Additionneurs Binaires

Laboratoire Conception Numérique

Contenu

1 Objectif

Ce laboratoire vise à apprendre la conception de circuits arithmétiques itératifs et à les mettre en œuvre de manière pratique. L'accent est mis sur le développement d'un additionneur en tant que circuit arithmétique de base.

De plus, il est montré comment les additionneurs créés peuvent être utilisés pour réaliser un soustracteur. Par des modifications ciblées, le concept de soustraction binaire est enseigné, permettant de développer une compréhension approfondie des opérations arithmétiques dans les circuits numériques.

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:

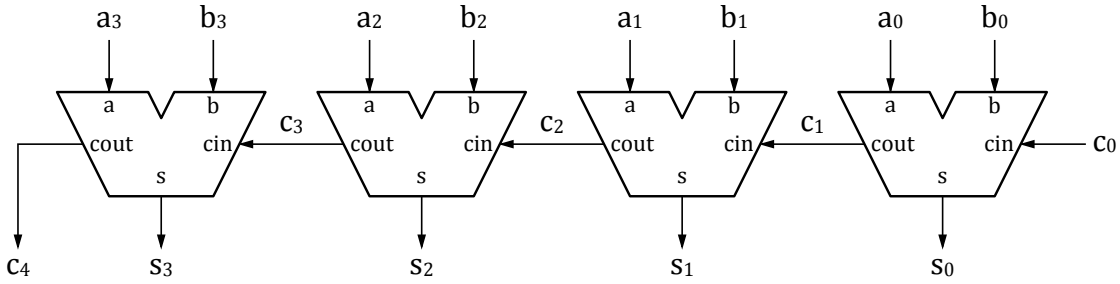


Figure 1 - Additionneur à propagation de report

2.2 Réalisation

Commencez par étable 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

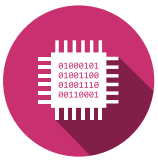
Commencez par établir la table de vérité, en déduire l'équation booléenne pour le bit de somme s_i et la retenue c_{i+1}. Enfin développez le circuit combinatoire en utilisant des portes INV, AND, OR, XOR.

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 $s = a + b + c_{\text{in}} = 0 + 0 + 0 = 0$ est correcte. Sinon, le message **test 1 wrong** est affiché dans le terminal Modelsim.

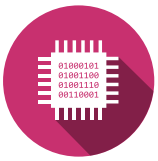
```
1  --------------------------------------------------------------------------------
2  -- Test 1:    0 + 0 + 0 = 0
3  --
4  a   <="0000";
5  b   <="0000";
6  cin <='0';
7  WAIT FOR clockPeriod;
8  assert (cout = '0') AND (s="0000")
9    report "test 1 wrong"
10   severity note;
```

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 Soustracteur

Un soustracteur est un circuit qui effectue la soustraction de deux nombres. Comme nous l'avons appris dans le cours, il est possible d'inverser le signe d'un nombre en complément à deux. Cela forme la base d'un soustracteur.

3.1 Circuit

Le circuit d'un soustracteur peut être réalisé sur la base de l'additionneur précédemment développé. Pour effectuer cette soustraction, on peut ajouter le complément de la valeur à soustraire:

$$a - b = a + (-b) \tag{1}$$

On obtient le complément à deux d'un nombre en inversant tous les bits du nombre et en ajoutant 1 à ce résultat intermédiaire. L'inversion de tous les bits se fait avec un inverseur bit à bit. L'addition de 1 peut être effectuée en agissant sur la toute première retenue de l'additionneur **`c_0`**.

Dans le bloc **`ADD_Test/sub8_tb`**, il y a 2 additionneurs de 4 bits chacun, chaînés pour former un additionneur de 8 bits. Il y a aussi un bloc pour inverser les bits du nombre b.

3.2 Réalisation

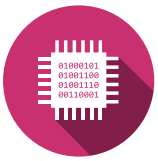
Modifiez le circuit pour créer un soustracteur 8 bits.

Dessinez le schéma du bloc **`forSubtraction`**, qui inverse les bits du nombre b. Expliquez pourquoi le signal c_{in} est inversé.

3.3 Simulation

Chaque circuit implémenté doit être correctement testé.

Simulez le band de test **`ADD_Test/sub8_tb`** avec le fichier de simulation **`$SIMULATION_DIR/ADD3.do`**.



4 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.
- [ ] Simulations
  - [ ] Les tests spécifiques des différentes bancs de test ont été adaptés au circuit.
- [ ] 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.