Université Paul Sabatier

Rapport VHDL

Barre franche

Auteurs :
Gautier Jobert
Alexandre regnere
Encadrant :
Thierry PERISSE

Table des matières

1 Introduction

A travers ce projet, nous allons apprendre a developper sur carte FPGA avec le langage VHDL au travers de l'asservissement d'une barre franche. Le developpement se fera coté hardware mais également en partie software avec le bus Avalon.
Le contexte de l'asservissement d'une barre franche nous permettra de traiter des données physiques (ex : force du vent) ainsi que la direction d'un cap.

Vous pourrez retrouvez notre projet avec le code et toutes les informations complémentaires au lien suivant :

`https://github.com/GautierDev31/M2SME_VHDL_GR4`

2 Présentation du projet

2.1 Cahier des charges

Parmi toutes les fonctions de cette barre franche, nous allons réaliser deux fonction :
- Une fonction simple F1 : Conversion info vent ou Gestion anémomètre
- Une fonction complexe F7 : Gestion commandes et indications barreur

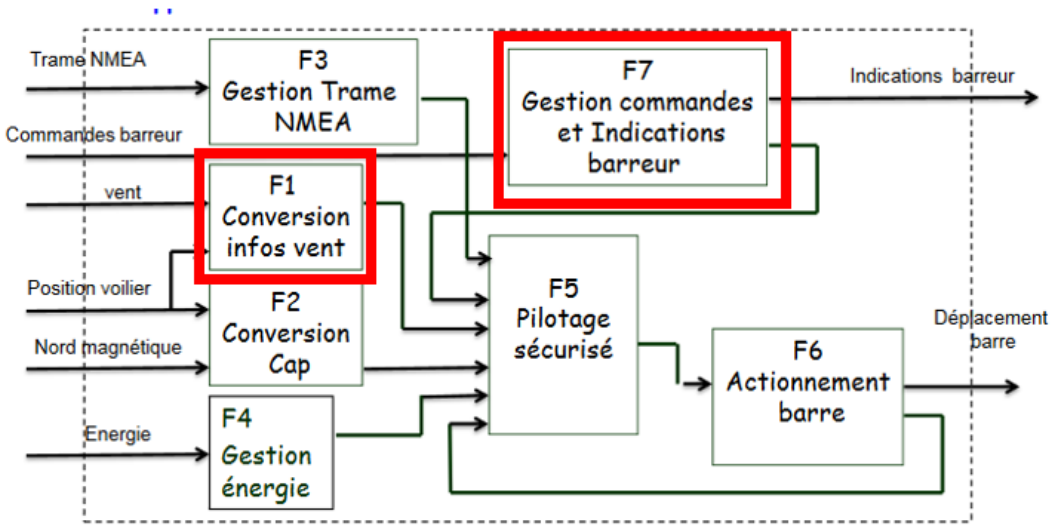

Schéma global avec les fonctions réalisées

2.2 Matériel et outils

Voici les outils que nous avons utilisés :
Carte FPGA DEO : Carte sur laquelle nous avons travaillé.
Quartus 9 et 11 : Developpement du code VHDL et assignation des pin pour le FPGA. Quastus 9 nous a permis dans un premier temps de simuler le code, nous avons ensuite utilisé Quartus 11 avec ModelSim pour les simulations.
Nios II et Eclipse : Developpement de la partie Software en language C qui utilise le bus avalon.

2.3 Partie software - Bus avalon

Le Bus Avalon est un bus qui a été developpé par Altera afin de permettre l'integration entre différentes parties. Il va permettre l'interconnexion entre le processeur NIOS II et les différents périphériques des composants correspondant a chaques fonctions.

Lors de ce BE, le bus Avalon va nous permettre principalement d'afficher sur la console via le bus série les informations lues par les fonctions comme la vitesse du vent pour la fonction anemometre et les condes des différentes actions pour la fonction commande barreur.

2.4 Organisation du BE

Pour travailler a 2 sur le même code, nous avons travaillé en utilisant l'outil Git en hebergeant nos fichiers sur GitHub.Cette plateforme nous permet d'heberger le code et de le rendre accessible en sauvegardant tout les changements effectués.
Pour avoir des hebergements de travail différent et ne pas introduire des bug lorsque l'on fais des test, nous avons créé différentes branches. Par exemple lorsque l'on travaillais sur le SOPC nous faisions un "Push" (envoie du code) vers la branche "SOPC" et lorsque nous travaillons sur la fonction F1 nous faisiont un Push vers la branche "Dev F1".
Lorsque le travaille est terminé et testé, nous avont fait un "merge" vers la branche principale, c'est a dire que nous avons assemblé les différentes branche entre elles. Le travail qui est accessible sur le GitHub est le travail sur la branche principale.

Pour le reste des communication, nous avons travaillé sur "Discord" entre nous et sur "Slack" pendant les heures de BE.

3 Réalisation

3.1 F1 - Gestion anémomètre

3.1.1 Cahier des charges

La fonction doit convertir la vitesse du vent en un signal de 8 bit ainsi qu'un signal indiquent que la valeur est disponible.
La fonction comportera 2 modes de lectures :
Mode continue :Lis la vitesse du vent en continue et l'affiche toute les secondes.
Mode Start and Stop : Commence a lire la vitesse du vent lorsque l'on appuis sur Start et affiche la vitesse du vent en moyenne en appuyant sur Stop.

Entrées :
clk 50M : hologe 50MHz
raz n : rest actif à 0 => initialise le circuit
in freq anemometre : signal de fréquence variable de 0 à250 HZ
continu : si=0 mode monocoup, si=1 mode continu
start stop : en monocoup si=1 démarre une acquisition, si =0 –remet à 0 le signal data valid

Sorties :
data valid : =1 lorsque une mesure est valide–est remis à 0 quand start
stop passe à 0
data anemometre : vitesse vent codée sur 8 bits

3.1.2 Schéma fonctionnel

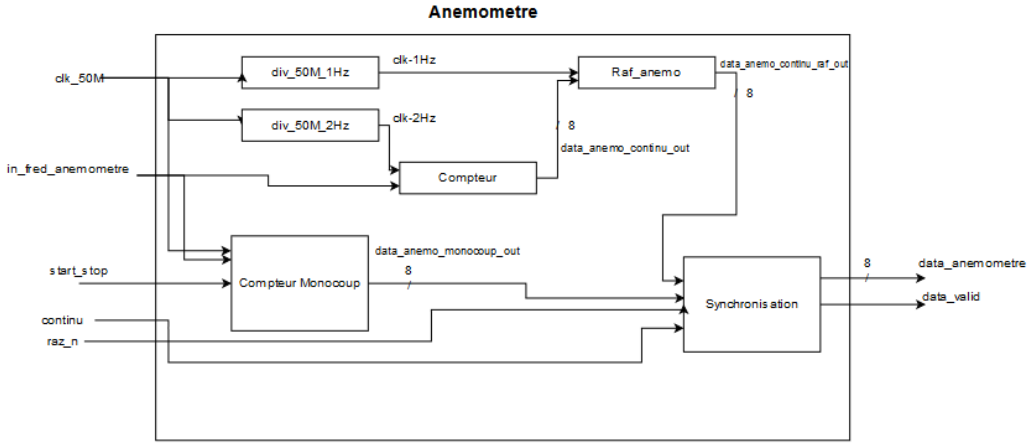

Schéma fonctionnel F1 : Gestion anémomètre

3.1.3 Simulation

3.1.4 Test sur FPGA

3.1.5 Partie software

3.2 F7 - Gestion commandes et indications barreur

3.2.1 Cahier des charges

3.2.2 Schéma fonctionnel

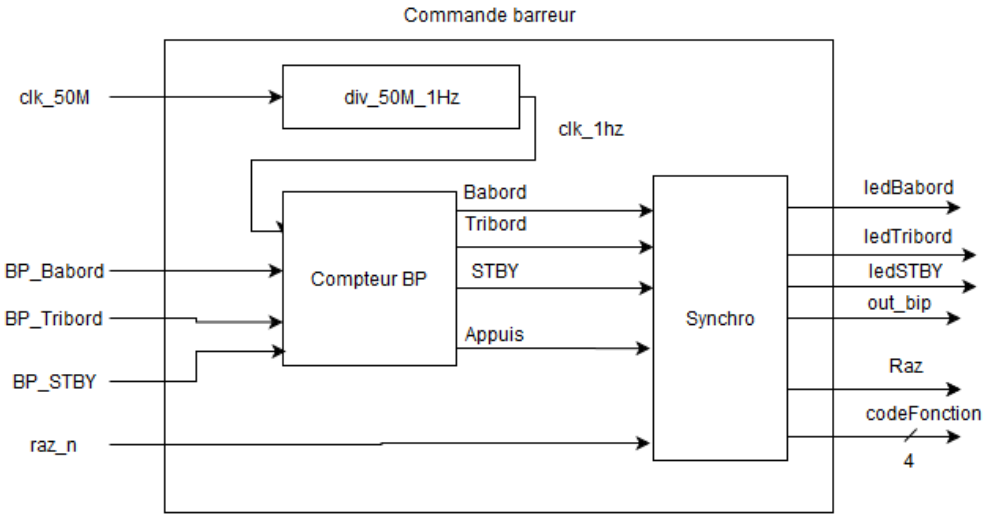

Schéma fonctionnel F7 : Gestion commandes et indications barreur

3.2.3 Simulation

3.2.4 Test sur FPGA

3.2.5 Partie software

4 Conclusion