

UNIVERSITÉ PAUL SABATIER

## Rapport VHDL

---

# BARRE FRANCHE

---

*Auteurs :*

Gautier JOBERT

Alexandre REGNERE

*Encadrant :*

Thierry PERISSE



# Table des matières

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Présentation du projet</b>	<b>2</b>
2.1	Cahier des charges . . . . .	2
2.2	Matériel et outils . . . . .	3
2.3	Partie software - Bus avalon . . . . .	3
2.4	Organisation du BE . . . . .	4
<b>3</b>	<b>Réalisation</b>	<b>5</b>
3.1	F1 - Gestion anémomètre . . . . .	5
3.1.1	Cahier des charges . . . . .	5
3.1.2	Schéma fonctionnel . . . . .	6
3.1.3	Partie software . . . . .	7
3.2	F7 - Gestion commandes et indications barreur . . . . .	8
3.2.1	Cahier des charges . . . . .	8
3.2.2	Schéma fonctionnel . . . . .	9
3.2.3	Platine de tests . . . . .	9
3.2.4	Partie software . . . . .	10
<b>4</b>	<b>Conclusion</b>	<b>11</b>

# 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 retrouver notre projet avec le code et toutes les informations complémentaires au lien suivant :

[https://github.com/GautierDev31/M2SME\\_VHDL\\_GR4](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

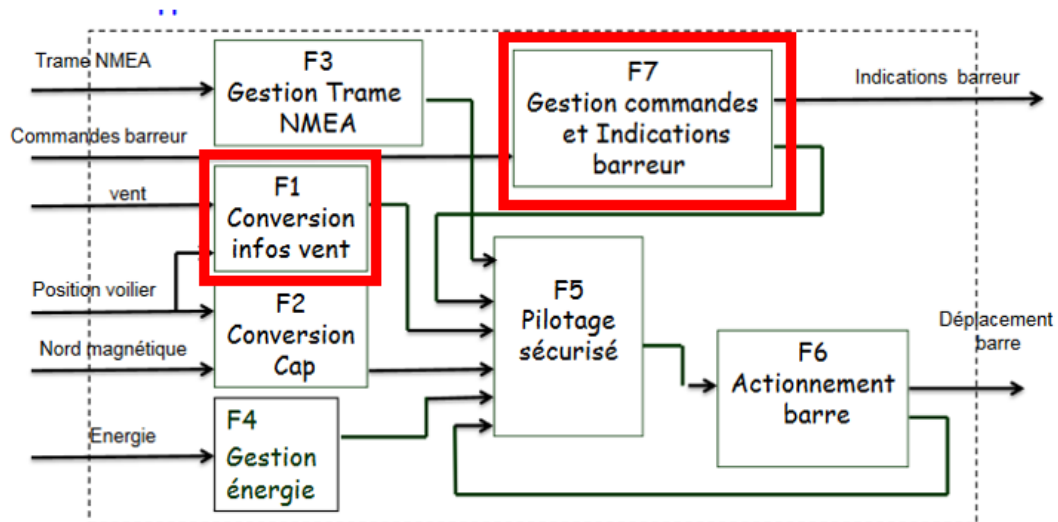


Schéma global avec les fonctions réalisées

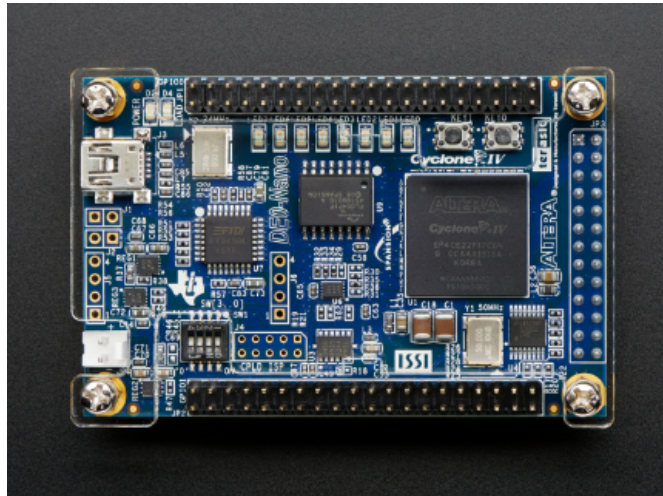
## 2.2 Matériel et outils

Voici les outils que nous avons utilisés :

Carte FPGA DEO : Carte sur laquelle nous avons travaillé.

Quartus 9 et 11 : Développement du code VHDL et assignation des pin pour le FPGA. Quartus 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 : Développement de la partie Software en langage C qui utilise le bus avalon.



Carte Altera DEO

## 2.3 Partie software - Bus avalon

Le Bus Avalon est un bus qui a été développé par Altera afin de permettre l'intégration entre différentes parties. Il va permettre l'interconnexion entre le processeur NIOS II et les différents périphériques des composants correspondant à chaque fonction.

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 condés 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 hébergeant nos fichiers sur GitHub. Cette plateforme nous permet d'héberger le code et de le rendre accessible en sauvegardant tout les changements effectués.

Pour avoir des hébergements 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 faisons un "Push" (envoi du code) vers la branche "SOPC" et lorsque nous travaillons sur la fonction F1 nous faisons un Push vers la branche "Dev F1".

Lorsque le travail est terminé et testé, nous avons 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.

Pour pouvoir s'y retrouver dans les fichiers sur GitHub, Voici l'arborescence :

Barre franche : Contient les codes VHDL des composants et des fonctions

Barre franche > Components : Contient les différenst codes des composants utilisés dans les fonctions

Barre franche > F1 et F7 : Contient le code des deux fonctions qui est un mappage entre les différents composants

PWM : Code d'un component pour la fonction F1

SOPC : Code SOPC des fonctions F1 et F7

TP Base : Regroupe les codes pour les TB de bases

images : Images utilisés pour les descriptions

Rapport et présentation : Contient le rapport, la présentation et la video

## 3 Réalisation

### 3.1 F1 - Gestion anémomètre

#### 3.1.1 Cahier des charges

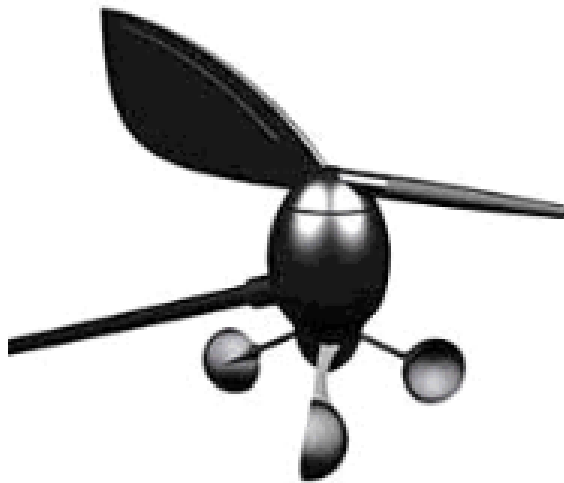
La fonction doit convertir la vitesse du vent en un signal de 8 bit ainsi qu'un signal indiquent que la valeur est disponible.

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: reset actif à 0 => initialise le circuit
--in_pwm_compas: signal PWM de la boussole, durée varie de 1ms à 36,9ms
--continu : si=0 mode monocoup, si=1 mode continu
-- en mode continu la donnée est rafraîchie toute les secondes
--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
-- out_1s : signal de contrôle du top seconde (normalement pas utilisé)
-- data_compas : valeur du cap réel exprimé en degré codé sur 9 bits
__*****
```



Anemometre



### 3.1.2 Schéma fonctionnel

Afin de décomposer la fonction en différentes parties, nous avons fait un schéma fonctionnel.

Diviseur de fréquence : L'horloge du FPGA est de 50 MHz, pour la ramener a une echelle de temps exploitable, nous avons fait deux diviseurs de fréquences, à 2Hz et 1Hz.

Compteur : Le compteur va permettre de savoir combien de temps l'entrée est a l'état haut. En sortie il indiquera ce temps sur 8 bits.

Raf anemo : Cette fonction permet de rafraichir la valeur de temps haut reçu toute les secondes.

Compteur Monocoup : Ce composant permet d'activer le comptage de l'état haut lorsque le bouton start est activé.

Synchro anemo : Cette fonction permet de renvoyer l'information de la valeur de l'anemometre en prenant la bonne valeur entre le monocoup et le mode continu. Il renvois également le data valid pour savoir si la valeur est bien valide.

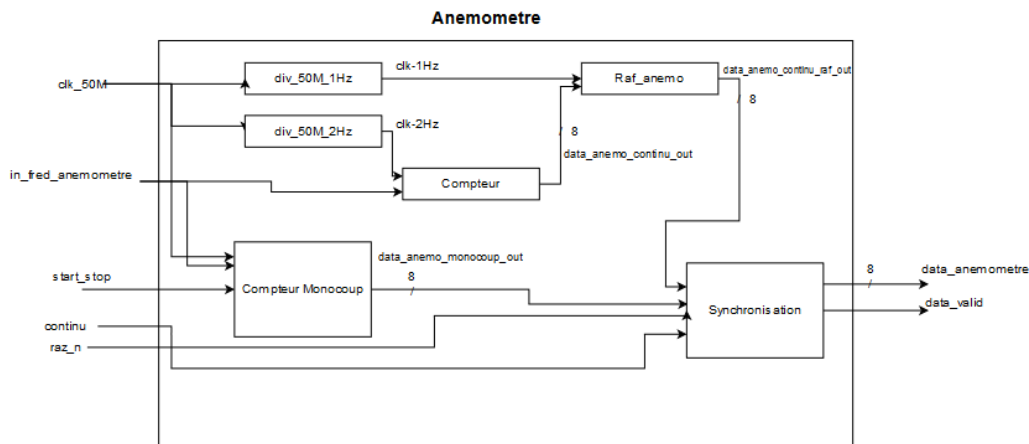
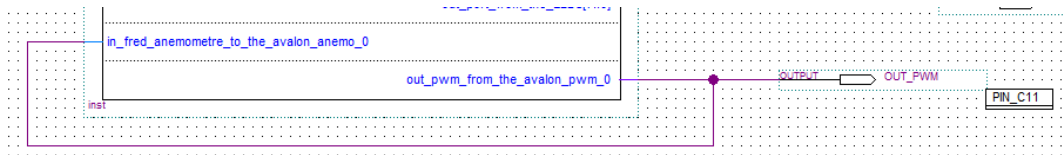


Schéma fonctionnel F1 : Gestion anémomètre

### 3.1.3 Partie software

Afin de pouvoir tester la partie software avec une seule platine, nous allons générer un signal PWM en sortie du FPGA que nous allons branché sur une entrée pour pouvoir la lire. Le câblage sera fait virtuellement, on peut voir sur le fichier .bdf ci-dessous la sortie relié a l'entrée.

Pour le code en C, nous avons définis dans la première partie la fréquence et le rapport cyclique qui allais être envoyé. Dans la seconde partie on affiche la valeur lue.



Anemometre BDF

Du coté software, en langage C, nous avons définis

```
*freq = 0x2625A0 ; // CLK divisee par 2 500 000 => freq = 20 Hz
*duty = 0x1312D0 ; //10% de 2 500 000 => duty = 10%
*control = 0x3 ;

*config = 0x2 ;
```

Code C Anemometre : Declaration des variables

```
/* ***** DATA ANEMOMETRE ***** */
/* ***** */
*data = *data&0x3FF ;
printf("reg = 0x%08X\n", *data) ;
alt_printf("config = 0x%x\n", *config) ;
```

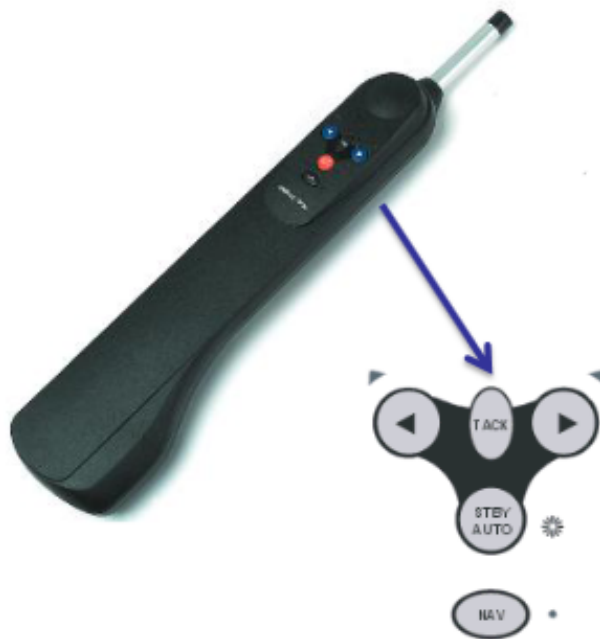
Code C Anemometre : Affichage de la valeur de l'anemometre

## 3.2 F7 - Gestion commandes et indications barreur

### 3.2.1 Cahier des charges

```
__*****
-- entrées: BP_Babord,BP_Tribord, BP_STBY, clk_50M, raz_n
-- sorties: codeFonction, ledBabord, ledTribord,ledSTBY, out_bip
__*****

--clk_50M: horloge à 50MHz
-- raz_n: actif à 0 => initialise le circuit
-- valeurs de codeFonction:
-- =0000: pas d'action, le pilote est en veille
-- =0001: mode manuel action vérin babord
-- =0010: mode manuel action vérin tribord
-- =0011: mode pilote automatique/cap
-- =0100: incrément de 1° consigne de cap
-- =0101: incrément de 10° consigne de cap
-- =0111: décrétement de 1° consigne de cap
-- =0110: décrétement de 10° consigne de cap
__*****
```



Commandes barreur

### 3.2.2 Schéma fonctionnel

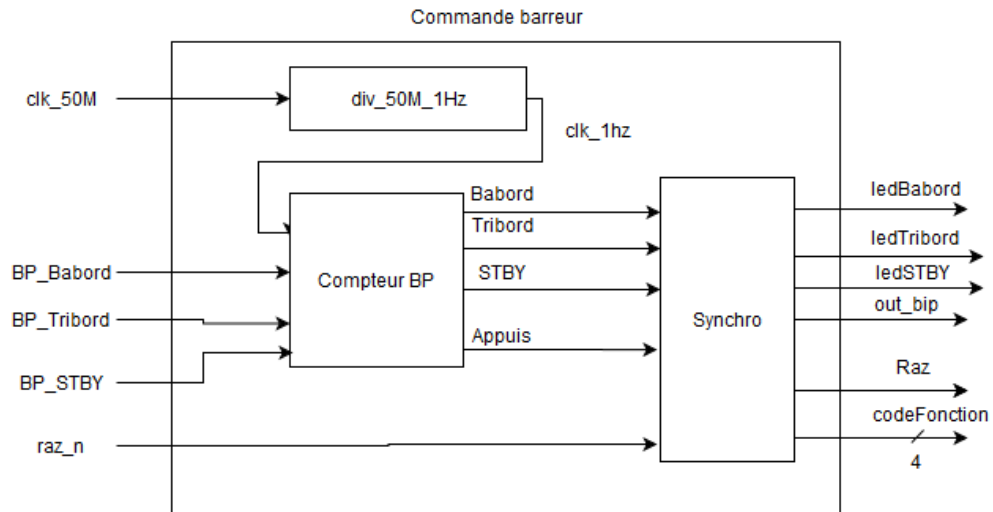
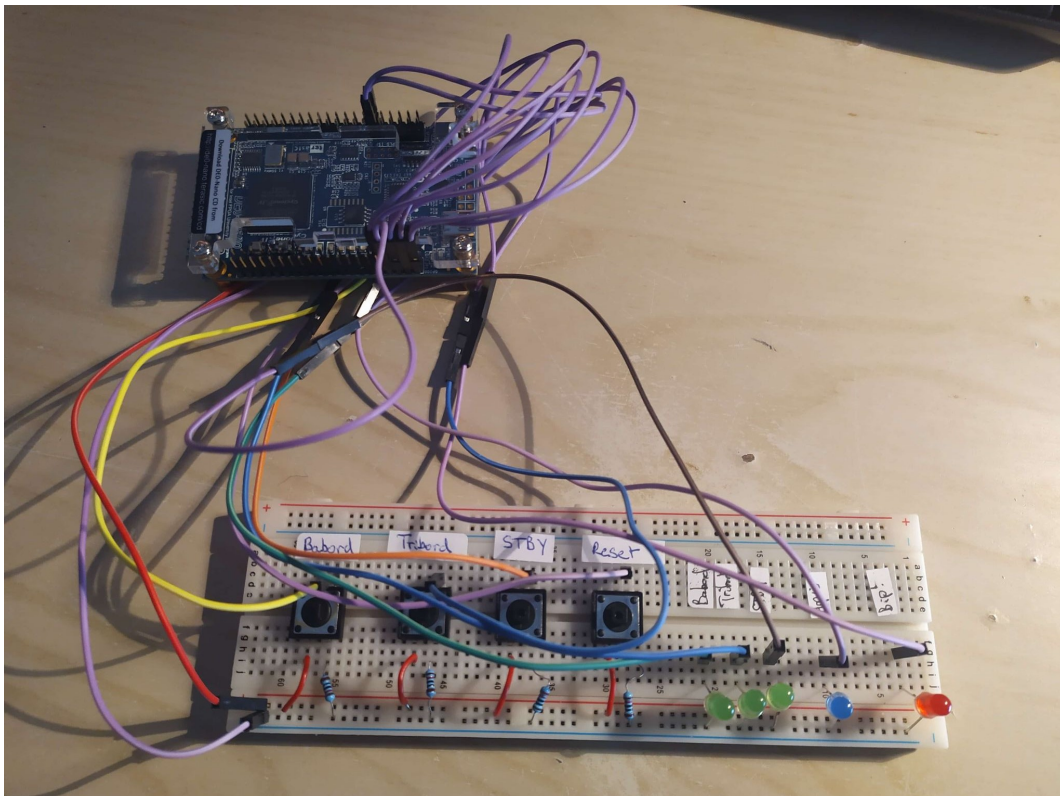


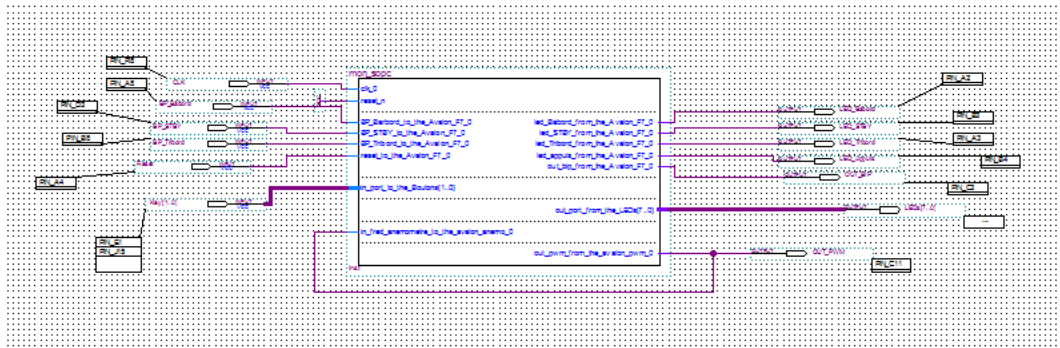
Schéma fonctionnel F7 : Gestion commandes et indications barreur

### 3.2.3 Platine de tests



Platine de test

### 3.2.4 Partie software



## Commandes barreur BDF

## 4 Conclusion

Pour conclure, on peut dire que grâce à ce projet nous avons acquis un large spectre de compétences de bases pour le développement en VHDL.

Le langage VHDL étant un langage de description machine, il ne se code pas de la même façon que les autres langages appris durant notre formation. Ce projet nous aura donc permis d'avoir de l'expérience dans le code de langage de description.

Nous avons aussi développer nos compétence en conception en réfléchissants aux différents composants pour réaliser une fonction avec des schémas fonctionnels. Ces composants auront été câblé par la suite en faisant un MAP en VHDL.

Grâce au bus Avalon développé par Altera nous avons appris à faire une partie software en faisant communiquer notre PC avec le FPGA.

Enfin, une compétence importante que nous avons expérimenté avec ce projet à été de gerer notre projet avec Git, ce qui deviens un compétence indispensable aujourd'hui pour le travail en entreprise.