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 0 : 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 0 <= syscon;
 sys rst <= sys_rst_i;</pre>
 sys_clk <= sys_clk_i;</pre>
 debug <=
 (
 others => 'Z'
 );
 -- architecture implementation ----
 syscon.rst <= sys rst;</pre>
 syscon.clk <= sys_clk;</pre>
 end RTL;
```

Le Bus AXI

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



AXI Interconnect Block Connects Multiple Master / Slave Pairs

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*.

🥂 📑 📰 🖙 🖓 🐘 🗎	K 🐼 🕭 🕨 🚵 🚳 💥 ∑ 🧑 😐 Default Layout 🔹	夏 秦 朱 40				w	rite_bitstream Complet		
Flow Navigator «	Block Design dop)		
Q. II #	() '/so_fiter_0' block in this design should be upgraded. Report P Status Upgrade Later								
	Design L X Maddress Editor x Se Dearam x 0. IP Catalog x								
Project Manager	옥 🌋 🗶 등	20 Search: Q,- ^1	l						
Add Sources	External Interfaces	min Name	AXI4	Status	License	VLNV			
Language Templates	E C Ports	Accumulator		Production	Included	xlinx.com:ip			
UF IP Catalog	🕀 🧰 Nets	Adder/Subtracter		Production	Included	xilinx.com:ip			
IP Integrator	Gk_wiz_0 (Clocking Wizard:5.1) Grocessing system7 0 (Z/NO7 Processing System:5.5)	AHB-Lite to AXI Bridge	AXI4	Production	Included	xiinx.com:ip			
Create Block Design	processing_system7_0_axi_periph	AUGO LO ITANSHITTER RECEIVER	AXI4 AXI4-Stream	Production	Included	xlinx.com:in			
Coen Block Design	rst_ck_wiz_0_50M (Processor System Reset:5.0)	AXI4-Stream Accelerator Adapter	AXI4, AXI4-Stream	Beta	Included	xilinx.com:ip			
Su open block besign		AXI4-Stream Broadcaster	AXI4-Stream	Production	Included	xilinx.com:ip			
Generate Block Design		AX14-Stream Clock Converter	AXI4-Stream	Production	Included	xlinx.com:ip			
Simulation		AVIA-Stream Data ETEO	AXI4-Stream	Production	Included	xiinx.com:ip			
Ginulation Settings		AXI4-Stream Data Width Converter	AXI4-Stream	Production	Included	xiinx.com:ip			
Con Con Interne		AXI4-Stream Interconnect	AXI4-Stream	Production	Included	xilinx.com:ip			
w Kun Sinuadon		AXI4-Stream Protocol Checker	AXI4-Stream	Production	Included	xilinx.com:ip			
 RTL Analysis 		AXI4-Stream Register Silce	AX14-Stream	Production	Included	xiinx.com:ip			
Open Elaborated Design		AXI4-Stream Switch	AXI4-Stream	Production	Included	xiinx.com:ip			
	👃 Sources 🚦 Design 📾 Signals 📲 Board	AXI4-Stream to Video Out	AXI4-Stream	Production	Included	xilinx.com:ip			
 Synthesis 		AXI AHBLite Bridge	AXI4	Production	Included	xilinx.com:ip			
🚯 Synthesis Settings		G AXI APB Bridge	AXI4	Production	Included	xiinx.com:ip			
Run Synthesis	← → ¹⁰ k	AXI BEM Controller	AX14, AX14-Stream	Production	Purchase	xenx.com:ip			
Coan Suntherized Darin		O AXI CAN	AXI4	Production	Purchase	xilinx.com:ip			
P B Open Synerestee besy		AXI Central Direct Memory Access	AXI4	Production	Included	xiinx.com:ip			
 Implementation 		AXI Chip2Chip Bridge	AXI4, AXI4-Stream	Production	Included	xilinx.com:ip			
G Implementation Settings		U AXI Clock Converter	AXI4	Production	Included	xlinx.com:ip			
Pup Implementation		AXI Crossbar	AX14	Production	Included	viny comin			
		G AXI DataMover	AXI4, AXI4-Stream	Production	Included	xilinx.com:ip			
Open Implemented Desi		AXI Data Width Converter	AXI4	Production	Included	xilinx.com:ip			
Program and Debug		AXI Direct Memory Access	AXI4, AXI4-Stream	Production	Included	xilinx.com:ip			
Bitetream Settinge		ANTERC	AXI4	Production	Included	xlinx.com:ip			
	Tcl Console	AXI Ethernett ite	AX14	Production	Included	xiinx.com:ip	- 0 C ×		
Generate bitstream	copen project: Time (s): cpu = 00:01:19 ; elapse	1 = 010 AXI Ethernet Subsystem	AXI4, AXI4-Stream	Production	Included	xilinx.com:ip	^		
Open Hardware Manage	open_bd_design (D:/MY_DATA/ACTIVITES_SCIENTIFIQ	JES/T C AXI GPIO	AXI4	Production	Included	xilinx.com:ip			
	Adding component instance block xilinx.com:ij	pipro AXI HWICAP	AXI4	Production	Included	xilinx.com:ip			
	Adding component instance block xilinx.com:ip	p:clk ANI IIC	AXI4	Production	Included	xlinx.com:ip			
	Adding component instance block xilinx.com:ij	AXI Intercurrent Controller	AXI4	Production	Included	xlinx.com:ip			
	Adding component instance block IAEA:sg_filty	AXI Memory Mapped to Stream Mapper	AXI4, AXI4-Stream	Production	Included	xilinx.com:ip			
	× Successfully read diagram (don) from BD file (D	AXI MMU	AXI4	Production	Included	xilinx.com:ip	urces 1/bd/dpp/c		
	open bd design: Time (s); cpu = 00:00:28 ; elap	= AXI Performance Monitor	AXI4, AXI4-Stream	Production	Included	xilinx.com:ip	arous_r, so, app, c		
		AXI Protocol Checker	AX14	Production	Included	xiinx.com:ip			
	<	G AXI Quad SPI	AXI4	Production	Included	xiinx.com:ip	, ,		
	Type a Tol command here	AXI Register Slice	AXI4	Production	Included	xilinx.com:ip			
	Tel Canada O Managera (El Las D Danata D Danas D	AXI TFT Controller	AXI4	Production	Included	xiinx.com:ip			
	I Console La Messages Log Log Log Reports La Design Ru	AXI Timebase Watchdog Timer	AXI4	Production	Included	xilinx.com:ip			
		ANI filmer	AXI4 AXI4 AXI4.Sho	Production	Included	xilinx.com:ip			
-		AXI LIART 16550	AXI4	Production	Included	xiinx.comip			
Je suis Cortana	. Posez-moi une question. 📋 📻 🔁 💷	AXI Uartite	AXI4	Production	Included	xilinx.com:ip	FRA 07/07/2016		
		AVI USB2 Device	AXI4	Production	Purchase	xilinx.com:ip			
		AXI Video Direct Memory Access	AXI4, AXI4-Stream	Production	Included	xilinx.com:ip			
		E F AXI Virtual FIFO Controller	AXI4-Stream, AXI4	deta	Tockuded	xlinx.com:ip			

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

Stilinx - Project Navigator - C:\Documents and Settings\Hery\Mes	🚯 LG) 🙋 🔬 💿	ezUSB52\INSTN_UnIO52_MIMCA\INSTN_UNIO_MIMC			
File Edit View Project Source Process Simulation Window Help		» _ 8 ×			
	🖻 🖻 🗠 🖂 桶 🆗	- S			
		Port INSTN_MIMCA			
Courses in Project	71				
	72				
monofige.vhd Architecture	73 architeetare	Architecture_INSTN_MIMCA of (INSTN_MIMCA is			
E- □ xc2s150.664208	75 constant	KVersionNr: std logic vector(7 downto 0) := "0010(
<u>V</u> (instn_mimcatarchitecture_instn_mimcat(INSTN_UNIO_MIMCA.vhd)	76				
inter fichier	77 constant	K24MEGAHZ : std_logic_vector(31 downto 0) := "(
mca dpram 1024 32		TRICTATED			
	80 constant	LOW8 : std logic vector(7 downto 0) := (ot)			
2 rom_lut_8_256	81				
	82 constant	LOW16 : std_logic_vector(15 downto 0) := (other:			
	83 constant	HIGH16 : std_logic_vector(15 downto 0) := (ot)			
	84 constant	EINSIS : Std_logic_vector(IS downto U) := "UU			
Module View Snapshot View U Library View	86 constant	LOW32 : std logic vector(31 downto 0) := (other:			
× ×	87 constant	HIGH32 : std_logic_vector(31 downto 0) := (ot)			
Processes for Source: "monoflop-behav"	88 constant	EINS32 : std_logic_vector(31 downto 0) := "00(
Create New Source	89	FUENT TYDE 10 (20 21 22 22 24 25 4			
View Design Summary	91	S16, S17, S18, S19, S20, S21, S2			
E-12 Design Utilities	92	,,,			
View Command Line Log File	93				
View HDL Instantiation Template	94 component	IBUF			
🗉 💅 User Constraints	as port(1:	in std_logic;			
Create Timing Constraints	97 end compo	nent;			
Assign Package Pins	98				
Edit Constraints	99	✓			
Synthesize - XST	100 Component	TOTOC >			
	Thering Sum	UNI Da Viliov Innia			
- Process view		Contra California andressa			
×					
	110				
I Lonsole M End in Files X Litrors X Warnings					
Ready Ln 1285 Col 29 VHDL					
🛃 démarrer 🛛 🧐 🍯 💁 " 🐚 INSTN_UnIOS2_MIMCA 🛛 👔	Xilinx - Project Naviga 🛛 🦉 S	ans titre - Paint FR 🔦 🗒 🕵 K 19:22			

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 :

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

Dans le domaine discret on a

$$y(t) = \frac{A_0}{n_a T_{clkn}} n T_{clkn} \cdot h(n) - \frac{A_0}{n_a T_{clkn}} (n T_{clkn} - n_a T_{clkn}) \cdot h(n - n_a)$$

$$- \frac{A_0}{n_b T_{clkn}} (n T_{clkn} - n_b T_{clkn}) \cdot h(n - n_b) + \frac{A_0}{n_c T_{clkn}} (n T_{clkn} - n_c T_{clkn}) \cdot h(n - n_c)$$
(5.2)
120

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

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

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

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

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

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

$$H(z) = \left(1 - z^{-n_a}\right) \left(1 - z^{-n_b}\right) \frac{1}{n_a} \cdot \frac{z - \exp^{-\frac{I \cdot l \cdot n}{\tau_d}}}{(z - 1)^2}$$
(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 :

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

$$H = H_1 \cdot H_2 \cdot H_3 \cdot H_4 \tag{5.9}$$

où:

$$\begin{cases} H_{1} = 1 - \exp^{-\frac{T_{ckn}}{\tau_{d}}} \cdot z^{-1} \qquad (5.10) \\ H_{2} = \frac{1 - z^{-n_{a}}}{1 - z^{-1}} \\ H_{3} = \frac{1 - z^{-n_{b}}}{1 - z^{-1}} \\ H_{4} = \frac{z^{-1}}{n_{a}} \end{cases}$$

121



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 :

$$H_{1}(z) = 1 - \exp^{-\frac{T_{clkn}}{\tau_{d}}} \cdot z^{-1} = \frac{Y_{1}(z)}{X_{1}(z)}$$
(5.11)

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

$$Y_{1}(z) = X_{1}(z) - \exp^{-\frac{T_{clkn}}{\tau_{d}}} \cdot X_{1}(z) \cdot z^{-1}$$
(5.13)

$$y_{1,n} = x_{1,n} - \exp^{-\frac{-\epsilon n\pi}{\tau_d}} \cdot x_{1,n-1}$$
 (5.14)

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

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





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 :

$$n_a = \frac{t_a}{Tclkn} \tag{5.18}$$

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



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



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*.

Add Source						
Search src + 4						
Organize 🔻 New folder 🛛 👔 👻 🗍 🔞						
☆ Favorites	-	Na	me	Date modified	Туре	
🧮 Desktop			efm01_top.vhd	24/02/2012 17:57	VHD File	
🚺 Downloads		C	fx2_slfifo_ctrl.vhd	22/10/2010 13:13	VHD File	
🔛 Recent Places		C	sfifo_hd_a1Kx18b0K5x36.vhd	22/10/2010 13:13	VHD File	
		C	sg_filter.vhd	30/04/2013 19:58	VHD File	
🥽 Libraries	в		sg_filter_cw.vhd	30/04/2013 19:58	VHD File	
Documents		C	sync_fifo16.vhd	22/10/2010 13:13	VHD File	
🁌 Music		C	wb_ma_fx2.vhd	18/01/2012 17:22	VHD File	
Pictures			C	wb_sl_filter.vhd	05/04/2013 00:39	VHD File
😸 Videos				wb_sl_intercon.vhd	18/01/2012 17:23	VHD File
			wb_syscon.vhd	18/01/2012 17:21	VHD File	
👰 Computer			wishbone.vhd	24/02/2012 18:02	VHD File	
🏭 Win7-ICTP (C:)						
👝 Local Disk (D:)						
👝 Local Disk (F:)		(٢	
				[
File <u>n</u> ame:			"wishbone.vhd" "efm01_top.vhd" "fx 👻	Sources(*.txt *.vhd *.vh	ndl *.v *.l ▼	
				<u>O</u> pen ▼	Cancel	

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

NCC City

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

a fee 0.1 days		
cpp	Documents library	Arrange by: Name 🔻
🎍 _ngo	adder_subtracter_spartan3e_7_0_ac4bda47f43e4ccc.ngo	efm01_top.bgn
🗼 _xmsgs	adder_subtracter_spartan3e_7_0_b505acf65cba7d6c.edn	A efm01_top
src	adder_subtracter_spartan3e_7_0_b505acf65cba7d6c.ngo	efm01_top.bit
Jan xst	adder_subtracter_spartan3e_7_0_e909078fcc12a505.edn	efm01_top.bld
tools vp old driver	adder_subtracter_spartan3e_7_0_e909078fcc12a505.ngo	efm01_top.cmd_lo
efm01 filter	adder_subtracter_spartan3e_7_0_f6f07e52841a7137.edn	A efm01_top
documents	adder_subtracter_spartan3e_7_0_f6f07e52841a7137.ngo	efm01_top.lfp
Instructions	adder_subtracter_spartan3e_7_0_f6f07e52841a7137.ngo	efm01_top.lso
📙 ISE_template	adder_subtracter_spartan3e_7_0_f8ec047c754c594c.edn	efm01_top.ncd
🎉 literature	adder_subtracter_spartan3e_7_0_f8ec047c754c594c.ngo	efm01_top.ngc
matlab_sysgen_getting_started	binary_counter_spartan3e_9_1_93f16ad8276aa207.edn	efm01_top.ngd
MCA	binary_counter_spartan3e_9_1_93f16ad8276aa207.ngo	efm01_top.ngr
b thrash	binary_counter_spartan3e_9_1_93f16ad8276aa207_c_counter_binary_v9_1_xst_1.ngc	efm01_top.pad
DDR UNIO52	bitgen.xmsgs	efm01_top.par
DRIVER INSTALLATION UDKAPI	bmg_24_s3e_72c42583885ea25e.ngc	efm01_top.pcf
DPP TRAINING	bmg_24_s3e_86a42e32923bcba8.ngc	efm01_top.prj
DPP_UNIO52	bmg_24_s3e_d6b34954af156b13.ngc	efm01_top.stx
DPP_ZYNQ	@efm01	efm01_top.syr
DSSC_GRATZEL	rssfefm01	efm01_top.twr
EFM01_LABVIEW	efm01.ise_ISE_Backup	efm01_top.twx
ELECTRONIQUE	efm01.ntrc_log	efm01_top.unrou
	efm01.restore	efm01_top.ut
	Définition des pins et contraintes	efm01_top.vhd
	efm01_perf.vhd électriques	efm01_top.xpi

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*).

NGC FIE	
binary_counter_spartan3e_9_1_93f16ad8276aa207_c_counter_binary_v9_1_xst_1.ng	multiplier_spartan3e_10_0_0f4078a96e9c0b62_mult_gen_v10_0_xst_1.ngc
bmg_24_s3e_72c42583885ea25e.ngc	multiplier_spartan3e_10_0_3bef387f0be028f1_mult_gen_v10_0_xst_1.ngc
bmg_24_s3e_86a42e32923bcba8.ngc	multiplier_spartan3e_10_0_6ebc8f310dc0421e_mult_gen_v10_0_xst_1.ngc
bmg_24_s3e_d6b34954af156b13.ngc	multiplier_spartan3e_10_0_73af255884479f37_mult_gen_v10_0_xst_1.ngc
efm01_top.ngc	multiplier_spartan3e_10_0_983a0971113c935b_mult_gen_v10_0_xst_1.ngc
fifo_g33_s3e_3a0af9d338c36575_fifo_generator_v3_3_xst_1.ngc	multiplier_spartan3e_10_0_4434f25ba03a85f6_mult_gen_v10_0_xst_1.ngc
fifo_g33_s3e_4a1fbf59e84fb5cb_fifo_generator_v3_3_xst_1.ngc	multiplier_spartan3e_10_0_72571ac374e7bb27_mult_gen_v10_0_xst_1.ngc
fifo_g33_s3e_7fdb9f6383cc0b1e_fifo_generator_v3_3_xst_1.ngc	multiplier_spartan3e_10_0_a48c6876e889b540_mult_gen_v10_0_xst_1.ngc
fifo_g33_s3e_8e7834bc28c8ce50_fifo_generator_v3_3_xst_1.ngc	multiplier_spartan3e_10_0_a0478a25d7dc1a72_mult_gen_v10_0_xst_1.ngc
fifo_g33_s3e_15bfa990a4a79332_fifo_generator_v3_3_xst_1.ngc	multiplier_spartan3e_10_0_a233930f111bf920_mult_gen_v10_0_xst_1.ngc
fifo_g33_s3e_948cf6cb176c6326_fifo_generator_v3_3_xst_1.ngc	multiplier_spartan3e_10_0_df3a9254e8abef83_mult_gen_v10_0_xst_1.ngc
fifo_g33_s3e_1513d4838306e9b6_fifo_generator_v3_3_xst_1.ngc	multiplier_spartan3e_10_0_ef422ef134ed32f4_mult_gen_v10_0_xst_1.ngc
fifo_g33_s3e_52861a0e3a87ea56_fifo_generator_v3_3_xst_1.ngc	multiplier_spartan3e_10_0_f72a872a6c80dc3d_mult_gen_v10_0_xst_1.ngc
fifo_g33_s3e_79885dade33596bd_fifo_generator_v3_3_xst_1.ngc	multiplier_spartan3e_10_0_fb627c965d4e5ced_mult_gen_v10_0_xst_1.ngc
fifo_g33_s3e_ac4f952b929346e6_fifo_generator_v3_3_xst_1.ngc	
fifo_g33_s3e_bcd2281b41aa3c96_fifo_generator_v3_3_xst_1.ngc	
fifo_g33_s3e_bf842dfd43037f45_fifo_generator_v3_3_xst_1.ngc	
fifo_g33_s3e_c2d75a6318ced771_fifo_generator_v3_3_xst_1.ngc	
fifo_g33_s3e_c95fd3492aee4715_fifo_generator_v3_3_xst_1.ngc	
fifo_g33_s3e_c24974ca4b4613e3_fifo_generator_v3_3_xst_1.ngc	
fifo_g33_s3e_de689d22cc92cf7c_fifo_generator_v3_3_xst_1.ngc	
fifo_g33_s3e_e7300515ffbb2da5_fifo_generator_v3_3_xst_1.ngc	
fifo_g33_s3e_eadfa0bc4fe8e982_fifo_generator_v3_3_xst_1.ngc	
multiplier_spartan3e_10_0_000d8fa86adbf0fb_mult_gen_v10_0_xst_1.ngc	

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

Arrange by:

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.