

Information numérique – Projet n°1

Question 3.3.1

La retenue entrante de poids 0 est celle correspondante au bit de poids faible de la ligne « Retenues ».

Lorsqu'on additionne deux nombres de 8 bits, on obtient un nombre de 9 bits. La retenue se propage du bit de poids le plus faible vers le bit de poids le plus fort.

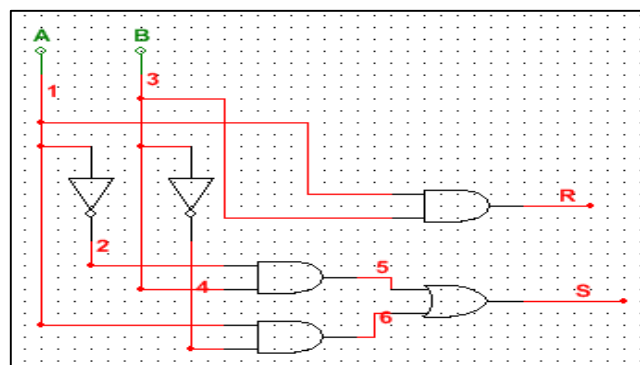
Question 4.1.1

La table de vérité d'un demi-additionneur est la suivante :

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

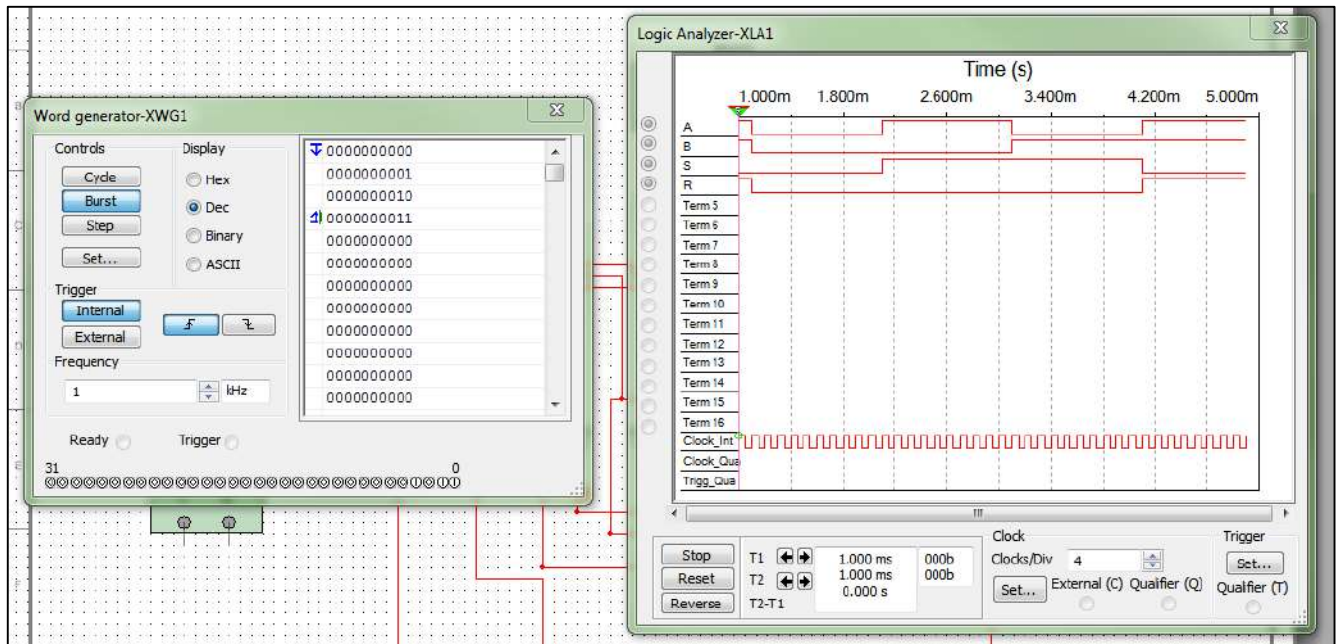
- L'équation logique de S est $A \text{ XOR } B$
- L'équation logique de R est $A.B$

Leur schéma correspondant ci-dessous :



Le schéma de S est en fait l'équivalent d'un XOR. Soit $S = A \text{ XOR } B$.

Question 4.1.2)



Les résultats correspondent à la table de vérité dressée en **4.1.1**.

Question 4.2.1)

La table de vérité d'un additionneur complet est la suivante :

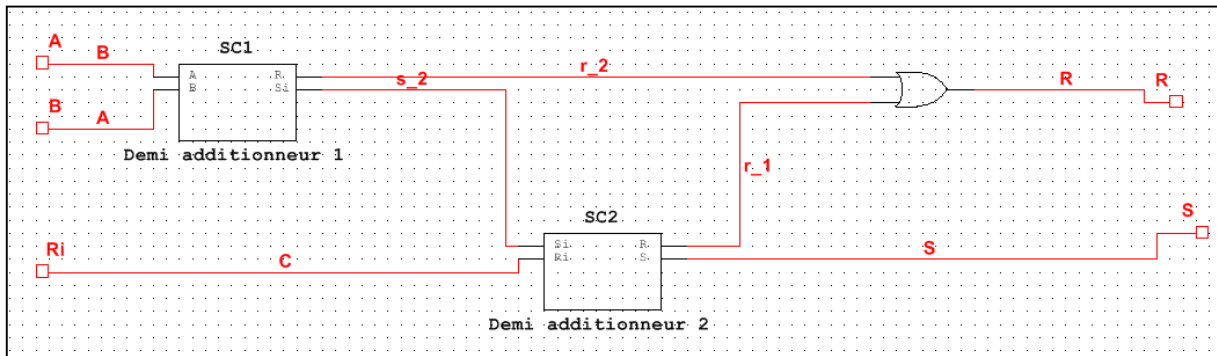
ri	ai	bi	Si	Ri
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- L'équation logique de Si est $ri/.ai/.bi + ri/.ai.bi/ + ri.ai/.bi/ + ri.ai.bi$
- L'équation logique de Ri est $ri.bi + ri.ai + ai.bi$

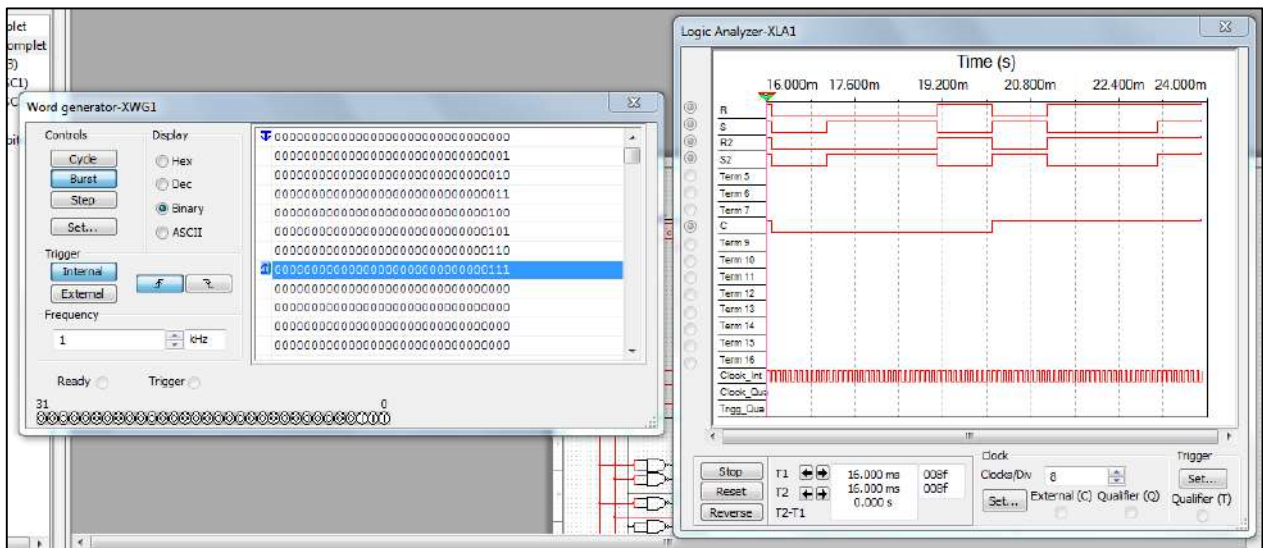
L'additionneur complet est bien formé de deux demi-additionneurs car il effectue dans un premier lieu la somme de ai et bi, puis la somme du résultat avec ri. Il est donc formé par des portes OR.

Question 4.2.2, question 4.2.3)

Ci-dessous l'intérieur du sous circuit d'un Additionneur Complet :

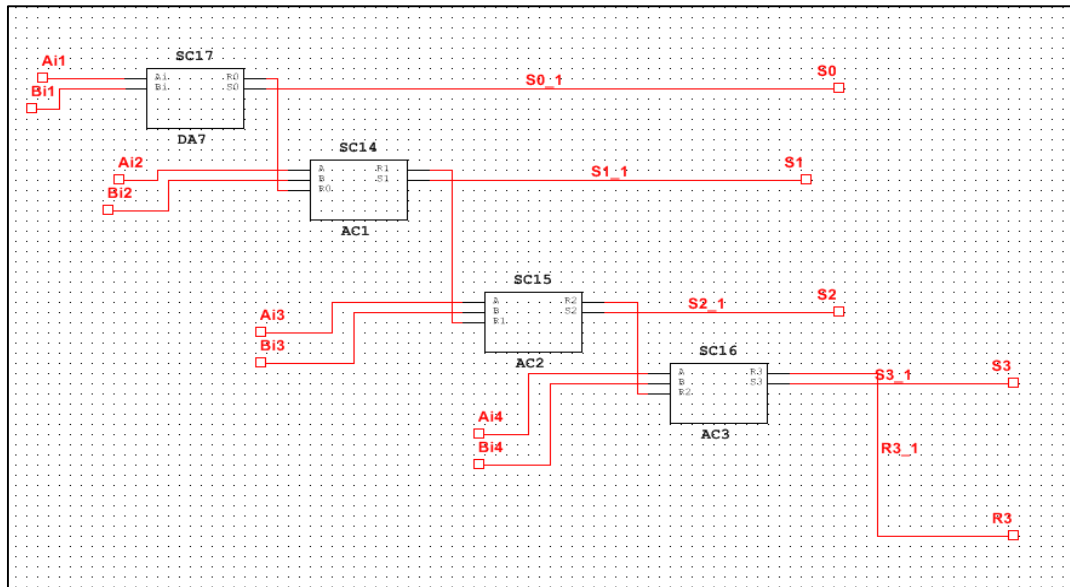


Il est bien constitué de deux Demi Additionneurs et d'une porte OR, les logigrammes suivant prouvent le bon fonctionnement du circuit :

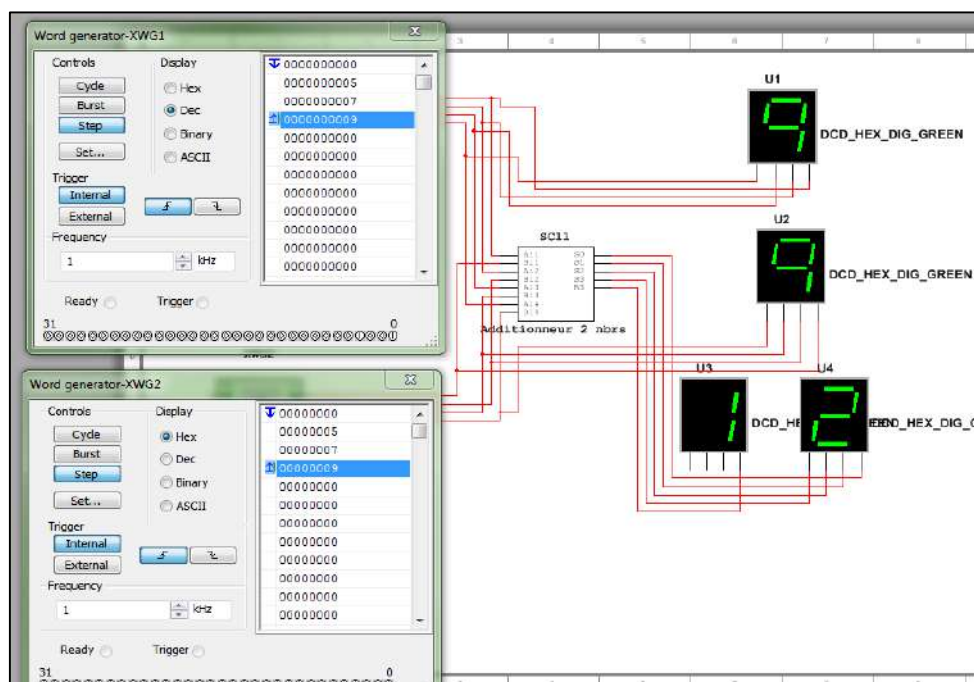


Question 4.3.1)

Voici la constitution du sous circuit de l'additionneur de deux nombres de 4 bits à propagation de retenue, on constate un Demi-additionneur ainsi que 3 additionneurs complets :



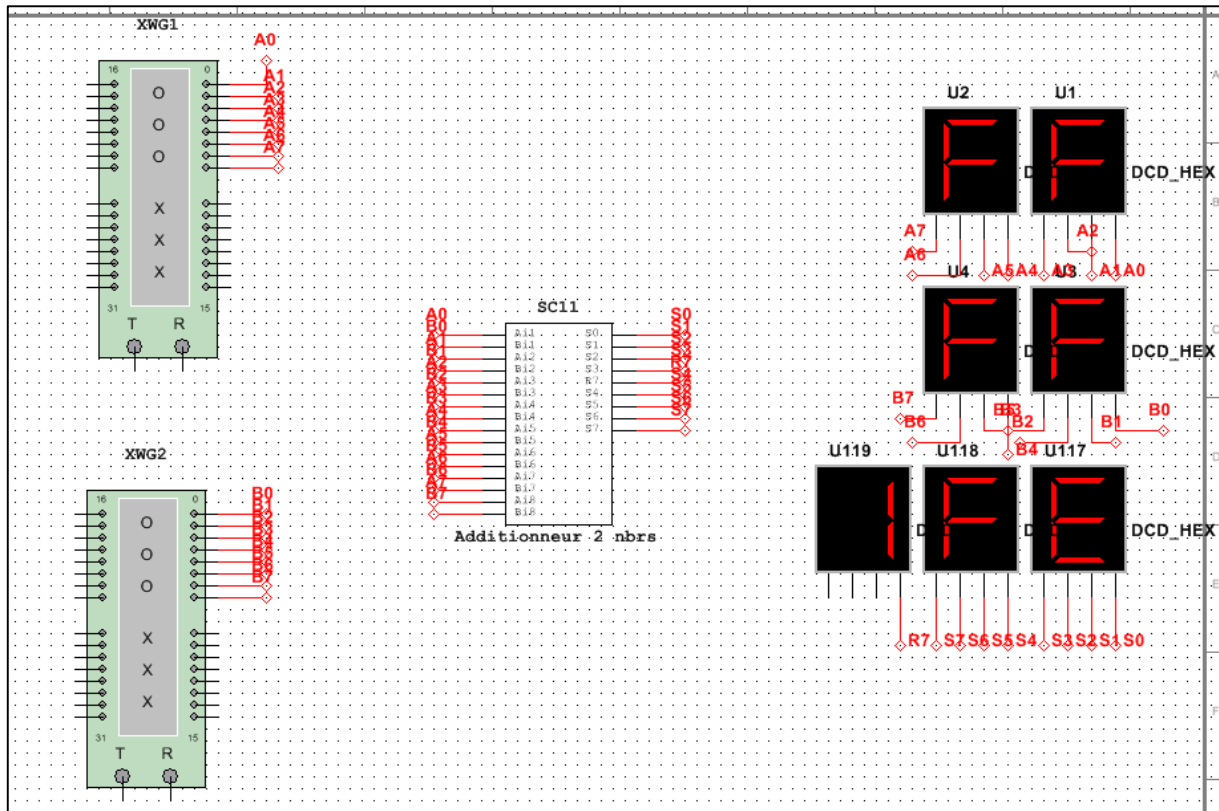
Simulation :



Question 4.4.1)

L'additionneur de deux nombre de 8 bits à propagation de retenues fonctionne sur la même base que le précédent, on rajoute simplement 4 AC dans le sous circuit (puisque'il y a 4 bits de plus à traiter)

On effectue le montage, ainsi que la simulation :



Ici les afficheurs indiquent la somme de 1111 (FF) et 1111 (FF), elle vaut bien **111111110 (1FE)**.

Question 5.1.1)

- Si on factorise l'équation logique de R_i trouvée précédemment, on obtient :
 $R_i = a_i.b_i + r_i.(b_i + a_i)$.
Comme $G_i = a_i.b_i$ et $P_i = a_i + b_i$, on peut écrire **$R_i = G_i + P_i.r_i$**
- En reprenant la question précédente, on peut dire :
$$S_i = b_i.(r_i/.a_i/ + r_i.a_i) + b_i/.(r_i/.a_i + r_i.a_i/) = b_i.(r_i \text{ XOR } a_i)/ + b_i/.(r_i \text{ XOR } a_i)$$

 $= b_i \text{ XOR } r_i \text{ XOR } a_i$

Question 6.1.1)

$$R0 = G0 + P0.0 = G0$$

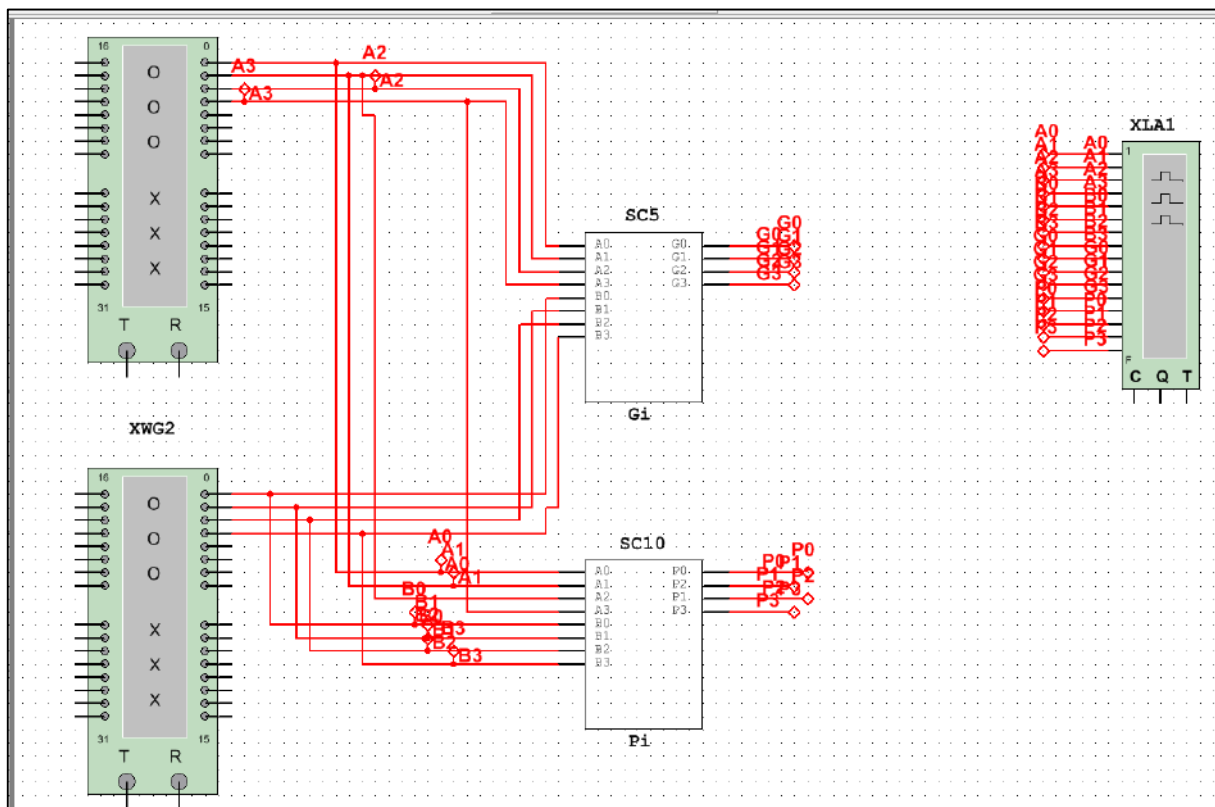
$$R1 = G1 + P1.G0$$

$$R2 = G2 + P2 (G1 + P1.G0)$$

$$R3 = G3 + P3 (G2 + P2 (G1 + P1.G0))$$

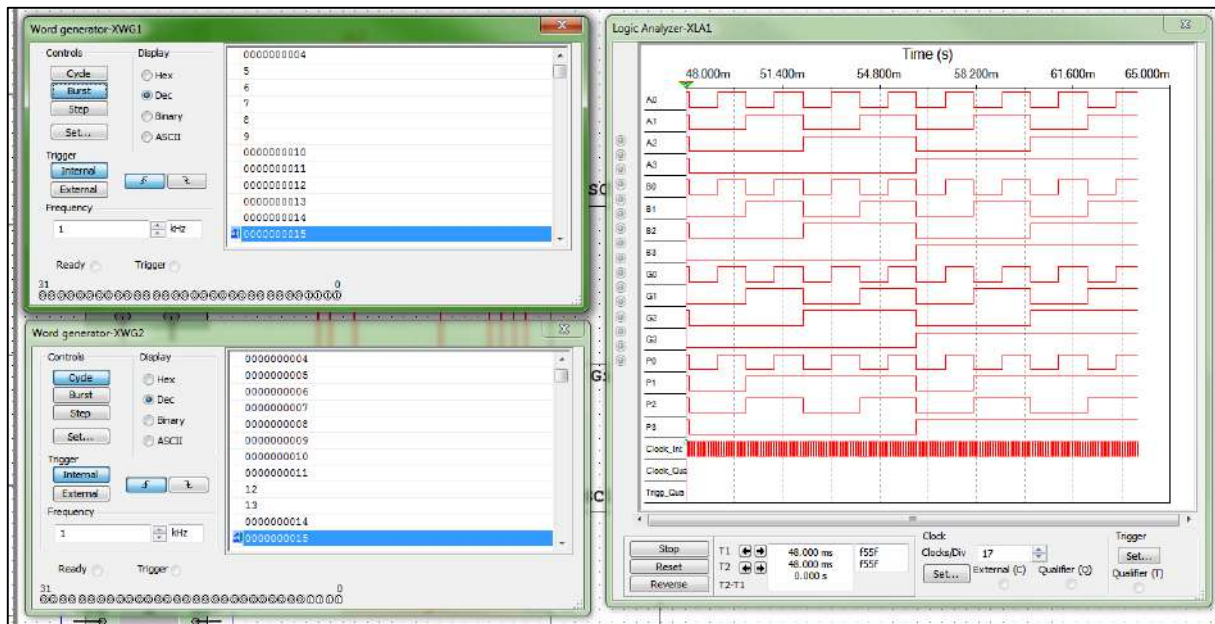
Question 6.1.2)

Montage :



(Simulation page suivante)

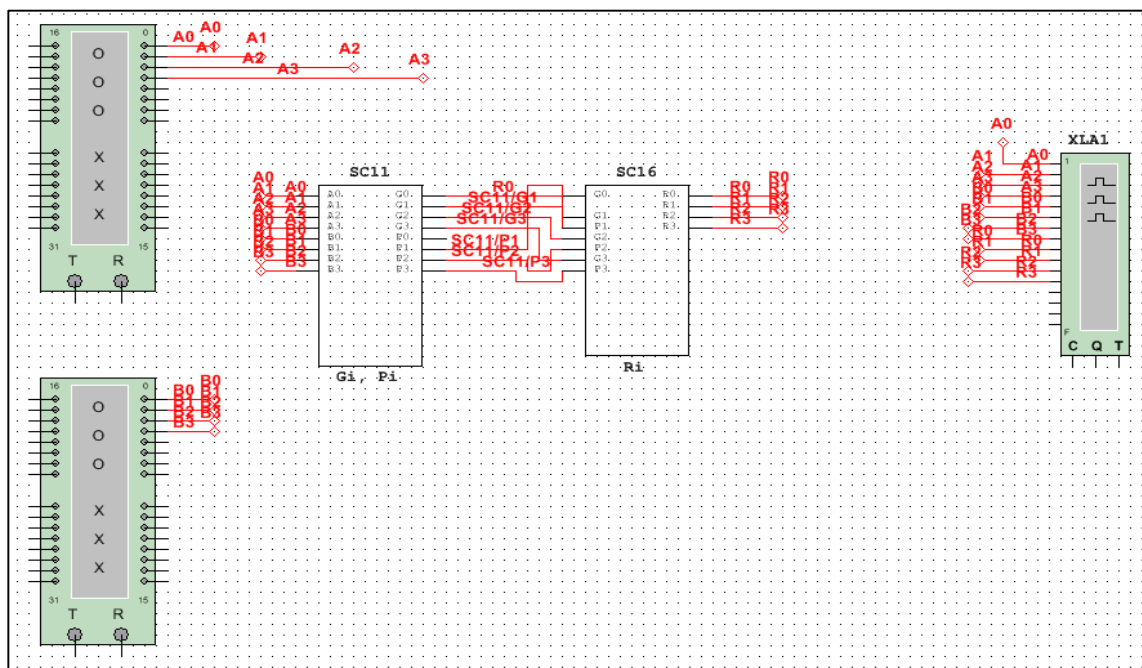
Simulation :



On constate que G_i vaut bien $a_i.b_i$ et P_i vaut bien $a_i + b_i$

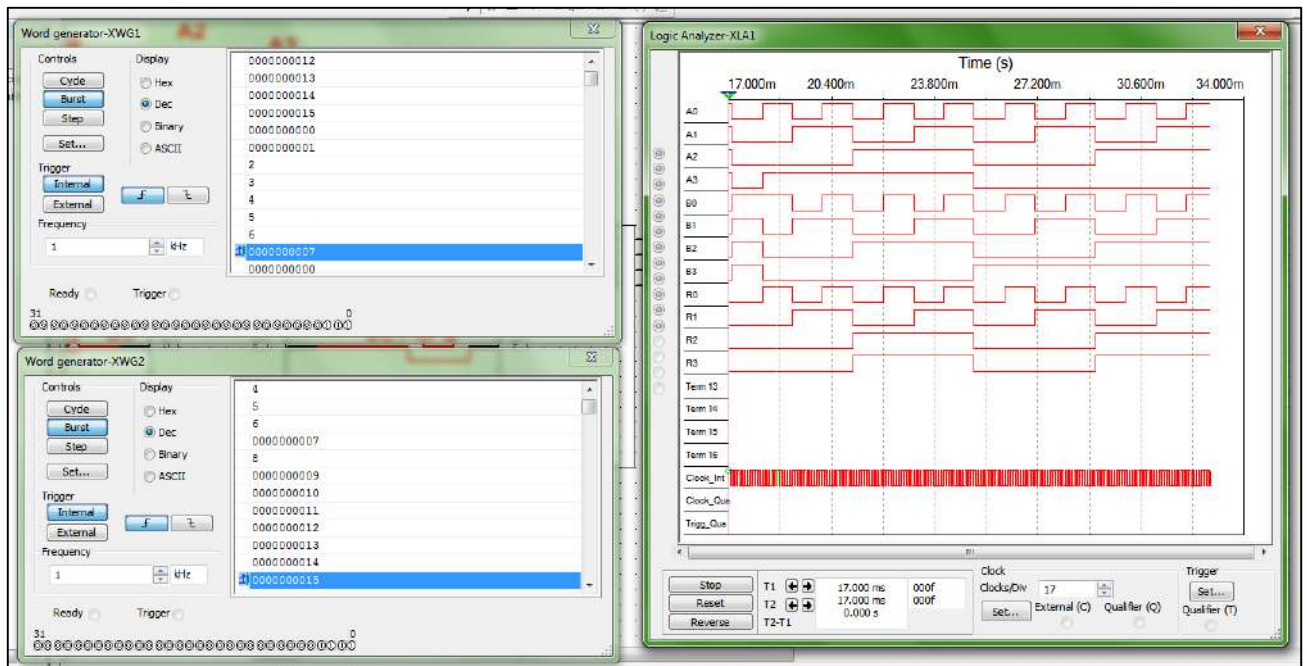
Question 6.1.3)

Montage :



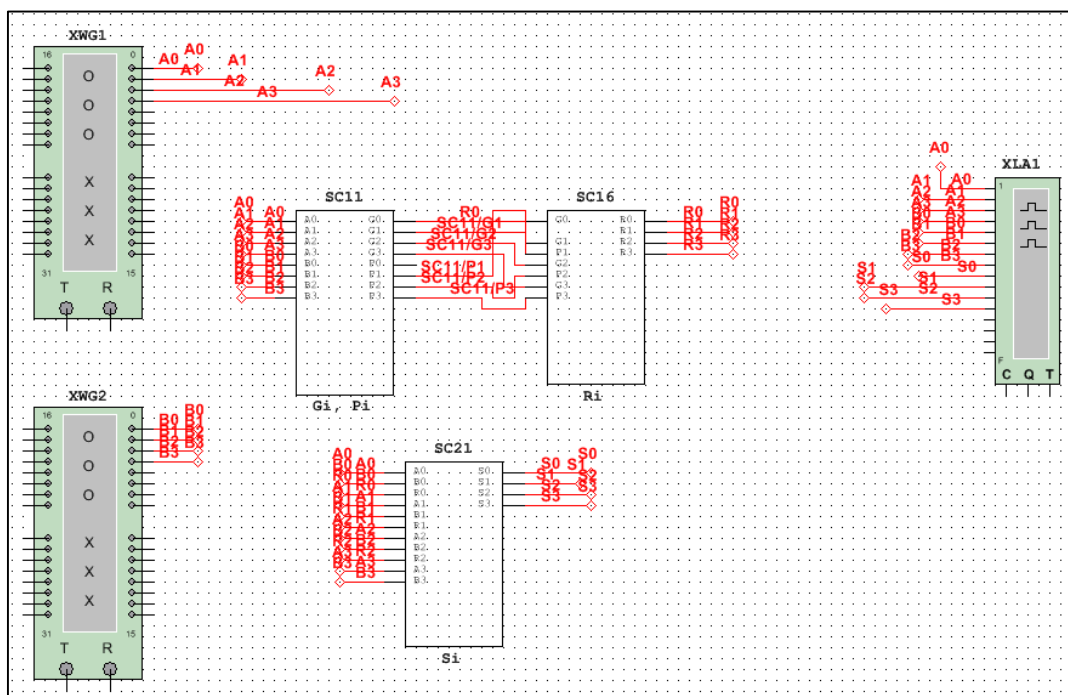
Les R_i sont calculés en fonction des G_i et des P_i .

Simulation :

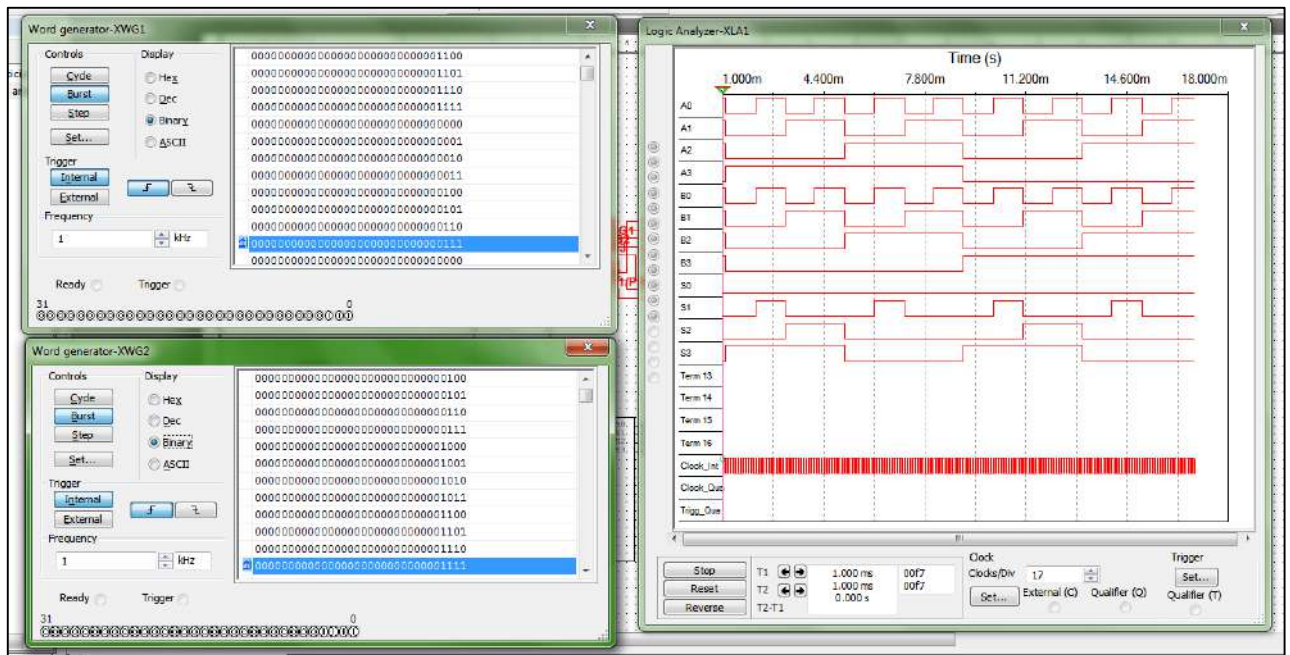


Les Ri valent bien $G_i.P_i + R(i-1)$

Montage :



Simulation :

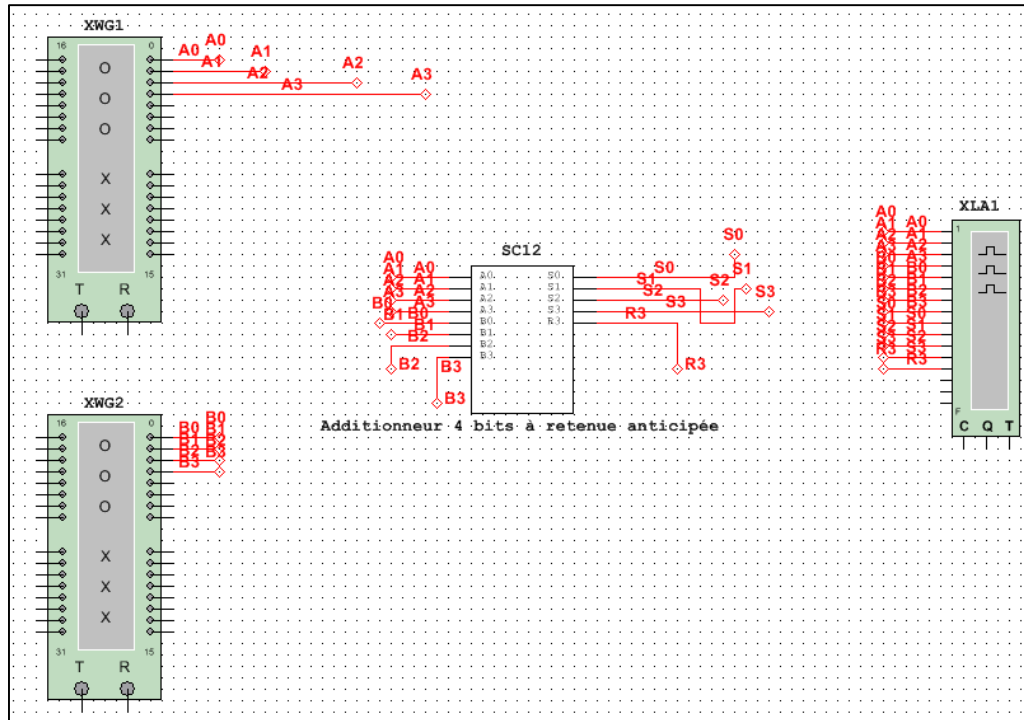


Les Si sont calculés en fonction des Ai, Bi et Ri.

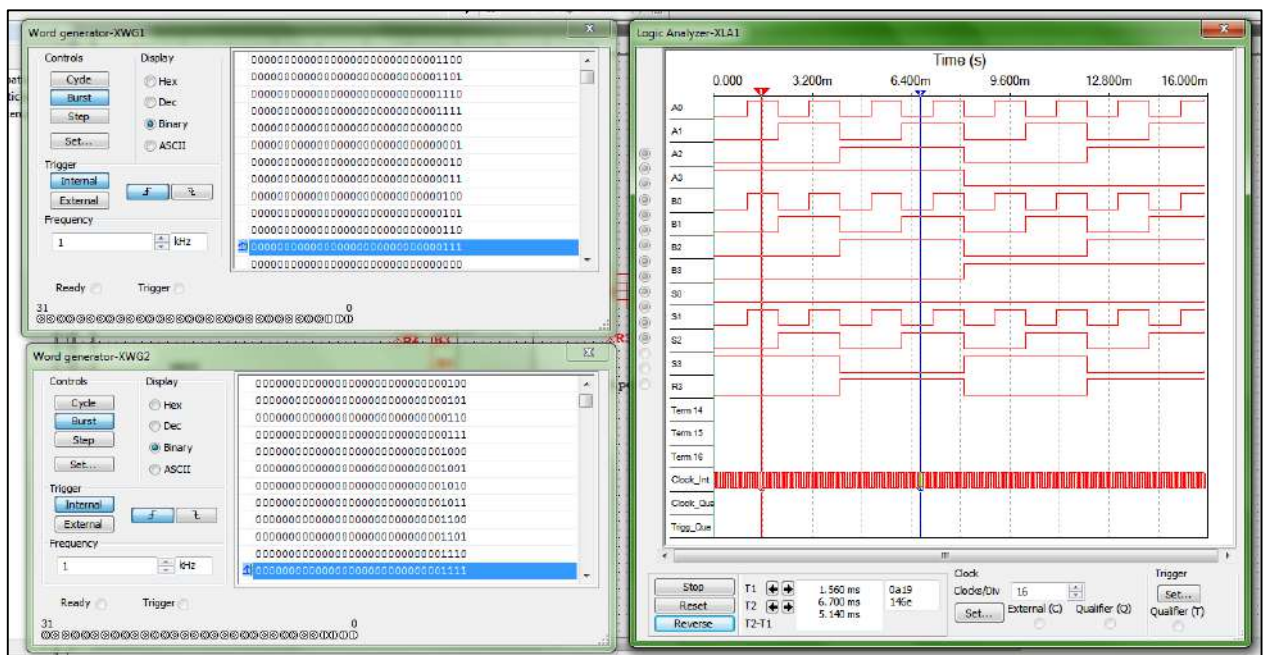
Question 6.1.4)

Montage :

On rassemble le tout dans un sous circuit :



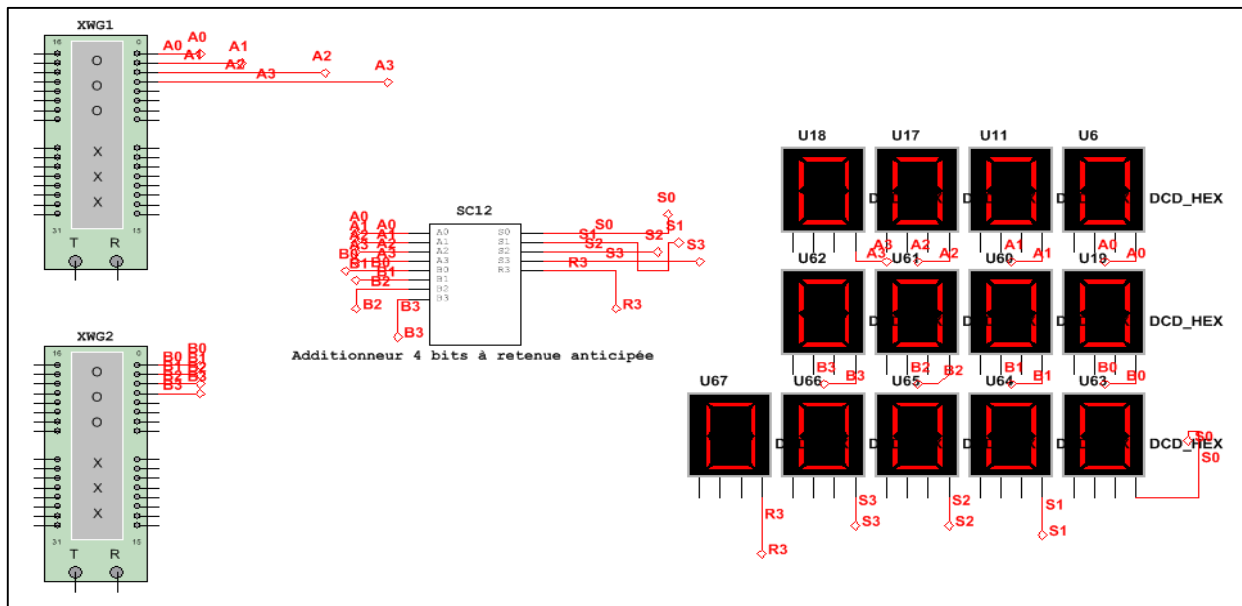
Simulation :



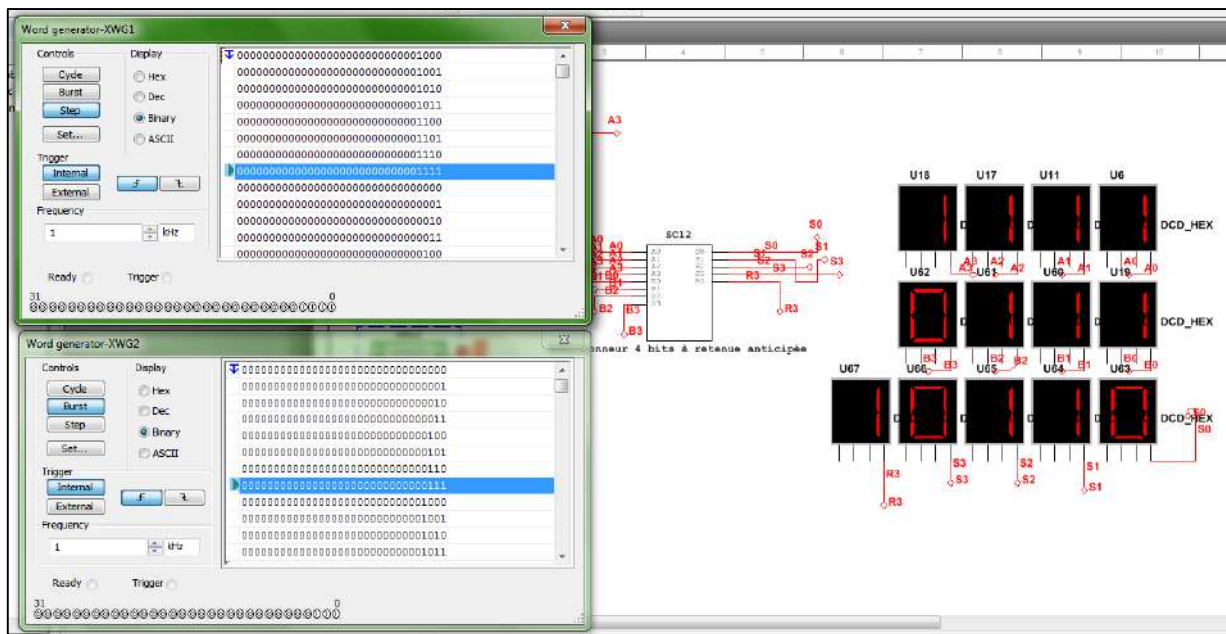
Le repère bleu montre l'addition de 1110 et 0110, qui vaut bien 10100.

On peut afficher la pose de la somme grâce à des afficheurs 7 segments. La ligne du haut affiche le nombre binaire A, celle du milieu affiche B, et la ligne du bas affiche la somme.

Montage :



Simulation :



Ici, l'additionneur indique la somme de 1111 avec 0111.

On constate que cela vaut bien 10110 puisque $15 (1111) + 7 (0111) = 22 (10110)$.

Question 7.1.1)

- Un Demi-Additionneur est composé de 7 portes ET, il travaille donc en $7 \times 10\text{ns} = 70\text{ns}$.
- Un Additionneur Complet est constitué de deux Demi-Additionneurs et d'une porte OU. Soit $2 \times 70\text{ns} + 10\text{ns} = 150\text{ ns}$.

Question 7.1.2)

Un Additionneur de 4 bits à propagation est constitué de 3 Additionneur Complets et d'un Demi-Additionneur. Soit $3 \times 150\text{ns} + 70\text{ns} = 520\text{ ns}$.

Question 7.1.3)

- Le calcul d'un G se fait grâce à une porte ET, soit 10 ns.
- Le calcul d'un P se fait grâce à une porte OU, soit 10 ns.
- Le calcul d'un R se fait grâce à une porte ET et une porte OU, soit $2 \times 10\text{ ns} = 20\text{ ns}$.
- Le calcul d'un S sans retenue entrante se fait grâce à 2 portes ET et une porte OU, soit $2 \times 10\text{ns} + 10\text{ns} = 30\text{ ns}$.
- Le calcul d'un S avec une retenue entrante, en considérant que cette retenue est déjà calculée, se fait grâce à 8 portes ET et 3 portes OU, soit $8 \times 10\text{ns} + 3 \times 10\text{ns} = 110\text{ ns}$.

Question 7.1.4)

Un Additionneur à 4 bits à anticipation est constitué de 4 calculs de G, 4 calculs de P, 3 calculs de R, un calcul de S sans retenue entrante, et 3 calculs de S avec retenue entrante. Soit $4 \times 10\text{ns} + 4 \times 10\text{ns} + 3 \times 20\text{ns} + 30\text{ns} + 3 \times 110\text{ns} = 500\text{ ns}$.

Question 7.1.5)

Un additionneur de 8 bits est constitué de deux fois plus de G et P, et de 7 R et S avec retenue entrante (avec toujours 1 S sans retenue entrante). Soit $8 \times 10\text{ns} + 8 \times 10\text{ns} + 7 \times 20\text{ns} + 30\text{ns} + 7 \times 110\text{ns} = 1100\text{ ns}$.

Question 7.1.6)

Les temps de calculs seront forcément plus faibles si on disposait de portes OU et ET à 3 ou 4 entrées... Avec des portes à trois entrées, les temps de calcul peuvent être réduits de presque un tiers, et de moitié avec des portes à 4 entrées.

Question 7.1.7)

Additionneur 4 bits à propagation		Additionneur 4 bits à anticipation	
Avantages	Inconvénients	Avantages	Inconvénients
Calcul de seulement 2 éléments, R et S.	La dernière retenue n'est connue qu'après le calcul du dernier S.	Calcul de la dernière retenue possible sans connaître le dernier S.	Calcul de 4 éléments: G, P, R et S
	Le temps de calcul des S est de 520 ns.	Le temps de calcul des S est de seulement 500 ns.	

Question 8.1.1)

Le circuit 74LS83N demande 8 bits en entrée (2 mots de 4 bits + la retenue de base, normalement égale à 0), et 4 bits + 1 en sortie (somme des 2 mots + dernière retenue). L'additionneur calcule la dernière retenue avant la dernière somme, il fonctionne donc par retenue anticipée.

Puis il effectue l'addition des deux bits de poids les plus faibles, en ajoutant la première retenue, puis celle des deux bits suivants, en ajoutant la retenue trouvée précédemment, et ce jusqu'à avoir calculé la somme de chacun des 4 bits des 2 mots.

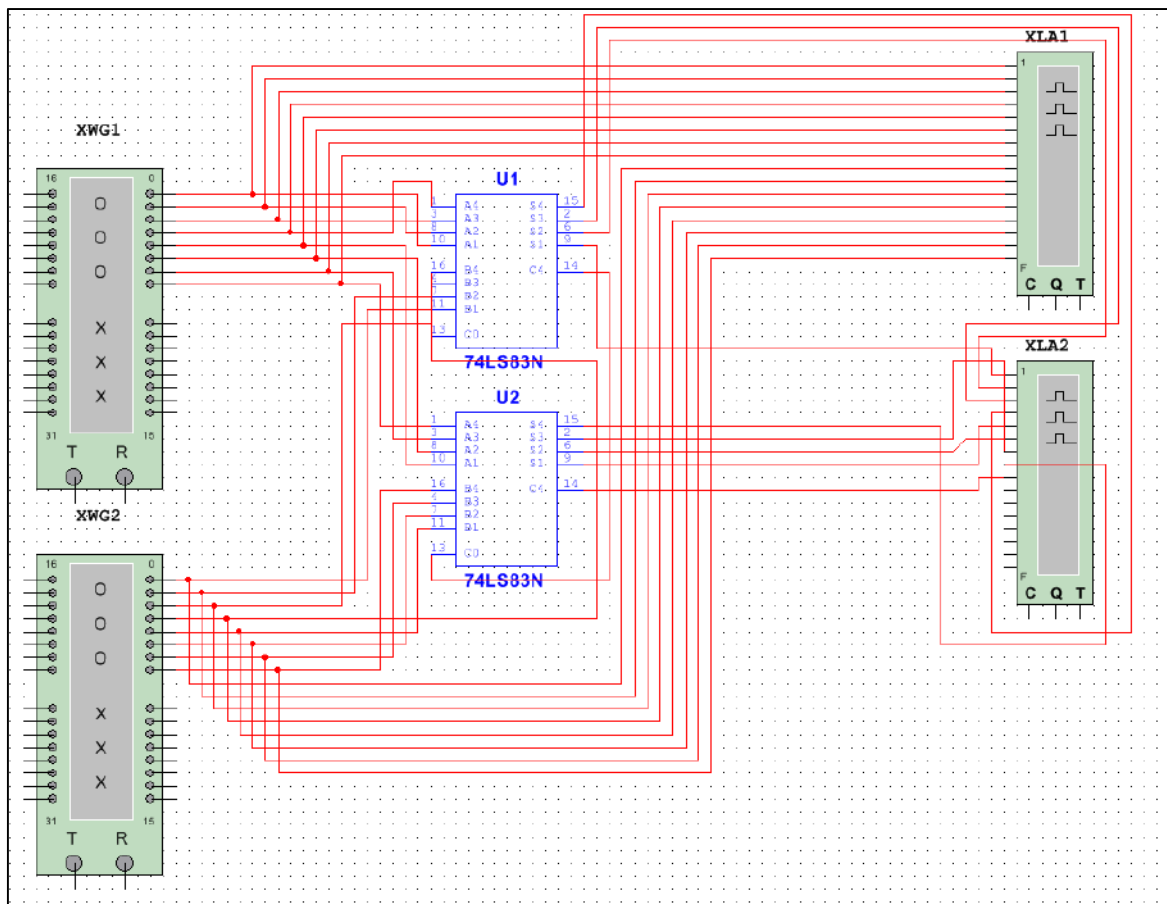
Entrées			Sorties	
C0	An	Bn	Σn	Cn
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

N varie de 1 à 4 et correspond au repère des bits à additionner.

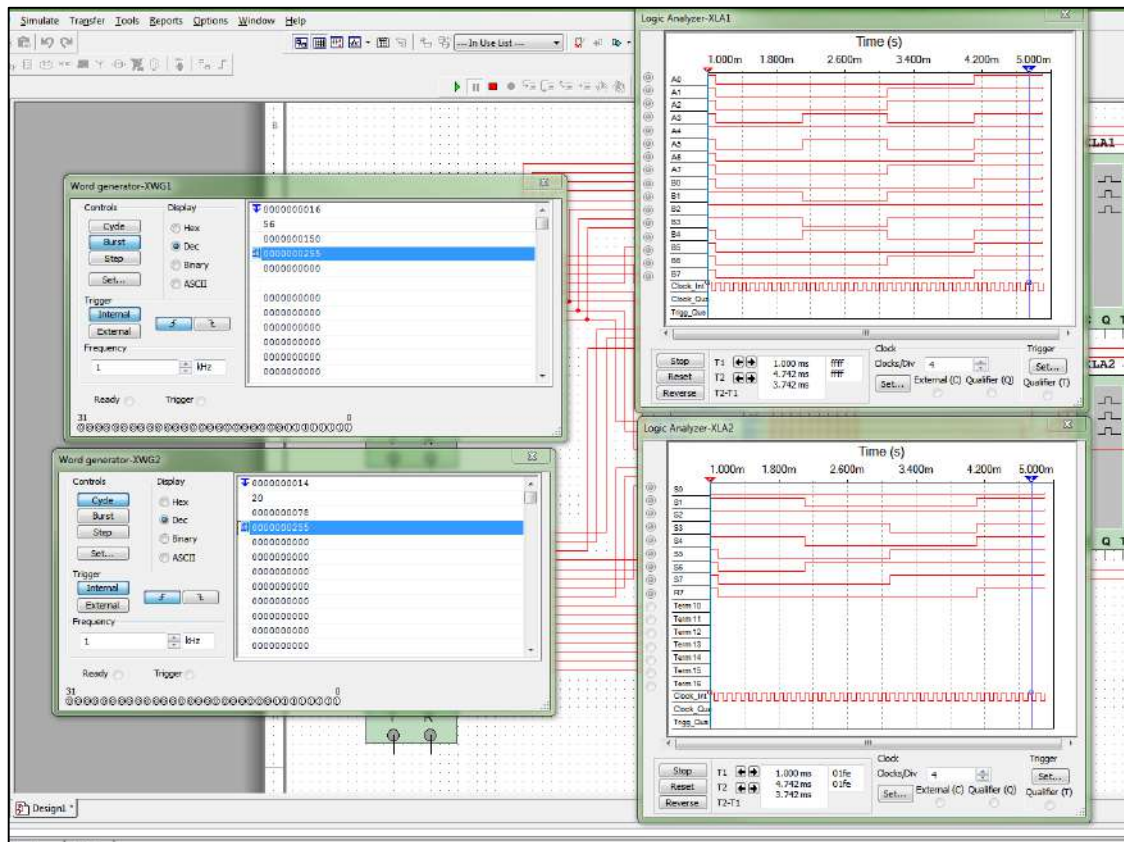
Question 8.1.2)

En utilisant deux 74LS83N, on peut additionner 2 nombres de 8 bits. Il se suffit de connecter la retenue sortante du premier à la retenue sortante du second.

Montage :

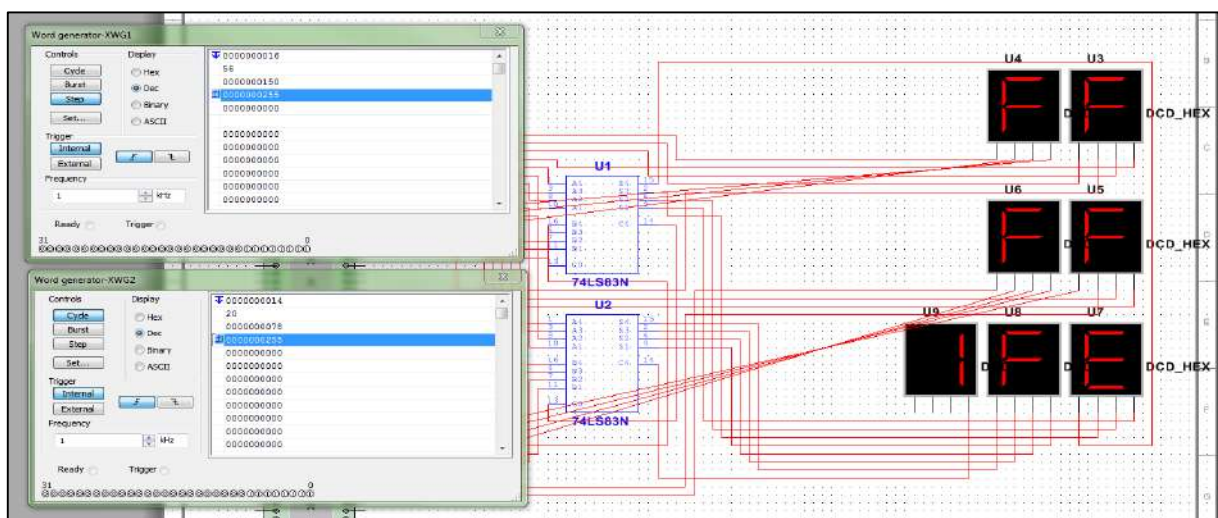


Simulation :



Le Logic Analyser du haut indique les A et B, celui du bas la somme. Par exemple, Au niveau des deux indicateurs bleus, on observe la somme de 1111 et 1111, qui est bien égale à 11111110.

Avec des indicateurs :



FF (255) + FF (255) vaut bien 1FE (510).