

Laboratoire VSN

semestre de printemps 2020 - 2021

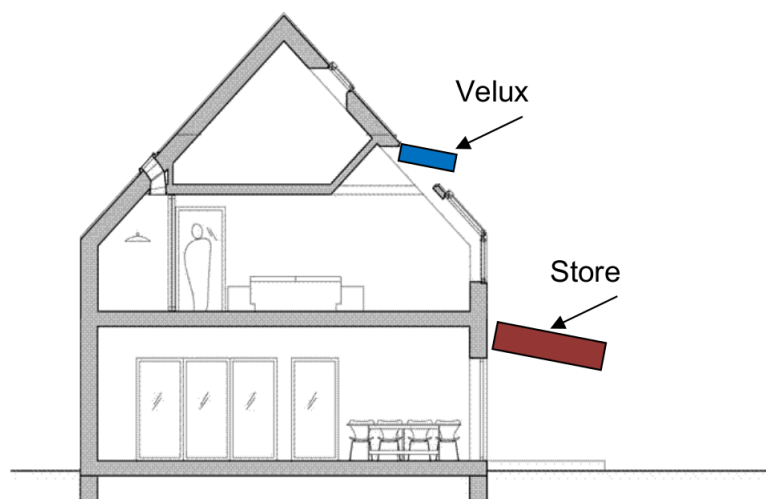
Commande de velux et store

Composant à tester

Nous souhaitons régler la température de plusieurs pièces d'une maison par une commande électronique. Nous disposons de capteurs qui nous fournissent des informations sur la température, l'ensoleillement et s'il pleut ou vente. D'autre part nous disposons de deux actuators, soit l'entraînement électrique du velux et du store.

Description du système

Voici la vue de la maison avec le velux et le store.



Description des entrées/sorties :

Nom	Dir.	Taille	Description
temp_i	in	N	entrée de N bits indiquant la température intérieure
sun_i	in	N	entrée de N bits indiquant l'ensoleillement
rain_i	in	1	entrée un bit. L'état actif ('1') indique qu'il pleut
wind_i	in	1	entrée un bit. L'état actif ('1') indique qu'il souffle
velux_o	out	3	ouverture du velux commandée par une valeur codée sur 3 bits (0 à 7)
store_o	out	8	déroulement du store commandé par une valeur linéaire sur 8 bits

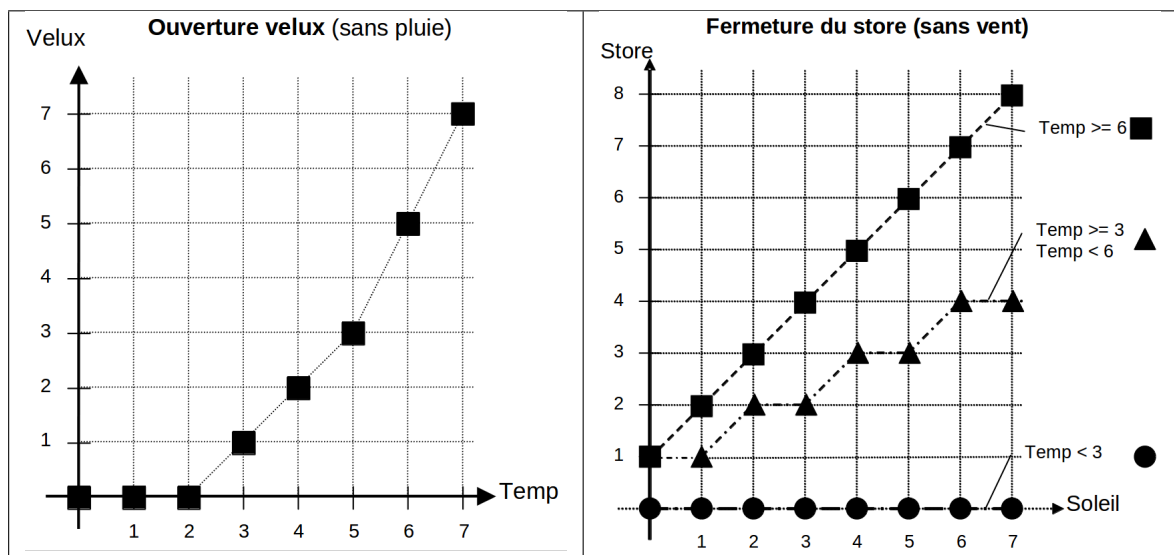
N est un entier supérieur ou égal à 3.

Comportement du store :

Fermé	0	0	0	0	0	0	0	0	Store 0% déroulé
Ouverture :									
1 ^{er} cran	0	0	0	0	0	0	0	1	
2 ^{ème} cran	0	0	0	0	0	0	1	1	
.....									
8 ^{ème} cran	1	1	1	1	1	1	1	1	Store 100% déroulé

Description du fonctionnement :

- L'ouverture du velux dépend de la température intérieure selon la relation définie par la courbe de réglage (voir graphique ci-dessous) pour autant qu'il ne pleuve pas.
- Lorsque le détecteur de pluie est actif, le velux doit être fermé indépendamment de la température.
- Le déroulement du store dépend de l'ensoleillement et de la température intérieure. Le graphique ci-dessous vous indique la correspondance selon trois courbes qui dépendent de la température dans la pièce pour autant qu'il ne vente pas.
- Lorsque le détecteur de vent est actif, le store doit être enroulé quelles que soient les conditions d'ensoleillement ou de température.



Sur les deux graphiques ci-dessus, la température et l'ensoleillement sont encodés sur 3 bits. Un passage à l'échelle est donc nécessaire en fonction du nombre de bits disponibles. Si N est le nombre de bits, et que $Velux_3(t)$, pour $t \in [0, 7]$ correspond au schéma ci-dessus, alors :

$$Velux_N(t) = Velux_3(\lfloor t / (2^{N-3}) \rfloor)$$


La même transformation est à appliquer pour la gestion du store.

⚠ Le store a une particularité : S'il pleut et qu'il y a beaucoup de soleil (valeur 6 ou 7 du schéma ci-dessus), alors le comportement du store est indéfini. Oui, c'est un peu bizarre, mais le fabricant a considéré qu'il n'était pas possible qu'il y ait énormément de soleil en même temps que de la pluie, et donc ses spécifications ont mené à un choix de design un peu particulier.

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 2, le résultat est valide ;
2. S'il est compris entre 3 et 8, 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`.

 L'entité à tester ne doit pas être modifiée.

Trois bancs de tests générés avec TbGenerator vous sont fournis. Profitez d'observer leur structure, et exploitez `code3` pour réaliser le labo.

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 `comp` et lancez

```
do ../scripts/sim.do.
```
 - (b) Dans le terminal, allez dans le répertoire `comp` 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 `comp` et lancez

```
do ../scripts/sim.do all 0 5 0
```
 - (b) Dans le terminal, allez dans le répertoire `comp` et lancez

```
vsim -c -do "do ../scripts/sim.do all 0 5 0"
```
4. Récupérez le logger que vous avez réalisé lors du premier labo. Faites en sorte de le compiler via `sim.do`, et de pouvoir l'utiliser dans le testbench.

Etape 2

1. Définissez les scénarios que vous voudriez jouer.
2. Pensez également à exploiter l'aléatoire.
3. Utilisez le fichier `com_store_velux.xml` pour lister les testcases.

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.
2. Modifiez la constante `ERRNO`, via le lancement de `sim.do`, pour observer le bon fonctionnement de votre banc de test.

Etape 5

Le Verification Run Manager permet de vérifier plusieurs cas. Un fichier `default.rmdb` est présent dans le répertoire `code3`. Vous pouvez lancer la commande `vrun directed` pour démarrer toutes les vérifications (`vrun -gui directed` pour la version GUI).

1. Modifiez les commandes de `default.rmdb` si vous avez ajouté des fichiers sources afin que tout se passe parfaitement.
2. Ajoutez également les différents testcases, en y faisant le lien avec le fichier `com_store_velux.xml`.

Il y a évidemment un nombre conséquent de combinaisons de `N`, `TESTCASE`, et `ERRNO`. Le fichier `default.rmbd` définit déjà 8 tests exploitant le testcase 0 et essayant tous les `ERRNO` différents. Laissez ces tests tel quel, et ajoutez les vôtres avec `ERRNO=0`.

Après avoir lancé `vrun`, exécutez la commande `vsim -viewcov VRMDATA/merge.ucdb` afin d'observer la couverture de vos tests.

Travail à rendre

- Générez une archive du projet à l'aide du script `vsn_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 `code3`.
- Copiez l'archive sur Cyberlearn.

Barème de correction

Documentation	20%
Pertinence des testcases	20%
Fonction de vérification	10%
Détection / non détection correcte	30%
Codage	10%
Commentaires au niveau du code	10%