CHAPITRE 6
COMPTEURS SYNCHRONES

- Séquenceur : Circuit ou système qui passe à travers une série d'états successifs grâce à une horloge synchrone ou asynchrone.
  Ex : Machine à laver.

- Au chapitre 6, nous allons voir les séquenceurs fabriqués à l'aide de Flips Flops tous connectés à une horloge commune.
  Ex : Compteur (avec comme seule entrée extérieure : CK, l'horloge)
  + avec de façon optionnelle : reset, load, halt (mise à zéro, chargement, arrêt).

  Si d'autres entrées sont présentes, on parlera de machines d'état ("State machine", vus au chapitre 7).

  État (STATE) : À tout moment, les sorties de tous les flips flops constituent son état.

  Une table : État présent – État futur (Present-State → Next State ou PS-NS) : sera employée pour le design.

- Aussi : registres synchrones.
- Avec le chapitre 6, on pourra faire le design de n'importe quel compteur synchrone !

6.1 Circuit séquentiel, état, horloge

Circuit synchrone : élément de mémoire : Flip Flop, même horloge pour tous les Flips Flops
Circuit séquentiel : élément de mémoire : latch et Flip Flop

Rappel : Flip Flop, l'horloge est "edge-sensitive" et Latch, l'horloge est "level-sensitive".

6.2 État et sortie

- Pour déterminer l'état futur :
  - Circuit combinatoire : il faut connaître : les entrées présentes + diagramme des portes

- Circuit séquentiel : il faut connaître : - l'histoire des entrées
  - les entrées
  - le diagramme du circuit

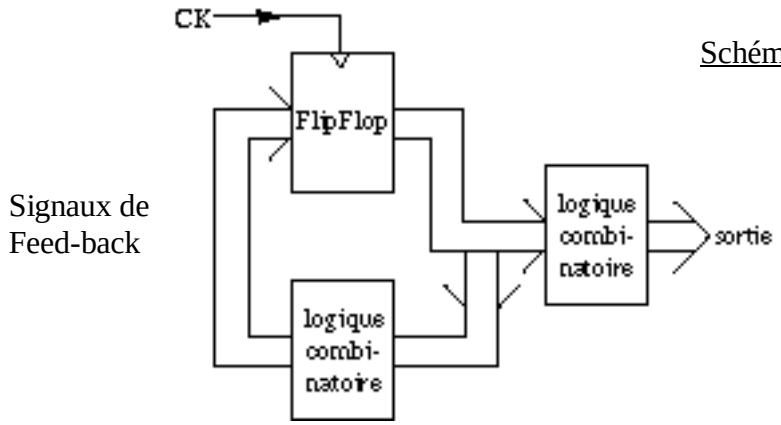
"L'histoire des entrées" = sorties courantes des Flips Flops, c'est l'état du circuit.

Si N Flip Flop $\Rightarrow$ au maximum 2^N différents états.

L'état du circuit n'est pas toujours égal à la sortie des Flips Flops (si logique combinatoire en sortie).

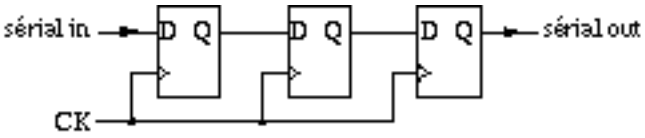
Exemple :

Schéma général d'une "Machine de Moore"



Exemple : Shift register.

La sortie = dernier FF seulement.



circuit FIFO (First In - First Out)

Les Flips Flops ne sont pas directement reliés à la sortie. Ils sont dits cachés ("hidden").

Si on veut une machine à M états (cycles de M états) avec N Flip Flop $\Rightarrow$ $\boxed{2^N \geq M}$
(Parfois on peut mettre plus de FF que le minimum requis).

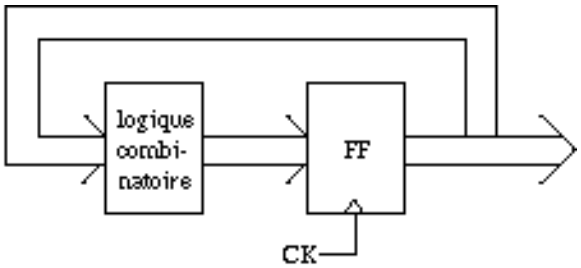
Les ÉTATS d'un circuit séquentiel peuvent aussi avoir un nom : ATTENTE, ACCEPTE, DÉMARRE LE MOTEUR, etc.

Un choix judicieux des noms des états et des variables peut faciliter le design comme on verra plus tard.

6.3 Fréquence maximum avec 1 seule horloge

- On emploie 1 seule fréquence d'horloge pour tous les FF et tous les FF sont du même type ↑ ou ↓.
- L'horloge ne doit pas être retardée par des portes logiques, ceci causerait des arrivées de coups d'horloge à des instants différents à cause du "Skew" de l'horloge.
- À cause des avantages de synchronisme, plus de 90 % des circuits emploient une seule horloge et sont du type "séquentiel". (Ça élimine les courses et le "catching").
- Sorties stables qui ne changent qu'au coup d'horloge.
- Quelle est la fréquence Max de l'horloge ?

On a la topologie suivante :



Délai Δ_C Délai Δ_f + Set uptime + hold time

⇒ période minimum $\boxed{t_{MIN} = \Delta_C + \Delta_f + t_{setup} + t_{hold}}$

⇒ $\boxed{f = \frac{1}{T \min}}$

Ici $t_{hold} = 0$ (négligeable)
Ex : 1 ns pour 74F175

L'horloge doit être stable aussi

Ex : pour une montre, on emploie un cristal de quartz.

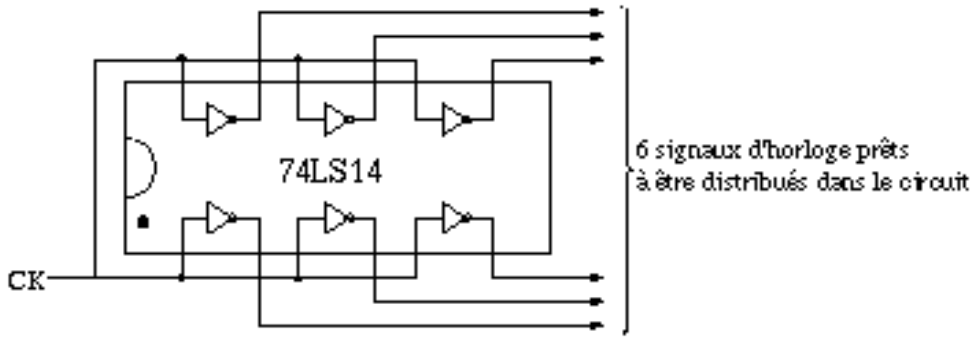
Circuit de communication à 9600 bits par seconde (BPS), on emploie un cristal de 11.059 Mhz (dans le microcontrôleur 8051, par exemple).

Fan-Out (Distribution Maximale)

Puisque l'horloge est distribuée à tous les FF, l'horloge peut se dégrader.

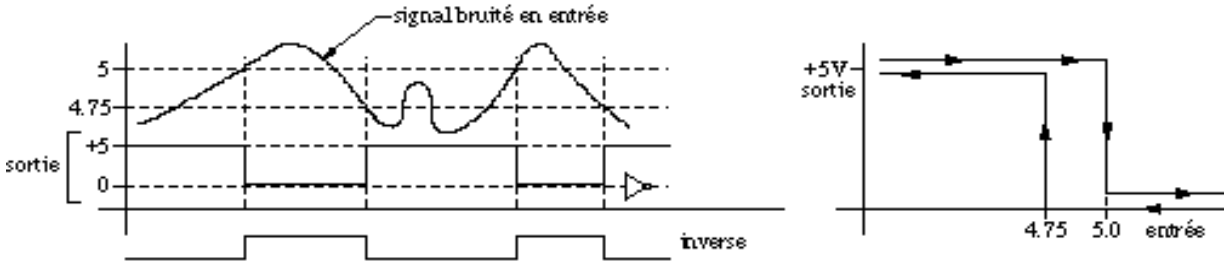
Solutions :

1) Circuit spécial d'horloge qui fournit beaucoup de courant.
2) Employer des buffers comme suit :



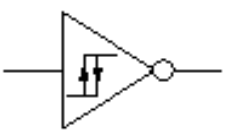
- l'inversion n'affecte pas le système
- permet de maintenir un "fast rise time" (horloge n'est pas "skewed")

3) Emploi des Schimitt trigger : "transform slowly changing input signals into sharply defined signals". Il y a donc une bonne immunité au bruit.



(tiré de p. 231 - *The Art of Electronics*)

symbole :



6.4 Étapes d'analyse et de design

A- Étapes d'analyse (circuit synchrone avec Flips Flops, sans entrée)

1. Séparer les Flips Flops et la logique combinatoire de stimulation.
2. Étudier la logique de stimulation et obtenir une équation booléenne pour chacune des entrées des Flips Flops.
3. Assumer qu'au début, le circuit est dans un état zéro, donc, mettre "0" dans les équations booléennes.
4. Obtenir les nouvelles valeurs de sortie et les mettre dans les équations pour déterminer l'état suivant.
5. Reprendre l'étape 4 jusqu'à ce que tous les états suivants soient déterminés. Indiquer les états dans des tables de Karnaugh, diagramme en anneaux.

B- Étapes de design (circuit synchrone avec Flips Flops, sans entrée)

1. Exprimer chaque nombre de la séquence en binaire. Chaque nombre correspondra à un État.
2. Déterminer une méthode pour assigner les États M aux sorties des Flips Flops.
3. Établir la table État-Présent $\Rightarrow$ État Suivant.
4. Choisir le type de Flip Flop : T, JK, D (D par défaut).
5. Établir les relations entre les États.
6. Employer les États présents comme entrées dans les tables de vérité.
7. Assigner l'état de retour aux états inemployés.
8. Trouver les équations à partir des tables de vérité du point 6.
9. Monter les sorties comme combinaison des sorties des Flips Flops.
10. Réaliser la logique combinatoire de stimulation à partir des équations.

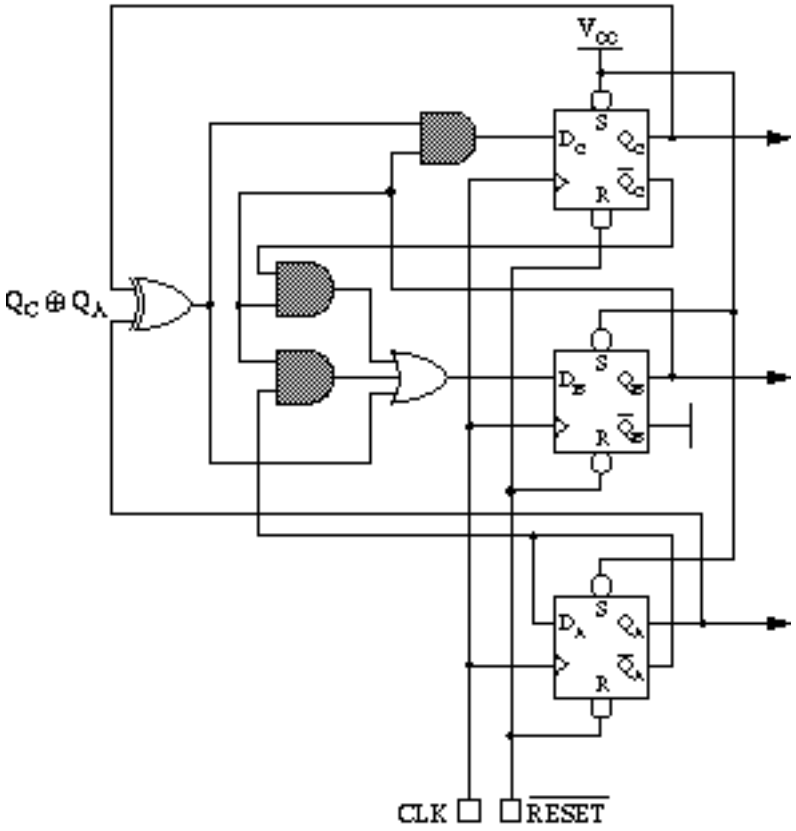
C- Étapes de design (registres à décalage cyclique – cyclic shift register)

1. Établir la liste successive des états dans la table ÉTAT PRÉSENT $\rightarrow$ ÉTAT SUIVANT *(Present State ® Nest State)*.
2. Ajouter le(s) cas d'initialisation pour le démarrage automatique (e.g. 000 000).
3. Synthétiser l'Entrée série (*Serialln*) comme la somme des mintermes pour lesquels *Serialln* = 1 dans la liste successive des états.

Table 6.1 State Changes for Three Flip-Flops (Extrait de "Digital Design from Zero to One" p.312)

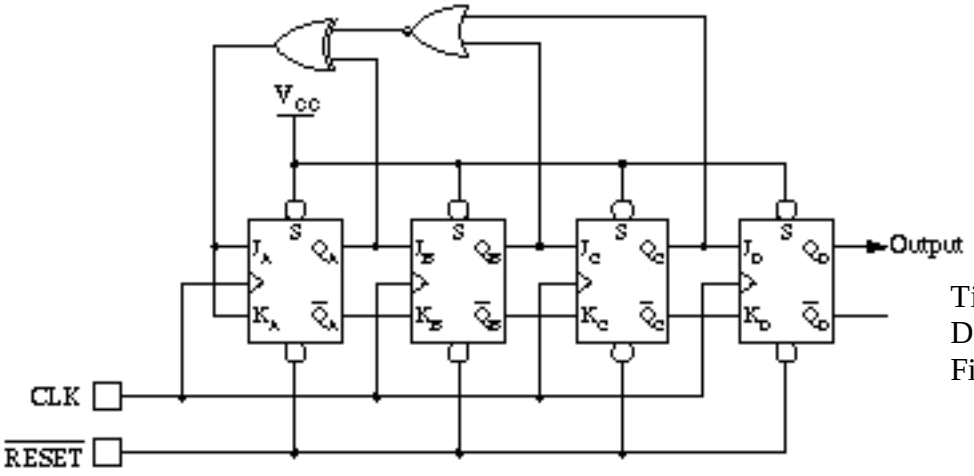
Q_{N+1}	Q_N	D Flip-Flop	Toggle Flip-Flop	JK Flip-Flop			
		D	ENABLE	J	K	J	K
0	0	0	0	0	1	0	X
				0	0		
0	1	1	1	1	0	1	X
				1	1		
1	0	0	1	0	1	X	1
				1	1		
1	1	1	0	0	0	X	0
				1	0		

Exemple 1 : (sera fait en classe)



Tiré de "Digital Design", J. D. Daniels, 1ere édition, p.313, Fig. 6.10

Exemple 2 : (sera fait en classe)



Tiré de "Digital Design", J. D. Daniels, 1ere édition, p.315, Fig. 6.12

6.5 Design de compteurs synchrones

- Maintenant : étant donné la séquence → quel est le design ?
- À considérer : • minimisation/optimisation des états
  • choix du Flip Flop (habituellement "D")
  • assignation des noms des variables
  • éviter les états non désirés "lock out" Auto-correcteur

Considérations pratiques

1. S'il y a M états ⇒ au moins N Flip Flop, $2^N \geq M$
2. Parfois pratique d'avoir + d'état que le minimum.
3. Cas extrême : 1 FF/état
4. Prévoir un état retour ou "ground state" : point de retour pour tous les états inemployés. Souvent 0000.
   Si États inemployés = "X" dans le design des tables, pas sûr qu'on pourra revenir en cas de défaillance (attention !).

   Exemple : Faire le design d'un compteur que répète la séquence 0, 3, 6, 9, 12, 0...
   Employer le code binaire pour le codage des FF D, employer les sorties des FF comme sortie.
   Ces spécifications ne donnent pas beaucoup de choix au designer.

Étape 1 et 2 : La sortie des Flips Flops est déterminée par les spécifications.

#	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
3	0	0	1	1
6	0	1	1	0
9	1	0	0	1
12	1	1	0	0

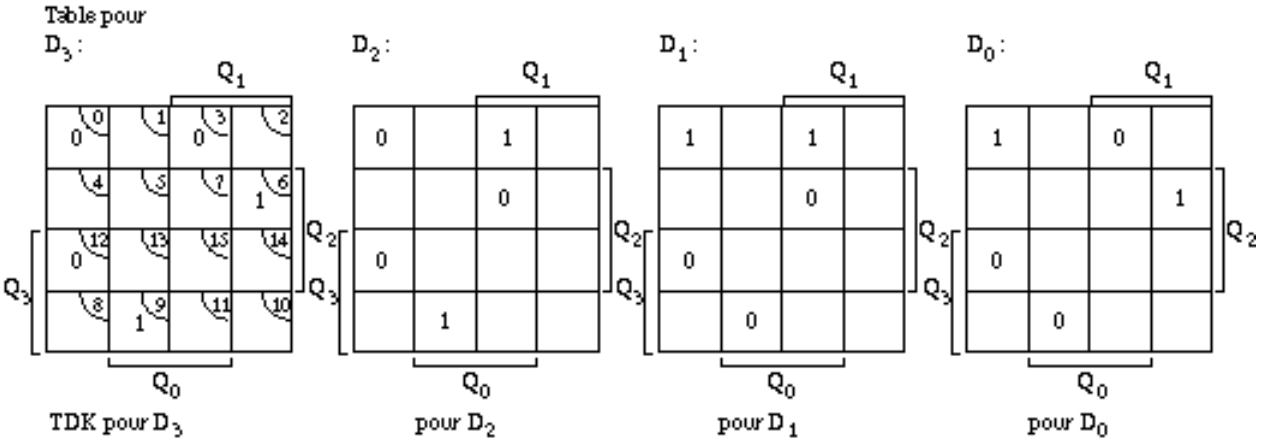
Étape 3 :

	État présent					État suivant							
	Q_3	Q_2	Q_1	Q_0		Q_3	Q_2	Q_1	Q_0	D_3	D_2	D_1	D_0
0	0	0	0	0	→	0	0	1	1	0	0	1	1
3	0	0	1	1	→	0	1	1	0	0	1	1	0
6	0	1	1	0	→	1	0	0	1	1	0	0	1
9	1	0	0	1	→	1	1	0	0	1	1	0	0
12	1	1	0	0	→	0	0	0	0	0	0	0	0

Étape 4 : On emploie FF – D (Avantage des FF-D, le design est plus aisé).

Étape 5 : Relation entre les états : facile avec FF-D.
Si $Q_3Q_2Q_1Q_0 = 0000$ et $D_3D_2D_1D_0 = 0011$
L'état futur sera $Q_3Q_2Q_1Q_0 = 0011$

Étape 6 : Obtenu de la table de l'étape 3 :

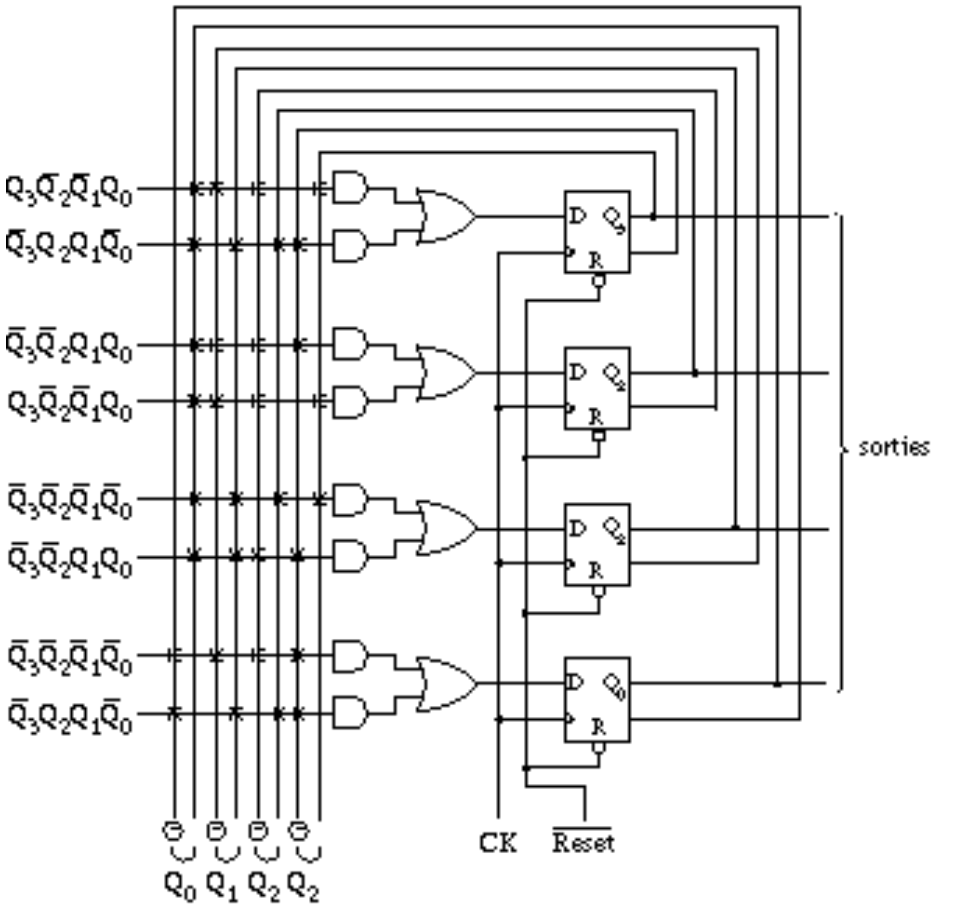


Étape 7 : Il est sage de laisser des zéros dans les tables de Karnaugh (T-K) précédentes pour avoir un état de retour connu "ground state".

Étape 8 : On obtient les équations.

$$D_0 = \overline{Q}_3 Q_2 Q_1 \overline{Q}_0 + \overline{Q}_3 \overline{Q}_2 \overline{Q}_1 \overline{Q}_0$$
$$D_1 = \overline{Q}_3 \overline{Q}_2 Q_1 Q_0 + \overline{Q}_3 \overline{Q}_2 \overline{Q}_1 \overline{Q}_0$$
$$D_2 = \overline{Q}_3 \overline{Q}_2 Q_1 Q_0 + Q_3 \overline{Q}_2 \overline{Q}_1 Q_0$$
$$D_3 = \overline{Q}_3 Q_2 Q_1 \overline{Q}_0 + Q_3 \overline{Q}_2 \overline{Q}_1 Q_0$$

Étape 9 : Les sorties sont directement les sorties des FF.



On a choisi une représentation genre PAL ici.

6.6 Compteurs à codage positionnel

- Ce sont des compteurs dont la sortie s'incrémente ou décrémente à chaque coup d'horloge.
- État des Flips Flops directement à la sortie du circuit.

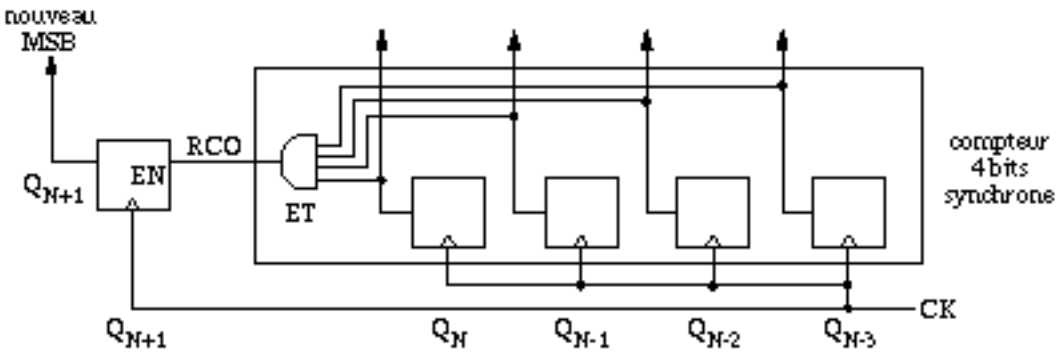
6.7 Compteur à décade décrémental avec FF-D Toggle

Ce sont des compteurs de type 9---0---9---0---9 : 10 états ⇒ compteur à décade.

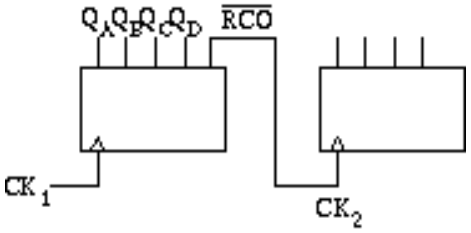
6.8 Compte supérieur en employant le "Ripple Carry output" (RCO)

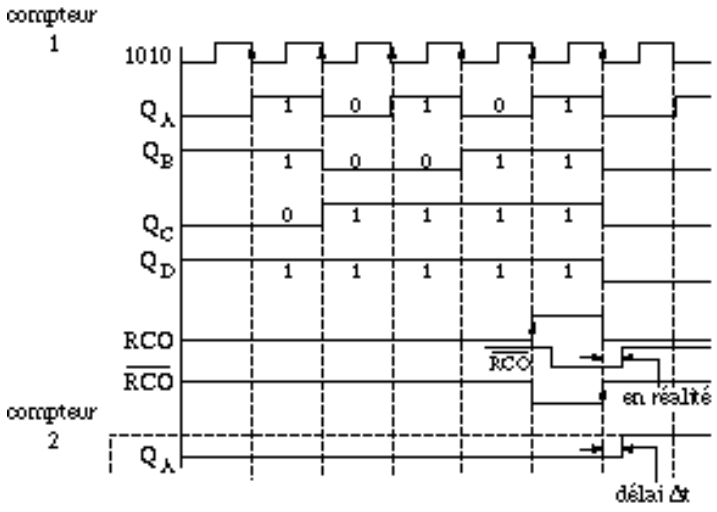
- On peut réaliser des compteurs à 8, 12, 16 bits avec la méthode précédente, mais le coût est élevé à cause de la logique d'excitation combinatoire ⇒ approche pas intéressante.
- Approche plus intéressante : cascade de circuits de compteur MSI (Medium Scale Integration). Exemple : CB4CLED de Xilinx.
  CB4CLEB : compteur 4 bits up/down avec external load, 2 Enable. Permet de réaliser en cascadant N unités, des compteurs à 4 N états.
- Le $\overline{\text{Rco}}$ peut être employé pour créer un bit supplémentaire en l'envoyant à un "toggle enable" d'un FF externe.

Exemple :



- Que faire pour cascader des compteurs 4 bits ?
  On a pas accès au contrôle de l'entrée du 1er FF
  Si on a seulement accès à l'horloge, alors on propage RCO sur CK2



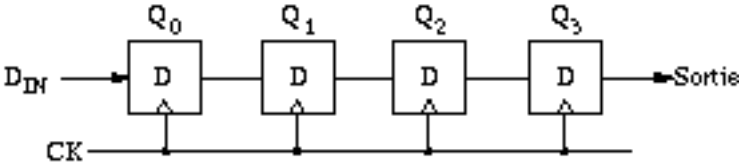


- On voit qu'il faut employer $\overline{Rco}$ pour des compteurs ↑ positive edge triggened.
- Il y aura 1 délai Δ = tpd (temps propagation dans porte ET synthèse $\overline{Rco}$.
- Approche moins pire qu'un "full ripple" compteur avec 8 x tpd (ici 1 seul tpd).

6.9 Registre à décalage pour le design de compteurs synchrones

- On va voir quelles sont les séquences disponibles avec un "Shift register" (MSI).

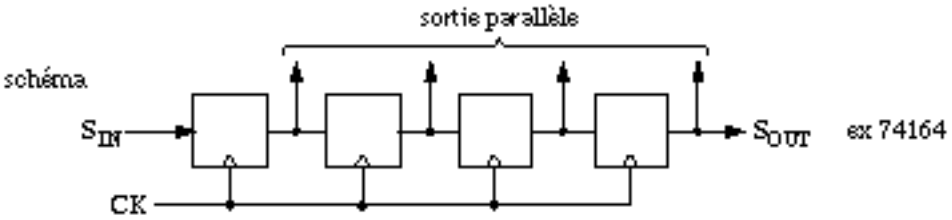
Déf : ensemble de bascules "D" edge-triggened connectées dos à dos, horloge commune.



D_{IN} = vient de l'extérieur ou peut-être généré de l'intérieur (cas des compteurs).

Pour D_M, $M \neq 0$ on a D_M = Qm-1, donc il y a des limitations aux séquences disponibles.

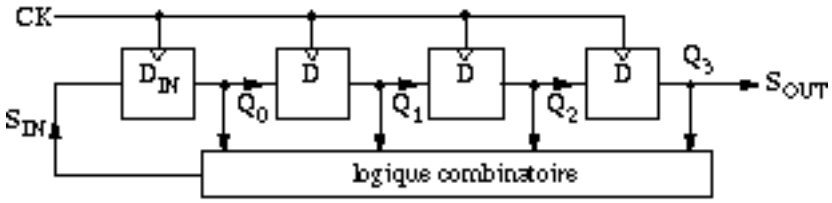
6.10 Serial In – Parrallel Out shift register (SR) ou SI PO



Registre à décalage cyclique (RDC)

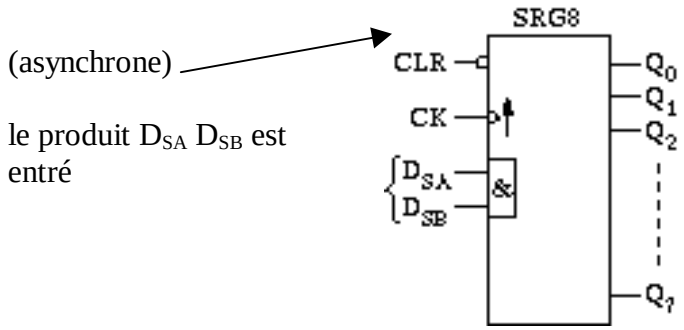
On a aussi des "cyclic SR" *SIPO pour lequel la Serial input est une combinaison des sorties des FF.

* pas d'entrée externe admise.



Registres
- série – série SI SO
- série – parallèle SI PO
- parallèle – série PI SO
- parallèle – parallèle PI PO

Exemple circuit TTL #74164 (SI PO) avec mise à zéro asynchrone



Shift register 8 bits

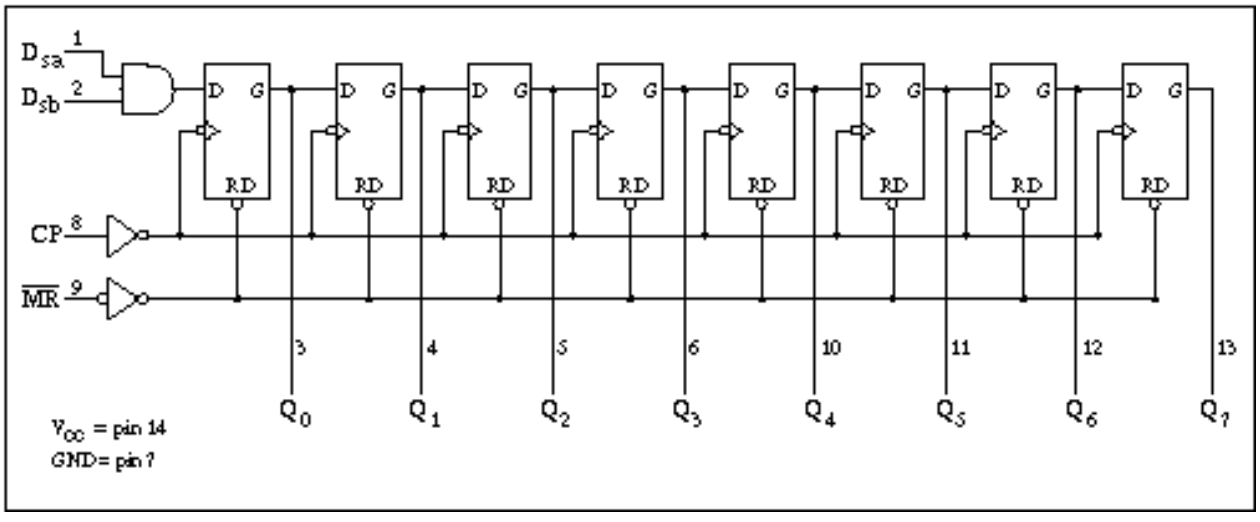
Registre à 8 bits :

Table :

$\overline{CLR}$	CK	D_{SA}	D_{SB}	Q_0	Q_1	…	Q_7	Mode
L	X	X	X	L	L	…	L	Clear
H	↑	L	L	L	Q_0	…	Q_6	
H	↑	L	H	L	Q_0	…	Q_6	Shift
H	↑	H	L	L	Q_0	…	Q_6	
H	↑	H	H	H	Q_0	…	Q_6	

8-bi serial-in parallel-out shift register

LOGIC DIAGRAM

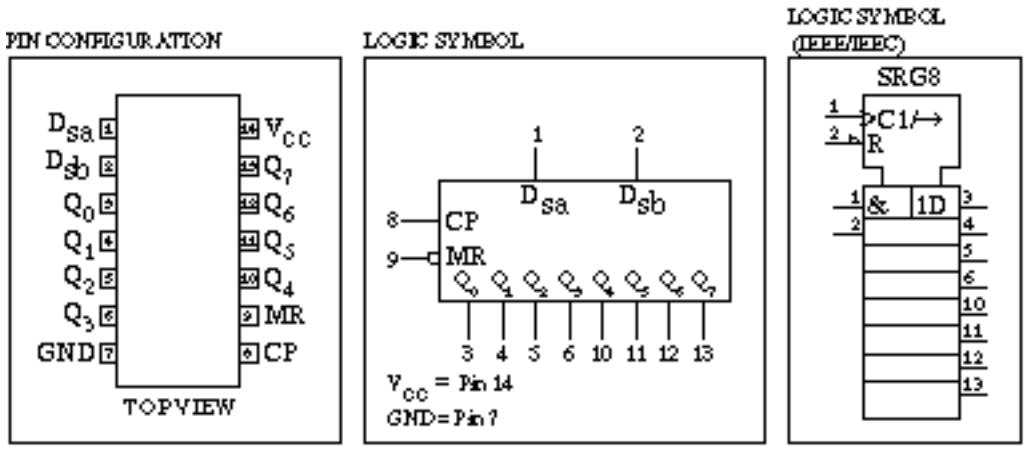


Note : MR = Master Reset

FUNCTION TABLE

INPUTS				OUTPUTS			OPERATING MODE
MR	CP	D_{sa}	D_{sb}	Q_0	Q_1….	Q_7	
L	X	X	X	L	L	L	Reset (clear)
H	↑	l	l	L	q_0	q_6	Shift
H	↑	l	h	L	q_0	q_6	
H	↑	h	l	L	q_0	q_6	
H	↑	h	h	H	q_0	q_6	

H = High voltage level
h = High voltage level one set-up time prior to the Low-to-High clock transition
L = Low voltage level
l = Low voltage level one set-up time prior to the Low-to-High clock transition
Q_n = Lower case letters indicate the state of the referenced input (or output) on setup time prior to the Low-to-High clock transition
X = Don't care
↑ = Low-to-High clock transition



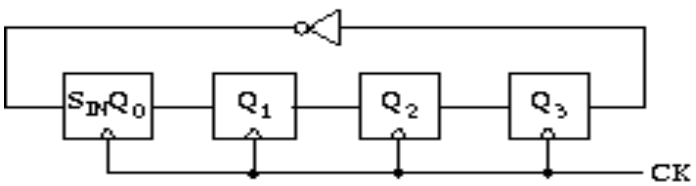
6.11 Analyse des Registres SR Cycliques [CSR] cyclic Shift Register

On décode comme ceci : $Qi \rightarrow Qi + 1$, $i = 0, \ldots N-1$ N = Nb de bits
$Qo = f(Qi, i = 0, \ldots N) \leq$ fonction booléenne

Après un certain nb de cycles, $\left(\leq 2^N\right)$ la séquence recommence

- Pour déterminer la séquence : { - simulation
  - la faire à la maison

Ex : CSR



Très simple
Un simple inverseur constitue la logique de Sin
Quelle est la séquence ?
On commence avec '0000' (contenu des FF).
Solution : le SR va se remplir de 1 puis se vider avec au total 8 états

	Q_1	Q_2	Q_3	Q_4
1-	0	0	0	0
2-	1	0	0	0
3-	1	1	0	0
4-	1	1	1	0
5-	1	1	1	1
6-	0	1	1	1
7-	0	0	1	1
8-	0	0	0	1

8 états

Si on commence à zéro
Séquence de MOEBIUS

État suivant diffère d'un seul bit (code gray)

Q_1	Q_2	Q_3	Q_4
0	1	0	1
0	0	1	0
1	0	0	1
0	1	0	0
1	0	1	0
1	1	0	1
0	1	1	0
1	0	1	1
0	1	0	1

Si on commence ailleurs
Séquence ANTI-MOEBIUS

Tous les FF changent après le coup d'horloge sauf 1

Switch – tailed
Ring compteur

Design avec les CSR

- On peut implanter toute séquence qui "se déroule avec un patron décalé" (*that un folds with a horizontal shift pattern*).
- En faisant le design, voir si on peut avoir un patron qui se déroule à gauche ou à droite.

Ex : - 5 jointures/articulations d'un robot qui doivent être pliées en séquence comme suit :

A	B	C	D	E	Serial in
1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	1

Solution :

- On emploie 5 FF, chaque sortie FF commande une articulation, un joint.
- On aurait pu y aller avec 3FF ($2^3 = 8 > 5$ états requis) mais avec 2 FF de plus, aucun décodage de sortie !

- Quelle est la logique combinatoire que va faire circuler un 1 ?
- Une approche : Considérons les 5 sorties $\Rightarrow 2^5 = 32$ cas
  De ceux-ci les cas :

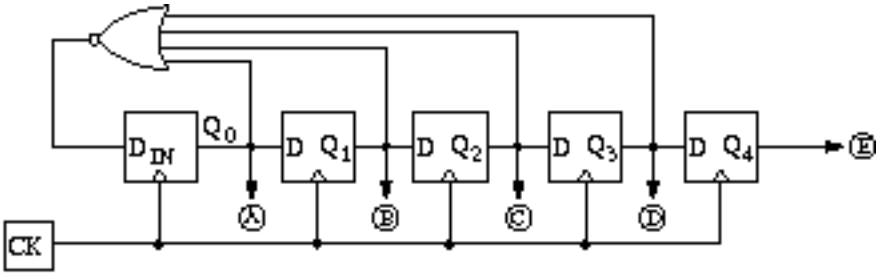
	A	B	C	D	E		
	Q_0	Q_1	Q_2	Q_3	Q_4	SIN	
On recommence la séquence ⟵	0	0	0	0	1	1	ABCDE
Cas d'un reset ⟵	0	0	0	0	0	1	0000X

On peut synthétiser Sin directement avec :

Sin = $\overline{Q_0 + Q_1 + Q_2 + Q_3}$, Q_4 n'a pas d'importance ici ($Q_4 = 1$ ou $Q_4 = 0$ et SIN = 1)

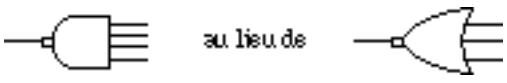
= $\overline{Q_0} \cdot \overline{Q_1} \cdot \overline{Q_2} \cdot \overline{Q_3}$

On a alors le design suivant :



Il faut A = B = C = D = 0 pour avoir D_{IN} mis à 1.

- Avec ce design peut importe l'état des FF, après 5 coups d'horloge au maximum, un 1 va circuler dans le système (tous les 1 sont réinjectés à 0 à cause de la porte NOR).
- Q_4 peut être vu comme un témoin qui avertit lors qu'on a accompli un cycle, peut servir à arrêter le système.
- Ce système est appelé "one-hot sequencer".
  Assigne 1 FF par état $\Rightarrow$ pas efficace comme usage des FF
  Très employés.
- Si on avait voulu faire circuler un "0" au lieu d'un 1
  a) mettre des inverseurs c'est un "one-cold sequencer"
  b) employer une porte NAND (NAND = 0 lorsque toutes les entrées sont 1).



Q_0	Q_1	Q_2	Q_3	Q_4		SIN	
1	1	1	1	1	$\Rightarrow$	0	- reset
1	1	1	1	1	$\Rightarrow$	0	- début nouveau cycle

- On peut faire circuler plus qu'un 1 à la fois, mais n'importe quel mot d'un état à la fois.

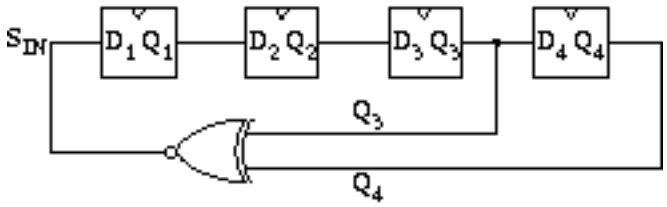
Généralisons la méthode de design :

1- Établir la liste de la succession des états présent → états futurs incluant le "Serial in".
2- Ajouter le (les) cas d'initialisation pour le "démarrage automatique" ex : 0000 0000.
3- Synthétiser Serial in
   Serial in = la somme des mintermes pour lesquels Serial in = 1 dans la liste successive des états.

6.12 Générateur pseudo-aléatoire

C'est un CSR avec un grand nombre de bascules. Le compteur passe dans beaucoup d'états (2^N au maximum).

Utilité en cryptage de séquence série
Utilité en génération de bruits répétitif



Passe dans 15 des 16 états sauf 1111 qui est un cul-de-sac.

On a :

			PS		
	Q_1	Q_2	Q_3	Q_4	Sin
0)	0	0	0	0	1
8)	1	0	0	0	1
12)	1	1	0	0	1
14)	1	1	1	0	0
7)	0	1	1	1	1
11)	1	0	1	1	1
13)	1	1	0	1	0
6)	0	1	1	0	0
3)	0	0	1	1	1
9)	1	0	0	1	0
4)	0	1	0	0	1
10)	1	0	1	0	0
5)	0	1	0	1	0
2)	0	0	1	0	0
1)	0	0	0	1	0
0)	0	0	0	0	1

Q_1	Q_2	Q_3	Q_4
1	1	1	1
1	1	1	1

Q_3	Q_4	Sin
0	0	1
0	1	0
1	0	0
1	1	1

1111 est un cul-de-sac ici !

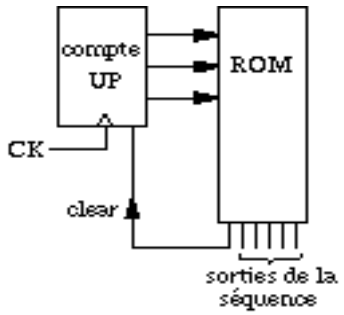
La sortie du générateur pseudo-aléatoire peut-être n'importe quel sortie des FF puisque le patron n'est pas important ici.

6.13 Unités cachées

Elles peuvent parfois résoudre le problème lors de la répétition des mêmes nombres.

Exemple : séquence $5 \rightarrow 5 \rightarrow 4 \rightarrow 2$ puis $5 \rightarrow 5 \rightarrow 4 \rightarrow 2$ …

6.14 Compteur + ROM pour créer une séquence arbitraire



$2^M \geq N$

N = longueur de la séquence
M = nb de bits dans le compteur

Si N n'est pas une puissance de 2, il faut prévoir un mécanisme de rechargement du compteur ou effacement avec CLR synchrone qui peut être commandé avec une des lignes du ROM

On peut aussi employer un contrôleur microprogrammé qui permet en plus de réaliser : sauts, sous-routines, boucles.

Définition : Circuit séquentiel qui assure la fonction de comptage (binaire ou autre) avec 2^N états sans aucun décodeur de sortie. Sorties = état des FF à chaque coup d'horloge.

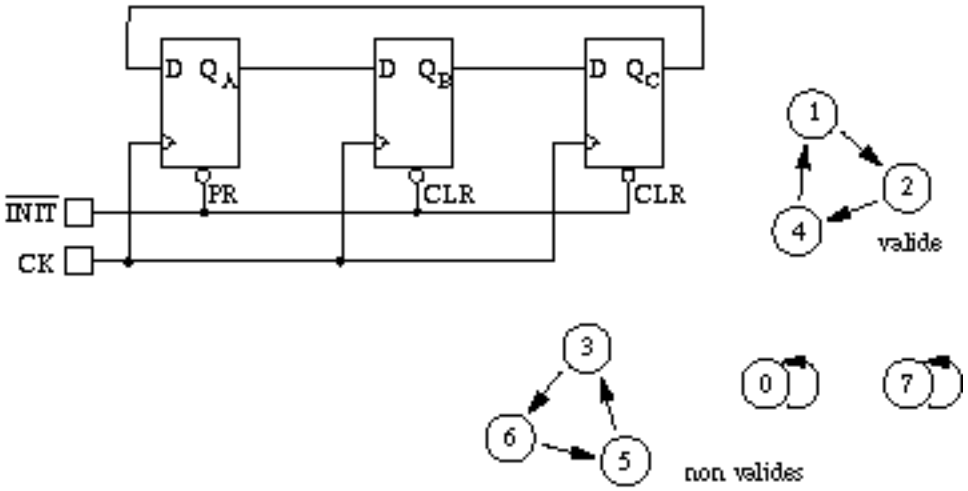
6.15 Types de compteur

1- Ripple (série) ou à débordement (asynchrone)

2- Binaire :
- up/down, sortie en binaire
- ck commune
- la longueur de la séquence est le modulo

3- En anneau :
- fait circuler un 1 ou un 0
- possède autant d'états que de FF
- plusieurs cycles sont réalisés, un seul est valide $\Rightarrow$ initialisation requise

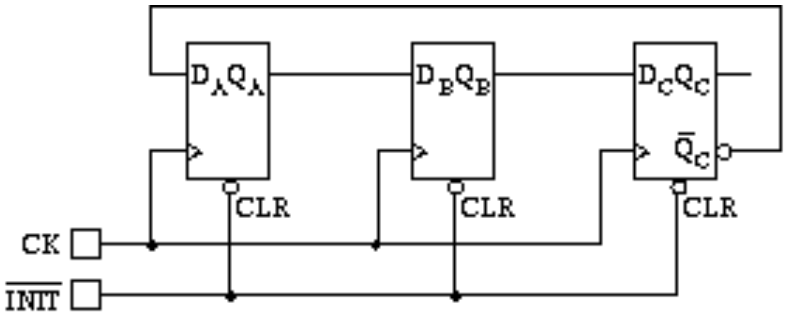
Exemple : one-hot, two-hot
one-cold, two-cold
sequencer



4- Johnson

- similaire au compteur en anneau
- offre 2 fois plus d'états que de FF grâce à un décodage simple.

Ex :



Exemple de circuit 74163

5. Anneau à séquence arbitraire

Q_A	Q_B	Q_C	Décodeur d'état
0	0	0	$\overline{Q}_A\overline{Q}_C$
1	0	0	$Q_A\overline{Q}_B$
1	1	0	$Q_B\overline{Q}_C$
1	1	1	Q_AQ_C
0	1	1	$\overline{Q}_AQ_B$
0	0	1	$\overline{Q}_BQ_C$

Si K bits ⇒ séquence de 2K états

Problème les états inemployés ⇒ lock-out (cul-de-sac) :

PS				NS					
Q_A	Q_B	Q_C		D_A	D_B	D_C	Q_A	Q_B	Q_C
0	1	0	→	1	0	1	1	0	1
1	0	1	→	0	1	0	0	1	0

- Caractéristique : Le dernier bit inversé est injecté en D_A
- Appelés : "Switch-tailed ring" compteurs.

6- Compteurs à séquence quelconqu