

UAS 1GI

Chapitre 2: Logique Combinatoire et la logique Séquentielle (Parties 1 et 2)

Elaboré par : Moncef DRIRA

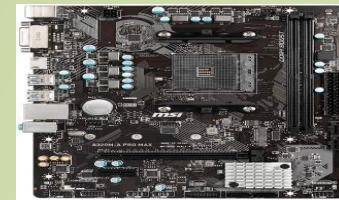
Février 2023

- ❑ **Introduction**
- ❑ **Algèbre de Boole**
- ❑ **Simplification des fonctions logiques**
- ❑ **Réalisation Technologique**
- ❑ **Les circuits combinatoires**
- ❑ **Les circuits séquentiels**

Introduction

3

Les machines numériques (ordinateurs, tablettes, téléphones,...) sont constituées d'un ensemble de circuits électroniques.



A

B

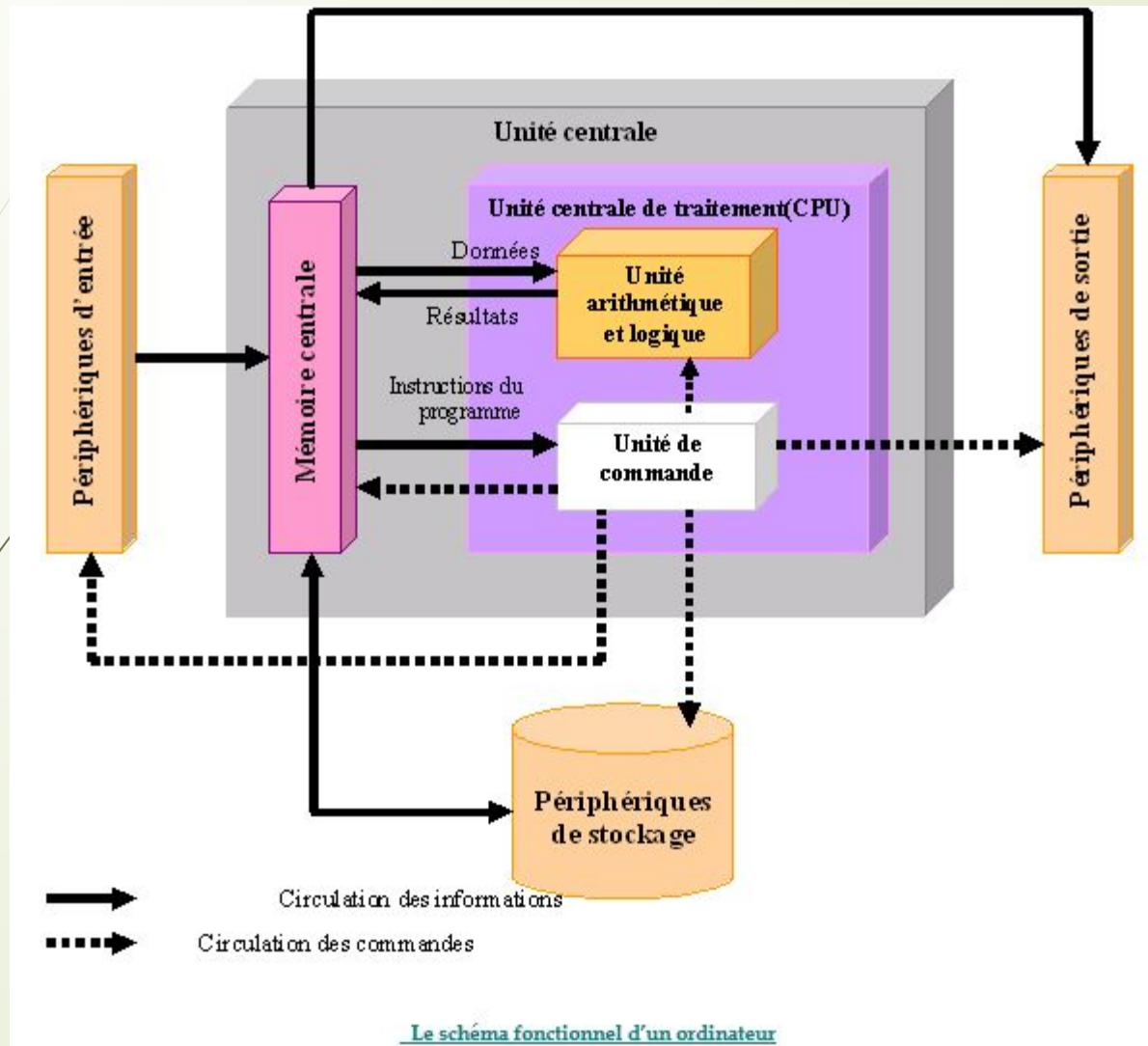
Circuit

$F(A,B)$

Chaque circuit fournit une **fonction logique** bien déterminée; opérations logiques ou arithmétiques (addition, soustraction, comparaison ,....).

Structure d'un ordinateur

4

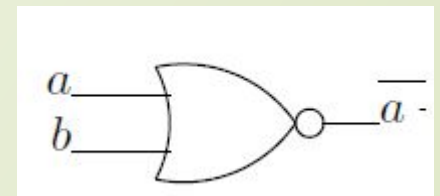
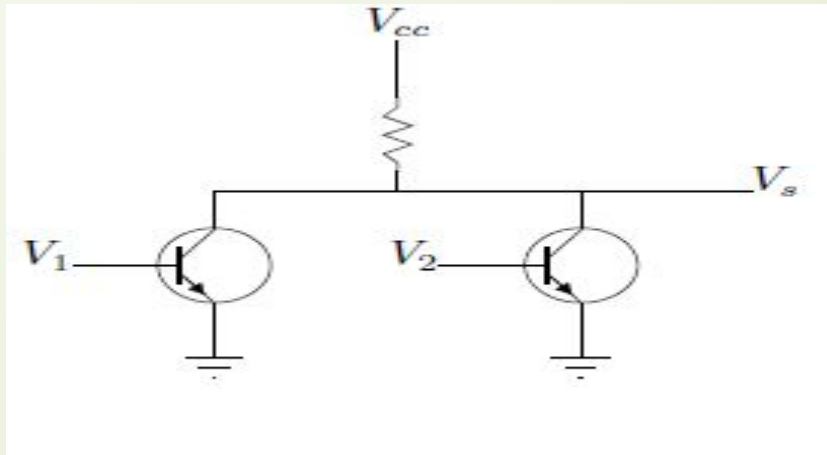


5

Une fonction logique de base est réalisée à l'aide des portes logiques qui permettent d'effectuer des opérations élémentaires.



Ces portes logiques sont aujourd'hui réalisées à l'aide de transistors.



Pour concevoir et réaliser ce circuit on doit avoir **un modèle mathématique** de la fonction réalisée par ce circuit .

6



Ce modèle doit prendre en considération **le système binaire**.



Le modèle mathématique utilisé est celui de **Boole**.

1854 : Georges Boole propose une **algèbre**

Propositions **vraies** ou **fausses**  Algèbre de **Boole**
et **opérateurs** possibles

Étude des **systèmes binaires** :

Possédant **deux états s'excluant mutuellement**

C'est le cas des systèmes numériques

(des sous ensembles : **les circuits logiques**).

Définitions:

• États logiques:

0 et 1, Vrai et Faux, H et L

• Variable logique:

symbole peut prendre comme valeur des états logiques

Exemples: A, B, c,

• Fonction logique:

Expression de variables et d'opérateurs

Une fonction logique ne peut prendre que deux valeurs 0 et 1.

Fonction logique à n variables $f(a,b,c,d,...,n)$:

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

La manière de représenter une fonction logique: **table de vérité**.

L'opérateur OU (OR):

9 Le **OU (OR)** est un **opérateur binaire** (deux variables):

- Sert à réaliser la **somme logique** entre deux variables logiques.
- Le OU fait la **disjonction** entre deux variables.
- Le OU est défini par **$F(A,B) = A + B$**
(Ne pas confondre avec la somme arithmétique)

table de vérité

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

$S = A + B$
S vrai si A OU B est vrai

L'opérateur ET (AND):

10

Le ET (AND) est un opérateur **binaire** (deux variables):

- Il sert à réaliser le **produit logique** entre deux variables booléennes.
- Le ET fait la **conjonction** entre deux variables.
- Le ET est défini par: $F(A,B) = A.B$

table de vérité

A	B	A . B
0	0	0
0	1	0
1	0	0
1	1	1

$$P=A.B$$

S vrai si **A** et **B** sont vraies

L'opérateur NON (NOT):

11

Le **NON (NOT)** est un opérateur **unaire** (une seule variable):
sert à **inverser** la valeur d'une variable booléenne .

$$F(A) = \text{Non } A = \overline{A}$$

(lire : **A barre**)

table de vérité

A	\overline{A}
0	1
1	0

$$I = \overline{A}$$

I vrai si **A est faux**

Les opérateurs NAND et NOR:

12

Les opérateurs NAND (Not AND) et NOR (Not OR) sont des opérateurs binaires:

NAND: $S = a \uparrow b = \overline{a.b}$

S est vrai si a OU b est faux.

table de vérité

b \ a	0	1
0	1	1
1	1	0

NOR: $S = a \downarrow b = \overline{a+b}$

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

table de vérité

b \ a	0	1
0	1	0
1	0	0

NAND et NOR ne sont pas associatives

L'opérateur XOR:

13

Le **XOR** (le **OU exclusif**) est un opérateur **binaire** (deux variables):

sert à **inverser** la valeur d'une variable booléenne .

$$S = a \oplus b = a \cdot b + a \cdot b$$

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

table de vérité

	0	1
1	0	1
0	1	0

XOR est associative:

$$S = a \oplus b \oplus c \oplus d \dots \oplus n$$

Simplification des fonctions logiques

14

La simplification des fonctions logiques vise:

- * réduire le nombre de termes dans une fonction
- * réduire le nombre de variables dans un terme

 **le nombre de portes logiques utilisées est réduit**
le coût du circuit est réduit

Plusieurs méthodes existent pour la **simplification** :

1) Les méthodes algébriques

2) Les méthodes graphiques :

(ex : tableaux de **Karnaugh**)

Les méthodes algébriques:

15

* Commutativité:

$$a + b = b + a$$

$$a \cdot b = b \cdot a$$

* Associativité:

$$a + (b + c) = (a + b) + c$$

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c$$

* Distributivité:

$$a \cdot (b + c) = a \cdot b + a \cdot c$$

$$a + (b \cdot c) = (a + b) \cdot (a + c)$$

* Idempotence:

$$a + a = a$$

$$a \cdot a = a$$

* Absorption:

$$a + a \cdot b = a$$

$$a \cdot (a + b) = a$$

* Involution:

$$\overline{\overline{a}} = a$$

*** Élément neutre:**

$$a+0 = a$$

$$a.1 = a$$

•Élément absorbant:

$$a+1 = 1$$

$$a.0 = 0$$

•Inverse:

$$a + \overline{a} = 1$$

$$a . a = 0$$

*** Théorème de "De Morgan"**

$$\overline{a + b} = \overline{a} . \overline{b}$$

$$\overline{a . b} = \overline{a} + \overline{b}$$

Théorème du Consensus:

$$a . x + b . \overline{x} + a . b = a . x + b . \overline{x}$$

$$(a + x)(\overline{b} + x)(a + b) = (a + x)(\overline{b} + x)$$

Description du tableau de Karnaugh:

17

- La méthode consiste à mettre en évidence par une méthode **graphique (un tableau)** tous les termes qui sont **adjacents** (qui ne diffèrent que par **l'état d'une seule variable**).
- Un tableau de Karnaugh= table de vérité de 2^n **cases avec un changement unique entre 2 cases voisines** d'où des codes cycliques (Gray ou binaire réfléchi).
- La méthode peut s'appliquer aux fonctions logiques de **2,3,4,5 et 6 variables**.
- Les tableaux de Karnaugh comportent 2^n cases(n: est le nombre de variables).

- **Règles de regroupement:**

- groupe de 2^n cases: **1, 2, 4 ou 8**

18

- en ligne, colonne, rectangle, carré, **mais pas en diagonale**

- tous des 1, mais pas des 0, au moins une fois dans les groupements

- **Règles de minimisation (simplification) de la fonction:**

- Rechercher les groupements en commençant par les cases qui n'ont qu'une seule façon de se grouper;

- Rechercher les groupements les plus grands;

- Les groupements doivent contenir au moins un 1 non utilisé par les autres groupements;

- L'expression logique finale est la réunion (**la somme**) des groupements après simplification et élimination des variables qui changent d'état.

- Exemple:**

Transformer une table de vérité en un tableau de Karnaugh:.

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



Code Gray ou binaire réfléchi
= 1 seul changement entre deux codes successifs

bc	00	01	11	10
a				
0	0	1	0	0
1	1	0	1	0
	f(a,b,c)			

- Tableau à 3 variables:

ab	00	01	11	10
c 0	0	1	0	0
1	1	1	1	1

$$f(a,b,c) = \overline{a} \cdot b + c$$

- Tableau à 4 variables:

ab	00	01	11	10
cd				
00	0	0	0	1
01	1	1	1	1
11	0	0	0	0
11	0	1	0	0

$$f(a,b,c) = \overline{c} \cdot d + \overline{a} \cdot b \cdot \overline{c} + \overline{a} \cdot b \cdot d$$

• Tableau à 5 variables:

ab	00	01	11	10
cd				
00	1	1	1	1
01	0	0	1	0
11	0	0	1	0
10	0	0	0	0

U=0

ab	00	01	11	10
cd				
00	1	1	1	1
01	0	0	0	1
11	1	1	0	0
10	0	0	0	0

U=1

$$f(a,b,c) = \underline{c} . \underline{d} + \underline{u} . \underline{a} . \underline{b} . \underline{d} + u . a . \overline{b} . \overline{c} . d + \underline{u} . \underline{a} . \underline{c} . d$$

Réalisation Technologique

23


- * Un ordinateur travaille en base 2.

- *Électroniquement:

le zéro (0) correspond(ait) à une tension de 0 à 0.8 V

le un (1) à une tension comprise entre 2.8 et 5 V.

- * On a vu que toute **fonction binaire** peut être représentée par une **expression booléenne**.

 tout **circuit électrique ou électronique à deux valeurs de tension** peut être représentée par une **expression booléenne**.

- * Plutôt que de représenter les circuits par des expressions booléennes, on préfère les représenter en utilisant des **symboles logiques** dits **portes logiques**

- * Ces portes sont présentées en :

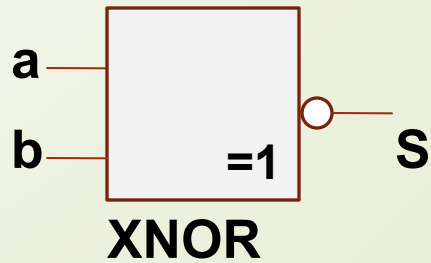
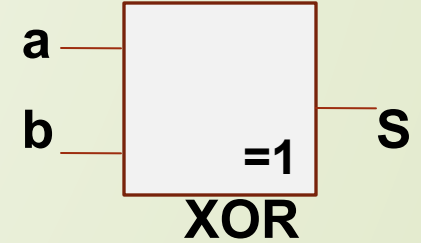
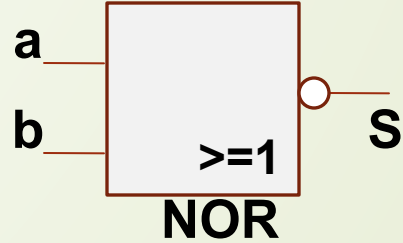
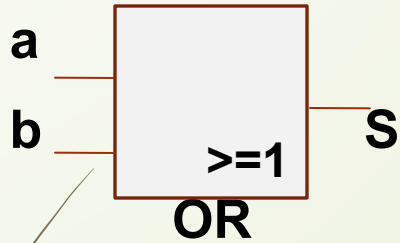
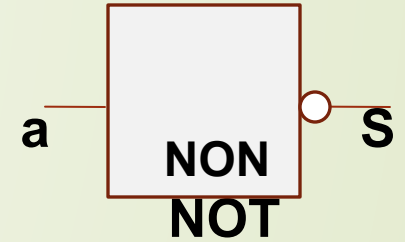
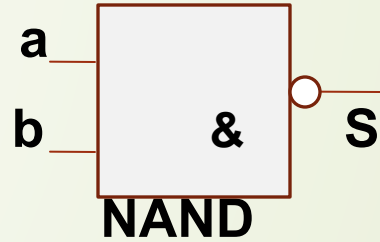
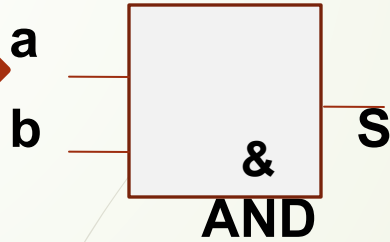
Normes **françaises**

Normes **américaines**

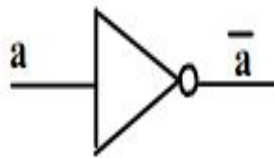
Représentation graphique : Norme française

Portes logiques

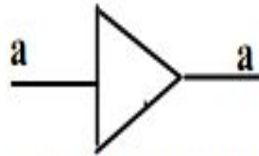
24



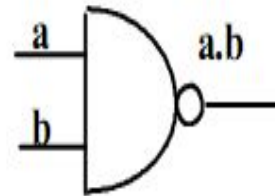
Représentation graphique : Norme américaine



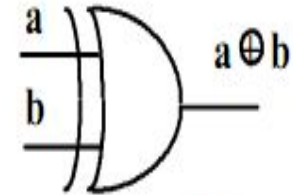
inverseur



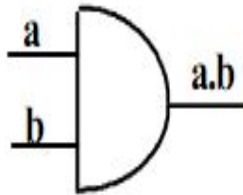
tampon (buffer)



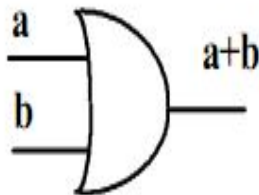
porte NON ET
(NAND)



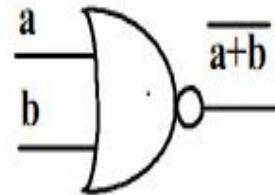
porte ~~ET~~ XOR



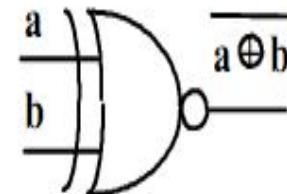
porte ET



porte OU



porte NON OU (NOR)



porte ~~ET~~ XNOR

* Concrètement chaque porte logique est **réalisée électroniquement par un ou deux transistors**.

* Plusieurs portes logiques forment **un *circuit logique***.

* La réalisation d'un circuit passe d'abord par la recherche des **expressions booléennes**, ensuite par leur simplification basée sur l'algèbre de Boole(avec les fonctions logiques ou graphiques)

On distingue deux types de circuits logiques :

– les circuits combinatoires qui ne font que combiner selon une table de vérité les variables d'entrée (**la valeur des sorties dépend de la valeur des entrées**)

– les circuits séquentiels construits à partir de circuits combinatoires et qui se caractérisent **par une capacité de mémorisation: la valeur des sorties à l'instant t dépend de la valeur des entrées $e(t)$ et de la valeur des sorties à l'instant $t-1$**

Objectifs pédagogiques:

27

- Apprendre la **structure** de quelques circuits combinatoires souvent utilisés (**demi additionneur, additionneur complet, comparateur, multiplexeur, démultiplexeur, codeur et décodeur,**).

- Apprendre **comment** utiliser des **circuits combinatoires** pour concevoir d'autres circuits plus complexes.

Circuit séquentiel

n variables d'entrées

m variables de sortie

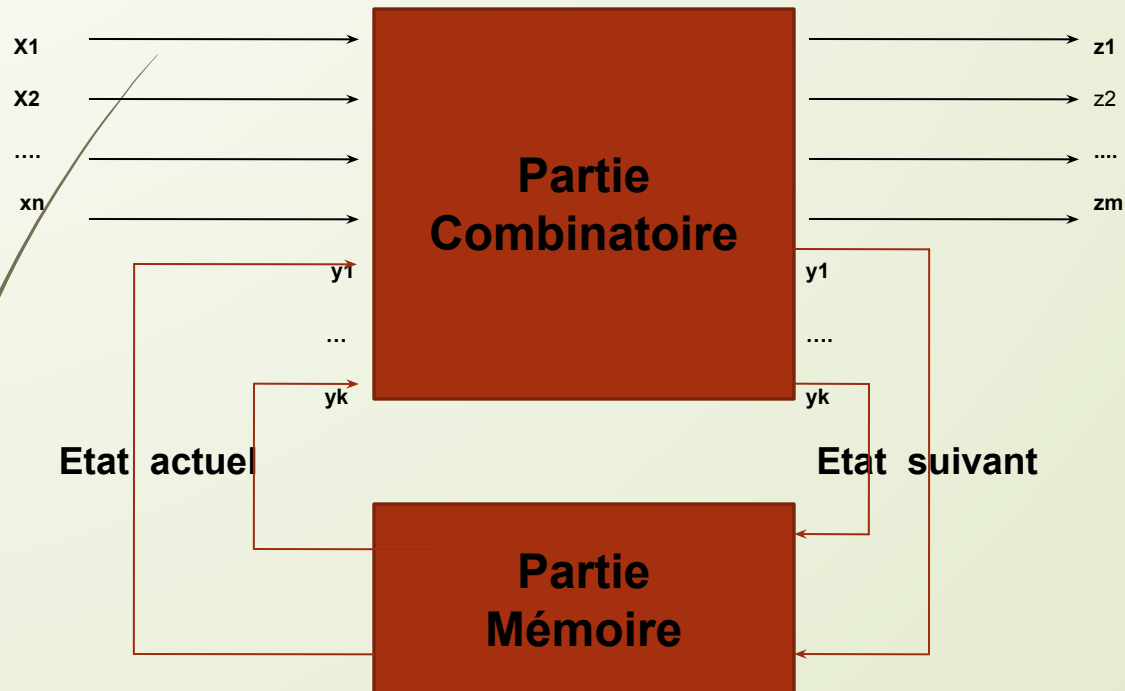


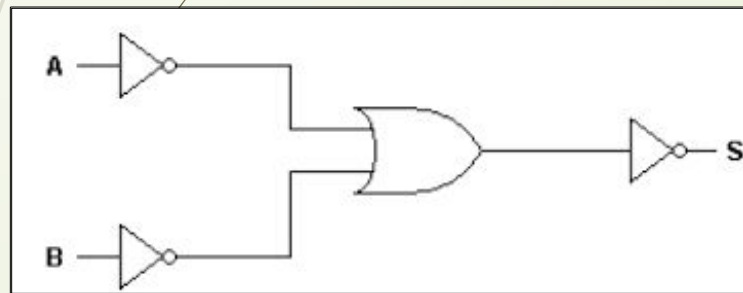
Schéma d'un circuit logique (Logigramme)

29

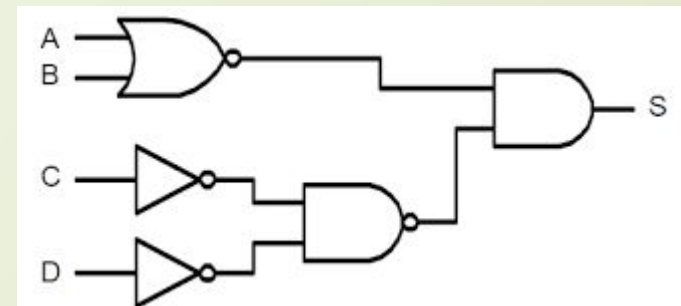
Un logigramme est la **traduction** de la fonction logique en un **schéma électronique**.

Le principe consiste à **remplacer** chaque **opérateur logique** par la **porte logique** qui lui correspond.

Exemples: $F(A,B) = \overline{\overline{A}} + \overline{\overline{B}}$



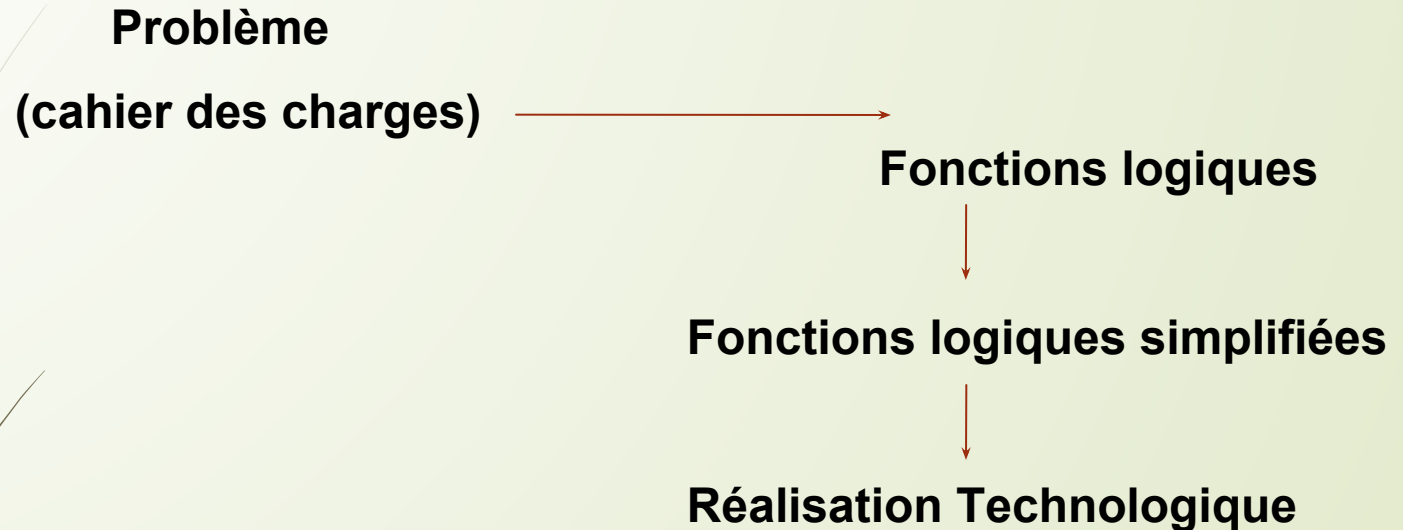
$F(A, B, C, D) = \overline{\overline{A} + \overline{\overline{B}}} \cdot \overline{\overline{C} \cdot \overline{\overline{D}}}$



Les circuits combinatoires

30

Moyens physiques de réalisation des fonctions logiques:

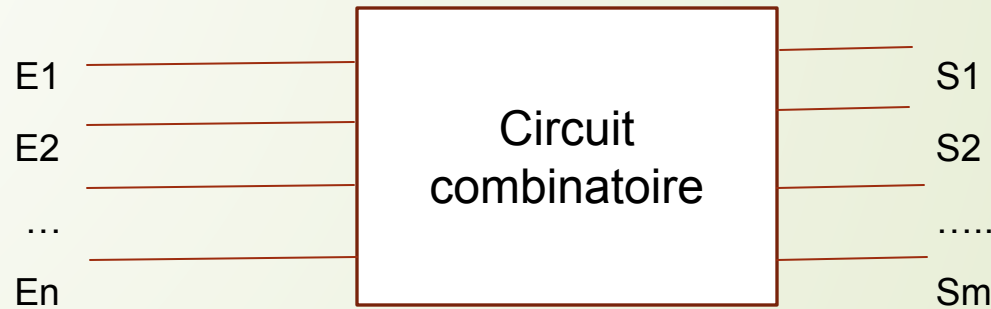


Définition:

Un circuit **combinatoire** est un **circuit numérique** dont les **sorties** dépendent uniquement des **entrées**.

$$S_i = F(E_i)$$

exemple: $S_i = F(E_1, E_2, \dots, E_n)$ (avec $i=1..n$)



C'est possible d'utiliser des circuits combinatoires pour réaliser d'autres circuits **plus complexes**

Exemples de circuits combinatoires:

- Additionneur, comparateurs
- Inverseurs
- Multiplexeur / démultiplexeur
- Codeurs / Décodeurs
- Transcodeurs
- Unité arithmétique et logique UAL

1- Additionneur

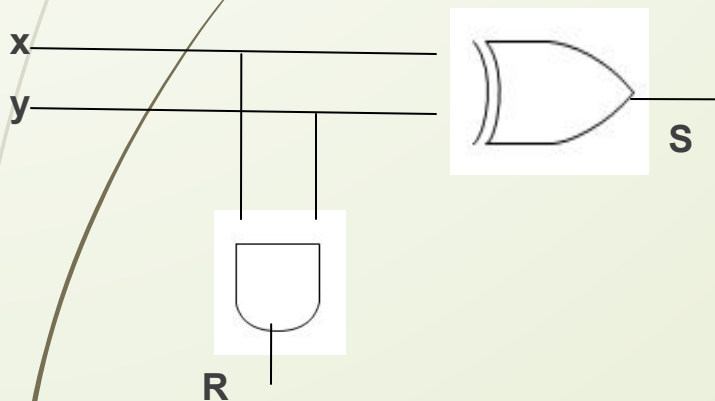
33

– Addition de 2 bits x et y (demi-additionneur HA) :

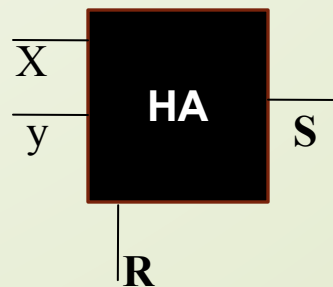
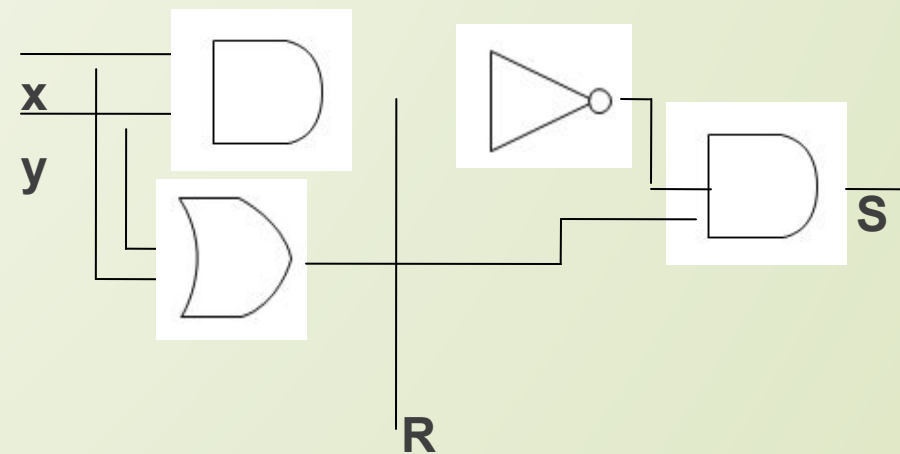
x	y	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

table de vérité □ Somme $S = x \oplus y$
 $= \bar{x}.y + x.\bar{y}$
 $= (x + y).(x + \bar{y})$
 $= (x + y).(\overline{x.y})$
 retenue $R = x.y$

$$S = x \oplus y, R = x.y$$



$$S = (x + y).(\overline{x.y}), R = x.y$$



– Additionneur:

34

Un additionneur se réalise à l'aide d'additionneur 3 bits mis bout à bout.

Dans une addition de deux nombres entiers, on additionne **bit à bit** en considérant la **retenue du rang précédent** .

➡ Somme de 3 bits = additionneur complet.

Formules :

La somme **S** vaut un si entre les bits x , y et la retenue d'entrée re **le nombre de bits à un est impair** ;

la retenue de sortie **R_s** vaut un si les bits x et y valent un, ou si l'un ou l'autre de ces bits vaut un alors que la retenue d'entrée re vaut un.

-Table de vérité:

35

x	y	Re	S	Rs
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

$$Rs = \bar{x} \cdot y \cdot re + x \cdot \bar{y} \cdot re + x \cdot y \cdot \bar{re} + x \cdot y \cdot re$$

$$= x \cdot re \cdot (\bar{y} + y) + y \cdot (\bar{x} \cdot re + x \cdot \bar{re})$$

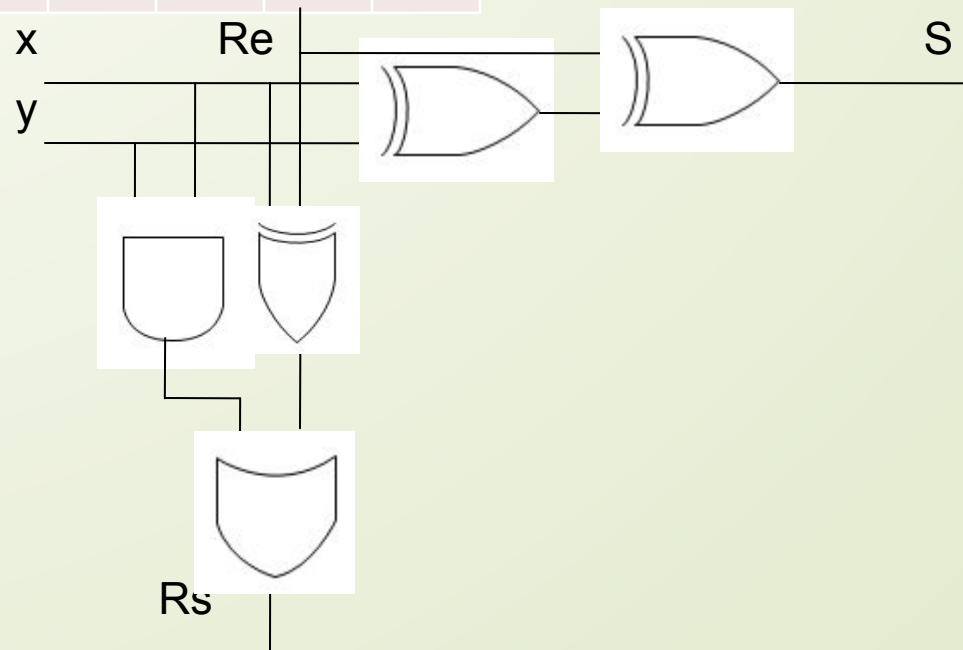
$$= x \cdot re + y \cdot (x \oplus re)$$

$$S = \bar{x} \cdot \bar{y} \cdot re + \bar{x} \cdot y \cdot \bar{re} + x \cdot \bar{y} \cdot \bar{re} + x \cdot y \cdot re$$

$$= \bar{x} \cdot (\bar{y} \cdot re + y \cdot \bar{re}) + x \cdot (\bar{y} \cdot \bar{re} + y \cdot re)$$

$$= \bar{x} \cdot (re \oplus y) + x \cdot (\overline{y \oplus re}) \quad (\text{XOR est associative})$$

$$= x \oplus y \oplus re$$



Additionneur sur 4 bits:

Un additionneur sur 4 bits est un circuit qui permet de faire l'addition de deux nombres A et B de **4 bits chacun**

$-A(a_3 a_2 a_1 a_0)$

$-B(b_3 b_2 b_1 b_0)$



En plus il prend en compte de la retenue entrante ($r_0 = 0$)

En sortie on va avoir le résultat sur **4 bits** ainsi que la retenue (**5 bits en sortie**)



Au total le circuit possède 9 entrées et 5 sorties.

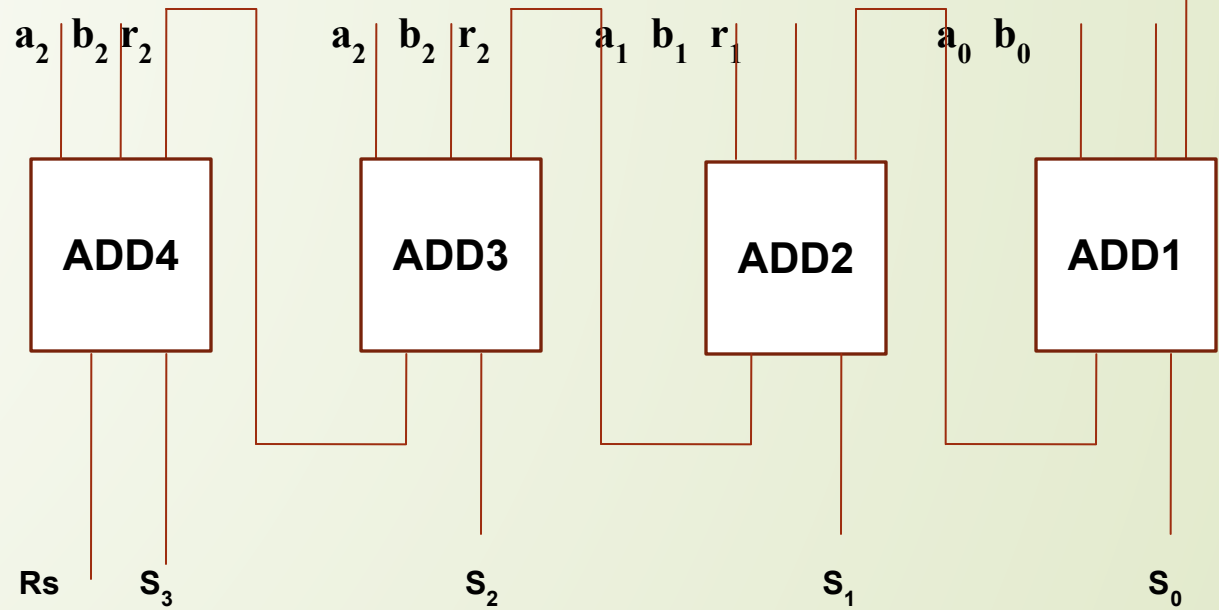
Avec 9 entrées on a $2^9 = 512$ **combinaisons donc TV complexe**

Rechercher une solution **plus facile** et efficace pour concevoir le circuit

- Les mots $A(a_3 a_2 a_1 a_0)$ et $B(b_3 b_2 b_1 b_0)$

$r_0 = 0$

37



2- Comparateur:

– Comparateur sur 1 bit:

C'est un circuit combinatoire qui permet de comparer entre **deux nombres binaires** A et B.

• Il possède 2 entrées :

– A : sur n bits

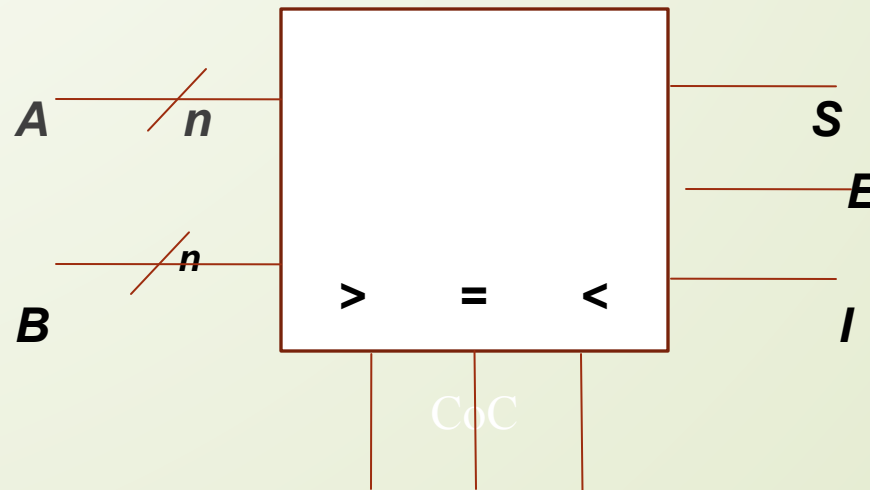
– B : sur n bits

Il possède 3 sorties

E : égalité ($A=B$)

I : inférieur ($A < B$)

S : supérieur ($A > B$)



Entrées de cascading

(pour une comparaison à n autres bits)

Comparateur sur un bit:

– Il possède deux entrées

A sur un bit et B sur un bit

39

x	y		S	E	I
0	0		0	1	0
0	1		0	0	1
1	0		1	0	0
1	1		0	1	0

\oplus

table de vérité □

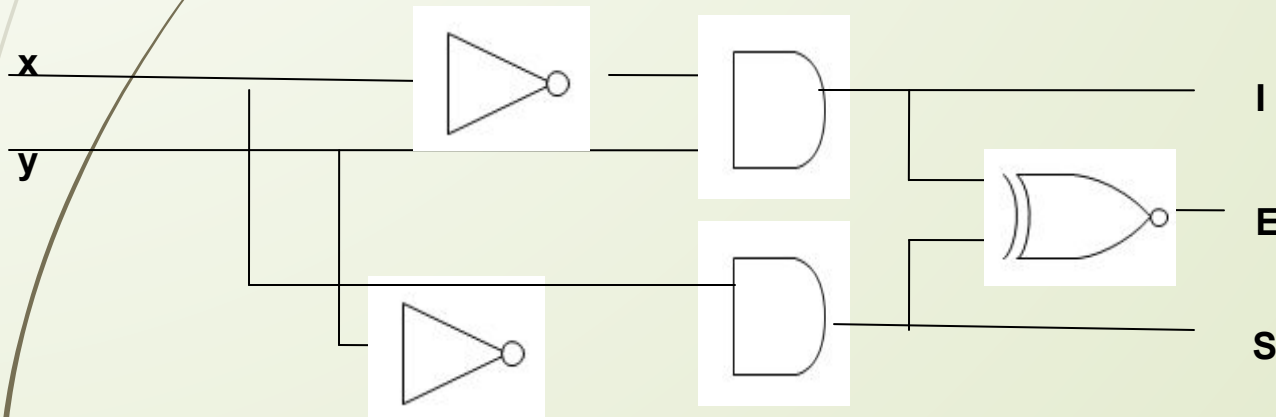
$$S = x \cdot y$$

$$I = \bar{x} \cdot y$$

$$E = \bar{x} \cdot \bar{y} + x \cdot y$$

$$= \bar{x} \oplus \bar{y}$$

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



1- Additionneur

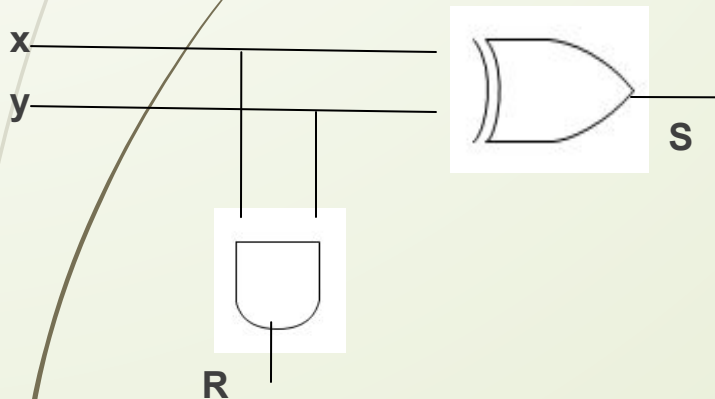
40

– Addition de 2 bits x et y (demi-additionneur) :

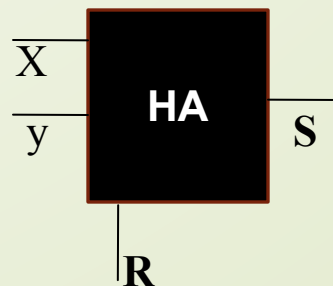
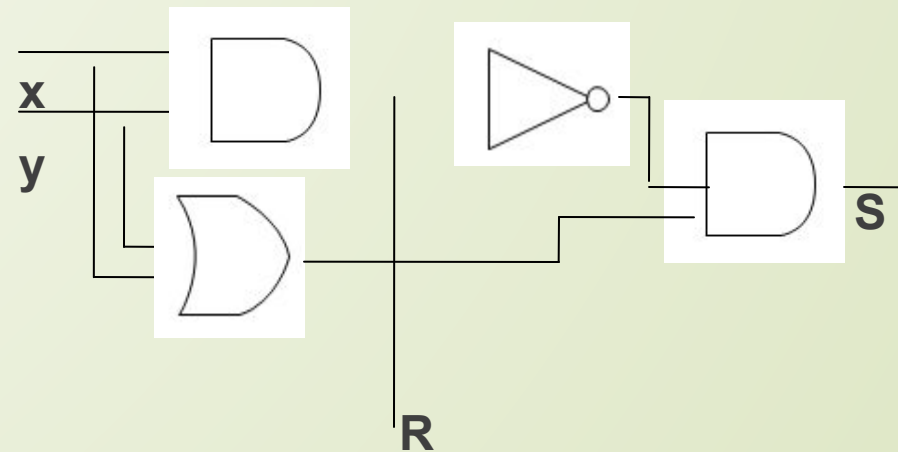
x	y	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

table de vérité □ Somme $S = x \oplus y$
 $= \bar{x}.y + x.\bar{y}$
 $= (x + y).(x + \bar{y})$
 $= (x + y).(\overline{x.y})$
 retenue $R = x.y$

$$S = x \oplus y, R = x.y$$



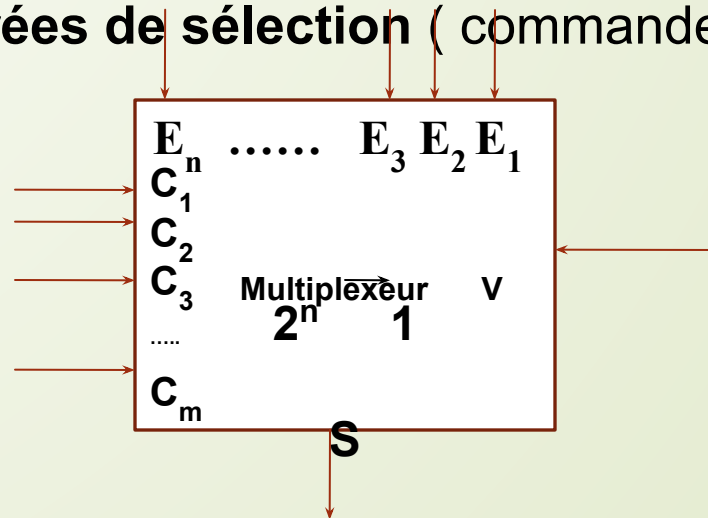
$$S = (x + y).(\overline{x.y}), R = x.y$$



3- Multiplexeur:

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
 - **m entrées de sélection** (commandes)



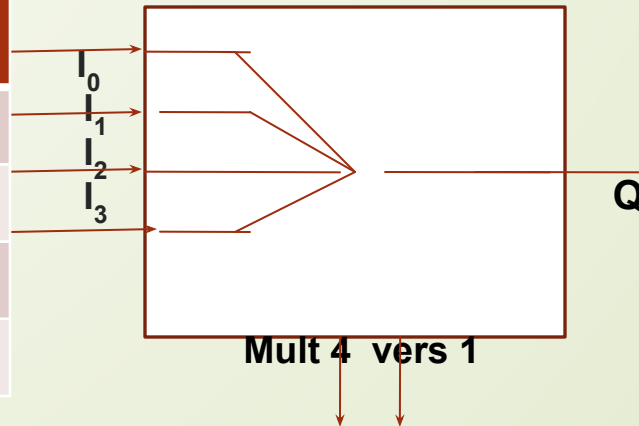
Applications des multiplexeurs

Conversion parallèle/série : aiguiller les informations présentes en parallèle à l'entrée du MUX en des informations de type série en sortie ; toutes les combinaisons d'adresses sont énumérées une par une sur les entrées de sélection.

Exemple: Multiplexeur $2^2 = 4$ vers 1

T. Vérité

S_1	S_0		Q
0	0		I_0
0	1		I_1
1	0		I_2
1	1		I_3



$$Q = \bar{S}_1 \cdot \bar{S}_0 \cdot I_0 + \bar{S}_1 \cdot S_0 \cdot I_1 + S_1 \cdot \bar{S}_0 \cdot I_2 + S_1 \cdot S_0 \cdot I_3$$

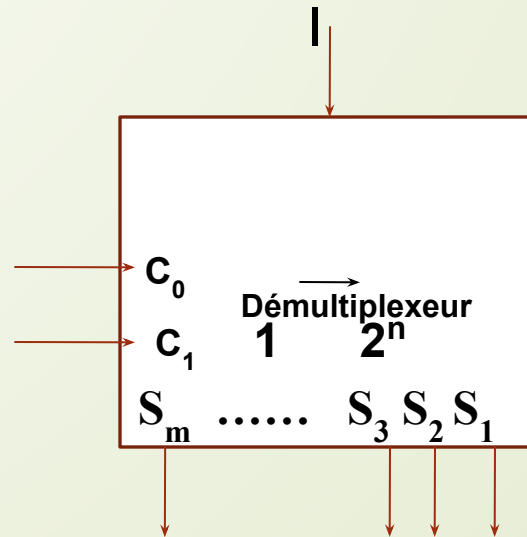
4- Démultiplexeur:

43

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**
- **m entrées de sélection (commandes)**



5- Codeur (ou encodeur):

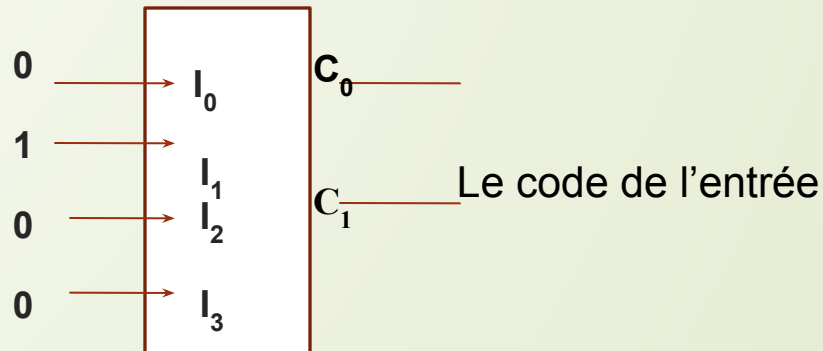
Faire correspondre un mot code à un symbole.

44

2^n entrées

n sorties

1 entrée parmi 2^n
est active



Mot

Code

Traduit le rang de **l'entrée active** en un **code binaire**.

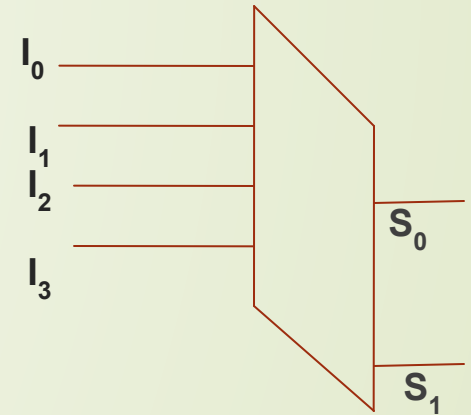
Exemple : Clavier / Scan code
 Caractère / Code ASCII

Exemple: Encodeur binaire (4 \rightarrow 2) :

Table de vérité .

45

Entrées					Sorties	
Codage 1 parmi 2 ⁿ					Nombre binaire de n bits(n=2)	
I ₃	I ₂	I ₁	I 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

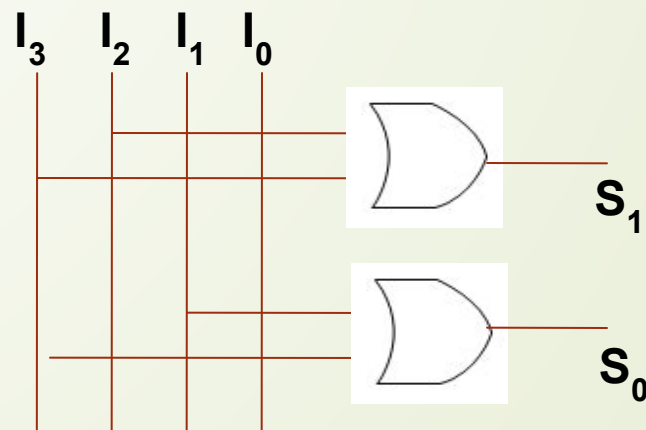


Fonctions logiques:

S1=1 si (I2=1) ou (I3=1) ; **S1=I2+I3**

S0=1 si (I1=1) ou (I3=1) ; **S0=I1+I3**

Logigramme du codeur:



Entrées					Sorties	
Codage 1 parmi 2^n					Nombre binaire de n bits	
I_3	I_2	I_1	I_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

Si nous activons **simultanément les entrées I_1 et I_2** du codeur ci-dessus, les sorties de S_1S_0 présente le nombre **11** qui ne correspond pas au code de l'une ou de l'autre des entrées activées.

C'est plutôt le code qui représente l'activation de S_3 .

➡ On utilise un **codeur de priorité** qui **choisit le plus grand nombre** lorsque plusieurs entrées **sont activées à la fois**.

Exemple: lorsque I_1 et I_2 sont activées simultanément S_1S_0 sera égale à 10 (2 en binaire) qui représentent l'activation de I_2

6- Le décodeur binaire:

C'est un circuit combinatoire qui est constitué de :

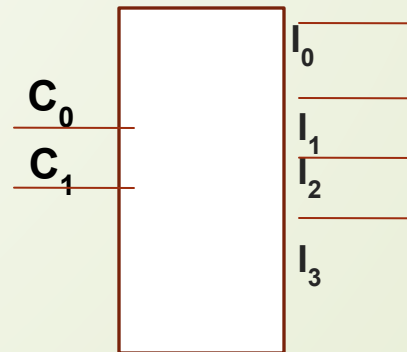
– n : entrées de données

– 2^n sorties

– Pour chaque combinaison en entrée **une seule sortie est active à la fois**

n entrées

2^n sorties



Active **la ligne de sortie** correspondant au **code binaire** présent en entrée

Suivant le type de décodeur, la sortie peut traduire **deux fonctions**:

- **Convertisseur de code**: à un code d'entrée correspond un code de sortie.
Exemple: Un décodeur binaire octal **possède 3 bits d'entrés** permettant $2^3 = 8$ combinaisons pour activer chacun **des 8 sorties de l'octal**.
- **Sélecteur de sortie**: Une **seule sortie parmi les M disponibles est activée** à la fois en fonction de la **valeur binaire affichée** à l'entrée.

Ces fonctions permettent d'activer (sélectionner) un circuit intégré parmi plusieurs.

Principe d'un décodeur 1 parmi 4

2 bits permettant $2^2 = 4$ combinaisons

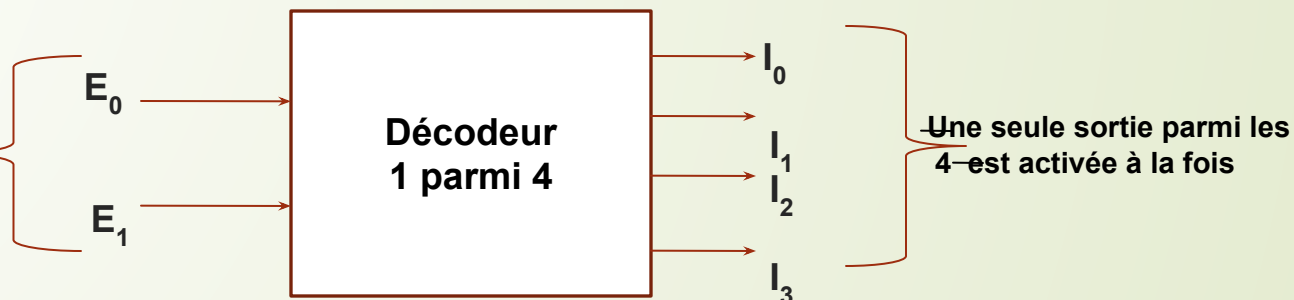


Table de fonctionnement

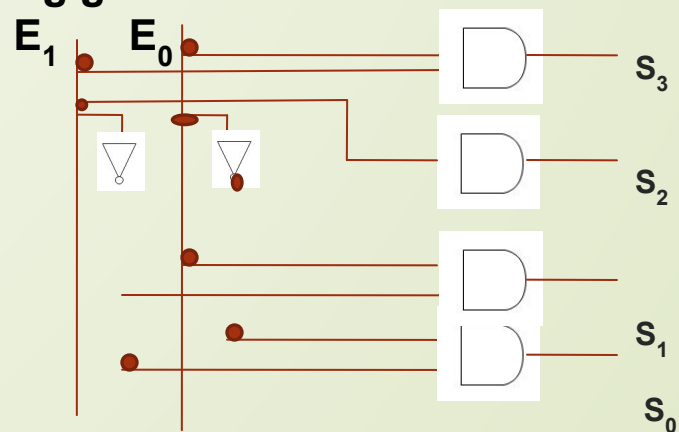
Code binaire d'entrée		Codage 1 parmi 4 sorties			
E_1	E_0	S_3	S_2	S_1	S_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Equations de sorties:

$$S_0 = \overline{E_1} \cdot \overline{E_0} \quad S_1 = E_1 \cdot \overline{E_0}$$

$$S_2 = \overline{E_1} \cdot E_0 \quad S_3 = E_1 \cdot E_0$$

Logigramme du décodeur:

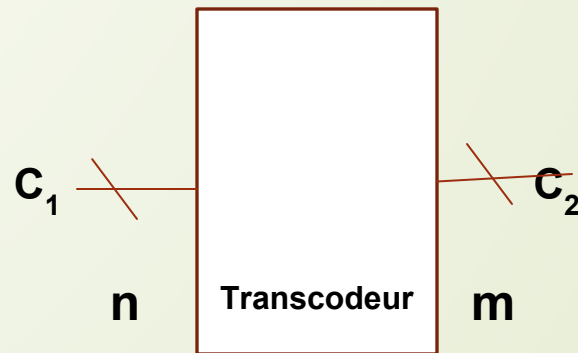


6- Le transcodeur:

50

C'est un circuit combinatoire qui permet de **transformer un code X (sur n bits) en entrée** en un **code Y (sur m bits) en sortie**.

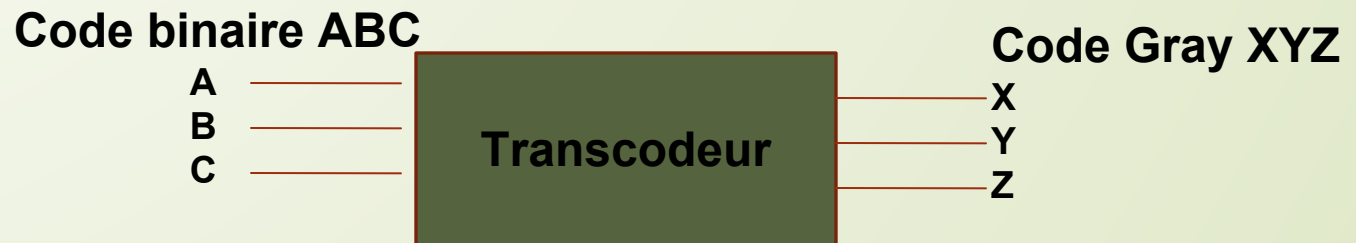
– Passage d'un code C_1 à un code C_2



Active la ligne de sortie correspondant au code binaire présent en entrée

Exemples:

1- Cherchons le circuit d'un transcodeur qui permet de convertir le **code binaire 3 bits en code Gray**.



Construction du code Gray sur 3 bits:

code construit de telle façon qu'à partir du chiffre 0, chaque nombre consécutif diffère du précédent immédiat d'un seul digit.

52

Table de vérité

Entrées			Sorties		
A	B	C	X	Y	Z
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

En passant par Karnaugh pour les sorties X, Y et Z:

$$X=A$$

$$Y = \bar{A}.B + A.\bar{B} \\ = A \oplus B$$

$$Z = \bar{B}.C + B.\bar{C} \\ = B \oplus C$$

- 2^{ème} Exemple:

53

Transcodeur qui transforme **les chiffres de 0 à 9** (sous **format binaire**) en une **configuration alimentation en diodes** (ou **LCD**).

Pour coder 0 en 9 en binaire; **il faut 4bits** (de 0000 à 1001).

Exemple peut **être étendu** pour les caractères hexadécimal (0 à 9 et a à f)

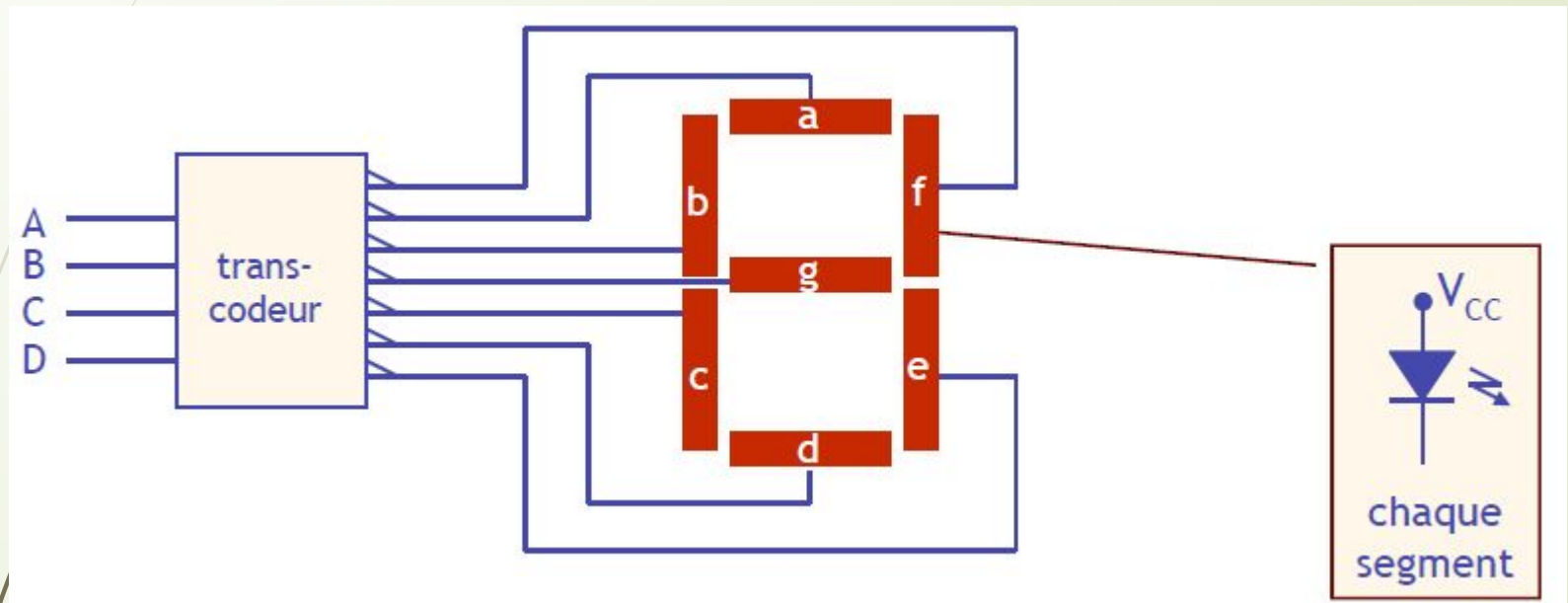


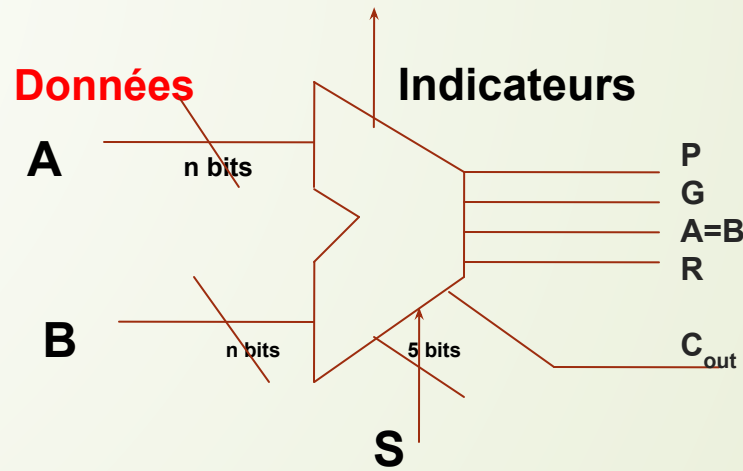
Schéma fonctionnel du transcodeur BCD 7 segments

Exercice:

- Etablir la table de vérité de ce transcodeur,
- Avec les tableaux de Karnaugh établir la fonction logique associée à chacune des sorties de ce transcodeur (en faisant introduire les **segments a, b, c, d, e, f et g**)

7- Unité arithmétique et logique (UAL):

54



Sélection d'une opération (32 cas)

Instruction

Représentation schématique d'une UAL

UAL effectue les opérations de base : opérations logiques, *opérations arithmétiques, comparaisons.*

Un code d'entrée détermine la partie du circuit qui va fournir le résultat.

À la fin de l'opération des indicateurs sont fournis.

Résultat

Exemple :

$$R = A + \bar{B}$$

$$R = A + B$$

$$R = A + B + 1$$

...

$$R = A \text{ ou } B$$

$$R = A \text{ nand } B$$

...

Exercice:

Dresser le circuit combinatoire d'une UAL opérant **sur 2 données binaires x et y** en entrée et qui possède **4 opérations logiques (et, ou, non et addition binaire)**

Donc, code opérations sur **2 bits** (F0 et F1)

et : code opération = **00**

ou : code opération = **01**

non : code opération = **10**

addition : code opération = **11**

Les circuits séquentiels

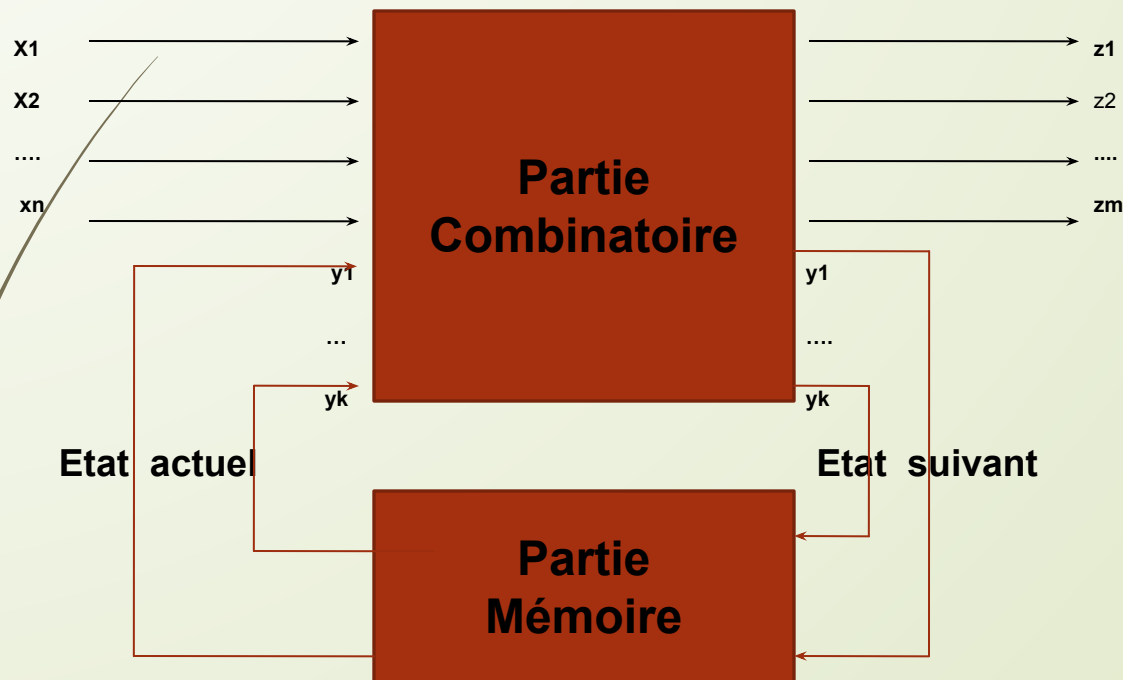
56

Les circuits séquentiels :

- * Construits à partir de **circuits combinatoires**;
- * Se caractérisent par une **capacité de mémorisation**: la valeur des **sorties à l'instant t** dépend de la valeur des entrées $e(t)$ et de la valeur des sorties à l'instant $t-1$.

n variables d'entrées

m variables de sortie

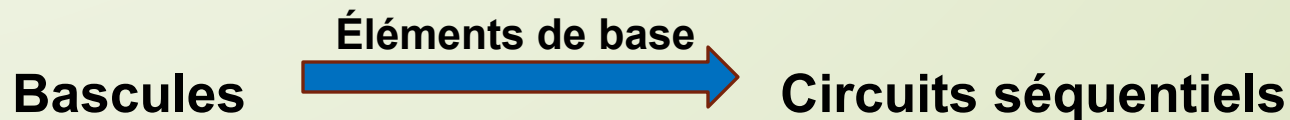


Les **bascules** sont des composants électroniques largement répandus et indispensables à la plupart des circuits.

Une bascule possède la fonction de **mémorisation et de basculement**.

Une bascule constitue une **cellule mémoire élémentaire** car l'état de sortie reflète l'état des entrées.

Les bascules sont des éléments **bistables**, c'est-à-dire que chacune des deux sorties possède deux **états stables**, le passage d'un état à l'autre **étant provoqué par des signaux de commande**.



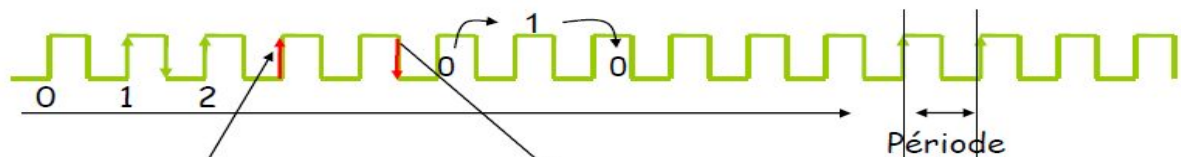
En principe les états des deux **sorties** de la bascule sont **complémentaires** (0 et 1 ou Q et \overline{Q})

Les bascules sont **capables de conserver l'état de leur sortie même si la combinaison des signaux d'entrée ayant provoqué cet état de sortie disparaît**

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**.

Horloge (Clock)

- **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.



Front
montant

Front
descendant

La période T est en seconde

Fréquence = nombre de changement par seconde en hertz (Hz)

Fréquence = $1/\text{période}$

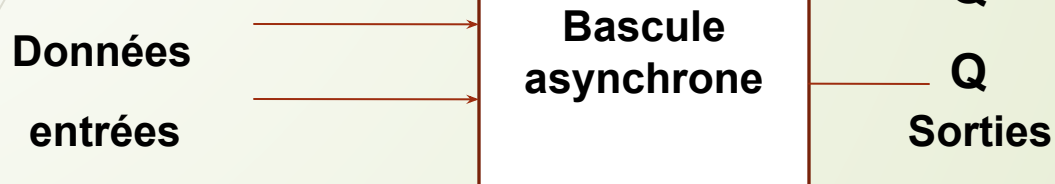
Une horloge de 1 hertz a une période de 1 seconde

..... 1 megahertz..... 1 microseconde

..... 1 gigaHz..... 1 nanoseconde

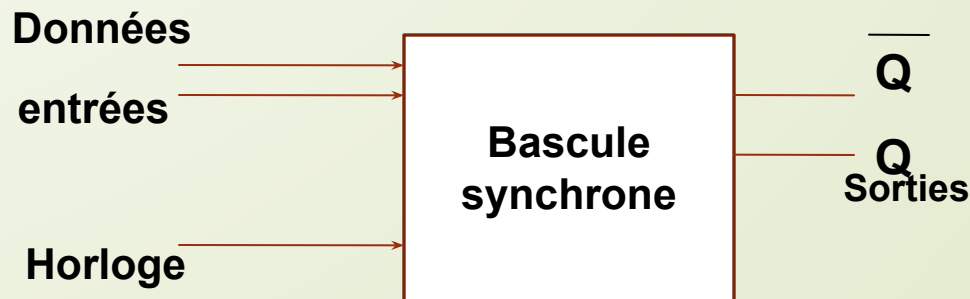
60

Une bascule est dite **asynchrone** lorsqu'elle ne dépend pas du **signal d'horloge (Clock ou CLK)**; elles ne sont pas **asservies à une horloge** et prenant en compte **leurs entrées à tout moment**.



Les bascules **synchrones** sont asservies à **des impulsions d'horloge** et donc **insensibles aux bruits entre deux tops**

Par conséquent, un reset **synchrone** ne prendra effet que lors du **flanc actif du coup de clock suivant**..

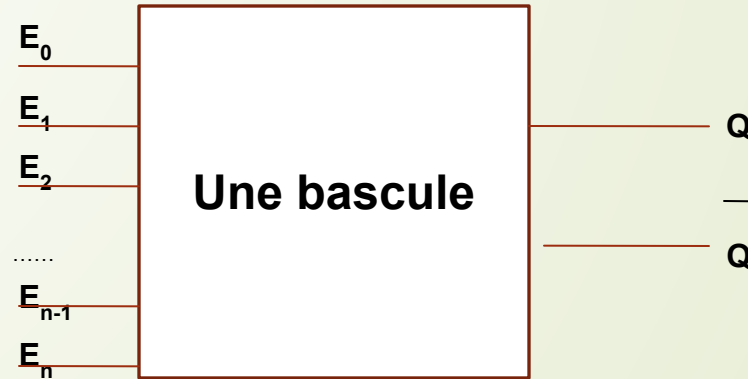


On distingue les bascules de type: **RS, D, RSH, JK et T**

Une bascule constitue une **cellule mémoire élémentaire** car **l'état de sortie reflète l'état des entrées**.

61

Chaque bascule possède **des entrées et deux sorties Q** et **Q**



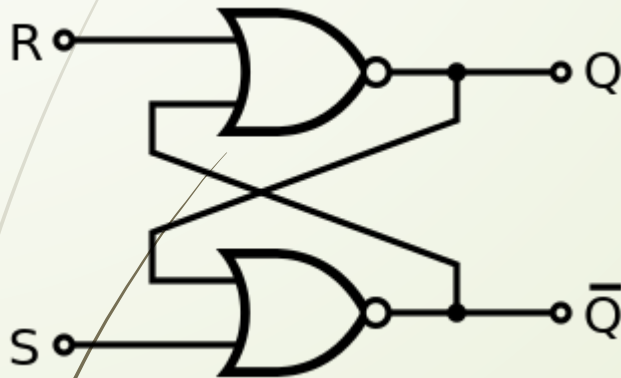
$$Q^+ = F(E_i, Q)$$

Types de bascules : **RS, RSH ,D ,JK , T**

A- Bascule R – S :

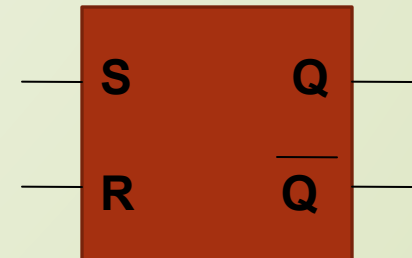
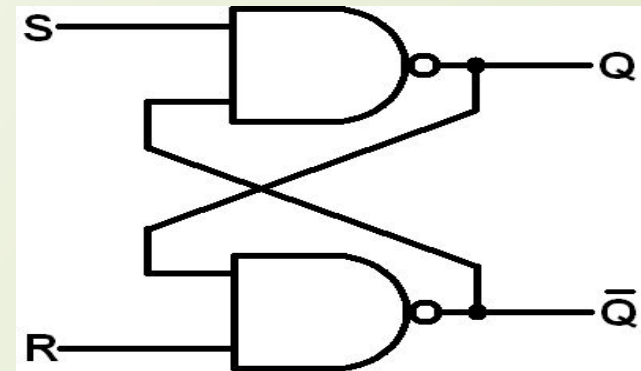
62 C'est une bascule à deux entrées nommées **S** (pour **Set** qui correspond à une mise à 1 de la sortie) et **R** (pour **Reset** ou mise à zéro de la sortie).

Pour la réalisation, on peut utiliser soit deux **portes NAND** soit deux **portes NOR**. Dans les deux cas, une entrée de chaque porte est reliée à la sortie de l'autre: c'est un montage à réaction totale.



T vérité

R	S	Q_{t+1}	Commentaire
0	0	Q_t	Ne change pas d'état
0	1	1	Mise à 1
1	0	0	Mise à 0
1	1	?	Interdit



B- Bascule R – S - H:

C'est une bascule RS avec une condition complémentaire : en **entrée H actif**

63 Si **H=1** mémoire classique

Si **H=0** mémoire figée

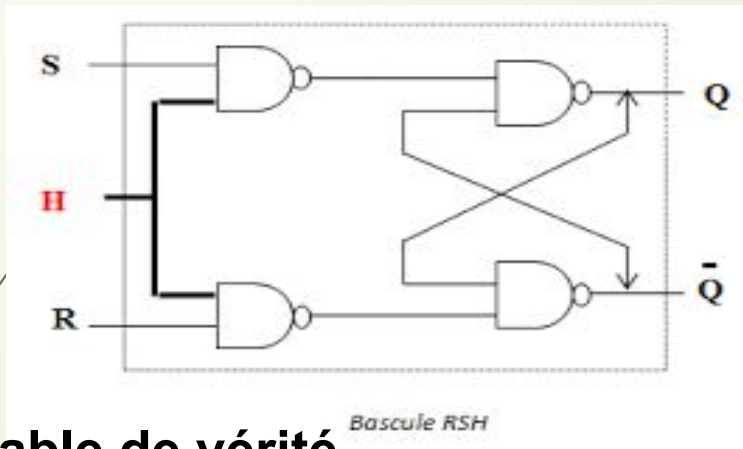
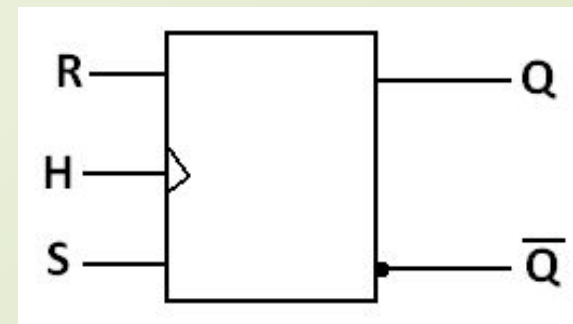


Table de vérité

R	S	Q_{t+1}	Commentaire
0	0	Q_t	Ne change pas d'état
0	1	1	Mise à 1
1	0	0	Mise à 0
1	1	?	Interdit



C- Bascule D:

C'est une bascule **synchronisée** sur le front montant ou descendant

64

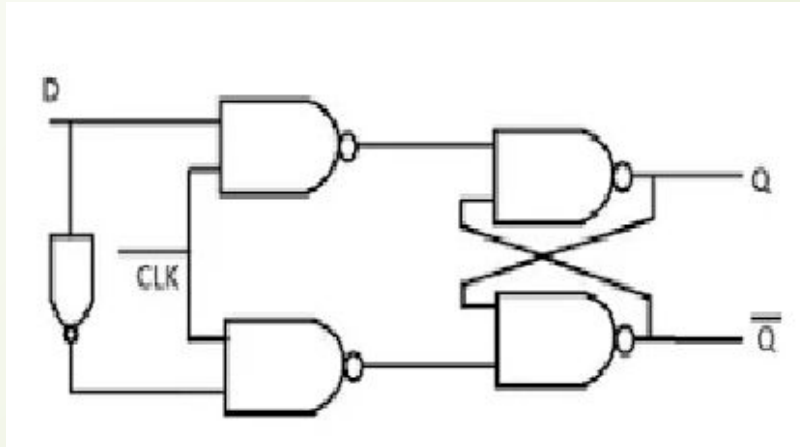
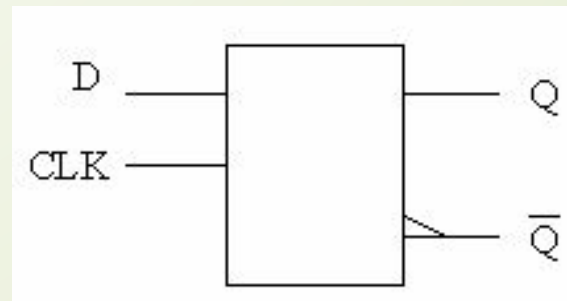


Table de vérité

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



D- Bascule JK:

Une autre manière de **lever les cas interdits** est la bascule JK (pas de signification particulière pour J et K).

65 . Cette bascule a le même comportement qu'une bascule **RS** sauf que si $J = K = 1$) alors $Q_{t+1} = \overline{Q}_t$.

*En utilisant les portes **NAND** on obtient le logigramme suivant:*

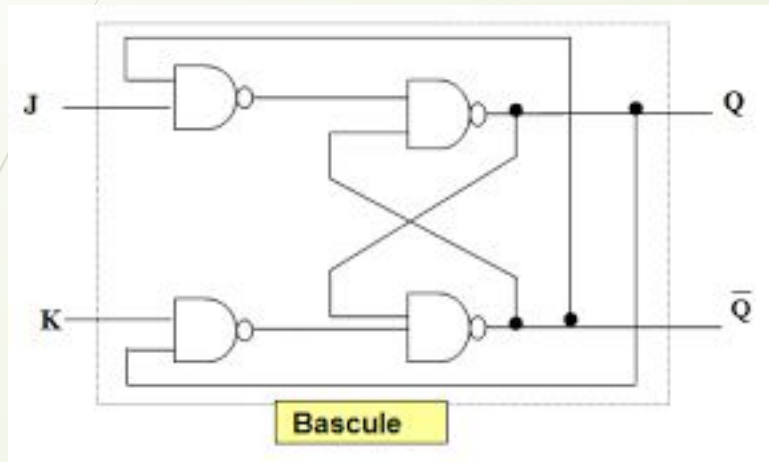
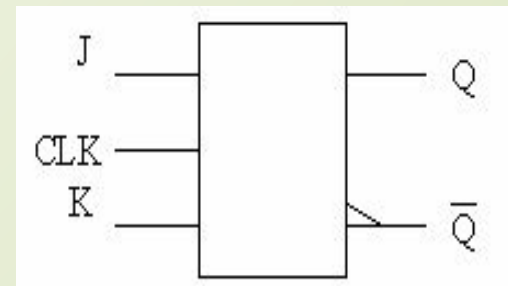


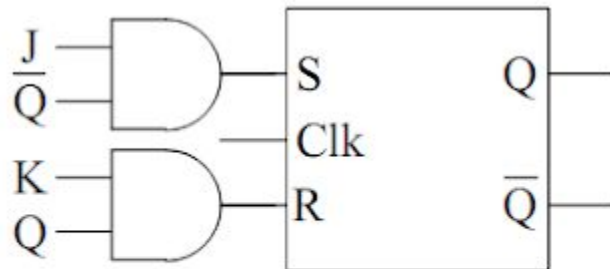
Table de vérité

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

Le circuit:



Une autre façon de dresser le logigramme de la bascule JK en utilisant les portes AND est comme suit et ce, en asservissant les entrées R et S aux sorties Q et \bar{Q} :



Nous avons alors pour les signaux **R** et **S**
 $\mathbf{S = J \cdot Q}$ et $\mathbf{R = K \cdot Q}$

Figure 10

Ce qui nous permet de construire la table de vérité de la bascule JK

J_n	K_n	Q_n	\bar{Q}_n	S	R	Q_{n+1}
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	1	0	1	0	0	0
0	1	1	0	0	1	0
1	0	0	1	1	0	1
1	0	1	0	0	0	1
1	1	0	1	1	0	1
1	1	1	0	0	1	0

Cette table peut **se résumer** sous la **forme suivante** :

J_n	K_n	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	\bar{Q}_n