**Synthèse et mise en œuvre des systèmes**

**Bureau d’étude BE VHDL M2 SME**

Université Paul Sabatier Toulouse III

HADJ HASSEN Yosri & HAMICI Mohamed Anis



Table des matières

[Table des figures 2](#_Toc150709097)

[I. Introduction 3](#_Toc150709098)

[II. Analyse des besoins 4](#_Toc150709099)

[III. Réalisations du BE : Pilote de Barre Franche 5](#_Toc150709100)

[1. Mise en œuvre de la gestion du cap : 5](#_Toc150709101)

[2. Analyse fonctionnelle : 5](#_Toc150709102)

[3. Implémentation et simulation 6](#_Toc150709103)

[IV. Réalisations du BE : SOPC 1er version 7](#_Toc150709104)

[1. Mise en œuvre : 7](#_Toc150709105)

[2. Analyse fonctionnelle : 7](#_Toc150709106)

[3. Implémentation et simulation 7](#_Toc150709107)

[4. Tests et simulation de la fonction SOCP : 8](#_Toc150709108)

# Table des figures

[Figure 1: Présentation de pilote de barre franche. 3](#_Toc150709084)

[Figure 2: Les blocs de Pilote de barre franche 4](#_Toc150709085)

[Figure 3: Diagramme contexte du système 4](#_Toc150709086)

[Figure 4: diagramme du signal PWM Compas 5](#_Toc150709087)

[Figure 5: Analyse Fonctionnelle du compas 5](#_Toc150709088)

[Figure 6: Implémentation de la fonction Compas sur Quartus 6](#_Toc150709089)

[Figure 7: Simulation compas 6](#_Toc150709090)

[Figure 8: Analyse Fonctionnelle du SOPC 1er version 7](#_Toc150709091)

[Figure 9: la fonction sur SOPC Builder 7](#_Toc150709092)

[Figure 10: Code des registres de communications 8](#_Toc150709093)

[Figure 11: bloc SOPC sur Quartus 8](#_Toc150709094)

[Figure 12: Code SOPC test 9](#_Toc150709095)

[Figure 13: visualisation sur l'oscilloscope 9](#_Toc150709096)

# Introduction

Notre but principal, dans le cadre de notre unité d'enseignement "Synthèse et Mise en Œuvre de Systèmes", est de développer un pilote de barre franche sous la forme d'une puce programmable SOPC (System On Programmable Chip). Cela implique l'utilisation du langage VHDL pour décrire le système, basé sur une analyse détaillée des spécifications et une structuration fonctionnelle du système. Nous allons concevoir des circuits d'interfaces numériques en VHDL pour le simuler et le valider sur une maquette. Ensuite, nous établirons des connexions avec les bus de microprocesseurs tels que NIOS, Altéra et Avalon pour vérifier la fonctionnalité du SOPC par des manipulations spécifiques.

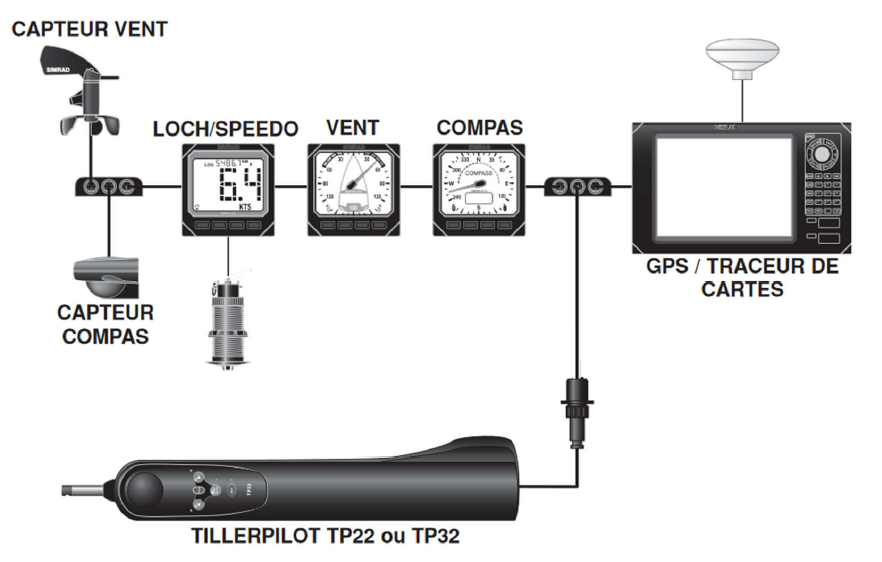


Figure 1: Présentation de pilote de barre franche.

Un pilote automatique pour voilier est un dispositif électrique ou hydraulique conçu pour maintenir le cap d'un voilier sans intervention humaine. Il s'avère d'une grande utilité pour les navigateurs solitaires ou en équipage réduit. Ce pilote se compose essentiellement de trois éléments : un compas, une unité électronique et une unité de puissance. Dans les versions récentes, le compas, électronique, transmet en continu au système de traitement le cap actuel du bateau. L'unité de traitement, quant à elle, définit le cap à suivre. En permanence, elle compare ces deux caps et si une différence est détectée, elle ordonne à l'unité de puissance d'intervenir sur la barre pour réaligner le bateau sur son cap désiré.

Pour les pilotes de barre franche, l'unité de puissance est souvent un vérin linéaire. Ce vérin, installé entre le banc du cockpit et la barre, réagit aux variations de cap en agissant sur la barre en conséquence.

Ce projet d'étude porte précisément sur la conception d'un pilote de barre franche pour un voilier, visant à automatiser sa navigation. Ce système englobe une commande de pilotage équipée d'un vérin. Les commandes de cette unité permettent de contrôler l'extension et la rétractation du vérin, influençant ainsi la barre franche du voilier. Ces commandes permettent également de choisir différents modes de navigation pour le voilier. En parallèle, ce système est connecté à une boussole (compas), un GPS avec une interface NMEA, une girouette pour déterminer la direction du vent et un anémomètre pour mesurer sa vitesse.

Toutes ces données seront utilisées pour calculer la trajectoire du voilier et lui permettre de naviguer selon divers modes tels que le Conservateur d'Allure, le mode Automatique, et bien d'autres.

Le système à réaliser est divisé en sous-systèmes, représenté dans la figure ci-dessous :

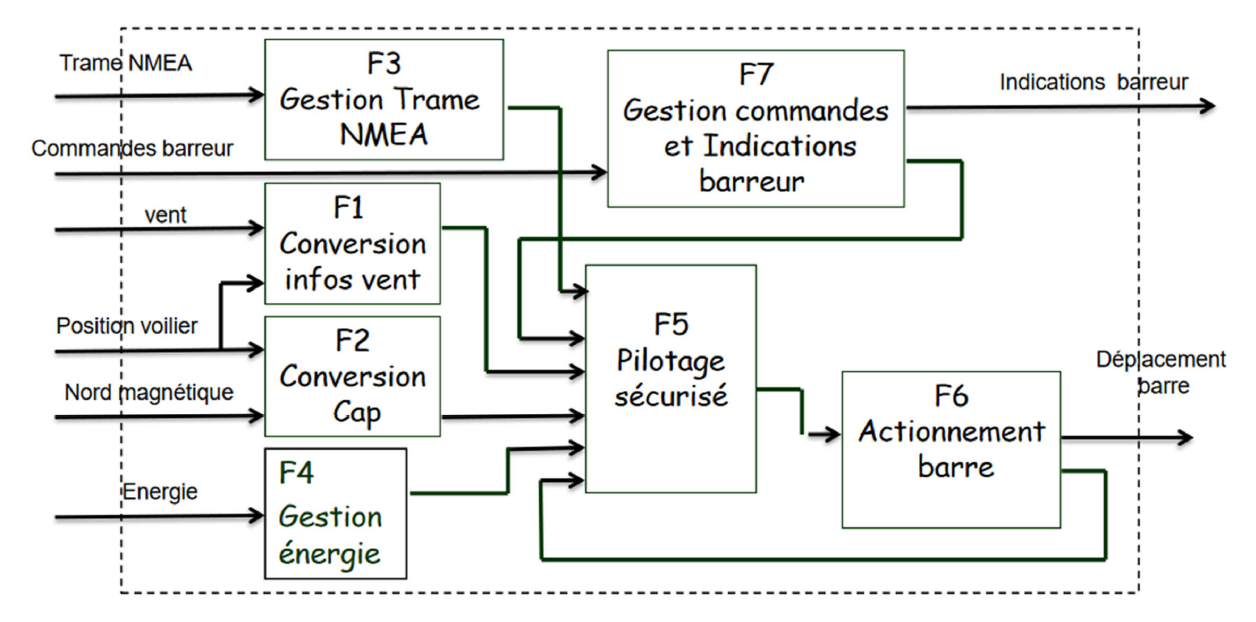


Figure 2: Les blocs de Pilote de barre franche

# Analyse des besoins

Pour répondre aux besoins de la barre franche, le système à développer comprendra différentes fonctions et flux d'informations essentiels pour son fonctionnement.

Il comportera :

- Une fonction dédiée à la lecture de la vitesse du vent (0-250 km/h), en interprétant la sortie logique de fréquence variable (0 à 250 Hz) de l'anémomètre.

- Une fonction de génération de signal PWM, intégrée ultérieurement dans le SOPC du FPGA, pour produire un signal utilisable via le Bus Avalon.

- Une fonction de gestion du vérin, responsable du contrôle de la barre franche.

- Une fonction exploitant un compas pour obtenir les mesures d'angle horizontales du voilier, déterminant ainsi sa direction.

- Une fonction de gestion de l'interface Homme-Système, comprenant divers boutons (Bâbord, Tribord, Auto/Manuel), des LED et un buzzer.

- L'intégration d'un MCU (Microcontrôleur) dans le FPGA via l'outil SOPC d'Altera. Ce MCU assurera le traitement et l'affichage des différentes variables du projet telles que le cap du voilier, la position du vérin, les données GPS, la gestion du signal PWM et la vitesse du vent.



Figure 3: Diagramme contexte du système

# Réalisations du BE : Pilote de Barre Franche

## Mise en œuvre de la gestion du cap :

Ce module de boussole a été spécialement conçu pour être utilisé dans les robots comme aide à la navigation. L'objectif était de produire un nombre unique pour représenter la direction vers laquelle le robot est orienté. La boussole utilise le capteur de champ magnétique Philips KMZ51, qui est suffisamment sensible pour détecter le champ magnétique terrestre.

Nous nous intéressons à la sortie donnée par ce module sous la forme d'un signal PWM dont la largeur positive de l'impulsion représente l'angle. La largeur de l'impulsion varie de 1mS (0°) à 36,99mS (359,9°) - soit 100uS/° avec un décalage de +1mS. Le signal est en état bas pendant 65 ms entre les impulsions, de sorte que le temps de cycle est de 65 ms + la largeur de l'impulsion.

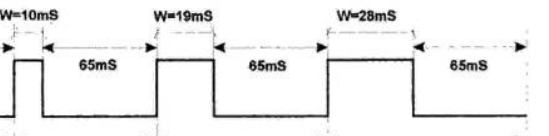


Figure 4: diagramme du signal PWM Compas

## Analyse fonctionnelle :

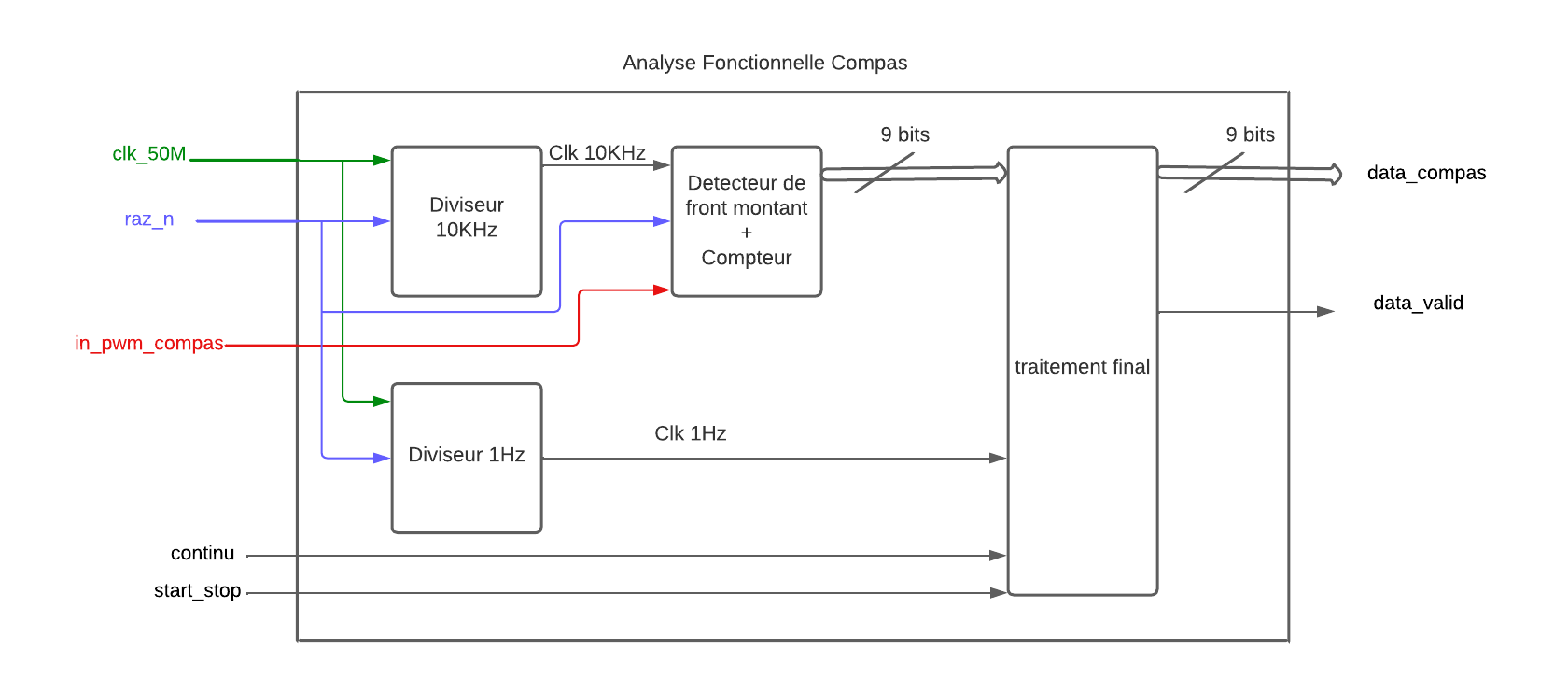


Figure 5: Analyse Fonctionnelle du compas

**Diviseur 10Khz** : Ce diviseur prend l'horloge 50Mhz native de nos cartes DE2/DE0 NANO et la divise pour créer une sortie d'horloge 10Khz, ce qui signifie que la période produite est de 100us, ce qui est conforme aux 100us/degré donnés par le compas, ensuite nous utilisons ce diviseur pour incrémenter notre compteur d'état haut pour obtenir sa durée dans les unités commodes.

**Diviseur 1Hz** : Ce diviseur prend l'horloge 50Mhz native de nos cartes DE2/DE0 NANO et la divise pour créer une sortie d’horloge 1 Hz donc 1 seconde. Il est utilisé pour mettre en œuvre le mode continu qui génère des mesures toutes les secondes.

**Détecteur de front montant/Compteur** : Ce bloc contient 2 fonctionnalités

1- Détecteur de front montant : pour repérer le moment où notre impulsion démarre et initier le comptage, puis arrêter et mémoriser la valeur du comptage lorsque le signal passe au niveau bas.

2- le compteur démarre lorsque nous détectons un front haut, il s'incrémente tous les cycles d'horloge de 10 KHz, c'est à dire tous les 100us, ce qui signifie +1 degré à chaque fois que le compteur s'incrémente. Enfin cela donne une sortie (la valeur du comptage) codée sur 9 bits.

**Traitement final :** Ce bloc contient la logique de la fonction attendue pour la boussole, l'initiation du mode continu, la vérification de data\_valid et la logique de start\_stop.

## Implémentation et simulation

Après avoir effectué l'analyse fonctionnelle de notre circuit, nous avons procédé à son implémentation dans Quartus en utilisant des blocs en VHDL et en suivant le code VHDL tel que représenté dans la figure ci-dessous :

Figure 6: Implémentation de la fonction Compas sur Quartus

L'échelle de temps a été réduite pour que le simulateur soit fiable sans prendre trop de temps, afin de garantir un fonctionnement correct.

Ensuite nous avons utilisé notre carte DE2, nous avons attribué des pins à nos entrées et des leds sur les 9 bits de la sortie, généré un signal sur notre GBF avec offset pour avoir 0 à +5v pour nos états on off, 65ms + 20 ms = > résultat lu sur les leds 200 degrés

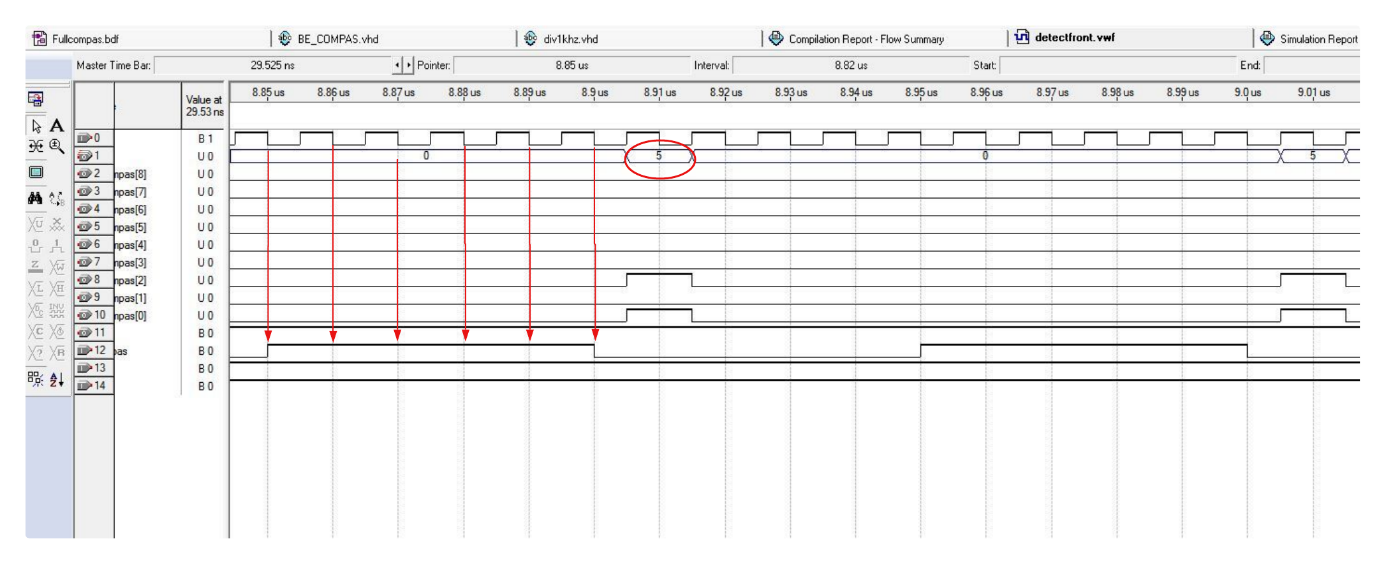


Figure 7: Simulation compas

# Réalisations du BE : SOPC 1er version

## Mise en œuvre :

Le microcontrôleur conçu est basé sur le cœur Nios II, en suivant les spécifications du cahier des charges. Avec une mémoire RAM on-chip de 20 ko ajoutés. Deux PIO (Parallel Inputs Outputs) sont intégrés, l'un pour les boutons en entrée (2 bits) et l'autre pour les LEDs en sortie (8 bits). Un composant JTAG UART est inclus pour la communication avec le PC hôte et le débogage du programme, avec une assignation d'interruption. Un composant SysId est ajouté pour attribuer un numéro d'identification au système, renforçant la sécurité. Enfin, en générant le fichier VHDL, le SOPC (System on a Programmable Chip) est créé avec toutes les configurations définies

## Analyse fonctionnelle :

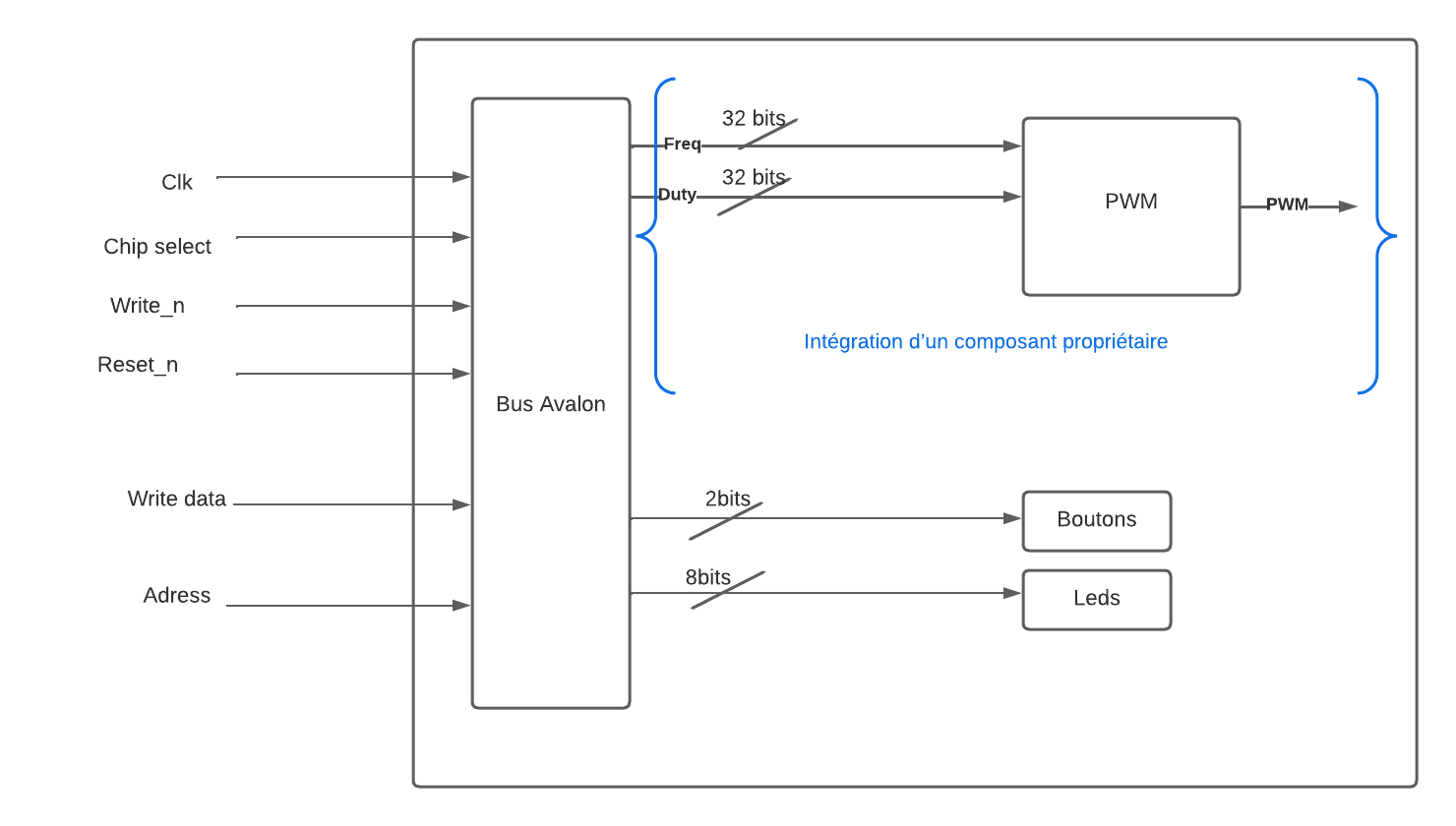


Figure 8: Analyse Fonctionnelle du SOPC 1er version

## Implémentation et simulation

Pour transférer le circuit sur la carte, nous avons employé le SOPC Builder pour élaborer le microprocesseur et les divers périphériques en intégrant l'Avalon PWM, conformément aux indications figurant dans les schémas ci-dessous. Cela nous a permis de tester notre SOPC et de préparer l'intégration du même modèle dans les fonctions à réaliser au cours de notre BE

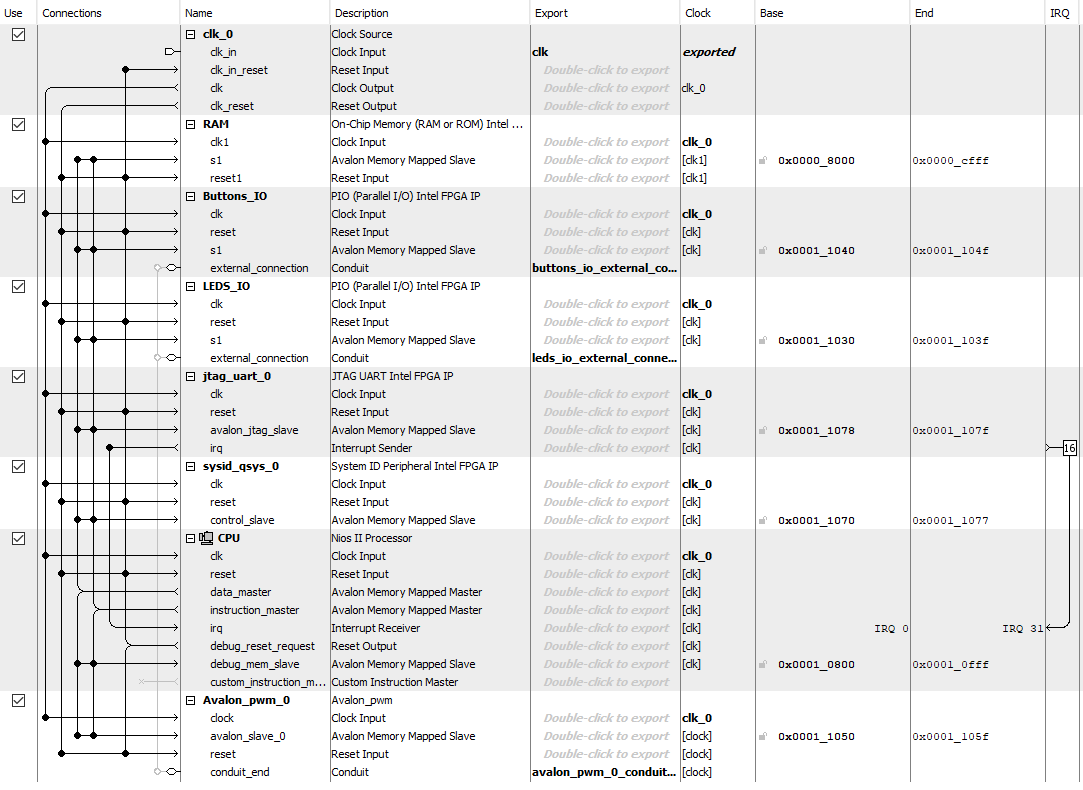


Figure 9: la fonction sur SOPC Builder

Après avoir intégré le code et la logique pour le PWM avec 32 bits de service et de fréquence, nous avons besoin d'un code des registres pour établir la communication et récupérer les valeurs, nous utilisons :

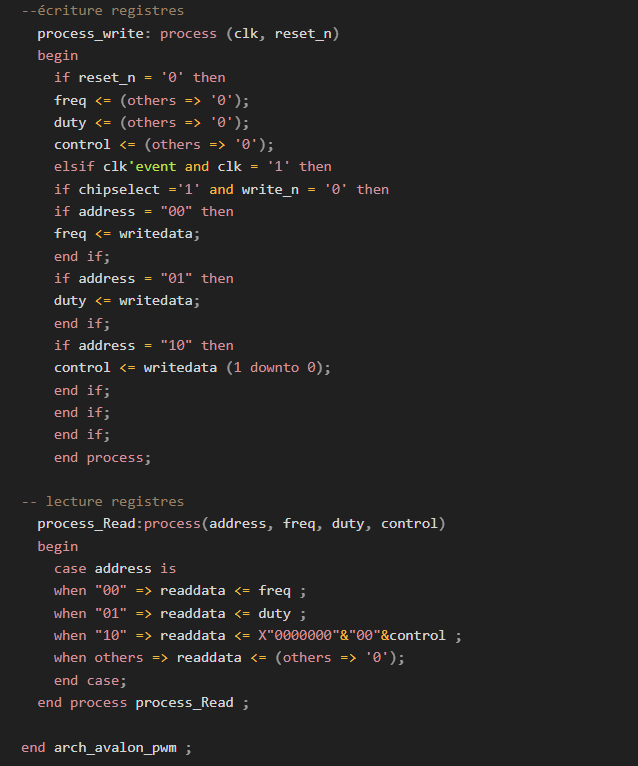


Figure 10: Code des registres de communications

Une fois notre système assemblé via SOPC Builder, avec l'intégration de l'Avalon PWM, des boutons et des LEDs, nous avons généré l'interface présentée dans la figure ci-dessous. Celle-ci représente le bloc SOPC actuel qui évoluera au fil de notre BE pour intégrer les fonctionnalités spécifiques à la Barre-Franche.

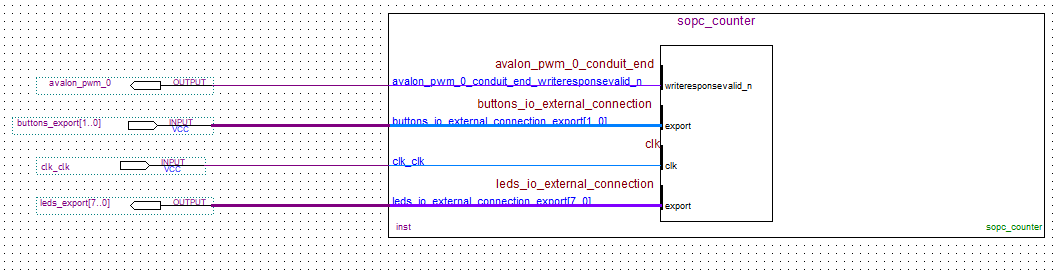


Figure 11: bloc SOPC sur Quartus

## Tests et simulation de la fonction SOCP :

Notre objectif avec le code est de diviser la fréquence interne de 50 MHz par 1000, tout en maintenant un rapport cyclique de 50%. Après compilation et exécution du code sur notre SOPC, nous anticipons des résultats de 50 MHz/1024 avec un rapport cyclique de 50%, conformes à ce qui est affiché dans la visualisation sur un oscilloscope. Ces résultats correspondent aux attentes fixées.

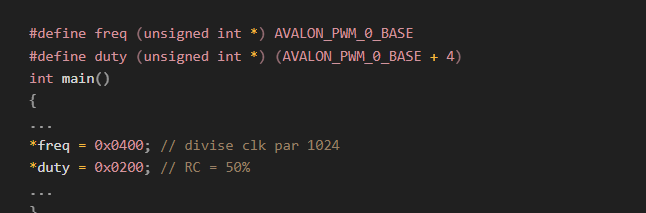


Figure 12: Code SOPC test

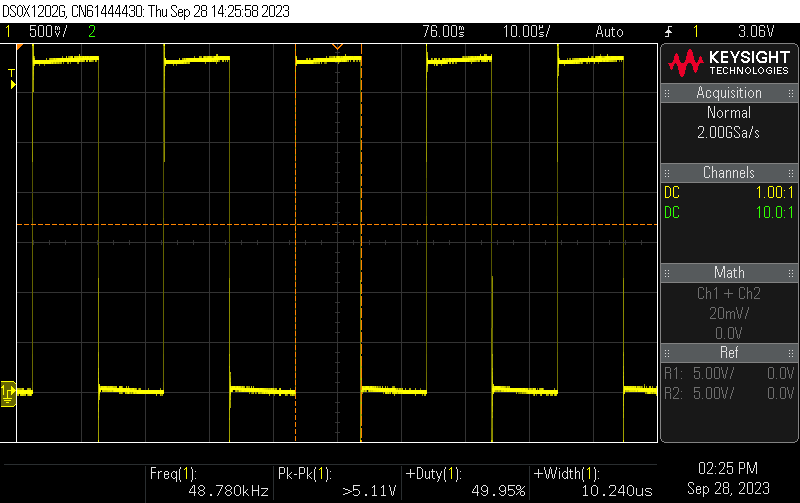


Figure 13: visualisation sur l'oscilloscope