



UNIVERSITÉ
TOULOUSE III
PAUL SABATIER



Rapport BE VHDL

Djaroud Mohamed

12 Décembre 2022

Master 2 SME

Encadrant :

Thierry PERISSE
Pedro CARVALHO

Année universitaire 2022/2023

Tables des matières :

1. INTRODUCTION.....	2
2. PRÉSENTATION GLOBALE DU PROJET.....	3
2.1.Présentation générale du projet.....	3
2.2.Présentation technique du projet.....	4
3. FONCTION SIMPLE : ANÉMOMÈTRE.....	5
3.1.Présentation de l'anémomètre.....	5
3.2.Analyse fonctionnelle de la fonction gestion anémomètre.....	6
3.3.Implémentation de la fonction gestion anémomètre en VHDL.....	8
3.4.Conception du SOPC.....	10
4. FONCTION COMPLEXE : VÉRIN.....	11
4.1.Analyse fonctionnelle du vérin	11
4.2.Implémentation de la fonction gestion vérin en VHDL.....	13
4.3.Gestion du convertisseur AN MCP 3201.....	13
4.4.Conception du SOPC.....	15
5. CONCLUSION.....	16

1.INTRODUCTION :

L'objectif de ce bureau d'étude est de concevoir le pilote de barre franche sous forme d'un système sur puce programmable SOPC (System On Programmable Chip) décrite à l'aide du langage de description de Hardware VHDL (Very High Speed Hardware Description Langage) en se basant sur l'analyse de spécifications et découpage fonctionnel du système choisi et la conception de circuits d'interfaces numériques en VHDL pour le simuler et le valider sur la maquette, puis faire des interfaçages avec les bus microprocesseur tels que NIOS, Altera, Avalon pour la validation du SOPC en manipulation.

Objectif du BE :

- Analyse de spécifications et découpage fonctionnel du système choisi.
- Conception de circuits d'interfaces numériques en VHDL (conception, simulation, vérification sur maquette)
- Notion de Co-design et règles de conception
- Interfaçage avec bus microprocesseur (NIOS + Altera Avalon)
- Conception d'un SOPC et intégration D'IP (Intellectual Properties) propriétaires et fournisseurs tiers
- Notions de simulation « Hardware In the Loop »
- Validation du SOPC en simulation (pour parties) et sur maquette.

2. PRÉSENTATION GLOBALE DU PROJET

2.1 Présentation générale du système

Un pilote automatique pour voilier est un équipement électrique ou hydraulique destiné à maintenir le cap d'un voilier à la place d'un équipier. Ces pilotes automatiques sont très utiles aux navigateurs solitaires ou en équipage réduit. Le pilote est constitué de trois éléments principaux : un compas, une unité électronique et une unité de puissance. Sur tous les pilotes de la nouvelle génération, le compas est électronique, il donne continuellement à l'unité de traitement le cap suivi par le bateau. Cette même unité a comme consigne le cap que l'on souhaite suivre. Elle compare en permanence ces deux caps, s'ils ne sont pas identiques, elle donne l'ordre à l'unité de puissance d'agir sur la barre pour ramener le bateau sur son cap. L'unité de puissance pour les pilotes pour barre franche est un vérin linéaire. Ce vérin a une extrémité fixée sur le banc de cockpit, l'autre sur la barre. Toute variation de cap lui est transmise et il agit en conséquence sur la barre.

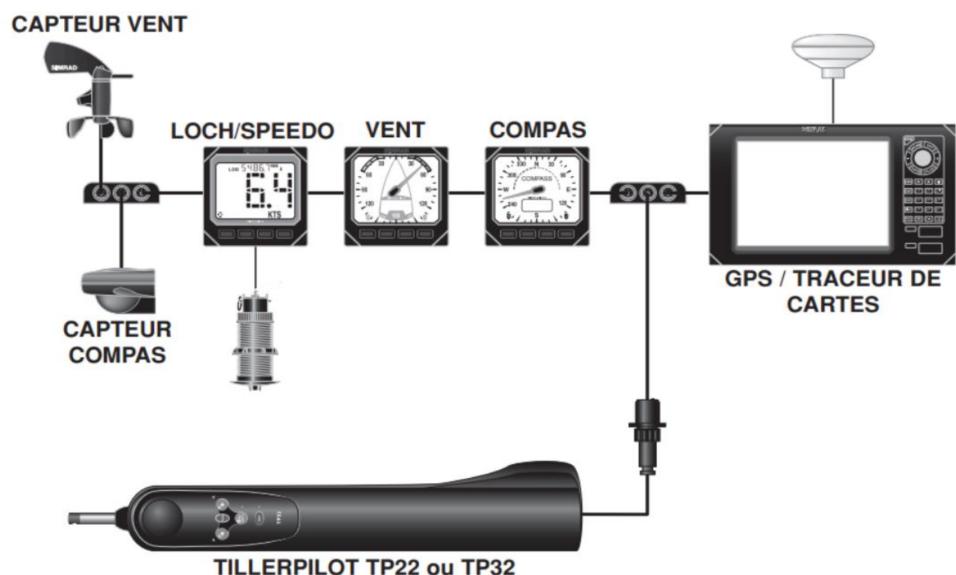


FIGURE 1 : Pilote de barre franche SIMRAD TP10

Le système devra :

- Utiliser deux appareils de mesure :
 - Anémomètre : Capteur d'acquisition de vitesse du vent
 - Compas : Conversion Cap
- Avoir un Vérin qui change le cap du voilier

- Une trame NMEA sera utilisée pour communiquer les données entre le FPGA et l'interface
- IHM (Boutons, Leds..)

2.2 Présentation technique du système

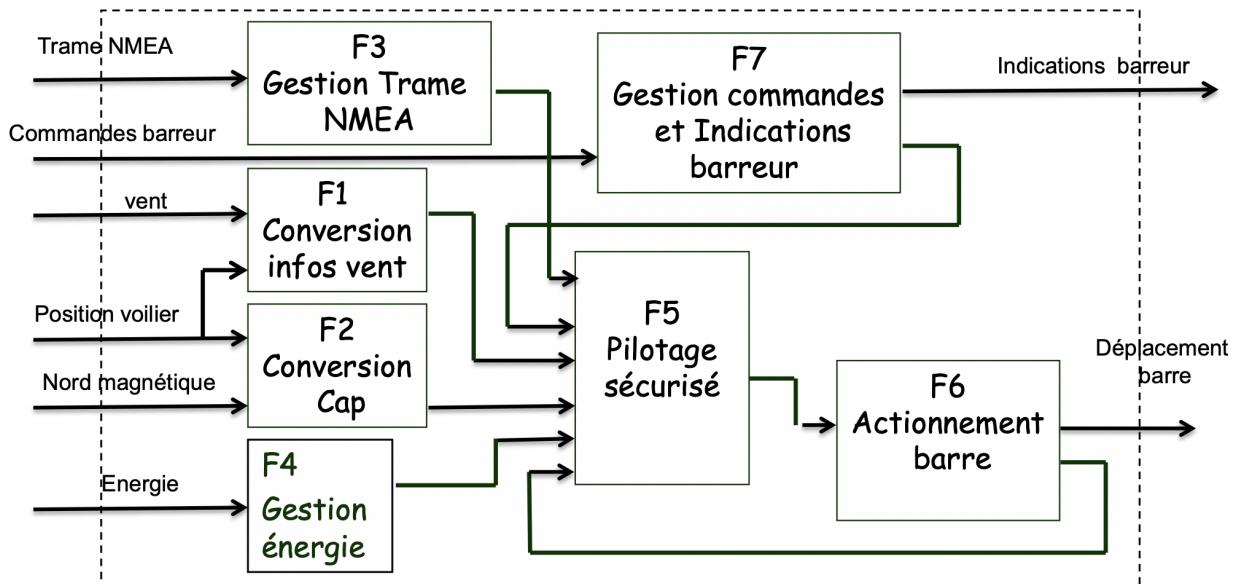


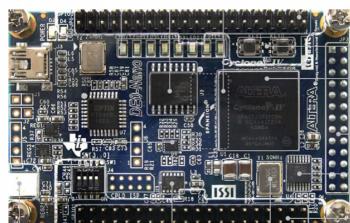
FIGURE 2 : Architecture globale du projet.

Ces différentes fonctions devront être décomposées en sous fonction plus simple afin d'être développées.

Il y aura dans ce gros bloc des fonctions simples et des fonctions complexe. Le but étant pour chaque groupe de travailler sur au moins 2 sous fonctions, une fonction simple et une fonction complexe.

Matériel à disposition :

- Quartus prime lite 18.1
- Carte DE0 Nano équipée d'un cyclone 4 CE22 (BE), Altera cyclone 4 : ref : EP4CE22F17C6N.



- Altera DE2 à base de FPGA Cyclone II, Altera cyclone 2 : ref : EP2C35F672C6N



3. FONCTION SIMPLE : ANÉMOMÈTRE

3.1. Présentation de l'anémomètre

L' anémomètre est un instrument qui mesure le vent (vitesse, direction et pression). Le matériel choisi mesure la vitesse du vent et délivre un signal carré en sortie de fréquence variable en fonction de la vitesse du vent.

La plage de mesure de l'anémomètre est comprise entre 0 – 250 km/h, et délivre en sortie un signal carré de fréquence correspondant à la vitesse du vent, soit 0 – 250 Hz.



FIGURE 3 : Vitesse du vent, Anémomètre.

3.2. Analyse fonctionnel de la fonction gestion anémomètre

Voici les spécifications du module gestion anémomètre :

entrées:

- clk_50M : horloge 50MHz
- raz_n: rest actif à 0 => initialise le circuit
- in_freq_anemometre: signal de fréquence variable de 0 à 250 HZ
- 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 lorsqu'une mesure est valide
- est remis à 0 quand start_stop passe à 0
- data_anemometre : vitesse vent codée sur 8 bits

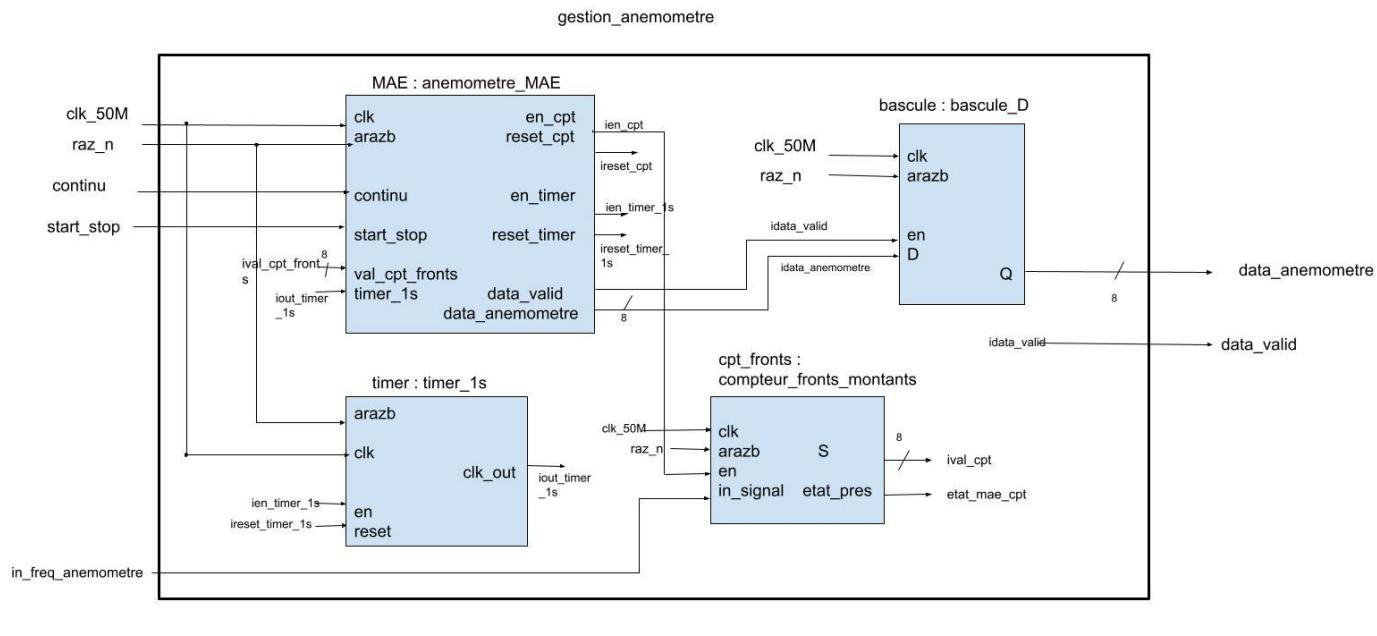


FIGURE 4 : Schéma fonctionnel Anémomètre.

Pour répondre bien aux exigences de notre circuit à concevoir, nous avons réalisé la description fonctionnelle suivant :

Machine à état : Bloc qui nous permet de récupérer à l'état 3 la valeur du 'data_anémomètre' et de savoir si l'on se met en mode continu ou en mode monocoup.

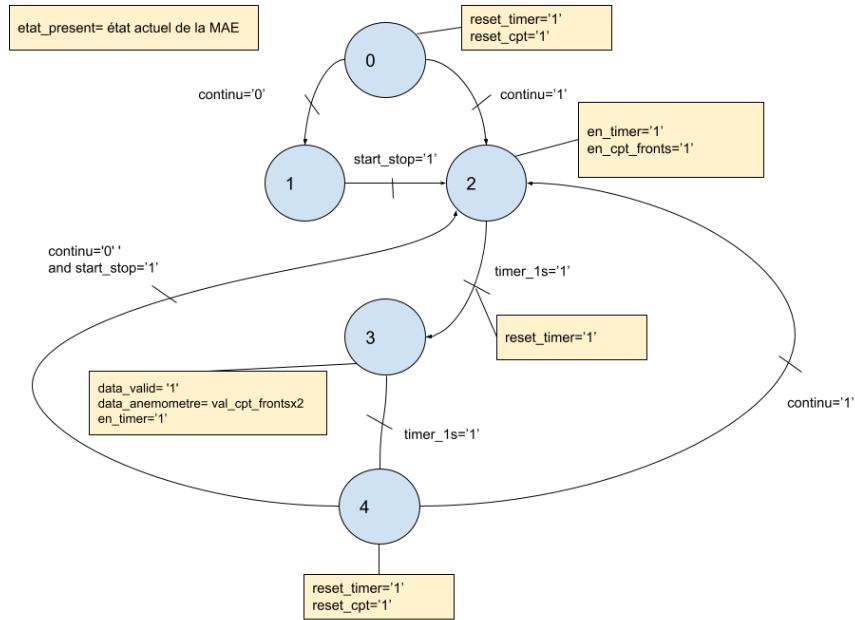


FIGURE 5 : Fonctionnement de la machine à état de l'anémomètre.

Bascule D : Va nous permettre de retarder le signal d'entrée d'un coup d'horloge.

Compteur de front montants : C'est un bloc qui permet de compter le nombre des fronts montant du signal numérique dans le but de mesurer la fréquence.

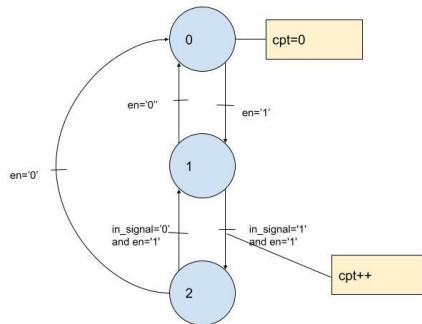


FIGURE 6 : Fonctionnement de la machine à état du compteur de fronts montants.

Timer_1s : Génère une horloge de 1Hz.

3.3. Implémentation de la fonction gestion anémomètre en VHDL

Après avoir réalisé l'analyse fonctionnelle de notre circuit nous avons basculé vers Quartus pour implémenter notre système par des blocs en VHDL comme il est indiqué dans la figure ci-dessous :

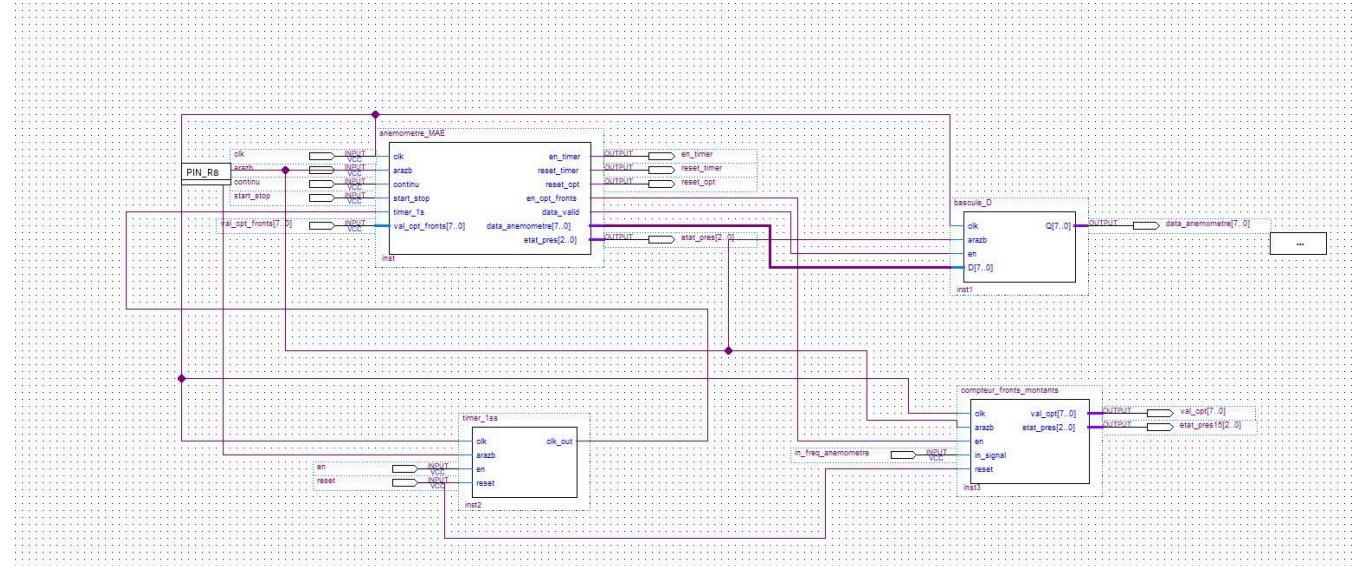


FIGURE 7 : Implémentation de l'anémomètre sur Quartus.

Avant de manipuler le fonctionnement de notre système nous avons lancé plusieurs simulations sur Quartus 9 comme il est indiqué dans les figures ci dessous :

→ Analyse des signaux de la machine à état:

On peut voir sur la figure ci-dessous qu'à travers la simulation, qu'elle répond parfaitement à ce que l'on cherchait à obtenir comme on peut le voir sur le schéma du mode de fonctionnement de la machine à état de la figure 5.

Par exemple, lorsque nous sommes à l'état 3, comme nous le montre 'état_pres', nous nous retrouvons avec un 'data-valid' à 1 et le 'data_anémomètre' qui récupère deux fois la valeur du compteur de fronts montants.

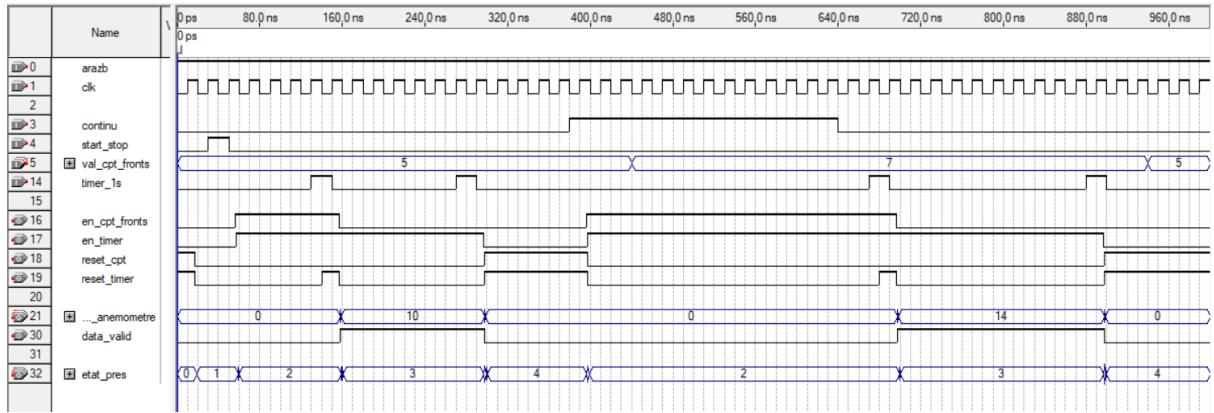


FIGURE 8 : Simulation sur Quartus 9 du bloc Machine à État.

Ensuite nous avons réalisé une simulation pour observer le comportement des signaux sur le compteur de front montants. Nous voulions vérifier que la machine à état de la figure 6 était respecté.

Comme on peut le voir sur la figure 9, lorsque ‘en’ est égale à un, c'est à dire que l'on autorise le fonctionnement, l'état passe à 1 et si on reçoit un signal sur ‘in_signal’ qui est le signal de la fréquence d'entrée de l'anémomètre, alors nous passons à l'état 2 et on incrémenté le compteur sur ‘val_cpt’.

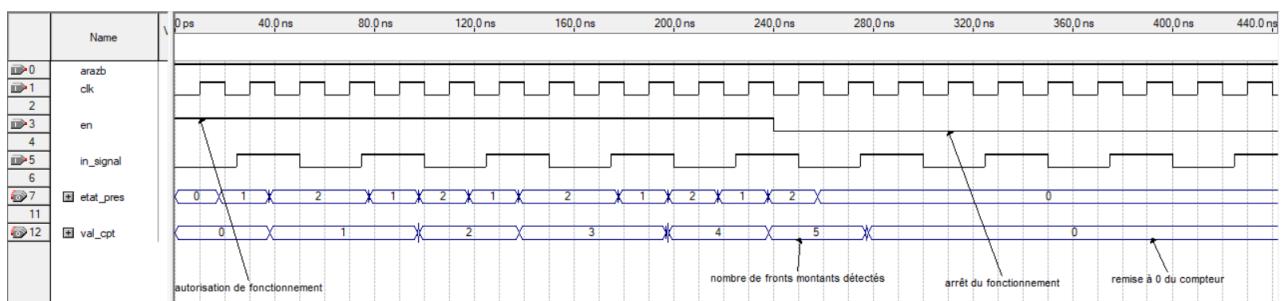


FIGURE 9 : Simulation sur Quartus 9 du bloc compteur de fronts montants.

Ci dessous la simulation de toute la gestion de l'anémomètre.

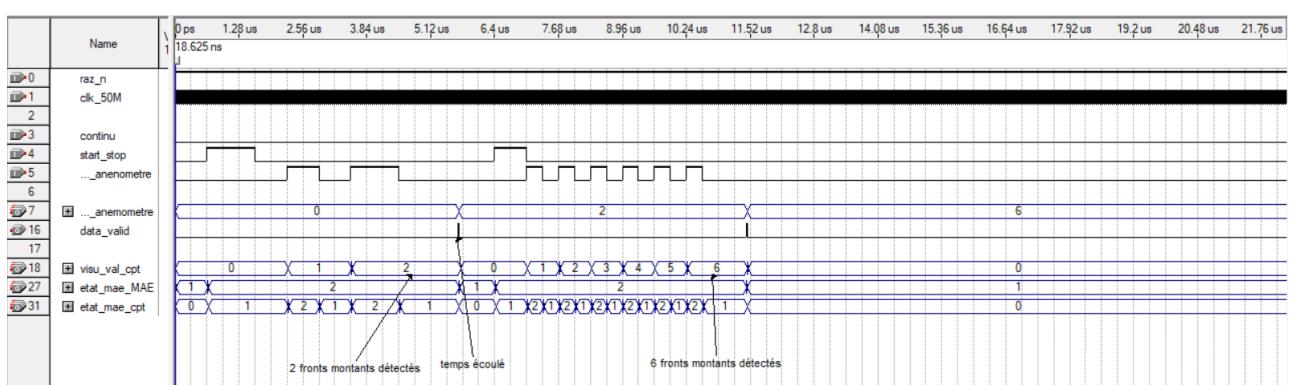


FIGURE 10 : Simulation sur Quartus 9 du bloc de gestion de l'anémomètre.

3.4. Conception du SOPC

Afin de télécharger le circuit sur la carte DE0 nous avons utilisé le SOPC Builder afin de créer le microprocesseur et les éléments périphériques en intégrant l'anémomètre et Avalon PWM comme il est indiqué dans les figures ci-dessous :

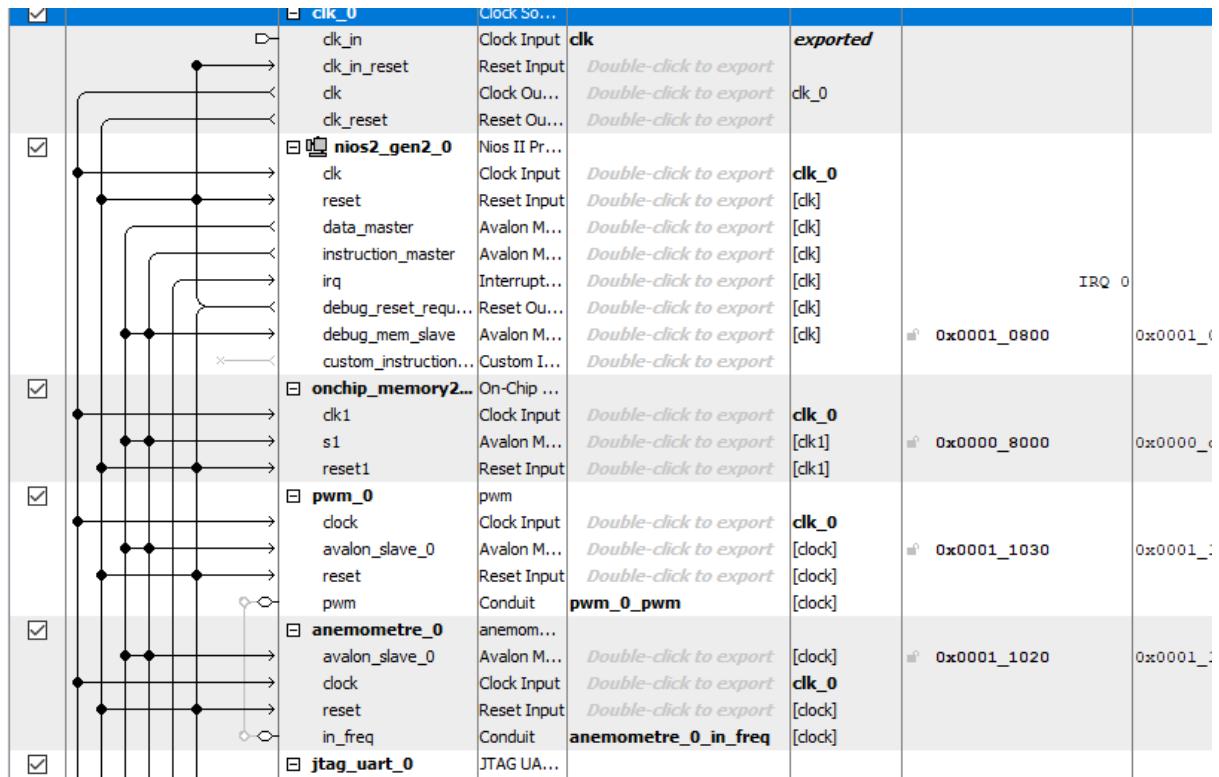


FIGURE 11 : Conception du SOPC sur Platform Designer.

Après avoir construit notre système en SOPC Builder et intégré l'Avalon PWM et l'anémomètre dedans, nous avons obtenu le bsf qui est indiqué dans la figure ci-dessous :

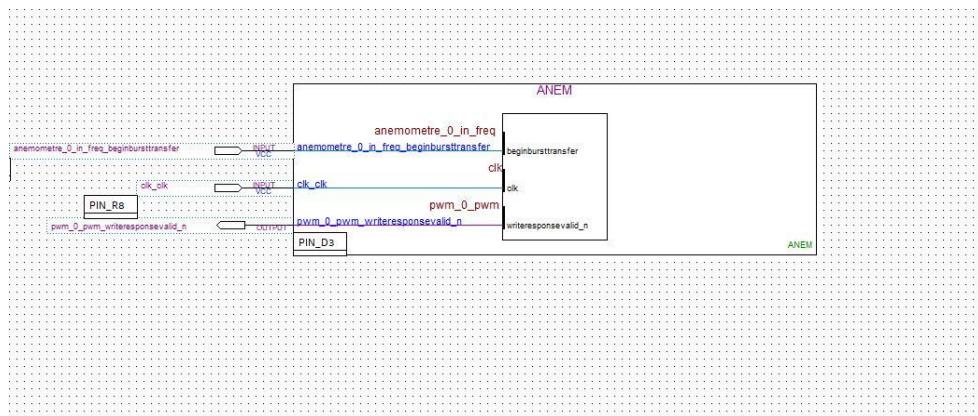


FIGURE 12 : Circuit sur Quartus 18 de l'avalon Anémomètre.

À la suite de ce travail réalisé sur Platform Designer, j'ai ensuite réalisé le code en C sur Nios II pour faire fonctionner le tout et avoir la valeur de data anémomètre en sortie. Malheureusement je n'ai pas pris de capture pour l'affichage mais je vous joins ci dessous le code en C qui nous permettait d'afficher la vitesse du vent.

```

6 #define freq (unsigned int *) PWM_0_BASE
7 #define duty (unsigned int *) (PWM_0_BASE+4)
8 #define control (unsigned int *) (PWM_0_BASE+8)
9 #define config (unsigned int *) (ANEMOMETRE_0_BASE)
10#define data_anemometre (unsigned int *) (ANEMOMETRE_0_BASE+4)
11
12
13 int main()
14 {
15
16     alt_putstr("Hello from Nios II!\n");
17     *freq=1024;
18     *duty=512;
19     *control=3;
20     *config=3;
21
22
23     /* Event loop never exits. */
24     while (1){
25         // alt_putstr("Hello from the boucle\n");
26
27         printf("config : %d \n", *config);
28         printf("frequence : %d \n", *freq);
29         usleep(10000);
30         printf("data_anemometre : %d \n", *data_anemometre);
31         usleep(10000);
32     }
33     return 0;
34 }
```

FIGURE 13 : Code en C sur Nios II pour l'affichage du data anémomètre.

4. FONCTION COMPLEXE : VÉRIN

4.1. Analyse fonctionnelle du vérin

Le circuit de gestion du vérin est constitué de quatre fonctions principales :

Gestion de la PWM : elle fait appel à 2 fonctions secondaires :

- Une fonction qui fixe la fréquence de la PWM (process divide)

- Une fonction qui fixe le rapport cyclique et génère le signal PWM en sortie (process compare)

Contrôle des butées : utilise la fonction principale « `controle_butées` » :

- met le signal « `PWM` » à 0 si « `angle_barre` » se situe en dehors des butées « `butee_g` » et « `butee_d` » et selon le sens de rotation du moteur.
- génère les signaux « `fin_course_d` » et « `fin_course_g` »

Gestion du convertisseur AN MCP 3201 : fait appel à 4 fonctions secondaires :

- registre à décalage pour récupérer la donnée du convertisseur (process `rec_dec`)
- génération du 1MHz pour la machine à état (process `gene_1M`)
- génération périodique (toutes les 100ms) du signal « `start_conv` » (process `gene_start_conv`)
- ChipSelect

Interface avec le bus Avalon : fait appel à 2 fonctions secondaires (process) :

- Ecriture des data circulant sur le bus du NIOS dans des registres (`ecriture`)
- Relecture de signaux par renvoi sur le bus du NIOS (`lecture`)

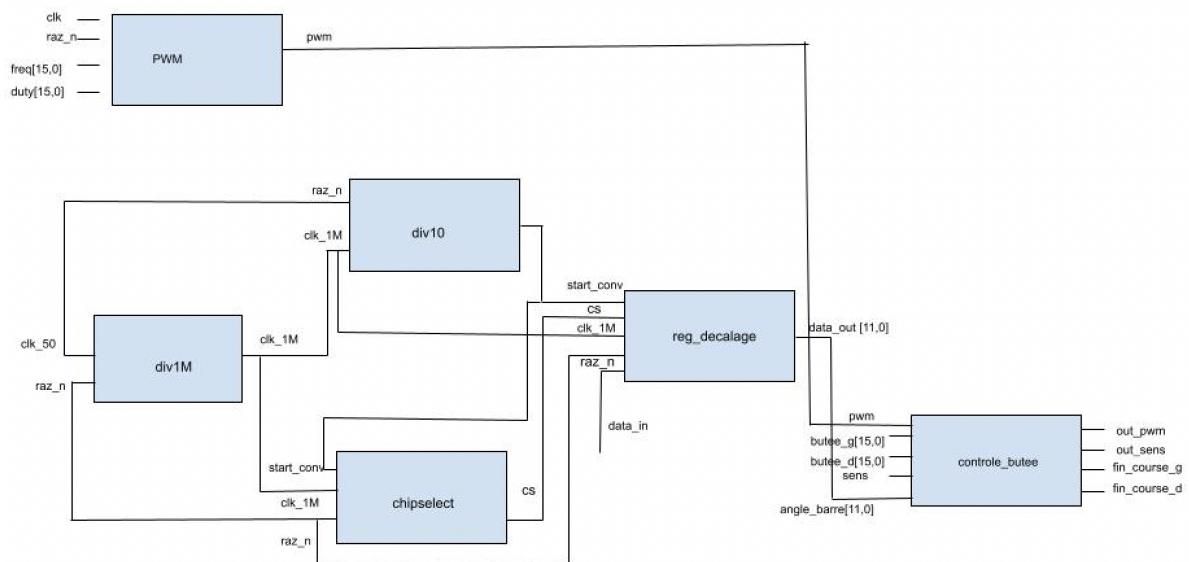


FIGURE 14 : Analyse fonctionnelle du Vérin.

4.2. Implémentation de la fonction gestion Vérin en VHDL

Après avoir réalisé l'analyse fonctionnelle de notre circuit nous avons basculé vers Quartus pour implémenter notre système par des blocs en VHDL comme il est indiqué dans la figure ci-dessous :

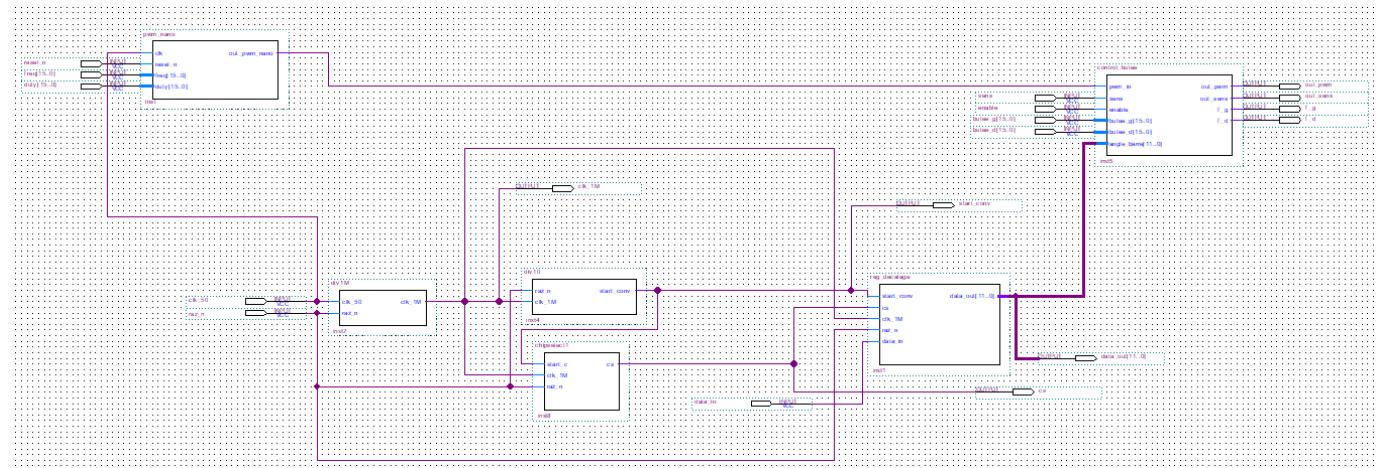


FIGURE 15 : Implémentation du Vérin sur Quartus 18.

4.3. Gestion du convertisseur AN MCP 3201

Le programme vhdl MCP3201 est constitué de 4 sous blocs principaux qui sont :

- Un générateur de signal 1 MHZ
- Un registre à décalage
- Générateur de signal start_conv chaque 100 ms
- Un Chip Select

Fonctionnement du convertisseur :

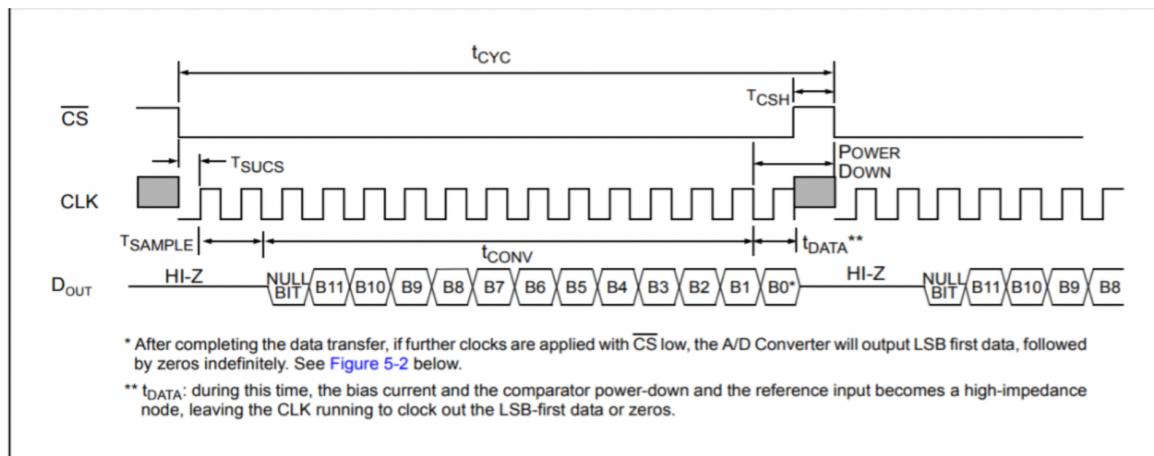


FIGURE 16 : Datasheet du convertisseur.

Comme nous pouvons le voir sur la figure ci-dessus, le convertisseur utilise trois signaux pour fonctionner :

- CS : un chip select. On doit le mettre à l'état bas pour activer le composant et démarrer une acquisition.
- CLK : l'horloge de référence du convertisseur que l'on doit établir à 1Mhz pour ce BE.
- Dout: la sortie du convertisseur qui sera relié au FPGA pour récupérer les données converties

Ci dessous la simulation sur l'analyseur de réseau disponible sur Quartus 18.



FIGURE 17 : Simulation du convertisseur sur l'analyseur de réseau.

En fonction de la variabilité du data_in, on aura une valeur de data_out allant de 0 à 4095. On peut voir ici sur la figure précédente que lorsque le start_conv est à 0 et que le chip select est à l'état bas, on démarre l'acquisition et on obtient notre data_out grâce au registre de décalage qui est programmée comme suit:

```

process(clk_1M, raz_n)
variable cnt : integer range 0 to 16;
variable cnt2 : integer range 0 to 16;

begin
  if raz_n = '0' then
    data_out <= x"000";
  elsif falling_edge(clk_1M) then
    if start_conv = '0' and cs = '0' then
      cnt := cnt + 1;
      cnt2 := cnt2 + 1;
      if cnt >= 5 then
        if cnt2 <= 15 then
          decalage(16- cnt2) <= data_in;
        end if;
      end if;
    else
      cnt := 0;
      cnt2 := 0;
    end if;
  end if;
  data_out <= decalage;
end process;

```

FIGURE 18 : Code VHDL du registre de décalage.

Entre l'état bas et l'état haut du chip select nous avons 16 périodes de 1 MHz. On utilisera les compteurs sur les fronts descendants.

Cnt sert à aller jusqu'à 5 tandis que le cnt2 compte tout le temps de 5 jusqu'à 15.

Nous avons donc deux if imbriqués, et nous commencerons par un décalage (16-5) donc 11.

On aura de B11 à B0 comme nous le montre la figure ci-dessus.

4.4. Conception du SOPC

Afin de télécharger le circuit sur la carte DE0 nous avons utilisé le SOPC Builder afin de créer le microprocesseur et les éléments périphériques en intégrant la gestion du véerin et Avalon PWM comme il est indiquée dans les figures ci-dessous :

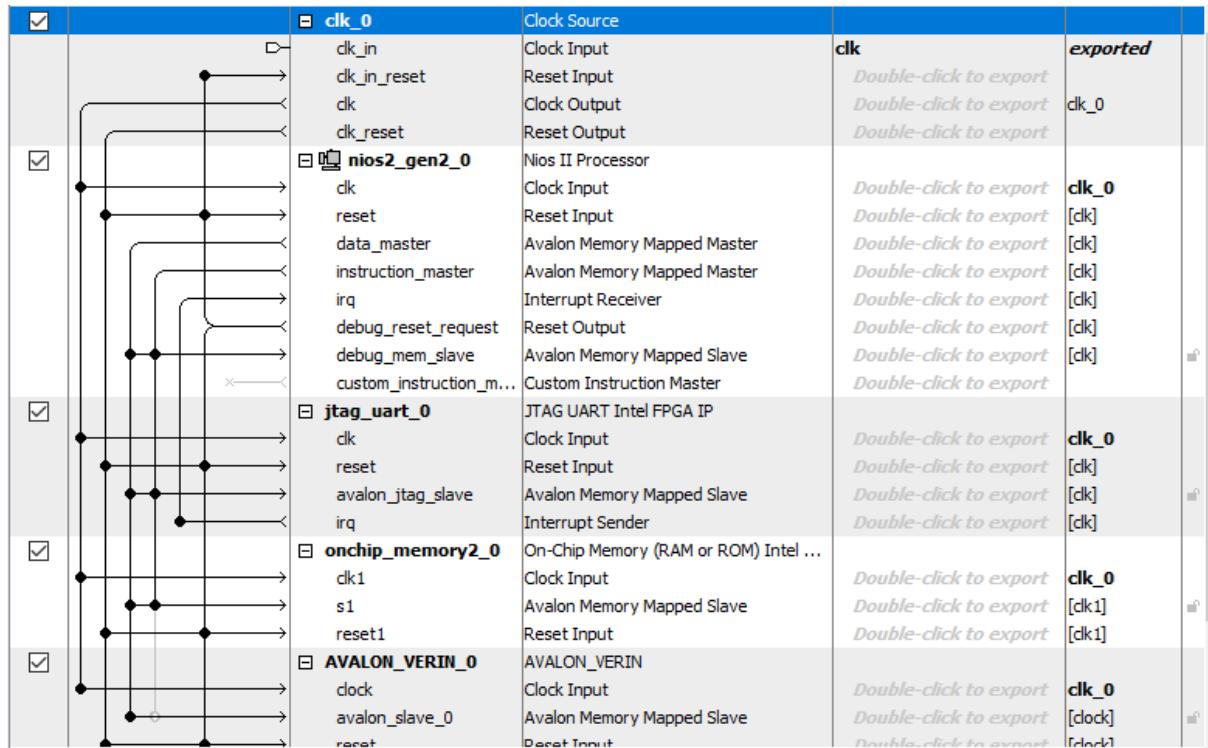


FIGURE 19 : Conception du SOPC du Véerin sur Platform Designer.

Après avoir construit notre système en SOPC Builder et intégré l'Avalon PWM et l'avalon du gestion Véerin dedans, nous avons obtenu le bsf qui est indiqué dans la figure ci-dessous :

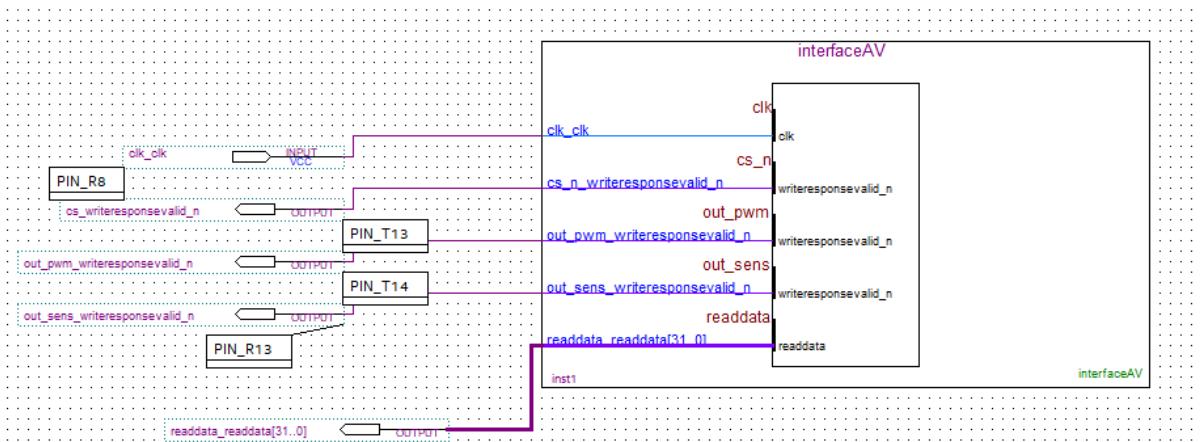


FIGURE 20 :Circuit sur Quartus 18 de l'avalon Véerin.

5.CONCLUSION

Pour conclure, ce projet fut très enrichissant. Il m'a permis de concevoir, valider par la simulation et l'implémentation à partir d'outils appropriés, tout ou une partie des fonctions identifiées lors de la phase d'analyse. J'ai pu réaliser la synthèse et la réalisation de fonctions logiques en langage VHDL, la vérification du fonctionnement en simulation et sur maquette (carte DE2 d'Altera), l'interfaçage des composants développées avec le bus du processeur (bus Avalon et processeur NIOS 32 bits) et pour finir le test et la validation du système complet.

Ce projet m'a permis d'acquérir des compétences techniques, de mettre en œuvre les différentes compétences acquises lors de nos cours magistraux et de nous confronter aux difficultés techniques entraînées par la réalisation d'un projet. De pouvoir concevoir, tester et observer un produit développé et réalisé de ses propres mains est quelque chose d'important et qui accorde une grande fierté et satisfaction.