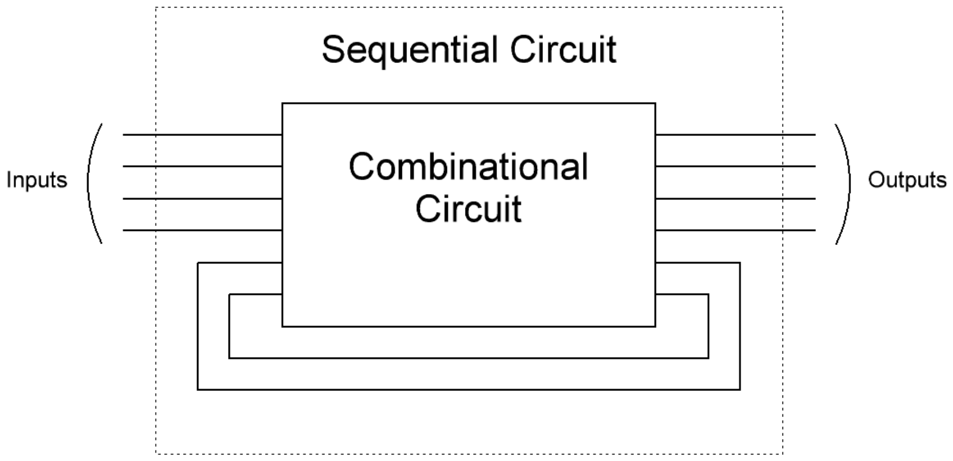
Chapter 2
Sequential Logic

Latest update: 31/01/2017

Table of Contents

I. Introduction

Sequential logic is the discipline used to design sequential circuits. A sequential circuit is a digital circuit whose outputs are not only functions of its present inputs, but also of its previous inputs. It is distinguishable from a combinational circuit by some of its outputs that are fed back to its inputs, which generates a form of memory element.

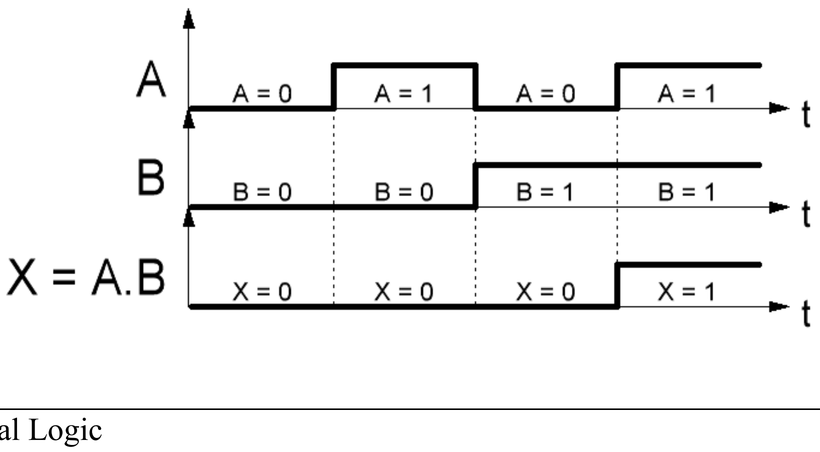


The basic building blocks of sequential circuits are flip-flops and latches.

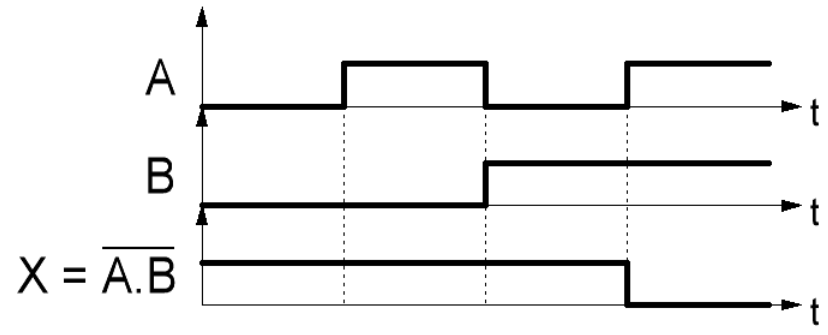
A flip-flop has its outputs affected by its inputs either at any time or only when a special input is triggered. In the former case, the flip-flop can also be called a 'latch'. In the latter case, the special input is called 'clock input'. In other words, a latch is a flip-flop without a clock input.

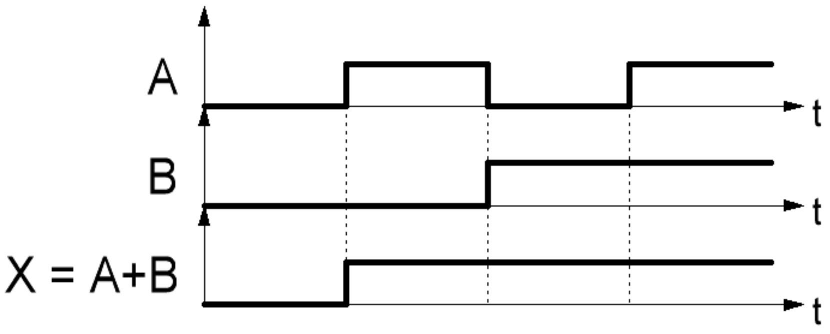
II. Timing Diagrams

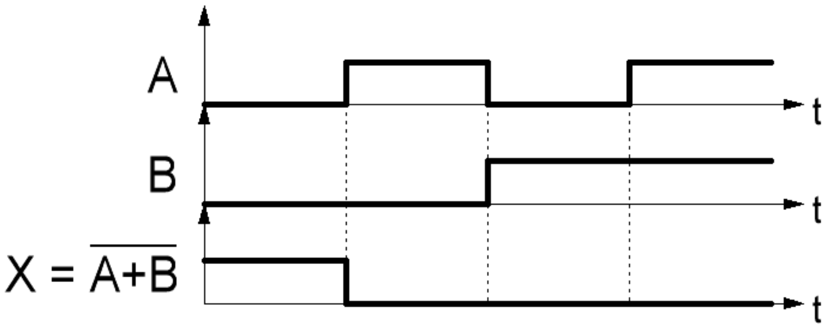
Until now, we have seen different ways to describe the behavior of combinational circuits: Boolean expressions, truth tables and Karnaugh maps. However, these representations are not sufficient when it comes to representing signals in the time domain. Therefore, we need to introduce timing diagrams, which give an overview of the timing relationships. For example, here is the timing diagram of an AND gate:

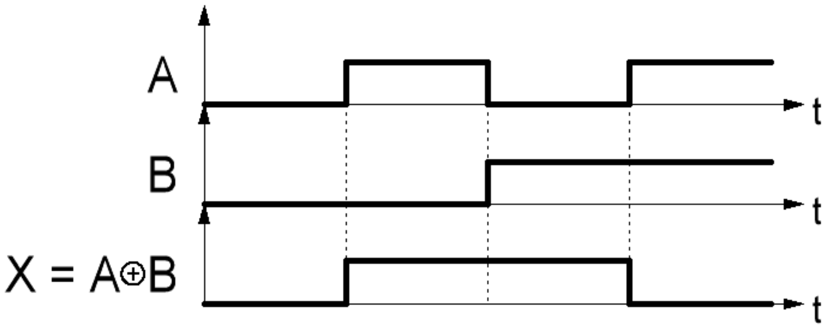


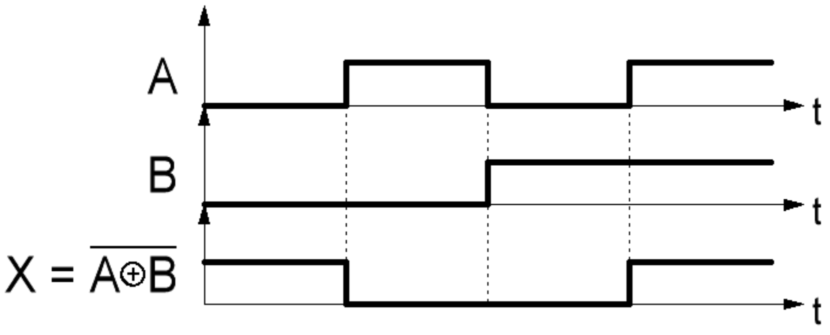
Here are the timing diagrams of some other gates:



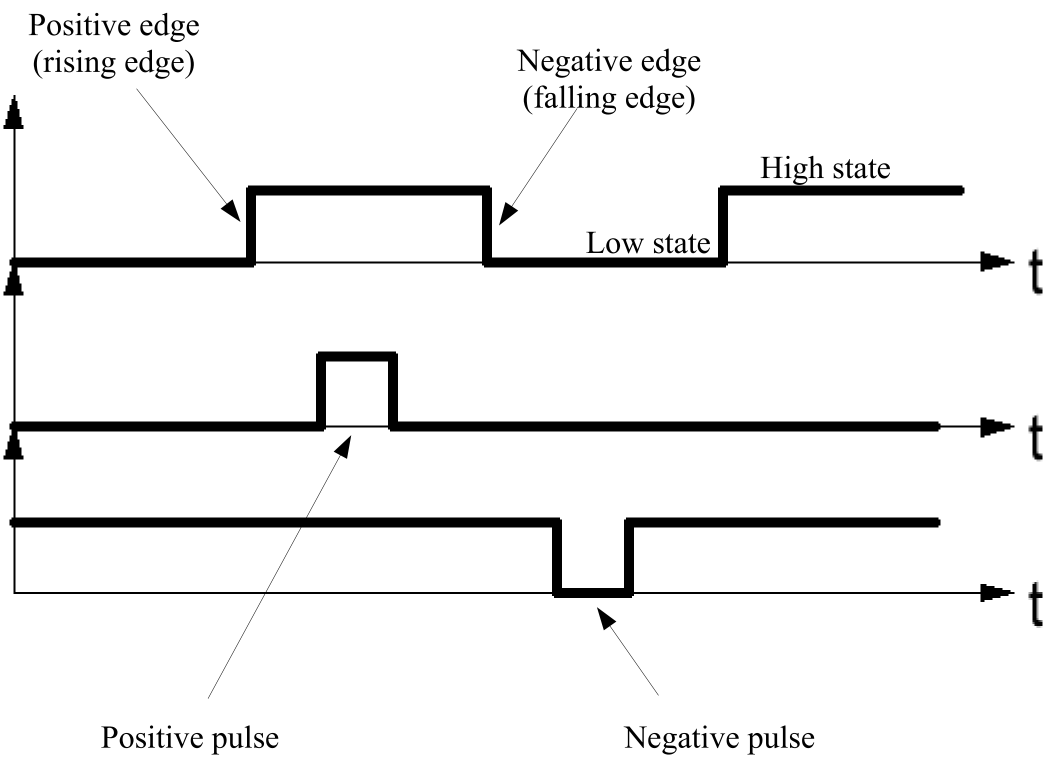








Terminology

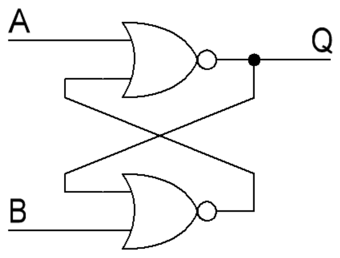


III. The RS Flip-Flop

1. The RS Latch

1.1. The RS NOR Latch

Let us consider the two following cross-coupled NOR gates:



Considering that *A* and *B* are the inputs and *Q* the output, the truth table of this circuit is:

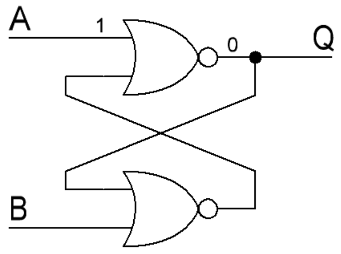
	A	B	Q	
①	0	0	q	← *q* = value of *Q* just before *A* and *B* are both set to 0.
②	0	1	1	
③	1	0	0	
④	1	1	0	

The last three lines can be easily determined from the truth table of a NOR gate:

X	Y	$\overline{X+Y}$
0	0	1
0	1	0
1	0	0
1	1	0

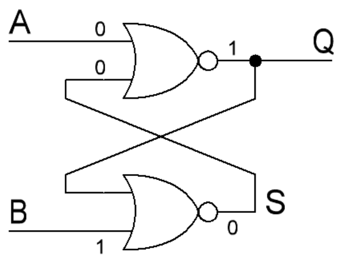
It can be seen that the output of a NOR gate is 0, if either of its inputs is 1.

Explanation for lines ③ and ④ (A = 1):



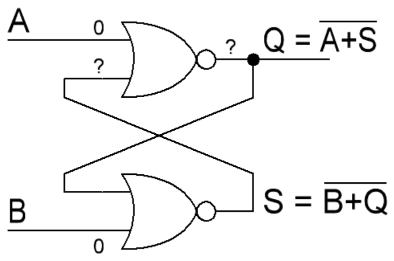
When the *A* input is 1, the *Q* output is 0. Knowing the other input of the NOR gate is not necessary.

Explanation for line ② (A = 0, B = 1):



Let us call the output of the second NOR gate *S*.
When the *B* input is 1, the *S* output is 0. Knowing the other input of the NOR gate is not necessary. Therefore $Q = \overline{0+0} = 1$.

Explanation for line ① (A = 0, B = 0):



This case is slightly more difficult to investigate. *Q* must be known to determine *S* ($S = \overline{B + Q}$), and *S* must be known to determine *Q* ($Q = \overline{A + S}$). To sum up, *Q* must be known to determine *Q*.

Consequently, we have: $Q=\overline{A+\overline{B+Q}}=\overline{A}.(B+Q)$

To be more precise, we should actually say that **the previous value of *Q*** must be known to determine **the new value of *Q***. So, let us call *q* the previous value of *Q*; that is to say, the value of *Q* just before *A* and *B* are both set to 0.

Therefore, we can write down that: $\mathbf{Q = \overline{A}.(B + q)}$

Moreover, we know that in this case *A* = *B* = 0.

Therefore:
$Q = \overline{0}.(0 + q)$
$Q = 1.(q)$
$\mathbf{Q = q}$

In conclusion:
- If *Q* is 0 just before *A* and *B* are both set to 0, then *Q* remains 0.
- If *Q* is 1 just before *A* and *B* are both set to 0, then *Q* remains 1.

This state can be called: **'no change state'**.

Now let us watch carefully the truth table of this circuit.

A	B	Q	
0	0	q	← *q* = value of *Q* just before *A* and *B* are both set to 0.
0	1	1	
1	0	0	
1	1	0	

Assuming that *A* and *B* are active-high:

- When *A* is active and *B* inactive, the output is reset to 0 (*A* acts as a reset input).
- When *A* is inactive and *B* active, the output is set to 1 (*B* acts as a set input).
- When both *A* and *B* are inactive, the output does not change (no change state).
- When both *A* and *B* are active, the output is reset to 0 (the reset input is predominant).

Therefore, the *A* and *B* inputs can be renamed *R* and *S* respectively:

- *R* stands for 'Active-High **R**eset Input'
- *S* stands for 'Active-High **S**et Input'.

This circuit is called an 'active-high RS latch' or an 'RS latch'.

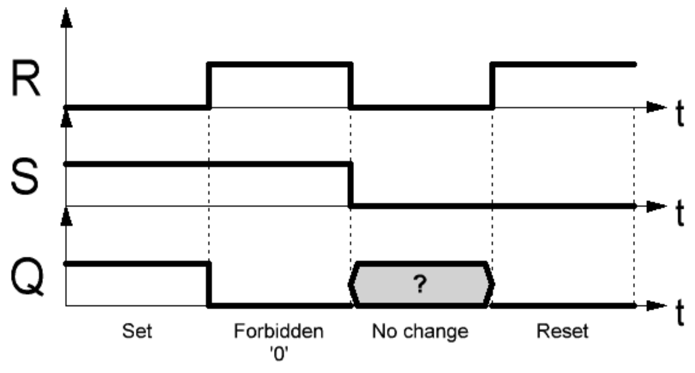
Moreover, even if the reset input is predominant, it is strongly advised to avoid the state where both inputs are active ($R = S = 1$) because it is not coherent to set and reset an output at the same time.

As a result, we have a new version of the truth table:

R	S	Q	
0	0	q	← No change
0	1	1	← Set
1	0	0	← Reset
1	1	x	← Forbidden

Caution! In this truth table, the character 'x' does not mean undefined or don't care conditions. The value of *Q* is defined and is 0. Anyway, this state is forbidden for the reason previously said.

In fact, there is another reason for avoiding this forbidden state. For instance, have a look at the following timing diagram:



An undefined state occurs when the latch goes from the forbidden state to the no change state; that is to say, when both *R* and *S* go from 1 to 0. The reason is that, physically speaking, *R* and *S* cannot change exactly at the same time, and even if they could, the transistors that make up the gates are not perfectly identical. Consequently, one signal (*R* or *S*) is always taken into account before the other.

The latch cannot go directly from the forbidden state to the no change state.

R	S	Q
0	0	q
0	1	1
1	0	0
1	1	x

First, it goes through either:

the reset state

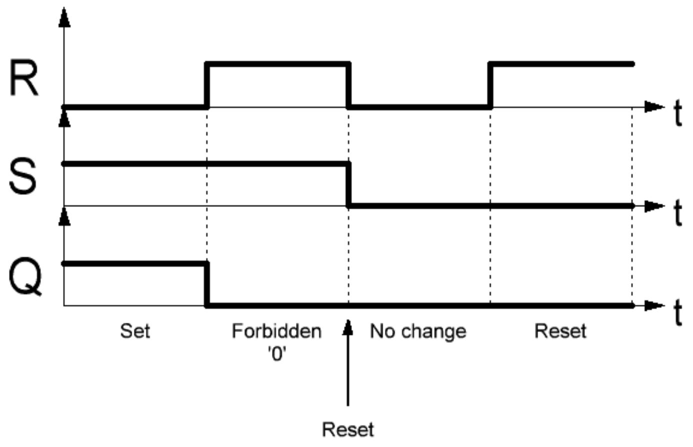
R	S	Q
0	0	q
0	1	1
1	0	0
1	1	x

or

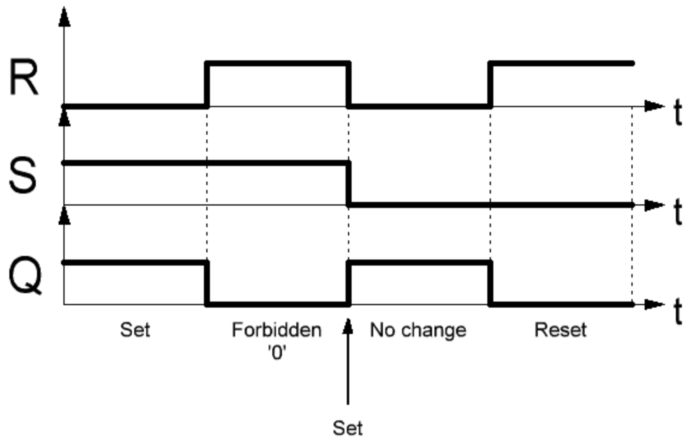
the set state

R	S	Q
0	0	q
0	1	1
1	0	0
1	1	x

In the former case (the reset state), the *Q* output is first reset to 0 before reaching the no change state:

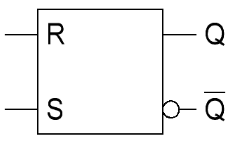


In the latter case (the set state), the *Q* output is first set to 1 before reaching the no change state:



The problem is that theory is not enough to predict which case will occur (reset or set). The only way to find out is to physically implement the circuit.

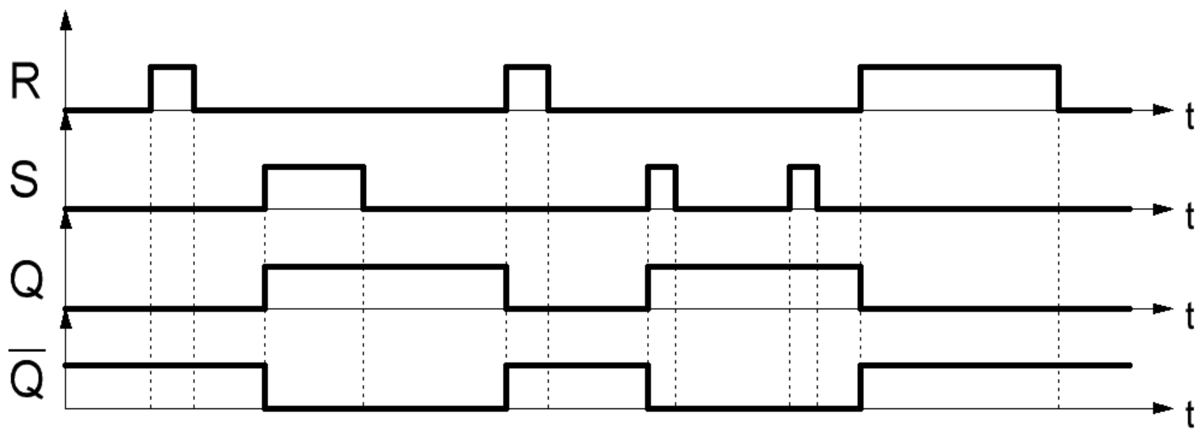
For the sake of convenience, we usually use the symbol of the RS latch instead of the two NOR gates. Furthermore, most RS latches are provided with a complemented output. Here are the symbol of an RS latch and its truth table:



R	**S**	**Q**	**$\overline{Q}$**	
0	0	q	$\overline{q}$	← No change
0	1	1	0	← Set
1	0	0	1	← Reset
1	1	x	x	← Forbidden

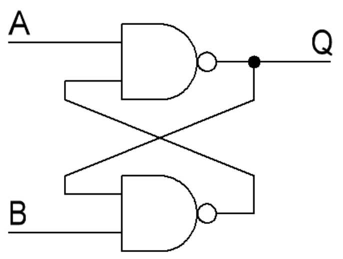
q = value of *Q* just before *R* and *S* are both set to 0.

Finally, here is an example of a timing diagram for the RS latch:



1.2. The Active-Low RS NAND Latch

Let us consider the two following cross-coupled NAND gates:



Considering that *A* and *B* are the inputs and *Q* the output, the truth table of this circuit is:

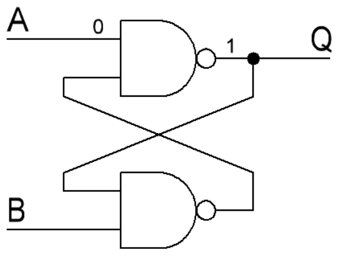
	A	B	Q	
①	0	0	1	
②	0	1	1	
③	1	0	0	
④	1	1	q	← *q* = value of *Q* just before *A* and *B* are both set to 1.

The first three lines can be easily determined from the truth table of a NAND gate:

X	Y	$\overline{X.Y}$
0	0	1
0	1	1
1	0	1
1	1	0

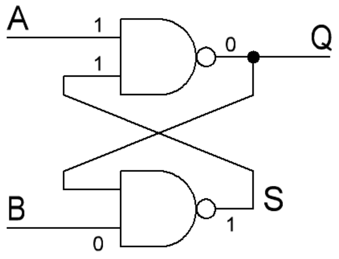
It can be seen that the output of a NAND gate is 1, if either of its inputs is 0.

Explanation for lines ① and ② (A = 0):



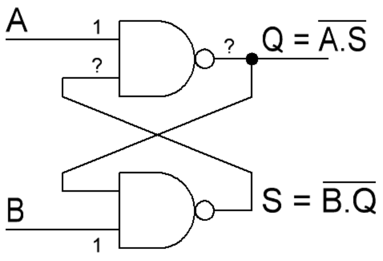
When the *A* input is 0, the *Q* output is 1. Knowing the other input of the NAND gate is not necessary.

Explanation for line ③ (A = 1, B = 0):



Let us call the output of the second NAND gate *S*.
When the *B* input is 0, the *S* output is 1. Knowing the other input of the NAND gate is not necessary. Therefore $Q = \overline{1.1} = 0$.

Explanation for line ④ (A = 1, B = 1):



This case is slightly more difficult to investigate. *Q* must be known to determine *S* ($S = \overline{B.Q}$), and *S* must be known to determine *Q* ($Q = \overline{A.S}$). To sum up, *Q* must be known to determine *Q*.

Consequently, we have: $Q = \overline{A.\overline{B.Q}} = \overline{A} + B.Q$

To be more precise, we should actually say that **the previous value of *Q*** must be known to determine **the new value of *Q***. So, let us call *q* the previous value of *Q*; that is to say, the value of *Q* just before *A* and *B* are both set to 1.

Therefore, we can write down that: $\mathbf{Q = \overline{A} + B.q}$

Moreover, we know that in this case *A* = *B* = 1.

Therefore:
$Q = \overline{1} + 1.q$
$Q = 0 + q$
$\mathbf{Q = q}$

In conclusion:

- If *Q* is 0 just before *A* and *B* are both set to 1, then *Q* remains 0.
- If *Q* is 1 just before *A* and *B* are both set to 1, then *Q* remains 1.

This state can be called: **‘no change state’**.

Now let us watch carefully the truth table of this circuit.

A	B	Q	
0	0	1	
0	1	1	
1	0	0	
1	1	q	← *q* = value of *Q* just before *A* and *B* are both set to 1.

Assuming that *A* and *B* are active-low:

- When *A* is active and *B* inactive, the output is set to 1 (*A* acts as a set input).
- When *A* is inactive and *B* active, the output is reset to 0 (*B* acts as a reset input).
- When both *A* and *B* are inactive, the output does not change (no change state).
- When both *A* and *B* are active, the output is set to 1 (the set input is predominant).

Therefore, the *A* and *B* inputs can be renamed $\overline{S}$ and $\overline{R}$ respectively:

- $\overline{R}$ stands for ‘Active-Low **R**eset Input’
- $\overline{S}$ stands for ‘Active-Low **S**et Input’.

This circuit is called an ‘active-low RS latch’ or a ‘$\overline{\textbf{RS}}$ latch’.

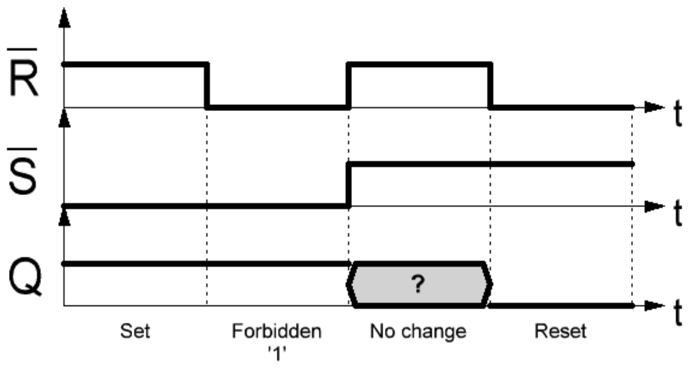
Moreover, even if the set input is predominant, it is strongly advised to avoid the state where both inputs are active ($\overline{R} = \overline{S} = 0$) because it is not coherent to set and reset an output at the same time.

As a result, we have a new version of the truth table:

(B) $\overline{\textbf{R}}$	(A) $\overline{\textbf{S}}$	Q	
0	0	x	← Forbidden
0	1	0	← Reset
1	0	1	← Set
1	1	q	← No change

Caution! In this truth table, the character ‘x’ does not mean undefined or don’t care conditions. The value of *Q* is defined and is 1. Anyway, this state is forbidden for the reason previously said.

In fact, there is another reason for avoiding this forbidden state. For instance, have a look at the following timing diagram:



An undefined state occurs when the latch goes from the forbidden state to the no change state; that is to say, when both $\overline{R}$ and $\overline{S}$ go from 0 to 1. The reason is that, physically speaking, $\overline{R}$ and $\overline{S}$ cannot change exactly at the same time, and even if they could, the transistors that make up the gates are not perfectly identical. Consequently, one signal ($\overline{R}$ or $\overline{S}$) is always taken into account before the other.

The latch cannot go directly from the forbidden state to the no change state.

$\overline{\mathbf{R}}$	$\overline{\mathbf{S}}$	**Q**
0	0	x
0	1	0
1	0	1
1	1	q

First, it goes through either:

the reset state

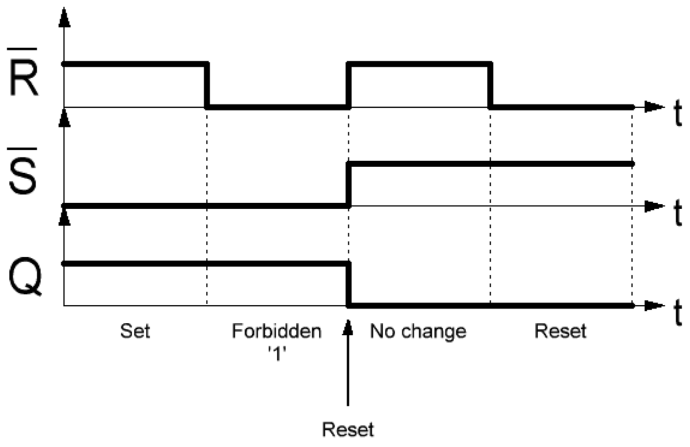
$\overline{\mathbf{R}}$	$\overline{\mathbf{S}}$	**Q**
0	0	x
0	1	0
1	0	1
1	1	q

or

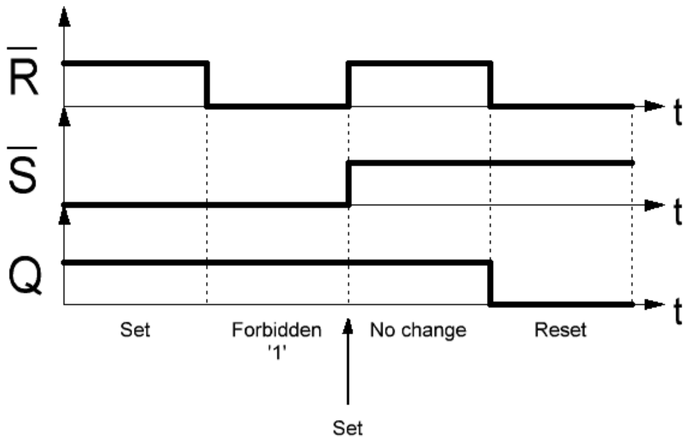
the set state

$\overline{\mathbf{R}}$	$\overline{\mathbf{S}}$	**Q**
0	0	x
0	1	0
1	0	1
1	1	q

In the former case (the reset state), the *Q* output is first reset to 0 before reaching the no change state:

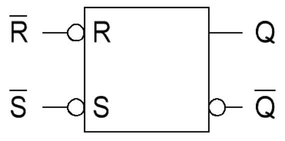


In the latter case (the set state), the *Q* output is first set to 1 before reaching the no change state:



The problem is that theory is not enough to predict which case will occur (reset or set). The only way to find out is to physically implement the circuit.

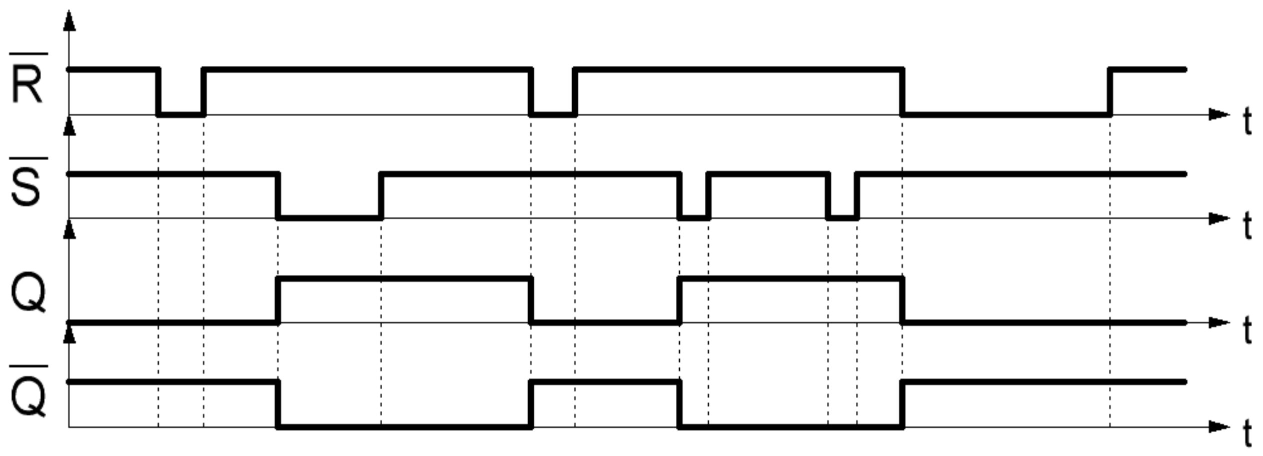
For the sake of convenience, we usually use the symbol of the active-low RS latch instead of the two NAND gates. Furthermore, most active-low RS latches are provided with a complemented output. Here are the symbol of an active-low RS latch and its truth table:



$\overline{R}$	$\overline{S}$	Q	$\overline{Q}$	
0	0	x	x	← Forbidden
0	1	0	1	← Reset
1	0	1	0	← Set
1	1	q	$\overline{q}$	← No change

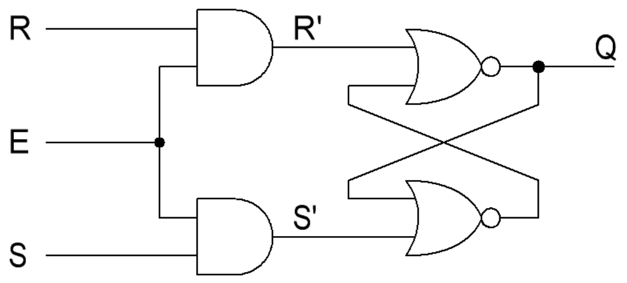
q = value of *Q* just before *R* and *S* are both set to 1.

Finally, here is an example of a timing diagram for the active-low RS latch:



2. The Gated RS Latch

Let us consider the following circuit diagram:



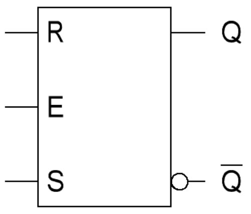
R' and *S'* are respectively the reset and set inputs of an active-high RS latch (made up of two cross-coupled NOR gates).

E stands for 'Enable' and is active-high:

- When *E* is 1, then *R'* = *R* and *S'* = *S*. Therefore this circuit behaves like an active-high RS latch.
- When *E* is 0, then *R'* = *0* and *S'* = *0*. This circuit is in a no change state.

E	**R'**	**S'**	**Behaviour of the circuit**
0	0	0	No change state
1	R	S	Active-high RS latch

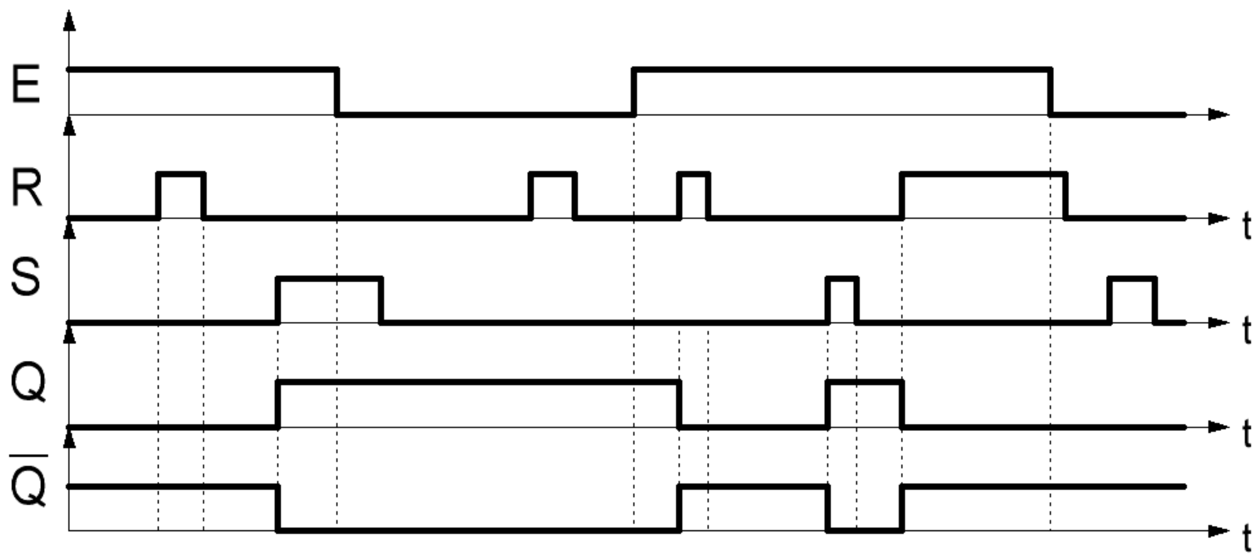
This circuit is called a **'gated RS latch'**. Here are its symbol and its truth table:



E	**R**	**S**	**Q**	$\overline{\mathbf{Q}}$	
0	Φ	Φ	q	$\overline{q}$	← No change
1	0	0	q	$\overline{q}$	← No change
1	0	1	1	0	← Set
1	1	0	0	1	← Reset
1	1	1	x	x	← Forbidden

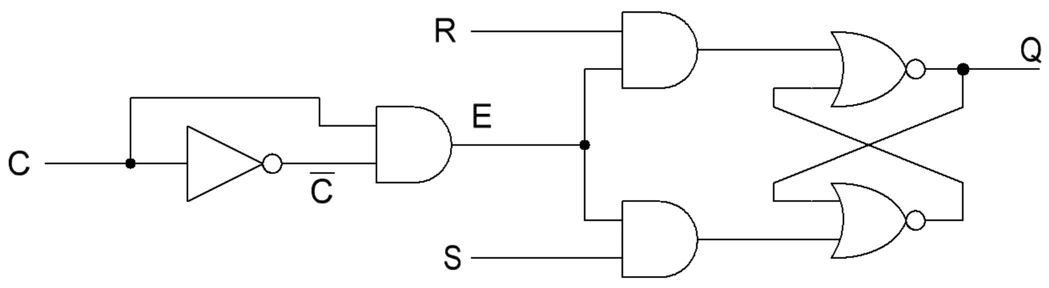
q = value of *Q* just before the latch goes into the no change state.
The character 'Φ' means 'don't care'.
The character 'x' means 'forbidden'.

Here is an example of a timing diagram for the gated RS latch:

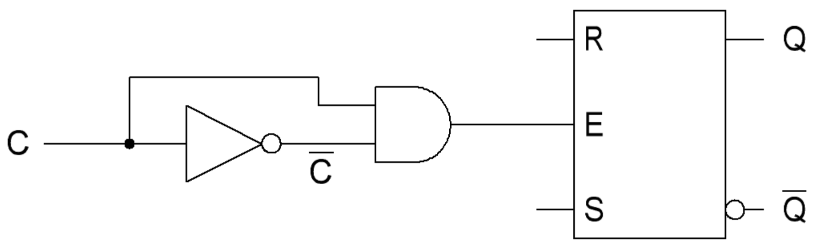


3. The Positive-Edge-Triggered RS Flip-Flop

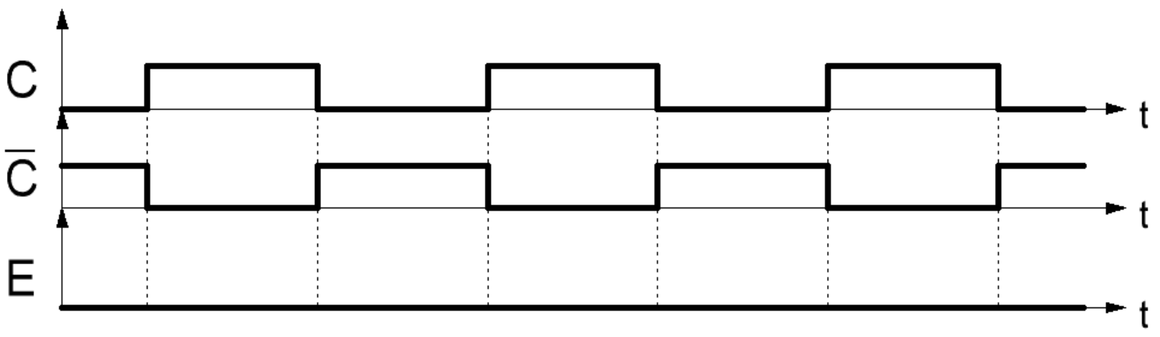
Let us consider the following circuit:



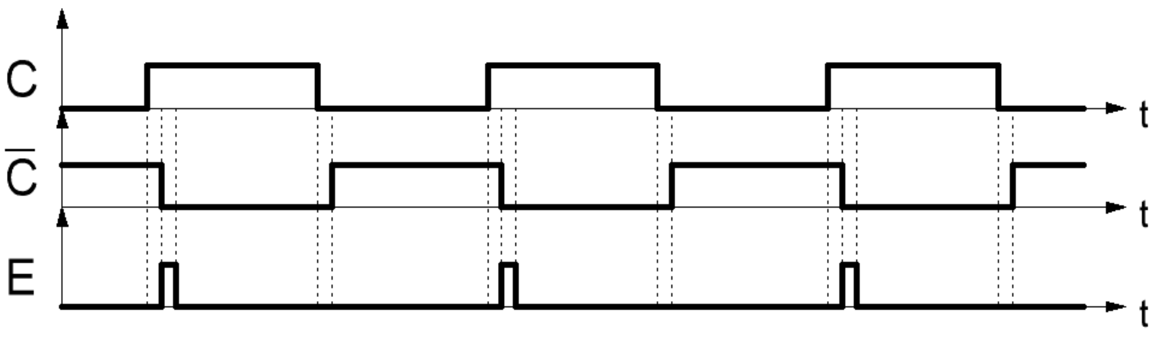
On the right side, we can identify a gated RS latch:



Assuming that the gates are ideal, let us draw the timing diagrams of **C**, $\overline{\mathbf{C}}$ and **E**.



It seems that *E* is always 0, meaning the gated RS latch is always in its no change state. In fact, it is not as simple as that. In practice, each gate needs a latency, which is a time delay between stimulation and response.

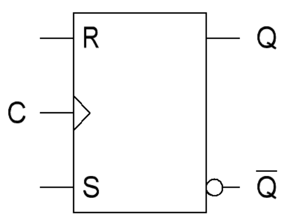


The NOT gate does not change immediately after a transition of its input; a short time delay (in the region of a few nanoseconds) appears between the transitions of *C* and $\overline{C}$. Therefore, these two signals are both ones momentarily. In addition, the AND gate also has a latency. So, the *E* output is set to one a short time later. In other words, *E* is set to one momentarily just after the positive edge of *C*.

Consequently, **the outputs of this circuit can change only on the positive edge of *C*.**

This circuit is a **'positive-edge-triggered RS flip-flop'** and the *C* input is called **'clock input'**.

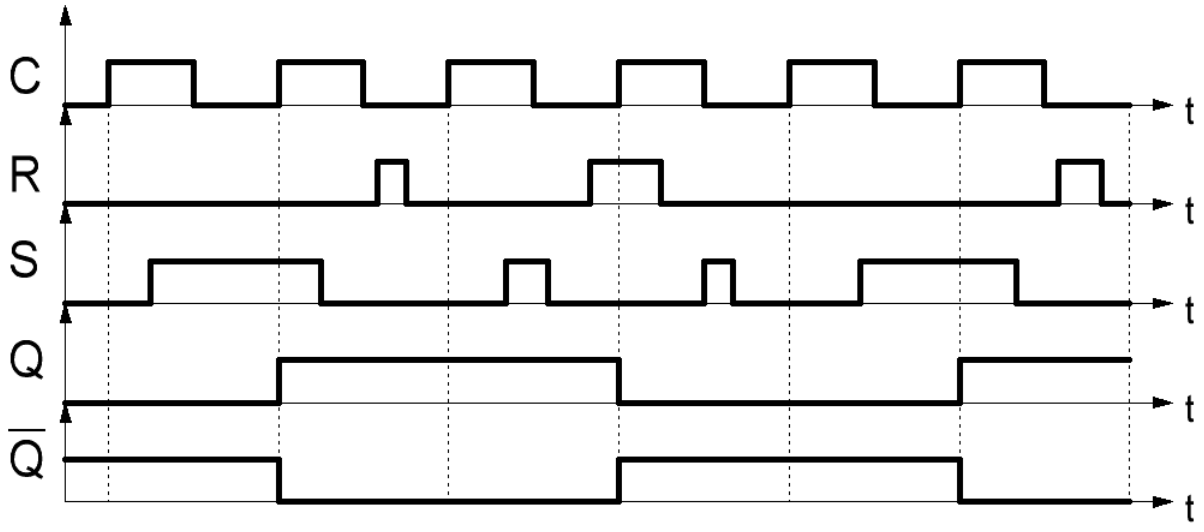
Here are its symbol and its truth table:



C	**R**	**S**	**Q**	$\overline{\mathbf{Q}}$	
↑	0	0	q	$\overline{q}$	← No change
↑	0	1	1	0	← Set
↑	1	0	0	1	← Reset
↑	1	1	x	x	← Forbidden

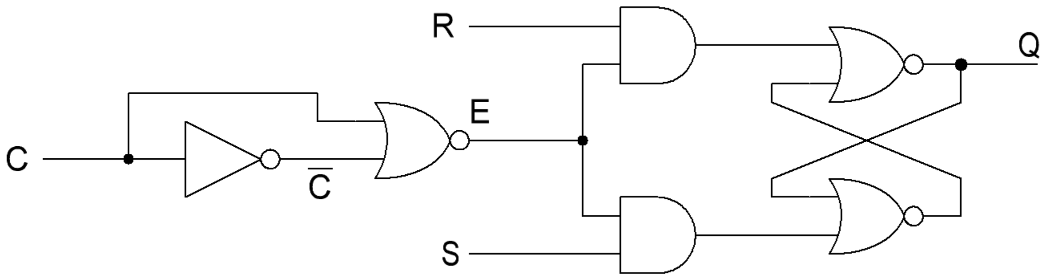
q = value of *Q* just before the positive edge of *C*.

Finally, here is an example of a timing diagram for the positive-edge-triggered RS flip-flop:

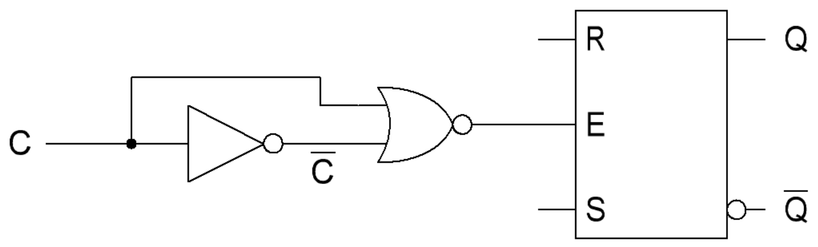


4. The Negative-Edge-Triggered RS Flip-Flop

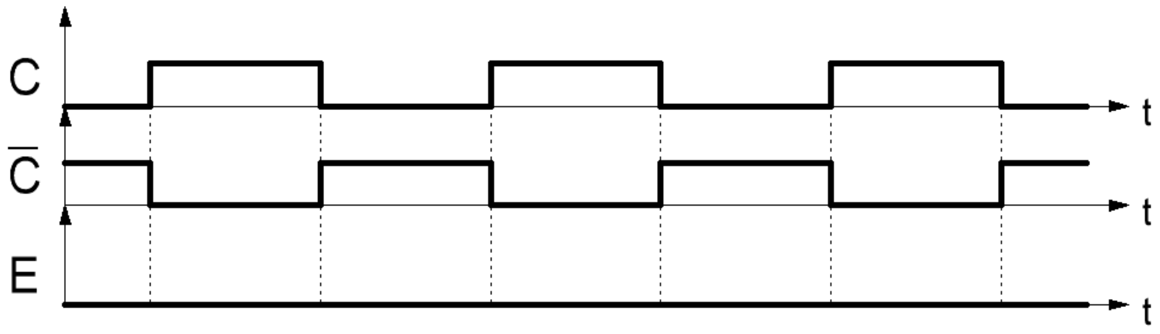
In the same way, we can design a negative-edge-triggered RS flip-flop:



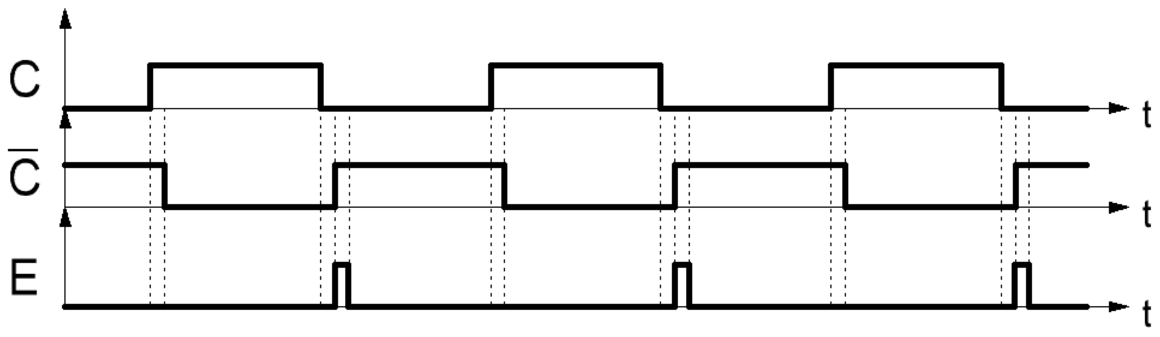
On the right side, we can identify a gated RS latch:



Assuming that the gates are ideal, let us draw the timing diagrams of **C**, $\overline{\mathbf{C}}$ and **E**.



It seems that *E* is always 0, meaning the gated RS latch is always in its no change state. But as said previously, each gate has a latency.

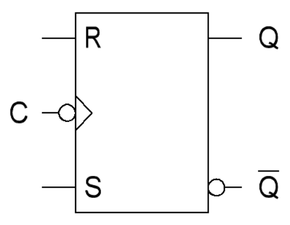


The NOT gate does not change immediately after a transition of its input; a short time delay (in the region of a few nanoseconds) appears between the transitions of *C* and $\overline{C}$. Therefore, these two signals are both zeros momentarily. In addition, the NOR gate also has a latency. So, the *E* output is set to one a short time later. In other words, *E* is set to one momentarily just after the negative edge of *C*.

Consequently, **the outputs of this circuit can change only on the negative edge of *C*.**

This circuit is a **'negative-edge-triggered RS flip-flop'** and the *C* input is called **'clock input'**.

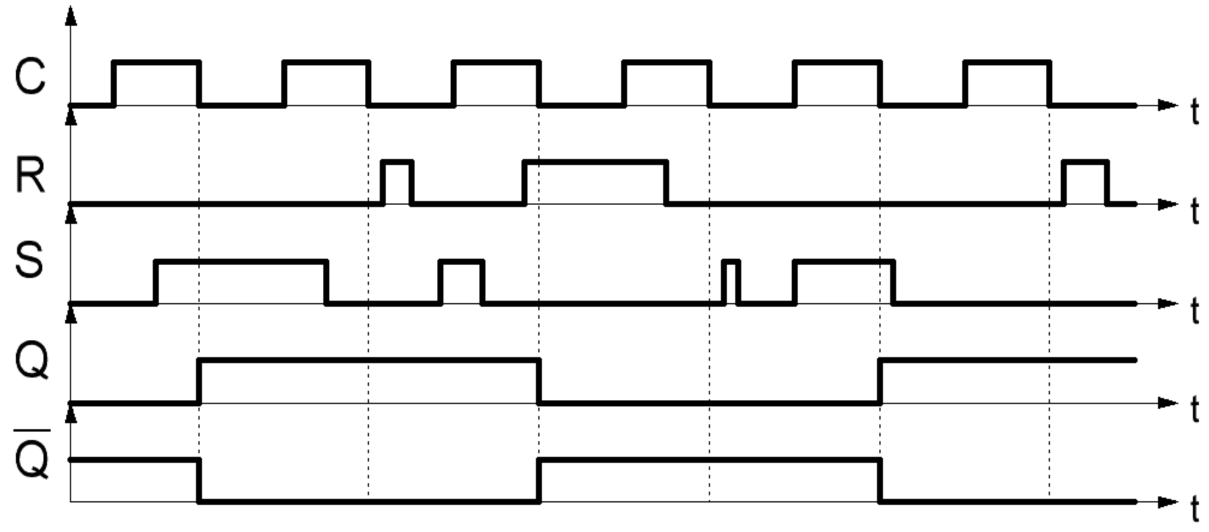
Here are its symbol and its truth table:



C	R	S	Q	$\overline{Q}$	
↓	0	0	q	$\overline{q}$	← No change
↓	0	1	1	0	← Set
↓	1	0	0	1	← Reset
↓	1	1	x	x	← Forbidden

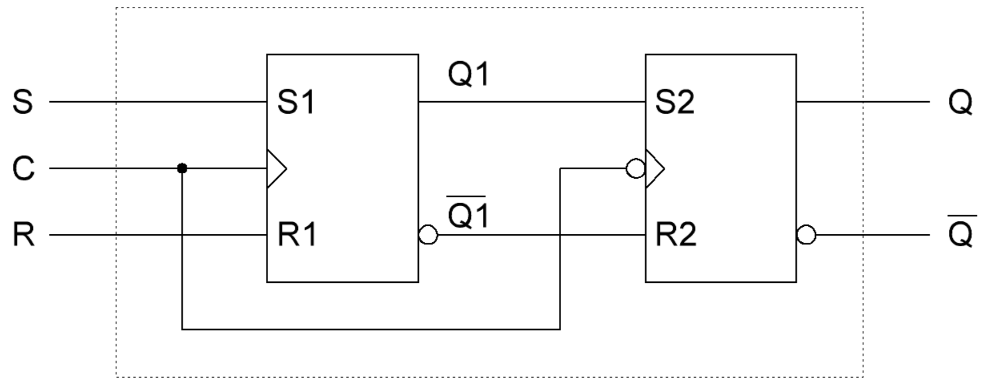
q = value of *Q* just before the negative edge of *C*.

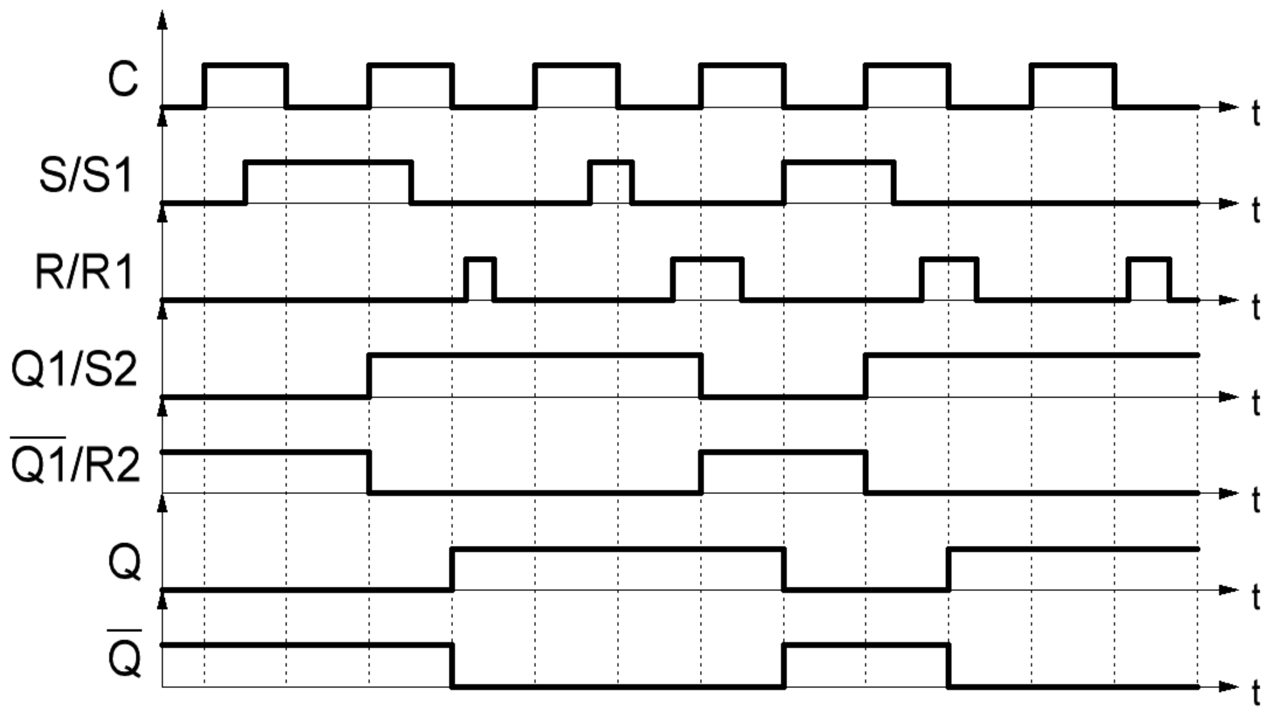
Finally, here is an example of a timing diagram for the negative-edge-triggered RS flip-flop:



5. The Master-Slave RS Flip-Flop

Let us consider the following circuit and its timing diagram:

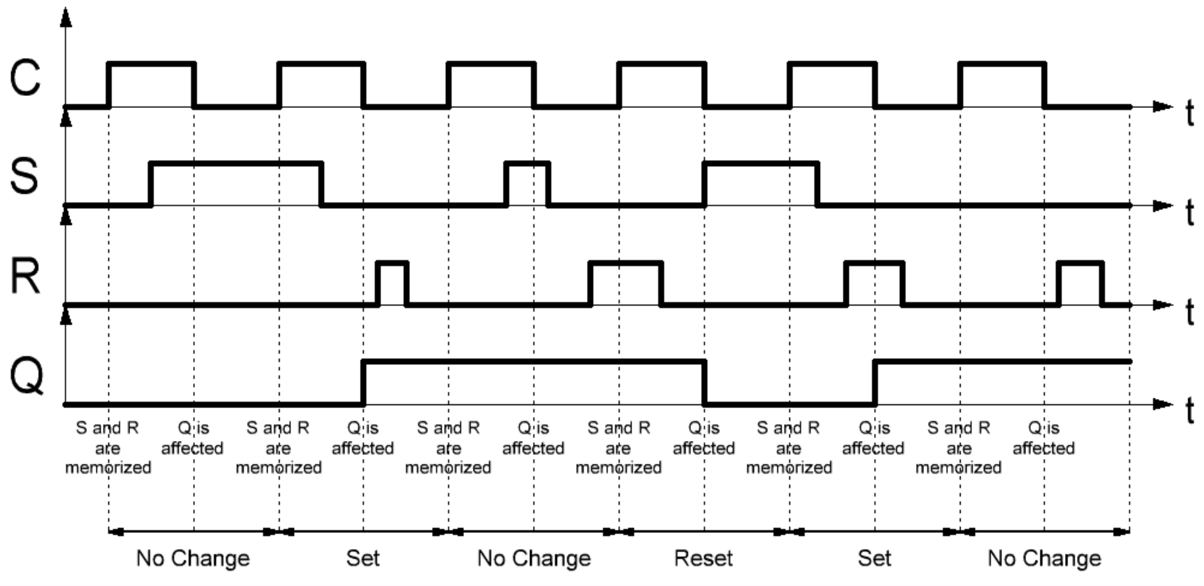




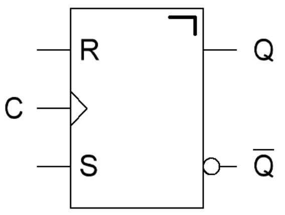
- The first flip-flop is the master flip-flop. Its outputs (*Q1*/*S2* and $\overline{Q1}$/*R2*) are affected on each positive edge of the clock signal according to the present values of *S* and *R*. But at this stage, the *Q* output has not been affected yet.

- The second flip-flop is the slave flip-flop. Its outputs (*Q* and $\overline{Q}$) are affected on each negative edge of the clock signal according to the present values of *Q1*/*S2* and $\overline{Q1}$/*R2*; that is to say, according to the values of *S* and *R* at the previous positive edge.

Let us focus on the *C*, *S*, *R* inputs and the *Q* output:



This circuit is a **'positive-edge-triggered master-slave RS flip-flop'** or a **'master-slave RS flip-flop'**. Here are its symbol and its truth table:



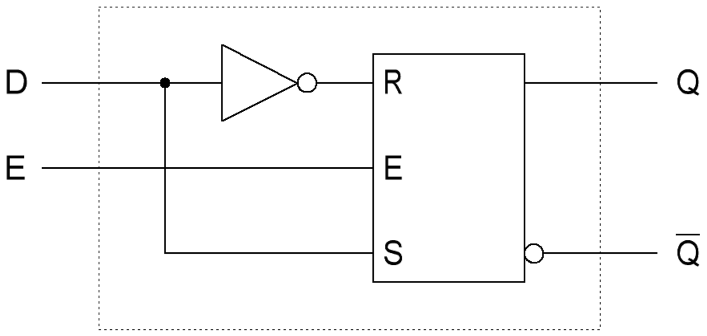
C	**R**	**S**	**Q**	$\overline{\mathbf{Q}}$	
⎍	0	0	q	$\overline{q}$	← No change
⎍	0	1	1	0	← Set
⎍	1	0	0	1	← Reset
⎍	1	1	x	x	← Forbidden

q = value of *Q* just before the positive edge of *C*.

IV. The D Flip-flop

1. The Gated D Latch

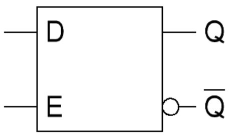
Let us consider the following circuit:



- When $E = 0 \rightarrow$ the outputs do not change (whatever the value of D).
- When $E = 1$ and $D = 0 \rightarrow R = 1$ and S = 0 $\rightarrow Q = 0$.
- When $E = 1$ and $D = 1 \rightarrow R = 0$ and S = 1 $\rightarrow Q = 1$.

Therefore, when $E = 1$, $Q = D$.

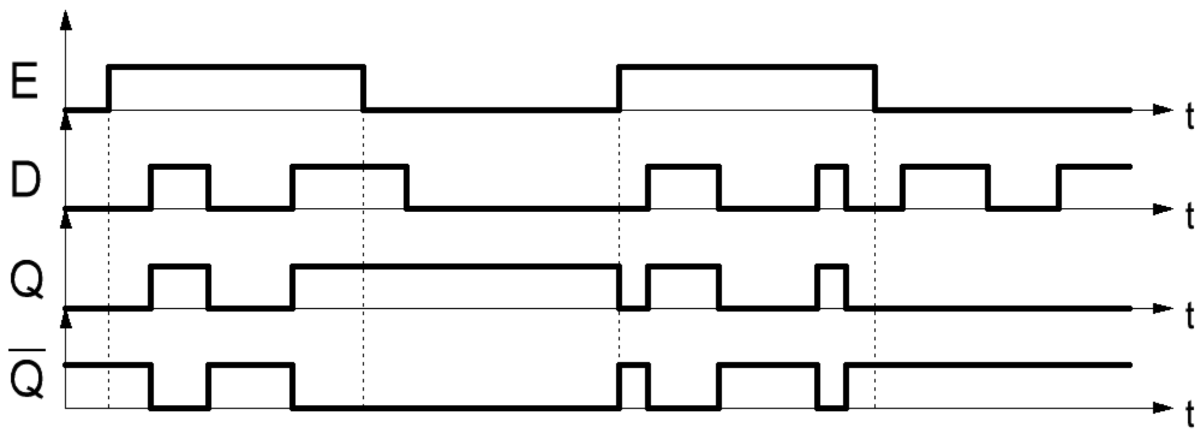
This circuit is a **'gated D latch'** or a **'transparent D latch'** (*D* stands for data). Here are its symbol and its truth table:



E	**D**	**Q**	$\overline{\mathbf{Q}}$	
0	Φ	q	$\overline{q}$	← No change
1	0	0	1	
1	1	1	0	

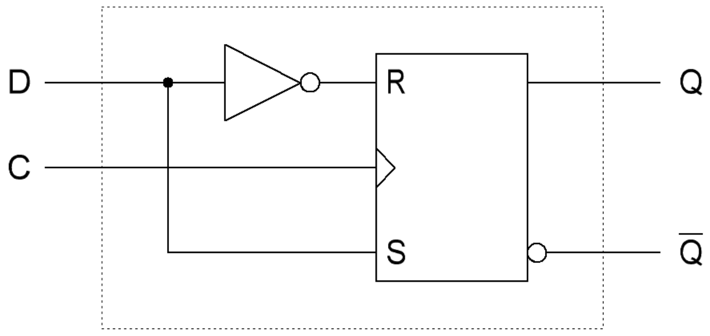
q = value of *Q* just before the latch goes into the no change state.
The character 'Φ' means 'don't care'.

Finally, here is an example of a timing diagram for the gated D latch:



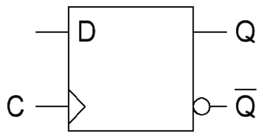
2. The Positive-Edge-Triggered D Flip-Flop

Let us consider the following circuit:



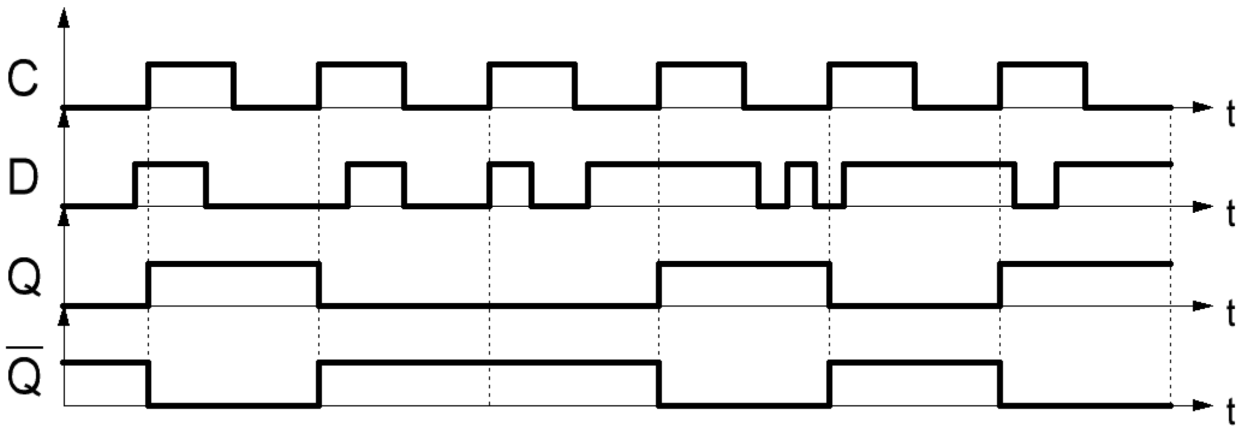
This circuit is similar to the gated D latch except that the *Q* and $\overline{Q}$ outputs are affected only on each positive edge of the clock signal. In other words, the *Q* output copies the value of *D* on each positive edge of the clock (the outputs do not change between two positive edges).

Therefore, this circuit is a **'positive-edge-triggered D flip-flop'**. Here are its symbol and its truth table:



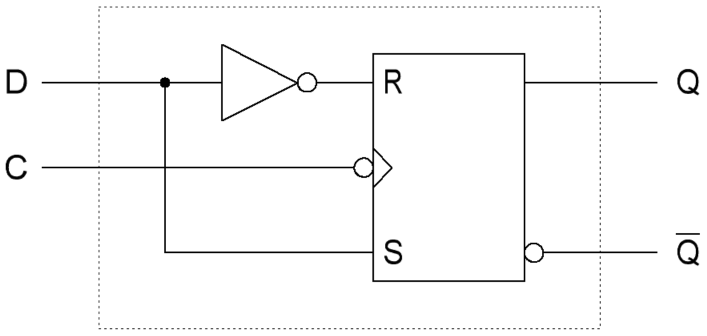
C	D	Q	$\overline{Q}$
↑	0	0	1
↑	1	1	0

Finally, here is an example of a timing diagram for the positive-edge-triggered D latch:



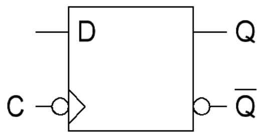
3. The Negative-Edge-Triggered D Flip-Flop

Let us consider the following circuit:



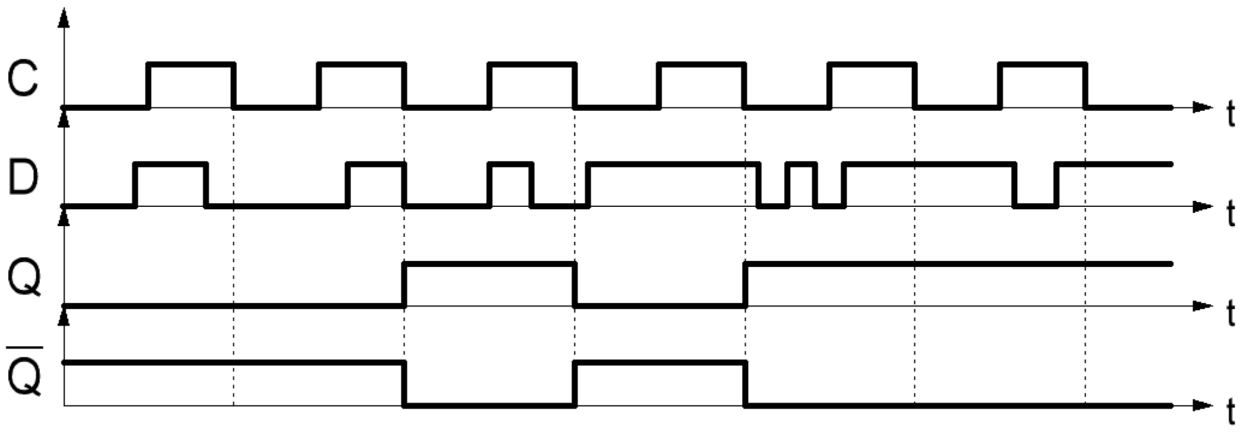
This circuit is similar to the positive-edge-triggered D flip-flop except that the *Q* and $\overline{Q}$ outputs are affected only on each negative edge of the clock signal. In other words, the *Q* output copies the value of *D* on each negative edge of the clock (the outputs do not change between two negative edges).

Therefore, this circuit is a **'negative-edge-triggered D flip-flop'**. Here are its symbol and its truth table:



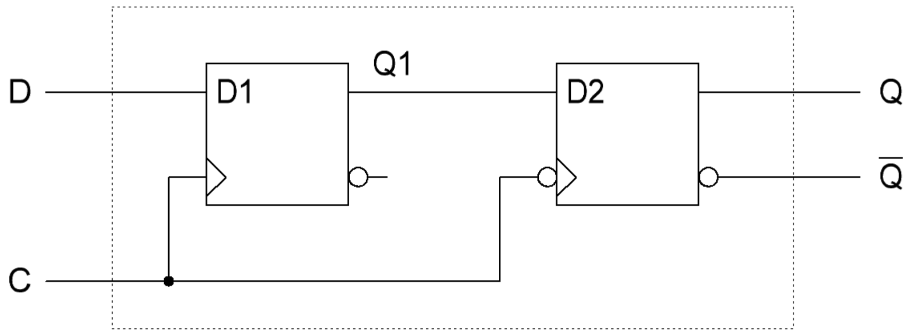
C	D	Q	$\overline{Q}$
↓	0	0	1
↓	1	1	0

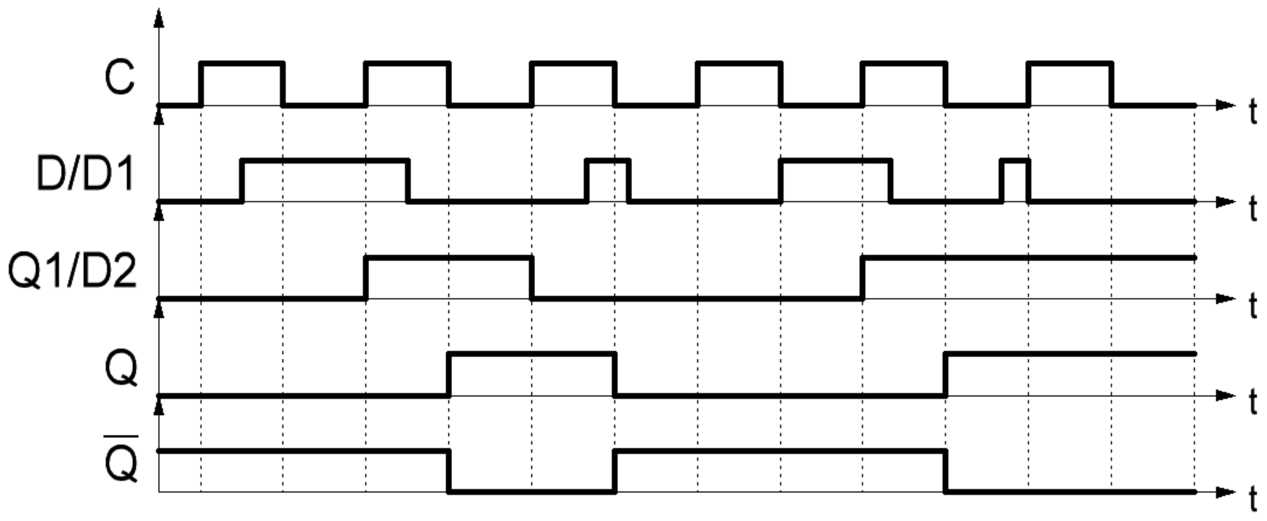
Finally, here is an example of a timing diagram for the negative-edge-triggered D latch:



4. The Master-Slave D Flip-Flop

Let us consider the following circuit and its timing diagram:

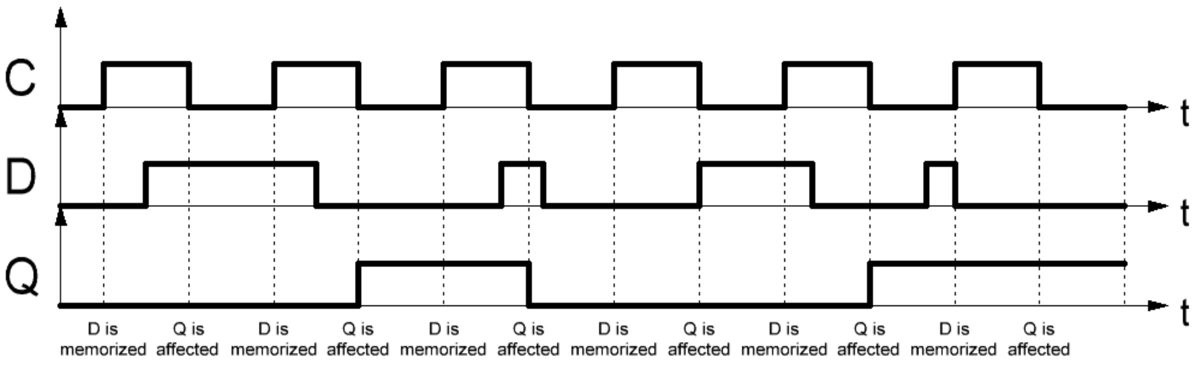




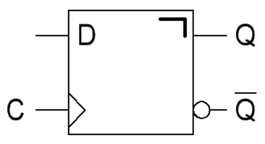
- The first flip-flop is the master flip-flop. Its output (*Q1*/*D2*) is affected on each positive edge of the clock signal according to the present values of *D*. But at this stage, the *Q* output has not been affected yet.

- The second flip-flop is the slave flip-flop. Its outputs (*Q* and $\overline{Q}$) are affected on each negative edge of the clock signal according to the present values of *Q1*/*D2*; that is to say, according to the values of *D* at the previous positive edge.

Let us focus on the *C*, *D* inputs and the *Q* output:



This circuit is a **‘positive-edge-triggered master-slave D flip-flop’** or a **‘master-slave D flip-flop’**. Here are its symbol and its truth table:

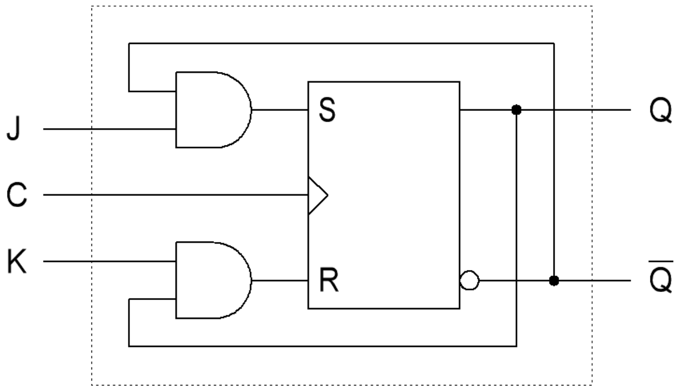


C	D	Q	$\overline{Q}$
⎍	0	0	1
⎍	1	1	0

V. The JK Flip-Flop

1. The Positive-Edge-Triggered JK Flip-Flop

Let us consider the following circuit:



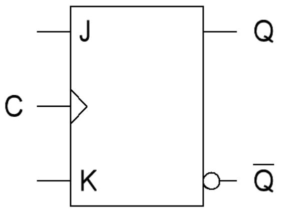
We can write down that:

- $S = J.\overline{Q}$
- $R = K.Q$

Assuming that *q* is the value of *Q* just before the positive edge of *C*, let us draw the following truth table:

C	J	K	q	R	S	Q	
↑	0	0	0	0	0	0	Q = q
↑	0	0	1	0	0	1	
↑	0	1	0	0	0	0	Q = 0
↑	0	1	1	1	0	0	
↑	1	0	0	0	1	1	Q = 1
↑	1	0	1	0	0	1	
↑	1	1	0	0	1	1	$Q = \overline{q}$
↑	1	1	1	1	0	0	

This circuit is a **'positive-edge-triggered JK flip-flop'**. Here are its symbol and its truth table:

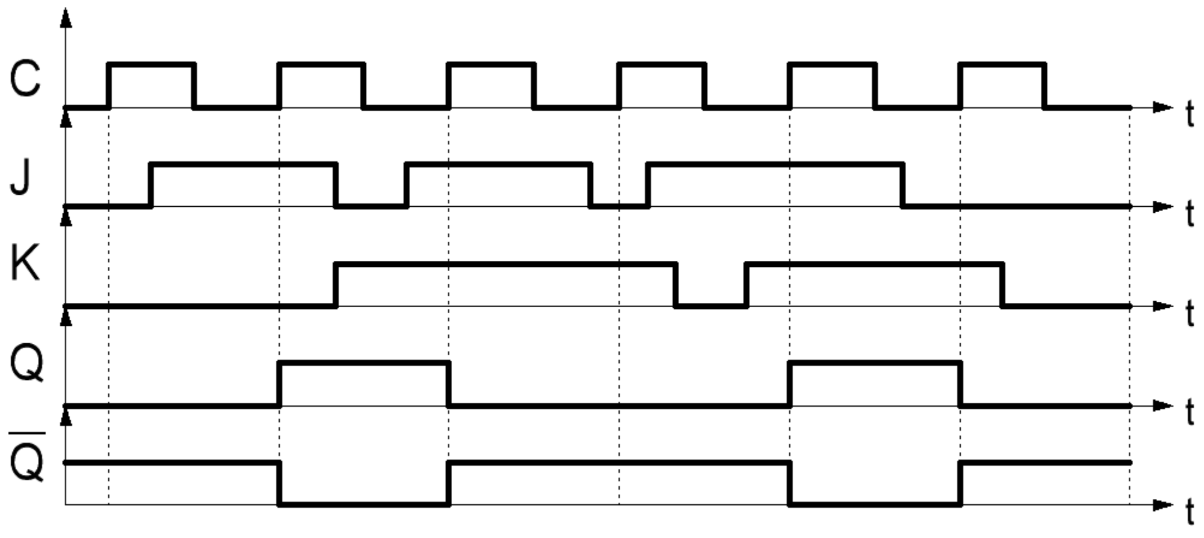


C	J	K	Q	$\overline{Q}$	
↑	0	0	q	$\overline{q}$	← No change
↑	0	1	0	1	← Reset
↑	1	0	1	0	← Set
↑	1	1	$\overline{q}$	q	← Toggle

q = value of *Q* just before the positive edge of *C*.

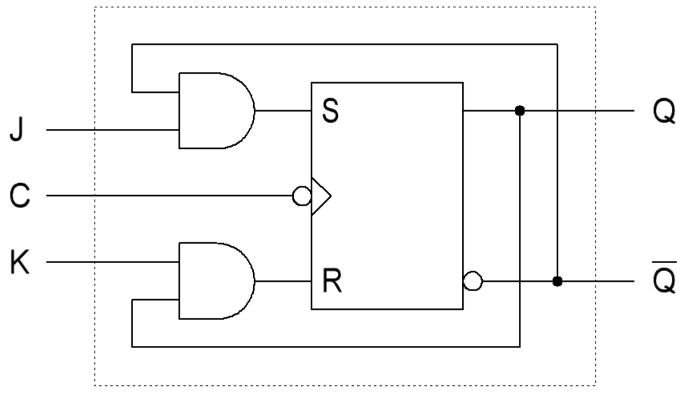
The advantage of a JK flip-flop over an RS flip-flop is that the former has a 'toggle state' instead of a 'forbidden state'. When *J* and *K* are 1s, the *Q* output toggles; in other words, *Q* changes state each time the flip-flop is clocked.

Finally, here is an example of a timing diagram for the positive-edge-triggered JK flip-flop:

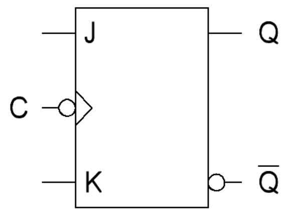


2. The Negative-Edge-Triggered JK Flip-Flop

In the same way, we can design a negative-edge-triggered JK flip-flop:



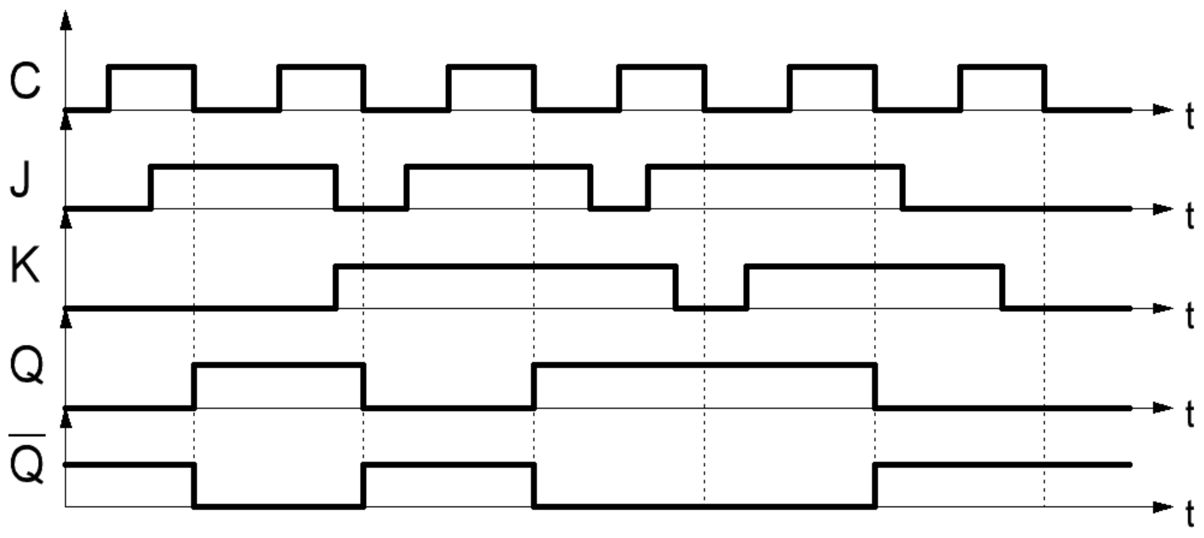
Here are its symbol and its truth table:



C	J	K	Q	$\overline{Q}$	
↓	0	0	q	$\overline{q}$	← No change
↓	0	1	0	1	← Reset
↓	1	0	1	0	← Set
↓	1	1	$\overline{q}$	q	← Toggle

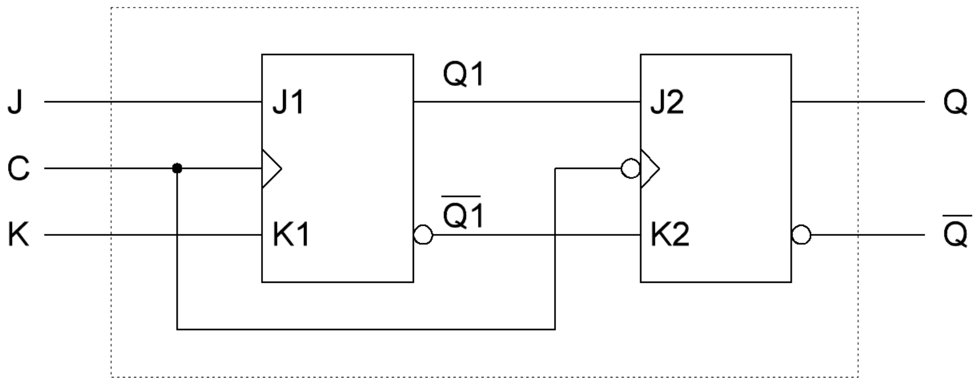
q = value of *Q* just before the negative edge of *C*.

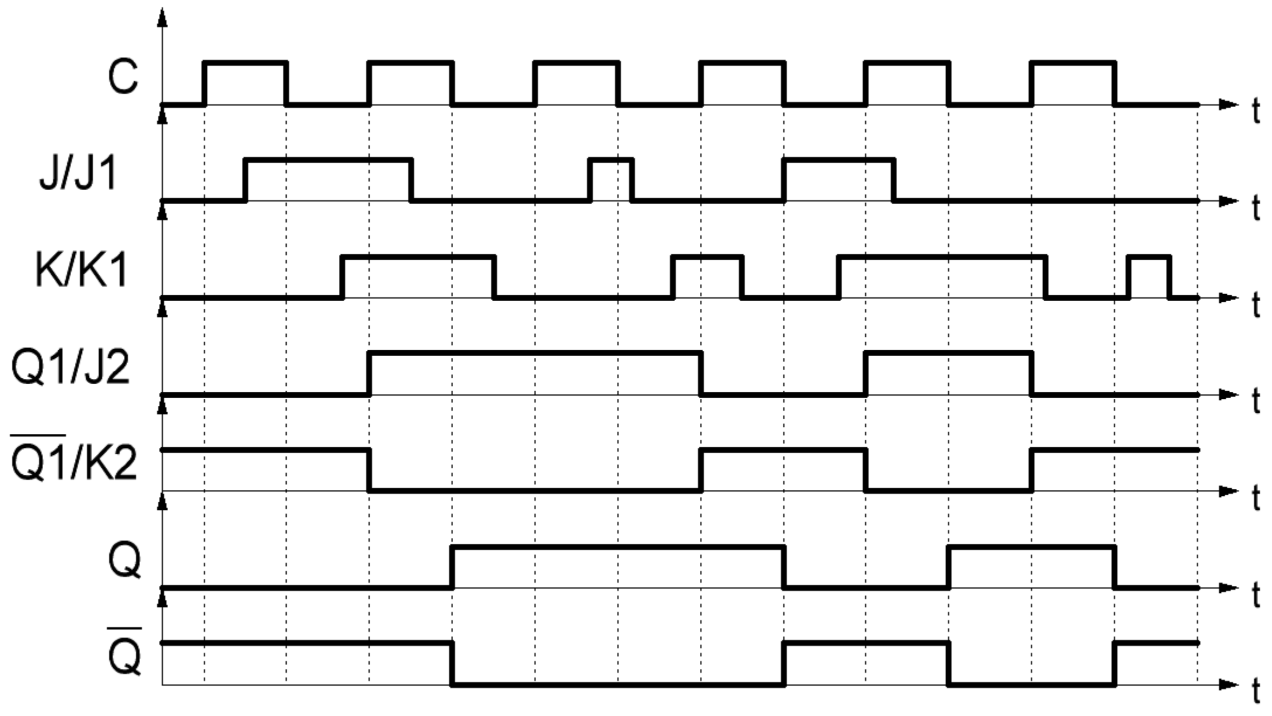
Finally, here is an example of a timing diagram for the negative-edge-triggered JK flip-flop:



3. The Master-Slave JK Flip-Flop

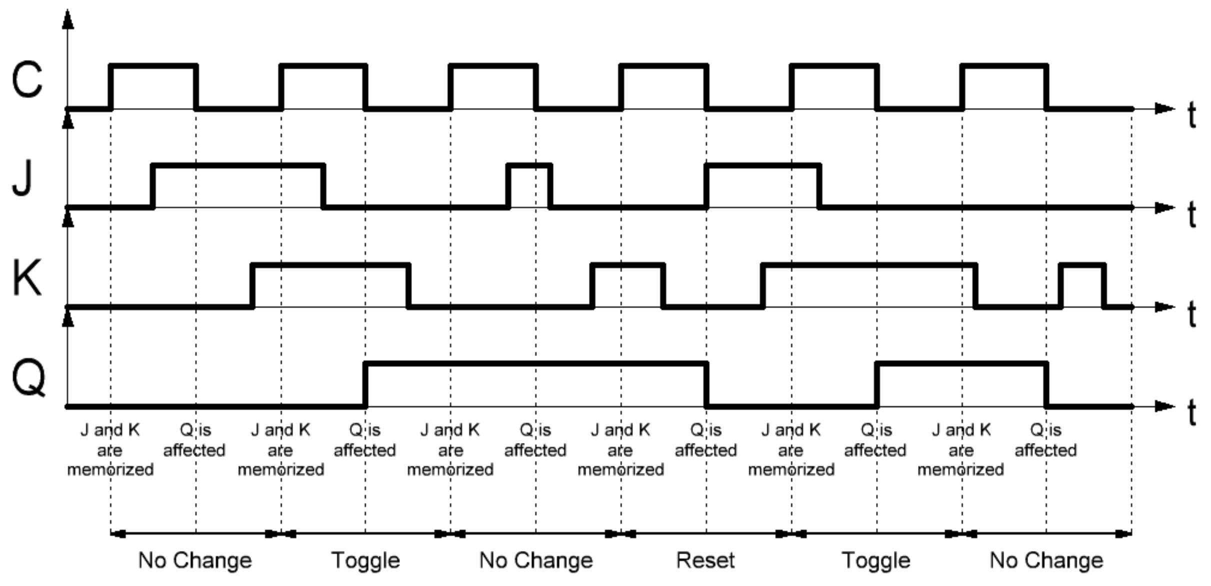
Let us consider the following circuit and its timing diagram:



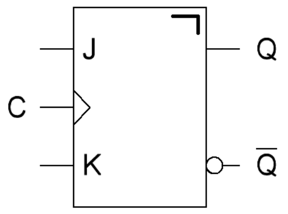


- The first flip-flop is the master flip-flop. Its outputs (*Q1*/*J2* and $\overline{Q1}$/*K2*) are affected on each positive edge of the clock signal according to the present values of *J* and *K*. But at this stage, the *Q* output has not been affected yet.
- The second flip-flop is the slave flip-flop. Its outputs (*Q* and $\overline{Q}$) are affected on each negative edge of the clock signal according to the present values of *Q1*/*J2* and $\overline{Q1}$/*K2*; that is to say, according to the values of *J* and *K* at the previous positive edge.

Let us focus on the *C*, *J*, *K* inputs and the *Q* output:



This circuit is a **‘positive-edge-triggered master-slave JK flip-flop’** or a **‘master-slave JK flip-flop’**. Here are its symbol and its truth table:



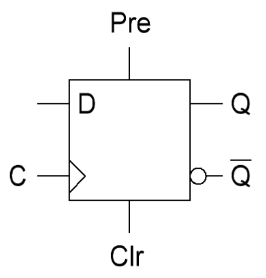
C	**J**	**K**	**Q**	**$\overline{Q}$**	
⎍	0	0	q	$\overline{q}$	← No change
⎍	0	1	0	1	← Reset
⎍	1	0	1	0	← Set
⎍	1	1	$\overline{q}$	q	← Toggle

q = value of *Q* just before the positive edge of *C*.

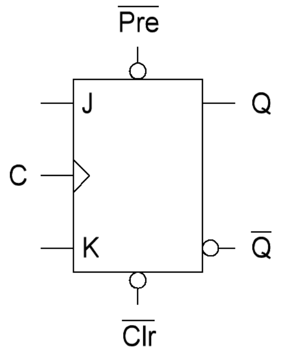
VI. Asynchronous Preset and Clear Inputs

Most D and JK flip-flops are provided with asynchronous preset and clear inputs. These inputs can also be called *set* and *reset* inputs respectively. They can be either active-high or active-low.

Examples:



Active-high preset and clear inputs



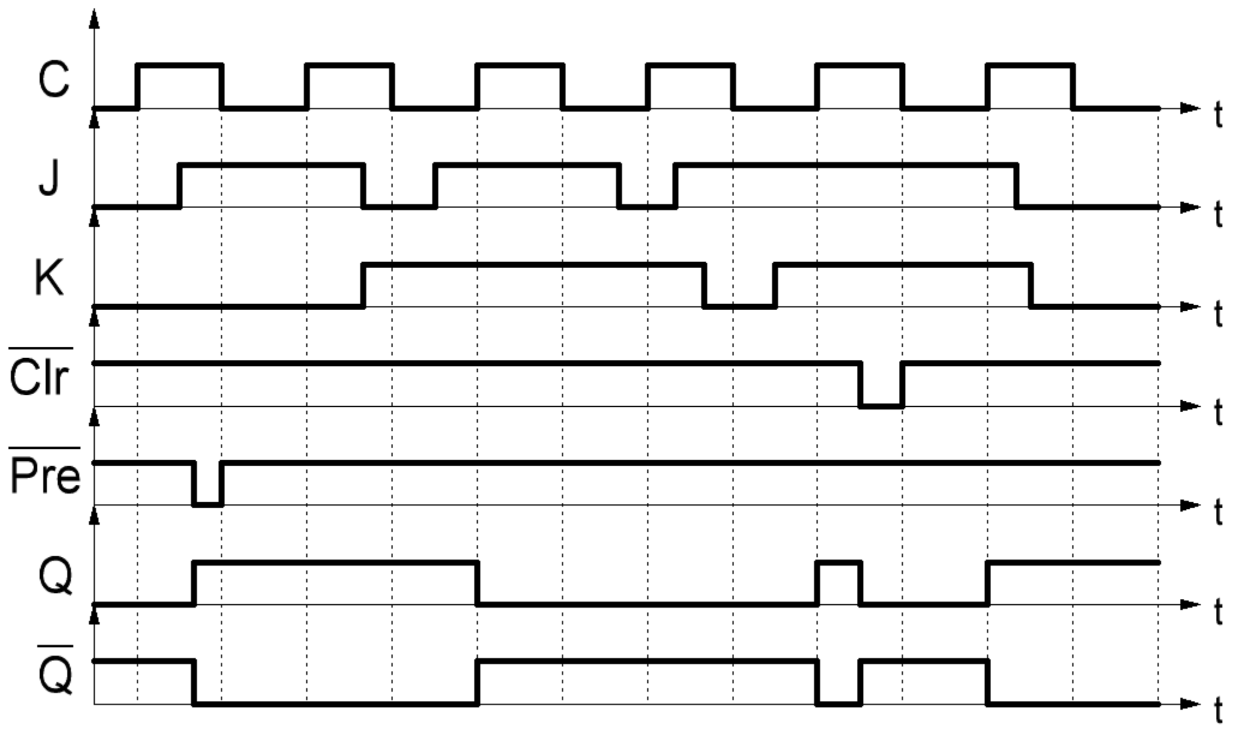
Active-low preset and clear inputs

These inputs are used to set (preset input) or reset (clear input) the *Q* output. These inputs are asynchronous and unconditional, meaning they override all the other inputs of the flip-flop:

- When the preset input is active, the *Q* output is set to one regardless of the other inputs.
- When the clear input is active, the *Q* output is set to zero regardless of the other inputs.

These two inputs should never be active at the same time (it is forbidden).

Here is an example of a timing diagram for a positive-edge-triggered JK flip-flop with active-low preset and clear inputs:


C
J
K
$\overline{Clr}$
$\overline{Pre}$
Q
$\overline{Q}$
t