ARC1 - Cours n°7

CIRCUITS SEQUENTIELS SIMPLES

1. GENERALITES

1. GENERALITES

- Circuit combinatoire :

- ✓ Sans rétroaction (sans retour des sorties dans les entrées)
- ✓ Signaux de sortie ne dépendent donc que des signaux d'entrée au même instant

- Circuit séquentiel :

  - ✓ Possède des rétroactions : signaux de sortie ne dépendent pas seulement des entrées mais aussi de leur séquence
  - ✓ Se rappelle des entrées et des états précédents : mémoire du passé
  - ✓ **Etude des circuits combinatoires : algèbre de Boole**
  - ✓ **Etude des circuits séquentiels : théorie des automates finis**
  - ✓ Peut être utilisé pour mémoriser des informations

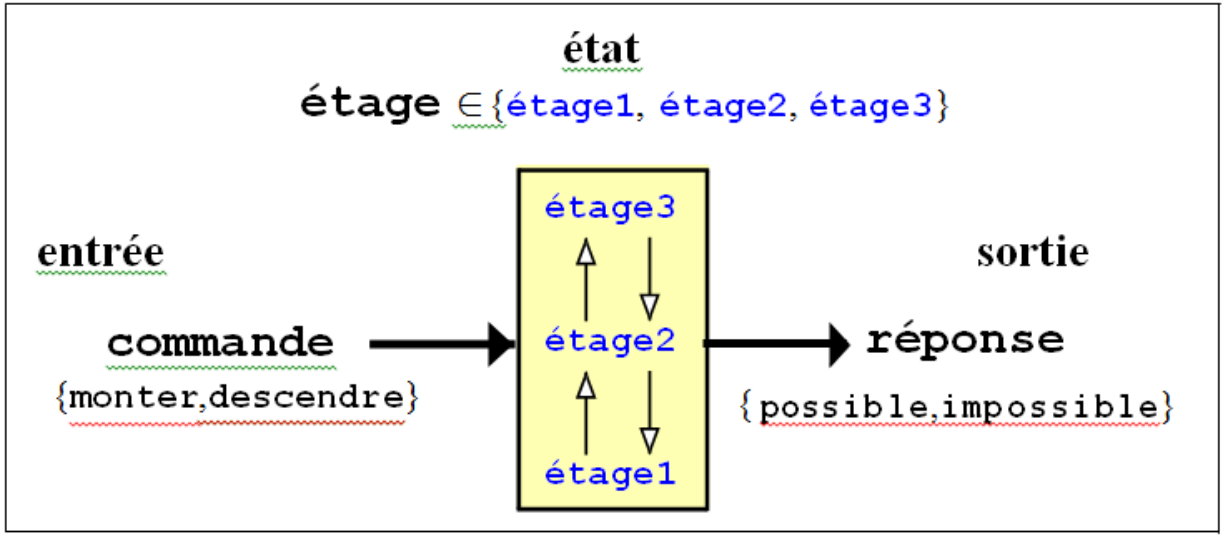
- Automate fini :

  - ✓ Ne peut prendre qu'un nombre fini de valeurs, appelés états internes
  - ✓ Caractérisé par : sa réponse (sortie), son entrée, son état
  - ✓ Description tient compte des temps (t, t+1)

1.1. Définitions

• Système séquentiel : notion d'état interne - entrée : événement → changement d'état - sortie : fonction de l'état et de l'entrée	• Système séquentiel simple : - décrit facilement par diagramme d'états ou table des états - petit nombre d'états (maximum quelques dizaines)

- Exemple 1 : ascenseur à trois étages



✓ Commande (entrée)
= monter

✓ Réponse (sortie)
= possible pour étage1 , étage2
= impossible pour étage3

- Exemple 2 : Riche ou pauvre *(Ref. A. Dipanda Circuits séquentiels)*

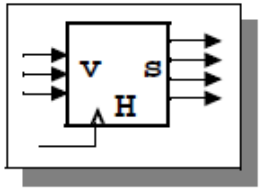
– une personne a trois états (pauvre, modeste, riche)
– deux événements (argent, facture).
En fonction de l'événement, la personne va changer d'état.

		top		top		top		top	
état	pauvre	→	modeste	→	riche	→	modeste	→	pauvre
évén^t		argent		argent		facture		facture	
sortie		plaisir		plaisir		décept°		décept°	

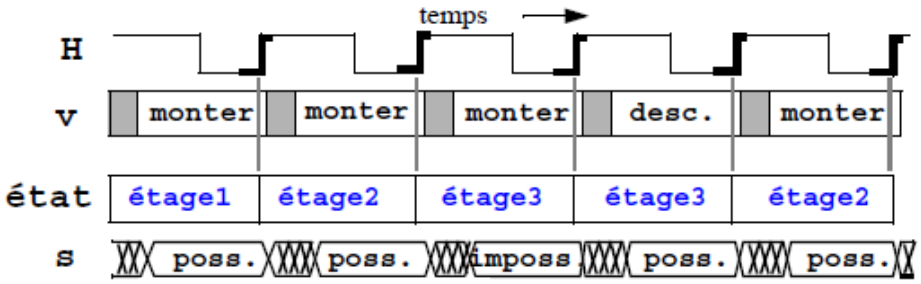
- Système synchrone :

  - ✓ Changement d'état par événements instantanés, tops d'horloge :
  - ✓ Etat à t+1 (nouvel état) dépend de l'état au temps t (état courant) et entrée au temps t.
  - ✓ La sortie ne peut changer qu'aux moments des fronts d'horloge.

entrées de valeurs **v** entrée d'horloge **H** sorties **s**



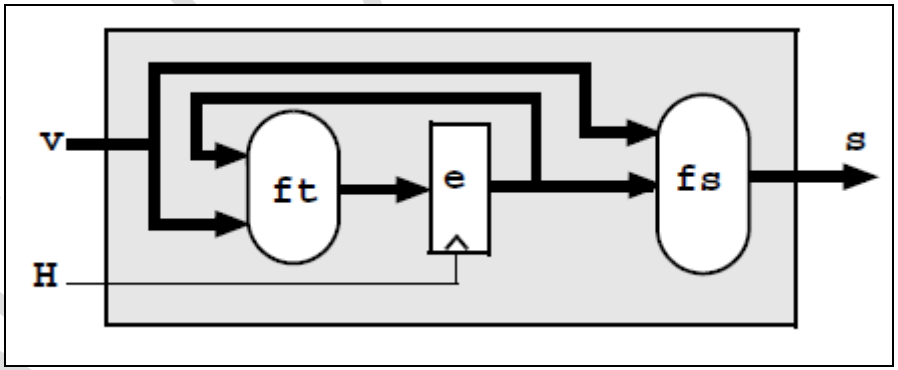
seul H provoque le changements d'état
convention : signal logique -top = 0↑1



- Modèle général : **Machine (automate) à états finis**

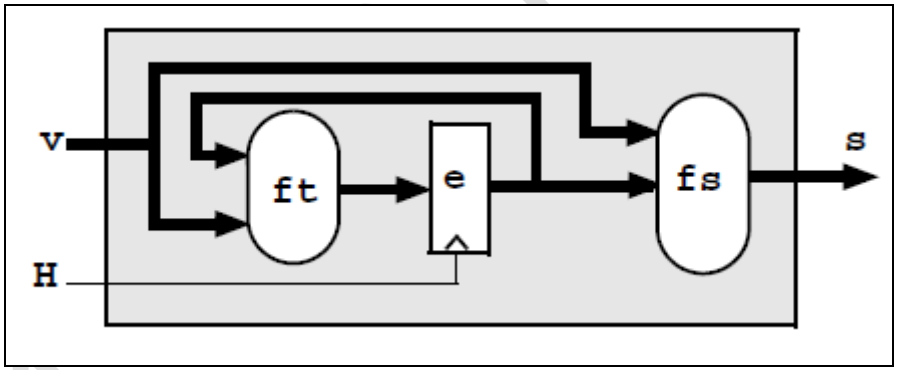
Rm : une partie (appelée « unité de contrôle ») du processeur de l'ordinateur est un machine à états finis

- ✓ Entrée de valeur v ∈ V
- ✓ Entrée d'horloge H : séquence d'évènements
- ✓ Etat interne e ∈ E (E fini)
- ✓ Sortie s ∈ S
- ✓ **Fonction de transition** :
  $\boldsymbol{ft}: ExV \rightarrow E$ *le futur état*
- ✓ **Fonction de sortie** :
  $\boldsymbol{fs}: ExV \rightarrow S$ *la sortie*



- Description machine à états finis :
  - ✓ Système séquentiel fonctionne par passage d'un état à un autre, et génère des **sorties** (des actions)
  - ✓ C'est la fonction de **transition** (« next-state logic circuit ») qui détermine vers quel état aller (le prochain état, l'état suivant), en se basant sur l'état courant, et les entrées.
  - ✓ La **fonction de transition** est **appliquée à l'état interne** (elle change l'état courant) afin de **trouver l'état suivant** (et ceci aux moments des tops d'horloge).
  - ✓ Analyser un système séquentiel consiste donc à déterminer la **fonction de transition** et la **fonction de sortie** de façon à pouvoir **prédire le comportement du système**.

- ✓ Entrée de valeur v ∈ V
- ✓ Entrée d'horloge H : séquence d'évènements
- ✓ Etat interne e ∈ E (E fini)
- ✓ Sortie s ∈ S
- ✓ **Fonction de transition** :
  $\boldsymbol{ft}: ExV \rightarrow E\ le\ futur\ état$
- ✓ **Fonction de sortie** :
  $\boldsymbol{fs}: ExV \rightarrow S\ la\ sortie$



ft et **fs** sont des circuits combinatoires

e : mémoire stockant l'état interne du système

- Description machine à états finis :
  - ✓ Système séquentiel fonctionne par passage d'un état à un autre, et génère des **sorties** (des actions)
  - ✓ C'est la fonction de **transition** (« next-state logic circuit ») qui détermine vers quel état aller (le prochain état, l'état suivant), en se basant sur l'état courant, et les entrées.
  - ✓ La **fonction de transition** est **appliquée à l'état interne** (elle change l'état courant) afin de **trouver l'état suivant** (et ceci aux moments des tops d'horloge).
  - ✓ Analyser un système séquentiel consiste donc à déterminer la **fonction de transition** et la **fonction de sortie** de façon à pouvoir **prédire le comportement du système**.

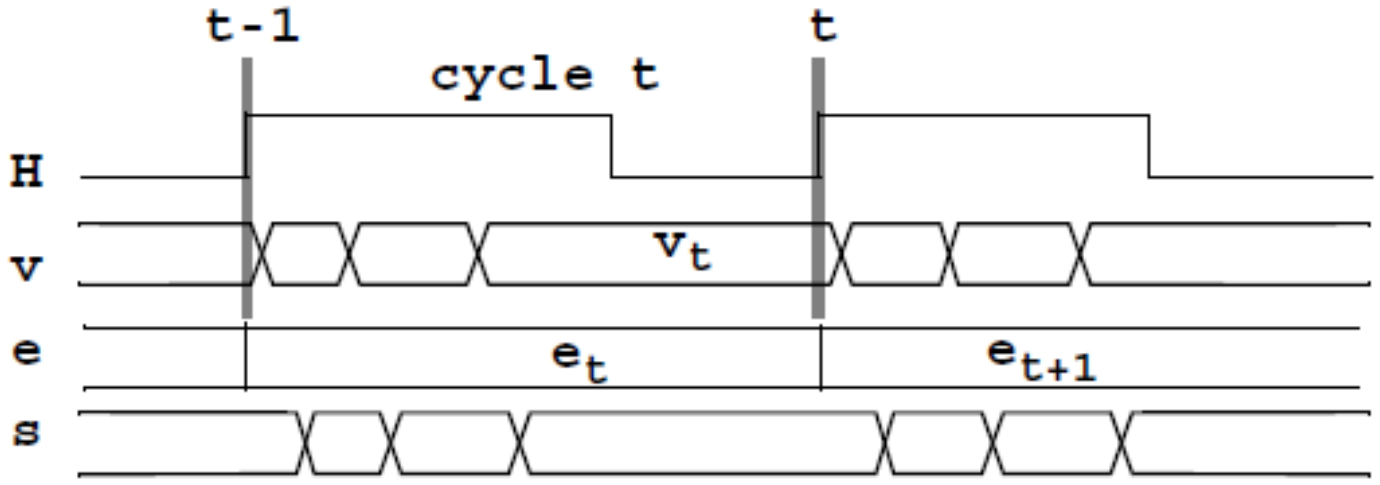
- Cycle t = intervalle entre le top, d'horloge t-1 et le top t

  e_t : état au cycle t
  v_t : entrée au top t

- Deux notations possibles :

  ✓ *Notation 1* : $\boldsymbol{s_t = fs(e_t, V_t)}$ $\quad$ $\boldsymbol{e_{t+1} = f_t(e_t, v_t)}$ ***(ou encore*** $\boldsymbol{e^+ = f_t(e_t, v_t)}$***)***

  ✓ *Notation 2* : $\boldsymbol{s = fs(e, V)}$ $\quad$ $\boldsymbol{e := f_t(e, v)}$



1.2. Représentation

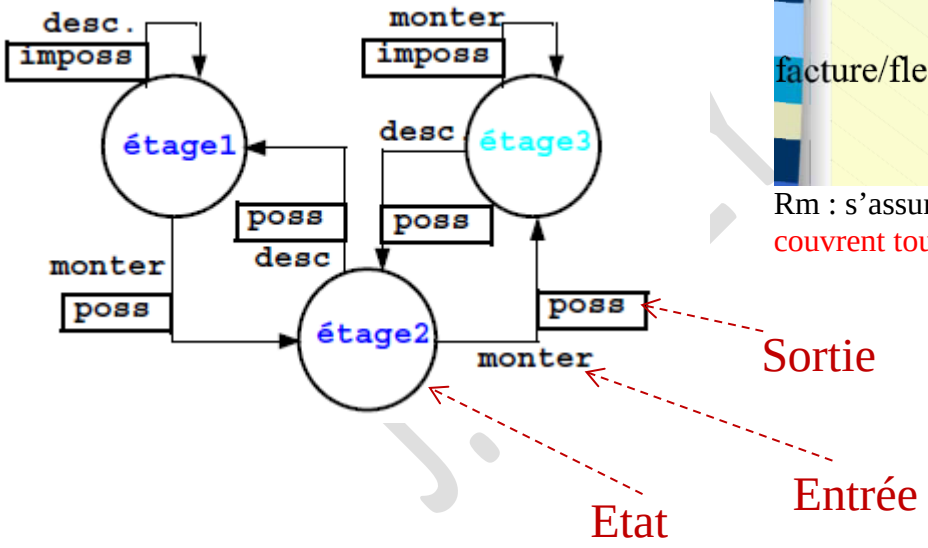
La spécification du comportement d'un système séquentiel est faite à partir de deux types de représentations : la table des états et le graphe des états.

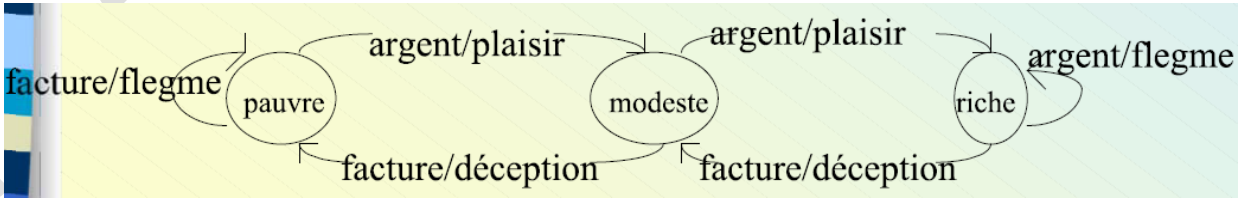
- Table des états

v	e	ft	fs
monter	étage1	étage2	possible
monter	étage2	étage3	possible
monter	étage3	étage3	imposs
desc	étage1	étage1	imposs
desc	étage2	étage1	possible
desc	étage3	étage2	possible

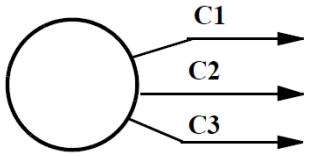
V	E	$f_T(e_t, v_t)$.	$f_s(e_t, v_t)$
facture	pauvre	pauvre	flegme
facture	modeste	pauvre	déception
facture	riche	modeste	déception
argent	pauvre	modeste	plaisir
argent	modeste	riche	plaisir
argent	riche	riche	flegme

- Diagramme des états





Rm : s'assurer que les conditions issues d'un même état sont mutuellement exclusives et couvrent tous les cas possibles :



$C_i \cdot C_j = 0$

$C1 + C2 + C3 = 1$

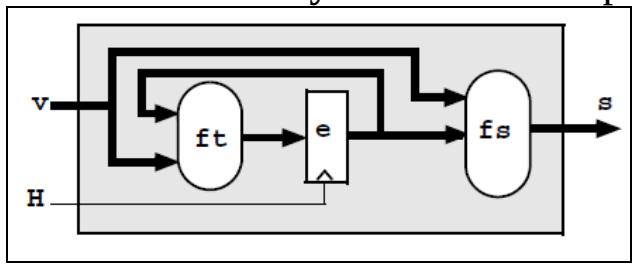
- Par langage de description

```
e:= si v=monter alors
        si e=ét1 alors
            ét2
        sinon
            ét3
    sinon
        si e=ét3 alors
            ét2
        sinon
            ét1
```

```
s = si (v=monter et e=ét3) ou (v=desc et e=ét1)alors
        imposs
    sinon
        poss
```

1.3. Modèles (Mealy, Moore)

- Machine de **Mealy** : c'est le cas le plus général



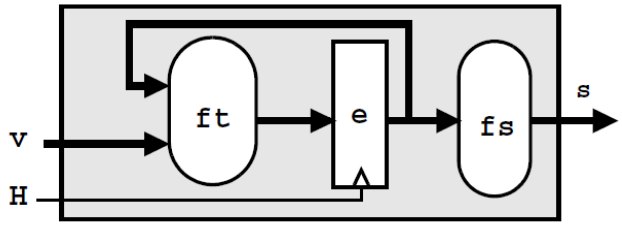
La sortie dépend des entrées et de l'état courant

$$e := ft(e, v)$$

$$s = fs(e, v)$$

Rm : L'opérateur := signifie « e prend sa nouvel valeur au prochain top d'horloge »

- Machine de **Moore** : cas particulier



La sortie ne dépend pas des entrées mais seulement de l'état courant

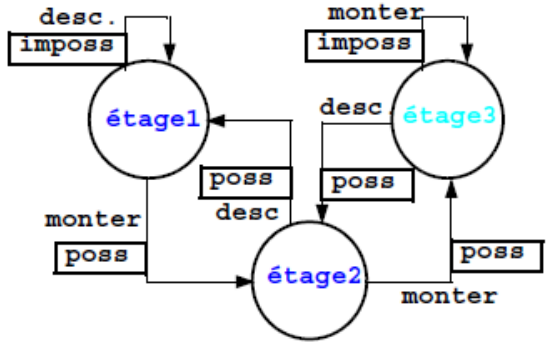
$$e := ft(e, v)$$

$$s = fs(e)$$

- 2 façons de modéliser le système de l'ascenseur:

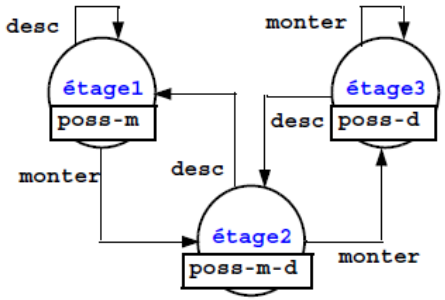
Mealy : 1 fonction de sortie (« possible »)

- ✓ Dépend de **l'entrée et de l'état**
- ✓ **Représentée sur les flèches** du graphe



Moore : 2 fonctions de sortie (« possibilité de monter» « possibilité de descendre »)

- ✓ Ne dépendent que de l'**état**
- ✓ **Réprésentées sur les états** du graphe



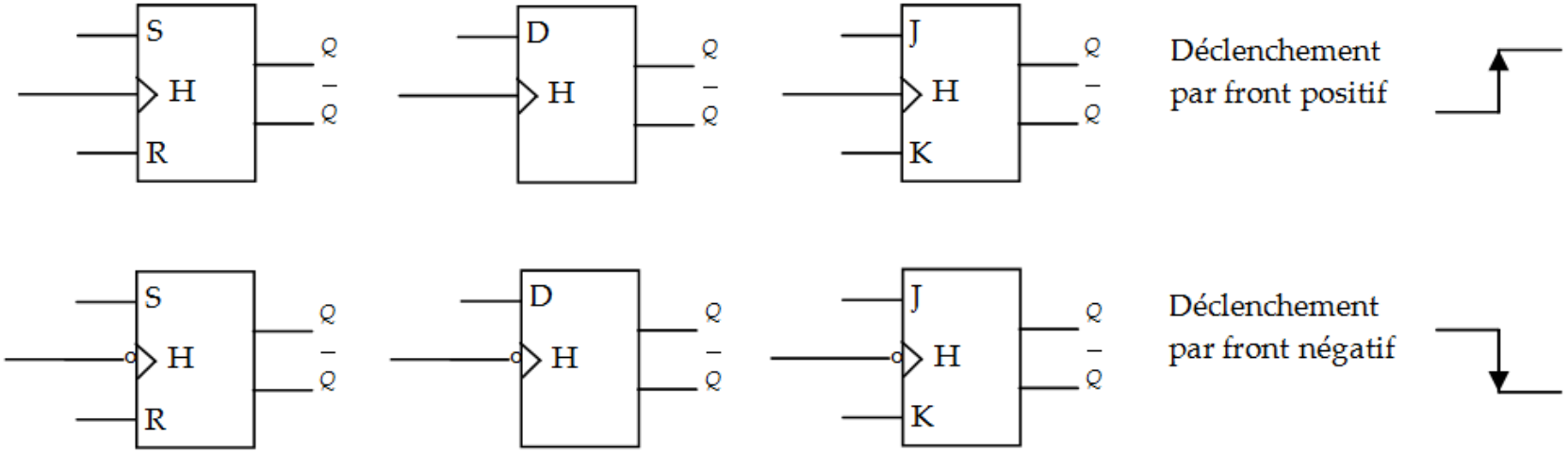
Sauf contraintes de type physique spécifiées dans le cahier des charges, **tout problème peut être résolu par une machine de type Mealy ou Moore, indifféremment.**

2. EXEMPLES DE CIRCUITS SEQUENTIELS

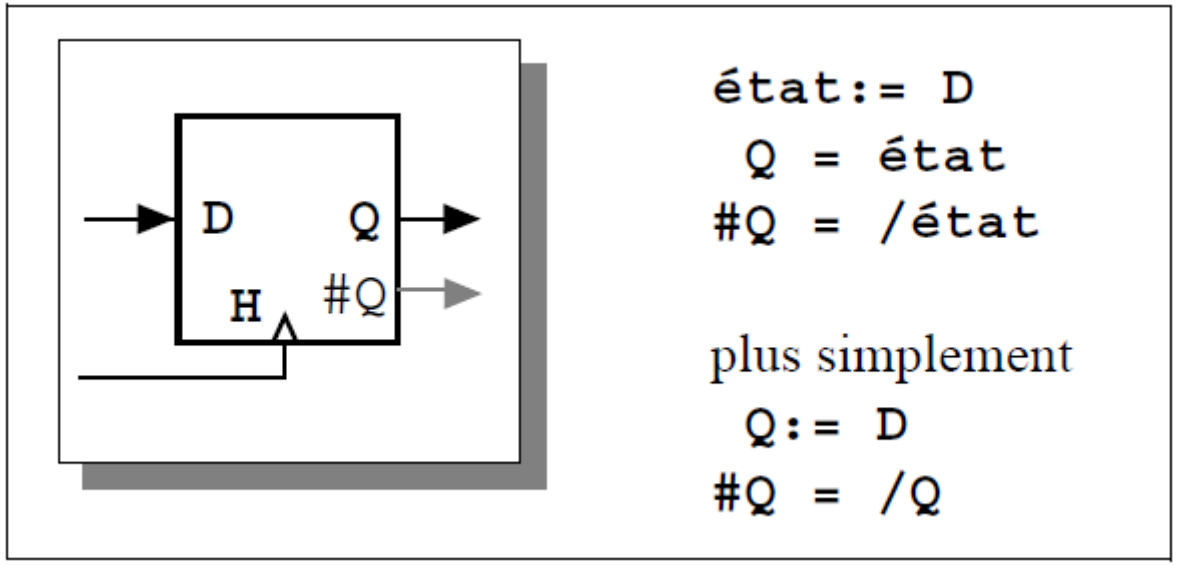
2.1. Bascules (Fonction mémorisation)

- Eléments bistables synchrones
  - ✓ entrée signal d'**horloge** H (signal carré)
  - ✓ **entrées de commande** (RS, JK, D)
  - ✓ changements sortie synchronisés par H
  - ✓ Etat ∈ {0,1}
  - ✓ **Sortie Q : directement l'état**
  - ✓ Parfois une sortie $\bar{Q}$

- Symboles bascules RS, JK, D



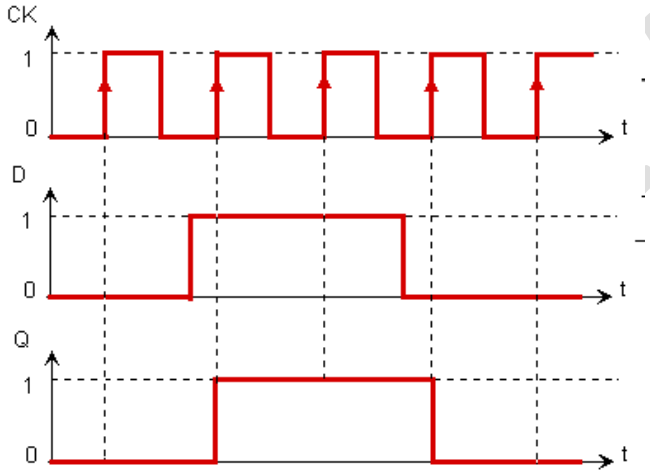
- Bascule D :
  - ✓ simple et utile (mémoire 1 bit) :



Au top d'horloge, la sortie prend la valeur de l'entrée

Q :=D *(parfois noté* $Q^+=D$*, si on note* Q^+ *l'état suivant)*

✓Chronogramme et schéma interne



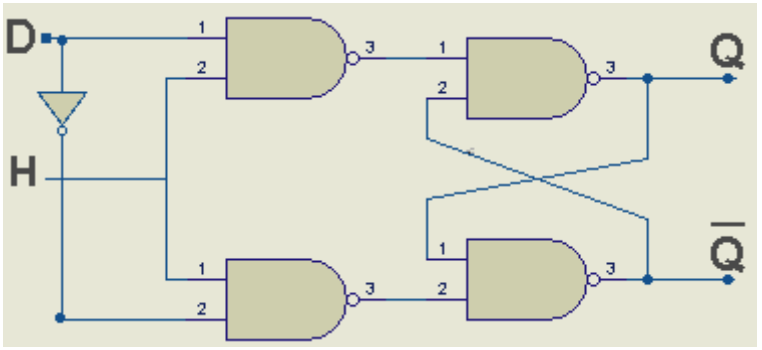
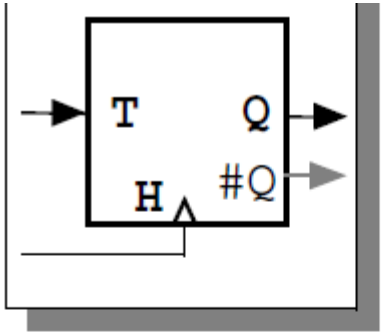


Schéma interne d'une bascule D (avec des portes NAND)

- Bascules T, JK, RS :

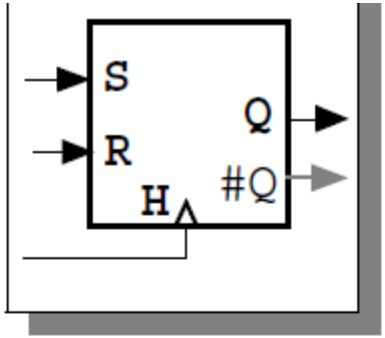


T	Q_{t+1}
0	Q
1	/Q

Q := si T alors /Q
sinon Q

ou encore

Q := /T•Q + T•/Q

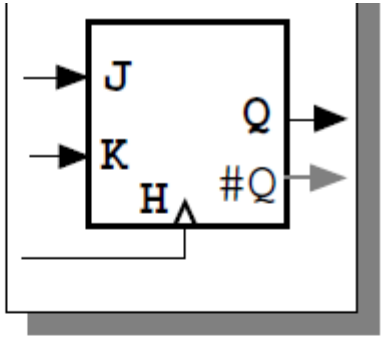


S	R	Q_{t+1}
0	0	Q
0	1	0
1	0	1

Q := cas RS
soit 00 : Q
soit 01 : 1
soit 10 : 0

ou encore

Q := S + /R•Q
avec RS≠11

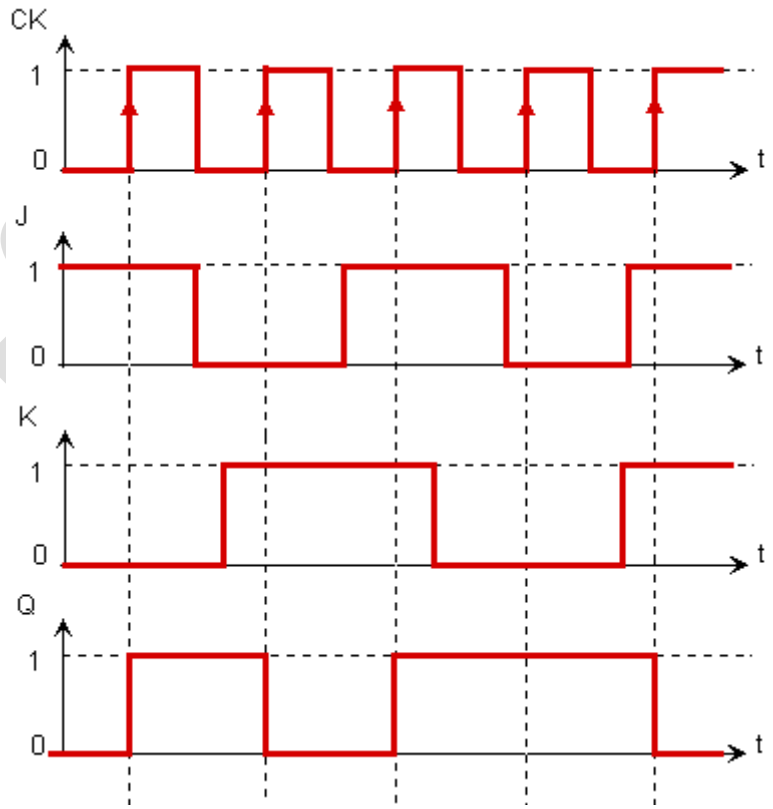


J	K	Q_{t+1}
0	0	Q
0	1	0
1	0	1
1	1	/Q

Q := cas KJ
soit 00 : Q
soit 01 : 1
soit 10 : 0
soit 11 : /Q

ou encore

Q := J•/Q + /K•Q



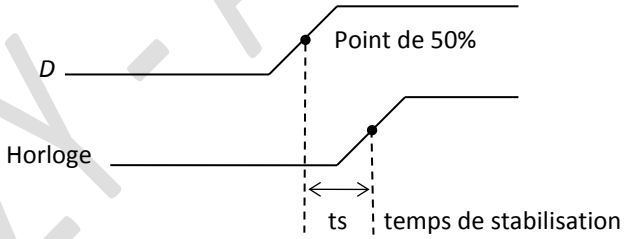
Chronogramme bascule JK

- Contraintes temporelles

  - **Fonctionnement correct** d'un système séquentiel réalisé à base de bascules si respect des contraintes liées aux **délais internes des bascules** :

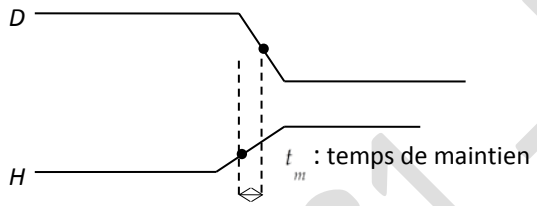
    ✓ Temps de stabilisation (ts)
    Intervalle minimum pendant lequel les niveaux logiques des entrées doivent être maintenus de façon constante (J et K, ou S et R, ou D) avant le front déclencheur du signal d'horloge, de façon à garantir un déclenchement fiable de la bascule.

    Exemple pour une bascule D :



✓ Temps de maintien

Intervalle minimum pendant lequel les niveaux logiques des entrées doivent être maintenus de façon constante après le front déclencheur du signal d'horloge, de façon à garantir un déclenchement fiable de la bascule.



✓ Fréquence de synchronisation maximale

fmax : vitesse la plus élevée du signal d'horloge, à laquelle une bascule peut être déclenchée de façon fiable.

- **Conséquences pour le calcul de l'état futur et des sorties du système séquentiel réalisé à partir de bascules**

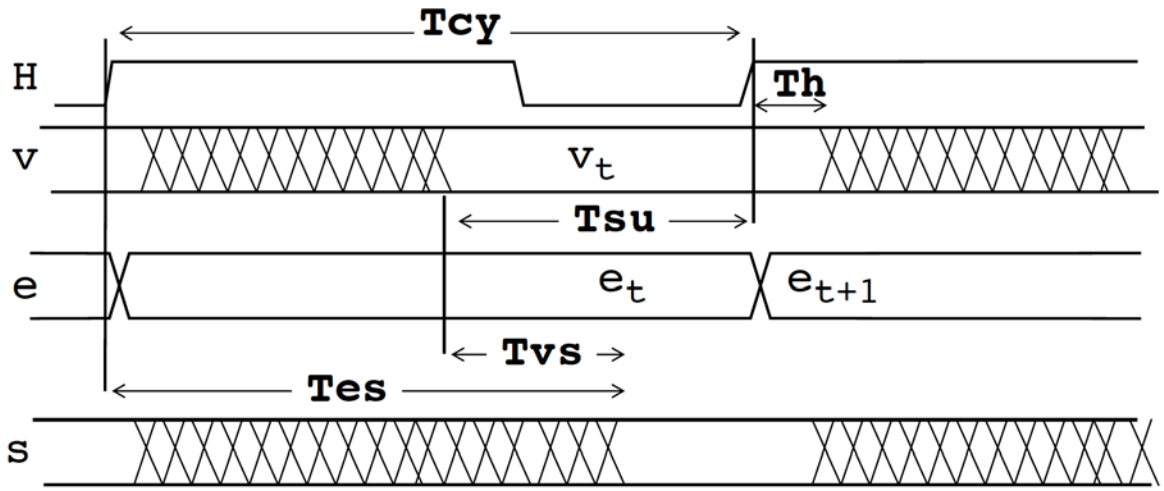


Schéma issu du Cours S. Derrien – ISTIC – Université de Rennes 1

$\mathbf{T_{su}}$: temps de calcul de l'état futur à partie des entrées

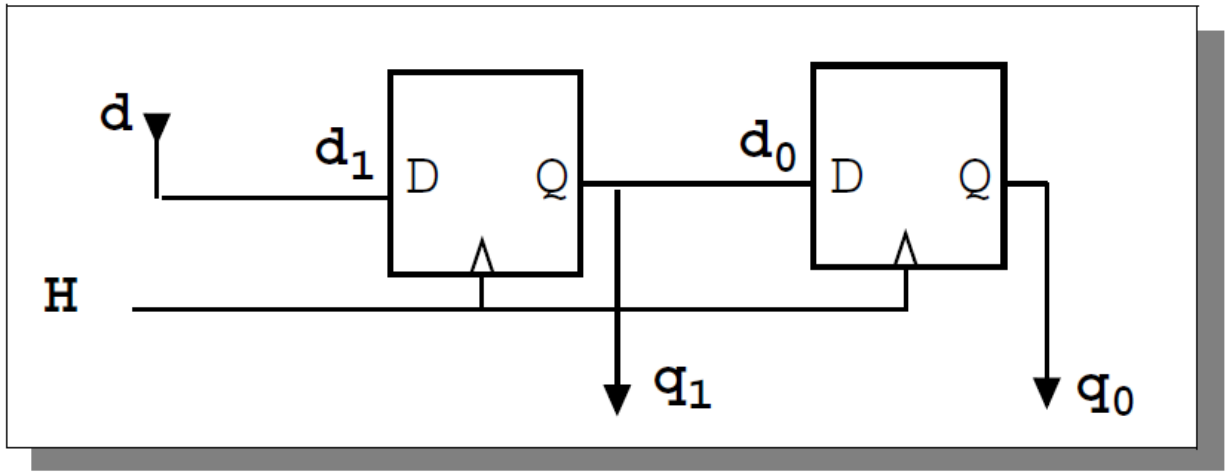
$\mathbf{T_{cy}}$: temps de calcul de l'état futur à partir de l'état présent

$\mathbf{T_{vs}}$: temps de calcul des sorties à partir des entrées

$\mathbf{T_{es}}$: temps de calcul des sorties à partir de l'état

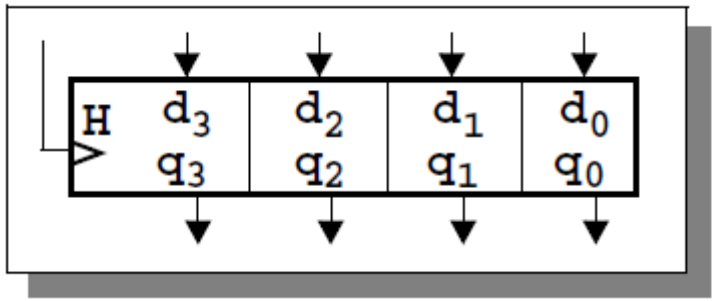
2.2. Registres (Fonction stockage, décalage)

- Registre = élément synchrone car assemblage de bascules synchronisées sur le même signal d'horloge H :



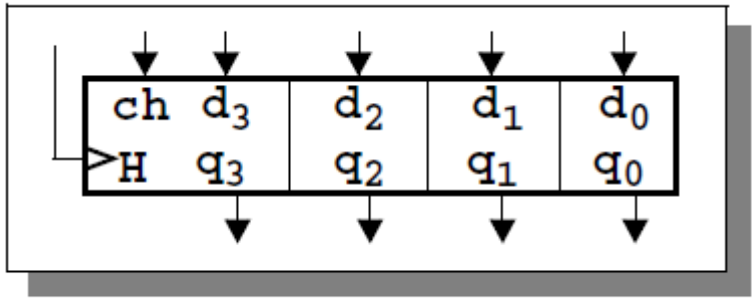
q1 :=d1
q0 :=d0
Connexions → d1=d, d0=q1
q1:=d
q0:=q1

- Registre à **chargement systématique** (pure mémoire n bits) :



$q_{3-0} := d_{3-0}$

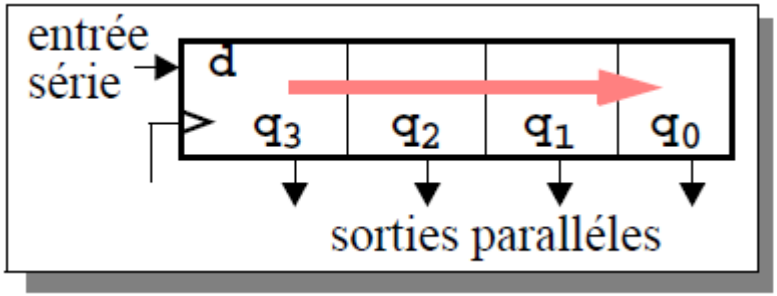
- Registre à **chargement commandé** (commande de chargement ch) :



ch	Q_{t+1}
0	Q
1	D

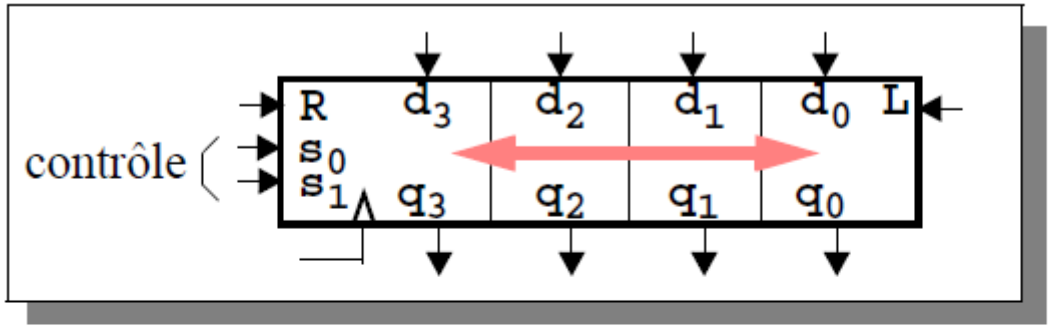
Q := **si** ch **alors** D
sinon Q

- Registre à décalage systématique :



$q_3q_2q_1q_0 := dq_3q_2q_1$

- Registre à **décalage droite, gauche et chargement** :

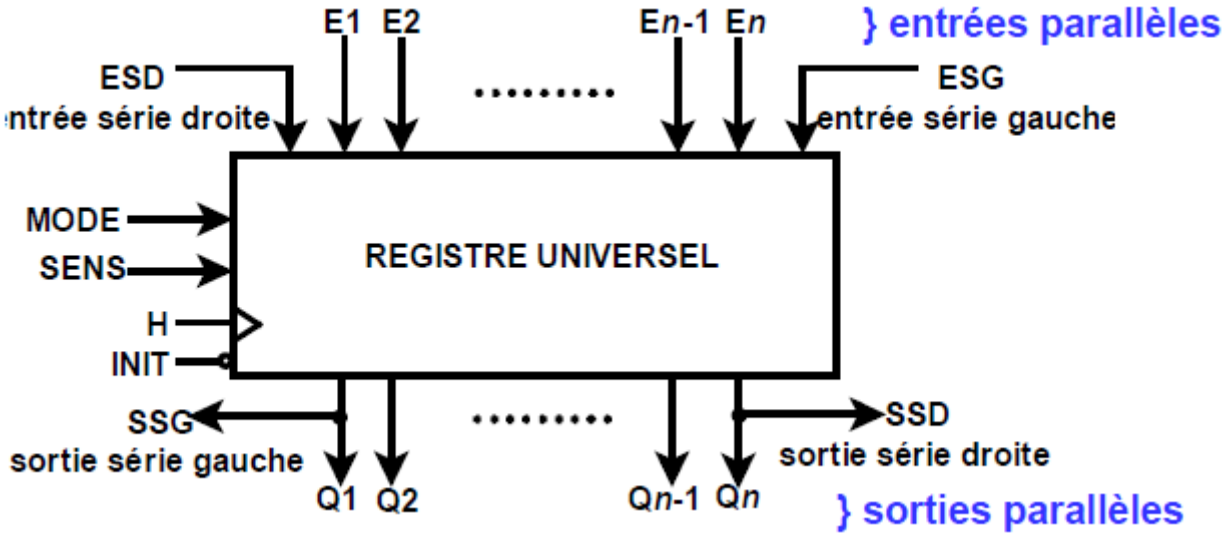


$q_3q_2q_1q_0$:= **cas** s_1s_0

soit 00 : $q_3q_2q_1q_0$	état inchangé
soit 01 : R $q_3q_2q_1$	décalage droite
soit 10 : $q_2q_1q_0$ L	décalage gauche
soit 11 : $d_3d_2d_1d_0$	chargement

- Registre universel :

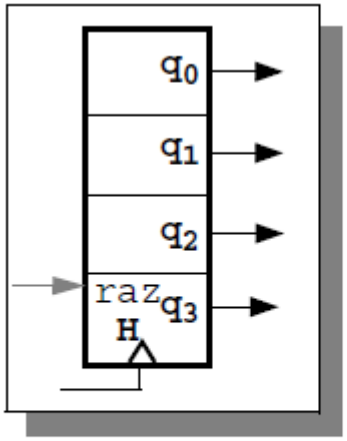
  ✓ Chargement Série ou parallèle (MODE, ESD, ESG, ou Ei)
  ✓ Décalage à droite et à gauche (SENS)
  ✓ Lecture série ou parallèle (SSG, SSD, ou Qi)
  ✓ Initialisation (INIT)

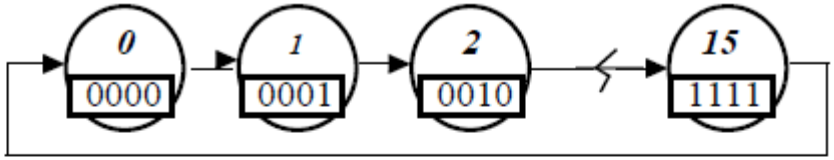


2.3. Compteurs (Fonction comptage)

- Compteur=circuit dont les sorties sont directement liées au nombre d'impulsions appliquées sur son entrée d'horloge

- **Compteur binaire modulo n** (modulo n : n états de 0 à n-1)
  - ✓ **Entrée** : Horloge H
  - ✓ **Etat** : CPT$\in [0, 2^{n-1}]$
  - ✓ **Sorties** : $q_{n-1}\ldots q_0$ (représentation binaire de CPT)





$q_3q_2q_1q_0 = [\mathbf{CPT}]_2$

CPT := (**CPT** + 1) modulo 16

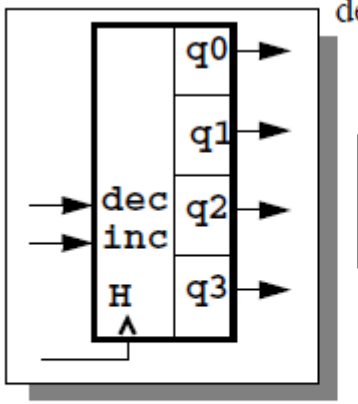
CPT_t	CPT_{t+1}	$q_3q_2q_1q_0$
0	*1*	0000
1	*2*	0001
2	*3*	0010
⋮	⋮	⋮
14	*15*	1110
15	*0*	1111

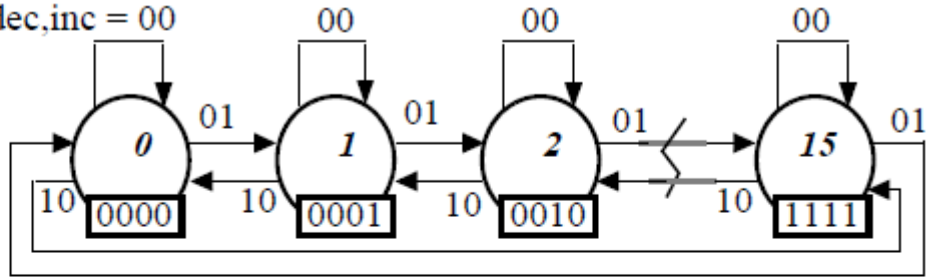
- ✓ Compteur avec RAZ (Remise A Zéro) : initialisation possible de l'état à 0

CPT : = si RAZ alors 0 sinon CPT+1 (modulo16)

- **Compteur – Décompteur**

✓ Entrées : Horloge H, inc (incrémentation), dec (décrémentation)

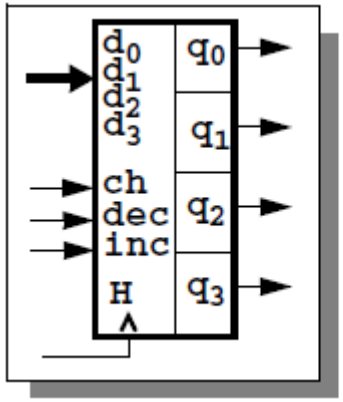




CPT := **cas** (**inc**, **dec**)
soit (0,0) : **CPT**
soit (0,1) : (**CPT**-1) mod 16
soit (1,0) : (**CPT**+1) mod 16

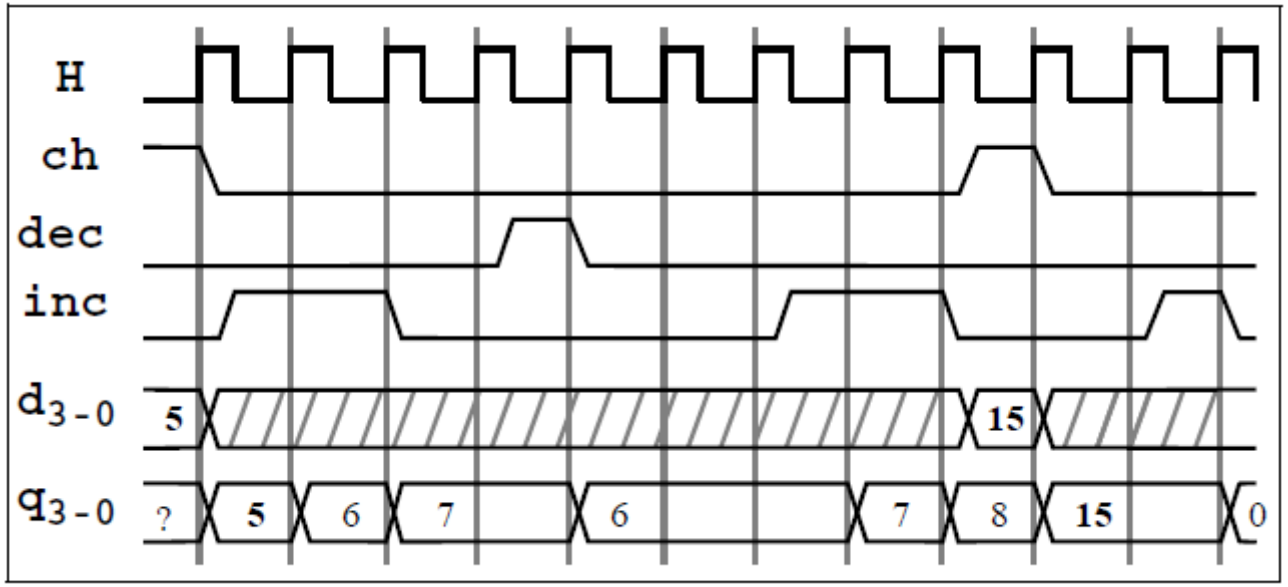
$q_3q_2q_1q_0 = [\mathbf{CPT}]_2$

✓ Avec commande chargement (ch)



CPT := **cas** (**inc**, **dec**, **ch**)
soit (0,0,0) : **CPT**
soit (0,0,1) : $_2[\mathbf{d_3d_2d_1d_0}]$
soit (0,1,0) : (**CPT**-1) mod 16
soit (1,0,0) : (**CPT**+1) mod 16

✓ Exemple de chronogramme illustrant le fonctionnement du compteur-décompteur :

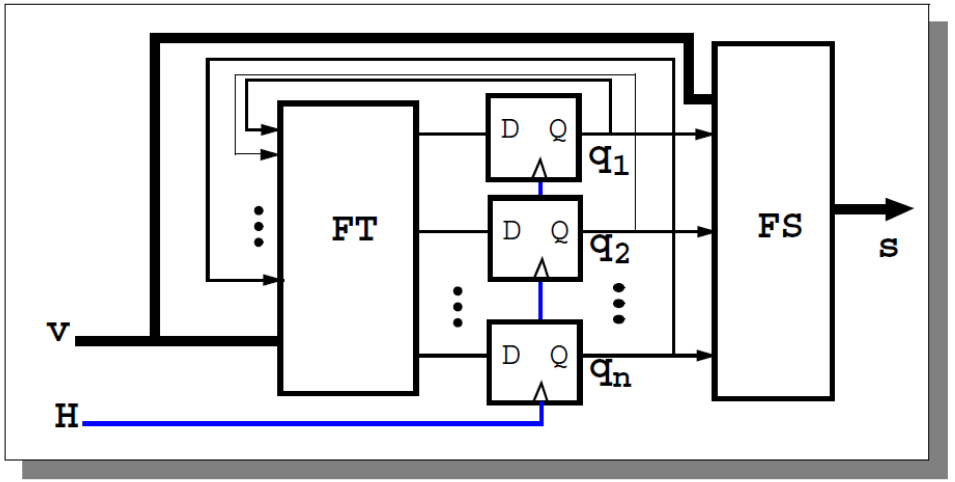


3. SYNTHESE (MISE EN ŒUVRE) D'UN CIRCUIT SEQUENTIEL SIMPLE AVEC BASCULES

- Système simple :
  - ✓ décrit facilement par un diagramme d'états ou tables
  - ✓ petits nombres d'états

3.1. Les étapes de la synthèse

- Coder les états à l'aide de bits $q_1 \ldots q_n$
- Rechercher FS et FT en tenant compte des codes
- Utiliser des bascules D (ex : une bascule pour chaque bit du code si codage binaire, une bascule par état si codage à un bit par état)
- Appliquer la fonction de transition FT(q_1, .. q_n, v) sur les entrées des bascules



3.2. Codage des états

Consiste à associer un code binaire à un nom d'état. Chaque état doit avoir un code différent des autres mais le codage des états peut être quelconque.
Par contre, le codage influence la structure du système, et peut donc jouer sur sa complexité.

a) Codage **binaire**

- Chaque numéro d'état est représenté par le **nombre binaire correspondant**
- Avantage : représentation intuitive
- Décodage de l'état nécessaire pour connaître l'état suivant
- Exemple pour un système à 4 états :

Etat	Codage binaire de l'état
Etat 1	001
Etat 2	010
Etat 3	011
Etat 4	100

b) Codage **Gray**

- Chaque numéro d'état est représenté par son **code Gray correspondant**
- Exemple pour un système à 4 états :

Etat	Codage binaire de l'état
Etat 1	001
Etat 2	011
Etat 3	010
Etat 4	100

- Décodage de l'état nécessaire pour connaître l'état suivant

c) Codage à **1 bit par état (« Machine à jeton »)**

- Un seul bit est à 1 pour chaque état
- Exemple pour un système à 4 états :
-

Etat	Codage Machine à jeton
Etat 1	000**1**
Etat 2	00**1**0
Etat 3	0**1**00
Etat 4	**1**000

- Une bascule est utilisée pour chaque état du système
- Nécessite plus de bascules lorsque le nombre d'états augmente
- Pas de décodage nécessaire pour connaître l'état suivant (il suffit de changer le bit à 1 d'une place vers la gauche)

d) Codage **incorporant les sorties**

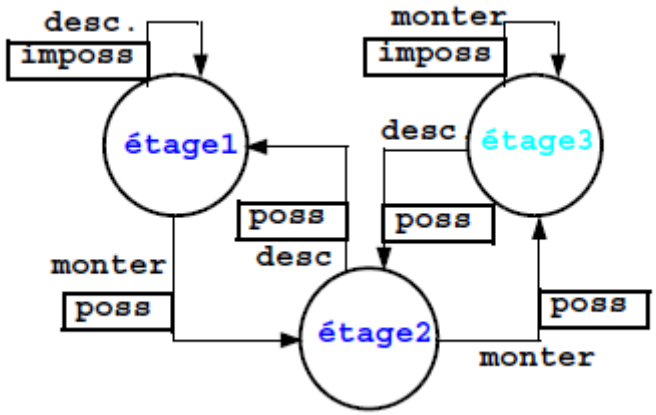
- Utiliser certaines sorties comme bits d'état
- En général : il faut compléter le codage par d'autres bits d'état

3.3. Exemple de synthèse avec **codage binaire**

1) Ascenseur version 1 (Modèle de Mealy) en utilisant le codage binaire (l'état est codé sur 2 bascules q_1, q_2 comme ci-dessous) :

	q1	q2
étage1	0	0
étage2	1	0
étage3	1	1

On rappelle le diagramme des états :



2) Idem en prévoyant une initialisation du système pour que celui-ci ne se retrouve pas dans un état illicite

1) Sans init

a) Fonction de **transition** :

FT v (monter : 1, desc : 0)

q_1 q_2	v = 0	v = 1
0 0	0 0	1 0
0 1	? ?	? ?
1 1	1 0	1 1
1 0	0 0	1 1

$q_1 := q_2 + v \quad q_2 := q_1 \bullet v$

Rm : v = entrée

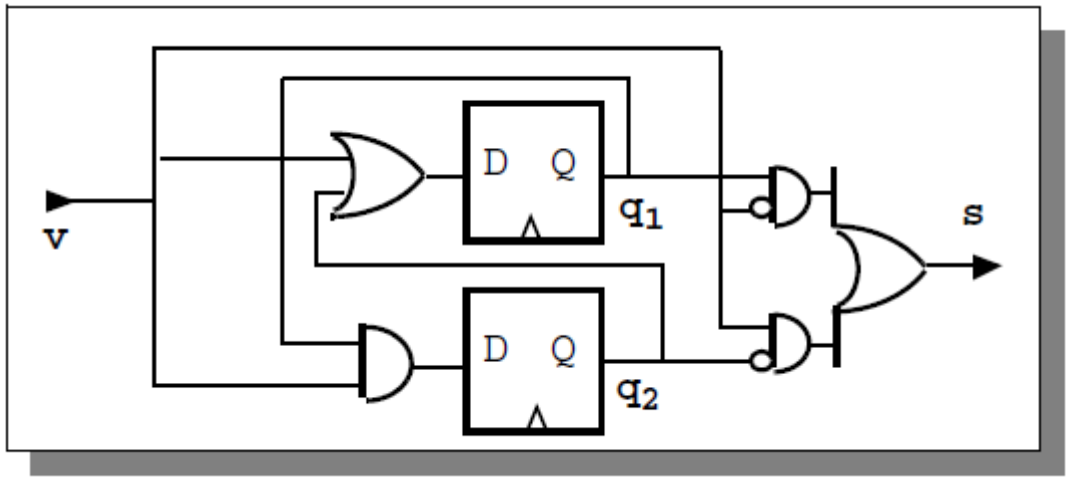
b) Fonction de **sortie** :

FS s (poss : 1, impossible : 0)

q_1 q_2	v = 0	v = 1
0 0	0	1
0 1	?	?
1 1	1	0
1 0	1	1

$s = q_1 \bullet /v + /q_2 \bullet v$

c) Circuit :



Explication circuit :

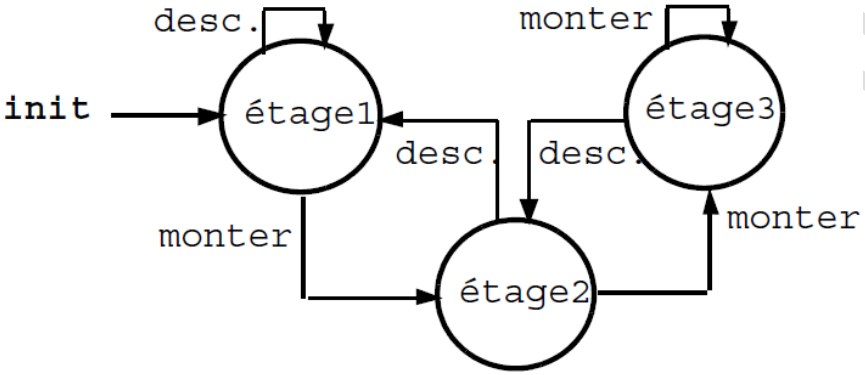
L'état est codé en binaire
2 bits donc 2 bascules

Comme q :=D pour une bascule D, on met :
q_2+v sur D1
q_1.v sur D2

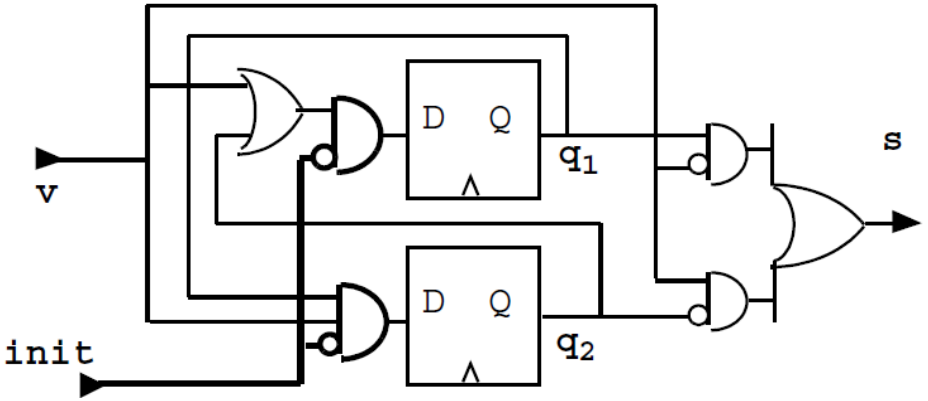
2) Avec init

- Certains états de bascules (codes) ne correspondent à aucun état physique (ex : 01 pour l'ascenseur)
- Prévoir alors une commande d'initialisation (init, reset) qui place le système dans un autre état que le ou les états illicites
- Exemple : pour l'ascenseur, l'état q1q0=01 est illicite donc la commande init place le circuit dans n'importe quel des autres états (par exemple dans l'état étage1 codé 00)

Nouveau graphe des états :

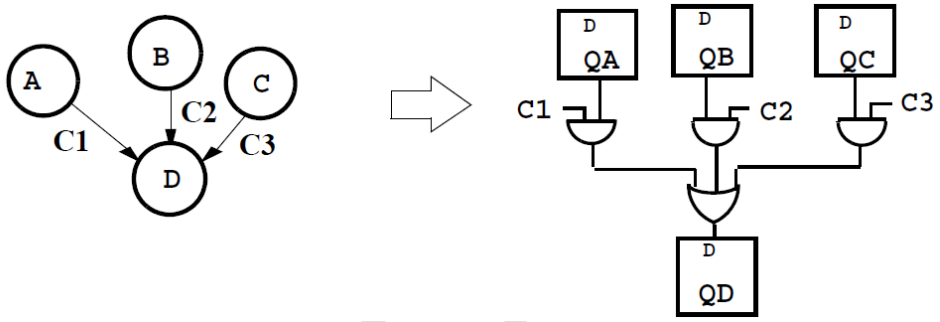


Nouveau circuit :

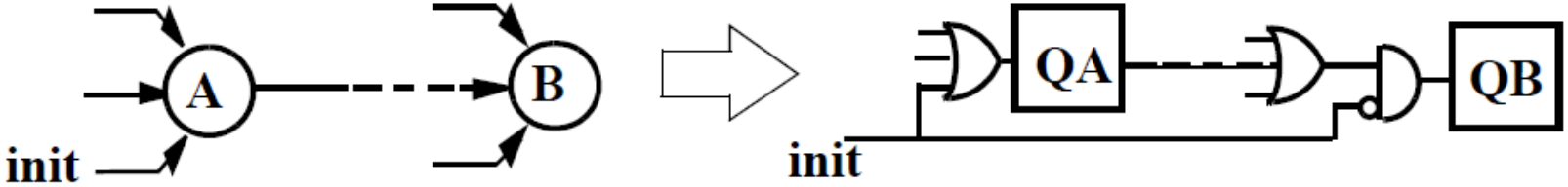


3.4. Exemple de synthèse avec **codage machine à jeton (1 bit par état)**

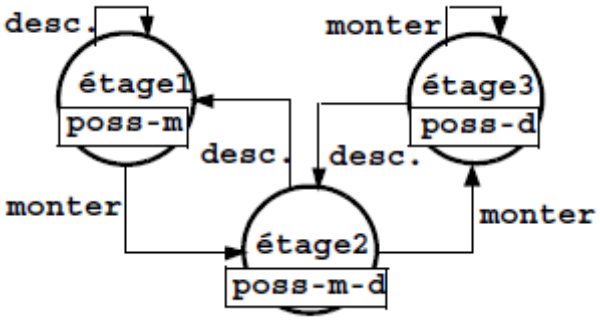
- Méthode : circuit calqué sur le diagramme des états (principal avantage) :

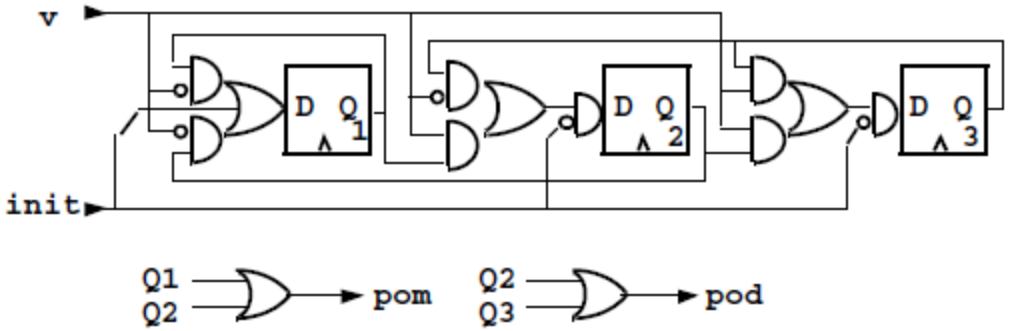


- Initialisation :



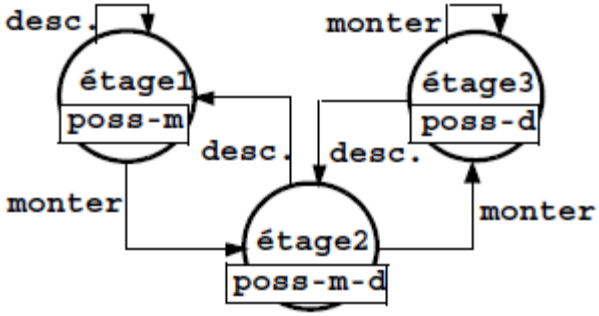
- Ascenseur version 2 (Moore) : traduction directe du graphe en circuit
- 2 fonctions de sorties : **poss-m** (possibilité de monter - vrai pour étage 1 et étage 2) **poss-d** (possibilité de descendre - vrai pour étage 2 et étage 3)
- Entrée : v (=1 pour monter, 0 pour descendre)





3.5. Exemple de synthèse avec **codage incorporant les sorties**

- Ascenseur version 2 (Moore) :
  Diagramme des états :

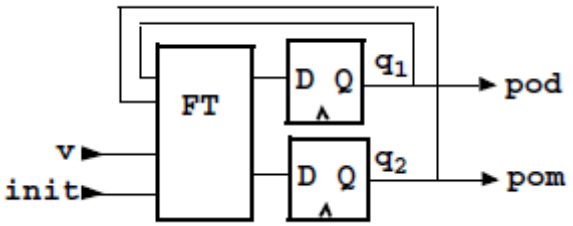


- *Sorties* :
  poss-m : possibilité de monter
  (vrai pour étage 1 et étage 2)
  poss-d : possibilité de descendre
  (vrai pour étage 2 et étage 3)
- *Entrée* :
  v (=1 pour monter, 0 pour descendre)

Chaque bit du code va être **défini à partir d'une sortie** : q1=poss-d, q2=poss-m.
Dans ce cas, 2 bits suffisent à différencier les 3 états, pas besoin d'ajouter un bit.

Etat	Sorties	Codage

	poss-d poss-m	q1(=poss-d) q2(=poss-m)
étage1	01	01
étage2	11	11
étage3	10	10



Avantage : les fonctions de sorties sont obtenues directement

Remarques sur la conception de circuits séquentiels :

- Difficulté principale : définition des états et des transitions (construction du graphe des états)
- Méthodes : cours de C. Wolinski (description du système sous forme de machine à états finis – FSM, construction du graphe des états, modification du graphe des états en fonction des contraintes liées aux éléments disponibles, optimisation du graphe des états)