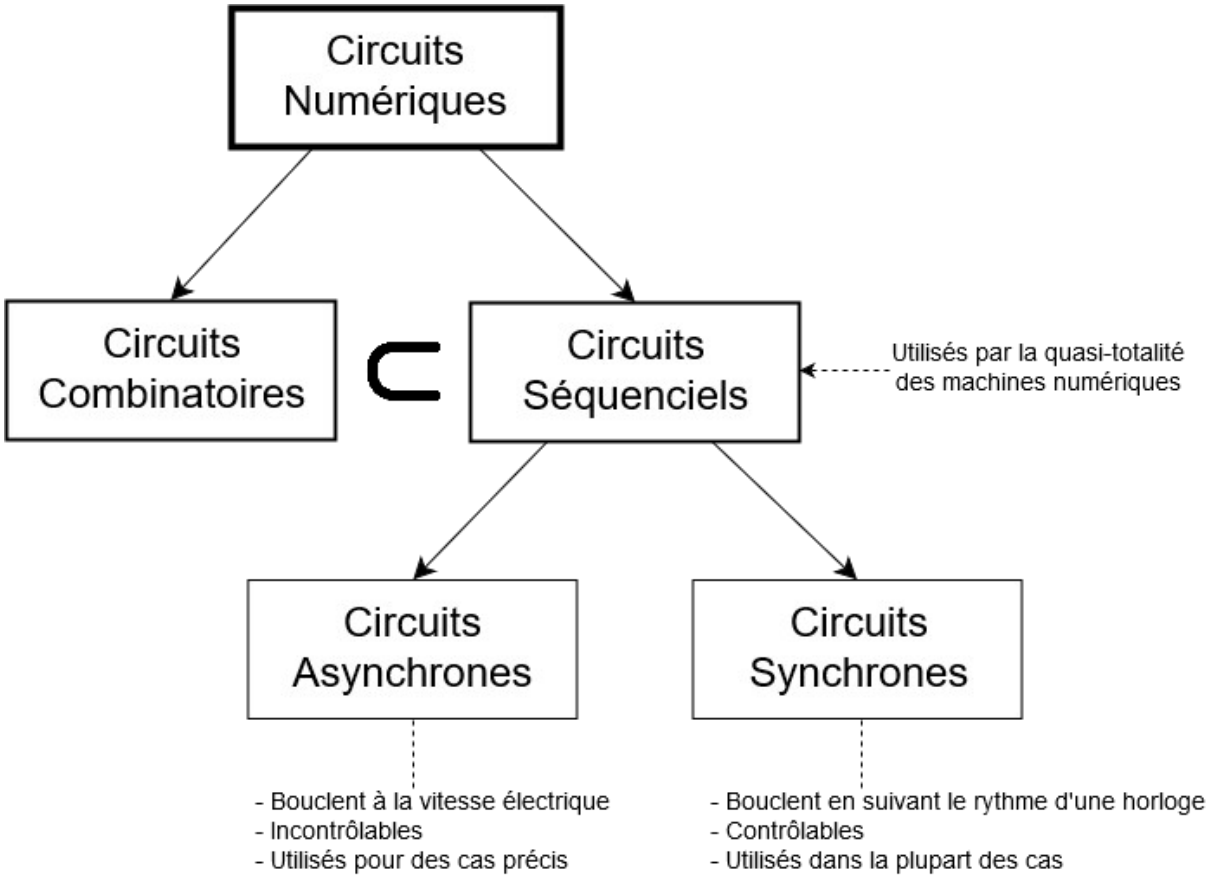
Chapitre 2 : Les Circuits Séquentiels

1. Introduction



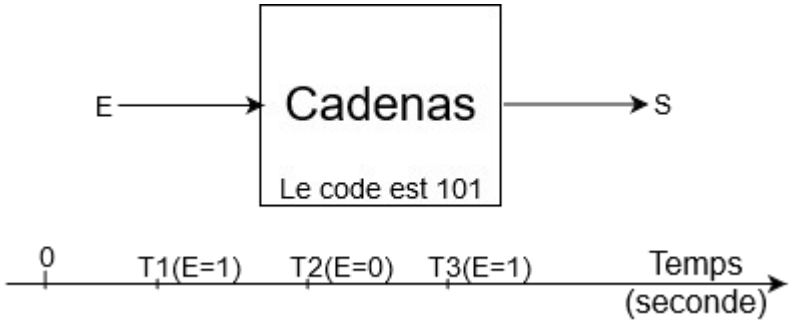
Remarque : Tout au long de ce cours l'appellation Circuits Séquentiels (ou CS) implicitement désigne les Circuits Séquentiels Synchrones.

2. Définition

Un Circuit Séquentiel est un circuit numérique dans lequel sa sortie ne dépend pas seulement que des entrées, comme pour les circuits combinatoires, mais aussi de l'historique des entrées précédentes.

Exemple :

On prend l'exemple du cadenas électronique, mais cette fois-ci au-lieu d'avoir plusieurs entrées comme pour celle du Circuit Combinatoire, elle n'utilise qu'une seule entrée. On peut voir la représentation sur le schéma.



Le code 101 doit être entré chiffre par chiffre à travers le temps. Si on est au 3-ième chiffre le Circuit Séquentiel doit savoir que les 2 chiffres précédents sont 1 suivi de 0 pour ouvrir, sinon ça reste fermé. Ainsi on dit que le Circuit Séquentiel a une mémoire pour se rappeler de la séquence précédente. Le terme *Séquentiel* viens en rapport à ce comportement.

3. Caractéristiques des Circuits Séquentiels

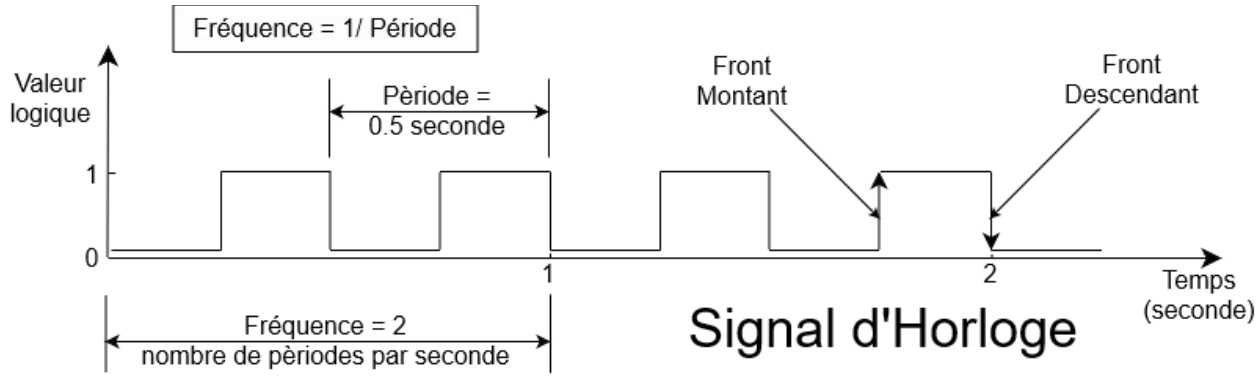
- ➔ Ils sont dynamiques, ils évoluent dans le temps, la sortie d'un Circuit Séquentiel peut changer à travers le temps même si les entrées restent inchangées.
- ➔ Contrairement aux Circuits Combinatoires, ils doivent contenir une boucle. Le plus commun est le retour d'une partie des sorties vers une partie des entrées.
- ➔ Doivent contenir une mémoire pour avoir un aperçu des séquences d'entrées précédentes.
- ➔ La cadence d'évolution d'un Circuit Séquentiel est rythmée par une horloge, généralement une itération (boucle) est effectuée pour chaque tic (impulsion) d'horloge.

Remarque : Il est bon de faire noter qu'il existe aussi un type de Circuits Séquentiels dans lesquels il n'y a pas de boucles, ce sont ce qu'on appelle les *Pipelines*, ils seront étudiés après dans ce chapitre.

<u>L'horloge</u> :

- ➔ C'est un signal numérique périodique à base de 0 et de 1, se caractérisant par une *Période* et une *Fréquence* (comme sur la figure).
- ➔ Le signal est généré par des composants électroniques numériques souvent appelés aussi Horloges.
- ➔ Le signal permet de rythmer la cadence d'opération d'un Circuit Séquentiel.

Le schéma représente l'évolution d'un signal d'horloge dans le temps.

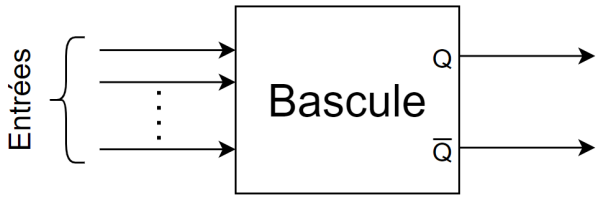


- ➔ Généralement, c'est le front montant dans le signal qui représente le tic d'horloge, indiquant au Circuit Séquentiel de terminer l'opération actuelle et de passer à l'opération suivante.
- ➔ Une seule opération s'effectue par le Circuit Séquentiel sur une période d'horloge, ainsi la période est délimitée par 2 fronts montants successifs.
- ➔ Tout Circuit Séquentiel doit avoir une entrée d'horloge, souvent nommé *Clock*(*clk*).
- ➔ Contrairement aux autres entrées, l'entrée du signal d'horloge n'est pas une entrée d'informations ou de données.
- ➔ Un processeur avec comme exemple 4 Ghz de fréquence, fait réellement 4 milliards d'opérations par seconde. 4 Ghz c'est la fréquence de son horloge.

4. Les mémoires en Bascules (Latch et FlipFlop)

Pour comprendre la technologie de mémoire utilisée dans les Circuits Numériques, appelée aussi *Bascules*, on doit parcourir les étapes d'évolution de la technologie à travers son histoire. Cette évolution est grossièrement déclinée en 4 étapes :

1. Les Bi-stables
2. Les RS-Latch
3. Les D-Latch
4. Les D-FlipFlop

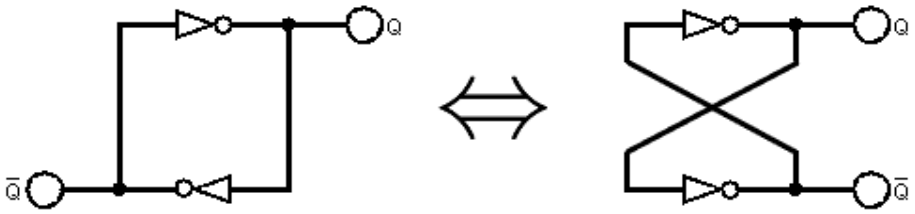


- ➔ Tous les 4 circuits en haut sont des circuits élémentaires appelés cellule-mémoire, ils représentent des circuits pour sauvegarder ou mémoriser 1 seul bit, ils peuvent mémoriser la valeur 0 ou 1.
- ➔ Les sorties des cellules-mémoires indiquent la valeur sauvegardée à l'intérieur de la cellule, la valeur est dans Q ($\overline{Q}$ est toujours l'inverse de Q).
- ➔ Les entrées permettent d'entrer une nouvelle valeur à mémoriser ou de commander la cellule de mémoriser la valeur actuelle.

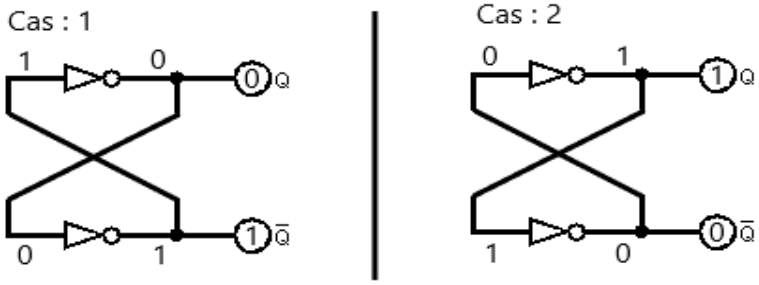
4.1. Bi-stable

- ➔ Un circuit Bi-stable est formé suivant le schéma en bas, où le circuit forme un cycle dans lequel la sortie de la 1-ière porte NOT arrive à l'entrée de la 2-ième porte NOT, et la sortie du 2-ième NOT arrive à l'entrée de la 1-ière.
- ➔ Le circuit Bi-stable ne contient pas d'entrées, et contient 2 sorties Q et $\overline{Q}$.

Schéma d'un circuit Bi-stable :



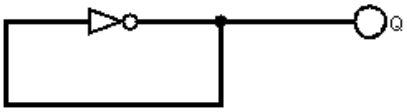
- ➔ Lors de la mise en marche du circuit (Temps = 0 seconde), le circuit peut suivre 2 scénarios possibles, sur le schéma en bas, le circuit peut se stabiliser sur le Cas 1 : Q = 0 et $\overline{Q}$ = 1, ou le Cas 2 : Q = 1 et $\overline{Q}$ = 0, et le signal boucle infiniment d'une façon stable sur l'un des 2 Cas. D'où le nom Bi-stable.
- ➔ Expérimentalement, à la mise en marche, il est impossible de savoir sur quel Cas le circuit va se stabiliser, c'est totalement aléatoire, parfois c'est par rapport à la porte la plus rapide entre les 2.
- ➔ Mais dès que le circuit s'est stabilisé sur un Cas donné, sa configuration ne va plus changer tout au long du temps de fonctionnement. Ce phénomène de stabilité permanente incarne le comportement de la mémoire.



- ➔ Ainsi un circuit Bi-stable représente une cellule-mémoire de 1 bit.
- ➔ Par convention, une cellule-mémoire du type Bascule doit toujours posséder 2 sorties nommées Q et $\overline{Q}$ ($\overline{Q}$ est nécessairement l'inverse de Q).
- ➔ Le circuit Bi-stable est expérimental et non réel, il permet de démontrer le concept et la construction de mémoire à partir de portes logiques.
- ➔ Mais il reste impraticable dans les circuits réels, en raison de l'impossibilité de contrôler le contenu de la mémoire en absence d'entrées sur le circuit.
- ➔ C'est la raison d'être des Bascules dites RS-Latch pour les circuits réels, car le contenu de la cellule peut être modifié à partir de ses entrées.

Remarque 1: Il existe un 3-ième État à part les 2 États de stabilité mentionnés, c'est un État appelé *méta-stable*, sur lequel une valeur au milieu de 0 et 5 Volts (proche de 2,5 Volts) se propageant en boucle dans le circuit, lui attribuant une configuration indéfinie interdite. Quoique la probabilité pour qu'un Bi-stable tombe dans l'État *méta-stable* est très faible.

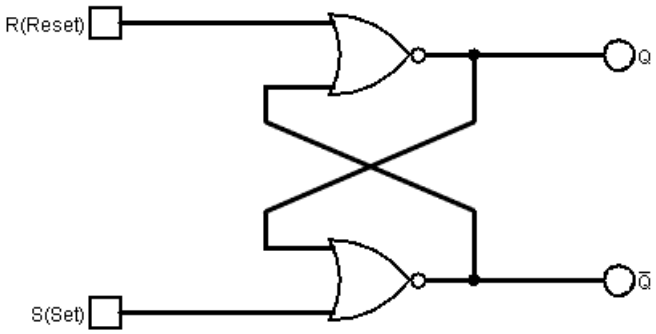
Remarque 2: Pas tous les circuits en cycle tombent dans des États stables, d'ailleurs c'est souvent le contraire, on peut voir par exemple le circuit simple en bas, il ne va jamais tomber dans un État stable, la sortie Q va alterner indéfiniment entre 0 et 1 pour chaque itération (boucle) du signal. Ce genre de circuit est appelé *Astable*, ils sont généralement utilisés pour produire un signal d'horloge.



Remarque 3: La Bi-stable comme pour la RS-Latch, qui vient juste après, sont des circuits asynchrones, elles ne sont pas contrôlées par une horloge et ils bouclent à la vitesse de l'électricité (proche de la vitesse de la lumière).

4.2. RS-Latch

➔ Le circuit RS-Latch (Latch signifie *croché* en Français) est représenté sur le schéma d'en bas, il est formé par 2 portes NOR croisées, tel que la sortie de l'une est l'entrée de l'autre et vice-versa, et former ainsi un cycle.



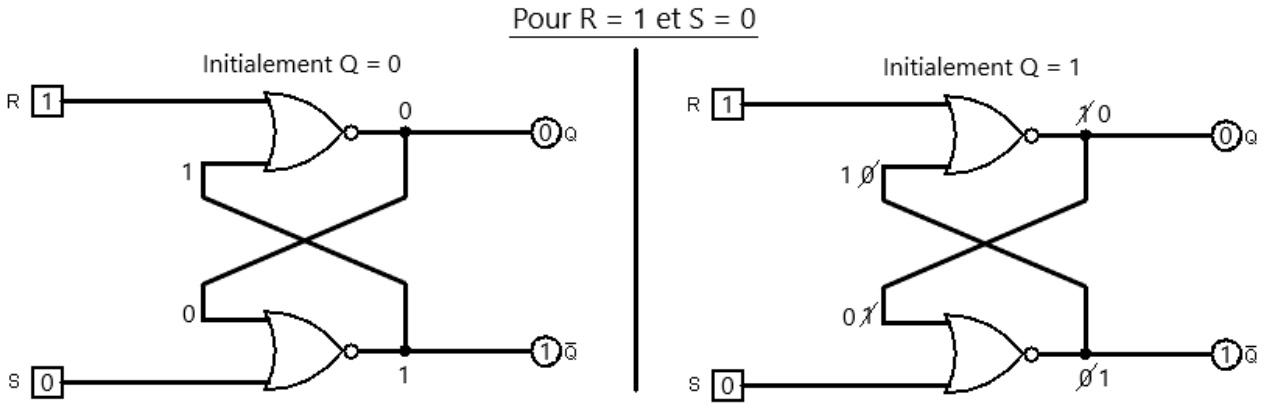
➔ Pour la compréhension du fonctionnement du circuit, une analyse de son comportement sur tous les cas possibles d'entrées est faite sur les 4 schémas en bas.

➔ Les schémas représentent successivement le cas R=1 S=0, R=0 S=1, R=0 S=0, et le cas R=1 S=1. R pour *Reset* (Réinitialiser) et S pour *Set* (Mettre).

➔ Initialement à T=0, lors de la mise en marche du circuit, l'État du circuit est aléatoire

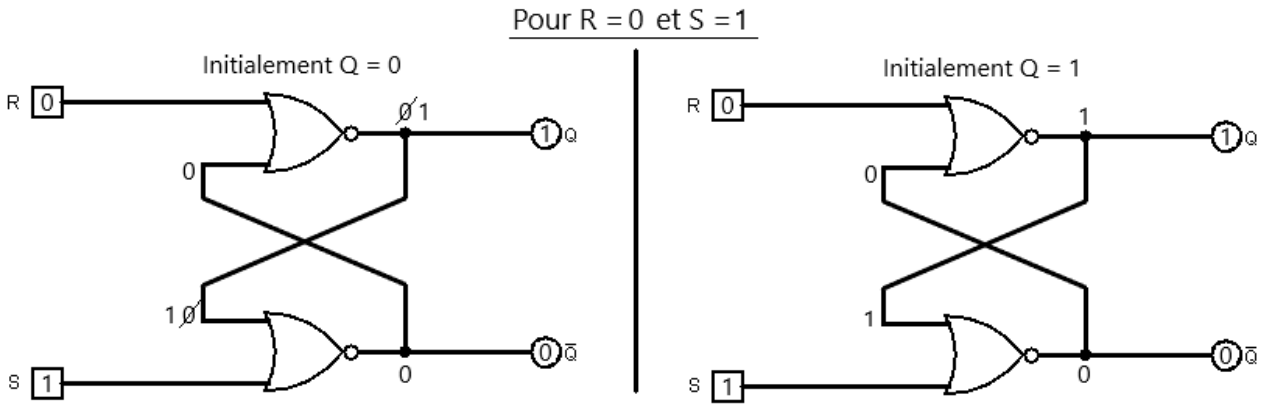
et inconnu, quoique les deux cas de figure possibles sont Q=0 ou Q=1, pour chaque Cas d'entrée, ces 2 cas de figure doivent être étudiés.

➔ Il est aussi possible de suivre les valeurs de $\overline{Q}$ au-lieu de Q pour l'État initial, mais au final ça reviendrait au même.



➔ Pour le Cas R=1 S=0 et initialement Q=0, la valeur de Q=0 entre dans le 2-ième NOR en bas et donne $\overline{Q}=\overline{S+0}=1$. La valeur de $\overline{Q}=1$ entre dans le 1-ier NOR et produit $Q=\overline{R+1}=0$. Le cycle est fait sans que point initial Q change de valeurs, et ça va boucler ainsi infiniment, donc Q et $\overline{Q}$ sont stabilisés respectivement sur les valeurs 0 et 1.

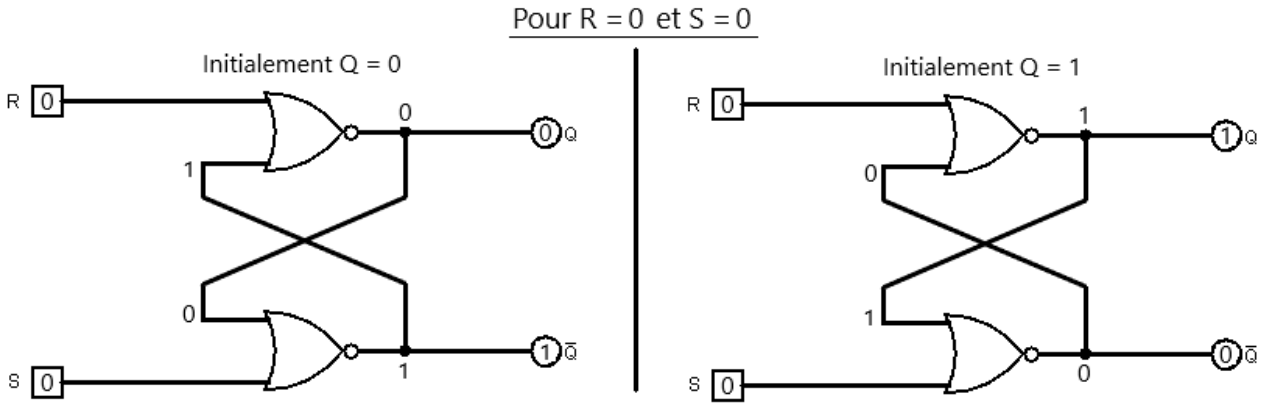
➔ Pour le cas initial Q=1, sa valeur est utilisée dans le 2-ième NOR, donc $\overline{Q}=0$. De la même manière $\overline{Q}$ est utilisé dans le premier NOR et donne cette fois-ci 0 qui va écraser 1 de Q de l'État initial. À ce point le circuit n'est pas encore stabilisé et on doit continuer l'exécution. La 2-ième porte avec Q=0 produit cette fois 1 écrasant l'ancien 0, qui est utilisé à son tour par la 1-ière porte et produit 0. Donc à ce point, à la 2-ième itération Q n'est pas écrasé et reste 0 et le circuit vient de se stabiliser sur Q=0 et $\overline{Q}=1$.



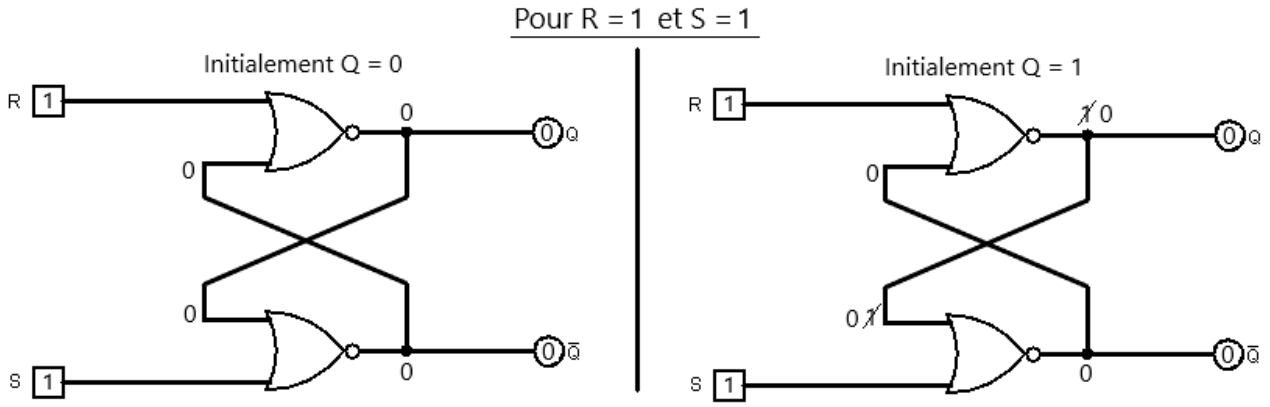
➔ De la même manière, l'exécution des circuits des Cas restants sont exécutés.

➔ Il y a des Cas où le circuit se stabilise dès la 1-ière itération, et parfois ça se stabilise dans la 2-ème itération.

➔ Un circuit se stabilise lorsque l'exécution fait un cycle complet et la valeur du point de départ n'a pas changé.



- ➔ On remarque que pour R=0 S=1, quelque soit l'État initial, le RS-Latch se stabilise sur Q=0 $\overline{Q}$=1. Et inversement pour R=1 S=0, qui lui se stabilise sur Q=1 $\overline{Q}$=0.
- ➔ Pour R=0 S=0 c'est différent, dans le sens où la Bascule se stabilise en rapport avec l'État initial, si l'État initial est Q=0 le circuit se stabilise sur Q=0 $\overline{Q}$=1, et inversement si Q=1 le circuit se stabilise sur Q=1 $\overline{Q}$=0.
- ➔ Une autre manière d'analyser le circuit R=0 S=0 est de généraliser cette observation et de déduire que le circuit conserve la valeur de la boucle précédente Q` juste avant que le circuit passe à R=0 S=0. Si avant de passer à R=0 S=0 le circuit bouclé sur Q`=0, la cellule va conserver cette valeur (Q=0 et $\overline{Q}$=1) pour les cycles à venir. Et inversement si Q`=1, la cellule va sauvegarder Q=1 et $\overline{Q}$=0.
- ➔ Cette dernière propriété est très importante, elle démontre le mécanisme de mémorisation d'une valeur nouvellement changée dans la cellule-mémoire. Ce qui manquait dans une cellule Bi-stable.



- ➔ Pour les circuits R=1 S=1, quelque soit l'État initial, en sortie on aurait Q=0 $\overline{Q}$=0, ce qui représente une sortie incohérente, Q doit toujours être l'inverse de $\overline{Q}$. C'est pour cette raison que cette combinaison d'entrée R=1 S=1 est interdite pour les Bascules RS-Latch, elle les fait entrer dans un État incohérent.
- ➔ On résume les 4 combinaisons d'entrées pour le fonctionnement d'une Bascule RS-Latch sur le tableau en bas. Les modes R=1 S=0 (Reset) et R=0 S=1 (Set) permettent de changer la valeur de la cellule. Le mode R=0 S=0 (Mémoriser) est symbolisé dans le tableau par Q` et $\overline{Q}$`, qui représentent respectivement la valeur de Q $\overline{Q}$ dans l'itération précédente. La combinaison R=1 S=1 est interdite.

R	S	Q	$\overline{Q}$	
1	0	0	1	L'entrée mode Reset remet Q à 0
0	1	1	0	L'entrée mode Set remet Q à 1
0	0	Q`	$\overline{Q}$`	L'entrée Mémoriser (00) conserve Q
1	1	0	0	L'entrée (11) est interdite

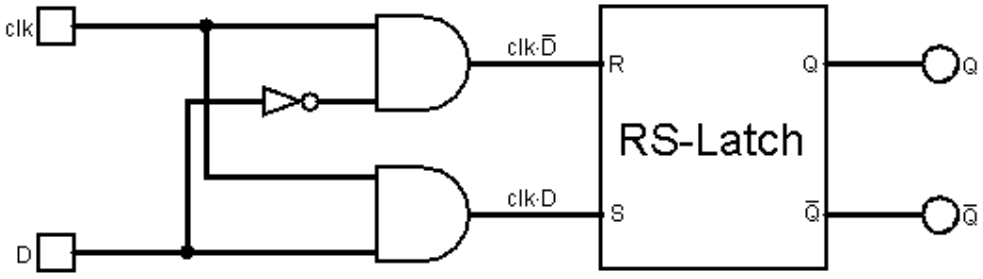
➔ Le Schéma Global d'une bascule RS-Latch est représenté sur la figure en bas.

4.3. D-Latch

➔ Le schéma du circuit de la D-Latch est représenté en bas, sur lequel on peut remarquer qu'une D-Latch utilise une RS-Latch pour son fonctionnement.

➔ Dans le nom de la D-Latch, le D vient de Data (*donnée* en Anglais).

➔ La D-Latch est une amélioration de la RS-Latch, dans le sens où une seule entrée est dédiée pour le changement de valeur de la cellule-mémoire, au-lieu de 2 pour la RS-Latch. La 2-ième entrée est dédiée pour l'horloge (clk : Clock), entrée indispensable pour tout Circuit Séquentiel Synchrone.



➔ Pour comprendre le fonctionnement de la D-Latch, une exécution des 4 possibles combinaisons d'entrées est effectuée et son résultat est exposé dans la table suivante :

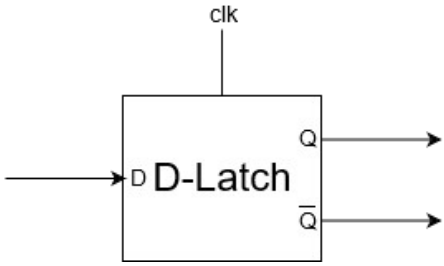
clk	D(Data)	Q	$\overline{Q}$	
0	0	Q`	$\overline{Q}$`	Mémorisation pour clk=0
0	1	Q`	$\overline{Q}$`	Mémorisation pour clk=0
1	0	0	1	Q=0 pour D=0 lorsque clk=1
1	1	1	0	Q=1 pour D=1 lorsque clk=1

➔ Par observation, l'horloge permet de contrôler les 2 modes ; Mémoriser si l'horloge est à 1, et le mode Changement de la valeur de la cellule si l'horloge est à 0, dans ce dernier cas c'est l'entrée D qui change directement la valeur de la cellule.

➔ Lors du mode Mémoriser, la D-Latch est parfois dite *fermée* (ou *opaque*) en raison que la valeur D ne peut pas changer l'intérieur de la cellule. Lors du mode Changement, la bascule est dite *ouverte* (ou *transparente*), la valeur de D peut entrer et modifier l'intérieur de la cellule (qui est toujours accessible sur Q et $\overline{Q}$).

➔ Un autre avantage de la D-Latch par rapport à la RS-Latch c'est qu'il n'y a pas de risque de tomber dans le cas interdit 11 comme dans la bascule RS-Latch.

➔ La représentation du Schéma Global de la bascule D-Latch est sur la figure en bas.

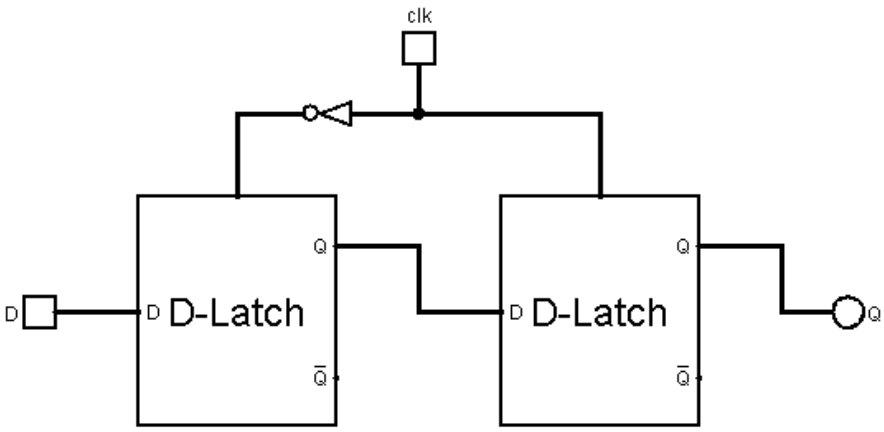


Remarque : Il est aussi très simple de créer une D-Latch avec un seul Mux 2-1, que je vous laisse deviner la solution vous-même.

4.4. D-FlipFlop

➔ Une D-FlipFlop est une Bascule constituée internement par 2 D-Latch mises en série l'une après l'autre, l'une des 2 est rythmée par l'horloge inversée, comme sur le schéma en bas.

➔ La D-FlipFlop fonctionne d'une manière similaire à la D-Latch, la seule différence c'est que le temps de transparence, de modification de la valeur de la cellule et très réduit, c'est lors du front montant de l'horloge.



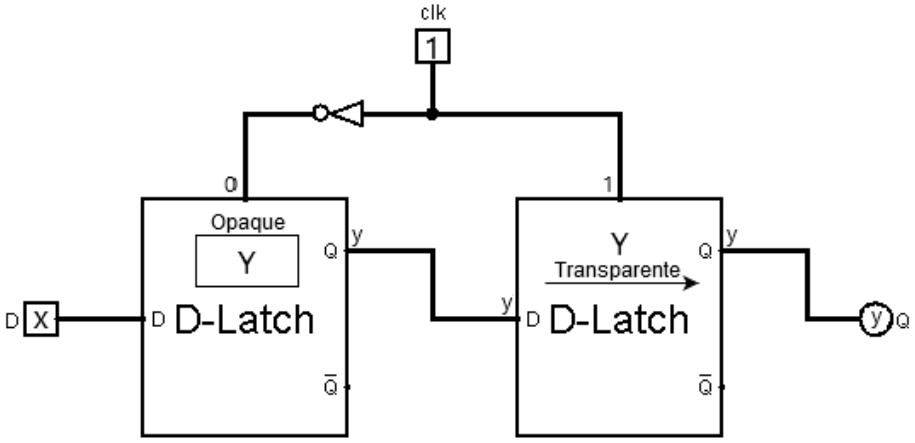
➔ Ainsi, les quelques picosecondes du temps du front montant de l'horloge, qui représente le changement de l'horloge de la valeur 0 vers 1 (voir le schéma de l'horloge en haut), est le seul instant où l'intérieur de la cellule peut être modifié.

➔ Selon le schéma interne de la D-FlipFlop, en raison de l'horloge inversée, les 2 D-Latch fonctionnent toujours en alternance, si l'une est transparente l'autre est opaque, et inversement.

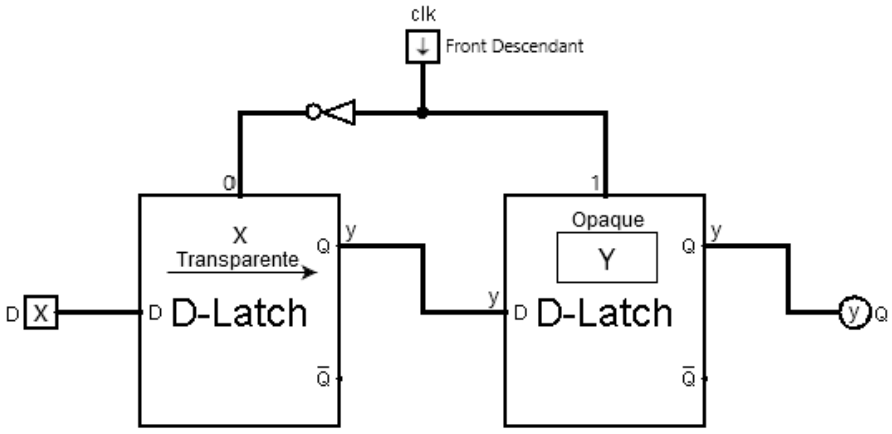
➔ Pour comprendre le fonctionnement de la D-FlipFlop on va faire une exécution sur un exemple traversant les étapes successives dans une période d'horloge, qui sont la valeur 1, le front descendant, la valeur 0, et le front montant.

➔ On suppose sur l'exemple que la cellule contient Y à l'intérieur (valeur binaire), et l'entrée D est fournie avec la valeur X (binaire aussi). Le plus important sur l'exemple est d'observer le moment de l'entrer de la valeur X à l'intérieur de la cellule et d'écraser la valeur Y.

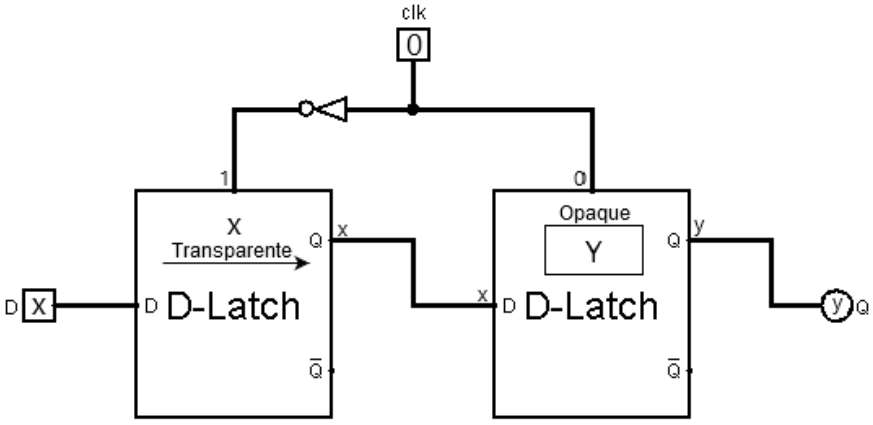
➔ Lorsque l'horloge est à 1 comme sur le schéma en bas, la 1-ière D-Latch (à gauche) est fermée et contient Y, et la 2-ième est transparente, ce qui fait passer Y de la sortie de la 1-ière D-Latch vers la sortie de la 2-ième et ainsi de la D-FlipFlop.



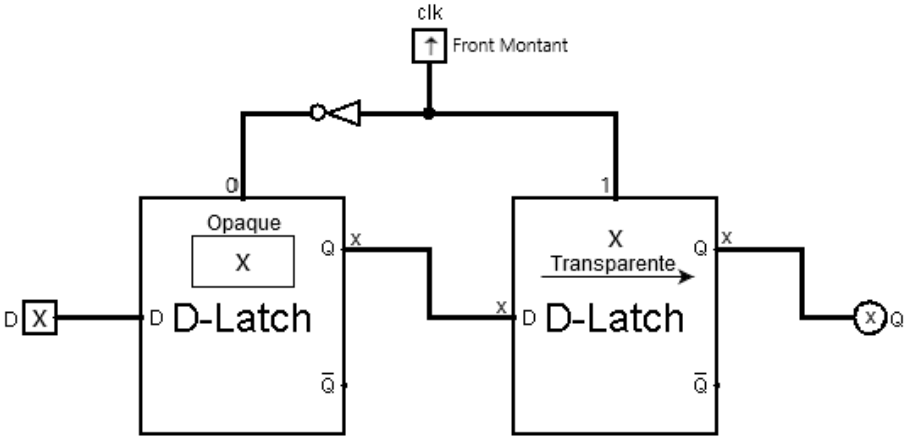
➔ La 2-ième étape dans la période de l'horloge est le front descendant. Sur le schéma avec l'horloge à 0, la 1-ière D-Latch devient transparente et fait passer X de l'entrée D, en <u>même temps</u> la 2-ième D-Latch se referme, puisque son horloge devient 1, et arrive à mémoriser la valeur Y avant qu'elle se transforme en X puisque la 1-ière D-Latch est entrain de s'ouvrir. Et la D-FlipFlop toujours détient la valeur Y.



➔ La 3-ième étape dans la période est la valeur 0. Le schéma en bas montre que lorsque l'horloge est à 0, la 1-ière D-Latch transparente fait passer X de l'entrée D jusqu'à l'entrée de la 2-ième D-Latch, cette dernière est opaque et mémorise la valeur Y anciennement sauvegardée lors du front descendant de l'horloge.



➔ Finalement, lors du front montant et seulement à cet instant, que la sortie Q de la D-FlipFlop (et aussi la sortie de la 2-ième D-Latch) va changer de Y vers X (schéma en bas). À cette étape la valeur X est mémorisée dans la 1-ière D-Latch qui devient opaque, et X passe jusqu'à la sortie la 2-ième D-Latch qui devient transparente.

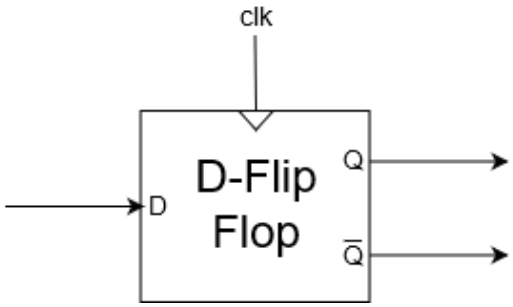


➔ L'horloge après passe à la valeur 1 de la période suivante, dans lequel X ne va plus changer, et ces 4 étapes vont se répéter pour les valeurs suivantes de l'horloge.

➔ On conclusion, le seul moment où la D-FlipFlop peut changer, c'est lors du front montant, le reste de la période d'horloge elle mémorise la valeur à l'intérieur.

➔ Le terme FlipFlop en Anglais signifie *claquette* faisant référence à l'alternance entre Opaque et Transparence des 2 D-Latch.

➔ Le Schéma Global de la D-FlipFlop est sur le schéma en bas. À noter le triangle dans l'entrée de l'horloge, il symbolise l'utilisation du front montant sur le signal de cette dernière.



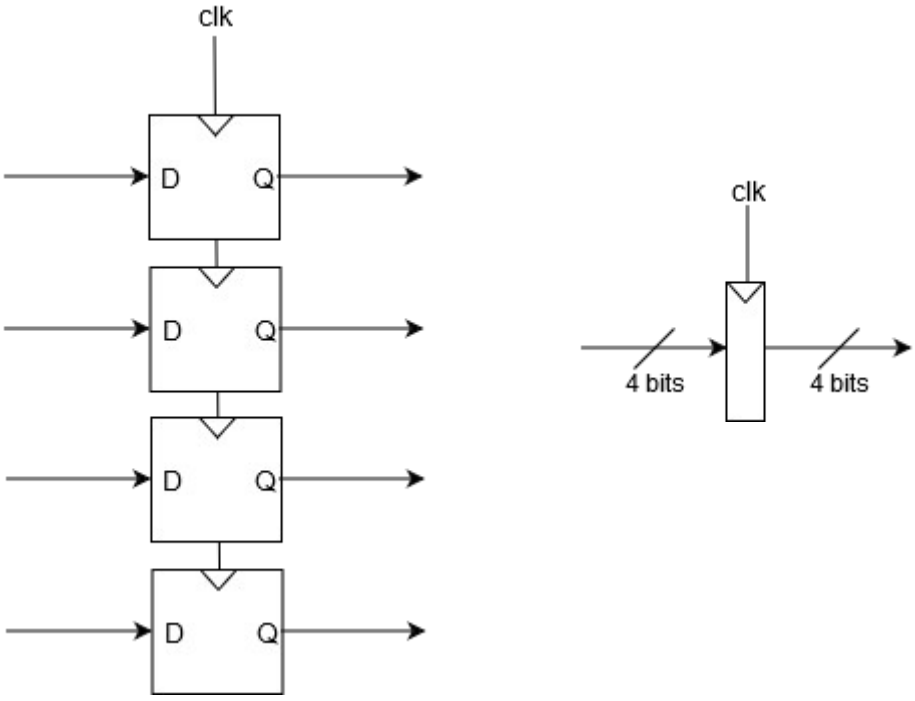
Remarque 1: La majeure partie des circuits intégrés actuels (Processeurs, GPU, Unités d'Entrées/Sorties...) utilisent les D-FlipFlop comme mémoire d'information à l'intérieur, mine de rien il est possible de trouver des circuits numériques qui utilisent les RS-Latch et D-Latch, et c'est particulièrement vrai pour les anciens systèmes. Le Bi-stable est un circuit théorique non pratique, il a pour vocation la démonstration du concept de mémoire.

Remarque 2: Il ne faut pas confondre les mémoires à technologie de Bascule avec les mémoires comme la RAM, la ROM, ou les mémoires de stockage externes; Disque Dur, CD-Rom, Mémoire Flash...etc, qui eux utilisent des technologies différentes.

4.5. Registres

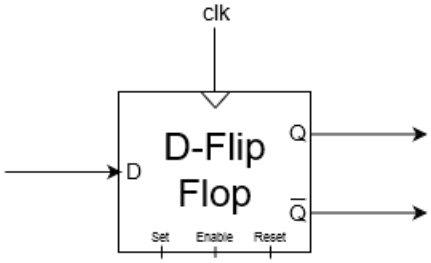
Les Registres représentent un moyen de mémoriser un mot d'information (ou mot mémoire) sur plusieurs bits; 4 bits, 8 bits, 16 bits, 32 bits...etc, en utilisant un tableau de plusieurs D-FlipFlop synchronisées avec le même signal d'horloge.

Exemple : un Registre de 4 bits est réalisé et représenté comme suite :



4.6. Enable, Set et Reset

Il est possible de trouver sur des D-FlipFlop des entrées de commandes additionnelles, telque *Set*, *Reset* et *Enable,* comme c'est représenté sur la figure en bas, ils permettent d'avoir plus de contrôle sur la D-FlipFlop.



- ➔ Le Enable (*Activer* en Français) permet avec 1 d'activer ou avec 0 de désactiver une FlipFlop.
- ➔ Si la FlipFlop est activée elle fonctionne normalement, dans le cas contraire, la désactiver implique le gel et la conservation de la donnée, et ainsi l'impossibilité de modifier la valeur de la cellule lors du front montant de l'horloge.
- ➔ Le Set et le Reset s'ils sont mis à 1, permettent respectivement de modifier la valeur interne de la cellule à 1 et à 0 (Set à 1 et Reset à 0).
- ➔ 2 procédés sont possibles pour appliquer le Set et le Reset; Synchrone et Asynchrone.
- ➔ Un Set/Reset Synchrone implique une modification à 1/0 de la cellule que lors du front montant de l'horloge, tandis que l'Asynchrone la modification se fasse instantanément, à n'importe quel moment de la période d'horloge.

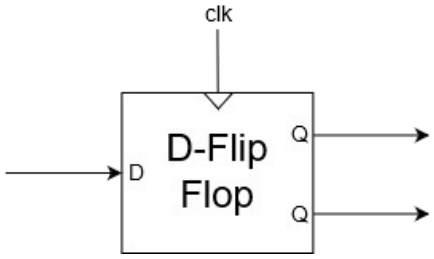
4.7. Autres types de FlipFlop

Malgré que la D-FlipFlop reste la plus connue et la plus utilisée dans le domaine de l'électronique numérique, il existe des types de FlipFlop qui sont bien plus adaptés pour certaines situations particulières.

<u>D-FlipFlop :</u>

- ➔ La plus connue et la plus facile à utiliser.

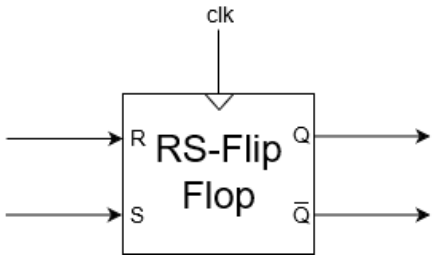
D	Q	$\overline{Q}$
0	0	1
1	1	0



RS-FlipFlop :

- ➔ Prend la même logique d'utilisation que l'RS-Latch.
- ➔ À ne pas confondre avec l'RS-Latch.

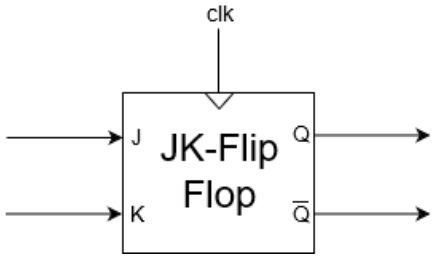
R	S	Q	$\overline{Q}$
0	0	Q`	$\overline{Q}$`
0	1	1	0
1	0	0	1
1	1	-	-



JK-FlipFlop :

- ➔ Semblable à l'RS-FlipFlop dans sa logique d'utilisation, mais en plus elle offre une correction du cas interdit (11) en inversion le contenu sauvegardé de la cellule.
- ➔ Les lettres JK non pas de signification particulière, c'est juste la suite dans l'alphabet des 2 lettres JK.
- ➔ Mais dans le domaine, pour se rappeler de leurs utilisations, K est baptisée *Kill* qui tue le 1 et devient 0, et J *Jump* qui saute du 0 vers 1.

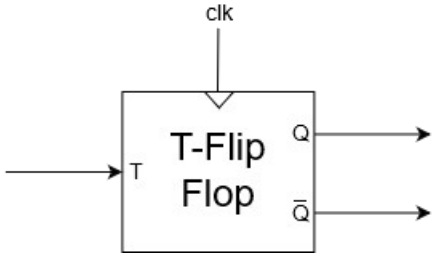
J	K	Q	$\overline{Q}$
0	0	Q`	$\overline{Q}$`
0	1	0	1
1	0	1	0
1	1	$\overline{Q}$`	Q`



T-FlipFlop :

- ➔ T pour *Trigger* (déclencheur en Anglais).
- ➔ Elle a une seule entrée, si elle est mise à 1 elle va inverser la valeur sauvegardée.
- ➔ Pour contrôler le contenu initial de la T-FlipFlip, les entrées Set et Reset sont généralement utilisées.

T	Q	$\overline{Q}$
0	Q`	$\overline{Q}$`
1	$\overline{Q}$`	Q`



Remarque importante: Les tables de vérités décrites ici ne sont applicables que lors du

front montant de l'horloge, le reste de la période la FlipFlop sauvegarde sa valeur. D'où l'utilisation du symbole triangle dans les 2 traits de séparation dans ces Tables de Vérité.

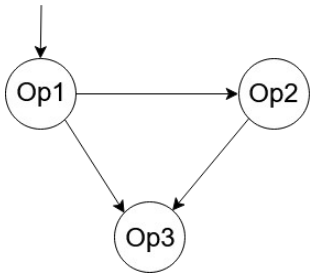
Remarque : Il existe énormément de manières différentes, en utilisant les portes logiques, permettant d'implémenter les Latch et les FlipFlop.

5. Spécification d'un Circuit Séquentiel

La description de fonctionnement d'un Circuit Séquentiel est assurée formellement par ce qu'on appel mathématiquement un Automate fini. Le terme Automate est proche du terme automatique et ce n'est pas un hasard, l'Automate permet de modéliser le fonctionnement des machines automatiques le plus souvent commandées par un Circuit Séquentiel.

Exemple :

On suppose une machine qui peut effectuer 3 opérations différentes, chaque opération peut être commandée par un Circuit Combinatoire différent. Dans un processus de fabrication d'un produit par la machine, l'opération 1 doit toujours être la 1-ière opération à effectuer, l'opération 2 doit suivre l'opération 1, l'opération 3 peut suivre l'opération 1 ou opération 2. On peut facilement modéliser la séquence des opérations par un Automate comme suite :



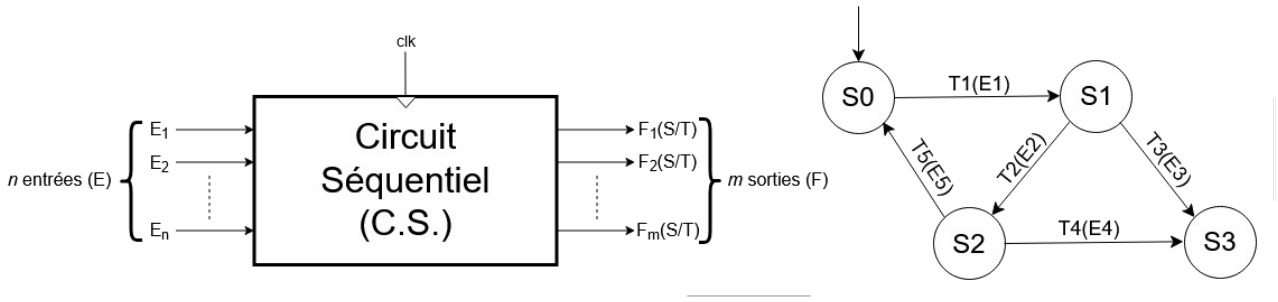
La commande de la machine peut être manuelle, et c'est à l'humain de s'assurer que le séquencement des opérations est correct, mais si on voulait un fonctionnement automatique de la machine, une utilisation d'un Circuit Séquentiel qui doit remplacer les 3 Circuits Combinatoires est à envisageable. Le Circuit Séquentiel doit avoir comme entrées des capteurs lui permettant de savoir la fin d'une opération, et une entrée de l'utilisateur pour choisir la séquence voulue; op1, op2, op3 ou op1, op3.

5.1. Définition d'un Automate :

L'Automate Fini (*FSM* : *Finite State Machine* en anglais) est un modèle mathématique abstrait très utilisé principalement dans le domaine de l'ingénierie, ça permet de décrire un système sur différents États et son passage d'un État à l'autre. une définition des Automates doit respecter les quelques points suivants :

- L'Automate est un graphe dont les nœuds sont appelés *États* (*S* : State) et les liens appelés Transitions (T). Comme sur le schéma en bas.
- L'*État* ne représente pas un composant du système mais le système tout entier dans un État donné **ex** : Si une voiture est un système, sa roue n'est un État mais la voiture tout entière, si le moteur est arrêté la voiture est dans l'État Arrêté.
- La *Transition* représente une condition ou un événement qui fait passer le système d'un État à un autre **ex** : Tourner la clé de la voiture est l'événement qui fait passer la voiture de l'État Arrêtée vers l'État Démarré.
- L'Automate fonctionne de telle-sort que le système reçoit différents événements, ces événements déclenchent les Transitions adéquates et le système passe d'un État à l'autre pour chaque événement d'une façon perpétuelle.
- Le système ne peut pas être dans 2 États ou plus à un instant donné, Quelque soit le temps de marche, un seul État est actif à la fois sur tout l'Automate.

Le schéma à gauche représente le Schéma Global d'un Circuit Séquentiel, et celui à droite l'Automate décrivant la logique de fonctionnement interne du Circuit Séquentiel :



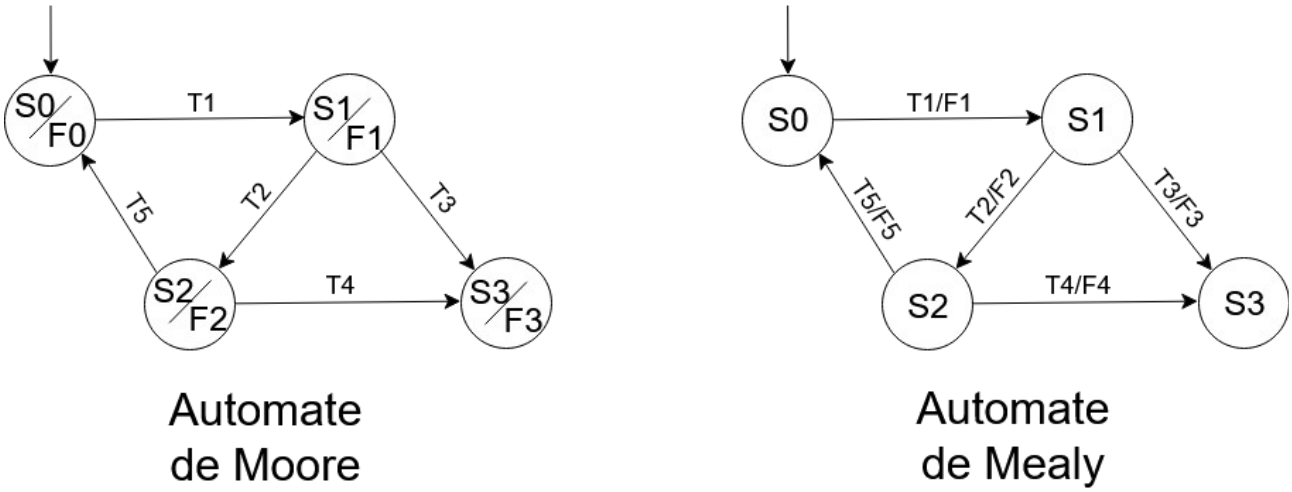
- La Transition orpheline en haut de S0, indique que S0 est l'État initial, à T0 (Temps = 0) le système démarre à partir de cette État.
- Normalement les Automates possèdent un État final, dans lequel l'Automate et le Système s'arrête, mais les Circuits Séquentiels sont supposés ne jamais s'arrêter.
- La Transition d'un État à l'autre est déclenchée par une valeur E (sur *n* bits) à l'entrée du circuit. Les valeurs sur les entrées du circuit sont les événements ou conditions de Transition.

Remarque : Il ne faut pas confondre E1 avec E_1 (sur le schéma), le premier c'est une combinaison de *n* bits sur l'entrée du circuit, le deuxième c'est le premier bit de cette combinaison.

5.2. Sorties des Circuits Séquentiels :

- La sortie d'un Circuit Combinatoire est une fonction de ses entrées, par contre la sortie d'un Circuit Séquentiel peut prendre 2 formes sur 2 modèles d'Automates différents, Moore et Mealy, comme sur le diagramme en bas.

Figure représentant la sortie F sur un Automate de Moore à gauche, et la sortie sur un Automate de Mealy à droite.



- La sortie F sur l'Automate de Moore est représentée par /F dans le cercle du nœud du graphe, chaque sortie est une fonction directe F(S) avec l'État sur lequel se trouve le système.
- Cependant une sortie F sur l'Automate de Mealy est représentée par /F sur les flèches de Transitions du graphe, dans ce cas la sortie est une fonction directe F(T) de la Transition prise par le système.
- En résumé la sortie dans un Automate de Moore est en rapport avec l'État sur lequel se trouve le système, et la sortie dans un Automate de Mealy est en relation avec la Transition prise par le système pour passer d'un État à l'autre.

5.3. Table de Transition d'un Automate :

Tout Automate peut être représenté par une table appelée Table de Transition, définie par les quelques points suivants :

- Une Table de Transition est une collection de toutes les Transitions appartenant à un l'Automate, chaque ligne dans la table représente une Transition T. Comme sur le tableau en bas.
- Chaque Transition pour qu'elle soit parcourue dans un Automate a besoin de 2 informations; l'État actuel et l'Événement enclencheur (Entrée/Condition).
- Ainsi la Table de Transition définie mathématiquement les Transitions comme une fonction (ou application), décrite par T(S,E) = S` (S` est l'État suivant).
- Autrement dit, la Table de Transition a en entrée l'État actuel et l'Entrée E (Événement/Condition) des Transitions, et produit en sortie l'État suivant.
- La Table de Transition permet entre autres de représenter l'Automate sous une forme que le Circuit Séquentiel peut l'exploiter électroniquement pour implémenter cet Automate.

La Table de Transition de l'Automate de la figure en haut :

Transitions	État actuel	Entrée	État suivant
T1	S0	E1	S1
T2	S1	E2	S2
T3	S1	E3	S3
T4	S2	E4	S3
T5	S2	E5	S0

Remarque : Un Circuit Séquentiel étant défini comme un circuit qui produit ses sorties en fonctions des entrées, plus l'historique des séquences précédentes d'entrées, effectivement c'est l'État actuel qui correspond à cette séquence, sachant que pour arriver à l'État actuel il fallait passer par une séquence préalable d'entrées. Autrement dit, l'État actuel mémorise un format de séquence précédemment entré.

5.4. Exécution d'Automate par un Circuit Séquentiel

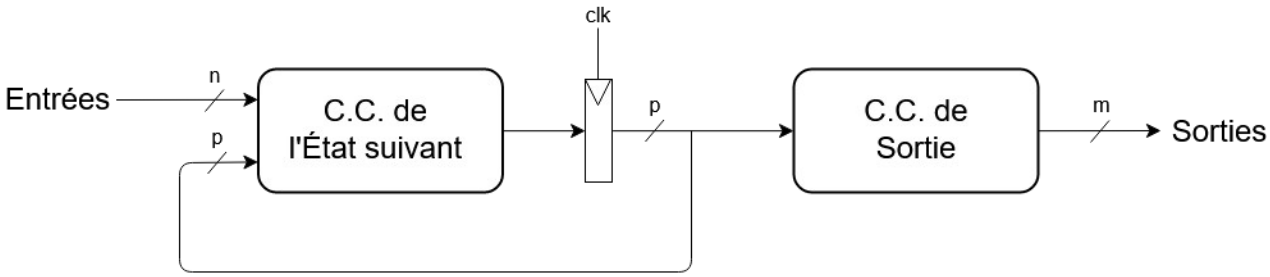
Un Automate spécifiant un Circuit Séquentiel s'exécute suivant les quelques règles suivantes :

- L'horloge est obligatoire pour un Circuit Séquentiel, comme il a été représenté par un triangle sur le schéma global en haut.
- Pour chaque période d'horloge Le système reste établi sur un État donné, et passe à l'État suivant à la fin de celle-ci lors du front montant de l'horloge.
- La Transition d'un État à l'autre se fait lors du front montant de l'horloge.
- Ainsi, la sortie pour un Automate de Moore perdure tout au long de la période, pratiquement elle est récupérée lors du front montant de la fin de période.
- Tendit que la sortie pour un Automate de Maily est acquise lors du front montant de début de période de l'État actuelle.
- L'entrée E pour un Circuit Séquentiel est toujours lue lors du front montant de début de période.

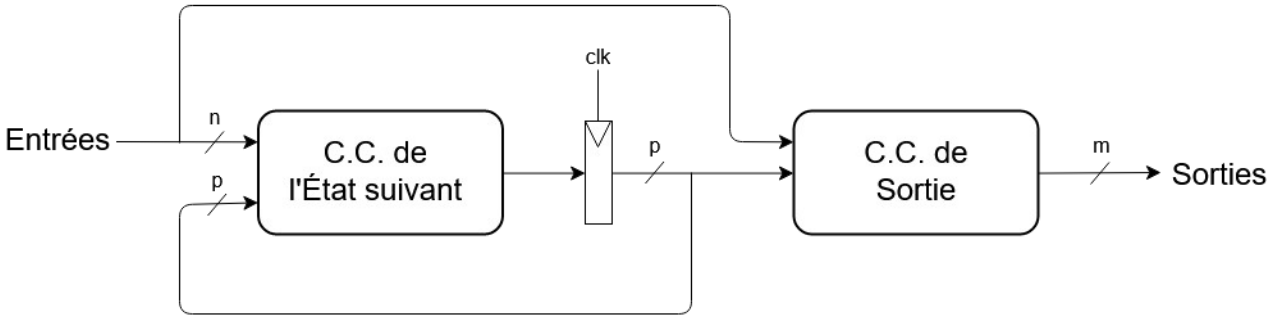
5.5. Modèles logiques de construction d'Automates

Le circuit logique permettant l'implémentation des Circuits Séquentiels à base d'Automate est représenté sur l'une des 2 figures en bas. La première est La Machine de Moore pour l'Automate de Moore et la deuxième Machine de Mealy pour l'Automate de Mealy.

La figure suivante illustre le diagramme de la Machine de Moore :



Une figure du diagramme de la Machine de Mealy est comme suite :



La Machine est choisie selon l'Automate adéquat, et le processus de conception d'un Circuit Séquentiel sera limité à la recherche des 3 inconnus suivants représentés sur le diagramme :

- La valeur de *p* (la taille en bits du Registre).
- Le Circuit Combinatoire de l'État suivant.
- Le Circuit Combinatoire de la Sortie.

Résoudre ces 3 inconnus revient à dessiner le logigramme du Circuit Séquentiel, pour le faire une méthode appelée *Méthode à 7 étapes* est exécutée.

6. Méthode conventionnelle de construction des Circuits Séquentiels

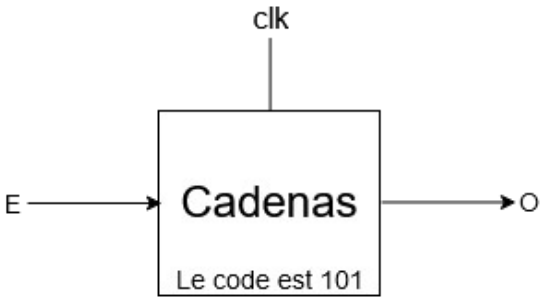
Le processus de construction des Circuits Séquentiels s'appuie sur la Méthode à 7 étapes, et ça implique l'application en séquence des 7 étapes suivantes :

- Schéma global
- Automate
- Table de Transition
- Encodage des États et Table des Sorties
- Table de Transition Encodée
- Formules Logiques
- Logigramme

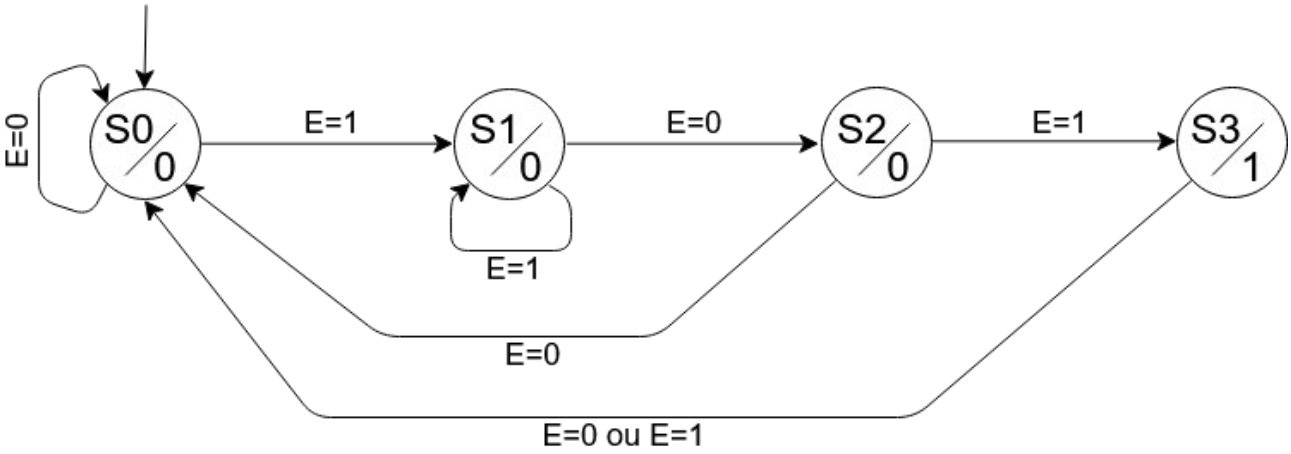
Exemple 1: (Machine de Moore)

L'exemple est celui du cadenas électronique vu précédemment dans ce chapitre.

<u>Étape 1</u> : Schéma global



<u>Étape 2</u> : Automate



Remarque 1: L'Automate est pratiquement l'étape la plus délicate et importante parmi les 7 étapes de construction d'un Circuit Séquentiel.

Remarque 2: La création d'un Automate consiste généralement de donner une signification pour chaque État, par exemple S0 dans notre Automate signifie que le circuit n'a reconnu aucun chiffre du code correct, alors que S1 signifie que le circuit reconnaît le premier chiffre 1 qui est correct, S2 que la séquence 10 entrée jusque-là est correcte, et S3 que tout le code 101 est correct.

Remarque 3: Les événements représentent la valeur entrée dans E (1 ou 0), mais le plus optimal lorsque le circuit contient plusieurs entrées est d'exprimer la condition de Transition par une formule logique avec ses variables d'entrées. **ex** : une Transition avec la condition $\overline{E}_2 \cdot E_1 + \overline{E}_0$, ne peut être prise que lorsque ($\overline{E}_2$=0 et E_1=1) ou E_0=0.

Remarque 4: La solution de l'Automate reste aussi correcte en modifiant la Transition S3 avec E=1 allant vers l'État S1 au-lieu de S0. Ça veut dire que le 1 juste après la reconnaissance du code est prise comme le premier 1 reconnu pour le code suivant.

<u>Étape 3</u> : Table de Transition

État actuel	Entrée (E)	État suivant
S0	0	S0
S0	1	S1
S1	0	S2
S1	1	S1
S2	0	S0
S2	1	S3
S3	0	S0
S3	1	S0

<u>Étape 4</u> : Encodage des États et Table des Sorties

États	S_1	S_0	O
S0	0	0	0
S1	0	1	0
S2	1	0	0
S3	1	1	1

Pour encoder 4 États il en faut 2 bits; S_1 et S_0.

Remarque 1: Rappel que pour connaître le nombre de bits nécessaires pour encoder N valeurs en binaire est selon la formule $Log_2(N)$, avec un arrondissement en haut.

Remarque 2: Attention à ne pas confondre entre les qualificatifs comme S1 et S_1, le premier désigne l'État S1 dans l'Automate, le deuxième désigne le deuxième bit dans l'encodage des États.

Remarque 3: Trouver le nombre de bits pour l'encodage des États revient à élucider le premier inconnu p (p=2), ainsi la taille du Registre pour cette Machine est de 2 bits.

Remarque 4: Le Registre dans les Machines (Moore ou Mealy) a le rôle de mémoriser en binaire l'État dans lequel l'Automate se trouve, et de fournir cette information aux 2 Circuits Combinatoires (État suivant et Sortie).

Remarque 5: La Table des Sorties a comme entrées les 2 colonnes S_1 et S_0, et la colonne O (Output) comme sortie. Effectivement elle représente la Table de Vérité du Circuit Combinatoire de Sortie pour la Machine de Moore.

Remarque 6: Trouver La Table des Sorties revient à trouve le deuxième inconnu, qui est le circuit de Sortie, c'est un Circuit Combinatoire qui a en entrée l'État actuel de l'Automate, et qui produit comme résultat la sortie correspondante à l'État actuel.

<u>Étape 5</u> : Table de Transition Encodée

S_1	S_0	Entrée (E)	$S`_1$	$S`_0$
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	0	0

C'est la même Table de Transition mais avec l'encodage des États

Remarque 1: $S`_1$ et $S`_0$ représentent les bits d'encodage de l'État suivant.

Remarque 2: La Table de Transition Encodée est une Table de Vérité avec comme entrées S_1 ,S_0 et E, et comme sortie $S`_1$ et $S`_0$. Elle représente réellement le troisième inconnu de la Machine de Moore, qui est le Circuit Combinatoire de l'État suivant. Et réellement c'est la table qui représente l'Automate du Circuit Séquentiel.

Remarque 3: Le circuit de l'État suivant selon le diagramme de la Machine de Moore se trouve au cœur de la boucle du Circuit Séquentiel. Il prend 2 types d'informations en entrée; l'État actuel du circuit à partir du Registre, et l'événement (condition de Transition) à partir de l'entrée E, et effectue la Transition en produisant l'État suivant, qui sera sauvegardée dans le Registre lors du prochain front montant de l'horloge.

Remarque 4: Le Circuit Séquentiel passe d'un État à l'autre lors de chaque période ou front montant de l'horloge, c'est le Registre qui mémorise le code de l'État actuel dans lequel se trouve le circuit. Le circuit de l'État suivant est responsable d'effectuer la Transition vers l'État suivant en combinant l'entrée E et l'État actuel. Le Circuit Séquentiel produit une sortie différente en fonction de l'État où se trouve le circuit, c'est le circuit de la Sortie qui est responsable d'accomplir cette fonction entre l'État actuel et la sortie.

Remarque 5: Pour un Circuit Séquentiel Synchrone c'est l'horloge qui rythme la cadence de bouclage sur la Machine (Moore ou Mealy), de même la cadence de passer d'un État à l'autre sur l'Automate.

<u>Étape 6</u> : Formules Logiques

Les Formules Logiques sont les fonctions réduites de l'étude des 2 Circuits Combinatoires, Circuit de Sortie et Circuit de l'État suivant.

Visuellement on peut déduire à partir de la T.d.V. de la Sortie que : $\mathbf{O(S_1,S_0) = S_1 \cdot S_0}$

On doit calculer la Table de Karnaugh pour le Circuit de l'État suivant :

E \ $S_1 S_0$	00	01	11	10
0	0	[1]	0	0
1	0	0	0	[1]

$\mathbf{S`_1(S_1,S_0,E) = \overline{S_1}\cdot S_0\cdot\overline{E} + S_1\cdot\overline{S_0}\cdot E}$

E \ $S_1 S_0$	00	01	11	10
0	0	0	0	0
1	[1]	[1]	0	[1]

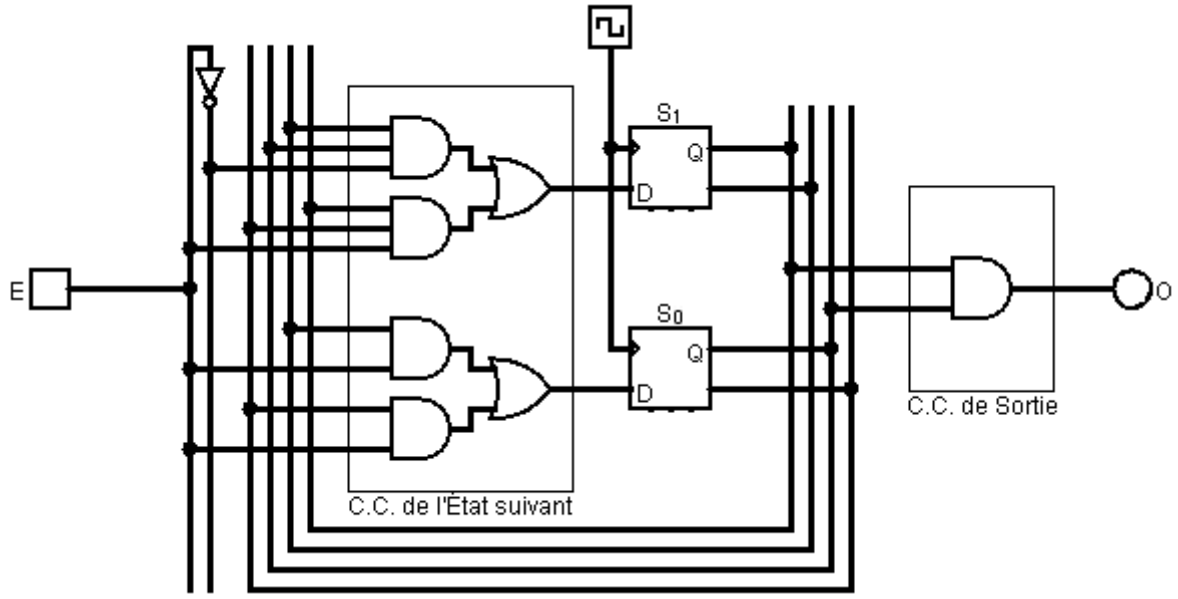
$\mathbf{S`_0(S_1,S_0,E) = \overline{S_1}\cdot E + \overline{S_0}\cdot E}$

Remarque 1: Cette étape est équivalente à l'étape 3 et 4 de la méthode des 5 étapes de construction d'un Circuit Combinatoire, on peut négliger l'étape des Formes Canoniques et se concentrer sur l'étape de la réduction.

Remarque 2: La fonction de $O(S_1,S_0)$ a été déduite directement en raison de sa simplicité inhérente, qui ne nécessite pas le développement d'une Table de Karnaugh.

Remarque 3: Comme la méthode à 5 étapes de construction d'un Circuit Combinatoire, il est possible dans certaines situations de choisir une minimisation avec l'algèbre de Boole ou d'user des traits de réduction pour les cas impossibles pour plus de réduction.

<u>Étape 7</u> : Logigramme



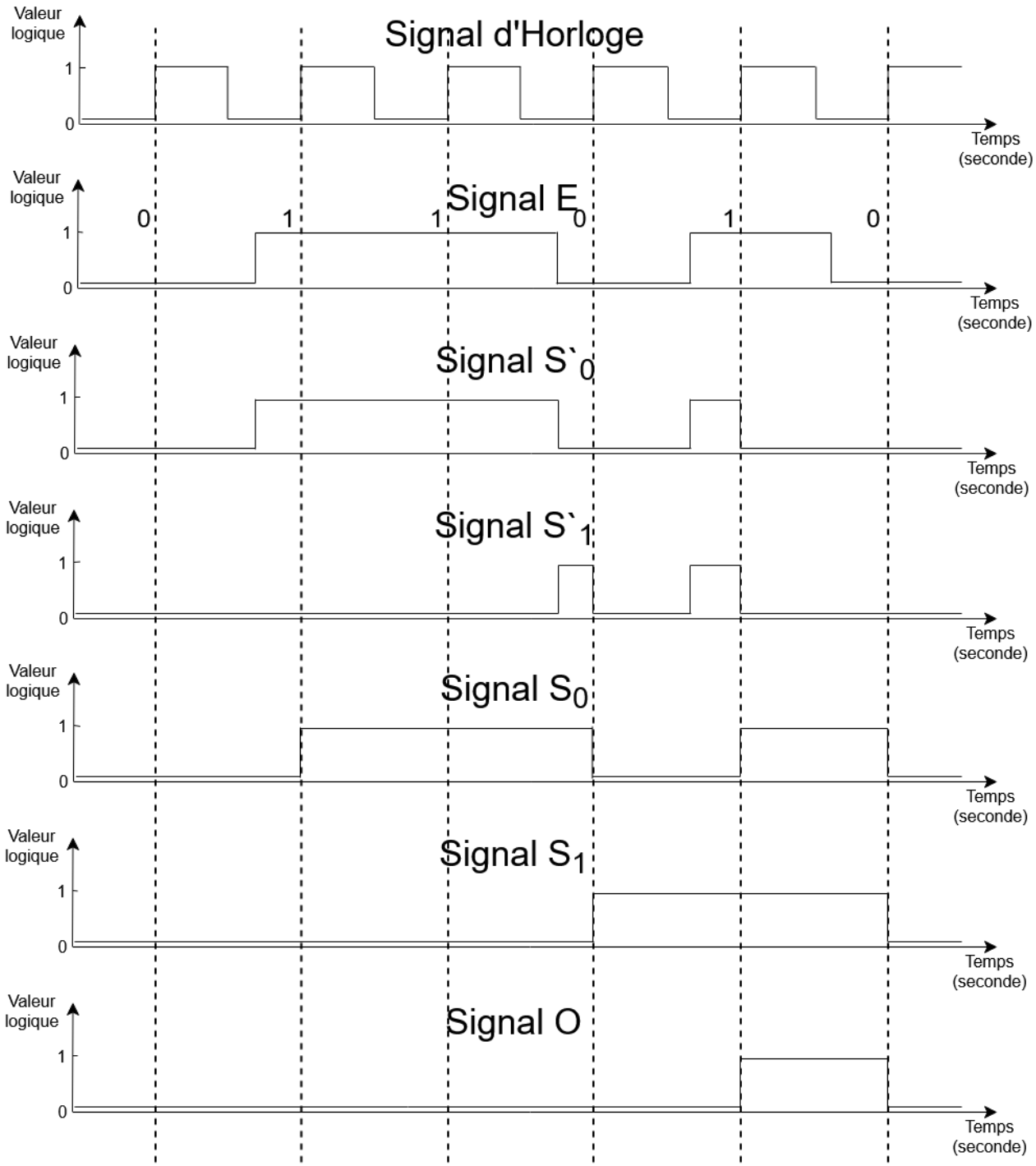
Remarque 1: La construction du Logigramme est calquée sur le schéma de la Machine de Moore, tous les éléments du Logigramme sont disposés dans le même emplacement que dans la Machine.

Remarque 2: Lors de l'exécution sur circuit, il est possible à n'importe quel moment de connaître l'État sur lequel se trouve l'Automate, il suffit de lire le code de l'État dans le Registre avec les cellules S_1 et S_0.

Remarque 3: Les principaux caractéristiques d'un Circuit Séquentiel sont directement visibles sur le Logigramme; la boucle, la mémoire et l'horloge. Le dynamisme est observable lors d'une exécution, c'est visible sur l'exécution en bas.

Remarque 4: La mémoire Registre ne sauvegarde pas directement l'historique de la séquence entrée précédente, mais elle le fait indirectement par le code de l'État actuel, sachant que l'évolution vers un certain État dans un Automate nécessite une forme de séquence entrée.

Exécution : Pour bien comprendre le fonctionnement interne du Circuit Séquentiel, on va faire une exécution sur un scénario d'entrées simples avec comme séquence 011010 (allant de gauche à droite), en suivant sur le chronogramme l'évolution par période d'horloge des signaux $S`_0$, $S`_1$ (les sorties du Circuit C
ombinatoire de l'État suivant), S_0, S_1 (les sorties du Registre) et O (la sortie du Circuit Combinatoire de Sortie et en même temps du Circuit Séquentiel). On doit suivre la propagation du signal à travers les portes logiques dans les 2 Circuits Combinatoires. Il ne faut pas oublier que lors de l'État initial les valeurs dans le Registre sont $S_0 = 0$ et $S_1 = 0$.



Remarque importante 1: Le signal d'entrée (E dans notre cas), doit obligatoirement toujours arriver à l'entrée du circuit avant le front montant de l'horloge, ça donne le temps pour sa propagation dans les Circuits Combinatoires et se conformer au *setup-time* des FlipFlops.

Remarque importante 2: Le Circuit Séquentiel reste stable dans un État donné tout au long de la période, et transite vers un autre État lors des Fronts Montants, ce qui implique que la sortie du circuit en passant par le Circuit Combinatoire de Sortie reste aussi stable tout long de la période, la sortie est recueillie lors du front montant de fin de la période.

Remarque 1: Les FlipFlops sont caractérisées par ce qu'on appelle le *setup-time*, c'est un temps minimum dans lequel le signal dans D doit rester stable avant l'arriver du front montant.

Remarque 2: Il existe des formules concrètes pour le calcul temporel du signal dans les circuits combinatoires et séquentiels, ça appartient au domaine de l'étude temporel des circuits logiques (c'est en dehors de ce cours).

Encodage en One-hot : ou encodage à *1-bit-à-1* en français, est une forme d'encodage des États qui stipule que la valeur *p* est égale au nombre d'États, où chaque État est représenté par un bit unique dans le Registre, à n'importe quel moment il n'y a qu'un seul bit actif (mis-à-1) sur tout le Registre, celui de l'État actuel, les autres bits sont à 0.

Exemple 2: (Encodage en One-hot)

On va refaire l'exemple 1 précédent du cadenas électrique avec l'encodage One-hot. Les 3 premières étapes sont les mêmes, pas besoin de les refaire, on continue à partir de l'étape 4.

Étape 4 : Encodage en One-hot des États et Table des Sorties

États	S_0	S_1	S_2	S_3	O
S0	1	0	0	0	0
S1	0	1	0	0	0
S2	0	0	1	0	0
S3	0	0	0	1	1

Pour encoder 4 États en One-hot il en faut 4 bits.

Remarque 1: La Table des Sortie reste pratiquement la même, sauf pour l'encodage des États qui est différent.

Remarque 2: Réellement la Table est constituée de 16 lignes d'entrées en sachant qu'elle possède 4 variables d'entrées (2^4 = 16 possibilités). Les États encodées avec plus d'un seul 1 (comme 0111 ou 011) ou aucun 1 (comme 0000) sont supposés des États impossibles dans le One-hot. Ils ne sont pas représentés sur la table pour éviter d'avoir une table trop longue.

<u>Étape 5</u> : Table de Transition Encodée en One-hot

S_0	S_1	S_2	S_3	E	$S`_0$	$S`_1$	$S`_2$	$S`_3$
1	0	0	0	0	1	0	0	0
1	0	0	0	1	0	1	0	0
0	1	0	0	0	0	0	1	0
0	1	0	0	1	0	1	0	0
0	0	1	0	0	1	0	0	0
0	0	1	0	1	0	0	0	1
0	0	0	1	0	1	0	0	0
0	0	0	1	1	1	0	0	0

Table de Transition Encodée en One-hot

Remarque : Même remarque ici, la table a comme entrée 5 variables donc 32 lignes, que la plupart ont des très de sorties indéfinies (don't care).

<u>Étape 6</u> : Formules Logiques

Visuellement on peut déduire à partir de la T.d.V. de Sortie que : $\mathbf{O(S_0,S_1,S_2,S_3) = S_3}$

On utilise la minimisation algébrique sur la Table de Transition Encodée en One-hot :

$S`_0(S_0,S_1,S_2,S_3,E) = S_0\cdot\overline{S_1}\cdot\overline{S_2}\cdot\overline{S_3}\cdot\overline{E} + \overline{S_0}\cdot\overline{S_1}\cdot S_2\cdot\overline{S_3}\cdot\overline{E} + \overline{S_0}\cdot\overline{S_1}\cdot\overline{S_2}\cdot S_3\cdot\overline{E} + \overline{S_0}\cdot\overline{S_1}\cdot\overline{S_2}\cdot S_3\cdot E$
$S`_0(S_0,S_1,S_2,S_3,E) = (S_0\cdot\overline{S_1}\cdot\overline{S_2}\cdot\overline{S_3}\cdot\overline{E} + \overline{S_0}\cdot\overline{S_1}\cdot S_2\cdot\overline{S_3}\cdot\overline{E}) + (\overline{S_0}\cdot\overline{S_1}\cdot\overline{S_2}\cdot S_3\cdot\overline{E} + \overline{S_0}\cdot\overline{S_1}\cdot\overline{S_2}\cdot S_3\cdot E)$
$S`_0(S_0,S_1,S_2,S_3,E) = \overline{S_1}\cdot\overline{S_3}\cdot\overline{E}\cdot(S_0\cdot\overline{S_2} + \overline{S_0}\cdot S_2) + \overline{S_0}\cdot\overline{S_1}\cdot\overline{S_2}\cdot S_3$
$\mathbf{S`_0(S_0,S_1,S_2,S_3,E) = \overline{S_1}\cdot\overline{S_3}\cdot\overline{E}\cdot(S_0\oplus S_2) + \overline{S_0}\cdot\overline{S_1}\cdot\overline{S_2}\cdot S_3}$

$S`_1(S_0,S_1,S_2,S_3,E) = S_0\cdot\overline{S_1}\cdot\overline{S_2}\cdot\overline{S_3}\cdot E + \overline{S_0}\cdot S_1\cdot\overline{S_2}\cdot\overline{S_3}\cdot E$
$\mathbf{S`_1(S_0,S_1,S_2,S_3,E) = (S_0\oplus S_1)\cdot\overline{S_2}\cdot\overline{S_3}\cdot E}$

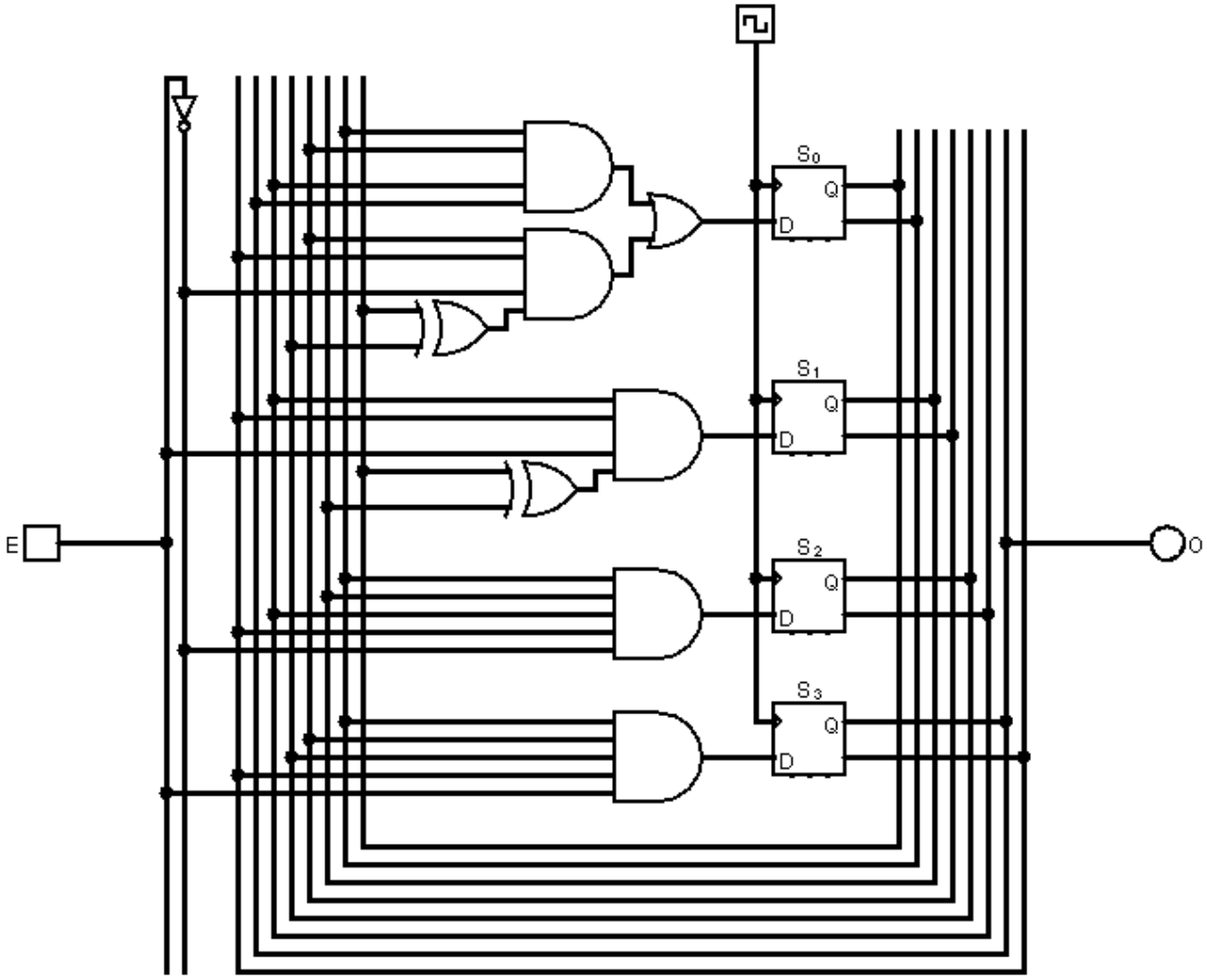
$\mathbf{S`_2(S_0,S_1,S_2,S_3,E) = \overline{S_0}\cdot S_1\cdot\overline{S_2}\cdot\overline{S_3}\cdot\overline{E}}$

$\mathbf{S`_3(S_0,S_1,S_2,S_3,E) = \overline{S_0}\cdot\overline{S_1}\cdot S_2\cdot\overline{S_3}\cdot E}$

Remarque 1: On était obligé de choisir d'utiliser la réduction Algébrique au-lieu des tables de Karnaugh, dans le sens où une minimisation avec une table de Karnaugh sur 5 variables est sensiblement différente et plus difficile (en plus qu'on l'a pas vu en cours).

Remarque 2: La Table de Karnaugh reste applicable pour un nombre de variables pas plus de 6, malgré qu'elle reste plus délicate pour 5 et 6 variables. La méthode de Quine–McCluskey (pas vu en cours) est plus adaptée pour ce genre de situation.

<u>Étape 7</u> : Logigramme



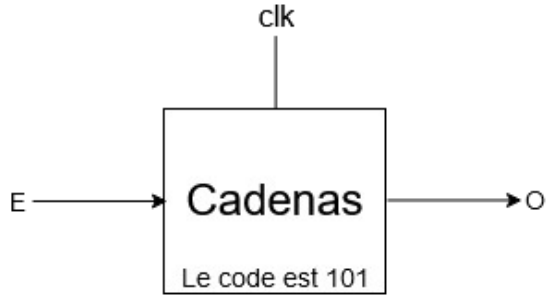
Remarque 1: Malgré que ce ne soit pas le cas pour cet exemple, en règle générale l'encodage One-hot a l'avantage par rapport à l'encodage normal de minimiser le nombre de portes logiques dans les 2 Circuits Combinatoires, en contrepartie il accroît le nombre de cellules-mémoires.

Remarque 2: Il existe aussi une autre variante semblable au One-hot, appelée One-Cold (*1-bit-à-0* en français). C'est le complément de One-hot, ça veut dire le bit représentant l'État est à 0 et les autres à 1, **ex** : l'État S0 est encodé 0111.

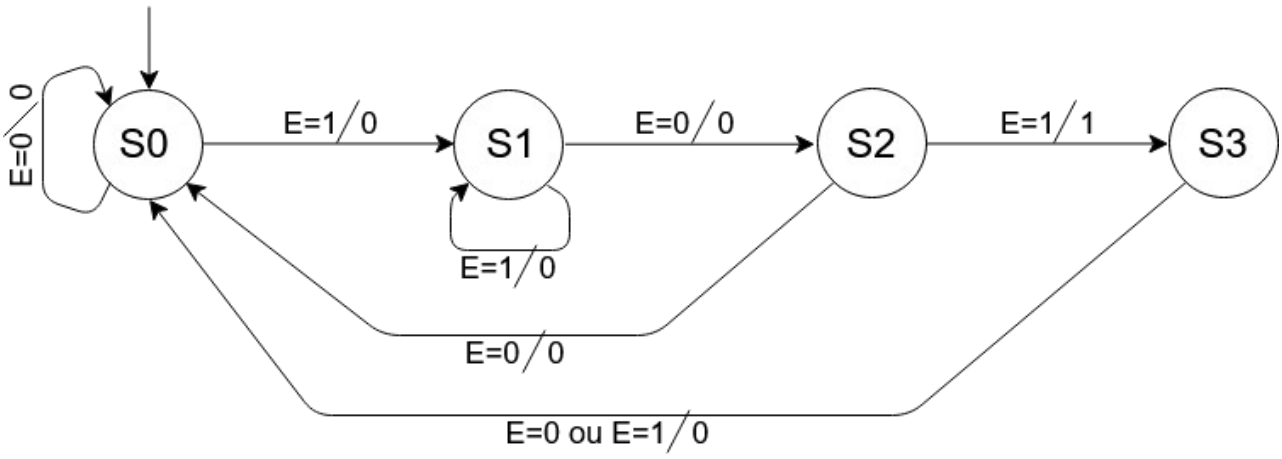
Exemple 3: (la Machine de Mealy)

On va refaire le même exemple du cadenas avec cette fois-ci la Machine de Mealy.

<u>Étape 1</u> : Schéma global



<u>Étape 2</u> : Automate



Remarque : Il n'y a pas de différence entre les 2 Automates de Moore et de Mealy, la seule différence c'est que la sortie dans Moore est dans les États, alors que dans Mealy elle est dans les Transitions.

<u>Étape 3</u> : Table de Transition

État actuel	Entrée (E)	État suivant
S0	0	S0
S0	1	S1
S1	0	S2
S1	1	S1
S2	0	S0
S2	1	S3
S3	0	S0
S3	1	S0

Remarque : La Table de Transition est exactement la même que celle de la Machine de Moore.

<u>Étape 4</u> : Encodage des États et Table des Sorties

États	S_1	S_0	E	O
S0	0	0	0	0
S0	0	0	1	0
S1	0	1	0	0
S1	0	1	1	0
S2	1	0	0	0
S2	1	0	1	1
S3	1	1	0	0
S3	1	1	1	0

Table d'encodage et de Sortie pour la Machine de Mealy

Remarque : En observant la différence entre La Machine de Moore et celle de Mealy sur le diagramme des 2 machines (page 19), la seule différence entre les 2 c'est dans leurs circuits de Sortie. La machine de Moore n'a besoin que de l'État actuel en provenance du Registre pour produire la sortie, alors que la Machine de Mealy a besoin pour émettre sa sortie lors de la Transition des 2 informations; l'entrée E et l'État actuel. Sur la base de ces 2 informations le circuit combinatoire de Sortie de Mealy peut reconnaître lors du front montant la Transition et ainsi la sortie correspondante.

<u>Étape 5</u> : Table de Transition Encodée

S_1	S_0	Entrée (E)	$S`_1$	$S`_0$
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	0	0

Remarque : La Table de Transition Encodée est exactement la même que celle de la Machine de Moore.

<u>Étape 6</u> : Formules Logiques

Le Circuit de l'État suivant produit les mêmes formules que celles de la Machine de Moore, sachant que la Table des Transition Encodée n'a pas changé.

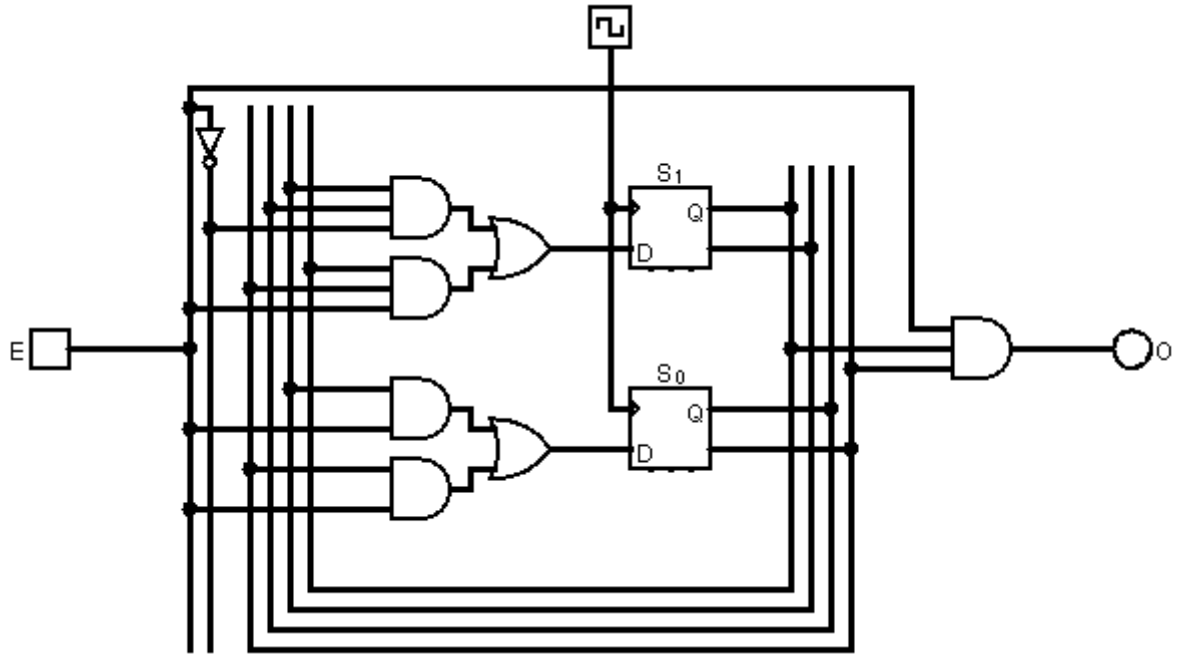
$$S`_0(S_1,S_0,E) = \overline{S}_1 \cdot E + \overline{S}_0 \cdot E$$

$$S`_1(S_1,S_0,E) = \overline{S}_1 \cdot S_0 \cdot \overline{E} + S_1 \cdot \overline{S}_0 \cdot E$$

La Table des Sortie est si simple que ça formule peut être extraite directement :

$$O(S_1,S_0,E) = S_1 \cdot \overline{S}_0 \cdot E$$

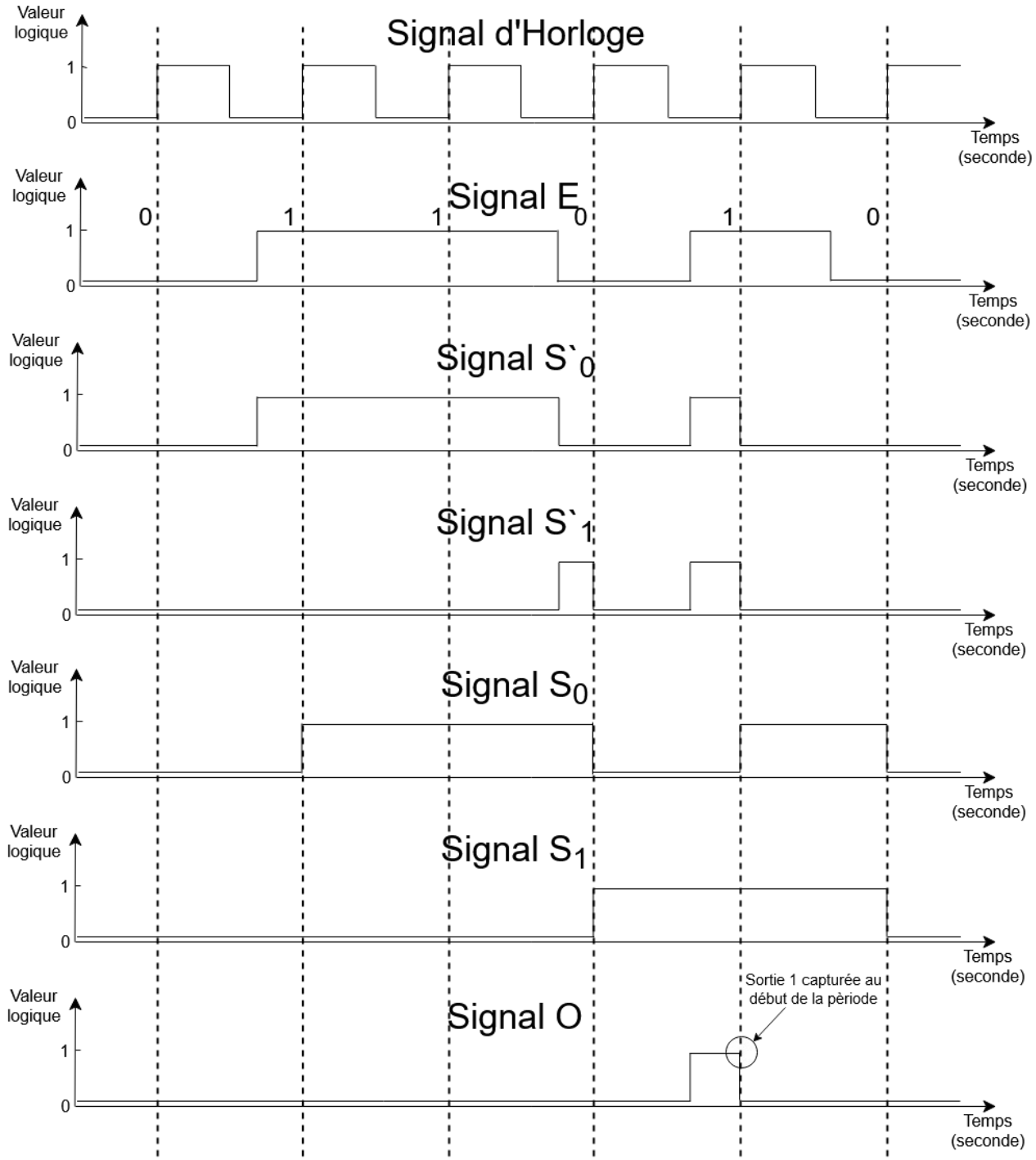
<u>Étape 7</u> : Logigramme



Remarque 1: Un encodage avec One-hot ou One-cold est aussi possible sur la Machine de Mealy.

Remarque 2: Pour la conception d'un Circuit Séquentiel, le choix entre la Machine de Moore et la Machine de Mealy, avec chaque machine 3 formes d'encodage; normal, One-hot et One-cold, est généralement orienté par des contraintes comme: le nombre de portes disponibles, le nombre de cellules-mémoires disponibles, la vitesse ou la fréquence voulue pour le circuit, la consommation électrique maximale...etc. Toutes ces contraintes influencent généralement les décisions de conception d'un circuit électronique numérique.

Exécution : On refait une exécution identique à celle de l'exemple 1 avec un scénario des entrées en séquence 011010. l'exécution est schématisée sur le chronogramme en bas.



Remarque 1: La principale différence dans l'exécution entre la Machine de Moore et la Machine de Mealy est que la valeur 1 de sortie est capturée lors du front montant terminant la période de l'État S3 pour celle de Moore, alors que la sortie 1 est capturée lors du front montant du début de période de l'État S3 pour la Machine de Mealy.

Remarque 2: Il est possible de capturer la sortie dans la Machine de Mealy la durée d'une période, il suffit de rajouter un registre à la sortie de la machine.

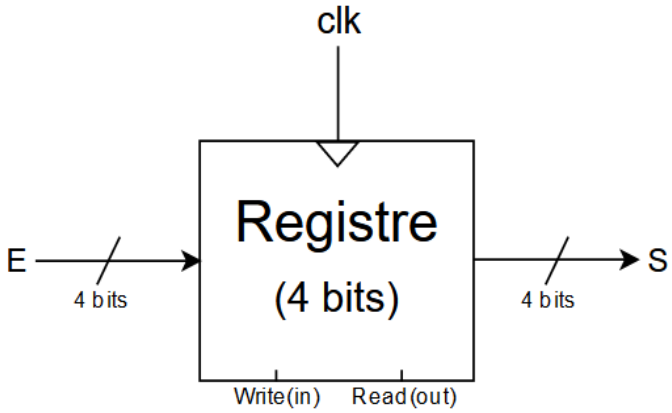
7. Les Circuits Séquentiels les plus connus

Comme pour les Circuits Combinatoires il existe quelques Circuits Séquentiels qui sont très connus et très utilisés.

7.1. Les Registres

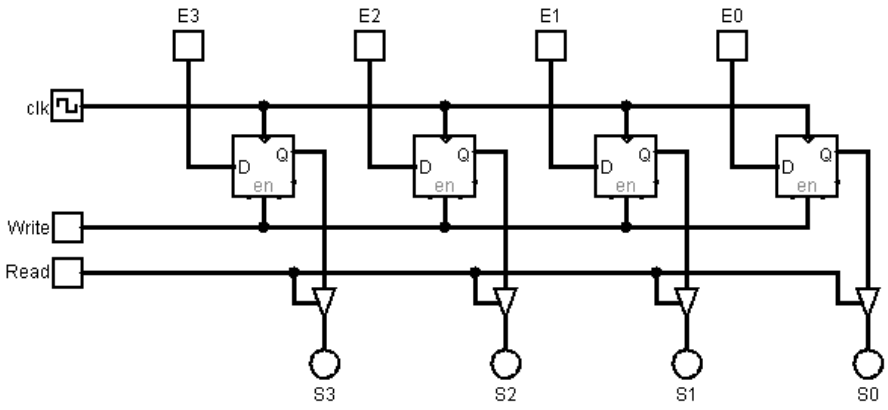
- ➔ On a déjà vu la forme la plus simple d'un registre, c'est un tableau de D-FlipFlop avec une horloge commune.
- ➔ Le Registre peut avoir plus de fonctionnalités, comme la lecture et l'écriture.
- ➔ La lecture et l'écriture sont réalisées par 2 entrées de commande comme sur la figure en bas.

La figure représente un Registre de 4 bits avec en plus les entrées de commande de Lecture et d'Écriture.



- ➔ Le Write (ou in) s'il est mis-à-1 va permettre à la valeur dans E d'entrer lors du Front Montant et d'écraser l'ancienne valeur dans le Registre, par contre s'il est mis-à-0 la valeur ne va pas entrer et l'information sauvegardée reste inchangée.
- ➔ La commande Read (out) donne au Registre la faculté de ne pas sortir l'information sauvegardée à l'intérieur. Si Read est mis-à-1 la valeur dans le Registre sort dans S, par contre s'il est mise-à-0 c'est la valeur flottante Z qui sera mise sur la sortie S.

Une implémentation basique de la lecture et de l'écriture est détaillée sur la figure suivante



➔ Le Write dans la figure est implémenté par l'entrée Enable de la D-FlipFlop, ainsi s'il elle est mise-à-0 les valeurs de E ne vont pas entrer lors du Front Montant.

➔ Le Read est implémenté en utilisant la porte Tristate buffer entre les sorties Q des D-FlipFlop et la sortie S, ainsi si la commande de la Tristate buffer est mise-à-0 sa sortie produisent Z, qui est supposé comme un signal avec aucune valeur.

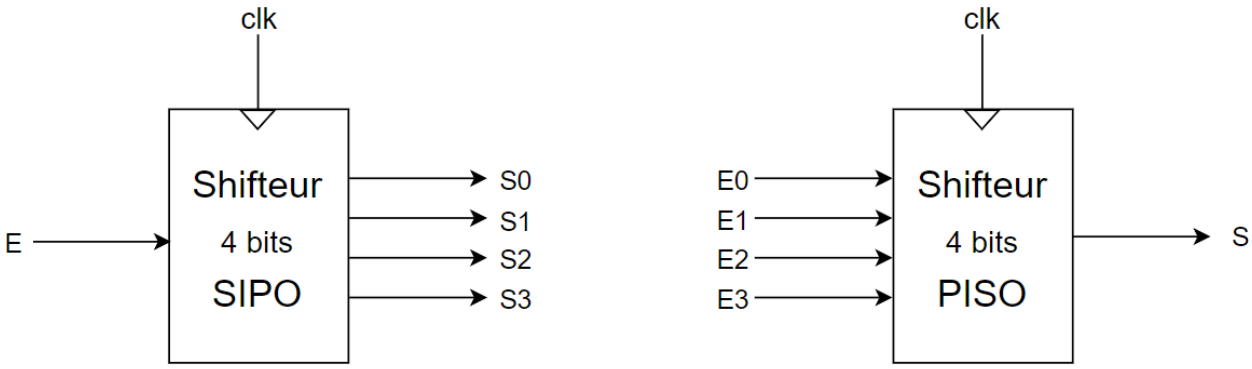
7.2. Les Shifteurs

➔ Comme il existe un Shifteur Combinatoire il existe aussi un Shifteur Séquentiel.

➔ La différence entre un Shifteur Combinatoire et un Shifteur Séquentiel, c'est que dans le Combinatoire on doit préciser le nombre de bits à décaler, alors que dans le Séquentiel un bit est automatiquement décalé pour chaque Front Montant d'horloge.

➔ Deux types de Shifteurs Séquentielles existent, Entrée Série / Sortie Parallèle (en Anglais : Serial In Parallel Out ou SIPO) et du type Entrée Parallèle / Sortie Série (en Anglais : Parallel In Serial Out ou PISO).

La figure suivante illustre les 2 types de Shifteurs, à gauche le Shifteur Séquentiel 4 bits Entrée Série / Sortie Parallèle (SIPO), et à droite le Shifteur Séquentiel 4 bits Entrée Parallèle / Sortie Série (PISO).



➔ Un Shifteur SIPO fonctionne de telle sorte qu'à chaque Front Montant l'entrée E fait entrer une nouvelle valeur, les valeurs entrées en chaîne l'une après l'autre vont sortir suivant le même ordre en parallèle sur les sorties S0 jusqu'à S4.

➔ Par exemple si au 1-ier Front Montant on faisait entrer la valeur 1, celle-ci va sortir dans S0, et si dans le 2-ième Front Montant on faisait entrer la valeur 0, la sortie serait 1 dans S1 et 0 dans S0, et de même pour le 3-ième Front Montant, si on faisait entrer la valeur 1 c'est la valeur 101 qui va sortir respectivement dans S2 S1 S0, et ainsi de suite.

➔ Ça veut dire que le nouveau bit entré va décaler les autres bits d'une position pour frayer ça propre place en première position, et de même pour le bit qui va le suivre.

➔ Dans un Shifteur PISO c'est l'inverse, ça veut dire qu'on fait enter les 4 valeurs en même temps en parallèle, et ceux-ci vont après sortir successivement l'une après l'autre dans la sortie E pour chaque Front Montant d'horloge.

- ➔ Il y a une commande indispensable dans le Shifteur PISO qui n'est pas représenté dans le diagramme en haut, c'est la commande qui va choisir entre l'opération de lire les 4 valeurs en parallèle lors du Front Montant, ou l'opération de les faire sortir l'une après l'autre en série pour chaque Front Montant.
- ➔ Les Shifteur PISO et SIPO ont tous les deux la capacité d'adhérer à la composition en cascade, ce qui les rend très flexibles pour créer des Shifteurs de tailles variées, il suffit juste de les rattacher en cascade pour obtenir des Shifteur de plus grandes tailles.

Remarque 1: Pour les Shifteurs Séquentiels, connaître l'orientation du décalage gauche ou droite n'a trop d'importance comme pour les Shifteurs Combinatoires, puisque c'est une transmission bit par bit entre les 2 représentations Série et Parallèle.

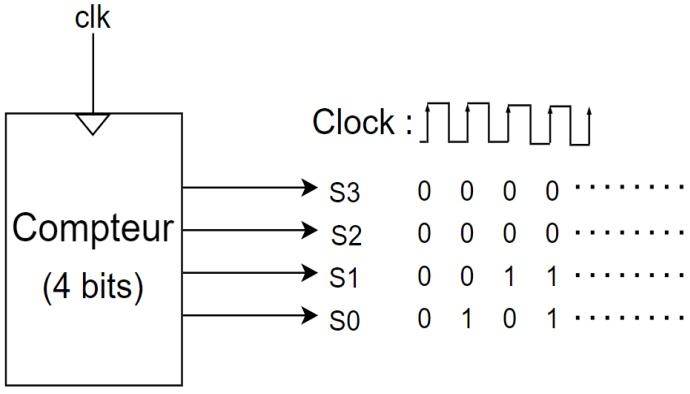
Remarque 2: Les Shifteurs Séquentiels ont un rôle primordial pour faire office d'interface de conversion Série/Parallèle ou Parallèle/Série.

Remarque 3: Les Shifteurs SIPO sont aussi très utilisés par les processeurs pour étendre leur nombre de sorties. Par exemple, pour une calculatrice avec un affichage de 10 afficheurs 7 segments, normalement son processeur aurait besoin de 10 x 7 = 70 ports de sorties, ce qui est énorme. En réalité le processeur utilise une seule sortie, sur laquelle il va sortir les 70 valeurs de segments en série, qui seront fournies en parallèle à l'aide des SIPO pour les 10 afficheurs 7 segments.

7.3. Les Compteurs

- ➔ Le Compteur est un Circuit Séquentiel qui peut compter en binaire.
- ➔ C'est un circuit qui ne possède pas d'entrées mais dispose de n sorties pour représenter une valeur binaire sur n bits.
- ➔ Pour chaque Front Montant il va compter, commençant par la valeur 0, puis 1, puis 2...jusqu'à (2^n-1), pour ensuite retourner à la valeur initiale 0 et recommencer.

Sur la figure en bas on peut voir comment le Compteur sur 4 bits peut compter en binaire en synchronisant avec chaque Front Montant du signal d'horloge.



➔ On peut voir sur le diagramme que sur le 1-ier Front Montant le Compteur fait sortir la valeur 0 (0000), le 2-ième la valeur 1 (0001), le 3-ième la valeur 2 (0010)...etc.

➔ Le Compteur peut disposer une entrée de commande nommée Reset, si elle est mise-à-1 elle va réinitialiser le comptage à 0, même si le Compteur n'a pas encore atteint sa dernière valeur (2^n-1).

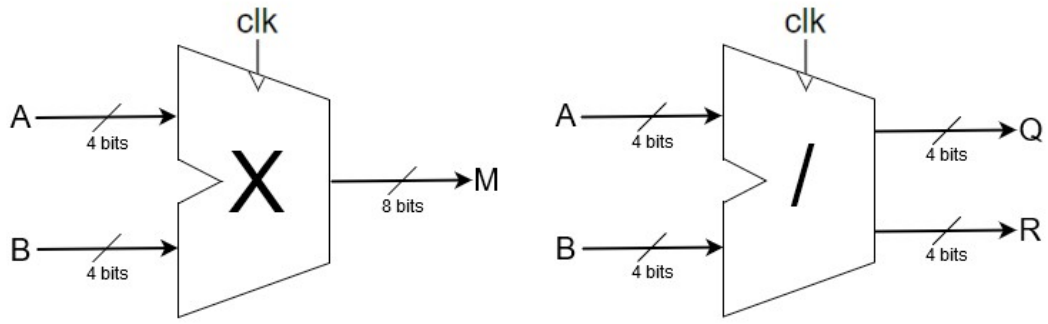
Remarque : Il n'est pas rare de trouver des Compteurs avec une entrée sur n bits, ça leurs permet de forcer une valeur initiale différente de 0 si la nécessité se fait sentir.

7.4. Le Multiplicateur et le Diviseur

➔ Les Multiplicateurs et les Diviseurs Séquentiels sont identiques aux Multiplicateurs et Diviseurs Combinatoires, la seule différence c'est que ces derniers possèdent en plus une entrée d'horloge.

➔ Car le calcul dans ces derniers doit consumer un peu de temps avant de se terminer.

Un Multiplicateur 4 bits est représenté en bas à gauche de la figure, et un Diviseur sur 4 bits est représenté en bas à droite.



➔ L'opération de la multiplication doit attendre un temps de *2n* périodes d'horloge pour se terminer, et la division a besoin de *n* périodes d'horloge.

➔ En réalité, le Multiplicateur et le Diviseur calculent un seul bit pour chaque Front Montant d'horloge.

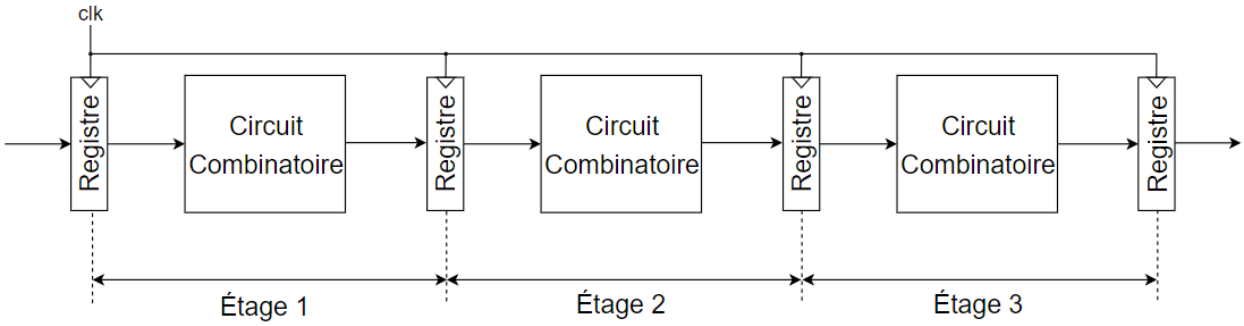
➔ Rappelons que la multiplication et la division combinatoire effectuent leurs calculs instantanément.

Remarque 1: La différence entre les Multiplicateur/Diviseur Combinatoire et Séquentiel, c'est que les combinatoires ont l'avantage d'être beaucoup plus rapides que les séquentiels. Par contre, la complexité et le nombre de portes pour la construction des séquentiels sont beaucoup plus faibles que celle des combinatoires.

Remarque 2: Parmi les Circuits Séquentiels les plus connus, il y a aussi les Unité de Contrôle et de Commande (UCC), ils ont le rôle de contrôler les Processeurs de l'intérieur.

8. Le Pipeline

Le Pipeline est une forme de Circuits Séquentiels qui n'utilisent pas de Boucles dans leur fonctionnement. Le circuit en Pipeline se constitue d'un nombre fixe d'étage. On peut voir sur la figure en bas un Pipeline de 3 étages par exemple.



- ➔ Le Pipeline peut être considéré comme un cas particulier des Circuit Séquentiel avec un traitement constant et fixe des données, sans variation dans le traitement comme pour les Circuits Séquentiels à base de boucles.
- ➔ Le Pipeline est généralement utilisé pour des applications de flux de données (ou data streaming en Anglais), avec un traitement fixe et par étapes sur ces données.
- ➔ Le Pipeline exerce un parallélisme du traitement, dans le sens où plusieurs données sont exécutées en même temps mais chacune à une étape différente.

Remarque : L'exemple le plus apparent pour l'utilisation des Pipelines, c'est sans doute les décodeurs Hardware des flux audio et vidéo, se trouvant généralement à l'intérieur des Cartes Graphiques, ils permettent de visualiser des vidéos et d'écouter de l'audio.