

Année 2007-2008

L1 : Information numérique

PROJET: D'INFORMATION NUMERIQUE

SEANCE 1

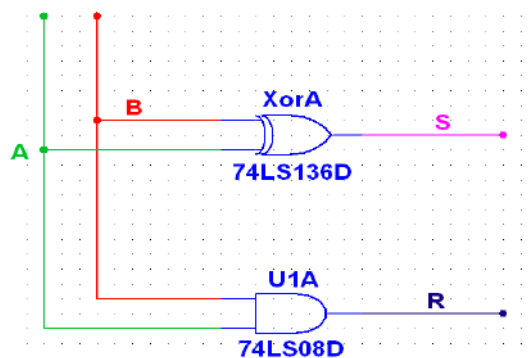
1.1 Demi additionneur

	<i>A</i>		<i>B</i>		<i>R</i>		<i>S</i>
0		0		0		0	
0		1		0		1	
1		0		0		1	
1		1		1		0	

$$S = \overline{A}B \oplus A\overline{B} = A (+) B$$

$$R = AB$$

Schéma demi-additionneur:



1.2 Additionneur complet

Table de vérité

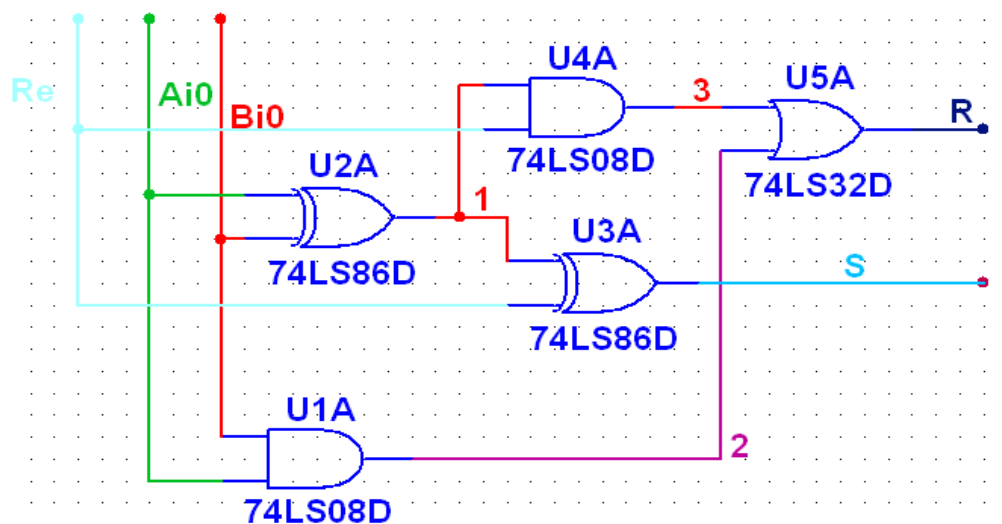
	A_i	B_i	R_i	R_{i+1}	S_i
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1

Équations de sortie:

$$S_i = \bar{A}_i \bar{B}_i R_i \oplus \bar{A}_i B_i \bar{R}_i \oplus A_i \bar{B}_i \bar{R}_i \oplus A_i B_i R_i$$

$$R_{i+1} = B_i R_i \oplus A_i R_i \oplus A_i B_i$$

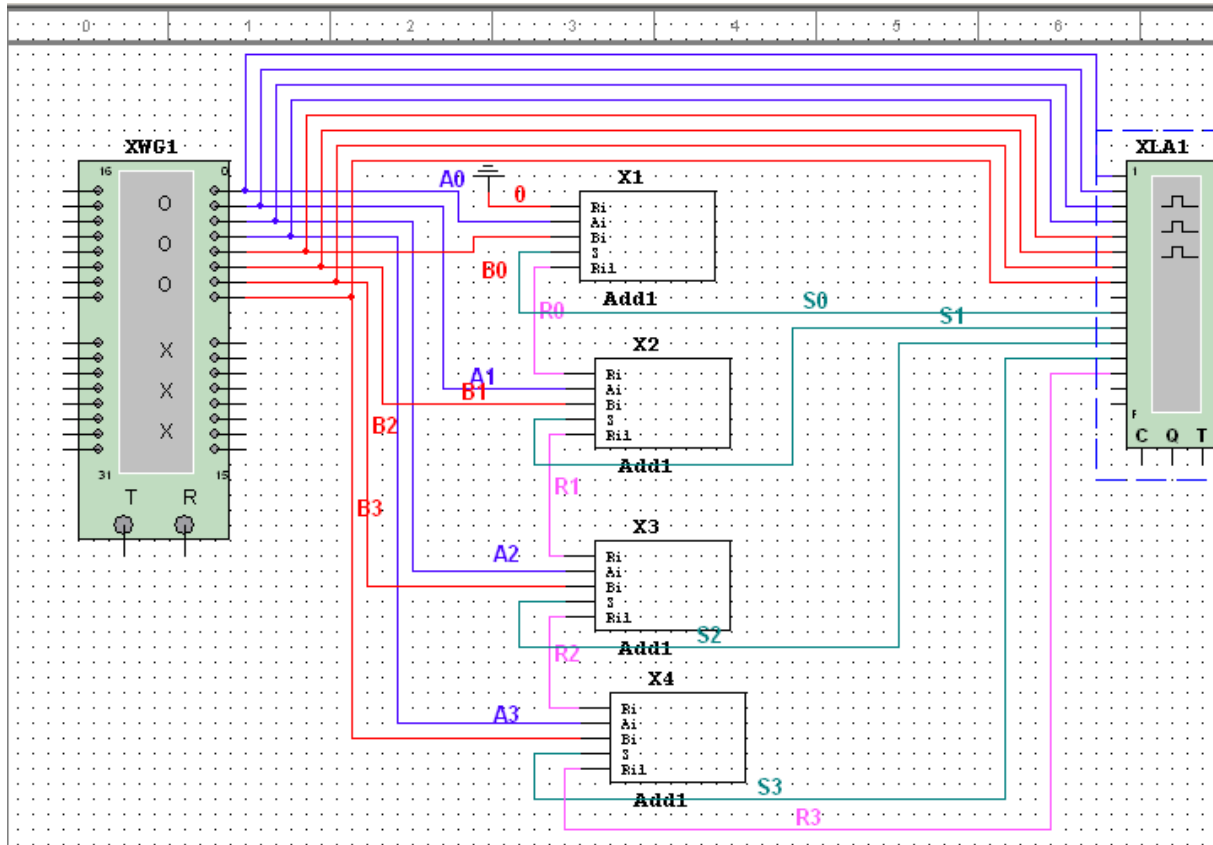
Schéma de l'additionneur complet:



On voit bien qu'un additionneur complet est l'association de deux demi-additionneurs plus une porte OU.

1.3 Additionneur à propagation de retenue

Schéma de l'additionneur à propagation de retenue:



L'énoncé nous dit que le temps total de calcul d'un additionneur à propagation de retenue est de n

$$n * t = T_{tot}$$

(bits de chaque nombre) * (temps de calcul pour un étage) = (Temps de calcul total)

Nos nombres sont formés de 4 bits chacun. Donc $n=4$

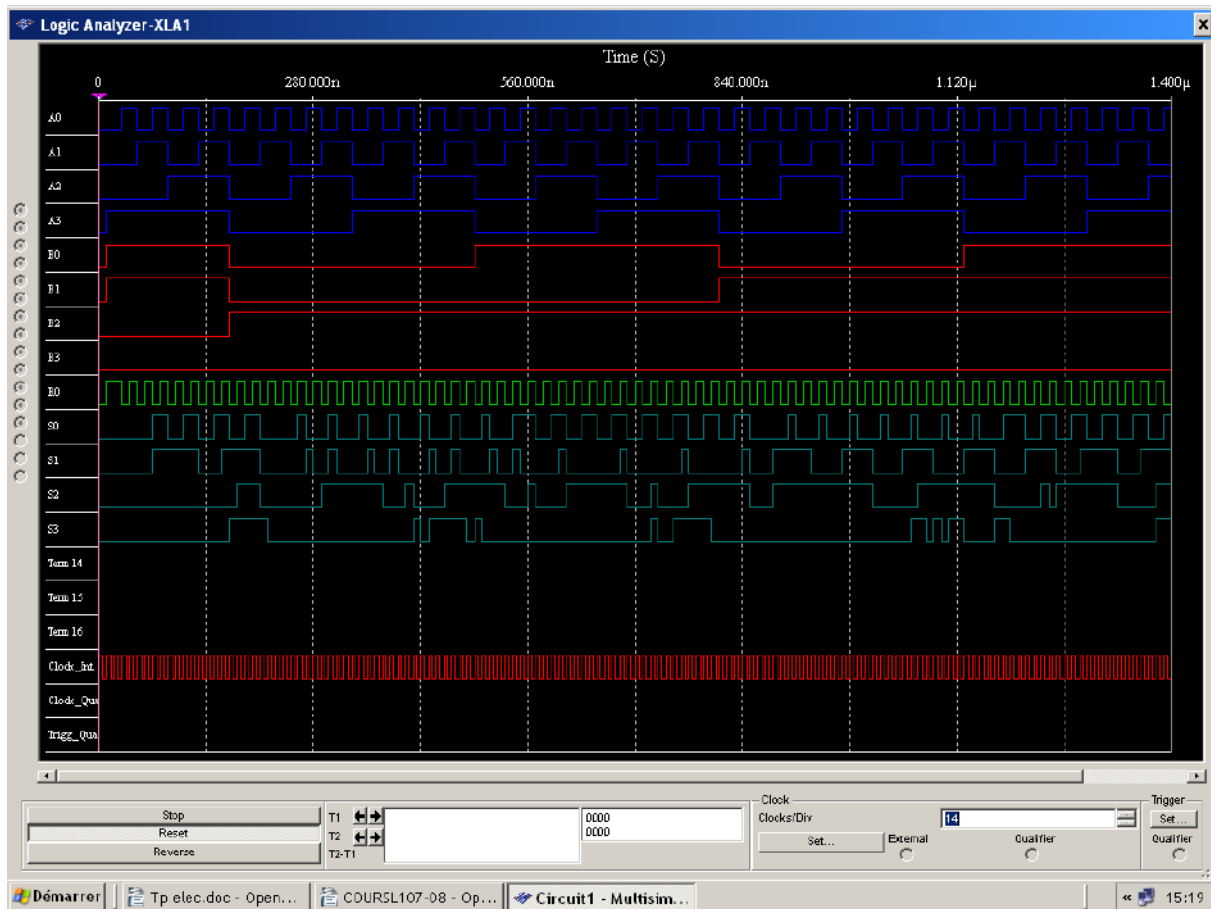
Un étage est formé de 3 passages de portes soit $3 * 10 \text{ ns/porte} = 30 \text{ ns}$

Le temps de calcul d'un étage est donc de 30ns : $t=30$.

On conclue ainsi que le temps **total de calcul d'un additionneur à propagation de retenue de 4bits** est de $30 * 4 = 120 \text{ ns}$.

Ainsi, un **additionneur à propagation de retenue de 8 bits** aurait un **temps de calcul total** de $30 * 8 = 240 \text{ ns}$.

On obtient sur le logic analyser :



1.4 Additionneur à retenue anticipée

L'équation de la retenue d'un additionneur complet est :

$$R_{i+1} = B_i \cdot R_i \oplus A_i \cdot R_i \oplus A_i \cdot B_i$$

$$R_{i+1} = A_i \cdot B_i \oplus R_i \oplus A_i \oplus B_i$$

$$R_{i+1} = G_i \oplus R_i \oplus P_i$$

$$R_i = G_{i-1} \oplus R_{i-1} \oplus P_{i-1}$$

On en déduit :

$$R_0 = 0 \quad (\text{donné})$$

$$R_1 = G_0 \oplus R_0 \oplus P_0$$

$$R_1 = G_0$$

$$R_2 = G_1 \oplus R_1 \oplus P_1$$

$$R_2 = G_1 \oplus G_0 \oplus P_1$$

$$R_3 = G_2 \oplus R_2 \oplus P_2$$

$$R3 = G2 \oplus G1 \oplus G0.P1 \oplus P2$$

L'équation de Si est :

$$Si = A_i \text{ XOR } B_i \text{ XOR } R_i$$

$$S0 = A0 \text{ XOR } B0$$

$$S1 = A1 \text{ XOR } B1 \text{ XOR } R1$$

$$S1 = A1 \text{ XOR } B1 \text{ XOR } G0$$

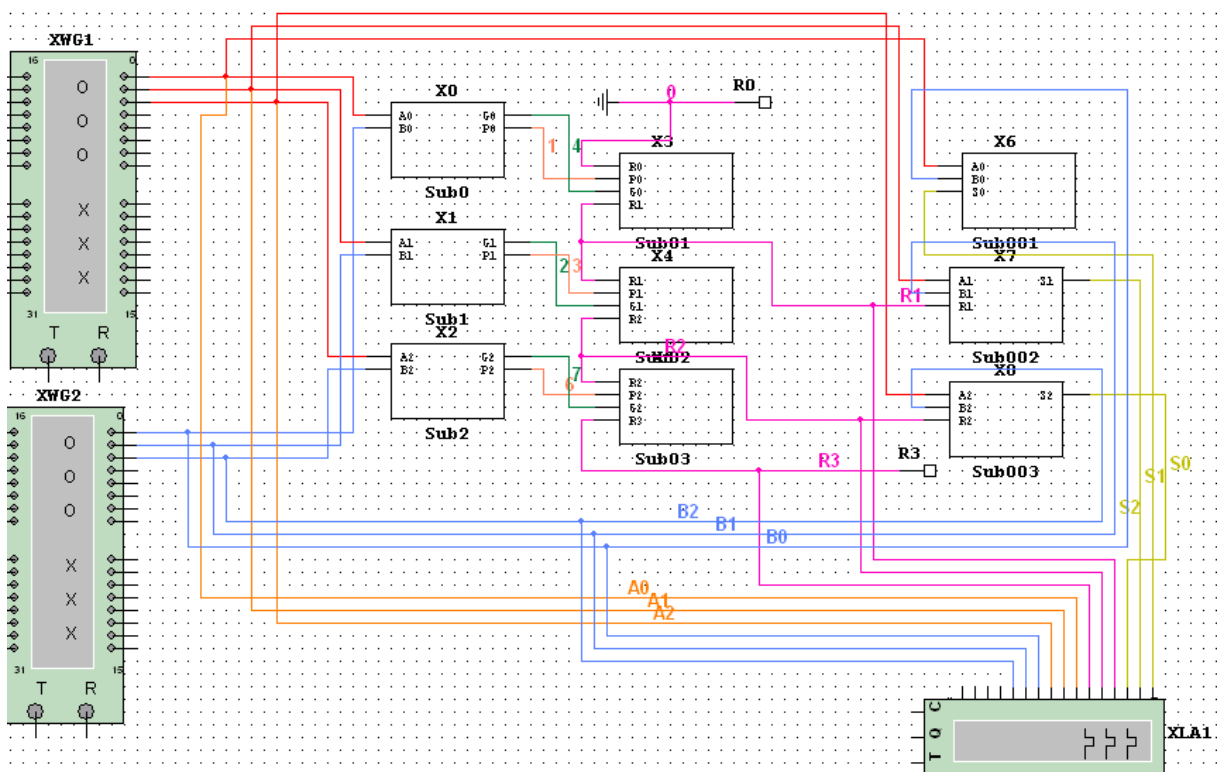
$$S2 = A2 \text{ XOR } B2 \text{ XOR } R2$$

$$S2 = A2 \text{ XOR } B2 \text{ XOR } G1 \oplus G0.P1$$

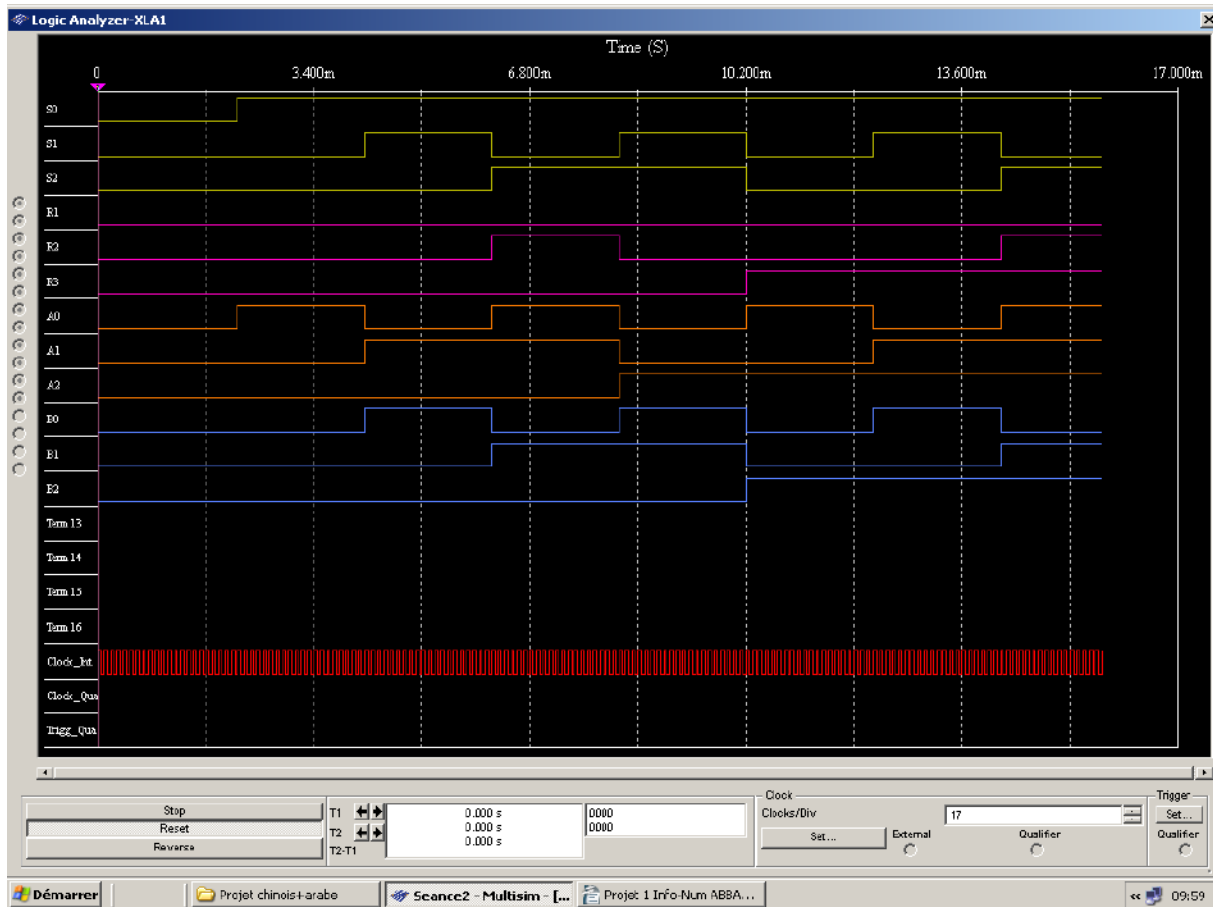
$$S3 = A3 \text{ XOR } B3 \text{ XOR } R3$$

$$S3 = A3 \text{ XOR } B3 \text{ XOR } G2 \oplus G1 \oplus G0.P1 \oplus P2$$

Schéma de l'additionneur à retenue anticipée :



Analyse Graphique de l'additionneur à retenue anticipée :



1.5 Comparaison des deux méthodes

Un demi-additionneur travaille en un temps de 10ns

Un additionneur complet travaille en un temps de 30ns (avec 3 rangées de portes)

Car, au maximum, 3 portes en série sont parcourues (3 rangées de portes).

Un additionneur de 4bits travaille en un temps de 120ns (4 additionneurs complets).

Un additionneur de 8bits travaille en un temps de 240ns (8 additionneurs complets).

$$Gi = Ai \cdot Bi$$

$$Pi = Ai \oplus Bi$$

Ces deux calculs ne nécessitent qu'un passage de porte.
Donc le temps de travaille est de 10ns pour Gi et Pi.

$$Ri = Gi - 1 \square \square Ri - 1 . Pi - 1 \square$$

Pi-1.

Il faut 10ns pour calculer Gi-1 et Pi-1 puis 10ns pour Ri-1 .

Enfin il faut encore 10ns pour effectuer Gi-1 + (Ri-1 . Pi-1).
Soit un total de temps de travail p=30 ns.

Pour un additionneur à retenue anticipée de 4 bits :

$$S_i = \overline{A_i} . B_i . R_{i-1} + \overline{A_i} . B_i . R_i + A_i . \overline{B_i} . R_{i-1} + A_i . B_i . R_i$$

données :

En considérant les conjugués comme des entrées

OU

précédentes.

Il y a dans cette opération 4 portes ET en parallèle avec 3 entrées (Ai, Bi, Ri ou leurs conjugués) ainsi que 1 portes avec pour entrées les 4 sorties des portes ET

Soit un temps total de travail p=**20 ns**.

Pour un additionneur à retenue anticipée de 8 bits, Deux additionneurs 4bits sont en parallèle et leurs sorties respectives passent par une ultime porte ET (rajoutant ainsi un passage de porte supplémentaire) : p=20+10=**30 ns**.

1.5 Comparaison des deux méthodes

En résumé :

un additionneur à propagation de retenue de 4bits =**120 ns**.

un additionneur à propagation de retenue de 8bits =**240 ns**.

un additionneur à retenue anticipée de 4bits =**20 ns**.

un additionneur à retenue anticipée de 8bits =**30 ns**.

	Propagation de retenue	Retenue anticipée	
4bits	120	20	
8bits	240	30	

Le temps de calcul total d'un additionneur à retenue anticipée est largement plus court qu'un additionneur à propagation (rapport de 1/6).

SEANCE 2

Pour les questions du :

1 Fin de la séance 1 : réalisation d'additionneurs

1.1 Additionneur à retenue anticipée

1.2 Comparaison de deux méthodes

Se référer aux pages précédentes de la fin du rapport de la séance 1.

1.3 Réalisation d'un additionneur 8bits

A Walider la semaine prochaine

2 Réalisation d'un additionneur soustracteur

2.1 Réalisation d'un demi-soustracteur de deux éléments binaires.

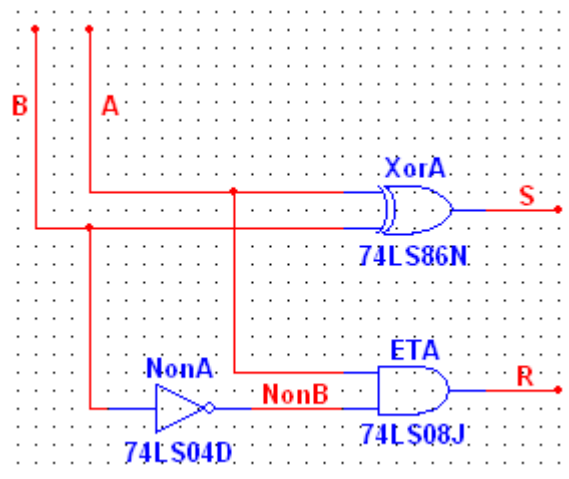
Table vérité de $S_i = B - A$

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

$S_i = A \oplus B$

$R = A \cdot B$

Schéma du demi soustracteur:



2.2 Réalisation d'un soustracteur complet

Schéma d'un soustracteur complet:

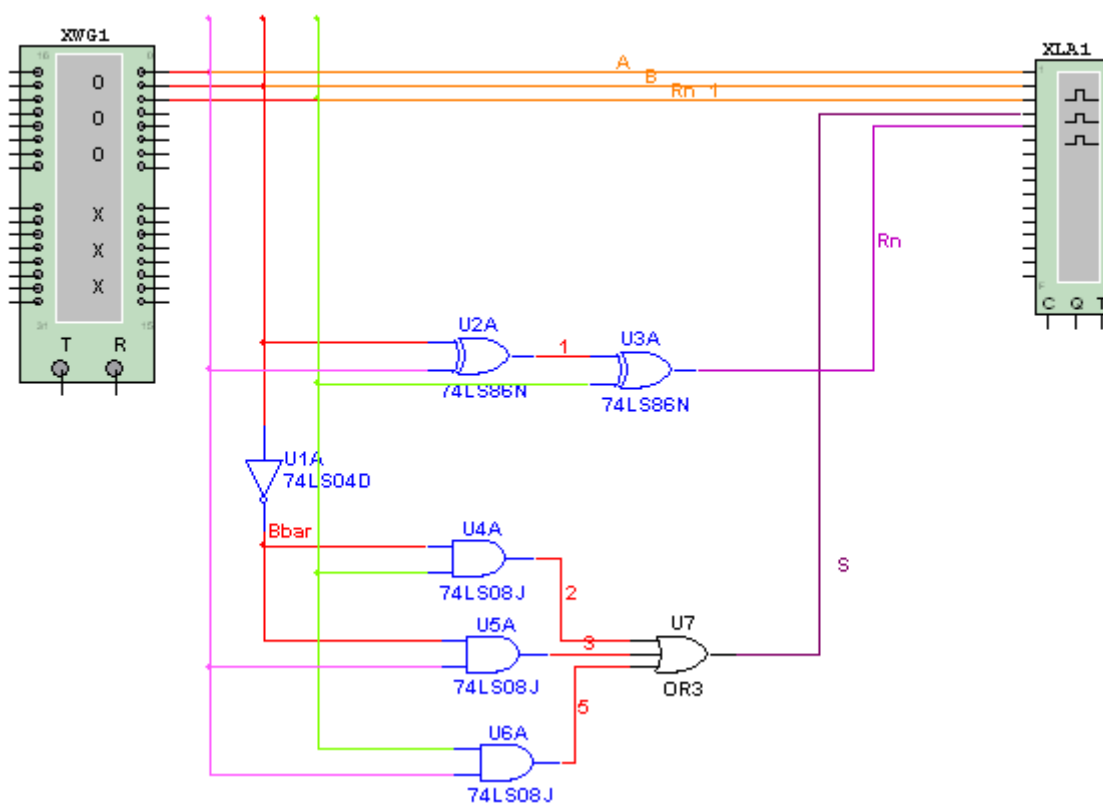


Table de vérité du soustracteur complet $S=B-(A+R_{n-1})$

A	B	R _{n-1}	S	R _n
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	0	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1

Méthode de Karnaugh:

Equation de S

AB//R _{n-1}	0	1
00	0	1
01	0	0
11	0	1
10	1	1

$$S = B \cdot R_{n-1} \oplus A \cdot B \oplus A \cdot R_{n-1}$$

Equation de R_n

AB//R _{n-1}	0	1
00	0	1
01	1	0
11	0	1
10	1	0

L'équation n'est pas simplifiable.

On trouve avec un logic analyser

$$R_n = \overline{A} \cdot B \cdot R_{n-1} \oplus \overline{A} \cdot B \cdot \overline{R_{n-1}} \oplus A \cdot B \cdot R_{n-1} \oplus A \cdot B \cdot \overline{R_{n-1}}$$

Soit :

$$R_n = A \oplus B \oplus R_{n-1}$$

2.3 Réalisation d'un additionneur-soustracteur complet