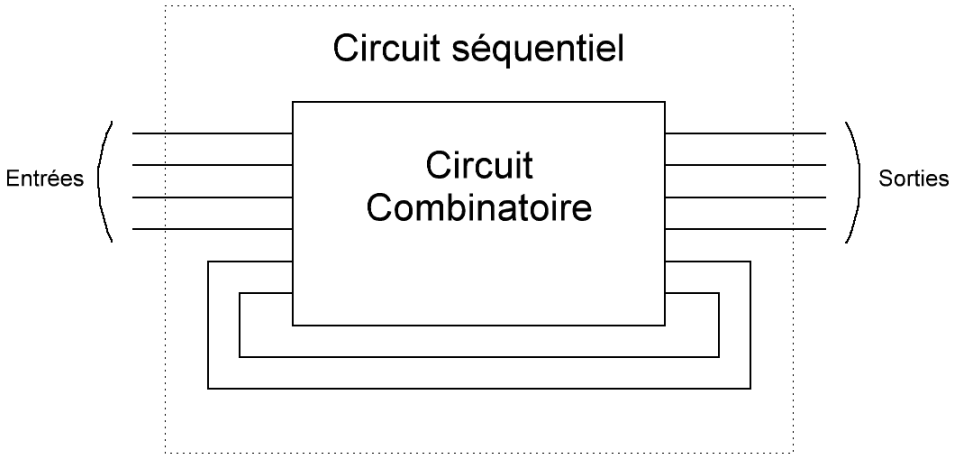
Chapitre 2
Logique séquentielle

Version du 30/01/2018

Table des matières

I. Introduction

La logique séquentielle permet de réaliser des circuits séquentiels, c'est-à-dire des circuits dont les sorties ne sont plus seulement fonction des entrées à un instant donné, mais aussi fonction des états précédents des entrées. Elle se distingue de la logique combinatoire par un bouclage de certaines sorties vers les entrées, ce qui génère des éléments de mémorisation.

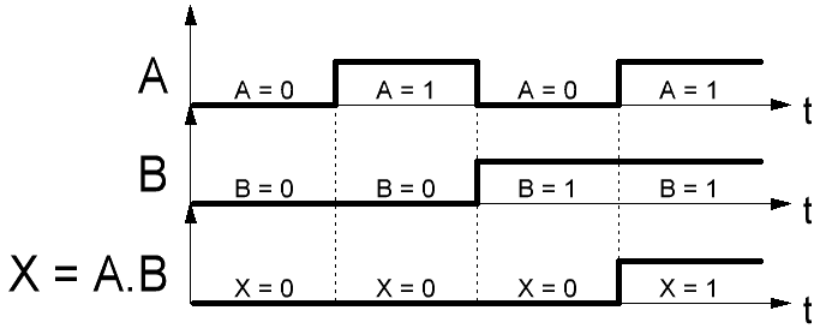


Les briques de base de la logique séquentielle sont les bascules et les verrous.

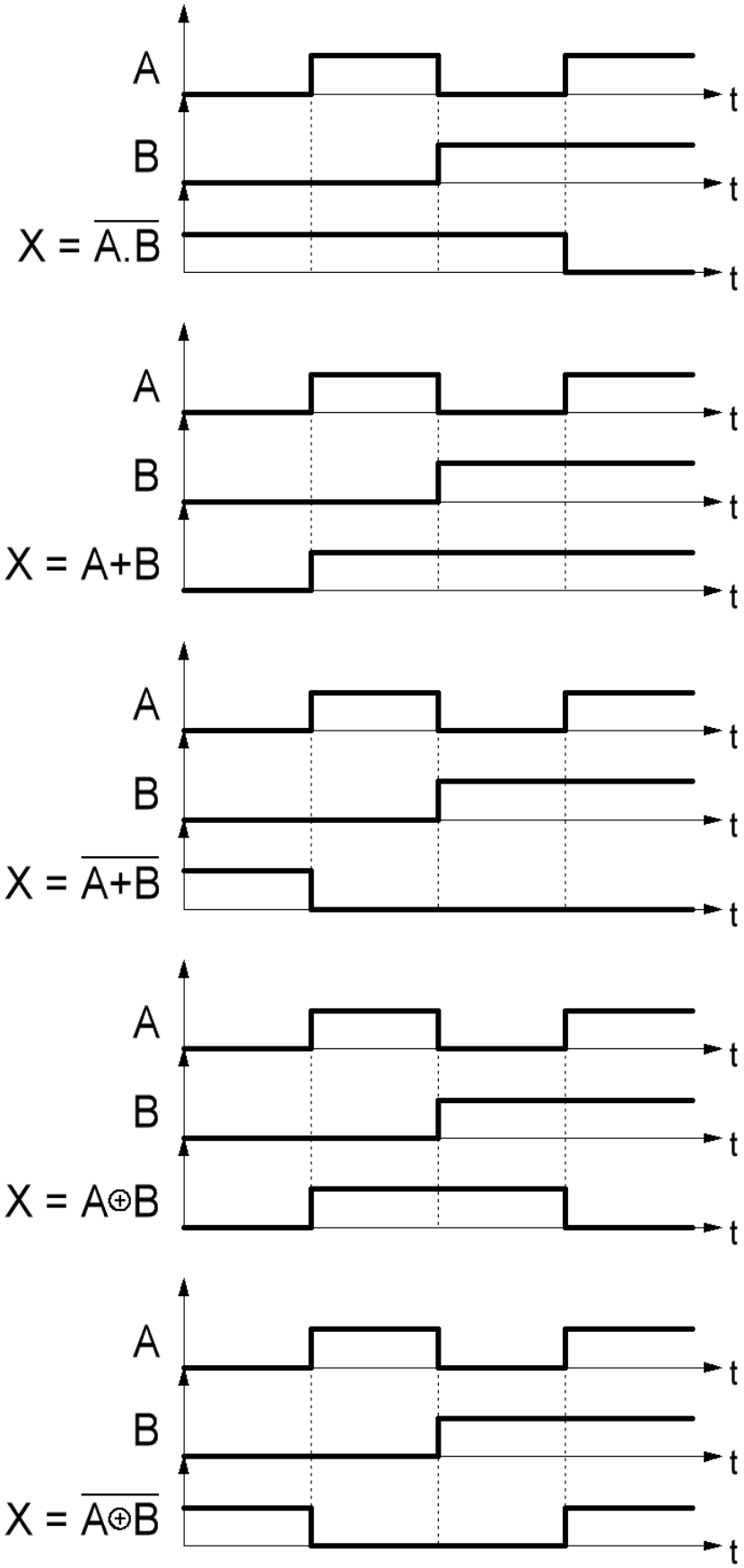
Une bascule peut voir sa sortie modifiée à tout instant ou bien uniquement lors de l'activation d'une entrée spéciale appelée entrée d'horloge. Dans le premier cas, nous dirons que la bascule est asynchrone, dans le second cas, nous dirons qu'elle est synchrone. En d'autres termes, une bascule asynchrone ne possède pas d'entrée d'horloge.

II. Les chronogrammes

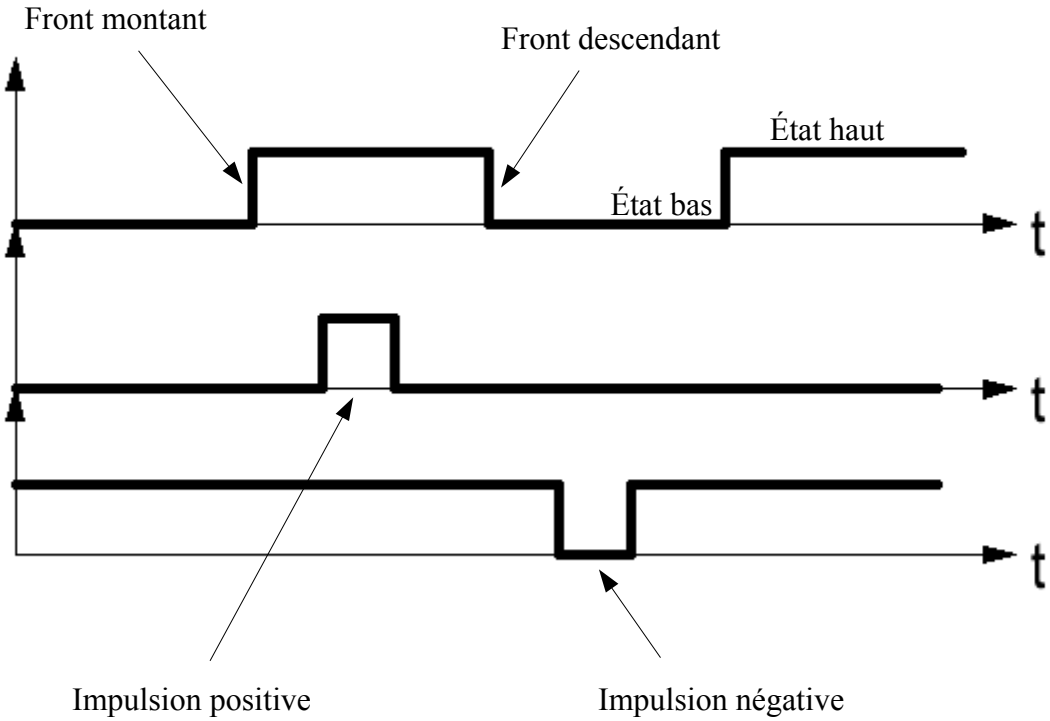
Jusqu'à présent, nous avons vu différentes façons de décrire un circuit combinatoire : expression booléenne, tables de vérité et diagrammes de Karnaugh. Toutefois, ces représentations ne sont plus suffisantes quand une analyse temporelle devient nécessaire. C'est pourquoi nous allons maintenant introduire les chronogrammes. Ces derniers permettent de visualiser l'évolution d'un signal dans le temps. Par exemple, voici le chronogramme d'une porte ET :



Voici également les chronogrammes de quelques autres portes :



Terminologie

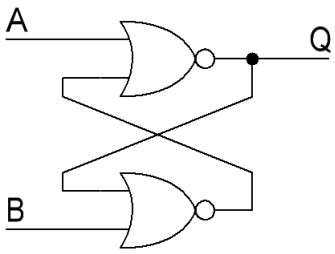


III. La bascule RS

1. Les bascules asynchrones

1.1. La bascule RS asynchrone active à l'état haut (constituée de portes NON-OU)

Soit les deux portes NON-OU suivantes :



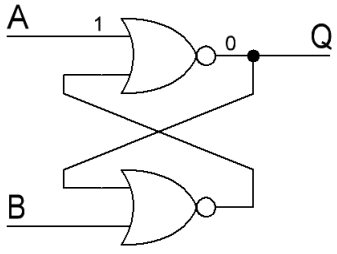
Si l'on considère que *A* et *B* sont les entrées et que *Q* est la sortie, voici la table de vérité de ce montage :

	A	**B**	**Q**	
①	0	0	q	← *q* = valeur de *Q* juste avant le passage à 0 des entrées *A* et *B*.
②	0	1	1	
③	1	0	0	
④	1	1	0	

Les trois dernières lignes s'obtiennent sans difficulté à partir de la table de vérité d'une porte NON-OU :

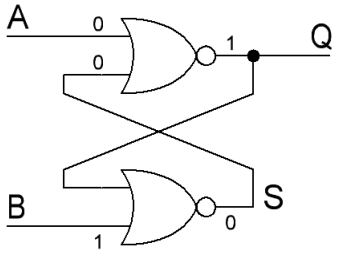
X	**Y**	$\overline{\mathbf{X+Y}}$	
0	0	1	
0	1	0	On peut remarquer que si l'entrée d'une porte NON-OU est à 1, alors sa sortie est à 0 quelle que soit la valeur présente sur son autre entrée.
1	0	0	
1	1	0	

Explication pour les lignes ③ et ④ (A = 1) :



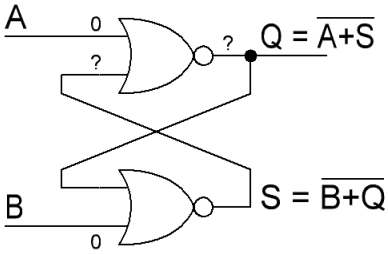
Lorsque l'entrée *A* du montage vaut 1, alors sa sortie *Q* vaut 0. Il n'est pas utile de connaître la valeur présente sur l'autre entrée de la porte NON-OU.

Explication pour la ligne ② (A = 0, B = 1):



On note *S* la sortie de la seconde porte NON-OU.
Lorsque l'entrée *B* du montage vaut 1, alors la sortie *S* vaut 0. Il n'est pas utile de connaître la valeur présente sur l'autre entrée de la porte NON-OU. On a donc $Q = \overline{0+0} = 1$.

Explication pour la ligne ① (A = 0, B = 0):



Ce cas est légèrement plus difficile. *Q* doit être connu pour trouver *S* ($S = \overline{B + Q}$), et *S* doit être connu pour trouver *Q* ($Q = \overline{A + S}$). Autrement dit, pour trouver *Q*, il faut connaître *Q*.

Nous avons donc : $Q = \overline{A + \overline{B+Q}} = \overline{A}.(B+Q)$

Pour être plus précis, il faudrait formuler les choses ainsi : pour trouver **la nouvelle valeur de *Q***, il faut connaître **la valeur précédente de *Q***. Pour éviter toute confusion, nous allons donc appeler *q* la valeur précédente de *Q* ; c'est-à-dire la valeur de *Q* juste avant le passage à 0 des entrées *A* et *B*.

Par conséquent, nous pouvons écrire que : $\boldsymbol{Q = \overline{A}.(B + q)}$

Nous savons également que dans ce cas $A = B = 0$.

Ce qui donne :
$Q = \overline{0}.(0 + q)$
$Q = 1.(q)$
$\boldsymbol{Q = q}$

La nouvelle valeur de *Q* est identique à son ancienne valeur. Nous pouvons conclure que la sortie *Q* ne change pas. Nous dirons qu'elle a été mémorisée.

Cet état est appelé **l'état mémoire**.

Observons maintenant plus en détail la table de vérité de ce circuit.

A	B	Q	
0	0	q	← *q* = valeur de *Q* juste avant le passage à 0 des entrées *A* et *B*.
0	1	1	
1	0	0	
1	1	0	

En supposant que les entrées *A* et *B* sont actives à l'état haut :
- Quand *A* est actif et *B* inactif, la sortie est à 0 : *A* agit comme une entrée de mise à 0 (*reset*).
- Quand *A* est inactif et *B* actif, la sortie est à 1 : *B* agit comme une entrée de mise à 1 (*set*).
- Quand *A* et *B* sont inactifs, la sortie ne change pas (état mémoire).
- Quand *A* et *B* sont actifs, la sortie est à 0 (la mise à 0 est prioritaire).

Par conséquent, les entrées *A* et *B* peuvent être renommées respectivement *R* et *S* :
- *R* pour « entrée ***reset*** active à l'état haut ».
- *S* pour « entrée ***set*** active à l'état haut ».

Ce circuit est appelé une **bascule RS asynchrone active à l'état haut** ou plus simplement une **bascule RS asynchrone**. On retrouve également le terme de **verrou RS**.

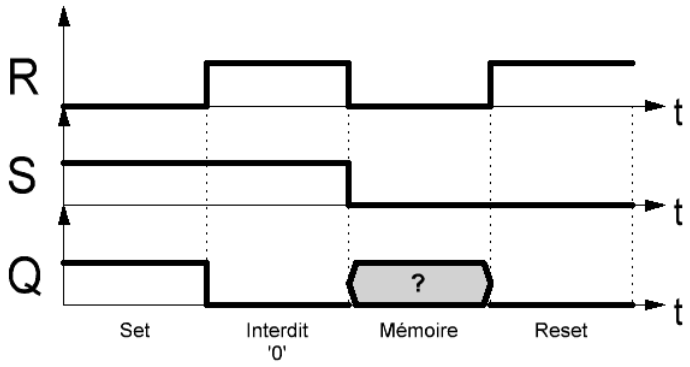
Il est important de préciser que même si l'entrée *reset* est prioritaire, il est fortement recommandé d'éviter cet état. Il n'est pas cohérent d'activer une mise à 1 (*S = 1*) et une mise à 0 (*R = 1*) en même temps.

Nous obtenons donc une nouvelle version de cette table de vérité :

R	S	Q	
0	0	q	← État mémoire
0	1	1	← Mise à 1 (*set*)
1	0	0	← Mise à 0 (*reset*)
1	1	x	← État interdit

Attention, dans cette table de vérité, le caractère *x* ne signifie pas indéfini ou indéterminé. La valeur de *Q* est définie et vaut 0. Peu importe cette valeur, cet état est interdit pour les raisons précédemment citées.

En fait, il existe une autre raison d'éviter l'état interdit. Par exemple, jetons un coup d'œil aux chronogrammes suivants :



Un état indéterminé apparaît quand la bascule passe de l'état interdit à l'état mémoire, c'est-à-dire quand *R* et *S* passe tous les deux de 1 à 0. L'explication vient du fait que physiquement *R* et *S* ne peuvent pas changer exactement au même moment, et même s'ils le pouvaient, les transistors dont ils sont constitués ne sont pas parfaitement identiques et n'ont pas les mêmes temps de réponse. Ce qui signifie qu'un signal (*R* ou *S*) est toujours pris en compte avant l'autre.

La bascule ne peut pas directement passer de l'état interdit à l'état mémoire.

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

Elle doit d'abord passer soit

par un *reset*

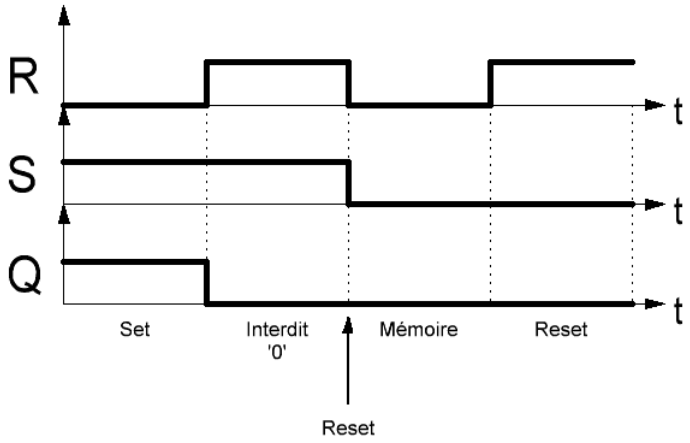
R	**S**	**Q**
0	0	q
0	1	1
1	0	0
1	1	x

ou soit

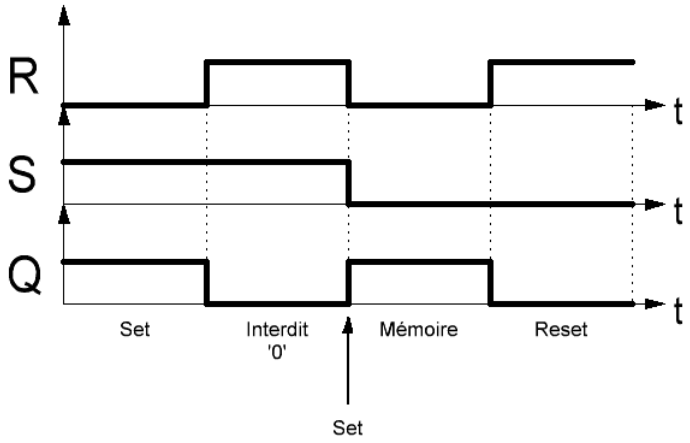
par un *set*.

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

Dans le premier cas (*reset*), la sortie *Q* est d'abord mise à 0 avant d'atteindre l'état mémoire :

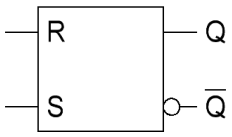


Dans le second cas (*set*), la sortie *Q* est d'abord mise à 1 avant d'atteindre l'état mémoire.



Le problème est que la théorie n'est pas suffisante pour prédire le cas qui se produira (*reset* ou *set*). La seule façon de le savoir est de câbler le circuit et de le tester.

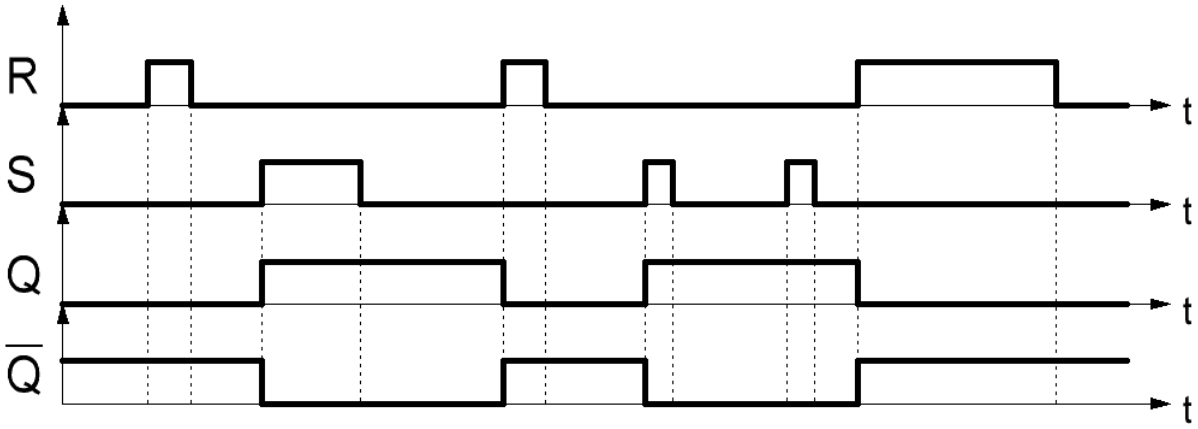
Par souci de simplicité, on utilise habituellement le symbole de la bascule RS asynchrone à la place des deux portes NON-OU. La plupart des bascules RS sont également livrées avec une sortie complémentée. Ci-dessous, le symbole et la table de vérité d'une bascule RS asynchrone :



R	**S**	**Q**	$\overline{\mathbf{Q}}$	
0	0	q	$\overline{q}$	← État mémoire
0	1	1	0	← Mise à 1 (*set*)
1	0	0	1	← Mise à 0 (*reset*)
1	1	x	x	← État interdit

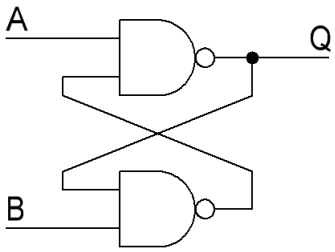
q = *Q* juste avant le passage à 0 des entrées *R* et *S*.

Pour finir, voici un exemple de chronogramme pour la bascule RS asynchrone :



1.2. La bascule RS asynchrone active à l'état bas (constituée de portes NON-ET)

Soit les deux portes NON-ET suivantes :



Si l'on considère que *A* et *B* sont les entrées et que *Q* est la sortie, voici la table de vérité de ce montage :

	A	**B**	**Q**
①	0	0	1
②	0	1	1
③	1	0	0
④	1	1	q

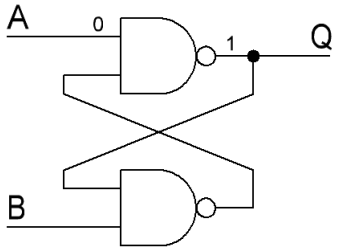
← *q* = valeur de *Q* juste avant le passage à 1 des entrées *A* et *B*.

Les trois premières lignes s'obtiennent sans difficulté à partir de la table de vérité d'une porte NON-ET :

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

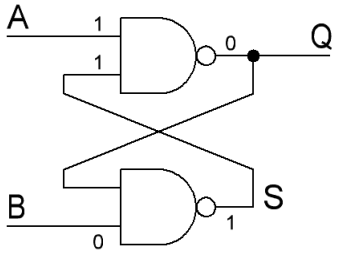
On peut remarquer que si l'entrée d'une porte NON-ET est à 0, alors sa sortie est à 1 quelle que soit la valeur présente sur son autre entrée.

Explication pour les lignes ① et ② (A = 0) :



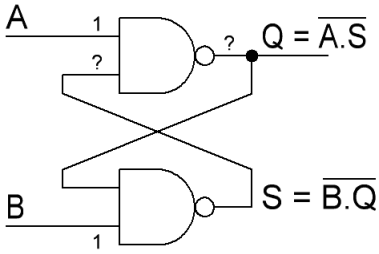
Lorsque l'entrée *A* du montage vaut 0, alors sa sortie *Q* vaut 1. Il n'est pas utile de connaître la valeur présente sur l'autre entrée de la porte NON-ET.

Explication pour la ligne ③ (A = 1, B = 0) :



On note *S* la sortie de la seconde porte NON-ET.
Lorsque l'entrée *B* du montage vaut 0, alors la sortie *S* vaut 1. Il n'est pas utile de connaître la valeur présente sur l'autre entrée de la porte NON-ET. On a donc $Q = \overline{1.1} = 0$

Explication pour la ligne ④ (A = 1, B = 1) :



Ce cas est légèrement plus difficile. *Q* doit être connu pour trouver *S* ($S = \overline{B.Q}$), et *S* doit être connu pour trouver *Q* ($Q = \overline{A.S}$). Autrement dit, pour trouver *Q*, il faut connaître *Q*.

Nous avons donc : $Q = \overline{\overline{A}.\overline{B.Q}} = \overline{A} + B.Q$

Pour être plus précis, il faudrait formuler les choses ainsi : pour trouver **la nouvelle valeur de *Q***, il faut connaître **la valeur précédente de *Q***. Pour éviter toute confusion, nous allons donc appeler *q* la valeur précédente de *Q* ; c'est-à-dire la valeur de *Q* juste avant le passage à 1 des entrées *A* et *B*.

Par conséquent, nous pouvons écrire que : **$Q = \overline{A} + B.q$**

Nous savons également que dans ce cas $A = B = 1$.

Ce qui donne :
$Q = \overline{1} + 1.q$
$Q = 0 + q$
$Q = q$

La nouvelle valeur de Q est identique à son ancienne valeur. Nous pouvons conclure que la sortie Q ne change pas. Nous dirons qu'elle a été mémorisée.

Cet état est appelé **l'état mémoire**.

Observons maintenant plus en détail la table de vérité de ce circuit.

A	B	Q	
0	0	1	
0	1	1	
1	0	0	
1	1	q	← q = valeur de Q juste avant le passage à 1 des entrées A et B.

En supposant que les entrées A et B sont actives à l'état bas :
- Quand A est actif et B inactif, la sortie est à 1 : A agit comme une entrée de mise à 1 (*set*).
- Quand A est inactif et B actif, la sortie est à 0 : B agit comme une entrée de mise à 0 (*reset*).
- Quand A et B sont inactifs, la sortie ne change pas (état mémoire).
- Quand A et B sont actifs, la sortie est à 1 (la mise à 1 est prioritaire).

Par conséquent, les entrées A et B peuvent être renommées respectivement $\overline{R}$ et $\overline{S}$:
- $\overline{R}$ pour « entrée ***reset*** active à l'état bas ».
- $\overline{S}$ pour « entrée ***set*** active à l'état bas ».

Ce circuit est appelé une **bascule RS asynchrone active à l'état bas** ou plus simplement une **bascule $\overline{\text{RS}}$ asynchrone**. On retrouve également le terme de **verrou $\overline{\text{RS}}$**.

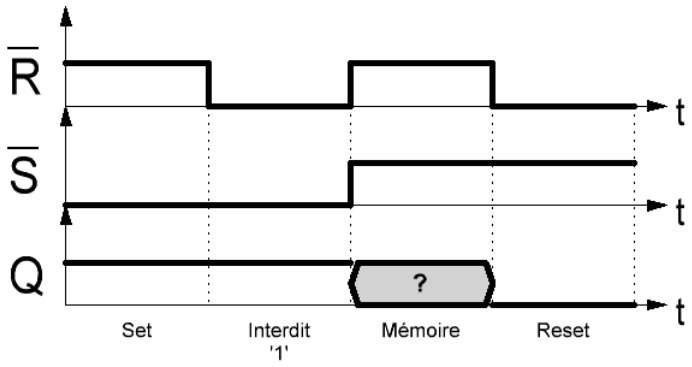
Il est important de préciser que même si l'entrée *reset* est prioritaire, il est fortement recommandé d'éviter cet état. Il n'est pas cohérent d'activer une mise à 1 ($\overline{S} = 0$) et une mise à 0 ($\overline{R} = 0$) en même temps.

Nous obtenons donc une nouvelle version de cette table de vérité :

(B) $\overline{\text{R}}$	(A) $\overline{\text{S}}$	Q	
0	0	x	← État interdit
0	1	0	← Mise à 0 (*reset*)
1	0	1	← Mise à 1 (*set*)
1	1	q	← État mémoire

Attention, dans cette table de vérité, le caractère *x* ne signifie pas indéfini ou indéterminé. La valeur de Q est définie et vaut 1. Peu importe cette valeur, cet état est interdit pour les raisons précédemment citées.

En fait, il existe une autre raison d'éviter l'état interdit. Par exemple, jetons un coup d'œil aux chronogrammes suivants :



Un état indéterminé apparaît quand la bascule passe de l'état interdit à l'état mémoire, c'est-à-dire quand $\overline{R}$ et $\overline{S}$ passe tous les deux de 0 à 1. L'explication vient du fait que physiquement $\overline{R}$ et $\overline{S}$ ne peuvent pas changer exactement au même moment, et même s'ils le pouvaient, les transistors dont ils sont constitués ne sont pas parfaitement identiques et n'ont pas les mêmes temps de réponse. Ce qui signifie qu'un signal ($\overline{R}$ ou $\overline{S}$) est toujours pris en compte avant l'autre.

La bascule ne peut pas directement passer de l'état interdit à l'état mémoire.

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

Elle doit d'abord passer soit

par un *reset*

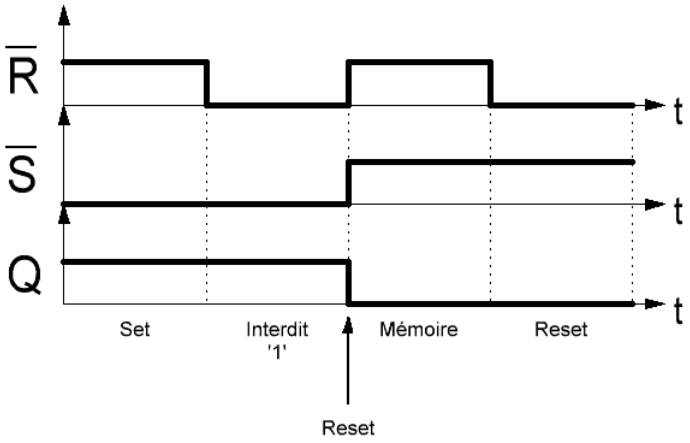
$\overline{\mathbf{R}}$	$\overline{\mathbf{S}}$	**Q**
0	0	x
0	1	0
1	0	1
1	1	q

ou soit

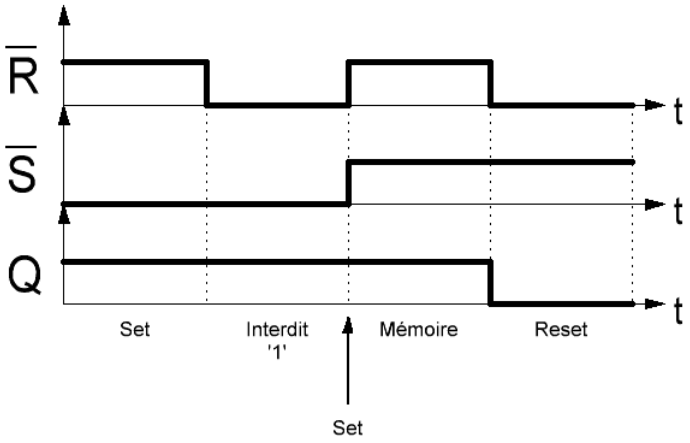
par un *set*.

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

Dans le premier cas (*reset*), la sortie *Q* est d'abord mise à 0 avant d'atteindre l'état mémoire :

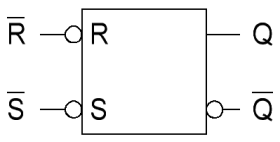


Dans le second cas (*set*), la sortie *Q* est d'abord mise à 1 avant d'atteindre l'état mémoire.



Le problème est que la théorie n'est pas suffisante pour prédire le cas qui se produira (*reset* ou *set*). La seule façon de le savoir est de câbler le circuit et de le tester.

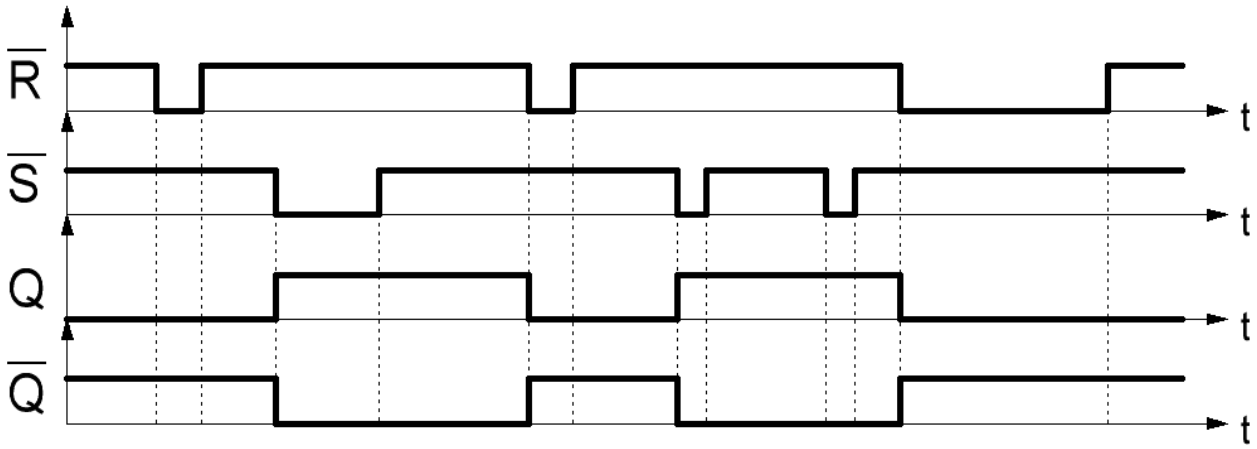
Par souci de simplicité, on utilise habituellement le symbole de la bascule $\overline{RS}$ asynchrone à la place des deux portes NON-ET. La plupart des bascules $\overline{RS}$ sont également livrées avec une sortie complémentée. Ci-dessous, le symbole et la table de vérité d'une bascule $\overline{RS}$ asynchrone :



$\overline{R}$	$\overline{S}$	Q	$\overline{Q}$	
0	0	x	x	← État interdit
0	1	0	1	← Mise à 0 (*reset*)
1	0	1	0	← Mise à 1 (*set*)
1	1	q	$\overline{q}$	← État mémoire

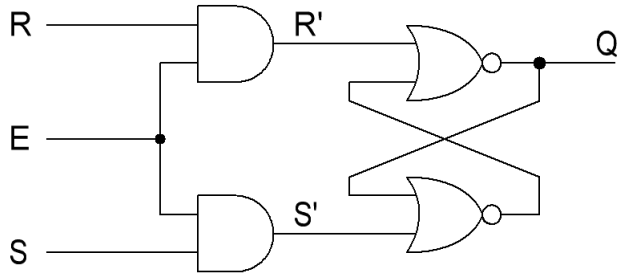
q = *Q* juste avant le passage à 1 des entrées *R* et *S*.

Pour finir, voici un exemple de chronogramme pour la bascule RS asynchrone active à l'état bas :



2. La bascule RS synchronisée sur état

Soit le circuit suivant :



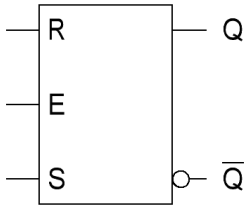
R' et *S'* sont respectivement les entrées *reset* et *set* d'une bascule RS asynchrone (constituée de deux portes NON-OU).

E (*Enable*) est une entrée d'activation sur état active à l'état haut.

- Quand $E = 1$, alors $R' = R$ et $S' = S$ → Le circuit se comporte comme une bascule RS asynchrone.
- Quand $E = 0$, alors $R' = 0$ et $S' = 0$ → Le circuit est dans l'état mémoire.

E	R'	S'	Comportement du circuit
0	0	0	État mémoire
1	R	S	Bascule RS asynchrone

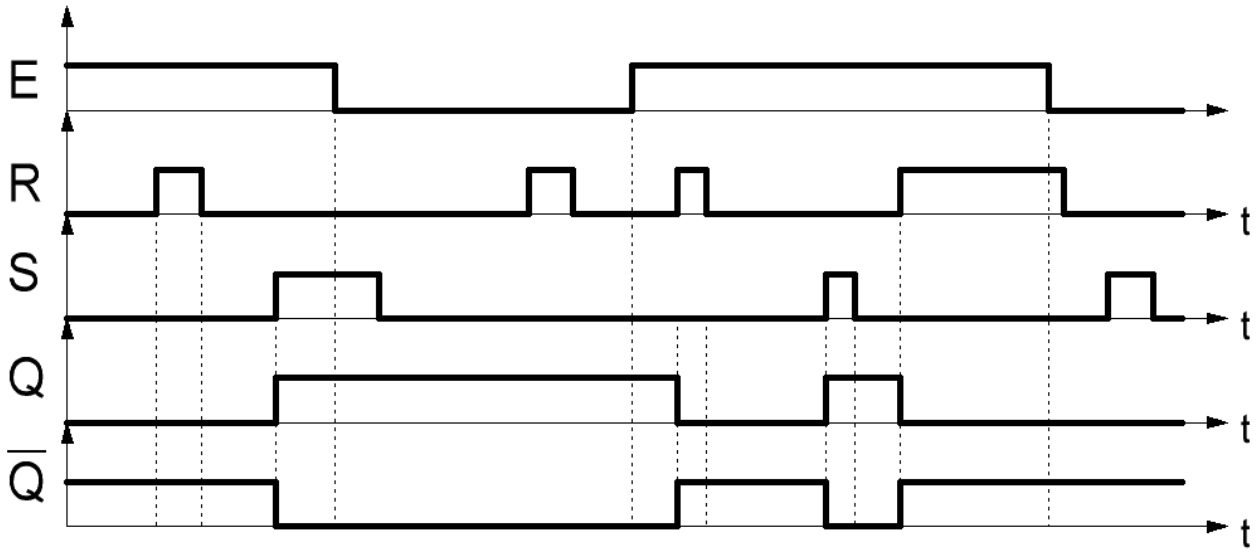
Ce circuit est une **bascule RS synchronisée sur état (haut)** ou un **verrou RS synchrone**. Sa table de vérité est la suivante :



E	R	S	Q	$\overline{Q}$	
0	Φ	Φ	q	$\overline{q}$	← État mémoire
1	0	0	q	$\overline{q}$	← État mémoire
1	0	1	1	0	← Mise à 1 (*set*)
1	1	0	0	1	← Mise à 0 (*reset*)
1	1	1	x	x	← État interdit

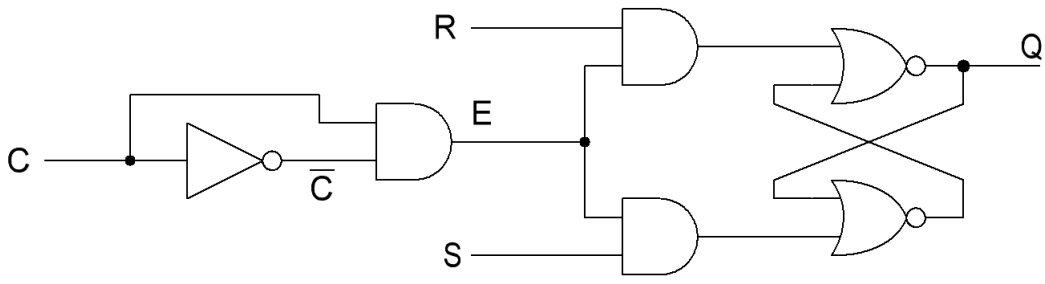
q = valeur de *Q* juste avant le passage dans l'état mémoire.
Le caractère « Φ » signifie 0 ou 1.
Le caractère « x » signifie interdit.

Voici un exemple de chronogramme pour la bascule RS synchronisée sur état :

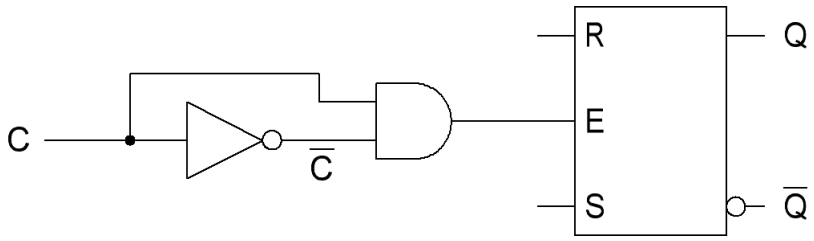


3. La bascule RS synchronisée sur front montant

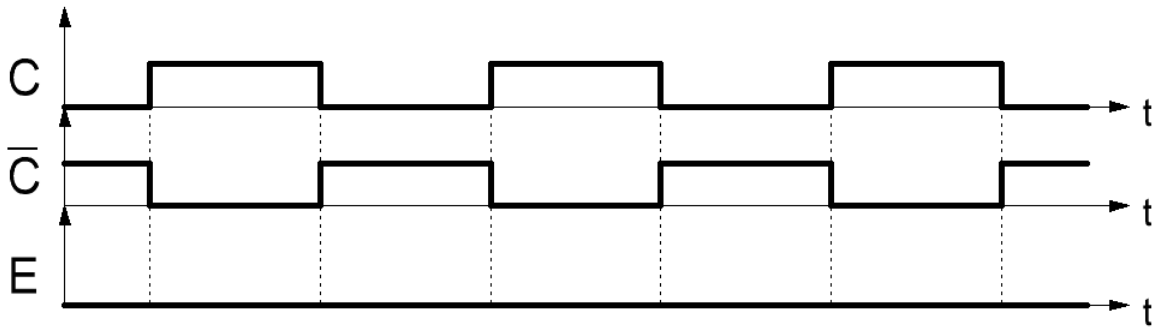
Soit le circuit suivant :



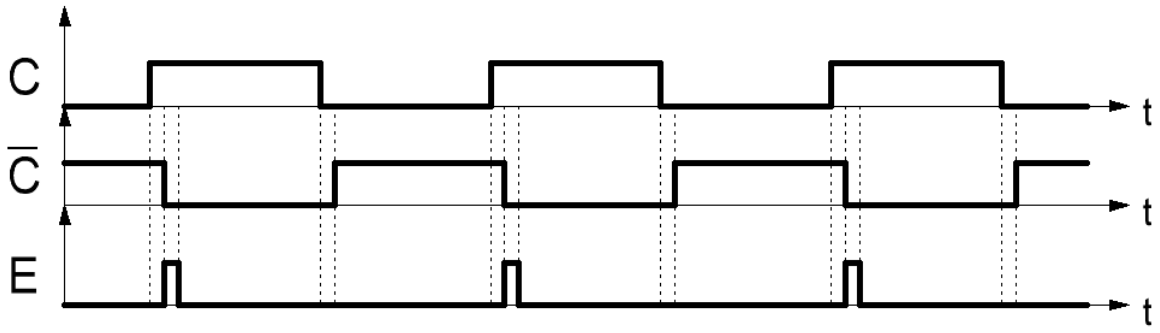
On peut reconnaître à droite une bascule RS synchronisée sur état :



En supposant que les portes soient parfaites, dessinons les chronogrammes de **C**, $\overline{\mathbf{C}}$ et **E** :



Il semble que *E* soit toujours à 0, ce qui signifierait que la bascule est toujours dans l'état mémoire. En fait, ce n'est pas aussi simple que ça. En pratique, chaque porte possède un temps de réponse. Il s'agit du temps entre la stimulation et la réponse de la porte.

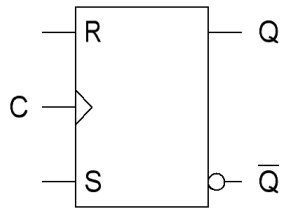


La porte NON ne change pas immédiatement d'état après la transition de son entrée. Un petit temps de réponse (de l'ordre de la nanoseconde) apparaît entre les transitions de *C* et de $\overline{C}$. Par conséquent, ces deux signaux sont à 1 en même temps pendant un très court laps de temps. La porte ET possède également un temps de réponse. La sortie *E* est donc mise à 1 légèrement plus tard. Pour résumer, la sortie *E* est mise à 1 pendant un très bref instant juste après le front montant de *C*.

Nous pouvons en déduire que la sortie *Q* ne peut changer que sur les fronts montants de *C*.

Ce circuit est une **bascule RS synchronisée sur front montant** et l'entrée *C* est son **entrée d'horloge**.

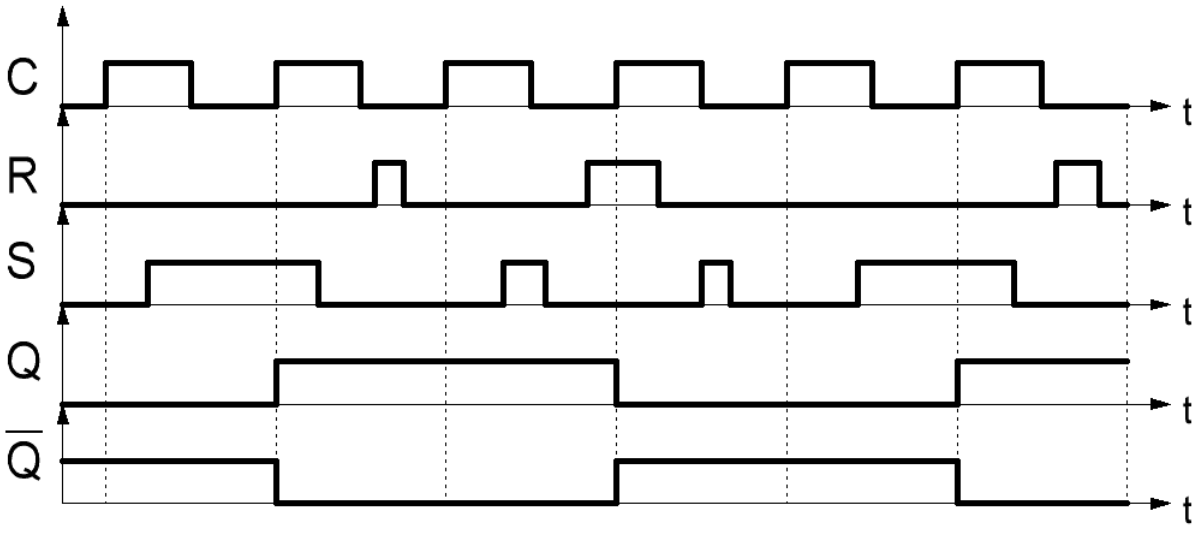
Son symbole et sa table de vérité sont les suivants :



C	R	S	Q	$\overline{Q}$	
↑	0	0	q	$\overline{q}$	← État mémoire
↑	0	1	1	0	← Mise à 1 (*set*)
↑	1	0	0	1	← Mise à 0 (*reset*)
↑	1	1	x	x	← État interdit

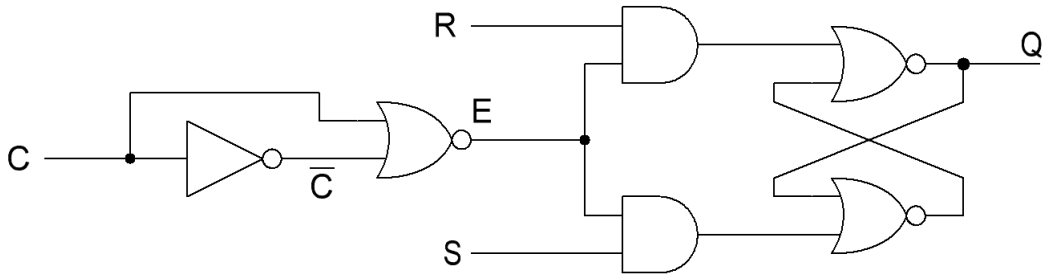
q = valeur de *Q* juste avant le front montant de *C*.

Pour finir, voici un exemple de chronogramme pour la bascule RS synchronisée sur front montant :

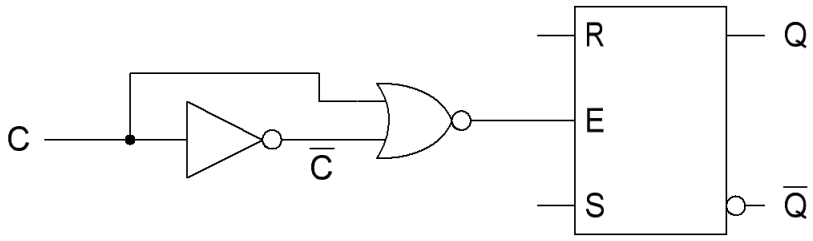


4. La bascule RS synchronisée sur front descendant

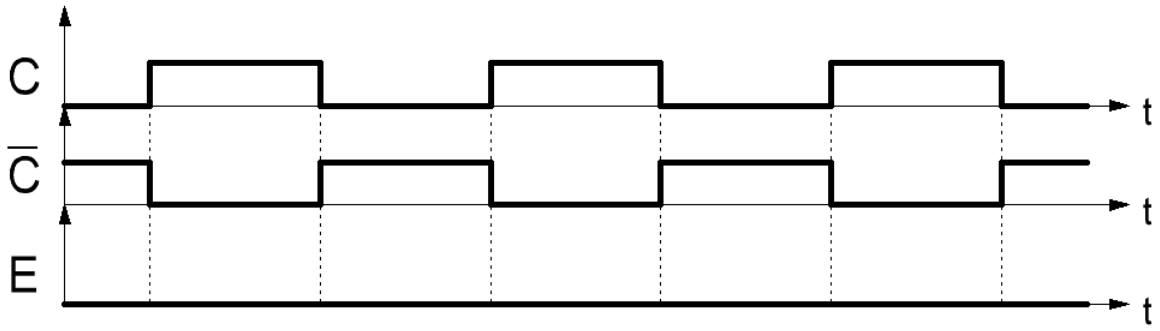
Dans le même esprit, il est possible de concevoir une bascule RS synchronisée sur front descendant :



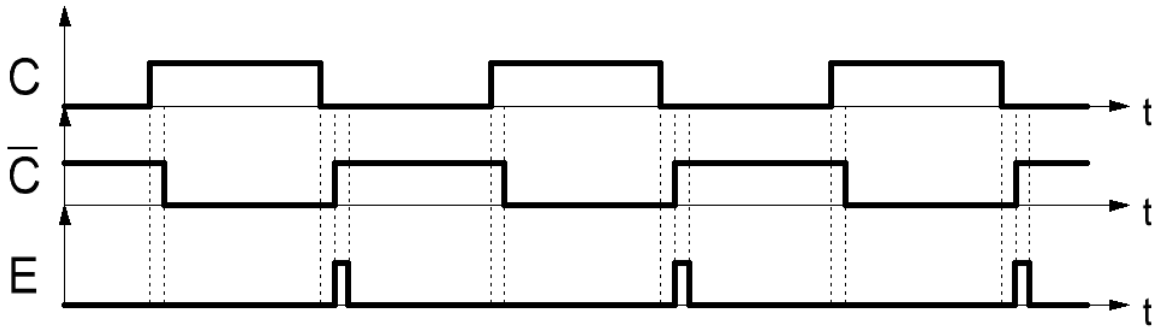
On peut reconnaître à droite une bascule RS synchronisée sur état :



En supposant que les portes soient parfaites, dessinons les chronogrammes de **C**, $\overline{\mathbf{C}}$ et **E** :



Il semble que *E* soit toujours à 0, ce qui signifierait que la bascule est toujours dans l'état mémoire. Mais comme cela a été dit précédemment, chaque porte possède un temps de réponse.

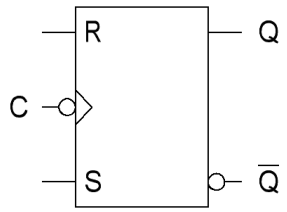


La porte NON ne change pas immédiatement d'état après la transition de son entrée. Un petit temps de réponse (de l'ordre de la nanoseconde) apparaît entre les transitions de *C* et de $\overline{C}$. Par conséquent, ces deux signaux sont à 0 en même temps pendant un très court laps de temps. La porte NON-OU possède également un temps de réponse. La sortie *E* est donc mise à 1 légèrement plus tard. Pour résumer, la sortie *E* est mise à 1 pendant un très bref instant juste après le front descendant de *C*.

Nous pouvons en déduire que la sortie *Q* ne peut changer que sur les fronts descendants de *C*.

Ce circuit est une **bascule RS synchronisée sur front descendant** et l'entrée *C* est son **entrée d'horloge**.

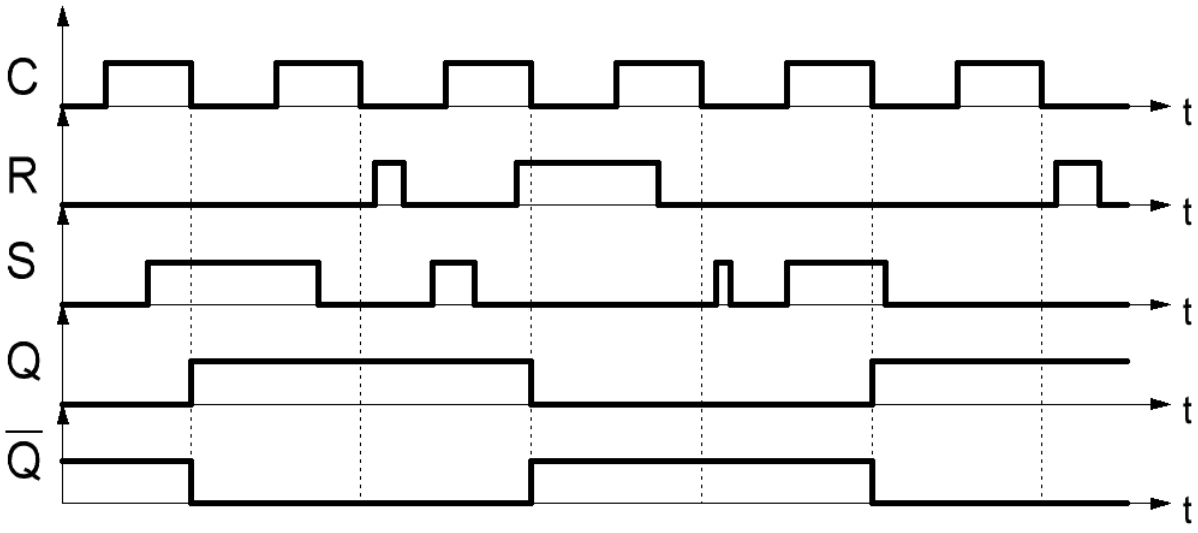
Son symbole et sa table de vérité sont les suivants :



C	R	S	Q	$\overline{Q}$	
↓	0	0	q	$\overline{q}$	← État mémoire
↓	0	1	1	0	← Mise à 1 (*set*)
↓	1	0	0	1	← Mise à 0 (*reset*)
↓	1	1	x	x	← État interdit

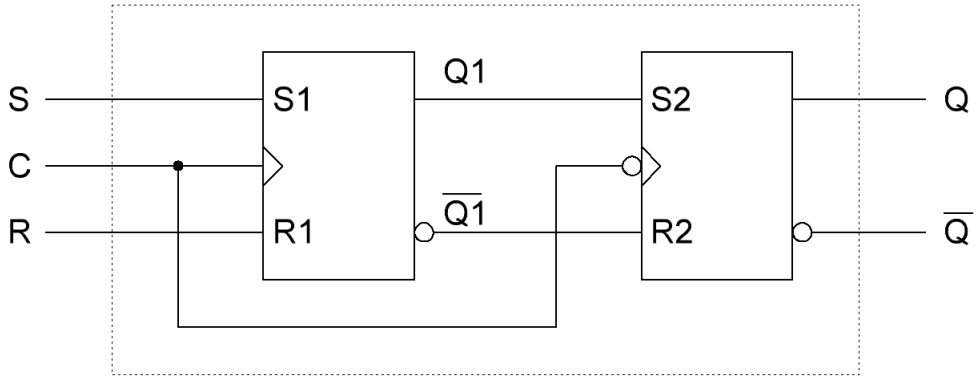
q = valeur de *Q* juste avant le front descendant de *C*.

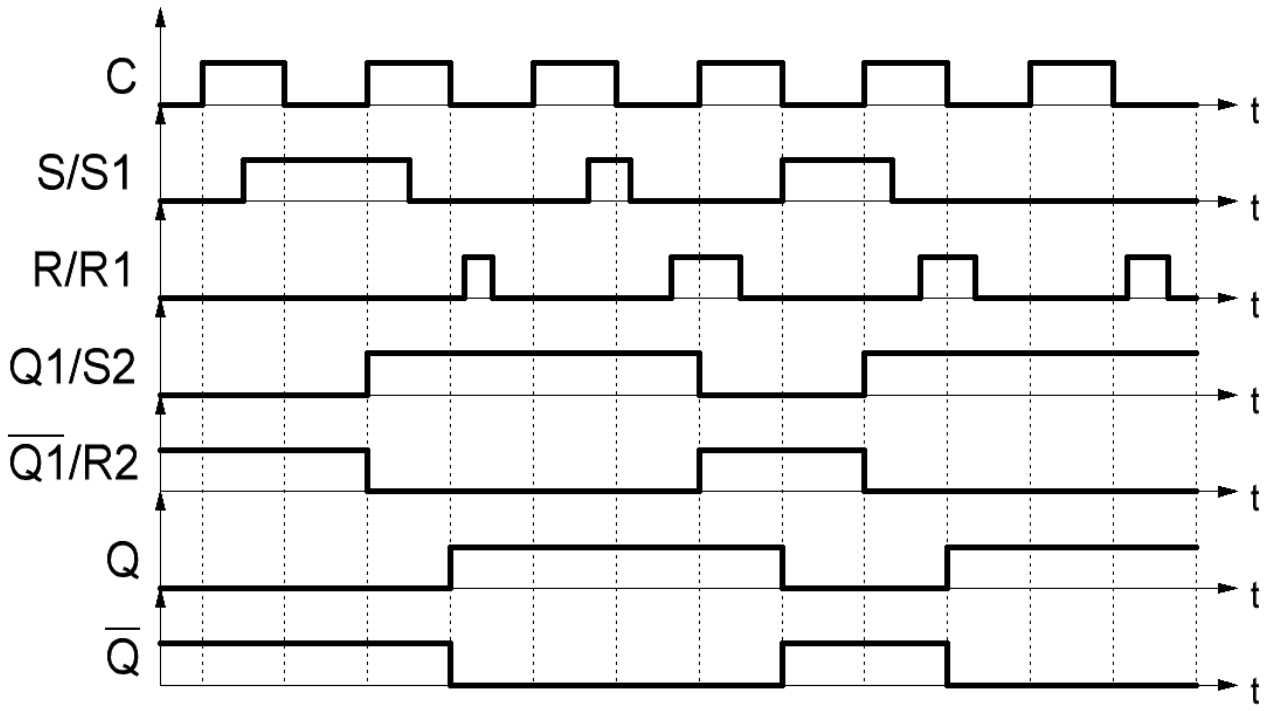
Pour finir, voici un exemple de chronogramme pour la bascule RS synchronisée sur front descendant :



5. La bascule RS synchronisée sur impulsion (bascule RS maître-esclave)

Considérons le circuit ci-dessous et son chronogramme :

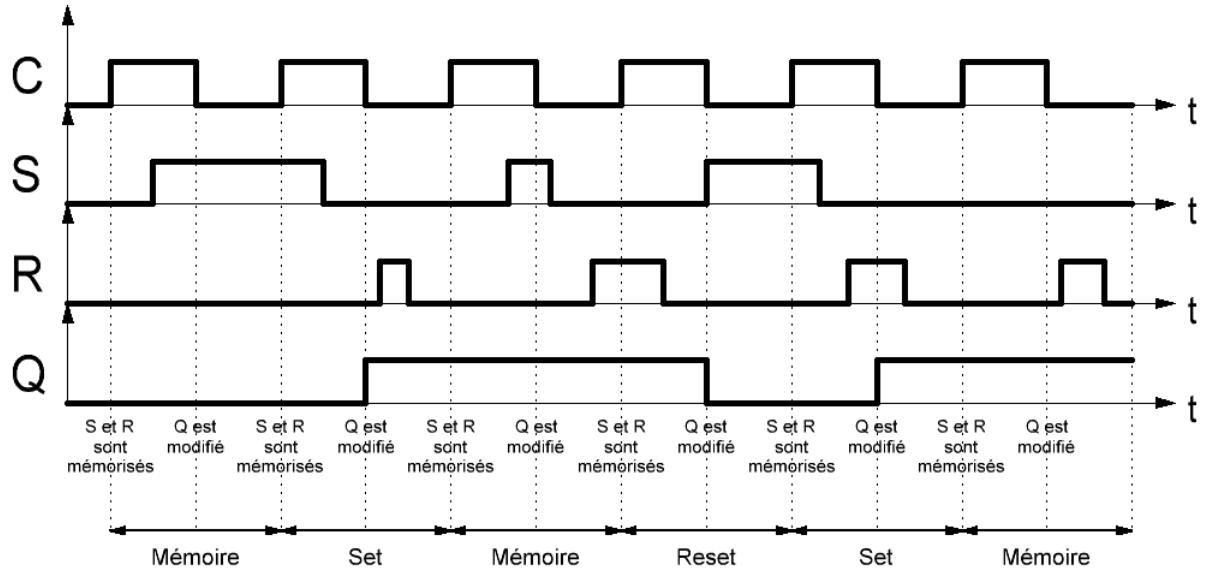




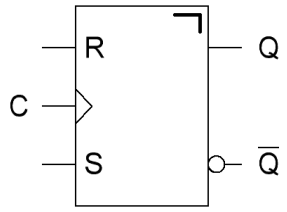
- La première bascule joue le rôle de maître. Ses sorties (*Q1*/*S2* et $\overline{Q1}$/*R2*) sont modifiées à chaque front montant du signal d'horloge en fonction des états de *S* et de *R*. Mais à ce stade, la sortie *Q* n'a pas encore été modifiée.

- La seconde bascule joue le rôle d'esclave. Ses sorties (*Q* et $\overline{Q}$) sont modifiées à chaque front descendant du signal d'horloge en fonction des états de *Q1*/*S2* et de $\overline{Q1}$/*R2* ; c'est-à-dire en fonction des états de *S* et de *R* au moment du précédent front montant.

Concentrons-nous plus particulièrement sur les entrées *C*, *S*, *R* et sur la sortie *Q* :



Ce circuit est une **bascule RS synchronisée sur impulsion** ou encore une **bascule RS maître-esclave**. Son symbole et sa table de vérité sont les suivants :



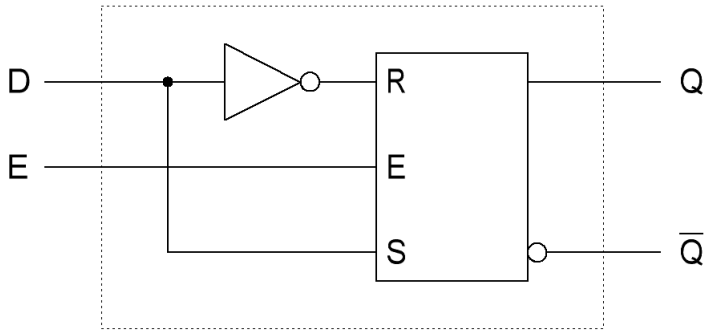
C	R	S	Q	$\overline{Q}$	
⎍	0	0	q	$\overline{q}$	← État mémoire
⎍	0	1	1	0	← Mise à 1 (*set*)
⎍	1	0	0	1	← Mise à 0 (*reset*)
⎍	1	1	x	x	← État interdit

q = valeur de *Q* juste avant le front montant de *C*.

IV. La bascule D

1. La bascule D synchronisée sur état (verrou D)

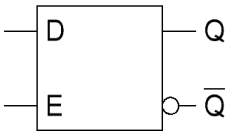
Soit le circuit suivant :



- Quand $E = 0 \rightarrow$ La sortie ne change pas (quelle que soit la valeur de D).
- Quand $E = 1$ et $D = 0 \rightarrow R = 1$ et $S = 0 \rightarrow Q = 0$.
- Quand $E = 1$ et $D = 1 \rightarrow R = 0$ et $S = 1 \rightarrow Q = 1$.

Par conséquent, quand $E = 1$, $Q = D$.

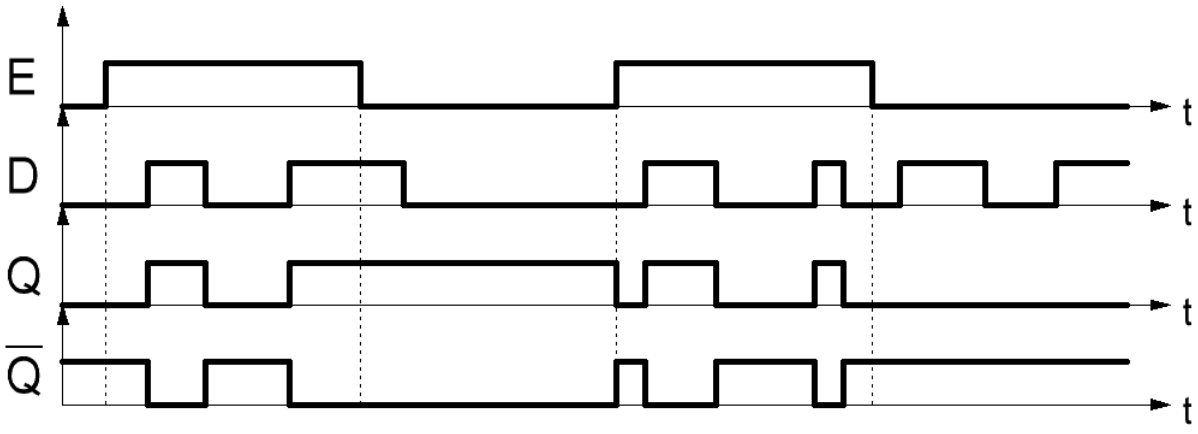
Ce circuit est une **bascule D synchronisée sur état** ou encore un **verrou D** (D signifie *data*). Son symbole et sa table de vérité son les suivants :



E	**D**	**Q**	**$\overline{Q}$**	
0	Φ	q	$\overline{q}$	← État mémoire
1	0	0	1	
1	1	1	0	

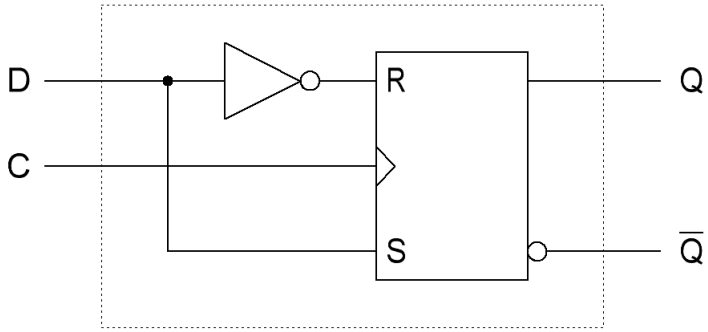
q = valeur de *Q* juste avant le passage dans l'état mémoire.
Le caractère « Φ » signifie 0 ou 1.

Pour finir, voici un exemple de chronogramme pour le verrou D :



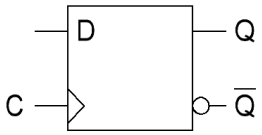
2. La bascule D synchronisée sur front montant

Soit le circuit suivant :



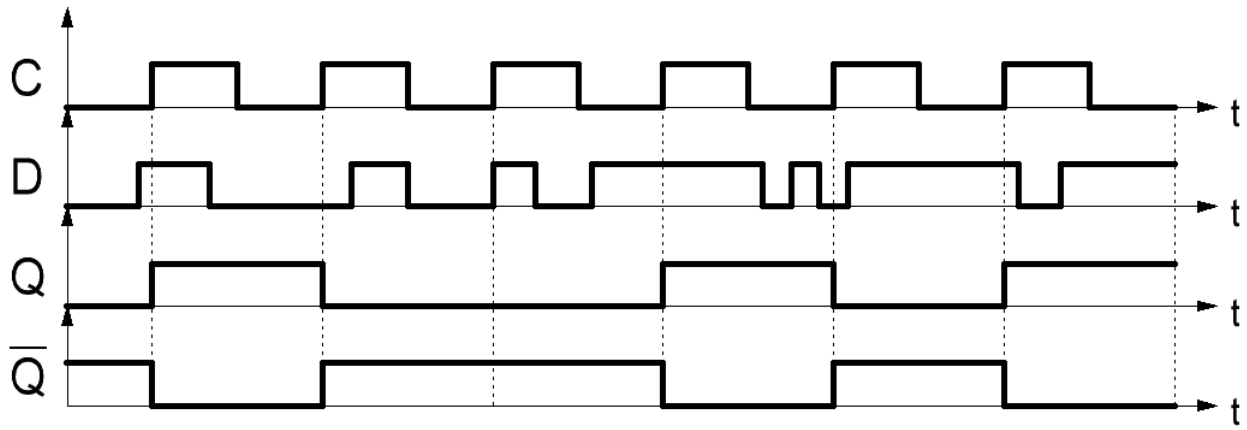
Ce circuit est semblable au verrou D sauf que les sorties Q et $\overline{Q}$ sont modifiées uniquement sur les fronts montant de l'horloge. Autrement dit, la sortie Q copie la valeur de D sur chaque front montant de l'horloge (les sorties ne changent pas entre deux fronts montants).

Ce circuit est une **bascule D synchronisée sur front montant**. Son symbole et sa table de vérité sont les suivants :



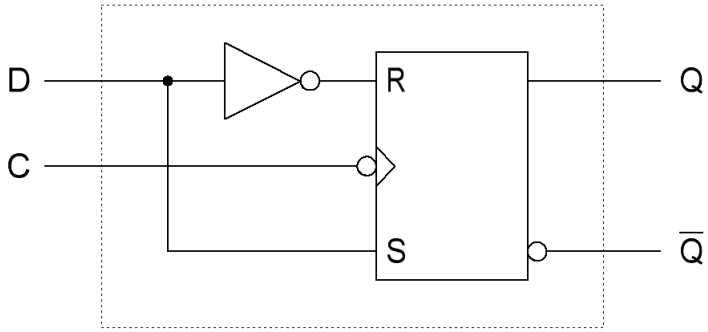
C	D	Q	$\overline{Q}$
↑	0	0	1
↑	1	1	0

Pour finir, voici un exemple de chronogramme pour la bascule D synchronisée sur front montant :



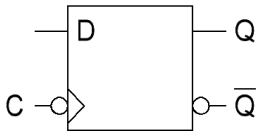
3. La bascule D synchronisée sur front descendant

Soit le circuit suivant :



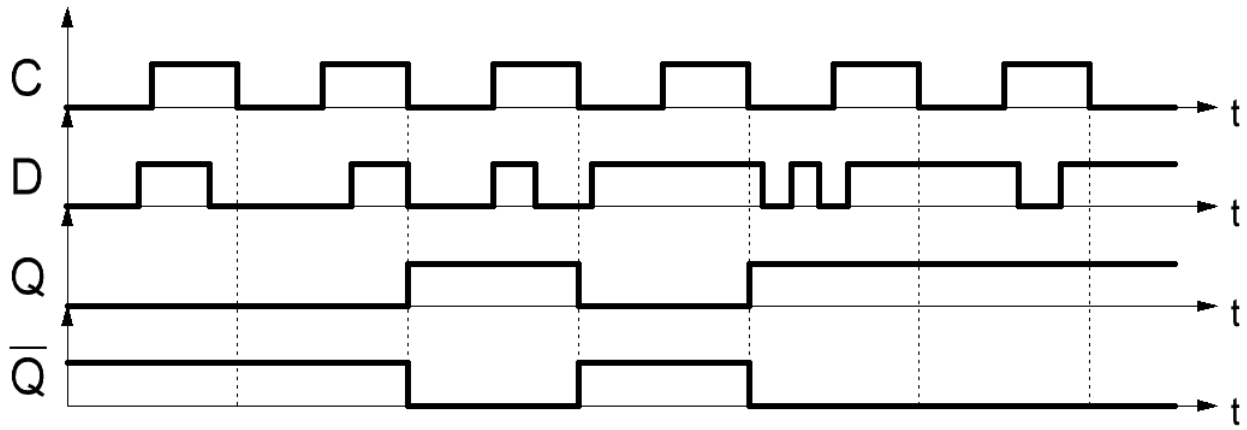
Ce circuit est semblable à la bascule D synchronisée sur front montant sauf que les sorties Q et $\overline{Q}$ sont modifiées uniquement sur les fronts descendants de l'horloge. Autrement dit, la sortie Q copie la valeur de D sur chaque front descendant de l'horloge (les sorties ne changent pas entre deux fronts descendants).

Ce circuit est une **bascule D synchronisée sur front descendant**. Son symbole et sa table de vérité sont les suivants :



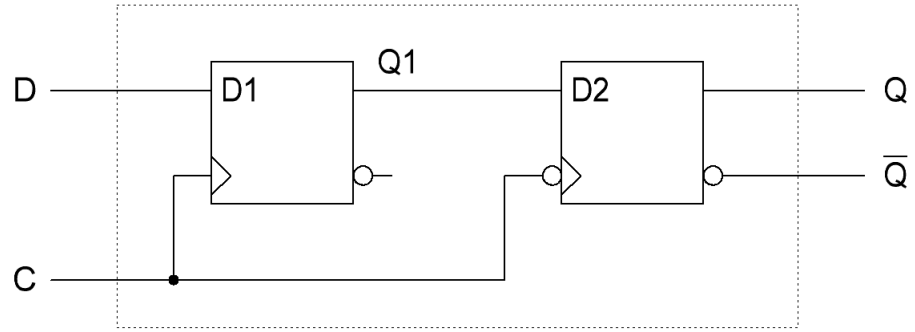
C	D	Q	$\overline{Q}$
↓	0	0	1
↓	1	1	0

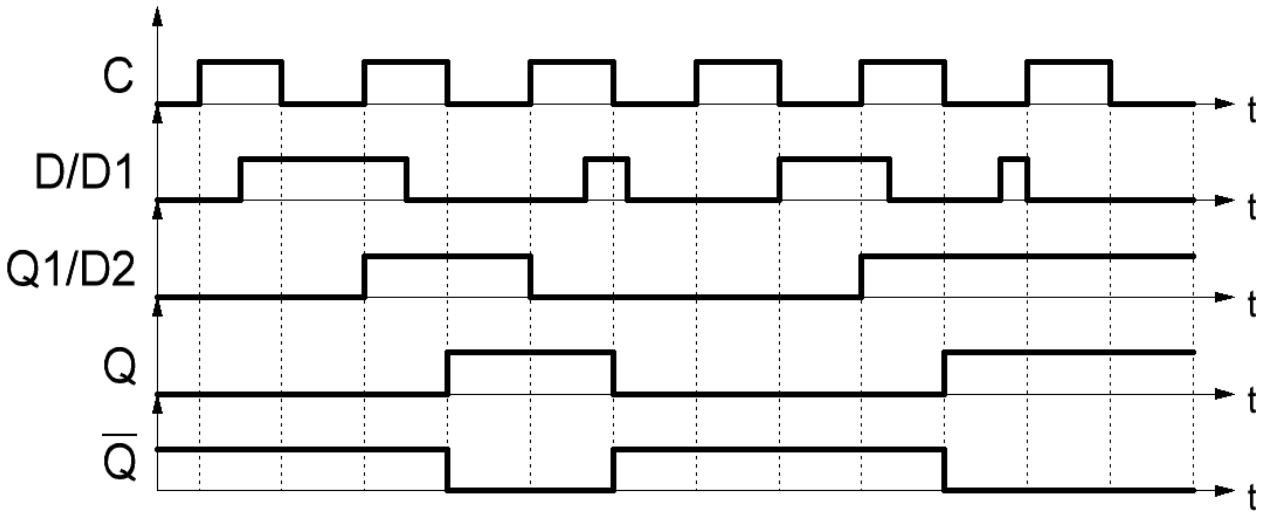
Pour finir, voici un exemple de chronogramme pour la bascule D synchronisée sur front descendant :



4. La bascule D synchronisée sur impulsion (bascule D maître-esclave)

Considérons le circuit ci-dessous et son chronogramme :

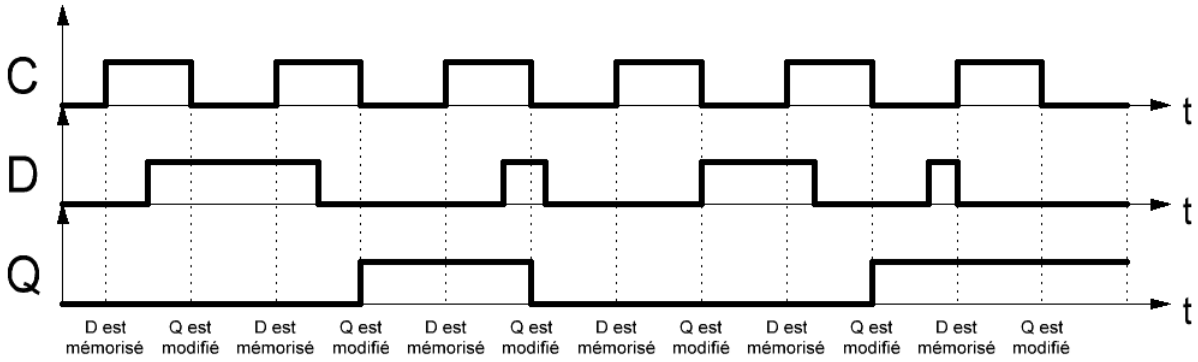




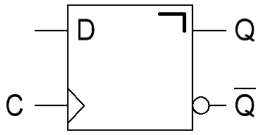
- La première bascule joue le rôle de maître. Sa sortie (*Q1/D2*) est modifiée à chaque front montant du signal d'horloge en fonction de l'entrée *D*. Mais à ce stade, la sortie *Q* n'a pas encore été modifiée.

- La seconde bascule joue le rôle d'esclave. Ses sorties (*Q* et $\overline{Q}$) sont modifiées à chaque front descendant du signal d'horloge en fonction de *Q1*/*D2* ; c'est-à-dire en fonction de la valeur de *D* au moment du précédent front montant.

Concentrons-nous plus particulièrement sur les entrées *C*, *D* et sur la sortie *Q* :



Ce circuit est une **bascule D synchronisée sur impulsion** ou encore une **bascule D maître-esclave**. Son symbole et sa table de vérité sont les suivants :

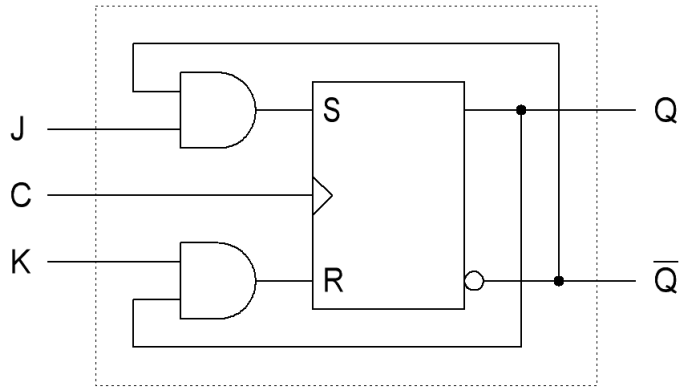


C	**D**	**Q**	**$\overline{Q}$**
⎍	0	0	1
⎍	1	1	0

V. La bascule JK

1. La bascule JK synchronisée sur front montant

Soit le circuit suivant :

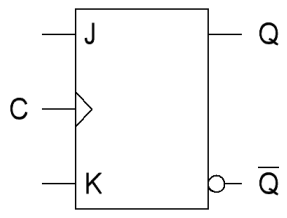


Nous pouvons écrire que :
- $S = J.\overline{Q}$
- $R = K.Q$

En posant q = valeur de Q jute avant le front montant de C, nous obtenons la table de vérité suivante :

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	

Ce circuit est une **bascule JK synchronisée sur front montant**. Son symbole et sa table de vérité sont les suivants :

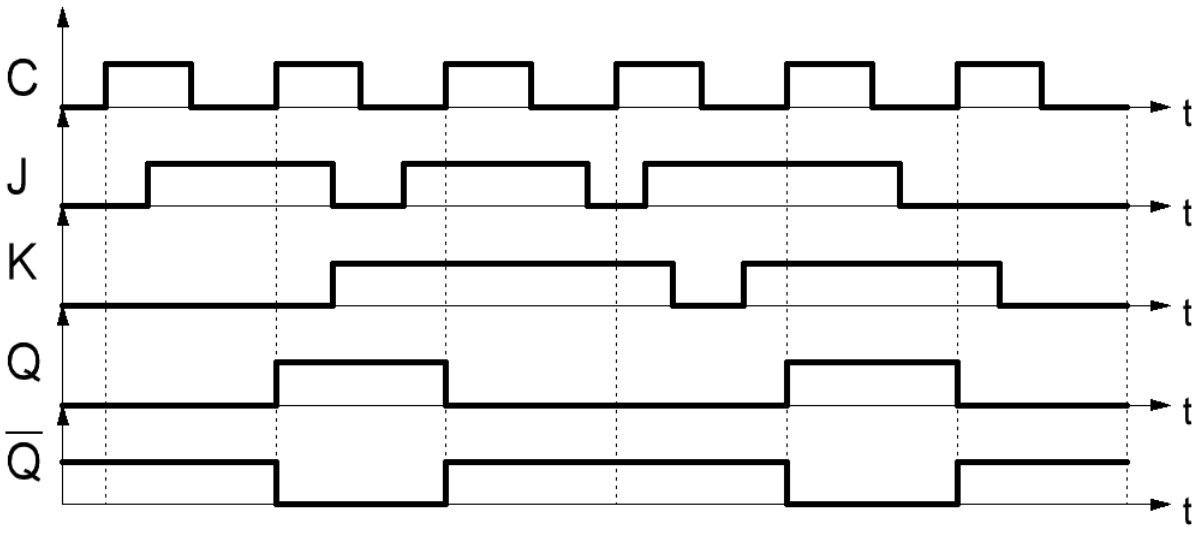


C	J	K	Q	$\overline{Q}$	
↑	0	0	q	$\overline{q}$	← État mémoire
↑	0	1	0	1	← Mise à 0 (*reset*)
↑	1	0	1	0	← Mise à 1 (*set*)
↑	1	1	$\overline{q}$	q	← Basculement

q = valeur de Q juste avant le front montant de C.

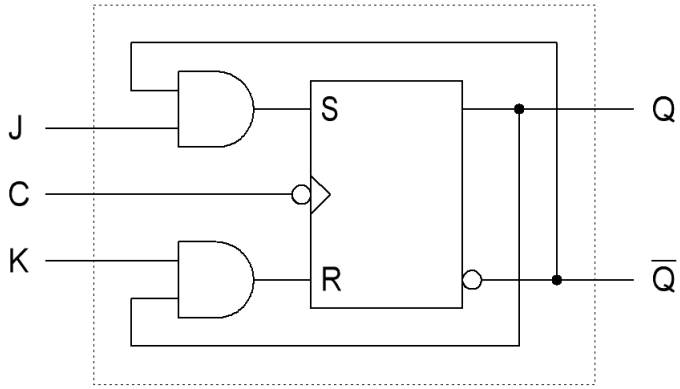
L'avantage d'une bascule JK sur une bascule RS est que l'état interdit est remplacé par l'état de basculement. Quand *J* et *K* sont à 1, la sortie *Q* bascule ; c'est-à-dire qu'elle change d'état sur à chaque front montant de l'horloge.

Pour finir, voici un exemple de chronogramme pour la bascule JK synchronisée sur front montant :

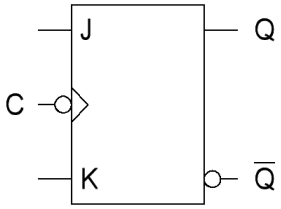


2. La bascule JK synchronisée sur front descendant

Dans le même esprit, il est possible de concevoir une bascule JK synchronisée sur front descendant :



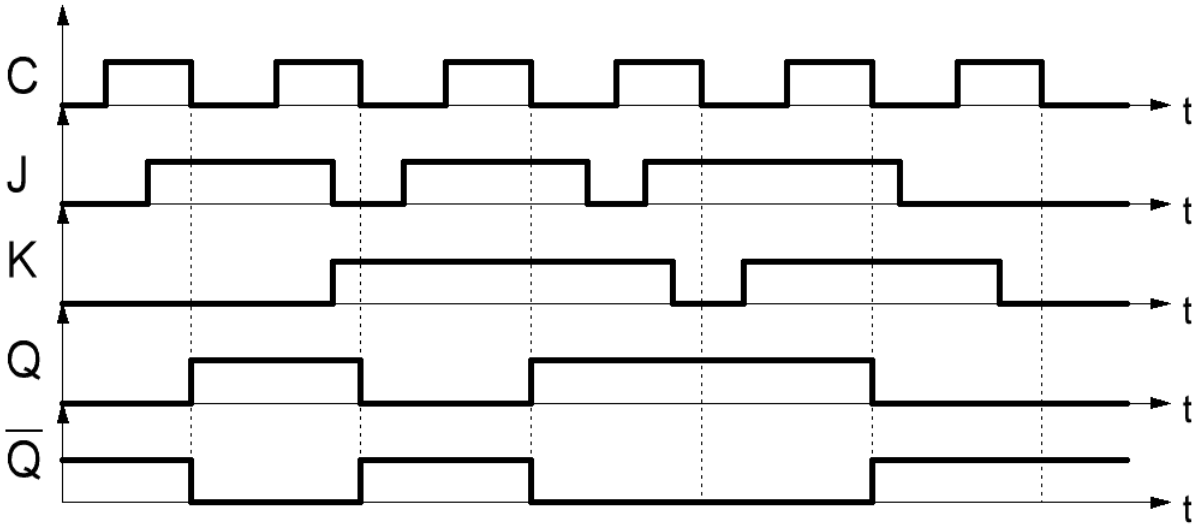
Son symbole et sa table de vérité sont les suivants :



C	J	K	Q	$\overline{Q}$	
↓	0	0	q	$\overline{q}$	← État mémoire
↓	0	1	0	1	← Mise à 0 (*reset*)
↓	1	0	1	0	← Mise à 1 (*set*)
↓	1	1	$\overline{q}$	q	← Basculement

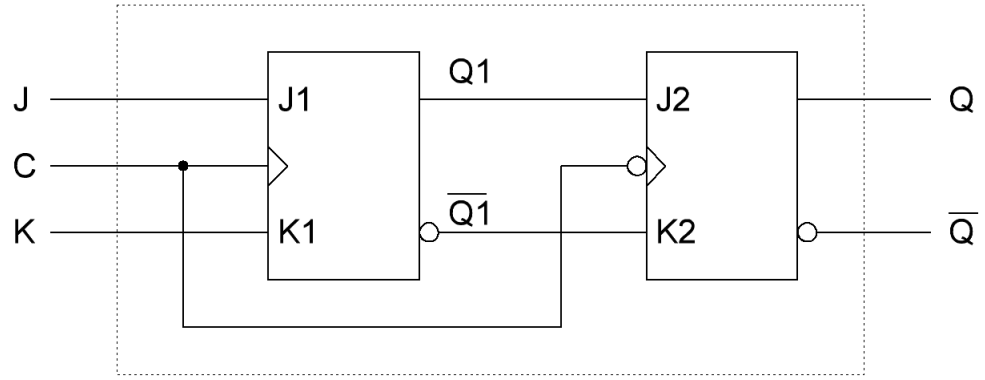
q = valeur de *Q* juste avant le front descendant de *C*.

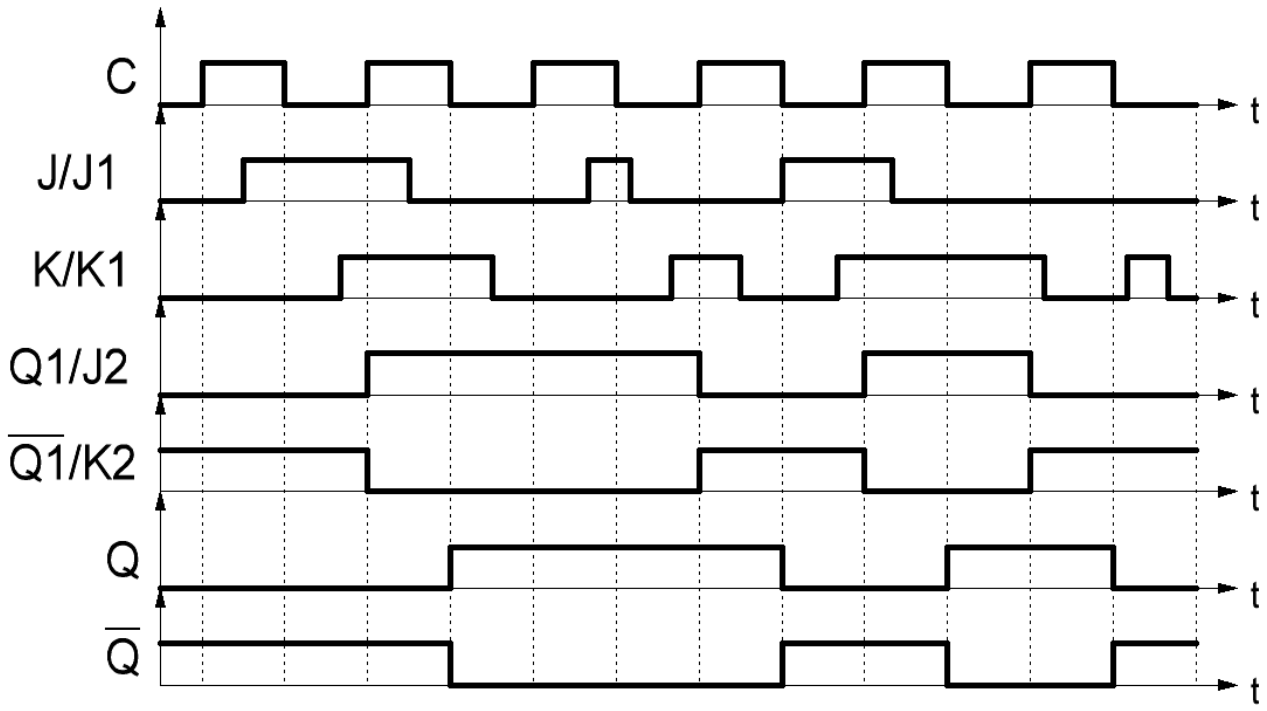
Pour finir, voici un exemple de chronogramme pour la bascule JK synchronisée sur front descendant :



3. La bascule JK synchronisée sur impulsion (bascule JK maître-esclave)

Considérons le circuit ci-dessous et son chronogramme :

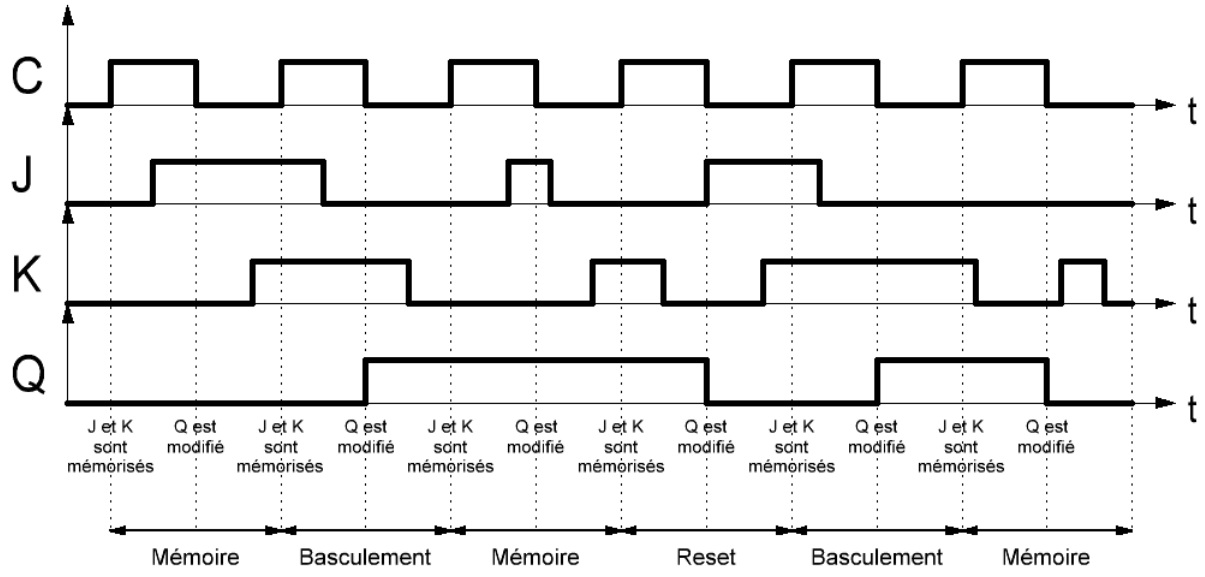




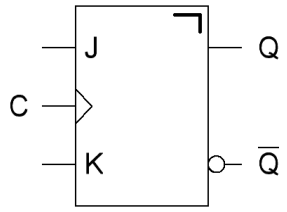
- La première bascule joue le rôle de maître. Ses sorties (*Q1*/J2 et $\overline{Q1}$/*K2*) sont modifiées à chaque front montant du signal d'horloge en fonction des états de *J* et de *K*. Mais à ce stade, la sortie *Q* n'a pas encore été modifiée.

- La seconde bascule joue le rôle d'esclave. Ses sorties (*Q* et $\overline{Q}$) sont modifiées à chaque front descendant du signal d'horloge en fonction des états de *Q1*/J2 et de $\overline{Q1}$/K2 ; c'est-à-dire en fonction des états de *J* et de *K* au moment du précédent front montant.

Concentrons-nous plus particulièrement sur les entrées *C*, *J*, *K* et sur la sortie *Q* :



Ce circuit est une **bascule JK synchronisée sur impulsion** ou encore une **bascule JK maître-esclave**. Son symbole et sa table de vérité sont les suivants :



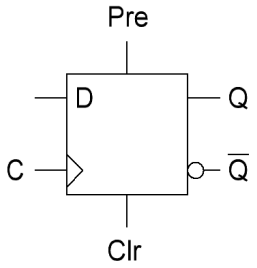
C	**J**	**K**	**Q**	**$\overline{Q}$**	
⎍	0	0	q	$\overline{q}$	← État mémoire
⎍	0	1	0	1	← Mise à 0 (*reset*)
⎍	1	0	1	0	← Mise à 1 (*set*)
⎍	1	1	$\overline{q}$	q	← Basculement

q = valeur de *Q* juste avant le front montant de *C*.

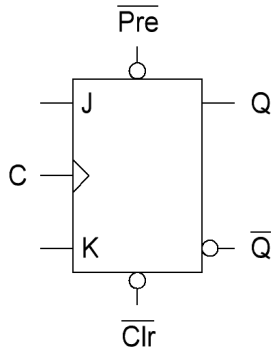
VI. Entrées de forçage asynchrone (*Preset* et *Clear*)

La plupart des circuits intégrés de bascules D et JK sont munis d'entrées asynchrones prioritaires de mise à 0 (appelées *clear* ou *reset*) et de mise à 1 (appelée *preset* ou *set*). Certaines sont actives à l'état haut, d'autres à l'état bas.

Exemples :



Entrées *preset* et *clear* actives à l'état haut



Entrées *preset* et *clear* actives à l'état bas

Ces entrées sont utilisées pour forcer à 1 (*preset*) ou à 0 (*clear*) la sortie *Q* quel que soit l'état des autres entrées. Elles sont donc asynchrones et prioritaires.

- Quand l'entrée *preset* est active, la sortie *Q* est mise à 1 immédiatement.
- Quand l'entrée *clear* est active, la sortie *Q* est mise à 0 immédiatement.

Ces deux entrées ne doivent jamais être actives en même temps (cela est interdit).

Exemple d'un chronogramme d'une bascule JK synchronisée sur front montant avec $\overline{clear}$ et $\overline{preset}$: