Architecture des ordinateurs - TD 06

1 Bref retour sur IEEE754

1. Coder en IEEE754 demi-précision (1 bit de signe, 5 bits exposant, 10 bits mantisse), les nombres suivants : 1.3 et 4.5

Solution:

```
13/10

1101    | 1010
1010    |------
----v   | 1.01001100110011...
001100
  1010
  ----v
  0010000 <-.
     1010   |
     ----v  |
     01100  |
      1010  |
      ----  |
      0010 -.

positif  exposant(0)  mantisse
0        01111        0100110011


4.5 --> 100.1
    --> 1.001 . 2^2

positif exposant(2)   mantisse
0       10001         0010000000
```

2. Calculer leur somme en demi-précision.

Solution: On réecrit le premier nombre avec un exposant de 2 :

$$1.0100110011 \times 2^0 = 0.010100110011 \times 2^2$$

Avec un arrondi au plus près : 0.0101001100
On fait la somme des deux mantisses :

```
0.0101001100
1.0010000000
------------
1.0111001100
```

```
Le résultat est déjà normalisé,
0 10001 0111001100
```

3. Calculer leur produit en demi-précision.

Solution: On fait la somme des exposants : $2 + 0 = 2$
On fait le produit des mantisses :

```
    1.0100110011
    x      1.001
    ------------
     10100110011
  10100110011000
  --------------
  10111011001011
             ---

Arrondi au plus près cela fait 1.0111011001

Résultat: 0 10001 0111011001
```

2 Afficheur 7 segments

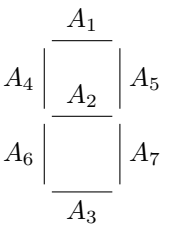
On souhaite réaliser un circuit pour afficher les chiffres de 0 à 9 sur un afficheur 7 segments (voir figure ci-dessous). Un afficheur 7 segments est composé de 7 leds commandées par 7 signaux distincts $A_1 \dots A_7$. Le chiffre à afficher est codé en BCD, chaque bit correspond à un signal b_3 à b_0.

Nous souhaitons réaliser le circuit permettant de commander le signal A_3 correspondant à l'allumage du segment inférieur.

Codage BCD

déc.	$b_3b_2b_1b_0$
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Afficheur 7 segments



1. Écrire la table de vérité du signal A_3. Lorsque la valeur de A_3 est indeterminée écrire X (par exemple pour B=13).

Solution:

b_3	b_2	b_1	b_0	A_3
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

2. Écrire le tableau de Karnaugh correspondant à la table de vérité de A_3.

Solution:

b_3b_2 \ b_1b_0	00	01	11	10
00	1	0	1	1
01	0	1	0	1
11	X	X	X	X
10	1	1	X	X

3. Simplifier l'expression. Proposer un circuit pour commander A_3.

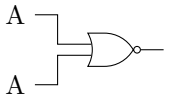
Solution:

$$b_3 + b_1.\overline{b_0} + \overline{b_0}.\overline{b_2} + b_2.\overline{b_1}.b_0 + \overline{b_2}.b_1$$

3 Circuits basiques

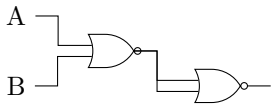
1. Fabriquer avec uniquement des portes NOR les circuits suivants :
   — porte NON

   Solution: $\overline{(A+A)} = \overline{A}$

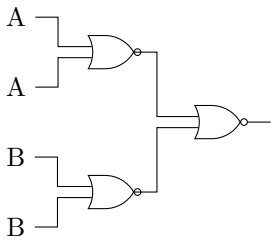
   — porte OU

   Solution: $\overline{\overline{(A+B)}} = A+B$

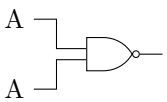
   — porte ET

   Solution: $\overline{(\overline{A}+\overline{B})} = A.B$

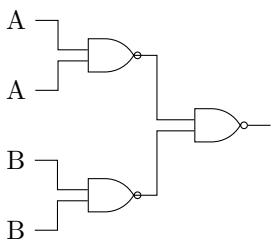
   Qu'en deduisez vous ?
2. Fabriquer avec uniquement des portes NAND les circuits suivants :
   — porte NON

   Solution: $\overline{(A.A)} = \overline{A}$

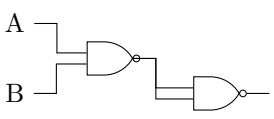
   — porte OU

   Solution: $\overline{(\overline{A}.\overline{B})} = A+B$

   — porte ET

   Solution: $\overline{\overline{(A.B)}} = A.B$

— porte OU à quatre entrées

Solution: $A + B + C + D = (A + B) + (C + D) \rightarrow 9$ portes NAND

— porte ET à quatre entrées

Solution: $A.B.C.D = (A.B).(C.D) \rightarrow 9$ portes NAND

3. Donner un circuit permettant de tester l'égalité de deux nombres de 4 bits.

Solution: Pour tester l'égalité de deux bits a_0 et b_0, on utilise la fonction $XNOR = \overline{a_0 \oplus b_0}$.
Pour tester l'égalité de 4 bits on réalise le circuit :

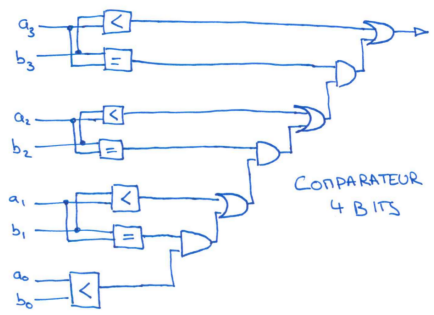
$$(a_0 XNOR b_0).(a_1 XNOR b_1).....(a_3 XNOR b_3)$$

4. Soit deux nombres positifs : $A = a_3a_2a_1a_0$ et $B = b_3b_2b_1b_0$, implémenter le circuit de la fonction $C(A, B) = A < B$.

Solution: A < B il faut considérer trois cas
— SI $a_3 < b_3$ alors VRAI
— SI $a_3 = b_3$ alors il faut comparer $a_2a_1a_0$ et $b_2b_1b_0$.
— SINON FAUX
En développant ce raisonnement on obtient :

$$\begin{aligned} A < B =& (a_3 < b_3) + (a_3 = b_3).(a_2a_1a_0 < b_2b_1b_0) && (1)\\ =& (a_3 < b_3) + (a_3 = b_3).((a_2 < b_2) + (a_2 = b_2).(a_1a_0 < b_1b_0)) && (2)\\ =& (a_3 < b_3) + (a_3 = b_3).((a_2 < b_2) + (a_2 = b_2).((a_1 < b_1) + (a_1 = b_1).(a_0 < b_0))) && (3) \end{aligned}$$

Cela donne le circuit suivant :



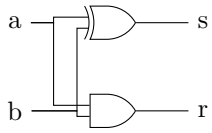
4 Circuits additioneurs

1. Un demi additionneur est composé de :
— deux entrées : les bits a et b
— deux sorties : s la somme des deux bits et r l'éventuelle retenue.
Donner les expressions booléennes de s et r en fonction de a et b.

Solution: $s = a \oplus b$ et $r = ab$.

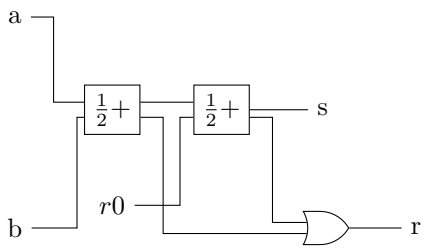
2. Proposer un circuit pour le demi additionneur.

Solution:



3. Un additionneur complet est composé de :
   — trois entrées : les bits a et b et r_0 la retenue propagé par l'additionneur précédent.
   — deux sorties : $s = a + b + r_0$ et r_1 l'éventuelle retenue lors du calcul de s.
   Construire un circuit pour l'additionneur complet. Pour cela utiliser deux demi-additionneur ainsi qu'une porte OU.

Solution:



4. Additionneur 4 bits. Un additionneur 4 bits possède 8 signaux en entrée et 5 signaux en sortie :
   — entrées : $A = a_3a_2a_1a_0$ et $B = b_3b_2b_1b_0$ deux nombres codés sur 4 bits chacun.
   — sorties : $C = c_3c_2c_1c_0$ et r. C est la somme de A et B et r l'éventuelle retenue.
   En utilisant les circuits des questions précédentes proposer un circuit pour l'additionneur 4 bits.

Solution: On met en série : demi-additionneur et trois additioneurs complets. La retenue se propage d'une additionneur au suivant.

Attention le temps de calcul est limité par la propagagtion de la retenue. Pour des mots grande taille (16 bits), il vaut mieux utiliser des additionneurs plus complexes comme le Carry Look Ahead (CLA) addder.