Section : **S**	Option : ***Sciences de l'ingénieur***	Discipline : ***Génie Électrique***

Les opérations arithmétiques sur les nombres binaires

Domaine d'application : **Traitement programmé de l'information**	Type de document : **Cours**	Classe : **Terminale**	Date :

I – Principe de l'addition binaire

I – 1 – Table d'addition élémentaire

En binaire il existe seulement 2 chiffres : le 0 et le 1. Un nombre binaire est un ensemble de chiffres binaires, appelés des bits. Lors de l'addition de 2 chiffres binaires (de 2 bits) il n'y a que 3 possibilités (en effet en raison de la commutativité de l'addition on a 0 + 1 = 1 + 0) :

0 + 0 = 0

0 + 1 = 1

1 + 1 = 10 = 0 et on retient 1

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

S est la somme binaire des deux bits **A** et **B**, et **R** est la retenue éventuelle.

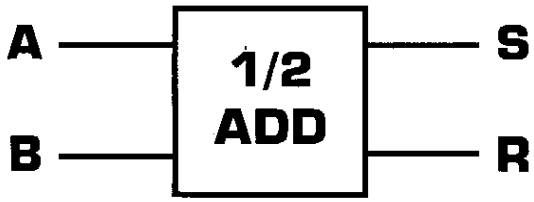
Exemples d'additions binaires :

$$\begin{array}{r} 1010 \\ +\ 0100 \\ \hline 1110 \end{array} \Leftrightarrow \begin{array}{r} 10 \\ +\ 4 \\ \hline 14 \end{array} \qquad \begin{array}{r} 1\ \ 1 \\ 11101 \\ +\ 01001 \\ \hline 100110 \end{array} \Leftrightarrow \begin{array}{r} 29 \\ +\ 9 \\ \hline 38 \end{array}$$

I – 2 – Le montage demi-additionneur

Un demi-additionneur est un montage logique permettant d'effectuer la somme binaire de 2 bits A et B, en générant une éventuelle retenue R.

Symbole du demi-additionneur :

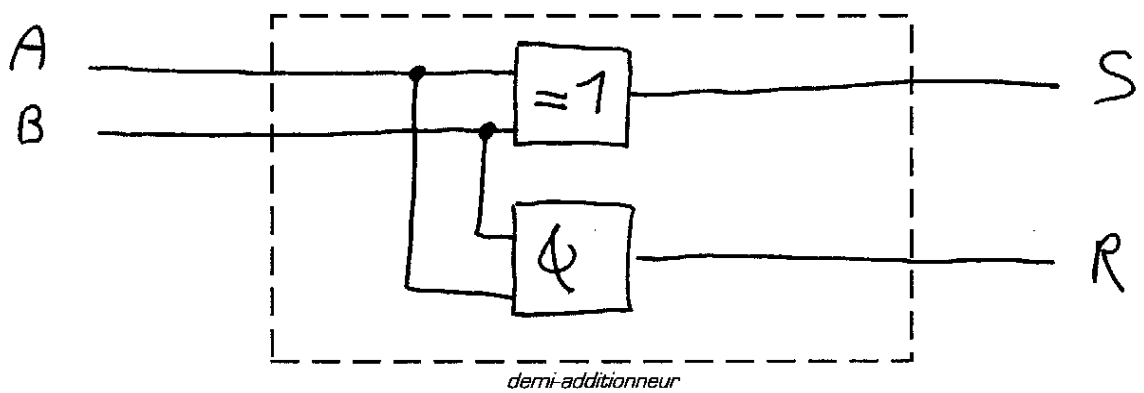


Equation des sortie d'après la table d'addition élémentaire :

S = $A \oplus B$

R = $A . B$

Structure interne d'un demi-additionneur :



demi-additionneur

Limites du ½ additionneur :

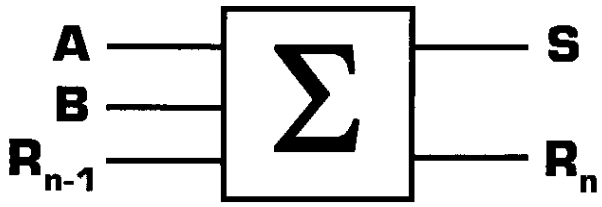
le demi-additionneur peut seulement additionner 2 bits | mais il ne permet pas d'additionner 2 nombres binaire | car il ne peut pas prendre en compte la retenue éventuelle | provenant de la somme des bits de poids inférieur. |

II – L'additionneur complet cascadable

II – 1 – Le montage additionneur complet

A la différence du ½ additionneur, l'additionneur complet dispose d'une 3ème entrée permettant de prendre en compte une éventuelle retenue. En mettant en cascade plusieurs additionneur complet il est alors possible d'effectuer l'addition [c'est-à-dire de calculer la somme] de 2 nombres binaires d'une taille quelconque.

Symbole de l'additionneur complet :



Remarque :

la lettre Σ [lettre grecque **Sigma** majuscule] au centre de l'additionneur symbolise la somme en arithmétique.

L'additionneur complet possède 3 entrées [A, B et R_{n-1}] et 2 sorties [S et R_n]. A et B sont les 2 bits de rang n à additionner, S est la somme des bits A, B et R_{n-1}, R_n est la retenue éventuelle de rang n, et R_{n-1} est la retenue éventuelle du rang précédent, c'est-à-dire la retenue du rang n-1.

L'additionneur complet doit calculer la somme de 3 bits : $A+B+R_{n-1}$. Le 0 étant l'élément neutre de l'addition, lors de l'addition de 3 bits un seul cas est nouveau par rapport à l'addition de 2 bits :

1 + 1 + 1 = 11 = 1 et on retient 1

Table de vérité de l'additionneur complet :

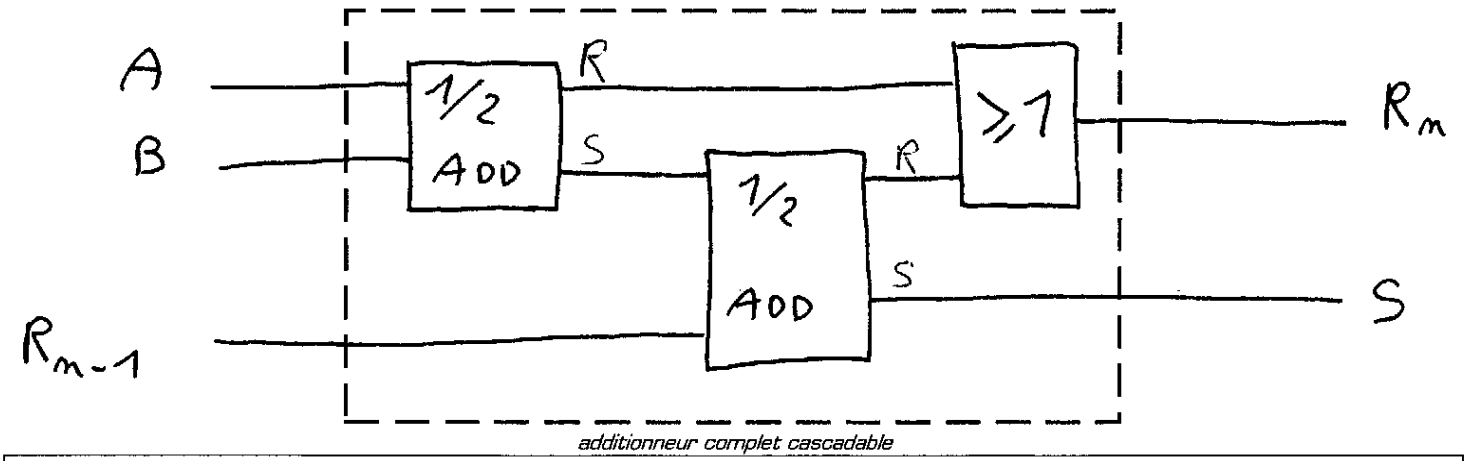
A	B	R_{n-1}	S	R_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Equations logiques des sorties :

$$S = A \oplus B \oplus R_{n-1}$$

$$R_n = A.B + R_{n-1} \cdot (A \oplus B)$$

Structure interne de l'additionneur complet cascadable :

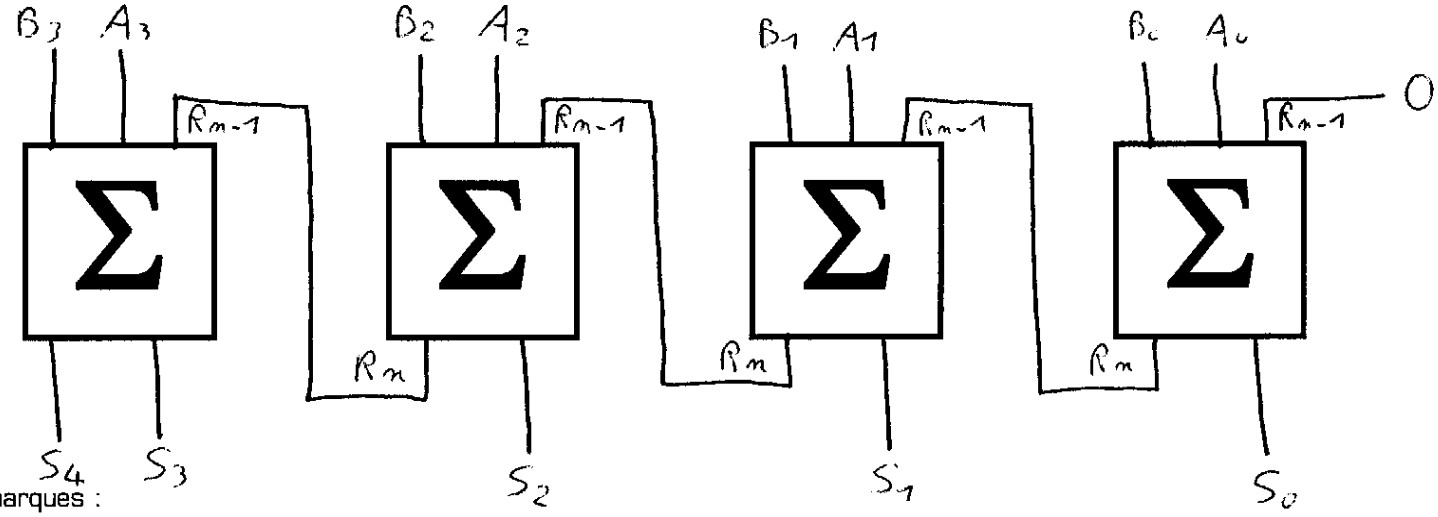


additionneur complet cascadable

II – 2 – Mise en cascade de plusieurs additionneurs complets

L'additionneur complet prenant en compte la retenue du rang précédent, il est cascadable. La mise en cascade permet d'effectuer une addition entre 2 nombres binaires d'une taille quelconque en connectant entre eux plusieurs additionneurs complet, chacun prenant en compte la retenue précédente.

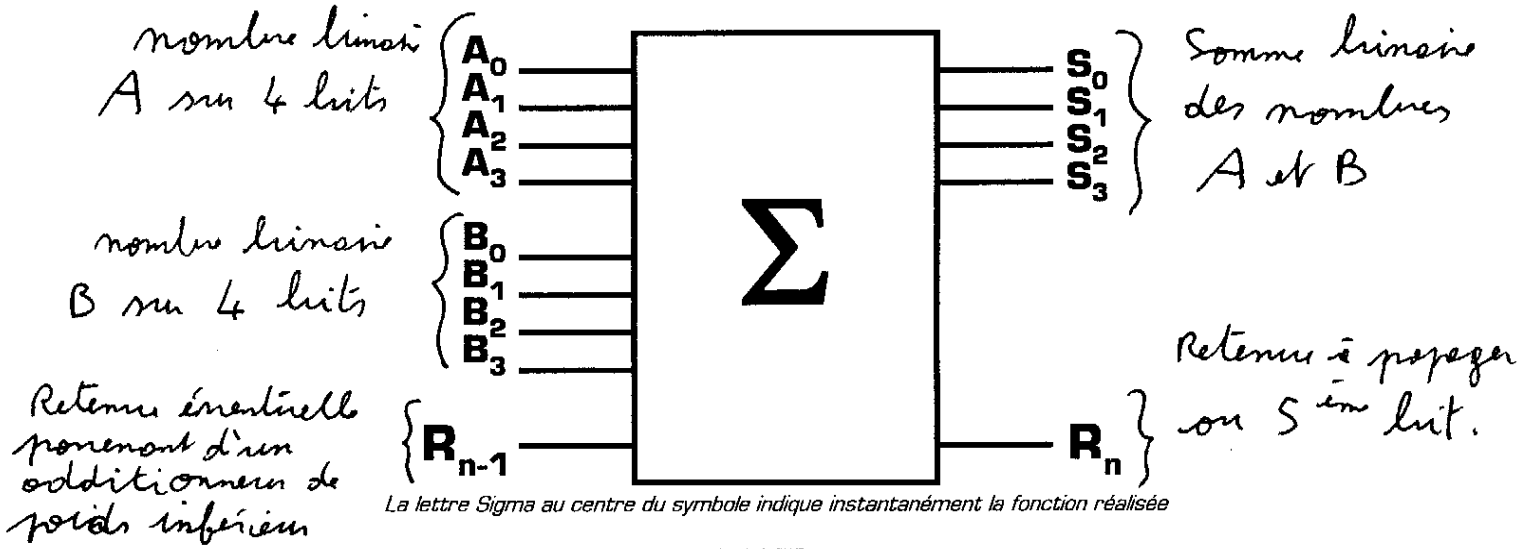
Exemple de mise en cascade de 4 additionneurs complets pour calculer la somme de deux nombres binaires de 4 bits chacun :



Remarques :

- A_3 A_2 A_1 et A_0 sont les 4 bits du 1er nombre binaire [A_0 est le LSB et A_3 est le MSB]
- B_3 B_2 B_1 et B_0 sont les 4 bits du 1er nombre binaire [B_0 est le LSB et B_3 est le MSB]
- Le résultat est composé des 5 bits S_4 S_3 S_2 S_1 et S_0
- L'entrée de retenue du premier additionneur [le poids faible] est mise à 0
- La sortie de retenue du dernier additionneur [le poids forts] constitue le 5ème bit du résultat
- Chaque entrée R_{n-1} est reliée à la sortie R_n de l'additionneur complet de rang inférieur
- Ce montage effectue l'addition bit à bit en tenant compte des retenues éventuelles, telle qu'on la fait à la main

Symbole générique d'un additionneur complet 4 bits :



La lettre Sigma au centre du symbole indique instantanément la fonction réalisée

III – Représentation des nombres négatifs en binaire

Rappel : on appelle **complément à 1** d'un nombre le nombre obtenu en complétant chacun de ses bits.

le complément à 1 du nombre N est noté $\overline{N}$

Exemple :

$23_{(10)} \equiv 10111_{(2)}$ donc $\overline{23} \equiv 01000_{(2)}$

le complément à 1 de 23 est 01000.

$52_{(10)} \equiv 110100_{(2)}$

donc $\overline{52} \equiv 001011_{(2)}$

Limites du complément à 1 :

Le complément à 1 ne permet pas de coder les nombres négatifs car le code obtenu peut représenter celui d'un nombre positif.

Exemple : $9_{(10)} \equiv 1001_{(2)}$ $\overline{9}_{(10)} \equiv 0110_{(2)}$ → on obtient le code de $6_{(10)}$

$6_{(10)} \equiv 110_{(2)}$ $\overline{6}_{(10)} \equiv 001_{(2)}$ si on a 3 bits → code du 1

$\equiv 1001_{(2)}$ si on a 4 bits → code du 9

Pour représenter les nombres négatifs en binaire il y a deux solutions.

III - 1 - Solution 1

On ajoute un bit de signe à gauche de la représentation binaire du nombre :

- Si le bit de signe = 0 alors le nombre est **positif**
- Si le bit de signe = 1 alors le nombre est **négatif**

Exemple : 12 s'écrit 0 | 1100 (signe | nombre)

−12 s'écrit 1 | 1100 (signe | nombre)

Avantage : cette représentation permet de distinguer sans ambiguïté les nombres positifs et les nombres négatifs.

Limites de la représentation des nombres négatifs avec un bit de signe :

Ce codage ne permet pas d'effectuer des calculs avec des nombres négatifs. La représentation avec le bit de signe sera donc utilisé pour coder simplement des entiers relatifs lorsqu'on n'a aucun calcul à effectuer.

Exemple : pour afficher la valeur numérique d'une température on pourra utiliser un bit de signe pour coder les nombres relatifs. Dans ce cas le bit de signe représente l'état du signe moins de l'afficheur.

III - 2 - Solution 2 : le complément à 2

Le complément à 2 d'un nombre N est égal à la somme du complément à 1 de ce nombre et de 1. On l'utilise pour coder les nombres négatifs et on le note − N.

$$-\mathbf{N} = \overline{N} + 1$$

Avantage du complément à 2 : le complément à 2 permet de représenter des nombres signés [positifs et négatifs] sans ambiguïté. De plus il permet d'effectuer des opérations [somme et différence] sur les nombres relatifs en utilisant un simple additionneur.

Inconvénient du complément à 2 : il est plus difficile d'effectuer $\overline{N} + 1$ que de coder $\overline{N}$ ou d'ajouter simplement un bit de signe. La fonction complément à 2 ne fait pas partie des fonctions de base de la logique ou de l'arithmétique.

Exemple d'application du complément à 2 : pour effectuer la soustraction A − B on va ajouter à A le complément à 2 de B : $A - B = A + \overline{B} + 1$

on peut effectuer la soustraction 53-17

Exemples numériques :

$53_{(10)} \equiv 110101_{(2)}$

$17_{(10)} \equiv 010001_{(2)}$

$\overline{17}_{(10)} \equiv 101110_{(2)}$

$-17 = \overline{17}+1 \equiv 101111$

Pour calculer 53-17 il faut additionner 110101 et 101111 :

$$\begin{array}{r} 1\,1\,1\,1\,1 \\ 1\,1\,0\,1\,0\,1 \\ +\,1\,0\,1\,1\,1\,1 \\ \hline 1\,1\,0\,0\,1\,0\,0 \end{array} \qquad \begin{array}{r} 53 \\ +\ (-17) \\ \hline 36 \end{array}$$

Résultat sur 6 bits

autre exemple :

6-9 :

0110 → 6

+0111 → $\overline{9}+1$ → -9

1101 → $\overline{3}+1$ → -3

1101-1 = 1100

$\overline{1100}$ = 0011 ≡ 3

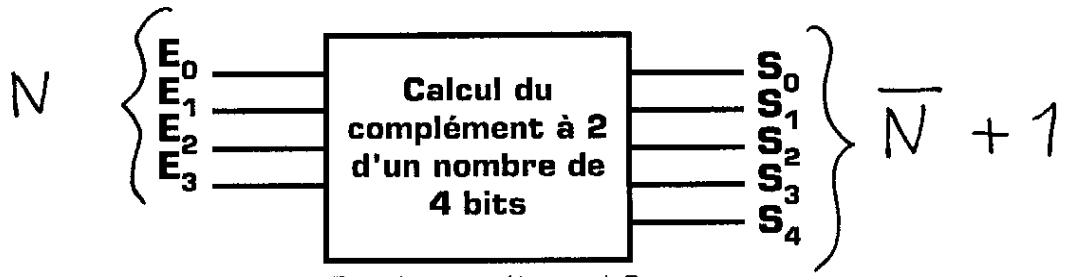
ATTENTION : tous les nombres, y compris le résultat, doivent être exprimés dans le même format (par exemple sur 4 bits).

Remarque : le complément à 2 du complément à 2 du nombre N est le nombre N lui même. La fonction réciproque du complément à 2 est donc la fonction complément à 2 elle-même. On dit alors que la fonction complément à 2 est une **involution** :

$$-(-N) = N \quad \text{ou encore} \quad \overline{\overline{N}+1}+1 = N$$

IV - Montage calculant le complément à 2 d'un nombre

On désire réaliser un montage logique fournissant en sortie le complément à 2 d'un nombre N donné sur 4 bits à l'entrée du montage :



Fonction complément à 2

Remarque : si N est sur 4 bits son complément à 2 est sur 5 bits.

Pour réaliser la fonction complément à 2, il y a deux solutions.

IV - 1 - Solution 1

On considère que la fonction complément à deux est un montage en logique combinatoire classique (comme un transcodeur) et on recherche les équations de chacune des sorties en fonction des entrées.

Table de vérité de la fonction complément à 2 :

N	E0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
	E1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	E2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	E3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$\overline{N}+1$	S0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
	S1	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	S2	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0
	S3	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
	S4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Equations immédiate d'après la table de vérité :

$S_0 = E_0$ $\qquad$ $S_4 = \overline{E_0}.\overline{E_1}.\overline{E_2}.\overline{E_3}$

Voir en bas de la page 8/8 pour l'analyse poussée détaillée.

Pour les sorties S1, S2 et S3, une analyse poussée de la table de vérité permet d'en dégager les équations optimisées. Ces équations contenant des ou-exclusifs, l'utilisation des tableaux de Karnaugh ne serait pas optimale :

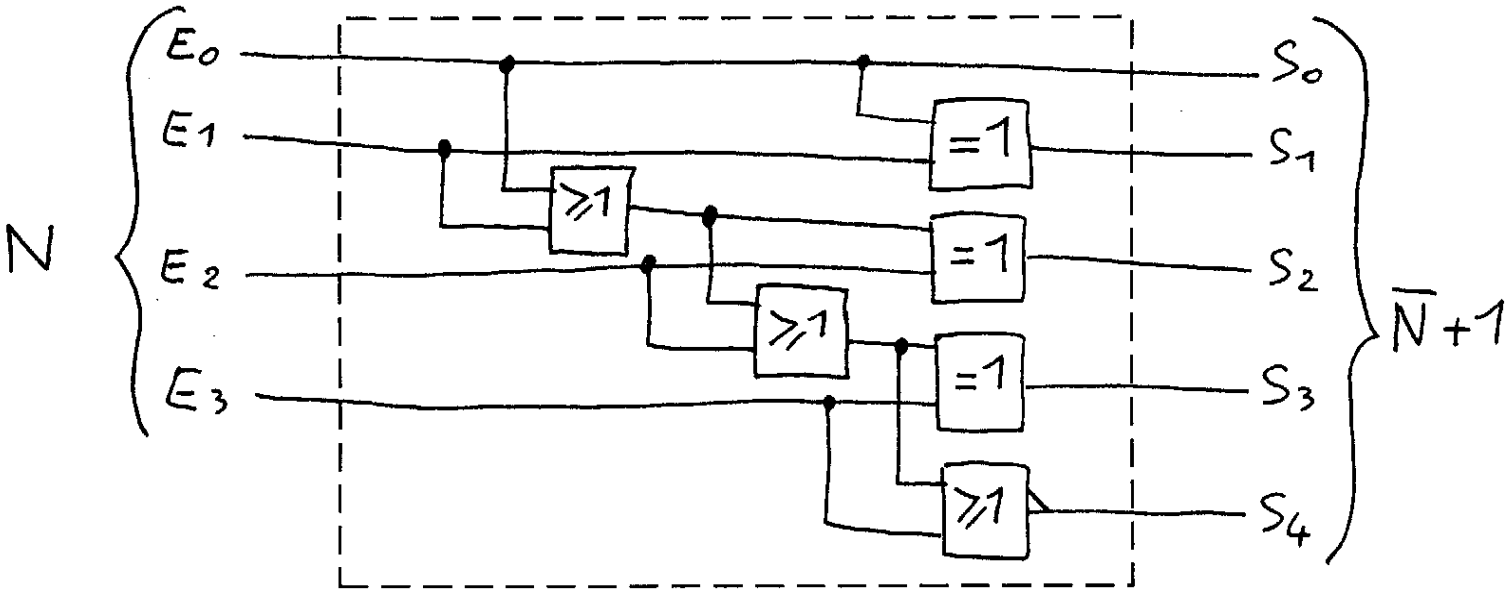
$$S_1 = E_0 \oplus E_1$$

$$S_2 = (E_0 + E_1) \oplus E_2$$

$$S_3 = (E_0 + E_1 + E_2) \oplus E_3$$

Les tableaux de Karnaugh peuvent permettre d'aboutir à ces équations optimisées, mais à condition de factoriser et de reconnaître les ou-exclusifs ...

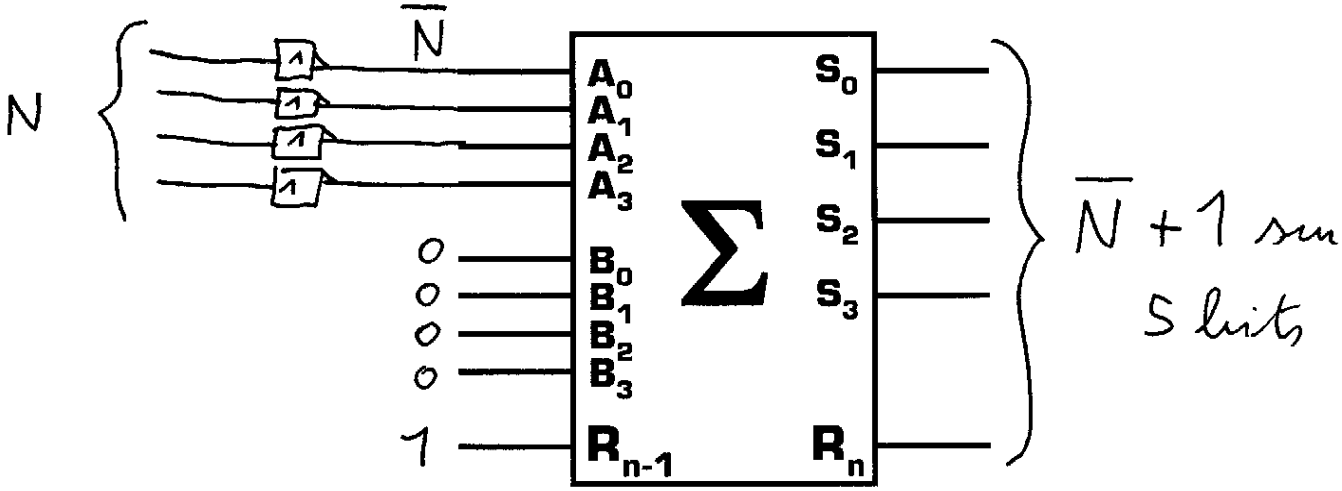
Logigramme de la fonction « Complément à 2 » :



Fonction complément à 2 d'un nombre de 4 bits

IV - 2 - Solution 2

Une autre solution pour calculer le complément à 2 consiste à effectuer l'opération $\overline{N} + 1$ en utilisant un additionneur 4 bits :

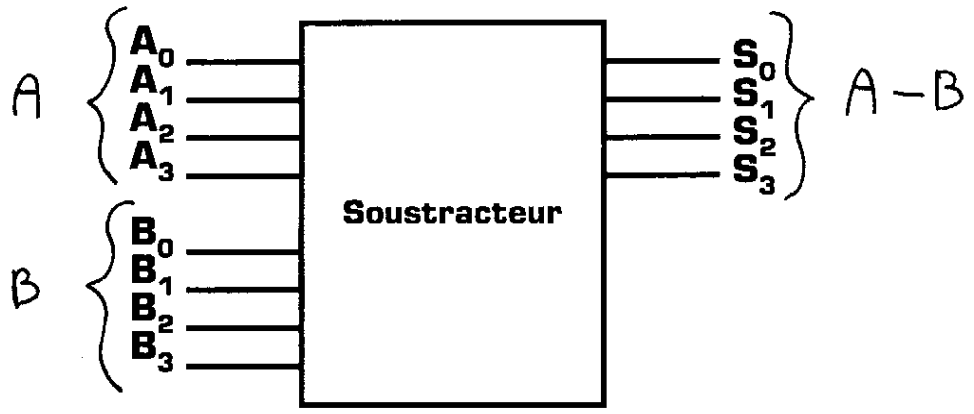


Utilisation d'un additionneur pour calculer le complément à 2

Comparaison des deux solutions : l'additionneur complet 4 bits est composé de 20 portes logiques en interne. Avec les 4 portes NON, la solution 2 coûte 24 portes logiques au total (contre 6 portes logiques pour la solution 1). La solution 1 sera donc préférable si on doit réaliser la fonction complément à 2 « à partir de rien », alors que la solution 2 sera mise en œuvre lorsqu'on dispose déjà d'un additionneur complet 4 bits « tout fait ».

V - Montage soustracteur

On désire réaliser un montage logique effectuant la différence A – B entre deux nombres binaires positifs exprimés chacun sur 4 bits :



Fonction soustracteur

Pour réaliser cette fonction soustracteur il y a 2 solutions :

V - 1 - Solution 1

On réalise un demi-soustracteur, puis un soustracteur cascadable, et enfin un soustracteur complet par mise en cascade du précédent.

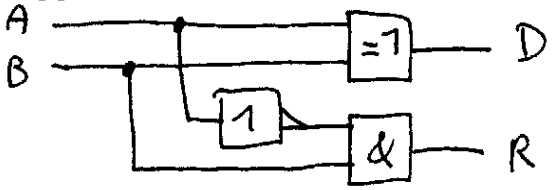
A	B	D	R
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Equations des sorties :

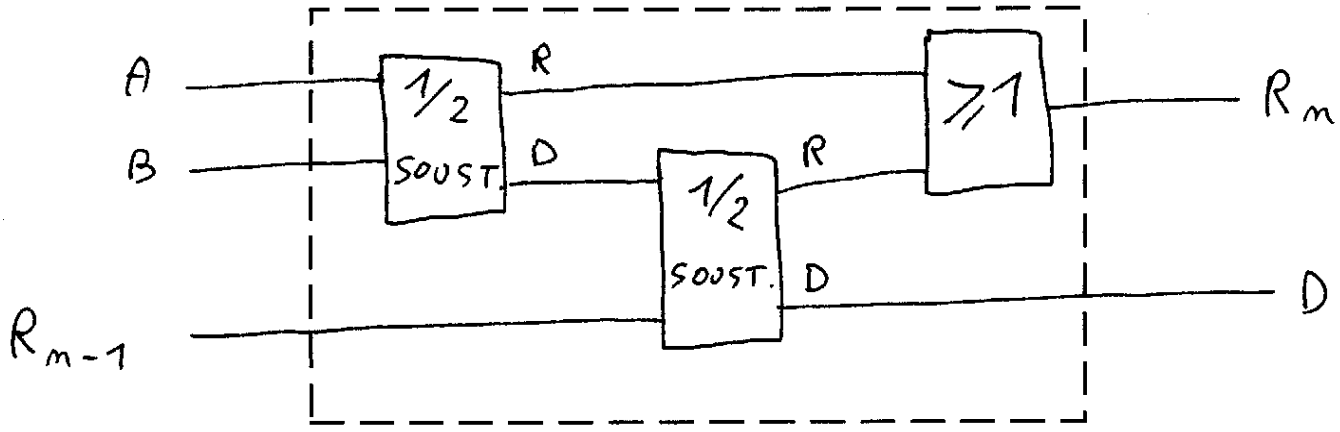
$D = A \oplus B$

$R = \overline{A} \cdot B$

Logigramme du demi-soustracteur :

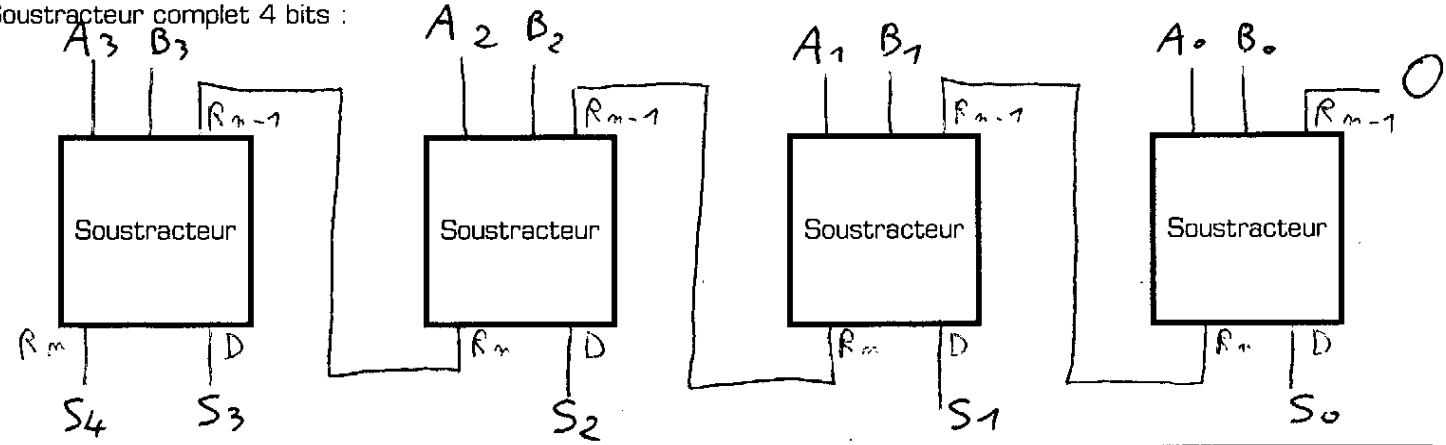


Soustracteur cascadable élémentaire :



Soustracteur cascadable

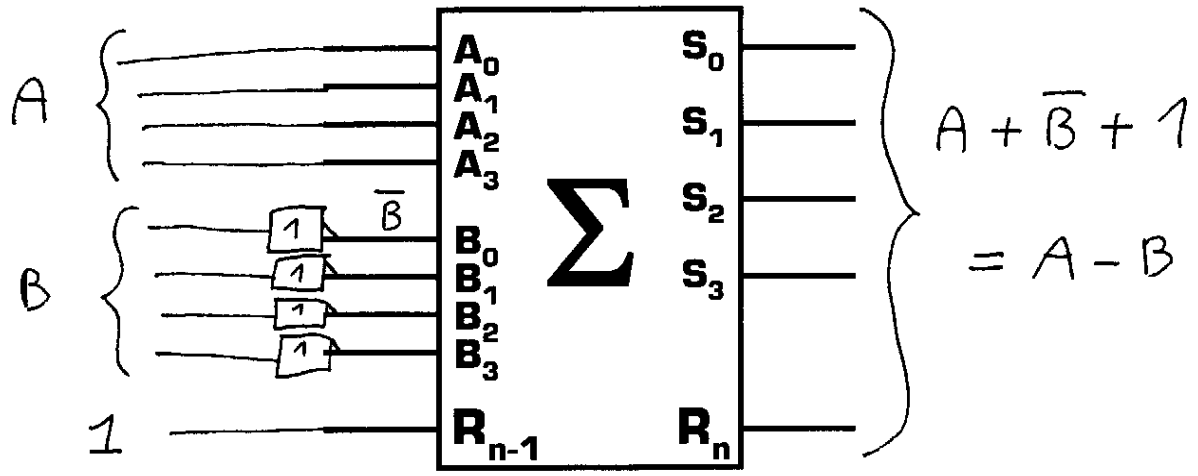
Soustracteur complet 4 bits :



V - 2 - Solution 2

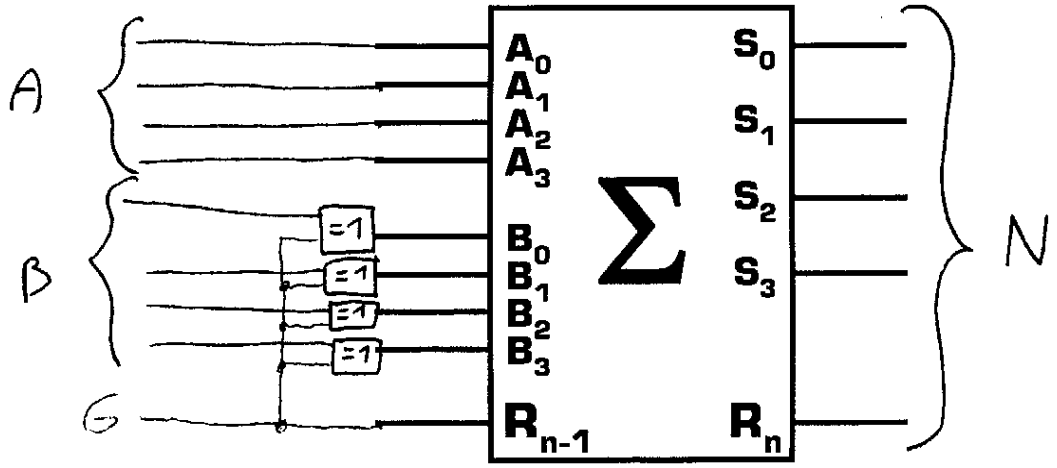
On utilise un additionneur complet 4 bits avec lequel on effectue l'opération A + $\overline{B}$ + 1

Rappel : A – B = A + [– B] = A + [$\overline{B}$ + 1]



Utilisation d'un additionneur effectuer une soustraction

Idée et extension possible : comment réaliser un montage additionneur / soustracteur qui effectue soit la somme soit la différence entre 2 nombres A et B, en fonction de l'état d'une entrée de commande G ? Solution : on va modifier le montage précédent en utilisant des portes Ou-Exclusifs en tant qu'inverseurs commandés afin de calculer soit A + B, soit A + $\overline{B}$ + 1, en fonction de l'entrée G :



Fonction addition / soustraction

✱ Si G = 0 alors **N = A + B** : le montage est un ***additionneur***

✱ Si G = 1 alors **N = A – B** : le montage est un ***soustracteur***

Notes personnelles :

Page 6 :

✱ $S_1 = E_1$ si $E_0 = 0$ et $S_1 = \overline{E_1}$ si $E_0 = 1$: le second bit est recopié si $E_0 = 0$ ou complémenté si $E_0 = 1$. On en déduit que : $S_1 = E_0 \oplus E_1$

✱ le 3ème bit est recopié si E_0 et E_1 sont tous les deux à 0, ou complémenté si un moins l'un des deux est à 1. Donc $S_2 = (E_0 + E_1) \oplus E_2$

✱ de même pour le 4ème bit : $S_3 = (E_0 + E_1 + E_2) \oplus E_3$

Correction scannée et imprimée le 18 février 2012