UNIVERSITÉ TOULOUSE III PAUL SABATIER

RAPPORT BE

PILOTE DE BARRE FRANCHE POUR VOILIERS AVEC VHDL

Encadré par :

Mr Thierry PERISSE

Réalisé par :

Abdelhak ELALAOUI

MASTER 2 SYSTÈMES ET MICROSYSTÈMES EMBARQUÉS

2023 -- 2024

TABLEAU DES MATIÈRES

TABLEAU DES FIGURES

INTRODUCTION

L'objectif central qui guide les travaux de ce TP consiste à élaborer le pilote de barre franche sous une forme innovante, à savoir un système sur puce programmable (SOPC). Cette conception novatrice se déploie à travers une approche méthodique et rigoureuse, où la description du matériel s'effectue au moyen du langage VHDL (Very High Speed Hardware Description Language), réputé pour sa robustesse et son efficacité.

Le processus démarre par une analyse détaillée des spécifications du projet, offrant ainsi une base solide pour le découpage fonctionnel du système sélectionné. C'est à partir de cette analyse préliminaire que les ingénieurs du bureau d'études se lancent dans la conception de circuits d'interfaces numériques en VHDL. Ces circuits jouent un rôle crucial dans la simulation et la validation du système sur la maquette, garantissant ainsi la cohérence et la fiabilité des résultats obtenus.

Une étape conséquente de cette démarche implique l'interfaçage avec des bus microprocesseurs, tels que NIOS, Altéra, Avalon. Cette phase d'interconnexion vise à valider le SOPC par le biais de manipulations avancées, assurant ainsi une compatibilité et une intégration fluides avec les technologies existantes.

Ce travail s'engage à travers un processus itératif et complet, débutant par l'analyse approfondie des spécifications, passant par la conception rigoureuse des circuits en VHDL, pour finalement aboutir à la validation pratique du pilote de barre franche sur une maquette d’essai. Cette approche holistique témoigne de l'engagement du bureau d'études à réaliser un développement technologique de pointe, harmonisant les exigences du projet avec les possibilités offertes par les SOPC et le langage VHDL.

I) PRÉSENTATION DU SYSTÈME

1) INTRODUCTION

Un pilote automatique destiné aux voiliers représente un équipement électrique ou hydraulique conçu pour assurer le maintien du cap d'un voilier, en remplacement d'un équipier. Ces dispositifs revêtent une importance particulière pour les navigateurs solitaires ou ceux évoluant en équipage restreint. La structure fondamentale du pilote automatique se compose de trois éléments clés, à savoir un compas, une unité électronique, et une unité de puissance.

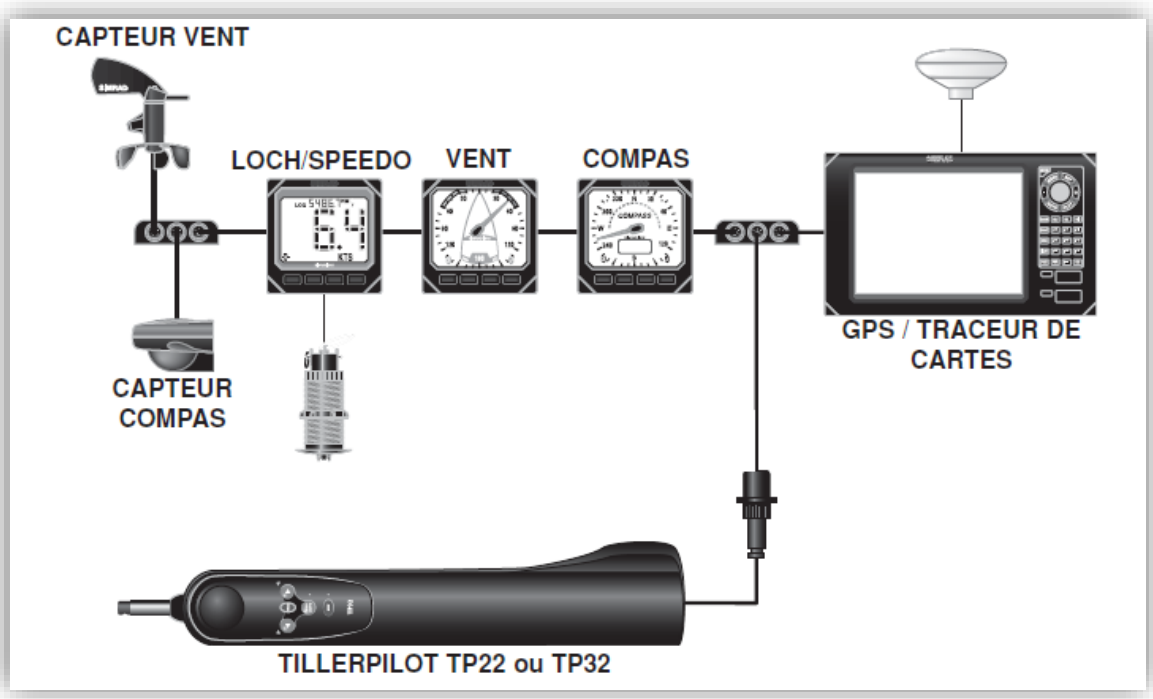


Figure 1 : Système de commande de la barre franche

Dans la dernière génération de pilotes automatiques, le compas est électronique, transmettant de manière continue au module de traitement le cap suivi par le bateau. L'unité électronique, quant à elle, se voit attribuer comme consigne le cap souhaité à maintenir. Elle procède à une comparaison constante entre ces deux caps ; si ces derniers ne concordent pas, elle émet une directive à l'unité de puissance pour intervenir sur la barre, réalignant ainsi le bateau sur son cap désiré. Pour les pilotes automatiques adaptés aux barres franches, l'unité de puissance se matérialise sous la forme d'un vérin linéaire.

Ce vérin, fixé à une extrémité sur le banc de cockpit et à l'autre sur la barre, réagit de manière instantanée à toute variation de cap qui lui est transmise, agissant en conséquence sur la position de la barre. Cette interaction dynamique entre le compas électronique, l'unité de traitement, et le vérin linéaire constitue le mécanisme essentiel qui permet au pilote automatique d'assurer le maintien précis du cap du voilier, offrant ainsi une assistance précieuse aux marins dans des conditions de navigation diverses.

2) MISE EN SERVICE DE LA BARRE FRANCHE

L'ajustement précis de l'angle entre le Tillerpilot ou le système et la barre franche constitue un élément essentiel, exigeant une configuration exacte de 90°. Prendre l'exemple des Tillerpilots de la société Simrad Ltd, tels que les modèles TP10, TP22 et TP32, dévoile une ingénierie méticuleuse visant à fusionner un niveau avancé de technologie avec des caractéristiques de pointe propres aux pilotes automatiques, le tout géré par un clavier ergonomique composé de cinq touches.

Les Tillerpilots sont spécifiquement conçus pour offrir une combinaison harmonieuse entre une technologie de pointe et des fonctionnalités de pilote automatique, tout en conservant un mode opératoire à la fois simple et complet, grâce à un clavier ergonomique intuitif. Ce dernier permet d'effectuer des réglages de cap précis et de tirer parti de l'ensemble des fonctions de navigation disponibles.

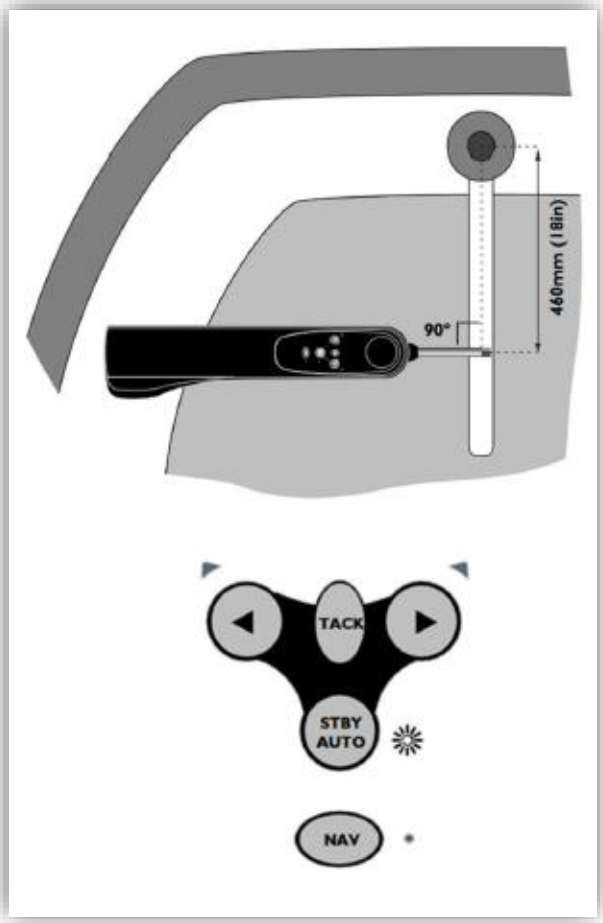


Figure 2 : Ajustement du Tillerpilot et la barre franche

En ce qui concerne les modes de fonctionnement, les Tillerpilots proposent plusieurs options, notamment le mode Veille, le mode Pilote automatique (STBY/AUTO), et le Mode NAV, qui permet au Tillerpilot de maintenir un cap précis en se basant sur des données provenant de sources telles que le GPS ou le protocole NMEA.

Les réglages de cap sont également personnalisables, offrant des options telles que des ajustements de 1° ou 10°, du côté bâbord ou tribord, avec des commandes à pression

courte ou prolongée. De plus, la possibilité de choisir entre un bip unique ou double bip, ainsi que des indicateurs lumineux tels que l'éclat LED (B ou T) ou le clignotement LED (B ou T), ajoute une flexibilité supplémentaire à l'utilisation du Tillerpilot, garantissant ainsi une adaptation optimale aux besoins spécifiques de la navigation.

3) LA MAQUETTE DE SYSTÈME

Le système embarqué étudié que nous tenterons de reproduire en utilisant des capteurs tels qu'un anémomètre, une girouette et une boussole, ainsi qu'une maquette élaborée sur mesure.

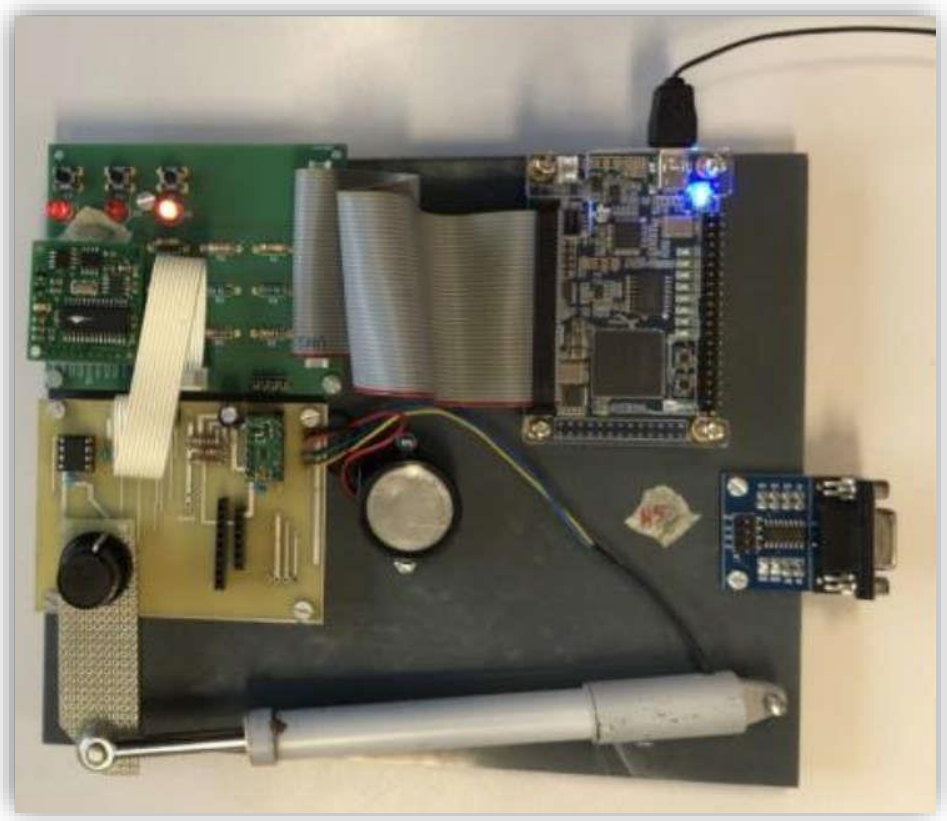


Figure 4 : Maquette de système de la barre franche

Cette maquette comportera un vérin contrôlé par un signal PWM via un pont en H, tandis que l'angle de la barre sera mesuré par un convertisseur analogique-numérique (CAN) à l'aide d'un potentiomètre. Les données NMEA seront transmises à travers le port série RS232, offrant ainsi une représentation fidèle du système embarqué que nous nous apprêtons à étudier et à reproduire expérimentalement.

Pour bien comprendre le système étudié, voici un schéma représentatif du système :

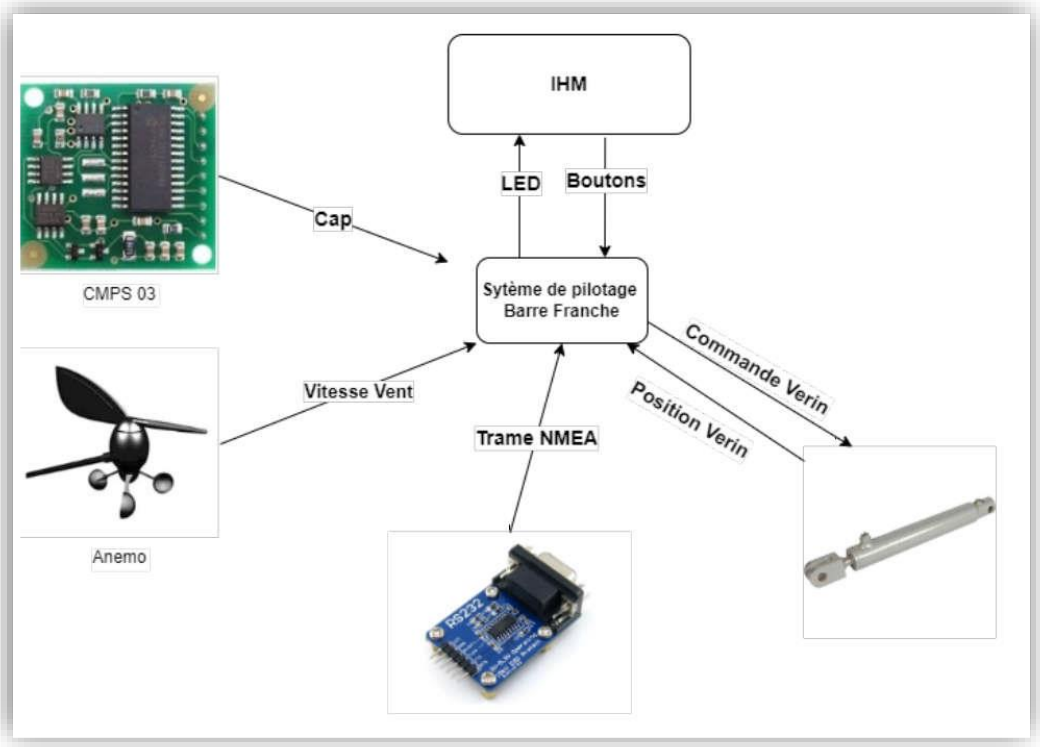


Figure 5 : Schéma représentatif du système étudié

4) TRAVAIL DEMANDÉ À FAIRE

Pour comprendre la répartition des fonctions de ce système on explore un exemple architectural dans cette figure :

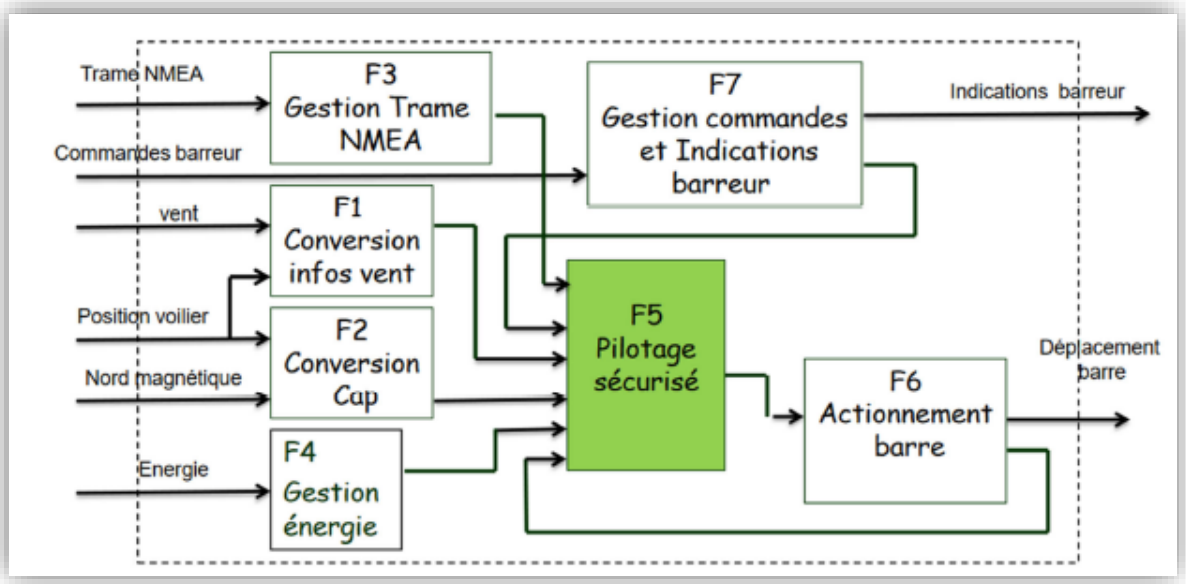


Figure 6 : Exemple architectural du système

Il faut choisir l'une de deux fonctions principales à implémenter :

- La fonction de compas qui est la boussole de voilier qui doit envoyer l'angle à suivre au système d'après une sortie de PWM qui sera détaillé par la suite.
- La fonction de l'anémomètre qui doit envoyer les données de la vitesse de vent sous une trame de 8 bits.

5) MATÉRIEL UTILISÉ DANS LE TP :

Carte de développement FPGA DEO – NANO

La carte DE0-Nano, plateforme FPGA, est dédiée à la conception de prototypes pour des applications comme les robots et les dispositifs portables. Elle vise une simplicité d'utilisation avec des composants Cyclone IV jusqu'à 22 320 éléments logiques et permet une extension au-delà de ses limites grâce à deux I/O générales externes.

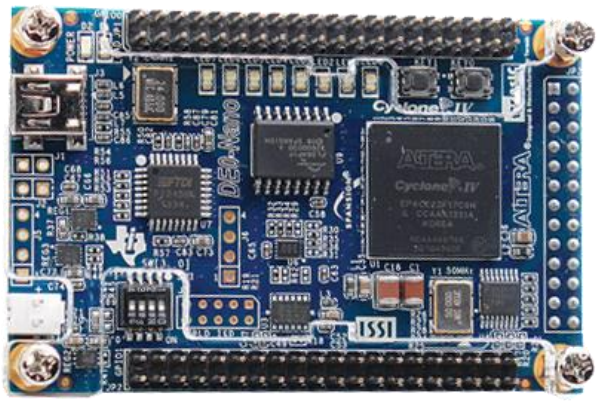


Figure 7 : Carte de développement DEO NANO

La carte intègre des fonctionnalités avancées, dont une gestion étendue de la mémoire avec SDRAM et EEPROM, ainsi que des périphériques améliorés tels que LED et boutons-poussoirs. Compacte, légère, reconfigurable sans matériel superflu, elle s'adapte parfaitement aux designs mobiles prioritaires en termes de puissance, offrant trois options de gestion d'alimentation.

Carte de développement DE2

Figure 8 : Carte de développement DE2

La carte DE2 est équipée d'un FPGA Cyclone II 2C35 de pointe, logé dans un boîtier à 672 broches, offrant un contrôle exhaustif sur tous les composants de la carte. Pour des expériences simples, elle comporte un ensemble d'interrupteurs robustes et des LED et des affichages à 7 segments.

Pour des projets plus avancés, la carte propose des SRAM, SDRAM, des puces de mémoire Flash et un affichage de 16 x 2 caractères. L'intégration du processeur Nios II d'Altéra avec des interfaces standard telles que RS-232 et PS/2 facilite les expériences nécessitant des fonctions de traitement et d'E/S. Les fonctionnalités audio et vidéo sont prises en charge avec des connecteurs standard pour microphone, entrée ligne, sortie ligne, entrée vidéo et VGA (DAC 10 bits), permettant la création d'applications audio de qualité CD et de vidéos professionnelles.

Le Compas CMPS03

Ce module boussole a été spécialement conçu pour être utilisé dans les robots comme aide à la navigation. L'objectif était de produire un numéro unique pour représenter la direction dans laquelle le système fait face.

Figure 9 : Le compas CMPS03

La boussole utilise le capteur de champ magnétique Philips KMZ51, suffisamment sensible pour détecter le champ magnétique terrestre. La sortie de deux d'entre eux montés à angle droit l'un par rapport à l'autre est utilisée pour calculer la direction de la composante horizontale du champ magnétique terrestre

II) MISE EN ŒUVRE DE SOPC

1) INTRODUCTION

Le SOPC Builder offre la possibilité de concevoir des microcontrôleurs spécifiques à une application, comprenant une section processeur associée à divers périphériques tels que des PIO, des Timers, des UART, des interfaces USB, des composants propriétaires, et plus encore, ainsi que de la mémoire. Cette mémoire peut être intégrée au sein du FPGA, formant ainsi ce que l'on appelle une RAM/ROM On Chip, ou elle peut être située à l'extérieur du composant FPGA. Le composant microprocesseur central utilisé est le NIOS II d'Altéra, aujourd'hui propriété d'Intel, un processeur 32 bits disponible en trois versions : économique, standard et rapide. La version économique, étant la moins puissante, utilise le minimum de ressources du FPGA. Il est également possible d'intégrer d'autres types de processeurs, à condition d'avoir accès à leurs modèles (VHDL, Verilog, etc.).

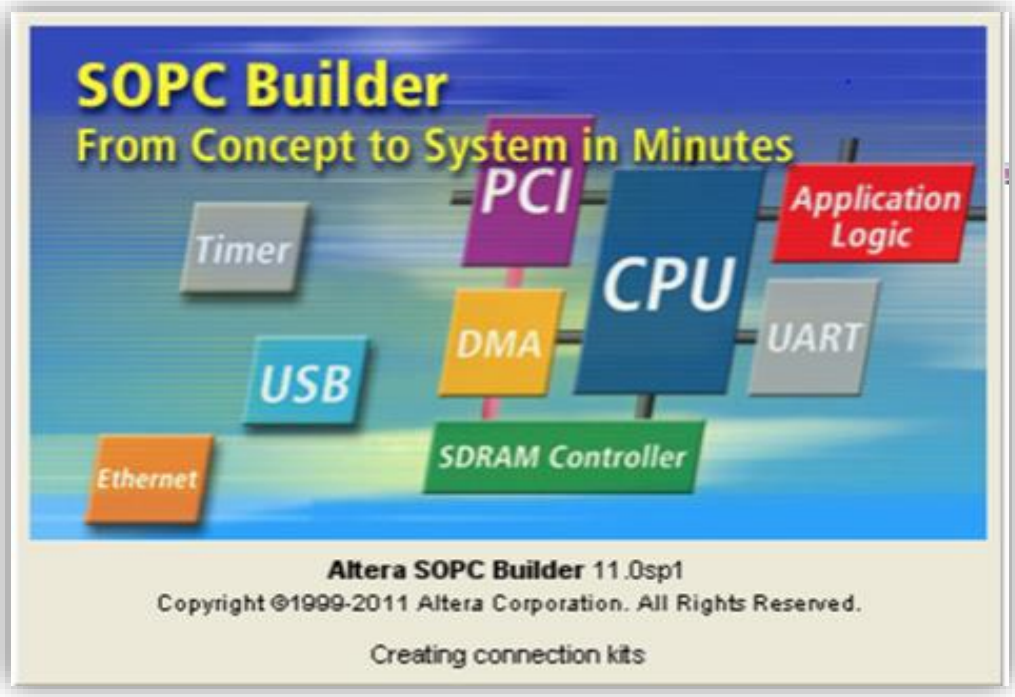


Figure 10 : SOPC Builder

Le but de cette configuration c'est créer un système ou un système programmable qui va simuler les connections entre les différents composants de SOPC (CPU, RAM, JTAG, SYS ID, PIO…) avec l'ajout d'autres composants externes qu'on doit créer comme l'Avalon PWM, le compas, l'anémomètre…

Pour commencer dans la création de notre SOPC, on va créer un Avalon PWM qui est un système qui génère un signal PWM qu'on peut ajuster la fréquence et longueur d'impulsion pour l'introduire dans le système NIOS par la suite avec la création d'un compteur à la même fois.

2) LA PARTIE MATÉRIEL

Pour accéder à la partie matérielle de Quartus pour configurer notre SOPC on doit ouvrir l'outil PLATFORM DESIGNER dans la fenêtre Outils :

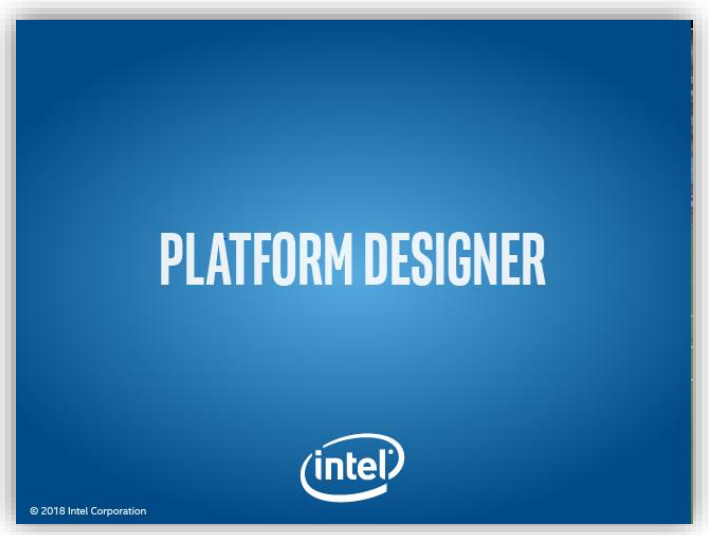


Figure 11 : Logo de PLATFORM DESIGNER

Pour la partie matérielle on utilisera dans notre SOPC les composants suivants :

- CPU : Le processeur NIOS II Processor et choisir la sélection économique
- RAM : RAM On-Chip Memory Intel avec une valeur de 20k
- JTAG : JTAG UART Intel pour la programmation de PORT avec une valeur d'interruption de 16.
- SYS ID : System ID Péripheral Intel pour ID de composant
- PIO : les entrées sorties de SOPC que dans notre cas sera deux : 2 boutons d'entrée et 8 LEDs en sortie pour le compteur.

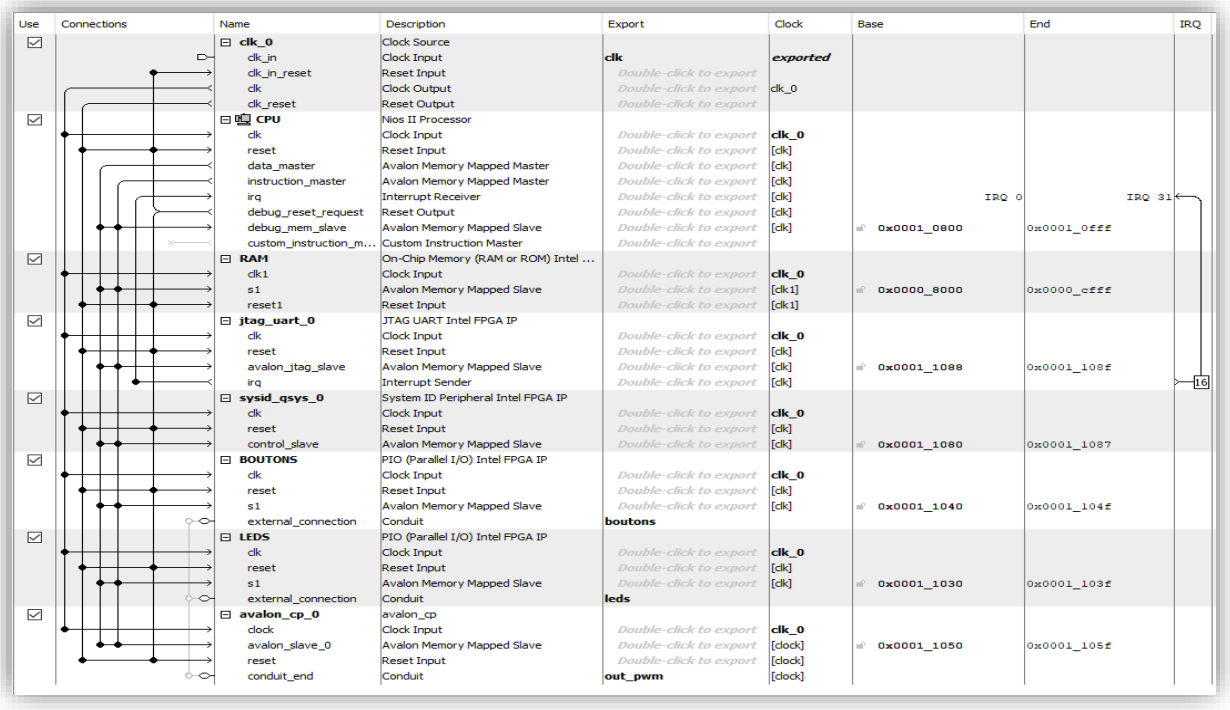


Figure 12 : Figure des composants de Hardware de SOPC

Pour le composant Avalon PWM il faut l'introduire dans la partie de création de composant :

- Nommer le composant **avalon_pwm**
- Dans la partie **Files**, ajouter le fichier de code VHDL déjà crée dans le dossier de projet et analyser les fichiers.
- Dans la partie **signals & interfaces,** ajouter une interface **conduite** où on doit ajouter le signal de sortie **out_pwm**

Associer le **reset** pour **l'avalon_slave** et l'interface de conduit.

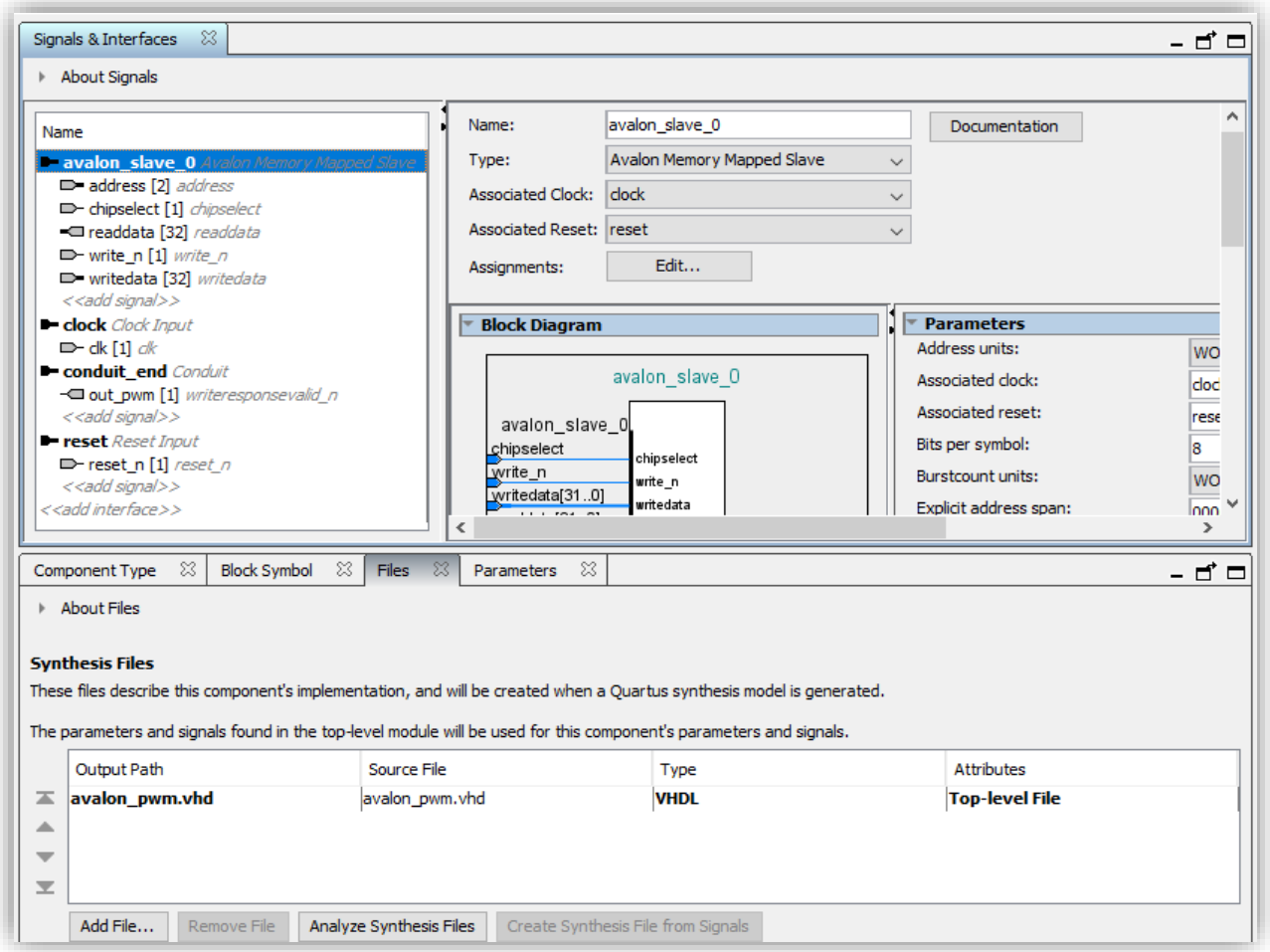


Figure 13 : Paramètres de l'avalon PWM

Après paramétrer tous les composants du SOPC, on doit les lier en appuyant sur les points blancs à gauche. Ne pas oublier d'ajuster la RAM dans le CPU dans les cases **reset vector** et **exception vector.**

Avant de finir, il faut assigner les connections en appuyant sur **System -> Assign Base Adresses** et ensuite générer le HDL en appuyant sur **Generate HDL** en on choisis la case **VHDL** dans **Synthesis**.

Dans ce moment, la partie matérielle est presque finie, il nous reste d'introduire le SOPC dans un schéma de bloc et assigner les ports d'entrées sorties pour le système avant de compiler le projet :

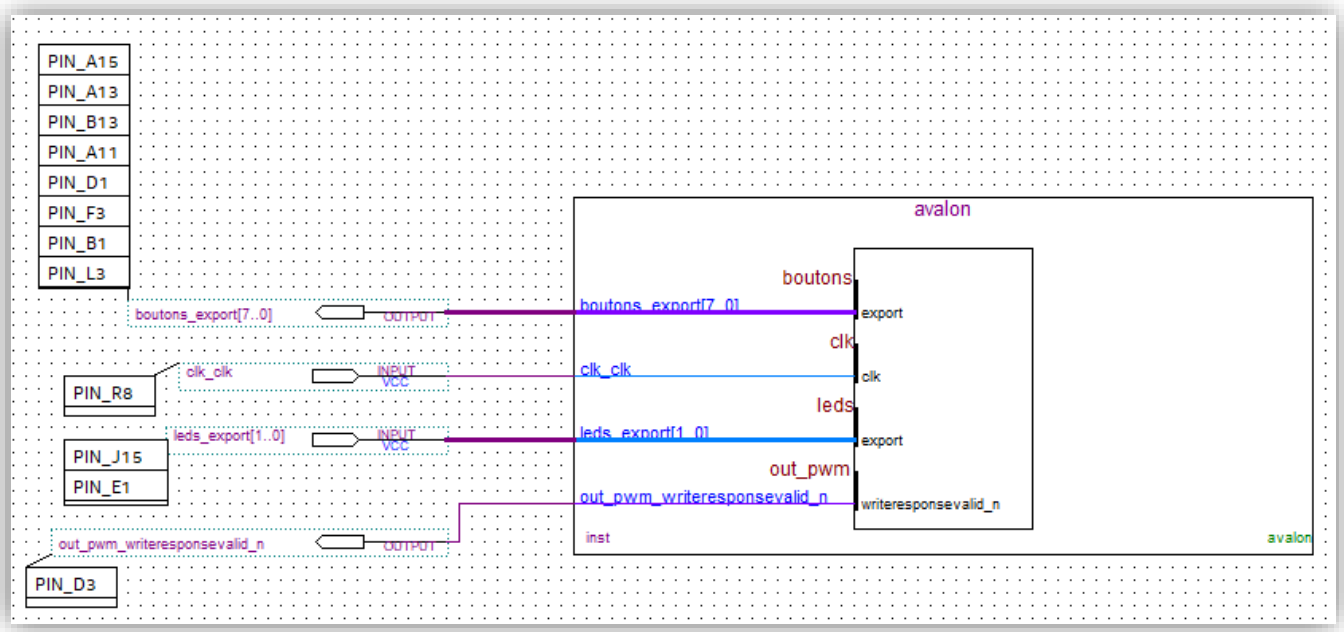


Figure 14 : Figure de bloc de l'avalon PWM

C'est le moment de compiler le projet avant de passer à la partie logicielle de projet en Eclipse pour teste le SOPC.

3) LA PARTIE LOGICIEL

Dans cette partie, on ouvre l'outil de NOIS II Software Tool ou Eclipse pour créer un nouveau projet pour programmer notre SOPC.

Avec la création de projet on doit sélectionner le fichier de **avalon.sopinfo** pour implémenter et le programmer. Il faut bluider le projet crée et l'exécuter sur **RUN as NIOS Hardware.**

Après l'ajout de programme de **pwm** dans le SOPC et l'exécution de programme on doit tester la sortie de pwm.

Voici le résultat de la sortie de pwm dans l'oscilloscope avec une fréquence de 200 KHz avec un Duty de 50 % comme assigné dans le code :

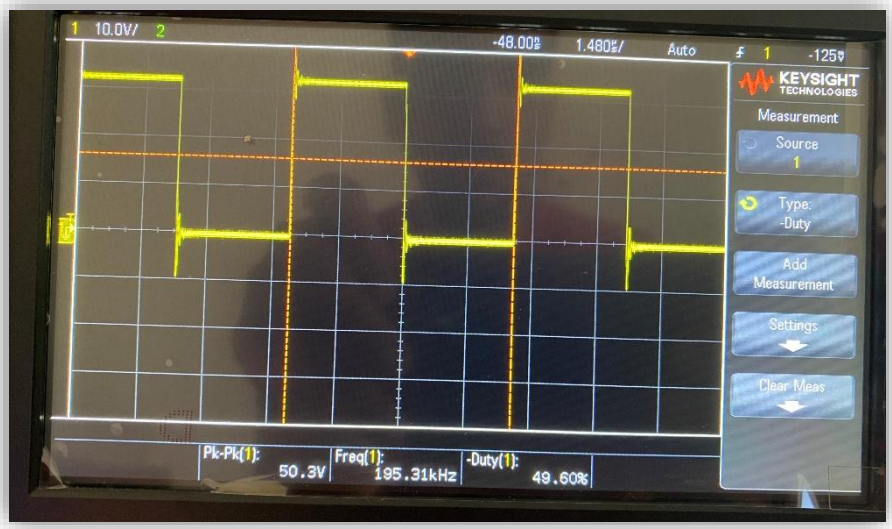


Figure 15 : Visualisation de la sortie pwm dans l'oscilloscope

4) INTRODUCTION DE SOPC DANS LE SYSTÈME

Le système SOPC est réalisé avec succès dans cette première partie pour savoir comment implémenter tous les composants et aussi la création par la suite d'un Bus Avalon qui va communiquer via serial avec le système comme on observe dans la figure ci-dessous :

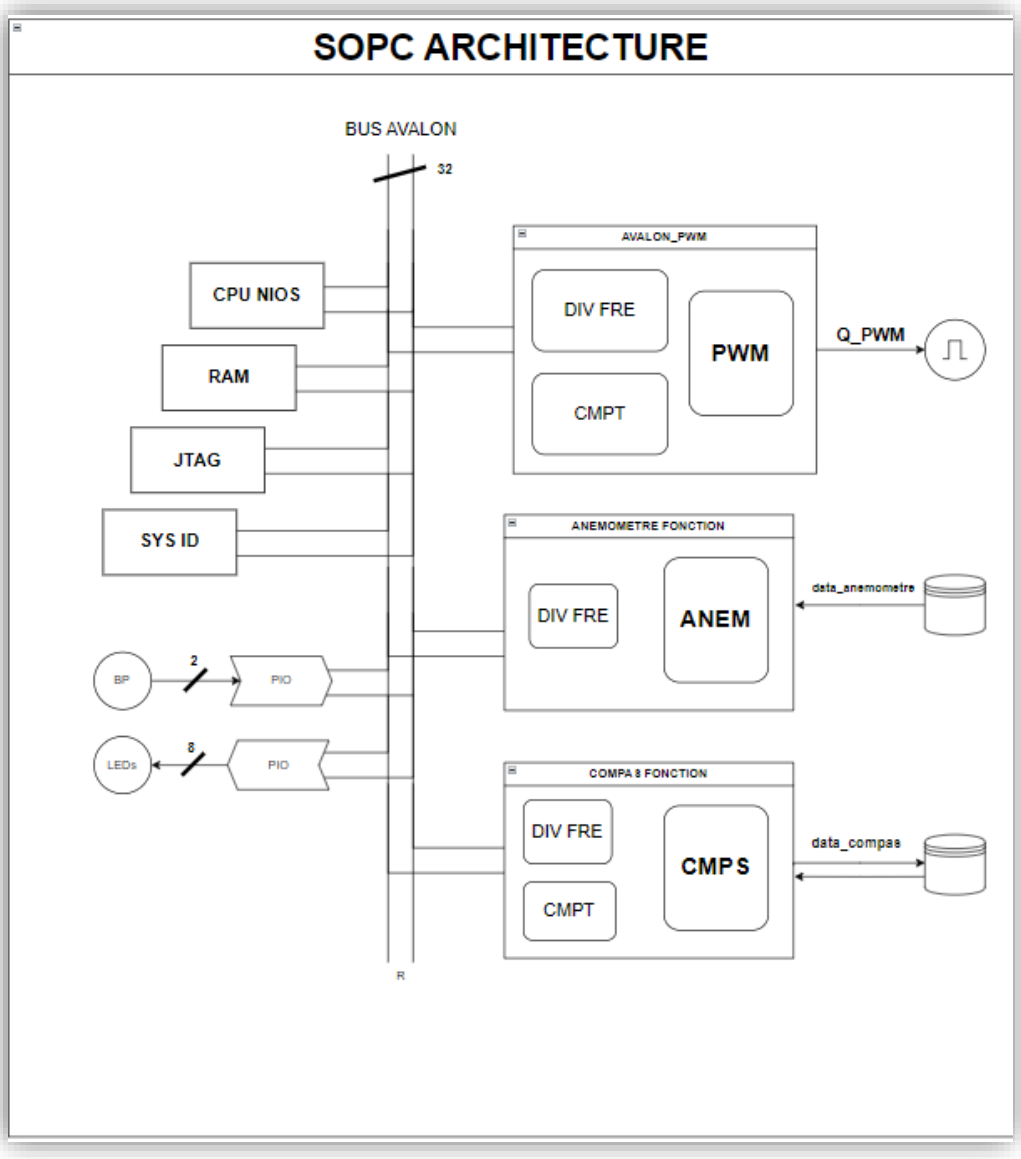


Figure 16 : L'architecture matérielle de SOPC

La création d'un composant compas et anémomètre est obligatoire pour les implémenter par la suite dans le **bus Avalon**. En passant maintenant aux fonctions principales de système : Le système de compas ou anémomètre (dans mon cas c'est le compas) et la fonction de vérin ou IHM (le vérin dans mon cas).

III) LA FONCTION DE COMPAS

La fonction du compas c'est l'une des fonctions qu'on va utiliser dans notre système de l'avalon avec l'anémomètre. Le compas gère à donner une valeur d'angle à suivre pour le système codé sur 9 bites car la valeur maximale de donnée c'est 360º, où la valeur maximale de 8 bits (pour un octet) est de 8 bits.

1) INTRODUCTION

Pour commencer il faut savoir la technologie qu'on utilise avec ce composant. Le CMPS03 détecte le champ magnétique terrestre pour définir un angle spécifique à suivre par le système ou le voilier dans notre cas.

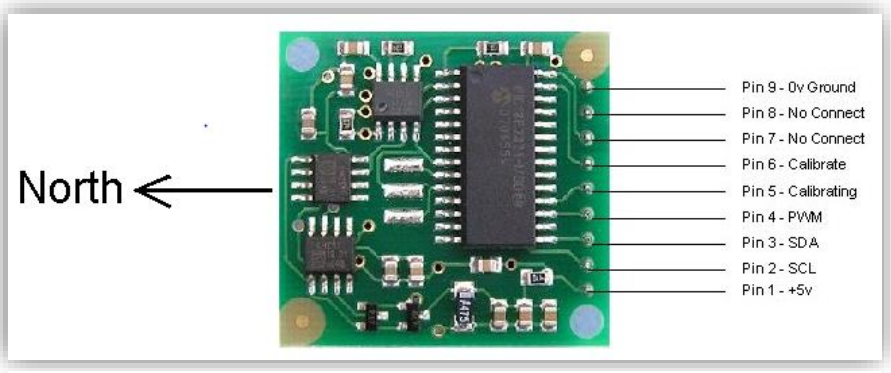


Figure 17 : Brochage du composant CMPS03

D'après la datasheet de composant on constate qu'on peut concevoir les données d'après deux types :

- Communication **I2C** : C'est un protocole qui généré des données du composant par un pin **SDA** en se basent sur une horloge pour la synchronisation des données sur un pin qui est **SCL.**

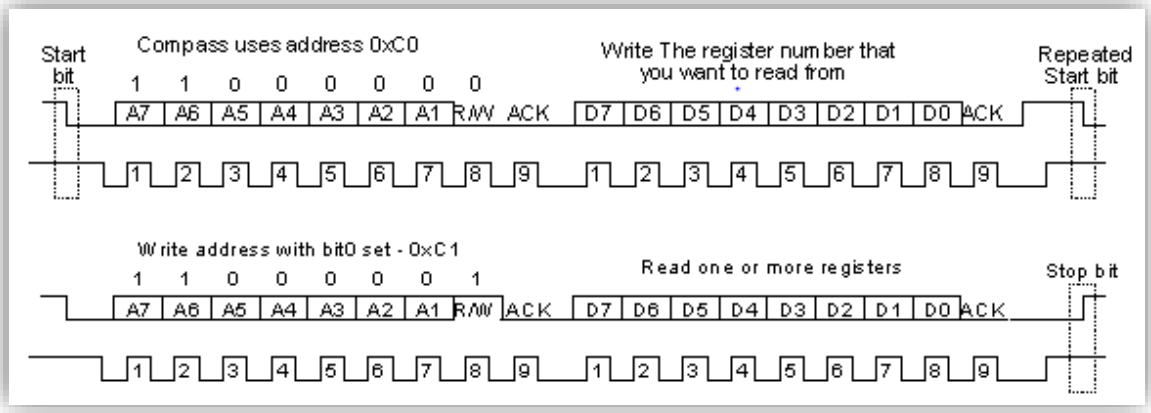


Figure 18 : Protocole de communication I2C pour le CMPS03

- L'impulsion de **PWM** (Pulse Width modulation): C'est un protocole qui varie la largeur de l'impulsion en parallèle de la fréquence ou la période.

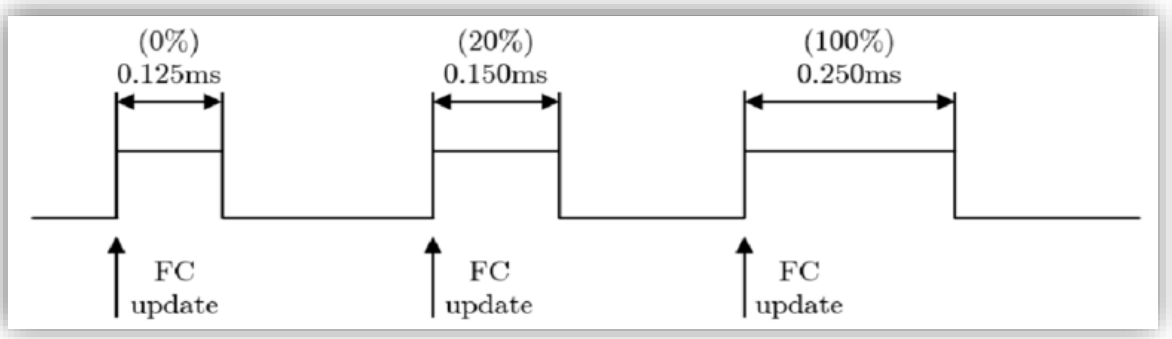


Figure 19 : Protocole de PWM

Dans notre cas on va utiliser la deuxième méthode de PWM pour avoir la valeur de l'angle qui sera détaillé par la suite.

2) LA MÉTHODE DE PWM

La méthode du PWM c'est parmi les protocoles les plus utilisé dans l'envoi d'information comme dans les moteurs par exemple.

Dans le CMPS03 existe un pin qui est le pin 4 de module du PWM ou l'acronyme de modulation de largeur d'impulsion, est un signal où la largeur positive de l'impulsion représente l'angle.

Cette largeur d'impulsion varie de 1 ms (0°) à 36,99 ms (359,9°), offrant une résolution de 100 us par degré, avec un décalage de +1 ms. Entre les impulsions, le signal devient faible pendant 65 ms, ce qui donne un temps de cycle total de 66 à 102 ms, en incluant la largeur d'impulsion. L'impulsion est générée par un temporisateur 16 bits dans le processeur, offrant une résolution fine de 1 microseconde. Il est cependant déconseillé de mesurer au-delà de 0,1 degré (10 us).

C'est bien de savoir comment le CMPS03 génère l'impulsion PWM pour synchroniser ces impulsions par rapport une période ou une fréquence adéquate avec a production de signal.

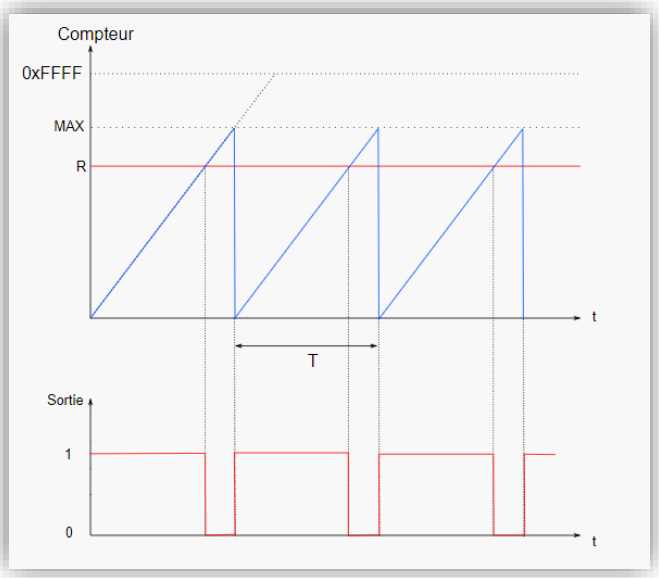


Figure 20 : Exemple d'un signal PWM par rapport un compteur

Pour bien comprendre, cette figure représente les valeurs d'angle pour le PWM par rapport la période T :

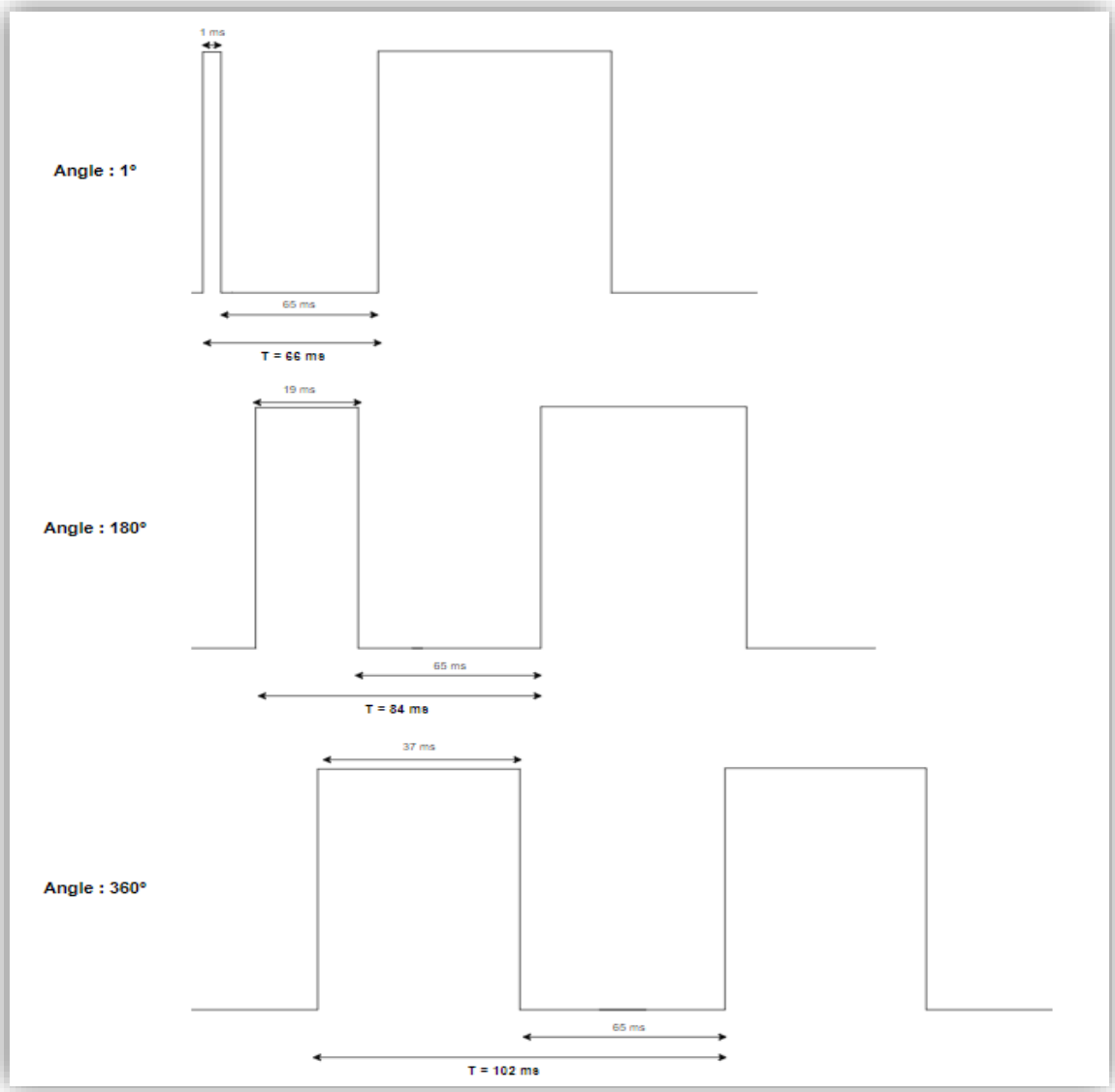


Figure 21 : Relation entre la valeur de l'angle et l'impulsion PWM

Cette figure représente la relation entre le PWM et l'angle de la direction de CMPS03, la largeur de l'impulsion change en fonction de l'angle d'une valeur de 1ms pour un angle de 1º et une période maximale de **66ms** et une valeur maximale de 37ms pour un angle et une période de **102ms**.

Ces valeurs seront importantes par la suite pour ajuster la bonne fréquence à utiliser dans les blocs de Quartus, et pour synchroniser avec d'autres données introduites dans le système général à concevoir.

3) SYSTÈME FONCTIONNEL DE COMPAS

Pour le système de compas, voici un schéma représentatif qui résume le comportement fonctionnel de composant :

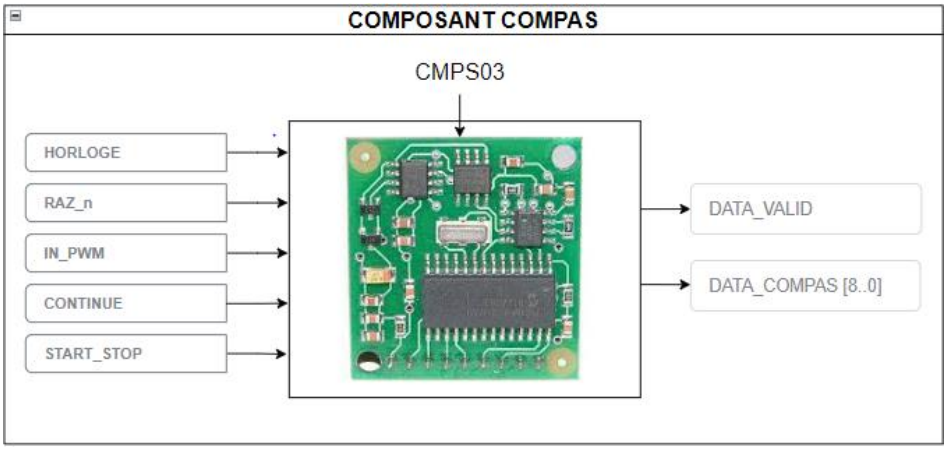


Figure 22 : Les entrées sorties de composant CMPS03

- **Horloge** : la fonction de compas va utiliser une horloge de 1Khz au lieu de 50Mhz car la fréquence de 1khz génère une période de signal de 1ms qui sera adéquate avec le signal de PWM sorti de module CMPS03.
- **Raz_n** : Le pin de reset dans l'entrée de système.
- **IN_PWM** : C'est la sortie de PWM de composant qui désigne l'angle de direction.
- **CONTINUE** : Bouton poussoir qui décrit le mode continu de système.
- **START_STOP** : Bouton poussoir qui décrit la marche ou arrêt de système.
- **DATA_VALID** : C'est le pin qui définit la validation des données.
- **DATA_COMPAS** : Une trame de 9 bits qui génère la donnée de l'angle de 0º à 360º.

Pour se concentrer dans la fonction générale de compas, il faut utiliser d'autres blocs ou sous-systèmes qui sont obligés d'après cette figure :

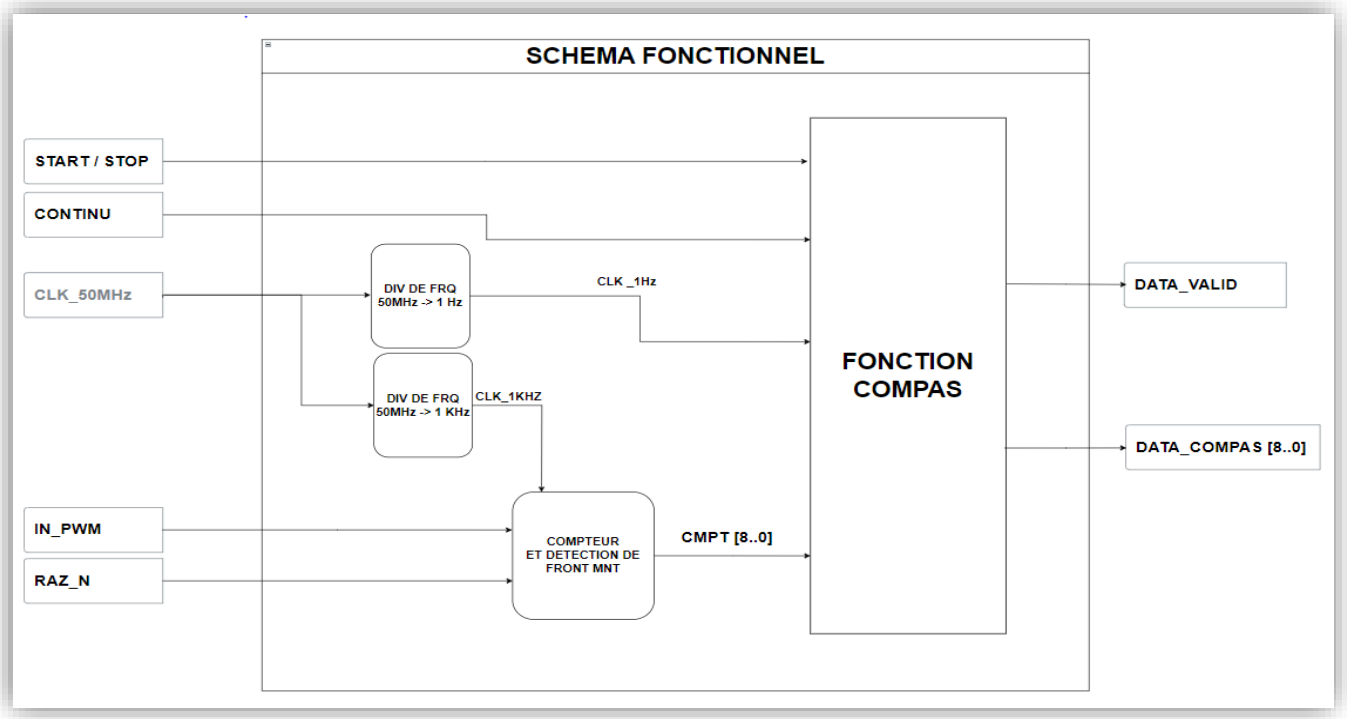


Figure 23 : Système fonctionnel de compas CMPS03

Pour chaque bloc de notre système, on doit écrire un code VHDL qui va générer le diagramme et ces blocs de diagramme y aura deux méthodes pour le regrouper, soit par le code ; écrire un code VHDL qui définit les différents composants avec l'affectation des entrées sorties entres les composants crées.

La deuxième méthode c'est créer les blocs des diagrammes des différents composant crées en VHD (clic droit sur le **code VHDL** de composant et **Create Symbol Files For Current File**)

J'ai utilisé la deuxième méthode qui nous donne un schéma fonctionnel de système avec les différents blocs utilisé :

- **Div_fre_1Hz** : C'est un diviseur de 1Hz par rapport l'horloge de 50Mhz de notre carte de développement DEO NANO, génère une période de 1 seconde pour le mode continu.
- **Div_fre_1kHz** : C'est un diviseur de 1kHz par rapport l'horloge de 50Mhz de notre carte de développement DEO NANO, qui va synchroniser l'entrée de PWM.
- **Detect_front** : C'est un bloc qui détecte le front montant des impulsions de PWM et synchroniser ces valeurs avec l'horloge de 1khz, avec le **reset** des valeurs si besoin et avoir une sortie de 9 bits qui code la valeur de l'angle de 0º à 360º.
- **Fct_compas** : Il s'agit de la fonction de compas qui va gérer le système en mode **continu**, avec un bit de **start_stop** qui faire marcher le système. Le système va avoir une sortie d'un bit pour valider les données et le mettre dans une trame de 9 bits **data_compas** qui sera la sortie de bloc de **Detect_front**.

Ce schéma représente le système de compas sur Quartus :

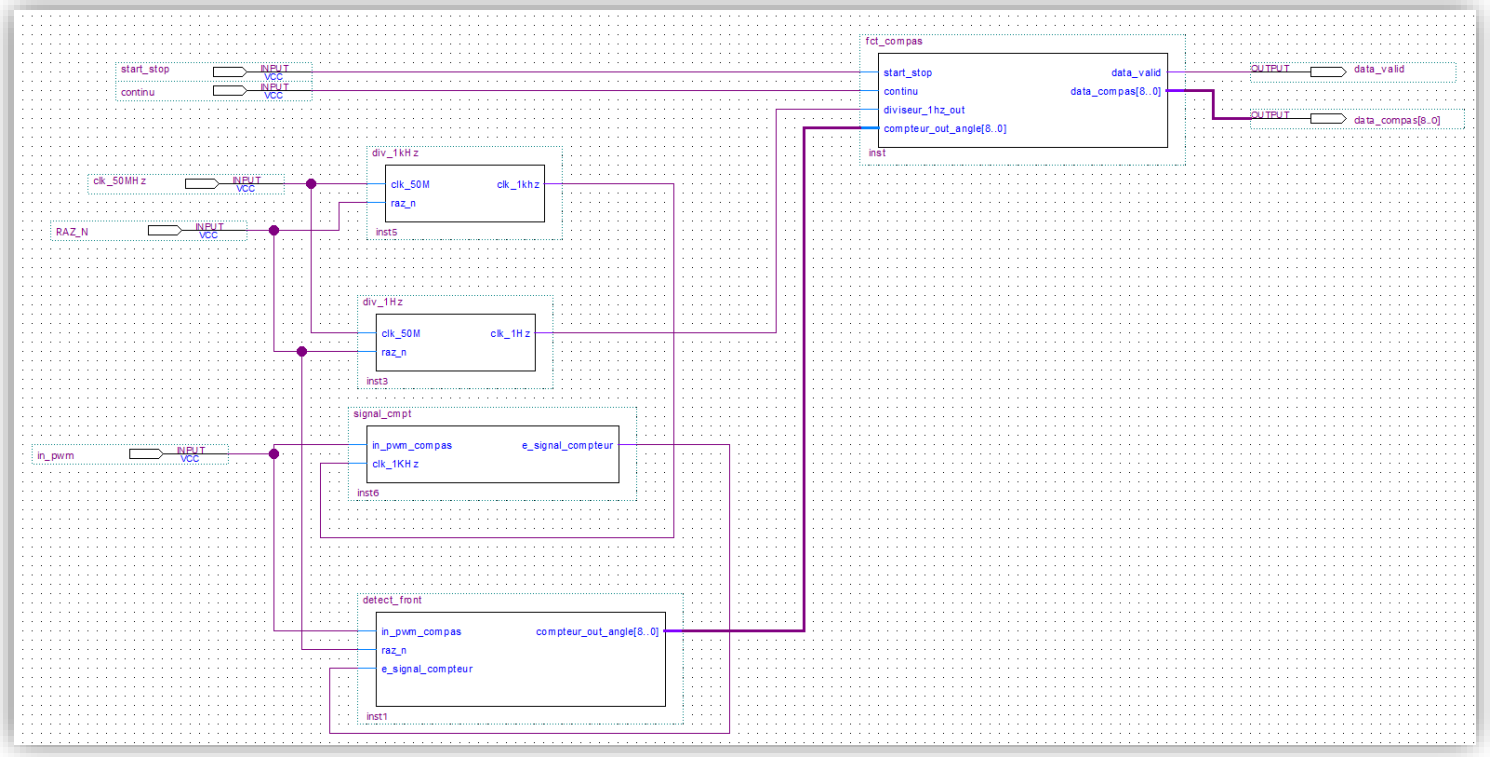


Figure 24 : Schéma fonctionnel de la fonction de compas

4) TEST DE SYSTÈME FONCTIONNEL DE COMPAS

Pour tester le système de compas, il existe plusieurs méthodes de test pour savoir si le système marche comme il faut ou pas :

- **GBF** : On peut utiliser le GBF pour générer un signal PWM avec une fréquence variable et aussi pour le Duty. La broche de PWM concerné est déjà assignée dans Quartus comme celle-ci de l'horloge.

Figure 25 : Générateur de basses fréquences GBF

- **MODELSIM** : C'est une plateforme de test qui est indépendant au Quartus mais fonctionnel pour les tests des systèmes pour ce type de fonctions. Pour tester le système de compas il nous faut créer un **Test Bench** pour tester et varier les valeurs de **fréquence** et de **DUTY**.

Dans cet exemple, j'ai mis une fréquence de **200 kHz** (valeur de Hexa = 0x40) et une DUTY de **50%** (valeur de Hexa = 0x20) :

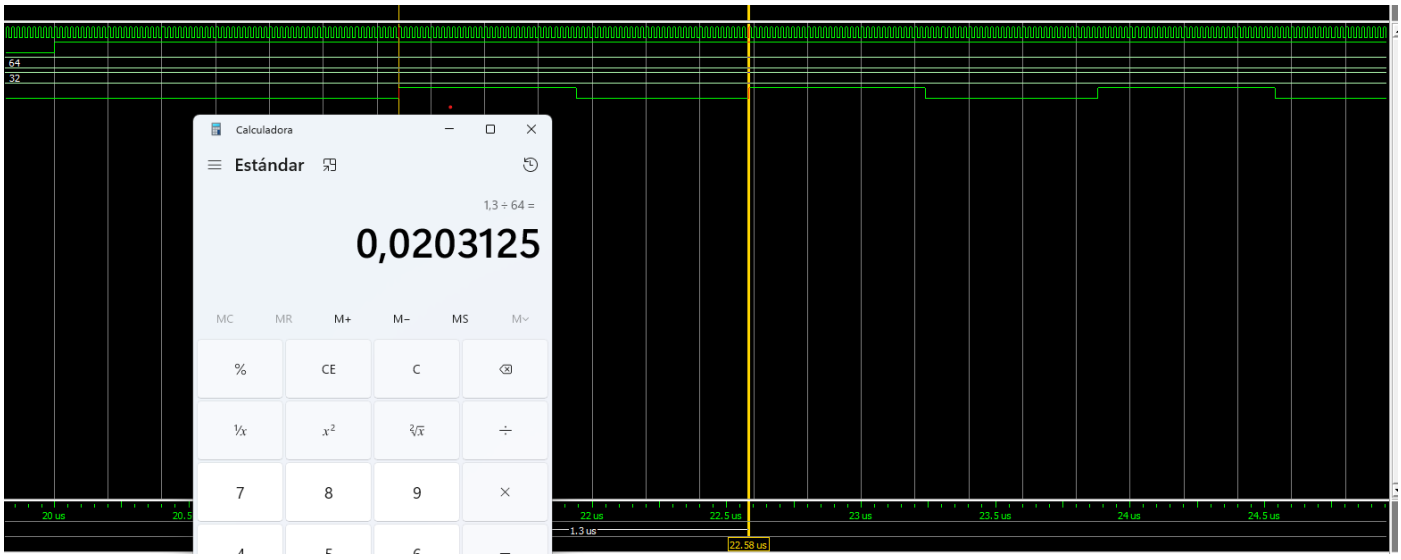


Figure 26 : Résultat de PWM avec une DUTY de 50%

- **AVALON_PWM** : Il y a autre méthode, on utilise l'avalon PWM qu'on a déjà crée et simulé avec Eclipse le code de test. Il suffit d'ajouter l'avalon PWM dans le schéma de fonction et liée la broche de sortie de out_pwm de l'avalon dans l'entrée de la fonction :

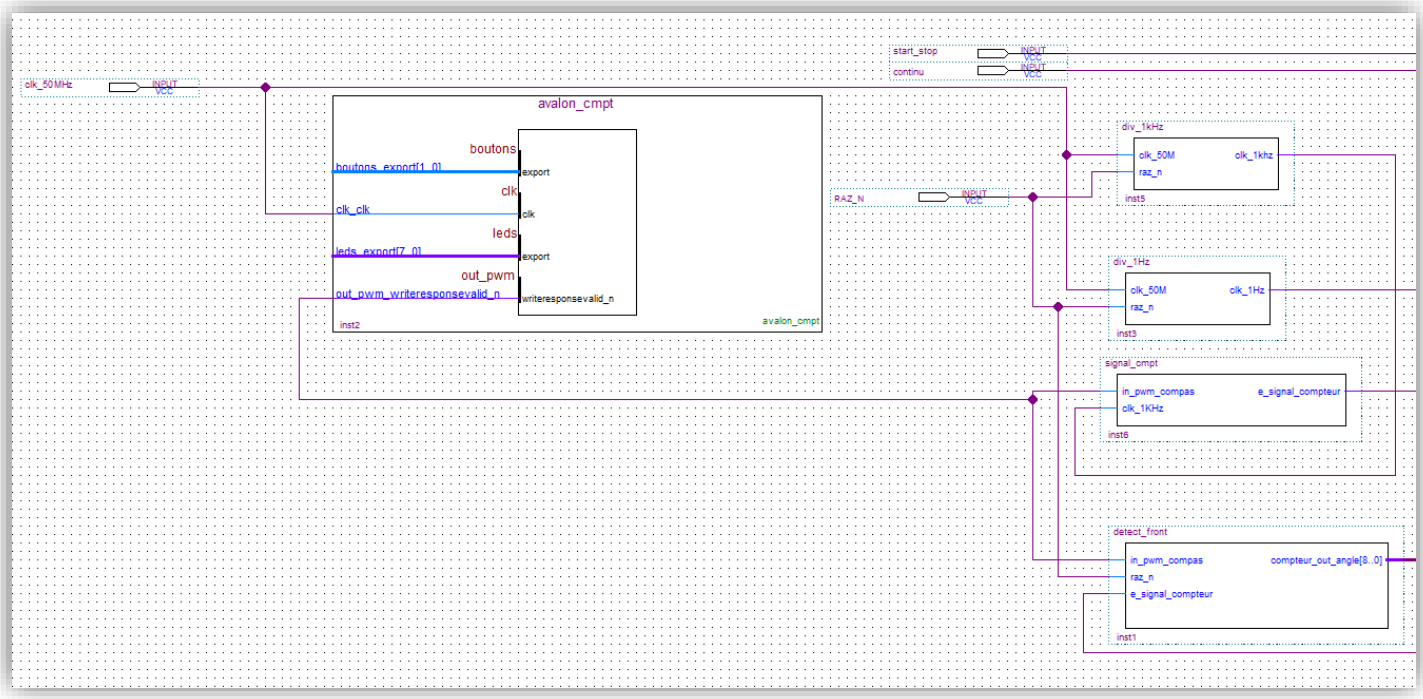


Figure 27 : Schéma général du système avec l'avalon PWM

Après cette partie, on passe vers NIOS Eclipse pour créer un nouveau programme BSP, après **Build Project** et après **Run As** et sélectionner **le NIOS II Hardware** et varier la **fréquence** et **DUTY**.

Comme déjà noté, la valeur de fréquence de la sortie de PWM est de **9,8 Hz** qui traduit la période de **102ms**. Le DUTY est de **37 %** qui traduit les **37ms** la valeur maximale de 360º de compas.

Ce qui traduit en **hexa** une valeur de **fréquence = 0x0066** et un **DUTY = 0x0025.**

```
*freq = 0x0066 ; // divise 102 avec l'horloge de 1khz/9.8 Hz
*duty = 0x0025 ; // RC = 37% -> 360º avec 100us/1º
*control = 0x0003;
```

Pour teste la valeur de degré de sortie ou les données on peux utiliser l'outil de Quartus **Signal Tap Logic Analyser** pour visualiser la trame de sortie. Il suffit d'ajuster la bonne horloge et introduire les signaux qu'on veut utiliser, dans notre cas sera que le pin de reset **Raz_n**, le pin **in_pwm** généré et la trame de sortie qui donne la valeur exacte de degré, dans notre test sera la valeur maximale de 360º.

Voici la figure qui représente le résultat de test :

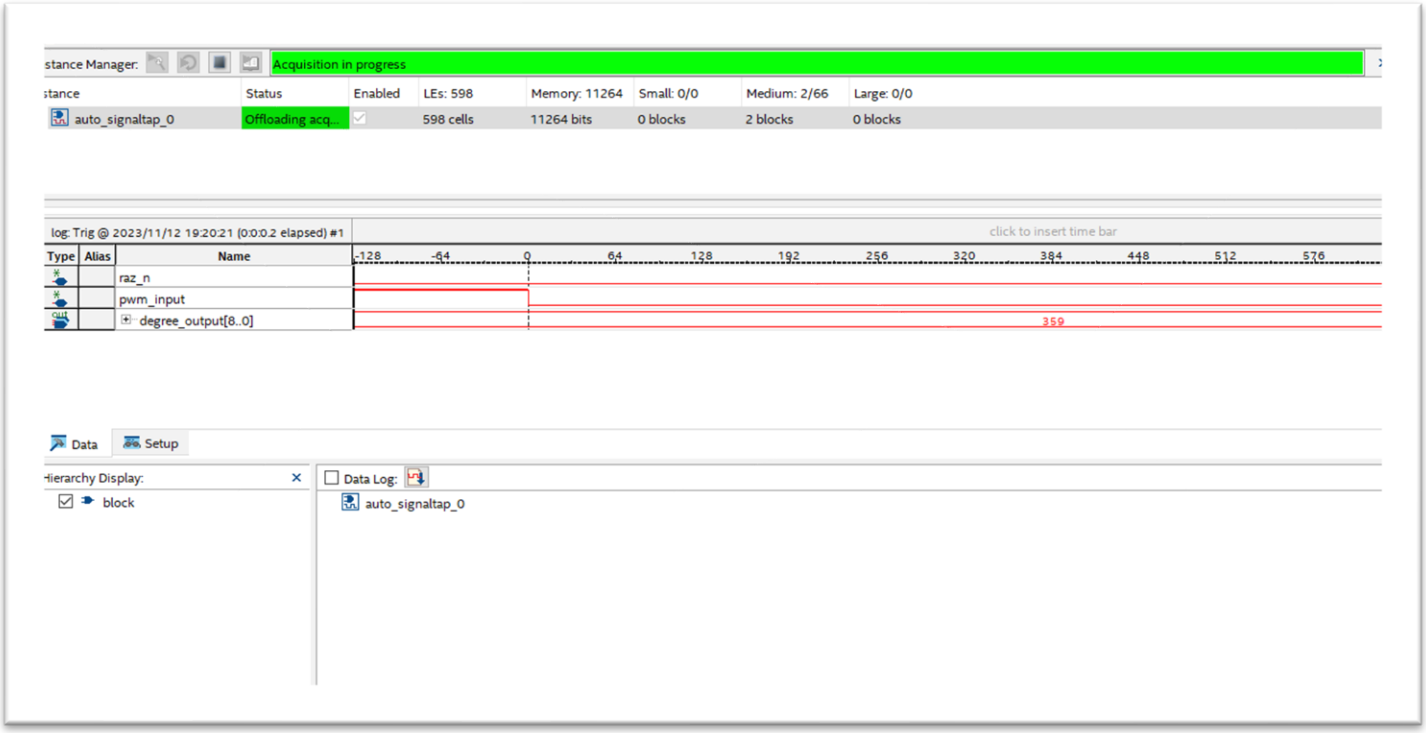
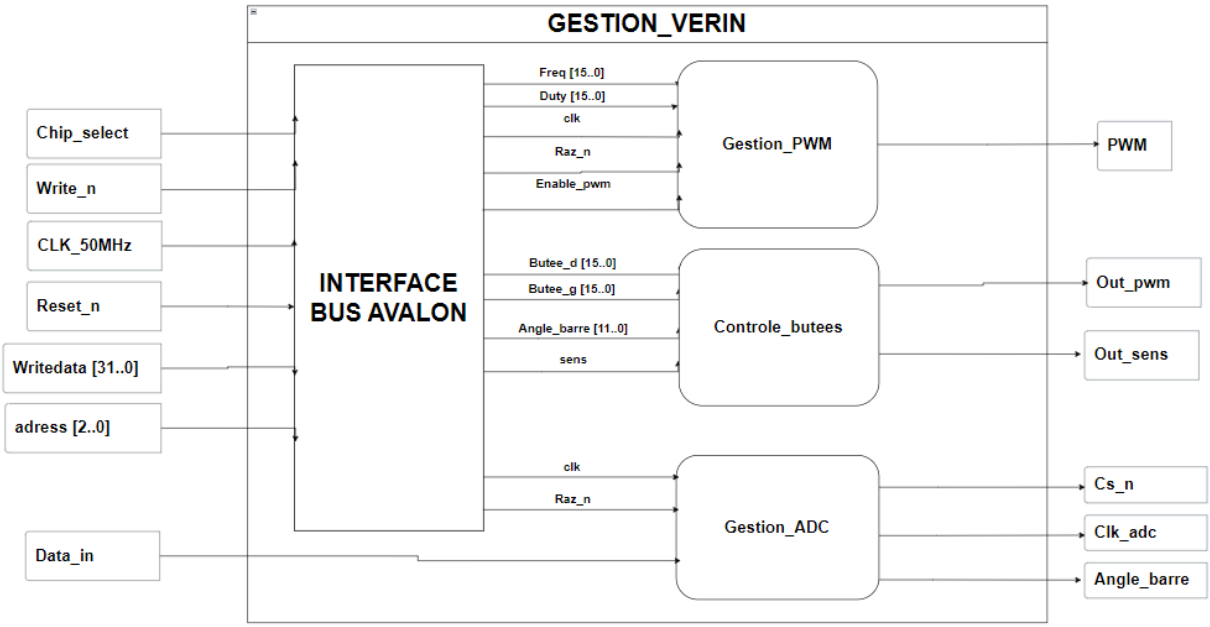


Figure 28 : Le résultat de test de visualisation de trama de sortie de degré

IV) IMPLÉMETATION DE VERIN



V) ASSEMBLAGE DE SYSTÈME