Institut National des Sciences Appliquées de Toulouse
Département Sciences et Technologies Pour l'Ingénieur

Logique Séquentielle

Enseignants :
E. CHANTHERY
G. LE CORRE
A. SUBIAS

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Table des matières

UF Logique et structure des ordinateurs

Séquencement & Evaluations

- Logique séquentielle (G. Le Corre) (3 séances de CM/ 1TD)

Les enseignements de cette UF sont évalués par compétences.

- Un examen de séquentiel (1h15)

Objectifs du cours de logique séquentielle

Le cours de logique séquentielle vise un certain nombre de compétences. Parmi elles :

- savoir différencier un système combinatoire / séquentiel
- à partir d'un cahier des charges, savoir spécifier le fonctionnement du système (graphe, tables, chronogramme)
- connaître les bascules élémentaires (RS, JK, D, T) (prérequis)
- Savoir synthétiser un système séquentiel suivant la méthode d'Huffman
- Savoir faire l'analyse d'un système à partir de ses équations logiques

Seules deux compétences sont évaluées :

- Synthèse d'un système logique séquentiel
- Analyse d'un système logique séquentiel

Site moodle :

https://moodle.insa-toulouse.fr/course/view.php?id=1858

**Travail personnel attendu*

- Avant chaque cours, revoir au moins les illustrations des cours précédents (10mn).
- Avant chaque TD: relire les cours concernés (15mn)
- Après chaque TD: relire les exercices rédigés en séance (20mn)
- Avant l'examen, vérifier sous moodle « les points essentiels à connaître ».

Bibliographie

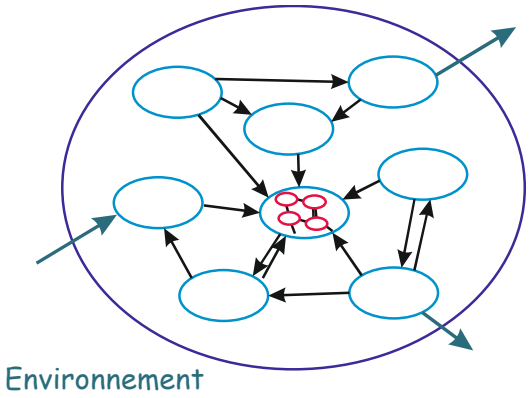
- Logique combinatoire et séquentielle par Brie Claude, Ellipses
- Logique combinatoire et séquentielle par Erceau Jean, Lagasse Jean
- Synthèse des systèmes logiques par Daclin Éric, Blanchard Michel, Csech Joseph
- Logique séquentielle par Gindre Marcel, Roux Denis

Chapitre 1

Principes des Systèmes Séquentiels

Introduction : notion de système, cadre du cours

Un système est un ensemble d'éléments dynamiques qui interagissent entre eux et avec leur environnement et qui sont organisés pour former un tout cohérent adapté à un certain objectif.



Notion d'entrée :

Les **entrées** du système peuvent être identifiées en trouvant comment l'usager du système peut agir dessus : ce sont des *entrées maîtrisées*.

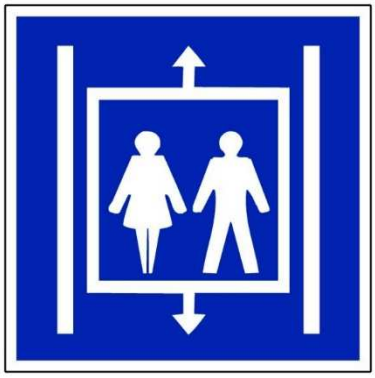
Certains systèmes subissent des perturbations (vent, frottement, etc) : ce seront des *entrées non maîtrisées.*

Notion de sortie :

Les **sorties** du système peuvent être identifiées en trouvant les grandeurs qui intéressent l'usager. On dit souvent que ce sont les grandeurs commandées, ou contrôlées.

On peut classer les systèmes suivant différents critères, par exemple

- Selon la nature des grandeurs à l'intérieur du système, qui peuvent être continues (comme la vitesse d'une voiture par exemple), ou bien discrètes (un interrupteur ON/OFF par exemple),
- Selon la nature du temps : un système peut être à temps continu ou à temps discret
- Selon la nature des variables : les variables sont soit déterministes (connues à chaque instant de manière précise), ou bien aléatoire (on parle alors en terme de probabilité).



Dans le cours, on traitera de systèmes à grandeurs discrètes, à temps discret, avec des variables déterministes.

Quelques exemples de systèmes traités dans ce cours : distributeurs de boissons, ascenseur, montre…

Système combinatoire ou systèmes séquentiel

Dans le cas des systèmes logiques, on appelle **système combinatoire** un système dont les valeurs de sortie ne dépendent que des entrées.

On appelle **système séquentiel** un système dont les sorties dépendent à la fois des entrées et d'un autre type de grandeur, appelé état, qui modélise une « mémorisation » de l'histoire du système.

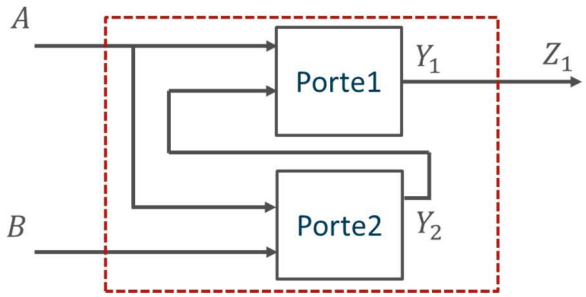
Système combinatoire (système statique) : sortie = fonction (entrées)
Système séquentiel (système dynamique) : sortie = fonction (entrées, état du système)

Etat, variables d'état

Pour décrire un système séquentiel, on a besoin d'un certain nombre de variables en plus des entrées, que l'on appelle **variables d'état**.
Les variables d'état peuvent être mesurées, et peuvent correspondre aux sorties du système
Les variables d'état peuvent aussi être uniquement internes au système.

Les valeurs des variables d'état à un instant donné constituent l'**état** du système.

Exemple :

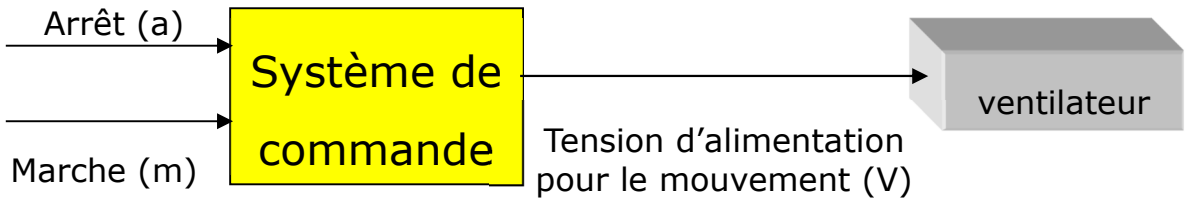


Etat stable

On dit qu'un système séquentiel est dans un état d'équilibre (ou **état stable**) si le système n'évolue pas si aucune entrée ne change. Cela signifie que l'**état suivant (ou état futur) du système est égal à l'état présent**.

1. Exemples de systèmes séquentiels

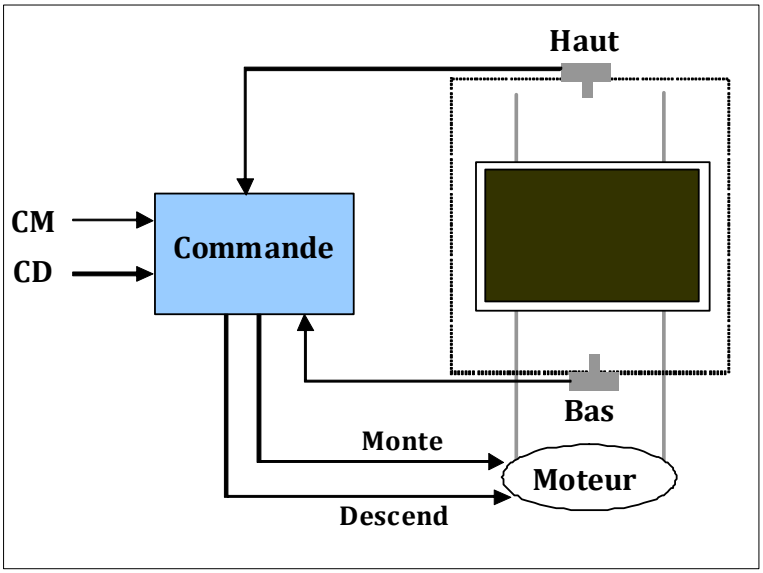
1.1. Système de commande d'un ventilateur



Règles de fonctionnement (cahier des charges) :

- Au repos le moteur est à l'arrêt, aucune des entrées n'est activée.
- Si l'on appuie sur m et que a n'est pas activé, le ventilateur se met en mouvement.
- Si l'on relâche m et que a n'est toujours pas activé, le ventilateur reste en mouvement.
- Le moteur étant en mouvement et aucune entrée n'étant activée, si l'on appuie sur a, alors que m est relâché, le ventilateur s'arrête.
- Si l'on relâche a, m étant toujours relâché le ventilateur reste à l'arrêt.
- Lorsque seul m (resp. a) est appuyé et que l'on appuie sur a (resp. m), le ventilateur reste (ou se met) en mouvement.

1.2. Système de commande de déplacement d'un tableau noir



Règles de fonctionnement

- capteurs Haut et Bas non considérés.
- initialement, le tableau est arrêté et les deux boutons CM et CD sont à 0.

Premier cahier des charges (A)

Si on appuie sur CM, le tableau monte tant que CM est appuyé
Si on appuie sur CD, le tableau descend tant que CD est appuyé
Si on appuie sur CM et CD, le tableau s'arrête

Second cahier des charges (B). La 3ème proposition de A est remplacée par :

Si on appuie sur les deux boutons CM et CD, le tableau s'arrête
Si ensuite on relâche CM ou CD, il reste arrêté

Troisième cahier des charges (C). La 3ème proposition de A est remplacée par :

Si on appuie sur CM et CD, le sens de déplacement reste inchangé
Si on relâche ensuite CM ou CD le sens reste inchangé
Si on relâche les deux boutons, le tableau s'arrête

CONCLUSIONS

- Importance du cahier des charges pour définir le caractère combinatoire ou séquentiel d'un système logique.
- Importance de la notion d'état pour mémoriser le passé d'un système séquentiel et en déduire son évolution.

2. Problématique

A partir du cahier des charges :

- Comment spécifier le fonctionnement du système?

➔ Graphes, tables, …

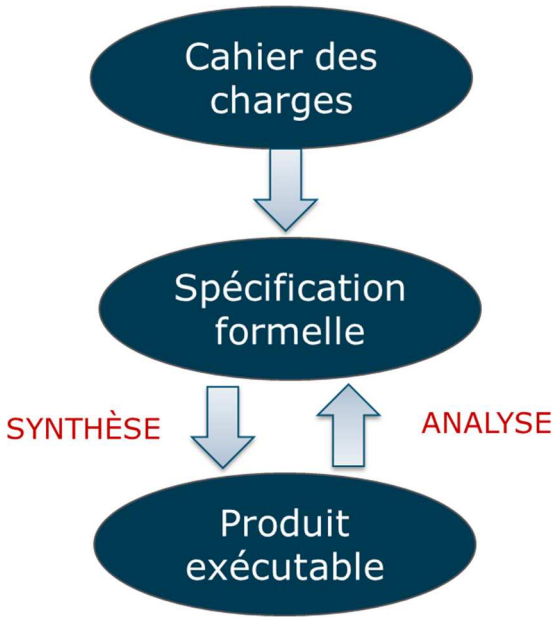
- Comment concevoir un modèle du système logique de commande ?

➔ Modèle d'états

- Comment faire la réalisation du système de commande ?

➔ A partir de portes logiques

➔ A partir de bascules élémentaires



3. Modélisation des systèmes logiques séquentiels (SLS)

Ce paragraphe introduit les principaux modèles de représentation des systèmes logiques séquentiels considérés d'un point de vue externe puis interne.
Il présente également quelques étapes et modèles importants de la conception logique de ces systèmes.

3.1. Vue Externe : modèle Entrées/Sorties

Ce sont des modèles externes de comportement, sans connaissance sur l'intérieur du système, pour lesquels on fait intervenir les entrées et les sorties du système.

Les **entrées** sont les grandeurs avec lesquelles l'usager interagit avec le système ainsi que des grandeurs « observées » ou « perçues » par le système. Il peut par exemple s'agir d'une valeur de consigne, de perturbations. Dans le cas des SLS, ce sont souvent des boutons poussoirs, des capteurs…

Les **sorties** sont les grandeurs qui intéressent l'usager (et non pas de la matière qui « sort » du système). Ce sont les grandeurs commandées ou contrôlées du système.

Lorsqu'on regarde le système de haut niveau, on peut le schématiser par ce qu'on appelle un **schéma bloc**. Cette représentation ne montre que les entrées et les sorties du système. Cette représentation peut représenter tous les types de systèmes.

Exemple sur le ventilateur :

3.1.1. Séquence des combinaisons entrées / sorties :

((e1,s1), (e2,s2), … , (eq, sq))

Exemple sur le ventilateur :
Cahier des charges :

- Au repos le moteur est à l'arrêt, aucune des entrées n'est activée.
- Si l'on appuie sur m et que a n'est pas activé, le ventilateur se met en mouvement.
- Si l'on relâche m et que a n'est toujours pas activé, le ventilateur reste en mouvement.
- Le moteur étant en mouvement et aucune entrée n'étant activée, si l'on appuie sur a, alors que m est relâché, le ventilateur s'arrête.
- Si l'on relâche a, m étant toujours relâché le ventilateur reste à l'arrêt.
- Lorsque seul m (resp. a) est appuyé et que l'on appuie sur a (resp. m), le ventilateur reste (ou se met) en mouvement.

3.1.2. Graphe d'état primitif: exemple du moteur

Le graphe d'état primitif représente le modèle de fonctionnement d'un système.
On distingue :

- des cercles que l'on appelle « **nœuds** », qui représentent les différents états du système
- des flèches appelées « **transitions** », qui représentent les transitions entre les états.
- On donne à côté du graphe la signification des notations : les entrées en haut, les sorties en bas.
- On distingue un état particulier qui est l'**état initial** (celui indiqué par une flèche seule)

Exemple sur le ventilateur :

Remarques et règles sur le graphe d'état primitif :

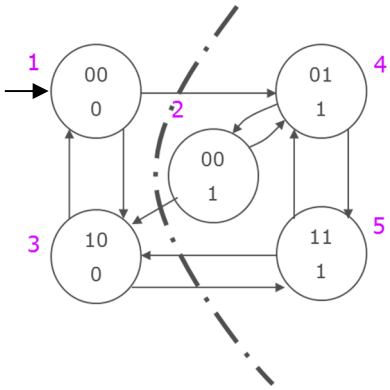
- Il est possible d'avoir deux états avec même entrées et mêmes sorties s'il existe des conditions internes différentes qui y mènent
- Il est impossible d'avoir deux états destination avec la même combinaison d'entrée (indéterminisme)
- Il est impossible de changer deux entrées en une seule transition
- Il y a autant de flèches en sortie d'un état que de nombre d'entrées (sauf si des combinaisons d'entrées sont interdites)

3.1.3. Graphe d'état réduit

On peut également représenter un graphe réduit aux valeurs des états. Pour construire un graphe d'état réduit, on fusionne les états dont les valeurs de l'état sont identiques.

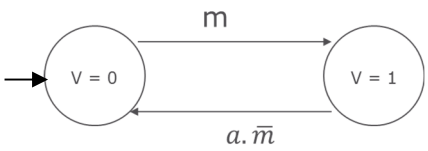
Dans le cas du ventilateur, on prend comme variable d'état la sortie V du ventilateur.

Graphe d'état primitif

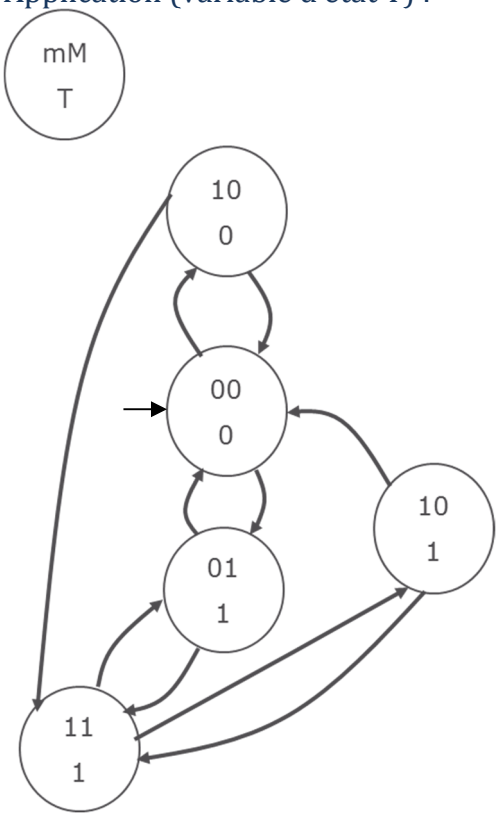


Graphe réduit (vers une vue interne)

regroupement d'états : (1, 2, 3) et (4, 5)

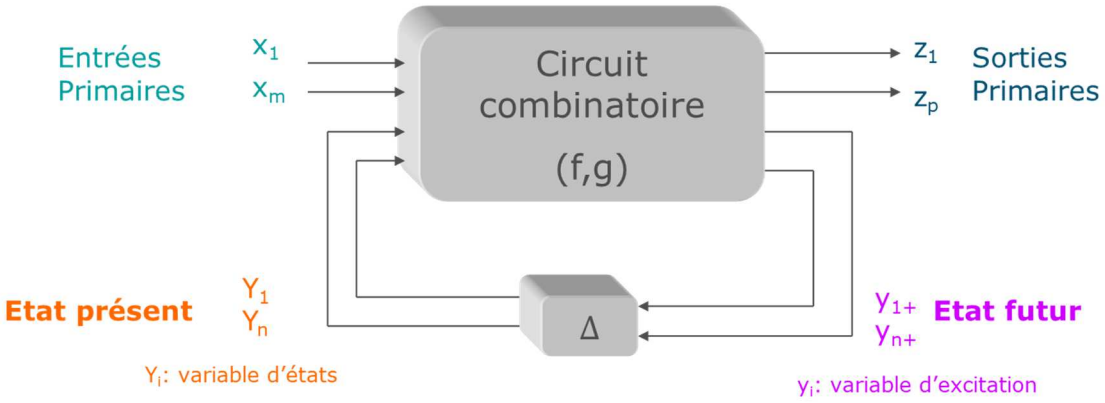


Application (variable d'état T) :



3.2. Vue interne : modèle d'états

3.2.1. Modèle d'Huffman



Etat présent : Yi : variables internes – variables d'états
Etat futur : yi+ : variables d'excitation
Δ: retard

Les variables de l'état total sont les entrées primaires et les états présents.

Equations d'évolution

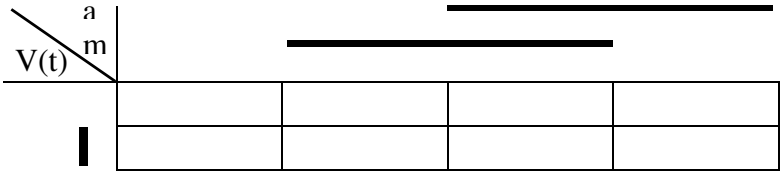
$$z = g(x, Y)$$ **Evolution des sorties**
$$y+ = f(x, Y)$$ **Evolution de l'état**

3.2.2. Illustration sur l'exemple du ventilateur:

L'état est donné par la sortie.
Table de vérité du comportement

a	m	V(t)	V(t+δt)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Cette table peut être mise sous forme d'une table de Karnaugh pour faire la synthèse du circuit:



$$\begin{cases} Y(t+\Delta t)=V(t+\delta t)=m+V(t).\bar{a} & \text{Equation d'évolution de l'état} \\ Z(t)=Y(t)=V(t) & \text{Equation de la sortie} \end{cases}$$

Représentation des états stables

On peut représenter sur la table de Karnaugh les états stables su système. Ce sont comme on l'a dit précédemment **les états pour lesquels l'état futur (à l'intérieur de la table) est égal à l'état présent (sur le côté gauche de la table)**. Par exemple, la première case du tableau en haut à gauche représente un état stable puisque $V(t) = V(t + \delta t) = 0$.
Par convention on entoure les états stables dans la table de Karnaugh.

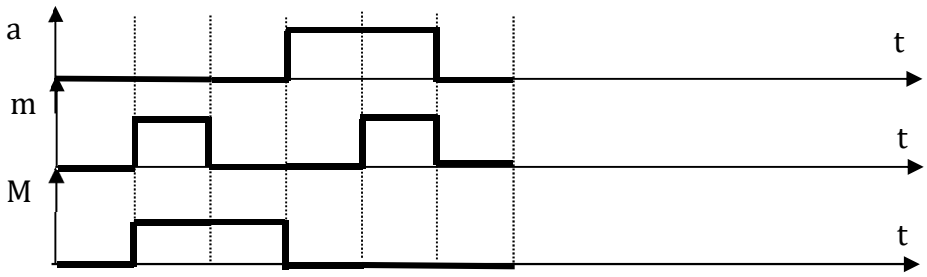
Application :

y_1y_2 \ AB	00	10	00	11
	01	01	11	11
	10	01	11	11
	00	10	00	-

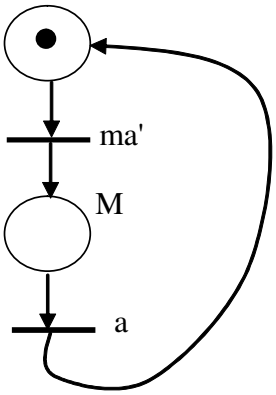
$y_1^+y_2^+$

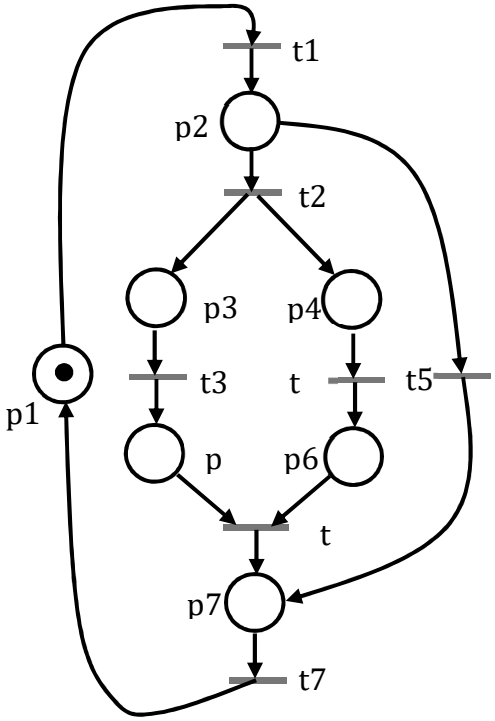
3.3. Autres représentations des systèmes logiques séquentiels (SLS)

3.3.1. Modèles temporels : chronogramme

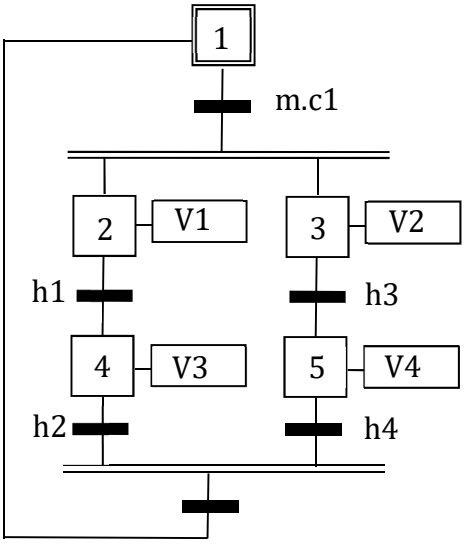


3.3.2. Réseaux de Petri

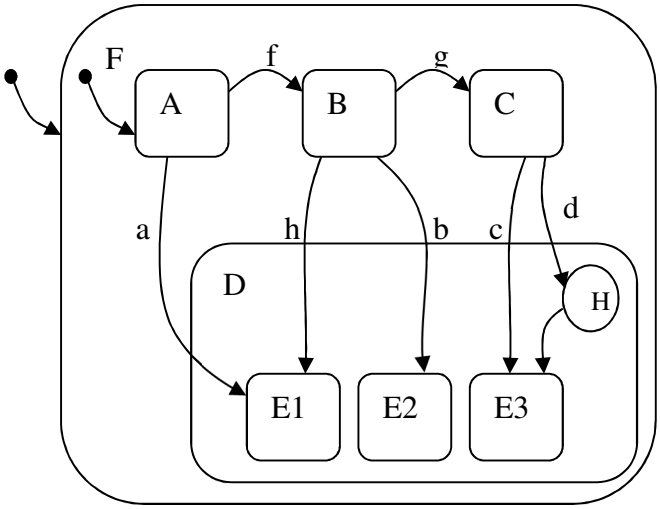




3.3.3. Grafcet



3.3.4. Statecharts



...plus de détails dans le cours de MCSED

4. Classification des systèmes logiques séquentiels (SLS)

Hypothèses générales : deux modes de fonctionnement

- Mode FONDAMENTAL Une machine séquentielle fonctionne en mode fondamental si et seulement si on suppose qu'elle dispose toujours d'un temps suffisamment long entre deux changements d'entrée pour se stabiliser.

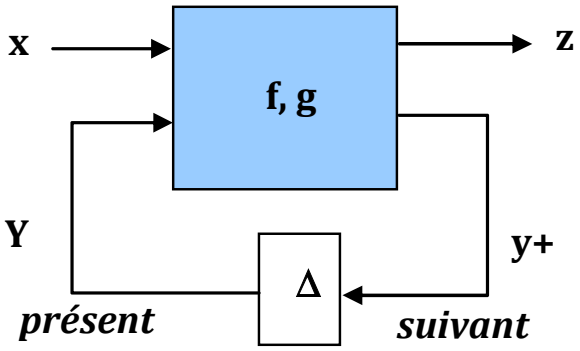
- Mode NORMAL Vecteurs d'entrée successifs adjacents

4.1. En fonction de la nature de l'évolution

- Evolution libre — Systèmes asynchrones
- Evolution contrôlée — systèmes synchrones (signal d'horloge pour la synchronisation)

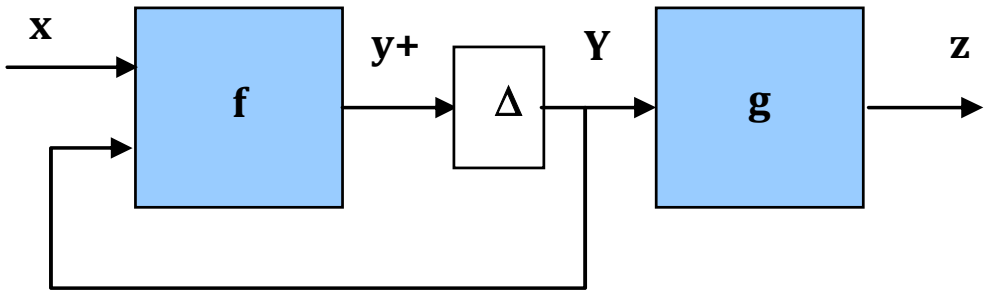
4.2. En fonction de l'élaboration des sorties

4.2.1. Cas général : Machines de MEALY: z = g (x, Y)



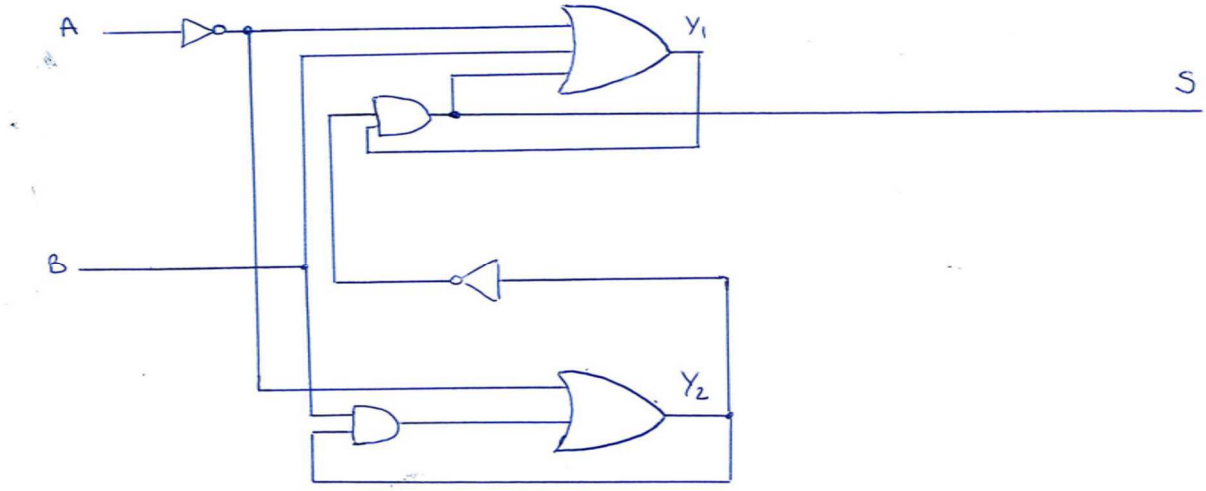
- **Modèle de Mealy :** les valeurs des sorties dépendent de l'état total

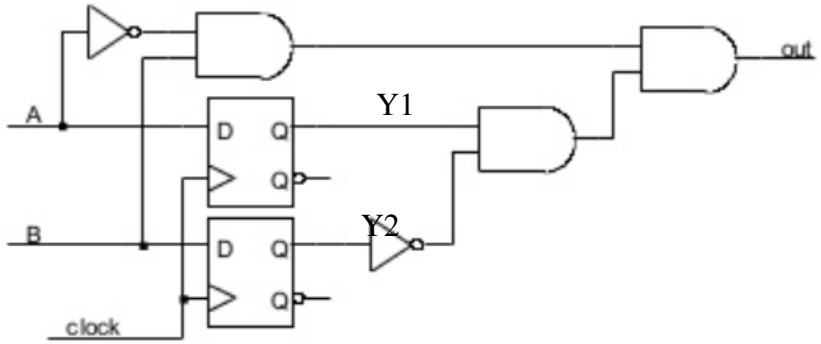
4.2.2. Cas particulier : Machines de MOORE : z = g (Y)

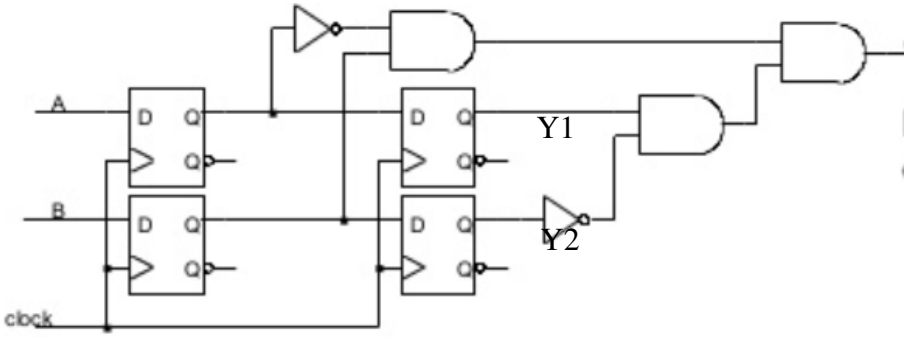


- **Modèle de Moore :** les valeurs des sorties ne dépendent que de l'état interne.

Exemples :







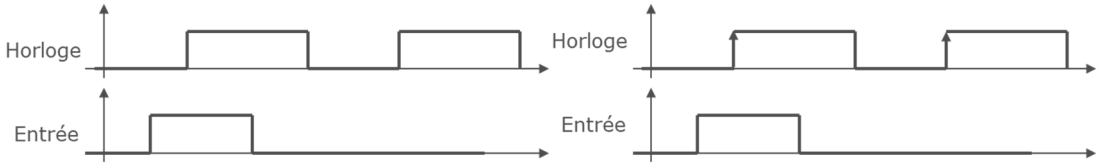
Chapitre 2

Systèmes Séquentiels Elémentaires

1. Introduction

Définition :
Un système est asynchrone si on le laisse évoluer par lui-même. Dans ce cas, la sortie peut être lue n'importe quand.
Dans le cas d'un système synchrone, une horloge cadence la marche du système, et on connaît les instants où l'on peut lire l'ensemble des sorties.
Dans le cas d'un système synchrone, la synchronisation peut se faire sur front (montant ou descendant) (Edge triggered) ou sur niveau.

Illustration :



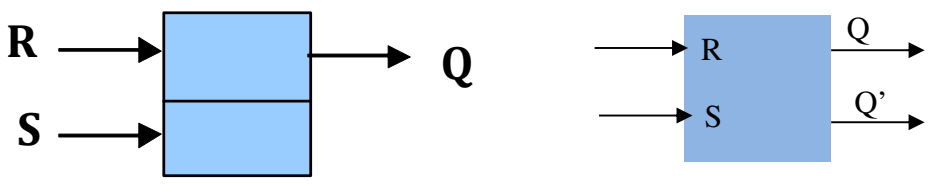
Notations :

Sur front montant Sur front descendant

2. Opérateurs séquentiels: bascules

2.1. La fonction RS : bascules RS associées

- Fonction mémoire unitaire
- 2 entrées R (Reset) et S (Set) - 1 sortie Q



Souvent on fait également apparaître une deuxième sortie $Q' = \bar{Q}$

- Fonctionnement :

R	S	Q	
0	0	Q	Mémorisation
0	1	1	Mise à 1
1	0	0	Mise à 0
1	1	-	-

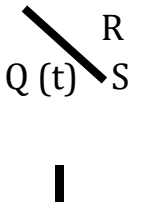
2 cas:

- Cas 1 : R = S = 1 ⇒ Q = 0 : fonction RS à effacement prioritaire
- Cas 2 : R = S = 1 ⇒ Q = 1 : fonction RS à inscription prioritaire

- Table de transition :

Q_i	→	Q_{i+1}	R	S
0	→	0	-	0
0	→	1	0	1
1	→	0	1	0
1	→	1	0	-

- Table de Karnaugh :



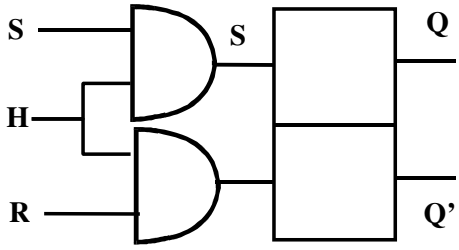
0	1	-	0
1	1	-	0

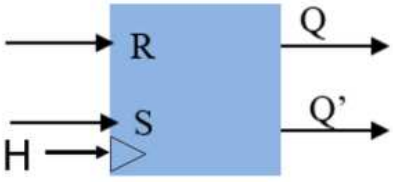
- Effacement prioritaire : Q= R'(S+Q) ⇒ Réalisation en NOR

- Inscription prioritaire : Q=S+R'Q ⇒ Réalisation en NAND

Remarque : le système de commande du moteur est une bascule RS à effacement prioritaire

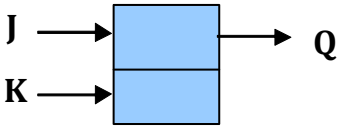
Cas de la bascule RS synchrone (notée parfois RSH) : on fait apparaître en plus l'horloge.

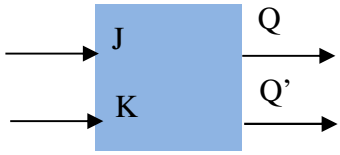




2 fonctionnements selon H : bascule verrouillée ou bascule active

2.2. La fonction JK : bascules JK associées





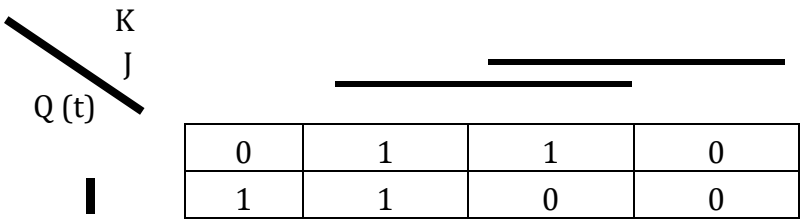
- Fonction mémoire unitaire avec séquence supplémentaire
- 2 entrées J (set) et K (reset) - 1 sortie Q (éventuellement Q')

- Fonctionnement :

J	K	Q	
0	0	**Q**	Mémorisation
0	1	**0**	Mise à 0
1	0	**1**	Mise à 1
1	1	**Q'**	Changement de valeur de sortie

- Table de transition :

Q_i	→	Q_{i+1}	J	K
0	→	0	0	-
0	→	1	1	-
1	→	0	-	1
1	→	1	-	0

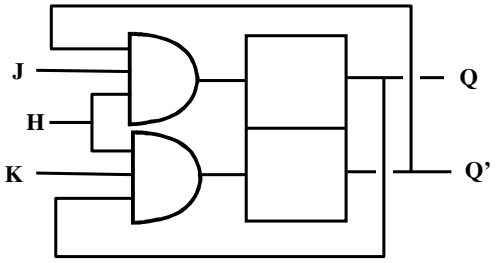


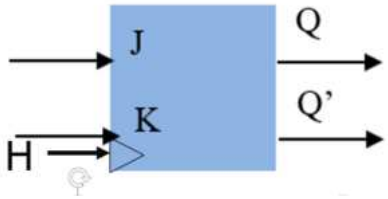
0	1	1	0
1	1	0	0

Q=K'Q+JQ'+K'J ⇒ Réalisation en NAND
ou réalisation à partir de bascule RS

S = JQ' et R = KQ

Attention : Si J = K = 1 : oscillations
-> Utilisation de JK synchrones

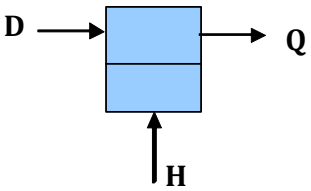




Bascule JK synchrone

2.3. La fonction D (appelée aussi latch) : bascule D associée

- Fonction suiveuse (D pour « data »)
- Une entrée D (Data), une entrée de synchronisation (H), une sortie Q



- Fonctionnement :

D	Q
0	0
1	1

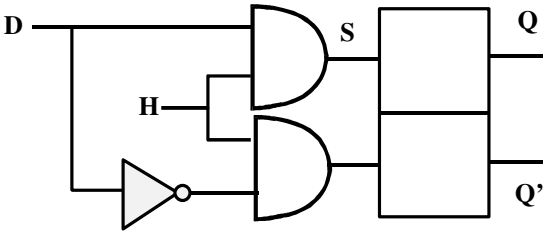
On constate sur la table de fonctionnement qu'un fonctionnement asynchrone n'aurait aucun intérêt. Avec une horloge, la sortie Q ne prend la valeur D que lorsque l'horloge H est à 1. On utilise donc forcément une bascule synchrone.

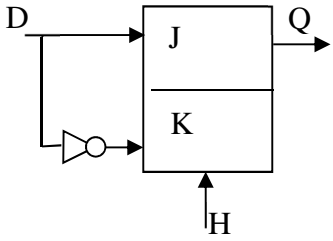
Table de transition :

Q_i	→	Q_{i+1}	H	D
0	→	0	0	-
1	→	1	0	-
-	→	0	1	0
-	→	1	1	1

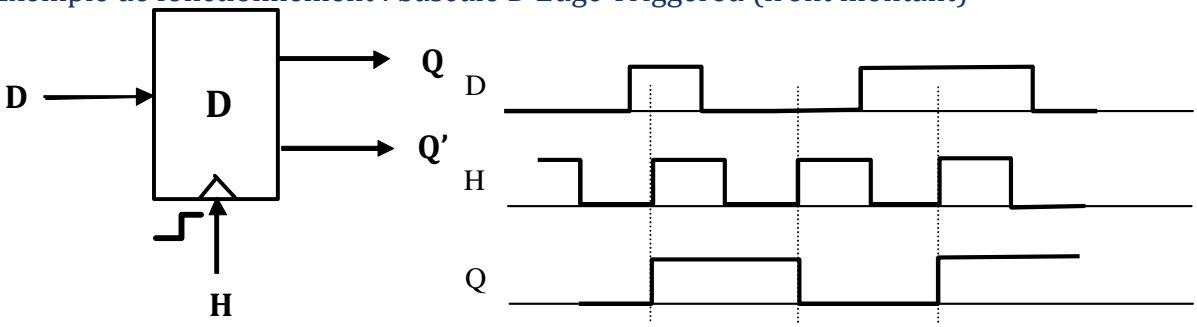
Q_i	→	Q_{i+1}	D
-	→	0	0
-	→	1	1

Exemples de réalisations



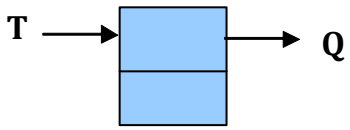


Exemple de fonctionnement : bascule D Edge Triggered (front montant)



2.4. La fonction T : bascule T associée

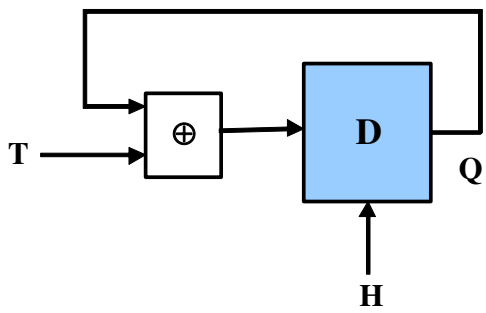
- Fonction Trigger (‘détente, gâchette’) :
- 1 entrée T (qui peut être l'entrée de synchronisation), 1 sortie Q



- Fonctionnement :

T	Q
0	Q
1	Q'

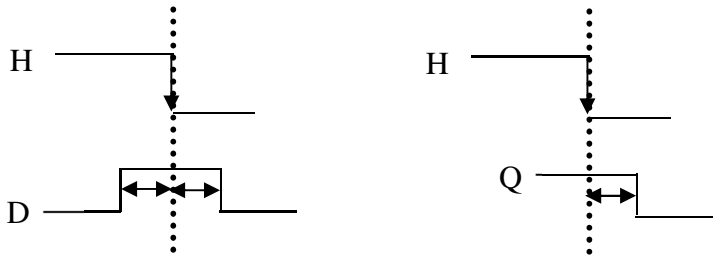
Exemples de réalisations



D=T’Q+TQ’

2.5. Remarques sur le temps de commutation

- Pour qu'une bascule fonctionne correctement il est nécessaire que le signal présent sur les entrées de la bascule soit stabilisé depuis un certain temps lorsque le front de l'horloge actif intervient (temps de « setup ») et reste stable pendant un certain temps après ce front d'horloge (temps de maintien ou encore temps de « hold »).
- **La commutation des sorties** d'une bascule se fait avec un certain temps de retard par rapport au signal qui a produit cette commutation (Horloge, ou Reset). Ces retards peuvent être différents selon le signal qui a produit la commutation mais également selon qu'il s'agit d'une commutation montante ou descendante.



- **Initialisation des bascules :** Tous les systèmes doivent être initialisés. On utilise des variables internes vraies au premier tic d'horloge pour initialiser les bascules → l'état du système

Chapitre 3

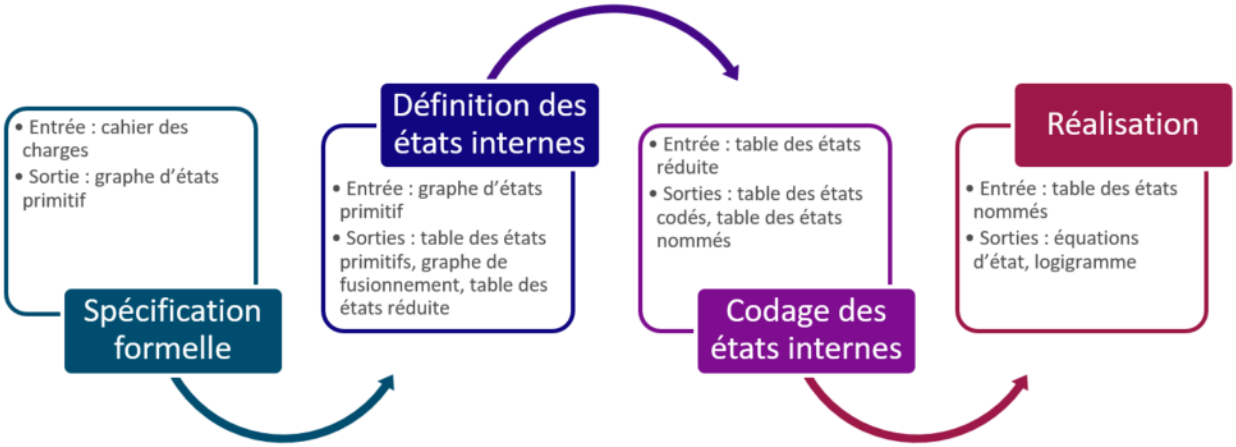
Synthèse des Systèmes Logiques Séquentiels

OBJECTIF :

Déterminer les équations d'évolutions (modèle d'états) à partir du cahier des charges

1. Principe de la méthode (méthode d'Huffman)

Elle comprend 4 étapes consécutives



2. Synthèse d'Huffman en détail

2.1. Etape de spécification formelle :

- Description du processus sous forme d'un graphe : **graphe d'états primitif.**
- Le graphe représente toutes les évolutions possibles.
- Le graphe est composé des états stables (sommets) et des transitions orientées entre états (arcs).
- Les signaux (entrées, sorties) de type niveau sont notés dans les états stables, les signaux de type impulsionnel sont notés sur les transitions.
- Dans le cas des systèmes synchrones, le signal d'horloge n'est pas représenté et seules apparaissent les entrées primaires. Dans certains cas il n'y a pas d'entrée primaire (par exemple un compteur), c'est l'horloge qui provoque l'évolution.

Le graphe d'état a déjà été décrit (p14). Pour rappel :

- Les nœuds du graphe sont les états du système.
- Une étiquette (nom symbolique) est assignée à chaque état.

- Les arcs du graphe sont orientés et représentent les évolutions entre états.
- Les états sont représentés par des cercles.

Remarques et règles pour l'obtention d'un graphe d'état primitif (voir p14)

Exemple d'application pour l'obtention d'un graphe d'état primitif :
Une machine à café distribue un café lorsqu'on introduit 30 centimes dans celle-ci. La machine possède deux détecteurs de pièces, un pour les pièces de 10 centimes (« P10 »), et un pour les pièces de 20 centimes (« P20 »). La machine possède deux sorties C et R. Quand « C » est à 1, il y a une distribution de café. Quand « R » est à 1, la machine rend la monnaie lorsque le montant introduit est supérieur à 30 centimes.
Le cahier des charges est le suivant :

- La machine n'accepte que des pièces de 10 ou 20 centimes
- Deux pièces ne peuvent pas être introduites/détectées en même temps
- Une pièce n'est détectée que pendant un certain temps. Les deux capteurs passent ensuite systématiquement par un état où ils sont à 0 tous les deux.

2.2. Etape de définition des états internes :

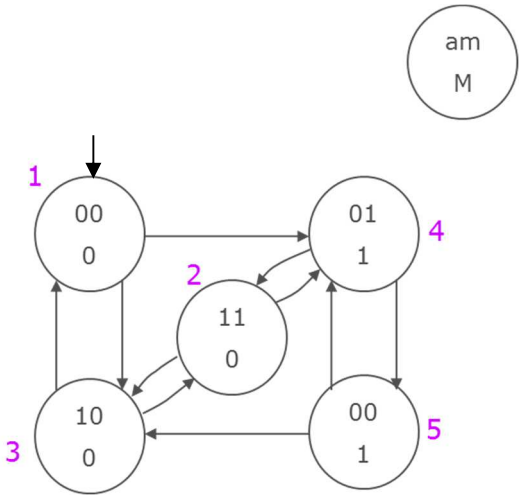
- Mise en place de la **table des états primitive.**
- Pour chaque état stable la table indique l'état suivant en fonctions des combinaisons d'entrées.
- Recherche des états redondants ou compatibles : pour des mêmes valeurs d'entrées et de sorties les transitions à partir de ces états amènent soit à des états identiques soit à des états redondants. On passe par la mise en place d'un **graphe de fusionnement.**

- **Réduction de la table** : étape de fusionnement des lignes par recherche des états compatibles pour obtenir une table réduite. L'objectif est de réduire le nombre de variables internes qui dépend du nombre d'états i.e. du nombre de lignes.

Remarque : la réduction des états n'est pas toujours "réalisable".

2.2.1. Table des états primitive

Reprenons l'exemple du moteur dont le graphe primitif est donné ci-dessous :



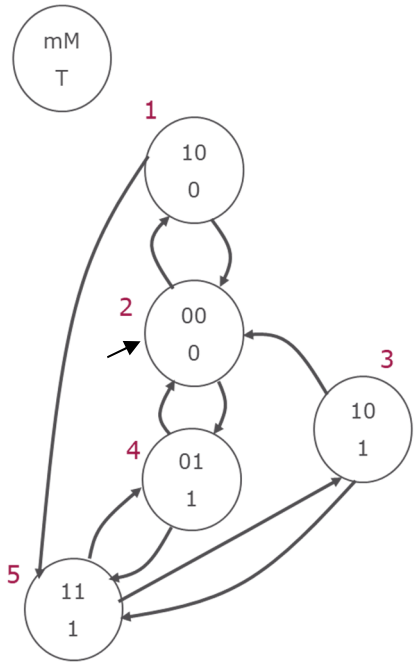
La table primitive des états déduite de ce graphe qui possède 5 lignes d'états, 4 colonnes des vecteurs d'entrée et une colonne de sortie M :

am Etat	00	01	11	10	S
1	①	4	-	3	0
2	-	4	②	3	0
3	1	-	2	③	0
4	5	④	2	-	1
5	⑤	4	-	3	1

Table primitive des états

- Chaque case indique l'état interne que va atteindre le système pour les entrées appliquées (colonne).
- Un état stable est cerclé.
- La sortie prend une valeur pour chaque ligne.
- Les cases notées '–' ne peuvent être atteintes depuis les états stables si on ne modifie qu'une seule entrée à la fois (principe d'adjacence des entrées consécutives). De manière générale des états '–' peuvent être mis en place lorsqu'une transition est non spécifiée par le cahier des charges.
- Cette mise en table correspond au modèle rebouclé d'Huffman.

Exercice :



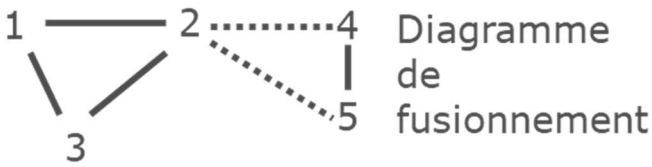
2.2.2. Graphe de fusionnement

Sur la table, les états internes 1 et 2 (lignes 1 et 2) ont la même sortie et conduisent aux mêmes états ou à des -. Ces 2 états peuvent être alors fusionnés en un seul état interne qui donnera la sortie 0 ➔ une seule ligne.

C'est également vrai des couples d'états (1, 3) (2, 3) ;

Ces 3 états sont dits **compatibles** car ils conduisent à des états identiques ou -. On peut donc les réduire en un seul état (1, 2, 3). Les états 4 et 5 peuvent aussi être fusionnés.

Ce travail de fusionnement est facilité par l'emploi d'un **diagramme de fusionnement** d'états (ou diagramme de réduction d'états) qui indique tous les couples compatibles. Les compatibilités entre états avec valeurs de sorties identiques sont indiquées en trait plein celles entre états à valeurs de sorties différentes sont indiquées en trait pointillé. Le diagramme de fusionnement est aussi appelé graphe de fusionnement.



Exercice :

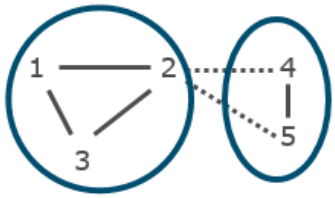
mM	00	01	11	10	T
1	2	-	5	①	0
2	②	4	-	1	0
3	2	-	5	③	1
4	2	④	5	-	1
5	-	4	⑤	3	1

2.2.3. Choix des états internes et table des états réduite

A partir de ce diagramme sont obtenus les regroupements d'états optimaux par rapport au nombre de variables internes nécessaires pour coder ces états par la suite.
Dans le cas général, le diagramme de fusionnement peut mettre en évidence plusieurs regroupements d'états possibles qui sont équivalents en termes de nombre de variables internes nécessaires. Plusieurs tables réduites sont alors envisageables.
Pour l'exemple il est possible de regrouper (1-2-3) et (4-5). On peut également de regrouper (1-3) et (2-4-5). On ne peut regrouper que des états qui sont deux à deux compatibles.

Du diagramme de fusionnement se déduit la **table réduite des états** (obtenue en considérant les regroupements d'états optimaux par rapport au nombre de variables internes nécessaires pour coder ces états par la suite) qui aura le même comportement séquentiel que la table primitive.

Sur le moteur, on fait par exemple les regroupements suivants :



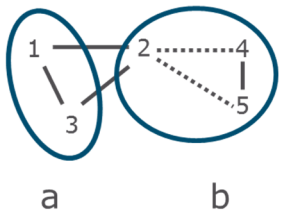
Si on appelle 'a' l'état qui correspond aux états 1-2-3 fusionnés et 'b' l'état qui correspond aux états 4-5 fusionnés, cela mène à la table réduite :

am Etat	00	01	11	10	M
a	a	b	a	a	0
b	b	b	a	a	1

Cette table n'ayant que 2 états internes au lieu de 5 pour la table primitive, sa réalisation sous forme d'un circuit logique sera « statistiquement » plus simple (donc moins chère), les états pouvant être matérialisés par moins de variables logiques (2 au lieu de 3).

- Une table d'états (réduite ou pas) est de type Moore si est seulement il n'y a qu'une seule valeur des sorties pour chaque ligne.
- Une table d'états est de type Mealy si et seulement si certains états totaux d'une même ligne n'ont pas les mêmes valeurs de sortie.
- La fusion d'états ayant même valeur de sortie conduit à une machine de Moore alors que le regroupement d'états à valeurs de sorties différentes conduit vers une machine de Mealy.
- Les machines de Mealy conduisent généralement à des circuits plus simples que les machines de Moore.

am Etat	00	01	11	10	M
1	①	4	-	3	0
2	-	4	②	3	0
3	1	-	2	③	0
4	5	④	2	-	1
5	⑤	4	-	3	1



→ Sorties différentes : synthèse de Mealy

On en déduit la table réduite des états...

am Etat	00	01	11	10
a				
b			_	

... et la table de la sortie

am Etat	00	01	11	10
a				
b				

2.3. Etape de codage des états internes :

Pour aboutir à un circuit séquentiel, il faut coder chaque état interne par une configuration de variables logiques appelées variables internes dans le modèle d'Huffman.
Les fonctions y = f (x, Y) et z = g (x, Y) se déduisent ensuite du modèle.

- A chaque ligne de la table (réduite) i.e. à chaque état interne est affectée une combinaison des variables internes : un **code**
- Le choix du code dépend du type de système séquentiel.
  - Dans le cas des systèmes asynchrones des contraintes d'adjacence doivent être respectées (problème de course critique).
  - Dans le cas des systèmes synchrones le codage des états est libre (choix d'un code "compact").
- Mise en place de la **table des états codés**

Pour l'exemple simple du moteur, une seule variable interne y suffit. Le codage est le suivant : les deux états internes :

- Etat a par y = 0
- Etat b par y = 1.

La **table des excitations** ou la **table des états codés** peut alors être mise en place.

Chaque ligne correspond à une valeur de Y qui est l'état interne du système (celui dans lequel se trouve le système à l'instant présent). Chaque case correspond à la valeur de l'état suivant donc de la variable y.

Y \ a m					M
0	0	1	0	0	0
1	1	1	0	0	1

Table des excitations

2 états internes - 5 états totaux stables

Exercice d'application :

mM Etat	00	01	11	10	T
a	b	a	c	a	0
b	b	d	b	a	0
c	b	d	c	a	1
d	b	d	b	b	1

mM	00	01	11	10	T

2.4. Evolution du système

Le comportement du système est totalement décrit par la table des états (primitive, réduite, codée). Plusieurs types d'évolutions sont possibles :

- Etat stable → état stable :
- Etat stable → état(s) transitoire(s) → état stable
- Possible régime permanent oscillatoire : état stable → état transitoire

Application :

y_1y_2 \ AB	00	01	10	11
00	00	01	01	10
10	10	11	01	11
11	10	11	01	10

$y_1^+y_2^+$

2.5. Etape de réalisation

2.5.1. Principes

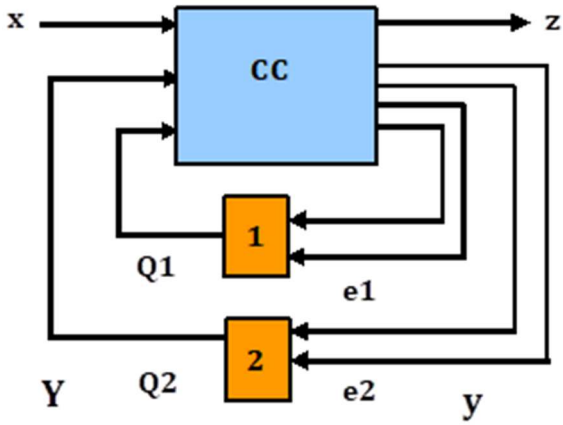
Une fois le codage des états effectué, la **réalisation consiste à trouver un circuit logique qui permette la mise en œuvre des états et des sorties**.

La mise en œuvre peut se faire à l'aide de portes logiques (AND, OR, NOT…). Dans ce cas, il suffit de trouver l'expression logique des états et des sorties à partir des tables de Karnaugh précédemment obtenues.
Les circuits intégrés étant souvent composés uniquement de portes identiques (NAND ou NOR), il est utile de savoir trouver l'expression logique en portes NAND ou en porte NOR des états et des sorties (voir Annexe).
Dans d'autres cas, les circuits sont uniquement composés de bascules : c'est le choix fait dans ce cours.

PRINCIPE :

Les états sont matérialisés par des bascules asynchrones ou synchrones

Chaque variable interne est traduite par une bascule.



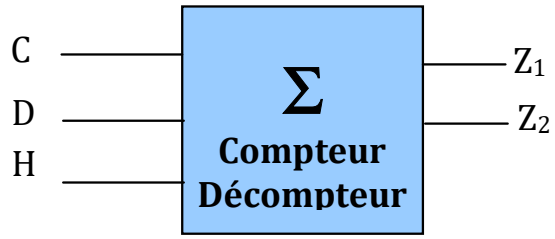
La réalisation s'effectue en deux étapes :

- Etape 1 : Recherche des commandes des bascules = expression des entrées des bascules, qui assurent que les sorties des bascules suivent la table des états codés. (utilisation des tables de transition des bascules)
- Etape 2 : Recherche des expressions des sorties

Le principe de synthèse synchrone est exactement le même dans le cas de bascules RS, JK, D ou T.

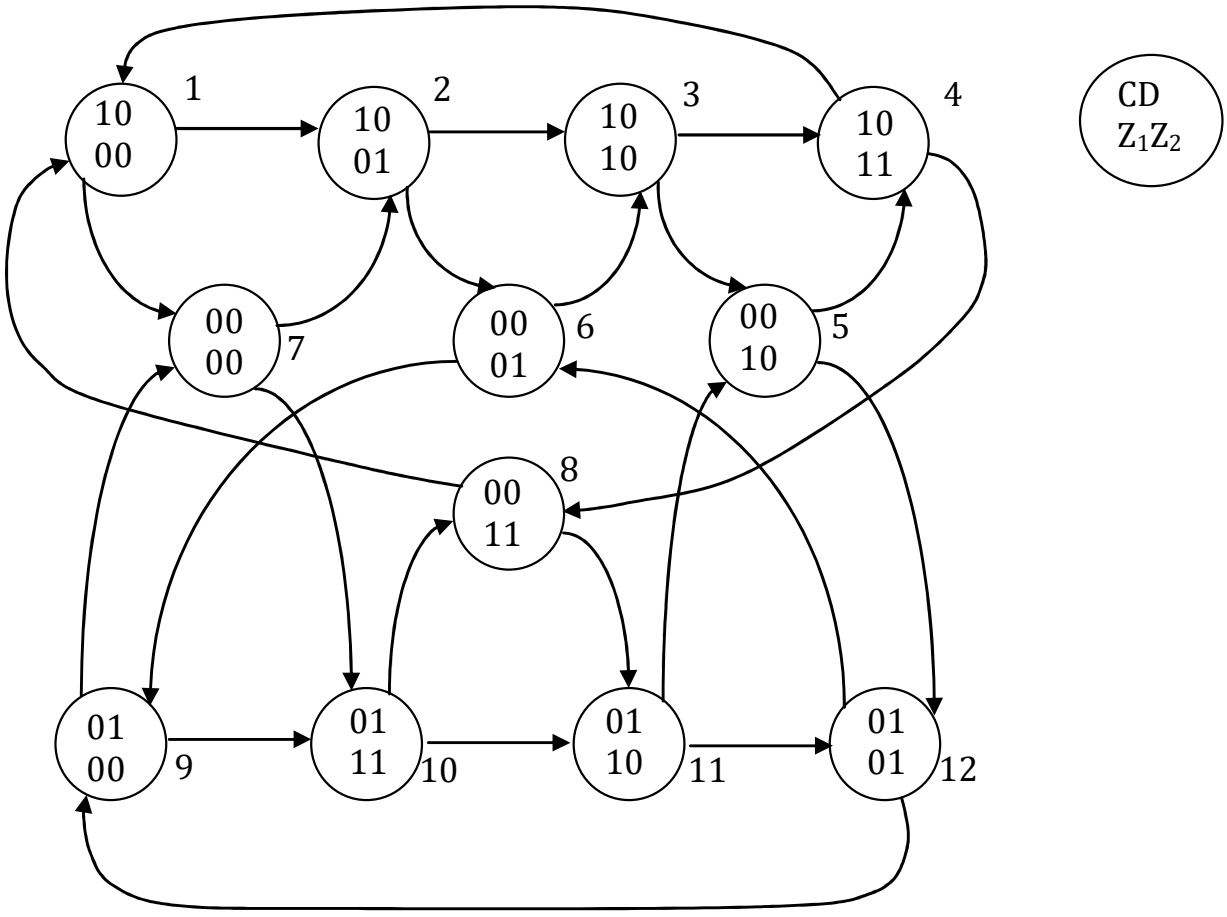
2.5.2. Présentation de la méthode sur un exemple: compteur - décompteur synchrone modulo 4.

Soit un système synchrone avec deux entrées C et D de type niveau (C. D = 0), une entrée d'horloge H et deux sorties Z1 et Z2.



- Lorsque l'entrée C est active le système doit compter de 0 à 3 et afficher le résultat en binaire sur les deux sorties.
- Lorsque c'est l'entrée D, le système doit décompter.
- Si aucune entrée n'est active, le système conserve la valeur des sorties (mémorisation). Le système évolue sur chaque front montant de l'horloge H.

2.5.2.1. Graphe d'états primitif



2.5.2.2. Table primitive des états et table réduite

La table primitive déduite du graphe a 12 lignes et 4 colonnes.

CD Etat	00	01	11	10	Z1 Z2
1	7	-	-	2	00
2	6	-	-	3	01
3	5	-	-	4	10
4	8	-	-	1	11
5	⑤	12	-	4	10
6	⑥	9	-	3	01
7	⑦	10	-	2	00
8	⑧	11	-	1	11
9	7	10	-	-	00
10	8	11	-	-	11
11	5	12	-	-	10
12	6	9	-	-	01

Notion d'états compatibles - graphe de fusionnement :

La table primitive peut être réduite à 4 états (4 lignes) : ces 4 états correspondent aux quatre valeurs de sortie possibles : 0, 1, 2 et 3.
Table réduite : a : (1-7-9); b :(2-6-12) ; c :(3-5-11) ; d :(4-8-10)
Le diagramme de fusionnement n'a pas été tracé ici.

CD Etat	00	01	11	10	Z1 Z2
a	ⓐ	d	-	b	00
b	ⓑ	a	-	c	01
c	ⓒ	b	-	d	10
d	ⓓ	c	-	a	11

2.5.2.3. Table des états codés

C / D / Etat					Z1Z2
00	00	11	-	01	00
01	01	00	-	10	01
11	11	10	-	00	11
10	10	01	-	11	10

2.5.2.4. Table de commande des bascules RS synchrones

Le nombre de bascules à utiliser correspond aux nombres de variables internes à utiliser pour coder les états, sachant que n variables vont permettre de coder 2^n états.
Ici on utilisera donc 2 bascules (par exemple des bascules RS).

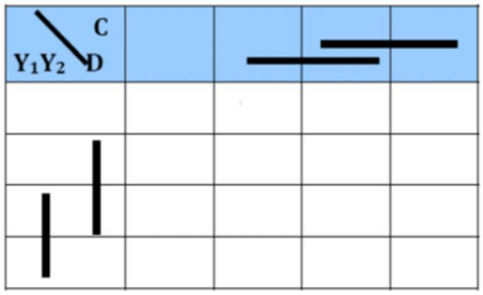
Utilisation de la table des transitions de la bascule RS :

Q_i	Q_{i+1}	R	S
0	0	-	0
0	1	0	1
1	0	1	0
1	1	0	-

En partant de la table des états codés, et en regardant la table des transitions de la bascule, on retrouve la table des entrées (commande) de chaque bascule.

CD Y1 Y2	00	01	11	10
00	0	1	-	0
01	0	0	-	1
11	1	1	-	0
10	1	0	-	1

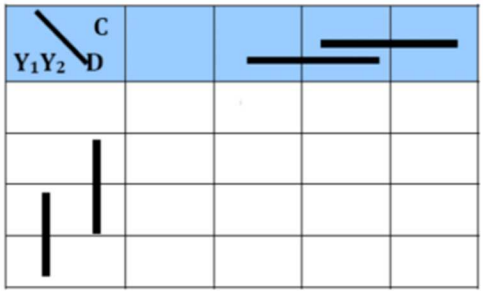
Table de y_1



Entrées (R et S) de la bascule RS dont la sortie est y_1

CD Y1 Y2	00	01	11	10
00				
01				
11				
10				

Table de y2



Entrées (R et S) de la bascule RS dont la sortie est y_2

D'où les commandes des 2 bascules :

$R_1 = Y_1Y_2'D + Y_1Y_2C$
$S_1 = Y_1'Y_2'D + Y_1'Y_2C$

Sorties : Z1 = Y1= Q1 et Z2 = Y2 = Q2

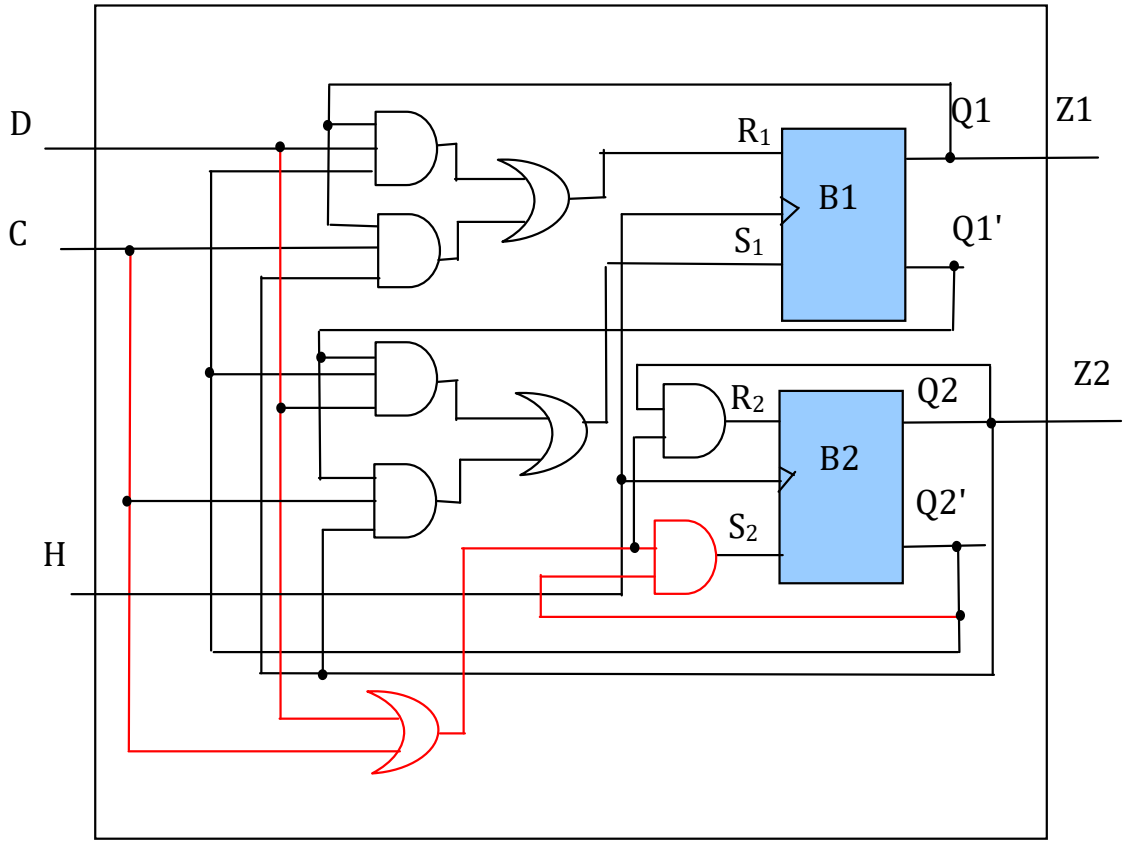
Exercice d'application :

Suite à la spécification d'un système avec deux entrées A et B et une sortie M, la table des états réduite suivante pour un système logique est la suivante :

q \ A B					M
a	a	b	b	a	0
b	b	b	a	a	1

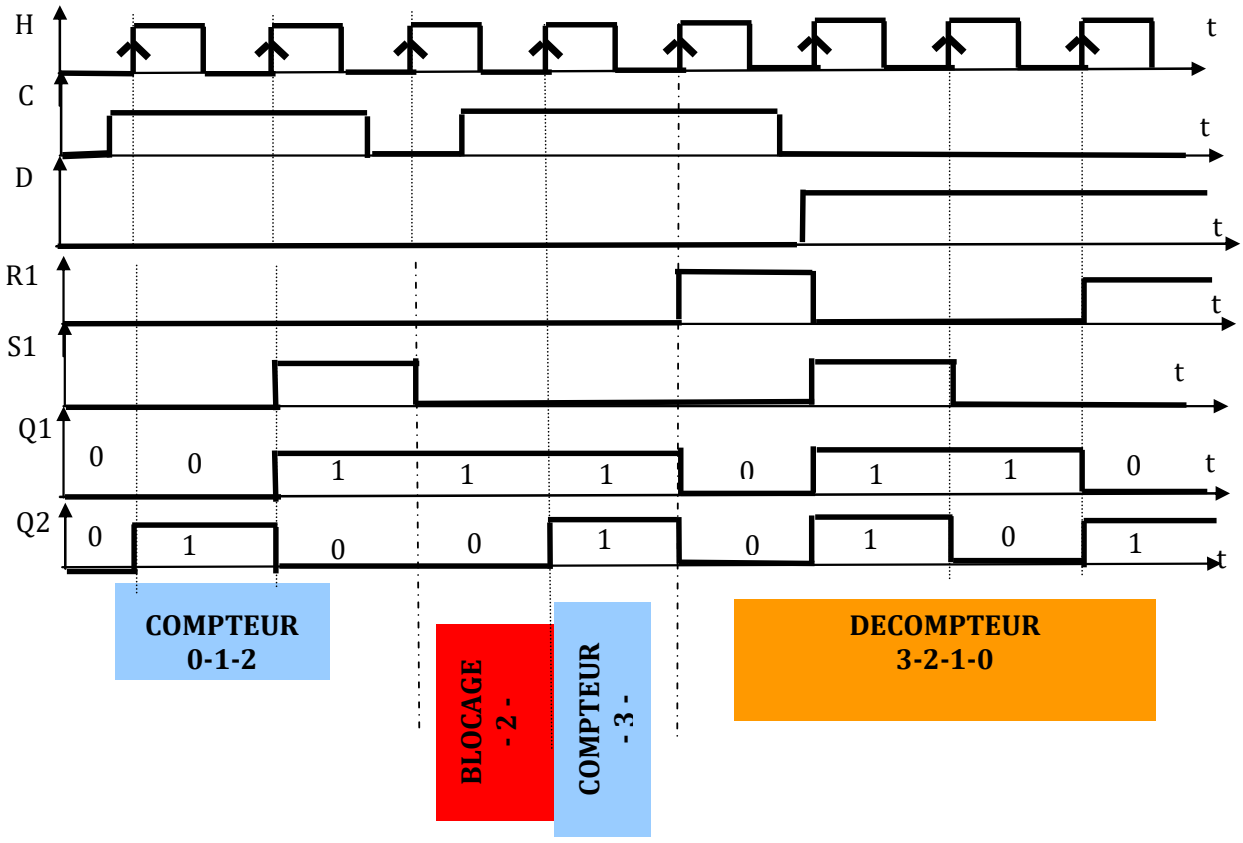
Proposer un codage, puis une réalisation de ce système en bascule(s) RS sous la forme d'équations logiques.

2.5.2.5. Réalisation : logigramme



$S_2 = DY_2' + C\,Y_2' = (D+C).Q'_2$

2.5.2.6. Chronogramme d'une séquence de fonctionnement du circuit séquentiel



Annexe 1 : Méthode simplifiée dans le cas général d'un compteur (séquenceur)

Un séquenceur est un système délivrant une séquence de combinaisons particulières en boucle. Pour ce genre de système, la seule entrée est l'horloge.

Lorsque le système à réaliser est un compteur ou de manière générale un séquenceur, la méthode d'Huffman peut se simplifier:

- Les sorties du système bascules peuvent être directement portées par les sorties des bascules.
- L'étape de spécification formelle peut être supprimée puisque le graphe d'états se réduit à une simple succession d'états.
- La table (primitive) des états est donnée par la séquence à réaliser et permet directement de déterminer les équations de commande des bascules.

Exemple d'un système d'affichage : ce système permet de définir l'allumage de lampes suivant la séquence définie ci-dessous. A la fin de cette séquence l'afficheur revient à l'état initial et recommence. Il s'agit donc d'un système séquentiel synchrone de signal d'horloge H ou plus simplement d'un séquenceur.

G E I
G E I
G **E** I
G E **I**
G E **I**
G **E I**
G E I

Ce système n'a pas d'entrée en dehors de l'horloge H. Ses sorties correspondent aux trois lampes (G, E et I). Chaque affichage constitue un état du système. Un état est donc une combinaison de l'état (allumée (1) ou éteinte (0)) des trois lampes. Par exemple à l'état initial toutes les lampes sont éteintes, la combinaison associée est 000. Le fonctionnement complet du système est décrit par une succession d'affichages. Le graphe d'états est une simple séquence des 7 états. La séquence terminée revient à son état initial le graphe constitue un cycle. La table des états est donc obtenue directement à partir de la séquence donnée dans le cahier des charges :

Etats Présents GEI	Etat Suivant GEI
000	100
100	010
010	001
001	101
101	011
011	111
111	000

Table des états

1. Une bascule JK (par exemple) est associée à la commande de chaque lampe. La sortie de chaque bascule est donc associée à chaque variable interne représentant l'état des lampes. Les valeurs des entrées des trois bascules sont déterminées à partir de la table de commande d'une JK et de la table des états :

Etats Présents GEI	Etat Suivant GEI		J_G K_G	J_E K_E	J_I K_I
000	100		1 -	0 -	0 -
100	010		- 1	1 -	0 -
010	001		0 -	- 1	1 -
001	101		1 -	0 -	- 0
101	011		- 1	1 -	- 0
011	111		1 -	- 0	- 0
111	000		- 1	- 1	- 1

2. L'expression (minimale) des entrées des 3 bascules (notées 'G', 'E' et 'I') est alors déduite par table de Karnaugh.

Remarque : dans cet exemple ma combinaison 110 n'est pas spécifiée.

EI G	00	01	11	10
0	1-	1 -	1-	0-
1	-1	-1	-1	--

$$J_G = \bar{E} + \mathrm{I}$$

$$K_G = 1$$

EI G	00	01	11	10
0	0-	0-	-0	-1
1	1-	1-	-1	--

$$J_E = G$$

$$K_E = G + \bar{I}$$

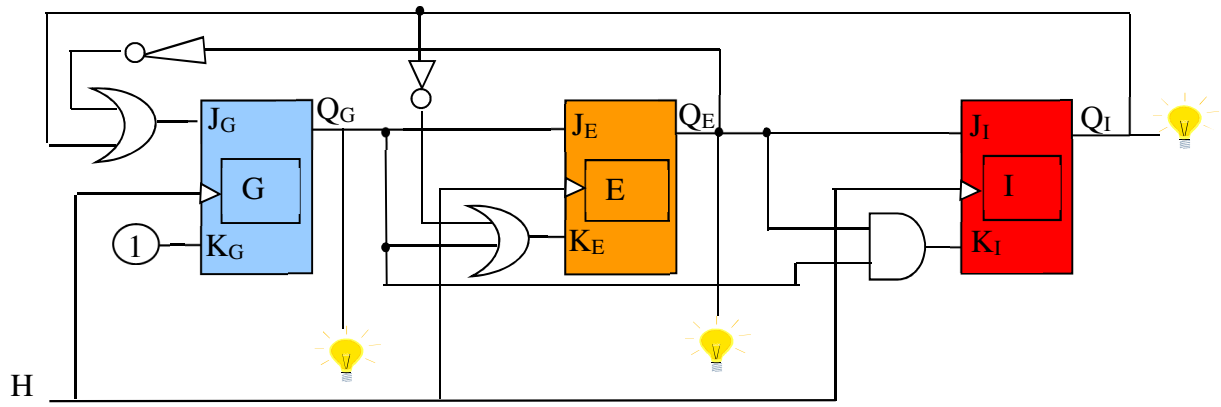
EI G	00	01	11	10
0	0-	-0	-0	1-
1	0-	-0	-1	--

$$J_I = E$$

$$K_I = E\mathrm{G}$$

Les sorties des bascules correspondent directement aux commandes des lampes.

3. Réalisation : le logigramme



Exercice d'application :
Soit un compteur synchrone à 2 bits, b1, b2 qui à chaque top d'horloge compte 0-1-2-3-0-1… sans fin. Chaque valeur correspond à une combinaison des bits :
0 : b1=0, b2=0 / 1 : b1=0, b2=1 / 2 : b1=1, b2=0 /3 : b1=1, b2=1.

1. Combien d'entrées possède ce système ?
2. Combien de sorties ?
3. Combien d'états différents va prendre ce système ?
4. On souhaite faire une synthèse à l'aide de bascule JK. De combien de bascules aura-t-on besoin ?

Effectuer la synthèse.

Résumé de la méthode pour les séquenceurs :

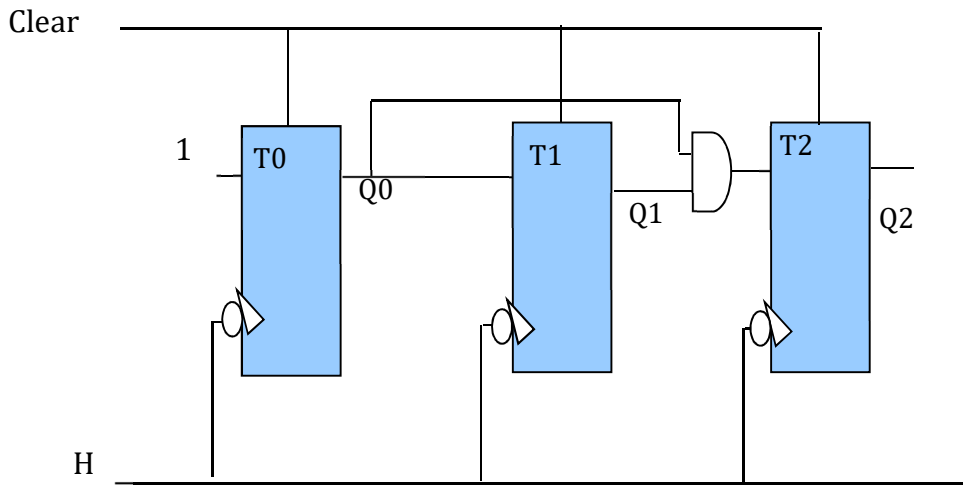
- Généralement pas d'entrée en dehors d'une horloge
- X sorties → X variables d'état → X bascules
- Écriture de la table des états futurs en fonction des états présents
- Déduction des tables pour chaque bascule
- Déduction des équations logiques des entrées de chaque bascule
- Écriture du logigramme

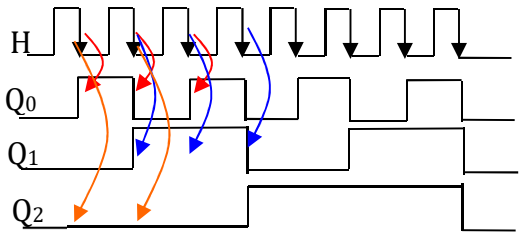
Exemple du compteur synchrone modulo 8

Etat Présent			Etat Suivant			T2	T1	T0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

T0=1
T1=Q0 et T2=Q0.Q1

Les expressions de T0, T1, T2 étant évidentes il n'est pas utile de mettre en place les tables de Karnaugh de T0,T1 et T2 en fonction de Q0,Q1 et Q2





Annexe 2 : Quelques circuits élémentaires à base de bascules

1.1. Compteur - Décompteur

Un compteur est une **association de n bascules** permettant de décrire au rythme d'une horloge une séquence déterminée qui peut avoir au maximum 2^n combinaisons différentes. Les combinaisons apparaissent toujours dans le même ordre. Chaque combinaison forme un mot de n bits qui évolue en croissant ou décroissant (décompteur) au rythme de l'horloge.

Une combinaison de sortie d'un compteur est appelée « état ». Le nombre d'états différents pour un compteur est appelé **le modulo** de ce compteur.

Dans le cas des compteurs ou décompteurs en binaire naturel la synthèse est assez intuitive. Mais lorsque la séquence de comptage à générer ne dispose pas de propriétés particulières nous verrons au chapitre 3 qu'il faut une méthode plus systématique de synthèse.

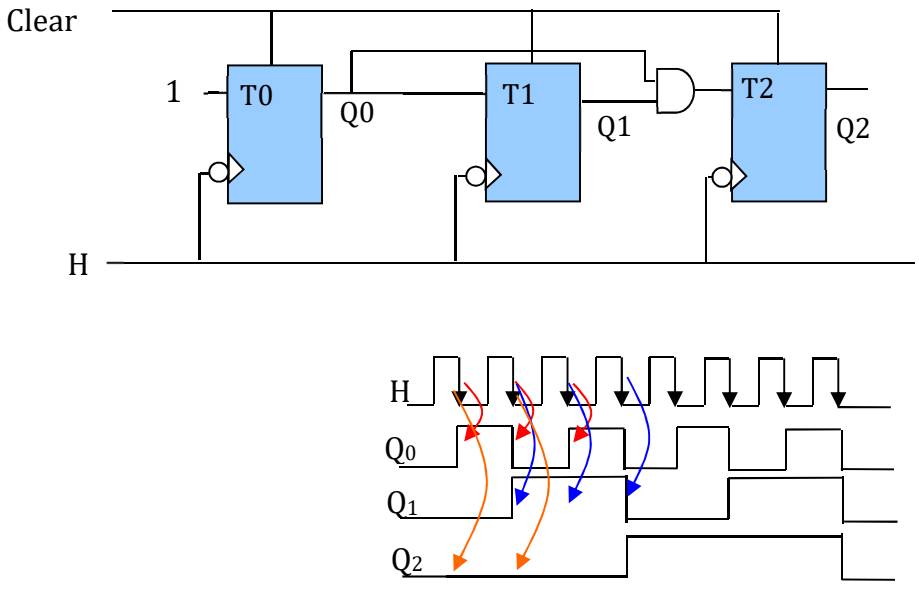
1.1.1. Compteur synchrone

Toutes les bascules sont commandées pas le même signal d'horloge

Les réalisations des compteurs reposent sur des structures de diviseur de fréquence à base de bascules T, D, ou JK permettant de réaliser des structures fonctionnellement équivalentes.

Exemple : compteur modulo 8 :
3 bascules T sur front descendant en cascades. Inversion de la sortie Q pour T=1

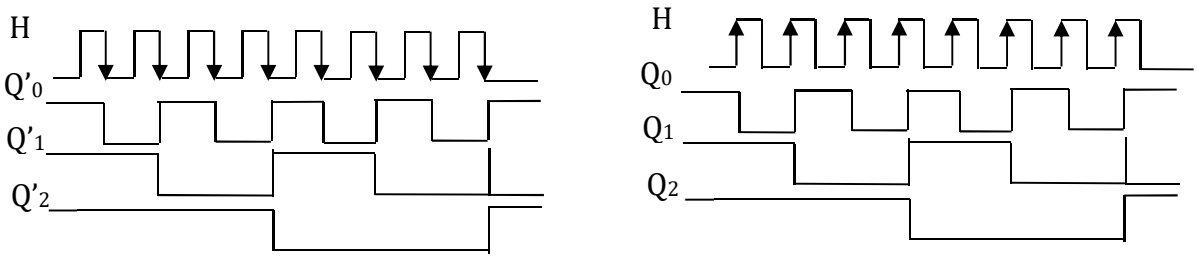
Q2	Q1	Q0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1



T0=1 ; T1=Q0 ; T2=Q0.Q1

1.1.2. Décompteur synchrone :

Un décompteur modulo 8 peut s'obtenir de la même façon en considérant les sorties complémentées ou en réalisant le même circuit avec des bascules sur front montant.

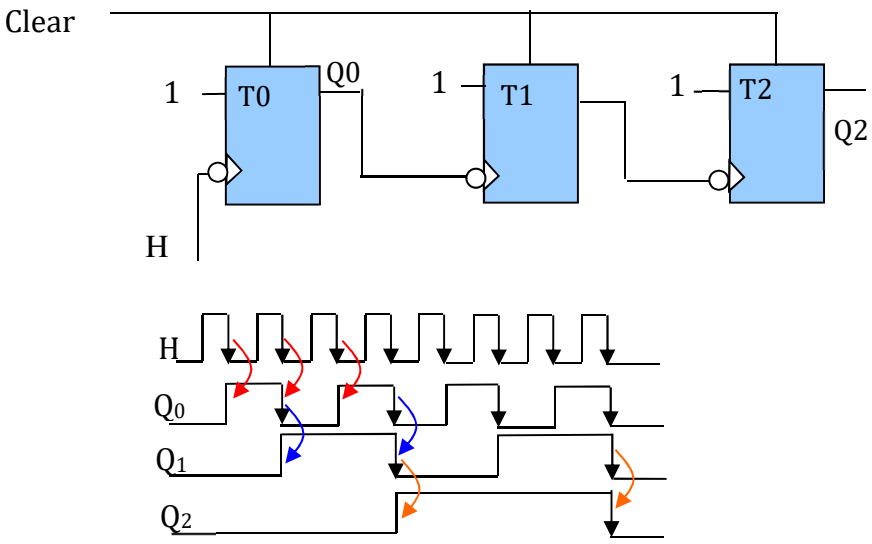


1.1.3. Compteur - décompteur asynchrone :

Toutes les bascules ne sont pas commandées par le même signal d'horloge.

Attention appellation « asynchrone » mais il y a une horloge ! Les différentes bascules fonctionnent de manière asynchrone.

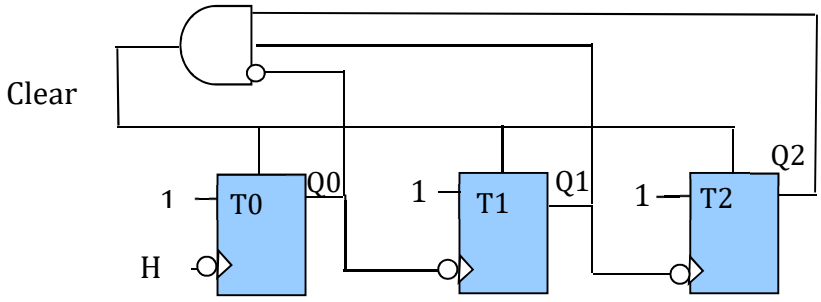
Exemple : compteur asynchrone modulo 8 sur front descendant avec des bascules T :



1.1.4. Compteur – décompteur à modulo quelconque :

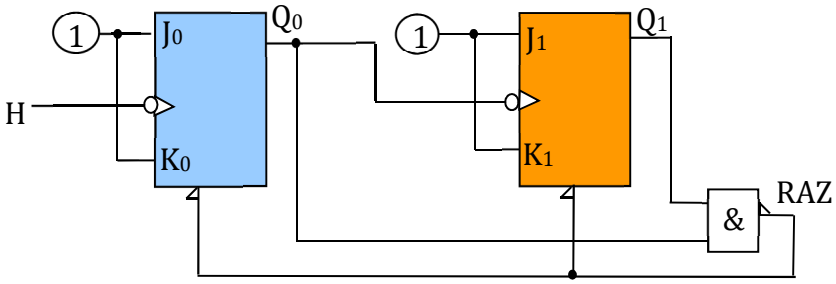
Pour réaliser un compteur ou un décompteur dont le modulo n'est pas une puissance de 2, une solution est d'agir sur les entrées de forçage à zéro (clear) lorsque la combinaison correspondant au modulo du compteur se produit sur les sorties de celui-ci.

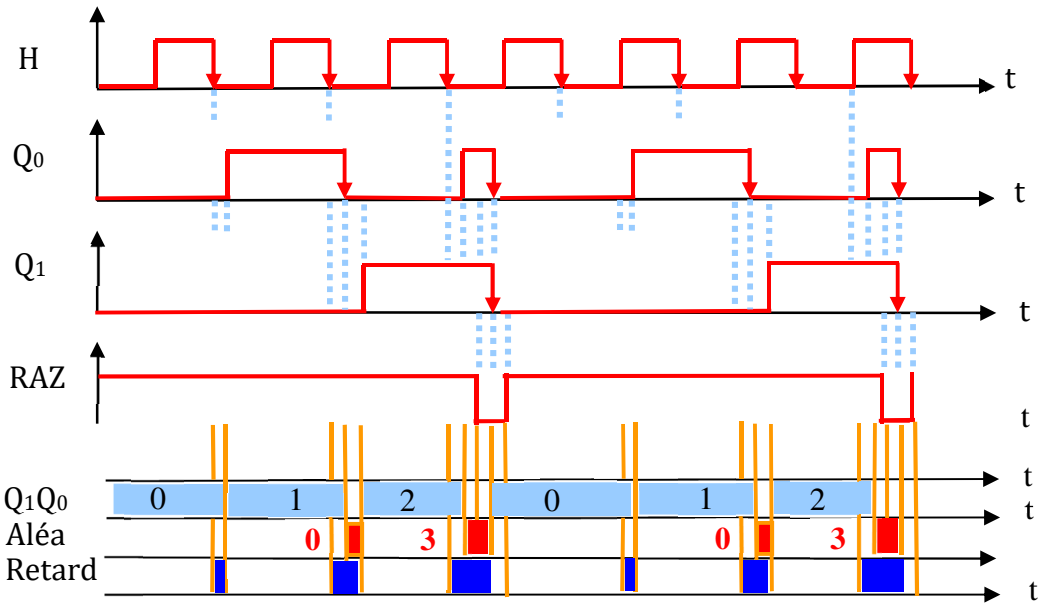
Exemple : compteur asynchrone par 6. Lorsque la combinaison 6 est détectée (Q2Q1Q0=110) elle est renvoyée sur le signal « RAZ ».



1.1.5. Problèmes sur l'asynchrone

Les bascules ne commutant pas sur le même signal d'horloge, les retards de commutation se cumulent sur chacune des bascules du compteur. En effet, c'est la commutation de la première bascule qui entraîne l'activation de la seconde qui elle-même entraîne l'activation de la troisième etc. comme, l'illustre la figure ci-dessous dans le cas d'un compteur asynchrone modulo 3.





Un autre problème vient du fait que la structure d'un compteur asynchrone nécessite de la logique sur des signaux asynchrone (horloge générée par une bascule, ou clear). Cette logique

peut engendrer des aléas de fonctionnement (courses critiques) qui entraînent des dysfonctionnements du compteur (voir cours de 3ème année).

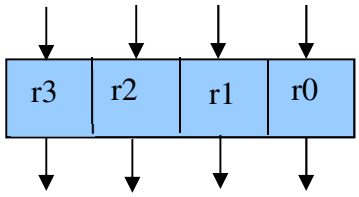
Pour les compteurs synchrones dont le modulo n'est pas une puissance de 2, pour éviter la mise en place de la logique de remise à zéro, le mieux est de redéfinir les fonctions d'entrées des bascules comme nous le verrons au chapitre 3.

1.2. Registre

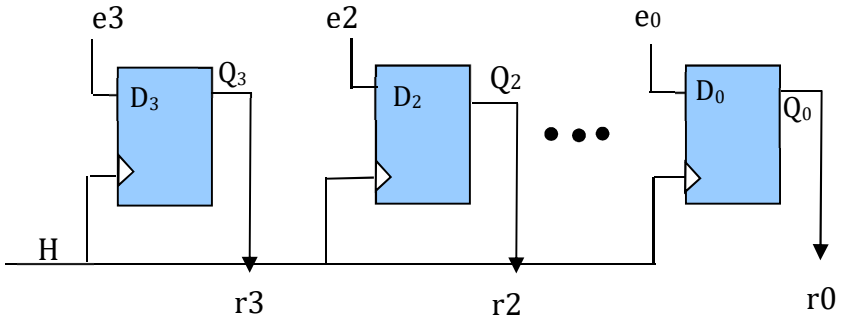
Un registre résulte de l'assemblage d'un **ensemble combinatoire et séquentiel** permettant le stockage d'informations binaires en vue d'une **mémorisation temporaire avec ou sans traitement.**

1.2.1. Registre de mémorisation

Encore appelé registre à entrées parallèles ou encore à écriture parallèle.
C'est le registre élémentaire. Il permet de mémoriser un mot binaire. L'écriture d'un registre constitue le chargement des données d'entrée dans le registre. La lecture consiste à la récupération des données de sorties.



Le registre élémentaire est constitué d'une juxtaposition de bascules (D) permettant de mémoriser un mot binaire.

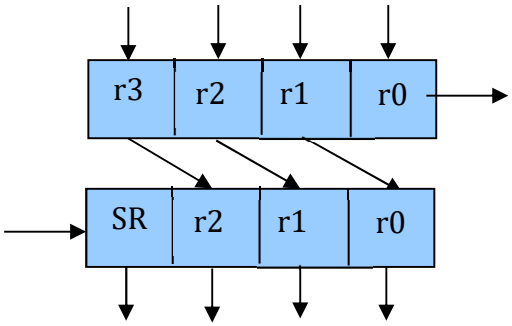


En général une entrée de contrôle est rajoutée afin d'interdire le chargement du registre lorsque celui-ci n'est pas souhaité : Di=C.ei+C'Qi
Cette entrée revient à inhiber l'horloge.

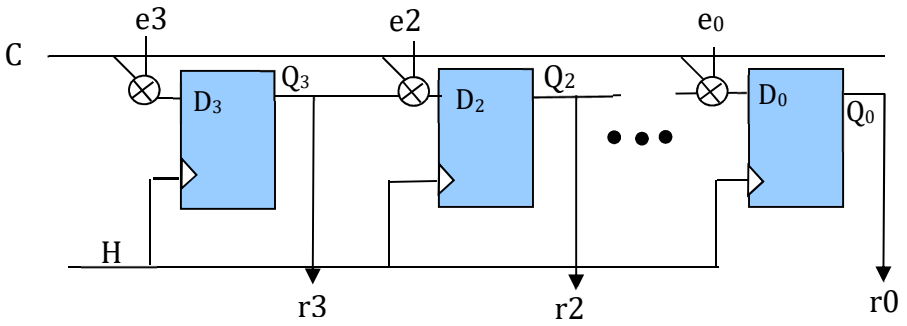
1.2.2. Registre à décalage

C'est une association de bascules permettant de décaler un mot binaire. Le décalage à droite ou à gauche rend libre une « case » du registre pour recevoir un bit série au travers de l'entrée correspondante. L'autre bit extrême est perdu.

Exemple : registre écriture série décalage à droite



Dans la figure ci-dessous, l'entrée d'un mot peut se faire en fonction d'une entrée de contrôle (C), soit par chargement parallèle comme dans le cas du registre élémentaire soit par décalage à partir d'une entrée série. L'entrée e3 joue à la fois le rôle d'entrée parallèle est d'entrée série.

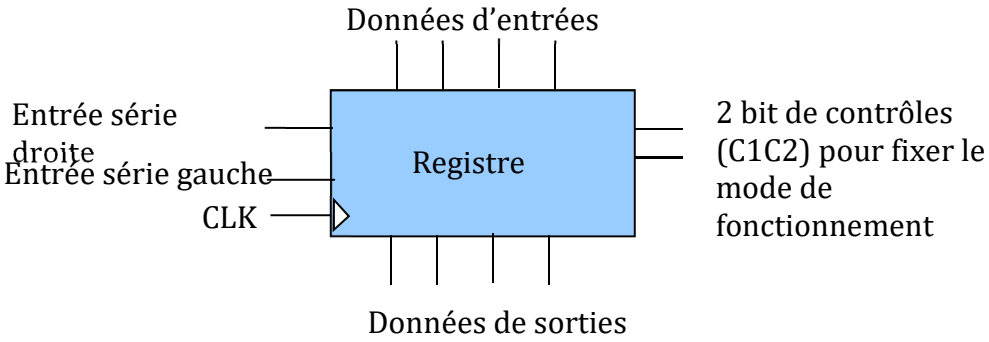


Di=C.ei+C'Qi-1

Remarque : un registre à décalage à droite peut être utilisé comme un diviseur par 2 alors qu'un registre à décalage à gauche peut être utilisé comme un multiplieur par 2.

1.2.3. Registres universels :

C'est une association de bascules permettant les quatre modes de fonctionnement à partir de 2 bits de contrôle :



C1C2=00 chargement parallèle, C1C2=01 décalage à droite, C1C2=10 décalage à gauche, C1C2=11 Inhibition de l'horloge (mémorisation).
Pour sélectionner le mode de fonctionnement selon les valeurs des 2 bits de contrôle chaque entrée de bascule est précédée d'un multiplexeur 1 parmi 4. Ce multiplexeur est un circuit d'aiguillage à 2 entrées (les 2 bits de contrôle) et une sortie (Di) : Di=C'1C'2.ei+C'1C2Qi-1+C1C'2Qi+1 + C1C2Qi

Annexe 3 : Etape de réalisation : synthèse en portes NAND, synthèse en portes NOR

Synthèse en portes NAND :

- On part d'une expression SIGMA-PI de f (si possible minimale)
- Chaque terme ET devient un NAND, le OU devient un NAND
- Les littéraux se conservent sauf si un terme ET n'a qu'un seul littéral qui est alors complémenté

Ex: $f = a + \bar{a}cd + a\bar{b}c = \bar{a}|(\bar{a}|c|d)|(a|\bar{b}|c)$

Synthèse en portes NOR

- On part d'une expression PI-SIGMA de f (si possible minimale)
- Chaque terme OU devient un NOR, le ET devient un NOR
- Les littéraux se conservent sauf si un terme OU n'a qu'un seul littéral qui est alors complémenté
-

Ex: $f = a(\bar{a} + b)(a + \bar{b} + c) = \bar{a}\downarrow(\bar{a}\downarrow b)\downarrow(a\,\bar{\downarrow}\,\bar{b}\,\bar{\downarrow}\,c)$

Application :

Moteur : synthèse en portes NOR

am y	00	01	11	10	M
0	⓪	1	⓪	0	0
1	①	①	0	0	1

Synthèse PI-Sigma: constituée du produit logique des maxtermes aux points faux de la fonction (maxterme : somme des compléments des variables)

$$y = M = \bar{a}.(y + m)$$

$$Y = M = a\downarrow(y\,\bar{\downarrow}\,m)$$