



# Additionneurs Binaires

Laboratoire Conception Numérique

## Contenu

|     |  |   |
|-----|--|---|
| 1   | Objectif .....                             | 1 |
| 2   | Additionneur à propagation de report ..... | 2 |
| 2.1 | Circuit .....                              | 2 |
| 2.2 | Réalisation .....                          | 2 |
| 2.3 | Simulation .....                           | 2 |
| 3   | Additionneur à prevision de report .....   | 4 |
| 3.1 | Circuit .....                              | 4 |
| 3.2 | Réalisation .....                          | 4 |
| 3.3 | Réalisation .....                          | 5 |
| 3.4 | Simulation .....                           | 5 |
| 4   | Comparaison .....                          | 6 |
| 5   | Checkout .....                             | 7 |

## 1 | Objectif

Ce laboratoire vise à approfondir la compréhension de la représentation binaire des nombres et de leur traitement dans les circuits numériques. Différentes architectures d'additionneurs sont étudiées pour analyser leur fonctionnement, leurs avantages et inconvénients, ainsi que leur impact sur la vitesse de calcul.

Deux variantes d'additionneurs sont mises en avant :

- Additionneur à propagation de retenue : une méthode de base où la retenue se propage progressivement à travers le circuit.
- Additionneur à anticipation de retenue : une version optimisée qui calcule la retenue à l'avance pour réduire le delay.



## 2 | Additionneur à propagation de report

En électronique numérique, les additionneurs sont utilisés pour additionner des nombres binaires. Une implémentation de base est l'additionneur à propagation de report (Carry Ripple Adder). Il est composé de plusieurs blocs d'addition en cascade, dans lesquels la retenue se propage d'une position à l'autre.

### 2.1 Circuit

Un additionneur à propagation de report est composé de plusieurs blocs itératifs, qui additionnent chacun deux bits équivalents ( $a_i, b_i$ ) avec une retenue d'entrée ( $c_i$ ). Chaque bloc génère:

- Un bit de somme  $s_i$ , qui représente le résultat de l'addition.
- Une retenue de sortie  $c_{i+1}$ , qui est transmise au bloc suivant.

La Figure 1 montre le principe de fonctionnement d'un tel additionneur à propagation de report:

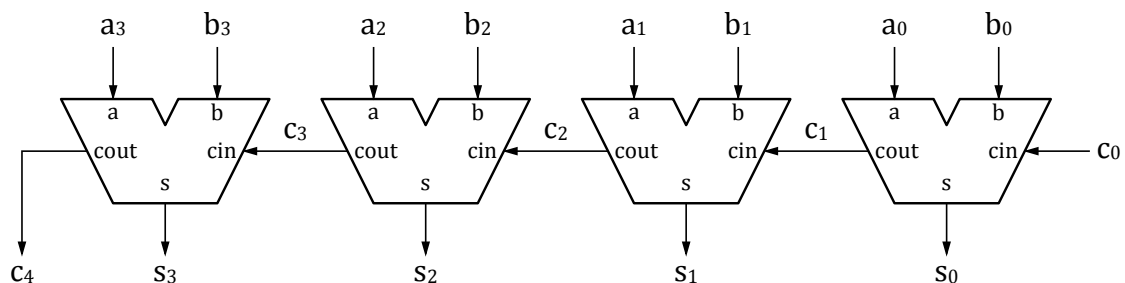


Figure 1 - Additionneur à propagation de report

### 2.2 Réalisation

Commencez par établir la table de vérité, en déduire l'équation logique pour le bit de somme. Enfin développez le circuit combinatoire en utilisant des portes INV, AND, OR, XOR.

Commencez par établir la table de vérité, en déduire l'équation booléenne pour le bit de somme  $s_i$  et la retenue  $c_{i+1}$ . Enfin développez le circuit combinatoire en utilisant des portes INV, AND, OR, XOR.



Implémentez ensuite le schéma itératif d'un additionneur 4 bits à propagation de report dans le projet **ADD/add4**.

### 2.3 Simulation

Chaque circuit implémenté doit être correctement testé. Le banc d'essai **ADD\_test/add4\_tester** doit vérifier le **bon fonctionnement** du circuit. Répondez aux questions suivantes:

1. Combien de tests sont théoriquement nécessaires pour vérifier le bon fonctionnement de l'additionneur?
2. Quels scénarios de test sont nécessaires pour vérifier le bon fonctionnement de l'additionneur ?
3. Combien de temps dure la simulation avec les cas de test pratiques?



Dans le **add4\_tester**, quelques tests d'*exemple* sont déjà implémentés, vous pouvez les utiliser comme inspiration et vous devez les adapter. Dans Listing 1, vous trouverez un tel test, qui vérifie si l'addition  $s = a + b + c_{in} = 0 + 0 + 0 = 0$  est correcte. Sinon, le message **test 1 wrong** est affiché dans le terminal Modelsim.

```
1  -----
2  -- Test 1:    0 + 0 + 0 = 0
3  --
4  a  <="0000";
5  b  <="0000";
6  cin <='0';
7  WAIT FOR clockPeriod;
8  assert (cout = '0') AND (s="0000")
9      report "test 1 wrong"
10     severity note;
```

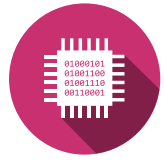
Listing 1 - Exemple de cas de test 1



Complétez le **ADD\_test/add4\_tester** avec les tests nécessaires à la vérification complète du fonctionnement de l'additionneur.

Simulez le band de test **ADD\_test/add4\_tb** avec le fichier de simulation **\$SIMULATION\_DIR/ADD1.do**.

Enquêtez et trouvez le chemin de propagation le plus long dans l'additionneur.



### 3 | Additionneur à prevision de report

Cet additionneur est optimisé pour réduire le délai en calculant la report à l'avance.

#### 3.1 Circuit

La Figure 2 montre le circuit d'un additionneur à prevision de report. Il est similaire à l'additionneur à propagation de report. Cependant, la chaîne de report a été interrompue par 4 nouveaux blocs. Ces blocs doivent calculer les reports  $c_0, c_1, c_2$  et  $c_3$  respectives avec moins de portes séquentielles. Tout en réduisant la cascade/chaîne de portes logiques pour limiter le délai de propagation.

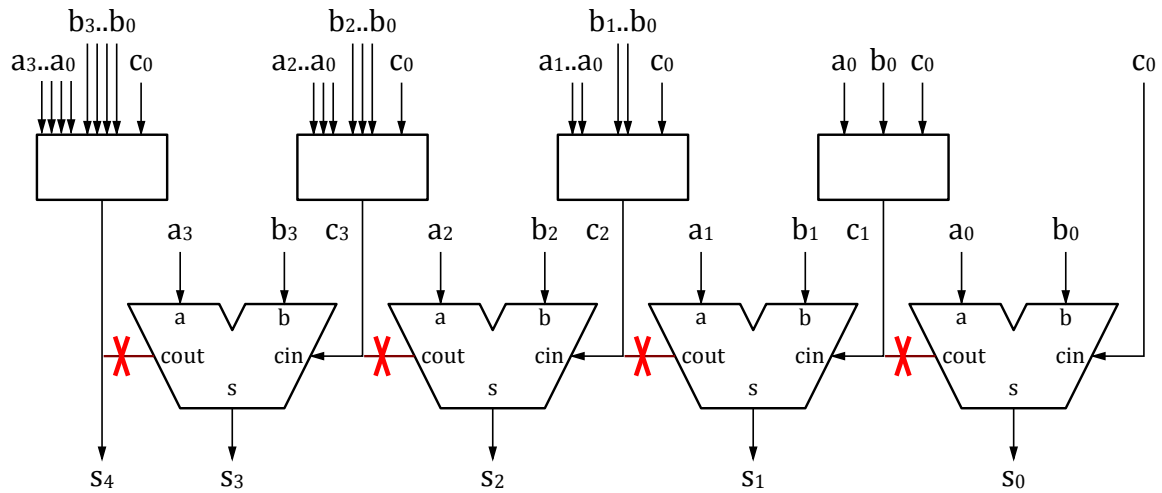


Figure 2 - Additionneur à prévision de report

#### 3.2 Réalisation

Analysons d'abord l'additionneur de la Figure 1.



Écrivez l'équation polynomiale de  $c_1$  et  $c_2$ . Estimez le nombre de termes de l'équation polynomiale de  $c_3$  et  $c_4$ .

Pour simplifier la création des reports d'entrée, deux transformations sont introduites,  $g_i$  (générer) et  $p_i$  (propager):

$$\begin{aligned} g_i &= a_i * b_i \\ p_i &= a_i + b_i \end{aligned} \quad (1)$$

Montrez que la carry de sortie et le bit de somme peuvent être écrits comme suit:

$$\begin{aligned} c_{out} &= g + p * c_{in} \\ s &= (\overline{g} * p) \oplus c_{in} \end{aligned} \quad (2)$$

$$\begin{aligned} c_{i+1} &= g_i + p_i * c_i \\ s_i &= (\overline{g_i} * p_i) \oplus c_i \end{aligned} \quad (3)$$



En utilisant cette transformation, écrivez la forme polynomiale de  $c_1$  à  $c_4$  en fonction de  $g_i$ ,  $p_i$  et  $c_0$ .

Combien de termes y a-t-il dans ce cas?

### 3.3 Réalisation

À l'aide de portes INV, ET, OU et XOR, dessinez le schéma du bloc qui calcule les reports (**ADD/lookAheadCarry**).



Ajoutez ensuite le schéma itératif de l'additionneur 4 bits à anticipation de report fourni (**ADD/addLookAhead4**).

### 3.4 Simulation

Réfléchissez combien de tests sont nécessaires pour vérifier le bon fonctionnement de l'additionneur.



Simulez le band de test **ADD\_test/addLookAhead4\_tb** avec le fichier de simulation **\$SIMULATION\_DIR/ADD2.do**.



## 4 | Comparaison

Calculez et comparez la propagation maximale des deux additionneurs créés, l'un avec propagation de report Figure 1 et l'autre avec prevision de report Figure 2.



Quels avantages offre l'additionneur à prevision de report par rapport à l'additionneur à propagation de report? Comment se comportent-ils en termes de vitesse et de taille du circuit?



## 5 | Checkout

Avant de quitter le laboratoire, assurez-vous d'avoir accompli les tâches suivantes :

- ☐ Conception du circuit terminée
  - ☐ L'additionneur avec propagation de report a été conçu et testé.
  - ☐ L'additionneur avec prévision de report a été conçu et testé.
- ☐ Simulations
  - ☐ Les tests spécifiques des différentes bancs de test ont été adaptés au circuit.
  - ☐ Les deux additionneurs ont été comparés en termes de performances et d'utilisation des ressources?
- ☐ Documentation et fichiers de projet
  - ☐ Assurez-vous que toutes les étapes (conception, circuit, simulations) sont bien documentées dans votre rapport de laboratoire.
  - ☐ Enregistrez le projet sur une clé USB ou le lecteur réseau partagé (\\filer01.hevs.ch).
  - ☐ Partagez les fichiers avec votre partenaire de laboratoire pour assurer la continuité du travail.