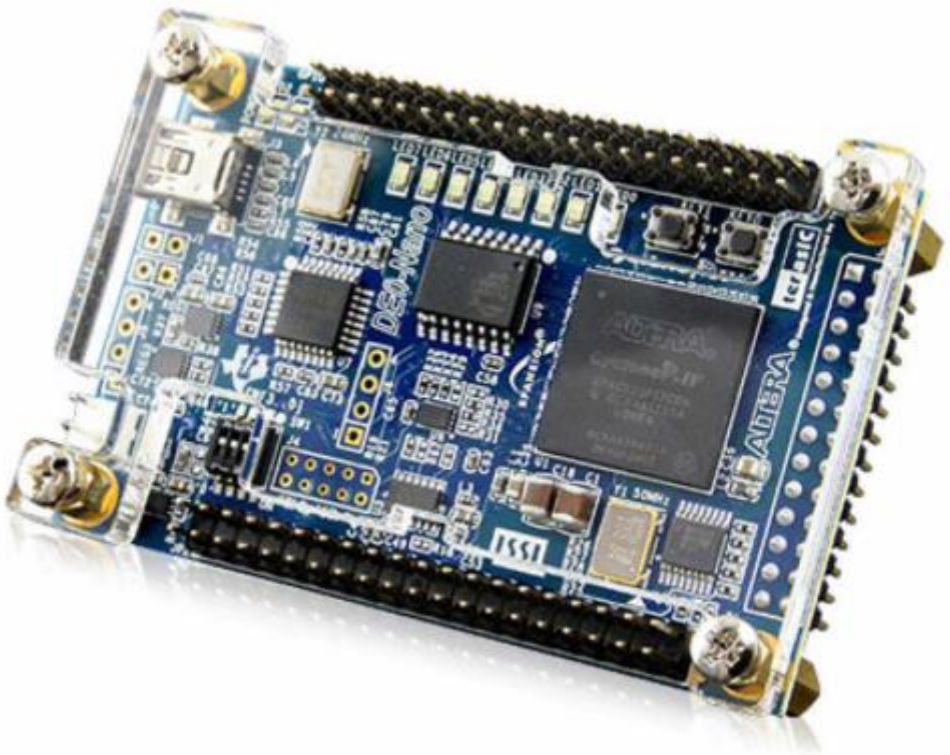


DEVELOPPEMENT D'UN SYSTEME DE BARRE FRANCHE

SYNTHÈSE ET MISE EN ŒUVRE DE SYSTÈME

Solène Badibanga Kalenda | Jennifer Ramos | janvier 2021

Introduction
Dans le cadre de notre master, nous sommes amenés à travailler sur le développement de différentes interfaces d'une barre franche, décrites en langage vhdl et la barre sera simulée par une carte FPGA DE0 nano.
Le pilote de barre franche est composé de 5 interfaces différentes chargées des fonctions spécifiques à accomplir. Ces interfaces seront décrites par la suite.

1. Spécifications du circuit << gestion vérin >>

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

- Gestion de la PWM moteur
- Contrôle des butées de fin de course du vérin
- Gestion du convertisseur AN MCP 3201 de recopie de position de barre
- Interface avec le bus Avalon du NIOS

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 « contrôle_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.
• généré les signaux « fin_course_d » et « fin_course_g »

Gestion du convertisseur AN MCP 3201 : fait appel à 5 fonctions secondaires :

• machine à état pour piloter l'adc et mémoriser la donnée « angle_barre » (process pilote_adc)
• comptage des fronts d'horloge de clk_adc (process compt_fronts)
• 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)

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

• Ecriture des data circulant sur le bus du NIOS dans des registres (écriture)
• Relecture de signaux par renvoi sur le bus du NIOS (lecture)

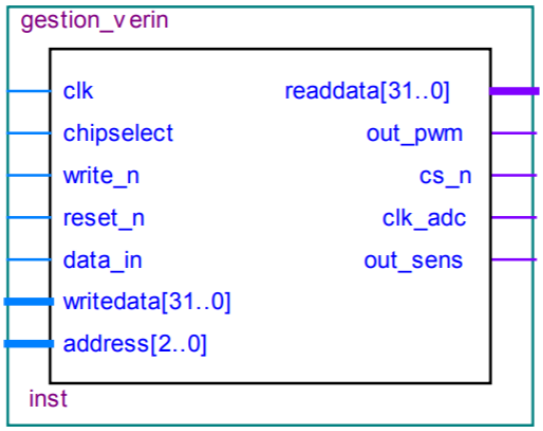
Freq : fixe la fréquence de la PWM moteur (exemple : si freq = 2000 alors la fréquence de la PWM = clk/2000 soit 25KHz si clk=50MHz)

Duty : fixe le rapport cyclique (exemple si duty=freq/2 alors le rapport cyclique est de 50%)
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. La description du rôle de chaque bit est donnée dans le tableau.

Angle_barre : il donne la valeur de l'angle de barre codée sur 12 bits (résultat de la conversion analogique numérique).



2. **<u>Spécifications circuit interface commandes et indications barreur (module gestion des boutons poussoirs)</u>**

Entrées:

BP_Babord
BP_Tribord
BP_STBY
clk_50M
raz_n

Sorties:
codeFonction
ledBabord
ledTribord
ledSTBY
out_bip

-- clk_50M: horloge à 50MHz
-- raz_n: actif à 0 => initialise le circuit
-- valeurs de codeFonction:
-- =0000: pas d'action, le pilote est en veille

-- =0001: mode manuel action vérin babord
-- =0010: mode manuel action vérin tribord
-- =0011: mode pilote automatique/cap
-- =0100: incrément de 1° consigne de cap
-- =0101: incrément de 10° consigne de cap
-- =0111: décrément de 1° consigne de cap
-- =0110: décrément de 10° consigne de cap

3. Spécifications circuit acquisition vitesse vent (module gestion_anemometre)

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_anémomètre : vitesse vent codée sur 8 bits

4. Spécifications circuit interface NMEA (RS232)

Mode émission :

Envoi d'une trame à 4800 bauds, 1 start, 1 stop, pas de parité et constituée de quatre octets codés ASCII : SCDU.

S : signifie le début du message
CDU: centaines, dizaines et unités de degrés correspondant à l'angle de barre.

La trame est émise avec une périodicité d'une seconde (à tester).
Un signal start/stop=1 démarre la transmission, =0 arrête la transmission et revient au repos.
Un signal fin_transmit=1 indique que la transmission est terminée.
Un signal raz_n=0 inhibe le circuit.
Mode réception :
Réception d'une trame à 4800 bauds, 1 start, 1 stop, pas de parité, ayant le même format que l'émission (SCDU).
Un signal mode=1 fait fonctionner le circuit en continu, =0 le récepteur dépend de l'état du signal start/stop.

Le signal start/stop=1 active la réception, =0 arrête la réception et revient au repos.
Un signal data_valid=1 indique que la réception est terminée.
Un signal raz_n=0 inhibe le circuit.

5. Spécifications circuit interface acquisition cap (module gestion compas pour boussole CMPS03 ou CMPS10)

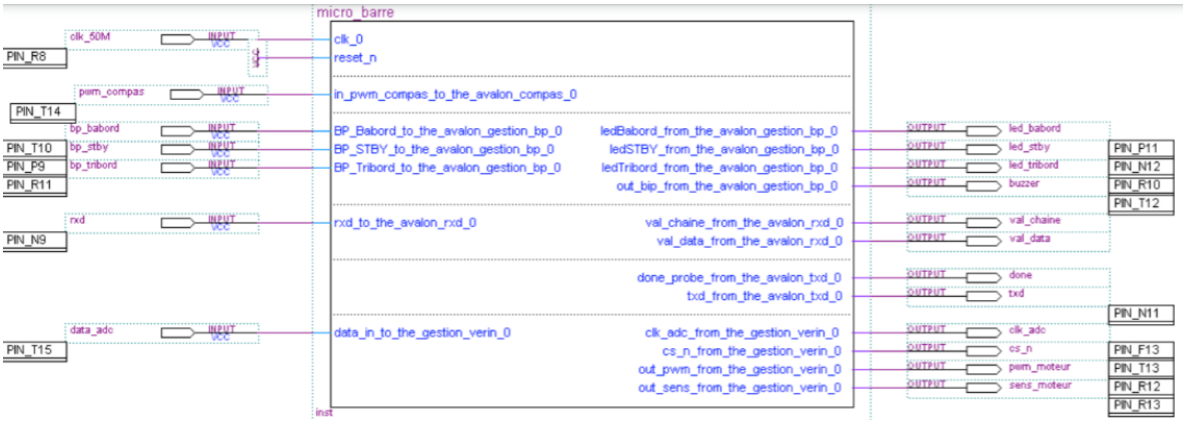
Entrèes:

- clk_50M : hologe 50MHz
- raz_n: reset actif à 0 => initialise le circuit
- in_pwm_compas: signal PWM de la boussole, durée varie de 1ms à 36,9ms
- continu : si=0 mode monocoup, si=1 mode continu
- En mode continu la donnée est rafraichie toutes 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 et est remis à 0 quand start_stop passe à 0
- out_1s : signal de contrôle du top seconde (normalement pas utilisé)
- data_compas : valeur du cap réel exprimé en degré codé sur 9 bits

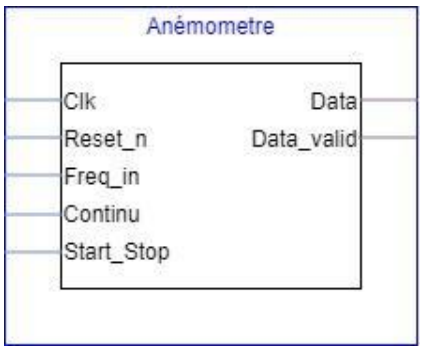
Pour ces travaux pratiques nous allons réaliser l'implémentation de deux de ces interfaces nommé dernièrement.
De plus on est amené a créé un SOPC pour la carte qui intègre les deux interfaces.
Ci-dessous est présentée une figure qui montre le SOPC avec toutes les interfaces implémentées.



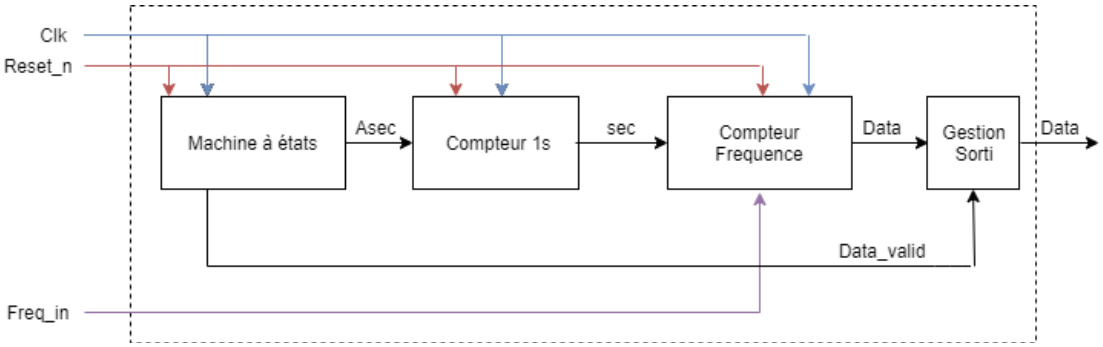
CONCEPTION DE L''INTERFACE ANEMOMETRE

Afin de pouvoir mesurer la vitesse du vent nous allons faire une interface qui soit capable de mesurer une fréquence d'entrée qui varie de 0 à 250 Hz et donner sa valeur en sortie sur un bus de huit bits.

Dans la figure ci-dessous se montre l'interface développée.

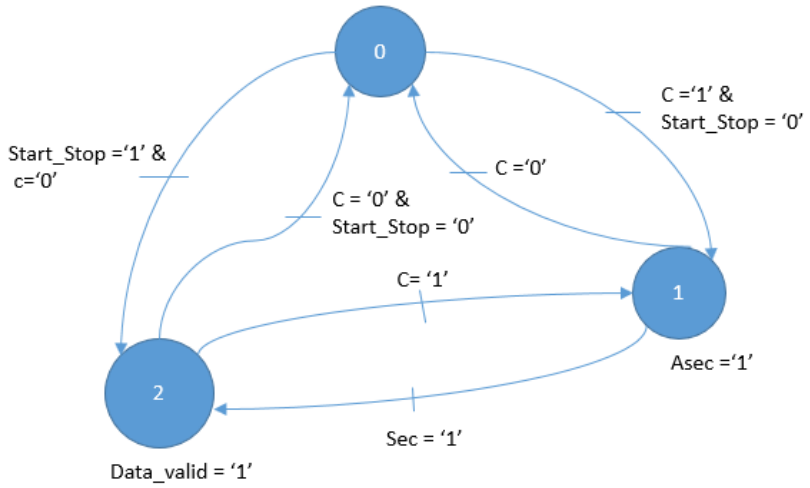


Pour développer l'interface nous somme fait une décomposition fonctionnelle montre comme ci-dessous dans la figure.



Par la suite nous allons décrire chaque fonction.

Machine à états



Cette machine à états montre le comportement de l'interface anémomètre, quand le mode continu est activé, la mesure de fréquence est faite toutes les secondes, quand le monde continu est désactivé et le bouton Start_Stop est appuyé une mesure de fréquence doit être faite.

Pour le mode continu dans l'état 2 s'active un signal Asec qui active le compteur des secondes, et ce dernier envoie un signal.

Compteur 1s

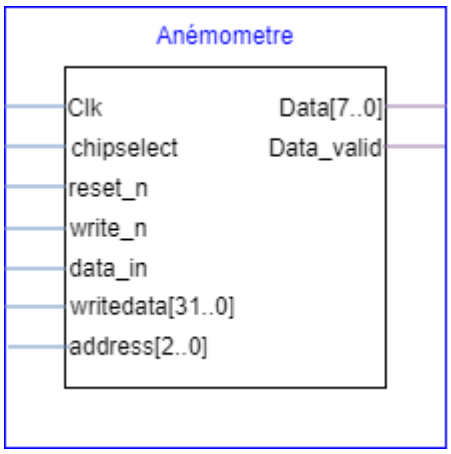
Cette fonction est chargée de compter 50000000 flancs d'horloge pour avoir 1 seconde, Ce compteur est remis à 0 quand le signal Asec est égal à '1', ce compteur comptera en permanence et activera le signal data valide à l'arrivée d'une seconde.

Compteur Fréquence

Cette fonction est chargée de compter le front de montée d'un signal d'entrée pendant une seconde afin de connaître la fréquence du signal, le signal data_valid s'active toutes les secondes pour finaliser et obtenir la mesure de fréquence.

Intégration de l'anémomètre sur le SOPC

Il faut une interface de communication avec le bus AVALON de la carte pour avoir accès aux périphériques de la carte. Cette interface sera développée comme le montre ci-dessous dans la figure.



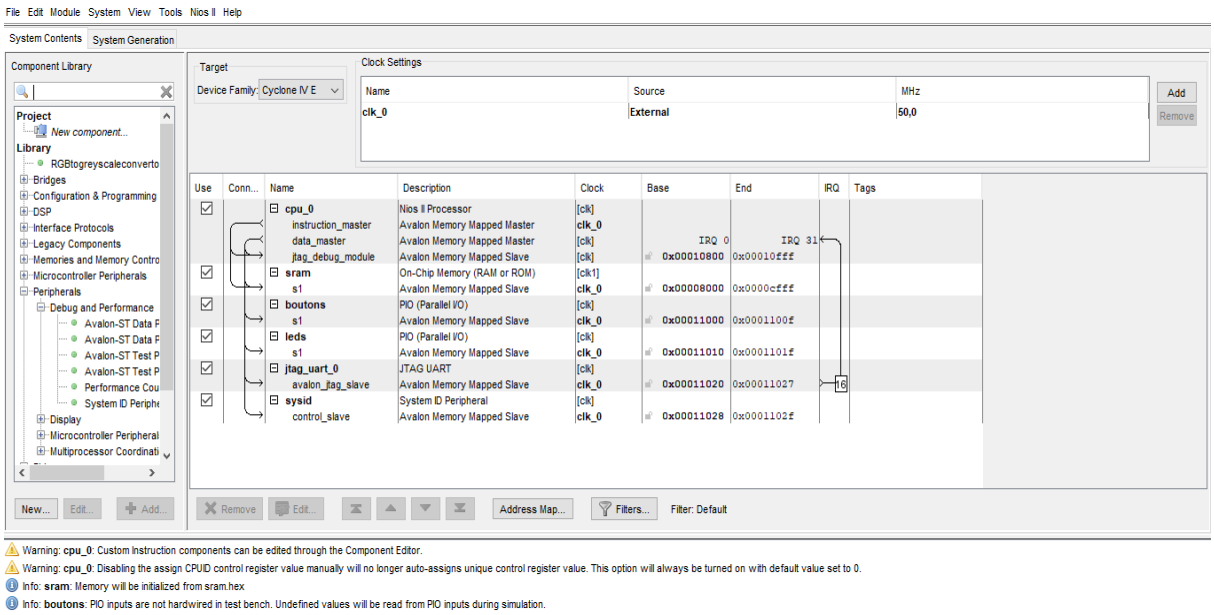
Finalement pour créer le composant avec l'interface anémomètre nous avons utilisé SOPC Builder sur Quartus II version 11.1 .

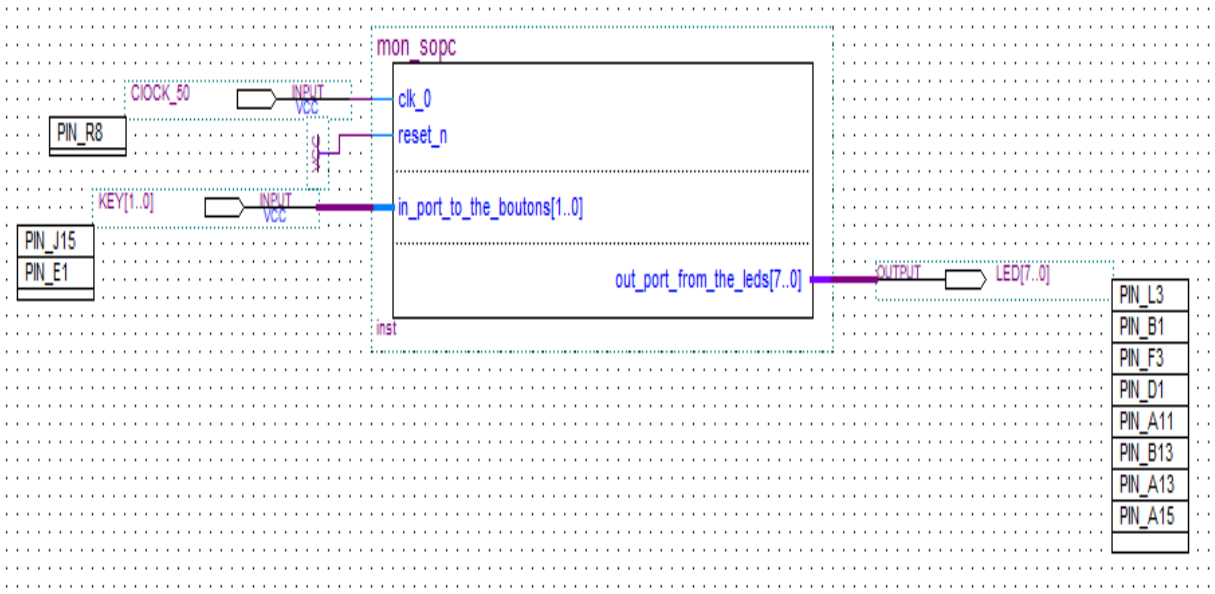
Le composant obtenu se montre dans la figure ci-dessous.

En ayant obtenu le dispositive, il nous reste de l'intégrer sur le SOPC, pour celà nous avons créé un fichier block diagram(.bfd). Mais tout d'abord nous allons montrer la conception de notre SOPC.

CONCEPTION DU SOPC

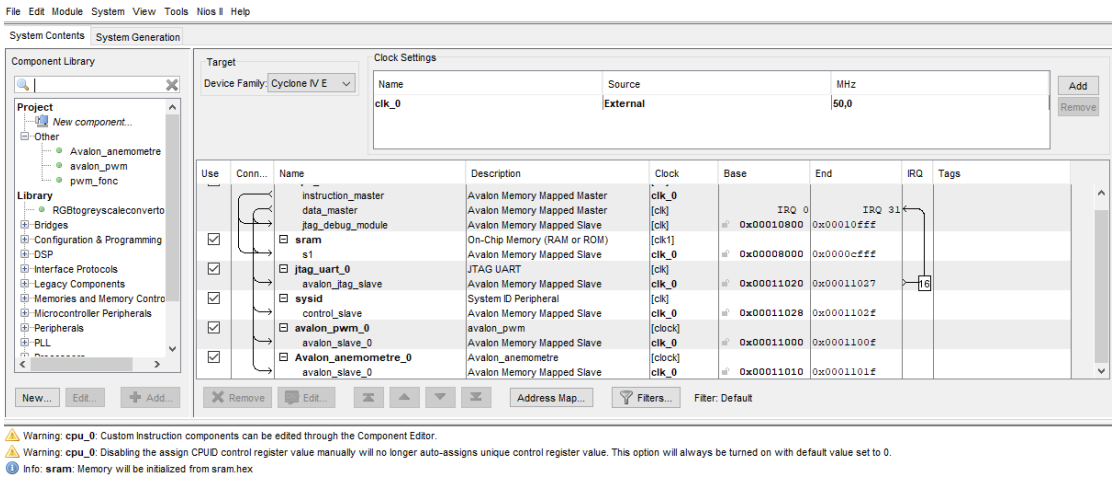
Nous avons utilisé l'outil Sopc Builder pour construire le SOPC, tout d'abord nous avons fait une Sopc simple capable de gérer les boutons d'entrée et des Leds en sortie. Au moment de le créer nous avons ajoutés des composants indispensables tel que le processeur (Nios II Processor), la mémoire Ram, périphérique de communication serial pour télécharger le programme (jtag_uart), un system ID (sysid), un port d'entrée de deux bits et un port en sorti de huit bits. Le SOPC conçu est montré dans la figure ci-dessous.



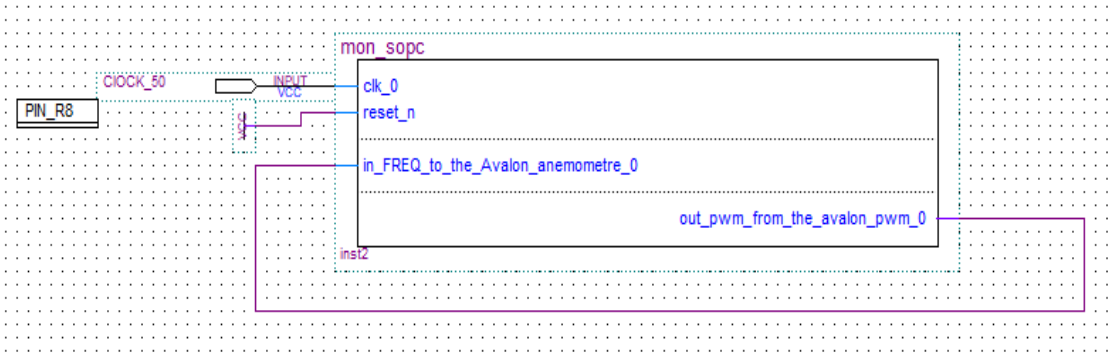


Par la suite, afin de pouvoir tester l'interface anémomètre nous avons intégré une interface pwm au Sopc, laquelle doit permettre de générer un signal à fréquence et rapport cyclique réglable, pour la fréquence minimum de 0 à 250Hz.
Pareille que pour l'anémomètre nous allons faire pour ce composant une interface de communication avec le bus Avalon.

Il peut s'observer dans la figure ci-dessous les deux interfaces ajoutées.

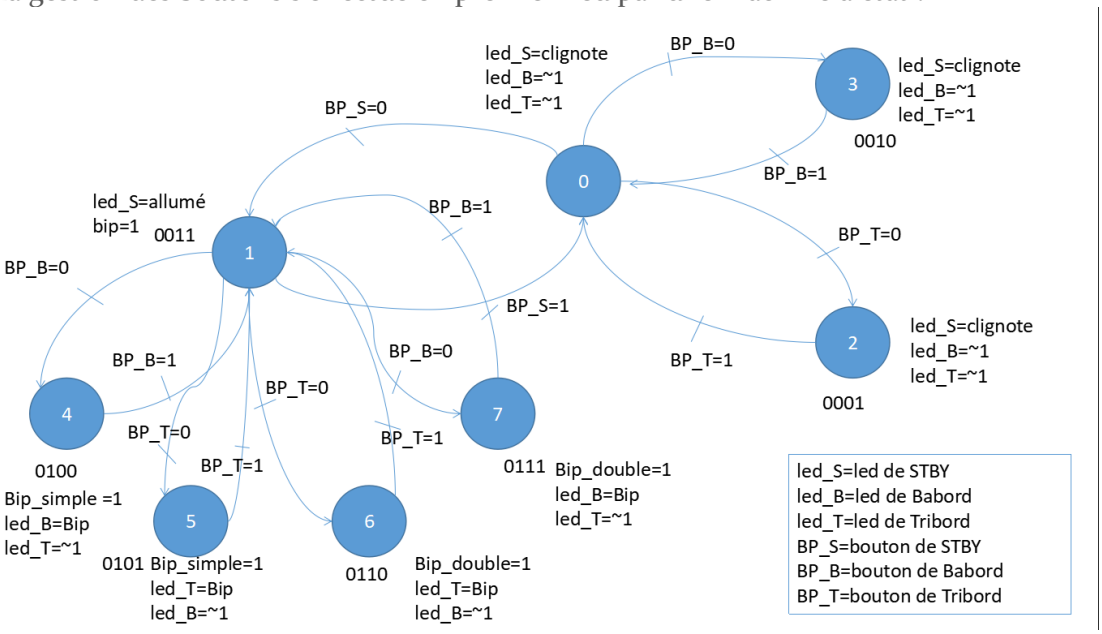


Finalement nous avons obtenu un Sopc comme s'est montré dans la figure ci-dessous à tester.



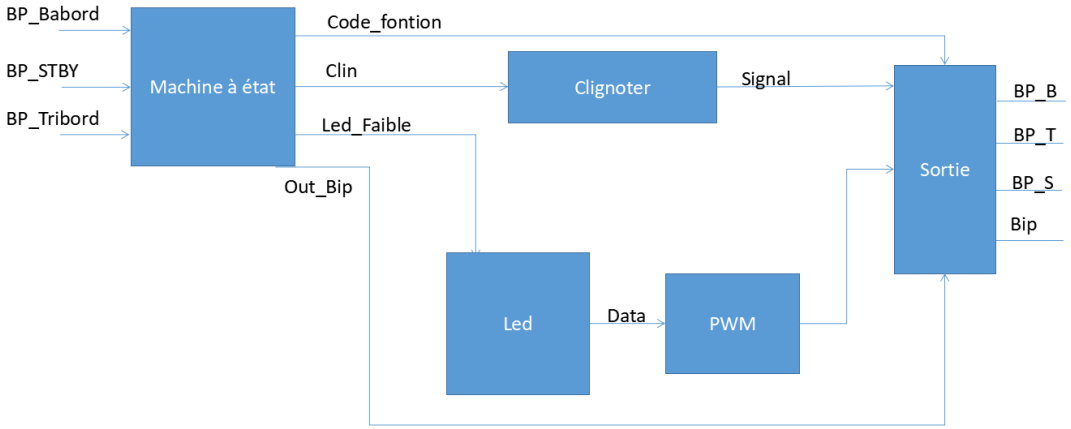
Gestion des boutons

La gestion des boutons s'effectue en premier lieu par une machine à état :



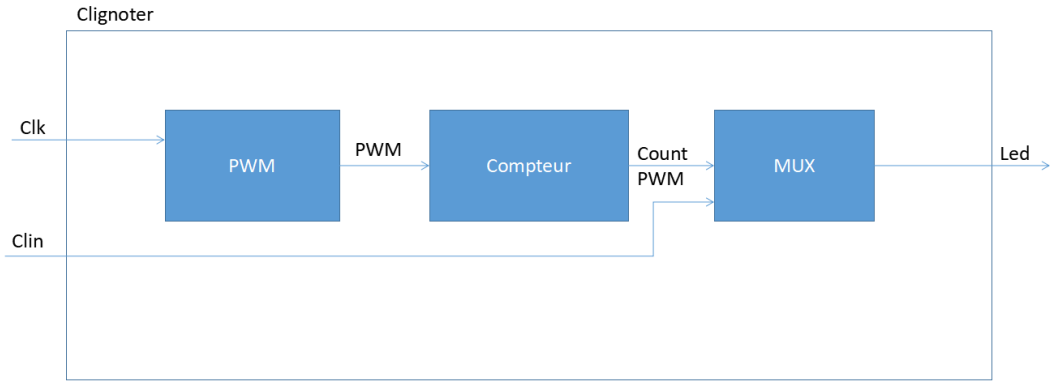
Il y a 7 états dans cette machine :

-- état 0 : pas d'action, le pilote est en veille
-- état 1 : mode manuel action vérin babord
-- état 2 : mode manuel action vérin tribord
-- état 3 : mode pilote automatique/cap
-- état 4 : incrément de 1° consigne de cap
-- état 5 : incrément de 10° consigne de cap
-- état 6 : décrément de 1° consigne de cap
-- état 7 : décrément de 10° consigne de cap



Ce schéma fonctionnel illustre le fonctionnement de notre code. Premièrement la machine à état traite les entrées et détermine les commandes nécessaires à la création des signaux de sortis. Cette partie est combinatoire. La mise au point des signaux de sorties est gérée par Clignoter qui détermine le rythme des bips, Led qui détermine l'état des leds et PWM qui contrôle l'intensité lumineuse des leds.

Le schéma ci-dessous détail le bloque clignoter. En fonction de l'horloge et du signal de l'état de l'entrée clin le clignotement peut être déclenché. Clin autorise le clignotement et l'horloge détermine sa fréquence. Le bloque PWM ralentit la fréquence de l'horloge, le compteur contrôle la durée du bip, le multiplexeur passe la led à 1 si clin et le compteur le sont.



Le code est composé de 6 process dont 3 pour la MAE :

- Duty_pwm
- Sequentiel_maj_etat
- Change_duty
- MAE

Sequentiel_maj_etat permet de mettre toutes les variables à l'état initiale et de contrôler l'horloge. Elle est également utilisée pour passer à l'état suivant dans la MAE.

Duty_pwm permet de définir l'intensité lumineuse des leds en fonction du duty.

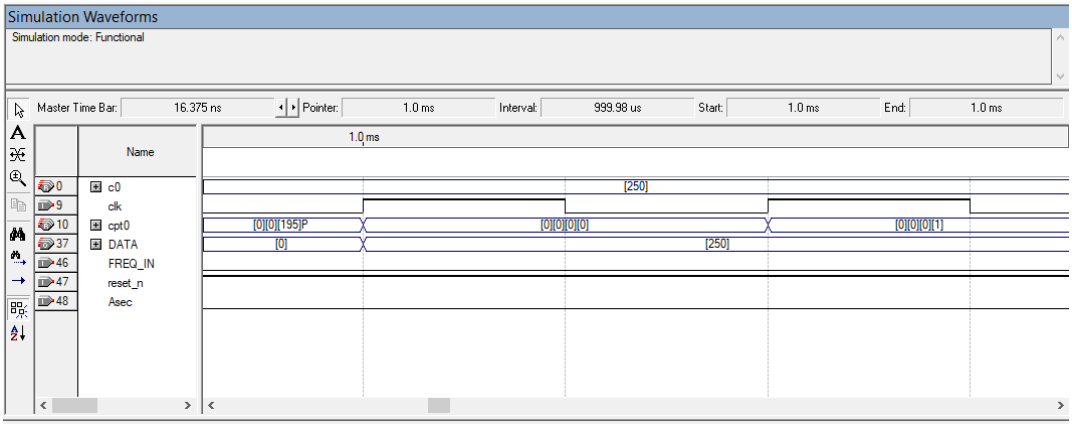
Change_duty définit les deux états de duty possible pour luminosité faible ou forte.

La **MAE** est découpée en trois parties la première contrôle le changement d'état en fonction des entrées, la deuxième gère les sorties en fonction des états et la dernière gère l'intensité lumineuse des leds en fonction des numéros des états.

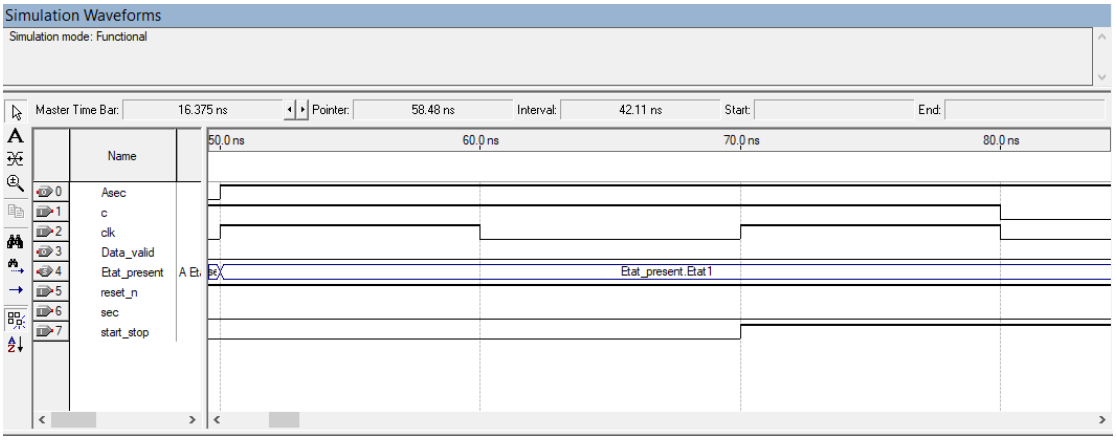
resultats

Machine à états

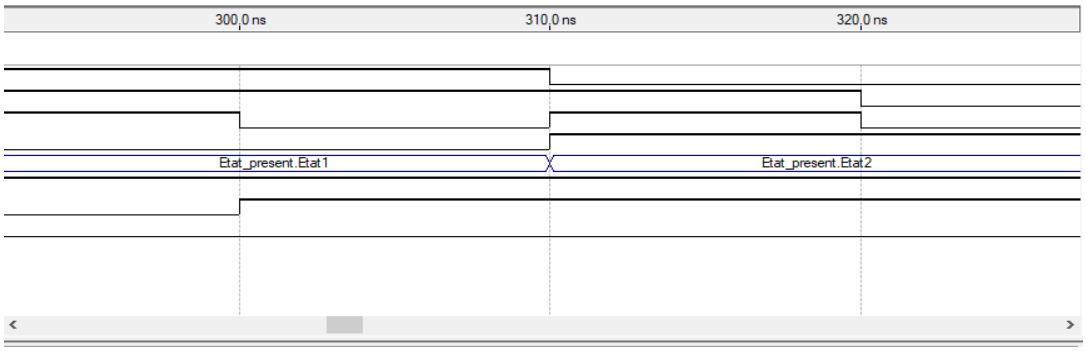
Dans la première simulation s'observe le fonctionnement de reset_n, quand sa valeur est 'o' la machine est à l'état initial ainsi que les variables en sorti Asec et data_valid.



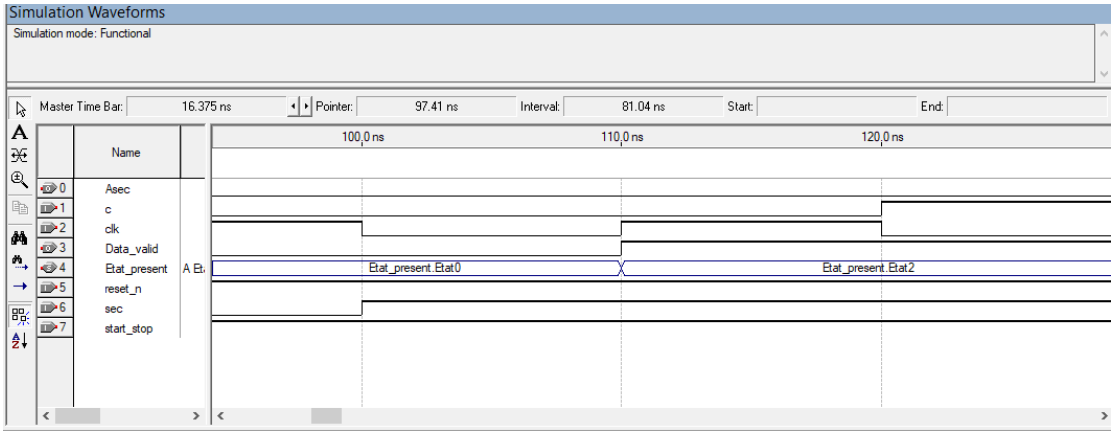
Sur la figure de la suite se montre l'évolution de la machine à l'état 'ı' quand le signal c est active, est cet état déclenche le signal Asec, ainsi se remarque que la machine á état donne la priorité à la signal c au moment que start_stop est appuyé et c est active a même temps



A continuation dans la figure s'observe que la machine passe à l'état '2' quand le signal sec est égal à '1' et donc le signal data_valid laquelle alerte qui le donne est disponible est déclenché.

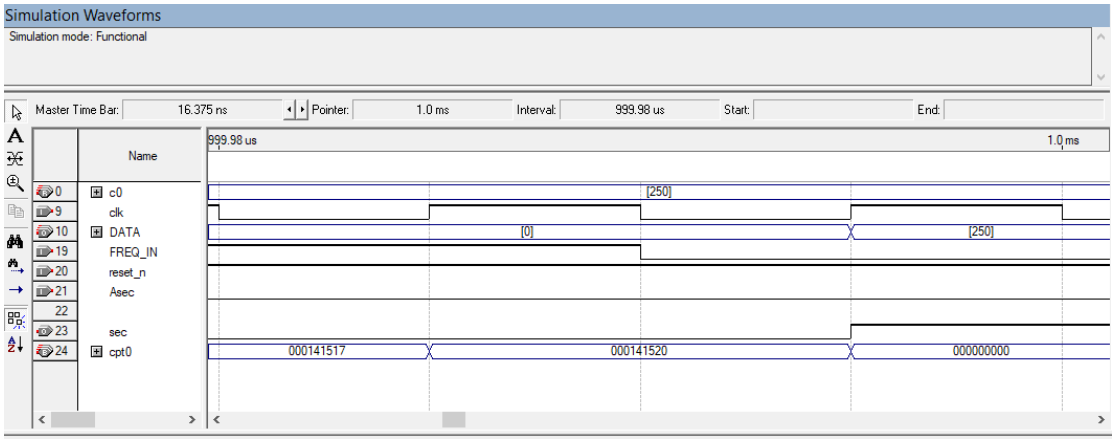


Par la suite s'observa le pas de l'état 'o' à l'état 2 quand le mode continuos est désactivé est le boutons start_stop est appuyé, à ce moment là encore une fois le signal data_valid sera déclenché

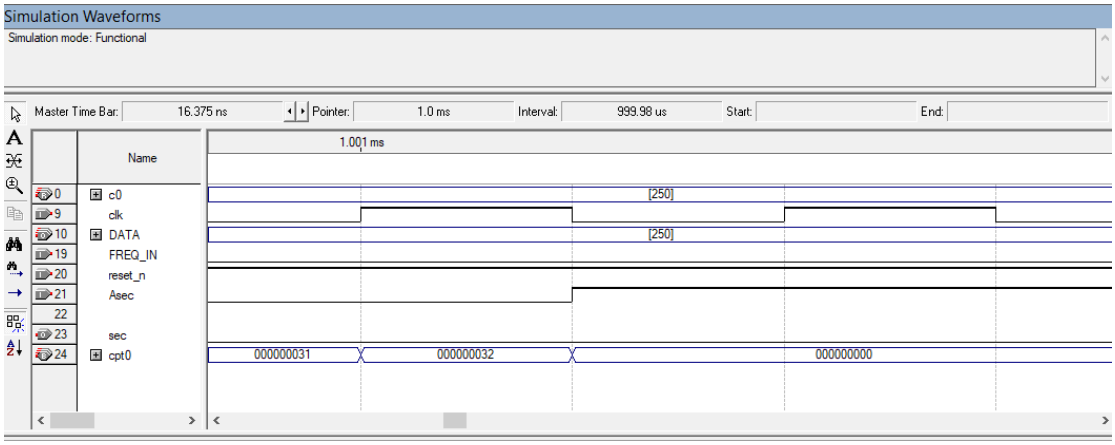


Compteur de Fréquence, et compteur de secondes

Sur la simulation suivante est ressemblé le fonctionnement de le deux fonctionnes. Pour cette simulation le coaptateur de secondes va à compter microsecondes afin de que la simulation soit généré plus rapidement. S'observe un coaptateur de flancs 'co' d'horloge et un coaptateur de flanc 'cpto' de le signal à mesuré. 'FREQ_IN'. Les flancs de le signal FREQ_IN son mesure pendant une seconde, après le valeur et transmis a la variable DATA, et quand le seconde est coule le signal 'sec' est activé.

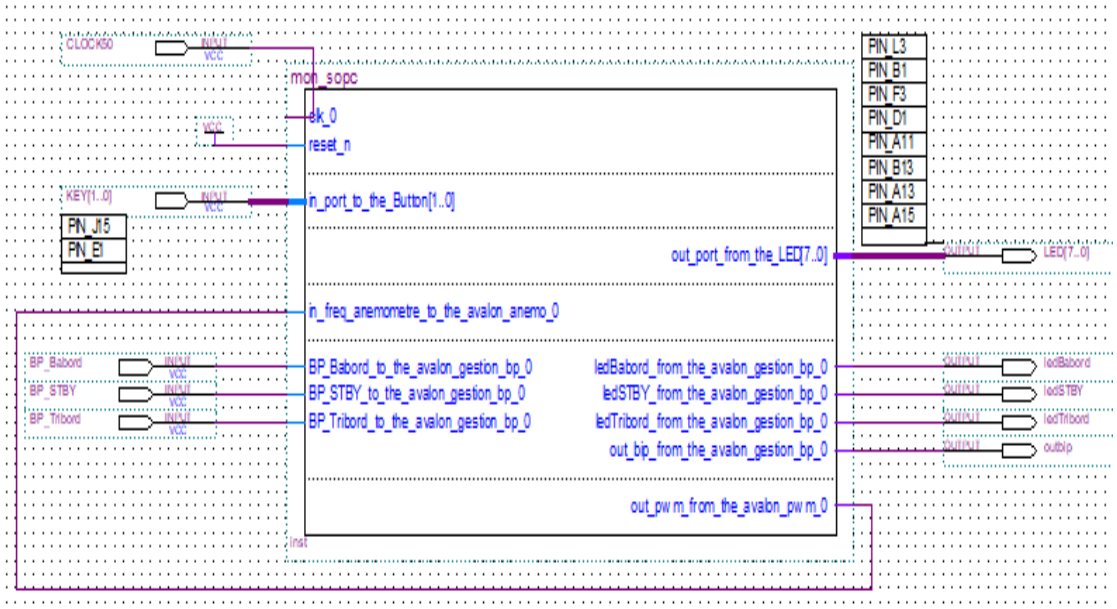


A continuation sur la figure suivante, se peut observer, si le signal Asec est activé le compteur de secondes revient à o afin de pouvoir mettre en marche le mode continuos sur la machine à états.



Sopc + interface anémomètre et gestion de boutons

En dernier nous avons intégré les deux interfaces sur le Sopc a l'aide de l'outil Sopc Builder, et le résultat se montre sur la figure ci-dessous.



Nous avons programmé le Sopc sur la carte DEo Nano et en suit nous avons téléchargé le programme écrit sur Nios II.

Sur la figure ci-dessous se montre le fonctionnement du Sopc avec les deux interfaces intégrées, ici on représente la mesure de la fréquence come une mesure de la vitesse du vent.