RAPPORT DE TP SYSTÈMES NUMÉRIQUES

Encadré par : Pr. Jamal ZBITOU

Préparé par:

Nom :BOUSSETA Hatim
Apogée :22049272

Nom :DIALLO Abdoul-Moumouni
Apogée :22053505

25 décembre 2024

Table des matières

1 TP1 : Portes logiques élémentaires

Les portes logiques sont les éléments de base des circuits numériques. Elles permettent de réaliser des opérations logiques fondamentales.

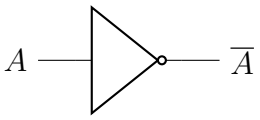
Parmi elles, on peut citer :

— **NOT** : renvoie le complémennttaire de l'entrée.
— **AND** : renvoie 1 en sortie si et seulement si toutes les entrées sont à 1.
— **OR** : renvoie 1 si au moins une des entrées est à 1.
— **XOR** : renvoie 1 si exactement une seule des entrées est à 1.
— **NAND** : inverse la sortie de la porte AND.

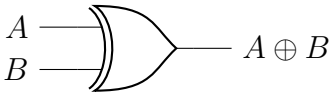
Ces portes logiques, combinées entre elles, constituent la base de fonctionnement des systèmes numériques modernes tels que les processeurs, les mémoires et les circuits intégrés comme nous le verrons dans la suite de ce rapport.

1.1 Schémas

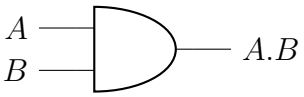
Représentation conventionnelle de quelques portes logiques élémentaires :



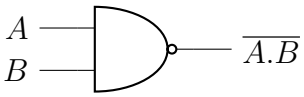
NOT



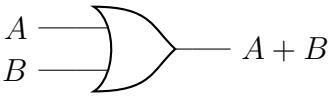
XOR



AND



NAND



OR

1.2 Table de vérité combinée

Table combinée des portes logiques

A	B	$\overline{A}$ (NOT)	$A + B$ (OR)	$A.B$ (AND)	A⊕B (XOR)	$\overline{A.B}$ (NAND)
0	0	1	0	0	0	1
0	1	1	1	0	1	1
1	0	0	1	0	1	1
1	1	0	1	1	0	0

1.3 Simulation

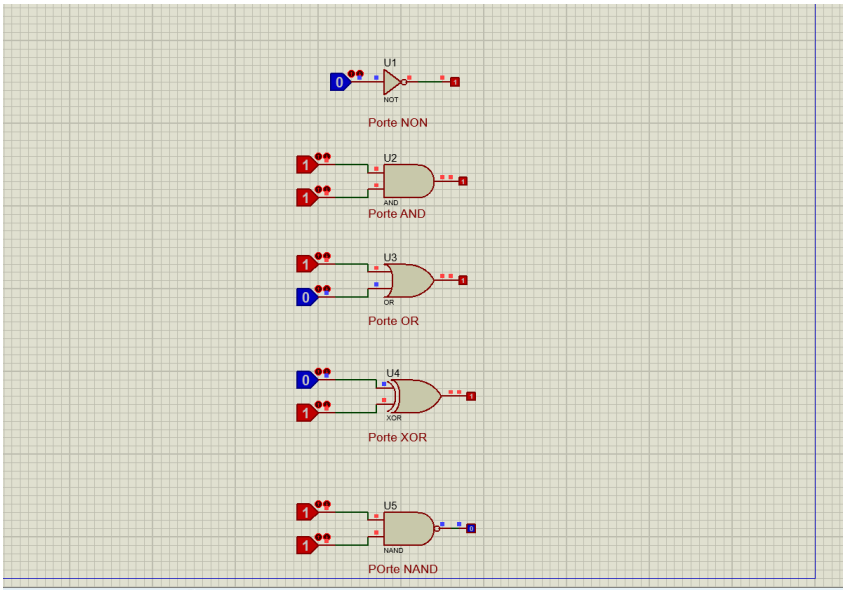


FIGURE 1 – Simulation des portes avec Proteus

CONCLUSION

`La simulation confirme bien la table de vérité établie plus haut.`

2 TP2 : Circuits logiques combinatoires et bascules

2.1 Additionneur

2.1.1 Demi-additionneur

Un demi-additionneur effectue la somme entre deux bits A et B, et produit :
— S : Somme, calculée comme $S = A \oplus B$ (XOR).
— C : Retenue, calculé comme $C = A.B$.

2.1.1.1 Schémas

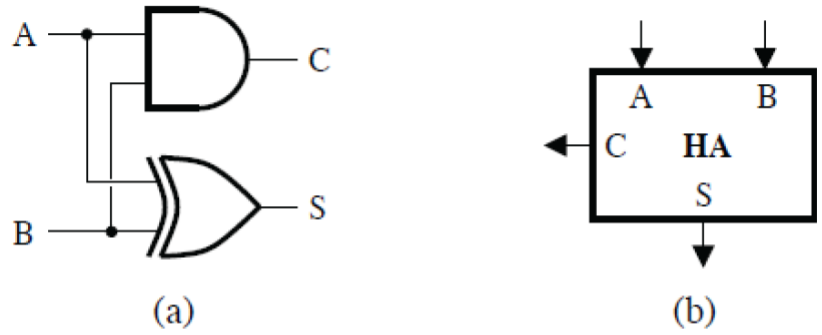


FIGURE 2 – Demi-additionneur

2.1.1.2 Table de vérité

A	B	$S = A \oplus B$	$C = A.B$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

2.1.1.3 Simulation

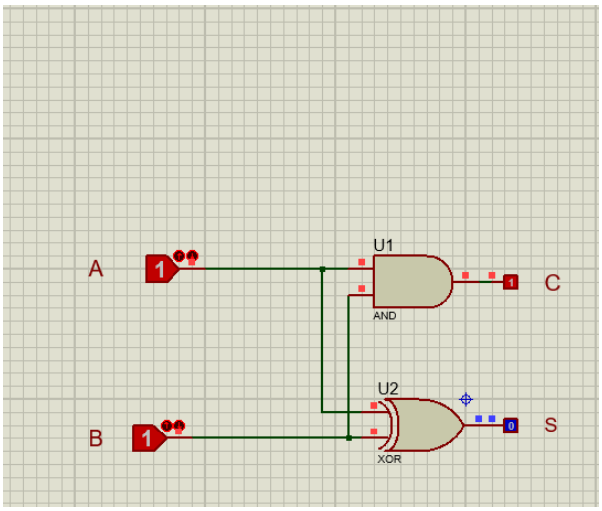


FIGURE 3 – Simulation du demi-additionneur

CONCLUSION

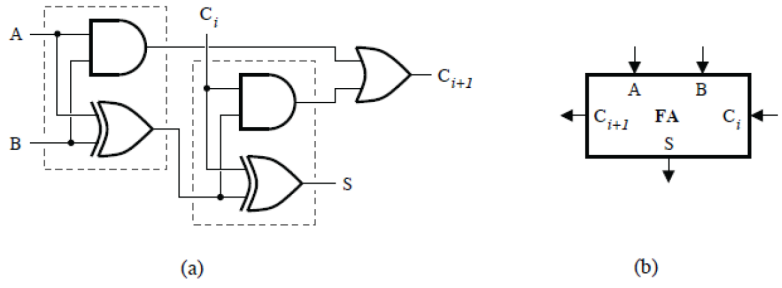
Pour deux bits 1, on obtient, comme l'indique la table de vérité, 0 comme somme et 1 en retenue (carry).

2.1.2 Additionneur complet

L'additionneur complet est un circuit combinatoire qui permet d'additionner deux bits binaires A et B, en tenant compte d'une retenue en entrée C_{in} (carry-in). Il produit deux sorties :

— La **somme** S, résultant de l'addition des trois bits (A, B, C_{in}).
— La **retenue en sortie** C_{out}, correspondant à la propagation de la retenue.

2.1.2.1 Schémas



Full adder: a) logic circuit; b) symbol

FIGURE 4 – Additionneur complet

2.1.2.2 Table de vérité

A	B	C_i	S	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

2.1.2.3 Simulation

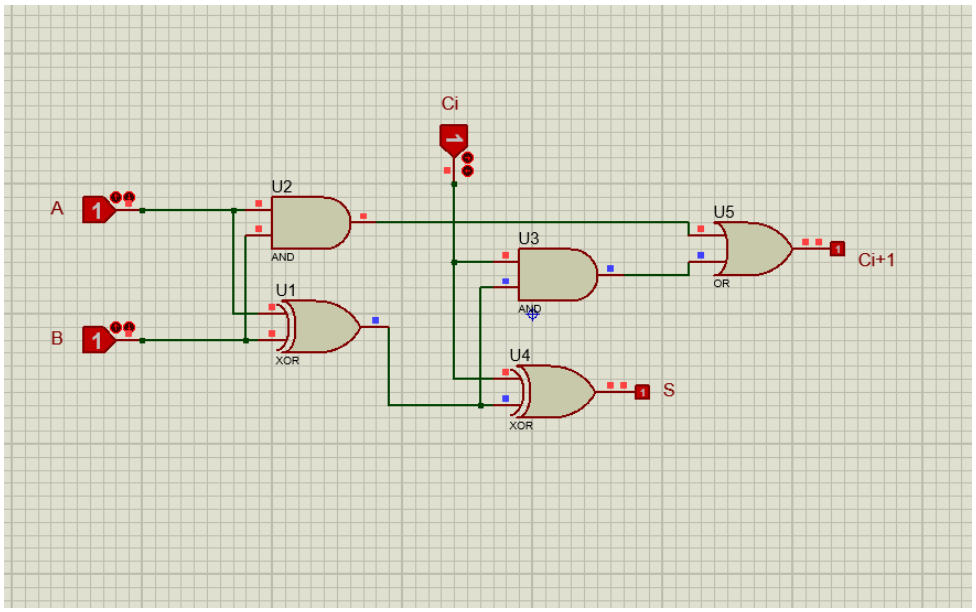


FIGURE 5 – Simulation de l'additionneur complet

CONCLUSION

La simulation confirme bien la table de vérité. Avec deux bits 1 à sommer et une retenue (C_{in}) 1 en entrée, on obtient une somme égale à 1 et une retenue C_{out} égale à 1.

2.2 Soustracteur

2.2.1 Demi-soustracteur

Un demi-soustracteur effectue la soustraction entre deux bits A et B, et produit :
— D : Différence, calculée comme $D = A \oplus B$ (XOR).
— b : Emprunt, calculé comme $b = \overline{A}.B$.

2.2.1.1 Schémas

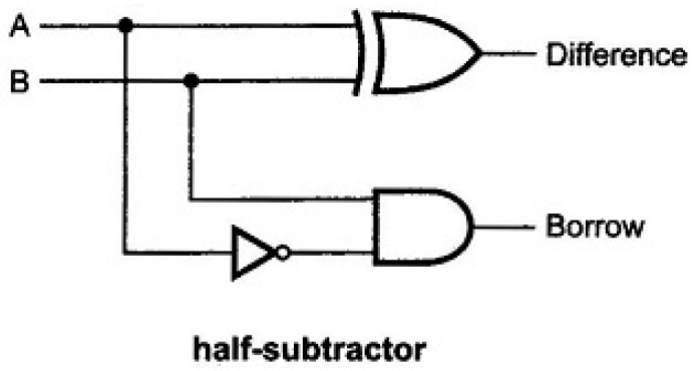


FIGURE 6 – Demi-soustracteur

2.2.1.2 Table de vérité

A	B	$D = A \oplus B$	$b = \overline{A} \wedge B$
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

2.2.1.3 Simulation

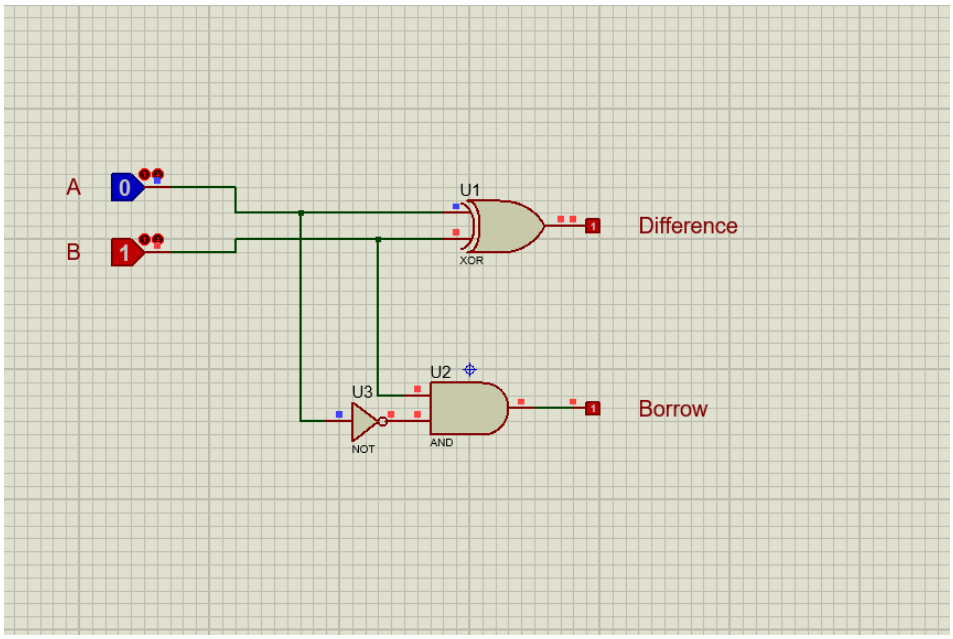


FIGURE 7 – Simulation du demi-soustrateur

CONCLUSION

On a bien le résultat comme indiqué dans la table de vérité.

2.2.2 Soustracteur complet

Un soustracteur complet effectue la soustraction entre deux bits A et B, avec un bit d'emprunt entrant b_{in}, et produit :

— D : Différence, calculée comme $D = A \oplus B \oplus b_{\text{in}}$ (XOR triple).

— b_{out} : Emprunt sortant, calculé comme $b_{\text{out}} = \overline{A}.B + b_{\text{in}}.B + \overline{A}.b_{\text{in}}$.

2.2.2.1 Schémas

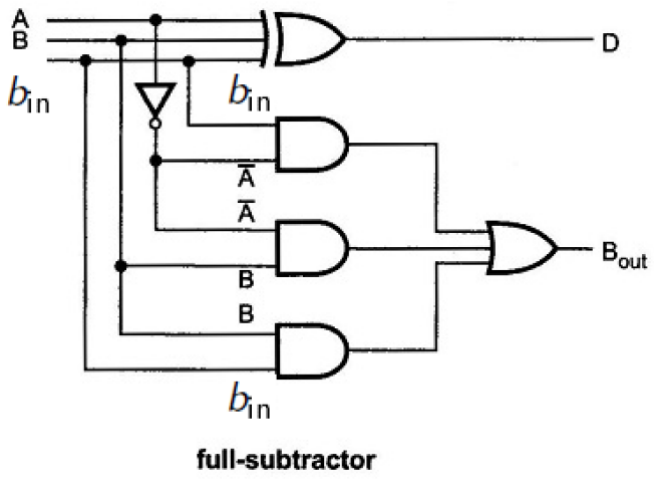


FIGURE 8 – Soustracteur complet

2.2.2.2 Table de vérité

A	B	b_{in}	D	b_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

2.2.2.3 Simulation

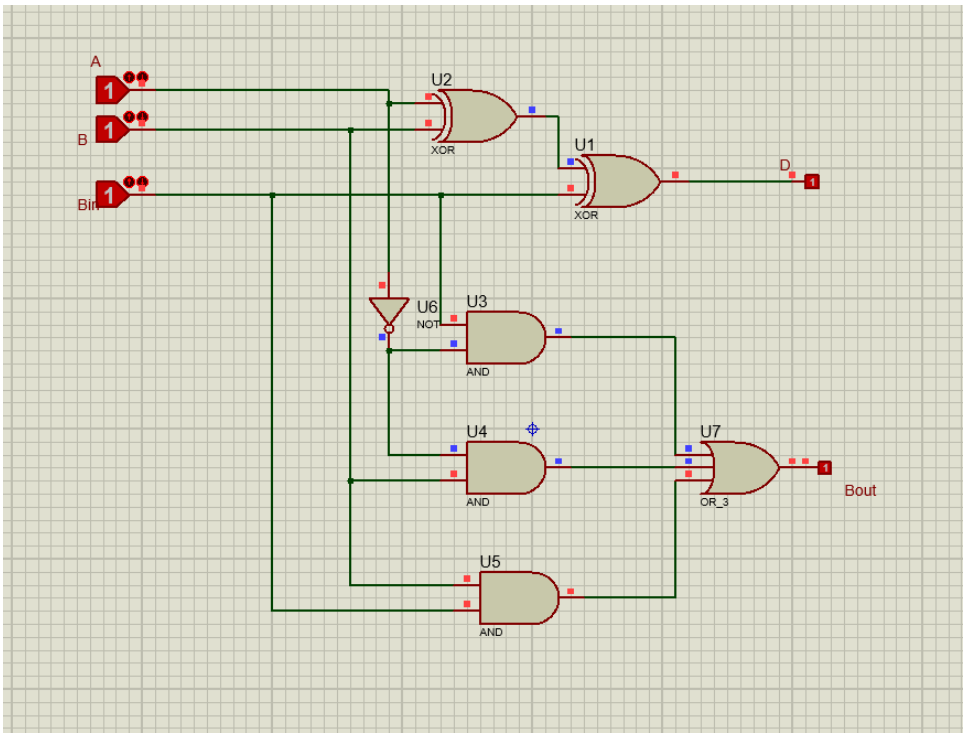


FIGURE 9 – Simulation du soustracteur complet

CONCLUSION

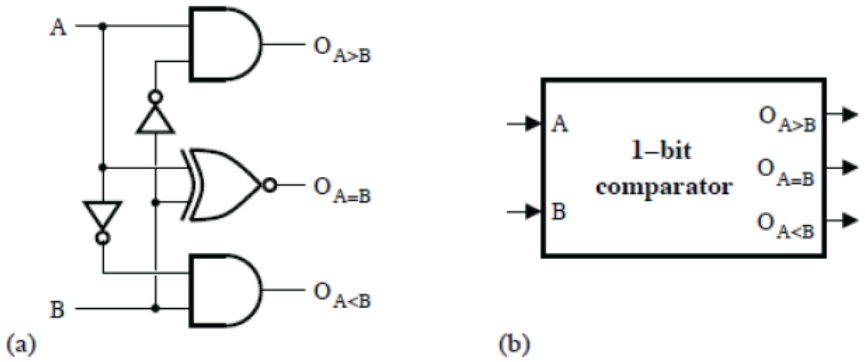
Pour la simulation, par manque d'une porte XOR à quatre entrées, on en utilise deux à deux entrées. On obtient bien le résultat comme indiqué dans la table de vérité.

2.3 Comparateur

Un comparateur compare deux bits, A et B. Il prend deux bits en entrées et produit trois sorties qui sont activées en fonction de la relation entre les deux bits :

- $O_{A>B}$: Activée lorsque $A > B$.
- $O_{A=B}$: Activée lorsque $A = B$.
- $O_{A<B}$: Activée lorsque $A < B$.

2.3.1 Schémas



One-bit comparator: a) logic circuit; b) symbol

FIGURE 10 – Comparateur

2.3.2 Table de vérité

A	B	$O_{A>B}$	$O_{A=B}$	$O_{A<B}$
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

2.3.3 Simulation

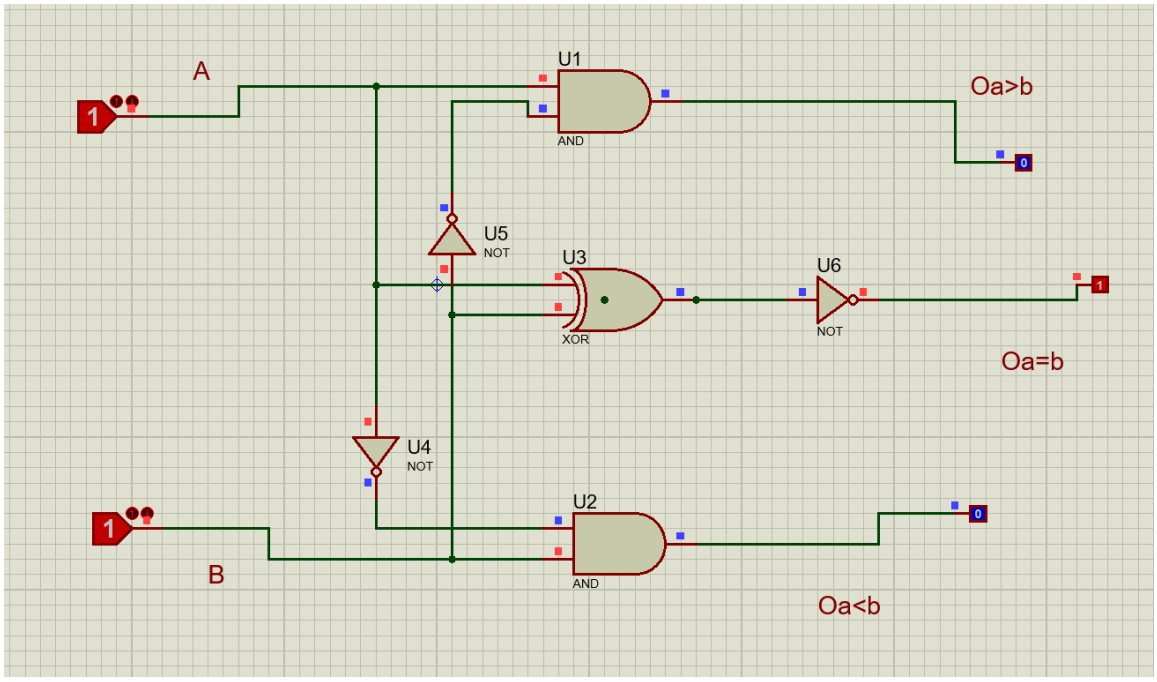


FIGURE 11 – Simulation du comparateur

CONCLUSION

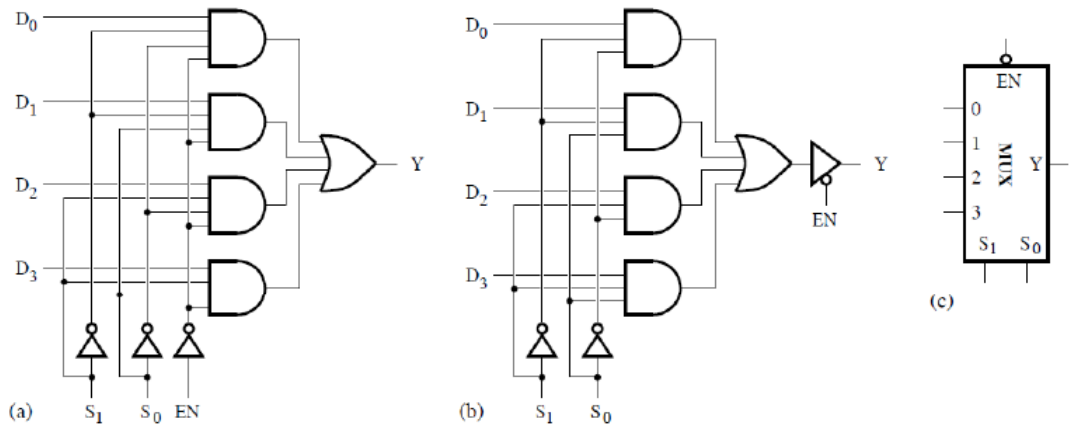
Pour deux bits A et B égaux à 1, on obtient bien la sortie $A = B$ activé.

2.4 Multiplexeur 4 :1

Un MUX 4 :1 est un circuit combinatoire qui sélectionne une des quatre entrées D_0, D_1, D_2, D_3 et la connecte à la sortie Y, en fonction des valeurs des deux bits de sélection S_1 et S_0.

Le signal d'activation EN contrôle le fonctionnement du multiplexeur. Si $EN = 1$, le multiplexeur est désactivé, et la sortie Y est forcée à 0. Si $EN = 0$, la sortie Y dépend des bits de sélection S_1 et S_0

2.4.1 Schémas



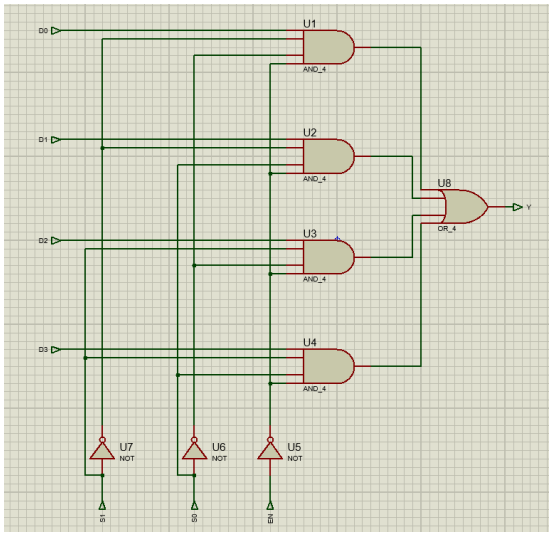
4-to-1 multiplexer with an enable input: a) output set to zero when $EN = 1$; b) output set to high impedance state when $EN = 1$; c) symbol

FIGURE 12 – MUX 4 :1

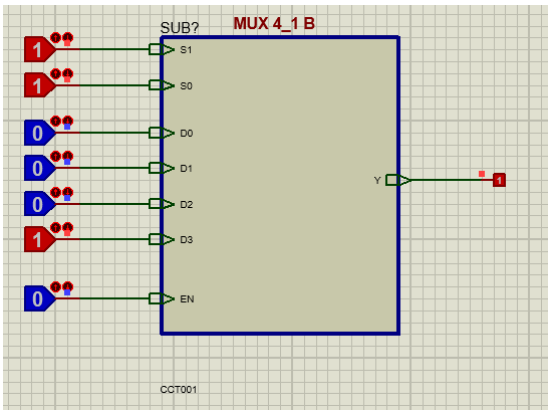
2.4.2 Table de vérité (cas du a.)

EN	S_1	S_0	Y
1	x	x	0
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3

2.4.3 Simulation



(a) Schéma



(b) Intégration dans une boite noire

FIGURE 13 – Simulation du MUX 4 :1

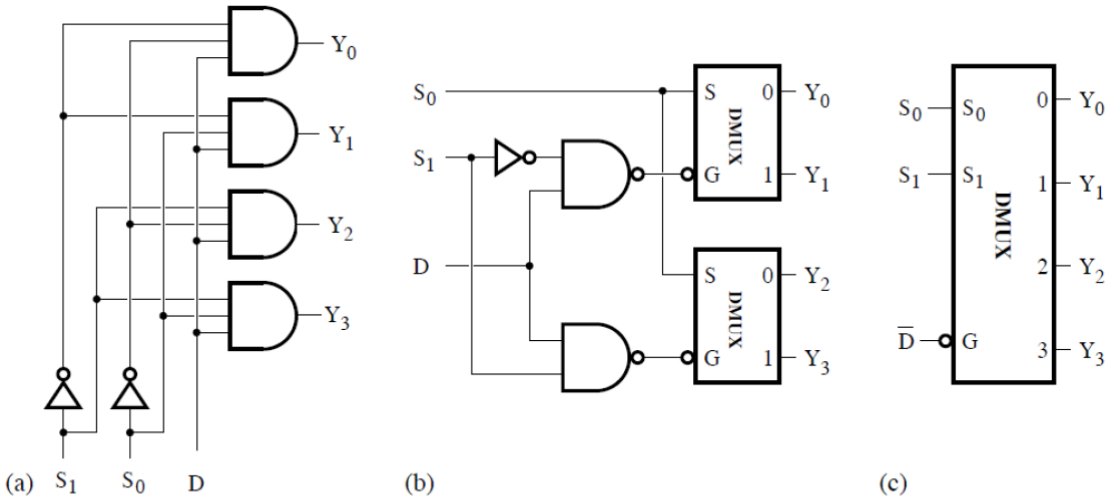
CONCLUSION

On a créé d'abord une boîte dans laquelle on a intégré le circuit du mux. Après simulation, on obtient bien le résultat. Pour S_1S_0=11, c'est l'entrée D_3 qui est sélectionnée en sortie.

2.5 Démultiplexeur 1 :4

Un DMUX 1 :4 est un circuit combinatoire qui dirige une donnée d'entrée D vers l'une des quatre sorties Y_3, Y_2, Y_1, Y_0, en fonction des valeurs des bits de sélection S_1 et S_0.

2.5.1 Schémas



$$Y_0 = \overline{S_1} \cdot \overline{S_0} \cdot D,\ Y_1 = \overline{S_1} \cdot S_0 \cdot D,\ Y_2 = S_1 \cdot \overline{S_0} \cdot D, \text{ and } Y_3 = S_1 \cdot S_0 \cdot D$$

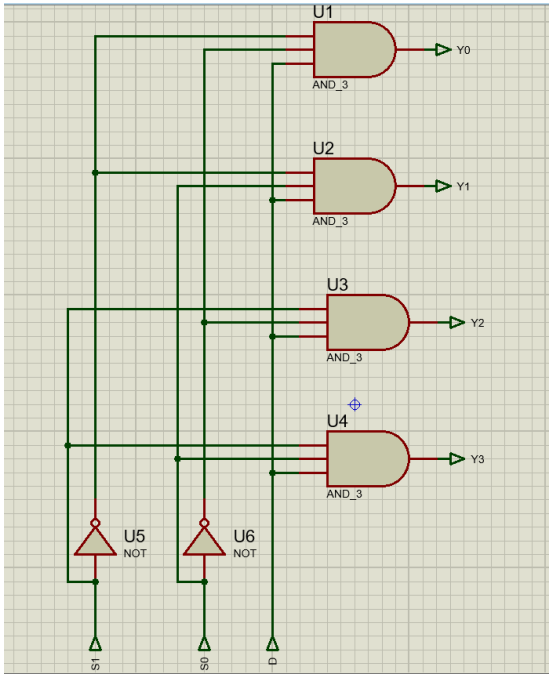
1-to-4 demultiplexers: implemented using a) logic gates or b) 1-to-2 demultiplexers; c) symbol

FIGURE 14 – DMUX 1 :4

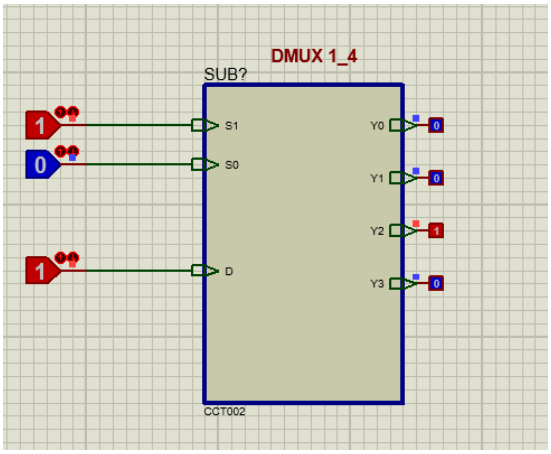
2.5.2 Table de vérité

S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	D
0	1	0	0	D	0
1	0	0	D	0	0
1	1	D	0	0	0

2.5.3 Simulation



(a) Schéma



(b) Intégration dans une boite noire

FIGURE 15 – Simulation du DMUX 1 :4

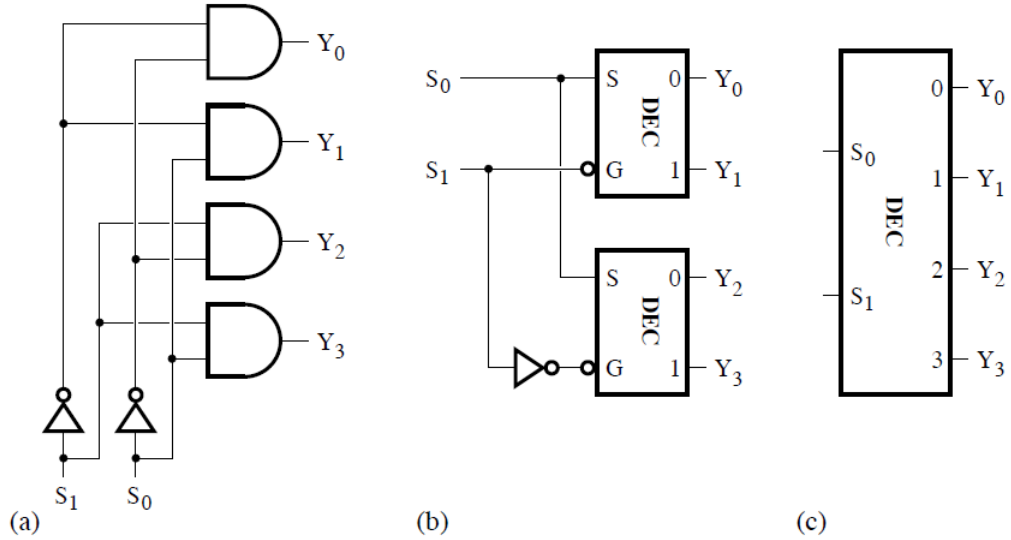
CONCLUSION

On fait la même chose pour le DMUX. Pour S_1S_0=10, c'est vers la sortie $Y2$ que D est dirigée.

2.6 Décodeur 2 :4

Un décodeur 2 :4 est un circuit combinatoire qui active l'une des quatre sorties Y_3, Y_2, Y_1, Y_0, en fonction des valeurs des bits de sélection S_1 et S_0.

2.6.1 Schémas



$$Y_0 = \overline{S_1} \cdot \overline{S_0}, \quad Y_1 = \overline{S_1} \cdot S_0, \quad Y_2 = S_1 \cdot \overline{S_0}, \quad Y_3 = S_1 \cdot S_0$$

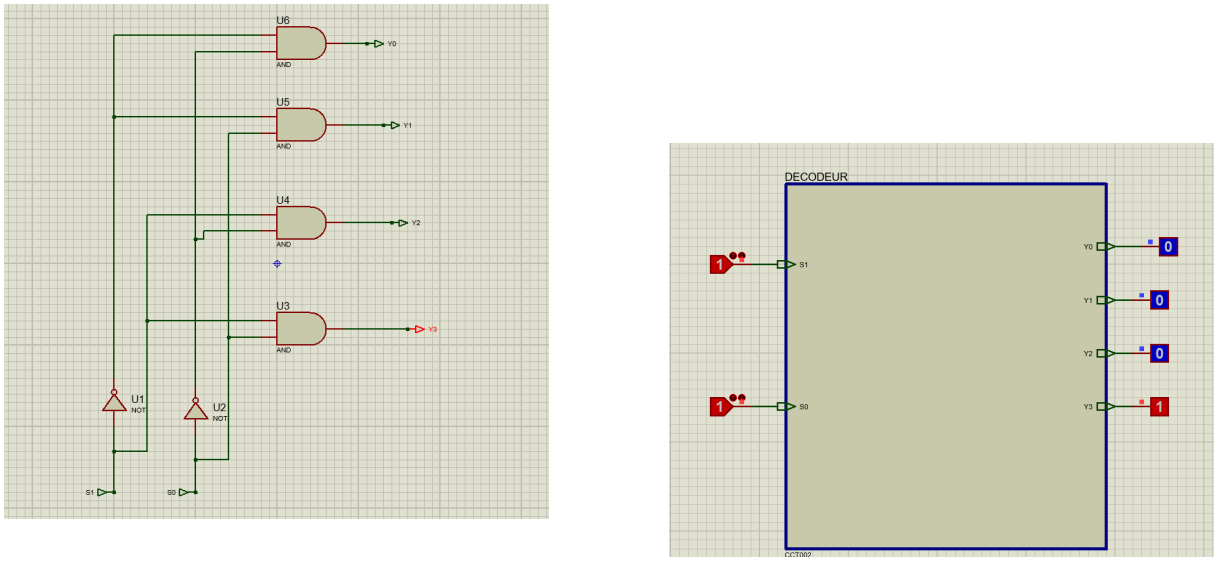
2-out-of-4 decoder: implemented using a) logic gates or b) 1-out-2 decoders; c) symbol

FIGURE 16 – DEC 2 :4

2.6.2 Table des états

S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

2.6.3 Simulation



(a) Schéma (b) Intégration dans une boîte noire

FIGURE 17 – Simulation du décodeur 2 :4

CONCLUSION

Le decodeur active la sortie Y_3 pour 11 en entrée.

2.7 Bascule RS

La bascule RS asynchrone est une bascule simple sans signal d'horloge. Elle a deux entrées : R (Reset) et S (Set)
Deux sorties :
-Q^+, qui représente l'état suivant.
-$\overline{Q}^+$, qui est le complémentaire de Q^+.
1. Si $S = 1$ et $R = 0$, la bascule passe en état **Set** ($Q^+ = 1$, $\overline{Q}^+ = 0$).
2. Si $S = 0$ et $R = 1$, la bascule passe en état **Reset** ($Q^+ = 0$, $\overline{Q}^+ = 1$).
3. Si $S = 0$ et $R = 0$, la bascule maintient son état précédent ($Q^+ = Q$).
4. Si $S = 1$ et $R = 1$, l'état est interdit.

2.7.1 Schémas

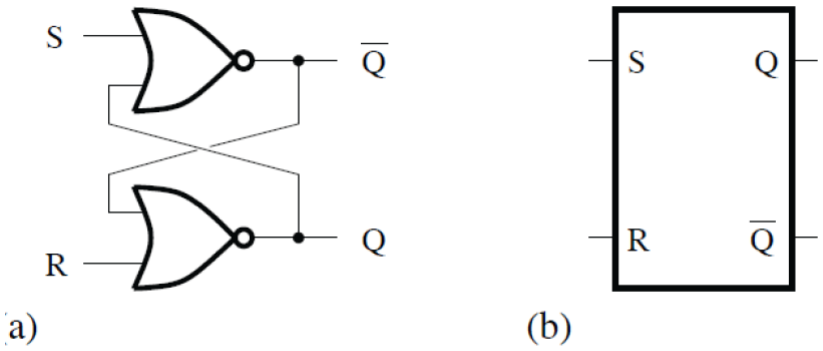


FIGURE 18 – Bascule RS

2.7.2 Table de vérité

R	S	Q^+	$\overline{Q}^+$
0	0	Q	$\overline{Q}$
0	1	1	0
1	0	0	1
1	1	0	0

2.7.3 Simulation

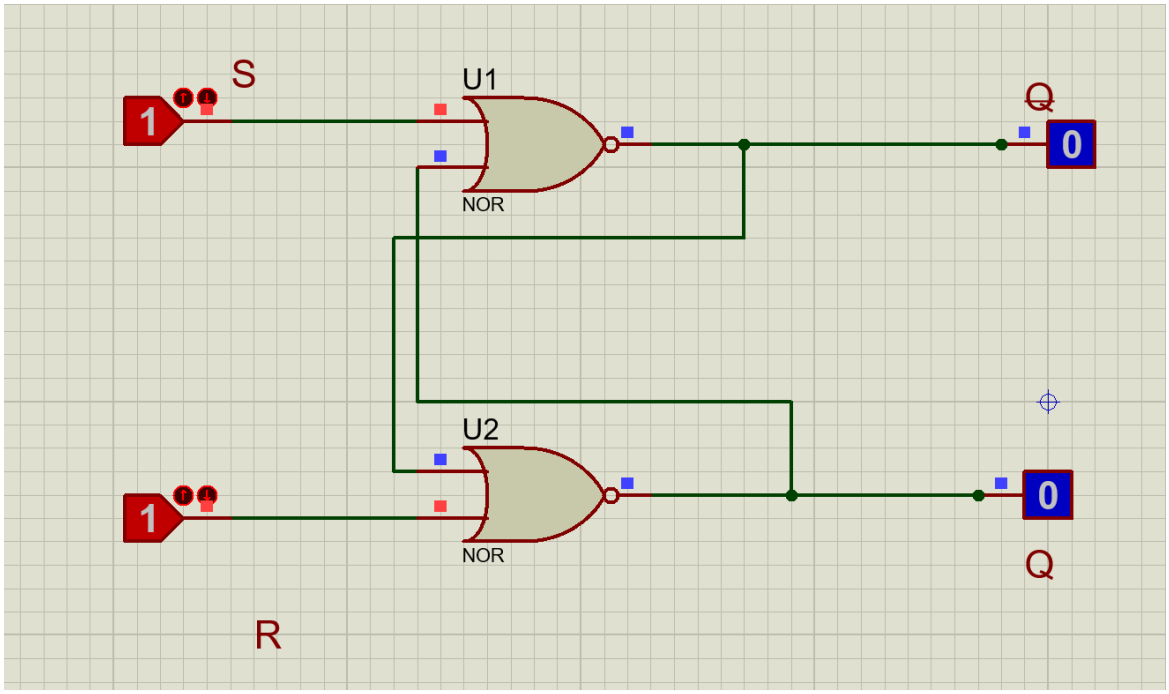


FIGURE 19 – Simulation d'une bascule asynchrone RS

CONCLUSION

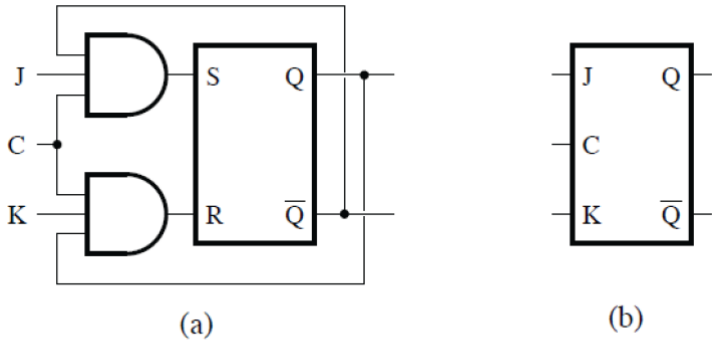
Pour le cas du forbidden state $R = 1$ et $S = 1$, on obtient 0 pour toutes les deux sorties. Ce problème peut être résolu avec l'utilisation de la bascule D en connectant un inverseur aux entrées R et S pour éviter l'occurence de l'état interdit.

2.8 Bascule JK

La bascule JK fonctionne avec des entrées J (Set) et K (Reset), synchronisées par un signal d'horloge C. Elle corrige la bascule RS en éliminant l'état interdit ($R = S = 1$).

La sortie Q^+ dépend de J, K, de l'état actuel Q, et du signal d'horloge C. Les états sont définis comme suit :
- $J = 0$, $K = 0$: No change (*Mémoire*).
- $J = 0$, $K = 1$: Reset ($Q^+ = 0$).
- $J = 1$, $K = 0$: Set ($Q^+ = 1$).
- $J = 1$, $K = 1$: Toggle (Basculement) ($Q^+ = \overline{Q}$).

2.8.1 Schémas



Basic JK flip-flop: a) logic circuit; b) symbol

FIGURE 20 – Bascule JK

2.8.2 Table de vérité

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

2.8.3 Simulation

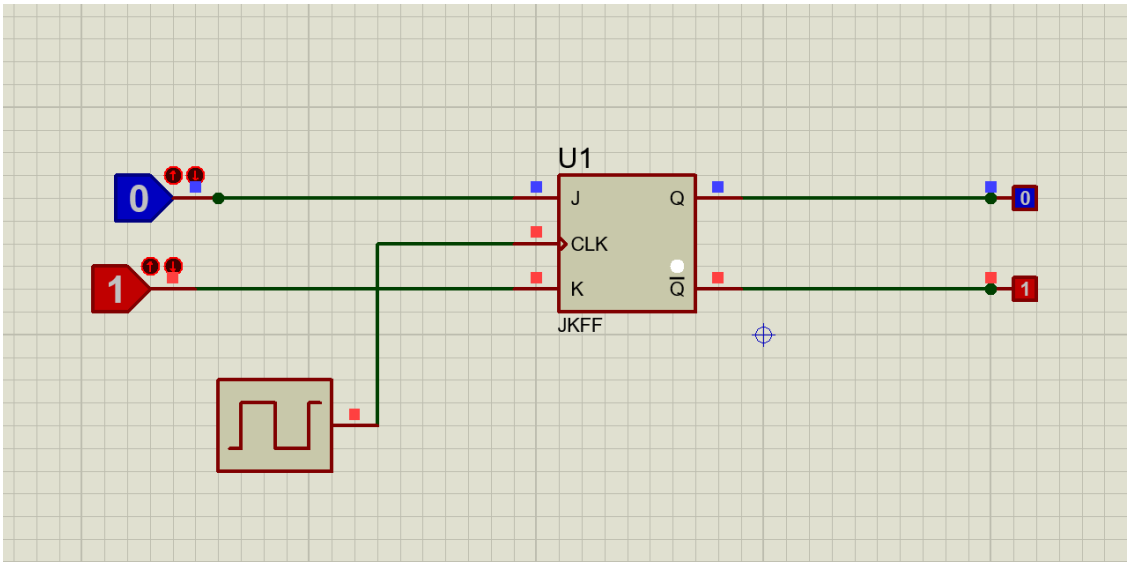


FIGURE 21 – Simulation d'une bascule JK

CONCLUSION

La simulation confirme bien l'explication du fonctionnement détaillée plus haut.

3 TP3 : Compteurs et registres

3.1 Compteur asynchrone modulo 16 à base de bascules JK

Un compteur asynchrone modulo 16 utilise une série de 4 bascules JK connectées en cascade. Chaque bascule change d'état au front négatif de l'horloge provenant de la sortie de la bascule précédente.
Ainsi :
- La première bascule (Q_0) bascule à chaque impulsion d'horloge.
- La deuxième bascule (Q_1) bascule à chaque fois que Q_0 passe de 1 à 0, et ainsi de suite.

Il compte ainsi de 0 à 15 puis revient à 0.

3.1.1 Schémas

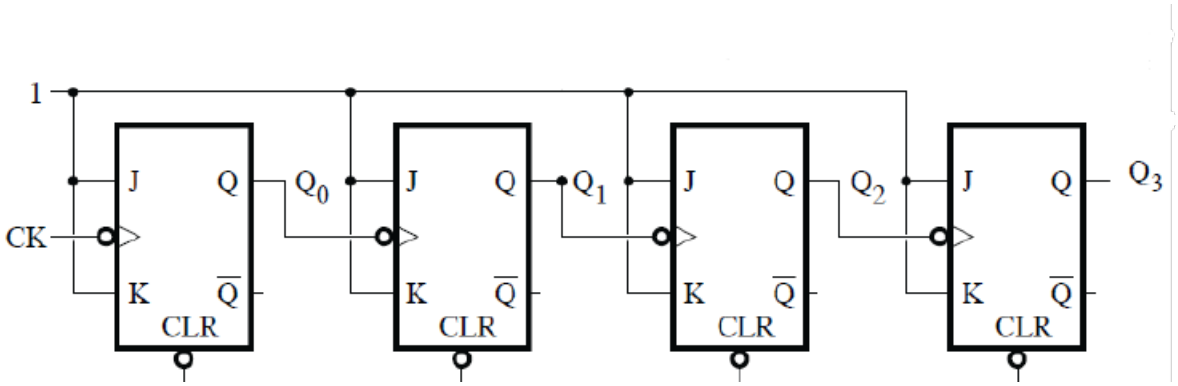


FIGURE 22 – Compteur asyscrone modulo 16

3.1.2 Table des états

Valeur	**État Présent**				**État Futur**			
	Q_3	Q_2	Q_1	Q_0	Q_3^+	Q_2^+	Q_1^+	Q_0^+
0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	1	0
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	1	0	0
4	0	1	0	0	0	1	0	1
5	0	1	0	1	0	1	1	0
6	0	1	1	0	0	1	1	1
7	0	1	1	1	1	0	0	0
8	1	0	0	0	1	0	0	1
9	1	0	0	1	1	0	1	0
10	1	0	1	0	1	0	1	1
11	1	0	1	1	1	1	0	0
12	1	1	0	0	1	1	0	1
13	1	1	0	1	1	1	1	0
14	1	1	1	0	1	1	1	1
15	1	1	1	1	0	0	0	0

3.1.3 Simulation

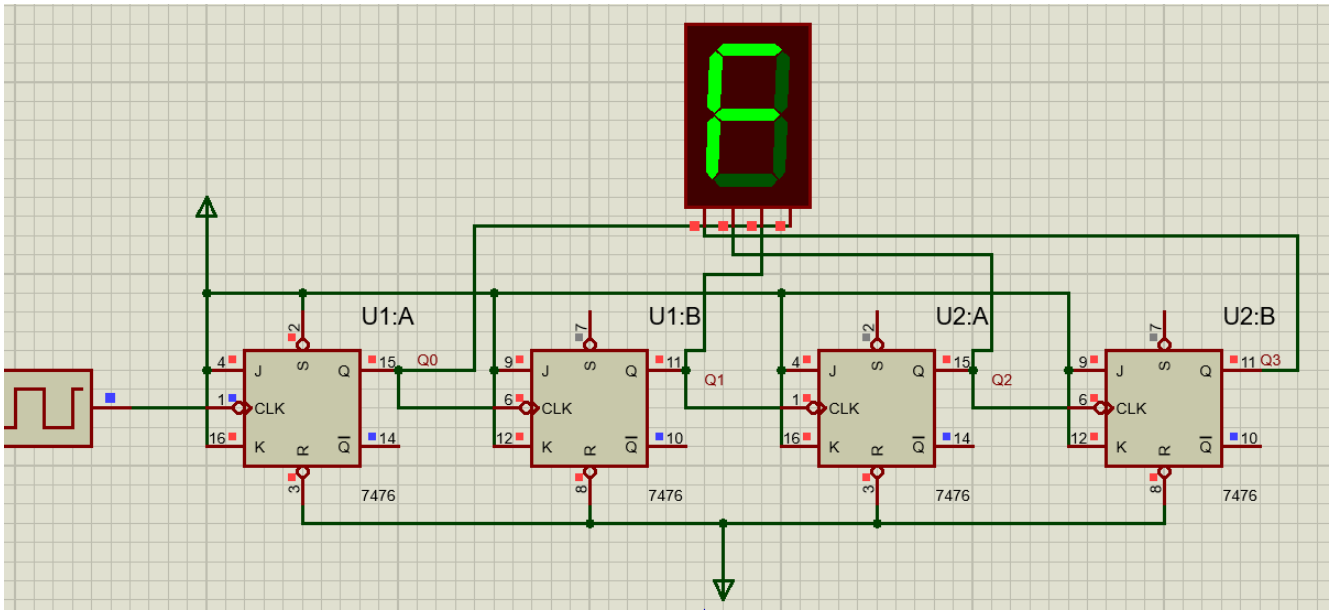


FIGURE 23 – Simulation d'un compteur asynchrone modulo 16

CONCLUSION

Ici, on utilise un afficheur 7 segemts pour afficher la valeur héxadécimal du comptage. Notre compteur compte effectivement jusqu'à F soit 15 avant de revenir à 0 comme expliqué plus haut.

3.2 Compteur synchrone modulo 4 à base de bascules D

Un compteur synchrone modulo 4 est un circuit séquentiel qui compte de 0 à 3 (en binaire) et revient à 0. Toutes les bascules de ce compteur sont activées par le même signal d'horloge.

Ce compteur utilise deux bascules D pour représenter les états avec deux bits (Q_1 et Q_0).

Les entrées :

- $D_0 = \overline{Q_0}$
- $D_1 = Q_1 \oplus Q_0$

3.2.1 Schémas

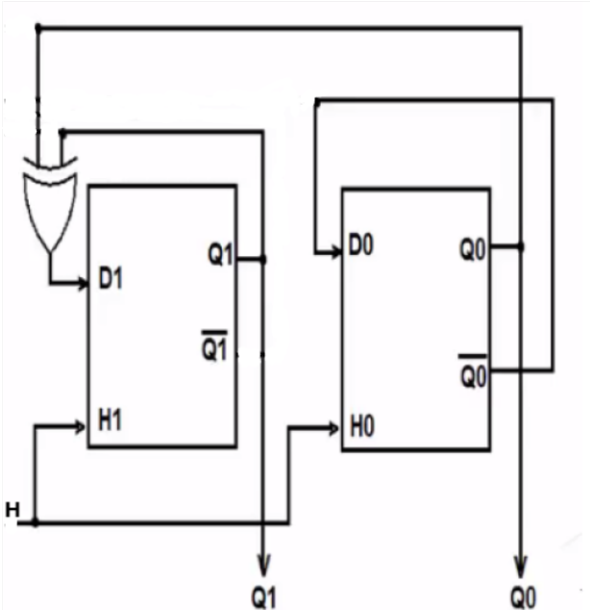


FIGURE 24 – Compteur syscrone modulo 4

3.2.2 Table des états

Valeur	**État présent** (Q_1, Q_0)	**État futur** (Q_1^+, Q_0^+)	D_1	D_0
0	00	01	0	1
1	01	10	1	0
2	10	11	1	1
3	11	00	0	0

3.2.3 Simulation

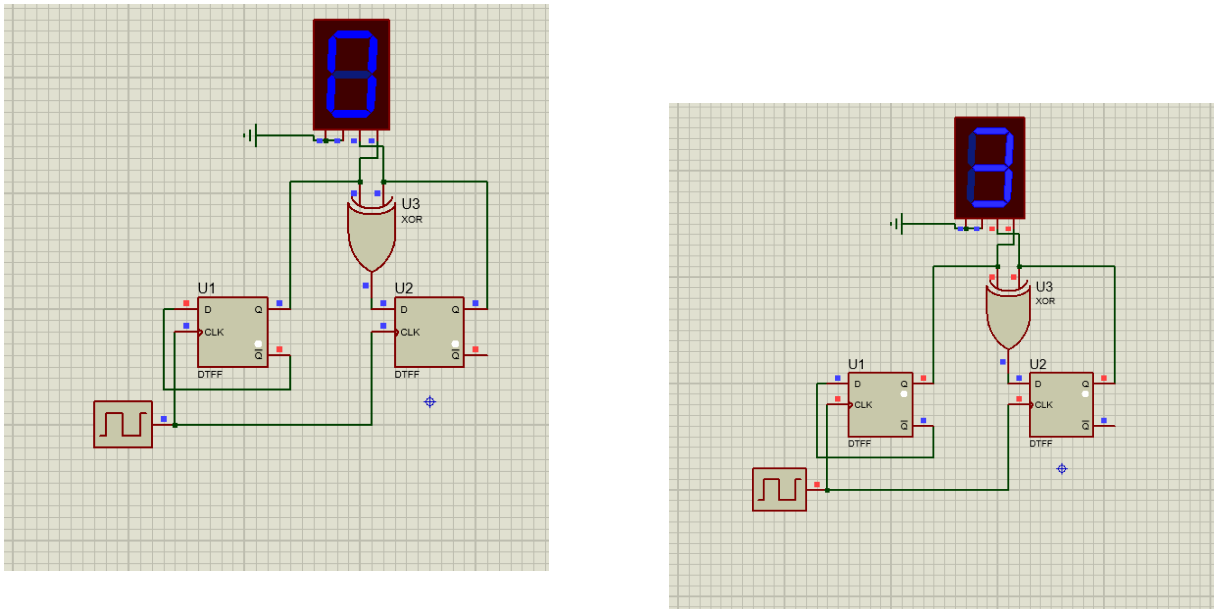


FIGURE 25 – Simulation du compteur modulo 4

CONCLUSION

Le compteur compte jusqu'à 3 puis revient à 0. Ce qui confirme notre table des états.

3.3 Registre à décalage

Un registre à décalage est un circuit séquentiel à base de bascules (de type D dans notre cas) connectées en série utilisé pour déplacer des données d'une position à l'autre sous le contrôle d'une horloge.

Ce registre est composé de quatre bascules D synchrones, avec des entrées et des sorties en série. À chaque front d'horloge, une nouvelle donnée est introduite à l'entrée série, tandis que l'ancienne est décalée vers la sortie série.

3.3.1 Schémas

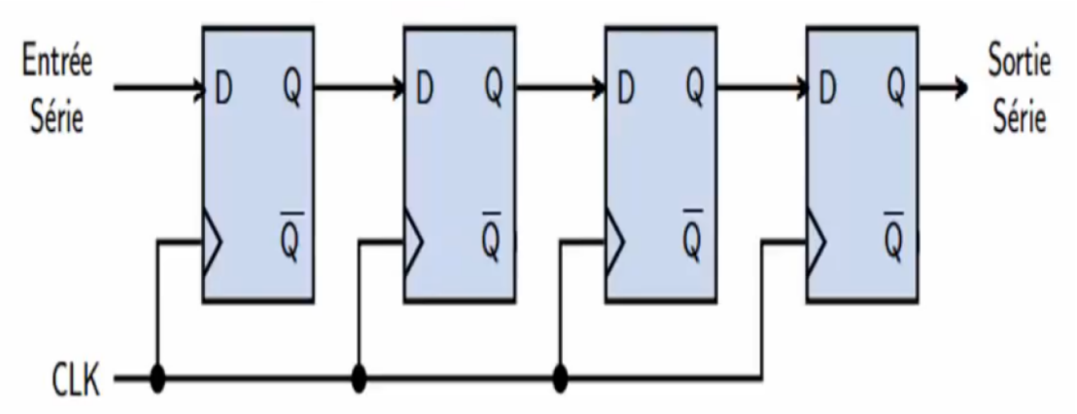
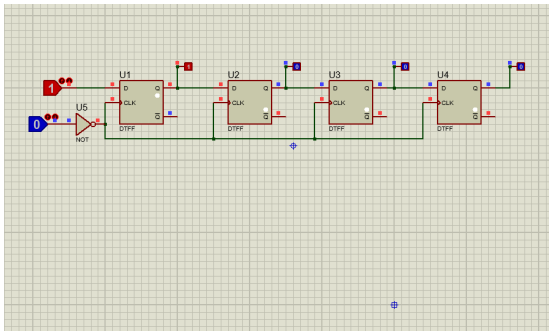


FIGURE 26 – Registre à décalage

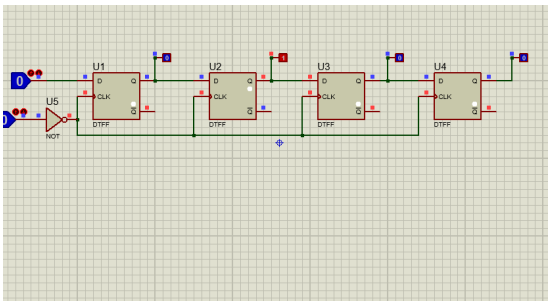
3.3.2 Table des états

Cycle d'Horloge	**Entrée Série (S_{in})**	Q_3	Q_2	Q_1	Q_0
0	1	0	0	0	0
1	0	1	0	0	0
2	1	0	1	0	0

3.3.3 Simulation



(a) front d'horloge 1 avec 1 en entrée



(b) Front d'horloge 2 avec 0 en entrée

FIGURE 27 – Simulation du registre à décalage

CONCLUSION

Dans notre cas, on introduit d'abord 1 en entrée puis on déclenche l'horloge. Le bit est décallé vers la sortie Q_3. Ensuite, on introduit 0 et la valeur de Q_3 passe dans Q_2 et celle de l'entrée dans Q_3 comme indiqué dans la table des états ci-dessus.

Table des figures

Liste des tableaux