Leçon A1-1 : L'addition et la multiplication en binaire

Objectifs :

OS 1 - Exécuter en binaire une opération arithmétique de base.

OS 2 - Représenter un nombre entier relatif.

OS 3 - Mettre en œuvre un circuit arithmétique

A-Mise en situation : système gestion de parking Voir manuel de cours Page 6 et 7.

B-Les opérations binaires de bases :

Généralement les opérations de base que l'on est amené à mettre en œuvre dans des systèmes numériques de ce type, sont souvent :

La **somme**, ou …… et l'opération inverse qui est la **différence** ou ……

Le **produit**, ou …… et l'opération inverse qui est le **quotient** ou …….

L'**élévation** à une **puissance n**ième et son opération inverse, l'extraction de la ……

Le contenu proposé, traitera uniquement des quatre premières opérations, à savoir l'**addition**, la **soustraction**, la **multiplication** et la **division**.

Pour distinguer ces opérations, on utilise comme convenu les symboles classiques relatifs à l'algèbre classique, à savoir (le **+** pour *l'addition*, le **-** pour la *soustraction*, le **x** pour la *multiplication* et le **/** pour la *division*).

1-L'addition binaire :

1-1Principe :

Quand vous faites une addition en décimal, vous faites la somme des chiffres se trouvant dans une même colonne. Si la somme est inférieure à 10, alors vous posez le résultat obtenu et passez à la colonne suivante. Si la somme est supérieure à 10, alors vous posez le chiffre des unités et gardez en retenue le chiffre des dizaines. Si vous faites la somme de 2 nombres, alors la retenue ne pourra être supérieure à 1.

Le principe est exactement le même en binaire. Vous faites la somme, posez le chiffre des unités, et retenez le chiffre de la seconde colonne en retenue (qu'il vaut mieux, évidemment, éviter d'appeler les« dizaines »).Si vous faites la somme de 2 nombres, alors il n'y a que 4 cas possibles :

- 0 + 0 = 0, on pose 0
- 0 + 1 = 1, on pose 1
- 1 + 1 = 10, on pose 0 et on retient 1
- 1 + 1 + une retenue de 1 = 11, on pose 1 et on retient 1

Exemple : Soit additionner en décimal les deux nombres suivants : 19 et 71.

	Décimal		Binaire						
Opération	Chiffre	Retenue:							
Addition	71	Addition	1	0	0	0	1	1	1
	19		0	0	1	0	0	1	1
Total:		Total:							

1-2 Réalisation industrielle :

a-Demi additionneur à un bit :

Pour une addition de deux nombres A ($A = a_0$) et B ($B = b_0$) de 1 bit, quatre combinaisons sont possible, et le résultat occupera 2 bits, un bit pour la somme (S) et un autre pour la retenue (r). Ce dispositif est également appelé demi additionneur ou en anglais « Half-adder »

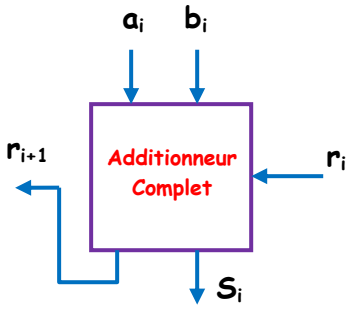
b-Activité 1 Page 6-7 et 8 :

c-Additionneur complet de 2 nombres à un bit :

Le demi additionneur ne permet pas en réalité de réaliser totalement une addition, car dans le cas d'une addition de deux nombres à plusieurs bits, on s'aperçoit qu'on a additionné 2 nombres

de 3 bits et non pas de 2 bits, le troisième provenant d'une retenue. On doit donc, construire un additionneur complet dont la structure est la suivante.

Symbole :



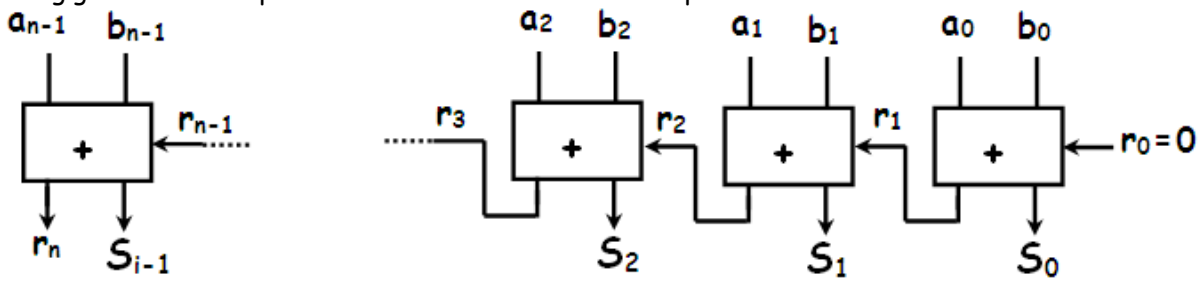
d-Activité 2 Page 9-10 et 11 :

1-3 Généralisation à n bits :

Le principe étant le même, en généralisant, nous pouvons construire un additionneur de deux mots à n bits.

Exemple : Soit A = a_{n-1} ……$a_3a_2a_1a_0$ B = b_{n-1} ……$b_3b_2b_1b_0$

Le logigramme correspondant à base d'additionneur complet de deux nombres est le suivant :



a-Activité 3 Page 12-13 :

1-4 Représentation d'un nombre binaire signé : Dans le système décrit précédant, nous n'avons tenu compte que des nombres positifs. La plupart des dispositifs numériques traitent également les nombres négatifs, ce qui impose de prendre en compte deux symboles supplémentaires + et -. Pour cela un nombre binaire signé est composé de :

-Un ensemble de bits qui représente la valeur absolue du nombre.

-Un bit de signe indiquant si le nombre positif ou négatif. Ce bit de signe est le premier à gauche de l'ensemble de bits représentant le nombre binaire. Lorsqu'il est égal à 0, le nombre est positif lorsqu'il est égal à 1, le nombre est négatif.

Exemples :

(+ 47) = **0** 101111 (- 47) = **1** 101111

Bit de signe — Valeur absolue — Bit de signe — Valeur absolue

Cette représentation est simple, mais non convenable pour effectuer des calculs arithmétiques, car elle présente l'inconvénient d'avoir deux représentations possibles du zéro (+0 et -0).
De plus les calculateurs n'y ont pas recours, en raison de la complexité des circuits qui matérialisent cette notation. **Pour ces raisons, il est préférable d'utiliser pour écrire les nombres binaires signés la représentation en complément à 2.**

Représentation en complément à 2 :

Méthode pratique :

Une méthode pratique peut donner le complément à 2 d'un nombre binaire N en changeant chaque 0 par 1 et chaque 1 par 0 (Cette étape est appelée complément à 1 de N, noté $\overline{N}$) et en ajoutant 1 au bit de poids le plus faible. On a alors $-N = \overline{N} + 1$.

La représentation en complément à 2 est très utile dans les opérations arithmétiques car avec, on peut faire une soustraction en effectuant en réalités une addition. Cela est très important dans le cas des ordinateurs et les calculateurs, puisqu'avec les mêmes circuits, on effectue l'addition et la soustraction.

1-5 Addition binaire :

Les opérations seront traitées sur 4 bits (un quarté).

1 er cas : Addition de deux nombres positifs :

Soit à additionner (**+9**) et (**+4**). L'addition dans ce cas est immédiate

$(+9)_{(10)}$ ➔ **0** 1001

$(+4)_{(10)}$ ➔ **0** 0100

$(13)_{(10)}$ ➔ **0** 1101

2ème cas : Addition d'un nombre positif et d'un nombre négatif plus petit :

Soit à additionner (**+9**) et (**-4**). Dans ce cas (**+9**) sera remplacé par son équivalent binaire exact; par contre (-4) doit être remplacé par son complément à 2, d'où :

$(+9)_{(10)}$ ➔ **0** 1001

$(+4)_{(10)}$ ➔ **0** 0100

Complément à 2 de (+4) est **1** 1100

3ème cas : Addition d'un nombre positif et d'un nombre négatif plus grand :

Soit à additionner (**-9**) et (**+4**). Dans ce cas (-9) doit être exprimé par son complément à 2, par contre (+4) est replacé par son équivalent binaire exact, d'où :

$(+4)_{(10)}$ ➔ **0** 0100

$(+9)_{(10)}$ ➔ **0** 1001

Complément à 1 de (+9) = **1** 0110 et donc le complément à 2 = **1** 0111. Puisque le bit de signe est 1, le résultat est supposé négatif, dans ce cas la norme du résultat représente le complément à 2 de la valeur recherchée; donc pour obtenir la valeur exacte de l'opération, il faut complémenter à 2 cette norme; d'où : le complément à 1 de **1011** = 0100 ➔ complément à 2 = 0101, c'est bien $5_{(10)}$ et puisque le bit de signe est 1 le résultat réel est (-5).

4ème cas : Addition de deux nombres négatifs :

Soit à additionner (**-9**) et (**-4**).Dans ce cas les deux nombres doivent être exprimés dans la notation en complément à 2.

(-9) ➔ **1** 0111 et (-4) ➔ **1** 1100

Puisque le bit de signe est 1, le résultat est négatif; la norme du résultat trouvé n'est autre que le complément à 2 de la valeur exacte, pour l'obtenir il faut ré-complémenter à 2 la norme du résultat trouvé; d'où 0011 ➔ Le complément à 2 = 1101 c'est l'équivalent binaire de $13_{(10)}$.

5ème cas : Addition de deux nombres égaux et opposés :

Dans ce cas le résultat est évidement …… .

1-6 Avantage et inconvénient :

Ce type de structure présente l'avantage d'être simple à mettre en œuvre, facile à comprendre, en contre partie, il présente l'inconvénient majeur d'avoir un temps de réponse relativement long à cause de la propagation de la retenue d'un bloc à l'autre, qui suivant les cas, peut aboutir à des résultats erronés, d'où l'appellation de ce type d'additionneur : d'**additionneur** à **propagation** de **retenues** ou **additionneur itératif**. Pour faire face à cet inconvénient et pour effectuer la somme plus rapidement, il faut éliminer ou d'au moins limiter ce temps de propagation.

Pour cela on se base sur l'hypothèse que les termes de la somme sont connus et disponibles avant même que commence l'opération d'addition. On peut alors calculer, **en anticipant**, la retenue pour chaque étage indépendamment des étages précédents. Il s'agit de pouvoir disposer de toutes les retenues simultanément et dans un temps le plus court possible. Autrement dit, il faut calculer la retenue r_1 à partir des bits a_0, b_0 et r_0, la retenue r_2 à partir des bits a_0, b_0, r_0, a_1 et b_1 et ainsi de suite. Ce type d'additionneur est appelé : **additionneur à anticipation de retenues** ou à **report anticipé**.

1-7 Additionneur à anticipation de retenues :

a-Schéma synoptique :

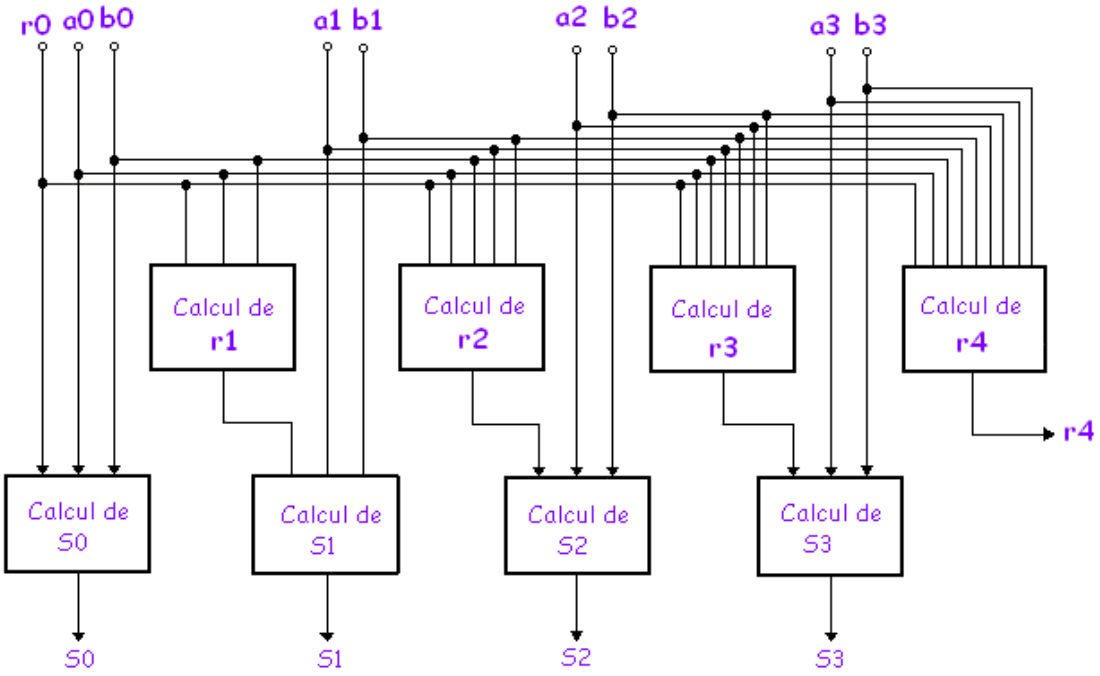


fig:Synoptique d'un additionneur 4 bits à retenue anticipée

b-Principe :

Le synoptique suivant, explique le principe de fonctionnement de ce circuit :

Pour effectuer le calcul des retenues de façon anticipée, il faut transformer l'équation de la retenue r_{i+1} vue précédemment : $r_{i+1}=a_i\overline{b_i}r_i+a_ib_i+\overline{a_i}b_ir_i$ cette transformation s'effectuera

en se basant sur la constatation suivante : r_{i+1} = ... si a_i=.....,b_i=..... et r_i=..... Donc on peut ajouter $a_ib_ir_i$ à l'expression de r_{i+1} autant de fois que l'on veut dans notre exemple on va ajouter seulement 2 fois.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

c-Activité 4 Page 14-15-16 et 7 :

d-Exemple :

Etudions un cas particulier en supposant **i = 4** et bien entendu (**r_0 = 0**) ➔ additionneur à **4 bits**.

Calculons les différentes retenues:

r_1 = ……

r_2 = ……

r_3 = ……

r_4 = ……

Tracer alors le circuit d'anticipation des retenues.

……

……

……

……

……

……

……

……

……

……

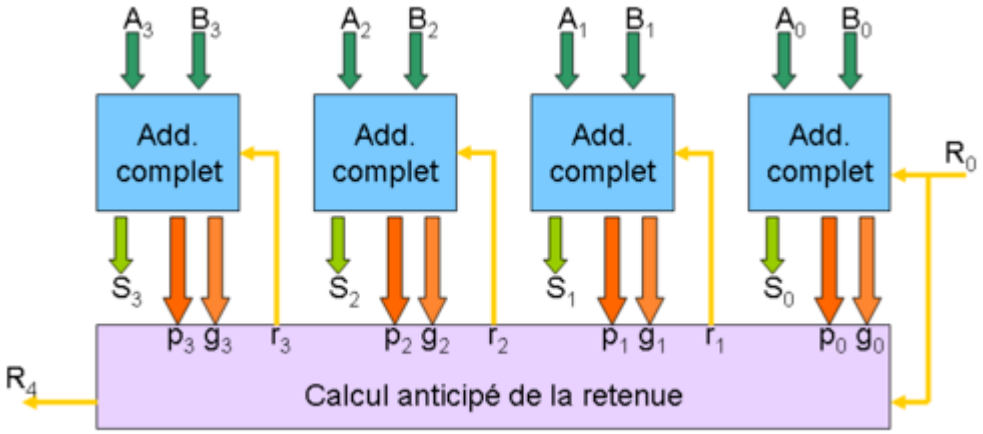
……

……

……

……

Finalement on obtient le schéma complet de l'additionneur **4 bits** avec son circuit d'anticipation des retenues :



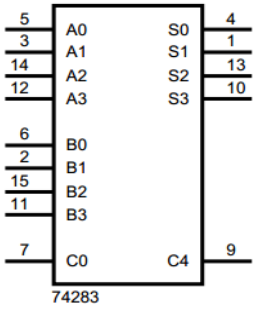
e- L'additionneur et ses périphériques :

Les additionneurs nécessitent des périphériques pour leur fonctionnement tel que des composants de stockage ; de transfert de calcul.

f-Activité 5 Page18-19 et 20 :

le 74283 est un additionneur de 2 nombres à 4 bits A et B:

- $A(A_3A_2A_1A_0)$ et $B(B_3B_2B_1B_0)$ + une entrée de retenue extérieure C_0.
- Le résultat sera affiché par les sorties $C_4S_3S_2S_1S_0$



1-8 Additionneur DCB " Décimal Codé Binaire " noté en anglais " BCD " :

a-Principe :

Une notation fréquemment utilisée dans les ordinateurs est la notation **BCD**.
Dans cette notation, chaque chiffre du nombre décimal est représenté en binaire, soit sur **4 bits** (**BCD compacté**), soit sur **8 bits**. Par exemple, en **BCD** compacté, $237_{(10)}$ s'écrit donc : **0010 0011 0111**. Voici comment on peut effectuer l'addition en **BCD**.
Si l'on additionne en arithmétique binaire les codes **BCD** correspondant à des chiffres dont la somme ne dépasse pas **9**, on aura évidemment un résultat donnant la représentation correcte de la somme en **BCD**. Par exemple, **64 + 21** donne bien **85**. Cherchons en premier l'équivalent en **BCD** des deux nombres soit :

……
……
……
……

Cependant, dès que la somme dépasse **9** dans un quartet de **4 bits**, il faut apporter une correction. Donc pour remédier à ce problème on procède comme suit :
Chaque fois que le résultat d'un quartet est **supérieur** à **9** on ajoute **6(10)** c'est-à-dire **0110(2)** au résultat pour retrouver la transcription en **DCB** de ce résultat.
Exemple : soit à additionner $84_{(10)} + 37_{(10)} = 121_{(10)}$

……
……
……
……

b-Activité 6 Page21-22 :

2-La multiplication ou le produit binaire :

a-Principe : Comme pour le système à **base 10**, la multiplication par **0** entraine un résultat égal à 0. La multiplication par **1** recopie le multiplicande. La procédure d'obtention du résultat est identique à celle de la multiplication en décimal.
Exemple :

$$\begin{array}{rr} & 25 \\ \times & 6 \\ \hline \end{array} \qquad \begin{array}{r} 11001 \\ 110 \\ \hline \end{array}$$

b-Réalisation industrielle :

Exemple : multiplicateur binaire de deux nombres à **2 bits**.
Soit à réaliser la multiplication de deux nombres binaires à 2 bits A (a_1a_0) et B (b_1b_0).

$$\begin{array}{cccc} & & a_1 & a_0 \\ & & b_1 & b_0 \\ \hline r_2 & r_1 & a_1b_0 & a_0b_0 \\ & a_1b_1 & a_0b_1 & \\ \hline C_3 & C_2 & C_1 & C_0 \end{array}$$

Interprétation :

Composants nécessaires :

- ……
- ……
- ……
- ……

Logigramme :

……
……
……
……
……
……
……
……
……
……
……

c-Activité 7 Page 23 et 24 :

d-Activité 8 Page 25 et 26 :