Des additionneurs

Objectifs

Le but de ce TP est d'écrire et de tester vos premiers composants électroniques conçus en VHDL, et ceci afin de vous familiariser avec les principes de base du langage VHDL.

Première partie

Sources VHDL

Récupérez le fichier d'archive additionneur.tar et placez-le dans un répertoire VHDL/Sources à créer ; cette archive contient

- une porte `ou`, un demi-additionneur,
- l'additionneur 1 bit
- un fichier de test de l'additionneur, configuré pour tester la vue flot de données de l'additionneur.

Attention : Le logiciel que nous utilisons pour ce module VHDL n'a pas l'air d'aimer les noms de répertoires avec espace : évitez donc d'utiliser de tels noms.

Deuxième partie

Testons l'additionneur 1 bit

1 lancement de `Xilinx ISE`

`Xilinx ISE` se lance par le menu par la commande `xilinx` qui est un script qui positionne des variables d'environnement et appelle l'exécutable.

Au bout de quelques temps, un environnement, style `Eclipse` s'ouvre (Fig.1).

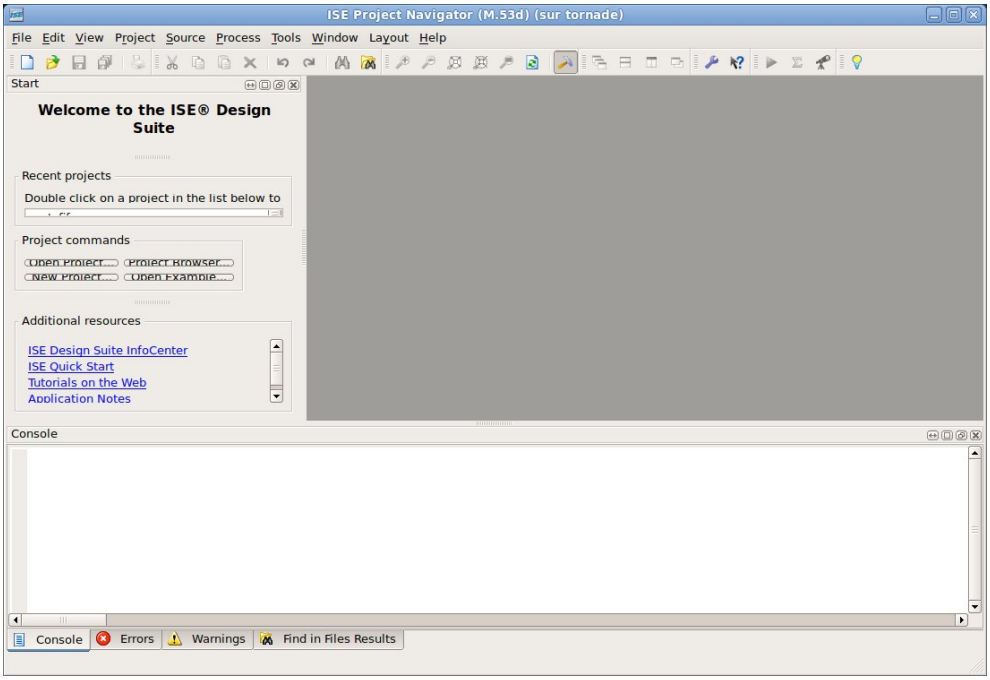

Figure 1 – `Xilinx ISE` Project Navigator

2 Création d'un projet pour tester la vue flot de donnée de l'additionneur

Suivre le menu **File** → **New Project** ; la fenêtre de la Fig. 2 s'ouvre.

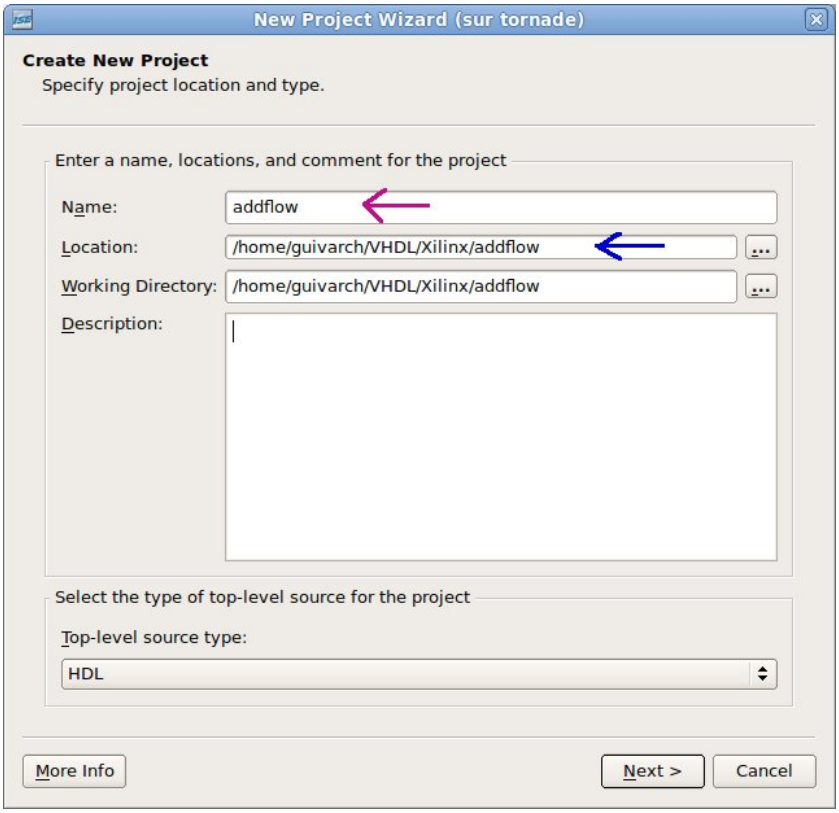

Figure 2 – Création d'un nouveau projet

2.1 Nom du projet

À l'endroit de la flèche bleue, taper le nom d'un répertoire dans lequel `Xilinx ISE` mettra ses fichiers de travail (ici répertoire `Xilinx` dans VHDL) [1].

À l'endroit de la flèche pourpre, taper le nom du projet (ici `addflow`).

Taper **Next** pour passer à la fenêtre suivante.

Remarque

Pensez à créer ce fichier `Xilinx` à un endroit différent de celui où vous placerez vos sources.

Les projets `Xilinx` occupent beaucoup de place et les regrouper dans un seul endroit, sans les sources, vous permettra de les effacer facilement tout en gardant ce qui est intéressant.

1. Ce répertoire n'est a priori qu'à créer la première fois, `Xilinx ISE` le gardant en mémoire.

2.2 Paramétrage de la carte FPGA et du simulateur

Rentrer exactement les valeurs données par la Fig.3 : les paramètres de la carte FPGA ne seront pas très utiles pour ce premier TP par contre il est important de positionner le simulateur à ISE Simulator [2].

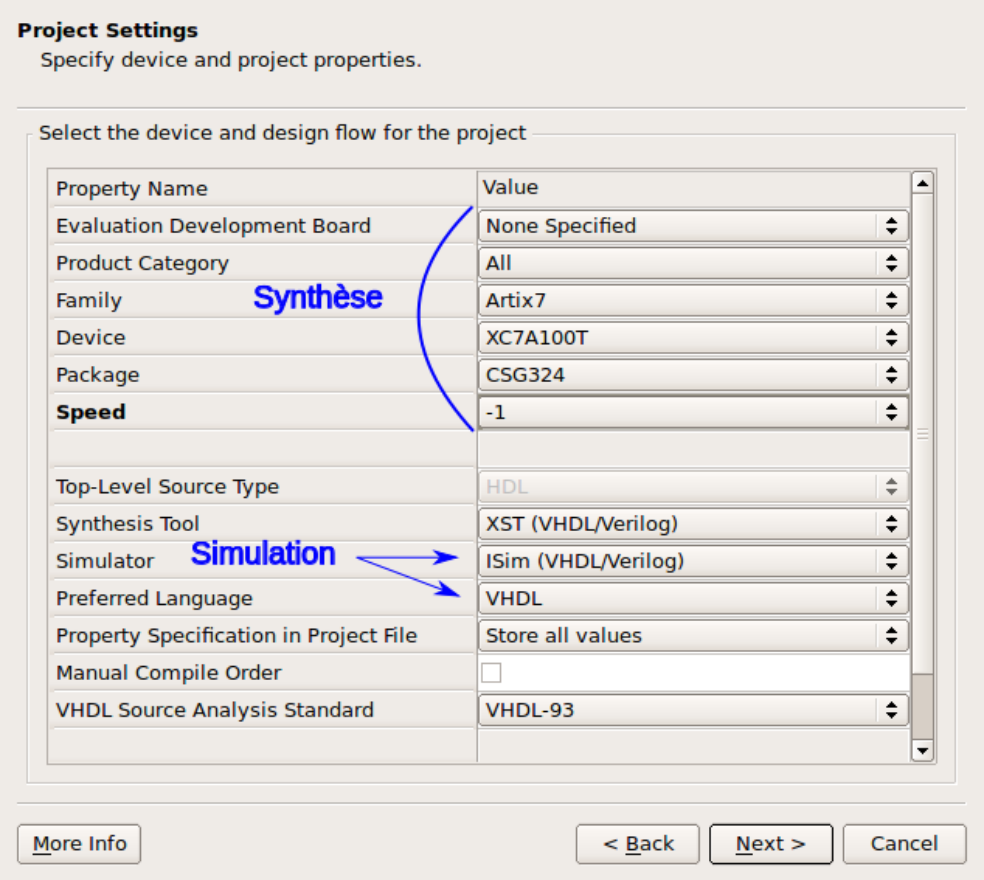

Figure 3 – Paramétrage du simulateur

Taper **Next** pour passer à la fenêtre suivante.

2.3 Fin de la création d'un projet

Taper **Finish** pour revenir au Xilinx ISE Project Navigator.

2. Là aussi, ce paramétrage n'est à effectuer qu'une fois.

3 Ajout des fichiers sources du projet

Suivre le menu **Project** → **Add Source** ; la fenêtre de la Fig.4 s'ouvre.

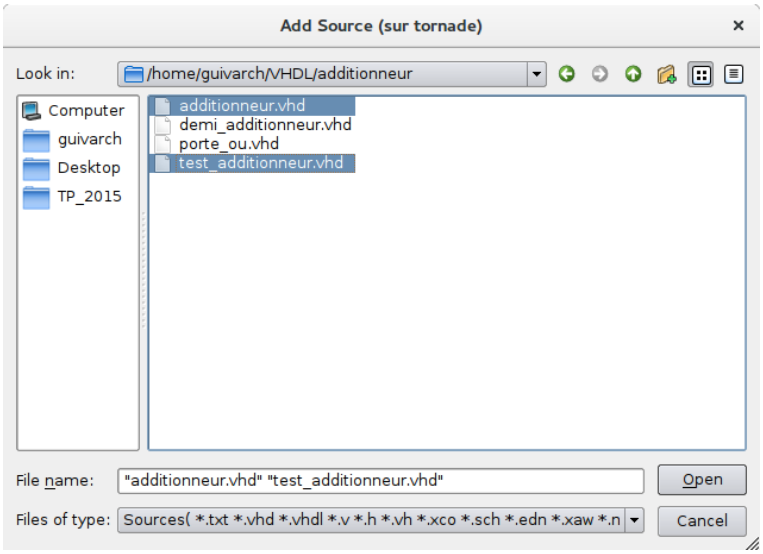

Figure 4 – Ajout des fichiers sources

Naviguer pour trouver les fichiers sources fournis et sélectionner les 2 fichiers nécessaires :

1. le fichier de l'additionneur, `additionneur.vhd`,
2. le fichier de test de l'additionneur `test_additionneur.vhd`

En validant ce choix en cliquant sur **Open**, une petite analyse syntaxique des fichiers est réalisée ; ils sont ajoutés au projet en appuyant sur **OK**.

4 Simulation de l'additionneur avec le fichier vhdl de test fourni

Il a été vu en cours que l'on pouvait tester un composant par l'intermédiaire d'un composant "spécial" de test qui permet de fournir des valeurs en entrée du composant (stimuli) : c'est le rôle du composant donné par le fichier VHDL `test_additionneur.vhd`.

4.1 Se positionner en mode `Simulation`

Dans le menu indiqué par la flèche bleue de la Fig.5, choisir la vue `Simulation`.

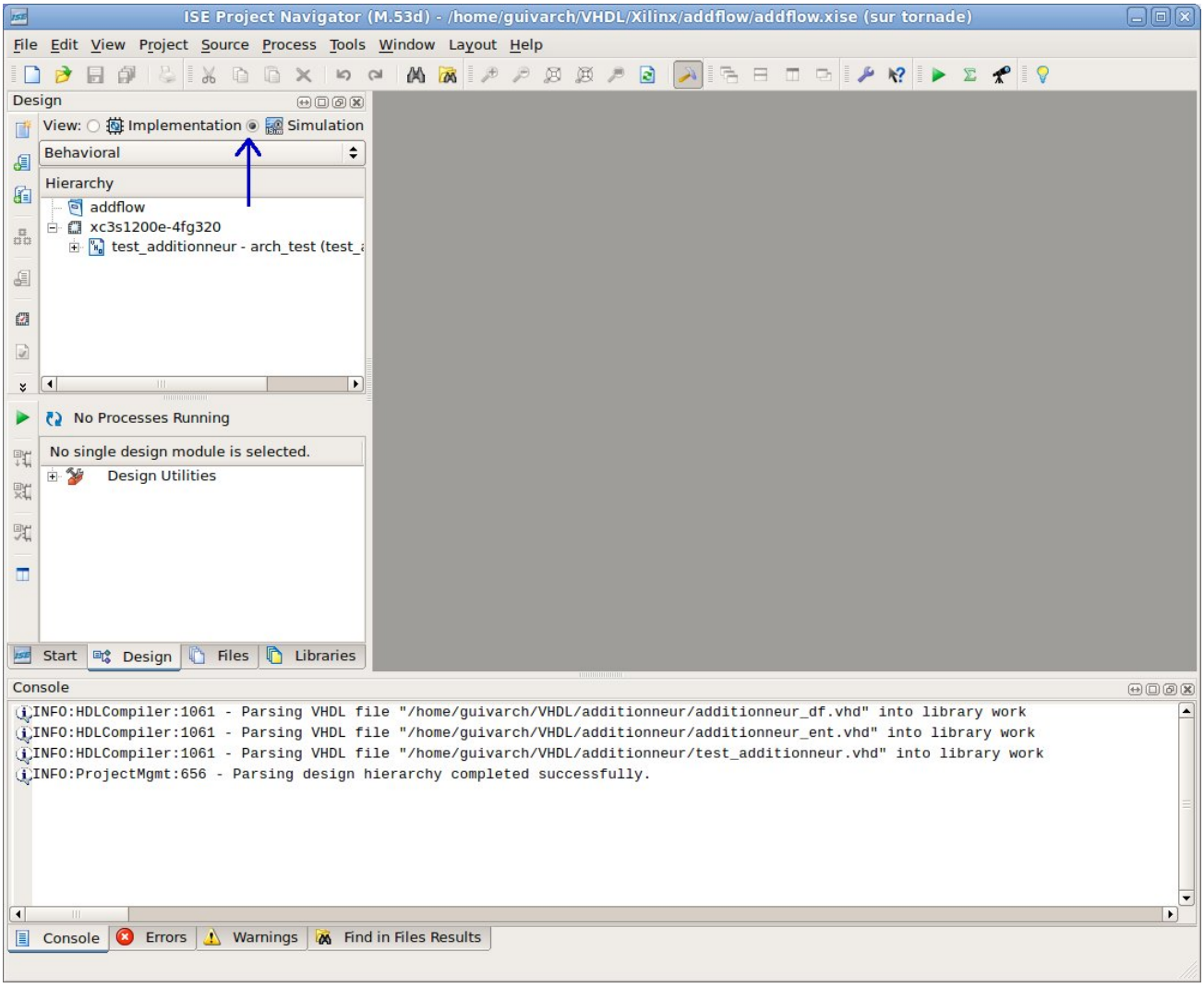

FIGURE 5 – Simulation

4.2 Lancement de la simulation

Le fichier de test peut être édité en double cliquant dessus (flèche bleue de la Fig.6) par un éditeur interne à `Xilinx ISE`.

Pour lancer la simulation, sélectionner `test_additionneur` (flèche bleue) et double-cliquer sur `Simulate Behavorial Model` (flèche pourpre).

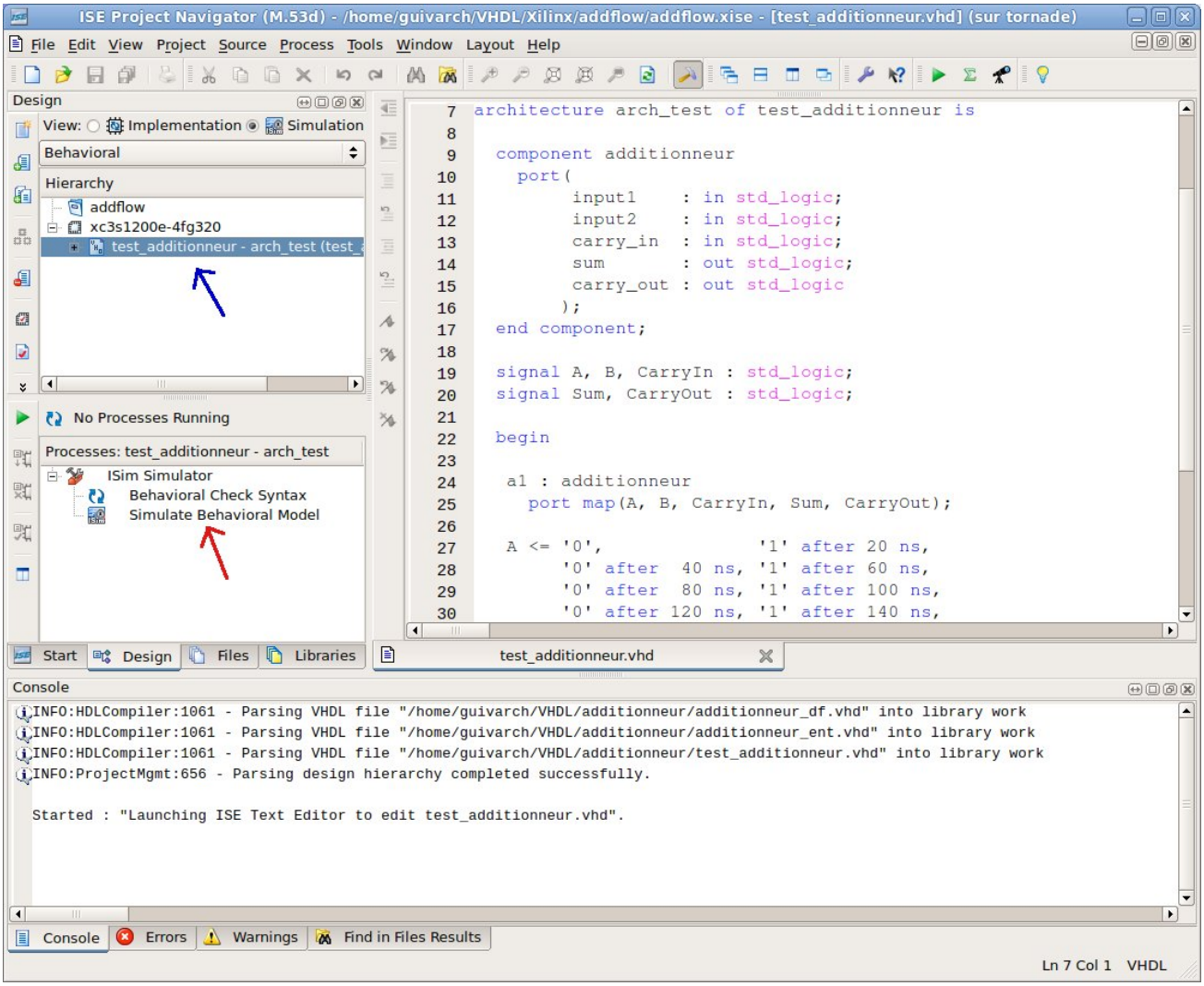

Figure 6 – Lancement de la simulation du test VHDL de l'additionneur

4.3 Analyse de la simulation

Après quelque temps de compilation (messages d'erreur possibles dans la `Console`[3]), une nouvelle application `ISim` s'ouvre qui va vous permettre de visualiser la simulation, essentiellement sous forme de chronogrammes (*Waveform* 7) (zoom peut-être nécessaire, icônes entourées de vert).

La simulation peut être rejouée en la ré-initialisant (flèche bleue) et en la relançant (flèche pourpre).

Si vous avez modifié le code, vous pouvez regénérer la simulation via le bouton *Re-launch* (flèche jaune).

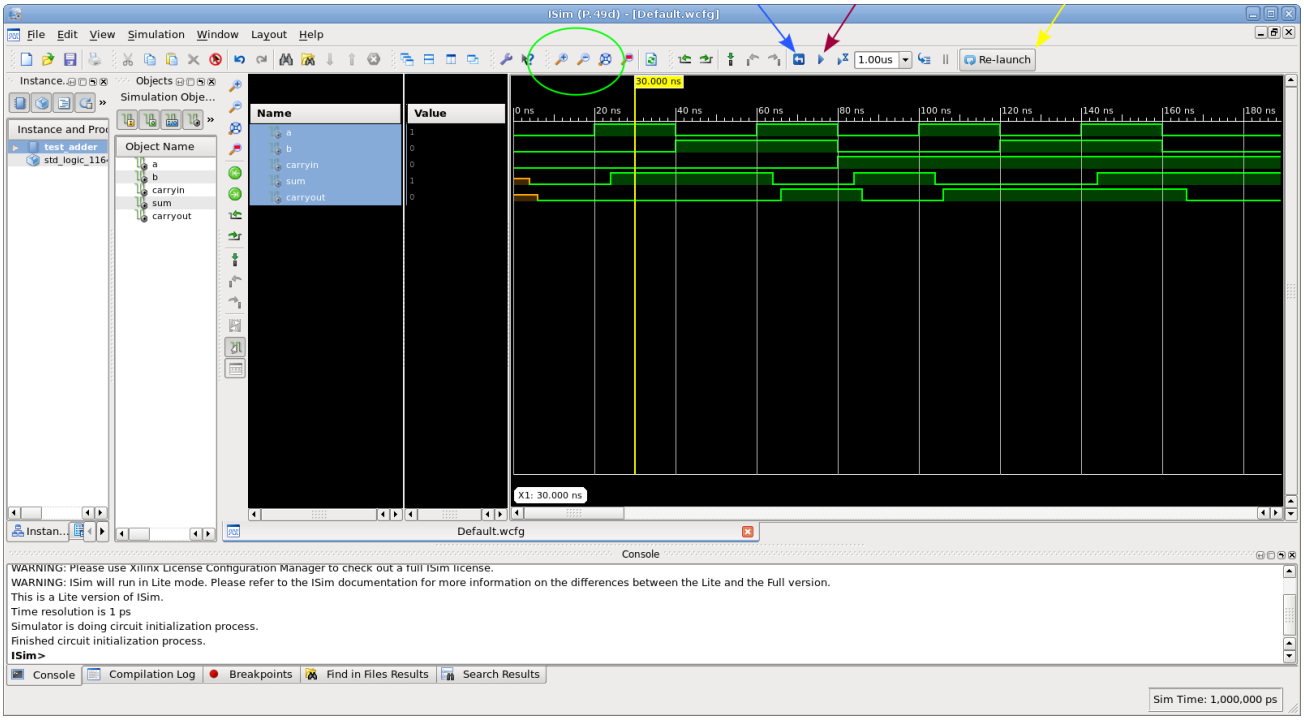

Figure 7 – Résultats de la simulation de l'additionneur sous forme de chronogrammes

Travail à faire

Vérifiez que cette vue de l'additionneur fonctionne comme attendu.

3. Pas avec les fichiers fournis!!

4.4 Ajout de signaux aux chronogrammes

Par défaut, seuls les signaux internes du fichier VHDL de test sont donnés par les chronogrammes. Pour rajouter, un signal interne (par exemple le signal i de l'additionneur) (Fig.8),

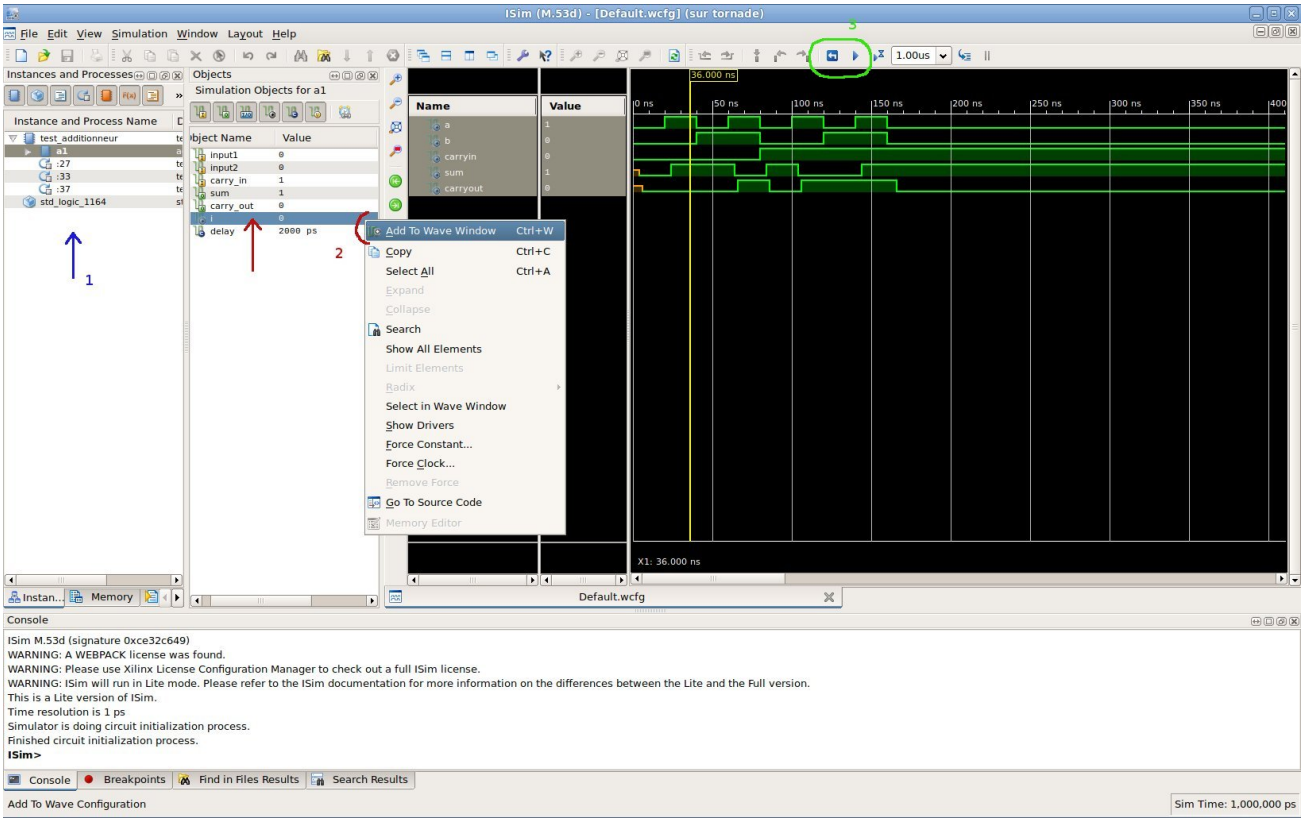

FIGURE 8 – Ajout d'un signal interne à l'additionneur

1. dans la sous fenêtre `Instance and Process Name`, sélectionnez le composant `a1` (c'est l'additionneur)
2. trouver le signal i dans la sous-fenêtre `Simulation Objects` et par un click-droit le rajouter aux chronogrammes (flèche pourpre),
3. ré-initialiser la simulation et la rejouer,

Travail à faire

Tester les 2 autres architectures de l'additionneur en changeant l'architecture considérée via la configuration (ligne 24 du fichier `test_additionneur.vhd`).

1. **structurelle**, en n'oubliant pas les sous-composants,
2. **comportementale**.

On pourra envisager de tester les 3 architectures à la fois en l'instanciant 3 fois avec 3 configurations différentes.

Troisième partie

Un peu de développement

5 Additionneur 4 bits

Travail à faire

Développer et tester (**avec un nouveau projet**) un additionneur 4 bits en utilisant de manière structurelle 4 additionneurs 1 bit (circuit rappellé par la Fig.9).

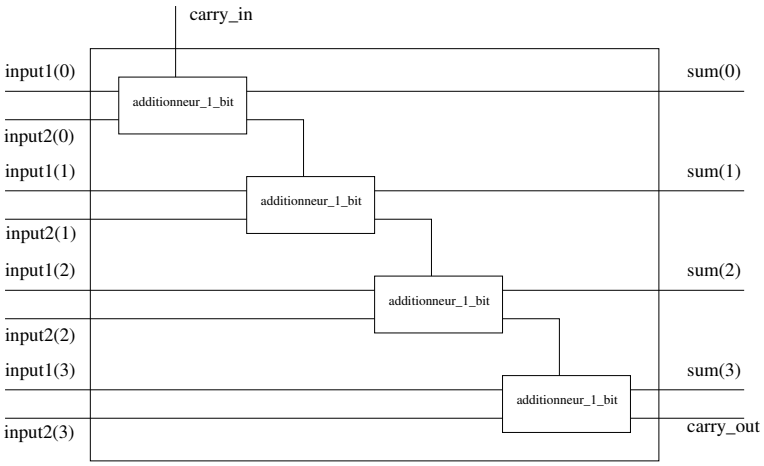

Figure 9 – Additionneur 4 bits à partir de 4 additionneurs 1 bit

Pour cela nous allons nous appuyer sur les outils fournis par ISE pour générer des squelettes de composants et de test (voir pages suivantes).

5.1 Génération d'un vhdl Module

Suivre le menu **Project** → **New Source** ; une fenêtre **New Source Wizard** Fig. 10 s'ouvre.

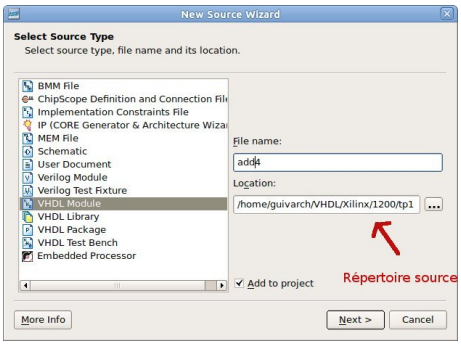

FIGURE 10 – New Source Wizard

Choisir **VHDL Module** en lui indiquant un nom de composant (par exemple `add4`). Attention à indiquer l'emplacement (*location*) de votre répertoire qui contient les sources pour pouvoir retrouver par la suite votre composant.

Dans la fenêtre suivante Fig. 11, indiquer le nom que vous voulez donner à l'architecture, ainsi que la liste des ports en entrée et en sortie de l'additionneur 4 bits. Attention à la manière de spécifier les bus.

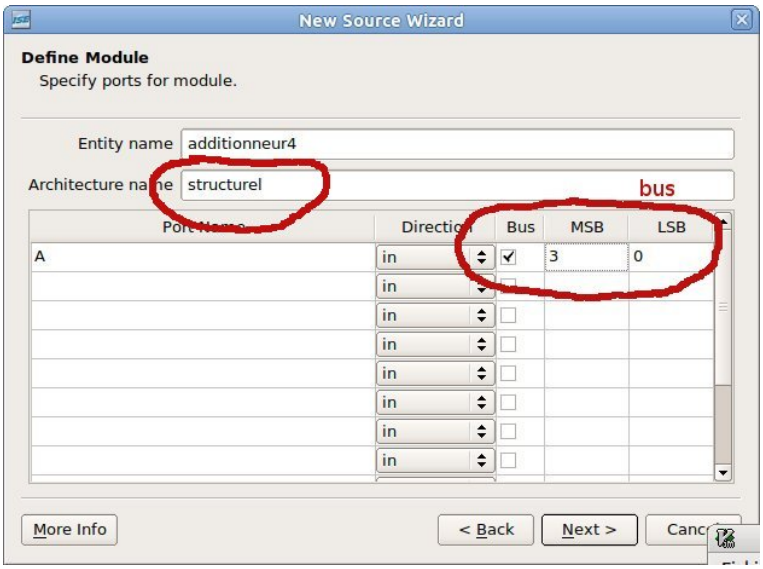

FIGURE 11 – New Source Wizard : architecture & bus

Après avoir bien vérifié l'interface de votre composant, vous obtenez un squelette avec l'`entity` remplie et l'architecture à compléter.

5.2 Utilisation d'un *Template* pour intégrer un sous-composant

L'additionneur 4 bits a comme sous-composants 4 additionneur 1 bit.

Dans l'architecture de l'additionneur 4 bits, il faut :

1. rappeler l'interface de l'additionneur 1 bit dans la zone de déclaration,
2. instancier et connecter 4 additionneurs 1 bit dans le corps.

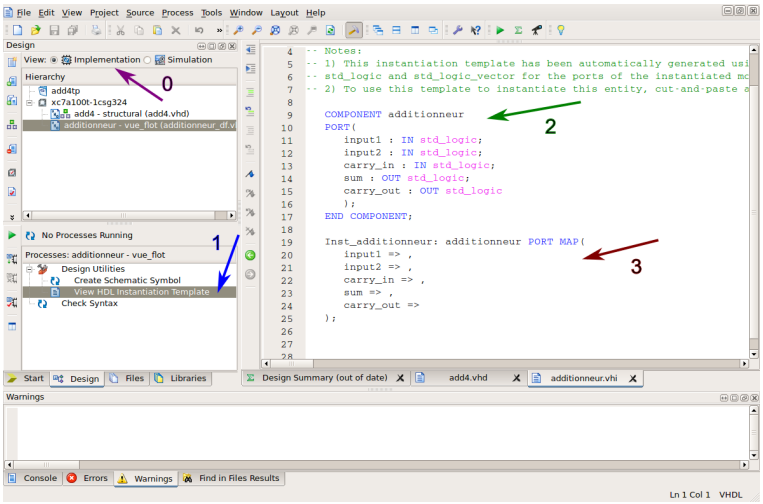

Figure 12 – HDL Instantiation Template

Un *Template* de ces lignes de code peut être générer automatiquement. Pour cela,

1. se positionner en mode `Implementation` (0)
2. se positionner sur l'additionneur 1 bit (qu'il faut ajouter au projet)
3. double-cliquer sur `View HDL Instantiation Template` (1)
4. les lignes de code sont générées : vous pouvez les copier dans la zone de déclaration (2) ou dans le corps de l'architecture (3) et connecter les sous-composants

5.3 Génération d'un fichier de test : Testbench

Suivre le menu **Project** → **New Source**.

Choisir **VHDL Test Bench** en lui précisant son nom et son emplacement.

Dans la fenêtre suivante Fig. 13, indiquer le composant que vous voulez tester (a priori `add4`).

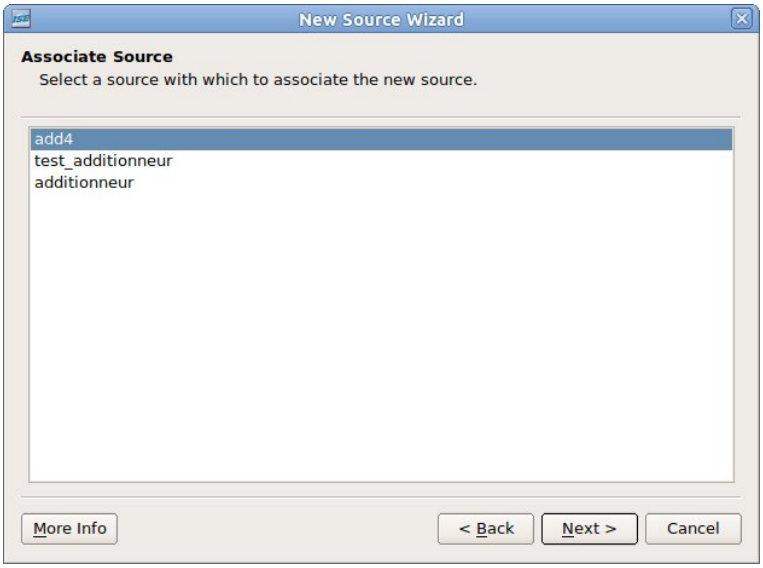

Figure 13 – New Source Wizard : Test Bench

Après avoir confirmé, vous obtenez un squelette de test dans lequel le composant à tester est instancié et dans lequel vous devez fournir des valeurs sur les ports en entrée. Pour ce composant :

- commencer par enlever tout ce qui fait référence à l'horloge (`clock`),
- inutile de garder le process `stim_proc`,
- par des affectations de signaux concurrentes, donner des valeurs aux deux bus d'entrée (rappel syntaxique : exemple de valeur d'un signal sur 4 bits `"0001"`) ainsi qu'à la retenue entrante,
- lancer la simulation.

5.4 Ajustement de la durée de la simulation

Vous aurez peut être besoin, pour voir toutes les variations des signaux, de changer le temps final de la simulation (1000 ns par défaut). Pour cela, il vous faut faire un click droit sur **Simulate Behavorial Model** → **Process properties** et changer le temps **Simulation Run Time** (voir Fig.14).

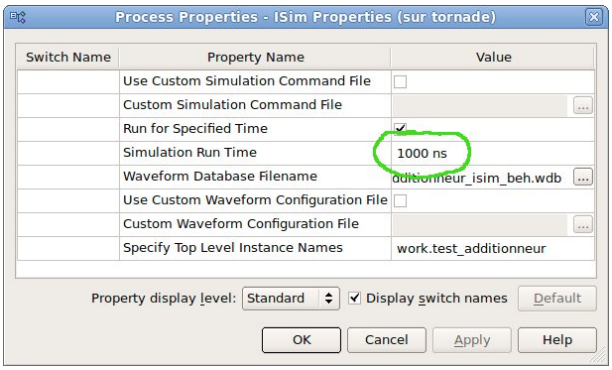

Figure 14 – Changer le temps final de la simulation