CHAPITRE 2 : CIRCUITS COMBINATOIRES

Mme KAZI TANI ADILA

Structure Machine 2

Introduction

- Les machines numériques modernes (ordinateurs, tablettes, smartphones, etc.) sont constituées de deux types de circuits :
  - Combinatoires
  - Séquentiels

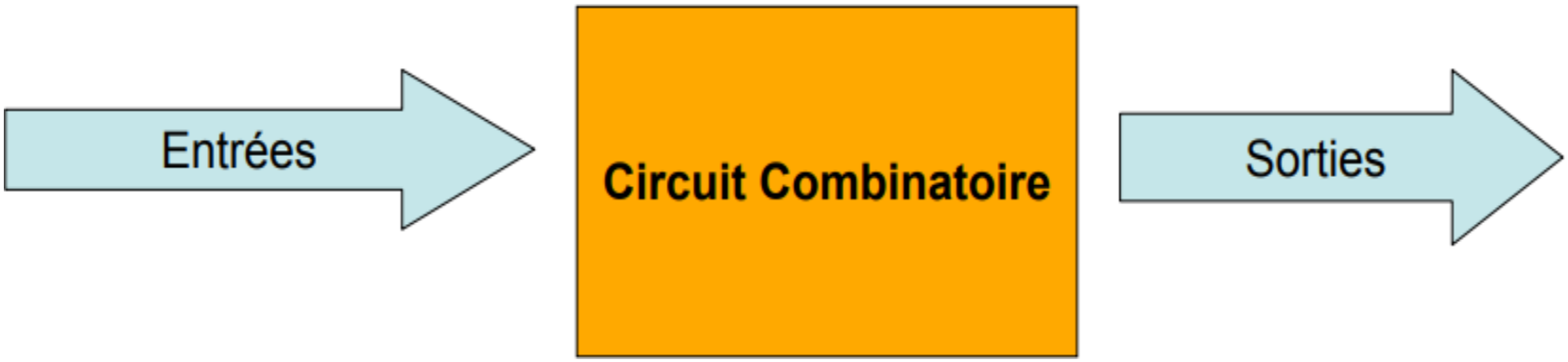
Introduction

- **Circuit logique combinatoire** est un circuit numérique dont les sortiesne dépendentque de l'état logique de ses entrées.

- **Circuit logique séquentiel** est un circuit numérique dont les sorties dépendent de l'état logique de ses entrées, ainsi que de l'´etat actuel de ce circuit.

Circuit combinatoire

- Un circuit combinatoire est constitué d'éléments logiques élémentaires appelés portes logiques, elle reçoivent des signaux appliqués en entrée et produisent des signaux en sortie.



Symbole logique d'un circuit combinatoire

Synthèse d'un circuit combinatoire

- Dans ce chapitre, nous nous intéressons principalement à la **synthèse des circuits logiques** combinatoires de base (les additionneurs, les décodeurs, les multiplexeurs, etc.), à partir desquels on peut concevoir d'autres circuits plus complexes.
- La synthèse d'un circuit combinatoire consiste tout simplement à réaliser ce circuit à partir de l'énoncé ou d'un cahier des charges décrivant les fonctions ou le rôle que le circuit doit remplir.

Synthèse d'un circuit combinatoire

- Il s'agit donc de déterminer le logigramme associé aux fonctions logiques constituant le circuit en connaissant la définition de chacune de ces fonctions.

Synthèse d'un circuit combinatoire

- Voici les étapes à suivre pour réaliser la synthèse d'un circuit logique combinatoire :

1. **Déterminer les entrées et les sorties du circuit** à partir de la description du problème (c'est l'étape la plus importante, il faut bien comprendre l'énoncé du problème afin de déterminer correctement le nombre de variables d'entrée et de variables de sortie du circuit à réaliser).

Synthèse d'un circuit combinatoire

2. Etablir la **table de vérité** des différentes sorties en fonction des entrées.
3. Établir les **équations logiques.**
4. **Simplifierles équations** de chacune des fonctions logiques.
5. **Établir le logigramme** (c.à.d. le circuit logique).

Synthèse d'un circuit combinatoire

Exemple : (Circuit 2/3)

Etablissons le logigramme d'un circuit logique comportant 3 entrées et 1 sortie, celle-ci étant à l'état 1 si au moins 2 des trois entrées sont à l'état 1.

Synthèse d'un circuit combinatoire

Exemple : (Circuit 2/3)

- Table de vérité :

a	b	c	f(a, b, c)
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

- Expression logique .

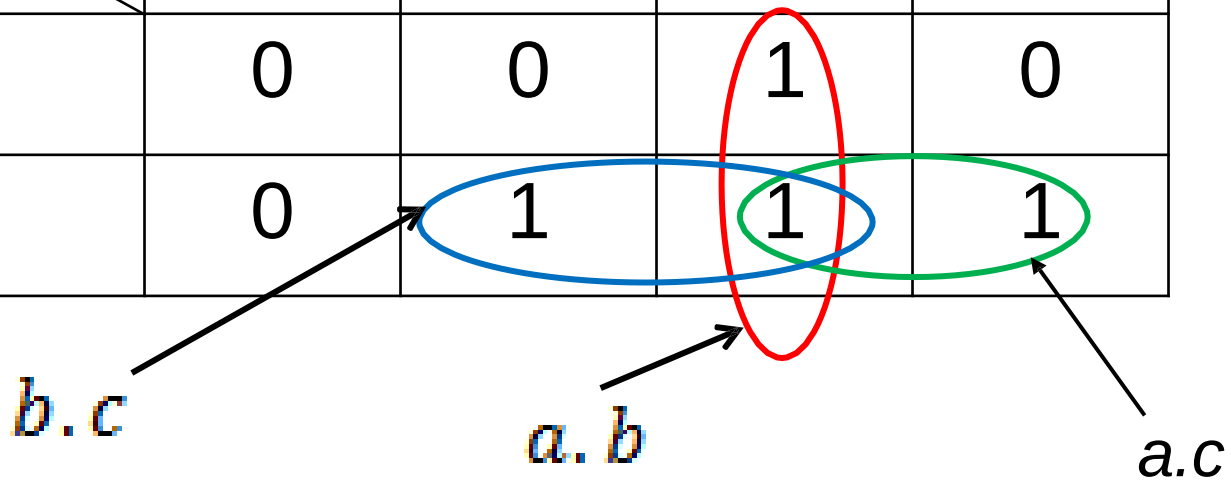
$$f(a,b,c) = \bar{a}.b.c + a.\bar{b}.c + a.b.\bar{c} + a.b.c$$

Synthèse d'un circuit combinatoire

Exemple : (Circuit 2/3)

- Simplification:

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



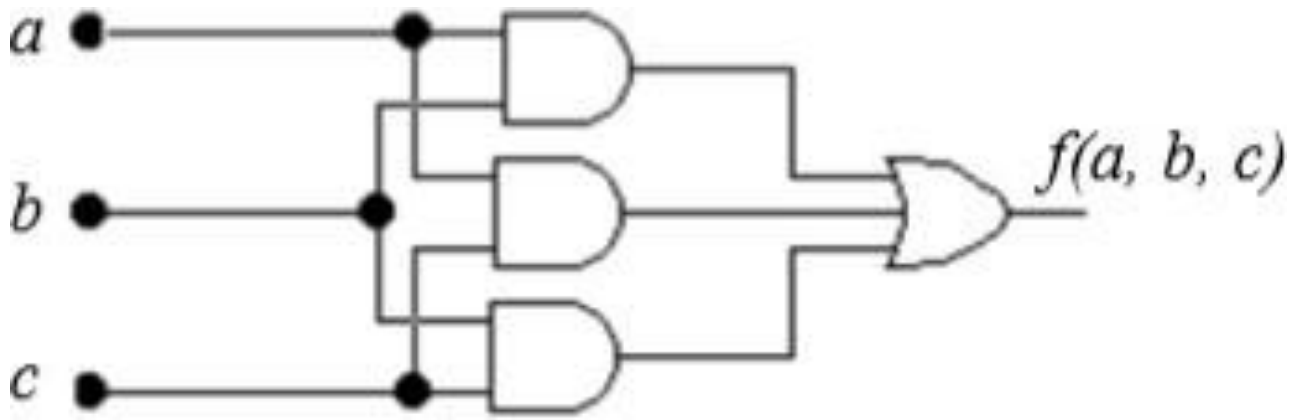
- Expression logique :

$$f(a,b,c) = a.b + a.c + b.c$$

Synthèse d'un circuit combinatoire

Exemple : (Circuit 2/3)

- Logigramme :



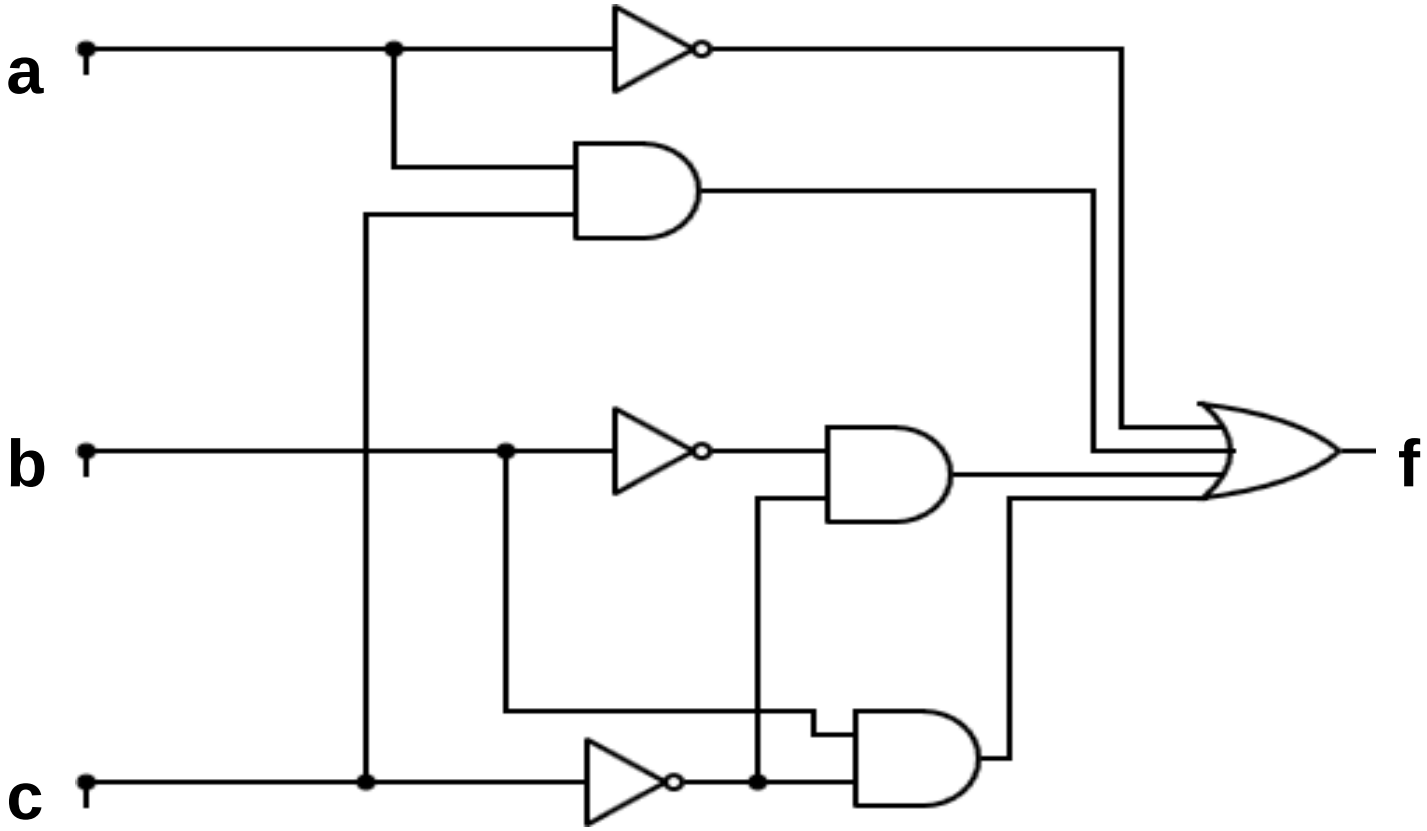
Analyse d'un circuit combinatoire

- Pour analyser un circuit combinatoire, on suit les étapes suivantes :

1. Déterminer les **expressions logiques** des variables de sortie en fonction des valeurs de ses entrées.
2. Dresser la **table de vérité** du circuit.
3. Déduire par un énoncé décrivant le **rôle du circuit.**

Analyse d'un circuit combinatoire

Exemple : Analyser le circuit logique suivant :



Analyse d'un circuit combinatoire

Exemple :

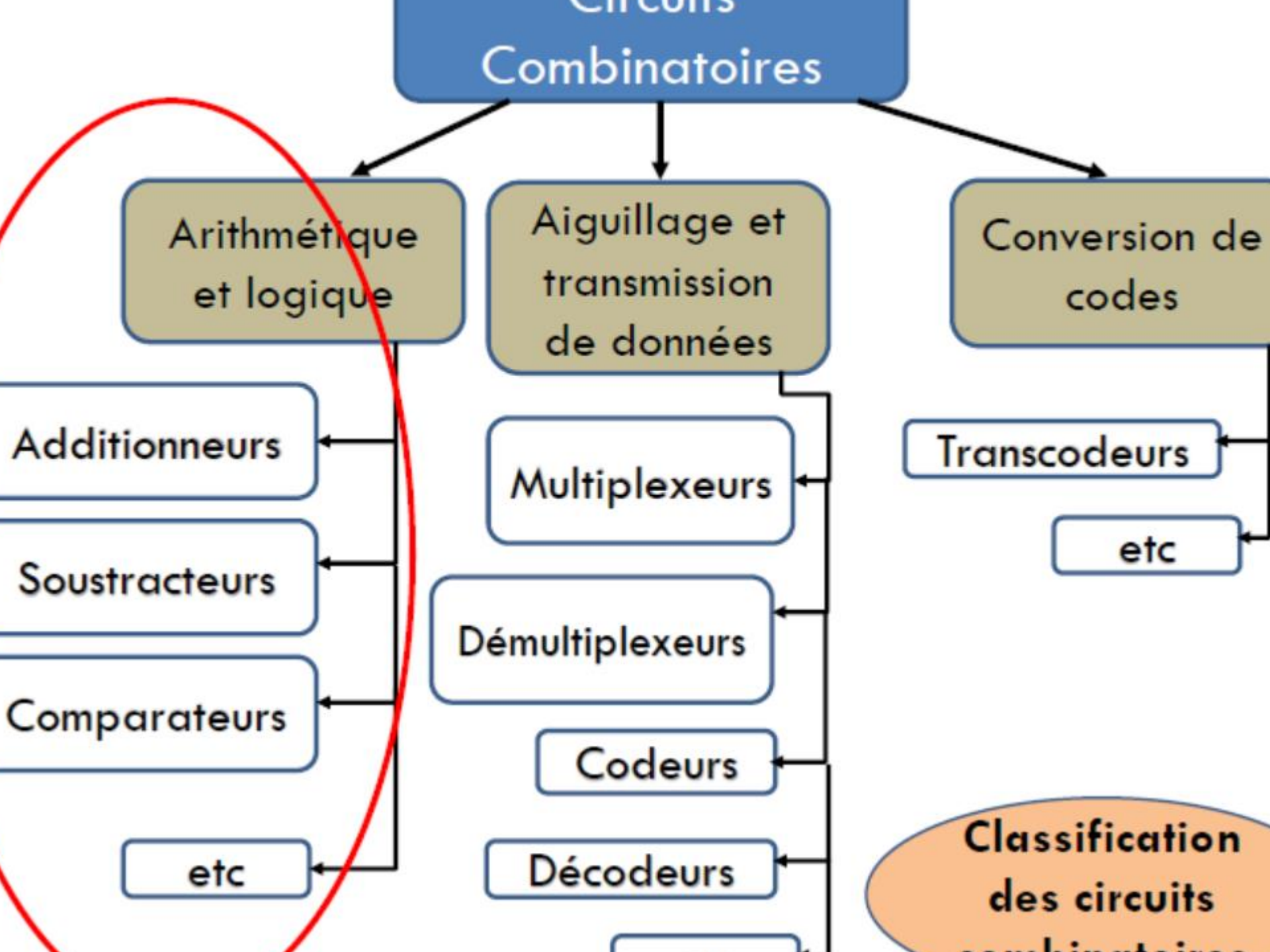
- Expression logique : $f(a,b,c) = \bar{a} + a.c + \bar{b}.\bar{c} + b.\bar{c}$
- Table de vérité :
- Le rôle du circuit est de produire la constante 1.

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

Classification des circuits combinatoires

Dans un ordinateur, nous pouvons distinguer trois différentes classes de circuits logiques combinatoires :

- Les circuits de calcul **arithmétiques et logiques.**
- Les circuits **d'aiguillage et de transmission de données.**
- Les circuits **de conversion de codes.**


Circuits Combinatoires
Arithmétique et logique
Aiguillage et transmission de données
Conversion de codes
Additionneurs
Soustracteurs
Comparateurs
etc
Multiplexeurs
Démultiplexeurs
Codeurs
Décodeurs
Transcodeurs
etc
Classification des circuits

Circuits arithmétiques et logiques

- Les circuits arithmétiques et logiques combinatoires permettant d'effectuer des calculs arithmétiques (addition, soustraction, multiplication) sur des entiers ou des nombre en virgule flottantes et des opérations logiques comme des négations, des ET, des OU ou des OU Exclusifs.
- On les trouve le plus souvent dans les unité de calculs des ordinateur communément appelées ALU (arithmetic logic unit) en anglais.

Circuits arithmétiques et logiques

- Nous allons détailler les circuits arithmétiques et logiques suivants :
  - Additionneur.
  - Soustracteurs.
  - Comparateurs.

Additionneurs

- Additionneurs :
  - **Demi additionneur** : 2 entrées sur 1 bit, deux sorties sur 1 bit.
  - **Additionneurcomplet** : 3 entréessur 1 bit, deux sorties sur 1 bit.
  - **Additionneur sur n bits.**

Additionneurs

- **Demi additionneur:**

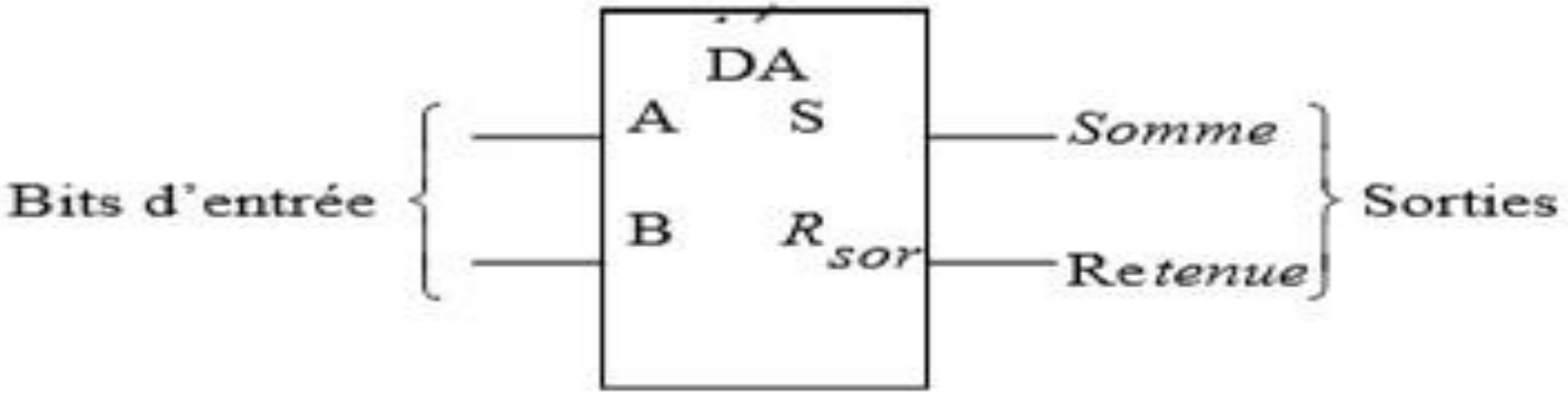
Rappelons les règles d'addition binaire :

Entrées A B	*Sorties* Somme (S)	Retenue (R_{sor})
0 + 0	0	0
0 + 1	1	0
1 + 0	1	0
1 + 1	0	1

Additionneurs

Demi additionneur :

Ces opérations s'effectuent par un circuit logique appelé un **demi-additionneur**, qu'on note DA. Un DA est symbolisé par le symbole logique suivant :



Symbole logique d'un DA

Additionneurs

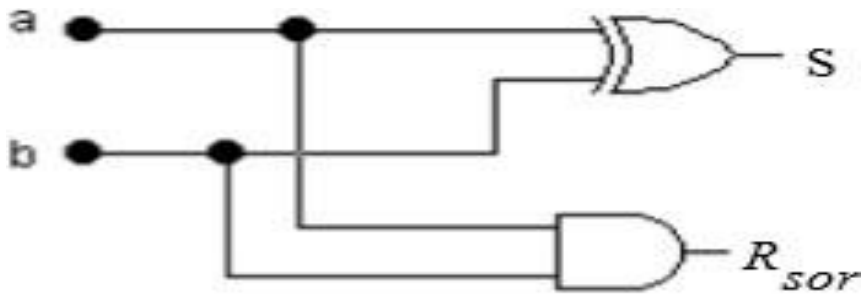
Demi additionneur :

- Expressions logiques

$$S = A \oplus B$$

$$R_{sor} = A.B$$

- Logigramme



Demi-Additionneur

Additionneurs

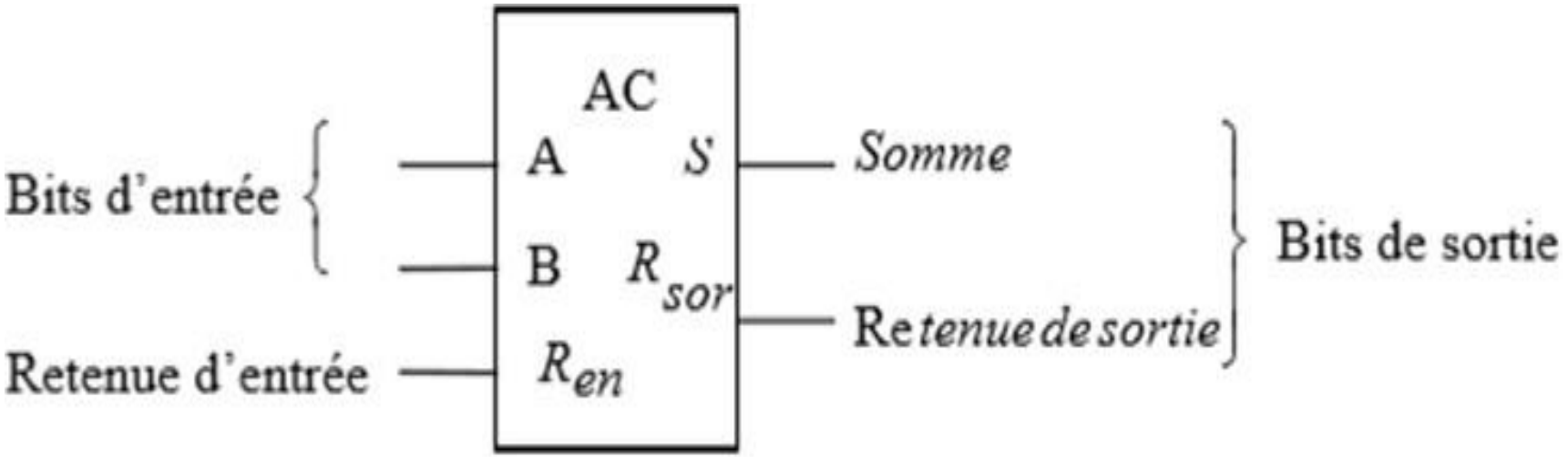
Additionneur complet :

L'additionneur complet (noté AC) est un circuit combinatoire qui permet de réaliser la somme sur un bit de deux nombre A et B tout en tenant en compte la retenue précédente R_{en}. Ce circuit contient donc trois entrées A, B et R_{en} et génère deux sorties : S qui représente la somme de A et B et R_{en} sur un bit et R_{sor} qui représente la retenue.

Additionneurs

- **Additionneur complet :**

le symbole logique d'un AC est donné par le schéma suivant :



Symbole logique d'un AC

Additionneurs

- **Additionneur complet :**

Table de vérité

A	B	R_{en}	S	R_{sor}
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

Additionneurs

- **Additionneur complet :**

Expressions logiques :

$$S = \bar{A}.\bar{B}.Re_n + \bar{A}.B.\overline{R_{en}} + A.\bar{B}.\overline{R_{en}} + A.B.R_{en}$$

$$= \bar{A}.\bar{B}.Re_n + R_{en}.(\bar{A}.B. + A.\bar{B}) + A.B.R_{en}$$

$$= R_{en}.(\bar{A}.\bar{B}. + A.B) + R_{en}.(\bar{A}.B. + A.\bar{B})$$

$$= R_{en}.(\overline{A \oplus B}) + R_{en}.(A \oplus B)$$

$$S = R_{en} \oplus (A \oplus B)$$

Additionneurs

- **Additionneur complet :**

Expressions logiques :

$$R_{sor} = \bar{A}.B.R_{en} + A.\bar{B}.R_{en} + A.B.\overline{R_{en}} + A.B.R_{en}$$

$$R_{sor} = \bar{A}.B.R_{en} + A.\bar{B}.R_{en} + A.B.\overline{R_{en}} + A.B.R_{en}$$

$$R_{sor} = R_{en}.(\bar{A}.B + A.\bar{B}) + A.B.(\overline{R_{en}} + R_{en})$$

$$R_{sor} = R_{en}\,(A \oplus B) + A.B$$

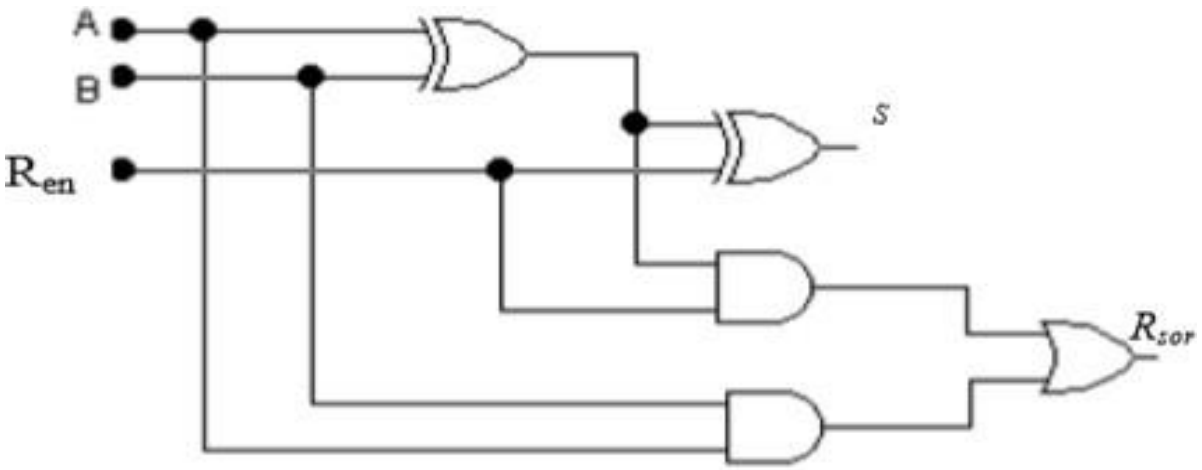
Additionneurs

- **Additionneur complet :**

Expressions logiques :

$$S = R_{en} \oplus (A \oplus B)$$
$$R_{sor} = R_{en}\,(A \oplus B) + A.B$$

Logigramme :



Additionneur complet

Additionneurs

- **Réalisation d'un additionneur complet à l'aide de deux demi-additionneurs :**

Nous avons déjà vu que les sorties d'un AC s'écrivent comme suit :

$$S = R_{en} \oplus (A \oplus B)$$

$$R_{sor} = R_{en}\,(A \oplus B) + A.B$$

Et les sortiesdes demi-additionneurs'écriventcomme suit :

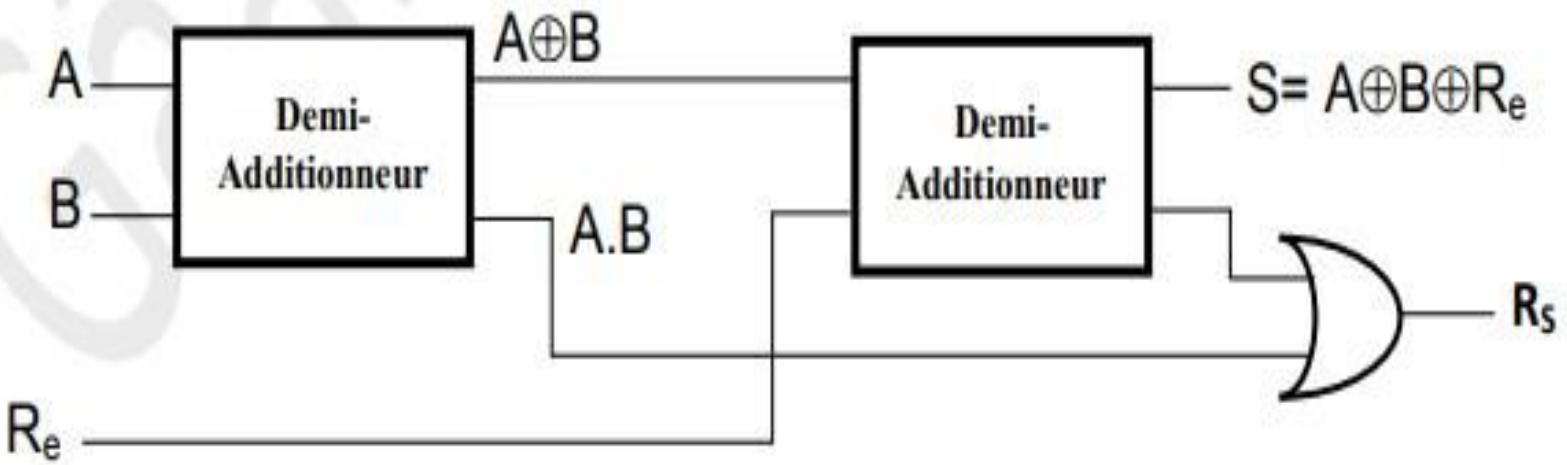
$$S_{DA} = A \oplus B$$

$$R_{DA} = A.B$$

Additionneurs

Réalisation d'un additionneur complet à l'aide de deux demi-additionneurs :

L'additionneur complet est réalisé donc à partir de deux demi- additionneurs : le premier réalise l'addition des deux nombres
A et B et le deuxième réalise l'addition des deux nombre A et E



Additionneurs

- **Additionneur sur n bits :**

Sachant qu'un additionneur complet ne peut traiter que deux nombres de 1 bit et une retenue d'entrée ; pour additionner des nombres de plus d'un bit, il faut utiliser des additionneurs complets supplémentaires.

Un additionneur parallèle à n bits est le branchement en cascade de n additionneurs complets, où la sortie de retenue de chaque additionneur est connectée à l'entrée de retenue de l'additionneur du bit de rang plus élevé suivant.

Additionneurs

- **Additionneur sur n bits :**

L'analyse de ce problème nous apprend que nous avons 2n entrées et n+1 sorties au moins.

Maintenant essayons de comprendre le lien entre les sorties et les entrées. Ce lien est déduit, bien évidemment des règles d'addition que nous avons déjà vue dans le chapitre sur les systèmes de numération que voici :

0 + 0 = 0 ,

1 + 0 = 1 ,

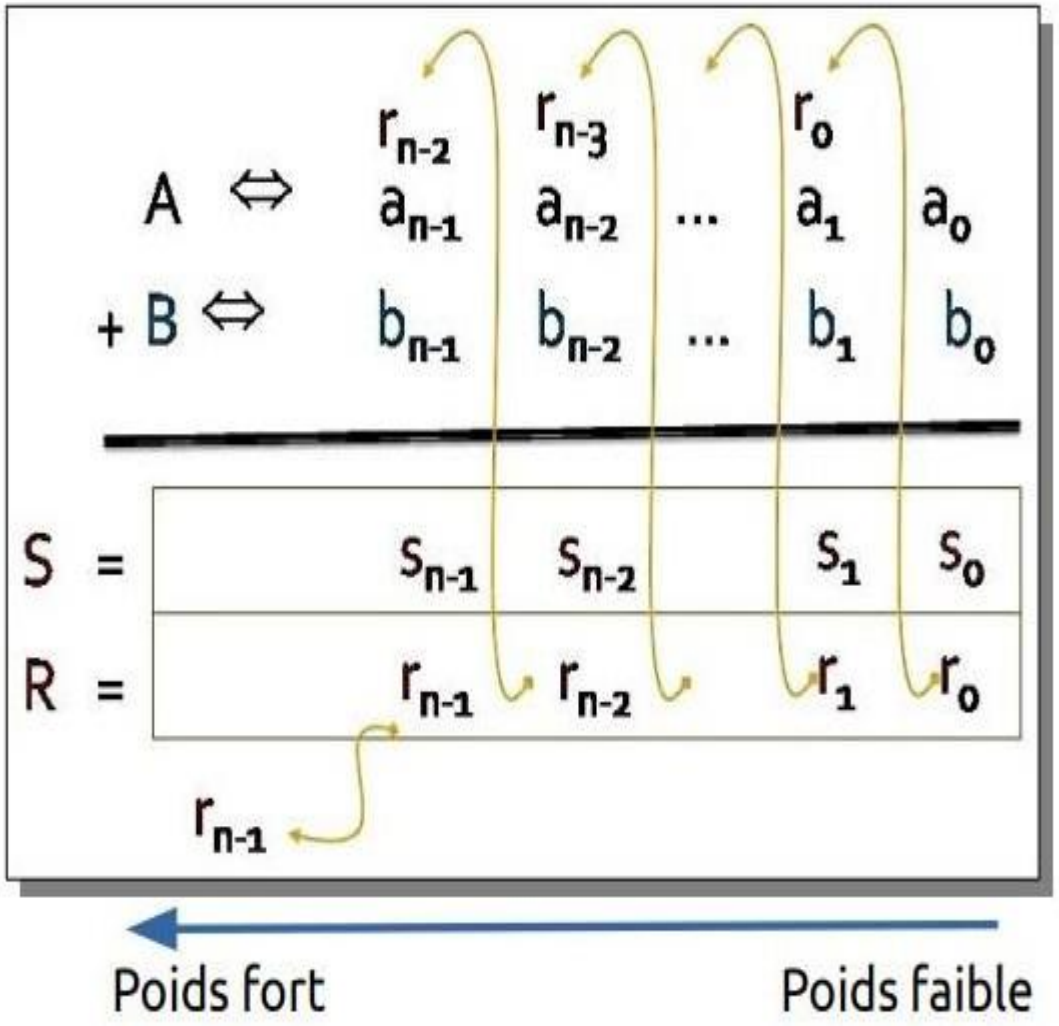
0 + 1 = 1,

1 + 1 = 0 et on retient 1, soit 10

Additionneurs

- **Additionneur sur n bits :**

Ces règles sont applicables pour chaque niveau des bits des deux nombres A et B. Ainsi le calcul se fait en allant du bit de poids faible vers le bits de poids fort.

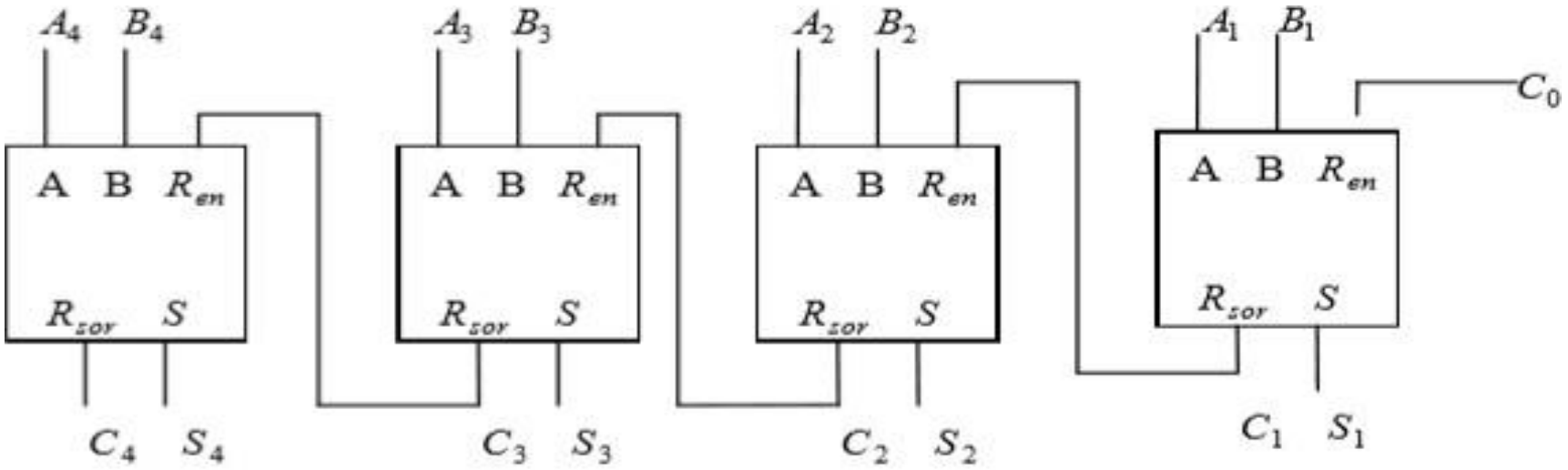


Additionneurs

- **Additionneur sur n bits :**

Exemple : additionneur parallèle à 4 bits

Soit à additionner les nombres binaires : A= A_4 A_3 A_2 A_1 et B = B_4 B_3 B_2 B_1



Additionneur parallèle à 4 bits

Additionneurs

- **Additionneur sur n bits :**

Remarque :

Notez qu'on peut utiliser un DA pour la position de poids le plus faible, ou relier l'entrée de retenue d'un AC à la masse (0), puisqu'il n'y a pas d'entrée de retenue pour la position de bit de poids le plus faible.

Additionneurs

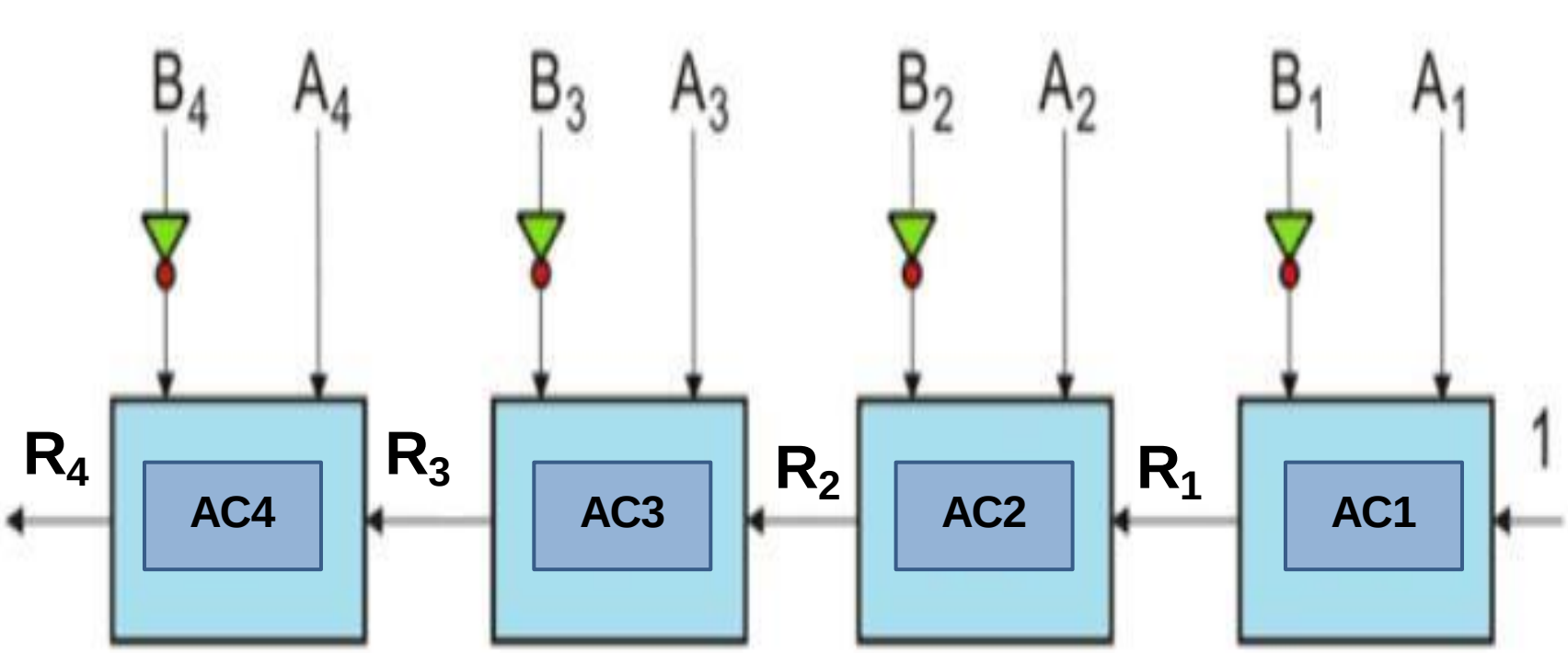
- **L'Additionneur / Soustracteur (en complément à deux)**

Concevoir un circuit qui permet de faire la soustraction en utilisant un additionneur de deux nombres binaires A et B de 4 bit. On rappelle que dans la représentation en complément à 2,

$$A - B = A + \bar{B} + 1$$

Additionneurs

- **L'Additionneur/ Soustracteur (en complément à deux)**



Soustracteurs

- **Demi soustracteur** : 2 entrées sur 1 bit deux sorties sur 1 bits.
- **Soustracteur complet** : 3 entrées sur 1 bit deux sorties sur 2 bits.
- **Soustracteur sur n bits.**

Soustracteurs

– **Demi soustracteur** :

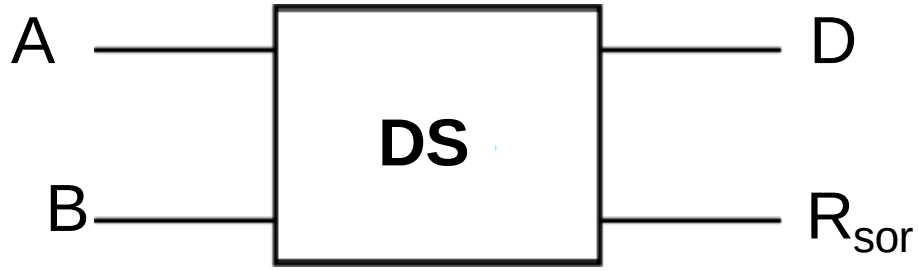
C'est un circuit qui fait la soustraction de deux bits A et B de même poids, il ne tient pas compte d'un éventuel report provenant des bits de poids inférieurs. La table de vérité de ce circuit est la suivante :

A	B	D	R_{sor}
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Soustracteurs

– **Demi soustracteur** :

Le symbole logique d'un demi soustracteurest comme suit :



Symbole logique d'un DS

Soustracteurs

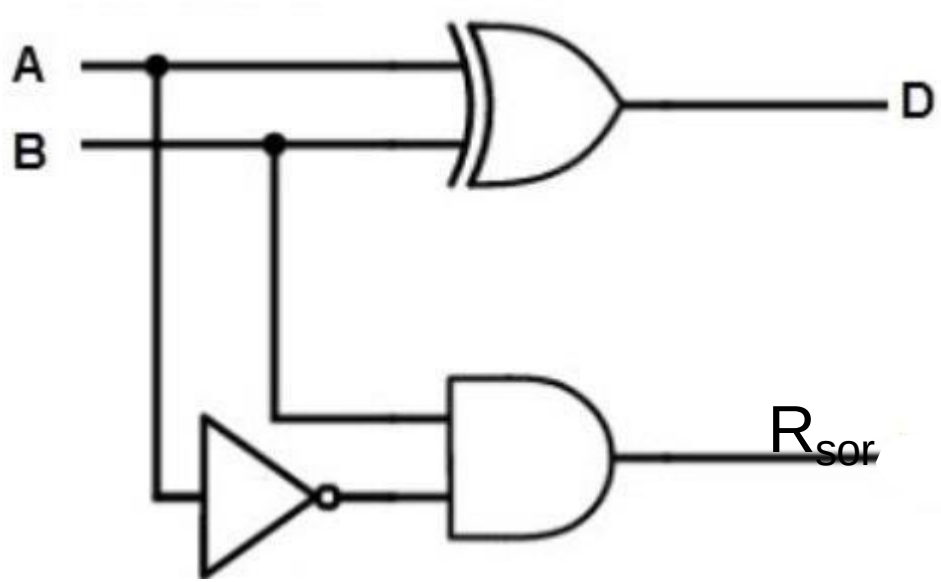
– **Demi soustracteur** :

Expressions logiques :

$$D = A \oplus B$$

$$R_{sor} = \bar{A}.B$$

Logigramme :



Soustracteurs

– Soustracteur complet :

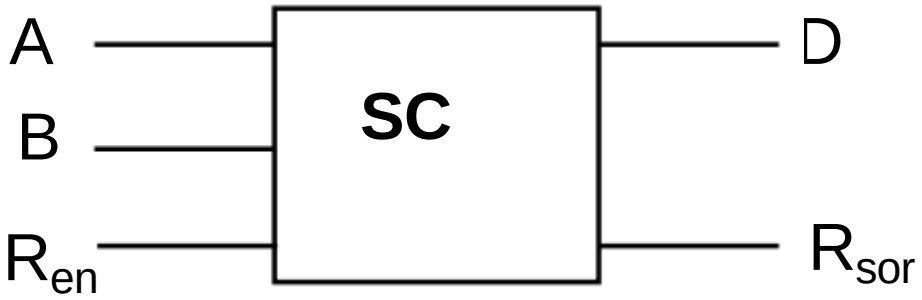
C'est un circuit qui fait la soustraction de deux bits A et B de même poids plus le report de l'étape précédente R_{en}

A	B	R_{en}	D	R_{sor}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Soustracteurs

– Soustracteur complet :

Le symbole logique d'un demi soustracteur est comme suit :



Symbole logique d'un SC

Soustracteurs

– **Soustracteur complet** :

Expression logique :

$$D = \bar{A}.\bar{B}.R_{en} + \bar{A}.B.\overline{R_{en}} + A.\bar{B}.\overline{R_{en}} + A.B.R_{en}$$

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

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

$$= R_{en}.(\overline{A \oplus B}) + \overline{R_{en}}.(A \oplus B)$$

$$D = R_{en} \oplus (A \oplus B)$$

Soustracteurs

– Soustracteur complet :

Expression logique :

$$R_{sor} = \bar{A}.\bar{B}.R_{en} + \bar{A}.B.\overline{R_{en}} + \bar{A}.B.R_{en} + A.B.R_{en}$$

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

$$R_{sor} = R_{en}.(\overline{A \oplus B}) + \bar{A}.B$$

Soustracteurs

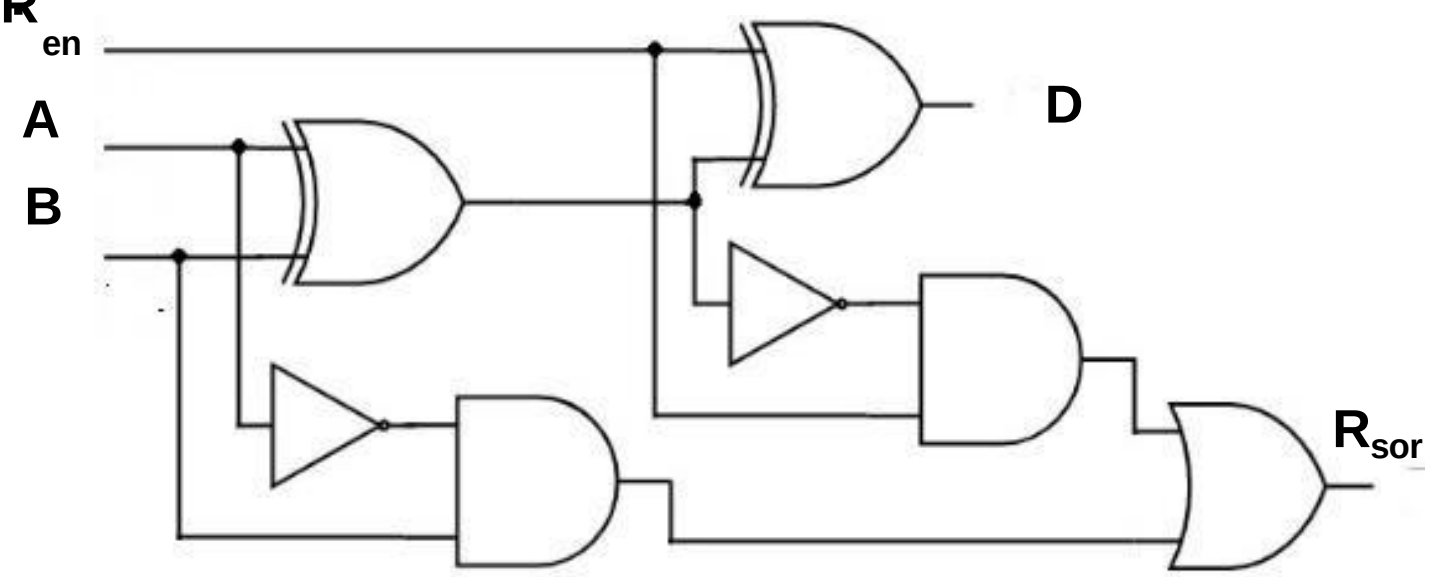
– Soustracteur complet :

Expressions logiques

$$D = R_{en} \oplus (A \oplus B)$$

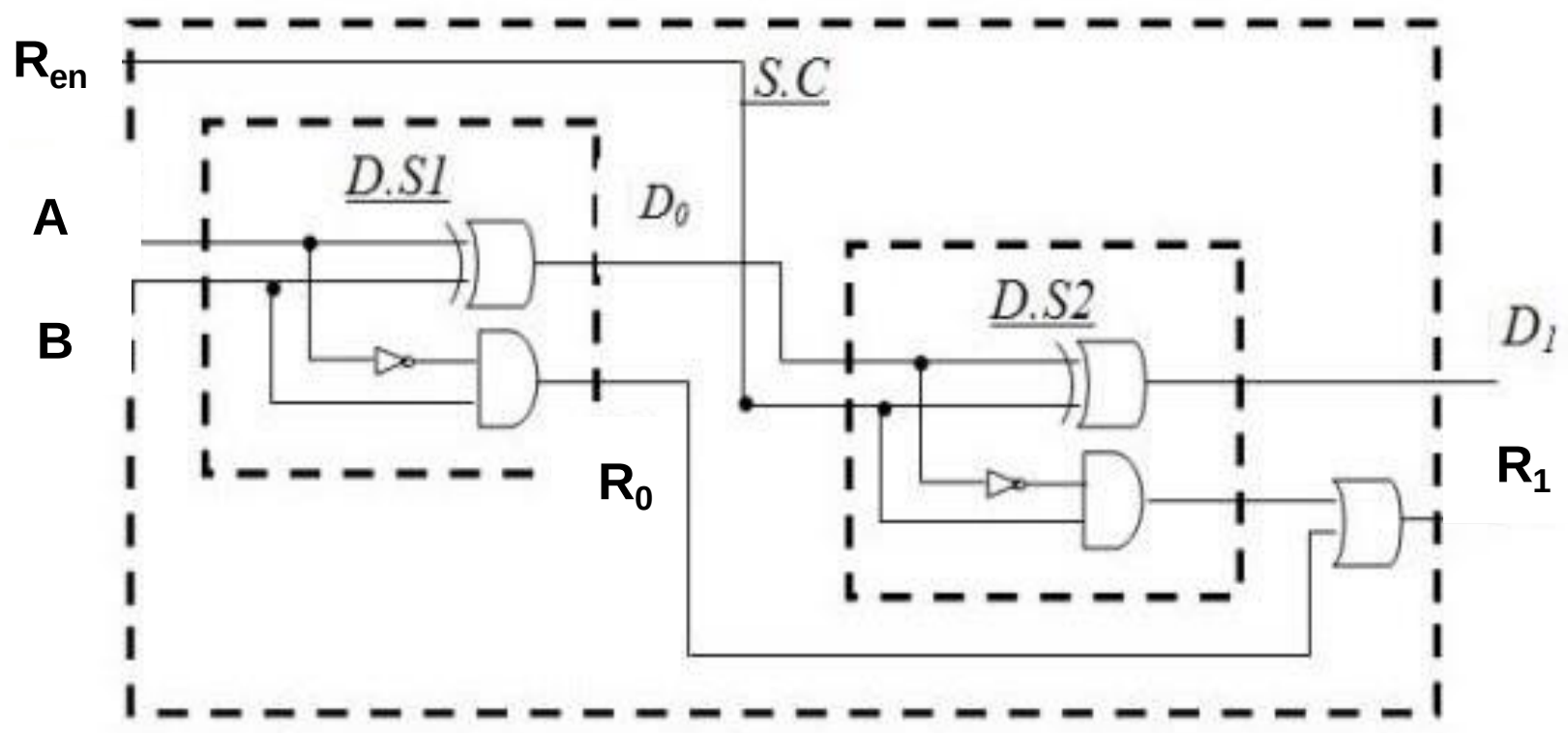
$$R_{sor} = R_{en}.(\overline{A \oplus B}) + \bar{A}.B$$

Logigramme :



Soustracteurs

– **Réalisation d'un soustracteur complet à l'aide de deux demi-soustracteurs :**



Logigramme d'un Soustracteur Complet

Soustracteurs

– **Soustracteur sur n bits** :

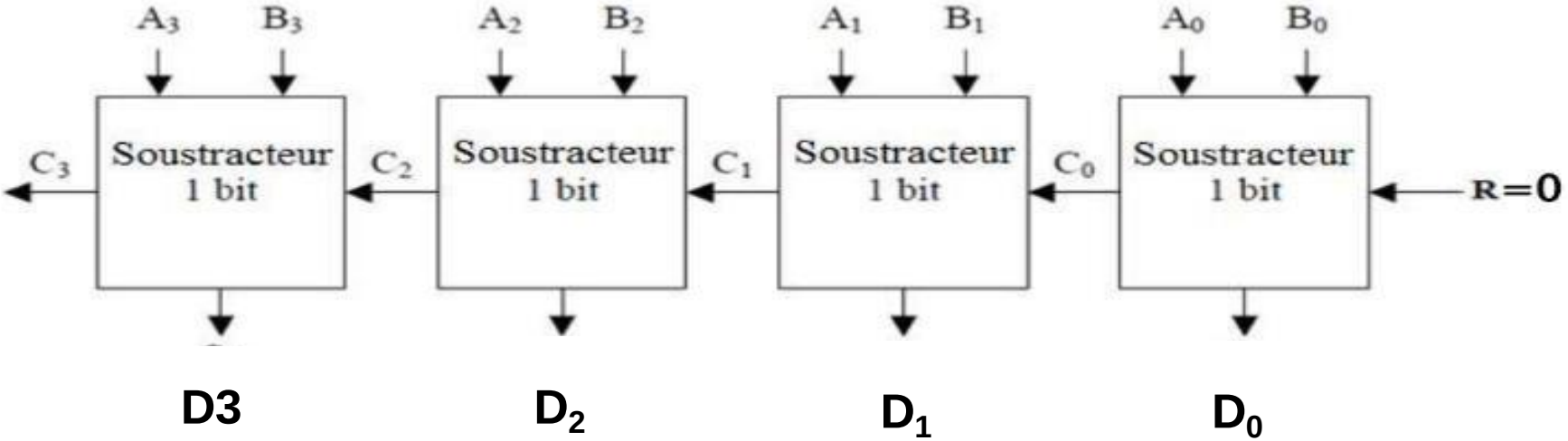
Sachant qu'un soustracteur complet ne peut traiter que deux nombres de 1 bit et une retenue d'entrée ; pour soustraire des nombres de plus d'un bit, il faut utiliser des soustracteurs complets supplémentaires.

Un soustracteur parallèle à n bits est le branchement en cascade de n soustracteur complets ; où la sortie de retenue de chaque soustracteur est connectée à l'entrée de retenue du soustracteur du bit de rang plus élevé suivant.

Soustracteurs

– **Soustracteur sur n bits** :

Exemple : soustracteurs parallèle à 4 bits. Soit à soustraire les nombres binaires : A= A_3 A_2 A_1 A_0 et B = B_3 B_2 B_1 B_0



Remarque : le premier soustracteur à droite peur être remplacé par un demi-soustracteur.

Comparateurs

- Un comparateur logique est un circuit logique qui effectue la comparaison entre deux nombres binaires généralement notés A et B.
- IL possède 3 sorties possibles notées :
  - fe : égalité (**A = B** quand A est égal à B)
  - fs : supérieur (**A>B** quand A est strictement supérieur au nombre B)
  - fi : inférieur (**A<B** quand A est strictement inférieur au nombre B)

Comparateurs

- **Comparateur de deux nombres à 1 bit:**

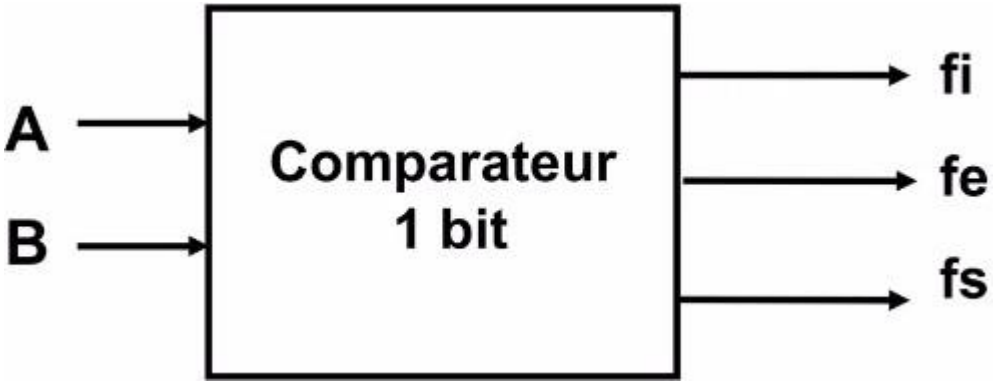
C'est la forme la plus simple des comparateurs.

- Il possède deux entrées (deux nombres sur 1 bit) :
  - A sur 1 bit
  - B sur 1 bit
- Et il possède trois sorties :
  - fe : égalité (**A = B**)
  - fs : supérieur (**A>B**)
  - fi : inférieur (**A<B**)

Comparateurs

- **Comparateur de deux nombres à 1 bit:**

Le symbole logique d'un comparateur est le suivant :



Symbole logique d'un comparateur à un bit

Comparateurs

- **Comparateur de deux nombres à 1 bit:**

- Si A = B, la sortie A = B passe à l'état **1** tandis que les sorties A > B et A < B passent à l'état **0**.
- Si le nombre A est strictement supérieur au nombre B, alors la sortie A> B passe à l'état **1** tandis que les sorties A = B et A < B passent à l'état **0**.
- Si le nombre A est strictement inférieur au nombre B, seule la sortie A < B passe à l'état **1**.

Comparateurs

- **Comparateur de deux nombres à 1 bit:**

Table de vérité :

A	B	fs	fe	fi
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Comparateurs

Comparateur de deux nombres à 1 bit :

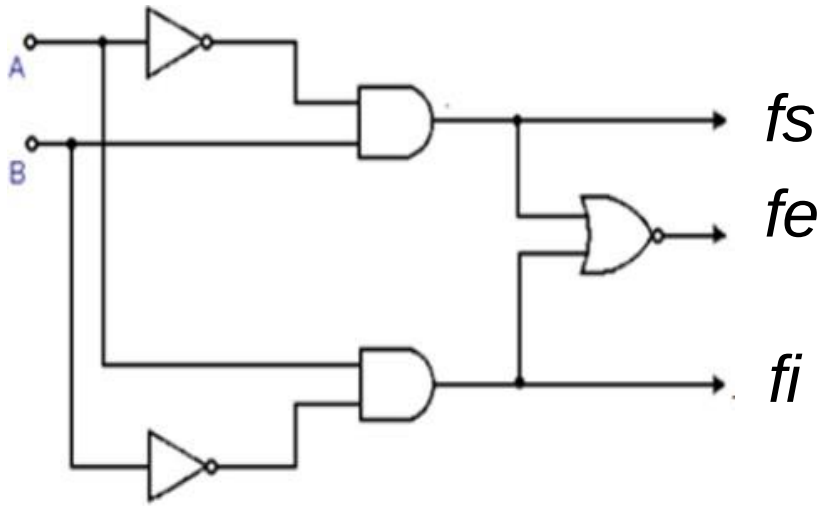
- Les expressions logiques :

$$fi = \bar{A}.B$$

$$fe = \bar{A}.\bar{B} + A.B = \overline{A \oplus B}$$

$$fs = A.\bar{B}$$

- Logigramme :



Comparateurs

- **Comparateur de deux nombres à 2 bit:**

Il permet la comparaison entre deux chiffres binaires $A(a_1a_0)$ et $B(b_1b_0)$ chacun sur 2 bits.

Symbole logique d'un comparateur à deux bit

Comparateurs

- **Comparateur de deux nombres à 2 bit:**

Le fonctionnement du comparateur à 2 bits peut être déduit de celui à 1 seul bit. Pour comparer deux nombres à 2 bits, il faut comparer les bits de même rang :

$A > B$ si $(a_1 > b_1)$ ou $(a_1 = b_1$ et $a_0 > b_0)$

$A = B$ si $(a_1 = b_1)$ et $(a_0 = b_0)$

$A < B$ si $(a_1 < b_1)$ ou $(a_1 = b_1$ et $a_0 < b_0)$

- Remarque: En général, pour comparer deux nombres à n bits, il faut utiliser n comparateurs à 1 seul bit.

a_1	a_0	b_1	b_0	fs	fe	fi
0	0	0	0	0	1	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	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

Comparateurs

- Comparateur de deux nombres à 2 bit:

Expression logiques de sorties :

$$fs = \overline{a_1}.a_0.\overline{b_1}.\overline{b_0} + a_1.\overline{a_0}.\overline{b_1}.\overline{b_0} + a_1.\overline{a_0}.b_1.b_0 + a_1.a_0.\overline{b_1}.\overline{b_0} + a_1.a_0.\overline{b_1}.b_0 + a_1.a_0.b_1.\overline{b_0}$$

$$= a_0.\overline{b_0}.(\overline{a_1}.\overline{b_1} + a_1.b_1) + a_1.\overline{a_0}.\overline{b_1}.(\overline{b_0} + b_0) + a_1.a_0.\overline{b_1}.(\overline{b_0} + b_0)$$

$$= a_0.\overline{b_0}.(\overline{a_1 \oplus b_1}) + a_1.\overline{b_1}.(\overline{a_0} + a_0)$$

$$fs = a_0.\overline{b_0}.(\overline{a_1 \oplus b_1}) + a_1.\overline{b_1}$$

Comparateurs

- **Comparateur de deux nombres à 2 bit:**

Expression logiques de sorties :

$$fe = \overline{a_1}.\overline{a_0}.\overline{b_1}.\overline{b_0} + \overline{a_1}.a_0.\overline{b_1}.b_0 + a_1.\overline{a_0}.b_1.\overline{b_0} + a_1.a_0.b_1.b_0$$

$$= \overline{a_1}.\overline{b_1}.(\overline{a_0}.\overline{b_0} + a_0.b_0) + a_1.b_1.(\overline{a_0}.\overline{b_0} + a_0.b_0)$$

$$= \overline{a_1}.\overline{b_1}.(\overline{a_0 \oplus b_0}) + a_1.b_1.(\overline{a_0 \oplus b_0})$$

$$= (\overline{a_0 \oplus b_0}).(\overline{a_1}.\overline{b_1} + a_1.b_1)$$

$$fe = (\overline{a_0 \oplus b_0}).(\overline{a_1 \oplus b_1})$$

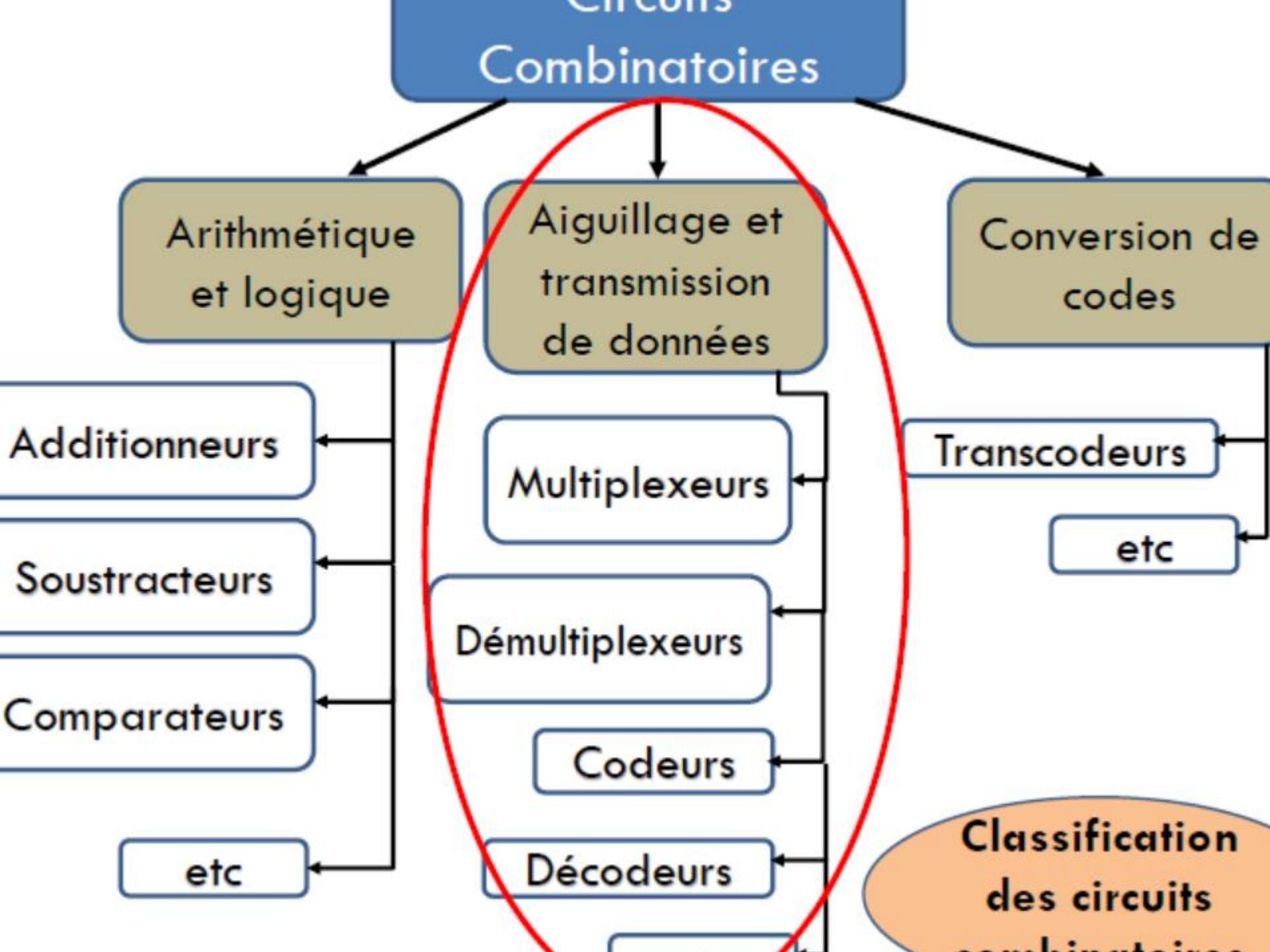
Comparateurs

- **Comparateur de deux nombres à 2 bit:**

Expression logiques de sorties :

$$fi = \overline{a_1}.\overline{a_0}.\overline{b_1}.b_0 + \overline{a_1}.\overline{a_0}.b_1.\overline{b_0} + \overline{a_1}.\overline{a_0}.b_1.b_0 + \overline{a_1}.a_0.b_1.\overline{b_0}$$
$$\overline{a_1}.a_0.b_1.b_0 + a_1.\overline{a_0}.b_1.b_0$$
$$= \overline{a_1}.\overline{a_0}.\overline{b_1}.b_0 + \overline{a_1}.\overline{a_0}.b_1.(\overline{b_0} + b_0) + \overline{a_1}.a_0.b_1.(\overline{b_0} + b_0) +$$
$$a_1.\overline{a_0}.b_1.\overline{b_0}$$
$$= \overline{a_0}.b_0(\overline{a_1}.\overline{b_1}. + a_1.b_1) + \overline{a_1}.\overline{a_0}.b_1 + \overline{a_1}.a_0.b_1$$
$$= \overline{a_0}.b_0.(\overline{a_1 \oplus b_1}) + \overline{a_1}.b_1.(\overline{a_0} + a_0)$$

$$fi = \overline{a_0}.b_0.(\overline{a_1 \oplus b_1}) + \overline{a_1}.b_1$$


Circuits Combinatoires
Arithmétique et logique
Aiguillage et transmission de données
Conversion de codes
Additionneurs
Soustracteurs
Comparateurs
etc
Multiplexeurs
Démultiplexeurs
Codeurs
Décodeurs
Transcodeurs
etc
Classification des circuits

Circuits d'aiguillage et transmission de données

- C'est un groupe de circuits permettant d'aiguiller les informations (données) binaires à travers des lignes électriques (souvent appelé BUS) d'une source (une petite mémoire appelée registre ou des capteurs, interrupteurs ou boutons poussoirs) vers une destination (registre ou un afficheur par exemple).

Circuits d'aiguillage et transmission de données

- Nous allons détailler les circuits d'aiguillage et transmission suivants :
  - Codeurs.
  - Décodeurs
  - Multiplexeurs.
  - démultiplexeurs.

Codeurs (Encodeur)

- Le codeur (ou encodeur) possède **2^N entrées**, dont une seule est activée et **N sorties**.
- Le principe de fonctionnementd'un codeur est le suivant : lorsqu'une entrée est activée, les sorties affichent le code binaire équivalent au numéro de l'entrée activée.

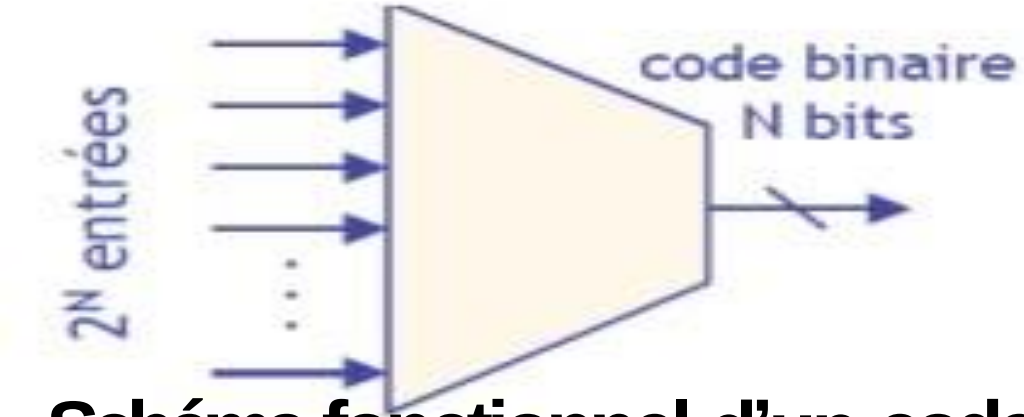
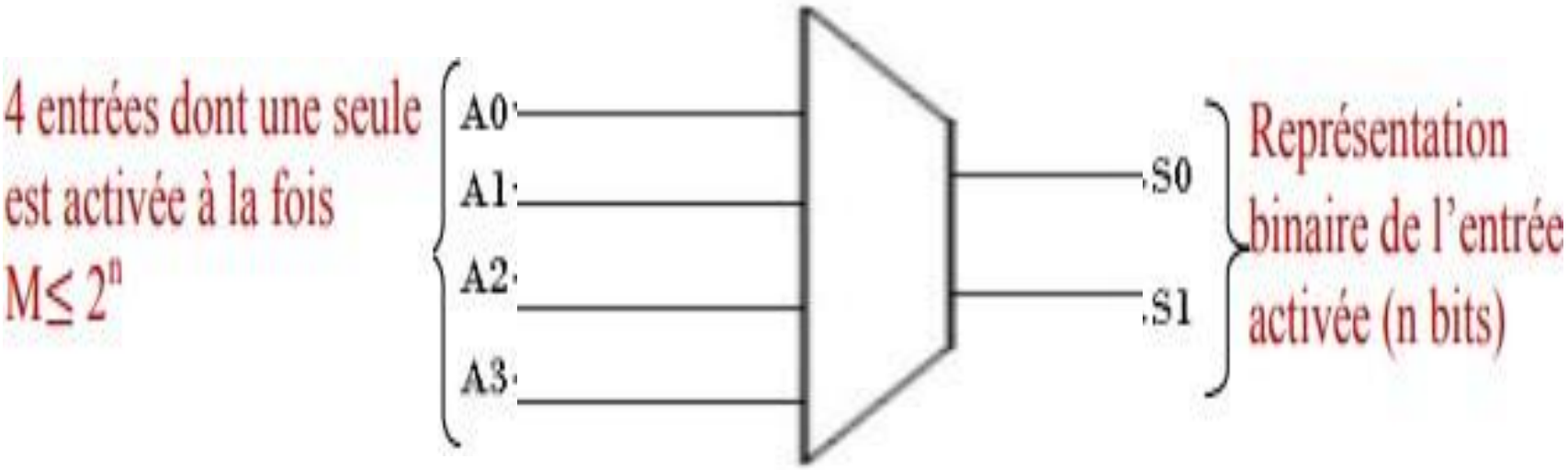


Schéma fonctionnel d'un codeur

Codeurs (Encodeur)

Exemple d'un codeur 4 voies d'entrées et 2 bits de sortie :

- **Schéma fonctionnel :**



Codeurs (Encodeur)

Exemple d'un codeur 4 voies d'entrées et 2 bits de sortie :

- **Table de vérité :**

ENTREES				SORTIE	
Codage 1 parmi 2^n				Nombre binaire de n bits	
A_3	A_2	A_1	A_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

Codeurs (Encodeur)

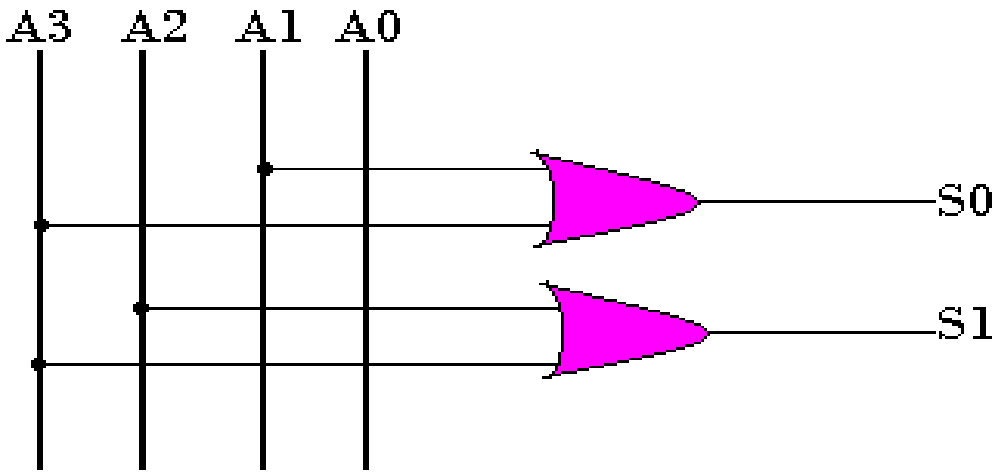
Exemple d'un codeur 4 voies d'entrées et 2 bits de sortie :

- **Equations logiques de sortie :**

$S_1 = 1$ si $(A_2 = 1)$ ou $(A_3 = 1)$; $\mathbf{S_1 = A_2 + A_3}$

$S_0 = 1$ si $(A_1 = 1)$ ou $(A_3 = 1)$; $\mathbf{S_0 = A_1 + A_3}$

- **Logigramme :**



Codeurs (Encodeur)

- Par contre, si plusieurs entrées sont actives simultanément la sortie peut prendre une valeur mal définie (sans signification). Par exemple, si les deux lignes A_1 et A_2 sont dans l'état 1 (frappe simultanée des deux touches) .
- Pour éviter ce problème on utilise un **encodeur prioritaire.**

Codeurs de priorité

- Pour un codeur (encodeur) **prioritaire** si plusieurs lignes d'entrées sont activés simultanément il génère un code binaire correspondant à une **seule** parmi celle-ci.

- La règle est de mettre en sortie le code correspondant à la ligne d'entrée d'indice le plus **élevé**.

Codeurs de priorité

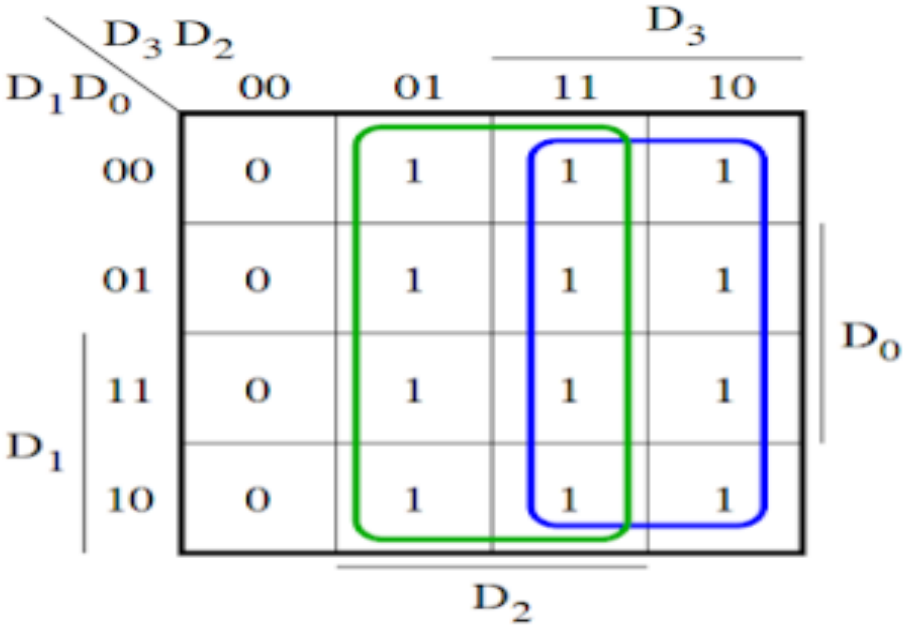
- **Codeur prioritaire 4 : 2**
- La table de vérité :

D_3	D_2	D_1	D_0	Y_1	Y_0	V
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

où x représente un état sans importance. Pour un mot d'entrée, le bit actif avec le poids le plus élevé est prioritaire

Codeurs de priorité

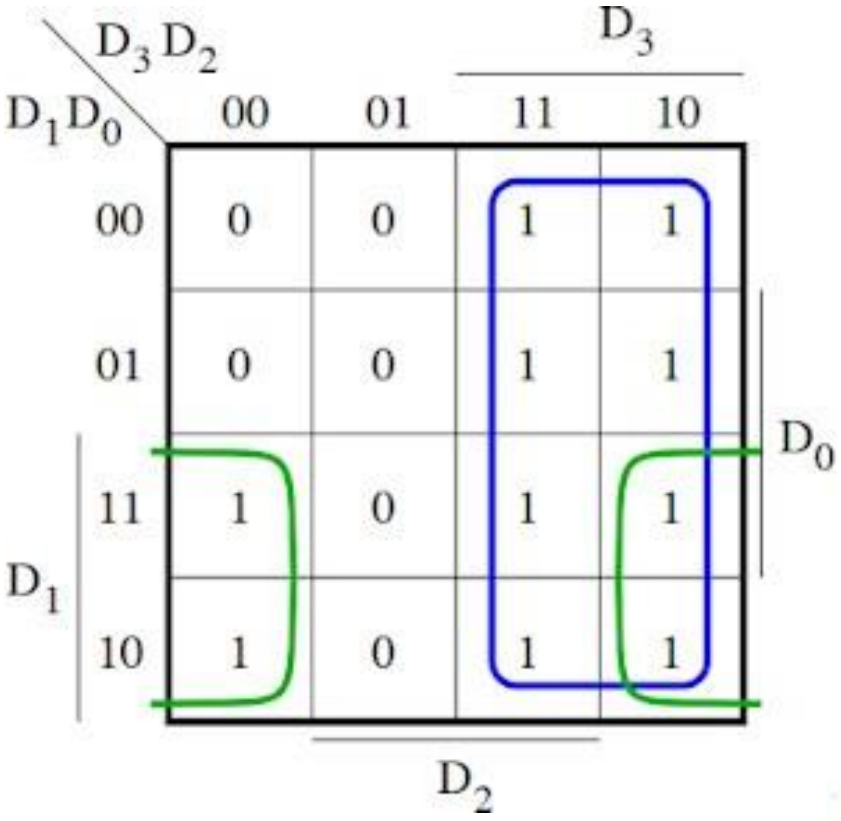
- **Codeur prioritaire 4 : 2**
- Les tables de Karnaugh sont construites en supposant que chaque état indifférent peut prendre le niveau logique 0 ou le niveau logique 1.



$Y_1 = D_3 + D_2$

Codeurs de priorité

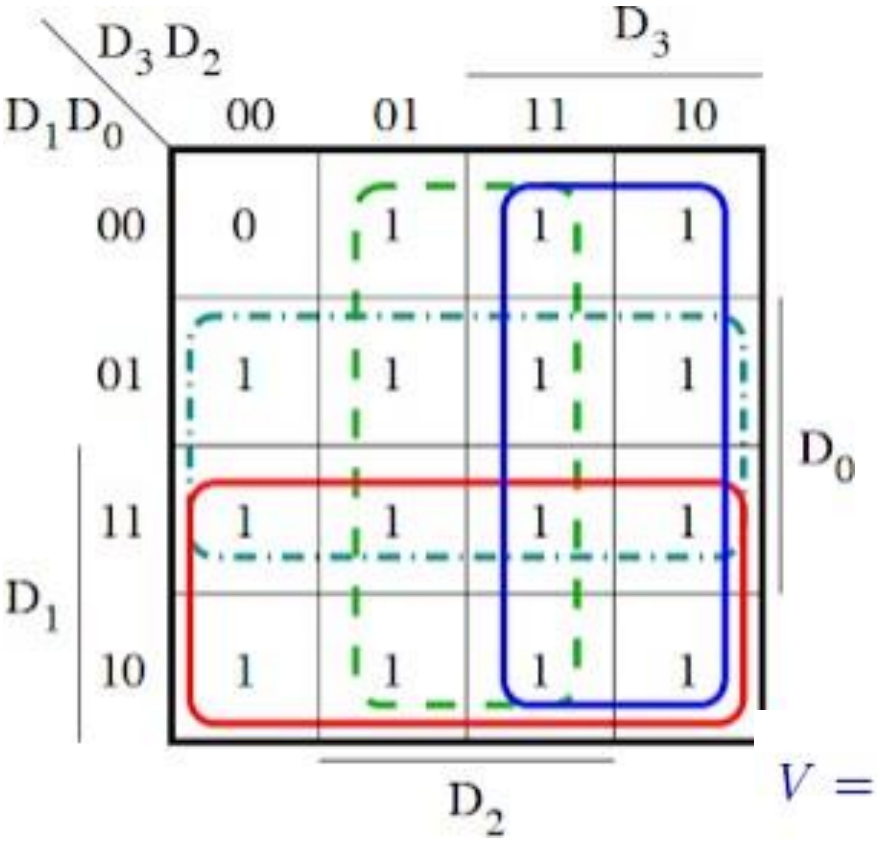
- **Codeur prioritaire 4 : 2**



$$Y_0 = D_3 + \overline{D_2} D_1$$

Codeurs de priorité

- **Codeur prioritaire 4 : 2**



$$V = D_3 + D_2 + D_1 + D_0$$

Codeurs de priorité

- **Codeur prioritaire 4 : 2**

Les équations logiques résultantes peuvent être écrites comme suit:

$$Y_1 = D_3 + D_2$$

$$Y_0 = D_3 + \overline{D_2} \cdot D_1$$

$$V = D_3 + D_2 + D_1 + D_0$$

Codeurs de priorité

- Codeur prioritaire 4 : 2
- Le logigramme