HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Prof. Yann Thoma

Laboratoire de vérification VHDL
Additionneur BCD combinatoire
semestre printemps 2016 - 2017

Composant à tester

Nous souhaitons tester un module capable d'additionneur deux nombres codés en BCD (Binaire Codé Décimal). Le nombre de digits de l'entrée est un paramètre générique `NDIGITS` (dénoté N dans la suite du document).

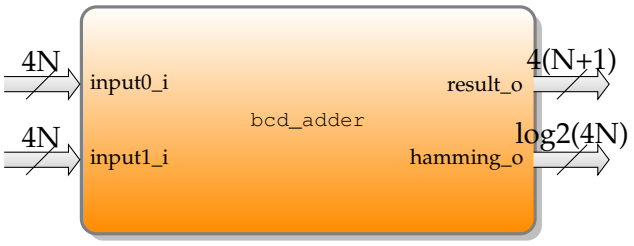
Spécifications

Le système doit réaliser deux opérations. La première est l'addition des deux entrées, fournies en BCD. Chaque entrée, sur $4N$ bits, code donc une valeur comprise entre 0 et $10^N - 1$. Le résultat sera sur $4(N+1)$ bits BCD.
La deuxième opération du système consiste à convertir les deux nombres en binaire, puis de fournir la distance de Hamming séparant ces deux nombres binaires. La distance de Hamming correspond au nombre de bits différents entre deux vecteurs de bits.

Entrées/sorties

Les entrées/sorties sont :

Nom	Taille	Dir	Description
`input0_i`	$4N$	in	Premier nombre
`input1_i`	$4N$	in	Deuxième nombre
`result_o`	$4(N+1)$	in	Résultat
`hamming_o`	$log2(4N)$	out	distance de Hamming



Il est important de noter que pour des raisons de bonne pratique de design, les ports ne sont pas de simples vecteurs, mais des `bcd_number`. Un `bcd_number` étant défini dans le fichier `bcd_pkg.vhd` comme :

```
type bcd_digit is std_logic_vector(3 downto 0);
type bcd_number is array(<>) of bcd_digit;
```

Si l'entrée ne correspond pas à des nombres BCD, alors la sortie est indéfinie.
Un paramètre générique `ERRNO` permet d'injecter artificiellement des erreurs dans le design. Il s'agit d'un entier qui offre le comportement suivant :

1. S'il est compris entre 0 et 3, le résultat est valide ;
2. S'il est compris entre 4 et 7, le résultat n'est pas valide.

L'additionneur possède un paramètre générique, `NDIGITS`, qui permet de définir le nombre de digits de l'additionneur. Le banc de test devra être capable de le prendre en compte.
Un deuxième paramètre générique, `ERRNO`, permet de générer des comportements différents Ce paramètre générique vous permettra de valider votre banc de test, en l'essayant avec toutes les valeurs de `ERRNO`. Ce paramètre générique est aussi un paramètre du banc de test. Il permet de lancer plusieurs simulations sans recompiler le code. Observez, dans le fichier `sim.do` où se situe la détermination de ce paramètre. Vous pouvez imaginer lancer automatiquement plusieurs simulations de suite en modifiant ce paramètre dans le fichier `sim.do`.
⚠ L'entité à tester ne doit pas être modifiée.

Exercice 1

1. Reprenez les fichiers fournis, et lancez une simulation avec QuestaSim. La simulation doit être lancée depuis le répertoire `sim`, avec la commande
   `do ../scripts/sim.do all X`
   où `X` est la valeur de `ERRNO` voulue.
2. Modifiez le fichier `sim.do` de manière à grouper automatiquement les signaux `*_sti` et `*_obs`.
3. Le banc de test fourni ne fait que stimuler le système de manière très basique. Modifiez-le de manière à disposer d'une procédure pour l'application des entrées du DUT.
4. Ajoutez de quoi simuler différents scénarios.
5. Dans le choix de vos scénarios, effectuez quelques tests dirigés puis exploitez l'aléatoire.

Exercice 2

Le banc de test n'effectue aucune vérification.

1. Modifiez-le de manière à vérifier le bon fonctionnement du DUT, dans le processus existant.
2. Arrangez-vous pour disposer d'une procédure pour la vérification.
3. Faites des tests avec différentes valeurs de `ERRNO` pour observer le bon fonctionnement de votre banc de test. Pour ce faire vous pouvez passer un argument au script `sim.do` correspondant au numéro de l'erreur.
4. Affichez des informations pertinentes permettant d'identifier facilement les erreurs. Intéressant : la fonction `to_hstring()`.
5. Développez un type protégé pour pouvoir centraliser les messages. Une variable partagée pourra alors être exploitée par les différents processus pour centraliser les informations (messages, nombre d'erreurs, ...)
6. Affichez en fin de simulation le nombre d'erreurs observées.
7. Modifiez le banc de test de manière à ce que la vérification se fasse dans un processus dédié, commandé par un signal de synchronisation.