Document Explicatif – Testbench du module controle_barriere

Projet : Système de gestion intelligente de parking

Ce document explique en détail le fonctionnement du **testbench tb_controle_barriere**, utilisé pour vérifier le comportement du module VHDL *controle_barriere*. Il est rédigé de façon simple et claire pour aider tous les membres du groupe à comprendre comment la simulation valide la logique du module.

---

1. Rôle du testbench

Le testbench sert à **simuler** le fonctionnement de la barrière sans matériel réel. Il permet de :

- fournir des signaux externes au module (horloge, capteurs, ordre d'ouverture),
- observer les réactions du module (activation des moteurs),
- vérifier que la FSM suit bien les transitions prévues.

---

2. Structure générale du testbench

Le testbench contient 5 grandes parties :

1. Déclaration du module à tester (DUT)
2. Création des signaux qui imitent les entrées/sorties
3. Instanciation du DUT
4. Génération de l'horloge
5. Scénario de simulation (stimuli)

---

3. Déclaration du composant : le DUT

Le testbench commence par déclarer le composant **controle_barriere** afin de pouvoir l'utiliser.

- Les ports définis sont identiques à ceux du fichier du module.
- Cela permet ensuite de connecter les signaux du testbench à ces ports.

---

4. Déclaration des signaux de test

Le testbench crée ensuite des signaux locaux qui joueront le rôle :

Des entrées du module

- `s_clk` : horloge
- `s_rst` : reset
- `s_trigger_open` : ordre d'ouverture
- `s_sensor_passage` : capteur de passage
- `s_sensor_open_limit` : capteur limite haute
- `s_sensor_closed_limit` : capteur limite basse

Des sorties du module

- `s_motor_open`
- `s_motor_close`

Ces signaux permettent d'agir sur le module comme si on avait du vrai matériel.

Un paramètre important :

- **CLK_PERIOD = 10 ns**, ce qui correspond à une horloge de **100 MHz**.

---

5. Instanciation du module (DUT)

Le testbench instancie le module à tester sous le nom **DUT** :

- chaque port du module est relié au signal correspondant du testbench,
- ainsi, on peut envoyer des stimuli et lire les sorties.

---

6. Génération de l'horloge

Le processus :

```
s_clk <= '0'; wait for CLK_PERIOD/2;
s_clk <= '1'; wait for CLK_PERIOD/2;
```

crée une onde carrée régulière.

Cette horloge synchronise tous les processus internes du module
(*comme en vrai FPGA*).

---

7. Processus de simulation (scénario complet)

C’est la partie essentielle : elle simule un cas réel.

Le scénario se déroule en **3 phases**.

Phase 1 : RESET du système

Objectif : mettre la FSM dans son état initial.

- `s_rst` passe à `'1'` pendant 20 ns.
- Puis retour à `'0'`.

À ce moment-là :

- la FSM doit être en **IDLE_CLOSED**,
- `motor_open` et `motor_close` doivent être `'0'`.

C’est le démarrage normal du système.

Phase 2 : Scénario d’ouverture

Le module principal demande l’ouverture :

1. `s_trigger_open <= '1'` pendant 1 cycle
2. Retour à `'0'` (c’est une impulsion)

Ce qui doit arriver :

- la FSM entre dans **OPENING**,
- `s_motor_open` doit passer à `'1'`.

Ensuite, on simule le temps d’ouverture :

- Attente de 50 ns
- Puis activation du capteur `sensor_open_limit <= '1'` pendant 1 cycle

Le module doit alors :

- entrer dans **IDLE_OPEN**,

- éteindre le moteur d’ouverture.

On simule ensuite le passage de la voiture :

- Attente de 100 ns

Phase 3 : Scénario de fermeture

La voiture a quitté la zone de la barrière :

- `s_sensor_passage <= '1'` pendant 1 cycle

Ce qui doit arriver :

- la FSM passe à **CLOSING**,
- `s_motor_close` doit passer à `'1'`.

Ensuite :

- Attente 50 ns pour simuler le temps de fermeture
- Activation de `sensor_closed_limit <= '1'` pendant 1 cycle

Résultat attendu :

- FSM revient à **IDLE_CLOSED**,
- les moteurs sont à `'0'`.

La simulation se termine avec `WAIT;`.

---

8. Ce que permet de vérifier ce testbench

Ce testbench valide :

- le bon fonctionnement de la FSM,
- les transitions entre états,
- l’activation correcte des moteurs,
- l’absence de conflits moteur (les deux jamais à `'1'`),
- la cohérence des capteurs dans le cycle d’ouverture/fermeture.

Il reproduit fidèlement un cycle complet et réaliste de la barrière :
reset → ouverture → attente → fermeture → repos.

---

9. Conclusion

Le testbench fournit un environnement complet et cohérent pour vérifier le module *controle_barriere*.
Grâce à un scénario clair et progressif, il permet à n'importe quel membre du groupe de comprendre :

- comment le module doit réagir,
- comment tester ou modifier la FSM,
- comment ajouter d'autres scénarios si nécessaire.