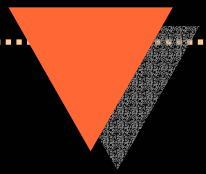


Fonction mémoire / Fonction comptage

Brique de base *stable*, Caractérisation des fonctions mémoires

RS, RST (Clear/Preset), D latch

RS M/S, DataLockOut, DT-ME, D edge, T, JK

Tables de transitions

Machines séquentielles asynchrones synchronisées (synthèse)

Système séquentiel : briques de base

Système de vecteur d'entrée E et de sortie S
états du vecteur d'entrée décomposés en ensembles Ei

FONCTION MEMOIRE

E	s(t)	s(t+dt)	
E_0	x	0	
E_1	x	1	
E_2	x	x	← Séquentiel

Après stabilisation

Avant application de E

Entrée appliquée au temps t

FONCTION COMPTAGE

E	s(t)	s(t+dt)	
E_0	x	0	
E_1	x	1	
E_2	x	x	
E_3	x	not(x)	← Comptage ou instable

Si E_3 : S = 0, 1, 0, 1, 0
(dt ou T)
000,001,010,011,100,...

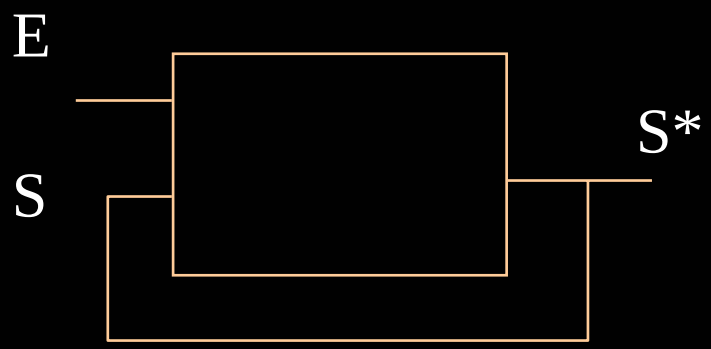
Fonction mémoire : réalisation

E	s(t)	s(t+dt)
E_0	x	0
E_1	x	1
E_2	x	x

On pose s(t+dt) = s*, (on omet le temps)

$$s^* = F(s,E) = s.F(1,E) + \bar{s}.F(0,E)$$

$$= s.A(E) + \bar{s}.B(E)$$



A(E)	B(E)	s*	
0	0	0	Reset
1	1	1	Set
1	0	s	Mémoire
0	1	not(s)	Instable

On va éliminer le cas A = 0, B = 1

Réalisations $s^* = s.A + \bar{s}.B$

On veut éviter A = 0 et B = 1, on va imposer :

A = 0 implique B = 0 (1) ou B = 1 implique A = 1 (2)

ou l'union de ces implications (3)

(1) On pose $B = \bar{R}.S$ et $A = \bar{R}$

$s^* = s.\bar{R} + \bar{s}.\bar{R}.S$ $s^* = (s + S).\bar{R}$

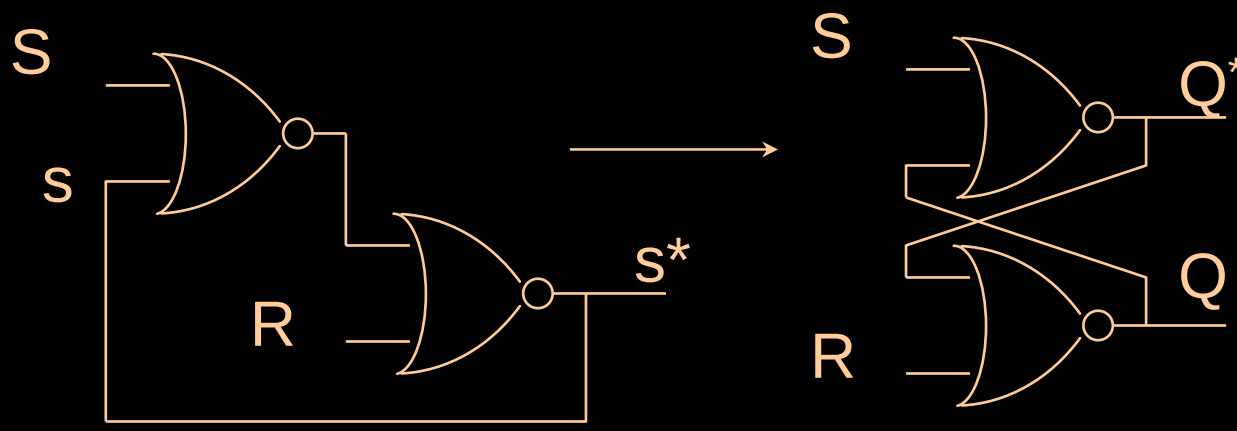
(2) On pose B = S et $A = S + \bar{R}$

$s^* = s.S + s.\bar{R} + \bar{s}.S$ $s^* = s.\bar{R} + S$

(3) $B = S.\bar{R}$ et $A = S + \bar{R}$ $s^* = (s + S.\bar{R}).(S + \bar{R})$

Circuits correspondants (1)

$$s^* = (s + S).\overline{R} = (s + S).(\overline{R} + 0) = (s \downarrow S) \downarrow R$$

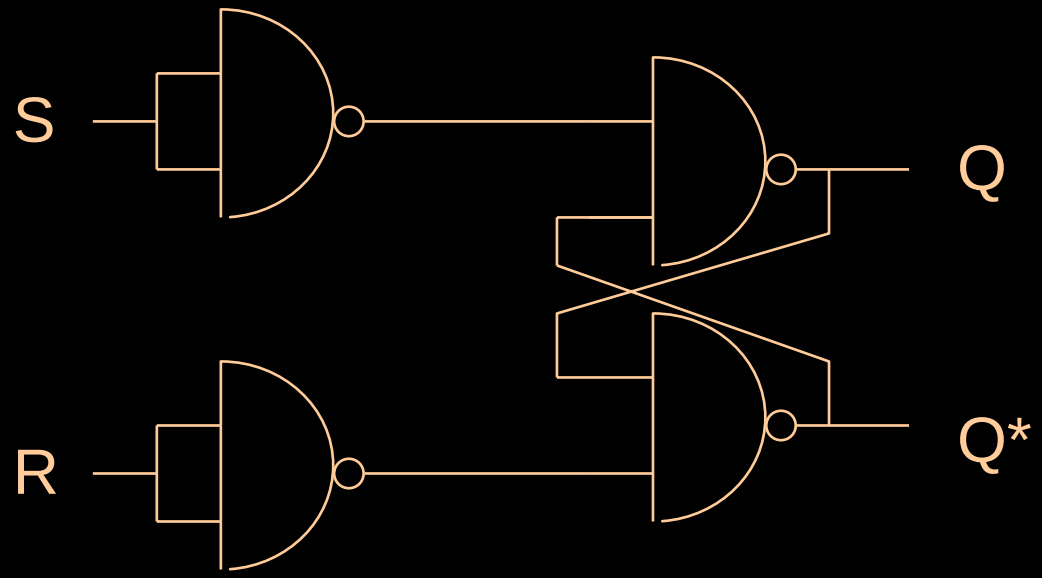


S	R	Q	Q*
0	0	Q_p	Q^*_p
1	0	1	0
0	1	0	1
1	1	0	0

On dit que R est prioritaire
ou que la mémoire est prioritaire au 0

Circuits correspondants (2)

$$s^* = s.\overline{R} + S = (s \uparrow \overline{R}) \uparrow \overline{S}$$



On dit que S est prioritaire
ou que la mémoire est prioritaire au 1

S	R	Q	Q*
0	0	Q_p	Q^*_p
1	0	1	0
0	1	0	1
1	1	1	1

Circuits correspondants (3)

$$s^* = (s + S.\overline{R}).(S + \overline{R})$$

Circuit dit « sans priorité » : jamais utilisé

S=R=1 : Combinaison interdite ?

Dans certains ouvrages : « La combinaison R = S = 1 est interdite »
« ... dangereuse »
« ... provoque une sortie indéfinie »

On remarque que :

- Q = not(Q*) sauf quand R = S = 1
- Les trois mémoires sont identiques sauf quand R = S = 1
- La combinaison intéressante est R=S=0 (mémoire)
  pour passer de R=S=1 à R=S=0 il faut passer par RS = 01 ou RS = 10
- Si aucune précaution n'est prise (ce que l'on vient de faire)
  R = S = 1 provoque l'instabilité (faux dans les circuits du commerce)

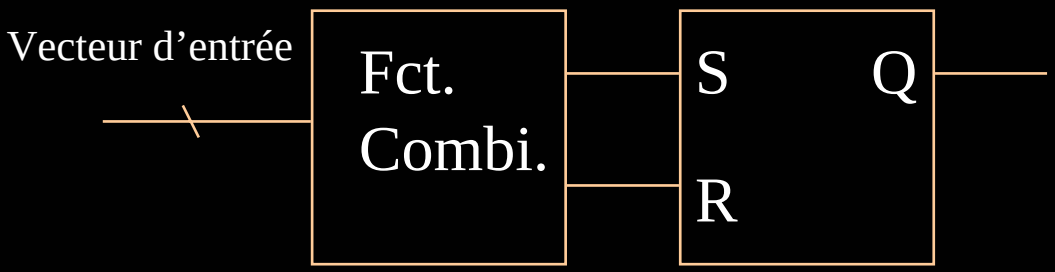
R = S = 1 est une combinaison «déconseillée»

Caractérisation des fonctions mémoires

Exemple : trois capteurs d'effraction : x,y,z

deux capteurs d'extinction a,b et une sirène S

Si x+y+z alors sirène, et si a.b alors plus de sirène



Vecteur d'entrée X

Trois groupes X_S, X_R, X_M

X_S implique Q = 1

X_R implique Q = 0

X_M implique Q = Q_p

Il faut avoir X_S inter X_R inter X_M = ensemble vide

Combinaisons impossibles : simplification de X_S OU X_R OU X_M

Alors R=Expression de X_R et S=Expression de X_S

Caractérisation : exemple

(x,y,z) Set = R(0,7), Reset = R(1,6), Mem = R(3,4), Phi = R(2,5)

x \ yz	00	01	11	10
0	1	0	0	-
1	0	-	1	0

S

$S = x \text{ XOR } z$

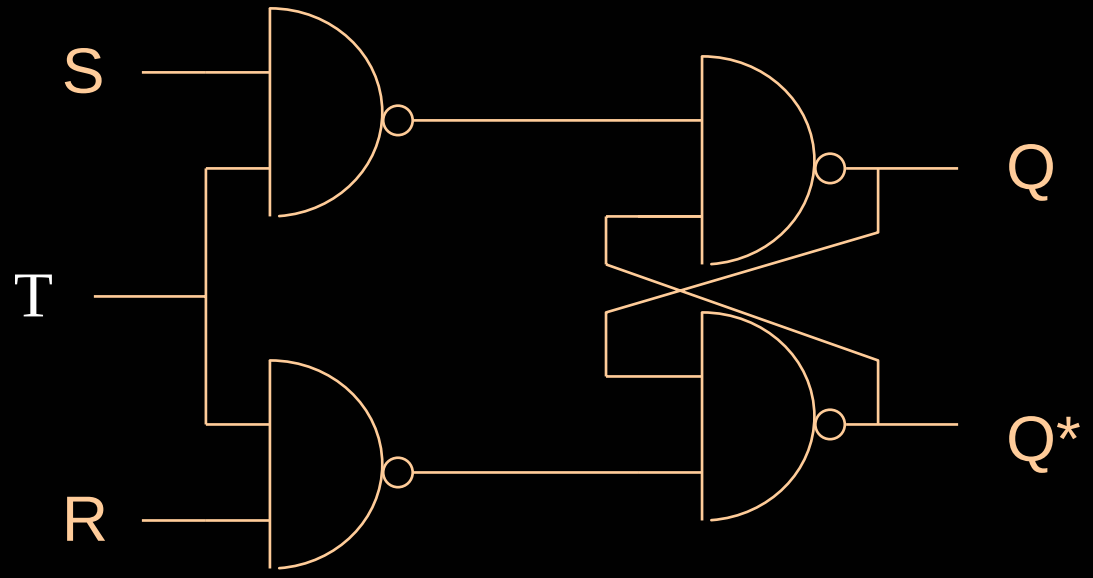
x \ yz	00	01	11	10
0	0	1	0	-
1	0	-	0	1

R

$R = \bar{x}.y.z + x.y.\bar{z}$

On a bien R.S = 0

Mémoire RST



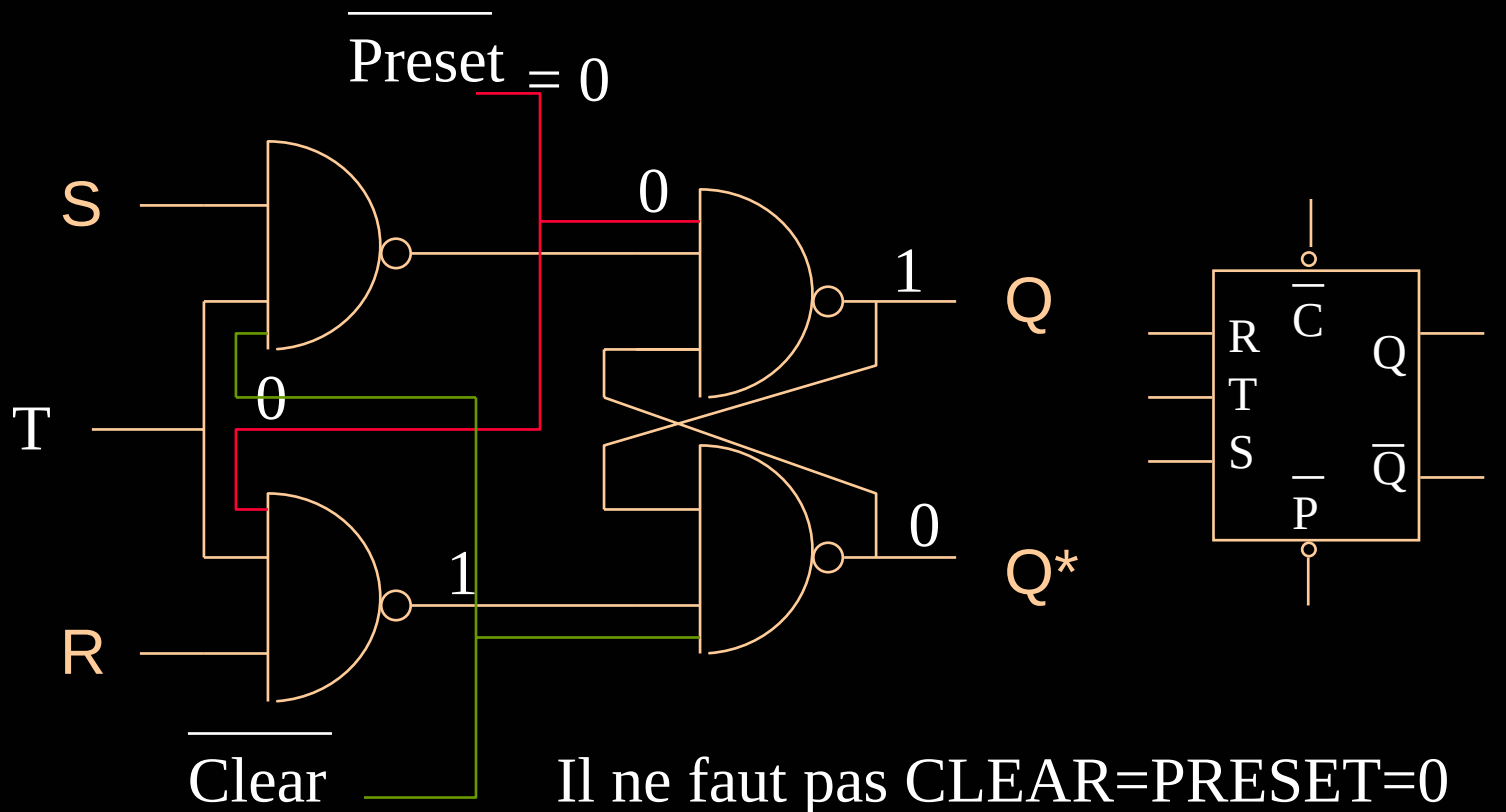
T = 1: Mémoire classique T = 0 : force R=S=0 mémoire «gelée»

Avantages : Préparer S,R puis T = 1
Isoler la mémoire des parasites

Inconvénients : il faut T=1 un certain tps
R=S=1 tjs génant
variation de T avec RS stable

Mémoire RST : initialisation

Mise à 1



Mise à 0

Il ne faut pas CLEAR=PRESET=0

CLEAR et PRESET commandes asynchrones

RST : Comptage ? (1)

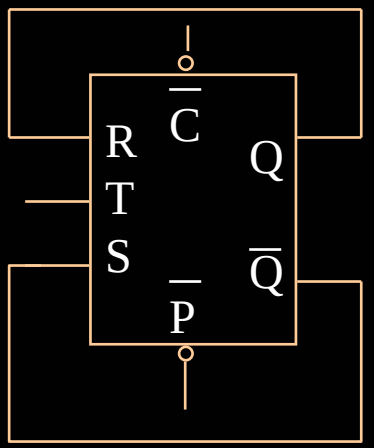
Comptage = diviseur de fréquence par 2 (fonction dibinaire)

Nombre de transitions en sortie = moitié de celles en entrée

Essai : RST déclenchée par front montant de T (pour la démonstration)

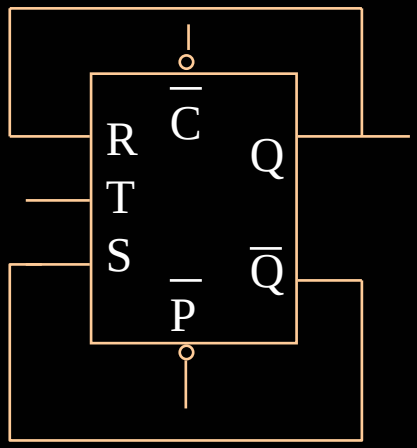
Si Q=0 il faut Q:0→1 quand T:0 →1; il faut donc S=1 et R=0
Si Q=1 il faut Q:1→0 quand T:0 →1; il faut donc S=0 et R=1

On peut faire R = Q et S = $\overline{Q}$



MAIS !!

RST : Comptage ? (2)



Analyse réelle (sur niveau de T)

Hypothèse : T=0, Q=0, Q*=1 d'où S=1, R=0

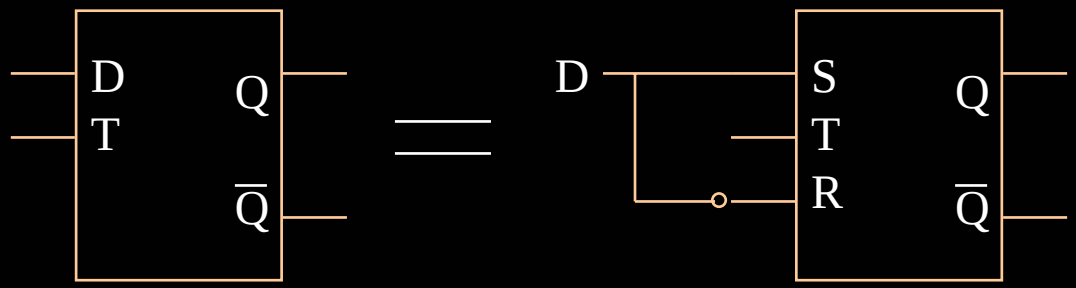
si T passe à 1 : Q=1, Q*=0 (d'où S=0, R=1)
Q=0, Q*=1 (d'où S=1, R=0)
.....

La période dépend du temps de transfert dans la mémoire et est hors de contôle de l'utilisateur : INSTABILITE

LA MEMOIRE RST NE PEUT PAS FONCTIONNER EN COMPTEUR

Mémoire D à verrouillage (verrou ou latch)

RST avec R = $\overline{S}$



Remarque : circuits du commerce sans entrée asynchrone

Si T = 0 les sorties sont «gelées»
Si T = 1 recopie de D sur Q

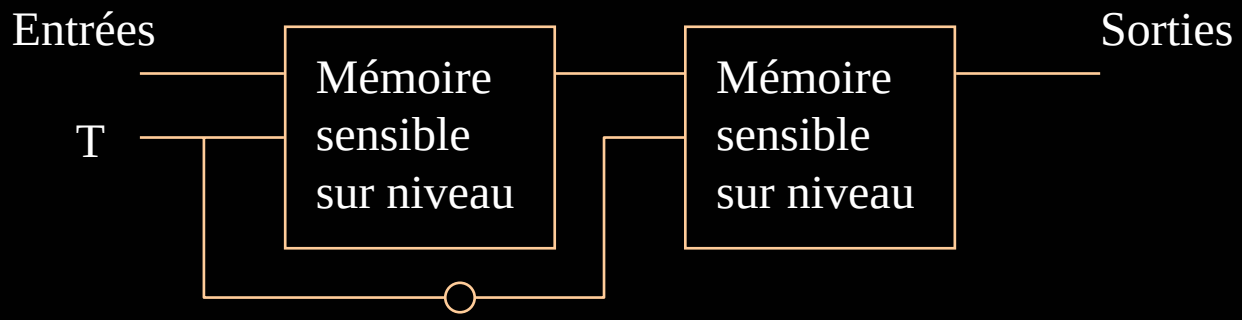
Comme la RST : ne peut pas compter

D	T	Q	Q*
0	0	Q_p	Q^*_p
1	0	Q_p	Q^*_p
0	1	0	1
1	1	1	0

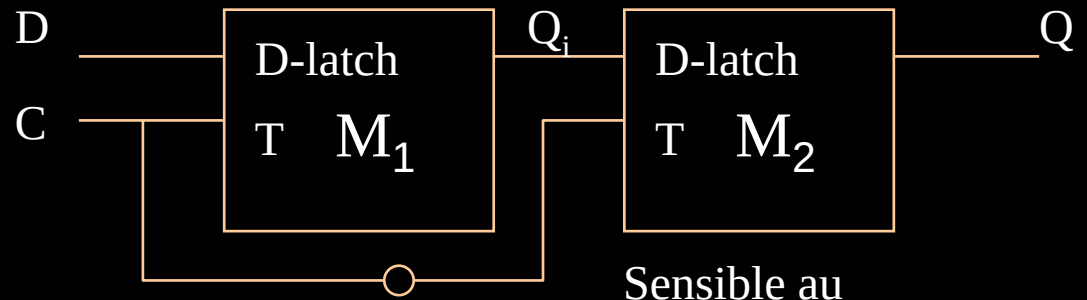
Les bascules (flip-flop)

Mémoires : Problème de sensibilité aux parasites (a)
Ne peut pas compter

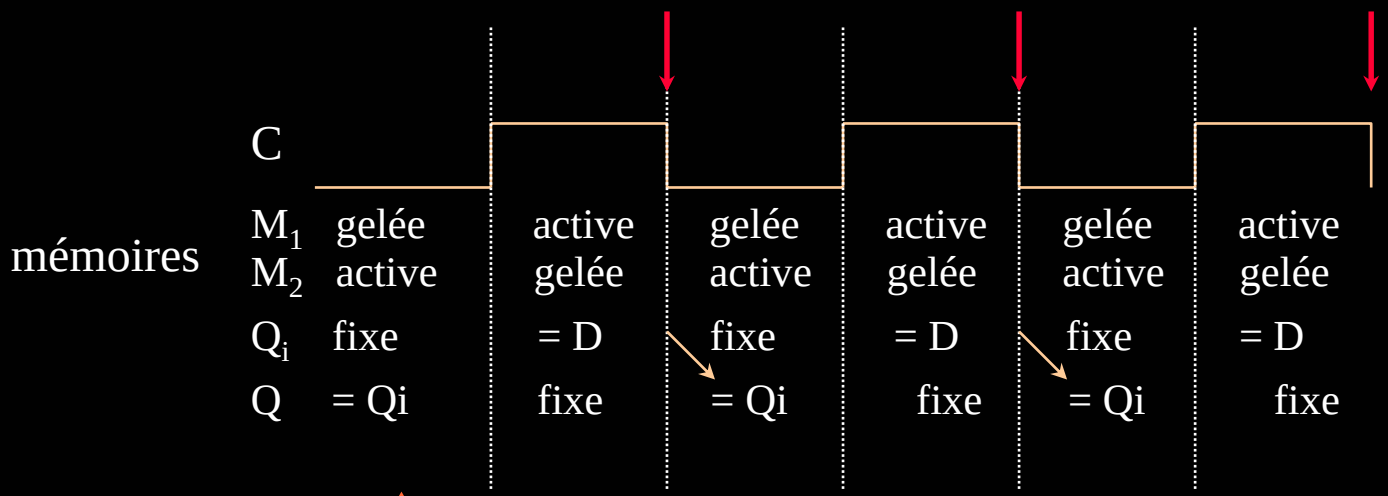
Solution à (a) : Sensible sur niveau et réduire la durée de ceux-ci
Circuit sensible sur front de T (montant ou descendant)
Structure maitre-esclave



Maitre-esclave : explication



Sensible au niveau bas de C



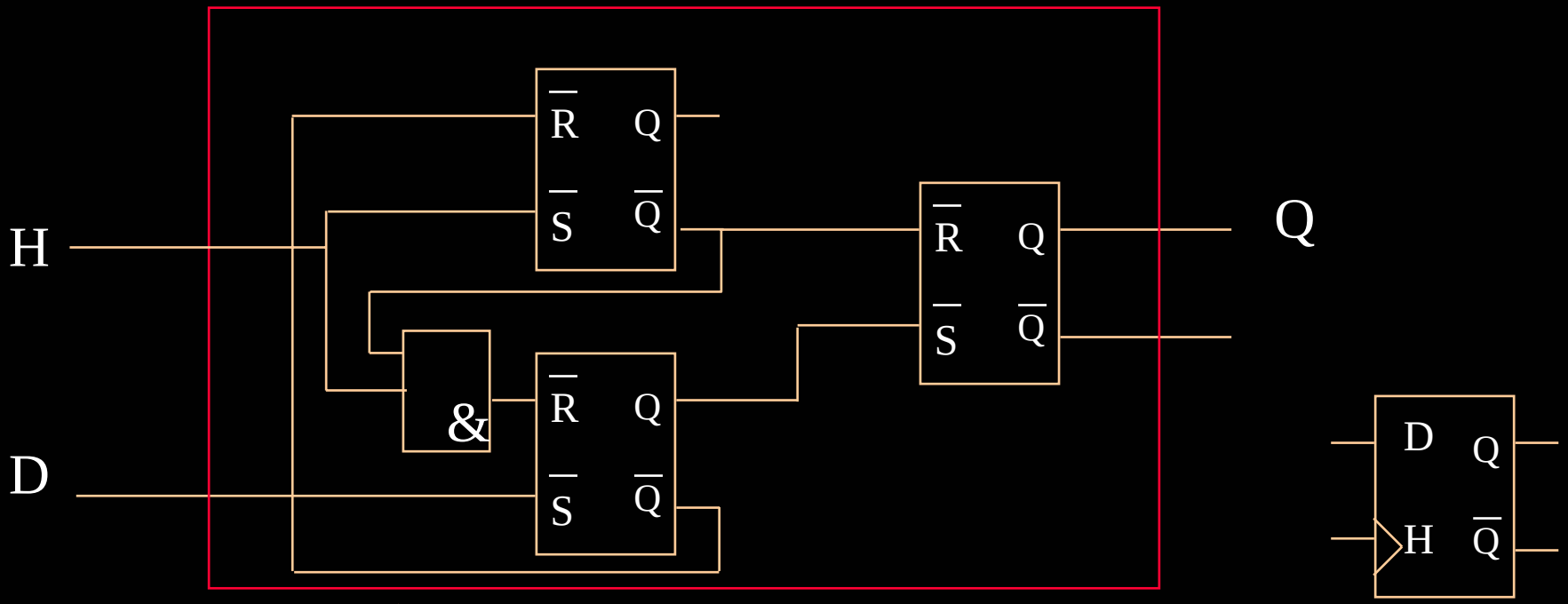
Passage de l'info entre M_1 et M_2 = comportement sur fronts ↓

1er flip-flop à triodes : 1906

Sensibilité sur front

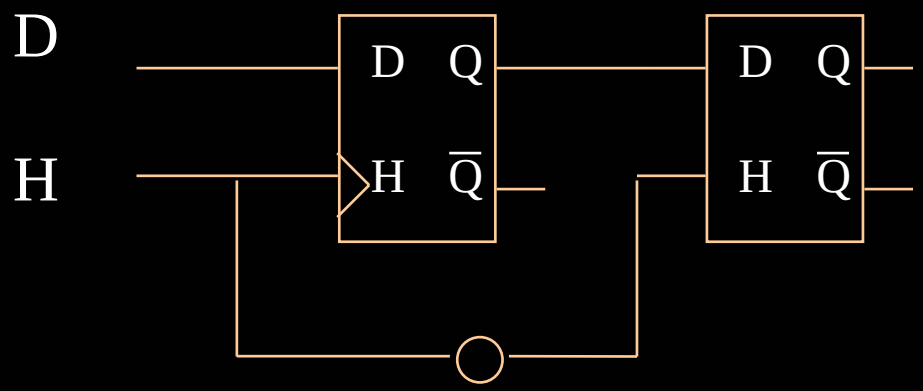
Voir TD Nº 4

Par association de mémoires : Exemple Bascule D (edge-triggered)



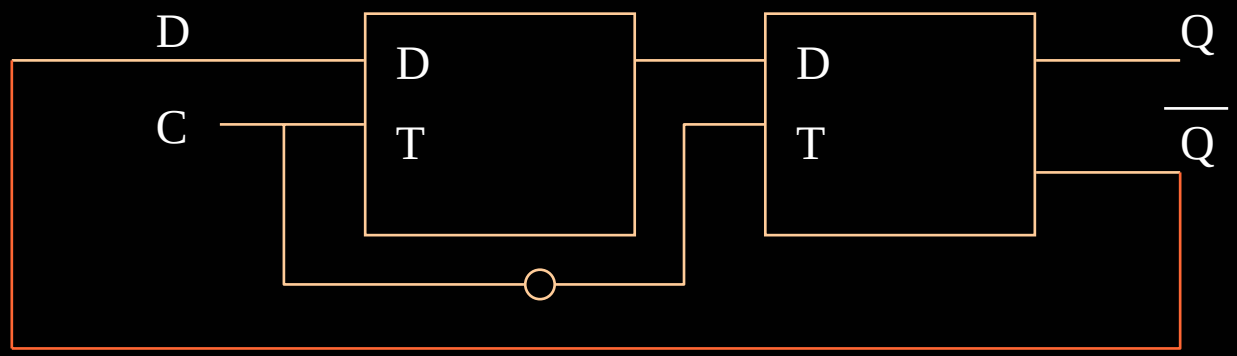
Maitre - esclave + front

Maitre-esclave à capture des entrées ou M/S Data Lock Out



Les données sont capturées sur le front montant de l'horloge et sont répercutées sur les sorties au moment du front descendant

Bascule RST-ME et DT-ME



DT-ME peut elle compter ? (voir transparent n° 13)

Analyse : Si D = 1 : au front descendant de C Q passe à 1 et D à 0

AU PROCHAIN FRONT DE C Q passe à 0 et D à 1

.....

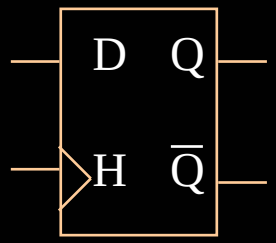
Inversion de la sortie à chaque période d'horloge (sous contrôle utilisateur) = comptage

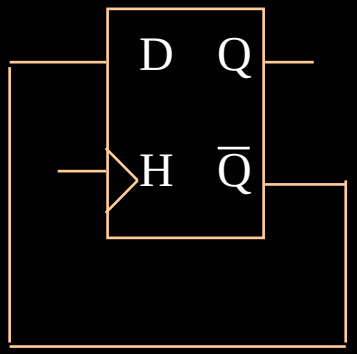
Bascule D-edge (ou bascule D)

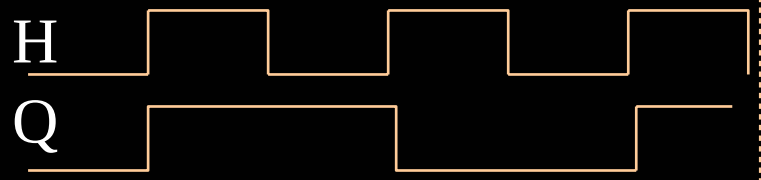
(D comme Data ou Donnée)

Q recopie D sur le front de actif

Si D = $\overline{Q}$ on a un diviseur de fréquence par 2



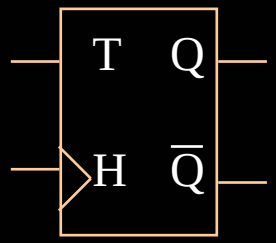




Bascule T-edge (ou bascule T)

(T comme Toggle)

Si T = 0 la sortie est «gelée»
Si T = 1 la sortie est inversée à chaque front actif
(diviseur de fréquence par deux)



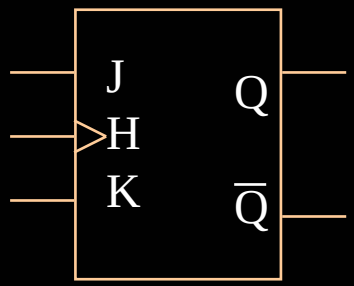
N’existe pas en tant que tel dans les catalogues !!

Bascule JK

(Sous réserve :JK comme Jack / King, sortie Q comme Queen)

RS sur front «améliorée»

	J	K	Q	$\overline{Q}$
Mémorisation	0	0	Q_p	$\overline{Q_p}$
Mise à 1	1	0	1	0
Mise à 0	0	1	0	1
Toggle	1	1	Q_p	$\overline{Q_p}$



Toujours sensible sur front

Récapitulatif

Mémoires (Latches)

RS ou anti-rebond (voir TP)

RST

DT latch ou verrou

Une mémoire ne peut pas compter !!!

Bascules (Flip-flop)

RST/ME et DT/ME

D ou D-edge

T pas fabriquée

JK/ME

JK sur front implicitement

JK Data lock out

Table de transition d'une bascule

Définition : On appelle t_{xy} la transition de la sortie du niveau x au niveau y provoquée par le front d'horloge.

4 cas possibles : t_{00} t_{01} t_{10} t_{11}

Table de transition : les niveaux qui doivent être présents sur les entrées pour provoquer la transition voulue au front d'horloge.

	t_{00}	t_{01}	t_{10}	t_{11}
D	0	1	0	1

	t_{00}	t_{01}	t_{10}	t_{11}
T	0	1	1	0

Table de transition : bascule JK

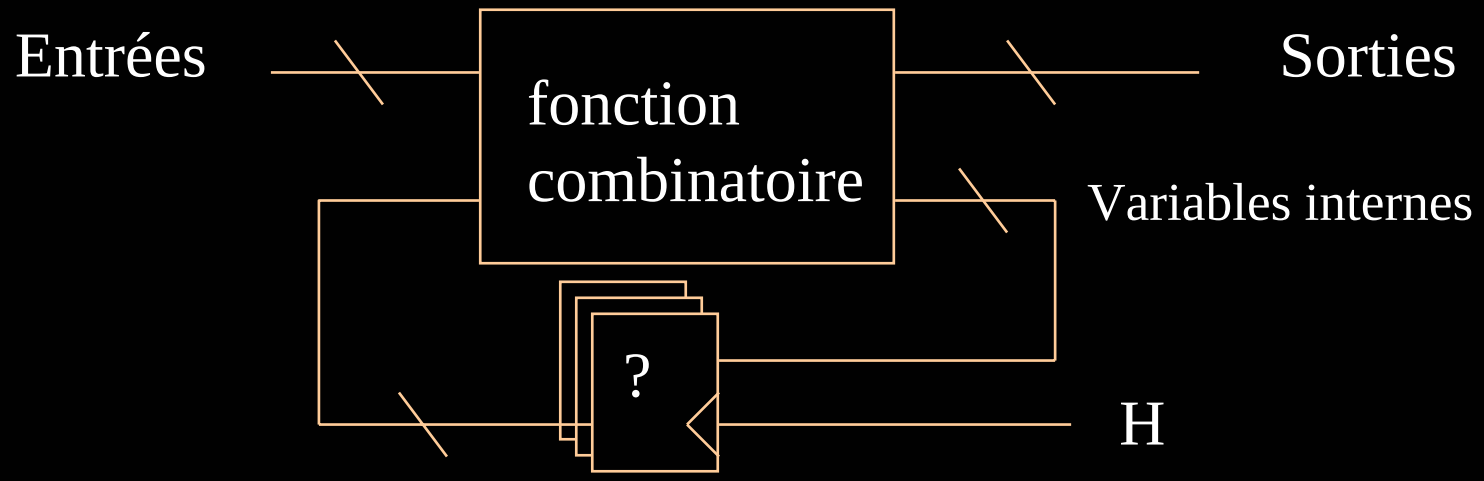
	t_{00}	t_{01}	t_{10}	t_{11}
JK	0-	1-	-1	-0

t_{00}		t_{01}		t_{10}		t_{11}	
mise à 0	01	mise à 1	10	mise à 0	01	mise à 1	10
mémorisation	00	toggle	11	toggle	11	mémorisation	00

Machines séquentielles asynchrones synchronisées

Problème : valider le bouclage des variables internes sur un front d'horloge.

Moyen : forcer les variables internes comme des sorties de bascules



Mach.Séq.Asynch.Synchr. : synthèse

Idem méthode d'huffmann : Au moment de la table d'excitation

- choix d'une bascule,
- remplissage avec la table transition,
- calcul des fonction à mettre à l'entrée des bascules pour obtenir ces transitions

d_1d_2 \ AB	00	01	11	10
00	5	2	1	8
01	3	2	7	8
11	3	6	7	4
10	5	6	1	4

d_1d_2 \ AB	00	01	11	10
00	t_{01}	t_{00}	t_{00}	t_{00}
01	t_{01}	t_{00}	t_{00}	t_{00}
11	t_{11}	t_{11}	t_{10}	t_{11}
10	t_{11}	t_{11}	t_{10}	t_{11}

Q_1Q_2 \ AB	00	01	11	10
00	1	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	1

Détecteur de sens

	t_{00}	t_{01}	t_{10}	t_{11}
D	0	1	0	1

Q_1 D_1

$D_1 = f(A, B, d_1, d_2)$