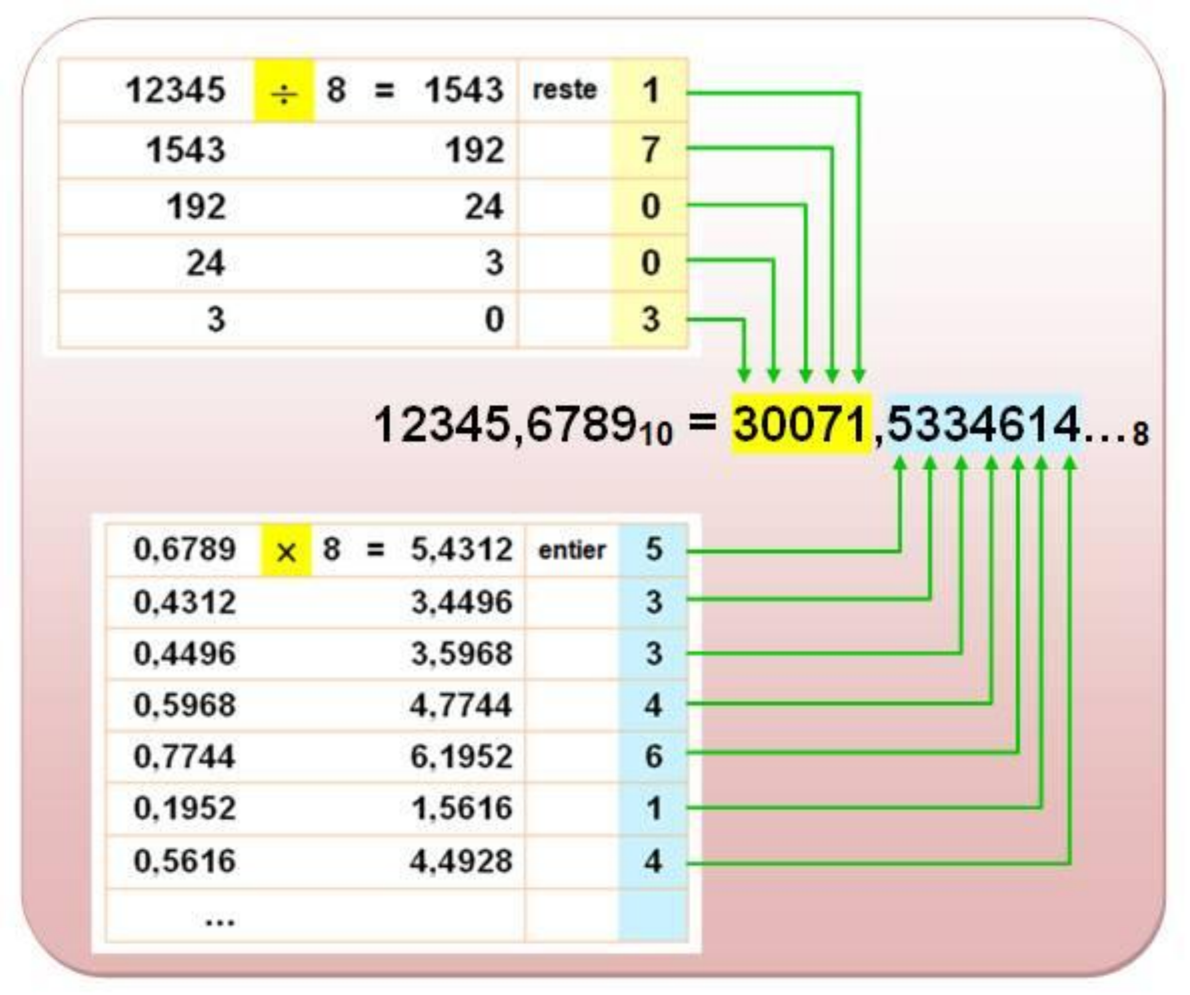
UNIVERSITÉ CHOUAIB DOUKKALI
Ecole Supérieure de Technologie
Sidi Bennour

Cours: Informatique Industrielle (suite-2)

Année, 2018-2019

Représentation des nombres « Codage »

Exemple: 12345, 6789 = ( )$_8$



12345	÷ 8 =	1543	reste	1
1543		192		7
192		24		0
24		3		0
3		0		3

$$12345,6789_{10} = 30071,5334614\ldots_8$$

0,6789	× 8 =	5,4312	entier	5
0,4312		3,4496		3
0,4496		3,5968		3
0,5968		4,7744		4
0,7744		6,1952		6
0,1952		1,5616		1
0,5616		4,4928		4
...				

Unité 1: Informatique Industrielle

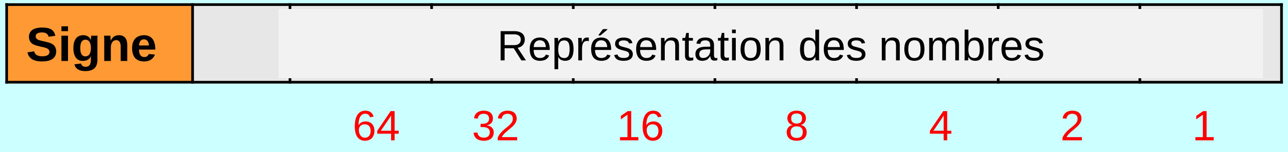
Représentation des nombres « Codage »

Représentation des nombres signés

Représentation module plus signe

Règle:

- On ajoute un bit de signe S : 0 pour le signe +, 1 pour le signe –
- Le bit de signe est placé en tête (bit de poids fort).



Exemple:

$N_{10} = + 4$; trouver N_2

$N_{10} = - 3$; trouver N_2

Représentation des nombres « Codage »

Nombre maximal et nombre minimal

- Si on utilise (n+1) bits (n pour le module et 1 pour le signe), on a alors :

$N_{max} = + (2^n-1)$

$N_{min} = - (2^n-1)$

- Exemple: n = 3 ; -7 <= N <= +7
- Avantage:

  - simplicité
- Inconvénients:

  - le zéro possède deux représentations.

  - Elle ne convient pas pour les opérations arithmétique.

Test: (- 1) + (+ 3) = + 2

(1001) + (0011) = 1100 (-→ - 4 ≠ + 2)

Nombre positifs:	Nombre négatifs:
0111 , 7	1111 , -7
0110 , 6	1110 , -6
0101 , 5	1101 , -5
0100 , 4	1100 , -4
0011 , 3	1011 , -3
0010 , 2	1010 , -2
0001 , 1	1001 , -1
0000 , 0	1000 , -0

Représentation des nombres « Codage »

Représentation avec le complément

- **Complément à 1 d'un nombre N**

Il est obtenu en inversant chaque bit de N. on le Note : $\overline{N}$

Exemple :

N = 100110 ; $\overline{N}$ = 011001

- **Complément à 2 d'un nombre N**

Il correspond à $\overline{N}$ + 1

Exemple : N = 100110 ; $\overline{N}$ + 1 = 011010

- **Règle:**

On a : (-N) = $\overline{N}$ + 1

On utilise cette règle pour représenter les nombres signés.

Unité 1: Informatique Industrielle

Représentation des nombres « Codage »

Exemple : n = 3 (signe non compris);

Le nombre 1000 existe aussi. Il correspond à (- 8).

Car: (- 8) + 1 = - 7

1000 + 0001 = 1001

- **Nombre maximal et nombre minimal**

Si on utilise (n + 1) bits

N_{max} **= + (2^n - 1)**

N_{min} **= - (2^n)**

Nombre positifs:	Nombre négatifs:
0111 , 7	, -7
0110 , 6	, -6
0101 , 5	, -5
0100 , 4	, -4
0011 , 3	1101 , -3
0010 , 2	1110 , -2
0001 , 1	1111 , -1
0000 , 0	0000 , -0

Remarques:

on a : - (- N) = N (pour N ≠ - 2^n)

cette représentation est souvent utiliser, car elle facilite le calcul arithmétique.

Opérations en binaire:

✓ Addition

- Demi-additionneur (HA)

Il fait l’addition de deux nombres à 1 bit (1BHA) et génère deux bits qui représentent la somme et la retenue (carry).

Schéma bloc du 1BHA

Unité 1: Informatique Industrielle

Représentation des nombres « Codage »

Table décrivant la fonction du 1BHA

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

$$S_0 = A_0 \oplus B_0$$

$$C_1 = A_0 . B_0$$

Additionneur complet (FA)

Il tient compte de la retenue (carry_in) provenant de la somme des chiffres de la colonne précédente.

Schéma bloc du FA à 1 bit

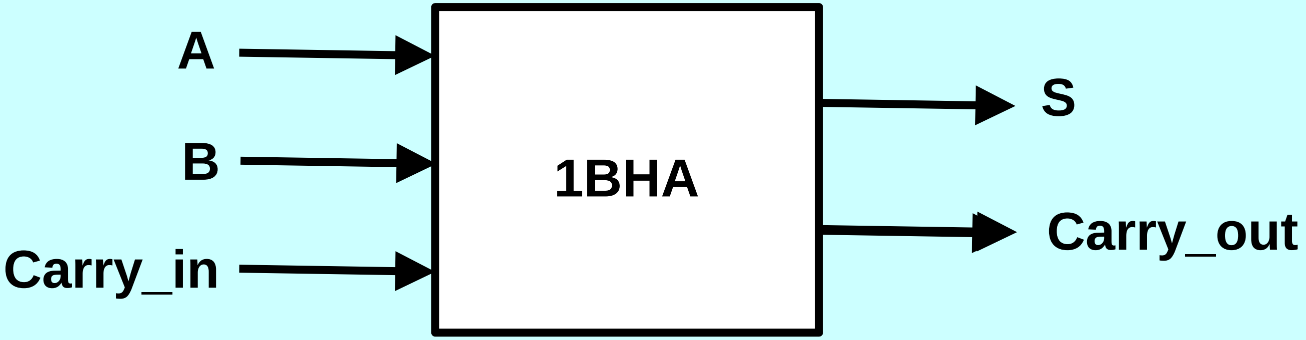


Table décrivant la fonction du 1BFA

A	B	C_in	S	C_out
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

$$\begin{aligned} C_{n+1} &= \overline{A_n}.B_n.C_n + A_n.\overline{B_n}.C_n + A_n.B_n.\overline{C_n} + A_n.B_n.C_n \\ &= (\overline{A_n}.B_n + A_n.\overline{B_n}).C_n + A_n.B_n.(\overline{C_n}.C_n) \\ &= A_n.B_n + C_n.(A_n \oplus B_n) \end{aligned}$$

$$\begin{aligned} S_n &= \overline{A_n}.\overline{B_n}.C_n + \overline{A_n}.B_n.\overline{C_n} + A_n.\overline{B_n}.\overline{C_n} + A_n.B_n.C_n \\ &= \overline{A_n}.(B_n \oplus C_n) + A_n.\left(\overline{B_n \oplus C_n}\right) \\ &= A_n \oplus B_n \oplus C_n \end{aligned}$$

Représentation des nombres « Codage »

Addition de deux nombres quelconques

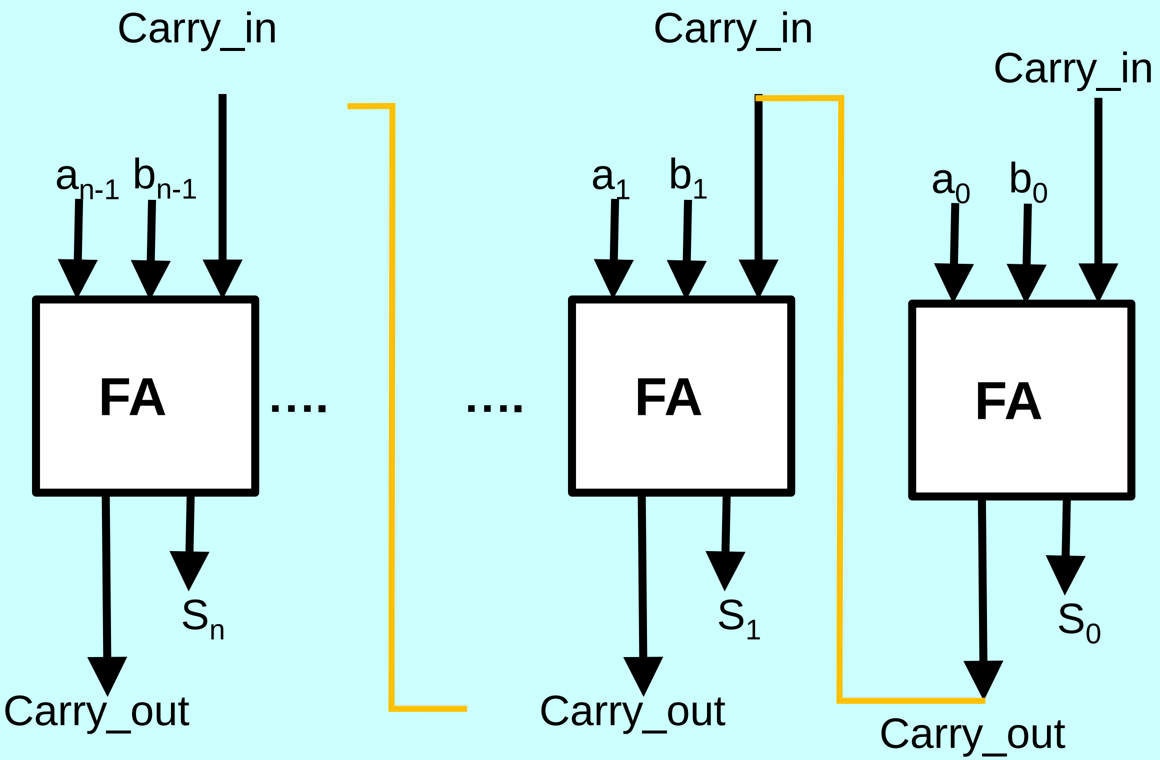
$$S = A + B \; ; \; A = a_{n-1} \, ... \, a_1 a_0 \; ; \; B = b_{n-1} \, ... \, b_1 b_0$$

Méthode

La méthode est semblable a celle du système décimal.

Exemple : 1011101 + 110010,101

Circuit : il faut mettre n FA en cascade.



Unité 1: Informatique Industrielle

Représentation des nombres « Codage »

Soustraction

- **Demi-soustracteur (HS)**

Il fait la soustracteur de deux nombre à 1 bit (1 BHS) et génère deux bits qui représentent la somme et la retenue (carry).

Schéma bloc du 1BHA

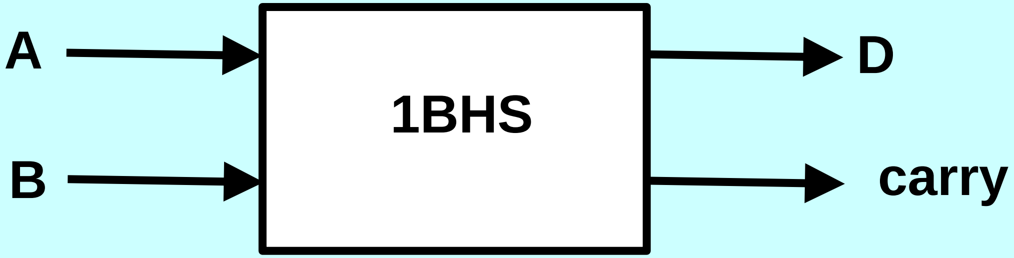


Table décrivant la fonction du 1BHS

$$D_0 = A_0 \oplus B_0$$

$$C_1 = /A_0.B_0$$

A	B	D	Carry
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Soustracteur complet (FS)

Il tient compte de la retenue (carry_in) provenant de la soustraction des chiffres de la colonne précédente.

Schéma bloc du FS à 1 bit

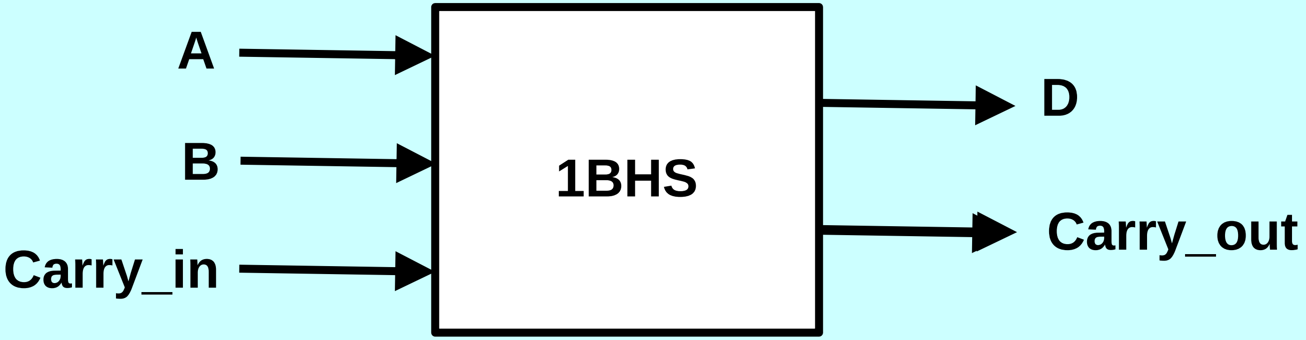


Table décrivant la fonction du 1BFS

A	B	C_in	D	C_out
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

$$D_n = A_n \oplus B_n \oplus C_n$$

$$C_{n+1} = \overline{A_n}.B_n + (\overline{A_n \oplus B_n}).C_n$$

Représentation des nombres « Codage »

Soustraction de deux nombres quelconques

S = A - B ; A = a_{n-1} ….. a_1a_0 ; B = b_{n-1} ….. b_1b_0

A et B sont représentés en virgule fixe. Ils peuvent être des nombres fractionnaires et / ou signés.

Méthode

A et B sont représentés en virgule fixe. La méthode est semblable a celle du système décimal.

Exemple : 1011101 + 110010,101

Circuit : il faut mettre n FA en cascade.

Représentation des nombres « Codage »

Code Binaires

- Il y a deux représentations possible d'une grandeur:
  - **Analogique**
  - **Numérique**

Pour présenter numériquement un nombre on utilise un **code**.

Coder un nombre ou symbole en binaire consiste à lui faire correspondre une **suite de 0 et 1 appelée Code**. Une suite de 0 et 1 s'appelle aussi combinaison binaire.

- ✓ Combinaison binaire à 4 bits = nibble
- ✓ Combinaison binaire à 8 bits = byte ou octet
- ✓ Combinaison binaire à 16 bits = Word ou mot
- ✓ Combinaison binaire à n bits = String ou chaîne

Représentation des nombres « Codage »

Code Binaire pur

Appelé aussi code binaire naturel ou code binaire positionnel. Il correspond à l’écriture du nombre à code en base 2.

Exemple : 49 → 110001

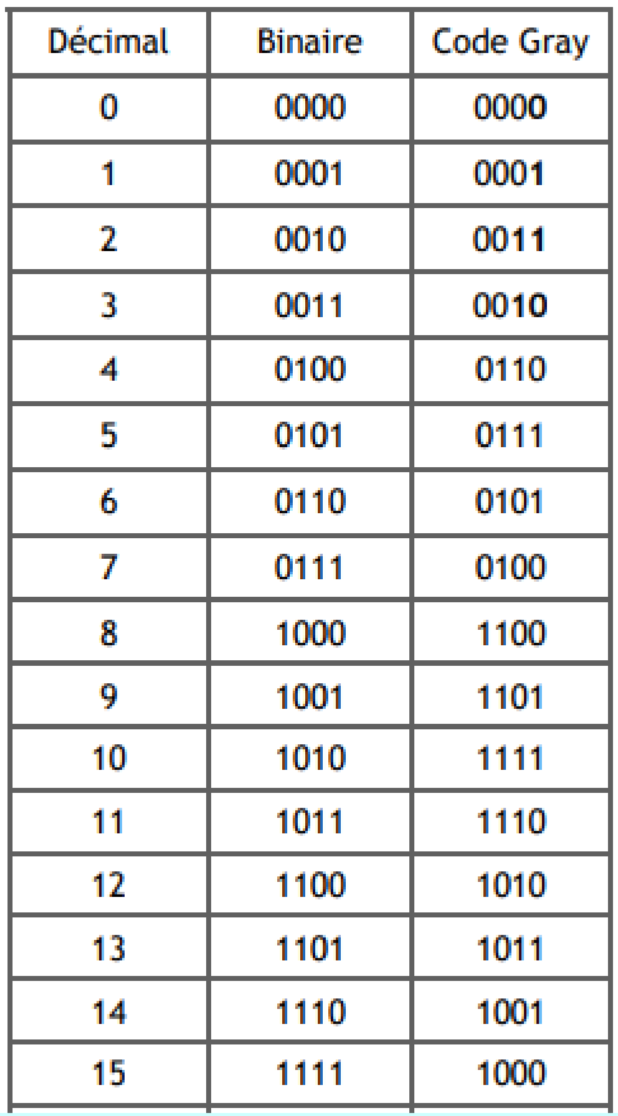
Représentation des nombres « Codage »

Code à distance unité

Quand on passe d'une combinaison à la combinaison suivante, le code ne change que d'un bit.

Exemple : le code le plus connu: **Code de GRAY**

Le code GRAY est aussi utilisé dans l'écriture des tableaux de Karnaugh (c'est pour plus tard…)



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

Les autres codes.

- **Le code ASCII**

Il tire son appellation de l'abréviation américaine : **American Standard Code Interchange Information**.

C'est un code qui permet la représentation des caractères alphanumériques d'un micro-ordinateur, chaque caractère étant codé par un mot de 8 bits appelé **octet** . Ce code est très répandu dans le milieu de la micro-informatique.

Unité 1: Informatique Industrielle

Représentation des nombres « Codage »

Table des caractères ASCII

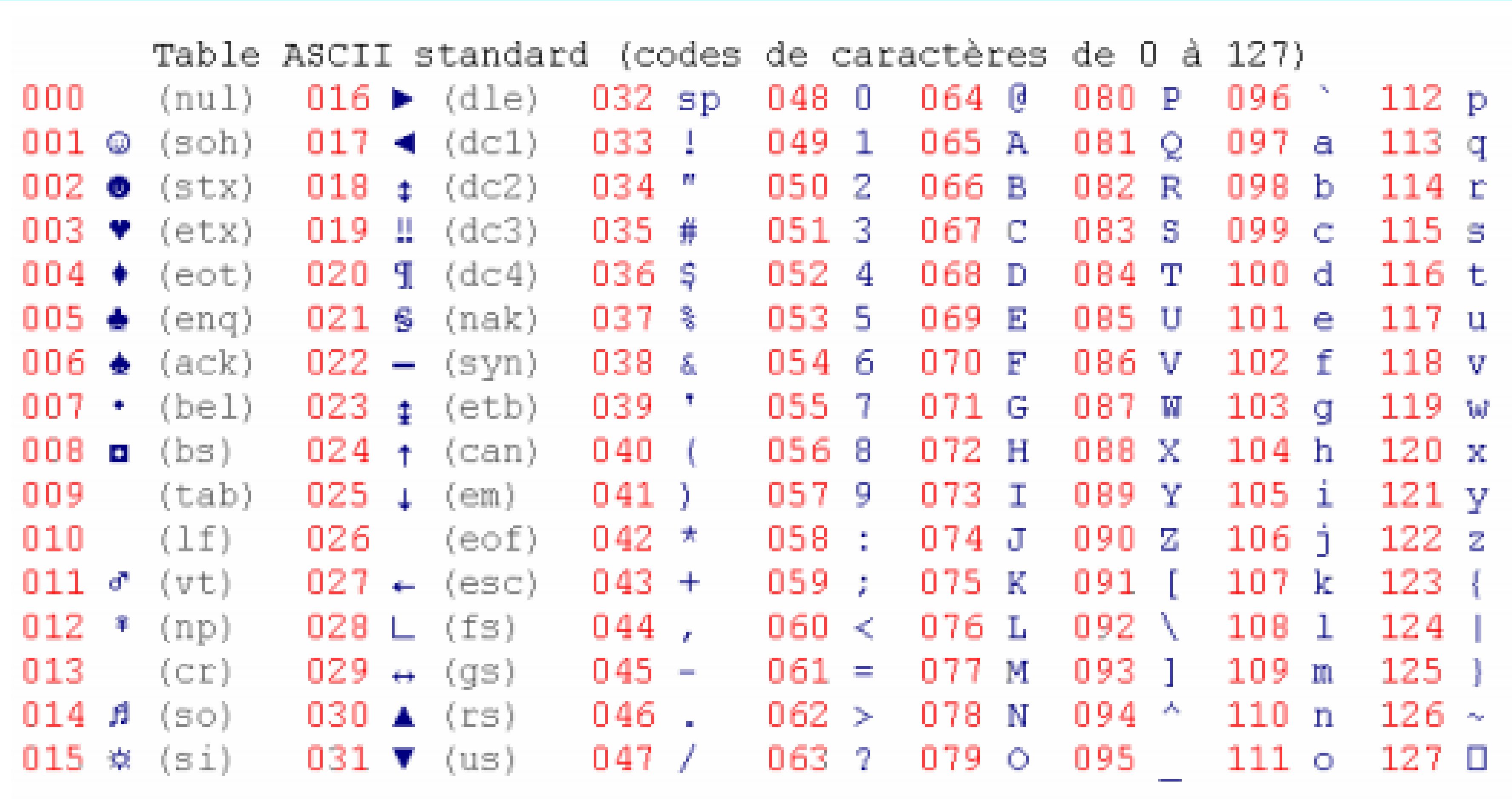


Table ASCII standard (codes de caractères de 0 à 127)

000		(nul)	016	►	(dle)	032	sp	048	0	064	@	080	P	096	`	112	p
001	☺	(soh)	017	◄	(dc1)	033	!	049	1	065	A	081	Q	097	a	113	q
002	☻	(stx)	018	↕	(dc2)	034	"	050	2	066	B	082	R	098	b	114	r
003	♥	(etx)	019	‼	(dc3)	035	#	051	3	067	C	083	S	099	c	115	s
004	♦	(eot)	020	¶	(dc4)	036	$	052	4	068	D	084	T	100	d	116	t
005	♣	(enq)	021	§	(nak)	037	%	053	5	069	E	085	U	101	e	117	u
006	♠	(ack)	022	▬	(syn)	038	&	054	6	070	F	086	V	102	f	118	v
007	•	(bel)	023	↨	(etb)	039	'	055	7	071	G	087	W	103	g	119	w
008	◘	(bs)	024	↑	(can)	040	(	056	8	072	H	088	X	104	h	120	x
009		(tab)	025	↓	(em)	041	)	057	9	073	I	089	Y	105	i	121	y
010		(lf)	026		(eof)	042	*	058	:	074	J	090	Z	106	j	122	z
011	♂	(vt)	027	←	(esc)	043	+	059	;	075	K	091	[	107	k	123	{
012	♀	(np)	028	∟	(fs)	044	,	060	<	076	L	092	\	108	l	124	|
013		(cr)	029	↔	(gs)	045	-	061	=	077	M	093	]	109	m	125	}
014	♫	(so)	030	▲	(rs)	046	.	062	>	078	N	094	^	110	n	126	~
015	☼	(si)	031	▼	(us)	047	/	063	?	079	O	095	_	111	o	127	□

Unité 1: Informatique Industrielle

Représentation des nombres « Codage »

Le Code Barre

Ce principe de codage, apparu dans les années 80, est largement utilisé sur les produits de grande consommation, car il facilite la gestion des produits. Le marquage comporte un certain nombre de barres verticales ainsi que 13 chiffres :

- Le 1er chiffre désigne le pays d'origine : 3 = France, 4 = Allemagne, 0 = U.S.A, Canada etc …
- Les cinq suivants sont ceux du code « fabricant »,
- Les six autres sont ceux du code de l'article,
- Le dernier étant une clé de contrôle Les barres représentent le codage de ces chiffres sur 7 bits , à chaque chiffre est attribué un ensemble de 7 espace blancs ou noirs.

3 222471 836552

Expression d’une fonction logique

Il existe deux méthodes pour exprimer une fonction logique :

- ✓ soit donner directement son équation logique.

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

- ✓ soit utiliser une table de vérité.

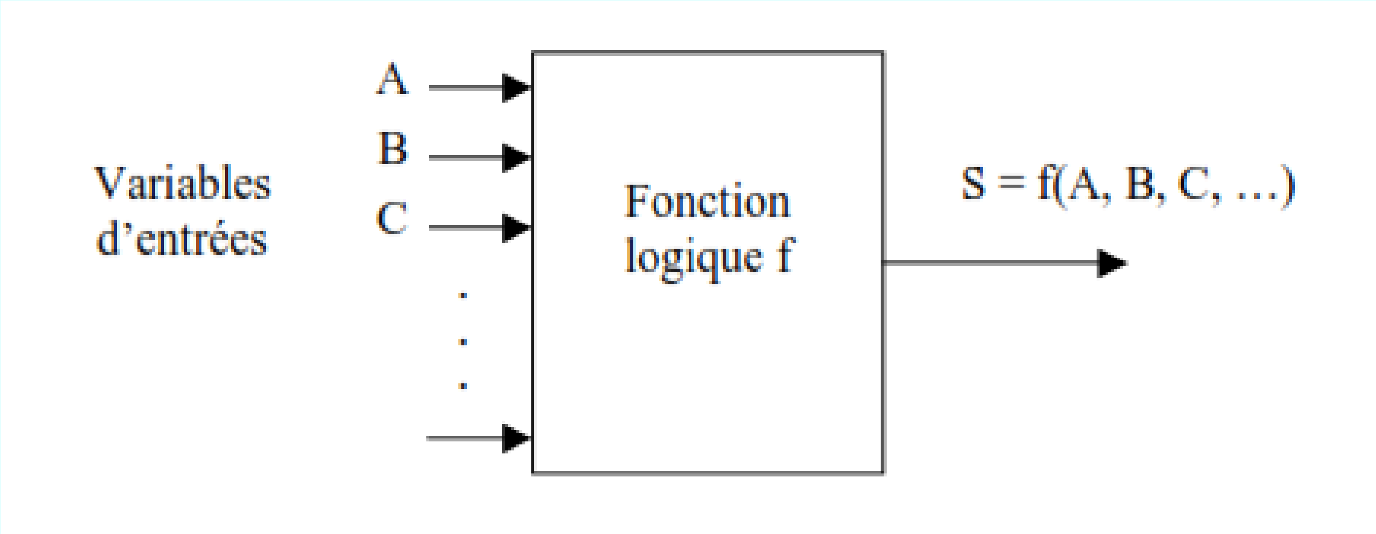


Table de vérité

Sous sa forme complète, l’équation logique de S se lit directement. C’est la somme du ET

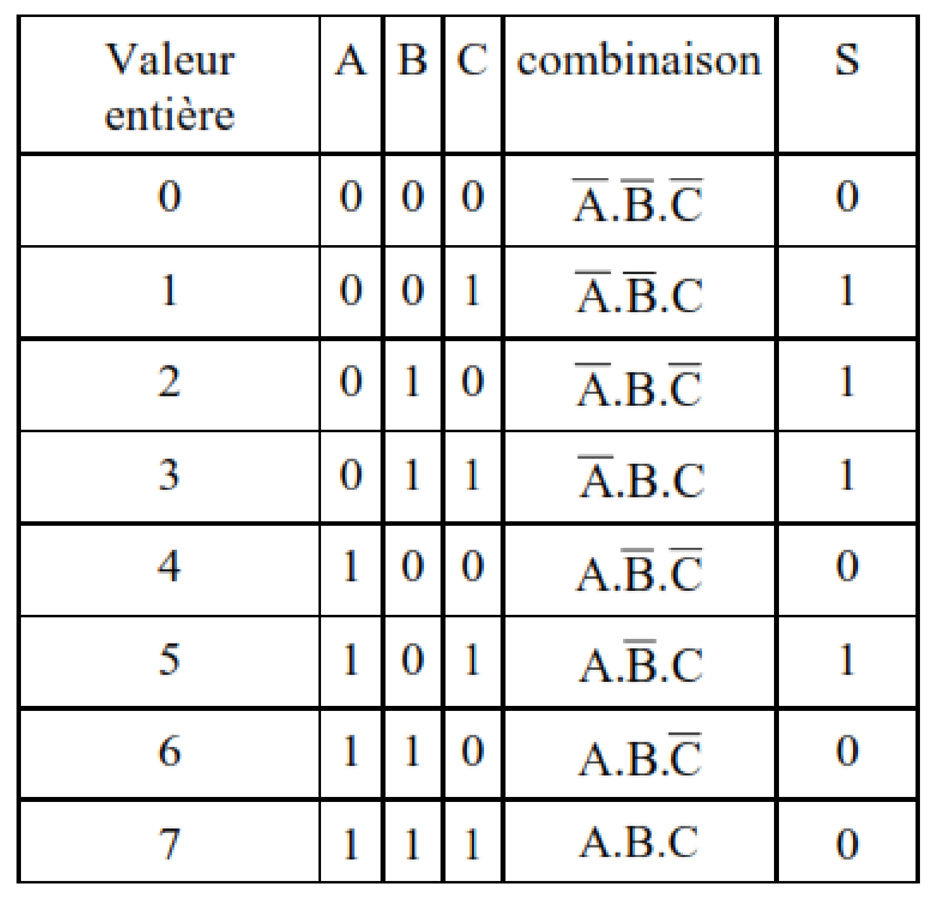
logique de chaque combinaison avec l’état de S correspondant. Ici, on obtient la forme

canonique complète :

$$S = \overline{A}.\overline{B}.\overline{C}.0 + \overline{A}.\overline{B}.C.1 + \overline{A}.B.\overline{C}.1 + \overline{A}.B.C.1 + A.\overline{B}.\overline{C}.0 + A.\overline{B}.C.1 + A.B.\overline{C}.0 + A.B.C.0$$

$$S = \overline{A}.\overline{B}.C.1 + \overline{A}.B.\overline{C}.1 + \overline{A}.B.C.1 + A.\overline{B}.C.1$$

C’est simplement la somme de combinaisons pour lesquelles S vaut **1**.



Valeur entière	A	B	C	combinaison	S
0	0	0	0	$\overline{A}.\overline{B}.\overline{C}$	0
1	0	0	1	$\overline{A}.\overline{B}.C$	1
2	0	1	0	$\overline{A}.B.\overline{C}$	1
3	0	1	1	$\overline{A}.B.C$	1
4	1	0	0	$A.\overline{B}.\overline{C}$	0
5	1	0	1	$A.\overline{B}.C$	1
6	1	1	0	$A.B.\overline{C}$	0
7	1	1	1	$A.B.C$	0

Simplification des fonctions logiques

Objectif : Fabriquer un système

- ✓ **à moindre coût**
- ✓ **rapide**
- ✓ **fiable**
- ✓ **peu consommateur**

L’expression d’une fonction logique sous sa forme la plus simple n’est pas quelque chose d’évident. Trois méthodes sont utilisables :

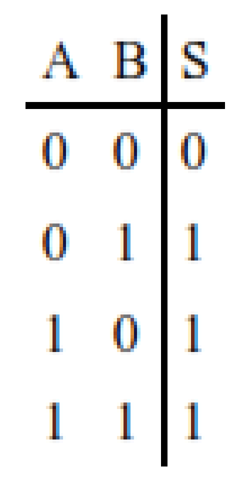
- **Le raisonnement**. On cherche, à partir du problème à résoudre, l’expression la plus simple possible. Evidemment, cette méthode ne garantit pas un résultat optimal.
- **La table de vérité** et **les propriétés de l’algèbre de BOOLE**. C’est plus efficace, mais il est facile de rater une simplification, notamment quand la fonction est compliquée.
- **La méthode graphique des tableaux de Karnaugh**. C’est la méthode la plus efficace car elle garantit le bon résultat. Cette méthode est utilisée sous forme informatique dans tous les outils de CAO.

Simplification algébrique

- On a deux voyants A et B. On veut déclencher une alarme quand au moins un des deux voyants est allumé, c’est-à-dire : soit A allumé avec B éteint, soit B allumé avec A éteint, soit A et B allumés.

Par le raisonnement, le lecteur attentif aura trouvé que **S = A + B**.

Ecrivons la table de vérité :



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

On obtient donc :

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

D’où on tire (d’après les propriétés de l’algèbre de BOOLE) :

$$S = A(\overline{B} + B) + B\overline{A} = A.1 + B\overline{A} = A + B\overline{A} = A + B$$

On voit bien que cette méthode de simplification devient peu évidente quand la fonction est plus complexe. Voyons maintenant une méthode plus efficace.

Simplification par les tableaux de Karnaugh

Cette méthode permet :

- ✓ d’avoir pratiquement l’expression logique la plus simple pour une fonction F.
- ✓ de trouver des termes communs pour un système à plusieurs sorties, dans le but de limiter le nombre de portes.

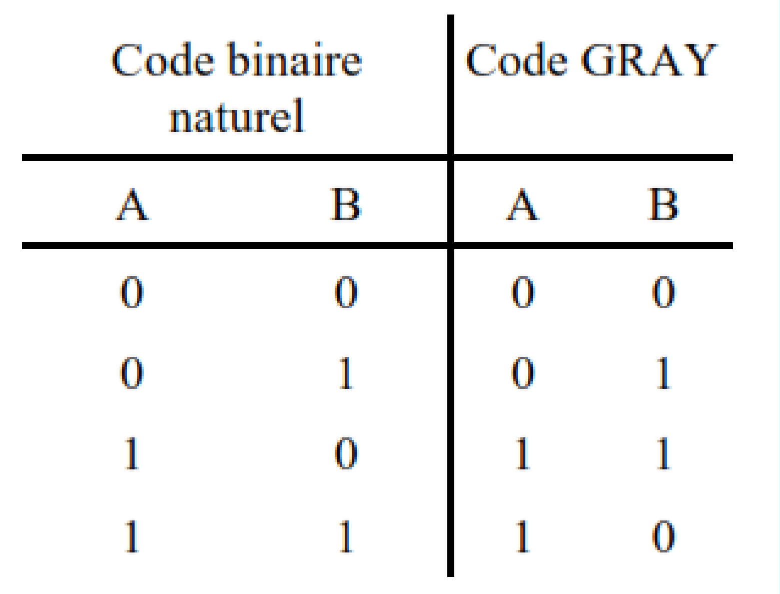
- ❖ A la main, on traite 4 variables sans difficultés. Au maximum, on peut aller jusqu’à 5 voir 6 variables.
- ❖ En pratique, on utilise un programme informatique qui implémente généralement l’algorithme de **QUINE-McCLUSKEY**, qui reprend la méthode de Karnaugh, mais de manière systématique et non visuelle.

Simplification par les tableaux de Karnaugh

Définition :

dans un code adjacent, seul un bit change d’une valeur à la valeur suivante.

Exemple avec deux variables :



Code binaire naturel		Code GRAY	
A	B	A	B
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Simplification par les tableaux de Karnaugh

Définition :

Les tableaux de Karnaugh sont une variante des tables de vérité. Ils sont organisés de telle façon que les termes à 1 (ou à 0) adjacents soient systématiquement regroupés dans des cases voisines, donc faciles à identifier visuellement. En effet, deux termes adjacents (qui ne différent que par une variable) peuvent se simplifier facilement :

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

On représente les valeurs de la fonction dans un tableau aussi carré que possible, dans lequel chaque ligne et chaque colonne correspondent à une combinaison des variables d’entrée exprimée avec un code adjacent (le code GRAY en général).

Simplification par les tableaux de Karnaugh

Règle:

Pour la lecture et la simplification de l’expression, on cherche des paquets les plus gros possibles (de 1, 2, 4 ou 8 variables), en se rappelant que le code est aussi adjacent sur les bords (bord supérieur avec bord inférieur, bord gauche avec bord droit).

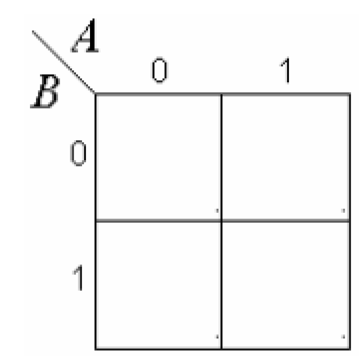
On effectue une lecture par « intersection » en recherchant la ou les variables ne changeant pas pour le paquet.

On ajoute alors les paquets. On obtient l’expression sous la forme d’une somme de produit.

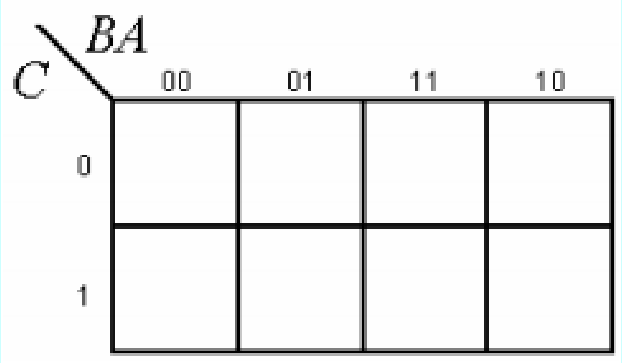
Une case peut être reprise dans plusieurs paquets.

Diagrammes de Karnaugh

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



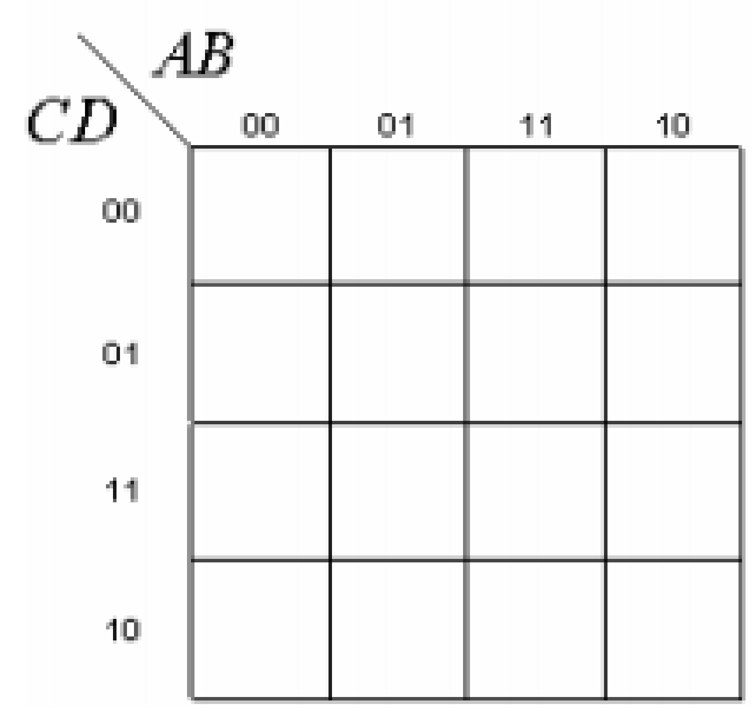
- Avec n = 3:
  - ✓ Entrées A, B et C
  - ✓ 8 cases



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

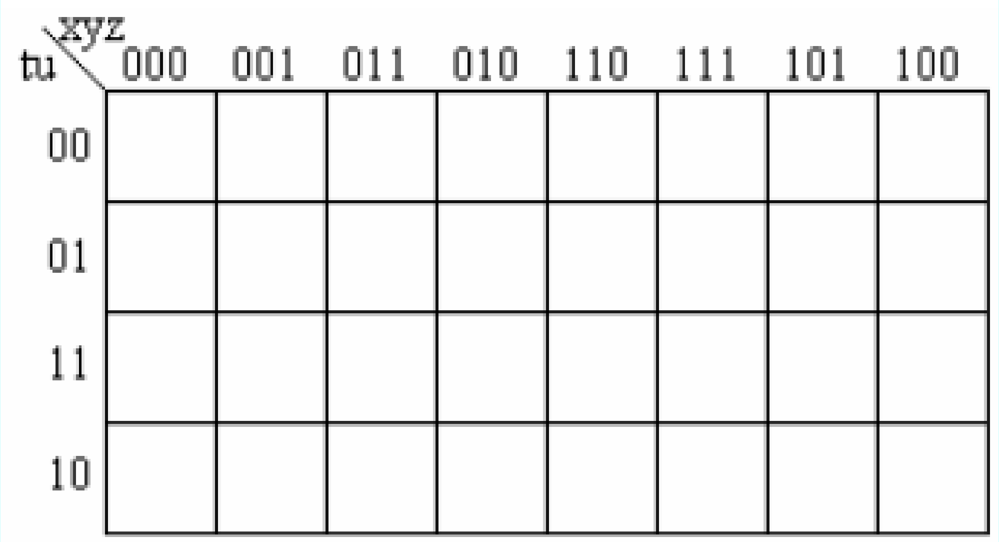
Diagrammes de Karnaugh

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



CD \ AB	00	01	11	10
00				
01				
11				
10				

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

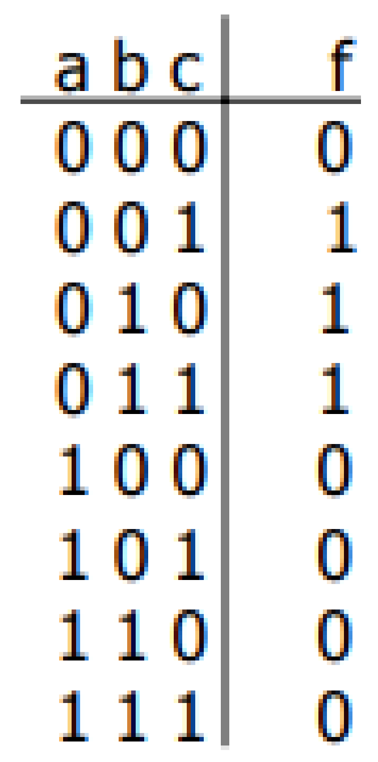


tu \ xyz	000	001	011	010	110	111	101	100
00								
01								
11								
10								

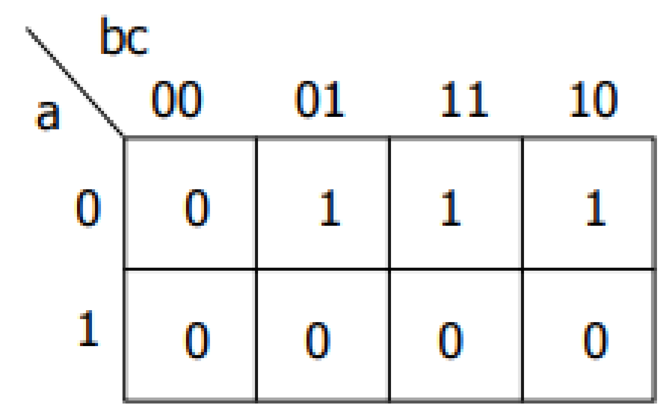
Remarque: Une seule variable change d’état entre 2 cases adjacentes

Simplification graphique

Exemple: Depuis une table de vérité:



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

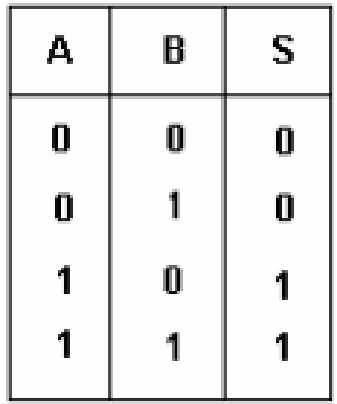


a \ bc	00	01	11	10
0	0	1	1	1
1	0	0	0	0

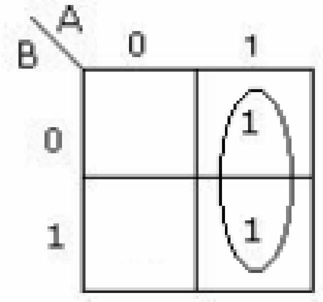
Règles de simplification

1) : Les groupements comportent une puissance de deux cases,
2) : Les 2k cases forment un rectangle,
3) : On élimine variable(s) qui change(nt) d'état Groupement de 2^k cases --→ On élimine k variables.
    - **A. 2 cases --→ on élimine 1 variable;**
    - **B. 4 cases --→ on élimine 2 variables;**
    - **C. 8 cases --→ on élimine 3 variables;**
4) : Il faut utiliser au moins une fois chaque 1, le résultat est donné par la réunion logique de chaque groupement,
5) : Expression minimale si :
    - les groupements les plus grands possibles
    - utiliser les 1 un minimum de fois

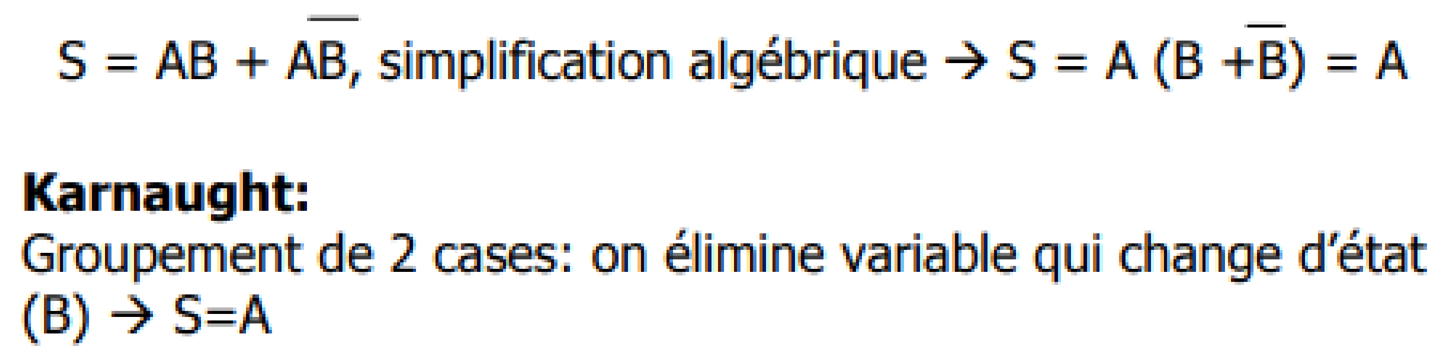
Exemple 1



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



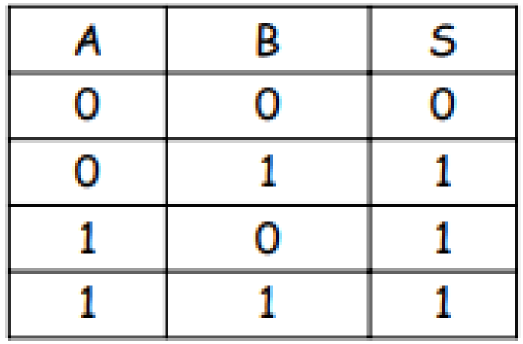
B \ A	0	1
0		1
1		1



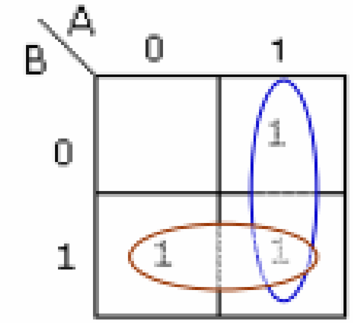
$S = AB + A\overline{B}$, simplification algébrique → $S = A\,(B + \overline{B}) = A$

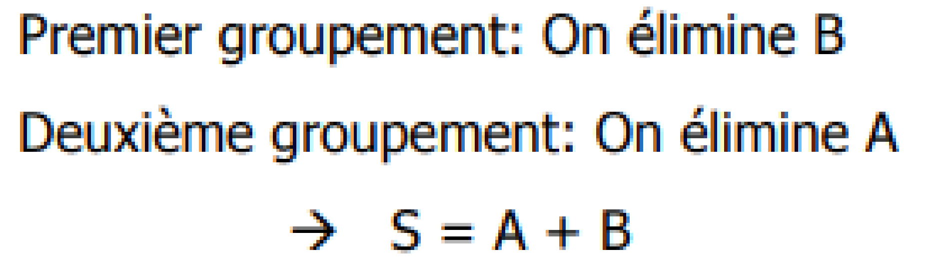
Karnaught:
Groupement de 2 cases: on élimine variable qui change d'état (B) → S=A

Exemple 2



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





Premier groupement: On élimine B

Deuxième groupement: On élimine A

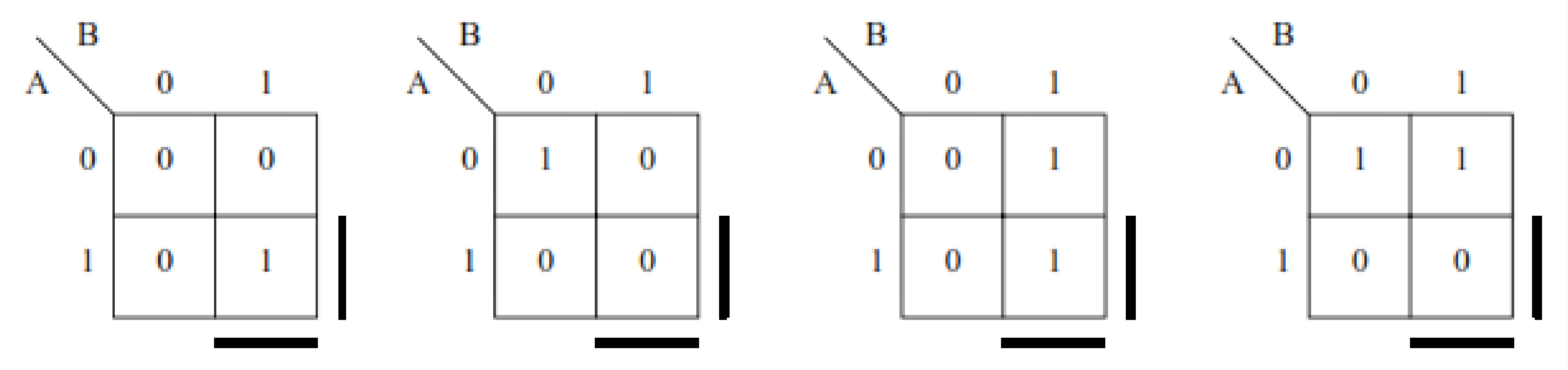
→ S = A + B

Exemple 3 : deux variables d'entrée A et B

1 case = produit de 2 variables

2 cases = 1 variable

4 cases = 1 ou 0 Si



A \ B	0	1
0	0	0
1	0	1

A \ B	0	1
0	1	0
1	0	0

A \ B	0	1
0	0	1
1	0	1

A \ B	0	1
0	1	1
1	0	0

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

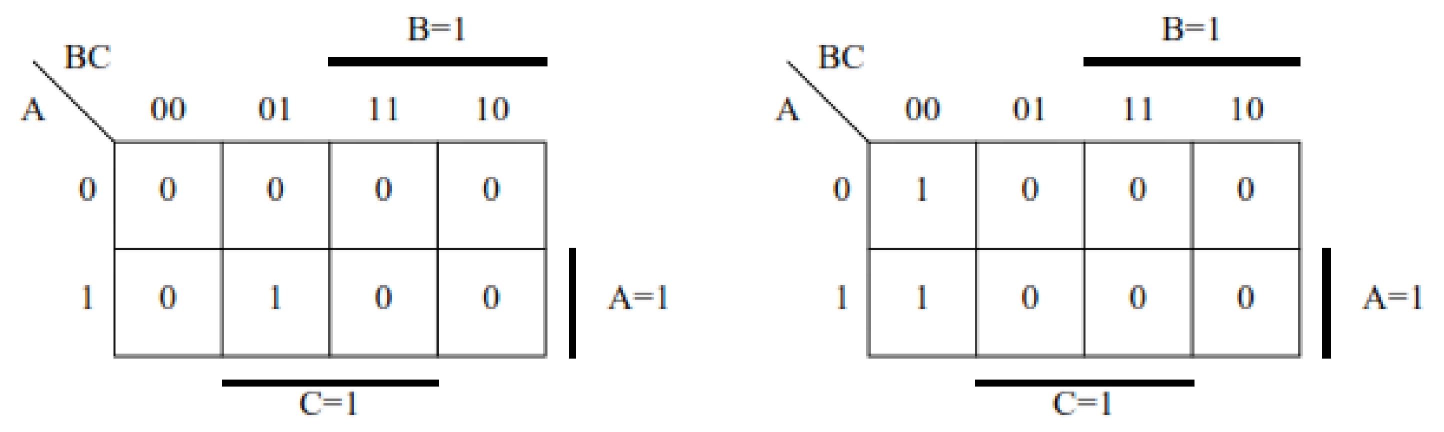
Exemple 4 : trois variables d’entrée A, B et C

1 case = produit de 3 variables

2 cases = produit de 2 variables

4 cases = 1 variable

8 cases = 1 ou 0



A \ BC	00	01	11	10
0	0	0	0	0
1	0	1	0	0

A \ BC	00	01	11	10
0	1	0	0	0
1	1	0	0	0

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

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

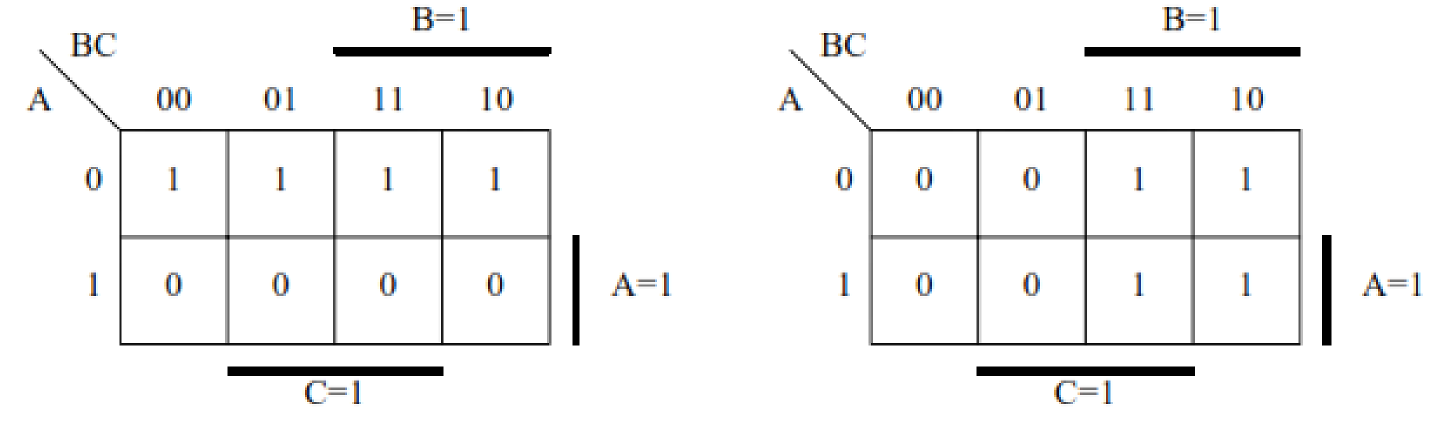
Exemple 5 : trois variables d’entrée A, B et C

1 case = produit de 3 variables

2 cases = produit de 2 variables

4 cases = 1 variable

8 cases = 1 ou 0



A \ BC	00	01	11	10
0	1	1	1	1
1	0	0	0	0

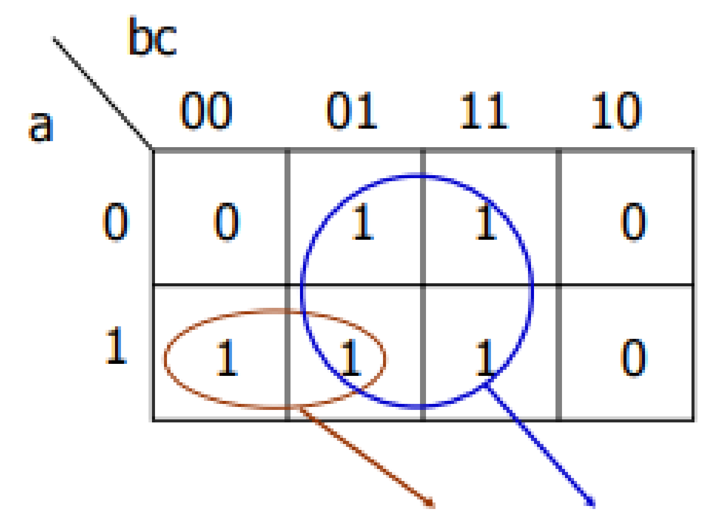
A \ BC	00	01	11	10
0	0	0	1	1
1	0	0	1	1

$S = \overline{A}$

$S = B$

Exemple 6

Tous les 1 sont groupés !



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

Equation :

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

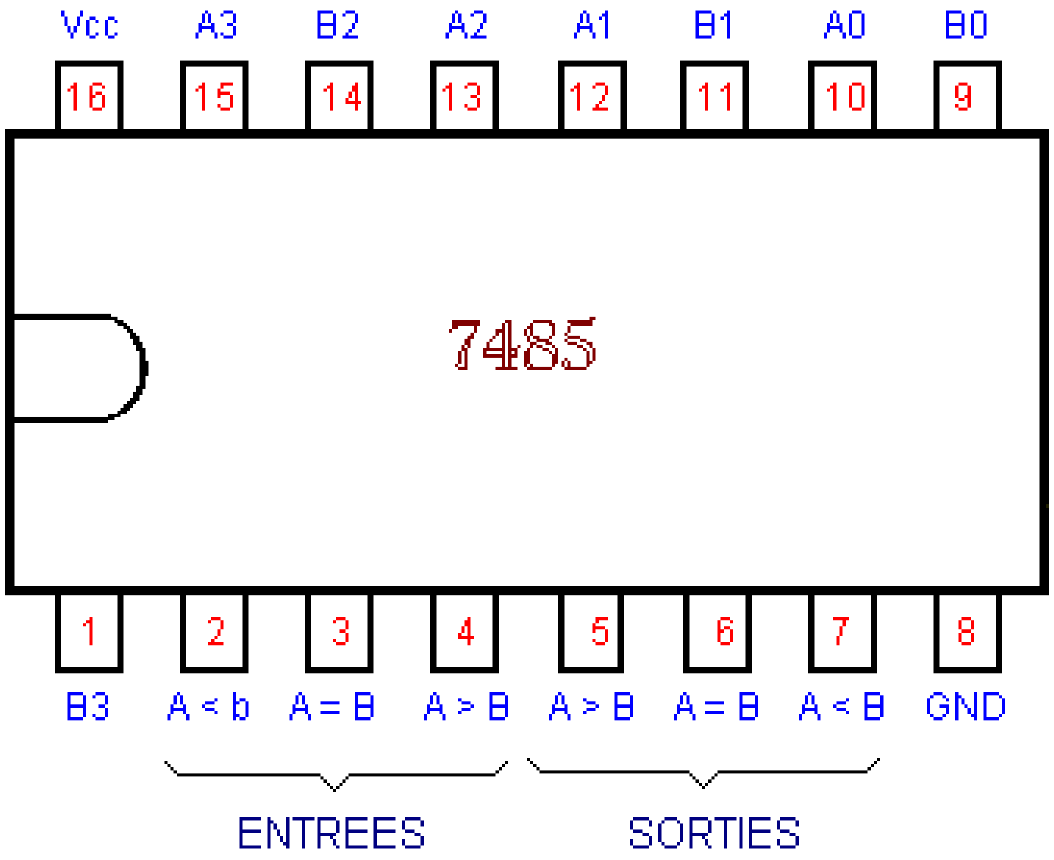


Fig. 21. - Brochage du circuit intégré 7485.

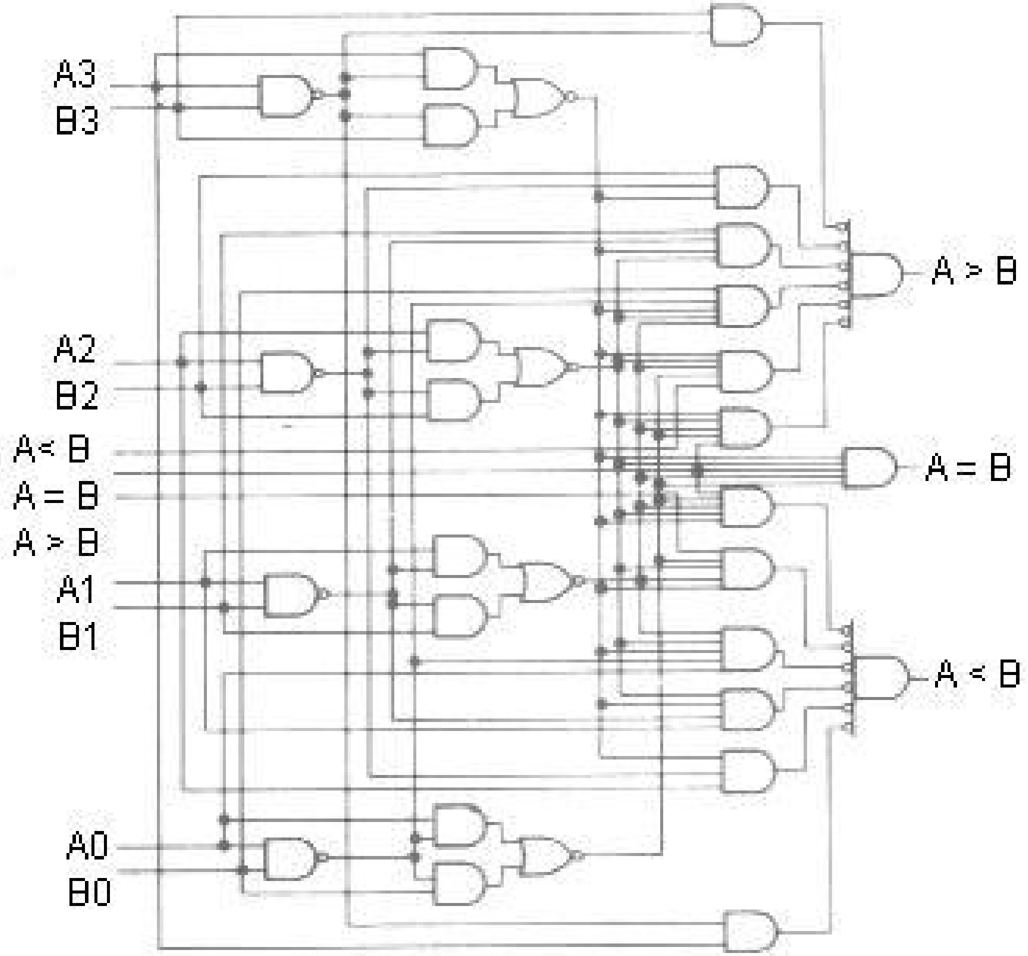


Fig. 22. - Schéma logique du circuit intégré 7485.

Conception et réalisation des réseaux combinatoires

Unité 1: Informatique Industrielle

Circuits logiques combinatoires

Conception et réalisation des réseaux combinatoires :

Le tableau suivant résume ce que nous avons déjà vu. Ces circuits logiques existent avec 2, 3 voir 4 entrées.

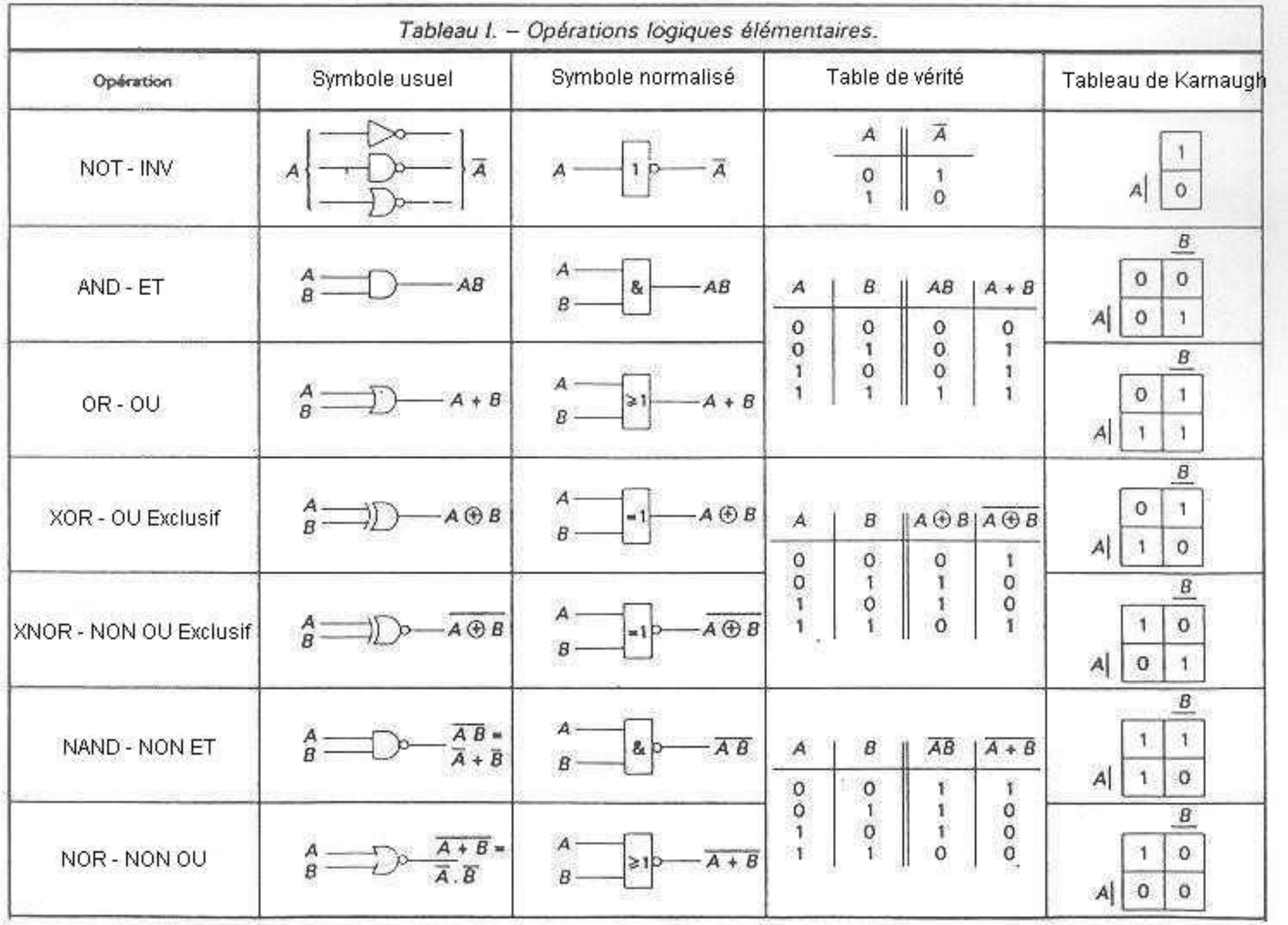


Tableau I. – Opérations logiques élémentaires.

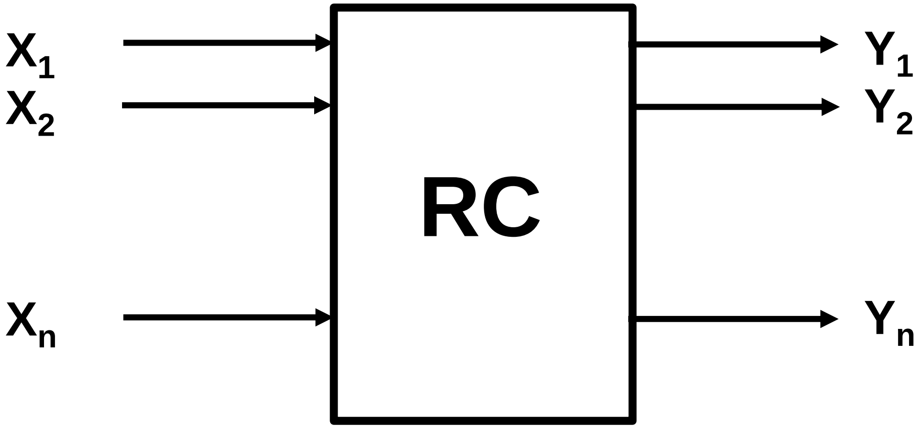
Opération	Symbole usuel	Symbole normalisé	Table de vérité	Tableau de Karnaugh
NOT - INV	A → $\overline{A}$	A – 1 – $\overline{A}$	A || $\overline{A}$ 0 || 1 1 || 0	1 A 0
AND - ET	A, B → AB	A, B – & – AB	A | B || AB | $A+B$ 0 | 0 || 0 | 0 0 | 1 || 0 | 1 1 | 0 || 0 | 1 1 | 1 || 1 | 1	B 0 0 A 0 1
OR - OU	A, B → $A+B$	A, B – ≥1 – $A+B$		B 0 1 A 1 1
XOR - OU Exclusif	A, B → $A \oplus B$	A, B – =1 – $A \oplus B$	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	B 0 1 A 1 0
XNOR - NON OU Exclusif	A, B → $\overline{A \oplus B}$	A, B – =1 – $\overline{A \oplus B}$		B 1 0 A 0 1
NAND - NON ET	A, B → $\overline{A\,B} = \overline{A} + \overline{B}$	A, B – & – $\overline{A\,B}$	A | B || $\overline{AB}$ | $\overline{A+B}$ 0 | 0 || 1 | 1 0 | 1 || 1 | 0 1 | 0 || 1 | 0 1 | 1 || 0 | 0	B 1 1 A 1 0
NOR - NON OU	A, B → $\overline{A+B} = \overline{A}.\overline{B}$	A, B – ≥1 – $\overline{A+B}$		B 1 0 A 0 0

Conception et réalisation des réseaux combinatoires :

Les réseaux (circuits ou systèmes) combinatoires (RC) sont des circuits réalisant des fonctions binaires.

Caractéristiques des RC:

- ✓ La sortie ne dépend que des valeurs instantanées des entrées.
- ✓ Les signaux vont tous dans le même sens (de l’entrée vers la sortie)



Sens des signaux

Conception et réalisation des réseaux combinatoires :

- ✓ Les circuits logiques combinatoires sont des circuits constitués des portes ci-dessus fonctionnant simultanément et réalisant une ou plusieurs fonctions logiques.

- ✓ A une combinaison d’entrées (l’entrée) ne correspond qu’une seule combinaison de sorties (la sortie). La « sortie » apparaît après application de l’ « entrée » avec un certain retard qui est le temps de propagation dans la logique interne. Ce temps est déterminé par la technologie utilisée, le nombre de portes traversées et la longueur des interconnections métalliques.

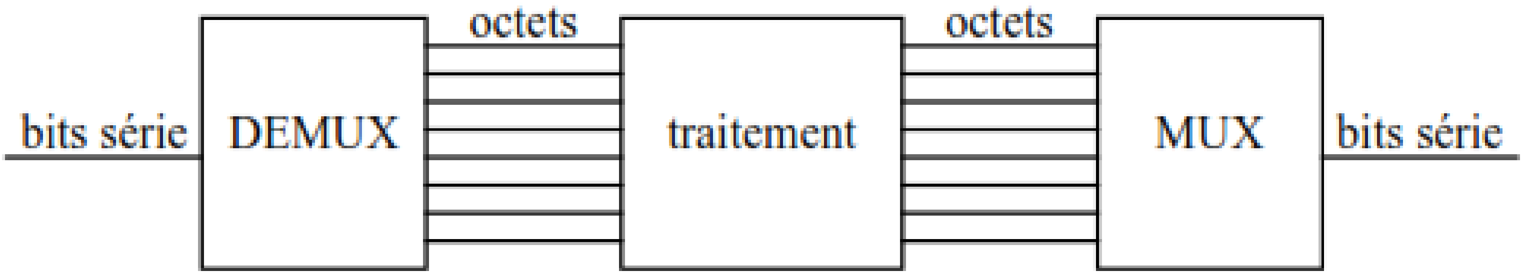
Circuits logiques combinatoires

Conception et réalisation des réseaux combinatoires :

Les circuits combinatoires peuvent servir par exemple

- ✓ à traduire des bits en chiffres représentant un nombre (ou des lettres ou un code particulier). On appelle ces circuits des codeurs (ou bien des décodeurs pour l'opération inverse). Par exemple, un codeur Gray.
- ✓ à effectuer des opérations arithmétiques sur des nombres. Par exemple, un additionneur ou un multiplieur.
- ✓ à transmettre ou recevoir des informations sur une ligne unique de transmission (une ligne série), ce qui nécessite de transformer un nombre écrit sous forme parallèle en une suite de bits mis en série et vice-versa. C'est le rôle des circuits multiplexeur/démultiplexeur.

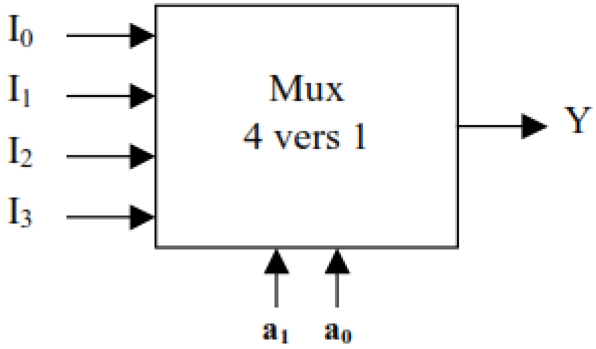
Voici par exemple une transformation série/parallèle suivie d'une transformation parallèle/série :



Conception et réalisation des réseaux combinatoires :

✓ Le multiplexeur

- Le multiplexeur est la fonction inverse du démultiplexeur. C’est un sélecteur de données.
- Il peut transformer une information apparaissant sous forme de n bits en parallèle en une information se présentant sous forme de n bits en série.
- C’est un circuit a 2^n entrées et une sortie choisie parmi les 2n entrées, le choix de l’entrée qui sort est fait par n lignes de sélection.



	a_1	a_0	Y
0	0	0	I_0
1	0	1	I_1
2	1	0	I_2
3	1	1	I_3

Son équation de sortie est égale à :

ave $Y = I_0.\overline{a_1}.\overline{a_0} + I_1.\overline{a_1}.a_0 + I_2.a_1.\overline{a_0} + I_3.a_1.a_0$

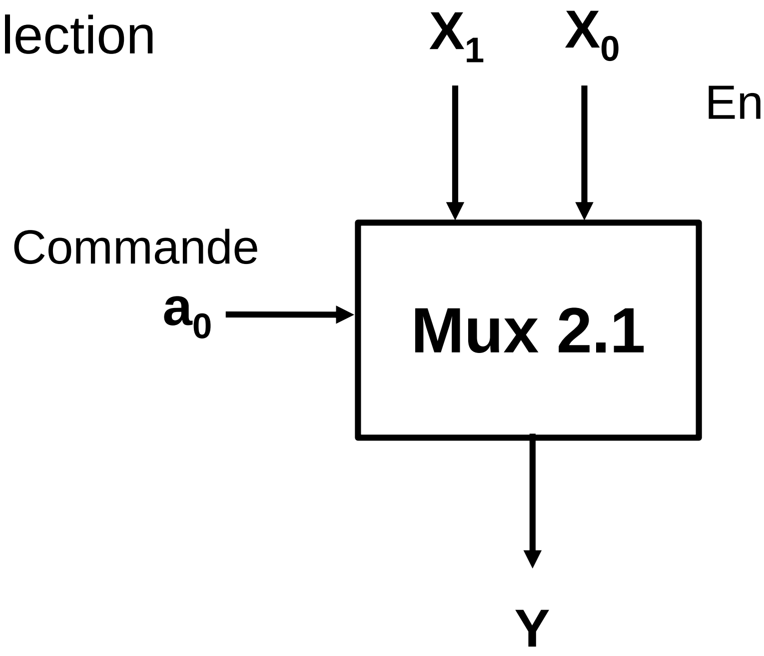
Circuits logiques combinatoires

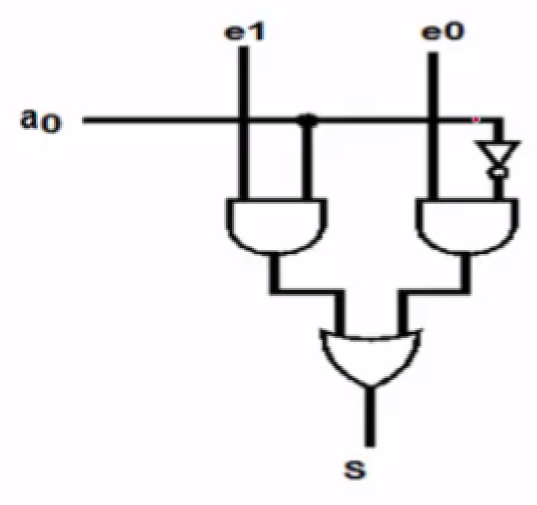
Conception et réalisation des réseaux combinatoires :

✓ Le Multiplexeur (2.1) « 2 vers 1 »

2 entrées = 2^1 donc une ligne de sélection

A_0	S
0	X_0
1	X_1

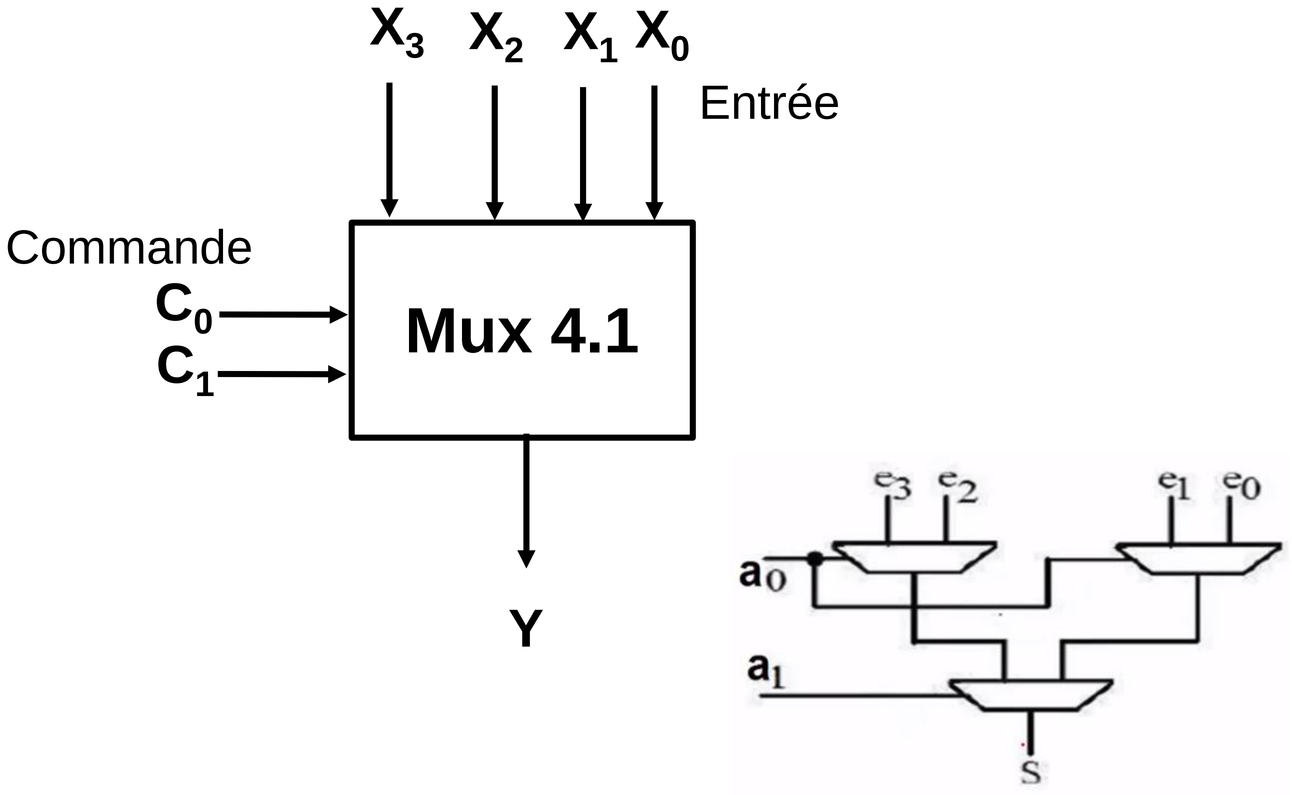




Conception et réalisation des réseaux combinatoires :

✓ Le Multiplexeur (4.1) « 4 vers 1 »

C_0	C_1	Y	
0	0	1	X_0
0	1	0	X_1
1	0	1	X_2
1	1	1	X_3

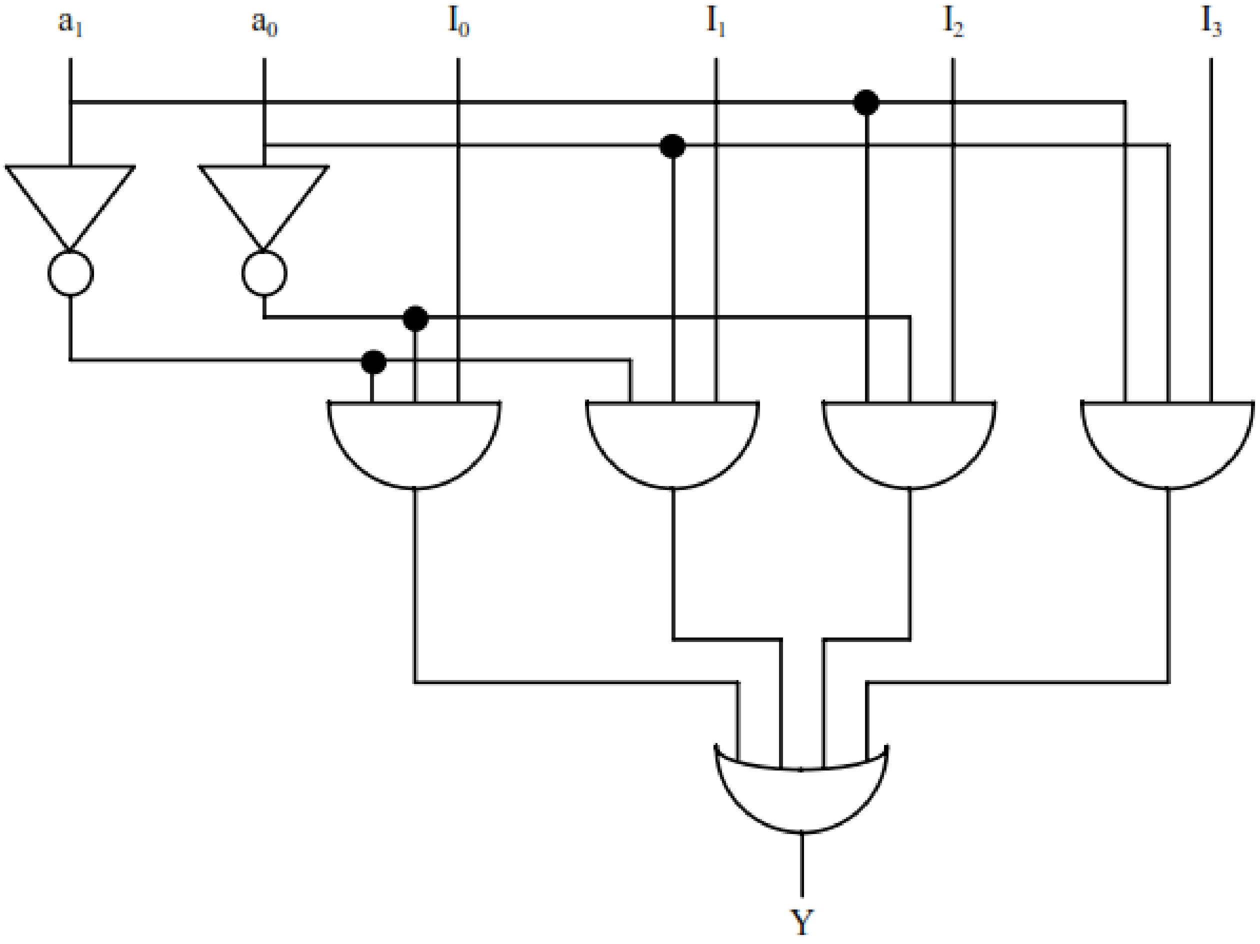


Conception et réalisation des réseaux combinatoires :

✓ **Le multiplexeur**

Le schéma logique est le suivant :

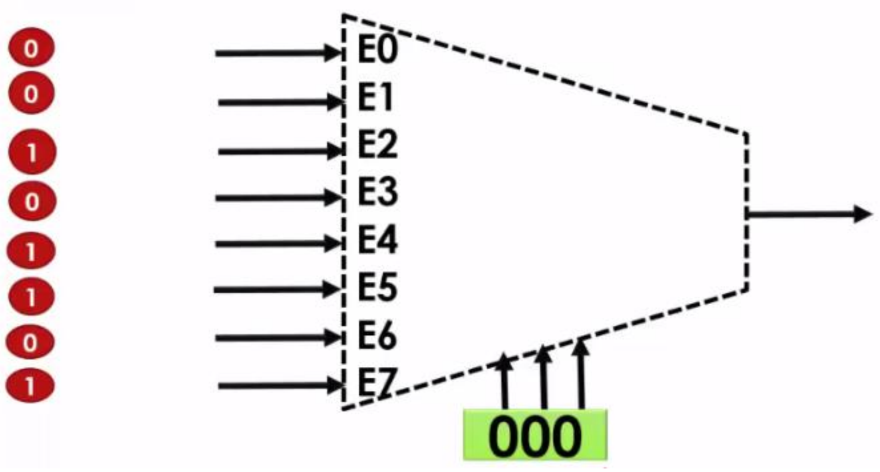
Structure Interne de Mux



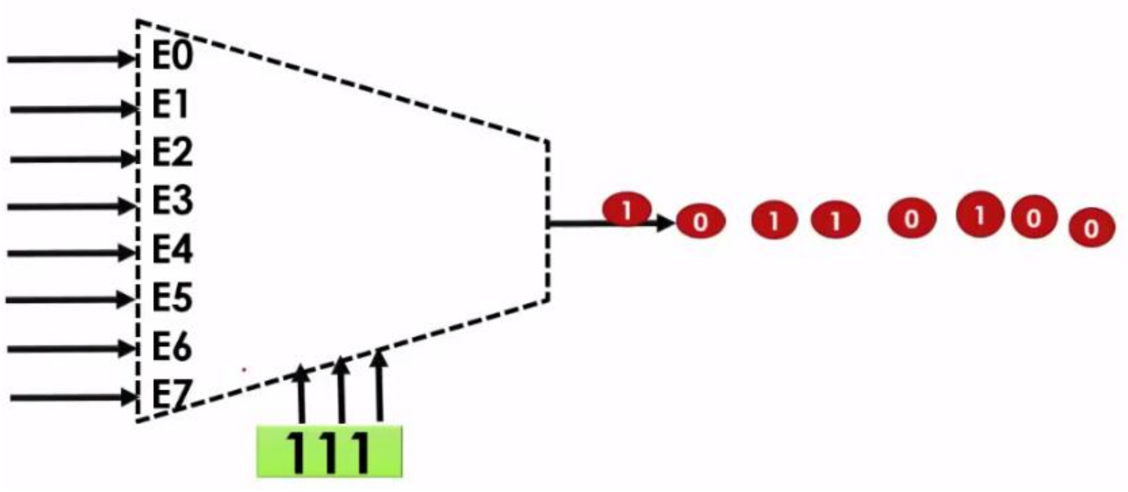
Conception et réalisation des réseaux combinatoires :

✓ Application d’un Multiplexeur

La conversion Parallèle – Série.



Avant Multiplexage

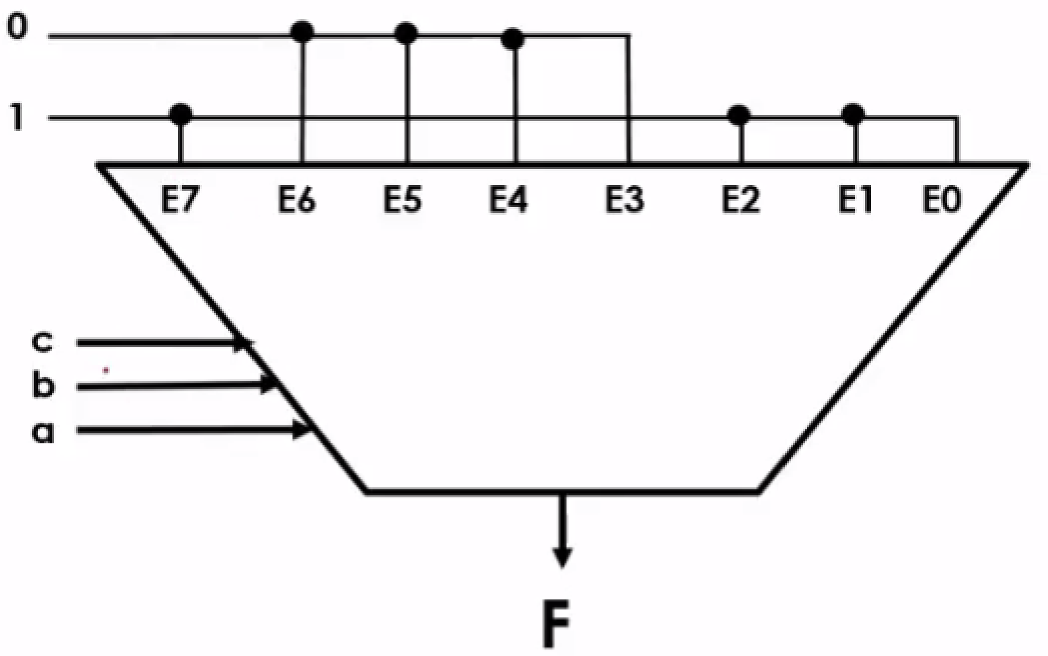


Après Multiplexage

Conception et réalisation des réseaux combinatoires :

✓ Utilisation d'un Multiplexeur 8 vers 1

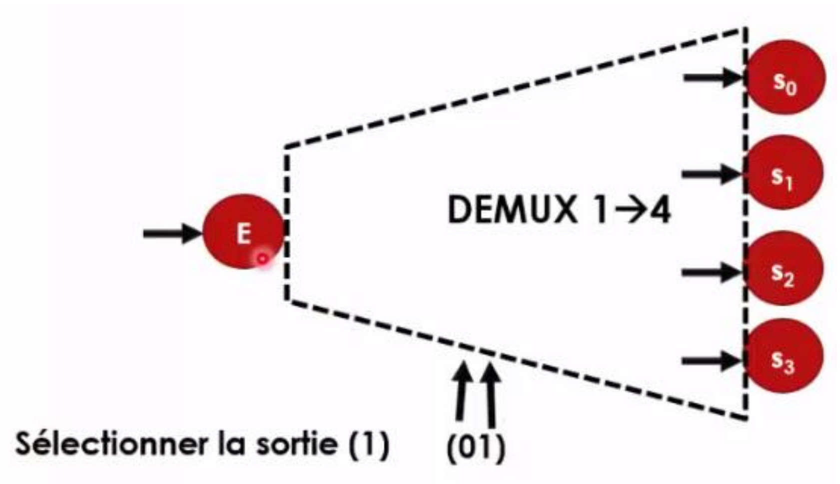
a	b	c	f	Ei
0	0	0	1	E0=1
0	0	1	1	E1=1
0	1	0	1	E2=1
0	1	1	0	E3=0
1	0	0	0	E4=0
1	0	1	0	E5=0
1	1	0	0	E6=0
1	1	1	1	E7=1

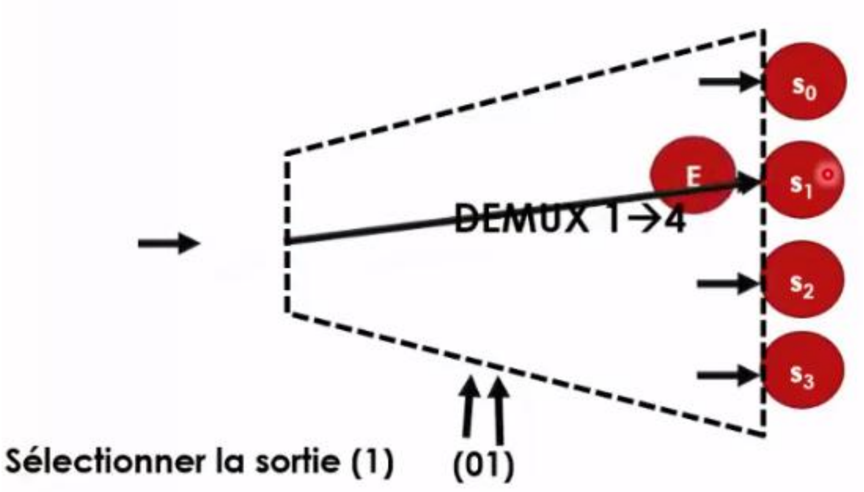


Conception et réalisation des réseaux combinatoires :

✓ Le démultiplexeur

C'est un circuit a une entrée **E** et **2^n** sorties tel que l'entrée **E** est dirigée vers une sortie parmi les **2^n** sorties, le choix de la ligne de sortie s'effectue selon la valeur des **n** lignes de sélection



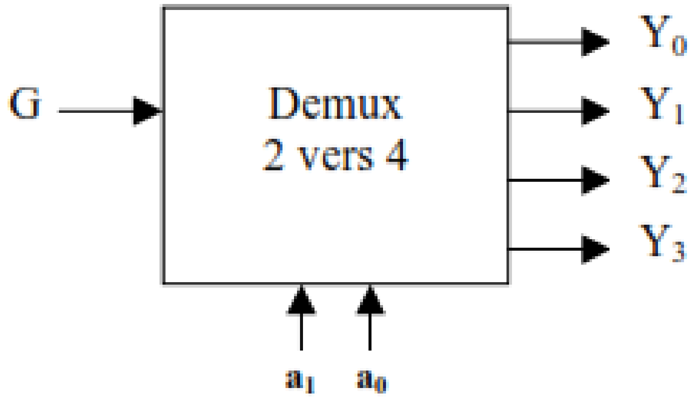


Circuits logiques combinatoires

Conception et réalisation des réseaux combinatoires :

✓ Le démultiplexeur

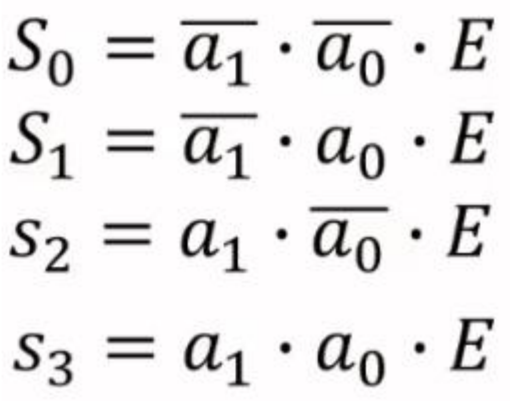
C’est un circuit qui aiguille une entrée vers une sortie dont on donne l’adresse sous forme d’un nombre codé en binaire.



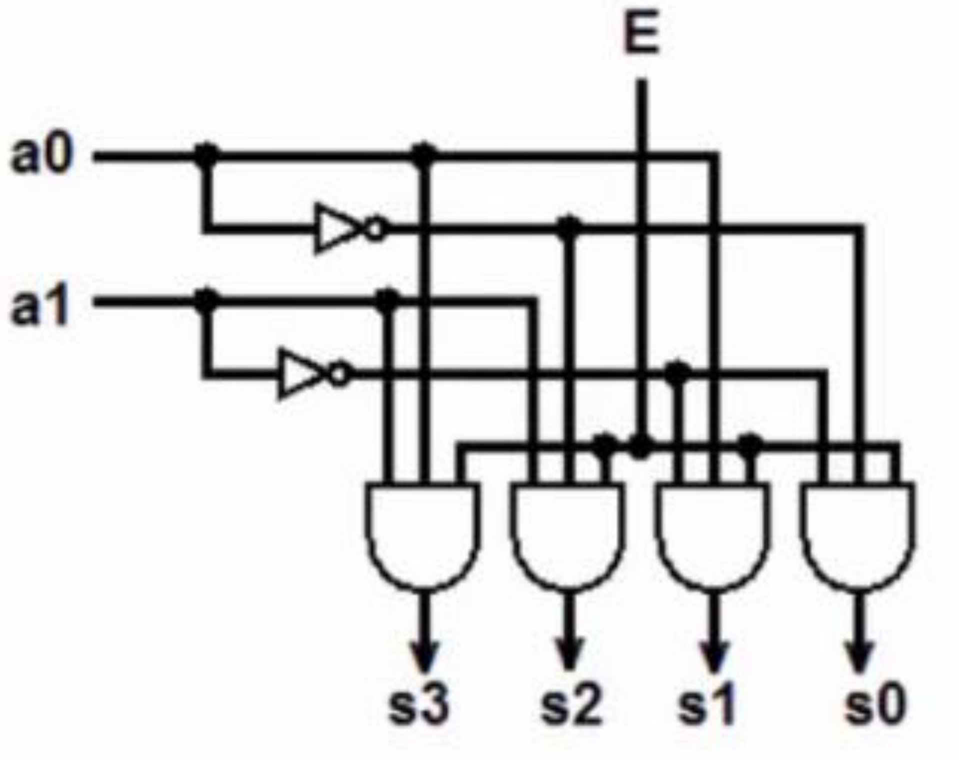
Conception et réalisation des réseaux combinatoires :

- ✓ Le démultiplexeur 1 vers 4

a1	a0	S3	S2	S1	S0
0	0	/	/	/	E
0	1	/	/	E	/
1	0	/	E	/	/
1	1	E	/	/	/



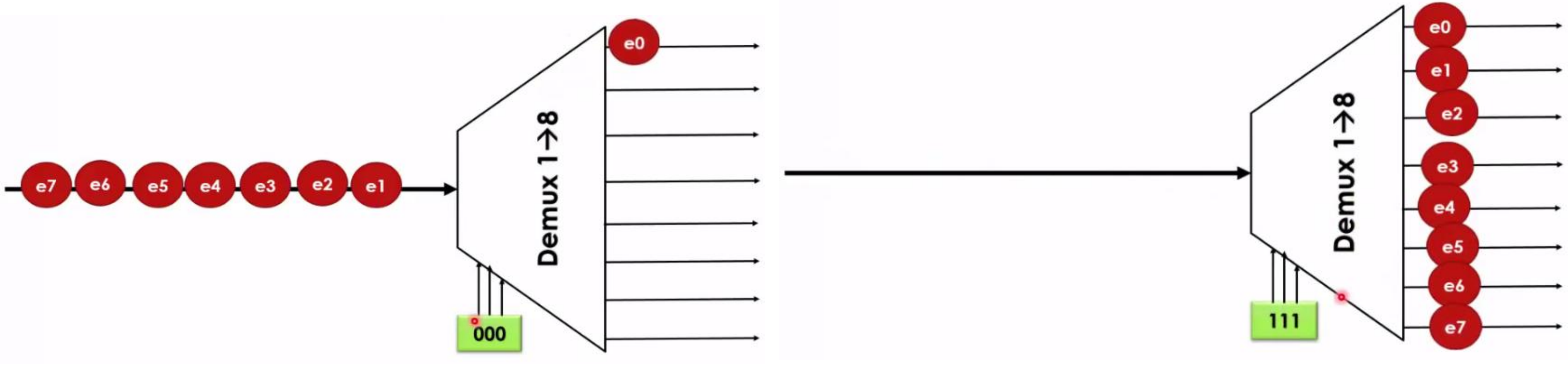
$$S_0 = \overline{a_1} \cdot \overline{a_0} \cdot E$$
$$S_1 = \overline{a_1} \cdot a_0 \cdot E$$
$$s_2 = a_1 \cdot \overline{a_0} \cdot E$$
$$s_3 = a_1 \cdot a_0 \cdot E$$



Conception et réalisation des réseaux combinatoires :

✓ Application d'un démultiplexeur

Conversion série - parallèle : télephone --→ pc



Avant démultiplexage

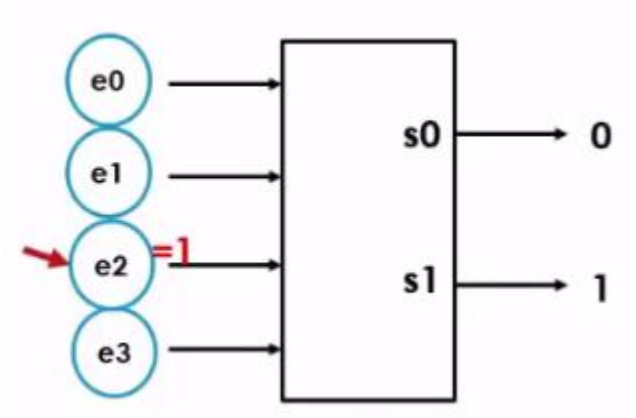
Après démultiplexage

Circuits logiques combinatoires

Conception et réalisation des réseaux combinatoires :

- ✓ Codeur (ENCODEUR)

C’est un circuit à 2^n entrées et ^{n} sorties, il génère un code binaire équivalent au numéro d’une entée activée,



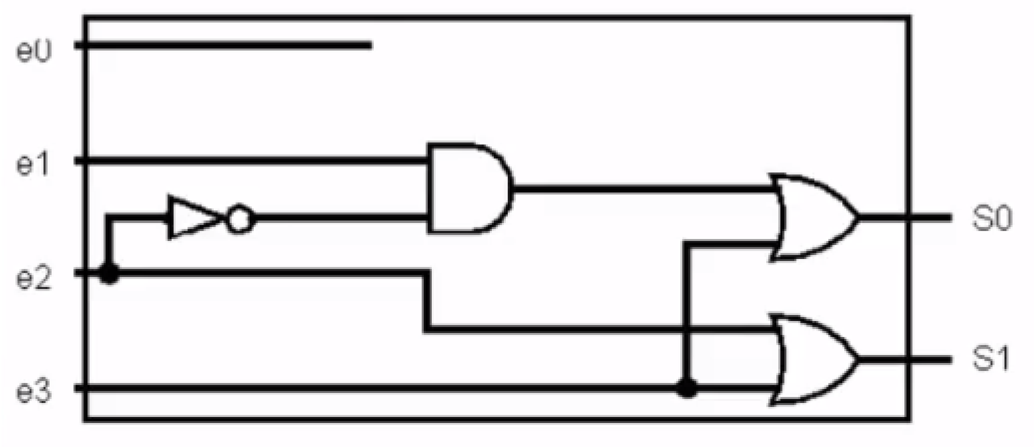
e_3	e_2	e_1	e_0	S1	S0
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	1	0
1	0	0	0	1	1
1	0	0	1	1	1
1	0	1	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1

Conception et réalisation des réseaux combinatoires :

Conception d’un codeur prioritaire 4 ---- > 2

Après simplification on obtient:

$$S_1 = e_2 + e_3 \quad et \quad S_0 = e_3 + \bar{e}_2 e_1$$



Conception et réalisation des réseaux combinatoires :

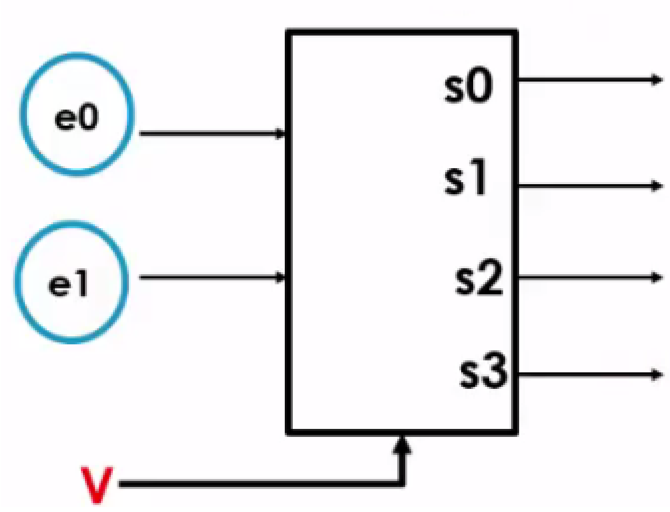
✓ Application d’un Codeur (ENCODEUR)

- Claviers des calculatrice; télécommande; ordinateurs:
- Où il transforme une touche appuyée du clavier en code binaire équivalent (code ASCII en cas de clavier d’ordinateur).

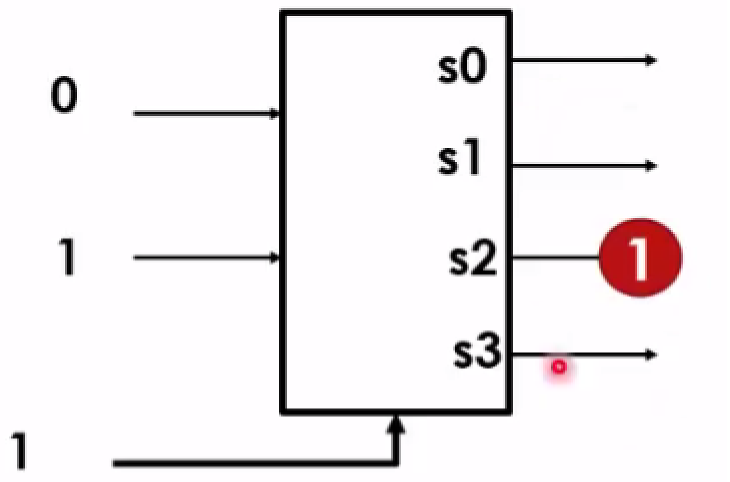
Conception et réalisation des réseaux combinatoires :

✓ Le décodeur

- C'est un circuit a **n** entrées et **2^n** sorties, il active la sortie qui porte le numéro équivalent au nombre binaire entré et tandis que les autres sorties sont inactives (restent a zéro)
- L'entrée **V** active ou désactive le décodeur.

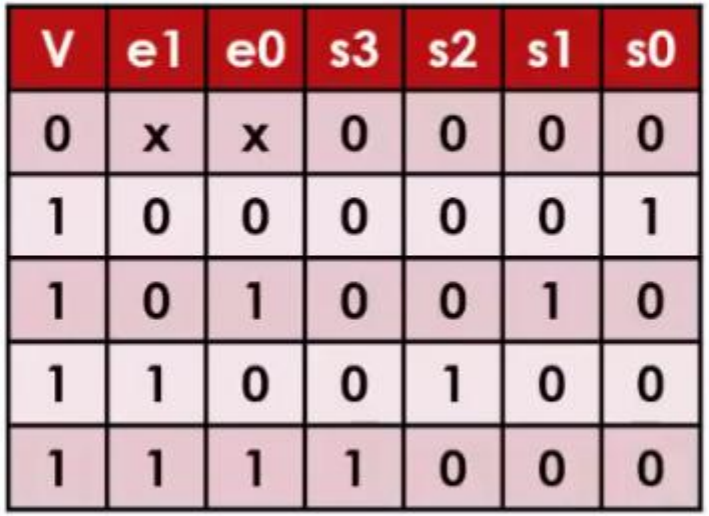


Exemple:

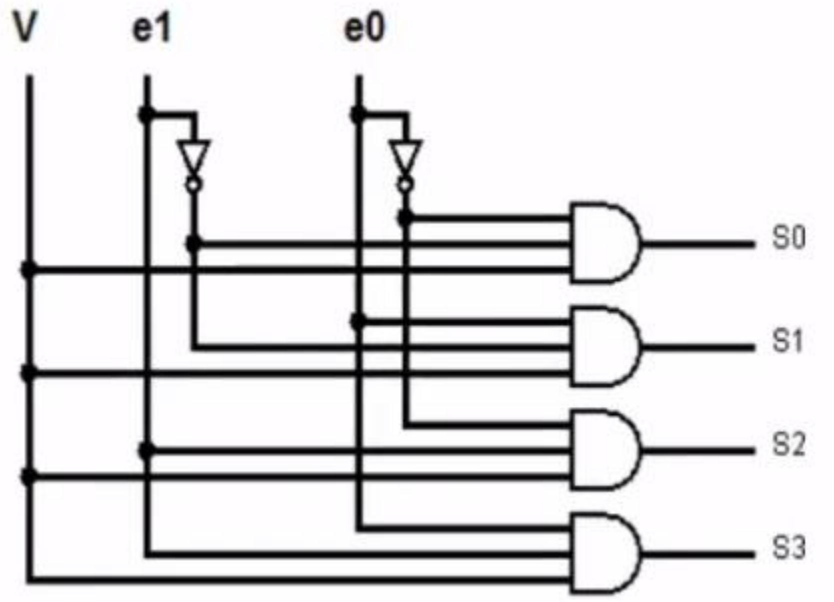


Conception et réalisation des réseaux combinatoires :

Conception d’un décodeur 2 vers 4



V	e1	e0	s3	s2	s1	s0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0



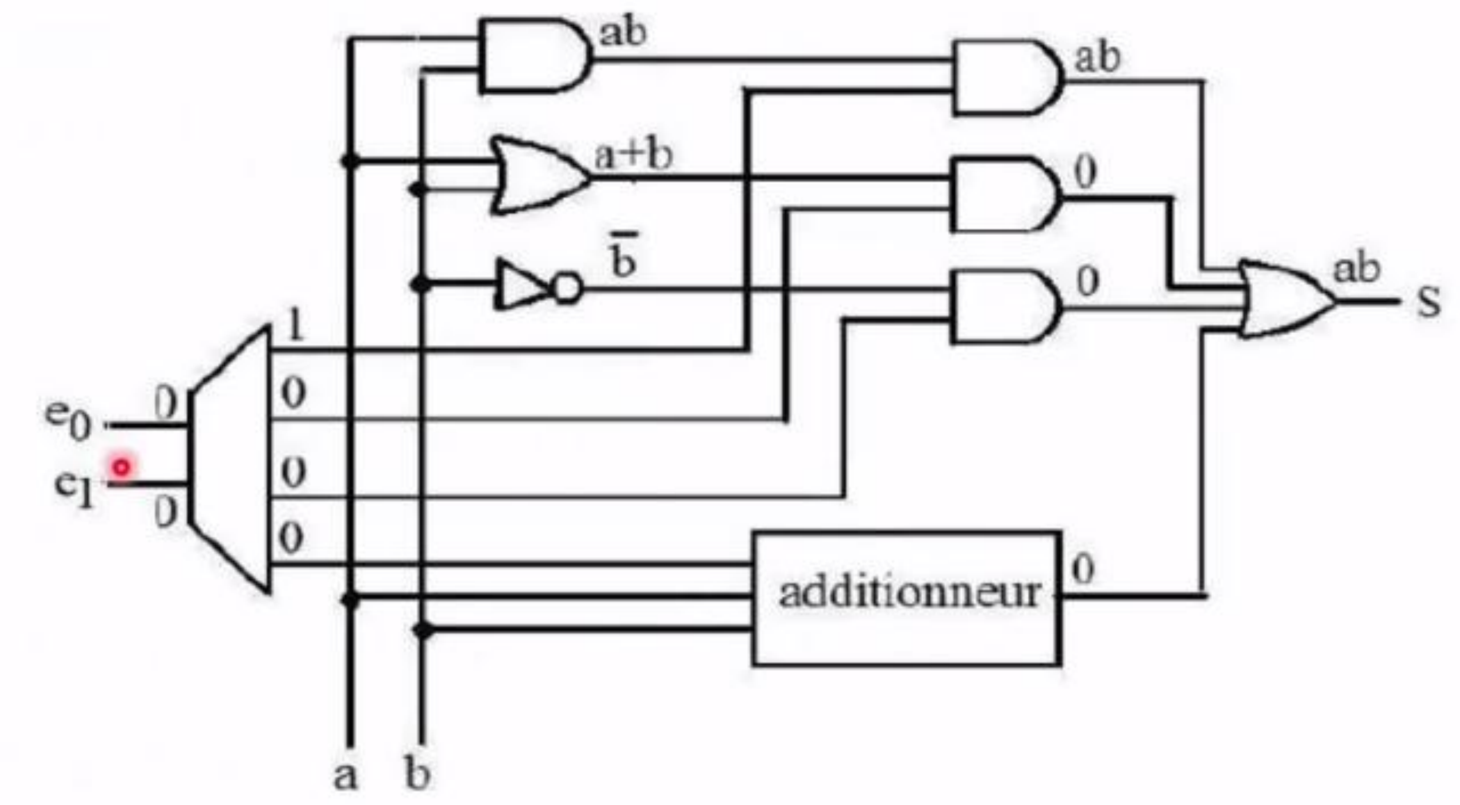
$S_0 = V\bar{e}_1\bar{e}_0$ et $S_1 = V\bar{e}_1 e_0$ et $S_2 = V e_1\bar{e}_0$ et $S_3 = V e_1 e_0$

Circuits logiques combinatoires

Conception et réalisation des réseaux combinatoires :

- **Application d’un décodeur**

Permet de sélectionner l’opération à exécuter dans l’UAL

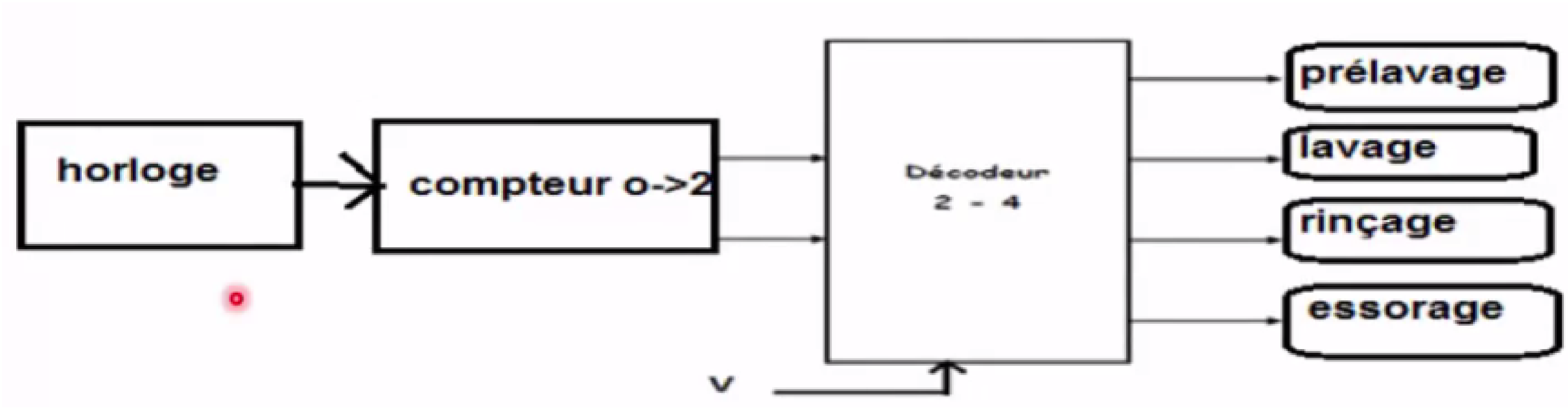


Conception et réalisation des réseaux combinatoires :

Application d'un décodeur

Ou dans n'importe quelle machine

Exemple: machine à laver automatique



Conception et réalisation des réseaux combinatoires :

- **Comparateur**

C’est un circuit qui permet de faire la comparaison entre deux nombres binaires, il a comme sorties:

Fe (A = B) **Fi (A < B)** **Fs (A > B)**

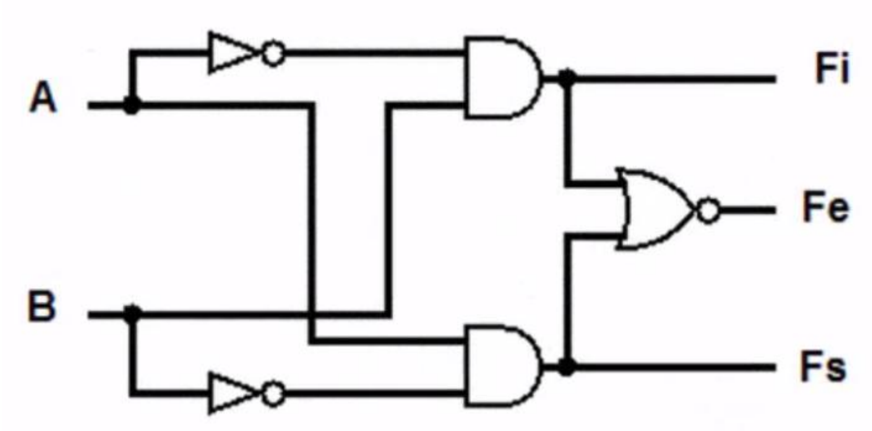
Conception et réalisation des réseaux combinatoires :

- **Comparateur**

Concevoir un comparateur de deux nombres sur un bit chacun:

Table de vérité:

A	B	Fi	Fe	Fs
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0



$$Fi = \bar{A}B \quad et\ Fs = A\bar{B} \quad et\ Fe = \bar{A}\bar{B} + AB = \overline{A \oplus B} = \overline{Fs + Fi}$$

Conception et réalisation des réseaux combinatoires :

- **Comparateur**

Concevoir un comparateur de deux nombres sur deux bits chacun:

En utilisant des comparateurs 1 bit

Soit A=a_1a_0 et B= b_1b_0

Il faut faire la comparaison entre a1 et b1 d’une part et entre a0 et b0 d’autre part,

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 $(a_0 < b_0)]$

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

Table de vérité:

Conception et réalisation des réseaux combinatoires :

H