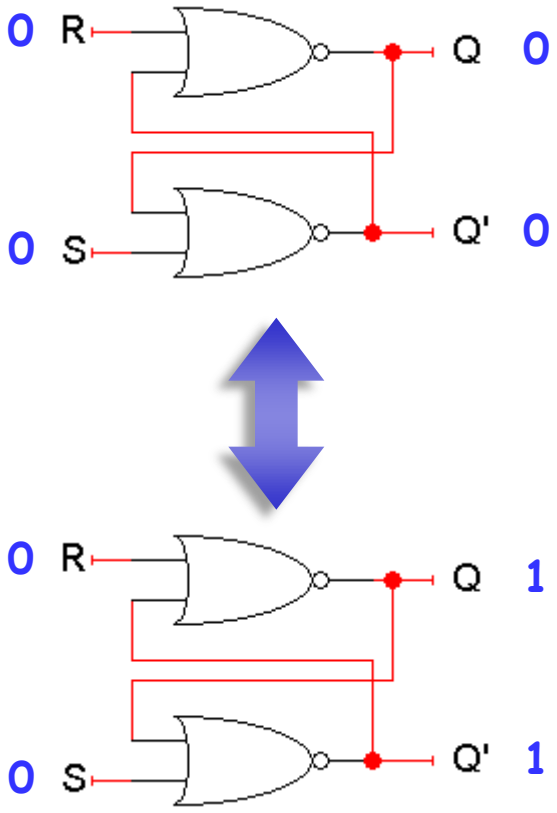
$$Q_{next} = (0 + 0)' = 1$$
$$Q'_{next} = (0 + 0)' = 1$$

- But these new values go back into the NOR gates, and in the next step we get:

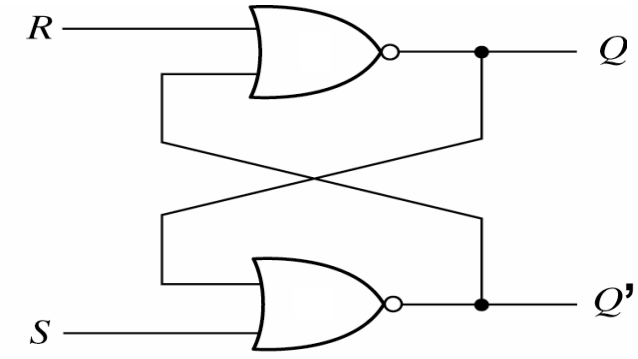
$$Q_{next} = (0 + 1)' = 0$$
$$Q'_{next} = (0 + 1)' = 0$$

- The circuit enters an infinite loop, where Q and Q' cycle between 0 and 1 forever.
- **This indeterminate state should be avoided.**

$$Q_{next} = (R + Q'_{current})'$$
$$Q'_{next} = (S + Q_{current})'$$



RS Latch



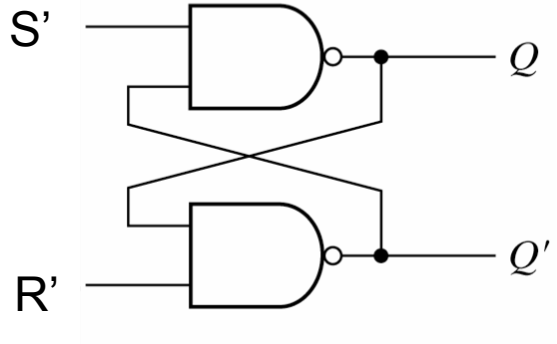
S	R	Q
0	0	No change
0	1	0 (reset)
1	0	1 (set)
1	1	Indeterminate

- Now we can see that in an RS Latch, the same inputs can yield different outputs, depending on whether the latch was previously set or reset.
- This is very different from the combinational circuits that we have seen so far, where the same inputs always yield the same outputs.

Inputs		Current		Next	
S	R	Q	Q'	Q	Q'
0	0	0	1	0	1
0	0	1	0	1	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	0	1	0

RS Latch

- Another common implementation of the RS latch uses **NAND** gates.
- In this implementation the logic levels are inverted. Low S' (logic-0) set Q to 1 and low R' (logic-0) reset Q to 0.

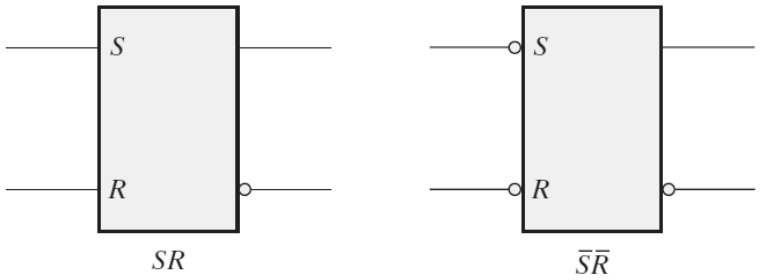


S'	R'	Q
1	1	No change
1	0	0 (reset)
0	1	1 (set)
0	0	Indeterminate

- Also called $\bar{S} - \bar{R}$ Latch

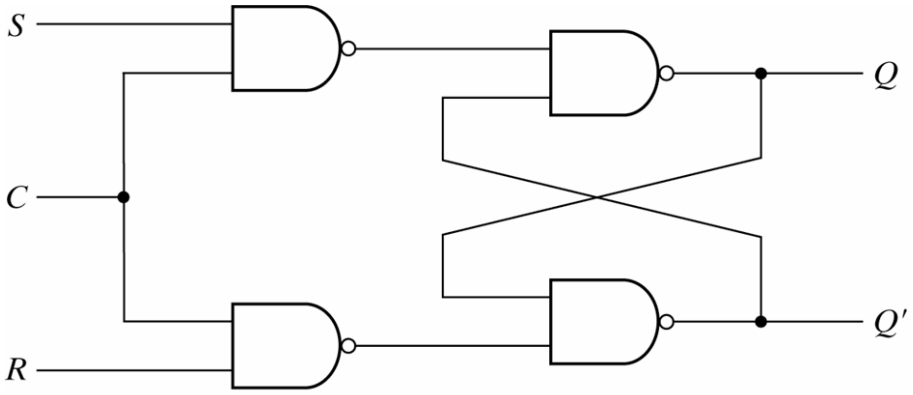
Graphic Symbols for Latches

- A latch is designated by a rectangular block with inputs on the left and outputs on the right.
- One output designates the normal output, and the other (with the bubble designation) designates the complement output.
- In the case of a NOR gate latch, setting and resetting occur with a logic-1 signal.
- In the case of a NAND gate latch, bubbles are added to the inputs to indicate that setting and resetting occur with a logic-0 signal.



Clocked/Enabled RS Latch

- The RS latch can be augmented with the addition of a control or enable input C.
- When C=1 the circuit operates as normal and is said to be enabled.
- When C=0 the circuit is disabled and the R & S will not effect the output state.



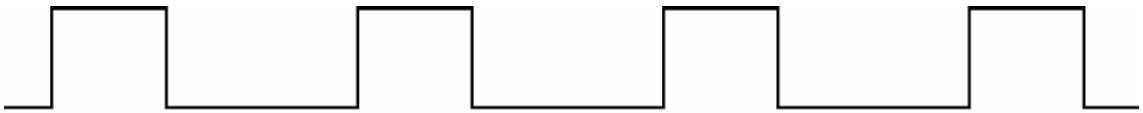
(a) Logic diagram

C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$; Reset state
1	1	0	$Q = 1$; set state
1	1	1	Indeterminate

(b) Function table

Clocked RS Latch

- The control input can be connected to a square-wave generator



in this case the latch is said to be *clocked*.

- When the clock level goes high the latch is enabled and remains enabled until the clock level goes low again.

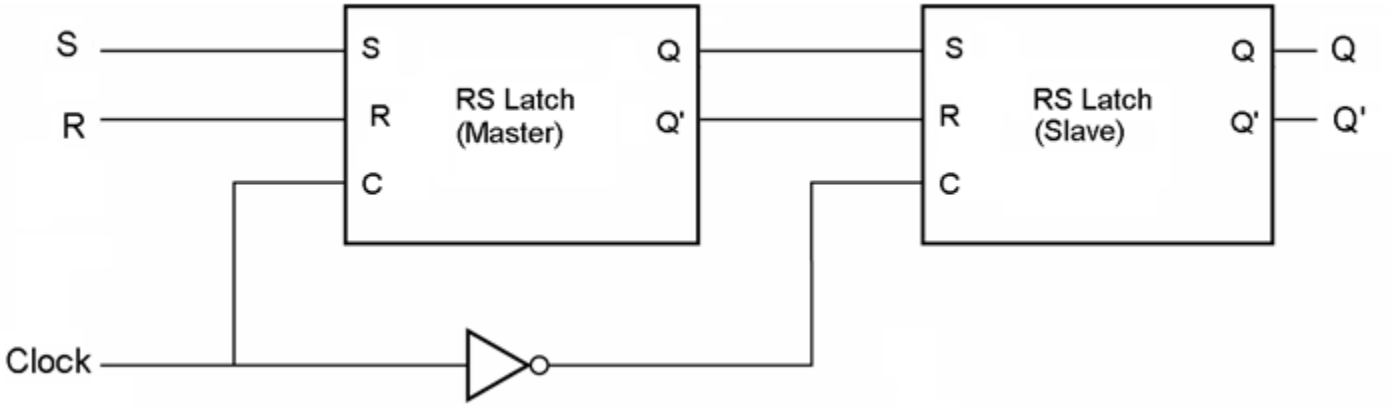
- This kind of operation is known as level triggering.

Clocked RS Latch

- During a positive clock level latches are free to respond to inputs and change state multiple times before the logic level goes to zero.
- When level triggering is used in a circuit, latches can change unpredictably in an asynchronous way.
- Asynchronous design is difficult. It is often more desirable to synchronise the operation of the latches.

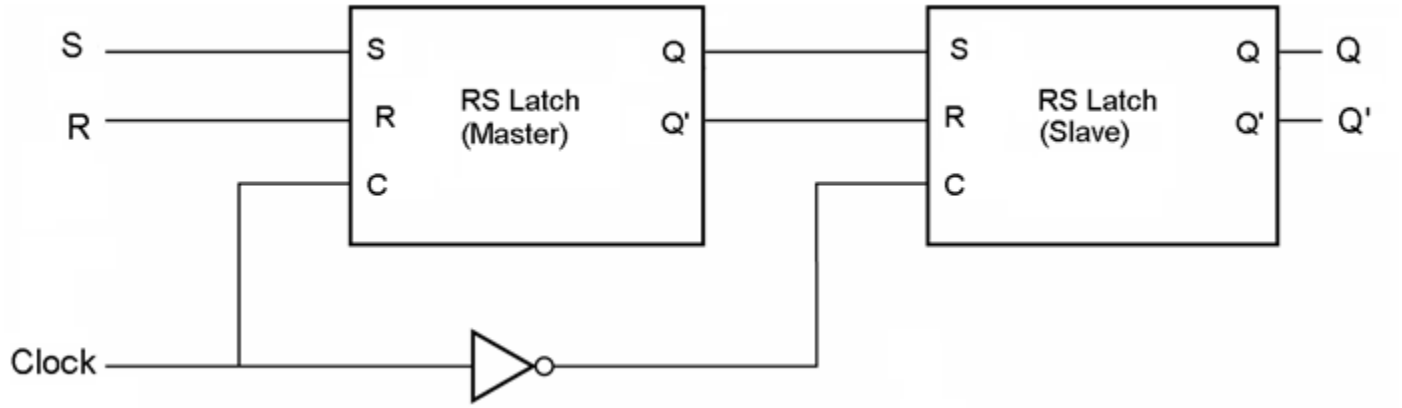
RS Flip-Flop

- Synchronisation is performed by only allowing triggering during level changes.
- Latching circuits which are synchronised to a clock are known as flip-flop circuits.



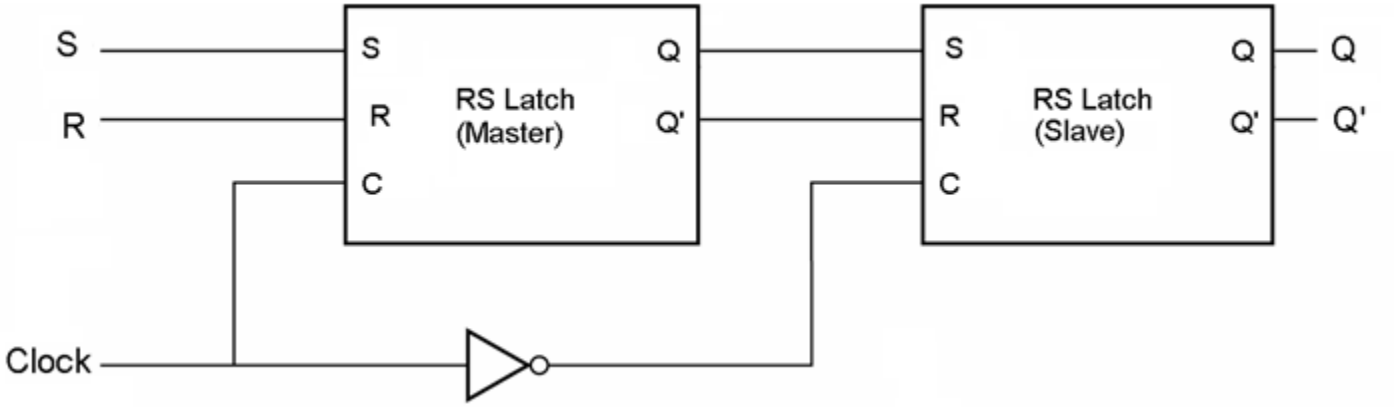
RS Flip-Flop

- The RS flip-flop is constructed using two RS latches and an inverter.
- The inverter ensures that both latches will operate at different clock levels.



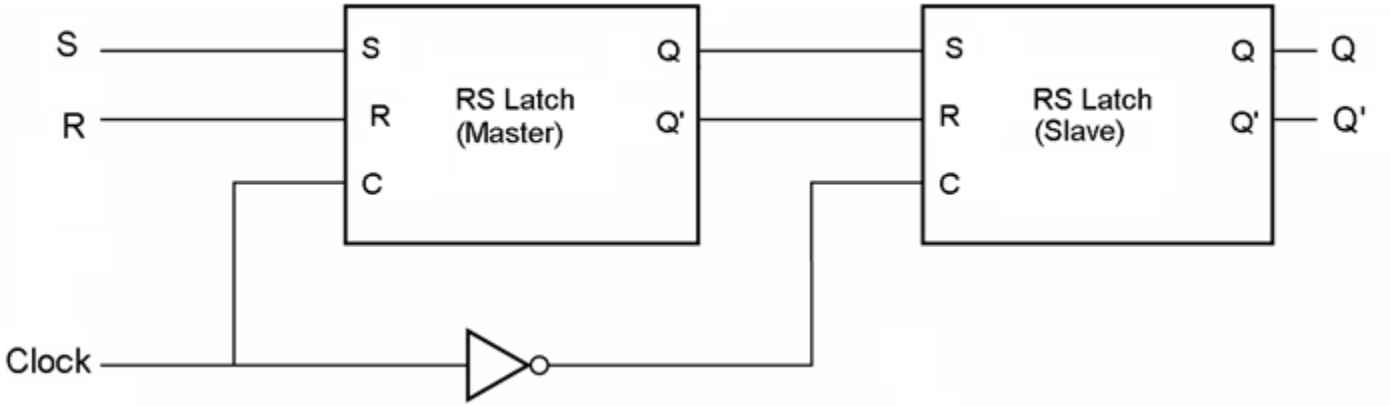
RS Flip-Flop

- When the clock signal changes to logic 1 the master latch is activated. The inverted clock disables the slave latch.
- The input signals pass into the master latch and alter the master outputs.



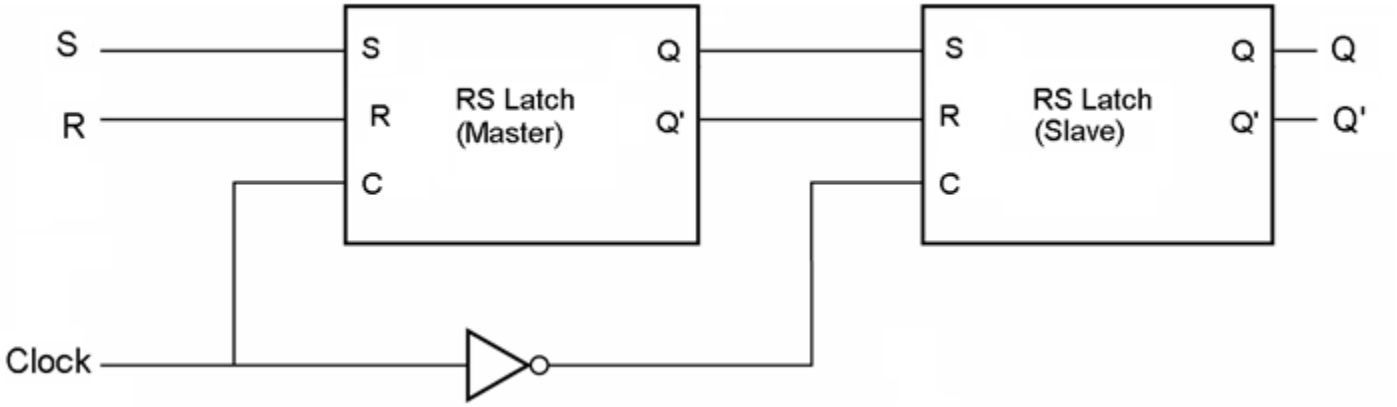
RS Flip-Flop

- When the clock signal changes to logic 0 the master latch is disabled and slave latch is activated.
- The master output signals now pass into the slave latch and alter the slave outputs. Since the master is now disabled no other input signals will affect the flip flop.



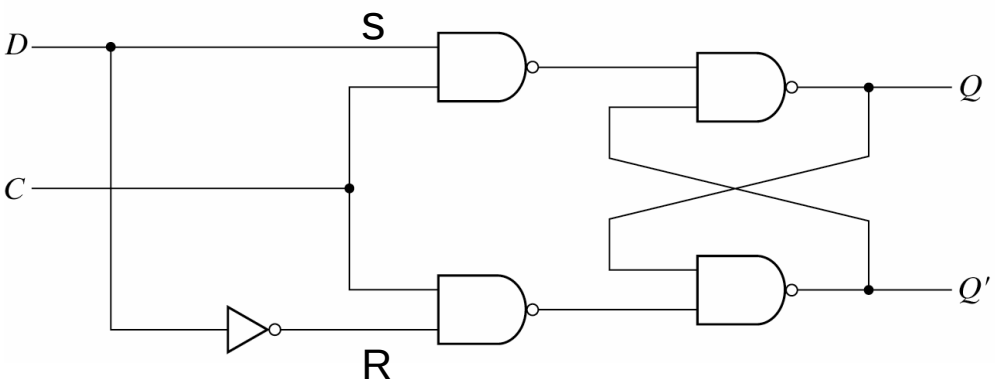
RS Flip-Flop

- The output of this RS flip-flop can only change during a clock transition from logic 1 to 0 (or during the negative transition).
- As a result this circuit is known as negative edge triggered flip flop.



D Latch

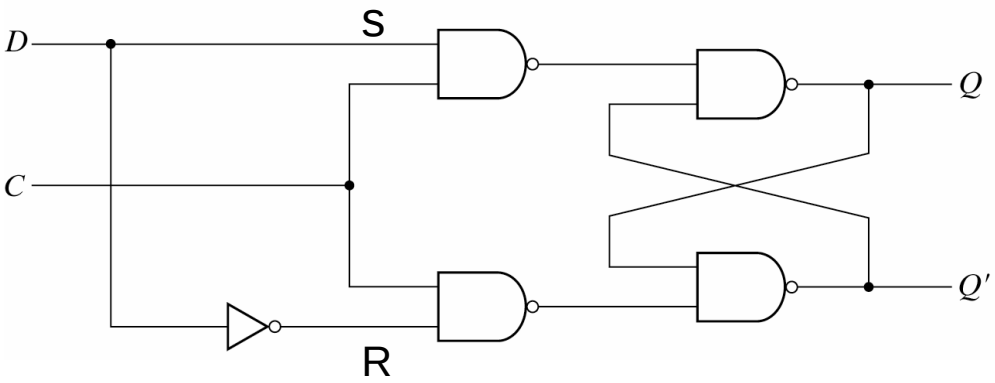
- Direct use of the RS latch is usually avoided because of the indeterminate state arising when R=1 and S=1.
- A simple solution to the problem of having an indeterminate state in the RS latch is the D latch.
- The D latch solution involves ensuring that the R and S inputs are always complemented and so R=S=1 cannot occur.



C	D	Q_{t+1}
0	x	Q_t
1	0	0
1	1	1

D Latch

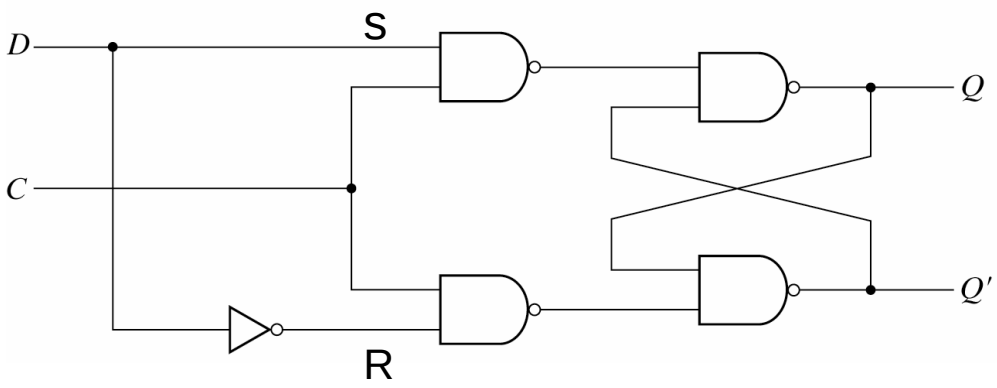
- This is achieved by connecting the R input to the S input via an inverter.
- The R input now becomes the D input.
- An additional input C is included in the design to control whether or not the D input passes into the RS latch.



C	D	Q_{t+1}
0	x	Q_t
1	0	0
1	1	1

D Latch

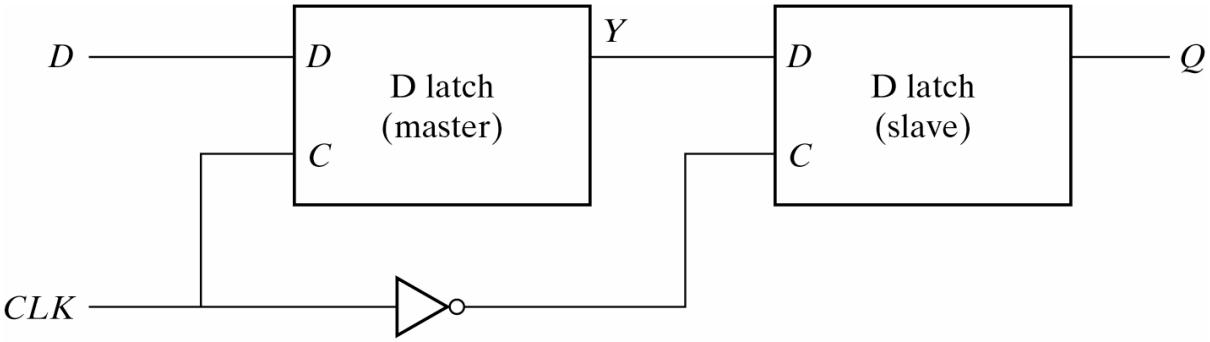
- Now when C = 1 the D (data) input is passed to the output Q.
- When C = 0 the D input is not permitted to change the output and so the previous bit is saved.



C	D	Q_{t+1}
0	x	Q_t
1	0	0
1	1	1

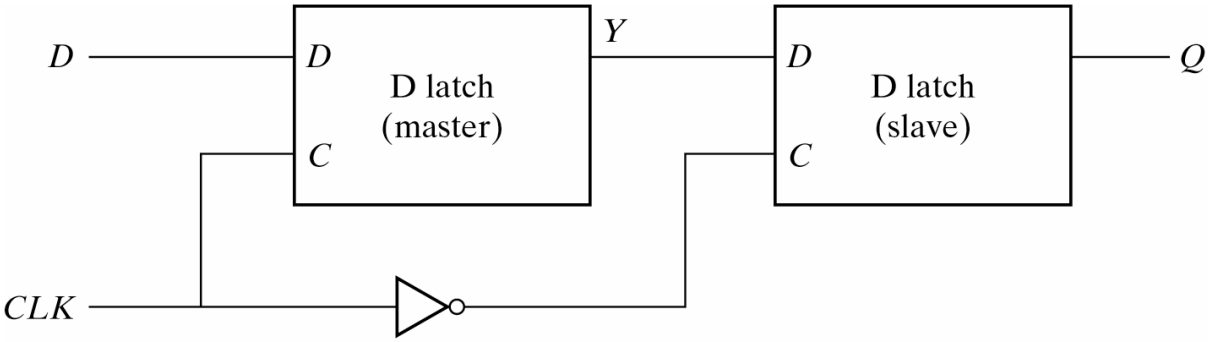
D Flip-Flop

- We saw how two RS latches could be used along with an inverter to create a negative edge triggered flip-flop earlier.
- The same can be done with two D latches to create a D flip-flop.



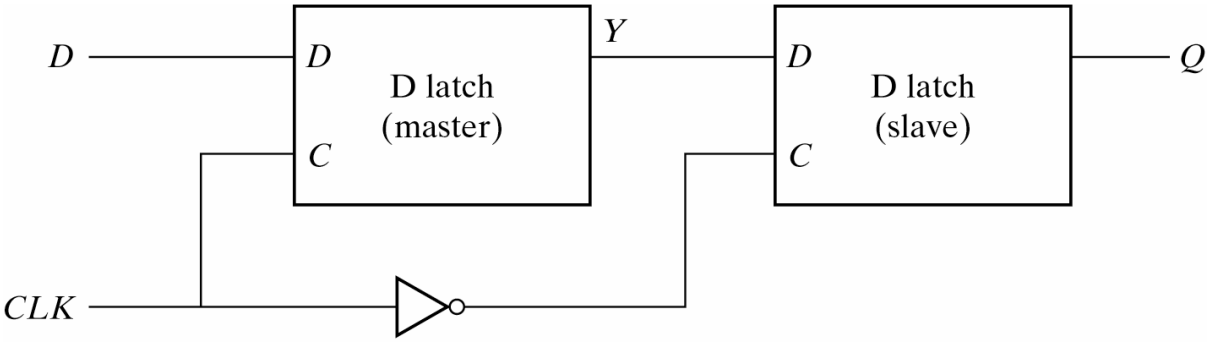
D Flip-Flop

- When the clock is at logic 1 the master D latch is enabled and the slave D latch is disabled.
- While the clock remains at logic 1 data can pass into the master latch to affect the master output Y.
- The master output Y cannot affect the slave latch.



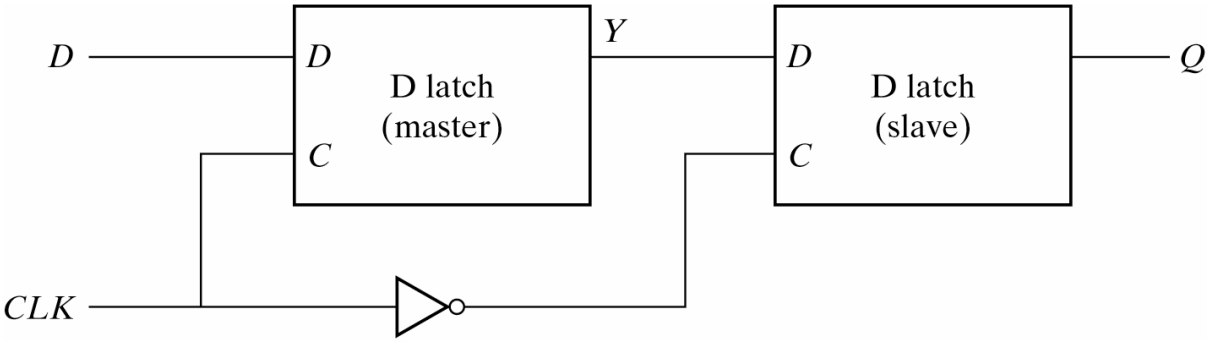
D Flip-Flop

- When the clock goes negative the master D latch is disabled and the slave D latch is enabled.
- The logic level present at the beginning of the positive to negative transition at master output Y passes into the the slave latch and appears at the output Q.



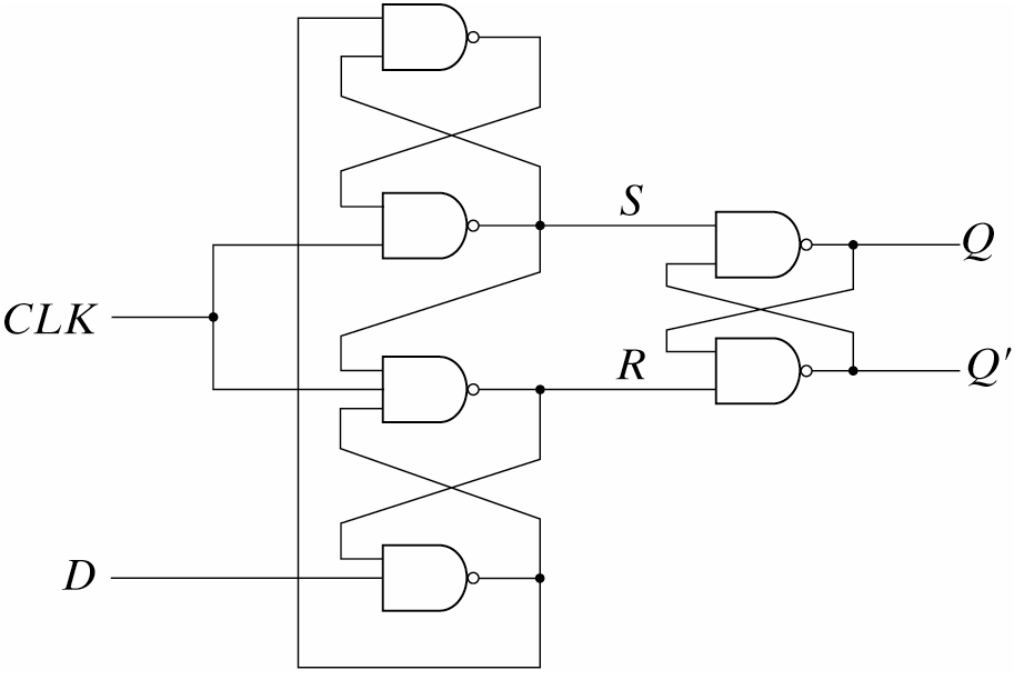
D Flip-Flop

- Since the master latch is disabled until the clock goes high again no more logic changes at the input D can effect the output Q.
- Only data presented during a positive to negative clock change is stored at output Q.



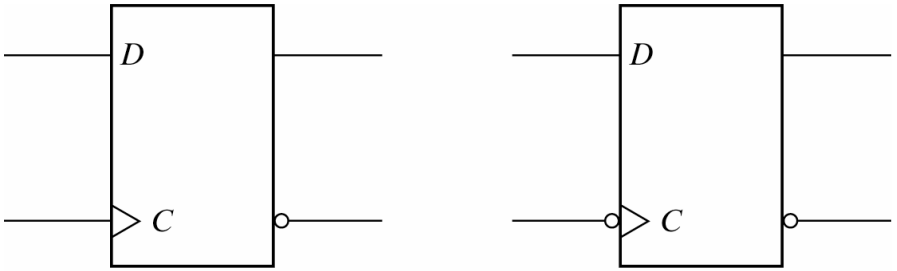
D Flip-Flop

- The master-slave D flip-flop uses eight gates.
- A more efficient construction of the **positive** edge triggered D flip-flop uses three RS latches.



D Flip-Flop

- The D flip-flop is a popular choice because of its economical and efficient construction.
- The graphical symbols for positive-edge and negative-edge triggered flip-flops are as follows.

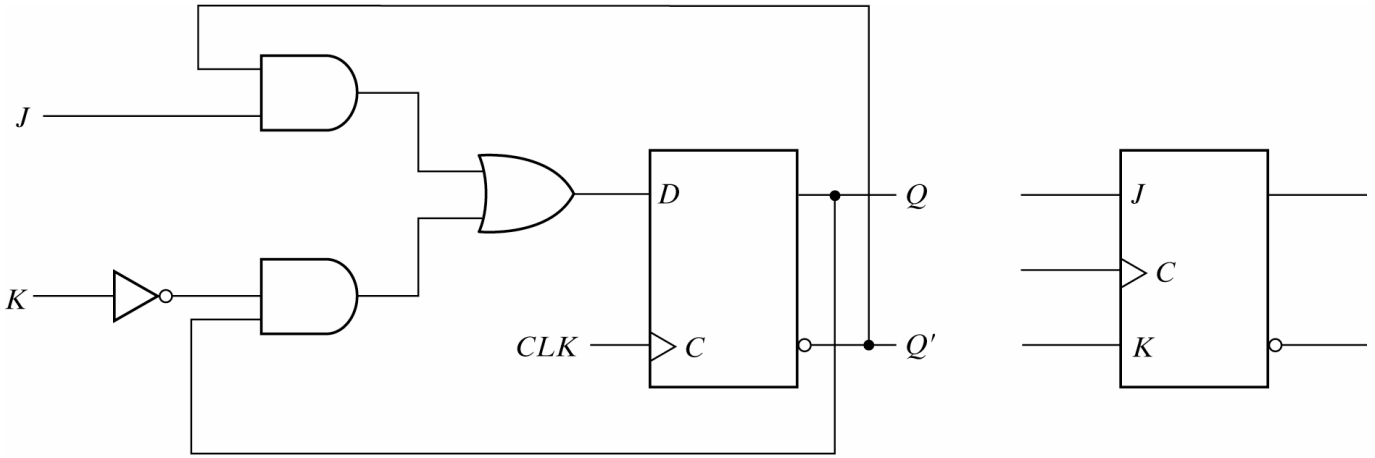


(a) Positive-edge (a) Negative-edge

JK Flip-Flop

- The JK flip-flop is similar to the RS flip-flop.
- But now rather than producing an indeterminate state the when both inputs are 1 the output is toggled or inverted.

J	K	Q_{t+1}	Q'_{t+1}
0	0	Q_t	Q'_t
0	1	0	1
1	0	1	0
1	1	Q'_t	Q_t



(a) Circuit diagram (b) Graphic symbol

JK Flip-Flop

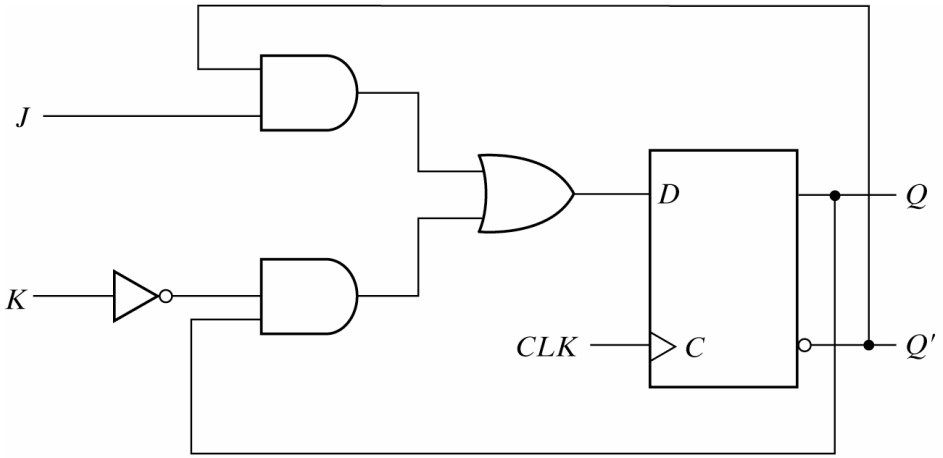
- When J=K=0

$$J.\overline{Q} + \overline{K}.Q$$

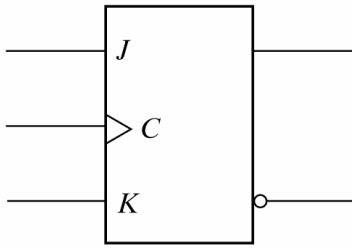
$$= 0.\overline{Q} + 1.Q = Q$$

and the previous state is remains at the output.

J	K	Q_{t+1}	Q'_{t+1}
0	0	Q_t	Q'_t
0	1	0	1
1	0	1	0
1	1	Q'_t	Q_t



(a) Circuit diagram



(b) Graphic symbol

JK Flip-Flop

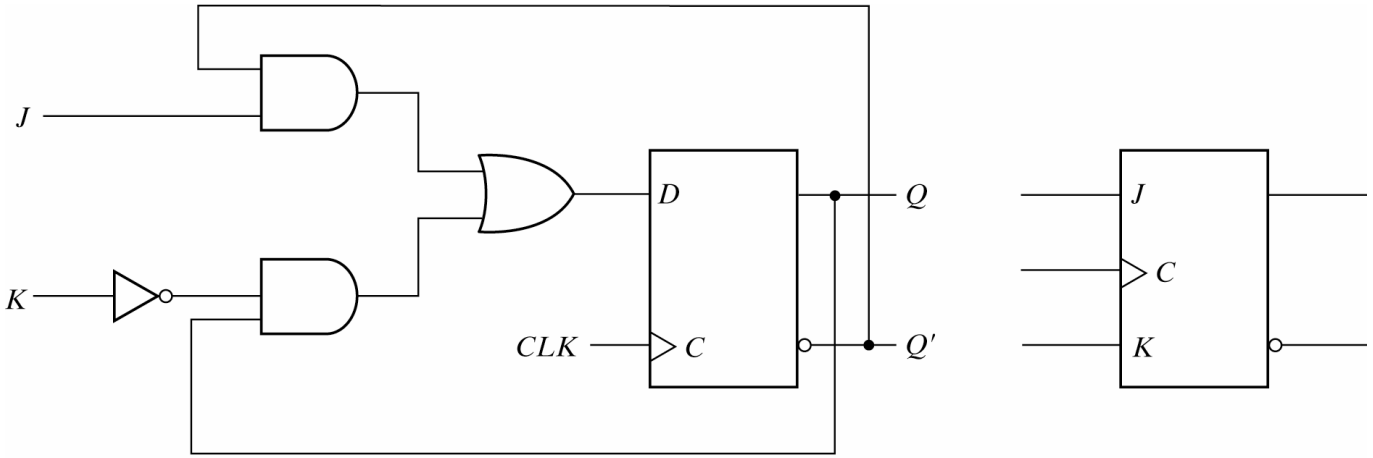
- When J=K=1

$$J.\overline{Q} + \overline{K}.Q$$

$$= 1.\overline{Q} + 0.Q = \overline{Q}$$

and the previous state is complemented.

J	K	Q_{t+1}	Q'_{t+1}
0	0	Q_t	Q'_t
0	1	0	1
1	0	1	0
1	1	Q'_t	Q_t

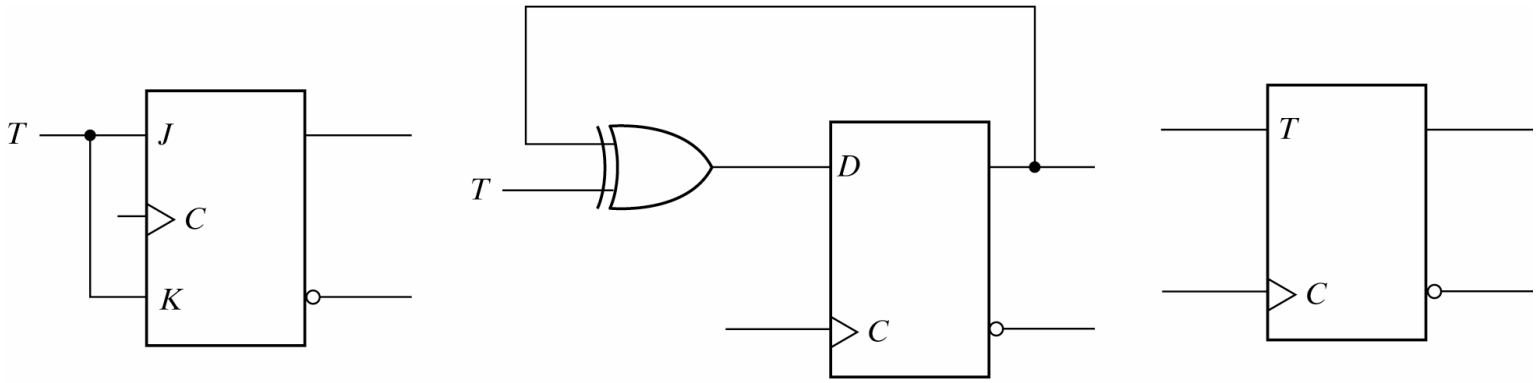


(a) Circuit diagram (b) Graphic symbol

T Flip-Flop

- The T (toggle) flip-flop complements the previous input upon each clock edge when T = 1.
- The output value is left unchanged when T = 0.

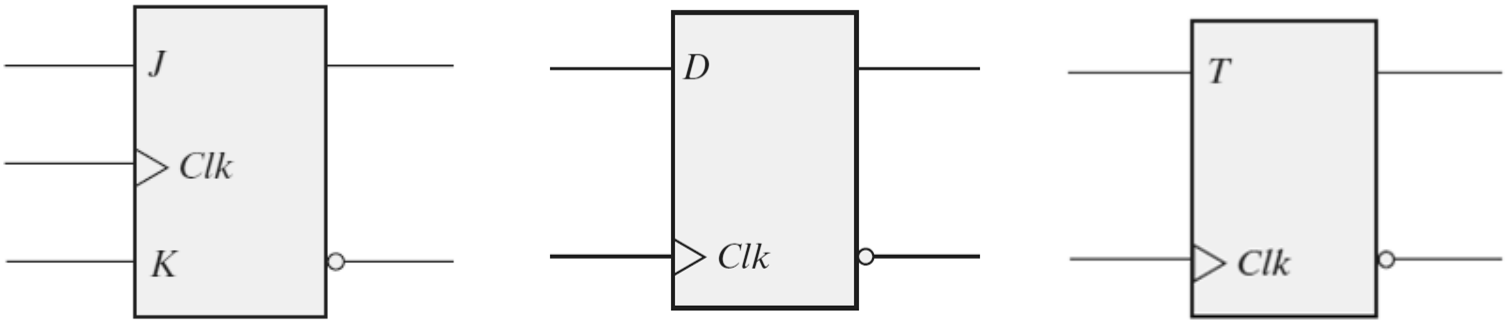
T	Q_{t+1}	Q'_{t+1}
0	Q_t	Q'_t
1	Q'_t	Q_t



(a) From *JK* flip-flop (b) From *D* flip-flop (c) Graphic symbol

Flip-Flops Summary

- There are three flip-flops widely used in digital systems: JK, D, and T flip-flops.



Characteristic Equations/Tables

- The properties of each flip-flop can be described algebraically using **Characteristic Equations**.

$$Q_{t+1} = J\overline{Q} + \overline{K}Q$$

$$Q_{t+1} = D$$

$$Q_{t+1} = T\overline{Q} + \overline{T}Q$$

These equations are equivalent to the truth tables associated with each flip-flop, which is called **Characteristic Tables**.

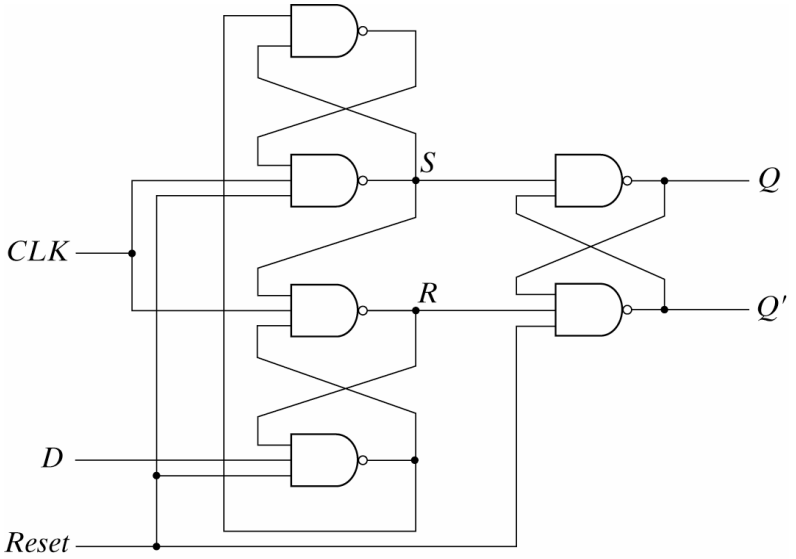
J	K	Q(t + 1)	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

D	Q(t + 1)	
0	0	Reset
1	1	Set

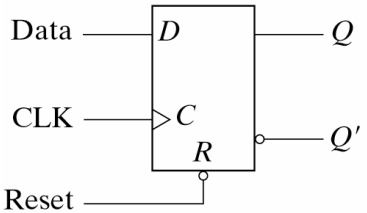
T	Q(t + 1)	
0	$Q(t)$	No change
1	$Q'(t)$	Complement

Asynchronous Reset Inputs

- In practical digital circuits flip-flops are often equipped with an asynchronous reset input.
- Upon power up of a digital circuit the flip-flops are in unknown states (as in the first step of the D flip-flop simulation).
- The reset input allows all flip-flops be brought a known state.



(a) Circuit diagram



(b) Graphic symbol

R	C	D	Q	Q′
0	X	X	0	1
1	↑	0	0	1
1	↑	1	1	0

(b) Function table