Codification et Représentation de l'Information (CRI)

MI – USTHB – SEC4

Par Dr L.ABADA

abada.lyes@gmail.com

références : CRI - N.HADJI

- **Chapitre 1 : Codification et représentation des nombres,**
- **Chapitre 2 : Algèbre de Boole,**
- **Chapitre 3 : circuits combinatoires :**
  - L’Additionneur,
  - Le Décodeur,
  - Le Multiplexeur
  - …….

Introduction

- Notre langage écrit utilise un code basé sur 26 lettres (majuscules et minuscules), 10 chiffres, des symboles de ponctuation et des signes mathématiques.
- Grâce à ce code et à ces règles nous pouvons transmettre des informations, donner des instructions, dénombrer...
- Bien que les ordinateurs soient qualifiés "d'intelligence artificielle", ils n'ont aucune faculté d'appréhender le monde extérieur.

Introduction

- Bien que les ordinateurs soient qualifiés "d'intelligence artificielle", ils n'ont aucune faculté d'appréhender le monde extérieur.
- Leur seule intelligence réside dans leur rapidité d'exécution de combinaisons d'ordre à deux états (0 , 1)équivalent à (éteint , allumé).
- Le terme bit signifie « binary digit ». Il s'agit de la plus petite unité d'information manipulable par une machine numérique.
- Il est possible de représenter physiquement cette information binaire par un signal électrique ou magnétique

Introduction

- La **codification** consiste à établir une correspondance entre la représentation externe de l'information dont nous sommes utilisateurs et sa représentation interne dans la machine, qui est une suite de bits (suite de 0 et 1)

Systèmes de numération

- Un système de numération est défini par un ensemble de **symboles (chiffres ou lettres)** et des **règles d'écriture** pour le **positionnement** de ces symboles.
- L'exemple le plus répandu de système de numération est la **numération décimale**.
- Ce système est composé de dix chiffres : **0, 1, 2, 3, 4, 5, 6, 7, 8, 9**
- Un nombre est représenté par une succession de chiffres.
- Chaque chiffre possède **un poids**.

Systèmes de numération

$2_3\,3_2\,4_1\,1_0 = 2*10^3+3*10^2+4*10^1+1*10^0$

$abcd = a*10^3+b*10^2+c*10^1+d*10^0$

Base 10

0
1
2
3
4
5
6
7
8
9
10

Base	Base16	Base12	Base 10	Base 9	Base8	Base 5	Base 3	Base 2
	0	0	00	00	0	0	0	00
	1	1	01	01	1	1	1	01
	2	2	02	02	2	2	2	10
	3	3	03	03	3	3	10	11
	4	4	04	04	4	4	11	100
	5	5	05	05	5	10	12	101
	6	6	06	06	6	11	20	110
	7	7	07	07	7	12	21	111
	8	8	08	08	10	13	22	1000
	9	9	09	10	11	14	100	1001
	A	A	10	11	12	20	101	1010
	B	B	11	12	13	21	102	1011
	C	10	12	13	14	22	110	1100
	D	11	13	14	15	23	111	1101
	E	12	14	15	16	24	112	1110
	F	13	15	16	17	31	120	1111
	10	14	16	17	20		121	10000
	11	15	17	18	21		122	10001
	12	16	18	20	22		200	10010
	13	17	19	21	23		201	
		18	20	22	24	44		
		19	***21***	***23***	25	100		
	19	1A	22	24	26			
	1A	1B	23					
	1B	20	24					
	1C			88	77			
			99	100				
	21		100					
	1F							

Base 10	Base 8
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	10
9	11
10	12
11	13
12	14
13	15
14	16
15	17
16	20
17	21
18	22
19	23
20	***24***
21	25
22	26
23	27
24	30
25	31
26	32
27	33

$$(24)_8 = 2*8^1+4*8^0 = 16+4 = (20)_{10}$$

Base 10	Base 8
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	10
9	11
10	12
11	13
12	14
13	15
14	16
15	17
16	20
17	21
18	22
19	23
20	***24***
21	25
22	26
23	27
24	30
25	31
26	32
27	33

$$(24)_8 = 2*8^1+4*8^0 = 16+4 = (20)_{10}$$

$$(a_3a_2a_1a_0)_{10} = a_3*10^3+a_2*10^2+a_1*10^1+a_0*10^0$$

$$(a_3\,a_2\,a_1\,a_0)_8 = a_3*8^3+a_2*8^2+a_1*8^1+a_0*8^0$$

$$(a_3\,a_2\,a_1\,a_0)_{10} = a_3*10^3+a_2*10^2+a_1*10^1+a_0*10^0$$

$$(a_3\,a_2\,a_1\,a_0)_{8} = a_3*8^3+a_2*8^2+a_1*8^1+a_0*8^0$$

Bases

Un système de numération à base B est défini par B symboles (chiffres ou lettres)

Soit N un nombre de n chiffres représenté en base B

$$\mathbf{N = a_{n-1}a_{n-2}....a_i...a_0 \ i \quad a_i < B}$$

(Tous les symboles sont strictement inférieurs à B)

Quelque soit la base, la forme polynomiale de N est :

$$\mathbf{N = a_{n-1} * B^{n-1} + a_{n-2} * B^{n-2} \ldots . + a_i * B^i \ldots . + a_0 * B^0}$$

Système Binaire (Base 2)

→{0,1}

Système Octal (Base 8)

→{0,1,2,3,4,5,6,7}

Système Hexadécimal (Base 16)

→{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

Changement de base : Base B→Base 10:

il suffit de représenter le nombre sous sa forme polynomiale et calculer la somme de tous les termes

Bases

Un système de numération à base B est défini par B symboles (chiffres ou lettres)

Soit N un nombre de n chiffres représenté en base B

$$N = a_{n-1}a_{n-2}....a_i...a_0 \text{ i } a_i < B$$

(Tous les symboles sont strictement inférieurs à B)

Quelque soit la base, la forme polynomiale de N est :

$$N = a_{n-1} * B^{n-1} + a_{n-2} * B^{n-2} .... + a_i * B^i .... + a_0 * B^0$$

Changement de base : Base B→Base 10:

Exemple :

B = 2

$N= (1111011)_2 = 1*2^6 + 1*2^5 + 1*2^4 +1*2^3 + 0*2^2 + 1*2^1 +1*2^0 = (123)_{10}$

B = 16

$N = (7B)_{16} = (7*16^1 + 11*16^0)_{10} = 123$

Changement de base : Base B→Base 10:

Pour les nombres décimaux, on utilisera des exposants négatifs.

Exemple :

B = 16

$N = (7_1B_0,8_14_2)_{16} = 7*16^1 + 11*16^0 + 8*16^{-1} + 4*16^{-2} = (123,515625)_{10}$

Changement de base : Base 10 → Base B:

La règle à suivre est celle des divisions successives,

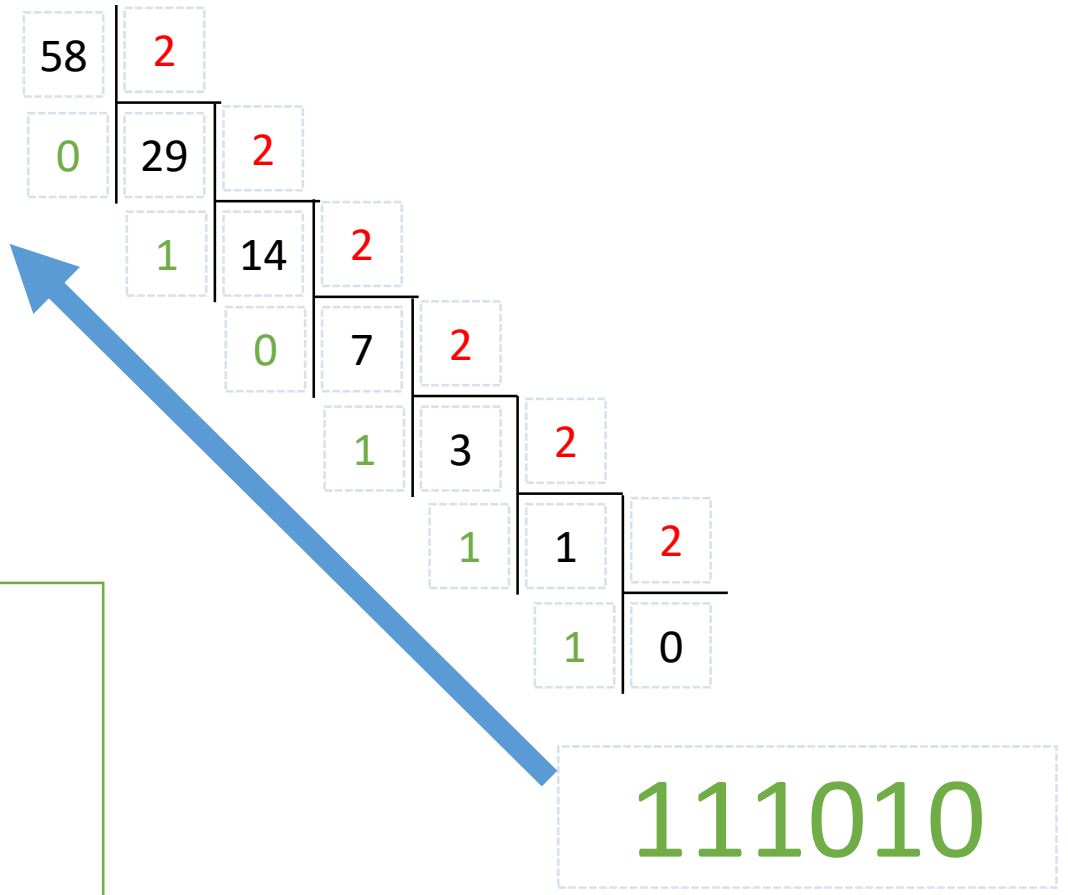
1. On divise le nombre par **B**
2. On sauvegarde **le reste** puis **on divise le quotient** par **B**
3. Ainsi de suite jusqu'à obtention d'un quotient nul

La suite des restes correspond au nombre de la base visée

Le premier reste correspond au **poids faible** et **le dernier** au **poids fort**

Changement de base : Base 10 → Base B

Exemple : binaire (base2) → $N=(58)_{10}$ → $(???)_2$



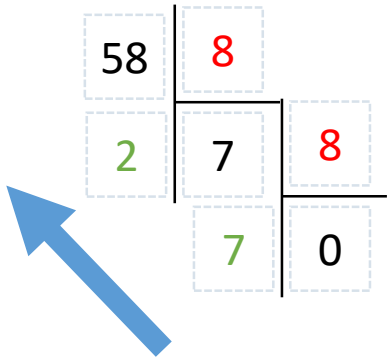
La règle à suivre est celle des divisions successives,

1. On divise le nombre par **B**
2. On sauvegarde **le reste** puis **on divise le quotient** par **B**
3. Ainsi de suite jusqu'à obtention d'un quotient nul

La suite des restes correspond au nombre de la base visée
Le premier reste correspond au **poids faible** et **le dernier** au **poids fort**

Changement de base : Base 10 → Base B:

Exemple : octale (base8) → $N=(58)_{10}$ → $(???)_8$



72

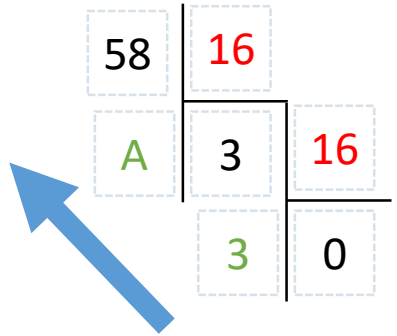
La règle à suivre est celle des divisions successives,

1. On divise le nombre par **B**
2. On sauvegarde **le reste** puis **on divise le quotient** par **B**
3. Ainsi de suite jusqu'à obtention d'un quotient nul

La suite des restes correspond au nombre de la base visée
Le premier reste correspond au **poids faible** et **le dernier** au **poids fort**

Changement de base : Base 10 → Base B:

Exemple : hexadécimale (base16) → $N=(58)_{10}$ → $(???)_{16}$



3A

La règle à suivre est celle des divisions successives,

1. On divise le nombre par **B**
2. On sauvegarde **le reste** puis **on divise le quotient** par **B**
3. Ainsi de suite jusqu'à obtention d'un quotient nul

La suite des restes correspond au nombre de la base visée
Le premier reste correspond au **poids faible** et **le dernier** au **poids fort**

Changement de base : Base B1 → Base B2:

Pour passer d'une base B1 vers une base B2 il faut 2 opérations.

1. Il faut d'abord passer de la base B1 vers la base 10
2. puis de la base 10 vers la base B2

Changement de base : Base B1 → Base B2:

Exemple : Convertir (3141)5 vers la base 16

$(3141)_5 = (421)_{10}$

$(421)_{10} = (1A5)_{16}$

→ $(3141)_5 = (1A5)_{16}$

Pour passer d'une base B1 vers une base B2 il faut 2 opérations.

1. Il faut d'abord passer de la base B1 vers la base 10
2. puis de la base 10 vers la base B2

Changement de base :
Applications aux bases 2, 8 et 16

2➔ 8

Pour convertir un nombre de la base 2 vers la base 8, il faut découper ce nombre en groupes de 3 bits et remplacer chaque groupe par sa valeur octale (en partant de la droite).

Exemple : Convertir $(1\ 011\ 010)_2$ vers la base 8

$(001\quad 011\quad 010)_2 = (1\ 3\ 2)_8$

Base 2	Base 8
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Changement de base :
Applications aux bases 2, 8 et 16

2 ← 8

Pour convertir un nombre de la base 8 vers la base 2, il suffit de transcrire chaque chiffre de ce nombre en binaire sur 3 bits (en partant du poids faible).

Exemple : Convertir $(645)_8$ vers la base 2

$(420)_8 = (\ 100\ \ 010\ \ 000)_2$

Base 2	Base 8
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Changement de base : Applications aux bases 2, 8 et 16

2 ➔ 16

Pour convertir un nombre de la base 2 vers la base 16, il faut découper le nombre en groupes de 4 bits et remplacer chaque groupe par sa valeur hexadécimale (en partant de la droite).

Exemple :
Convertir $(101\ \ 1010)_2$ vers la base 16
$(0101\ \ 1010)_2 = (5\ A)_{16}$

Base 2	Base 16
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Changement de base : Applications aux bases 2, 8 et 16

2 ← 16

Pour convertir un nombre de la base 16 vers la base 2, il suffit de transcrire chaque chiffre de ce nombre en binaire sur 4 bits en partant du poids faible.

Exemple : Convertir $(A15)_{16}$ vers la base 2

$(A15)_{16} = (1010\ \ 0001\ 0101)_2$

Base 2	Base 16
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Changement de base : Applications aux bases 2, 8 et 16

8 → 16

Pour convertir un nombre de la base 8 vers la base 16 ou inversement, on peut transiter par la base 10 mais on peut également passer par la base 2

Base8 → base2 → base16
Base16 → base2 → base8

Exemple : Convertir $(232)_8$ vers la base 16
$(232)_8 = (010\ \ 011\ \ 010)_2 = (\ 0000\ \ 1001\ \ 1010)_2 = (0\ 9\ A)_{16}$

Base 2	Base 16
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Changement de base : Applications aux bases 2, 8 et 16

des nombres décimaux

Pour la conversion des nombres décimaux, on sépare la partie entière de la partie décimale.
La partie entière est traitée comme indiqué précédemment.
La conversion de la partie décimale se fait de droite à gauche.

$(011101,010\ 110)_2 = (\ 011\ \ 101,\ 010\ 110\ )_2 = (35,26)_8$

$(0001\ 1101\ \ ,0101\ \ 1000)_2 = (\ 0001\ \ 1101,\ 0101\ 1000\ )_2 = (1D,58)_{16}$

Base 2	Base 16

Arithmétique binaire

1

(Exemples d'opérations réalisées en binaire)

Addition

1 1 1 1		
1 1 1 0 1		29
+ 1 1 1		+ 7
----------		-----
1 0 0 1 0 0		36

0+0 = 0

0+1 = 1

1+0 = 1

1+1 = 10 = 2_{10}

1+1+1 = 11 = 3_{10}

Arithmétique binaire

(Exemples d'opérations réalisées en binaire)

1

1

Soustraction

	1	1	1	1	1 0 1		27
-			1		1 1		- 9
		1	1				
	1 0		1		1 0		22

0-0 = 0

0-1 =

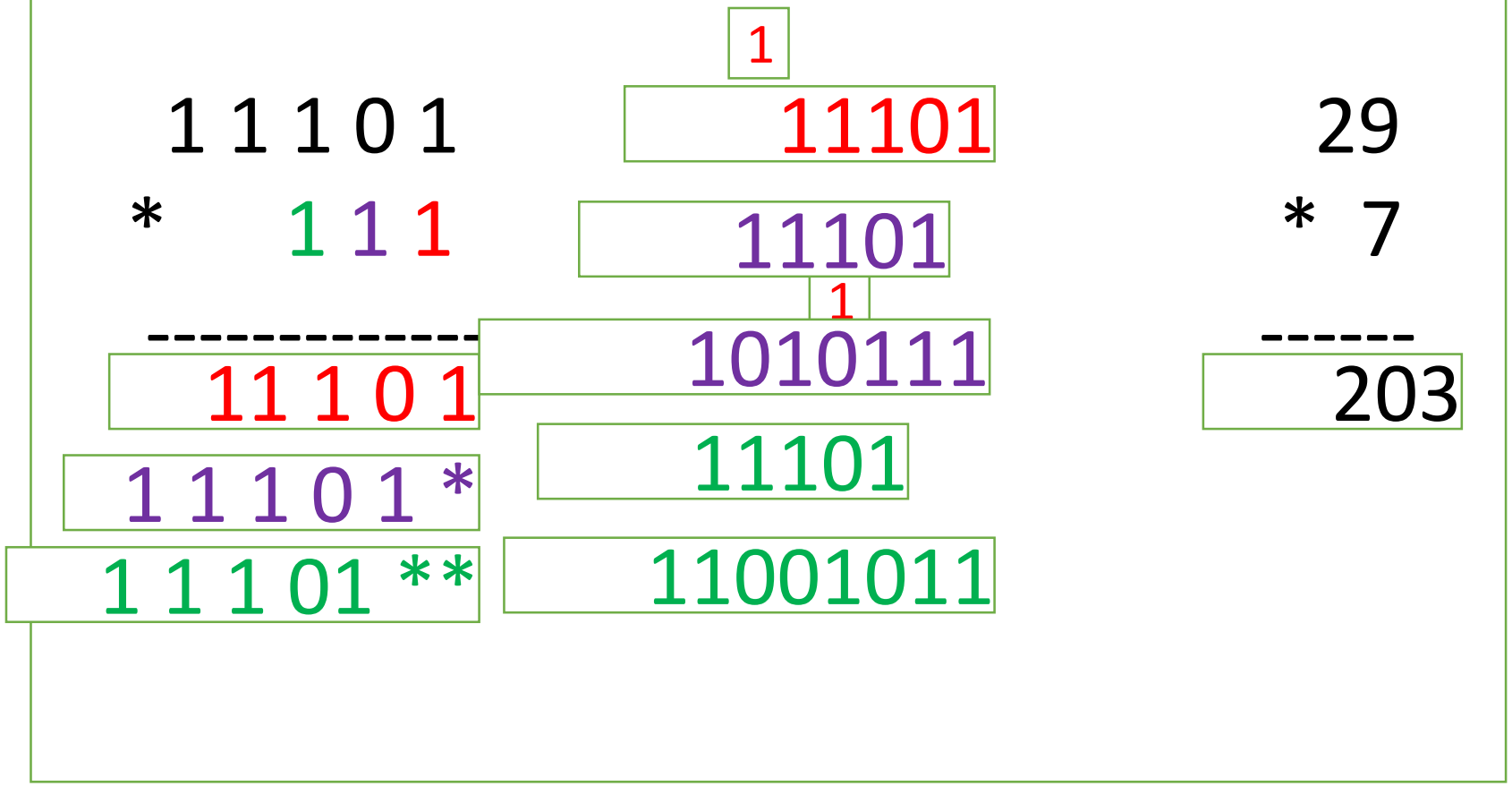
1-0 = 1

1-1 = 0

Arithmétique binaire

(Exemples d'opérations réalisées en binaire)

Multiplication



0*0 =
0*1 =
1*0 =
1*1 =

Arithmétique binaire

(Exemples d'opérations réalisées en binaire)

Division

1 1 1 0 1	1 1 1
0 0 0 0 1	1 0 0

2 9	7
1	4

0*0 =
0*1 =
1*0 =
1*1 =

Arithmétique binaire

(Exemples d'opérations réalisées en binaire)

Division

```
11000001 | 101
-------- | ---
001000   | 100110
  00110  |
    0011 |
```

```
193 | 5
--- | --
 43 | 38
  3 |
```

0*0 =
0*1 =
1*0 =
1*1 =

Représentation des nombres entiers relatifs

Les entiers relatifs sont représentés en binaire dans un format fixe (on ne peut pas comparer par exemple un nombre de 5 bits avec un nombre de 8 bits).

Le bit de poids fort représente le signe, il est égal à 0 si le nombre est positif et il est égal à 1 si le nombre est négatif

Représentation des nombres entiers relatifs

- **Représentation en signe et valeur absolue (SVA)**
- **Représentation en complément à 1 (C1)**
- **Représentation complément à 2 (C2)**
- ~~**Représentation d'un nombre en virgule flottante dans le format IEEE 754**~~

Représentation en Signe et valeur absolue (SVA)

Le **bit de poids fort** représente le signe du nombre (**0 pour +** et 1 pour –)
Les autres bits représentent la valeur absolue du nombre.
X représenté sur un format de n bits $-(2^{n-1}-1) < X < +(2^{n-1}-1)$

$000 = 7 = 2^n-1$

Exemples : Représentation sur **8 bits**

A = +25

$00011001_{(SVA)}$

$-(2^7-1) < X < +(2^7-1)$ -

127,127

A = -25

$10011001_{(SVA)}$

Représentation en signe et valeur absolue (SVA)

Le bit de poids fort représente le signe du nombre (0 pour + et 1 pour –)
Les autres bits représentent la valeur absolue du nombre.
X représenté sur un format de n bits $-(2^{n-1}-1) < X < +(2^{n-1}-1)$
000

Exemples : Représentation sur n=**8 bits**
B = + 525 = $10\ 0000\ 1101_{(2)}$
$525 > 2^{11}-1 = 127$

525 :

N'est pas possible,

Représentation en complément à 1 (C1)

Le complément à 1 d'un nombre est obtenu en inversant tous les bits.
Le bit de poids fort représente le signe du nombre (0 pour + et 1 pour –).
X représenté sur un format de n bits - ($2^{n-1} – 1$) < X < + ($2^{n-1} – 1$)

Exemple : Représentation sur 8 bits

A = 1 0 0 1 1 0 0 1
C1(A) = 0 1 1 0 0 1 1 0

A + C1(A) = 11111111+1 =0000000

Représentation complément à 2 (C2)

Le complément à 2 d'un nombre est égal au complément à 1 (C1) du nombre auquel on ajoute 1 : C(2)= C(1)+1,

X représenté sur un format de n bits - (2^{n-1}) < X < + ($2^{n-1} - 1$)

(-X)= C2 (X)

Exemples : représentation sur **8 bits**

A = 25 = $00011001_{(2)}$

CA1(25)= $11100110_{(ca1)}$

CA2(25) = 11100110 +1 = $11100111_{(ca2)}$

CA2(25) = -25 (sur 8bit)

Représentation complément à 2 (C2)

Exemple avec n=4:

E = $[-2^{4-1}, 2^{4-1}-1]$

E = [-8,7]

0= 0000 1111+1 = 0000
1= 0001 1110+1=1111
2= 0010 1101+1 = 1110
3= 0011 1100+1 = 1101
4= 0100
5= 0101
6= 0110
7= 0111 1000+1 = 1001
1111

1110➔ 0001+1
= 0010

-0= C2(0) = 0000
-1= C2(1) = 1111
-2= C2(2) = 1110
-3= C2(3) = 1101
-4= C2(4) = 1100
-5= C2(5) = 1011
-6= C2(6) = 1010
-7= C2(7)= 1001
-8= C2(8)= 1000

Représentation complément à 2 (C2)

Soit A et B 2 nombres représentés en C2 sur 8 bits, trouver leurs valeurs décimales.

Représenter en CA2 :

A = 53

$A = 00110101_{(2)} = 00110101_{(ca2)}$

A = -53

A = -53 = CA2(53) = CA2(00110101) = CA1(00110101)+1

$= 11001010+1 = 11001011_{(ca2)}$

Calculer le CA2 :

A = 13

CA2(A) = CA2(13) = CA1(00001101) +1 = 11110010+1 = 11110011 = -13

Opérations arithmétiques en complément à 2

Dans cette représentation la soustraction doit est traitée comme une addition. Pour les opérations arithmétiques on doit appliquer la règle suivante :

$$\forall (A,B) \in E \quad si \quad X = A + B \quad alors \quad X \in E$$

Exemples d'opérations arithmétiques avec n=4 (E = [- 8,+ 7]) :

Addition

```
    5
+   1
-------
```

```
  0101
+ 0001
-------
  0110
```

Dépassement ?

Pas de déplacement.

Opérations arithmétiques en complément à 2

Dans cette représentation la soustraction doit est traitée comme une addition. Pour les opérations arithmétiques on doit appliquer la règle suivante :

$\forall (A,B) \in E \quad si \quad X = A + B \quad alors \quad X \in E$

Exemples d'opérations arithmétiques avec n=4 (E = [- 8,+ 7]) :

Addition

```
   -7
+  -4
------
   -4
```

```
   1001
+  1100
-------------
  ~~1~~0101
```

Dépassement ?

Il y a un déplacement,
(+) + (+) = (-)

Opérations arithmétiques en complément à 2

Dans cette représentation la soustraction doit est traitée comme une addition. Pour les opérations arithmétiques on doit appliquer la règle suivante :

$\forall (A,B) \in E \quad si \quad X = A + B \quad alors \quad X \in E$

Exemples d'opérations arithmétiques avec n=4 (E = [- 8,+ 7]) :

Soustraction

```
   5
-  3
------
  +2
```

```
  0101
+ 1101
-------------
(+2) = 10010
```

Dépassement ?

Traitement du dépassement de capacité pour une addition:

- Si les deux opérandes sont de même signe et que le résultat est du même signe que les opérandes, il n'y a pas de dépassement

➢ 0001 + 0101 = 0110 ⇔ 1+5= 6

➢ 1111 + 1011 = ~~1~~1010 ⇔ -1-5=-6

- Si les deux opérandes sont de même signe et que le résultat est du signe opposé alors il y a dépassement.

➢ 0101+0100=1001 ⇔ ~~5+4 = -7~~

- Si les deux opérandes sont de signes opposés, il n'y a jamais de dépassement

➢ 1110+0111=~~1~~0101 ⇔ -2 + 7 = 5

Traitement du dépassement de capacité pour une addition:

Remarque :

- Il ne faut pas confondre les 2 expressions :

1. « Représenter X en format C2 » : revient à écrire X en C2
2. « Donner le C2 de X » reviens à écrire l'opposé de X

Ex : X = 35

Représenter 35 en CA2

$35 = 00100011_{(ca2)}$

Représenter -35 en CA2

-35 = CA2(35) = CA1(00100011)+1 = 11011101

Calculer/donner le CA2 de 35

CA2(35) = -35 = CA1(00100011)+1 = 11011101

Le code BCD

Le code BCD, Binary Coded Decimal (Décimal Codé en Binaire) est un code qui s'applique uniquement aux chiffres de la base 10. Chaque chiffre décimal est représenté directement par sa valeur binaire sur un format de 4 bits

Table des codes BCD

DEC	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Exemple :

$(987)_{10} = (1001 \ \ 1000 \ 0111)_{BCD}$

$(13)_{10} = (0001 \ \ 0011 \ \ )_{BCD}$

Addition BCD

Binaire	BCD
0 → 0000	0000 ← 0
1 → 0001	0001 ← 1
2 → 0010	0010 ← 2
3 → 0011	0011 ← 3
4 → 0100	0100 ← 4
5 → 0101	0101 ← 5
6 → 0110	0110 ← 6
7 → 0111	0111 ← 7
8 → 1000	1000 ← 8
9 → 1001	1001 ← 9
10→1010	0001 0000←10
11→1011	0001 0001←11
12→1100	0001 0010←12
13→1101	0001 0011←13
14→1110	0001 0100←14
15→1111	0001 0101←15
16→10000	1 0110←16

+

---

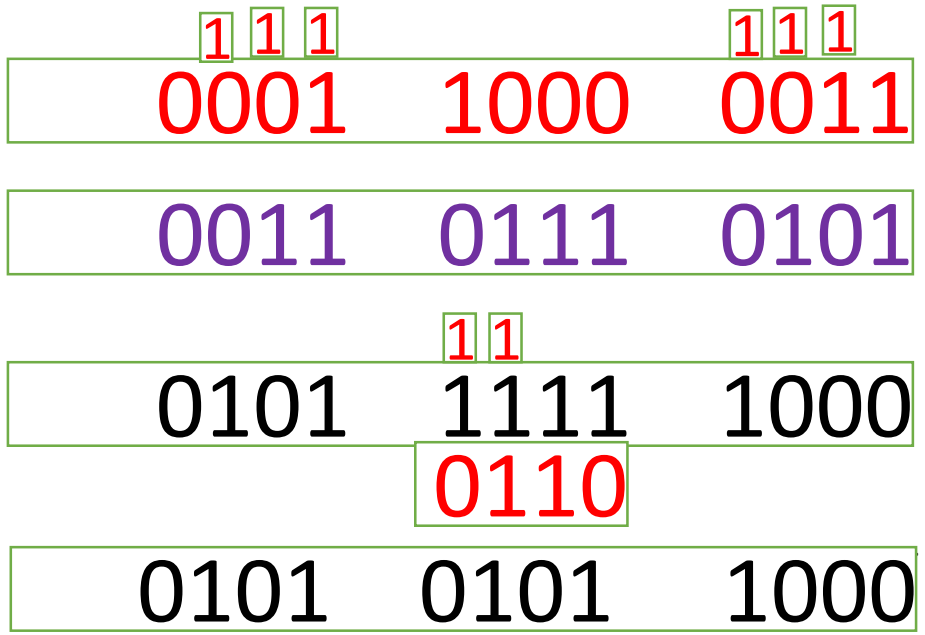
1 0011

Addition BCD

SI la somme de 2 chiffres codés en BCD est inférieure ou égale à 9 (1001) alors on ne fait rien sinon on ajoute 6 (0110) pour corriger et on retient 1.

Exemple : Réaliser l'addition A + B en BCD :

$A = 183_{10} = (\ 0001\ 1000\ 0011\ )_{BCD}$ $B = 375_{10} = (\ 0011\ 0111\ 0101\ )_{BCD}$



Addition BCD

1 0110

SI la somme de 2 chiffres codés en BCD est inférieure ou égale à 9 (1001) alors on ne fait rien sinon on ajoute 6 (0110) pour corriger et on retient 1.

Exemple 2

4 0000 0100

+ 3 0000 0011

= ----------

0000 0111

Exemple 3

4 0100

+ 9 1001

= ----------

1101

0110

0001 0011

Le code GRAY

Le code de Gray est un code binaire qui permet de passer d'un nombre entier N au nombre suivant (N+1) en changeant un seul digit(bit). (On l'appelle également Code Réfléchi). Ce code sera utilisé principalement dans les Tableaux de Karnaugh pour simplifier les fonctions booléennes.

0000	**0**		**11**		**22**
0001	**1**		**12**		**23**
0011	**2**		**13**		**24**
0010	**3**		**14**		**25**
0110	**4**		**15**		**26**
0111	**5**		**16**		**27**
0101	**6**		**17**		**28**
0100	**7**		**18**		**29**
1100	**8**		**19**		**30**
1101	**9**		**20**		**31**
1111	**10**		**21**		**32**

Conversion du code binaire vers le code Gray

Soit X = B_n B_{n-1}..... B_0 représenté en binaire Pour convertir X en code de Gray il faut suivre les règles suivantes :

$G_n = B_n$

$G_i = 0$ si $B_i = B_{i+1}$

$G_i = 1$ si $B_i <> B_{i+1}$

Exemple : pour n = 4 X = 10001 en binaire $X = (11001)_{gray}$

1 0 0 0 1

B4 B3 B2 B1 B0

1 1 0 0 1

G4 G3 G2 G1 G0

Le code GRAY

Le code de Gray est un code binaire qui permet de passer d'un nombre entier N au nombre suivant (N+1) en changeant un seul digit. (On l'appelle également Code Réfléchi). Ce code sera utilisé principalement dans les Tableaux de Karnaugh pour simplifier les fonctions booléennes.

0000	**0**
0001	**1**
0011	**2**
0010	**3**
0110	**4**
0111	**5**
0101	**6**
0100	**7**
1100	**8**
1101	**9**
1111	**10**

1110	**11**
1010	**12**
1011	**13**
1001	**14**
1000	**15**
11000	**16**
11001	**17**
11011	**18**
11010	**19**
11110	**20**
11111	**21**

11101	**22**
11100	**23**
10100	**24**
10101	**25**
10111	**26**
10110	**27**
10010	**28**
10011	**29**
10001	**30**
10000	**31**
110000	**32**

Conversion du code Gray vers le code binaire

Soit $X = G_n\ G_{n-1}.....G_0$ représenté en code Gray
Pour convertir X en binaire il faut suivre les règles suivantes :

Bn = Gn
$B_i = 0$ si $B_{i+1} = G_i$
$B_i = 1$ si $B_{i+1} <> G_i$

Exemple : pour n = 4 X = 10101 en code Gray $X = (\ 11001\ )_2$

1 0 1 0 1
G4 G3 G2 G1 G0
1 1 0 0 1
B4 B3 B2 B1 B0

Conversion du code Gray vers le code binaire

Pour passer d'un nombre au nombre suivant (successeur) :
si le nombre de 1 est pair, il faut inverser le bit de poids faible,
si le nombre de 1 est impair, il faut inverser le chiffre situé à gauche du 1 le plus à droite.

Exemple 1 : 32 = 110000

110000 : Le nombre de 1 est paire ➔ nombre suivant (33) est : 110001
110001 : Le nombre de 1 est impaire ➔ nombre suivant (34) est : 110011
110011 : Le nombre de 1 est paire ➔ nombre suivant (35) est : 110010
……

Exemple 2 : X = 1011101010
1011101010 : Le nombre de 1 est paire ➔ nombre suivant est : 1011101011
1011101011 : Le nombre de 1 est impaire ➔ nombre suivant est : 1011101001
1011101001 : Le nombre de 1 est paire ➔ nombre suivant est : 1011101000
1011101000 : Le nombre de 1 est impaire ➔ nombre suivant est : 1011111000
……

Codification et représentation -Numérique

Codification et représentation –Numérique : Le code ASCII

Le code ASCII

Définition Le code ASCII : American Standard Code For Information Interchange,

Est une norme de codage en informatique mise au point dans les années 60.

Ce code définit 128 caractères représentés sur 8 bits.

Codification et représentation – Numérique : Le code ASCII

Cette table est présentée sous une forme condensée, fondée sur la base 16. Chaque caractère se trouve au croisement d'une ligne et d'une colonne. Le numéro de la ligne suivi du numéro de la colonne représentent le code du caractère en hexadécimal.

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

Codification et représentation – Numérique : Le code ASCII

Exemple : La lettre M se trouve au croisement de la ligne 4 et de la colonne D
son code est 4D en hexadécimal.
Sa représentation est donc : 0100 1101 en code ASCII 0

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

Codification et représentation –Numérique : Le code ASCII

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US

Caractères de contrôle :

On peut considérer que l'ASCII dispose d'une trentaine de caractères de contrôle plus ou moins utilisés.

Les caractères usuels sont NUL, LF, CR, DEL et ESC

- **NUL :** indique la fin d'une chaine de caractères notamment en langage C.
- **LF** et **CR** : indiquent la fin d'une ligne.
- On utilisera **LF, CR** ou les 2 selon le système d'exploitation :
  - Sous Linux par exemple ce sera **LF**,
  - sous Mac OS on utilise **CR**
  - sous Windows ce sera **CR** suivi de **LF**.
- **ESC** indique la sortie d'un texte.
- **DEL** indique l'effacement d'un caractère.

Codification et représentation –Numérique : Le code ASCII

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O

Exemple : Dans un éditeur de texte ou dans un champ de saisie taper sans relâcher la touche alt + le code de caractère en décimal :

$4D_{(16)} = 77_{(10)}$ ➔ M
$30_{(16)} = 48_{(10)}$ ➔ 0
$3F_{(16)} = 63_{(10)}$ ➔ ?
$45_{(16)} = 69_{(10)}$ ➔ E

Codification et représentation –Numérique : Le code ASCII

<u>Exercise N°7:</u> (à faire comme un exemple en cours)

1/ En code ASCII (41)16 correspond à 'A' et (30)16 correspond à '0', sans l'utilisation de la table du code ASCII déduire le codage du message suivant : Covid-19

2/ Décoder le message suivant :4269656E76656E757320656E204F51

Codification et représentation –Numérique : Le code ASCII

<u>Exercice N°7:</u> (à faire comme un exemple en cours)

1/ En code ASCII

$(41)_{16}$ ➔ 'A'

$(61)_{16}$ ➔ 'a'

$(30)_{16}$ ➔ '0',

$(2D)_{16}$ ➔ '-'

Sans l'utilisation de la table du code ASCII déduire le codage du message suivant :

Covid-19

'C' = 43 'o' = 6F 'v' = 76 'i'= 69 'd' = 64

'-' = 2D '1'= 31 '9'= 39

Codification et représentation –Numérique : Le code ASCII

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

Exercice N°7:

2/ Décoder le message suivant :42 69 65 6E 76 65 6E 75 73 20 65 6E 20 4D 49

Codification et représentation –Numérique : Le code UNICODE

Le code UNICODE

’Unicode est une norme de codage mise au point dans les années 90;
il définit plus de **60000** caractères de plusieurs langues, codés sur **16 bits**.
Le code ASCII est inclus dans l’Unicode.

Le code ASCII est uniquement basé sur les lettres anglo-saxonnes, on n’y trouve pas les lettres accentuées de la langue française comme le **à** ou le **é** par exemple.
On peut retrouver ces lettres dans le tableau UNICODE suivant :

Codification et représentation –Numérique : Le code UNICODE

Le code UNICODE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
008	PAD	HOP	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
009	DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	SGCI	SCI	CSI	ST	OSC	PM	APC
00A	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY -	®	¯
00B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
00C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
00D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
00E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
00F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Codification et représentation –Numérique : Le code UNICODE

Le code UNICODE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
008	PAD	HOP	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
009	DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	SGCI	SCI	CSI	ST	OSC	PM	APC
00A	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY -	®	¯
00B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
00C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
00D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
00E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
00F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Codification et représentation –Numérique : Le code UNICODE

Le code UNICODE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D
060	ـــ	ـــ	ـــ	ـص	□	□	□	□	□	□	□	ف	،	م
061	ـَّ	ـُّ	ـِّ	ـّ	ـَ	ـْ	ـِ	ـِ	ـِ	ـِ	ـِ	؛	ALM	
062	□	ء	آ	أ	ؤ	إ	ئ	ا	ب	ة	ت	ث	ج	ح
063	ذ	ر	ز	س	ش	ص	ض	ط	ظ	ع	غ	□	□	□
064	ـ	ف	ق	ك	ل	م	ن	ه	و	ى	ي	ـً	ـّ	ـٍ
065	ـِ	ـَ	ـْ	ـَ	ـُ	ـِ	ـِ	ـُ	ـُ	ـَ	ـَ	ـُ	ـِ	ـْ
066	٠	١	٢	٣	٤	٥	٦	٧	٨	٩	٪	,	،	*
067	ـٰ	آ	أ	إ	ٔ	ؤ	ؤ	ؤ	ئ	ٹ	ٹ	پ	ټ	ٹ
068	ڀ	څ	څ	ڃ	ڃ	څ	چ	چ	ڈ	ډ	ډ	ڈ	ډ	ډ
069	ڈ	ڑ	ز	ړ	ړ	ړ	ړ	ړ	ڑ	ړ	ښ	ږ	ږ	ږ
06A	څ	ڤ	ڤ	ڤ	ڤ	ڤ	ڤ	ف	ق	ک	ک	ک	ک	ک
06B	گ	گ	ګ	ګ	ګ	ڵ	ڵ	ڵ	ڵ	ڼ	ں	ڻ	ڼ	ڼ

Chapitre 2 :

Algèbre de Boole

Algèbre de Boole

Introduction

L'ordinateur est constitué de circuits logiques

L'élément de base de ces circuits est le transistor, on a deux états 0 et 1

0 =Bloqué 1 = Conducteur

Les variables d'entrée sont celles sur lesquelles on peut agir directement.
Ce sont des variables logiques indépendantes.

La variable de sortie est celle qui contient l'état de la fonction après l'évaluation des opérateurs logiques sur les variables d'entrée.

Pour réaliser ces circuits et déterminer les variables d'entrée et les variables de sortie on utilisera

l'Algèbre de Boole

Al

Terminologie

- Somme **(OR)** s = a **+** b ou bien s = a **or** b
- Produit **(AND)** s = a ***** b ou bien s = a **and** b
- A ceci s'ajoute une application unaire :
- Complémentation **(NOT)** $\bar{s}$ ou bien **not**(s)

Algèbre de Boole

Terminologie

- Somme **(OR)** s = a **+** b ou bien s = a **or** b
- Produit **(AND)** s = a ***** b ou bien s = a **and** b ou s = a.b
- A ceci s'ajoute une application unaire :
- Complémentation **(NOT)** $\bar{s}$ ou bien **not**(s)

s = a . b

a	b	a . b
0	0	0
0	1	0
1	0	0
1	1	1

s = a + b

A	B	a + b
0	0	0
0	1	1
1	0	1
1	1	1

s = $\bar{a}$

A	$\bar{a}$
0	1
1	0

Al

Remarque :

On peut utiliser la terminologie française :

ET pour **AND**

OU pour **OR**

NON pour **NOT**

$s = a \cdot b$

a	b	a . b
0	0	0
0	1	0
1	0	0
1	1	1

$s = a + b$

A	B	a + b
0	0	0
0	1	1
1	0	1
1	1	1

$s = \bar{a}$

A	$\bar{a}$
0	1
1	0

Algèbre de Boole

Théorèmes et postulats de l'algèbre de Boole

Une Algèbre de Boole est constituée d'un ensemble E = {0,1} et de deux lois de composition internes (AND) et (OR) :

Propriétés de l'Algèbre de Boole

1- Commutativité

$a + b = b + a$

$a \cdot b = b \cdot a$

2- Associativité

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

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

Algèbre de Boole

Propriétés de l'Algèbre de Boole

3- Distributivité

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

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

4- Eléments neutre

$a + 0 = a$

$a \cdot 1 = a$

5- Eléments symétrique

$a + \bar{a} = 1$

$a \cdot \bar{a} = 0$

Algèbre de Boole

Propriétés déduites

1- Idempotence

$a + a = a$

$a \cdot a = a$

2- Élément absorbant

$a + 1 = 1$

$a \cdot 0 = 0$

3- Expressions usuelles simplifiées

$a + a \cdot b = a$

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

$a + \bar{a}\, b = a + b$

Algèbre de Boole

Les opérateurs **NAND** et **NOR** tables de vérité

$s = \overline{a \cdot b}$

a	b	$\overline{a \cdot b}$
0	0	1
0	1	1
1	0	1
1	1	0

$s = \overline{a + b}$

a	b	$\overline{a + b}$
0	0	1
0	1	0
1	0	0
1	1	0

Algèbre de Boole

Le OR exclusif (XOR) et son complément

$s = a \oplus b$

$s = 1$ si $a \neq b$

$s = 0$ si $a = b$

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

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

table de vérité

a	b	$a \oplus b$	$\overline{a \oplus b}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

Algèbre de Boole

Expressions booléennes

Fonctions logiques et formes normales

On appelle **min terme** de n variables, l'un des produits de ces n variables ou de leurs complémentaires.

Exemple : $a\,\bar{b}\,c$, $\bar{a}\,b\,\bar{c}$ et $a\,b\,c$

sont des min termes d'une fonction de 3 variables a, b, et c

Abc /abc a/bc ab/c /a/bc….

Algèbre de Boole

Expressions booléennes

Fonctions logiques et formes normales

On appelle **max terme** de n variables, l'une des sommes de ces n variables ou de leurs complémentaires.

Exemple : $(a + \bar{b} + c)$, $(\bar{a} + b + \bar{c})$ et $(a + b + c)$

sont des max termes d'une fonction de 3 variables a, b, et c

Algèbre de Boole

Première forme normale ou forme disjonctive

Une fonction est sous **forme disjonctive** si elle est représentée par une somme de min termes (somme de produits)

Exemple :

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

Algèbre de Boole

Deuxième forme normale ou forme conjonctive

Une fonction est sous **forme conjonctive** si elle est représentée par un produit de max termes (produit de sommes)

Exemple :

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

Remarque :
On peut passer d'une forme à l'autre en utilisant la distributivité

Algèbre de Boole

Lois de Morgan (A vérifier à l'aide d'une table de vérité)

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

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

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

Algèbre de Boole

Complément d'une fonction

∀F une fonction booléenne ∃ G tel que G = /F

Pour calculer /F il faut utiliser les Lois de Morgan

Exemple : $F(a, b, c) = a\,\overline{b}\,c + \overline{a}\,b\,\overline{c} + a\,b\,c$

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

$$\overline{F}(a, b, c) = \overline{a\,\overline{b}\,c} * \overline{\overline{a}\,b\,\overline{c}} * \overline{a\,b\,c}$$

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

On remarque que pour trouver /F il suffit d'inverser chaque variable et chaque opérateur

Algèbre de Boole

Circuits logiques

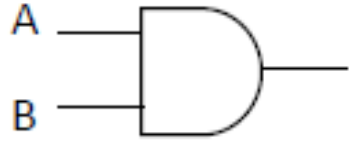
Un circuit logique est un ensemble de portes logiques reliées entre elles correspondant à une expression algébrique

Portes logiques (correspondant à un opérateur logique) :

Porte Or

$Y = A + B$

Porte And



$Y = A \cdot B$

Porte Not

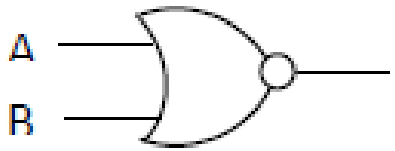
$Y = \overline{A}$

Algèbre de Boole

Circuits logiques

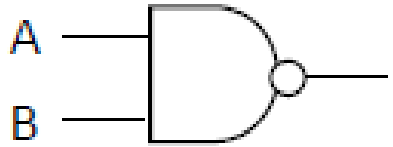
Portes dérivées :

Porte Nor



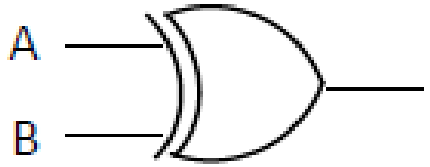
$Y = \overline{A + B}$

Porte Nand



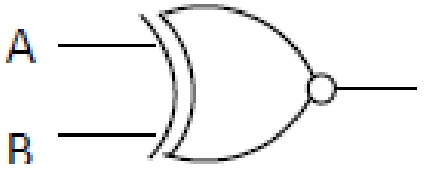
$Y = \overline{A \,.\, B}$

Porte Xor



$Y = A \oplus B$

Porte NXOR

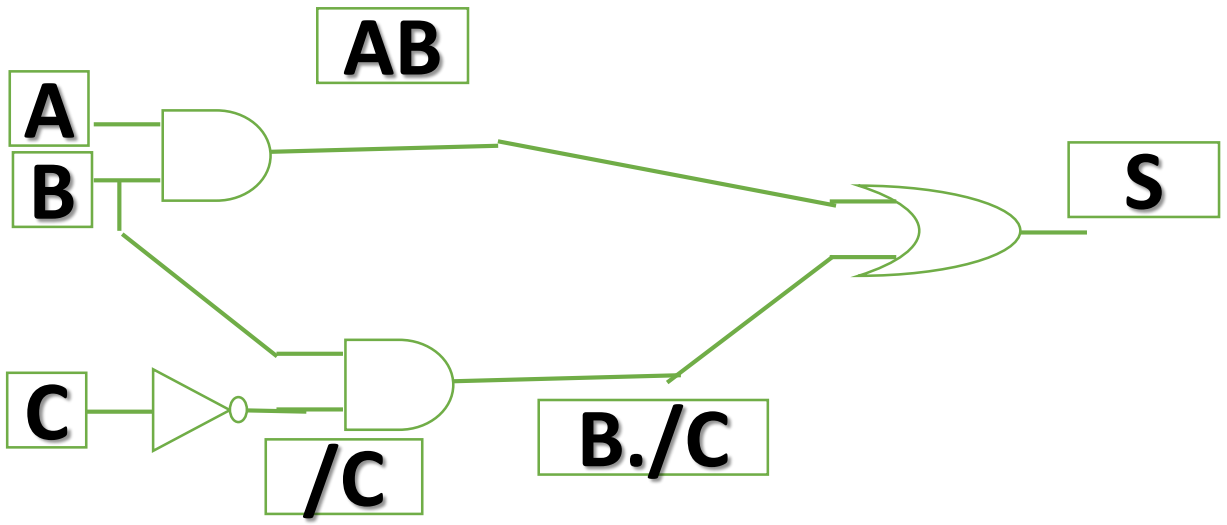


$Y = \overline{A \oplus B}$

Algèbre de Boole

Circuits logiques

Circuit logique correspondant à l'expression algébrique :
S = A B + B./C



Algèbre de Boole

Table de vérité d'une fonction

La Table de Vérité d'une fonction consiste à retrouver les valeurs de celle-ci pour chaque combinaison de variables.

Algèbre de Boole

Table de vérité d'une fonction

Soit la fonction :

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

F(A, B, C) = 1 si un de ses termes est égal à 1

$\overline{A}\, B\, C = 1$ si A = 0 B = 1 et C = 1 F(0 1 1) = 1

$A\, \overline{B}\, \overline{C} = 1$ si A = 1 B = 0 et C = 0 F(1 0 0) = 1

A B = 1 si A = 1 et B = 1 F(1 1 0) = 1 et F(1 1 1) = 1

Algèbre de Boole /F = /A/B/C + /A/BC +/AB/C + A/BC

La Table de Vérité de F est :

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Pour tirer l'expression d'une fonction à partir d'une Table de Vérité on fait la somme des min termes où F = 1

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

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

$\overline{F}(A, B, C)$ est obtenu en faisant la somme des min termes où F = 0

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

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

Algèbr

On retrouve la forme conjonctive de cette fonction à partir de $\overline{F}$. $F(A, B, C) = \overline{\overline{F}}(A, B, C)$

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

La Table

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Pour tirer l'expression d'une fonction à partir d'une Table de

Vérité on fait la somme des min termes où F = 1

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

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

$\overline{F}(A, B, C)$ est obtenu en faisant la somme des min termes où F = 0

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

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

Algèbre de Boole

Simplification des fonctions booléennes

1- Simplification algébrique

Pour simplifier algébriquement une fonction booléenne, on utilise les propriétés de l'algèbre de Boole : idempotence, absorption, distributivité, mise en facteur …etc

Algèbre de Boole

Exemple :

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

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

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

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

$$F(a,b,c) = b + a\, \overline{b} \qquad (b + a)(b + \overline{b})$$

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

Algèbre de Boole

2- Simplification par le tableau de Karnaugh

Un tableau de Karnaugh est une table de vérité à 2 dimensions.

La numérotation des lignes et des colonnes se fait selon **le code de Gray,**
On passe d'une ligne à la suivante **en changeant un seul bit** et d'une colonne à la suivante **en changeant un seul bit** également.

Algèbre de Boole

Simplification par le tableau de Karnaugh

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

A	B	C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Algèbre de Boole

Simplification par le tableau de Karnaugh

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

Chaque case représente une combinaison.

Si on prend une case dans un tableau de Karnaugh, toutes les cases qui lui sont adjacentes n'auront qu'un seul bit qui change
donc une seule variable change

A	B	C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Algèbre de Boole

Simplification par le tableau de Karnaugh

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

Cha

Si c

Kar ntes

n'a

dor

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

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

Algèbre de Boole

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

d) = /a/c d + /ac/d+ /b/c d

Algèbre de Boole

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

,d) = cd + /c/d+ ab + ac

Pour la forme conjonctive (produit de sommes) on travaille avec les zéro.

Algèbre de Boole

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

$\overline{F}$(a,b,c,d) = $\overline{c}\overline{d} + ab + cd + ac$

$F = \overline{\overline{F}}(a,b,c,d) = \overline{\overline{c}\overline{d} + ab + cd + ac}$

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

Fonction incomplète

On dit qu'une fonction est incomplète si elle n'est pas définie en tous ses points.

Dans ce cas les points où elle n'est pas définie prendront la valeur X.

Lors de la simplification de la fonction par le tableau de Karnaugh, si une case contenant X est adjacente à une case contenant 1, on pourra donner la valeur 1 à ce X de façon à simplifier la fonction.

A	B	C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Fonction incomplète

ab / cd	00	01	11	10
00	0	0	X	0
01	1	1	X	1
11	0	0	X	X
10	1	1	x	x

A	B	C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Fonction incomplète

ab / cd	00	01	11	10
00	0	0	X	0
01	1	1	X=1	1
11	0	0	X	X
10	1	1	X=1	X=1

$F(a,b,c,d) = \cd + c\d$

Fonction incomplète

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

$$F(a,b,c,d) = ac + \bar{b}\bar{d} + b\bar{c}\bar{d}$$

Fonction incomplète

Une serrure de sécurité s'ouvre en fonction de quatre clés A, B, C et D.

Le fonctionnement de la serrure est définie comme suit :

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

- S(A B C D)= 0 sinon

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

A	B	C	D	S
0	**0**	**0**	**0**	0
0	**0**	**0**	**1**	0
0	**0**	**1**	**0**	0
0	**0**	**1**	**1**	1
0	**1**	**0**	**0**	0
0	**1**	**0**	**1**	1
0	**1**	**1**	**0**	1
0	**1**	**1**	**1**	1
1	**0**	**0**	**0**	0
1	**0**	**0**	**1**	X
1	**0**	**1**	**0**	1
1	**0**	**1**	**1**	**X**
1	**1**	**0**	**0**	1
1	**1**	**0**	**1**	X
1	**1**	**1**	**0**	1
1	**1**	**1**	**1**	X

Fonction incomplète

Un
qu

Le
co

S (
uti

S(A
pa

ab / cd	00	01	11	10
00	0	0	1	0
01	0	1	X	X
11	1	1	x	x
10	0	1	1	1

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

F

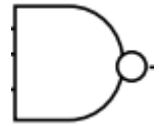
ab / cd	00	01	11	10
00	0	0	1	0
01	0	1	X	X
11	1	1	X	X
10	0	1	1	1

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

F(a,b,c,d) = cd + bd+ bc+ ab+ ac

Algèbre de Boole

Utilisation des portes NAND

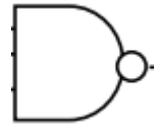


Pour réaliser le circuit d'une fonction à l'aide de portes NAND il faut que la fonction soit sous forme disjonctive, il suffit alors de la complémenter 2 fois et d'utiliser les lois de Morgan.

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

Algèbre de Boole

Utilisation des portes NAND



Pour réaliser le circuit d'une fonction à l'aide de portes NAND il faut que la fonction soit sous forme disjonctive, il suffit alors de la complémenter 2 fois et d'utiliser les lois de Morgan.

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

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

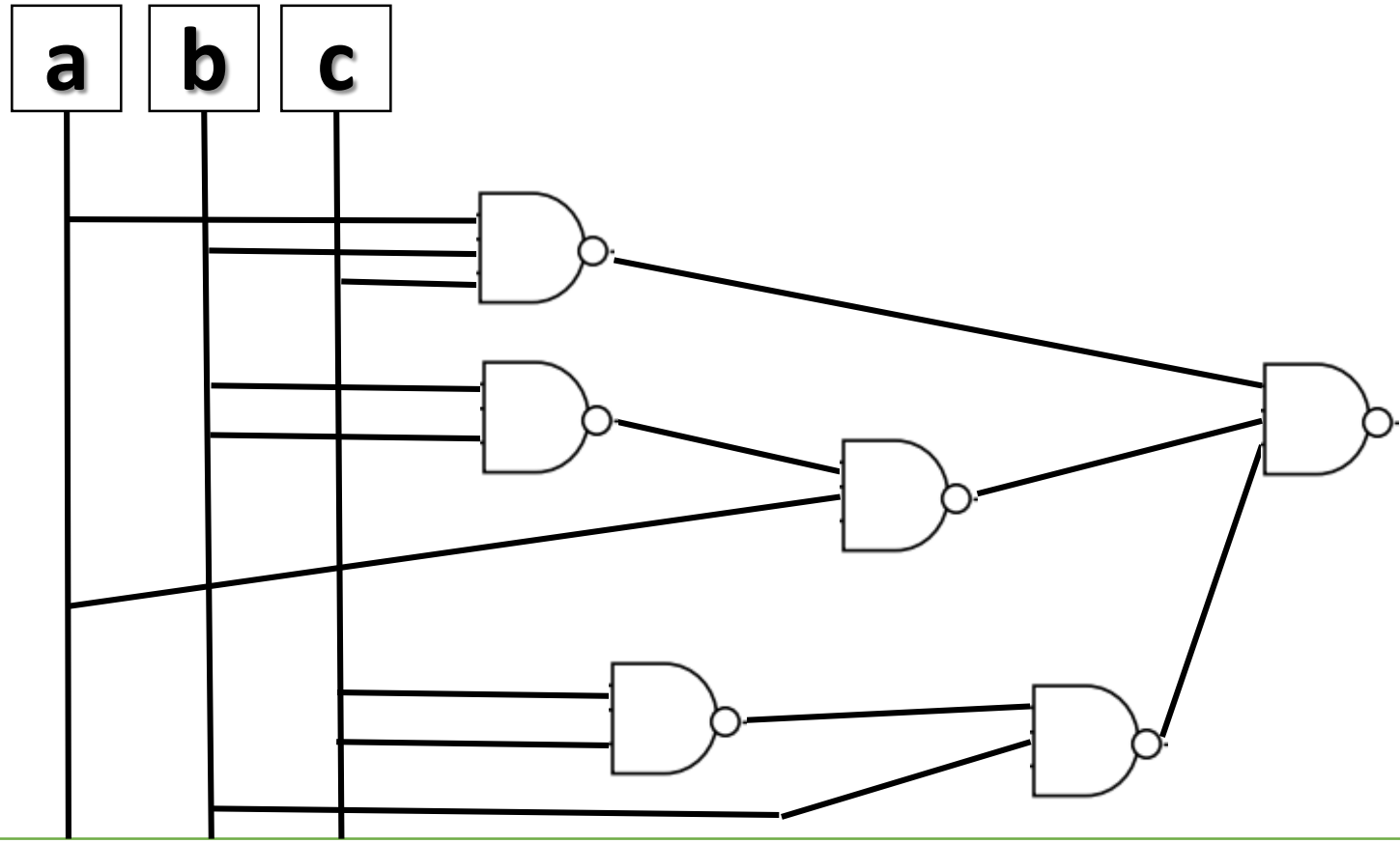
Algèbre de Boole

Utilisation des portes NAND

/B = /(BB) = /(B+B) **B*B = B**

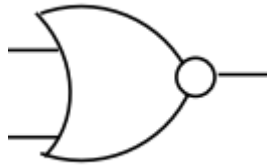
$$F(ABC) = A\,B\,C + A\,\overline{B} + B\,\overline{C}$$

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



Algèbre de Boole

Utilisation des portes NOR



Pour réaliser le circuit d'une fonction à l'aide de portes NOR il faut la fonction soit sous forme **conjonctive**, il suffit alors de la **complémenter 2 fois** et d'utiliser les lois de Morgan.

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

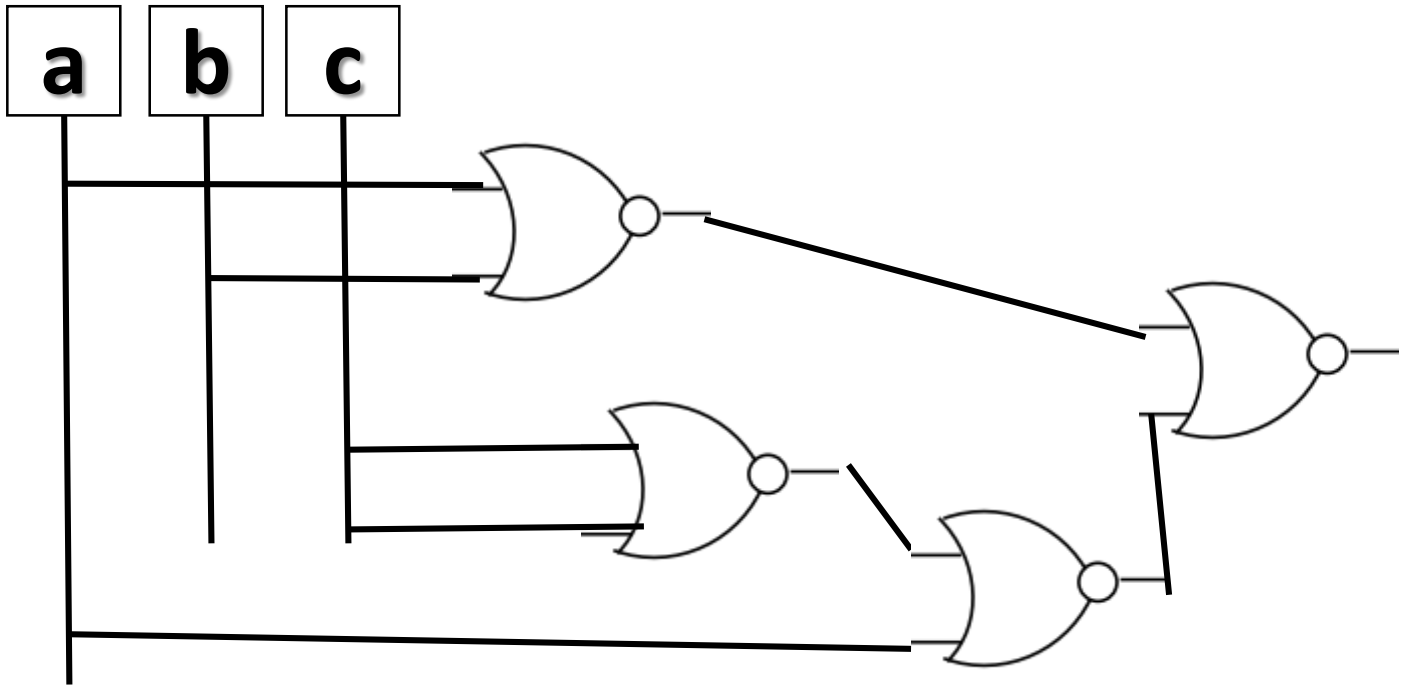
$$G(ABC) = \overline{\overline{(A + B)(A + \overline{C})}} = \overline{\overline{(A + B)} + \overline{(A + \overline{C})}}$$

Algèbre de Boole

Utilisation des portes NOR

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

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



Chapitre 3 :

circuits combinatoires

L'Additionneur

Le Demi-Additionneur

Le Demi Additionneur à un bit est un circuit qui permet d'additionner 2 chiffres binaires sans retenue rentrante

Lorsqu'on fait l'addition de 2 chiffres binaires A et B on obtient la somme S et une retenue R.

A + B = S retenue R
S = /ab + a/b = a xor b
R = ab

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

L'Additionneur

Le Demi-Additionneur

Le Demi Additionneur à un bit est un circuit qui permet d'additionner 2 chiffres binaires sans retenue rentrante

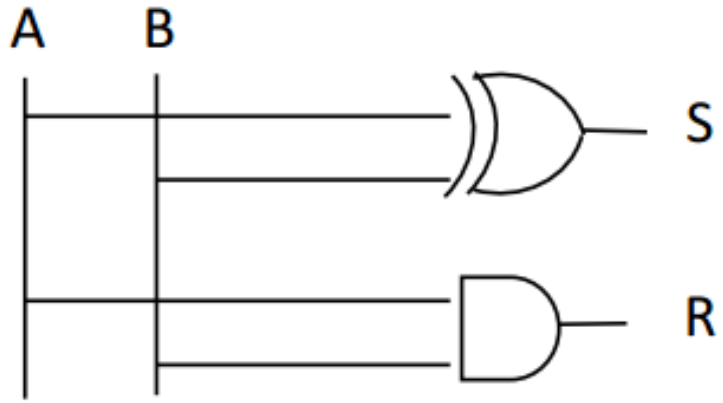
Lorsqu'on fait l'addition de 2 chiffres binaires A et B on obtient la somme S et une retenue R.

A + B = S retenue R

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

$$S = A \oplus B$$

$$R = A\,B$$



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

L'Additionneur

L'Additionneur complet

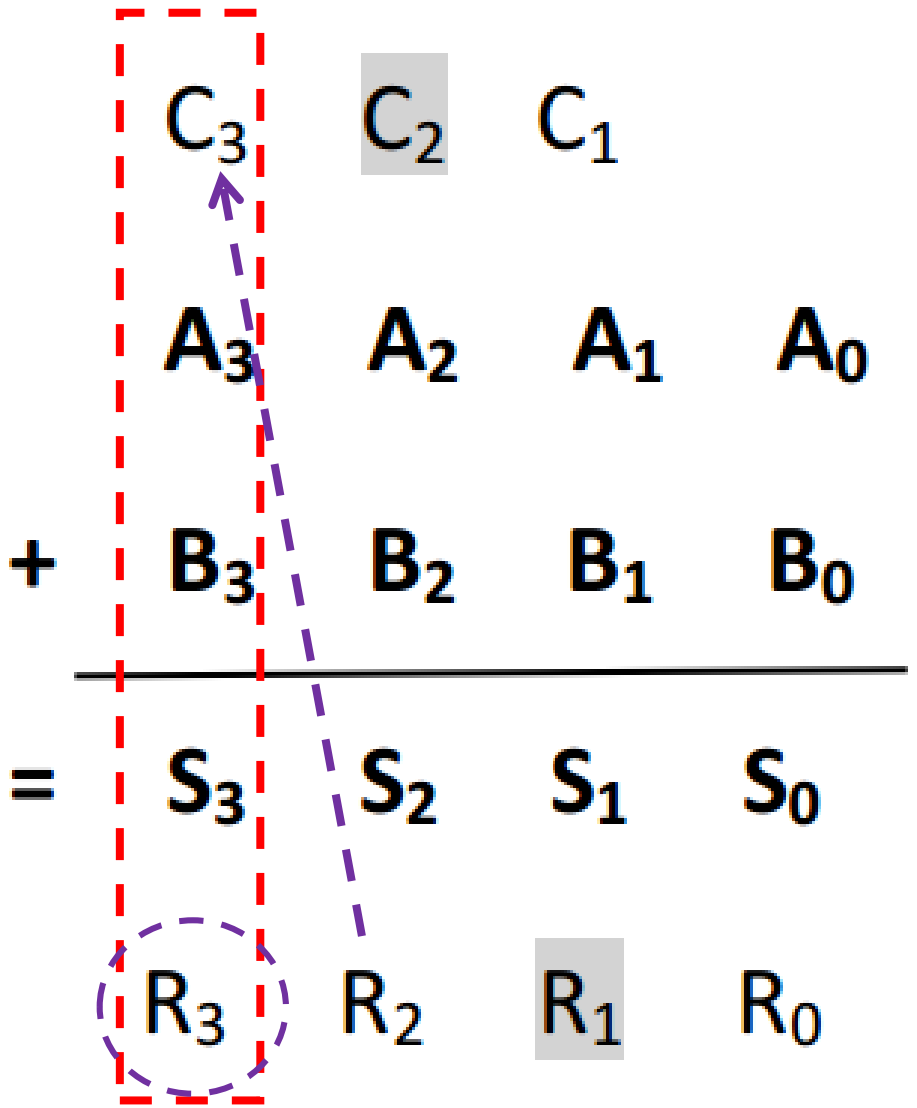
L'Additionneur complet à un bit est un circuit qui permet d'additionner 2 chiffres binaires avec une retenue rentrante

Lorsqu'on fait l'addition de 2 nombres binaires, on travaille par rangées,

on additionne les chiffres 2 à 2 selon leurs poids (de droite à gauche).

A la fin de chaque addition on obtient une somme et une retenue sortante,

cette retenue s'ajoutera à la somme de la rangée précédente et deviendra une retenue rentrante.

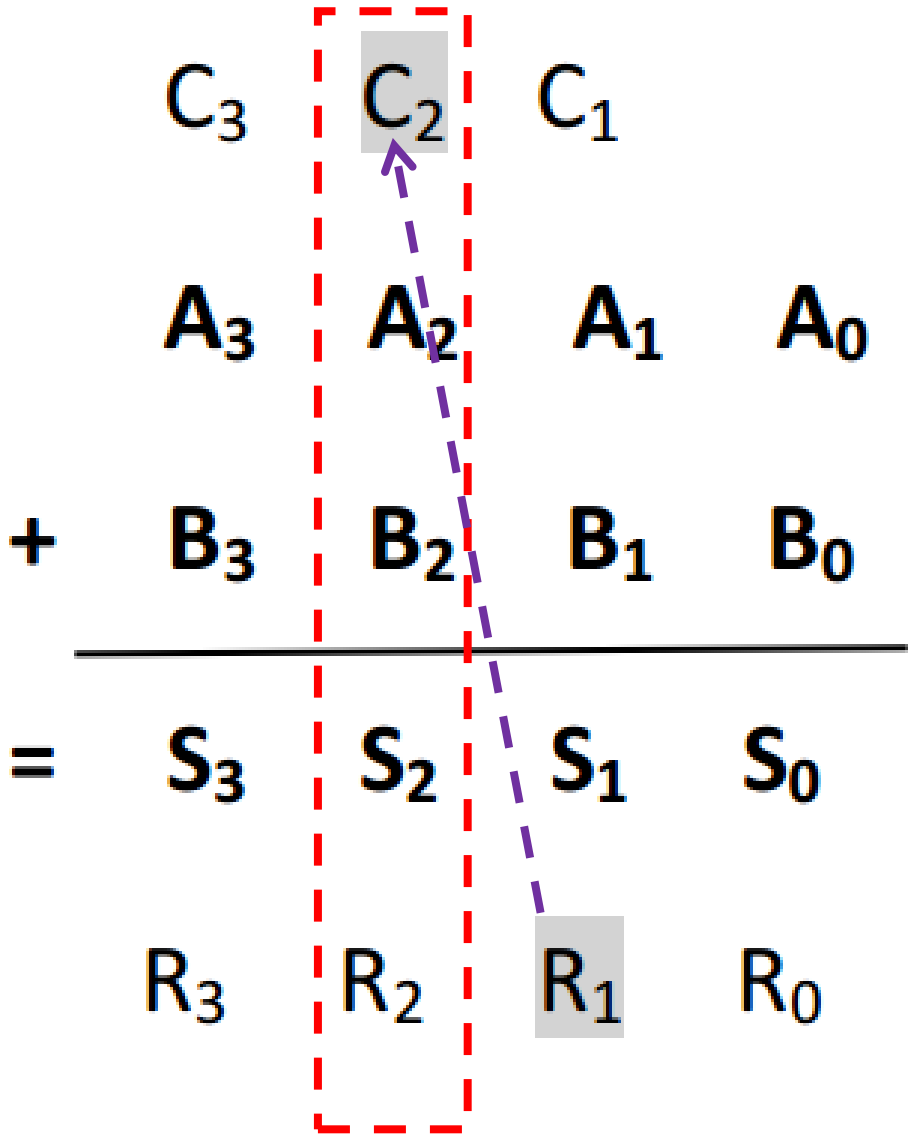


L'Additionneur

L'Additionneur complet

L' Il n'y a jamais de retenue rentrante sur la première rangée (bits de poids faibles)

Pour cette rangée on a donc un demi-additionneur, pour toutes les autres on aura des additionneurs complets



L'Additionneur

$$R(a,b,c) = ab + bc + ac$$

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

A	B	C	S	R
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

$$S(a,b,c) = /a(/bc + b/c) + a(bc + /b/c)$$

$$S(a,b,c) = /a(b \text{ xor } c) + a\overline{(b \text{ xor } c)} = a \text{ xor } b \text{ xor } c$$

L'Additionneur

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

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

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

A	B	C	S	R
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

$$S = \bar{a}\bar{b}c + \bar{a}b\bar{c} + abc + abc + ab + a\bar{b}\bar{c}$$

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

$$S = \bar{a}(b \oplus c) + a\overline{(b \oplus c)}$$

$$S = a \oplus b \oplus c$$

L'Additionneur

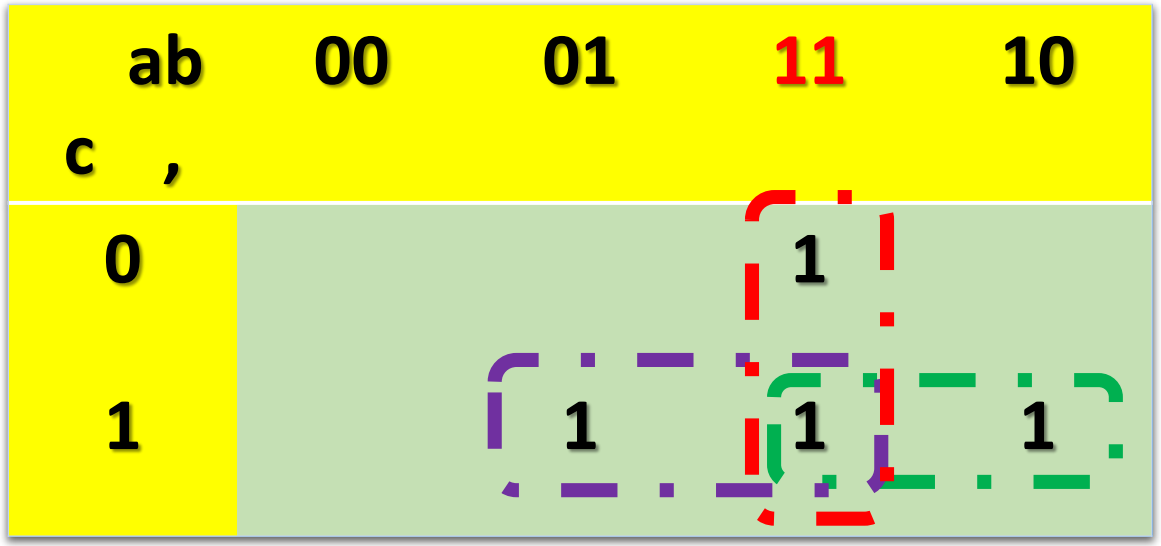
$$a \oplus b = \bar{a}.b + a.\bar{b}$$

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

ab c ,	00	01	11	10
0				
1				

A	*B*	*C*	S	R
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

L'Additionneur



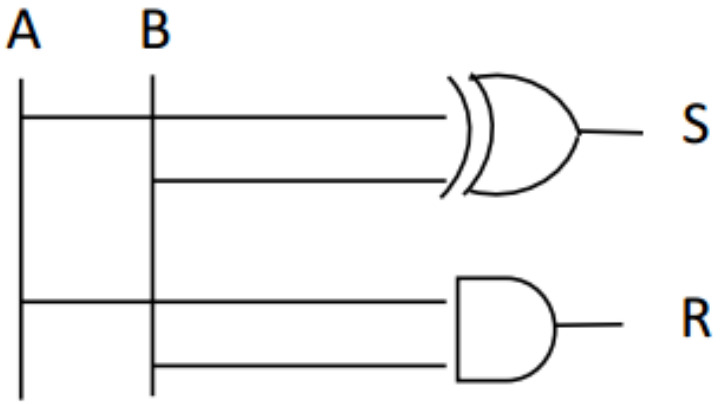
R = AB+ AC + BC

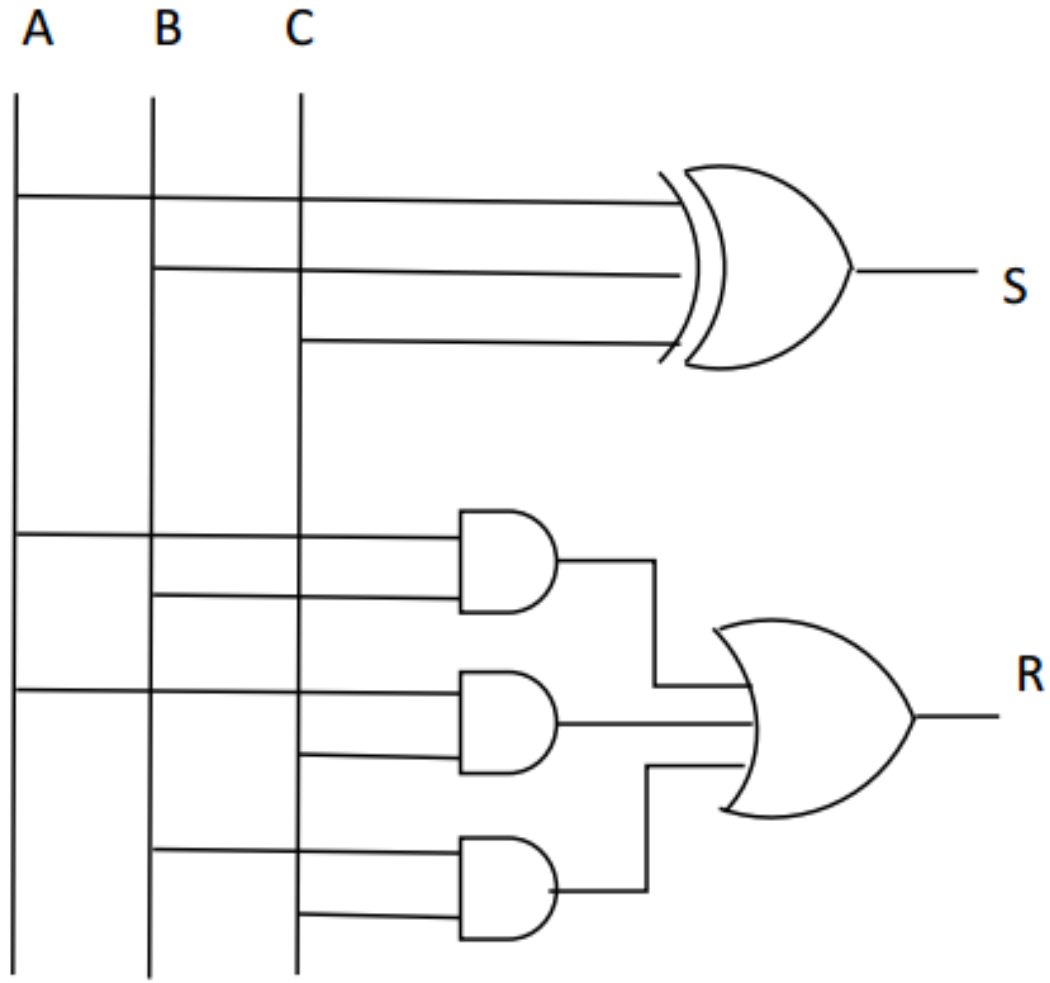
A	B	C	S	R
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

L'Additionneur

S = $a \oplus b \oplus c$

R = AB+ AC + BC



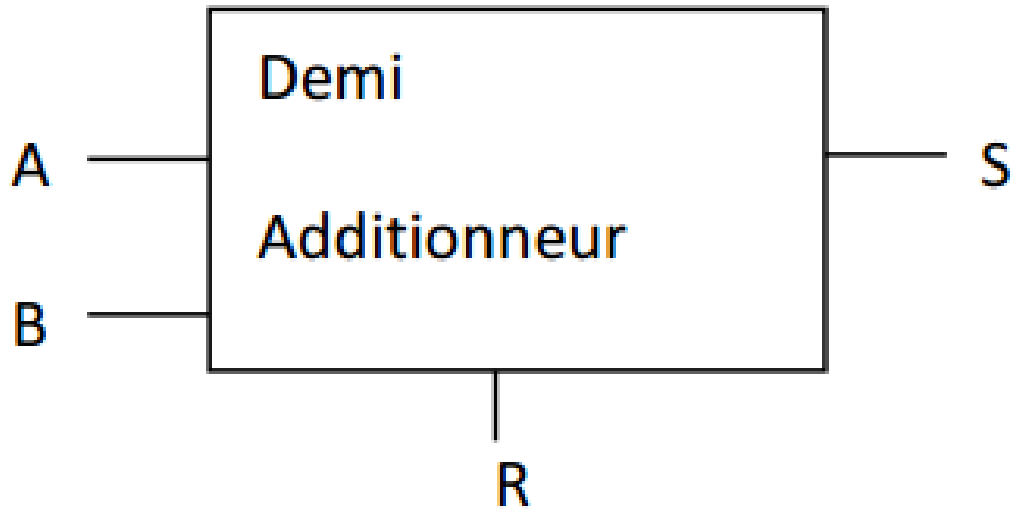


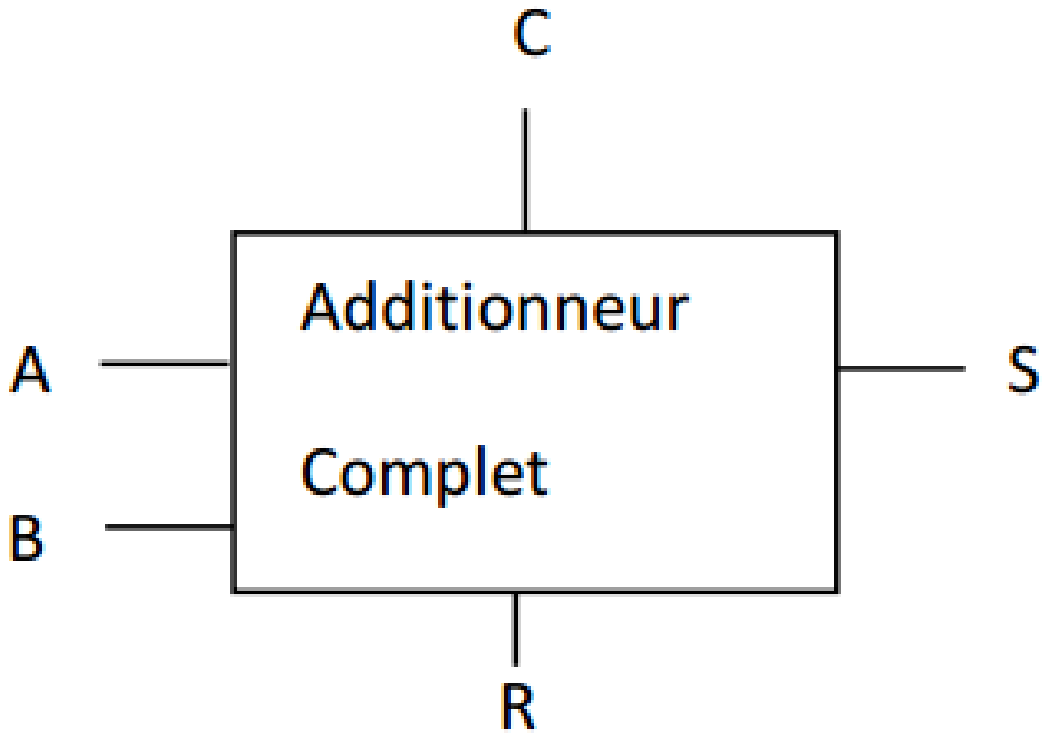
L'Additionneur

Circuits fonctionnels

Un circuit fonctionnel est un composant portant le nom de la fonction qu'il représente.

Il possède les entrées et les sorties de cette fonction





Circuit d’un Additionneur à 4 bits

Pour réaliser un additionneur à 4 bits il faut connecter 4 additionneurs à un bit,

un demi- additionneur et trois additionneurs complets.

La connexion se fait par les retenues Ci = Ri-1

Ce circuits fait l’addition de deux nombres

A3 A2 A1 A0

et B3 B2 B1 B0

Le résultat est **S3 S2 S1 S0**

et la retenue finale est **R3**

Circuit d’un Additionneur à 4 bits

Pour réaliser un additionneur à 4 bits il faut connecter 4 additionneurs à un bit,

un demi- additionneur et trois additionneurs complets.

La connexion se fait par les retenues Ci = Ri-1

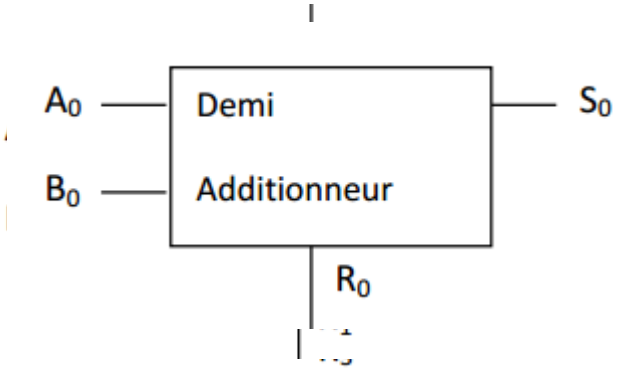
Ce circuits fait l’addition de deux nombres

A3 A2 A1 A0

et B3 B2 B1 B0

Le résultat est **S3 S2 S1 S0**

et la retenue finale est **R3**



Circuit d'un Additionneur à 4 bits

Les R_i sont les retenues sortantes

Les C_i sont les retenues rentrantes

$\mathbf{C_i = R_{i-1}}$

$A_1 + B_1 + C_1 = S_1$ retenue R_1 $\quad C_2 = R_1$

$A_2 + B_2 + C_2 = S_2$ retenue R_2

$$\begin{array}{cccccc} & C_3 & C_2 & C_1 & & \\ & A_3 & A_2 & A_1 & A_0 & \\ + & B_3 & B_2 & B_1 & B_0 & \\ \hline = & S_3 & S_2 & S_1 & S_0 & \\ & R_3 & R_2 & R_1 & R_0 & \end{array}$$

Circuit d'un Additionneur à 4 bits

Pour réaliser un additionneur à 4 bits il faut connecter 4 additionneurs à un bit, un demi- additionneur et trois additionneurs complets.

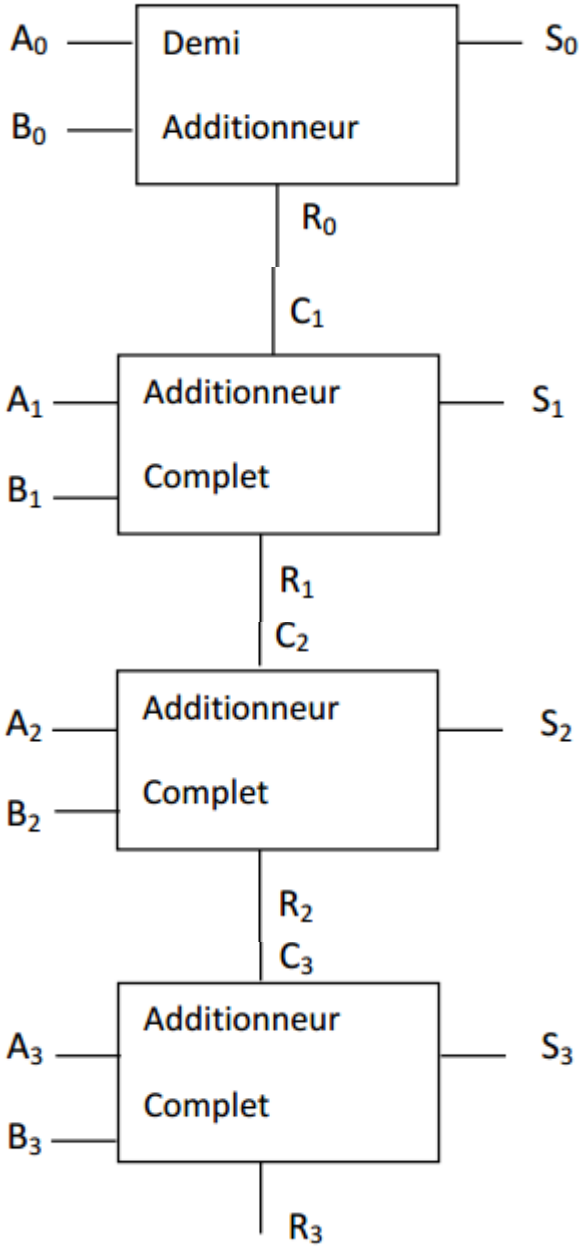
La connexion se fait par les retenues

$C_i = R_{i-1}$

Ce circuits fait l'addition de deux nombres

$A_3 A_2 A_1 A_0$ et $B_3 B_2 B_1 B_0$

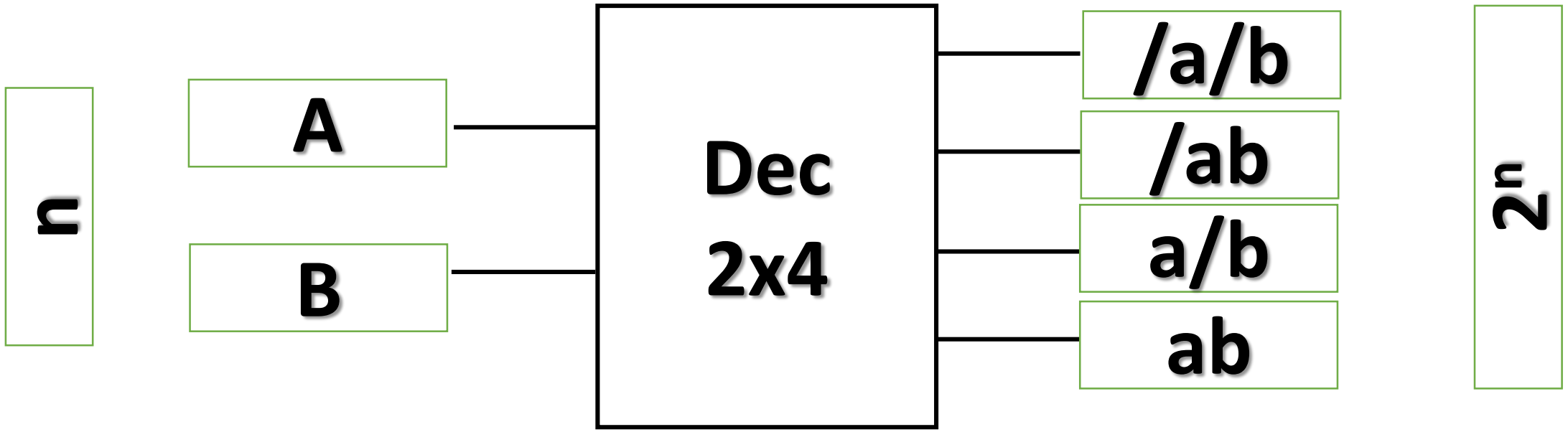
Le résultat est $S_3 S_2 S_1 S_0$ et la retenue finale est R_3



Le décodeur

Un décodeur est un circuit combinatoire qui a n entrées et 2^n sorties dont une seule est égale à 1

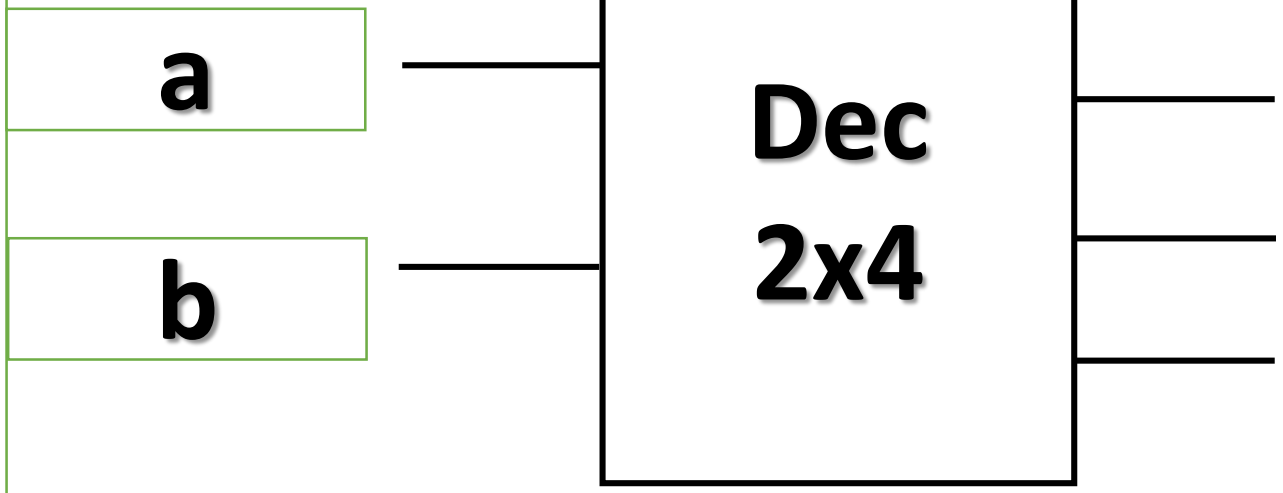
L'exemple suivant représente un décodeur 2x4

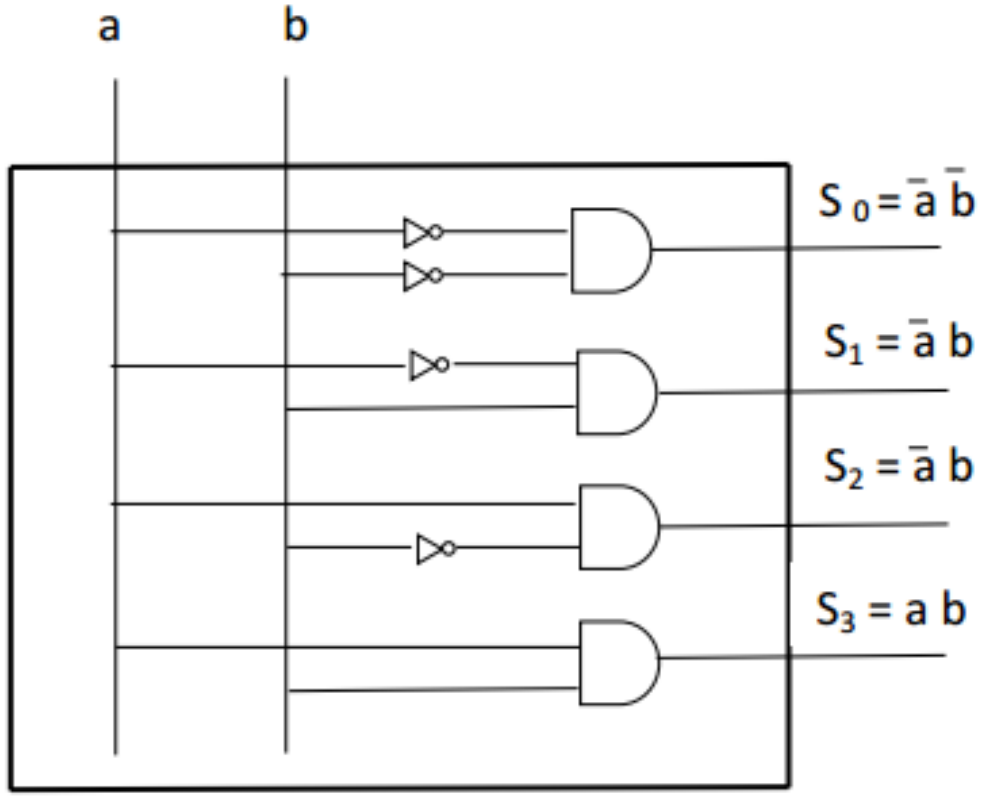


Le décodeur

Un décodeur est un circuit combinatoire qui a n entrées et 2^n sorties dont une seule est égale à 1

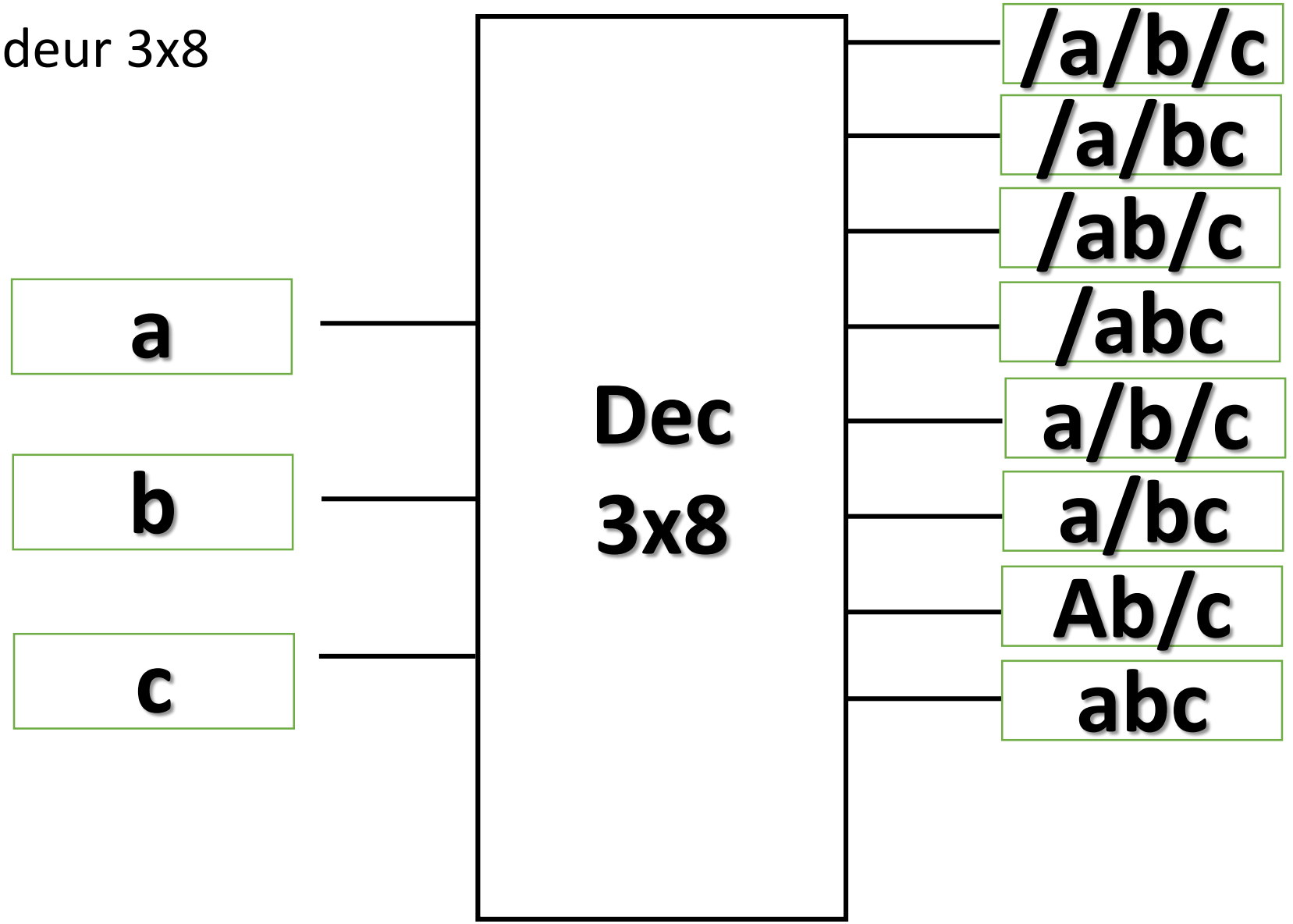
L'exemple suivant représente un décodeur 2x4





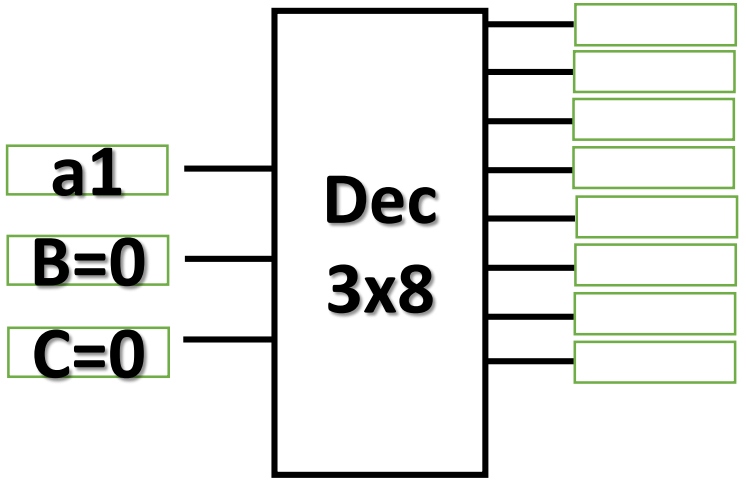
Le décodeur

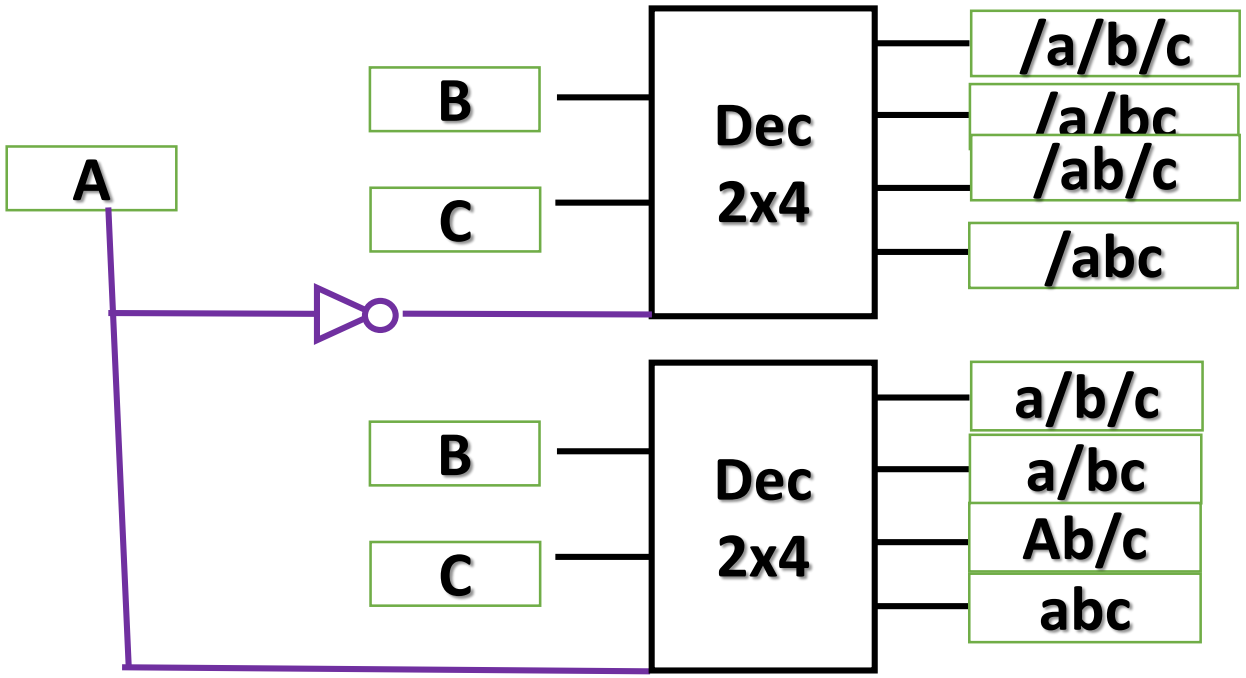
un décodeur 3x8



Le décodeur

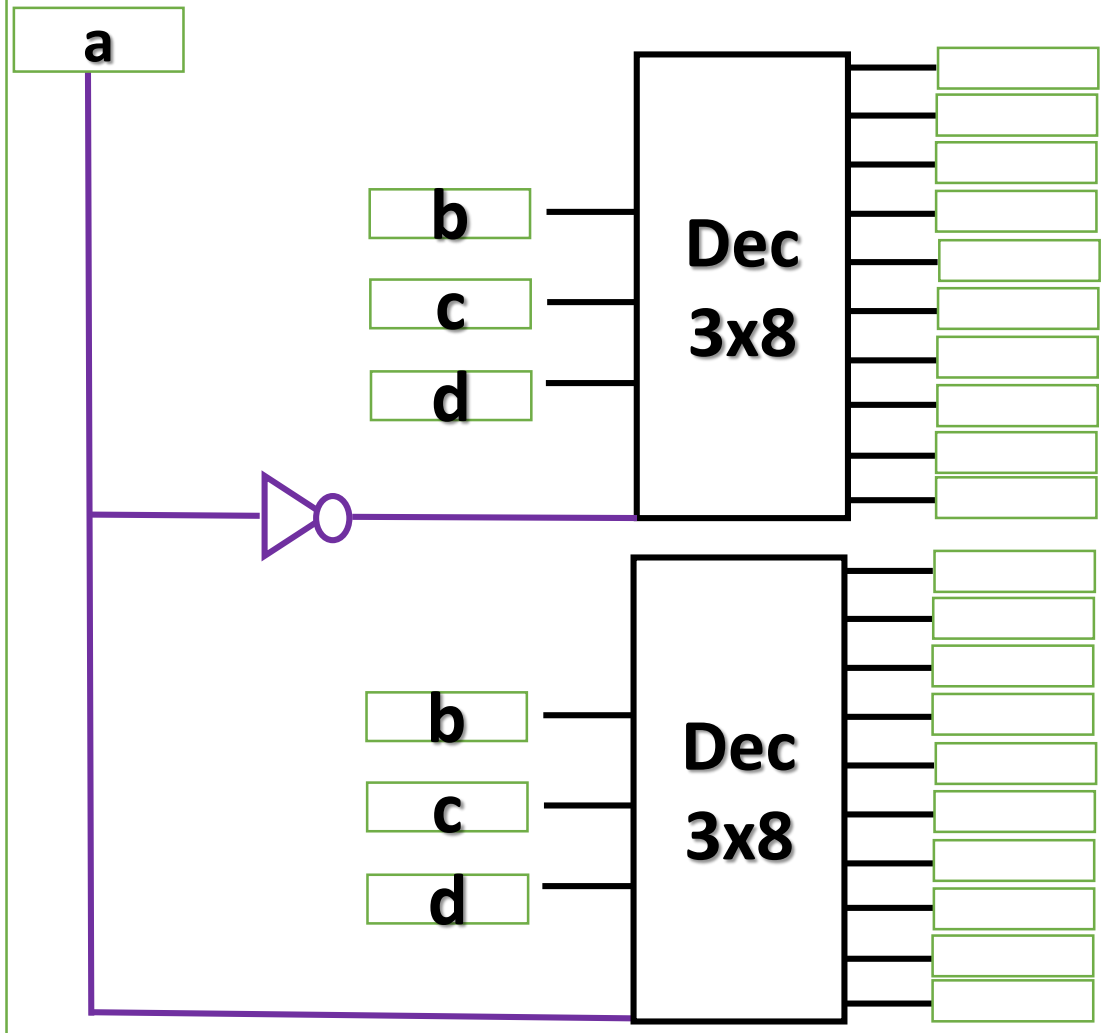
un décodeur 3x8 à l'aide de 2x4





Le décodeur

un décodeur 4x16 à l'aide de 3x8



Le DEC 4x16 aura comme entrées a b c d

On utilisera 2 DEC 3x8 qui auront comme entrées communes b c d

et 16 sorties (2x8)

a sera utilisé pour valider l'un ou l'autre des 2 DEC 3x8

$V1 = \bar{a}$ et $V2 = a$

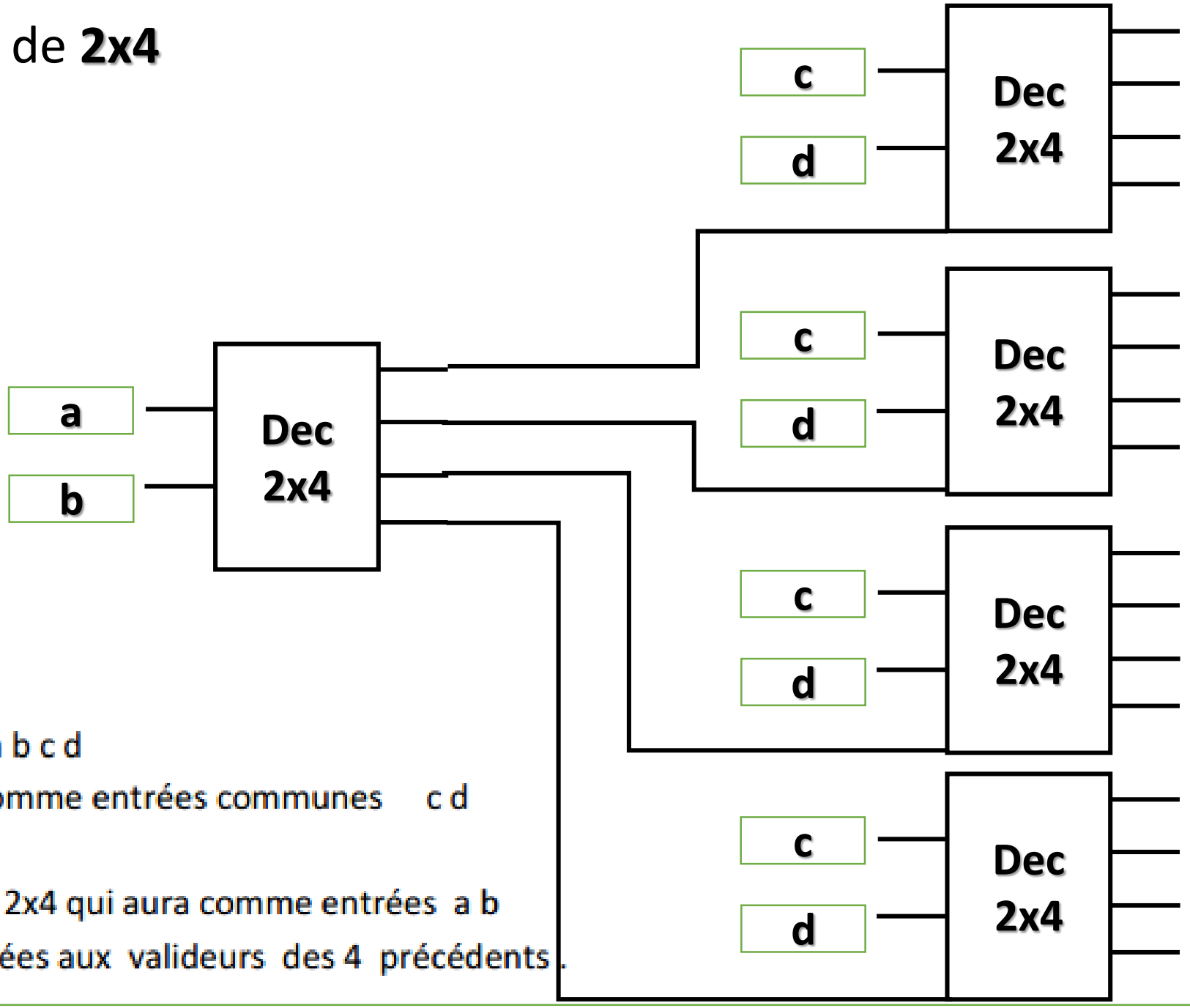
Si a = 0 alors V1 = 1 c'est le premier DEC qui sera actif

Si a = 1 alors V2 = 1 c'est le deuxième DEC qui sera actif

donc une seule sortie parmi les 16 sera égale à 1

(celle du DEC validé)

un décodeur **4x16** à l'aide de **2x4**



Le DEC 4x16 aura comme entrées a b c d

On utilisera 4 DEC 2x4 qui auront comme entrées communes c d

et 16 sorties (4x4)

On utilisera également un autre DEC 2x4 qui aura comme entrées a b

Les sorties de ce DEC seront connectées aux valideurs des 4 précédents.

Réaliser une fonction de n variables à l'aide d'un DEC p x 2^p

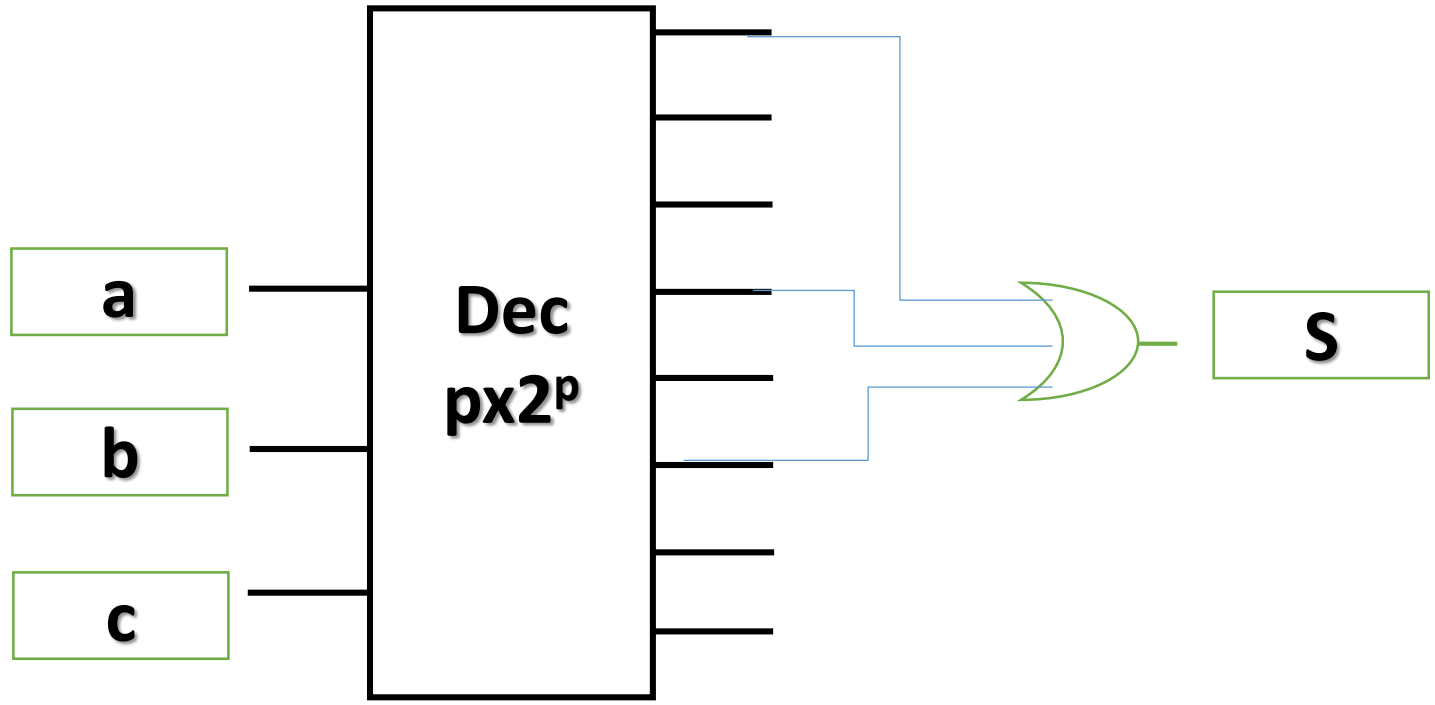
1) P = n

Chaque sortie du décodeur correspond à un minterme. On fera la somme logique **(OR)** de toutes les sorties du décodeur correspondant au min termes pour lesquels la fonction est égale à 1.

Exemple : n = 3 et P = 3

S(abc) = $\bar{a}\bar{b}\bar{c} + \bar{a}bc + a\bar{b}c$

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



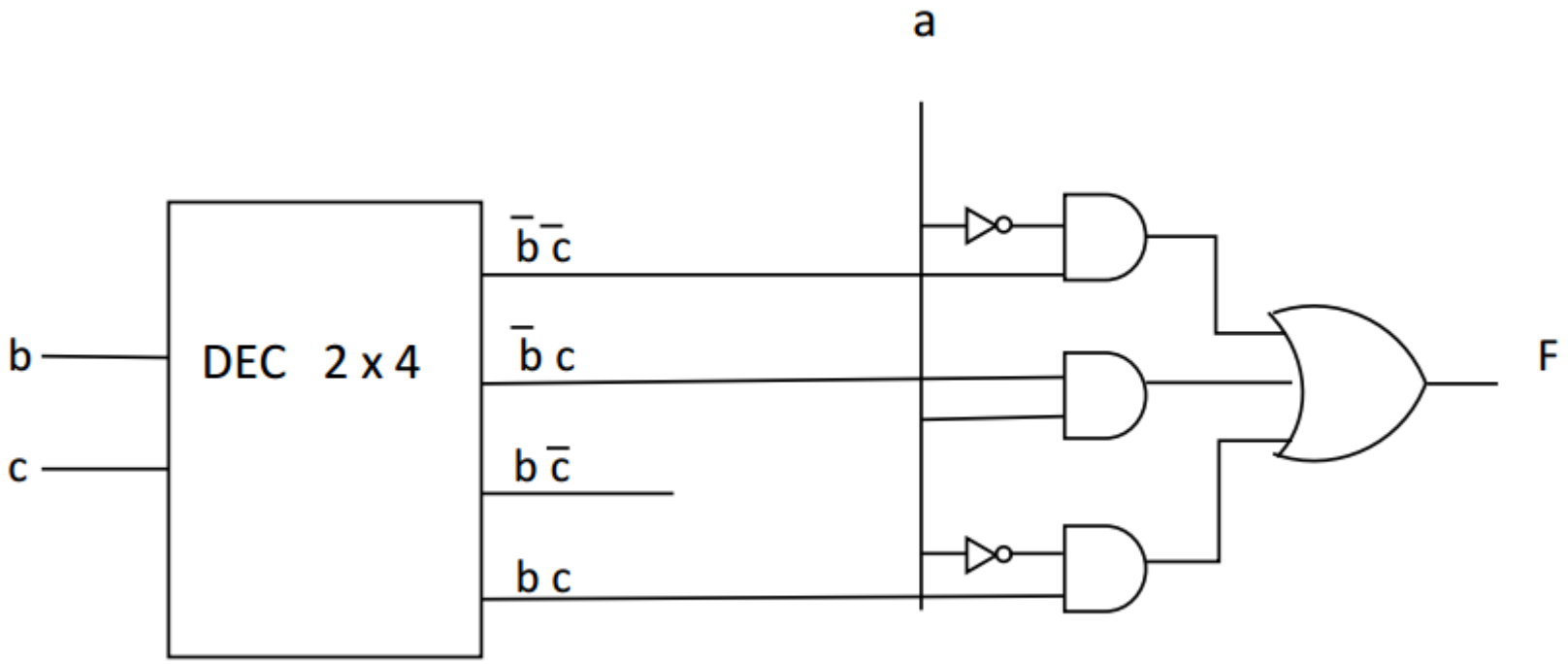
Réaliser une fonction de n variables à l'aide d'un DEC p x 2^p

2) p < n

P variables seront les entrées du décodeur. Les **(n-p)** variables restantes seront à l'extérieur du décodeur. La fonction sera formée par la combinaison des sorties du décodeur et des variables extérieures.

Exemple : **n = 3** et **P = 2**

F(abc) = $\bar{a}\bar{b}\bar{c} + \bar{a}bc + a\bar{b}c$



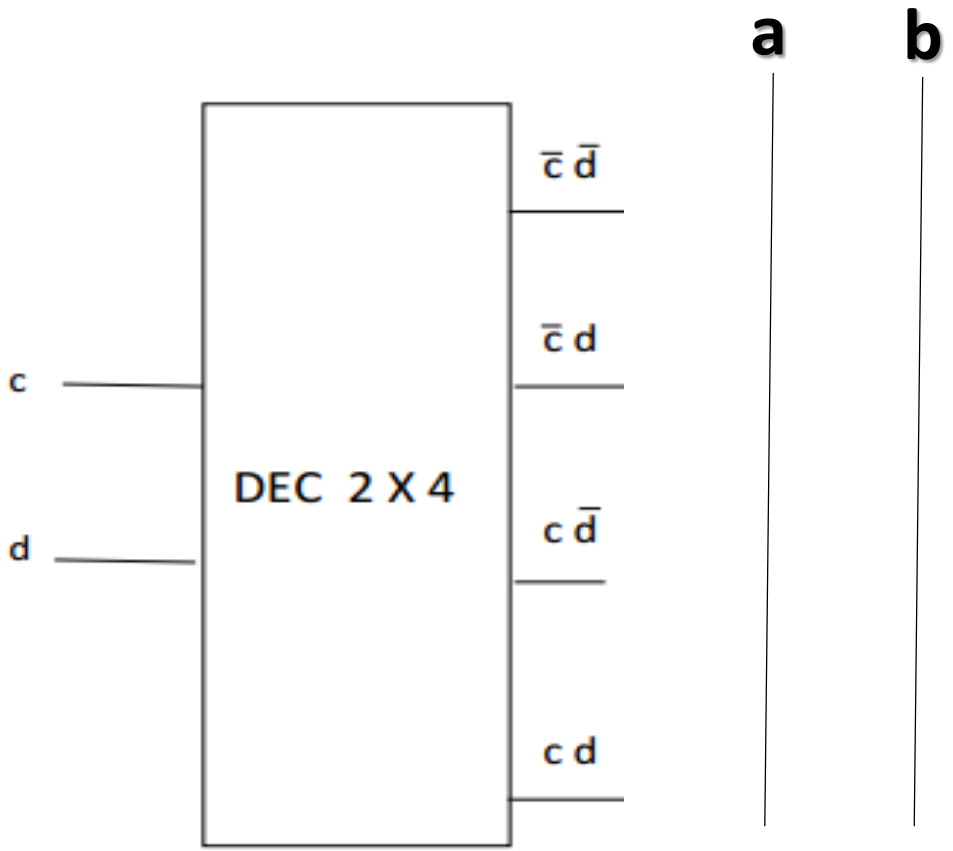
Réaliser une fonction de n variables à l'aide d'un DEC p x 2^p

Exemple 2 : **n = 4** et **p = 2**

F(abcd) = $a\bar{b}cd + \bar{a}bd(c + \bar{c}) + a\bar{c}(d + \bar{d})$

Si on sort les variables **a** et **b**, les entrées du **DEC 2 X 4** seront **c** et **d.**

c et **d** sont **indissociables**, Il faut donc faire apparaitre **c** et **d** dans chaque terme.



Réaliser une fonction de n variables à l'aide d'un DEC p x 2^p

Exemple 2 : **n = 3** et **P = 2**

F(abc) = $a\bar{b}cd + \bar{a}bd + a\bar{c}$

Si on sort les variables a et b, les entrées du DEC 2 X 4 seront c et d c et d sont **indissociables**,

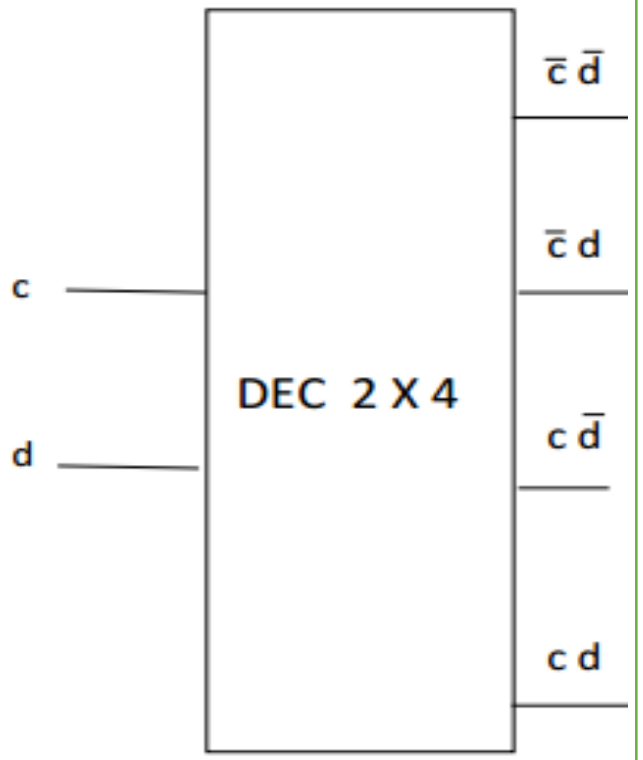
Il faut donc faire apparaitre c et d dans chaque terme.

$$F(abc) = a\,\bar{b}\,c\,d + \bar{a}\,b\,d(c + \bar{c}) + a\,\bar{c}\,(d + \bar{d})$$

$$F(abc) = a\,\bar{b}\,c\,d + \bar{a}\,b\,c\,d + \bar{a}\,b\,\bar{c}\,d + a\,\bar{c}\,d + a\,\bar{c}\,\bar{d}$$

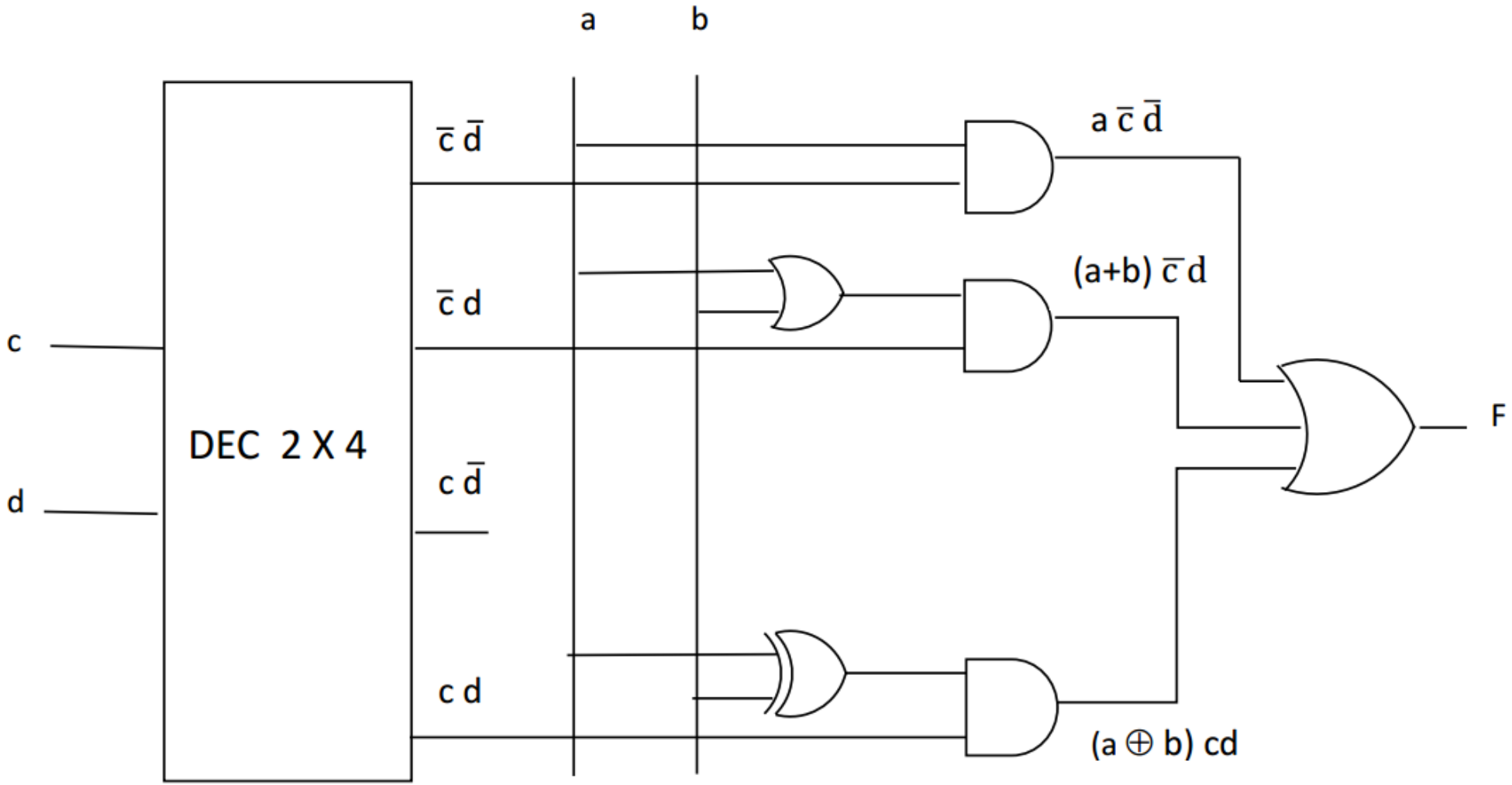
$$F(abc) = c\,d\,(a\,\bar{b} + \bar{a}\,b) + \bar{c}\,d\,\underbrace{(\bar{a}\,b + a)} + a\,\bar{c}\,\bar{d}$$

$$F(abc) = c\,d\,(a \oplus b) + \bar{c}\,d\,\underbrace{(a + b)} + a\,\bar{c}\,\bar{d}$$



Réaliser une fonction de n variables à l'aide d'un DEC p x 2^p

$$F(abc) = c\,d\,(a \oplus b) + \bar{c}\,d\,\underbrace{(a + b)} + a\,\bar{c}\,\bar{d}$$



Réaliser une fonction de n variables à l'aide d'un DEC p x 2^p

Exemple 3 : **n = 4 et p = 2**

Si un terme ne contient ni **c** ni **d** on utilise uniquement les variables extérieures Par exemple

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

Le premier terme (a b) ne contient ni **c** ni **d** donc on le laisse tel quel.

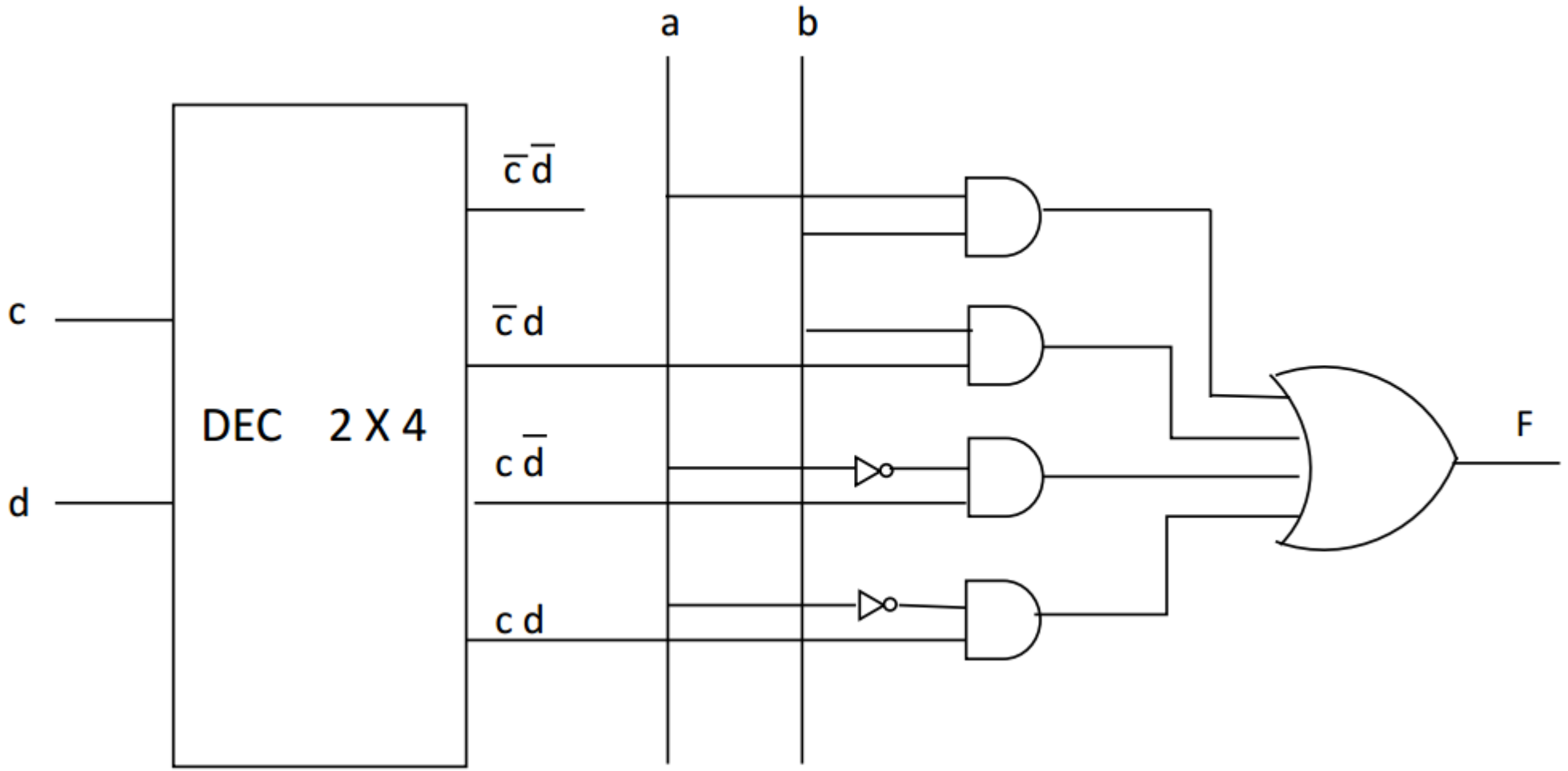
Le deuxième terme **(a c)** contient **c** mais pas **d** donc on le transforme :

/a c = /a c (**$\bar{d}$ + d**)

F(a b c d) = a b + b $\bar{c}$d + /a c **$\bar{d}$**+ /a c **d**

Réaliser une fonction de n variables à l'aide d'un DEC p x 2^p

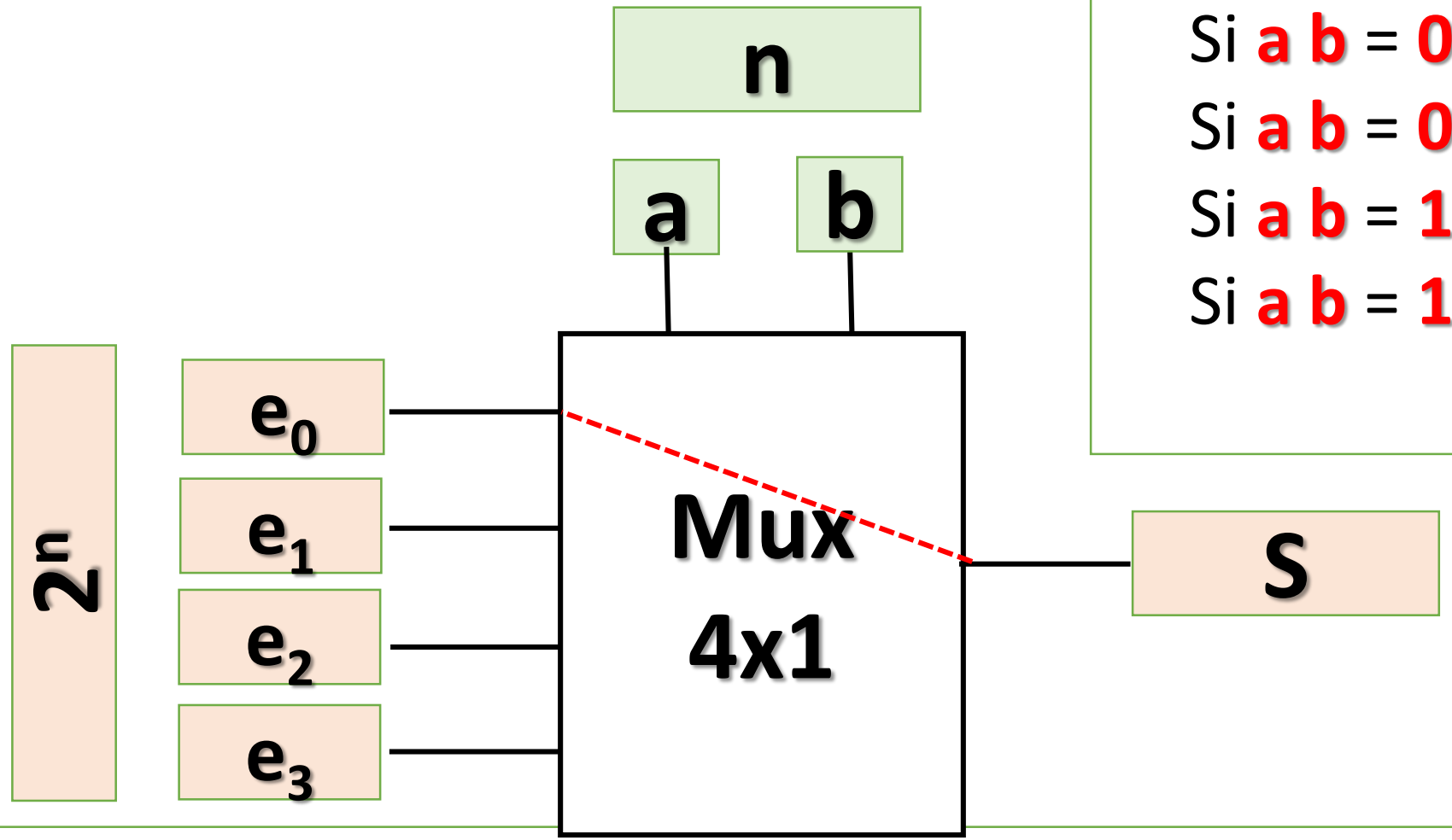
F(a b c d) = a b + b $\bar{c}$d + /a c $\bar{d}$+ /a c d



Le Multiplexeur

Un multiplexeur est un circuit combinatoire qui a 2^n entrées , une sortie et **n** lignes de sélection

L'exemple suivant représente un multiplexeur 4x1



Si a b = 0 0 S = e_0

Si a b = 0 1 S = e_1

Si a b = 1 0 S = e_2

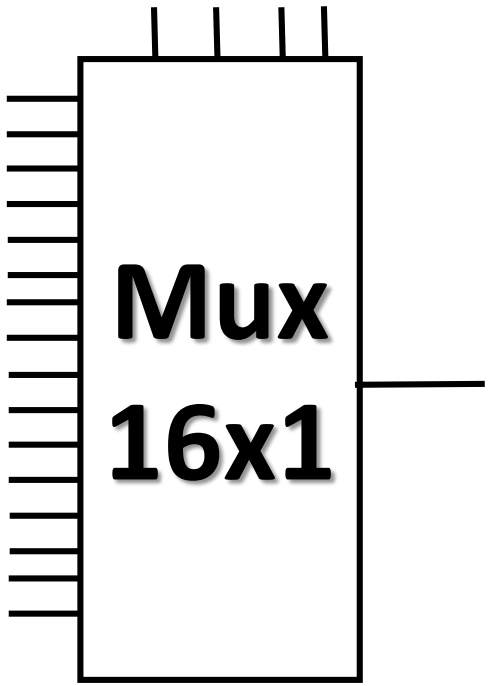
Si a b = 1 1 S = e_3

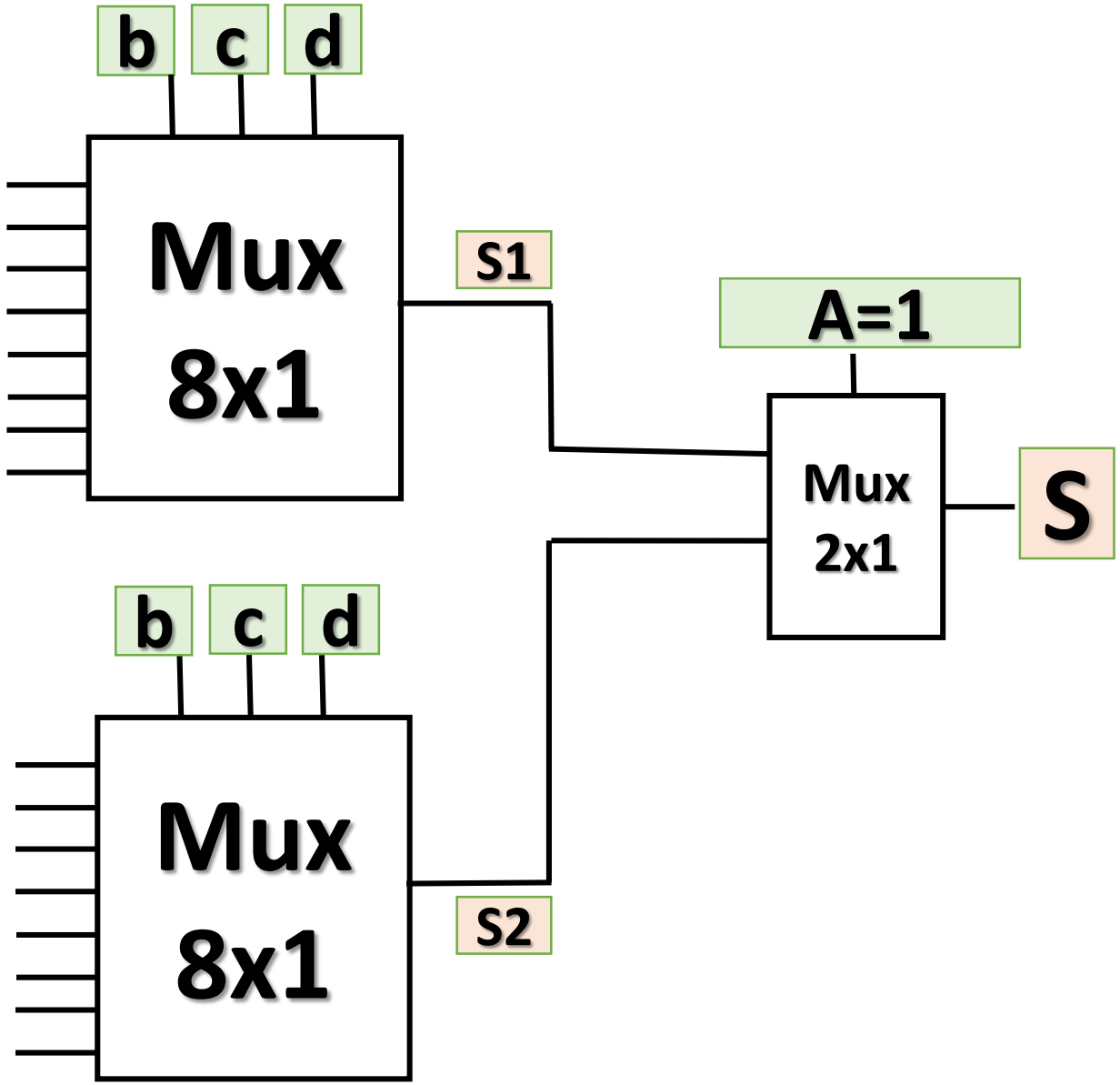
Réaliser un MUX 16x1(2^4x1) à l'aide de MUX 8x1(2^3x1)

Le MUX 16x1 doit avoir **16 entrées** , une sortie et **4 lignes de sélection a b c d** .

On utilisera **2 MUX 8x1** qui auront chacun une sortie et **3 lignes de sélection** communes **b c d** et un MUX **2x1** qui aura comme ligne de sélection **a**

Si **a = 0** alors **S = S1**
Si **a = 1** alors **S = S2**





Réaliser une fonction de n variables à l'aide d'un MUX 2^px1

1/ p = n

Chaque entrée du MUX représentera une valeur de la fonction

Exemple : n = 3 et P = 3

F(abc) = $\bar{a}\,\bar{b}\,\bar{c} + \bar{a}\,b\,c + a\,\bar{b}\,c$

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

$e_0 = 1$

$e_1 = 0$

$e_2 = 0$

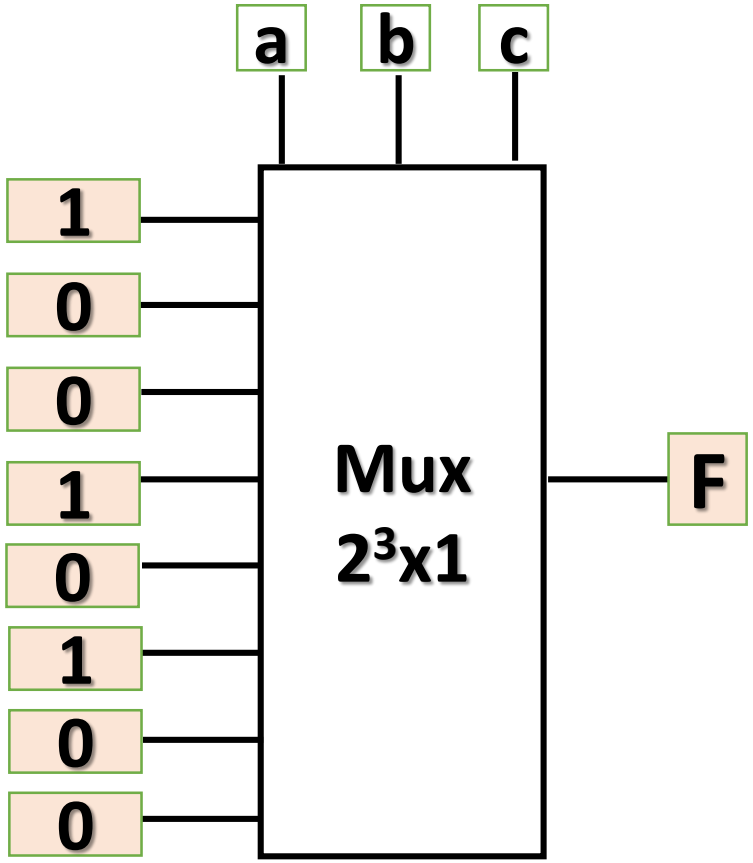
$e_3 = 1$

$e_4 = 0$

$e_5 = 1$

$e_6 = 0$

$e_7 = 0$



Réaliser une fonction de n variables à l'aide d'un MUX 2^px1

P < n

Exemple : N = 4 et p = 2

F(abcd) = ad + bd + cd + abc

On prendra c d comme lignes de sélection a b c d et a b resteront à l'extérieur

F(abcd) = ad(c+/c) + bd(c+/c) + cd + abc(d+/d)

F(abcd) = acd + a/cd + bcd + b/cd + cd + abcd + abc/d

F = cd(a+b+1+ab) + $\bar{c}$d(a+b) + c$\bar{d}$(ab)

F = cd + $\bar{c}$d(a+b) + c$\bar{d}$ (ab)