

MASTER 2
SYSTÈME ET MICROSYSTÈME EMBARQUÉS

COMPTE RENDU
FONCTION 1 : ANÉMOMÈTRE

Rédigé par :
Bilal JIDAN
Amira BERKI

14 novembre 2023

TABLE DES MATIÈRES

0.1	Interfaces pilote de barre franche	1
0.1.1	Introduction	1
0.1.2	Gestion anémomètre (fonction simple)	2
0.1.2.1	Spécification de l'anémomètre	2
0.1.2.2	Schéma fonctionnel de l'anémomètre	2
0.1.2.3	Intégration de l'anémomètre au SOPC	7

TABLE DES FIGURES

1	Présentation de pilote de barre franche.	1
2	Les blocs de Pilote de barre franche.	1
3	Architecture de l'anémomètre	2
4	Schéma Fonctionnel de l'anémomètre	3
5	Connexion entre l'anémomètre et les composants dans le SOPC	7

0.1 Interfaces pilote de barre franche

0.1.1 Introduction

Dans le cadre de notre projet de bureau d'étude, il nous a été demandé de mettre en oeuvre un système de contrôle de trajectoire d'un voilier avec un mode de fonctionnement automatique et un mode manuel (monocoup). Ce contrôle de trajectoire est réalisé en faisant varier l'angle de la barre franche avec un moteur DC présent dans la partie Vérin . Le système mis en place possède un anémomètre qui nous permettra de connaître la vitesse du vent, une boussole pour connaître la direction, un verin pour gérer l'angle de la barre et des boutons poussoirs pour piloter le système.

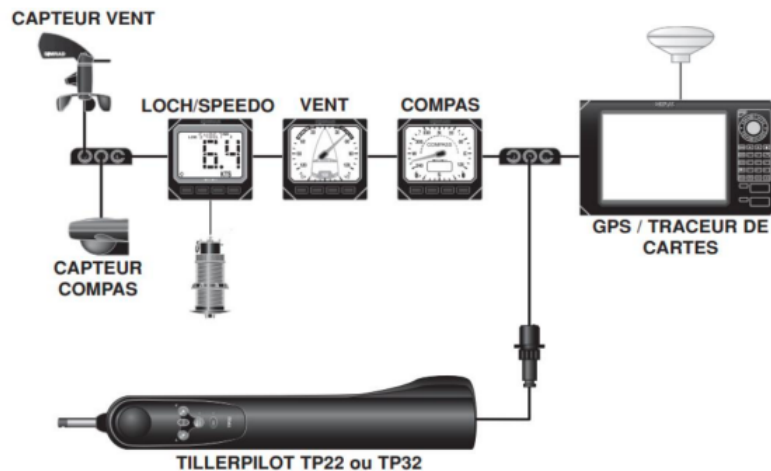


FIGURE 1 – Présentation de pilote de barre franche.

Le pilote de barre franche se constitue de plusieurs fonctions :

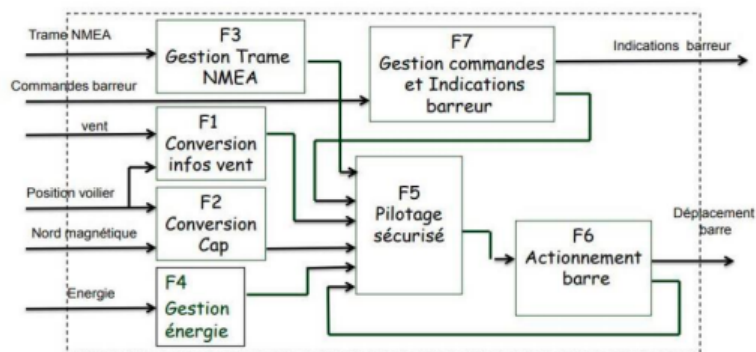


FIGURE 2 – Les blocs de Pilote de barre franche.

L'objectif de ce BE consiste à concevoir deux fonctions de l'interface du pilote de barre franche ; une première fonction simple et une deuxième difficile. Notre choix s'est porté sur l'anémomètre (fonction simple) et le vérin (fonction difficile).


La réalisation de chacune de ces fonctions s'est effectuée sur plusieurs parties :

- Analyse des spécifications,
- Schéma fonctionnel,
- Implementation du code en VHDL,
- Simulation sur Quartus II,
- Essai sur la carte DE0 NANO,
- Interfaçage avec microprocesseur (NIOS + Altera Avalon).

0.1.2 Gestion anémomètre (fonction simple)

0.1.2.1 Spécification de l'anémomètre

En mesurant la fréquence issue d'un capteur de vent, l'anémomètre permet de mesurer la vitesse du vent. Cette fréquence est variable entre 0 et 250Hz qui correspondent respectivement à la vitesse de 0 et 250km/h.



Type de mesure	Mesure de la vitesse du vent
Signal de sortie	logique fréquence variable 0 à 250 Hz
Alimentation	non alimenté
Capteur	type alternateur
Protection	IP65
Connexion	Connecteur étanche IP65 à 4 broches
Boitier	Aluminium anodisé
Matière palette	Plastique
Protection électrique	Diode Transzorb
Filtres EMI	EN50081 EN50082
Résistance de charge	Min 2 KOhms
Limite de destruction	Supérieure à 75 m/s (270 km/h)
Température de fonctionnement	-30 à +70 C
Plage	0-250Km/h

FIGURE 3 – Architecture de l'anémomètre

Principe de fonctionnement :

La fonction calcul le nombre de front montant du signal entrant `in_freq_anemometre` sur une période de 1s (donc calcul la fréquence) et l'affecte à la sortie `data_anemometre` qui correspond à la vitesse du vent. La vérification de la disponibilité de la donnée se fait grâce à `data_valid`.

0.1.2.2 Schéma fonctionnel de l'anémomètre

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

Diviseur : C'est un bloc qui permet de générer une horloge de 1 khz afin de synchroniser les différents process de notre circuit.

Détecteur des front montants : C'est un bloc qui permet de détecter les fronts montants du signal divisé par le diviseur afin de mettre sa sortie à 0 s'il ne détecte pas des fronts montants, et à 1 dans le cas contraire.

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

Choix_mode : C'est une fonction qui gère les modes de mesure de la fréquence `in_freq`, qui sera mémorisée dans une variable de sortie codé de 8 bits nommé `data_anemometre`, lorsqu'une mesure est valide, le circuit met sa sortie `data_anemo` et `data_valid`.

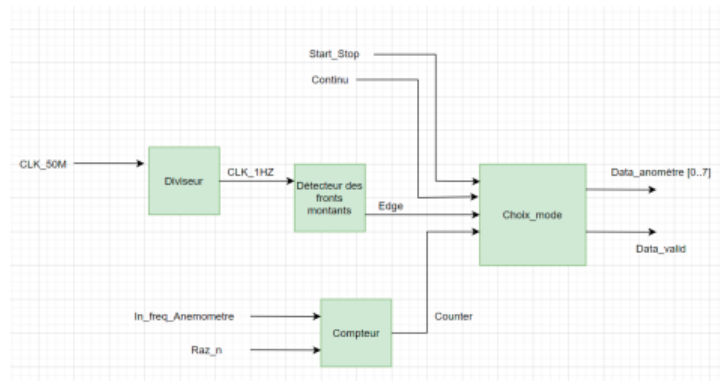
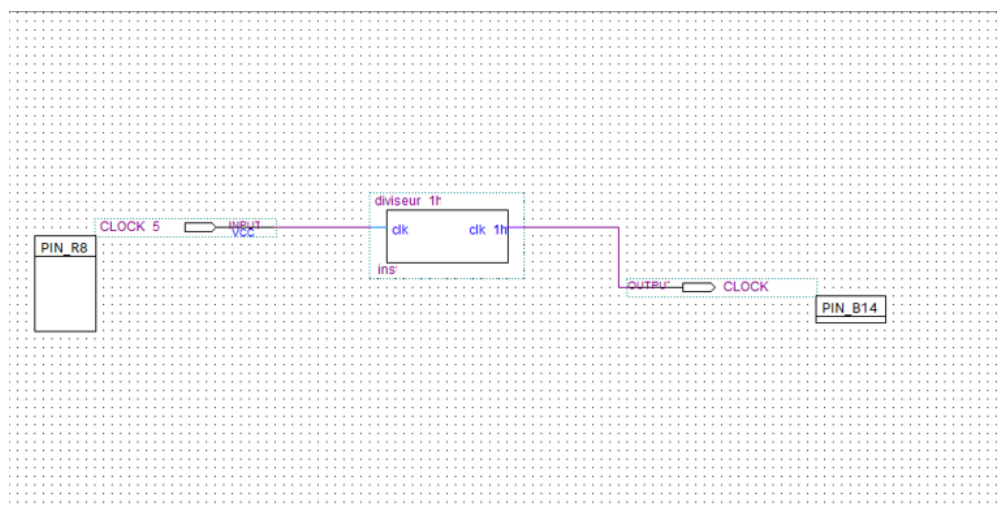


FIGURE 4 – Schéma Fonctionnel de l'anémomètre

Bloc diviseur de fréquence :



Code VHDL utilisé :

```
library IEEE;
use IEEE.STD LOGIC 1164.ALL;
use IEEE.numeric std.ALL;

entity diviseur_1hz is
port ( clk: in std_logic;
      clk_1hz : buffer
      std_logic);
end diviseur_1hz;

architecture bhv of diviseur_1hz
isbegin

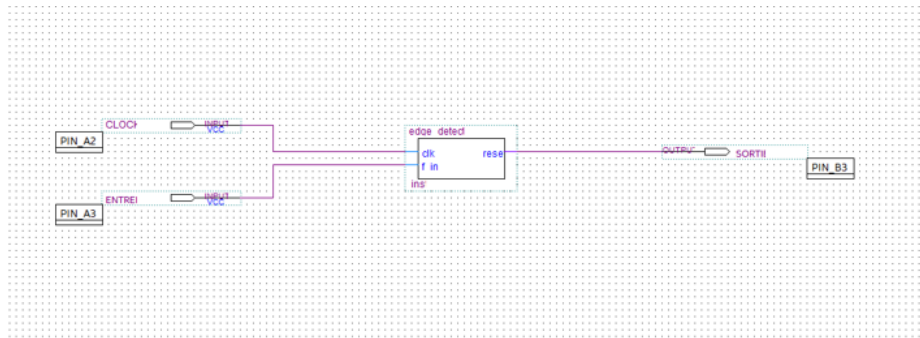
process(clk)
variable cpt : integer range 0 to 49 999 999;-- diviser par 50M pour
obtenir 1hz
begin

if(rising edge(clk)) then
cpt := cpt + 1;
if cpt < 25 000 000 then clk_1hz <= '1';--mettre 1 sur la 1ere moitié
de la période

else clk_1hz <= '0';-- mettre 0 pendant la deuxieme
moitié de la periode
if cpt = 49 999 999 then cpt := 0;
end if;
end if;

end process;
end bhv;
```

Bloc détecteur de front montant :



Code VHDL utilisé :

```

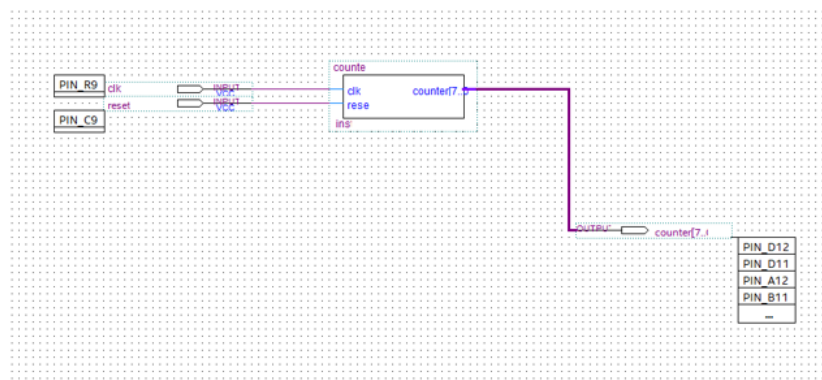
Library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity edge_detect is
    port(
        clk, f_in : in std_logic;
        reset : out std_logic);
end edge_detect;

architecture arc of edge_detect is
    signal etat : std_logic_vector (1 downto 0) := "00";
begin
    process(clk) is
    begin
        if rising_edge(clk) then
            if etat = "00" then
                if f_in = '1' then
                    etat <= "11";
                    reset <= '1';
                end if;
            end if;
            if etat = "11" then
                etat <= "10";
                reset <= '0';
            end if;
            if etat = "10" then
                if f_in = '0' then
                    etat <= "00";
                    reset <= '0';
                end if;
            end if;
        end if;
    end process;
end arc;

```

Bloc Compteur :



Code VHDL utilisé :


```

Library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity counter is
    port(
        clk : in std_logic;
        reset : in std_logic;
        counter : out unsigned (7 downto 0));
end counter;

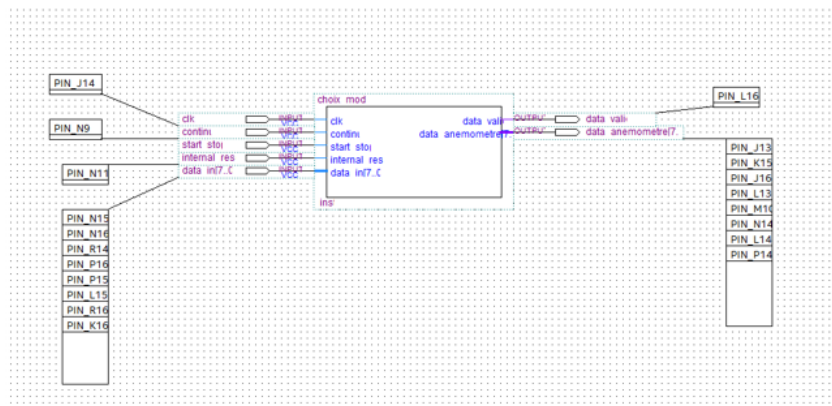
architecture arc of counter is

begin

    process(clk, reset) is
        variable cpt : unsigned (7 downto 0);
    begin
        if reset = '1' then
            counter <= "00000000";
            cpt := "00000000";
        elsif rising_edge(clk) then
            cpt := cpt + 1;
            counter <= cpt ;
        end if;
    end process p_asynchronous_reset;
end arc;

```

Bloc choix de mode :



Code VHDL utilisé :

```

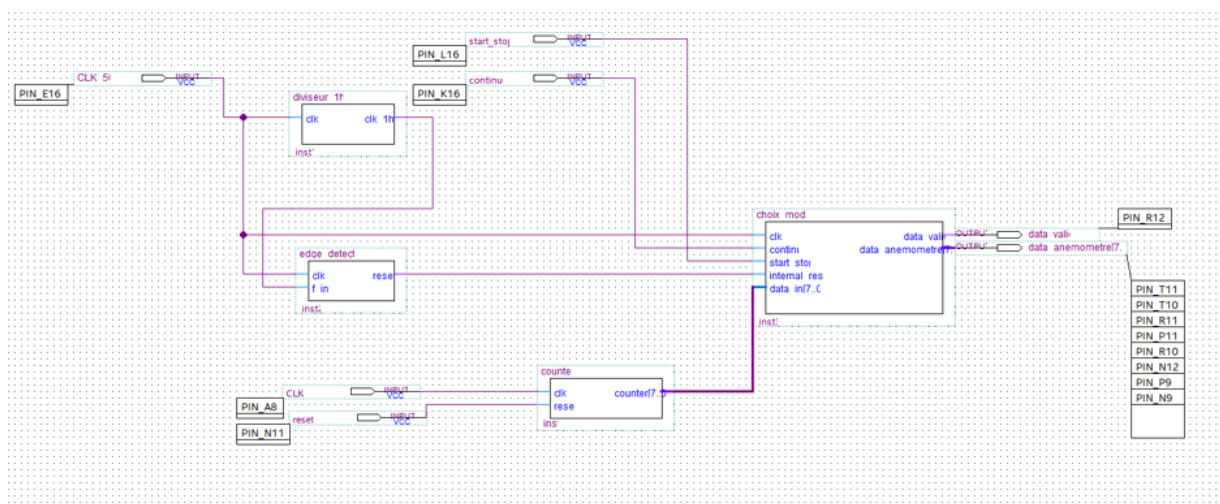
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;

entity choix_mode is
port ( clk, continu, start_stop, internal_reset: in std_logic; data_in : in std_logic_vector (7 downto 0); data_valid
data_anemometre : out std_logic_vector (7 downto 0));
end choix_mode;

architecture bhv of choix_mode is
signal tmp, tmp1 : std_logic_vector (7 downto 0);
begin process(clk)
begin
if(clk'event and clk='1') then if continu = '1' then
if internal_reset = '1' then
data_anemometre <= tmp; data_valid <= '1';
else
tmp <= data_in; data_valid <= '1';
end if;
else
if start_stop = '1' then
data_anemometre <= tmp; data_valid <= '1';
else
data_anemometre <= "00000000";
data_valid <= '0';
if internal_reset='0' then
tmp1 <= data_in;
else
tmp <= tmp1;
end if;
end if;
end if;
end if;
end process; end bhv;

```

Shéma regroupant tous les blocs avec assignements de pins :



0.1.2.3 Intégration de l'anémomètre au SOPC

Afin de lier la fonction anémomètre au Nios, on a eu besoin de créer l'interface avalon. Pour cela nous avons implémenté le "anemometre_avalon.vhd"

Entité de l'interface de l'avalon de l'anémomètre :

```
entity anemometre_avalon is
    port(
        clk, chipselect      : in std_logic;
        write_n, reset_n     : in std_logic;
        writedata             : in std_logic_vector (31 downto 0);
        readdata              : out std_logic_vector (31 downto 0);
        address               : in std_logic;
        in_freq_anemometre   : in std_logic
    );
end entity;
```

Ensuite, nous avons inséré notre composant dans le SOPC et connecter les différents composants (voir figure ci-dessous)

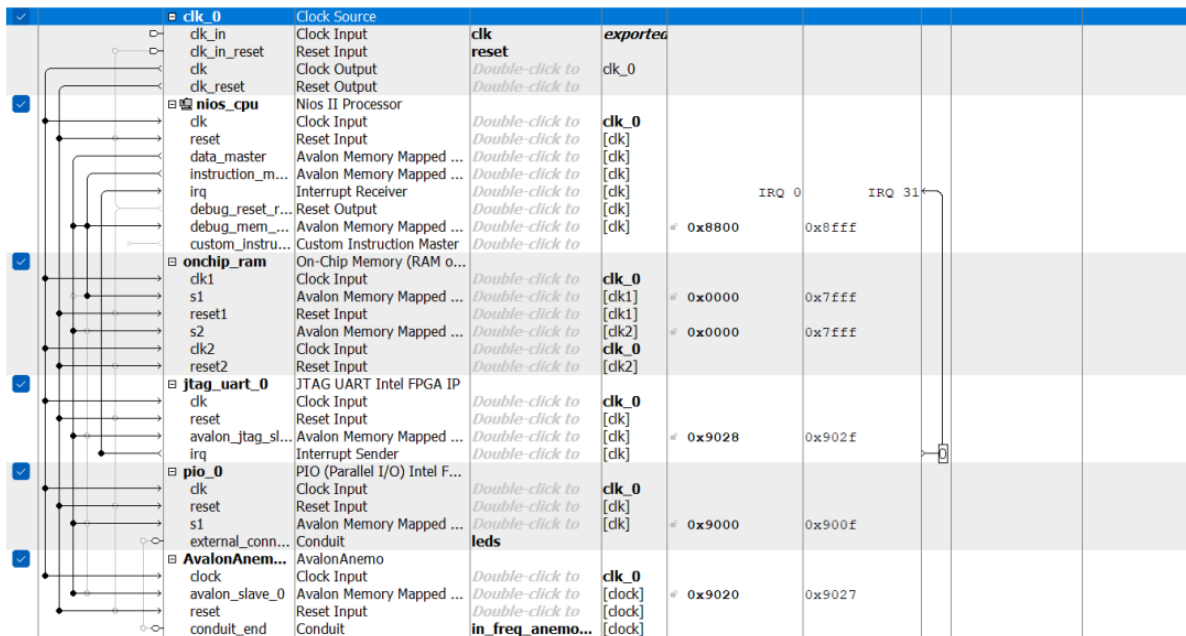


FIGURE 5 – Connexion entre l'anémomètre et les composants dans le SOPC

Une fois le code HDL généré, nous avons programmé le SOPC sous Eclipse.