Correction PCD5 ELP111 : Additionneur binaire

Section 3.1 : Addition

Rappels :

- La retenue sortante notée Carry n'est utilisée que dans le cas d'addition de nombres non signés codés en binaire naturel.
- Le bit de dépassement (OVR) est utilisée dans le cas de l'addition de nombres signés représentés en CA2 et lorsqu'il n'y a pas d'extension de bit de signe.

A	B	A+B	Carry	OVR
1100	0010	1110	0	0
1010	1001	0011	1	1
0010	0111	1001	0	1
0011	1111	0010	1	0

Ligne 1

```
    1  1  0  0      12 en binaire naturel,  -4 en CA2
+   0  0  1  0       2 en binaire naturel,  +2 en CA2

 0(C)1  1  1  0      14 en binaire naturel,  -2  en CA2 sur 4 bits
```

Le résultat est correct sur 4 bits en CA2, il n'y a pas de dépassement de capacité

Ligne 2

```
    1  0  1  0      10  en binaire naturel,  -6 en CA2
+   1  0  0  1       9 en binaire naturel,  -7  en CA2

 1(C)0  0  1  1      19 en binaire naturel (en utilisant Carry, C)
                      +3  en CA2 si on utilise le résultat sur 4 bits
                     On a OVR = 1 car le résultat n'est pas correct sur 4 bits en CA2
```

Ligne 3

```
    0  0  1  0      2 en binaire naturel,  +2  en CA2
+   0  1  1  1      7  en binaire naturel,  +7 en CA2

 0(C)1  0  0  1      9 en binaire naturel,  -7  en CA2 sur 4 bits

                     OVR = 1 car le résultat n'est pas correct sur 4 bits en CA2
```

Ligne 4

```
      0  0  1  1      3 en binaire naturel,  +3 en CA2
+     1  1  1  1      15 en binaire naturel,  -1 en CA2

   1(C)0  0  1  0     18 en binaire naturel (en utilisant carry)
                      +2 en CA2 sur 4 bits
```

OVR = 0 car le résultat est correct sur 4 bits en CA2

Soustraction

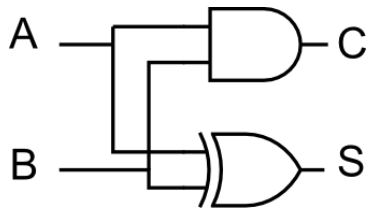
+A **A3 A2 A1 A0**	**+B** **B3 B2 B1 B0**	**+A+(-B)** **S3 S2 S1 S0**	**OVR**
1100	0010	1010	0
1010	1001	0001	0
0010	0111	1011	0
0011	1111	0100	0

Section 3.2

Le demi-additionneur binaire est un circuit qui prend en entrée 2 nombres de 1 bit (A_k et B_k) et génère leur somme (S_k) et une retenue (C_k).

A_k	B_k	S_k	C_k
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Le schéma ci-après représente la structure d'un "demi-additionneur binaire" :



A partir des informations ci-dessus, on souhaite réaliser un additionneur complet qui prenne en compte une retenue (C_{k-1}) générée à partir d'un étage précédent.

Section 3.2.1

C'est une addition de 3 bits, dont la somme est Sk et la retenue sortante est Ck

Ak	Bk	Ck-1	Sk	Ck
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

$$S_k = \overline{A_k}\,\overline{B_k}\,C_{k-1} + \overline{A_k}\,B_k\overline{C_{k-1}} + A_k\overline{B_k}\,\overline{C_{k-1}} + A_kB_kC_{k-1}$$
$$C_k = \overline{A_k}\,B_k\,C_{k-1} + A_k\overline{B_k}C_{k-1} + A_kB_k\overline{C_{k-1}} + A_kB_kC_{k-1}$$

Section 3.2.2

Tableau de Karnaugh de Sk

Ck-1 \ Ak	Ak		Ak	
	0	1	0	1
Ck-1	1	0	1	0

Bk

Equation de Sk

$$S_k = C_{k-1} \oplus A_k \oplus B_k$$

La fonction logique de Sk est composée de 2 XOR, chacun étant présent dans le ½ additionneur. En effet, pour obtenir le résultat de la somme, nous devons tout d'abord sommer Ak et Bk puis le résultat de cette somme est sommé avec Ck-1.

Equation de Ck

Pour obtenir Ck, il faut manipuler sa 1ere forme canonique et faire apparaître le XOR.

Ainsi on peut remarquer que :

$$C_k = \overline{C_{k-1}}A_kB_k + C_{k-1}\overline{A_k}B_k + C_{k-1}A_k\overline{B_k}$$
$$C_k = A_kB_k(\overline{C_{k-1}} + C_{k-1}) + C_{k-1}(\overline{A_k}B_k + A_k\overline{B_k})$$
$$C_k = A_kB_k + C_{k-1}(A_k \oplus B_k)$$

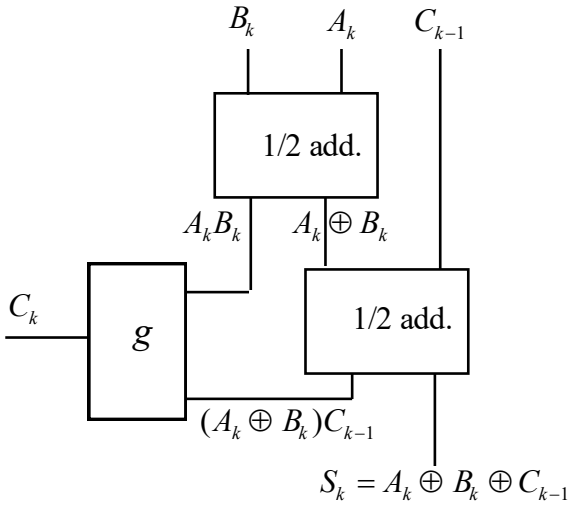
Remarque : L'utilisation du tableau de Karnaugh donne un résultat correct de la fonction simplifiée égal à

$C_k = A_kB_k + C_{k-1}B_k + C_{k-1}A_k$

$C_k = A_kB_k + C_{k-1}(A_k + B_k)$

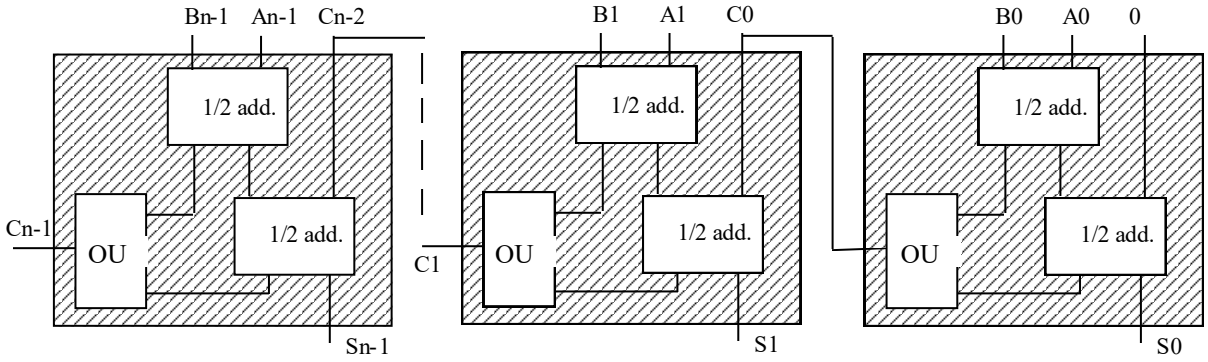
mais ne permet pas de ré-utiliser le matériel (c.a.d. les portes logiques) présent dans le ½ additionneur.

Section 3.2.3



L'identification de l'équation établie en question 3.2.2 nous montre que le bloc g est la fonction OU.

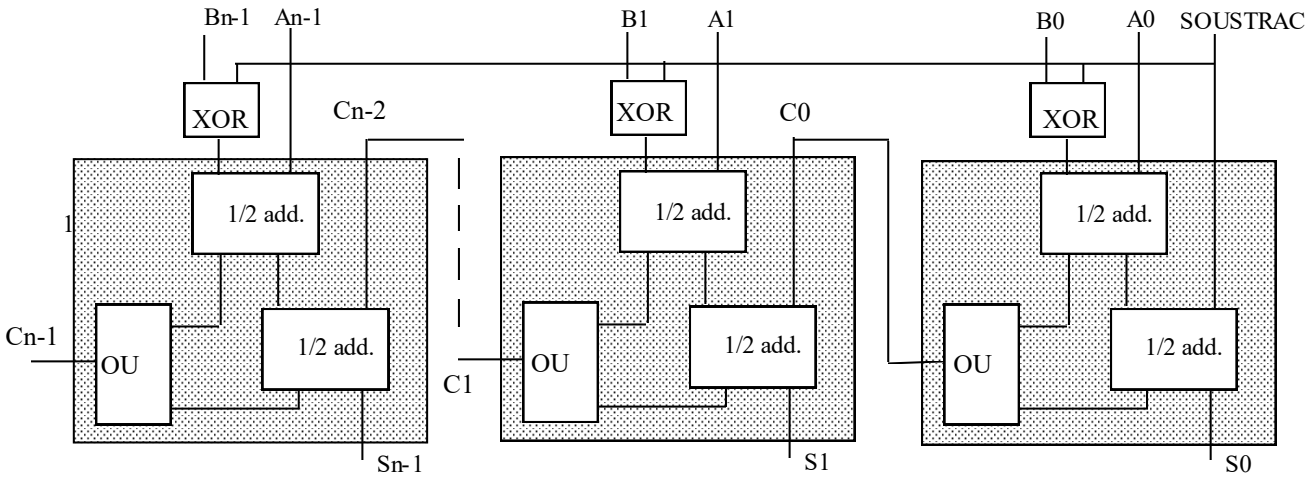
Section 3.2.4 :



N.B. Le temps de calcul est proportionnel au nombre d'additionneurs élémentaires, à cause de la retenue propagée.

Section 3.2.5 :

Pour effectuer la soustraction *A - B* il faut complémenter chacun des bits formant le nombre B à l'aide de portes OUEX, puis rajouter le nombre 1 par l'intermédiaire de l'entrée de retenue.



Section 3.2.6 :

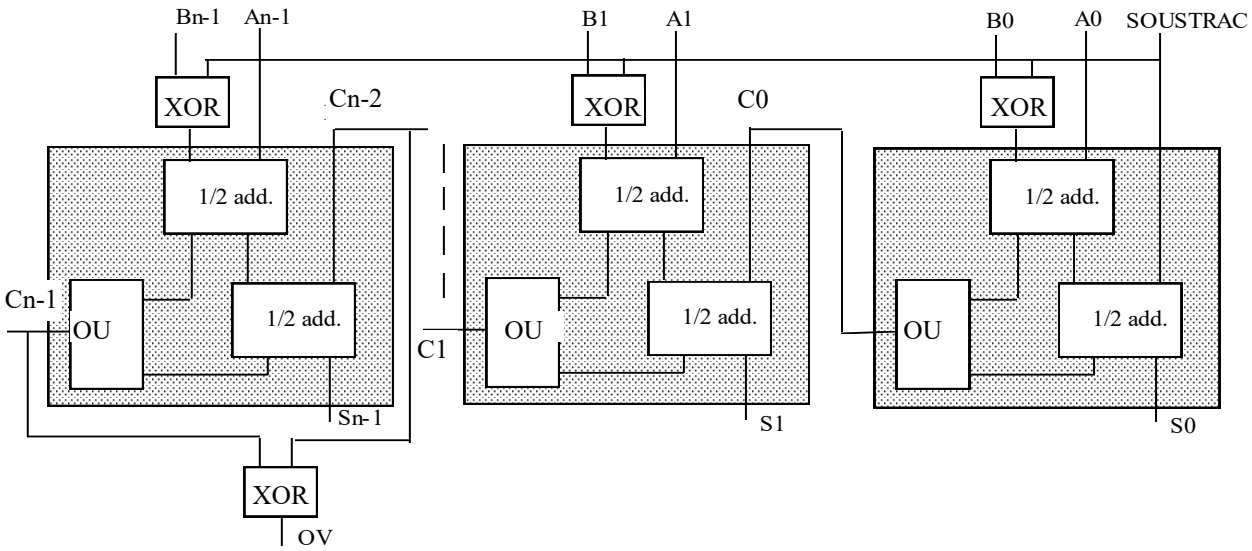
Il y aura débordement de capacité quand les bits de signe des deux opérandes à l'entrée de l'additionneur (c'est-à-dire les bits de poids fort (MSB) dans la représentation en complément à 2) sont égaux entre eux mais différents de celui du résultat *S* ($A_{n-1} = B_{n-1} \oplus SOUSTRAC \neq S_{n-1}$).

Cn-2	*An*-1	*Bn*-1⊕*SOUSTRAC*	*Sn*-1	*Cn*-1	*OVF*
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	0	1	1
1	0	0	1	0	1
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	1	1	0

Sur cette table nous voyons que pour obtenir ($A_{n-1} = B_{n-1} \oplus SOUSTRAC \neq S_{n-1}$) , il faut que *Cn*-2 soit différent de *Cn*-1 d'où l'équation de l'indicateur *OVR* :

$$OVR = C_{n-2} \oplus C_{n-1}$$

d'où le schéma final :


Bn-1
An-1
B1
A1
B0
A0
SOUSTRAC
XOR
XOR
XOR
Cn-2
C0
1/2 add.
1/2 add.
1/2 add.
Cn-1
OU
1/2 add.
OU
1/2 add.
OU
1/2 add.
C1
Sn-1
S1
S0
XOR
OV