

# Université Paul Sabatier

---

## Master Systèmes & Microsystèmes Embarqués

### Bureau d'étude : Pilote de barre franche



***Réalisé par :***

- Abderrahmane AMOUR
- Najlae ELMAAZOUZ

***Encadré par :***

Mr. Thierry PERISSE

*Année Universitaire : 2023 – 2024*

## Sommaire :

I. Introduction .....	3
II. Cahier des charges du projet «Barre-Franche ».....	3
III. Conception Matérielle .....	4
III.1 Carte DE0 – Nano .....	4
III.2 Pilote de barre Franche analyse et spécification .....	4
III.3 MISE EN ŒUVRE DE SOPC .....	5
III.4 Platform designer .....	6
III.5 Partie software .....	7
IV. Conception d’une fonction simple : l’anémomètre.....	8
IV.1 Analyse fonctionnelle.....	8
IV.2 Implantation.....	11
IV.3 Validation de la fonction anémomètre .....	12
IV.4 MISE EN ŒUVRE DE SOPC «Anémomètre ».....	12
IV.5 Partie software NOIS II .....	14
V. Conception d’une fonction complexe : Gestion Vérin .....	15
V.1 Analyse fonctionnelle .....	15
V.2 Implantation .....	16
V.3 Simulation et validation .....	16
V.4 MISE EN ŒUVRE DE SOPC « Vérin » .....	16
VI. Conclusion.....	19

## I.Introduction :

Dans le cadre de notre formation M2 Systèmes et microsystèmes embarqués, nous avons participé à un Bureau d'études pour synthétiser et mettre en œuvre des systèmes, simples et complexes. Le but de ce bureau d'études était de mettre en œuvre une solution de synthèse, matérielle et logicielle pour piloter automatiquement un voilier de barre franche d'un bateau. Pour réaliser ce système, nous avons commencé par prendre en main l'utilisation du logiciel Quartus et le langage VHDL, grâce aux TPS de base proposés par notre encadrant. Ensuite, nous nous sommes intéressés à analyser les besoins et les spécifications que le système doit satisfaire, et à faire un découpage fonctionnel et logiciel du système en utilisant un algorithme et du code en VHDL, pour l'implémenter sur la carte DE0 (Cyclone IV)

Le cahier des charges du projet "Barre-Franche" porte sur la réalisation d'un dispositif embarqué basé sur un FPGA de chez Altera : le DE0 nano. Nous devons développer les deux parties matérielles et logicielles afin de mettre en œuvre l'asservissement d'une barre franche. Le lien entre les deux parties sera établi par l'implémentation d'un bus de données : le bus Avalon. Pour fournir un système complètement fonctionnel, différents blocs fonctionnels devront être implémentés sur le FPGA. Ce rapport présente les différentes étapes de la réalisation de ce projet, ainsi que les résultats obtenus.

## II.Cahier des charges du projet «Barre-Franche »

Ce projet a pour objectif de créer un dispositif embarqué utilisant un FPGA d'Altera, plus précisément le modèle DE0 nano. Il nécessite le développement à la fois de composants matériels et logiciels pour contrôler une barre franche. La section matérielle reposera sur un code VHDL exécuté sur le FPGA, tandis que la section logicielle sera développée en C. La communication entre ces deux sections sera assurée par l'intégration d'un bus de données, nommé bus Avalon.

Pour assurer la pleine fonctionnalité du système, divers blocs fonctionnels devront être implémentés sur le FPGA :

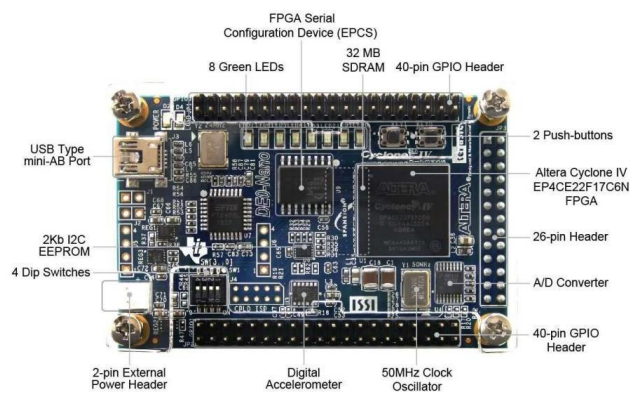
- Un bloc qui permettra de lire la mesure de la vitesse du vent (0-250km/h). Il devra lire la sortie de l'anémomètre qui est une sortie logique de fréquence variable (0 à 250 Hz).
- Un bloc générant un signal PWM qui sera utilisé dans plusieurs parties du projet. Il sera intégré plus tard dans le SOPC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

### III. Conception Matérielle :

#### III.1 Carte DE0 – Nano :

La DE0-Nano est une plateforme de développement FPGA compacte et efficace, idéale pour la création de prototypes de circuits, notamment pour les robots et les projets mobiles. Conçue pour une mise en œuvre simplifiée, elle cible le circuit Cyclone IV, doté de jusqu'à 22 320 éléments logiques. Les atouts majeurs de la DE0-Nano résident dans sa petite taille et son faible poids, ainsi que dans sa capacité à être reconfigurée facilement, sans nécessiter d'équipement additionnel.

Ces aspects la différencient des cartes de développement plus génériques. En outre, elle offre un accès ouvert à tous les fichiers de conception pour le microcontrôleur Propeller à 8 noyaux de Parallax.



#### III.2 Pilote de barre Franche analyse et spécification :

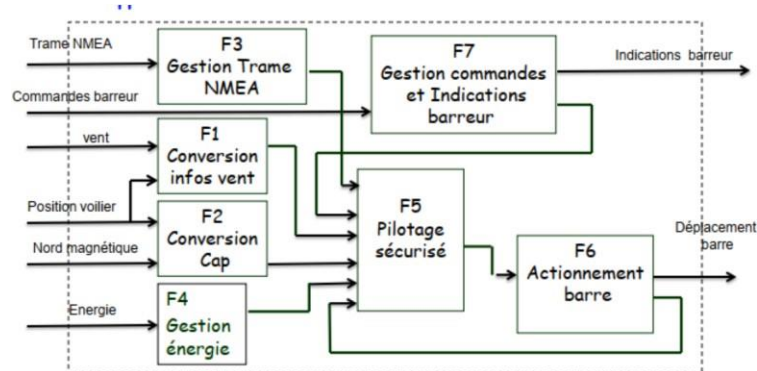
Un pilote de barre franche est un dispositif utilisé pour piloter un bateau et conserver sa direction. Il comprend un compas, chargé de collecter les informations relatives à la direction du vent ou de la girouette.

Ce compas produit une différence de potentiel correspondant à l'écart de la route détecté. Le système inclut également une composante électronique qui travaille en tandem avec le compas, effectuant une comparaison continue entre la direction indiquée par le compas et le cap souhaité. Lorsque le bateau dévie de son itinéraire prévu, cette composante électronique active un vérin, qui fait partie du système de puissance, pour corriger la trajectoire.

Le cahier des charges pour ce système spécifie plusieurs fonctions et exigences clés, qui sont énumérées ci-dessous :

- Mesure de la Fréquence et Détection des Données de Vitesse du Vent : Le système doit pouvoir mesurer la fréquence et détecter les informations relatives à la vitesse du vent, fournies par l'anémomètre.

- Collecte des Informations de Trajectoire Réelle de la Boussole : Il doit récupérer les données de trajectoire réelle qui sont transmises par une boussole.
- Acquisition de la Trajectoire Réelle via GPS : Le système doit être capable de récupérer la trajectoire réelle à partir des données fournies par un GPS.
- Interaction avec l'Utilisateur : Le système doit interagir avec l'utilisateur au moyen de LEDs, de boutons et de signaux sonores.
- Commande du Vérin : Il doit pouvoir commander le vérin, qui est un élément clé pour ajuster la direction du bateau.
- Détection de l'Écart d'Angle de Déviation avec la Girouette : Enfin, le système doit être capable de détecter et de récupérer l'écart d'angle de déviation par rapport à la direction indiquée par la girouette.



Il est évident que nous sommes confrontés à un système complexe. En conséquence, il nous a été demandé de nous concentrer sur deux fonctions spécifiques : une fonction simple et une autre plus complexe. Nous avons sélectionné la fonction de l'anémomètre comme notre fonction simple, tandis que la gestion du vérin a été identifiée comme la fonction complexe sur laquelle nous allons travailler.

### III.3 MISE EN ŒUVRE DE SOPC :

L'objectif de cette configuration est de concevoir un système ou un dispositif programmable qui va reproduire les interconnexions entre les divers éléments du SOPC tels que le CPU, la RAM, le JTAG, le SYS ID, les PIO, etc. Ce système intégrera également d'autres composants externes à concevoir, tels que l'Avalon PWM, un compas, un anémomètre, parmi d'autres.

Pour initier la construction de notre SOPC, nous allons élaborer un Avalon PWM. Ce module est destiné à produire un signal de modulation de largeur d'impulsion (PWM) dont nous pouvons régler la fréquence et la durée d'impulsion. Ce signal sera ensuite intégré dans le système NIOS, simultanément avec le développement d'un compteur.

### III.4 Platform designer :

Pour accéder à la configuration matérielle de notre Système sur Puce (SOPC) dans Quartus, nous ouvrons la fonction PLATFORM DESIGNER via le menu Outils. Notre SOPC intégrera les composants matériels suivants :

**CPU:** Utilisation du processeur NIOS II Processor en mode économique.

**RAM:** Mise en place d'une RAM On-Chip Memory Intel avec une capacité de 20k.

**JTAG:** Implémentation d'un JTAG UART Intel pour la programmation de PORT, configuré avec une priorité d'interruption fixée à 16.

**SYS ID:** Intégration du System ID Peripheral Intel pour l'identification des composants.

**PIO:** Configuration des entrées/sorties du SOPC, incluant deux boutons d'entrée et huit LEDs en sortie pour le compteur.

Pour l'ajout du composant Avalon PWM, procédez comme suit :

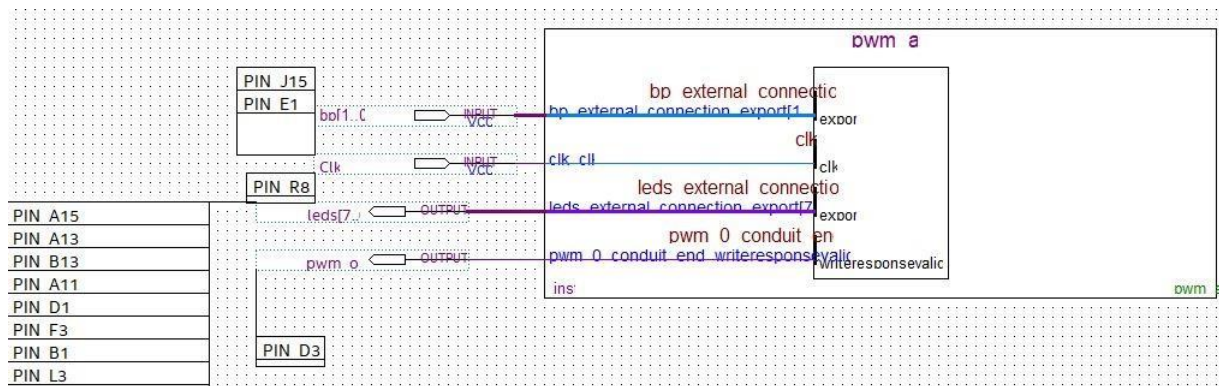
- Nommez le composant "avalon\_pwm".
- Dans la section "Files", ajoutez le fichier de code VHDL préalablement créé dans le dossier du projet et procédez à l'analyse des fichiers.
- Dans "Signals & Interfaces", ajoutez une interface de pilotage avec le signal de sortie "out\_pwm".

Après la configuration de tous les composants du SOPC, il est nécessaire de les interconnecter. Cela se fait en cliquant sur les points blancs à gauche de l'interface. Assurez-vous également d'ajuster la RAM dans le CPU, spécifiquement dans les champs "reset vector" et "exception vector".

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcode Name
✓		<b>clk_0</b>	Clock Source							
		clk_in	Clock Input	clk	exported					
		clk_in_reset	Reset Input	Double-click to export						
		clk	Clock Output	Double-click to export	clk_0					
		clk_reset	Reset Output	Double-click to export						
✓		<b>ram</b>	On-Chip Memory (RAM or ROM)...							
		clk1	Clock Input	Double-click to export	clk_0					
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	0x0000_8000	0x0000_cfff			
		reset1	Reset Input	Double-click to export	[clk1]					
✓		<b>cpu</b>	Nios II Processor							
		clk	Clock Input	Double-click to export	clk_0					
		reset	Reset Input	Double-click to export	[clk]					
		data_master	Avalon Memory Mapped Master	Double-click to export	[clk]					
		instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]					
		irq	Interrupt Receiver	Double-click to export	[clk]			IRQ 0	IRQ 31	
		debug_reset_requ...	Reset Output	Double-click to export	[clk]					
		debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0001_0800	0x0001_0fff			
		custom_instructio...	Custom Instruction Master	Double-click to export	[clk]					
✓		<b>leds</b>	PIO (Parallel I/O) Intel FPGA IP							
		clk	Clock Input	Double-click to export	clk_0					
		reset	Reset Input	Double-click to export	[clk]					
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0001_1050	0x0001_105f			
		external_connection	Conduit	leds_external_conne...						
✓		<b>bp</b>	PIO (Parallel I/O) Intel FPGA IP							
		clk	Clock Input	Double-click to export	clk_0					
		reset	Reset Input	Double-click to export	[clk]					
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0001_1040	0x0001_104f			
		external_connection	Conduit	bp_external_connecc...						
✓		<b>jtag_uart_0</b>	JTAG UART Intel FPGA IP							
		clk	Clock Input	Double-click to export	clk_0					
		reset	Reset Input	Double-click to export	[clk]					
		avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0001_1068	0x0001_106f			
		irq	Interrupt Sender	Double-click to export	[clk]					
✓		<b>pwm_0</b>								
		clock	Clock Input	Double-click to export	clk_0					
		avalon_slave_0	Avalon Memory Mapped Slave	Double-click to export	[clock]	0x0001_1030	0x0001_103f			
		reset	Reset Input	Double-click to export	[clock]					
		conduit_end	Conduit	pwm_0_conduit_end	[clock]					



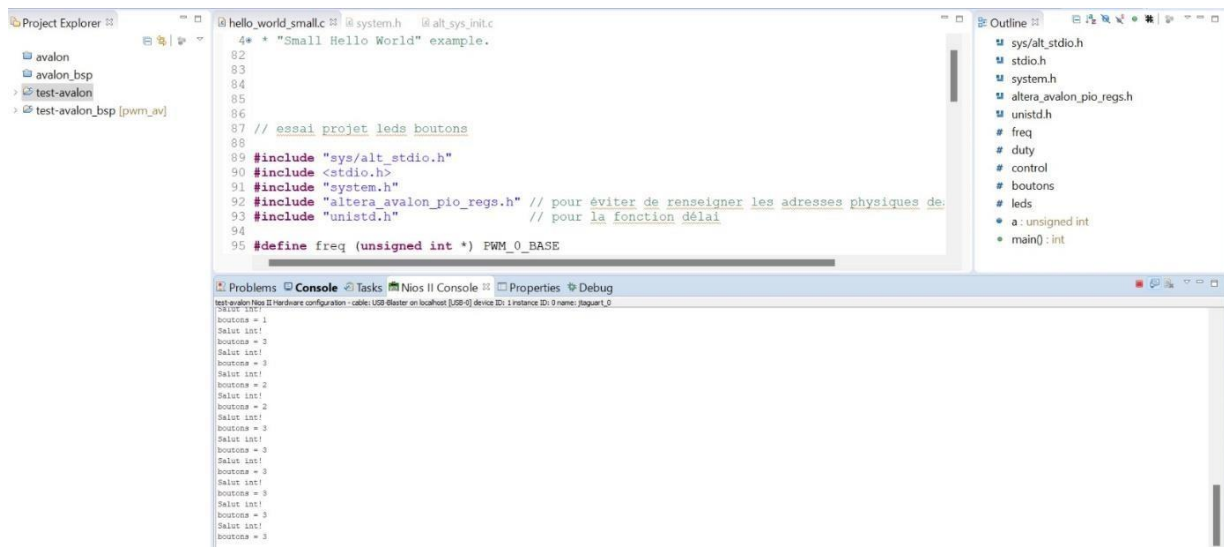
Il faut donc introduire le SOPC dans un schéma de bloc et assigner les ports d'entrées sorties pour le système avant de compiler le projet :



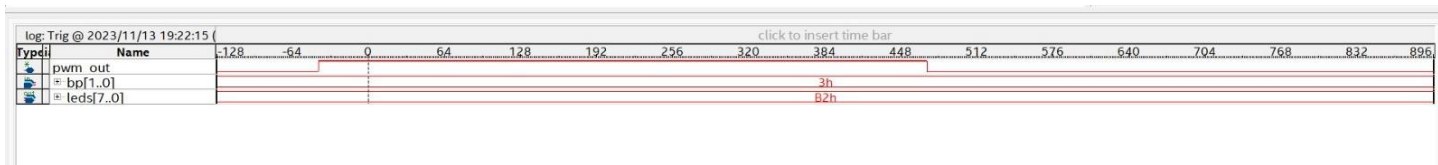
### III.5 Partie software :

Dans cette étape, nous utilisons l'outil NOIS II Software Tool ou Eclipse pour démarrer un nouveau projet destiné à la programmation de notre SOPC.

Lors de la création du projet, il est nécessaire de choisir le fichier avalon.sopinfo, qui sera utilisé pour l'implémentation et la programmation. Après avoir créé le projet, nous devons le compiler et ensuite l'exécuter en utilisant l'option "RUN as NIOS Hardware".



Suite à l'intégration du programme de modulation de largeur d'impulsion (PWM) dans le SOPC et à son exécution, il est nécessaire de procéder à un test de la sortie PWM pour vérifier son fonctionnement.

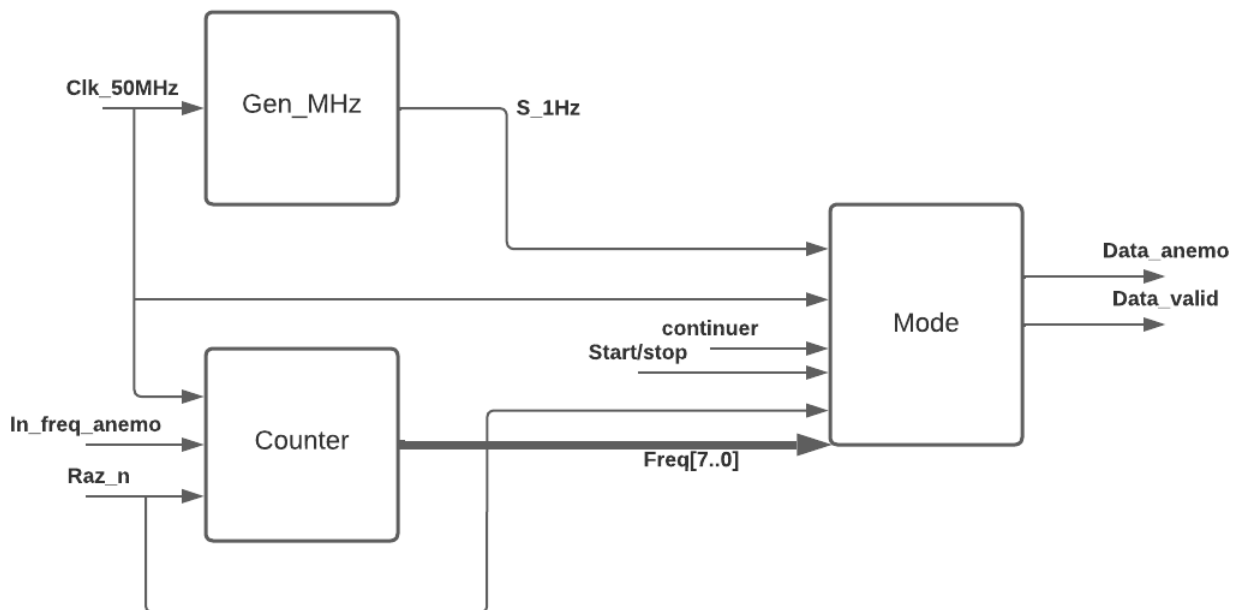


## IV. Conception d'une fonction simple : l'anémomètre

### IV.1 Analyse fonctionnelle

La tâche d'acquérir la vitesse du vent, qui peut varier de 0 à 250 Km/h, est accomplie au moyen d'un anémomètre. Cet appareil est conçu pour transformer la vitesse du vent en une fréquence variable, allant de 0 à 250 Hz. Pour réaliser cette fonction simple « *Gestion Anémomètre* », il faut répondre aux besoins et exigences.

Mais avant de commencer à l'implémentation, nous allons par un schéma fonctionnel sous forme des blocs pour mieux cerner les fonctions principales suivant en respectant le cahier de charge :



#### **Gen MHz :**

Ce composant constitue un bloc permettant de générer une horloge de 1 Hz, assurant ainsi la synchronisation des divers processus au sein de notre circuit.

#### **Counter :**

Le compteur est un élément clé qui compte le nombre de fronts montants du signal numérique, ce qui permet de mesurer la fréquence.

#### **Mode :**

La fonction anemo gère les différents modes de mesure de la fréquence (in\_freq).



Les résultats de cette mesure sont enregistrés dans une variable de sortie, codée sur 8 bits et nommée data\_anemometre. Lorsqu'une mesure est validée, le circuit émet en sortie à la fois les données data\_anemometre et le signal de validation des données (data valid).

### Générateur d'Horloge ("Gen 1MHz") :

Le générateur d'horloge fournit le signal d'horloge synchronisé nécessaire pour le fonctionnement de tous les composants numériques. Il reçoit le signal d'horloge de la carte principale (Clk 50M) et peut être configuré pour générer fréquence 1 Hz.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity Gen_MHz is
    port (
        clk_50M : in std_logic;
        S_1Hz : out std_logic
    );
end Gen_MHz;

architecture arch_1Hz of Gen_MHz is
    signal tmp: std_logic := '0';
begin
    process (clk_50M) is
        variable cnt: integer := 0;
    begin
        if rising_edge(clk_50M) then
            cnt := cnt + 1;
            if cnt = 25000000 then
                tmp <= not tmp;
                cnt := 0;
            end if;
        end if;
        S_1Hz <= tmp;
    end process;
end arch_1Hz;
```

### Compteur ("counter") :

Ce composant est au cœur de la mesure de la vitesse du vent. Il détecte les fronts montants du signal d'entrée de l'anémomètre (in\_freq\_anemometre) et génère un signal d'impulsion correspondant à chaque rotation de l'anémomètre.

```
entity counter is
  port(
    clk_50M : in std_logic;
    in_freq_anemo : in std_logic;
    raz_n : in std_logic;
    frq : out integer range 0 to 255
  );
end counter;

architecture arch_counter of counter is
  signal compt1 : integer := 0;
  signal compt2 : integer range 0 to 255 := 0;
  signal stop : std_logic := '0';

begin
  clk : process(clk_50M, raz_n)
  begin
    if raz_n = '0' then
      compt1 <= 0;
    elsif rising_edge(clk_50M) then
      compt1 <= compt1 + 1;
      if compt1 > 50000000 then
        stop <= '1';
        if compt2 = 0 then
          stop <= '0';
          compt1 <= 0;
        end if;
      end if;
    end if;
  end process clk;

  -- Processus pour le calcul de fréquence basé sur in_freq
  freq : process(clk_50M, raz_n)
  begin
    if raz_n = '0' then
      compt2 <= 0;
    elsif rising_edge(in_freq_anemo) then
      if stop = '0' then
        compt2 <= compt2 + 1;
      else
        frq <= compt2;
        compt2 <= 0;
      end if;
    end if;
  end process freq;
end arch_counter;
```

### Contrôleur de Mode ("mode") :

Le contrôleur de mode sélectionne le mode de fonctionnement de l'anémomètre, permettant soit une opération continue, soit une activation/désactivation manuelle pour des mesures spécifiques ou des tests.

Le bloc de sortie de données reçoit le compte du compteur et le convertit en une mesure de vitesse du vent, qui peut ensuite être affichée ou transmise à d'autres systèmes pour une analyse plus poussée.

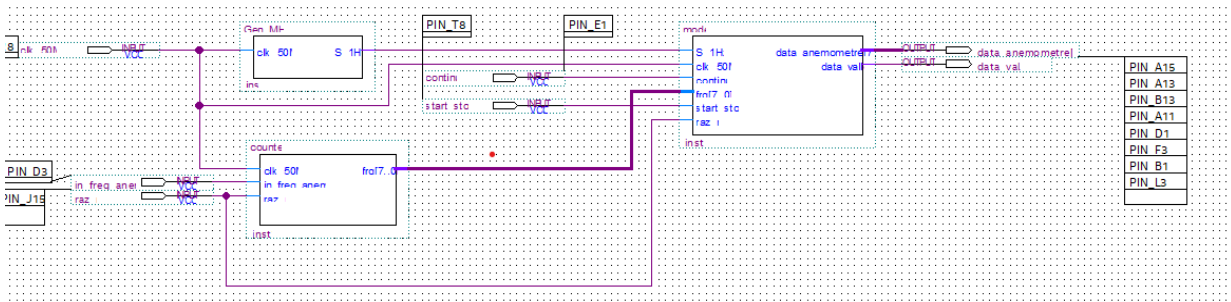
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity mode is
    port (
        S_1Hz : in std_logic;           -- Signal d'entrée HZ
        clk_50M : in std_logic;         -- Horloge d'entrée à 50 MHz
        continu : in std_logic;          -- Signal pour le mode continu
        frq : in integer range 0 to 255; -- Fréquence d'entrée
        start_stop : in std_logic;       -- Signal pour démarrer/arrêter
        raz_n : in std_logic;            -- Signal de remise à zéro
        data_anemometre : out std_logic_vector(7 downto 0); -- Données de l'anémomètre
        data_valid : out std_logic       -- Validité des données
    );
end entity mode;

architecture arch_la_fct of mode is
begin
    -- Processus principal
    process(clk_50M, S_1Hz)
    begin
        if raz_n = '0' then
            data_valid <= '0';
            data_anemometre <= (others => '0');
        elsif rising_edge(S_1Hz) then
            if continu = '0' then
                if start_stop = '0' then
                    data_valid <= '0';
                else
                    data_valid <= '1';
                    data_anemometre <= std_logic_vector(to_unsigned(frq, 8));
                end if;
            else
                data_anemometre <= std_logic_vector(to_unsigned(frq, 8));
                data_valid <= '1';
            end if;
        end if;
    end process;
end architecture arch_la_fct;
```

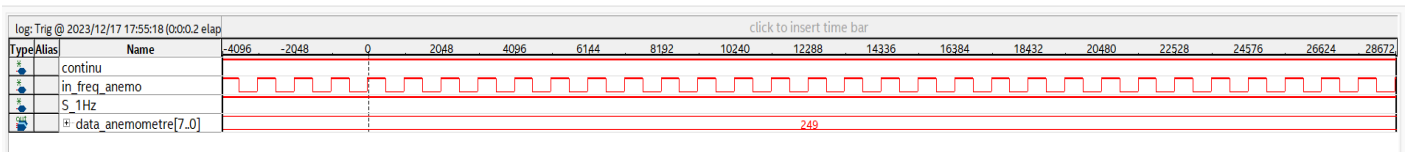
## IV.2 Implantation

Voici le schéma fonctionnel global de chaque bloc, le fonctionnement de l'anémomètre commence par la détection des fronts montants du signal de l'anémomètre, qui sont comptés par le compteur. Chaque seconde, le compteur est réinitialisé par le signal de période d'une seconde, et la valeur comptée est envoyée au bloc de sortie de données pour conversion en vitesse du vent.



### IV.3 Validation de la fonction anémomètre :

Pour vérifier le bon fonctionnement de notre code VHDL, nous avons utilisé un GBF pour générer des signaux carrés avec des paramètres spécifiques, permettant ainsi une validation de notre fonction de "gestion de l'anémomètre". Nous avons observé le signal d'entrée sur l'oscilloscope, lequel varie de 0 à 250 Hz (nous avons choisi 249Hz par exemple avec un rapport cyclique de 50% dans ce cas). La correspondance de la vitesse, affichée sur les LEDs de notre carte, va de 0 à 250 km/h.

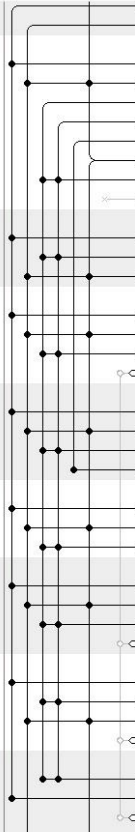
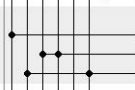
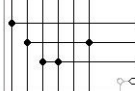
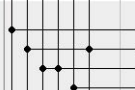
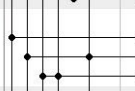

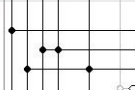
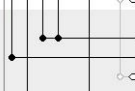


### IV.4 MISE EN ŒUVRE DE SOPC «Anémomètre »:

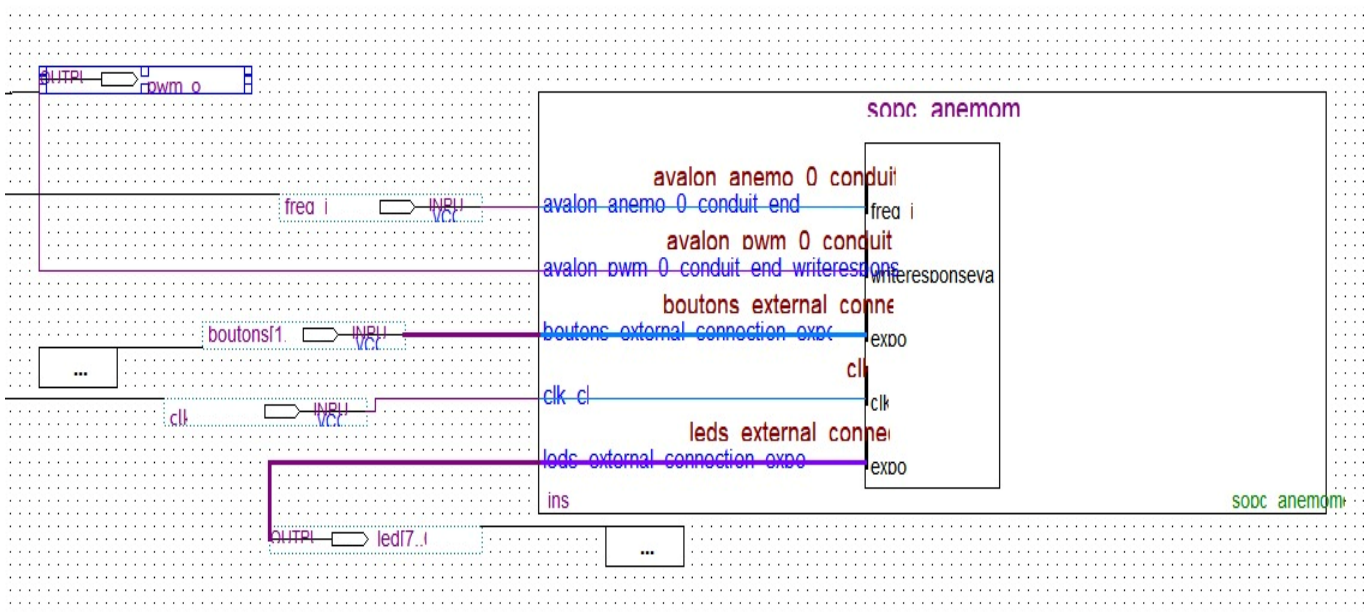
L'interface Avalon dispose de 2 registres tels que décrits ci-dessous :

Registre	Adresse	Type	Bits concernés
Config	0	R/W	b2=Start/Stop, b1=continu, b0=raz_n
Code	1 (4)	R/W	b9=valid, b7...b0= data_anemometre

Pour télécharger le schéma sur la carte DE0, nous avons employé Platform Designer pour concevoir le microprocesseur et les composants périphériques, y compris l'intégration de l'anémomètre via une interface Avalon. Nous avons inséré le code VHDL de cette interface dans Platform Designer, où il a été positionné aux côtés d'un CPU, de la mémoire on-chip, du Jtag, et de quelques PIO.

Use	Connections	Name	Description	Export	Clock	Base	End	I...	Tags	Opcode Name
<input checked="" type="checkbox"/>		<b>CPU</b> clk clk_reset data_master instruction_m... irq debug_reset_r... debug_mem_... custom_instru...	Clock Output Reset Output Nios II Processor Clock Input Reset Input Avalon Memory Mapped ... Avalon Memory Mapped ... Interrupt Receiver Reset Output Avalon Memory Mapped ... Custom Instruction Master	<a href="#">Double-click to</a> <a href="#">Double-click to</a> <a href="#">Double-click to</a> <a href="#">Double-click to</a> <a href="#">Double-click to</a> <a href="#">Double-click to</a> <a href="#">Double-click to</a> <a href="#">Double-click to</a>	clk_0 [clk] [clk] [clk] [clk] [clk] [clk] [clk]					
<input checked="" type="checkbox"/>		<b>RAM</b> clk1 s1 reset1	On-Chip Memory (RAM o... Clock Input Avalon Memory Mapped ... Reset Input	<a href="#">Double-click to</a> <a href="#">Double-click to</a> <a href="#">Double-click to</a>	clk_0 [clk1] [clk1]	# 0x0000 8000	0x0000_cfff			
<input checked="" type="checkbox"/>		<b>LEDS</b> clk reset s1 external_conn...	PIO (Parallel I/O) Intel F... Clock Input Reset Input Avalon Memory Mapped ... Conduit	<a href="#">Double-click to</a> <a href="#">Double-click to</a> <a href="#">Double-click to</a> <a href="#">Double-click to</a>	clk_0 [clk] [clk]	# 0x0001 1050	0x0001_105f			
<input checked="" type="checkbox"/>		<b>JTAG_UART</b> clk reset avalon_jtag_sl... irq	JTAG UART Intel FPGA IP Clock Input Reset Input Avalon Memory Mapped ... Interrupt Sender	<a href="#">Double-click to</a> <a href="#">Double-click to</a> <a href="#">Double-click to</a> <a href="#">Double-click to</a>	clk_0 [clk] [clk] [clk]	# 0x0001 1098	0x0001_109f		16	
<input checked="" type="checkbox"/>		<b>SYST_ID</b> clk reset control_slave	System ID Peripheral Inte... Clock Input Reset Input Avalon Memory Mapped ...	<a href="#">Double-click to</a> <a href="#">Double-click to</a> <a href="#">Double-click to</a>	clk_0 [clk] [clk]	# 0x0001 1090	0x0001_1097			
<input checked="" type="checkbox"/>		<b>BOUTONS</b> clk reset s1 external_conn...	PIO (Parallel I/O) Intel F... Clock Input Reset Input Avalon Memory Mapped ... Conduit	<a href="#">Double-click to</a> <a href="#">Double-click to</a> <a href="#">Double-click to</a> <a href="#">Double-click to</a>	clk_0 [clk] [clk]	# 0x0001 1040	0x0001_104f			
<input checked="" type="checkbox"/>		<b>avalon_pwm_0</b> avalon_pwm clock avalon_slave_0 reset conduit_end	avalon_pwm Clock Input Avalon Memory Mapped ... Reset Input Conduit	<a href="#">Double-click to</a> <a href="#">Double-click to</a> <a href="#">Double-click to</a> <a href="#">Double-click to</a> <a href="#">Double-click to</a>	clk_0 [clock] [clock] [clock]	# 0x0001 1060	0x0001_106f			
<input checked="" type="checkbox"/>		<b>avalon_anem...</b> avalon_slave clock conduit_end reset	avalon_anemo Avalon Memory Mapped ... Clock Input Conduit Reset Input	<a href="#">Double-click to</a> <a href="#">Double-click to</a> <a href="#">Double-click to</a> <a href="#">Double-click to</a>	[clock] clk_0 [clock]	# 0x0001 1070	0x0001_107f			

Il faut donc introduire le SOPC dans un schéma de bloc et assigner les ports d'entrées sorties pour le système avant de compiler le projet :



## IV.5 Partie software NOIS II :

Dans cette étape, nous utilisons l'outil NOIS II Software Tool ou Eclipse pour créer le code en C et faire fonctionner tout le système et voir la sortie data\_anemometre et Config.

Lors de la création du projet, il est nécessaire de choisir le fichier avalon\_anemo.sopcinfo, qui sera utilisé pour l'implémentation et la programmation. Après avoir créé le projet, nous devons le compiler et ensuite l'exécuter en utilisant l'option "RUN as NIOS Hardware".

```
#include "sys/alt_stdio.h"
#include <stdio.h>
#include "system.h"
#include "altera_avalon_pio_regs.h"
#include "unistd.h"

#define config (unsigned int *) (AVALON_ANEMO_0_BASE)
#define code (unsigned int *) (AVALON_ANEMO_0_BASE + 4)

unsigned int a;
int main()
{
    *config = 7;
    int resultat = 0;
    int data = 0;
    int i = 0xFF;

    alt_putstr("Salut ext!\n");
    while (1)
    {
        alt_putstr("comm OK !\n");
        // anemometre
        resultat = *code;
        data = i & resultat;
        printf("Anemo data = %d | config = %X \n", data, *config);
    }

    return 0;
}
```

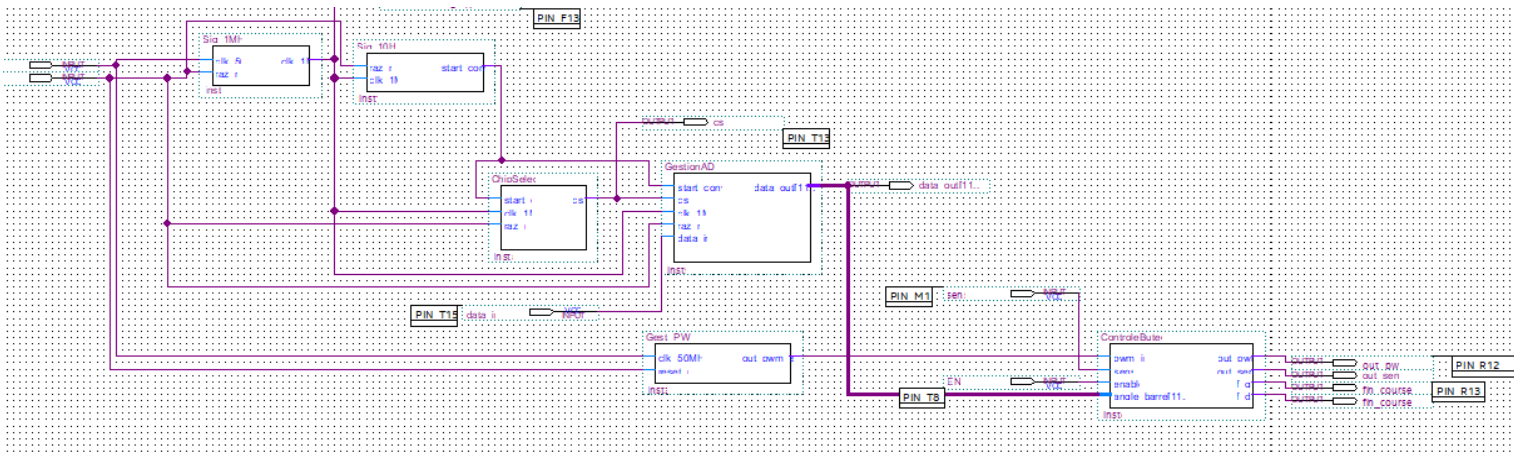
On peut voir les résultats de notre système dans la figure suivant :

```
comm OK !
Anemo data = 249 | config = 2F9
comm OK !
Anemo data = 249 | config = 2F9
comm OK !
Anemo data = 250 | config = 2FA
comm OK !
Anemo data = 250 | config = 2FA
comm OK !
Anemo data = 250 | config = 2FA
comm OK !
Anemo data = 250 | config = 2FA
```



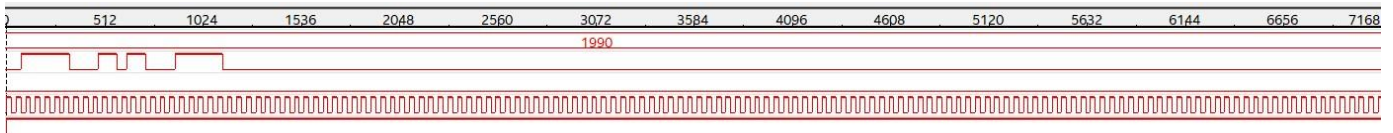


## V.2 Implantation :



## V.3 Simulation et validation :

Pour confirmer le bon fonctionnement de ce bloc, nous avons effectué une simulation à l'aide de l'outil Logic Analyzer, dont les résultats sont représentés dans la figure ci-dessous.



D'après les résultats du signal Tap Analyzer, nous pouvons dire que notre système fonctionne correctement.

## V.4 MISE EN ŒUVRE DE SOPC « Vérin » :

En utilisant platform designer, on assemble notre composant avec une ram, un cpu et le bus Avalon, pour utiliser le processeur et développer en soft notre système avec le langage C.



L'interface Avalon dispose de 6 registres tels que décrits ci-dessous :

```
#define freq (int *)VER_COMPONENT_0_BASE
#define duty (int *) (VER_COMPONENT_0_BASE + 4)
#define butee_g (int *) (VER_COMPONENT_0_BASE + 8)
#define butee_d (int *) (VER_COMPONENT_0_BASE + 12)
#define config (int *) (VER_COMPONENT_0_BASE + 16)
#define angle_barre (int *) (VER_COMPONENT_0_BASE + 20)
```

- **Freq** : fixe la fréquence de la PWM moteur (exemple : si freq = 2000 alors la fréquence de la PWM =  $\text{clk}/2000$  soit 25KHz si clk=50MHz)
- **Duty** : fixe le rapport cyclique
- **Butee\_g et butee\_d** : réglables de 0 à 4095 elles fixent les valeurs limites que le vérin (angle\_barre) ne doit pas dépasser. En dehors de ces valeurs, le moteur est automatiquement coupé. Ces valeurs sont mises à 0 par défaut et doivent être initialisées au démarrage du système.
- **Config** : registre de configuration du circuit.
- **Angle\_barre** : il donne la valeur de l'angle de barre codée sur 12 bits (résultat de la conversion analogique numérique) ...

## **VI. Conclusion**

Au cours de ce semestre, notre participation aux projets présentés dans ce document a renforcé nos compétences en VHDL, en particulier dans le domaine de l'électronique numérique.

Ce bureau d'études nous a offert une perspective unique sur la programmation en VHDL, qui est cruciale pour créer des descriptions matérielles qui modélisent et conçoivent le comportement et l'architecture des systèmes électroniques numériques.

Pendant la conception de certaines fonctions, nous avons été confrontés à des contraintes matérielles, nécessitant une compréhension approfondie de chaque composant numérique pour assurer le fonctionnement de certains blocs fonctionnels. Il est aussi important de souligner l'efficacité des outils de simulation fournis par les logiciels de programmation VHDL, qui sont extrêmement utiles pour le développement de systèmes électroniques grâce à la conception assistée par ordinateur.