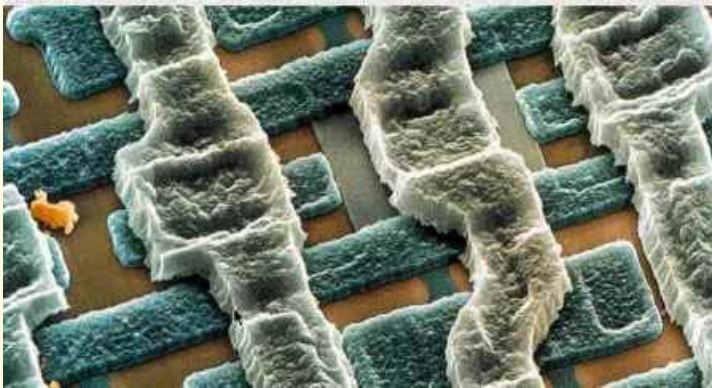
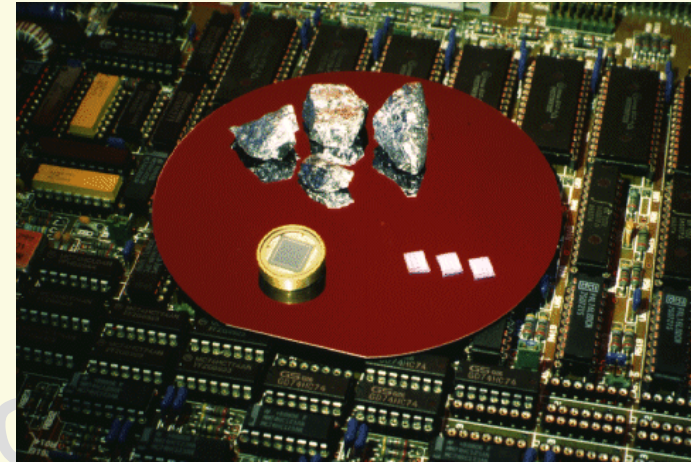
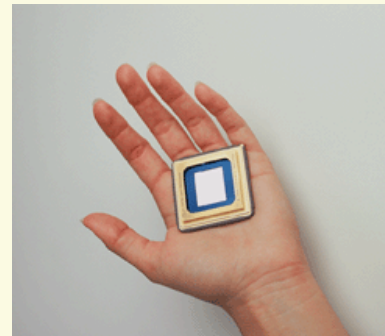


Micro électronique

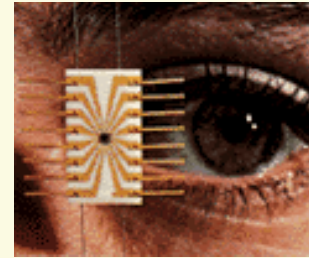


Logique Séquentielle

Dominique GINHAC
dginhac@u-bourgogne.fr



Plan du cours



1- Présentation générale et historique

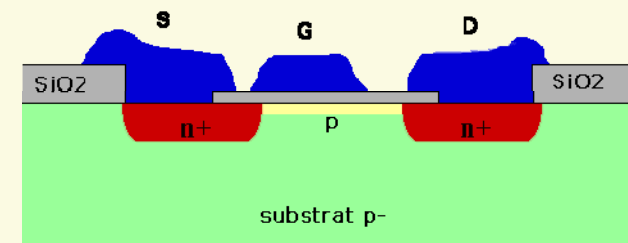
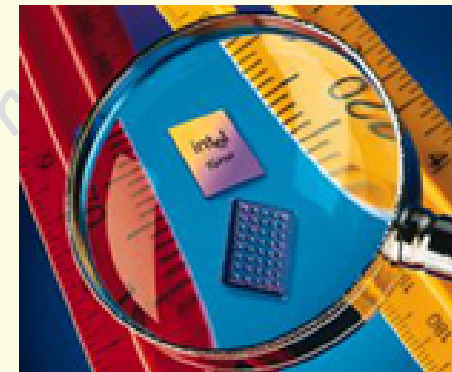
2- Physique des semi conducteurs

3- Technologie CMOS

4- Design de portes élémentaires

5- Design de fonctions complexes

6- Technologie des composants



Logique séquentielle

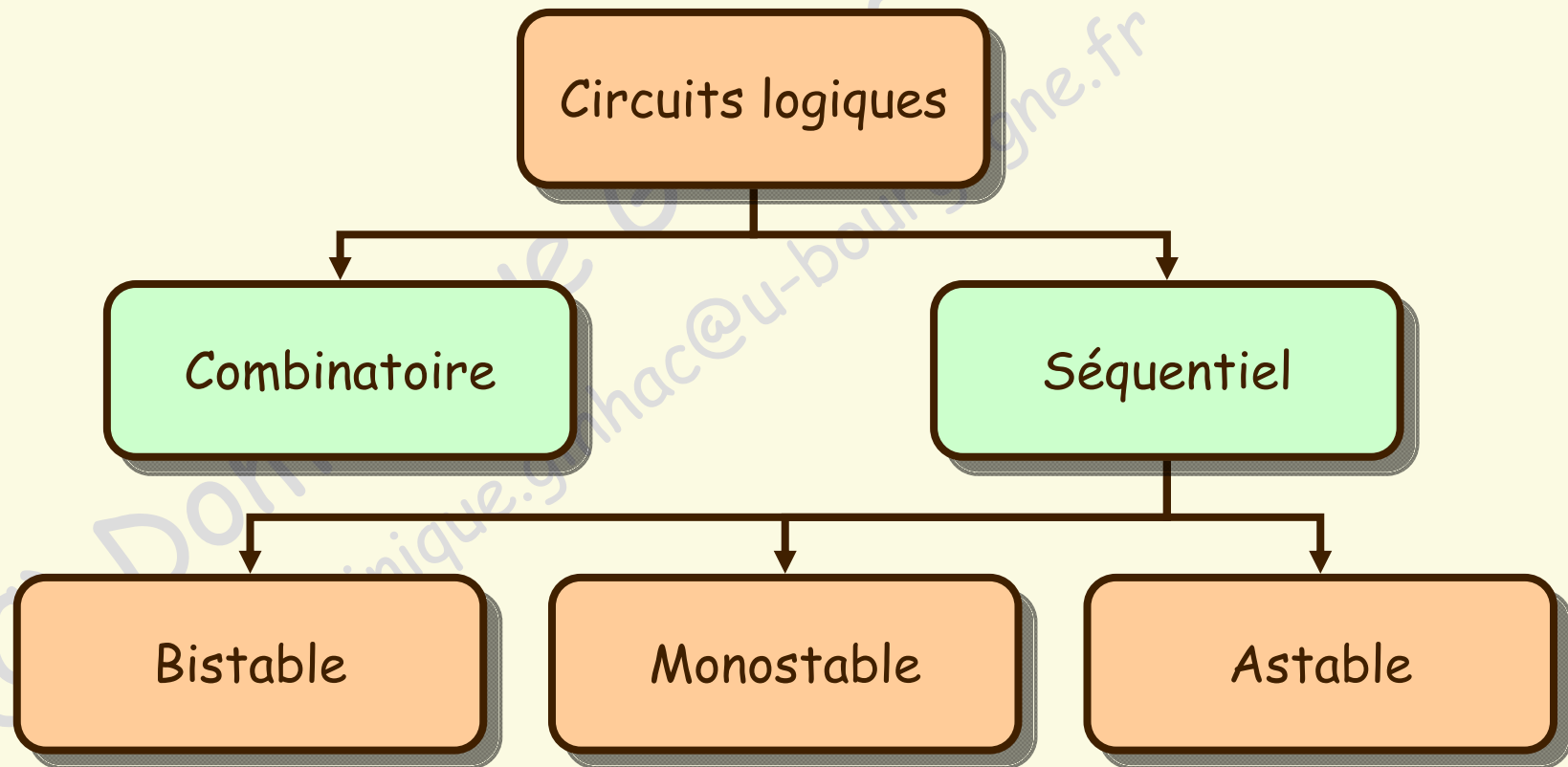
Généralités



La sortie est déterminée à partir des **entrées à l'état courant** et des **sorties à l'état précédent**

Logique séquentielle

Généralités



Systèmes bistables

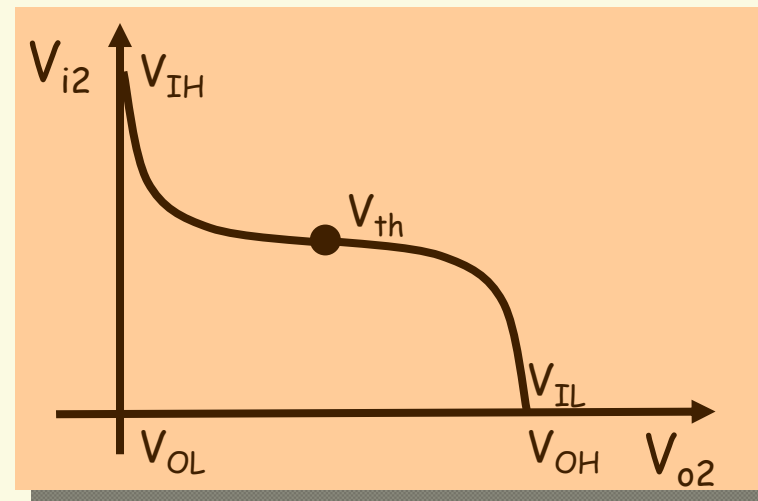
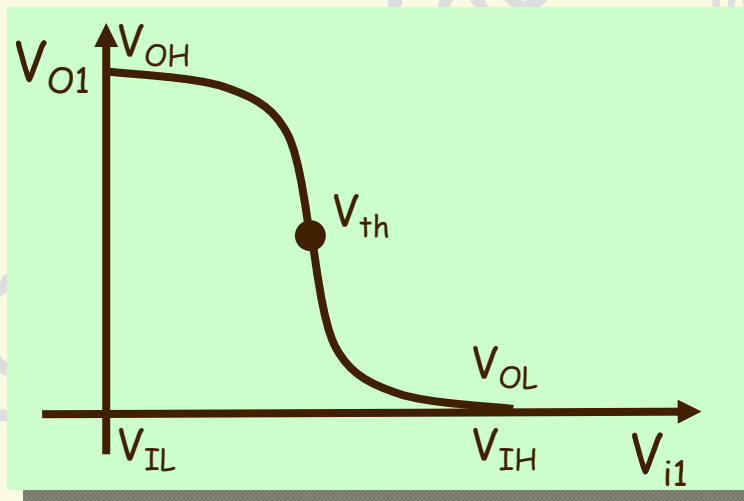
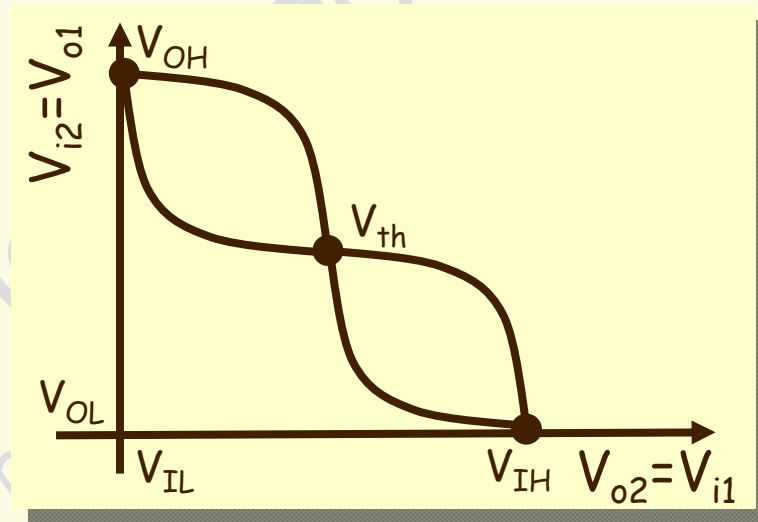
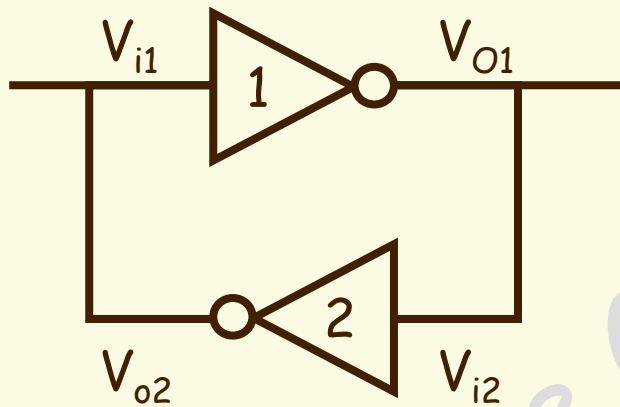
Par définition, un **système bistable** possède **2 états stables**

Des exemples très courants en électronique numérique :

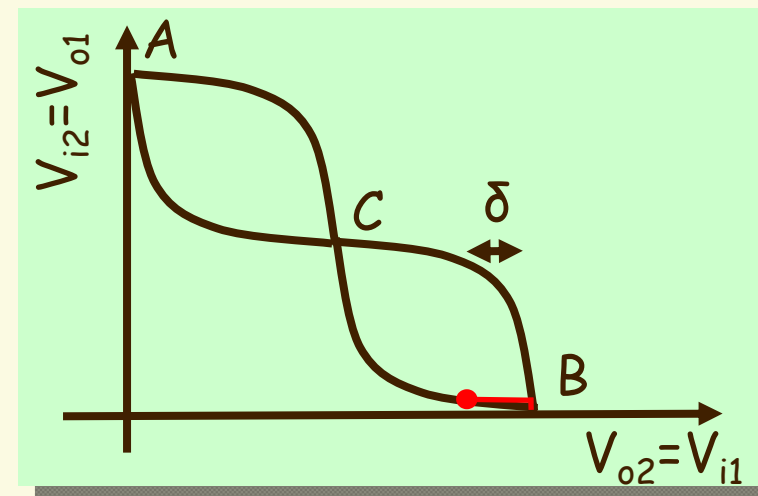
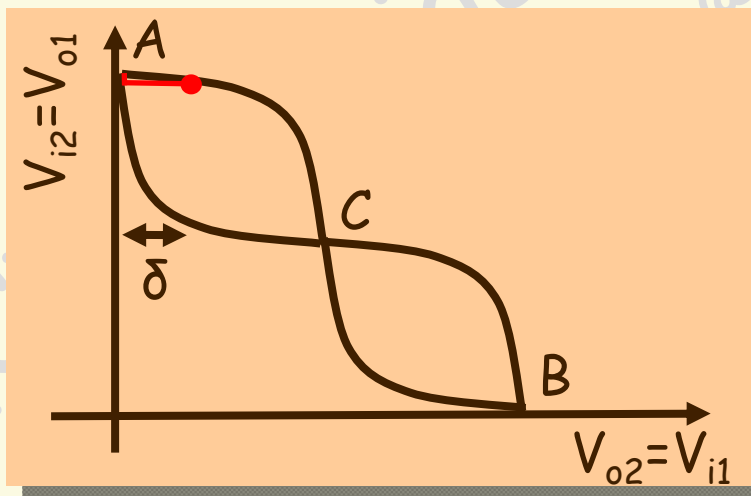
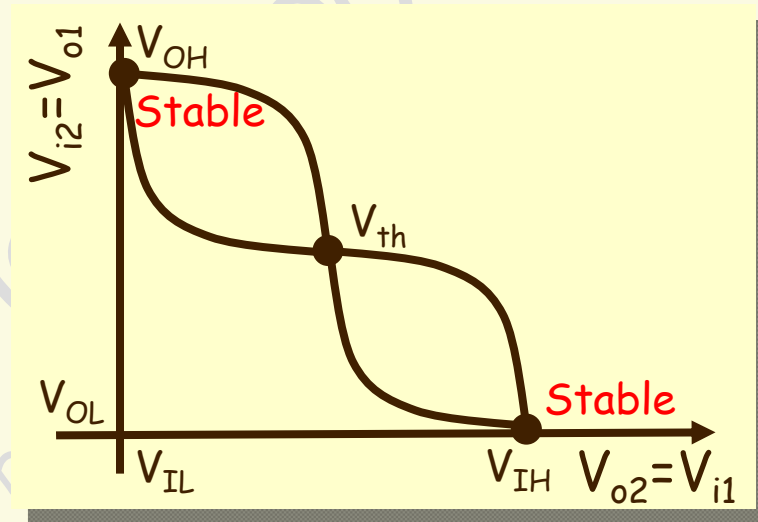
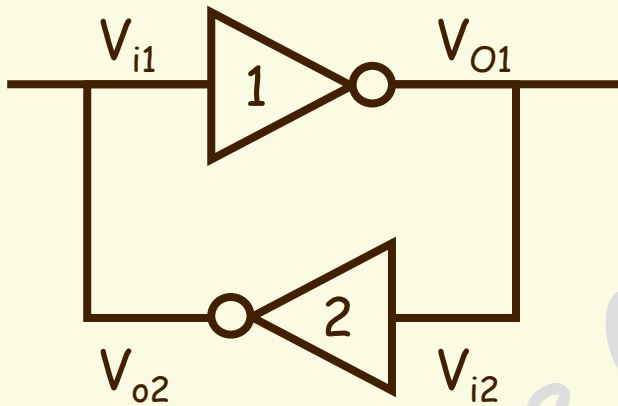
- ✓ **Bascules** (Latch, Flip Flop),
- ✓ **Registres à décalages**,
- ✓ **Éléments de mémoire.**

Considérons tout d'abord un **exemple trivial** de système bistable composé de **2 inverseurs**

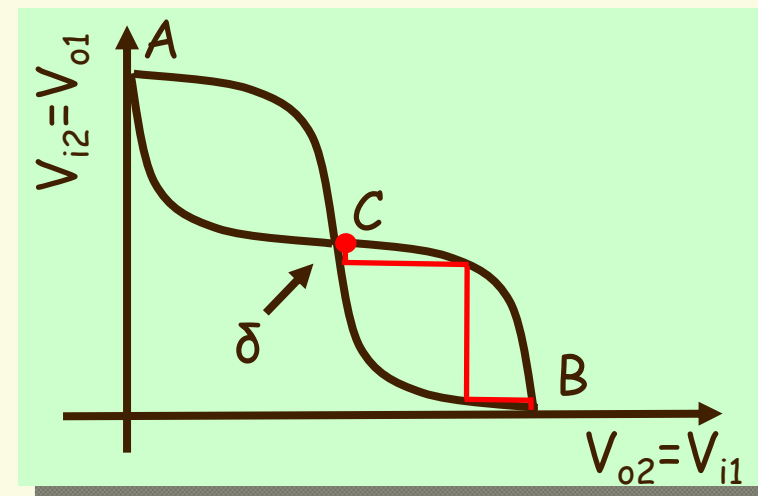
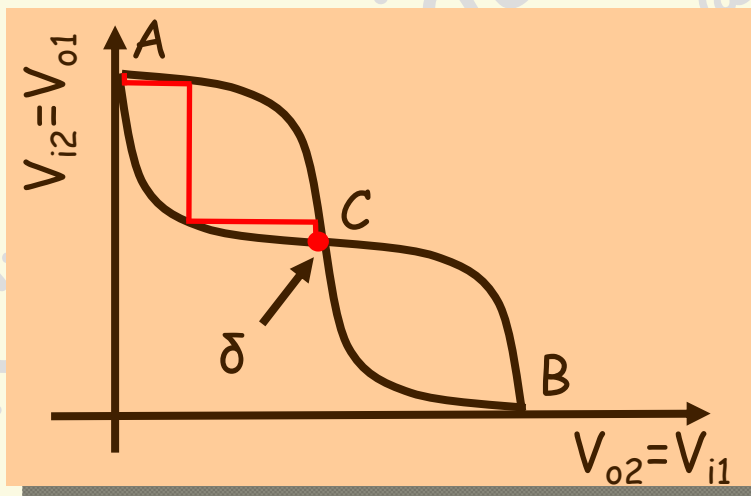
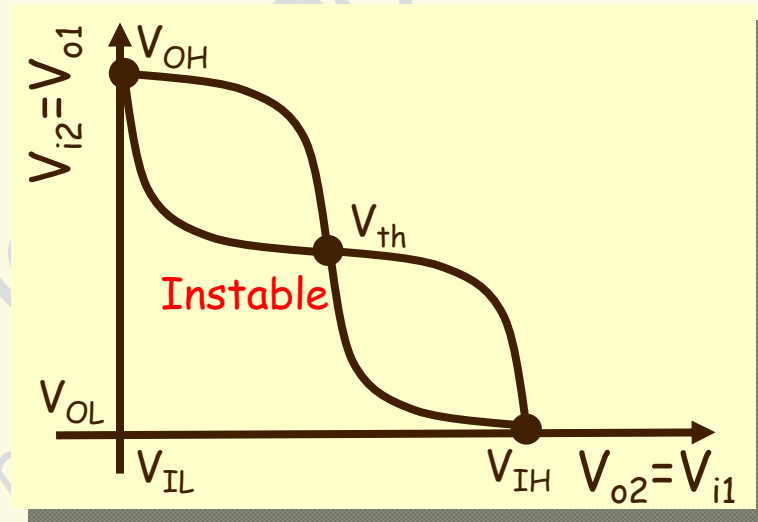
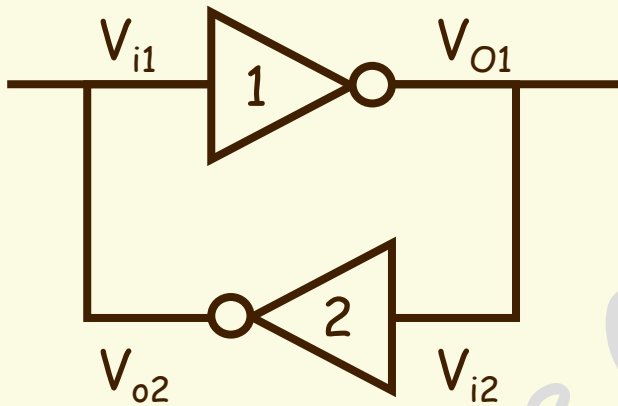
Un exemple de système bistable



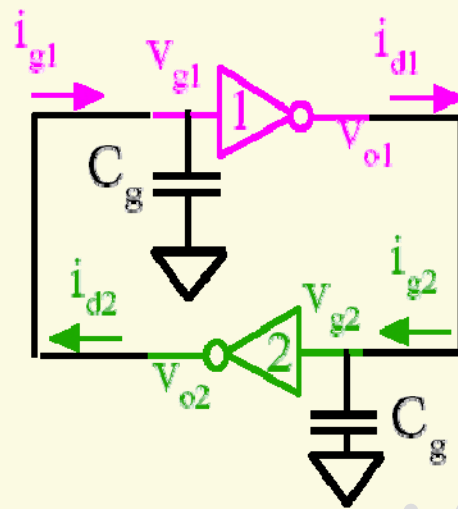
Un exemple de système bistable



Un exemple de système bistable



Un exemple de système bistable



On suppose :

$$C_g \gg C_d$$

$$V_{o1}(0) = V_{o2}(0) = V_{th}$$

$$i_{g1} = i_{d2} = g_m V_{g2} \text{ avec } i_{g1} = C_g \frac{dv_{g1}}{dt} \text{ et } V_{o1} = V_{g2}$$

$$i_{g2} = i_{d1} = g_m V_{g1} \text{ avec } i_{g2} = C_g \frac{dv_{g2}}{dt} \text{ et } V_{o2} = V_{g1}$$

$$\left[\begin{array}{l} g_m V_{g2} = C_g \frac{dv_{g1}}{dt} \\ g_m V_{g1} = C_g \frac{dv_{g2}}{dt} \end{array} \right] \Rightarrow V_{oi} = \left(\frac{C_g}{g_m} \right)^2 \frac{d^2 V_{oi}}{dt^2} \quad \text{for } i = 1, 2$$

$$= \left(\frac{1}{\tau_0} \right)^2 \frac{d^2 V_{oi}}{dt^2}$$

Un exemple de système bistable

$$v_{oi} = \left(\frac{C_g}{g_m} \right)^2 \frac{d^2 v_{oi}}{dt^2} = \left(\frac{1}{\tau_0} \right)^2 \frac{d^2 v_{oi}}{dt^2} \quad \text{for } i = 1, 2$$

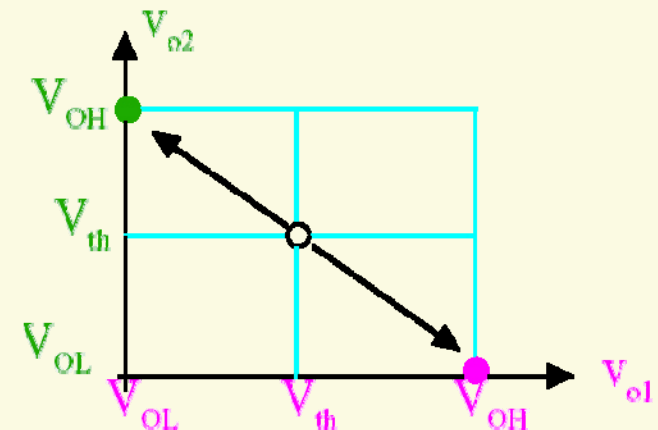
donc

$$\begin{cases} v_{o1}(t) = \frac{1}{2} \left(v_{o1}(0) - \tau_0 v'_{o1}(0) \right) e^{-\frac{t}{\tau_0}} + \frac{1}{2} \left(v_{o1}(0) + \tau_0 v'_{o1}(0) \right) e^{+\frac{t}{\tau_0}} \\ v_{o2}(t) = \frac{1}{2} \left(v_{o2}(0) - \tau_0 v'_{o2}(0) \right) e^{-\frac{t}{\tau_0}} + \frac{1}{2} \left(v_{o2}(0) + \tau_0 v'_{o2}(0) \right) e^{+\frac{t}{\tau_0}} \end{cases}$$

(Note: Red arrows in the original image point to the exponential terms with the note "≈ 0 si t >> τ₀")

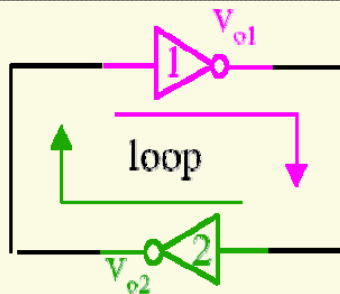
OR $v'_{o1}(0) = \frac{dv_{o1}}{dt}(0) = -v'_{o2}(0) = \frac{dv_{o2}}{dt}(0)$

$v_{o1}; V_{th} \rightarrow V_{OH} \text{ or } V_{OL}$
 $v_{o2}; V_{th} \rightarrow V_{OL} \text{ or } V_{OH}$



Un exemple de système bistable

Le passage d'un état instable à un état stable nécessite plusieurs passages par les 2 inverseurs

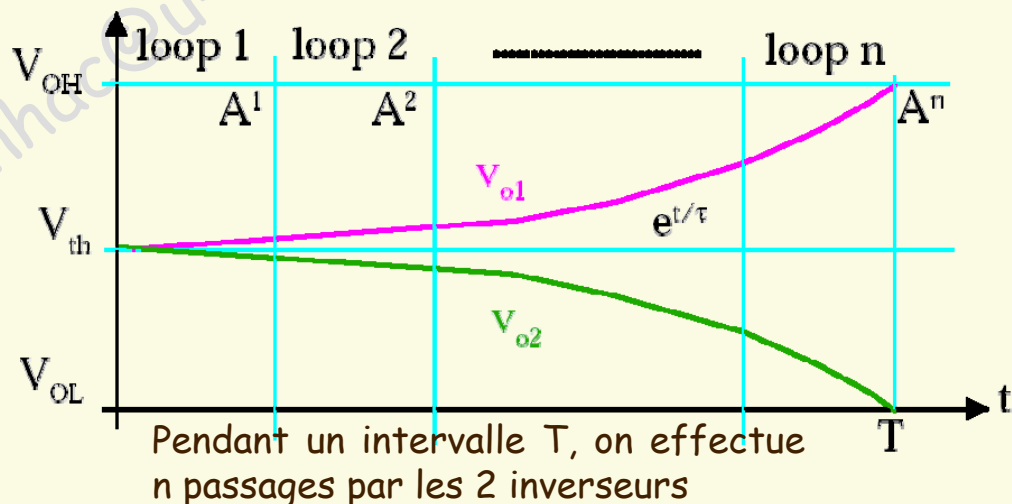


Temporellement, la sortie des inverseurs s'exprime sous la forme :

$$\frac{v_{o1}(t)}{v_{o1}(0)} \approx e^{+\frac{t}{\tau_0}}$$

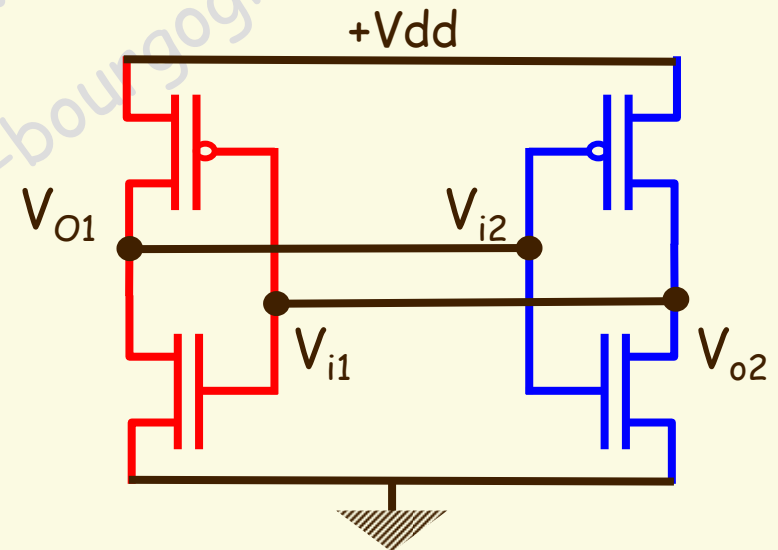
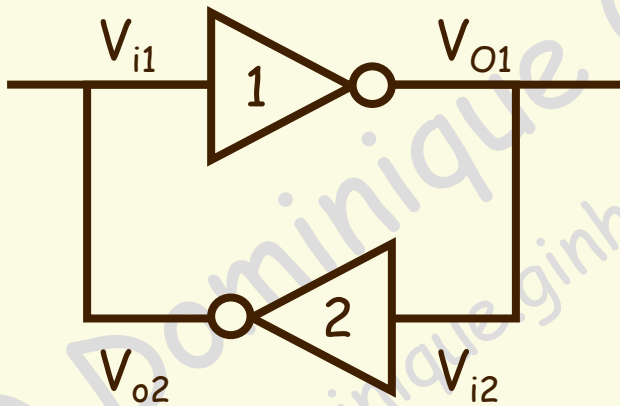
On a alors : $A^n \approx e^{+\frac{T}{\tau_0}}$

avec A représentant le gain en tension des 2 inverseurs



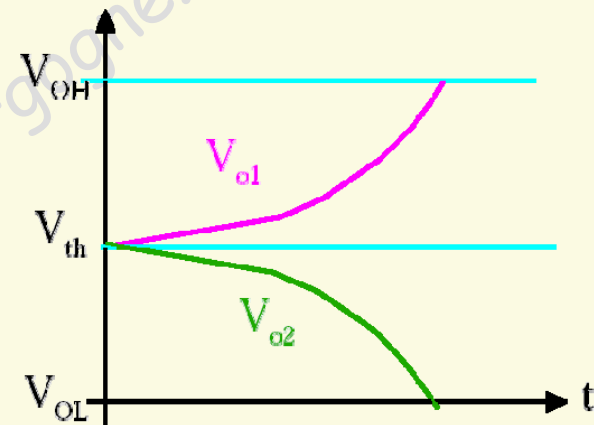
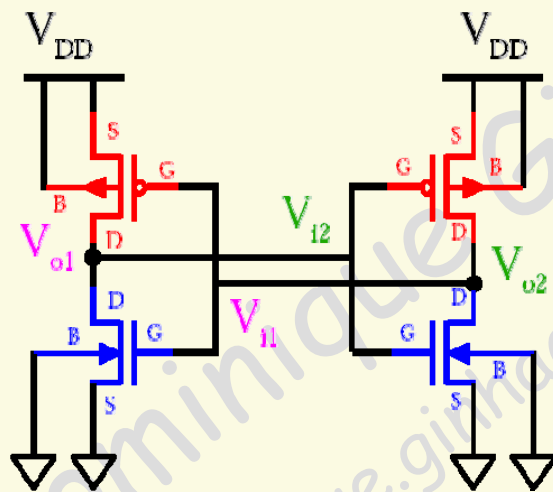
Un exemple de système bistable

En technologie CMOS



Un exemple de système bistable

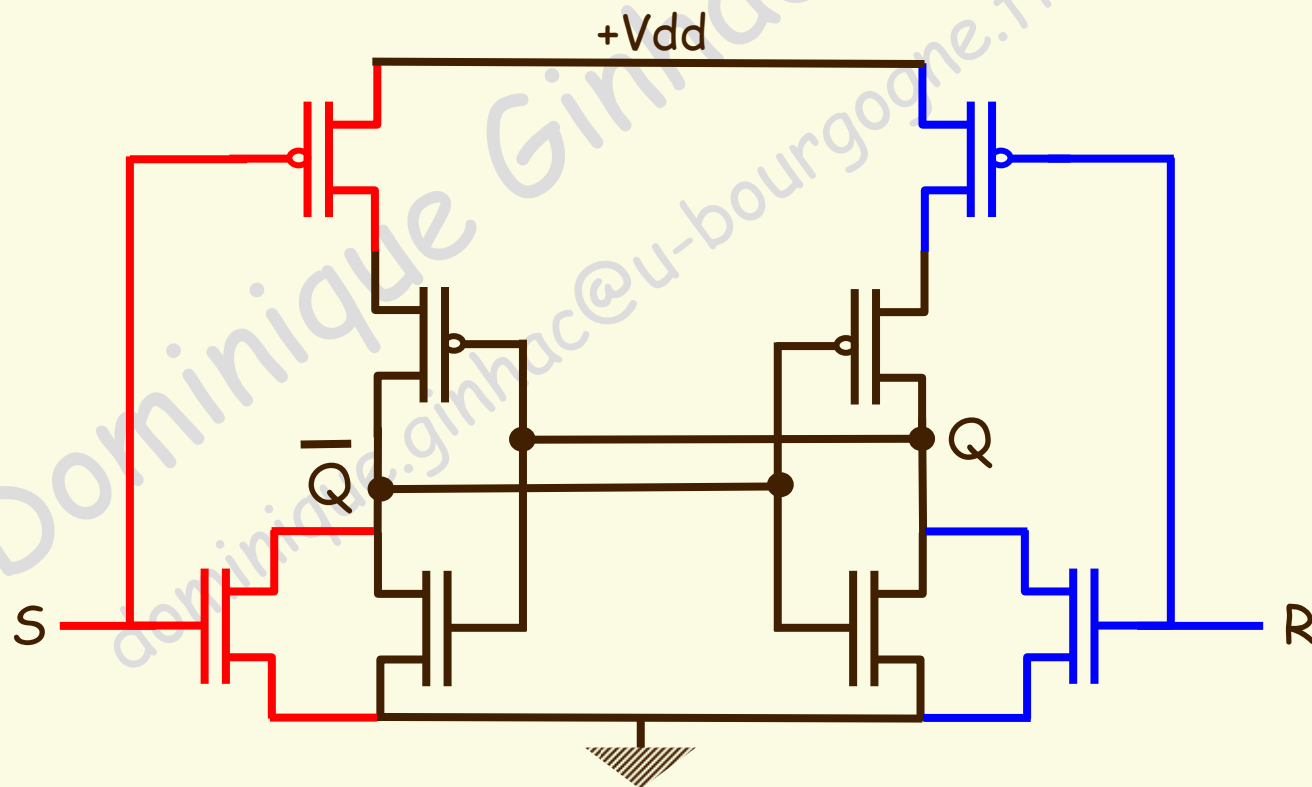
En technologie CMOS



Problème : Ce système va se **figer** dans un des 2 états stables (tant que le système est alimenté) et ne peut pas facilement **commuter** vers le deuxième état stable

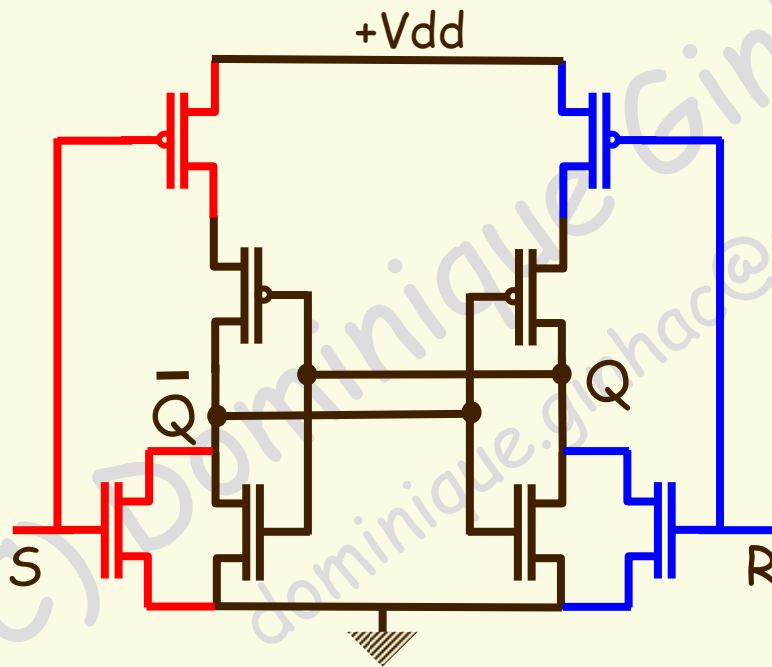
Bascule RS

Pour permettre le passage d'un état à l'autre, il faut modifier le schéma précédent



Bascule RS

Principe de fonctionnement :



✓ 2 entrées : S et R

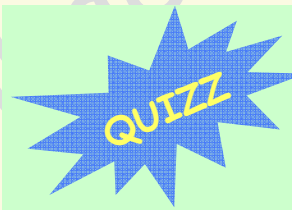
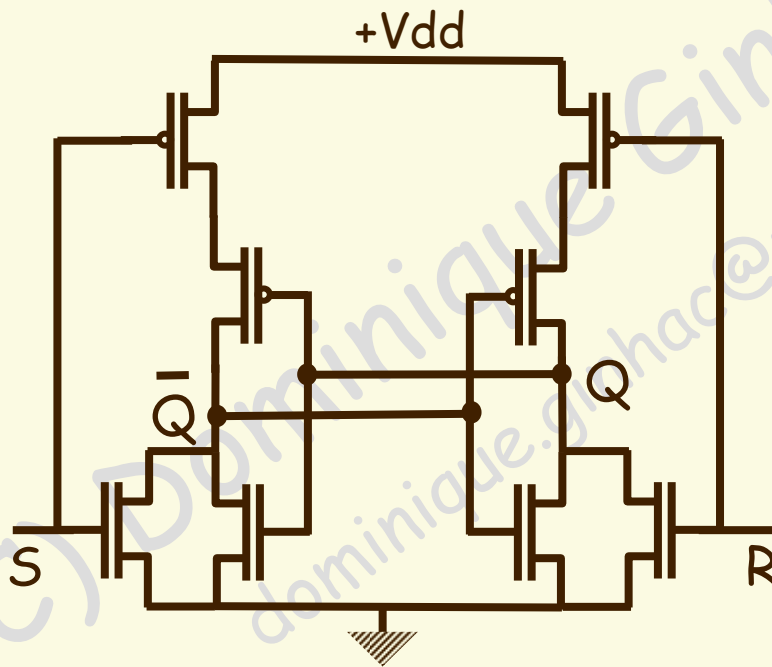
✓ 2 sorties : Q et \bar{Q}

La bascule RS est dite dans l'état « **Set** » si $Q=1$ et $\bar{Q}=0$

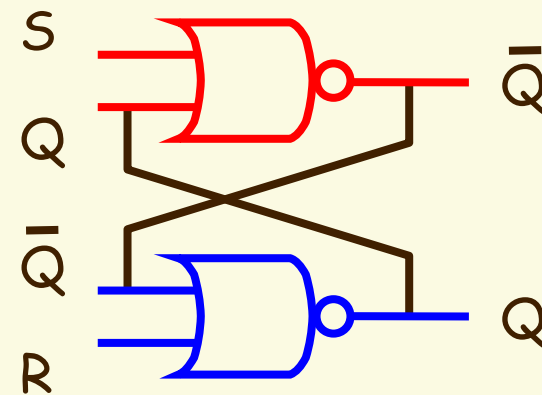
La bascule RS est dite dans l'état « **Reset** » si $Q=0$ et $\bar{Q}=1$

Bascule RS

Principe de fonctionnement :

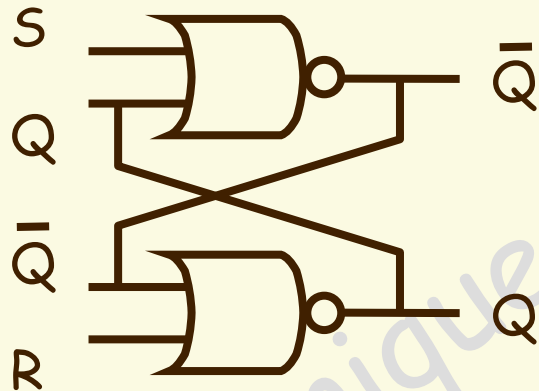


2 Nmos en parallèle
2 Pmos en série



Bascule RS

Principe de fonctionnement :



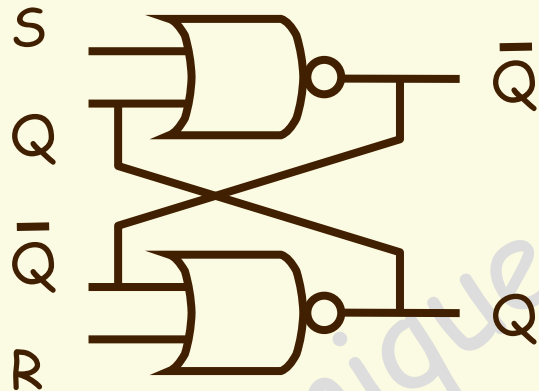
Les **entrées** de commande **S** et **R** sont des **entrées directes** qui agissent **directement** sur l'état des sorties **Q** et **\bar{Q}**

S = 0, R = 0 : La **sortie** des portes NOR est égale à l'**inverse** de la **deuxième entrée**

Maintien (hold) de l'**état précédent** pour les sorties **Q** et **\bar{Q}**

Bascule RS

Principe de fonctionnement :



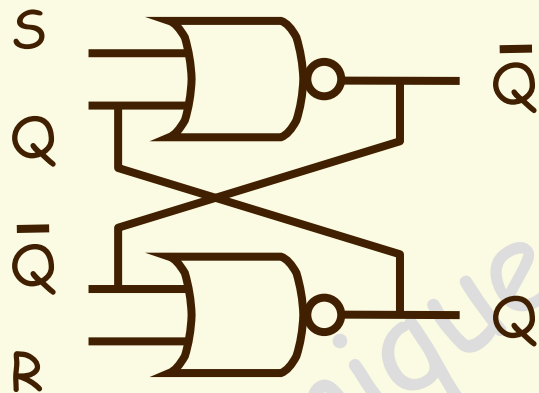
Les **entrées** de commande **S** et **R** sont des **entrées directes** qui agissent **directement** sur l'état des sorties **Q** et **Q̄**

S = 1, R = 0 : La **sortie Q** est **forcée** à **1** et la **sortie Q̄** est **forcée** à **0**

La **bascul**e est mis à « **set** » quel que soit l'état précédent

Bascule RS

Principe de fonctionnement :



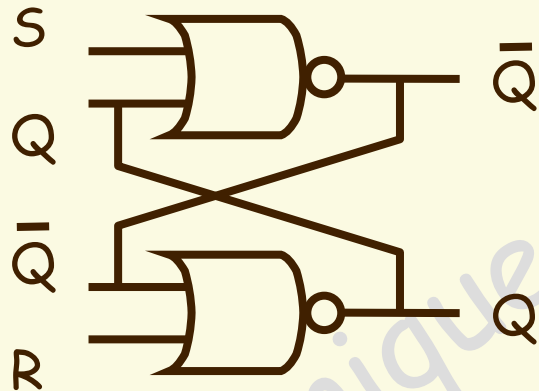
Les **entrées** de commande **S** et **R** sont des **entrées directes** qui agissent **directement** sur l'état des sorties **Q** et **Q̄**

S = 0, R = 1 : La **sortie Q** est **forcée** à **0** et la **sortie Q̄** est **forcée** à **1**

La **bascul**e est mis à « **reset** » quel que soit l'état précédent

Bascule RS

Principe de fonctionnement :



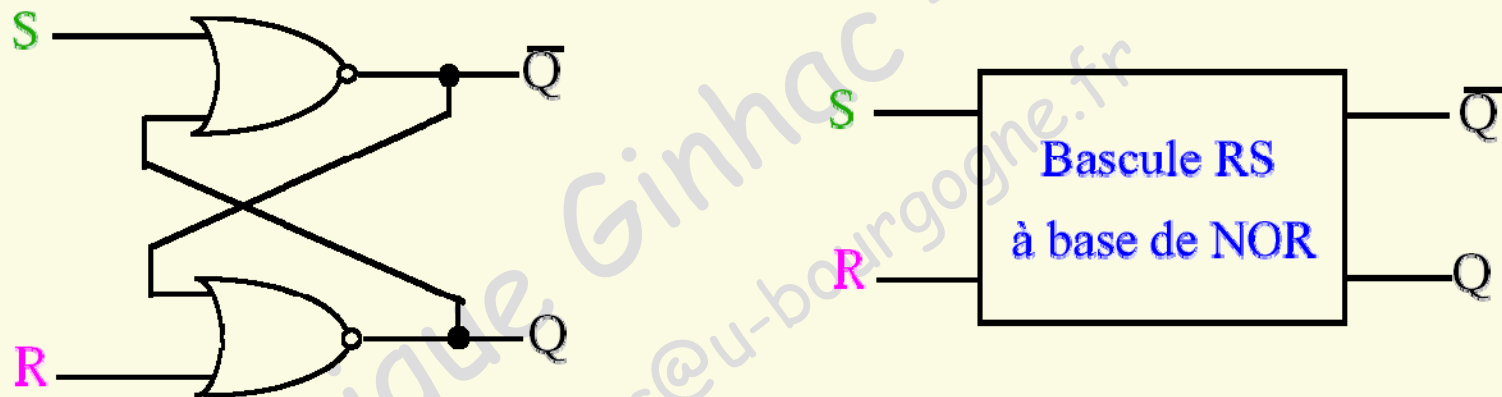
Les **entrées** de commande **S** et **R** sont des **entrées directes** qui agissent **directement** sur l'état des sorties **Q** et **\bar{Q}**

$S = 1, R = 1$: La **sortie Q** est **forcée à 0** et la **sortie \bar{Q}** est **forcée à 0**

Cas INTERDIT (non autorisé en fonctionnement normal)

Bascule RS

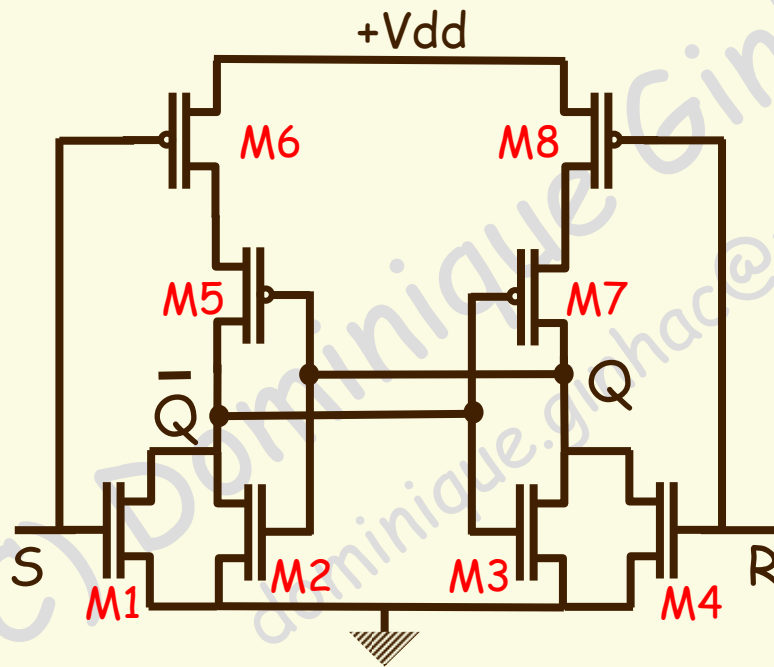
En résumé :



S	R	Q_{n+1}	\overline{Q}_{n+1}	Operation
0	0	Q_n	\overline{Q}_n	maintien
1	0	1	0	set
0	1	0	1	reset
1	1	0	0	NON Autorisé

Bascule RS

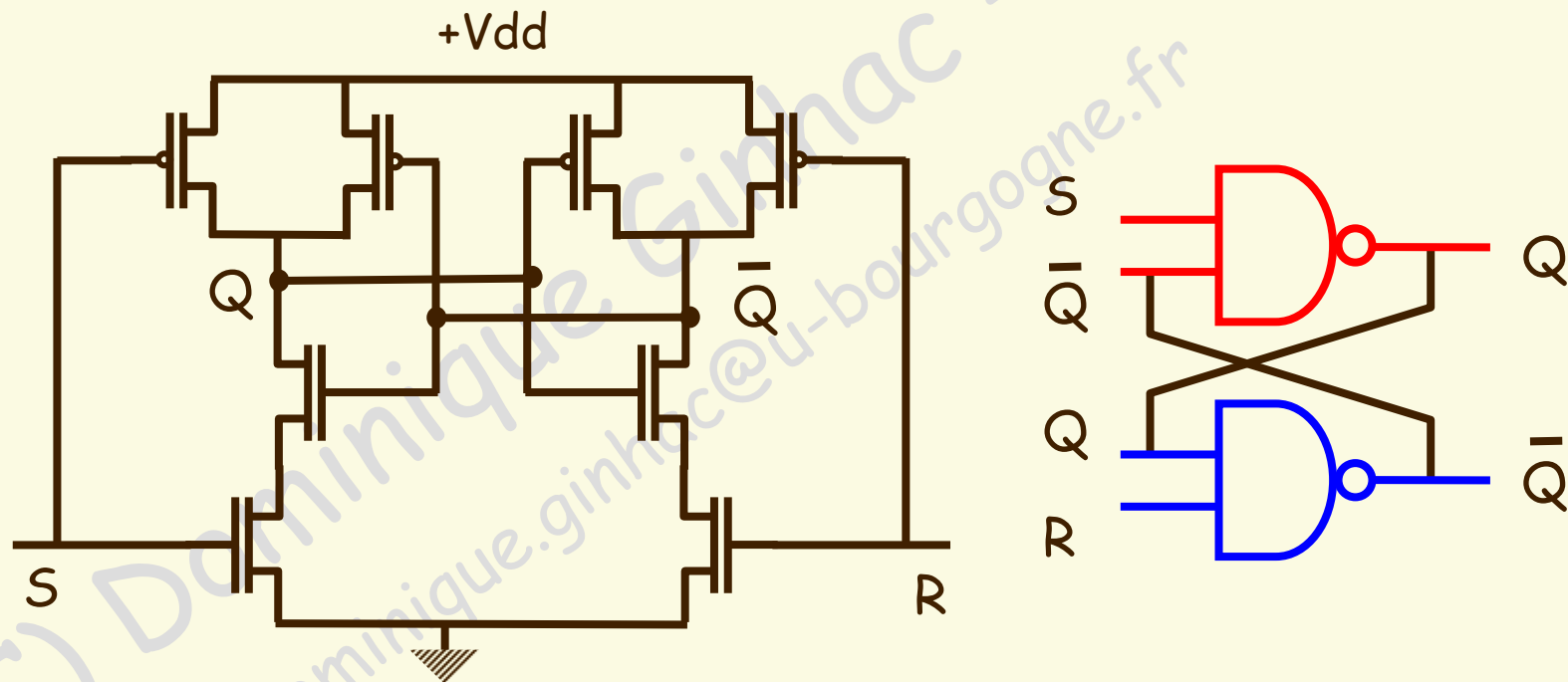
Au niveau transistor :



S	R	Q_n	\bar{Q}_n	transistors
0	0	0	1	M1,M4, M2 bloqués M3 passant
0	0	1	0	M1,M3,M4 bloqués M2 passant
1	0	1	0	M1,M2 passants M3, M4 bloqués
0	1	0	1	M1,M2 bloqués M3, M4 passants
1	1	0	0	M1,M4 passants M2, M3 bloqués

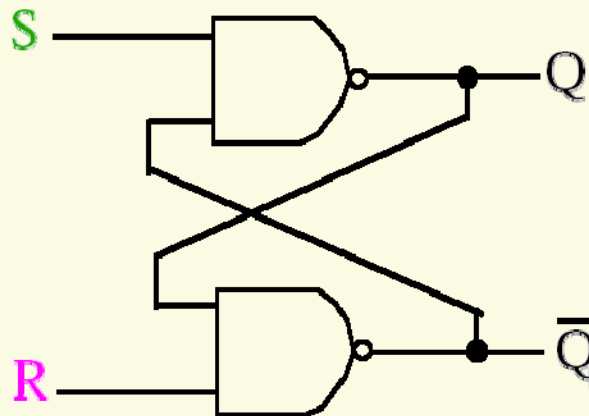
Bascule RS

Une autre version en portes NAND



Fonctionnement dual de la version précédente

Bascule RS

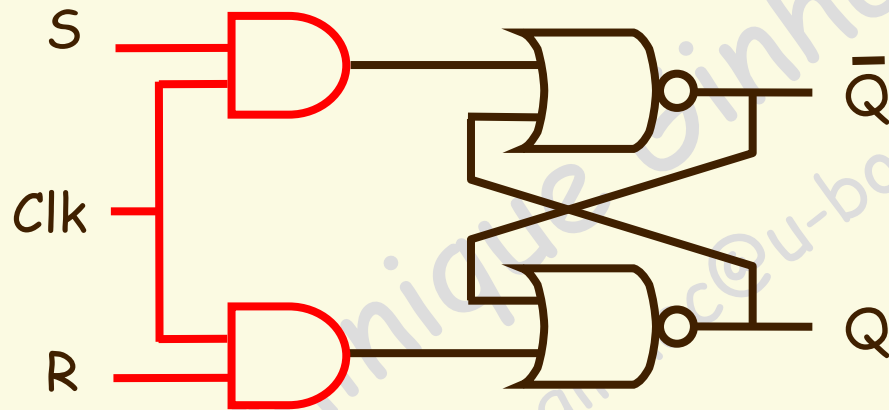


S	R	Q_{n+1}	\bar{Q}_{n+1}	Operation
0	0	1	1	NON Autorisé
0	1	1	0	set
1	0	0	1	reset
1	1	Q_n	\bar{Q}_n	maintien

NOTE: S, R (NAND2) = \bar{S}, \bar{R} (NOR2)

Bascule RS synchrone

Utilisation d'une **horloge** pilotant la bascule RS



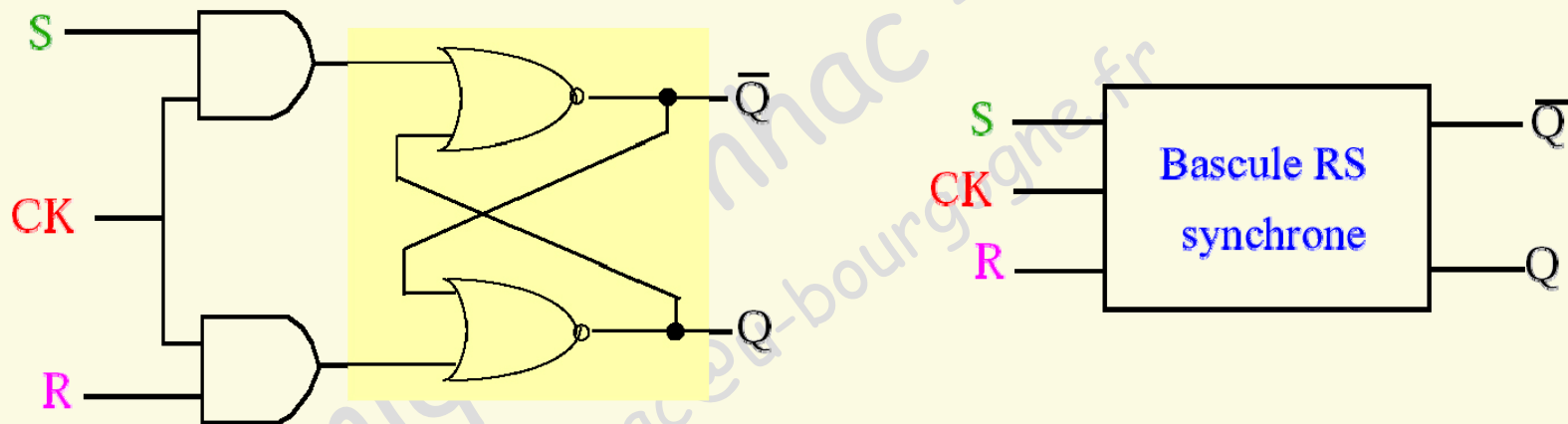
Les **entrées** appliquées à **R** et **S** sont **prises en compte** uniquement si **Clk = 1**

Si **Clk=0**, il n'y a aucune influence des entrées **R** et **S** : **Maintien des sorties**

Si **Clk=1**, on est dans le cas du **fonctionnement classique** de la **bascule RS asynchrone**

Bascule RS synchrone

En résumé :

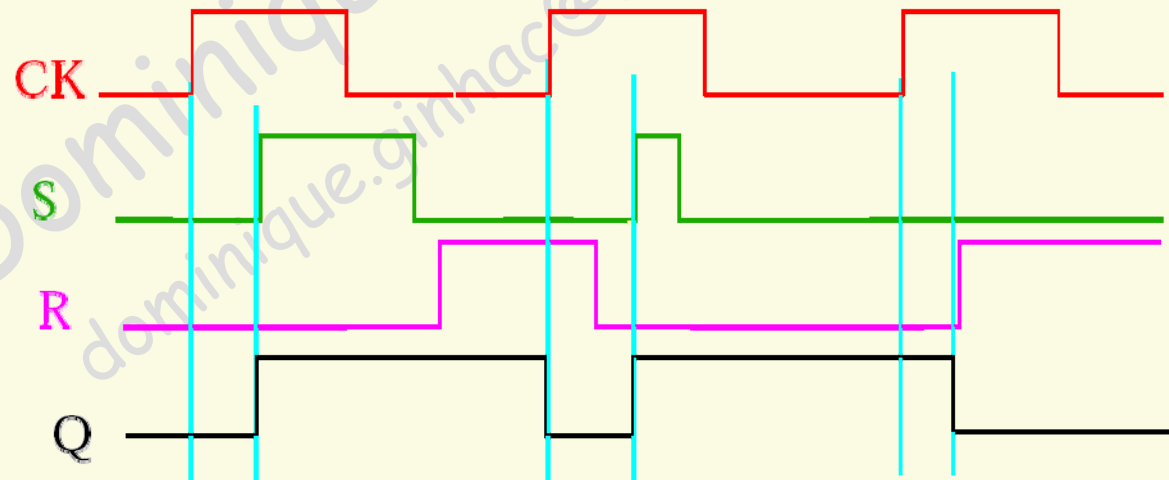


CK	S	R	Q_{n+1}	\bar{Q}_{n+1}	Operation
0	X	X	Q_n	\bar{Q}_n	maintien
1	0	0	Q_n	\bar{Q}_n	maintien
1	1	0	1	0	set
1	0	1	0	1	reset
1	1	1	0	0	NON Autorisé

Bascule RS synchrone

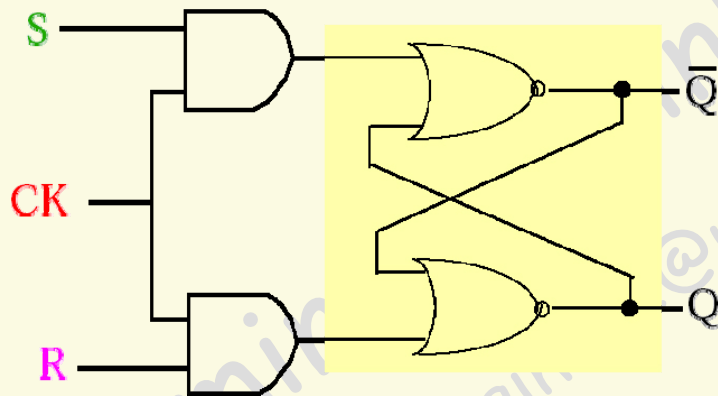
Un chronogramme :

maintien	$CK = 0, S = X, R = X$
maintien	$CK = 1, S = 0, R = 0$
set	$CK = 1, S = 1, R = 0$
reset	$CK = 1, S = 0, R = 1$
NON Autorisé	$CK = 1, S = 1, R = 1$



Bascule RS synchrone

Du point de vue des transistors :



Combien de transistors ?

☐ 10

☒ 12

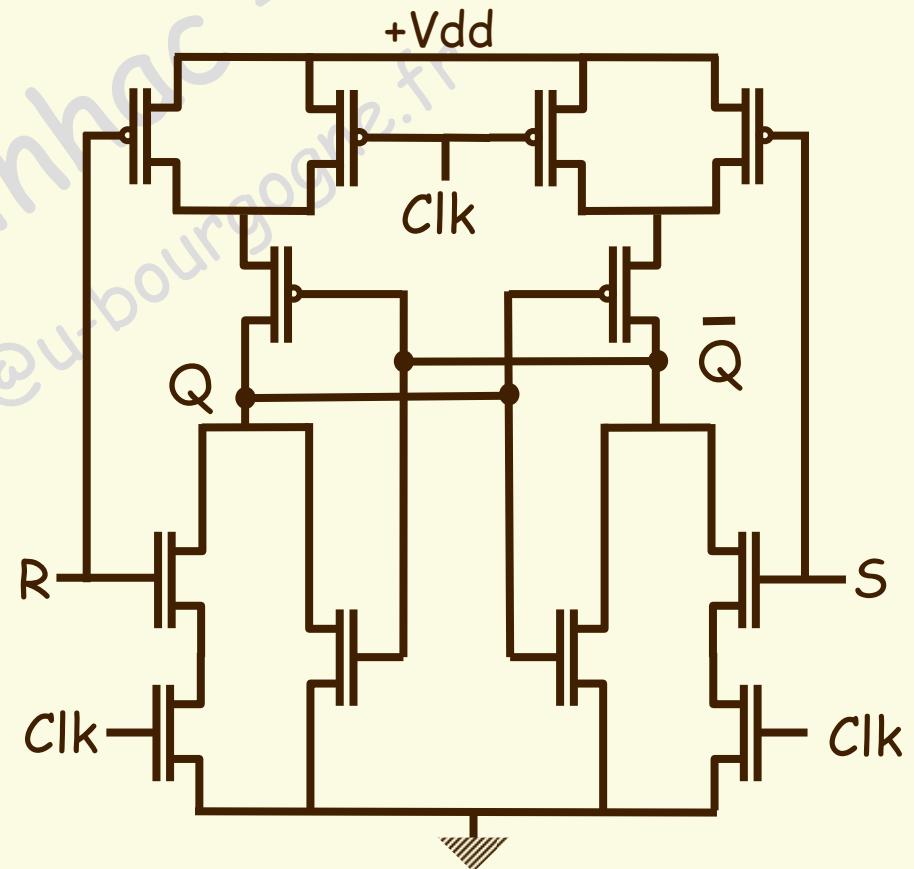
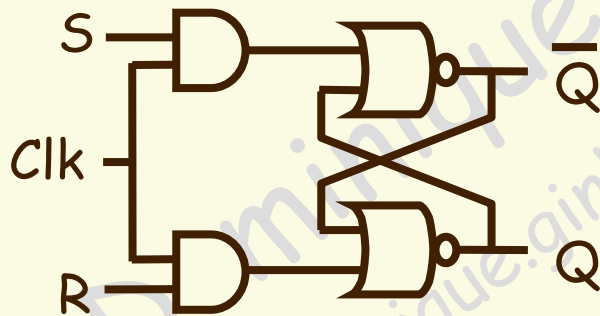
☐ 14

☐ 16

☐ 18

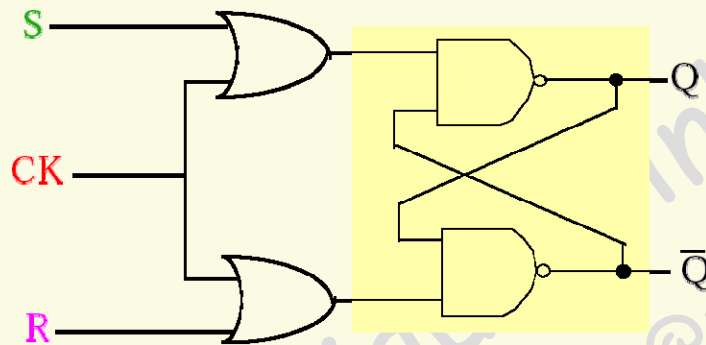
Bascule RS synchrone

Du point de vue des transistors :



Bascule RS synchrone

D'autres solutions :-)



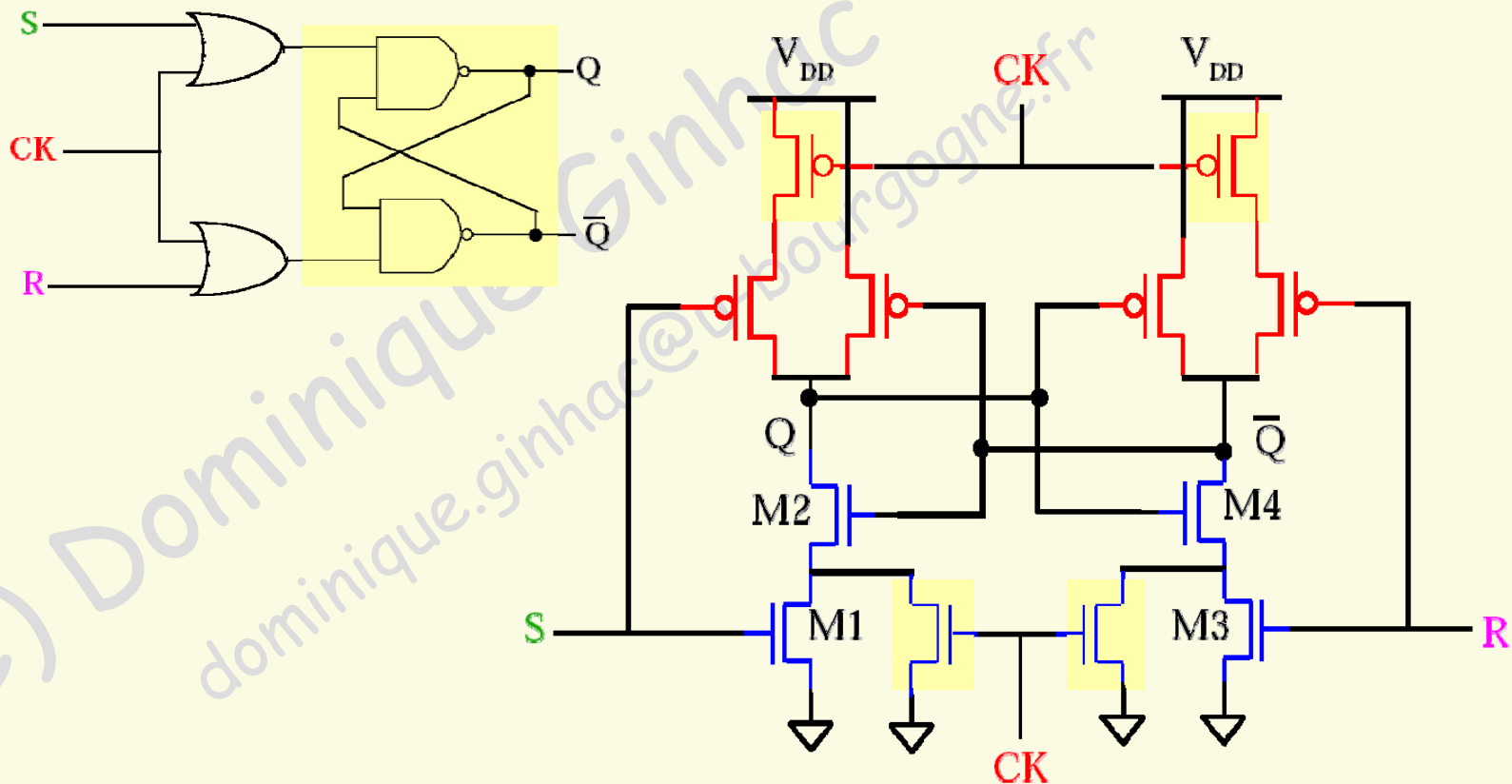
Version **complémentaire** :
(OR suivies de NAND)

Bascule active à **l'état bas**

CK	S	R	Q_{n+1}	\bar{Q}_{n+1}	Operation
1	X	X	Q_n	\bar{Q}_n	maintien
0	0	0	1	1	NON Autorisé
0	1	0	0	1	reset
0	0	1	1	0	set
0	1	1	Q_n	\bar{Q}_n	maintien

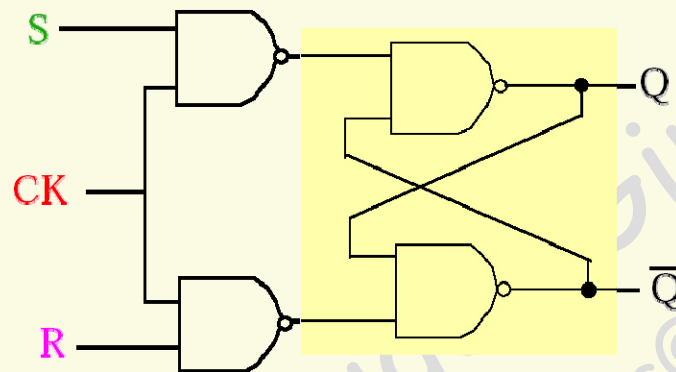
Bascule RS synchrone

Du point de vue des transistors :



Bascule RS synchrone

D'autres solutions (2)



Version tout NAND

Bascule active à l'état haut

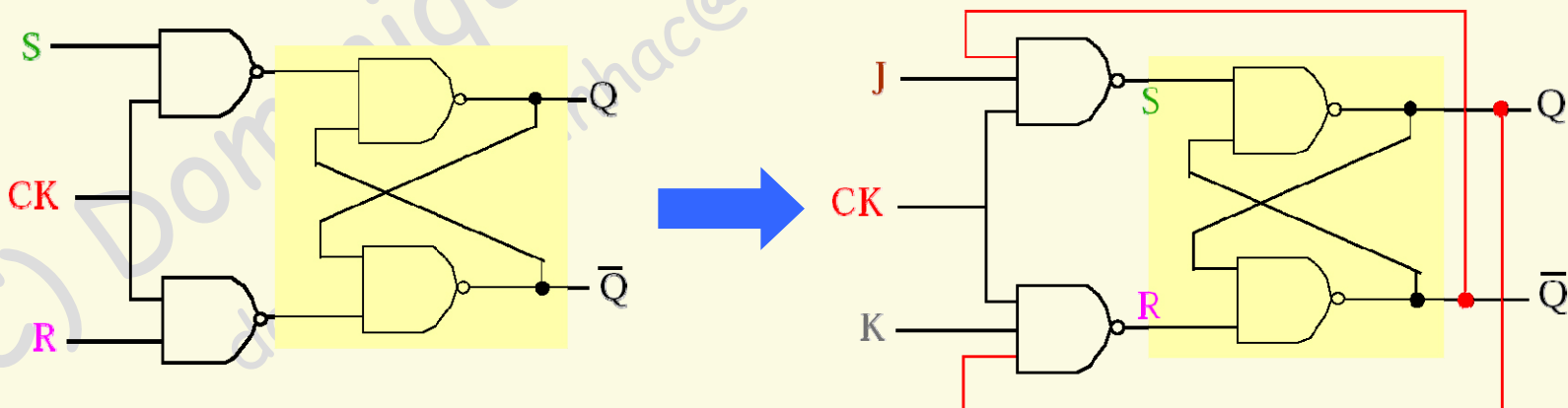
Plus de transistors

CK	S	R	Q_{n+1}	\bar{Q}_{n+1}	Operation
0	X	X	Q_n	\bar{Q}_n	maintien
1	0	0	Q_n	\bar{Q}_n	maintien
1	1	0	1	0	set
1	0	1	0	1	reset
1	1	1	0	0	NON Autorisé

Bascule JK

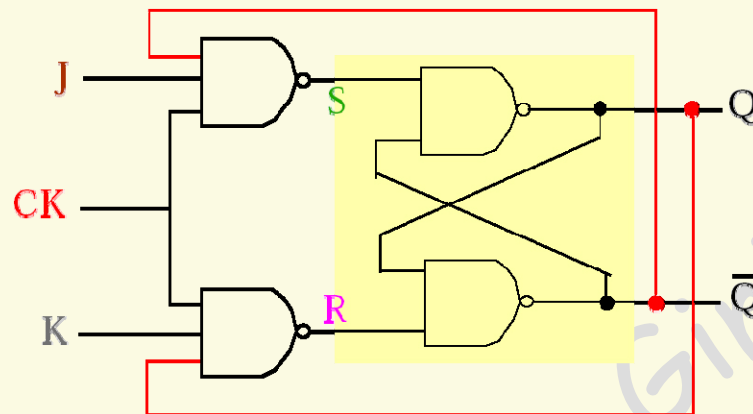
L'inconvénient majeur des bascules RS est son état interdit ($S=R=1$ ou $S=R=0$)

Ce problème peut être facilement surmonté en transformant la bascule RS en bascule JK



Exemple de bascule JK réalisée à partir d'une bascule RS tout NAND

Fonctionnement des bascules JK

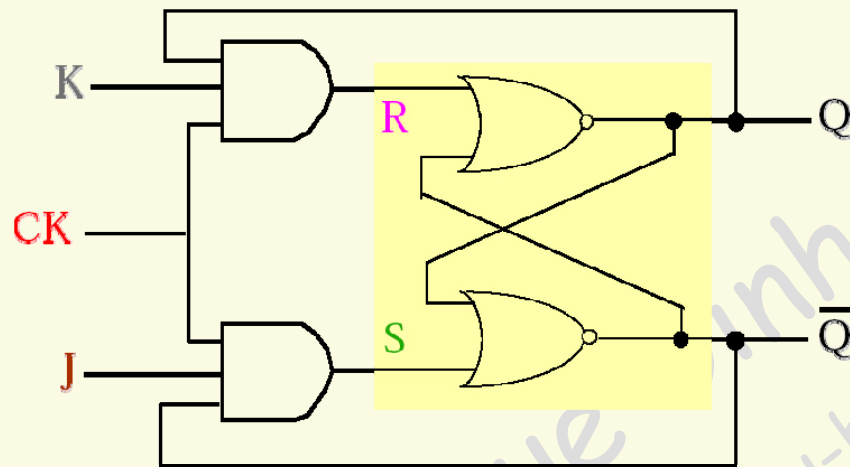


$CK = 0 \Rightarrow$ maintien

$CK = 1 \Rightarrow$ bascule active

J	K	Q_n	\bar{Q}_n	S	R	Q_{n+1}	\bar{Q}_{n+1}	Operation
0	0	0	1	1	1	0	1	maintien
0	0	1	0	1	1	1	0	maintien
0	1	0	1	1	1	0	1	reset
0	1	1	0	1	0	0	1	reset
1	0	0	1	0	1	1	0	set
1	0	1	0	1	1	1	0	set
1	1	0	1	0	1	1	0	toggle
1	1	1	0	1	0	0	1	toggle

Une bascule JK plus simple



Version CMOS classique



Combien de transistors ?

☐ 12

☐ 14

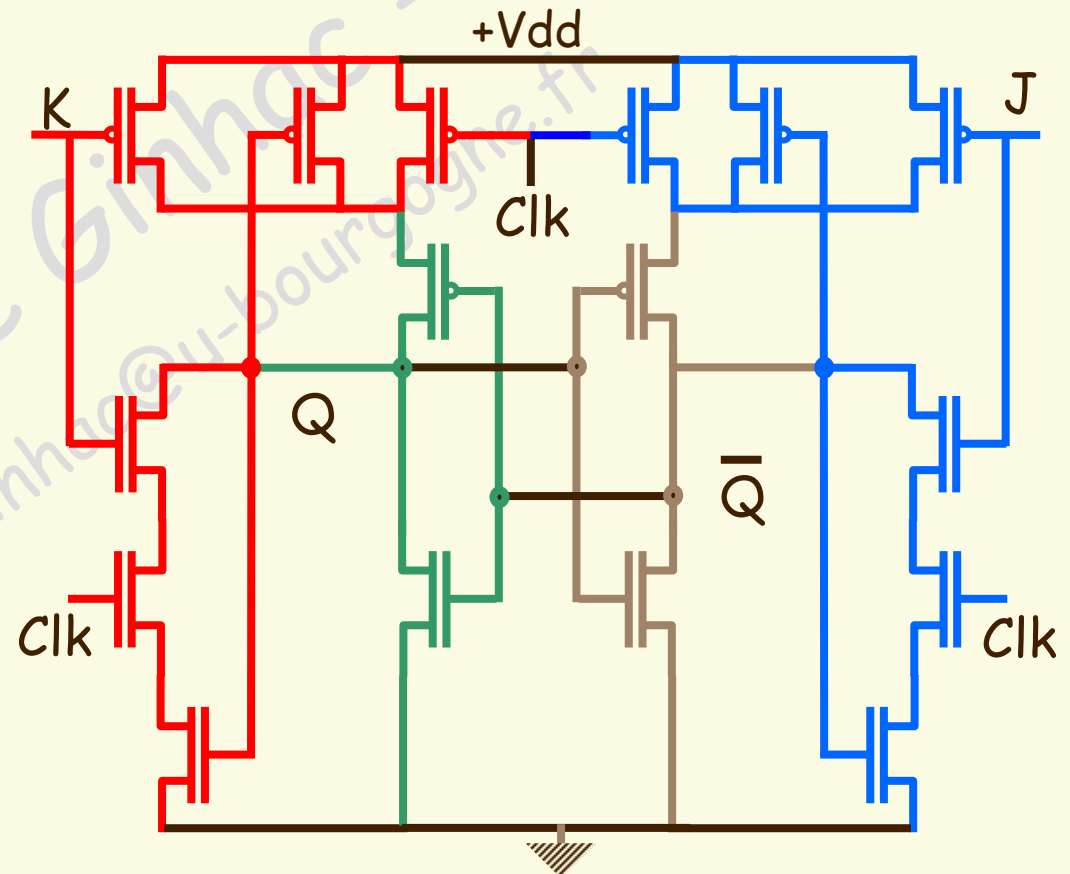
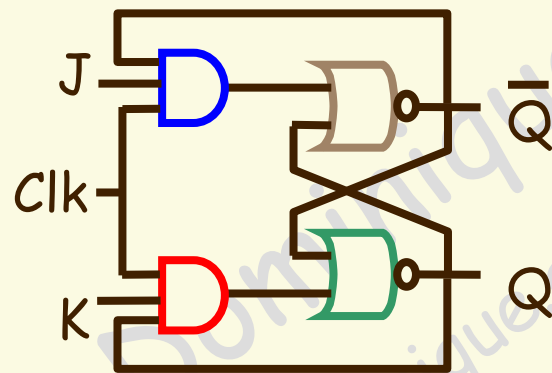
☒ 16

☐ 18

☐ 20

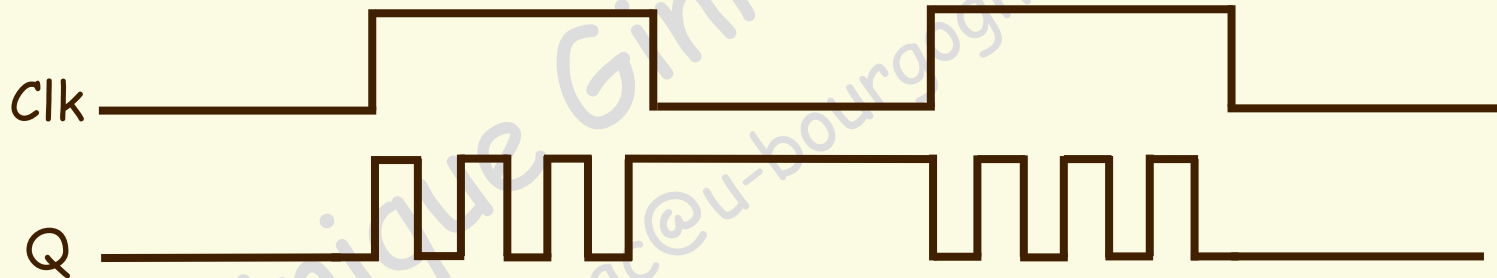
Bascule JK

En technologie CMOS :



Bascule JK

Un **inconvenient majeur** de la bascule JK est sa possibilité de **faire osciller les sorties** si $J=K=1$

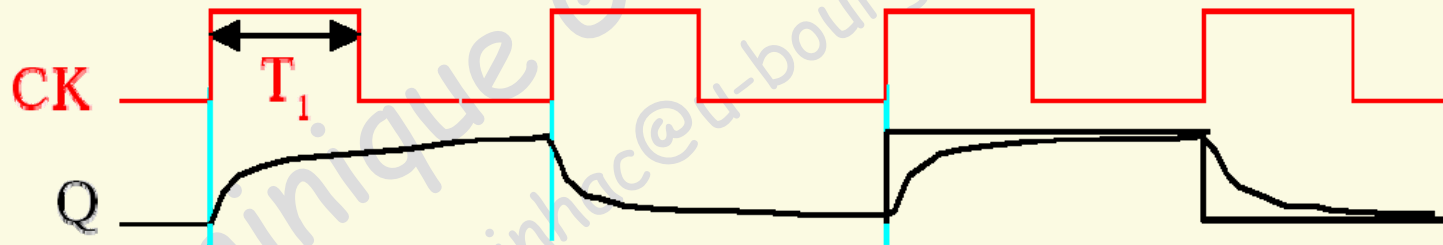


2 solutions :

- ✓ Utiliser une **horloge de fréquence importante**,
- ✓ Utiliser une **bascule JK Maître Esclave**

Horloge de fréquence importante

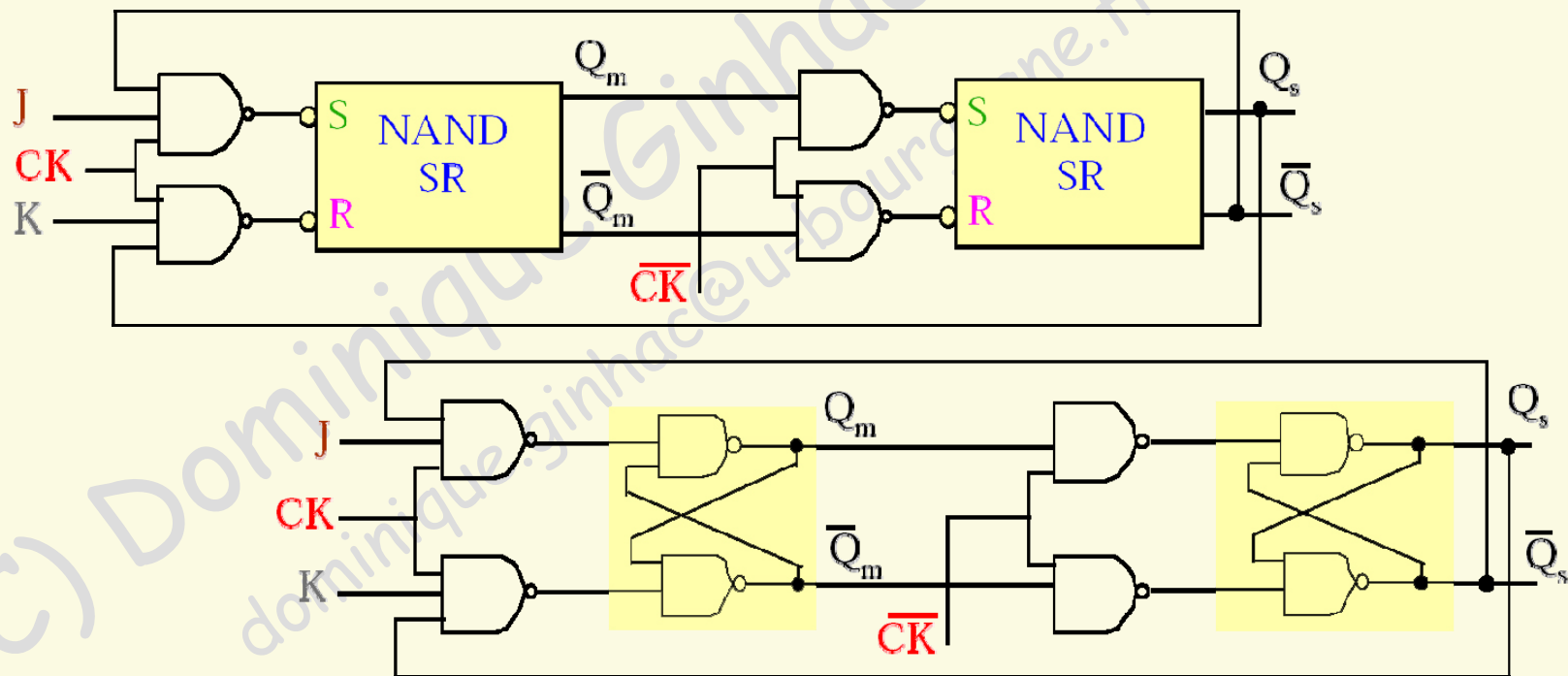
Si le **temps de réponse** de la bascule JK est **supérieur** à la **période de l'horloge**, alors on assiste à un **seul changement** par période d'horloge



Cette solution est **difficile à mettre en œuvre** puisque la période de l'horloge doit approximativement correspondre avec le temps de propagation de la porte

Bascule JK maître esclave

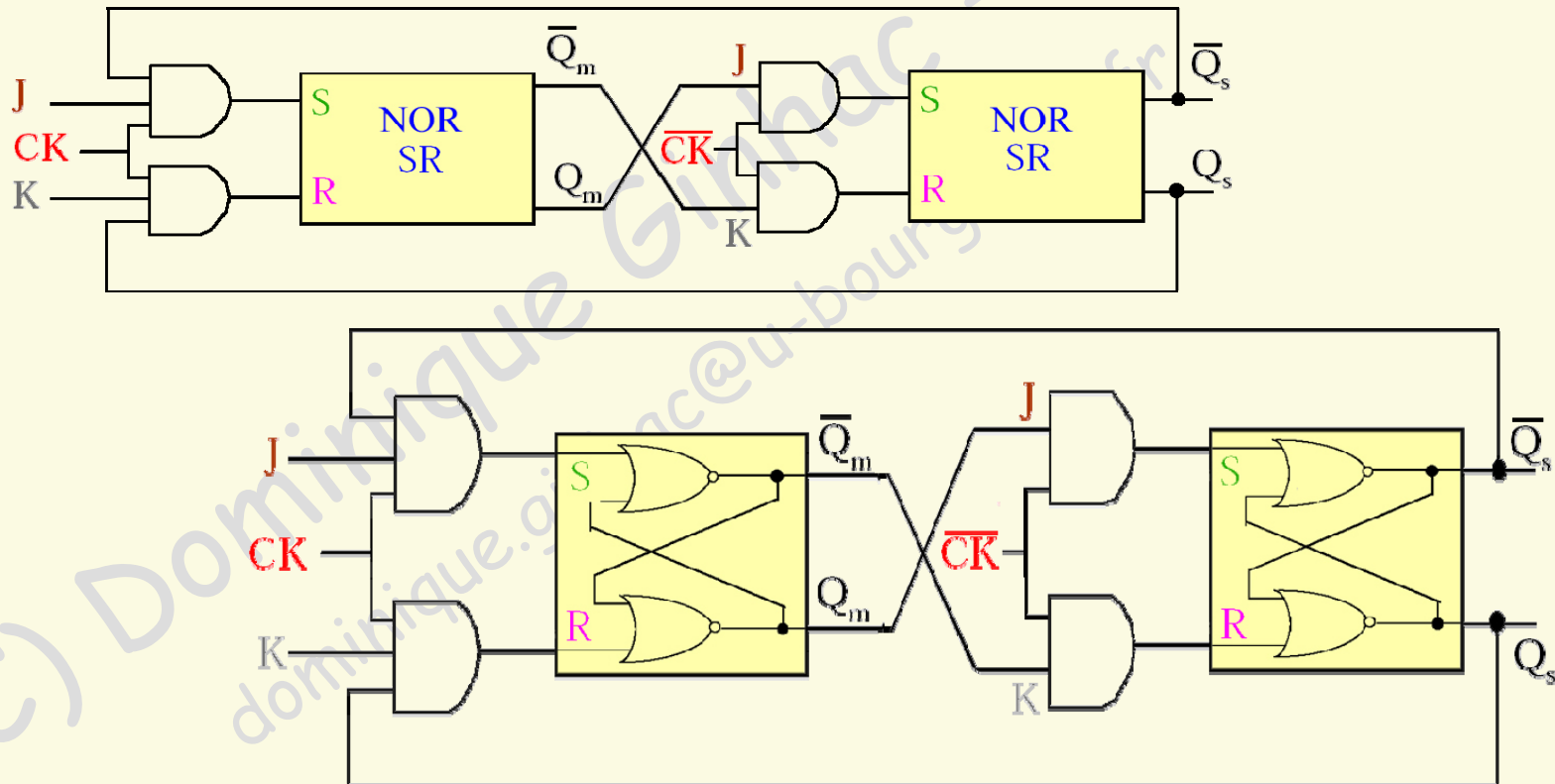
Le principe consiste à enchaîner 2 bascules JK d'horloges complémentaires



Basculer JK Maître Esclave réalisée à partir de bascules JK tout NAND

Bascule JK maître esclave

Une deuxième variante « plus économe »



Bascule JK Maître Esclave réalisée à partir de bascules JK NOR

Bascule JK maître esclave

Principe de fonctionnement :

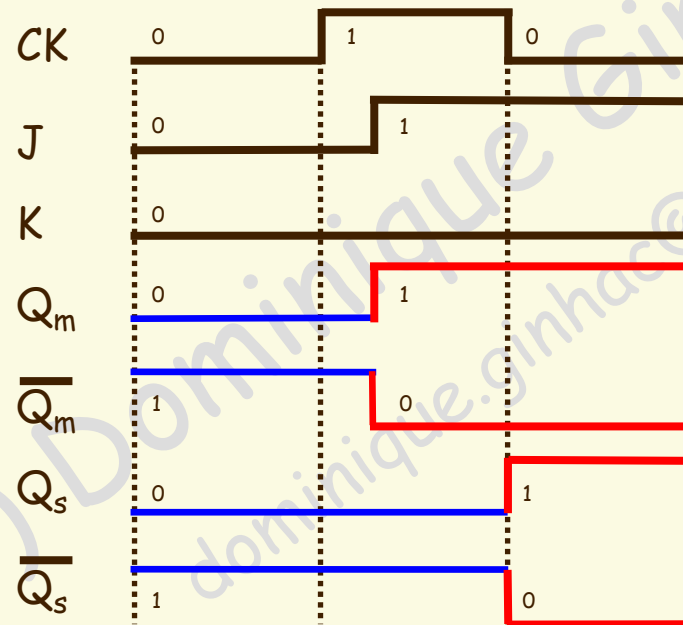
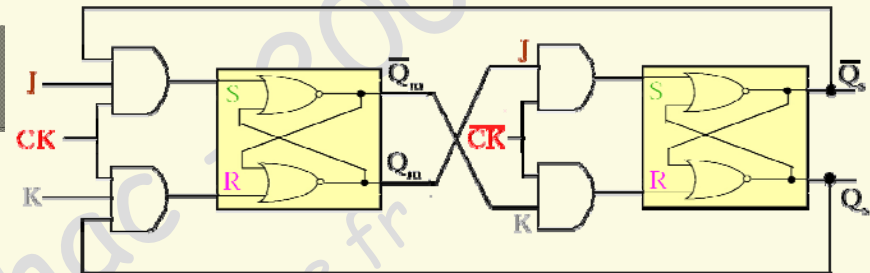
A tout moment, **une et une seule** des bascules JK « **fonctionne** », l'autre est en phase de **maintien** :

- ✓ **CK = 1** : Bascule 1 (« Master ») active et Bascule 2 (« Slave ») en maintien
- ✓ **CK = 0** : Bascule 1 (« Master ») en maintien et Bascule 2 (« Slave ») active

Les 2 étages sont donc **découplés** et empêchent ainsi une **propagation directe** des entrées sur les sorties

Bascule JK maître esclave

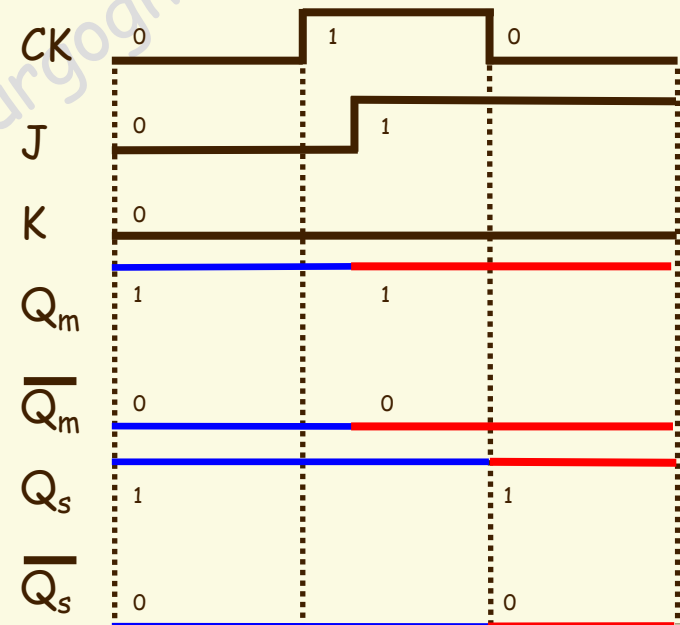
Mise à 1 : $J = 1, K = 0$



$A \uparrow = 0, Q_m = Q_s = 0$

JK 1
active

JK 2
active



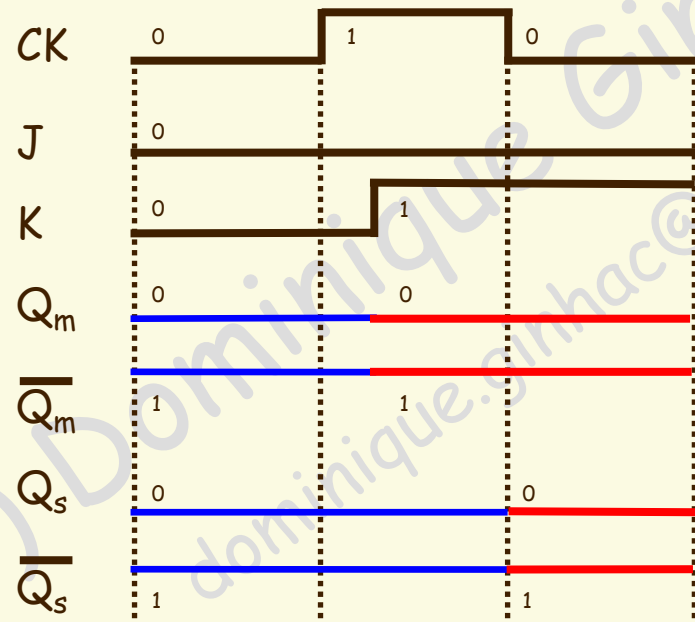
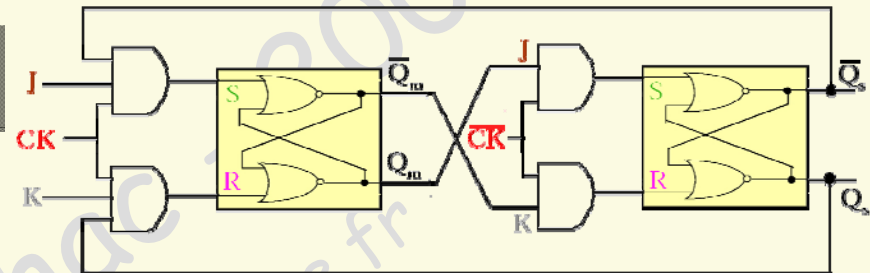
$A \uparrow = 0, Q_m = Q_s = 1$

JK 1
active

JK 2
active

Bascule JK maître esclave

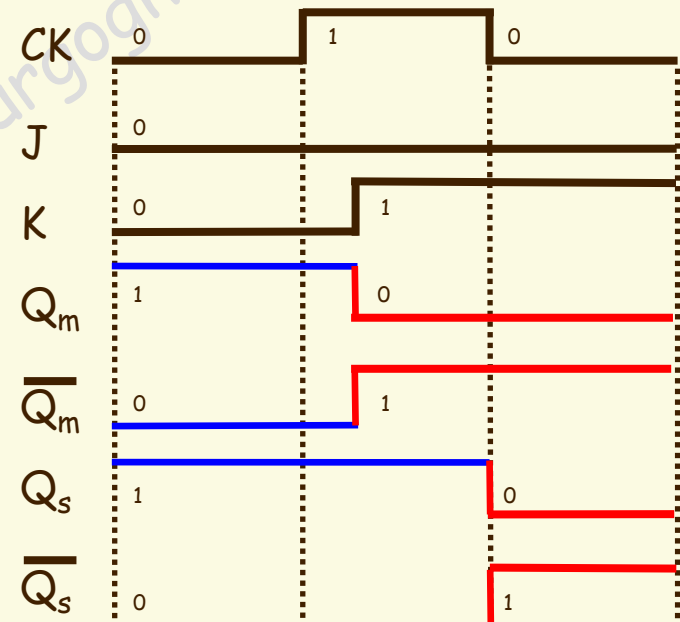
Mise à 0 : $J = 0$, $K = 1$



$A \uparrow = 0, Q_m = Q_s = 0$

JK 1
active

JK 2
active



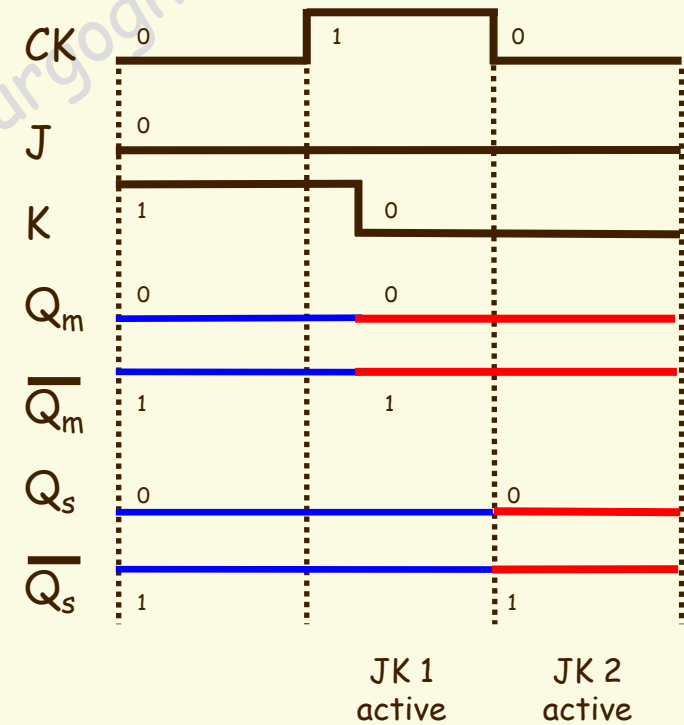
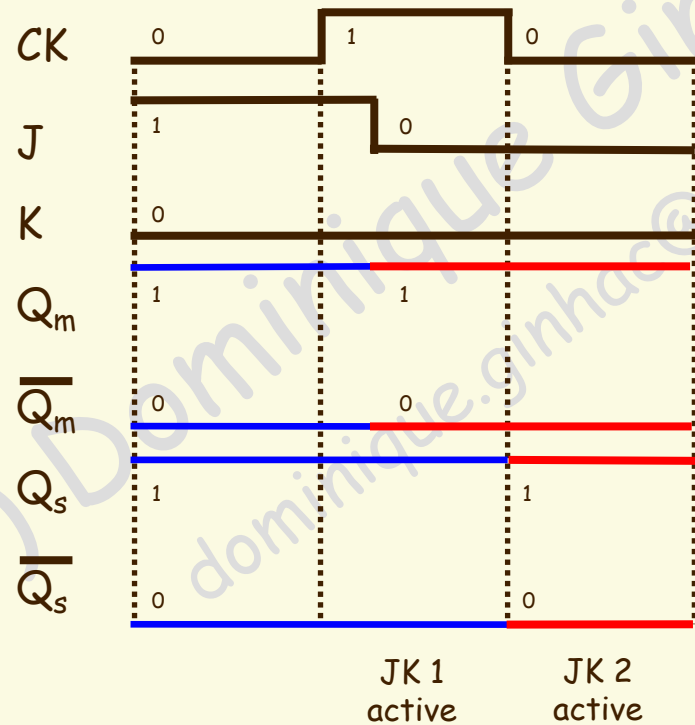
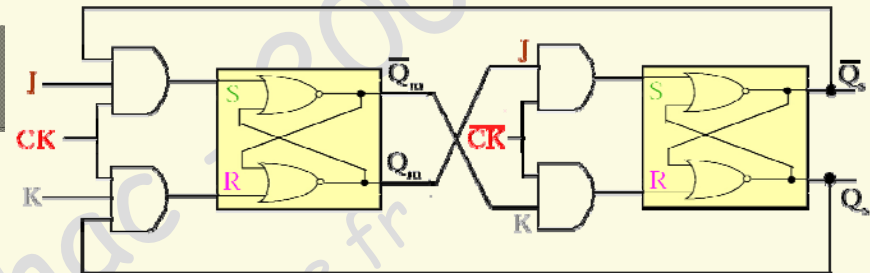
$A \uparrow = 0, Q_m = Q_s = 1$

JK 1
active

JK 2
active

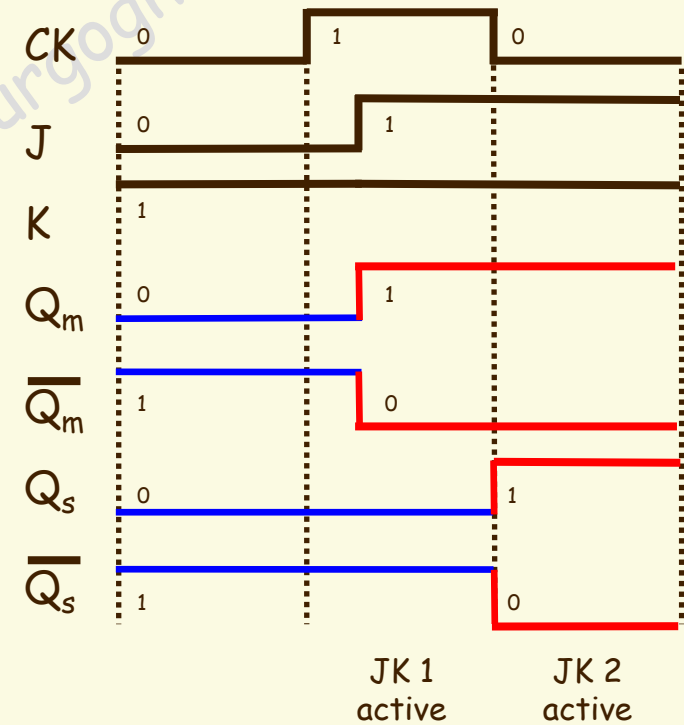
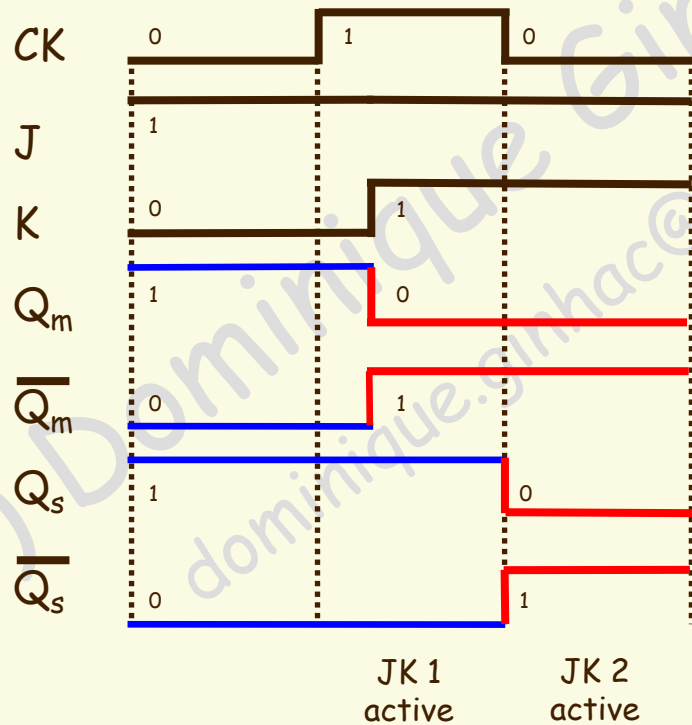
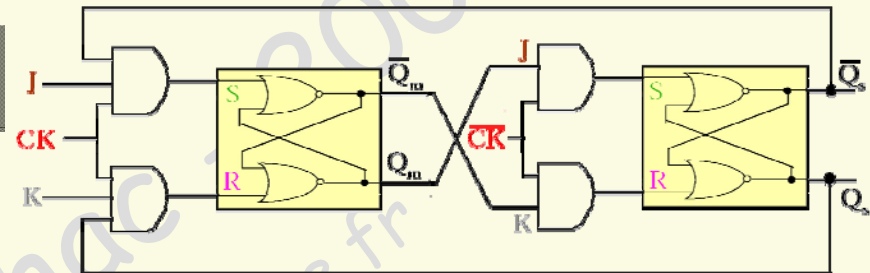
Bascule JK maître esclave

Maintien : $J = 0$, $K = 0$



Bascule JK maître esclave

Toggle : $J = 1$, $K = 1$



Premières conclusions

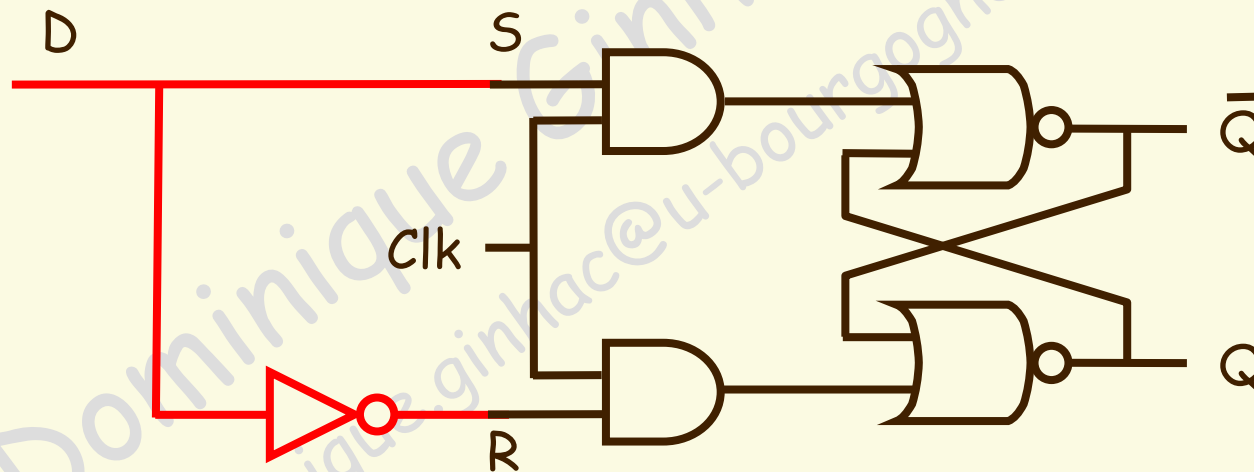
En conception microélectronique, les bascules sont très utilisées

Cependant, des implantations purement CMOS de bascules de type JK maître-esclave sont coûteuses en nombre de transistors

Il existe des optimisations pour réaliser de telles bascules mais il est toutefois nécessaire de prendre certaines précautions

Bascule D

La bascule D est une variante de la bascule RS synchrone dans laquelle on a $R = \overline{S}$



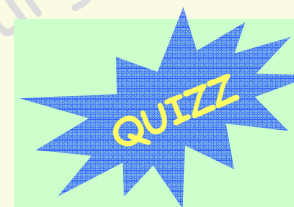
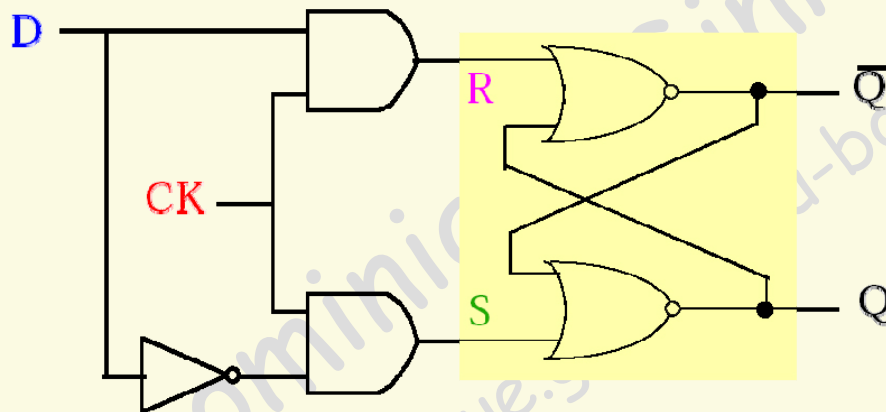
QUIZZ

Comment ça marche ?

Si $CK = 1$: $Q_{n+1} = D$
Si $CK = 0$: $Q_{n+1} = Q_n$

Bascule D

La bascule D est une variante de la bascule RS synchrone dans laquelle on a $R = \bar{S}$



Combien de transistors ?

☐ 10

☐ 12

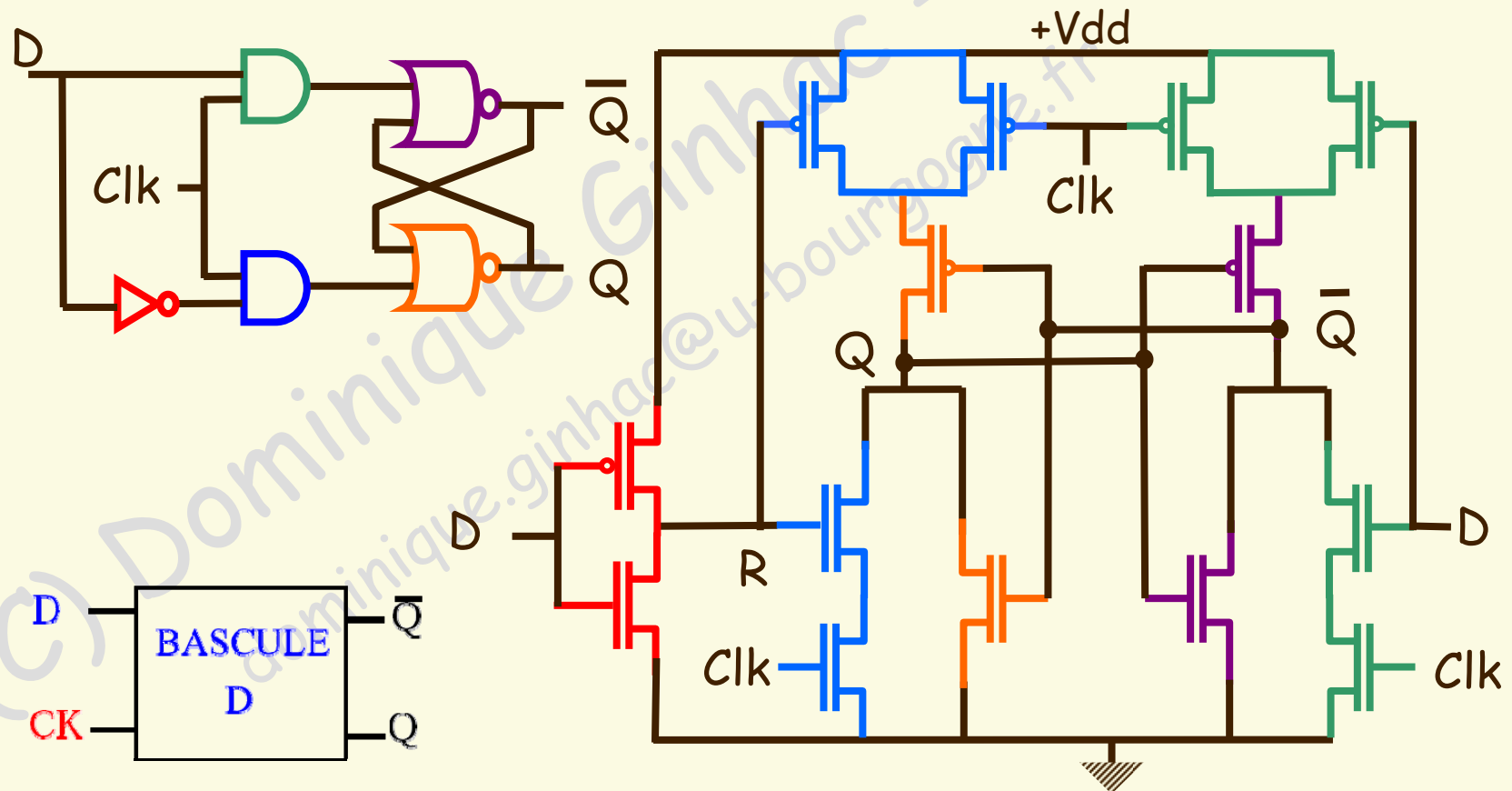
☒ 14

☐ 16

☐ 18

Bascule D

Du point de vue des transistors :

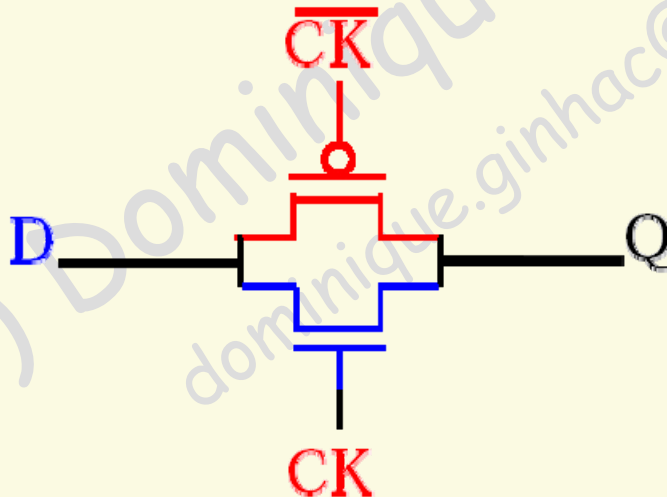


Une première optimisation

La bascule D possède **2 modes** de fonctionnement :

- ✓ Clk = 1 : **propagation de D vers Q**
- ✓ Clk = 0 : **maintien de Q**

Premier cas : Si **CK = 1** : $Q_{n+1} = D$



Utilisation d'une **porte de transmission** pilotée par :

- ✓ CK sur le transistor Nmos
- ✓ \overline{CK} sur le transistor Pmos

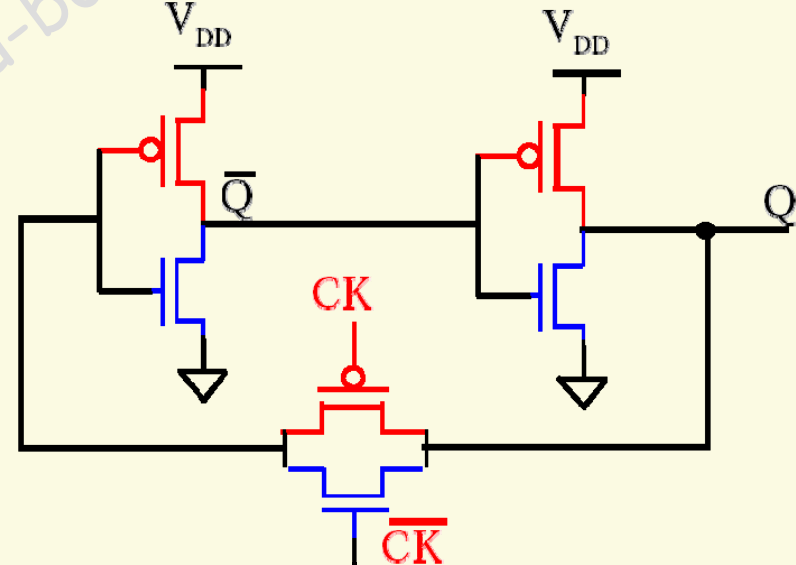
Une première optimisation

La bascule D possède 2 modes de fonctionnement :

- ✓ Clk = 1 : propagation de D vers Q
- ✓ Clk = 0 : maintien de Q

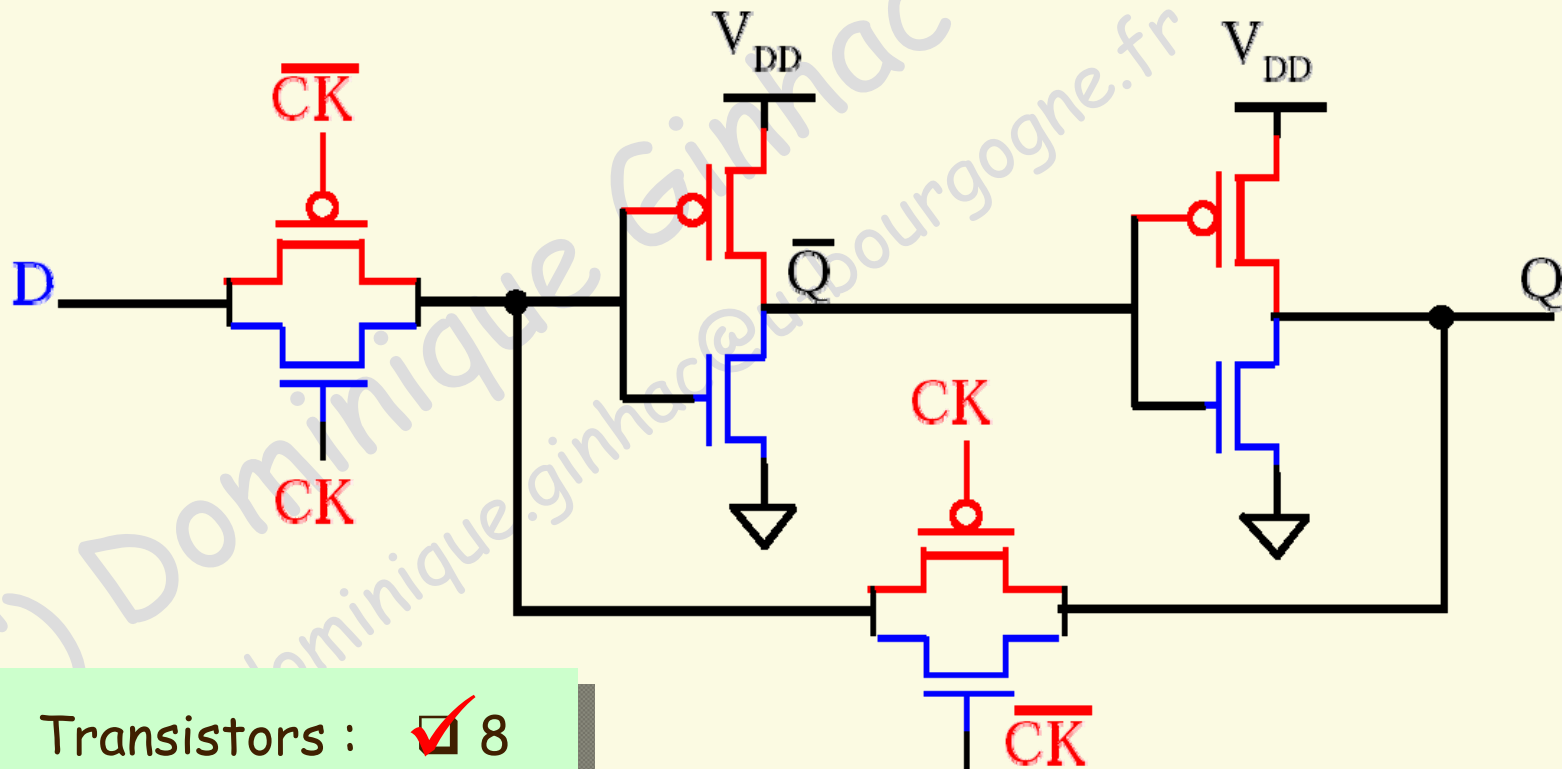
Deuxième cas : Si $CK = 0$: $Q_{n+1} = Q_n$

Maintien de la valeur de Q par utilisation d'un système bistable commandé par une porte de transmission



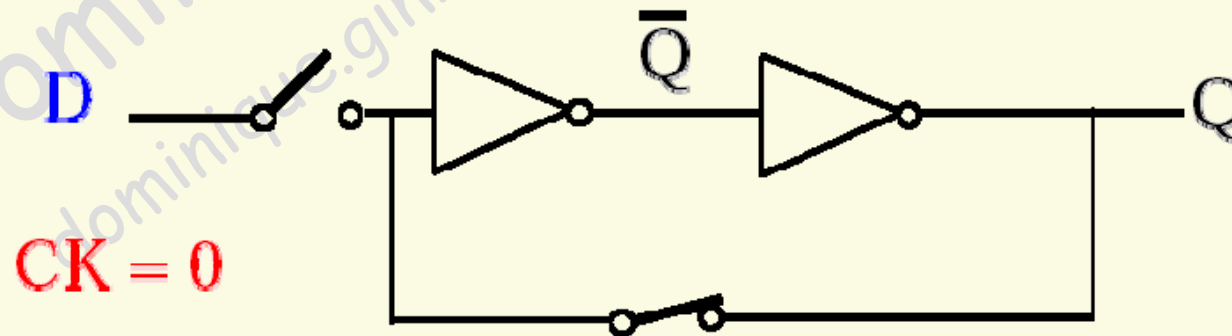
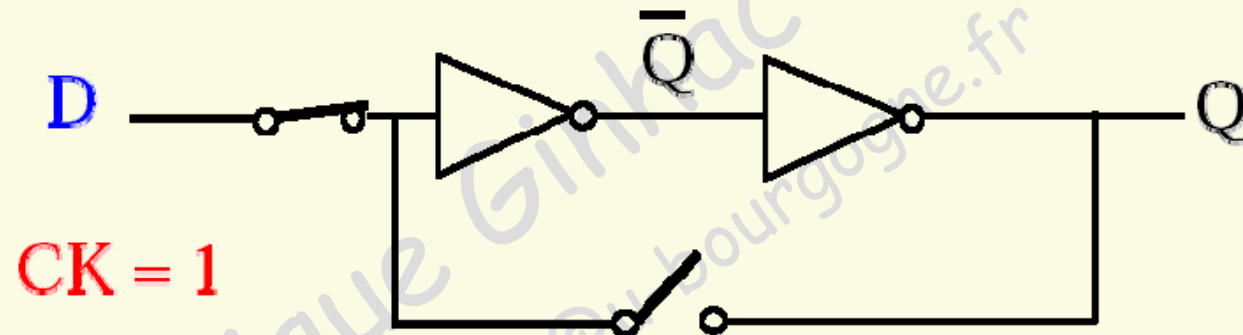
Une première optimisation

Au final :



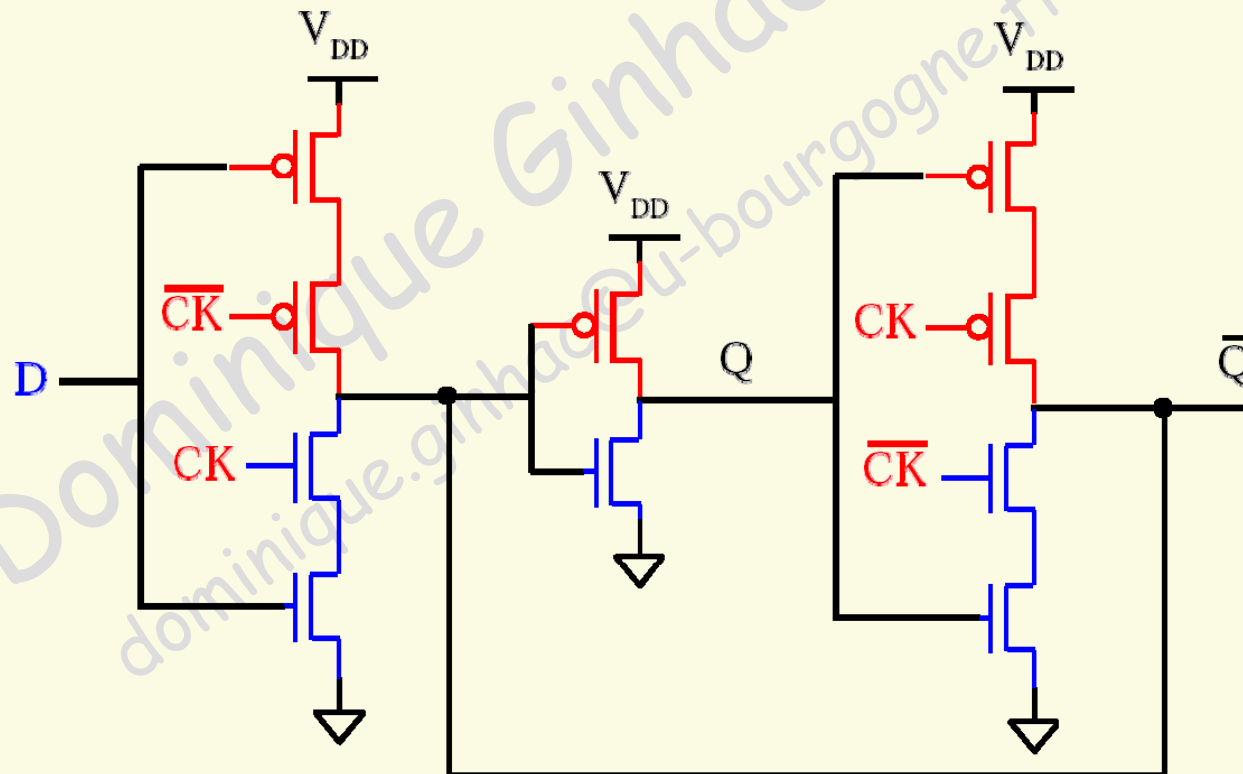
Une première optimisation

Au final :



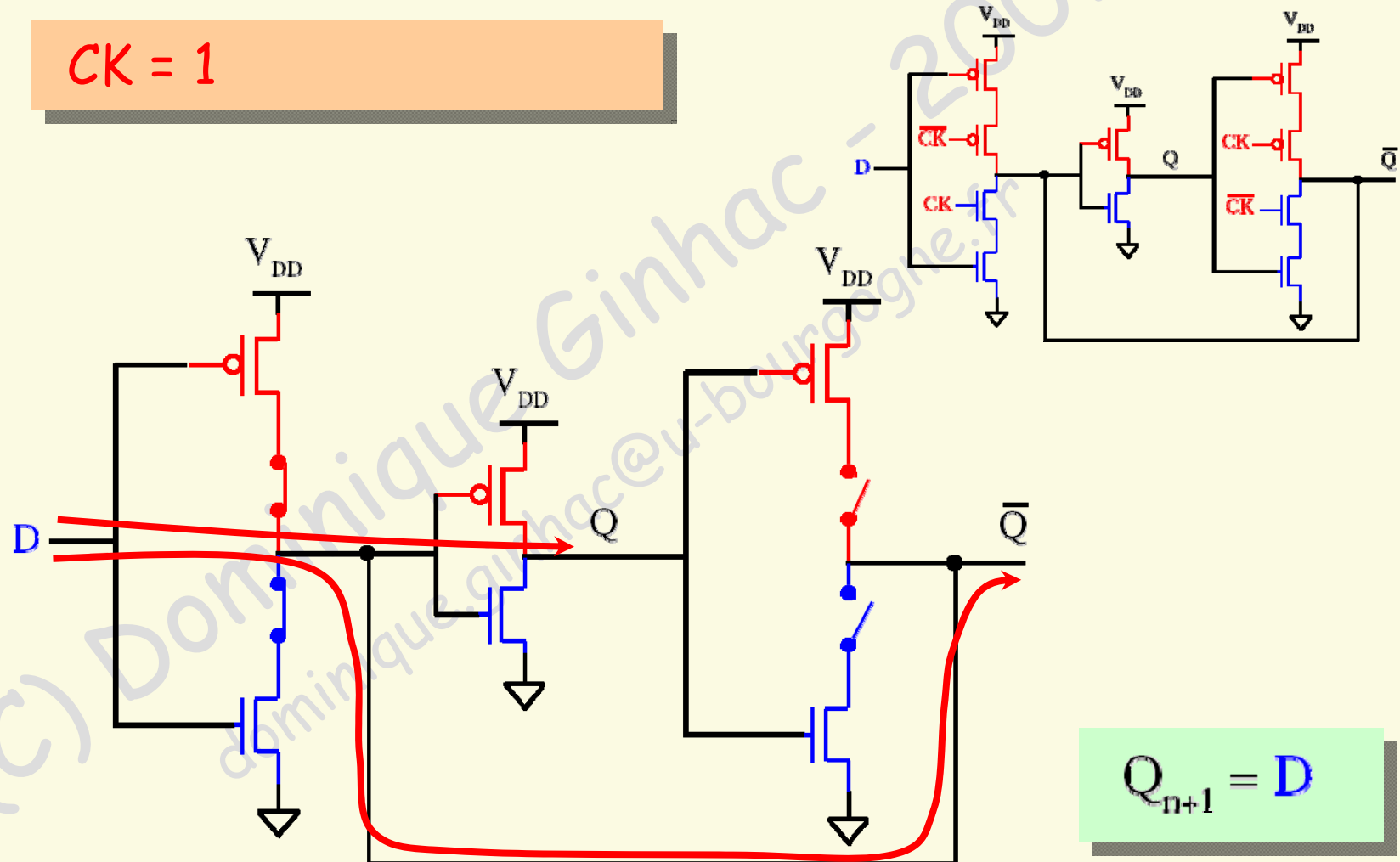
Une deuxième optimisation

Il est possible de réaliser une bascule D en utilisant des **inverseurs 3 états**



Une deuxième optimisation

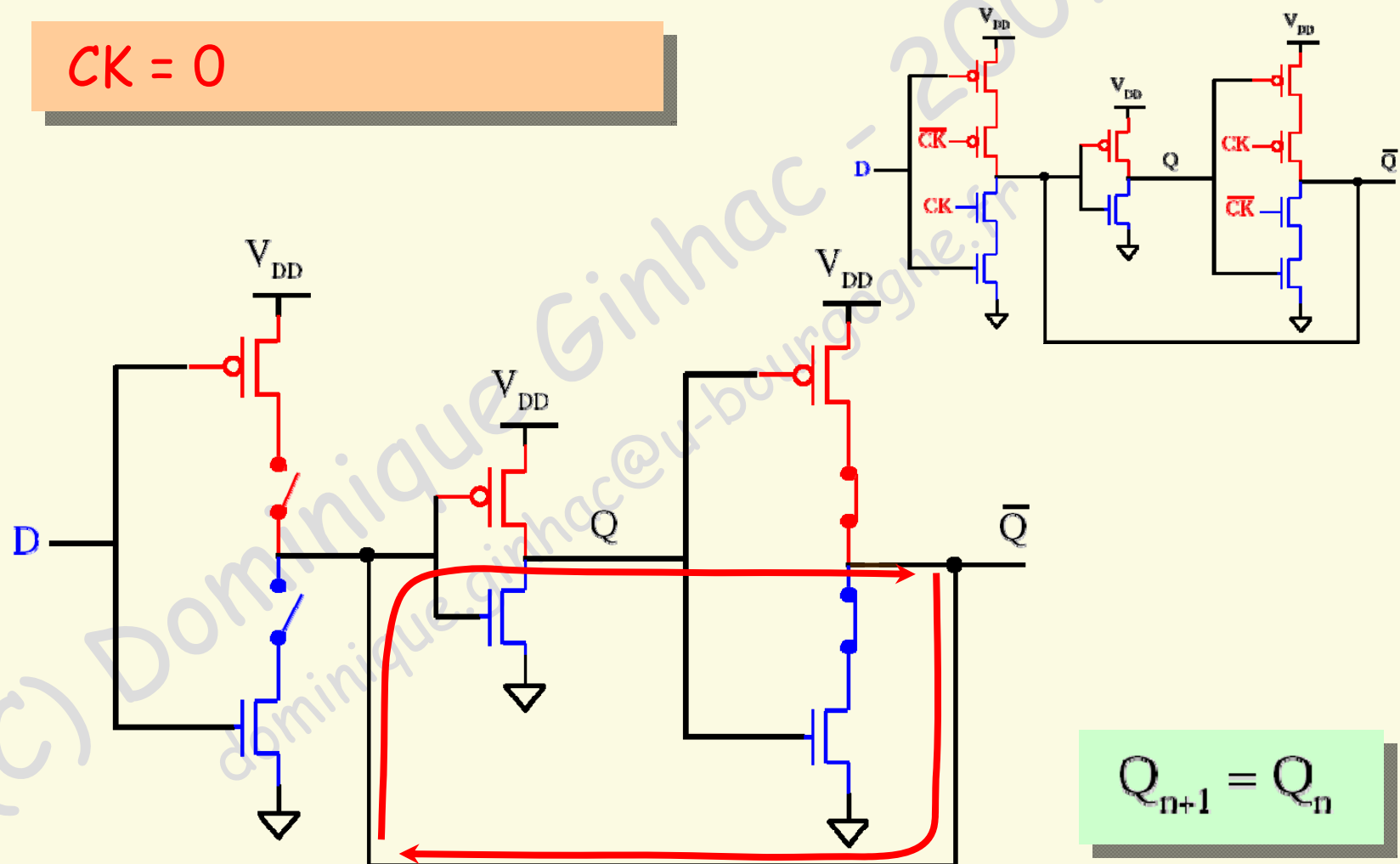
$CK = 1$



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

Une deuxième optimisation

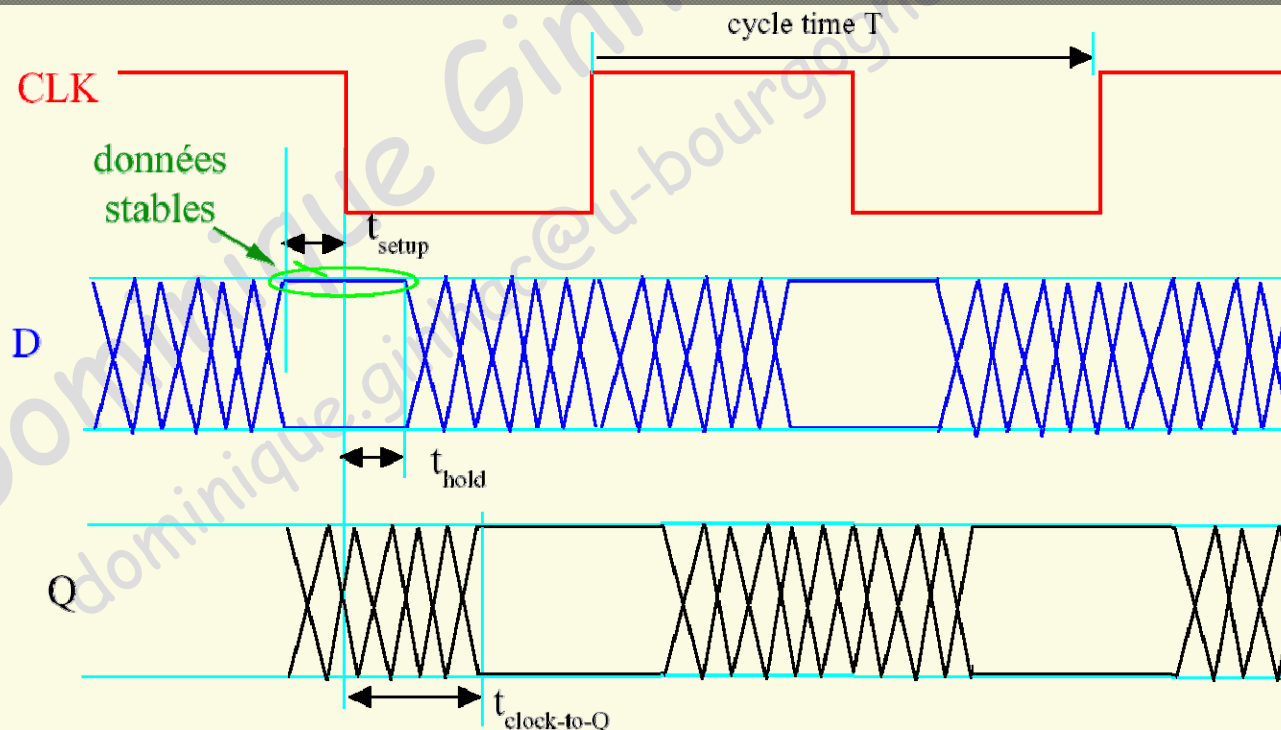
$CK = 0$



$$Q_{n+1} = Q_n$$

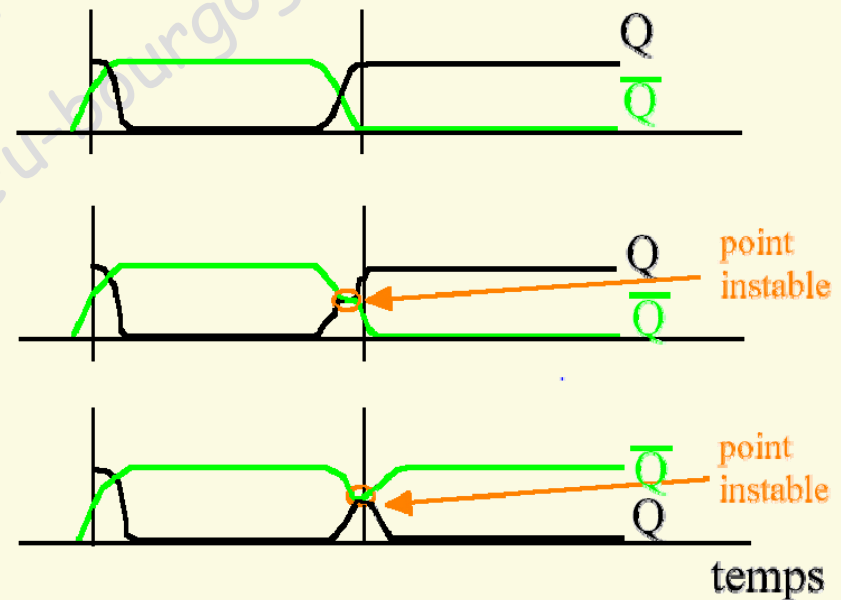
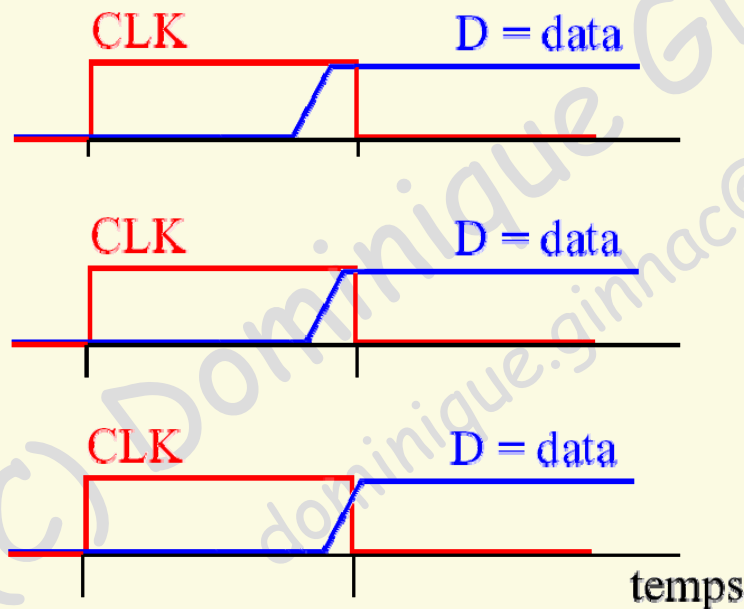
Problèmes potentiels

Attention : il est nécessaire de **garantir** la **stabilité des données** lors de la **commutation** de l'horloge de 1 à 0



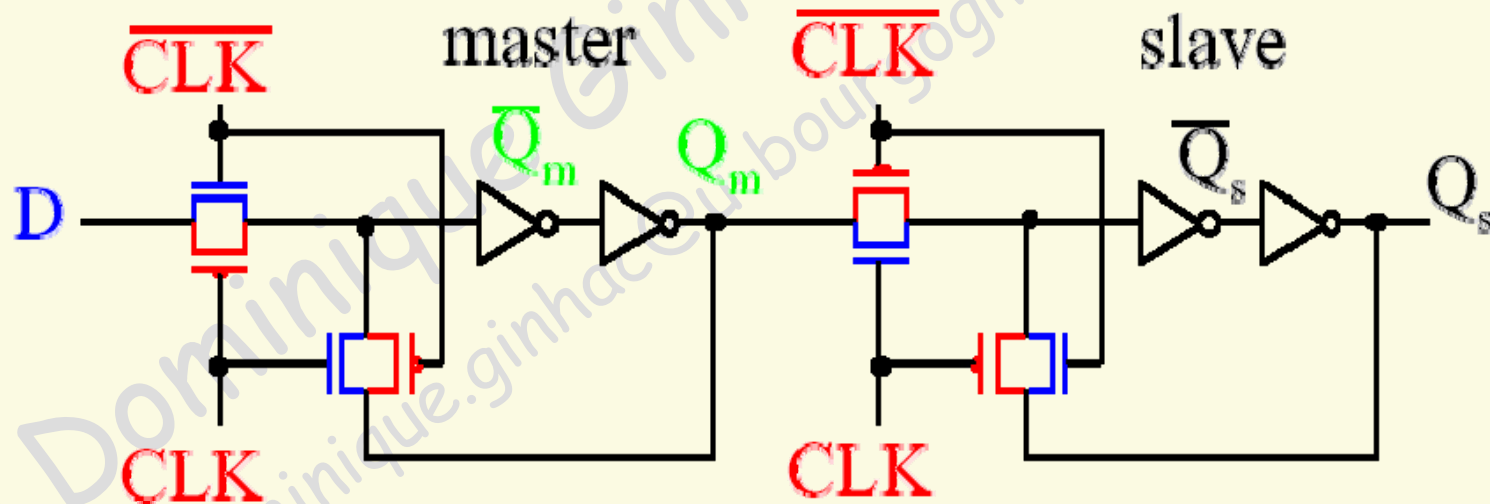
Problèmes potentiels

Si la **stabilité** de D n'est pas **garantie**, la sortie Q peut avoir une **valeur erronée** due au caractère bistable des 2 inverseurs



Une troisième optimisation

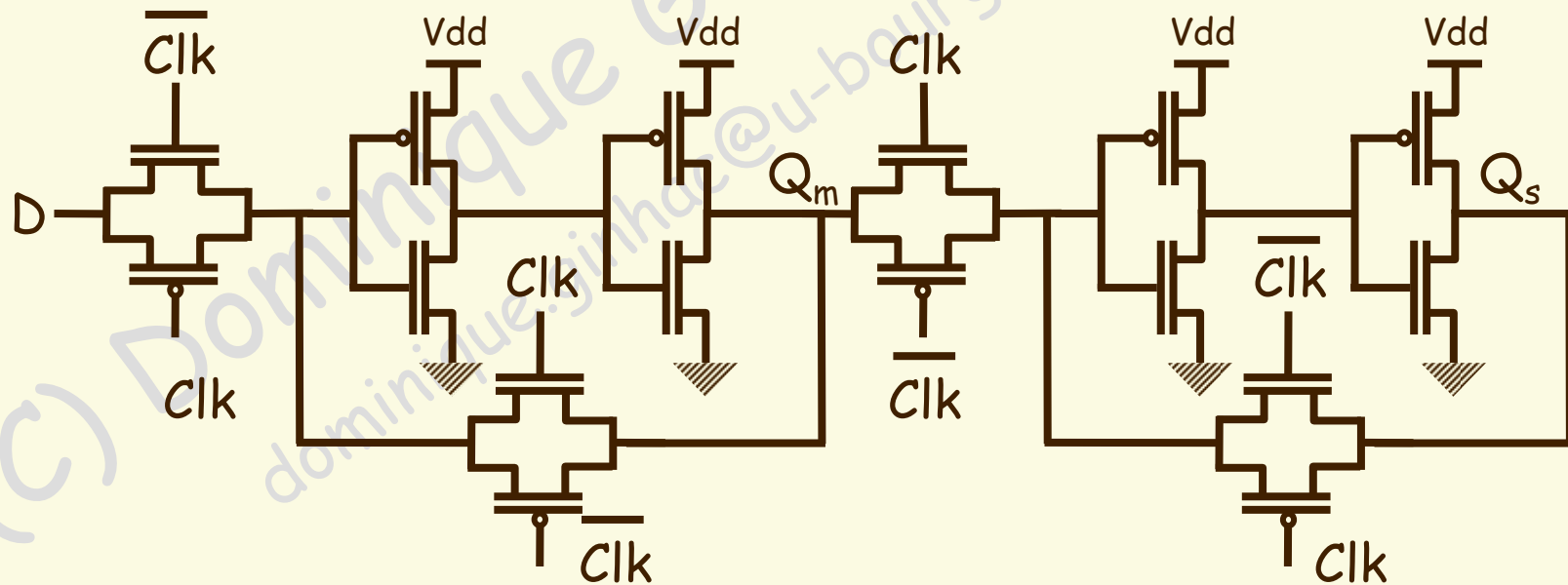
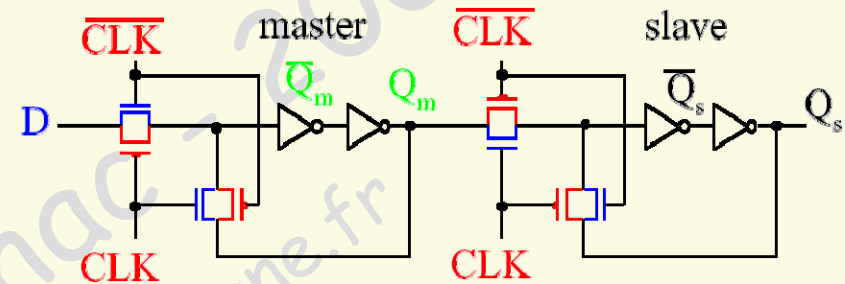
Réalisation d'une bascule maître esclave à partir de 2 bascules D d'horloges complémentaires



On obtient une bascule à commande par front montant

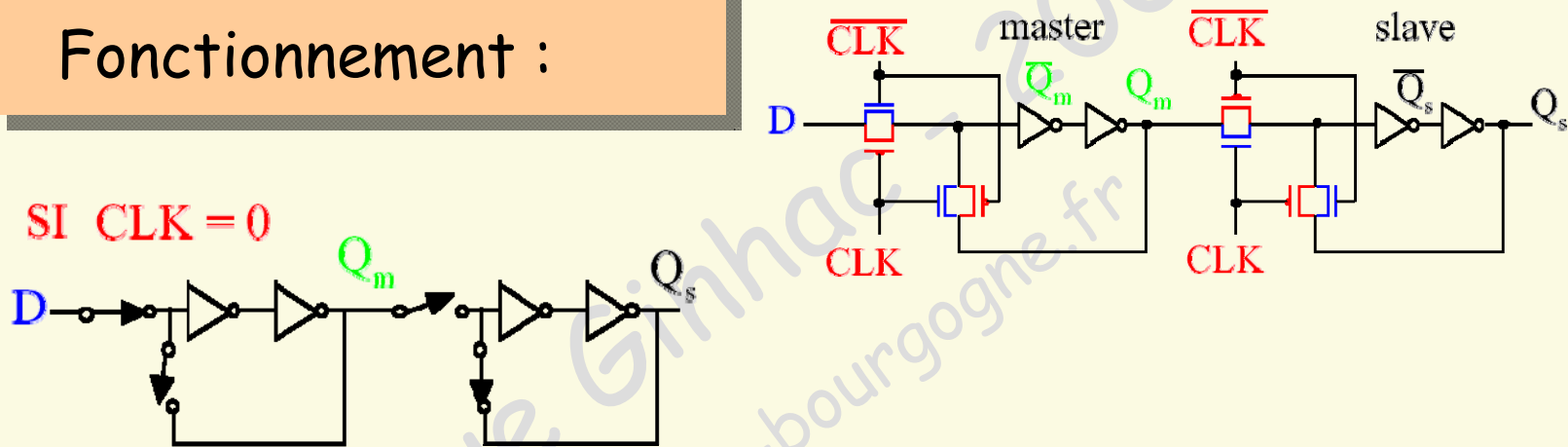
Une troisième optimisation

Au niveau transistor :



Une troisième optimisation

Fonctionnement :

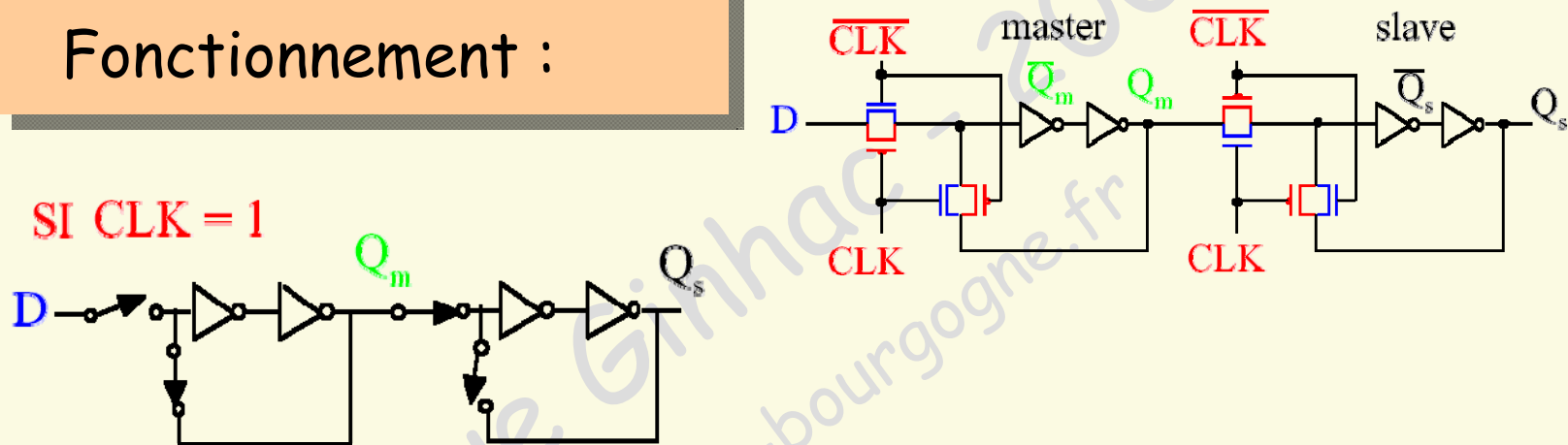


- SI $CLK = 0$
- $CLK = 0$:
- ✓ Propagation de D sur Q_m pour la première bascule
 - ✓ Maintien de Q_s (c'est-à-dire la valeur précédente de D) sur la deuxième bascule

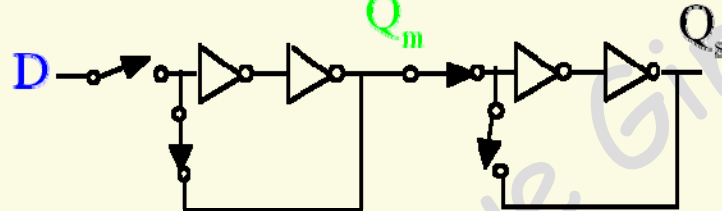
Q_s est insensible aux variations de D pour $CLK = 0$

Une troisième optimisation

Fonctionnement :



SI $CLK = 1$

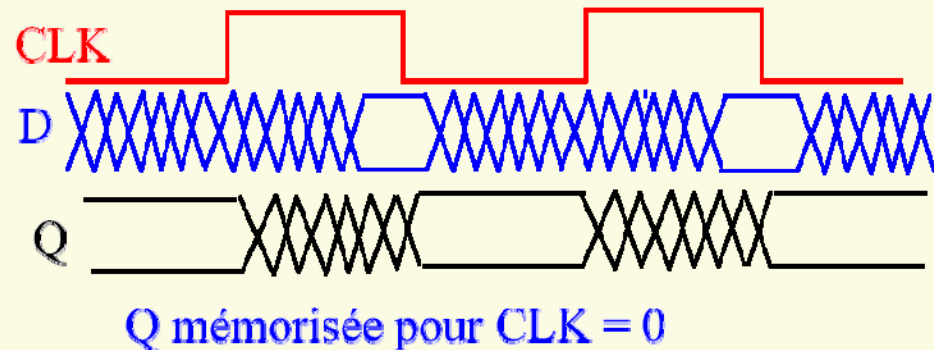
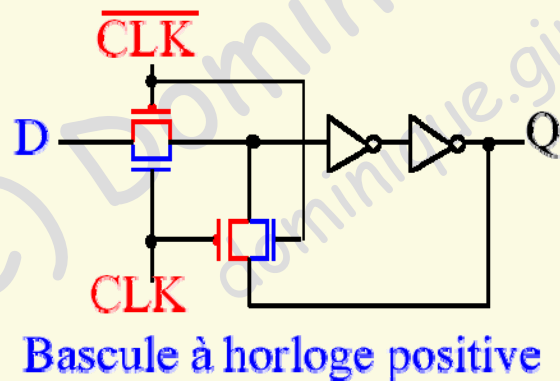
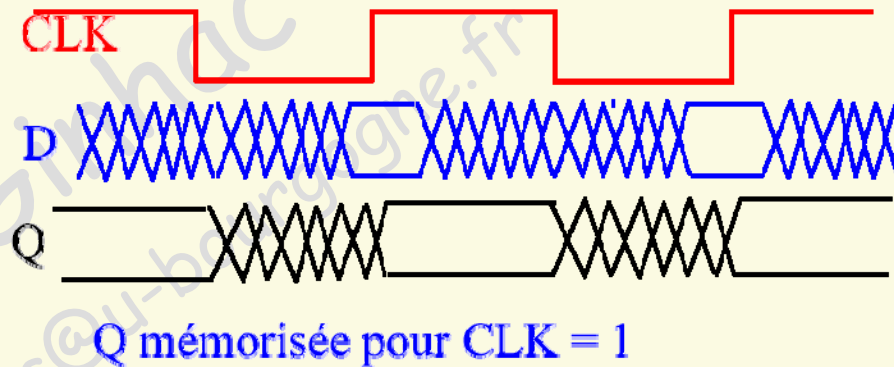
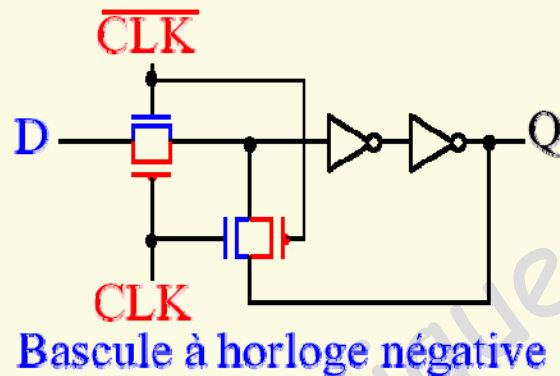


- $CLK = 1$:
- ✓ **Maintien de Q_m** (c'est-à-dire la valeur de D lorsque CLK était égale à 0) sur la première bascule
 - ✓ **Propagation de Q_m** (valeur de D lorsque CLK était égale à 0) **sur Q_s** sur la deuxième bascule

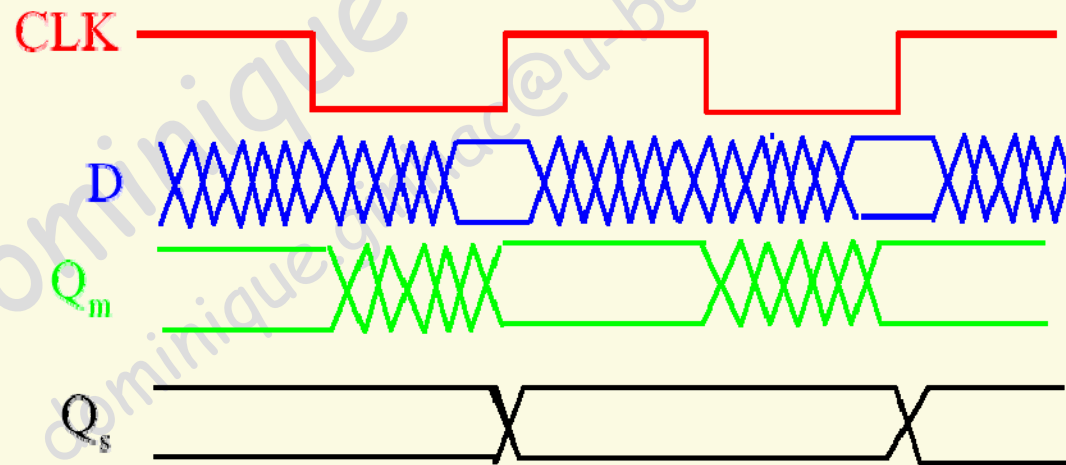
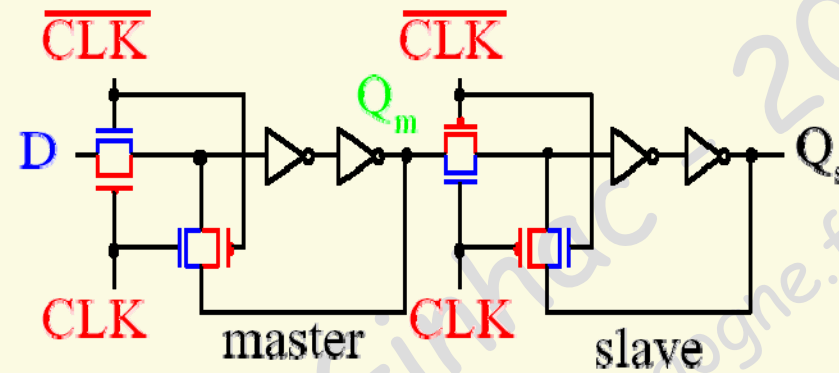
Q_m et Q_s sont **insensibles** aux variations de D pour $CLK = 1$

Deux Types de bascules

Décomposition en 2 bascules complémentaires



Au final



Q_s mémorisé pour $\text{CLK} = 1$ et disponible quel que soit CLK

