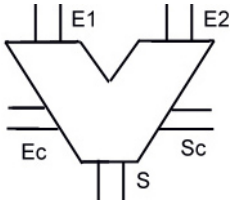
UNITE ARITHMETIQUE ET LOGIQUE
OPERATIONS EN CHAINE

1/ Pose du problème

Nous allons nous poser le problème suivant :
A l'aide d'une Unité Arithmétique et Logique (UAL) 8 bits effectuer l'opération suivante : (A+B+C-D).où A,B,C,D sont des nombres de 8 bits, qui sont introduits par 8 lignes , le résultat de l'opération sera accessible sur ces 8 mêmes lignes.

2/ Matériel utilisé

Une UAL 8 bits dont le symbole est dessiné ci-dessous



E1 et E2 représentent les deux entrées de 8 bits
Ec est l'entrée de configuration sur laquelle on applique le code qui détermine la fonction que doit réaliser l'UAL exemple Additionner E1 + E2 ou soustraire E1 – E2
S est la sortie résultat de l'opération
Sc est une sortie complémentaire sur laquelle ont trouvera les bits suivants :

Z bit qui passera à 1 lorsque l'opération donne un résultat nul (les 8 bits à 0)
C qui passera à 1 lorsque l' opération génère un report (le 9ème bit dans le cas d'une addition)
N qui est une copie du bit de poids fort du résultat
V appelé overflow nous indiquera que, en binaire signé (code complément à 2) le résultat est aberrant quatre cas peuvent se présenter

Addition de 2 nombres positifs donnant un résultat négatif
Ex : 0100 0000 + 0100 0000 = 1000 0000 (64 + 64 = -128) , +128 n'existe pas en code binaire signé 8bits

Soustraction d'un nombre négatif à un nombre positif donnant un résultat négatif
Ex 0100 0000 – 1011 1010 = 1000 0110 (64 – [-70] = – 82) 1011 1010 est le complément à 2 de 70 l'opération se ramène à +64 – (-70) = +134 qui n'existe pas en code binaire signé 8bits

Addition de deux nombres négatifs donnant un résultat positif
Ex 1011 1010 + 1011 1010 = 0111 0100 (-70 + (- 70) = + 116 au lieu de – 140

Soustraction d'un nombre positif à un nombre négatif donnant un résultat positif
Ex 1011 1010 - 0100 0000 = 0111 1010 (-70 – 64 = + 122) au lieu de – 134

On pourra se rappeler des **4 cas aberrants** :

(+) + (+) = (–)
(+) – (–) = (–)
(–) + (–) = (+)
(–) – (+) = (+)

Des **registres à entrées et sortie parallèles** :

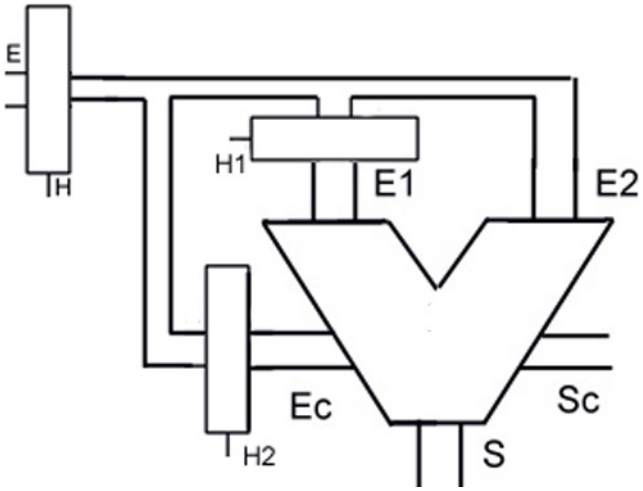
E est un ensemble de 8 entrées
S est un ensemble de 8 sorties
H est l'entrée d'Horloge. Une impulsion sur l'entrée H (passage de 0 à 1 puis de 1 à 0) mémorise dans le registre la donnée présente sur E . Cette donnée est alors présente sur S

Des **buffers tristat** (circuit tampon à sortie 3 états)

E est un ensemble de 8 entrées
S est un ensemble de 8 sorties
OE est l'entrée d'Horloge. Lorsque OE est à 1 les sorties S se comportent comme toutes sorties TTL ou CMOS, placée à 0 les sorties sont en Haute Impédance, c'est à dire le circuit est déconnecté du reste du bus (ensemble des 8 lignes)

3/ Première étape : **Additionner B à A**

Réalisons le montage ci-dessous :



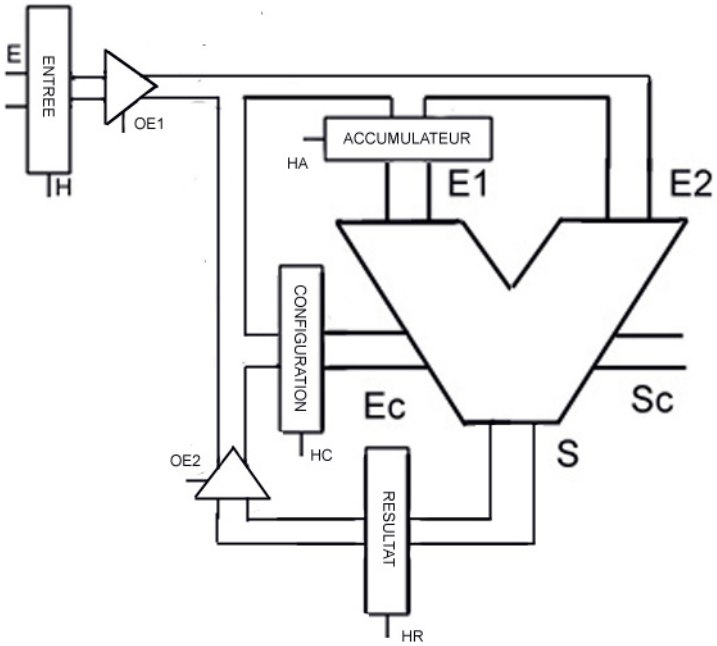
Nous allons maintenant effectuer les actions suivantes :

1/ présenter sur l'entrée E le nombre A
2/ appliquer sur H une impulsion (E2 est alors égale à A)
3/ appliquer sur H1 une impulsion (E1 est alors égale à A)
4/ appliquer sur E le code « addition »
5/ appliquer sur H une impulsion (E2 est alors égale au code addition)
6/ appliquer sur H2 une impulsion (Ec reçoit alors le code addition)
7/ présenter sur E le nombre B
8/ appliquer sur H une impulsion (E2 est alors égale à B)

A l'issue de ces 8 actions, nous avons sur E1 le nombre A, sur E2 le nombre B, sur l'entrée Ec le code de l'addition, la sortie S nous fournit donc le résultat A+B.

4/ Faire **évoluer le schéma de façon à réaliser A+B+C**

Le problème consiste à introduire le résultat de A+B dans le registre placé sur E1, de façon à lui additionner le nombre C. Il est donc nécessaire de faire remonter le résultat A+B vers le registre placé sur l'entrée E1. que nous appellerons Accumulateur (Accu) Il va donc être nécessaire de placer des buffers 3états de façon à éviter les conflits entre le registre d'entrée et la sortie S . Cependant, ce n'est pas aussi simple car si nous mettons la sortie du registre d'entrée en haute impédance , instantanément le nombre B disparaît de l'entrée E2 et A+B disparaît également. Il est donc nécessaire de mémoriser provisoirement le résultat de A+B dans un registre



Décomposons le travail en Tâches sans entrer dans le détail

1/ mettre A dans l'accumulateur
2/ additionner B au contenu de l'Accu résultat dans l'accu
3/ additionner C au contenu de l'Accu résultat dans l'accu

On voit que les tâches 2 et 3 sont identiques seuls les nombres changent . Si le travail à faire est A+B+C–D, pour soustraire D le travail est identique seul le code de la soustraction se substituera à celui de l'addition. Nous noterons (Accu) contenu de l'accu.

Décomposons maintenant chacune des tâches

Mettre A dans l'Accu

- présenter **A** sur E
- pulse sur H
- 1 sur OE1
- pulse sur HA
- 0 sur OE1
-

Addition de B avec (Accu) résultat dans l'accu

- présenter le **code de l'addition** sur E
- pulse sur H
- 1 sur OE1
- pulse sur HC
- 0 sur OE1
- présenter **B** sur E
- pulse sur H
- 1 sur OE1
- pulse sur HR
- 0 sur OE1
- 1sur OE2
- pulse sur HA
- 0 sur OE2

-

Addition de C avec (Accu) résultat dans l'accu

- présenter le **code de l'addition** sur E
- pulse sur H
- 1 sur OE1
- pulse sur HC
- 0 sur OE1
- présenter **C** sur E
- pulse sur H
- 1 sur OE1
- pulse sur HR
- 0 sur OE1
- 1sur OE2
- pulse sur HA
- 0 sur OE2

-

Soustraction de D avec (Accu) résultat dans l'accu

- présenter le **code de la soustraction** sur E
- pulse sur H
- 1 sur OE1
- pulse sur HC
- 0 sur OE1
- présenter **D** sur E
- pulse sur H
- 1 sur OE1
- pulse sur HR
- 0 sur OE1
- 1sur OE2
- pulse sur HA
- 0 sur OE2

Nous venons de faire un pas vers les microprocesseurs, les taches décrites ci dessus en rouge sont des **instructions** (tout au moins un début d'instruction) et la décomposition de celles-ci les **micro instructions**. Les trois dernières micro-instructions sont identiques seul les termes en rouges diffèrent