Document Explicatif – Module VHDL « controle_barriere »

Projet : Système de gestion intelligente de parking

Ce document a pour objectif d'expliquer clairement le fonctionnement du module **controle_barriere** afin que tous les membres du groupe puissent comprendre la logique, modifier le code si nécessaire et l'intégrer correctement au reste du projet. Le but est d'avoir une compréhension simple, concrète et non technique au-delà de ce qui est strictement nécessaire.

---

1. Rôle du module

Le module **controle_barriere** gère uniquement le mouvement mécanique de la barrière : - il ouvre la barrière quand on lui en donne l'ordre, - il attend que la voiture passe, - puis il referme la barrière.

Ce module **ne décide pas** si une voiture est autorisée à entrer ou non. Cette décision est prise par le **module principal**, qui lui envoie simplement le signal « ouvre maintenant ».

Le module est conçu sous forme d'une **Machine à États Finis (FSM)**, ce qui assure un comportement stable, prévisible et facile à maintenir.

---

2. Interface du module : les signaux échangés

Le tableau ci-dessous résume les ports d'entrée et de sortie. Il permet de comprendre comment le module « discute » avec le reste du système.

Entrées

- **clk** : Horloge (ex. 100 MHz) synchronisant la FSM.
- **rst** : Réinitialisation. Remet la barrière en état fermé.
- **trigger_open** : Impulsion envoyée par le module principal pour lancer le cycle d'ouverture.
- **sensor_open_limit** : Signal du capteur indiquant que la barrière est totalement ouverte.
- **sensor_passage** : Signal indiquant que la voiture est passée sous la barrière.
- **sensor_closed_limit** : Signal du capteur indiquant que la barrière est totalement fermée.

Sorties

- **motor_open** : Commande pour activer le moteur dans le sens « ouverture ».
- **motor_close** : Commande pour activer le moteur dans le sens « fermeture ».

Schématiquement, on peut visualiser ce module comme une **boîte noire** qui reçoit des informations de l'extérieur (capteurs, ordre d'ouverture) et renvoie des commandes moteur.

---

3. La machine à états (FSM)

Le fonctionnement est découpé en **4 états**, qui représentent chaque étape du mouvement de la barrière.

1. IDLE_CLOSED – Barrière fermée (état initial)

- La barrière est en bas.
- Aucun moteur n'est activé.
- On attend le signal `trigger_open`.

2. OPENING – Barrière en ouverture

- Le moteur d'ouverture est activé.
- On reste dans cet état tant que la barrière n'a pas atteint la position haute.

3. IDLE_OPEN – Barrière ouverte

- La barrière est en haut, moteurs coupés.
- On attend que la voiture passe (`sensor_passage`).

4. CLOSING – Barrière en fermeture

- Le moteur de fermeture est activé.
- Quand la position basse est atteinte (`sensor_closed_limit`), retour à IDLE_CLOSED.

L'ensemble forme un cycle très simple : **Fermé → Ouverture → Ouvert → Fermeture → Fermé**.

---

4. Structure interne du code : la méthode des 3 processus

Le code VHDL utilise la méthode classique des **3 processus**, qui sépare clairement : 1. la mémoire (registre d'état), 2. la logique décisionnelle (quel est le prochain état ?), 3. les actions (que faire dans cet état ?).

Cette structure améliore la lisibilité et réduit le risque de bugs.

Processus 1 – Registre d'état (stockage)

Ce processus mémorise l'état actuel de la FSM. - À chaque **front montant de l'horloge**, l'état est mis à jour. - Si `rst = 1`, la barrière retourne automatiquement à l'état fermé.

C'est la partie la plus simple : un enregistrement d'état.

Processus 2 – Logique du prochain état (décision)

Ce processus décide de la transition d'état selon : - l'état actuel, - les capteurs, - le signal `trigger_open`.

Par exemple : - Si on est en *IDLE_CLOSED* et que `trigger_open = 1`, on passe en *OPENING*. - Si on est en *OPENING* et que la barrière est totalement ouverte, on passe en *IDLE_OPEN*.

Ce processus implémente directement le diagramme d'état.

Processus 3 – Logique des sorties (actions)

Ce processus contrôle les moteurs selon l'état actuel. - **OPENING → motor_open activé** - **CLOSING → motor_close activé** - **IDLE_CLOSED & IDLE_OPEN → moteurs arrêtés**

Les valeurs par défaut (`motor_open = 0`, `motor_close = 0`) garantissent qu'aucun moteur ne reste activé accidentellement.

---

5. Intégration dans le projet

Dans le module principal (ou top-level) : - on instancie ce module, - on lui fournit `clk` et `rst`, - on lui envoie une impulsion sur `trigger_open` quand une voiture est autorisée à entrer ou sortir, - on récupère ses sorties `motor_open` et `motor_close` pour piloter le matériel ou la simulation.

Le gros avantage est que **toute la gestion mécanique est centralisée** dans ce module. Le module principal n'a pas à connaître les états internes, les capteurs ou la séquence du mouvement : il dit juste « ouvre » et la FSM fait le reste.

---

6. Notes

- Le module est autonome : il gère entièrement l'ouverture et la fermeture.
- La structure en 3 processus rend le code clair et modulaire.
- Le module principal se contente d'envoyer un ordre simple.
- La FSM garantit un comportement propre, sécurisé et sans conflit de moteurs.

Ce document peut servir de base pour les tests, les modifications futures ou l'intégration dans le top-level.