Gestion de parking

Objectif :

1. Modéliser un système de gestion de parking de voitures

2. Validation et simulation du système de gestion de parking

3. test et validation en utilisant l'objet **assert** condition **report** " message" **severity** [note, error, warning, failure]

Descriptif

Le but de ce système est de gérer la disponibilité des places d'un parking. En effet il permet à travers la sortie nb_places_dispos de donner le nombre de places disponibles dans un parking. Si aucune place n'est disponible la sortie complet =1.

à travers les entrées capt_entrée et capt_sortie (en provenances de deux capteurs de présence), le système met à jours le nombre de place disponibles.

à chaque fois une voiture entre dans le parking, le compteur rajoute +1 et met à jours le nombre de places disponibles (soustraction avec le nombre maximale disponible).

à chaque fois une voiture sort du parking, le compteur soustrait 1 de count et met à jours le nombre de places disponibles (soustraction avec le nombre maximale disponible).

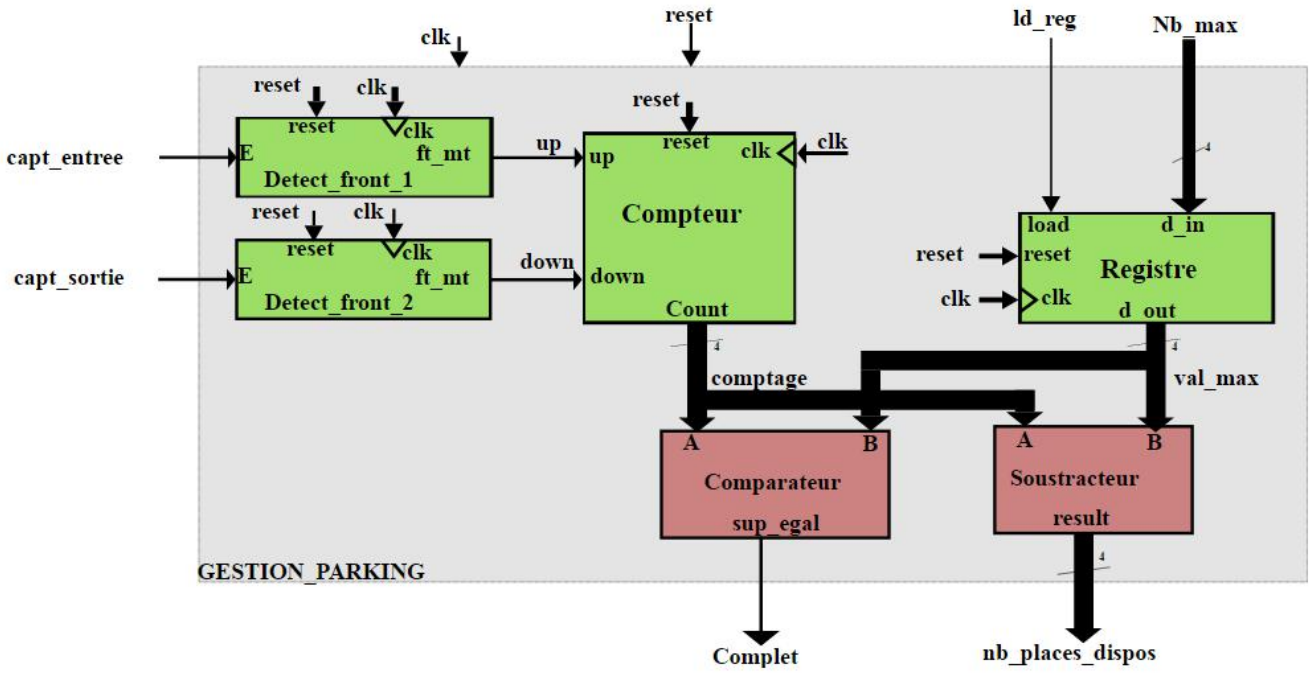

Figure 1 : architecture du système de gestion du parking

L'élément principal du système est un compteur qui s'incrémente si le capteur d'entrée vaut 1 et se décrémente si le capteur de sortie vaut 1.

Un registre permet de stocker le nombre de places maximales du parking.

Une comparaison entre la valeur de comptage et celle du registre permet d'indiquer si le parking est complet ou pas.

1. Décrire en VHDL Synthétisable le composant compteur/décompteur 4 bits avec un reset asynchrone.

si up =1 --> comptage

si down --> décomptage

2. Décrire en VHDL Synthétisable le composant registre :

- Le registre permet de stocker et par conséquent de modifier facilement la capacité maximale du parking en nombre de places.

- utiliser l'objet **assert** condition **report** " message" **severity** [note, error, warning, failure] pour valider le fonctionnement de votre registre.

- vérifier les différents types de **sevirity** et noter la différence dans le comportement du simultateur.

3. Décrire en VHDL synthétisable le composant comparateur : il permet de comparer la sortie du compteur (nombre de voiture dans le parking) avec le nombre maximale du parking.

utiliser une instruction concurrente conditionnelle.

4. Décrire en VHDL Synthétisable le composant soustracteur

5. décrire le composant Detect_front_1 en VHDL synthétisable.

Le capteur peut fournir une information à 1 pendant plusieurs coups d'horloge. Or le compteur ne doit s'incrémenter ou se décrémenter qu'une fois pour une voiture.

La solution consiste à utiliser une machine d'état pour détecter un front montant en provenance du capteur.

A l'issue d'un front montant de l'entrée, la sortie vaut 1 pendant une période de clk (figure ci dessous).

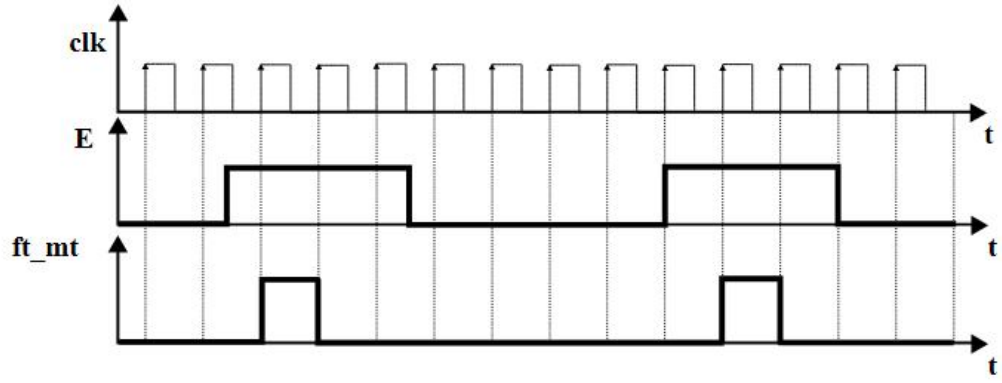

figure 2 : histogramme du Detect front

5.1 : donner le digramme d'état de ce composant.

5.2. décrire le fonctionnement de ce composant en VHDL synthétisable et le valider avec les signaux de la figure 2.

6. Décrire le composant Gestion_Parking en instanciant tous les composants déjà décrits, compilés et existants dans la bibliothèque work de Modelsim puis le valider.