Université Paul Sabatier

Rapport BE VHDL

Mots clés : VHDL, Anémomètre, FPGA, DE0, PWM, Synthèse, Quartus, Altera

---

Réalisé par :

TOUIL Hajar
DEDJEH Ove-Onselme
KOUSSAIMY MOUNIR

Encadré par :

Mr. P. CARVALHO

2022-2023

Table des matières

I. Introduction

Dans le cadre de notre formation M2 Systèmes et microsystèmes embarqués, plusieurs séances ont été organisées comme un Bureau d'études pour synthétiser et mettre en œuvre des systèmes, simples et complexes. Le but de ce bureau d'études c'est de mettre en œuvre une solution de synthèse, matérielle et logicielle pour piloter automatiquement un voilier de barre franche d'un bateau.

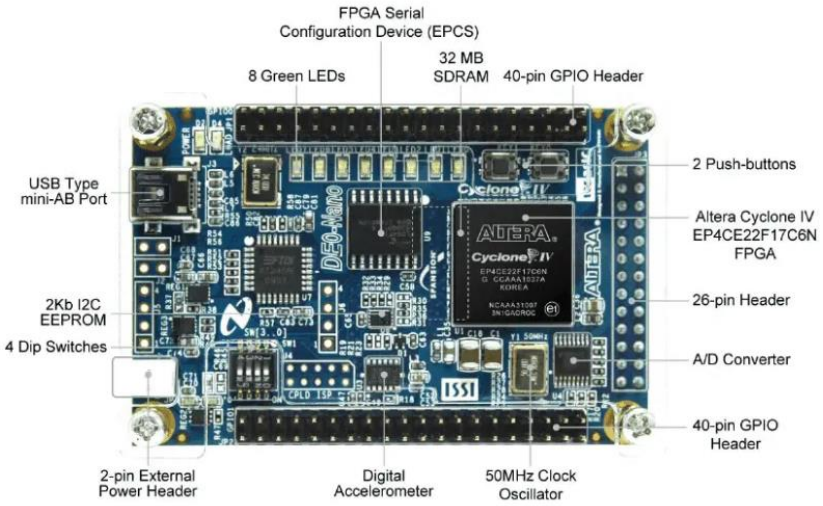
Pour réaliser ce système, on commence d'abord à prendre en main l'utilisation du logiciel Quartus et le langage VHDL, grâce aux TPS de base proposés par notre encadrant.

Ensuite, on s'intéresse à analyser les besoins et les spécifications que le système doit satisfaire, et faire un découpage fonctionnel et logiciel du système en utilisant un algorithme et code en VHDL et l'implémenter sur la carte DE0 (Cyclone IV).

II. Cahier des charges du projet « Barre-Franche »

Ce projet porte sur la réalisation, d'un dispositif embarqué basé sur un FPGA de chez Altéra : le DE0 nano. On doit développer les deux parties matériel et logiciel afin de mettre en œuvre l'asservissement d'une barre franche. La partie hardware se basera sur un code VHDL implémenté sur un FPGA et la partie Software intégré sera développé en C. le lien entre les deux parties sera établi par l'implémentation d'un bus de donnée : le bus Avalon. Afin de fournir un système complètement fonctionnel, différents blocs fonctionnels devront être implémenté sur le FPGA :

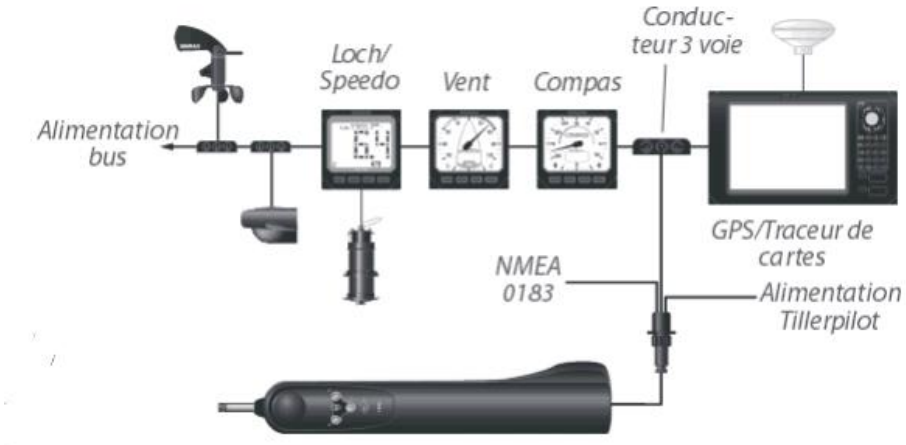
- Un bloc qui permettra de lire la mesure de la vitesse du vent (0-250km/h). Il devra lire la sortie de l'anémomètre qui est une sortie logique de fréquence variable (0 à 250 Hz).
- Un bloc générant un signal PWM qui sera utilisé dans plusieurs parties du projet. Il sera intégré plus tard dans le SOPC du FPGA permettant la génération d'un signal PWM qu'on utilisera au travers du Bus Avalon.
- Un bloc de gestion vérin gérant le pilotage de la barre franche
- Un bloc qui utilise un compas pour récupérer les mesures d'angles sur le plan horizontale du voilier, permettant de donner un cap à celui-ci
- Un bloc qui permet la gestion de l'interface Homme système (composée de différents boutons [Bâbord, Tribord, Auto/Manuel], des LEDS ainsi qu'un buzzer).
- Une implémentation d'un MCU dans le FPGA grâce à l'outil SOPC du logiciel d'Altéra. Celui-ci permettra le traitement et l'affichage des différentes variables du projet (cap du voilier, position vérin, position GPS, gestion des PWM et la vitesse du vent).



Carte DE0 nano

III. Pilote de barre Franche analyse et spécification :

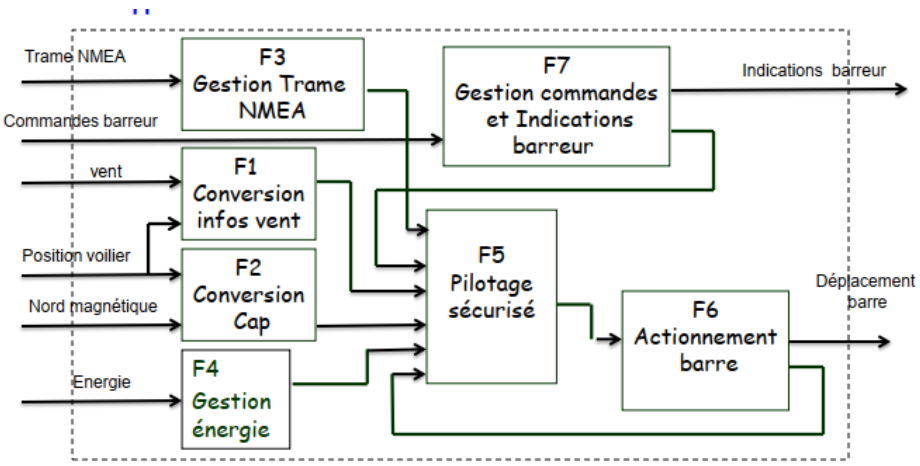
Un pilote de barre franche est un équipement permettant de diriger le bateau et maintenir le cap. Il est constitué d'un compas, qui a comme fonction la récupération des informations de la direction du vent, ou ce qu'on appelle la girouette, il délivre ensuite une différence de potentielle proportionnelle à l'écart de routé détecté, d'une partie électronique qui interagit avec le compas pour comparer à chaque instant la valeur donnée par le compas et la valeur du cap souhaité, et actionner le vérin qui représente la partie puissance du système dès que le bateau s'écarte de sa trajectoire.



Dans l'intégralité, le cahier de charge nous impose plusieurs fonctions et exigences pour arriver à la finalité de ce système qu'on liste ci-dessous :

1. Mesurer la fréquence et détecter les informations liées à la vitesse du vent fournies par l'anémomètre.
2. Récupérer les informations de trajectoire réelle fournies par une boussole.
3. Récupérer la trajectoire réelle d'un GPS

4. Interagir avec l'utilisateur via des LEDs, des boutons et des bips sonores.
5. Commander le vérin
6. Et récupérer l'écart d'angle de déviation avec la girouette.



D'ici on voit bien qu'il s'agit d'un système complexe, c'est pour cela il nous a été demandé de travailler sur deux fonctions dont une fonction simple et une autre complexe.
Nous avons choisi donc la fonction anémomètre comme fonction simple et gestion du vérin comme fonction complexe.

IV. Conception d'une fonction simple : l'anémomètre

La fonction réalisant l'acquisition de la vitesse du vent (0 à 250 Km/h) se fait à l'aide d'un anémomètre qui sert à convertir la vitesse du vent en fréquence variable (0 à 250 Hz).
Pour cela, nous avons développé différents composants pour réaliser la fonction "anémomètre" :

1. Générateur d'impulsion à 1Hz
2. Détecteur de fronts montants
3. Compteur
4. Registre

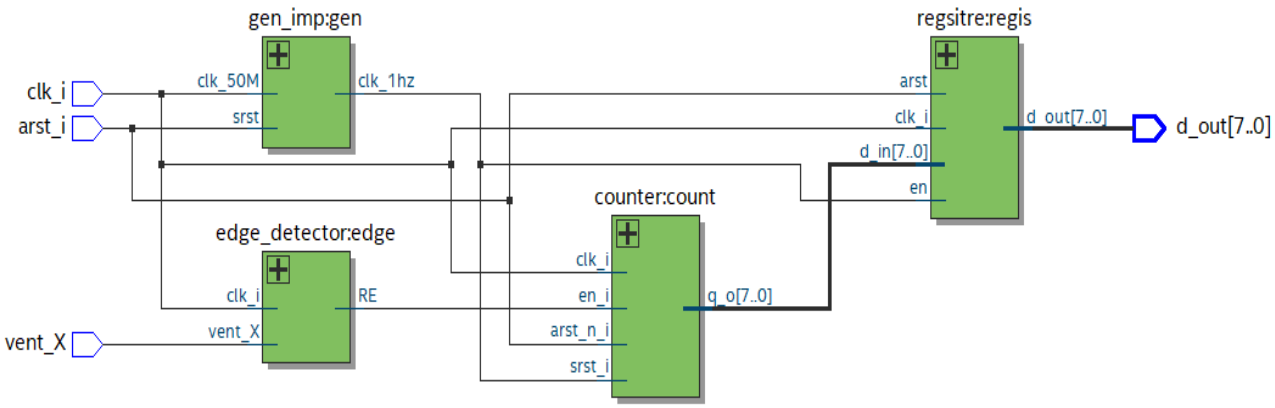


Schéma fonctionnel de l'anémomètre

1. Générateur d'impulsion à 1Hz

Le composant "gen_imp" génère une horloge de 1 Hz à partir de l'horloge 50 MHz donnée par l'horloge du FPGA. L'horloge de 1 Hz sera utilisée pour l'acquisition de la fréquence de l'anémomètre toutes les 1 seconde.

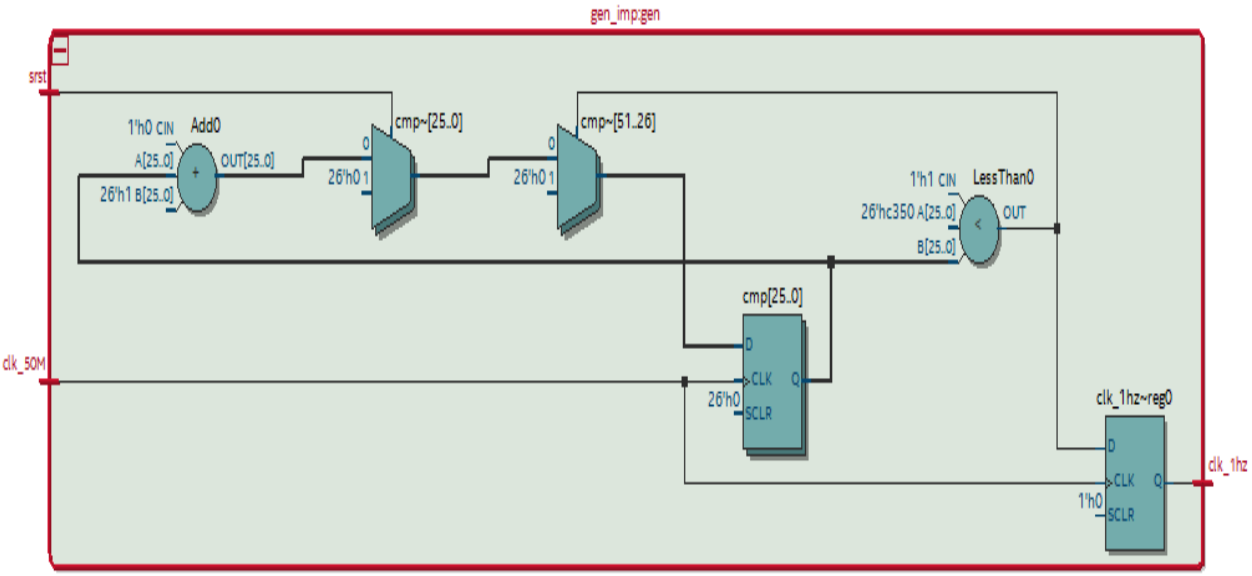


Schéma fonctionnel du générateur d'impulsion

2. Détecteur de fronts montants

Le composant "edge_detector" sert à détecter chaque front montant du signal « vent » généré pour transformer la fréquence générée en Hertz en une donnée interprétable par le composant "counter".

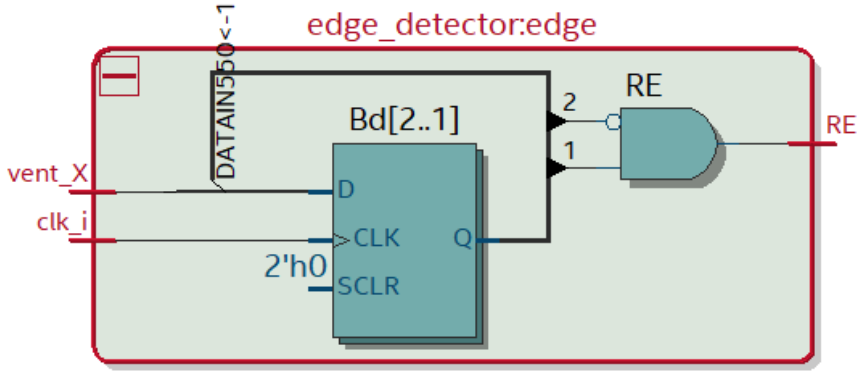


*Schéma fonctionnel du d*étecteur de fronts montants

3. Compteur

Le composant "counter" va permettre de compter les fronts montants générés par le composant "edge_detector" toutes les secondes.

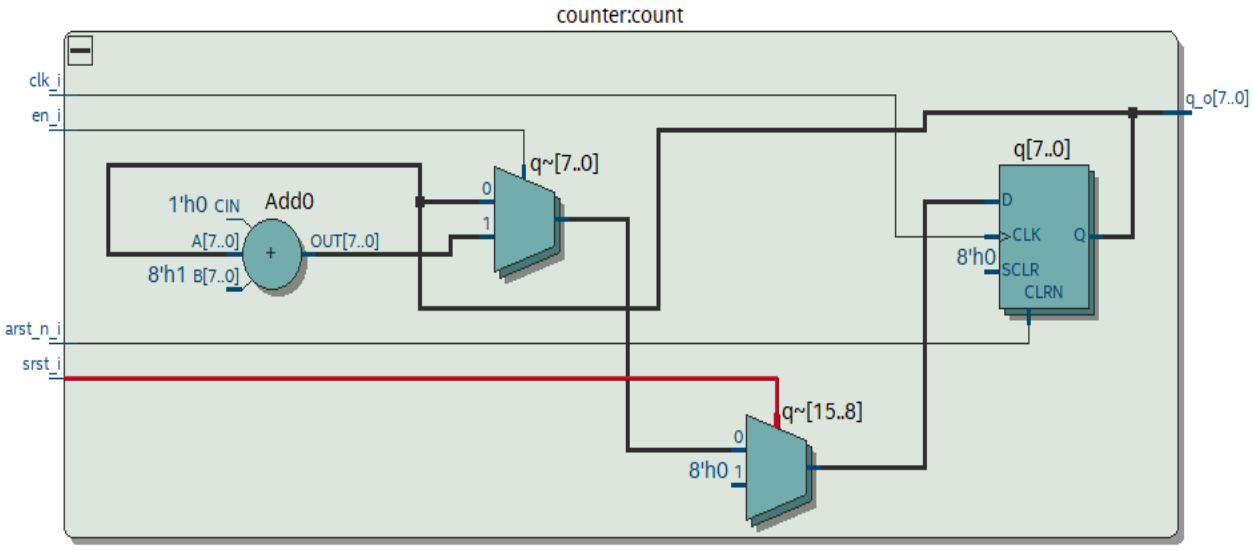


Schéma fonctionnel du compteur

4. Registre

Le composant "Registre" va permettre de stocker la valeur de la fréquence convertie dans un registre ce qui va permettre de venir traiter ou effectuer des calculs une fois celle-ci acquise.

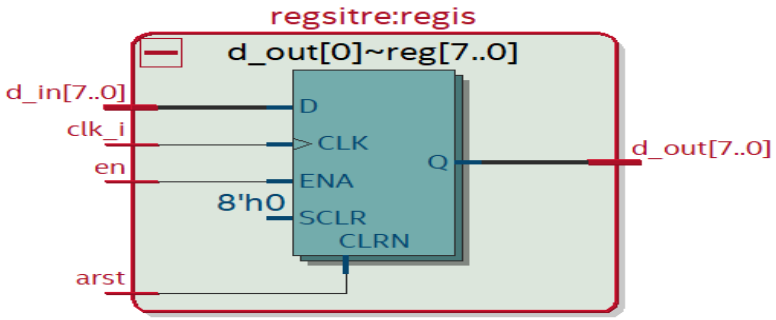
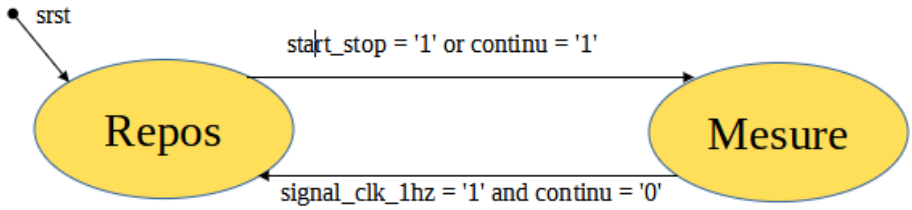


Schéma fonctionnel du registre

5. Machine à états (non implémenté dans le fonctionnement)

Le composant "MEF" est chargé de gérer le mode de fonctionnement du circuit, ce qui permet à ce bloc fonctionnel de savoir quand il doit être remis à zéro "Start_Stop" ou en mode "Continu" ou "Monocoup".
Malheureusement, cette partie n'a pas été implémenté dans le schéma fonctionnel de l'anémomètre, alors l'acquisition de notre anémomètre ici sera toujours en continu.



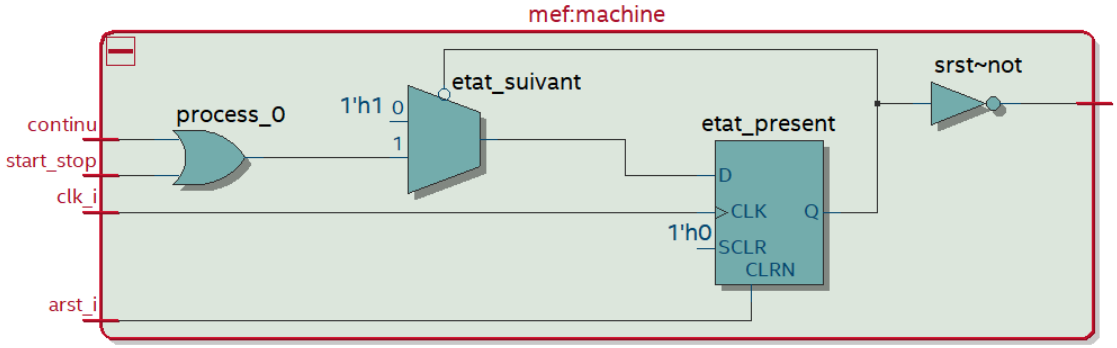
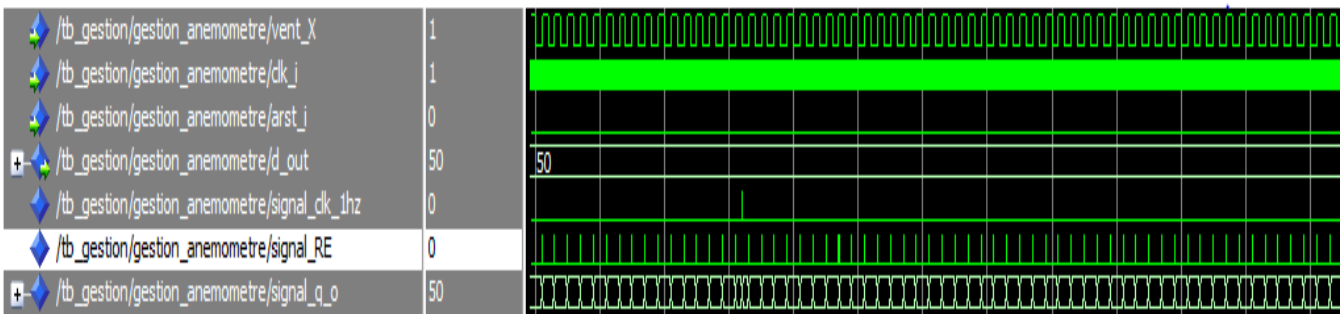


Schéma fonctionnel de la machine à états

Remarque : Au projet, une partie interface Avalon doit être ajoutée qui va permettre l'implémentation et l'interfaçage des fonctions entre la partie NIOS II et la partie matérielle FPGA, mais nous n'avons pas pu arriver jusqu'au bout et on a effectué la simulation sur Modelsim à partir de la génération d'un banc de test.

6. Simulation et validation de la fonction anémomètre



Simulation de la gestion anémomètre

D'après la simulation ci-dessus, on remarque bien que la fonction "edge_detector" qui prend en entrée le signal de l'horloge et le signal du vent et en sortie on arrive à détecter chaque front montant de notre signal vent (vent_X). Le comptage des fronts montant sera à travers le compteur "counter" et qui va être réinitialiser quand signal_clk_1hz = '1' et activer le registre, donc à chaque 1s on aura un signal de sortie (signal_q_o) qui va aller vers le registre pour enregistrer la valeur et ce dernier donnera la valeur calculée (d_out).

V. Conception d'une fonction complexe : Gestion Vérin

Afin, de mettre en mouvement la barre franche, un circuit vérin y est ajouté. Cette partie du projet consiste à réaliser la fonction de gestion de ce vérin. Pour ce faire nous devons développer quatre fonctions principales qui ensemble permettront le pilotage du vérin qui contrôle la barre franche :

- La fonction Gestion_PWM
- La fonction Gestion_butee
- La fonction Gestion_MCP3201
- La fonction d'interfaçage du bus Avalon

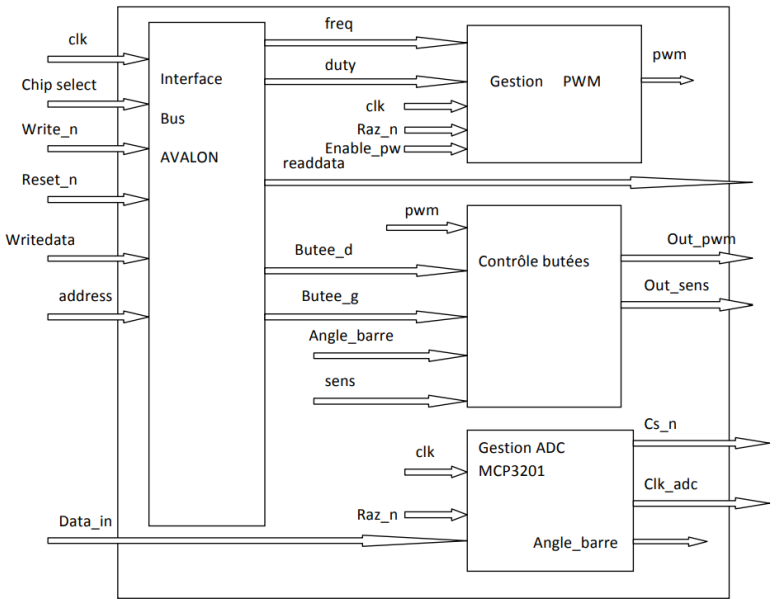


Schéma fonctionnel du vérin

1. Gestion_PWM

Le but de cette fonction est de mouvoir le vérin. Son fonctionnement se décompose comme suit :

On a un comparateur qui compare la valeur de du compteur et de l'entrée Duty. Tant que count est inférieur à Duty en sorti on un signal de pwm='1'. Lorsque cette condition n'est plus vérifiée, pwm passe à zéro. Un second comparateur est chargé d'envoyer un signal de réinitialisation lorsque count est égale au signal « freq ».

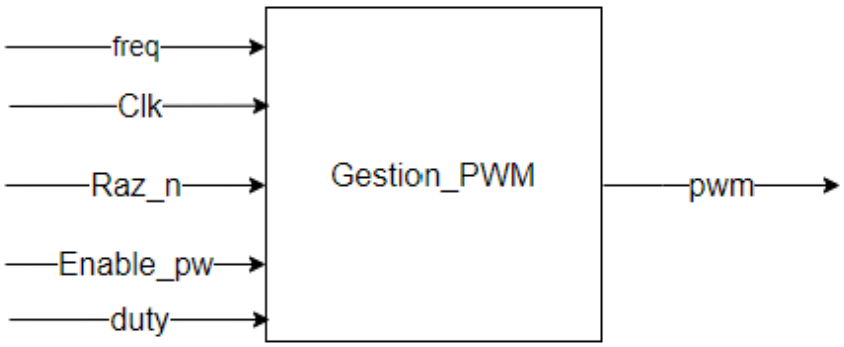
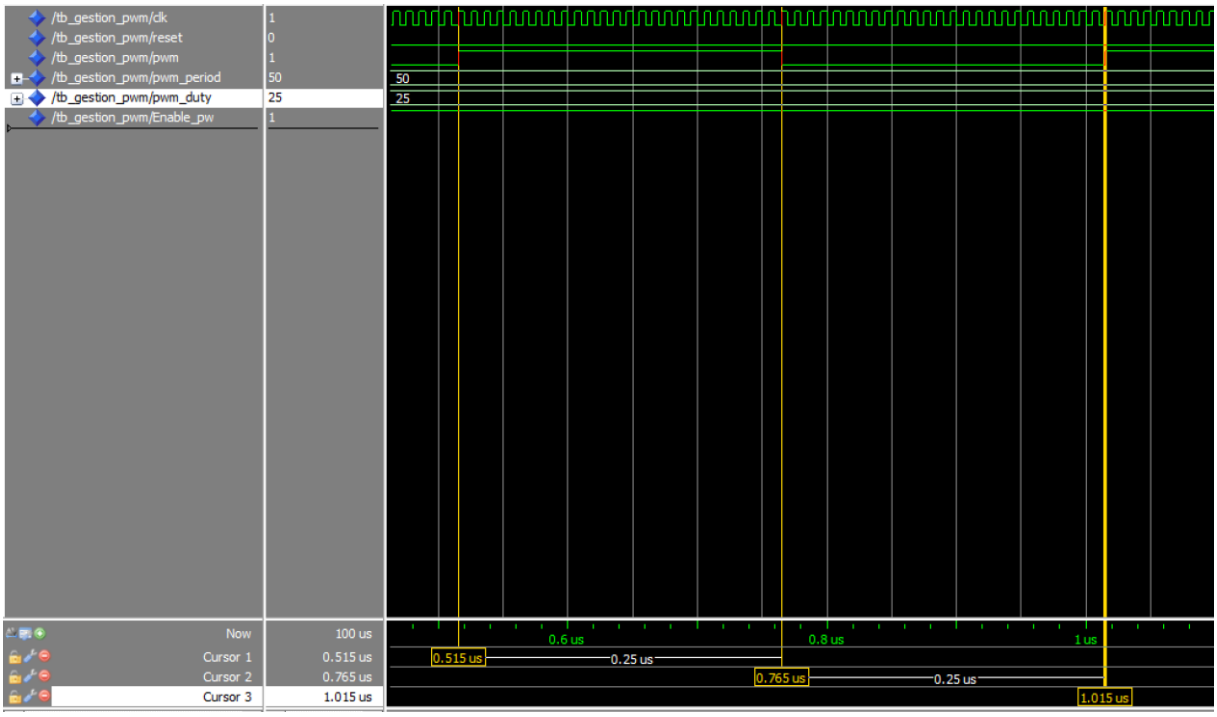


Schéma fonctionnel du PWM

La simulation ci-dessous, démontre le bon fonctionnement de cette fonction. On constate que pour un rapport cyclique de 0.5, les temps à l'état haut et bas du signal « pwm » sont strictement égaux.



Simulation de la gestion du PWM

2. Gestion_butee

Afin d'assuré la longévité du vérin et d'évité toute casse, il est nécessaire d'avoir des conditions d'arrêt du moteur du vérin. Pour ce faire, la gestion de la butée détecte en fonction des conditions de butée configuré, si le vérin est en fin de course. Si c'est le cas, il se charge de mettre le moteur à l'arrêt. Pour cette opération, une comparaison est effectuée entre la valeur de l'angle barrée et celle des condition de butée gauche ou droite en fonction du sens de déplacement du vérin. Ensuite une action est déclenchée sur la sortie pwm en fonction de l'état actuel du moteur.

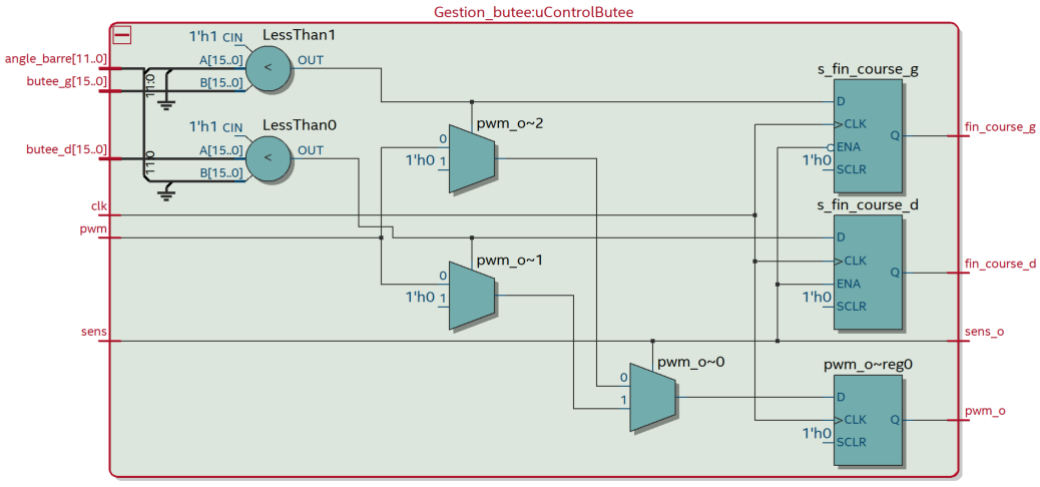
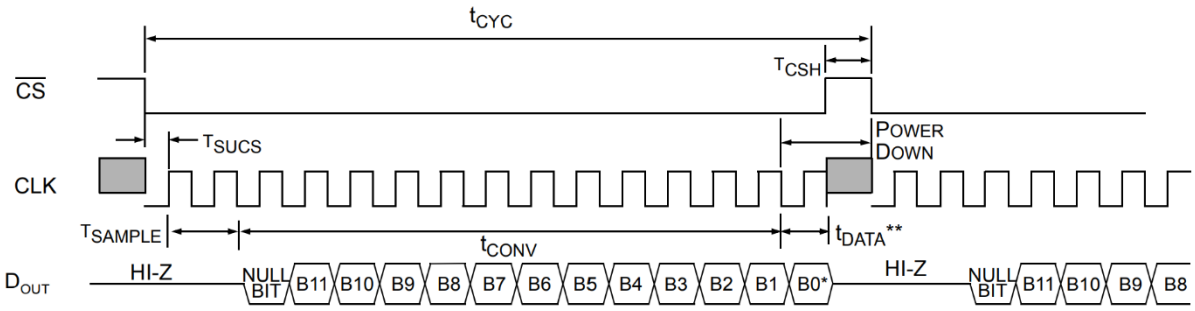


Schéma fonctionnel de la Gestion_butee

3. Gestion_MCP3201

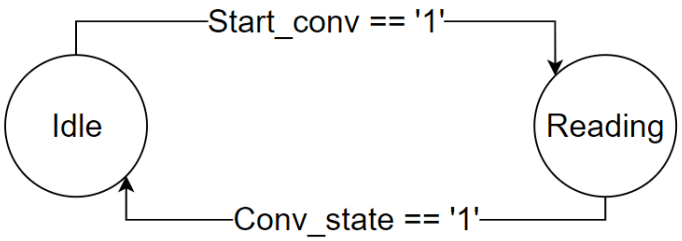
L'angle barrée est une grandeur analogique, pour la lire, il est nécessaire de joindre à notre système un convertisseur analogique numérique. Pour notre système le choix s'est porté sur l'ADC MCP3201. Il s'agit d'un convertisseur 12 Bits avec une interface SPI série. Il est donc nécessaire d'implémenter une interface de communication SPI sur notre composant vérin.

Ce pilote est basé sur une machine à état qui est chargé de générer une horloge chaque 100ms afin de faire l'acquisition des données dans le respect des spécifications de la datasheet du MCP3201.



Protocole de communication du MCP3201

Le pilotage de la communication avec l'ADC, dépend des signaux « Start_conv » pour passer de l'état « IDLE » à l'état « Reading » (lecture des données de l'ADC) et du signal « Conv_state » (Cs_n) qui signale la fin de la conversion et le passage de l'état « Reading » à l'état « IDLE ».



MEF pilotage ADC

La fonction Gestion_MCP3201 sert, outre piloter la communication avec le convertisseur analogique numérique, à enregistrer la valeur de l'angle barrée transmis à la Gestion_butee. L'entrée « Data_in » correspond aux valeurs envoyées par l'ADC.

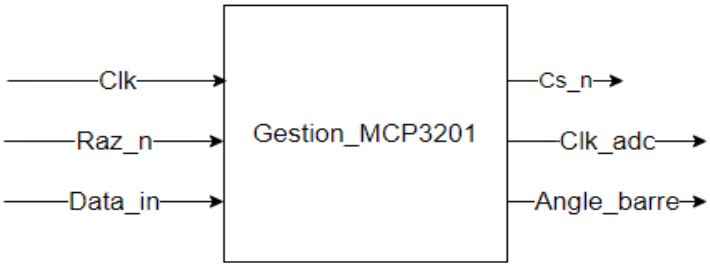
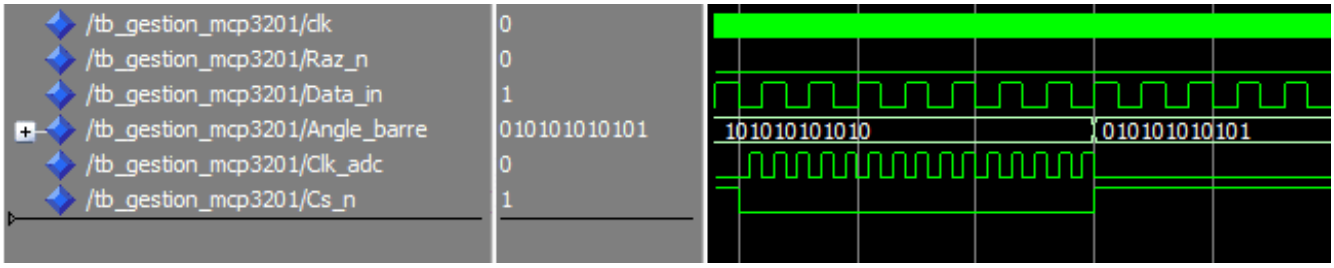


Schéma fonctionnel de la Gestion_MCP3201

Après développement de la fonction, nous avons eu le même comportement que spécifié dans la datasheet pour la communication avec l'ADC.

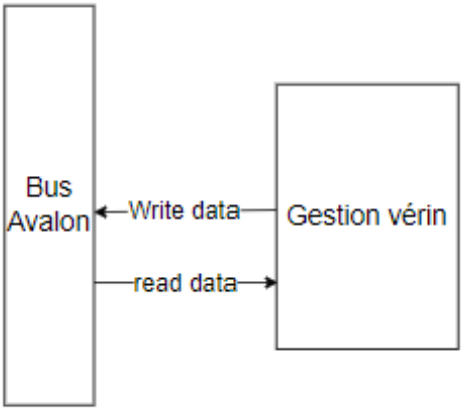


Simulation de la gestion de la Gestion_MCP3201

4. Réalisation de l'interface Avalon

La création d'une interface processeur et matériel programmable est nécessaire au bon fonctionnement du projet. Altéra a développé un bus informatique appelé bus "Avalon" destiné à l'implémentation matériel/composants sur FPGA. Il va permettre l'interconnexion entre le processeur NIOS II (réalisé à l'aide de SOPC Builder) et les périphériques des différents composants que nous avons créés précédemment.

Pour réaliser cette interface il a fallu respecter le cahier des charges donné en début de projet pour chaque interface, qui correspondent à des circuits combinatoires/séquentiels utilisés pour l'écriture et la lecture du bus (Address, read, write, chip select...). Ci-dessous est représenté le circuit que nous avons implémenté au projet pour le gros bloc fonctionnel du vérin.



Interface Avalon-Gestion_verin

VI. Conclusion :

Ce bureau d'études était pour nous l'occasion de voir l'utilité des FPGA, l'intégration des systèmes, l'implémentation du code en utilisant VHDL.

Même si les différentes contraintes qu'on a eu durant ce projet, et qui nous ont empêché de finir notre but, on a pu implémenter des fonctions, on a commencé sans quasiment rien savoir sur le vhdl et la synthèse, et au bout de ces heures dédiées à ce projet on arrive à acquérir des compétences en programmation en VHDL, simulation des architectures avec des test Bench.

Nous tenons donc à remercier le professeur pour son encadrement durant toute
la durée du projet, et pour son aide afin de développer nos compétences dans le monde des circuits programmables, l’architecture logicielle et l’ingénierie système.