Electronique numérique

LIE CNED 2eme année.

Analyse et synthèse des systèmes séquentiels

Frédéric LECOMTE, Luc MUSEUR

Université Paris 13, Institut Galilée.

Chapitre 1

Introduction

1.1. *Un peu de vocabulaire.*

Les automates que nous allons étudier dans ce cours n'ont qu'un rapport très éloigné avec les machines mécaniques fabriquées depuis le 17 siècle ; ce sont des objets abstraits dont le nom varie selon les domaines dans lesquels ils sont utilisés. Les automaticiens parlent *d'automates finis*, les concepteurs de circuits numériques de *machines à nombres finis d'états* et les concepteurs d'unités centrales d'ordinateurs de *séquenceurs*.

1.2. *Où trouve-t-on des automates?*

A peu près partout, la plus part des systèmes de commandes sont des automates. On peut citer, par exemple, de façon assez évidente :

Le système de commande d'un ascenseur.

Le système de régulation des feux à un carrefour

Le digicode contrôlant l'ouverture d'une porte

On utilise des automates dans tous les systèmes dont le fonctionnement peut être décrit par une **chronologie faisant intervenir un nombre fini d'opérations distinctes conditionnées par des entrées extérieures**. Les applications typiques sont :

La gestion d'un processus.

L'utilisation et/ou le partage d'une ressource.

Le traitement de données.

1.3. *Un 1er exemple d'automate : La commande d'un monte charge.*

1.3.1. Le cahier des charges.

On considère un monte charge effectuant la liaison entre les 2 niveaux d'un bâtiment (figure 1). Le moteur est commandé par 2 signaux *M* (pour monter) et *D* (pour descendre). Ces commandes sont élaborées à partir des entrées :

B : bouton poussoir (*B*=1 si le bouton est enfoncé)

*p*0 et *p*1 : détecteurs de position aux niveaux 0 et 1 (*pi*=1 si le monte charge est présent au niveau *i*)

Le fonctionnement du système est décrit par le cahier des charges suivant :

1. Quand la cabine est à un étage si *B* est inactif elle s'arrête, si *B* est actif elle change d'étage.

2. Quand la cabine est entre 2 étages la dernière commande (*M* ou *D*) est maintenue.

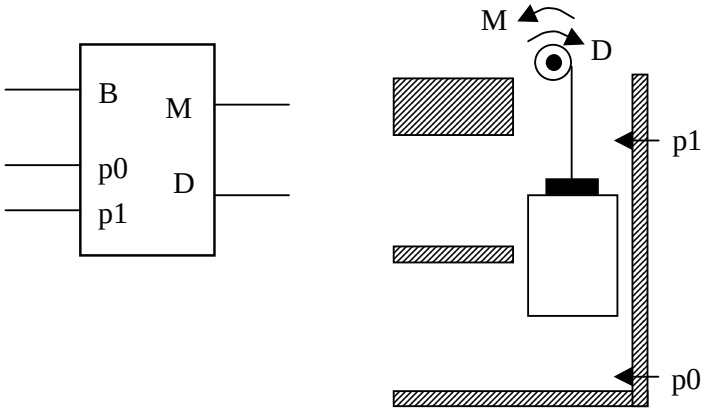


figure 1

1.3.2. Le diagramme d'états.

Le fonctionnement du monte charge peut être décrit sous la forme d'un diagramme (figure 2), que l'on appelle diagramme d'états[1].

[1]En automatique on utilise une approche semblable et les diagrammes d'états s'appellent des GRAFCET

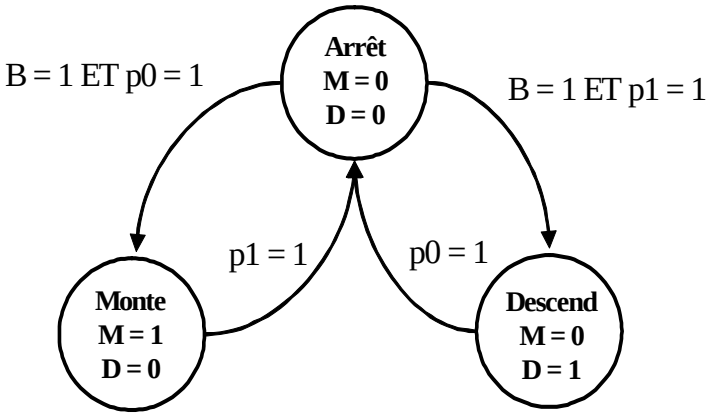


figure 2

On retrouve sur ce diagramme les principes importants des automates :

1. La solution du problème est décrite sous forme d'étapes bien définies et bien séparées les unes des autres.

2. Chaque étape est représentée par un ensemble fini de valeurs que l'on appelle l'état interne de l'automate Cet état est physiquement matérialisé par les valeurs contenues dans une mémoire. **Attention cette mémoire représente l'état de l'automate et non celui du système commandé**. Dans l'exemple du monte charge, aucun des états n'indique la position de la cabine.

3. Les entrées du système provoquent des transitions entre les états. Une transition n'est réalisée que si le système est dans l'état de départ de la transition considérée et que la ou les conditions extérieures sont vérifiées.

4. Si aucune des transitions issues d'un état n'est active, toutes les conditions étant fausses, alors le système reste dans l'état considéré. Par défaut il y a mémorisation de l'état présent.

1.4. *La conception des circuits numériques.*

1.4.1. Structure générale des automates.

La figure 3 donne la structure générale d'un automate pouvant réaliser n'importe quelle fonction logique séquentielle synchrone. Nous reviendrons sur ce type d'architecture dans la suite du cours. Pour le moment, il suffit de savoir que le registre d'état est une assemblée de bascules passant d'un état à un autre, en fonction des entrées, à chaque impulsion du signal d'horloge.

Le registre constitue le coeur de l'automate. Ces sorties constituent l'état présent de l'automate. Les parties G et F sont des blocs combinatoires calculant, respectivement, les sorties et l'état futur de l'automate.

Etats futurs = F(état présent, entrées)

sorties = G(état présent) ou G(état présent, entrées)

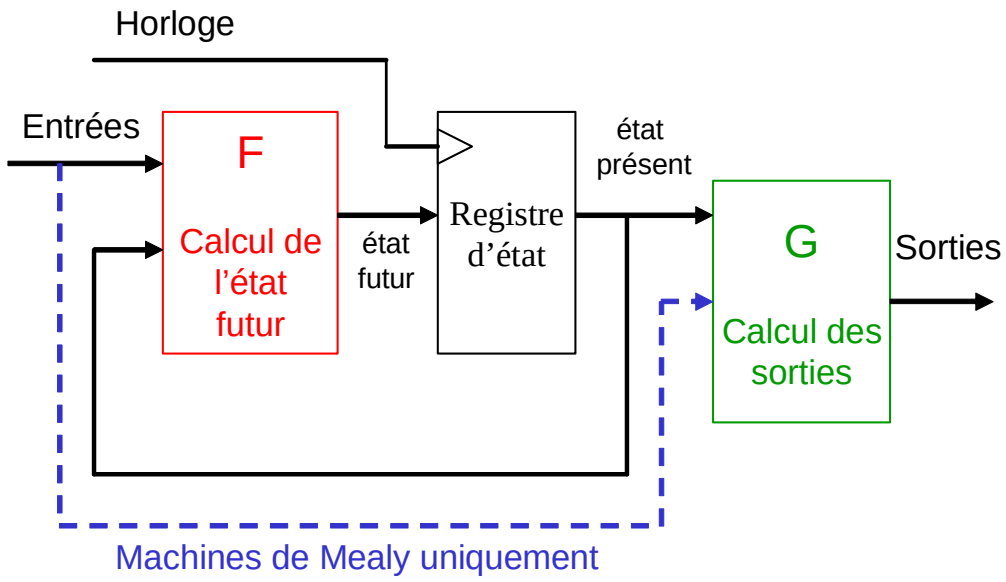


figure 3 Structure générale des automates séquentiels

En pratique on distingue deux types d'architectures : **les machines de Moore et les machines de Mealy**. Dans les machines de Moore les sorties ne dépendent que de l'état présent. Au contraire dans les machines de Mealy les sorties dépendent à la fois de l'état présent et des entrées du système.

Avant de voir plus en détails la conception et la synthèse des automates câblés, il est indispensable de savoir analyser un circuit séquentiel, autrement dit d'être capable de comprendre le fonctionnement d'un circuit en étudiant son schéma. C'est l'objet du second chapitre.

1.4.2. Réalisation physique des automates.

Aujourd'hui, pour fabriquer un automate on n'utilise quasiment plus les circuits intégrés réalisés en technologie *SSI* (Small Scale Integration, la fameuse série 7400). Cette technologie, très en vogue dans les années 1970, présente de nombreux inconvénients :

Chaque composant ne comporte que quelques portes logiques.

Pour changer de fonction logique il faut changer de composant et refaire le circuit.

Une fois le circuit réalisé les connexions sont fixes.

Depuis une quinzaine d'années les fabricants proposent des circuits qui intègrent un grand nombre de portes logiques dont les interconnexions ne sont pas fixes. Ces composants sont donc reconfigurables à volonté par l'utilisateur. Les deux grandes familles actuelles sont :

Les circuits programmables par l'utilisateur. (*PLD*[2])

Les circuits intégrés spécifiques (ASIC[3])

De nombreux outils de conception assistée par ordinateur ont été développés pour utiliser ces composants. Ils libèrent le concepteur des taches les plus fastidieuses et lui permettent de se concentrer sur les points importants du processus de conception. Le chapitre 4 présente les principales caractéristiques de ces composants.

[2] Programmable Logic Device

[3] Application Specific Integrated Circuit

Chapitre 2

Analyse des systèmes séquentiels.

2.1. *Modélisation des systèmes séquentiels asynchrones.*

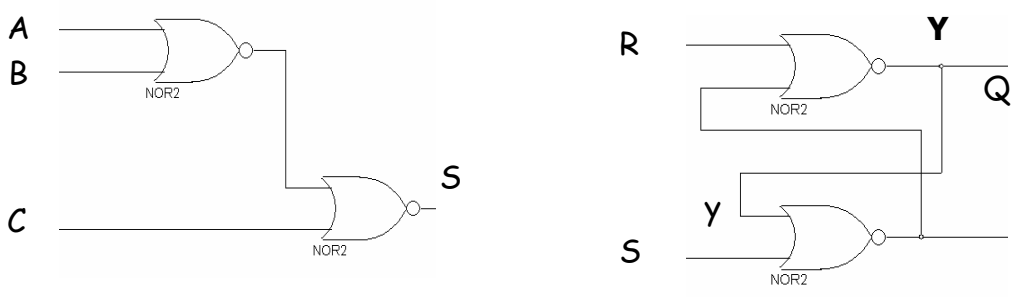
Autant il est relativement facile de trouver l'état stationnaire d'un système séquentiel, autant il est difficile de se représenter son évolution. Il est très utile de disposer d'un modèle. Nous commencerons par nous intéresser aux circuits asynchrones. Pour mémoire, il s'agit des circuits dont les entrées sont sensibles à des niveaux de tension (0 ou 1 par opposition aux transitions ↑ ou ↓)

2.1.1. Retour sur les systèmes combinatoires et séquentiels.

Dans un système combinatoire les différentes grandeurs logiques peuvent se calculer de proche en proche, en partant des entrées et en se dirigeant vers les sorties (voir la figure 4.a). A un instant donné, l'état du système est entièrement déterminé par les valeurs des entrées, mais aussi par les retards qu'introduisent les différents composants[4]. Une modification des entrées n'entraîne pas un changement instantané de la sortie!

Un circuit séquentiel est un circuit dans lequel les grandeurs logiques (*y* par exemple) ne peuvent pas toujours être calculées de proche en proche, du fait d'un certain nombre de bouclage qui ramènent la sortie vers l'entrée de un ou plusieurs circuits. Sur la figure 4.b La connaissance de *y* est indispensable au calcul de *Q*, mais *y* n'est pas déterminée uniquement par *R* et *S* : *y* est devenue une variable indépendante, on dira que c'est **une variable secondaire**. Le problème est que c'est une variable indépendante spéciale puisqu'elle influe sur sa propre valeur par l'intermédiaire de *Q*!

[4] Les retards sont liès au temps de propagation du signal à travers le circuits.



a) b)

figure 4 a) exemple de circuit combinatoire. b) exemple de circuit séquentiel

2.1.2. Un modèle de système séquentiel pour les circuits asynchrones.

Peut-on attribuer une valeur à une variable secondaire, telle que *y*, qui est elle même générée par le système? La réponse est oui, grâce précisément aux retards. Il existe un temps T pendant lequel la variable secondaire ne subit pas le contre-coup du bouclage (c'est à dire que le système n'a pas encore eu le temps d'agir sur elle), et constitue donc bien une variable indépendante.

On peut modéliser un circuit séquentiel par le schéma représenté sur la figure 5. Les variables secondaires et les entrées commandent un ensemble combinatoire "instantané", c'est à dire supposé dépourvu de tout retard ; c'est le bloc C qui génère les excitations secondaires *Y*. Ces dernières, après un certain retard symbolisé par le bloc *T*, fournissent les nouvelles valeurs des variables secondaires *y* et ainsi de suite :

$$y(t)=Y(t-T)$$

Les excitations secondaires Y représentent les valeurs futures des variables secondaires y. En général, dans ce polycopié, on utilise plutôt les termes valeur future pour Y et valeur présente pour y.

Les sorties sont fabriquées directement à partir des variables secondaires et des entrées (circuit C' et retard T' sur la figure 5)

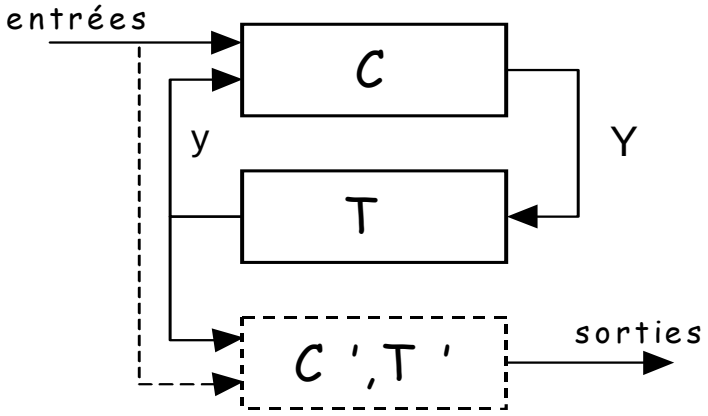


figure **5** Modélisation d'un système séquentiel

2.1.3. Etat présent, état futur.

Le modèle précédent nous permet d'introduire les notions d'état présent et d'état futur. Quand nous parlerons de l'état d'un système il s'agira de l'état présent.

L'état présent est défini par la valeur des variables secondaires. La donnée de l'état présent et des entrées détermine entièrement le système, y compris sont évolution future. En particulier les sorties se calculent à partir de l'état présent et des entrées.

L'état futur est défini par la valeur des excitations secondaires. C'est à dire par les futures valeurs des variables secondaires. Rappelons que les excitations secondaires se calculent à partir des variables secondaires et des entrées en utilisant les expressions combinatoires des circuits C.

L'état futur devient le nouvel état présent au bout d'un temps de l'ordre du plus grand des retards *T* utilisés dans le modèle. Si l'on ne prend en compte que les retards induits par la propagation du signal, alors les limites sont imposées par la technologie.

Le système évolue jusqu'à ce que l'état présent et l'état futur soient identiques, on dit alors que le système est dans un **état stable**. A l'inverse, si le système ne trouve pas d'état stable, il oscille indéfiniment entre plusieurs états.

2.1.4. Un exemple simple : la bascule RS.

Reprenons l'exemple de la bascule RS à base de portes NOR. Choisissons comme variable secondaire la deuxième entrée de la porte *S*, c'est à dire aussi la sortie *Q* (voir la figure 6). L'excitation secondaire *Y* est égale à , et *y* et *Q* deviennent égales à *Y* au bout d'un temps égal au temps de propagation du signal à travers les deux portes NOR. On note souvent l'excitation secondaire *Y* directement sur le schéma électronique (ici à la place de *Q*) pour symboliser l'état futur ; mais attention, il ne faut pas perdre de vue que *Q* et *y* sont des

grandeurs réelles alors que *Y* n'existe que dans notre modèle de représentation des systèmes séquentiels

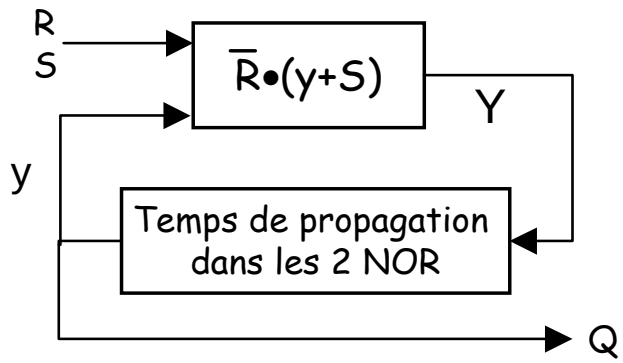


figure 6 Modélisation d'une bascule RS

Ecrivons les états présents et futurs pour les diverses combinaisons des entrées.

S = 1, *R* = 0 impose *Y* = 1 donc : $\begin{cases}\text{présent } y=0 \Rightarrow \text{futur } Y=1\\ \text{présent } y=1 \Rightarrow \text{futur } Y=1\end{cases}$ l'état *y* = 1 est stable.

S = 0, *R* = 1 impose *Y* = 0 donc : $\begin{cases}\text{présent } y=0 \Rightarrow \text{futur } Y=0\\ \text{présent } y=1 \Rightarrow \text{futur } Y=0\end{cases}$ l'état *y* = 0 est stable.

S = 0, *R* = 0 impose *Y* = *y* donc : $\begin{cases}\text{présent } y=0 \Rightarrow \text{futur } Y=0\\ \text{présent } y=1 \Rightarrow \text{futur } Y=1\end{cases}$ les états *y* = 1 et *y* = 0 sont stables.

Les résultats précédents peuvent être regroupés en une seule table donnant la valeur de l'excitation secondaire *Y* en fonction des variables *R*, *S* et *y*, et que l'on appelle table des excitations secondaires

	R	**0**	**0**	**1**	**1**
y	***S***	**0**	**1**	**1**	**0**
0		0	1	0	0
1		1	1	0	0

Y

Précisons que la disposition des entrées en combinaisons adjacentes n'est pas

obligatoire, mais simplement commode.

2.1.5. Deuxième exemple : l'astable.

Considérons maintenant le montage de la figure 7. La variable secondaire *y* est la seconde entrée de la porte NAND et l'excitation secondaire la sortie de cette même porte. Contrairement à l'exemple précédent, le retard *T* n'est plus fixé par le temps de propagation du signal à travers la porte logique, mais par le circuit *RC* intégré dans la boucle de retour. La sortie est *S=y*

L'excitation secondaire s'écrit $Y = \overline{E \bullet y} = \overline{E} + \overline{y}$. On a donc les deux situations suivantes :

$E = 0 \Rightarrow Y = 1$, l'état *y* = 1 est donc stable.

$E = 1 \Rightarrow Y = \overline{y}$ par conséquent $\begin{cases} \text{si l'état présent est y} = 1 \text{ alors l'état futur est Y} = 0 \\ \text{si l'état présent est y} = 0 \text{ alors l'état futur est Y} = 1 \end{cases}$

Ce qui donne la table des transitions suivante :

E / ***y***	**0**	**1**
0	1	1
1	1	0

Y

Pour *E* = 1 il n'y a pas d'état stable, le circuit oscille entre les deux états $y = 0$ et $y = 1$ avec une périodicité de l'ordre de *T* déterminée par le circuit *RC*. On peut utiliser ce type de circuit pour réaliser une horloge commandable (*E*=0 l'horloge est bloquée, *E* = 1 l'horloge fonctionne)

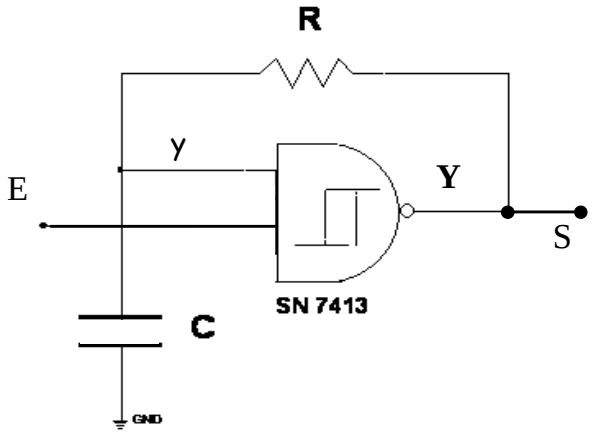


figure 7 Circuit astable

2.1.6. Conclusion pour les circuits asynchrones.

Il y a au moins trois points importants à retenir des deux exemples précédents.

1. La correspondance état présent ⇒ état futur est spécifique de chaque combinaison des entrées. En particulier un état stable pour certaines valeurs des entrées est instable pour d'autres

2. Il peut arriver que pour certaines entrées il n'existe aucun état stable. Le système asynchrone oscille alors librement entre plusieurs états à une fréquence caractéristique des constantes du système.

3. Les constantes de temps d'un système asynchrone sont souvent imposées par les temps de retard introduit par les circuits logiques mais pas uniquement. Le concepteur peut introduire volontairement des retards artificiels supplémentaires comme dans le cas de l'astable.

2.2. *Modélisation des systèmes synchrones.*

2.2.1. Systèmes synchrones avec horloge.

Ce sont des systèmes particuliers répondant aux caractéristiques suivantes :

- Ils possèdent une entrée particulière, baptisée horloge, alimentée par des impulsions.
- Les signaux sur les autres entrées sont des niveaux 0 ou 1.

A chaque impulsion d'horloge il se produit au plus un changement d'état (l'état futur devient de l'état présent).

En général les circuits synchrones avec horloge sont simplement appelés **circuits ou**

systèmes synchrones. Ils sont de plus en plus répandus. En effet, dans de nombreux systèmes complexes, les ordinateurs par exemple, les opérations doivent être parfaitement cadencées de manière à se produire dans un ordre bien déterminé. La synchronisation par une horloge commune est alors indispensable.

Les systèmes synchrones complexes sont construits à partir des circuits élémentaires que sont les bascules synchrones (*JK*, *D*, *RS*).

2.2.2. Evolution temporelle des systèmes synchrones.

Dans les systèmes synchrones la présence de l'horloge introduit une temporisation "infinie" entre l'état présent et l'état futur, ce qui simplifie beaucoup la modélisation. En particulier,

L'état présent est défini par les valeurs des sorties Q_i des bascules. Par contre, l'état futur est entièrement déterminé par les valeurs des entrées des bascules avant l'impulsion d'horloge. Dans la plus part des ouvrages on note Q_i^+ les valeurs futures des sorties des bascules : $Q_i^+ = f$ (entrées des bascules)

On peut se dispenser de la modélisation en variables secondaires et excitations secondaires. On écrit directement les valeurs des entrées dans l'état présent et on en déduit les futures valeurs des sorties.

Le système n'évoluant pas entre deux impulsions d'horloge, **il n'y a plus de notion d'état stable** ou, ce qui revient au même, tous les états sont stables!

A la différence des systèmes asynchrones, l'état présent et l'état futur sont physiquement accessibles sur le circuit.

Considérons, par exemple, le circuit représenté sur la figure 2.5. L'état présent est défini par la valeur *Q* de la sortie de la bascule. Puisqu'il s'agit d'une bascule *D* la valeur de l'entrée à l'instant *t* constitue la valeur future de la sortie $D = Q^+$. Comme par ailleurs, $D = \overline{Q}$ on a $Q^+ = \overline{Q}$ et on comprend immédiatement que la sortie Q va commuter à chaque impulsion d'horloge (fonctionnement en bascule *T*).

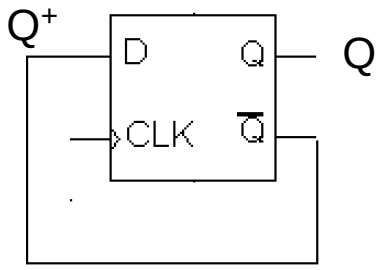


figure 8 Exemple de système séquentiel

2.3. *Analyse fonctionnelle des systèmes séquentiels.*

Le problème est le suivant : comment comprendre et analyser le fonctionnement d'un circuit dont on ne connaît que le schéma de câblage? La réponse est assez simple si on utilise le modèle précédent de représentation des circuits séquentiels et que l'on procède par étapes.

2.3.1. Les différentes étapes.

1ere étape : repérage des boucles de retour et identification des variables secondaires.

Sur le schéma on repère les différentes boucles de retour ainsi que les blocs séquentiels (par exemple des bascules). On affecte à chaque boucle et à chaque bloc séquentiel une variable secondaire (et donc une excitation secondaire).

2eme étape : écriture de la table des transitions (ou des excitations secondaires).

A partir du schéma du circuit, on déduit les expressions logiques combinatoires permettant de calculer les valeurs futures Q_i^+ en fonction des valeurs présentes Q_i s'il s'agit d'un circuit synchrone, ou bien s'il s'agit d'un circuit asynchrone, les excitations secondaires X_i en fonction des variables secondaires x_i. On remplit ensuite une table donnant les valeurs futures pour toutes les combinaisons possibles des valeurs présentes et des entrées : c'est la table des transitions (ou des excitations secondaires). Comme nous avons déjà commencé à le voir, c'est un tableau à double entrée ; les différentes combinaisons des variables secondaires sont écrites verticalement, et celles des entrées horizontalement. Une remarque néanmoins sur la présentation de ces tables. Dans le cas des systèmes synchrones avec horloge, on ne représente pas l'entrée d'horloge. Il est toujours implicite que le passage d'un état à un autre, se produit au moment de l'impulsion d'horloge.

Pour fixer les idées considérons un système à deux entrées A et B, et deux variables secondaires *x* et *y*, les tables des excitations secondaires se présentent donc de la façon suivante :

	A	**0**	**0**	**1**	**1**
x ***y***	**B**	**0**	**1**	**1**	**0**
0 **0**		0 0	0 1	0 1	0 1
0 **1**		1 1	0 1	0 0	1 1
1 **1**		1 1	1 0	1 0	1 0
1 **0**		0 0	1 0	1 1	0 0

X, Y

3eme étape : construction de la table des états et de la table des sorties.

Pour construire la table des états on donne un nom (il s'agit souvent d'un numéro ou d'une lettre) à chaque état présent ($x\ y$) et on réécrit la tables des transitions avec ce codage. Décidons, par exemple, du codage suivant :

$x\ y$ = 0 0 est l'état E1

$x\ y$ = 0 1 est l'état E2

$x\ y$ = 1 1 est l'état E3

$x\ y$ = 1 0 est l'état E4

La table des états est alors :

A	**0**	**0**	**1**	**1**
présent B	**0**	**1**	**1**	**0**
E1	E1	E2	E2	E2
E2	E3	E2	E1	E3
E3	E3	E4	E4	E4
E4	E1	E4	E3	E1

futur

Les tables des états sont un peu redondantes avec celles des transitions et apportent peu d'informations supplémentaires lors de l'analyse d'un circuit. Nous verrons plus loin dans ce cours quelles sont beaucoup plus utiles lors de la synthèse d'un système séquentiel.

La, ou les sorties, du système dépendent des valeurs présentes Q_i (ou des variables secondaires x, y....) et éventuellement des entrées A, B ... Pour dresser la table des sorties on établi à partir du schéma les expressions logiques combinatoires donnant la valeur des différentes sorties. On reporte ensuite dans une table les valeurs des sorties pour chaque état et, éventuellement, pour chaque combinaison des entrées.

4eme étape : représentation du diagramme d'états et identification de la finalité du circuit.

Un diagramme d'états visualise les changements d'états du système et permet d'en comprendre le fonctionnement. Nous en avons vu un exemple dans l'introduction lorsque nous avons décrit le fonctionnement d'un monte-charge.

Dans ce diagramme chaque état est représenté par un rond qui porte son nom (ou son numéro). A partir de chaque rond des flèches indiquent les transitions possibles, chacune d'entre elles étant associée à une combinaison des entrées. Là encore, pour un système synchrone pulsé, les combinaisons des entrées sont remplacées par la désignation de l'entrée active. Les flèches bouclées sur elles mêmes indiquent, s'il-y-a lieu, les oscillations. Enfin, on peut également y inscrire la valeur des sorties dans les différents états. Le diagramme correspondant aux tables précédentes est donné sur la figure 9.

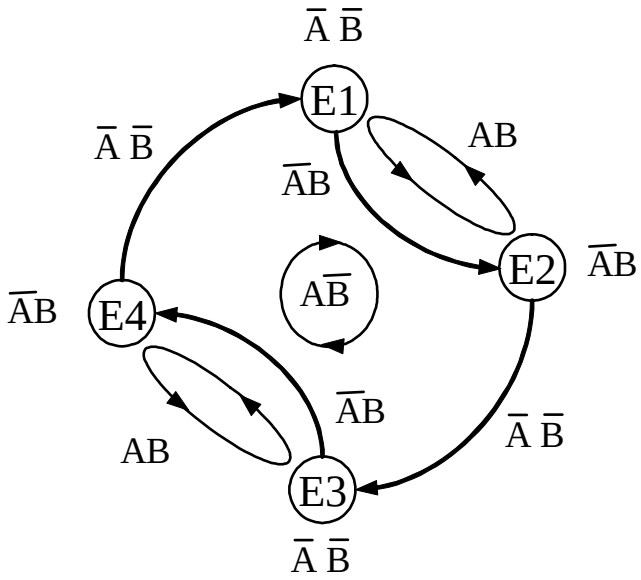


figure 9 Diagramme d'états correspondant aux tables précédentes. Pour les synchrones avec horloge les transitions se produisent au moment des impulsions d'horloge

Nous allons appliquer cette méthode d'analyse sur des exemples concrets de systèmes asynchrones ou synchrones avec horloge.

2.3.2. Analyse d'un système asynchrone.

Etudions le système asynchrone représenté sur la figure 10. Le circuit comporte deux boucles de retour et aucun bloc séquentiel non détaillé. Deux variables secondaires, x et y, sont placées sur le schéma ainsi que les deux excitations secondaires correspondantes X et Y.

1. A partir du schéma on établi les expressions combinatoires permettant de calculer les excitations secondaires X et Y.

$$X = \overline{B} \bullet y + B \bullet x$$

$$Y = \overline{A} \bullet (B \bullet \overline{x} + \overline{B} \bullet y) + A \bullet (\overline{B} \bullet \overline{x} + B \bullet \overline{y})$$

On remplit ensuite la table des excitations secondaires :

	A	**0**	**0**	**1**	**1**
x y	**B**	**0**	**1**	**1**	**0**
0 0		0 0	0 1	0 1	0 1

0 1	1 1	0 1	0 0	1 1
1 1	1 1	1 0	1 0	1 0
1 0	0 0	1 0	1 1	0 0

X, Y

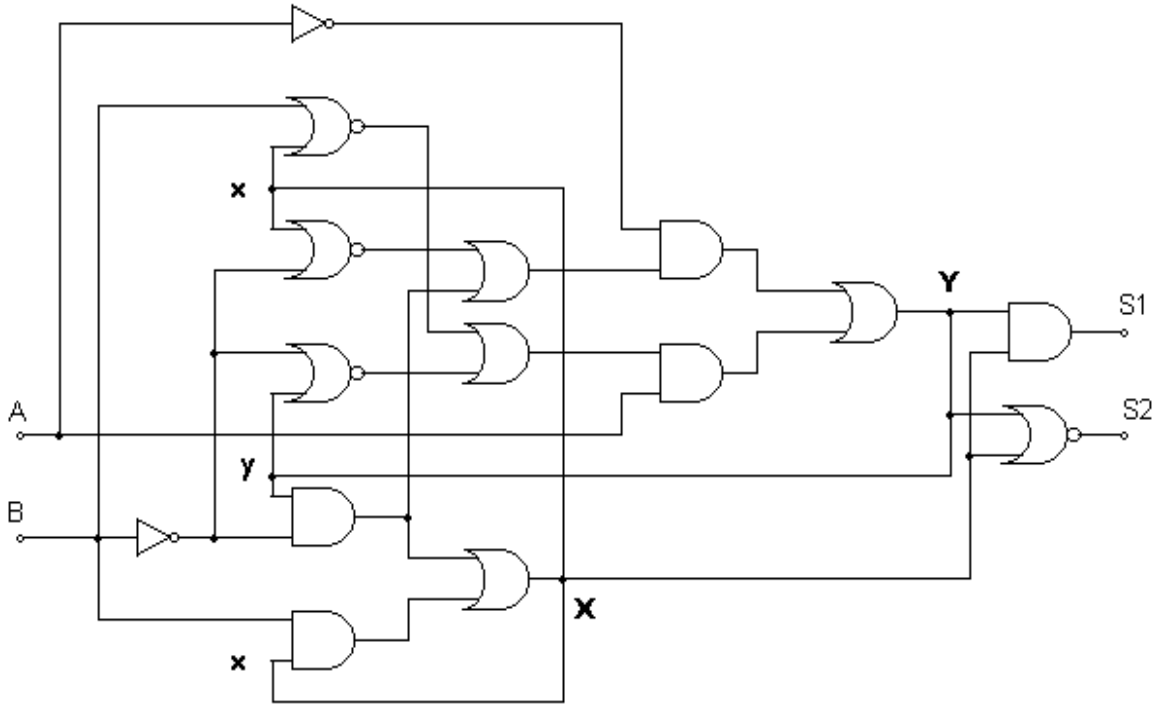


figure 10 : Exemple de circuit asynchrone. Les deux variables secondaires x et y sont indiquées sur le schéma ainsi que les excitations secondaires correspondantes

2. La table des excitations secondaires est identique à celle que nous avons écrite dans le paragraphe 2.3. En reprenant le même codage pour les états on obtient immédiatement la table des états.

le codage

$x\,y$ = 0 0 $\Rightarrow$ l'état E1

$x\,y$ = 0 1 $\Rightarrow$ l'état E2

$x\,y$ = 1 1 $\Rightarrow$ l'état E3

$x\,y$ = 1 0 $\Rightarrow$ l'état E4

donne

A	**0**	**0**	**1**	**1**
présent B	**0**	**1**	**1**	**0**
E1	E1	E2	E2	E2
E2	E3	E2	E1	E3
E3	E3	E4	E4	E4
E4	E1	E4	E3	E1

futur

A partir de la table des états, on remarque que les états E1 et E3 sont stables lorsque *AB* = 00. De la même façon les états E2 et E4 sont stables pour *AB* = 01. A l'inverse il n'existe aucun état stable pour les entrées *AB* = 11 et *AB* = 10.

Les sorties dépendent des entrées A et B et de l'état présent $x\ y$. A partir du schéma on établi les expressions combinatoires :

$$S_1 = \left(\overline{B} \bullet y + B \bullet x\right) \bullet \left(\overline{A} \bullet \left(B \bullet \overline{x} + \overline{B} \bullet y\right) + A \bullet \left(\overline{B} \bullet \overline{x} + B \bullet \overline{y}\right)\right)$$

$$S_2 = \overline{\left(\overline{B} \bullet y + B \bullet x\right) + \left(\overline{A} \bullet \left(B \bullet \overline{x} + \overline{B} \bullet y\right) + A \bullet \left(\overline{B} \bullet \overline{x} + B \bullet \overline{y}\right)\right)}$$

que l'on peut réécrire avec beaucoup de prudence sous la forme :

$$S_1 = X \bullet Y$$

$$S_2 = \overline{X + Y}$$

en se souvenant que les excitations secondaires n'ont aucune existence réelle. On remplie aisément la table des sorties.

	A	**0**	**0**	**1**	**1**
x y	**B**	**0**	**1**	**1**	**0**
0 0		0 1	0 0	0 0	0 0
0 1		1 0	0 0	0 1	1 0
1 1		1 0	0 0	0 0	0 0
1 0		0 1	0 0	1 0	0 1

S₁, S₂

3. Le diagramme d'états à déjà été présenté à la fin du paragraphe 2.3.1 Nous le re-dessinons sur la figure 11 en y ajoutant les valeurs des sorties S_1 S_2 et en séparant les différents modes de fonctionnement. A gauche est représenté le "régime stable" (*A* = 0) dans lequel les sorties oscillent entre 0 et 1 en opposition de phase en fonction des cycles 0→1→0 sur l'entrée *B*. Au centre on trouve l'oscillation "automatique" sur les 4 états lorsque *AB* = 10. Les sorties évoluent toujours en opposition de phase. Enfin, à droite est représentée l'oscillation du système sur 2 états lorsque *AB* = 11. Selon les conditions initiales, c'est à dire selon l'état de départ, une sortie reste à 0 pendant que l'autre oscille

entre 0 et 1.

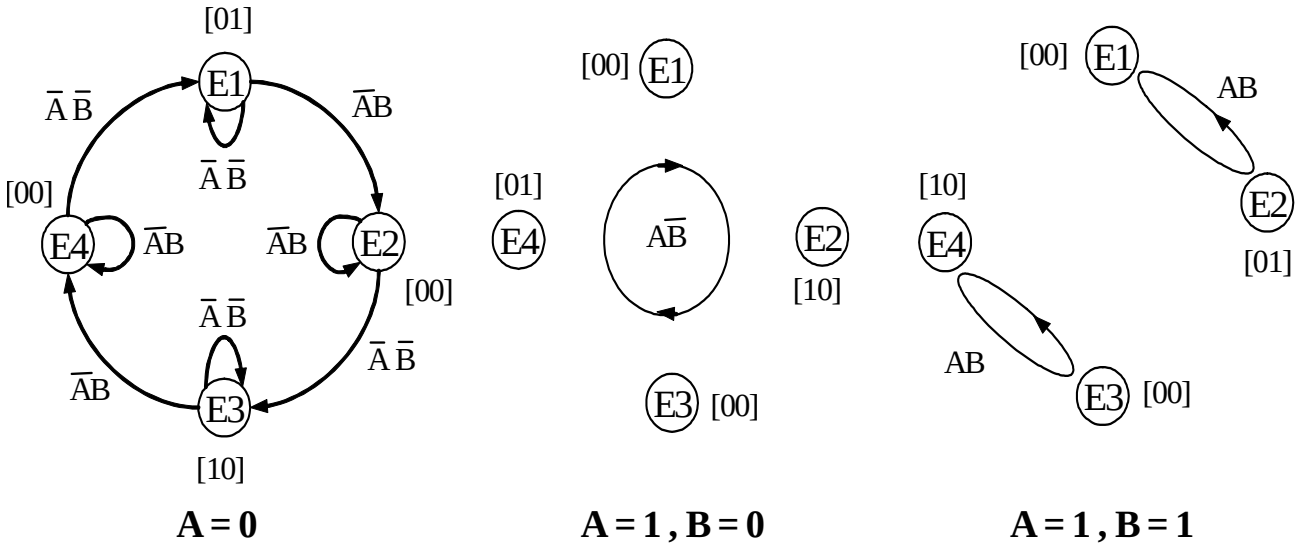


figure 11 Les 3 modes de fonctionnement du circuit asynchrone de la figure 10

2.3.3. Analyse d'un système synchrone (1er exemple).

Nous allons analyser le fonctionnement du circuit représenté sur la figure 12. La démarche est similaire à celle utilisée pour un système asynchrone. L'état présent est maintenant défini par les valeurs des sorties $Q_1\ Q_0$ des deux bascules, alors que l'état futur $Q_1^+\ Q_0^+$ est entièrement déterminé par les valeurs des entrées des bascules avant l'impulsion d'horloge $Q_i^+ = f(\text{entrées des bascules})$. Les entrées D_0 et D_1 sont gouvernées par les expressions :

$$D_1 = A \bullet (Q_0 \oplus Q_1) + \overline{A} \bullet Q_1$$

$$D_0 = Q_0 \oplus A$$

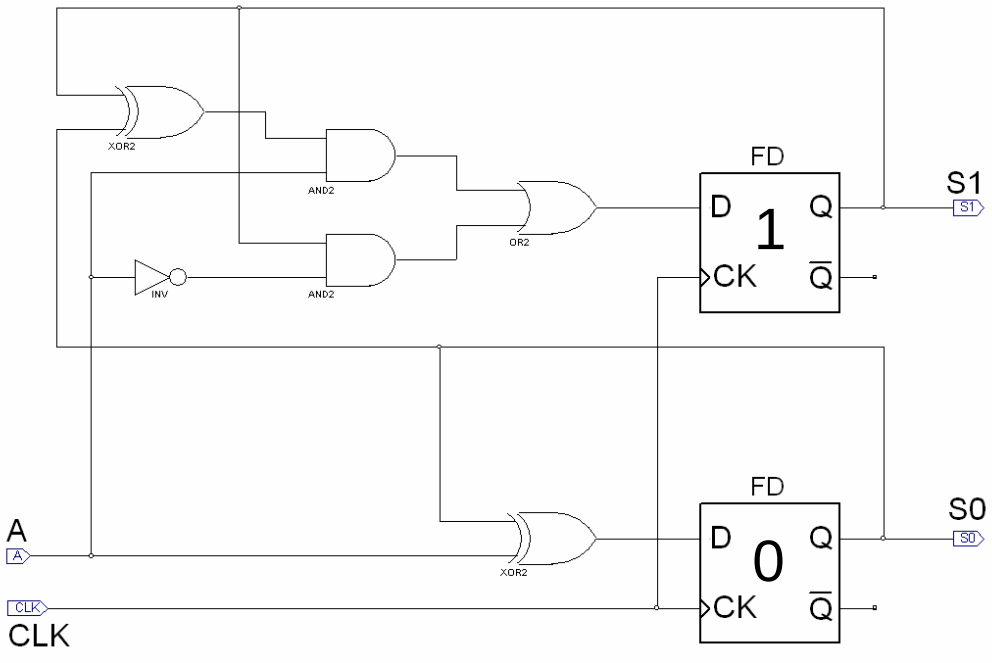


figure 12

Pour obtenir les valeurs futures Q_i^+ il faut se souvenir qu'une bascule D recopie sur sa sortie la valeur présente sur son entrée au moment de l'impulsion d'horloge. $Q_i^+(t) = D_i(t-1)$. On obtient ainsi la table des transitions :

A Q_1 Q_0	**1**	**0**
0 0	0 1	0 0
0 1	1 0	0 1
1 1	0 0	1 1
1 0	1 1	1 0

Q_1^+ Q_0^+

D_1 D_0

On code ensuite les états et on en déduit la table des états[5]:

le codage

$Q_1\ Q_0$ = 0 0 $\Rightarrow$ état E1

$Q_1\ Q_0$= 0 1 $\Rightarrow$ état E2

$Q_1\ Q_0$ = 1 0 $\Rightarrow$ état E3

$Q_1\ Q_0$ = 1 1 $\Rightarrow$ état E4

donne

A présent	**1**	**0**
E1	E2	E1
E2	E3	E2
E3	E4	E3
E4	E1	E4

état futur

Il faut bien comprendre que l'horloge est toujours présente, même si elle ne figure explicitement dans aucune de ces tables. Les changements d'états ne se produisent qu'au moment des impulsions d'horloge.

Dans cet exemple, les sorties du circuit ne dépendent que de l'état présent $Q_1 Q_0$ et jamais de l'entrée A. Elles s'expriment :

$$S_1 = Q_1$$

$$S_0 = Q_0$$

puisqu'elles correspondent directement aux sorties des bascules ! La table des sorties est ainsi

état	**S_1 S_0**
E1	0 0
E2	0 1
E3	1 0
E4	1 1

[5] Rien n'oblige à utiliser un codage identique à celui de l'exemple précédent.

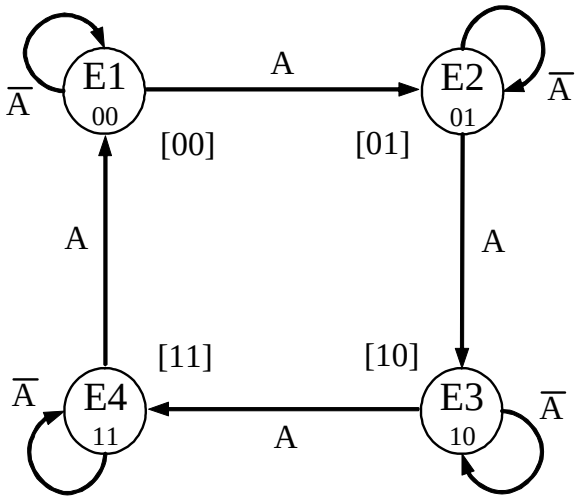


figure 13 Diagramme des transitions. Pour chaque état le codage Q_1Q_0 est indiqué. Les sorties $[S_1S_0]$ associées à chaque état sont également reportées

Le diagramme d'états illustrant le fonctionnement de ce système est donné sur la figure 13. Quel que soit l'état de départ et la valeur de l'entrée *A*, il y a toujours une transition active ; le système change ainsi d'état à chaque front montant du signal d'horloge. A partir de l'état E1 ($Q_1Q_0 = 00$) l'évolution du système, **au fur et à mesure des impulsions d'horloge**, est la suivante :

- Si *A* = 1 l'automate décrit un cycle en passant successivement dans les états E1, E2, E3 et E4.
- Si *A* = 0 l'automate reste dans l'état ou il se trouve.

La fonction réalisée par ce montage est obtenue en analysant l'évolution des sorties. Lorsque *A* = 1 les sorties S_1S_0 prennent successivement les valeurs 00, 01, 10, 11, 00 ... Il s'agit donc d'un compteur par 4 actif lorsque *A* = 1. L'entrée *A* = 0 bloque le compteur dans son état présent.

2.3.4. Analyse d'un système synchrone (2eme exemple).

Analysons maintenant le circuit représenté sur la figure 14. Le circuit ressemble beaucoup à celui que nous avons analysé précédemment. Il y a pourtant une différence importante ; elle concerne la sortie qui maintenant dépend explicitement de l'entrée A. Nous verrons que cela affecte significativement le fonctionnement du circuit et nous oblige a présenter un peu différemment le diagramme d'états.

Pour le moment procédons comme dans le paragraphe 2.3.3 pour analyser le circuit. Les équations de commande des deux bascules D s'écrivent :

$$D_1 = A \oplus Q_0 \oplus Q_1$$

$$D_0 = A \bullet Q_0 + Q_1$$

On obtient ainsi la table des transitions :

A Q_1 Q_0	**1**	**0**
0 0	1 0	0 0
0 1	0 1	1 0
1 1	1 1	0 1
1 0	0 1	1 1

Q_1^+ Q_0^+

D_1 D_0

On code ensuite les états et on en déduit la table des états:

le codage

$Q_1 Q_0$ = 0 0 $\Rightarrow$ état E1

$Q_1 Q_0$= 0 1 $\Rightarrow$ état E2

$Q_1 Q_0$ = 1 1 $\Rightarrow$ état E3

$Q_1 Q_0$ = 1 0 $\Rightarrow$ état E4

donne

A **présent**	**1**	**0**
E1	E4	E1
E2	E2	E4
E3	E3	E2
E4	E2	E3

état futur

Une fois encore l'horloge est explicitement présente dans ces deux tables et les changements d'états ne se produisent qu'au moment des impulsions d'horloge.

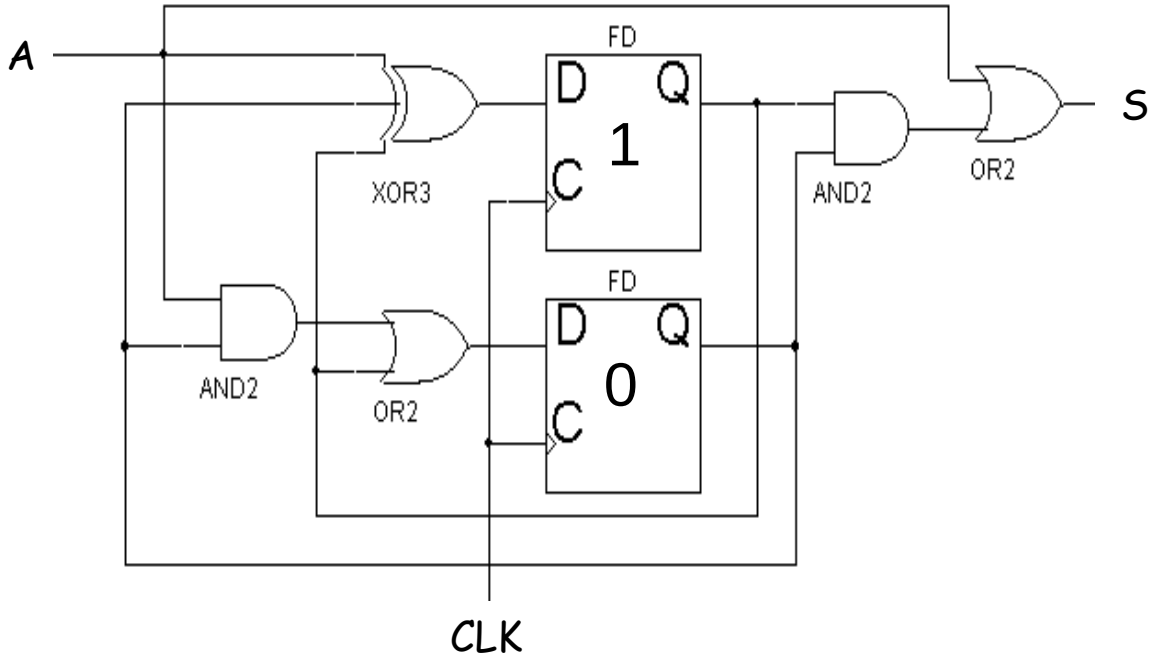


figure 14

La sortie S dépend à la fois de l'état présent $Q_1 Q_0$ **et de l'entrée *A***. Elle s'exprime:

$$S = A + Q_0 \bullet Q_1$$

On en déduit la table des sorties dans laquelle figure maintenant l'entrée A

A / Q_1 Q_0	**1**	**0**
0 0	1	0
0 1	1	0
1 1	1	1
1 0	1	0

S

A la différence du circuit synchrone étudié dans le paragraphe 2.3.3, on ne peut plus associer une valeur de la sortie à chaque état ! Pour un même état la valeur de la sortie dépend de l'entrée *A*. Ainsi, pour l'état E1 ($Q_1\ Q_0 = 0\,0$) on a *S = 1* si *A = 1* et *S = 0* si *A = 0*.

Le diagramme d'états associé à ce circuit est représenté sur la figure 15. Comme dans les exemples précédents, les flèches indiquent les transitions entre les états lorsque la condition sur l'entrée A est vérifiée au moment de l'impulsion d'horloge.

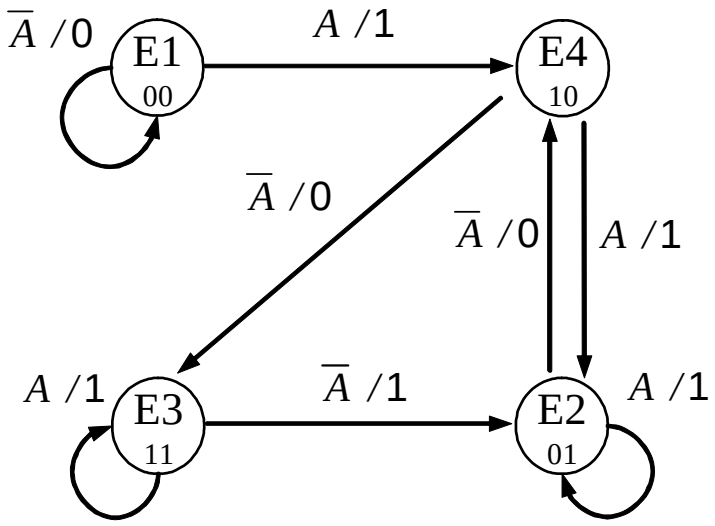


figure 15 Diagramme d'états du circuit représenté sur la figure 14

Pour tenir compte de l'action immédiate de l'entrée sur la sortie, on fait figurer la valeur de la sortie à coté de la condition de transition. Le diagramme de la figure 15 se lit donc de la façon suivante pour l'état E:

- Tant que A = 0 le système reste dans l'état E1 et la sortie S = 0.
- Dés que A = 1 la sortie devient S = 1 mais le système reste dans l'état E1 (pas d'impulsion d'horloge donc pas de transition)
- Si au moment de l'impulsion d'horloge on a toujours A = 1 alors le système évolue vers l'état E2 et la sortie reste S = 1 (elle repassera à 0 dés que A reviendra à 0).

Sur le chronogramme de la figure qui illustre le fonctionnement de ce circuit, on observe bien .

2.3.5. Comparaison des deux circuits synchrones analysés.

Les deux systèmes synchrones que nous venons d'analyser sont de nature différente. Celui du paragraphe 2.3.3 dans lequel la sortie ne dépend que de l'état présent (c'est-à-dire uniquement des sorties des bascules) est une machine de Moore. Sa sortie est toujours synchrone avec les impulsions de l'horloge.

A l'inverse, le circuit du paragraphe 2.3.4 est une machine de Mealy. Sa sortie dépend de l'état présent et de l'entrée. Elle n'est donc plus automatiquement synchrone avec les impulsions de l'horloge. Ce comportement est illustré sur le chronogramme de la figure 16 sur lequel on constate que la sortie S peut être synchrone avec l'entrée A et changer entre deux impulsions d'horloge. Nous reviendrons plus en détail sur ces deux types de machines dans le paragraphe 3.3

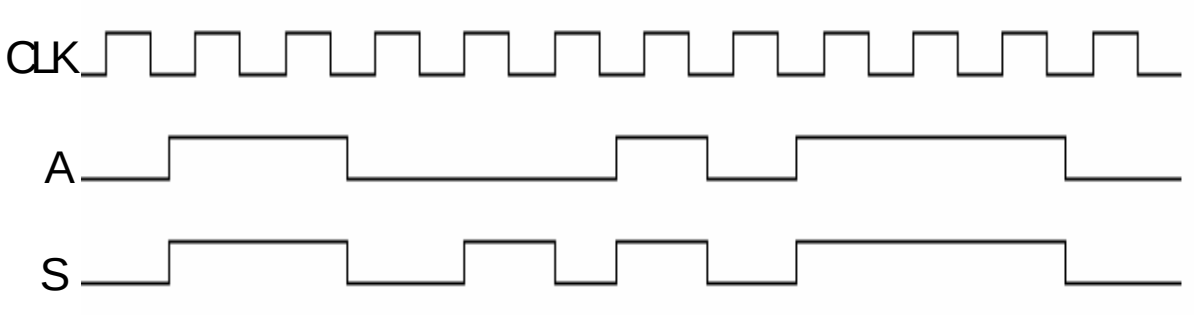


figure 16 Chronogramme illustrant le fonctionnement du circuit de la figure 14

2.4. *Intérêt des circuits synchrones avec horloge.*

En pratique on utilise rarement des circuits séquentiels asynchrones. La principale raison du manque de fiabilité des systèmes asynchrones est liée à la notion de course critique.

2.4.1. Notion de course critique : aléas.

La course critique est l'équivalent pour les systèmes séquentiels de l'aléa logique pour les systèmes combinatoires. Les courses critiques n'apparaissent que dans les systèmes asynchrones.

Considérons la table des excitations secondaires ci-dessous, et plaçons nous dans l'état $y_1\, y_2 = 01$ stable sous l'entrée *A* = 0 (2eme ligne). Si *A* passe à la valeur 1 l'état présent va devenir $y_1\, y_2 = 1\,0$. Les deux variables secondaires y_1 et y_2 doivent donc changer de valeur "simultanément".

En réalité, une des variables va obligatoirement commuter un très court laps de temps avant l'autre et ceci peut gravement perturber le fonctionnement d'un système asynchrone. Ainsi, si la transition $y_1\, y_2 = 01 \rightarrow y_1\, y_2 = 1\,0$ passe par l'état intermédiaire $y_1\, y_2 = 0\,0$ l'état final sera bien $y_1\, y_2 = 1\,0$ qui est stable pour *A*=1. A l'inverse, si l'état intermédiaire est $y_1\, y_2 = 1\,1$ (3eme ligne), le système se retrouve piégé dans cet état qui n'évolue pas quelque soit la valeur de l'entrée *A*. L'évolution finale de l'état $y_1\, y_2 = 0\,1$ est donc aléatoire et dépend de la rapidité des circuits. Dans cet exemple le phénomène est ici d'autant plus grave que l'état $y_1\, y_2 = 1\,1$ est un "piège" dans lequel le système restera bloqué quelle que soit la valeur de l'entrée. En pratique, dès que le circuit est un peu compliqué, il est impossible de connaître la variable secondaire qui va changer en premier ; tout dépend des retards introduits par les circuits et de la façon dont est câblé l'ensemble du montage.

La course A = 0 → A=1 est dite critique. Avant la généralisation des circuits synchrones, le remède consistait à éviter les changements non adjacents des variables secondaires par

un choix judicieux du codage des états. Aujourd'hui le remède consiste à ne jamais utiliser de circuits séquentiels asynchrones, et à toujours concevoir des systèmes à base de circuits synchrones avec horloge.

2.4.2. Le rôle de l'horloge.

Dans une réalisation synchrone le rôle de l'horloge est de supprimer toute possibilité d'aléas dans l'évolution de l'état. Idéalement entre deux transitions actives de l'horloge le système est figé, son état ne peut pas changer. Dans la réalité le temps qui sépare deux front consécutifs du signal d'horloge est mis à profit pour que les circuits combinatoires puissent effectuer leurs calculs sans que leurs temps de retard (toujours non nuls) ne risque de provoquer d'ambiguïté dans la valeur de l'état futur.

2.5. *Analyse temporelle des systèmes séquentiels.*

Dans les paragraphes précédents nous avons vu comment analyser un circuit pour en comprendre le fonctionnement (analyse fonctionnelle). Pour le moment, nous n'avons absolument pas tenu compte des contraintes, notamment temporelles, qu'impose la technologie sur le fonctionnement d'un circuit.

2.5.1. Contraintes temporelles sur les bascules synchrones.

Pour fonctionner de façon nominale les circuits séquentiels, en particulier les bascules, doivent respecter un certain nombre de contraintes concernant la stabilité de leurs entrées avant et après l'impulsion d'horloge ou encore la fréquence maximum de l'horloge.

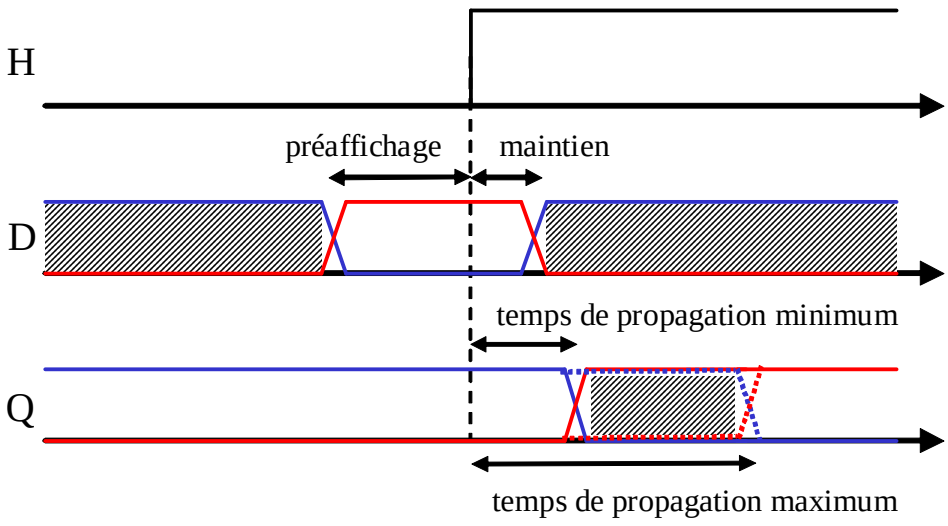


figure 17 Les temps caractéristiques de variation des entrées et sorties d'une bascule D synchrone

Les contraintes.

Considérons une bascule *D* synchrone. Son fonctionnement est bien connu : à chaque

impulsion d'horloge la sortie de la bascule recopie la valeur présente sur l'entrée. En réalité les choses sont un peu plus compliquées :

- L'entrée de la bascule doit être stable un certain temps avant l'impulsion d'horloge. Ce temps est appelé temps de *préaffichage* ou *setup time* en anglais.
- L'entrée de la bascule doit restée stable après l'impulsion. Cette durée correspond au *temps de maintien* ou *hold time* en anglais.
- La sortie change avec un certain retard par rapport à l'impulsion d'horloge. Ce retard correspond au *temps de propagation* du signal à travers la bascule.

Ces temps caractéristiques sont valables pour tous les types de bascules synchrones *JK*, *RS*, *T*). Ils sont représentés schématiquement sur la figure 17.

Les conséquences.

Evidement les contraintes précédentes ne sont pas sans conséquences sur le fonctionnement d'un circuit synchrone. Prenons l'exemple du circuit représenté sur la figure 18.

Quelles sont les contraintes sur les variations des entrées *A* et *EN* du système pour que les temps de préaffichage et de maintien de la bascule ne soient pas violés? Pour fixer les idées considérons les temps caractéristiques suivants :

- temps de préaffichage (setup time) $t_{set} = 2\,ns$.
- temps de maintien (hold time) $t_{hold} = 1\,ns$.
- temps de propagation à travers une porte logique combinatoire $t_{pc} = 1ns$.
- temps de propagation à travers la bascule (ou temps de basculement) $t_{pff} = 1,5\,ns$.

La situation est décrite sur le chronogramme de la figure 19. On constate que l'entrée *A* doit être stable entre 4 ns et 1 ns avant l'impulsion d'horloge.

A titre d'exercice on pourra montrer que si le temps de propagation a travers une porte logique (t_{pc}) varie entre 0,4 et 1,5 ns, alors l'entrée *A* doit être stable entre 5 ns avant et 0,2 ns après l'impulsion d'horloge.

La sortie change 1,5 ns après l'impulsion d'horloge.

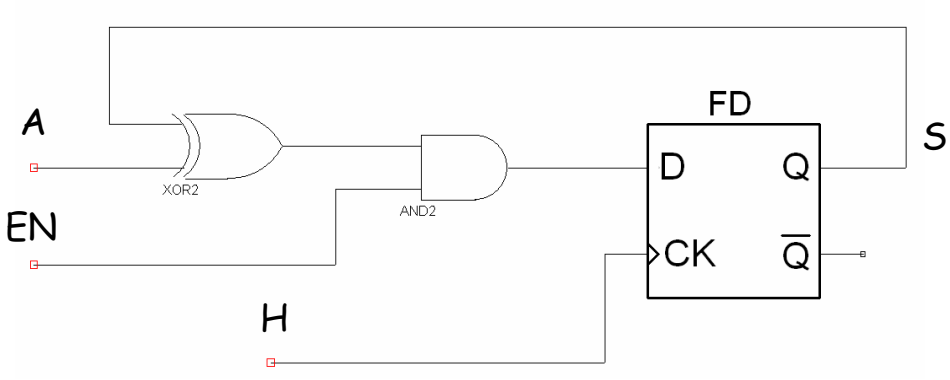


figure 18

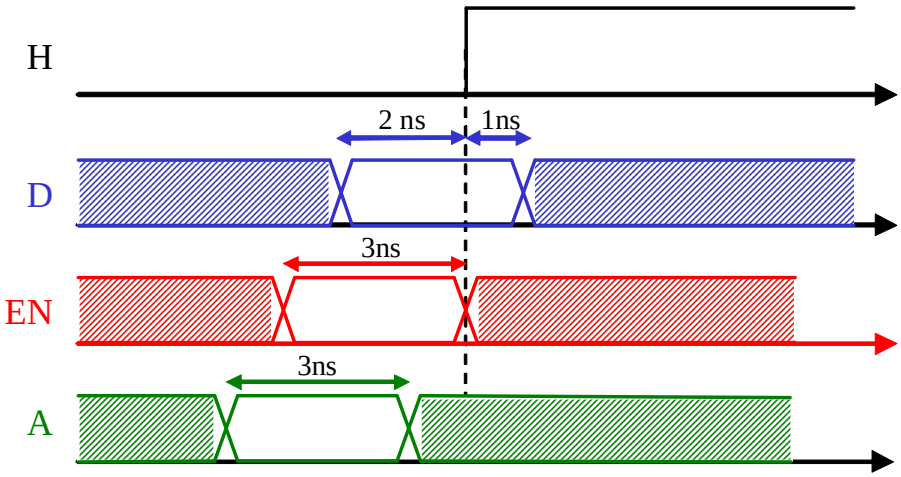


figure 19

2.5.2. Analyse temporelle d'un circuit.

Lors de l'analyse d'un circuit comportant des bascules connectées entre elles par l'intermédiaire de portes combinatoires plusieurs questions se posent. Nous allons les énoncer, et y répondre, en considérant le circuit schématisé sur la figure 20. Sur ce circuit deux bascules D sont reliées entre elles par un bloc combinatoire, la sortie de chacune des bascules également rebouclée sur son entrée.

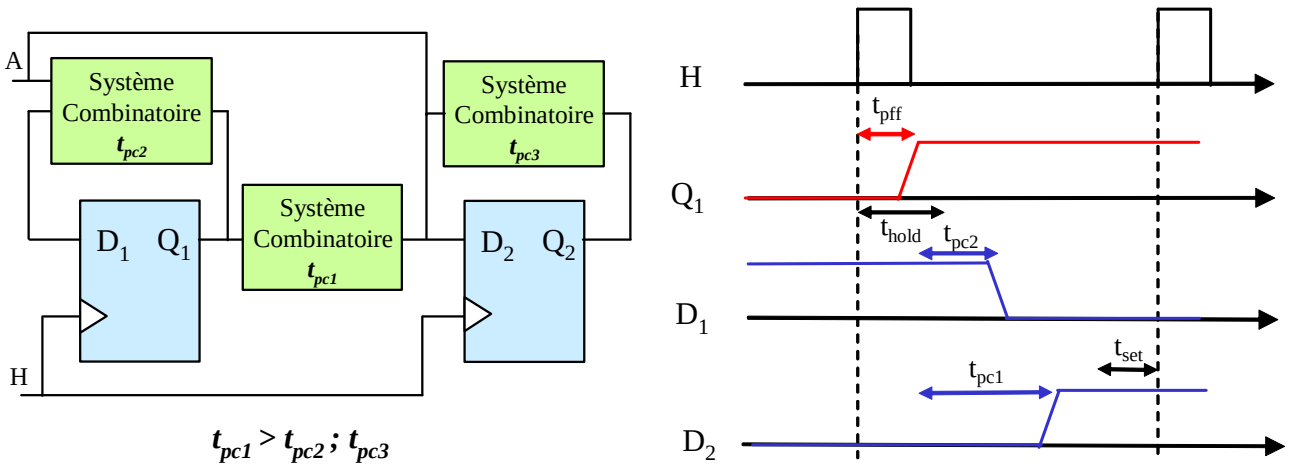


figure 20

1. Les retards introduits par les différents composants entraînent-ils des violations internes du temps de maintien des bascules?

Pour répondre il faut, pour chaque bascule, repérer la boucle la plus courte reliant les entrées et les sorties. Pour chacune des bascule il faut ensuite vérifier que le temps de basculement additionner au temps de propagation à travers les différentes portes logiques est supérieur au temps de maintien de la bascule :

$$t_{pff} + t_{pc2} > t_{hold}$$

$$t_{pff} + t_{pc3} > t_{hold}$$

Si, dans la documentation du circuit, les temps caractéristiques sont donnés sous la forme d'un interval entre une valeur minimum et une valeur maximum, la relation précédente doit bien sûr être vérifiée en utilisant les valeurs minimums des temps caractéristiques (figure 20).

2. Compte tenu du temps de préaffichage des bascules, quelle est la fréquence maximum autorisée pour l'horloge?

On doit s'assurer que les différentes contraintes temporelles sont respectées entre deux impulsions d'horloge. Pour cela on identifie la ou les connexions *sortie de bascule – entrée de bascule* pour lesquelles les retards accumulés sont maximums. La période minimum de l'horloge est alors, en supposant que $t_{pc1} > t_{pc2}$ sur la figure 20.

$$T_{\min} = t_{pff} + t_{pc1} + t_{set}$$

Cette fois la condition la plus stricte est obtenue avec les valeurs maximums des temps caractéristiques (figure 20).

3. Durant quelle période (par rapport à l'impulsion d'horloge.) les entrées doivent-elles être stables?

Il faut repérer sur le schéma les valeurs maximum et minimum des retards entre les entrées du circuit et les entrées des bascules, respectivement $d\ell_{\max}$ et $d\ell_{\min}$. Les deux situations "extrêmes" sont représentées sur le chronogramme de la figure 21. L'entrée *A* doit être stable entre les instants t_1 et t_2 définis par:

$$t_1 = t_H - (t_{set} + d\ell_{\max})$$

$$t_2 = t_H - (d\ell_{\min} - t_{hold})$$

4. A quel moment les sorties sont-elles susceptibles de changer?

Les sorties des bascules commuttent avec le retard t_{pff} par rapport à leurs entrées. A ce retard il faut ajouter les effets des éventuelles portes supplémentaires introduites entre les sorties des bascules et les sorties du circuit.

2.5.3. Exemple du compteur.

On reprend le circuit de la figure 12 et on définit les temps caractéristiques suivants :

- temps de préaffichage (setup time) $t_{set} = 2ns$.
- temps de maintien (hold time) $t_{hold} = 1ns$.
- temps de propagation à travers une porte logique combinatoire $0,5ns < t_{pc} < 1ns$.
- temps de propagation à travers la bascule $0,5ns < t_{pff} < 2ns$

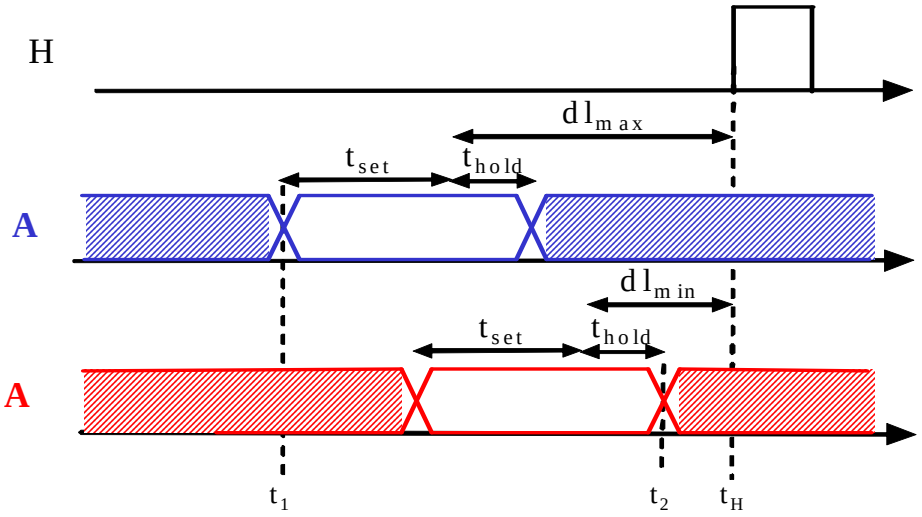


figure 21

1. Y-a-t-il un risque de violation interne des temps de maintien?

Non pour la bascule 1 : $t_{pff,\min} + 2 \times t_{pc,\min} = 0,5\,ns + 2 \times 0,5\,ns > t_{hold} = 1ns$

Par contre pour la bascule 0 on a : $t_{pff,\min} + t_{pc,\min} = 0,5\,ns + 0,5\,ns = t_{hold} = 1ns$

ce qui peut poser problème.

2. Quelle est la fréquence maximum autorisée pour l'horloge?

On calcule la période minimum à partir de l'expression :

$$T_{\min} = t_{set} + t_{pff,\max} + 3 \times t_{pc,\max} = 2\,ns + 2\,ns + 3 \times 1ns = 7\,ns$$

On en déduit la fréquence maximum $f_{\max} = 143\,MHz$.

3. A quels moments l'entrée *A* doit-elle être stable?

Sur le schéma on identifie les délais $d\ell_{\max} = 3t_{pc,\max}$ et $d\ell_{\min} = t_{pc,\min}$. L'entrée *A* doit être stable entre $t_{set} + d\ell_{\max} = 2ns + 3 \times 1ns = 5ns$ avant l'impulsion d'horloge et $d\ell_{\min} + t_{hold} = 0,5\,ns - 1ns = -0,5\,ns$ après l'impulsion d'horloge.

4. Les sorties du circuit sont directement reliées aux sorties des bascules, elles changent donc entre 0,5 *ns* et 2*ns* après l'impulsion d'horloge.

2.5.4. Remarques.

Pour ne pas compliquer les raisonnements précédents, nous avons supposé que les bascules étaient toutes excitées par une horloge parfaitement synchrone. Ceci n'est pas

toujours rigoureusement vrai, les signaux d'horloge. peuvent être légèrement décalés en différents points du circuit. Ce décalage, appelé *clock skew* en anglais, oblige à augmenter les temps de préaffichage et de maintien à prendre en compte lors de l'analyse temporelle d'un circuit (figure 22). Nous verrons dans le chapitre suivant les précautions à prendre pour limiter au maximum cet effet.

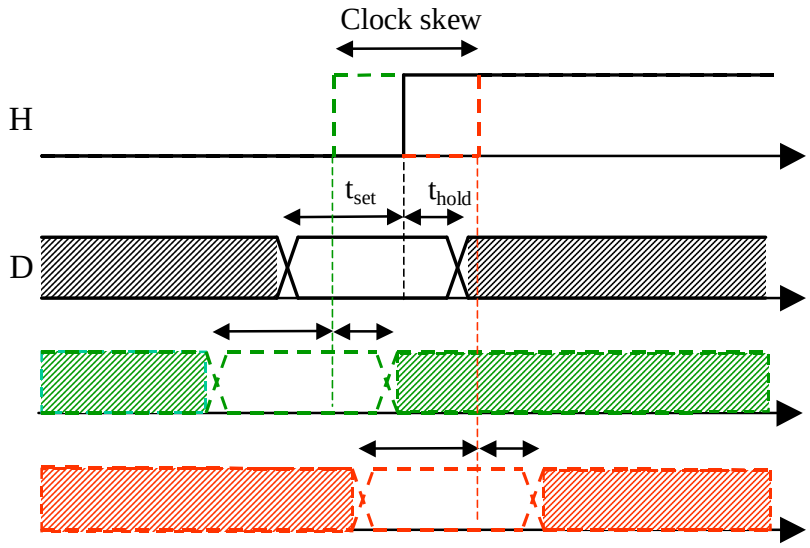


figure 22

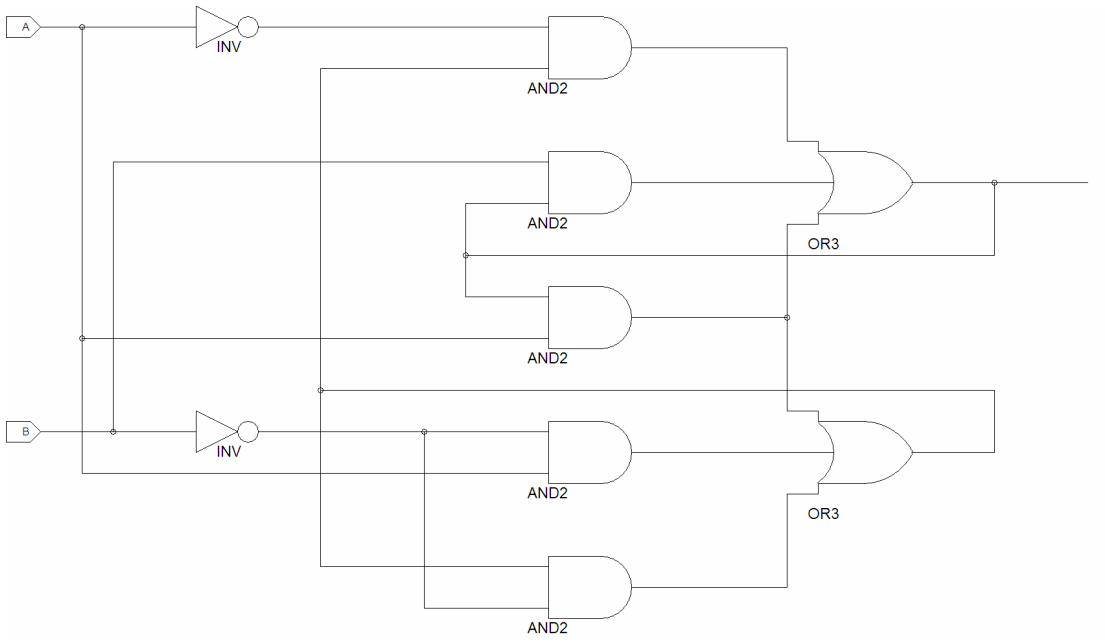
2.6. *Exercices corrigés.*

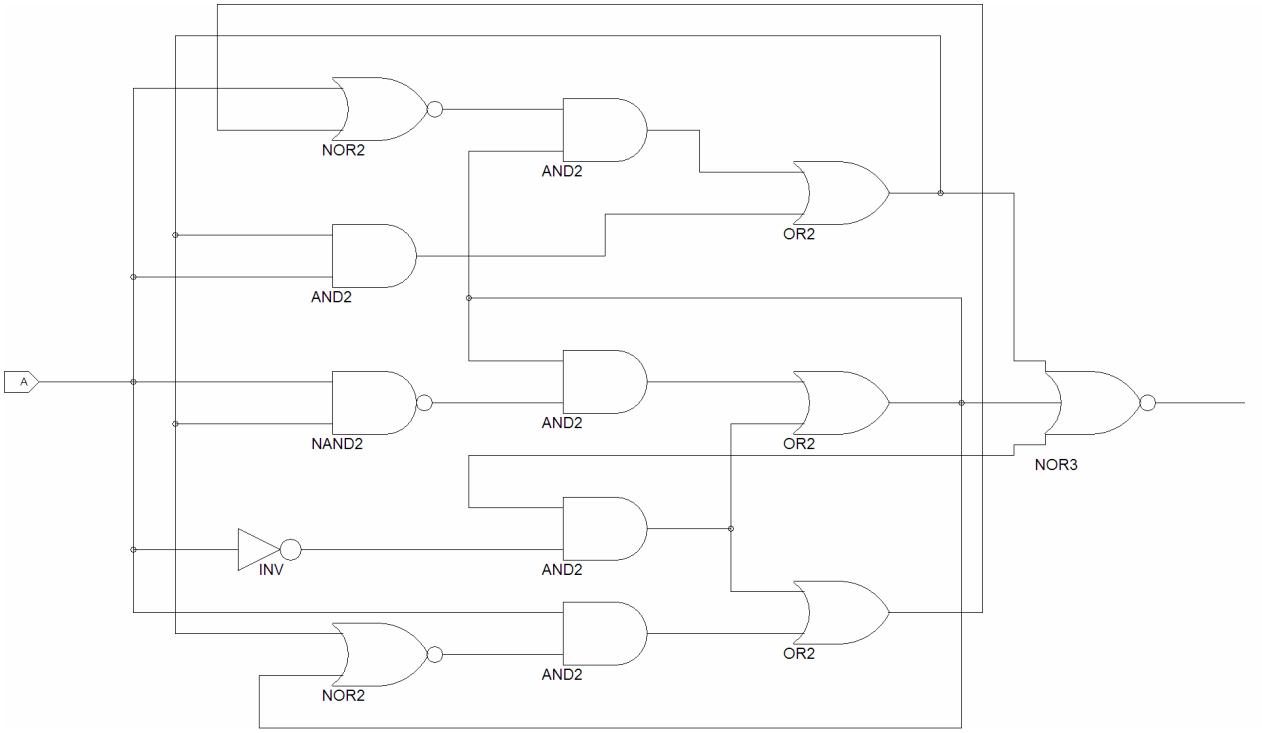
Exercice 1 : *Analyse d'un système asynchrone classique*.

Donner le fonctionnement de la bascule $\overline{R}\overline{S}$ (bascule formée de portes NAND) : Schéma, tables de vérité, états stables…

Exercice 2 : *Analyse d'un système Asynchrone*.

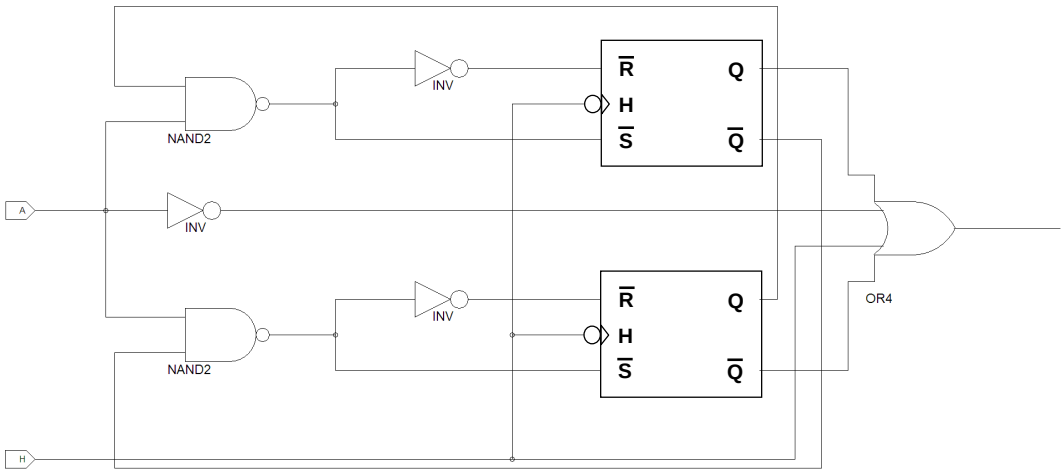
A et B sont des entrées des circuits suivants. Analyser ces deux circuits et interpréter le diagramme des transitions afin de proposer une fonction.





Exercice 3 : *Analyse.*

1. Quelle la différence fondamentale entre le montage suivant et les montages de l'exercice précédent?



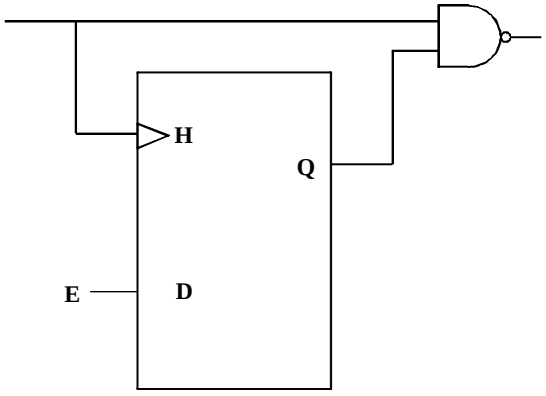
2. En supposant que ces bascules RS, sensibles aux fronts descendants, ont la table de vérité suivante:

R **S**	**0** **0**	**0** **1**	**1** **1**	**1** **0**
Q_{n+1}	X	1	Q_n	0

faire l'étude du circuit en donnant le diagramme de transition. Comment interpréter ce diagramme.

Exercice 4 : *La bascule D.*

Rappeler la table de vérité de la bascule D. Puis établir le chronogramme du schéma de la figure suivante contenant une bascule D. On suppose que le signal d'horloge envoyé sur l'entrée **H** de la bascule est signal carré.



Exercice 5 *: Analyse d'un système séquentiel*.

On considère le schéma de la figure 1 réalisé avec des bascules **RS asynchrones à base de portes NOR.**

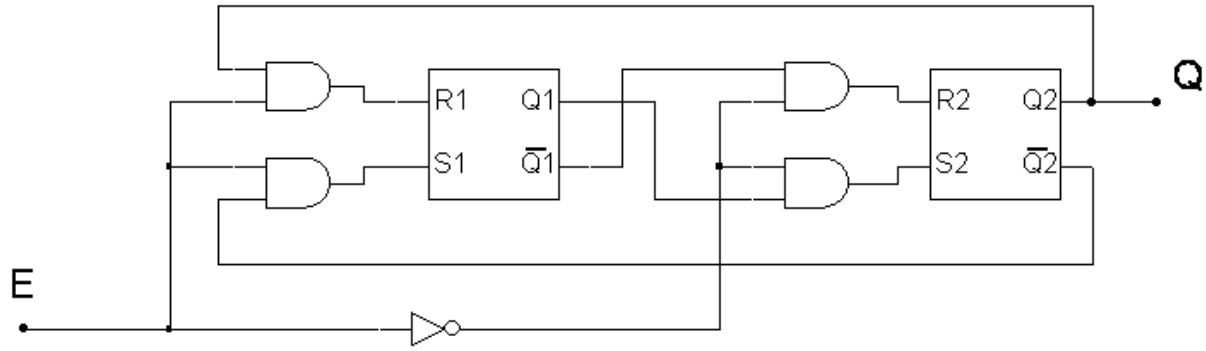
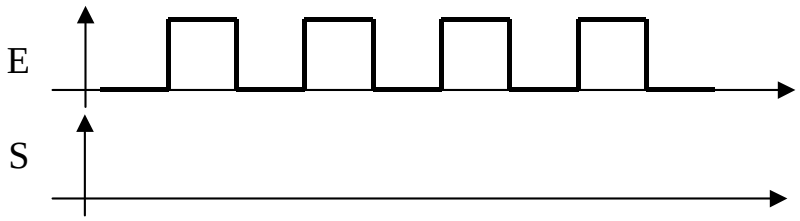


Figure 1

1. Analyser le fonctionnement de ce système. On donnera les tables des excitations secondaires et des transitions, et l'on commentera le comportement des bascules 1 et 2 lorsque E = 0 ou E = 1.

2. Compléter le chronogramme suivant.



3. Quelle est la fonction réalisée par ce montage? S'agit-il d'un système synchrone ou d'un système asynchrone? justifier clairement votre réponse.

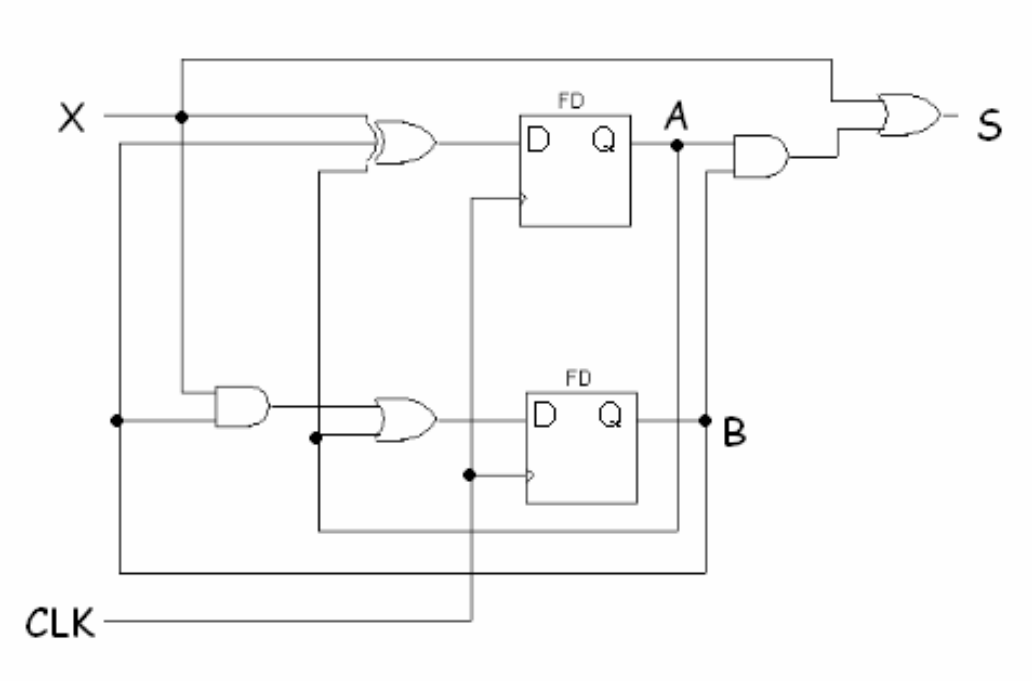
On rappelle que la table des transitions d'une bascule RS à base de portes NOR est :

R q \S	0 0	0 1	1 1	1 0
0	0	1	0	0
1	1	1	0	0

Q

Exercice 6 : *Analyse d'un système*.

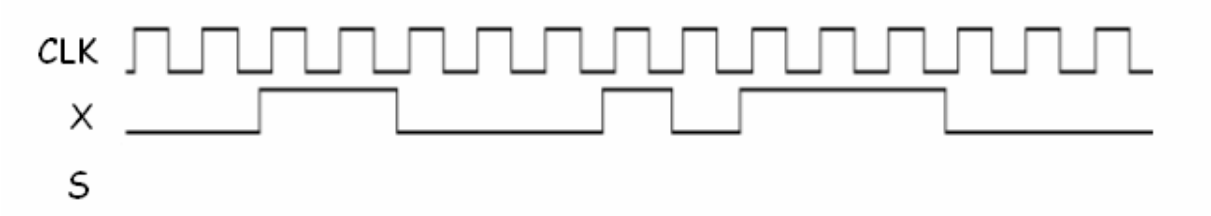
On considère le circuit suivant réalisé avec des bascules D synchrones sensibles à des fronts montants.



1. Etablir les équations de commandes des bascules et l'expression de la sortie S.
2. Ecrire la table de transitions reliant les valeurs futures A^+ B^+ en fonction des valeurs présentes A et B. Donner la table de la sortie S

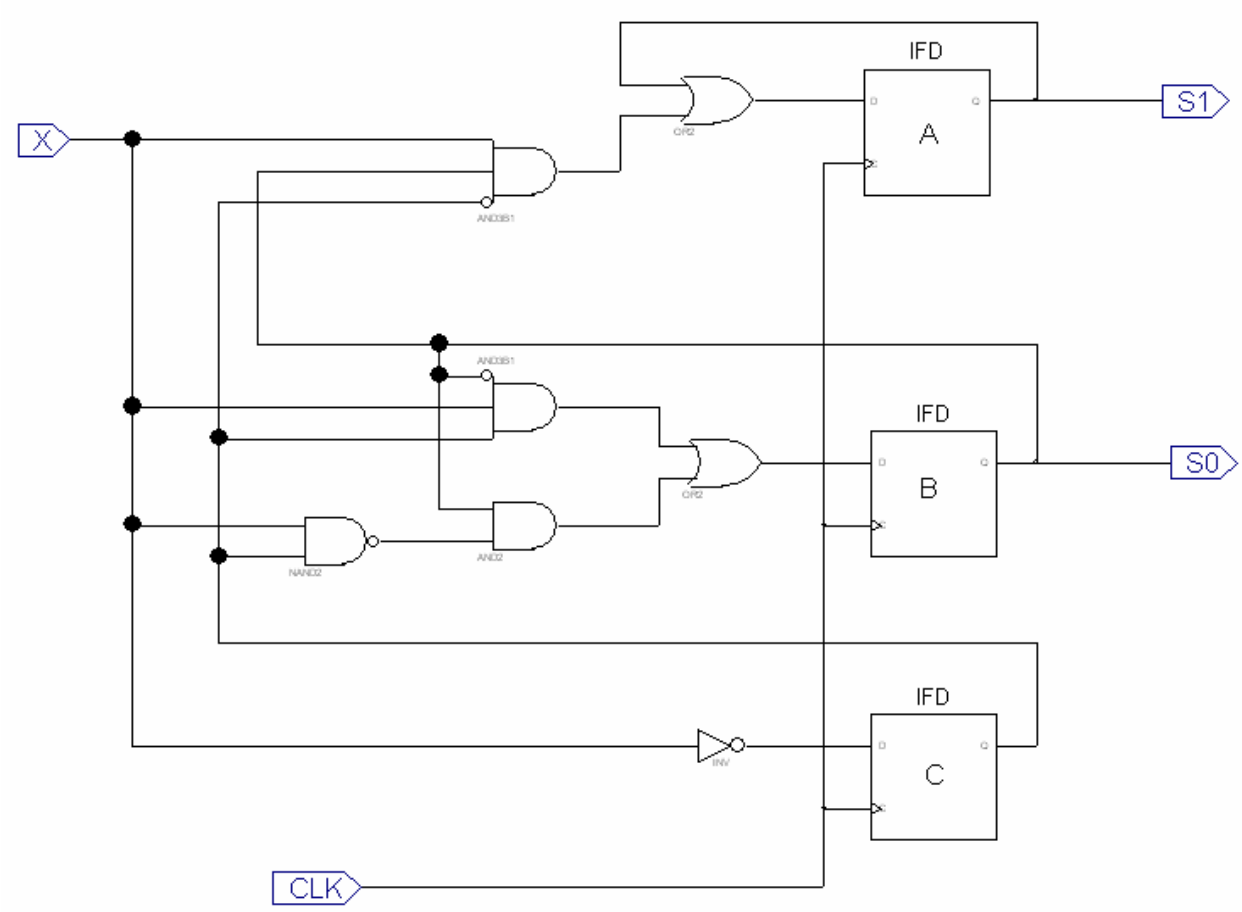
3. Tracer le diagramme d'états.

4. Compléter le chronogramme suivant. On supposera qu'initialement A = B = 0.



Exercice 7 : *Analyse d'un système séquentiel.*

On considère le circuit de la figure suivante réalisé avec des bascules D synchrones sensibles aux fronts montants d'une horloge.

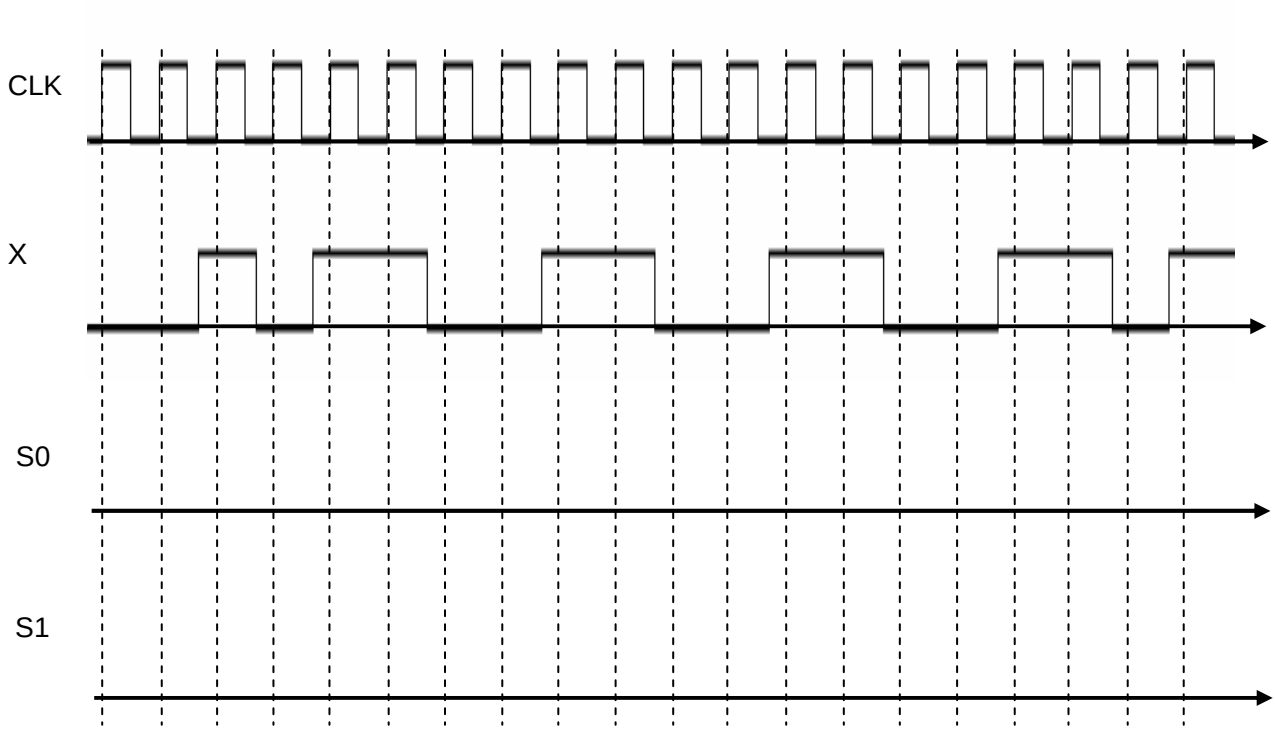


1. Etablir les équations de commandes des entrées D_i (i = A, B ou C) des bascules.

2. Remplir la table des transitions reliant les valeurs futures Q_i^+ aux valeurs présentes Q_i et à l'entrée X. Donner la table des sorties.

3. Représenter **<u>clairement</u>** l'évolution du circuit sous la forme d'un diagramme.

4. Compléter le chronogramme suivant. On supposera qu'initialement les sorties des 3 bascules sont au niveau bas.



5. Après que les sorties ont pris la valeurs S_1S_0 = 11, est-il possible de retrouver une situation ou S_1S_0 = 00 ? Justifier votre réponse.

6. On suppose que les caractéristiques temporelles des différents composants sont les suivantes :

- Le retard t_{pc} introduit par une porte logique est compris entre 0,5 et 1 ns.
- Le temps de propagation t_{pff} à travers les bascules est compris entre 1 et 2 ns.
- Les temps de préaffichage et de maintien des bascules sont respectivement t_{set} = 2 ns et t_{hold} = 1 ns

a. Existe-t-il un risque de violation interne des temps de maintien ?

b. Quelle est la fréquence maximum de fonctionnement du circuit ?

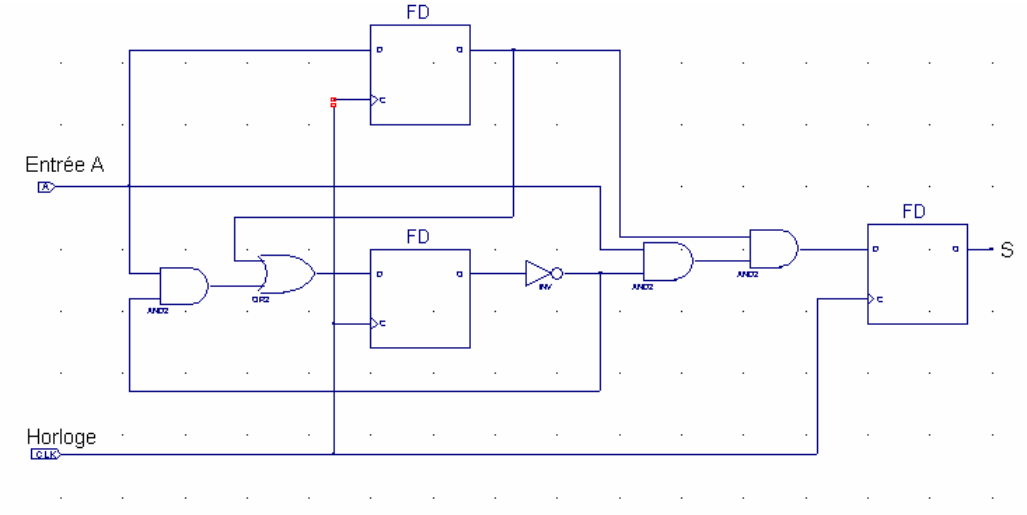
c. Pendant quel intervalle de temps l'entrée X doit-elle être constante ?

Exercice 8 : *Analyse temporelle d'un montage séquentiel.*

1. Quels sont les temps caractéristiques utiles à l'étude temporelle (on ne cherche plus à faire l'analyse fonctionnelle) d'un circuit contenant des bascules ? On reprendra les notations du cours : t_{set}, t_{hold}, …, t_{pff} (on estime le temps de propagation dans une porte entre 0.25 et 0.5 ns, dans une bascule à 1ns, maintien 1ns et préaffichage 2ns). Donner une brève description de chacune de ces caractéristiques.

2. Le circuit présente au moins une rétroaction, quel est la première condition liée au temps de maintien d'une bascule qui doit être satisfaite ? On examine ensuite les éventuelles violations qui pourraient se produire entre deux fronts actifs d'horloge. Ces considérations permettront de déterminer la fréquence d'horloge maximum envisageable pour ce montage et ces temps caractéristiques.

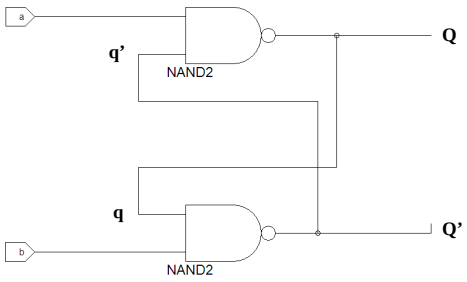
3. Le dernier point à vérifier est lié à intervalle temporel durant lequel les entrées doivent être stables. Déterminez cet intervalle.



Autocorrection

Exercice 1 : *Analyse d'un système asynchrone classique.*

Les circuits des bascules RS (il y a plusieurs versions) sont particulièrement symétriques, on peut donc s'attendre à une fonctionnement symétrique. Le montage d'une bascule RS basée sur l'utilisation de portes NAND :



Ce montage présente deux rebouclages que l'on notera q et q' mais il n'y a aucune horloge ni synchronisation, Nous sommes en présence d'un système séquentiel asynchrone. Vu la relative simplicité du circuit, on pourrait être tenté de réaliser une étude intuitive.

Ici nous réaliserons l'étude exhaustive selon la méthode vue en cours, le lecteur pourra retrouver les résultats par la méthode « économique ».

Les entrées a et b

a **q q' \ b**	**0** **0**	**0** **1**	**1** **1**	**1** **0**
0 0	1 1	1 1	1 1	1 1
0 1	1 1	1 1	0 1	0 1
1 1	1 1	1 0	0 0	0 1
1 0	1 1	1 0	1 0	1 1

Q Q'

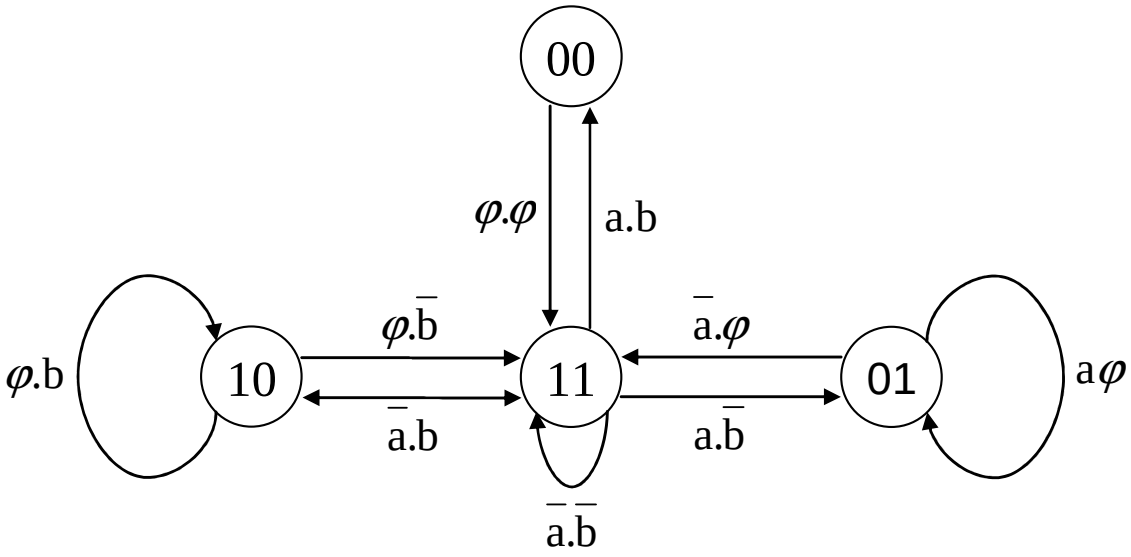
Dans ce tableau nous calculons l'état futur QQ' pour tous les états initiaux qq' possibles et pour toutes les combinaisons des entrées a et b possibles. Les premières conclusions

concernent la stabilités des états : l'état initial (00) ne sera jamais stable quelque soit la valeur des entrées a et b, en revanche tous les autres états (01), (10) et (11) peuvent être stables.

D'autre part étant donné que notre circuit est asynchrone il faut déterminer la présence d'éventuelles transitions interdites qui correspondent au changement simultané d'au moins deux bits de l'état. Prenons la transition de qq'=11 vers QQ' =00 pour les entrées ab=11. Les deux bits q=1 et q'=1 tombent à zéro, bien sur ils ne tomberont pas exactement au même moment donc en réalité nous auront une transition de 11 vers 10 ou 10 sans pouvoir déterminer l'une ou l'autre. Comme nous pouvons prévoir de manière exacte l'évolution du système il faudra éviter d'utiliser cette transition.

qq'=00 vers QQ' =11 pour les entrées ab=φφ sont aussi des transitions interdites.

L'état (00) n'est pas stable et toutes les transitions partant de cet état sont interdites, il semble préférable, si on peut, d'éviter d'utiliser cet état.



Les cercles représentent les états notés (qq').

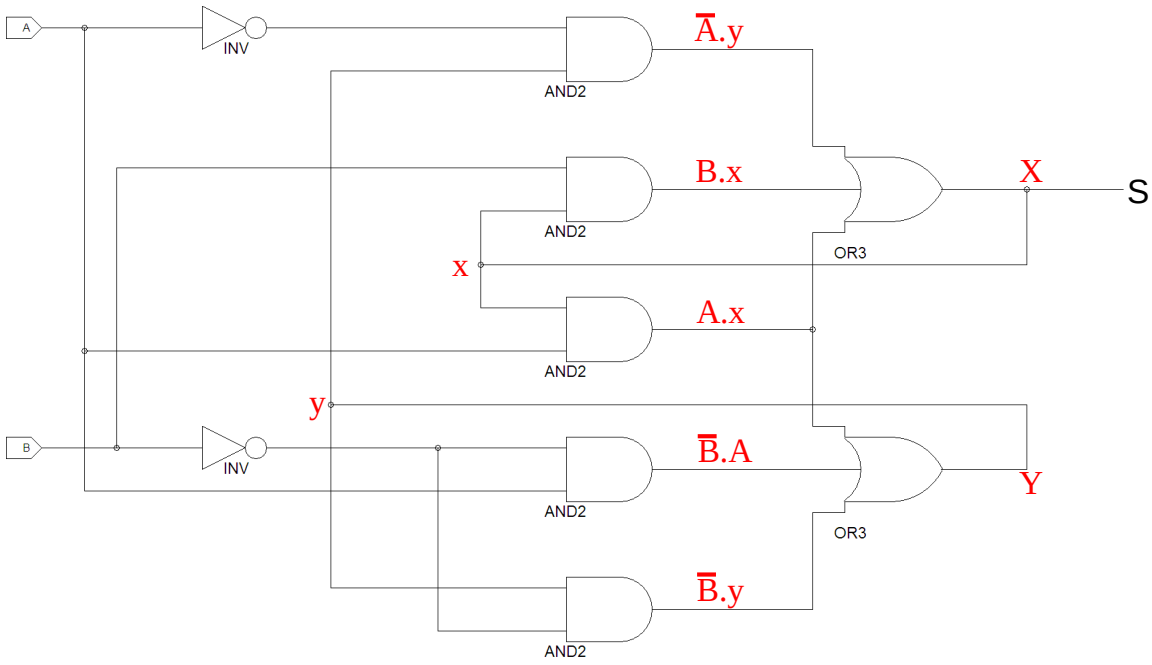
Le diagramme des transitions est parfaitement symétrique et on voit qu'il est possible d'avoir un fonctionnement cyclique entre les états (10) et (01) sans utiliser (00). En appliquant b=0, le système décrit la partie gauche avec q=1. En appliquant a=0, le système décrit la partie droite avec q=0. Il faut noter que les sorties sont complémentées. On choisira donc les notations en rapport avec les fonctions des bascules : l'entrées a deviendra Set et b, Reset ; l'état actif est 0 (pour faire un reset on applique 0 à b), on parle alors de bascule $\overline{R}.\overline{S}$.

$\overline{S}$ $\overline{R}$	**Sortie Q**	**fonction**
1 1	q	Mémoire ou Interdit
0 1	1	set
1 0	0	reset
0 0	1	RAZ du système

Pour résumé, nous avons les fonctions set et reset lorsque les entrées sont complémentées, avec les deux entrées à ‘1’ la dernière valeur est mémorisée et l’état interdit (00) sera à éviter.

Exercice 2 : *Analyse d'un système Asynchrone*.

- Premier montage



Nous sommes en présence d'un circuit asynchrone, pas d'horloge ni étage de bascules, par contre nous identifions deux entrées A et B et deux rebouclages ou excitations secondaires X et Y. Nous noterons x et y les variables secondaires. Dans ce genre de montage il est très utile de noter la valeur de sortie de chaque porte. Les excitations secondaires sont : $X = \overline{A}.y + B.x + A.x$ et $X = \overline{B}.y + \overline{B}.A + A.x$.

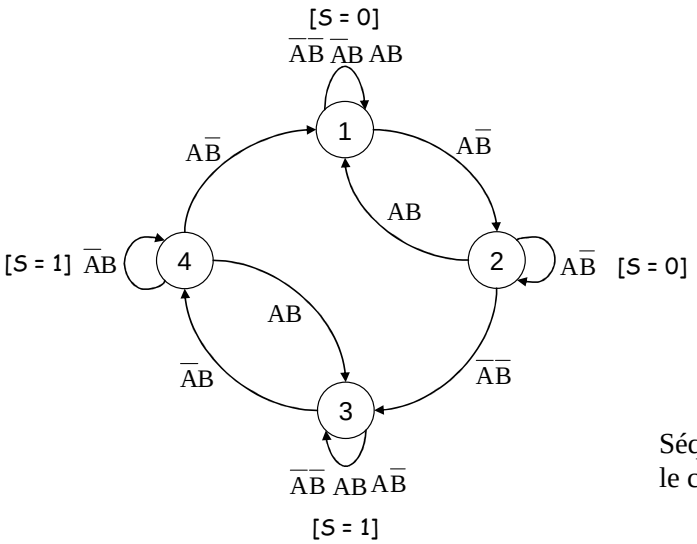
A / **xy \ B**	**0** **0**	**0** **1**	**1** **1**	**1** **0**
0 0	00	00	00	01
0 1	11	10	00	01
1 1	11	10	11	11
1 0	00	10	11	11

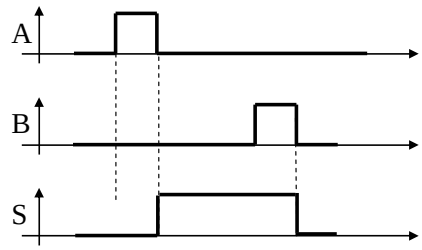
Table des excitations secondaires : X Y

état \ présent	A B	0 0	0 1	1 1	1 0
1		(1)	(1)	(1)	2
2		3	x	1	(2)
3		(3)	4	(3)	(3)
4		1	(4)	3	x

Tables des transitions **Etats futurs**

Les transitions interdites sont notées « x » afin d'éviter de les inclure dans le diagramme, on note aussi les stabilités des états par des parenthèses. Pour un circuit synchrone, on peut se permettre d'avoir un état non stable (on sait par définition que le système restera une période d'horloge dans cet état non stable). Dans le cas asynchrones, on ne pas prévoir le temps que restera le système dans cet état instable (ce temps est très court et dépend de la propagation et de la commutation des portes). On préfère alors n'avoir que des états stables. Ce qui est le cas ici.





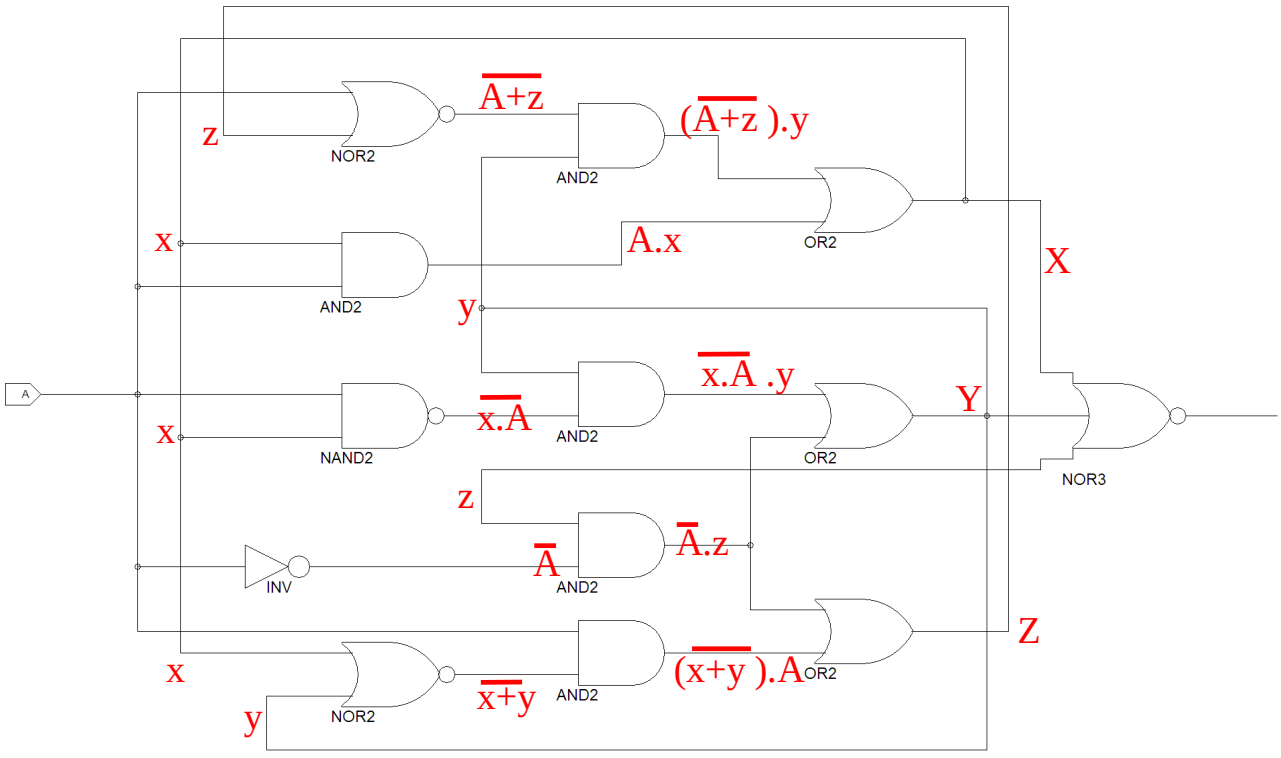
Séquences des entrées pour parcourir le cycle

Le diagramme des transitions permet présenter de manière synthétique la valeur des sorties (ici, triviale car S=X) et les transitions, même pour des diagramme avec un faible nombre d'états il est préférable de compter le nombre total de transitions : 16 – 2 transitions interdites = 14. Le système est cyclique et sa fonction repose certainement sur ce cycle, le chronogramme de droite est construit de manière à parcourir le cycle. La séquence

temporelle des entrées A et B nécessaires à cela ainsi que la valeur de la sortie sont facilement interprétables. Un premier bouton poussoir A active la sortie sur son front descendant tandis qu'un second bouton poussoir la désactive.

Remarque : il n'est jamais évident de proposer une fonction pour un circuit à partir de son diagramme d'état. Par contre, on peut conseiller de commencer l'analyser par les caractéristiques suivantes : la stabilité des états et la symétrie du diagramme d'état sont rarement le fruit du hasard et la fonction finale en dépend très certainement. La présence de cycles, de sous-cycles, de cycles imbriqués est capitale. A contrario l'absence de cycle est inquiétante, une machine qui ne marche qu'une fois est a priori mal conçue. Si le résultat de votre analyse amène à un diagramme sans structure ni symétrie évidente, il y a peut-être une erreur.

- Second montage



La encore, nous avons 1 entrée et 3 rebouclages notés respectivement X,Y et Z mais aucun bloc séquentiel, le système est donc asynchrone. Les variables secondaires sont

notées respectivement x,y et z. l'étude du schéma donne :

$$X = A.x + \overline{(A+z)}.y$$
$$Y = \overline{A}.z + \overline{A.x}.y \qquad \text{et} \quad S = \overline{X+Y+Z} = \overline{X} + \overline{Y} + \overline{Z}$$
$$Z = \overline{A}.z + \overline{(x+y)}.A$$

Pour remplir la table des excitations secondaires il est préférable de développer les expressions en mintermes soit

$$X = A.x + \overline{(A+z)}.y = A.x + \overline{A.z}.y$$
$$Y = \overline{A}.z + \overline{A.x}.y = \overline{A}.z + \overline{A}.y + \overline{x}.y$$
$$Z = \overline{A}.z + \overline{(x+y)}.A = \overline{A}.z + \overline{x}.\overline{y}.A$$

Pour la colonne des excitations secondaires nous avons choisi le code gray, bien sur cela ne change pas le résultat, la lecture sera juste plus ou moins aisée. Par contre étant donné le problème des transitions interdites il vaut prendre un code adjacent. On voit se dessiner une progression d'une ligne à l'autre (une transition état n vers état n+1, pour E=0 et E=1 alternativement).

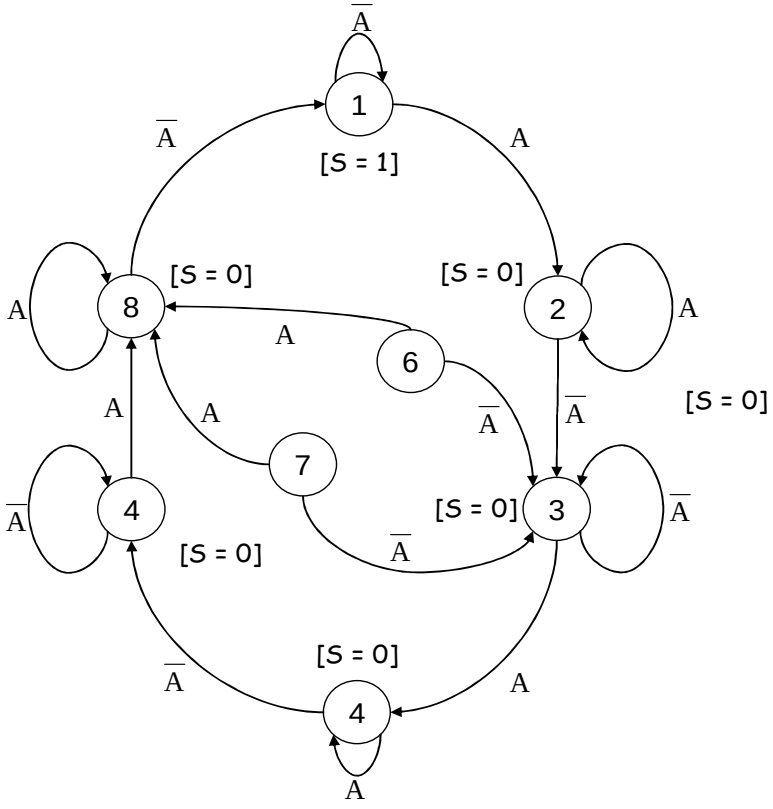
A / **xyz **	**0**	**1**
0 0 0	(000)	001
0 0 1	011	(001)
0 1 1	(011)	010
0 1 0	110	(010)
1 1 0	(110)	100
1 1 1	011	100
1 0 1	011	100
1 0 0	000	(100)

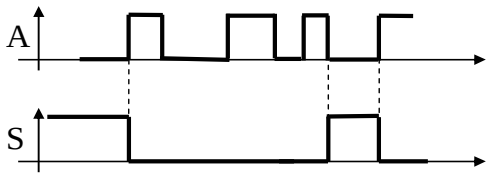
Valeurs futures de X Y Z

Etat \ **A** présent	**0**	**1**
1	(1)	2
2	3	(2)
3	(3)	4
4	5	(4)
5	(5)	8
6	3	8
7	3	8
8	0	(8)

ETATS FUTURS

Remarque : la numérotation des états ne sert qu'à alléger le diagramme et peut-être choisie au hasard.





Séquences des entrées pour parcourir le cycle

Le diagramme présente clairement une structure cyclique avec un seul état pour lequel la sortie est active. La fonction de ce montage est de reconnaître une séquence temporelle sur l'unique entrée A, la sortie est alors activée jusqu'au prochain front montant sur l'entrée. Cette séquence présentée sur le chronogramme de gauche est composée de 3 créneaux dont les largeurs quelconques. L'alternance (stabilité pour A=x, transition pour A=$\overline{x}$) pour l'état n et (stabilité pour A=$\overline{x}$, transition pour A=x) pour l'état n+1 est typique des circuits asynchrones.

Remarque : l'analyse montre ici deux états 6 et 7, instables et a priori inaccessibles, ils sont qualifiés d'états parasites. Ils sont présents à chaque fois que le nombre de possibilités de codage des états est supérieur au nombre d'états nécessaire à la fonction. Les trois excitations secondaires permettent 2^3 états possibles alors que seulement 6 états sont nécessaires pour reconnaître la séquence des 3 impulsions sur l'entrée.

Exercice 3 : *Analyse.*

Il faut noter en plus des rebouclages que ce montage présente un bloc synchrone constitué de deux bascules $\overline{R}\overline{S}$ synchrones (à la différence de l'exercice 1). Ces bascules ne commutent sous l'effet des entrées $\overline{R}$ et $\overline{S}$ (si il y a lieu) que sur un front descendant de l'horloge H. Dans la mesure où la même horloge est appliquée sur les deux bascules, on parle d'un unique bloc synchrone.

Remarque : La méthode d'analyse est la même que pour un circuit asynchrone à ceci près qu'il n'est plus important de se soucier des transitions interdites qui sont donc, à présent, autorisées.

Les entrées $\overline{R}\overline{S}$ des bascules sont toujours complémentées pour éviter d'atteindre l'état interdit (on note aussi qu'il n'y aura pas de d'état mémoire seulement des « Set » ou « Reset ») et que les sorties des bascules sont forcement complémentées. Ainsi sur les 4 rebouclages, il ne faut retenir que Q_1 et Q_2 car juste 4 états sont possibles (et non 8, il y a redondance). D'après la table de vérité de la $\overline{R}\overline{S}$, $Q_i = R_i$.

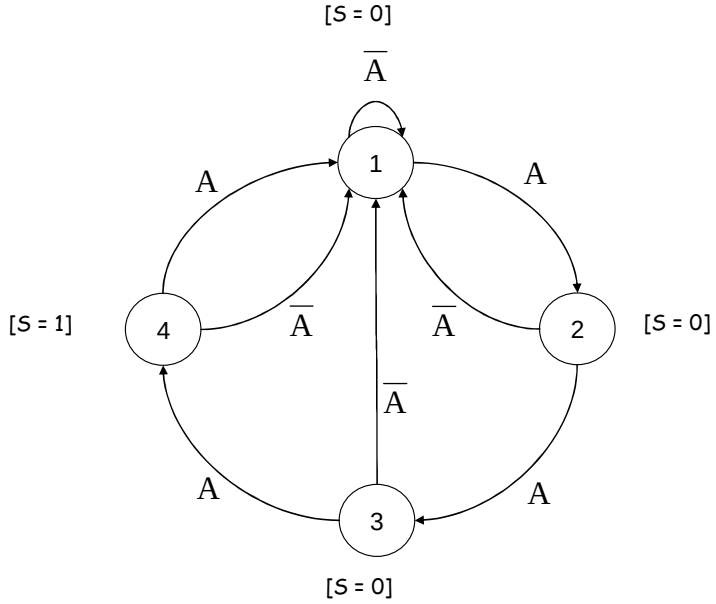
A q_1 q_2 \	0	1
0 0	(00)	01
0 1	00	11
1 1	00	10
1 0	00	00

Excitations secondaires Q_1 Q_2

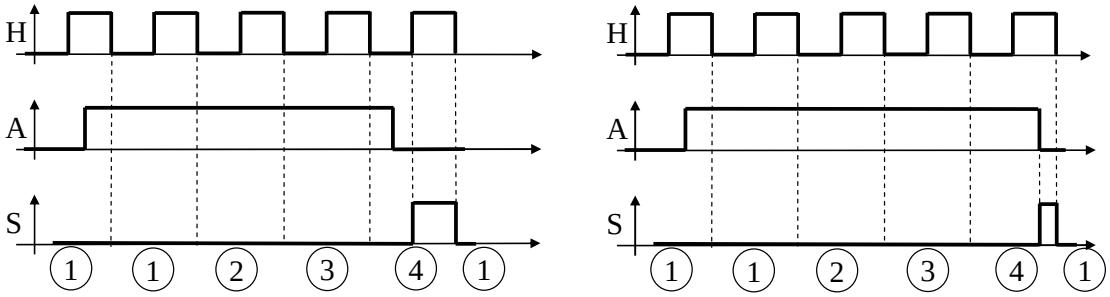
état présent A \	0	1	Sortie S
1	1	2	0
2	1	3	0
3	1	4	0
4	1	1	1

Etats futurs

La sortie $S = Q_1.\overline{A}.H.\overline{Q_2}$ recopie un pulse identique à l'horloge si l'entrée a vaut '0' et si le système se trouve dans l'état 4.



état présent A \	0	1
1	0	0
2	0	0
3	0	0
4	1	0
	$Q_1.\overline{A}.\overline{Q_2}$	



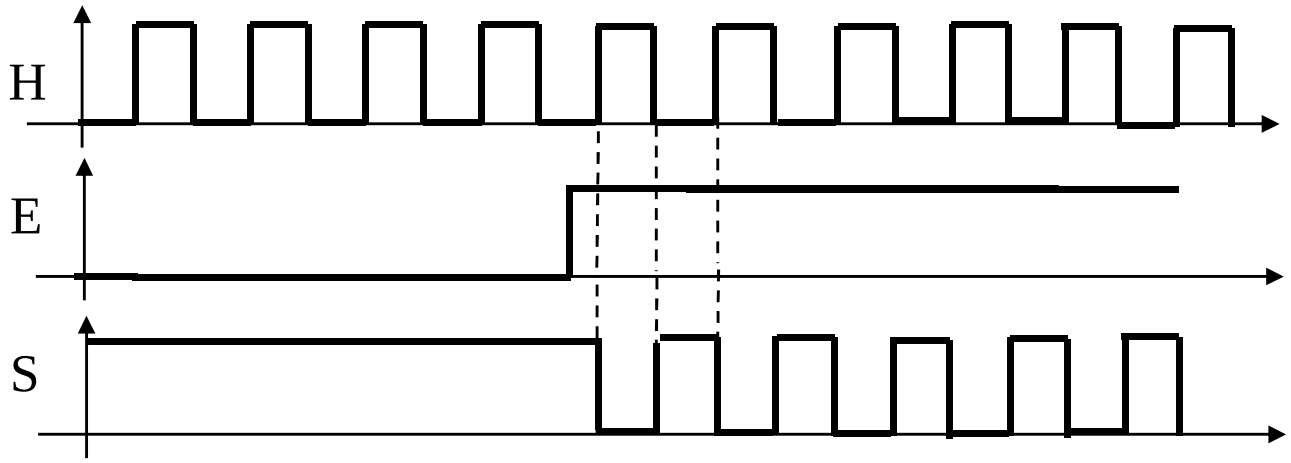
Le chronogramme nous montre bien que la sortie n'est pas synchrone avec l'horloge

mais avec l'entrée, même si le système évolue au rythme de l'horloge la dernière porte logique $S = Q_1.\overline{A}.H.\overline{Q_2}$ remet en cause cette synchronisation. C'est la distinction entre les structures de Moore (sorties synchrones avec l'horloge) et de Mealy (sorties synchrones avec l'une des entrées).

Exercice 4 : *La bascule D.*

La bascule D recopie la valeur de son entrée D manière à chaque front actif de l'horloge. Comme dans l'énoncé rien n'est précisé à cet égard, nous supposerons, comme c'est majoritairement le cas que le front montant est actif.

Cet exercice est assez piégeur puisque pour ce type de montage relativement simple, nous avons tendance à se demander si il est possible de faire l'économie d'une analyse fonctionnelle exhaustive en perdant de vue la structure du circuit. Alors que la seule première question pertinente à se poser est la présence ou non de rebouclage. Il n'y en a pas le système n'étant donc pas séquentiel, la valeur des sorties de dépendent pas de la valeur précédente. L'étude est donc triviale.



Au front montant suivant la transition de E, Q passe à 1 et la sortie devient alors le complément de l'horloge.

Exercice 5 : *Analyse d'un système séquentiel*.

1. Les variables secondaires, facilement identifiables sur le schéma, sont q_1 et q_2. Les entrées de chacune des bascules s'écrivent alors :

$$R_1 = E \bullet q_2, \quad S_1 = E \bullet \overline{q_2}, \quad R_2 = \overline{E} \bullet \overline{q_1}, \quad S_2 = \overline{E} \bullet q_1$$

Ainsi pour E = 0 la bascule 1 est en position mémoire ($R_1 = S_1 = 0$) et la bascule 2 recopie les sorties de la bascule 1 ($R_2 = \overline{q_1}$, $S_2 = q_1$). A l'inverse lorsque E = 1 c'est la bascule 2 qui est en position mémoire ($R_2 = S_2 = 0$). Ce type de structure, appelée maître-esclave, est à la base des bascules synchrones.

A partir de la table des excitations secondaires d'une bascule RS a base de NOR rappelée dans l'énoncé, on obtient la table des excitations secondaires du système étudié.

q_1 q_2	E = 0	E = 1
0 0	0 0	1 0
0 1	0 0	0 1
1 1	1 1	0 1
1 0	1 1	1 0

Q_1, Q_2

La table des transitions s'obtient alors facilement :

0 0 → état 1

0 1 → état 2

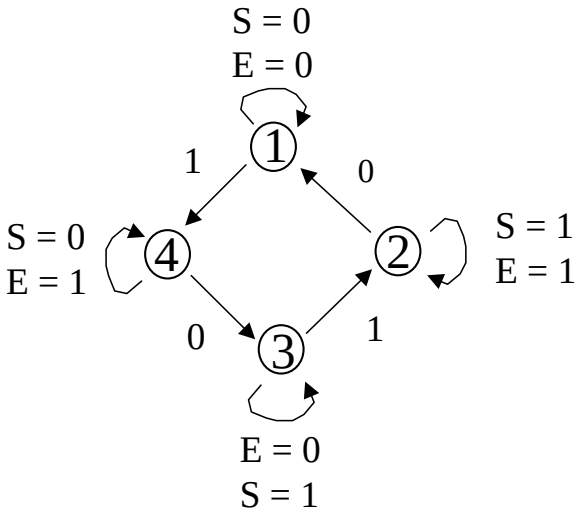
1 1 → état 3

1 0 → état 4

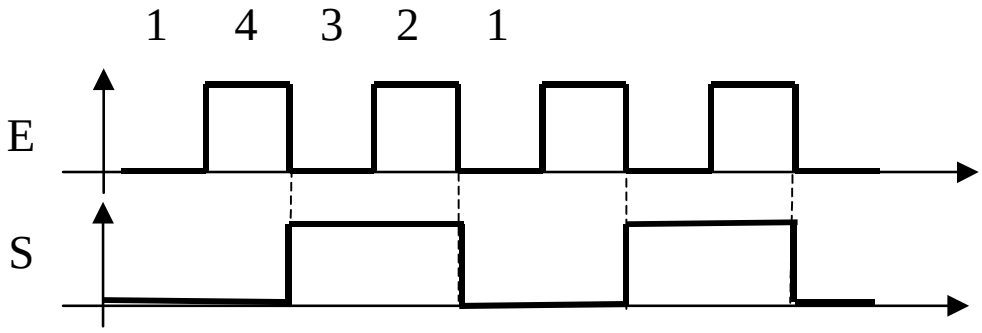
présent	E = 0	E = 1
1	**1**	4
2	1	**2**
3	**3**	2
4	3	**4**

États futurs

Puisque le système est à priori asynchrone, nous avons indiqué les états stables en gras. La sortie S est donnée par $S = Q_2$ et le diagramme des transitions est alors :



2. On obtient le chronogramme suivant :



3. C'est un diviseur de fréquence par 2, actif sur le front descendant de l'entrée E. Il s'agit d'un système synchrone puisque les changements d'état se produisent au moment où l'entrée E reçoit une impulsion.

Exercice 6 : *Analyse d'un système*.

1. On établi directement : $D_A = X \oplus A \oplus B$, $D_B = X \bullet B + A$ et $S = X + A \bullet B$
A la différence du montage de l'exercice 1 la sortie S dépend directement de l'entrée X. La sortie change donc simultanément avec l'entrée X et non plus lorsque le système change d'état. Ce type de système est appelé une machine de Mealy.

2. Pour une bascule D la valeur future de la sortie est la valeur présente sur l'entrée. On obtient ainsi :

A B	**X = 0**	**X= 1**
0 0	0 0	1 0
0 1	1 0	0 1
1 1	0 1	1 1
1 0	1 1	0 1

A^+ B^+

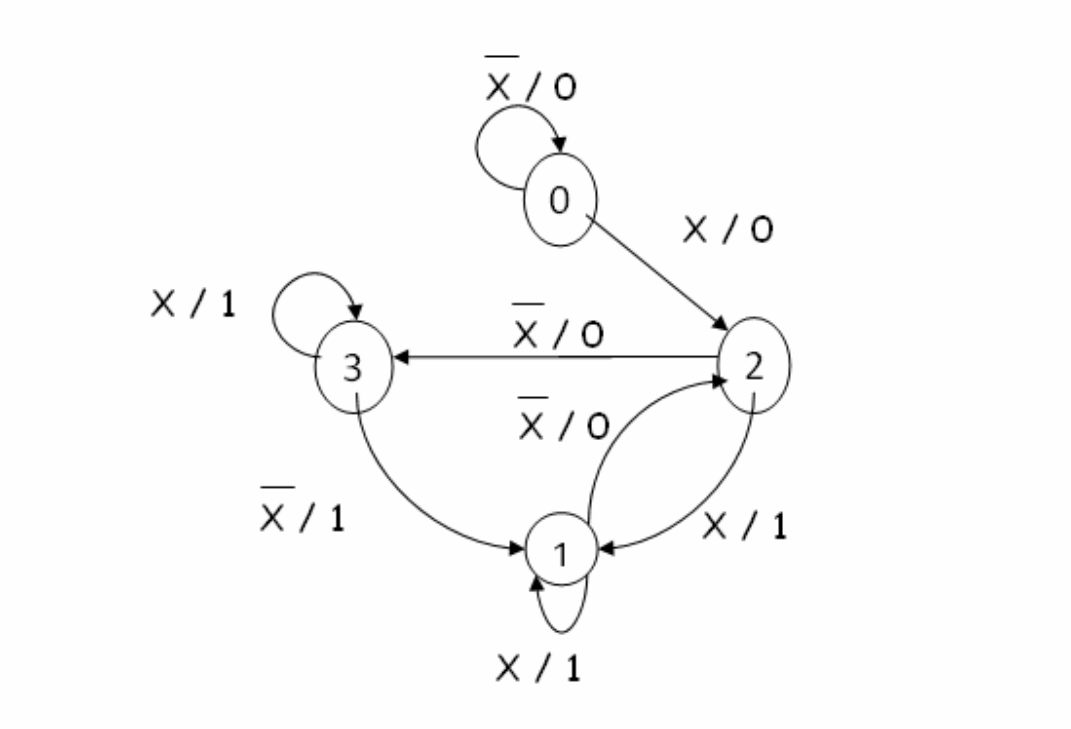
X = 0	**X = 1**
0	0
0	1
1	1
0	1

S

3. En établissant le diagramme d'états il faut bien faire apparaître que la sortie S ne dépend plus uniquement de l'état du système mais aussi de l'entrée X. Pour un même état on peut avoir deux valeurs différentes de la sortie. Par exemple, lorsque le système est dans l'état AB = 01 la sortie vaut 1 si X = 1 et 0 si X = 0. En adoptant la numérotation suivante pour les 4 états :

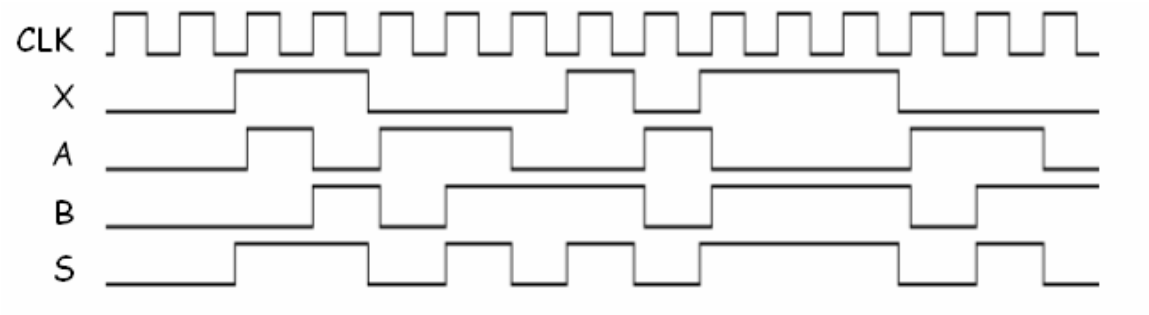
0 0 → 0	1 0 → 2
0 1 → 1	1 1 → 3

On obtient le diagramme d'états.



Depuis l'état 2 (A B = 1 0) la sortie Y = X + A • B vaut 0 ou 1 selon la valeur de X et évolue en même temps que l'entrée. Par contre l'évolution vers l'état futur (ici 1 ou 3) ne se fait que au moment des impulsions d'horloge.

4. On obtient alors le chronogramme:



Encore une fois les variations de la sortie peuvent être simultanées avec les variations de l'entrée et non pas uniquement avec celles de l'horloge.

Exercice 7 : *Analyse d'un montage séquentiel.*

1. les équations des commandes des bascules s'expriment en fonction des excitations secondaires :

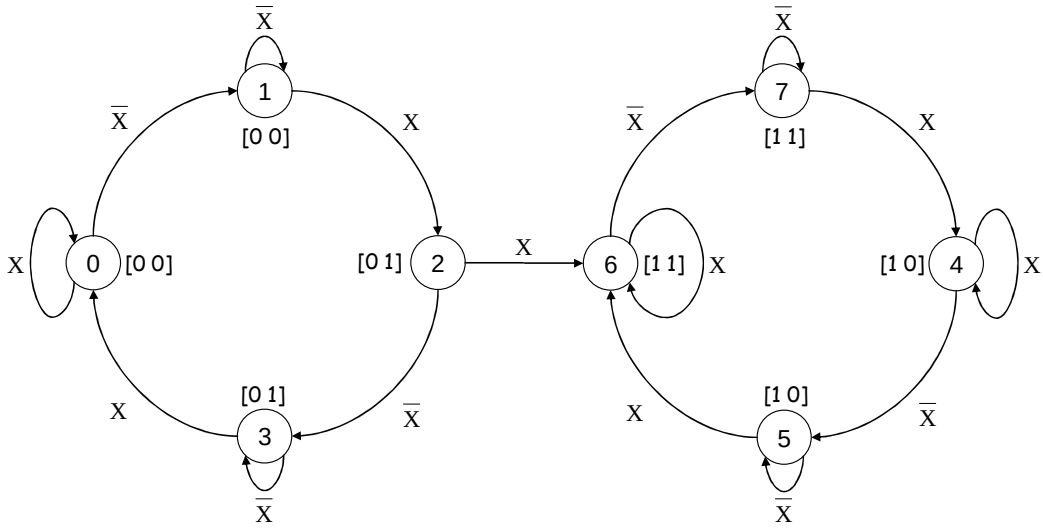
$$D_A = Q_A + Q_B.\overline{Q_C}.X$$
$$D_B = Q_B.\overline{Q_C.X} + \overline{Q_B}.Q_C.X \quad \text{et} \quad S_1 = Q_A$$
$$D_C = \overline{X} \qquad S_0 = Q_B$$

2. le tableau des excitations secondaires :

$Q_A\ Q_B\ Q_C$	X = 1	X = 0
0 0 0	0 0 0	0 0 1
0 0 1	0 1 0	0 0 1
0 1 0	1 1 0	0 1 1
0 1 1	0 0 0	0 1 1
1 0 0	1 0 0	1 0 1
1 0 1	1 1 0	1 0 1
1 1 0	1 1 0	1 1 1
1 1 1	1 0 0	1 1 1

S_1	S_0
0	0
0	0
0	1
0	1
1	0
1	0
1	1
1	1

3. le diagramme d'état



4. le chronogramme : afin de faciliter la relecture, on préfère couramment noter directement sur le chronogramme le nom des états pour chaque créneau d'horloge (voir Figure).

5. Une fois que le système a atteint le second cycle (états de 4 à 7), il n'est pas possible de revenir au premier quelque soit la valeur de X, la sortie restera donc bloquée à '1'.

6. Analyse temporelle.

a) Pour chaque bascule, on doit vérifier que :

$t_{pff} + \Sigma t_{pc} > t_{hold}$ avec t_{pff} = 1ns, t_{hold} = 1ns et Σt_{pc} temps de rétroaction minimum

Bascule A : 1ns +1*0.5ns = 1.5ns > 1ns donc pas de Problème de violation interne sur la A

Bascule B : 1ns +2*0.5ns = 2ns > 1ns donc pas de Problème de violation interne sur la B

Bascule C : Pas de rétroaction

b) La fréquence maximum est donnée par T_{min}

$T_{min} = t_{pff} + t_{pc} + t_{set}$ (en prenant les valeurs maximum des temps)

$= 2ns + 3*1ns + 2ns$

$= 7ns$

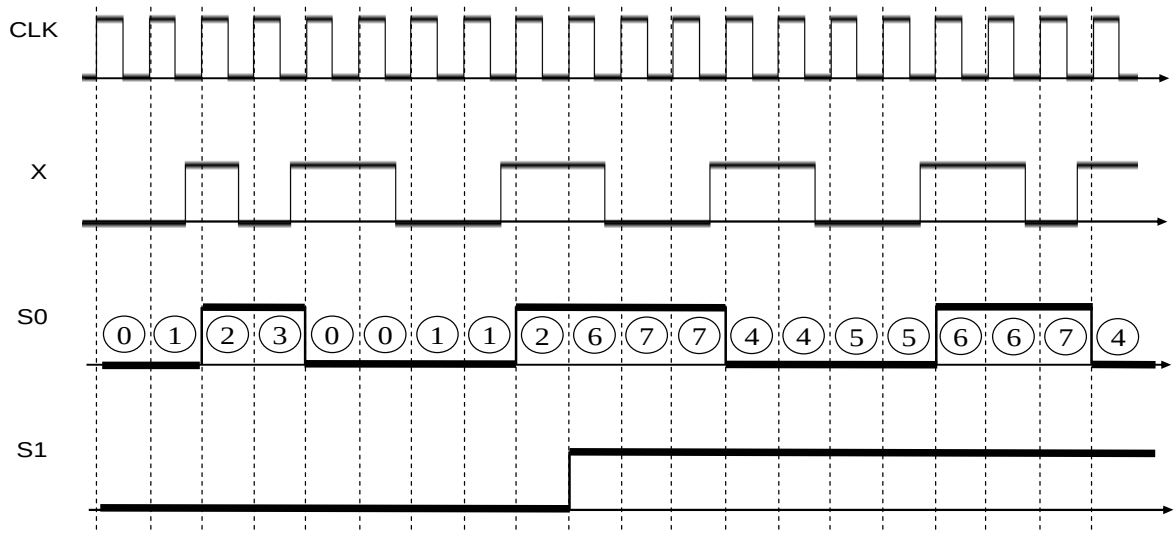
donc F_{max} =142,8 MHz

c) L'entrée X doit être constante entre :

$t_1 = t_H - (t_{set} + dl_{max}) = t_H - (2+3*1ns) = t_H - 5ns$

$t_2 = t_H - (dl_{min} - t_{hold}) = t_H - (0.5ns - 1ns) = t_H - (-0,5ns)$

L'entrée doit donc être constante entre 5 ns avant et 0.5 ns après le front montant de l'impulsion de l'horloge.



Exercice 8 : *Analyse temporelle d'un montage séquentiel.*

2. a) Pour chaque bascule qui présente une rétroaction, il faut se demander si, pour le même front d'horloge, la nouvelle valeur calculée va perturber la bascule par l'intermédiaire de la rétroaction. Autrement dit, les entrées d'une bascule ne doivent pas changer durant t_{hold} or par le rebouclage il est tout a fait possible que la nouvelle donnée résultante du même front d'horloge arrive avant la fin du t_{hold} (Donc dans ce cas il convient de prendre les temps minimum de propagation).

FD1 reboucle sur FD2 : $t_{pff}(FD1) + t_{pc}(OR) = 1ns + 0,25ns > t_{hold} = 1ns$

FD2 reboucle sur elle-même : $t_{pff}(FD2) + t_{pc}(OR) = 1ns + 0,25ns > t_{hold} = 1ns$

Donc pas de problèmes.

b) Pour les violations entre deux pulses d'horloge consécutifs, il faut vérifier que le niveau logique calculé sur le front 1 arrive bien avant le temps de préaffichage du front d'horloge suivant donc dans ce cas il convient de prendre les temps de propagations le plus longs.

Sur le rebouclage de FD1 : $t_{pff}(FD1) + t_{pc}(or) + t_{\underline{set}}(FD2)$

$= 1ns + 0,5ns + 2ns = 3,5ns$

Sur le rebouclage de FD2 : $t_{pff}(FD2) + t_{pc}(inv) + t_{pc}(and) + t_{pc}(or) + t_{\underline{set}}(FD2)$

$= 1ns + 3*0,5ns + 2ns = 4,5ns$

Entre FD2 et FD3 : $t_{pff}(FD2) + t_{pc}(inv) + t_{pc}(and) + t_{pc}(and) + t_{\underline{set}}(FD3)$

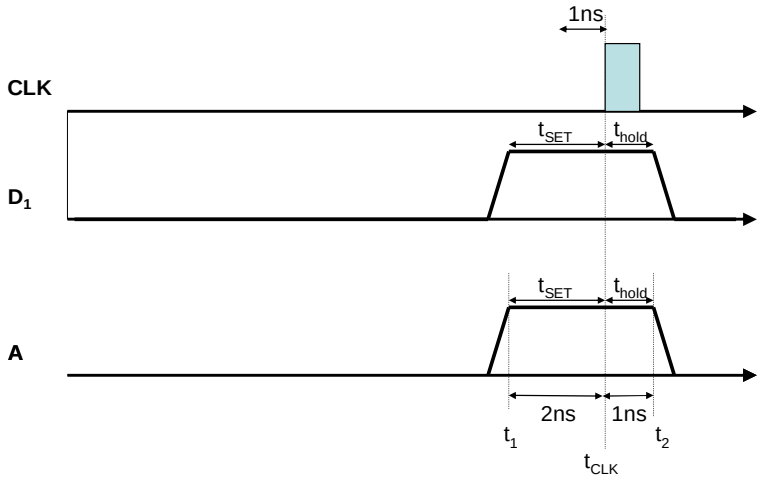
$= 1ns + 3*0,5ns + 2ns = 4,5ns$

dans les deux cas les plus défavorables, la fréquence de l'horloge doit être

inférieure à 220MHz pour respecter le temps de préaffichage des bascules.

3. Jusqu'à présent nous avons examiné si les rebouclages n'induisaient pas de violations des temps de fonctionnement, maintenant il faut définir les conditions de stabilité sur les entrées. La situation est plus simple car il n'y pas de contraintes introduites par le rebouclage, il revient à l'utilisateur de veiller sur un intervalle de temps approprié où les entrées doivent rester stables. La seule entrée est « **A »**

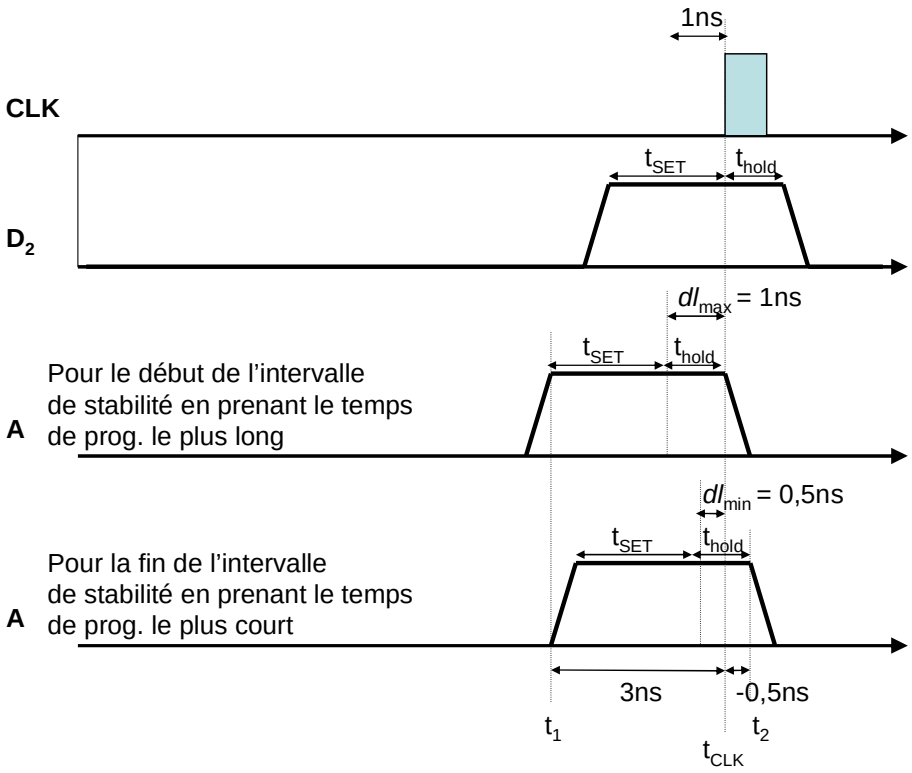
Sur FD1 :



L'entrée est directement reliée à la bascule et doit rester stable entre t_1 et t_2 soit durant 3ns et ce 2ns avant le pulse d'horloge

Sur FD2 :

il y a 2 portes entre l'entrée et la bascule et on considéra les temps de propagation minimum dl_{min} et maximum dl_{max} pour connaître l'intervalle de temps le plus large (c'est-à-dire le plus contraignant) durant lequel l'entrée doit être stable.

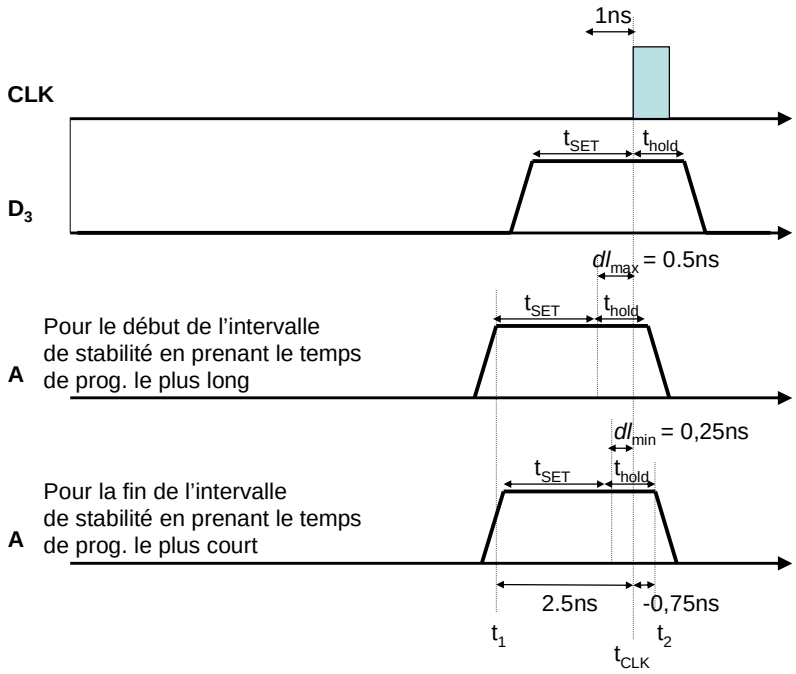


$t_{\underline{set}}$ (FD2) + dl_{max} = 2+2*0,5 = 3ns et

dl_{min} – t_{hold} (FD2) = 2*0,25 – 1 = -0,5ns

A doit donc rester stable entre t_1 et t_2 soit durant 3.5ns et ce 3ns avant le pulse d'horloge.

Sur FD3 : il y a 1 porte entre l'entrée et la bascule



$t_{\underline{set}}$ (FD3) + dl_{max} = 2+0,5 = 2,5ns et

$dl_{min} - t_{hold}$ (FD3) = 0,25 – 1 = -0,75ns

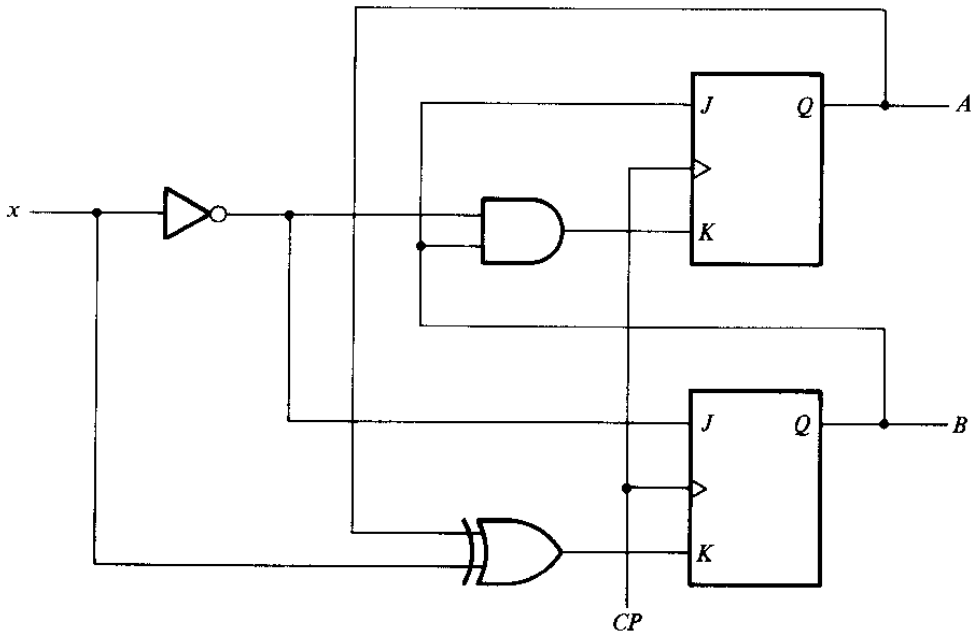
A doit rester stable entre t_1 et t_2 soit durant 3.25ns et ce 2,5ns avant le pulse d'horloge

Si on retient les temps extrêmes des 3 cas envisagés, **A** doit rester stable durant 4ns et ce 3ns avant le pulse d'horloge

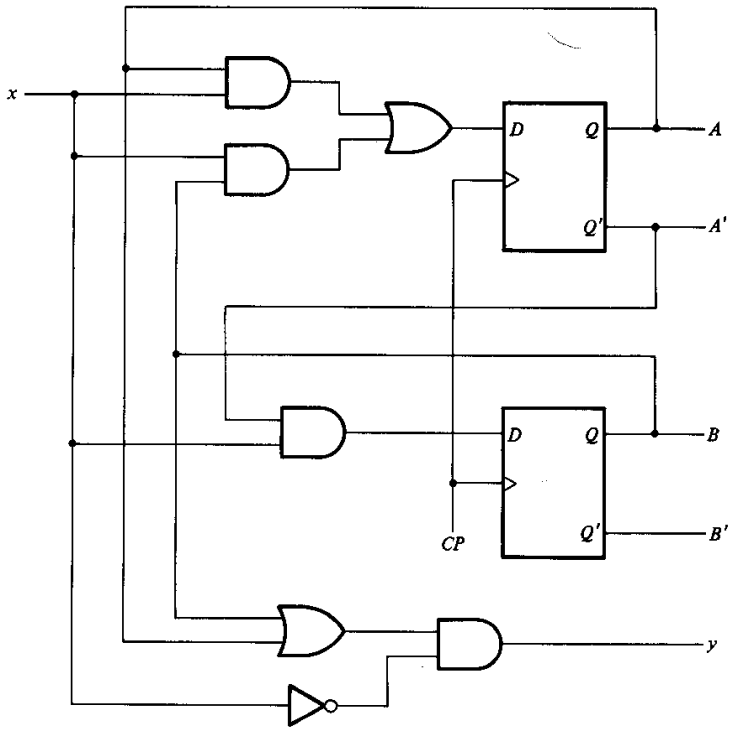
2.7. *Exercices supplémentaires.*

Exercice 1 : *Analyse temporelle d'un montage séquentiel.*

A.



B.



1. S'agit-il d'une machine de Moore ou de Mealy ? Justifier.

2. Etablir les équations de commande des 3 bascules D.

3. Etablir les équations des entrées de chaque bascule.

4. Ecrire la table des transitions donnant les valeurs futures des sorties en fonction des valeurs présentes et des entrées.

5. Tracer le diagramme de transition.

6. Donner les fonctions possibles.

Chapitre 3

Synthèse des systèmes séquentiels synchrones.

Dans le chapitre précédent nous sommes partis du schéma d'un circuit pour aboutir au diagramme d'états qui en décrit le fonctionnement. Le concepteur de circuits numériques est confronté au problème inverse ; à partir d'un cahier des charges il souhaite obtenir un diagramme d'états dont il pourra tirer les équations de commande des bascules du registre d'état.

3.1. *Principe de la synthèse.*

3.1.1. La méthode.

C'est, un peu sommairement, l'inverse de l'analyse. Les principales étapes sont les suivantes :

1. Enumération des états indispensables au fonctionnement du système (à partir du cahier des charges), et établissement du diagramme des transitions.

2. Construction de la table des états et de la table des sorties.

3. Codage des états. En synthèse le nombre M d'états nécessaires au fonctionnement du système est issu du cahier des charges et apparaît dans le diagramme d'états. Cela fixe la taille minimum du registre d'états et donc le nombre de bascules nécessaires au circuit. Si n est le nombre de bascules dans le registre alors le nombre N d'états accessibles est donné par : $N = 2^n$

Lorsque tous les états disponibles ne sont pas utilisés ($M < N$). Il faut impérativement penser à étudier l'évolution des N - M états "inutilisés" sous peine de graves dysfonctionnements du système.

4. Ecriture de la table des transitions et obtention des équations de commandes des bascules. Calcul des sorties.

5. Réalisation du circuit.

Aujourd'hui, les logiciels de synthèse libèrent le concepteur de circuits numériques d'une partie des étapes précédentes (principalement les deux dernières), lui permettant de se concentrer sur les points importants de la synthèse. Avant de revenir en détail sur certains de ces points, donnons d'ores et déjà un premier exemple de synthèse.

3.1.2. Retour sur le compteur.

Reprenons l'exemple du compteur du paragraphe 2.3.3. On souhaite modifier le montage en y ajoutant une entrée supplémentaire *B* telle que :

- Si *B* = 1 le système fonctionne comme précédemment. Il compte si *A* = 1 et reste bloqué dans son état actuel lorsque *A* = 0.
- Si *B* = 0 le compteur est remis à 0 quelque soit la valeur de l'entrée *A*. Autrement dit le compteur retourne dans l'état 1 pour lequel $S_1 S_0 = 00$.

Le nouveau diagramme d'états est représenté sur la figure 23.

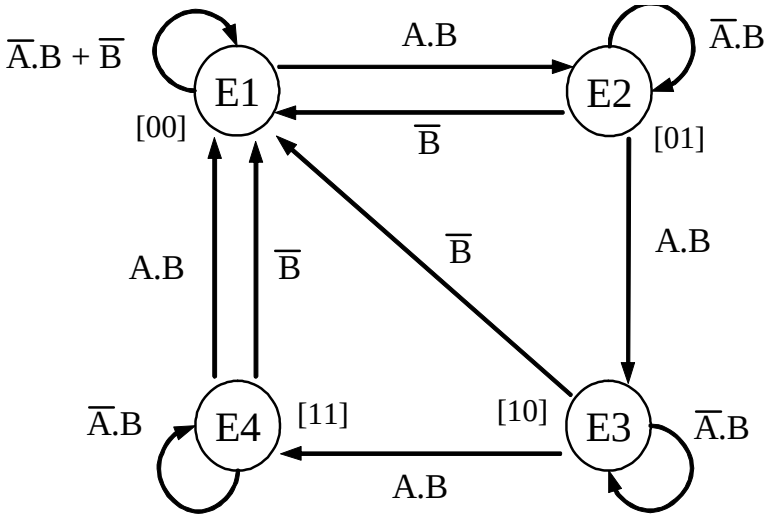


figure 23 Diagramme de transition du compteur. Pour chaque état les valeurs des sorties [$S_1 S_0$] est indiquée

Le problème est maintenant de savoir comment doivent être commandées les deux

bascules *D* pour que le système se comporte conformément au diagramme. On peut procéder de deux façons. Soit écrivant la table des transitions, soit en obtenant directement les équations de commande des bascules à partir du diagramme.

Raisonnement avec les tables.

On commence par écrire la table des états donnant l'évolution de chaque état pour toutes les combinaisons possibles des entrées *A* et *B*.

	A	**0**	**0**	**1**	**1**
présent	**B**	**0**	**1**	**1**	**0**
E1		E1	E1	E2	E1
E2		E1	E2	E3	E1
E3		E1	E3	E4	E1
E4		E1	E4	E1	E1

futur

On choisit ensuite un codage et on en déduit la table des transitions. Pour une bascule *D* l'état futur de la sortie est simplement la valeur présente sur l'entrée au moment de l'impulsion. La table des transitions donne donc directement les valeurs des entrées des bascules. En utilisant le codage utilisé dans le paragraphe 2.3.3, on obtient la table suivante (attention on a inversé l'ordre des lignes des états 3 et 4 pour faire apparaître directement une table de Karnaugh) :

	A	**0**	**0**	**1**	**1**
$Q_1\ Q_0$	**B**	**0**	**1**	**1**	**0**
0 0		0 0	0 0	0 1	0 0
0 1		0 0	0 1	1 0	0 0
1 1		0 0	1 1	0 0	0 0
1 0		0 0	1 0	1 1	0 0

Q_1^+, Q_0^+

D_1, D_0

Soit directement, soit en séparant la table précédente en deux tables de Karnaugh, on obtient les expressions logiques qui commandent les entrées D_1 et D_0.

$$D_1 = \overline{A} \bullet B \bullet Q_1 + A \bullet B \bullet \overline{Q_1} \bullet Q_0 + B \bullet Q_1 \bullet \overline{Q_0}$$

$$D_0 = \overline{A} \bullet B \bullet Q_0 + A \bullet B \bullet \overline{Q_0} = B \bullet (A \oplus Q_0)$$

Le schéma du circuit ne présente pas de difficultés il est donné sur la figure 24. Sur cette figure on a fait apparaître explicitement le registre d'état dont les sorties définissent l'état présent, et le bloc combinatoire F qui calcule l'état futur.

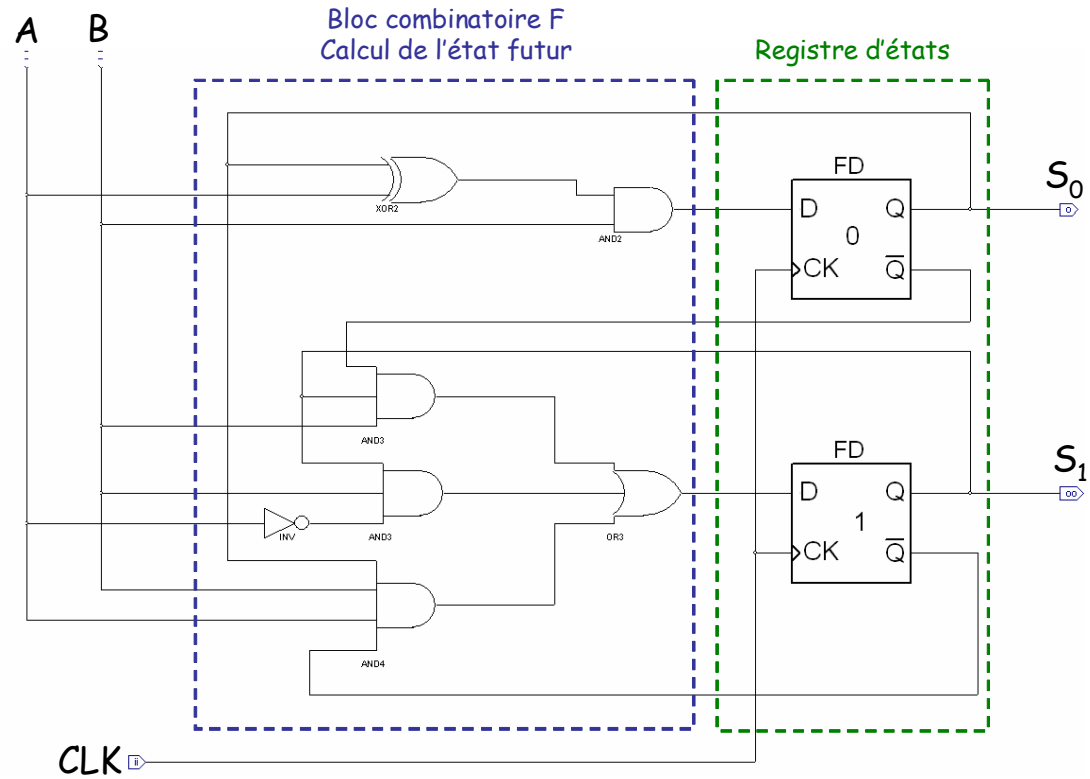


figure 24 Schéma de la nouvelle version du compteur. Les sorties du registre d'état correspondent directement aux sorties du compteur. Il n'y a donc pas de bloc de calcul pour les sorties du compteur

Raisonnement direct à partir du diagramme d'états.

La méthode précédente est systématique mais s'avère rapidement très lourde lorsque

le nombre de variables augmente. En fait, à partir du moment où le codage des états est connu, le passage du diagramme d'étatsaux équations de commandes des bascules est immédiat.

Pour chaque bascule on recense toutes les transitions (dont éventuellement celles qui laissent l'état inchangé) à l'issue desquelles la sortie de la bascule est à 1. Dans ce système il y a deux bascules :

- La sortie Q_1 est au niveau 1 après les transitions E2 → E3 (en conséquence on doit avoir D_1=1 lorsque $Q_1\,Q_0 = 0\,1$ et A = 1,B = 1 au moment de l'impulsion d'horloge.), **OU** E3 → E4 (on doit avoir D_1=1 quand $Q_1\,Q_0 = 1\,0$ et A = 1, B = 1 au moment de l'impulsion) **OU** E3 → E3 (D_1=1 quand $Q_1\,Q_0 = 1\,0$ et A = 0, B = 1 au moment de l'impulsion) **OU** E4 → E4 (D_1=1 quand $Q_1\,Q_0 = 1\,1$ et A = 0, B = 1 au moment de l'impulsion). On aboutit ainsi à l'expression :

$$D_1 = \underbrace{A \bullet B \bullet \overline{Q_1} \bullet Q_0}_{\text{état 2 et A=1, B=1}} + \underbrace{A \bullet B \bullet Q_1 \bullet \overline{Q_0}}_{\text{état 3 et A=1, B=1}} + \underbrace{\overline{A} \bullet B \bullet Q_1 \bullet \overline{Q_0}}_{\text{état 3 et A=0, B=1}} + \underbrace{\overline{A} \bullet B \bullet Q_1 \bullet Q_0}_{\text{état 4 et A=0, B=1}}$$

$$D_1 = \overline{A} \bullet B \bullet Q_1 + A \bullet B \bullet \overline{Q_1} \bullet Q_0 + B \bullet Q_1 \bullet \overline{Q_0} \quad \text{après simplifications}$$

- La sortie Q_0 est au niveau 1 après les transitions E1 → E2 (on doit avoir D_0=1 lorsque $Q_1\,Q_0 = 0\,0$ et A = 1, B = 1 au moment de l'impulsion d'horloge.) **OU** E2 → E2 (on doit avoir D_0=1 quand $Q_1\,Q_0 = 0\,1$ et A = 0, B = 1 au moment de l'impulsion) **OU** E3 → E4 (donc D_0=1 quand $Q_1\,Q_0 = 1\,0$ et A = 1, B = 1 au moment de l'impulsion) **OU** E4 → E4 (donc D_0=1 quand $Q_1\,Q_0 = 1\,1$ et A = 0, B = 1 au moment de l'impulsion). On aboutit ainsi à l'expression :

$$D_0 = \underbrace{A \bullet B \bullet \overline{Q_1} \bullet \overline{Q_0}}_{\text{état 1 et A=1, B=1}} + \underbrace{\overline{A} \bullet B \bullet \overline{Q_1} \bullet Q_0}_{\text{état 3 et A=0, B=1}} + \underbrace{A \bullet B \bullet Q_1 \bullet \overline{Q_0}}_{\text{état 3 et A=1, B=1}} + \underbrace{\overline{A} \bullet B \bullet Q_1 \bullet Q_0}_{\text{état 4 et A=0, B=1}}$$

$$D_0 = A \bullet B \bullet \overline{Q_0} + \overline{A} \bullet B \bullet Q_0 = B \bullet (A \oplus Q_0) \quad \text{après simplifications}$$

Cette seconde méthode est peut être un peu moins simple que la première, mais nettement plus souple et rapide lorsque l'on a un peu d'habitude

Comme le montre cet exemple, la réalisation d'un diagramme d'états correct et sans ambiguïtés est primordiale lors de la synthèse d'un circuit numérique. On ne peut pas espérer obtenir un système fiable à partir d'un diagramme ambiguë ou incomplet. Nous allons donc détailler dans le paragraphe suivant les précautions à prendre pour établir un diagramme d'états correct.

Précisions sur le codage des états.

La solution précédente dépend complètement du codage. Le codage du paragraphe 2.3.3, que nous venons d'utiliser, est intéressant car il permet d'associer directement les sorties des bascules aux sorties des compteurs ($S_1 = Q_1$ et $S_0 = Q_0$). Cependant, il faut bien comprendre que ce n'est pas la seule possibilité et que d'autres codages sont possibles. Il faudra simplement ajouter des portes combinatoires pour calculer les sorties S_1 et S_0 à partir des sorties Q_1 et Q_0 des bascules. Considérons, par exemple, le codage suivant :

$Q_1\ Q_0$ = 0 0 $\Rightarrow$ état E1

$Q_1\ Q_0$= 0 1 $\Rightarrow$ état E2

$Q_1\ Q_0$ = 1 0 $\Rightarrow$ état E3

$Q_1\ Q_0$ = 1 1 $\Rightarrow$ état E4

On obtient alors les expressions suivantes pour les entrées des deux bascules:

$$D_0 = \overline{B} + \overline{A} \bullet B \bullet Q_0 + \overline{Q_1} \bullet Q_0 + A \bullet B \bullet Q_1$$

$$D_1 = \overline{A} \bullet B \bullet (Q_1 \oplus Q_0) + A \bullet B \bullet \overline{(Q_1 \oplus Q_0)}$$

et pour les sorties les expressions :

$$S_0 = Q_1 \bullet Q_0 + \overline{Q_1} \bullet \overline{Q_0} = \overline{(Q_1 \oplus Q_0)}$$

$$S_1 = Q_1 \bullet \overline{Q_0} + \overline{Q_1} \bullet \overline{Q_0} = \overline{Q_0}$$

puisque la sortie S_0 doit être à 1 pour les états E2 et E4, et la sortie S_1 égale 1 pour les états E3 et E4.

3.2. *Les diagrammes de transitions*

Que l'on effectue une synthèse à la main (ce qui est de plus en plus rare), ou que l'on utilise un logiciel de conception assistée par ordinateur, il y a des règles strictes à respecter et des pièges à éviter lorsque l'on établit un diagramme d'états

3.2.1. Les règles importantes.

Deux règles sont impératives :

1. L'automate est toujours dans l'un des états représenté. Ce qui signifie que la condition de maintien dans un état est le complément de toutes les conditions de sortie (figure 25.a).

2. L'automate n'est jamais dans plus d'un état à la fois. Les conditions de sortie d'un état doivent donc être mutuellement exclusives (figure 25.b).

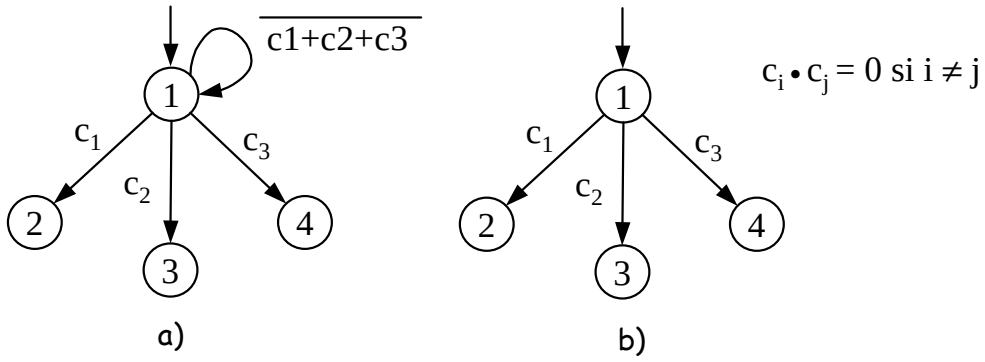


figure 25 Illustration des règles de grammaire à respecter lors nde l'élaboration d'un diagramme d'états

Même si ces règles paraissent évidentes et de bon sens, il est très facile, lors de l'élaboration d'un diagramme de ne pas les respecter. Ceci est particulièrement vrai lorsque le cahier des charges, volontairement ou non, laisse une grande liberté au concepteur du circuit. Pour illustrer ce point reprenons une nouvelle fois l'exemple du compteur en lui apportant les modifications suivantes :

- Lorsque *A* = 1 le compteur compte comme précédemment (0, 1, 2, 3, 0, ...). Lorsque *A* = 0 il reste bloqué dans l'état où il se trouve.
- Lorsque *B* = 0 le compteur doit être remis à zéro. Par contre lorsque *B* = 1 il doit décompté (0, 3, 2, 1, 0, 3 ...).

Enoncé de cette façon le cahier des charges n'est pas aussi précis qu'il y paraît et peut entraîner des problèmes. Deux versions du diagramme d'états pour ce nouvel automate

sont proposées sur la figure 26. La version de gauche n'est pas correcte et comporte plusieurs erreurs.

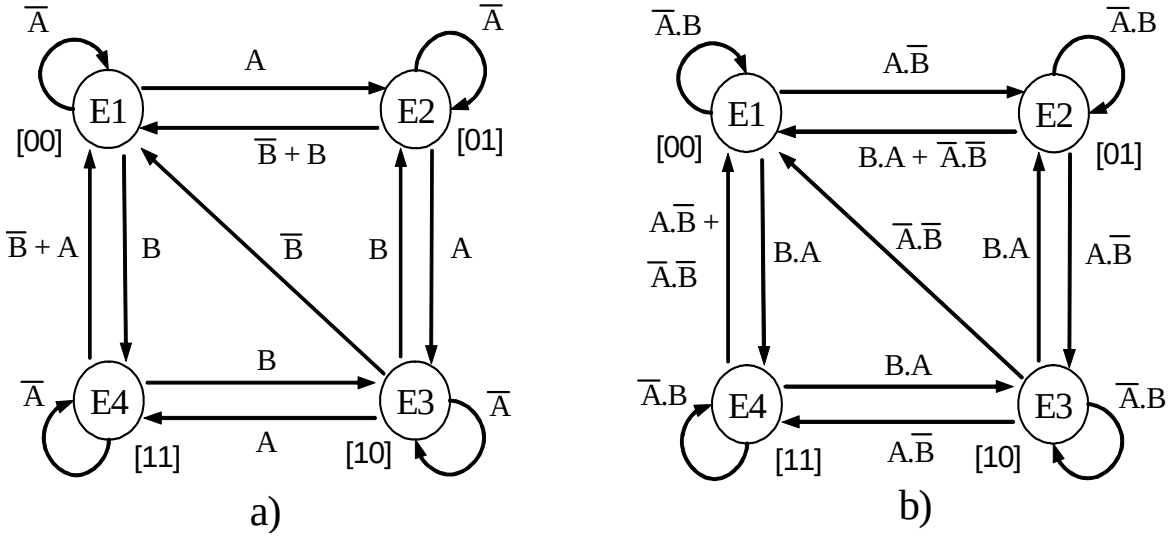


figure 26

- La règle n°2 n'est pas respectée ce qui entraîne des conflits sur l'évolution des différents états. En effet, à partir de n'importe lequel des 4 états, si *A* et *B* sont tous les deux à 1 le diagramme indique des évolutions contradictoires. Par exemple, à partir de l'état E3, *A* = 1 oblige le système à évoluer vers E4 mais, dans le même temps, *B* = 1 l'oblige à évoluer vers E2!
- Depuis l'état E1 l'évolution n'est pas précisée lorsque *B* = 0 .

Le diagramme de droite corrige ces problèmes. Pour cela il a fallu préciser le cahier des charges et établir le mode de fonctionnement suivant.

- Le compteur compte lorsque *A* = 1 et *B* = 0.
- Il est bloqué lorsque *A* = 0 et *B* = 1.
- Il décompte lorsque *B* = 1 et *A* = 1.
- Il est remis à 0 lorsque *B* = 0 et *A* = 0.

3.2.2. Les états inutilisés (états parasites).

Nous avons déjà évoqué l'existence des états inutilisés, ou parasites, dans le paragraphe 3.1.1. Ils apparaissent lorsque le nombre d'états disponibles dans le registre d'états ($N=2^n$ où n est le nombre de bascules) est supérieur au nombre d'états inclus

explicitement dans le diagramme d'états (M). Ces états peuvent entraîner des disfonctionnements du circuit. C'est le cas, par exemple, lorsque le système peut accéder à l'un de ces états sans pouvoir en ressortir, ou alors lorsque le système oscille indéfiniment entre plusieurs états parasites. Il s'agit alors d'états pièges dans lesquels le système reste bloqué. Bien que cela soit rare, cette situation peut éventuellement être voulue par le concepteur du circuit. Dans ce cas le ou les états pièges apparaissent explicitement dans le diagramme d'états et ne posent pas de problèmes. Les ennuis peuvent survenir lorsque les états inutilisés ne sont pas visibles directement sur le diagramme d'état et sont négligés lors de la conception du circuit.

On peut illustrer cette situation en réalisant un système séquentiel dont les sorties, parfaitement synchrones, oscillent en opposition de phase. On commence par étudier une solution basée sur deux bascules synchrones dont les sorties correspondent directement aux sorties du système. Deux solutions sont proposées sur la figure 27.

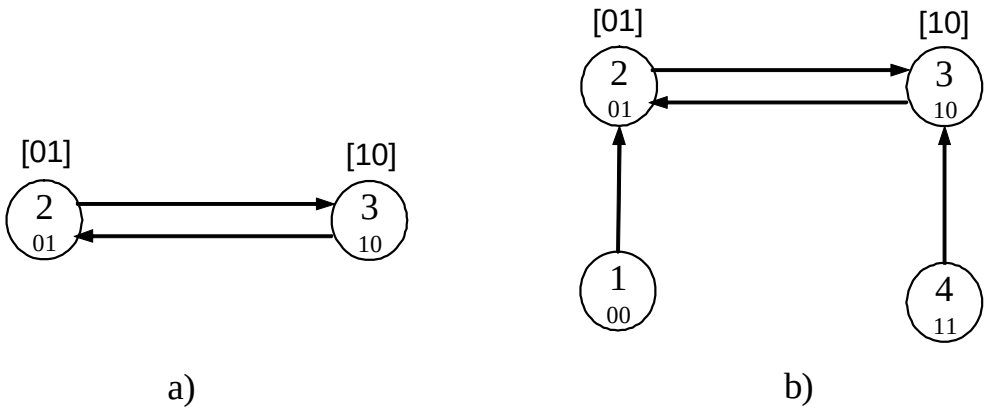


figure 27

1ere solution.

Les sorties du système devant évoluer entre les valeurs $S_1 = 1,\ S_0 = 0$ et

$S_1 = 0,\ S_0 = 1$, on considère le diagramme à 2 états[6] représenté sur la figure 27.a) Le circuit synthétisé à partir de ce diagramme ne fonctionne pas! Le problème vient de l'évolution des états 1 et 4, qui n'apparaissent pas dans le diagramme mais qui existent pourtant bel et bien.

En effectuant la synthèse pour deux bascules D, on obtient aisément les équations de commandes :

$$D_1 = \overline{Q_1} \bullet Q_0$$

$$D_0 = Q_1 \bullet \overline{Q_0}$$

desquelles on déduit l'évolution de l'ensemble des états de l'automate :

$Q_1\ Q_0$	$Q_1^+\ Q_0^+$
0 0	0 0
1 1	0 0
1 0	0 1
0 1	1 0

soit

présent	*futur*
1	1
4	1
3	2
2	3

Les états inutilisés 4 et 1 évoluent séparément des états 2 et 3 figurant sur le diagramme d'états. Si, au moment de la mise sous tension ou sous l'action d'une perturbation extérieure, le système se retrouve dans l'un de ces deux états parasites il restera bloqué dans l'état 1. Ce type de situation peut se produire, par exemple lorsque l'on implémente ces équations dans un circuit programmable (du type de ceux présentés dans le chapitre 4); En effet le système reste obstinément bloqué dans l'état

[6] le numéro des états est donné conformément au codage que nous utilisons depuis le début du chapitre :

$Q_1Q_0 = 01$ état 2

$Q_1Q_0 = 10$ état3

1 après être passé par l'état 4 à la mise sous tension[7].

2eme solution.

On trace un diagramme d'états complet incluant explicitement tous les 4 états du registre d'états. Celui proposé sur la figure 27.b) est plus correct, puisqu'il précise l'évolution des états inutilisés 1 et 4. Si on le synthétise avec des bascules D, les équations de commande correspondantes sont :

$$D_1 = \overline{Q_1} \bullet Q_0 + Q_1 \bullet Q_0 = Q_0$$

$$D_0 = Q_1 \bullet \overline{Q_0} + \overline{Q_1} \bullet \overline{Q_0} = \overline{Q}_0$$

qui ont, par ailleurs, l'avantage d'être plus simples que celles obtenues avec la 1ere solution.

Contrairement à l'exemple précédent, en règle générale les synthèses effectuées en incluant explicitement tous les états dans le diagramme d'états conduisent à des circuits plus compliqués que ceux résultant d'une synthèse oubliant les états inutilisés. C'est alors au concepteur du circuit de faire le choix entre une stratégie de *minimisation des risques* (prise en compte explicite des états inutilisés aboutissant à éventuellement à des circuits complexes) et une stratégie de *minimisation des coûts* (non prise en compte des états inutilisés ce qui donne habituellement des circuits plus simples).

3.2.3. Simplification du diagramme d'états.

Dans l'élaboration d'un diagramme d'états on a plutôt intérêt à choisir le minimum d'états pour assurer le fonctionnement de l'automate Les premiers versions d'un diagramme d'états comportent souvent des états inutiles ou redondants[8]. En soi ce n'est pas dramatique. Cela étant, la suppression des états redondants peut être intéressante si elle permet de réduire la taille du registre d'état (et donc le nombre de bascules utilisées).

Comment reconnaître les états redondants et les supprimer? Directement sur le diagramme d'états on cherche les états qui, ayant les mêmes valeurs des sorties à l'instant *t*, engendre les mêmes valeurs futures des sorties quelles que soient les séquences d'entrées. Ces états sont équivalents entre eux et on peut tous les supprimer

[7] Dans tous les circuits programmables il y a une phase d'initialisation au cours de laquelle toutes les sorties des bascules sont systématiquement mises au niveau 1

[8] On les appelle parfois états équivalents.

sauf un. C'est le cas des états 2 et 4 dans le diagramme de la figure 28

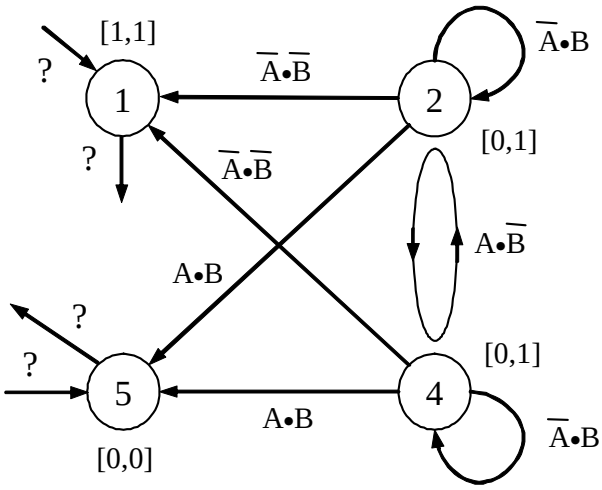


figure 28 Exemple d'états redondants. Les valeurs des sorties sont indiquées entre crochets [S_1 S_0]

Il est peut être plus facile d'adopter la même démarche à partir de la table des transitions et de la table des sorties. On écrit les deux tables cote à cote. Si deux lignes se transforment intégralement l'une en l'autre, par simple échange des numéros de lignes (c'est à dire des états présents), alors les deux états sont redondants et on doit en supprimer un avec toute sa ligne. Dans l'exemple ci dessous la ligne 4 se transforme en ligne 2 par simple changement de 2 en 4 : mêmes états futurs, mêmes valeurs des sorties[9]. On peut donc supprimer l'état (et la ligne) 4.

[9] Si en plus de l'état présent, les sorties dépendent des entrées, il faut écrire la table complète faisant intervenir *A* et *B*.

présent	A / B	0 0	0 1	1 1	1 0
1		**…**	**…**	**…**	**…**
2		1	2	5	4
3		**…**	**…**	**…**	**…**
4		1	4	5	2
5		**…**	**…**	**…**	**…**

futur

présent	S_1	S_0
1	**…**	**…**
2	0	1
3	**…**	**…**
4	0	1
5	**…**	**…**

Sorties

3.3. *Conception des automates.*

L'utilisation de logiciels de synthèse permet aux concepteurs de circuits numériques de se concentrer sur les choix importants d'architecture sans avoir à se soucier des calculs menant aux équations de commandes des bascules. Le travail du concepteur porte donc principalement sur :

- Le découpage du système en sous ensembles simples.
- L'établissement des diagrammes de transitions
- Le choix du type de codage des états.
- Le choix de l'architecture des sorties, et le traitement des entrées et des sorties

3.3.1. Architecture des sorties : machines de Mealy et de Moore.

Selon que les sorties dépendent uniquement des sorties du registre d'états (c'est à dire de l'état présent), ou des sorties du registre d'états *et* des entrées, on distingue deux types d'automates ; les machines de Moore et les machines de Mealy (

figure **29**).

Les machines de Moore.

Les sorties dépendent uniquement de l'état présent ; les variations des sorties sont donc synchrones avec les changements d'états de l'automate *Tous les circuits synchrones que nous avons synthétisés ou analysés jusqu'à présent étaient des machines de Moore.*

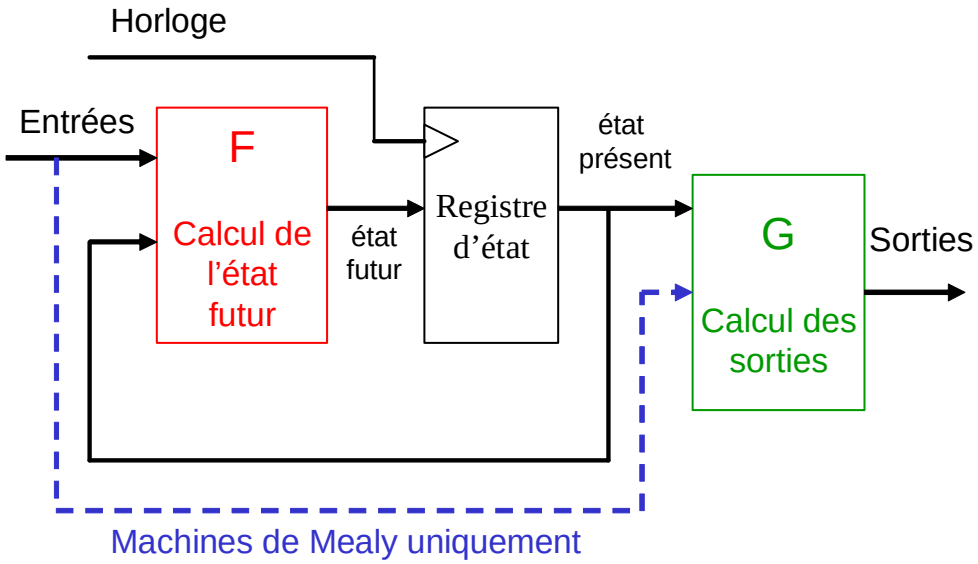


figure 29 Représentation schématique des machines de Moore et Mealy

Les machines de Mealy.

Dans une machine de Mealy les sorties dépendent, en plus de l'état présent, des entrées du système. Contrairement au cas des machines de Moore, les sorties peuvent donc varier entre deux impulsions d'horloge et sont synchrones avec les entrées.

Afin de souligner les différences entre les deux types de machines, nous allons donner un exemple d'automate synthétisé avec les deux méthodes.

3.3.2. Exemple de machine de Moore.

On souhaite réaliser un système dont la sortie *S* passe à 1 lorsque la séquence "1011" a été détectée sur l'entrée *A*. Deux séquences correctes peuvent se superposées. Par exemple, l'entrée *A* = "...1011011..." entraîne la sortie *S* = "...0001001..." le dernier 1 de la première séquence étant le premier 1 de la seconde.

Dans une machine de Moore différentes valeurs des sorties correspondent à des états différents. Le diagramme de transition peut être construit pas à pas.

- Appelons E1 l'état de départ ; si un 1 arrive sur l'entrée cela peut être le début d'une bonne séquence et le système évolue vers l'état E2. A l'inverse si l'entrée reçoit un 0 l'automate reste dans l'état 1(*S* = 0).

- A partir de l'état E2 ; si l'entrée reçoit un 0 c'est la suite d'une bonne séquence et le système évolue vers l'état E3. Par contre si un 1 arrive sur l'entrée la séquence précédente est stoppée et une nouvelle bonne (?) séquence commence : l'automate reste dans l'état E2.

- Depuis l'état E3, l'arrivée d'un 1 signifie la suite de la bonne séquence et le système passe dans l'état E4. A l'inverse un 0 sur l'entrée ramène le système dans l'état E1 puisque l'on est certain que toute la séquence précédente (100) est fausse!

- Depuis l'état E4 l'arrivée d'un 1 valide une séquence correcte "1011", le système passe dans l'état E5 et la sortie *S* = 1.

- Une fois dans l'état E5 l'automate évolue vers soit vers l'état E2 si *A* = 1, soit vers l'état E3 si *A* = 0 (le 1 de l'étape précédente est gardé pour une nouvelle séquence).

Le diagramme d'états est dessiné sur la figure 30.

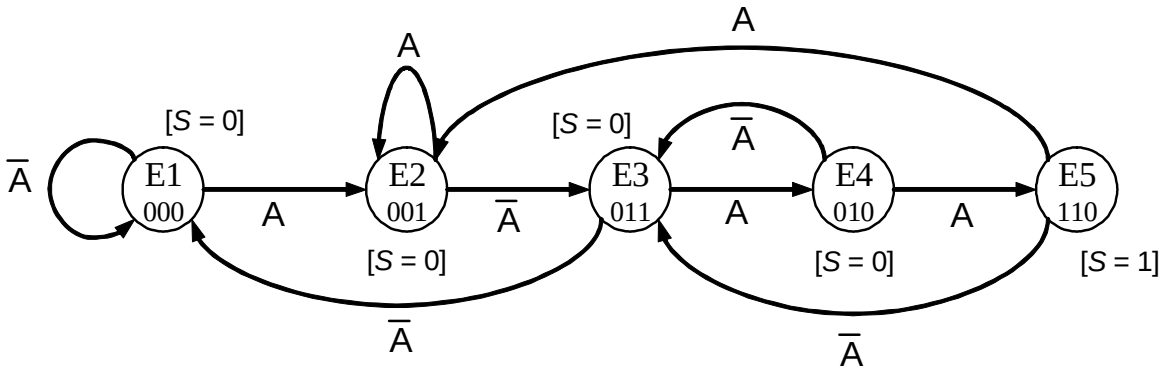


figure 30: Diagramme de transition du détecteur de séquence sous forme de machine de Moore

Le diagramme d'états faisant intervenir 5 états, le registre d'état est composé de 3 bascules. Les exemples précédents de machine de Moore, nous ont montré qu'il était possible de générer les sorties directement à partir du registre d'état. Le choix du codage est donc particulièrement important dans ce type d'architecture. Nous adoptons le codage suivant qui nous permet d'associer la sortie *S* du montage à la sortie Q_2 du registre d'état :

$$\left.\begin{array}{l}\text{état E1} \Rightarrow Q_2\,Q_1\,Q_0 = 000\\ \text{état E2} \Rightarrow Q_2\,Q_1\,Q_0 = 001\\ \text{état E3} \Rightarrow Q_2\,Q_1\,Q_0 = 011\\ \text{état E4} \Rightarrow Q_2\,Q_1\,Q_0 = 010\\ \text{état E5} \Rightarrow Q_2\,Q_1\,Q_0 = 110\end{array}\right\} \text{ et } S = Q_2$$

Par commodité on synthétise le registre d'état avec 3 bascules *D*. Pour établir les équations de commande de ces bascules on écrit d'abord la table des transitions donnant les valeurs futures en fonction des valeurs présentes et de l'entrée *A*, puis on en déduit les expressions logiques des entrées D_i des bascules.

$Q_2\,Q_1\,Q_0$	A = 0	A = 1
0 0 0	0 0 0	0 0 1
0 0 1	0 1 1	0 0 1
0 1 1	0 0 0	0 1 0
0 1 0	0 1 1	1 1 0
1 1 0	0 1 1	0 0 1

Ce qui peut se réécrire sous forme d'une table de Karnaugh:

A / $Q_2\,Q_1$ \ Q_0	**0** **0**	**0** **1**	**1** **1**	**1** **0**
0 0	0 0 0	0 1 1	0 0 1	0 0 1
0 1	0 1 1	0 0 0	0 0 0	1 1 0
1 1	0 1 1	φ φ φ	φ φ φ	0 0 1
1 0	φ φ φ	φ φ φ	φ φ φ	φ φ φ

Q_2^+, Q_1^+, Q_0^+

D_2, D_1, D_0

Bien évidement les φφφ qui apparaissent dans la table précédente sont liés aux états supplémentaires ($Q_2\,Q_1\,Q_0 = 111, 100, 101$) qui n'interviennent pas dans le diagramme d'états mais dont il faut surveiller l'évolution si l'on veut s'assurer du fonctionnement de l'automate! Les équations de commandes des trois bascules sont[10] :

$$D_0 = Q_2 + Q_0 \bullet \overline{Q_1} + A \bullet \overline{Q_1} + Q_1 \bullet \overline{Q_0} \bullet \overline{A}$$

$$D_1 = Q_1 \bullet \overline{Q_0} \bullet \overline{A} + \overline{Q_1} \bullet Q_0 \bullet \overline{A} + \overline{Q_2} \bullet Q_1 \bullet A$$

$$D_2 = \overline{Q_2} \bullet Q_1 \bullet \overline{Q_0} \bullet A$$

On montre facilement que les états "parasites" ne peuvent entraîner de blocage du système.

La figure 31 représente l'évolution du registre d'états et de la sortie *S* en fonction de l'entrée *A*. On constate bien que la sortie *S* passe à 1 une fois que la séquence 1011 a été détectée et que le système est dans l'état E5.

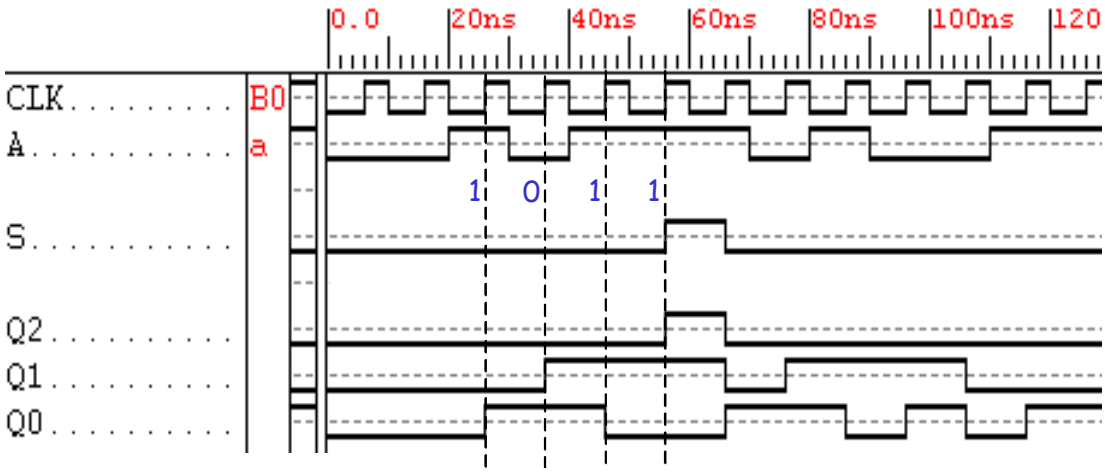


figure 31 Chronogramme du détecteur de séquence synthétisé sous forme de machine de Moore

[10] On peut les obtenir directement à partir de la table de Karnaugh précédente, ou bien scinder celle ci en trois tables indépendantes.

3.3.3. Exemple machine de Mealy.

Nous avons déjà indiqué dans le paragraphe 2.3.4 comment représenter le diagramme d'états d'une machine de Mealy. Pour mémoire, le diagramme représenté sur la figure 32 se lit de la façon suivante :

- Quand l'automate est dans l'état E1, la sortie reste à 1 tant que *A* = 0. Lorsque *A* passe à 1, la sortie devient immédiatement 0 et l'automate effectue la transition $E1 \rightarrow E2$ à l'impulsion d'horloge suivante.

- Quand l'automate est dans l'état E2, la sortie reste à 0 tant que *B* = 1. Lorsque *B* = 0, la sortie passe à 1 et le système effectue la transition $E2 \rightarrow E1$ à l'impulsion d'horloge suivante.

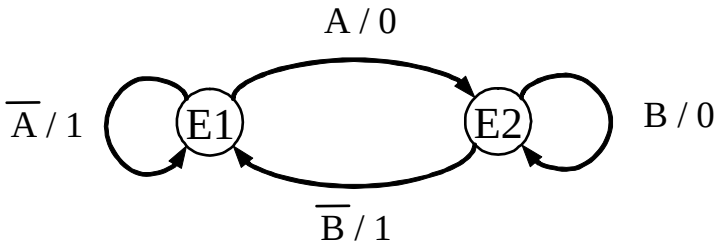


figure 32 Diagramme d'états d'une machine de Mealy

Détecteur de séquence sous forme de machine de Mealy

On considère le même problème que précédemment mais on souhaite le résoudre avec une machine de Mealy. Contrairement au cas des machines de Moore, les sorties ne sont plus associées aux états mais aux transitions. Pour établir le diagramme d'états on garde la même démarche que précédemment en l'adaptant aux machines de Mealy.

- Appelons E1 l'état de départ, si un 1 arrive sur l'entrée cela peut être le début d'une bonne séquence : la sortie reste à 0 et le système évolue vers l'état E2 à l'impulsion d'horloge. A l'inverse si l'entrée reçoit un 0 l'automate reste dans l'état E1 (*S*=0).

- A partir de l'état E2, si l'entrée reçoit un 0 c'est la suite d'une bonne séquence, la sortie reste à 0 et le système évolue vers l'état E3 à l'impulsion d'horloge. Par contre si un 1 arrive sur l'entrée la séquence précédente est stoppée et une nouvelle séquence (bonne?) commence : l'automate reste dans l'état E2.

- Depuis l'état E3, l'arrivée d'un 1 signifie la suite de la bonne séquence la sortie reste à 0 et le système passe dans l'état E4 à l'impulsion d'horloge. A l'inverse un 0 sur l'entrée ramène le système dans l'état E1 puisque l'on est certain que toute la séquence précédente (100) est fausse!
- Depuis l'état E4 l'arrivée d'un 1 valide une séquence correcte "1011" la sortie passe *immédiatement* à 1 et le système retourne dans l'état E2 à l'impulsion d'horloge.

Le diagramme d'états est représenté sur la figure 33.

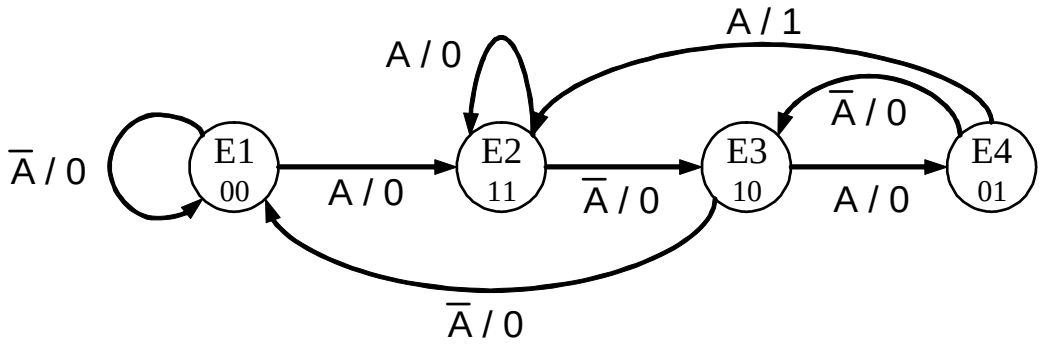


figure 33 Diagramme d'états du détecteur de séquence sous forme de machine de Mealy

On adopte le codage suivant pour représenter les états[11] :

$$\text{état } 1 \Rightarrow Q_1Q_0 = 00$$

$$\text{état } 2 \Rightarrow Q_1Q_0 = 11$$

$$\text{état } 3 \Rightarrow Q_1Q_0 = 10$$

$$\text{état } 4 \Rightarrow Q_1Q_0 = 01$$

La table des transitions et la table des sorties sont alors :

[11] A titre d'exercice on peut vérifier que ce codage permet d'obtenir des équations de commandes plus simples que celles obtenues avec le codage proposé dans le paragraphe 2.3.1

Q_1 Q_0 \ A	**1**	**0**
0 0	0 0	1 1
0 1	1 0	1 1
1 1	1 0	1 1
1 0	0 0	0 1

D_1, D_0

et

Q_1 Q_0 \ A	**1**	**0**
0 0	0	0
0 1	0	1
1 1	0	0
1 0	0	0

S

On obtient alors les équations de commandes des deux bascules et l'expression de la sortie[12] :

$$D_0 = A$$

$$D_1 = Q_0 + \overline{Q_1} \bullet A$$

$$S = \overline{Q_1} \bullet Q_0 \bullet A$$

Le circuit correspondant à cet automate est représenté sur la figure 34. Le chronogramme de la figure 35 représente l'évolution du registre d'états et de la sortie *S* en fonction de l'entrée *A*.

La sortie *S* passe à 1 dès que l'automate arrive dans l'état E4, c'est à dire une période d'horloge plus plutôt que dans la machine de Moore. Cela peut paraître troublant car on a l'impression que *S* = 1 tout de suite après que la séquence "101" a été détectée sur l'entrée. En fait il faut se souvenir que l'on étudie un système synchrone, les seules valeurs a prendre en compte sont celles présentes sur les entrées juste avant l'impulsion d'horloge. Réexaminé sous cet angle, le fonctionnement de l'automate est tout à fait correct! Cela étant, lorsque le système est dans l'état E4 toute variation de l'entrée *A* est reproduite sur la sortie *S* et on peut donc observer des variations intempestives de *S* entre deux impulsions d'horloge. C'est d'ailleurs ce que l'on voit sur la figure 35.

[12] Ici la table des sorties est franchement inutile puisque *S* = 1 uniquement lorsque l'automate est dans l'état E4 et que l'entrée *A* passe à 1!

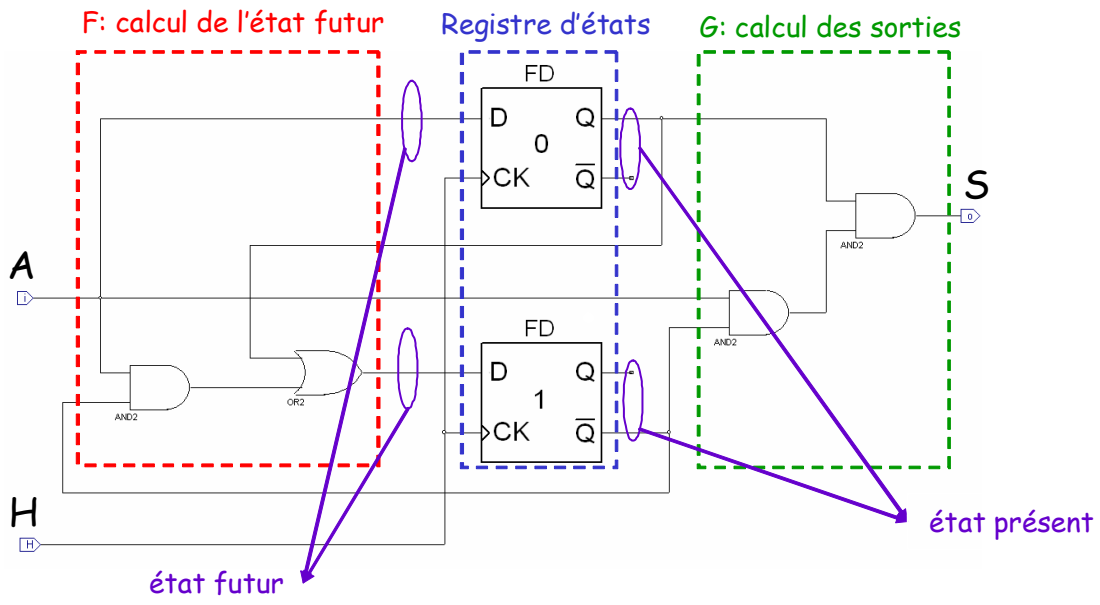


figure 34 Schéma logique du détecteur de séquence synthétisé sous forme de machine de Mealy

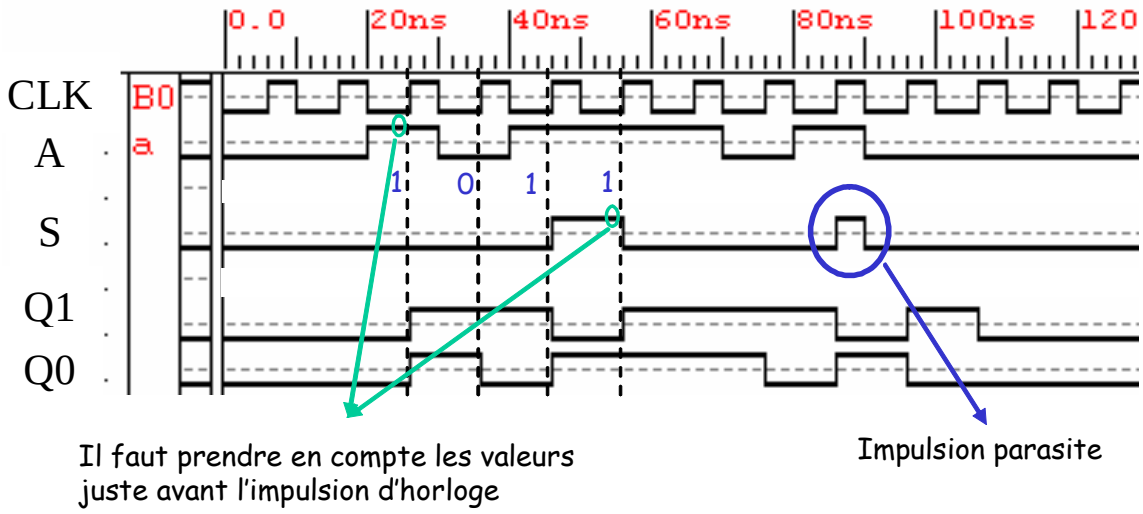


figure 35 Chronogramme du détecteur de séquence sous forme de machine de Mealy

Le problème est bien sûr lié au caractère purement asynchrone du calcul des sorties dans une machine de Mealy. Si on regarde les choses encore plus attentivement on remarque que les retards introduits par les portes combinatoires qui calculent la sortie peuvent également entraîner des risques d'aléas[13] sur la sortie. On pourra d'ailleurs vérifier que c'est le cas avec le circuit de la figure 34.

Machine de Mealy resynchronisée.

La solution aux problèmes précédents est simple : il faut synchroniser la sortie avec l'horloge! On ajoute donc une bascule *D* sur la sortie pour assurer la stabilité entre deux impulsions d'horloge (figure 36). Evidemment cela décale la sortie d'une période d'horloge par rapport aux sorties du registre d'état (voir le chronogramme de la figure 37) mais on ainsi élimine les risques d'aléas et d'impulsions parasites. On peut éventuellement compenser ce retard en calculant la sortie non pas à partir de l'état présent mais à partir de l'état futur.

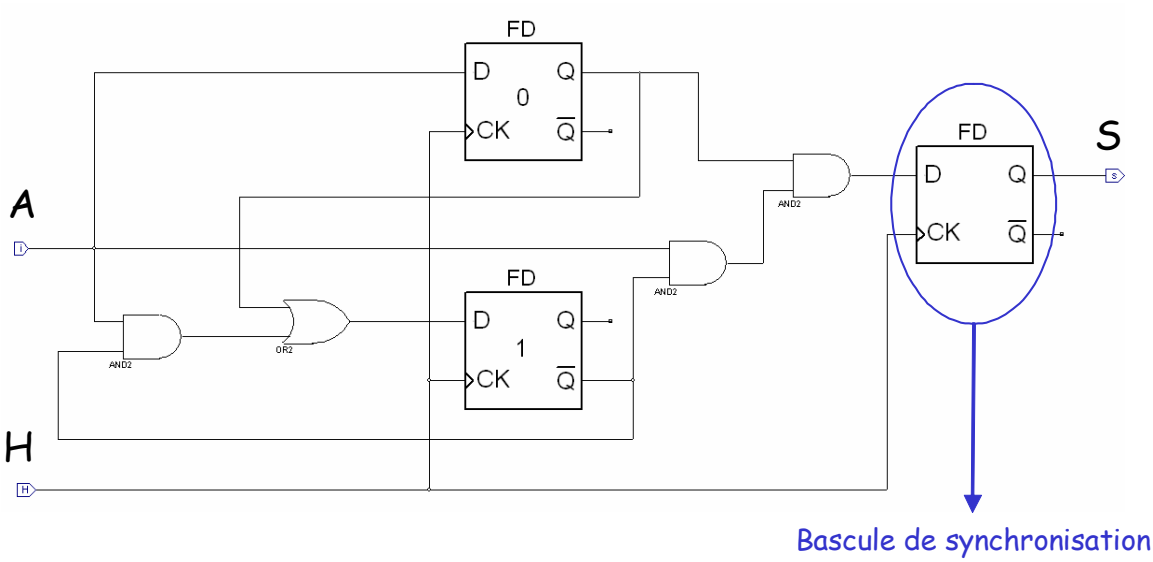


figure 36 Machine de Mealy resynchronisée

Le chronogramme de la machine de Mealy resynchronisée est identique à celui de la machine de Moore (figure 31).

[13] De façon générale un aléa risque de se manifester à la sortie d'un circuit élémentaire **AND** ou **OR** à l'occasion de transitions non simultanées (01)↔(10) sur les entrées.

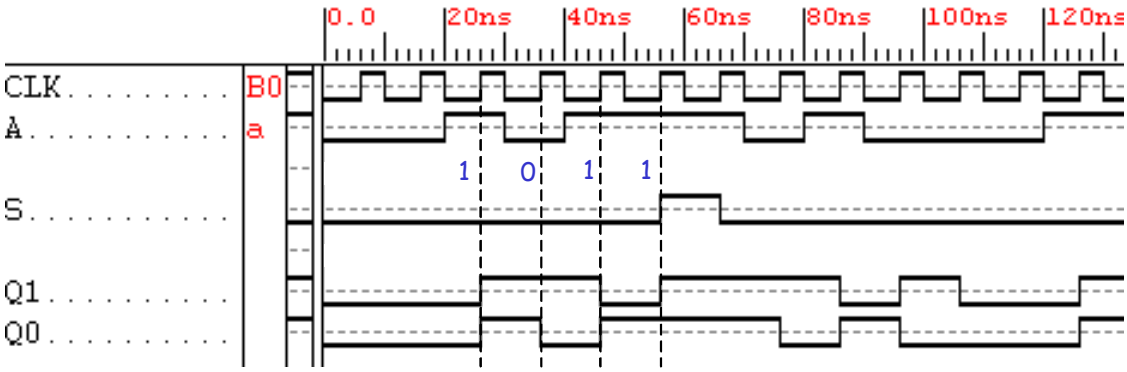


figure 37 Chronogramme de la machine de Mealy resynchronisée.

3.3.4. Avantages et inconvénients des deux types d'architectures.

Evidemment chaque type de machine présente ses avantages et ses inconvénients. Les solutions adoptées en pratique sont souvent un mélange des deux approches dans lesquelles certaines sorties sont traitées en machine de Moore et d'autres en machine de Mealy. Cela étant, les points importants à retenir sont les suivants :

- Puisque les sorties sont associées aux transitions plutôt qu'aux états, les machines de Mealy génèrent les mêmes séquences de sorties avec moins d'états, et donc moins de bascules, que les machines de Moore.
- Les machines de Moore sont d'un emploi plus sûr puisque leurs sorties ne changent qu'au moment des impulsions d'horloge. A l'inverse, les sorties des machines de Mealy changent immédiatement avec les entrées, ce qui peut poser de sérieux problèmes lorsque l'on veut interconnecter plusieurs machines. L'inconvénient est généralement résolu en resynchronisant les sorties.
- On peut bien sur retourner l'argument précédent. Les machines de Mealy ont l'avantage de réagir beaucoup plus rapidement aux changements des entrées. Dans les machines de Moore les sorties sont décalées d'une période d'horloge par rapport aux entrées.

3.3.5. Choix du codage.

Les critères de choix.

Le nombre de portes et de bascules nécessaires à l'implémentation d'un automate

séquentiel, dépend bien sûr du nombre d'états introduits dans le diagramme d'états mais aussi du codage choisi pour représenter chacun de ces états. Il n'existe malheureusement pas de règle absolue permettant de trouver le codage le mieux adapté à un problème donné. Le concepteur de circuits numériques doit donc naviguer entre les contraintes que lui impose le diagramme d'états (essentiellement le nombre d'états) et les critères qu'il souhaite mettre en avant dans la réalisation de l'automate, par exemple:

- minimisation du nombre de bascules,
- choix du type de registre d'états utilisé (bascules D ou JK),
- simplification du calcul de l'état futur et/ou des sorties
- choix de l'architecture des entrées et des sorties (produit de sommes ou somme de produits)

Les contraintes.

Le nombre d'états dans le diagramme (*M*) ainsi que le nombre *n* de bits utilisés pour coder ces états, déterminent la taille du registre d'états. Ces deux nombres vérifient la relation :

$$n \leq M \leq 2^n$$

Lorsque *n* = *M* on utilise un *code très dilué*, une bascule par état, qui évite souvent d'avoir à calculer les sorties mais présente l'inconvénient de générer un grand nombre d'états parasites (il y a toujours 2^n états accessibles). A l'inverse lorsque $M = 2^n$ on parle de *codage dense ou fort*. Il n'y a plus d'états inutilisés mais il faudra vraisemblablement ajouter des fonctions combinatoires pour calculer les sorties.

Quelques règles plus ou moins simples (mais pas obligatoires).

Même si il n'existe pas de règle absolue, un certain nombre de règles peuvent être énoncées pour guider le concepteur de circuit dans le choix du meilleur codage.

- Choisir pour l'état initial un codage correspondant à un état de l'automate facilement accessible par une commande simple telles que *RESET* ou *PRESET* (00…00, ou 11…11).
- Essayer de minimiser le nombre de bits changeant de valeur à chaque transition.
- Donner des codes adjacents (c'est à dire identiques à un bit près) a tous les

états évoluant vers le même état final.

- Donner des codes adjacents à tous les états ayant le même état initial.
- Essayer d'attribuer à chaque bit ou groupe de bits du code une signification particulière concernant les effets des entrées sur le système ou le comportement des sorties.
- Utiliser plus de bit que le minimum nécessaire imposé par le diagramme d'états.

Il existe beaucoup de codages différents et nous nous contenterons de 4 exemples.

Codage aléatoire

C'est bien entendu le plus simple ! On utilise le nombre de bit minimum et on code les différents états de façon totalement aléatoire.

Codage de Moore

On code les états de façon que les sorties de l'automate correspondent directement aux sorties du registre d'état. On simplifie ainsi le calcul des sorties Nous avons utilisé ce type de codage (ou de calcul) consacré aux machines de Moore.

Codes adjacents.

On code les états de façon qu'une transition entre deux états n'induise de changement que sur un seul bit, on parle alors de code adjacent. Ce type de code est bien adapté aux automates dont le diagramme d'état est cyclique comme par exemple les compteurs. En général, même s'il est rarement possible de respecter la règle d'adjacence pour toutes les transitions, il est souvent judicieux de la respecter au moins partiellement.

Le codage proposé dans le paragraphe 3.3.2 pour le détecteur de séquence

$$\left.\begin{array}{l} \text{état E1} \Rightarrow Q_2\ Q_1\ Q_0 = 000 \\ \text{état E2} \Rightarrow Q_2\ Q_1\ Q_0 = 001 \\ \text{état E3} \Rightarrow Q_2\ Q_1\ Q_0 = 011 \\ \text{état E4} \Rightarrow Q_2\ Q_1\ Q_0 = 010 \\ \text{état E5} \Rightarrow Q_2\ Q_1\ Q_0 = 110 \end{array}\right\} \text{ et } S = Q_2$$

en plus d'être un code de type Moore est également un code partiellement adjacent.

Codes un seul actif (one hot encoding).

Les codes ”un seul actif” sont les plus dilués possible puisqu’ils utilisent une bascule par état, l’idée étant de simplifier au maximum le calcul des sorties et de l’état futur. Pour un automate avec *M* états on utilise un code de *n* = *M* bits dont un seul bit est à 1 par état. En termes de bascules cela signifie qu’il y a une seule bascule active par état toutes les autres étant inactives. L'inconvénient de ce type de code réside bien sûr dans le nombre de bascules utilisées. Ils présentent ce pendant l'avantage de simplifier les équations de commande des bascules et s'avèrent avantageux lorsque l'on souhaite obtenir une seule sortie à *1 parmi n*.

Un exemple de "code un seul actif" pour la machine de Moore étudiée dans le paragraphe 3.3.2 serait

$$\left.\begin{array}{l}\text{état E1} \Rightarrow Q_4\ Q_3\ Q_2\ Q_1\ Q_0 = 00001 \\ \text{état E2} \Rightarrow Q_4\ Q_3\ Q_2\ Q_1\ Q_0 = 00010 \\ \text{état E3} \Rightarrow Q_4\ Q_3\ Q_2\ Q_1\ Q_0 = 00100 \\ \text{état E4} \Rightarrow Q_4\ Q_3\ Q_2\ Q_1\ Q_0 = 01000 \\ \text{état E5} \Rightarrow Q_4\ Q_3\ Q_2\ Q_1\ Q_0 = 10000\end{array}\right\} \text{ et } S = Q_4$$

3.4. *Exercices corrigés.*

Exercice 1 : *Questions pour un champion*.

On désire réaliser le système de commande pour un jeu télévisé où les trois candidats A, B, et C doivent appuyer le premier pour répondre à la question de l'animateur. Un indicateur lumineux du candidat en "question" s'allume.

Le présentateur a sa disposition un bouton de remise à zéro afin de pouvoir passer à la question suivante.

1. Déterminer le nombre d'entrées - sorties du système, la forme du diagramme d'état, puis le nombre d'états minimum pour le réaliser.

2. Etablir la table des transitions et réaliser cette machine d'état à l'aide de bascules D.

3. Le présentateur a une crampe. On supprime le bouton de remise à zéro, refaire la synthèse d'un dispositif adapté en imaginant une solution facile.

Remarque : Etant donné sa table de vérité, l'utilisation de bascule D permet de simplifier les calculs. En seconde lecture, vous pouvez réaliser la synthèse avec des bascules JK ou autres.

Exercice 2 : *le diviseur de fréquence*.

On souhaite réaliser un diviseur de fréquence dont le nombre diviseur peut être choisi par l'utilisateur parmi deux valeurs : 2 et 4.

On utilisera une machine de Moore pour réaliser notre circuit.

1. Proposer un diagramme d'état d'une machine de Mealy pour ce problème, est-il avantageux d'opter pour ce type de structure?

2. A présent, on veut avoir simultanément sur deux sorties différentes, les deux diviseurs de fréquence.

Exercice 3 : *Synthèse d'un compteur modulo 6.*

Nous désirons réaliser un compteur binaire synchrone modulo 6 dont l'entrée A permet de sélectionner sens de comptage.

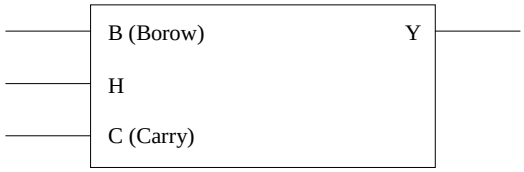
Lorsque l'entrée A prend la valeur '1' le compteur évolue dans le sens croissant (il compte).

Lorsque l'entrée A prend la valeur '0' le compteur évolue dans le sens décroissant (il décompte).

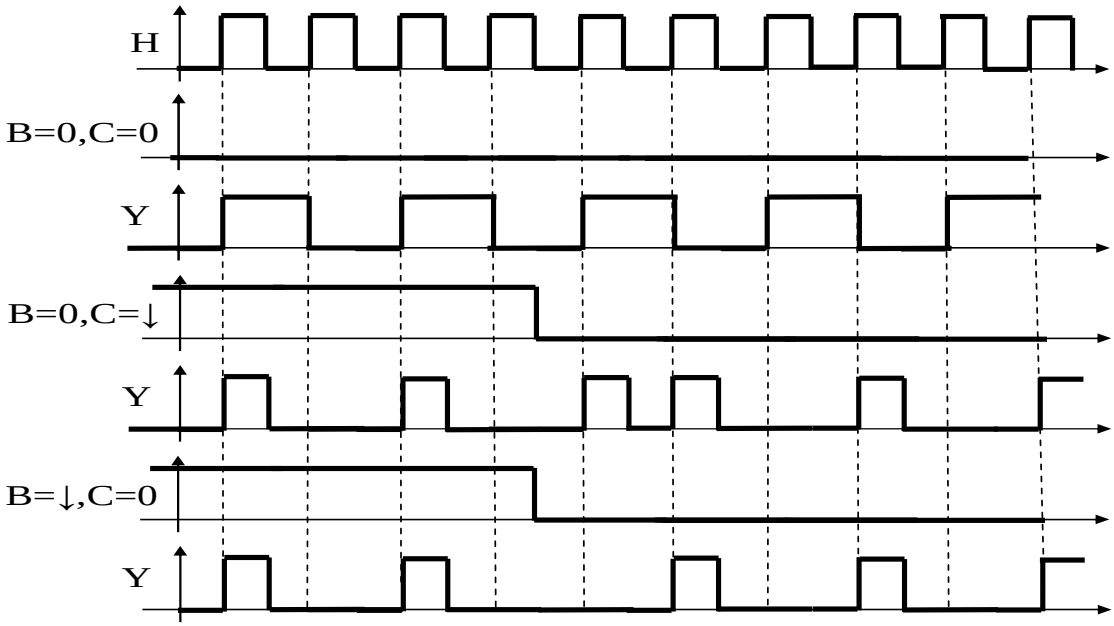
1. Faire la synthèse de ce compteur à l'aide de bascules D.
2. Dans la mesure où nous devions intégrer une remise à zéro (RAZ). Expliquer la différence entre une RAZ synchrone et asynchrone, proposer alors une solution pour les deux cas.

Exercice 4 : *Synthèse d'un diviseur par deux avec ajout et suppression d'impulsion.*

Voici sur la figure suivante, le schéma synthétique du circuit synchrone que l'on veut réaliser, le principe est le suivant :

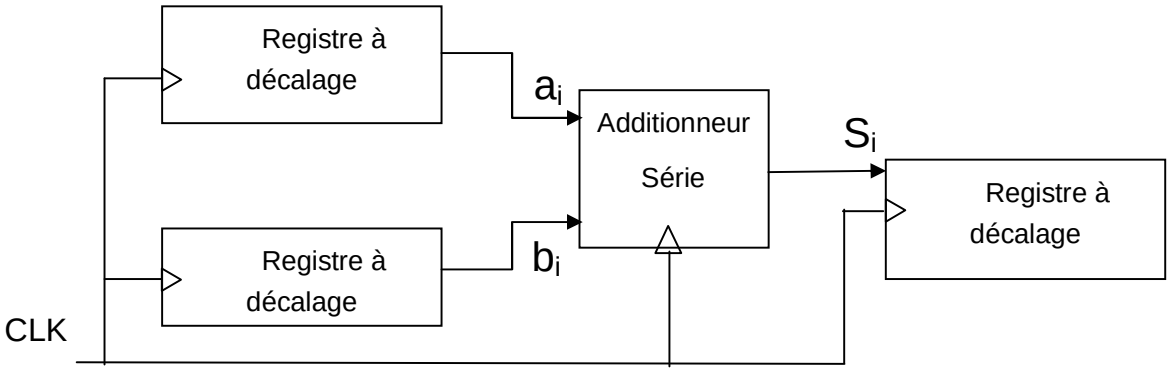


- Le cas (B=0 et C=1) ne se produit pas.
- Pour (B=0 et C=0), le circuit délivre en sortie Y une impulsion toutes les deux impulsions d'horloge. Le circuit fonction alors en diviseur par 2.
- Pour (B=0 et C=1) ou (B=1 et C=0), le fonctionnement est le même que dans le cas précédent.
- Pour (B=0 et C=1), une impulsion est ajoutée et la suite des impulsions est décalée d'une période.
- Pour (B=0 et C=1), une impulsion est retirée et la suite des impulsions est décalée d'une période.



Exercice 5 : *Synthèse d'un additionneur série*

Soit A = a_{n-1} a_{n-2} …a_0 et B = b_{n-1} b_{n-2} …b_0 deux nombres non signés dont on veut calculer la somme S = S_{n-1} S_{n-2} …S_0. On souhaite réaliser **une machine de Moore** qui effectuera cette opération bit après bit à chaque impulsion d'horloge en commençant par les bits de poids faibles a_0 et b_0. Lors de l'addition des bits a_i et b_i le système doit prendre en compte la retenue c_{i-1} engendrée, éventuellement, à l'étape précédente. Un exemple de schéma d'ensemble du système est donné sur la figure suivante.



1. Selon les valeurs de la sortie S (0 ou 1) et de la retenue c_{i-1} à prendre en compte (0 ou 1), quatre états sont nécessaires pour décrire le fonctionnement de l'additionneur. Par convention on nomme ces états de la façon suivante :

- L_0 correspond à S =0 et c_{i-1} = 0.
- L_1 correspond à S =1 et c_{i-1} = 0.
- H_0 correspond à S =0 et c_{i-1} = 1.
- H_1 correspond à S =1 et c_{i-1} = 1.

Donner le diagramme d'états décrivant le fonctionnement de l'additionneur série. En déduire la tables des transitions et la table des sorties.

2. Deux variables, au moins, sont nécessaires pour coder les quatre états. On adopte le codage suivant :

- $L_0 \rightarrow q_2q_1 = 00$
- $L_1 \rightarrow q_2q_1 = 01$
- $H_0 \rightarrow q_2q_1 = 11$

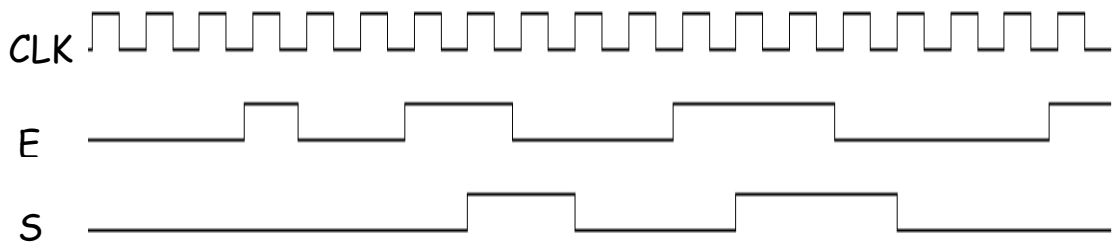
- $H_1 \rightarrow q_2q_1 = 10$

En déduire la table des transitions.

3. On souhaite réaliser le montage avec deux bascules D synchrones. Donner les expressions des entrées D_i de chacune des bascules ainsi que de la sortie S.

Exercice 6 : *Synthèse de machine d'état et états parasites*

On souhaite synthétiser un système séquentiel synchrone de « type Moore » dont la sortie S recopie l'entrée E si celle-ci est la même sur deux front actifs d'horloge successifs. Le fonctionnement est décrit sur le chronogramme suivant :



Chronogramme de principe

1. Montrer qu'il est possible de décrire ce système avec quatre états. Donner le diagramme des transitions.

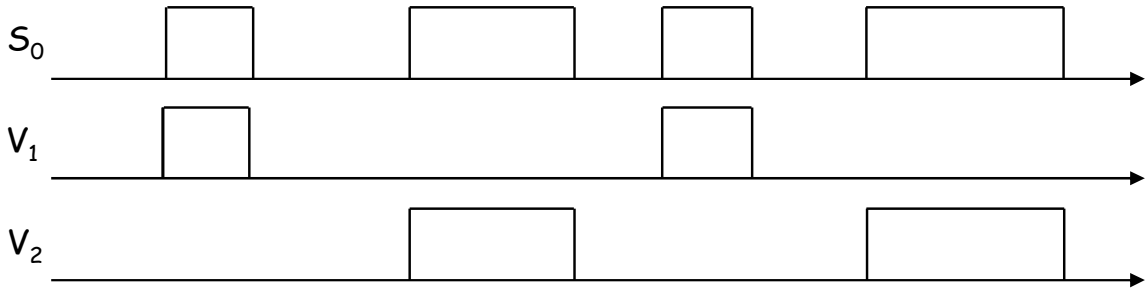
2. Proposer un codage pour ces quatre états et en déduire la table des transitions et la table de la sortie S. Pour notre exercice, est-il possible selon le codage choisi de simplifier le montage

3. On souhaite réaliser le montage avec des bascules D. Donner les expressions logiques des entrées D_i et de la sortie S.

Exercice 7 : *Synthèse d'un système séquentiel de contrôle de température*

La chaleur dégagée part un processeur dépend directement de son utilisation. Dans cet exercice on considère un supercalculateur dont les processeurs sont refroidis, entre autres, par deux ventilateurs V_1 et V_2. On se propose d'étudier le circuit électronique assurant fonctionnement de ces deux ventilateurs. Deux solutions, légèrement différentes, seront considérées.

A. Un détecteur de température S_0 indique si la température T du processeur dépasse la température seuil T_{S0} fixée par le constructeur ($S_0 = 1$ si $T > T_{S0}$). Dans ce cas l'un des ventilateurs V_1 ou V_2 doit être activé ($V_i = 1$ avec i =1 ou 2) jusqu'à ce que la température T redevienne inférieure à T_{S0} ($S_0 = 0$ si $T < T_{S0}$). Pour assurer la sécurité du système et éviter les pannes, les ventilateurs sont activés en alternance comme indiqué sur la figure suivante. Le circuit de contrôle comporte donc une entrée S_0 qui reçoit les informations du détecteur, et deux sorties, V_1 et V_2 contrôlant les ventilateurs.



Donner le diagramme d'états d'une machine de Moore synchrone décrivant ce système (4 états sont nécessaires). On notera **E_0** l'état initial dans lequel les deux ventilateurs sont éteints ($V_1 = V_2 = 0$). Les trois autres états seront notés **E_1**, **E_2** et **E_3**. Les ventilateurs V_1 etV_2 seront respectivement actifs dans les états **E_1** et **E_3**

A 1) On utilise le codage suivant pour les différents états :

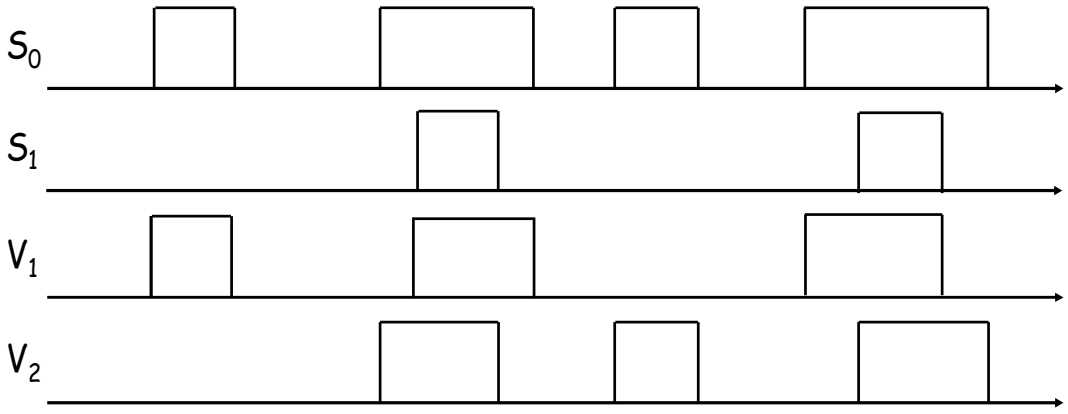
E_0 $\rightarrow$ $Q_2Q_1 = 0\ 0$	**E_2** $\rightarrow$ $Q_2Q_1 = 1\ 1$
E_1 $\rightarrow$ $Q_2Q_1 = 0\ 1$	**E_3** $\rightarrow$ $Q_2Q_1 = 1\ 0$

Compléter la table de transitions reliant les valeurs futures Q_2 et Q_1 aux valeurs

présentes q_2 et q_1. Donner la table des sorties V_1 et V_2.

A 2) On souhaite réaliser le montage avec des deux bascules D. Donner les expressions des entrées D_i (i = 0 ou 1) de chacune des bascules, ainsi que les expressions des sorties V1 et V2.

B. Dans certaines conditions, il peut arriver qu'un seul ventilateur ne soit pas suffisant pour dissiper la chaleur dégagée par le processeur. On étudie donc une autre configuration dans laquelle les deux ventilateurs vont pouvoir fonctionner simultanément. Un second détecteur de température S_1 est placé sur le processeur. Il se déclenche si la température T du processeur dépasse un nouveau seuil T_{S1} plus élevé que T_{S0} ($S_1 = 1$ si $T > T_{S1} > T_{S0}$). Dans ces conditions les deux ventilateurs fonctionnent en même temps jusqu'à ce que la température T redevienne inférieure à T_{S1}. Le nouveau circuit comporte donc deux entrées, S0 et S1, et deux sorties, V_1 et V_2. Son fonctionnement est décrit sur la figure suivante. Comme précédemment on souhaite utiliser les deux ventilateurs de façon similaire, par conséquent ils doivent fonctionner en alternance dans les cas ou un seul d'entre eux doit être utilisé ($T_{S1} > T > T_{S0}$).



B 1) Donner le diagramme d'états d'une machine de Moore synchrone décrivant ce système. Pour cela, le diagramme de la question **A1)** devra être complété, d'une part en tenant compte de l'entrée S1 dans les transitions déjà présentes et, d'autre

part, en ajoutant deux nouveaux états, $\mathbf{E_4}$ et $\mathbf{E_5}$, dans lesquels les deux ventilateurs sont en fonctionnement ($V_1 = V_2 = 1$).

B 2) Donner la table de transitions reliant les états futurs aux entrées et aux états présents. **On identifiera la ou les combinaisons de S_1 et S_0 physiquement impossibles. On considèrera, par ailleurs, que la température ne varie jamais rapidement si bien que le cahier des charges n'impose rien sur ces situations.**

B 3) Six états étant nécessaires au fonctionnement du circuit, le codage doit se faire sur 3 bits $Q_3Q_2Q_1$. On utilisera le codage suivant :

E0 $\rightarrow$ $Q_3Q_2Q_1 = 0\ 0\ 0$	**E3** $\rightarrow$ $Q_3Q_2Q_1 = 0\ 1\ 0$
E1 $\rightarrow$ $Q_3Q_2Q_1 = 0\ 0\ 1$	**E4** $\rightarrow$ $Q_3Q_2Q_1 = 0\ 1\ 1$
E2 $\rightarrow$ $Q_3Q_2Q_1 = 1\ 0\ 0$	**E5** $\rightarrow$ $Q_3Q_2Q_1 = 1\ 1\ 1$

En déduire la table des transitions et préciser la table des sorties V_1 et V_2.

B 4) On souhaite réaliser le montage avec deux bascules D. A partir du diagramme de transition de la question **B1)**, donner les expressions des entrées D_i (i = 0 ou 1) de chacune des bascules.

B 5) Quelles sont les expressions des sorties V_1 et V_2 ? Quel est l'intérêt du codage utilisé ?

B 6) Le codage se faisant sur 3 bits, il y a deux états, $\mathbf{E_6}$ et $\mathbf{E_7}$, non utilisés dans ce système.

B6a) Quels sont les codes $Q_2Q_1Q_0$ de ces deux états.

B6b) Comment évoluent-ils? Reporter cette évolution sur le diagramme de transition.

B6c) Comment doit-on modifier la table de transitions de la question **B2)** pour s'assurer que les états $\mathbf{E_6}$ et $\mathbf{E_7}$ évoluent vers l'état $\mathbf{E_0}$ en un seul coup d'horloge quelque soit la valeur des entrées S_1 et S_0 ?

Exercice 8 : *Synthèse d'un système séquentiel de codage*

Les données transférées sur les ports USB d'un ordinateur doivent être dans un format particulier appelé « non retour à zéro inversé » (NRZI). On souhaite synthétiser un circuit permettant de convertir une séquence de 0 et 1 dans le format NRZI. Les caractéristiques du circuit sont les suivantes :

- Le circuit dispose d'une entrée **E**, qui reçoit le message initial, et d'une sortie **Z**, sur laquelle on retrouve la traduction du message initial en format NRZI.
- Si un 0 est présent sur l'entrée, la sortie **Z** commute immédiatement de 1 à 0, ou de 0 à 1, selon la dernière valeur présente sur la sortie.
- Lorsque qu'un 1 est présent sur l'entrée, la sortie **Z** ne change pas. Elle reste à 0 ou 1 selon la dernière valeur présente sur l'entrée.

Ce fonctionnement est illustré sur la séquence suivante dans laquelle nous avons supposé que la valeur initiale sur la sortie **Z** est 1.

Message sur l'entrée **E**	Début : 10001110011010
Traduction NRZI sur la sortie **Z**	Début : 10100001000101

1. Proposer le diagramme de transition d'une machine de Mealy décrivant le fonctionnement de ce système.
2. Déduire de la question précédente la table de transition des états (état présent - état futur) ainsi que la table des sorties.
3. Proposer un codage. En déduire la table des transitions donnant les valeurs futures Q_i^+ en fonction des valeurs présentes Q_i. Donner l'expression logique de la sortie **Z.**
4. Proposer un schéma pour la réalisation de ce circuit en utilisant une bascule D et le minimum de portes logiques.

Remarque : Cet exercice est un bon test pour une autoévaluation. Il peut être terminé et rédiger en moins de 20 minutes.

Remarque : Par curiosité le lecteur pourra réaliser la synthèse avec une structure de type Moore afin de pouvoir faire la comparaison.

Exercice 9 : *Synthèse d'un compteur-décompteur*

On souhaite synthétiser un compteur en code Gray synchrone ayant pour entrées A et B (en plus de l'entrée d'horloge H) et pour sorties Q_1 (MSB) et Q_0 (LSB). Le fonctionnement du compteur est le suivant :

- Si AB = 10 le compteur compte, suivant la séquence $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 0$ … les impulsions arrivant sur CK.
- Si AB = 11 il les décompte $0 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 0$ …
- Si AB = 00 le compteur est remis dans l'état 0
- Si AB = 01 le compteur reste bloqué.

1. Expliquer pourquoi les machines de Moore sont plus appropriées que les machines de Mealy pour réaliser ce type de système. Donner le diagramme de transition décrivant le fonctionnement du compteur.

2. En déduire la table de transition reliant les valeurs futures aux valeurs présentes.

3. On souhaite réaliser le compteur avec des bascules D. Quelles sont les équations de commandes des bascules et des sorties?

Problème : *Synthèse du Code Manchester différentiel*.

Nous allons réaliser un décodeur utilisé dans certain protocole de communications série entre ordinateur : le codage *Manchester différentiel.* En codage Manchester différentiel, chaque intervalle de temps élémentaire, pendant lequel nous voulons transmettre un bit « 0 » ou « 1 », appelé « temps bit », présente une structure particulière :

- au début du temps bit :

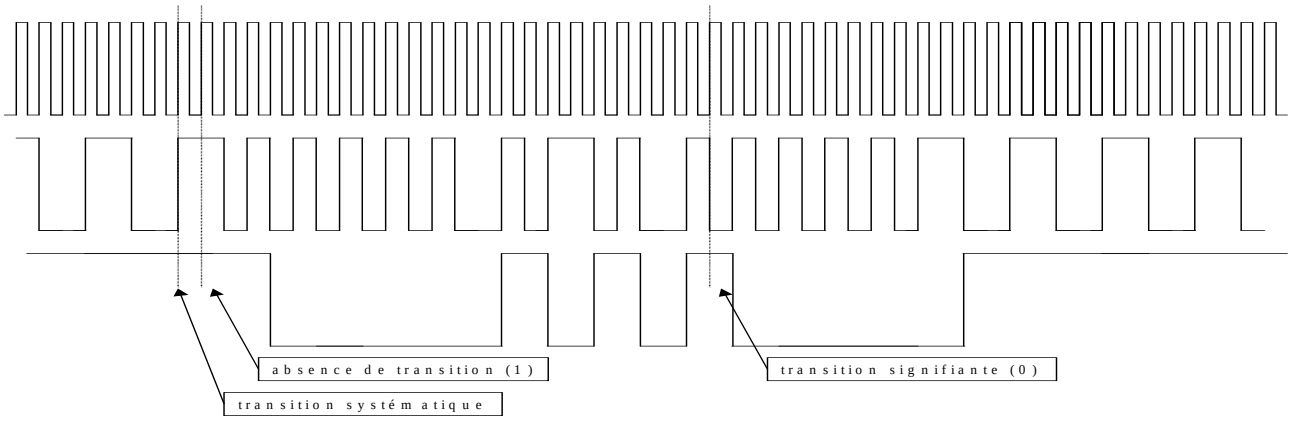
  1) le signal binaire « 0 » est représente par une transition

  2) le signal binaire « 1 » est représenté par l'absence de transition.

- Exactement au milieu du temps bit, il y a toujours une transition.

1. Sur le chronogramme suivant, la première courbe correspond à l'horloge du codeur et la seconde représente le signal de sortie du codeur, tandis que la dernière donne le signal décodé comme nous le désirons. Le signal *Man* est codé selon l'algorithme précédent et *bin* est le signal binaire décodé associé.

Comment peut-on reconnaître un temps bit ? D'autre part, si le décodeur manque une transition pour une raison quelconque comment peut-il différencier dans la séquence la transition systématique de la transition signifiante ?



2. Interpréter la sortie *bin* du décodeur en fonction du signal *Man,* c'est-à-dire identifier les temps bits et les valeurs binaires associées.

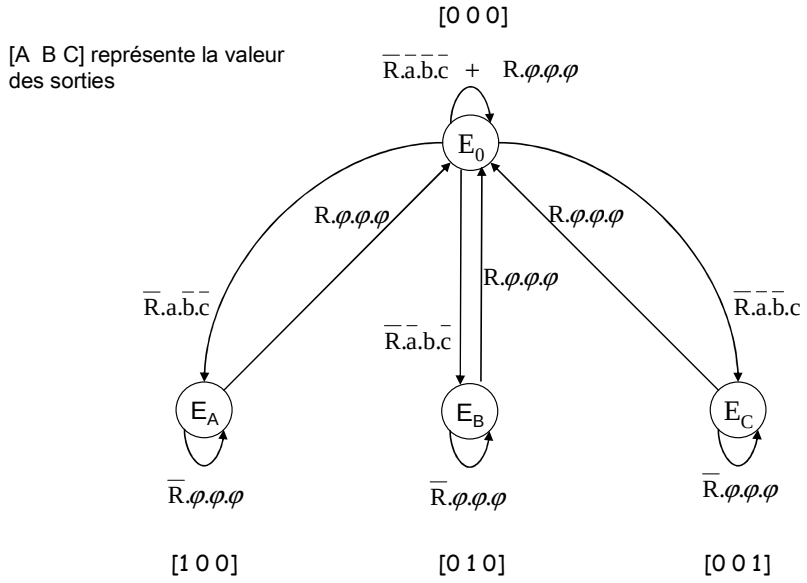
3. Avez-vous remarqué que la sortie du décodeur est décalée d'une période d'horloge *hor* ? Réaliser le décodeur décrit en synthèse de *Mealy* puis en synthèse de *Moore*. Comparer les deux montages.

Autocorrection

Exercice 1 : *Questions pour un champion*.

1. Dans cet exercice il faut bien remarquer la symétrie du système : il n'y a pas de différence entre les candidats. Il faut nécessairement un état d'attente stable. Le candidat le plus rapide permet de sortir du débouclage en prenant la main, son indicateur lumineux est actif jusqu'à la remise à zéro de notre animateur ce qui est traduit par un rebouclage sur l'état du candidat correspondant.

Après analyse du cahier des charges nous avons donc 4 entrées : la remise à zéro du présentateurs et les 3 boutons poussoirs des candidats a, b et c. Les sorties sont au nombre de 3 : A, B et C dont l'état haut, actif, déclenche sonnerie et signal lumineux.
4 états seront nécessaires.



2. Avant d'écrire les équations pour une synthèse à l'aide de bascule D ou JK. Il nous faut établir les tables de transitions et définir un codage pour les états utiles.

Selon l'usage, on place les états sur les lignes et les combinaisons d'entrées sur les colonnes. Comme nous avons 4 entrées nous devrions en toute rigueur dessiner un tableau avec 16 colonnes.

Combinaisons des **Entrées** **R,a,b et c**

état présent \ R a b c	**0 0 0 0**	**0 0 0 1**	**0 0 1 1**	**0 0 1 0**	**0 1 1 0**	**0 1 1 1**	**0 1 0 1**	**0 1 0 0**	**1 φ φ φ**
E_0	**E_0**	**E_C**	**E_0**	**E_B**	**E_0**	**E_0**	**E_0**	**E_A**	**E_0**
E_A	**E_C**	**E_A**	**E_A**	**E_A**	**E_A**	**E_A**	**E_A**	**E_A**	**E_0**
E_B	**E_B**	**E_B**	**E_B**	**E_B**	**E_B**	**E_B**	**E_B**	**E_B**	**E_0**
E_C	**E_c**	**E_C**	**E_C**	**E_C**	**E_C**	**E_C**	**E_C**	**E_C**	**E_0**

Etats futurs

Si nous voulons utiliser le moins possible de bascules, 2 bascules permettent de coder 4 états. Choisissons le codage suivant :

- $E_0 \rightarrow q_2q_1 = 00$
- $E_A \rightarrow q_2q_1 = 01$
- $E_B \rightarrow q_2q_1 = 11$
- $E_C \rightarrow q_2q_1 = 10$

Où q_2 et q_1 sont les sorties des deux bascules. Dans ce cas il est nécessaire d'intercaler un bloc de logique combinatoire entre le registre de bascule et les sorties, afin d'interpréter le « nom » de l'état en valeur pour les sorties (voir figure 29 du chapitre, bloc G).

Dans notre cas :

$q_1\ q_2$ \	**A**	**B**	**C**
0 0	0	0	0
0 1	1	0	0
1 1	0	1	0
1 0	0	0	1

Les équations booléennes des sorties sont triviales : $A = \overline{q_1}.q_2$, $B = q_1.q_2$ *et* $C = q_1.\overline{q_2}$

Remarque : dans cet exercice on voit clairement qu'en acceptant d'utiliser 3 bascules (choix non optimal) le tableau précédent devient en choisissant un codage approprié (« one hot encoding », voir p95):

$q_1\ q_2\ q_3$ \	**A**	**B**	**C**
0 0 0	0	0	0
0 0 1	1	0	0
0 1 0	0	1	0
1 0 0	0	0	1

Et les équations booléennes des sorties sont encore plus simples : $A = q_3$, $B = q_2$ *et* $C = q_1$, on peut aussi s'attendre à une simplification des équations des commandes des bascules.

Revenons au codage limitant les ressources matérielles avec 2 bascules :

- $E_0 \rightarrow q_2q_1 = 00$
- $E_A \rightarrow q_2q_1 = 01$

- $E_B \rightarrow q_2q_1 = 11$
- $E_C \rightarrow q_2q_1 = 10$

R	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
a	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**φ**
état **b**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**φ**
présent **c**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**φ**
****									
$E_0 \rightarrow q_2q_1 = 00$	0 0	1 0	0 0	1 1	0 0	0 0	0 0	0 1	0 0
$E_A \rightarrow q_2q_1 = 01$	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 0
$E_B \rightarrow q_2q_1 = 11$	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	0 0
$E_C \rightarrow q_2q_1 = 10$	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0	0 0

Valeurs futures des sorties des bascules Q_2 Q_1

- Si on choisit les bascules D qui ont la caractéristique de recopier la valeur de l'entrée D en Q au front actif suivant, soit :

$$D_i = Q_i$$

$$D_2 = Q_2 = q_2 + \overline{R}.\overline{a}.\overline{b}.c.\overline{q_1} + \overline{R}.\overline{a}.b.\overline{c}.\overline{q_1} = q_2 + \overline{R}.\overline{a}.\overline{q_1}(\overline{b}.c + b.\overline{c}) = q_2 + \overline{R}.\overline{a}.\overline{q_1}(b \oplus .c)$$

$$D_1 = Q_1 = q_1 + \overline{R}.\overline{a}.\overline{b}.c.\overline{q_2} + \overline{R}.a.\overline{b}.\overline{c}.\overline{q_2} = q_2 + \overline{R}.\overline{c}.\overline{q_2}(\overline{a}.b + a.\overline{b}) = q_1 + \overline{R}.\overline{c}.\overline{q_2}(a \oplus .b)$$

Etant donné que l'on effectue les regroupements de Karnaugh sur les valeurs '1', la seconde partie du tableau où R='1' (remise à zéro) ne nous apporte aucun terme dans nos expressions (ceci est vrai si on code l'état d'attente '00', ce qui est très souvent le cas).

- Si on choisit les bascules JK qui ont une table de vérité plus complexe, il est préférable d'établir la table de vérité des J_iK_i

On rappelle la table de vérité de la bascule JK (table de vérité à gauche). La table de droite correspond à une autre manière de présenter cette table de vérité qui est sûrement plus facile à mémoriser :

Si l'ancienne valeur de la bascule est '0', on recopie au prochain front actif la valeur de J.
Si l'ancienne valeur de la bascule est '1', on recopie au prochain front actif la valeur de $\overline{K}$.

J K	Q^+
0 0	Q
0 1	0
1 1	$\overline{Q}$
1 0	1

Q	J K	Q^+
0	0 φ	0
0	1 φ	1
1	φ 0	1
1	φ 1	0

R	0	0	0	0	0	0	0	0	1
a	0	0	0	0	1	1	1	1	φ
b	0	0	1	1	1	1	0	0	φ
Etat **c**	0	1	1	0	0	1	1	0	φ
n									
q_2 q_1	Q_2 Q_1	Q_2 Q_1	Q_2 Q_1	Q_2 Q_1	Q_2 Q_1	Q_2 Q_1	Q_2 Q_1	Q_2 Q_1	Q_2 Q_1
00	0 0	1 0	0 0	1 1	0 0	0 0	0 0	0 1	0 0
01	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 0
11	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	0 0
10	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0	0 0
q_2 q_1	$J_2K_2J_1K_1$	$J_2K_2J_1K_1$	$J_2K_2J_1K_1$	$J_2K_2J_1K_1$	$J_2K_2J_1K_1$	$J_2K_2J_1K_1$	$J_2K_2J_1K_1$	$J_2K_2J_1K_1$	$J_2K_2J_1K_1$
00	0φ 0φ	1φ 0φ	0φ 0φ	φ1 φ1	0φ 0φ	0φ 0φ	0φ 0φ	0φ φ1	0φ 0φ
01	0φ φ0	0φ φ0	0φ φ0	0φ φ0	0φ φ0	0φ φ0	0φ φ0	0φ φ0	0φ φ1
11	φ0 φ0	φ0 φ0	φ0 φ0	φ0 φ0	φ0 φ0	φ0 φ0	φ0 φ0	φ0 φ0	φ1 φ1
10	φ0 0φ	φ0 0φ	φ0 0φ	φ0 0φ	φ0 0φ	φ0 0φ	φ0 0φ	φ0 0φ	φ1 0φ

Après cette étape, il faut extraire les colonnes correspondant à chaque entrée des bascules et comme la table est déjà sous forme de tableau de Karnaugh, on en tire l'expression :

Pour J_2 :

0	1	0	φ	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ
φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ

$$J_2 = \overline{q_2}.\overline{q_1}.\overline{R}.a.\overline{b}.c$$

Pour K_2 :

φ	φ	φ	1	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ
φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

$$K_2 = \overline{q_2}.\overline{q_1}.\overline{R}.a.\overline{b}.\overline{c} + R.q_2$$

Pour J_1 :

0	0	0	φ	0	0	0	φ	0	0	0	0	0	0	0	0
φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ
φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$$J_1 = 0$$

Pour K_1 :

φ	φ	φ	1	φ	φ	φ	1	φ	φ	φ	φ	φ	φ	φ	φ
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ	φ

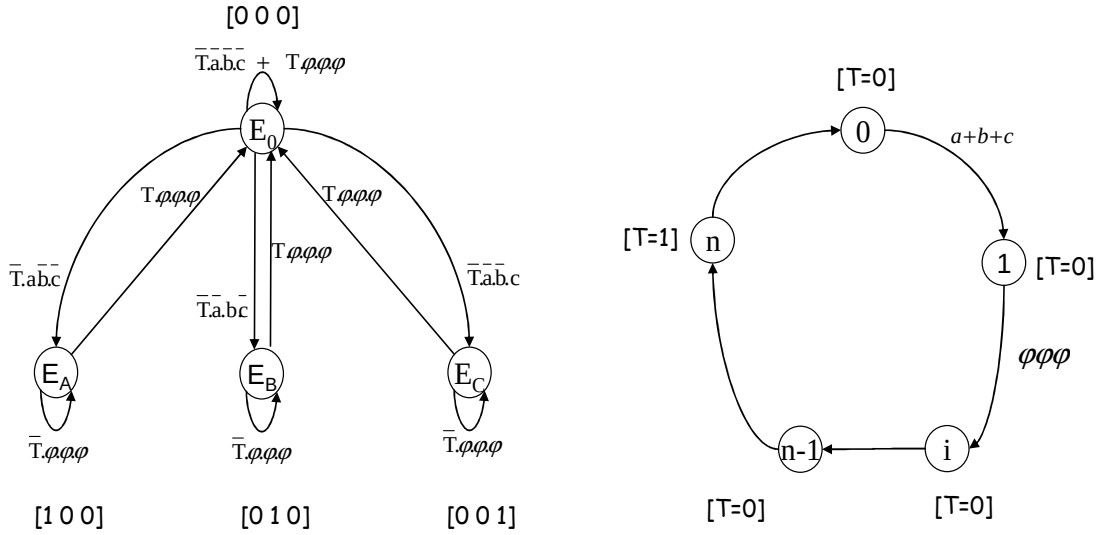
$$K_1 = \overline{q_2}.\overline{q_1}.\overline{R}.a.\overline{b}.\overline{c} + \overline{q_2}.\overline{q_1}.\overline{R}.a.\overline{b}.\overline{c} + R.q_1$$

Remarque : On notera qu'il y a deux possibilité d'obtenir la valeur

ceci correspond aux différentes fonctions des bascules soit la valeur mémorisée est utilisée mais si elle ne convient pas il suffira de réaliser un Set ou Reset. Dans les tables de Karnaugh, ces deux possibilités se traduiront par un φ avec des simplifications supplémentaires.

Remarque : la synthèse à l'aide de bascules JK de diffère pas beaucoup avec celle basée sur des bascules D. Il faudra juste établir les tables de vérité des J_iK_i, cette étape bien que longue ne mène pas à des calculs complexes étant le nombre de simplifications importants au sein des tables de Karnaugh.

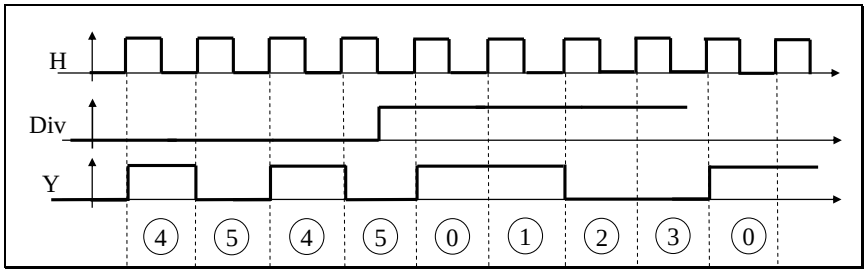
3. Si on considère le schéma de la question **1.**, les transition E_i où i=A,B et C assurent un fonctionnement cyclique Si le commentateur n'a pas de moyen d'action pour faire la remise à zéro, il est impératif de prévoir une remise à zéro 'automatique' sous la forme d'une temporisation.



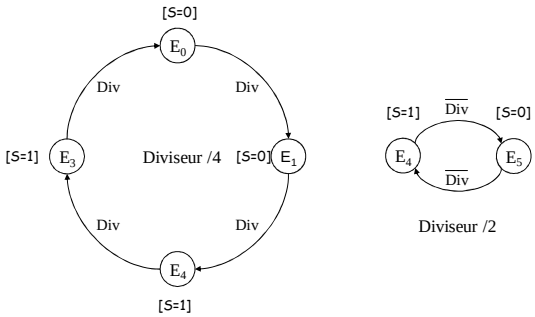
En fonction des périodes des horloge que l'on dispose, on détermine la taille n du compteur et lorsque le compteur à dénombrer le nombre de périodes prévues il délivre une impulsion sur une sortie 'temporisation' notée T.
Il faut bien voir que les deux cycles ne sont pas indépendants, le comptage ne débute que si l'uns des états E_i est atteints et le système restera bloqué en E_i tant que le comptage ne sera pas terminés. On peut alors d'états E_i et de sous-états de E_i.

Exercice 2 *: le diviseur de fréquence*.

1. Soit Div l'entrée permettant d'opter pour la valeur du diviseur, pour Div='0' l'horloge est divisée par 2 et pour Div='1' l'horloge est divisée par 4.

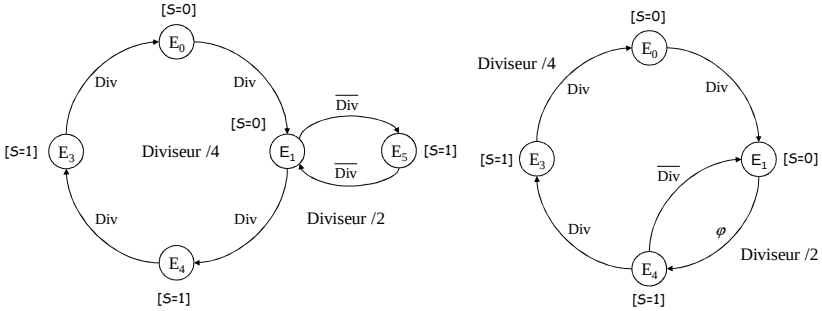


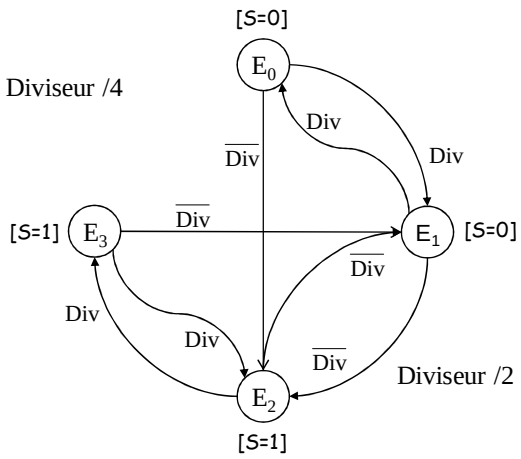
On dénombre 6 états différents sur le chronogramme.



Donc le diagramme présentera un cycle à 4 états et un cycle à 2 états. A chaque front actif de l'horloge, le système progressera sur l'un des deux cycles selon la valeur de l'entrée Div. Ce diagramme d'état est bien sur incomplet, nous avons rempli le cahier des charges uniquement pour les « régimes permanents ». Il faut donc traiter la façon dont le système passe d'un fonctionnement à un autre c'est-à-dire la manière dont sont couplés les deux cycles.

Voici plusieurs propositions :





La différence entre les trois diagrammes est bien le nombre de coups d'horloge nécessaires pour atteindre le nouveau fonctionnement et on se rend bien compte que dans les deux premiers diagrammes, cela dépend dans quel état se trouve le système.

Exemple pour le premier diagramme : pour aller vers le diviseur par 2, en partant de E2 il faut 3 coups d'horloge tandis qu'en partant de E1 il faut juste un coup. Autre dysfonctionnement possible, dans certains cas la sortie peut garder la même valeur pendant 3 périodes d'horloge ce qui n'est pas acceptable.

Pour cette raison seul le diagramme de la dernière figure est satisfaisant puisqu'il minimise le nombre d'états et quelque soit l'état dans lequel le système se trouve, il suffit d'un coup d'horloge pour aller vers le fonctionnement désiré.

Remarque : on cherchera toujours la manière la plus claire de représenter un diagramme de transitions et dans le cas précédent, on préférera sûrement une disposition linéaire.

Div **Etat pres **	**0**	**1**	**Sortie** **S**
E_0	**E_2**	**E_1**	0
E_1	**E_2**	**E_0**	0
E_2	**E_1**	**E_3**	1
E_3	**E_1**	**E_2**	1

Etats futurs

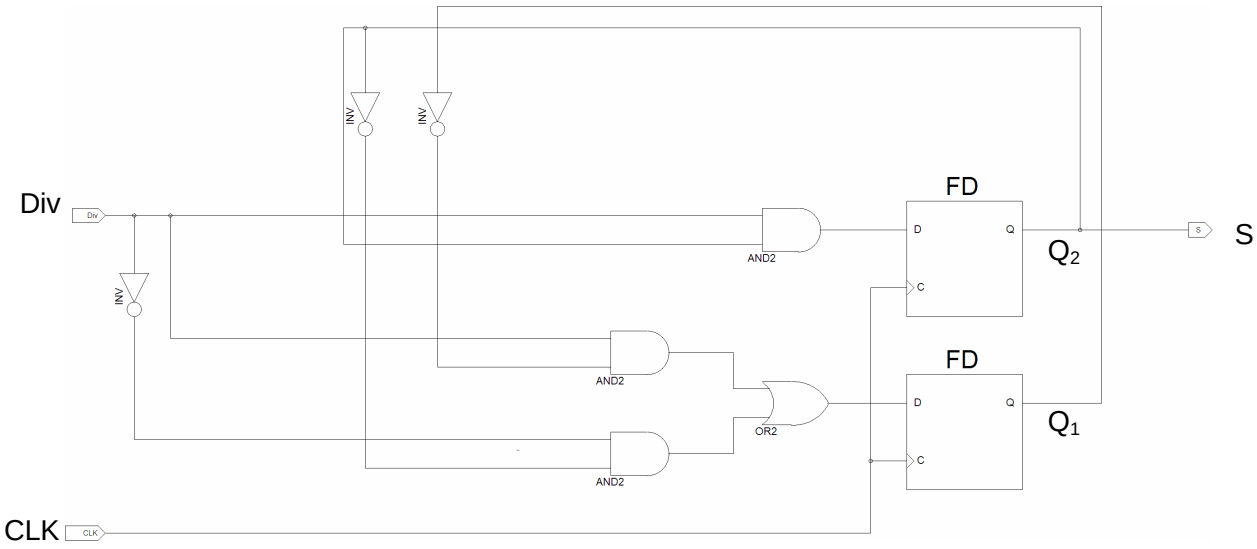
En prenant le codage habituel (qui n'est peut-être pas le meilleur):

Div $q_2\ q_1$ \	0	1	Sortie S
0 0	0 1	0 1	0
0 1	0 1	0 0	0
1 1	0 0	1 0	1
1 0	0 0	1 1	1

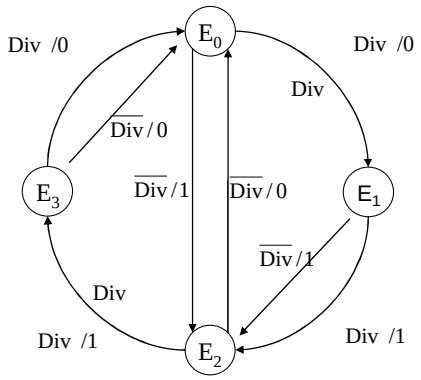
On a : $Q_2 = q_2.\mathrm{Div}$ et $Q_1 = \overline{q_2}.\overline{\mathrm{Div}} + \overline{q_1}.\mathrm{Div}$

Et $S = q_2$

En utilisant des bascules D, $D_i = Q_i$.



2. Si on accepte d'utiliser les entrées pour calculer les sorties alors en se basant sur un simple cycle à 4 on peut répondre à la question.



Etat pres \ Div	Sortie S : 0	Sortie S : 1
E0 (00)	**0**	**0**
E1 (01)	**1**	**0**
E2 (11)	**0**	**1**
E3 (10)	**1**	**1**

$$S = Q_2.Div + \overline{Div}.(Q_2 \oplus Q_1)$$

Si on abandonne la contrainte « synchrone à l'horloge », l'utilisation des entrées dans le calcul des sorties offre des possibilités supplémentaires qui permettent de réduire la complexité des expressions. En effet le simple compteur par 4 s'écrit comme :

Pour Div='1' :

$q_2\ q_1$ \	$Q_2\ Q_1$
0 0	0 1
0 1	1 1
1 1	1 0
1 0	0 0

$$Q_2 = q_1$$
$$Q_1 = \overline{q_2}$$

Pour Div='0' :

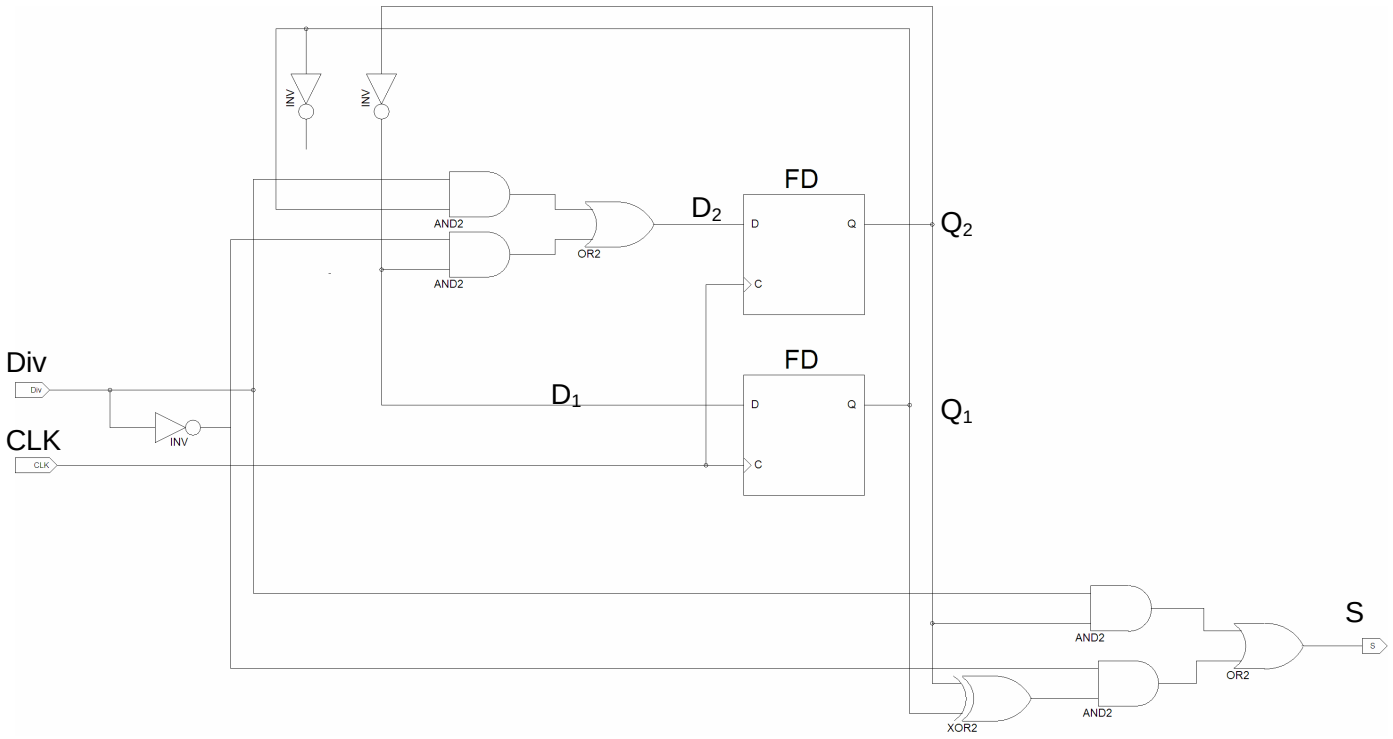
$q_2\ q_1$ \	$Q_2\ Q_1$
0 0	1 1
0 1	1 1
1 1	0 0
1 0	0 0

$$Q_2 = \overline{q_2}$$
$$Q_1 = \overline{q_2}$$

Soit $Q_2 = div.q_1 + \overline{div}.\overline{q_2}$ et $Q_1 = \overline{q_2}$

L'inconvénient de cette structure est une perte de la synchronisation, cela n'est pas forcement préjudiciable pour l'application en question mais la plupart du temps notre diviseur de fréquence fait partie d'un système plus complexe, comme un maillon d'une chaîne si les sorties de chaque élément ont des transitions entre les tops d'horloge, il sera difficile d'éviter des effets indésirables. A contrario si chaque maillon réagit dans un temps assez court après l'horloge, il sera possible de supprimer les aléas. D'autre part dans cet exercice, la solution de type Mealy ne permet pas de diminuer le nombre d'états nécessaires et la complexité des équations est comparable dans les deux architectures. Il

sera donc préférable de garde une structure de ‘Moore’.

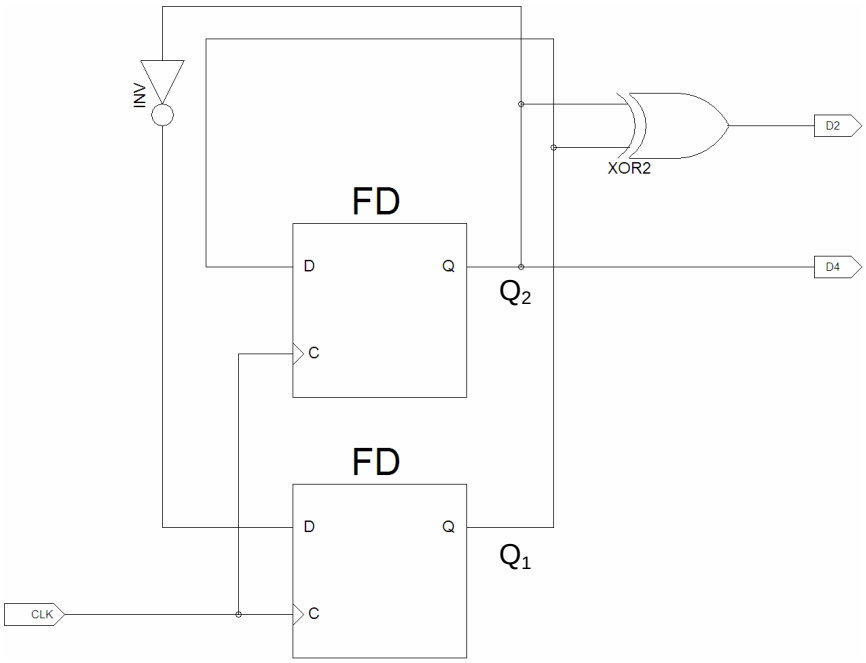


3. Dans la mesure où il n’y a plus qu’un seul mode de fonctionnement, on supprime la difficulté principale. On parvient à nos fins uniquement sur la logique intercalée entre les bascules et les sorties.

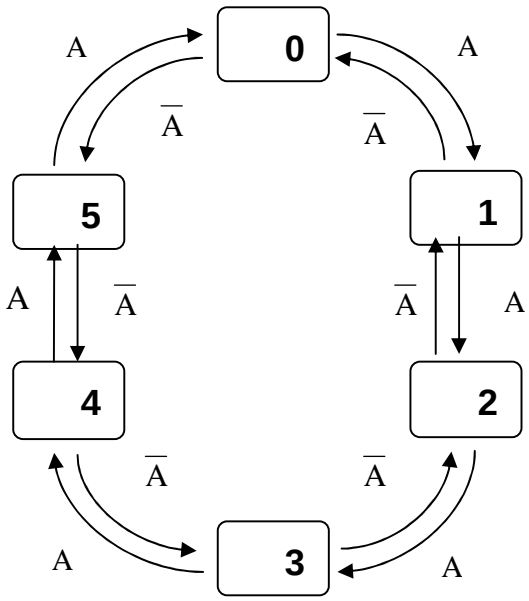
		Sortie D2 et D4	
**Etat pres **	**Q_2 Q_1**	**D2**	**D4**
E_0 (00)	0 1	**0**	**0**
E_1 (01)	1 1	**1**	**0**
E_2 (11)	1 0	**0**	**1**
E_3 (10)	0 0	**1**	**1**

$$Q_2 = q_1$$
$$Q_1 = \overline{q_2}$$

$$D2 = (Q_2 \oplus Q_1)$$
$$D4 = Q_2$$


INV
XOR2
D2
FD
D
Q
D4
Q2
C
FD
D
Q
Q1
CLK
C

Exercice 3 : *Synthèse d'un compteur modulo 6.*



1. Le diagramme des transitions comporte 6 états E_i et qu'une entrée A et la valeur de la sortie est codée en binaire sur 3 bits. Comme le cahier des charges ne stipule pas l'utilisation d'un type de bascule en particulier, nous utiliserons des bascules D qui facilite la synthèse et les équations.

2.

	A		**A**	
Etat présent	**0**	**1**	**0**	**1**
0	**5**	**1**	101	001
1	**0**	**2**	000	010
2	**1**	**3**	001	011
3	**2**	**4**	010	100
4	**3**	**5**	011	101
5	**4**	**0**	100	000

6	**φ**	**φ**	φφφ	φφφ
7	**φ**	**φ**	φφφ	φφφ
	Etat futur		**Sortie Q_2 Q_1 Q_0**	

Etant donné le choix du type de bascule, on peut écrire les tables de vérité des entrées des trois bascules.

A / **Q_1 Q_0 \ Q_2**	**0 0**	**0 1**	**1 1**	**1 0**	**0 0**	**0 1**	**1 1**	**1 0**	**0 0**	**0 1**	**1 1**	**1 0**
0 0	1	0	1	0	0	1	0	0	1	1	1	1
0 1	0	1	0	0	0	0	0	1	0	0	0	0
1 1	0	**φ / 1**	**φ / 1**	1	1	**φ / 1**	φ	0	0	φ	φ	0
1 0	0	φ	**φ / 1**	0	0	**φ / 1**	**φ / 1**	1	1	**φ / 1**	**φ / 1**	1

$$D_2 = \overline{Q_0}.\overline{Q_1}.\overline{Q_2}.\overline{A} + Q_0.Q_2.\overline{A} + Q_0.Q_1.A + \overline{Q_0}.Q_2.A$$

$$D_1 = \overline{Q_0}.Q_2.\overline{A} + Q_0.Q_1.\overline{A} + \overline{Q_0}.Q_1.A + Q_0.\overline{Q_1}.\overline{Q_2}.A$$

$$D_0 = \overline{Q_0}$$

On remarque que si on prend un arrangement des variables différents pour les lignes et colonnes de la table de vérité, les choix pour les états indéterminés φ peuvent être différents. Les équations sont donc légèrement changées.

A $Q_2 Q_1 \backslash Q_0$	0 0	0 1	1 1	1 0	0 0	0 1	1 1	1 0	0 0	0 1	1 1	1 0
0 0	1	1	0	0	0	0	1	0	1	0	0	1
0 1	0	0	1	0	0	1	0	1	1	0	0	1
1 1	φ	φ **/ 1**	φ **/ 1**	φ**/ 1**	φ	φ **/ 1**	φ	φ **/ 1**	φ **/ 1**	φ	φ	φ **/ 1**
1 0	0	1	0	1	1	0	0	0	1	0	0	1

$$D_2 = \overline{Q_1}.\overline{Q_2}.\overline{A} + Q_0.Q_2.\overline{A} + Q_0.Q_1.A + \overline{Q_0}.Q_2.A$$

$$D_1 = \overline{Q_0}.Q_2.\overline{A} + Q_0.Q_1.\overline{A} + \overline{Q_0}.Q_1.A + Q_0.\overline{Q_1}.\overline{Q_2}.A$$

$$D_0 = \overline{Q_0}$$

Donc selon le choix des états indéterminés φ, on vérifie vers quels états futurs, les états 6 et 7 évoluent

6	001	011	**1**	**7**
7	110	100	**6**	**4**

Sortie $Q_2 Q_1 Q_0$ Etat futur

Il n'y a donc ni de stabilité ni d'état piège pour ces états parasites 6 et 7. On pourrait éventuellement fixer les états futurs pour ces états non compris dans le cycle de comptage et dans ce cas on préfèrera que ces états évoluent tous vers le même état (l'état '00' étant le meilleur choix). Etant donné que le cahier des charges ne précise rien pour les états 'hors comptage' ou états parasites autant conserver l'avantage des transitions indéterminées φφφ lors des simplifications dans les tables de Karnaugh.

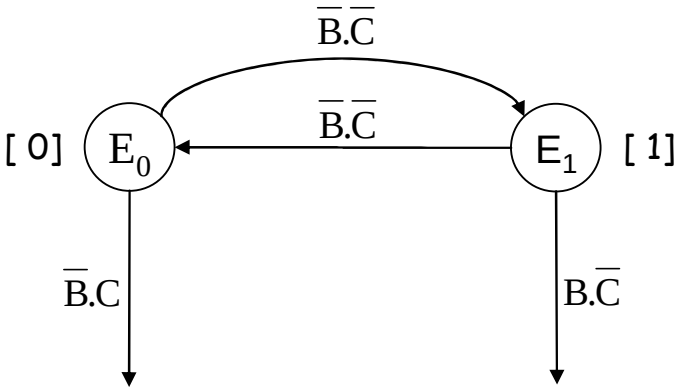
3. RAZ Synchrone ou Asynchrone.

Une remise à zéro synchrone doit, en agissant sur la valeur des variables de commandes des bascules, imposer une transition vers l'état '00'. Ce retour à l'état '00' ne pourra s'opérer qu'après un front actif d'horloge suivant le changement sur les entrées. Dans notre cas, nous avons qu'une seule entrée. Il faut donc ajouter une seconde entrée reset (active à '0') qui remet les D_i à '0', soit D'_i=Reset.D_i. Pour la remise à zéro asynchrone, on se contente d'agir directement sur les sorties du compteur Q'_i=Reset.Q_i.

Exercice 4 : *Synthèse d'un diviseur par deux avec ajout et suppression d'impulsion.*

La difficulté de cet exercice par rapport aux précédents est que le système n'est plus sensible aux valeurs des entrées mais aux transitions sur celles-ci. Il est donc très important de bien interpréter l'énoncé avant de commence le diagramme, la donnée du chronogramme dans le cahier des charges ne laisse aucune souplesse au concepteur : après d'une transition descendante sur B ou sur C, le système doit intercaler sur la sortie une période d'horloge avec ou sans impulsion entre des trains d'impulsion de fréquence deux fois plus petite (correspondant au mode normal lorsque BC= 00).

Dans un premier temps, on tachera de décomposer le problème en se focalisant un sur le fonctionnement normal : diviseur de fréquence entre les états E_0 et E_1.



Le système sort de cette boucle lorsque un front montant sur B ou C est mémorisé en vue de l'ajout ou de la suppression d'impulsion. Seulement il doit bien être clair qu'il faut attendre le front descendant pour réaliser la fonction en question et en attendant la division de fréquence sur la sortie S ne doit pas être interrompue. Il faut donc 2 cycles 'annexes', soit 2*2 états en plus du fonctionnement normal.

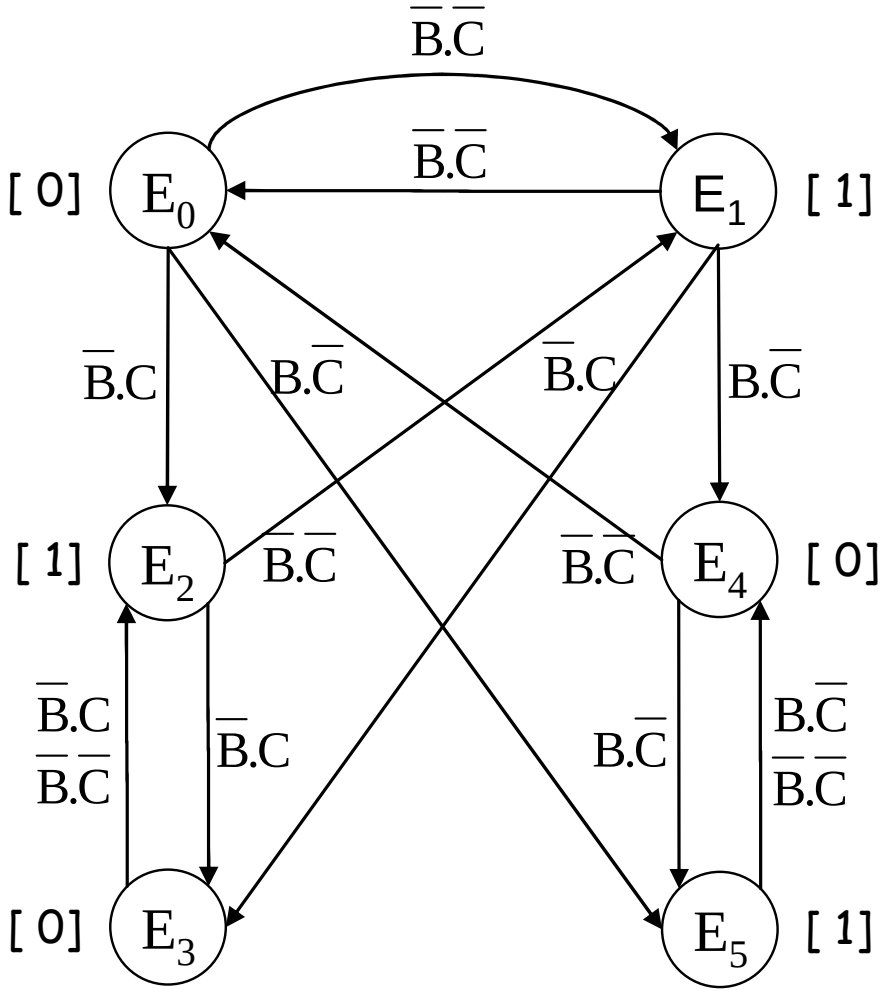
Donc nous pouvons décomposer le diagramme des états final en 3 cycles de 2 états ayant tous le même but : division par 2 de l'horloge :

- premier cycle correspondant au fonctionnement normal BC='00',

- un cycle permettant de poursuivre la division tout en mémorisant B='1',

- et un cycle permettant de poursuivre la division tout en mémorisant C='1'.

Un front descendant sur l'entrée en question permet de retrouver le cycle normal.



L'analyse du cahier des charges et du diagramme proposé montre que certains cas ne puissent se présenter. Par exemple les états 2 et 3 sont accessibles si la valeur '1' est détectée sur l'entrée C donc aucune transitions n'est prévue lorsque B='1' à partir de ces états et d'autre part, la combinaison BC='11' ne peut se produire. Pour ces raisons, la table des transitions présente 10 transitions indéterminées φ.

B C Etat	0 0	0 1	1 1	1 0
E_0	**E_1**	**E_2**	φ	**E_5**
E_1	**E_0**	**E_3**	φ	**E_4**
E_2	**E_1**	**E_3**	φ	φ
E_3	**E_2**	**E_2**	φ	φ
E_4	**E_0**	φ	φ	**E_5**
E_5	**E_4**	φ	φ	**E_4**

Le codage : bien que le diagramme soit structuré et symétrique, la valeur des sorties ne présentent pas de symétrie. Nous avons besoin de trois bascules pour coder les 6 états et on choisit $Q_o = S$.

- $E_0 \rightarrow q_2q_1q_0 = 000$
- $E_1 \rightarrow q_2q_1q_0 = 001$
- $E_2 \rightarrow q_2q_1q_0 = 111$
- $E_3 \rightarrow q_2q_1q_0 = 110$
- $E_4 \rightarrow q_2q_1q_0 = 100$
- $E_5 \rightarrow q_2q_1q_0 = 101$

Les états non utilisés sont :

- $E_6 \rightarrow q_2q_1q_0 = 010$
- $E_7 \rightarrow q_2q_1q_0 = 011$

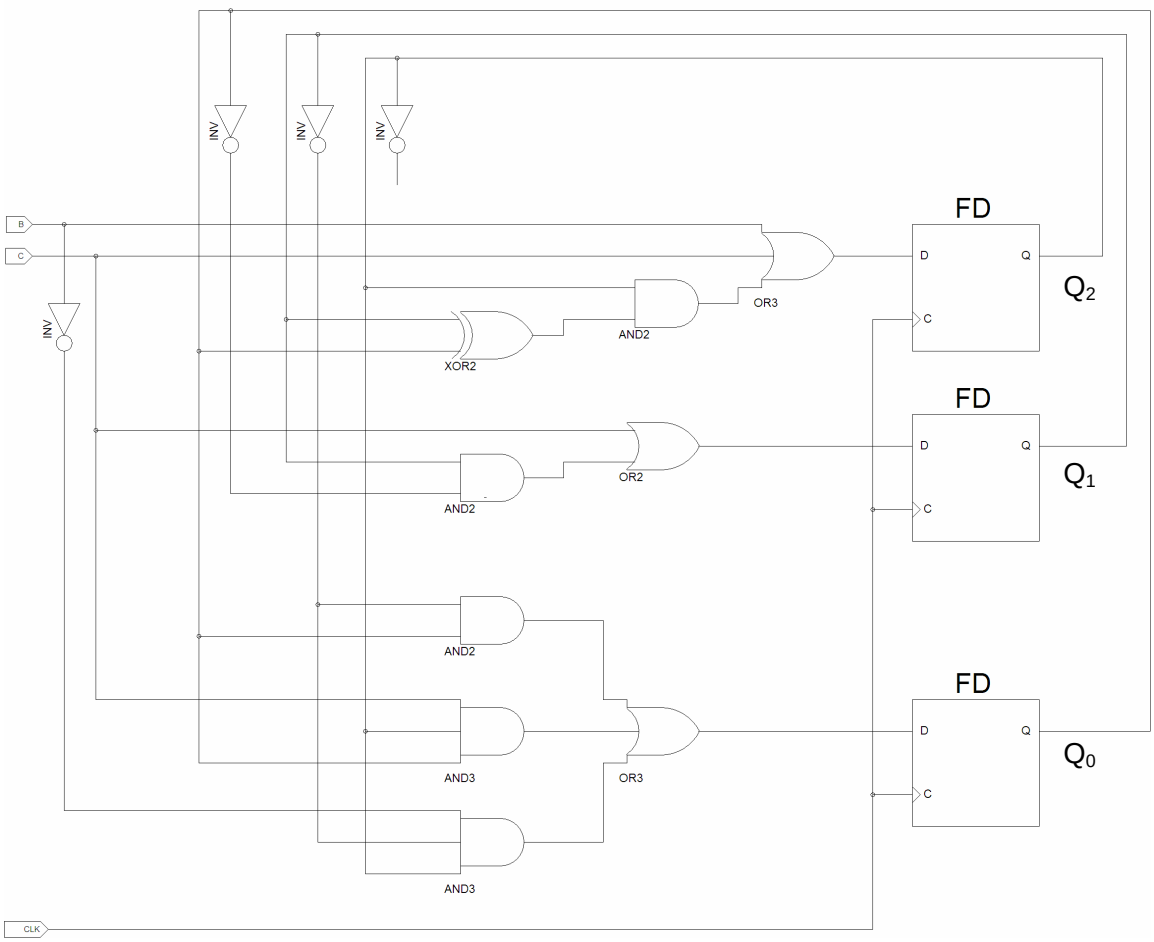
avec les transitions indéterminées associées.

B C Etat	0 0	0 1	1 1	1 0
$E_0 \rightarrow 000$	001	111	φφφ	101
$E_1 \rightarrow 001$	000	110	φφφ	100
$E_7 \rightarrow 011$	φφφ	φφφ	φφφ	φφφ
$E_6 \rightarrow 010$	φφφ	φφφ	φφφ	φφφ
$E_3 \rightarrow 110$	111	111	φφφ	φφφ
$E_2 \rightarrow 111$	001	110	φφφ	φφφ
$E_5 \rightarrow 101$	100	φφφ	φφφ	100
$E_4 \rightarrow 100$	000	φφφ	φφφ	101

$$D_2 = Q_2 = B + C + q_2.(q_1 \oplus q_0)$$

$$D_1 = Q_1 = C + q_1.\overline{q_0}$$

$$\overline{D_0} = \overline{Q_0} = \overline{q_1}.q_0 + q_2.q_0.C + q_2.\overline{q_1}.\overline{B}$$


INV
INV
INV
B
C
INV
XOR2
AND2
OR3
FD
D
Q
C
Q_2
AND2
OR2
FD
D
Q
C
Q_1
AND2
AND3
OR3
FD
D
Q
C
Q_0
AND3
CLK

Exercice 5 *: Synthèse de l'additionneur.*

1. On peut procéder en 5 étapes pour établir le diagramme d'états.

Etape 1 : l'état initial (avant l'addition de a_0 et b_0) est nécessairement L_0. Tant que les entrées sont **ab = 00** il n'y a pas de retenue générée, la sortie reste à 0 et le système ne change pas d'état. Si **ab = 10** ou **01** alors S = 1 et il n'y a toujours pas de retenue. Le système évolue vers l'état L_1.

Etape 2 : Tant que les entrées sont **ab = 10** ou **01** alors S = 1 et la retenue est nulle. Le système reste dans l'état L_1. Par contre si **ab = 00** alors S = 0 et le système doit retourner dans l'état L_0.

Pour le moment nous n'avons considéré que des situations ou il n'y a pas de retenue.

Etape 3 : Depuis L_0 si **ab = 11** alors S = 0 mais il faut tenir compte d'une retenue pour la prochaine addition. Il est incorrect de laisser le système vers l'état L_0 puis que cet état considère une retenue nulle pour la prochaine addition. Le système évolue vers l'état H_0

La même remarque s'applique pour L_1 lorsque **ab = 11.**

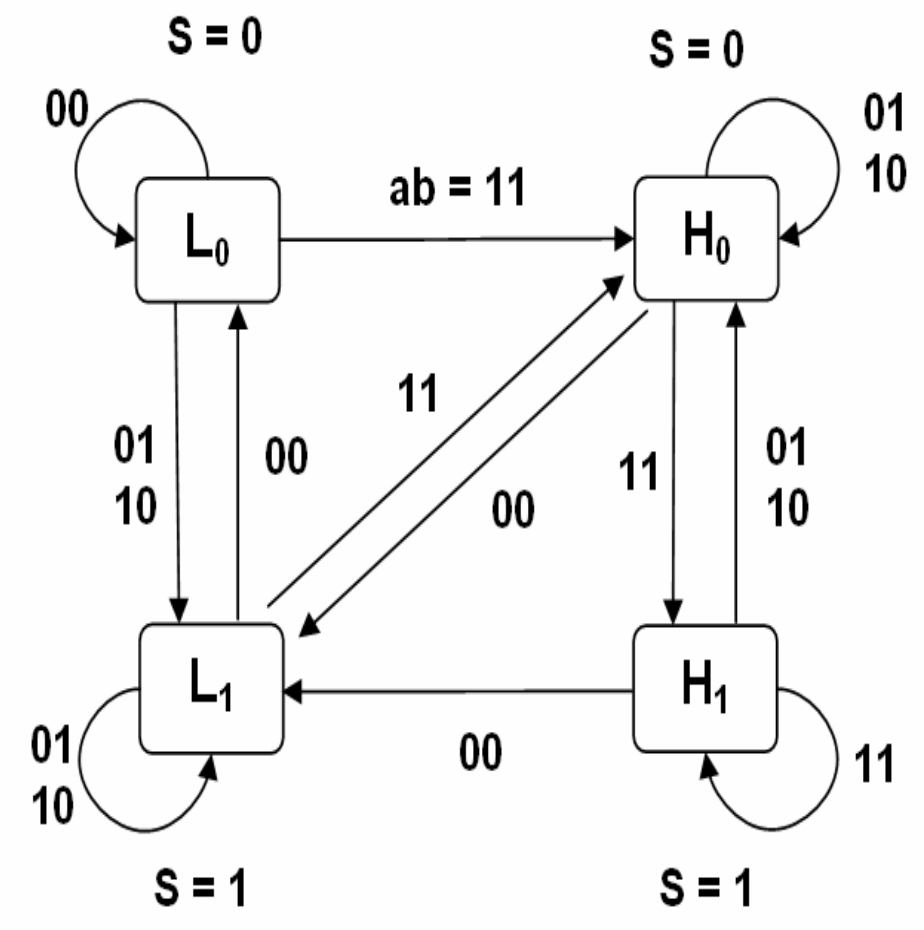
Etape 4 : Depuis H_0

- Si **ab = 01** ou **10** alors S = 0 (on calcule a + b+ retenue) la retenue est toujours 1 et le système reste dans l'état H_0.
- Si **ab = 00** alors S = 1 la retenue devient nulle et le système évolue vers l'état L_1.
- Si **ab = 11** alors S = 1 mais il y a toujours une retenue à 1. Le système évolue vers l'état H_1.

Etape 5 : Depuis H_1.

- Si **ab = 11** alors S = 1 et il y a toujours une retenue égale à 1. Le système reste dans l'état H_1.
- Si **ab = 00** alors S = 1 et la retenue est nulle. Le système évolue vers L_1.
- Si **ab = 10** ou **01** alors S = 0 mais la retenue est toujours à 1. Le système évolue vers H_0.

Au final on obtient le diagramme d'états suivant.



Les tables de transitions et des sorties s'obtiennent facilement à partir de ce diagramme :

a **présent ** **b**	**0** **0**	**0** **1**	**1** **1**	**1** **0**	**Sortie** **S**
L_0	L_0	L_1	H_0	L_1	0
L_1	L_0	L_1	H_0	L_1	1
H_0	L_1	H_0	H_1	H_0	0
H_1	L_1	H_0	H_1	H_0	1

Etat futur

2. En utilisant le codage proposé dans l'énoncé on obtient la table des excitations secondaires :

a / **q_2 q_1 \ b**	**0** **0**	**0** **1**	**1** **1**	**1** **0**	**Sortie** **S**
0 0	0 0	0 1	1 1	0 1	0
0 1	0 0	0 1	1 1	0 1	1
1 1	0 1	1 1	1 0	1 1	0
1 0	0 1	1 1	1 0	1 1	1
	Q^+_2 Q^+_1				

Q^+_2 et Q^+_1 représentent les valeurs futures des variables secondaires q_2 et q_1.

3. Les équations de commande des 2 bascules D sont $D_2 = Q_2^+, D_1 = Q_1^+$ soit en interprétant la table des excitations précédentes:

$D_2 = Q_2^+ = a.b + q_2.(a+b), D_1 = Q_1^+ = q_2.\bar{a} + \overline{q_2}.b + a.\bar{b}$ et $S = q_2 \oplus q_1, C_{i-1} = q_2$

Exercice 6 : *Synthèse d'une machine d'état et états parasites.*

1. Le principe de fonctionnement impose de mémoriser sur deux fronts actifs d'horloge deux états haut « 1 » successifs ou deux états bas « 0 » consécutifs : il faut donc 2*2 états.

L'état A a vu passer deux états « 0 » successifs : **la sortie S est à « 0 »**

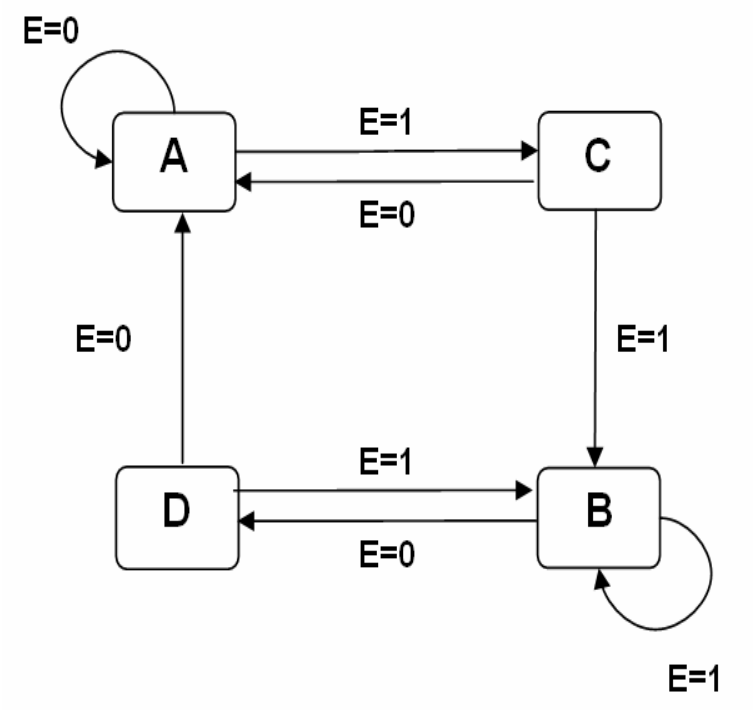
L'état B a vu passer deux états « 1 » successifs **: la sortie S est à « 1 »**

L'état C a vu passer un seul état « 1 » : **la sortie S est encore à « 0 »**

L'état D a vu passer un seul état « 0 » : **la sortie S est encore à « 1 »**

Dans le cahier des charges ou sur le chronogramme on remarque que si on a un unique front actif sur l'entrée au lieu des deux fronts successifs, la sortie ne change pas. La machine d'état peut donc revenir sur l'état A ou B selon le cas.

Au final, on obtient le diagramme d'états suivant.



Remarque : avec un machine de type Mealy, on obtient le même résultat avec uniquement 3 états. Le nombre de bascules est donc identique mais le machine n'est plus synchrone comme convenue dans le texte.

2. Pour le codage, les 4 états réclament deux bascules Q_1Q_0. Si on prend un codage symétrique on aura une combinatoire simplifiée et une des sorties égale à l'une des sorties des bascules l'état A « 00 », l'état B « 11 », l'état C « 01 », et l'état D « 10 » .

Les tables de transitions et des sorties s'obtiennent facilement à partir du diagramme de la question **1.** :

Q_1Q_0 \ E	0	1	Sortie S
A	A	C	0
C	A	B	1
B	D	B	0
D	A	B	1

Etat futur $Q_1^+Q_0^+$

En utilisant ce codage :

Q_1Q_0 \ E	0	1	Sortie S
A : 00	00	01	0
C : 01	00	11	1
B : 11	10	11	0
D : 10	00	11	1

Etat futur $Q_1^+Q_0^+$

3. Etant donné l'équation de la bascule D : $Q_i^+=D_i$ après un front actif. Nous pouvons déduire les équations D_i directement de la table de vérité.

$$D_0= Q_0^+ = E$$

$$D_1= Q_1^+ = Q_1.Q_0 + Q_1.E + Q_0.E$$

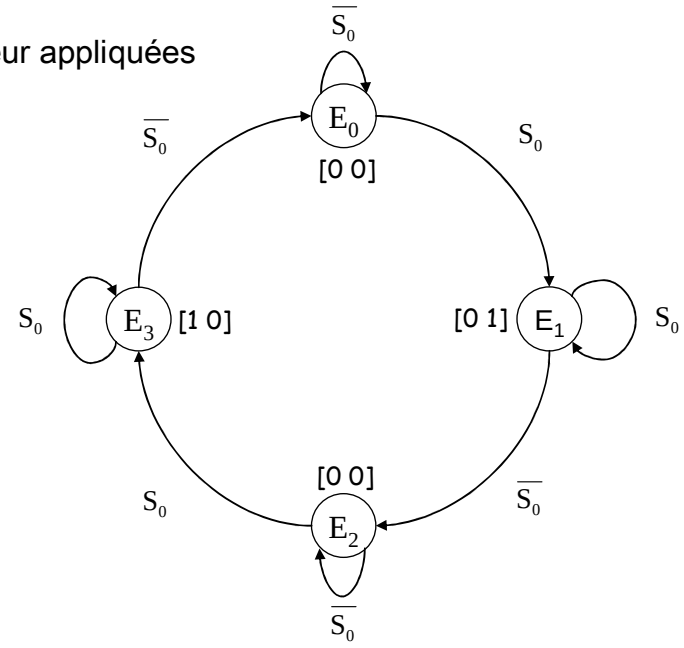
$$\text{Et } S = Q_1$$

Exercice 7 : *Synthèse d'un système séquentiel de contrôle de température.*

A. Un seul détecteur de température délivre le signal logique S_0, ce sera l'unique entrée de notre système et nous avons deux sorties correspondant aux deux signaux logiques de commande des ventilateurs V_1 et V_2.

A 1) Il faut réaliser une machine d'état cyclique permettant d'osciller entre un état de température basse et un état de température haute en passant successivement par le ventilateur V_1 puis le ventilateur V_2. Cela nous mène au diagramme suivant :

[0 0] représente la valeur appliquées aux ventilateurs



A 2) Il faut à présent choisir un codage pour les états E_i. Etant donné que nous 4 états, 2 bascules sont nécessaires et il se trouve que justement nous avons 2 sorties, On peut donc choisir un codage qui limitera les l'équations de calcul de sortie en fonction des valeurs de sorties des bascules.

	S_0 q_2 q_1 \	0	1
E_0	**0 0**	0 1	0 0
E_1	**0 1**	0 1	1 1
E_2	**1 1**	1 0	1 1
E_3	**1 0**	1 0	0 0

Etats futurs codés Q_2 Q_1

	q_2 q_1	V_1	V_2
E_0	**0 0**	0	0
E_1	**0 1**	0	1
E_2	**1 1**	0	0
E_3	**1 0**	1	0

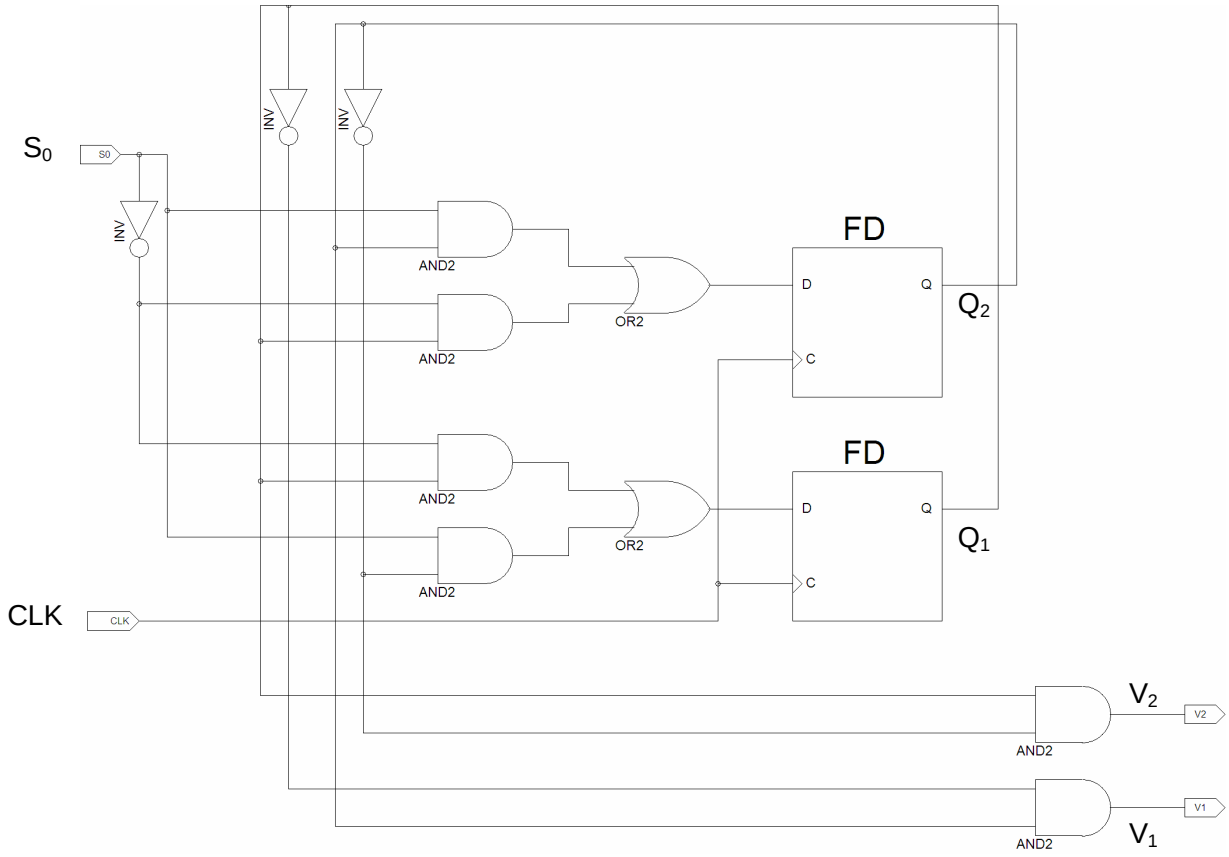
Tables des sorties

A 3) Pour faire la synthèse on nous propose de prendre des bascules D on a donc une seule entrée de commande pour la bascule et $D_i = Q_i$ au front actif suivant. Les tables précédentes nous donne les équations :

$$D_2 = S_0.q_2 + \overline{S_0}.q_1 \qquad D_1 = \overline{S_0}.q_1 + S_0.\overline{q_2}$$

et

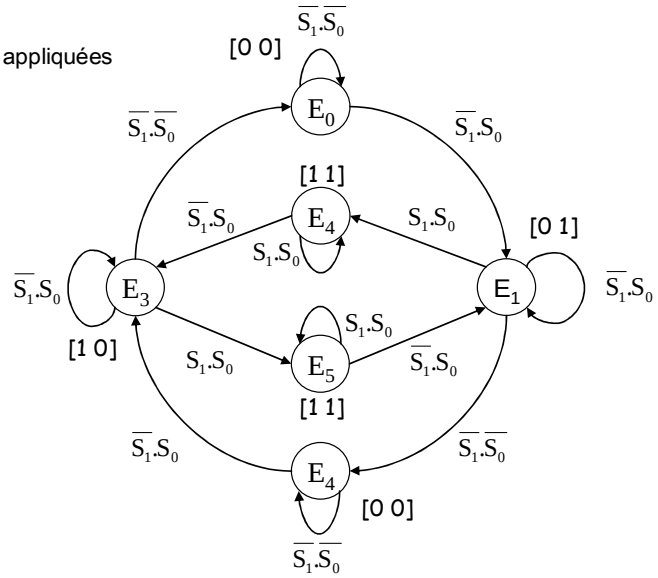
$$V_2 = Q_1.\overline{Q_2} \qquad V_1 = \overline{Q_1}.Q_2$$


S0
INV
INV
INV
AND2
AND2
OR2
FD
D
Q
C
Q2
AND2
AND2
OR2
FD
D
Q
C
Q1
CLK
CLK
V2
AND2
V2
V1
AND2
V1

B. A présent on utilise 2 détecteurs.

B 1) Dans le cas d'un fonctionnement normal, des transitions ne peuvent se produire : par exemple en partant de E_0, il n'est pas possible que S_1=1 et S_0=0. Il est existe 10 transitions impossibles pour les états utilisés et pour simplifier l'analyse, il est important de bien identifier ces cas avant de commencer le diagramme. D'autre part, le cahier des charges, en nous donnant une seconde entrée, ajoute un cas particulier qui ne remet pas en cause le cycle de la partie A. Il faut créer deux états supplémentaires afin de différentier quel ventilateur, V_1 où V_2 à démarrer en premier afin de l'éteindre lorsque la température est redescendue entre T_{S1} et T_{S0}.

[0 0] représente la valeur appliquées aux ventilateurs



B 2) On utilise le codage suivant pour les différents états :

A partir du diagramme nous pouvons établir la table des transitions en ajoutant les transitions impossibles et les états « parasites » c'est-à-dire non utilisés dans le fonctionnement de base, respectivement notés φ et φφ.

S1 S0 présent	0 0	0 1	1 1	1 0
E_0	E_0	E_1	φ	φ
E_1	E_2	E_1	E_4	φ
E_2	E_2	E_3	φ	φ
E_3	E_0	E_3	E_5	φ
E_4	φ	E_3	E_4	φ
E_5	φ	E_1	E_5	φ
E_6	φφ	φφ	φφ	φφ
E_7	φφ	φφ	φφ	φφ

Etats futurs

	q_3 q_2 q_1	V_1	V_2
E_0	0 0 0	0	0
E_1	0 0 1	0	1
E_2	1 0 0	0	0
E_3	0 1 0	1	0
E_4	0 1 1	1	1
E_5	1 1 1	1	1

Les sorties

B 3) La table des transitions.

S_1 S_0 $q_3\ q_1\ q_1$	**0** **0**	**0** **1**	**1** **1**	**1** **0**
000	000	001	φ	φ
001	100	001	011	φ
100	100	010	φ	φ
010	000	010	111	φ
011	φ	010	011	φ
111	φ	001	111	φ
101	φφ	φφ	φφ	φφ
110	φφ	φφ	φφ	φφ

B 4) avec des deux bascules D, on a $D_i = Q_i$:

$$D_3 = \begin{matrix} \overline{S_1}.\overline{S_0}.\overline{q_3}.\overline{q_2}.q_1 \ + \\ \overline{S_1}.\overline{S_0}.q_3.\overline{q_2}.\overline{q_1} \ + \\ S_1.S_0.\overline{q_3}.q_2.\overline{q_1} \ + \\ S_1.S_0.q_3.q_2.q_1 \end{matrix} = \begin{matrix} \overline{S_1}.\overline{S_0}.\overline{q_2}(q_3 \oplus q_1) \ + \\ S_1.S_0.q_2.\overline{(q_3 \oplus q_1)} \end{matrix}$$

$$D_2 = \begin{array}{l} S_0.\overline{q_3}.q_2.q_1 \ + \\ S_0.\overline{q_3}.q_2.\overline{q_1} \ + \\ S_1.S_0.q_2.q_1 \ + \\ S_1.S_0.q_3.q_1 \ + \\ \overline{S_1}.S_0.q_3.\overline{q_1}.\overline{q_0} \end{array} = \begin{array}{l} S_0.\overline{q_3}.q_2 \ + \\ S_1.S_0.q_2.(q_3+q_2)+ \\ \overline{S_1}.S_0.q_3.\overline{q_1}.\overline{q_0} \end{array}$$

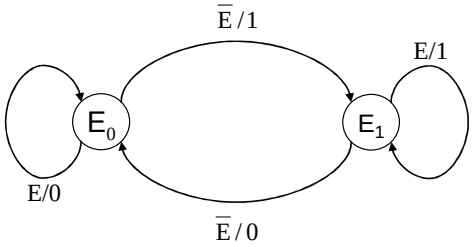
$$D_1 = \begin{array}{l} \overline{S_1}.S_0.\overline{q_3}.\overline{q_2} \ + \\ S_1.S_0.\overline{q_3}.q_1 \ + \\ S_1.S_0.q_2.q_1 \ + \\ S_1.S_0.\overline{q_3}.q_2 \end{array} = S_0.\overline{q_3}.(\overline{S_1}.\overline{q_2}+S_1.q_2+S_1.q_1) \ + \ S_1.S_0.q_2.q_1$$

B 5) L'intérêt du ce codage est qu'il permet d'associer directement les sorties des bascules 2 et 1 aux sorties V_2 et V_1.

Exercice 8 : *Synthèse d'un système séquentiel de codage.*

Selon la valeur de l'entrée E soit il faut soit commuter la valeur de la sortie soit la conserver, donc quelque soit le cas il est nécessaire de mémoriser cette valeur. Deux valeur sont possibles '0' ou '1', il faut a priori deux états : E_0 (état où Z=0 pour E=1) et E_1 (état où Z=1 pour E=1).

1. Nous proposons le diagramme d'état suivant :



Deux états sont suffisants pour réaliser la fonction. Vérifions qu'en 'envoyant' la séquence de l'énoncé nous retrouvons bien la séquence de sortie imposée.

2. la table des transitions et la table des sorties sont alors :

E / Etat présent ****	**0**	**1**
E_0	**E_1**	**E_0**
E_1	**E_0**	**E_1**

Etats Futurs

E / Etat présent ****	**0**	**1**
E_0	**1**	**0**
E_1	**0**	**1**

Sortie Z

3. le code est imposé par la sortie puisque nous avons 2 états (soit 1 bascule) et 2 valeurs possibles pour la sortie donc autant prendre $Q_0 = Z$.

- $E_0 \rightarrow Q_0 = 0$
- $E_1 \rightarrow Q_0 = 1$

E / q_0 ****	**0**	**1**	**Sortie Z**
0	**1**	**0**	0
1	**0**	**1**	1

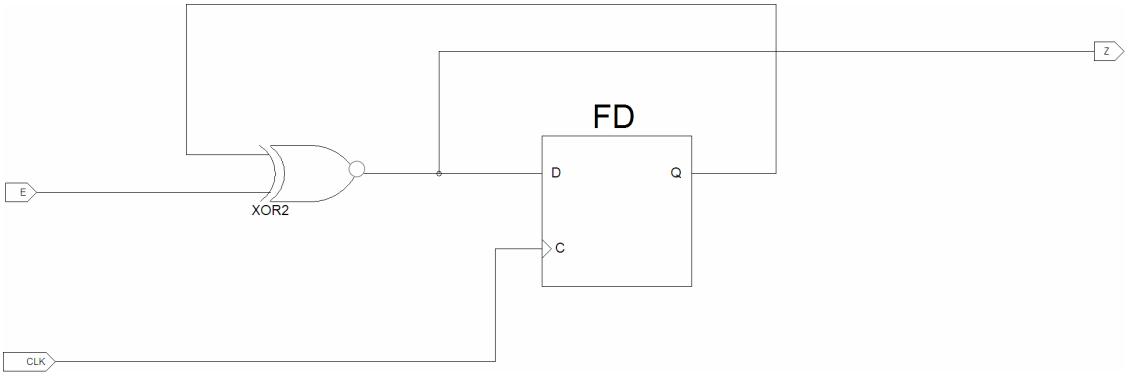
Q_0^+

Les expressions sont alors $Q_0^+ = E.Q_0 + \overline{E}.\overline{Q_0} = \overline{E \oplus Q_0}$ et $Z = Q_0^+$

4. Les équations précédentes imposent la valeur future de la bascule et donc l'état dans lequel doit être le système. A présent il nous faut câbler les entrées de cette bascule. Sachant que l'énoncé impose l'utilisation de bascules D et que celle-ci recopie la valeur de son entrée D sur sa sortie au front actif suivant, il vient que :

$$D_0 = Q_0^+ = \overline{E \oplus Q_0}$$

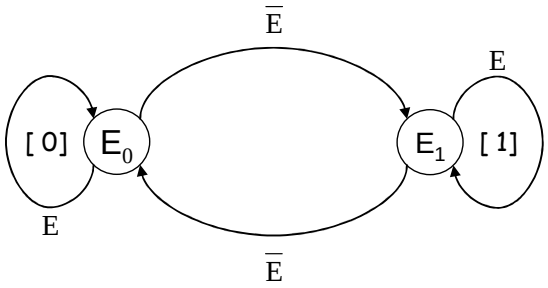
et :



Structure de Moore :

Il faut toujours deux états E_0 (au coup d'horloge précédent, la valeur précédente fut '0') et E_1 (la valeur précédente fut '1').

1. Nous proposons le diagramme d'état suivant :



2. la table des transitions est alors :

E Etat présent ****	**0**	**1**	**Sortie** **Z**
E_0	**E_1**	**E_0**	0
E_1	**E_0**	**E_1**	1

Etats Futurs

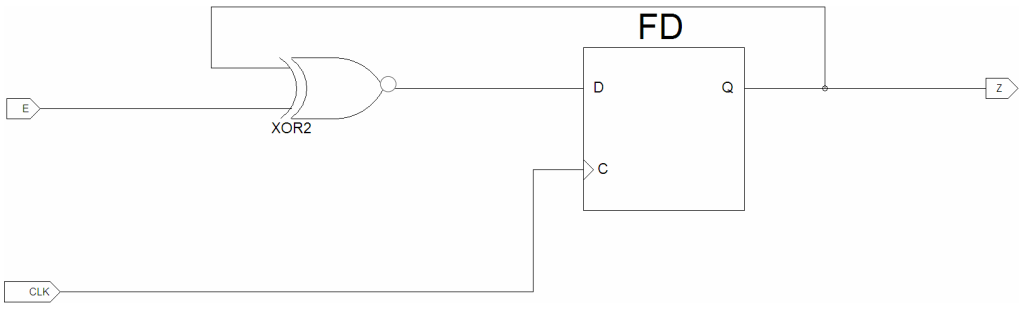
3. le code est imposé par la sortie puisque nous avons 2 états (soit 1 bascule) et 2 valeurs possibles pour la sortie donc autant prendre $Q_0 = Z$.

E q_0 ****	**0**	**1**	**Sortie** **Z**
0	**1**	**0**	0
1	**0**	**1**	1

Q_0^+

Les expressions sont presque identiques $Q_0^+ = E.Q_0 + \overline{E}.\overline{Q_0} = \overline{E \oplus Q_0}$ et $Z = Q_0^+$

4. $D_0 = Q_0^+ = \overline{E \oplus Q_0}$



On peut voir que sur un cas simple il y a peu de différences entre une structure de Moore et de Mealy. Bien sûr, la différence qui est, a contrario, fondamentale concerne la synchronisation des sorties. Dans le cas de Moore les sorties sont synchrones avec l'horloge et dans le cas de Mealy si un changement s'opère sur la valeur de l'entrée au milieu d'une période d'horloge, la sortie changera en suivant.

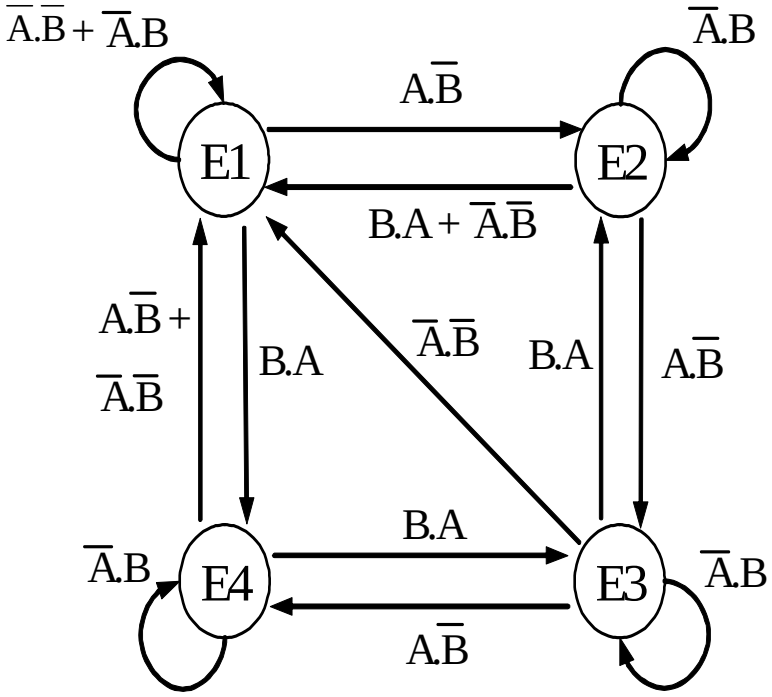
Exercice 9 : *Synthèse d'un compteur-décompteur.*

Ici, nous devons réaliser un compteur multimodes à 4 états donc 2 bascules (Q_1,Q_0) sont nécessaires. Le fait de compter en Gray ne rajoute pas de difficulté, la valeur de l'état (Q_1,Q_0) devra être décodé en code Gray à l'aide d'un bloc combinatoire intercalé entre l'étage de bascules et les sorties finales (G_1 en MSB, G_0). On veillera à ce que le choix du codage des états permette de simplifier les équations du décodeur. Les 4 différentes modes (comptage, décomptage, RAZ et blocage) sont contrôlés par 2 entrées A et B.

1. Lorsqu'on réalise un compteur, il est important que toutes les sorties soient synchrones. Si on utilise une structure de Mealy, il est possible que le calcul d'une des sorties utilise les entrées A,B ou l'horloge contrairement aux autres. Alors un décalage temporel entre les sorties pourrait être observé.

2. Le diagramme des transitions :

On retrouve la structure cyclique caractéristique d'un compteur. Sachant que nous prévoyons 4 modes de fonctionnement, il faut bien vérifier que 4 flèches partant de chaque état soit au total 16 transitions.



3. la table de transition :

B / **Etats A** / **Présents **	**0** / **0**	**0** / **1**	**1** / **1**	**1** / **0**	**Sorties** $G_1 G_0$
E_1	E_1	E_2	E_4	E_1	0 0
E_2	E_1	E_3	E_1	E_2	0 1
E_3	E_1	E_4	E_2	E_3	1 1
E_4	E_1	E_0	E_3	E_4	1 0

Etats futurs

Etant donné que notre circuit doit compter en code GRAY, dans la colonne de droite nous donnons les quatre premières valeurs en Gray. On peux choisir le codage des états

tel que :

$Q_1 = G_1$ et **$Q_0 = G_0$**.

4. En utilisant le codage nous obtenons :

	B **$q_1\ q_0$** **\ A**	**0** **0**	**0** **1**	**1** **1**	**1** **0**	**Sorties** **$G_1\ G_0$**
E_1	**0 0**	0 0	0 1	1 0	0 0	**0 0**
E_2	**0 1**	0 0	1 1	0 0	0 1	**0 1**
E_3	**1 1**	0 0	1 0	0 1	1 1	**1 1**
E_4	**1 0**	0 0	0 0	1 1	1 0	**1 0**

Valeurs futures des bascules **$Q_1\ Q_0$**

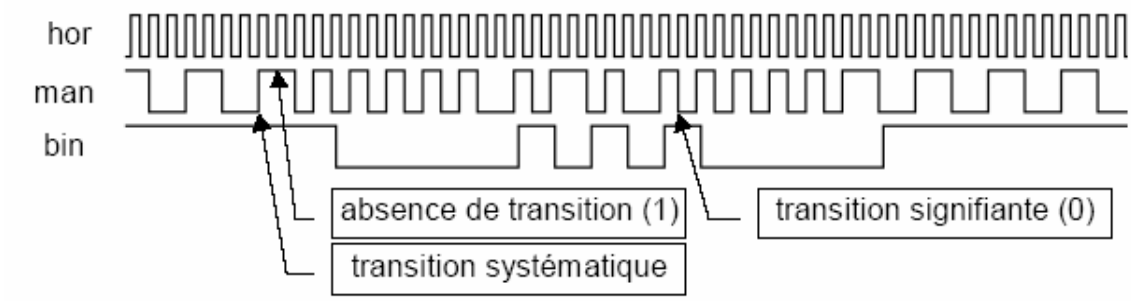
On souhaite réaliser le compteur avec des bascules D. Donc il nous faut déterminer les équations de commande des 2 bascules D_i et Comme $D_i = Q_i$.

$D_1 = Q_1 =$ $q_0.A.\overline{B} + \overline{q_0}.A.B + q_1.\overline{A}.B$

$D_0 = Q_1 =$ $\overline{q_1}.A.\overline{B} + q_1.A.B + q_0.\overline{A}.B$

Remarque : les tables de Karnaugh des deux variables Q_1 et Q_0 présentent des similitudes. L'idée est de réaliser pour les deux des regroupements suivant les colonnes ainsi nous retrouvons les mêmes groupes booléens ce qui permet de diminuer le nombre de portes logiques nécessaires.

Problème : *Synthèse du Code Manchester différentiel*.



Dans les communications séries entre ordinateurs on utilise généralement des techniques particulières de codage pour les signaux qui circulent sur le câble, par exemple, le codage *Manchester différentiel*. En codage Manchester différentiel, chaque intervalle de temps élémentaire, pendant lequel un signal binaire est placé sur le câble, nommé le plus souvent « temps bit », est divisé en deux parties de durées égales avec les conventions suivantes :

- Un signal binaire 1 est représenté par une absence de transition au début du

temps bit correspondant.

- Un signal binaire 0 est représenté par la présence d'une transition au début du temps bit considéré.

- Au milieu du temps bit il y a *toujours* une transition.

Le chronogramme de la figure précédente représente un exemple d'allure des signaux. Les données à décoder, le signal man, sont synchrones d'une horloge *hor* ; on souhaite réaliser un décodeur qui fournit en sortie le code binaire correspondant, *bin*. Etant donné la spécificité du codage qui consiste à rajouter une transition au milieu du temps bit, il faut travailler avec une horloge de fréquence plus élevé que la fréquence du « message ». Mais le véritable problème est qu'il va falloir être capable de distinguer les transitions signifiantes des transitions systématiques. Et ceci n'est pas possible lors d'une séquence de '0', il faut attendre le prochain '1' pour être sûr du début du temps bit. Il faudra donc plusieurs temps bits pour pouvoir décoder la suite or au début de la transmission, on ne peut pas savoir dans quel état se trouve le système. Comme il n'est pas acceptable de perdre l'information du début de transmission, il faudra nécessairement envoyer une séquence de resynchronisation au bout de laquelle la machine d'état

(quelque soit l'histoire précédente) se trouvera dans un état connu.

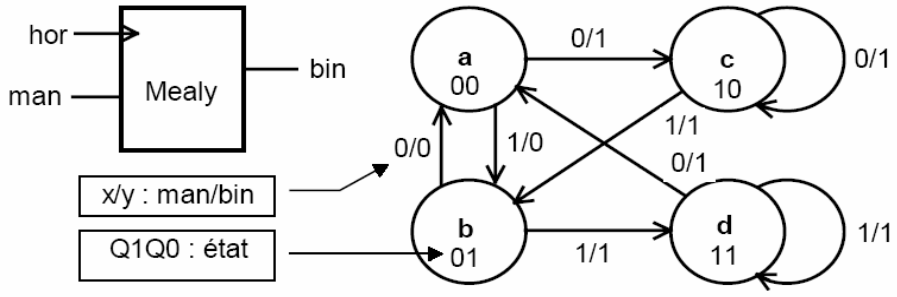
D'autre part, la sortie du décodeur est retardée d'une période d'horloge par rapport à l'information d'entrée, en effet il faut attendre d'avoir la transition signifiante afin d'interpréter le signal. En général, les sorties des décodeurs sont retardées d'une période entière, ici comme on travaille avec une horloge à T/2 le décalage peut être minimisé.

Analyse du problème :

L'idée est assez simple, nous allons construire une machine d'états qui, parcourt un premier cycle quand le signal d'entrée change à chaque période d'horloge, ce qui correspond à un '0' transmis, et change de cycle quand elle détecte une absence de changement du signal d'entrée, qui correspond à un '1' transmis.

Machine de Mealy :

L'absence de changement peut se produire tant pour un niveau haut que pour un niveau bas du signal d'entrée, d'où l'ébauche de diagramme de transitions de la figure suivant



On se convaincra facilement que les cycles parcourus sont :

- a→b→a→b→a→b→a... ou b→a→b→a→b→a→b... pour trois '0' consécutifs transmis,

- a→c→b→d→a→c→b... ou b→d→a→c→b→d→a... pour trois '1' consécutifs transmis.

En fonctionnement permanent, une fois le système synchronisé et sauf erreur dans le code d'entrée, les conditions de maintien dans les états c et d sont toujours fausses, elles servent à la synchronisation en début de réception. Du diagramme précédent on

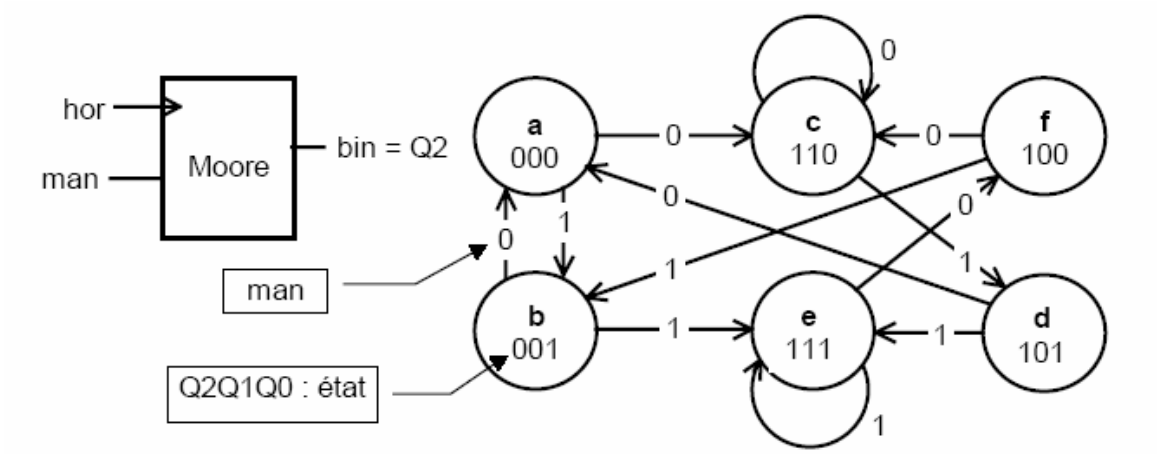
déduit les équations de commandes des bascules, D1 et D0, et celle de la sortie ; après quelques simplifications on obtient :

$D_0 = man$

$D_1 = \overline{Q_0 \oplus man}$

bin = $Q_1 + \overline{Q_0 \oplus man}$

Machine de Moore



Dans une machine de Moore, différentes valeurs des sorties correspondent à des états différents, le diagramme de transitions doit donc contenir plus d'états. Le diagramme ne représente pas deux états inutilisés, 2 et 3, en décimal. Leur affectation se fait lors du calcul des équations de commandes, de façon à les simplifier au maximum (si man = '1', 3→7 et 2→5 ; si man = '0', 3→4 et 2→6). On obtient, après quelques manipulations :

$D_0 = man$

$D_1 = \overline{Q_0 \oplus man}$ et bin = Q_2

$D_2 = Q_1 + \overline{Q_0 \oplus man}$

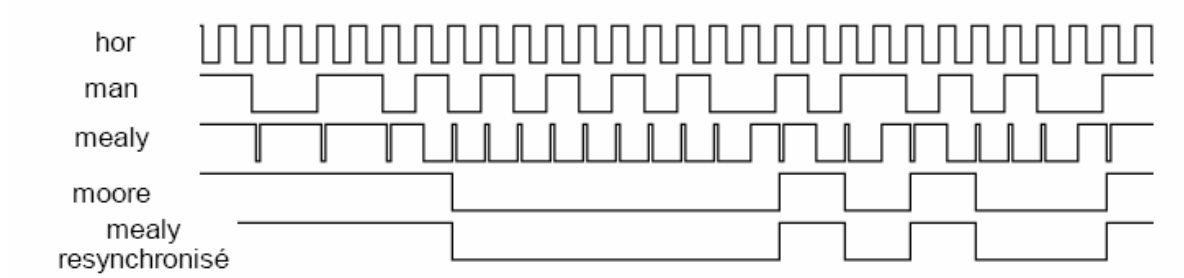
qui sont exactement les mêmes équations que celles obtenues dans le cas de la machine de Mealy, malgré l'apparente complexité du diagramme de transitions. Mais pour être complètement honnête, il est évident que le codage des états a été choisi de façon à

retrouver le même résultat mais quelque soit le codage, la complexité des équations est à peu près similaire.

Comparaison :

Sur l'exemple que nous venons de traiter, il apparaît comme seule différence une bascule supplémentaire en sortie, pour générer le signal *bin*, dans le cas de la machine de Moore.

En regardant plus attentivement l'architecture des deux systèmes, on constate que la sortie combinatoire de la machine de Mealy risque de nous réserver quelques surprises : son équation fait intervenir des signaux logiques qui changent d'état simultanément, d'où des risques de création d'impulsions parasites étroites au moment des commutations. Une simulation confirme ce risque :



3.5. *Exercices supplémentaires.*

Exercice 1 : *Questions pour un champion*.

Faire la synthèse du premier des exercices corrigés du chapitre à l'aide de bascules JK à front montant.

Exercice 2 : *Compteur.*

Faire la synthèse d'un compteur binaire par 12. Est il possible de mettre en série des compteurs pour réaliser le compteur par 12.

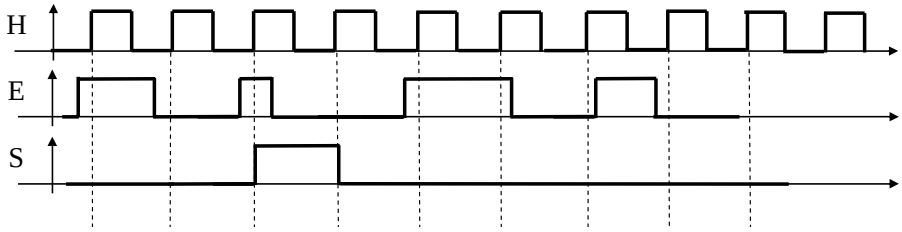
Exercice 3 : *diviseurs de fréquence.*

1. En s'appuyant sur les exercices du chapitre, faire un circuit possédant deux sorties. Une impulsion sur deux du signal d'horloge sera aiguiller vers la première sorties, les autres impulsions iront vers l'autre sorties. Faire un chronogramme. Que peut-on dire des signaux de sorties. Faire la synthèse en bascules D puis en bascule JK.

2. faire la synthèse d'un circuit synchrone qui peut sur la même sortie diviser le signal d'horloge par 2 ou par 3 selon la valeur de l'entrée

Exercice 4 : *détecteur de séquence.*

Faire la synthèse d'une machine de Moore qui reconnaît les séquences 1-0-1 ou 0-1-0 sur l'entrée E. De plus il faut que les séquences ne se superposent pas :



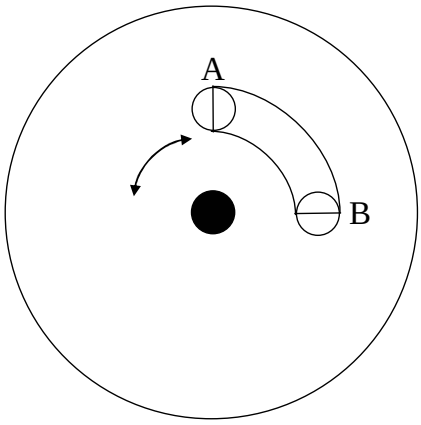
La sortie S passe à ‘1’ lors de la dernière valeur de la séquence.

Exercice 5 : *Machine de Mealy.*

Comme il est précisé dans la correction de l’exercice 5 des exercices corrigés, il est possible de tracer un diagramme d’état d’une machine de Mealy avec seulement 3 états. Retrouver ce diagramme d’état.

Exercice 6 : *Rotation d’un moteur.*

On cherche à déterminer le sens de rotation d’un moteur grâce à un disque en rotation sur son axe, percé d’une fente en arc de cercle comme le montre la figure suivante.



Le disque est placé entre deux couples de photodiodes (A et B) et de LEDS de manière à ce que la fente puisse masquer alternativement les photodiodes lors de la rotation du moteur. Le signal de la photodiode a donc la forme d'un créneau dont le rapport cyclique est déterminé entre autre par la longueur de la fente et la période par la vitesse de rotation. La fente est suffisamment longue pour que les deux photodiodes puissent être éclairées en même temps.

1. Montrer que dans ces conditions la séquence des signaux sur A et B permet de trouver le sens de rotation du disque.

2. Synthétiser la machine d'état ayant pour entrées A et B et pour sortie R, R='1' dans le cas d'une rotation dans le sens trigonométrique et R='0' dans le cas contraire.

Chapitre 4

Circuits logiques programmables.

4.1. *Introduction.*

Pour réaliser des circuits numériques les concepteurs utilisent de plus en plus la programmation. On distingue deux types de programmation : la logique programmée (programmation logicielle) et les circuits logiques programmables (**P**rogrammable **L**ogic **D**evice ou ***PLD***). Les processeurs et les microcontrôleurs appartiennent à la première catégorie. Ils exécutent de manière séquentielle une suite d'instructions prédéfinies. Le comportement physique du composant est toujours le même, son architecture est figée. Par exemple les broches 9 à 16 du microcontrôleur MC68HC11F1 correspondront toujours aux entrées du bus d'adresses. Bien qu'ils aient permis de révolutionner l'électronique numérique, ces composants ne répondent pas toujours aux exigences de rapidité et flexibilité.

A l'inverse, l'architecture des circuits programmables ***PLD*** est entièrement configurable par le concepteur qui peut ainsi décider des broches qui serviront d'entrées et de sorties, mais aussi de la fonction logique réalisée sur chacune des sorties. Un ***PLD*** peut être vu comme une boite noire contenant des portes logiques et des interrupteurs programmables. Les interrupteurs programmables permettent de relier "à façon" les portes logiques pour réaliser la fonction souhaitée. En pratique la programmation est réalisée directement à partir d'un ordinateur de type PC auquel est relié le composant.

Depuis vingt ans de nombreuses familles de circuits sont apparues avec des noms très divers selon les fabricants. Pour des raisons commerciales deux composants similaires peuvent être appelés différemment par deux constructeurs concurrents. Il n'est donc pas toujours très facile de se repérer parmi tous les noms de circuits. La figure 38 présente une taxinomie simplifiée des circuits logiques programmables (***PLD***).

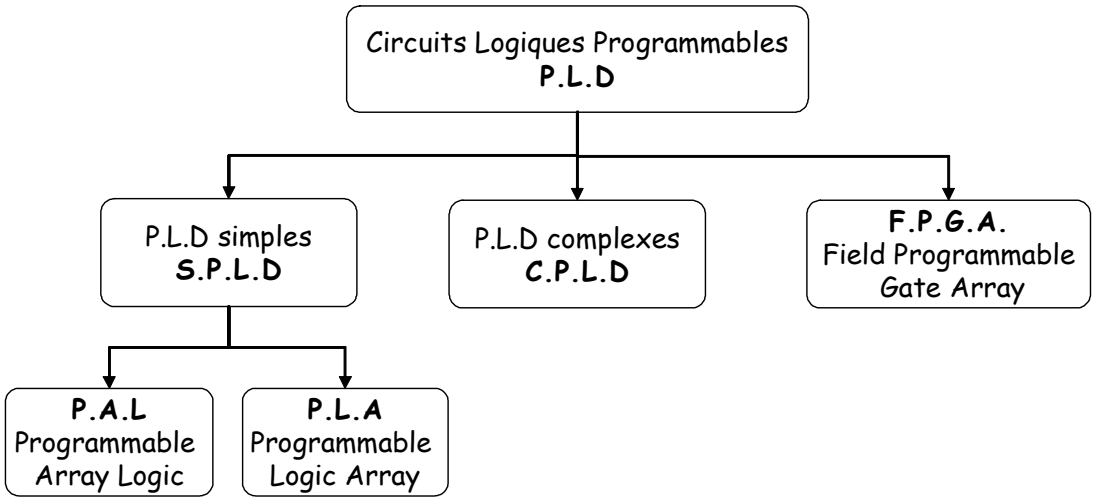


figure 38 Taxinomie simplifiée des différents types de composants logiques

4.2. *Les circuits logiques programmables simples* SPLD*.*

4.2.1. Structure générale des SPLD.

Tous les *SPLD* sont constitués :

- D'entrées I_1 à I_n avec $8 \leq n \leq 20$.
- De sorties O_1 à O_n ou d'entrées/sorties IO_1 à IO_m avec $2 \leq m \leq 15$.

Selon les composants on peut également trouver

- Une entrée d'horloge *CLK*
- Une entrée de validation des sorties *OE* (Output Enable)
- Une entrée de remise à zéro des registres *Reset*.

Sur le schéma fonctionnel d'un *SPLD* (figure 39) on distingue nettement deux parties. La matrice programmable d'une part, et la structure de sortie non programmable d'autre part. Nous allons d'abord détailler la matrice de programmation.

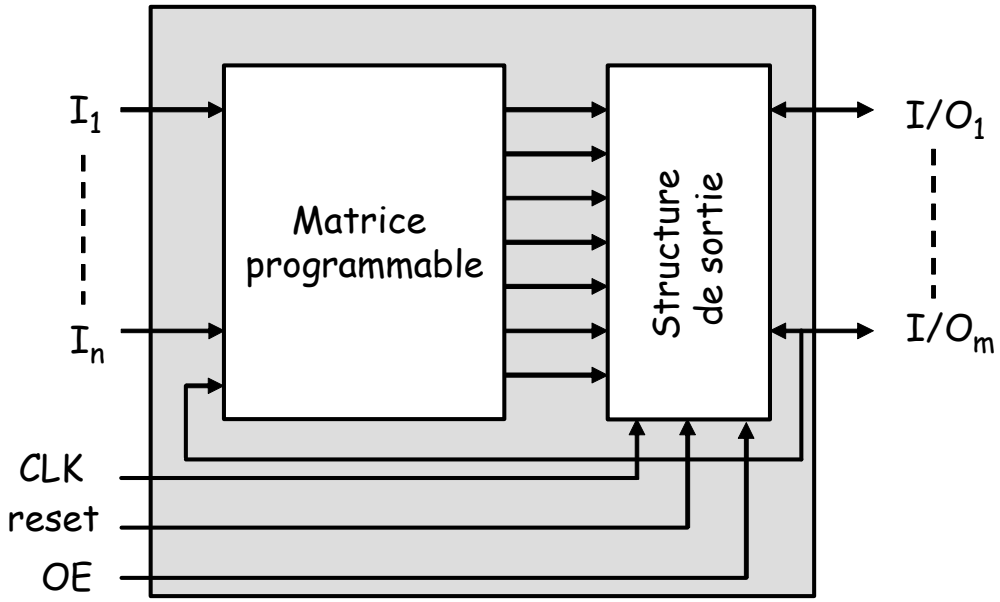


figure 39 Schéma général d'un SPLD

4.2.2. Les Programmable Logic Array PLA.

Plusieurs types de ***PLD*** sont commercialement disponibles aujourd'hui. Les premiers circuits développés furent les Programmable Logic Array (**PLA**). L'idée de base est que toutes les fonctions logiques peuvent être écrites sous la forme d'une somme de produits (forme canonique). Un **PLA** se compose donc d'un ensemble de portes **ET** relié à un jeu de portes **OU.** La structure générale d'un **PLA** est représentée sur la figure 40. Après avoir été complémentées, les entrées $x_1, \cdots, x_n$ arrivent sur un circuit appelé "*réseau ET*" (AND plane). Ce réseau peut être configuré pour que les sorties $P_1, \cdots, P_k$ réalisent n'importe quelle fonction **ET** des entrées $x_1, \cdots, x_n$. Les produits $P_1, \cdots, P_k$ servent d'entrées au second circuit, appelé *"réseau OU"* (OR plane), qui réalise les sorties $f_1, \cdots, f_m$. Là encore le réseau peut être configuré pour réaliser n'importe quelle somme de $P_1, \cdots, P_k$. On réalise ainsi arbitrairement toutes les sommes de produits possibles des variables d'entrées $x_1, \cdots, x_n$.

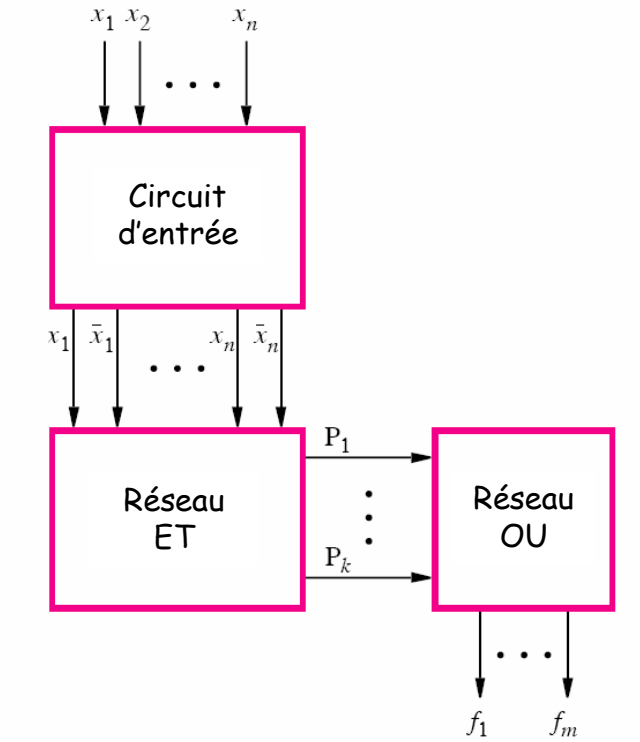


figure 40: Structure générale d'un PLA

Un exemple de PLA avec 3 entrées (x_1, x_2 , x_3), 4 termes produits et 2 sorties est représenté sur la figure 41. Chaque porte **ET** a six entrées correspondant à chaque variable x_i et à son complément $\overline{x_i}$. Les connexions sur une porte **ET** sont programmables ; une entrée connectée est représentée par un symbole ~. Le circuit est réalisé de façon que les entrées non connectées ne perturbent pas le fonctionnement de la porte **ET.** En pratique il existe plusieurs façons de réaliser les connexions programmables.

Sur la figure 41 la porte ET qui réalise P_1 est connectée uniquement à x_1 et x_2, on a donc $P_1 = x_1 \bullet x_2$ De la même façon $P_2 = x_1 \bullet \overline{x_3}$, $P_3 = \overline{x_1} \bullet \overline{x_2} \bullet x_3$ et $P_4 = x_1 \bullet x_3$. Il existe aussi des connexions programmables dans le réseau **OU**, ainsi la sortie f_1 est connectée à P_1 , P_2 , et P_3 et la sortie f_2 à P_1 , P_3 , et P_4 . Finalement on obtient :

$$f_1 = x_1 \bullet x_2 + x_1 \bullet \overline{x_3} + \overline{x_1} \bullet \overline{x_2} \bullet x_3$$

$$f_2 = x_1 \bullet x_2 + \overline{x_1} \bullet \overline{x_2} \bullet x_3 + x_1 \bullet x_3$$

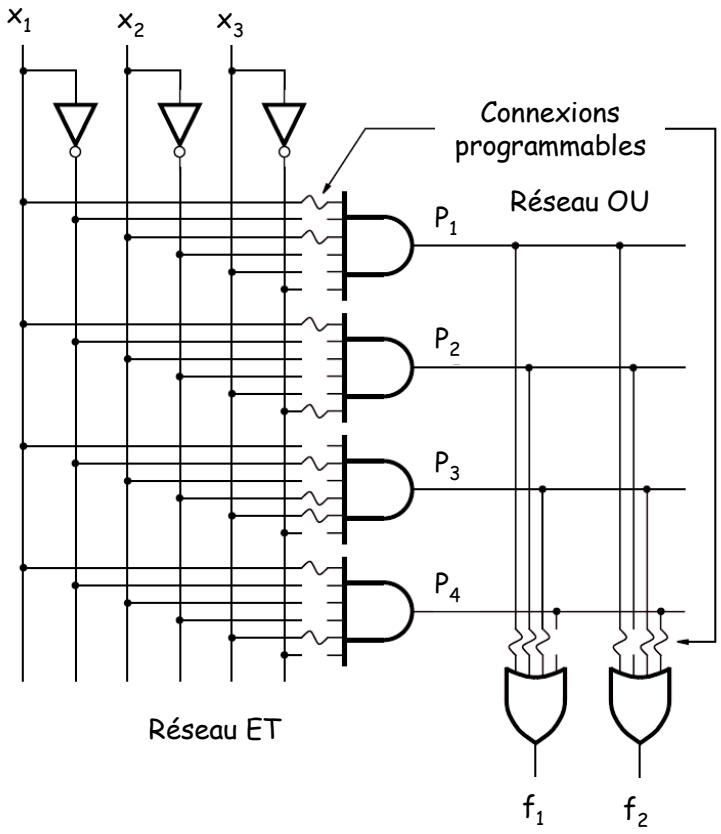


figure 41 Exemple détaillé de PLA

En modifiant la configuration des connexions programmables on peut obtenir en sortie d'autres fonctions logiques f_1 et f_2. La seule contrainte est imposée par la taille du réseau **ET**. Dans l'exemple de la figure 41 seuls quatre termes produit peuvent être obtenus. Typiquement sur les PLA disponibles dans le commerce ont peut trouver jusqu'à 16 entrées, 32 termes produits (32 portes **ET**) et 8 sorties (8 portes **OU**).

On peut difficilement décrire des PLA complexes (et donc plus réalistes) en utilisant les symboles classiques de la figure 41. On trouve généralement d'autres symboles dans la littérature spécialisée. Prenons l'exemple d'une porte **ET** à trois entrées (figure 42). Au lieu d'indiquer les trois entrées on trace une ligne horizontale, les trois variables sont représentées par des traits verticaux et, à chaque intersection, une connexion est indiquée par une croix $\times$. L'absence de croix signifie qu'il n'y a pas de connexion. Ceci s'applique bien sûr à n'importe quel type de portes logiques. La figure 43 représente un PLA vierge

avant programmation dans le quel toutes les connexions existent, et un PLA programmé identique à celui de la figure 41.

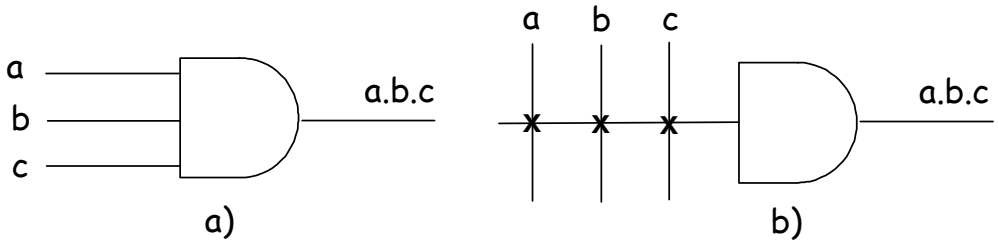


figure 42 Représentation classique d'une porte ET à trois entrées. b) Symbole utilisé pour schématiser les PLD

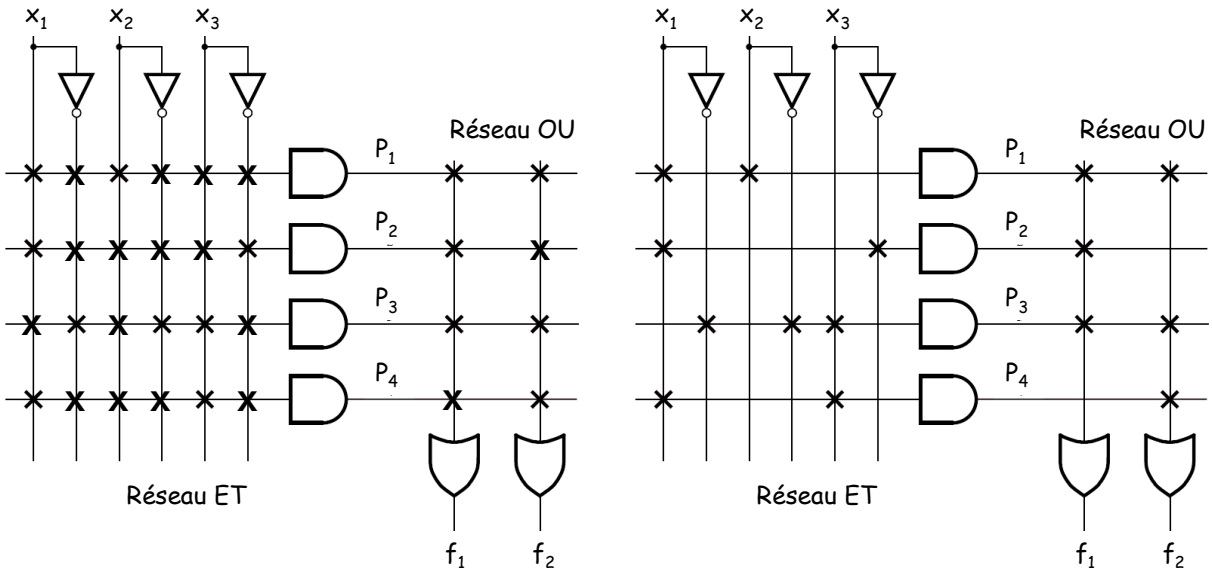


figure 43 Représentations schématiques d'un PLA vierge (à gauche) et du PLA de la figure 41 (à droite)

4.2.3. Les Programmable Array Logic PAL

Dans les PLA les réseaux **ET** et **OU** sont programmables. Historiquement les connexions programmables ont présentées deux difficultés majeures pour les constructeurs : elles étaient délicates à fabriquer correctement, et elles limitaient la vitesse des circuits implémentés sur le PLA. Ceci a conduit au développement de composants

similaires dans lesquels seul le réseau **ET** est programmable tandis que le réseau **OU** est fixe. On parle alors de Programmable Array Logic (PAL). Plus simples à fabriquer, moins chers et plus rapides que les PLA, les PAL ont rapidement trouvé leur place sur le marché des *PLD*.

Un exemple de PAL avec trois entrées, quatre termes de produits et deux sorties est présenté sur la figure **Erreur ! Source du renvoi introuvable.**. Les produits P_1 et P_2 sont connectés de façon fixe avec une porte **OU (**les • indiquent des connexions fixes et non programmables). Il en est de même pour les produits P_3 et P_4.. Le PAL représenté réalise les fonctions logiques :

$$f_1 = x_1 \bullet x_2 \bullet \overline{x_3} + \overline{x_1} \bullet x_2 \bullet x_3$$

$$f_2 = \overline{x_1} \bullet \overline{x_2} + x_1 \bullet x_2 \bullet x_3$$

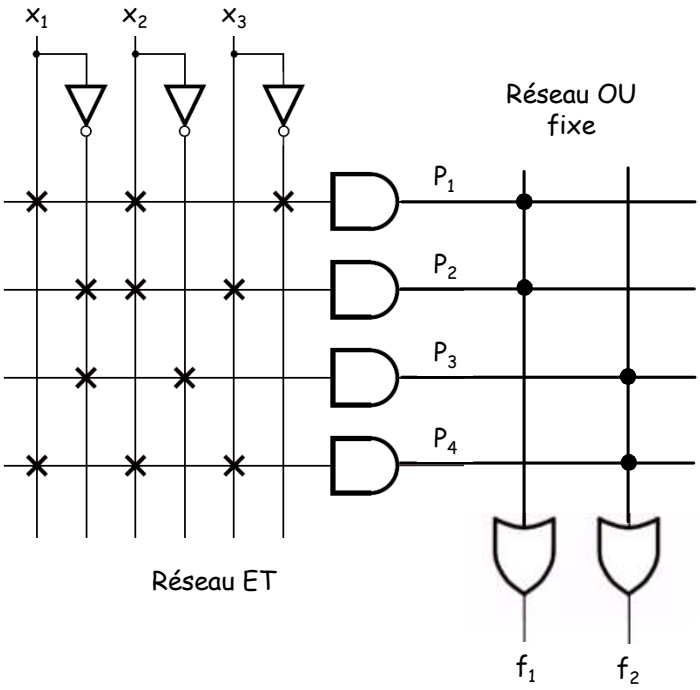


figure 44 Exemple de PAL

Par rapport au PLA présenté plus haut, le PAL est beaucoup moins flexible chaque porte **OU** ne peut sommer que deux termes au lieu de quatre dans le PLA.

4.2.4. La structure de sortie.

Les circuits précédents peuvent paraître bien décevants : au bout du compte on a

connecté des portes **ET** avec des portes **OU** pour réaliser des fonctions logiques combinatoires! En réalité, chaque porte **OU** est suivie d'un circuit de sortie spécifique "*Output Logic MacroCell*" (OLMC) qui permet un fonctionnement beaucoup plus versatile du *SPLD*. La structure des OLMC varie beaucoup selon les types de *SPLD* et les fabricants. Le schéma typique d'une OLMC est représenté sur la figure 45. Le multiplexeur 4 vers 1 permet deux modes de fonctionnement :

- Soit combinatoire, la sortie du réseau OU est alors directement reliée à la sortie du composant.
- Soit séquentiel, la sortie du réseau OU est connectée à une bascule D qui fait office de registre.

En sortie du multiplexeur on trouve un circuit tampon à trois états (*tri-state buffer*) dont le fonctionnement est similaire à celui d'un interrupteur commandable. Si *OE* = 1 l'interrupteur est fermé et la sortie du multiplexeur est reliée à la broche de sortie du composant (mode sortie). A l'inverse, si *OE* = 0 l'interrupteur est ouvert est le multiplexeur 4 vers 1 est isolé, il est alors possible d'utiliser la broche du composant comme une entrée (mode entrée).

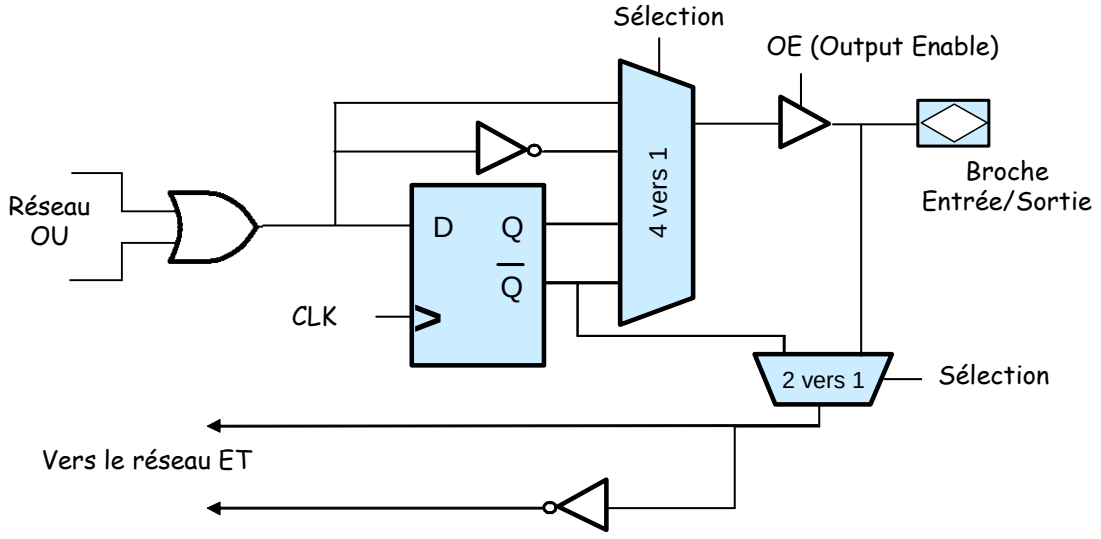


figure 45 Configuration caractéristique d'une macro cellule de sortie OLMC

Pour permettre la réalisation de fonctions logiques séquentielles avec un *SPLD*, un multiplexeur 2 vers 1 permet de réinjecter soit la sortie de la bascule D, soit la valeur

présente sur la broche d'entrée (si *OE* = 0), vers le réseau ET programmable.

4.3. *Les Complex Programmable Logic Device (*CPLD*).*

Les PAL et les PLA ne permettent pas d'implémenter des circuits trop complexes, typiquement le nombre d'entrées et sorties ne peut pas dépasser la trentaine. La nécessité de placer de plus en plus de fonctions sur un même circuit a ainsi conduit à l'intégration de plusieurs *SPLD* sur un même composant. Pour caractériser le niveau d'intégration d'un composant on utilise souvent son équivalence en portes logiques élémentaires telle que les 7400 (portes NAND à deux entrées). Les PAL et PLA représentent l'équivalent de quelques centaines de portes tandis que les *PLD* les plus récents "contiennent " quelques dizaines de milliers de portes élémentaires. Exception faite de leur niveau d'intégration plus élevé, les *CPLD* sont très similaires aux *SPLD*.

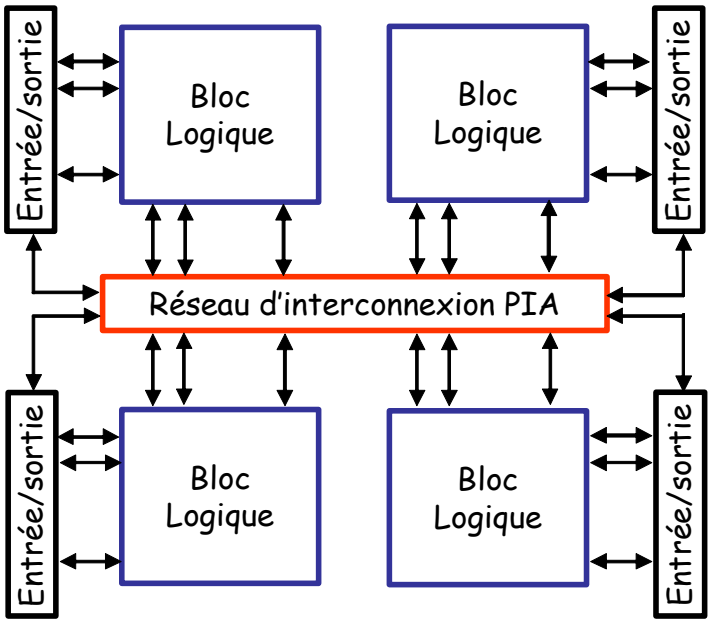


figure 46 Structure d'un Complexe Programmable Logic Device

Un circuit *CPLD* (figure 46) est constitué de plusieurs blocs logiques, équivalents à des PAL ou des PLA, reliés entre eux par un réseau d'interconnexion PIA (Programmable Interconnect Array) et reliés avec l'extérieur par l'intermédiaire de circuit de contrôle des entrées/sorties. Selon l'architecture, chaque bloc logique peut contenir de 4 à 16 macrocellules OLMC.

Les *CPLD* disponibles dans le commerce différent entre eux par le nombre de termes

des produits (ET logique) et le nombre de liaisons offertes par la matrice d'interconnexion. Les *CPLD* étant une évolution des *SPLD*, les outils de développement utilisés pour ces derniers peuvent être employés avec les *CPLD*.

4.4. *Les circuits Field Programmable Gate Array (*FPGA*).*

4.4.1. Architecture générale.

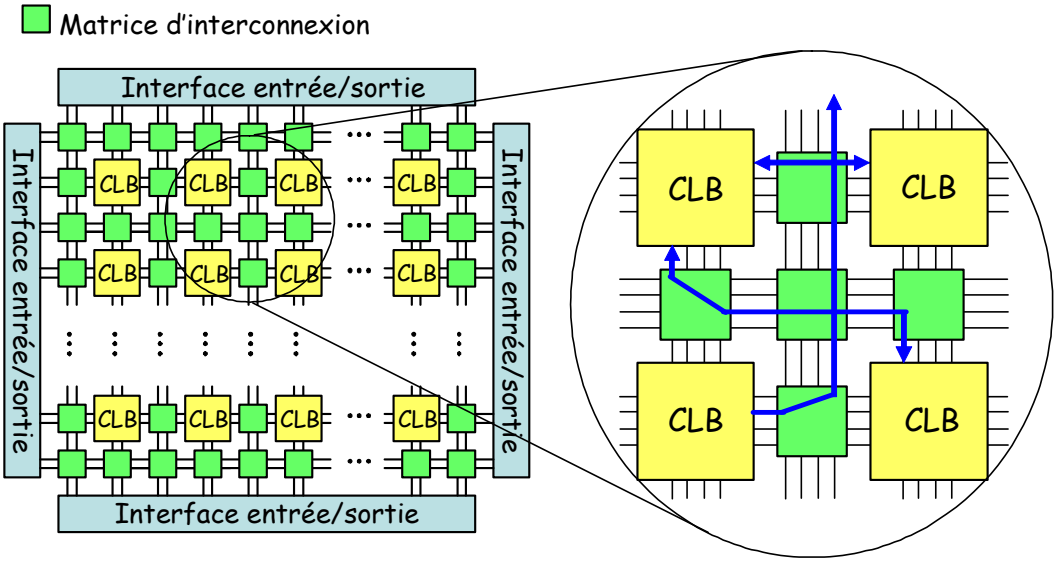


figure 47 Structure générale d'un FPGA

Pour de plus en plus d'applications le niveau d'intégration des *CPLD* n'est pas suffisant. Pour obtenir une plus forte densité d'intégration il faut changer d'architecture, d'où le développement des *FPGA* dont l'organisation est radicalement différente de celle des *CPLD* dans la mesure où ils n'intègrent ni réseaux ET ni réseaux OU, et que les interconnexions entre les différents blocs logiques ne sont pas centralisées. Les éléments de base pour l'implémentation des fonctions logiques sont les blocs logiques configurables (*CLB* pour *Configurable Logic Blocks*). La structure générale d'un *FPGA* est représentée sur la figure 47. On y trouve trois éléments principaux : des *CLB*, des circuits d'interface entrées/sorties et des matrices d'interrupteurs programmables (Switch Matrix). Les *CLB* sont répartis sur un réseau à deux dimensions et les lignes d'interconnexions sont organisées verticalement et horizontalement entre les *CLB*. Les interrupteurs programmables permettent de connecter entre eux les *CLB*, ils relient également les blocs entrée/sortie aux lignes d'interconnexions, ce qui rend l'organisation interne d'un *FPGA* complètement configurable par l'utilisateur. L'ordonnancement des différents CLB et de

leurs liaisons est effectué par des logiciels de placement et de routage.

Les deux principaux fabricants de *FPGA* sont ALTERA et XILINX. Ces deux sociétés proposent des *FPGA* pouvant implémenter des circuits équivalents à plusieurs millions de portes logiques. Certains composants intègrent même de la mémoire RAM.

4.4.2. Les Blocs Logiques Configurables (CLB).

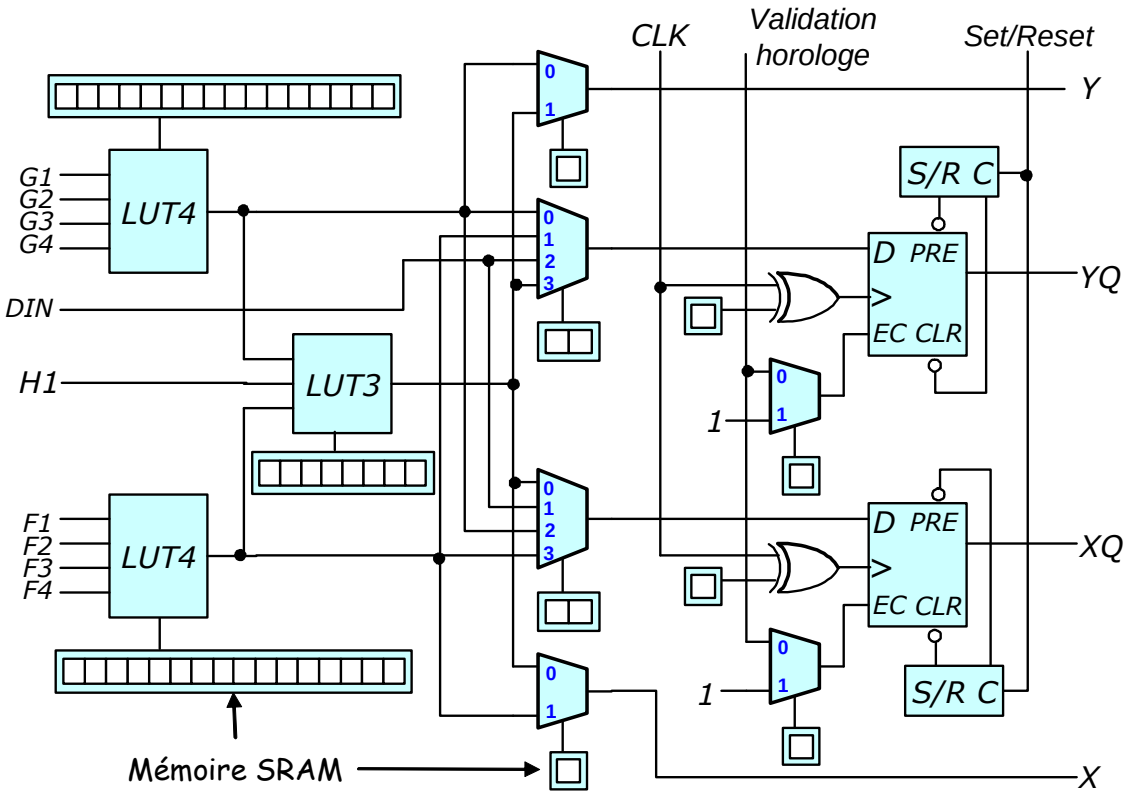


figure 48 Schéma d'un bloc logique configurable

Le schéma des *CLB* présents dans les *FPGA* de la famille XC4000 de XILINX est donné sur la figure 48. On y reconnaît à peu près tous les éléments : bascules D, multiplexeurs, portes XOR, seule la fonction des trois circuits *LUT* présents en entrée doit être précisée. Dans un circuit *LUT* (pour *Lookup Table)* on stocke la table de vérité d'une fonction logique. Une table *LUT* avec *n* entrées est constitué de 2^n cellules mémoire de 1 bit et permet de générer n'importe quelle fonction combinatoire de *n* variables exactement comme on pourrait le faire avec un multiplexeur. Les portes XOR permettent de sélectionner les fronts d'horloge actifs, en effet $CLK \oplus x = \overline{CLK}$ si $x = 1$ ou CLK si $x = 0$.

Configurer un *CLB* pour réaliser une fonction logique consiste à :

- Remplir tout ou partie des *LUT*.
- Définir les entrées de sélection des multiplexeurs.
- Identifier les fronts d'horloge actifs grâce aux portes XOR.

En pratique les contenus des LUT, les entrées de sélections des multiplexeurs ou les valeurs en entrée des portes XOR, sont stockés dans une mémoire RAM statique (SRAM). C'est le contenu de cette SRAM qui est rempli lors de la phase d'implémentation et qui est indiqué par des cases vides sur la figure 48.

Prenons un exemple, on souhaite faire la somme en série de deux nombres *A* et *B* codés sur *m* bits $A=a_{m-1}\ldots a_0$ et $B=b_{m-1}\ldots b_0$. Lors de l'addition des bits a_i et b_i il faut tenir compte de la retenue c_{i-1} obtenue lors de l'addition des bits précédents a_{i-1} et b_{i-1}. Le problème n'est pas très compliqué à résoudre, les tables de vérité pour le bit de somme S_i et le bit de retenue c_i sont :

b_i	a_i	c_{i-1}	S_i	c_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

et les expressions logiques correspondantes s'écrivent :

$$S_i = a_i \oplus b_i \oplus c_{i-1}$$

$$c_i = a_i \bullet b_i + a_i \bullet c_{i-1} + b_i \bullet c_{i-1}$$

On stocke chacune des deux fonctions dans une *LUT*, l'aspect séquentiel introduit par

la gestion de la retenue est facilement résolu avec une bascule D. L'implémentation de cet additionneur série dans un *CLB* est donnée sur la figure 49. Les bits a_i et b_i arrivent en série sur les entrées A et B. On peut noter que le *CLB* n'est pas complètement utilisé puisque la seconde bascule *D* reste disponible à travers la table *LUT3*.

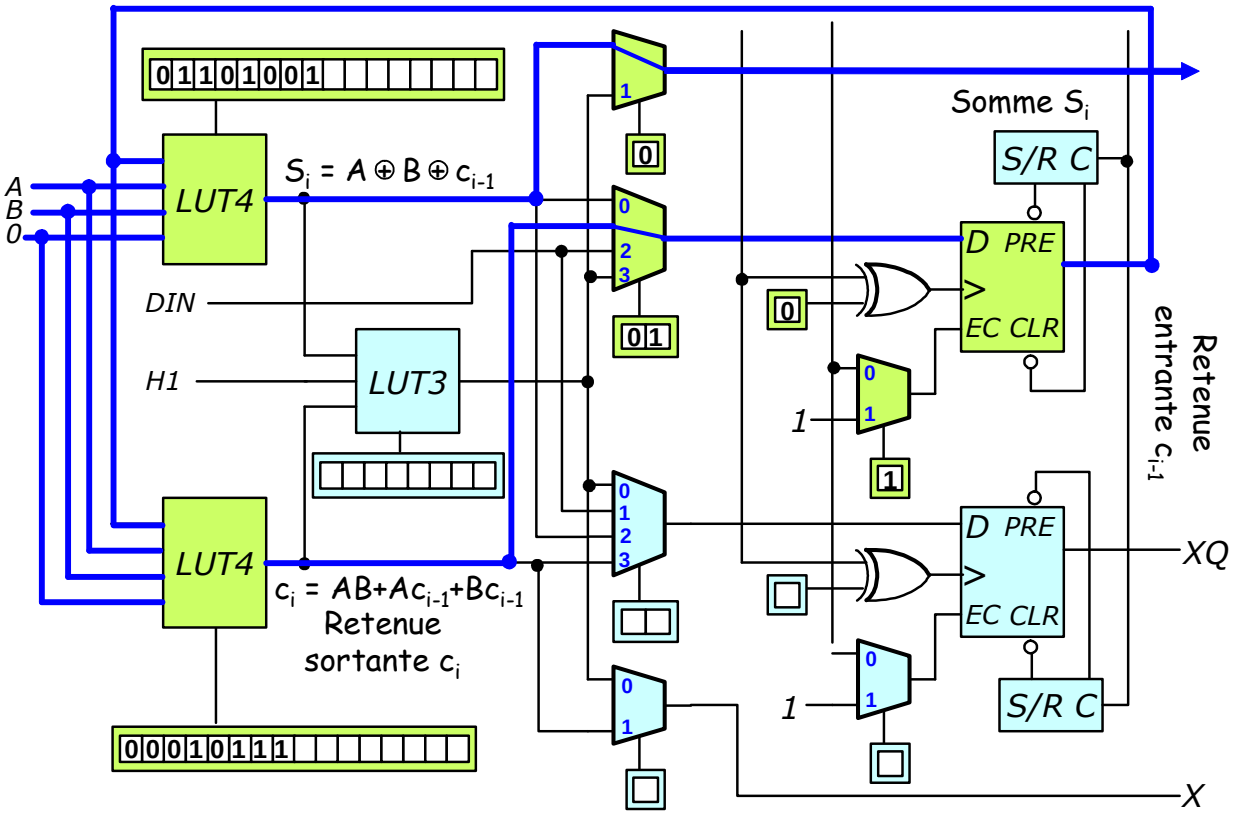


figure 49 Configuration d'un CLB pour la réalisation d'un additionneur série

4.4.3. Les matrices d'interconnexion.

Les transistors MOSFET.

Une étude complète des transistors sort largement du cadre de ce cours. Pour ce qui nous concerne, nous assimilerons les transistors à des interrupteurs. Les transistors les plus utilisés appartiennent à la famille des *Metal Oxide Semiconducteur Field-Effect Transistor (MOSFET).* Brièvement on en distingue deux types, les NMOS et les PMOS dont les symboles et le nom des différentes connexions électriques sont précisés sur la figure 50. Selon le type de transistor et la tension appliquée sur la porte, le transistor peut

être passant (un courant circule entre le drain et la source) ou bloqué (aucun courant ne circule).

En combinant entre eux ces transistors on peut réaliser l'ensemble des fonctions logiques.

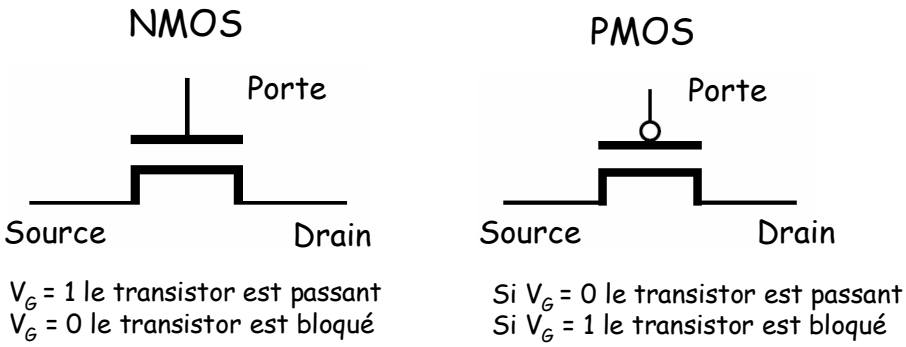


figure 50 Représentation des deux types de transistors de la famille MOSFET

Les matrices d'interrupteurs programmables.

Dans les *FPGA* il existe plusieurs types d'interconnexions, là encore nous ne rentrerons pas trop dans les détails. Le schéma général d'une matrice d'interconnexion est représenté sur la figure 51.a).On trouve six interrupteurs programmables à chaque intersection. Chaque interrupteur est implémenté par un transistor NMOS dont la porte est contrôlée par le contenu d'une SRAM. La connexion entre deux segments est fermée si un 1 est stocké dans la SRAM. A l'inverse si c'est un <<:0:>>0 qui est stocké dans la SRAM la connexion est ouverte. Des exemples de connexions sont illustrés sur la figure 51.b).

Comparaison avec les *CPLD*.

Dans les *FPGA* les blocs logiques sont plus nombreux et plus simples que dans les *CPLD*. La technologie des *FPGA* permet une très forte densité d'intégration comparé aux CPLB. Aujourd'hui il est possible d'implanter dans un *FPGA* une fonction aussi complexe qu'un microcontrôleur. On trouve des bibliothèques de fonctions sur les sites WEB des fabricants.

Dans les *CPLD* les connexions entre les blocs logiques sont fixes. Elles peuvent, ou non, être activées, mais le routage entre les blocs logiques est fixe. Les temps de propagation d'un bloc logique à un autre sont donc constants et prédictibles. A l'inverse dans les *FPGA* le nombre de points de connexions entre deux CLB est fonction de la

position relative des CLB et de l'état d'encombrement des matrices d'interconnexion. Les délais ne sont donc pas prédictibles et vont dépendre du routage réalisé par le logiciel. L'utilisation optimale des capacités d'intégration des *FPGA* ce fait donc au prix d'un routage complexe et délicat qui limite la vitesse de fonctionnement du circuit. On a perdu d'un coté ce que l'on a gagné de l'autre.

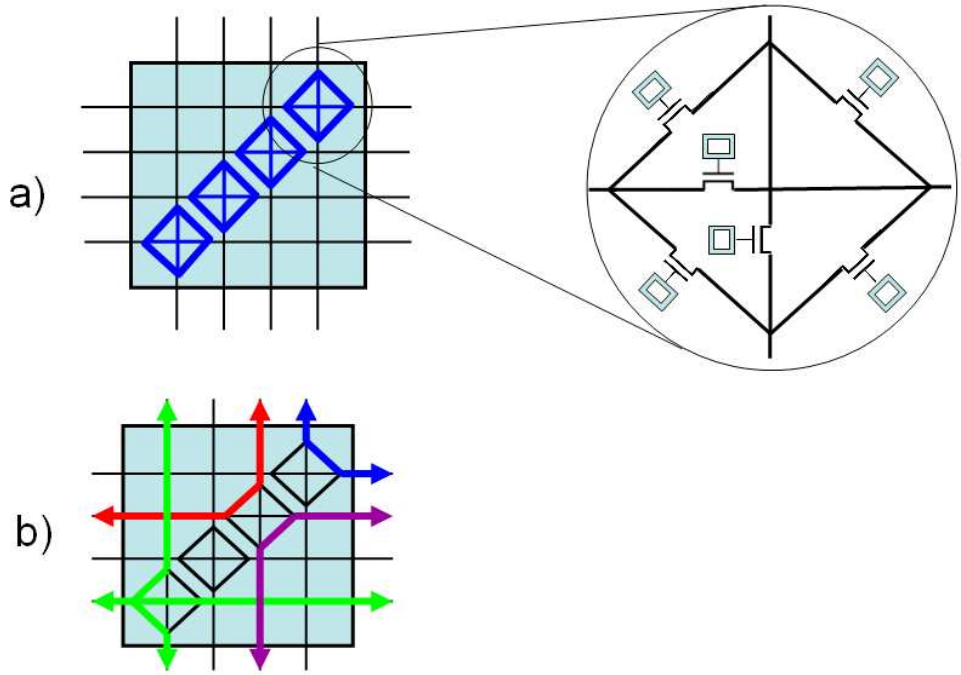


figure 51 Schéma d'une matrice d'interrupteurs programmables b) Exemples de connexions

4.5. *Les outils de développement.*

Les outils de développement différent selon le type de composant et le fabricant. Les principales étapes de la programmation d'un *PLD* sont néanmoins toujours à peu près les mêmes, elles sont représentées sur la figure 52. Dans l'ordre on trouve :

1. Description de la fonction logique, soit par un schéma, une machine d'états ou encore en utilisant un langage de description comportementale HDL (pour *Hardware Description Language*).

2. Compilation et simulation logique (ou fonctionnelle) de la fonction pour vérifier la cohérence de la description et détecter des erreurs de schéma ou de syntaxe.

Eventuellement retour à l'étape précédente.

3. Suivant le type de composant (*FPGA*, *CPLD,* modèle choisi) conversion de la fonction en portes logiques et bascules élémentaires. A ce stade il est possible de spécifier des contraintes techniques comme l'affectation des broches du composant ou les délais entre deux signaux.

4. Optimisation, placement, routage en fonction du composant. Génération du fichier de programmation

5. Simulation temporelle de la fonction incluant tous les paramètres induits par le placement et le routage des blocs logiques. Eventuellement retour aux étapes précédentes

6. Implémentation du fichier de programmation dans le composant.

4.6. *Conclusion.*

Nous avons passé en revue les différents types de circuits programmables. Les *SPLD* comme les *PLA* ou les *PAL*, qui sont peu chers et rapides, sont appropriés pour les fonctions logiques simples. Les *CPLD* peuvent être utilisés pour les mêmes applications que les *SPLD*, mais ils permettent aussi l'implémentation de circuits complexes (> 20000 portes logiques élémentaires). Beaucoup des applications réalisées avec les *CPLD* peuvent également être implémentées dans des *FPGA*. Le choix entre les deux types de circuits dépend de plusieurs facteurs sachant que les *FPGA* sont plus lents que les *CPLD* mais ont une plus grande capacité d'intégration.

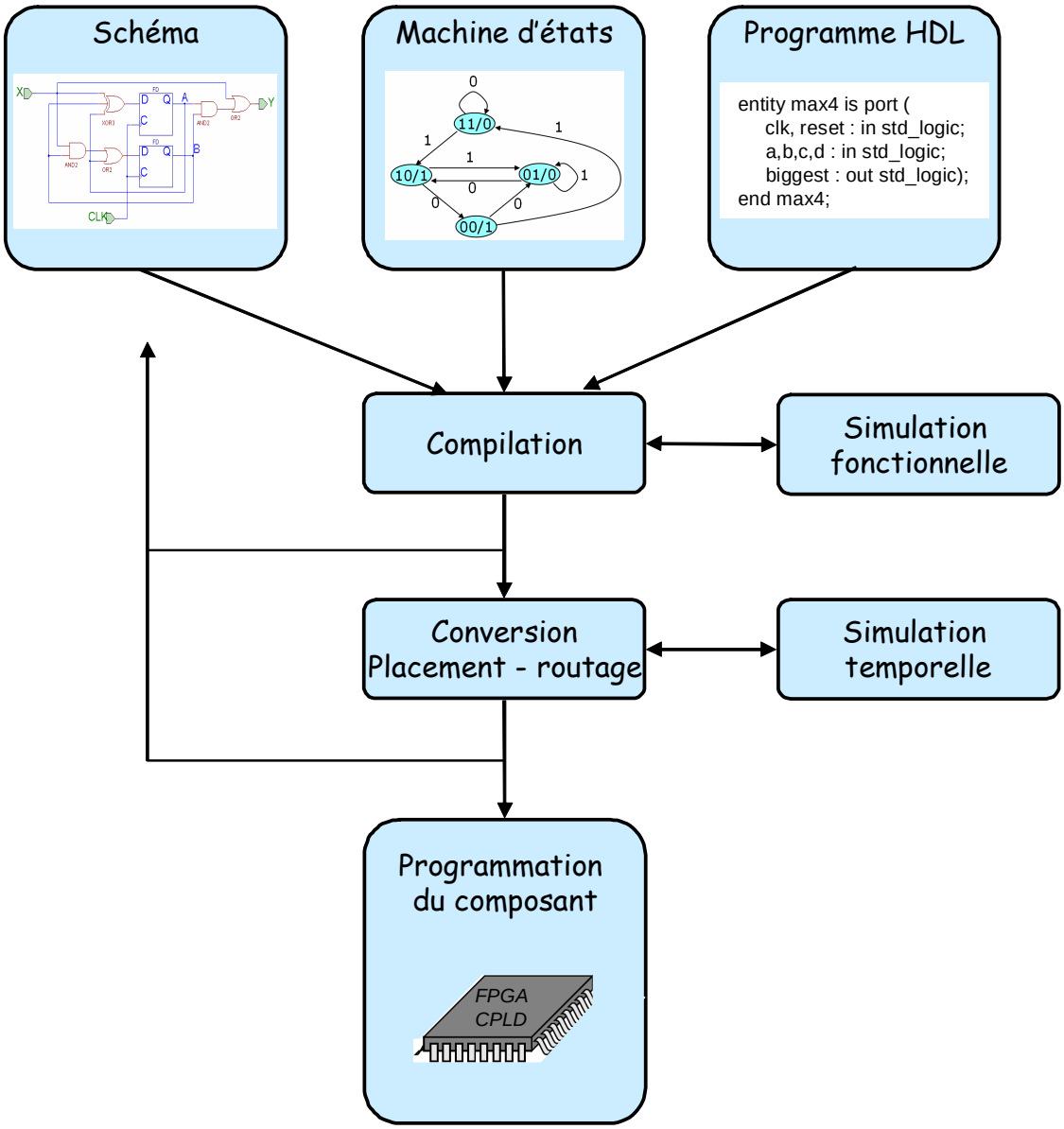


figure 52 Les différentes étapes du développement d'un projet en logique programmable