Guide pratique d'utilisation de Xilinx ISE 13.2

1 Introduction

1.1 Survol

Le logiciel Xilinx ISE est un outil de conception de circuit pour FPGA de Xilinx. Ce logiciel permet essentiellement d'effectuer les différentes étapes propres à la synthèse de circuits numériques sur FPGA. Il est alors possible d'en faire l'implémentation sur les différentes familles de puces fournies par Xilinx.

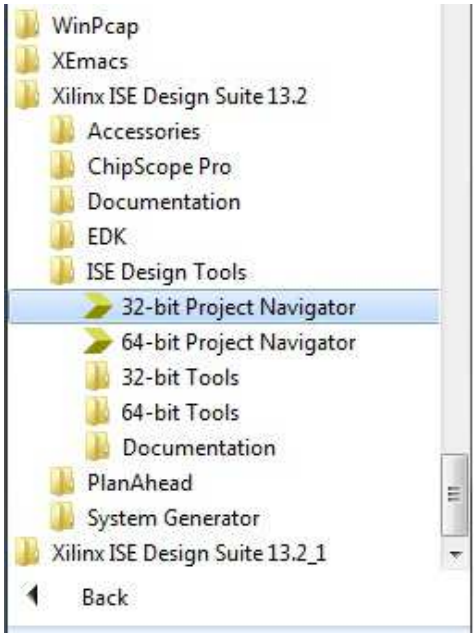
1.2 Objectif de ce guide

L'objectif de ce guide est d'énumérer et de décrire les étapes pour :

- créer un projet dans Xilinx ISE;
- décrire un circuit numérique à l'aide d'une description VHDL;
- décrire un circuit numérique à l'aide d'une description hiérarchique, basée sur un schéma, et contenant plusieurs modules décrits en VHDL dans des fichiers séparés;

1.3 Lancement de Xilinx ISE

Lancez Xilinx ISE en choisissant la commande correspondante dans le menu **Démarrer**.



2 Création d'un projet

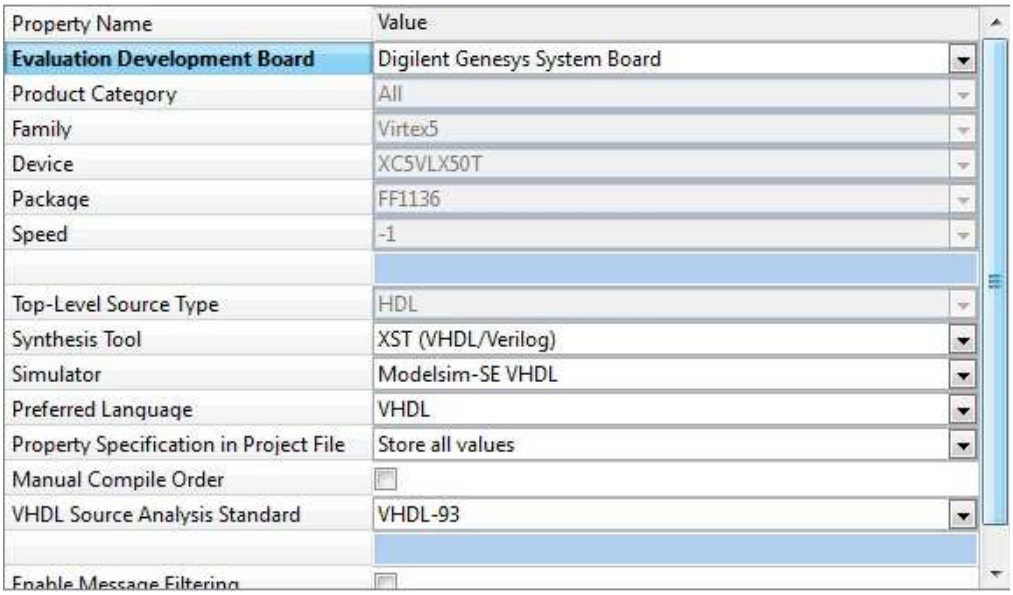
Un projet permet de regrouper plusieurs fichiers-sources pour un laboratoire ou un module en particulier. Après avoir lancé le programme, cliquez sur le bouton **New Project…** qui se situe à gauche (ou faites **File > New Project…**).

Choisissez un nom représentatif pour votre projet. Il faut aussi spécifier un répertoire où le projet sera sauvegardé. *Il est important que le chemin de ce répertoire ne contienne pas d'espaces, parce que certains outils invoqués peuvent ne pas les accepter. Dans tous les cas, essayer de toujours respecter cette pratique.*

> NOTE : Dans les laboratoires du GIGL, il est recommandé de travailler dans un répertoire personnel sur C:\temp\votrenom. *Une fois le travail terminé, il est important d'archiver ce répertoire et d'en emporter une copie avec vous. Le répertoire C:\temp\ des ordinateurs des laboratoires est régulièrement effacé.*

Conservez le « Top-level source type » en **HDL** et cliquez sur **Next**.

Vous devez remplir certains champs correspondants aux différents outils et FPGA utilisés. Reproduisez les choix de la figure suivante (pour le cours INF3500), puis cliquez sur **Next**, puis sur **Finish**.



3 Description d'un circuit numérique en VHDL

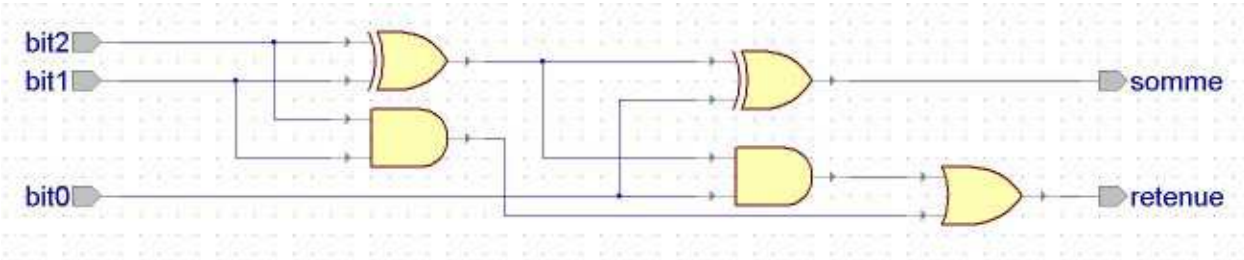
3.1 Circuit en exemple

Dans les instructions qui suivent, on construit un circuit arithmétique de base : un additionneur à 3 bits. Ce circuit accepte 3 bits en entrée et les additionne. Il a deux sorties : une retenue ainsi qu'une somme. Les sorties possibles sont donc (retenue, sortie) ∈ {(0, 0), (0, 1), (1, 0), (1, 1)}, correspondant respectivement aux cas où les trois bits d'entrée sont 0, un seul bit est 1, deux bits sont 1, et trois bits sont 1. Ce circuit indique donc le nombre de bits d'entrée qui valent 1.

Le tableau de vérité de ce circuit est donné ici :

bit0	**bit1**	**bit2**	**retenue**	**somme**
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

On peut dériver les équations pour les sorties *retenue* et *somme* grâce à un tableau de Karnaugh. Un schéma d'un circuit réalisant ces deux fonctions est donné ici :

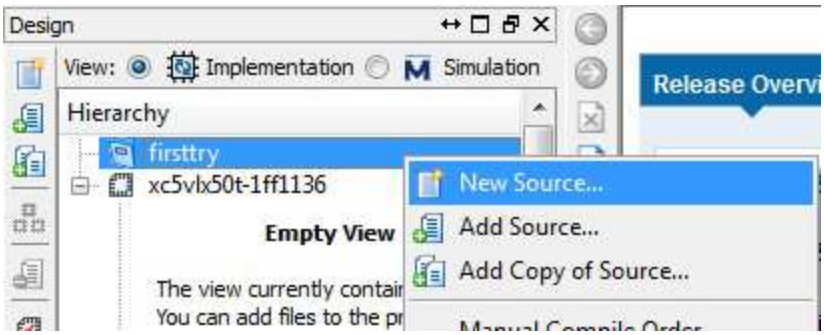


Le circuit est composé des composantes suivantes :

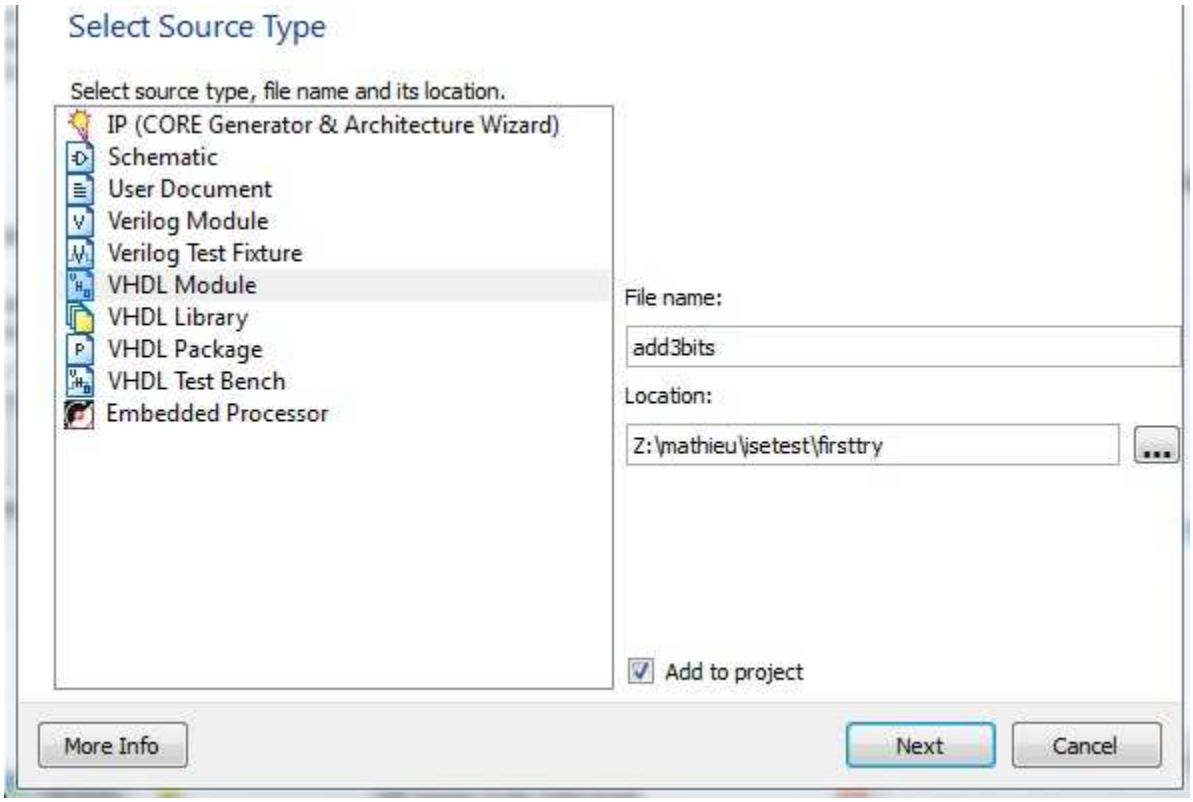
- deux portes OU-exclusif à deux entrées;
- deux portes ET à deux entrées;
- une porte OU à deux entrées; et,
- trois terminaux d'entrée et deux terminaux de sortie.

3.2 Procédure

Dans la fenêtre de Design, vous pouvez voir votre projet. Afin d'y ajouter des fichiers, Faites un clique droit sur le projet et cliquez sur **New Source…**



Dans la fenêtre qui apparait, choisissez comme type **VHDL Module** et donnez un nom représentatif au fichier (ici **add3bits**). Cliquez sur **Next**. Vous pouvez maintenant donner les entrées et sorties de votre module. Pour l'instant, cliquez sur **Next**, puis **Finish**.



Vous verrez alors un nouveau fichier VHDL dans votre projet qui comporte déjà un début de structure pour votre description matérielle. Remplacez ce qui est déjà présent par le code VHDL suivant :

```
----------------------------------
-- add3bits.vhd
-- additioneur à 3 bits
----------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity add3bits is
      port (
            bit0 : in std_logic;
            bit1 : in std_logic;
            bit2 : in std_logic;
            retenue : out std_logic;
            somme : out std_logic
      );
end add3bits;
architecture flotdonnees of add3bits is
signal T1 : std_logic;
signal T2 : std_logic;
signal T3 : std_logic;
begin
      somme <= T1 xor bit0;
      retenue <= T3 or T2;
      T1 <= bit1 xor bit2;
      T2 <= bit1 and bit2;
      T3 <= bit0 and T1;
end flotdonnees;
```

4 Synthèse et implémentation du projet

4.1 Description

La synthèse d'un circuit consiste à traduire la description du circuit en blocs disponibles dans la technologie utilisée. Par exemple, pour un circuit décrit avec un schéma et qui doit être réalisé sur un FPGA, le processus de synthèse convertit et regroupe les portes logiques du schéma en composantes réalisables sur le FPGA choisi.

L'implémentation du circuit est divisée en quatre sous étapes:

- la transformation (*mapping*) : regrouper les composantes obtenues lors de la synthèse dans des blocs spécifiques du FPGA;
- la disposition (*placement*) : choisir des endroits spécifiques sur le FPGA où disposer les blocs utilisés, et choisir les pattes du FPGA correspondant aux ports d'entrée et de sortie;
- le routage (*routing*) : établir des connexions électriques entre les blocs utilisés; et,
- la configuration (*configuration*) : convertir toute cette information en un fichier pouvant être téléchargé sur le FPGA pour le programmer.

4.2 Ports d'entrée et de sortie

Pendant l'étape de disposition de l'implémentation, il faut assigner des pattes spécifiques du FPGA à des ports d'entrée et de sortie de son design. Pour le design présent, les ports d'entrée sont `bit0`, `bit1` et `bit2`, et les ports des sortie sont `retenue` et `somme`.

L'assignation des ports se fait par l'entremise d'un fichier de contraintes avec l'extension « .ucf » (pour *user constraints file*).

Ouvrez un éditeur de texte (comme Notepad++ ou l'éditeur de texte d'Active-HDL) et copiez-y les lignes suivantes. On suppose que vous utilisez la planchette Digilent Genesys.

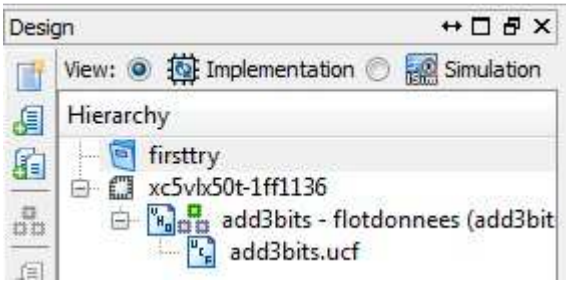
```
# add3bits.ucf
# pour planchette Digilent Genesys

# LEDs
NET "retenue" LOC = "AH8"; # LD1
NET "somme" LOC = "AG8"; # LD0

# commutateurs
NET "bit0" LOC = "J19"; # SW0
NET "bit1" LOC = "L18"; # SW1
NET "bit2" LOC = "K18"; # SW2
```

On note que chaque port est listé dans le fichier, exactement comme il apparaît dans le code. Le symbole du dièse (#) indique que le reste de la ligne est un commentaire (ne s'applique pas dans un fichier vhdl).

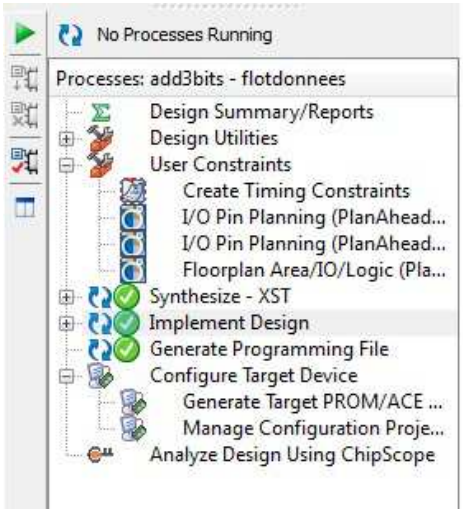
Sauvegardez le fichier dans le même répertoire que les autres fichiers source de votre design, sous le nom de `add3bits.ucf`. Puis, dans ISE, faites un clique-droit sur votre projet, mais au lieu de cliquez sur New Source…, cliquez sur **Add Source…** Vous pourrez alors choisir le fichier que vous venez de créer. Vous devriez avoir la vue suivante dans ISE :



Consultez le manuel de l'utilisateur de votre planchette de développement pour plus de détails.

4.3 Procédure

Il vous est maintenant possible de faire la synthèse de votre circuit en vue d'une implémentation sur le FPGA. Pour ce faire, il faudra utiliser la partie inférieure de l'onglet Design disponible dans ISE, que voici :



Pour réaliser les options **Synthesize –XST**, **Implement Design** et **Generate Programming File**, il faut faire un clique-droit sur chacun, puis **Run**. Notez qu'il est également possible d'utiliser l'option **Implement Top Module** pour réaliser les deux premières tâches.

Synthesize –XST : Permet de faire la synthèse du circuit en bloc configurable se retrouvant sur le FPGA.

Implement Design : Permet de faire les *mapping* et routage nécessaires pour placer le tout sur le FPGA.

Generate Programming File : Genere le fichier (.bit) utilisé pour programmer le FPGA.

Après chacune des étapes, un message de succès devrait s'afficher dans la console. Dans le cas contraire, il faut corriger les erreurs ou inspecter les avertissements (warnings) pour vous assurer qu'ils ne proviennent pas d'erreurs dans votre code. Idéalement, on devrait voir s'afficher les trois crochets verts illustrés à la figure précédente.

Lors de ces différentes étapes, plusieurs données deviendront disponibles dans le **Design Summary** (voir figure précédente). Vous aurez donc aisément accès, entre autres, aux ressources utilisées et aux résultats temporels (chemin critique, etc).

5 Programmation du FPGA et vérification

Mise en garde :

Les instructions suivantes sont particularisées pour la planchette Digilent Genesys.

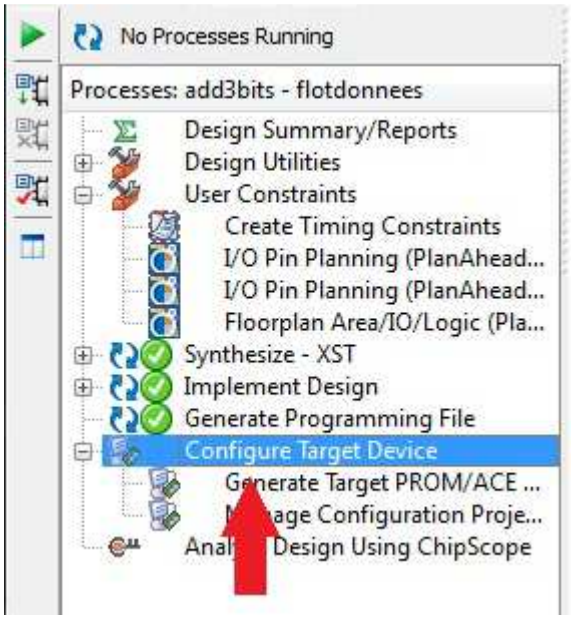
Voir le lien web suivant pour travailler avec la carte :

http://digilentinc.com/Products/Detail.cfm?NavPath=2,400,819&Prod=GENESYS

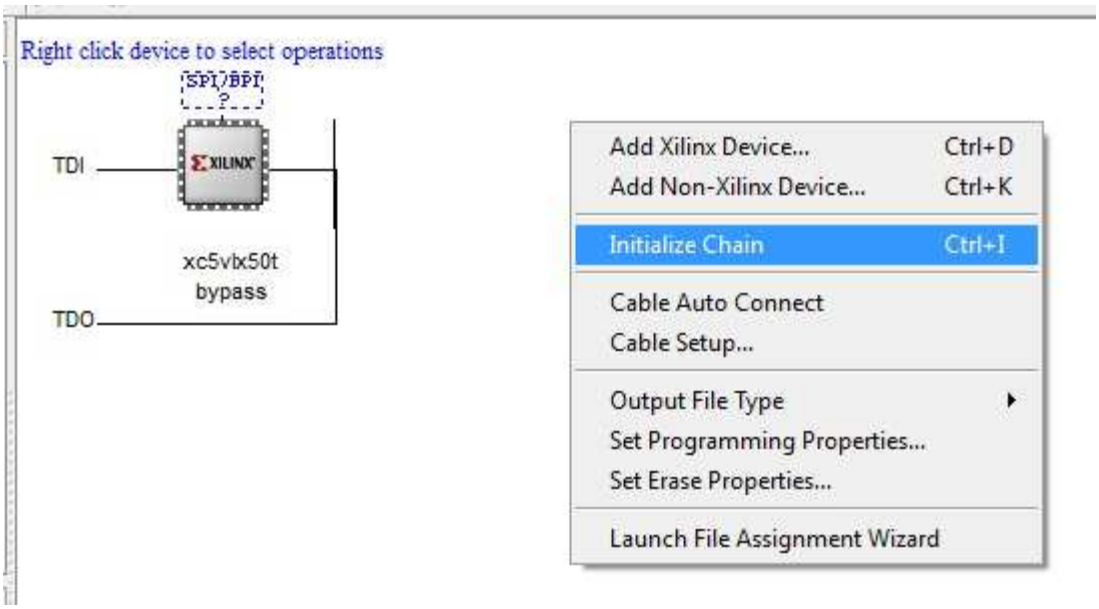
Si tout s'est bien passé lors de la synthèse, un fichier **add3bits.bit** est maintenant placé dans votre répertoire projet et est prêt à être chargé sur le FPGA de la planchette de développement. Au laboratoire, la

planchette devrait déjà être allumée et reliée au poste de travail par le câble USB2. Dans ce cas, passez à l'étape suivante. Si la planchette n'est pas allumée, informez le chargé de laboratoire.

Dans Xilinx ISE, cliquez deux fois sur **Configure Target Device**.



Lorsque l'outil **iMPACT** sera ouvert, créez un projet en faisant **File > New Project**. Si la fenêtre « Automatically create and save a project » s'affiche, cliquez sur **No**. Choisissez **create a new project**, cliquez sur **OK**, puis assurez-vous de mettre l'option de branchement automatique avant de cliquer de nouveau sur **OK**. Si la fenêtre « Auto Assign Configuration Files Query Dialog » s'affiche, cliquez sur **No**. Dans l'espace en blanc, cliquez sur le bouton de droite et choisissez **Initialize Chain**.



Cliquez sur **Yes** si la fenêtre « Auto Assign Configuration Files Query Dialog » s'affiche. Choisissez le fichier **add3bits.bit** que vous trouverez dans le dossier projet et cliquez sur **Open**. Si le logiciel vous demande d'attacher un SPI ou un BPI PROM, répondez par **No**. Cette option sert à charger la mémoire PROM de la planchette. Cliquez sur le symbole représentant le FPGA, puis sur le bouton de droite de la souris. Choisissez **Program …** Cliquez sur **OK** dans la fenêtre Device **Programming Properties** (si elle s'affiche), sans rien changer.

Voilà! Vous pouvez maintenant vérifier votre design sur le FPGA.

Essayez toutes les combinaisons possibles d'entrées et confirmez que les sorties sont bien conformes au tableau de vérité de la section 3.1

6 Description hiérarchique d'un projet comportant plusieurs modules

6.1 Principes de base et circuit en exemple

Les circuits numériques ne sont jamais décrits à l'aide d'un seul fichier contenant du code VHDL. La plupart regroupent plusieurs modules indépendants qui sont tirés de bibliothèques existantes ou bien qui sont développés sur mesure pour un projet en particulier. Quand un projet regroupe un grand nombre de modules décrits dans autant de fichiers, il peut devenir difficile de s'y retrouver. De plus, il est nécessaire de décrire les interconnexions entre les différents modules, et encore là il peut devenir difficile de s'y retrouver. Une description hiérarchique utilisant des schémas permet de gérer à la fois la complexité du projet et de décrire les interconnexions de façon efficace. Il est également possible de décrire en VHDL ces interconnexions.

Dans cette section, vous allez créer un projet et y ajouter des fichiers VHDL existants. Vous allez ensuite relier les modules en utilisant le fichier .bde créé lors du tutoriel sur ActiveHDL.

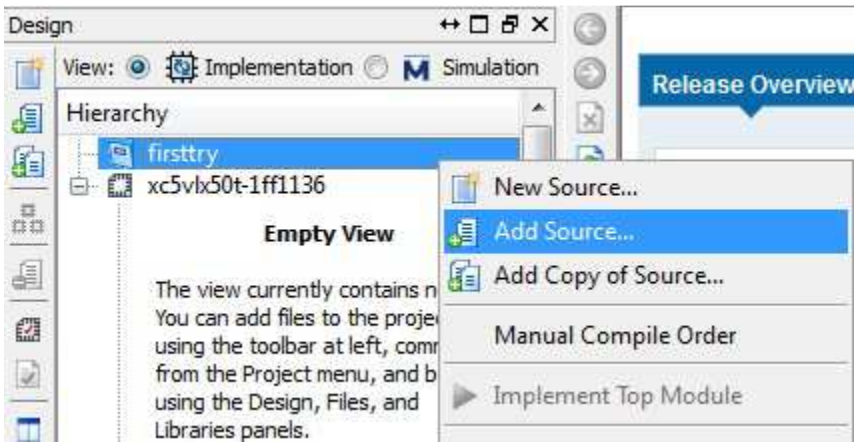
6.2 Procédure à suivre

Créez un nouveau projet en suivant les instructions de la section 2 et donnez-lui le nom « tutoriel ».

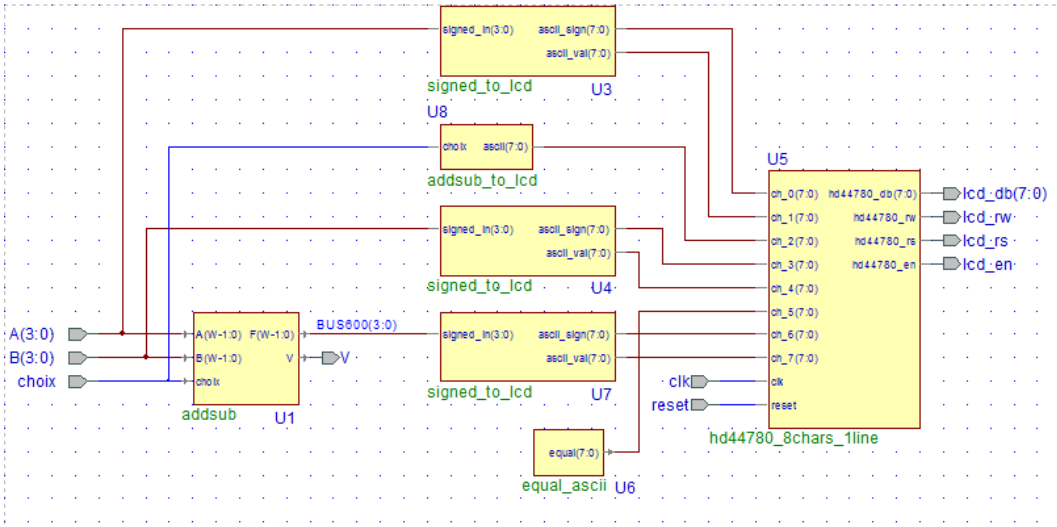
Copiez du site web du cours INF3500 dans votre dossier projet les fichiers :

- addsub.vhd
- hd44780_8chars_1line.vhd
- hd44780_simple.vhd
- equal_ascii.vhd
- signed_to_lcd.vhd
- addsub_to_lcd.vhd
- addsub.ucf

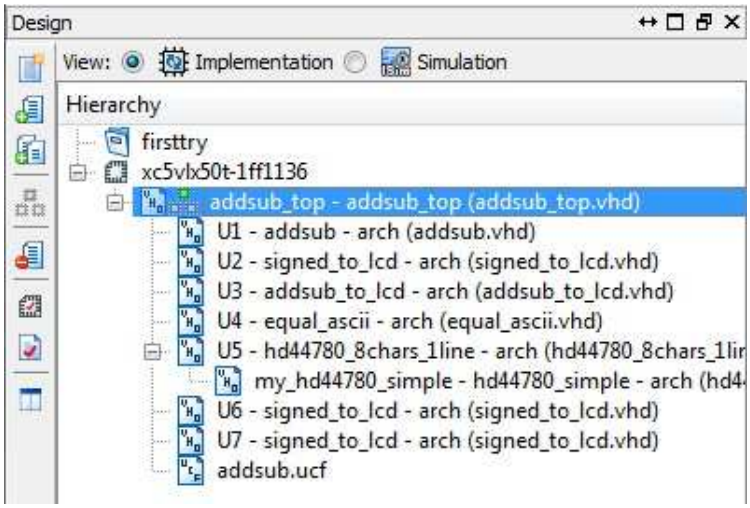
Dans Xilinx ISE, rajoutez ces fichiers dans votre projet en faisant un clique-droit, puis **Add Source…**



Inspectez les fichiers que vous avez copiés. Le fichier addsub.vhd décrit un module qui peut additionner ou soustraire deux nombres `A` et `B` selon la valeur d'un signal de sélection `choix`. Il a une sortie `F` représentant le résultat et `V` indiquant si un débordement a lieu. Le paramètre `W` indique la largeur des opérandes et du résultat. Lors du tutoriel sur ActiveHDL, vous avez réalisé le schéma suivant :



Afin de récupérer une partie du travail déjà réalisé, nous allons convertir ce fichier .bde en VHDL. Pour ce faire, ouvrez votre projet ActiveHDL ainsi que votre .bde (image ci-haut). Cliquez ensuite, dans la barre de menu supérieure, sur **Diagram > Generate HDL Code**. Un code VHDL qui instancie tous vos blocs selon vos connexions est généré. Vous pouvez alors le récupérer (dans le dossier « compile » de votre dossier de projet ActiveHDL) et l'insérer dans votre projet ISE. Celui-ci devrait avoir à peu près la forme suivante :



6.3 Implémentation du circuit et programmation du FPGA

Suivez les instructions des sections 4.3 pour faire l'implémentation du circuit, puis de la section 5 pour en faire l'implémentation sur la planchette.

Il se peut que vous voyiez des avertissements (*warnings*) lors de la synthèse. Assurez-vous simplement que ces avertissements ne concernent que le bloc hd44780_8chars_1line.vhd (hd44780_simple).

Notez que, pour l'implémentation, vous devez utiliser le fichier addsub.ucf qui est particularisé pour les noms de terminaux que vous avez réalisés dans ActiveHDL.