Logique combinatoire & séquentielle

Johann Laurent

MCF, UBS- Lab-STICC

CNRS UMR-6285

http://www.lab-sticc.eu

Plan

1. **Fonction logique de base & algèbre de Boole**
   1. Fonctions logiques de base
   2. Théorème fondamentaux
   3. Tableaux de Karnaugh

2. **Principales fonctions de la logique combinatoire**
   1. Circuits logiques intégrés
   2. Fonctions de parité
   3. Comparaison
   4. Multiplexage/démultiplexage
   5. Circuits arithmétiques

Plan

3. Logique séquentielle: introduction aux bascules

1. Bascules RS
2. Bascules synchrones
3. Bascules maître/esclave
4. Bascules D
5. Bascules JK

4. Logique séquentielle: applications des bascules

1. Comptage asynchrone
2. Registres

1.1 Fonctions logiques de base

- **L'information en électronique numérique est véhiculée sous forme de 0 ou de 1 appelé bit: 8 bits = 1 octet**

- **Le '0' représente un niveau de tension de 0V et le '1' un niveau de tension de VDD (5V; 3,3V; 2,5V .. Dépend de la techno utilisée)**

- **Les variables logiques (ou booléenne) représentent les variables d'entrées d'un système (capteurs) alors que les sorties représentent les actionneurs (moteurs, vérins…)**

- **Une fonction booléenne peut être défini par:**
  - Son équation logique : S = a.b + a.c
  - Sa table de vérité qui représente tous les états possibles du système

1.1 Fonctions logiques de base

- **Table de vérité**

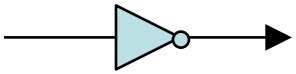
a	b	c	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

1.1 Fonctions logiques de base

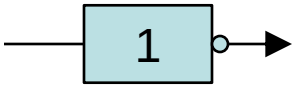
- **Non logique ou inverseur (NOT)**
  - $S = \bar{a}$

a	S
0	1
1	0

  - Symbole



Norme US



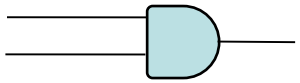
Norme UE

1.1 Fonctions logiques de base

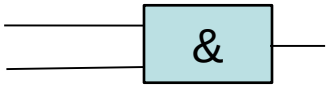
- **ET logique (AND)**
- $S =$ **a.b**

a	b	S
0	0	0
0	1	0
1	0	0
1	1	1

- Symbole



Norme US



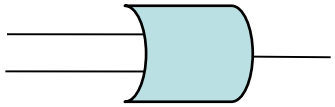
Norme UE

1.1 Fonctions logiques de base

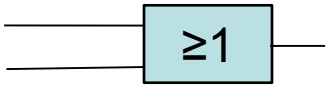
- **OU logique (OR)**
- $S = \mathbf{a + b}$

a	b	S
0	0	0
0	1	1
1	0	1
1	1	1

- Symbole



Norme US



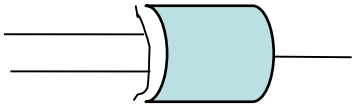
Norme UE

1.1 Fonctions logiques de base

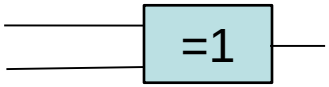
- **OU exclusif (XOR)**
  - S=$\bar{a}b + a\bar{b}$

a	b	S
0	0	0
0	1	1
1	0	1
1	1	0

  - Symbole



Norme US



Norme UE

  - Il est possible de combiner la porte NOT avec les autres portes pour obtenir les fonctions NAND, NOR ou XNOR.

1.2 Théorèmes fondamentaux

- **OR:** $a + 0 = a$; $a + 1 = 1$; $a + a = a$; $a + \overline{a} = 1$
- **AND:** $a \,.\, 0 = 0$; $a \,.\, 1 = a$; $a \,.\, a = a$; $a \,.\, \overline{a} = 0$
- **NOT:** $\overline{\overline{a}} = a$
- **Commutativité: a + b = b + a; a . b = b . a; idem pour le XOR**
- **Théorème d'absorption: a + ab = a**
- **Théorème d'allègement:** $a + \overline{a}b = a + b$
- **Théorème de De Morgan:** $\overline{a + b} = \overline{a} \,.\, \overline{b}$; $\overline{a.b} = \overline{a} + \overline{b}$

1.3 Tableau de Karnaugh

- **Utilisation du codage de Gray**
  - En codage de Gray un seul bit peut changer d'état par cycle
  - Ce tableau sert à simplifier des équations logiques

BINAIRE			GRAY		
B2	B1	B0	G2	G1	G0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

1.3 Tableau de Karnaugh

- **Utilisation du codage de Gray**
  - Exemple: $S = ab + abc + b + a\overline{bc}$

		b		a	
		00	01	11	10
c	0	0	1	1	1
	1	0	1	1	0

- **Il faut faire des regroupements en puissance de 2.**
  - Pour chaque regroupement on regarde quels sont les termes « stables » et on fait un OU logique entre les différents regroupements.
  - Pour notre exemple $S = b + a\overline{bc}$
  - Si la fonction n'est pas entièrement définie alors on utilise des X dans le tableau

1.3 Tableau de Karnaugh

- **Exemple**

c \ b	00	01	11	10
0	0	1	0	X
1	0	1	X	1

- $S = \bar{a}b + bc + a\bar{b} = a \oplus b + bc$

2.1 Circuits logiques intégrés

- **2 grandes familles de composants: TTL & CMOS**
  - A. Famille TTL:
    - *Famille la plus ancienne, faible coût utilisant des transistors bipolaires alimentés en 0-5v et ayant une grande consommation énergétique.*
  - B. Famille CMOS
    - *Famille faible consommation, grande densité d'intégration car utilisation de transistor MOSFET*

- **Toutes les fonctions de base peuvent être réalisées**

2.2 Fonction de parité

- **Fonction aussi appelée bit de parité permet la détection d'erreurs de transmission.**

- **Génération du bit de parité à l'émission et contrôle à la réception.**

- **La fonction de parité n'est en fait qu'un ou exclusif fait sur l'ensemble du mot binaire:**
  - $P = a_n \oplus a_{n-1} \oplus a_{n-2} \ldots \oplus a_1 \oplus a_0$
  - Si le nombre de bits à '1' est pair alors P = 0 et P=1 sinon.

- **Exemple de circuit: 74180 parité sur 1 octet**

2.3 Fonction de comparaison

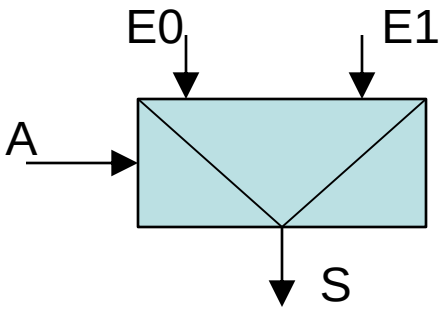
- **Cette fonction permet de comparer 2 nombres binaires**

a	b	a>b	a = b	a < b
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

- **a > b :** $a\overline{b}$
- **a = b :** $\overline{a \oplus b}$
- **a < b :** $\overline{a}b$

2.4 Fonction de multiplexage/démultiplexage

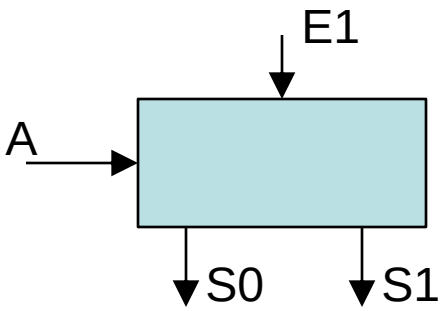
- **Le multiplexage**
  - Cette fonction est souvent appelée MUX ou aiguillage
  - Exemple de composant le 74150



  - S = E0 si A = 0 S = E1 si A = 1
  - Cette fonction permet de sélectionner une ligne d'entrée par une adresse et de l'aiguiller vers la sortie.
  - Le composant peut avoir autant d'entrées que nécessaire
  - Les données d'entrées peuvent être composées de plusieurs bits

2.4 Fonction de multiplexage/démultiplexage

- **Le démultiplexage**
  - C'est la fonction inverse du multiplexeur; il aiguille une entrée vers l'une des sorties possibles choisie par les adresses (A).



  - La principale utilisation de cette fonction est la transmission de données série/parallèle. Pour émettre les n entrées on utilise un mux qui va sérialiser l'information et on utilise un démux en sortir pour re paralléliser.

2.5 Fonctions arithmétiques

- **Additionneur**
  - Cette fonction est un opérateur de base pour le traitement de données numériques. Le calcul commence par l'addition des bites de poids faibles (LSB).
  - Demi additionneur (sur 1 bit)

a	b	R_n	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

  - S= $a \oplus b$ (somme) $R_n = ab$ (retenue)

2.5 Fonctions arithmétiques

- **Additionneur complet ou full adder (sur 1 bit)**

a	b	R_{n-1}	R_n	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

- S= $a \oplus b \oplus R_{n-1}$ $R_n = (a \oplus b)R_{n-1} + ab$

2.5 Fonctions arithmétiques

- **Le multiplieur binaire :**
  - La multiplication binaire revient à faire une addition suivie d'un décalage

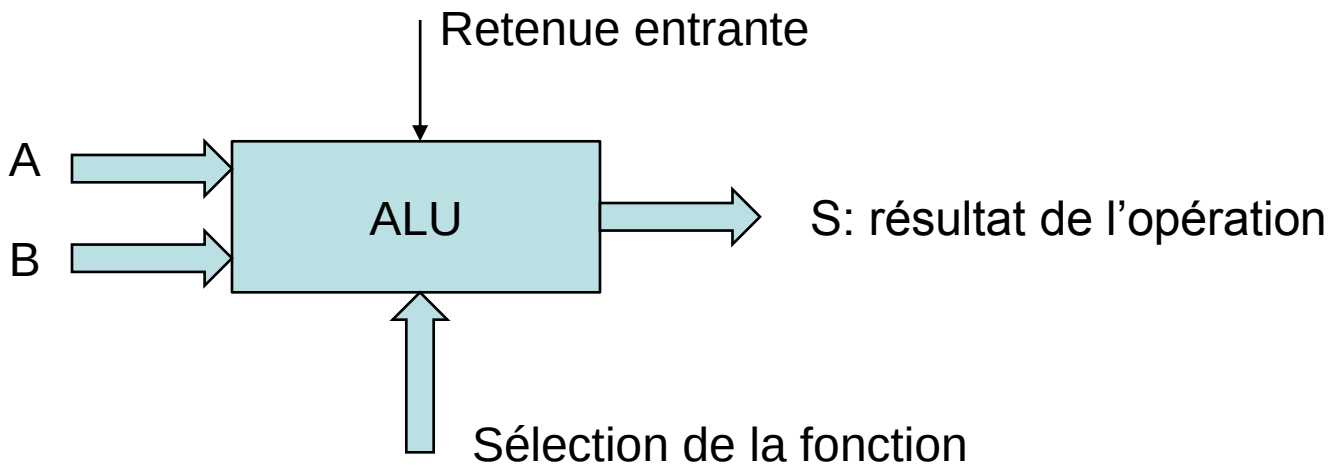
		A1	A0
		B1	B0
		A1B0	A0B0
	A1B1	A0B1	0
P3	P2	P1	P0

Retenue

Produit

2.5 Fonctions arithmétiques

- **Unité Arithmétique et Logique (UAL ou ALU en anglais):**
  - Une UAL est un composant permettant de réaliser plusieurs opérations logiques et arithmétiques différentes.



3. Logique séquentielle

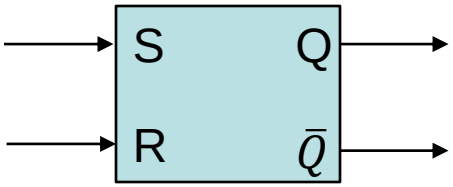
- En logique combinatoire la connaissance seule des entrées suffit à déterminer les sorties
- Pas de mémorisation des états passés du système
- 2 combinaisons identiques des entrées donnent toujours 2 combinaisons de sortie identique.

- En logique séquentielle l'état de sortie dépend des entrées mais également de l'état précédent du circuit (effet mémoire)
- 2 combinaisons identiques d'entrée peuvent donc donner 2 combinaisons différentes de sorties.

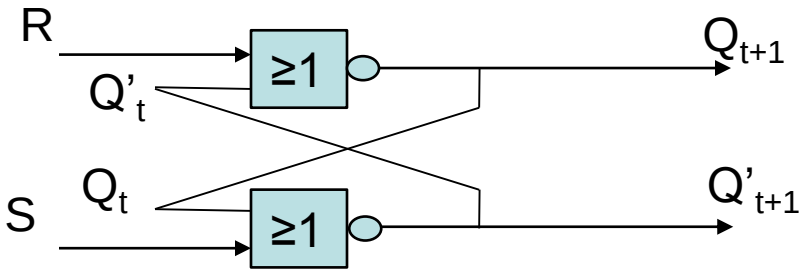
- Le circuit de mémorisation le plus répandu est la bascule

3.1 La bascule RS

- **Principe:**
  - C'est le circuit séquentiel le plus simple dont le rôle consiste à noter la présence d'une information et de conserver cet état lorsque l'information disparaît (principe du point mémoire)



  - 2 entrées R et S: S (set) est l'entrée de mémorisation et R (reset) est l'entrée d'effacement de la mémoire
  - 2 sorties Q et $\bar{Q}$: Q étant la sortie donnant l'information mémorisée
- **Constitution : portes NOR (Non Ou)**

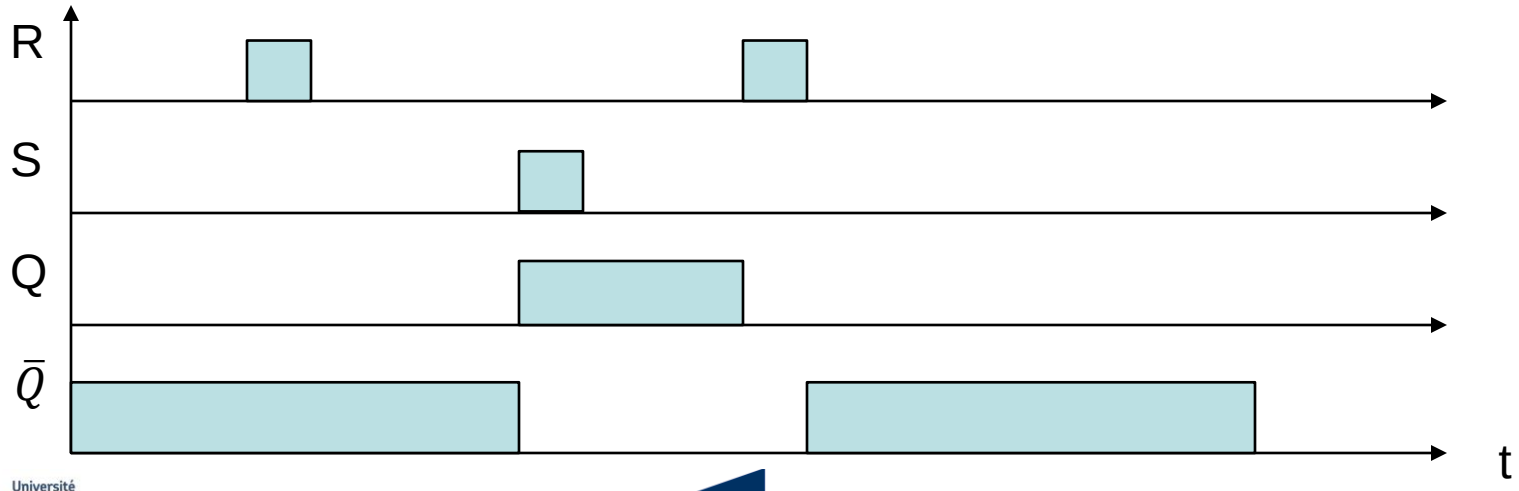


3.1 La bascule RS

- **Fonctionnement: table de vérité**

S	R	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	X

- $Q_{t+1}=\bar{R}(S+Q_t$



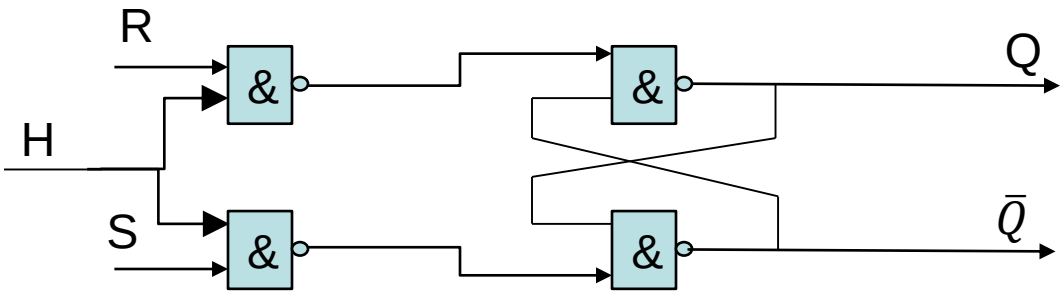
3.2 Les bascules synchrones

- **Inconvénient des circuits séquentiels asynchrone comme la RS c'est la sensibilité aux parasites**
- **Si parasité suffisamment long sur R ou S alors les sorties changent d'état**

- **Circuits synchrones: les entrées sont validées par un signal auxiliaire appelé horloge -> insensibilité aux parasites.**
  - Toute bascule dispose d'une entrée d'horloge (appelée H ou Clk) qui est un signal rectangulaire périodique. L'horloge peut être active sur front montant ou descendant et ce n'est que sur ce front actif que les changements d'état de la bascule sont possibles.

3.2 Les bascules synchrones

- **Bascules RSH ou RS synchrone**
  - R et S sont des entrées synchrones car elles dépendent de H; H verrouille la bascule,
  - Si H=1 alors les états peuvent changer sinon pas possible
  - Il faut que l'entrée est un temps de stabilisation suffisant avant le front d'horloge mais un temps de maintient suffisant pour être prise en compte.

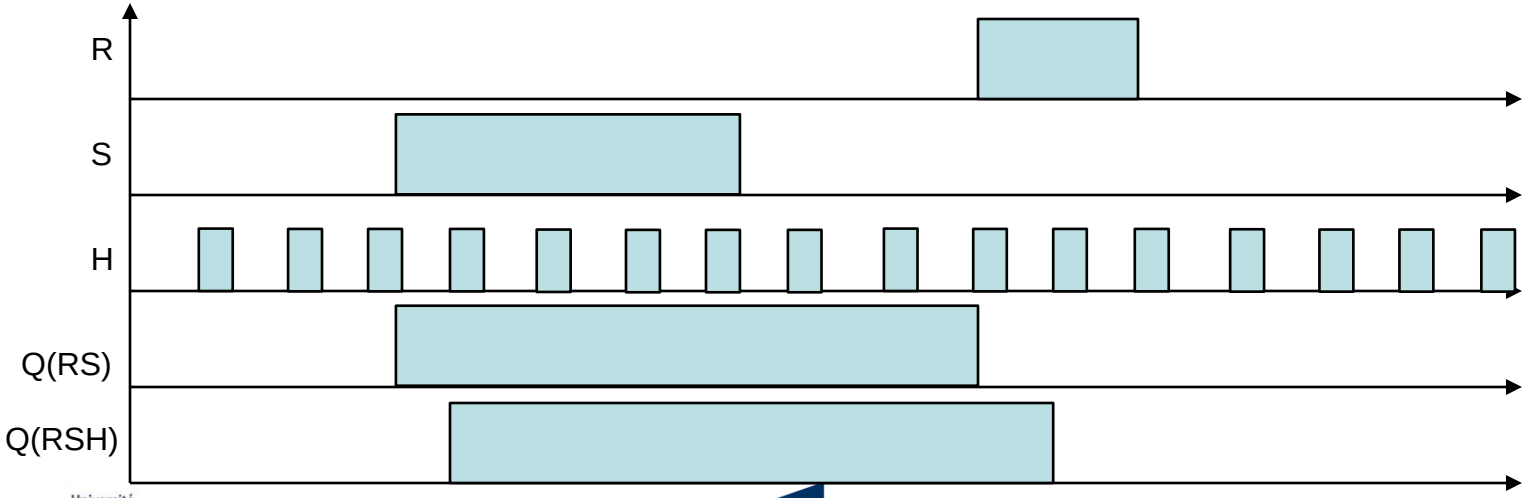


3.2 Les bascules synchrones

- **Table de vérité**

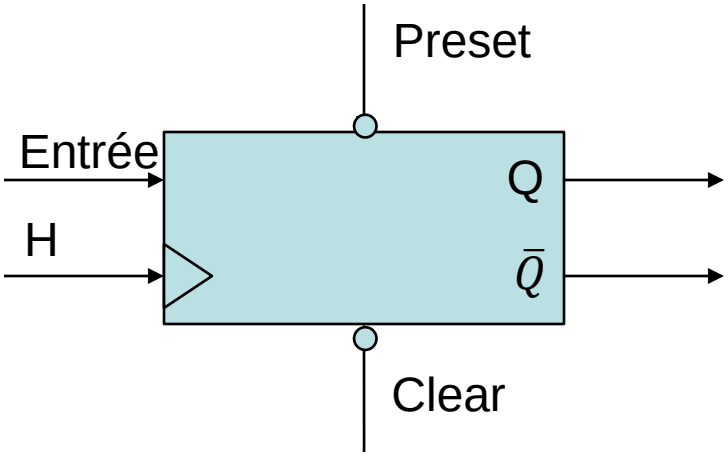
S	R	H	Q_{t+1}	$\overline{Q_{t+1}}$
X	X	0	Q_t	$\overline{Q_t}$
0	0	1	Q_t	$\overline{Q_t}$
0	1	1	0	1
1	0	1	1	0
1	1	1	X	X

  - Si H = 1 alors nous retrouvons le fonctionnement de la bascule RS vu plus haut



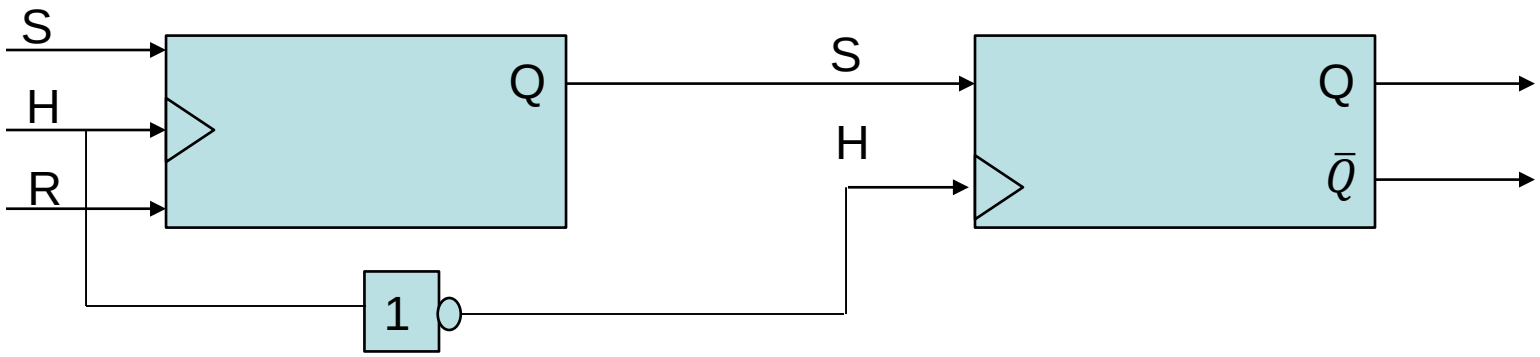
3.2 Les bascules synchrones

- **Entrées asynchrones**
  - Souvent appelées entrées de forçage elles permettent la mise à ‘1’ ou ‘0’ de la bascule même si l’horloge n’est pas valide.
  - La mise à ‘1’ se fait via une entrée Preset
  - La mise à ‘0’ se fait via une entrée Clear ou RAZ
  - Elles sont souvent actives sur niveau logique bas (‘0’)



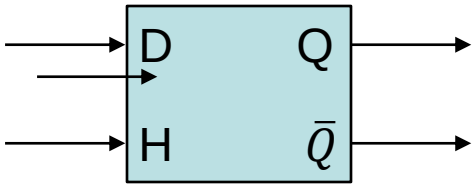
3.3 Bascules maître/esclave

- **Une bascule maître/esclave est formée par 2 bascules commutant sur des valeurs différentes d'horloge.**
- **La 1ère bascule est le maître et la seconde l'esclave**
- **C'est une cellule mémoire à 2 étages.**



3.4 Bascules D

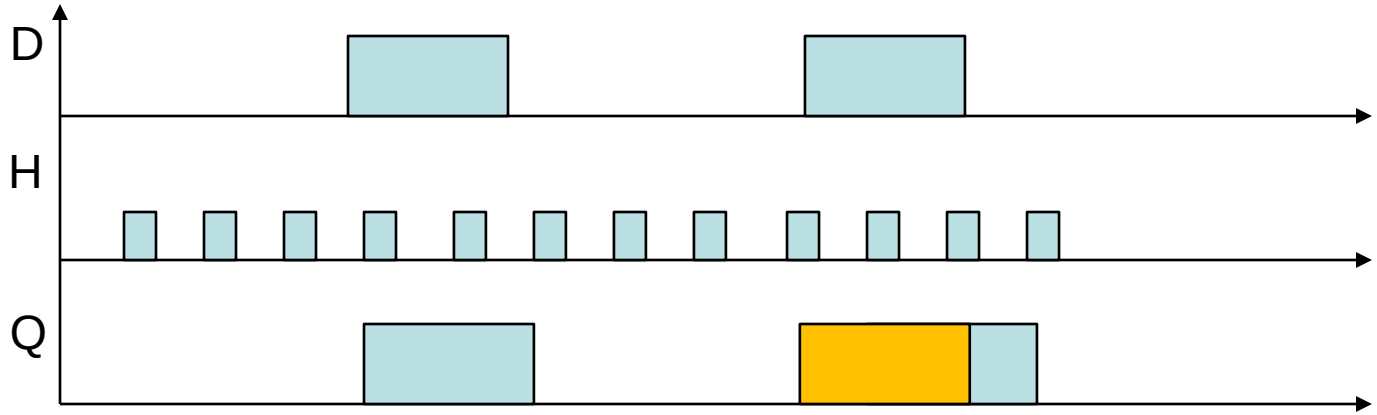
- **Latch D:**
  - Réalisé à partir d'une bascule RS avec un inverseur entre R &S interdisant la combinaison R = S = 1
  - Une seule variable d'entrée D (Delay) = S = $\bar{R}$
  - Si H = 1 on recopie l'entrée sinon on conserve l'information en mémoire



H	D	Q_{t+1}
0	X	Q_t
1	0	0
1	1	1

3.4 Bascules D

- **Chronogramme**



  - Ici la bascule fonctionne sur front montant
  - Possible d'avoir un fonctionnement sur état

3.5 Bascules JK

- **Table de vérité**

H	J	K	Q_{t+1}
↑	X	X	Q_t
↑	0	0	Q_t
↑	0	1	0
↑	1	0	1
↑	1	1	$\bar{Q}_t$

- **Elle est surtout utilisée dans les compteurs car protégée contre les parasites**
- **On peut câbler une JK en D en faisant D = J = K**

4.1 Comptage asynchrone

- **Compteurs utilisés pour:**
  - Effectuer le comptage d'évènements
  - Réaliser des opérations de séquencement
  - Réaliser de la division de fréquence
  - Réaliser des opérations arithmétiques

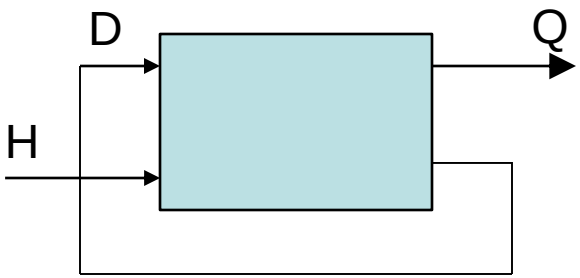
- **Le modulo d'un compteur est égal au nombre de combinaisons différentes obtenues en sortie**
- **Un compteur à n bits peut prendre 2^n combinaisons possibles**
- **Utilisation de bascules pour réaliser les compteurs**

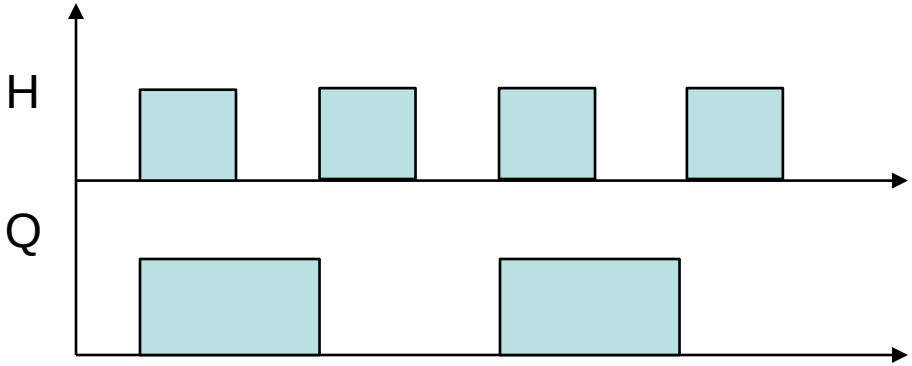
- **Un compteur possède un certain nombre de caractéristiques:**
  - La capacité de comptage
  - Le code comptage (binaire, BCD, Gray etc..)
  - Le sens du comptage
  - La vitesse de comptage
  - La possibilité de pré sélection

4.1 Comptage asynchrone

- **Compteurs asynchrones**
  - Appelés aussi compteurs séries
  - Propage en cascade les changements d'états
  - Elément de base du compteur asynchrone : le diviseur de fréquence

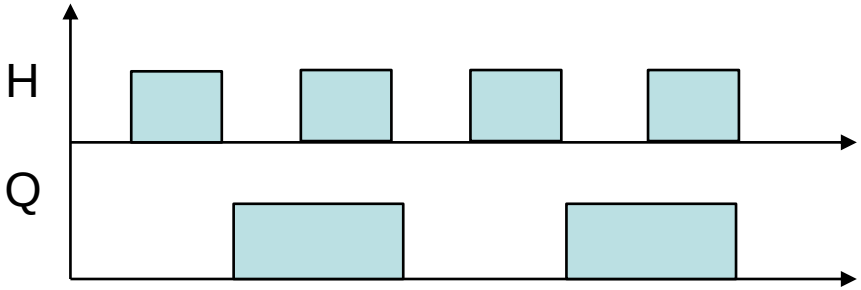
  - Diviseur de fréquence: il suffit de relier la sortie $\bar{Q}$ à l'entrée D pour qu'à front actif d'horloge qui est aussi l'entrée la sortie Q change d'état



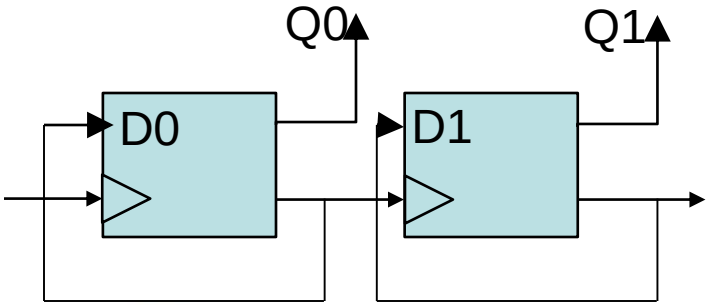


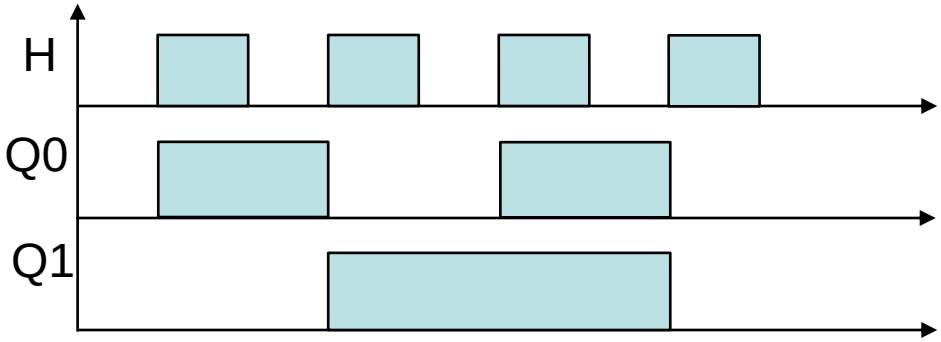
4.1 Comptage asynchrone

- **Compteurs asynchrones**
  - Avec bascules JK



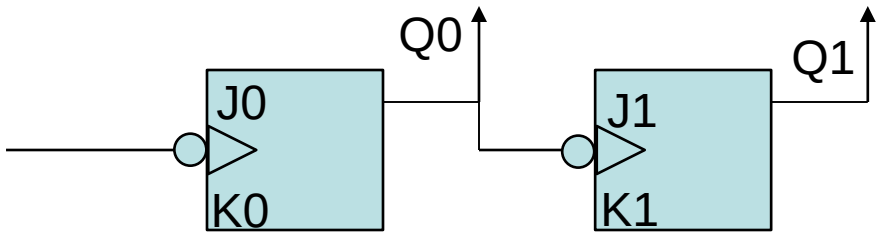
- **Compteur asynchrone modulo 2^n**
  - Pour réaliser un compteur asynchrone modulo 2^n il suffit de mettre en cascade n bascules câblées en diviseur de fréquence.
  - Le poids faible du compteur est la sortie Q de la première bascule
  - Un compteur modulo 2^n est un diviseur de fréquence par 2^n





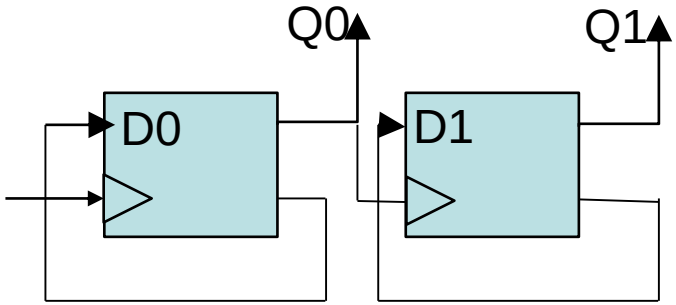
4.1 Comptage asynchrone

- **Compteur asynchrone modulo 2^n**
  - Bascules JK: toutes les entrées JK des bascules sont mises à ‘1’



  - Attention en logique TTL une entrée en « l’air » est par défaut mise à ‘1’ mais ce n’est pas vrai en technologie CMOS.

- **Décompteur asynchrone**



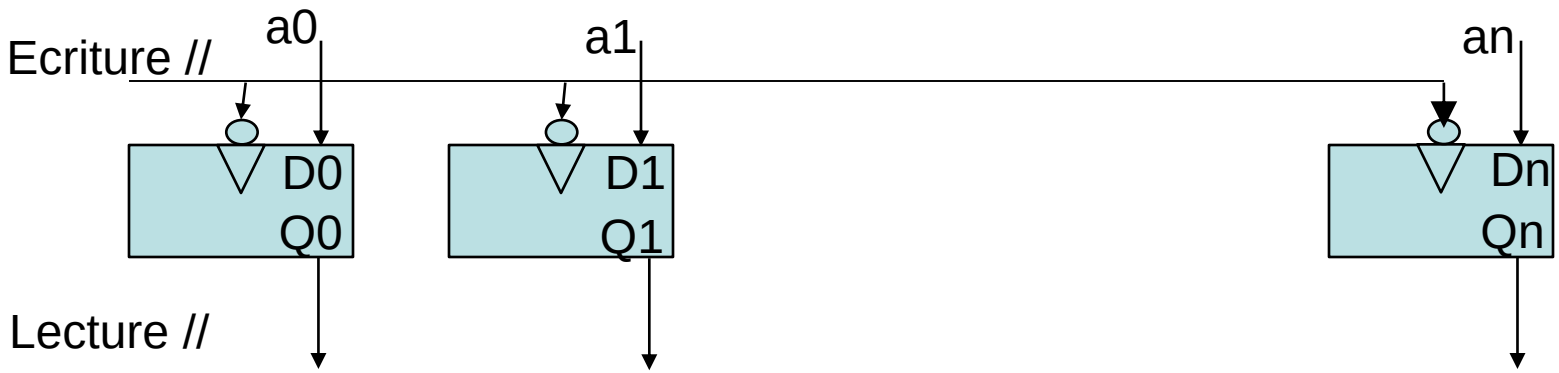
4.1 Comptage asynchrone

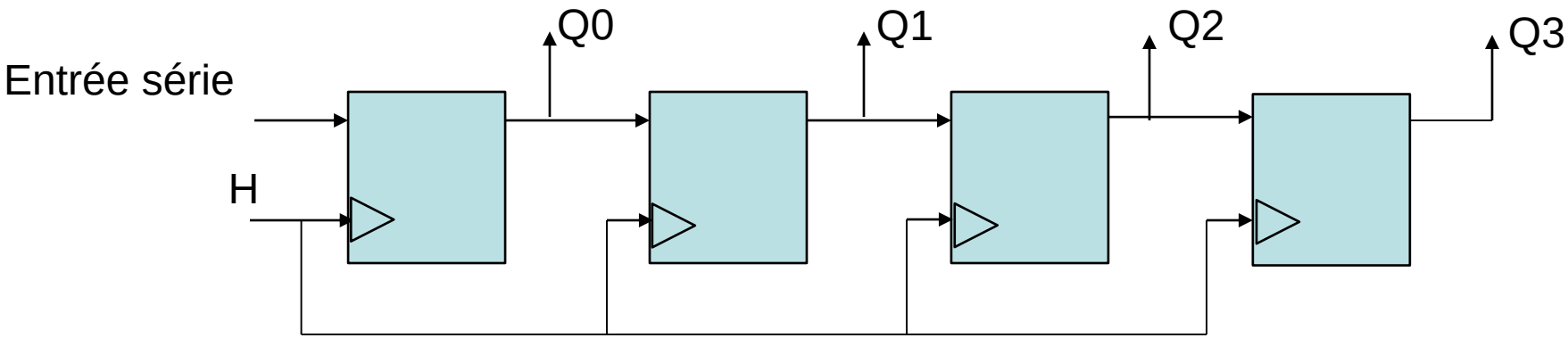
- **Remarque:**
  - La réalisation des compteurs asynchrones entraine par construction des délais de propagation car toutes les bascules n'utilisent pas le même signal d'horloge.
  - La solution est de réaliser des compteurs synchrones mais ils sont beaucoup plus complexes à mettre en œuvre : rajout de logique combinatoire autour des bascules.

4.2 Les registres

- **Un registre sert à mémoriser un certain nombre d'information de façon temporaire**
- **Il est constitué de bascules: une bascule par bit**
- **Deux grands types de registre:**
  - Registre de mémorisation
  - Registre à décalage

- **Registre de mémorisation**



4.2 Les registres

- **Registre à décalage**
  - Mémorisation série du mot binaire et non parallèle
  - Pour mémoriser n bascules pour mémoriser un mot binaire de n bits
  - Si un mot binaire est décalé n fois à droite alors c'est équivalent à le diviser par 2^n
  - Si un mot binaire est décalé n fois à gauche alors c'est équivalent à le multiplier par 2^n