Elément de module :

Architecture des ordinateurs

Enseigné par:

- **CHERIF Walid**

Année universitaire 2023/2024

Chapitre 3:

3.2 - Les circuits logiques combinatoires

Définition

Circuits Combinatoire :

Famille de circuits logiques dont les sorties dépendent uniquement des entrées:

$S_i = f(E1, E2, \ldots, En)$

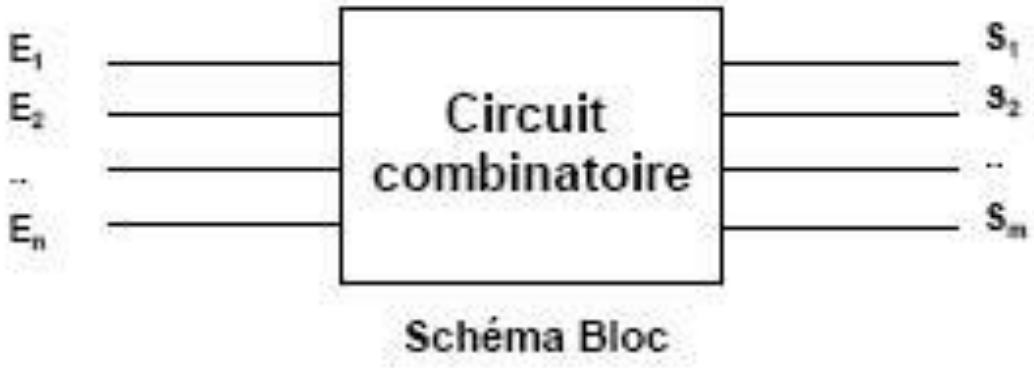


Schéma Bloc

Exemple: Unité arithmétique et logique

Exemples de circuits combinatoires

- Multiplexeur / Démultiplexeur
- Encodeur / Décodeur
- Additionneur
- Soustracteur
- Comparateur
- Circuit de contrôle de parité
- Unité arithmétique et logique

Multiplexage/Démultiplexage

Pour transmettre sur une seule ligne des informations en provenance de plusieurs sources possibles à destination de plusieurs cibles :

Le multiplexeur permet de sélectionner une information (1 bit) parmi 2^n valeurs en entrée.

Il permet de faire passer une information dans l'une des sorties selon les valeurs des entrées de commandes.

Multiplexeur 2^n ×n

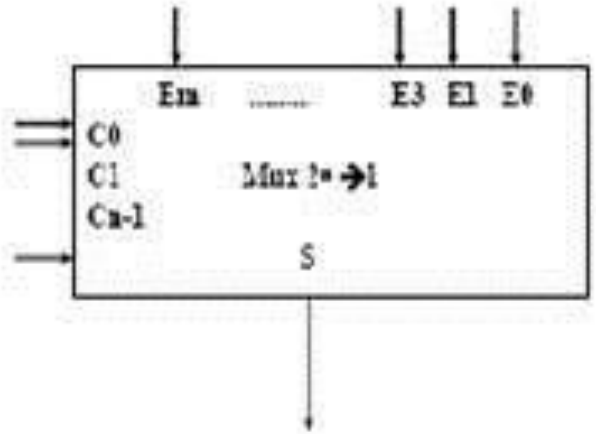
Entrées :

2^n lignes d'entrée (données) : D_0,… , D_{2^n-1}

n lignes de sélection : a, b, c,…

Sortie :

Une seule sortie S



Rôle :

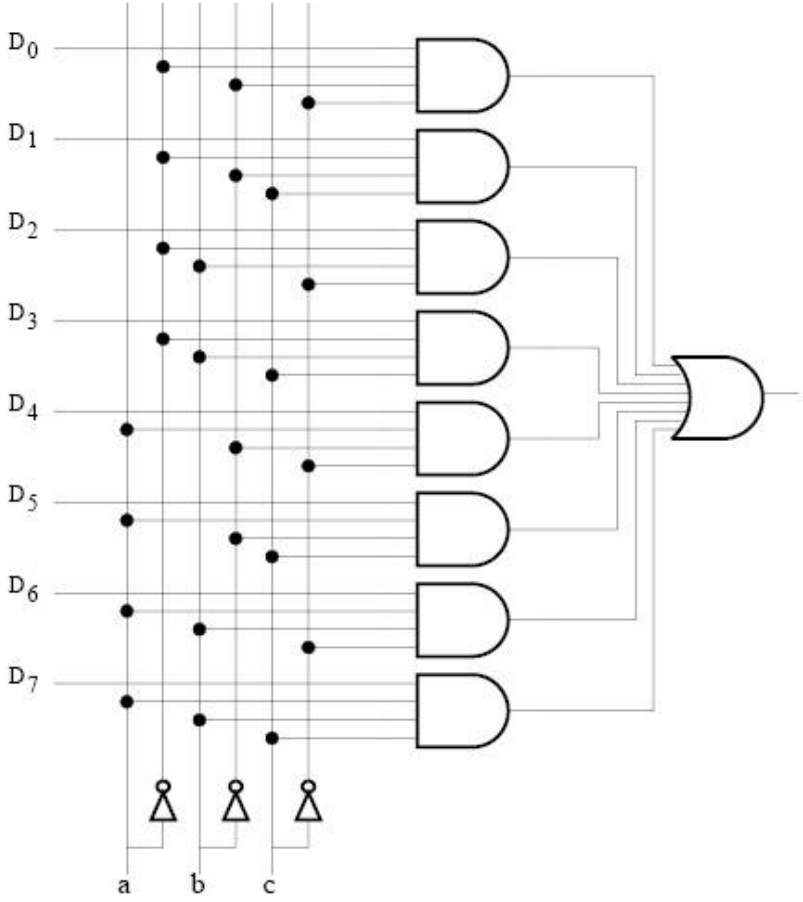
Aiguiller la valeur de l'une des 2^n lignes d'entrée vers la sortie S.
La ligne d'entrée choisie est désignée grâce aux bits de sélection.

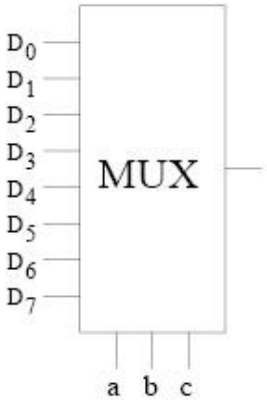
Exemple: Multiplexeur 8 ×3

L'expression logique de la fonction

a	b	c	S
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

Schéma de câblage: Multiplexeur 8 ×3

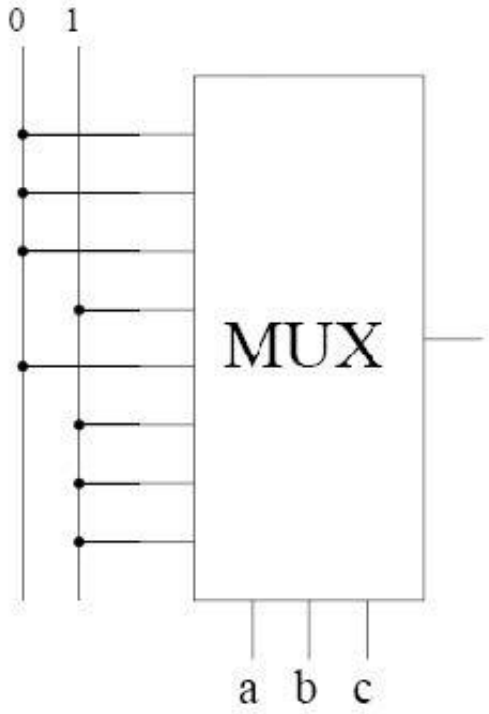




Exemple d'application du Multiplexeur 8 ×3

La fonction majoritaire

a	b	c	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Démultiplexeur 2^n ×n

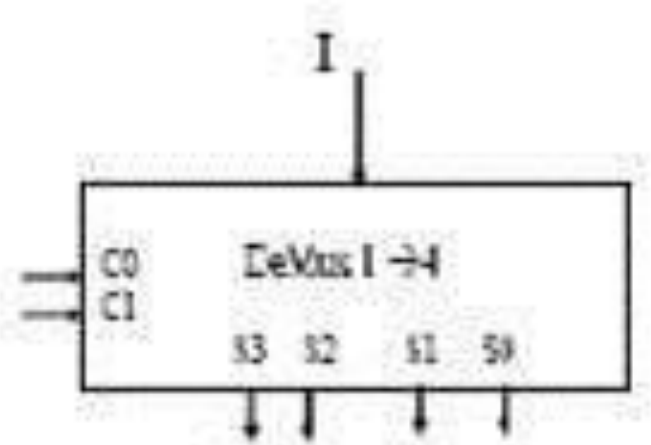
Entrées :

une ligne d'entrée (donnée) : I

n lignes de sélection : C_0,… , C_{2^n-1}

Sortie :

2^n lignes de sortie: S_0,… , S_{2^n-1}



Rôle :

Aiguiller l'entrée E vers l'une des 2^n lignes de sortie.
La ligne de sortie est designée grâce aux bits de sélection..

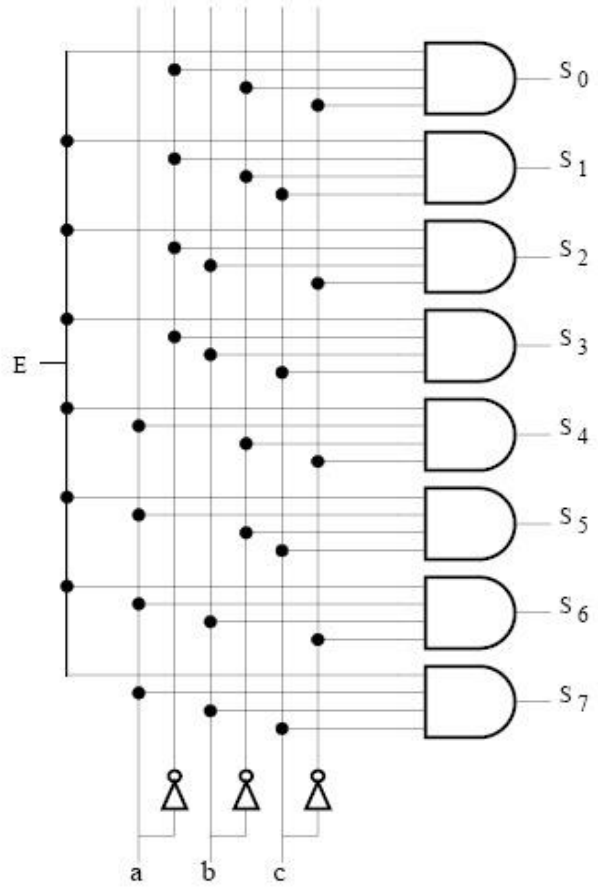
Exemple: Démultiplexeur 8 ×3

Table de vérité

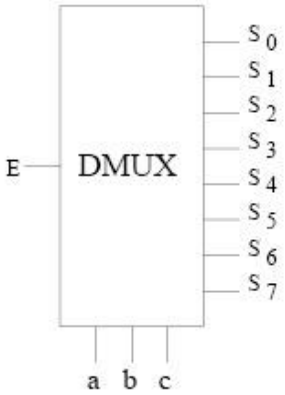
Expressions logiques des S_i

a	b	c	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	E	0	0	0	0	0	0	0
0	0	1	0	E	0	0	0	0	0	0
0	1	0	0	0	E	0	0	0	0	0
0	1	1	0	0	0	E	0	0	0	0
1	0	0	0	0	0	0	E	0	0	0
1	0	1	0	0	0	0	0	E	0	0
1	1	0	0	0	0	0	0	0	E	0
1	1	1	0	0	0	0	0	0	0	E

Schéma de câblage:



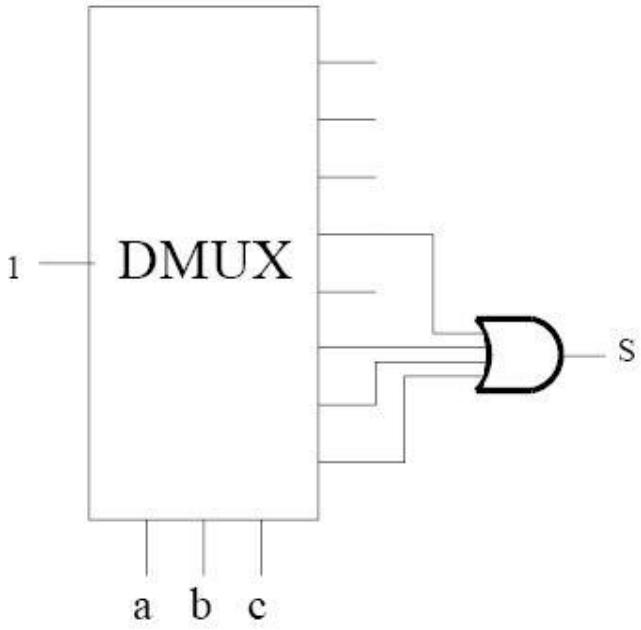
Démultiplexeur 8 ×3



Exemple d'application du Démultiplexeur 8 ×3

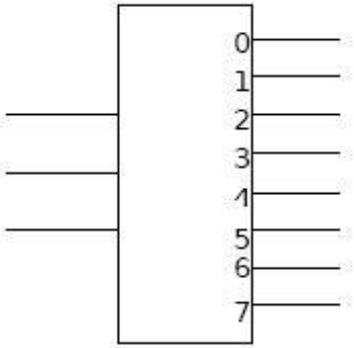
La fonction majoritaire avec un démultiplexeur

a	b	c	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



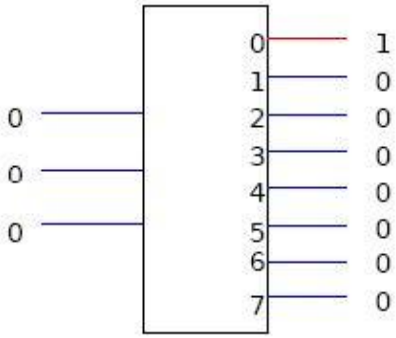
Décodage

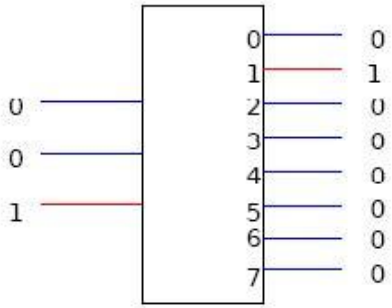
l'opération de décodage permet d'identifier un objet parmi N à partir d'un code qui l'identifie de façon unique

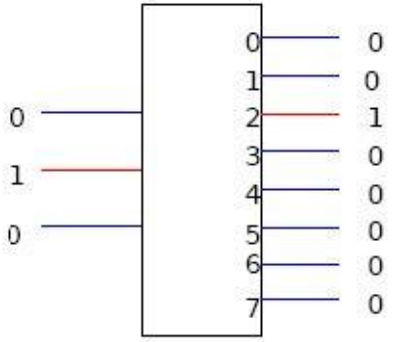


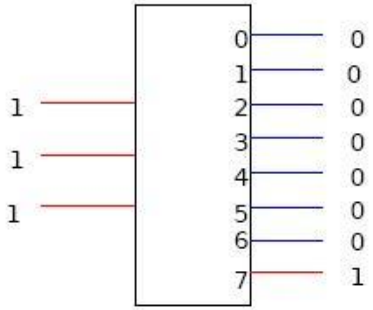
Exemple : 8 instructions codées sur 3 bits

utilisé dans les unités de contrôle des CPU









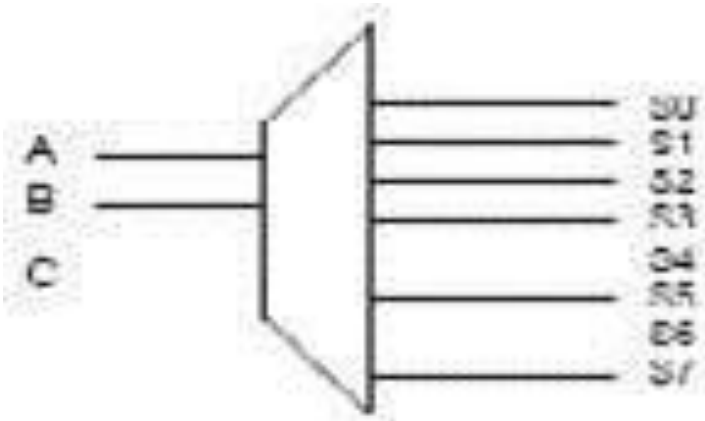
Décodeur 2^n ×n

Entrées :

n lignes de sélection : a, b, c,…

Sortie :

2^n lignes de sortie : S_0,… , S_{2^n-1}



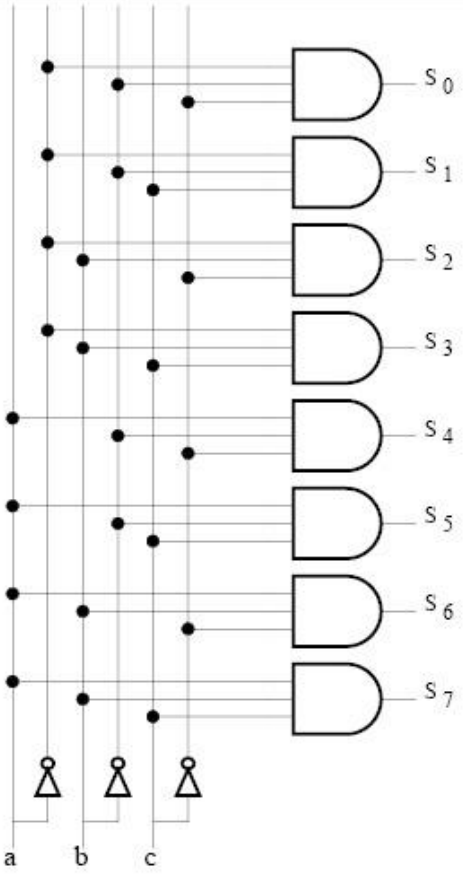
Rôle :

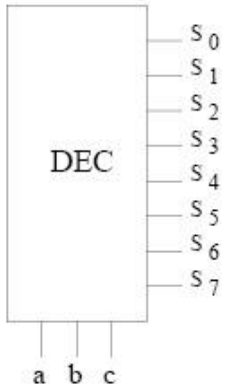
Sélectionner (mettre a 1) l'une des 2^n lignes de sortie.
La ligne de sortie est codée par les bits de sélection.

Exemple: Décodeur 8 ×3

a	b	c	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Schéma de câblage: Décodeur 8 ×3





Démultiplexeur avec E=1.

Encodage

Opération inverse du décodage

Entrées :

2^n lignes d'entrée (données): D_0,… , D_{2^n-1}

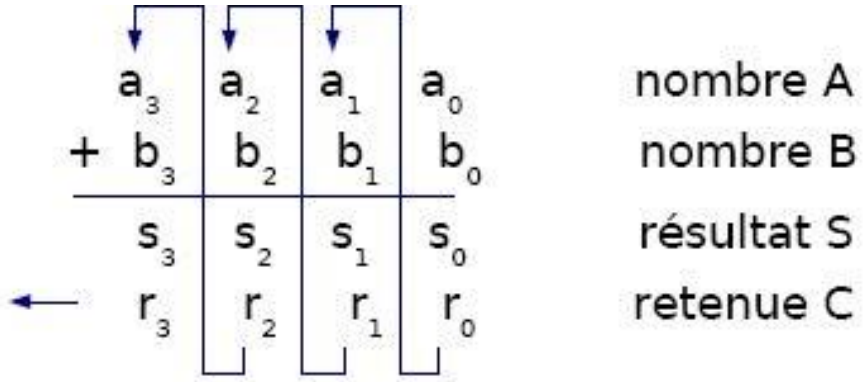
Sortie :

n lignes en sortie S

Rôle :

lorsque une des lignes est activée en entrée, l'encodeur fournit en sortie un mot de n bits correspondant au codage de l'information identifiée par la ligne d'entrée.

Additionneur binaire: principe



Addition de 2 bits en base 2
0 + 0 = 00
0 + 1 = 01
1 + 0 = 01
1 + 1 = 10
Deux bits pour coder l'addition
Avec prise en compte de la retenue

½ Additionneur (half Adder):

Entrées :

les deux bits à additionner a et b

Sortie :

la somme S = a + b

la retenue C

Rôle : Additionner a et b en conservant la retenue

½ Additionneur (half Adder):

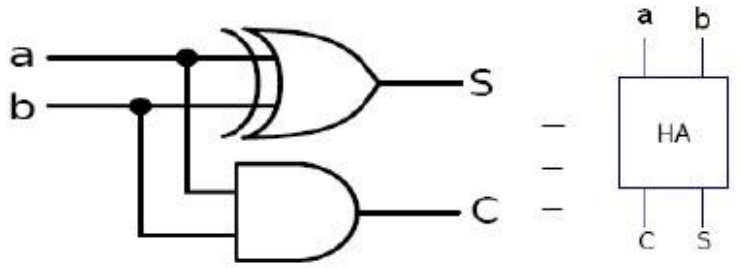
le circuit logique

a	b	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Logigramme :

$$S = a \oplus b$$

$$C = ab$$



Additionneur complet:

Entrées :

les deux bits à additionner a et b

La retenue d'entrée C_{in}

Sortie :

la somme S

la retenue C_{out}

Rôle :

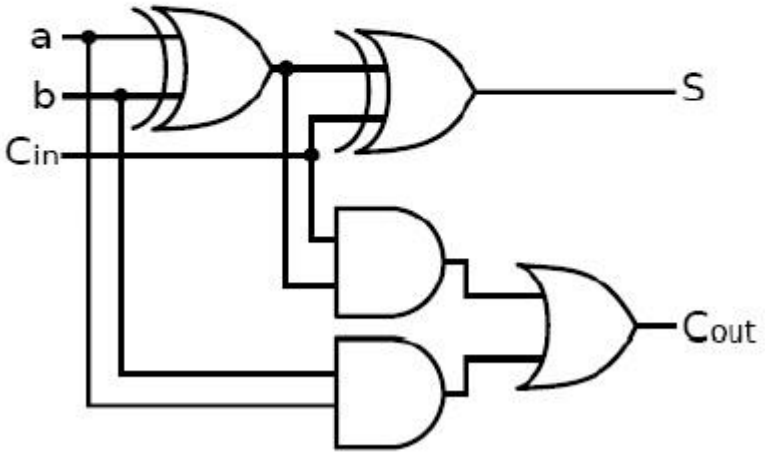
Additionner a et b en prenant en compte la retenue d'entrée C_{in} et en conservant la retenue de sortie C_{out}

Additionneur 1bit : circuit logique

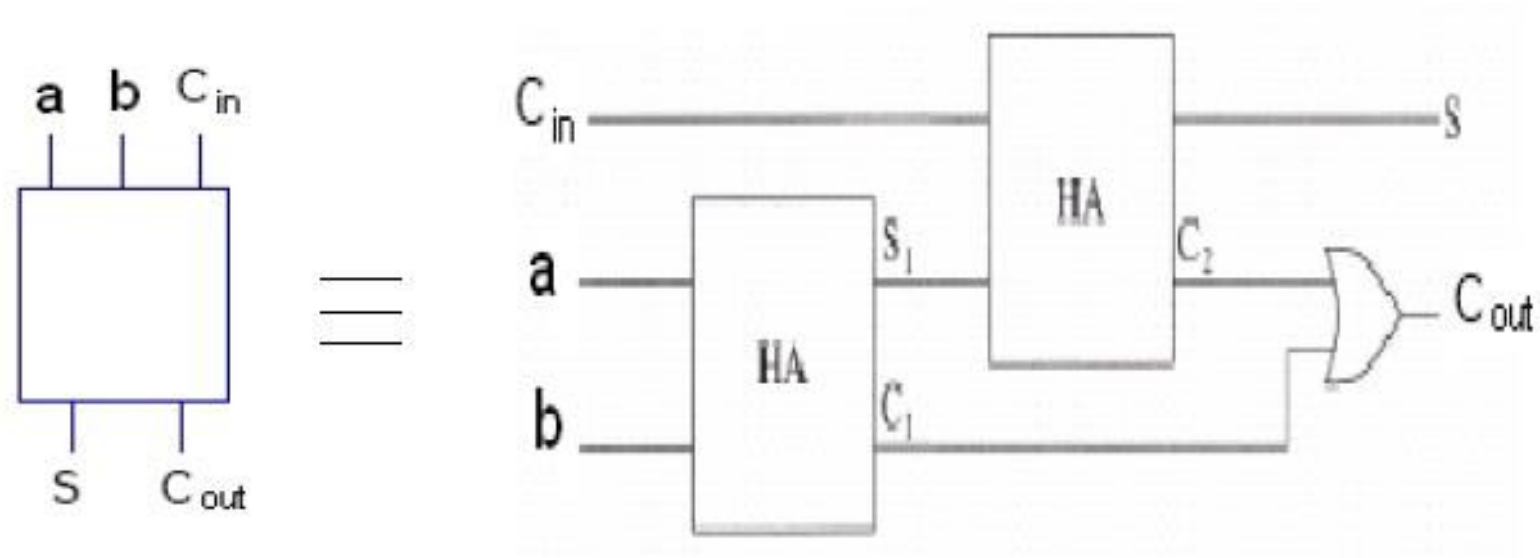
a	b	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S = a \oplus b \oplus C_{in}$$

$$C_{out} = (a \oplus b)C_{in} + ab$$



Additionneur 1bit = 2 demi-additionneurs



Additionneur complet sur 4 bit

Rôle :

Additionner deux nombres binaires A et B de 4 bits chacun en tenant compte de la retenue entrante

Entrées :

les deux nombres a additionner a et b

A = a3a2a1a0

B = b3b2b1b0

Sortie :

la somme S = A+B (sur 4 bit)

la retenue C

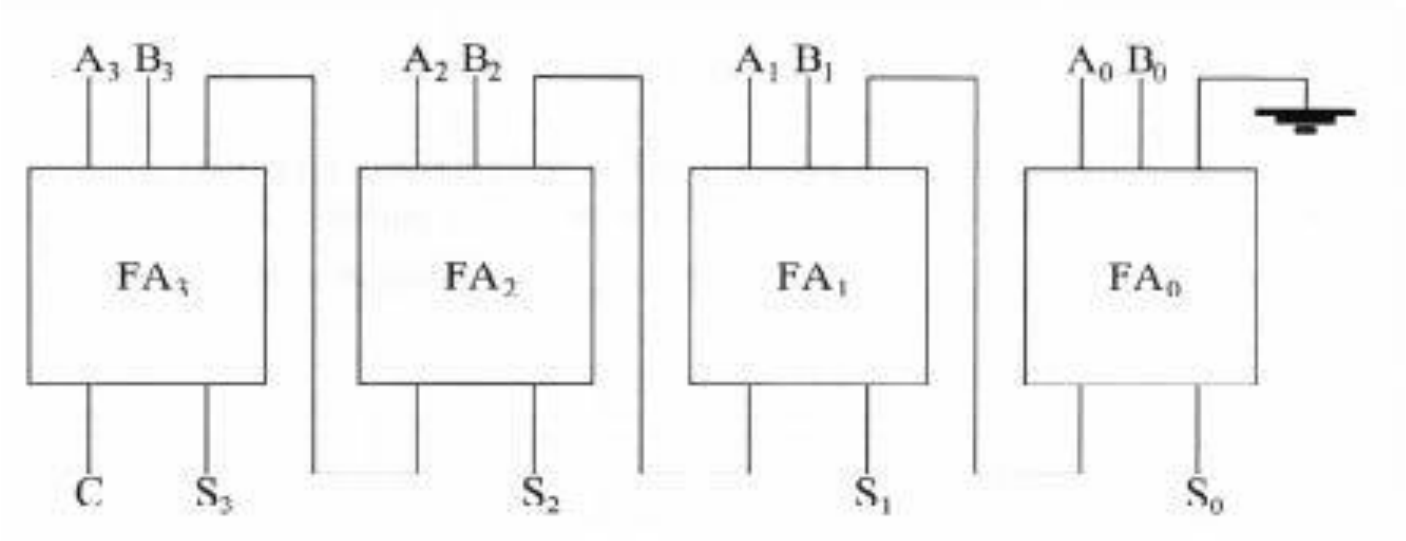
Additionneur complet sur 4 bit

9 entrées et 5 sorties
Table de vérité de taille (2^9 ×5) !

Existe t-il une solution plus efficace et plus simple pour concevoir ce circuit ?

Additionneur parallèle à propagation de retenue

Il est possible de chaîner plusieurs additionneurs un bit pour en fabriquer un capable de traiter des mots de longueurs arbitraires



Quatre additionneurs 1 bit chaînés pour former un additionneur 4 bits

Unité arithmétique et logique

Entrées :

A et B : les variables

F_0et F_1 : bits de choix du signal d'activation

C_{in} : retenue entrante

(optionnel) EN A et EN B : bits inhibiteurs de A et B

(optionnel) INV A : pour obtenir $\bar{A}$

Sortie :

S : résultat de l'opération

C_{out} : retenue de sortie

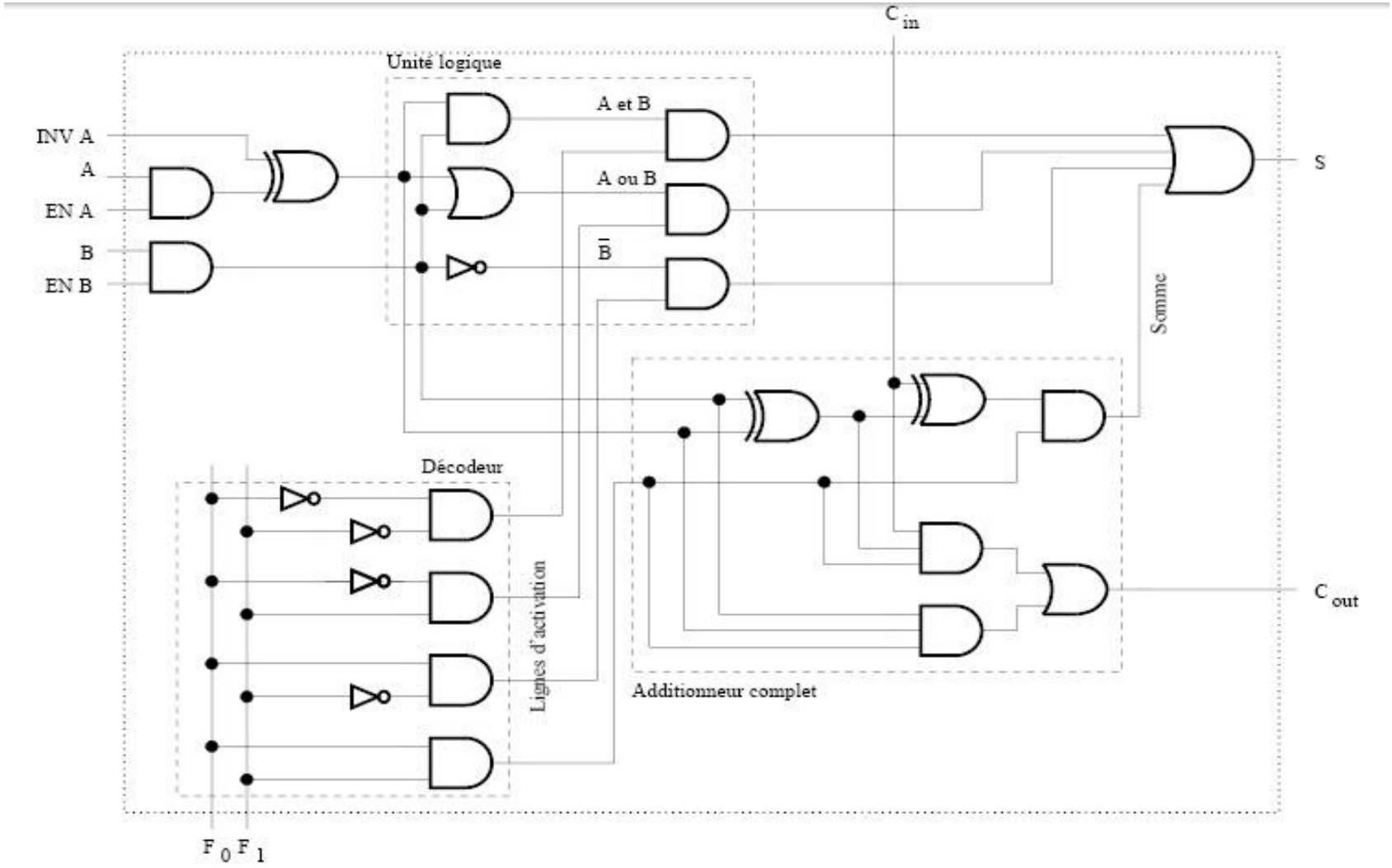
Rôle :

Faire l'une des 4 opérations (ou des variantes) :

A ET B, A OU B, , A + B + C_{in},

en fonction des bits d'activation choisis.

Elément de module : Architecture des ordinateurs



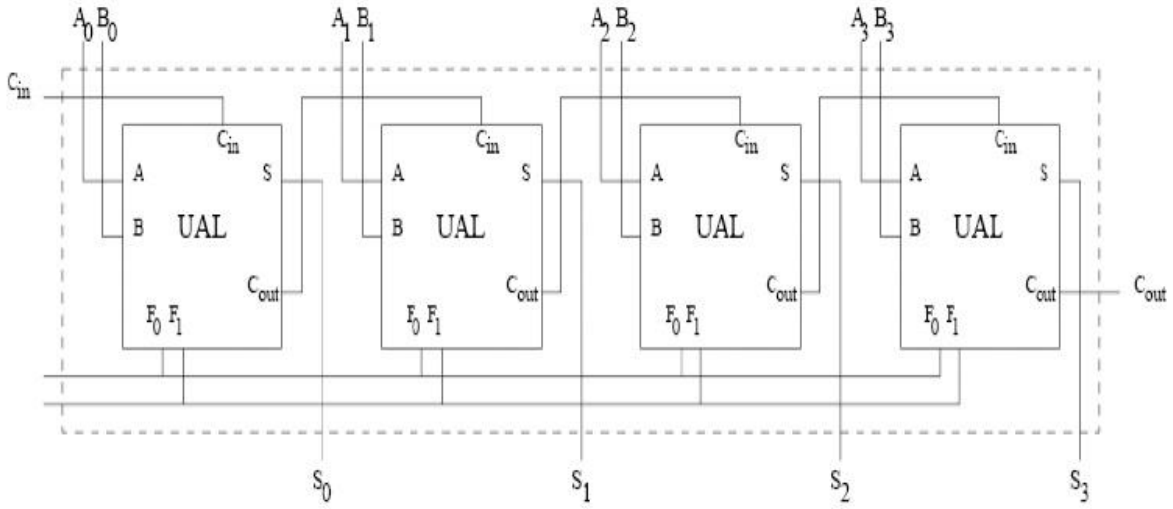
UAL - résumé des fonctions :

F_0	F_1	ENA	ENB	$INVA$	C_{in}	fonction
0	0	1	1	0	0	A ET B
0	1	1	1	0	0	A OU B
0	1	0	0	0	0	0
0	1	0	1	0	0	B
0	1	1	0	0	0	A
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	$A+B$
1	1	0	0	0	1	1
1	1	0	0	1	0	-1
1	1	0	1	0	1	$B+1$
1	1	0	1	1	0	$B-1$
1	1	1	0	0	1	$A+1$
1	1	1	0	1	1	$-A$
1	1	1	1	0	1	$A+B+1$
1	1	1	1	1	1	$B-A$

UAL - 4 bits :

Pour 2 bits d'entrée, l'UAL est un circuit qui a peut d'intérêt...
Mais, en connectant judicieusement les retenues de n UAL, on obtient une UAL n bits, telle que :

- Les opérations logiques sont des opérations bit à bit
- Les opérations arithmétiques sont effectuées sur des entiers en complément à 2 sur n bits.



Exercice 01

En utilisant la méthode des tableaux de Karnaugh, donner l'expression simplifiée de la fonction logique F de 4 variables définie par la table de vérité suivante :

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

Exercice 02

En utilisant les propriétés du codage des entiers relatifs en complément à deux et le mode de calcul du complément à deux,

Proposer un circuit pouvant réaliser la soustraction sur n bits à partir d'un additionneur n bits.

- Proposer un circuit utilisant un unique additionneur n bits qui peut réaliser soit l'addition soit la soustraction sur n bits suivant la valeur d'une nouvelle entrée : OP.

Par exemple : si OP vaut 0 le circuit réalise A + B et si OP vaut 1, c'est la soustraction A − B qui est effectuée.