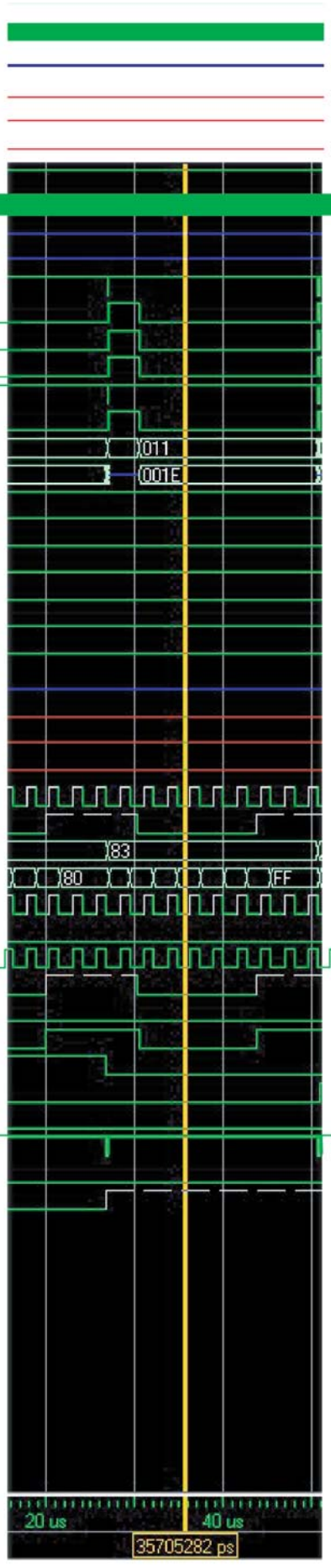
haute école
neuchâtel berne jura

arc ingénierie

www.he-arc.ch

Manuel d'utilisation du Kit Xilinx 7



Yves Meyer

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale

Table des matières

1. INTRODUCTION 3

1.1 DESCRIPTION DU KIT XILINX 7 3
1.2 AFFICHAGES 7 SEGMENTS 4
1.3 BARGRAPHE LEDS 4
1.4 BUZZER 4
1.5 DILSWITCHS 8BIT 4
1.6 BOUTONS POUSSOIRS 5
1.6.1 Poussoir RESET (actif bas) 5
1.7 SIGNAUX D'HORLOGE 5
1.7.1 100MHz 5
1.7.2 CLK_DIV 5
1.8 LEDS BICOLORES 6
1.9 INTERFACE SERIAL TO USB (CP2102-GM) 6
1.10 INTERFACE SERIAL TO USB 2 (FT232BQ) 6
1.11 CONVERTISSEUR A/D PARALLELE (MAX113CAG) 6
1.12 CONVERTISSEUR A/D SPI (ADCS7476AIMFE) 6
1.13 CONVERTISSEUR D/A SPI (AD5323ARUZ) 6
1.14 CONVERTISSEUR D/A AUDIO (CS4344-CZZ) 7
1.15 REAL TIME CLOCK I2C (PCF8563T) 7
1.16 EEPROM I2C (24LC256T-I/SN) 7
1.17 AFFICHAGE LCD PARALLELE (ECC1602B-NSWBBW-91LE) 7
1.18 DRIVER DE LED (IS31LT3360-SDLS3) 7
1.19 CONNECTEUR SERVO (J23) 7
1.20 CONNECTEUR I/O (J13) 7
1.21 CONNECTEURS PMOD 8
1.21.1 Connecteur PMOD 1 (J10) 8
1.21.2 Connecteur PMOD 2 (J11) 8
1.21.3 Connecteur PMOD 3 (J14) 8
1.21.4 Connecteur PMOD 4 (J18) 9
1.21.5 Connecteur PMOD 5 (J19) 9

2. GENERALITES SUR LA CREATION D'UN PROJET 10

2.1 EDITION D'UN PROJET 10
2.1.1 Design Flow 10
2.2 SIMULATION FONCTIONNELLE 10
2.3 SYNTHESE LOGIQUE 10
2.4 FITTER OU PLACEMENT ROUTAGE 10
2.5 PROGRAMMATION 11

3. MODE D'EMPLOI DU KIT XILINX 7 ET UTILISATION DE VIVADO 11

3.1 MISE SOUS TENSION DU KIT 11
3.2 EXEMPLE PRATIQUE DE PROGRAMMATION AVEC XILINX VIVADO 11
3.2.1 Informations générales sur Xilinx Vivado 11
3.2.2 Installation Vivado 11
3.2.3 Création d'un nouveau projet 12
3.2.4 Edition du fichier VHDL 17
3.2.5 Synthèse logique 19
3.2.6 Assignation des signaux sur les pattes de la FPGA (création du fichier de contraintes) 21
3.2.7 Implementer le projet 26
3.2.8 Générer le fichier de configuration 27
3.2.9 Programmer la FPGA 27
3.2.10 Programmer la FPGA avec la mémoire Flash SPI Micron (N25Q256A13ESF40G) 30
3.3 SIMULATION 33

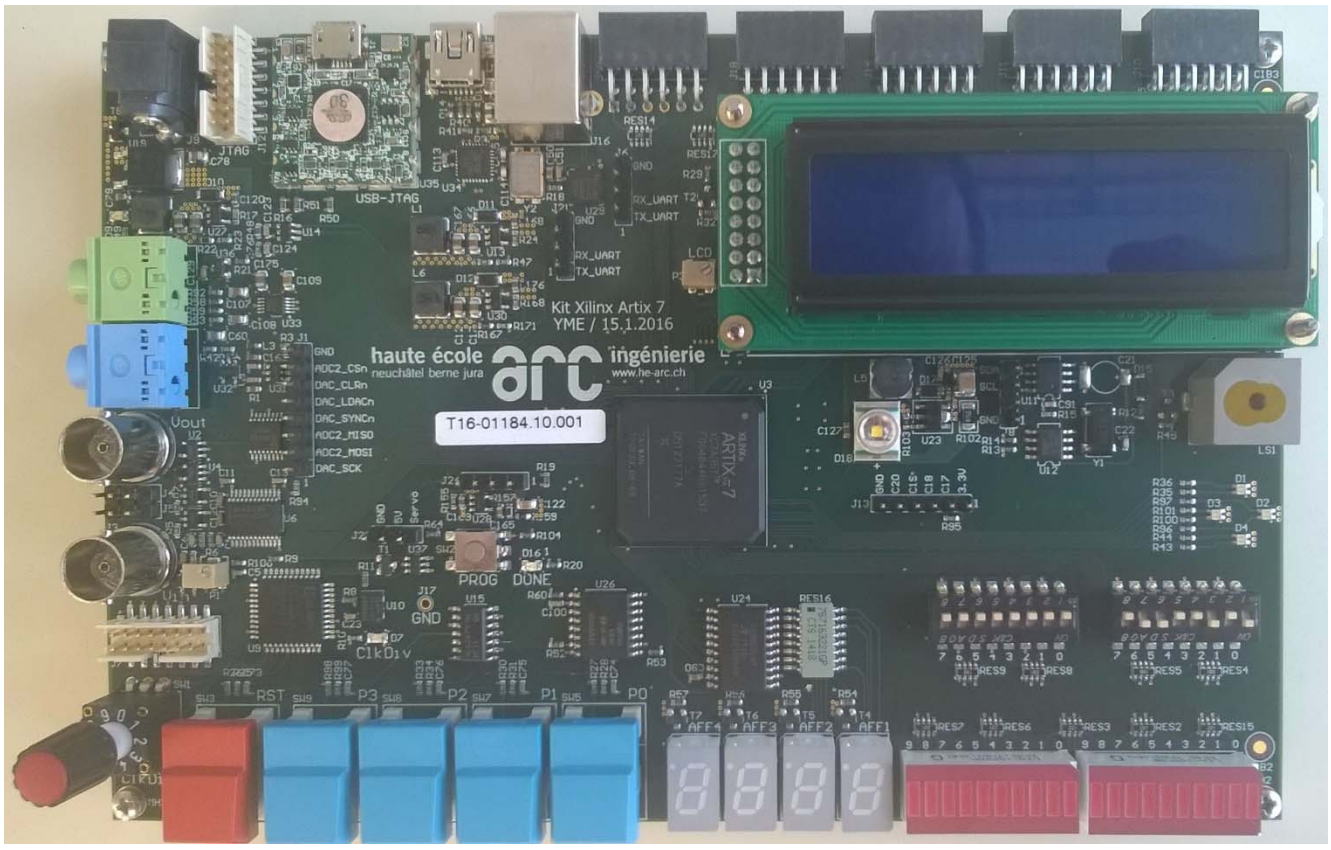
4. ASSIGNATION DES ENTREES/SORTIES DE LA FPGA DU KIT XILINX 7 36

1. Introduction

L'évolution importante dans le domaine de la L'électronique numérique aux cours de ces dernières années à permit de mettre au point toute une nouvelle gamme de circuits logiques programmables (CPLD et FPGA) selon les besoins de l'utilisateur.
Le but de ce kit d'apprentissage est donc de permettre aux étudiants de se familiariser avec la logique programmable, en particulier les CPLD et les FPGA ainsi que le langage de description matériel couramment utilisé dans ce domaine, à savoir le langage de description matériel VHDL.

1.1 Description du Kit Xilinx 7

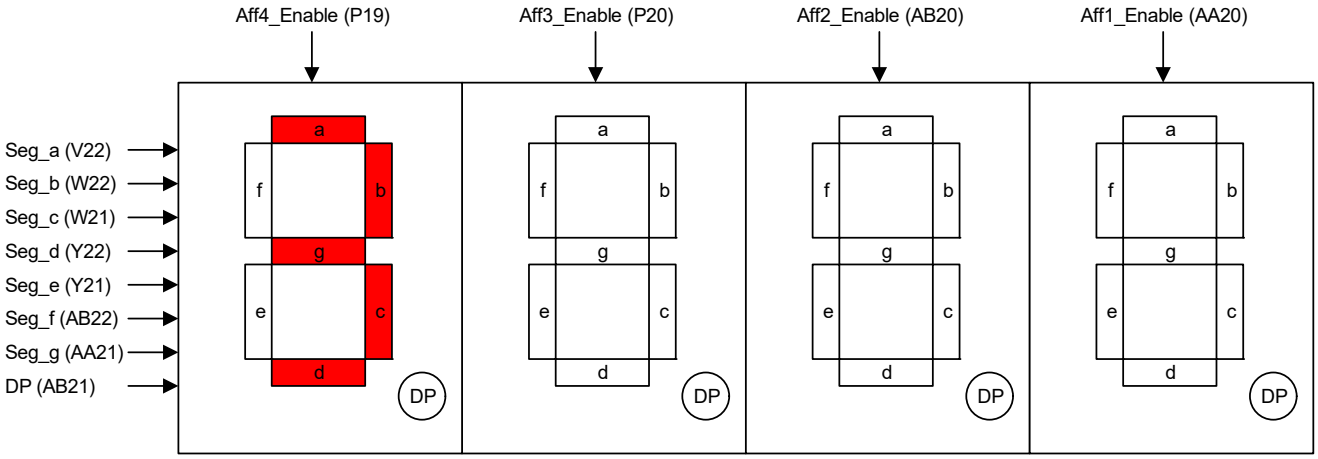
La platine d'expérimentation est composée d'une FPGA Xilinx de la famille Artix-7 de Xilinx (XC7A35T-1FGG484C) dans laquelle seront Implémentées les applications développées durant les cours et laboratoires de systèmes numériques, allant du simple système combinatoire jusqu'àu système on Chip très complexe.



La FPGA dans laquelle seront ciblées les différentes applications est connectée à plusieurs périphériques d'entrées/sorties externes (leds, poussoirs, clocks, affichages 7 segments, dilswitchs, …), permettant de tester le fonctionnement de l'application une fois celle-ci implémentée. Ces différents périphériques sont décrits dans les chapitres suivants.

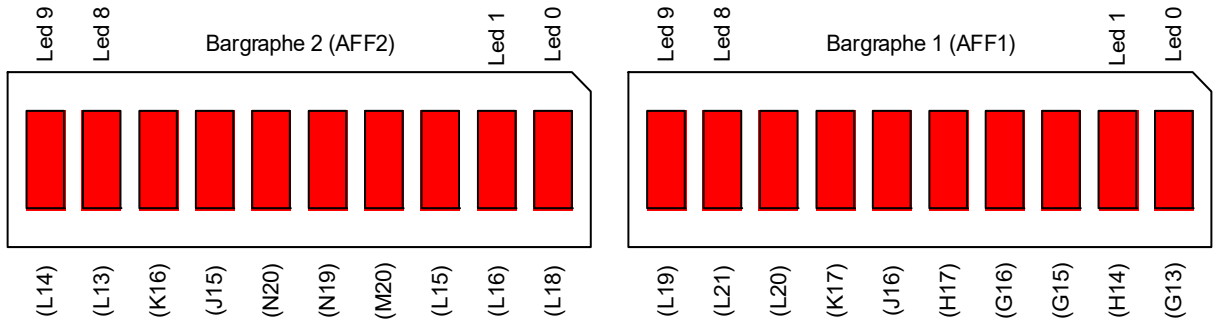
1.2 Affichages 7 segments

Les quatre affichages 7 segments sont reliés à la FPGA via un driver A2982ELW. Chaque segment s'allume lorsqu'on met le signal correspondant à l'état logique haut pour autant que l'affichage soit activé par sa commande Aff_Enable. La figure ci-dessous représente les quatres affichages tels qu'ils sont positionés sur le kit, ainsi que la dénomination des segments et les pattes de la FPGA associés à ces segement.



1.3 Bargraphe leds

Les LEDs du bargraphe sont reliées directement à la FPGA. Un état logique haut sur chacun de ces signaux allumera la LED correspondante. La figure ci-dessous représente le bargraphe avec la numérotation des leds, ainsi que les pattes de la FPGA associées à ces leds.

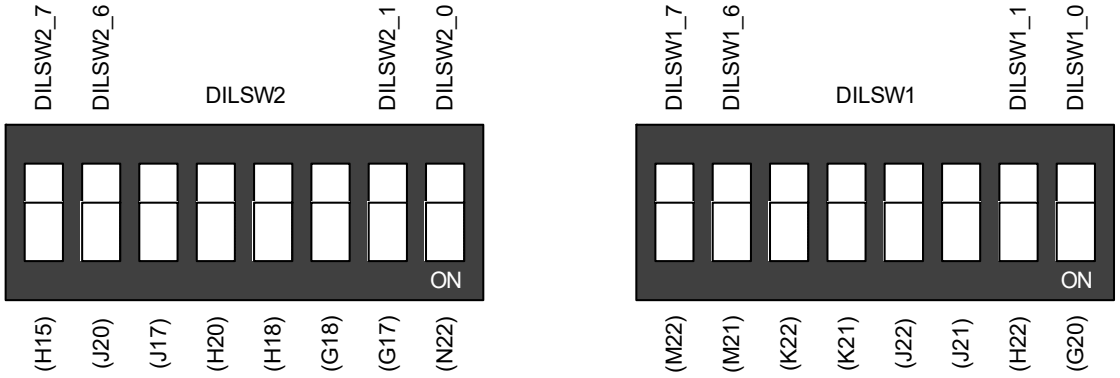


1.4 Buzzer

Le buzzer est relié à la FPGA via un MOSFET canal N. Un état logique haut sur le signal BUZZER (A16) fera sonner le buzzer.

1.5 Dilswitchs 8bit

Deux DIL switchs de 8 bits sont reliés à la FPGA 2. Sur la carte, le bit 0 se trouve tout à droite, et le signal est à l'état logique haut lorsque le switch correspondant est poussé vers le bas (position ON).



1.6 Boutons poussoirs

Les quatre boutons poussoirs permettent de générer un état logique haut lorsqu'ils sont pressés, ils sont donc actif haut. Les poussoirs ont les noms Poussoir0 (U21), Poussoir1 (V20), Poussoir2 (W20) et Poussoir3 (U20), le numéro 0 étant à droite.

1.6.1 Poussoir RESET (actif bas)

Le poussoir RESET (T18) rouge donne un état logique haut au repos et un état logique bas lorsqu'il est activé. Nous avons donc un poussoir actif bas qui sera utilisé comme reset asynchrone dans les systèmes logiques séquentiels synchrones.

1.7 Signaux d'horloge

Sur la FPGA nous avons deux sources de clock possibles, qui sont dirrectement connectés des entrées de la FPGA dédicacées aux signaux d'horloge.

1.7.1 100MHz

Signal d'hologe de 100MHz (J19) qui est la sortie de l'oscillateur externe U10.

1.7.2 CLK_DIV

Sur CLK_DIV (K19), on a un signal signal d'hologe de fréquence variable qui est généré par une CPLD XC2C64A-7VQG44C (U9) à partir du 100MHz ci-dessus. Cette fréquence peut être modifiée à l'aide du commutateur rotatif SW1. Une led rouge (D7) située au milieu du kit clignote à la fréquence du signal CLK_DIV. Ci dessous la table de la fréquence CLK_DIV en fonction de la position du commutateur.

Position commutateur SW1	Fréquence CLK_DIV (Hz)
0	0.25
1	0.5
2	1
3	2.5
4	5
5	10
6	50

7	100
8	1000
9	10000

1.8 Leds bicolores

Quatre leds bicolores permettent de générer 3 couleurs différentes (rouge, vert et orange), le rouge lorsqu'on met un état logique haut sur LED_Rx (x de 1 à 4), le vert lorsqu'on met un état logique haut sur LED_Vx et l'orange lorsqu'on met un état logique haut sur LED_Rx et sur LED_Vx.

1.9 Interface Serial to USB (CP2102-GM)

Un circuit convertisseur serial to USB CP2102-GM (U29) permet d'implémenter une connexion serial USB vers l'extérieur. Le connecteur USB type B est J16. Si l'on veut connecter J16 à un PC, il faut installer le driver fourni par SliconLabs (normalement il s'intalle automatiquement) avant de connecter le câble USB. Le circuit USB est vu comme un port com du côté PC.

1.10 Interface Serial to USB 2 (FT232BQ)

Un circuit convertisseur serial to USB FT232BQ (U34) permet d'implémenter une connexion serial USB vers l'extérieur. Le connecteur mini USB type B est J22. Si l'on veut connecter J22 à un PC, il faut installer le driver fourni par FTDI (normalement il s'intalle automatiquement) avant de connecter le câble USB. Le circuit USB est vu comme un port com du côté PC.

1.11 Convertisseur A/D parallèle (MAX113CAG)

Un convertisseur Analogique/Digital MAX113CAG permet de convertir quatre canaux analogique de 0V à 3.3V en numérique sur 8 bit. Les 4 entrées analogiques du MAX 113 sont les suivantes :

- Le canal 1 peut varier de 0V à 3.3V à l'aide du potentiomètre P1
- Le canal 2 est connecté sur le capteur de température TMP37GRTZ (U5)
- Le canal 3 est connecté sur le connecteur BNC J3 via un suiveur.
- Le canal 4 est connecté sur le connecteur barette 3 pin J5 via un suiveur.

Voir assignation des pattes et datasheet MAX113CAG.

1.12 Convertisseur A/D SPI (ADCS7476AIMFE)

Un convertisseur Analogique/Digital ADCS7476AIMFE permet de convertir une entrée analogique de 0V à 3.3V en numérique sur 12 bit. L'interface analogique sur l'entrée du convertisseur AD est dimensioné pour mesurer une entrée ligne audio, via le connecteur Jack bleu J15. Voir assignation des pattes et datasheet ADCS7476AIMFE.

1.13 Convertisseur D/A SPI (AD5323ARUZ)

Un convertisseur Digital/Analogique 12 bit AD5323ARUZ (U1) permet de convertir une tension numérique en analogique. Il possède deux canaux de sortie. On peut mesurer la tension de sortie du canal A via un suiveur de tension sur le connecteur BNC J2 et celle du canal B sur le connecteur barette 3 pin J4. Le bus sériel synchrone utilisé par le convertisseur D/A est le bus SPI. Le D/A est connecté en slave. Voir assignation des pattes et datasheet AD5323ARUZ.

1.14 Convertisseur D/A Audio (CS4344-CZZ)

Un convertisseur Digital/Analogique audio 24 bit CS4344-CZZ (U33) permet de générer un signal audio line out que l'on peut reprendre sur le connecteur Jack vert J20. Voir assignation des pattes et datasheet CS4344-CZZ.

1.15 Real Time Clock I2C (PCF8563T)

Un circuit externe PCF8563T (U12) contenant une horloge temps réel et un calendrier (RTC) avec oscillateur intégré est interfacé via un bus deux fils I^2C. Son adresse I2C est A2 (W) et A3 (R). Voir assignation des pattes et datasheet PCF8563T.

1.16 EEPROM I2C (24LC256T-I/SN)

Une EEPROM 24LC256T-I/SN (U11) de 256kbit (32k x 8) est également connectée sur le bus I2C. Son adresse I2C est A0 (W) et A1 (R). Voir assignation des pattes et datasheet 24LC256T-I/SN.

1.17 Affichage LCD parallèle (ECC1602B-NSWBBW-91LE)

Un affichage LCD alphanumérique de 16 caractères x 2 lignes ECC1602B-NSWBBW-91LE (U16) avec interface parallèle de 8 bit peut être connecté sur le connecteur femelle correspondant. Le contraste de l'affichage peut être règlé par le potentiomètre P3. Voir assignation des pattes et datasheet ECC1602B-NSWBBW-91LE.

1.18 Driver de led (IS31LT3360-SDLS3)

Un Driver de led IS31LT3360-SDLS3 (U23) est connecté à la FPGA via un signal PWM (pin N13) qui permet d'ajuster le courant, par conséquent la luminosité d'une led blanche (D18) en changeant le rapport cyclique du PWM.

1.19 Connecteur Servo (J23)

Un connecteur Servo de 3 pin (J23) est placé sur la carte. La pin 1 reçoit le PWM de la FPGA (pin V19), la pin 2 est connectée au 5V et la pin 3 au GND.

Pin FPGA	PIN J23	Descrition
V19	1	FPGA IO (PWM)
-	2	Alimentation 5V
-	3	GND

1.20 Connecteur I/O (J13)

Un connecteur de 6 pin (J13) est connecté sur des IO de la FPGA selon le tableau ci-dessous :

Pin FPGA	PIN J13	Descrition
-	1	Alimentation 3.3V
C17	2	FPGA IO (Pull-up 1k)
C18	3	FPGA IO
C19	4	FPGA IO
C20	5	FPGA IO
-	6	GND

1.21 Connecteurs PMOD

Cinq connecteurs de 12 pin (J10, J11, J14, J18 et J19) spécialement prévu pour connecter des modules PMOD qui est un standart défini par DIGILENT http://store.digilentinc.com/pmod-peripheral-modules/. Ci-dessous la définition de chacun de ces connecteurs.

1.21.1 Connecteur PMOD 1 (J10)

Ce connecteur est raccordé sur des IO de la FPGA selon le tableau ci-dessous. Ces signaux sont connectés en différentiel sur la FPGA et peuvent avoir une fonctionalité avec l'XADC si nécessaire.

Pin FPGA	PIN J10	Descrition
E1	1	FPGA IO_P
D1	2	FPGA IO_N
B1	3	FPGA IO_P
A1	4	FPGA IO_N
-	5	GND
-	6	Alimentation 3.3V
G1	7	FPGA IO_P
F1	8	FPGA IO_N
K1	9	FPGA IO_P
J1	10	FPGA IO_N
-	11	GND
-	12	Alimentation 3.3V

1.21.2 Connecteur PMOD 2 (J11)

Ce connecteur est raccordé sur des IO de la FPGA selon le tableau ci-dessous. Ces signaux sont connectés en différentiel sur la FPGA.

Pin FPGA	PIN J11	Descrition
H2	1	FPGA IO_P
G2	2	FPGA IO_N
E2	3	FPGA IO_P
D2	4	FPGA IO_N
-	5	GND
-	6	Alimentation 3.3V
K2	7	FPGA IO_P
J2	8	FPGA IO_N
C2	9	FPGA IO_P
B2	10	FPGA IO_N
-	11	GND
-	12	Alimentation 3.3V

1.21.3 Connecteur PMOD 3 (J14)

Ce connecteur est raccordé sur des IO de la FPGA selon le tableau ci-dessous.

Pin FPGA	PIN J14	Description
P5	1	FPGA IO
N5	2	FPGA IO
L6	3	FPGA IO
L5	4	FPGA IO
-	5	GND
-	6	Alimentation 3.3V
G3	7	FPGA IO
H3	8	FPGA IO
J4	9	FPGA IO
K3	10	FPGA IO
-	11	GND
-	12	Alimentation 3.3V

1.21.4 Connecteur PMOD 4 (J18)

Ce connecteur est raccordé sur des IO de la FPGA selon le tableau ci-dessous.

Pin FPGA	PIN J18	Description
N4	1	FPGA IO
N3	2	FPGA IO
M5	3	FPGA IO
M3	4	FPGA IO
-	5	GND
-	6	Alimentation 3.3V
P6	7	FPGA IO
L3	8	FPGA IO
L4	9	FPGA IO
M6	10	FPGA IO
-	11	GND
-	12	Alimentation 3.3V

1.21.5 Connecteur PMOD 5 (J19)

Ce connecteur est raccordé sur des IO de la FPGA selon le tableau ci-dessous.

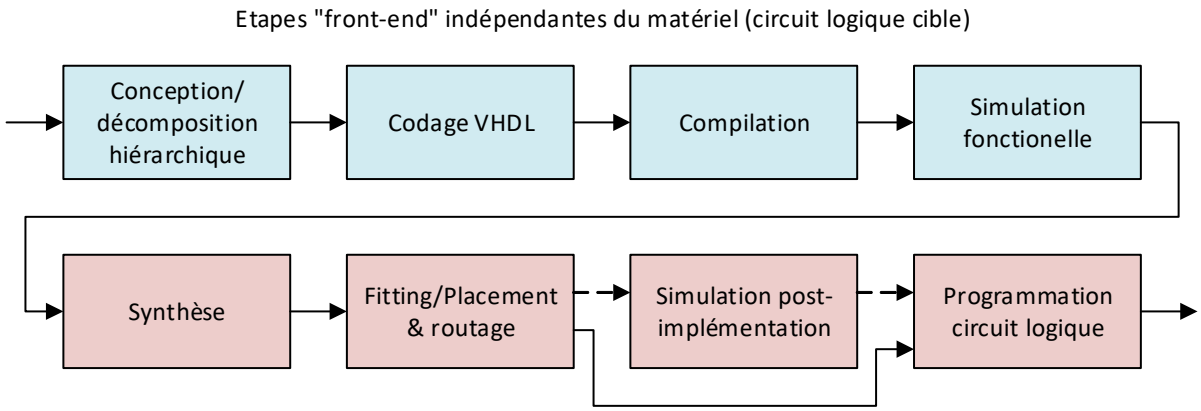
Pin FPGA	PIN J19	Description
R1	1	FPGA IO
P2	2	FPGA IO
P1	3	FPGA IO
N2	4	FPGA IO
-	5	GND
-	6	Alimentation 3.3V
P4	7	FPGA IO
L1	8	FPGA IO
M2	9	FPGA IO
M1	10	FPGA IO
-	11	GND
-	12	Alimentation 3.3V

2. Généralités sur la création d'un projet

2.1 Edition d'un projet

L'édition d'un projet peut se faire de plusieurs manières. La manière la plus simple, mais aussi la moins évoluée, consiste en l'écriture du code en VHDL directement dans un éditeur de texte. Cependant, un éditeur avec des fonctions spéciales conçues pour l'édition du VHDL est inclus dans la plupart des logiciels de développement.

2.1.1 Design Flow



2.2 Simulation fonctionnelle

Une fois la description VHDL synthétisable écrite, on aborde une phase importante du projet, qui est la simulation, pour ce faire il faut éditer un second fichier VHDL généralement nommé TESTBENCH, qui va générer les signaux d'entrée sur le module VHDL à tester, puis contrôler si les états des sorties du module correspondent aux valeurs attendues, on aura besoin d'un simulateur pour cette phase du développement.

2.3 Synthèse logique

La synthèse logique transforme le code VHDL en une « netlist » ou liste de connexions de composants primitifs appartenant au circuit cible. C'est l'apparition des synthétiseurs qui a révolutionné le monde de la conception numérique, car c'est le synthétiseur qui s'occupe de tout l'aspect simplification et génération de la netlist à partir d'une simple description comportementale en VHDL. Il faut cependant faire très attention, le synthétiseur ne dispose pas de l'intelligence nécessaire pour synthétiser toutes les descriptions VHDL, seule une petite partie de la norme VHDL peut être utilisée pour la synthèse logique. En effet à l'intérieur d'un circuit logique, on ne trouve que des portes élémentaires ou des flip-flops, alors que le VHDL permet l'écriture de fonctions haut niveau absolument pas synthétisables, qui par contre sont utilisées dans les fichiers de simulation (testbench).

2.4 Fitter ou placement routage

Le rôle du fitter est d'essayer d'intégrer les fonctions décrites par la netlist générée par le synthétiseur dans le circuit de logique programmable utilisé. Ce dernier va par exemple contrôler que le nombre de bascules disponibles dans le circuit est au moins aussi grand que celui nécessaire à la réalisation de la fonction décrite par l'utilisateur. C'est également lui qui va s'occuper du routage physique du circuit de logique programmable en tenant compte des contraintes définies par l'utilisateur (N° de pattes, temps de propagations max de certains signaux, …). Il va par exemple décider d'utiliser telle ou telle porte NAND pour réaliser une certaine partie de la fonction finale.

Comme le fitter est étroitement lié au chip physique, ce dernier est fournit par le fabricant du circuit intégré utilisé.

2.5 Programmation

La FPGA sur le kitXilinx est du type ISP (In Circuit Programmable) elle possède donc un port de programmation JTAG, et elle peut se configurer tout en étant soudée sur le circuit imprimé. Une fois configurée, elle perd sa configuration en cas de coupure de courant. C'est pour cette raison qu'une mémoire FLASH non volatile est connectée à la FPGA. Cette mémoire peut être programmée, et à chaque démarrage de l'alimentation, ou pression sur SW2 (PROG) la Flash charge la configuration dans la FPGA. La Flash ne perd pas son contenu même en cas de coupure d'alimentation.
Pour les circuits ISP, il ne faut qu'un ordinateur muni d'une interface USB qu'un câble de download. Pour programmer un circuit ISP, il suffit donc de relier le port du PC au port de programmation du chip via un câble de programmation, et de lancer le programme de configuration fournit par le fabricant du circuit.

3. Mode d'emploi du Kit Xilinx 7 et utilisation de Vivado

3.1 Mise sous tension du kit

La mise sous tension de ce produit se fait en introduisant le connecteur de l'alimentation (+6V) dans le connecteur jack d'alimentation (J9) situé en haut à gauche de la carte.

3.2 Exemple pratique de programmation avec Xilinx Vivado

3.2.1 Informations générales sur Xilinx Vivado

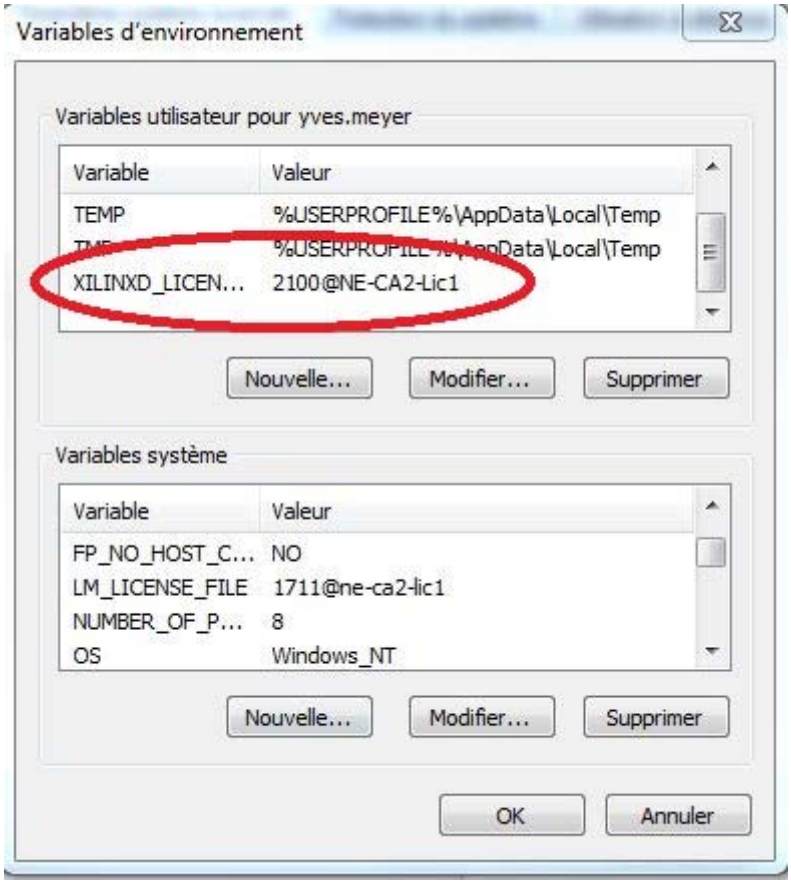
Le programme Xilinx Vivado est un logiciel pour implémenter les FPGA de Xilinx, et que l'on peut télécharger (après s'être inscrit en ligne) sur Internet. Il existe une version gratuite du logiciel, mais pour les cours nous utilisons la version payant avec license flottante.

3.2.2 Installation Vivado

L'installation de Vivado se fait très simplement, il suffit de télécharger le module d'installation sur le site www.Xilinx.com (il faut être enregistré) et de l'éxécuter. Ensuite, il suffit de se conformer aux instructions données par le programme d'installation et de répondre aux différentes questions posées par ce dernier.

3.2.2.1 License

Il faut ajouter la variable d'environnement suivante pour pouvoir utliser Vivado :



3.2.3 Création d'un nouveau projet

La première étape est de lancer l'application Vivado.

Pour cela il faut aller dans le menu Démarrer sous **Programmes** puis sous **Xilinx Design Tools** puis cliquer **Vivado 2018.2** .

Une fois que l'application a démaré, il faut cliquer dans la fenêtre Quick Start sur :

Create Project > pour créer un nouveau projet

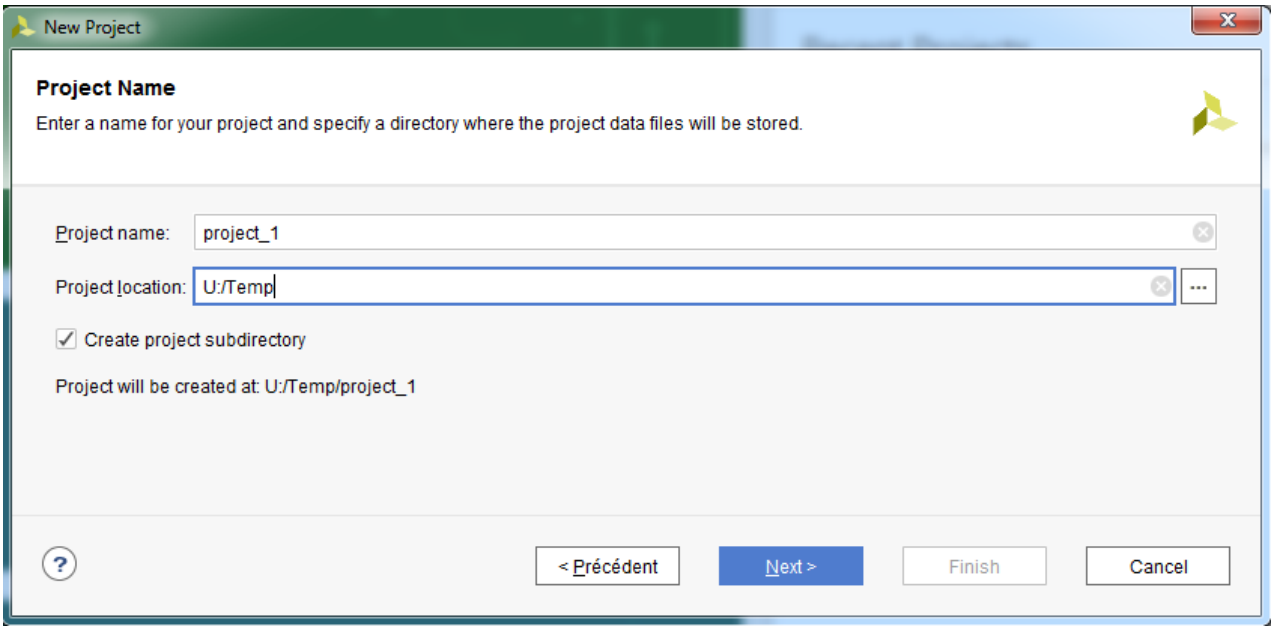
Open Projet > pour ouvrir un projet existant

Dans la fenêtre de doite **Recent Projects** vous avez des raccourcis pour ouvrir un projet récent

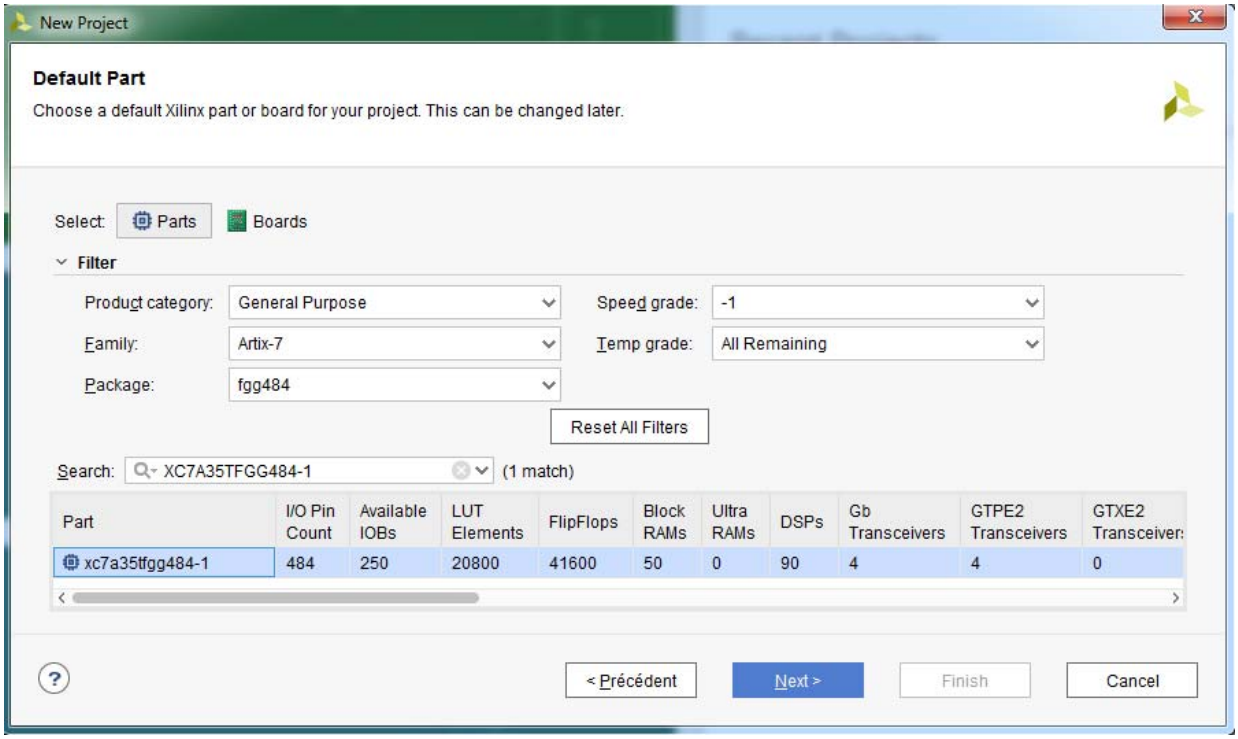
Une fois que l'on a choisi **Create Project** effectuer les actions dans l'ordre suivant :

1. **Next >**
2. **Project location** on va choisir le répertoire dans lequel le projet sera créé.

3. **Project name** contiendra le nom du projet (**sans caratères spéciaux ni espace**). Le logiciel va ajouter automatiquement un sous répertoire au nom du projet dans Location dans lequel devront se trouver **tous** les fichiers du projet. La case Create project subdirectory doit être cochée.



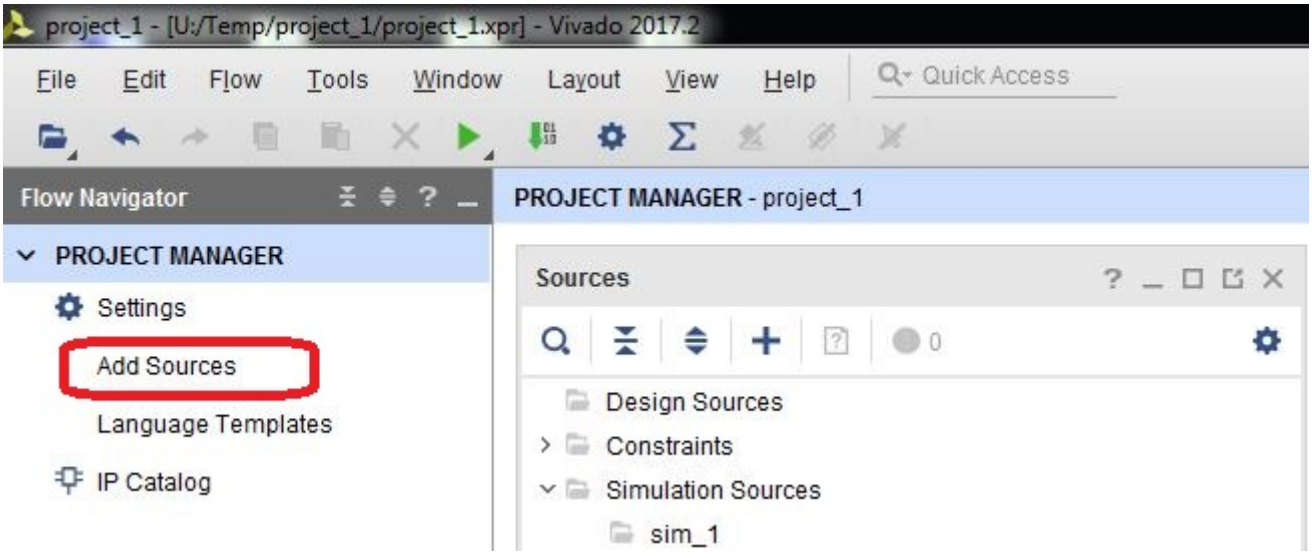
4. **Next >**
5. Sélectionner **RTL Projet**.
6. **Next >**
7. **Add Sources**, permet d'ajouter des fichiers sources (Ex : fichiers VHDL) au projet, mais nous le ferons plus loin. Par contre choisir **Target language** et **Simulator language** VHDL
8. **Next >**
9. **Add Constraints** (optional) permet d'ajouter des fichiers de contraintes, nous le ferons également plus tard.
10. **Next >**
11. **Default Part** permet de choisir la FPGA que nous avons sur le Kit Xilinx 7 à savoir de la famille Artix-7 **XC7A35TFGG484-1**



12. **Next >**
13. Dans la fenêtre New Project Summary vous avez un résumé de votre projet cliquer **Finish**

Nous venons donc de créer un nouveau projet nommé **projet_1.xpr**. Un projet seul ne sert à rien, il faut maintenant ajouter une ou plusieurs sources au projet. Ces sources dans notre cas seront des fichiers VHDL ou des fichiers de contraintes.

Dans la fenêtre Project Manager cliquer sur **Add Sources**



Dans la fenêtre **Add Sources** choisir :

- Add or create constraints — si l'on veut créer un fichier de contraintes
- Add or create design sources — si l'on veut créer un fichier VHDL
- Add or create simulation sources — si l'on veut créer un banc de test pour la simulation

Dans notre cas nous allons choisir **Add or create design sources**

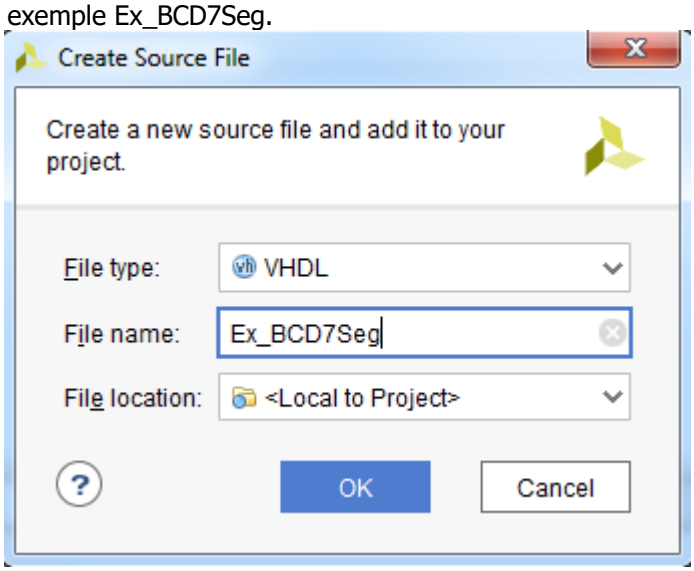
Cliquer **Next>**

Dans la fenêtre **Add or Create Design Sources** choisir :

- **Add files** pour ajouter des fichiers sources déjà existants dans le projet. Attention : activer la **case Copy Sources into project** si l'on veut copier un fichier source provenant dans autre projet.
- **Add Directories** pour ajouter tous les fichiers source d'un répertoire.
- **Create File** pour créer un nouveau fichier source.

Dans notre cas nous allons choisir **Create File**

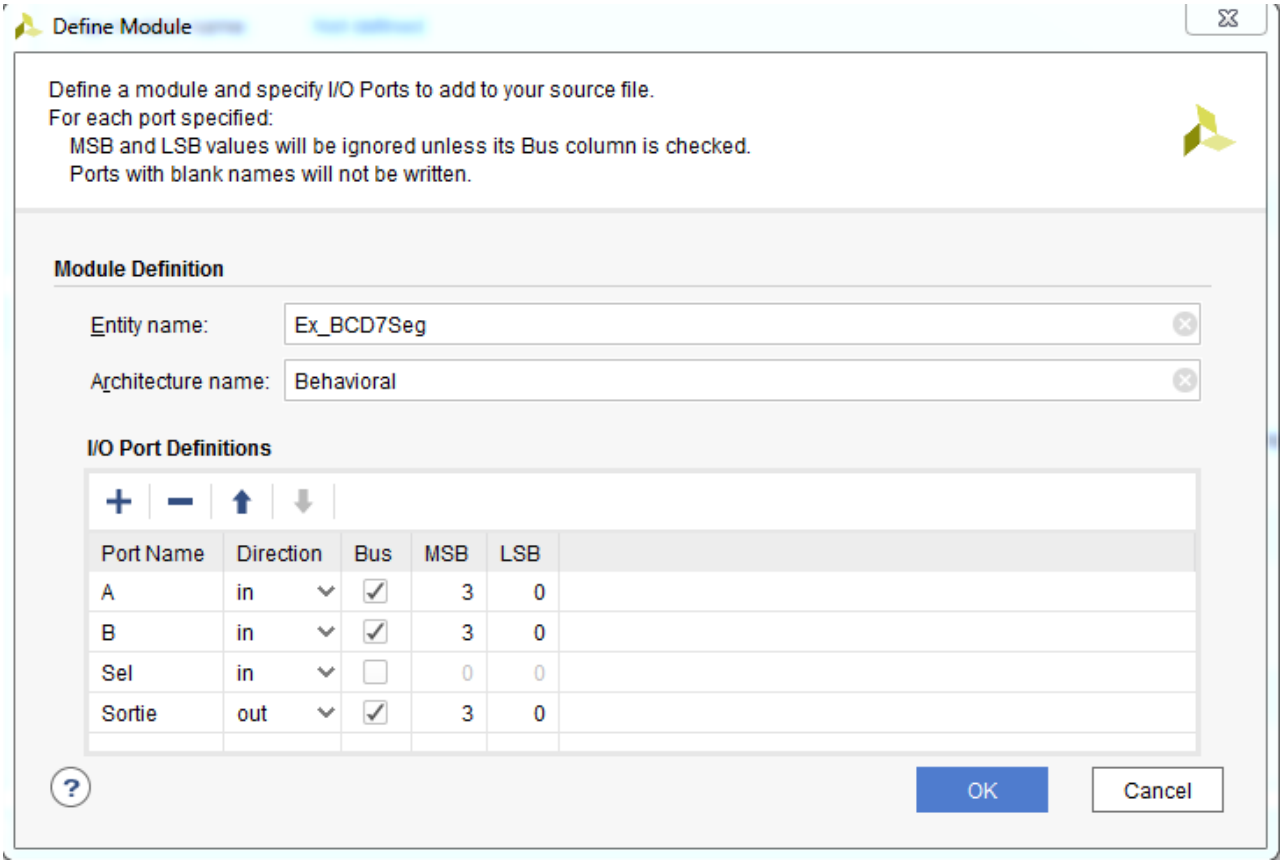
Dans la fênetre **Create Source File**, donner un nom au fichier sources que nous allons créer (chiffre, lettre ou _ pas d'autres caractères), exemple Ex_BCD7Seg.



Cliquer **OK**
Cliquer **Finish**

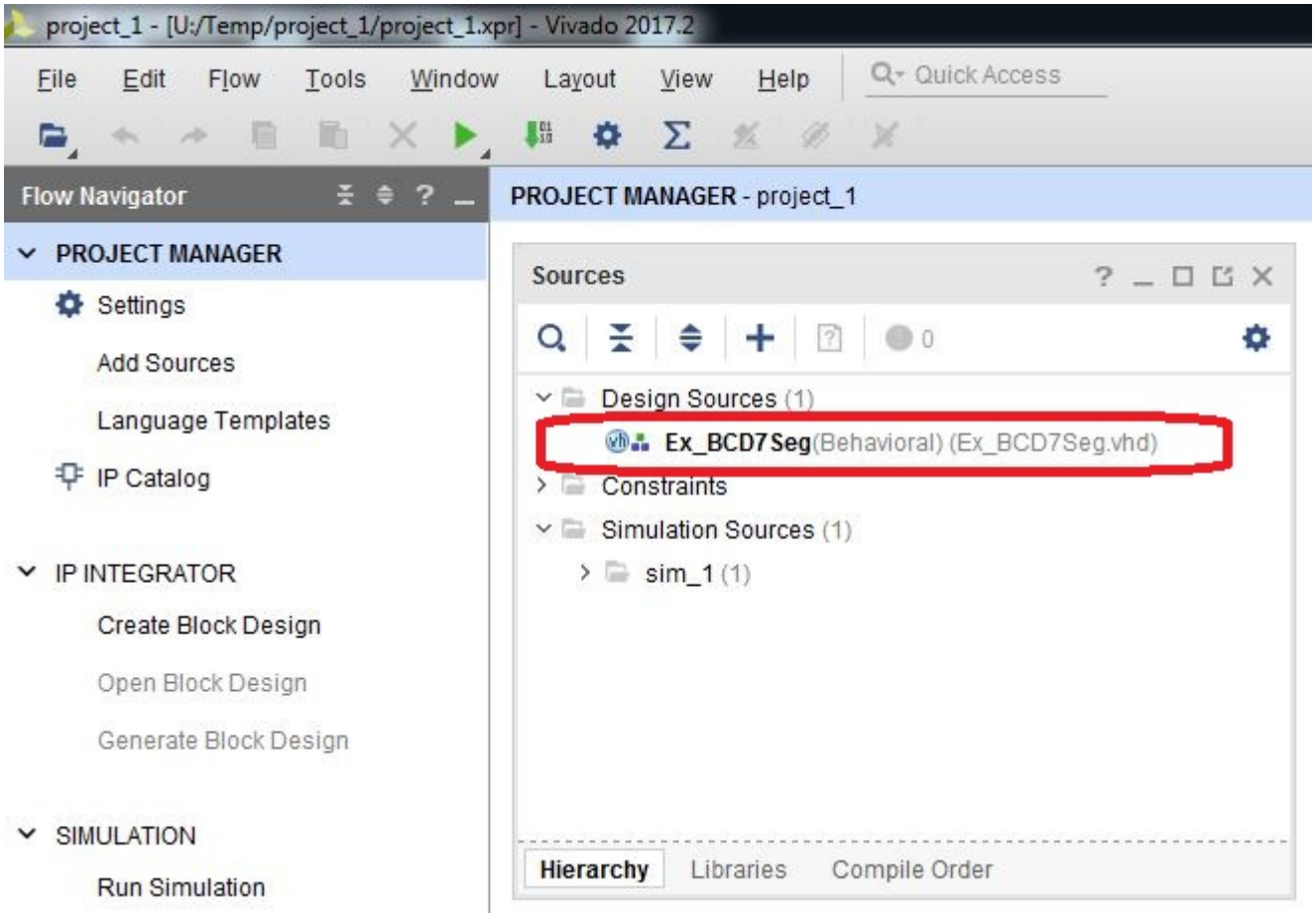
Une fenêtre **Define Module** permettant de spécifier les entrèes/sorties de notre bloc (entity VHDL) s'ouvre, on peut l'utiliser, mais ce n'est pas obligatoire, on peut très bien compléter le fichier VHDL une fois créé.

Si on veut l'utiliser, entrer le non que l'on veut donner à tous les signaux de l'entity dans Port Name, en prenant soin de modifier leur Direction et leur taille (MSB et LSB) si c'est un vecteur (cocher Bus), exemple :

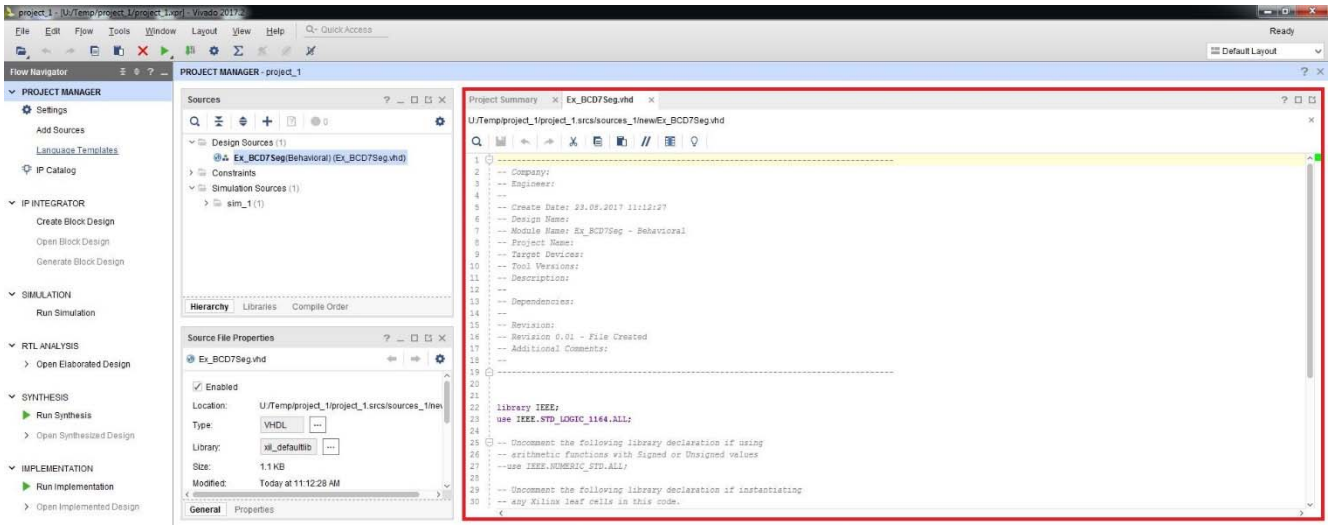


Cliquer **OK**

On peut voir dans la fenêtre **Sources** du **Project Manger** que le fichier source Ex_BCD7Seg.vhd à été créé

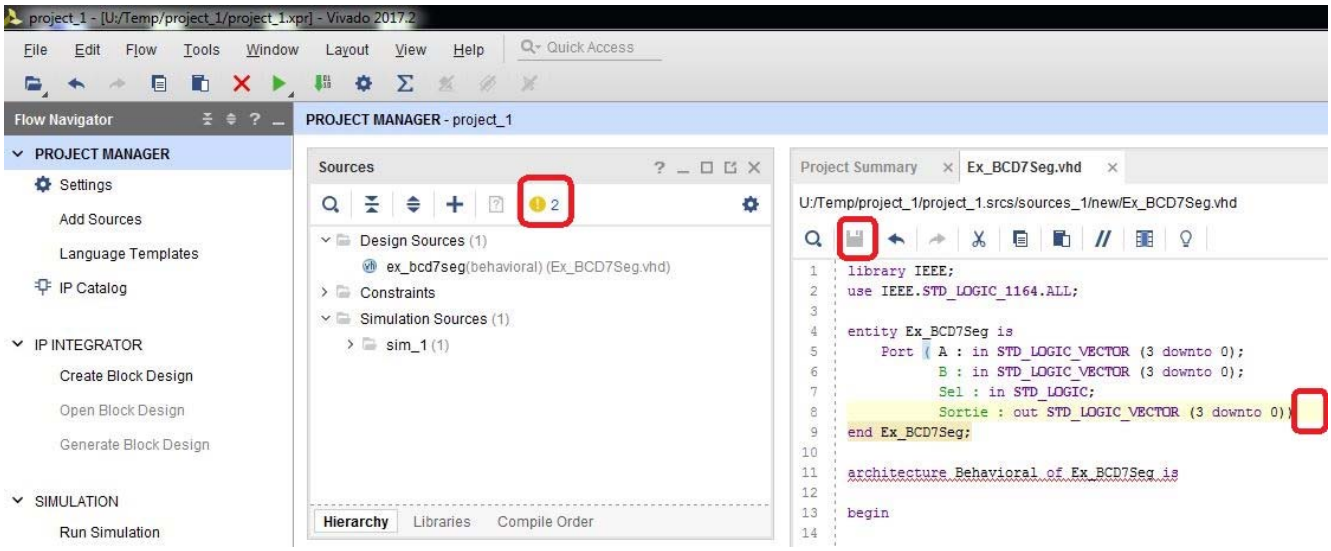


Il suffit de faire un double-clic sur le fichier source qui va s'ouvrir dans l'éditeur de texte dans la fenêtre principale, on peut alors compéter la description VHDL.



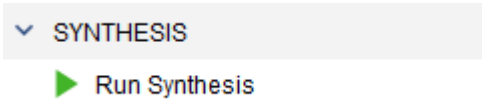
3.2.4 Edition du fichier VHDL

Vous pouvez éditer votre fichier VHDL, mais avant la synthèse, il faut supprimer toutes les fautes de syntaxe. Lorsque vous **sauvegardez** votre fichier avec la disquette en haut à gauche de la fenêtre d'édition, le message : **point d'exclamation orange** apparait dans la fenêtre **Sources** si vous avez des erreurs de syntaxe.



Dans cet expemple, il manque le point virgule à la fin de l'entité, mais si vous ne trouvez pas l'erreur vous pouvez cliquer sur le lien **hyper texte (numéro) à droite du point d'exclamation** pour mieux cibler les erreurs, un onglet **Messages** s'ouvre en bas de votre ecran. Grace aux explications de cette fenêtre, vous pourer mieux cibler les erreurs.

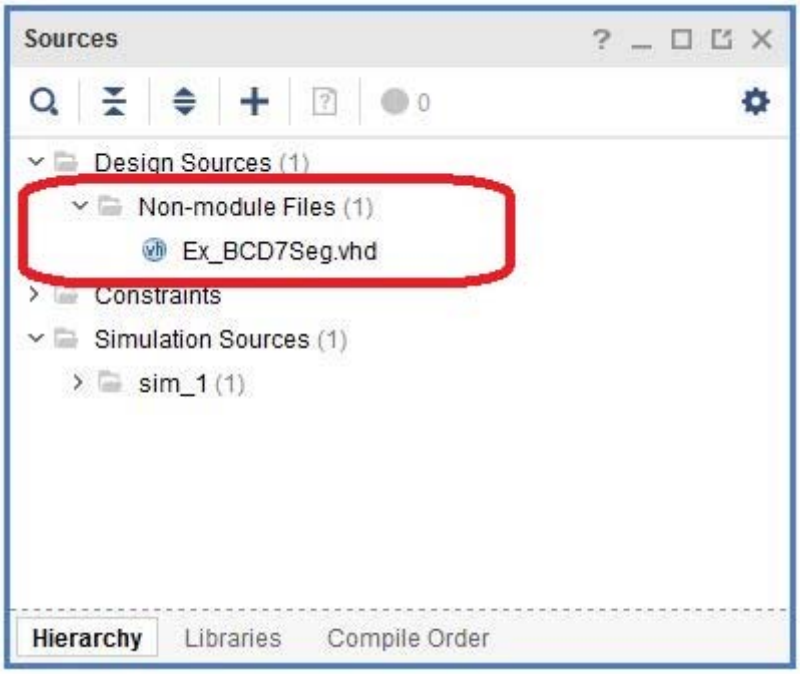
Une fois ces erreurs de syntaxe corrigées, vous pourrez lancer la synthèse en cliquant sur **Run Synthesis** dans la fenêtre **Project Manager** puis laisser les options par défaut et **OK**



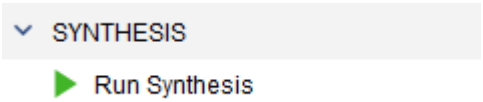
3.2.5 Synthèse logique

3.2.5.1 Erreurs de syntaxe dans le fichier VHDL

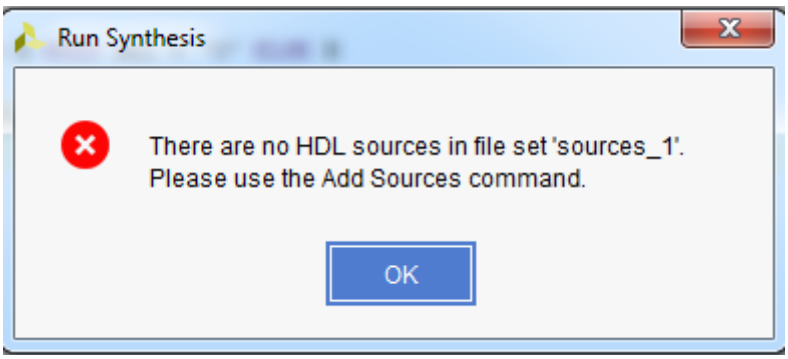
Si votre fichier VHDL est synthétisable, la synthèse ira à son terme, si ce n'est pas le cas votre fichier ira se placer dans un répertoire du projet appellé *Non-module Files*



Et lorsque vous activez la synthèse avec :



Vous avez le message d'erreur suivant

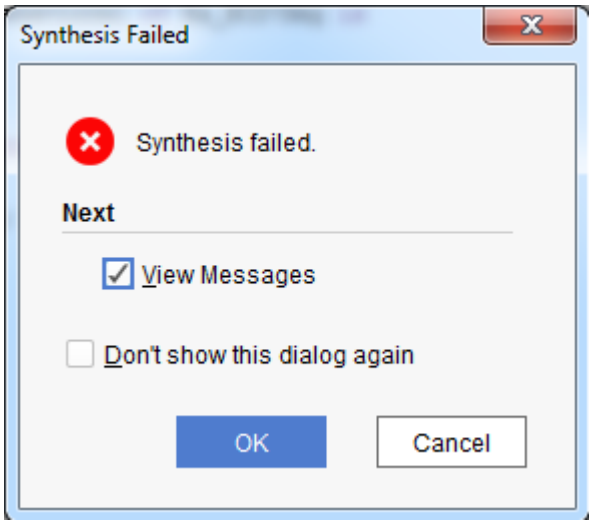


⇨ **IL faut corriger les erreurs de syntaxe pour pouvoir lancer la synthèse**

3.2.5.2 Erreurs de synthèse

Une fois que la synthèse a démarré, vous pouvez encore avoir des erreurs, qui sont cette fois des erreurs de synthèse, exemple des erreurs sur les tailles des vecteurs.

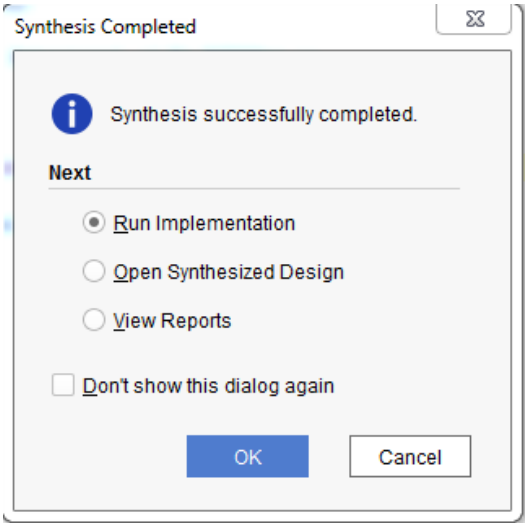
En cas d'erreur de synthèse le message suivant apparait :



Cliquer **OK**

Dans la fenêtre **Messages**, vous aurez une description des erreurs empêchant la synthèse :

Une fois ces erreurs corrigées, relancer la synthèse avec **Run Synthesis**, une fois celle-ci complète, vous aurez le message suivant :



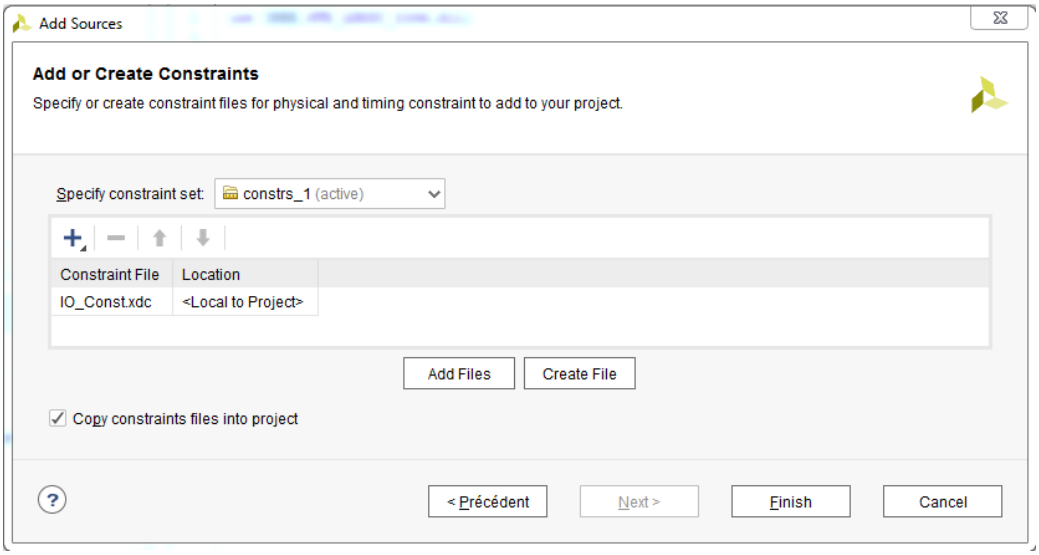
On peut directement lancer l'Implémentation et cliquer OK si l'on a déjà un fichier de contrainte dans le projet. Ce fichier de contraintes indique quel signal de notre système numérique doit aller sur quelle patte de la FPGA. Si l'on n'a pas ce fichier de contrainte dans le projet, il faut cliquer **Cancel**

3.2.6 Assignation des signaux sur les pattes de la FPGA (création du fichier de contraintes)

3.2.6.1 Méthode manuelle

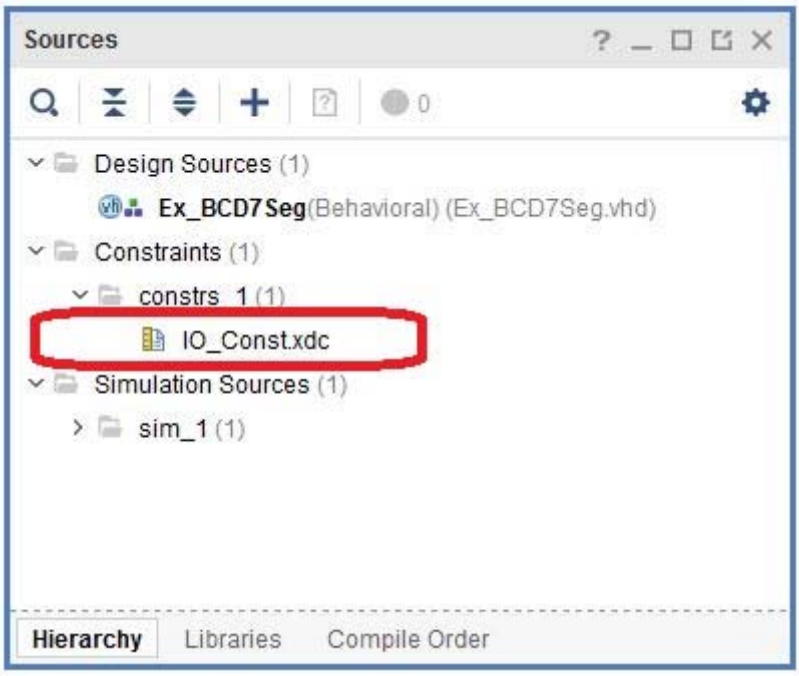
Les signaux présents dans l'entité du fichier TOP (au niveau hiérarchique) peuvent être assignés selon les désirs de l'utilisateur sur les pattes de la FPGA, pour ce faire nous devons créer un nouveau fichier source avec l'extension .xdc (User Constraints File).

1. Dans **Project Manager** cliquer sur **Add Sources**
2. Choisir **Add or create constraints**
3. **Next >**
4. **Create File**
5. Choisir un nom pour le fichier (ex : IO_Const)
6. Cocher la case **Copy constraints file into project**
7. Cliquer **OK**



8. **Finish**

Le fichier de contrainte qui figure dans la fenêtre Sources doit être édité



Il suffit de double-cliquer sur **IO_Const.xdc** qui va s'ouvrir dans un onglet dans la fenêtre d'édition.

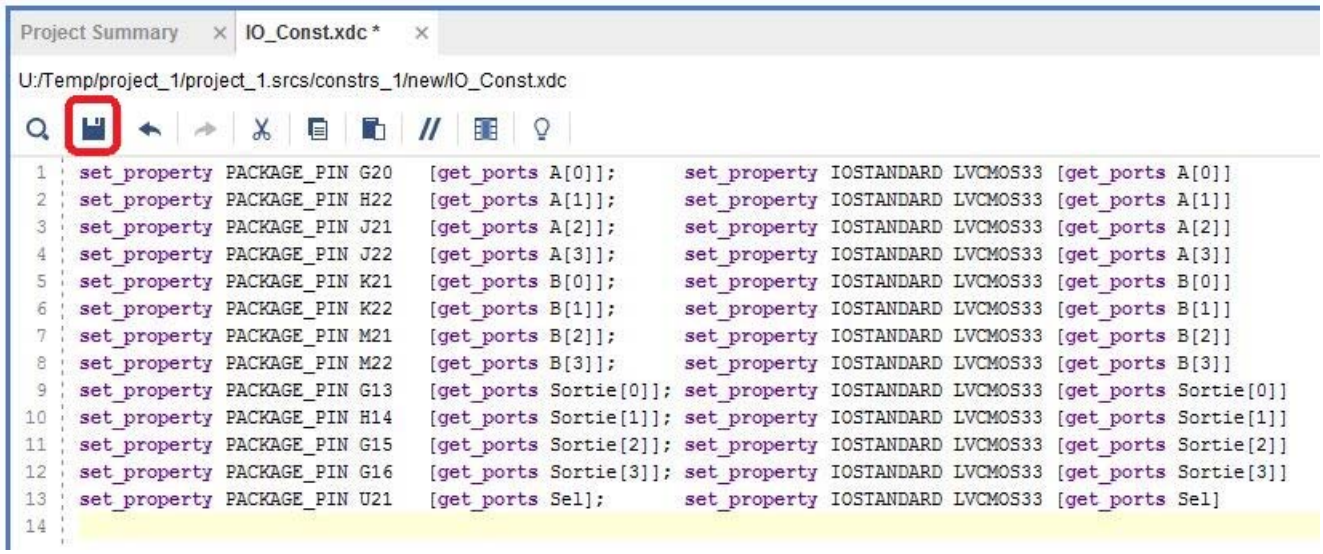
3.2.6.2 Edition manuelle du fichier de contraintes

La première méthode consiste à éditer manuellement le fichier de contrainte avec les 2 points suivants :

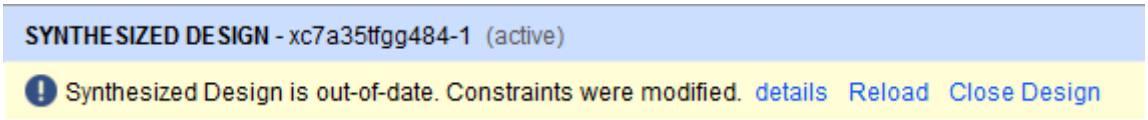
1. Quel signal va sur quel patte de la FPGA, utiliser la liste des IO du kit Xilinx 7 (voir chap 4.)
2. Quel est le niveau électrique de la patte sur la FPGA, pour tout le kit le niveau est LVCMOS33 (3.3V)

Ci-dessous un exemple de fichier de contraintes avec la syntaxe à utiliser :

```
set_property PACKAGE_PIN J19 [get_ports clk]                (signal clk sur patte J19 de la FPGA)
set_property IOSTANDARD LVCMOS33 [get_ports clk]            (signal clk niveau LVCMOS33)
set_property PACKAGE_PIN T18 [get_ports reset_n]            (signal reset_n sur patte T18 de la FPGA)
set_property IOSTANDARD LVCMOS33 [get_ports reset_n]        (signal reset_n niveau LVCMOS33)
….
```

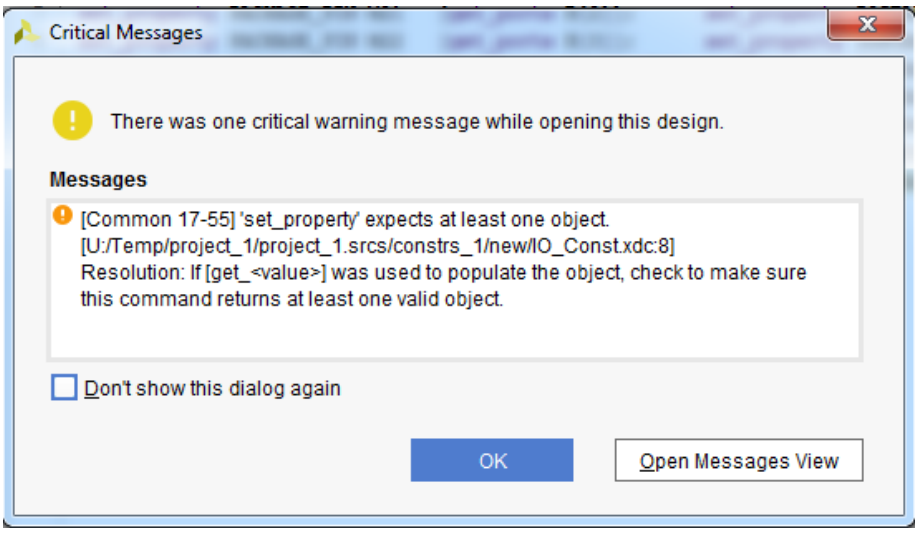


Lorsque vous sauvez votre fichier de contrainte en cliquant sur la disquette en haut à gauche, le message suivant apparait :



Cliquer **Reload**

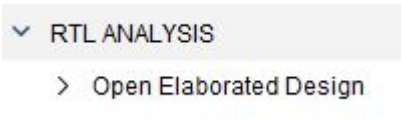
Si vous avez une erreur dans votre fichier de contrainte, la fenêtre suivante apparait :



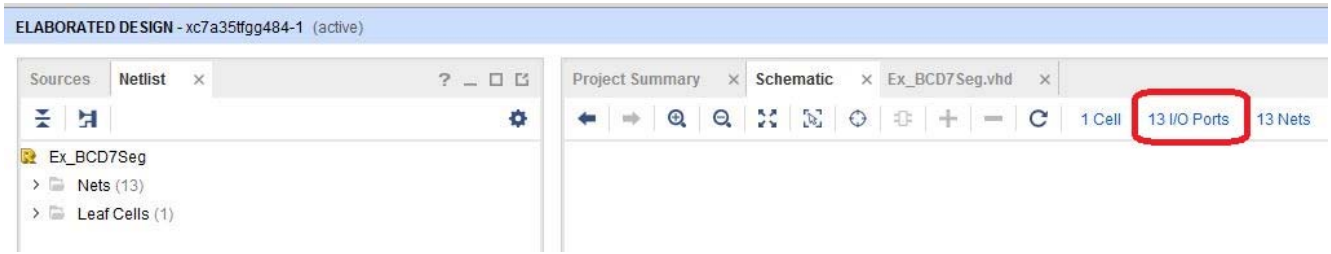
Corriger le fichier de contrainte jusqu'à ne plus avoir d'erreur.

3.2.6.3 Edition fichier de contraintes dans l'Elaborated Design

La deuxième méthode consiste à éditer le fichier de contrainte dans une fenêtre avec selection des possiblités. Une fois la synthèse terminé, cliquer sur **Open Elaborated Design** dans le menu RTL ANALYSIS

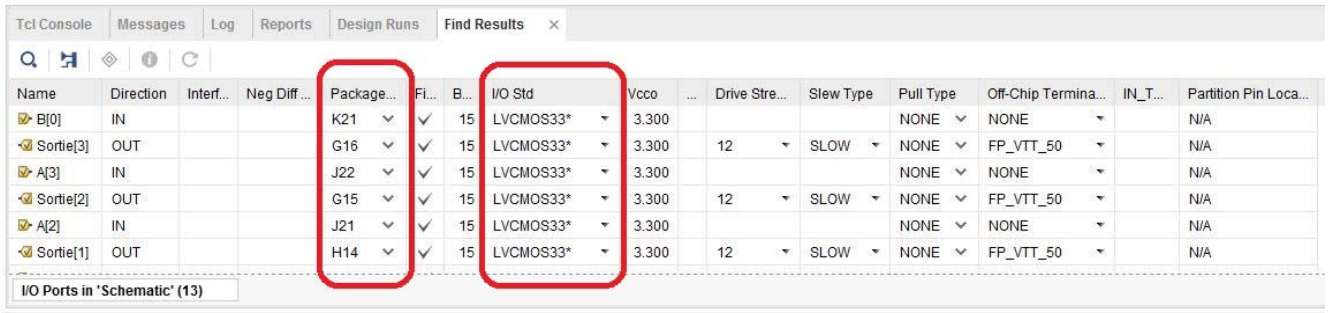


Dans la fenêtre ELEBORATED DESIGN qui s'ouvre si tout est correct, cliquer sur **xx I/O Ports**



Editer les 2 champs suivants dans l'onglet **Find Results** au bas de la fenêtre :

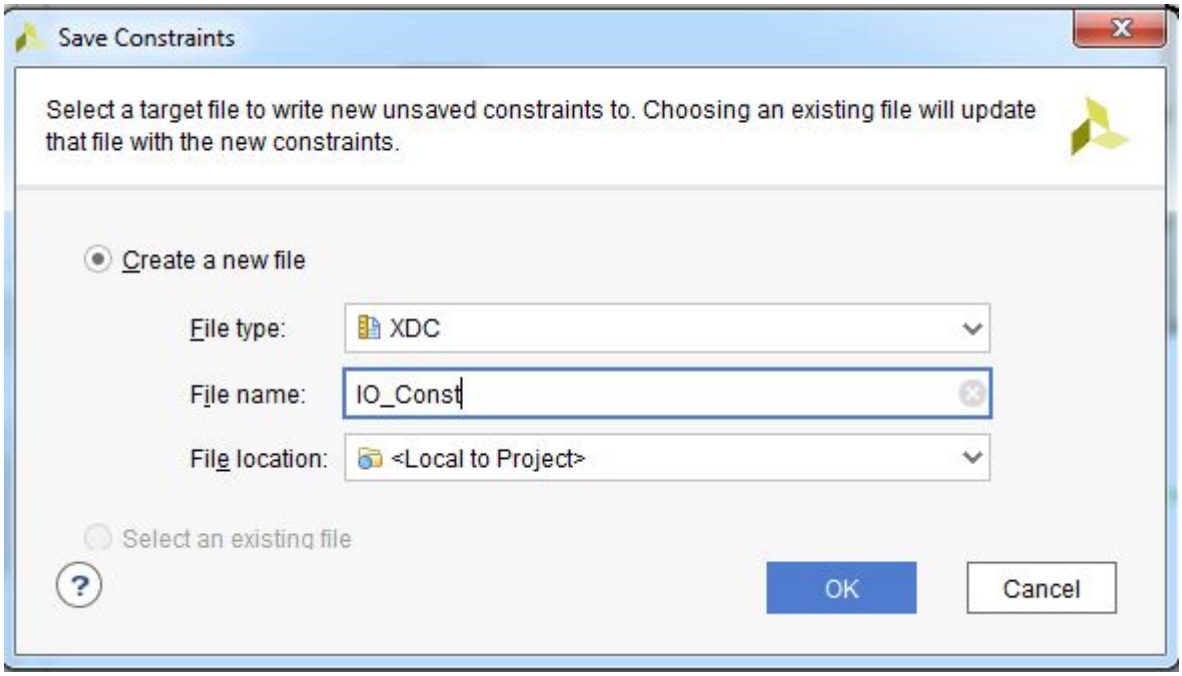
I/O Std choisir LVCMOS33 pour tous les ports
Package Pin choisir les pattes désirées sur la FPGA pour chaque signal



Cliquer sur la **disquette** en haut à gauche pour sauver le fichier de contrainte

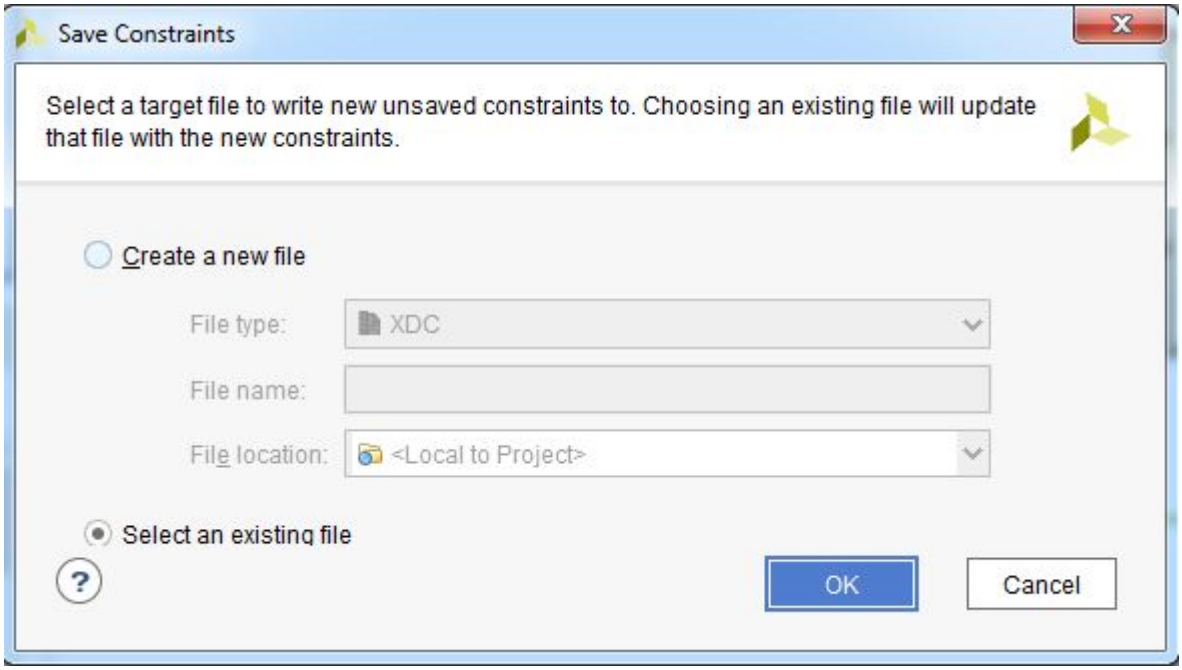
La vous avez **deux possiblités** :

1) si le fichier de contraintes n'a pas encore été créé, il faut sélectionner **Create a new file** et donner un nom au fichier de contraintes ex : IO_Const.



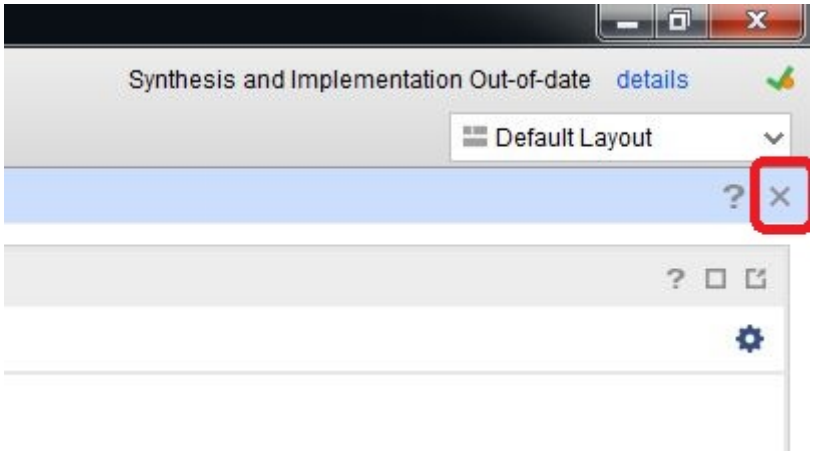
OK

2) Si le fichier de contrainte existe déjà dans le projet, il faut sélectionner Select an existing file



OK

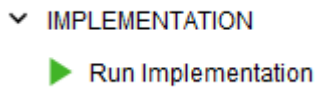
Quitter l'Elaborated Design en cliquant sur le x de la ligne bleue en haut à droite de l'écran.

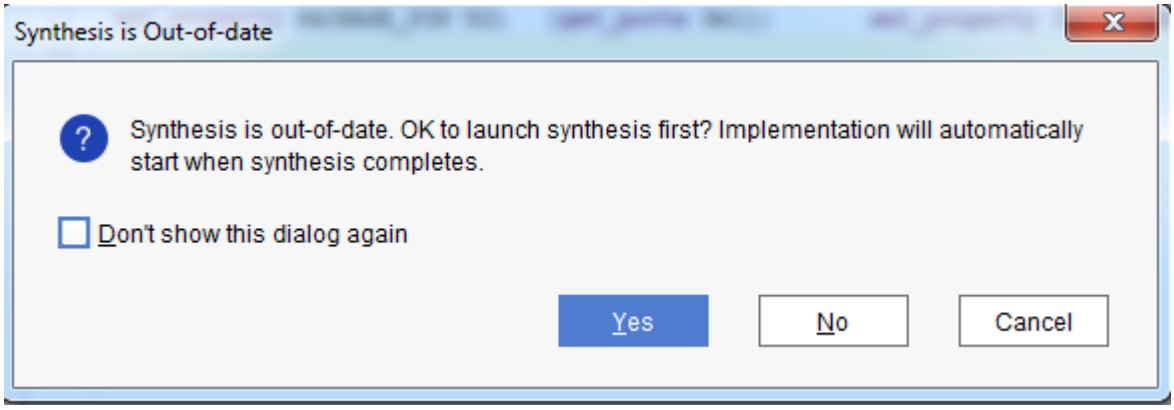


Après sauvegarde, vous pouvez ouvrir et éditer manuellement le fichier de contraintes avec l'éditeur de texte si nécessaire.

3.2.7 Implementer le projet

Pour Implementer votre projet (Place & Route), il faut cliquant sur **Run Implementation** dans la fenêtre **Project Manager**



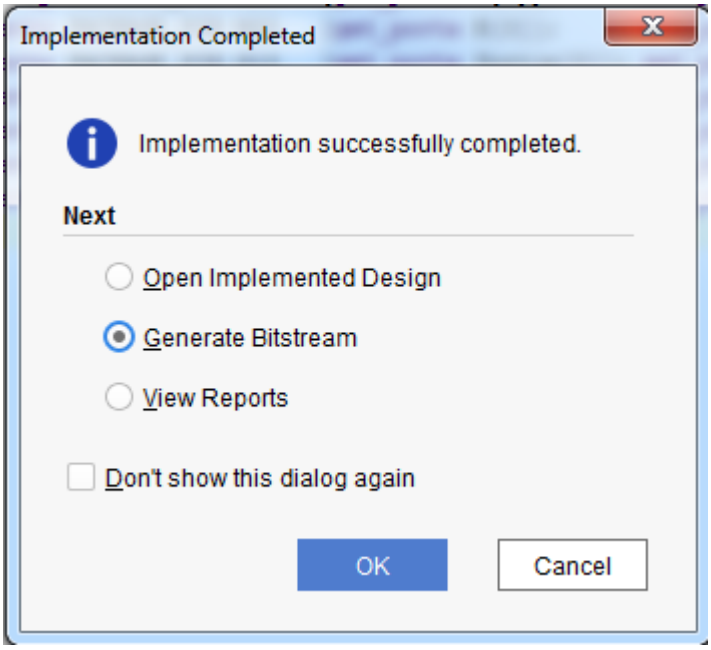


Yes
OK

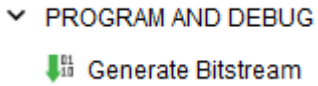
3.2.8 Générer le fichier de configuration

Pour générer le fichier de configuration (Bitstream) à envoyer é la FPGA, il faut cliquer sur **Generate Bitstrean**

1) Soit directement depuis la fenêtre qui s'ouvre après l'implémentation :



2) Soit depuis la fenêtre **Project Manager**



OK

Si une erreur survient dans cette étape, c'est du à une erreur dans le fichier des contraintes (signal non assigné ou I/O Std non défini pour un signal).

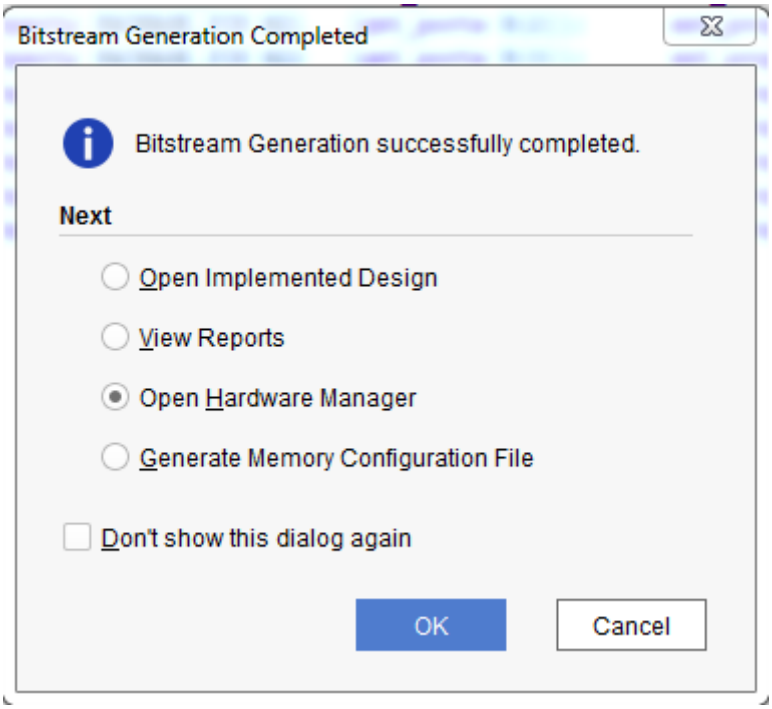
3.2.9 Programmer la FPGA

Cette méthode configure directement la FPGA, sans utiliser la Flash, donc si l'on coupe et que l'on rebranche l'alimentation, la configuration de la FPGA sera perdue.

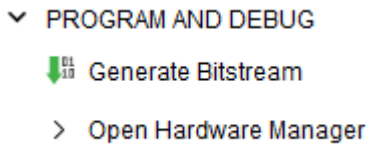
Alimenter le kit Xilinx 7 et connecter un port USB du PC, soit sur le module USB-JTAG avec un cable USB micro, soit sur le connecteur JTAG J12 via un Platform Cable USB de Xilinx.

Il faut lancer le **Hardware Manager**

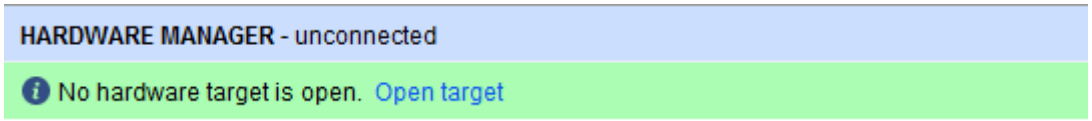
1) Soit directement depuis la fenêtre qui s'ouvre après l'implémentation :



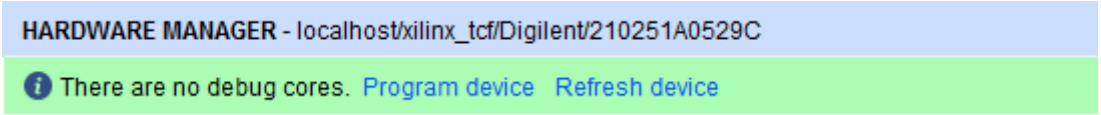
2) Soit en cliquant sur **Open Hardware Manager** dans la fenêtre **Project Manager**



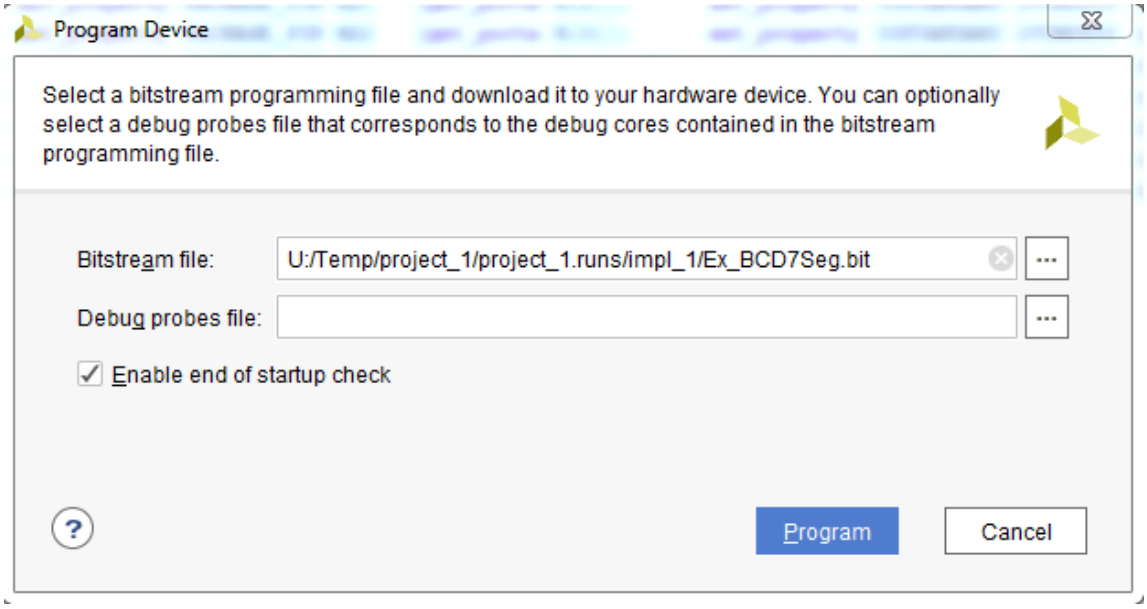
Puis cliquer sur le lien **Open target** en haut de la fenêtre Hardware Manager et choisir **Auto Connect**



Si la carte est correctement branchée la cible s'ouvre comme affiché dans la fenêtre ci-dessous et il suffit de cliquer sur le lien **Program device** en haut de la fenêtre Hardware Manager et choisir **xc7a35t**



Il suffit de choisir le fichier de configuration (xxx.bit) qui normalement est celui de notre projet



Cliquer **Program**

La lampe verte **DONE** au milieu du kit Xilinx 7 doit s'être allumée ci qui signifie que la FPGA est configurée. Vous pouvez tester votre application.

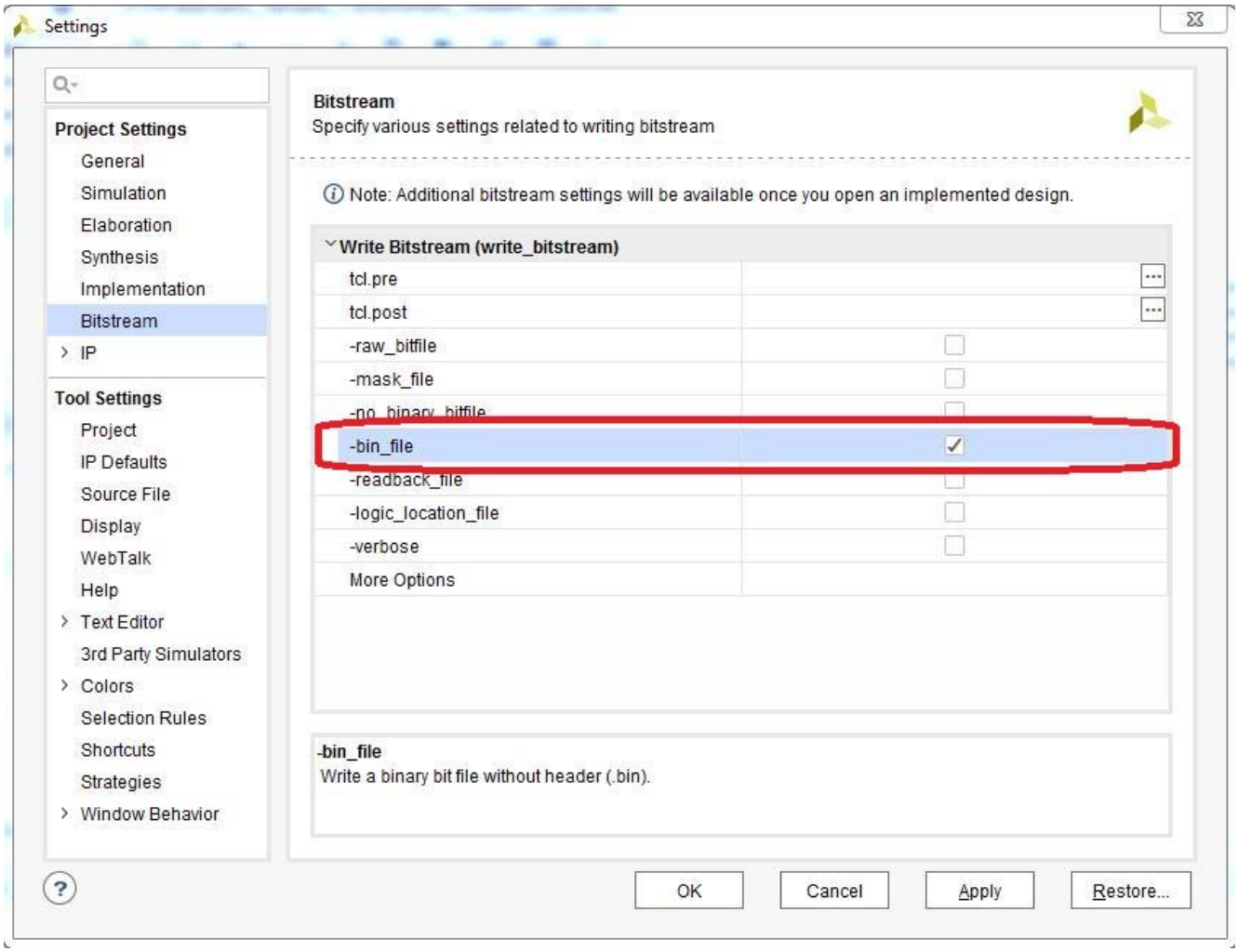
Remarque :

Cette méthode configure directement la FPGA, sans utiliser la Flash SPI, donc si l'on coupe et que l'on rebranche l'alimentation, la configuration de la FPGA sera perdue.

3.2.10 Programmer la FPGA avec la mémoire Flash SPI Micron (N25Q256A13ESF40G)

Si l'on aimerait que la configuration de la FPGA soit automatiquement chargée à la mise sous tension de la carte (ce qui n'est pas nécessaires pour les exercices), on peut configurer une mémoire Flash connectée à la FPGA par bus SPI en mode SPI 4 bits.

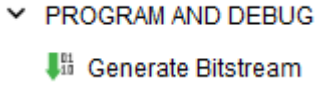
Il faut d'abord générer un autre fichier de configaration (xxx.bin), pour ce faire clic droit sur Generate Bitstream et choisir **Bitstream Settings...**



Dans la fenêtre **Project Sttings**, cocher l'option **–bin_file**

Cliquer **OK**

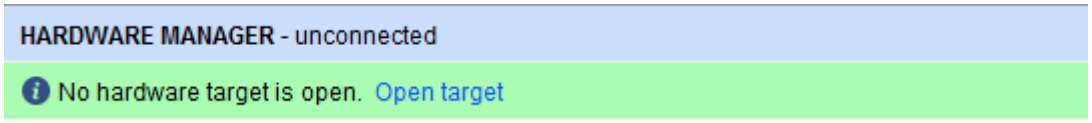
Pour générer le fichier de configuration (Bitstream) à envoyer é la FPGA, il faut cliquer sur **Generate Bitstrean** dans la fenêtre **Project Manager**



Cliquer **OK**

Une fois la génération des fichiers de contrainte terminée, clique sur **Open Hardware Manager**

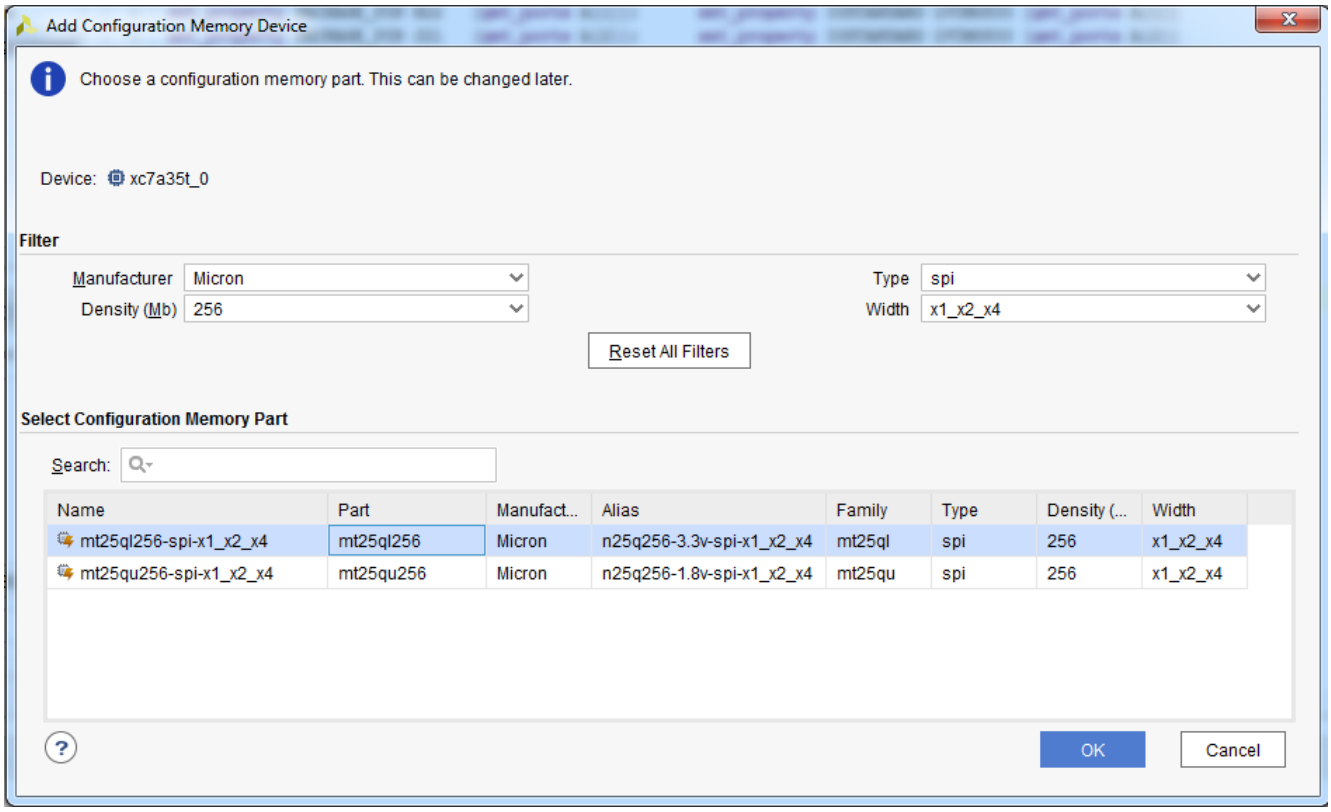
Puis cliquer sur le lien **Open target** en haut de la fenêtre Hardware Manager et choisir **Auto Connect**

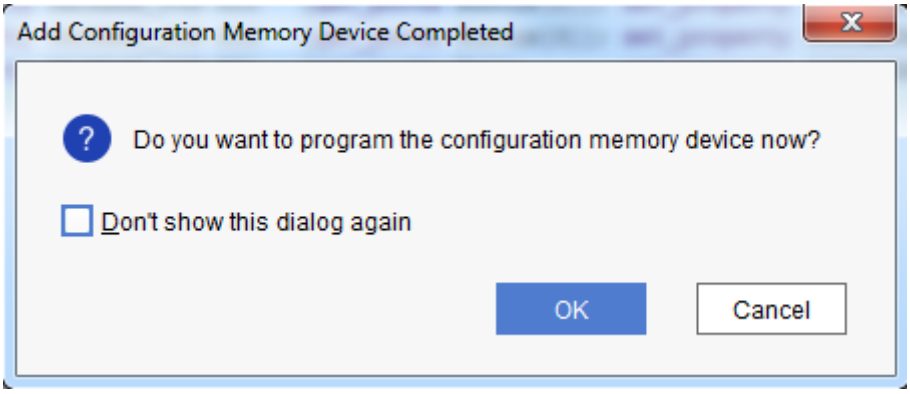


Cliquer sur **Add Configuration Memory Device** dans le menu **Hardware Manager**

Choisir **xc7a35t**

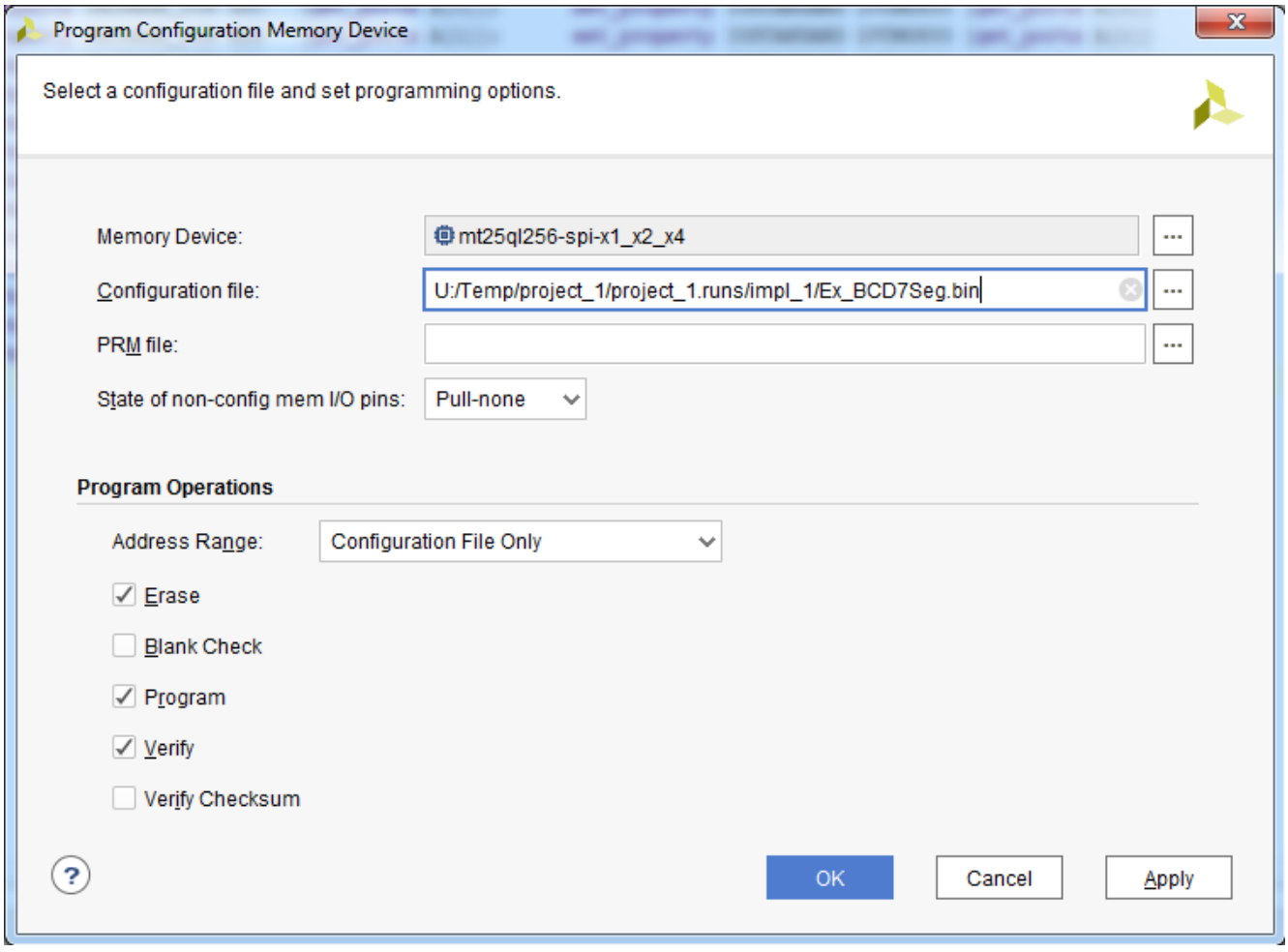
Dans la fenêtre **Add Configuration Memory Device**, choisir la flash n25q256 en 3.3V



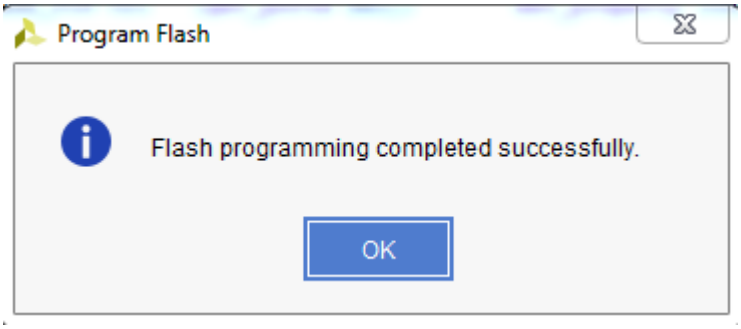


Cliquer **OK**

Slectionner le fichier de configuration (xxx.bin) dans **Configuration file :**



Cliquer **OK**



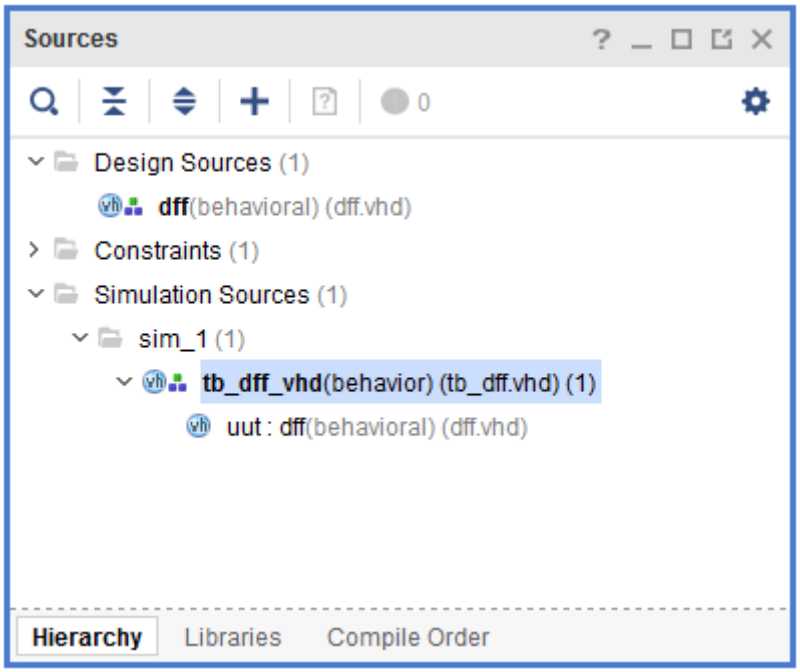
La Flash est programmée, il suffit de presser sur le switch **PROG** à gauche de la led DONE sur le kit Xilinx 7 et attendre que la led verte **DONE** s'allume.

3.3 Simulation

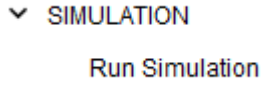
Pour faire une simulation, il faut tout d'abord créer un banc de test (testbench) dans le projet, selon la méthode ci-dessous :

1. Donc dans **Project Manager** il faut sélectionner **Add Sources**
2. Dans la fenêtre Add Sources, sélectionner **Add or create simulations sources**
3. **Next >**
4. Dans la fenêtre Add or Create Simulation Sources, sélectionner **Create Files**
5. Donner un nom au fichier => convention d'écriture : tb_NomdeProjet, dans notre cas tb_project_1
6. **OK**
7. **Finish**
8. **OK** (on ne complète pas le bloc Define Module)
9. **Yes**

On a maintenant le fichier qui est créé dans la fenêtre Sources



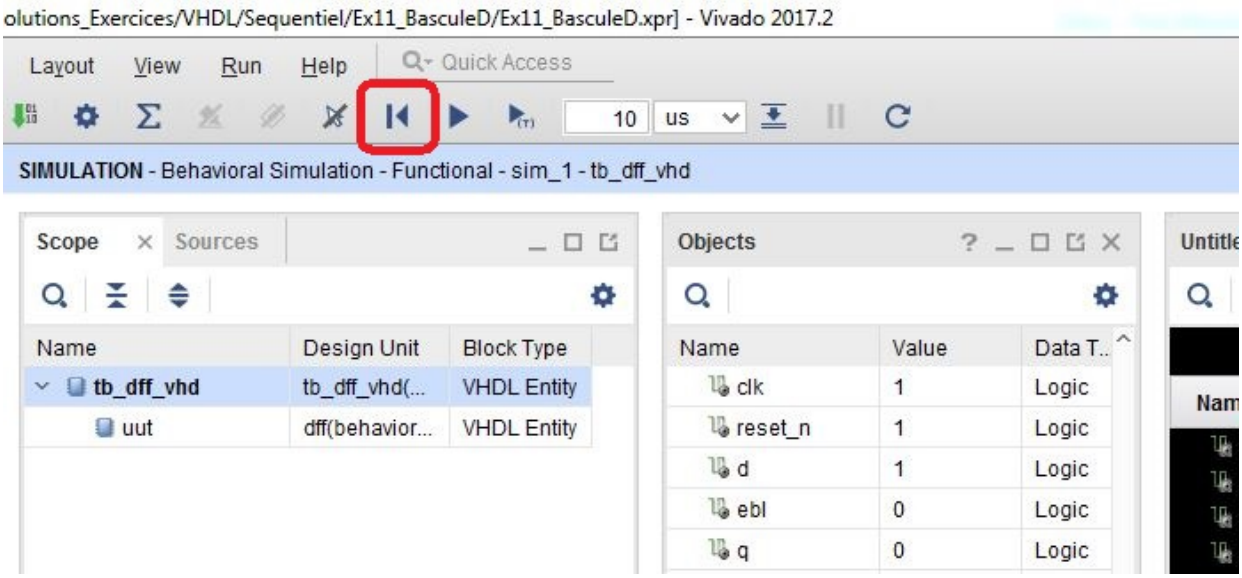
Il suffit d'effectuer un double-clic pour ouvrir le fichier banc de test dans la fenêtre d'édition et le compléter selon la méthode étudiée en cours. Une fois le banc de test complété et sans erreur de syntaxe, on peut lancer une simulation en exécutant **Run Simulation** dans la fenêtre Project Manager



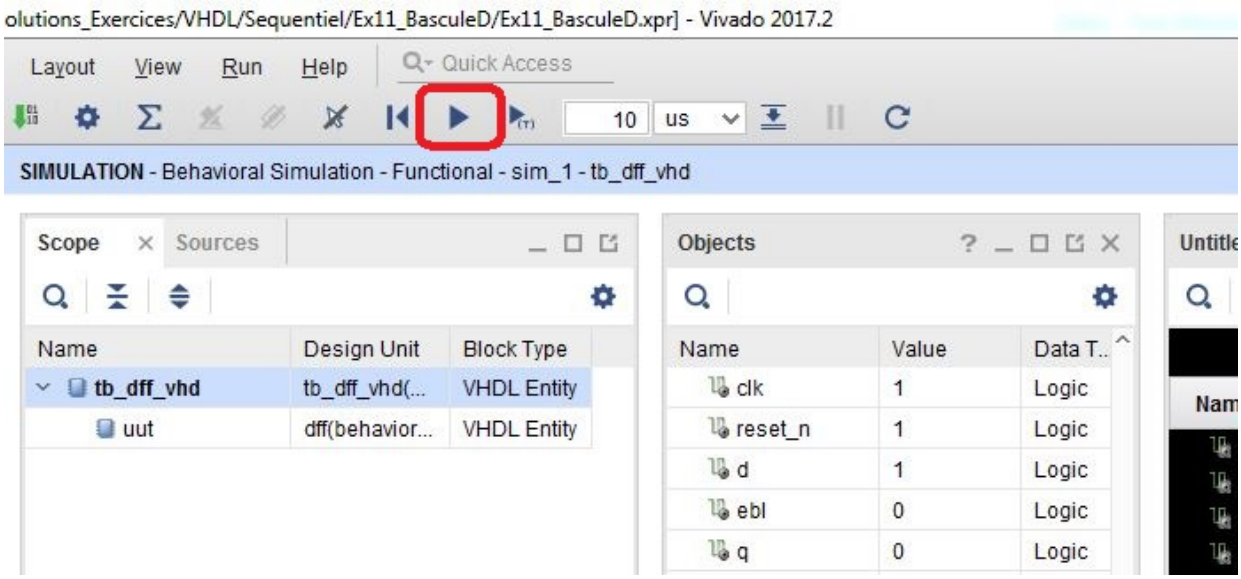
Normalement on effectue une simulation fonctionelle sans tenir compte des temps de propagation dans le circuit logique programmable, donc choisir **Run Behavioral Simulation**

La simulation démare et les fenêtres utiles à la simulation s'ouvrent (Scope, Objects, Wave). Par défaut lors du lancement, la simulation dure 1uSec, ce qui n'est pas forcément la durée de notre simulation. Il faut donc la relancer pour l'exécuter jusqu'à `sim_end = TRUE` en cliquant sur :

Restart (Ctrl+Maj+F5)



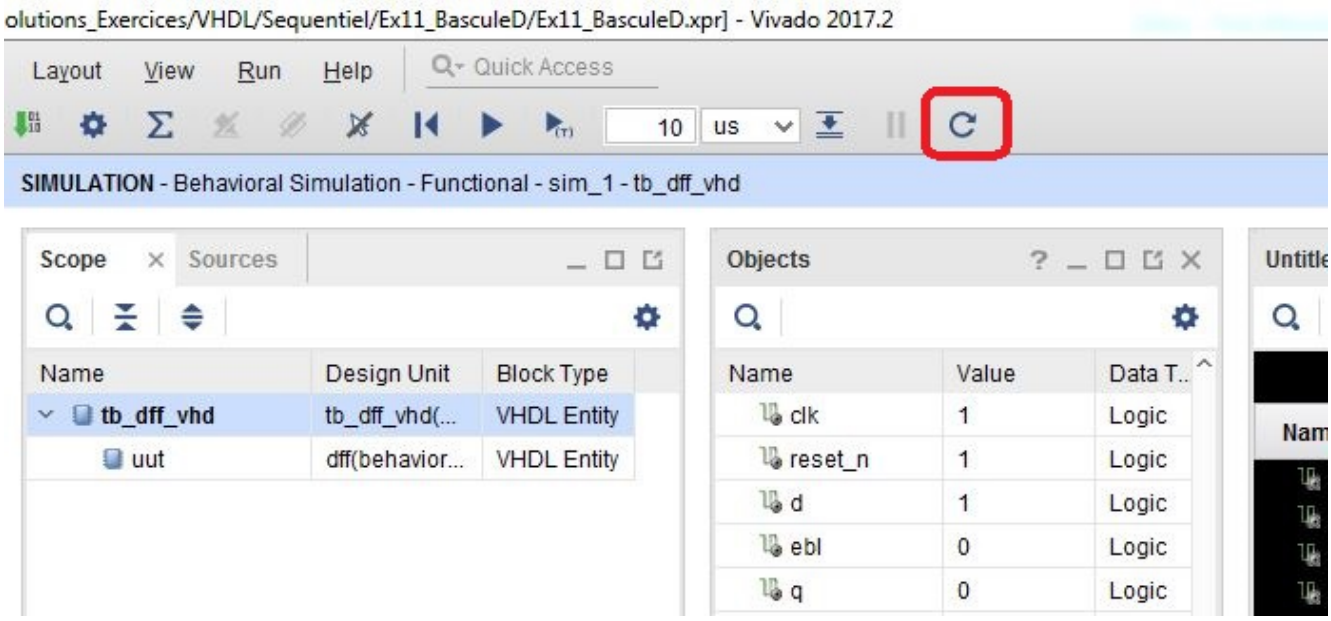
Puis **Run All** (F3)



Pour voir toute la simulation dans la fenêtre wave vous avez deux possibilités :

1. Après un clic-droit sur la fenêtre Wave, sélectioner **Full View**
2. Dans le menu View choisir **Zoom Fit**

Si la simulation donne des erreurs (mark_error, error_number), il faut corriger dans le fichier synthétisable à tester (UUT) ou dans le banc de test, puis relancer la simulation avec **Relaunch**



Il faut effectuer un Relaunch à chaque changement d'un des fichiers VHDL du projet.

On peut visualiser n'importe quel signal de la hiérarchie (par exemple dans le cas ou l'on a plusieurs fichiers avec du VHDL structurel), en sélectionnant le composant (fichier) dans la fenêtre **Scopes**, puis en faisant click droit **Add To Wave Windows** sur le signal à ajouter de la fenêtre **Objects** dans la simulation Wave.

4. Assignation des entrées/sorties de la FPGA du Kit Xilinx 7

Pin	Signal	Description
J19	100MHZ	Clock input 100MHz
AB3	ADC_A0	ADC MAX113 A0
AA3	ADC_A1	ADC MAX113 A1
W2	ADC_CS_N	ADC MAX113 CSn
AB5	ADC_D0	ADC MAX113 D0
AA6	ADC_D1	ADC MAX113 D1
AB6	ADC_D2	ADC MAX113 D2
AB7	ADC_D3	ADC MAX113 D3
Y2	ADC_D4	ADC MAX113 D4
AA1	ADC_D5	ADC MAX113 D5
AB1	ADC_D6	ADC MAX113 D6
AB2	ADC_D7	ADC MAX113 D7
AB8	ADC_INT_N	ADC MAX113 INTn
AA5	ADC_MODE	ADC MAX113 MODE
AA8	ADC_RD_N	ADC MAX113 RDn
Y1	ADC_WR_N/RDY	ADC MAX113 WRn/RDY
Y9	ACD2_MISO	ADC ADCS7476 MISO
Y7	ADC2_CSN	ADC ADCS7476 CSn
V9	ADC2_SCK/DAC_SCK	ADC ADCS7476 SCK (DAC AD5323 SCK)
G13	AFF1_0	Bargraphe 1 led 0
H14	AFF1_1	Bargraphe 1 led 1
G15	AFF1_2	Bargraphe 1 led 2
G16	AFF1_3	Bargraphe 1 led 3
H17	AFF1_4	Bargraphe 1 led 4
J16	AFF1_5	Bargraphe 1 led 5
K17	AFF1_6	Bargraphe 1 led 6
L20	AFF1_7	Bargraphe 1 led 7
L21	AFF1_8	Bargraphe 1 led 8
L19	AFF1_9	Bargraphe 1 led 9
L18	AFF2_0	Bargraphe 2 led 0
L16	AFF2_1	Bargraphe 2 led 1
L15	AFF2_2	Bargraphe 2 led 2
M20	AFF2_3	Bargraphe 2 led 3
N19	AFF2_4	Bargraphe 2 led 4
N20	AFF2_5	Bargraphe 2 led 5
J15	AFF2_6	Bargraphe 2 led 6
K16	AFF2_7	Bargraphe 2 led 7
L13	AFF2_8	Bargraphe 2 led 8
L14	AFF2_9	Bargraphe 2 led 9
AA20	AFF1_ENABLE	Affichage 7 segment 1 enable
AB20	AFF2_ENABLE	Affichage 7 segment 2 enable

P20	AFF3_ENABLE	Affichage 7 segment 3 enable
P19	AFF4_ENABLE	Affichage 7 segment 4 enable
Y4	AUDIO_OUT_LRCLK	Audio DAC CS4344 LRCLK
AA4	AUDIO_OUT_MCLK	Audio DAC CS4344 MCLK
W6	AUDIO_OUT_SCLK	Audio DAC CS4344 SCLK
Y6	AUDIO_OUT_SDIN	Audio DAC CS4344 SDIN
A16	BUZZER	Buzzer
K19	CLKDIV	Clock input Divided from CPLD
W7	DAC_CLR_N	DAC AD5323 CLRn
Y8	DAC_LDAC_N	DAC AD5323 LDACn
W9	DAC_MOSI	DAC AD5323 DIN
V9	DAC_SCK/ADC2_SCK	DAC AD5323 SCK (ADC ADCS7476 SCK)
V8	DAC_SYNC_N	DAC AD5323 SYNCn
G20	DILSW1_0	Dilswitch 1 bit 0
H22	DILSW1_1	Dilswitch 1 bit 1
J21	DILSW1_2	Dilswitch 1 bit 2
J22	DILSW1_3	Dilswitch 1 bit 3
K21	DILSW1_4	Dilswitch 1 bit 4
K22	DILSW1_5	Dilswitch 1 bit 5
M21	DILSW1_6	Dilswitch 1 bit 6
M22	DILSW1_7	Dilswitch 1 bit 7
N22	DILSW2_0	Dilswitch 2 bit 0
G17	DILSW2_1	Dilswitch 2 bit 1
G18	DILSW2_2	Dilswitch 2 bit 2
H18	DILSW2_3	Dilswitch 2 bit 3
H20	DILSW2_4	Dilswitch 2 bit 4
J17	DILSW2_5	Dilswitch 2 bit 5
J20	DILSW2_6	Dilswitch 2 bit 6
H15	DILSW2_7	Dilswitch 2 bit 7
C13	LCD_D0	LCD ECC1602B-NSWBBW-91LE D0
C14	LCD_D1	LCD ECC1602B-NSWBBW-91LE D1
B15	LCD_D2	LCD ECC1602B-NSWBBW-91LE D2
C15	LCD_D3	LCD ECC1602B-NSWBBW-91LE D3
D16	LCD_D4	LCD ECC1602B-NSWBBW-91LE D4
D17	LCD_D5	LCD ECC1602B-NSWBBW-91LE D5
D19	LCD_D6	LCD ECC1602B-NSWBBW-91LE D6
D20	LCD_D7	LCD ECC1602B-NSWBBW-91LE D7
F13	LCD_LIGHT	LCD ECC1602B-NSWBBW-91LE Backlight
B13	LCD_E	LCD ECC1602B-NSWBBW-91LE E
F14	LCD_RS	LCD ECC1602B-NSWBBW-91LE RS
E13	LCD_RW	LCD ECC1602B-NSWBBW-91LE R/W
A18	LED_R1	Led bicolore 1 rouge
B20	LED_R2	Led bicolore 2 rouge
A20	LED_R3	Led bicolore 3 rouge
B21	LED_R4	Led bicolore 4 rouge

B16	LED_V1	Led bicolore 1 verte
B18	LED_V2	Led bicolore 2 verte
A19	LED_V3	Led bicolore 3 verte
A21	LED_V4	Led bicolore 4 verte
C17	NetJ13_2	Connecteur J13 pin 2 (pull-up 1k)
C18	NetJ13_3	Connecteur J13 pin 3
C19	NetJ13_4	Connecteur J13 pin 4
C20	NetJ13_5	Connecteur J13 pin 5
L10	XADC VP_0	XADC VP_0 (J24 pin 2)
M9	XADC VN_0	XADC VP_0 (J24 pin 3)
E1	PMOD1_D0F_P	PMOD1 J10 pin 1 (diff)
D1	PMOD1_D0F_N	PMOD1 J10 pin 2 (diff)
B1	PMOD1_D1F_P	PMOD1 J10 pin 3 (diff)
A1	PMOD1_D1F_N	PMOD1 J10 pin 4 (diff)
G1	PMOD1_D2F_P	PMOD1 J10 pin 7 (diff)
F1	PMOD1_D2F_N	PMOD1 J10 pin 8 (diff)
K1	PMOD1_D3F_P	PMOD1 J10 pin 9 (diff)
J1	PMOD1_D3F_N	PMOD1 J10 pin 10 (diff)
H2	PMOD2_D0_P	PMOD2 J11 pin 1 (diff)
G2	PMOD2_D0_N	PMOD2 J11 pin 2 (diff)
E2	PMOD2_D1_P	PMOD2 J11 pin 3 (diff)
D2	PMOD2_D1_N	PMOD2 J11 pin 4 (diff)
K2	PMOD2_D2_P	PMOD2 J11 pin 7 (diff)
J2	PMOD2_D2_N	PMOD2 J11 pin 8 (diff)
C2	PMOD2_D3_P	PMOD2 J11 pin 9 (diff)
B2	PMOD2_D3_N	PMOD2 J11 pin 10 (diff)
P5	PMOD3_1	PMOD3 J14 pin 1
N5	PMOD3_2	PMOD3 J14 pin 2
L6	PMOD3_3	PMOD3 J14 pin 3
L5	PMOD3_4	PMOD3 J14 pin 4
G3	PMOD3_7	PMOD3 J14 pin 7
H3	PMOD3_8	PMOD3 J14 pin 8
J4	PMOD3_9	PMOD3 J14 pin 9
K3	PMOD3_10	PMOD3 J14 pin 10
N4	PMOD4_1	PMOD4 J18 pin 1
N3	PMOD4_2	PMOD4 J18 pin 2
M5	PMOD4_3	PMOD4 J18 pin 3
M3	PMOD4_4	PMOD4 J18 pin 4
P6	PMOD4_7	PMOD4 J18 pin 7
L3	PMOD4_8	PMOD4 J18 pin 8
L4	PMOD4_9	PMOD4 J18 pin 9
M6	PMOD4_10	PMOD4 J18 pin 10
R1	PMOD5_1	PMOD5 J19 pin 1
P2	PMOD5_2	PMOD5 J19 pin 2
P1	PMOD5_3	PMOD5 J19 pin 3

N2	PMOD5_4	PMOD5 J19 pin 4
P4	PMOD5_7	PMOD5 J19 pin 7
L1	PMOD5_8	PMOD5 J19 pin 8
M2	PMOD5_9	PMOD5 J19 pin 9
M1	PMOD5_10	PMOD5 J19 pin 10
T18	POUSSOIR_RESET_n	Poussoir reset (actif bas)
U21	POUSSOIR0	Poussoir 0
V20	POUSSOIR1	Poussoir 1
W20	POUSSOIR2	Poussoir 2
U20	POUSSOIR3	Poussoir 3
V19	PWM	PWM pour Servo sur J23
N13	PWM_I_LED	PWM IS31LT3360 (I LED Adjust)
R6	RTC_CLKOUT	RTC 8563 CLKOUT
U7	RTC_INTN	RTC 8563 INTn
W17	RXD_USB_TX_UART	CP2102 Serial to USB RXD
V17	TXD_USB_RX_UART	CP2102 Serial to USB TXD
AA18	RXD_USB_TX_UART_2	FT232BQ Serial to USB RXD
Y18	TXD_USB_RX_UART_2	FT232BQ Serial to USB TXD
U6	SCL	I2C SCL
T6	SDA	I2C SDA
V22	SEG_A	Affichage 7 segments A
W22	SEG_B	Affichage 7 segments B
W21	SEG_C	Affichage 7 segments C
Y22	SEG_D	Affichage 7 segments D
Y21	SEG_E	Affichage 7 segments E
AB22	SEG_F	Affichage 7 segments F
AA21	SEG_G	Affichage 7 segments G
AB21	DP	Affichage 7 segments point