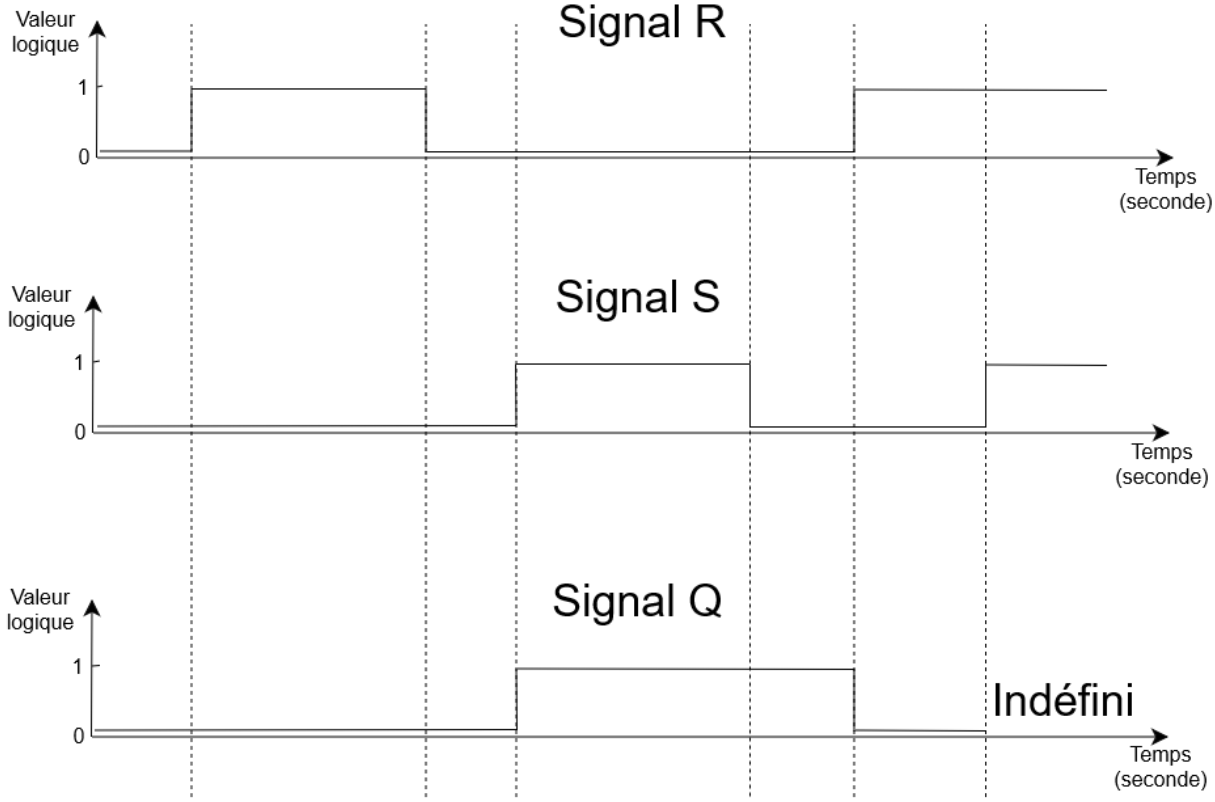
Université Sétif 1
Faculté des Sciences
Tronc Commun Mathématique Informatique

1-ière année (année2021/22)
Module Structures Machines 2
Enseignant Kara Abdelaziz

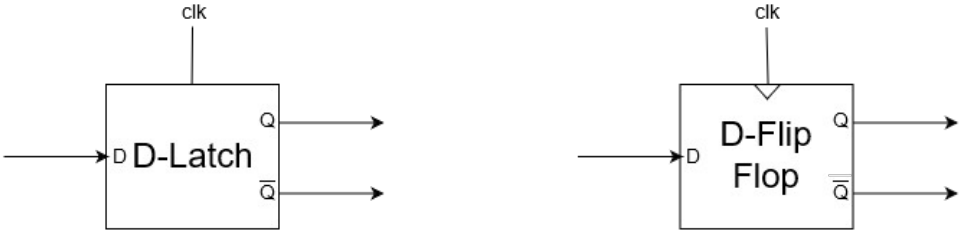
Solution Série 2
(Circuits Séquentiels)

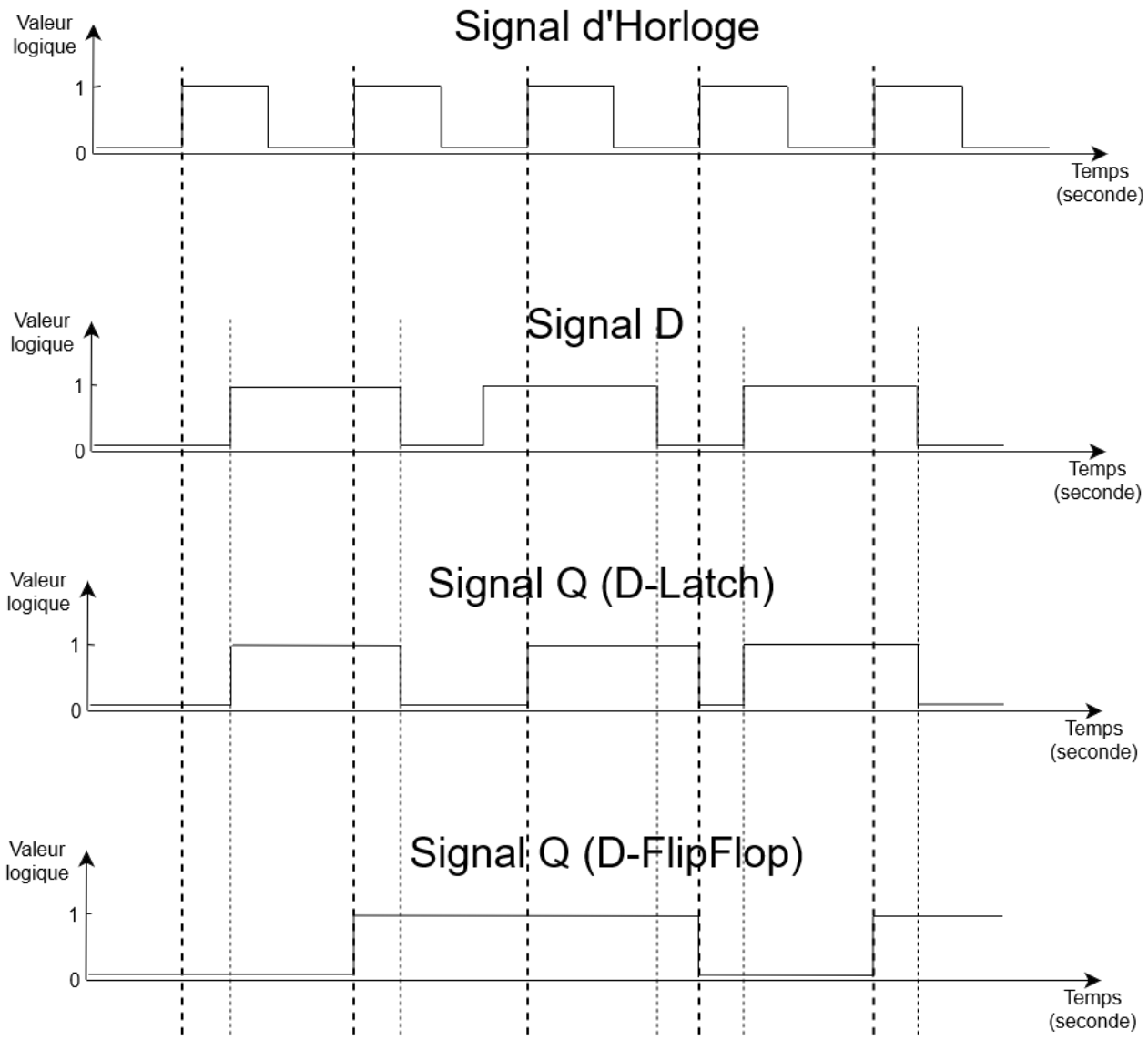
Exercice 01 :

1) Le chronogramme de la RS-Latch est comme suite :



2) Le chronogramme de la D-Latch et de D-FlipFlop :

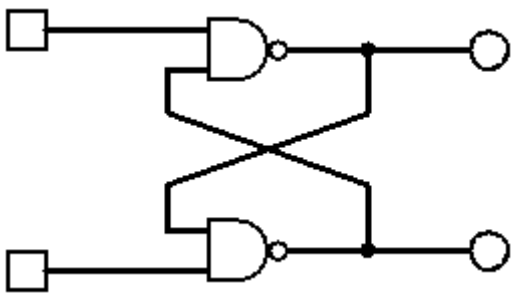




Exercice 02 :

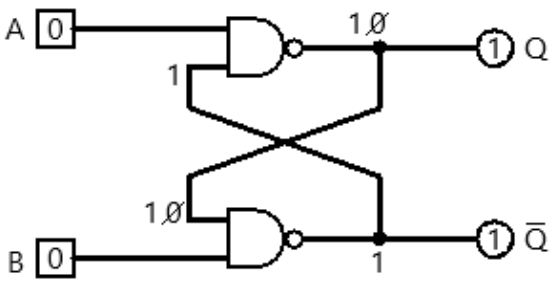
1)

1.

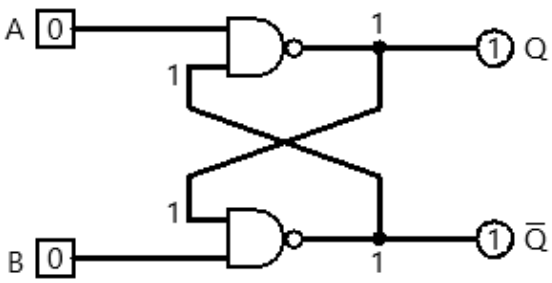


Pour A=0 et B=0

Initialement Q=0

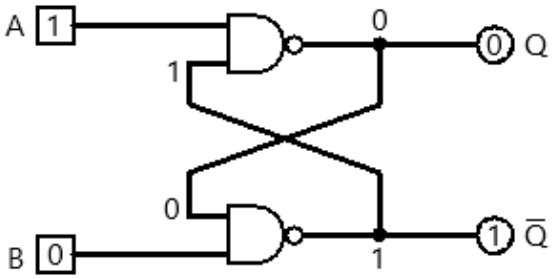


Initialement Q=1

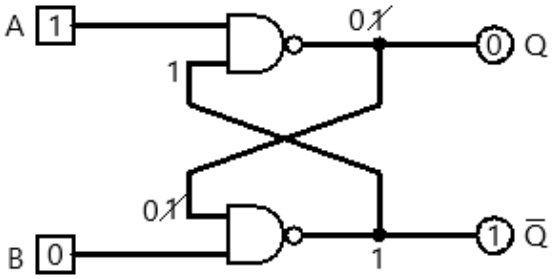


Pour A=1 et B=0

Initialement Q=0

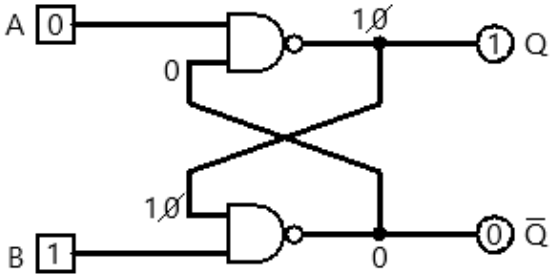


Initialement Q=1

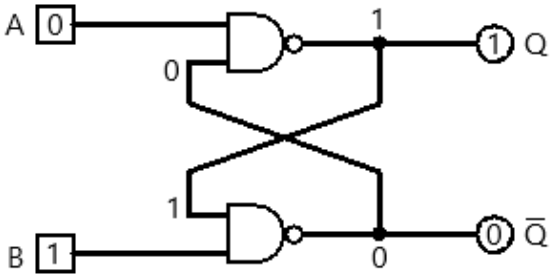


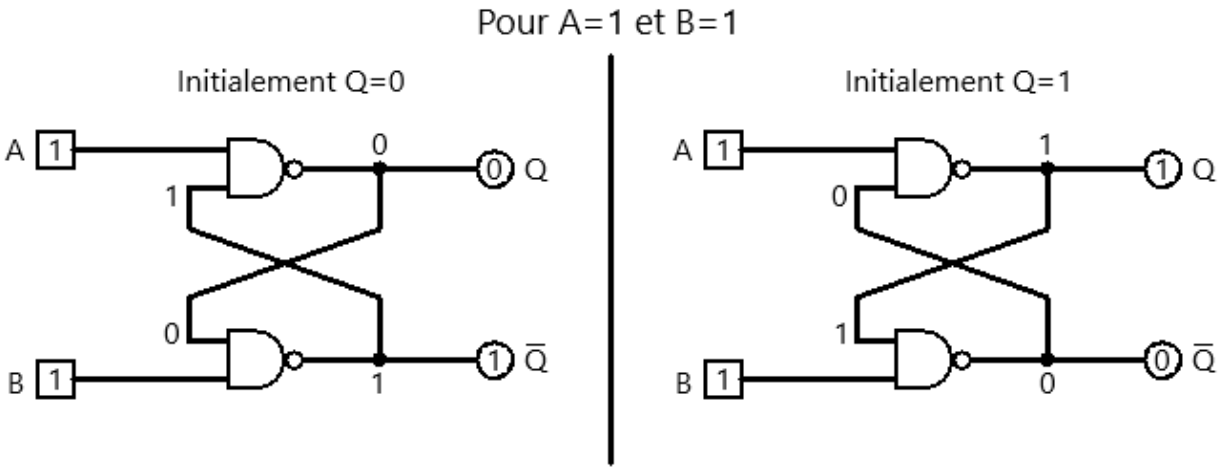
Pour A=0 et B=1

Initialement Q=0



Initialement Q=1

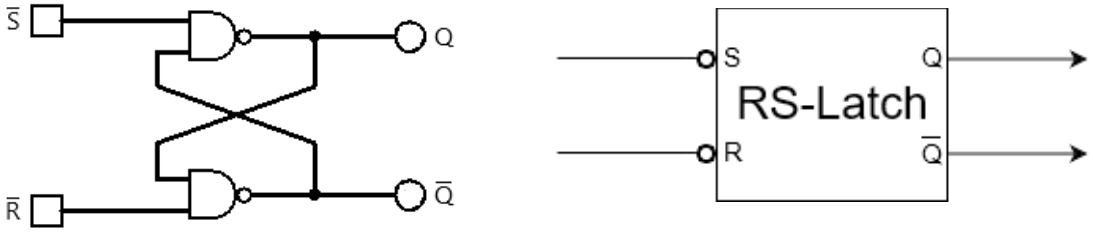




Ainsi on aurait une table de vérité :

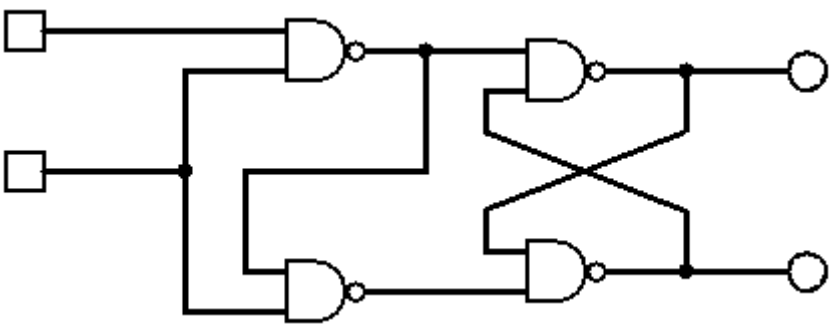
A	B	Q	$\overline{Q}$	
0	0	1	1	Non défini
0	1	1	0	Mettre à 1
1	0	0	1	Mettre à 0
1	1	Q`	$\overline{Q}$`	Mémorise

Par observation, la table de vérité représente un circuit RS-Latch avec A=$\overline{S}$ et B=$\overline{R}$

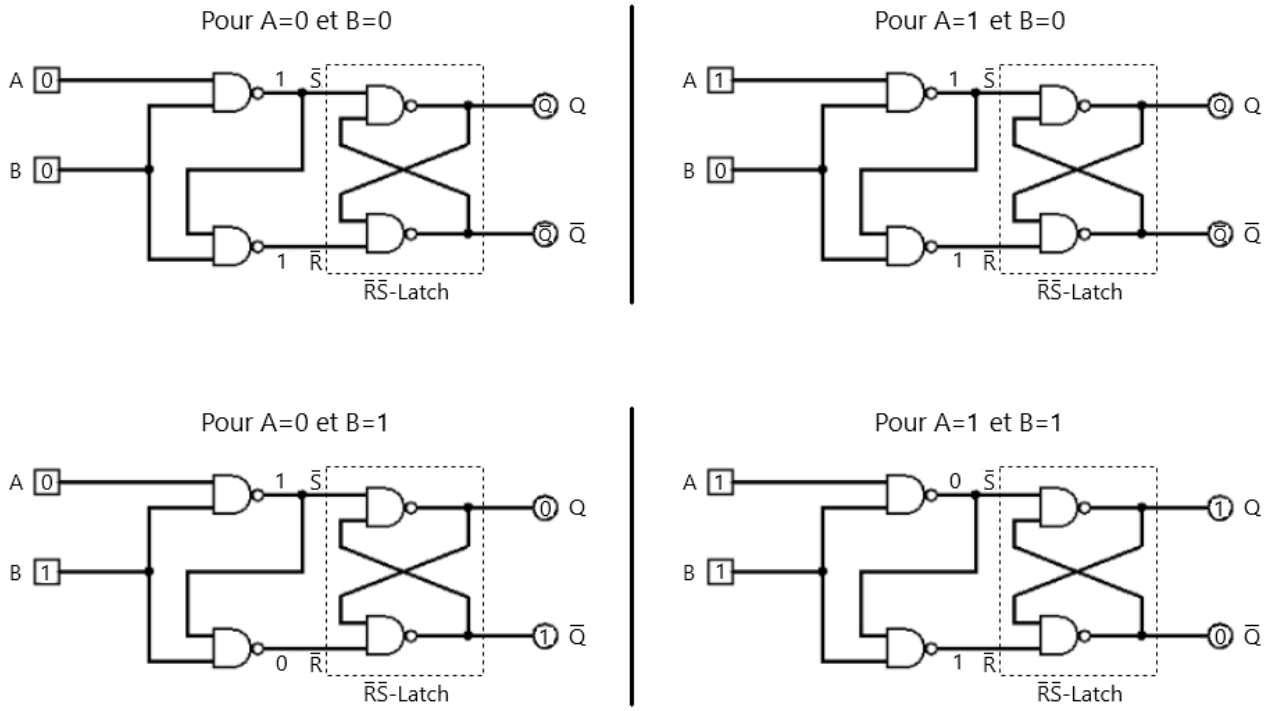


C'est ce qu'on appelle un circuit $\overline{R}\,\overline{S}$-Latch

2.



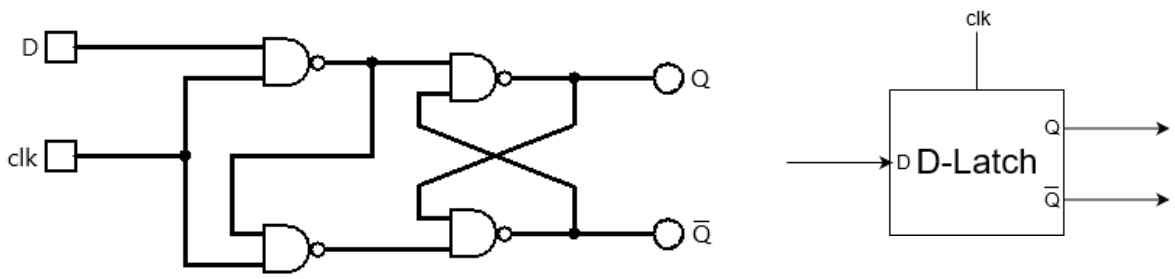
On remarque que les deux portes de sortie (les plus à droite) forment un circuit $\overline{R}\,\overline{S}$-Latch, ça va grandement simplifier la tâche d'analyse.



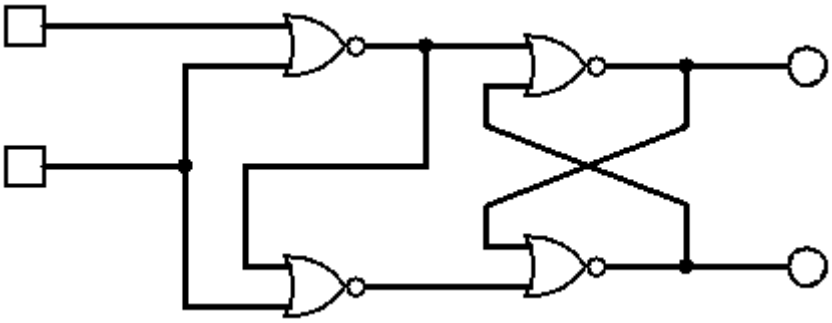
La table de vérité est :

A	B	Q	$\overline{Q}$	
0	0	Q`	$\overline{Q}$`	Mémorise
0	1	0	1	Mettre à 0
1	0	Q`	$\overline{Q}$`	Mémorise
1	1	1	0	Mettre à 1

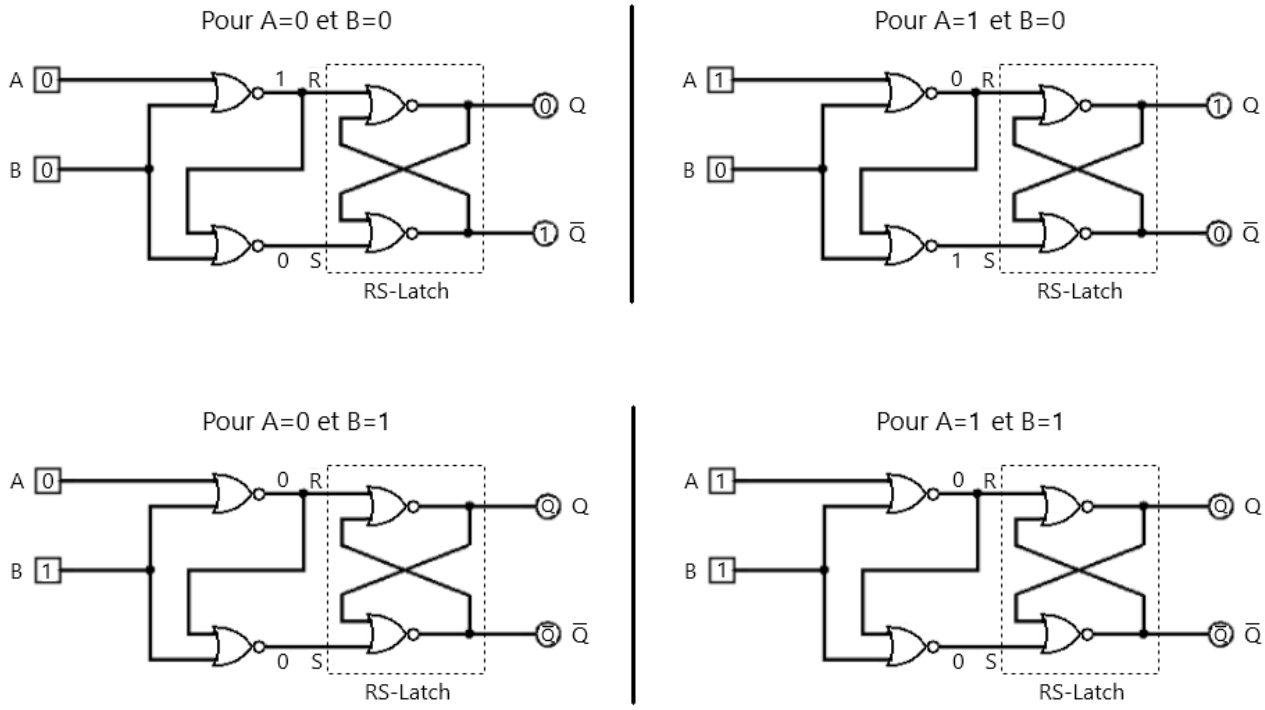
En comparaison avec les tables de vérité des autres bascules, cette table est identique à celle de la D-Latch, où A=D et B=clk. Donc le circuit représente la Bascule D-Latch :



3.



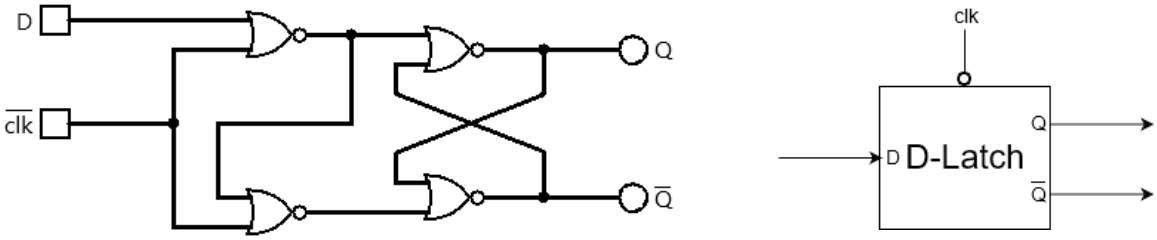
Par observation les deux dernières portes de sortie (les plus à droite) représentent le circuit RS-Latch, ce qui va nous faciliter le processus d'analyse du circuit.



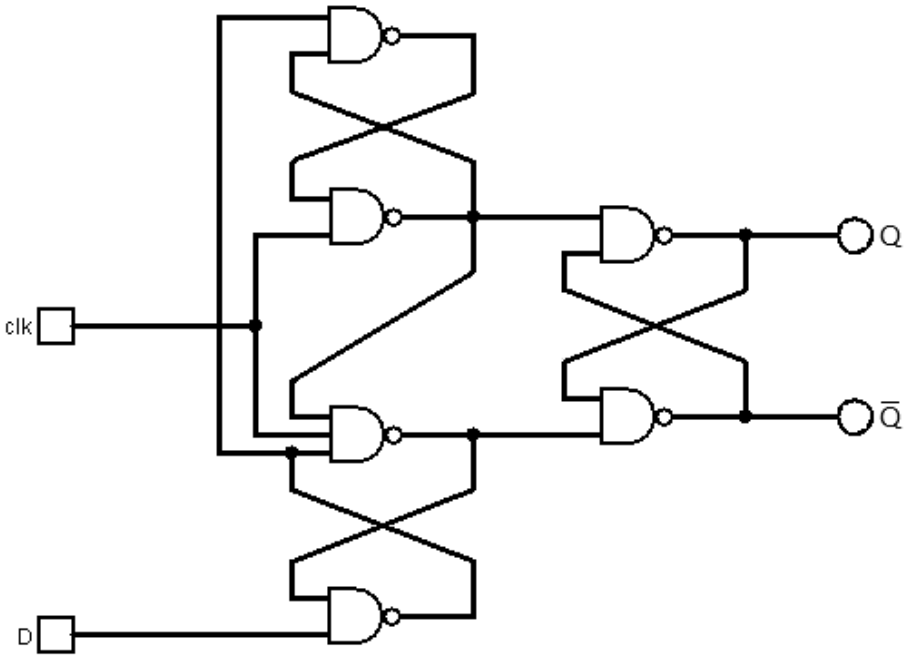
La table de vérité :

A	B	Q	$\overline{Q}$	
0	0	0	1	Mettre à 0
0	1	Q`	$\overline{Q}$`	Mémorise
1	0	1	0	Mettre à 1
1	1	Q`	$\overline{Q}$`	Mémorise

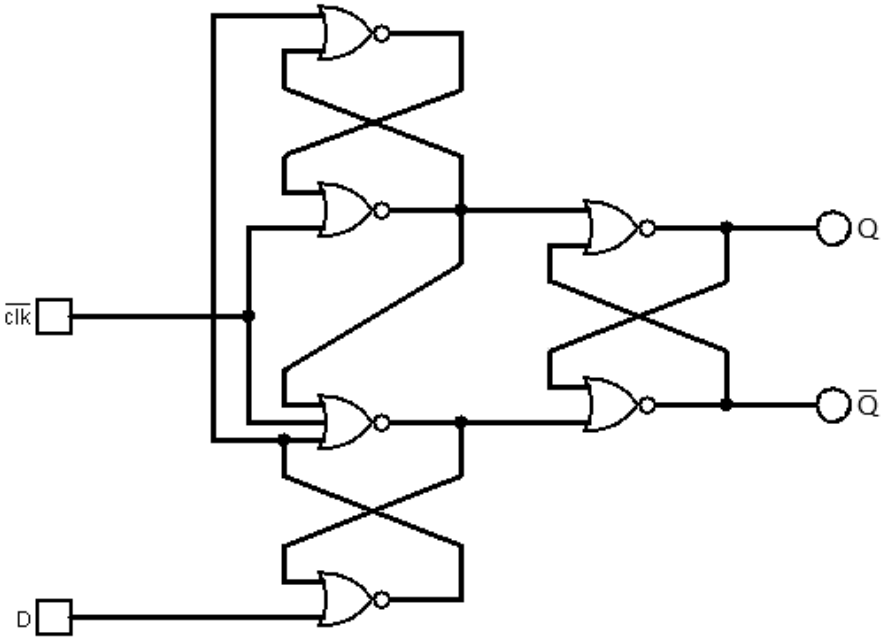
Par comparaison des tables de vérité, la table représente un circuit D-Latch avec A=D et B=$\overline{clk}$, C'est une D-Latch avec une horloge inversée :



2) La D-FlipFlop en 6 portes NAND :



La D- FlipFlop en 6 portes NOR :



Remarque 1: Plusieurs bascules utilisent une horloge inversée au-lieu d'une horloge normale, il est facile de penser qu'il suffit de mettre juste une porte NOT à l'entrée de l'horloge et ils deviennent des bascules normales, alors que dans la pratique il est très déconseillé d'ajouter des portes sur le signal d'horloge, ça va retarder le signal et accentuer un phénomène appelé *skew d'horloge*. Réellement ce phénomène touche principalement les systèmes hautement performants de haute fréquence, mine de rien il vaut mieux apprendre les bonnes pratiques de la conception Hardware et éviter d'ajouter des portes sur les signaux d'horloge.

Remarque 2: La solution pour l'horloge inversée est que dans la majorité des cas, les circuits qui fournissent le signal d'horloge produisent les 2 signaux clk et $\overline{clk}$ en même temps sur 2 fils différents, de cette manière les bascules et autres circuits peuvent choisir le signal approprié qui leur convient.

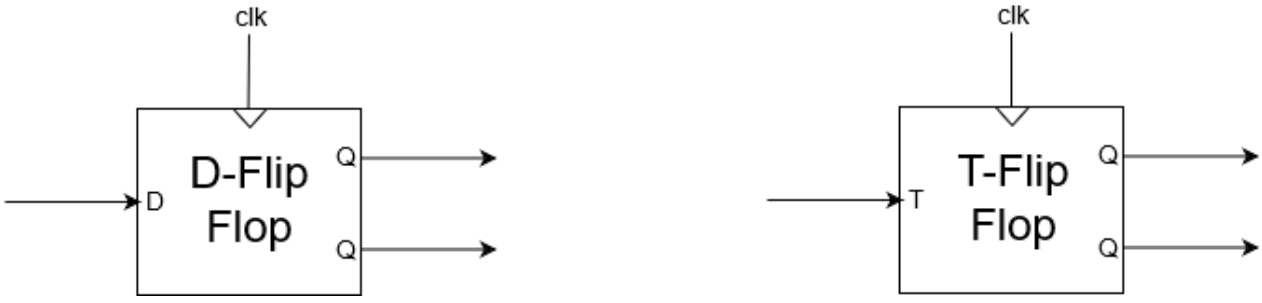
Remarque 3: Lors de la réalisation de la D-FlipFlop en utilisant 2 D-Latch (le chapitre 2 du cours) la première D-Latch reçoit le signal d'horloge inversé traversant une porte NOT, alors qu'on vient juste de dire qu'il est déconseillé de faire ceci. En réalité l'utilisation de cette porte NOT est obligatoire pour faire fonctionner correctement la D-FlipFlop, elle permet de retarder légèrement le signal $\overline{clk}$ pour la première D-Latch, ainsi lors de la phase du front descendant de l'horloge (voir le cours) la deuxième D-Latch serait plus rapide que la première, ce qui lui permettra de sauvegarder la valeur de Y avant que X entre et change la valeur de la première D-Latch.

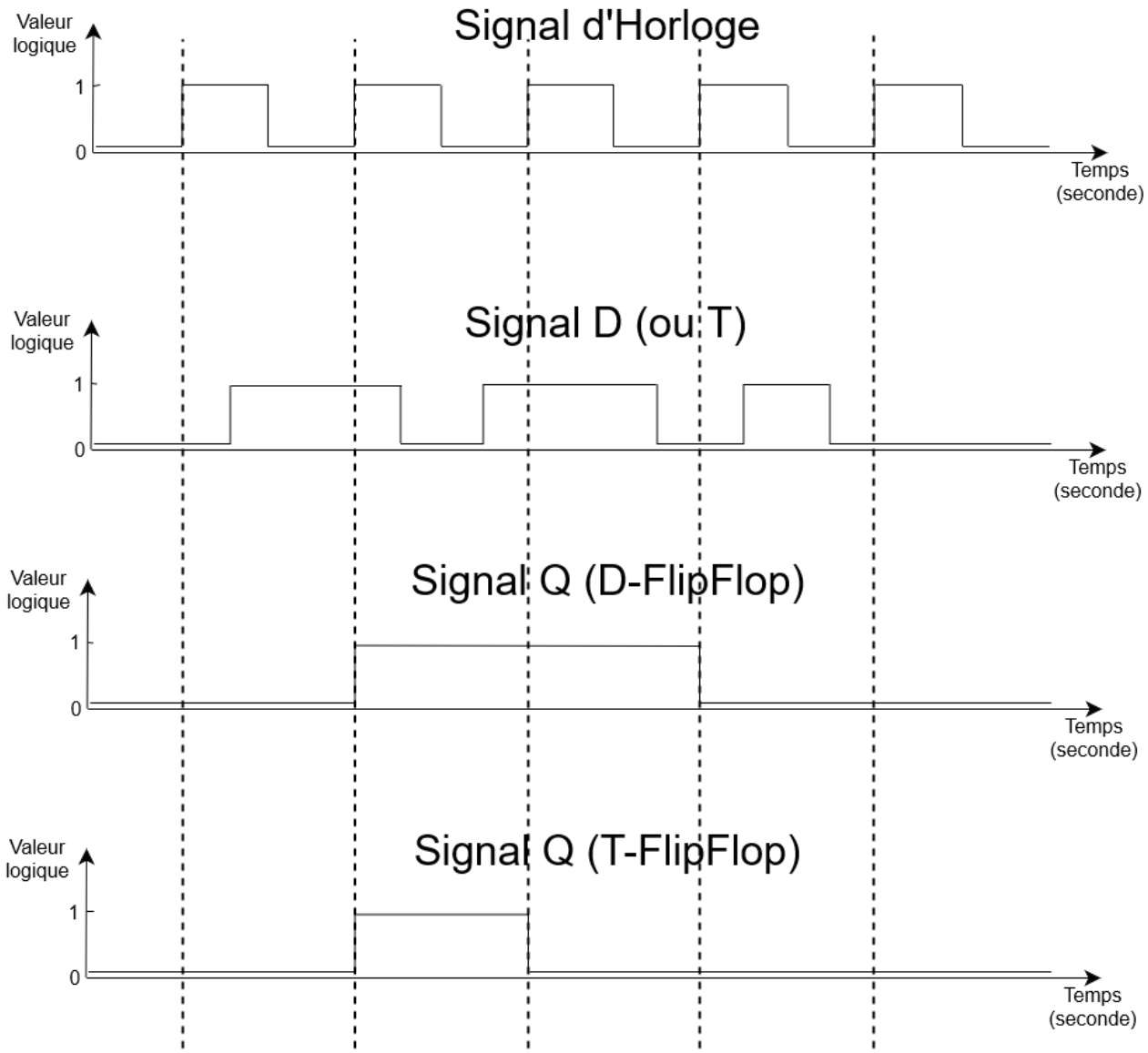
Remarque 4: Pour une implémentation des D-FlipFlop en 6 portes NAND et NOR en comparaison avec l'implémentation utilisant 2 D-Latch, cette dernière compte en tout 11 portes différentes. La première contient moins de portes donc elle est plus optimale, en plus l'utilisation des portes universelles réduit le nombre de transistors et elles sont électriquement plus optimales que les autres portes.

Remarque 5: Dans les circuits modernes et pour plus d'optimalité, les FlipFlop sont implémentées au niveau transistors, ils sont ainsi encore plus réduits.

Exercice 03 :

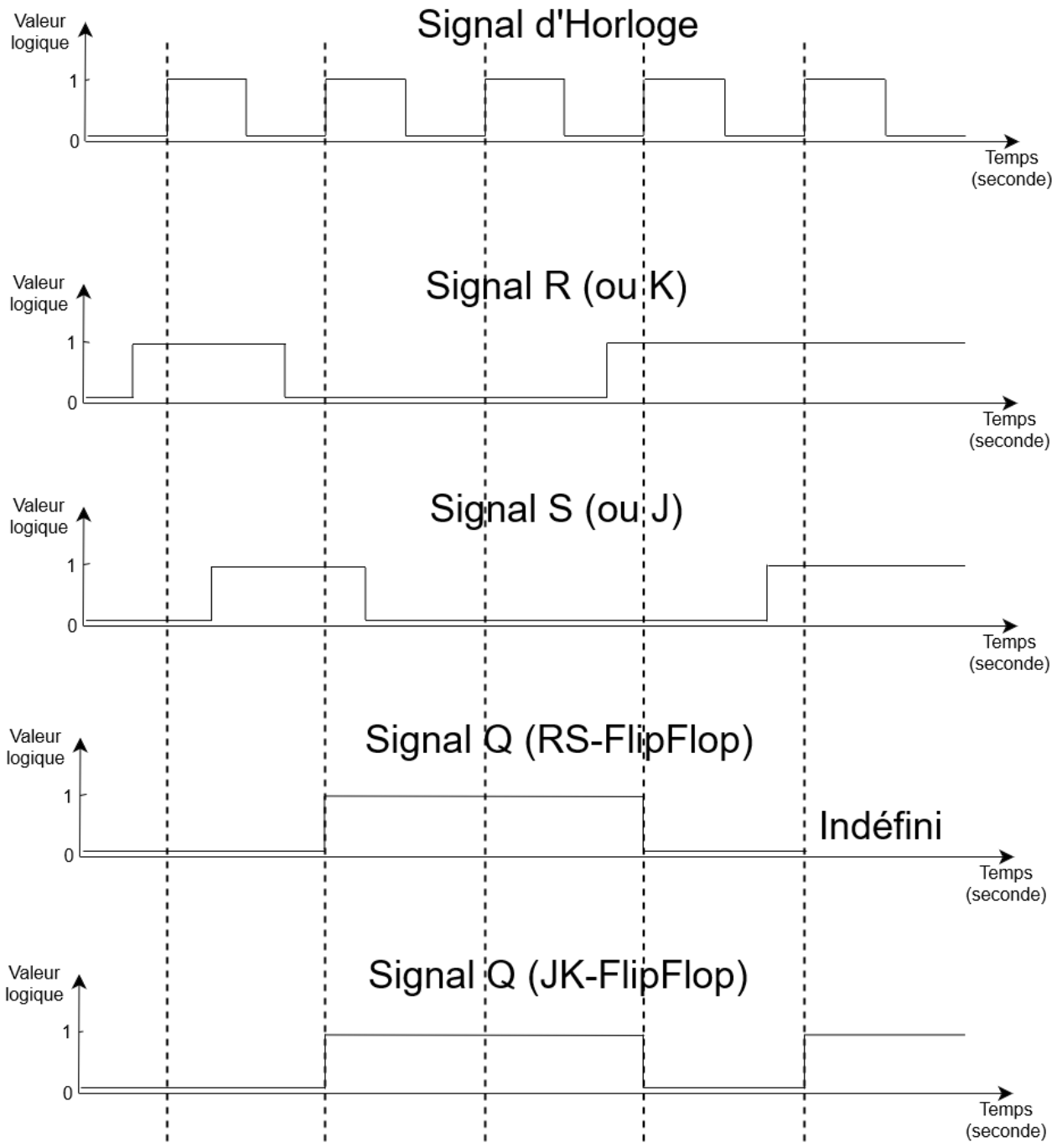
1)




Signal d'Horloge
Valeur logique
1
0
Temps (seconde)
Signal D (ou T)
Valeur logique
1
0
Temps (seconde)
Signal Q (D-FlipFlop)
Valeur logique
1
0
Temps (seconde)
Signal Q (T-FlipFlop)
Valeur logique
1
0
Temps (seconde)

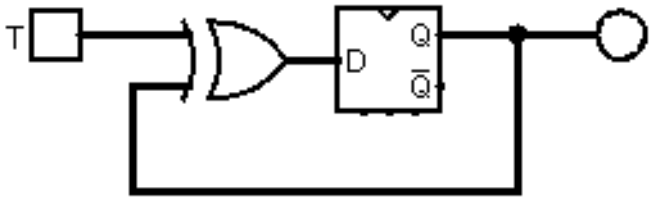
2)


R
S
RS-Flip Flop
Q
Q̄
K
J
JK-Flip Flop
Q
Q̄

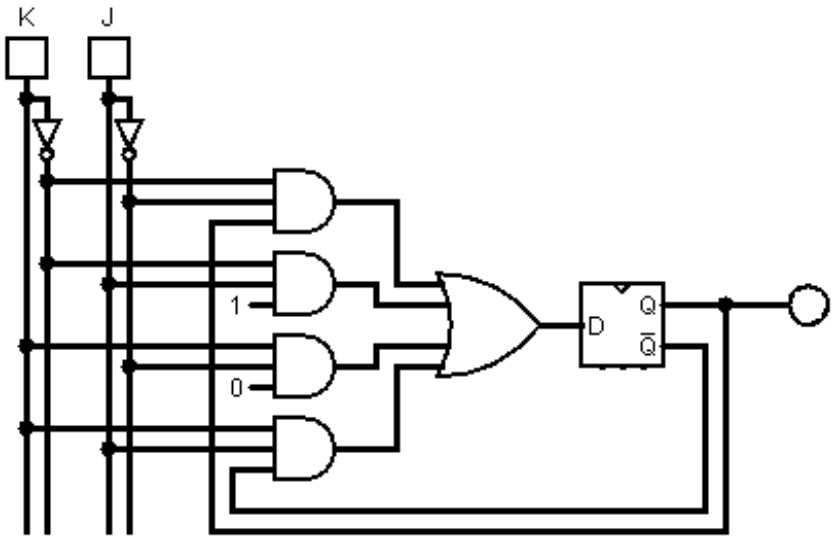


3) La construction d'une Bascule D-FlipFlop à partir d'une Bascule RS-FlipFlop :

4) La construction d'une Bascule T-FlipFlop à partir d'une Bascule D-FlipFlop :

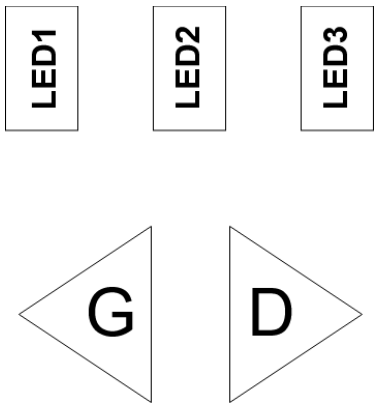


5) La construction d'une Bascule JK-FlipFlop à partir d'une Bascule D-FlipFlop :



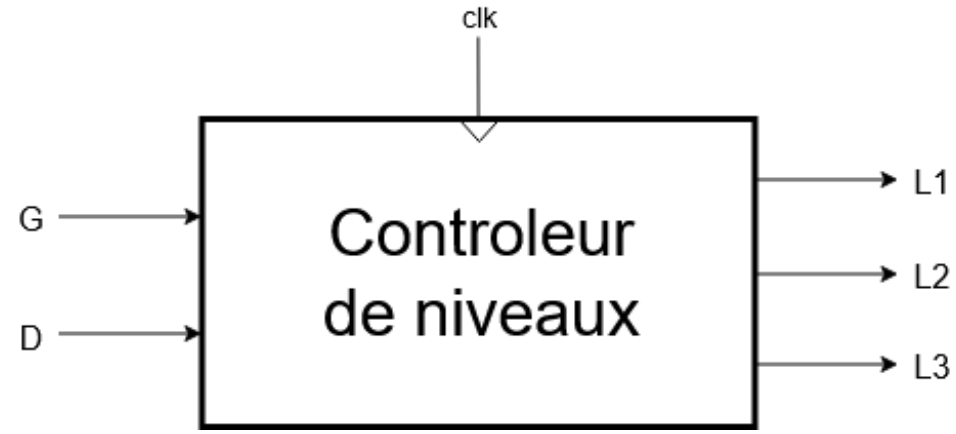
Remarque : Pour la JK-FlipFlop chaque porte AND représente une combinaison d'entrée possible (00, 01, 10, 11), si l'une des combinaisons est vérifiée la valeur adéquate est passée à la D-FlipFlop. Par exemple dans la première porte K=0 et J=0, si c'est vérifié, elle va retourner la valeur Q vers la cellule mémoire, ce qui suit exactement le comportement d'une bascule JK-FlipFlop. De même pour les 3 autres portes AND restantes.

Exercice 04 :

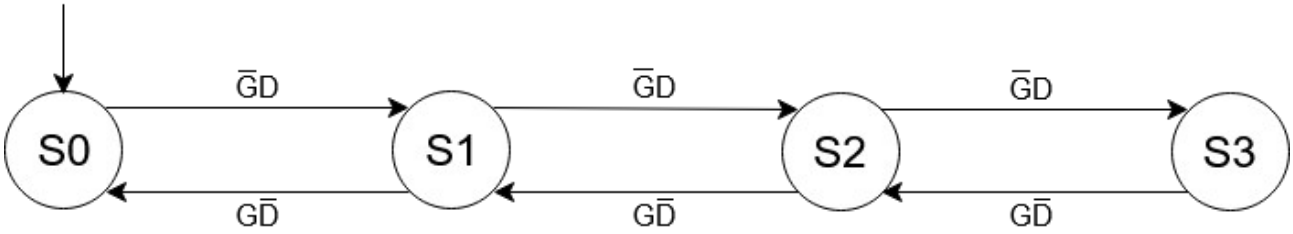


1)

<u>Étape 1</u> : Schéma global



<u>Étape 2</u> : Automate



S0 → Tous les LED sont éteintes
S1 → La LED1 est allumée
S2 → La LED1 et LED2 sont allumées
S3 → Tous les LED sont allumées

S0 → L1 = 0, L2 = 0, L3 = 0
S1 → L1 = 1, L2 = 0, L3 = 0
S2 → L1 = 1, L2 = 1, L3 = 0
S3 → L1 = 1, L2 = 1, L3 = 1

Remarque 1: Les sorties devraient normalement être représentées à l'intérieur du cercle de l'état, elles sont étés listées en dessous de l'Automate pour plus de simplicité.

Remarque 2: Les éventements sont décrits par des expressions booléennes au-lieu des valeurs binaires ($\overline{G}D \iff G=0$, $D=1$), ça permet de les réduire en écriture et de rassembler plusieurs combinaisons de valeurs sous la même expression.

<u>Étape 3</u> : Table de Transition

État actuel	G	D	État suivant
S0	0	0	S0
	0	1	S1
	1	0	S0
	1	1	S0
S1	0	0	S1
	0	1	S2
	1	0	S0
	1	1	S1
S2	0	0	S2
	0	1	S3
	1	0	S1
	1	1	S2
S3	0	0	S3
	0	1	S3
	1	0	S2
	1	1	S3

Remarque : La Table de Transition doit contenir toutes les combinaisons d'entrées possibles.

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

États	S_1	S_0	L1	L2	L3
S0	0	0	0	0	0
S1	0	1	1	0	0
S2	1	0	1	1	0
S3	1	1	1	1	1

Remarque : Les formules de L1 et L2 et L3 peuvent être extraites visuellement.

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

S_1	S_0	G	D	S'_1	S'_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	1
1	0	1	0	0	1
1	0	1	1	1	0
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	0
1	1	1	1	1	1

<u>Étape 6</u> : Formules Logiques

$L1 = S_1+S_0$
$L2 = S_1$
$L3 = S_1 \cdot S_0$

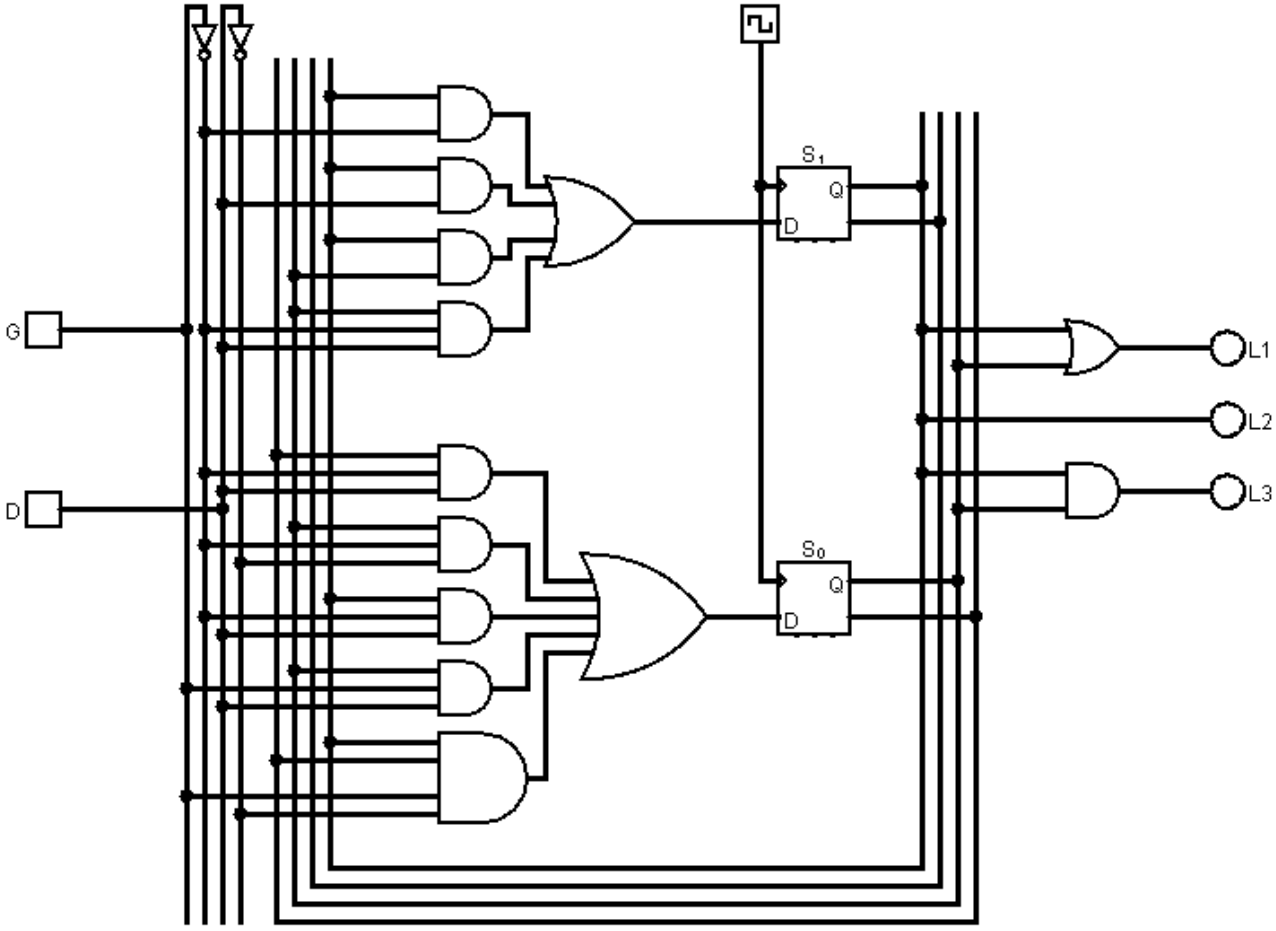
S_1S_0 / GD	00	01	11	10
00	0	0	1	1
01	0	1	1	1
11	0	0	1	1
10	0	0	1	0

S_1S_0 / GD	00	01	11	10
00	0	1	1	0
01	1	0	1	1
11	0	1	1	0
10	0	0	0	1

$S'_1(S_1,S_0,G,D) = S_1 \cdot \overline{G} + S_1 \cdot D + S_1 \cdot S_0 + S_0 \cdot \overline{G} \cdot D$

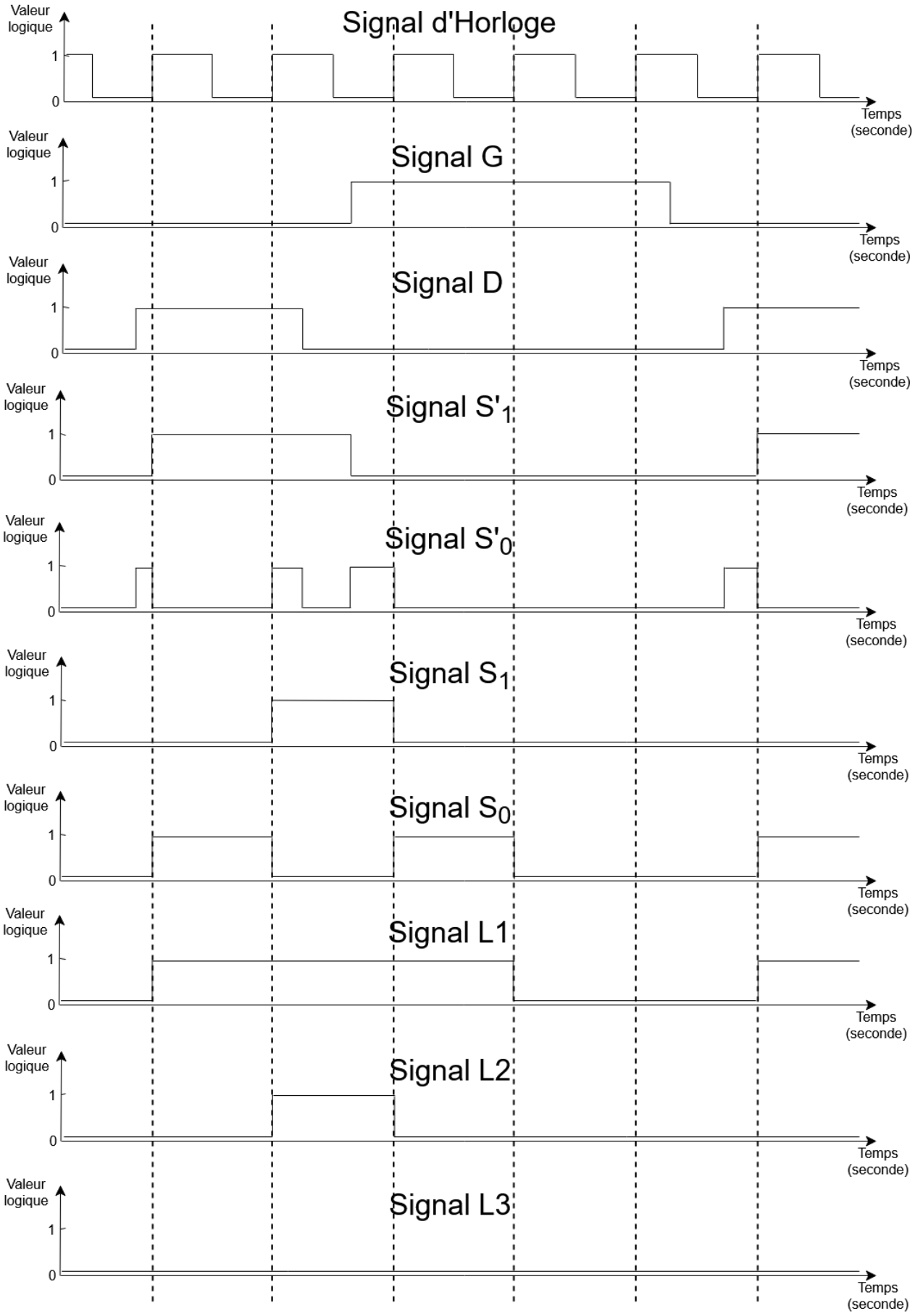
$S'_0(S_1,S_0,G,D) = \overline{S_0} \cdot \overline{G} \cdot D + S_0 \cdot \overline{G} \cdot \overline{D} + S_1 \cdot \overline{G} \cdot D + S_0 \cdot G \cdot D + S_1 \cdot \overline{S_0} \cdot G \cdot \overline{D}$

<u>Étape 7</u> : Logigramme



2) Lorsque l'utilisateur appuie sur les 2 boutons en même temps l'état ne vas pas changer, c'est aussi équivalent que lorsque l'utilisateur n'appuie sur aucun bouton. Cette opération n'est pas montrée sur l'Automate, en principe une Transition qui boucle sur le même état doit être faite pour illustrer ces 2 cas de figure, mais pour ne pas trop encombrer l'Automate ils ont été retirés.

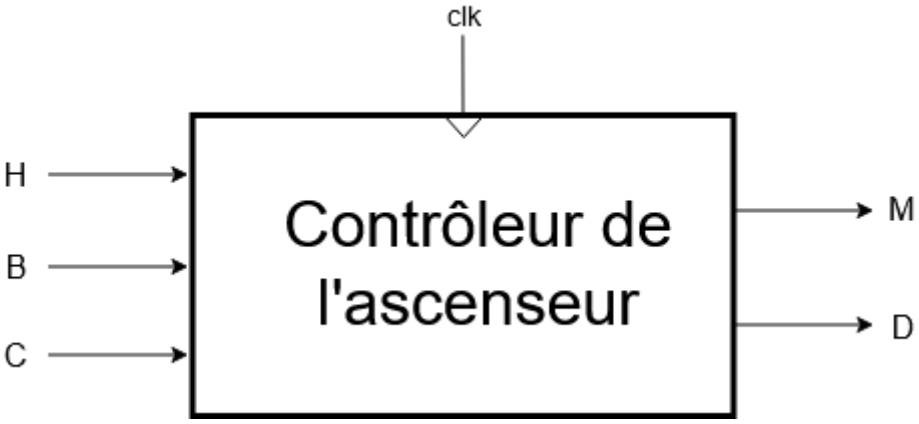
3)


Signal d'Horloge
Signal G
Signal D
Signal S'1
Signal S'0
Signal S1
Signal S0
Signal L1
Signal L2
Signal L3
Valeur logique
1
0
Temps (seconde)

Exercice 05 :

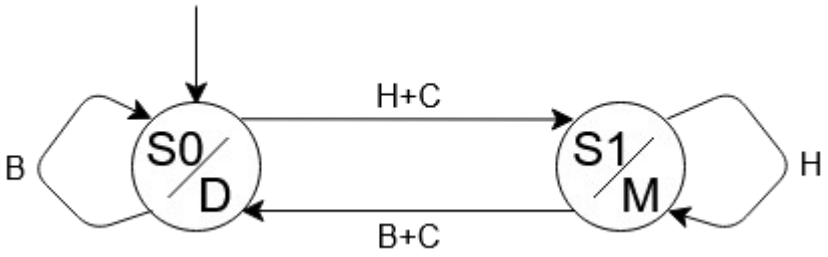
1) Machine de Moore pour l'ascenseur :

<u>Étape 1</u> : Schéma global



H : Étage en Haut
B : Étage en Bas
C : Changer d'étage
M : Monter
D : Descendre

<u>Étape 2</u> : Automate



S0 → L'ascenseur est au rez-de-chaussé
S1 → L'ascenseur est au premier étage

.
Remarque : Songer à une solution avec 4 états et aussi possible et correct. Un état lorsque l'ascenseur est en bas, un état où l'ascenseur remonte en haut, un état en haut, un état descendant en bas.

<u>Étape 3</u> : Table de Transition

État actuel	H	B	C	État suivant
S0	0	0	0	S0
	0	0	1	S1
	0	1	0	S0
	0	1	1	S1
	1	0	0	S1
	1	0	1	S1
	1	1	0	S0
	1	1	1	S1
S1	0	0	0	S1
	0	0	1	S0
	0	1	0	S0
	0	1	1	S0
	1	0	0	S1
	1	0	1	S0
	1	1	0	S1
	1	1	1	S0

Remarque 1: Il existe des cas conflictuels dans l'utilisation de l'ascenseur, comme par exemple si l'ascenseur est en bas et le contrôleur à une commande de changer d'étage et en même temps de descendre en bas (4-ième ligne), dans ce genre de situation la priorité est donnée à la commande de changer d'étage sur les autres.

Remarque 2: En cas de conflit entre une commande de monter et une de descendre en même temps (7-ième ligne), le contrôleur privilégie que l'ascenseur reste sur le même étage. Même en négligeant exceptionnellement l'expression sur les Transitions.

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

États	S	M	D
S0	0	0	1
S1	1	1	0

Remarque : Les formules de M et D sont directement devinables à partir de la Table de Sorties.

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

S	H	B	C	S'
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

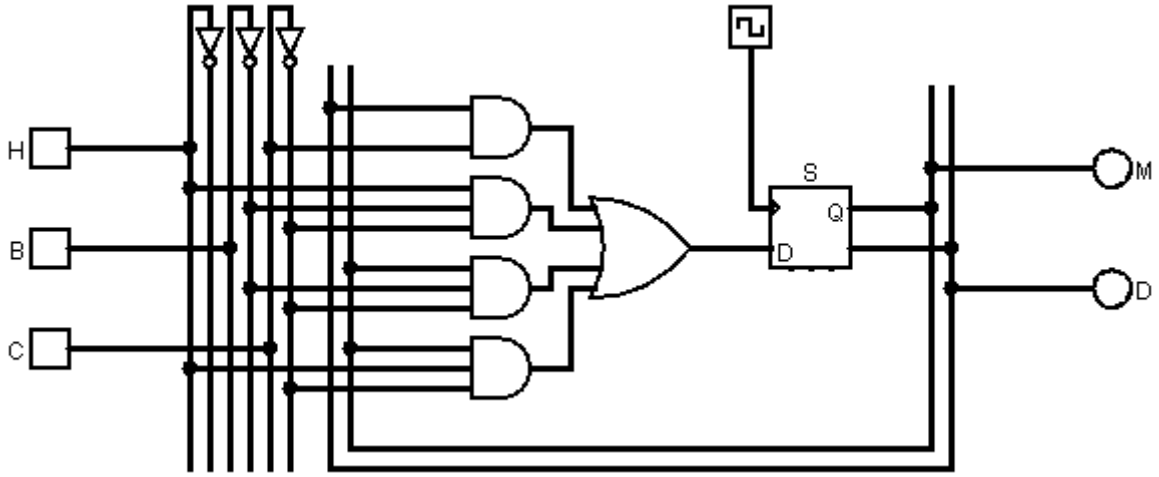
<u>Étape 6</u> : Formules Logiques

M = S
D = $\overline{S}$

SH / BC	00	01	11	10
00	0	1	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	0

$S'(S,H,B,C) = \overline{S}\cdot C + H\cdot\overline{B}\cdot\overline{C} + S\cdot\overline{B}\cdot\overline{C} + S\cdot H\cdot\overline{C}$

<u>Étape 7</u> : Logigramme

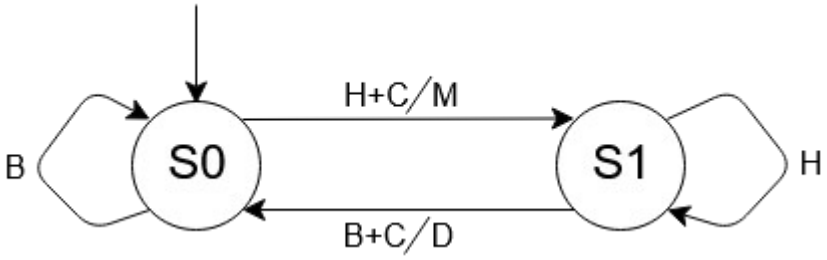


2) Machine de Mealy pour l'ascenseur :

<u>Étape 1</u> : Schéma global

Même schéma que celui de la machine de Moore.

<u>Étape 2</u> : Automate



<u>Étape 3</u> : Table de Transition

Même table que celle de la machine de Moore.

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

État	S	H	B	C	M	D
S0	0	0	0	0	0	0
		0	0	1	1	0
		0	1	0	0	0
		0	1	1	1	0
		1	0	0	1	0
		1	0	1	1	0
		1	1	0	0	0
		1	1	1	1	0
S1	1	0	0	0	0	0
		0	0	1	0	1
		0	1	0	0	1
		0	1	1	0	1
		1	0	0	0	0
		1	0	1	0	1
		1	1	0	0	0
		1	1	1	0	1

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

Même table que celle de la machine de Moore.

<u>Étape 6</u> : Formules Logiques

Les formules de Transition sont les mêmes : **$S'(S,H,B,C) = \overline{S}\cdot C + H\cdot\overline{B}\cdot\overline{C} + S\cdot\overline{B}\cdot\overline{C} + S\cdot H\cdot\overline{C}$**

Par contre les formules de la Tables des Sorties il faut les calculer :

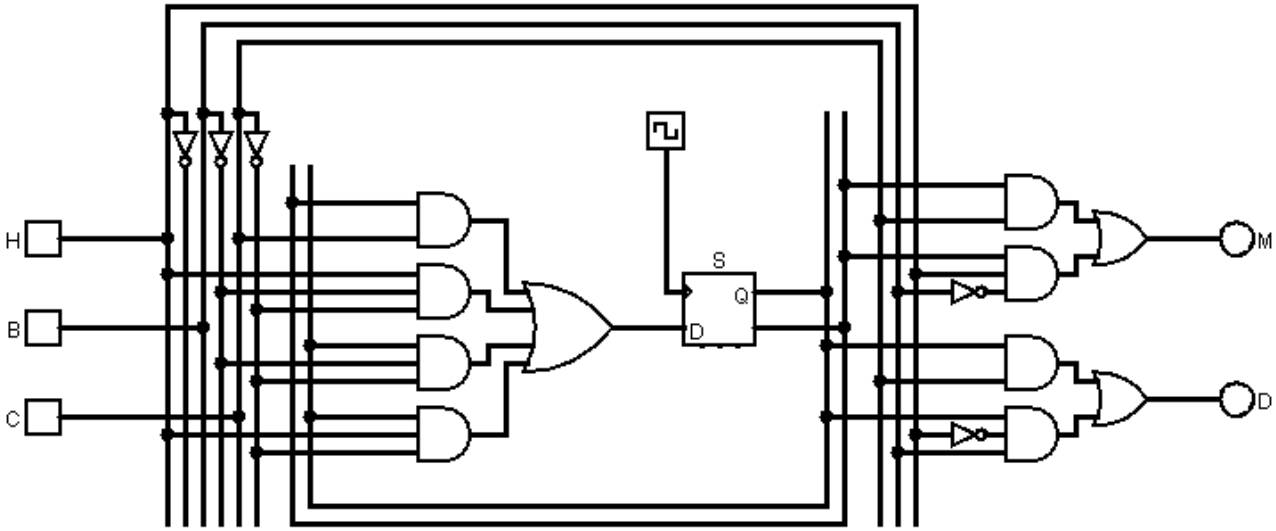
BC \ SH	00	01	11	10
00	0	1	0	0
01	1	1	0	0
11	1	1	0	0
10	0	0	0	0

$M(S,H,B,C) = \overline{S}\cdot C + \overline{S}\cdot H\cdot\overline{B}$

BC \ SH	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	0	0	1	1
10	0	0	0	1

$D(S,H,B,C) = S\cdot C + S\cdot\overline{H}\cdot B$

<u>Étape 7</u> : Logigramme



3) La principale différence entre la Machine de Moore et la Machine de Mealy dans leurs exécutions est lors du moment des sorties sur M et D, l'exécution de l'Automate est identique pour les 2 machines, mais la sortie est différente. La sortie dans la Machine de Moore est présente tout au long de la période où se trouve l'Automate, par exemple si l'Automate passe vers S1, tout au long de cette la période la sortie M=1 serait présente, il est de coutume de capturer la sortie lors des fronts montants, pour la machine de Moore ça serait lors du front montant de la fin de la période. Pour la même exécution, le passage de S0 vers S1 dans la machine de Mealy, le M=1 n'est garanti que lors du front montant entre les 2 états, dans ce cas ça représente le front montant du début de la période S1, et il ne peut être capturé qu'à ce moment précis.

4) Machine de Moore avec l'encodage One-hot :

<u>Étape 1</u> : Schéma global

Même schéma que celui de la machine de Moore.

<u>Étape 2</u> : Automate

Même Automate que celui de la machine de Moore.

<u>Étape 3</u> : Table de Transition

Même table que celle de la machine de Moore.

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

États	S_1	S_0	M	D
S0	0	1	0	1
S1	1	0	1	0

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

S_1	S_0	H	B	C	S'_1	S'_0
0	1	0	0	0	0	1
0	1	0	0	1	1	0
0	1	0	1	0	0	1
0	1	0	1	1	1	0
0	1	1	0	0	1	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	0	0	1	0	1
1	0	0	1	0	0	1
1	0	0	1	1	0	1
1	0	1	0	0	1	0
1	0	1	0	1	0	1
1	0	1	1	0	1	0
1	0	1	1	1	0	1

Remarque 1: Pour éviter une table trop longue, toutes les possibilités n'ont pas étés listées, comme S_1=0, S_1=0 ou S_1=1, S_1=1 qui sont des cas impossibles.

Remarque 2: On observe que sur les sorties de la table $S'_1 = \overline{S}'_0$, ainsi trouver l'un nous amène à trouve l'autre.

<u>Étape 6</u> : Formules Logiques

$M(S_1,S_0) = S_1$
$D(S_1,S_0) = S_0$

Le calcul de réduction de S'_1 et S'_0 nécessite l'utilisation de 5 variables, la seule méthode possible est la minimisation algébrique.

$$S'_1(S_1,S_0,H,B,C) = (\overline{S}_1 \cdot S_0 \cdot \overline{H} \cdot \overline{B} \cdot C + \overline{S}_1 \cdot S_0 \cdot \overline{H} \cdot B \cdot C) + (\overline{S}_1 \cdot S_0 \cdot H \cdot \overline{B} \cdot \overline{C} + \overline{S}_1 \cdot S_0 \cdot H \cdot \overline{B} \cdot C) + \overline{S}_1 \cdot S_0 \cdot H \cdot B \cdot C + S_1 \cdot \overline{S}_0 \cdot \overline{H} \cdot \overline{B} \cdot \overline{C} + (S_1 \cdot \overline{S}_0 \cdot H \cdot \overline{B} \cdot \overline{C} + S_1 \cdot \overline{S}_0 \cdot H \cdot B \cdot \overline{C})$$

$$S'_1(S_1,S_0,H,B,C) = \overline{S}_1 \cdot S_0 \cdot \overline{H} \cdot C + \overline{S}_1 \cdot S_0 \cdot H \cdot \overline{B} + \overline{S}_1 \cdot S_0 \cdot H \cdot B \cdot C + (S_1 \cdot \overline{S}_0 \cdot \overline{H} \cdot \overline{B} \cdot \overline{C} + S_1 \cdot \overline{S}_0 \cdot H \cdot \overline{C})$$

$$S'_1(S_1,S_0,H,B,C) = \overline{S}_1 \cdot S_0 \cdot \overline{H} \cdot C + \overline{S}_1 \cdot S_0 \cdot H \cdot \overline{B} + \overline{S}_1 \cdot S_0 \cdot H \cdot B \cdot C + S_1 \cdot \overline{S}_0 \cdot \overline{C} \cdot (\overline{H} \cdot \overline{B} + H)$$

$$S'_1(S_1,S_0,H,B,C) = \overline{S}_1 \cdot S_0 \cdot \overline{H} \cdot C + (\overline{S}_1 \cdot S_0 \cdot H \cdot \overline{B} + \overline{S}_1 \cdot S_0 \cdot H \cdot B \cdot C) + S_1 \cdot \overline{S}_0 \cdot \overline{C} \cdot (\overline{B} + H)$$

$$S'_1(S_1,S_0,H,B,C) = \overline{S}_1 \cdot S_0 \cdot \overline{H} \cdot C + \overline{S}_1 \cdot S_0 \cdot H \cdot (\overline{B} + B \cdot C) + S_1 \cdot \overline{S}_0 \cdot \overline{C} \cdot (\overline{B} + H)$$

$$S'_1(S_1,S_0,H,B,C) = \overline{S}_1 \cdot S_0 \cdot \overline{H} \cdot C + \overline{S}_1 \cdot S_0 \cdot H \cdot (\overline{B} + C) + S_1 \cdot \overline{S}_0 \cdot \overline{C} \cdot (\overline{B} + H)$$

$$S'_1(S_1,S_0,H,B,C) = \overline{S}_1 \cdot S_0 \cdot \overline{H} \cdot C + \overline{S}_1 \cdot S_0 \cdot H \cdot \overline{B} + \overline{S}_1 \cdot S_0 \cdot H \cdot C + S_1 \cdot \overline{S}_0 \cdot \overline{C} \cdot (\overline{B} + H)$$

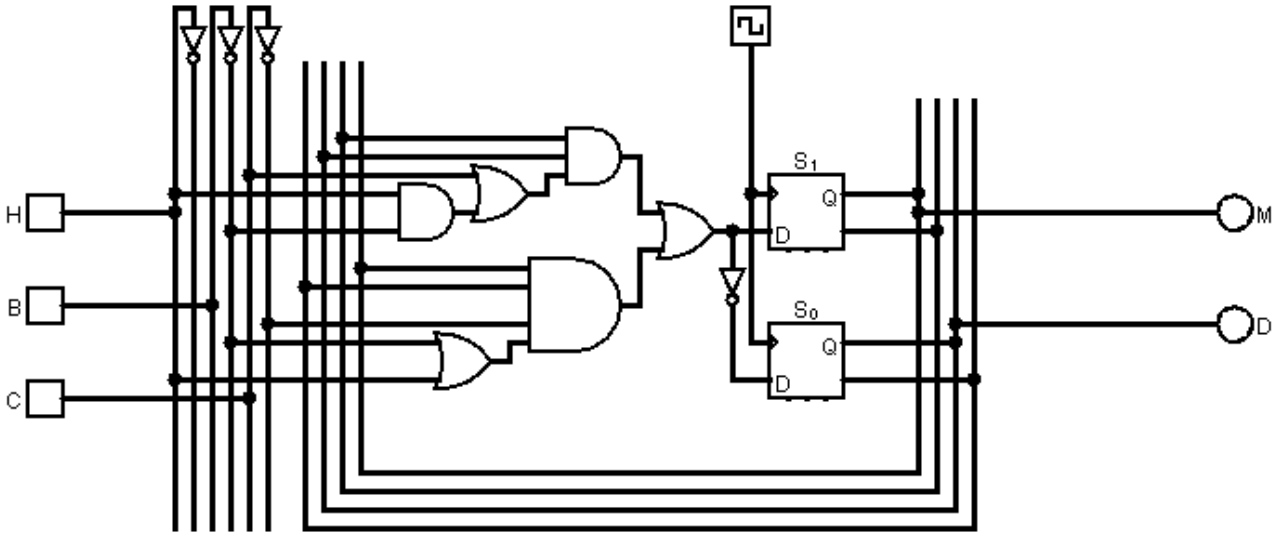
$$S'_1(S_1,S_0,H,B,C) = (\overline{S}_1 \cdot S_0 \cdot \overline{H} \cdot C + \overline{S}_1 \cdot S_0 \cdot H \cdot C) + \overline{S}_1 \cdot S_0 \cdot H \cdot \overline{B} + S_1 \cdot \overline{S}_0 \cdot \overline{C} \cdot (\overline{B} + H)$$

$$S'_1(S_1,S_0,H,B,C) = (\overline{S}_1 \cdot S_0 \cdot C + \overline{S}_1 \cdot S_0 \cdot H \cdot \overline{B}) + S_1 \cdot \overline{S}_0 \cdot \overline{C} \cdot (\overline{B} + H)$$

$$\mathbf{S'_1(S_1,S_0,H,B,C) = \overline{S}_1 \cdot S_0 \cdot (C + H \cdot \overline{B}) + S_1 \cdot \overline{S}_0 \cdot \overline{C} \cdot (\overline{B} + H)}$$

$$\mathbf{S'_0(S_1,S_0,H,B,C) = \overline{S}'_1(S_1,S_0,H,B,C)}$$

<u>Étape 7</u> : Logigramme



Machine de Mealy avec l'encodage One-hot :

<u>Étape 1</u> : Schéma global

Même schéma que celui de la machine de Mealy.

<u>Étape 2</u> : Automate

Même Automate que celui de la machine de Mealy.

<u>Étape 3</u> : Table de Transition

Même table que celle de la machine de Mealy.

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

État	S_1	S_0	H	B	C	M	D
S0	1	0	0	0	0	0	0
			0	0	1	1	0
			0	1	0	0	0
			0	1	1	1	0
			1	0	0	1	0
			1	0	1	1	0
			1	1	0	0	0
			1	1	1	1	0
S1	0	1	0	0	0	0	0
			0	0	1	0	1
			0	1	0	0	1
			0	1	1	0	1
			1	0	0	0	0
			1	0	1	0	1
			1	1	0	0	0
			1	1	1	0	1

Remarque 1: Pour ce cas, la table d'Encodage des États est pratiquement l'inverse ou le complément de la table d'Encodage des États de la machine en One-hot, ça va rendre très facile l'extraction des formules de l'État Suivant.

Remarque 2: La Table des Sorties est quasiment identique à celle de la machine de Mealy, à part dans l'encodage des états.

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

S_1	S_0	H	B	C	S'_1	S'_0
1	0	0	0	0	0	1
1	0	0	0	1	1	0
1	0	0	1	0	0	1
1	0	0	1	1	1	0
1	0	1	0	0	1	0
1	0	1	0	1	1	0
1	0	1	1	0	0	1
1	0	1	1	1	1	0
0	1	0	0	0	1	0
0	1	0	0	1	0	1
0	1	0	1	0	0	1
0	1	0	1	1	0	1
0	1	1	0	0	1	0
0	1	1	0	1	0	1
0	1	1	1	0	1	0
0	1	1	1	1	0	1

Remarque : La table de Transition Encodée est identique à celle de la table de la machine en One-hot, à l'exception de l'encodage de S0 et S1 qui sont inversés.

<u>Étape 6</u> : Formules Logiques

On peut utiliser les mêmes équations de la machine de Mealy avec $S=\overline{S}_1 \cdot S_0$ et $\overline{S}=S_1 \cdot \overline{S}_0$

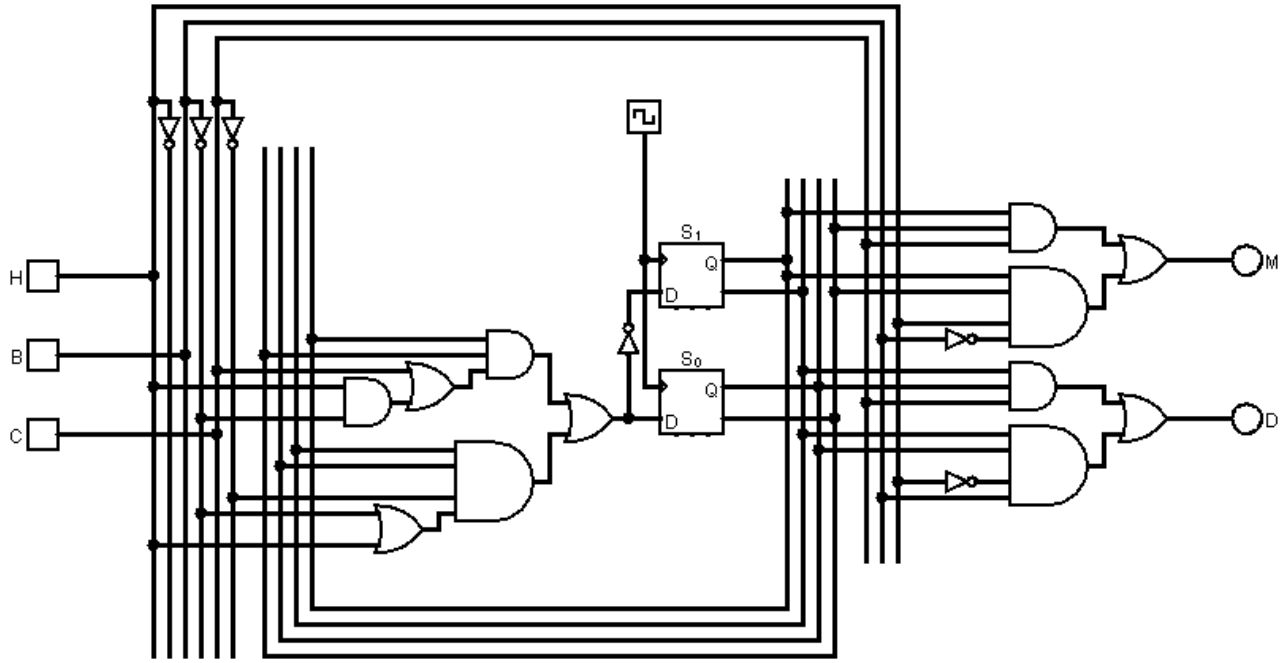
$$\mathbf{M(S_1,S_0,H,B,C) = S_1 \cdot \overline{S}_0 \cdot C + S_1 \cdot \overline{S}_0 \cdot H \cdot \overline{B}}$$
$$\mathbf{D(S_1,S_0,H,B,C) = \overline{S}_1 \cdot S_0 \cdot C + \overline{S}_1 \cdot S_0 \cdot \overline{H} \cdot B}$$

Remarque : Ici $S=\overline{S}_1 \cdot S_0$ parce que si S=1 alors l'Automate est dans l'état S1 donc $(S_1,S_0)=0,1$ exprimé par $\overline{S}_1 \cdot S_0$, et inversement pour S0.

Les formules pour l'État Suivant sont à l'inverse de la machine à One-hot, ainsi S_1 devient $\overline{S}_1$ et S_1 devient $\overline{S}_1$, et de même pour S_0 :

$$\mathbf{S'_0(S_1,S_0,H,B,C) = S_1 \cdot \overline{S}_0 \cdot (C + H \cdot \overline{B}) + \overline{S}_1 \cdot S_0 \cdot \overline{C} \cdot (\overline{B} + H)}$$
$$\mathbf{S'_1(S_1,S_0,H,B,C) = \overline{S'}_0(S_1,S_0,H,B,C)}$$

Étape 7 : Logigramme



5) Le tableau suivant résume le nombre de portes et de cellules mémoires pour les 4 circuits conçus :

Machine - Encodage	Porte logique	Cellule mémoire
Moore – Normal	8	1
Mealy – Normal	14	1
Moore – One-hot	10	2
Mealy – One-cold	16	2

On peut conclure à partir du tableau que la machine de Moore avec encodage normal est la plus optimale entre les 4, que ça soit dans le nombre de portes logiques ou dans le nombre de cellules mémoires.

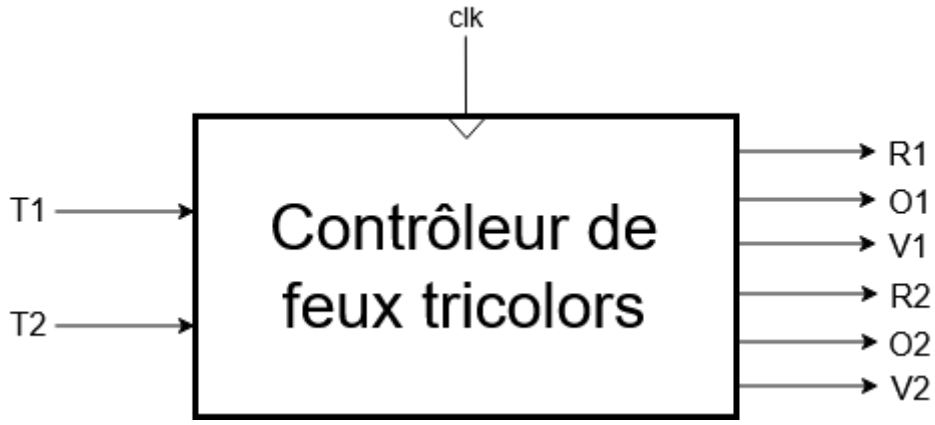
Remarque 1: Les portes NOT dans les 2 machines de Mealy dans le circuit de Sortie ne sont pas comptées, en raison que réellement le signal inverse des entrées normalement doit venir des signaux d'entrée, ils ont étés rajoutées juste pour simplifier le diagramme.

Remarque 2: La règle dans la construction des Circuit Séquentiels est que l'encodage en One-hot ou One-cold augmente le nombre de cellules mémoires et diminue le nombre de portes logiques, ce n'était pas le cas pour ces circuits mais la règle reste correcte pour les circuits plus complexes.

Exercice 06 :

1)

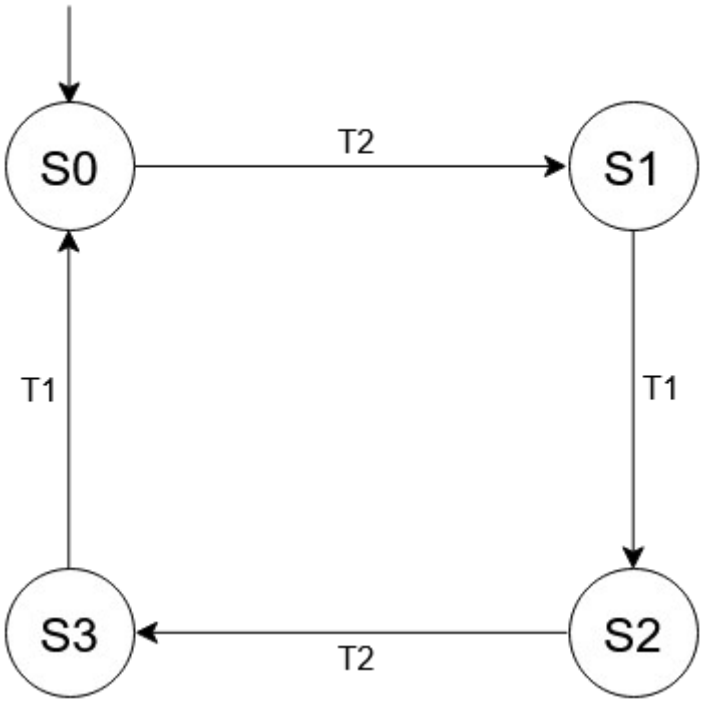
<u>Étape 1</u> : Schéma global



T : Timer
R : Couleur Rouge
O : Couleur Orange
V : Couleur Verte

R1, O1, V1 : les feux de F1 et F4
R2, O2, V2 : les feux de F2 et F3

<u>Étape 2</u> : Automate



S0 → R1=1, O1=0, V1=0 (F1 et F4 sont au rouge)
R2=0, O2=0, V2=1 (F2 et F3 sont au vert)

S1 → R1=1, O1=0, V1=0 (F1 et F4 sont au rouge)
R2=0,O2=1,V2=0 (F2 et F3 sont à l'orange)

S2 → R1=0, O1=0, V1=1 (F1 et F4 sont au vert)
R2=1, O2=0, V2=0 (F2 et F3 sont au rouge)

S3 → R1=0, O1=1, V1=0 (F1 et F4 sont à l'orange)
R2=1, O2=0, V2=0 (F2 et F3 sont au rouge)

Remarque 1: Le Timer 1 va produire un 1 à la fin de chaque 30 secondes, ça sera le signe au circuit de basculer du rouge au vert et inversement. Le Timer 2 est aussi de 30 secondes, il est en retard de 27 secondes ,ou en avance de 3 secondes (puisque c'est un signal périodique) par rapport au Timer 1, il est responsable du timing du feu orange.

Remarque 2: Les signaux R1, O1 et V1 sont identiques pour F1 et F4, et de même pour R2, O2 et V2 avec F2 et F3.

<u>Étape 3</u> : Table de Transition

État actuel	T1	T2	État suivant
S0	0	0	S0
	0	1	S1
	1	0	-
	1	1	-
S1	0	0	S1
	0	1	-
	1	0	S2
	1	1	-
S2	0	0	S2
	0	1	S3
	1	0	-
	1	1	-
S3	0	0	S3
	0	1	-
	1	0	S0
	1	1	-

Remarque 1: T1 et T2 sont des Timer et c'est impossible selon la disposition de notre circuit que les signaux T1 et T2 arrivent en même temps, ou que T2 dépasse T1. Ces cas impossibles sont représentés par des très don't care dans la table qui vont nous aider après dans la minimisation.

Remarque 2: Pour des raisons de simplicité les Transitions en boucle pour les cas T1=0 et T2=0 dans les 4 états n'étaient pas représentées sur l'Automate, mais ces Transitions sont réelles et elles sont à prendre en considération dans la Table de Transition.

Remarque 3: Souvent les Transitions en boucle qui ne font pas changer d'état sont ignorés dans l'Automate, mais ils sont toujours à mentionner dans la Table de Transition.

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

États	S_1	S_0	R1	O1	V1	R2	O2	V2
S0	0	0	1	0	0	0	0	1
S1	0	1	1	0	0	0	1	0
S2	1	0	0	0	1	1	0	0
S3	1	1	0	1	0	1	0	0

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

S_1	S_0	T1	T2	S'_1	S'_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	-	-
0	0	1	1	-	-
0	1	0	0	0	1
0	1	0	1	-	-
0	1	1	0	1	0
0	1	1	1	-	-
1	0	0	0	1	0
1	0	0	1	1	1
1	0	1	0	-	-
1	0	1	1	-	-
1	1	0	0	1	1
1	1	0	1	-	-
1	1	1	0	0	0
1	1	1	1	-	-

<u>Étape 6</u> : Formules Logiques

$R1(S_1,S_0) = \overline{S_1}$
$O1(S_1,S_0) = S_1 \cdot S_0$
$V1(S_1,S_0) = S_1 \cdot \overline{S_0}$

$R2(S_1,S_0) = S_1$
$O2(S_1,S_0) = \overline{S_1} \cdot S_0$
$V2(S_1,S_0) = \overline{S_1 + S_0}$

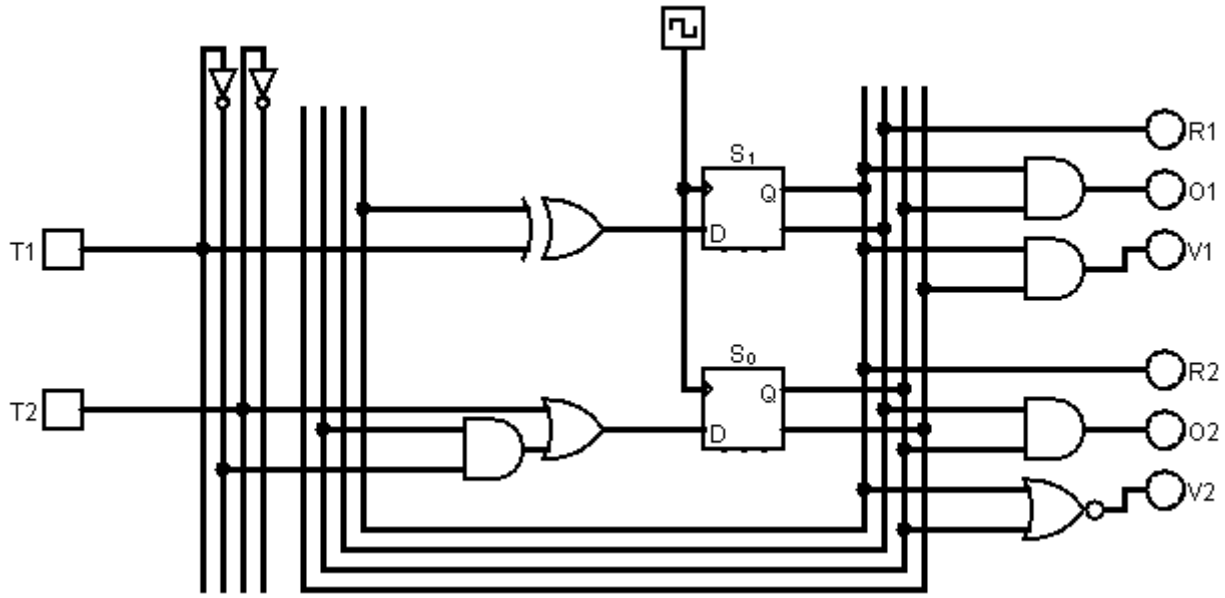
S_1S_0 / T1T2	00	01	11	10
00	0	0	1	1
01	0	-	-	1
11	-	-	-	-
10	-	1	0	-

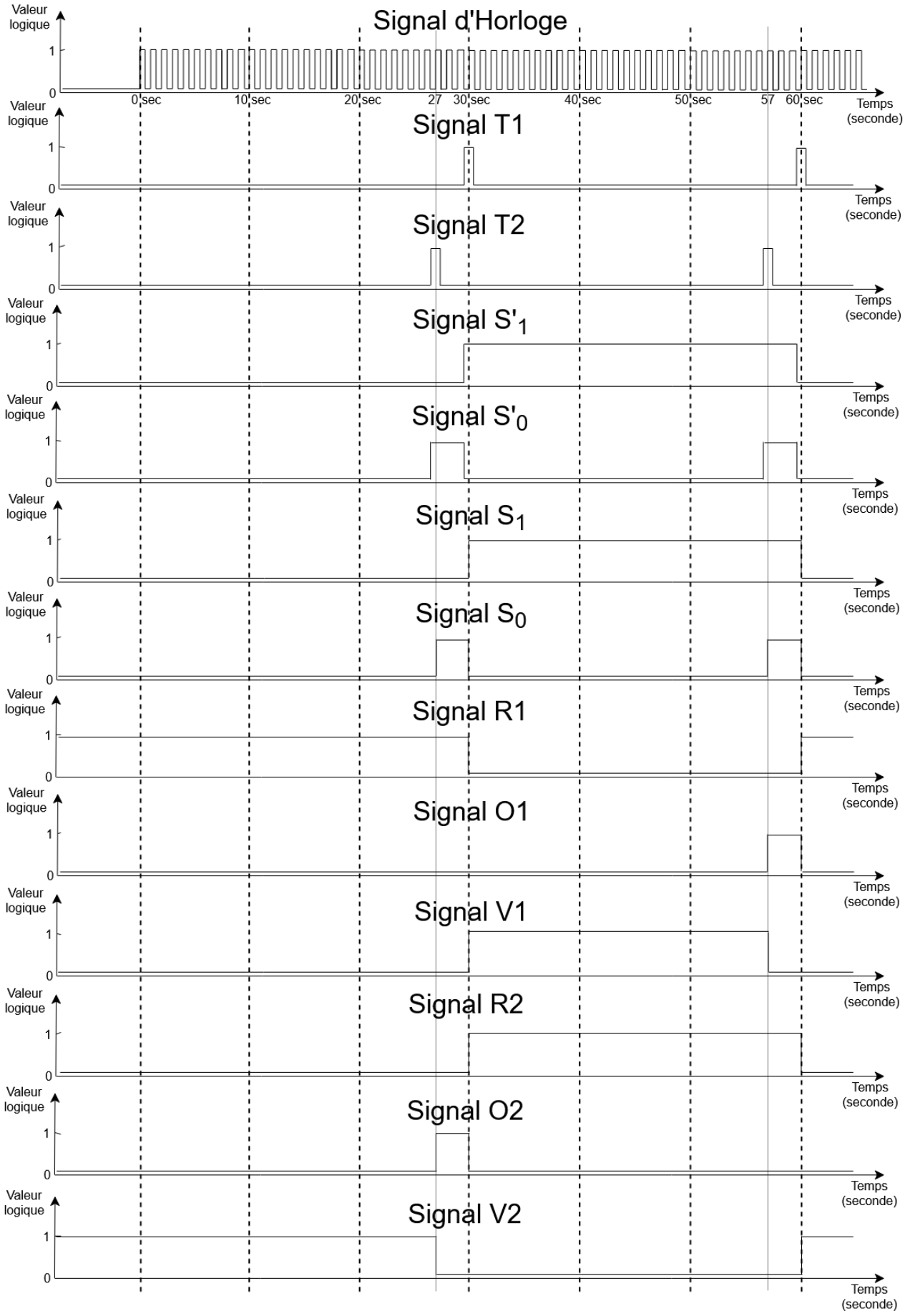
S_1S_0 / T1T2	00	01	11	10
00	0	1	1	0
01	1	-	-	1
11	-	-	-	-
10	-	0	0	-

$S'_1(S_1,S_0,T1,T2) = S_1 \cdot \overline{T1} + \overline{S_1} \cdot T1 = S_1 \oplus T1$

$S'_0(S_1,S_0,T1,T2) = T2 + S_0 \cdot \overline{T1}$

<u>Étape 7</u> : Logigramme




Signal d'Horloge
Valeur logique
1
0
0 sec
10 sec
20 sec
27
30 sec
40 sec
50 sec
57
60 sec
Temps (seconde)
Signal T1
Signal T2
Signal S'1
Signal S'0
Signal S1
Signal S0
Signal R1
Signal O1
Signal V1
Signal R2
Signal O2
Signal V2

Remarque 1: La fréquence de l'horloge n'était pas indiquée dans l'élancé de l'exercice, mais on peut déduire que la période ne peut pas dépasser les 3 secondes en raison que le changement de signal minimal dans le circuit est de 3 secondes, celui feu orange. On a choisi 1 seconde comme période, donc une fréquence de 1 Hertz.

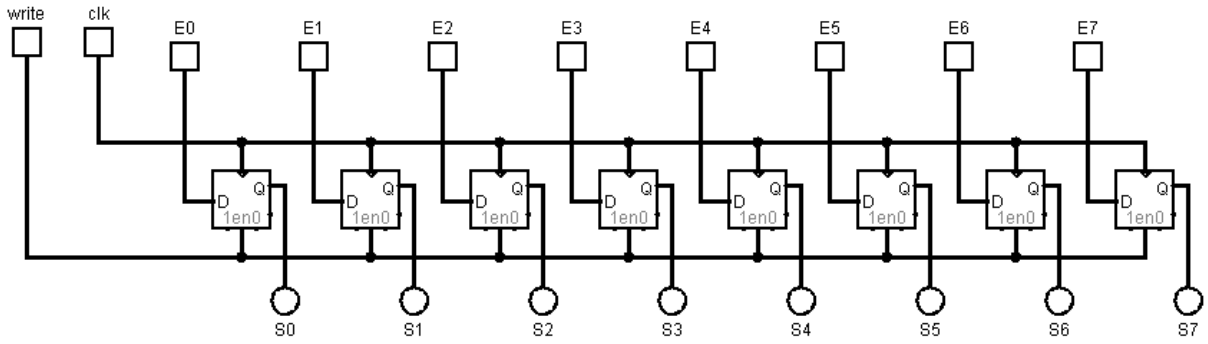
Remarque 2: Les signaux des 2 Timers sont décalés et avancés d'une demi-période par rapport au signal d'horloge, ça permet de donner assez de temps pour les D-FlipFlop et au signal dans les circuits combinatoires pour se propager correctement.

Remarque 3: Le suivi des signaux des cellules mémoires et des sorties du Circuit Combinatoire de Sortie sont assez facile, ils sont fixes sur toute une période. Par contre les signaux S'_1 et S'_0 du Circuit Combinatoire de l'État Suivant sont plus difficiles à suivre, parce qu'il faut les mettre-à-jour à chaque changement dans les entrées T1 et T2 et dans les cellules mémoires S_1 et S_0.

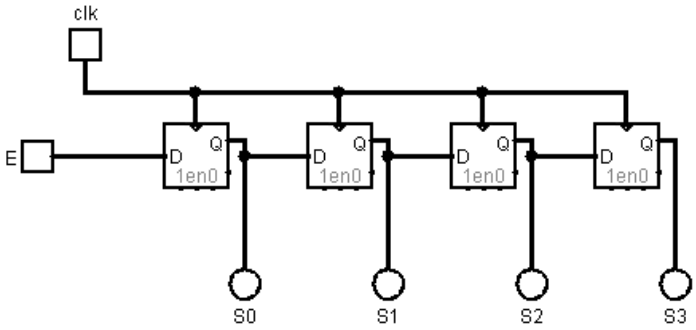
Remarque 4: Pour éviter trop de complications dans le chronogramme les signaux S'_1 et S'_0 n'étaient pas mis-à-jour dans la petite partie de la demi-période après T1 et T2.

Exercice 06 :

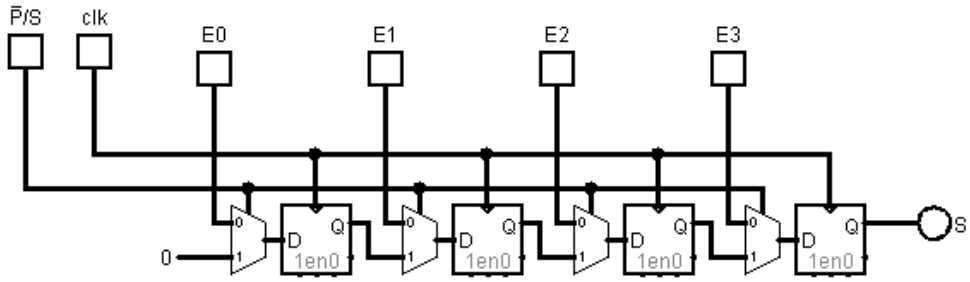
1) Le circuit Registre 8 bits avec la commande Write :



2) Le circuit Shift 4 bits de type SIPO :



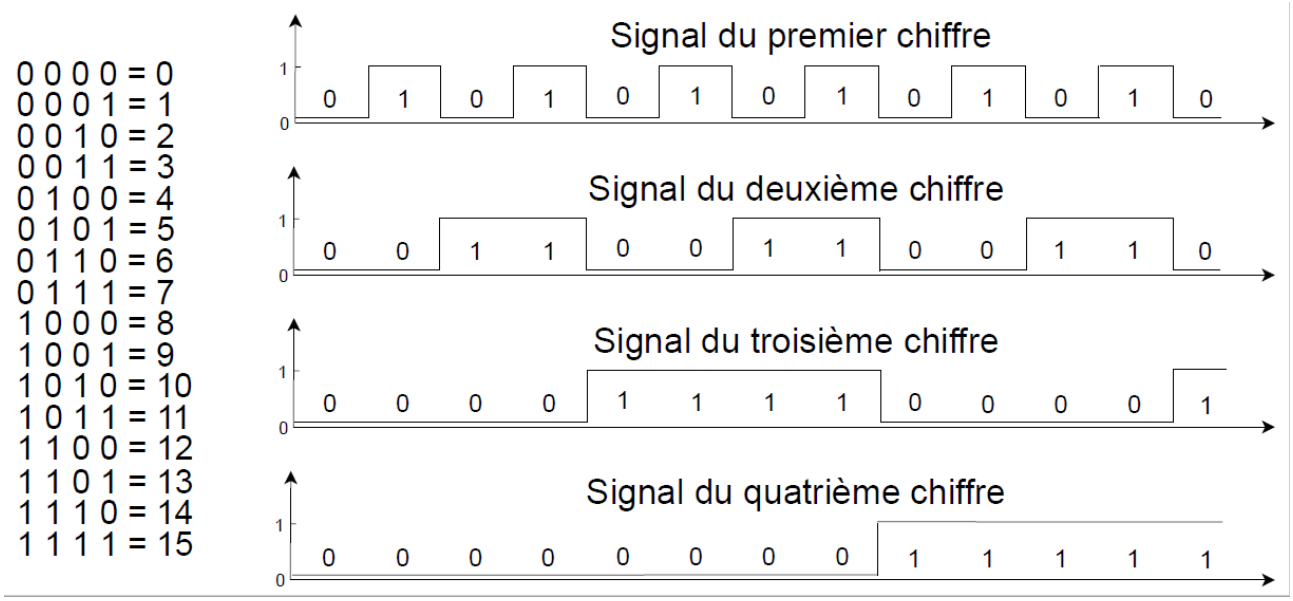
Le circuit Shift 4 bits de type PISO :



Remarque 1: L'entrée de commande $\overline{P}/S$ ($\overline{\text{Parallèle}}$/Série) permet de choisir entre, faire entrer en parallèle les 4 entrées, ou de faire décaler les valeurs à la chaîne à travers les cellules pour les faire sortir en série sur la sortie S.

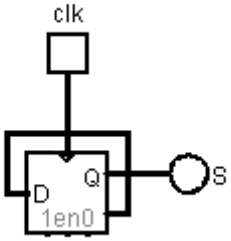
Remarque 2: La barre dans $\overline{P}/S$ indique quelle fonctionnalité est choisie par la valeur 0, ainsi si l'entrée est mise-à-0 c'est l'entrée en Parallèle qui va être utilisée.

3) Pour construire un Compteur il faut bien étudier et observer le format binaire de l'opération d'incrémentation. On peut voir les nombres qui s'incrémentent sur la figure en bas à gauche :

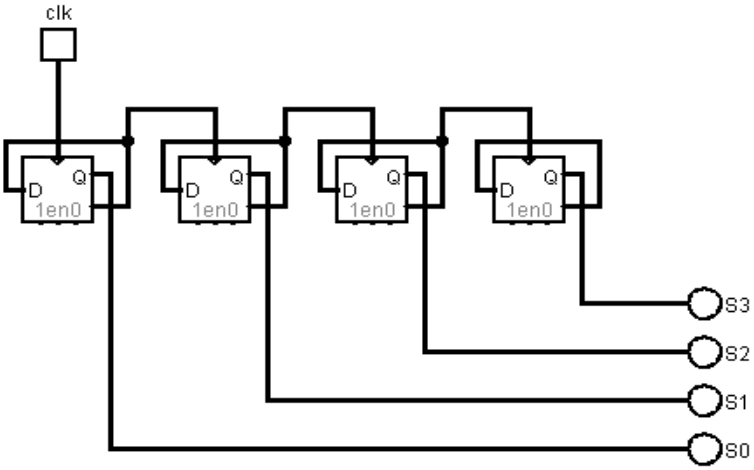


Si on se focalise sur une seule colonne pour un chiffre donné, par exemple sur le premier chiffre, on peut observer que c'est un signal périodique identique au signal d'horloge, ainsi que pour la deuxième colonne, avec la différence qu'il soit le double du premier, et le troisième aussi fait le double du deuxième, et ainsi de suite. Les signaux de tous ses chiffres sur 4 bits sont transcrit sur la figure en haut à droite.

Ainsi pour créer un Compteur, il suffit de générer un signal d'horloge pour chaque chiffre, avec la condition que le chiffre suivant doit avoir une période double du chiffre précédent. Pour créer un signal d'horloge c'est assez simple, il suffit d'utiliser une D-FlipFlop comme sur le schéma en bas, la sortie $\overline{Q}$ doit boucler vers l'entrée D, ainsi pour chaque Front Montant la FlipFlop va prendre la valeur inverse de la valeur actuelle.

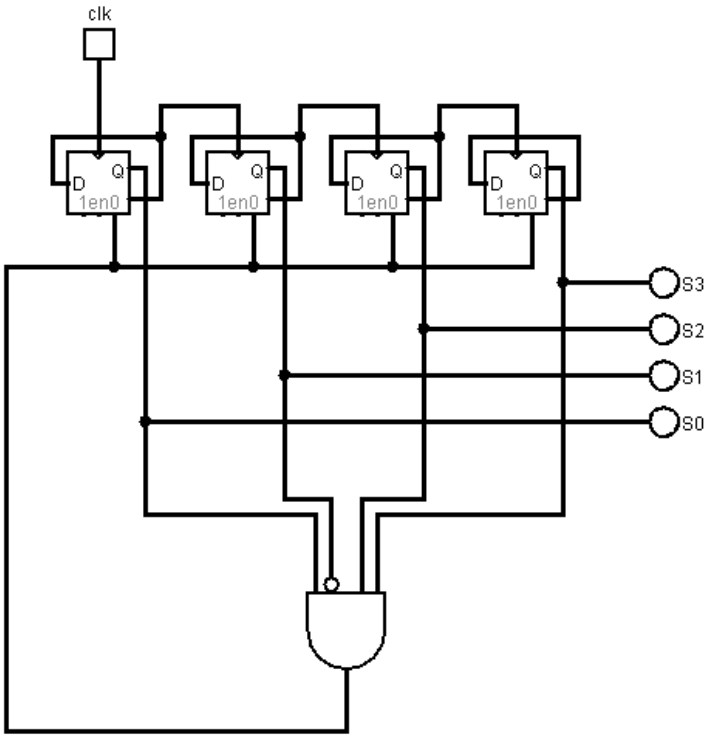


Pour que le signal du chiffre suivant prend le double de période que le chiffre précédent il suffit d'utiliser la sortie du chiffre précédent comme horloge pour le chiffre suivant, comme sur la figure en bas, malgré que sur la figure on prend le signal $\overline{Q}$ au-lieu de Q, en raison qu'en observant le graphe des signaux en haut on remarque que le signal suivant réellement change de valeur lors du Front Descendant du signal précédent, ça nous oblige ainsi à utiliser le signal inverse $\overline{Q}$ au lieu de Q.



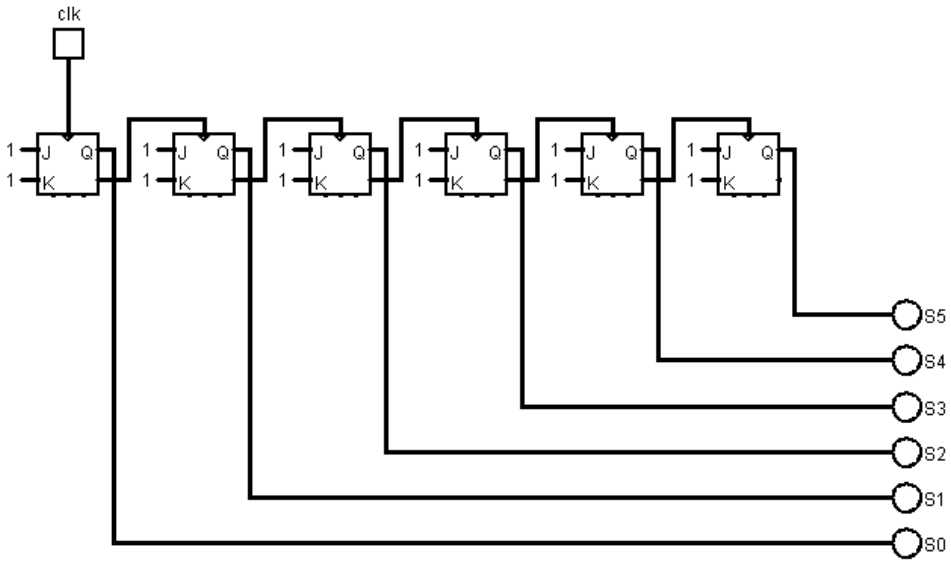
Pour limiter le comptage à la valeur 12 (1100), il est judicieux d'utiliser la commande Reset des FlipFlip qui va les remettre à 0 lorsque le circuit détecte qu'il a atteint la limite. La limite devrait être ici la valeur 13, la valeur juste après 12, qui devrait simplement être détectée par une porte AND. Le Reset utilisé ici est Asynchrone, ainsi il va réinitialiser la cellule immédiatement après avoir détecté la valeur 13, de cette manière elle n'aura jamais le temps de perdurer à la sortie du Compteur.

Le schéma un Compteur sur 4 bits qui compte jusqu'à la valeur 12 :

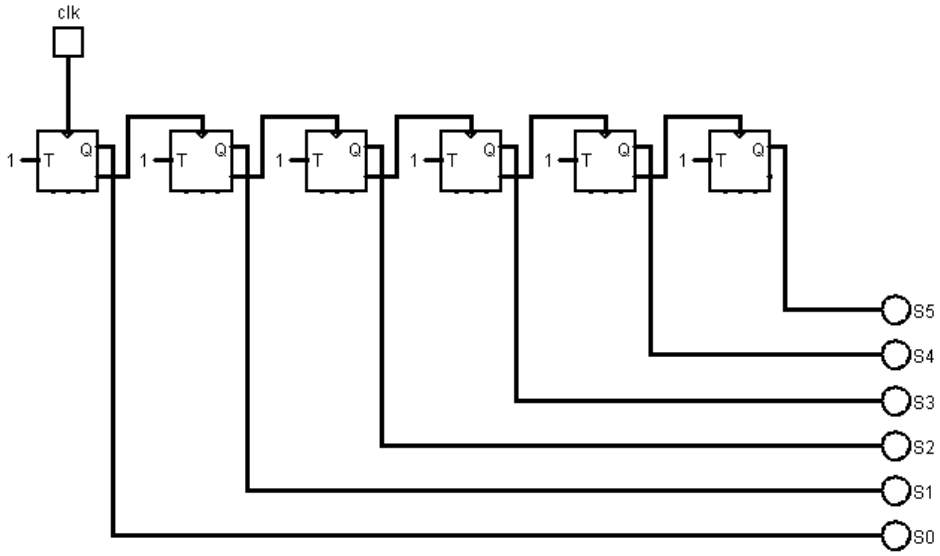


Remarque 3: Le Reset dans cette solution est supposé être Asynchrone, si c'était une commande Synchrone la solution devrait être différente.

4) Le circuit du Compteur 6 bits à base de JK-FlipFlop :



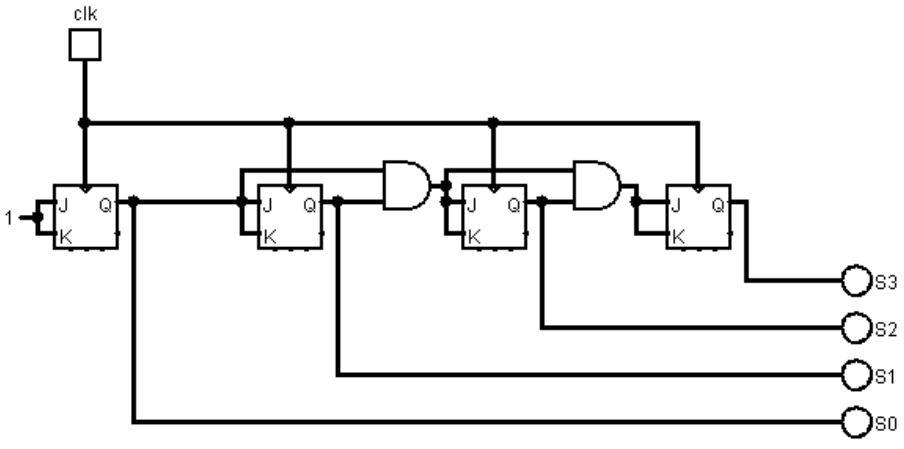
Le circuit du Compteur 6 bits à base de T-FlipFlop :



Remarque 4: Le Compteur a cette faculté de faire osciller ses sorties en doublant de période pour chaque sortie, ce qui lui procure la faculté d'être utilisé très souvent comme un diviseur de fréquences. Par exemple si on applique une fréquence de 1 KHz à son horloge, la première sortie va produire un signal de 500 Hz, la deuxième de 250 Hz, la troisième de 125 Hz, et ainsi de suite.

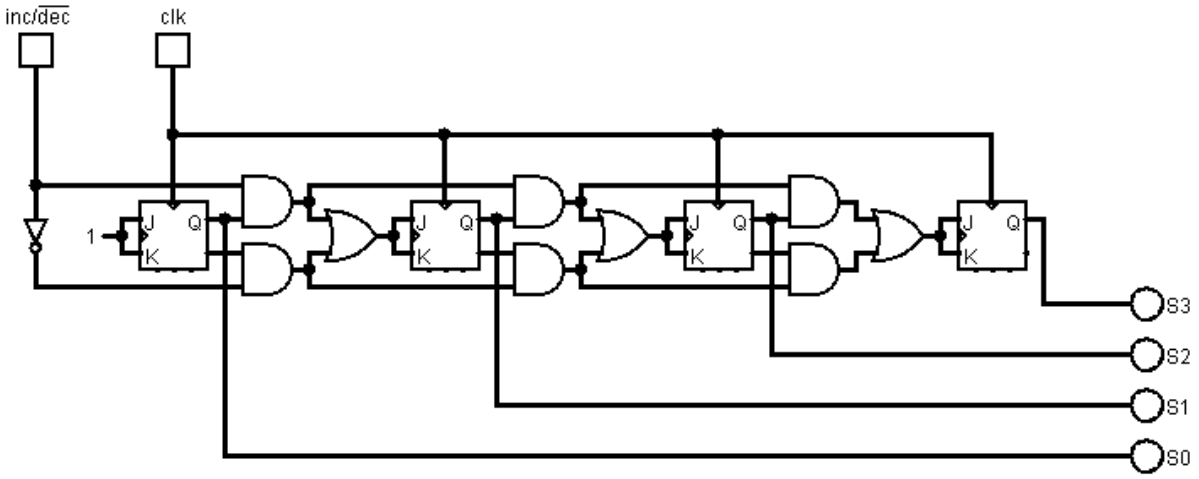
Remarque 5: Les 3 types assez simples de Compteur qu'on vient de construire sont le plus souvent appelés Compteurs Asynchrones, ou Ripple Counters en Anglais (Ripple étant une vague), en raison que la sortie d'une cellule est l'horloge de la suivante, ce qui va provoquer un petit retard d'horloge qui va s'accumuler à travers les cellules. Pour les circuits à faible fréquence inférieure à 1 MHz, normalement ça ne devrait pas poser de problèmes, mais pour les circuits à haute fréquence ils deviennent source de défaillance, les Compteurs dits Synchrones qui sont plus complexes sont généralement plus utilisés. Synchrone implique que toutes les cellules utilisent le même signal d'horloge.

5) Le circuit du Compteur Synchrone sur 4 bits à base de JK-FlipFlop :



La principale différence entre le Compteur Asynchrone et le Compteur Synchrone est entre autres, le fait que le signal d'horloge étant le même pour toutes les FlipFlops dans le Compteur Synchrone, et mis en disposition en cascade dans le Compteur Asynchrone, ce qui crée dans ce dernier un délai cumulatif à travers la succession des FlipFlops. L'approche pour construire le Compteur Synchrone est différente de l'Asynchrone, dans le sens où un autre principe concernant le comptage binaire a été exploité. Sur le logigramme on peut observer les portes AND, leurs rôle est de s'activer que lorsque tous les chiffres précédents la cellule actuelle sont à 1, et faire basculer la valeur dans la cellule. Ce principe de comptage se base sur le fait qu'un chiffre bascule de valeur en que lorsque tous les chiffres précédents sont à 1 (c'est comme avoir 99⋯9+1 dans le système décimal). Une porte AND s'assure que tous les chiffres précédents sont à 1 par un procédé de disposition en cascade, ainsi le AND s'assure que la cellule juste avant lui contient 1 et le AND précédent s'assure d'une manière récursive que les autres cellules précédentes contiennent 1 aussi.

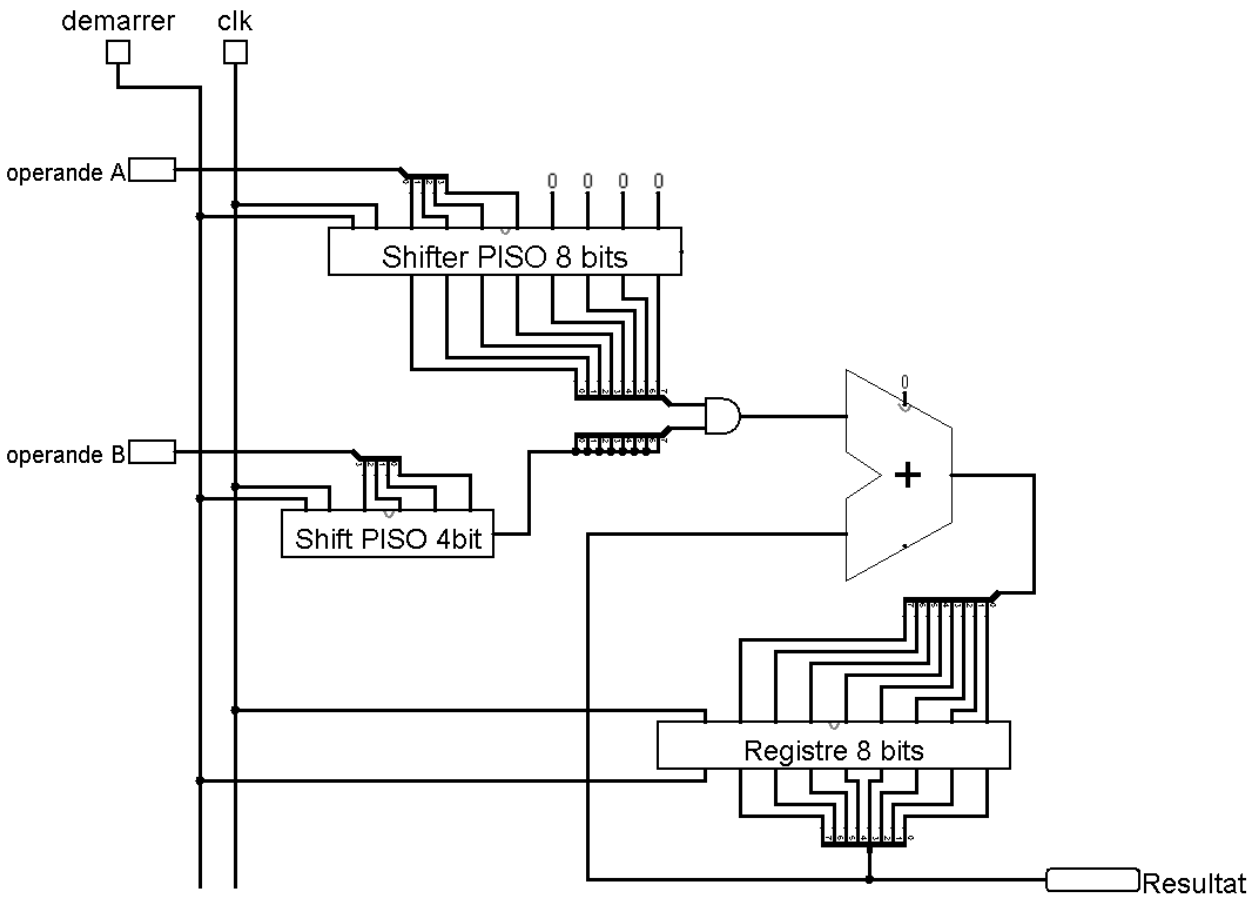
6) Le circuit du Compteur Universel Synchrone sur 4 bits :



Remarque 1: Le circuit est visuellement verticalement symétrique, les portes AND en haut sont utilisées pour calculer l'incrémentation et celles d'en bas pour calculer la décrémentation. L'entrée inc/$\overline{dec}$ est responsable activer la cascade de portes voulue, celle d'en haut ou celle d'en bas. Les portes OR au milieu sont utilisées uniquement pour faire passer les signaux de la partie active du circuit.

Remarque 2: La décrémentation fonctionne d'une manière similaire à l'incrémentation à l'exception d'utiliser le $\overline{Q}$ comme signal d'activation. Qui va normalement basculer la valeur de tous les premiers chiffres (avec le poids le plus faible) suivant le format 10⋯0, et les transformer en 01⋯1. Ce qui revient d'une autre manière, à exprimer l'opération de décrémentation.

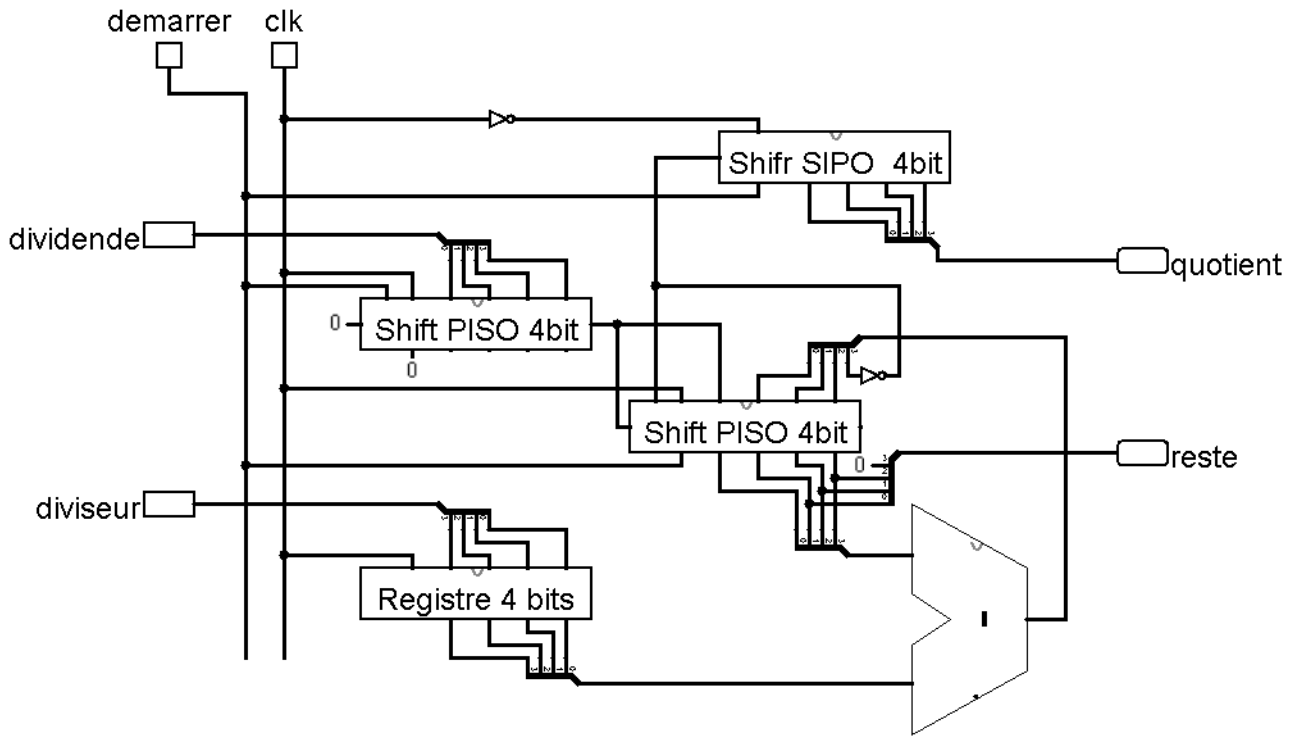
7) Le circuit du Multiplicateur Séquentiel sur 4 bits :



Remarque 1: Le circuit du Multiplicateur émule le processus de calcul à la main de la multiplication binaire. Ainsi le Shifteur PISO 4 bits va contenir B et faire sortir sa valeur bit par bit pour chaque Front Montant d'Horloge, tel-que chaque bit sorti est multiplié par A. Le Shifteur PISO 8 bits va contenir A, qui va être décalée à gauche à chaque Front Montant, contribuant à la somme lorsque A est multiplié par les chiffres de B, et mémorisée dans Le Registre 8 bits. La somme est calculée en boucle sur 4 répétitions en utilisant le circuit de l'Additionneur. La porte AND, en fonction du chiffre B en cours, s'il est à 0, l'Additionneur va rajouter 0, sinon si c'est 1, c'est la valeur décalée de A qui va être additionnée. Le taux de décalage de A est en rapport avec la position du chiffre B en cours.

Remarque 2: Le signal *démarrer* doit être activé au début et synchronisé avec l'horloge, pour chargé les valeurs de A et B dans les Shifteur, et réinitialiser le Registre 8 bits.

Le circuit du Diviseur Séquentiel sur 4 bits :



Remarque : Le circuit du Diviseur aussi était conçu en suivant la manière de faire le calcul à la main de la division binaire. Le Shifteur PISO 4 bits le plus à gauche est utilisé pour mémoriser le dividende, le 2$^{\text{-ième}}$ pour contenir le quotient, le Registre 4 bits pour le diviseur, et le Shifteur PISO 4 bits pour le reste. L'algorithme de division suit 3 étapes. La première, est de décaler à gauche 1 bit du dividende et de l'insérer en série de droite à gauche dans le Shifteur du reste. La deuxième étape, commence par faire la comparaison en utilisant le soustracteur, entre le reste et le diviseur. Le bit de signe est utilisé comme indicateur du résultat. Si le diviseur est inférieur au reste, il est soustrait du reste en utilisant le même résultat du soustracteur, le résultat est bouclé vers le reste et mis-à-jour lors du prochain Front Montant. Sinon, il y a aucune implication et l'algorithme passe à l'étape 3. La troisième étape consiste à insérer en série de droite à gauche le bit de signe, qui a indiqué si le diviseur était inférieur au reste dans le Shifteur du quotient (le 1 est inséré s'il y a eu soustraction, sinon c'est le 0), lors du Front Descendant avant de terminer la période d'Horloge. Le Front Descendant est implémenté par une porte NOT. En tout, 4 périodes sont nécessaires pour faire la division sur 4 bits. La première et la deuxième étape sont faites lors du Front Montant du début de la période, la troisième étape est faite lors du Front Descendant au milieu de la période, et la mis-à-jour du reste est faite lors du Front Montant de fin de période.