Bus WISHBONE

L'architecture de bus *WISHBONE* est implanté à partir d'un code source en *VHDL*. Pour faciliter l'expansion et la réutilisation de l'architecture, on a respecté certaines règles recommandées suivantes :

- Les fichiers et les modules qui se réfèrent au bus *WISHBONE* sont nommés en ajoutant le préfixe ***wb_***.
- Le module de l'architecture qui gère le bus *WISHBONE* est labellisé avec le préfixe ***ma_*** pour désigner le "*master*". Tandis que tous les autres modules esclaves sont désignés tous avec un préfixe ***sl_*** pour "*slave*".

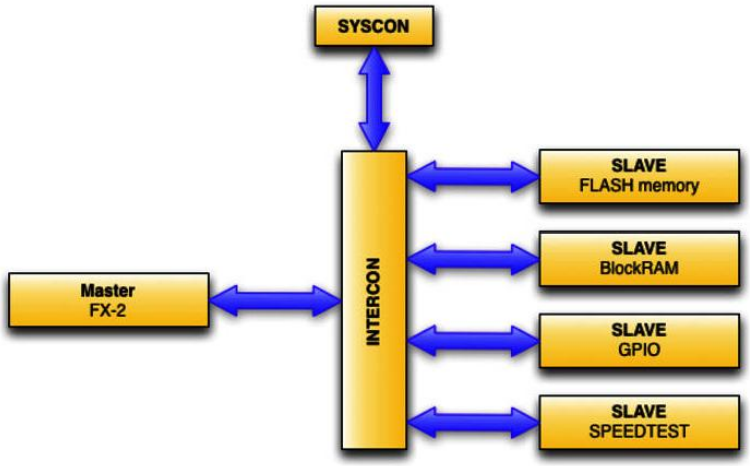


Figure-5. 11 : Structure d'un bus intégré WISHBONE

Les codes qui instancient les modules majeurs qui constituent l'architecture sont déjà disponibles et contenus dans des fichiers *VHDL* suivants :

wb_syscon.vhd:

Cette entité fournis les signaux d'horloge *SYSCL* et de *RESET* pour le bus *WISHBONE*.

N.B: le bus *WISHBONE* est totalement synchrone avec le système de registres tampons "*Slave FIFO*" du microcontrôleur *CY7C68013A*.

```
entity wb_syscon is
 generic
 (
 nr_of_dbgports : positive := 1
 );
```

```
port
 (
 SYSCON_O : out rec_syscon_port;

 sys_rst_i : in std_logic;
 sys_clk_i : in std_logic;
 debug : inout std_logic_vector((nr_of_dbgports-1) downto 0)
 );

end wb_syscon;

architecture RTL of wb_syscon is
 signal syscon : rec_syscon_port := syscon_default;
 signal sys_rst : std_logic := '1';
 signal sys_clk : std_logic := '0';

 begin

 -- connect ports to internal signals----------------------------

 SYSCON_O <= syscon;
 sys_rst <= sys_rst_i;
 sys_clk <= sys_clk_i;

 debug <=
 (
 others => 'Z'
 );

 -- architecture implementation ----
 syscon.rst <= sys_rst;
 syscon.clk <= sys_clk;
 end RTL;
```

Le Bus AXI

Le système de bus abstrait *AXI* est la technologie adoptée par Xilinx pour l'interconnexion des *Cores IP*.

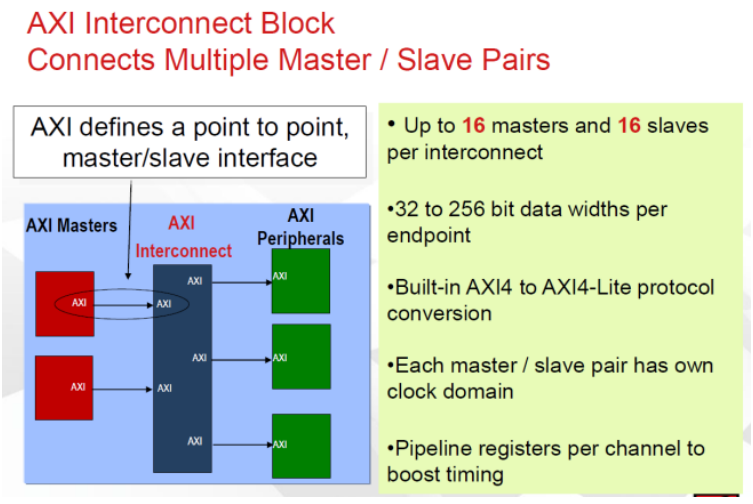


Figure-5. 12 : Caractéristiques d'un bus intégré AXI

Cette structure de bus est disponible dans la configuration des composants *ZYNQ* série-7000 est a été étudié pour la migration du système d'acquisition utilisant le *ZEDBOARD*.

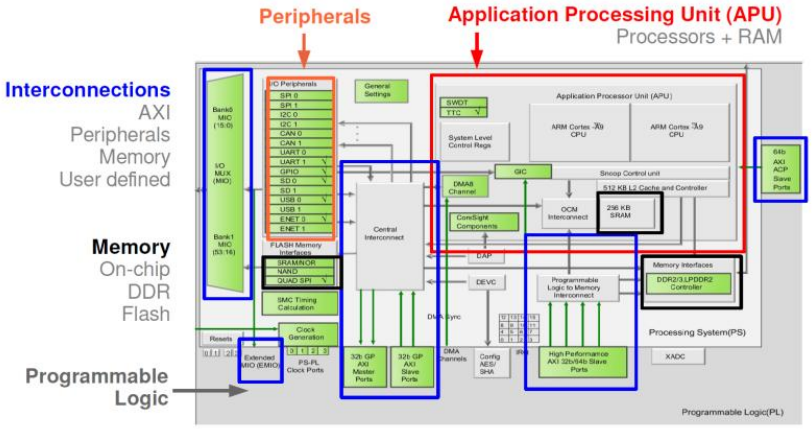


Figure-5. 13 : Implantation du bus intégré AXI sur le processeur ZYNQ

ZYNQ-VIVADO

Plusieurs modules d'interface *AXI* sont disponibles sur la plateforme *VIVADO de Xilinx* pour le développement du projet avec le *ZEDBOARD*.

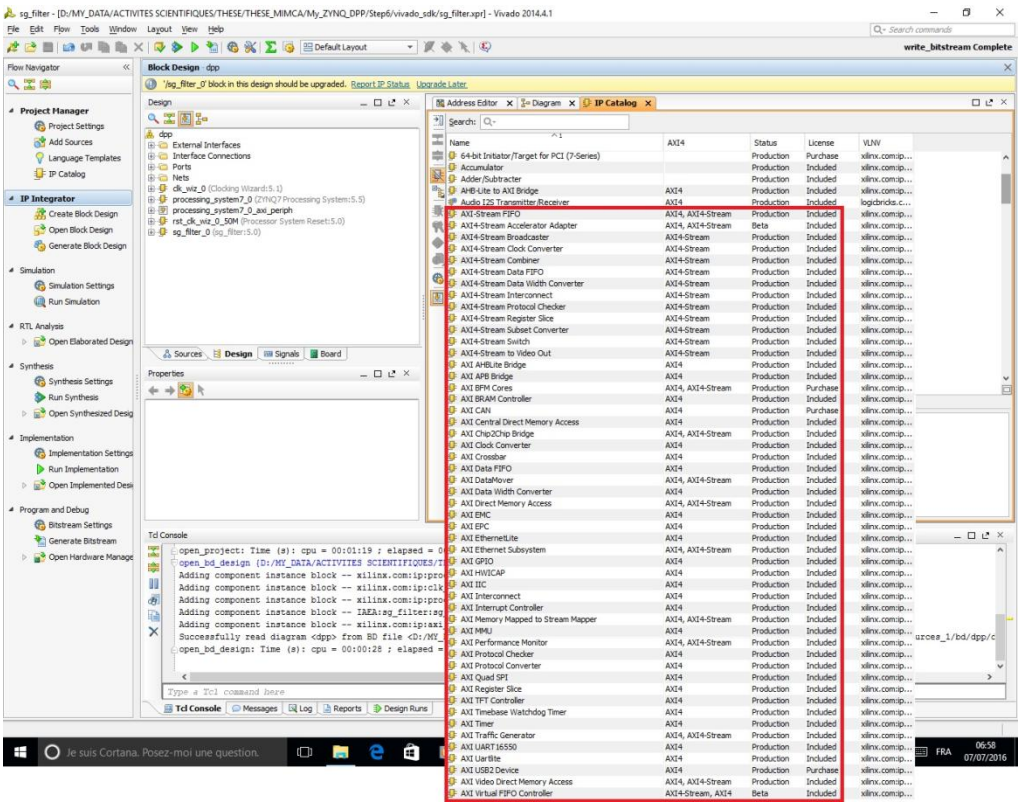


Figure-5. 14 : Modules IP disponibles sur VIVADO

Le module *IP AXI* interconnect est utilisé pour interfacer le module IP "*Zynq Processing system*" avec le module IP Xilinx sg_filter, module abstrait construit à partir de *MATLAB*.

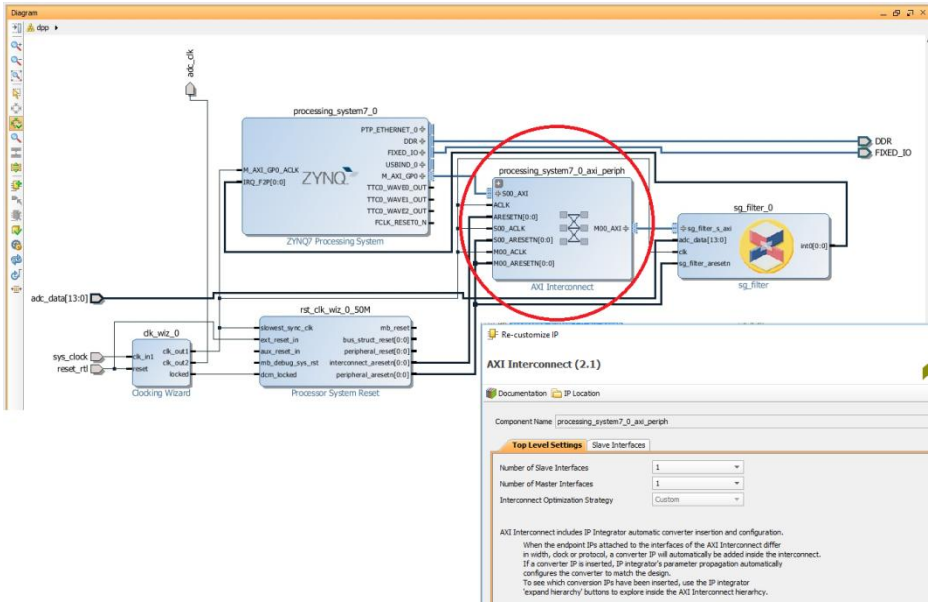


Figure-5. 15 : Module IP AXI implanté dans le projet de construction du système de traitement des signaux numériques sur ZYNQ

Le module d'interface IP "*AXI-Interconnect"* possède toutes les ressources nécessaires pour la séquence et le protocole pour la gestion des données, telles que l'horloge, les ports d'interface des données.

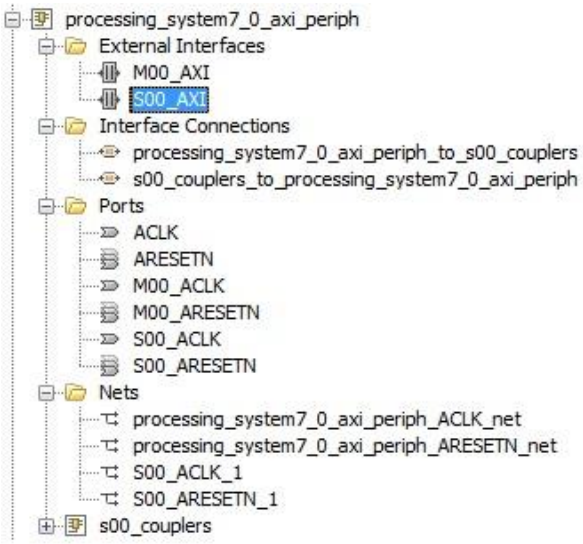


Figure-5. 16 : Ressources disponibles pour le module IP AXI

2. SYNTHESE MODULAIRE DES FONCTIONS NUMERIQUES (ISE-XILINX ; IP-CORE)

Hiérarchisés, les modules et les fonctions du projet de synthèse numérique d'une chaîne de traitement spectrométrique, sont facilement transférables pour migrer d'une version de plateforme de développement vers une autre. Le développement du projet est constitué de deux étapes bien distinctes :

- Configuration du *FPGA* avec l'outil de développement de *XILINX* (soit par le langage *VHDL*, soit par utilisation de modules *SIMULINK*)
- Programmation du code pour la gestion du système (Programme C, ou Programmation graphique Labview)

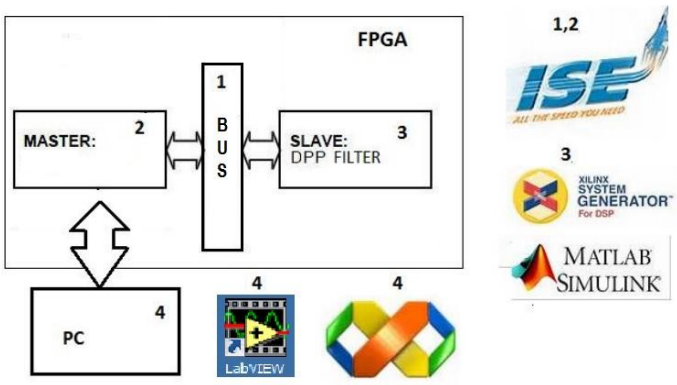
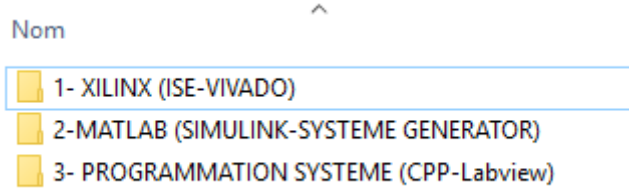


Figure-5. 17 : Séquence des différents outils de développement d'un projet sur FPGA

Pour faciliter l'accès lors de la migration vers différents plateformes de développement, les fichiers crées sont classés dans des répertoires bien définis et distincts.



Dans la majorité des cas, avec quelques adaptations mineures, les modules restent utilisables lors des mises à niveau.

2.1. Synthèse du module PHA sur la carte UNIO52 avec ISE9.2i et System Generator for DSP 9.2.00.967 (SpartanII)

Le projet de développement d'un système numérique d'acquisition spectrométrique, a été démarré en utilisant la carte *UNIO52*, ayant comme circuit

configurable le *FPGA* de type *Spartan II*. La configuration est effectuée utilisant la technique de programmation descriptive *VHDL*, avec l'outil *Xilinx ISE 9.2i*.

Le port entité du module *PHA* numérique est structuré, suivant la description en langage *VHDL*, de même que la configuration du processus d'analyse d'amplitude *PHA*, et dont leurs programmations sont définies dans le fichier *iaea52.vhd* en annexe.

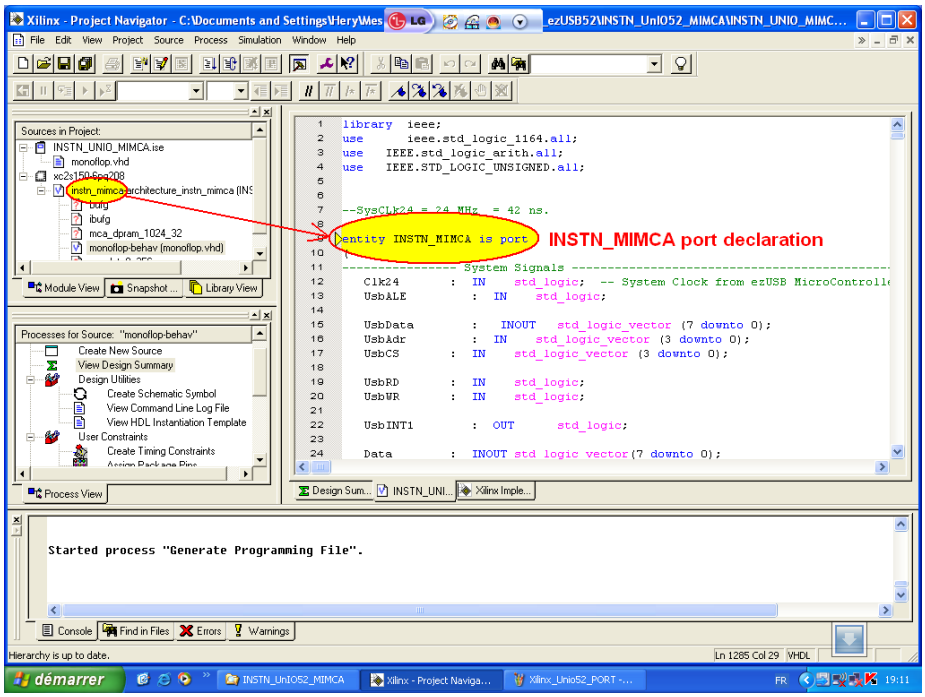


Figure-5. 18 : Déclaration d'une entité sur l'outil de développement ISE-Xilinx

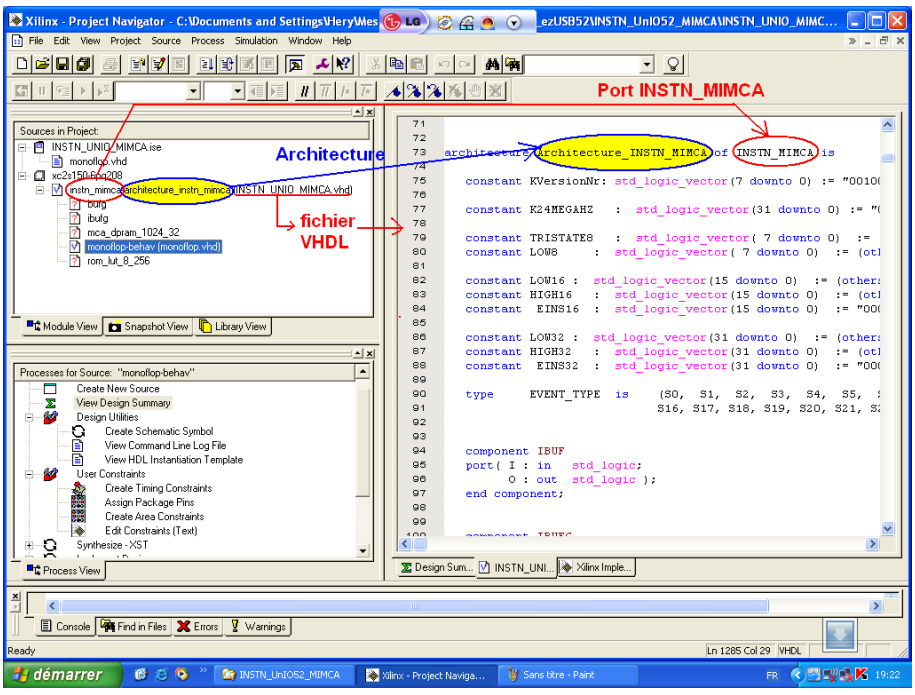


Figure-5. 19 : Déclaration d'une architecture sur l'outil de développement ISE-Xilinx

2.2. Mise à niveau du module PHA pour Spartan 3E de la carte CESYS-Efm01 (System Generator for DSP 13.1)

La technique de configuration utilisant la méthode de synthèse par bloc *SIMULINK* de *MATLAB* et du générateur de système (SYSGEN) propriété de *Xilinx*, a été utilisée pour faire migrer le système d'acquisition vers la plateforme *EFM01* de Cesys.

On a divisé le sous-système d'analyse *PHA* en deux blocs principaux.

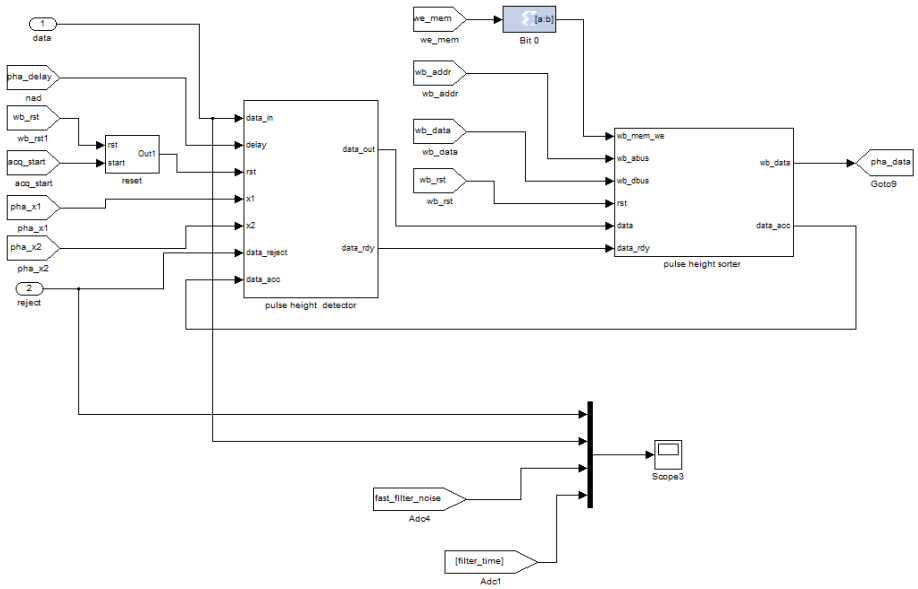


Figure-5. 20 : Schéma du module de traitement numérique de comptage d'amplitude (PHA)

- Le bloc de détection du pic d'amplitude ("*pulse height detector*" sur le schéma) est constitué des blocs "*delay line*" et "*pha_pkd*". Ce dernier est implanté à partir du fichier script *pha_pkd.m* décrit en bas, utilisant le module IP *Mcode* de *Xilinx*.

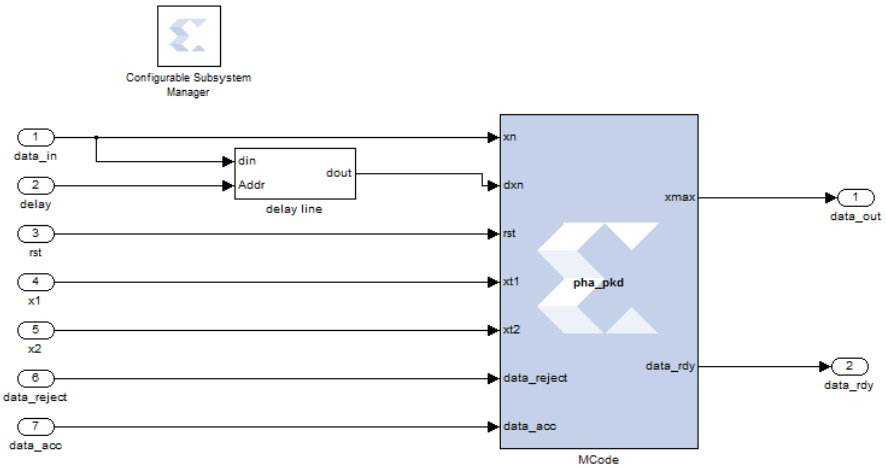


Figure-5. 21 : Construction sur module IP de Xilinx du Bloc de détection d'amplitude (PHA)

Le fonctionnement du module *MCode* est défini à partir du code de commandes scripts, décrit dans le fichier *pha_pkd.m* (en ***Annexe-A2***):

- Tandis que Le bloc "*pulse height sorter"* gère le tri et le classement de chacune des données selon sa valeur (relative à l'amplitude), et est composé de plusieurs modules IP *Xilinx* selon le schéma ci-dessous:

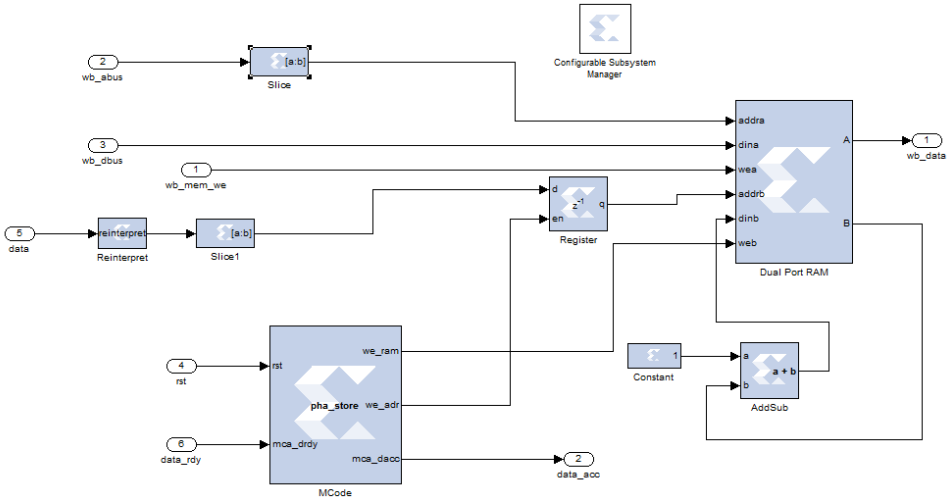


Figure-5. 22 : Modules IP de Xilinx, formant le Bloc de tri et de comptage des impulsions selon leurs amplitude

Le module IP *Xilinx MCode*, dénommé "*pha_store*" est responsable du rangement des différentes valeurs de données dans les adresses mémoires correspondantes (voir le fichier script *pha_store.m* en ***Annexe-A3***).

2.3. Mise à niveau du module de traitement spectrométrique d'amplitude sur un plateforme ZYNQ (ZYNQ et System Generator for DSP 14.1)

Le code programme synthétisé par *SIMULINK* de *MATLAB* et du générateur de système (SYSGEN) de *XILINX*, développé pour le *Spartan-3E* (*EFM01*) est aussi compatible pour être implanté directement sur le *ZYNQ*.

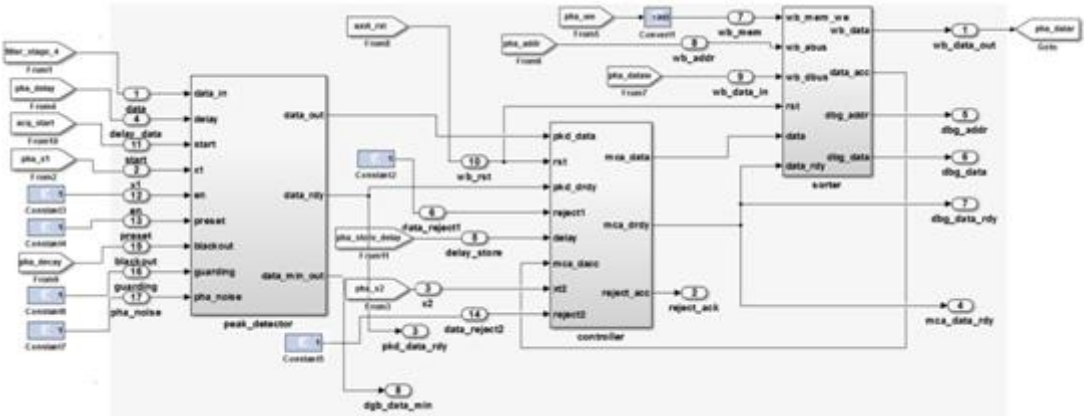


Figure-5. 23 : Schéma bloc des modules IP-Xilinx, configurant le bloc de traitement numérique de comptage d'amplitude (PHA) pour le processeur hybride ZYNQ

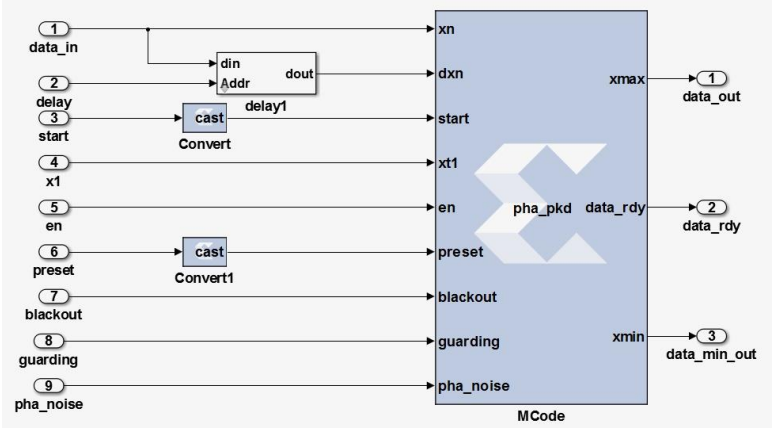


Figure-5. 24 : module IP de Xilinx du Bloc de détection d'amplitude (PHA) pour ZYNQ

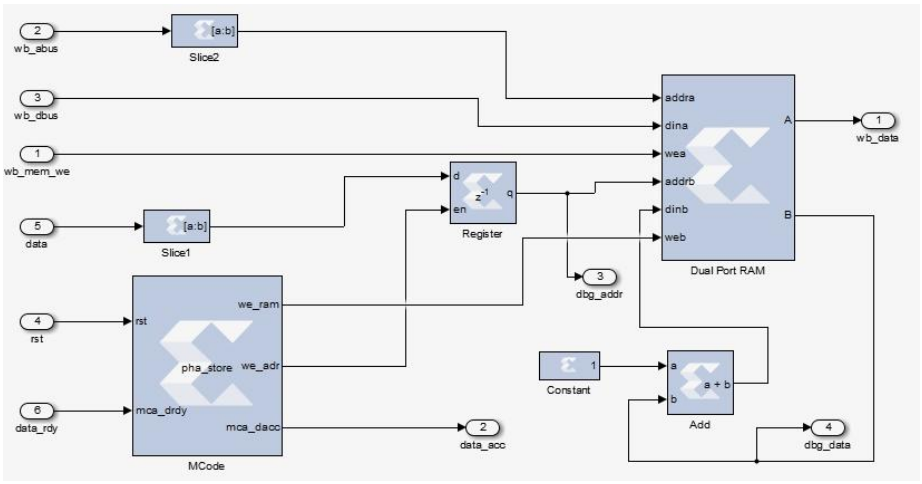


Figure-5. 25 : Modules IP de Xilinx, formant le Bloc de comptage des impulsions selon leurs amplitude

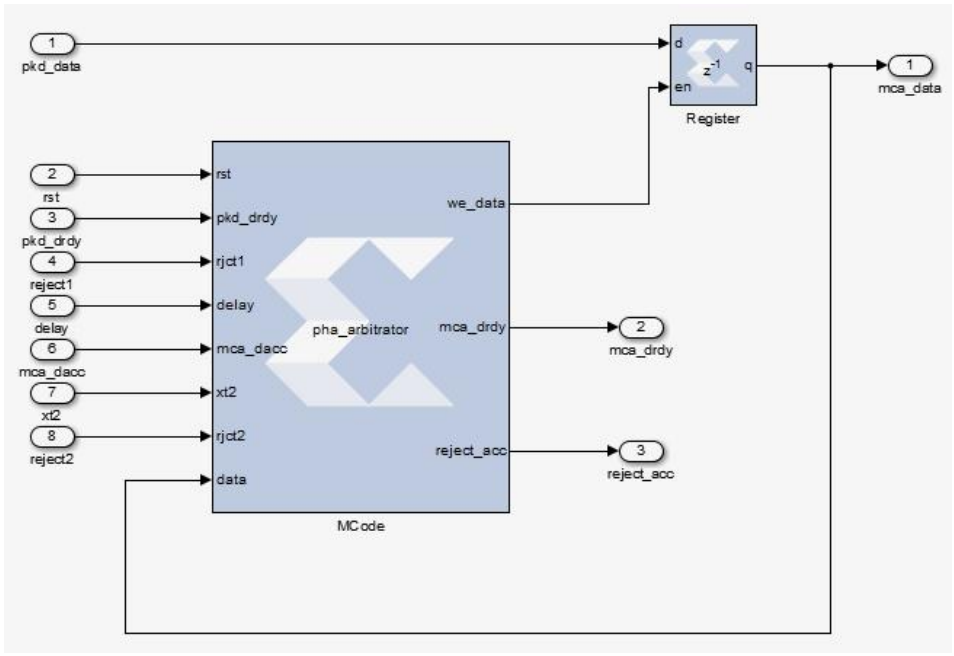


Figure-5. 26 : Modules IP de Xilinx, du Bloc responsable du tri des impulsions selon leurs amplitude

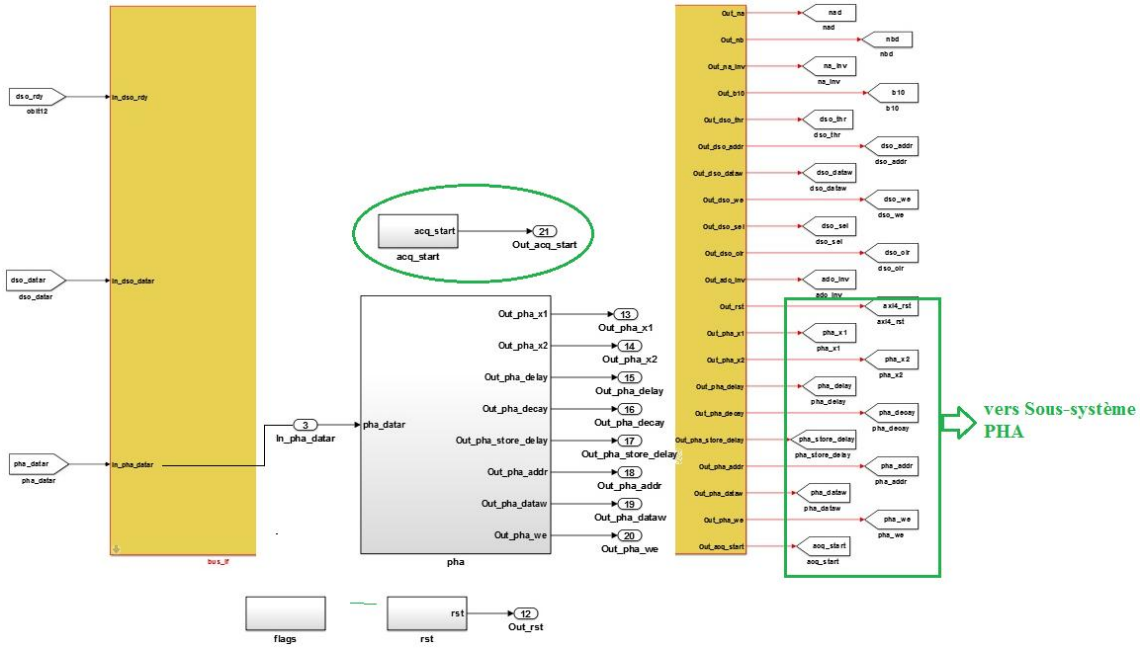


Figure-5. 27 : Schéma des blocs interfaces pour le système de commandes

3. RECONSTRUCTION DU FILTRE TRAPEZOÏDAL ADAPTABLE SUR FPGA

3.1. Filtre TRAPEZOIDAL

Le filtre optimisé a pour fonction de transformer un signal à constante de temps τ_d à décroissance exponentielle, en signal de forme trapézoïdale définie ci-dessous :

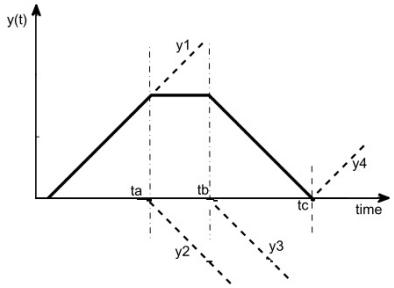


Figure-5. 28 : Construction géométrique d'une courbe de trapézoïde

Où $n_a=ta/T_{clkn}$, $n_b=tb/T_{clkn}$, $n_c=tc/T_{clkn}$, $n_a<n_b<n_c$, T_{clkn} la période d'échantillonnage.

Dans le domaine temporel, le signal trapézoïdal peut être synthétisé à partir de la somme de quatre fonctions élémentaires $y_1(t)$, $y_2(t)$, $y_3(t)$ et $y_4(t)$ telle que :

$$y(t) = y_1(t) + y_2(t) + y_3(t) + y_4(t) \qquad (5.1)$$

Dans le domaine discret on a

$$y(t) = \frac{A_0}{n_a T_{clkn}} nT_{clkn} \cdot h(n) - \frac{A_0}{n_a T_{clkn}} (nT_{clkn} - n_a T_{clkn}) \cdot h(n-n_a) - \frac{A_0}{n_b T_{clkn}} (nT_{clkn} - n_b T_{clkn}) \cdot h(n-n_b) + \frac{A_0}{n_c T_{clkn}} (nT_{clkn} - n_c T_{clkn}) \cdot h(n-n_c) \qquad (5.2)$$

Et passant dans l'espace complexe Z, et utilisant $n_c = n_a + n_b$, on a

$$Y(z) = \frac{A_0}{n_a} \cdot \frac{z}{(z-1)^2}\left[1 - z^{-n_a} - z^{-n_b} + z^{-(n_a+n_b)}\right] \quad (5.3)$$

Et le signal discret correspondant à un signal d'entrée de forme exponentielle aura comme expression

$$X(z) = \frac{z}{z - \exp^{-\frac{T_{clkn}}{\tau}}} \quad (5.4)$$

La fonction de transfert pour un filtre à réponse trapézoïdale serait donnée par :

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{n_a} \cdot \frac{\left(z - \exp^{-\frac{T_{clkn}}{\tau_d}}\right)}{(z-1)^2} \cdot \left[1 - z^{-n_a} - z^{-n_b} + z^{-(n_a+n_b)}\right] \quad (5.5)$$

$$H(z) = \left(1 - z^{-n_a}\right)\left(1 - z^{-n_b}\right)\frac{1}{n_a} \cdot \frac{z - \exp^{-\frac{T_{clkn}}{\tau_d}}}{(z-1)^2} \quad (5.6)$$

3.2. Synthèse numérique du filtre trapézoïdal sur ZYNQ (Matlab/Simulink et System Generator)

Dans la représentation dans l'espace des Z, la fonction de transfert de tout le système de filtrage numérique (trapézoïdal) définie en (5.6) peut se mettre sous l'expression suivante :

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{n_a} \cdot \frac{\left(z - \exp^{-\frac{T_{clkn}}{\tau_d}}\right)}{(z-1)^2} \cdot \left[1 - z^{-n_a} - z^{-n_b} + z^{-(n_a+n_b)}\right] \quad (5.7)$$

$$H(z) = \left(1 - \exp^{-\frac{T_{clkn}}{\tau_d}} \cdot z^{-1}\right)\left(\frac{1 - z^{-n_a}}{1 - z^{-1}}\right)\left(\frac{1 - z^{-n_b}}{1 - z^{-1}}\right)\left(\frac{z^{-1}}{n_a}\right) \quad (5.8)$$

Pour faciliter la synthèse de ce système de filtrage, l'expression ci-dessus peut être scindée en quatre étages montés en casacade, dont les fonctions de transferts sont factorisées les unes à la suite des autres.

$$H = H_1 \cdot H_2 \cdot H_3 \cdot H_4 \quad (5.9)$$

où:

$$\begin{cases} H_1 = 1 - \exp^{-\frac{T_{clkn}}{\tau_d}} \cdot z^{-1} \\ H_2 = \dfrac{1 - z^{-n_a}}{1 - z^{-1}} \\ H_3 = \dfrac{1 - z^{-n_b}}{1 - z^{-1}} \\ H_4 = \dfrac{z^{-1}}{n_a} \end{cases} \quad (5.10)$$

Figure-5. 29 : Cascade de blocs de fonctions de transferts

La relation de récurrence correspondante au premier sous-bloc de filtre H_1 est définie par :

$$H_1(z) = 1 - \exp^{-\frac{T_{clkn}}{\tau_d}} \cdot z^{-1} = \frac{Y_1(z)}{X_1(z)} \qquad (5.11)$$

$$\rightarrow Y_1(z) = X_1(z)\left[1 - \exp^{-\frac{T_{clkn}}{\tau_d}} \cdot z^{-1}\right] \qquad (5.12)$$

$$Y_1(z) = X_1(z) - \exp^{-\frac{T_{clkn}}{\tau_d}} \cdot X_1(z) \cdot z^{-1} \qquad (5.13)$$

$$\rightarrow \boxed{y_{1,n} = x_{1,n} - \exp^{-\frac{T_{clkn}}{\tau_d}} \cdot x_{1,n-1}} \qquad (5.14)$$

De même on déduit les relations de récursivité pour les sous-blocs H2, H3 et H4

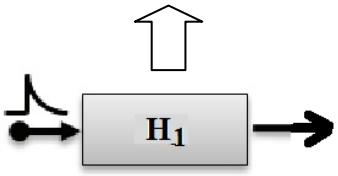
$$\begin{cases} H_2(z) = \dfrac{1 - z^{-n_a}}{1 - z^{-1}} = \dfrac{Y_2(z)}{X_2(z)} & \boxed{y_{2,n} = y_{2,n-1} + x_{2,n} - x_{2,n-n_a}} \quad (5.15) \\ H_3(z) = \dfrac{1 - z^{-n_b}}{1 - z^{-1}} = \dfrac{Y_3(z)}{X_3(z)} & \boxed{y_{3,n} = y_{3,n-1} + x_{3,n} - x_{3,n-n_b}} \quad (5.16) \\ H_4(z) = \dfrac{z^{-1}}{n_a} = \dfrac{Y_4(z)}{X_4(z)} & \boxed{y_{4,n} = \dfrac{1}{n_a} \cdot x_{4,n-1}} \quad (5.17) \end{cases}$$

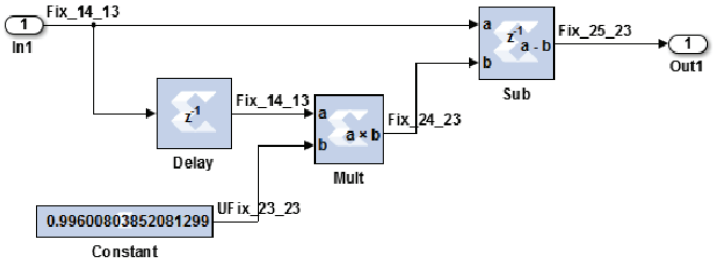
- L'étage, ***Filter1*** constitué par le filtre H1 est synthétisé comme suit :

$H_1 = 1 - \exp^{-\frac{T_{clkn}}{\tau_d}} \cdot z^{-1}$ ⇒ $y_{1,n} = x_{1,n} - \exp^{-\frac{T_{clkn}}{\tau_d}} \cdot x_{1,n-1}$

Fonction de transfert — Relation de récurrence

Synthèse sur SIMULINK



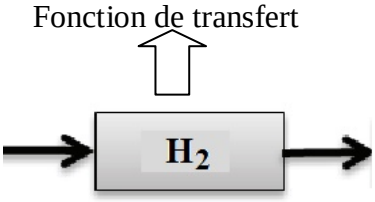


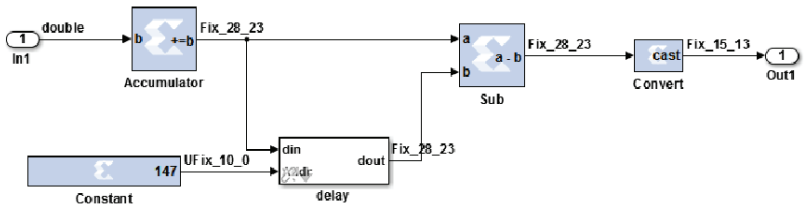
- L'étage, ***Filter2*** constitué par le filtre H2 est synthétisé numériquement comme suit :

$$H_2 = \frac{1 - z^{-n_a}}{1 - z^{-1}}$$ Relation de récurrence $$y_{2,n} = y_{2,n-1} + x_{2,n} - x_{2,n-n_a}$$

Fonction de transfert

Synthèse sur SIMULINK





L'étage Filter2 est constitué d'une variable de retard n_a qui est paramétrable ("*delay*"), d'un accumulateur pour garder la valeur précédente $y_{2,n-1}$ de la sortie y_2.

La valeur de *na* correspond à la durée t_a du front ascendant du pic ("*peaking time*"); exprimée en fonction de l'horloge d'échantillonnage T_{clkn} par la relation :

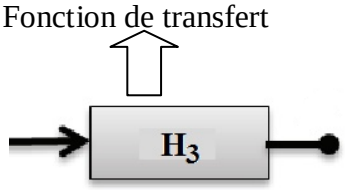
$$n_a = \frac{t_a}{Tclkn} \qquad (5.18)$$

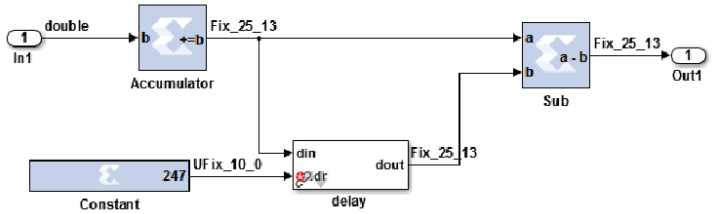
- L'étage, ***Filter3*** constitué par le filtre H3 est d'une structure identique à celle de ***Filter2***, et est synthétisé numériquement comme suit :

$$H_3 = \frac{1 - z^{-n_b}}{1 - z^{-1}}$$ Relation de récurrence $$y_{3,n} = y_{3,n-1} + x_{3,n} - x_{3,n-n}$$

Fonction de transfert

Synthèse sur SIMULINK



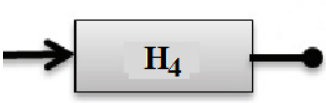


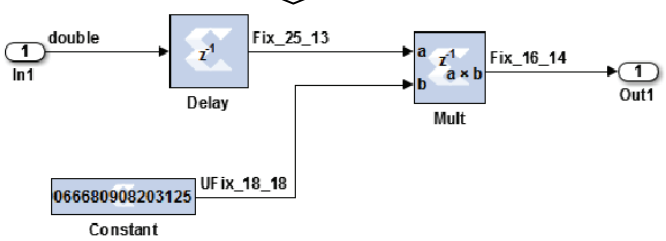
- L'étage, ***Filter4*** constitué par le filtre H4 est un filtre à retard (delay), et est synthétisé numériquement comme suit :

$$H_4 = \frac{z^{-1}}{n_a}$$ Relation de récurrence $$y_{4,n} = \frac{1}{n} \cdot x_{4,n-1}$$

Fonction de transfert

Synthèse sur SIMULINK





L'opération arithmétique de division par la variable n_a, est implantée avec l'opération multiplication en utilisant l'inverse ($1/n_a$) comme multiplicateur. Cette valeur inverse $1/n_a$ est stockée au préalable dans une registre.

Dans tous les calculs arithmétiques, les opérations sont effectuées à virgule binaire fixe.

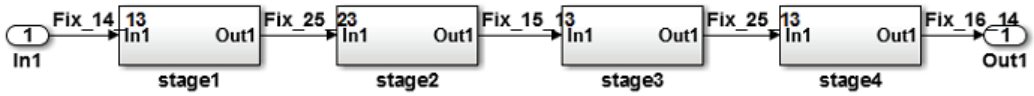


Figure-5. 30 : Cascade de filtres formant le système trapézoïdal

3.3. Implantation du modèle de Filtre sur le silicium (FPGA)

Pour pouvoir interagir dynamiquement avec le système, les paramètres définissant le filtre (n_a, n_d, b_{10} et n_a_inv) sont enregistrés dans un système de registres qui sont accessibles et pouvant être contrôlés par le microordinateur hôte. Ceux-ci sont groupés avec tous les autres registres accessibles, contenus dans un module unique (*bus_if*).

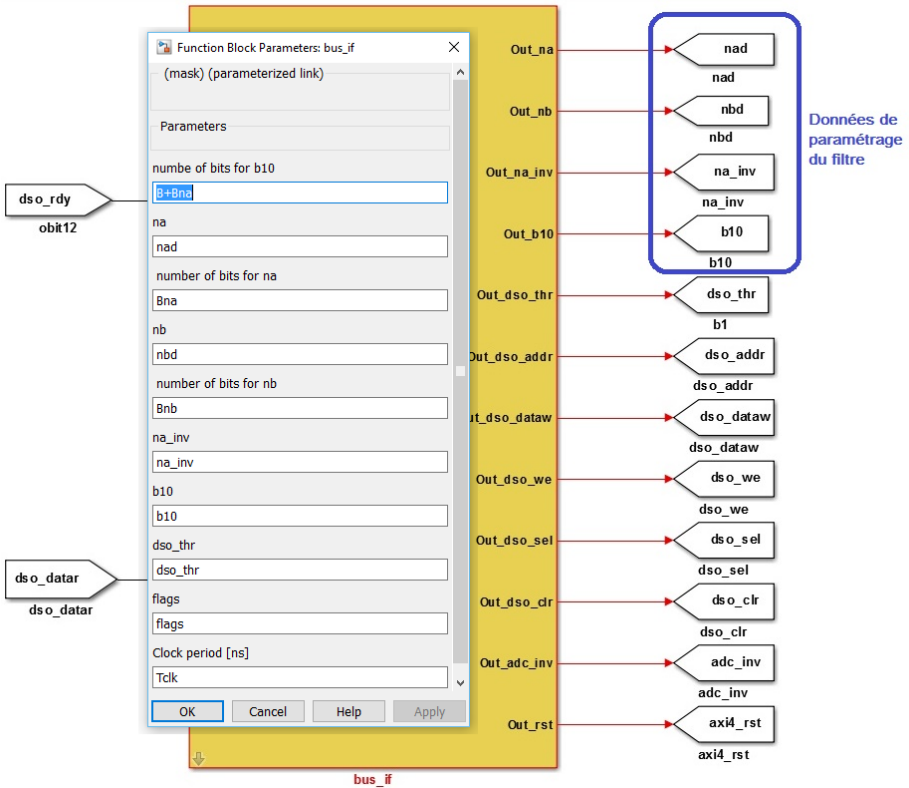


Figure-5. 31 : Paramètrage du filtre à travers le module IP d'interface

Les constantes de paramétrage sont alors transmises vers les étages respectifs qui leurs correspondent, par l'intermédiaire des blocs *FROM* et *GOTO* de *SIMULINK*, suivant les schémas ci-dessous.

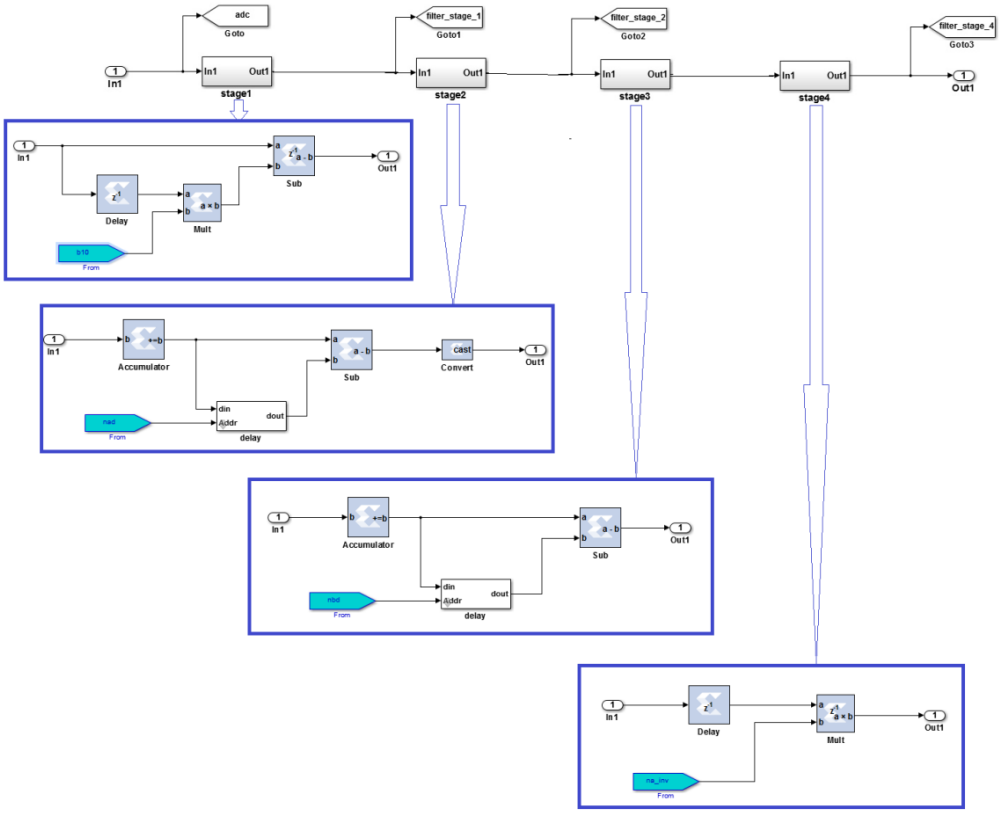


Figure-5. 32 : Schéma de principe des blocs constituant le filtre trapézoïdal

Dans le modèle *SIMULINK* de simulation seuls les modules *Xilinx* qui seront convertis en code *VHDL* et intégrés pour être construits en "*hardware*" par l'outil de développement (*ISE* ou *VIVADO*) de *Xilinx*. Cette transcription est effectuée par le module "System Generator", qui devrait y être inclus dès la création du modèle.

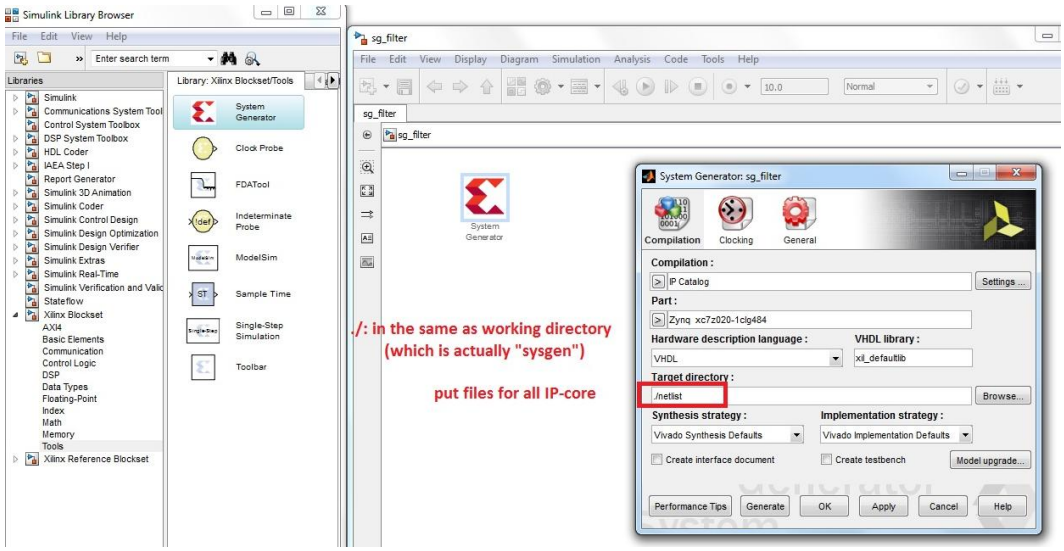


Figure-5. 33 : Bloc IP de Xilinx responsable de la construction des modèles numériques en système matériel ("hardware")

Suivant l'instruction qu'on a transmis pour le module "System Generator", les fichiers de transcription du modèle en *VHDL*, tels que ***sg_filter.vhd*** et ***sg_filter_cw.vhd*** sont créés dans le dossier */netlist* du répertoire de travail contenant modèle *SIMULINK*.

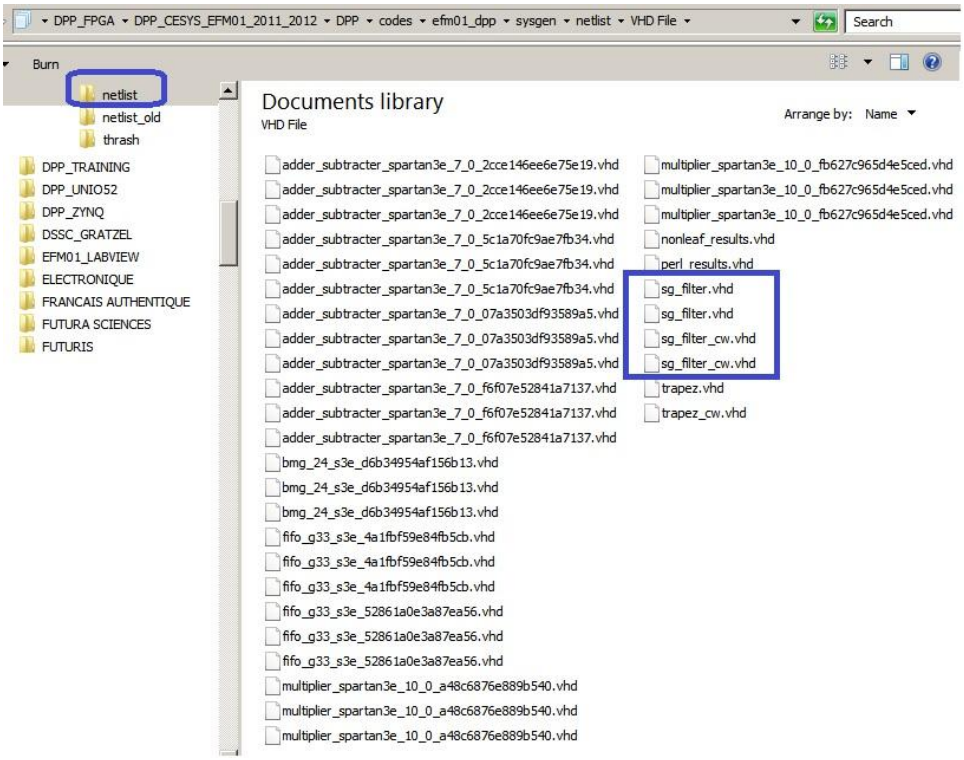


Figure-5. 34 : Fichiers de script VHDL définissant les diverses entités ainsi construits et leurs fonctionnements

ISE design suite – Xilinx

Outre les fichiers du modèle *SIMULINK* ci-dessus, plusieurs autres fichiers sont nécessaires et à ajouter au projet pour la description complète du matériel pour une synthèse avec l'outil de développement *ISE* (ou *VIVADO*) de *Xilinx*.

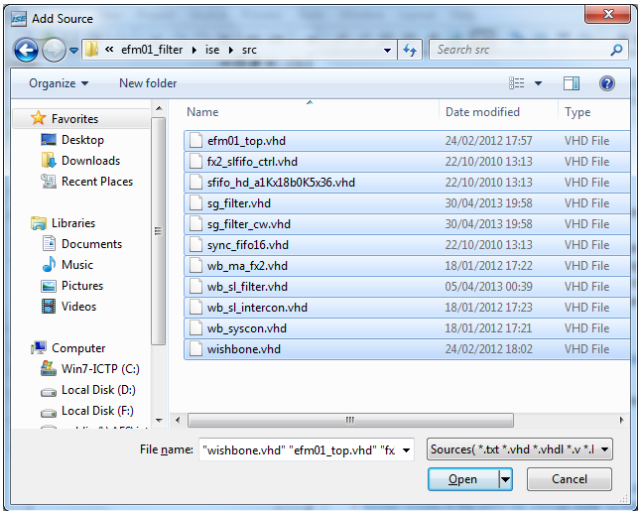


Figure-5. 35 : Ensemble de fichiers constructeurs et personnalisés nécessaires pour la prise en main du système numérique

Le fichier ***efm01.ucf*** est aussi nécessaire et à copier dans le dossier ise, car celui-ci détermine la configuration et les contraintes sur les pins du *FPGA*.

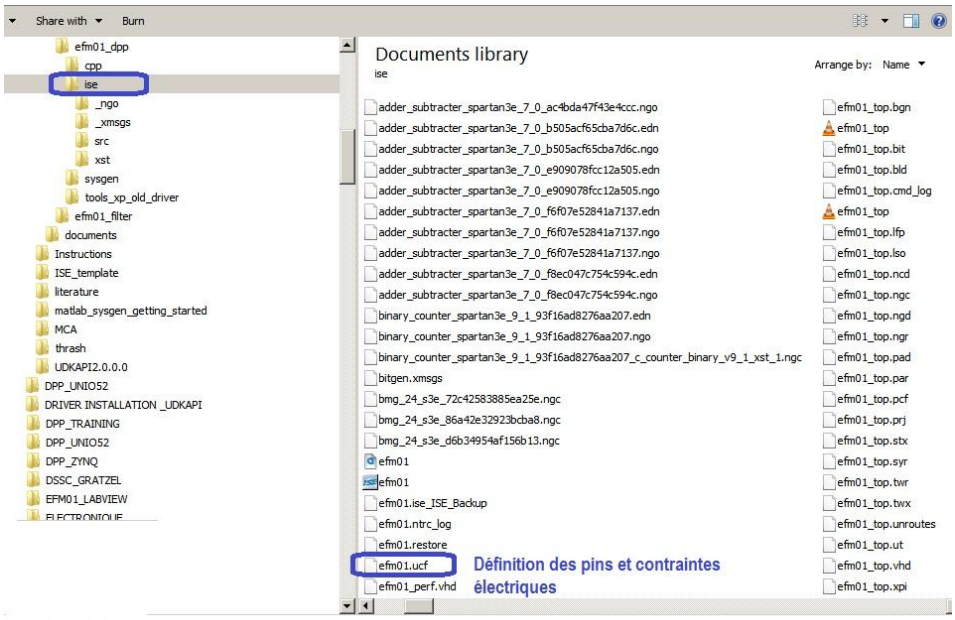


Figure-5. 36 : le fichier efm01.ucf, renfermant la configuration des pins du FPGA

En Effectuant la synthèse des codes *VHDL* (le plus haut niveau de hiérarchie du projet sur Xilinx : ***efm01_top – RTL***) avec la plateforme ISE, des fichiers définissant les composants abstraits réellement utilisés et leurs interconnexions sont créés (fichier avec extension *.ngc*).

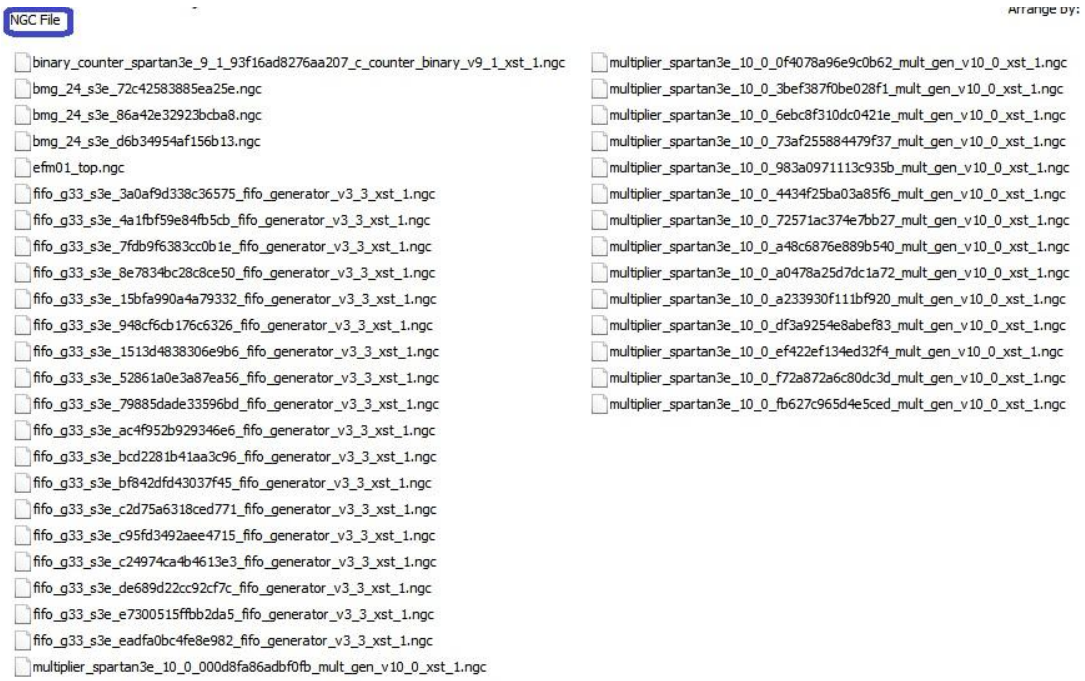


Figure-5. 37 : Fichiers de définitions des modules et de leurs interconnexions

Le processus de programmation en code binaire sur le *FPGA* sera exécuté en cliquant-droite sur "*Generate Programming File*" et lancer "*RUN*".

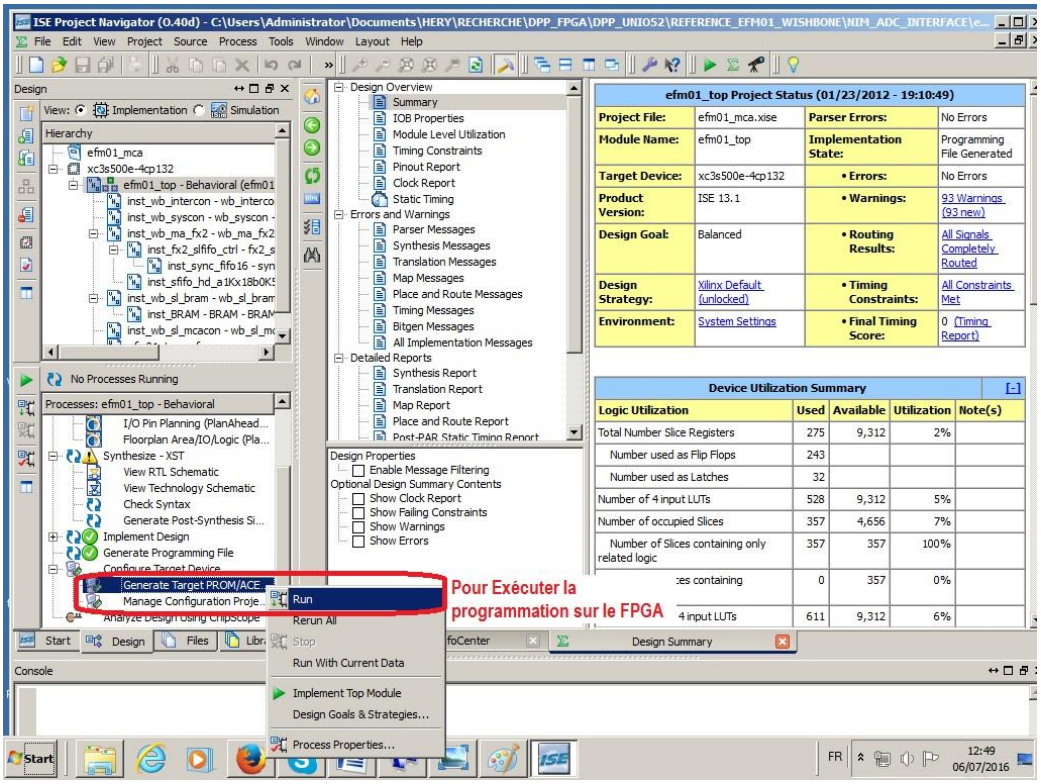


Figure-5. 38 : Fonction de commande déclenchant la construction matérielle sur FPGA

N.B: Il est important de vérifier que l'option pour la création de fichier de configuration binaire est bien active avant de lancer la programmation.