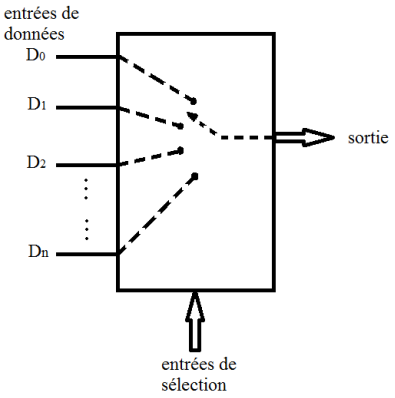
Chapitre III : Circuits Combinatoires complexes.

Les portes simples que nous avons vues jusqu'à maintenant permettent de construire des circuits de plus en plus complexes jusqu'aux microprocesseurs les plus puissants.

Nous allons voir quelques-uns des circuits les plus.

1. Multiplexeur :



Un MUX possède plusieurs entrées et une seule sortie. Il nous permet de choisir entre plusieurs entrées, laquelle nous souhaitons obtenir en sortie ; suivant la valeur d'une autre entrée dite de sélection.

Un MUX comporte donc deux types d'entrées : les entrées de données (ou d'informations), et les entrées de sélection dont les combinaisons servent à numéroter les entrées d'informations.

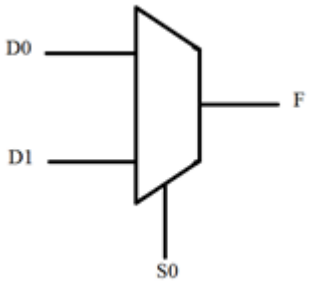
Pour sélectionner une entrée parmi 'n', il faut donner au MUX le numéro de cette entrée, évidemment sous forme binaire : on appellera ce numéro l'adresse de l'entrée correspondante.

Par exemple avec 3 entrées de sélection, on peut compter de 0 $(000)_2$ à 7 $(111)_2$ donc sélectionner une entrée parmi 8. On parlera alors de MUX 8 vers 1. De façon générale, avec 'n' entrées de sélection, on peut sélectionner une entrée de donnée parmi 2^n.

Etudions le cas du MUX le plus simple à deux entrées de données D_0 D_1 et une seule entrée de sélection S_0 :

Si $S_0 = 0 \Rightarrow F = D_0$

Si $S_0 = 1 \Rightarrow F = D_1$

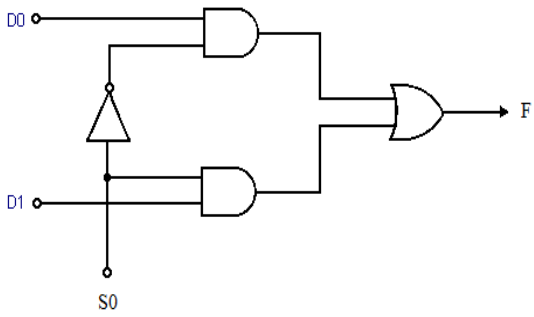


D_0	D_1	S_0	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

En peut réduire la table de vérité de ce MUX est donné comme suit :

S_0	F
0	D_0
1	D_1

Le circuit logique du MUX est donné comme suit :



L'expression logique de la sortie du MUX 2 → 1 peut être facilement déduite de la table de fonctionnement comme suit :

$$F = D_0.\overline{S}_0 + D_1.S_0$$

1.2. Réalisation d'une fonction logique par un MUX:

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

On désire réaliser la fonction logique F définit par la table de vérité ci-contre:

On va utiliser un MUX à trois entrées de sélection ; on place les variables (A, B et C) sur les lignes de sélection du MUX donc pour ce MUX le nombre de lignes d'adresses est égale au nombre de variables de la fonction à réaliser.

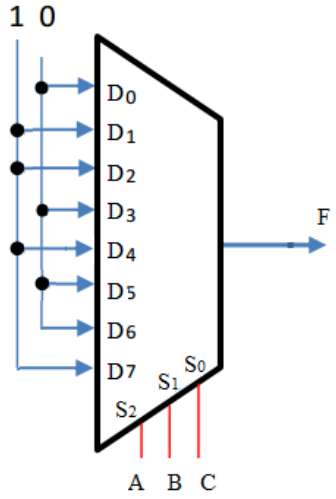
De la table de vérité on a : $F = \overline{A}.\overline{B}.\overline{C} + \overline{A}.B.\overline{C} + \overline{A}.B.C + A.B.\overline{C}$ ... **(1)**

Et d'après le principe du MUX on a :

$F = D_7.A.B.C + D_6.A.B.\overline{C} + D_5.A.\overline{B}.C + D_4.A.\overline{B}.\overline{C} + D_3.\overline{A}.B.C + D_2.\overline{A}.B.\overline{C} + D_1.\overline{A}.\overline{B}.C + D_0.\overline{A}.\overline{B}.\overline{C}$ ……..(2)

En identifiant (1) et (2) on obtient :

$D_0 = D_3 = D_5 = D_6 = 0$; $D_1 = D_2 = D_4 = D_7 = 1$.



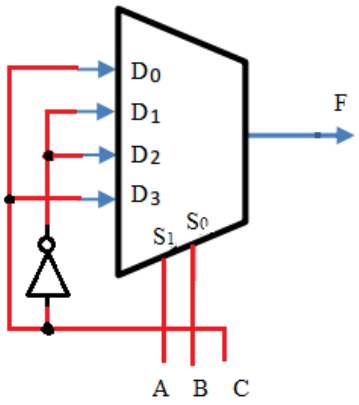
Donc le circuit qui réalise la fonction F à base du MUX à 3 entrées de sélection est :

On peut utiliser un MUX à deux entrées de sélection ; on place les variables sur les lignes de sélection du MUX mais on remarque que le nombre de lignes de sélection est 2 par contre le nombre de variables de la fonction à réaliser est 3. Alors dans ce cas on place juste deux variables (A et B) sur les lignes de sélection.

On établit directement les entées du MUX dans la table de vérité selon les variables A et B.

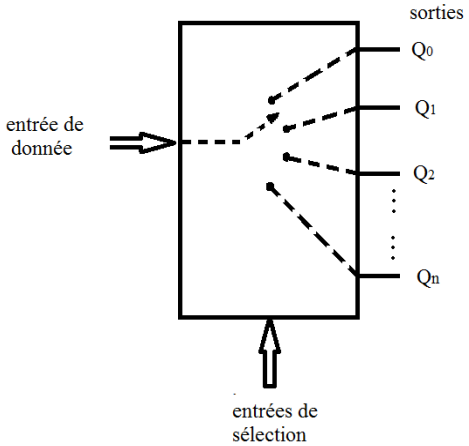
A	B	C	F		
0	0	0	0	D_0	C
0	0	1	1		
0	1	0	1	D_1	$\overline{C}$
0	1	1	0		
1	0	0	1	D_2	$\overline{C}$
1	0	1	0		
1	1	0	0	D_3	C
1	1	1	1		

Donc le circuit qui réalise la fonction F à base du MUX à 2 entrées de sélection est :



2. Démultiplexeur :

Le DEM effectue l'opération inverse de celle du MUX il n'a qu'une entrée de données qui est diriger vers une parmi plusieurs sorties suivant la valeur d'une autre entrée dite de sélection.



Un DEM comporte donc deux types d'entrées : une entrée de données (ou d'informations), et les entrées de sélection dont les combinaisons servent à numéroter les sorties.

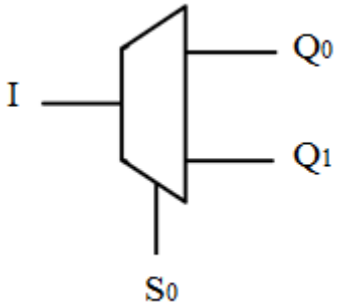
Pour sélectionner une sortie parmi 'n', il faut donner au DEM le numéro de cette sortie, évidemment sous forme binaire : on appellera ce numéro l'adresse de la sortie correspondante.

Par exemple avec 3 entrées de sélection, on peut compter de 0 $(000)_2$ à 7 $(111)_2$ donc sélectionner une sortie parmi 8. On parlera alors de DEM 1 vers 8. De façon générale, avec 'n' entrées de sélection, on peut sélectionner une sortie de donnée parmi 2^n.

Etudions le cas du DEM le plus simple à deux sorties Q_0 Q_1 et une seule entrée de sélection S_0 :

Si $S_0 = 0 \Rightarrow Q_0 = I$

Si $S_0 = 1 \Rightarrow Q_1 = I$

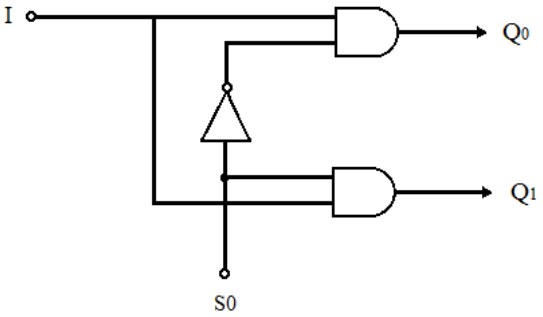


I	S_0	D_1	D_0
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

S_0	D_1	D_0
0	I	0
1	0	I

En peut réduire la table de vérité de ce DEM est donné comme suit :

Le circuit logique du DEM est donné comme suit :



L'expression logique de la sortie du DEM 1 → 2 peut être facilement déduite de la table de fonctionnement comme suit :

$$F = D_0.\overline{S}_0 + D_1.S_0$$

2.2. Réalisation d'une fonction logique par un DEM:

On désire réaliser la fonction logique F définit par la table de vérité ci-contre:

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

On va utiliser un DEM à trois entrées de sélection ; on place les variables

(A, B et C) sur les lignes de sélection du DEM.

De la table de vérité on a : $F = \overline{A}.\overline{B}.C + \overline{A}.B.C + A.\overline{B}.\overline{C} + A.B.\overline{C}$ ... **(1)**

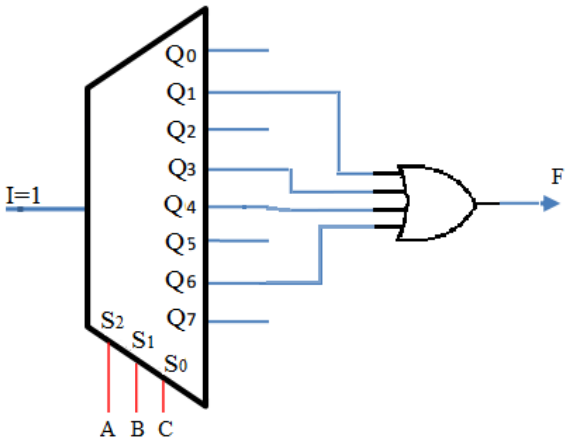
Et d'après le principe du DEM on a :

$Q_0 = \overline{A}.\overline{B}.\overline{C}.I$, $Q_1 = \overline{A}.\overline{B}.C.I$, $Q_2 = \overline{A}.B.\overline{C}.I$,

$Q_3 = \overline{A}.B.C.I$, $Q_4 = A.\overline{B}.\overline{C}.I$, $Q_5 = A.\overline{B}.C.I$,

$Q_6 = A.B.\overline{C}.I$, $Q_7 = A.B.C.I$

Alors $F = Q_1 + Q_3 + Q_4 + Q_6$ en posant I = 1.



3. Décodeur :

Un DEC est un circuit logique qui établit la correspondance entre un code d'entrée binaire de n bits et m lignes de sortie ($m \leq 2^n$). Pour chacune des combinaisons possibles des entrées une seule ligne de sortie est validée. Pour une combinaison binaire de n entrées une seule ligne de sortie sera mise à 1.

Certains DEC n'utilisent pas toute la gamme de 2^n codes d'entrée possible mais seulement un sous-ensemble de ceux-ci. Ils sont alors souvent conçus de façon à ce que les codes inutilisés n'activent aucune sortie lorsqu'ils se présentent à l'entrée du DEC (décodeur BCD-décimal possède 4 bits d'entrées et 10 sorties).

Le fonctionnement du DEC est similaire au fonctionnement d'un DEM avec une entrée de donnée fixé à 1.

On peut appeler un décodeur à n entrées et m sorties un décodeur 1 parmi m (par exemple un décodeur 1 parmi 8).

Etudions le cas du DEC 1 parmi 8 ($1 \rightarrow 8$) le plus simple à huit sorties Q_0 Q_1 Q_2 Q_3 Q_4 Q_5 Q_6 Q_7 et trois entrées de sélection S_0 S_1 S_2 La sortie activée est celle qui a son numéro décimal expriment en binaire en entrée.

La table de vérité du décodeur est donnée par :

S_2	S_1	S_0	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_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

L'équation simplifiée de chaque sortie est :

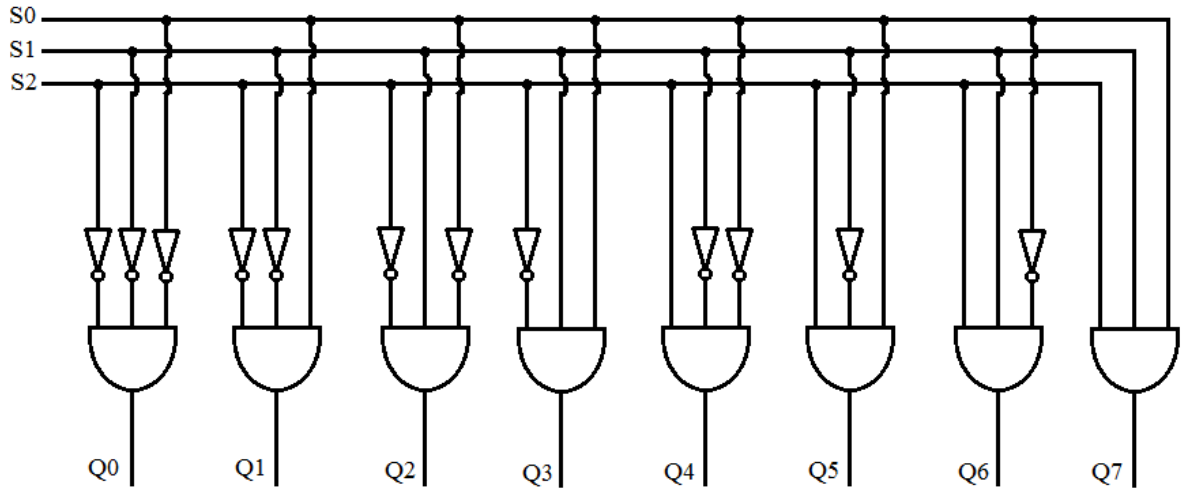
$Q_0 = \overline{A}.\overline{B}.\overline{C}$, $Q_1 = \overline{A}.\overline{B}.C$

$Q_2 = \overline{A}.B.\overline{C}$, $Q_3 = \overline{A}.B.C$

$Q_4 = A.\overline{B}.\overline{C}$, $Q_5 = A.\overline{B}.C$

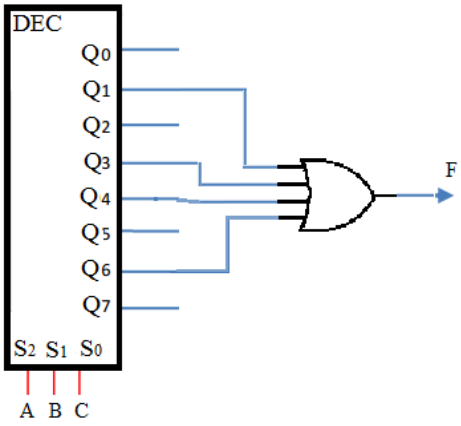
$Q_6 = A.B.\overline{C}$, $Q_7 = A.B.C$

Le logigramme des sorties est :



3.2. Réalisation d'une fonction logique par un DEM:

Puisque le DEC est similaire au DEM les étapes suivi pour réaliser une fonction logique sons les même. La fonction F de l'exemple précèdent peut être réalisé à l'aide d'un DEC comme suit :



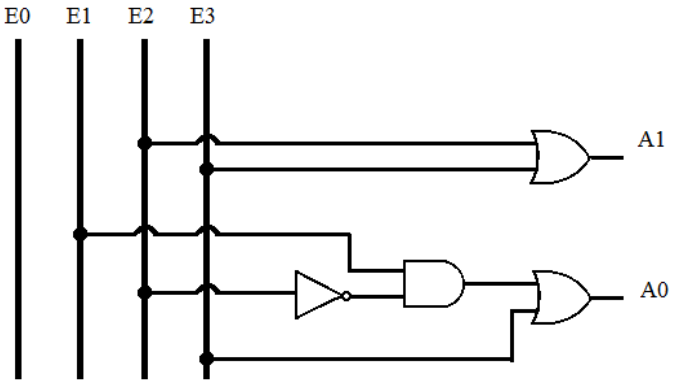
4. Codeur :

Le codeur est un circuit logique qui comporte N lignes d'entrées et n lignes de sorties. La sortie délivre le code de l'entrée active. S'il y a plusieurs l'entrée active le codeur donne en sortie le code de l'entrée prioritaire.

Si le code de la sortie est le binaire standard alors on a $N=2^n$ et le codeur est appelé codeur binaire. Il existe des codeurs pour le code DCB, GRAY, ASCII, …

Un codeur de priorité qui choisit le plus grand nombre lorsque plusieurs entrées sont activées à la fois. Le fonctionnement d'un codeur prioritaire à quatre entrées est décrit par la table de vérité suivante :

E0	E1	E2	E3	A1	A0
1	0	0	0	0	0
X	1	0	0	0	1
X	X	1	0	1	0
X	X	X	1	1	1

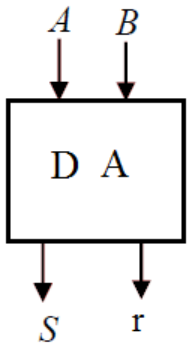


5. Additionneur :

Il existe deux types d'additionneurs, demi-additionneur (DA) et additionneur-complet (AC)

5.1. Demi-additionneur :

Un demi-additionneur réalise l'addition de deux bits A et B et produit la somme S et une retenue r.



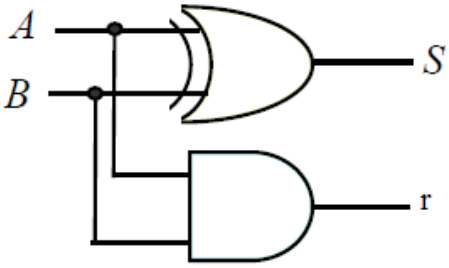
Voici les tables de vérité de s et c.

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

On déduit alors les expressions logiques de S et de r comme suit :

$S = A.\overline{B} + \overline{A}.B = A \oplus B$ et $r = A.B$

Le circuit logique du DA est donné par :

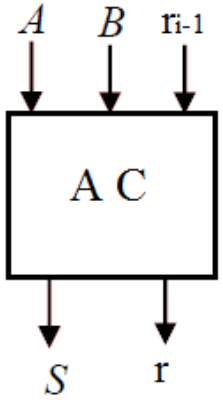


5.2. Additionneur complet :

Un additionneur complet est un circuit qui permet l'addition de deux chiffres binaires (Ai et Bi) et une retenue précédente r_{i-1}.

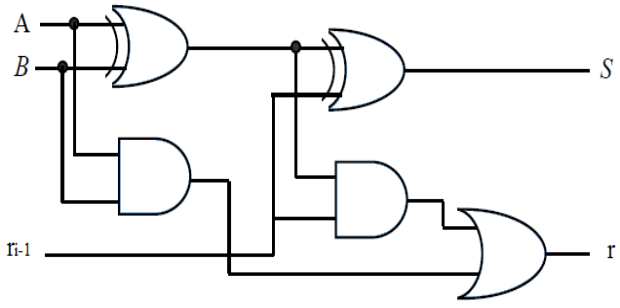
Sa table de vérité est :

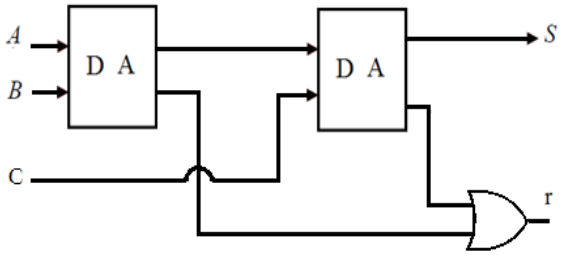
A	B	R_{i-1}	S_i	r
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



On déduit alors les expressions logiques de S et de r comme suit :

$S = A \oplus B \oplus r_{i-1}$ et $r = A.B + A.r_{i-1} + B.r_{i-1} + A.B.r_{i-1} = A.B + (A \oplus B).r_{i-1}$



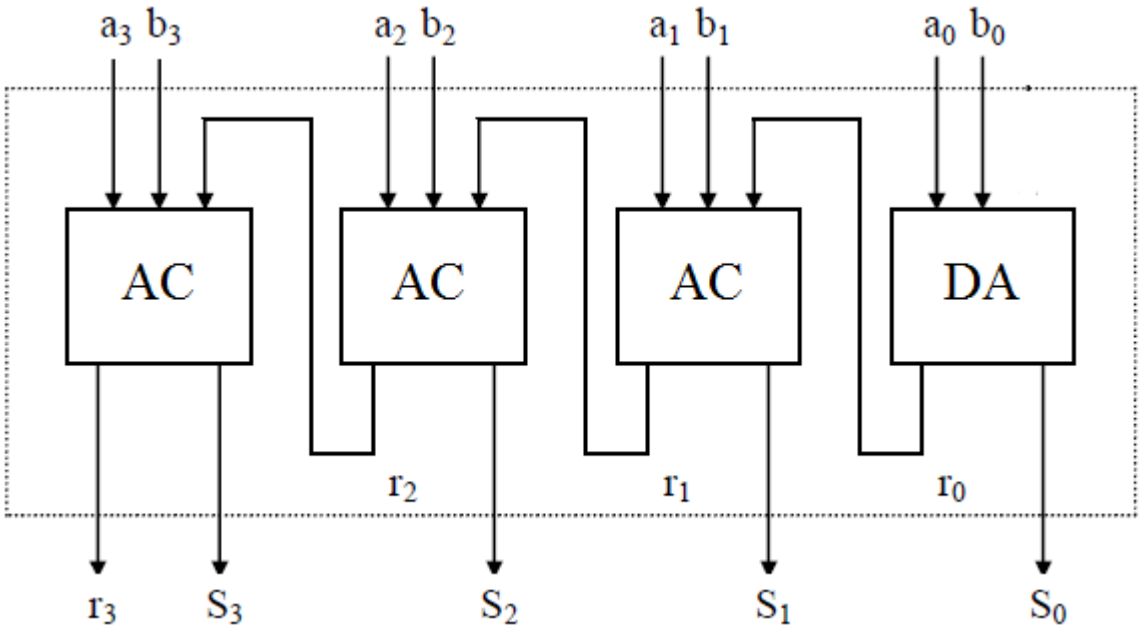


Pour réaliser une addition de deux nombres binaire, en utilise un DA pour le premier bit et des AC pour le reste, la sortie de retenue de chaque additionneur complet est connectée à l'entrée de retenue de l'étage de rang plus élevé suivant.

Considérons deux nombres binaires de 4 bits A et B comme entrées

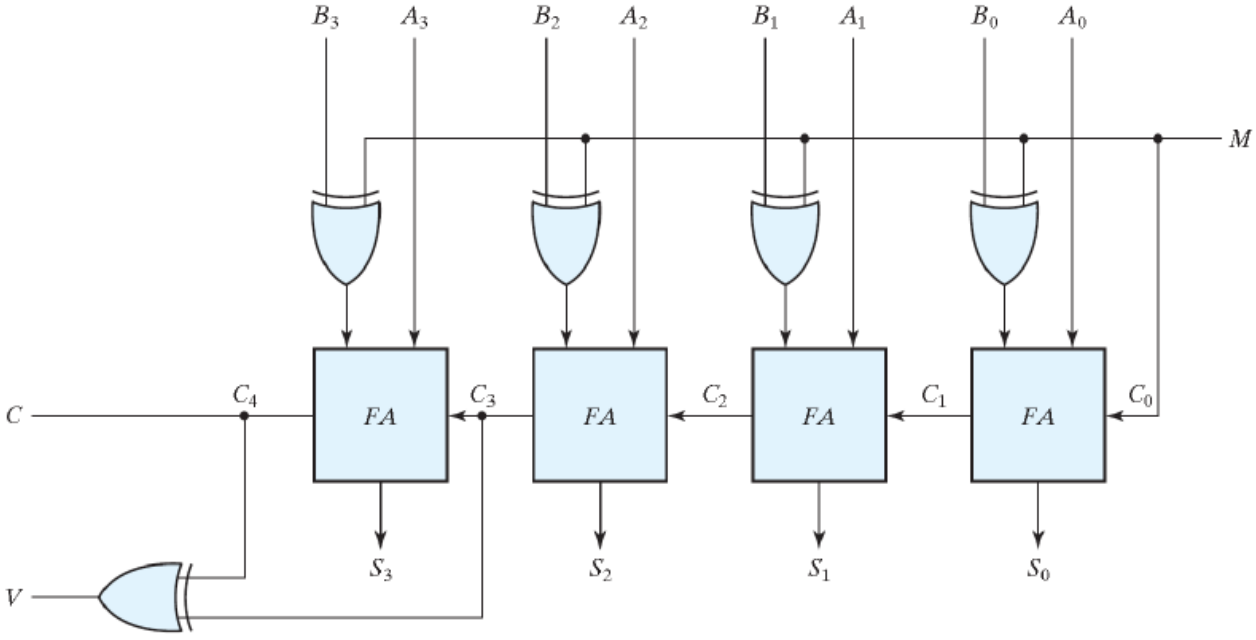
$$A=a_3\ a_2\ a_1\ a_0 \qquad \text{et} \qquad B=b_3\ b_2\ b_1\ b_0.$$

L'addition de deux nombres A et B peut être réalisé par le circuit suivant.



5.3. Additionneur-Soustracteur:

Un Additionneur/Soustracteur est circuit capable de réaliser à la fois l'addition et la soustraction de nombres binaires. L'opération en cours dépend de la valeur binaire que contient le signal de commande.



Le circuit se compose de 4 additionneurs complets puisque nous effectuons une opération sur des nombres à 4 bits. Il existe une ligne de contrôle M qui contient une valeur binaire de 0 ou 1 qui détermine que l'opération en cours est une addition ou une soustraction.

Comme le montre la figure, le premier additionneur complet a une ligne de contrôle directement comme entrée (entrée carry C_0), l'entrée A_0 (le bit le moins significatif de A) est directement entrée dans l'additionneur complet. La troisième entrée est $B_0 \oplus M$. Les deux sorties produites sont Sum/Difference (S_0) et Carry (C_1).

- Pour M=0 on a :

$B_0 \oplus 0 = B_0$ Alor $S_0 = A_0 + B_0 + 0 = A_0 + B_0$

Ceci suggère que lorsque M=0, l'opération effectuée sur les quatre nombres de bits est une addition.

- Pour M=1 on a :

$B_0 \oplus 1 = \overline{B}_0$ Alor $S_0 = A_0 + \overline{B}_0 + 1 = A_0 + (-B_0)$

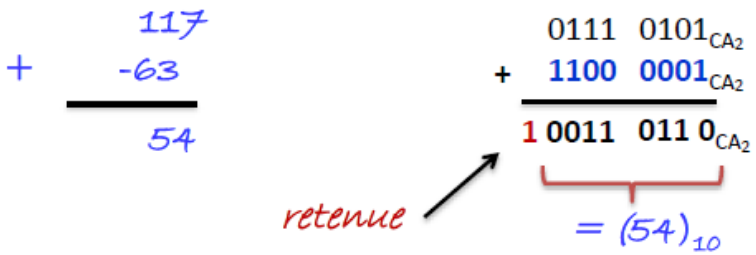
Ceci suggère que lorsque M=1, l'opération effectuée sur les quatre nombres de bits est une soustraction.

Notion de Carry (CF) et Overflow (OF)

Carry = Retenue :

Lors d'une ***opération arithmétique*** effectuée sur des nombres de ***n bits***, un ***(n+1)er bit*** peut être généré (**bit de carry**) :

Addition en CA_2 sur 8 bit



Overflow ou dépassement de capacité :

Lors d'une opération arithmétique mettant en jeu des nombres de n bits et de même signe, le résultat peut se révéler être trop grand ou trop petit pour être représentable sur les bits significatifs.

- Résultat est en dehors de l'intervalle des nombres représentables sur n bits ;
- Résultat erroné ;

- (+3) + (+2)

```
  0101
+ 0010
------
  0101
```

(+5) ← 0101

Pas de retenue : CF=0

No Overflow : OF=0

- (+3) + (+4)

```
  0101
+ 0110
------
  1011
```

Pas de retenue : CF=0

Overflow : OF=1

(-5) ← 1011

Résultat Faux

- (-3) + (-5)

```
  1101
+ 1011
------
1 1000
```

Retenue : CF=1

(A ignorer)

No Overflow : OF=0

(-8) ← 1000

Résultat vrai sur 4 bits

- (-3) + (-7)

```
  1101
+ 1001
------
1 0110
```

Retenue : CF=1

(A ignorer)

Overflow : OF=1

(+6) ← 0110

Résultat Faux sur 4 bit