M2 EEA SME

Projet Barre Franche - Novembre 2019

EIEAS3GM
SYNTHESE ET MISE EN ŒUVRE DES SYSTEMES

Auteurs :
Nicolas OTAL
Antoine ROUTIER

Encadrants :
M. PERISSE

Introduction

Dans le cadre de notre UE "Synthèse et Mise en œuvre de système", un bureau d'étude nous a été proposé avec pour objectif de mettre en œuvre une solution logicielle/matérielle qui répond au besoin de gestion et contrôle de trajectoire d'un voilier de barre franche.

A l'aide des bases acquises, durant notre formation, en VHDL, simulation, et conception systèmes l'objectif de cette UE sera la réalisation d'un contrôleur de Barre-France de voilier par FPGA (Altera). Il sera nécessaire d'étudier, décomposer, coder et implémenter toutes les fonctions une à une et d'intégrer la globalité du projet à l'aide d'un Bus Avalon permettant l'interconnexion des fonctions au MCU intégré au FPGA.

Pour réaliser cela, nous avons respecté le processus de développement consistant à réaliser une analyse des besoins et du contexte permettant d'identifier les différentes interfaces du système, nous avons par la suite réalisé la conception du système en décomposant notre fonction principale en différents blocs. Ces différents blocs ont fait l'objet d'une description fonctionnelle avant d'être implémenté. Suite à cela, nous avons réalisé un ensemble de simulation et test sur maquette pour vérifier le bon fonctionnement de chaque module intégré pour valider finalement notre projet sur une maquette.

Sigles et acronymes

FPGA	*Field Programmable Gate Arrays*
GPIO	*General Purpose Input/Output*
GPS	*Global Positioning System*
NMEA	*National Marine Electronics Association*
PWM	*Pulse Width Modulation*
SOPC	*System On Programmable Chip*
SRAM	*Static Random Access Memory*
UART	*Universal Asynchronous Receiver Transmitter*
VHDL	*VHSIC Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuits*

Table des matières

1 Cahier des charges du Projet "Barre-Franche"

1.1 Introduction technique

Le projet qui nous est demandé est basé sur les SoC et plus particulièrement sur un FPGA, de chez Altera, embarqué dans un voilier pour en piloter la barre-franche.
Ce dispositif électronique fonctionnera à l'aide de différentes entrées/sorties (gyroscope, anémomètre, GPS, convertisseur analogique/numérique, vérin, boutons, buzzer).

1.2 Cahier des charges général

Durant ce projet, nous allons utiliser les différentes compétences acquises à travers les différents cours de l'année et les mettre en corrélation pour mener à bien ce dernier. Le projet devra respecter certains critères présentés ci-dessous :

1. Le dispositif devra utiliser un appareil de mesure pour capter la valeur de la vitesse du vent.
2. Le dispositif devra utiliser un appareil de mesure pour capter la direction du vent.
3. Le dispositif devra réceptionner des données GPS, traiter ces données et agir sur le dispositif en fonction des résultats obtenus après traitement.
4. Le dispositif devra avoir une interface entre l'opérateur et le voilier (boutons, Leds et buzzer).
5. Le dispositif devra piloter la barre-franche du voilier à l'aide des différents appareils pour venir piloter un vérin (Vérin et compas).

La figure 1 modélise le système qui sera réalisé avec ses différentes fonctions et les différents flux d'informations nécessaires au système "Barre-Franche".

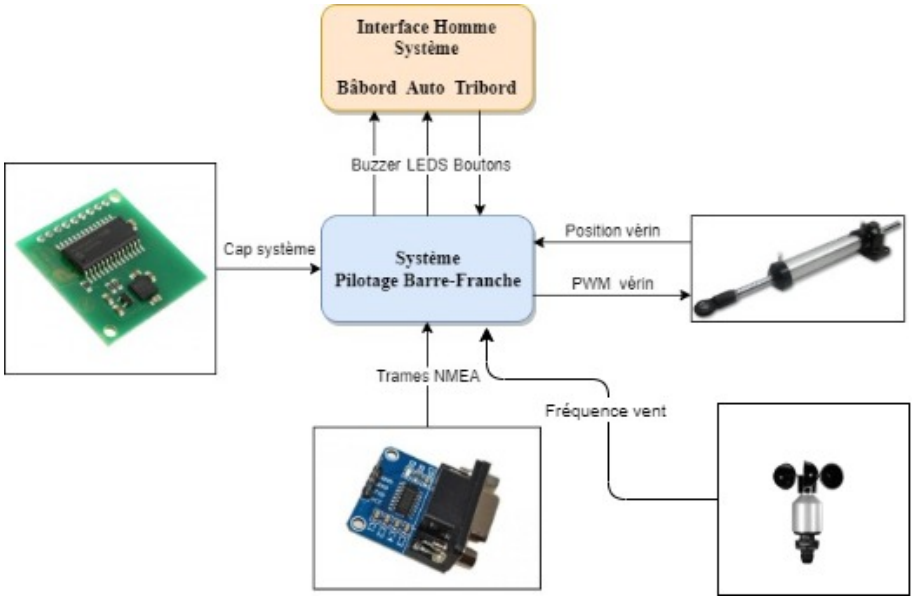


FIGURE 1 – Diagramme contexte du système

Pour des raisons pratiques et de temps nous avons implémenté la mesure de la vitesse du vent, la gestion du vérin, la gestion du compas ainsi que l'asservissement du vérin, nous n'avons pas implémenté la gestion des trames NMEA, mais avons ajouté l'interface hommes système (boutons, leds et buzzer).

1.3 Cahier des charges technique

Le projet se porte sur la réalisation d'un dispositif embarqué à base de FPGA, de chez Altera. Il sera composé de deux parties principales une partie Hardware sur le FPGA et une partie Software intégrée/développée dans le FPGA (SOPC). Les deux parties communiqueront par le biais du Bus Avalon, qui est le Bus développé par Altera pour leur SOC. Le cœur du projet sera composé d'un FPGA Cyclone IV EP4CE22F17C6N du fondeur Altera permettant le développement du projet Hardware et Software sur la même carte d'évaluation. Différentes fonctions seront implémentées :

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

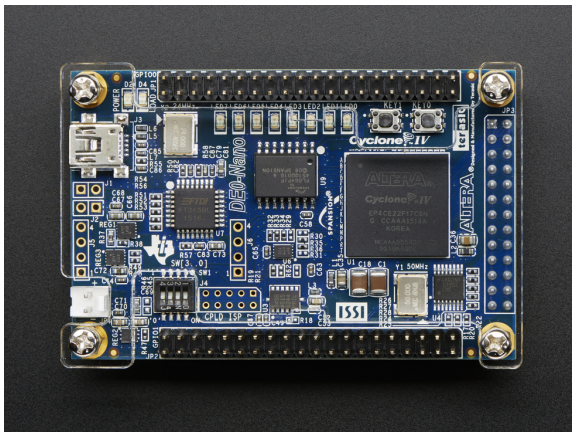


FIGURE 2 – DE0 Nano Altera

2 Conception Matérielle

Pour mettre en oeuvre les différentes parties du projet combinatoires et séquentielles, nous avons du passer par des étapes d'analyses comportementales de chaque blocs à réaliser pour ainsi créer des composants simples pouvant être utilisés à la composition de fonctions plus complexes plus tard. L'implémentation de chaque composants a été réalisée grâce au langage VHDL sur le logiciel de développement Quartus.

Ainsi nous avons réalisé différentes "boîtes" de composants simples pour réaliser des fonctions entières, et plus complexes, contenant les blocs simples.

2.1 Réalisation fonction simple - Gestion 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 de transducteur convertissant la vitesse du vent en fréquence variable (0 à 250 Hz).

Le composant doit fonctionner en deux modes :

1. Mode continu, dans ce mode le système actualise l'acquisition toutes les secondes de la vitesse du vent. Pour cela l'entrée "Continu" doit être à 1.
2. Mode mono-coup, dans ce mode le système effectue qu'une seule acquisition lorsque "Start/Stop" est activé. Une fois "data_valide" envoyé le système remet les entrées "Start/Stop" et "data_valide" à zéro après une acquisition et attend la prochaine entrée "Start/Stop".

La figure 3 représente la décomposition des composants utilisés pour la réalisation du bloc "Captage force vent".

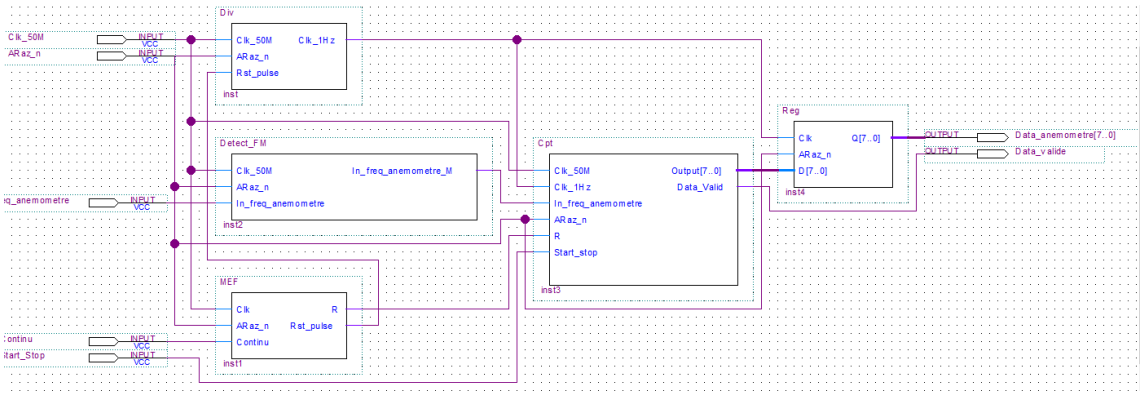


FIGURE 3 – Diagramme fonctionnel du circuit "Captation vitesse vent"

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

Le deuxième composant "Detect_FM" est en charge de détecter les fronts montants sortant de l'anémomètre pour transformer la fréquence générée en Hertz en une donnée interprétable par le composant "Cpt".

Le composant "Cpt" va permettre de récupérer la quantité de fronts montants comptés par le composant "Detect_FM", toutes les secondes quand le composant sera en mode continue et une seule fois lors du mode mono-coup.

Le composant "Reg" permettant 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 (ici dans le MCU qui sera réalisé prochainement).

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 ou savoir quand il est en mode "Continu" ou "Mono-coup".

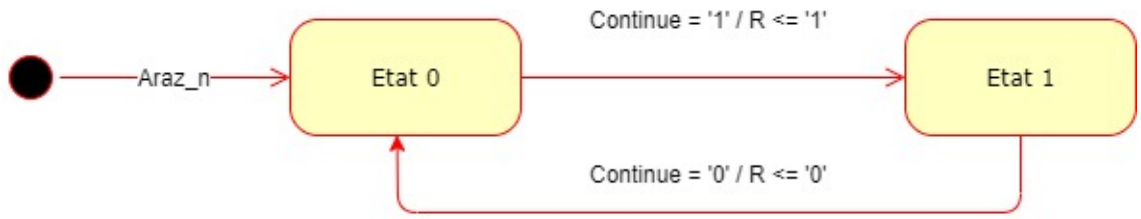


FIGURE 4 – Machine à états Gestion anémomètre

Au projet, une partie interface Avalon a été ajoutée permettant l'implémentation et l'interfaçage des fonctions entre la partie NIOS II et la partie matérielle FPGA.

2.2 Réalisation fonction complexe - Gestion vérin

La fonction réalisant le mouvement de la barre franche est réalisé avec le circuit vérin. Ce circuit est composé de 3 fonctions principales qui vont permettre le pilotage du vérin qui contrôle la barre franche du voilier :

1. Une gestion d'un signal PWM effectué par le process "PWM" qui va permettre de faire sortir ou rentrer le vérin en fonction d'une fréquence et le rapport cyclique.
2. Un contrôle des butées réalisé par le process "Gestion_Butée" qui sous certaines conditions fixées dans la partie logicielle du projet fera l'avance ou le recul du vérin jusqu'à celles-ci.
3. Une machine à états pour la gestion du convertisseur MCP3201 12 bits rendant possible le déplacement du vérin dans un sens puis dans un autre.
4. Une partie interface Avalon est ajouté au projet pour créer le lien entre la partie logicielle et la partie matérielle du FPGA/NIOS II.

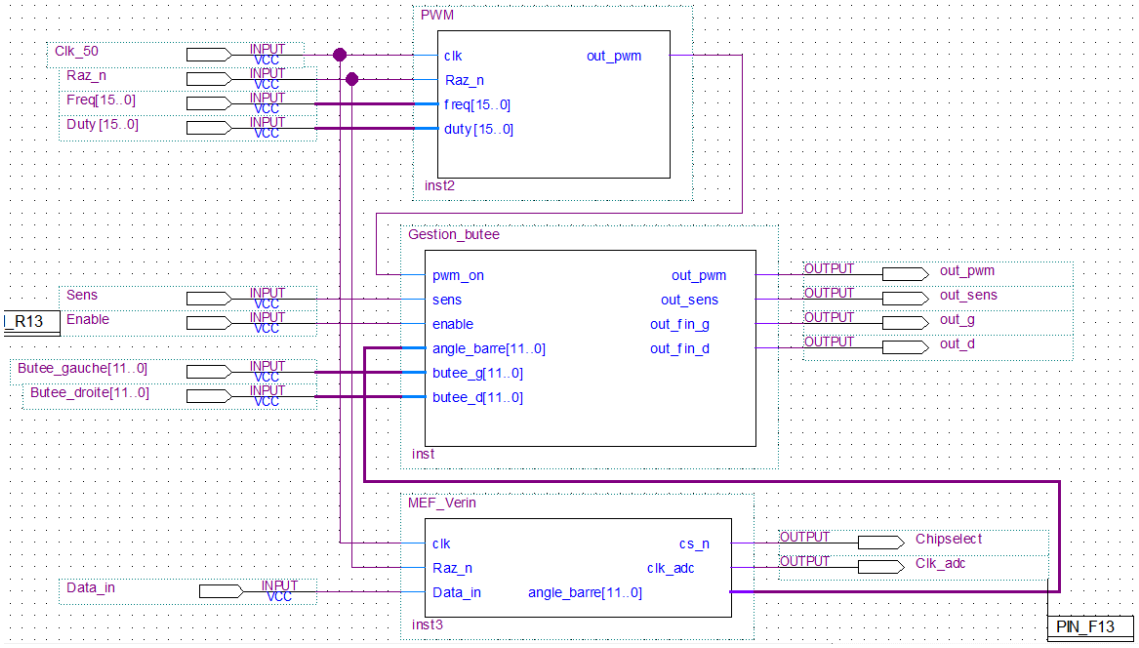


FIGURE 5 – Diagramme fonctionnel du circuit "Actionnement Barre"

La machine à états ci-dessous décrit le comportement du bloc Gestion vérin permettant ainsi de gérer le pilotage du convertisseur AN MCP3201 qui va mémoriser la donnée "angle_barre" pour ensuite l'écrire sur le bus du NIOS II puis dans des régistres.

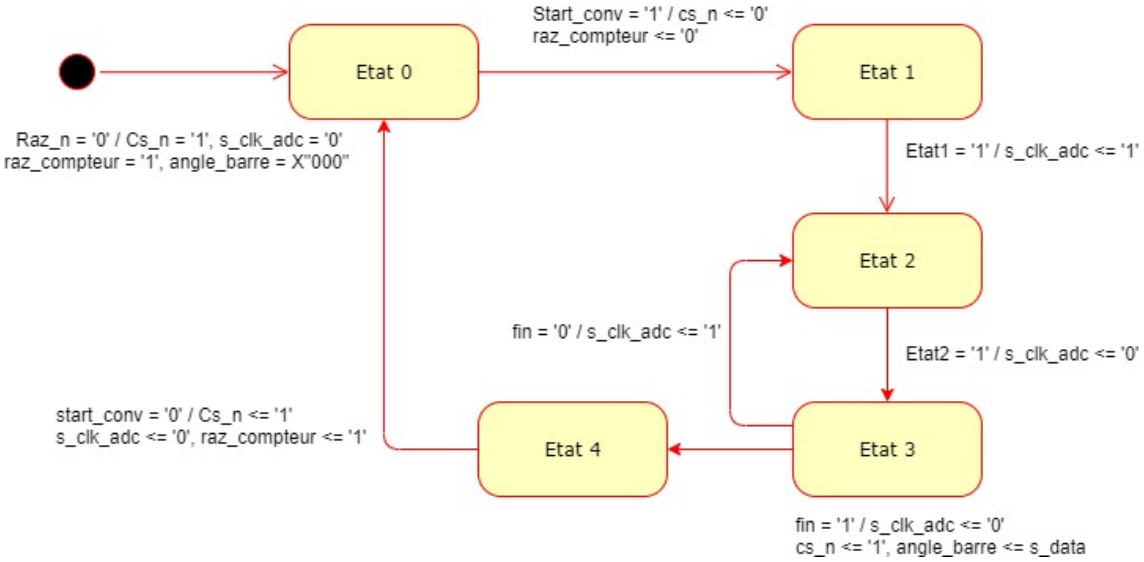


FIGURE 6 – Machine à états Gestion vérin

2.3 Réalisation Asservissement vérin (Compas + Interface)

Après l'implémentation des fonctions simple et complexe le temps nous a permit d'instancier l'asservissement du vérin en y ajoutant l'interface Homme système (boutons, Leds et buzzer) ainsi que la partie compas nécessaires à l'asservissement. Nous avons par manque de temps copié les fonctions déjà présente sur le site du projet mais avons développé nous mêmes les parties centrales pour éviter de copier un projet sans le comprendre. Les machines à états développées dans les fichiers VHDL sont donc développées par nous mêmes.

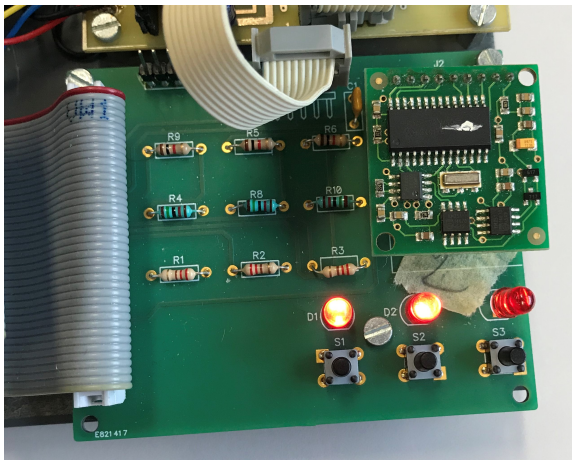


FIGURE 7 – Boutons poussoir, leds et buzzer

La partie compas développée nous a permit d'intégrer le composant CMPS03 qui est une boussole I2C (module à droite de la figure 7). Ce module va nous permettre dans l'asservissement de définir un cap à suivre pour le voilier. Ce bloc fonctionnel est complémenté par le bloc interface qui va permettre durant l'asservissement de corriger (+1/+10 degrés ou -1/-10 degrés) et d'afficher le cap que suit le voilier. Le passage du mode manuel vers le mode automatique se fera aussi à l'aide de l'interface implémentée en fin du bureau d'étude (en bas de la figure 7).

2.4 Réalisation interfaces Avalon

La création d'une interface processeur et matériel programmable est nécessaire au bon fonctionnement du projet. Altera 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 créé ultérieurement.

Pour réaliser cette interface il a fallut respecter le cahier des charges donné en début de projet pour chaque interfaces, qui correspondent à des circuits combinatoires/séquentiels utilisés pour l'écriture et la lecture du bus (Address, read, write, chip select...). Ci-dessous sont représentés les différents circuits que nous avons implémenté au projet.

La première interface correspond à la fonction Anémomètre.

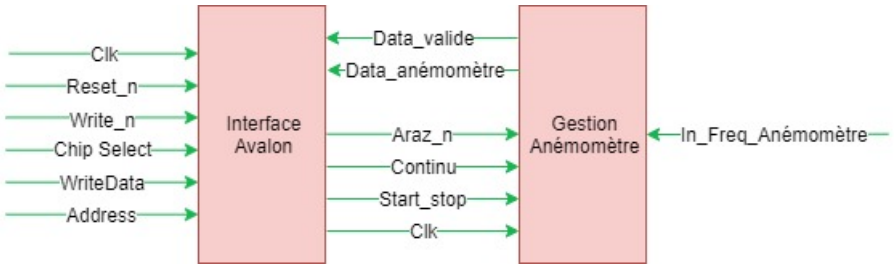


FIGURE 8 – Interface Avalon Gestion anémomètre

La deuxième interface correspond à la fonction Gestion vérin.

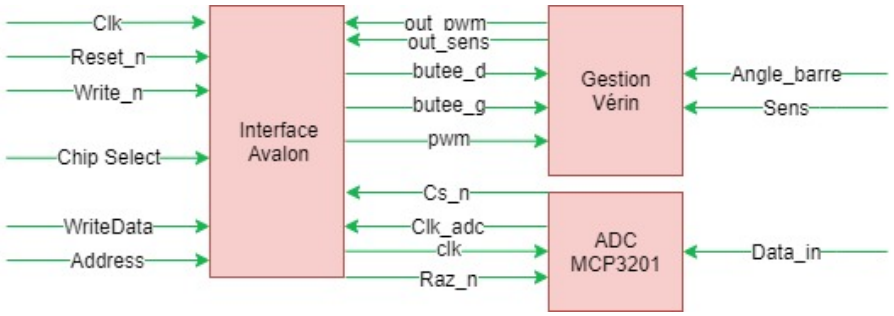
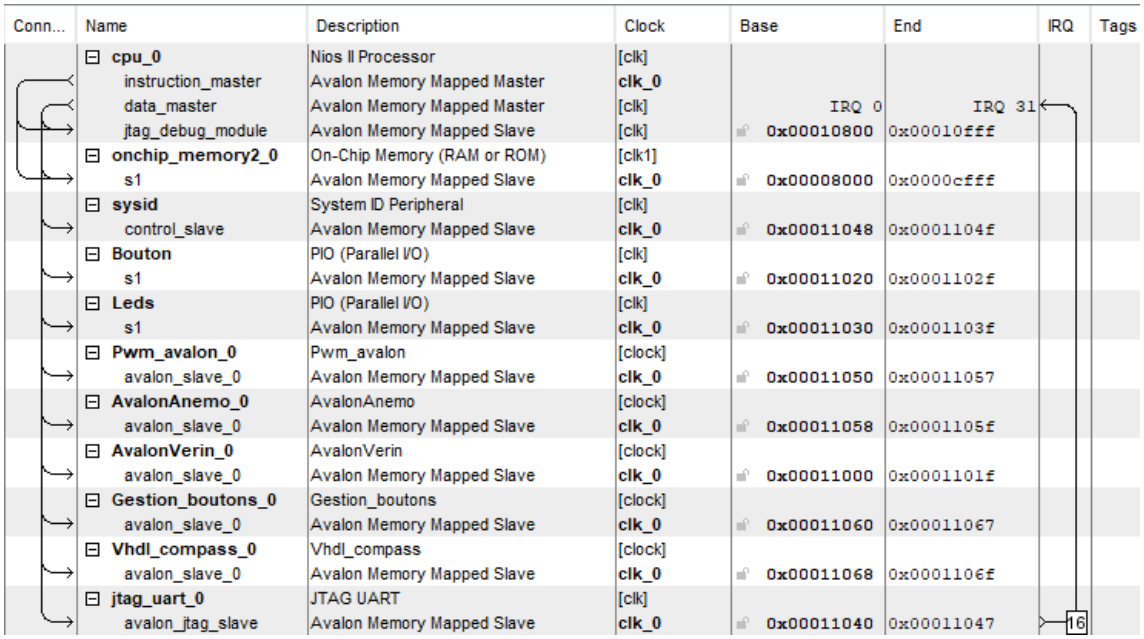


FIGURE 9 – Interface Avalon Gestion vérin

3 Conception conjointe Matérielle/Logicielle

La conception conjointe , est aussi appelée "codesign" qui correspond à une fonctionnalité complexe qui allie une partie logique programmée (flexibilité) et une logique câblée (performances). Utilisée pour réaliser des système complexes sur Silicium ou SOPC, dans notre cas (System On Programmable Chip) sur FPGA. Ce qui permet d'intégrer les périphériques ainsi que le processeur dans le FPGA, dans notre projet il s'agît de la réalisation d'un micrô-contrôleur.

Dans le système réalisé pour le projet nous avons généré, à l'aide de "SOPC Builder", un système SOPC comportant l'ensemble des fonctions du projet à partir de leurs fichiers VHDL. Dans le système on y retrouve le processeur NIOS II sur 32 Bits qui va permettre l'exécution des fonctions codées dans la partie logicielle, de la mémoire (type SRAM) de 20 Ko, un JTAG qui va permettre de créer une interface entre l'environnement de programmation (NIOS II) et le SOPC. Pour finir un composant "SYSID" qui va permettre l'identification du système et va protéger le système de tous mauvais téléchargements de programmes ne correspondant pas à l'application.



Conn...	Name	Description	Clock	Base	End	IRQ	Tags
	cpu_0	Nios II Processor	[clk]				
	instruction_master	Avalon Memory Mapped Master	**clk_0**				
	data_master	Avalon Memory Mapped Master	[clk]			IRQ 0	IRQ 31
	jtag_debug_module	Avalon Memory Mapped Slave	[clk]	**0x00010800**	0x00010fff		
	onchip_memory2_0	On-Chip Memory (RAM or ROM)	[clk1]				
	s1	Avalon Memory Mapped Slave	**clk_0**	**0x00008000**	0x0000cfff		
	sysid	System ID Peripheral	[clk]				
	control_slave	Avalon Memory Mapped Slave	**clk_0**	**0x00011048**	0x0001104f		
	Bouton	PIO (Parallel I/O)	[clk]				
	s1	Avalon Memory Mapped Slave	**clk_0**	**0x00011020**	0x0001102f		
	Leds	PIO (Parallel I/O)	[clk]				
	s1	Avalon Memory Mapped Slave	**clk_0**	**0x00011030**	0x0001103f		
	Pwm_avalon_0	Pwm_avalon	[clock]				
	avalon_slave_0	Avalon Memory Mapped Slave	**clk_0**	**0x00011050**	0x00011057		
	AvalonAnemo_0	AvalonAnemo	[clock]				
	avalon_slave_0	Avalon Memory Mapped Slave	**clk_0**	**0x00011058**	0x0001105f		
	AvalonVerin_0	AvalonVerin	[clock]				
	avalon_slave_0	Avalon Memory Mapped Slave	**clk_0**	**0x00011000**	0x0001101f		
	Gestion_boutons_0	Gestion_boutons	[clock]				
	avalon_slave_0	Avalon Memory Mapped Slave	**clk_0**	**0x00011060**	0x00011067		
	Vhdl_compass_0	Vhdl_compass	[clock]				
	avalon_slave_0	Avalon Memory Mapped Slave	**clk_0**	**0x00011068**	0x0001106f		
	jtag_uart_0	JTAG UART	[clk]				
	avalon_jtag_slave	Avalon Memory Mapped Slave	**clk_0**	**0x00011040**	0x00011047	16	

FIGURE 10 – Les différents composants du SOPC

Une fois les différents composants ajoutés avec leurs différents fichiers VHDL, une boite globale du projet est créée par "SOPC Builder".

FIGURE 11 – SOPC et ses différents ports GPIO

Il est nécessaire, une fois la fonction de Sopc réalisée, de gérer l'affectation des pins du FPGA avec celle générées dans notre fichier sopc. Il est important de respecter les pins déjà attribuées dans le cahier des charges car certaines sont déjà câblées sur la maquette de développement.

4 Développement Code Embarqué

Dans cette section nous présenterons l'utilisation de NIOS II qui est l'outil de développement de Altera permettant la programmation, compilation et le debug du logiciel que nous implémenterons sur le micro-contrôleur de notre FPGA réalisé précédemment (sous environnement Eclipse).

Dans cette partie nous avons utilisé les spécifications du circuit d'interface données dans le cahier des charges pour lier la lecture/écriture des adresses contenant les différents valeurs/données à traiter pour le projet. Si on prend l'exemple de la gestion vérin donnée dans le cahier des charges :

Registre	*adresse*	*type*	*Bits concernés*
freq	*0*	*R/W*	*b15..b0*
duty	*1 (4)*	*R/W*	*b15..b0*
Butee_g	*2 (8)*	*R/W*	*b15..b0*
Butee_d	*3 (12)*	*R/W*	*b15..b0*
config	*4 (16)*	*R/W*	*b0 :raz_n (à 0=reset circuit)*
		R/W	*b1 :enable_pwm (1= pwm actif)*
		R/W	*b2: sens rotation (0=gauche)*
		R	*b3:fin_course_d (=1 si fin course_d)*
		R	*b4: fin_course_g (=1 si fin course_g)*
Angle_barre	*5 (20)*	*R*	*b11..b0*

FIGURE 12 – Adressage des données Gestion Vérin

Il est donc nécessaire dans la partie logicielle embarquée de définir toutes les adresses correspondantes aux fonctions implémentées de tous les blocs matériels. En compilant le projet dans l'environnement Eclipse on peut récupérer les adresses correspondantes des composants dans le fichier "System.h" et les utiliser dans le "main.c" :

```
#define freq (unsigned int*)PWM_AVALON_0_BASE
#define duty (unsigned int*)(PWM_AVALON_0_BASE + 4)

#define vent (unsigned int*)(AVALONANEMO_0_BASE + 4)
#define config (unsigned int*)AVALONANEMO_0_BASE

#define butee_d (int *)(AVALONVERIN_0_BASE+12)
#define butee_g (int *)(AVALONVERIN_0_BASE+8)
#define freq_verrin (int *)AVALONVERIN_0_BASE
#define duty_verrin (int *)(AVALONVERIN_0_BASE+4)
#define config_verrin (int *)(AVALONVERIN_0_BASE+16)
#define angle_barre (int *)(AVALONVERIN_0_BASE+20)

#define code_sens (char *) BOUTON_BASE

#define config_av_button *(unsigned int*) (GESTION_BOUTONS_0_BASE)
#define code_av_button *(unsigned int*)(GESTION_BOUTONS_0_BASE+4)

#define config_compas (unsigned int*) VHDL_COMPASS_0_BASE
#define code_compas (unsigned int*) (VHDL_COMPASS_0_BASE+4)
```

FIGURE 13 – Implémentation en langage C des adresses matériels

5 Validation des Fonctions

Pour effectuer la validation de notre système nous avons téléversé notre projet sur le FPGA, la partie Hardware grâce au SOPC et la partie logicielle à l'aide de l'environnement Eclipse qui nous a permit de lancer/tester et débugger notre projet directement sur la maquette d'évaluation.

Nous avons aussi testé grâce à l'outil de simulation de Quartus certains composants comme le composant "Div" (Clock_1Hz), le composant "Detect_FM" (Détection fronts montants) et la gestion "PWM" qui est importante pour la gestion du vérin. Pour cela il faut créer un fichier "Vector Waveform File" et insérer les noeuds que nous souhaitons tester dans l'intervalle de temps, qui doit être adapté pour chaque simulations.

5.1 Validation Horloge 1 Hz - Anémomètre

Une simulation de la fonction "Div" (génération signal 1 Hz) a été réalisée avant implantation, une mesure toutes les secondes est nécessaire pour le bon fonctionnement de l'anémomètre c'était donc pour nous une des parties importantes à tester avant de poursuivre le reste du projet.

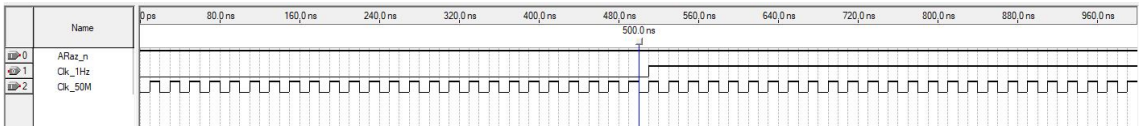


FIGURE 14 – Simulation du circuit de gestion d'horloge 1 Hz

5.2 Validation Détecteur Fronts montants - Anémomètre

Une simulation de la fonction "Detect_FM" a été réalisée avant implantation, dans cette seconde fonction importante on demande au système de détecter les fronts montants générés par l'anémomètre qui va permettre au système d'informer la vitesse du vent toutes les secondes.

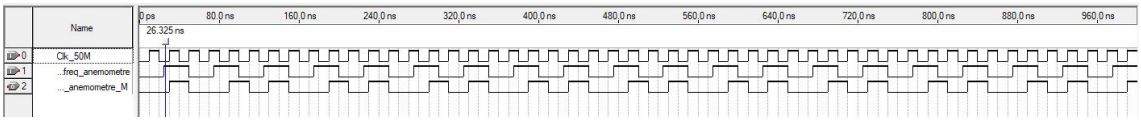


FIGURE 15 – Simulation du circuit de détection des fronts montants

5.3 Validation PWM - Gestion vérin

Une simulation de la fonction "PWM" a été réalisée avant implantation, dans cette fonction importante de la partie "Gestion vérin" on demande au système de générer un signal PWM qui va être interprété et traité par un composant électronique qui va à son tour piloter le vérin dans un sens ou dans l'autre. Dans cette simulation on observe le bon fonctionnement de celui-ci en faisant varier les valeurs de "Fréquence" et de "Duty".

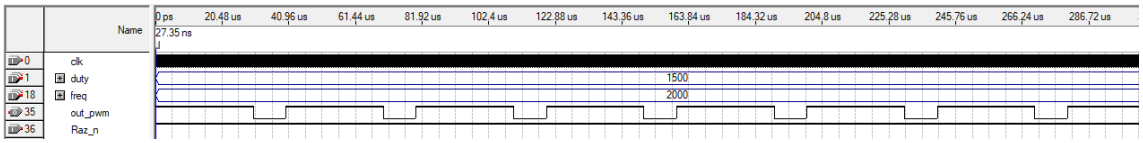


FIGURE 16 – Simulation du circuit de gestion du signal PWM - F = 2000, D = 1500

Nous avons réalisé trois tests pour valider le bon fonctionnement du projet. Pour le premier test nous avons utilisé un GBF qui nous a permit d'injecter directement sur le GPIO, du FPGA attribué, une fréquence qu'on peut modifier pour venir simuler la sortie de l'anémomètre. Après observation la fréquence est bien actualisée toutes les secondes comme demandé dans le cahier des charges. Pour le second test nous venons tester la fonction gestion vérin avec la fonction interface Homme système les boutons nous on permit de tester le bon fonctionnement qui contrôle le signal PWM du vérin. Pour la dernière fonction qui est l'asservissement du système complet nous observons un maintient du cap définit dans le logicielle téléversé sur la maquette de développement et que celle-ci respecte bien le cap fixé avant chaque passage en mode automatique.

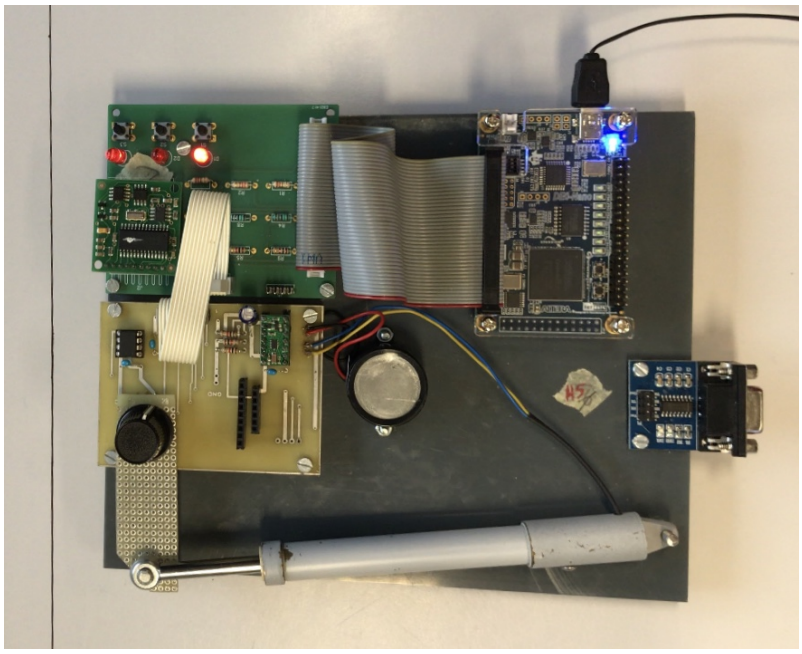


FIGURE 17 – Maquette validation projet

6 Conclusion

Pour la réalisation du projet nous avons eu peu d'expérience dans le domaine de la programmation VHDL et avons du se former à l'aide d'exemples fournit durant la conception du projet. Grâce aux cartes de développement DE0 Nano nous avons pu nous entraîner et implémenter des fonctions en parallèle des cours et des séances de bureau d'étude.

7 Annexes

Les différentes annexes (codes C, codes VHDL, fichiers sopc ainsi que le reste) sont disponibles sur le Github du projet Barre_Franche à cette adresse :

`https://github.com/NicolasO-git/Barre_Franche`

Le Github contient un README.md décrivant le projet et l'objectif du projet durant l'année universitaire du Master 2.

Il y a un dossier Image contenant les différentes images et captures d'écrans réalisées durant le projet.

Il y a un dossier Documentations contenant les différents pdf utiles à la bonne réalisation du projet.

Un dossier Compte Rendu Latex contenant le document pdf et les différents chapitres latex réalisées pour le compte rendu du projet.

Et un dossier Code embarqué comportant les différents composants simples (dossier Composants), les fonctions (F1 anémomètre, F2 compas et F6 Vérin) et l'asservissement ainsi que la partie logicielle du projet Barre_Franche. Pour Lancer le projet sous Quartus 11 il suffit d'ouvrir le fichier "Sopc_Barre.qpf" dans le dossier "Sopc_Barre_Franche".