HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD

Prof. Yann Thoma

Laboratoire VSE

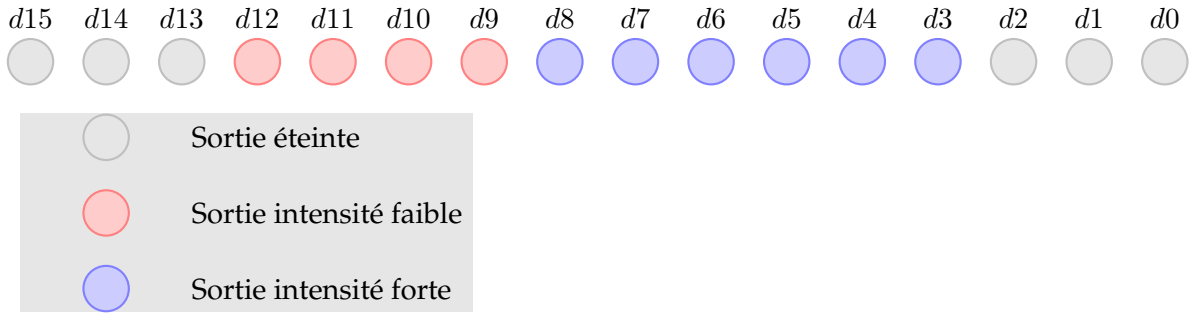
semestre d'automne 2024 - 2025

Affichage linéaire d'une valeur entre deux bornes

Composant à tester

Nous souhaitons tester un système permettant l'affichage linéaire d'une valeur. L'affichage sera commandé de façon à indiquer où se situe cette valeur par rapport à deux bornes Min et Max. L'exemple ci-dessous présente la sortie pour les valeurs suivantes :

- Max = 12
- Min = 3
- Valeur = 8



Paramètres génériques

Le composant est générique en taille. Le paramètre générique `VALSIZE` représente la taille des vecteurs représentant les valeurs d'entrée.
Un deuxième paramètre générique, `ERRNO`, a une fonctionnalité expliquée plus loin dans ce document.

Entrées/sorties

Les entrées/sorties sont :

Nom	Taille	Dir	Description
`Com_i`	2	in	signal de commande de 2 bits
`Val_i`	`VALSIZE`	in	valeur d'entrée à visualiser sur l'afficheur linéaire, signal de `VALSIZE` bits
`Min_i`	`VALSIZE`	in	borne inférieure de la plage, signal de `VALSIZE` bits
`Max_i`	`VALSIZE`	in	borne supérieure de la plage, signal de `VALSIZE` bits
`Osc_i`	1	in	signal d'oscillation pour obtenir une intensité faible
`Leds_o`	`2**VALSIZE`	out	Afficheur linéaire composé de leds, signal de $2^{VALSIZE}$ bits

Fonctionnement

L'intensité forte d'une led correspond à l'enclenchement constant de celle-ci. L'intensité faible d'une led est obtenue par l'enclenchement alternatif 1-0 de la led avec le signal `Osc_i`. Ainsi, la puissance émise est plus faible. Il s'agit donc, dans le cas de l'intensité faible, de combiner l'allumage avec le signal `Osc_i`. La sortie sur les Leds correspond donc à un `ET` logique entre la sortie prévue et `Osc_i`.

Commande	Fonction	Description
00	Marche normale	Si Valeur est compris entre Max et Min (bornes incluses) — les leds de Min à Val sont allumées avec intensité forte — les leds de (Val+1) à Max sont allumées avec intensité faible — toutes les autres leds sont éteintes Si Valeur est hors de l'intervalle [Max Min] toutes les leds sont éteintes
01	Mode linéaire	Affichage de Val en linéaire, Leds de 0 à Val sont allumées avec une intensité forte
10	Test éteint	Toutes les leds de l'afficheur sont éteintes (état '0')
11	Test allumé fort	Toutes les leds de l'afficheur sont allumées avec une intensité forte (état '1')

⚠ Pour un fonctionnement normal, Max doit être plus grand que Min. Si ce n'est pas le cas, la sortie sur les leds est indéfinie !
La version du DUT qui vous est fournie est parfaitement fonctionnelle, toutefois un paramètre générique `ERRNO` permettra d'injecter artificiellement des erreurs dans le design. Il s'agira 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 16 et 21, le résultat n'est pas valide.

Ce paramètre générique vous permettra de valider votre banc de test, en le simulant avec toutes les valeurs de `ERRNO`.

Etape 1

1. Reprenez les fichiers fournis, et lancez une simulation avec QuestaSim. Pour ce faire vous avez deux options :
   (a) Via l'interface graphique, allez dans le répertoire `sim` et lancez `do ../scripts/sim.do`.
   (b) Dans le terminal, allez dans le répertoire `sim` et lancez `vsim -c -do ../scripts/sim.do`.
2. Ajoutez le strict minimum pour pouvoir stimuler le système avec un seul testcase basique.
3. Testez le banc de test avec différents paramètres génériques. Vous pouvez le faire en lançant la simulation avec des paramètres (regardez dans le script l'ordre des paramètres) :
   (a) Via l'interface graphique, allez dans le répertoire `sim` et lancez `do ../scripts/sim.do all 0 5 0`
   (b) Dans le terminal, allez dans le répertoire `sim` et lancez `vsim -c -do "do ../scripts/sim.do all 0 5 0"`

Etape 2

1. Définissez les scénarios que vous voudriez jouer. Prenez bien conscience du problème de la taille `VALSIZE`. Il doit être possible de disposer d'un scénario acceptant de grandes valeurs.
2. Pensez également à exploiter l'aléatoire et de la couverture.

Etape 3

Le testbench a un paramètre générique `TESTCASE`, qui vise à pouvoir choisir le testcase à jouer. Utilisez ce paramètre. Afin de pouvoir automatiser les tests, faites que le testcase 0 lance tous les testcases à la suite.

1. Modifiez votre banc de test de manière à implémenter les testcases définis à l'étape 2.

Etape 4

Le banc de test n'effectue aucune vérification.

1. Modifiez-le de manière à vérifier le bon fonctionnement du DUV, dans la procédure correspondante.

Etape 5

Le Verification Run Manager permet de vérifier plusieurs cas. Un fichier `default.rmdb` est présent dans le répertoire `code`. Vous pouvez lancer la commange `vrun directed` pour démarrer toutes les vérifications (`vrun -gui directed` pour la version GUI).

1. Ajoutez des *runnable* dans `default.rmdb` de manière à tester des valeurs plus grandes de `VALSIZE` avec le/les testcases pertinents.

Travail à rendre

— Générez une archive du projet à l'aide du script `vse_rendu.sh`. Ce script vous est fourni avec le code et l'énoncé du laboratoire. Il génère une archive pour autant que le rapport (fichier `rapport.pdf`) et le fichier testbench soient présents. Attention donc à vous trouver dans le bon répertoire. Il faut donc que le script se trouve dans le même répertoire que le fichier `rapport.pdf` et que le répertoire `code`.
— Copiez l'archive sur Cyberlearn.

Barème de correction

Documentation	20%
Pertinence des testcases	10%
Fonction de vérification	10%
Exploitation d'aléatoire	10%
Exploitation de la couverture	10%
Fonction de vérification	10%
Détection / non détection correcte	10%
Codage	10%
Commentaires au niveau du code	10%