UNIVERSITÉ DU LITTORAL CÔTE D'OPALE

Architecture des ordinateurs - Logique 2019

Projet 0 - Prise en main de Xilinx ISE Design Suite

Auteur : E. Poisson Caillault

---

1 Objectifs des séances TP

L'objectif des séances TPs au travers de la réalisation de différents projets est d'apprendre :

- à manipuler un logiciel de flot de conception, simulation et implémentation sur carte d'un circuit numérique à base de portes via l'outil Vivado de Xilinx.
- à écrire et manipuler les structures de base du VHDL : entité/architecture, processus concurrentiel/séquentiel, signal/variable, types standard (integer, bit, std_logic, std_logic_vector, time).
- bien comprendre les éléments de logique classique : portes élémentaires.

Organisation et évaluation

Un ensemble de projets ordonnés est proposé. Vous avancerez à votre rythme, la durée associée à chaque projet est donnée pour information de manière approximative. Avant la cloture de chaque séance de TP, vous devrez faire un état d'avancement par écrit que vous soumettrez sur Sakai. Vous insérerez les réponses demandées, images des chronogrammes simulées, table de vérité ... dans ce rapport d'avancement que vous nommerez de la manière suivante : NOM1.NOM2.Rapport.ProjetX.odt, X étant le nom du projet. Ce rapport vous servira de support pour les évaluations TPs qui auront lieu en début de séance.

Pour chaque projet, vous devrez aussi présenter votre réalisation en séance à l'enseignant qui validera les compétences acquises. La notation tiendra compte de votre avancement et des compétences acquises pour chaque projet.

L'ensemble des documents nécessaires aux séances TP sont sur la plateforme Sakai ainsi que l'ensemble des cours, TDs, examen de l'an passé et autres documents utiles à la compréhension de ce module.

Compétences

A l'issue des séances, vous aurez développé les compétences suivantes de l'Informatique :

- Lecture d'un cahier des charges, identification des entrées-sorties.
- Mise en oeuvre de la validation d'un programme informatique en langage VHDL avec notamment l'écriture d'un scénario de test unitaire.

Il conviendra d'ajouter sur votre CV (*Curriculum Vitae)* les compétences informatiques suivantes dans la rubrique Language VHDL et dans Outils : VIVADO Xilinx.

1.1 Description succinte de la carte BASYS 3

Pour l'ensemble des TPs, nous utiliserons la carte Basys 3. Cette carte, représentée à la figure 1, est dotée d'un ensemble d'entrées (boutons-poussoirs, commutateurs, connecteurs, port USB pour des périphériques souris/claviers) et sorties (leds, afficheur 4 chiffres à 7 segments connecteurs, sortie VGA 12 bits, connecteurs). Au centre de la carte, elle possède son entité de calcul via un FPGA comportant

: 33 280 cellules logiques (dans 5200 tranches, chaque tranche contient quatre LUT à 6 entrées et 8 bascules), 90 tranches DSP permettant des opérations plus complexes, une mémoire RAM fast-block associée de 1 800 Kbits ainsi qu'une mémoire Flash série.

Nous utiliserons dans ces projets L1 une partie limitée des possibilités offertes par cette carte, à savoir :

- Flèche 15 sur la figure, le commutateur d'alimentation POWER SW16, qu'il faudra activer!
- Flèche 16, le pont JP2 pour spécifier le type d'alimentation externe, le jumper doit être en mode USB dans notre cas, si le jumper couvre EXT le déplacer pour obtenir USB;
- Flèche 5 : 16 commutateurs utilisateur, SWO à SW15;
- Flèche 6 : 16 leds utilisateur LD0 à LD15, d'autres leds seront activées automatiquement lors de l'utilisation de la carte, telle la led LD20 (Flèche 1) précisant l'alimentation active de la carte;
- Flèche 7 : 5 boutons-poussoirs utilisateur;
- Flèche 13 : le Port USB-JTAG Digilent pour la programmation FPGA avec le pont associé JP1 (Flèche 10) à vérifier (jumper en position haute);
- et pour faire des opérations son FPGA.

Dans l'espace TP sur Sakai, vous trouverez son manuel, ses schémas et un fichier d'extension xdc qui nous servira de base de l'assignation des entrées/sorties de la carte aux variables/signaux de vos programmes VHDL que vous reprendrez vierge pour chaque projet.

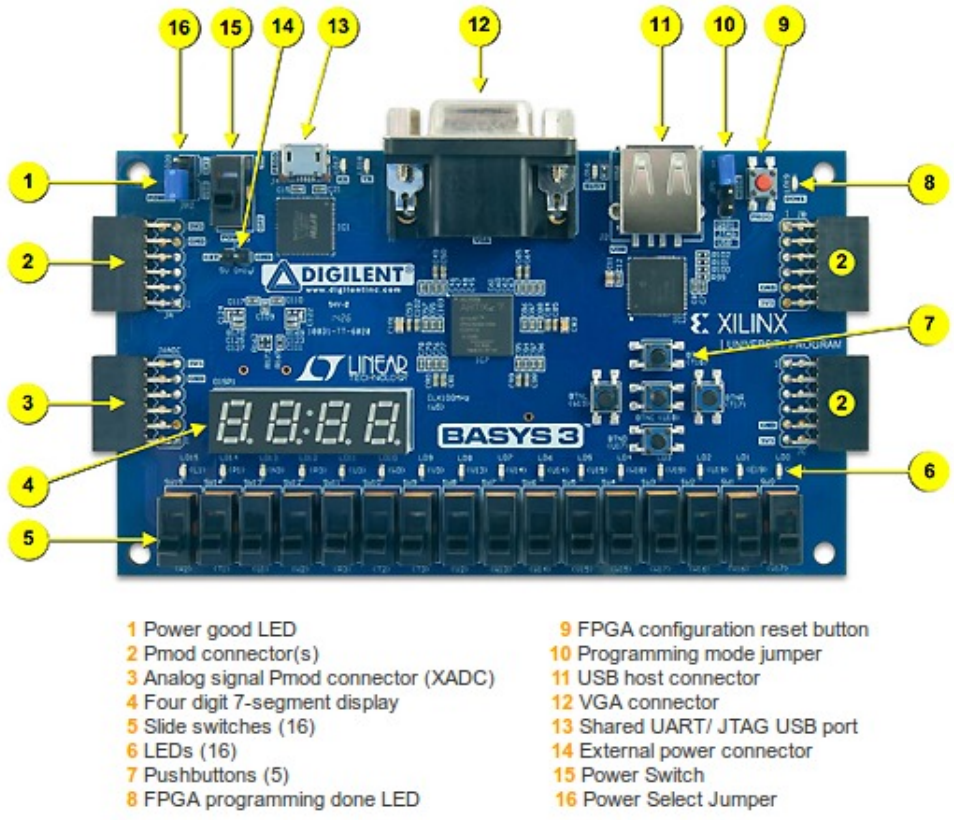

Figure 1: Image de la Carte Basys-3. DIGILENT

1.2 Description de l'outil XILINX

L'outil Vivado Design Suite - HLx Editions permet de simuler une description d'un circuit électronique et transcrire la description donnée dans un langage exécutable ou interprétable pour le processeur ou carte cible.
Il offre notamment différents types de saisie (saisie de portes/schematic, Programmes VHDL ou Verilog, ...). Dans la version Webpack, seule la saisie par programmes VHDL ou verilog est possible ou importation de tout fichier déjà compilé tels des schémas interprétés.
L'outil permet de faire une simulation fonctionnelle c'est-à-dire sans prise en compte des contraintes temporelles liées à la solution physique et ensuite de faire une simulation réelle prenant en compte l'aspect séquentiel de la cible (routage et délai de propagation dans les portes).

Cette plateforme intègre :

- pour la spécification du circuit : différents éditeurs textuels (VHDL et Verilog), graphiques, tabulaires, machine d'états ... accessibles depuis le « gestionnaire-navigateur » de projet **Flow Navigator**.
- Pour les simulations fonctionnelles, post-synthèses, post-routage : un simulateur VHDL/Verilog **Xilinx xSim**
- Pour la synthèse logique : le synthétiseur XST – Xilinx Synthesis Technology.
- L'implémentation sur la carte FPGA peut être faite via une interface **Diligent ExPort** ou directement dans Xilinx (depuis la version 2015) qui permet de programmer le composant via un bus nommé JTAG.

Tous les logiciels utilisés sont téléchargeables sur le net, fournis gratuitement par Xilinx. Il vous suffira de créer un compte xilinx en tant qu'étudiant et de l'activer via le mail envoyé lors de votre inscription (sous 24 à 48h), puis de télécharger vivado design suite sur le site xilinx (préférer la version légère web en spécifiant l'outil webpack).

2 Premier Projet : initiation. Durée estimée entre 2 et 4h max.

Pour ce premier projet, l'ensemble des sources vous seront fournis et commentés. A vous de prendre soin de les lire, écrire attentivement et suivre pas à pas le process complet de conception d'un programme et d'implémentation sur carte afin de savoir le refaire de manière autonome lors des prochains projets.
Le cahier des charges de ce TP est le suivant :

> Vous devez implémenter sur la carte BASYS 3 le pilotage de l'affichage d'une donnée d'entrée 1 bit que l'utilisateur aura initialisée selon l'état du commutateur/switch SW0. Les segments de l'afficheur permettront de lire le chiffre zéro lorsque le commutateur est en position basse, et le chiffre UN en position haute.

Figure 2: Représentation des segments de l'afficheur

Le schéma de l'afficheur est donné à la figure suivante **??**.

2.1 Étapes nécessaires à la bonne réalisation d'un projet

Afin de bien mener à terme la conception d'une application sur carte, il est nécessaire de passer par l'ensemble des étapes suivantes de manière chronologique. Si une étape n'est pas réussie, les suivantes n'aboutiront pas.

1. La première étape est essentielle à tout cahier des charges, savoir comprendre les spécifications du problème posé, de l'application à concevoir. Cela signifie être capable premièrement d'identifier et définir l'ensemble des entrées-sorties du système et ensuite, d'identifier les fonctionnalités principales du système/programme à concevoir. Si celles-ci sont élémentaires c'est-à-dire qu'elles peuvent être traduites automatiquement par une équation, une table de vérité, un schéma électrique, un graphe d'état, un logigramme ou un algorithme alors la conception fonctionnelle s'achève là. Dans le cas contraire, elles seront décrites par des sous-fonctions/sous-modules élémentaires.

2. Une fois la **décomposition fonctionnelle** terminée et élémentaire (facile à implémenter), il sera aisé de **programmer** chaque module élémentaire dit aussi **entité** en VHDL et d'assurer que le code écrit est valide (compilable).

3. L'ensemble des modules élémentaires devront être testés unitairement, pour cela il est nécessaire d'écrire un ou plusieurs scénarii de test en VHDL, appelé aussi **testbench** - banc d'essai. Ces scenarii reprennent toutes les combinaisons possibles des entrées du module. [Dans de grands projets, cela est extrêmement difficile et ne seront testés que le cas de marche et tous les scenarii de panne importante ou connue.]
   Vous devrez donc écrire la table de vérité de chacun des modules à programmer et vérifier via le simulateur xSim si les chronogrammes obtenus pour chaque module sont ceux désirés.

4. Ensuite il faudra faire de même pour chaque module englobant puis le module général de votre application toujours en simulation fonctionnelle, indépendamment de la technologie utilisée. C'est dans le jargon informaticien, des tests d'intégration.

5. L'étape suivante est de vérifier l'adéquation de votre solution avec l'architecture matérielle choisie, appelée **synthèse logique** qui prend en compte le nombre d'entrées-sorties de votre cible, la mémoire de celle-ci, les fonctions/portes disponibles.

6. Lorsque la solution proposée est compatible avec la cible choisie, il faut spécifier la correspondance entre les variables de votre programme et les entrées-sorties de la carte en écrivant **un fichier de contraintes** avec une extension xdc. Ensuite il est possible de laisser l'outil d'implémentation de placer (assignation des cellules) et router (assignation des bus) votre solution à travers les cellules du FPGA de manière automatique ou de configurer spécifiquement le placement-routage. Nous utiliserons que le placement routage automatique.

7. Une fois le placement-routage spécifié, votre solution sera traduite en langage cible via l'écriture automatique du **fichier bitstream** qui reprend votre description VHDL des modules et des contraintes d'E/S, de placement et de routage.

8. Ce fichier sera ensuite envoyé sur la carte. Et vous pourrez tester de visu votre application : cette étape s'appelle le test sur carte.

2.2 Mise en oeuvre du process sur le projet d'initiation

Nous considérons que l'étape 1, la décomposition fonctionnelle, a été achevée en cours-TD
Nous allons donc écrire le module de pilotage de l'afficheur 7 segments de la valeur d'un bit: chiffre zéro ou un en langage VHDL. Pour cela nous devons créer un projet, nous permettant de coder, tester puis implémenter notre module sur la carte FPGA.

Lancement de l'outil Vivado

Sous Windows :

- Dans la liste des programmes accessibles, recherchez l'outil Vivado de Xilinx

Sous Linux :

1. Aller dans le répertoire où est installé Xilinx, en tapant dans une console : cd "/DossierXilinx/bin/lin/" si le lien n'a pas été enregistré. Vous aurez besoin à la première installation sur votre machine personnelle de sourcer le fichier settings.

2. Lancer l'application, en tapant dans la même console : vivado & (Si vous ne mettez pas &, l'application se fermera si vous quittez la console même si le travail n'est pas terminé/enregistré). Depuis la version 2015, vous devrez lancer le script de configuration des drivers du câble USB : aller dans le répertoire data puis xicom pour lancer le script en mode root (sudo).

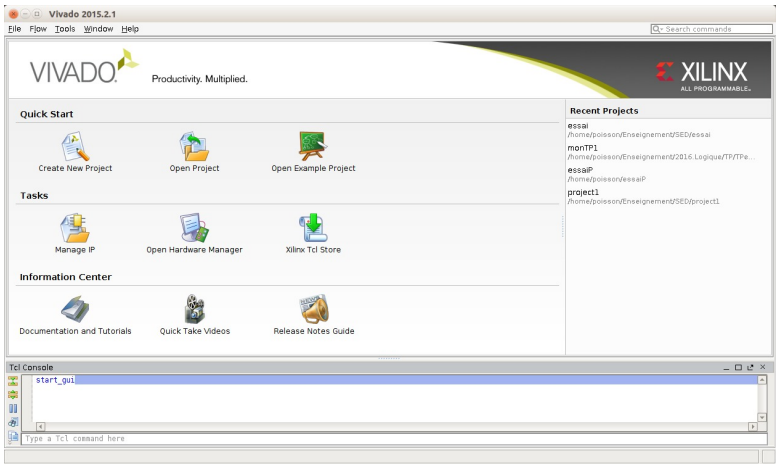

Figure 3: Copie écran - Ouverture du logiciel Vivado de Xilinx

La fenêtre d'ouverture du logiciel est découpée en trois grandes parties comme ci-après Figure 3:

- Dans la partie gauche centrale, elle-même divisée en trois parties :
    1. Quick start : correspondante à la création d'un projet ou l'ouverture de projets existants ou d'exemples.
    2. Task : management (import/export) de librairies, modules existants, management de cartes matérielles (ajout de librairies dédiées pour la synthèse).
    3. Information Center : documentation, aide relative à l'outil.
- A droite, une fenêtre reprenant l'ensemble des projets créés si existants.
- En bas, une console permettant de lancer en ligne de commande l'ouverture, la création d'un projet par exemple.

Création du projet initiation

1. Cliquez sur le bouton créer un projet, "Create new project". S'ouvre alors une fenêtre de dialogue, cliquez sur Next. Apparaît telle la figure 4 la demande de saisie d'information sur le projet.

2. Si ce n'est pas déjà fait créer un répertoire Vivado sur votre compte personnel. Commencez par sélectionner un répertoire de travail, bouton droit "..." à gauche de la ligne Project Location.

3. Puis remplissez le champ Project Name : TP1

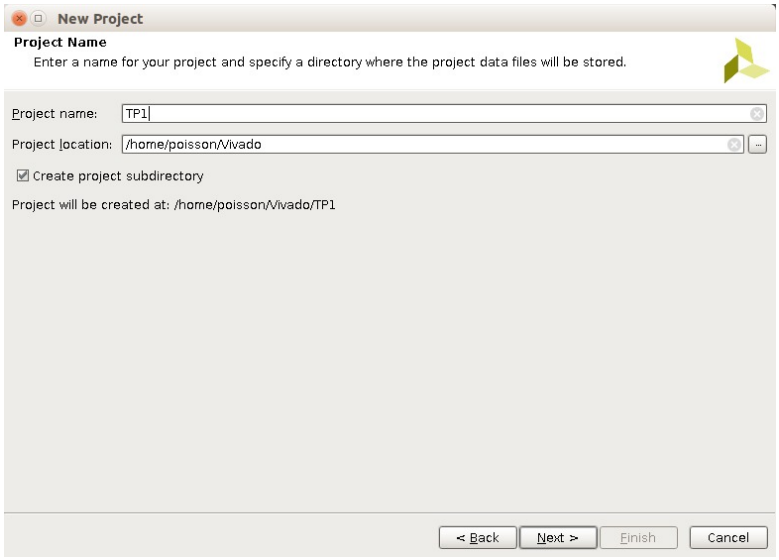

Figure 4: Copie écran - Création du projet, nom et répertoire de travail

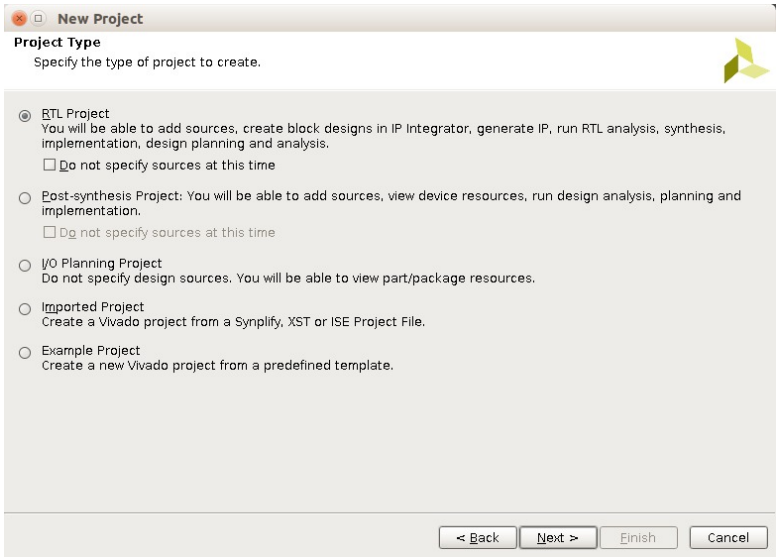

Figure 5: Copie écran - Création du projet, type de projet

4. Puis cliquez sur Next.
5. une fenêtre de configuration du type de projet apparaît (figure 5). Laissez la sélection par défaut, soit RTL project et cliquez sur next.
6. Une fenêtre permettant l'insertion et la création de sources est proposée. Nous créerons plus tard les fichiers nécessaires. Assurez-vous que le langage associé à la cible est VHDL ainsi que pour le simulateur. Vous devez avoir une fenêtre telle la figure 6. Cliquez ensuite sur Next.
7. Une fenêtre d'insertion de module IP existant est proposée. Cliquez sur Next.
8. Une fenêtre d'insertion du fichier des contraintes est demandée. Nous passerons cette étape pour le moment. Cliquez sur next.
9. Une fenêtre de paramétrage, Default part, apparaît pour choisir votre carte cible.
10. Sélectionner le champ Artix-7 dans Family
11. Sélectionner le champ cpg236 dans Package

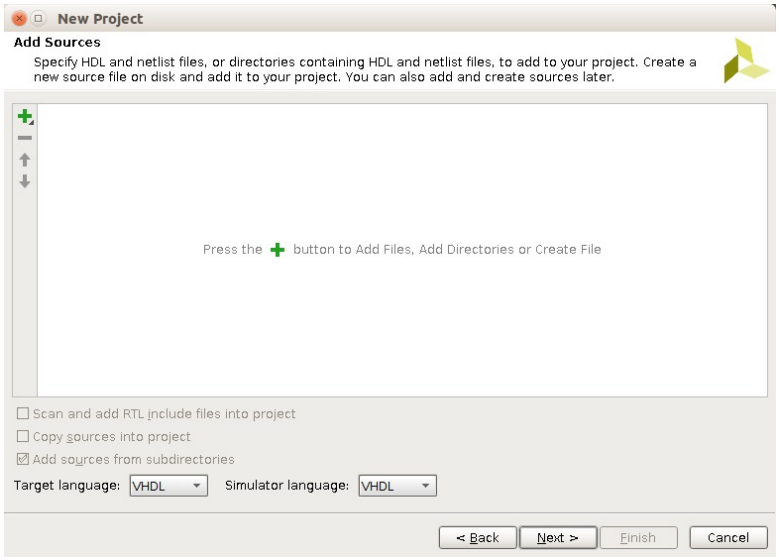

Figure 6: Copie écran - Création du projet, fichiers intégrés au projet

12. Sélectionner le champ -1 dans Speed grade

13. Apparaît alors plus que trois cartes possibles, sélectionner la xc7a35t. Vous devez voir apparaître cette figure 7 puis cliquez sur Next.

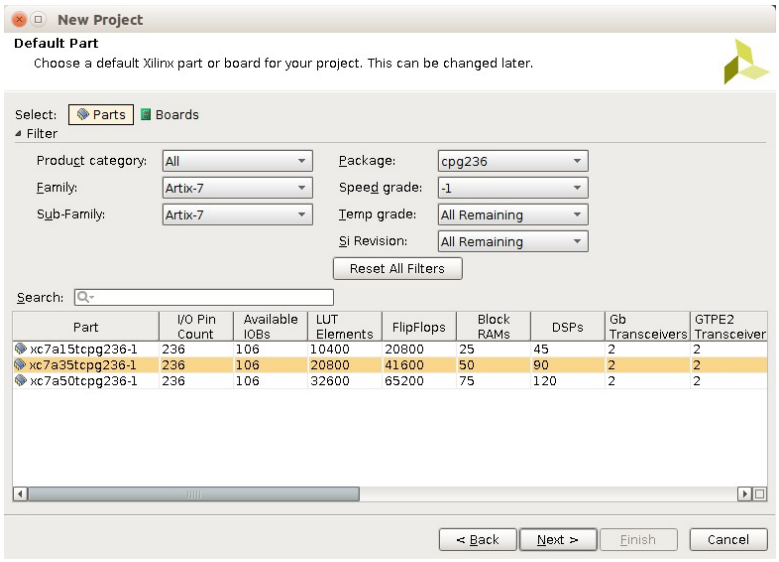

Figure 7: Copie écran - Création du projet, choix de la cible

14. Une fenêtre récapitule la configuration de votre projet, modifiable par la suite. Cliquez sur Finish.

Si par mégarde vous fermez vivado, après avoir passé cette étape, ne recréer pas un projet, ouvrez votre projet par la fenêtre droite de la figure 3.

Création du module AfficheurBinaire et compilation

L'écran principal fait alors apparaître plusieurs sous-parties comme ci-après Figure 8. La partie droite reprend tous les process nécessaires à l'élaboration de votre solution dans le flow navigator. La partie centrale correspond à l'état du projet, elle permet aussi d'éditer les sources ainsi que l'affichage de résultats (par exemple le résultat d'une simulation). La partie basse reprend l'ensemble des messages liés aux process de conception : erreur de compilation, log d'exécution, ...

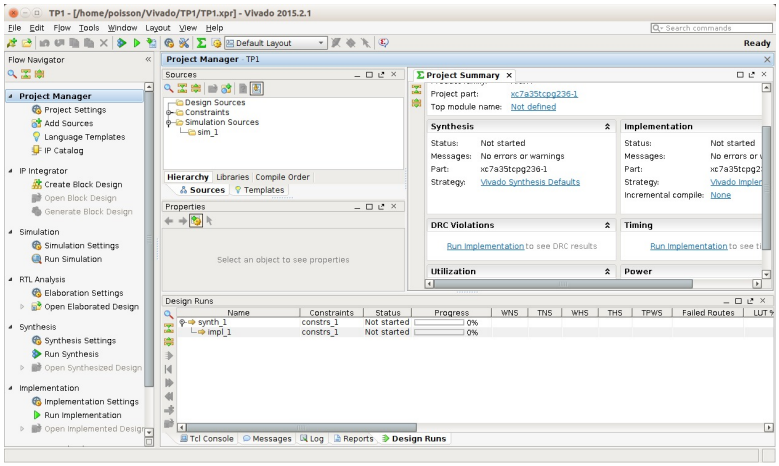

Figure 8: Copie écran - Fenêtre Principale du projet, Gestion des étapes de conception

1. Dans la partie gauche - Flow Navigator, cliquez sur Add sources.

2. Nous souhaitons programmer ce module par une description VHDL, choisissez l'option "add or create design sources".

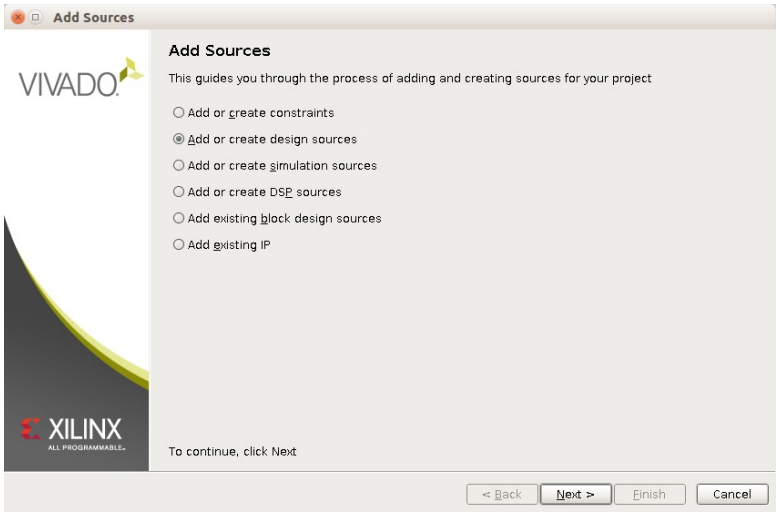

Figure 9: Copie écran - Fenêtre Principale du projet, Gestion des étapes de conception

3. Pour créer ou sélectionner un fichier, il faut cliquer sur la croix puis spécifier ce que l'on souhaite. Ici nous allons créer le fichier donc sélectionner "Create File".

4. Renseigner le nom du Module soit AfficheurBinaire et vérifier le langage associé VHDL puis cliquez sur OK puis sur Finish.

5. Une fenêtre vous demande de renseigner votre module, notamment le nom de son entité, de son architecture et les entrées-sorties de votre entité (type in/out/inout, taille bit ou mot/bus, et pour ce dernier son indexation). Compléter la fenêtre pour aboutir à la figure 11. Dans votre Project Manager, fenêtre principale, vous voyez votre fichier inséré dans les sources. Pour l'éditer, double-cliquez dessus. L'entité a été automatiquement générée selon vos spécifications. Reste alors à compléter son architecture.
   Ouvrir l'éditeur en double cliquant dans Project Manager sur le fichier design créé AfficheurBinaire. Puis recopier le fichier "AfficheurBinaire.vhd", le code vous est fourni.

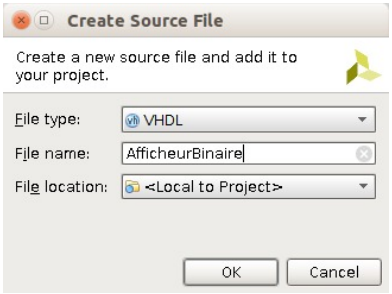

Figure 10: Copie écran - Création Fichier sources VDHL, spécification du nom et langage

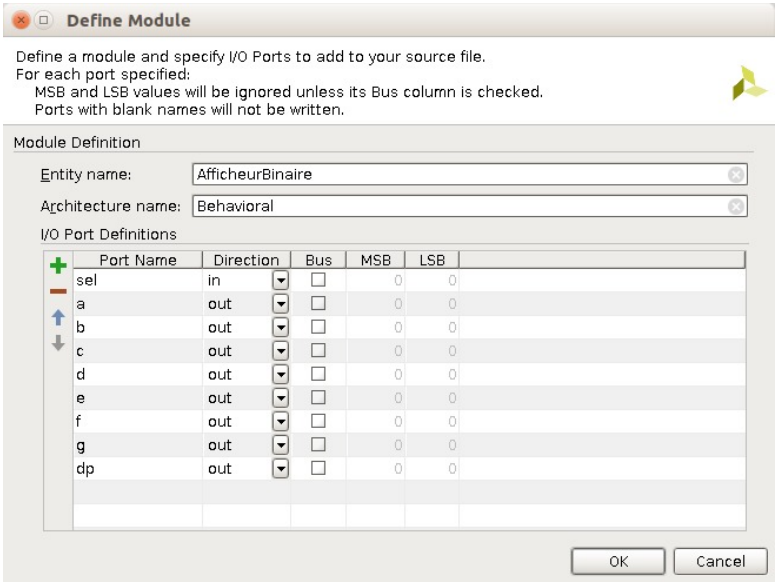

Figure 11: Copie écran - Création Fichier sources VDHL, spécification des E/S

```
1  ----------------------------------------------------------------------------------
2  -- Company: ULCO
3  -- Author/ teacher: E. Poisson Caillault
4  -- Create Date: 16.02.2016 00:15:00
5  -- Module Name: AfficheurBinaire - Behavioral
6  -- Description: Pilote un afficheur 7 segments.
7  -- Apparait le chiffre zero ou un selon la valeur binaire de l'entrée sel.
8  ----------------------------------------------------------------------------------
9
10 --description des codes couleurs
11 -- en violet ou vert, les mots réservés en langage VHDL
12 -- en marron-violet les types réservés en langage VHDL
13 -- rappel en VHDL la casse n'est pas respectée
14 -- attention indifférence minuscule/majuscule!
15
16 -- insertion des bibliothèques utilisées
17 library IEEE;
18 use IEEE.STD_LOGIC_1164.ALL;
19
20 -- declaration de l'interface du module : entité
21 -- definition des E/S
22 entity AfficheurBinaire is
23     Port ( sel : in STD_LOGIC;
24            a : out STD_LOGIC;
25            b : out STD_LOGIC;
26            c : out STD_LOGIC;
27            d : out STD_LOGIC;
28            e : out STD_LOGIC;
29            f : out STD_LOGIC;
30            g : out STD_LOGIC;
31            dp : out STD_LOGIC);
```

```vhdl
end AfficheurBinaire;

-- declaration d'un comportement associé à l'entité : architecture
architecture Behavioral of AfficheurBinaire is
begin
--plusieurs instructions concurentielles independantes
-- une instruction se termine toujours pas un point virgule
--when/else permet un assignement conditionnel d'un signal
--     remplace l'écriture if then conditionné à un process sequentiel.
    a<='1' when sel='1' else '0' when sel='0';
    b<='0';
    c<='0';
    d<='1' when sel='1' else '0' when sel='0';
    e<='1' when sel='1' else '0' when sel='0';
    f<='1' when sel='1' else '0' when sel='0';
    g<='1'; -- barre milieu
    dp<='1';-- le point
end Behavioral;
```

Code VHDL 1 : Module AfficheurBinaire

Vous noterez qu'il existe plusieurs transcriptions possibles VHDL pour éteindre les segments, ici une structure When-else est utilisée. Enregistrez votre solution à l'aide du bouton enregistrer (symbole Disquette) ou taper au clavier le raccourci CTRL-S, ainsi il ne doit plus rester d'astérisque accolé au nom de votre fichier.

En cas d'erreur de syntaxe, votre fichier sera inséré dans un sous-repertoire "syntax_error_files". Par exemple enlever le point virgule et sauvegarder. Dans la fenêtre de message en bas, vous pourrez lire les informations provenant du compilateur VHDL : le type d'erreur "EOF" avec le numéro de ligne correspondante (EOF=end of file). Lisez la console, puis corrigez et re-sauvegarder.
Note : Tant que votre fichier n'a pas été sauvegardé, le compilateur affiche l'erreur précédente.
Si tout est correct votre fichier s'affiche en gras (fichier principal du projet, équivalent au fichier avec un main en programmation C).
Votre fichier étant correct d'un point de vue syntaxique, il n'est peut-être pas juste vis-à-vis du comportement souhaité. Pour cela, nous allons passé au test unitaire.

Test unitaire du module AfficheurBinaire

Pour simuler notre module, il faut créer une autre source appelé **testbench** qui est un fichier vhdl permettant de simuler les états des entrées.

1. Comme nous avons déjà créé la source design nous allons retourner dans Project manager, et demander la création d'un nouveau fichier (add sources) cette fois de type simulation.

```vhdl
----------------------------------------------------------------------------------
-- Company: ULCO
-- Author/Teacher: E. Poisson Caillault
--
-- Create Date: 16.02.2016 12:11:36
-- Module Name: TestAfficheurBinaire - Behavioral
-- Description: testbench du module Afficheur Binaire
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Declaration de l'entité. Module de test aucune E/S
entity TestAfficheurBinaire is
--  Port ( );
end TestAfficheurBinaire;

architecture Behavioral of TestAfficheurBinaire is
    --declaration des variables utilisées et des sous-modules
    component AfficheurBinaire
        Port ( sel : in STD_LOGIC;
               a : out STD_LOGIC;
               b : out STD_LOGIC;
               c : out STD_LOGIC;
               d : out STD_LOGIC;
               e : out STD_LOGIC;
               f : out STD_LOGIC;
               g : out STD_LOGIC;
               dp : out STD_LOGIC);
      end component;

      signal sel: STD_LOGIC:='0';
      signal A: STD_LOGIC;
      signal B: STD_LOGIC;
      signal C: STD_LOGIC;
      signal D: STD_LOGIC;
      signal E: STD_LOGIC;
      signal F: STD_LOGIC;
      signal G: STD_LOGIC;
      signal dp : STD_Logic;

      -- declaration d'une constante de temps
      constant period:time :=10 ns;

begin
    -- process de generation de toute les cas possible de l'entrée.
    genereSel: process
    begin
        sel <='0';
        wait for period;
        sel <='1';
        wait for period;
        sel <='0';
        wait for period;
        sel <='1';
        wait; -- will wait for ever
   end process;

   -- process AfficheurBinaire
   Affiche : AfficheurBinaire PORT MAP(
       sel=>sel,
       A=>a,
       B=>b,
       C=>c,
       D=>d,
       E=>e,
       F=>f,
       G=>g,
       dp=>dp
   );
end Behavioral;
```

Code VHDL 2 : Testbench de AfficheurBinaire.

2. Ajouter un source VHDL Testbench, nommer le TestAfficheurBinaire. Laisser la définition des E/S vide (aucun utilisateur ne doit pouvoir influencer ce module de test, donc aucune interface extérieur n'est accepté). Passer l'ensemble des fenêtres et éditer votre fichier, insérer le code fourni : Code VHDL 2.

3. Si la syntaxe est correcte, lancer le simulateur (Run simulation) dans le "flow navigator". S'affiche alors le chronogramme associé à votre testbench (Figure 12). Pour zoomer, utiliser le clic droit, sélectionner "Full View" puis zoomer avec in/out pour obtenir un affichage aisé à lire.

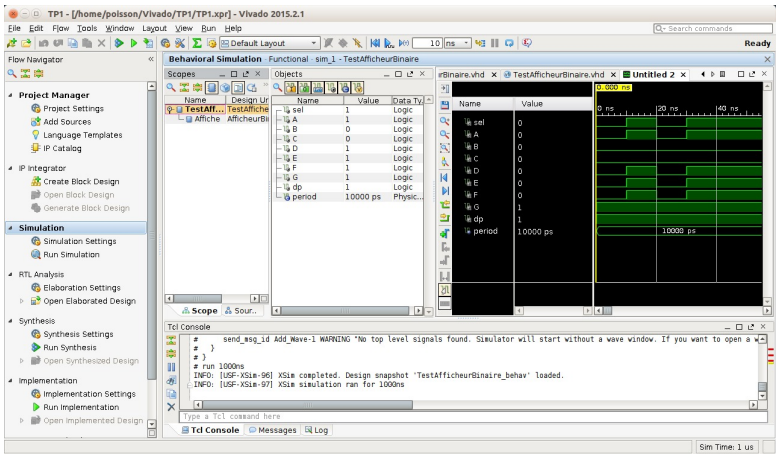

Figure 12: Copie écran - Simulation fonctionnelle

4. Après avoir vérifier le comportement attendu des sorties, fermer la fenêtre Behavioral simulation.

Test sur carte - Placement Routage

Pour faire le test sur carte, il est nécessaire de spécifier les E/S de votre module avec les E/S de la carte. Cela doit être transcrit via un fichier de contraintes.

1. Dans le volet "Flow navigator", ajouter un nouvelle source (add sources) de type "file constraints" via la ligne "add or create constraints".

2. Cliquez sur le plus pour créer un fichier des contraintes nommer le ContraintesAfficheur, éditer le pour copier le contenu du patron des contraintes que vous téléchargerez autant de fois que nécessaire sur Sakai.

3. Décommenter (en enlevant les dièses) dans le fichier les lignes concernant le commutateur (switch) SW0 et l'afficheur. Vous obtiendrez alors le listing suivant Code 3.

```
1  ## Switches
2  set_property PACKAGE_PIN V17 [get_ports {sel}]
3      set_property IOSTANDARD LVCMOS33 [get_ports {sel}]
4
5  ##7 segment display
6  set_property PACKAGE_PIN W7 [get_ports {a}]
7      set_property IOSTANDARD LVCMOS33 [get_ports {a}]
8  set_property PACKAGE_PIN W6 [get_ports {b}]
9      set_property IOSTANDARD LVCMOS33 [get_ports {b}]
10 set_property PACKAGE_PIN U8 [get_ports {c}]
11     set_property IOSTANDARD LVCMOS33 [get_ports {c}]
12 set_property PACKAGE_PIN V8 [get_ports {d}]
13     set_property IOSTANDARD LVCMOS33 [get_ports {d}]
14 set_property PACKAGE_PIN U5 [get_ports {e}]
15     set_property IOSTANDARD LVCMOS33 [get_ports {e}]
16 set_property PACKAGE_PIN V5 [get_ports {f}]
17     set_property IOSTANDARD LVCMOS33 [get_ports {f}]
18 set_property PACKAGE_PIN U7 [get_ports {g}]
```

```
19      set_property IOSTANDARD LVCMOS33 [get_ports {g}]
20
21  set_property PACKAGE_PIN V7 [get_ports dp]
22      set_property IOSTANDARD LVCMOS33 [get_ports dp]
```

Code VHDL 3 : Fichier xdc - association des variables au port de la carte BASYS3

Noter que pour chaque variable de ce programme d'association des variables utilisées aux adresses des ports de la carte, deux lignes sont nécessaires. La première ligne correspond à l'adressage c'est-à-dire la mise en concordance du nom de la variable et celui de la broche dite pin sur la carte. La seconde correspond au protocole électrique associé à cette broche.

4. Lancer l'implémentation,
5. Ensuite la génération du Bitstream,
6. Puis le transfert (Open Hardware Manager) vers la carte que vous aurez connecté via le cable USB à votre unité centrale en ayant vérifier l'état des ponts/jumpers et l'allumage de la carte. Si la carte n'est pas reconnue automatiquement, cliquer sur Open target puis auto-connect.

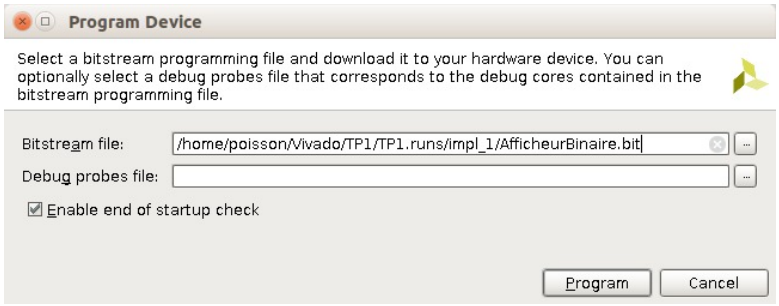

Figure 13: Copie écran - Sélection du fichier à transférer sur la cible.

7. Lancer le Program Device, une fenêtre vous demande alors de sélectionner le fichier que vous souhaitez transférer sur la carte (figure 13).
8. Le programme étant transféré, jouer avec le commutateur SWO.
9. Vous constatez que tous les afficheurs sont activés. Car nous n'avons pas sélectionner un seul afficheur (contraintes an[0] à an[3]).
10. Fermer la fenêtre Hardware Manager pour la suite.

3 A rendre pour ce projet avant la fin de la séance

Pour ce projet initiation, sur sakai:

- Décrire votre état d'avancement et blocage éventuel.
- Insérer l'image du chronogramme obtenu et donner la table de vérité associée.

Si vous n'avez pas terminé, vous pourrez ajouter vos documents en début de séance prochaine.
ce tutorial vous servira de support pour tous les projets suivants.