Université Mohammed V
Faculté des Sciences
Département de Mathématiques et Informatique
Filière : SMI

Algèbre binaire et Circuits logiques

(2007-2008)

Prof. Abdelhakim El Imrani

Plan

1. Algèbre de Boole
2. Circuits logiques
3. Circuits combinatoires
4. Circuits séquentiels

Algèbre de Boole

George Boole (1815-1864) est un mathématicien autodidacte anglais qui voulait faire un lien entre la logique (étude de la validité du raisonnement) et la représentation symbolique utilisée en mathématique.

Il a écrit deux ouvrages sur le sujet :

- ***Mathematical Analysis of Logic*** **(1847)**
- ***An Investigation of the Laws of Thought*** **(1854)**

Ces travaux n'ont pas connu d'intérêt particulier auprès de la communauté mathématique et scientifique de son époque, mis à part chez les logiciens

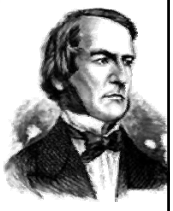
Algèbre de Boole

- C'est 70 ans plus tard que les travaux de Boole gagnent l'intérêt de tous, lorsque Claude Shannon fait le lien entre l'algèbre de Boole et la conception des circuits.
- Claude Shannon montre que l'algèbre de Boole peut-être utilisée pour **optimiser** les circuits. Cette nouvelle avenue de recherche va ouvrir la voie à l'ère numérique.

  → « En utilisant **l'algèbre de Boole** avec le **système binaire**, on peut concevoir des **circuits** capables d'effectuer des **opérations arithmétiques et logiques**
- Boole repose sur des axiomes, des postulats et des théorèmes qu'il faut connaître par coeur !

Algèbre de Boole

Propositions vraie ou fausses et opérateurs sur ces préposition ⟶ Algèbre de Boole



- Systèmes binaires: Vrai=1, Faux=0
- C'est le cas des systèmes numériques (circuits logiques)

- L'ordinateur est constitué de circuits logiques
- Élément de base est le transistor, deux états: Bloqué=0, Conducteur=1.

Transistor → Porte logique → Circuit logique
→ Unité d'un système informatique

Algèbre binaire

Définitions:

- **États logiques** : 0 et 1, Vrai et Faux

- **Variable logique** : Symbole pouvant prendre comme valeur des états logiques (A, b, c, ...)

- **Opérateurs logiques**: Or, And, Not, ...

- **Fonction logique** : Expression de variables et d'opérateurs logiques. (f = not(a) or (b OR c and d)

Éléments de base

- **Variables d'entrée**
  - Les variables d'entrée sont celles sur lesquelles on peut agir directement. Ce sont des variables logiques indépendantes.
- **Variable de sortie**
  - Variable contenant l'état de la fonction après l'évaluation des opérateurs logiques sur les variables d'entrée.
- **Simplification d'une fonction logique**
  - Trouver la représentation (l'écriture) la plus simple de la fonction réalisée: ***Algèbre de Boole***

Algèbre de Boole sur [0,1] = algèbre binaire

Structure d'algèbre de boole

- 2 lois de composition interne (Or, And)
- 1 application unaire (Not)

2 Lois de Composition Interne : ET, OU

Somme (OU, Réunion) $s = a + b$ = a **or** b

Produit (ET, intersection) $s = a \cdot b = ab$ = a **and** b

Nb: a+b se lit « a OU b » pas « a PLUS b »

Application unaire :

Not (complémentation, inversion) $s = \overline{a}$ = **not**(a)

NB: $\overline{a}$ se lit « a barre » ou « non a »

Fonctions logiques

Fonction logique à n variables f(a,b,c,d,…,n)

$[0,1]^n \longrightarrow [0,1]$

- Une fonction logique ne peut prendre que deux valeurs (0, 1)
- Les cas possibles forment un ensemble fini (card = 2^n)

La table de fonction logique = table de vérité

Définition : (a, b, c, …, n) = vecteur d'entrée

Table de vérité

• **Table de vérité**: Enumération ligne par ligne des valeurs prises par f en fonction des valeurs de ses paramètres.

Or

s = a + b

S est vrai si a OU b est vrai.

a	b	s
0	0	0
0	1	1
1	0	1
1	1	1

And

s = a . b

S est vrai si a ET b sont vrais.

a	b	s
0	0	0
0	1	0
1	0	0
1	1	1

Not

s = $\overline{a}$

S est vrai si a est faux

a	s
0	1
1	0

Notes sur les tables de vérité

f (a, c, d, .., n) fonction logique à N entrées
sera représentée par :

- une table à 2^N lignes

a b c	f(a,b,c)
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	0
1 1 0	0
1 1 1	1

Propriétés

- **Commutativité**
- a+b = b+a
- a.b = b.a
- **Associativité**
- 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)

- **Idempotence**
  - a+a = a
  - a.a = a
- **Absorption**
  - a+a.b = a
  - a.(a+b) = a

Démonstration distributivité

?

a.(b+c) = a.b+a.c

a	b	c	b+c	a.(b+c)	a.b	a.c	a.b+a.c
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

= ?

Propriétés (2)

- **Élément neutre**
- a+0 = a
- a.1 = a
- **Élément absorbant**
- a+1 =1
- a.0 = 0
- **Inverse**
- $a+\overline{a} = 1$
- $a.\overline{a} = 0$

- **Théorème de DE Morgan**
  - $\overline{a+b} = \overline{a} . \overline{b}$
  - $\overline{a.b} = \overline{a} + \overline{b}$

Équations logiques

On exprime f(a, b, c, ...) par une expression en a, b, c.. et des opérateurs logiques.

Exemple: $f = \bar{a}+b.c.(\bar{d}+e)$

Principe de dualité: Une expression reste vraie si on interverti les 1 par des 0 et les ET par des OU

Exemple: si $a+b=1$ alors $\bar{a}.\bar{b}=0$

Je suis riche si je suis bien payé et que je ne dépense pas tout mon argent = Je suis pauvre si je ne suis pas bien payé ou que je dépense tout mon argent

Les opérateurs NAND, NOR

a	b	s
0	0	1
0	1	1
1	0	1
1	1	0

$s = \overline{a.b}$

S est vrai si a OU b est faux.

NAND (No-AND)

a	b	s
0	0	0
0	1	0
1	0	0
1	1	1

$s = \overline{a+b}$

S est vrai si ni a, ni b ne sont vrais.

NOR (No-OR ou NI)

L'opérateur : XOR

a	b	s
0	0	0
0	1	1
1	0	1
1	1	0

$s = a \oplus b = \bar{a}.b + a.\bar{b}$

S est vrai si a OU b
est vrai mais pas les deux.

XOR (Ou-Exclusif) vaut 1 si a est différent de b
Opérateur de différence (disjonction)

Propriétés du XOR

XOR est associatif s = a ⊕ b ⊕ c ⊕..... n

vaut 1 si le nombre de variable à 1 est impaire.

$$s = \overline{a \oplus b} = \bar{a} \oplus b = a \oplus \bar{b} = a \text{ XNOR } b$$

$$\text{XNOR} = \overline{\text{XOR}} \text{ vaut 1 si } a = b$$

$a \oplus 1 = \bar{a}$ $a \oplus 0 = a$

Propriétés

$$a \oplus c = b \oplus c \Leftrightarrow a = b$$
$$a \oplus x = b \Leftrightarrow x = a \oplus b$$

Écriture des équations logiques

Définitions: Apparition d'une variable = **Lettre**
Produit de variables sous forme simple ou complémentées = **Monôme**
Somme de monômes = **Polynôme**

$z = a + \overline{b.c.(d + e)}$ Expression algébrique

$= a + \overline{b} + \overline{c} + \overline{(d + e)}$ Développement

$= a + \overline{b} + \overline{c} + \overline{d} \,.\, \overline{e}$ Polynôme de 4 monômes de 1 et 2 lettres

Fonctions logiques et formes canoniques

f fonction logique de n variables

- On appelle «***minterme***» de n variables, l'un des produits de ces variables ou de leurs complémentaires.
- On appelle «***maxterme***» de n variables, l'une des sommes de ces variables ou de leurs complémentaires.

exemple $n = 4$ variables $\{a, b, c, d\}$

$m = a \cdot b \cdot c \cdot d$ est un minterme

$m = \overline{a} \cdot \overline{b} \cdot c \cdot d$ est un autre minterme

$m = a \cdot \overline{b} \cdot c$ n'est pas un minterme

$M = a + b + c + d$ est un maxterme

$M = \overline{a} + \overline{b} + c + d$ est un autre maxterme

$M = a + \overline{b} + c$ n'est pas un maxterme

Formes canoniques

Une fonction est sous **forme canonique** (ou **normale**) si chaque terme contient toutes les variables. L'écriture sous forme canonique est unique.

Exemples :

$$f(x,y,z) = x.y.z + \bar{x}.\bar{y}.z + \bar{x}.y.z$$

Minterme

Première forme canonique ou forme normale disjonctive

$$f(x,y,z) = (x + \bar{y} + z).(\bar{x} + y + \bar{z})$$

Maxterme

Deuxième forme canonique ou forme normale conjonctive

Formes canoniques

Si la fonction n'est pas sous forme normale

i.e. une des variables (au moins) ne figure pas dans un des termes

⟶ La fonction est sous une forme simplifiée

$$f(x,y,z) = xyz + \bar{x}y\bar{z} + xy\bar{z}$$ Première forme canonique

$$= xy(z + \bar{z}) + \bar{x}y\bar{z}$$ Forme simplifiée

$$= y(x + \bar{x}\bar{z})$$ Forme simplifiée

$$= y(x + \bar{z})$$ Forme simplifiée

Formes canoniques: Choix

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

Simplification des fonctions

Objectif : Fabriquer un système
- à moindre coût
- rapide
- fiable
- peu consommateur

Méthodes : Algébriques
Graphiques
Programmables

Résultat : on cherche la forme minimale d'une fonction
nombre minimal de monômes/nombre minimal de lettre par monôme

Possibilité de plusieurs formes minimales: formes équivalentes

Simplification algébrique

Applications des principes et propriétés de l'algèbre de Boole

Identités remarquables :

1	$a.b+\bar{a}.b=b$	$(a+b).(\bar{a}+b)=b$
2	$a+a.b=a$	$a.(a+b)=a$
3	$a+\bar{a}.b=a+b$	$a.(\bar{a}+b)=a.b$

Démonstrations : 1 et 2 trivial

3 : $a+\bar{a}.b=\underbrace{a.a+a.b}_{a}+\underbrace{a.\bar{a}}_{0}+\bar{a}.b=(a+\bar{a}).(a+b)=a+b$

Simplification algébrique

Règles de simplification :

(Mintermes adjacents = 1 seule variable qui change)

1 : Deux mintermes adjacents → Il reste l'intersection commune

1': Deux maxtermes adjacents → Il reste la réunion commune

$$a.b.c+a.b.\bar{c}=a.b.(c+\bar{c})=a.b$$

$$(a+b+c).(a+b+\bar{c})=(a+b)(c+\bar{c})=a+b$$

2: On peut ajouter un terme déjà existant à une expression logique.
→ pas de coefficient en algèbre de Boole.

3: On ne change pas le résultat en multipliant l'un des termes par 1 ou en ajoutant 0.

Méthode algébrique toujours possible mais démarche intuitive qui dépend de l'habileté et de l'expérience.

Exercice 1

Remplissez la table de vérité suivante pour prouver le théorème de DeMorgan :

x	y	xy	$\overline{xy}$	$\overline{x}$	$\overline{y}$	$\overline{x}+\overline{y}$
0	0	**0**	**1**	**1**	**1**	**1**
0	1	**0**	**1**	**1**	**0**	**1**
1	0	**0**	**1**	**0**	**1**	**1**
1	1	**1**	**0**	**0**	**0**	**0**

Exercice 2

Considérons la fonction F définie par la table de vérité suivante :

x	y	z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Mintermes

$$F = \overline{x}\,y\,z + x\,\overline{y}\,z + x\,y\,\overline{z} + x\,y\,z$$

$$\begin{aligned} F &= \overline{x}\,y\,z + x\,\overline{y}\,z + x\,y\,\overline{z} + x\,y\,z \\ &= (\overline{x}\,y\,z + x\,y\,z) + (x\,\overline{y}\,z + x\,y\,z) + (x\,y\,\overline{z} + x\,y\,z) \\ &= y\,z\,(\overline{x} + x) + x\,z\,(\overline{y} + y) + x\,y\,(\overline{z} + z) \\ &= x\,y + y\,z + z\,x \end{aligned}$$

Exercice 3

- On désire concevoir un circuit qui permet de gérer les notes des examens, on donne: Examen final (45 %), Examen Partiel (35 %), TPs (20 %).
- Un étudiant est admis s'il dispose d'un pourcentage >= 55 %).
  - Exemple: Final=11, Partiel=8, Tps=10 → F=1, P=0, T=1 ⇒ Pourcentage = 65 % → R=1 (étudiant admis).

- Donner la table de vérité.
- Donner la fonction logique correspondante. Simplifier le fonction obtenue.

Simplification graphique: Karnaugh

- La méthode de Karnaugh permet de visualiser une fonction et d'en tirer naturellement une écriture simplifiée.
- L'élément de base de cette méthode est la table de Karnaugh qui représente toutes les combinaisons d'états possibles pour un nombre de variables donné.
- La table de Karnaugh est un outil graphique qui permet de simplifier de manière méthodique des expressions booléennes.
- La construction des tables de Karnaugh exploite le codage de l'information et la notion d'adjacence

Karnaugh – simplification graphique

Principe:

Mettre en évidence sur un graphique les mintermes (ou maxtermes) adjacents. Transformer les adjacences logiques en adjacences «géométriques».

Trois phases:

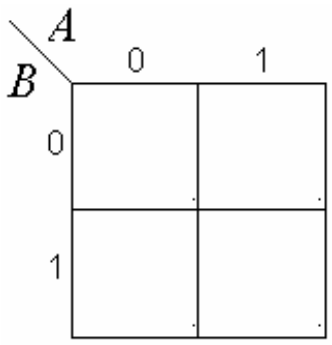
Transcrire la fonction dans un tableau codé, recherche des adjacents pour simplification équations des groupements effectués

Description: **Table de vérité** vs **Tableau de Karnaugh**

1 ligne	1 case
n variables	2^n cases

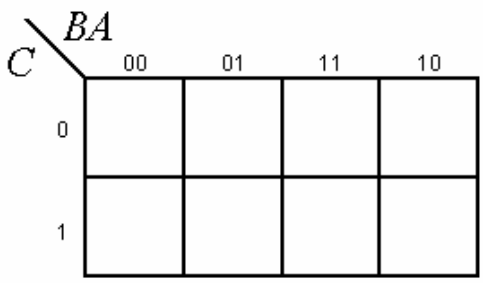
Diagrammes de Karnaugh

- Avec n = 2:
  - Entrées A et B
  - 4 cases



Diagrammes de Karnaugh

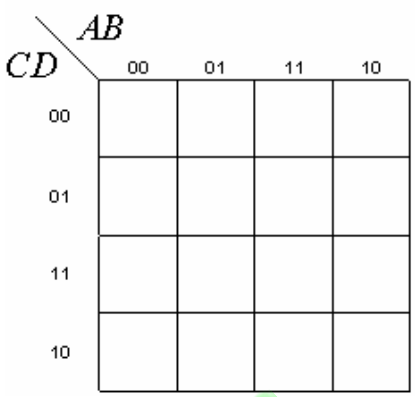
- **Avec n = 3:**
  - Entrées C, B et A
  - 8 cases



Remarque: Une seule variable change d'état entre 2 cases adjacentes

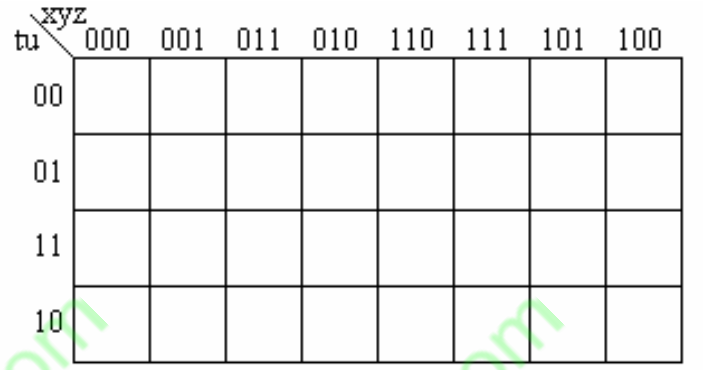
Diagrammes de Karnaugh

- **Avec n = 4:**
  - Entrées D, C, B et A
  - 16 cases



Diagrammes de Karnaugh

- **Avec n = 5:**
  - Entrées x, y, z, t et u
  - 32 cases



Simplification graphique

Exemple: Depuis une table de vérité

a b c	f
0 0 0	0
0 0 1	1
0 1 0	1
0 1 1	1
1 0 0	0
1 0 1	0
1 1 0	0
1 1 1	0

a \ bc	00	01	11	10
0	0	1	1	1
1	0	0	0	0

Exemple (Karnaugh)

Entrées			Sortie
C	B	A	S
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

TABLE DE VÉRITÉ

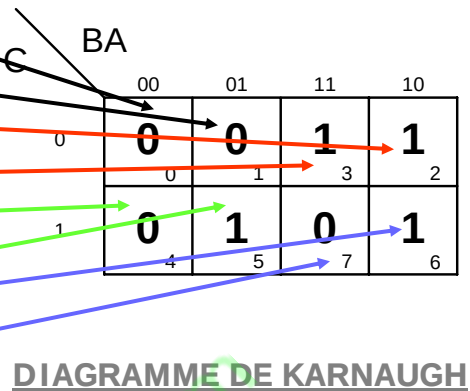
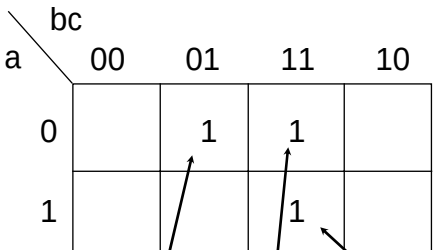


DIAGRAMME DE KARNAUGH

Simplification graphique

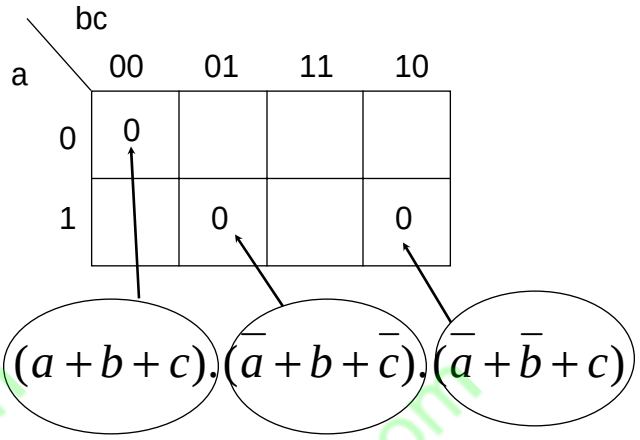
Exemple 2: Par une première forme canonique (Par les 1)



$$f(a,b,c) = \bar{a}.\bar{b}.c + \bar{a}.b.c + a.b.c$$

Simplification graphique

Exemple 2: Par une deuxième forme canonique (Par les 0)



$$f(a,b,c) = (a+b+c).(\bar{a}+b+\bar{c}).(\bar{a}+\bar{b}+c)$$

Simplification graphique

Règles de simplification

1 : Les groupements comportent une puissance de deux cases,

2 : Les 2^k cases forment un rectangle,

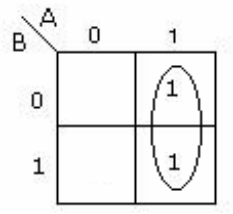
3 : On élimine variable(s) qui change(nt) d'état
Groupement de 2^k cases → On élimine k variables
2 cases → on élimine 1 variable;
4 cases → on élimine 2 variables;
8 cases → on élimine 3 variables;

4 : Il faut utiliser au moins une fois chaque 1, le résultat est donné par la réunion logique de chaque groupement,

5 : Expression minimale si :
- les groupements les plus grands possibles
- utiliser les 1 un minimum de fois

Exemple 1

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

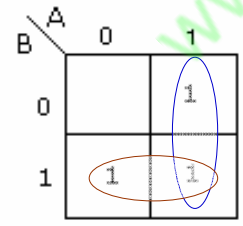


$S = AB + A\overline{B}$, simplification algébrique → $S = A (B + \overline{B}) = A$

Karnaught:
Groupement de 2 cases: on élimine variable qui change d'état (B) → S=A

Exemple 2

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



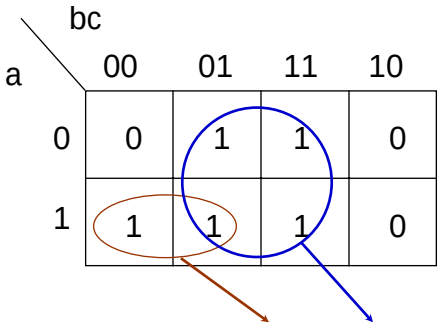
Premier groupement: On élimine B

Deuxième groupement: On élimine A

→ S = A + B

Exemple 3

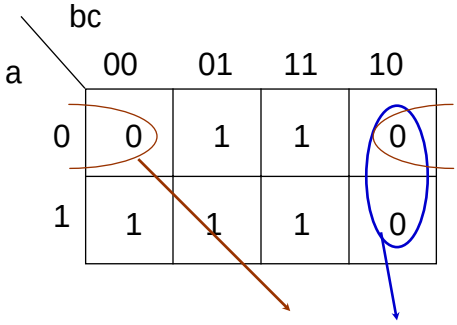
Tous les 1 sont groupés !



Equation : $F(a,b,c) = a.\bar{b} + c$

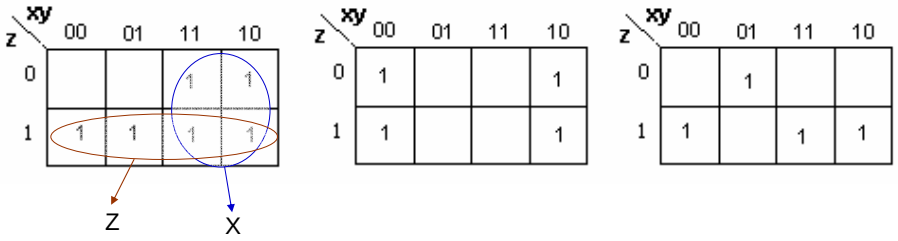
Exemple 4

Par les 0



Equation : $F(a,b,c) = (a+c).(\bar{b}+c)$

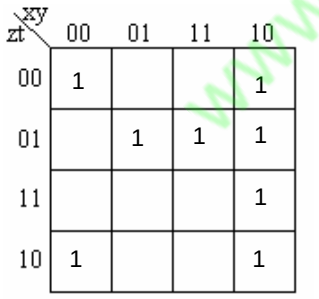
Exemple 5

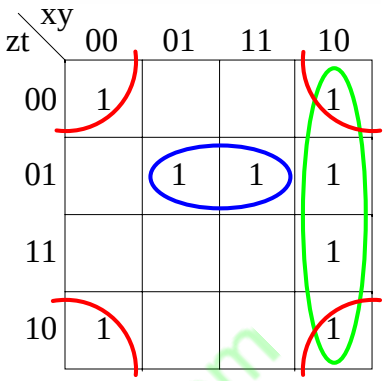


S = x + Z

Exemple 6

x	y	z	t	S
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0





Exercice 1

CD \ BA	00	01	11	10
00	1	1		
01	1	1		1
11	1	1		1
10	1	1		

CD \ BA	00	01	11	10
00	1	1		1
01	1	1		
11	1	1		
10	1	1		1

Exercice 2

CD \ BA	00	01	11	10
00			1	
01	1	1	1	1
11		1	1	1
10				

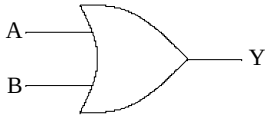
CD \ BA	00	01	11	10
00			1	
01	1	1	1	
11		1	1	1
10		1		

Circuits logiques

Circuit logique = Ensemble de portes logiques reliées entre elles correspondant à une expression algébrique.

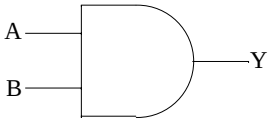
Porte logique (correspond à un opérateur logique)

Porte Or



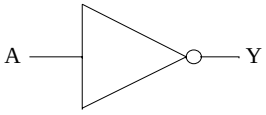
$Y = A + B$

Porte And



$Y = A \cdot B$

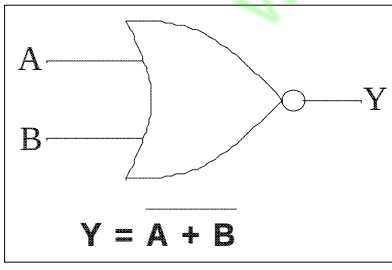
Porte Not



$Y = \overline{A}$

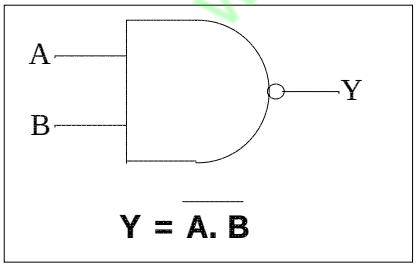
Portes dérivées

Porte Nor



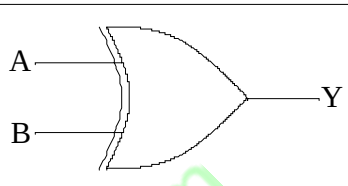
$Y = \overline{A + B}$

Porte Nand



$Y = \overline{A.B}$

Porte Xor



$Y = A \oplus B$

Conception d'un circuit logique

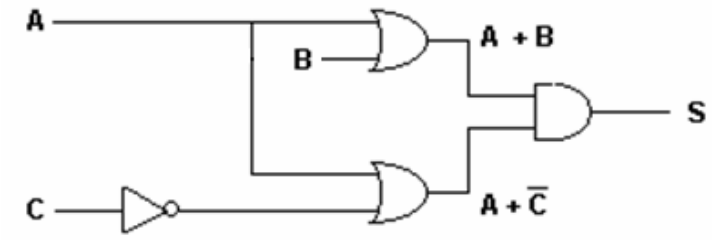
1. **Identifier les entrées et les sorties de la fonction.**
2. **Construire la table de vérité.**
3. **Identifier la fonction à partir de la table de vérité.**
4. **Simplifier la fonction.**
5. **Dessiner le schéma du circuit.**

Réalisation de circuits logiques

Exemple:

Circuit logique correspondant à l'expression algébrique:

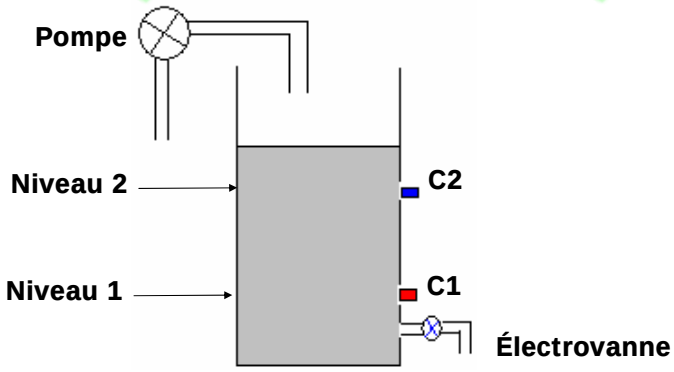
$$(A+B).(A+\overline{C})$$



Exercice 1

Donner le circuit (Exercice 3, simplification algébrique).

Exercice 2



Lorsque le niveau d'eau est inférieure au niveau 1 (Capteur C1), on déclenche la pompe pour remplir le réservoir.
Lorsque Niveau d'eau > Niveau 2, on commande l'électrovanne pour vider le réservoir.

1. Donner le circuit équivalent (sans simplification)
2. Donner le circuit simplifié.

Université Mohammed V
Faculté des Sciences
Département de Mathématiques et Informatique
SMI-4

Circuits combinatoires et Séquentiels

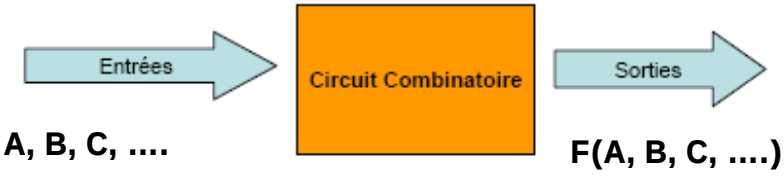
Prof. Abdelhakim El Imrani

Types de circuits logiques

- **Circuits combinatoire**
- **Circuits séquentielles**

Circuits combinatoires

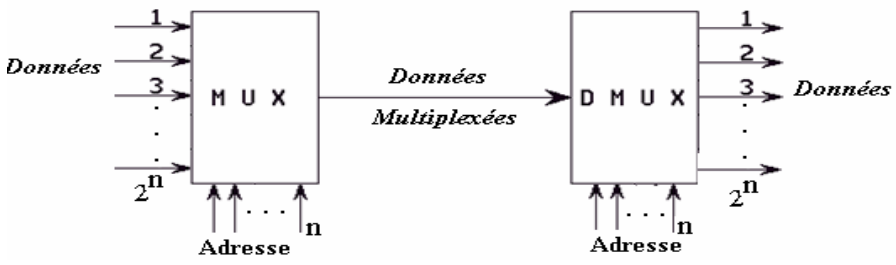
Les fonctions de sortie s'expriment selon des expressions logiques des seules variables d'entrée.



Multiplexeur - Demultiplexeur

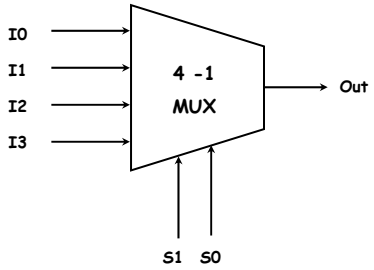
Multiplexeur
– 2^n entrées, 1 sortie
– Selon une adresse (n bits), la sortie prend la valeur de l'une des entrées

Démultiplexeur
– 1 entrée, X sorties
– Selon une adresse (n bits), une des X sorties prend la valeur de l'entrée



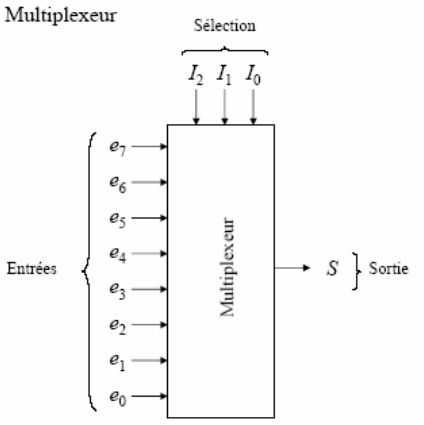
Application: **Conversion Série / Parallèle; Parallèle / Série**

Multiplexeur 4 à 1



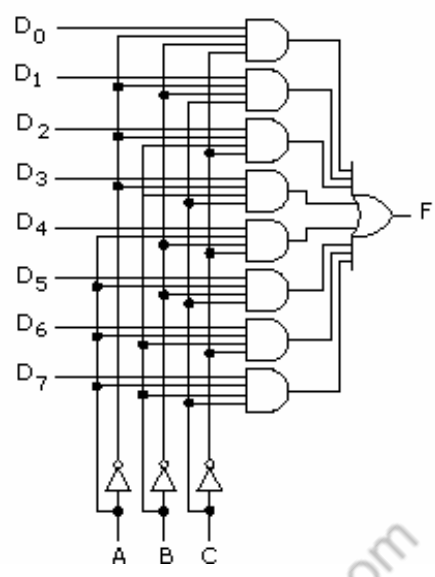
S1	S0	Out
0	0	I0
0	1	I1
1	0	I2
1	1	I3

Multiplexeur 8 à 1

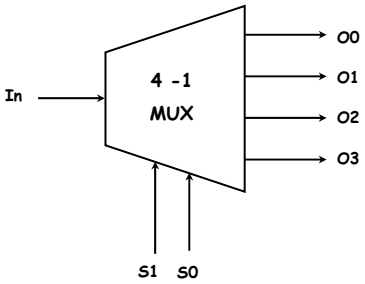


I_2	I_1	I_0	S
0	0	0	e_0
0	0	1	e_1
0	1	0	e_2
0	1	1	e_3
1	0	0	e_4
1	0	1	e_5
1	1	0	e_6
1	1	1	e_7

Exemple: Multiplexeur 8-1



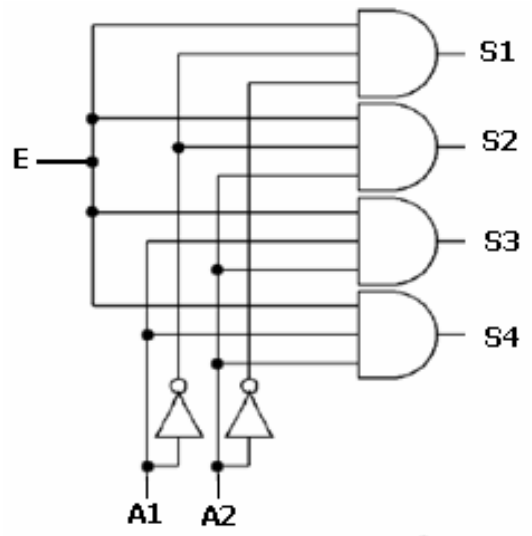
Demultiplexeur 1-4



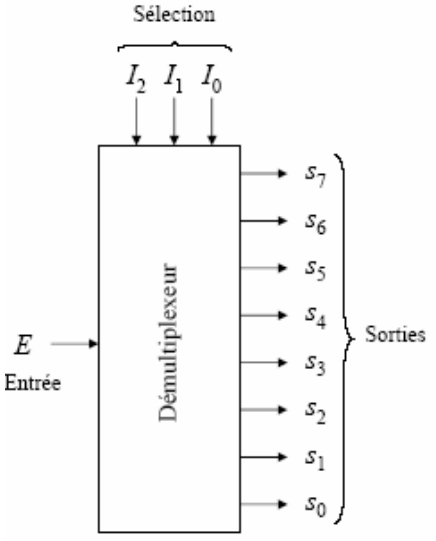
S1	S0	O0	O1	O2	O3
0	0	In	-	-	-
0	1	-	In	-	-
1	0	-	-	In	-
1	1	-	-	-	In

- : non utilisé

Exemple: Démultiplexeur 1-4



Demultiplexeur 1-8

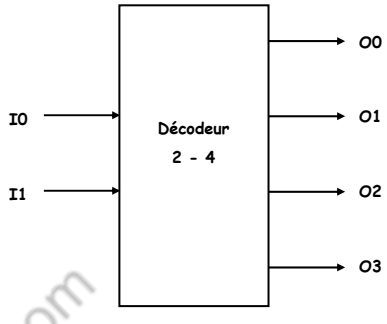


I_2	I_1	I_0	s_7	s_6	s_5	s_4	s_3	s_2	s_1	s_0
0	0	0	0	0	0	0	0	0	0	E
0	0	1	0	0	0	0	0	0	E	0
0	1	0	0	0	0	0	0	E	0	0
0	1	1	0	0	0	0	E	0	0	0
1	0	0	0	0	0	E	0	0	0	0
1	0	1	0	0	E	0	0	0	0	0
1	1	0	0	E	0	0	0	0	0	0
1	1	1	E	0	0	0	0	0	0	0

Décodeur

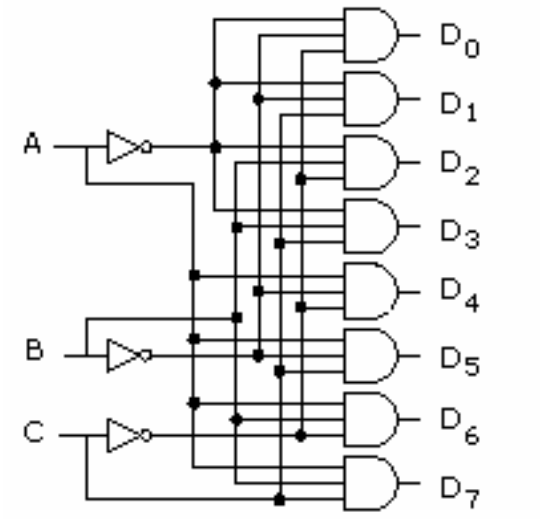
- Active une des X sorties selon un code
- Entrée sur n bits
- Nombre de sorties : 2^n
- Une seule sortie est mise à 1 selon la configuration des entrées
- **Application:** Sélection des circuits mémoire

Exemple: Décodeur 2 à 4



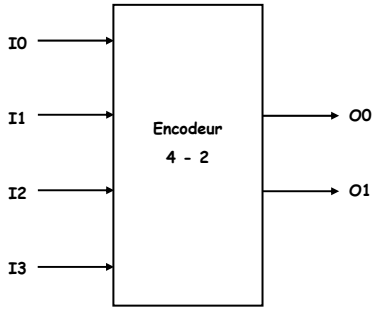
I1	I0	O0	O1	O2	O3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Exemple: Décodeur 3-8



Encodeur

- Active un code selon l'une des X entrées actives
- 2^n entrées, 1 entrée active (valeur 1), les autres sont toutes désactivées (valeur 0)
- Sortie : sur n bits



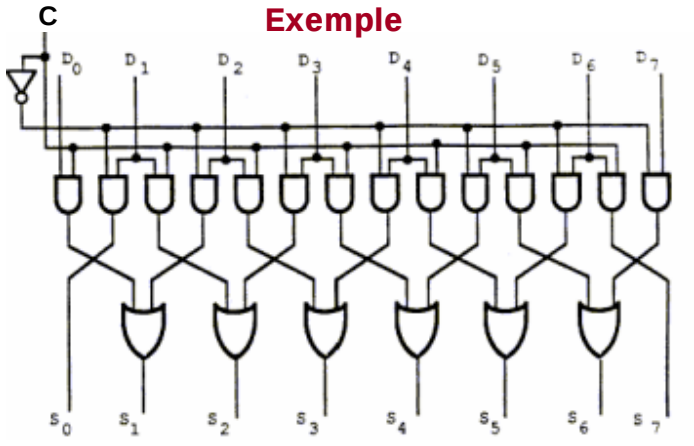
I0	I1	I2	I3	O1	O0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

Exemple: Encodeur 4-2

Circuit de décalage

Décalage de position d'un bit (à droite ou à gauche) sur les n bits

Exemple



Les lignes de sorties (S0 à S7) reflète les 8 bits d'entrée (D0 à D7) après décalage d'un bit à droite pour C=1 ou à gauche pour C=0.

Additionneur

Un additionneur est un circuit capable de faire l'addition de deux nombres de n bits. Une addition génère deux résultats.

- La somme
- La retenue

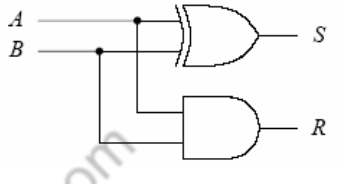
Exemple: addition de 2 bits

Somme (S) = A ⊕ B

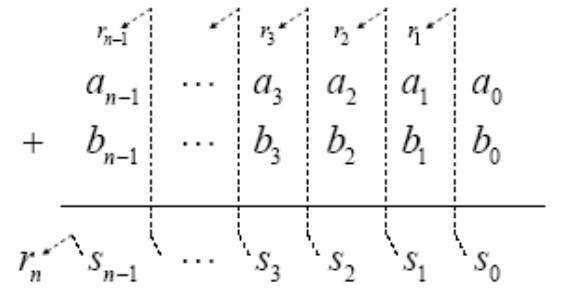
Retenue (R) = AB

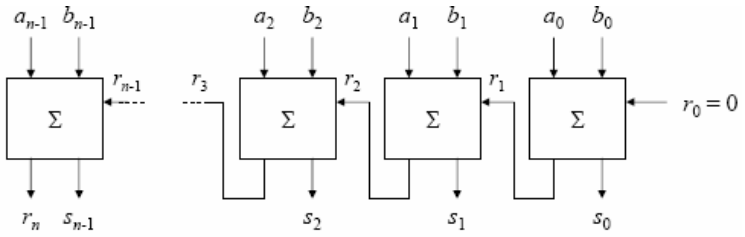
Entrée		Sortie	
A	B	R	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Demi-additionneur



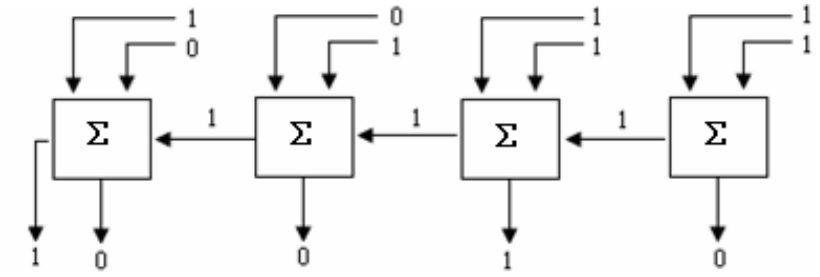
Additionneur complet





Exemple: Additionneur 4 bits

A		1	0	1	1
	+				
B		0	1	1	1
		1	0	0	1	0



Additionneur complet

r_i	a_i	b_i	s_i	r_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

a_ib_i

r_i	00	01	11	10
0	0	1	0	1
1	1	0	1	0

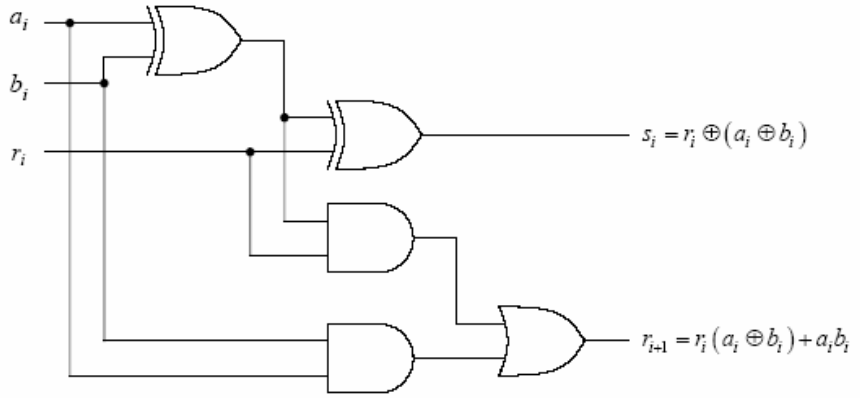
$$s_i = r_i\overline{a_i}\,\overline{b_i} + \overline{r_i}\,\overline{a_i}b_i + r_i a_i b_i + \overline{r_i}a_i\,\overline{b_i}$$
$$= r_i\left(\overline{a_i}\,\overline{b_i} + a_i b_i\right) + \overline{r_i}\left(\overline{a_i}b_i + a_i\,\overline{b_i}\right)$$
$$= r_i\overline{(a_i \oplus b_i)} + \overline{r_i}\,(a_i \oplus b_i) = r_i \oplus (a_i \oplus b_i)$$

a_ib_i

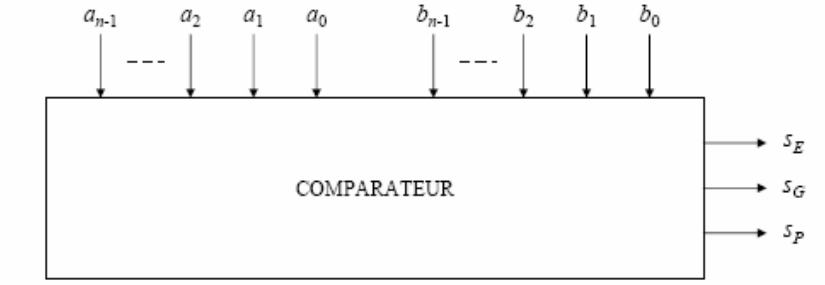
r_i	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$r_{i+1} = r_i a_i + r_i b_i + a_i b_i \quad \text{ou} \quad r_i\,\overline{a_i}b_i + r_i a_i\,\overline{b_i} + a_i b_i$$
$$= r_i\left(\overline{a_i}b_i + a_i\,\overline{b_i}\right) + a_i b_i$$
$$= r_i\,(a_i \oplus b_i) + a_i b_i$$

Additionneur complet

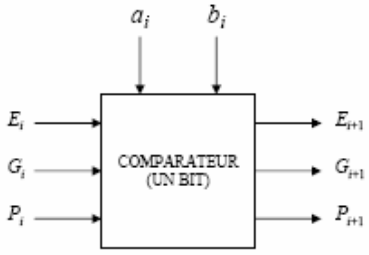


Comparateur



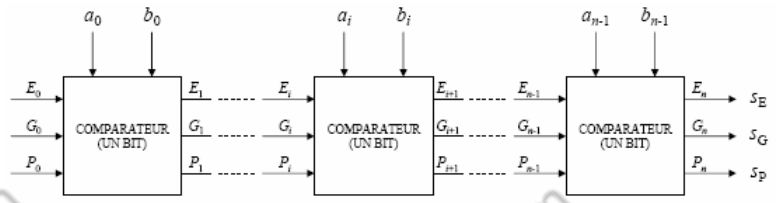
$$s_E s_G s_P = \begin{cases} 100 \text{ si } [a_{n-1}a_{n-2}\cdots a_2 a_1 a_0] = [b_{n-1}b_{n-2}\cdots b_2 b_1 b_0] \\ 010 \text{ si } [a_{n-1}a_{n-2}\cdots a_2 a_1 a_0] > [b_{n-1}b_{n-2}\cdots b_2 b_1 b_0] \\ 001 \text{ si } [a_{n-1}a_{n-2}\cdots a_2 a_1 a_0] < [b_{n-1}b_{n-2}\cdots b_2 b_1 b_0] \end{cases}$$

Réalisation avec des comparateurs 1 bit



a	b	E	G	P
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

Comparateur n bits

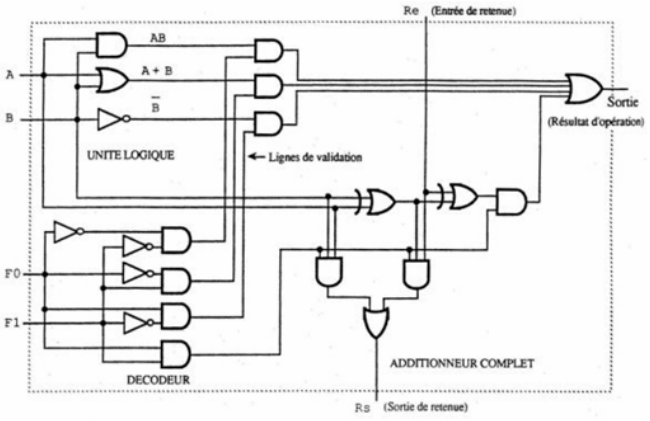


Note : $E_0 = 1$ et $G_0 = P_0 = 0$

Unité arithmétique et logique (UAL)

Effectue les opérations de base (arithmétiques et logiques). un code opération détermine la partie du circuit qui va effectuer les opérations.

Exemple: UAL (1 bit)

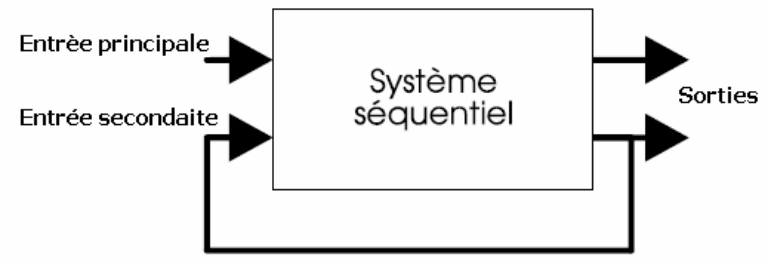


Selon code (r0,r1), le circuit calcule AB, A + B, B ou l'addition de A et $\overline{B}$.

Circuits séquentiels

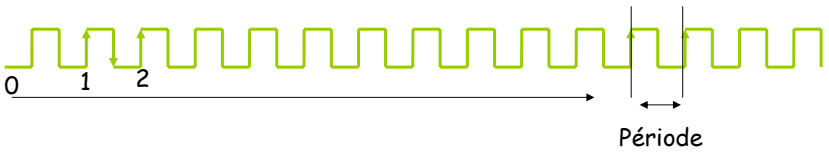
Les fonctions de sortie dépendent non seulement de l'état des variables d'entrée mais également de l'état antérieur de certaines variables de sortie (propriétés de mémorisation)

Table de vérité : on trouve en plus des entrées, la valeur de sortie à l'état précédent



Horloge (Clock)

- Les bascules sont généralement commandées par horloge
- Horloge : composant passant indéfiniment et régulièrement d'un niveau haut à un niveau bas (succession de 1 et de 0), chaque transition s'appelle un *top*.



Fréquence = nombre de changement par seconde en hertz (Hz)
Fréquence = 1/période
Une horloge de 1 hertz a une période de 1 seconde
……1 megahertz……1 millisec
……1 gigaHz……1 nanoSec

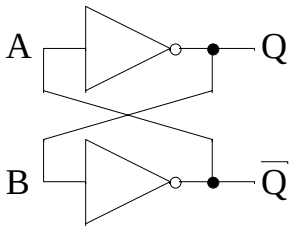
Circuits synchrone et asynchrone

- **Circuit synchrone**
  - Tous les éléments/composants du circuit devant être synchronisés le sont avec le même signal d'horloge
- **Circuit asynchrone**
  - Tous les éléments/composants du circuit devant être synchronisés ne le sont pas avec le même signal d'horloge

Les bascules

- Les circuits séquentiels de base sont les bascules
- Particularité : deux états stables = conservation de l'état de leur sortie même si la combinaison des signaux d'entrée l'ayant provoquée disparaît.
- Une bascule (flip-flop) a pour rôle de mémoriser une information élémentaire (mémoire à 1 bit).
- Une bascule possède deux sorties complémentaires Q et $\overline{Q}$.
- La mémorisation fait appel à un système de blocage (latch), dont le principe est représenté de la façon suivante.
- **Application:** Registres, Compteurs, etc.

Les bascules



$$\begin{cases}(Q=1)\Rightarrow(B=1)\Rightarrow(\overline{Q}=0)\Rightarrow(A=0)\Rightarrow(Q=1)\\(Q=0)\Rightarrow(B=0)\Rightarrow(\overline{Q}=1)\Rightarrow(A=1)\Rightarrow(Q=0)\end{cases}$$

Une bascule ne peut donc être que dans deux états:

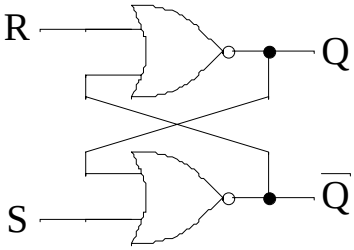
"1" : $(Q=1,\overline{Q}=0)$ " 0" : $(Q=0,\overline{Q}=1)$

Les interconnexions interdisent les deux autres combinaisons :

$Q=\overline{Q}=0$ ou $Q=\overline{Q}=1$.

Les bascules RS

- Les bascules les plus fréquemment utilisées sont réalisées avec deux portes NOR ou NAND.



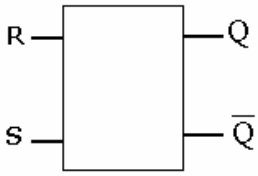


Schéma simplifié

(R) Set: Mise à 1

(S) Reset: Mise à Zéro

Les bascules RS

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

Sortie inchangée

Set: Mise à 1

Reset: remise à zéro

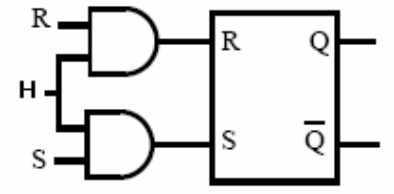
État interdit

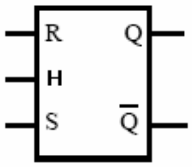
$$Q = \overline{R + \overline{Q}}$$

$$\overline{Q} = \overline{S + Q}$$

Bascule RS synchrone ou RST

Commandé par un signal horloge

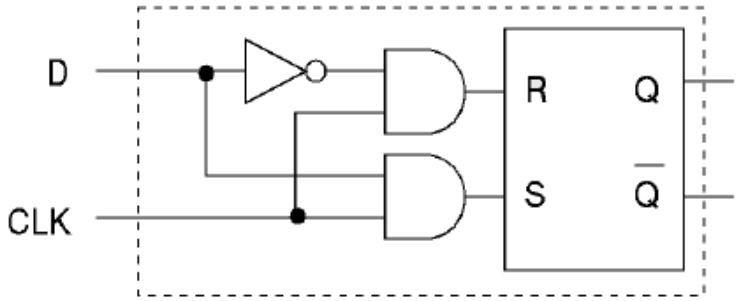




H = 1 → lecture
H = 0 → mémorisation

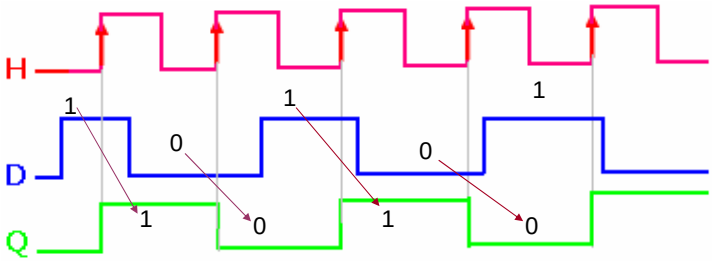
Bascule D

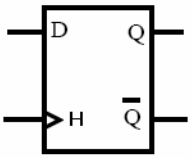
Pour éliminer l'état interdit S=R=1 → $Q=\overline{Q}$

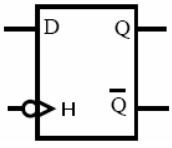


Bascule sur front d'horloge

Principe: saisir l'information lors du changement d'état de l'horloge







Bascule D **Front montant** **Front descendant**

Bascule D

D	H	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$Q_{n+1} = DH + Q_n\overline{H}$$

Versions condensées

(H=1)

D	Q_{n+1}
0	0
1	1

H	Q_{n+1}
0	Q_n
1	D

Bascule JK asynchrone

- JK = variante de RS
- Semblable à RS mais ajoute le cas R=S=1
- Si J = K = 1 alors $Q_{n+1} = \overline{Q_n}$

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

Q_n \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

J	K	Q_{n+1}
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

$$Q_{n+1} = J\overline{Q}_n + \overline{K}Q_n$$

Utilisation des bascules

Les bascules sont utilisées pour créer des circuits:

- **Compteurs**
- **Registres**
  - Mémorisation d'un mot mémoire, décalage vers la droite/gauche du mot ...
- **Mémoires** (SRAM)

Exemple: compteur cyclique sur 3 bits
- Valeur en décimal sur 3 bits
- Incrémentation de +1 à chaque période d'horloge
- Repasse à 0 après 7

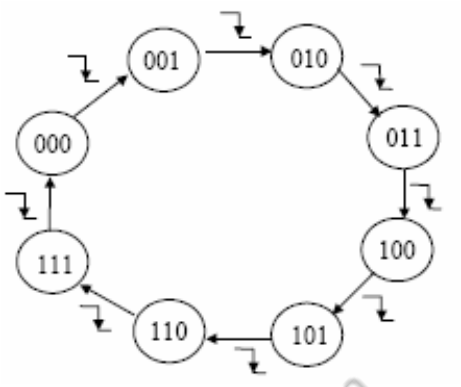
Les compteurs

Ensemble de n bascules interconnectées:

- Peuvent mémoriser des mots de n bits.
- Au rythme d'une horloge ils peuvent décrire une séquence déterminée c'est-à-dire une suite d'états binaires.
- De nombreuses applications industrielles:
  - Comptage du nombre de révolutions d'un moteur,
  - Division de fréquences,
  - Conversions de code, Conversion A/N et N/A, etc.
- Compteur binaire est dit modulo N lorsqu'il peut compter jusqu'à N-1, la Nième impulsion remet le compteur à zéro. $N=2^n$, où n représente le nombre d'étages.
  - Compteurs asynchrones
  - Compteurs synchrones

Exemple: Compteur modulo 8

Un compteur modulo 8 démarre à 0 et compte dans l'ordre binaire naturel de 0 à 7.

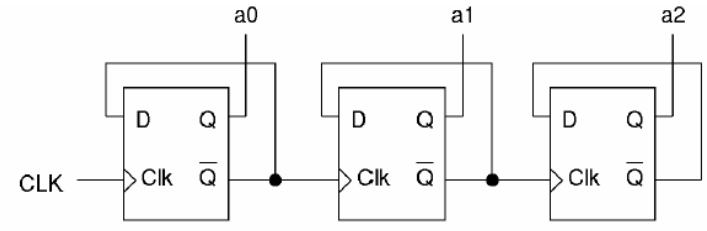


Exemple: Compteur modulo 8

Utilisation de 3 bascules D:

- Principe
  - Chaque bascule prend en entrée D un signal d'horloge
  - Fournit en sortie un signal d'horloge de fréquence divisée par 2

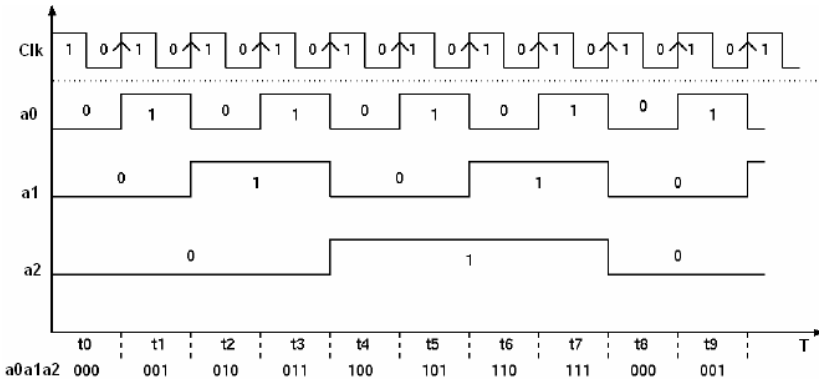
- En mettant en série les 3 bascules
  - 3 signaux d'horloge à 3 fréquences différentes
  - Représente les combinaisons de bits pour les valeurs de 0 à 7



Compteur asynchrone

Chronogramme du compteur 3 bits

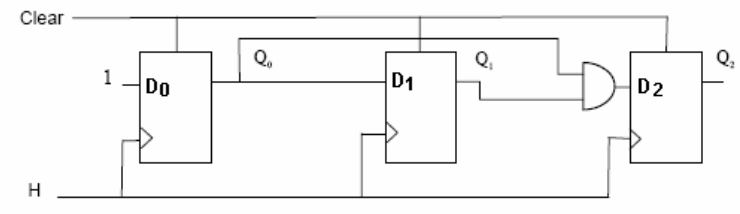
- Version idéale, ne prend pas en compte les temps de propagation à travers les bascules



Compteur synchrone

- Les bascules reçoivent en parallèle le même signal d'horloge.

D0 = 1
D1 = Q0
D2 = Q0.Q1
Dn = Q0.Q1....Qn-1



Exemple: Compteur modulo 8

Registres

- Registres : Mémoires du microprocesseur de X bits (8, 16, 32, etc.)
- Composant localisé dans un processeur pour stocker des informations lors de l'exécution d'un programme par ce processeur (instruction, donnée, état du processeur, etc.)

Exemple: Registre 4 bits

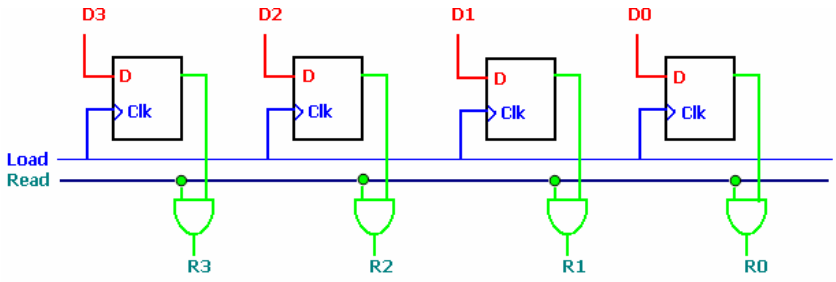
- 4 bascules D stockent les 4 bits
- 4 bits en entrée pour écrire le mot
- 4 bits en sortie pour récupérer la valeur du mot
- Une entrée L (pour « load ») précise si on conserve la valeur du mot stocké (valeur 0) ou écrit le mot avec les 4 bits en entrée (valeur 1).

Exemple: Registre 4 bits

Supposons que l'on ait 4 bits D0, D1, D2, D3 à transférer vers les sorties d'un registre R0, R1, R2, R3.

Le transfert est fait en deux étapes:

1. Les valeurs de Di , i = 0, ..., 3 sont transférées vers les sorties Q des 4 bascules D
2. Une impulsion sur la ligne Read permet le transfert vers la sortie des portes ET

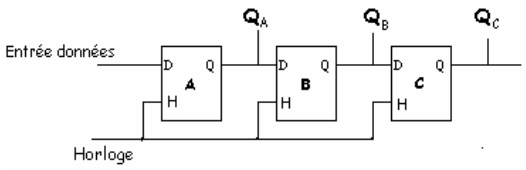


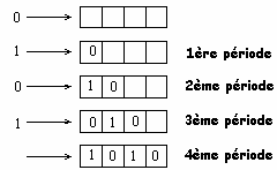
Registre Parallèle/Parallèle 4 bits

Registre à décalage

Bascules interconnectées de façon à ce que l'état logique de la bascule de rang i puisse être transmis à la bascule de rang i+1.

Exemple:
Les bascules sont commandées par le même signal horloge H. Sachant que dans une bascule D, l'état suivant n+1 de la sortie Q est égale à l'état présent D_n de l'entrée D, on a donc:
$A_{n+1} = D_{An} =$ information à l'entrée
$B_{n+1} = D_{Bn} = A_n$
$C_{n+1} = D_{Cn} = B_n$





Chargement de la valeur 1010 dans un registre 4 bits