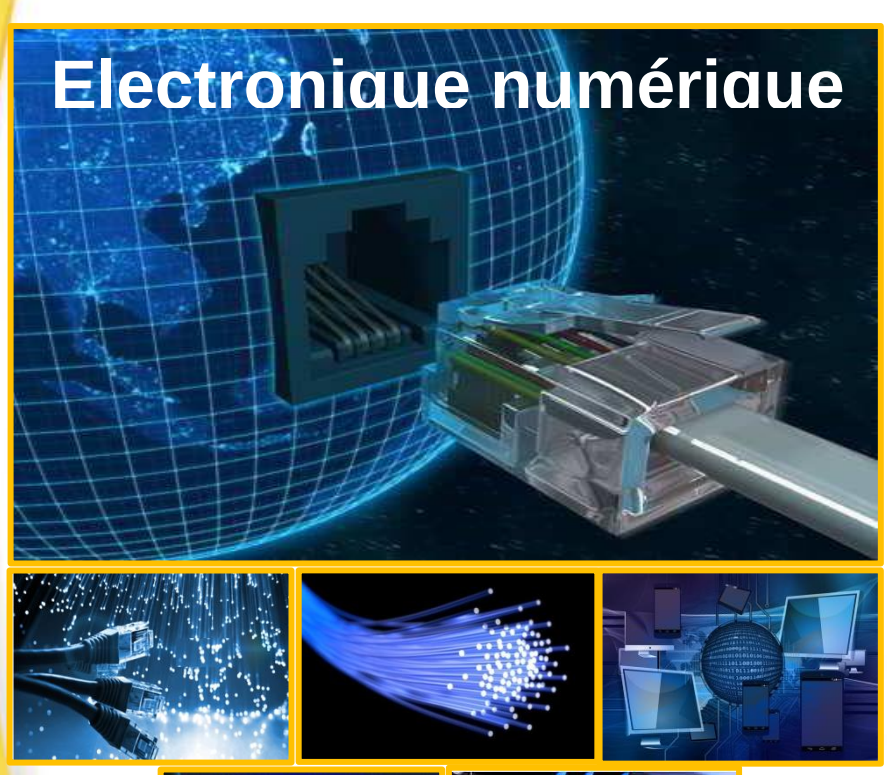
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE LA FORMATION ET DE L'ENSEIGNEMENT PROFESSIONNELS
INSTITUT DE FORMATION ET D'ENSEIGNEMENT PROFESSIONNELS
« SANHADRI ABDELHAFID - SIDI BEL-ABBES.

MANUEL TECHNIQUE ET ROFESSIONNEL DU STAGIAIRE

BTS : Réseaux Telecom Filaires

Electronique numérique



Conçu par : Mr HALAILI Mohamed

Decembre/ 2022

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE LA FORMATION ET DE L'ENSEIGNEMENT PROFESSIONNELS
INSTITUT DE FORMATION ET D'ENSEIGNEMENT PROFESSIONNELS
« SANHADRI ABDELHAFID - SIDI BEL-ABBES.

MANUEL TECHNIQUE ET ROFESSIONNEL DU STAGIAIRE

BTS : Réseaux Telecom Filaires

Conçu par : M' HALAILI Mohamed

Décembre/ 2022

Equipe de production

Conception et rédaction

M^R HALAILI Mohamed

Professeur spécialisé de la formation et de l'enseignement professionnel chargé d'ingénierie pédagogique (P.S.F.E.P.CIP) en électronique, institut de formation et d'enseignement professionnels« SANHADRI Abdelhafid - sidi bel-abbés.

Remerciements

Nous remercions toutes les personnes qui ont contribué, de près ou de loin à la conception et à la réalisation de ce manuel technico professionnel pour stagiaire (M.T.P.S) jusqu'à son édition finale,

Nous remercions également les organismes qui nous ont gracieusement autorisés à utiliser à des fins pédagogiques les illustrations et les textes tirés de leurs documentations.

TABLE DES MATIERES

CHAPITRE 2 L'ALGEBRE DE BOOLE

Préambule

Câbles sous-marin, poteaux filaires, fibre optique : le réseau télécom tisse la planète. Aujourd'hui et plus que jamais, les réseaux de télécommunications jouent un rôle essentiel dans la connexion du monde. Pour les opérateurs l'enjeu repose dans le choix d'infrastructures et de composants fiables, robustes et durables, ils vont donc mêler électronique, électronique numérique et informatique pour assurer un échange rapide des informations et ce quel que soit la distance

Ce manuel est rédigé à l'intention des stagiaires Technicien supérieure en Réseaux Telecom Filaires afin de leur apporter une aide précieuse dans l'analyse et la conception des circuits combinatoires et séquentiels (Circuits numériques).

- Le premier chapitre traite en détail des différents systèmes de numération, ainsi que les méthodes de conversion entre ces systèmes. Vous étudierez également les opérations arithmétiques sur les nombres binaires et vous terminerez ce chapitre par l'étude de plusieurs codes numériques.

- Le deuxième chapitre est consacré à l'algèbre de Boole qui va permettre au technicien de traduire les signaux électriques en expressions mathématiques en utilisant des variables logiques (Vrai /faux).Vous aborderez en détail les variables et fonctions logiques, les opérateurs logiques et leurs simplifications afin de réaliser le circuit logique le plus simplifier.

- Dans Le troisième chapitre, vous allez aborder les circuits combinatoires. Pour donner quelques exemples, on peut citer les circuits de transcodage, de multiplexage et les Circuits de calcul

- Le quatrième et dernier chapitre est consacré aux circuits séquentiels qui ont la capacité de mémoriser des informations et par conséquent de traiter des séquences de données, à titre d'exemple les bascules, les compteurs, les registres et les mémoires.

PRESENTATION DE LA PROFESSION

- **Branche professionnelle**: Electricité – Electronique – Energie
- **Famille de métiers**: Génie électrique
- **Dénomination de la profession** : BTS Réseaux Télécom Filaires
- **Définition de la profession** :

  Le titulaire du brevet de technicien supérieur des réseaux de télécommunication filaires est chargé de l'installation et de la maintenance des réseaux de communication en cuivre et fibre optique.
- **Tâches principales** : Le titulaire du brevet de technicien supérieur est chargé de :
  - Réaliser des réseaux de communication en cuivre et fibre optique qui permettent le transport aux particuliers et aux entreprises.
  - Intervenir sur l'ensemble des réseaux cuivre et fibre optique (du câblage à la maintenance).
  - Effectuer des tests et mesures sur les réseaux télécoms.

PRESENTATION DU MODULE

- **Module :** Electronique numérique
- **Code du module :** MC6
- **Durée :** 136h

Objectif modulaire

- **Comportement attendu :** A l'issu de ce module, le stagiaire doit être capable de vérifier un circuit numérique.
- **Conditions d'évaluation :** A partir de :
  - De directives
  - D'exercices et problèmes d'analyse de circuits combinatoires et séquentiels.
  - Schémas logiques et logigrammes
- **A l'aide de :**
  - Composants logiques (circuits intégrés)
  - De simulateurs logiques
  - Circuits logiques
- **CRITERES GENERAUX DE PERFORMANCE :**
  - Respect des spécifications de circuits logiques.
  - Utilisation appropriée des fiches techniques.
  - Analyse pertinente des circuits logiques.
  - Utilisation appropriée du matériel et instruments de mesure.
  - Respect des règles de santé, sécurité et d environnent.

Objectifs intermédiaires.	Critères particuliers de performance.	Eléments de contenu.
Effectuer des conversions entre des bases numériques et des codes	➢ Distinction des différents systèmes de numération ➢ Exactitude des conversions ➢ Calcul exact des opérations en binaire	➢ Systèmes de numération et codes usuels décimal, binaire, octal ; hexadécimal ; gray ; BCD ; Excess3 ; réfléchi Procéder aux conversions d'un système vers un autre. ➢ Arithmétique binaire : Représentation des nombres, addition ; soustraction ; multiplication ; division, virgules fixes et virgules flottantes.
Appliquer les notions d'algèbre booléenne	➢ Distinction des différents opérateurs logiques et de leurs tables de vérité ➢ Application correcte des postulats et théorèmes	➢ Algèbre de Boole : o Définition d'une variable binaire. o Définition d'une fonction booléenne, o Opérations logiques (inversion :porte NON), somme logique(porte OU), produit logique (porte ET) , opération NON OU(porte NI), opération NON ET(porte NAND, opération Ou exclusif (porte Ou exclusif), o Règles générales de l'algèbre de Boole (distributivité de la somme et du produit logique, formes canoniques d'une fonction Booléenne, théorème de Morgan, o Simplification des fonctions Booléennes) o Réalisation de portes logiques à l'aide d'autres portes : inverseur, OU, ET, NI , NAND
Etablir les tables de vérité d'un circuit et réduire les équations logiques par la méthode de Karnaugh	➢ Construction correcte des tables de vérité ➢ Regroupement optimal des variables ➢ Simplification optimale des fonctions ➢ Exactitude des résultats	➢ Simplification des fonctions logiques par la méthode graphique : o Tables de vérité, o Tableau de Karnaugh
Traduire les équations logiques en schémas et monter les circuits de base	➢ Conformité du schéma avec l'équation ➢ Conformité du montage avec le schéma ➢ Qualité du montage.	➢ Circuits combinatoires o Méthodes de recherche des équations logiques

Objectifs intermédiaires.	Critères particuliers de performance.	Eléments de contenu.
Distinguer les systèmes séquentiels synchrone et asynchrone	➢ Reconnaissance exacte d'un système séquentiel synchrone et asynchrone	➢ Système séquentiel o Système séquentiel synchrone o Système séquentiel asynchrone o Exemple de circuits séquentiels synchrone et asynchrone
Réaliser les différents types de bascules	➢ Etude correcte et réalisation juste de bascules ➢ Reconnaissance les circuits intégrés des bascules.	➢ Les bascules : RS, JK, D, T ➢ Les bascules en circuits intégrés
Réaliser un compteur binaire et à décade	➢ Définition exacte d'un compteur binaire et à décade ➢ Construction complète de compteurs (binaires, à décades)	➢ Compteurs binaires o Définition des concepts : Mode de départ, de marche et d'arrêt ; condition de départ dans un compteur binaire, un compteur décade (manuel, automatique, remise à zéro des compteurs)
Réaliser un registre binaire et à décades	➢ Définition exacte d'un registre binaire et à décade ➢ Construction complète de registres (binaires, à décades)	➢ Registres Définition des concepts : Mode de départ, de marche et d'arrêt ; condition de départ dans un registre à décade (manuel, automatique, remise à zéro des registres)
Définir et distinguer les types de mémoires	➢ Distinction correcte des mémoires ➢ Utilisation appropriée des mémoires	➢ Mémoires ➢ Différents types de mémoires et application : RAM. ROM. PROM. EPROM. EEPROM

CHAPITRE 1 SYSTEMES DE NUMERATION ET DE CODAGE

Quelle que soit la nature de l'information traitée par un ordinateur (image, son, texte, vidéo), elle est toujours représentée sous la forme d'un ensemble de nombres binaires**0** ou **1**(**0 Volt** ou **5Volt**). De nombreux systèmes de numération sont utilisés en technologie numérique, par exemple le système Binaire (base 2), le système Octal (base 8) et le système Hexadécimal (base 16).

De plus, ces systèmes pour pouvoir communiquer entre-autre (vers un autre système ou vers l'homme) utilisent certains codes (Code Gray, Code BCD, code Aiken…etc.) .

1.1 Systèmes de numération

De nombreux systèmes de numération sont utilisés en technologie numérique. Les plus utilisés sont les systèmes : Décimal (base 10), Binaire (base 2), Octal (base 8) et Hexadécimal (base 16).

1.1.1 Base d'un système de numération

La base d'un système de numération est le nombre de chiffres qu'utilise ce système.

- Dans une base ***X***, on utilise ***X*** symboles distincts pour représenter les nombres.
- La valeur de chaque symbole doit être strictement ***inférieure*** à la base ***X***.

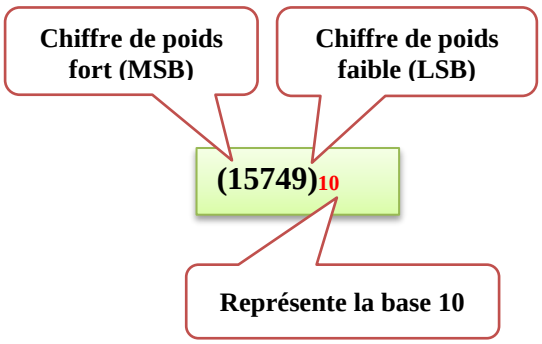
- Dans un système **décimal**, on utilise un maximum de dix symboles pour représenter un nombre quelconque N, soit : 0, 1, 2, 3, 4, 5, 6, 7, 8,9.
- Dans un système **binaire,** on utilise un maximum de **deux symboles** pour représenter un nombre quelconque N, soit : 0,1.
- Dans un système **octal,** on utilise un maximum de **huit symboles** pour représenter un nombre quelconque N, soit : 0, 1, 2, 3, 4, 5, 6,7.
- Dans un système **hexadécimal,** on utilise un maximum de **seize symboles** pour représenter un nombre quelconque N, soit :0,1,2,3,4,5,6,7,8,9,A ,B,C,D,E,F.Voir exemple suivant :

- **Exemple:** Donner les symboles que vous utiliseriez dans les systèmes de numération des bases : **5, 7**.
  - Le système dans la base =5, utilise 5 symboles (0, 1, 2, 3,4).
  - Le système dans la base =7, utilise 7 symboles (0, 1, 2, 3, 4, 5,6).

1.1.2 Rang d'un chiffre dans un nombre quelconque

Dans tous les systèmes de numération, le chiffre de poids le plus fort **(**Most Significant Bit en binaire :***MSB*)** d'un nombre est dans la colonne extrême gauche, le chiffre de poids le plus faible **(**Least Significant Bit en binaire :***LSB)*** est dans la colonne extrême droite. Le rang d'un chiffre dans un nombre est égal au numéro de sa colonne, la première colonne (***numéro 0*)** étant celle du poids le plus faible. Voir exemple suivant :

❖ **Exemple :** Soit à définir le rang des chiffres du nombre $(256987)_{10}$



Le poids le plus fort (MSB) = 2
Le poids le plus faible (LSB)=7

- Rang du chiffre 7 :0
- Rang du chiffre 8 : 1
- Rang du chiffre 9 : 2
- Rang du chiffre 6 : 3
- Rang du chiffre 5 : 4
- Rang du chiffre 2 : 5

1.1.3 Représentation polynomiale

Tout nombre ***N*** peut se décomposer en fonction des puissances entières de la base de son système de numération, voir exemples ci-dessous. Cette décomposition s'appelle la forme polynomiale du nombre ***N*** et qui est donnée par :

$$N = a_n.B^n + a_{n-1}.B^{n-1} + a_{n-2}.B^{n-2} + \ldots + a_2.B^2 + a_1.B^1 + a_0.B^0$$

- **B** : Base du système de numération, elle représente le nombre des différents chiffres qu'utilise ce système de numération.
- a_i: un chiffre (ou digit) parmi les chiffres de la base du système de numération.
- **i**: rang du chiffre a_i

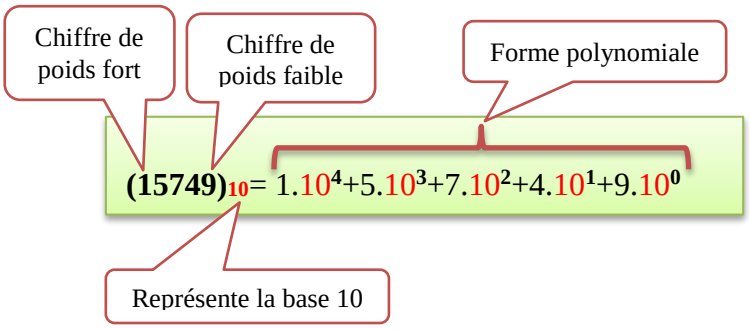
❖ **Exemples :**

- Dans le système décimal : $a_i \in \{0,1,2,3,4,5,6,7,8,9\}$
  Soit : $(54219)_{10} = 5.10^4+4.10^3+2.10^2+1.10^1+9.10^0$
- Dans le système à base 4 : $a_i \in \{0,1,2,3\}$
  Soit : $(30212)_4 = 3.4^4+0.4^3+2.4^2+1.4^1+2.4^0$
- Dans le système binaire (à base 2) : $a_i \in \{0,1\}$
  Soit : $(1011)_2 = 1.2^3+0.2^2+1.2^1+1.2^0$

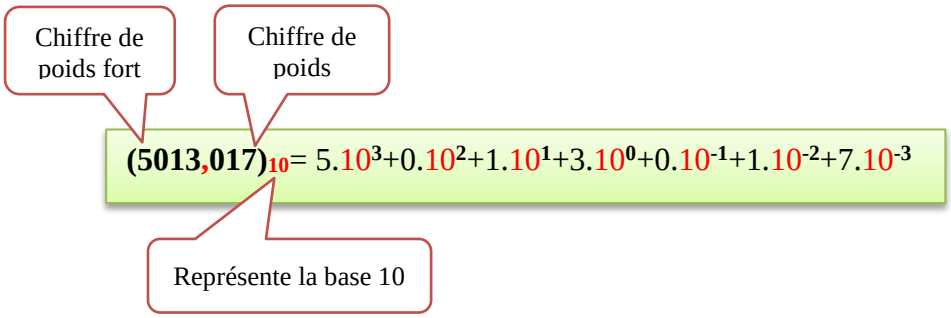
1.1.4 Système décimal (base 10)

Le système décimal comprend 10 chiffres qui sont {0, 1, 2, 3, 4, 5, 6, 7, 8,9}, comme le montre les exemples suivants.

❖ **Exemple 1 :** Forme polynomiale du nombre décimal entier $(15749)_{10}$



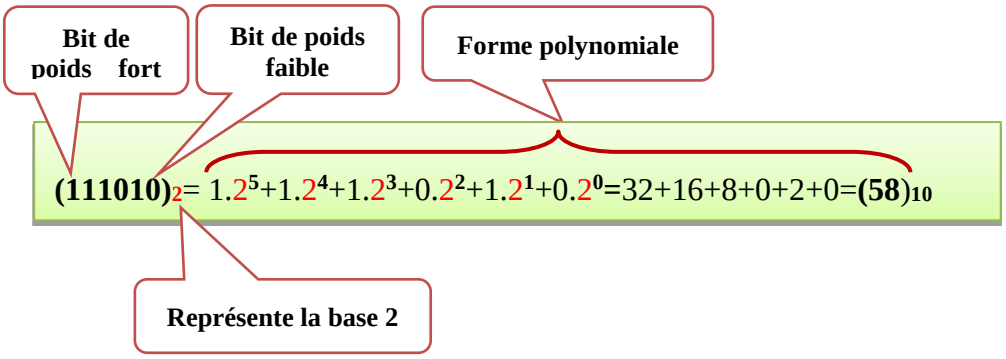
❖ **Exemple 2 :** Forme polynomiale du nombre décimal fractionnaire$(5013,017)_{10}$



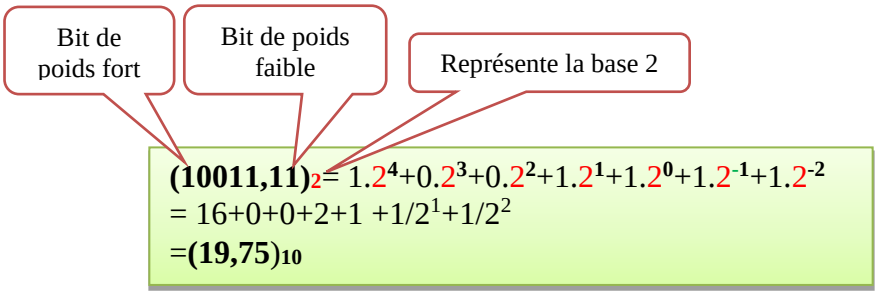
1.1.5 Système binaire (base 2)

Dans ce système de numération il n'y a que deux chiffres possibles {0, 1} qui sont souvent appelés bits « binary digit ». Comme le montre les exemples suivants.

❖ **Exemple 1 :** Forme polynomiale du nombre binaire entier $(111010)_2$



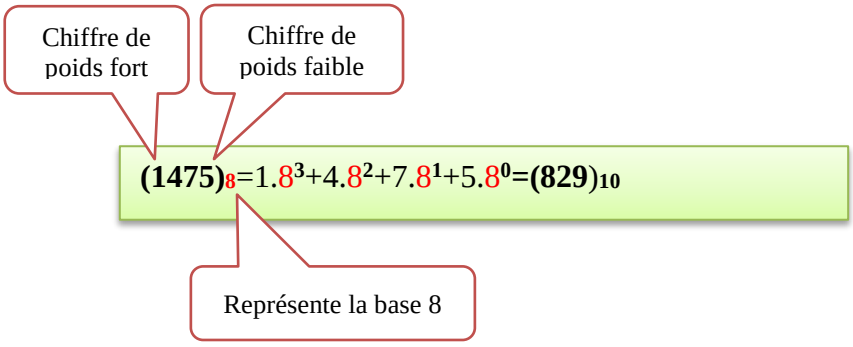
❖ **Exemple 2 :** Forme polynomiale du nombre binaire fractionnaire $(10011,11)_2$



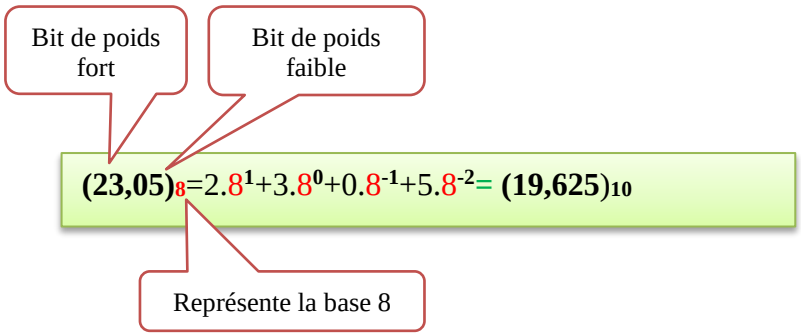
1.1.6 Système octal (base 8)

Ce système octal ou à base 8, comprend 8 chiffres qui sont {0,1,2,3,4,5,6,7}. Les chiffres 8 et 9n'existent pas dans cette base, voir (exemple 1, exemple 2).

❖ **Exemple 1 :** Forme polynomiale du nombre octal entier $(1475)_8$



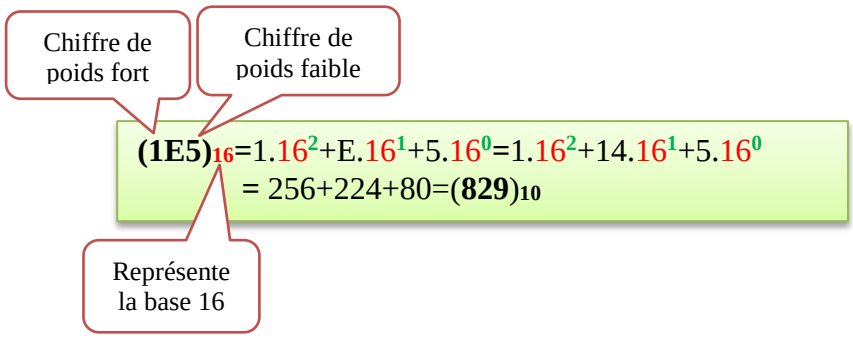
❖ **Exemple 2 :** Forme polynomiale du nombre octale fractionnaire $(23,05)_8$



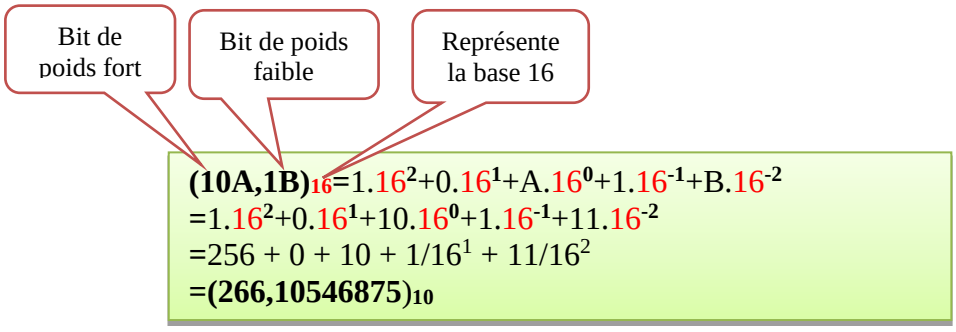
1.1.7 Système hexadécimal (base 16)

Le système hexadécimal ou base 16 contient seize éléments qui sont {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}.

❖ **Exemple 1 :** Forme polynomiale du nombre hexadécimal entier $(1E5)_{16}$



❖ **Exemple 2 :** Forme polynomiale du nombre hexadécimal fractionnaire $(10A,1B)_{16}$



1.1.8 Récapitulatif des principaux systèmes de numération

Tableau 01 Les trois principaux systèmes de numération

Décimal	Binaire naturel	Octal	Hexadécimale
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11

1.1.9 Exercice 01

Quelles sont parmi ces suites de nombre celles qui peuvent être la représentation d'un nombre en base 16, 8 ou 2 ?

Tableau 02 Représentation des nombres dans différents systèmes de

	Binaire (Base 2)	Octal (Base 8)	Hexadécimal (Base 16)
1001010210	Faux	Vrai	Vrai
1001011			
1200111201			
1431901			
2A21F			
9GF28			
1789012			

1.2 Changement de bases (conversions)

Les conversions de nombre interviennent pour passer d'un système de numération vers un autre, il s'agit du processus de conversion d'un nombre écrit dans une base b1 à une autre base b2.

1.2.1 Conversion d'un nombre N de base X en décimal

Pour convertir un nombre N de base X en décimal, il suffit de faire le développement en polynôme (***Forme polynomiale***) de ce nombre dans la base X, et de faire la somme par la suite, voir (exemple) suivant.

❖ **Exemples :** Soit à déterminer la valeur décimale des nombres de l'exemple précédent :

- $\mathbf{(1011)_2}= 1x2^0 +1x2^1 +0x2^2+1x2^3= \mathbf{(11)_{10}}$
- $\mathbf{(30212)_4}= 2x4^0 +1x4^1 +2x4^2+0x4^3+3x4^4 = \mathbf{(806)_{10}}$

1.2.2 Conversion d'un nombre entier décimal en un nombre de base X

❖ **Règle** : L'opération consiste à procéder à des divisions successives du nombre à convertir par la ***base*** du nouveau système tout en conservant les restes de ces divisions. On écrit ensuite tous les restes à partir de la fin de gauche à droite, en les convertissant en lettres s'il y a lieu**,** voir exemple 1 et 2

❖ **Exemple1 :** Soit à convertir le nombre N= $(231)_{10}$ en binaire.

230	:2= 115	reste **0**
115	:2= 57	reste **1**
57	:2= 28	reste **1**
28	:2= 14	reste **0**
14	:2= 7	reste **0**
7	:2= 3	reste **1**
3	:2=1	reste **1**
1	:2=0	reste **1**

↑ **N= (11100110)$_2$**

❖ **Exemple 2 :** Soit à convertir le nombre N= $(189520)_{10}$ en hexadécimal.

189520	:16=11845	reste **0**
11845	:16=740	reste **5**
740	:16=46	reste **4**
46	:16= 2	reste **14=E**
2	:16= 0	reste **2**

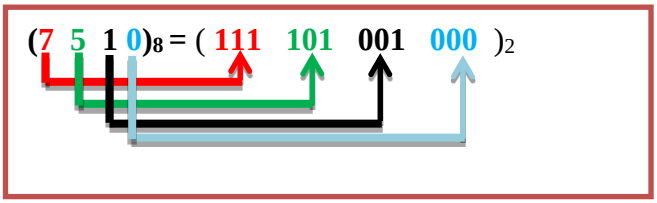
↑ **N= (2E450)$_{16}$**

1.2.3 Conversion octal-binaire et binaire-octal

❖ **Règle :**

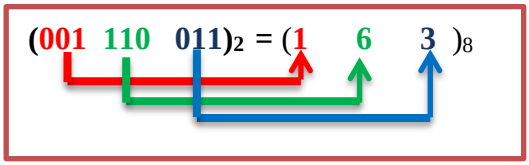
- **Conversion octal binaire :** Chaque chiffre octal est converti on binaire sur 3 bits, (voir exemple 1).
- **Conversion binaire-octal :** A partir de la virgule, grouper les bits par groupes de trois en allant vers la gauche pour la partie entière et vers la droite pour la partie fractionnaire. Convertir ensuite chaque bloc séparément en octal selon le code binaire naturel**, (**voir exemple 2).

❖ **Exemple (1) :** Soit à convertir en binaire le nombre en octal N= $(7510)_8$



N= $(7510)_8$ = $(111\ 101\ 001\ 000)_2$ = $(111101001000)_2$

❖ **Exemple(2) :** Soit à convertir en octal le nombre binaire N= $(001110011)_2$



N= $(001110011)_2$ = $(163)_8$

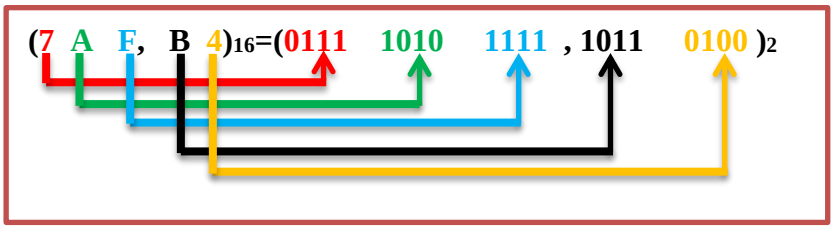
1.2.4 Conversion hexadécimal-binaire et binaire-hexadécimal

❖ **Règle :**

- **Conversion hexadécimal binaire :** Chaque chiffre hexadécimal est converti on binaire sur 4 bits, (voir exemple 1).
- **Conversion binaire-hexadécimal** : A partir de la virgule, grouper les bits par groupes de quatre en allant vers la gauche pour la partie entière et vers la droite pour la partie fractionnaire. Convertir ensuite chaque bloc séparément en hexadécimal,(voir exemple 2).

❖ **Exemple1 :** Soit à convertir en binaire le nombre en hexadécimal

$N=(7AF,B4)_{16}$



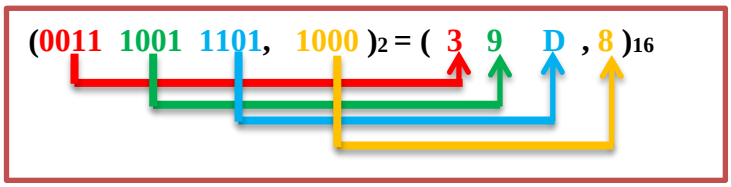
$N= (7AF, B4)_{16}= (0111\ \ 1010\ \ 0001\ \ 1111,\ \ 1011\ \ 0100\ \ 0110)_2$

$N= (0111101000011111,101101000110)_2$

❖ **Exemple 2 :** Soit à convertir en hexadécimal le nombre binaire

$N=(1110011101,10)_2$

$N=(0011\ \ 1001\ \ 1101\ ,\ 1000)_2$



$N=(1110011101,10)_2 = (0011\ \ 1001\ \ 1101, 1000)_2$

$N= (39D,8)_{16}$

1.2.5 Conversion d'un nombre décimal fractionnaire :

❖ **Règle :**

Pour convertir un nombre décimal fractionnaire (nombre à virgule) dans une ***base B*** quelconque, il faut :

- Convertir la ***partie entière*** en effectuant des divisions successives par **B** (comme nous l'avons vu précédemment). (voir exemple 1).
- Convertir la ***partie fractionnaire*** en effectuent des multiplications successives par **B** et en conservant à chaque fois le chiffre devenant entier.(voir exemple 2).

❖ **Exemple 1 :** Soit à convertir en binaire le nombre $(8,35)_{10}$

➢ **Partie entière : $(8)_{10} = (?)_2$**

8 :2= 4 reste 0
4 :2= 2 reste 0
2 :2= 1 reste 0
1 :2= 0 reste 1

$(8)_{10} = (1000)_2$

➢ **Partie fractionnaire : $(0.35)_{10} = (?)_2$**

0,35 x2 = 0, 70 **0**
0,70x2 = 1, 40 **1**
0,40x2 = 0, 80 **0**
0,80x2 = 1, 60 **1**
0,60x2 = 1, 20 **1**

$(0.35)_{10} = (0,01011)_2$

Donc $(8,35)_{10} = (1000,01011)_2$

❖ **Remarque :** En augmentant le nombre de multiplications, on améliore l'approximation.

1.2.6 Exercice 02

Complétez le tableau suivant, puis vérifier les résultats en faisant l'opération inverse

Tableau 03 : Conversion des nombres dans différents systèmes de numération

Base	Nombre	Equivalent	Dans la base
10	45		2
10	57		8
8	75		16
2	1101,101		10
10	42,0625		8
16	ABC, A		8
4	1320		16

1.3 Opérations arithmétiques en binaire

On peut évidemment effectuer les quatre opérations arithmétiques fondamentales (addition, soustraction, multiplication et division) non seulement dans le système décimal mais aussi dans les autres systèmes numériques et en particulier dans le système binaire ; les règles du système décimal seront valables pour ces opérations

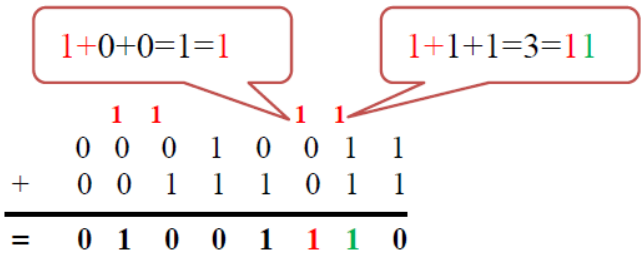
1.3.1 Addition en binaire

L'addition en binaire se fait avec les mêmes règles qu'en décimal : on commence par additionner les bits de poids faibles ; on a des retenues lorsque la somme de deux bits de même poids dépasse la valeur de l'unité la plus grande (dans le cas du binaire : 1) ; cette retenue est reportée sur le bit de poids plus fort suivant, voir exemple ci-dessous.

❖ **Règles de l'addition :**

0 + 0 = 0
1 + 1 = 0 ➡ report 1
1 + 0 = 1
1 + 1 + 1 = 1 ➡ report 1

❖ **Exemple :** Effectuer en binaire l'opération suivante $(00010011)_2 + (00111011)_2$



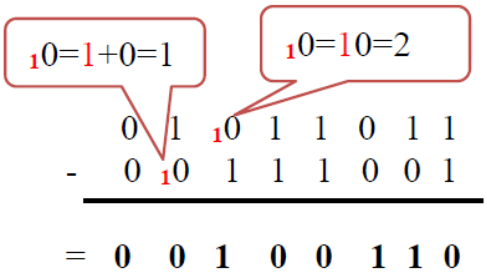
1.3.2 Soustraction en binaire

Dans la soustraction binaire, on procède comme en décimal. Quand la quantité à soustraire est supérieure à la quantité dont on soustrait, on emprunte 1au voisin de gauche. En binaire, ce 1 ajoute 2 à la quantité dont on soustrait, tandis qu'en décimal il ajoute 10, voir exemple ci-dessous.

❖ **Règles de la soustraction :**

0 - 0 = 0
1 - 1 = 0
1 - 0 = 1
0 - 1= 1 ➡ retenue 1

❖ **Exemple :** Effectuer en binaire l'opération suivante $(01011011)_2 + (00111001)_2$



1.3.3 La multiplication

La multiplication se fait en formant un produit partiel pour chaque chiffre du multiplieur (seuls les bits non nuls donneront un résultat non nul). Lorsque le bit du multiplieur est nul, le produit partiel est nul, lorsqu'il vaut un, le produit partiel est constitué du multiplicande décalé du nombre de positions égal au poids du bit du multiplieur, (voir exemple suivant).

❖ **Règles de multiplication :**

0 x 0 = 0
1 x 0 = 0
0 x 1 = 1
1 x 1 = 1

❖ **Exemple :** Effectuer en binaire l'opération suivante $(1011)_2 + (1101)_2$

```
  1011   (11)
 *1101   (13)
  1011
 1011•
1011
10001111  (143)
```

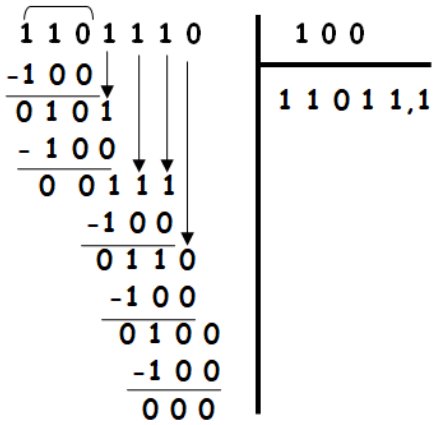
1.3.4 Division

La division binaire s'effectue à l'aide de soustractions et de décalages, comme la division décimale, sauf que les chiffres du quotient ne peuvent être que 1 ou 0. Le bit du quotient est 1 si on peut soustraire le diviseur, sinon il est 0, voir exemple ci-dessous

❖ **Règles de division :**

0 ÷ 0 = impossible
1 ÷ 0 = impossible
0 ÷ 1 = 0
1 ÷ 1 = 1

❖ Exemple : Réaliser la division suivante en binaire $(1101110)_2 \div (100)_2$



1.3.5 Exercice 03 Effectuez les opérations suivantes en binaire sur 8 bits

A. $(9)_{10} + (8)_{10}$
B. $(57)_{10} + (31)_{10}$
C. $(45)_{10} - (41)_{10}$
D. $(56)_{10} - (4)_{10}$
E. $(84)_{10} \div (5)_{10}$
F. $(25{,}25)_{10} \times (4)_{10}$

1.4 Nombres binaire signés

Nous avons travaillé jusqu'à maintenant qu'avec des nombres binaires positifs. Nous avons alors traité les groupes de bits comme des nombres réels dans un système à base 2. Toutefois, il est nécessaire de considérer ces groupes de bits comme des codes de nombres ; cela nous permettra de représenter des nombres binaires négatifs en employant seulement des 0 et des 1.

1.4.1 Complémentation

A. Complément à 1 d'un nombre binaire :

Pour obtenir le complément à 1 d'un nombre binaire, il suffit de complémenter chaque bit : devient 0 et 0 devient 1, voir exemple ci-dessous.

❖ **Exemple : Calculer le** complément à 1 du nombre $(011011101)_2$

Le Cà2$(011011101)_2$ = $(100100010)_2$.

La somme de ces deux nombres est égale à 111111111.

B. Complément à 2 d'un nombre binaire

Pour obtenir le complément à 2 d'un nombre binaire, il suffit d'ajouter 1 à son complément à 1, voir exemple ci-dessous.

Une **autre méthode** consiste à conserver tous les bits à partir de la droite jusqu'au premier 1 compris et de changer les autres bits de 0 en 1 ou de 1 en 0.

❖ **Exemple :** Le complément à 2 du nombre 011011101 est :

Cà2$(011011101)_2$=Cà1$(011011101)_2$+1
=(100100010)+1
=$(100100011)_2$

$$\begin{array}{rr} & 1\,0\,0\,1\,0\,0\,0\,1\,0 \\ + & \qquad\qquad 1 \\ \hline = & 1\,0\,0\,1\,0\,0\,0\,1\,1 \end{array}$$

1.4.2 Nombres binaires signés en complément à deux

Une contrainte très importante va intervenir dès qu'il s'agit de faire des opérations par des machines car celles-ci travaillent avec des registres de tailles fixes donc traitent des nombres qui ont toujours le même nombre de bits. Prenons l'exemple d'une machine qui travaille sur 4 bits, elle peut représenter $16=2^4$ nombres différents. Deux possibilités s'offrent à nous :

A. Les **2^4= 16** nombres seront considérés comme des entiers **non signés.** On aura donc **16nombres positifs** allant de **0 à 2^4-1** c'est-à-dire de **0 à 15,** voir (figure 04)

Tableau 04 : Les nombres entiers non signés représenté par 4 bits

Décimal	Nombres binaire Non signés			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

B. Les 16 nombres seront considérés comme des nombres signés. On aura donc **8 nombres positifs** (moitié) allant de **0 à 7** et **8 nombres négatifs** allant de **-8 à -1,** voir (figure 05)

Tableau 05 : Les nombres entiers signés représenté par 4 bits

DECIMAL	NBSCà2			
	signe	Nombre		
0	0	0	0	0
+1	0	0	0	1
+2	0	0	1	0
+3	0	0	1	1
+4	0	1	0	0
+5	0	1	0	1
+6	0	1	1	0
+7	0	1	1	1

DECIMAL	NBSCà2			
	signe	Nombre		
-8	1	0	0	0
-7	1	0	0	1
-6	1	0	1	0
-5	1	0	1	1
-4	1	1	0	0
-3	1	1	0	1
-2	1	1	1	0
-1	1	1	1	1

Remarque : Avec un nombre binaire de n bits on peut représenter

Machine n bits $\Rightarrow 2^n$ Nombres différents		
Nombres non signés	**Nombres signés**	
2^n Nombres	**$2^n/2$ Négatifs**	**$2^n/2$ Positifs**
$\mathbf{0 \longrightarrow 2^n - 1}$	$\mathbf{-1 \longrightarrow -2^n/2}$	$\mathbf{0 \longrightarrow (2^n/2) - 1}$

Pour calculer l'équivalent binaire signé complément à 2 d'un nombre décimal négatif, (Voir exemple 1 et 2).

Exemple1 : Convertir $(-7)_{10}$ en binaire signé complément à 2 sur 4bits

1. On commence par calculer l'équivalent binaire sur 4bits du nombre décimal $(+7)_{10}$
   $\Longrightarrow (+7)_{10} = (1010)_2$
2. Puis on calcule le complément à 2 du nombre $(1010)_2$
   Cà2$(1010)_2 = (0110)_2$

$\Longrightarrow$**$(-7)_{10} = (0110)_2$**

Exemple2 : Représenter les nombres décimaux suivants sous la forme binaire signée complément à2 sur (8bits). $(32)_{10}$, $(-32)_{10}$, $(27)_{10}$,$(-27)_{10}$, $(+130)_{10}$
Avec 8bits et on binaire signé Complément à2, les nombres qu'on peut représenter varies entres **[-1, $-2^8/2$] et [0 , $+(2^8/2) - 1$]**= [-1 , -128] et [0 , + 127] c'est-à-dire -128 et +127
La case de gauche est réservée au signe. La représentation retenue est de représenter le signe + par **0** et le signe – par **1** appelés bit de signes. Les nombre négatifs sont représentés par leur forme complément à 2.
$(130)_{10} \notin$ [-128, +127]

Décimal	Binaire signé Cà2(8bits)							
$(32)_{10}$	0	0	1	0	0	0	0	0
$(-32)_{10}$	1	1	1	0	0	0	0	0
$(27)_{10}$	0	0	0	1	1	0	1	1
$(-27)_{10}$	1	1	1	0	0	1	0	1
$(130)_{10}$								

1.4.3 Soustraction par complément à 2

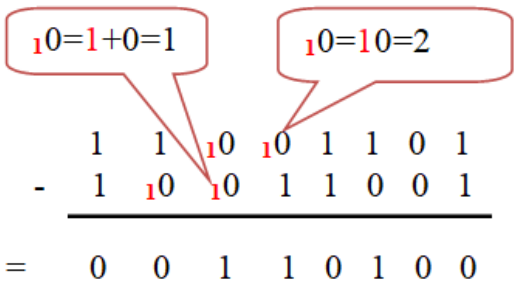
Les opérations par complément à deux s'opèrent toujours sur des nombres binaires ayant le même nombre de bits

❖ **Règle :**
Etape 1 : Additionner le 1er chiffre au complément à 2 du 2ieme chiffre
Etape 2 : Eliminer le 1 situé le plus à droite du résultat obtenu (débordement) , voir exemple ci-dessous.

❖ **Exemple :** Réaliser la soustraction suivante $(1100110001)_2 - (1001100110)_2$:
- **A.** Utiliser la méthode de la soustraction directe
- **B.** Utiliser la méthode du complément à 2

❖ **Solution :**

A. Soustraction directe : $(11001101)_2 - (10011001)_2$



Donc $(11001101)_2 - (10011001)_2 = (00110100)_2$

B. Soustraction (Complément à 2) : $(11001101)_2 - (10011001)_2$

$$(11001101)_2 - (10011001)_2 = (11001101)_2 + [-(10011001)_2]$$
$$= (11001101)_2 + Ca2\ (10011001)_2$$
$$= (11001101)_2 + (01100111)_2$$

$$\begin{array}{rcccccccc} & {}^{1}1 & 1 & 0 & {}^{1}0 & {}^{1}1 & {}^{1}1 & {}^{1}0 & 1 \\ + & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ \hline = & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{array}$$

Débordement à éliminer

Donc$(11001101)_2 - (10011001)_2 = (00110100)_2$

1.4.4 Exercice 04

1. **Complétez le tableau suivant sur 8bits**

Base10	Binaire signé Cà2
-125	
	11101011
+98	
	11111111

2. Effectuez les soustractions suivantes en binaire sur 8 bits en utilisant la méthode directe puis celle qui utilise le complément a 2

- $(42)_{10} - (29)_{10}$
- $(25)_{10} - (9)_{10}$
- $(12)_{10} - (52)_{10}$

1.5 Systèmes de codage

Les systèmes utilisés précédemment constituent des systèmes naturels ou codes naturels ou encore codes pondérés. Ils sont caractérisés par le fait que le poids du chiffre de rang **i** est **B** fois celui du rang **i-1**, **B** étant la base du système.

Il existe d'autres codes qui possèdent des avantages particuliers pour des utilisations particulières, on peut citer : le code Gray, code BCD (Binary Coded Décimal), code majoré de trois, code Aiken .

L'utilisation d'un code particulier peut rendre le traitement d'un message plus au moins économique du point de vue temps de traitement ou encombrement en mémoire ou encore en nombre de composant nécessaire pour effectuer le traitement.

1.5.1 Code Gray ou binaire réfléchi

Ce code est utilisé essentiellement dans la conversion d'une grandeur analogique en une grandeur numérique ; car dans ces conversions on a besoin d'un code dans lequel les grandeurs successives ne diffèrent que d'un seul caractère.

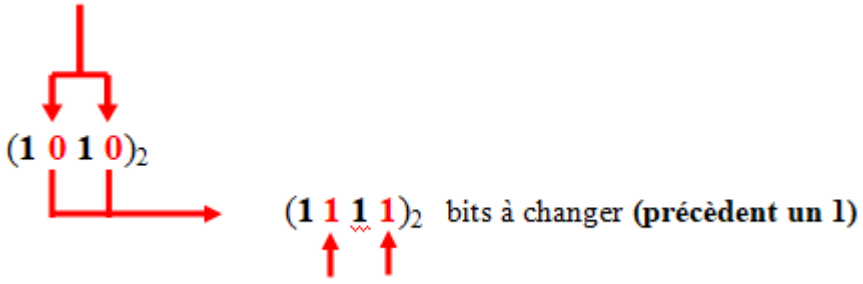
Le code binaire réfléchi est utilisé pour simplifier des équations dans les tableaux de KARNAUGH. Le principe consiste à changer l'état d'un seul bit entre deux nombres consécutifs.

Par exemple pour passer de 7 à 8 décimal, soit de 0111 à 1000 binaire naturel les quatre bits changent.

A. Conversion du Binaire Naturel vers le Binaire Réfléchi

❖ **Règle :** Pour convertir un nombre du binaire naturel au binaire réfléchi, il suffit de changer **le bit qui précède directement un bit1,** Voir (exemple 1 et 2).

❖ **Exemple1 :** Convertir le nombre binaire naturel $(1010)_2$ en binaire réfléchi.



❖ **Exemple2** Conversion des 15 premiers chiffres du binaire naturel en Gray

Tableau 06 : Exemple de conversion Binaire naturel - Gray

Décimal	Binaire naturel	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100

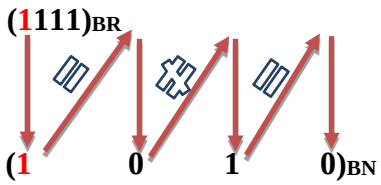
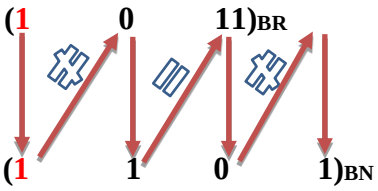
Décimal	Binaire naturel	Gray
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

B. Conversion du Binaire Réfléchi vers le Binaire Naturel

❖ **Règle** : Conversion du Binaire Réfléchi vers le Binaire Naturel :
Le premier bit à gauche reste inchangé
Il s'agit de comparer le bit b_{n+1} du **binaire naturel** et le bit b_n du **binaire réfléchi** le résultat est b_n du binaire naturel qui vaut **0** si $b_{n+1}=b_n$ ou **1** sinon, (voir exemple ci-dessous.)

❖ **Exemple:** Convertir les nombres binaires réfléchi $(1111)_{BR}$ et $(1111)_{BR}$ en binaire naturel

Tableau 07 : Principe de conversion du binaire réfléchi (Gray) en binaire naturel

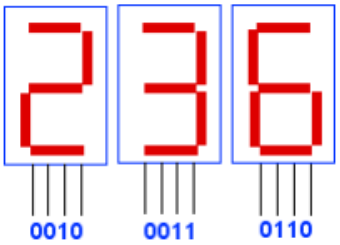
$(1111)_{BR}=(?)_{BN}$	$(1011)_{BR}=(?)_{BN}$
 $(1111)_{BR}=(1010)_{BN}$	 $(1011)_{BR}=(1101)_{BN}$

1.5.2 Code BCD : Binary Coded Décimal (Décimal codé en binaire) :

Ce code est utilisé par les calculateurs, il est destiné à l'affichage de valeurs décimales, chaque digit doit être codé en binaire sur 4 bits. Il est uniquement destiné à la saisie et à l'affichage de données, (voir exemple 1 et 2).

❖ **Exemple 1 :** 236 = 0010 0011 0110 (soit un mot de 12 bits)

Figure 08 : Nombre décimal code en DCB



Cours sur la numérotation, science de l'ingénieur en S, Académie Caen

❖ **Exemple 2:**

Tableau 09 : Exemple de conversion Binaire naturel - Gray

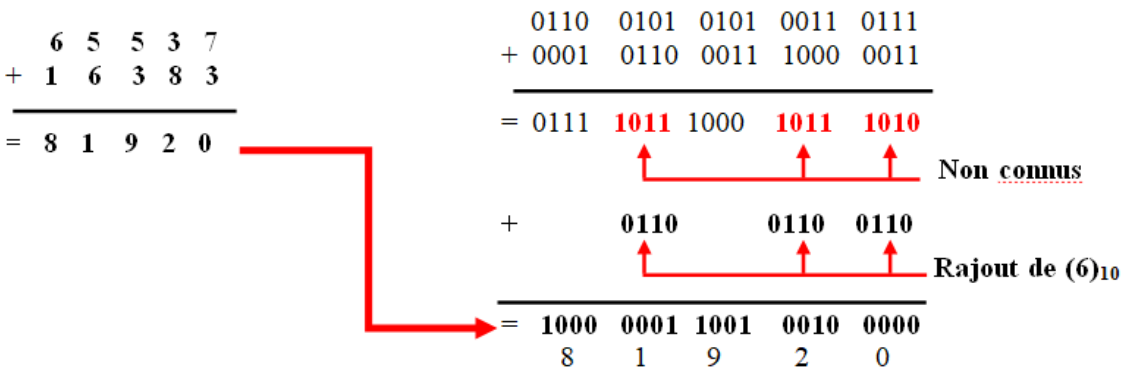
Décimal	Binaire	BCD
258	1 00000010	0010 0101 1000
32	10 0000	0011 0010
69	100 0101	0110 1001
4096	(- 1 0000 0000 0000	0100 0000 1001 0110

❖ **Addition dans le code BCD** :

Règle : Pour additionner deux nombres décimaux codés en binaire, il suffit de

1- Procéder à l'addition en binaire naturel ;
2- Ajouter le nombre $(6)_{10}$ c'est à dire $(0110)_2$, à tout quartet ne correspondant pas à une valeur connue par le code BCD (les nombres dépassant $(9)_{10}$ ou $(1001)_2$, voir exemple ci-dessous.

❖ **Exemple :** Faire l'addition suivante en DCB, $(65537)_{10} + (16383)_{10}$



1.5.3 Code majoré de trois (Excess 3)

Le code Excess-3 (aussi appelé code de Stibitz et parfois raccourci XS3 ou XS-3) est un code décimal binaire sur 4 bits (comme le BCD) créé pour optimiser certains calculs en base 10 sur d'anciens processeurs.

On a parfois recours à ce code en raison de la facilité avec laquelle on peut faire certains calculs arithmétiques. La valeur d'un mot en code majoré de trois est en fait égale au code DCB auquel on a ajouté 3.

❖ **Règle :** Le code majoré de trois consiste à prendre chaque chiffre décimal a part, à lui additionner 3, puis à convertir le résultat obtenu en binaire**,** comme le montre l'exemple suivant.

❖ **Exemple :** Convertir le nombre $(48)_{10}$ en code majoré de 3

4	8
+ 3	+ 3
= 7	= 11
↓	↓
0111	1011

$(48)_{10} = (0111\ 1011)_{XS3}$

1.5.4 Code binaire de Aiken

Pondéré par 2421, le code Aiken est un code auto complémentaire, c'est-à-dire les représentations de 2 chiffres dont la somme est 9 sont complémentaires l'une de l'autre.

❖ **Règle :**Le code Aiken peut être constitué par les règles suivantes :

- ✓ De 0 à 4 on code en binaire pur ;
- ✓ De 5 à 9 on ajoute 6 et on code en binaire pur , (voir exemple 1et 2)

Exemple 1: 5→5+6 =11

6→6+6= 12

❖ **Exemple 2**

Tableau 10 : Exemple de conversion Décimal - Aiken

Décimal	Aiken			
	2	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	0	1
8	1	1	1	0
9	1	1	1	1

1.5.5 Code ASCII

Le code **A.S.C.I.I**(American Standard Code for Information Interchange - traduisez « Code Americain Standard pour l'Echange d'Informations ») est un code alphanumérique, devenu une norme internationale. Il est utilisé pour la transmission entre ordinateurs ou entre un ordinateur et des périphériques. Sous sa forme standard, il utilise **7 bits**, ce qui permet de générer $\mathbf{2^7 = 128}$ caractères. Ce code représente les lettres alphabétiques majuscules et minuscules, les chiffres décimaux, des signes de ponctuation et des caractères de commande. Le code A.S.C.I.I est représenté sur le tableau suivant

Tableau 11 : Code ASCII

	0000 0	0001 1	0010 2	0011 3	0100 4	0101 5	0110 6	0111 7	1000 8	1001 9	1010 A	1011 B	1100 C	1101 D	1110 E	1111 F
- 000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
- 001	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
- 010	espace	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
- 011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
- 100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
- 101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
- 110		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
- 111	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Source : https://geekz0ne.fr

Chaque caractère possède un code sur **7 bits**. On ajoute un bit de contrôle appelé bit de parité calculé de telle façon que le nombre total de bits à 1 soit pair, (voir exemple suivant) :

❖ **Exemple :**

Calcul du bit de parité sur un exemple : **E =>? 100 0101**

E =>1100 0101 Soit : E => (C 5)H

Les caractères Spéciaux comme par exemple (**NUL, SOH, STX**……) sont des caractères de commande. Ils sont employés pour commander le fonctionnement d'un dispositif récepteur, telle qu'une imprimante, pour effectuer par exemple, un retour chariot ou l'avance ligne. Les caractères relatifs à ces codes ne sont pas supposés produire une marque sur le papier, toutefois, ils sont représentés par des symboles graphiques, si on les envoie à un écran d'affichage.

1.5.6 Représentation des nombres à virgule flottante

Pour représenter un nombre à virgule dans une machine informatique ou un système électronique, il a été nécessaire de trouver une écriture des nombres compatible avec la taille mémoire qu'on lui accorde. On a donc privilégié la notation scientifique et l'écriture en virgule flottante.

Cette méthode permet aussi de représenter des nombres très grands et très petits sans s'encombrer de zéros.

La représentation en virgule flottante consiste à représenter les nombres sous la forme suivante :

$$N_B = +/-1, M.B^E$$

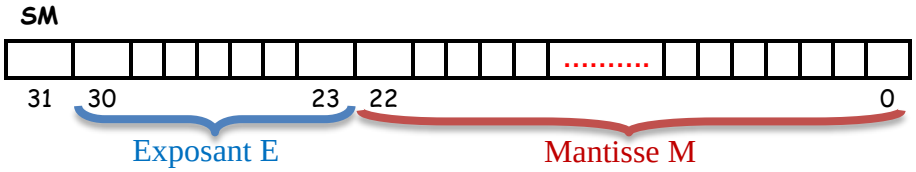
M est la mantisse,

B une base (2, 8, 10, 16, ...)

E l'exposant**.**

Souvent, on représente ces nombres par 1 bit pour le signe de la mantisse (noté SM), suivi d'un bloc de bits pour l'exposant et d'un autre bloc de bits pour la mantisse, (voir exemple 1 et 2).

Figure 12 : représentation en virgule flottante



Remarque 01 : Nous allons étudier la norme **IEEE754**, qui est la plus utilisée pour coder les réels, et qui utilise la méthode précédente avec la base **B = 2.**

La norme **IEEE754**utiliseplusieurs formats:

1. **IEEE754 simple précision** : sur 32 bits, SM : 1 bit ; Eb : 8 bits ; M : 23 bits (biais=127), (**Eb** =exposant + biais)
2. **IEEE754 double précision** : sur 64 bits, SM : 1 bit ; Eb : 11 bits ; M : 52 bits (biais=1023)
3. **IEEE754 double précision** : sur 80 bits, SM : 1 bit ; Eb : 15 bits ; M : 64 bits (biais=1023)

Exemple 1:

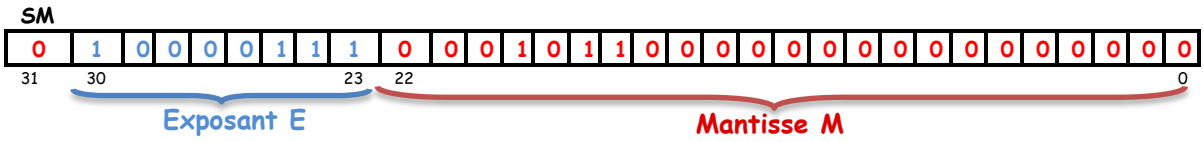
Nous voulons représenter $(278)_{10}$ au format IEEE754 simple précision. On commence par écrire 278 en base 2, puis on le met sous la forme 1, …….$x2^E$

$$(278)_{10} = (100010110)_2 = 1{,}0001011 \times 2^8$$

On a donc :

SM = 0 car 278 est positif
Eb = E + 127 = 8 + 127 = $(135)_{10}$ = $(10000111)_2$ sur 8 bits
M = 00010110000000000000000

On obtient la représentation suivante :



Exemple 02:

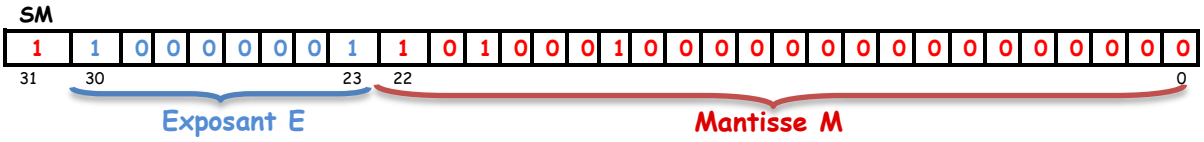
Nous voulons représenter $(-6{,}53125)_{10}$ au format IEEE754 simple précision. On commence par écrire -6,53125 en base 2, puis on le met sous la forme 1, …….$x2^E$

$$(6{,}53125)_{10} = (110{,}10001)_2 = 1{,}1010001 \times 2^2$$

On a donc :

SM =1 car -6,53125 est négatif
Eb = E + 127 = 2 + 127 = $(129)_{10}$ = $(10000001)_2$ sur 8 bits
M =10100010000000000000000

On obtient la représentation suivante :



1.5.7 Exercice 5Complétez le tableau suivant

Tableau 12: Tableau de conversion entre systèmes de numération et systèmes de codage

Code/Base	Nombre	Equivalent	Dans le Base/Code
2	111011101		Gray
10	7		XS3
2	110010		DCB
8	17		Gray
Gray	110010001010		2
Gray	1011		XS3

1.6 Résumé

On vient de terminer l'étude sur les systèmes de numérotation et de codage. Il apparaît intéressant de récapituler les points suivants.

Outre le système de numérotation décimale, il existe trois autres systèmes de numérotations couramment utilisés dans les dispositifs industriels, ce sont le système binaire, le système octal, et le système hexadécimal. Ces trois systèmes sont des systèmes pondérés où le poids de pondération de chaque chiffre dépend de son rang dans le nombre.

On se qui concerne les opérations (addition, soustraction, multiplication) leurs techniques de base restent exactement les mêmes qu'en notation décimale ; elles sont juste simplifiées de façon drastique parce qu'il n'y a que les deux chiffres 0 et 1

Parmi les techniques de codage les plus utilisées, on retrouve les codes *DCB*, le code *Gray* et le code *Aiken.*

- Le code DCB est un code pondéré. Chaque chiffre du système décimal est représenté dans ce code par une séquence de quatre bits. L'avantage de ce code est de fournir une méthode plus rapide et plus simple de représenter les nombres décimaux en nombres binaires.
- Le code Gray, aussi appelé "code binaire réfléchi", est un code non pondéré où les chiffres ne diffèrent des précédents que d'un seul bit. Ce code est très utilisé dans les convertisseurs analogiques numériques.
- Le code Aiken étant fréquemment utilisé dans les automatismes industriels, il est par conséquent nécessaire de le convertir aux interfaces en code 8421 et inversement Le code a été développé par Howard Hathaway Aiken et est encore utilisé aujourd'hui dans les horloges numériques , calculatrices de poche et autres appareils similaires.
- On a vu aussi comment sera représenté un nombre à virgule flottante dans une machine informatique ou un système électronique

Après avoir étudié les systèmes de numérotation et de codage, le deuxième chapitre va nous permettre de matérialiser ces circuits opérateurs arithmétiques et de codage, on utilisant le concept de l'algèbre de Boole

1.7 Exercice de synthèse N°1

- **Spécialité** : Réseaux Télécom Filaires
- **Module** : Electronique numérique
- **Durée :** 1h
- **But :** Familiariser les stagiaires avec les systèmes de numération et les codes utilisés par les appareils et équipements numériques et informatiques pour traiter les informations.
- **Matériel requis** : Documents et support de cours
- **Mise en situation :** Réaliser les conversions et opérations suivantes
- **Marche à suivre :**

A. Complétez le tableau13 suivant

Tableau 13 : Conversion entre systèmes de numération

Base10	Base 2	Base 8	Base16
125			
			F0CD
	1011,11		
		125,4	

B. Complétez le tableau 14 suivant

Tableau 14 : Conversion entre systèmes de numération et de codage

Base	Nombre	Equivalent	Dans la base
10	121,875		2
16	A01		DCB
DCB	10010110		Gray
8	25,45		16

C. Soit les deux tableaux suivants

- Complétez le tableau 15 suivant

Tableau 15 : Complémentations

Nombre	Complément à 1	Complément à 2
A=10000100		
B=01110001		

- Calculer en Binaire sur 8 bits

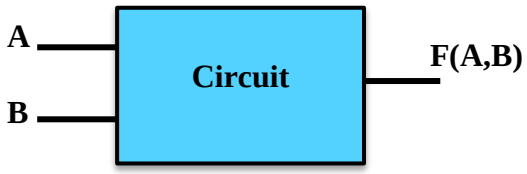
Tableau 16: Operations arithmétique

A+B	
A-B	
B-A	
-A-B	

CHAPITRE 2 ALGEBRE DE BOOLE

Les machines numériques sont constituées d'un ensemble de circuits électroniques. Chaque circuit fournit une fonction logique bien déterminée (addition, comparaison,….). La fonction F (A, B) (figure 17) peut être : la somme de A et B, ou le résultat de la comparaison de A et B ou une autre fonction

Figure 17 : Circuit électronique



Pour concevoir et réaliser ce circuit on doit avoir un modèle mathématique de la fonction réalisée par ce circuit et qui doit prendre en considération le système binaire.
Le modèle mathématique utilisé est ***l'Algèbre de BOLLE*** de ***George BOOLE*** qui est un mathématicien anglais (1815-1864).Il a fait des travaux dont les quels les fonctions (expressions) sont constituées par des variables qui peuvent prendre les valeurs '**OUI**' ou '**NON**'.
Ces travaux ont été utilisés pour faire l'étude des systèmes qui possèdent deux états

- Le système peut être uniquement dans deux états E1 et E2 tel que E1 est l'opposé de E2.
- Le système ne peut pas être dans l'état E1 et E2 en même temps

Ces travaux sont bien adaptés au Système binaire (0 et 1).

2.1 Variables et fonctions logiques

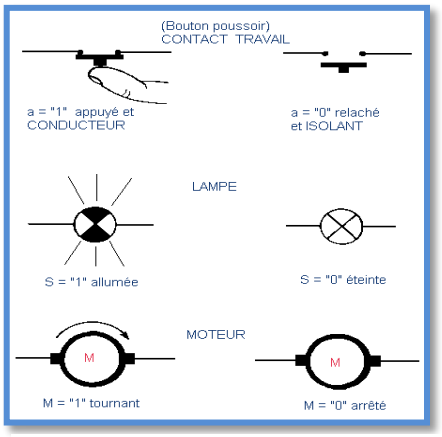
Le fonctionnement des systèmes numériques repose sur la manipulation de variables et fonctions dont les valeurs sont représentées par des grandeurs physiques dites binaires car ne pouvant prendre que deux valeurs (généralement notées 0 et 1). La structure mathématique permettant de formaliser les opérations de manipulation de ces grandeurs binaires est dites algèbre de Boole. Nous nous intéressons dans ce chapitre aux bases et aux propriétés fondamentales de l'algèbre de Boole indispensables à la compréhension du fonctionnement des systèmes numériques.

2.1.1 Variable logique (booléenne)

Une variable logique est une grandeur qui ne peut prendre que deux valeurs 0 ou 1(figure 15) ,de manière général, elle peut être associé à un événement, si ce dernier est vrai, alors la variable prend la valeur 1, si non 0, voire (exemple suivant) .

➢ **Exemple :** état d'un interrupteur, d'un bouton poussoir, la présence d'une tension,...Soit a la variable associée à l'état d'un bouton poussoir, alors ***A=0 (faux ou bas)*** signifiequ'il n'est pas actionné, ***A= 1 (vrai ou haut)*** signifie qu'il est actionné,voir (figure 18).

Figure 18 : Quelques illustrations de l'état logique



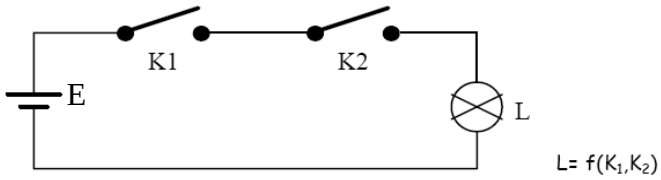
Source : https://www.electronique-et-informatique.fr/

2.1.2 Fonction logique (booléenne)

C'est une fonction d'une ou plusieurs variables booléennes, ne pouvant prendre elle-même qu'une des deux valeurs 0 ou 1. Pour la définir, il faut préciser sa valeur pour toutes les combinaisons possibles,comme le montre l'exemple suivante.

➢ **Exemple :** La lampe **L** s'allume lorsque les interrupteurs **K1** et **K2** sont fermés tous les deux. Elle est éteinte dans tous les autres cas, voir (figure19).

Figure 19 : Commutateurs en série



Source : Bases des systèmes Numériques (Par A. Oumnad)

Le fonctionnement est illustré par la table de vérité suivante :

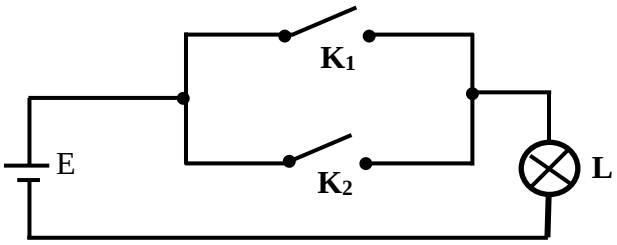
Le doublet des deux variables (**K1, K2**) peut prendre **2^2 valeurs** distinctes. Dans le cas général de **N** variables, il y aura **2^N** configurations possibles
Les variables **K1** et **K2** sont dites les entrées du système, **L** est dite la sortie du système.
De la table de vérité en peut déduire **$L= K_1 . K_2$**

Tableau 20:Table de vérité

K1	K2	L
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

2.1.3 Exercice 06 Donner la table de vérité du circuit électronique de la figure suivant

Figure 21 : Commutateurs en parallèle



2.2 Opérateurs Logiques (Portes logiques)

Les fonctions logiques sont conçues à partir d'un groupe d'opérateurs élémentaires appelés **« portes ».** Chaque opérateur est représenté par un symbole et sa fonction est définie par une table de vérité.

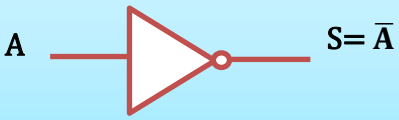
2.2.1 Porte NON (NOT)

❖ **Table de vérité :**

A	S
0	1
1	0

❖ **Equation : $S = \overline{A}$**

❖ **Symbole :**



$\overline{A}$ est le complément à 1 de A

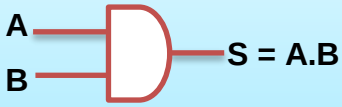
2.2.2 Porte ET (AND)

❖ **Table de vérité :**

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

❖ **Equation: S= A et B = A.B**

❖ **Symbole**



A.B est **vrai** si et seulement si **A est vraie et B est vraie**

2.2.3 Porte OU (OR)

❖ **Table de vérité :**

A	B	S
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

❖ **Equation: S= A ou B = A+B**

❖ **Symbole :**

A+B est vraie si au moins une des deux variables A ou B est vrai

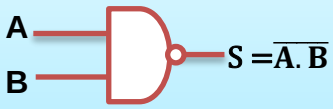
2.2.4 Porte NON-ET (NAND)

❖ **Table de vérité :**

A	B	S
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**0**

❖ **Equation:=** $\overline{A \; et \; B} = \overline{A.B}$

❖ **Symbole :**



C'est le **complément** de l'opérateur **ET.** C'est l'opérateur le plus couramment utilisé dans la pratique.

2.2.5 Porte NON-OU (NOR)

❖ **Table de vérité :**

A	B	S
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

❖ **Equation:** $S=\overline{A \; ou \; B} = \overline{A + B}$

❖ **Symbole :**

C'est le complément de l'opérateur OU

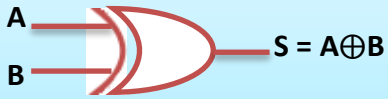
2.2.6 Porte OU EXCLUSIF (XOR)

❖ **Table de vérité :**

A	B	S
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

❖ **Equation: S=$\bar{A}$.B + A.$\bar{B}$ = A⊕B**

❖ **Symbole :**



A⊕Best **vrai**e si une des deux variables A ou B est **vrai**e mais pas les deux à la fois.

2.2.7 Porte NON-OU EXCLUSIF (XNOR)

❖ **Table de vérité :**

A	B	S
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**1**

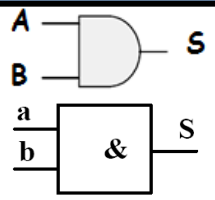
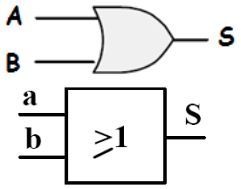
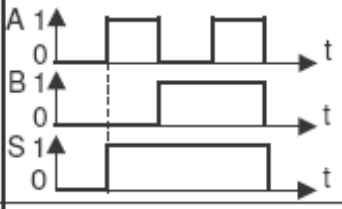
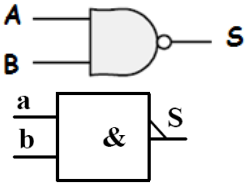
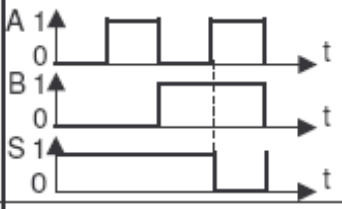
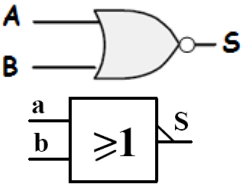
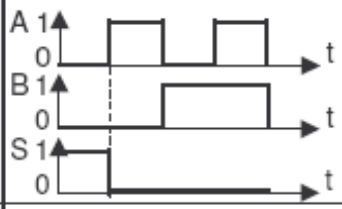
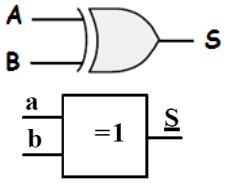
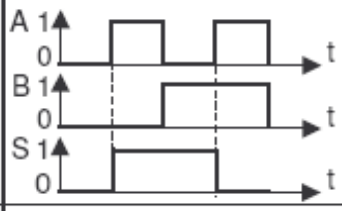
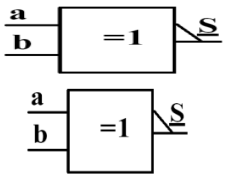
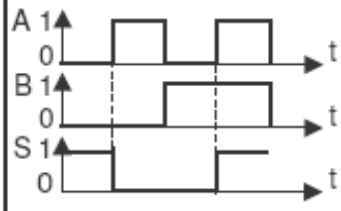
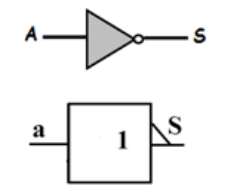
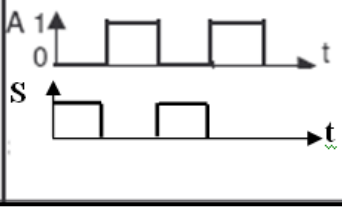
❖ **Equation: S= $\bar{A}.\bar{B}$ + A.B = $\overline{A \oplus B}$**

❖ **Symbole :**

$\overline{A \oplus B}$ est **vrai**e si **les deux variables à la fois** A et B sont **vrai**e

2.2.8 Représentation graphique des fonctions

Tableau *22* Commutateurs en parallèle

Fonction	Symbole	Equation	Table de vérité	Chronogramme
ET AND		$S = A \bullet B$	A B S 0 0 0 0 1 0 1 0 0 1 1 1	
OU OR		$S = A + B$	A B S 0 0 0 0 1 1 1 0 1 1 1 1	
ET-NON NAND		$S = \overline{A \bullet B}$	A B S 0 0 1 0 1 1 1 0 1 1 1 0	
OU-NON NOR		$S = \overline{A + B}$	A B S 0 0 1 0 1 0 1 0 0 1 1 0	
OU exclusif		$S = A \oplus B$ ou $S = A \bullet \overline{B} + \overline{A} \bullet B$	A B S 0 0 0 0 1 1 1 0 1 1 1 0	
NON OU exclusif		$S = \overline{A \oplus B}$ ou $S = \overline{A} \bullet \overline{B} + A \bullet B$	A B S 0 0 1 0 1 0 1 0 0 1 1 1	
NON (NOT)		$S = \overline{A}$	A S 0 1 1 0	

Remarque :

- **1^er Symbole :** Symbole **A.N.S.I.** (Américan National Standards Institute) ou « **AMERICAIN** ». C'est ceux que l'on utilisera dans nos schémas électriques
- **2^iem Symbole:** Symbole **I.E.E.E.** (Institute of Electrical and Electronics Ingenieers) ou « **EUREPEEN** »

2.2.9 Exercice 07 Tracer le circuit logique des fonctions suivantes :

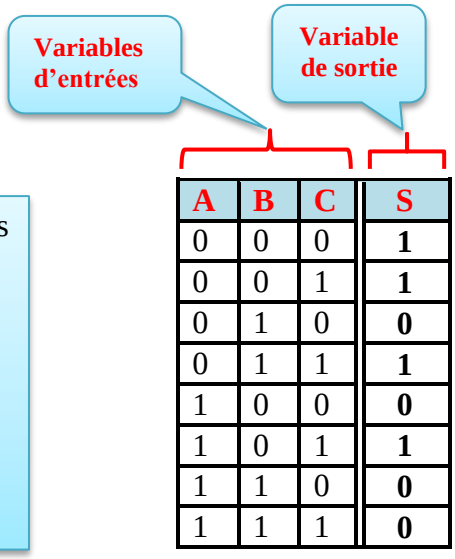
S1 = (A + B). (C + D)

S2 = $(\bar{A}$ + B). $\overline{C.D}$

2.3 Table de vérité

La table de vérité (***TV***) est un tableau qui permet de connaître systématiquement les états que peut prendre une fonction logique pour TOUTES les combinaisons des variables d'entrés. En d'autres termes, c'est un tableau qui interprète logiquement un cahier de charge en tenant compte de toutes les éventualités voir figure 23.

Tableau 23 : Exemple de TV à 3 variables



A	B	C	S
0	0	0	**1**
0	0	1	**1**
0	1	0	**0**
0	1	1	**1**
1	0	0	**0**
1	0	1	**1**
1	1	0	**0**
1	1	1	**0**

Dans une TV pour **n** variables, nous aurons **2^n** combinaisons différentes, c'est-à-dire :

- Pour **1** variable, la TV aura $2^1 = 2$ lignes
- Pour **2** variables, la TV aura $2^2 = 4$ lignes
- Pour **3** variables, la TV aura $2^3 = 8$ lignes
- Pour **4** variables, la TV aura $2^4 = 16$ lignes

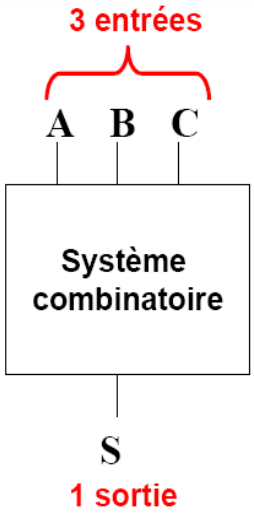
2.3.1 Formes canoniques d'une fonction Booléenne

Si une fonction logique possède **N** variables logiques on aura doc **2^N** combinaisons.Les **2^N** combinaisons sont représentés dans une table qui s'appelle table de vérité.

Une table de vérité recense l'ensemble des états d'une sortie pour toutes les combinaisons possibles des variables d'entrée, voir exemple suivant.

➢ **Exemple :** Soit un système combinatoire à 3 entrées **A**, **B**, **C** et une sortie **S**

Figure 24 : Système combinatoire



❖ **Table de vérité**

A	B	C	S	
0	0	0	0	→ $A+B+C$: max terme
0	0	1	0	→ $A+B+\bar{C}$: max terme
0	1	0	0	→ $A+\bar{B}+C$: max terme
0	1	1	1	→ $\bar{A}.B.C$: min terme
1	0	0	0	→ $\bar{A}+B+C$: max terme
1	0	1	1	→ $A.\bar{B}.C$: min terme
1	1	0	1	→ $A.B.\bar{C}$: min terme
1	1	1	1	→ $A.B.C$: min terme

D'après la table de vérité précédente en déduit

➢ S = somme min termes

S (A, B, C) = ($\bar{A}$.B.C) + (A.$\bar{B}$.C) + (A.B.$\bar{C}$) + (A.B.C)

C'est la **Première forme canonique (**somme des produits).Cette forme est la forme la plus utilisée.

➢ S = produit des max termes

S (A, B, C) = (A+B+C). (A+B+$\bar{C}$).(A+$\bar{B}$+C). ($\bar{A}$+B+C)

C'est la **Deuxième forme canonique (**produit des sommes).

❖ **Remarque :**

Première forme canonique = expression des 1 de la fonction

Deuxième forme canonique = expression des 0 de la fonction

Les deux formes canoniques sont équivalentes

On choisit celle qui donne le résultat le plus simple peu de 0 => deuxième forme / peu de 1 => première forme

2.3.2 Déduction de la TDV à partir de la fonction logique

Généralement on travaille avec la 1ère forme canonique afin de déduire la table de vérité, donc les monômes vont être représentés par (un 1, deux 1 ou quatre 1) dans la table de vérité selon le nombre de variables dans les monômes, voir (exemple 1 et 2).

- ✓ Monôme qui contient tous les variables sera représenté par 2^0 **(1)** dans la table de vérité, c'est à dire un seul **1**
- ✓ Monôme avec **1** variable de moins sera représenté par 2^1 **(1)** dans la table de vérité, c'est à dire deux **1**
- ✓ Monôme avec **2** variables de moins sera représenté par 2^2 **(1)** dans la table de vérité, c'est à dire quatre **1**
- ✓ Monôme avec **3** variables de moins sera représenté par 2^3 **(1)** dans la table de vérité, c'est à dire huit **1**

❖ **Exemple1 :** Déduire la TDV de la fonction suivante : F (X, Y, Z) $=\overline{X}.\ Y + \overline{Y}.Z$

➢ **Table de vérité**

X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

$\overline{X}.Y$

$\overline{Y}.Z$

❖ **Exemple2 :** Déduire la Table de vérité(TDV) de la fonction suivante :S(A, B, C, D) = $A.\overline{D} + \overline{A}.B.\overline{D}$

➢ **Table de vérité**

A	B	C	D	S
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	0	1	0

$\overline{A}.B.\overline{D}$

$A.\overline{D}$

2.3.3 Exercice 08

A. Donner la table de vérité de la fonction suivantes : $F_1 = X.Y.Z + X.(Y + \overline{Z})$

B. Déduire de la table de vérité la deuxième forme canonique de F_1

2.3.4 Exercice 9

Établir la table de vérité de la fonction suivante, puis l'écrire sous la 1ère forme canonique : F1= $(X + Y).(\overline{X} + Y + Z)$

2.4 Schéma d'un circuit logique (Logigramme)

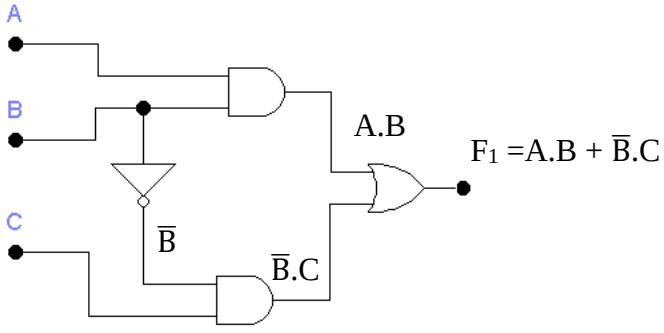
C'est la traduction de la fonction logique en un schéma électronique. Le principe consiste à remplacer chaque opérateur logique par la porte logique qui lui correspond, voir (exemple 1 et 2) .

2.4.1 Logigramme à l'aide des portes mixtes

❖ **Exemple1 :** Tracer le logigramme de la fonction suivante :

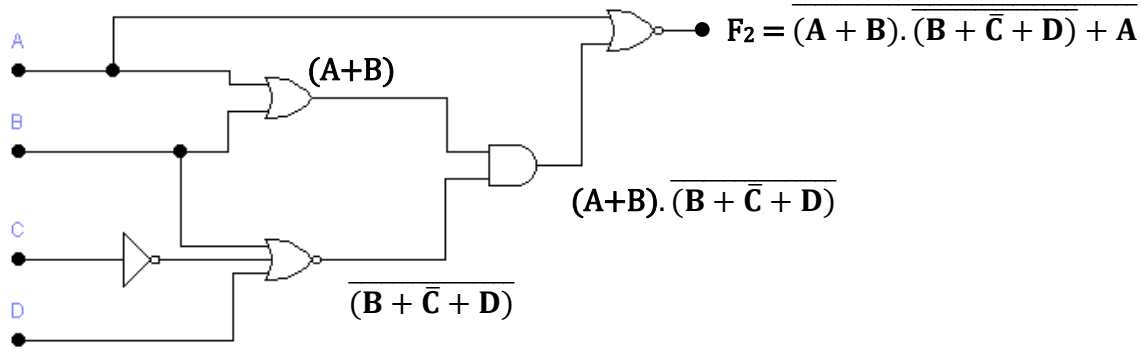
F_1 (A, B, C) =$A.B + \overline{B}.C$

Figure25 : Logigramme de la fonction F_1



❖ **Exemple2 : F_2 (A, B, C, D) =** $\overline{(A+B).\overline{(B+\bar{C}+D)}+A}$

Figure26 : Logigramme de la fonction F_2



2.4.2 Logigramme à l'aide des portes NAND ou NOR uniquement

Afin d'économisé le nombre de circuits intégré dans les circuits logique, ils sont généralement réaliser à l'aide de porte NAND ou de porte NOR uniquement, voir (exemple 1 et 2)

❖ **Exemple1 : Soit** la TDV suivante**,** trouver l'équation de F puis réaliser son circuit logique à l'aide de porte NAND uniquement.

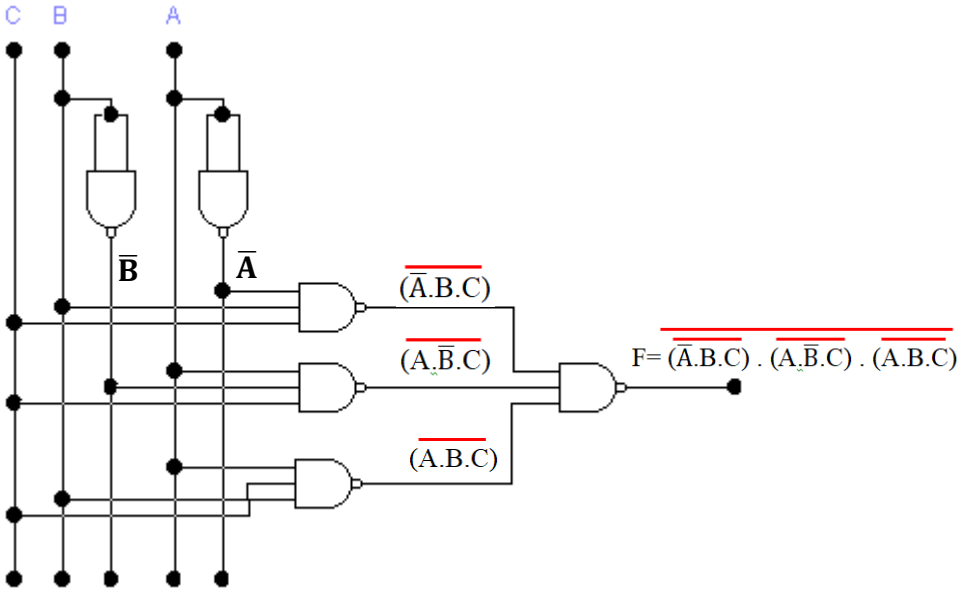
A	B	C	F	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	→ $\bar{A}.B.C$: min terme
1	0	0	0	
1	0	1	1	→ $A.\bar{B}.C$: min terme
1	1	0	0	
1	1	1	1	→ $A.B.C$: min terme

$F=\bar{A}.B.C + A.\bar{B}.C + A.B.C$ (**Première forme canonique** : somme des produits)

$F=\overline{\bar{A}.B.C + A.\bar{B}.C + A.B.C} = \overline{(\bar{A}.B.C)} \overline{+}\overline{(A.\bar{B}.C)} \overline{+}\overline{(A.B.C)}$

$F=\overline{(\bar{A}.B.C)} .\overline{(A.\bar{B}.C)} . \overline{(A.B.C)}$

Figure27 : Logigramme de F à l'aide des portes NAND



❖ **Exemple2 :** Soit la TDV précédente, trouver l'équation de F puis réaliser son circuit logique à l'aide de porte NOR uniquement.

A	B	C	F	
0	0	0	**0**	→ A+B+C: max terme
0	0	1	**0**	→ A+B+$\overline{C}$: max terme
0	1	0	**0**	→ A+$\overline{B}$+C: max terme
0	1	1	**1**	
1	0	0	**0**	→ $\overline{A}$+B+C: max terme
1	0	1	**1**	
1	1	0	**0**	→ $\overline{A}$+$\overline{B}$+C: max terme
1	1	1	**1**	

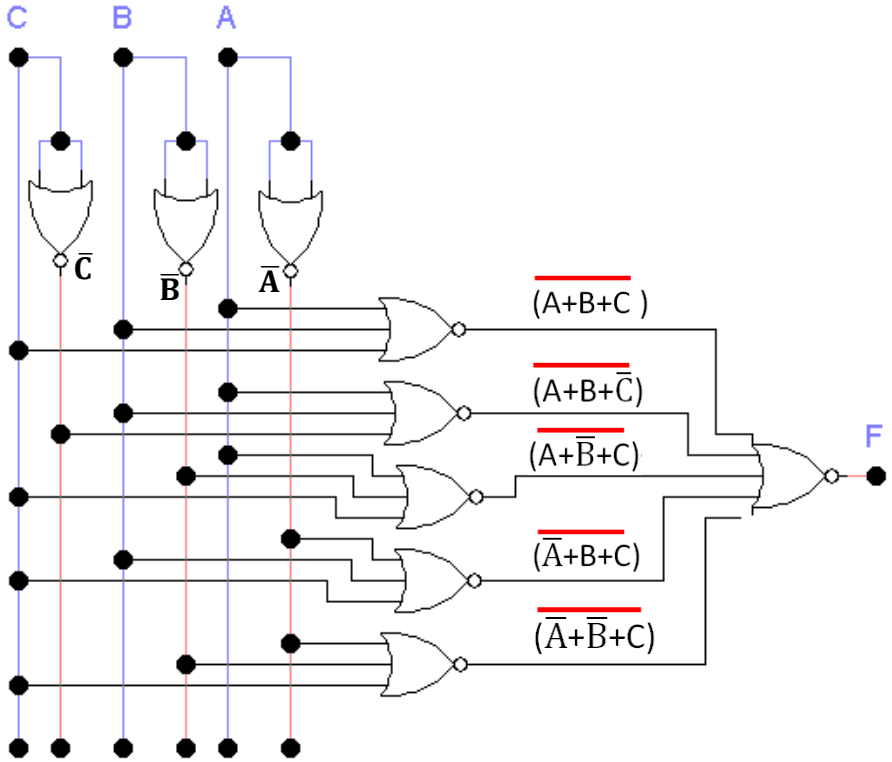
F=(A+B+C).(A+B+$\overline{C}$).(A+$\overline{B}$+C).($\overline{A}$+B+C).($\overline{A}$+$\overline{B}$+C)(**Deuxième forme canonique :** produit des sommes)

$\overline{F}$=$\overline{(A+B+C).(A+B+\overline{C}).(A+\overline{B}+C).(\overline{A}+B+C).(\overline{A}+\overline{B}+C)}$

F=$\overline{(A+B+C)}$. $\overline{(A+B+\overline{C})}$. $\overline{(A+\overline{B}+C)}$. $\overline{(\overline{A}+B+C)}$. $\overline{(\overline{A}+\overline{B}+C)}$

F=$\overline{(A+B+C)}$+$\overline{(A+B+\overline{C})}$+$\overline{(A+\overline{B}+C)}$+$\overline{(\overline{A}+B+C)}$+$\overline{(\overline{A}+\overline{B}+C)}$

Figure28 : Logigramme de F à l'aide des portes NOR



2.4.3 Exercice 10

Soit la TDV suivante :

A	B	C	S
			1
			1
			0
			0
			0
			1
			1
			0

a) Trouver la fonction S
b) Réalisez le circuit logique de S

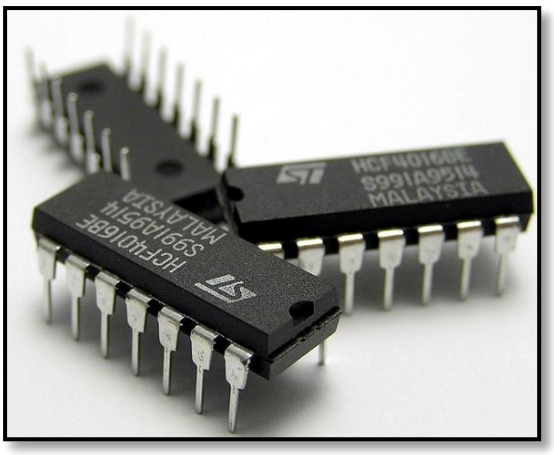
2.4.4 Exercice 11 Réaliser la porte EXOR a l'aide de portes NAND uniquement

2.5 Circuits intègres numériques

Le circuit intégré (***CI***), aussi appelé ***puce*** électronique, voir (figure 26) est un composant électronique reproduisant une ou plusieurs fonctions électroniques plus ou moins complexes. Il intègre souvent plusieurs types de composants électroniques de base dans un volume réduit, rendant le circuit facile à mettre en œuvre.

Les circuits intégrés se présentent généralement sous la forme de boîtiers pleins rectangulaires, noirs, équipés sur un ou plusieurs côtés voire sur une face, de 'pattes' permettant d'établir les connexions électriques avec l'extérieur du boîtier voir figure 29. Ces composants sont soudés sur un circuit imprimé, ou enfichés, à des fins de démontage, dans des supports eux-mêmes brasés sur un circuit imprimé.

Figure 29 : Circuits intégrés



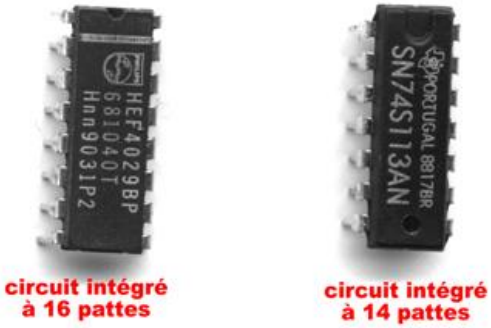
Source : http://for-ge.blogspot.com

2.5.1 Principe des circuits intégrés

En électronique, les portes logiques sont fabriquées et renfermées dans des **circuits intégrés**. Voir (Figure 30) .

On peut remarquer sur le haut des circuits intégré un petit creux appelé « ergo ». L'ergo permet d'orienter correctement le circuit intégré afin de repérer les différentes bornes.

Figure30 : Circuits intégrés



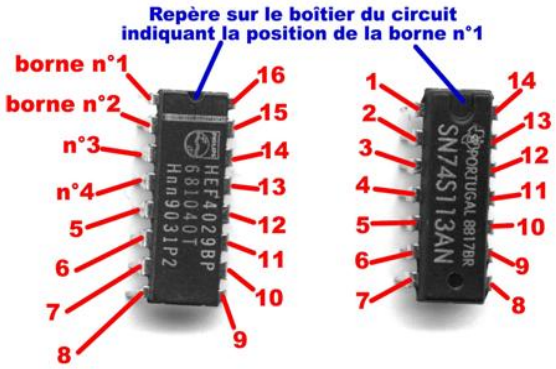
Source : www.gecif.net

Un circuit intégré renferme plusieurs portes logiques, dont les entrées et les sorties sont accessibles sur les différentes bornes (pattes) du circuit intégré. Pour identifier chaque borne sans ambiguïté, elles sont numérotées de manière normalisée en respectant le principe suivant :

- En regardant le circuit intégré avec le repère **(l'ergo)** vers le haut, la borne n°1 est la borne située en haut à gauche.
- Les autres bornes sont numérotées en tournant dans le sens inverse des aiguilles d'une montre.

Le principe de l'ergo reste vrai quel que soit le nombre de bornes (de « pattes ») du circuit intégré. La Figure 31 montre le numéro de chacune des bornes pour un circuit intégré à 16 bornes (16 « pattes ») et pour un circuit intégré à 14 bornes (14 « pattes »).

Figure31: *Bornes d'un circuit intégré*



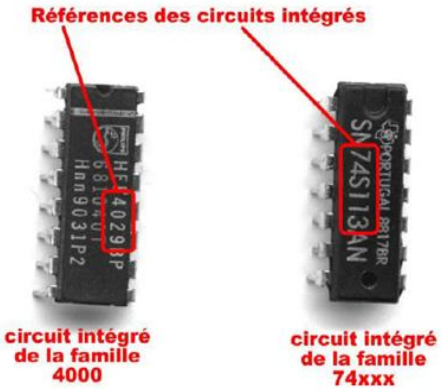
Source :www.gecif.net

2.5.2 Code de désignation d'un circuit intégrer

Un circuit intégré logique renferme des portes logiques. Mais comment savoir quel type de porte (des ***OU*** ? des ***ET-NON*** ? des ***Ou-Exclusif*** ? etc.). Pour cela, chaque circuit intégré possède une référence imprimée sur le dessus de son boîtier. Cette référence est composée de 4 à 7 caractères (chiffres et/ou lettres).

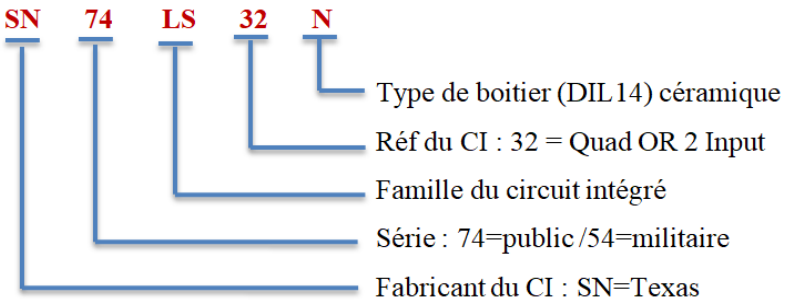
Par exemple, sur la (figure 32) ci-contre la référence du circuit de gauche est 4029, et la référence du circuit de droite est 74S113.

Figure32: *Références du circuit intégré*



Source :www.gecif.net

❖ Exemple du circuit intégrer **SN74LS32N**

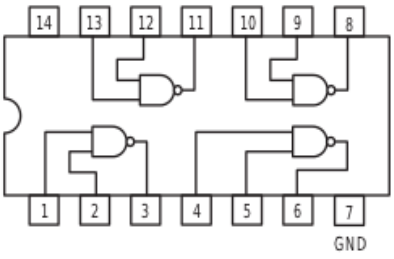


SN : Fabriquant TEXAS
74 : Série grand public (de 0°C à 70°C)
54 : Série militaire (de -55°C à 125°C)
LS : Famille Low power schottky
32 : Quadruple porte OU à 2 entrées
N : Boitier DIP 14 (Dual Inline Package : Double rangées de pattes)

2.5.3 Brochage d'un circuit intègre

En électronique tout comme en électrotechnique, le **brochage** d'un composant décrit le rôle de chacune des broches d'un connecteur ou d'un composant du plus simple au plus complexe des circuits intégrés. Le terme de brochage est synonyme de diagramme de connexion, voir (figure 33) brochage du CI 74 LS 00 (Quad 2-In NAND : 4 Portes NAND à 2 Entrées)

Figure33: Brochage du CI 74 LS 00



https://www.electronique-et-informatique.fr/

2.5.4 Familles des circuits intégrés logiques

A. Famille TTL (Transistor-Transistor-Logic)

La technologie TTL (Transistor Transistor Logic) a vu le jour en 1964. Ses circuits logiques sont réalisés avec des transistors bipolaires NPN. Cette technologie propose aujourd'hui le plus grand choix de circuits.

Les circuits intégrés de cette famille ont comme numéro d'identifications 74 ou 54. La série 54 se distingue par une plage plus large pour la tension d'alimentation et pour la température de fonctionnement (-55 à 125°C) elle est coûteuse, donc réservée aux utilisations dans des conditions ambiantes extrêmes : militaire, engins spatiaux,… Pour distinguer entre différents fabricants des C.I. un préfixe est réservé sur la face du circuit intégré.

Exemple : ***SN7402***, ***DM7402***, ***S7402***

- ✓ SN : Texas instruments
- ✓ DM : National semi conducteur
- ✓ S : Signetics

❖ **Les principales séries TTL**

La technologie TTL se décompose en 7 familles logiques voir (figure 32) :
Le tableau 34 regroupe les principales caractéristiques des différentes séries TTL.

Tableau 34: principales caractéristiques des différentes sériesTTL.

Série	Caractéristiques
74xx	**TTL standard** C'est la série standard elle offre un bon compromis entre vitesse et consommation
74Lxx	**TTL Low power :**faible consommation Possède une consommation réduite mais un temps de propagation plus élevé
74Fxx	**TTL Fast :** rapide Possède une vitesse de commutation plus rapide et consommation grande
74Sxx	**TTL Schottky :**réalisé avec des transistors schottky Elle est deux fois plus rapide que 74H pour la même consommation
74LSxx	**TTL Low power Schottky** :schottky faible consommation **Faible consommation et moins rapide que 74S**
74ASxx	**TTL Advanced Schottky** : technologie schottky avancée Fonctionne plus rapide et consomme moins, elle est meilleur que 74S
74ALSxx	**TTL Advanced Lowe power Schottky :**technologie schottky avancée faible consommation Meilleur version que 74LS

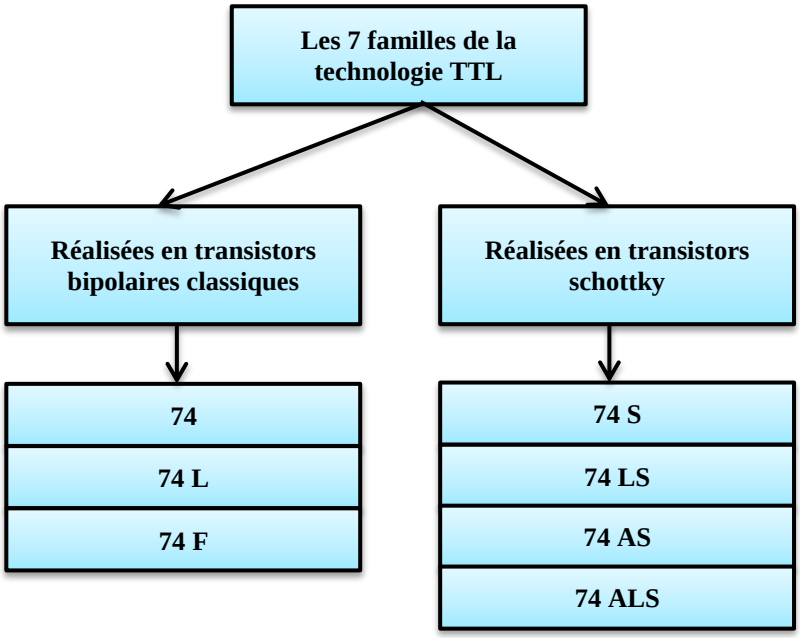
Dans la désignation d'une famille TTL, la signification des lettres L, S, F, et A est donc la suivante :

L = **L**ow power = faible consommation
S = Réalisée avec des transistors **S**chottky = rapidité
F = **F**ast = rapide
A= Technologie **A**vancée

❖ **Remarque :**

- ✓ Les 7 familles de la technologie TTL fonctionne avec une tension d'alimentation de +5V ± 5%.
- ✓ Les familles logiques les plus utilisées aujourd'hui en technologie TTL sont les familles LS et ALS.
- ✓ Les jonctions d'un transistor ***schottky*** sont réalisées à partir d'un semi-conducteur de type N ou P et d'un métal ; la conséquence est qu'un transistor ***schottky*** est bien plus rapide qu'un transistor bipolaire classique, du fait de la jonction Métal / Semi-conducteur.
- ✓ Parmi les 7 familles de la technologie TTL, 3 sont réalisées avec des transistors bipolaire classiques, et 4 avec des transistors shottky, voir (figure 35):

Figure 35 : Les 7 familles de la technologie TTL



B. Famille CMOS (Complementary Metal Oxide Semiconductor) :

La technologie CMOS a vu le jour en 1970. Ses circuits logiques sont réalisés avec des transistors MOS. L'avantage principal de cette technologie est la faible consommation (au détriment de la rapidité).

La technologie CMOS se décompose en 6 familles logiques ; on y distingue les CMOS classiques (2 familles) et les CMOS rapides (4 familles) , voir (figure 36):

1. Les 2 familles en CMOS classiques :

- ✓ la série 4000 (alimentation de 3 à 18 V)
- ✓ 74 C (même technologie que la série 4000, mais brochage et fonctions de la série 74)

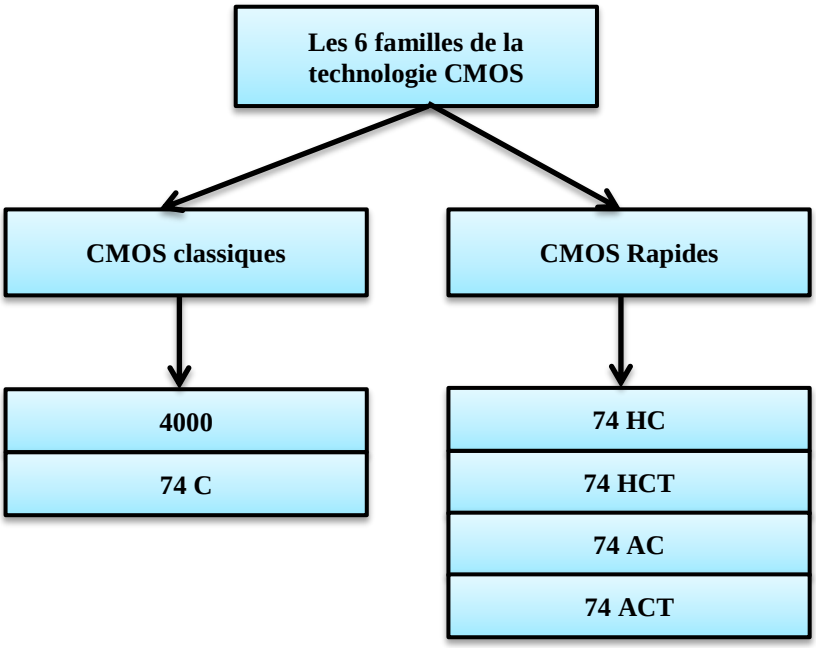
2. Les 4 familles en CMOS rapides :

- ✓ 74 HC (CMOS rapide comme la famille TTL LS, alimentation de 2 à 6 V)
- ✓ 74 HCT (compatibilité totale avec la famille TTL LS, alimentation 5V, rapidité et consommation de la famille CMOS HC)
- ✓ 74 AC (CMOS encore plus rapide que la famille HC)
- ✓ 74 ACT (CMOS AC compatible TTL)

Dans la désignation d'une famille CMOS, la signification des lettres C, H, T, et A est la suivante :

- ✓ **C** = technologie **C**MOS
- ✓ **H** = **H**igh speed = rapidité
- ✓ **T** = compatibilité avec la technologie **T**TL
- ✓ **A** = technologie **A**vancée

Figure 36 : Les 7 familles de la technologie CMOS



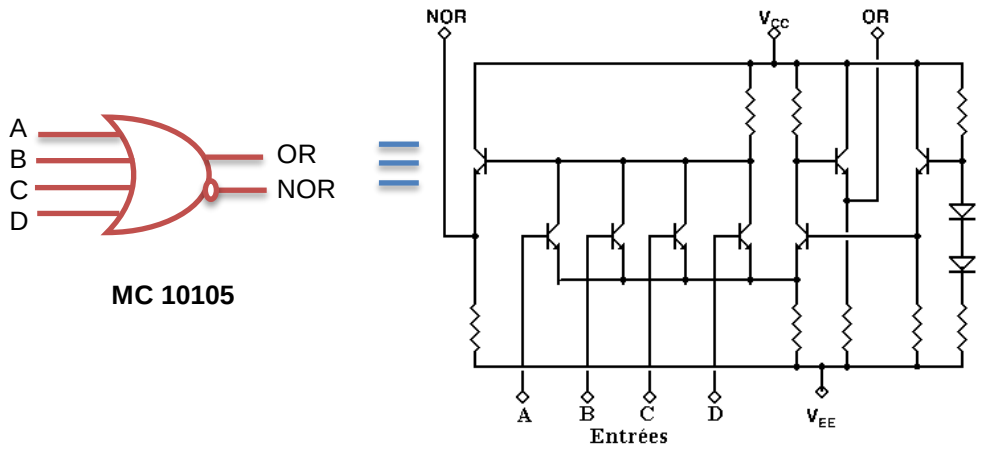
C. Famille ECL (Emitter Coupled Logic)

Cette famille logique introduite initialement par Motorola est la famille la plus rapide sur le marché, sa rapidité résulte de l'utilisation de la paire différentielle à couplage par les émetteurs, comme le montre la (figure 37).

La technologie ECL est la plus récente de toutes ; son avantage principal est la rapidité (100 fois plus rapide que la série 4000 en CMOS). Mais en échange d'un temps de propagation relativement faible (moins de 1 ns)

❖ Exemple de porte OR/NOR.

Figure 37: Porte OR /NOR en technologie ECL



Source : https://nanopdf.com

La famille ECL la plus répandue est la série 10000 (ou 10k) qui possède un temps de propagation de l'ordre de 2ns. Les séries les plus récentes 10kH et 100k possède un temps de propagation inférieure à 1 ns, cependant elles possèdent une consommation élevée.

Elle repose sur une technologie chère donc réservée aux applications où un gain en vitesse réduit le coût du système. Elle trouve son application aux hautes fréquences telles que les télécommunications, le traitement des signaux …

La technologie ECL présente les inconvénients suivants :

- ✓ consommation élevée
- ✓ difficulté de mise en œuvre
- ✓ prix des circuits logiques élevé
- ✓ nombre de fonctions logiques existantes limité dans cette technologie

Conclusion : Il existe donc aujourd'hui 7 familles en technologie TTL et 7 familles en technologie CMOS. Mais l'évolution des circuits intégrés n'est pas terminée : elle se poursuit dans l'utilisation de l'arséniure de gallium, qui permet de diminuer encore les durées de commutation des circuits pour atteindre seulement quelques dizaines de picosecondes (circuits VHIC, very high speed integrated circuits).

2.5.5 Échelle d'intégration

Un circuit intégré comprend sous des formes miniaturisées principalement des transistors, des diodes, des résistances, des condensateurs, plus rarement des inductances car elles sont plus difficilement miniaturisables.
L'échelle d'intégration définit le nombre de portes logiques par boîtier, voir(Tableau 38)

Tableau38 : Echelle d'intégration des circuits intégrer

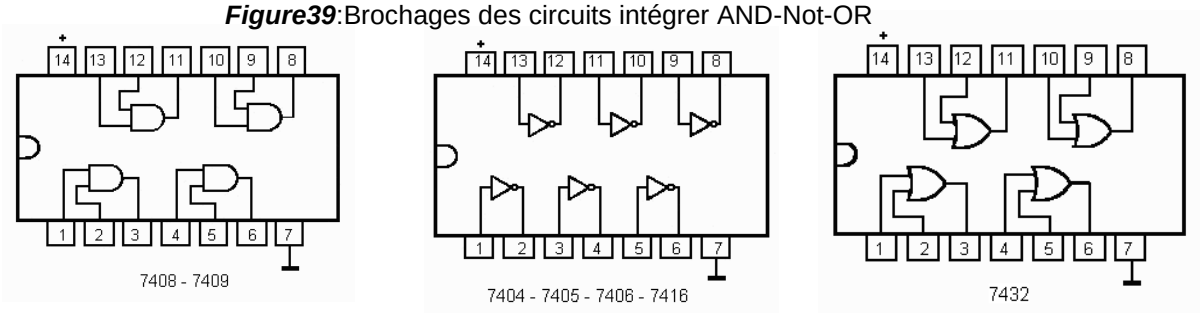
Nom	Signification	Année de sortie
SSI	Small Scale Integration : Intégration à petite échelle	1964
MSI	Mean Scale Integration : Intégration à moyenne échelle	1968
LSI	Large Scale Integration : Intégration à grande échelle	1971
VLSI	Very Large Scale Integration : Intégration à très grande échelle	1980
ULSI	ultra large-scale integration) Intégration ultra grande échelle	1984

Nom	Nombre de transistor par boitier	Nombre de porte logique par boitier	Utilisation
SSI	1 à 10	1 à 12	Réservées aux opérateurs élémentaires et aux bascules
MSI	10 à 500	13 à 99	Réservées pour les compteurs et les registres
LSI	500 à 20 000	100 à 9 999	Réservées aux circuits complexes tels que les mémoires et les microprocesseurs.
VLSI	20 000 à 1 000 000	10 000 à 99 999	
ULSI	1 000 000 et plus	1 00 000 et plus	

2.5.6 Exercice 12

Réaliser la fonction F= ($\overline{A}$+B).C, utiliser pour cela les circuits intégrer suivants, voir (figure 39)

Figure39:Brochages des circuits intégrer AND-Not-OR



Source :https://www.electronique-et-informatique.fr/

2.6 Simplification des fonctions logiques

Après la recherche de l'expression algébrique de la fonction, l'étape suivante consiste à minimiser le nombre de termes (de portes logiques) afin d'obtenir une réalisation matérielle plus simple donc plus facile à construire et à dépanner, en plus moins coûteuse.

Trois méthodes de simplification sont utilisées, on abordera que les deux premières:

- La méthode algébrique
- La méthode graphique (diagramme de KARNAUGH).
- La méthode de Quine-Mc Cluskey

2.6.1 Lois fondamentales de l'algèbre de Boole

Pour simplifier une fonction logique algébriquement, il est impératif de connaitre les lois fondamentales de l'algèbre de Boole voir (Tableau 40).

Tableau*40*: Lois fondamentales de l'algèbre de Boole

Propriétés	Opérateur OU	Opérateur ET
D'involution	$\overline{\overline{A}} = A$	
Commutativité	A+B = B+A	A. B = B. A
Associativité	(A+B)+C =A+ (B +C)=A+B+C	(A.B).C=A. (B.C)=A.B.C
Distributivité	A+(B.C)= (A+B). (A+C)	A. (B+C)=(A.B) +(A.C)
Complémentarité	A+$\overline{A}$=1	A.$\overline{A}$=0
Idempotence	A + A = A	A. A = A
Elément neutre	A + 0 = A A + 1 = 1	A. 0 = 0 A. 1 = A
Absorption	A + (A. B) = A A + ($\overline{A}$. B) = A + B	A. (A + B) = A A. ($\overline{A}$ + B) = A . B
Théorème de MORGAN	$\overline{A + B}=\overline{A} . \overline{B}$	$\overline{A . B}=\overline{A} + \overline{B}$

2.6.2 Simplification en utilisant la méthode algébrique

Il s'agit d'appliquer les théorèmes et les propriétés de l'algèbre de Boole pour obtenir une expression la plus simple de la fonction, voir (exemple 1 et 2).

❖ **Exemple 1** Simplifier algébriquement la fonction suivante :

S=A.B.C + A.$\bar{B}$. ($\overline{\bar{A}\,.\,\bar{C}}$)

S=A.B.C + A.$\bar{B}$. ($\bar{\bar{A}}$+$\bar{\bar{C}}$) **[$\bar{\bar{A}}$=A]**

S=A.B.C + A.$\overline{B}$. (A+C)

S=A.B.C + A.$\overline{B}$.A+A.$\overline{B}$.C **[A.A=A]**

S=A.B.C + A.$\overline{B}$ +A.$\overline{B}$.C

S=A.$\overline{B}$ + A.C.(B+$\overline{B}$) **[B+$\overline{B}$=1]**

S=A.$\overline{B}$ + A.C

S=A. ($\overline{B}$+C)

❖ **Exemple 2** Simplifier algébriquement la fonction suivante : F=$\overline{\bar{A}.B + (\overline{\bar{A} + B})}$

F=$\overline{\bar{A}.B + (\overline{\bar{A} + B})}$=$\overline{\bar{A}.B}$. ($\overline{\overline{\bar{A} + B}}$)

F=($\overline{\bar{A}.B}$).($\overline{A}$+B)= (A+$\overline{B}$).($\overline{A}$+B)

F= A.$\overline{A}$+A.B+$\overline{B}$.$\overline{A}$+$\overline{B}$.B

F= A.B+$\overline{A}$.$\overline{B}$

2.6.3 Tableau de KARNAUGH

Le diagramme de **Karnaugh** est un outil graphique, méthodique. Il permet d'obtenir une solution optimale à la simplification logique.

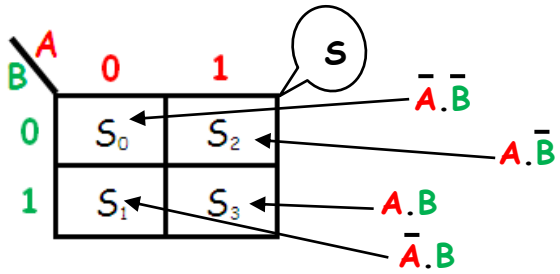
- Le diagramme de Karnaugh consiste à présenter les états d'une fonction logique, non pas sous la forme d'une table de vérité, mais en utilisant un tableau à double entrée.
- Chaque case du tableau correspond à une combinaison des variables d'entrées, donc à une ligne de la table de vérité.
- Le tableau de Karnaugh aura autant de cases que la table de vérité possède de lignes.
- Les lignes et les colonnes du tableau sont numérotées selon le code **binaire réfléchi** (**code de Gray**) : à chaque passage d'une case à l'autre, une **seule** variable change d'état (**cases adjacentes**).
- Le diagramme de Karnaugh est un tableau de **2^N cases**, **N** étant le nombre de variables.

2.6.3.1Diagramme de KARNAUGH à 2 variables

A. Table de vérité

	A	B	S
0	0	0	S_0
1	0	1	S_1
2	1	0	S_2
3	1	1	S_3

B. Tableau de KARNAUGH

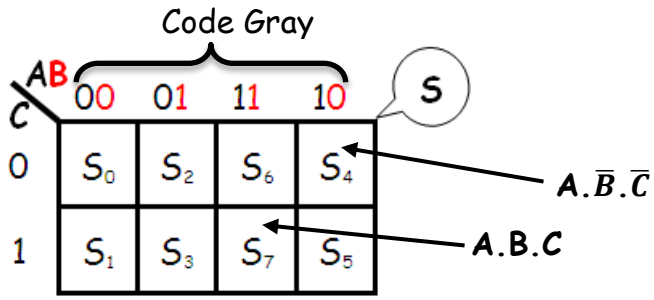


2.6.3.2Diagramme de KARNAUGH à 3 variables

A. Table de vérité

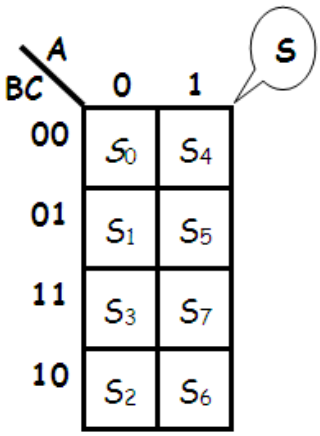
	A	B	C	S
0	0	0	0	S_0
1	0	0	1	S_1
2	0	1	0	S_2
3	0	1	1	S_3
4	1	0	0	S_4
5	1	0	1	S_5
6	1	1	0	S_6
7	1	1	1	S_7

B. Tableau de KARNAUGH



- **Remarque 01** Dans tableau de KARNAUGH deux cases sont dites **ADJACENTES** si **UNE** seule variable change d'état d'une case à l'autre
- **Exemple :** Les cases **S_0** et **S_2**sont des cases ADJACENTES
  (**$S_0=\bar{A}.\bar{B}.\bar{C}$ et $S_2=\bar{A}.B.\bar{C}$** Une seul variable change d'état c'est B)

Remarque 01 Autre représentation du tableau de KARNAUGH à 3 variables

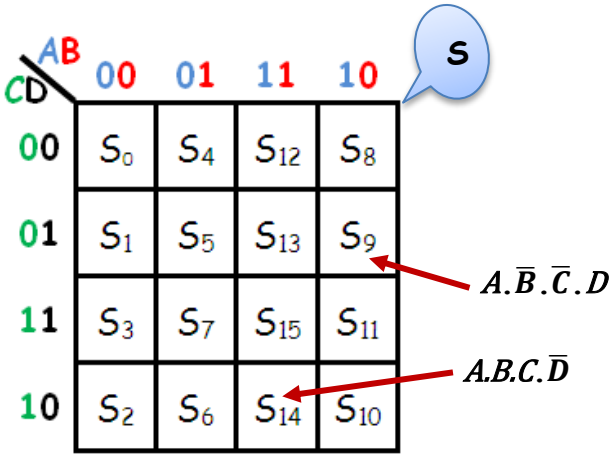


2.6.3.3Diagramme de KARNAUGH à 4 variables

A. Table de vérité

	A	B	C	D	S
0	0	0	0	0	S_0
1	0	0	0	1	S_1
2	0	0	1	0	S_2
3	0	0	1	1	S_3
4	0	1	0	0	S_4
5	0	1	0	1	S_5
6	0	1	1	0	S_6
7	0	1	1	1	S_7
8	1	0	0	0	S_8
9	1	0	0	1	S_9
10	1	0	1	0	S_{10}
11	1	0	1	1	S_{11}
12	1	1	0	0	S_{12}
13	1	1	0	1	S_{13}
14	1	1	1	0	S_{14}
15	1	1	1	1	S_{15}

B. Tableau de KARNAUGH



2.6.4 Simplification en utilisant le tableau de KARNAUGH

On a pu s'apercevoir que la méthode de simplification algébrique devenait vite très longue et fastidieuse dès que le nombre de variables devenait important.

La méthode du tableau de **KARNAUGH** va nous permettre d'effectuer des simplifications beaucoup plus rapidement sans avoir à écrire de longues équations

Règles de simplification :

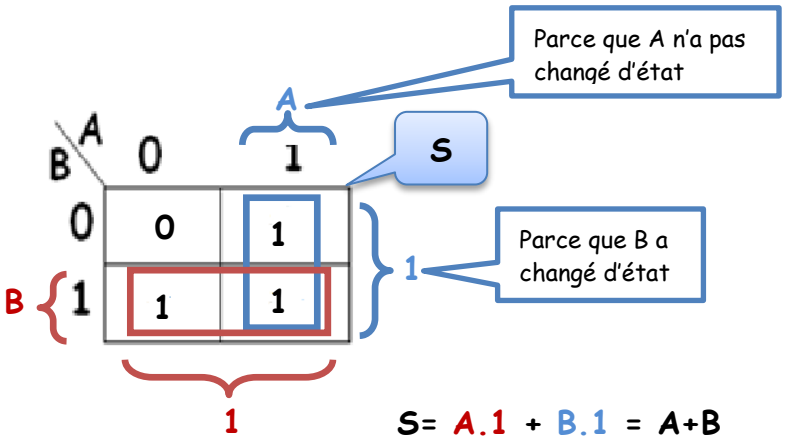
- ✓ Constituer des groupes de **1, 2, 4, 8,16**…… cases **ADJACENTES** (2^n cases) (on choisit des cases contenant des **1** ou des **0** pas les deux à la fois)
- ✓ Ces groupes de taille maximale, doivent être carré ou rectangulaire ;
- ✓ Ils doivent contenir un nombre de cases égal à une puissance de deux ;
- ✓ Les bords de la table doivent êtres adjacents ;
- ✓ On ne retient que les variables dont l'état logique d'entré n'est pas modifié à l'intérieur du groupement ;
- ✓ Les mêmes termes peuvent participer à plusieurs groupements,
- ✓ On doit avoir le moins de groupes possible, chaque groupe doit contenir le maximum de terme (1 ou 0)
- ✓ Les variables d'un même groupement sont liées par une fonction ET(.), les groupements sont liés par des fonctions OU (+).
- ✓ Après avoir fait la somme des fonctions de chaque groupement on obtiendra l'équation finale de **S**

2.6.4.1 Simplification d'une fonction à 2 variables

A. Table de vérité

	A	B	S
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

B. Tableau de KARNAUGH



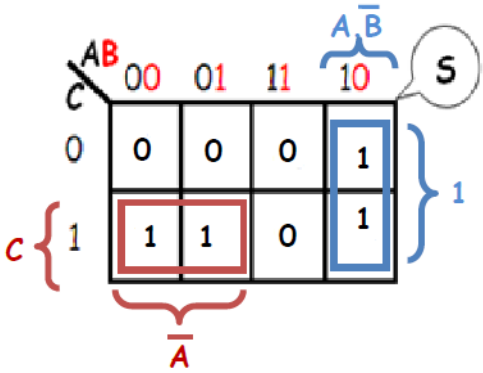
2.6.4.2 Simplification d'une fonction à 3 variables

A. Exemple 1

❖ Table de vérité

	A	B	C	S
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

❖ Tableau de KARNAUGH



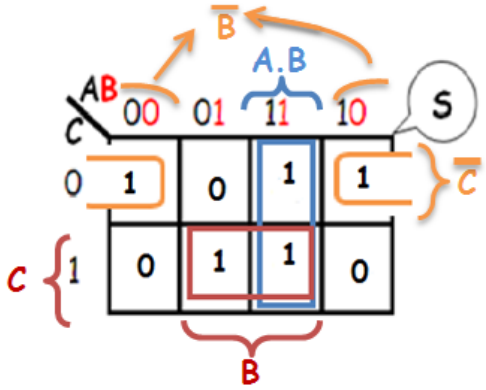
$S = \overline{A}.C + A.\overline{B}$

B. Exemple 2

❖ Table de vérité

	A	B	C	S
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

❖ Tableau de KARNAUGH



$S = B.C + A.B + \overline{B}.\overline{C}$

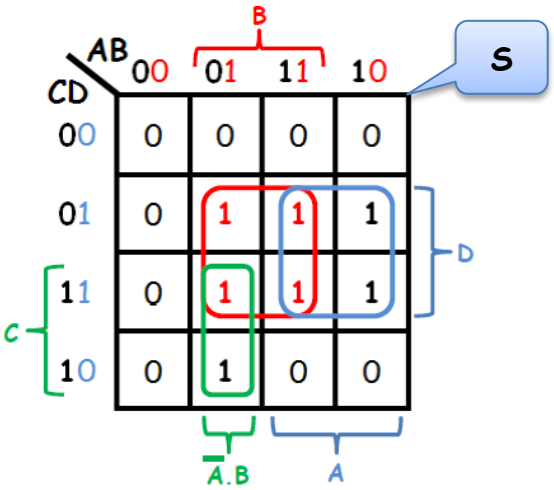
2.6.4.3 Simplification d'une fonction à 4 variables

A. Exemple 1 Simplifier la fonction de la TDV (Table de vérité) suivante en utilisant le tableau de KARNAUGH (TDK).

❖ Table de vérité

	A	B	C	D	S
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

❖ Tableau de KARNAUGH



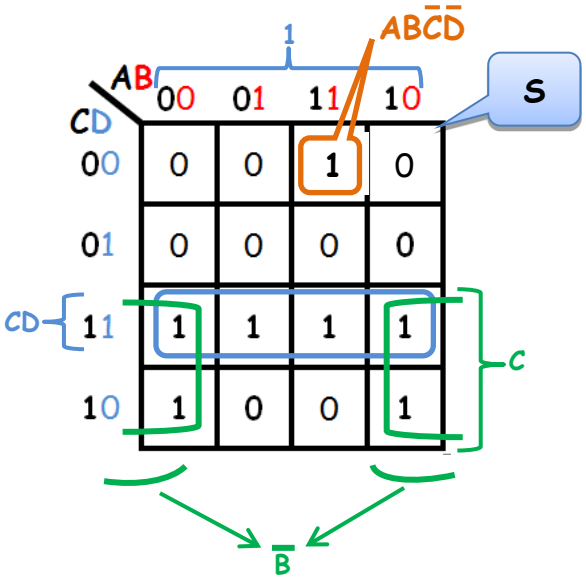
$S = A.D + B.D + \overline{A}.B.C$

B. Exemple 2 Simplifier la fonction de la TDV (Table de vérité) suivante en utilisant le tableau de KARNAUGH (TDK).

❖ Table de vérité

	A	B	C	D	S
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	1

❖ Tableau de KARNAUGH



$S = C.D + \overline{B}.C + A.B.\overline{C}.\overline{D}$

2.6.5 Fonctions incomplètement définies

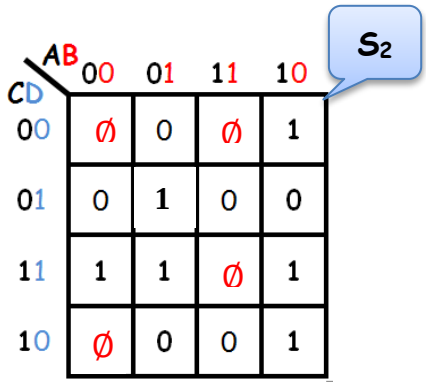
Parfois, la fonction à réaliser n'est pas complètement définie, c'est à dire que pour certaines combinaisons des variables, la valeur logique que prend la fonction n'est pas connue. Ceci pour plusieurs raisons:

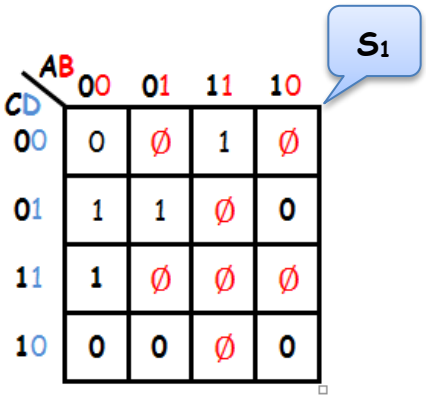
- ✓ Soit qu'on est indifférent à la valeur qu'elle peut prendre pour certaines combinaisons des variables.
- ✓ Soit que ces combinaisons correspondent, dans la pratique, à des configurations impossibles.

Les cases du tableau de KHARNAUGH correspondantes à ces combinaisons sont appelées des cases Ø .

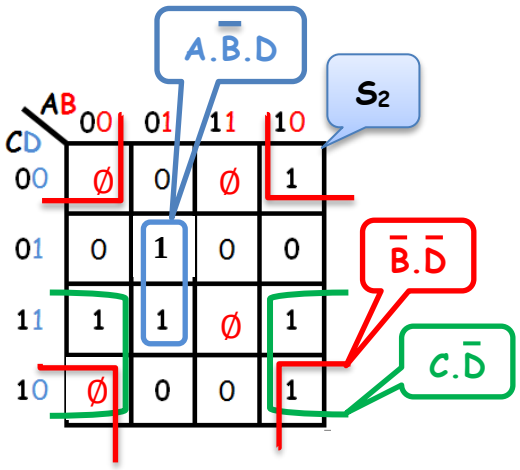
Pour trouver l'expression la plus simple de la fonction logique, on pourra utiliser ces casesØ, selon le cas soit comme des cases 1, soit comme des cases 0. Il n'y a pas de règle permettant un choix judicieux, mais en général on attribuera à chaque case Ø la valeur (0 ou 1) qui permet d'obtenir le plus petit nombre de contractions de plus grande taille possible qui englobe toutes les cases.

❖ **Exemple** Simplifier les fonctions suivantes :

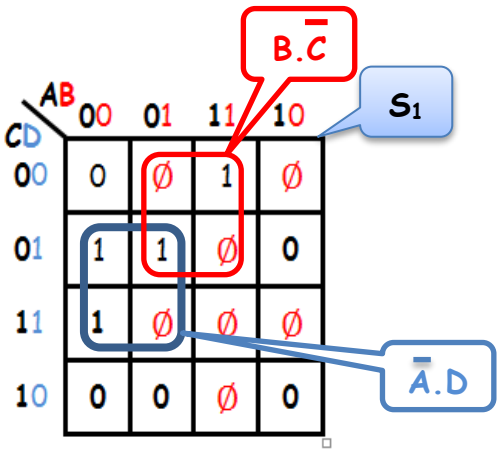




❖ **Solution**



$S_1 = \bar{A}.B.D + \bar{B}.\bar{D} + C.\bar{D}$



$S_1 = \bar{A}.D + \bar{B}.C$

2.6.6 Exercice 13

Simplifier algébriquement la fonction suivante

$$F2 = (X + Y + Z)(\bar{X} + Y + Z) + XY + YZ$$

2.6.7 Exercice 14

Soit la fonction booléenne suivante :

$$F(A, B, C, D) = A\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}D + A\bar{B}CD + A\bar{B}\bar{C}D + \bar{A}BCD + \bar{A}BC\bar{D} + ABC\bar{D}$$

A. Simplifier F par la méthode des diagrammes de KARNAUGH, sachant que quatre combinaisons de variables sont impossibles : $AB\bar{C}D + ABCD + \bar{A}\bar{B}C\bar{D} + A\bar{B}C\bar{D}$

B. Donner le schéma logique ou logigramme de la fonction simplifiée

2.6.8 Exercice 15

Une serrure de sécurité s'ouvre en fonction de quatre clés A, B, C D. Le fonctionnement de la serrure est défini comme suite :

S (A, B, C, D)= 1 si au moins deux clés sont utilisées

S (A, B, C, D)= 0 sinon

Les clés A et D ne peuvent pas être utilisées en même temps

On remarque que si la clé A et D sont utilisées en même temps l'état du système n'est pas déterminé. Ces cas sont appelés cas impossibles ou interdites.

A. Trouver la table de vérité

B. Simplifier la fonction en utilisant tableau de ***KHARNAUGH***

2.7 Résumé

Dans ce chapitre le plus important à retenir :

- Une variable logique est une grandeur qui ne peut prendre que deux valeurs 0 ou 1, est une fonction logique est constituer d'une ou plusieurs variables logiques, ne pouvant prendre elle-même qu'une des deux valeurs 0 ou 1
- Les fonctions logiques sont conçues à partir d'un groupe d'opérateurs élémentaires appelés **« Portes logiques »** et qui sont **:** Porte NON, Porte OU, Porte ET, Porte NON-ET, Porte NON-OU, Porte OU-Exclusif, Porte NON-OU Exclusif
- La table de vérité (***TV***) est un tableau qui permet de connaître systématiquement les états que peut prendre une fonction logique pour toutes les combinaisons des variables d'entrés et que dans une table de vérité de **n** variables, nous avons $\mathbf{2^n}$combinaisons différentes
- Un circuit logique ou logigramme est la traduction de la fonction logique en un schéma électronique. Le principe consiste à remplacer chaque opérateur logique par la porte logique qui lui correspond
- En électronique, les portes logiques sont fabriquées et renfermées dans des **circuits intégrés**
- Avant de traduire de la fonction logique en un circuit logique, il faut d'abord minimiser le nombre de portes logiques afin d'obtenir une réalisation matérielle plus simple donc plus facile à construire et à dépanner et moins coûteuse, pour cela il existe 3 méthodes de simplification :
  - ✓ La méthode algébrique (Elle devient difficile au de la de 3 variables)
  - ✓ La méthode graphique (diagramme de ***KARNAUGH***). Elle devient difficile au de la de 5 variables
  - ✓ La méthode de ***QUINE MAC CLUSKEY*** (Qui n'est pas aborder dans ce document)

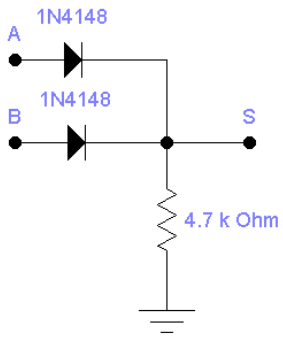
2.8 Exercice de synthèse N°2

- **Spécialité** : Réseaux Télécom Filaires
- **Module** : Electronique numérique
- **Durée :** 1h30
- **But :** Est d'étudier les propriétés des différents opérateurs logiques
- **Matériel requis** :
  1. Plaque d'essaie
  2. Fils de connections
  3. LEDs
  4. Diods 1N4148
  5. Résistance de 4,7KΩ
  6. Alimentation stabilisé
  7. Circuits intégrés (74LS32/74LS08/74LS00/74LS04/74LS02)
  8. Logiciel de simulation électronique
  9. Support de cours
- **Mise en situation :** Câbler différents circuits logique et vérifier leurs fonctionnements d
- **Marche à suivre :**

A. Portes à diodes

A.1. Premier montage

1. Câbler les composants (utiliser pour cela 2 Diodes 1N4148 et une Résistance de 4KΩ)



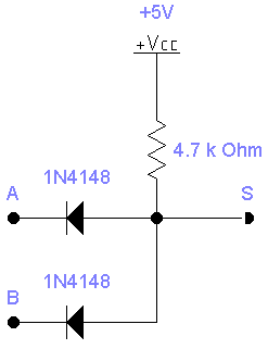
E1	E2	S
0	0	
0	1	
1	0	
1	1	

2. Compléter la table de vérité

3. Donner le nom de la fonction obtenue.

A.2. Deuxième montage :

1. Câbler les composants (utiliser pour cela 2 Diods 1N4148 et une Résistance de 4KΩ)



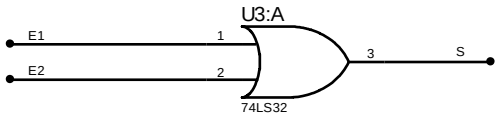
E1	E2	S
0	0	
0	1	
1	0	
1	1	

2. Compléter la table de vérité
3. Donner le nom de la fonction obtenue

B. Operateurs logiques

B.1. Premier montage

1. Câbler le composant (utiliser pour cela le CI 74LS32), voir brochage en annexe 2

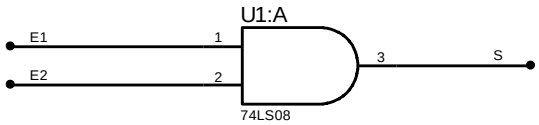


2. Compléter la table de vérité
3. Donner l'équation de S
4. Donner le nom de la fonction obtenue.

E1	E2	S
0	0	
0	1	
1	0	
1	1	

B.2. Deuxième montage

1. Câbler le composant (utiliser pour cela le CI 74LS08),voir brochage en annexe 2



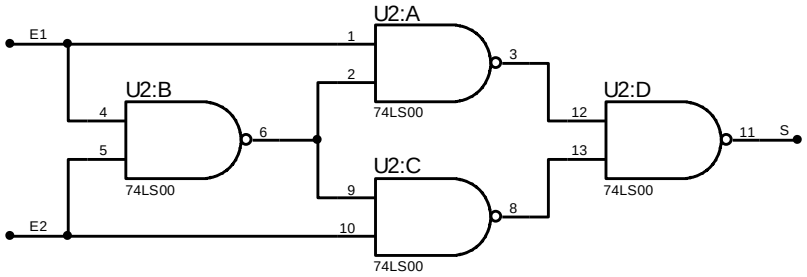
2. Compléter la table de vérité
3. Donner le nom de la fonction obtenue.
4. Donner l'équation de S

E1	E2	S
0	0	
0	1	
1	0	
1	1	

C. Fonctions logiques en logique câblée

C.1. Premier montage

1. Câbler les composants (utiliser pour cela le CI 74LS00), voir brochage en annexe



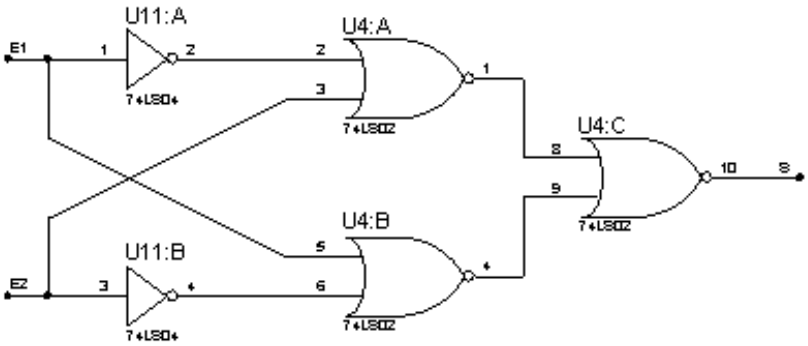
E1	E2	S
0	0	
0	1	
1	0	
1	1	

2. Compléter la table de vérité

3. Donner le nom de la fonction obtenue.

C.2. Deuxième montage

1. Câbler les composants (utiliser pour cela le CI 74LS04 et le CI 74LS02),voir brochage en annexe 2



2. Compléter la table de vérité

3. Donner le nom de la fonction obtenue.

E1	E2	S
0	0	
0	1	
1	0	
1	1	

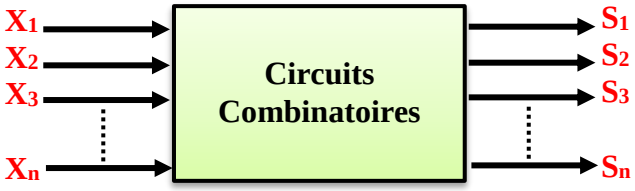
C.3. Troisième montage :$F= (A+\bar{C}).(\overline{C.B})$

1. Réaliser la fonction F , utiliser pour cela des CI que vous allez choisir vous-même , (voir brochage en annexe 2).
2. Déduire la table de vérité
3. Simplifier algébriquement la fonction F.
4. Réaliser F après simplification, utiliser pour cela des CI que vous allez choisir vous-même, (voir brochage en annexe 2).
5. Déduire la table de vérité
6. Qu'elle est votre conclusion ?

CHAPITRE 3 CIRCUITS COMBINATOIRES

La logique combinatoire concerne l'étude des fonctions dont la valeur de sortie ne dépend que de l'état logique des entrées se traduisant par une modification de la valeur des sorties et non pas non plus de ses états antérieurs : à chaque combinaison des variables d'entrée correspond toujours une seule combinaison des fonctions de sortie. Un circuit numérique réalisant une fonction combinatoire est un **circuit combinatoire,** voir figure ci-dessous.

Figure41: Circuits combinatoires



Outre les opérateurs élémentaires cités au chapitre 2, on distingue comme opérateurs combinatoires standard :

- ✓ Les opérateurs de transcodage,
- ✓ Les opérateurs d'aiguillage,
- ✓ Les opérateurs de comparaison,
- ✓ Les opérateurs arithmétiques.

3.1 Circuits de transcodage

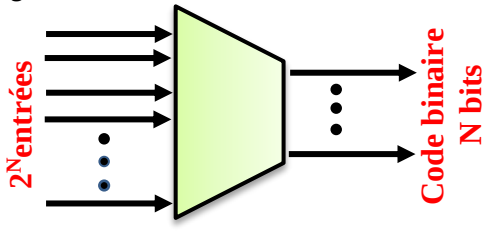
On désigne par circuit de **transcodage** un circuit qui traduit une information dans un code donné (**codeur** ou **encodeur**) ou bien qui, au contraire, permet d'extraire une ou des informations à partir d'un code donné (**décodeur**) ou bien encore réalise la conversion d'un code vers un autre (**convertisseur de code** ou **transcodeur**).

3.1.1 Codeur

Le codeur (ou encodeur) est un circuit logique qui possède $\mathbf{2^N}$ voies entrées, dont une seule est activée et N voies de sorties , avec $\mathbf{2^{N-1}<N\leq 2^N}$,il fournit en sortie le code binaire correspondant, voir (figure 39).

Le principe de fonctionnement d'un codeur est le suivant : lorsqu'une entrée est activée, les sorties affichent la valeur correspondant au numéro de l'entrée dans le code binaire choisi. Une seule entrée du codeur doit normalement être activée à la fois. Dans le cas où le code en sortie est le code binaire pur, voir exemple ci-dessous.

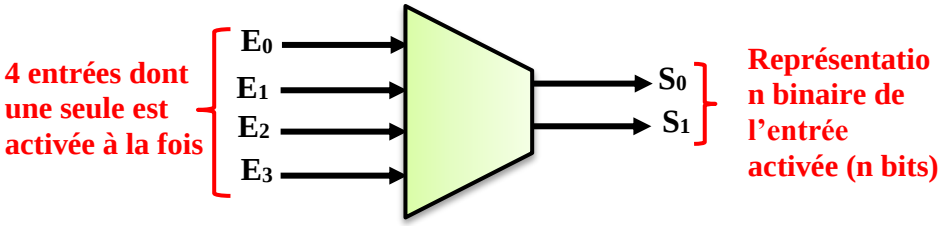
Figure42: Schéma fonctionnel d'un codeur



❖ Exemple d'un Codeur décimal vers binaire (4 entrées vers 2 sorties)

A. Schéma fonctionnel

Figure43: Schéma fonctionnel du codeur 4 voies d'entrées et 2 bits de sortie



B. Table de vérité

Entrées				Sorties	
Codage 1 parmi 2^N				Nombre binaire de N bits	
E_3	E_2	E_1	E_0	S_1	S_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

C. Table de vérité réduite du codeur 4 vers 2

Entrée Activée (=1)	Sorties S_1	Sorties S_0
E_0	0	0
E_1	0	1
E_2	1	0
E_3	1	1

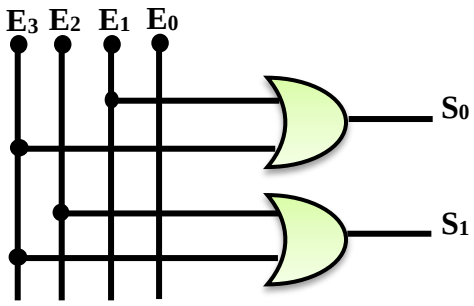
D. Equation des sorties

S_1=1 si (E_2=1) ou (E_3=1) ; $\mathbf{S_1=E_2+E_3}$

S_0=1 si (E_1=1) ou (E_3=1) ; $\mathbf{S_0=E_1+E_3}$

E. Logigramme

Figure44: Logigramme du codeur 4 vers 2

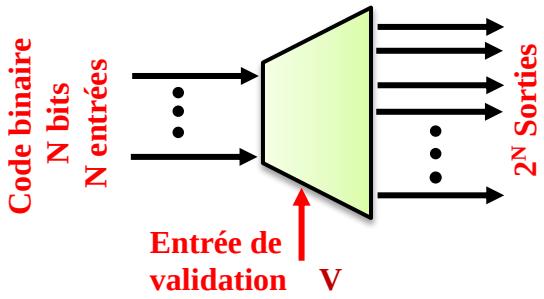


3.1.2 Décodeur C'est un circuit combinatoire qui est constitué de :

- ✓ **N** : entrées de données
- ✓ $\mathbf{2^N}$ sorties
- ✓ Pour chaque combinaison en entrée une seule sortie est active à la fois
- ✓ Une entrée de validation *V* (si l'entrée de validation *V* vaut 1, alors le décodage est autorisé ; dans le cas contraire (*V* = 0), les sorties sont inhibées et restent inactives), voir (figure 42).

Un décodeur est un circuit à N entrées et 2^N sorties dont une seule est active à la fois. Il détecte la présence d'une combinaison spécifique de bits (code) à ces entrées et l'indique par un niveau spécifique de sortie, voir exemple ci-dessous.

Figure 45:Schéma fonctionnel d'un décodeur

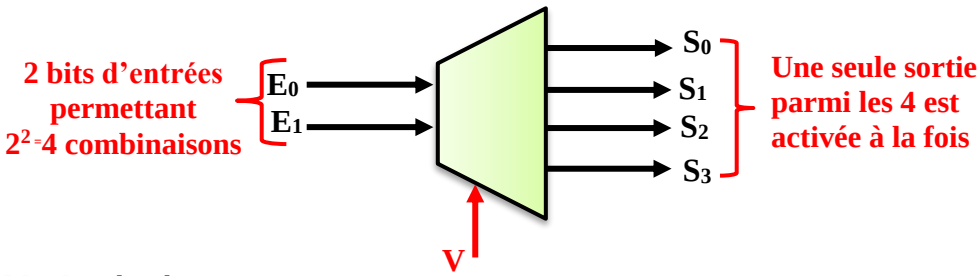


❖ **Exemple de décodeur 1 parmi 4**

Pour pouvoir activer toutes les 4 voies on à besoin de 2 bits à l'entrée car $2^2 = 4$ $\Longrightarrow$2 entrées et $2^2 = 4$ sorties (1 sortie parmi les 4 sera active) et une entrée de validation V

A. Schéma fonctionnel

Figure 46: Schéma fonctionnel d'un décodeur 1 parmi 4



B. Table de vérité

Code binaire d'entrée			Codage 1 parmi 4 sorties			
V	E_1	E_0	S_3	S_2	S_1	S_0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

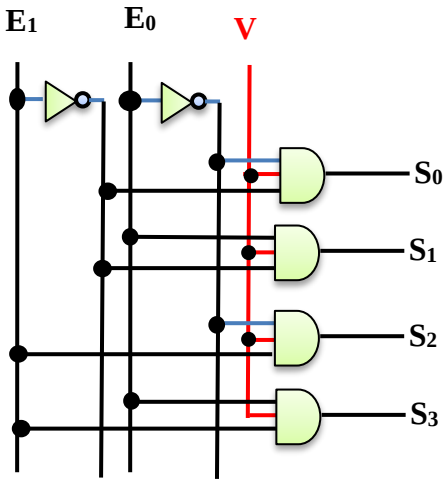
C. Equations de sorties

$S_0 = V.(\bar{E}_0.\bar{E}_1.)$, $S_1 = V(.E_0.\bar{E}_1)$,

$S_2 = V.(\bar{E}_0.E_1)$, $S_3 = V\ (E_1.E_0)$

D. Logigramme

Figure 47: Logigramme du décodeur 1 parmi 4



3.1.3 Convertisseurs

C'est un circuit combinatoire qui permet de transformer un code X (sur **n** bit) en entrée en un code Y (sur **m** bit) en sortie, voir (figure 45).

Figure48: Schéma fonctionnel d'un convertisseur

❖ **Exemple d'un transcodeur binaire Gray 2bits**

Pour passer d'un code à un autre, on utilisera un convertisseur de code. A titre d'illustration nous allons étudier le transcodeur binaire Gray.

A. Schéma fonctionnel

Figure49: Schéma fonctionnel d'un convertisseur binaire Gray 2bits

B. Table de vérité

Entrée Code binaire		Sortie Code gray	
B_1	B_0	G_1	G_0
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

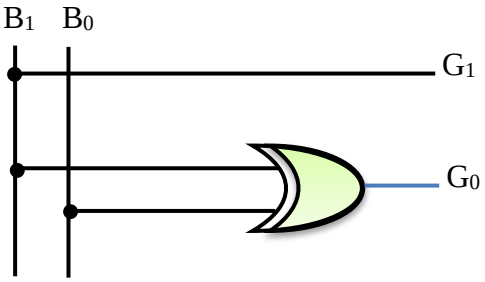
C. Equations de sorties

$$G_0 = \bar{B}_0 B_1 + B_0 \bar{B}_1 = B_0 \oplus B_1$$

$$G_0 = \bar{B}_0 B_1 + B_0 B_1 = B_1$$

D. Logigramme

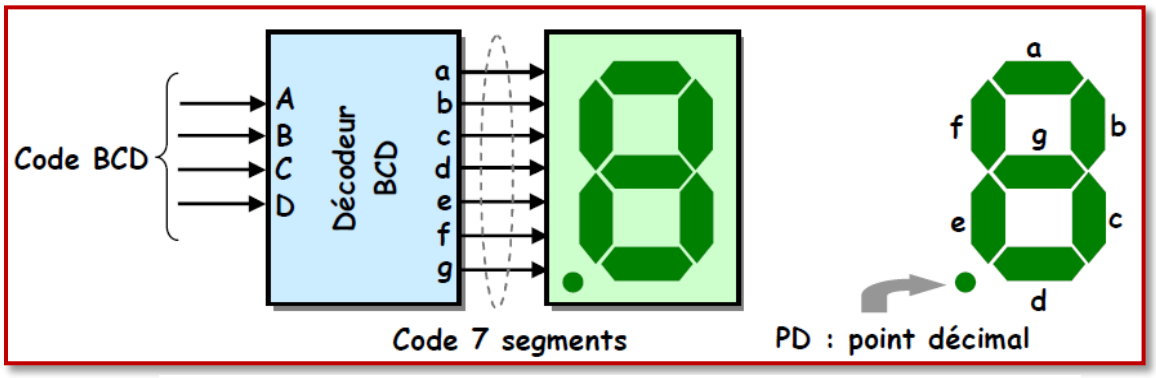
Figure 50 : Logigramme d'un transcodeur binaire Gray 2bits



3.1.4 Décodeur et afficheur 7 segments

Ce décodeur permet de convertir le code BCD, présent à son entrée sous 4 bits, en un code 7 segments disponible à sa sortie. Il est utilisé pour commander un afficheur 7 segments afin d'écrire des chiffres de 0 à 9, certaines lettres et aussi quelques symboles voir figure ci-dessous.

Figure 51 : Décodeur afficheur 7 segments

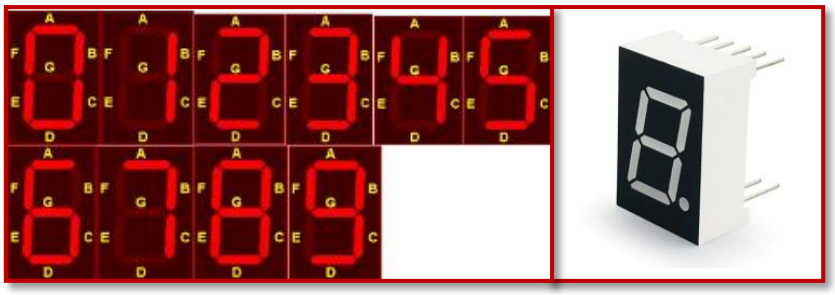


Source: Logique combinatoire : Partie 2 . M.SALMANI. Sciences et technologies électriques Niveau 1ère Sciences de l'ingénieur Unité ATC

A. Afficheur 7 segments

Un afficheur 7 segments est un circuit intégré qui contient 7 diodes (**a, b, c, d, e, f, g**) électroluminescents (LED) sous forme de segments et une LED du point décimal, voir (figure 52).

Figure 52: Afficheur 7 segments



Source: https://www.memoireonline.com

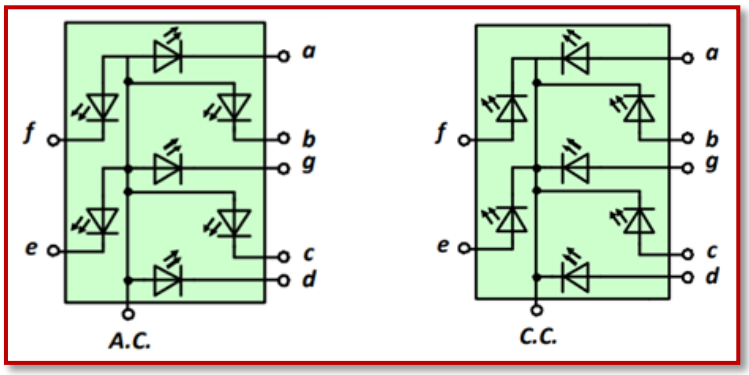
Chaque segment est constitué par une diode électroluminescente (LED) qui émet de la lumière lorsqu'elle est traversée par un courant d'intensité suffisante. Pour afficher les différents chiffres il faut allumer certains segments et éteindre les autres. Par exemple pour afficher le chiffre "6", il faut que les segments c, d, e, f et g soient éclairés et que les segments a et b soient éteints.

On distingue 2 types d'afficheurs : voir (figure 53).

- Afficheurs à **anode commune** voir (figure 50a), pour cela :
  - ✓ L'anode commune est portée au potentiel **+V_{CC}**.
  - ✓ Pour allumer une LED, on applique un potentiel **0 volt** à sa cathode (entrée).
- Afficheurs à **cathode commune** voir (figure 50b), pour cela :
  - ✓ La cathode commune est portée au potentiel **0 volt**.
  - ✓ Pour allumer une LED, on applique un potentiel **+V_{CC}** à son anode (entrée).

Figure 53: Afficheur 7 segments

(a) (b)

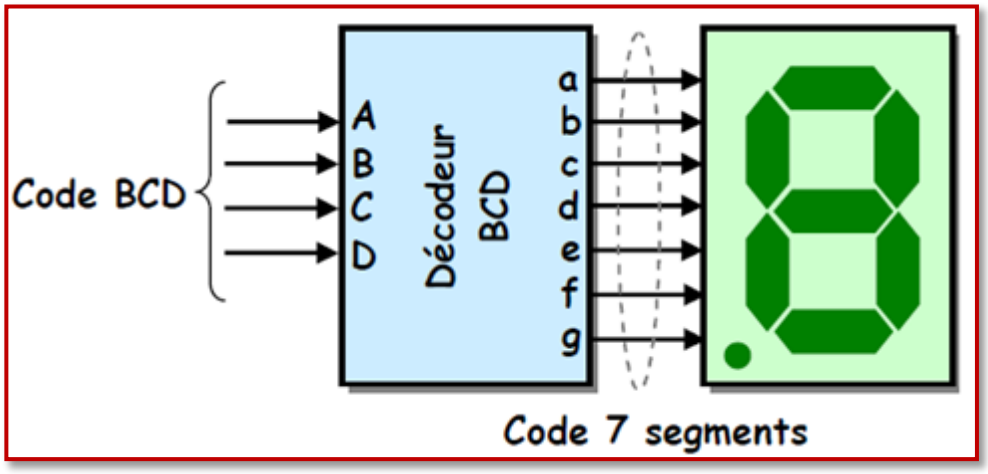


Source: Logique combinatoire : Partie 2 . M.SALMANI. Sciences et technologies électriques Niveau 1ère Sciences de l'ingénieur Unité ATC

B. .Décodeur BCD 7 segments

Le décodeur 7 segments accepte en entrée les 4 bits DCB (a0, a1, a2, a3) et rend actives les sorties qui vont permettre de faire passer un courant dans les segments d'un afficheur numérique pour former les chiffres décimaux (de 0 à 9) Voir figure ci-dessous.

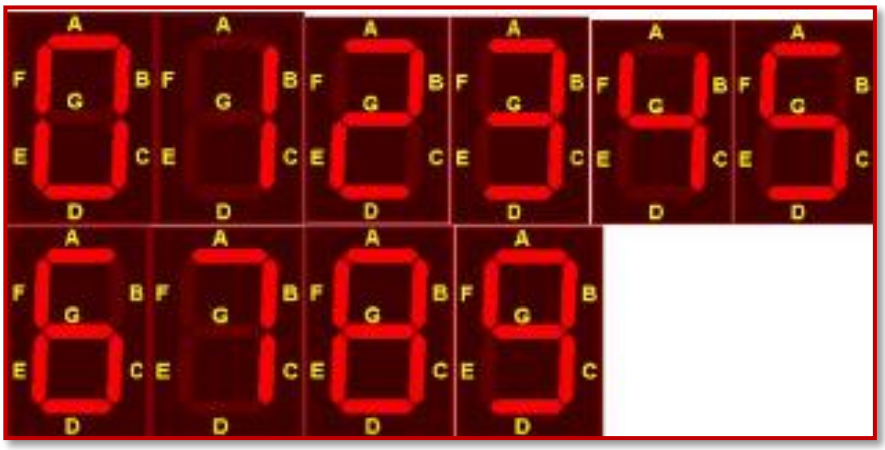
Figure 54: Décodeur BCD afficheur 7 segments



Source: Logique combinatoire : Partie 2 . M.SALMANI. Sciences et technologies électriques Niveau 1ère Sciences de l'ingénieur Unité ATC

Remarque : Il y'a 6 combinaisons intitulés 10, 11, 12, 13, 14, 15 que l'on notera Ø. Les autres chiffres sont affichés comme suit : voir (figure 55)

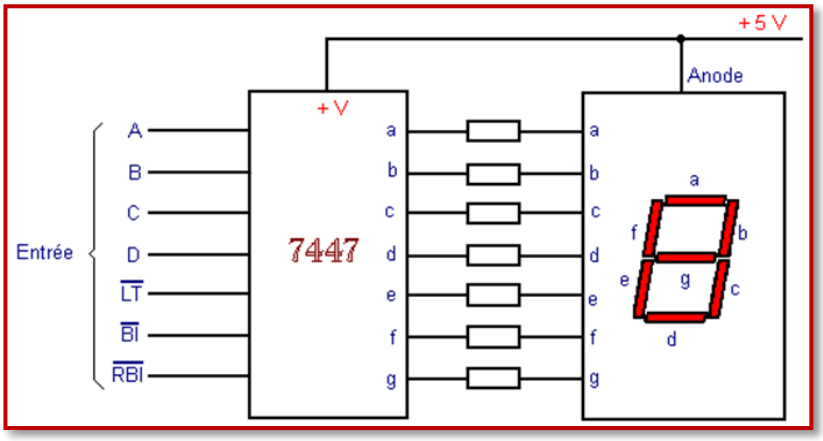
***Figure 55 : Les Chiffres* afficher** par un afficheur 7 segments



Source: https://www.memoireonline.com

C. Décodeur 7447 commandant un afficheur 7 segments à anode commune : voir (figure 56)

Figure 56 : Afficheur 7 segments commander par un décodeur 7447



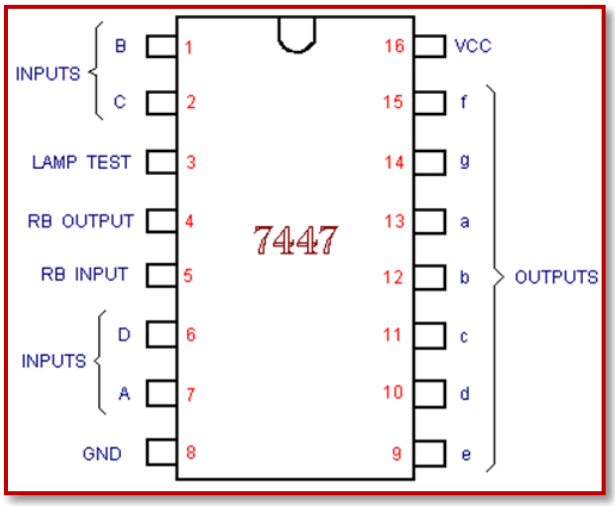
Source: https://www.electronique-et-informatique.fr

Ce décodeur dispose de sorties à collecteurs ouverts autorisant une liaison directe avec des afficheurs à anode commune. Des entrées supplémentaires sont aussi prévues :

- ✓ **LT ou «lamp test»** qui permet de vérifier le fonctionnement de l'afficheur en allumant tous les segments si BI est à l'état 1.
- ✓ **BI / RBO ou «blanking output»** qui permet l'effacement des segments de l'afficheur quel que soit l'état des autres entrées.
- ✓ **RBI ou «rippleblanking input»** qui permet l'effacement des 0 à gauche si A, B, C, D sont à 0.

❖ Brochage du décodeur 7447 : Voir figure ci-dessous.

Figure 57 : Afficheur 7 segments commander par un décodeur 7447



Source: https://www.electronique-et-informatique.fr

❖ La table de vérité du décodeur 7447

	Entrées							Sorties						
Décimal ou Fonction	LT	RBI	D	C	B	A	BI/RBO	a	b	c	d	e	f	g
0	1	1	0	0	0	0	1	0	0	0	0	0	0	1
1	1	X	0	0	0	1	1	1	0	0	1	1	1	1
2	1	X	0	0	1	0	1	0	0	1	0	0	1	0
3	1	X	0	0	1	1	1	0	0	0	0	1	1	0
4	1	X	0	1	0	0	1	1	0	0	1	1	0	0
5	1	X	0	1	0	1	1	0	1	0	0	1	0	0
6	1	X	0	1	1	0	1	1	1	0	0	0	0	0
7	1	X	0	1	1	1	1	0	0	0	1	1	1	1
8	1	X	1	0	0	0	1	0	0	0	0	0	0	0
9	1	X	1	0	0	1	1	0	0	0	1	1	0	0
10	1	X	1	0	1	0	1	1	1	1	0	0	1	0
11	1	X	1	0	1	1	1	1	1	0	0	1	1	0
12	1	X	1	1	0	0	1	1	0	1	1	1	0	0
13	1	X	1	1	0	1	1	0	1	1	0	1	0	0
14	1	X	1	1	1	0	1	1	1	1	0	0	0	0
15	1	X	1	1	1	1	1	1	1	1	1	1	1	1
BI	X	X	X	X	X	X	0	1	1	1	1	1	1	1
RBI	1	0	0	0	0	0	0	1	1	1	1	1	1	1
LT	0	X	X	X	X	X	1	0	0	0	0	0	0	0

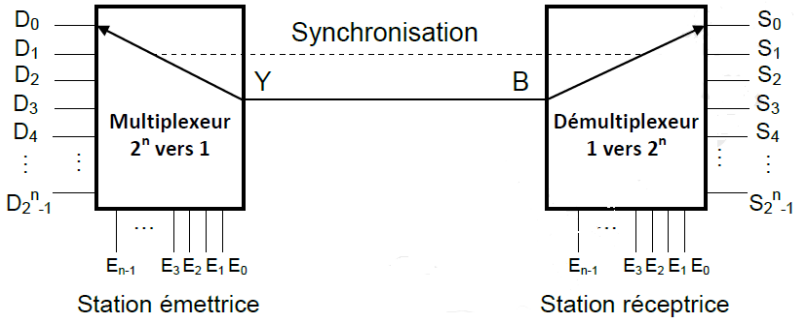
3.1.5 Exercice 16 Réaliser un convertisseur 4bits Binaire Naturel-Binaire Réfléchi

3.2 Circuits de multiplexage

La transmission des informations d'une station à une autre nécessite plusieurs lignes en parallèle, ce qui est difficile à réaliser et très couteux lorsque les stations sont géométriquement éloignées l'une de l'autre.

La solution est alors, transmettre en série sur une seule ligne, en utilisant à la station émettrice un convertisseur parallèle/série (Multiplexeur) et à la station réceptrice un convertisseur série/parallèle (Démultiplexeur), voir figure ci-dessous.

Figure58: Station de transmission des informations



Source : *Support de cours : systèmes Logiques (1) Logique combinatoire Institut Supérieur des Etudes Technologiques de Nabeul Département de Génie Electrique)*

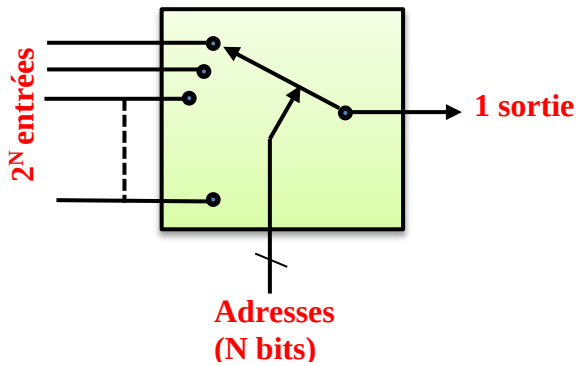
3.2.1 Multiplexeur (MUX)

Un multiplexeur est un circuit servant à concentrer sur une même voie de transmission différents types de liaisons (***informatique, télécopie, téléphonie, télétex***) en sélectionnant une entrée parmi N, c'est un circuit combinatoire qui permet de sélectionner une information (1 bit) parmi 2^N valeurs en entrée, voir (figure 59).

Il possède :

- 2^N entrées d'information
- Une seule sortie
- N entrées de sélection (commandes ou adresses)

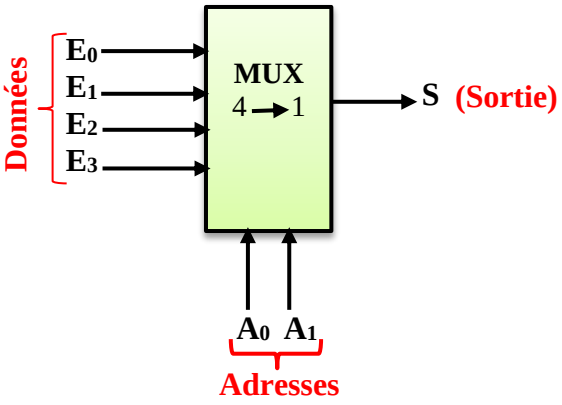
Figure59: Schéma fonctionnel d'un multiplexeur 2^N vers 1



A. Exemple d'un multiplexeur 4 vers 1 : voir (figure 60)

❖ **Schéma fonctionnel**

Figure60: Schéma fonctionnel d'un multiplexeur 4 vers 1



❖ **Table de vérité**

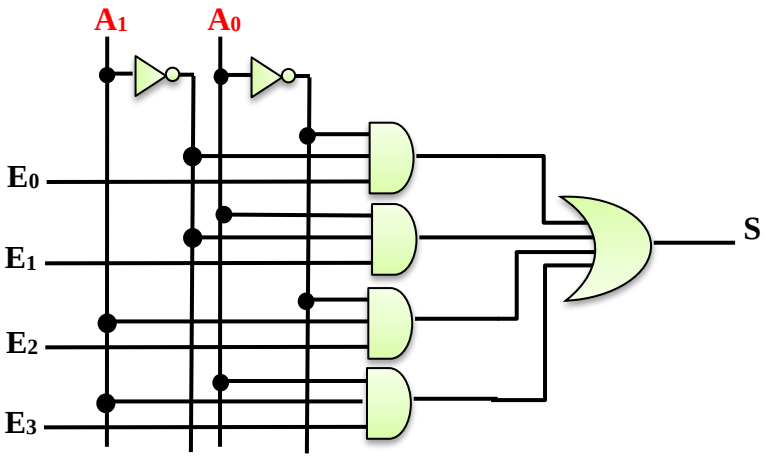
A_1	A_0	S
0	0	E_0
0	1	E_1
1	0	E_2
1	1	E_3

❖ **Equation de sortie**

$$S= \bar{A}_0\bar{A}_1(E_0) + A_0\bar{A}_1(E_1) + \bar{A}_0A_1(E_2) + A_0A_1(E_3)$$

❖ **Logigramme**

Figure61: Logigramme d'un multiplexeur 4 vers 1

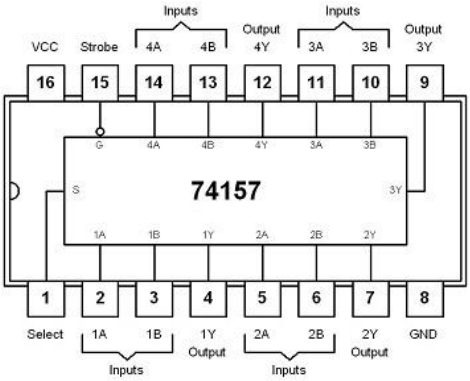


B. Exemples de circuits integres multiplexeurs : On trouve chez les constructeurs, les circuits multiplexeurs suivants :

- ✓ 2 vers 1 : 74157 (4 Mux 2 vers 1)
- ✓ 4 vers 1 : 74153 (2 Mux 4 vers 1)
- ✓ 8 vers 1 : 74151 (2 Sorties complémentaires), 74152 (1 Sortie complémentée)
- ✓ 16 vers 1 : 74150 (1 Sortie complémentée)

C. Brochage du 74LS157 : (4Mux 2 vers 1)

Figure62: Brochage du 74LS157



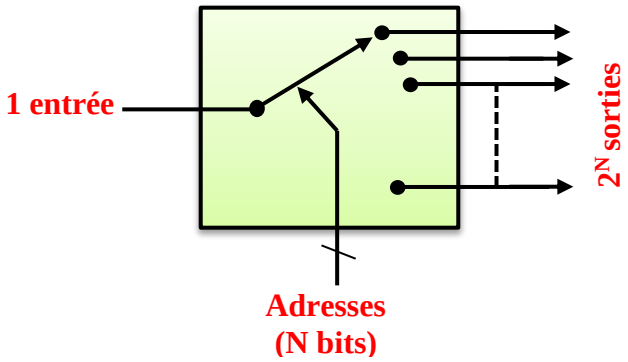
Source: https://www.electronique-et-informatique.fr/

3.2.2 Démultiplexeur (DEMUX)

Il joue le rôle inverse d'un multiplexeur, il permet de faire passer une information dans l'une des sorties selon les valeurs des entrées de commandes .Il possède :

- Une seule entrée
- 2^N sorties
- **N** entrées de sélection (commandes ou adresses), voir (figure 63).

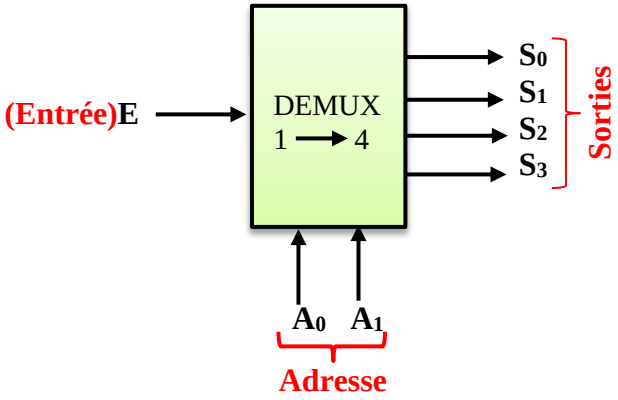
Figure63: Schéma fonctionnel d'un démultiplexeur 1 vers 2^N



A. Exemple d'un démultiplexeur 1 vers 4

❖ Schéma fonctionnel

Figure64: Schéma fonctionnel d'un démultiplexeur 1 vers 4



❖ Table de vérité

A_1	A_0	S_3	S_2	S_1	S_0
0	0	0	0	0	E
0	1	0	0	E	0
1	0	0	E	0	0
1	1	E	0	0	0

❖ Equations de sorties

$$S_0 = \overline{A_0}\,\overline{A_1}\,(E)$$

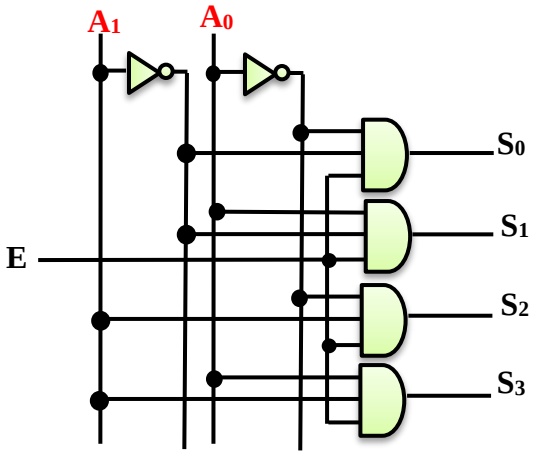
$$S_1 = A_0\,\overline{A_1}\,(E)$$

$$S_2 = \overline{A_0}\,A_1\,(E)$$

$$S_0 = A_0\,A_1\,(E)$$

❖ Logigramme

Figure 65: Logigramme d'un multiplexeur 4 vers 1

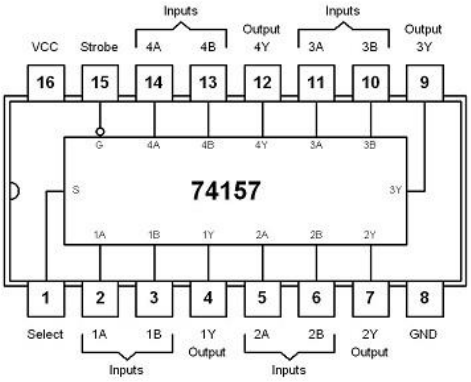


B. Exemples de circuits integres démultiplexeur : On trouve chez les constructeurs, des circuits Démultiplexeurs :

- ✓ 1 vers 4 : 74139 (2 DEMUX 1 vers 4, Sorties complémentées)
- ✓ 1 vers 8 : 74137, 74138 (Sorties complémentées)
- ✓ 1 vers 16 : 74154,74159 (Sorties complémentées)

C. Brochage du 74LS139 : (2 Demux 1 vers 4)

Figure 66: Brochage du 74LS139



Source: https://www.electronique-et-informatique.fr

3.2.3 Exercice 17

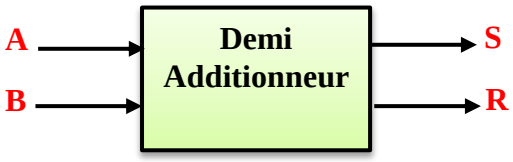
A. Donner le synoptique et la TDV d'un MUX 2 vers 1
B. Donner le synoptique et la TDV d'un MUX 8 vers 1
C. Réaliser le MUX 8 vers 1 a l'aide de MUX 2 vers 1 (Synoptique)

3.3 Circuits de calcul

3.3.1 Demi-Additionneur

Le demi additionneur est un circuit combinatoire qui permet de réaliser la somme arithmétique de deux nombres **A** et **B** sur un bit, a la sotie on va avoir la somme **S** et la retenu **R** (***Carry***), voir figure ci-dessous.

Figure67 : Schéma fonctionnel d'un demi-additionneur



A. Table de vérité

En binaire l'addition sur un seul bit se fait de la manière suivante

0 + 0 = 0 retenue 0
0 + 1 = 1 retenue 0
1 + 0 = 1 retenue 0
1 + 1 = 0 retenue 1

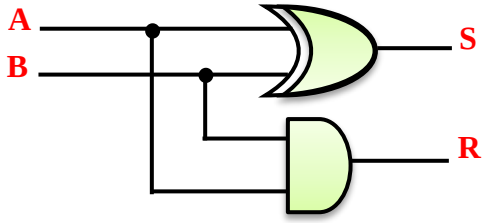
A	B	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

B. Equations de sorties

$S = \overline{A}B + A\overline{B} = A \oplus B$
$R = A\ B$

C. Logigramme

Figure 68: Logigramme d'un demi-additionneur 1 bit



3.3.2 Additionneur complet 1 bit

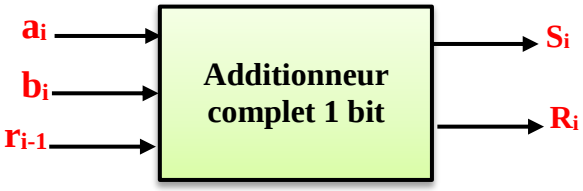
L'additionneur complet un bit comme le montre la (figure 69) possède 3 entrées :

- a_i : le premier nombre sur un bit.
- **bi** : le deuxième nombre sur un bit.
- r_{i-1} : le retenue entrante sur un bit.

Il possède deux sorties :

- **Si** : la somme
- **Ri** la retenue sortante

Figure69: Schéma fonctionnel d'un additionneur complet 1 bit



A. Table de vérité

a_i	b_i	r_{i-1}	S_i	R_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

B. Equations de sorties

$$S_i = \bar{a}_i \bar{b}_i r_{i-1} + \bar{a}_i b_i \bar{r}_{i-1} + a_i \bar{b}_i \bar{r}_{i-1} + a_i b_i r_{i-1}$$

$$S_i = \bar{a}_i (\bar{b}_i r_{i-1} + b_i \bar{r}_{i-1}) + a_i (\bar{b}_i \bar{r}_{i-1} + b_i r_{i-1})$$

$$S_i = \bar{a}_i (b_i \oplus r_{i-1}) + a_i \overline{(b_i \oplus r_{i-1})}$$

$$\mathbf{S_i = a_i \oplus b_i \oplus r_{i-1}}$$

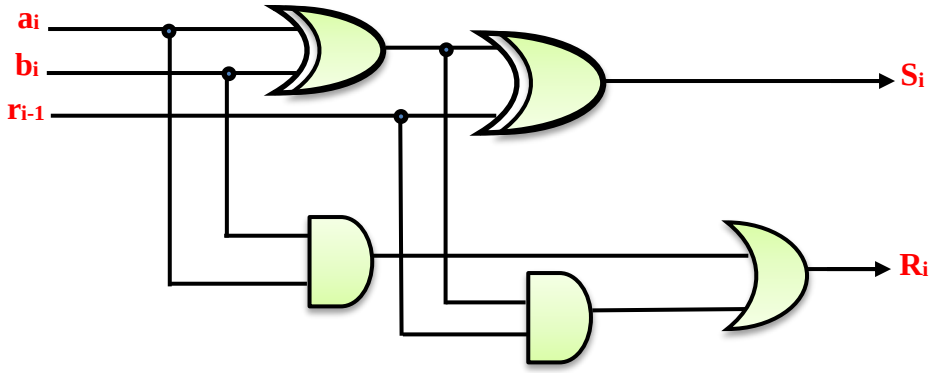
$$R_i = \bar{a}_i b_i r_{i-1} + a_i \bar{b}_i r_{i-1} + a_i b_i \bar{r}_{i-1} + a_i b_i r_{i-1}$$

$$R_i = r_{i-1} (\bar{a}_i b_i + a_i \bar{b}_i) + a_i b_i (\bar{r}_{i-1} + r_{i-1})$$

$$\mathbf{R_i = r_{i-1} (a_i \oplus b_i) + a_i b_i}$$

C. Logigramme

Figure 70: Logigramme d'un additionneur complet 1 bit



3.3.3 Comparateur 1 bit

C'est un circuit combinatoire qui permet de comparer entredeux nombres binaire A et B, voir (figure 71).

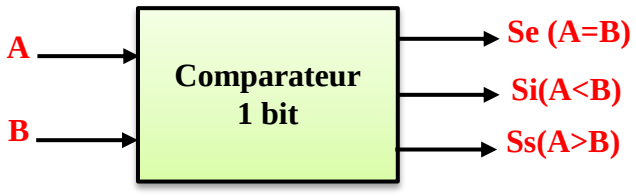
Il possède 2 entrées :

- **A** : sur un bit
- **B** : sur un bit

Il possède 3 sorties

- **Se** : égalité (A=B)
- **Si** : inférieur (A< B)
- **Ss** : supérieur (A > B)

Figure71: Schéma fonctionnel d'un comparateur 1 bit



A. Table de vérité

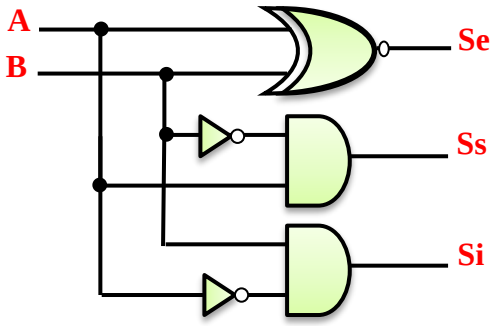
A	B	Ss	Se	Si
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

B. Equations de sorties

Ss= A.$\overline{B}$
Se= $\overline{A}\overline{B}$ + A B =$\overline{A \oplus B}$
Si= $\overline{A}$.B

C. Logigramme

Figure 72: Logigramme d'un comparateur 1 bit

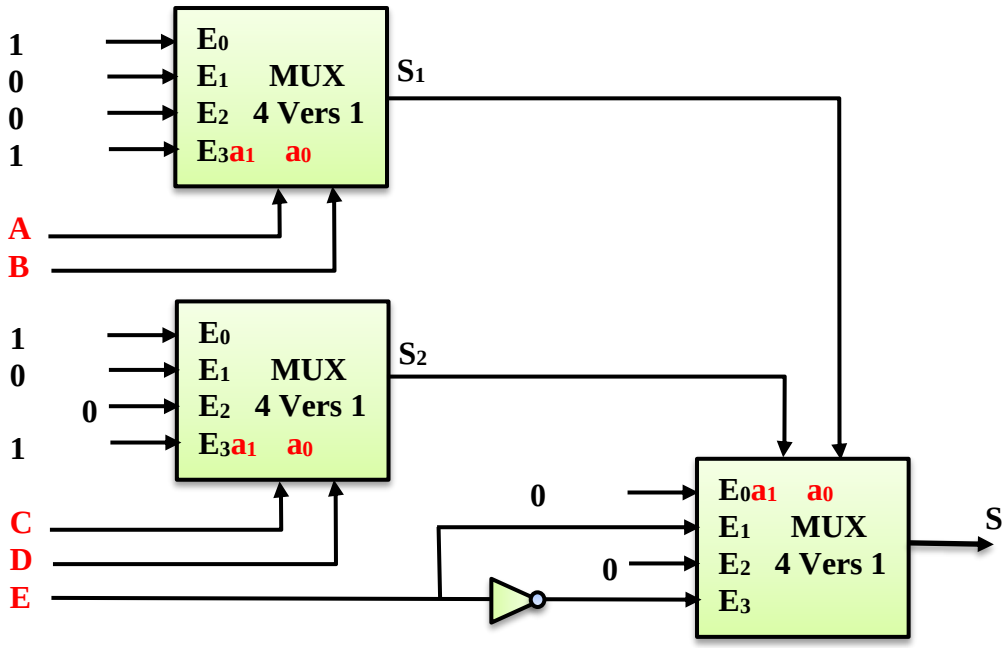


3.3.4 Exercice 18 Donner le schéma fonctionnel d'un additionneur complet 4 bits

3.3.5 Exercice 19 Donner le circuit logique d'un comparateur qui permet de faire la comparaison entre deux nombres A (A_1A_0) et B (B_1B_0) chacun sur deux bits

3.3.6 Exercice 20 Donner l'équation logique de la fonction S du montage suivant :

Figure 73: Montage de la fonction S



3.4 Résumé

On résume ce chapitre comme suite

- ✓ Dans un circuit combinatoire, chaque combinaison des variables d'entrée correspond toujours une seule combinaison des fonctions de sortie
- ✓ Un codeur est un circuit combinatoire qui possède $\mathbf{2^N}$ voies entrées, dont une seule est activée et **N** voies de sorties, avec $\mathbf{2^{N-1}<N\leq 2^N}$
- ✓ Un décodeur est un circuit combinatoire qui possède **N** entrées de données, $\mathbf{2^N}$ sorties, ou une seule sortie est active à la fois et une entrée de validation ***V*** (***V=0***le décodeur et bloqué, ***V=1***le décodage est autorisé)
- ✓ Un Convertisseurs est un circuit combinatoire qui permet de transformer un code **X** (sur **N** bit) en entrée en un code **Y** (sur **M** bit) en sortie
- ✓ Un multiplexeur est un circuit combinatoire qui permet de sélectionner une information (1 bit) parmi 2^N valeurs en entrée. Il possède 2^N entrées d'information, une seule sortie et N entrées d'adresses
- ✓ Un Démultiplexeur est l'inverse du multiplexer, il possède : Une seule entrée de donnée, $\mathbf{2^N}$ sorties et **N** entrées d'adresses
- ✓ Un additionneur complet un bit possède 3 entrées : $\mathbf{A_i}$: le premier nombre sur **N** bit, **Bi**le deuxième nombre sur **N** bit et $\mathbf{R_{i-1}}$ la retenue entrante sur un bit. Il possède aussi2 sorties : **Si** la somme et **Ri** la retenue sortante
- ✓ Un comparateur binaire est un circuit logique qui effectue la comparaison entre 2 nombres binaires généralement notés A et B. Il possède 3 sorties notées **A = B**,**A > B** et **A < B**

3.5 Exercice de synthèse 3

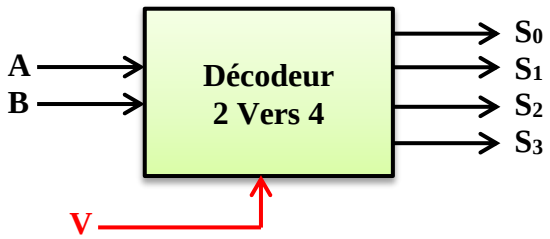
- **Spécialité :** Réseaux Télécom Filaires
- **Module :** Electronique numérique
- **Durée :** 2h
- **But :** Est de se familiariser avec les circuits combinatoires de décodages et d'en voir un exemple d'application.
- **Matériel requis** :
  10. Plaque d'essaie
  11. Fils de connections
  12. Afficheur 7 segments Anode commune
  13. Circuits intégrés décodeur 7 segments 74LS48
  14. Résistance de 470Ω
  15. Alimentation stabilisé
  16. Circuits intégrés 74LS04 et 74LS11
  17. Logiciel de simulation électronique
  18. Support de cours
- **Mise en situation :** Câbler différents circuits logique et vérifier leurs fonctionnements d
- **Marche à suivre :**

I. Décodeur 2vers 4

Un décodeur est un circuit qui possède **n** entrées et **2^n**sorties. Pour chacune des combinaisons possibles des entrées, seule une ligne de sortie est validée. Les décodeurs sont souvent dotes d'une ou plusieurs entrées de validation (EN: enable) qui servent à valider son fonctionnement.

1. Etablir les équations logiques des sorties S_0, S_1, S_2, S_3un décodeur 2 vers 4 de (figure 74) en fonction des variables d'entrée A et B .

Figure 74:Montage de la fonction S



2. Sur maquette, sur simulateur, ou avec un logiciel de simulation :

A. Câblez le circuit du décodeur à l'aide de portes logiques en utilisant les équations logiques de sortie S0, S1, S2, S3 de la première question. Utiliser les circuits intégré TTL du type 7411et 7404 (voir brochage annexe 2).

Figure 75 ; Circuits intégrer 74LS11 et 74LS04

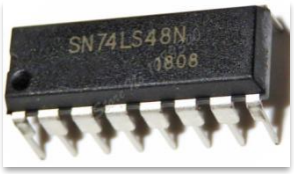
Source : https://www.aliexpress.com

B. Vérifier le fonctionnement à l'aide d'une table de vérité.

II. Afficheur 7 segments

On désire afficher, sur un afficheur 7 segments, les chiffres de **0** à **9** ainsi que les lettres de « **a** » à « **f** ». Nous allons donc utiliser pour cela le décodeur 7 segments 74LS48 (voir brochage annexe 2) qui va recevoir a son entrée un code binaire sur 4 bits (compris entre **0000** et **1111**), et fournissant en sortie 7 signaux qui permettront d'alimenter les segments de l'afficheur anode commune. Les entrées s'appellent **A, B, C** et **D**, A étant le bit de poids faible. Les sorties s'appelle **a, b, c, d, e, f, et g,** et alimentent respectivement les segments **a** à **g** de l'afficheur.

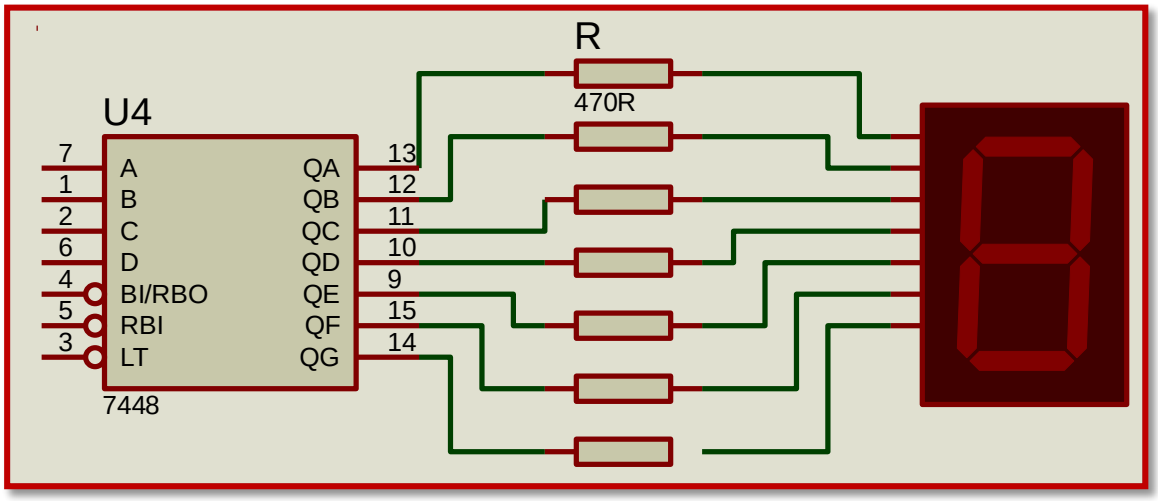
Figure 76 ; Circuit integrer 74LS48



Source : https://ar.aliexpress.com

A. Sur maquette, sur simulateur, ou avec un logiciel de simulation, câbler et tester le circuit suivant

Figure 77 : Circuit d'affichage 7 segments



B. Complétez la table de vérité ci-dessous du décodeur, puis recherchez, en utilisant les tableaux de Karnaugh, les équations simplifiées des 7 sorties du décodeur, en fonction des entrées A a D. Table de vérité du décodeur :

Entrées				Sorties de l'afficheur							Valeur décimale
D	C	B	A	a	b	c	d	e	f	g	
0	0	0	0								
0	0	0	1								
0	0	1	0								
0	0	1	1								
0	1	0	0								
0	1	0	1								
0	1	1	0								
0	1	1	1								
1	0	0	0								
1	0	0	1								

C. Equations simplifiées de chaque sortie (d'après les tableaux de Karnaugh) :

a=…… b=……

c=…… d=……

e=…… f=……

g=……

CHAPITRE 4 LES CIRCUITS SEQUENTIELS

Les circuits logiques combinatoires ne suffisent pas à eux seuls à la manipulation de l'information comme cela se fait dans les systèmes numériques modernes. Le caractère figé des circuits combinatoires que traduit la correspondance stricte entre les entrées et les sorties limite considérablement le champ de leurs applications. C'est là que les circuits séquentiels prennent toute leur importance.

Dans les systèmes numériques modernes (appareil de mesure, ordinateur…) La ***mémorisation*** des informations est une opération indispensable dans tout système de traitement, on veut on outre ***Compter*** (des impulsions), ***décaler*** d'un ou plusieurs pas (une suite de niveaux **1** et **0**),***recopier*** au rythme d'une horloge (des niveaux **1** et **0**), ***transmettre*** en série sur un fil des information, ***stockées*** (en parallèle)d'une façon permanente dans une mémoire, d'où la nécessite des circuits séquentiels.

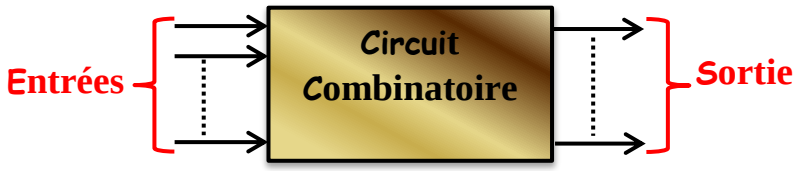
4.1 Circuits combinatoires et circuits séquentiels

4.1.1 Circuits combinatoires

Dans les opérateurs logiques examinés précédemment l'état de sortie ne dépend que de l'état présent des entrées, c'est à dire que pour chaque état d'une combinaison d'entrée correspond **une valeur toujours la même** à la sortie (Circuits combinatoires).

En logique combinatoire, à chaque combinaison d'entrée correspond une seule possibilité de combinaison sur les sorties, voir (figure 78).

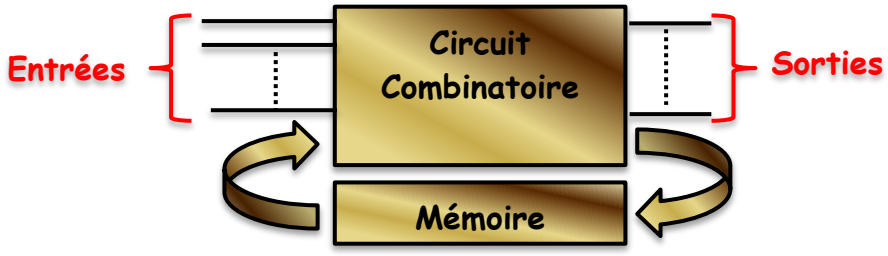
Figure78: Circuit combinatoire



4.1.2 Circuits séquentiels

Pour certains opérateurs, l'état de la sortie dépend non seulement de la combinaison appliquée à l'entrée (logique combinatoire) mais aussi de l'état ***précédent*** des sorties du circuit : ils sont dits séquentiels et ont un effet « mémoire ». La logique séquentielle est donc une logique combinatoire avec une mémorisation des sorties. Cette mémorisation est réalisée par ce qu'on appelle une bascule ; c'est un organe de mémorisation unitaire (mémorisation d'un seul bit), voir (figure 79) .

Figure79: Circuit séquentiel

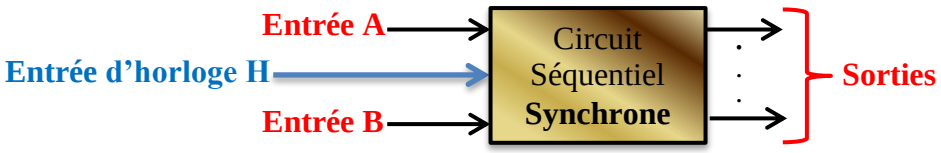


4.1.3 Fonctionnement des circuits séquentiels

Nous aurons deux grands types de fonctionnement, soit la valeur des entrées est significative à tout instant (lecture de l'entrée à toute instant), on parle alors de circuits asynchrones, voir (figure 80) , soit elle n'est prise en compte qu'à des instants précis synchronisés par une fréquence d'horloge, on parle alors de circuits synchrones, voir (figure 81).

Figure 80: Circuit séquentiel **Asynchrone**

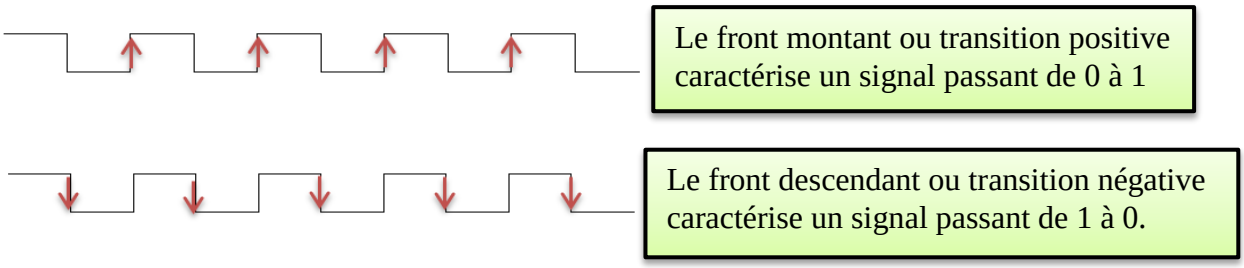
Figure 81: Circuit séquentiel **Synchrone**



Le signal d'horloge est généralement un signal carré, il est habituellement distribué à tous les étages du système, de sorte que la plupart des sorties changent d'état seulement quand le signal d'horloge effectue une transition.

Ces transitions, appelées fronts, sont identifiées sur un chronogramme au moyen d'une flèche dans le sens de variation du signal voir (figure 82).

Figure 82: front montant et front descendant d'un signal d'horloge



4.1.4 Exercice 21 Répondre par vrai ou faux

- Un signal d'horloge est généralement un signal sinusoïdal
- Un front descendant caractérise un signal passant de 0 à 1
- l'état de la sortie d'un circuit séquentiel dépend non seulement de la combinaison appliquée à ces l'entrées mais aussi de l'état précédent de ces sorties
- La logique séquentielle est une logique combinatoire avec une mémorisation des sorties
- Un circuit est dit synchrone si la lecture de ces entrées se fait à tout instant
- Dans un circuit l'état de sortie ne dépend que de l'état présent des entrées
- Une bascule est un organe de mémorisation unitaire

4.2 Bascules

Ce sont des circuits ayant deux états stables et dont la sortie ne dépend pas seulement de l'état des entrées à un instant donnée mais aussi de l'état antérieur, ils sont donc pas combinatoires mais séquentiels.

Les bascules sont des mémoires élémentaires qui ne peuvent mémoriser qu'un seul bit.

Exemple d'utilisation :

- Appel d'un ascenseur ; on appuie sur le bouton, l'appel est enregistré et le voyant s'allume. Si on relâche le bouton, le voyant reste allumé, il y a donc mémorisation.
- Le piéton qui appuie sur le bouton des feux tricolores pour traverser la route : l'information est mémorisée même si le piéton relâche le bouton.
- La commande « coupure du son » du téléviseur, etc.

4.2.1 Bascule RS (Bistable RS Asynchrone)

Les bascules, aussi appelées "flip-flop" sont les éléments de base des systèmes séquentiels. Elles sont capables de MEMORISER une information logique donnée sous forme « ***d'impulsion*** ».

Une bascule à deux sorties dont l'une est l'inverse de l'autre. Elles se notent **Q** et $\overline{\mathbf{Q}}$, voir (figure 83).

A. Synoptique

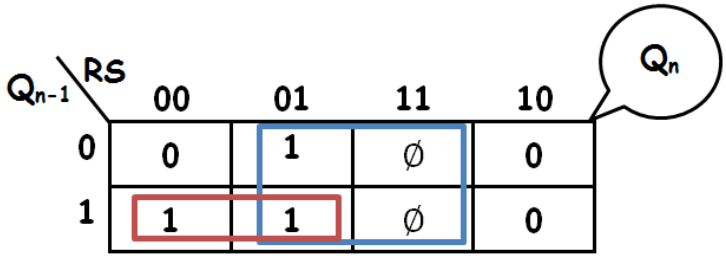
Figure83: Synoptique d'une Bascule RS Asynchrone

S : « SET » mise à 1
R : « RESET » remise à 0
R=S=0 : Conservation de l'état précèdent Q_n(mémorisation)
R=S=1 : Cas indéterminé
Remarque : la bascule RS est une bascule qui peut mémoriser qu'un seul bit.

R	S	Q_{n-1}	Qn	
0	0	0	**0**	Mémorisation
0	0	1	**1**	$Qn=Q_{n-1}$
0	1	0	**1**	Mise à 1
0	1	1	**1**	Qn=1
1	0	0	**0**	Mise à 0
1	0	1	**0**	Qn=0
1	1	0	**?**	Etat métastable
1	1	1	**?**	Cas indéterminé

R	S	Qn	
0	0	**0**	Mémorisation ($Qn=Q_{n-1}$)
0	1	**1**	Mise à 1
1	0	**0**	Mise à 0
1	1	**?**	Cas indéterminé

C. Tableau de KARNAUGH



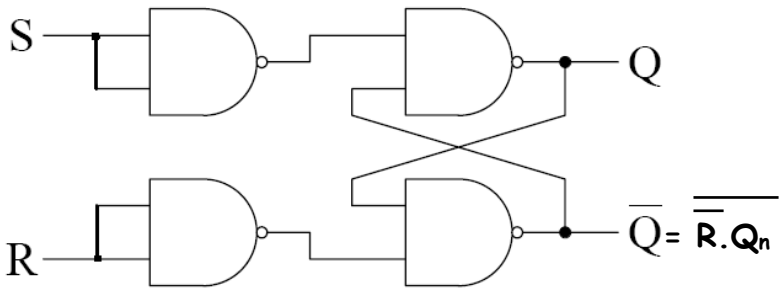
$Q_n=\overline{R}.Q_{n-1}+S$

D. Réalisation de la bascule RS à l'aide des portes NAND

$Qn=\overline{\overline{R}.Q_{n-1}+S}=\overline{\overline{R}.Q_{n-1}}.\overline{S}$ =(**Q_{n-1}** c'est **Q_n** à l'instant t-1) donc

$Qn=\overline{\overline{R}.Q_n}.\overline{S}$

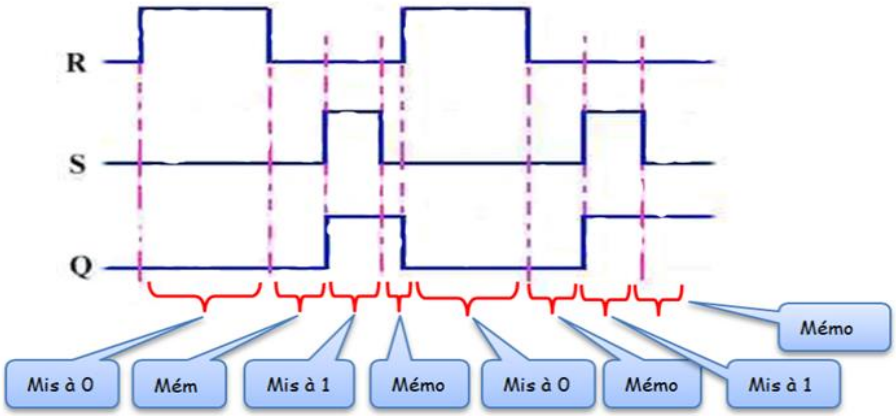
Figure84: Schémas logique d'une bascule **R**S asynchrone



E. Fonctionnement dynamique d'une bascule RS

En appliquant des impulsions à **R**, **S** on obtient le chronogramme suivant : voir (figure 85).

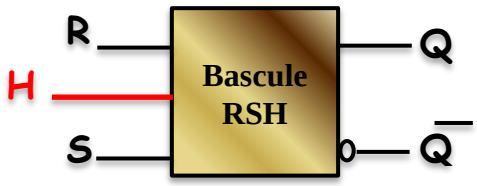
Figure85: Chronogramme d'une bascule **R**S asynchrone



4.2.2 Bascule RS Synchrone 'RSH'(Bistable RS synchrone)

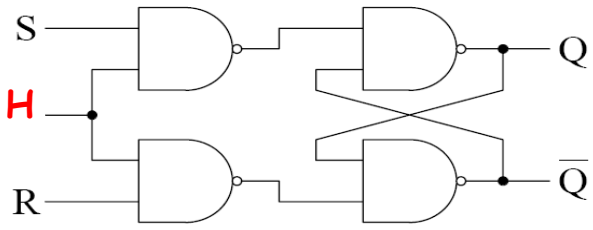
La bascule R.S.H. est une bascule pour laquelle les entrées S et R ne sont prises en compte qu'en coïncidence avec un signal de commande. Ce signal peut être fourni par une horloge, nous avons alors une bascule synchrone. Voir synoptique (figure 86) .

Figure86 : Synoptique d'une bascule **R**S synchrone



Lorsque le signal de commande, noté ici **H**, est à **1** la bascule fonctionne comme indiqué précédemment et les sorties suivent les variations des entrées **S** et **R**. Par contre, lorsque le signal de commande est à **0**, la bascule est **bloquée** : Q est indépendant des éventuels changements de **S** et **R**. L'état mémorisé correspond au dernier état avant le passage de la ligne de commande de **1** à **0,**voir schémas logique(figure 87) .

Figure87: Schémas logique d'une bascule RS synchrone



A. Table de vérité

H	R	S	Qn
0	0	0	Q_{n-1}
0	0	1	Q_{n-1}
0	1	0	Q_{n-1}
0	1	1	Q_{n-1}
1	0	0	Q_{n-1}
1	0	1	1
1	1	0	0
1	1	1	?

Mémorisation

Mise à 1

Mise à 0

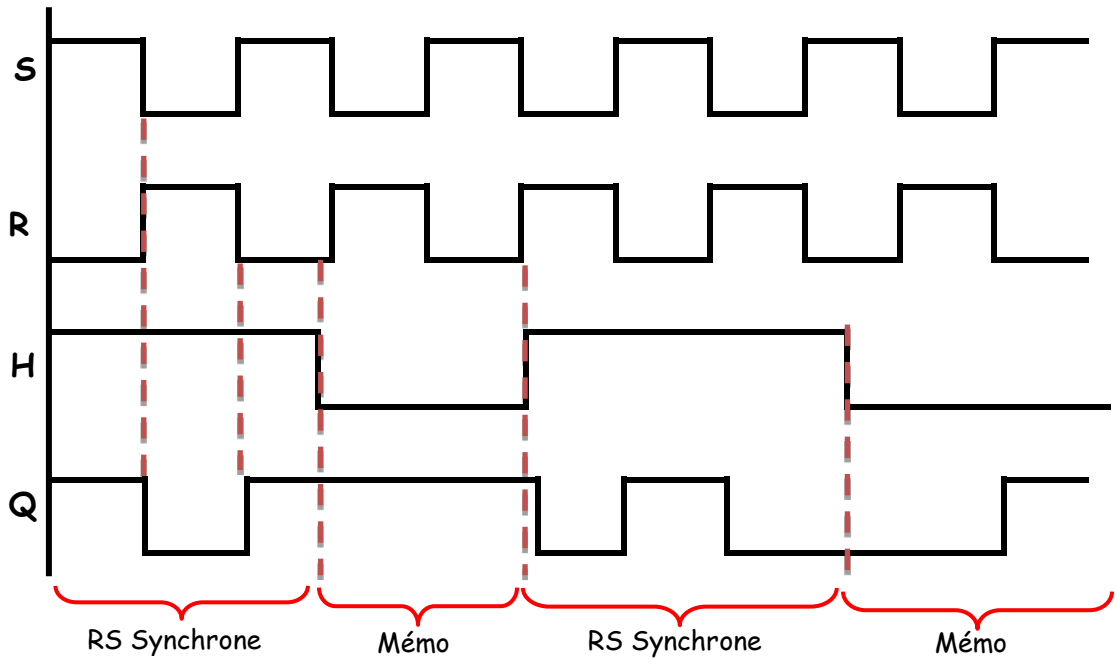
Cas indéterminé

- Si **H=0**(Conservation de l'état précèdent) On peut dire que la bascule est bloquée (Mémorisation)
- Si **H=1,** le RS Synchrone se comporte comme une bascule RS asynchrone

B. Fonctionnement dynamique d'une bascule RSH

En appliquant des impulsions à **R**, **S** et **H** on obtient le chronogramme suivant :

Figure88: Chronogramme d'une bascule RS synchrone



C. Les avantages des entrées synchronisées

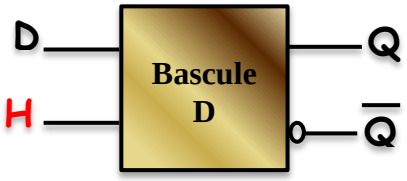
- Préparer les commandes R et S sans perturber la mémoire (H=0) et de ne les valider qu'au moment souhaité (H=0).
- Protéger la mémoire des parasites survenant en R et S tant que l'on à H=0 (on donne à H=1 la durée minimale).

4.2.3 Bascule D ou LATCH (Verrou) (Bistable D)

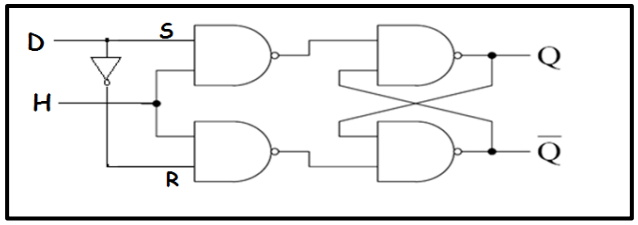
La bascule de type **D** ou **Latch** est dérivée de la bascule **R,S,H** . Elle possède quant à elle, une seul entrée « **D** » pour positionner les sorties. En effet, on place un inverseur entre l'entrée **S** et **R** de la bascule **R,S,H.**
L'entrée **S** devient l'entrée **D** de la bascule de type **D** dont le schéma et illustré dans la figure ci-dessous.

Figure89: Bascule **D**

(a): Synoptique



(b): Schémas logique



A. Table de vérité

H	D	Q_{n-1}	Qn	
0	0	0	0	Mémorisation $Qn=Q_{n-1}$
0	0	1	1	
0	1	0	0	
0	1	1	1	
1	0	0	0	Recopie D $Qn= D$
1	0	1	0	
1	1	0	1	
1	1	1	1	

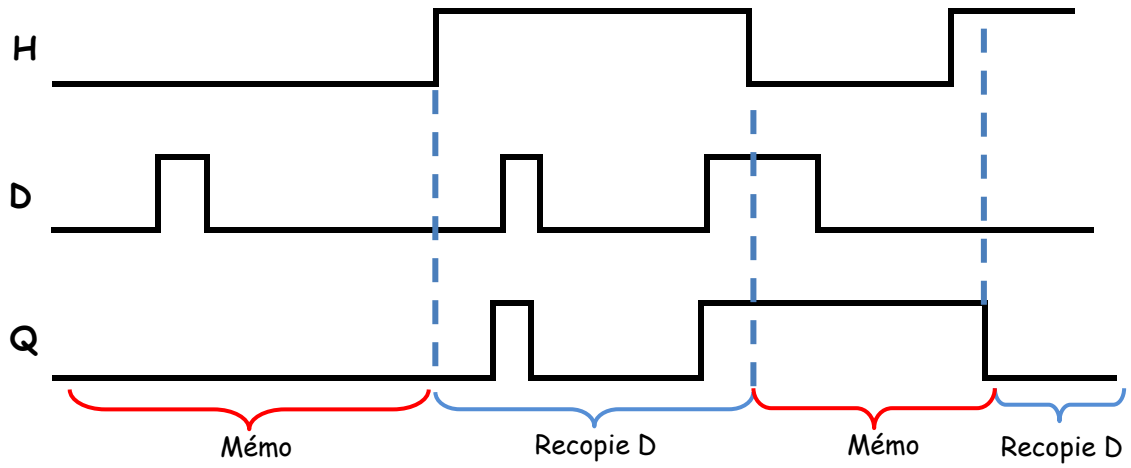
H	Qn
0	Q_{n-1}
1	D

Remarque : On remarque, d'après la table de vérité que le bistable D joue le rôle de mémoire élémentaire : le signale d'horloge représentant le signale de validation pour la donnée entrant dans le bistable

B. Fonctionnement dynamique d'une bascule D

En appliquant des impulsions à **D** et **H** on obtient le chronogramme suivant :

Figure90: Chronogramme d'une bascule D

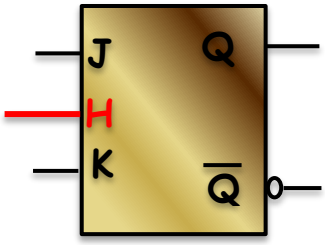


4.2.4 Bascule JK Asynchrone

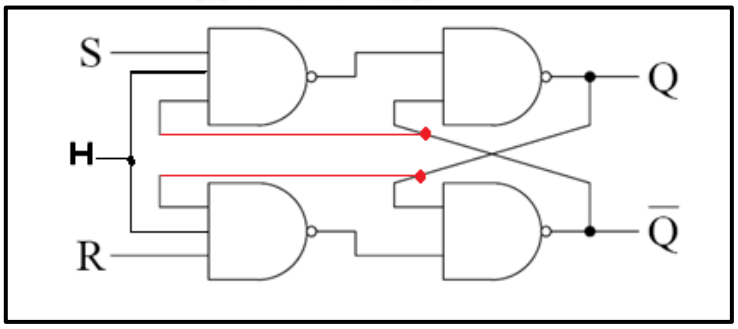
La bascule Bistable JK est similaire à la bascule RS, la seul différence est que la bascule JK permet d'éliminer l'indétermination (R=S=1) en ramenant les sorties Q et $\overline{Q}$ aux entrées, ce qui vas forcer la sortie à passer à son état complémentaire ($Q_n = \overline{Q}_{n-1}$). La seule modification et que les sorties du circuit logique sont reliées aux entrées, voir (figure 91).

Figure 91: Bascule **JK**

(a): Synoptique



(b): Schémas logique



J : mise à 1
K : « RESET » remise à 0
J=K=0 : Conservation de l'état précèdent Q_n(mémorisation)
J=K=1 : Basculement de l'état précèdent (1⟶0 ou 0⟶1)

A. Table de vérité

J	K	Q_{n-1}	Qn	
0	0	0	**0**	Mémorisation
0	0	1	**1**	Qn=Q_{n-1}
0	1	0	**0**	Mise à 0
0	1	1	**0**	Qn=0
1	0	0	**1**	Mise à 1
1	0	1	**1**	Qn=1
1	1	0	**1**	Basculement
1	1	1	**0**	Qn=$\overline{Q}_{n-1}$

J	K	Qn	
0	0	Q_{n-1}	Mémorisation (Qn=Q_{n-1})
0	1	**0**	Mise à 1
1	0	**1**	Mise à 0
1	1	$\overline{Q}_{n-1}$	Basculement

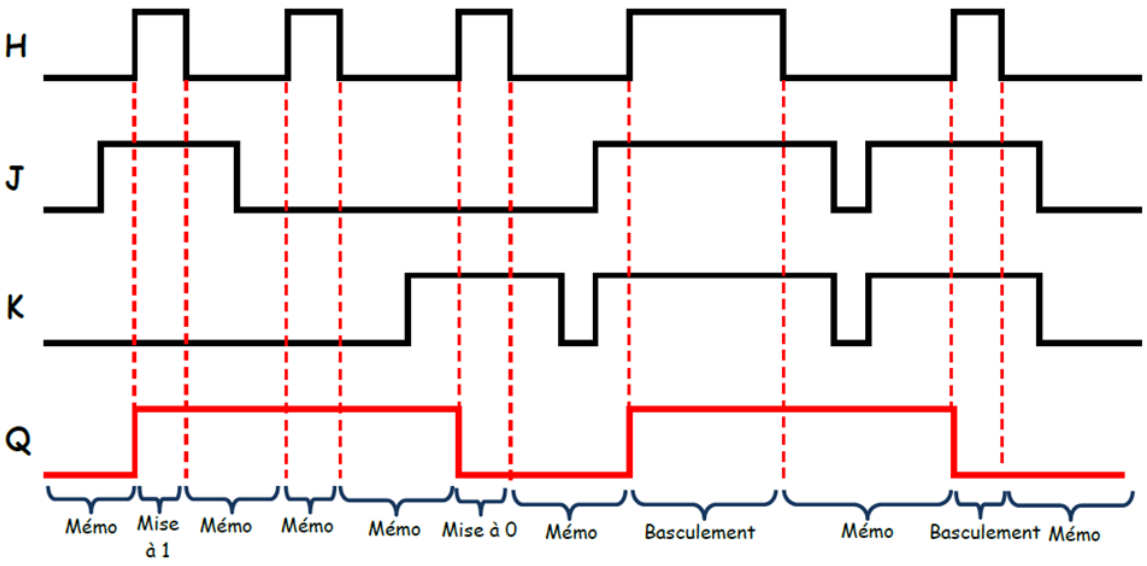
H=0 Mémorisation
H=1 Voire Tdv Bascule JK

Remarque : La bascule JK est la bascule la plus complète, offrant tous les modes de fonctionnement que l'on peut demander à une bascule.

B. Fonctionnement dynamique d'une bascule JK

En appliquant des impulsions à **J**, **K** et **H** on obtient le chronogramme suivant :voir (figure 92).

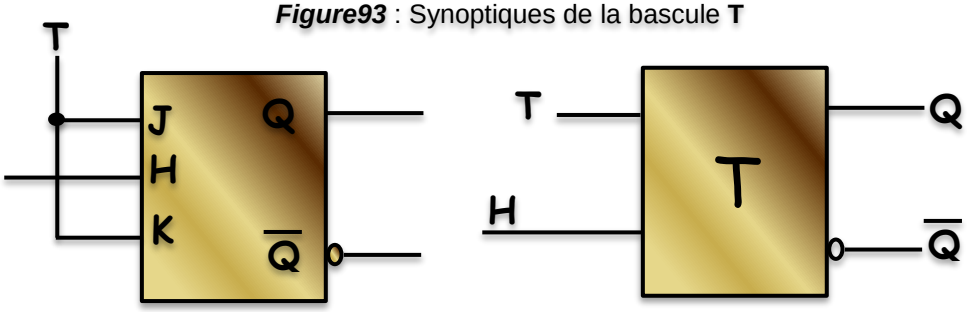
Figure 92: Chronogramme d'une bascule **JK**



4.2.5 Bascule T

La bascule **T** (Transit ou Toggle) comprend deux entrées « **T** » et « **H** », voir (figure 93). L'entrée **T** communique les mêmes niveaux au entrées **J** et **K**, donc **deux états** seulement sont possibles : le **Maintien** ou la **Complémentation**, la bascule **T** est connue spécialement pour son état de complémentation qui donne au circuit la fonction de **diviseur de fréquence (d'horloge) par deux**.

Figure93 : Synoptiques de la bascule **T**



A. Table de vérité

T	H	Q_{n-1}	Qn	
1	0	0	0	Mémorisation $Qn=Q_{n-1}$
1	0	1	1	
1	1	0	1	Basculement $Qn=\overline{Q}_{n-1}$
1	1	1	0	

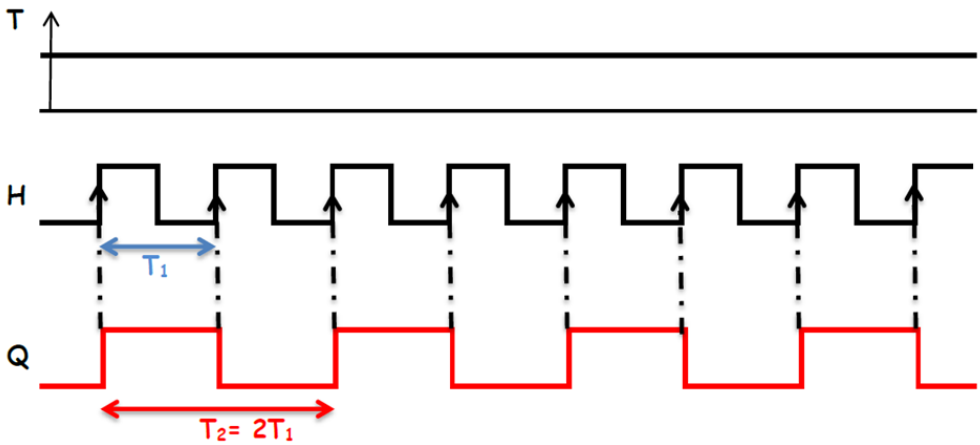
T	H	Qn
1	0	$\overline{Q}_{n-1}$
1	1	Q_{n-1}

Remarque : La bascule T change d'état à chaque impulsion d'horloge

B. Fonctionnement dynamique d'une bascule T

En appliquant des impulsions à **T** et **H** on obtient le chronogramme suivant : voir (figure 94).

Figure94 : Chronogramme d'une bascule **T**



Remarque : On remarque que si on donne à T le niveau logique 1, la période T_2 du signale de sortie est égale à deux fois la période T_1 du signale d'horloge, donc la bascule T est un diviseur de fréquence par deux

4.2.6 Déclenchement d'une bascule

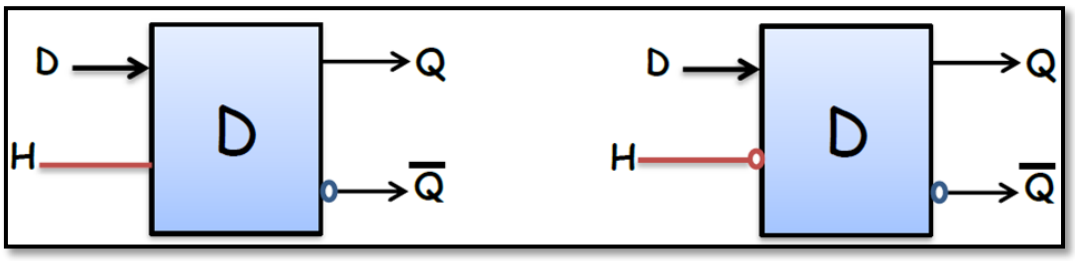
Une bascule asynchrone est déclenchée lorsque les signaux d'entrées changent. Dans le cas d'une bascule synchrone le déclenchement est plus compliqué, une bascule synchrone est pilotée par une horloge, de ce fait son déclenchement est provoquer par des impulsions. Donc on distingue deux types de bascules synchrones :

- Les bascules ***Latchs***
- Les bascules ***Flip_Flop***

A. Les bascules Latchs : Les bascules latchs réagissent sur un niveau d'horloge, cela revient à dire que ce type de bascule est déclencher quand l'horloge H=1 (***niveau haut***) ou quand H=0 (***niveau bas***), voir (figure 95).

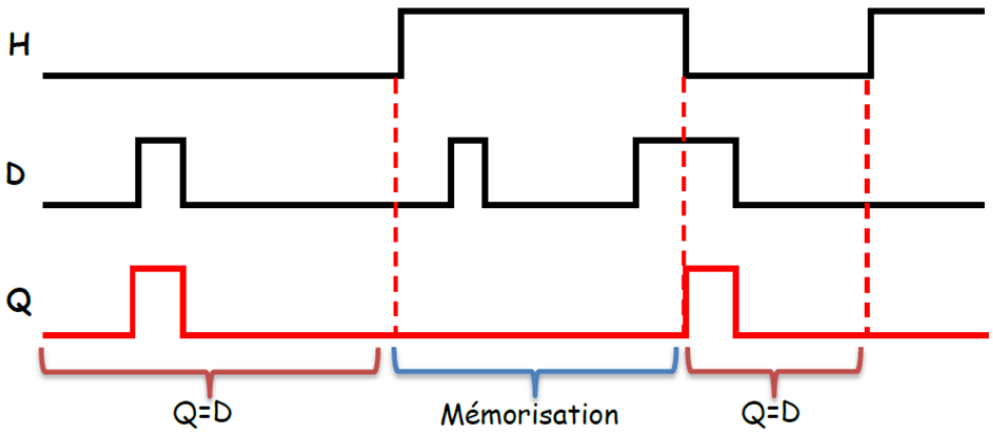
Figure95 : Synoptique d'une bascule **D Latch**

(a) : Déclenchable quand **H=1** ***(b)*** : Déclenchable quand **H=0**



❖ Fonctionnement dynamique d'une **bascule D** Latchs déclenchable par un niveau bas

Figure96 : Chronogramme d'une bascule **D Latch** déclenchable par niveau bas



B. Bascules Flip_Flop :

Ce type de bascule change d'état non pas quand l'horloge est au niveau 1 ou 0 , mais pondant la ***transition*** du signale d'horloge de l'état 0 à 1 (***Front montant***) ou l'ors de la transition de l'état 1 à 0 (***Front descendent***), voir (figure 97).

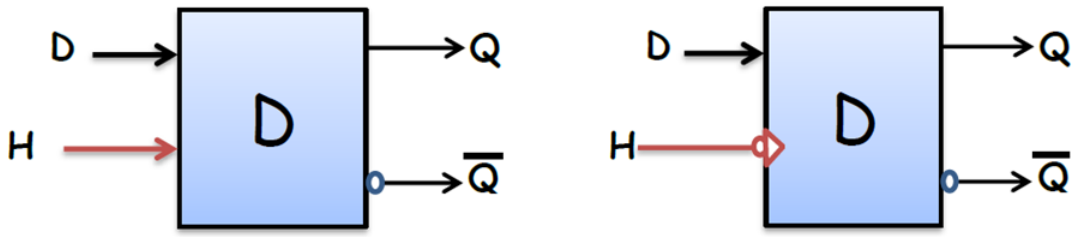
Figure97: Les fronts d'un signal d'horloge

a) Front montant du signale d'horloge

b) Front descendent du signale d'horloge

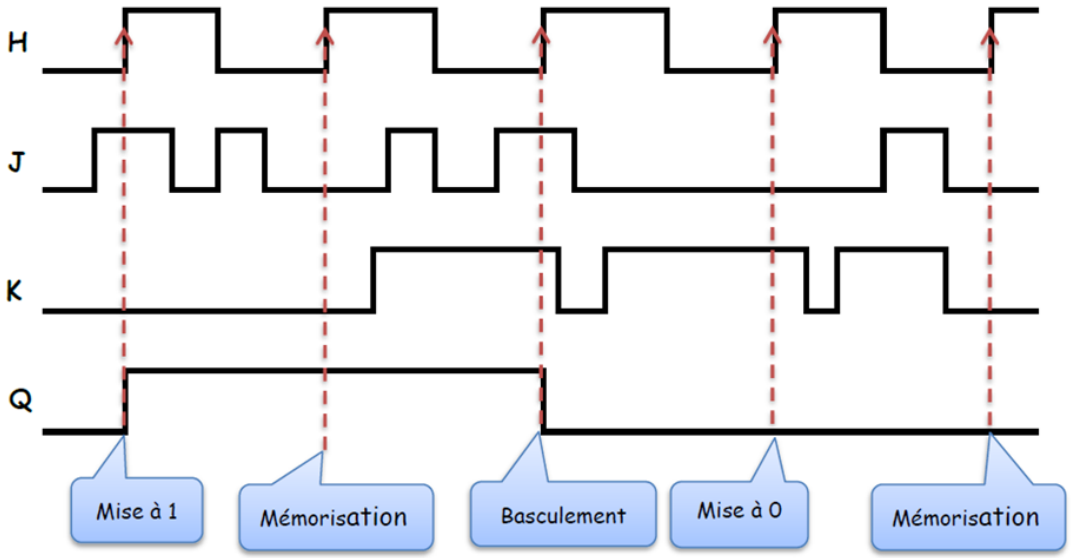
Figure98: Synoptique d'une bascule D Flip-Flop

(a) Déclenchable sur un front montant **(b)** Déclenchable sur un front descendent



❖ Fonctionnement dynamique d'une **bascule JK** Flip-Flop déclenchable sur un Front montant.

Figure 99: Chronogramme d'une bascule **JK** Flip-Flop déclenchable sur un Front montant.



4.2.7 Exercice 22 Donner les chronogrammes de Q1 et Q2 (On suppose qu'au début Q1 =0 et Q2=0)

Figure100: Bascule D et JK en cascade

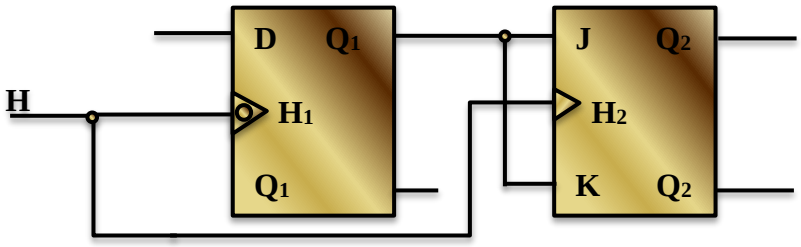
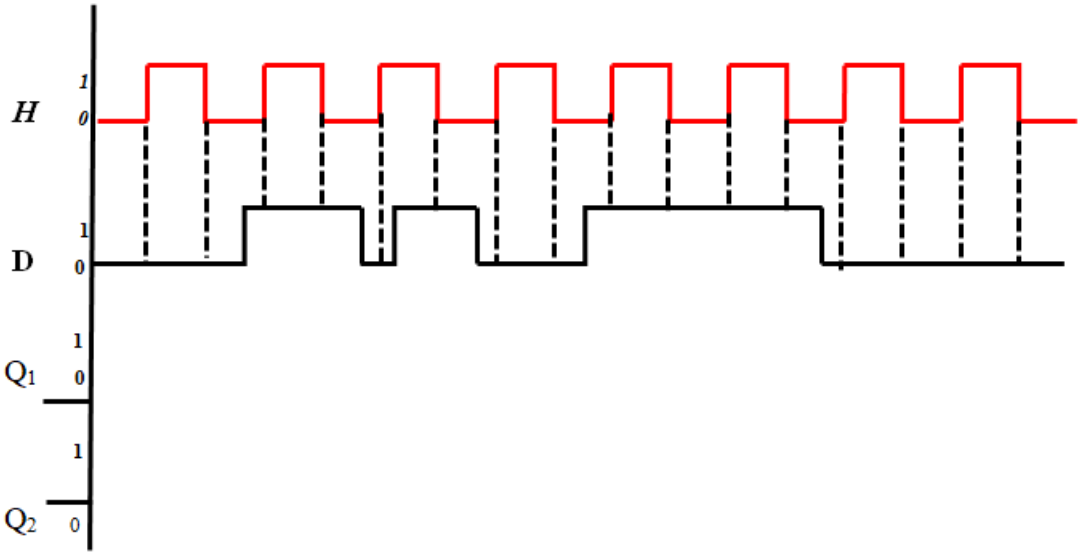


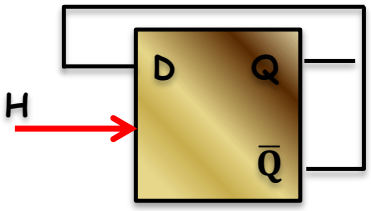
Figure101: Chronogrammes des bascule D et JK en cascade



4.2.8 Exercice 23

1. Soit la figure99 suivante

Figure102: Bascule D



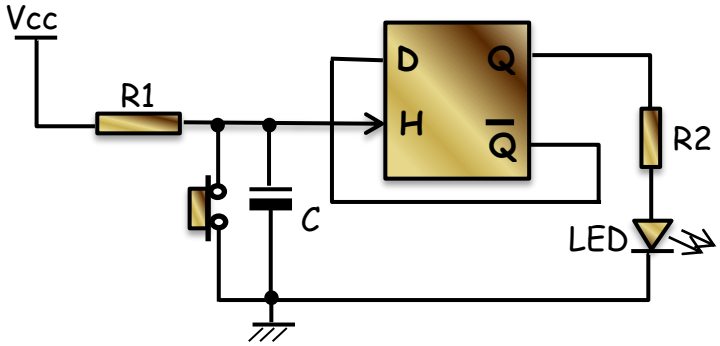
A. On applique à l'entrée d'horloge de la bascule D le signal d'horloge suivant :

Figure103: Signal d'horloge

B. Tracer le chronogramme obtenu de H et Q.

C. Expliquer le fonctionnement du montage de la figure suivant :

Figure104: Circuit à base de Bascule D



4.3 Compteurs

Un compteur est une succession de bascule JK ou T dans le mot binaire se modifie à chaque impulsion d'horloge, c'est-à-dire qu'il compte le nombre d'impulsion appliquer à son entrer d'horloge.

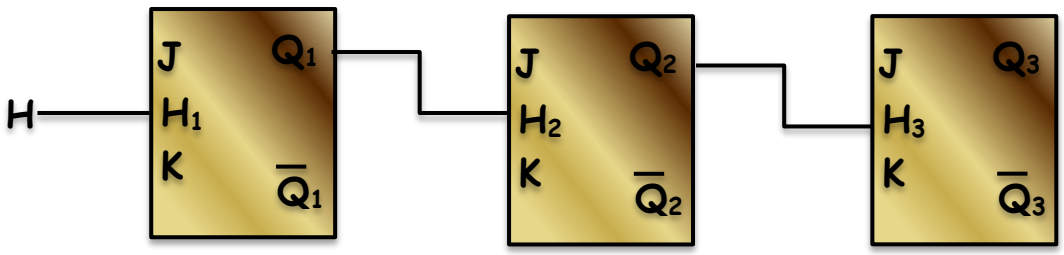
Il existe deux catégories de compteur :

- Les compteurs Asynchrone.
- Les compteurs Synchrone.

4.3.1 Compteurs Asynchrones

Un compteur est dit asynchrone, si les impulsions à compter (***impulsions d'horloge***) sont appliquer seulement sur l'entrée de la **première** bascule JK, puis l'**horloge** de la **deuxième** bascule JK est relié à la **sortie** ($\mathbf{Q_1}$ou $\overline{\mathbf{Q}}_\mathbf{1}$) de la première bascule JK, ensuite l'horloge de la **troisième** bascule JK est relié à la sortie ($\mathbf{Q_2}$ou $\overline{\mathbf{Q}}_\mathbf{2}$) de la **deuxième** bascule est ainsi de suite , voir (figure 105).

Figure 105: Synoptique d'un compteur asynchrone



A. Réalisation d'un compteur Asynchrone :

Pour réaliser un compteur binaire asynchrone, on utilise les propriétés de la bascule JK, pour cela en relient les entrées J et K au niveau 1(5V), puis en injectent des impulsions à l'entrée d'horloge.

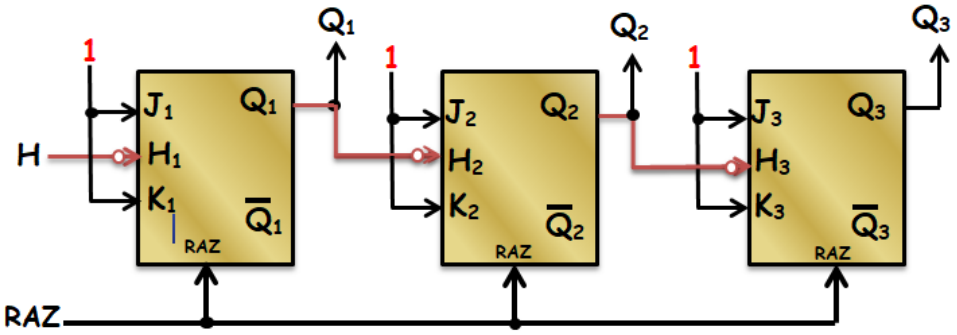
Remarque :

- ✓ Un compteur modulo **N** compte de **0** à **N-1** (**N** : capacité de comptage).
- ✓ $\mathbf{N \leq 2^X}$(**x** : nombres de bascules JK).

❖ Compteur asynchrone **modulo 8** (Flip Flop déclenchable sur front descendent).

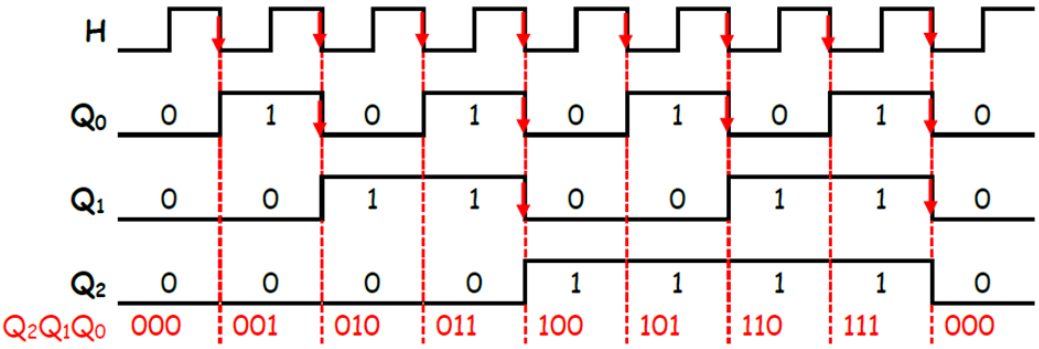
N=8=2^3⟹**3 bascules JK** (La bascule J_1 K_1 est celle du poids le plus faible)

Figure106: Synoptique d'un Compteur Asynchrone Modulo 8



❖ Fonctionnement dynamique d'un compteur Asynchrone Modulo 8 (Flip Flop déclenchable sur front descendent), voir (figure107).

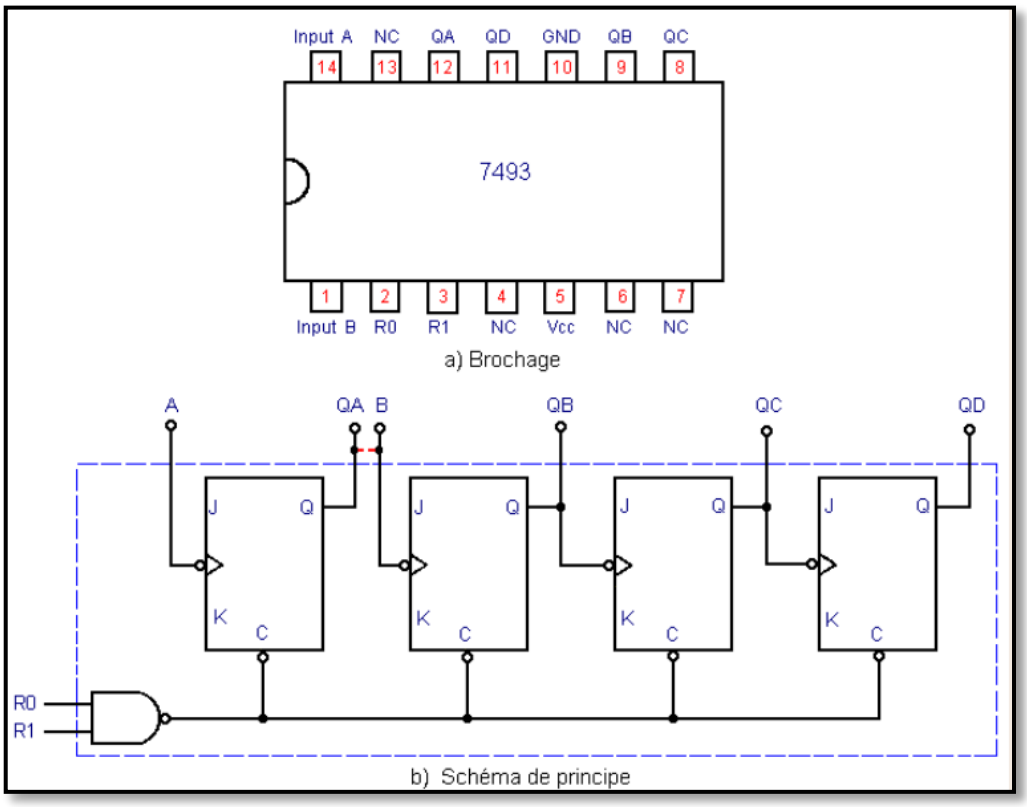
Figure 107 : Chronogramme d'un Compteur Asynchrone Modulo 8



B. Exemple de brochage d'un compteur Asynchrone

La figure 108 représente le schéma de principe et le brochage du compteur intégré 7493 réalisé en technologie TTL(Transistor Transistor logic) ainsi que son brochage.

Figure108: Compteur Asynchrone 7493



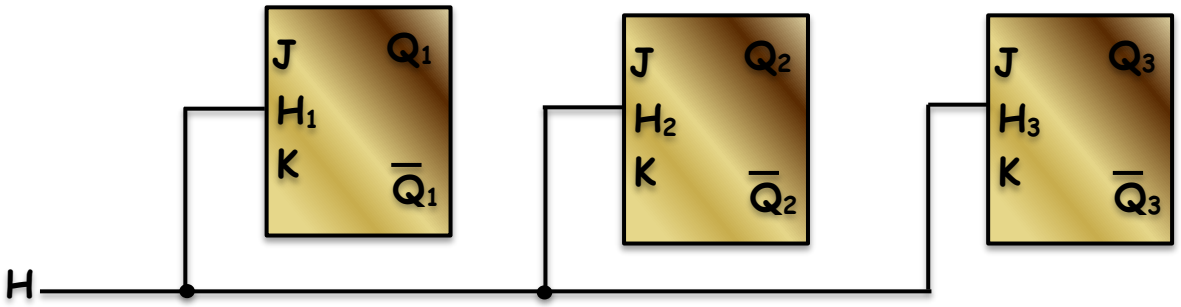
Source :https://www.electronique-et-informatique.fr/

- Le symbole «**NC**.» signifie «non connecté».
- Les entrées **J** et **K** des bascules sont câblées intérieurement à «**1**».
- Une remise à zéro générale asynchrone du compteur est possible grâce aux entrées**R0** et **R1.** Pour cela les deux entrées **R0** et **R1** doivent être simultanément à «**1**».
- Ce compteur peut fonctionner en **diviseur** par **8** en présentant l'horloge sur l'entrée **B** ou en diviseur par **16** en présentant l'horloge sur l'entrée **A** et en reliant la sortie **QA** à l'entrée **B**.

4.3.2 Compteurs Synchrones :

Un compteur est dit synchrone, si les impulsions à comptées (***horloge***) sont appliquer simultanément (aux même temps) sur l'entrée ***CLOCK*** (horloge) de toutes les bascule, voir (figure 109).

Figure109: Compteurs Synchrones



A. Réalisation d'un compteur Synchrone :

Pour réaliser un compteur Synchrone modulo N, on procède comme suite :

- ✓ On définit le nombre de bascule nécessaire pour réaliser ce compteur suivant la formule $N \leq 2^x$ (**x** : nombres de bascules JK).
- ✓ Ecrire le tableau de séquence (Table de vérité) des bascules A,B,C… selon la séquence de comptage.
- ✓ Etablir à partir des Table de karnaugh les équations de $(J_A,K_A)(J_B,K_B)(J_C,K_C)$ … de chaque bascule en fonction des sorties Q_A , Q_B,Q_C ……

  $J_A= f(Q_A ,Q_B ,Q_C …)$, $K_A=f(Q_A ,Q_B ,Q_C …)$
  $J_B= f(Q_A ,Q_B ,Q_C …)$, $K_B=f(Q_A ,Q_B ,Q_C …)$
  $J_C= f(Q_A ,Q_B ,Q_C …)$, $K_C=f(Q_A ,Q_B ,Q_C …)$
  .
  .
- ✓ Réaliser le schéma logique du compteur.

B. Compteur synchrone modulo 10(Flip Flop déclenchable sur front Montant).

Un compteur modulo 10 compte de 0 à 10-1

N=10≤2^4⟹ 4 bascules JK

❖ Table de vérité

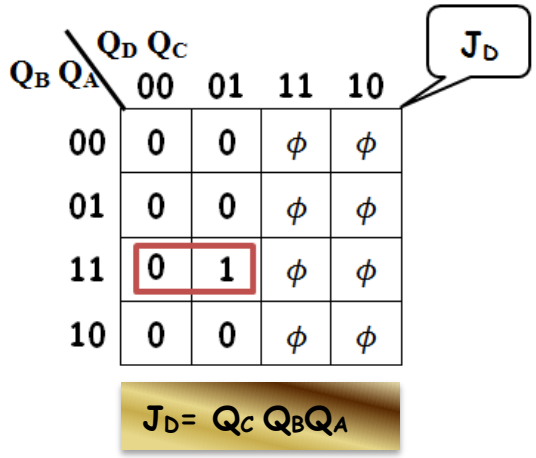
Q_n	Q_{n+1}	J	K
0	0	0	0
		1	0
		0	Ø
1	1	0	0
		0	1
		Ø	0

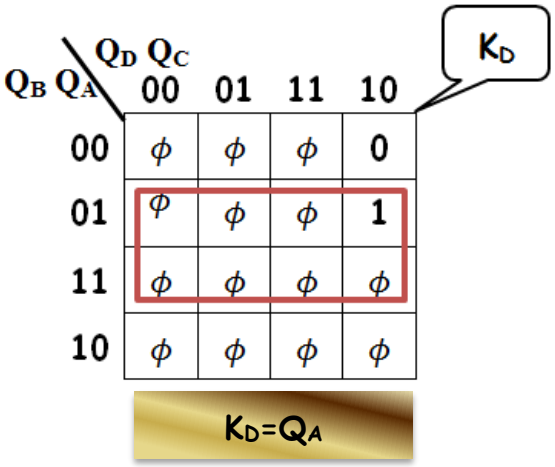
Q_n	Q_{n+1}	J	K
0	1	1	1
		1	0
		1	Ø
1	0	1	1
		0	1
		Ø	1

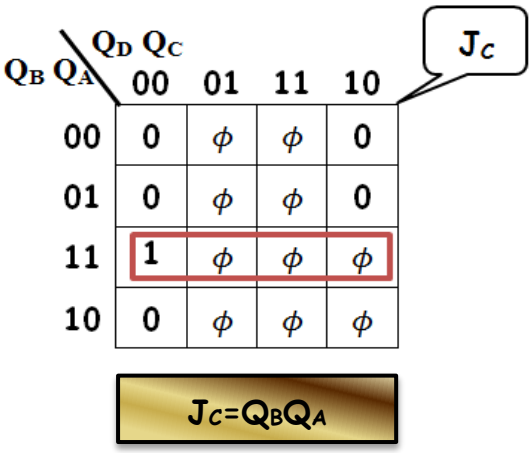
Qn	Qn+1	J	K
0	0	0	Ø
1	1	Ø	0
0	1	1	Ø
1	0	Ø	1

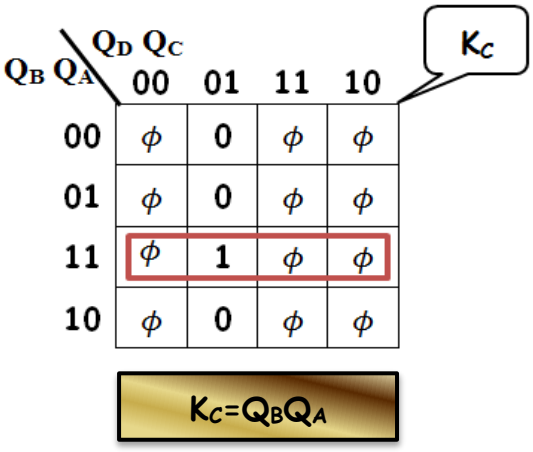
	Q_D	Q_C	Q_B	Q_A	J_D	K_D	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	0	ϕ	0	ϕ	0	ϕ	1	ϕ
1	0	0	0	1	0	ϕ	0	ϕ	1	ϕ	ϕ	1
2	0	0	1	0	0	ϕ	0	ϕ	ϕ	0	1	ϕ
3	0	0	1	1	0	ϕ	1	ϕ	ϕ	1	ϕ	1
4	0	1	0	0	0	ϕ	ϕ	0	0	ϕ	1	ϕ
5	0	1	0	1	0	ϕ	ϕ	0	1	ϕ	ϕ	1
6	0	1	1	0	0	ϕ	ϕ	0	ϕ	0	1	ϕ
7	0	1	1	1	1	ϕ	ϕ	1	ϕ	1	ϕ	1
8	1	0	0	0	ϕ	0	0	ϕ	0	ϕ	1	ϕ
9	1	0	0	1	ϕ	1	0	ϕ	0	ϕ	ϕ	1

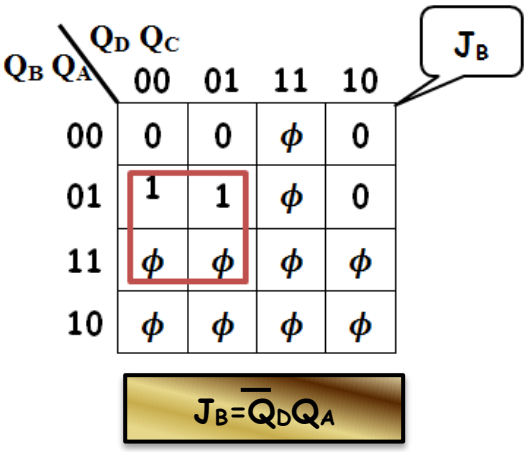
❖ **Tableau Karnaugh**

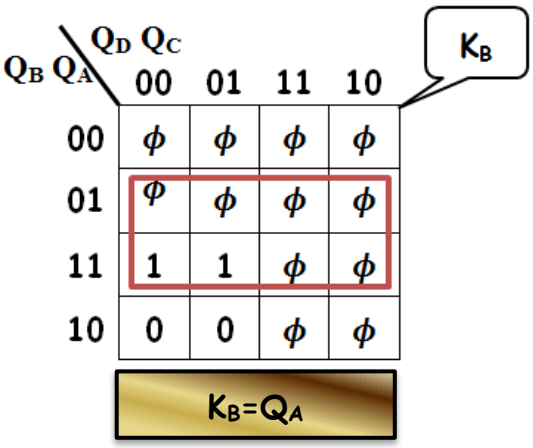












❖ **Equations :**

- ✓ $J_A=1$
- ✓ $K_A=1$

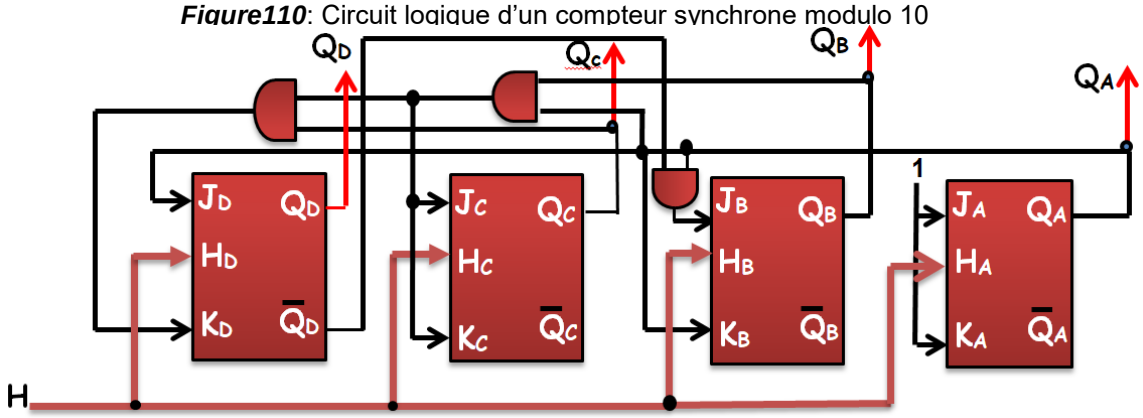
- ✓ $J_B=\overline{Q}_D Q_A$
- ✓ $K_B=Q_A$

- ✓ $K_C=Q_B Q_A$
- ✓ $J_C=Q_B Q_A$

- ✓ $J_D=Q_A$
- ✓ $K_D= Q_C Q_B Q_A$

❖ **Circuit logique**

Figure110: Circuit logique d'un compteur synchrone modulo 10



4.3.3 Compteur Synchrone réversible (Compteur décompteur)

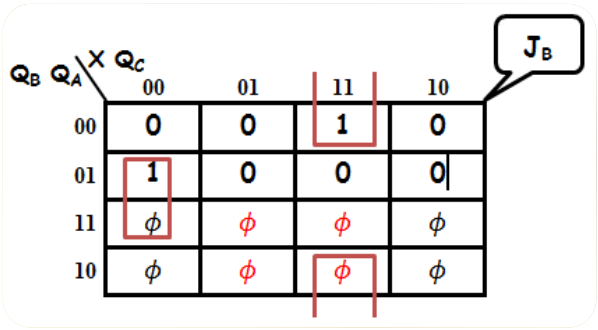
Un compteur réversible est un compteur, binaire ou décimal qui permet de compter ou décompter selon le niveau logique appliqué à une broche de commande appelée Sens de comptage (*Up/down*), voir exemple ci-dessous.

❖ **Exemple :** Réalisation d'un compteur réversible (compteur/décompteur modulo 6)

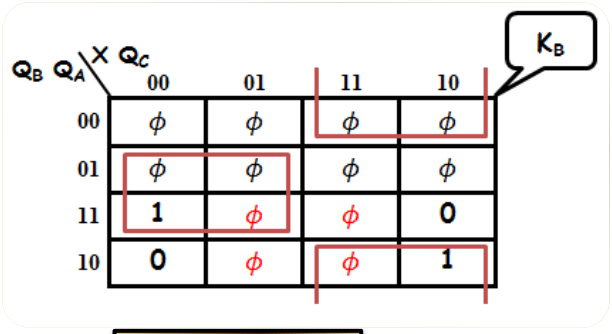
A. Table de vérité :

	Imp.	X	Q_C	Q_B	Q_A	J_C	K_C	J_B	K_B	J_A	K_A
COMPTAGE	1	0	0	0	0	0	ϕ	0	ϕ	1	ϕ
	2	0	0	0	1	0	ϕ	1	ϕ	ϕ	1
	3	0	0	1	0	0	ϕ	ϕ	0	1	ϕ
	4	0	0	1	1	1	ϕ	ϕ	1	ϕ	1
	5	0	1	0	0	ϕ	0	0	ϕ	1	ϕ
	6	0	1	0	1	ϕ	1	0	ϕ	ϕ	1
DECOMPTA	1	1	1	0	1	ϕ	0	0	ϕ	ϕ	1
	2	1	1	0	0	ϕ	1	1	ϕ	1	ϕ
	3	1	0	1	1	0	ϕ	ϕ	0	ϕ	1
	4	1	0	1	0	0	ϕ	ϕ	1	1	ϕ
	5	1	0	0	1	0	ϕ	0	ϕ	ϕ	1
	6	1	0	0	0	1	ϕ	0	ϕ	1	ϕ

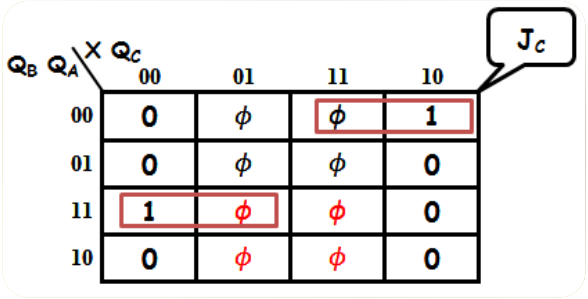
B. Tableau de Karnaugh :



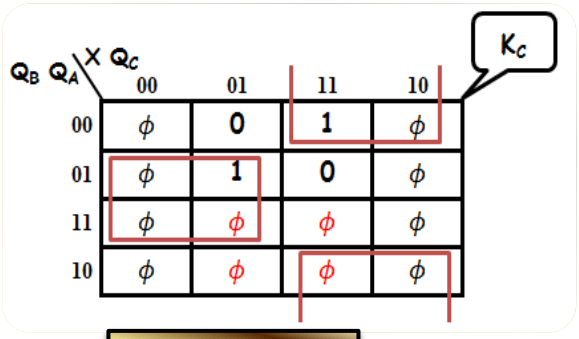
$J_B = X.Q_C + \overline{X}.\overline{Q}_C.Q_A$



$K_B = X \oplus Q_A$



$J_C = X.\overline{Q}_A.\overline{Q}_B + \overline{X}.Q_A.Q_B$



$K_C = X \oplus Q_A$

C. Equations :

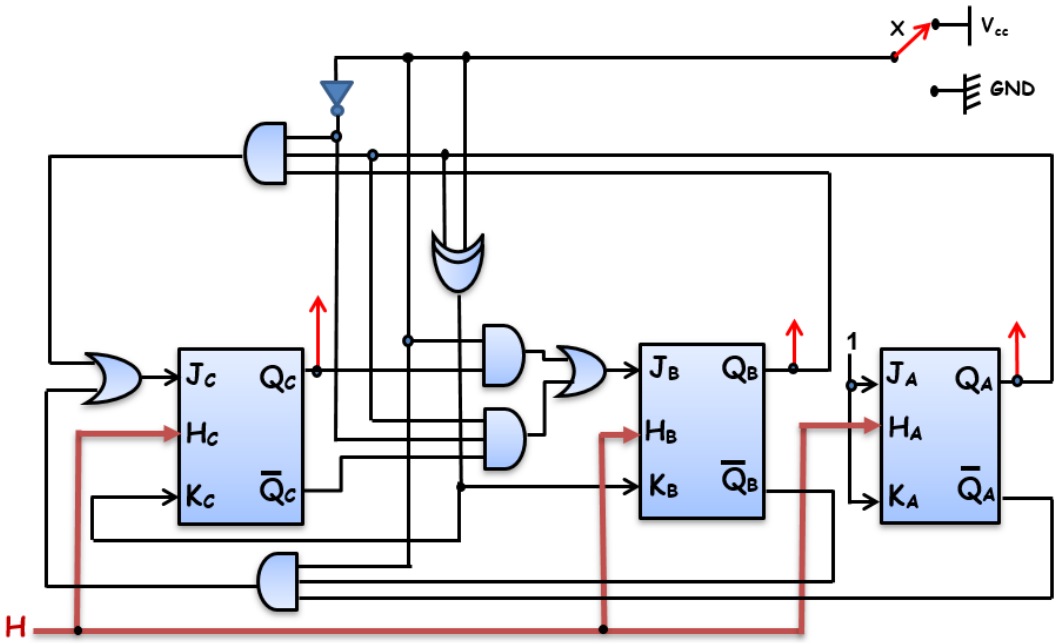
$J_A = 1$
$K_A = 1$

$J_B = X.Q_C + \overline{X}.\overline{Q}_C.Q_A$
$K_B = X \oplus Q_A$

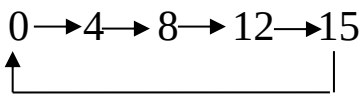
$J_C = X.\overline{Q}_A.\overline{Q}_B + \overline{X}.Q_A.Q_B$
$K_C = X \oplus Q_A$

D. Circuit logique

Figure111: Circuit logique d'un compteur décompteur modulo 6

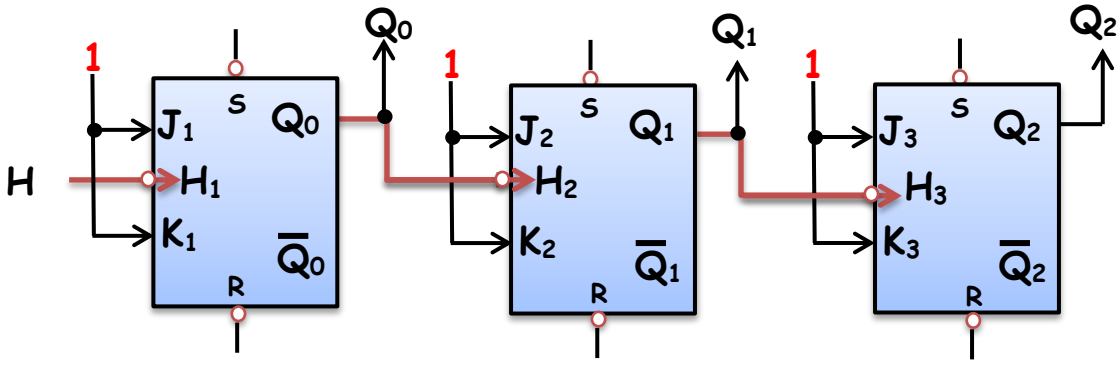


4.3.4 Exercice 24Réalisez un compteur synchrone qui affiche la séquence suivante :



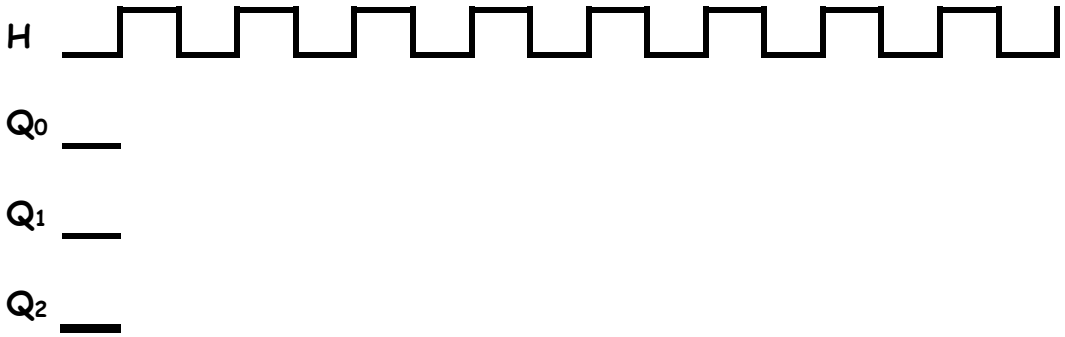
4.3.5 Exercice 25 Soit la suivante :

Figure112: Compteur Asynchrone



A. Compléter le chronogramme de la (figure 113) :

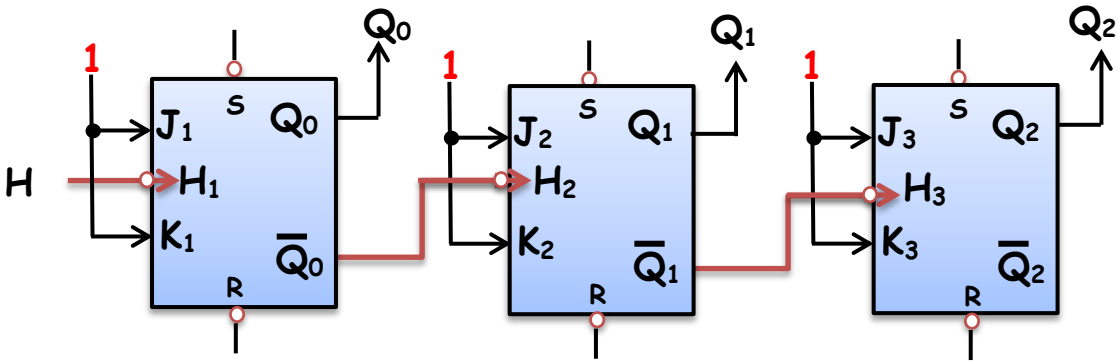
Figure113: Chronogramme du compteur asynchrone de la figure112



B. Quelle est la séquence obtenue ?
C. Quel est le modulo ?

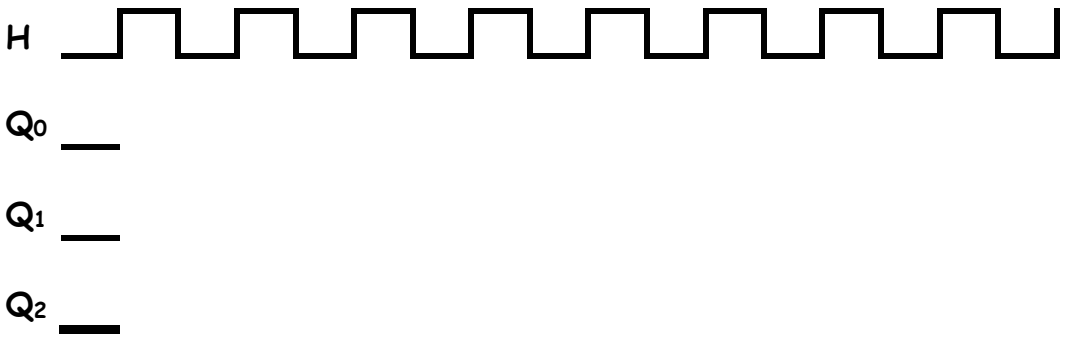
4.3.6 Exercice 26 Soit la figure ci-dessous :

Figure114 : Compteur Asynchrone



A. Compléter le chronogramme de la (figure 115) :

Figure115 : Chronogramme du compteur asynchrone de la figure114



B. Quelle est la séquence obtenue ?
C. Quel est le modulo ?
D. Conclure pour le compteur et le décompteur
E. A partir des 2 schémas (figure 112 et figure114) précédents concevoir un compteur/décompteur

4.4 Registres à décalage

Un registre est **une mémoire** qui peut mémoriser un mot (information) de **N** bits, avec possibilité de décalé à droite ou à gauche le contenu de ce registre, on parle alors de **registre à décalage**, dans ce type de registre le contenue de la bascule se décale d'un rang à chaque impulsion d'horloge.

4.4.1 Caractéristiques d'un registre

Tout registre est caractérisé par :

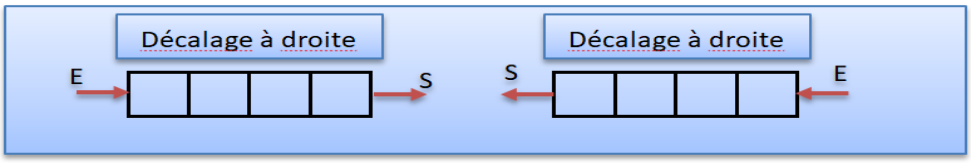
- La capacité: nombre de bits du mot binaire qu'il peut mémoriser.
- Le mode d'écriture ou de chargement: dépend du nombre d'entrées :
  - ✓ Ecriture série : génération bit par bit, avec transmission par un seul fil conducteur.
  - ✓ Ecriture parallèle : génération globale du mot de n bits, avec transmission par un bus de n bits (n fils conducteurs).
- Le mode de lecture:
  - ✓ Lecture série : exploitation bit par bit du mot (une seule sortie).
  - ✓ Lecture parallèle : exploitation globale du mot (n sorties).

4.4.2 Différents types de registres

Les registres à décalages différents par le nombre d'information (bits) qu'elles peuvent stockées et par le mode d'introduction et d'extraction de ces informations :

A. Registre à décalage 4 bits entrée série sortie série :SISO *(Serial IN- Serial OUT).*

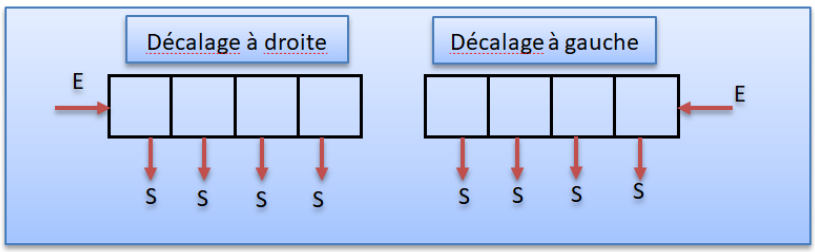
Figure116: Registre à décalage 4 bits entrée série



Exemple d'utilisation : ces registres permettent de décaler les informations

B. Registre à décalage 4 bits entrée série sortie parallèle :SIPO *(Serial IN- Parallel OUT).*

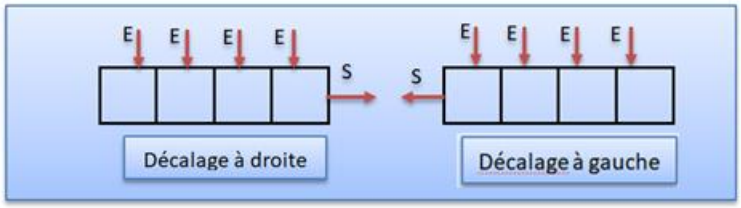
Figure117: Registre à décalage 4 bits entrée série sortie parallèle



Exemple d'utilisation : conversion série parallèle

C. Registre à décalage 4 bits entrée parallèle sortie série : PISO *(Parallel IN-Serial OUT).*

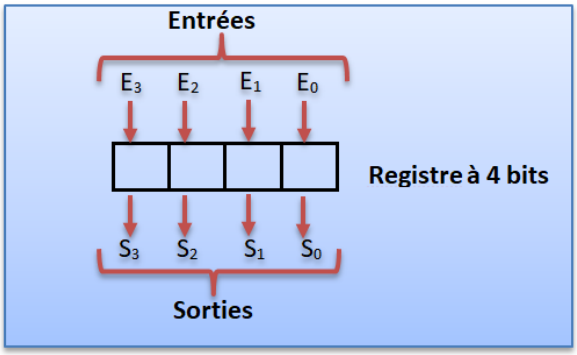
Figure118: Registre à décalage 4 bits entrée parallèle sortie série



Exemple d'utilisation: transmission en série des données sur une ligne (conversion parallèle série

D. Registre à décalage 4 bits entrée parallèle sortie parallèle :PIPO *(Parallel IN-Parallel OUT)*

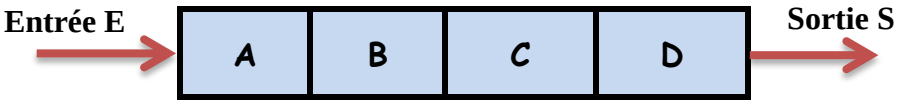
Figure119: Registre à décalage 4 bits entrée parallèle sortie parallèle



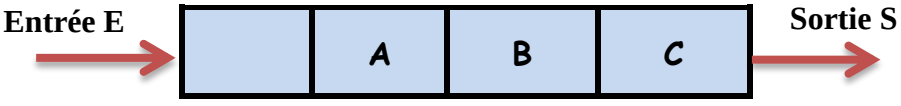
Exemple d'utilisation : mémorisation des états des sorties des compteurs avant affichage

4.4.3 Fonctionnement d'un registre à décalage

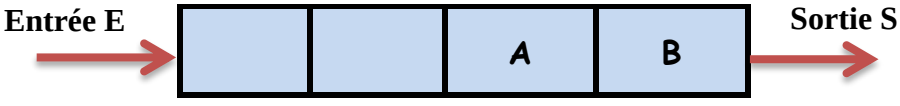
A. Principe de fonctionnement d'un registre à décalage 4 bits entrée/sortie série



➢ **1er Impulsion**

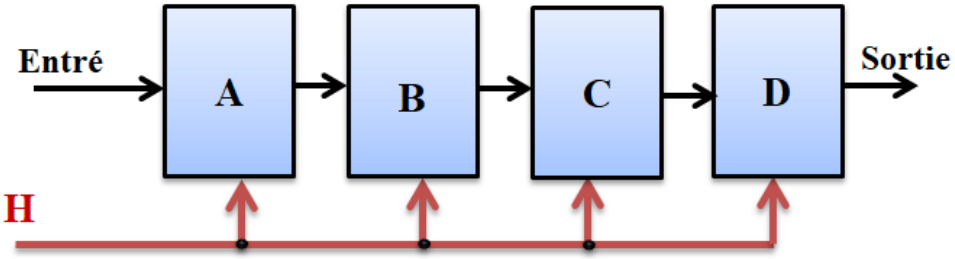


➢ **2eme Impulsion**



B. Registre à décalage 4 bits entrée/sortie série à base de bascule JK ou D

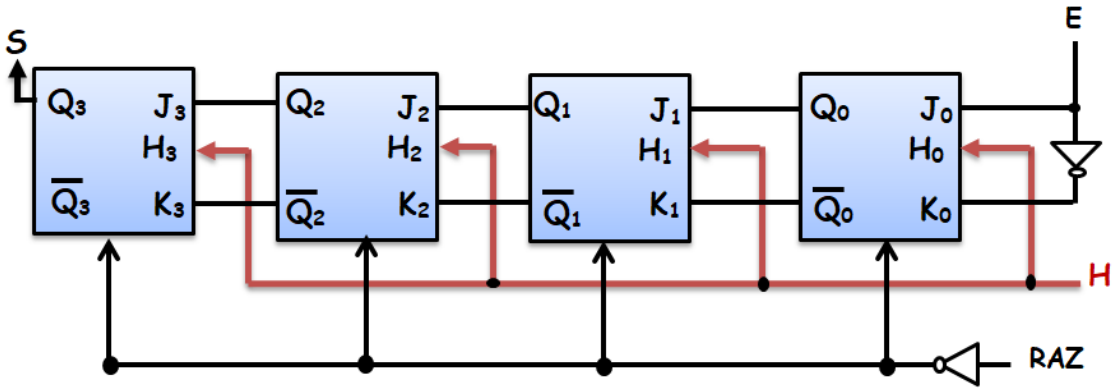
Figure120: Synoptique d'un Registre à décalage 4 bits entrée/sortie série



La 1ère impulsion d'horloge introduit la 1er information (1er bit) dans la **bascule A**, la 2eme impulsion décale l'information de la **bascule A** vers la **bascule B** et introduit la 2eme information (2eme bit) dans la **bascule A** et ainsi de suite jusqu'à la 4ieme impulsion.

4.4.4 Registre à décalage à l'aide de bascule JK

Figure121: Registre à décalage 4 bits entrée/sortie série à base de bascule JK



Ce registre contient 4 bascules JK, il peut stocker une information de 4 bits, les sorties Q et $\overline{Q}$ sont relier à J et K de la bascule suivante.

Exemple : On veut introduire l'information **1011** dans le registre

(Q_3=1, Q_2=0, Q_1=1,Q_0=1)

E=11erImp d'horloge

J_1=1, K_1=0 ⇒ Q_0=1

J_2=0, K_2=1 ⇒ Q_1=0

J_3=0, K_3=1 ⇒ Q_2=0

J_4=0, K_4=1 ⇒ Q_3=0

Le Registre affiche **(0001)**

E=02emeImp d'horloge

J_1=0, K_1=1 ⇒ Q_0=0

J_2=1, K_2=0 ⇒ Q_1=1

J_3=0, K_3=1 ⇒ Q_2=0

J_4=0, K_4=1 ⇒ Q_3=0

Le Registre affiche **(0010)**

E=1 3eme Imp d'horloge

J_1=1, K_1=0 ⇒ Q_0=1

J_2=0, K_2=1 ⇒ Q_1=0

J_3=1, K_3=0 ⇒ Q_2=1

J_4=0, K_4=1 ⇒ Q_3=0

Le Registre **affiche(0101)**

E=14emeImp d'horloge

J_1=1, K_1=0 ⇒ Q_0=1

J_2=1, K_2=0 ⇒ Q_1=1

J_3=0, K_3=1 ⇒ Q_2=0

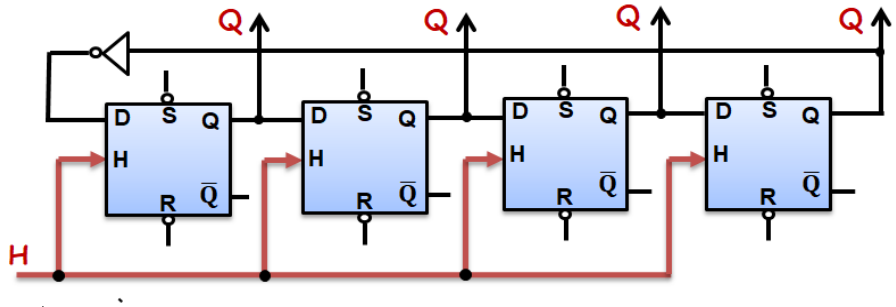
J_4=1, K_4=0 ⇒ Q_3=1

Le Registre affiche **(1011)**

	J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0	Q_3	Q_2	Q_1	Q_0
E=1 1erImp d'horloge	0	1	0	1	0	1	1	0	0	0	0	1
E=0 2emeImp d'horloge	0	1	0	1	1	0	0	1	0	0	1	0
E=1 3emeImp d'horloge	0	1	1	0	0	1	1	0	0	1	0	1
E=1 4emeImp d'horloge	1	0	0	1	1	0	1	0	1	0	1	1

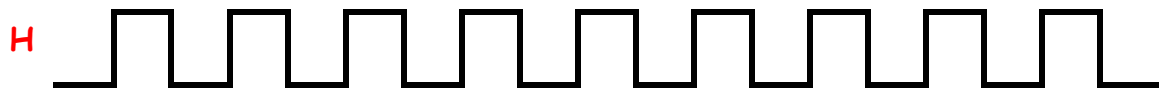
4.4.5 Exercice 27 Soit le circuit de (figure 122) suivant :

Figure122: Registre à décalage 4 bits à base



A. Compléter le chronogramme de la (figure 123) suivant :

Figure123: Chronogramme du registre à décalage 4 bits à base de bascule D



Q_0 _____

Q_1 _____

Q_2 _____

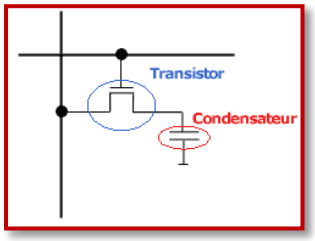
Q_3 _____

B. Quelle est la fonction réalisée?

4.5 Les mémoires

La mémoire se présente sous forme de composants électroniques ayant la capacité de retenir des informations (les informations étant de type « ***binaire*** », 0 ou 1). Chaque "bit" mémoire est composé d'un transistor (qui permet de lire ou d'écrire une valeur) accouplé à un condensateur (qui permet de retenir l'état binaire : 1 quand il est chargé et 0 quand il est déchargé), voir « figure 124 ». La mémoire est organisée sous forme de lignes et de colonnes. A chaque intersection correspond un bit de mémoire.

Figure 124: Représentation d'un bit mémoire



Source : https://www.vulgarisation-informatique.com

4.5.1 Organisation d'une mémoire:

Les cellules d'une mémoire sont organisées selon des lignes et des colonnes d'une structure matricielle. Les mémoires sont organisés on mots de 2, 4, 8, 16, 32,64bits

Une mémoire qui a la capacité de 1024 bits peut être organisée comme suite :

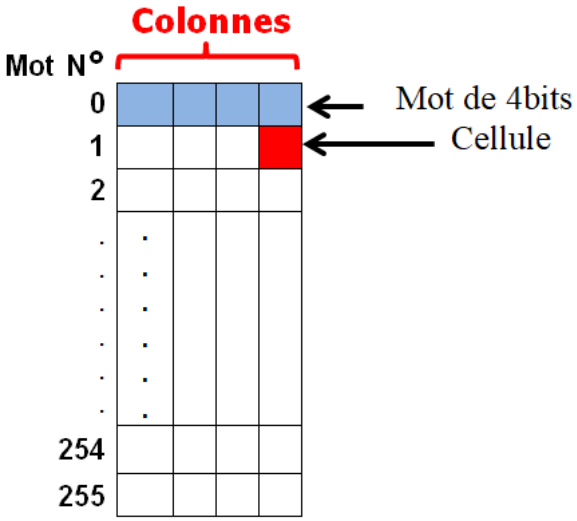
Capacité : 1024 bits :

- 512 mots de 2 bits
- 256 mots de 4 bits
- 128 mots de 8 bits
- 64 mots de 16 bits

Une mémoire peut être représentée comme une armoire de rangement constituée de différents tiroirs comme le montre la « figure 125 ». Chaque tiroir représente alors une case mémoire qui peut contenir un seul élément (des données). Le nombre de cases mémoires (mots) pouvant être très élevé, il est alors nécessaire de pouvoir les identifier par un numéro. Ce numéro est appelé adresse voir « figure 126 ». Chaque donnée devient alors accessible grâce à son adresse (bus d'adresse). Avec une adresse de **n** bits il est possible de référencer au plus $\mathbf{2^n}$ cases mémoire. Chaque case est remplie par un mot de données à travers le bus de donnée (sa longueur m est toujours une puissance de 2). Le nombre de fils d'adresses d'un boîtier mémoire définit donc le nombre de cases mémoire que comprend le boîtier. Le nombre de fils de données définit la taille des données que l'on peut sauvegarder dans chaque case mémoire.

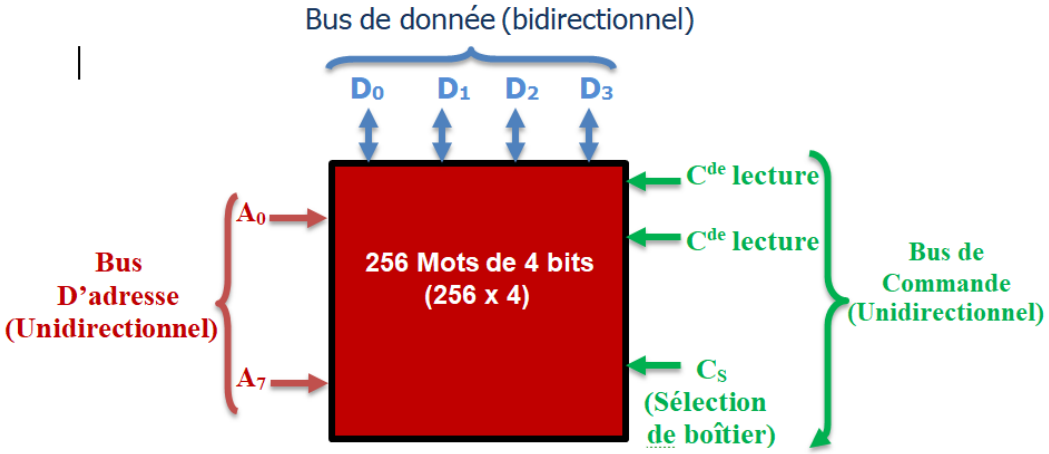
Exemple01 : Mémoire d'une capacité de 1024 bits organisé on 256 mots de 4 bits

Figure125 : Mémoire de 1024bits Organisé on mots de 4bits (256 mots de 4 bits)



En plus du bus d'adresses et du bus de données, un boîtier mémoire comprend un bus de commande qui permet de définir le type d'action que l'on effectue avec la mémoire (lecture/écriture) et une entrée de sélection du boitier mémoire voir « figure 126 ».

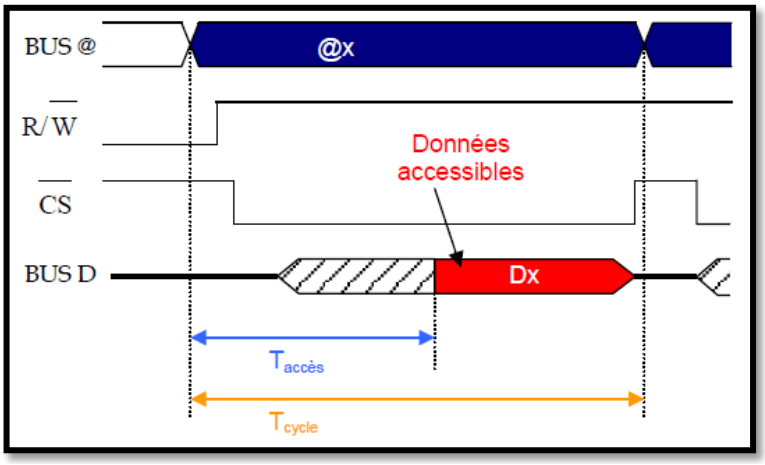
Figure126: Synoptique d'une mémoire de 1024bits Organisé on mots de 4bits (256 mots de 4 bits)



Une opération de lecture ou d'écriture de la mémoire suit toujours le même cycle Voir « figure 127 »:

1. Sélection de l'adresse
2. Choix de l'opération à effectuer (R/W)
3. Sélection de la mémoire (CS = 0)
4. Lecture ou écriture de la donnée

Figure127: Chronogramme d'un cycle de lecture



Source : Architecture des ordinateurs Note de cours « T.Dumartin »

4.5.2 Caractéristiques d'une mémoire

A. La capacité : c'est le nombre total de bits que contient la mémoire. Elle s'exprime aussi souvent en octet.

Capacité d'une mémoire = $2^{\text{Buse d'adresse (bits)}}$ x Buse de donnée (bits)

La capacité peut s'exprimer en :

- **Bit** : un bit est l'élément de base pour la représentation de l'information.
- **Octet** : 1 Octet = 8 bits
- **1koctet** = 1Ko = 1x2^{10}octets = 1x1024 octet = 1024 x 8 bits.
- **1Mega Octet**= 1Mo = 1024ko
- **1Giga Octet**=1Go = 1024Mo
- **1Tera Octet**= 1To = 1024 Go

Exemple : Une mémoire qui possède un bus d'adresse égale à 32, et un bus de donnée égale à 16. A une capacité égale :

Capacité = 2^{32} x 16 bits = 4294967296 x 16 bits = 4294967296 x 2 (octets)

= 8589934592 (octets)

= 8388608 Ko

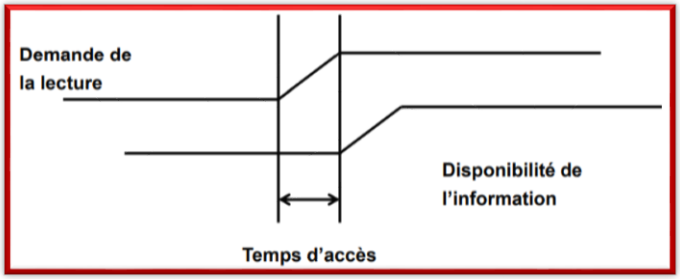
= 8192 Mo

= 8 Go

B. Le format des données (bus de données): c'est le nombre de bits que l'on peut mémoriser par case mémoire. On dit aussi que c'est la largeur du mot mémorisable. (mot de 8, 16, 32 ou 64 bits).

C. **Le temps d'accès :** c'est le temps qui s'écoule entre l'instant où a été lancée une opération de lecture/écriture en mémoire et l'instant où la première information est disponible sur le bus de données voir « figure 128 ».

Figure128: Temps d'accès



Le temps d'accès est un critère important pour déterminer les performances d'une mémoire ainsi que les performances d'une machine.

D. **Le temps de cycle :** il représente l'intervalle minimum qui doit séparer deux demandes successives de lecture ou d'écriture.

E. **Le débit :** c'est le nombre maximum d'informations lues ou écrites par seconde.

F. **Volatilité :** elle caractérise la permanence des informations dans la mémoire. L'information stockée est volatile si elle risque d'être altérée par un défaut d'alimentation électrique et non volatile dans le cas contraire.

4.5.3 Les différents types de mémoire

4.5.3.1 Les mémoires vives (RAM : Random Access Memory)

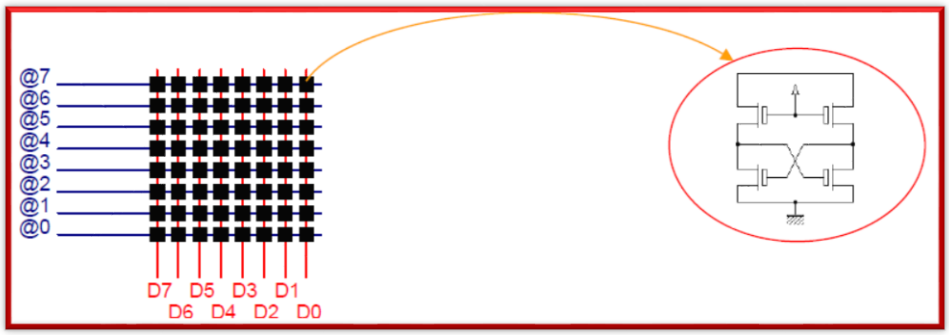
Une mémoire vive sert au stockage temporaire de données. Elle doit avoir un temps de cycle très court pour ne pas ralentir le microprocesseur. Les mémoires vives sont en général volatiles : elles perdent leurs informations en cas de coupure d'alimentation. Certaines d'entre elles, ayant une faible consommation, peuvent être rendues non volatiles par l'adjonction d'une batterie. Il existe deux grandes familles de mémoires RAM (Random Acces Memory : mémoire à accès aléatoire) :

- Les RAM statiques
- Les RAM dynamiques

A. **Les RAM statiques :** Le bit mémoire d'une RAM statique (SRAM) est composé d'une bascule. Chaque bascule contient entre 4 et 6 transistors voir figure ci-dessous.

Cette mémoire a l'immense avantage de pouvoir stocker une valeur pendant une longue période sans devoir être rafraîchie(les cellules ou sont stockés des 1 ne nécessite pas d'êtres son cesse recharger par une impulsion), Cela permet des temps d'accès très court (8−20ns). Les deux inconvénients sont son coût très élevé et son encombrement

Figure129: Représentation d'un bit mémoire d'une SRAM



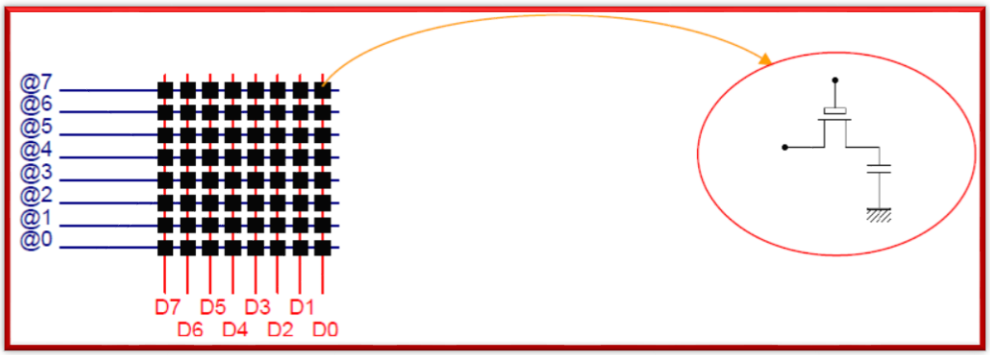
Source : Architecture des ordinateurs Note de cours « T.Dumartin »

B. Les RAM dynamiques :

Dans les RAM dynamiques (DRAM), l'information est mémorisée sous la forme d'une charge électrique stockée dans un condensateur voir figure ci-dessous..

Cette technique permet une plus grande densité d'intégration, car un point mémoire nécessite environ quatre fois moins de transistors que dans une mémoire statique voir figure xx. Sa consommation s'en retrouve donc aussi très réduite.

Figure130: Représentation d'un bit mémoire d'une DRAM



Source : Architecture des ordinateurs Note de cours « T.Dumartin »

L'inconvénient et que la présence de courants de fuite dans le condensateur contribue à sa décharge donc à l'inverse de la mémoire SRAM, la DRAM doit être rafraîchie plusieurs fois par secondes, ce qui augmente le temps d'accès (50−80ns). Par contre son coût est nettement inférieur.

4.5.3.2 Les mémoires mortes :

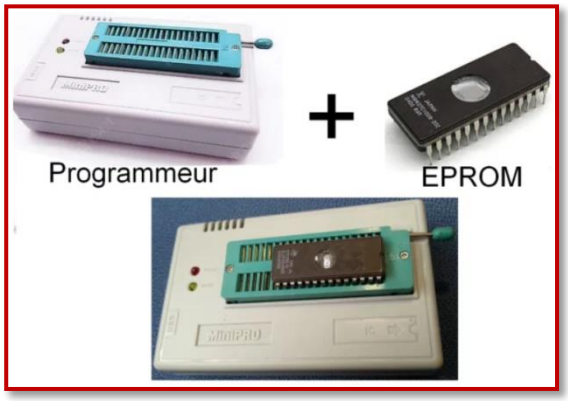
Pour certaines applications, il est nécessaire de pouvoir conserver des informations de façon permanente même lorsque l'alimentation électrique est interrompue. On utilise alors des mémoires mortes ou mémoires à lecture seule (**ROM** : *Read Only Memory*). Ces mémoires sont non volatiles, contrairement aux RAM, elles ne peuvent être que lue. L'inscription en mémoire des données restent possible à l'aide d'un programmateur de mémoires. Il existe donc plusieurs types de ROM :

A. **ROM (***Read Only Memory***):** Mémoire à lecture seul, son contenue est écrit par le fabriquent; l'utilisateur ne peut que la lire.

B. **PROM (***Programmable Read Only Memory***):** C'est une mémoire dans laquelle l'utilisateur écrie son contenue à l'aide d'un programmateur de mémoire et un logiciel de programmation voir « figure131 », après cette opération son contenue reste définitive (elle devient identique a une ROM).

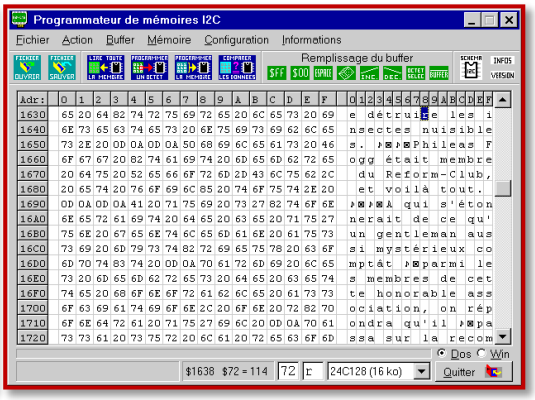
Figure131 **:** Outils de programmation d'une mémoire

(A) : Programmateur de mémoires



Source **:** https://apcpedagogie.com

(B) : Logiciel de programmation



Source : http://col2000.free.fr/index.htm#i2c

C. REPROM (*Reprogrammable Read Only memory***):**Le continue de cette mémoire peut être efface et reprogrammé à l'aide toujours d'un programmateur de mémoire, dans ce type de mémoire on trouve plusieurs :

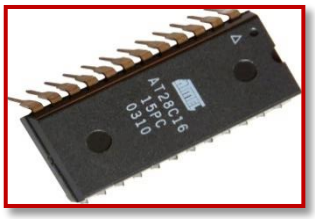
❖ **EPROM (***Erasable Programmable ROM***) :** Mémoire effaçable programmable a lecture seul, c'est une mémoire dans laquelle le contenue peut être effacé à l'aide de rayons ultraviolet, ce qui permet une nouvelle programmation, voir « figure 132 »

Figure132 : Mémoire EPROM 2048 ko 256 x 8

Source : https://www.conrad.fr

❖ **EEPROM, (***Electricaly Erasable PROM***):**
Mémoire programmable effaçable électriquement, c'est une mémoire dans laquelle son contenue peut être effacé à l'aide d'un phénomène électrique appelé avalanche, ce qui permet une nouvelle programmation à l'aide d'un programmateur de mémoire. Voir « figure 133 »

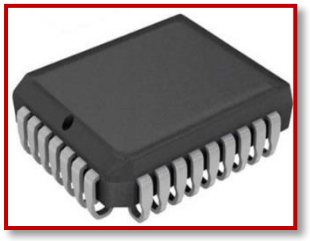
Figure133 : Mémoire EEPROM 16 ko 2 K x 8



Source : https://www.conrad.fr

❖ **Flash EPROM (FEPROM)**
La Flash EPROM plus souvent appelée mémoire **Flash** est un modèle de mémoire effaçable électriquement. Les opérations d'effacement et d'écriture sont plus rapides qu'avec les anciennes EEPROM. C'est ce qui justifie l'appellation "Flash". Cette mémoire, comme les autres ROM, conserve les données même quand elle n'est plus sous tension. Ce qui en fait le composant mémoire amovible idéal pour les appareils photos numériques, les GSM, les PDA et l'informatique embarquée. Voir « figure 134 »

Figure134 : Mémoire Flash 1024 ko (128 K x 8)

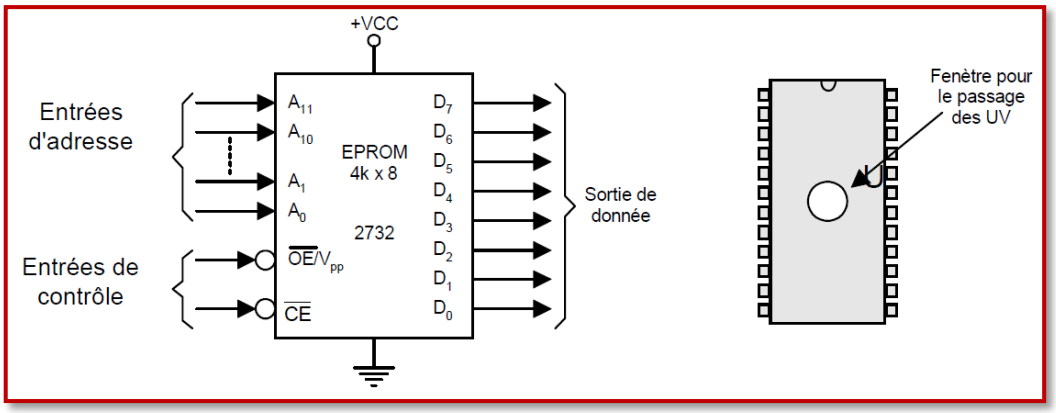


Source : https://picclick.fr

4.5.3.3 Exemples de mémoires mortes

La mémoire morte EPROM 4k x 8 (2732) voir « figure 135 » présente 12 entrées d'adresse, 8 bornes de sortie et deux entrées de contrôle. C'est l'entrée de validation de la puce que l'on utilise aussi pour mettre la puce en mode de réduction de consommation (attente). L'autre entrée de contrôle est OE / Vpp dont le rôle varie selon le mode opératoire de l'élément. Vpp est une tension de programmation spéciale que l'on applique quand on programme la puce. Cette mémoire a plusieurs modes opératoires qui sont activés selon la combinaison des tensions sur CE et sur OE / Vpp voir « Tableau 136 ».

Figure135 : Mémoire morte EPROM 4k x 8 (2732)



Source : http://module01-ofppt.blogspot.com

Tableau 136 : Modes opératoires de l'EPROM 2732

Mode	Entrées		Sorties
	$\overline{CE}$	$\overline{OE}/V_{PP}$	
Lecture vérification	V_{IL}	V_{IL}	Données $_{SORTIE}$
Invalidation sortie	V_{IH}	V_{IH}	Haute-Z
Attente	V_{IH}	X	Haute-Z
Programmation	V_{IL}	V_{PP}	Données $_{ENTREE}$

Note:
VIL= bas TTL
VIH = haut TTL
X = indifférent
Vpp = nominal 21 V

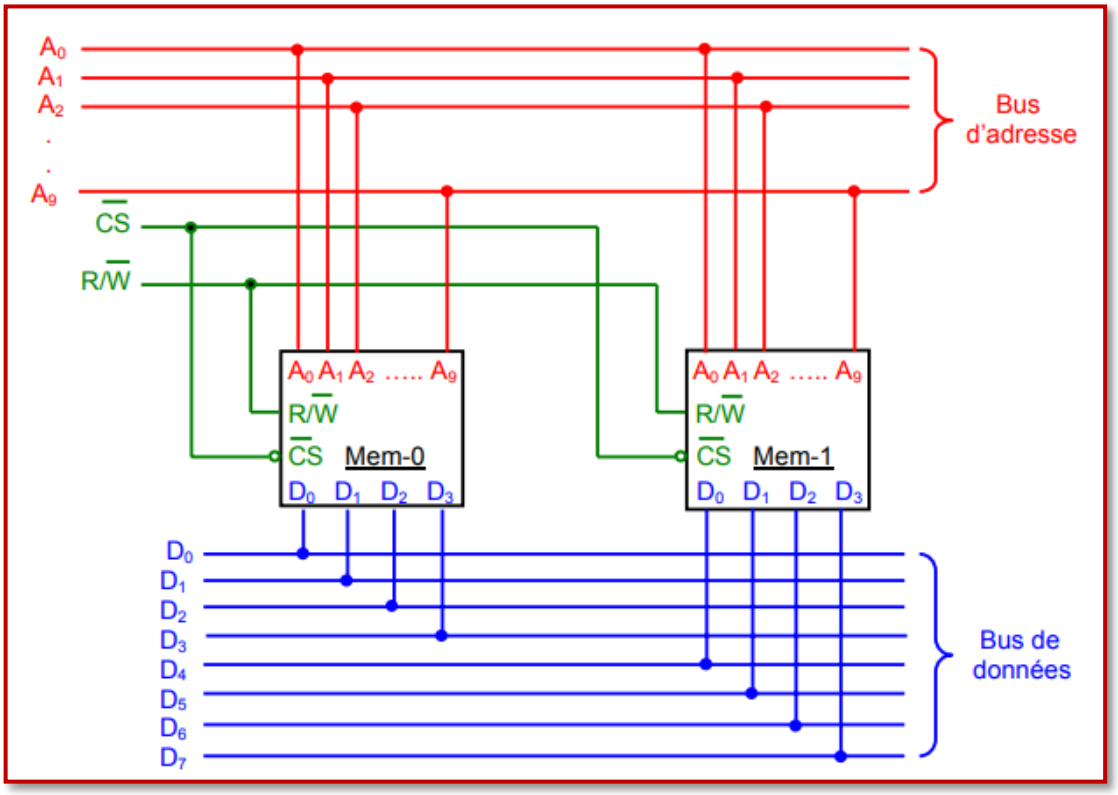
Source : http://module01-ofppt.blogspot.com

4.5.4 Extension de la capacité des mémoires

A. Extension de la longueur du mot.

Ce type d'extension sert à augmenter le nombre de bits qui constitue le mot mémoire. En montant en parallèle toutes les entrées d'adresses et de commande, et en laissant séparées les broches de données, il est possible d'atteindre l'extension requise. A titre d'exemple, supposons que l'on veuille obtenir une mémoire de 1 K x 8 bits à partir de mémoires de 1k x 4 bits. Il est possible d'assembler deux de ces mémoires de 1K x 4 bits pour obtenir la mémoire recherchée. La figure ci-dessous montre comment on peut assembler les deux mémoires.

Figure137 : Assemblage de deux mémoires 1k x 4bits pour constituer une mémoire 1K x 8bits



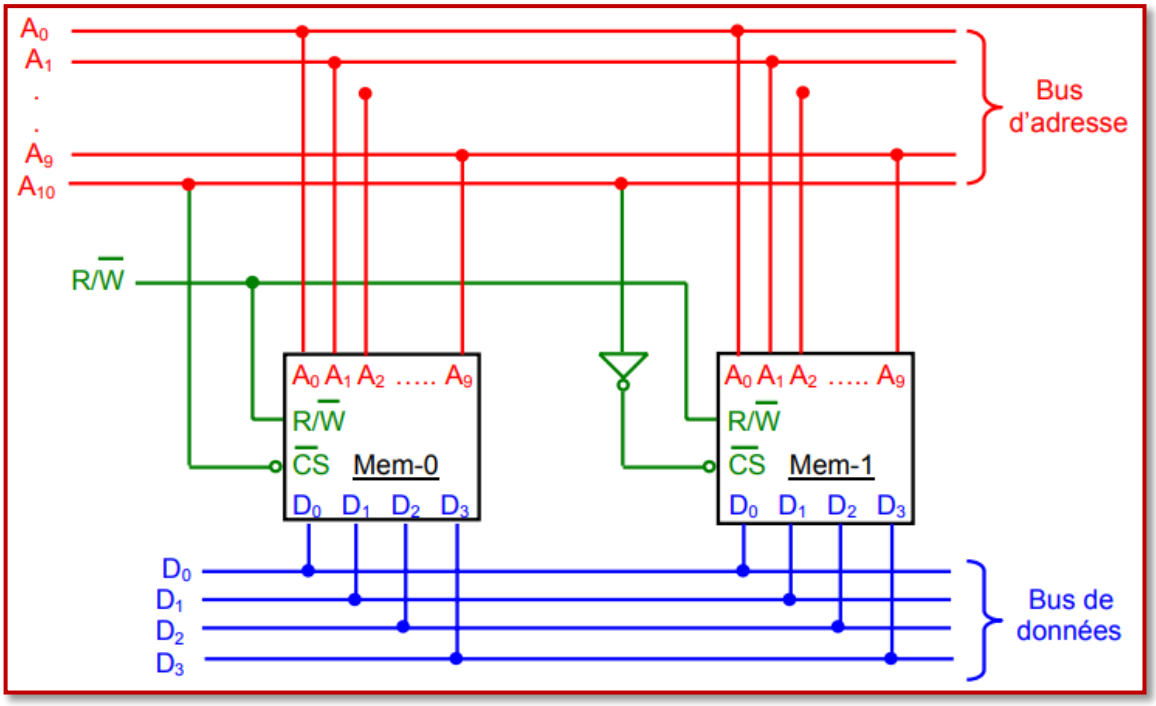
Source : Circuits logiques séquentiels (Mémoires) « TRABELSI Hichem » Université Virtuelle de Tunis

La mémoire **Mem-0** mémorise les 4 bits de rang inférieur de chacun des 1024 mots (1K) et la mémoire **Mem-1** mémorise les 4 bits de rang supérieur. Un mot de 8 bits se trouve alors sur le bus de donnée de la mémoire. L'assemblage de deux mémoires de 1K x 4 bits est équivalent à une seule mémoire de capacité 1K x 8 bits.

B. Extension de la capacité mémoire

L'extension de la capacité mémoire est l'augmentation du nombre de mots mémorisables tout en conservant le même nombre de bits par mot. Avec l'augmentation de la capacité mémoire, le nombre d'adresses nécessaires à la sélection des mots mémorisables doit augmenter aussi. Le doublement de la capacité mémoire entraîne 1 bit supplémentaire dans l'adresse. A titre d'exemple, réalisons une mémoire de 2K x 4 bits à partir de mémoires de 1K x 4 bits. Il est possible en assemblant deux mémoires de 1K x 4 bits d'obtenir une mémoire de 2K x 4 bits, comme le montre la figure ci-dessous.

Figure138 : Assemblage de deux mémoires 1k x 4bits pour constituer une mémoire 2K x 4bits



Source : Circuits logiques séquentiels (Mémoires) « TRABELSI Hichem » Université Virtuelle de Tunis

4.5.5 Exercice 28 : Associez chaque mot de la liste suivante avec une des définitions proposées :

A	SRAM	
C	EPROM	
E	ROM	
G	Bus d'adresse	

B	EEPROM	
D	Débit	
F	Bus de donnée	
H	Mémoire morte	

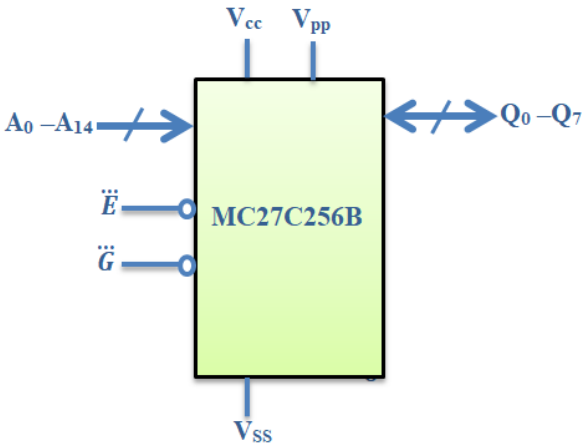
1. Mémoire programmable effaçable électriquement
2. Définit donc le nombre de cases que comprend un boîtier mémoire.
3. Mémoire interne dont le contenu peut être lu mais ne peut être modifié.
4. Mémoire effaçable par des rayons ultraviolets.
5. Mémoire volatile pouvant stocker des données pendant une longue période sans devoir être rafraîchie.
6. Nombre maximum d'informations lues ou écrites par seconde.
7. Définit la taille de l'information que l'on peut sauvegarder dans chaque case mémoire.
8. Permet de sélectionnée le mode lecture /écriture ou de sélectionné un boitier mémoire

4.5.6 Exercice 29 :

La figure ci-dessous représente une mémoire 27C256. Déterminer :

1 Le nombre de fils d'adresses
2 Le nombre de fils de données
3 Le nombre d'adresses
4 L'emplacement de la première adresse
5 L'emplacement de la dernière adresse
6 Le nombre de valeurs que peut prendre la donnée

Figure 139 : Mémoire 27C256



4.5 Résumé

- Dans un circuit ***combinatoire*** l'état de sortie ne dépend que de l'état présent des entrées, par contre dans un circuit ***séquentiel***, l'état de la sortie dépend non seulement de la combinaison appliquée à l'entrée mais aussi de l'état ***précédent*** des sorties du circuit.
- Dans un circuit asynchrone la lecture de l'entrée se fait à tout instant, alors que dans un circuit synchrone elle n'est prise en compte qu'à des instants précis synchronisés par une fréquence d'horloge
- Le signal d'horloge est généralement un signal carré qui est habituellement distribué à tous les étages du système
- Une bascule est une mémoire élémentaire qui ne peut mémoriser qu'un seul bit.
- Dans une bascule "***RS***", l'entrée ***S*** permet de mettre la sortie ***Qn*** à l'état logique 1(***Qn=1***) alors que l'entrée **R** la remet à l'état logique 0 (***Qn=0***), si ***R=S=0*** la bascule mémorise l'état précèdent (***Qn= Qn-1***), c'est une bascule qui possède un état indéterminé qui est ***R=S=1*** et qui la rend imprévisible.
- La bascule **D** (Verrou) joue le rôle de mémoire élémentaire, le signale d'horloge représentant le signale de validation ***V=0*** la bascule mémorise l'état précèdent, ***V=1*** la bascule recopie l'entrée ***D*** à la sortie.
- Dans une bascule "***JK***", l'entrée ***J*** permet de mettre la sortie ***Qn*** à l'état logique 1**(*Qn=1*)** alors que l'entrée ***K*** la remet à l'état logique 0 **(*Qn=0*)**, ***J=K=0*** la bascule mémorise l'état précèdent (***Qn=Qn-1***), **J=K=1** la bascule inverse l'état précèdent de sa sortie (**Qn=$\overline{Q}_{n-1}$**)
- La bascule **T** comprend deux entrées « ***T*** » et « ***H*** » le signale d'horloge ***H*** représentant le signale de validation ***H=0*** la bascule mémorise l'état précèdent, si ***H=1*** et ***T=1*** la bascule recopie à sa sortie l'inverse de la sortie précédente (**Qn=$\overline{Q}_{n-1}$**)
- la bascule ***T***est connue spécialement pour son état de complémentation qui donne au circuit la fonction de ***diviseur de fréquence (d'horloge) par deux***.
- Un compteur est une succession de bascule ***JK*** ou ***T*** dans le mot binaire se modifie à chaque impulsion d'horloge
- Dans un compteur synchrone, toute les bascules possède le même signal d'horloge, alors que dans un compteur synchrone l'entrée d'horloge de chaque bascule vient de l'une des sorties de la bascule précédente
- Un registre à décalage est un assemblage de bascules commandées par une horloge commune.
- Dans un registre à décalage, une donnée introduite à l'entrée de la première bascule se propage dans les bascules suivantes à chaque signal de l'horloge.
- Une mémoire est un circuit à semi-conducteur permettant d'enregistrer, de conserver et de restituer des informations, elle est constituée de trois bus (bus de données, bus d'adresses et bus de commandes)

4.6 Exercices de synthèse 4

- **Spécialité :** Réseaux Télécom Filaires
- **Module :** Electronique numérique
- **Durée :** 2h
- **But :** Est de se familiariser avec les circuits séquentiels (bascules, compteurs et registres) et de les utiliser dans des exemples d'application.
- **Matériel requis** :
  1. Plaque d'essaie
  2. Fils de connections
  3. Afficheur 7 segments Anode commune
  4. Circuits intégrés décodeur 7 segments 74LS48
  5. Résistance de 470Ω
  6. Alimentation stabilisé
  7. Circuits intégrés 74LS00
  8. Logiciel de simulation électronique
  9. Support de cours
- **Mise en situation :** Câbler différents circuits séquentiels et vérifier leurs fonctionnements
- **Marche à suivre :**

I. Bascule JK :

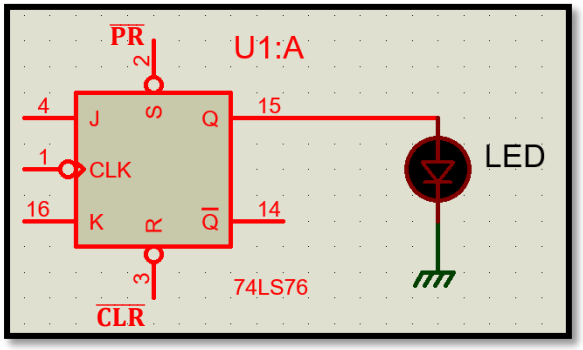
Le circuit intégré 7476, voir (figure 124) est très utilisée, il possède deux bascules Flip-Flop active sur un front montant, elles possèdent chacune deux entrées de forçage à 1 et à 0,$\overline{\mathbf{PRE}}$ et $\overline{\mathbf{CLR}}$ et les entrées J, K et l'entrée du signal d'horloge CLK.

Figure 124 : Circuit intégré 7476

Source : https://xenyltechbd.com

1. Réaliser le circuit de la (figure 125), utiliser pour cela la 1er bascule du 74LS76, voir brochage (annexe 2).

Figure 125 : La 1er Bascule JK du CI74LS76



Q=1 : LED allumée
Q=0: LED éteinte
$\overline{\mathbf{PR}}$**(S)** : Réalise une mise à 1 (Q=1)
$\overline{\boldsymbol{CLR}}$ ***(R)***: Réalise une remise à 0 (Q=0)

2. Vérifier la fonction de chacune des entrées $\overline{\mathbf{PRE}}$ et $\overline{\mathbf{CLR}}$ en regardant la sortie Q=f($\overline{\mathbf{PRE}}$, $\overline{\mathbf{CLR}}$)
3. En positionnant les entrées $\overline{\mathbf{PRE}}$ et $\overline{\mathbf{CLR}}$ à 1 et en reliant le module horloge (Générateur de signaux carrés GBF avec une fréquence de 1Hz) à l'entrée CLK , déterminer les fonctions des entrées J et K.
   Faire une table de vérité. Conclusion ?

II. Compteur binaire :

1. On utilisant des circuits intégrer de type 74LS76, voir brochage (annexe 2) réaliser un compteur asynchrone modulo 10, tester son fonctionnement on utilisant des LEDs relier à chaque sortie d'une bascule JK , fréquence de l'horloge 1Hz (Dresser pour cela une table de vérité)
2. On utilisant un décodeur 7 segment de type 7447 voir brochage (annexe 2) relier le compteur précédemment réaliser à un afficheur 7 segments, puis tester le fonctionnement (Dresser pour cela une table de vérité)

RÈSUMÈ GENÈRAL

On vient de terminer l'étude sur la logique combinatoire et la logique séquentielle. Il apparaît intéressant de récapituler les points suivants.

Dans le 1er chapitre on a constaté qu'on plus des systèmes de numérotation décimale, il existe trois autres systèmes de numérotions couramment utilisés dans les dispositifs industriels, ce sont le système binaire, le système octal, et le système hexadécimal. Ces trois systèmes sont des systèmes pondérés où le poids de pondération de chaque chiffre dépend de son rang dans le nombre, nous avons passé aussi en revue les codes numériques les plus fréquemment utilisés et avons appris à les manipuler.

Concernant le chapitre 2 nous avons fait connaissance avec les notions de base de l'algèbre de Boole : variables et fonctions Booléennes, opérations booléennes de base, table de vérité et propriétés des opérations logiques de base. Cette activité nous a permis de mettre en revue les bases de l'algèbre logique utilisée dans les circuits numériques, ces connaissances nous ont permis d'effectuer la synthèse des circuits logiques qui sont les composants de bases des systèmes numériques , on a aussi appris à faire la synthèse des fonctions logiques à partir de leurs tables de vérité et leurs expressions analytiques, à simplifier les expressions booléennes en utilisant la méthode algébrique ou la méthode de Karnaugh.

Dans cette partie nous avons abordé brièvement la famille TTL et la famille CMOS. Nous avons vu leurs compositions, leurs caractéristiques de base, leurs avantages et leurs inconvénients. Ce qui va nous permettre de choisir la technologie la plus adaptée aux systèmes qu'on veut réaliser.

Le chapitre 3, nous a permis de connaître la structure et le fonctionnement des circuits de bases utiliser dans le traitement des systèmes numériques que sont les décodeurs, multiplexeurs, additionneurs, comparateurs. Elle a permis aussi de connaître la méthodologie de conception des circuits logiques de ces dispositifs.

Le chapitre 4, nous a permis d'étudier la structure et le fonctionnement des circuits séquentiels de base utiliser dans le traitement des systèmes numériques séquentiels que sont les bascules, registres , compteurs et les mémoires. Elle a permis aussi de connaître la méthodologie de conception des circuits logiques de ces dispositifs.

ACTIVITÉ DE SYNTHÈSE

- **Spécialité :** Réseaux Télécom Filaires
- **Module :** Electronique numérique
- **Durée :** 6h
- **But :** Analyser les circuits logiques combinatoires et séquentiels.
- **Matériel requis** : Documents, supports de cours, simulateur électronique.
- **Mise en situation :** Afin de bien comprendre et de manipuler les circuits numérique simple, ils vous aient demandé de réaliser le circuit de commande d'un distributeur de boisson, un additionneur Soustracteur et un compteur asynchrone
- **Marche à suivre :**

Première partie

1. Comprendre le principe de fonctionnement du distributeur de boisson
2. Déduire et s'simplifier ces fonctions
3. Réaliser le circuit de commande du distributeur

Deuxième partie

1. Réaliser un demi-additionneur
2. Réaliser un soustracteur complet
3. Réaliser un additionneur soustracteur

Troisième partie

1. Réaliser un compteur asynchrone modulo 16
2. Modifier le compter précèdent afin d'obtenir un compteur modulo 12

I. Algèbre de Boole

Distributeur de boissons chaudes

- ✓ Un distributeur de boissons chaudes permet de distribuer du café ou du thé, avec ou sans lait, ou du lait seul.
- ✓ Trois boutons permettent de commander le distributeur : « café », « thé », « lait ». Pour obtenir l'une de ces boissons seule, il suffit d'appuyer sur le bouton correspondant. Pour obtenir une boisson avec lait, il faut appuyer en même temps sur le bouton correspondant à la boisson choisie et sur le bouton « lait ».
- ✓ De plus, le distributeur ne fonctionne que si un jeton a préalablement été introduit dans la fente de l'appareil. Une fausse manœuvre après introduction du jeton (par exemple, appui simultané sur « café » et « thé ») provoque la restitution du jeton. Le lait étant gratuit, le jeton est également restitué si du lait seul est choisi.
- ✓ Calculer et simplifier les fonctions de restitution du jeton, J, de distribution du café, C, du thé T, et du lait, L. On notera que la fonction de restitution du jeton peut indifféremment être active ou non lorsque aucun jeton n'est introduit dans l'appareil
- ✓ Réaliser le circuit

II. Circuits combinatoires

Demi-Soustracteur.

Réaliser un demi-soustracteur :

1. Ecrire la table de vérité.
2. Donner les équations de sortie.
3. Etablir le schéma logique.

Soustracteur complet

On veut réaliser un circuit qui effectue la soustraction Ai - Bi en tenant compte d'une éventuelle retenue R i-1. Ce circuit doit donc générer la différence Di et l'éventuelle retenue Ri à transmettre à la colonne de gauche.

1. Remplir la table de vérité de Di et Ri.
2. Remplir les tableaux de Karnaugh et en déduire les équations simplifiées de Di et Ri.
3. Dessiner le schéma de ces deux fonctions réunies en un seul bloc fonctionnel : le soustracteur complet.
4. Réaliser un soustracteur binaire complet (ou étage de soustracteur) selon deux modes :
   a. Avec deux demi-soustracteurs ;
   b. Avec un demi-additionneur et un demi-soustracteur.
5. Dessiner le schéma d'un soustracteur de 2 nombres de 4 bits en utilisant 4 blocs fonctionnels identiques.

Additionneur Soustracteur

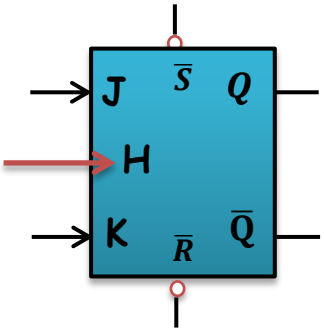
1. Réaliser un circuit qui inverse ou non l'état d'une entrée E selon qu'un bit de commande Cest à 1 ou à 0: si C= 0 on veut S= E, si C= 1 on veut S = $\overline{E}$
2. En utilisant cette fonction et un additionneur sur 4 bits, réaliser un circuit qui effectue l'addition de deux nombres de 4 bits (A + B) si un bit de commande C est à 0 et la soustraction (A – B) si C = 1.

III. Circuits séquentiels

On dispose de bascules JK synchronisées sur front montant de la (figure126).
Chaque bascule possède des entrées asynchrones prioritaires actives à l'état bas : ***set (S)*** **et** ***reset (R)***.

1. Réalisez un compteur asynchrone modulo 16.
2. Modifiez le montage pour en faire un compteur asynchrone modulo 12.
3. En partant de zéro, tracez son chronogramme sur un cycle complet.
4. Ajoutez un interrupteur automatique de remise à zéro à l'allumage.
5. Ajoutez un interrupteur manuel de remise à zéro.
6. Que suffit-il de faire pour remplacer les bascules JK par des bascules D

Figure126 : Bascule JK



BIBLIOGRAPHIE

- **BELAYACHI, Naima.** Cours et Exercices, Architecture des Ordinateurs 1.Département des Sciences Exactes, Filière d'Informatique,Ecole Normale Supérieure d'Oran (ENSO).
- **BENAMARA Mahmoud .GAALOUL**: 2015/2016. Département de Génie Electrique, Institut Supérieur des Etudes, Technologiques de Nabeul.
- **BENAMARA Mahmoud .GAALOUL Kamel.** Support de cours, Systèmes logiques 2, Logique séquentielle. Année universitaire: 2015/2016. Département de Génie Electrique, Institut Supérieur des Etudes, Technologiques de Nabeul.
- **CARIOU**, **Éric**. Cours, Algèbre de Boole, P29. Université de Pau et des Pays de l'Adour, Département Informatique.
- **CHRISTOPHE, Adessi.** Cours Electronique Numérique 2016-2017, Audioprothèse 2eme Année. Université Claude Bernard Lyon1.
- **C,Belleudy . D, Gaff.** Cours Premier semestre. Electronique Numérique. Université de Nice-Sophia, Antipolis, DEUG Première année.
- **DIOU, Camille**. Maître de Conférences, Cours, P187.D'électronique numérique. Laboratoire Interfaces Capteurs et Microélectronique. Université Paul Verlaine–Metz.
- **DOUILLARD, Catherine. THEPAUT, André.** Polycopié 1/2. Logique combinatoire et circuits MOS. **Année scolaire 2008-2009.**Telecom Bretagne, France
- **GUILLAUME, Blin.** Cours, Algèbre de Boole et circuits logiques, Architecture des ordinateurs. Université de Marne La Vallée, France
- **JEZEQUEL, Michel.** Cours 2, Circuits combinatoires. Département d'électronique .Telecom Bretagne, France.
- **KARA, Abdelaziz.** Cours, Les Circuits Combinatoires.1er année Tronc Commun Mathématique Informatique, 2018/19. Faculté des Sciences, Université Sétif 1.
- **KONATE**, **Karim.** Electronique numérique. Informatique appliquée: csi 2201. Université virtuelle africaine

- **LE BRUN, Philippe.** Cours, Electronique numérique. Lycée Louis ARMAND, Strasbourg, France.
- **M,SALMANI.** Logique combinatoire : Partie 2. Sciences et technologies électriques, Niveau 1ère Sciences de l'ingénieur Unité ATC.
- **MEZAACHE, Salah Eddine.** Architecture des ordinateurs et programmation, 2009/2010. P42.Département d'électronique, centre universitaire de bordj bouarréridj .
- **PIERRE, Mayé .**Aide-mémoire. Composants électroniques. 3e édition, L'usine Nouvelle, DUNOD.
- **TRABELSI, Hichem.** Cours, Circuits logiques combinatoires, Systèmes de numération et codes. Université virtuelle de Tunis.

- **TRABELSI, Hichem.** Cours, Circuits logiques séquentiels**,** Bascules bistables. Université Virtuelle de Tunis.
- **WEBER,Jacques . MEAUDRE, Maurice.** Circuits numériques et synthèse logique, un outil : VHDL. L'IUT de CACHAN, Université de Paris.

SITES WEB

- http://www.composelec.com/index.php « Composant électronique / Composant actif / Composant passif »
- https://www.electronique-et-informatique.fr « Bureau d'étude Electronique Informatique », Cree le 12 /06/2019 , mis à jour le 02/01/2020 .
- https://mrproof.blogspot.com«Bibliothèque de la programmation ».
- https://f2school.com/electronique-numerique/ « Bibliothèque en ligne F2School »
- https://openclassrooms.com « Se former en alternance avec Open Classrooms »
- https://www.technologuepro.com/ « Technologue pro - Ressources pédagogiques pour l'enseignement technologique »
- https://f2school.com/electronique-numerique/ « Bibliothèque en ligne F2School »Bibliothèque en ligne

ANNEXE 1

CORRIGÈS DES EXERCICES

❖ Exercice 01

Les suites de chiffres qui peuvent être la représentation d'un nombre en base 16, 8 ou 2

	Binaire (Base 2)	Octal (Base 8)	Hexadécimal (Base 16)
1001010210	Faux	Vrai	Vrai
1001011	Vrai	Vrai	Vrai
1200111201	Faux	Vrai	Vrai
1431901	Faux	Faux	Vrai
2A21F	Faux	Faux	Vrai
9GF28	Faux	Faux	Faux
1789012	Faux	Faux	Vrai

❖ Exercice 02

Base	Nombre	Equivalent	Dans la base
10	45	101101	2
10	57	71	8
8	75	3D	16
2	1101,101	13,5	10
10	42,0625	52,04	8
16	ABC, A	5274,50	8
4	1320	78	16

❖ Exercice 03 les opérations en binaire sur 8 bit

A. $(9)_{10} + (8)_{10}$

$$\begin{array}{rl} & 0\ 0\ 0\ {}_1 0\ 1\ \ 00\ 1 \\ + & 0\ 0\ 0\ \ 01\ \ 0\ 0\ 0 \\ \hline = & \ \ 00\ \ 010\ \ 0\ \ 0\ 1 \end{array} \qquad \begin{array}{l} (9)_{10} \\ +(8)_{10} \\ \hline = (17)_{10} \end{array}$$

B. $(57)_{10} + (31)_{10}$

$$\begin{array}{rl} & 0_1 0\ \ {}_1 1_1 1_1 1\ \ {}_1 0_1 0\ \ 1 \\ + & 0\ 0\ 0\ \ 11\ \ 1\ \ 11 \\ \hline = & \ \ 0\ \ 1\ \ 01\ \ 1\ 0\ \ 0\ 0 \end{array} \qquad \begin{array}{l} (57)_{10} \\ + (31)_{10} \\ \hline = (88)_{10} \end{array}$$

C. $(45)_{10}$ - $(41)_{10}$

```
  0 010 1 10 1          (45)10
- 0 0 1 01 0 0 1        _(41)10
-----------------
= 0 0 0 0 0 1 0 0       = (04)10
```

D. $(56)_{10}$ - $(4)_{10}$

```
  0 01 11 10 00         (56)10
- 0 0 0 0101 0 0        _ (04)10
-----------------
= 0 0 1 1 0 1 0 0       = (52)10
```

E. $(84)_{10}$ ÷ $(5)_{10}$

```
  0 1 0 1 0 1 0 1 | 1 0 0
- 1 0 0 ↓ ↓ ↓ ↓   | 1 0 1 0 1 ,0 1
= 0 0 0 1 0 1 ↓ ↓
    - 1 0 0 ↓ ↓
    = 0 0 1 0 1
        - 1 0 0
        = 0 0 1 0 0
            - 1 0 0
            = 0 0 0
```

F. $(25,25)_{10}$ x $(4)_{10}$

```
    2 5 ,2 5
x           4
-------------
= 1 0 1 ,0 0
```

```
      1 1 0 0 1 ,0 1
    x           1 0 0
-----------------------
      0 0 0 0 0 0 0
    0 0 0 0 0 0 0 .
1 1 0 0 1 0 1 . .
-----------------------
1 1 0 0 1 0 1 ,0 0
```

❖ Exercice 04

1. Conversions

Base10	Binaire signé Cà2
-125	10000011
-21	11101011
+98	1100010
-1	11111111

2. Les soustractions en binaire sur 8 bits

A. La méthode directe

➢ **(42) 10 – (29) 1**

$$\begin{array}{r} 0\ 01_1 0_1 1\ _1 0\ \ 1_1 0 \\ -\ \ 0\ \ 0\ \ _1 0_1 1\ _1 11\ _1 0\ \ 1 \\ \hline = \quad 0\,0\quad 0\quad 0\quad 1\quad 1\quad 0\quad 1 \end{array} \qquad \begin{array}{r} (42)_{10} \\ \_(29)_{10} \\ \hline = (13)_{10} \end{array}$$

➢ **(25) 10 – (9) 1**

$$\begin{array}{r} 0\ 00\ 110\ 0\ 1 \\ -\ \ 0\ \ 0\ \ \ 0\,010\ 0\ \ 1 \\ \hline = \quad 0\,0\quad 0\quad 1\quad 0\,0\quad 0\,0 \end{array} \qquad \begin{array}{r} (25)_{10} \\ \_(09)_{10} \\ \hline = (16)_{10} \end{array}$$

➢ **(12) 10 – (52)**

$$\begin{array}{r} {}_1 0_1 0_1 0_1 011\ \ 00 \\ -{}_1 0\ \ {}_1 0\ \ {}_1 1\ 101\ 0\ \ 0 \\ \hline = \mathbf{1}\ \ 1\quad 1\quad 0\quad 1\quad 1\,0\quad 0\,0 \end{array} \qquad \begin{array}{r} (12)_{10} \\ \_(52)_{10} \\ \hline = (-40)_{10} \end{array}$$

➢ **Remarque** :

- Le résultat est sur 8bits, c'est-à-dire que le 9ieme bit n'est pas pris en charge
- Le résultat est négatif, pour connaitre le nombre on doit faire le Cà2 du résultat

  Cà2 $(11011000)_2 = (00101000)_2 = (40)_{10}$

  $\Rightarrow (11011000)_2 = (-40)_{10}$

B. La méthode qui utilise le complément à 2

➢ **(42) 10 – (29) 10** = (42) 10+ (–29)10

(42) 10 =(00101010)2

(29)10 = (00011101)2

(–29)10 = Cà2(00011101)2 = (11100011)2

(42) 10+ (–29)10 =(00101010)2 + (11100011)2

$_1 0_1 0101\ _1 0\ \ 10$ $(42)_{10}$

\+ 1 1 10 0 0 11 _(29)10

= 1 0 0 0 0 1 1 0 1 = (13)10

➢ **(25) 10 – (9) 10** = (25) 10+ (–9)10

(25) 10 =(00011001)2

(9)10= (00001001)2

(-9)10= Cà2 (00001001)2 = (11110111)2

(25) 10+ (–9)10 = (00011001)2 + (11110111)2

$_1 0_1 0\ _1 0_1 1_1 1_1 0_1 0\ 1$ $(25)_{10}$

\+ 1 1 1 10 1 11 _(09)10

= 1 0 0 0 1 0 0 0 0 = (16)10

➢ **(12) 10 – (52) 10**

(12) 10 =(00001100)2

(52) 10= (00110100)2

(-52) 10= Cà2 (00110100)2 = (11001100)2

(12) 10 + (–52) 10 = (00001100)2 + (11001100)2

$_1 0_1 0\ 0_1 0_1 11\ \ 00$ $(12)_{10}$

\+ 1 $_1$1 0 011 0 0 _(52)10

= 1 1 1 0 1 1 0 0 0 = (- 40)10

❖ Exercice 05

Code/Base	Nombre	Equivalent	Dans le Base/Code
2	111011101	100110011	Gray
10	7	1010	XS3
2	110010	0101 0000	DCB
8	17	001000	Gray
Gray	110010001010	100011110011	2
Gray	1011	1110	XS3

❖ Exercice de synthèse1

A.

Base10	Base 2	Base 8	Base16
125	1111101	175	7D
61645	1111000011001101	170315	F0CD
11,75	1011,11	13,6	B,C
85,5	1010101,100	125,4	55,8

B.

Base	Nombre	Equivalent	Dans la base
10	121,875	1111001,111	2
16	A01	0010 0101 0110 0001	DCB
DCB	10010110	1010000	Gray
8	25,45	15,94	16

C.

➢ Complément d'un nombre binaire

Nombre	Complément à 1	Complément à 2
A=10000100	0111 1011	0111 1100
B=01110001	1000 1110	1000 1111

➢ Operations en binaire sur 8 bits

A+B	1111 0101
A-B	0001 0011
B-A	1110 1101
-A-B	0000 1011

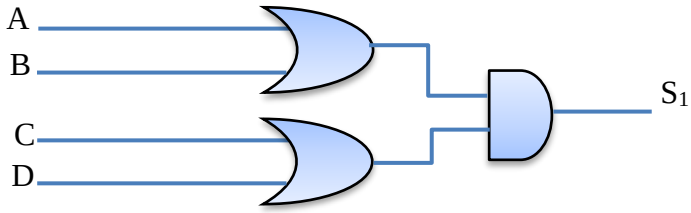
❖ **Exercice 06** Table de vérité du circuit électronique de la figure 18

➢ Table de vérité

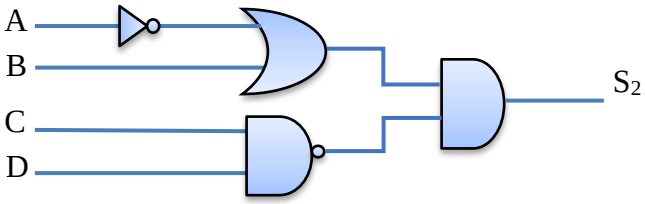
K_1	K_2	L
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

❖ **Exercice 07**Circuits logiques des fonctions S_1 et S_2

S1 = (A + B). (C + D)



S2 = $(\bar{A}$ + B). $\overline{C.D}$

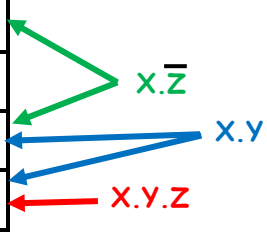


❖ **Exercice 8**:

A. La table de vérité de la fonction :F1 = X.Y.Z +X .(Y + $\bar{Z}$) = X.Y.Z + X.Y + X.$\bar{Z}$

➢ Table de vérité

X	Y	Z	F
0	0	0	**0**
0	0	1	**0**
0	1	0	**0**
0	1	1	**0**
1	0	0	**1**
1	0	1	**0**
1	1	0	**1**
1	1	1	**1**



B. La deuxième forme canonique de F_1

Table de vérité

X	Y	Z	F	
0	0	0	**0**	→ $X+Y+Z$
0	0	1	**0**	→ $X+Y+\overline{Z}$
0	1	0	**0**	→ $X+\overline{Y}+Z$
0	1	1	**0**	→ $X+\overline{Y}+\overline{Z}$
1	0	0	**1**	
1	0	1	**0**	→ $\overline{X}+Y+\overline{Z}$
1	1	0	**1**	
1	1	1	**1**	

$$F=(X+Y+Z)\ +(X+Y+\overline{Z})+(X+\overline{Y}+Z)+(X+\overline{Y}+\overline{Z})\ +(\overline{X}+Y+\overline{Z})$$

❖ Exercice 09

Table de vérité de la fonction $F_1 = (X+Y).(\bar{X}+Y+Z)$

X	Y	Z	F_1	
0	0	0	**0**	X+Y
0	0	1	**0**	
0	1	0	1	→ $\overline{X}.Y.\overline{Z}$
0	1	1	1	→ $\overline{X}.Y.Z$
1	0	0	**0**	$\bar{X}+Y+Z$
1	0	1	1	→ $X.\overline{Y}.Z$
1	1	0	1	→ $X.Y.\overline{Z}$
1	1	1	1	→ $X.Y.Z$

F_1sous la 1er forme canonique$F1 = \bar{X}.Y.\bar{Z} + \bar{X}.Y.Z + X.\bar{Y}.Z + X.Y.\bar{Z} + X.Y.Z$

❖ Exercice 10

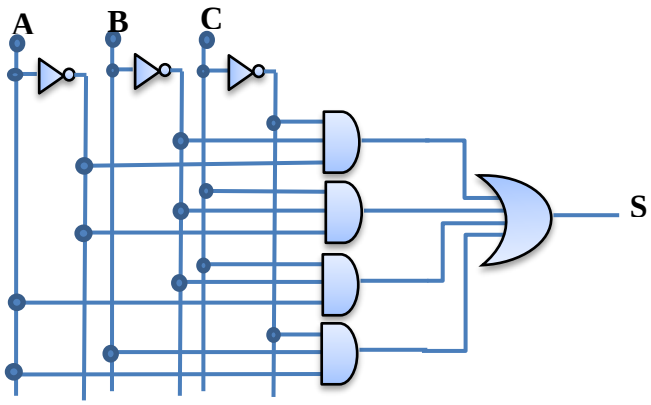
a) Table de vérité

A	B	C	S	
0	0	0	**1**	→ $\overline{A}.\overline{B}.\overline{C}$
0	0	1	**1**	→ $\overline{A}.\overline{B}.C$
0	1	0	**0**	
0	1	1	**0**	
1	0	0	**0**	
1	0	1	**1**	→ $A.\overline{B}.C$
1	1	0	**1**	→ $A.B.\overline{C}$
1	1	1	**0**	

b) La fonction S

$$S=\overline{A}.\overline{B}.\overline{C}\ +\overline{A}.\overline{B}.C+A.\overline{B}.C+A.B.\overline{C}$$

c) Circuit logique de S

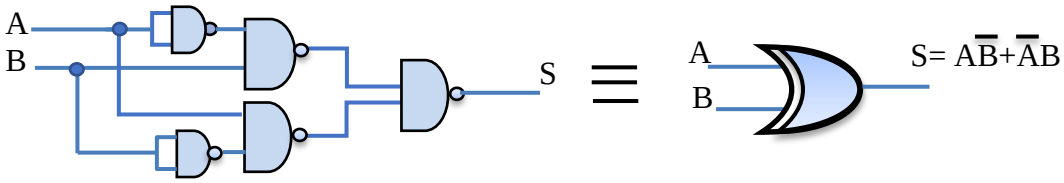


❖ **Exercice 11**:

✓ Equation de la porte EXOR S= A.$\overline{B}$+$\overline{A}$.B

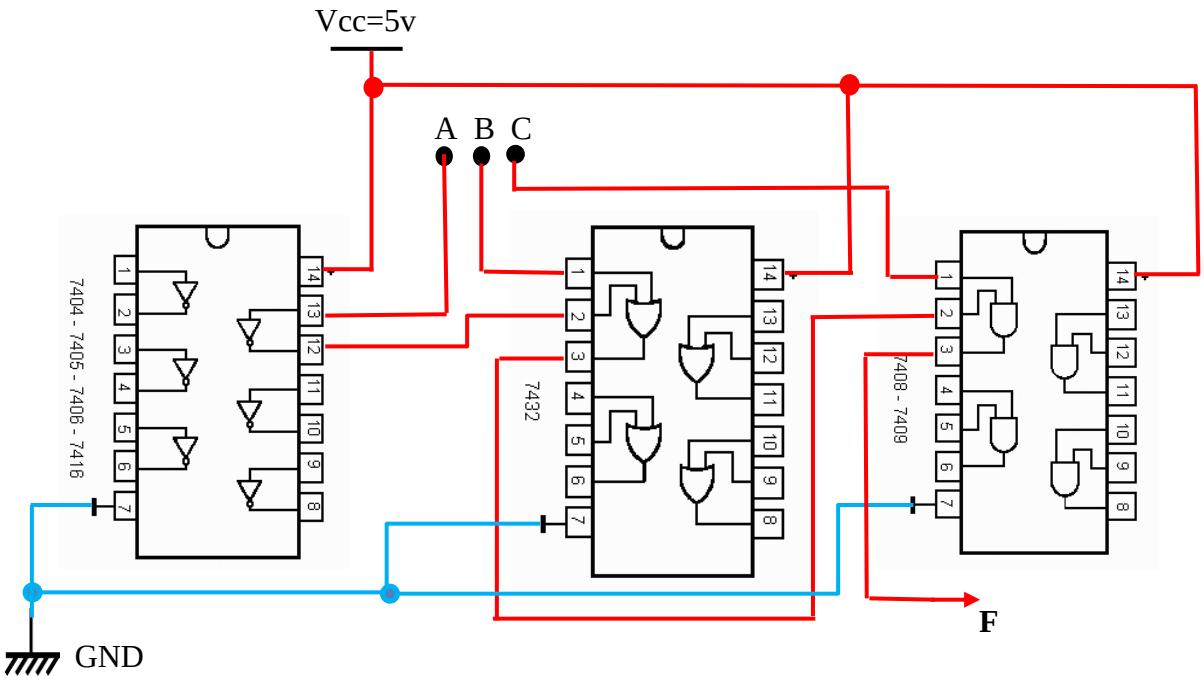
S= $\overline{A}\overline{B}$+$\overline{A}$B = $\overline{A}\overline{B}$. $\overline{A}$B

✓ Logigramme de la porte EXOR à l'aide de portes NAND



❖ **Exercice 12**

Circuit logique de la fonction F= ($\overline{\mathbf{A}}$+B).C en utilisant les circuits intégrer



❖ **Exercice 13**: Simplification algébrique de la fonction F_2

F2= $(X + Y + Z).(\bar{X}+Y+Z)+XY+YZ$

F2=$X\bar{X}+XY+XZ+Y\bar{X}+YY+YZ+Z\bar{X}+ZY+ZZ+XY+YZ$ Sachant que $X\bar{X}=0$, $YY=Y$, $ZZ=Z$

F2=$XY+XZ+Y\bar{X}+Y+YZ+Z\bar{X}+ZY+Z+XY+YZ$ Sachant que $YZ+ZY+YZ=YZ$, $XY+XY=XY$

F2=$XY+XZ+Y\bar{X}+Y+YZ+Z\bar{X}+Z$

F2=$Y(X+\bar{X}+1+Z)+Z(\bar{X}+1)$ Sachant que $(X+\bar{X}+1+Z)=1$, $(X+1)=1$

F2 = Y + Z

❖ **Exercice 14** :

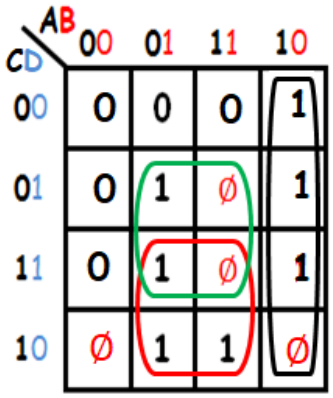
$F(A, B,C,D) = A\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,B\overline{C}\,D + A\overline{B}\,CD + A\overline{B}\,\overline{C}\,D + \overline{A}\,BC\,D + \overline{A}\,BC\overline{D} + ABC\overline{D}$

A. Simplification de F par la méthode des diagrammes de Karnaugh, sachant que quatre combinaisons de variables sont impossibles : $AB\overline{C}D$, $ABCD$, $\overline{A}\,B\overline{C}\overline{D}$, et $A\overline{B}\,C\overline{D}$

❖ Table de vérité

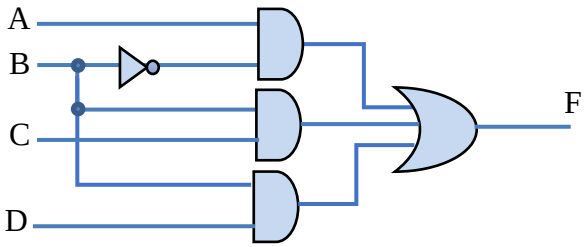
	A	B	C	D	F
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	Ø
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	Ø
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	Ø
14	1	1	1	0	1
15	1	1	1	1	Ø

B. Tableau de KARNAUGH



F= $A\overline{B}$ + BD + BC

B. Schéma logique ou logigramme de la fonction simplifiée

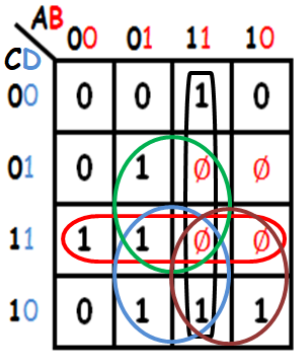


❖ Exercice 15

A. Table de vérité

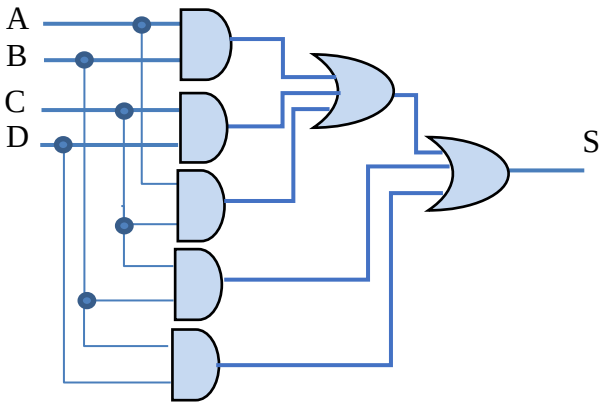
	A	B	C	D	S
0	0	0	0	0	**0**
1	0	0	0	1	**0**
2	0	0	1	0	**0**
3	0	0	1	1	**1**
4	0	1	0	0	**0**
5	0	1	0	1	**1**
6	0	1	1	0	**1**
7	0	1	1	1	**1**
8	1	0	0	0	**0**
9	**1**	0	0	**1**	∅
10	1	0	1	0	**1**
11	**1**	0	1	**1**	∅
12	1	1	0	0	**1**
13	**1**	1	0	**1**	∅
14	1	1	1	0	**1**
15	**1**	1	1	**1**	∅

B. Tableau de KARNAUGH



S= A.B + C.D + B.D + A.C + B.C

C. Circuit logique

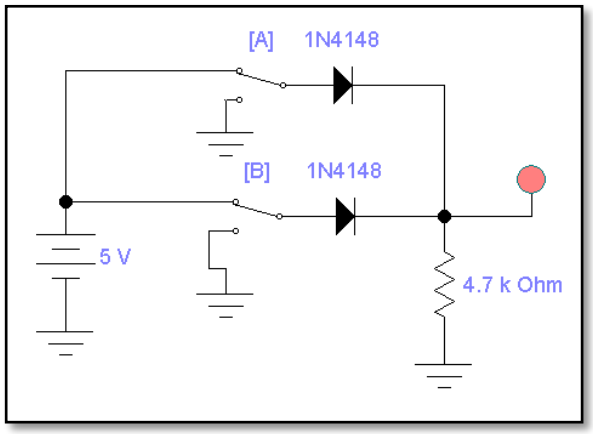


❖ Exercice de synthèse 2

A. Portes à diodes

A.1. Premier montage

A. Montage **2. Table de vérité**

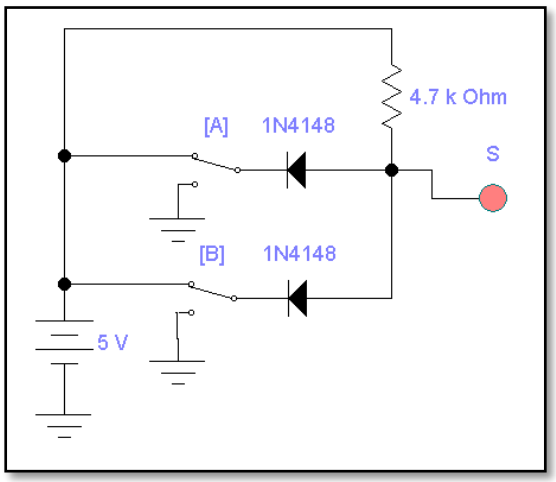


A	B	S
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

3. La fonction obtenue est : S=A OU B = A **OR** B= A+B

A.2. Deuxième montage :

1. Montage **2. Table de vérité**



A	B	S
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

3. La fonction obtenue est : S=A ET B = A **AND** B= A.B

B. Operateurs logiques

B.1. Premier montage

1. Circuit logique de **S** à base du CI 7432(4 portes OR à 2 IN)

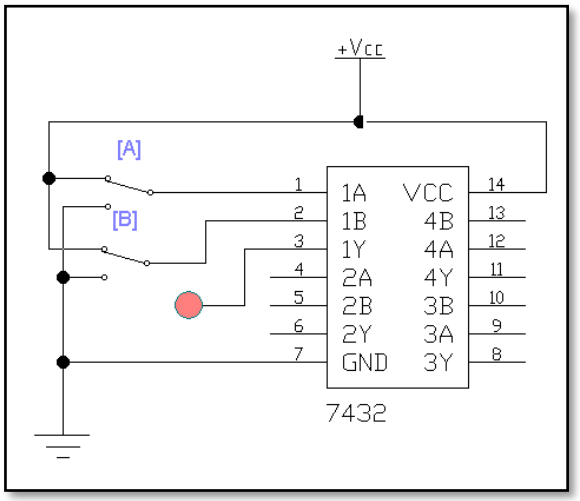
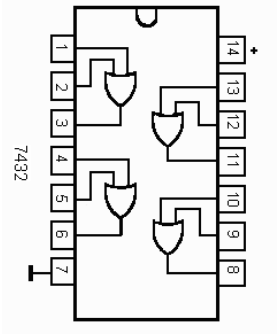


Figure 127 : Brochage du 7432



2. Table de vérité

A	B	S
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

3. L'équation **S = A + B**
Nom de la fonction : **OU (OR)**

B.2. Deuxième montage

1. Circuit logique de **S** à base du CI 74LS08 (4 portes AND à 2 IN)

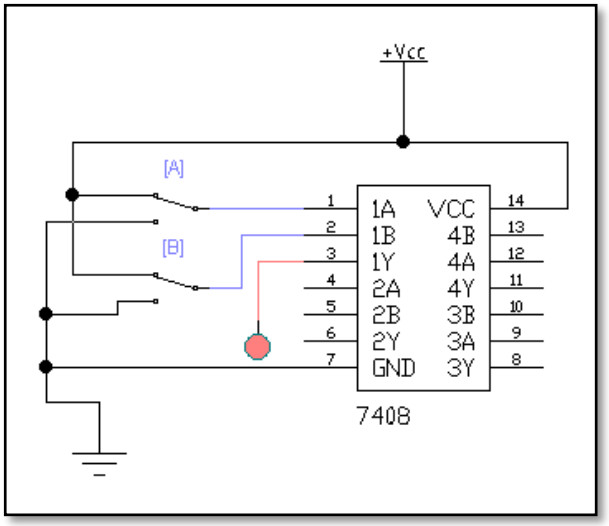
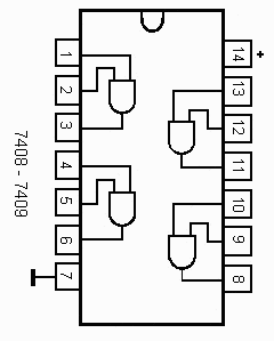


Figure128 : Brochage du 7408



2. Table de vérité

A	B	S
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

3. L'équation **S = A . B**
Nom de la fonction : **ET (AND)**

C. Fonctions logiques en logique câblée

C.1. Premier montage

1. Circuit logique de **S** à base du CI 74LS00 (4 portes NAND à 2 IN)

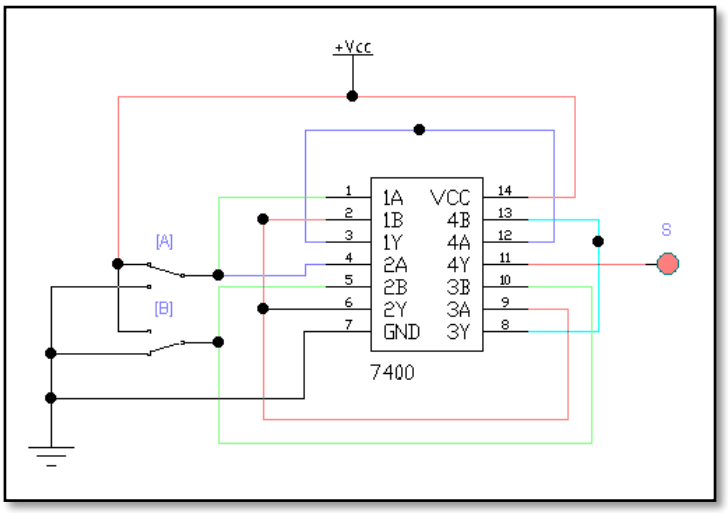
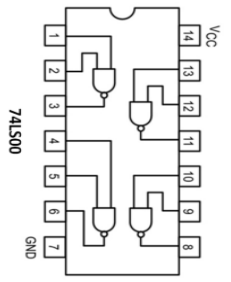


Figure129 : Brochage du 7400



2. Table de vérité

A	B	S
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

3. L'équation **$S= \overline{A}.B+A.\overline{B} = A \oplus B$**
Nom de la fonction obtenue : **EXOR (OU exclusive)**

C.2. Deuxième montage

1. Circuit logique de **S** à base du CI 74LS04 (6 portes NOT) et du CI 74LS02 (4 portes NOR à 2 IN)

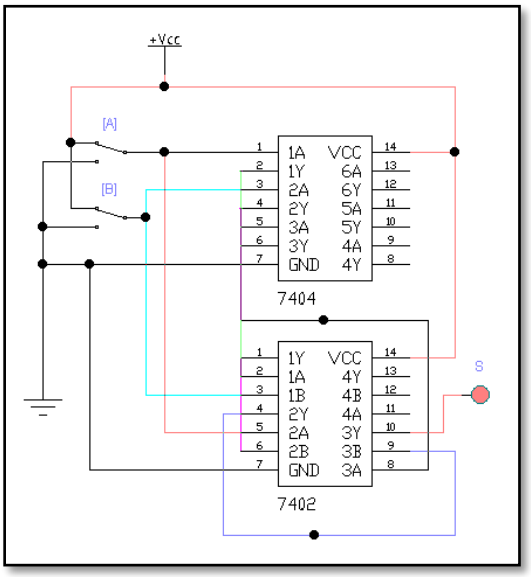
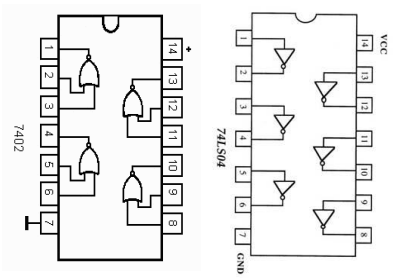


Figure130 : Brochages du 7404 et du 7402



2. Table de vérité

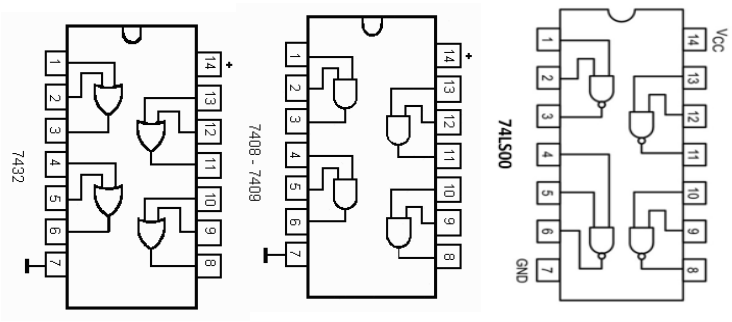
A	B	S
0	0	1
0	1	0
1	0	0
1	1	1

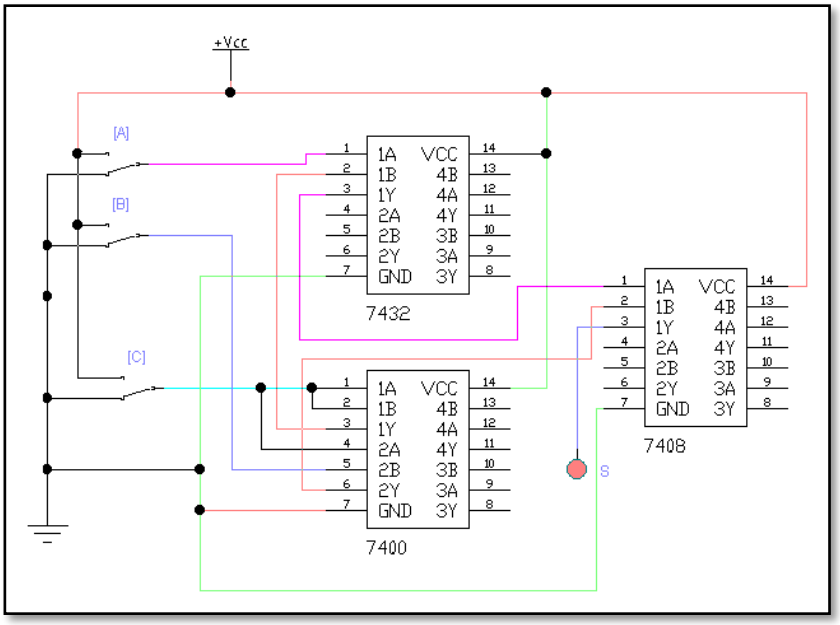
3. L'équation **$S= AB+\bar{A}.\bar{B} = \overline{A \oplus B}$**
Nom de la fonction obtenue : **EXNOR (NOR exclusive)**

C.3. Troisième montage : $F= (A+\bar{C}).(\overline{C.B})$

1. Circuit logique de **F** à base du CI 7432, du CI 7408 et du CI 7400

Figure131 : Brochages du CI 7432, du 7408 et du 7400





2. Table de vérité de F

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

3. Simplifier algébriquement la fonction F.

$F = (A+\bar{C}).(\overline{C.B}) = (A+\bar{C}).(\bar{C}+\bar{B}) = A.\bar{C} + A.\bar{B} + \bar{C}.\bar{C} + \bar{C}.\bar{B}$

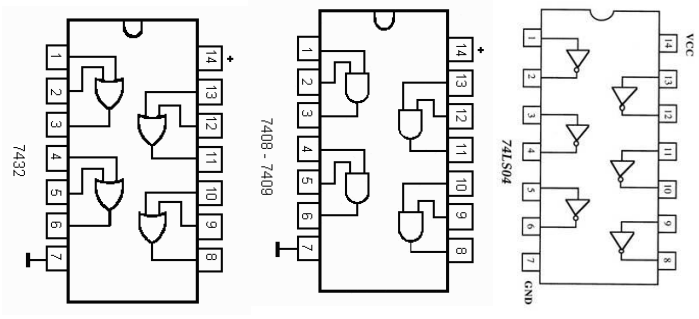
$F = A.\bar{C} + A.\bar{B} + .\bar{C} + \bar{C}.\bar{B} = \bar{C}.(A+1+\bar{B}) + A.\bar{B}$ [$(A+1+\bar{B})=1$]

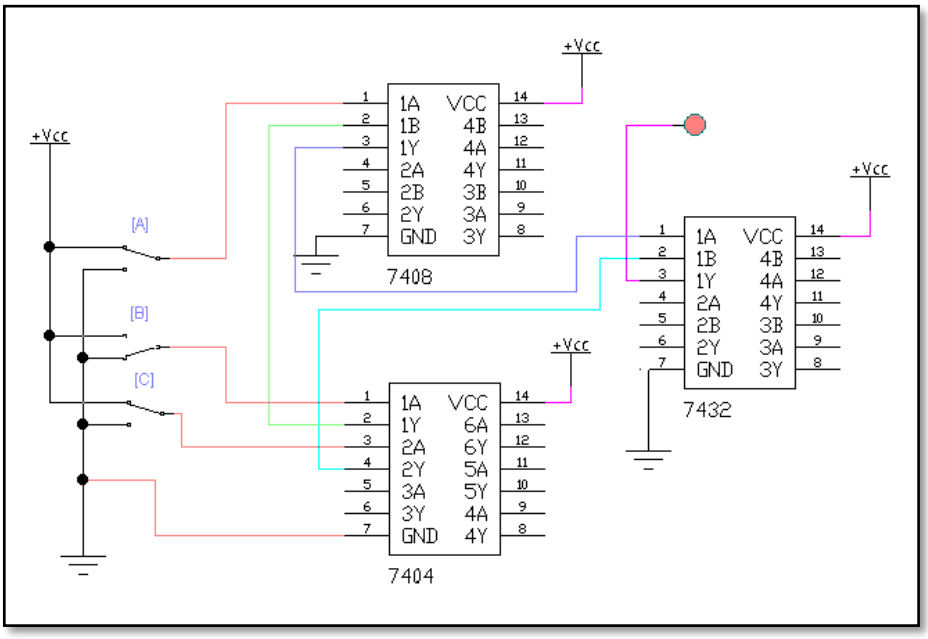
$F = \bar{C}.1 + A.\bar{B}$

$F = \bar{C} + A.\bar{B}$

4. Circuit logique de F après simplification. A base des CI 7432, 7408 et 7404

Figure133 : Brochages du CI 7432, du 7408 et du 7404





5. Table de vérité

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

6. Conclusion :

A. Après simplification en constate qu'on a obtenu la même table de vérité ce qui revient à dire que la simplification est juste.

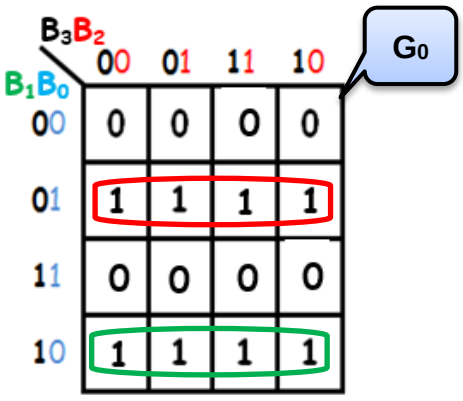
B. On remarque que après simplification le nombre de circuits intégrer est resté toujours à trois, ce qui revient à dire que la simplification ne nous permet pas toujours d'économiser le nombre de circuits intégrer.

❖ **Exercice 16 :** Réalisation d'un convertisseur Binaire Naturel-Binaire Réfléchi 4bits

A. Table de vérité

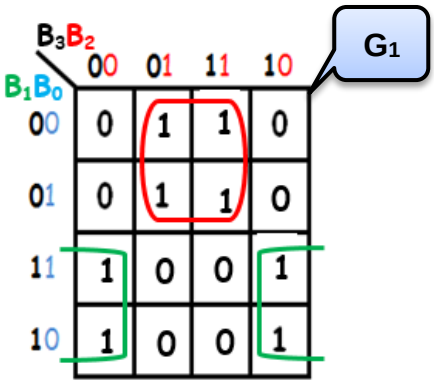
Entrée Binaire Naturel				Sortie Binaire Réfléchi			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

B. Tableau de KARNAUGH



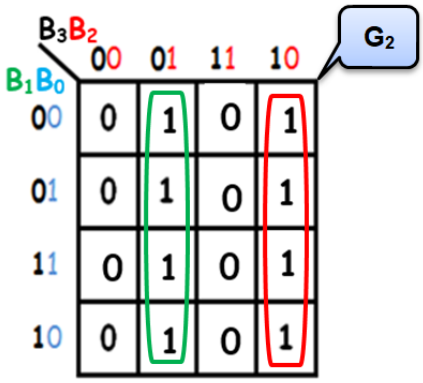
$G_0 = \overline{B_0}.B_1 + B_0.\overline{B_1}$

$G_0 = B_0 \oplus B_1$



$G_1 = \overline{B_1}.B_2 + B_1.\overline{B_2}$

$G_1 = B_1 \oplus B_2$

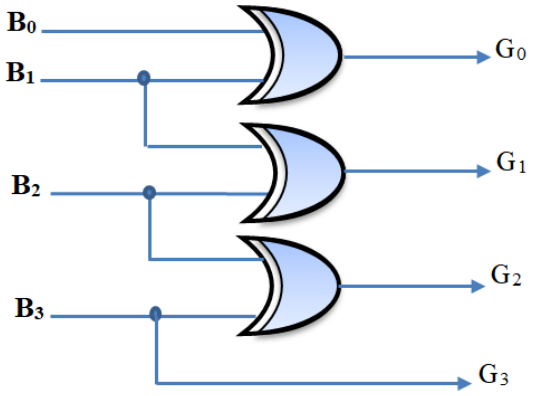


$G_2=\overline{B_3}.B_2 + B_3.\overline{B_2}$

$G_2=B_2 \oplus B_3$

$G_3=B_3$

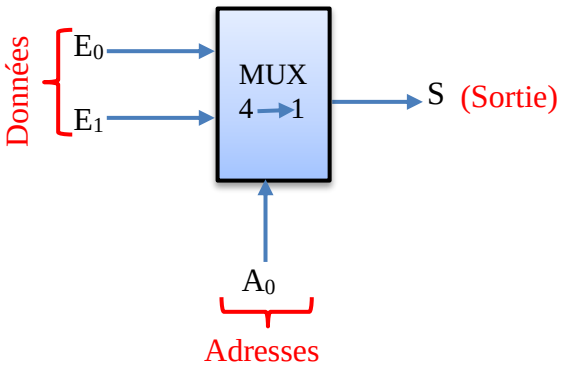
C. Circuit logique



❖ Exercice 17

A. MUX 2 vers 1

➢ **Synoptique**

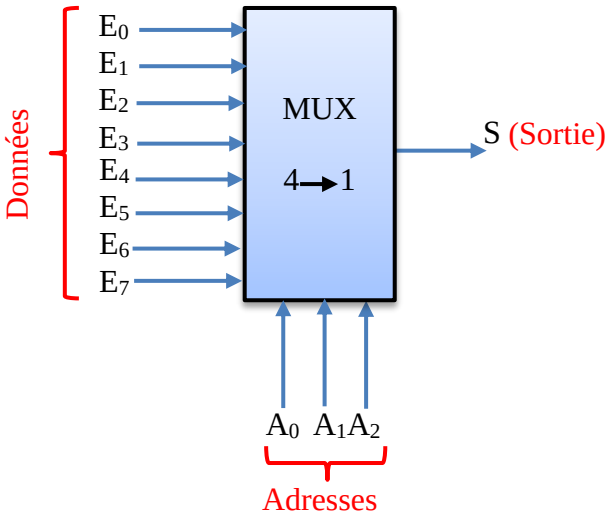


➢ **Table de vérité**

A_0	S
0	E_0
1	E_1

B. MUX 8 vers 1

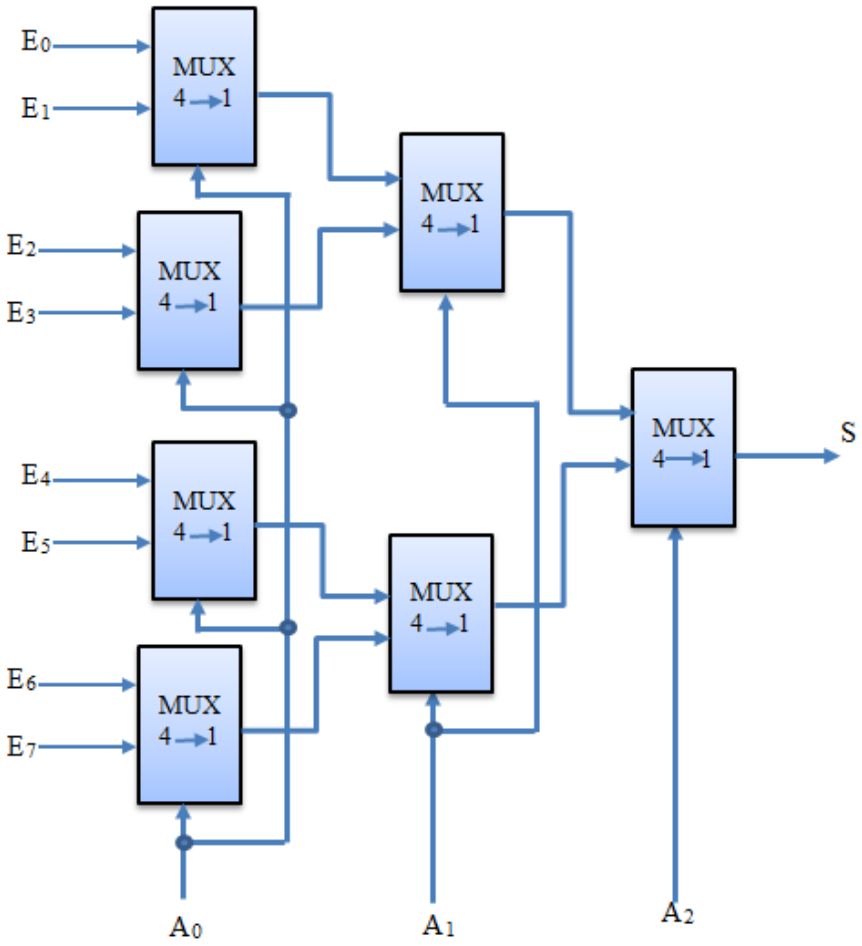
➢ **Synoptique**



➢ **Table de vérité**

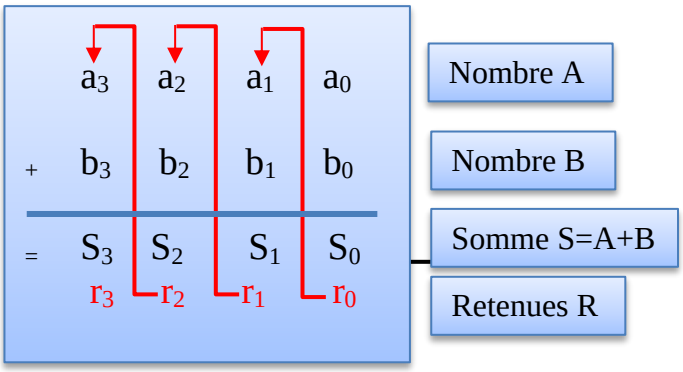
A_2	A_1	A_0	S
0	0	0	E_0
0	0	1	E_1
0	1	0	E_2
0	1	1	E_3
1	0	0	E_4
1	0	1	E_5
1	1	0	E_6
1	1	1	E_7

C. Synoptique d'un MUX 8 vers 1 à l'aide de MUX 2 vers 1

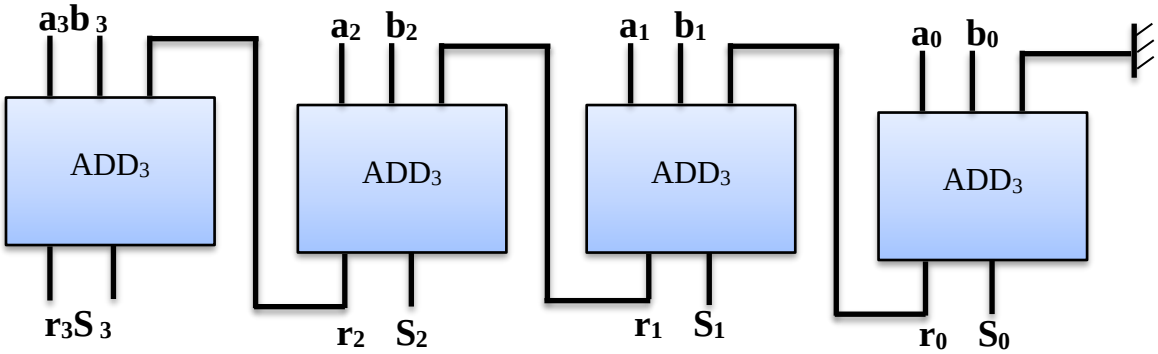


❖ **Exercice 18** Schéma fonctionnel d'un additionneur complet 4 bits

A. Décomposition de l'addition de deux nombres binaires de quatre bits.



B. Synoptique d'un additionneur complet 4 bits

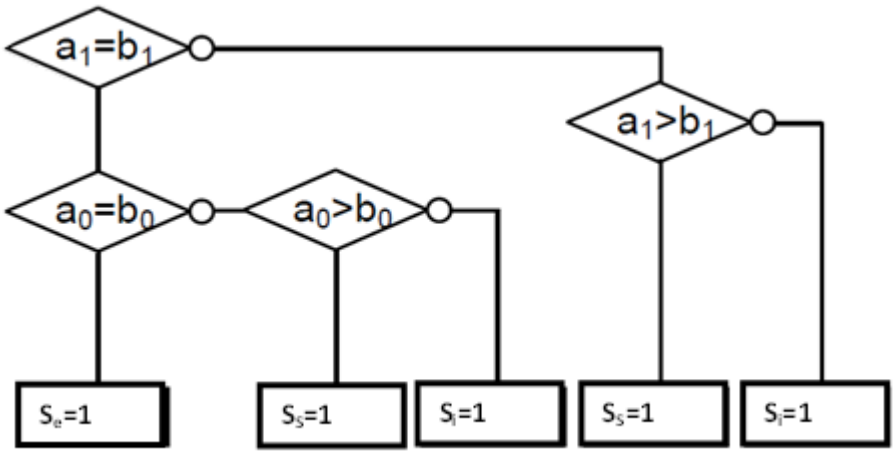


❖ **Exercice 19**

Comparateur 2 bits qui permet de faire la comparaison entre deux nombres A (A_1A_0) et B (B_1B_0) chacun sur deux bits

A. Synoptique du comparateur 2bits

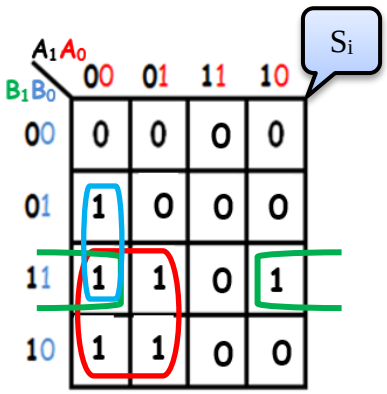
B. Organigramme du comparateur 2bits

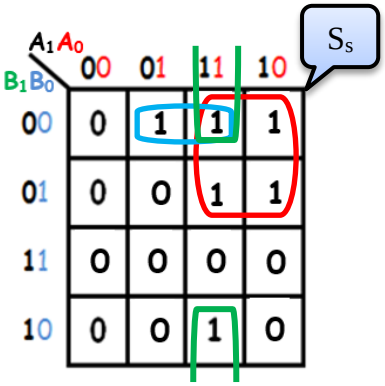


C. Table de vérité du comparateur 2bits

B_1	B_0	A_1	A_0	S_e(A=B)	S_i (A<B)	S_s(A>B)
0	0	0	0	1	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	0	1	0
1	0	0	1	0	1	0
1	0	1	0	1	0	0
1	0	1	1	0	0	1
1	1	0	0	0	1	0
1	1	0	1	0	1	0
1	1	1	0	0	1	0
1	1	1	1	1	0	0

D. Equations





$S_i = \bar{A}_1 . B_1 + \bar{A}_0 . \bar{A}_1 . B_0 + \bar{A}_0 . B_0 . B_1$

$\mathbf{S_i = \bar{A}_1 . B_1 + (A_1 \oplus B_1) . (\bar{A}_0 . B_0)}$

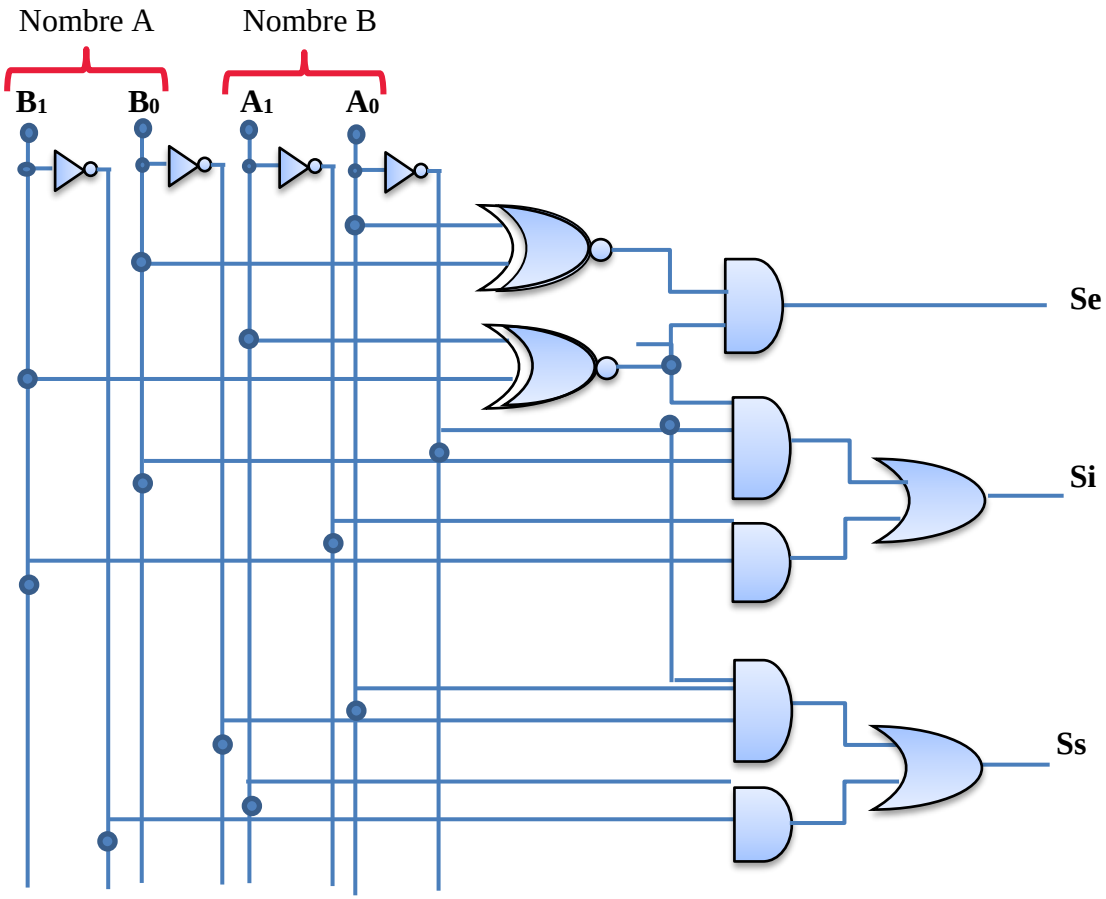
$S_S = A_1 . \bar{B}_1 + A_0 . \bar{B}_0 . \bar{B}_1 + A_0 . A_1 . \bar{B}_0$

$\mathbf{S_S = A_1 . \bar{B}_1 + (A_1 \oplus B_1) . (A_0 . \bar{B}_0)}$

$S_e = \bar{A}_0 . \bar{A}_1 . \bar{B}_0 . \bar{B}_1 + A_0 . \bar{A}_1 . B_0 . \bar{B}_1 + \bar{A}_0 . A_1 . \bar{B}_0 . B_1 + A_0 . A_1 . B_0 . B_1$

$\mathbf{S_e = \overline{(A_0 \oplus B_0)} . \overline{(A_1 \oplus B_1)}}$

E. Circuit logique du comparateur 2 bits



❖ **Exercice 20** D'après la figure 70 de la page 54, l'équation logique de la fonction S est :

$S1 = \overline{A}\overline{B} + AB = \overline{A \oplus B}$

$S_2 = \bar{C}\bar{D} + \bar{C}D + CD = \bar{C} + CD = \bar{C} + CD$

$\Rightarrow$ $S = \bar{s}_2 s_1 E + s_2 s_1 \bar{E} = S_1(\bar{s}_2 E + s_2 \bar{E}) = S_1(E \oplus S_2)$

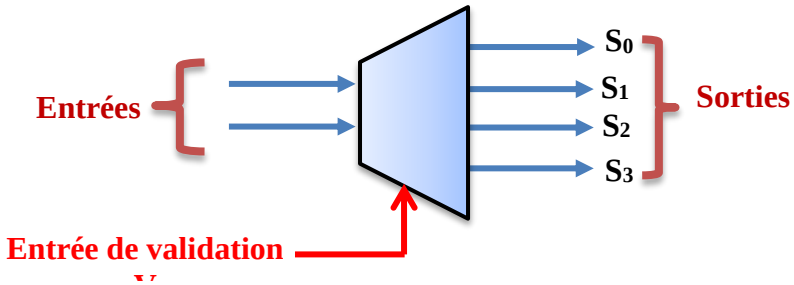
$\mathbf{S = \overline{A \oplus B}[E \oplus (\bar{C} + D)}$

❖ **Exercices de synthèse 3**

I. Décodeur 2vers 4

1. Etude de décodeur 2 vers 4

A. Schéma fonctionnel



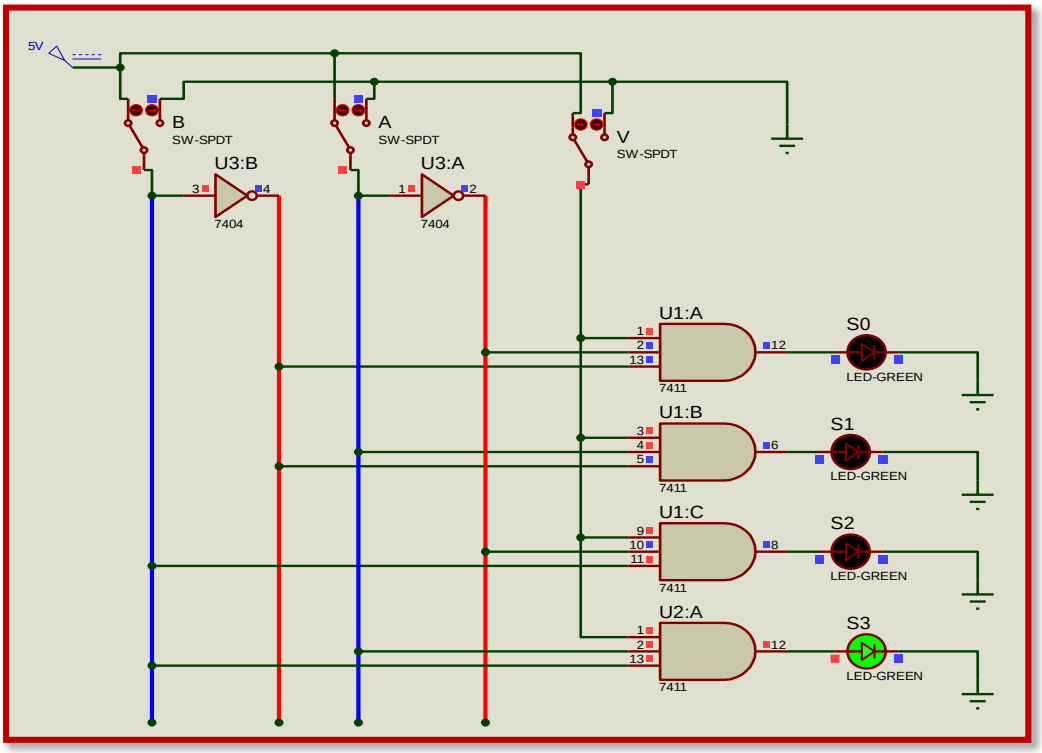
B. Table de vérité

Entrées			Sorties			
V	B	A	S_3	S_2	S_1	S_0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

C. Equations de sorties

$S_0 = V.\bar{A}.\bar{B}$, $S_1 = V.A.\bar{B}$, $S_2 = V.\bar{A}.B$, $S_3 = V.A.B$

D. Circuit logique du décodeur 2 Vers 4



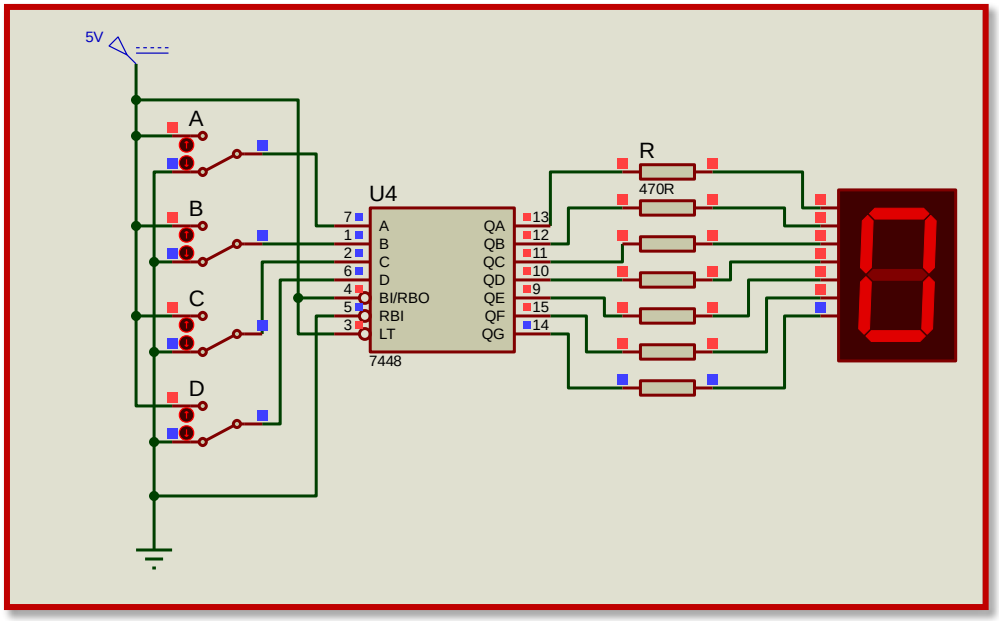
2. Tester le fonctionnement du décodeur2 Vers 4

Table de vérité

Entrées			Sorties			
V	B	A	S_3	S_2	S_1	S_0
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

II. Afficheur 7 segments

1. Câblage du circuit



2. Table de vérité

Entrées				Sorties de l'afficheur							Valeur décimale
D	C	B	A	a	b	c	d	e	f	g	
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	0	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	0	0	1	1	9

A. Equations simplifiées de chaque sortie (d'après les tableaux de Karnaugh) :

a= $\bar{A}\bar{C}$+AB+AC+D = $\overline{A \oplus C}$ +AB+D

b=$\bar{A}\bar{B}$+AB+C = $\overline{A \oplus B}$ +C

c=A+$\bar{B}$+C

d=$\bar{A}\bar{C}$+$\bar{A}$B+B$\bar{C}$+A$\bar{B}$C

e=$\bar{A}\bar{C}$+$\bar{A}$B=$\bar{A}$(B+$\bar{C}$)

f=$\bar{A}\bar{B}$+$\bar{B}$C+$\bar{A}$C+D

g=$\bar{A}$B + B$\bar{C}$+$\bar{B}$C +D = $\bar{A}$B+B$\oplus$C+D

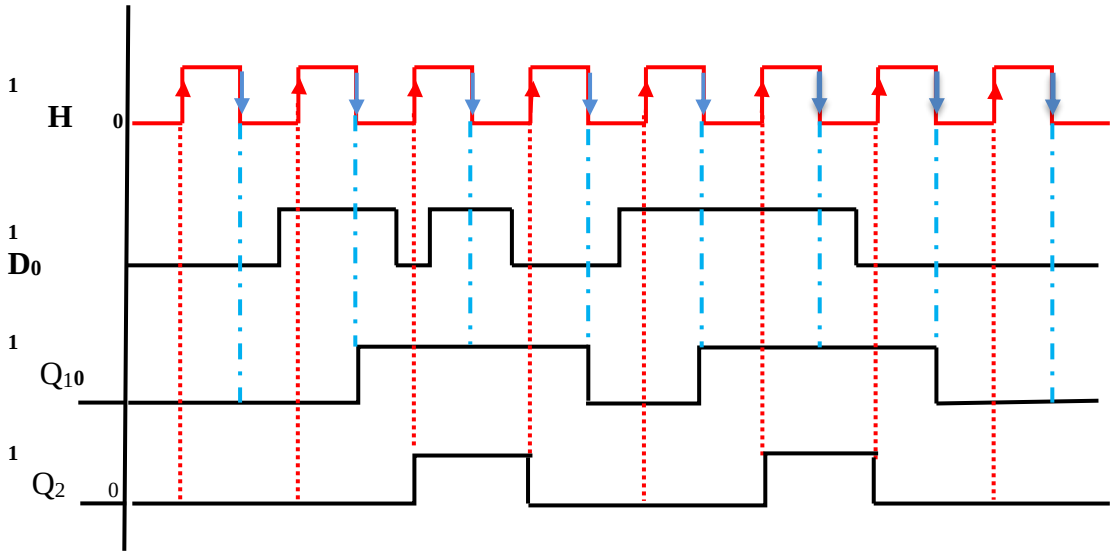
❖ **Exercice 21**Répondre par vrai ou faux

- Un signal d'horloge est généralement un signal sinusoïdal***(Faux)***
- Un front descendant caractérise un signal passant de 0 à 1 (**Faux**)
- l'état de la sortie d'un circuit séquentiel dépend non seulement de la combinaison appliquée à ces l'entrées mais aussi de l'état précédent de ces sorties (**Vrai**)
- La logique séquentielle est une logique combinatoire avec une mémorisation des sorties (**Vrai**)
- Un circuit est dit synchrone si la lecture de ces entrées se fait à tout instant (**Faux**)
- Dans un circuit l'état de sortie ne dépend que de l'état présent des entrées (**Vrai**)
- Une bascule est un organe de mémorisation unitaire (**Vrai**)

❖ **Exercice 22**

Les chronogrammes de Q1 et Q2 du circuit logique de la figure 100

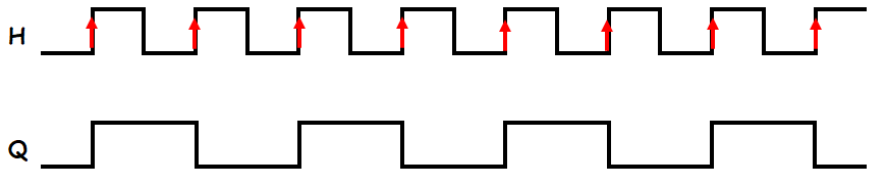
(On suppose qu'au début Q1 =0 et Q2=0)



❖ Exercice 23

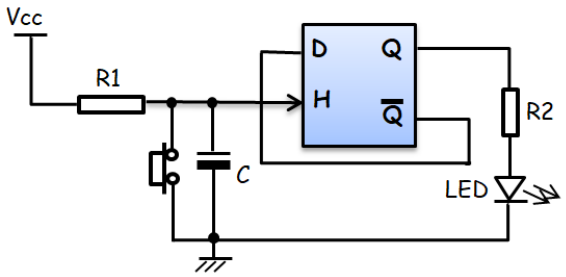
1. Soit la bascule D de la figure 102

A. Chronogramme obtenu de H et Q. (Q bascule à chaque front montant de H.)



B. Le fonctionnement du montage de figure 104:

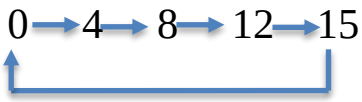
Figure104: Circuit à base de Bascule D



- ✓ Au départ le bouton poussoir n'est pas appuyé et le condensateur C est déchargé (Vc = 0V). Q = 0 et D = 1. Le condensateur se charge et H (entrée d'horloge) passe de 0 à 1 (front montant) et Q = 1 et D = 0 (***la LED s'allume***).
- ✓ Lorsqu'on appui sur bouton poussoir le condensateur C est court-circuité (Vc = 0V) et H passe de 1 à 0 (front descendant), Q ne change pas (Q = 1 et D = 0).
- ✓ Lorsqu'on relâche le bouton poussoir, le condensateur se charge et CLK passe de 0 à 1 (front montant) et Q = 0 et D = 1 (***la LED s'éteint***).

  Donc à chaque fois qu'on appui et on relâche le bouton poussoir, on a un basculement de Q (la LED passe d'un état à l'autre).

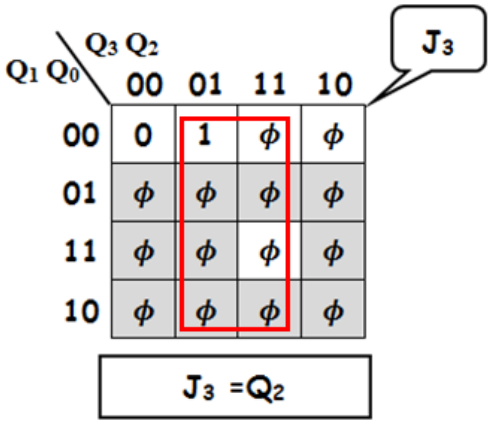
❖ **Exercice 24** : Réalisation d'un compteur synchrone qui affiche la séquence suivante :

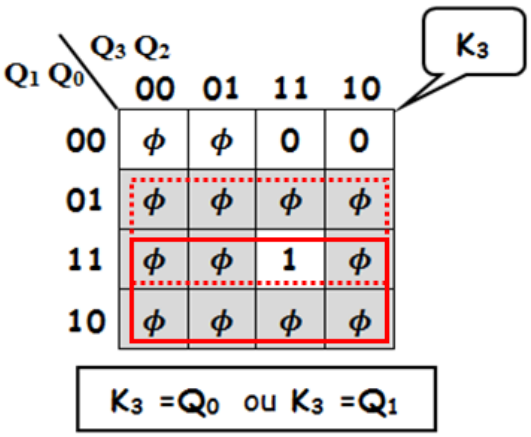


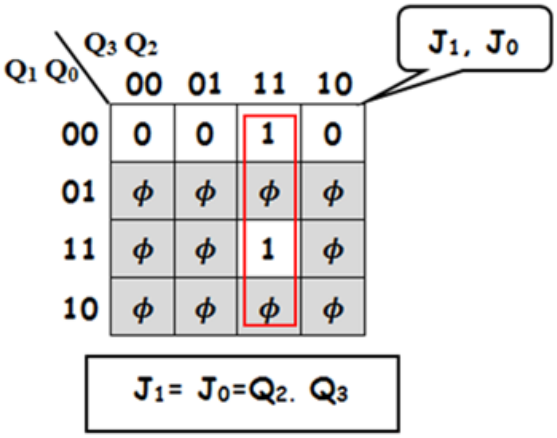
A. Table de Vérité

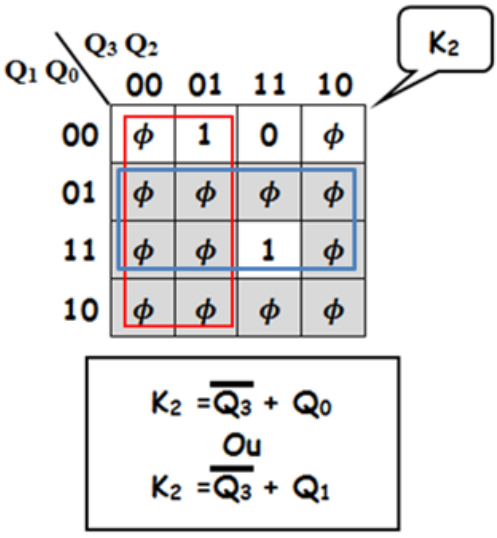
	Q_3	Q_2	Q_1	Q_0		J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0		0	Φ	1	Φ	0	Φ	0	Φ
4	0	1	0	0		1	Φ	Φ	1	0	Φ	0	Φ
8	1	0	0	0		Φ	0	1	Φ	0	Φ	0	Φ
12	1	1	0	0		Φ	0	Φ	0	1	Φ	1	Φ
15	1	1	1	1		Φ	1	Φ	1	Φ	1	Φ	1

B. Tableau de KARNAUGH









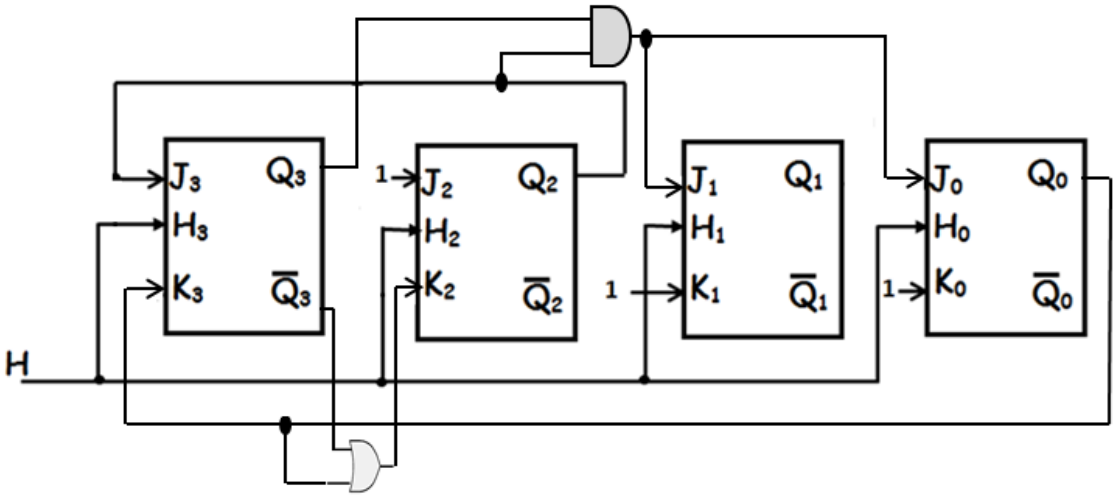
$J_0=Q_2.Q_3$ $K_0=1$

$J_1=Q_2.Q_3$ $K_1=1$

$J_2=1\ J_3=Q_2$ $K_2=\overline{Q_3}+Q_0$ **Ou** $K_2=\overline{Q_3}+Q_1$

$K_3=Q_0$ ou $K_3=Q_1$

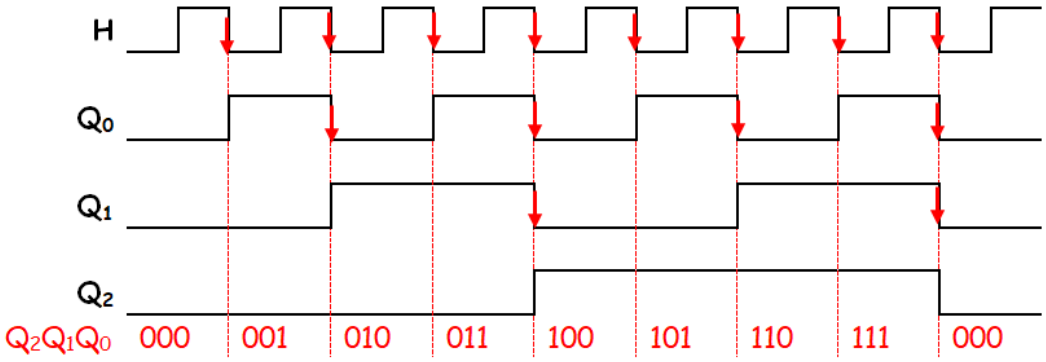
C. Logigramme du compteur synchrone



❖ **Exercice 25** Soit le compteur asynchrone de la figure 101 :

A. Chronogramme :

- ✓ Q_0 a pour horloge H ; donc à chaque front descendant de H, Q_0 change d'état (bascule de 0 à 1 ou de 1 à 0).
- ✓ Q_1 a pour horloge Q_0; donc à chaque front descendant de Q_0, Q_1 change d'état.
- ✓ Q_2 a pour horloge Q_1; donc à chaque front descendant de Q_1, Q_2 change d'état.



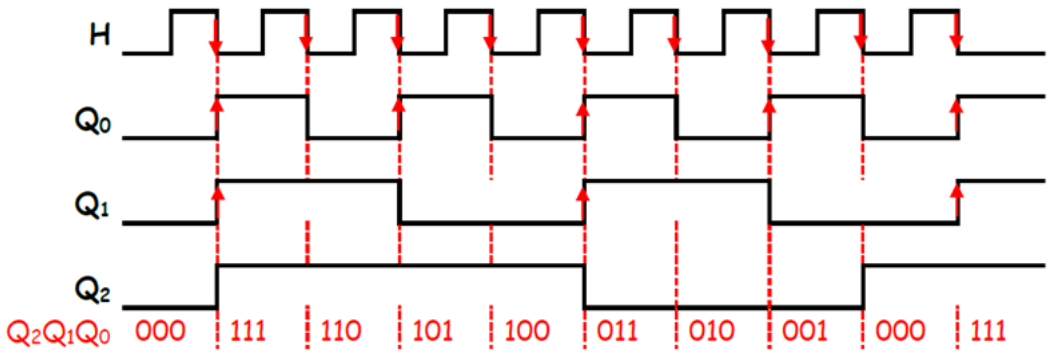
B. On obtient la séquence suivante : 0, 1, 2, 3, 4, 5, 6, 7,0

C. On a un compteur modulo 8.

❖ **Exercice 26** Soit le compteur asynchrone de la figure 114 :

A. Chronogramme :

- ✓ Q_0 a pour horloge H ; donc à chaque front descendant de H, Q_0 change d'état (bascule de 0 à 1 ou de 1 à 0).
- ✓ Q1 a pour horloge $\bar{Q}_0$; donc à chaque front descendant de $\bar{Q}_0$ (front montant de Q_0), Q_1 change d'état.
- ✓ Q_2 a pour horloge $\overline{Q_1}$; donc à chaque front descendant de $\overline{Q_1}$ (front montant de Q_1), Q_2 change d'état.



B. On obtient la séquence suivante : 0, 7, 6, 5, 4, 3, 2, 1, 0

C. On a un décompteur modulo 8.

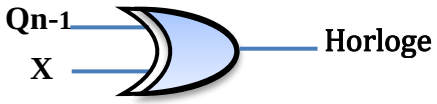
D. L'horloge de la 1ère bascule dans les 2 cas est H. Pour le compteur l'horloge de la bascule n est Qn-1 et pour le décompteur l'horloge de la bascule n est $\bar{Q}_{n-1}$

E. Pour concevoir un compteur/décompteur on doit choisir soit Qn-1, soit $\bar{Q}_{n-1}$

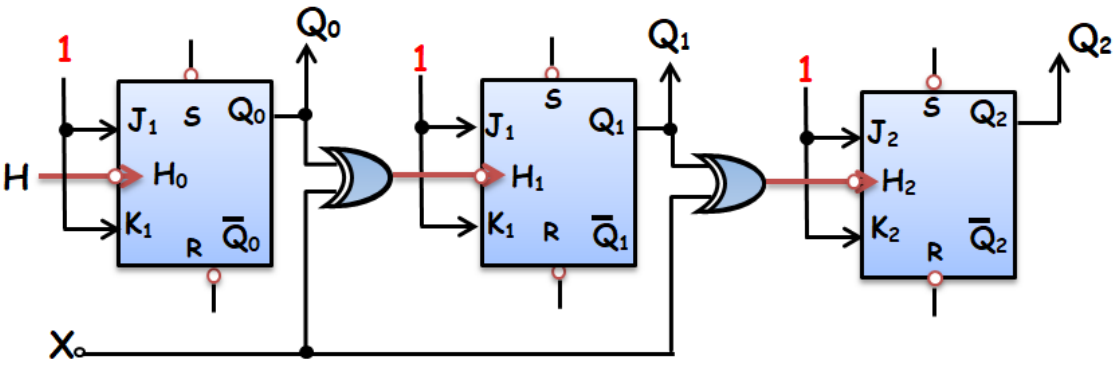
Pour faire le choix, on va utiliser une variable X, tel que :

X	Horloge	Mode
0	Qn-1	Compteur
1	$\bar{Q}_{n-1}$	Décompteur

$$Horloge = Q_{n-1}\bar{X} + \bar{Q}_{n-1}X = Q_{n-1} \oplus X$$



❖ **Circuit logique d'un compteur/décompteur synchrone**



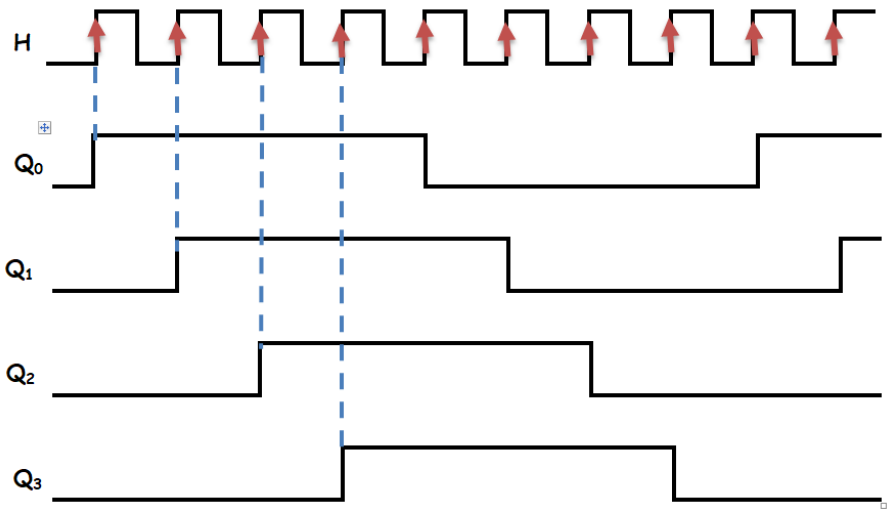
❖ **Exercice 27**D'après le circuit de la figure 122, on a :

$D_0 = \bar{Q}_3$; $D_1 = Q_0$; $D_2 = Q_1$; $D_3 = Q_2$

L'état initial est :

$Q_0 = 0$; $Q_1 = 0$; $Q_2 = 0$ et $Q_3 = 0$ $\Longrightarrow$ $D_0 = 1$; $D_1 = 0$; $D_2 = 0$ et $D_3 = 0$

A. Chronogramme :



B. La fonction réalisée est: registre à décalage circulaire à droite et le cycle réalisé est :

0 – 1 – 3 – 7 – F – E – C – 8 – 0

❖ **Exercice 28 :**

A	SRAM	**5**
C	EPROM	**4**
E	ROM	**3**
G	Bus d'adresse	**2**

B	EEPROM	**1**
D	Débit	**6**
F	Bus de commande	**8**
H	Bus de donnée	**7**

❖ **Exercice 29 :** Soit la mémoire 27C256 de la figure 139 :

1 Bus d'adresse : (A0-A14) soit 15 fils d'adresses

2 Bus de données : (Q0-Q7) soit 8 fils de données

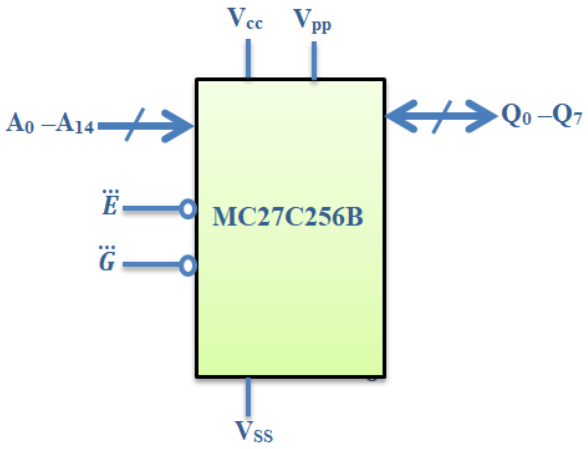
3 15 fils d'adresse (2 valeurs par fil)⇒ Le nombre d'adresses = 2^{15} = 32768 adresse

4 La première adresse a pour valeur $(0000000000000000)_2$

5 La dernière adresse a pour valeur $(1111111111111111)_2$

6 8 fils de données (de 2 valeurs par fil)⇒ Le nombre de valeurs que peut prendre la donnée = 2 8 = 256 valeurs

Figure 139 : Mémoire 27C256

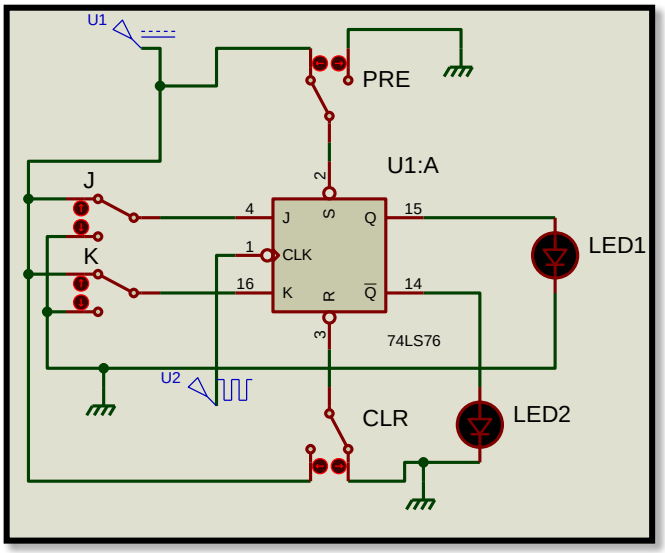


.

Exercices de synthèse 4

I. Bascule JK :

1. Réalisation du circuit de la bascule JK on utilisant la 1erbascule du 74LS76



2. **Q=f($\overline{PRE}$, $\overline{CLR}$)**

$\overline{S}$	$\overline{R}$	Fonctionnement
$\overline{PRE}$	$\overline{CLR}$	
1	1	Bascule JK
0	1	Forçage à 1
1	0	Forçage à 0
0	0	Etat indésirable

3. $\overline{PRE}$ = $\overline{CLR}$ =1 et en reliant le module horloge
Générateur de signaux carrés GBF avec une fréquence de 1Hz à l'entrée CLK ,

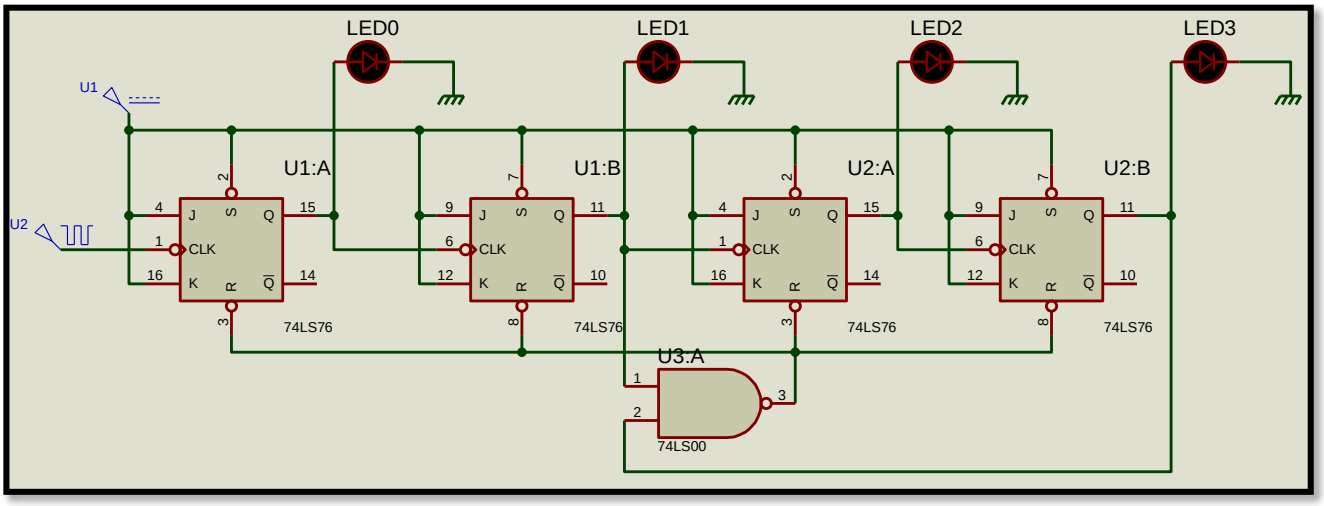
Entrées					Sorties		
PRE	CLR	CLK	J	K	Q	$\overline{Q}$	Effet
0	1	X	X	X	1	0	Forçage à 1
1	0	X	X	X	0	1	Forçage à 0
0	0	X	X	X	1	1	Etat indésirable
1	1	↓	0	0	Q_{-1}	$\overline{Q}_{-1}$	Mémorisation
1	1	↓	1	0	1	0	Mise à 1
1	1	↓	0	1	0	1	Mise à 0
1	1	↓	1	1	$\overline{Q}_{-1}$	Q_{-1}	Basculement
1	1	0	X	X	Q_{-1}	$\overline{Q}_{-1}$	Mémorisation
1	1	1	X	X	Q_{-1}	$\overline{Q}_{-1}$	Mémorisation

Conclusion :

- PRE=0 , Force la bascule à 1
- CLR=0, Force la bascule à 0
- PRE=CLR=1 , la bascule se comporte comme une bascule JK normale
- PRE=CLR=0 , est un état indésirable parce que **Q** =$\overline{\mathbf{Q}}$ ce qui est anormale

II. Compteur binaire :

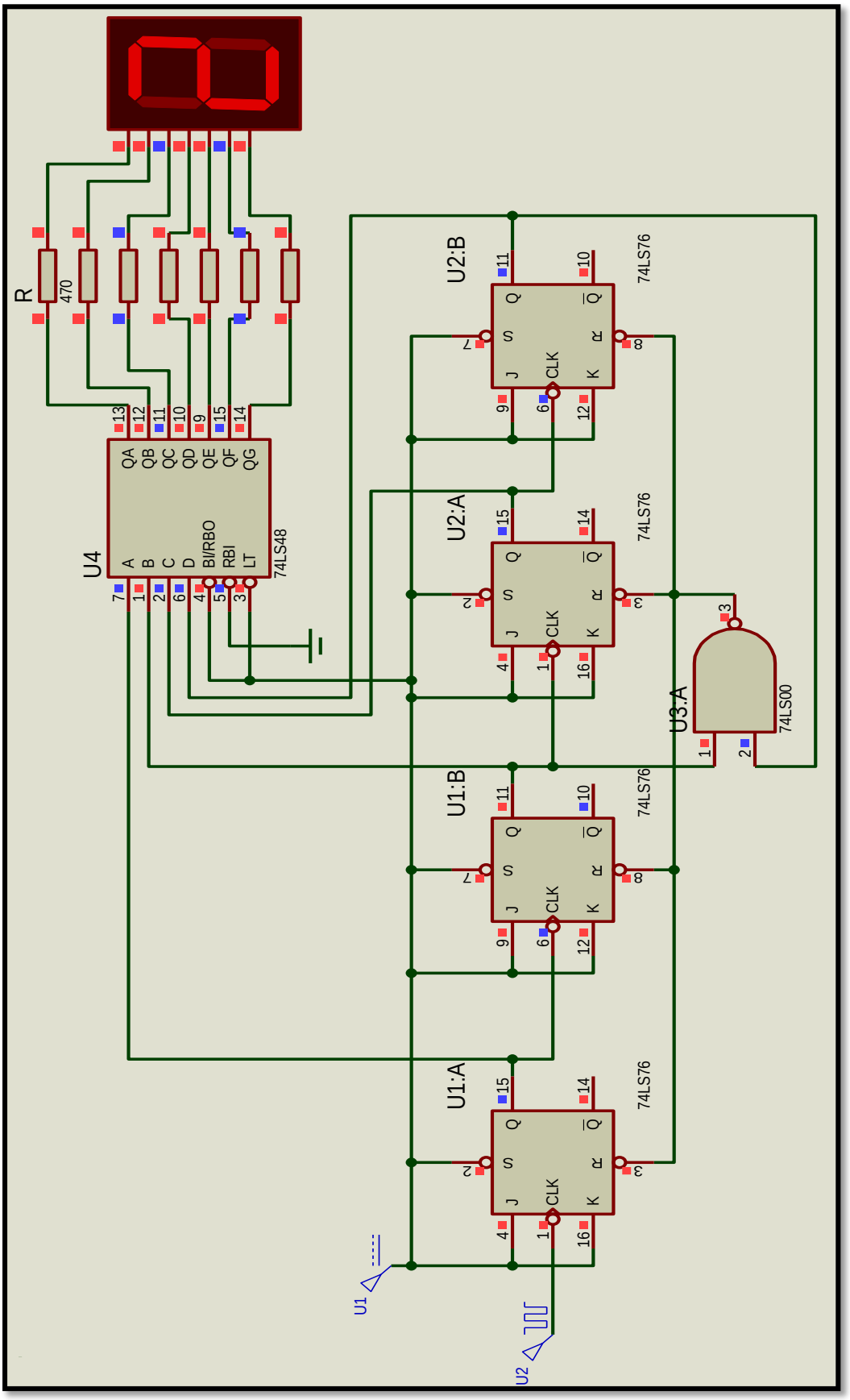
1. Compteur asynchrone modulo 10 à base de circuits intégrés de type 74LS76



- **Table de vérité :**

Entrées		Sorties			
Nombre d'impulsions	Horloge CLK	Q_3	Q_2	Q_1	Q_0
0	↓	0	0	0	0
1	↓	0	0	0	1
2	↓	0	0	1	0
3	↓	0	0	1	1
4	↓	0	1	0	0
5	↓	0	1	0	1
6	↓	0	1	1	0
7	↓	0	1	1	1
8	↓	1	0	0	0
9	↓	1	0	0	1

2. Compteur asynchrone modulo 10 à base de circuits intégrés de type 74LS76, décodeur 7 segment de type 7447 et un afficheur 7 segments



➢ **Table de vérité :**

Entrées		Sorties				
Nombre d'impulsions	**Horloge CLK**	**Q_3**	**Q_2**	**Q_1**	**Q_0**	**Afficheur 7 segments**
0	↓	0	0	0	0	**0**
1	↓	0	0	0	1	**1**
2	↓	0	0	1	0	**2**
3	↓	0	0	1	1	**3**
4	↓	0	1	0	0	**4**
5	↓	0	1	0	1	**5**
6	↓	0	1	1	0	**6**
7	↓	0	1	1	1	**7**
8	↓	1	0	0	0	**8**
9	↓	1	0	0	1	**9**

CORRECTION ACTIVITE DE SYNTHESE

I. Algèbre de Boole

Distributeur de boissons chaudes

Soient u, v, w, z les variables logiques correspondant aux propositions suivantes :

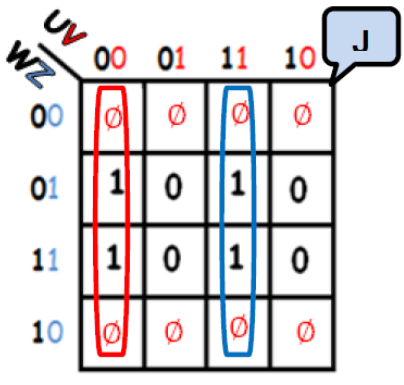
- Le bouton « café » est enfoncé : u = 1
- Le bouton « thé » est enfoncé : v = 1
- Le bouton « lait » est enfoncé : w = 1
- Un jeton a été introduit dans la fente de l'appareil : z = 1
- ∅: Indéfini (0 ou 1)

Simplification des fonctions de restitution du jeton, J, de distribution du café, C, du thé T, et du lait, L.

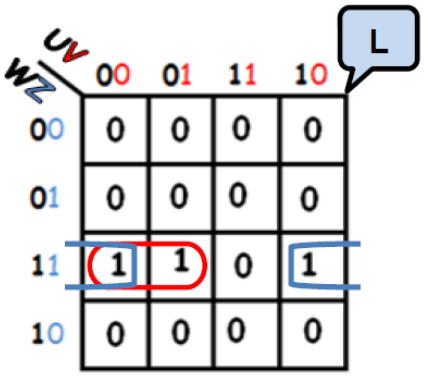
- Table de vérité de C, T, L et J :

u	v	w	z	C	T	L	J
0	0	0	0	0	0	0	∅
0	0	0	1	0	0	0	**1**
0	0	1	0	0	0	0	∅
0	0	1	1	0	0	**1**	**1**
0	1	0	0	0	0	0	∅
0	1	0	1	0	**1**	0	0
0	1	1	0	0	0	0	∅
0	1	1	1	0	**1**	**1**	0
1	0	0	0	0	0	0	∅
1	0	0	1	**1**	0	0	0
1	0	1	0	0	0	0	∅
1	0	1	1	**1**	0	**1**	0
1	1	0	0	0	0	0	∅
1	1	0	1	0	0	0	**1**
1	1	1	0	0	0	0	∅
1	1	1	1	0	0	0	**1**

- Tableau de KARNAUGH



$J = \overline{U}.\overline{V} + U.V = \overline{U \oplus V}$



$L = \overline{U}.W.Z + \overline{V}.W.Z = W.Z.(\overline{U} + \overline{V})$

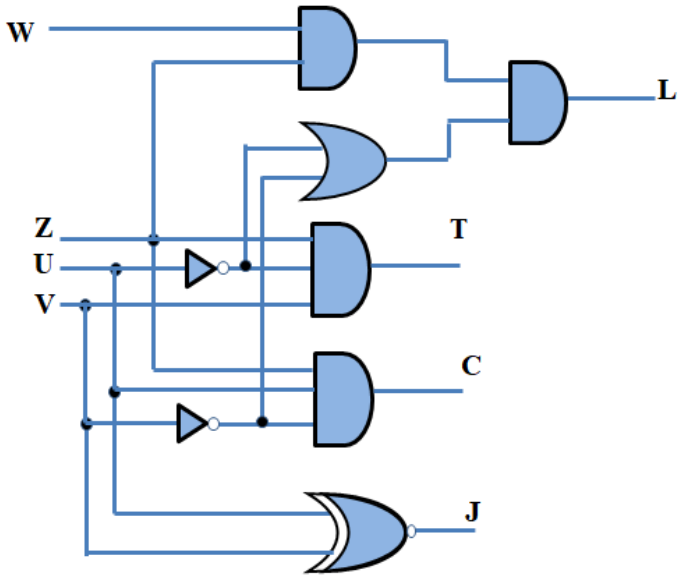
$C=U.\bar{V}.\bar{W}.Z + U.\bar{V}.W.Z = U.\bar{V}.Z.(W+\bar{W})$

$C= U.\bar{V}.Z$

$T= \bar{U}.V.\bar{W}.Z+\bar{U}.V.W.Z = \bar{U}.V.Z.(W+\bar{W})$

$T= \bar{U}.V.Z$

- **Circuit logiquedu distributeur de boissons chaudes**



II. Circuits combinatoires

Demi-Soustracteur.

1. Table de vérité :

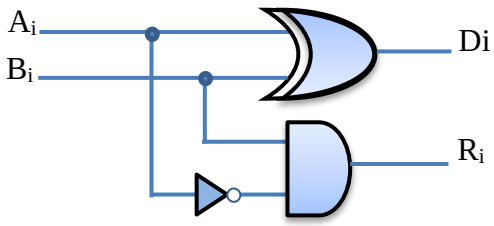
A_i	B_i	D_i	R_i
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

2. Equations :

$D_i=\bar{A_1}+B_i+ A_i+\bar{B_1}= A_i \oplus B_i$

$R_i=\bar{A_1}+B_i$

3. Logigrammedu demi-soustracteur



Soustracteur complet :

1. Table de vérité :

R_{i-1}	A_i	B_i	D_i	R_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	1	0
1	1	1	0	1

2. Equations :

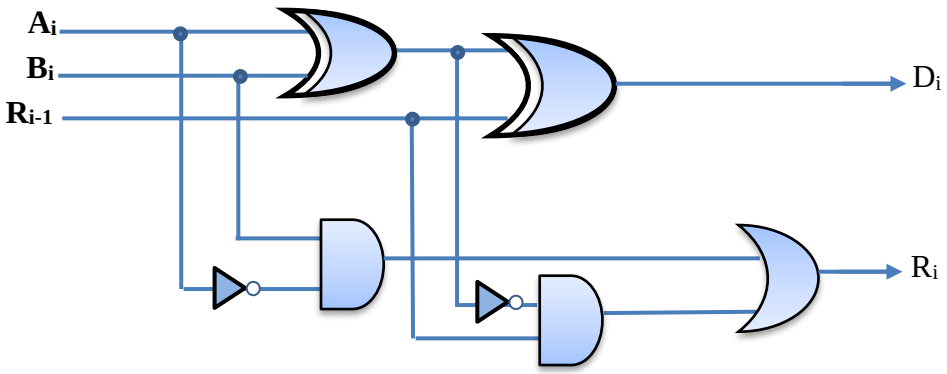
$D_i=\overline{R}_{i-1}\overline{A}_i.B_i+\overline{R}_{i-1}A_i.\overline{B}_i+R_{i-1}\overline{A}_i\overline{B} + R_{i-1}A_i\overline{B}_i$

$D_i= (A_i \oplus B_i) \oplus R_{i-1}$

$R_i= \overline{R}_{i-1}.\overline{A}_i.B_i+R_{i-1}.\overline{A}_i.\overline{B} + R_{i-1}.\overline{A}_i.B_i+R_{i-1}.A_i.B_i$

$R_i=\overline{A}_i.B_i + R_{i-1}\ (A_i \oplus B_i)$

3. Logigrammedu soustracteur complet



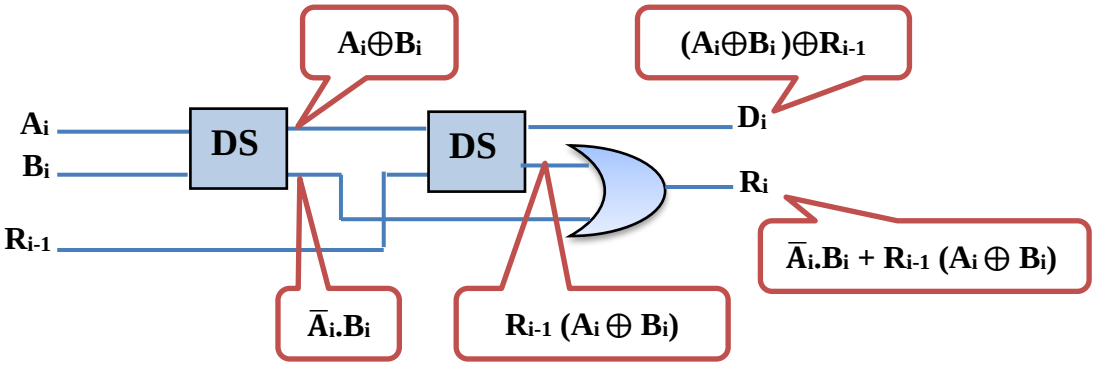
4.

A. Soustracteur complet à l'aide deux demi-soustracteurs :

Ce schéma correspond au fait que le soustracteur est réalisé en :

- Retranchant B_i de A_i (1er demi-soustracteur) (**DS**)
- Puis retranchant R_{i-1}de la différence obtenue

Circuit logique du soustracteur complet à l'aide de 2 demi-soustracteurs



A. Avec un demi-additionneur et un demi-soustracteur.

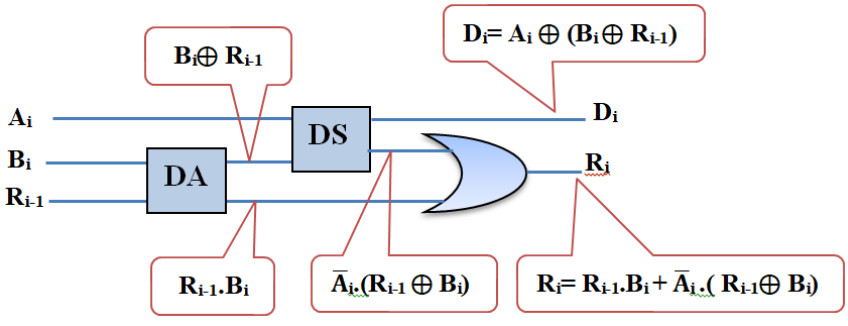
Une autre manière consiste à :

- Additionner Bi et Ri-1 avec un demi-additionneur (DA) (cette opération peut évidemment engendrer une retenue)
- Puis on retranche le résultat obtenu de Ai.

On peut écrire :

$$D_i = A_i \oplus (B_i \oplus R_{i-1})$$
$$R_i = R_{i-1}.B_i + \bar{A}_i.(R_{i-1} \oplus B_i)$$

- Circuit logique du soustracteur complet à l'aide d'un demi-additionneur et un demi-soustracteur



Additionneur Soustracteur :

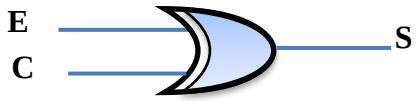
A. Circuit inverseur:

1. Table de vérité :

C	E	S
0	0	0
0	1	1
1	0	1
1	1	0

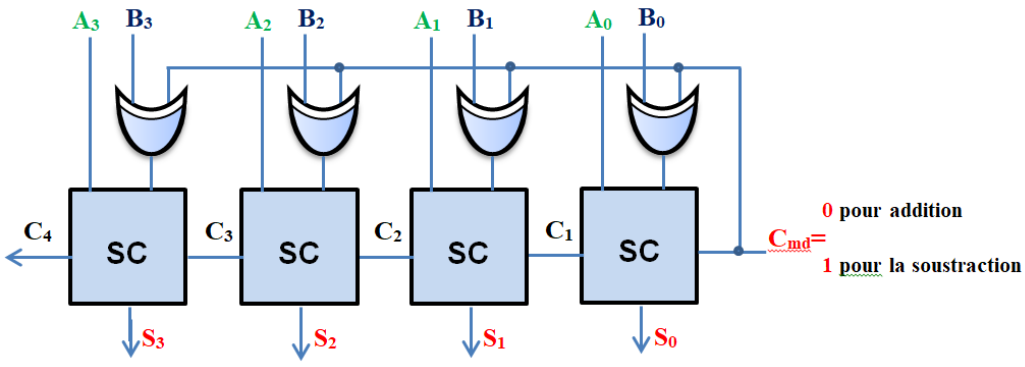
2. Equations :$S = E \oplus C$

3. Logigrammedu circuit inverseur



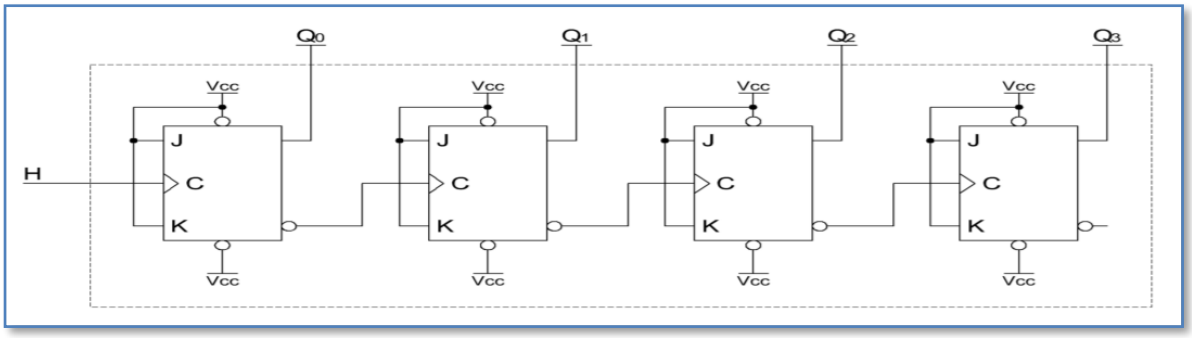
Pour calculer la différence **A - B** de deux nombre signés **A** et **B**, on utilise un circuit qui calcule d'abord l'opposé **-B** de **B** puis effectue la somme de **A** avec **-B** grâce à un additionneur. Le calcul de **-B** est réalisé en prenant la négation de **B** bit à bit puis en ajoutant **1** au résultat obtenu. Ce dernier **1** est en fait ajouté directement à la somme de **A** et **-B** en l'injectant comme retenue C_0 à l'additionneur. Le circuit ci-dessous effectue une somme ou une différence suivant la valeur de la commande Cmd. Si Cmd vaut 0, le circuit calcule la somme A + B. Si, au contraire, C_{md} vaut **1**, le circuit calcule la différence **A - B**. En effet, chacune des portes **XOR** effectue la négation ou non d'une entrée B_i suivant la valeur de C_{md}.

B. Circuit logique de l'additionneur Soustracteur :

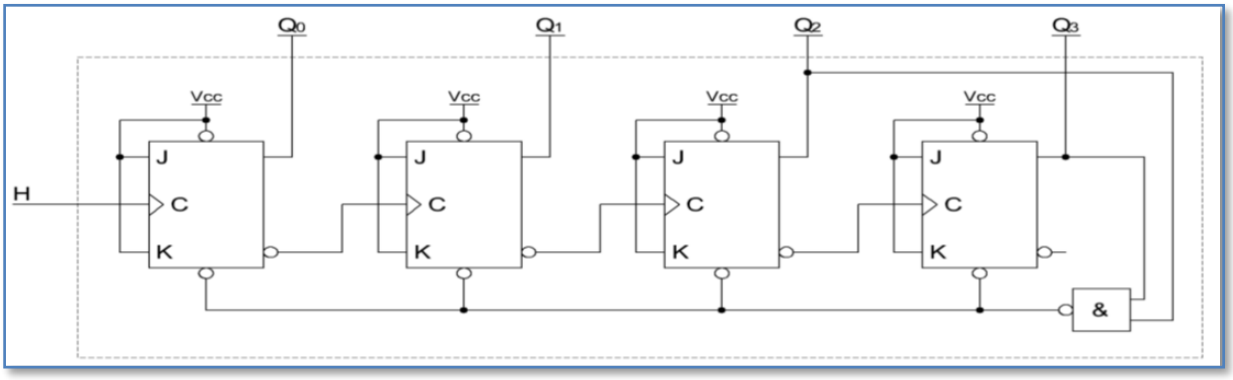


III. Circuits séquentiels

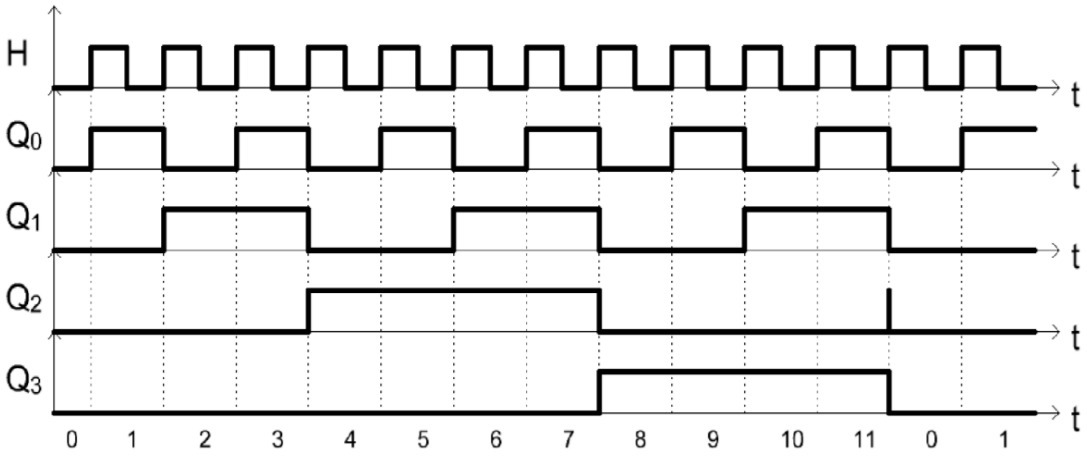
1. Circuit logique du compteur asynchrone modulo 16.



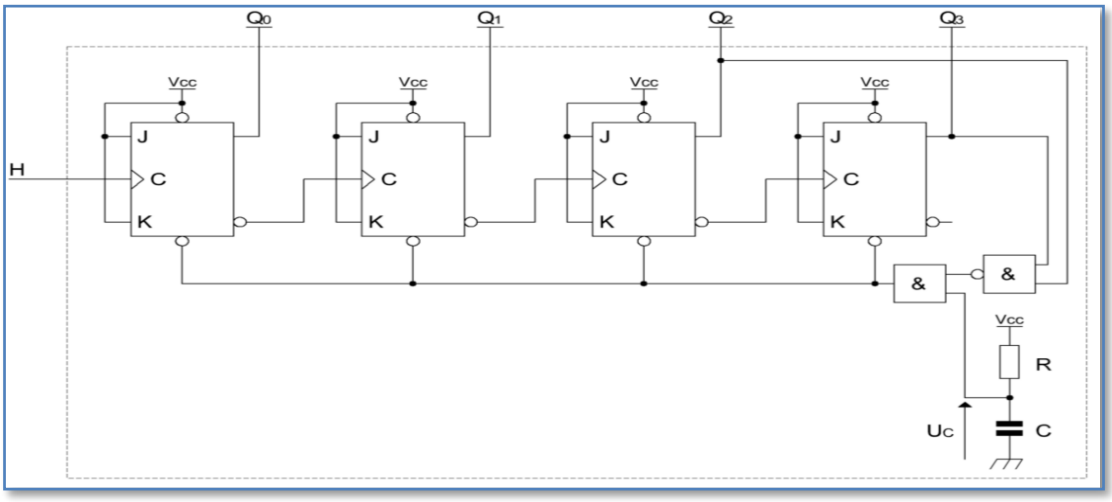
2. Circuit logique du compteur asynchrone modulo 12.



3. En partant de zéro, les chronogrammes du Compteur asynchrone modulo 12sur un cycle complet sont les suivants :

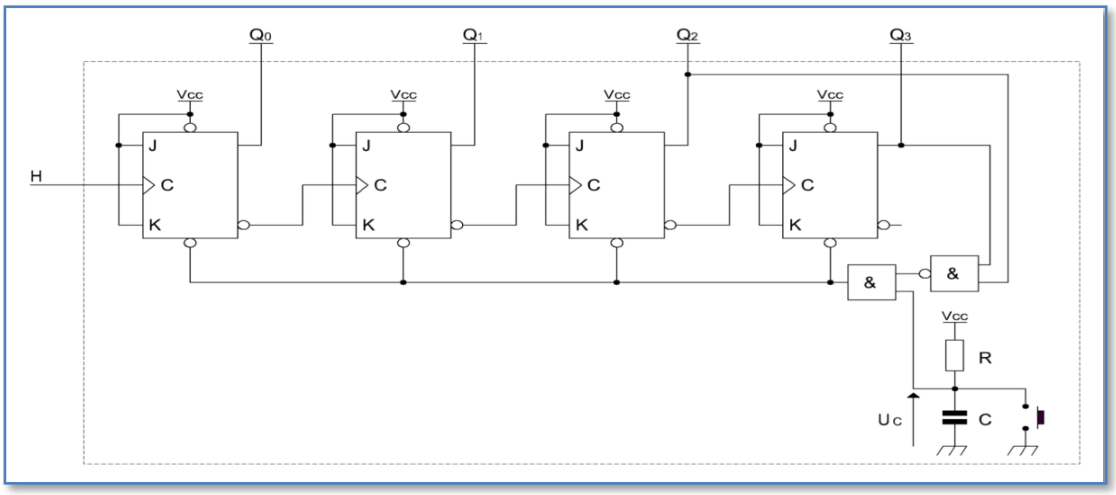


4. Interrupteur automatique de remise à zéro à l'allumage du Compteur asynchrone modulo 12



À l'allumage, le condensateur est déchargé et impose une tension de zéro volt à l'entrée de la porte ET. La porte considère cette tension nulle comme un niveau logique 0. Un *reset* est alors activé sur les bascules. Le condensateur se charge ensuite à travers la résistance jusqu'à la tension Vcc. Une fois que la tension a atteint un certain seuil (par ex : Vcc/2), la porte ET la considère comme un niveau logique 1. Le *reset* n'est plus imposé par la porte ET qui laisse passer le niveau de sortie de la porte NON-ET sur les entrées *reset* des bascules. Le compteur fonctionne alors dans son état normal.

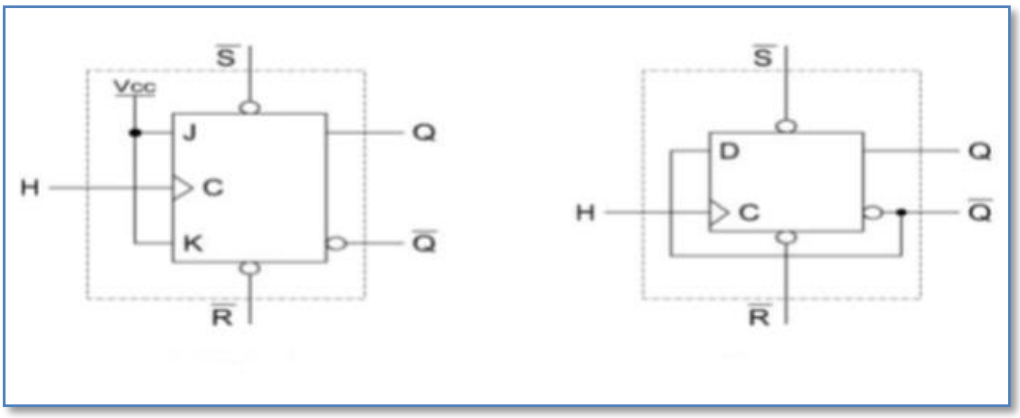
5. Circuit logique du Compteur asynchrone modulo 12 munit d'un Interrupteur manuel de remise à zéro à l'allumage



Décharger le condensateur permet de se replacer dans le même état qu'à l'allumage, et donc d'effectuer une remise à zéro. Pour décharger le condensateur, un simple bouton poussoir à ses bornes est suffisant. Un appui sur le bouton aura pour effet de court-circuiter le condensateur.

6. Remplacement des bascules JK par des bascules D

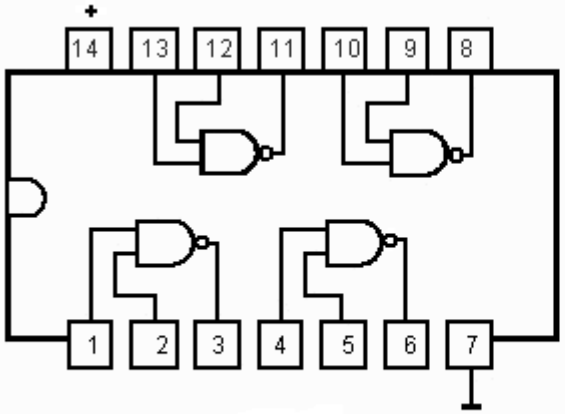
La cellule de base d'un compteur asynchrone est un diviseur de fréquence par deux. Il suffit donc de remplacer les bascules JK par des bascules D câblées en diviseur de fréquence par deux. Comme montrer dans la figureci-dessous:



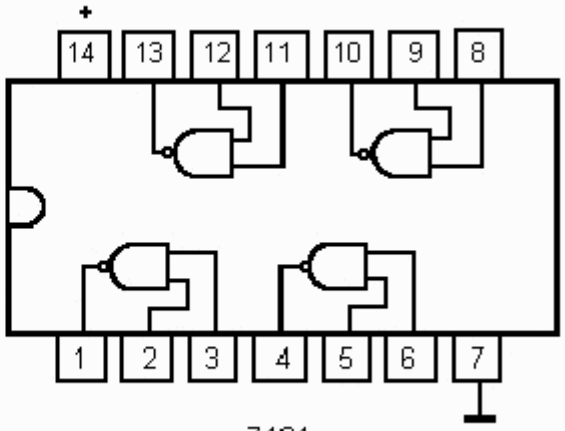
ANNEXE 2

BROCHAGES DES CIRCUITS INTEGRES

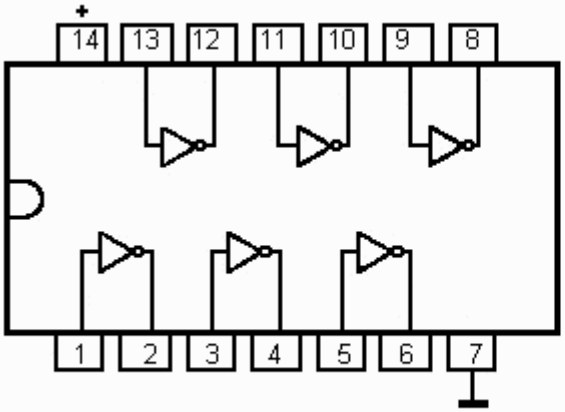
7400 N	Quadruple porte NON - ET à 2 entrées
7401 N	Quadruple porte NON - ET à 2 entrées avec collecteur ouvert
7402 N	Quadruple porte NON - OU à 2 entrées
7404 N	6 inverseurs
7408 N	Quadruple porte ET à 2 entrées
7410 N	Triple porte NON - ET à 3 entrées
7411 N	Triple porte ET à 3 entrée
7413 N	Double porte NON - ET à 4 entrées
7425 N	Double porte NON - OU à 4 entrées et strobe
7426 N	Quadruple porte NON - ET à 2 entrées - Haute tension
7427 N	Triple porte NON - OU à 3 entrées
7428 N	Quadruple porte NOR à 2 entrées
7430 N	Porte NON - ET à 8 entrées
7432 N	Quadruple porte OU à 2 entrées
7442 N	Décodeur décimal BCD
7443 N	Décodeur excès de 3 - décimal
7444 N	Décodeur excès de 3 Gray - décimal
7445 N	Décodeur BCD décimal
7447 N	Décodeur BCD à 7 segments
7448 N	Décodeur BCD 7 segments
7473 N	Flip-Flop maître esclave avec entrée reset
7474 N	Double Flip-Flop D synchrone
7475N	Quadruple Flip-Flop D asynchrone
7476 N	Double Flip-Flop JK maître esclave avec entrées set et reset
7480 N	Additionneur complet à 1 bit
7481 N	Mémoire à 16 bits écriture / lecture
7482 N	Additionneur complet à 2 bits
7483 N	Additionneur complet à 4 bits
7484 N	Mémoire à 16 bits écriture / lecture à 2 entrées d'écriture et de lecture
7485 N	Comparateur binaire à 4 bits
7486 N	Quadruple porte OU Exclusif
7489 N	Mémoire à 64 bits écriture / lecture à collecteur ouvert
7490N	Compteur décimal
7491 N	Registre à décalage à 8 bits série
7492 N	Diviseur par 12
7493 N	Compteur binaire
7494 N	Registre à décalage 4 bits à entrée parallèle
7495 AN	Registre à décalage 4 bits entrées et 4 sorties parallèles
7496 N	Registre à décalage 5 bits parallèle
7497 N	Diviseur de fréquence binaire synchrone programmable 6 bits
74100 N	Octo-Flip-flop D
74107 N	Double Flip-flop JK maître esclave avec entrée reset
74118 N	Sextuple Flip-flop RS à entrée de reset commune
74150 N	Sélecteur de données 16 bits / multiplexeur



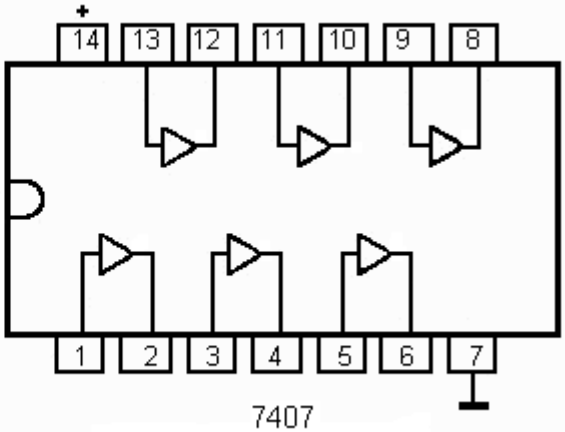
7400 - 7403 - 7437



7401

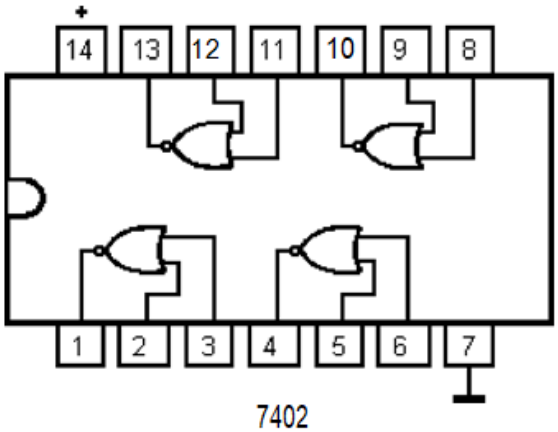


7404 - 7405 - 7406 - 7416

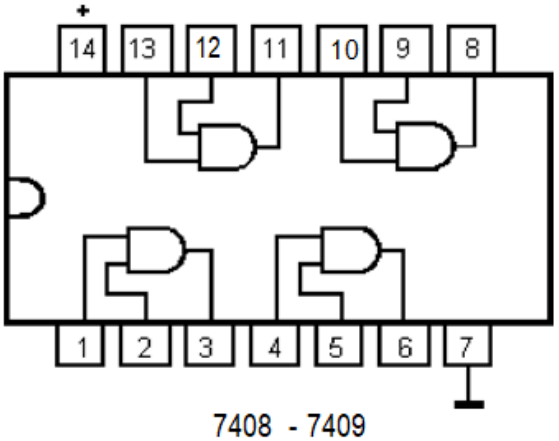


7407

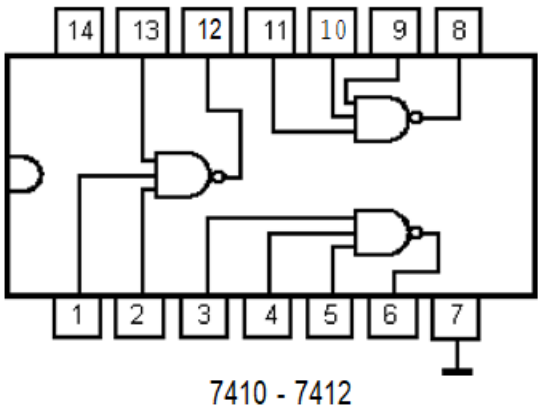
| 74198 N | **Registre à décalage synchrone 8 bits à entrée et sortie parallèles**



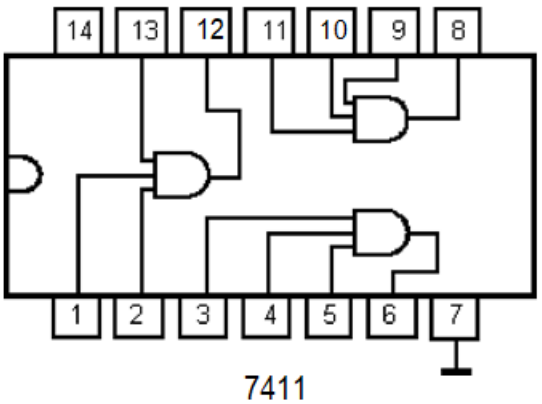
7402



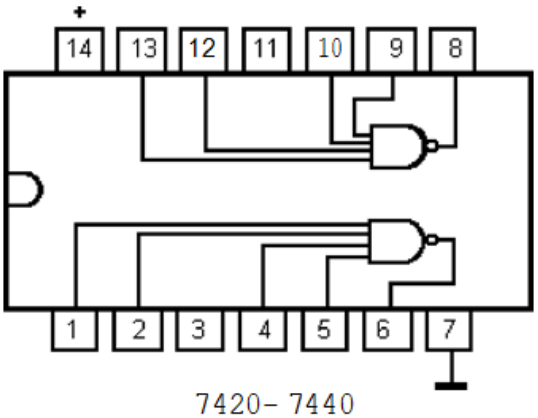
7408 - 7409



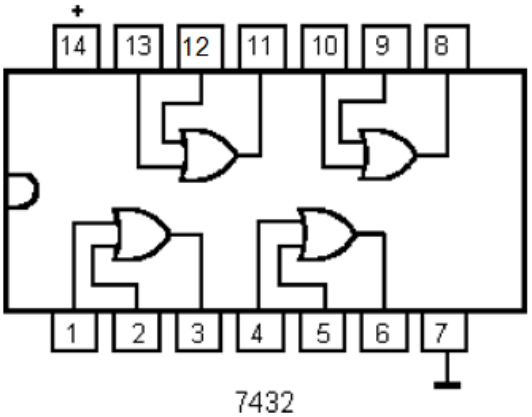
7410 - 7412



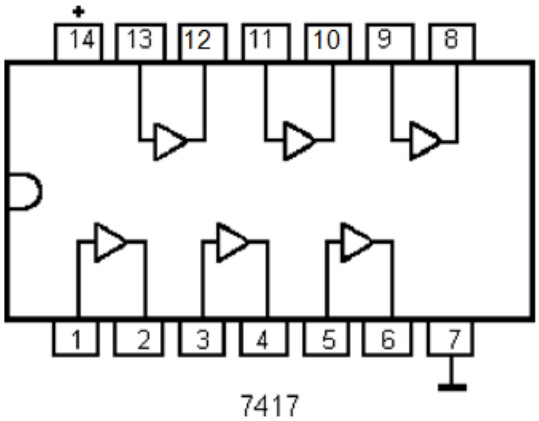
7411



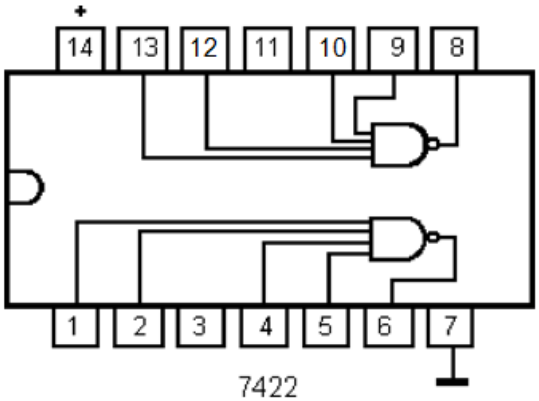
7420– 7440



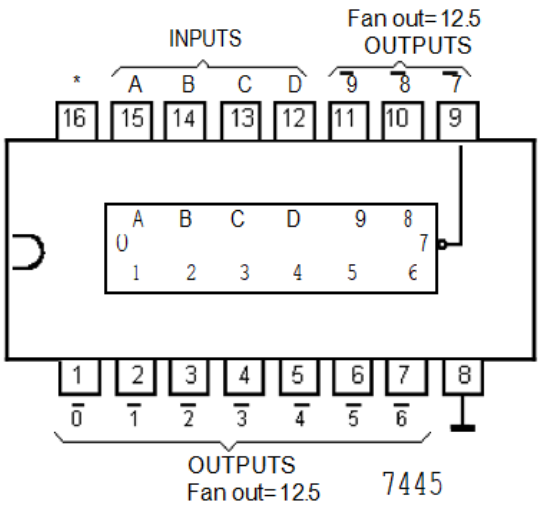
7432



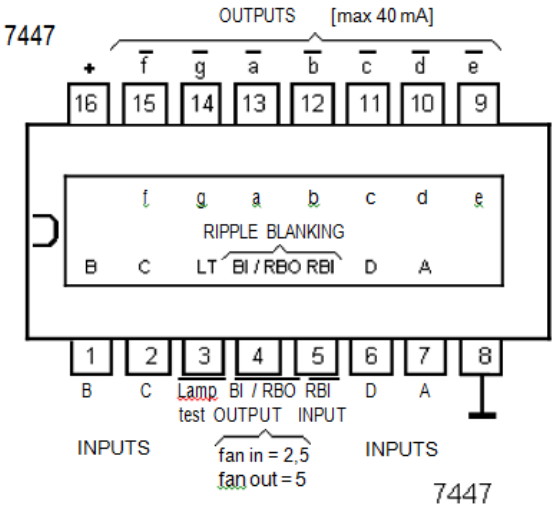
7417



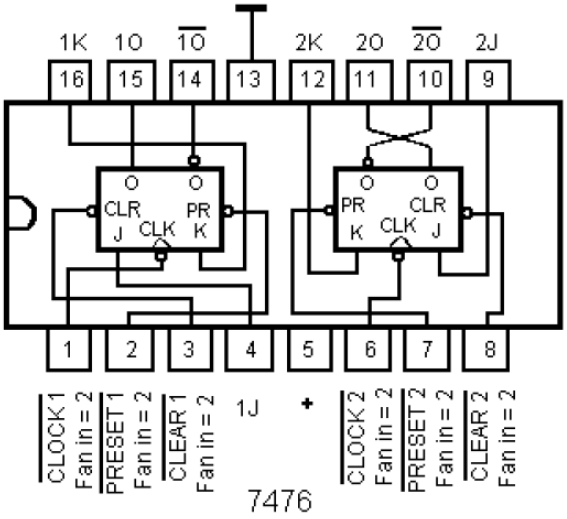
7422



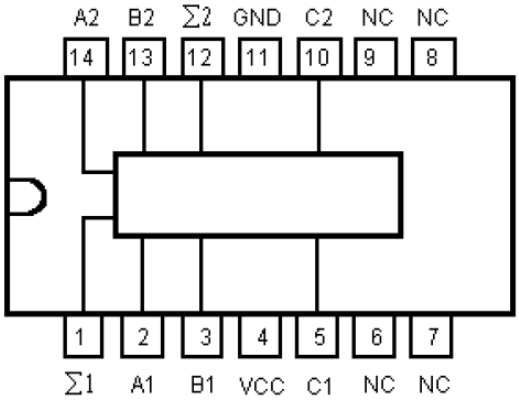
7445



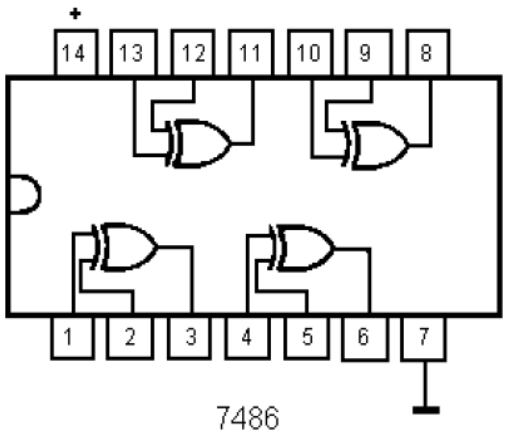
7447



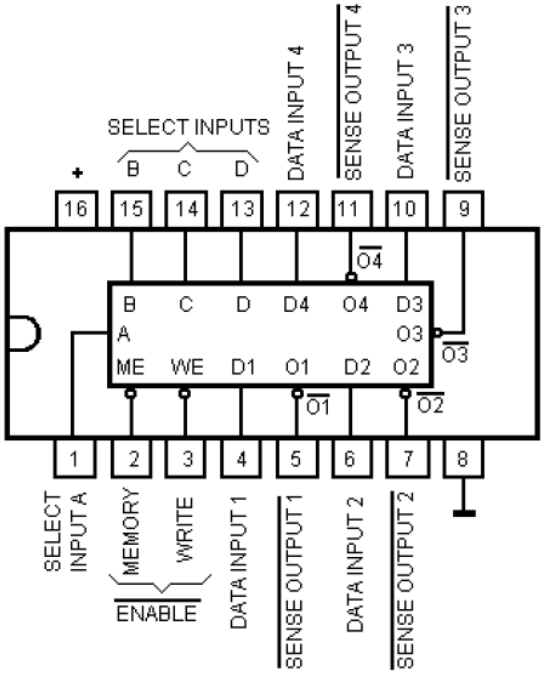
7476



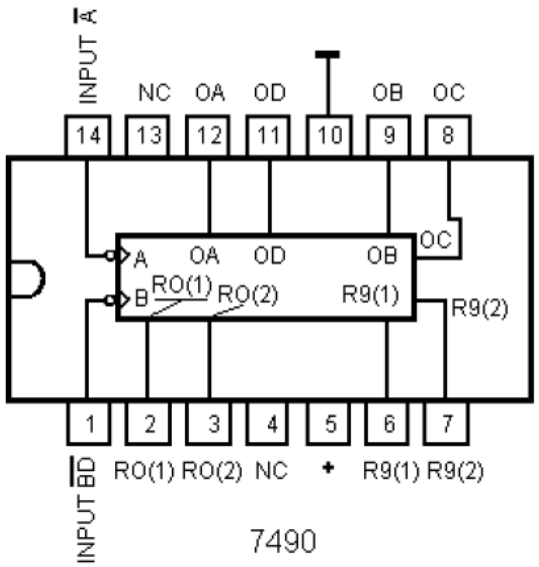
7482



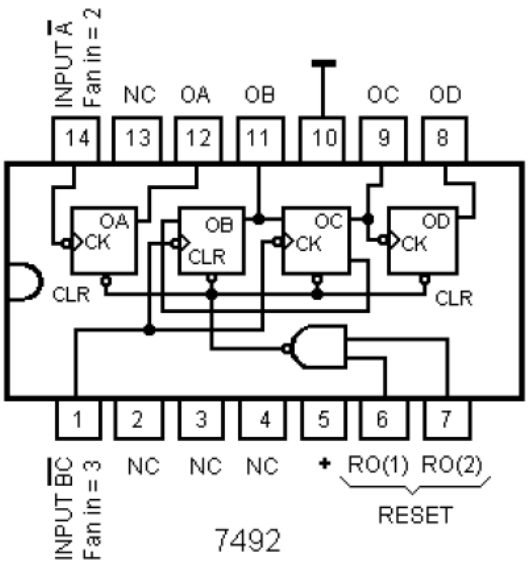
7486



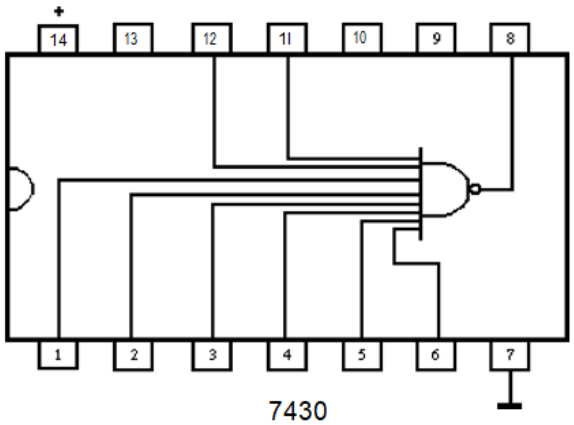
7489



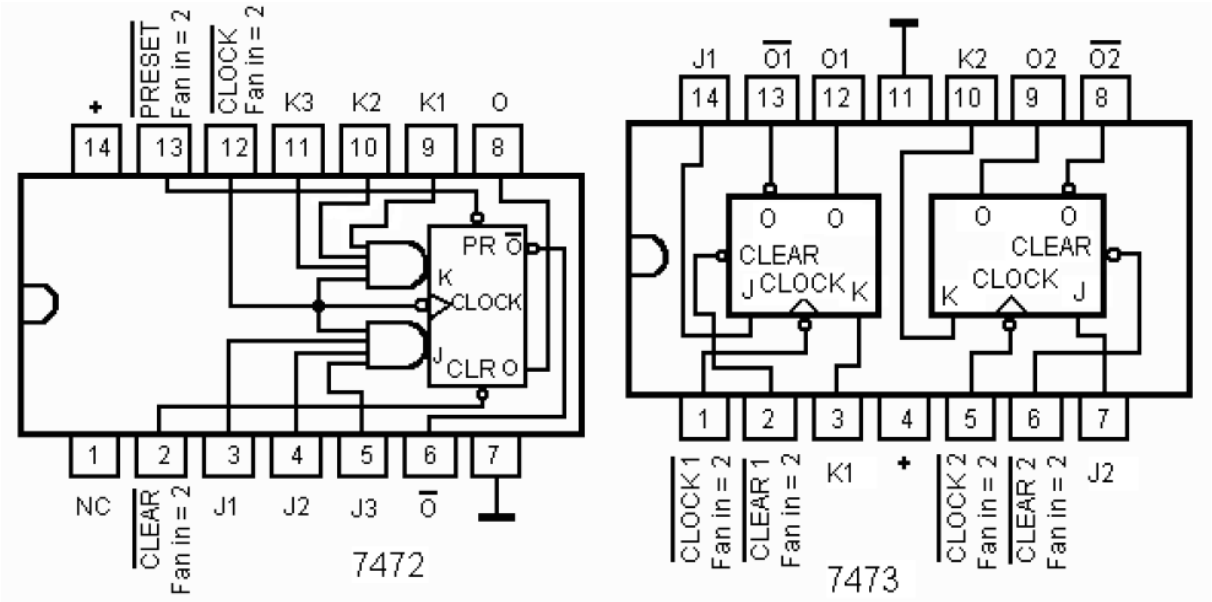
7490

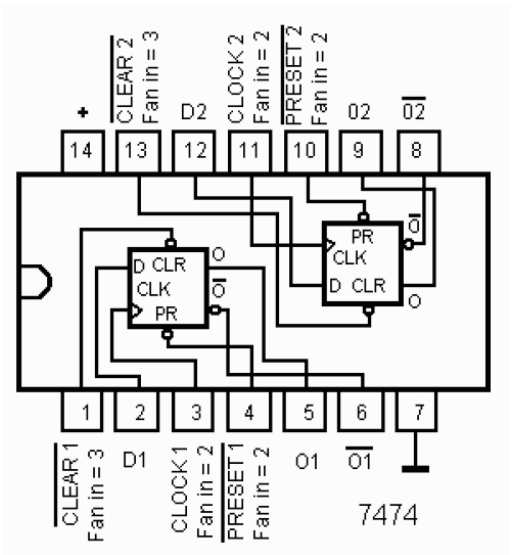


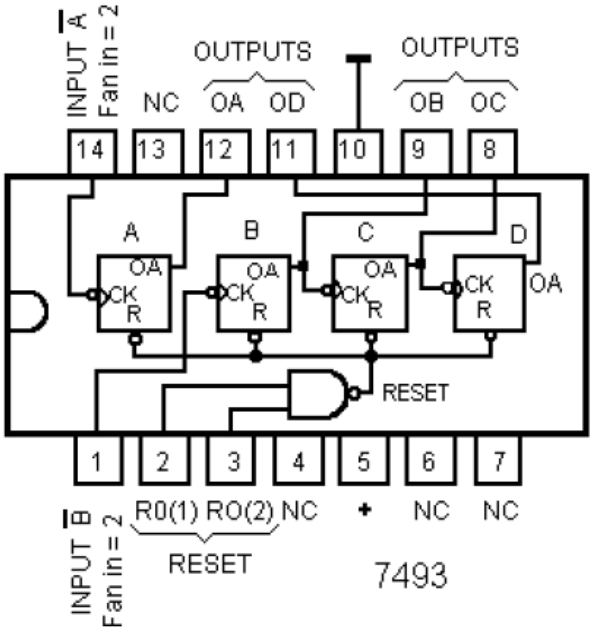
7492


+
14 13 12 11 10 9 8
1 2 3 4 5 6 7

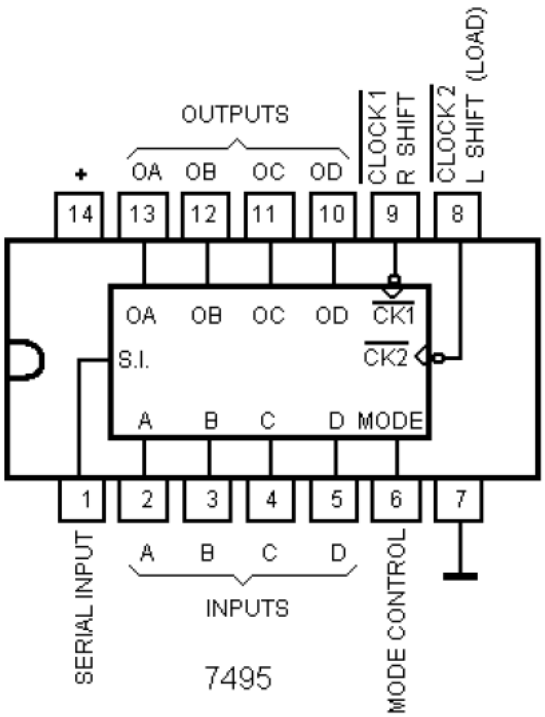
7430


PRESET Fan in = 2
CLOCK Fan in = 2
K3 K2 K1 O
PR
K
CLOCK
J
CLR
NC
CLEAR Fan in = 2
J1 J2 J3
7472
J1 O1 O1 K2 O2 O2
CLEAR
CLOCK
CLOCK 1 Fan in = 2
CLEAR 1 Fan in = 2
K1
CLOCK 2 Fan in = 2
CLEAR 2 Fan in = 2
J2
7473


CLEAR 2 Fan in = 3
D2
CLOCK 2 Fan in = 2
PRESET 2 Fan in = 2
O2 O2
D CLR
CLK
PR
CLEAR 1 Fan in = 3
D1
CLOCK 1 Fan in = 2
PRESET 1 Fan in = 2
O1 O1
7474
OUTPUTS
ENABLE 1 - 2 Fan in = 4
O1 O2 O2
O3 O3 O4
Q2
O3
E
D1 D2 E D3 D4
O1 O4
OUTPUT
INPUTS Fan in = 2
ENABLE 3 - 4 Fan in = 4
7475

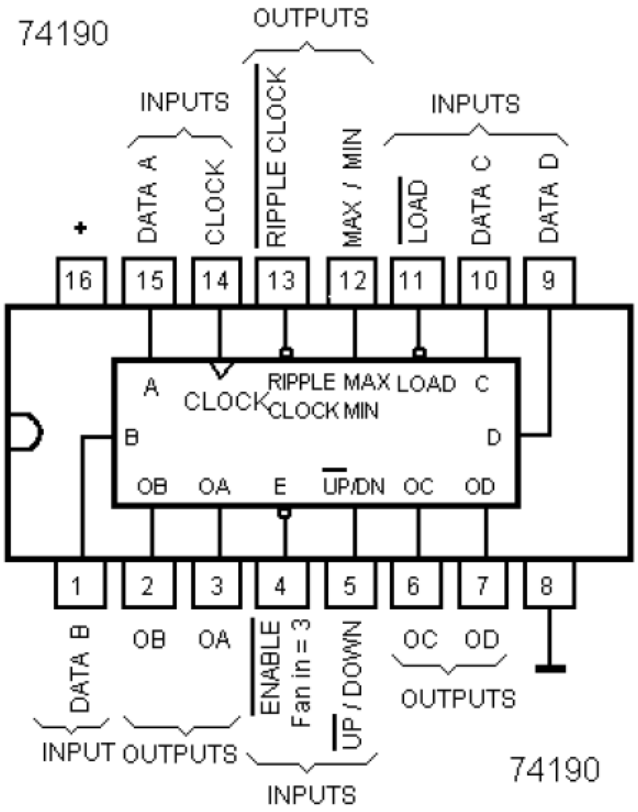


7493

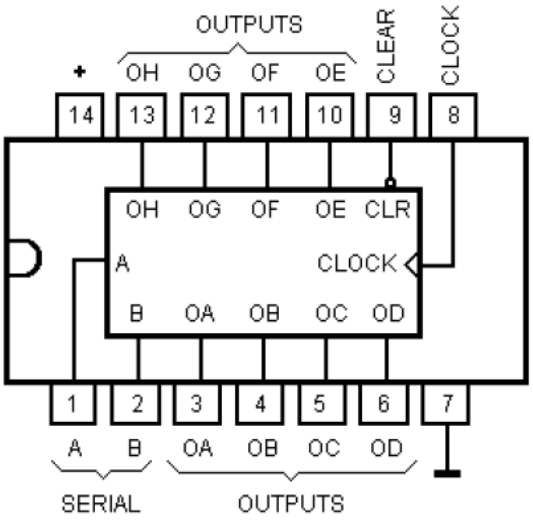


7495

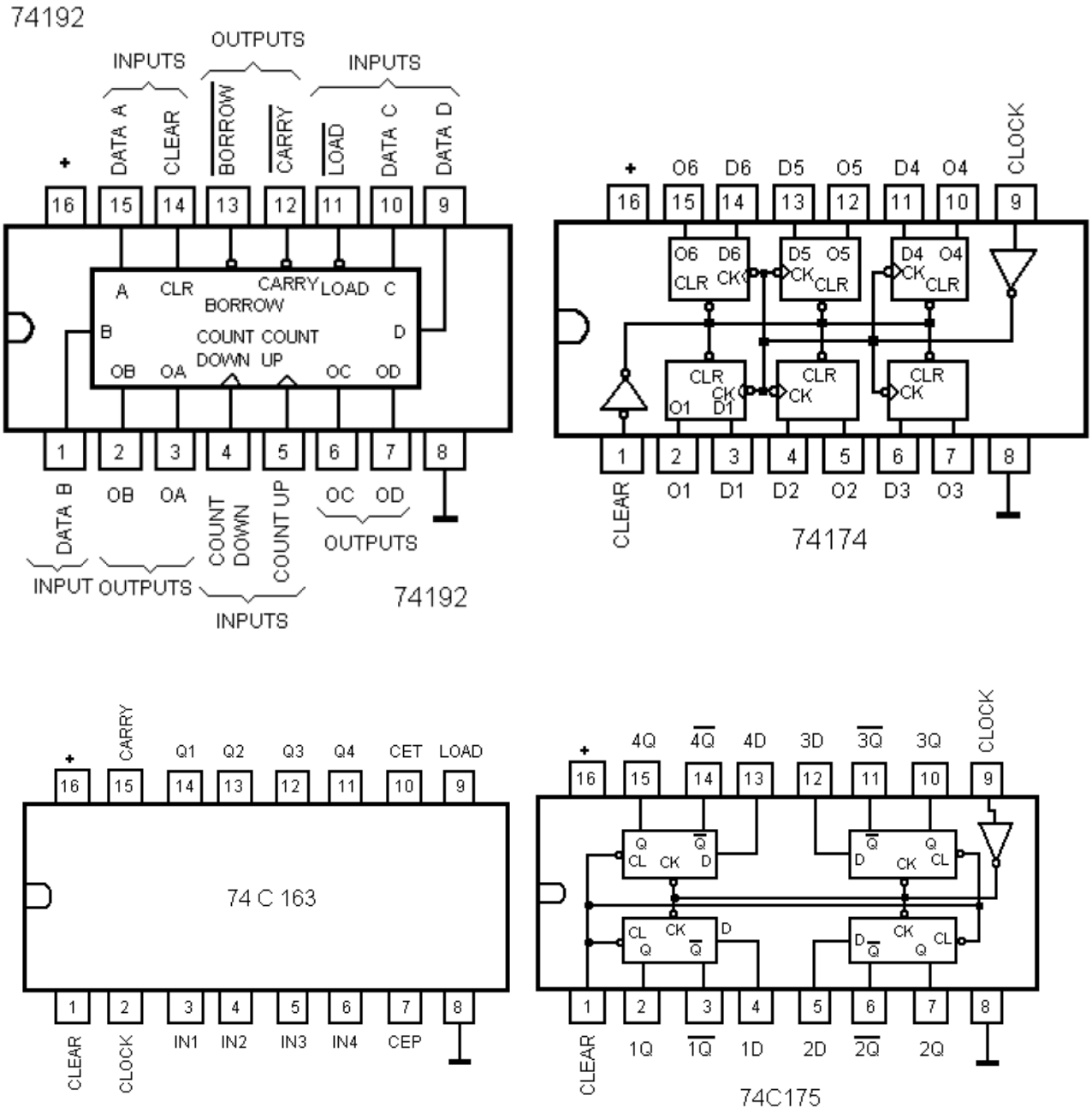
74190



74190



74164



Source : https://www.electronique-et-informatique.fr/Electronique-et-Informatique/Digit/Data_book.php