
 UNIVERSITÉ TOULOUSE III PAUL SABATIER	M2 EEA PARCOURS SYSTEMES ET MICROSYSTEMES EMBARQUES <i>Module : Synthèse et mise en œuvre des systèmes</i>	 <i>Année universitaire : 2018-2019</i>
---	--	---

Rapport BE :

Pilote de barre franche

- Gestion anémomètre**
- Gestion boutons poussoirs**

Encadré par : Pr.THIERRY PERISSE

Réalisé par : OUkZIZ Yasmine et MAHRAZ Anass

Table des matières

I.	Introduction.....	3
1.	Définition : System Embarqués :.....	3
2.	Architecture Systemes embarqués :.....	3
II.	Pilote de barre franche :.....	3
1.	Presentation.....	3
2-	Modes de fonctionnement	4
III.	Objectif de bureau d'étude :	4
1.	Cahier de charge :.....	4
2.	Outils de conception et développement.....	5
V.	Programme et réalisation	6
1.	Gestion anémomètre :.....	6
1.1	Analyse fonctionnelle :.....	6
1.2	Synthèse VHDL des fonctions :.....	6
1.3	Simulation partielle :	7
1.4	Synthèse VHDL globale du circuit :.....	7
2.	Gestion bouton poussoir :.....	8
2.1	Analyse Fonctionnelle :.....	8
2.2	Synthèse VHDL des fonctions :.....	9
2.3	Synthèse VHDL globale du circuit :.....	10

I. Introduction

1. Définition : System Embarqués :

Un système embarqué est un système électronique et informatique autonome qui est utilisé pour réaliser une tâche prédéfinie parfois en temps réel.

L'expression "système embarqué" peut s'appliquer sur le matériel informatique mais aussi les logiciels utilisés. La technologie des systèmes embarqués est utilisée dans plusieurs domaines comme l'astronautique, l'aéronautique, l'automobile, l'industrie, l'informatique....etc.

2. Architecture Systemes embarqués :

On peut également voir sur la figure que l'unité de calcul centrale (CPU) peut interagir, au sein du système embarqué, avec différents éléments tels que des convertisseurs A/D et D/A (convertisseur analogique vers numérique et inversement : A/D pour Analog-to-Digital et D/A pour Digital-to-Analog), de la mémoire et d'autres unités de calcul de type FPGA ou ASIC (FPGA pour Field-Programmable Gate Array est un circuit intégré reprogrammable et un ASIC pour Application-Specific Integrated Circuit est au contraire spécifique à une fonction et non modifiable).

La figure ci-dessous illustre les différentes parties en lesquelles se déclinent les circuits logiques qui constituent la base de tout circuit embarqué.

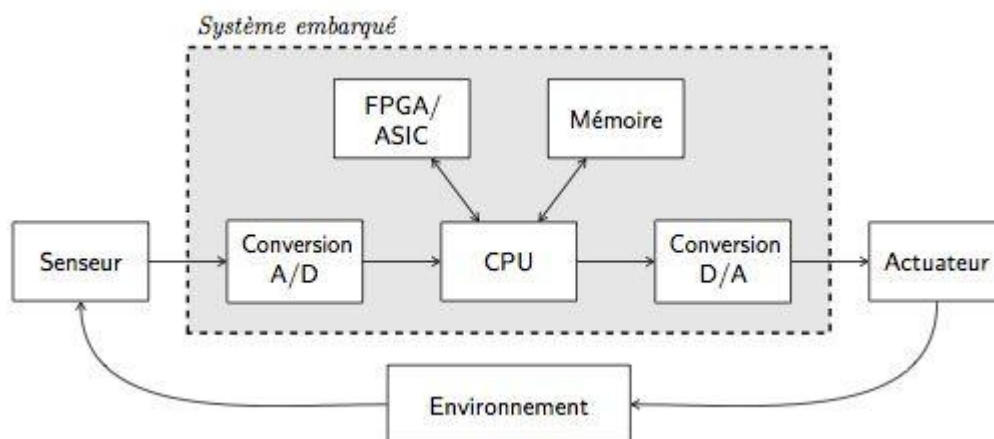


Figure 1 Le hardware d'un système embarqué (cadre gris) est souvent utilisé dans un mode "hardware in a loop", c'est-à-dire dans une boucle qui passe à travers le système embarqué et l'environnement.

CPU: Computer Processing Unit.

DSP: Digital Signal Processor.

CPLD: Complex Programmable Logic Device.

FPGA: Field Programmable Gate Array.

ASIC: Application Specific Integrated Circuit.

II. Pilote de barre franche :

1. Présentation

Un pilote automatique pour voilier est un équipement électrique ou hydraulique destiné à maintenir le cap d'un voilier à la place d'un équipier. Ces pilotes automatiques sont très utiles aux navigateurs solitaires ou en équipage réduit. Le pilote est constitué de trois éléments

principaux : un compas, une unité électronique et une unité de puissance. Sur tous les pilotes de la nouvelle génération, le compas est électronique ; il donne continuellement à l'unité de traitement le cap suivi par le bateau. Cette même unité a comme consigne le cap que l'on souhaite suivre. Elle compare en permanence ces deux caps, s'ils ne sont pas identiques, elle donne l'ordre à l'unité de puissance d'agir sur la barre pour ramener le bateau sur son cap. L'unité de puissance pour les pilotes pour barre franche est un vérin linéaire. Ce vérin a une extrémité fixée sur le banc de cockpit, l'autre sur la barre. Toute variation de cap lui est transmise et il agit en conséquence sur la barre.

2-Modes de fonctionnement

L'appareil fonctionne selon deux modes principaux : le mode veille et mode pilote automatique.

- Mode manuel : appelé aussi mode veille. Dans ce mode, le vérin le vérin peut être entré et sorti manuellement en appuyant sur les touches fléchées Bâbord (←) et Tribord (→), ce qui permet d'utiliser le Tillerpilot comme système de "barre motorisée".
- Mode pilote automatique : Le pilote est d'abord verrouillé sur le cap actuel (au moment du passage au mode pilotage automatique). Le réglage du cap de 1° ou de 10° , est réalisé grâce au mouvement de la barre vers Babord (gauche) ou tribord (droite) et s'effectue à l'aide des boutons correspondant du clavier



Figure 2 Clavier du Tillerpilot

III. Objectif de bureau d'étude :

1. Cahier de charge :

L'objectif de ce bureau d'étude est de concevoir le pilote de barre franche sous forme d'un système sur puce programmable SOPC (System On Programmable Chip) décrite à l'aide du langage de description de Hardware VHDL (Very High Speed Hardware Description Language) en suivant les étapes suivantes :

- ✓ analyse de spécifications et découpage fonctionnel du système choisi.
- ✓ Conception de circuits d'interfaces numériques en VHDL (conception, simulation, vérification sur maquette)
- ✓ Notion de Co-design et règles de conception - interfaçage avec bus microprocesseur (NIOS + Altera Avalon)
- ✓ conception d'un SOPC et intégration D'IP (Intellectual Properties) propriétaires et fournisseurs tiers - notions de simulation « Hardware In the Loop » - validation du SOPC en simulation (pour parties)

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

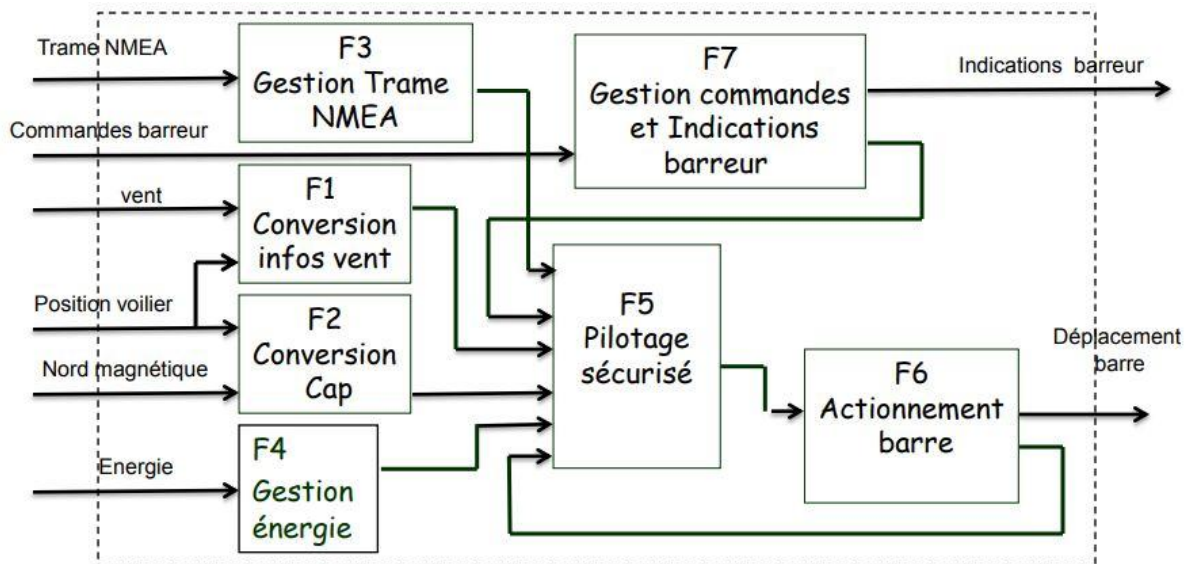


Figure 3 les systemes de pilote

2. Outils de conception et développement

• Technologie FPGA :

Un FPGA est un circuit en silicium reprogrammable. À l'aide de blocs logiques préconstruits et de ressources de routage programmables, il est possible de configurer ce circuit afin de mettre en œuvre des fonctionnalités matérielles personnalisées,

Si les FPGA rencontrent un tel succès dans tous les secteurs, c'est parce qu'ils réunissent le meilleur des ASIC et des systèmes basés processeur.

Les cinq principaux atouts de la technologie FPGA sont

- ✓ Performances
- ✓ Temps de mise sur le marché
- ✓ Coût
- ✓ Fiabilité
- ✓ Maintenance à long terme

• Conception conjointe

Les circuits FPGA permettent la conception conjointe matérielle/logicielle appelée « co-design », où une fonctionnalité complexe allie à la fois une logique programmée pour sa flexibilité et une logique câblée pour ses performances et ce grâce à la fonctionnalité SOPC (system on programmable chip) builder que propose Quartus II.

• Carte DE0 :

La carte DE0-Nano présente une plate-forme de développement FPGA compacte adaptée au prototypage de conceptions de circuits telles que les robots et les projets "portables".

Spécifications : Altera Cyclone IV EP4CE22F17C6N

• Softcore NIOS II

Le NIOS est un processeur softcore propriétaire d'Altera. Il est basé sur un cœur 32 bits et doté du bus Avalon.

Dans ce projet, l'architecture du processeur est décrite à l'aide du langage VHDL (Very High Speed Hardware Description Language) et synthétisé sur FPGA (sur carte DE0).

• logiciel Quartus v11 et v13

C'est un logiciel permettant la gestion complète d'un flot de conception CPLD ou FPGA. Ce logiciel permet de faire une saisie graphique ou une saisie texte (description VHDL) d'en réaliser une simulation, une synthèse et une implémentation sur cible reprogrammable, on a utilisé la version 9 pour développer l'ensemble de notre circuit VHDL et la version 11 pour les intégrer dans le SOPC (System On Programmable Chip).

V. Programme et réalisation

1. Gestion anémomètre :

Le but de ce projet est de développer un circuit « gestion anémomètre » permettant de mesurer un signal de fréquence variable de 0 à 250 Hz à un signal numérique codé sur 8 bits.

1.1 Analyse fonctionnelle :

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

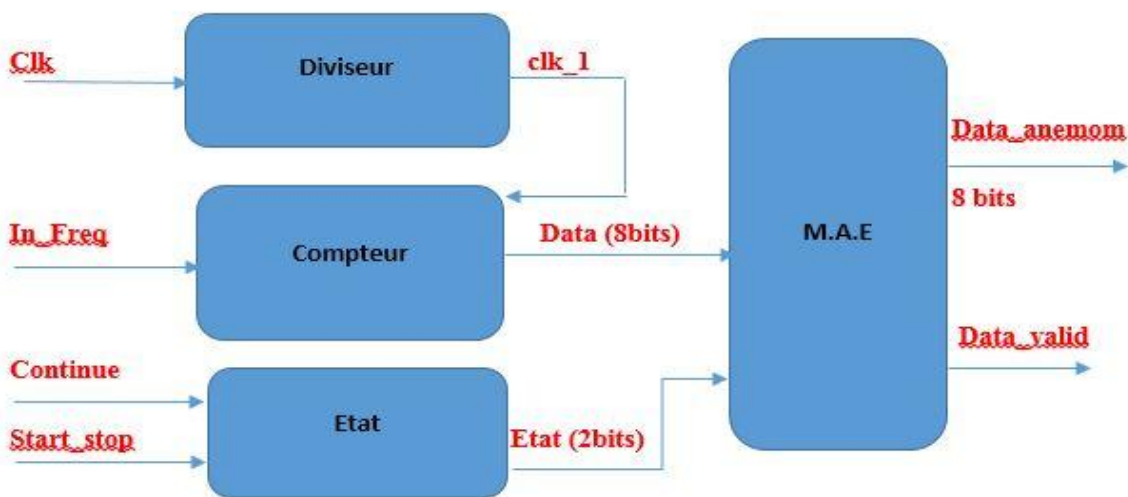


Figure 4 Schéma fonctionnel du circuit « gestion_anemometre ».

- **Diviseur** : C'est une fonction qui permet de générer une horloge de 10KHz pour synchroniser les différents process de notre circuit.
- **Compteur** : C'est une fonction qui permet de compter le nombre de fronts montant du signal numérique in_freq dans le but de mesurer sa fréquence.
- **M.A.E** : C'est une fonction qui gère les modes de mesure de la fréquence in_freq, qui sera mémorisée dans une variable de sortie codée de 8 bits nommée data_anemometre, lorsqu'une mesure est valide, le circuit met sa sortie data_valid à 1 sinon elle est à 0.

1.2 Synthèse VHDL des fonctions :

Le circuit synthétisable « gestion_anemometre » a été développé en langage VHDL pour une carte DE2 (FPGA, Cyclone II, EP2K35F672C6) et suivant une approche hiérarchique et modulaire (les sous-fonctions sont implémentées par des process).

- **Génération base fréquence_10Khz** : Pour implémenter cette fonction, nous avons développé un diviseur de fréquence permettant de générer une horloge 10Khz à partir de l'horloge interne du FPGA (50Mhz).
- **Comptage fronts in_freq** : Cette fonction a été implémentée sous forme d'un compteur de 8 bits avec un process sensible au signal in_freq

- **M.A.E** : Cette fonction a été implémentée sous forme d'une machine à état avec un process sensible à notre horloge de synchronisation clk1. Ce processus prend en charge la transition entre les différents états pour assurer le fonctionnement convenable des des deux modes de mesure (monocoup et continu).

1.3 Simulation partielle :

Pour tester la mise en œuvre des fonctions ci-dessus, nous avons fait pas mal de simulations, cela nous a permet de valider le bon fonctionnement des fonctions développé ainsi d'ajouter parfois des améliorations dans leurs implémentations (par exemple dans la machine à état du FS3, nous avons fusionné les états 2 et 3 en un seul état).

Ici, vous trouveriez un exemple de simulation de la fonction « Compteur » :

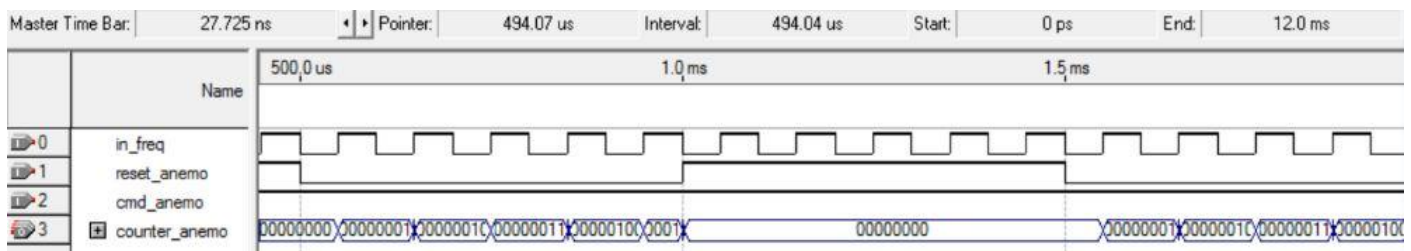


Figure 5 Simulation de la fonction « Compteur ».

- **Interprétation** : Sur le graphe de la simulation, nous arrivons à lire le nombre des fronts montants de la fréquence à mesurée, ce nombre est réinitialiser à 0 quand le reset devient à 1, ainsi il est activé quand l'entrée de commande est à 1. Donc on a bien vérifié le fonctionnement de notre compteur8 de fronts.

1.4 Synthèse VHDL globale du circuit :

Après avoir développé et validé l'implémentation des différentes fonctions, on les intégrer au fur et à mesure dans un projet « top level » appelé « gestion_anemometre » cela dans le but faire une synthèse globale de notre circuit.

- **Mise en œuvre et simulation de l'interface Avalon :**

Pour assurer l'interconnexion entre le processeur (NIOS II) et le circuit « gestion_anemometre » une interface Avalon a été développer, elle permet de lire et d'écrire sur quelques entrées/sorties de notre circuit selon les signaux de contrôle de son bus associé (chip select, write_n, adresse), le tableau suivant résume les différentes entrées sorties de l'interface Avalon.

Entrées	Role
Clock	Horloge de synchronisation de notre bus et notre module
ChipSelect	Activation du circuit
Write_n	Activation de lecture ou écriture
Adress	Adresse cible pour la lecture ou écriture
Reset_n	Remise à zéro
In_freq	Fréquence à mesurer
Writedata	Données d'écriture
Sortie	Role
Readdata	Données de lecture

Tableau 1 entrees et sortie de l'interface avalon

Afin de télécharger le circuit sur la carte DE0 nous avons utilisé le SOPC builder afin de créer le microprocesseur et les éléments périphériques

Use	Conn...	Name	Description	Clock	Base	End	IRQ	Tags
<input checked="" type="checkbox"/>		<input type="checkbox"/> cpu_0	Nios II Processor	[clk]				
		instruction_master	Avalon Memory Mapped Master	clk_0				
		data_master	Avalon Memory Mapped Master	[clk]				
		jtag_debug_module	Avalon Memory Mapped Slave	[clk]	0x00004800	0x00004fff	IRQ 0	IRQ 31
<input checked="" type="checkbox"/>		<input type="checkbox"/> onchip_memory2_0	On-Chip Memory (RAM or ROM)	[clk1]				
		s1	Avalon Memory Mapped Slave	clk_0	0x00002000	0x00003fff		
<input checked="" type="checkbox"/>		<input type="checkbox"/> jtag_uart_0	JTAG UART	[clk]				
		avalon_jtag_slave	Avalon Memory Mapped Slave	clk_0	0x00005000	0x00005007		16
<input checked="" type="checkbox"/>		<input type="checkbox"/> sysid_0	System ID Peripheral	[clk]				
		control_slave	Avalon Memory Mapped Slave	clk_0	0x00005008	0x0000500f		
<input checked="" type="checkbox"/>		<input type="checkbox"/> anemo_0	anemo	[clock]				
		avalon_slave_0	Avalon Memory Mapped Slave	clk_0	0x00005010	0x00005017		

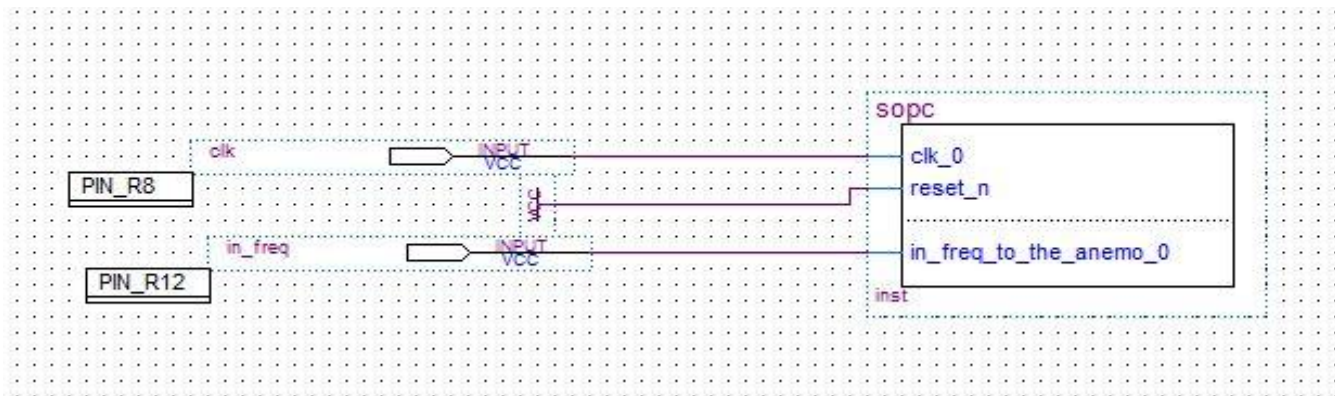


Figure 6Circuit d'interface Avalon/ anémomètre.

2. Gestion bouton poussoir :

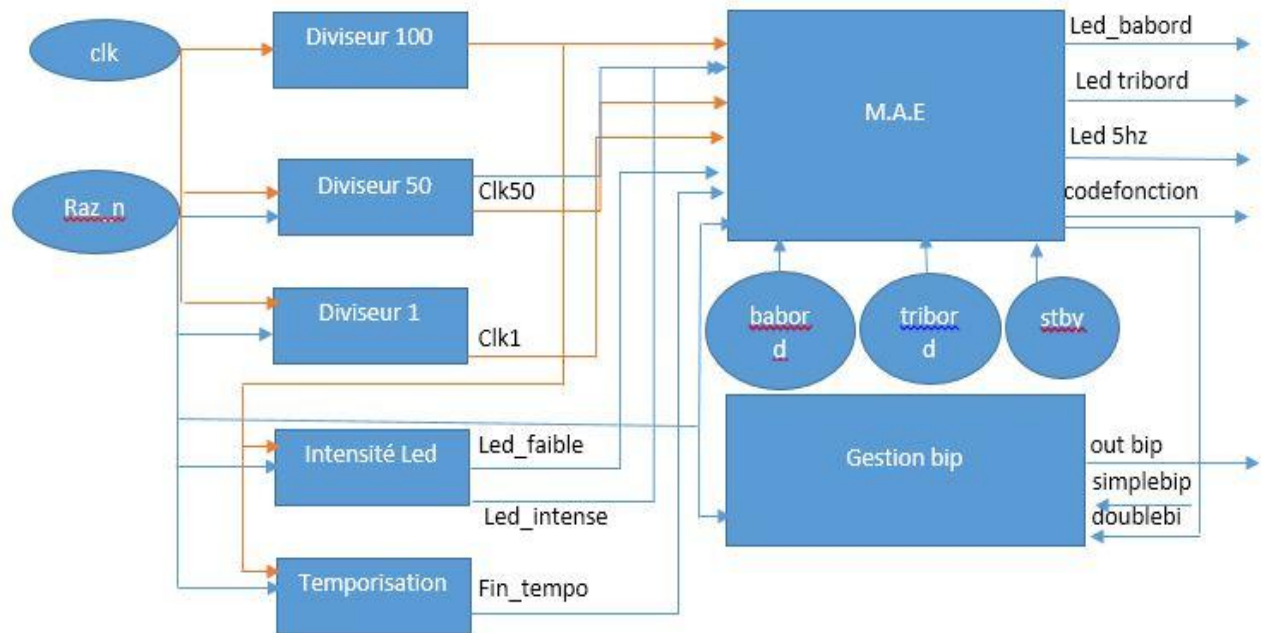
Dans cette partie on va développer un circuit de gestion des boutons poussoirs qui permet à la fois de piloter le vérin à travers les trois boutons selon le mode de fonctionnement at aussi d’indiquer à l’utilisateur le mode de fonctionnement et la validation de chaque tâche à travers les trois Leds (Bâbord, Tribord et STBY) et le bip sonore.

Le circuit doit fonctionner selon les huit modes suivants :

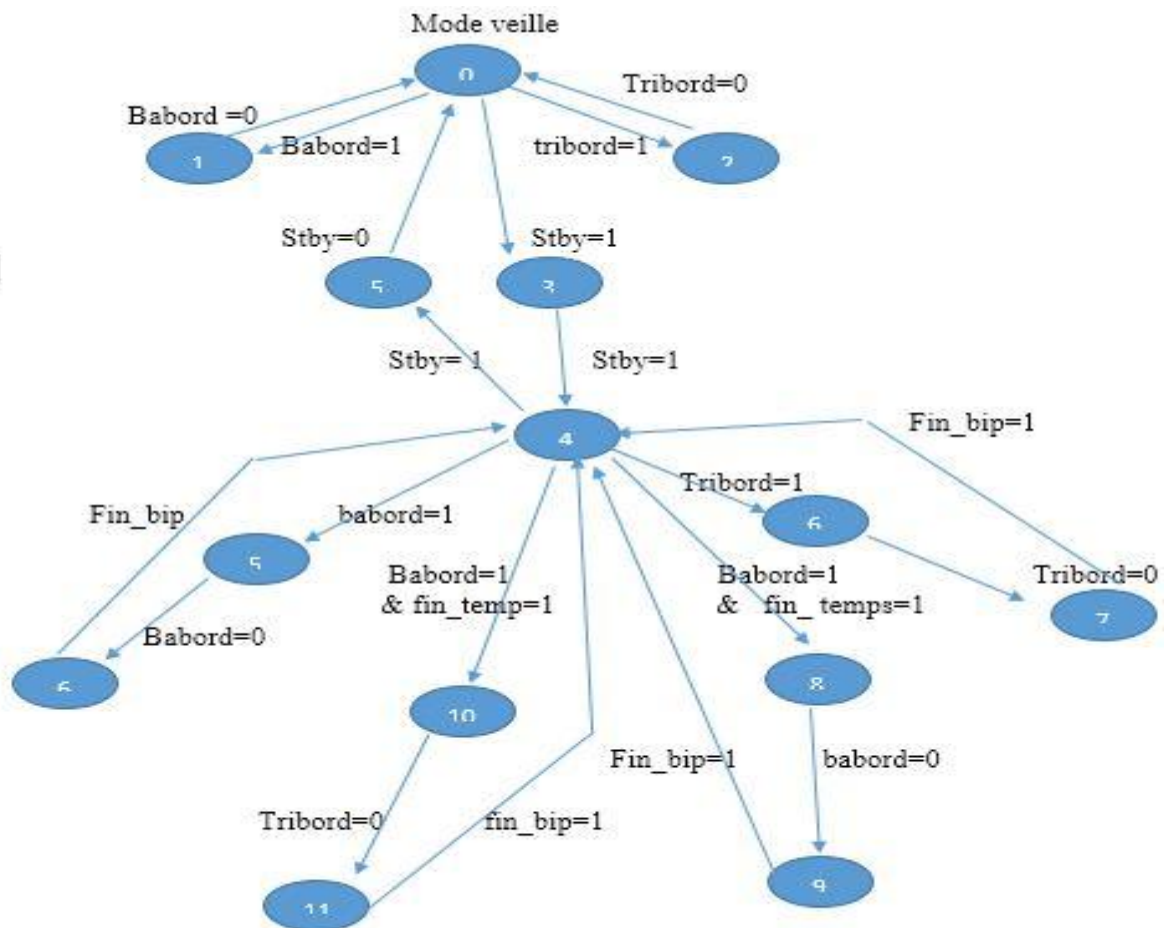
Mode	action	codefonction
1	Pas d’action	‘0000 ‘
2	Mode manuel : action vérin à gauche, le BP_Babord est appuyé.	‘0001’
3	Mode manuel : action vérin à droite, le BP_Tribord est appuyé.	‘0010’
4	Mode pilote automatique : Le bouton STBY est appuyé	‘0011’
5	Mode cap 1° Bâbord : incrémentation du cap 1° ainsi que le bouton Bâbord est relâché avant la fin de la temporisation.	‘0100’
6	Mode cap 1° Tribord : incrémentation du cap 1° ainsi que le bouton Tribord est relâché avant la fin de la temporisation	‘0111’
7	Mode cap 10° Bâbord : incrémentation du cap 10° ainsi que le bouton Bâbord reste appuyé après la fin de la temporisation	‘0101’
8	Mode cap 1° Tribord : incrémentation du cap 10° ainsi que le bouton Tribord reste appuyé après la fin de la temporisation	‘0110’

2.1 Analyse Fonctionnelle :

- Schéma fonctionnel



- Machine à état



2.2 Synthèse VHDL des fonctions :

Ce circuit « gestion_Boutons_poussoir » est composé de six fonctions principales :

- Génération base fréquence_100Hz :**

Ce bloc permet de générer une horloge de 100Hz à partir de l'horloge interne qui vaut 50MHz, il est utilisé pour synchroniser des différents process de notre programme.

- **Génération base fréquence_1Hz :**
génération d'une horloge 1Hz à partir d'horloge 100 Hz, ce bloc est utilisé dans ce projet pour clignoter la Led STBY en mode veille.
- **Génération base fréquence_50Hz :**
génération d'une horloge 1Hz à partir d'horloge 100 Hz, ce bloc est utilisé pour allumer les trois Leds .
- **Génération d'une temporisation 3S :**
Génération de la temporisation de 3 secondes afin de différencier les appuis longs et courts des boutons Bâbord et Tribord.
- **Machine à état génération du Bip :**
Ce bloc représente une machine à état qui a pour but de générer des Bip et double Bip selon la durée d'appui exercée sur les deux boutons poussoirs.
- **Machine à état gestion des boutons poussoirs :**
Ce bloc représente une machine à état de gestion boutons poussoirs qui est le cœur de notre programme, il est synchronisé par une horloge de 100Hz. Il assure la transition d'un état à l'autre selon les modes de fonctionnement et aussi de générer les valeurs de codeFonction en sortie.

2.3 Synthèse VHDL globale du circuit :

- **Mise en œuvre et simulation de l'interface Avalon :**

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

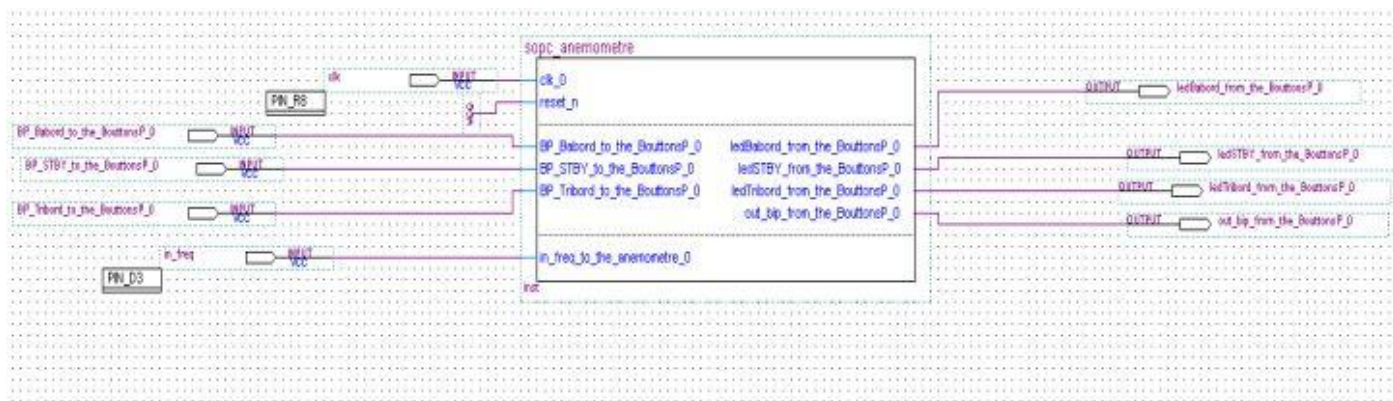
Registre	adresse	Type	Bits concernés
config	0	R/W	b0=raz_n
code	1(4)	R/W	b3..b0= Code_Fonction

Pour l'intégration du SPOC nous avons suivis les mêmes étapes pour l'anémomètre, nous avons ajouté le nouveau composant qui est BouttonP_0 Les composants ajoutés sont montrés dans la figure suivante :

Use	Conne...	Module Name	Description	Clock	Base	End	Tags	IRQ
<input checked="" type="checkbox"/>		cpu_0	Nios II Processor					
		instruction_master	Avalon Memory Mapped Master	clk_0				
		data_master	Avalon Memory Mapped Master		IRQ 0	IRQ 31		
		jtag_debug_module	Avalon Memory Mapped Slave		0x00010800	0x00010fff		
<input checked="" type="checkbox"/>		SRAM	On-Chip Memory (RAM or ROM)	clk_0	0x00003000	0x0000cfff		
		s1	Avalon Memory Mapped Slave					
<input checked="" type="checkbox"/>		jtag_uart_0	JTAG UART	clk_0	0x00011000	0x00011007		
		avalon_jtag_slave	Avalon Memory Mapped Slave					
<input checked="" type="checkbox"/>		sysid	System ID Peripheral	clk_0	0x00011008	0x0001100f		
		control_slave	Avalon Memory Mapped Slave					
<input checked="" type="checkbox"/>		anemometre_0	anemometre	clk_0	0x00011001	0x00011008		
		avalon_slave_0	Avalon Memory Mapped Slave					
<input checked="" type="checkbox"/>		BouttonsP_0	BouttonsP	clk_0	0x00001001	0x00001008		
		avalon_slave_0	Avalon Memory Mapped Slave					

Le résultat de la génération du SOPC est un ensemble de fichiers VHDL qui décrivent le circuit du système

g n r , la figure ci-dessous montre la g n ration SPOC du bloc global.



IV. Conclusion :

Dans le cadre de ce bureau d'étude pilote barre franche nous avons attaqué la conception et l'implantation sur d'une architecture d'un pilote de barre franche. La procédure abordée contient une première étape dans laquelle les besoins et le contexte ont été analysés pour identifier les différentes interfaces du système. Ensuite, dans l'étape de conception le système a été décomposé en un ensemble de blocs fonctionnels.

Ce travail a également été intéressant il nous a permis d'acquérir des connaissances solides sur le développement des circuits électroniques et des systèmes complexes avec le langage VHDL.

D'autre part ce projet nous a permis d'accroître nos compétences dans le domaine de réalisation des systèmes tout en appliquant par ordre les 4 Etapes nécessaires :

- Analyse du cahier des charges.
- Analyse fonctionnelle.
- Conception.
- Mise en œuvre.
- Test et la validation.

Ces techniques sont très intéressantes pour nous en tant que futures ingénieurs systèmes embarqués.

Avant de finir, nous tenons remercier Mr Jean Louis BOIZARD pour son encadrement durant les séances de ce bureau d'étude.