

I - Environnement de travail

I.a Environnement Quartus Prime et Questa

- Quartus est l'IDE qui vous permettra de développer des solutions pour les FPGA Intel. Vous pourrez y rédiger votre code, synthétiser vos solutions, consulter les rapports de synthèse, et programmer la carte FPGA. Vous pourrez également appeler le logiciel Questa depuis Quartus.
- Questa est le logiciel qui vous permettra de simuler vos implémentations VHDL.

I.b Rendus

Vous consignerez vos notes et observations **directement dans vos sources** :

- S'il vous est demandé de commenter des résultats, **consignez-vos observations dans un commentaire en en-tête** du fichier contenant votre fonction main.
- Commentez également votre code **dans les lignes**, de manière à expliquer vos choix

Il vous est demandé d'enregistrer des captures d'écran de vos simulations dans un dossier que vous créerez dans chaque projet.

Il n'est pas demandé de compte-rendu séparé. Les modalités de rendu des TP's seront discutées lors de la première séance.

II - Exercices

II.a Synthèse d'un additionneur

Créer un nouveau projet « TP1_additionneur » (voir Annexes)

Dans cette série d'exercices, on décrira un circuit additionneur en VHDL. A des fins pédagogiques, on s'attachera à utiliser les opérateurs logiques plutôt que d'utiliser les additionneurs présents sur le FPGA (on s'interdira donc l'utilisation du symbole « + »).

II.a.1 Synthèse d'un demi-additionneur 1-bit – Description d'un composant

Le bloc réalisé dans cet exercice peut être décrit comme suit :

Bloc demi-additionneur 1b			
	Nom	Taille	Description
Entrées	A	1 bit	Premier opérande
	B	1 bit	Deuxième opérande
Sorties	S	1 bit	Somme A + B
	C	1 bit	Retenue de A + B

Travail demandé :

1. Rappelez les opérations logiques du demi-additionneur permettant de calculer S et C en fonction de A et B .
2. Ajoutez à votre projet un nouveau fichier `half_adder.vhd`, dans lequel vous décrirez dans une *entity* nommée `half_adder` le fonctionnement du demi-additionneur
3. Ajoutez à votre projet un nouveau fichier de *testbench* `tb_half_adder.vhd`, dans lequel vous décrirez les entrées à appliquer au demi-additionneur pour tester son bon fonctionnement
4. Synthétisez et simulez votre demi-additionneur

II.a.2 Synthèse d'un additionneur complet 1-bit – Instanciation d'un composant en VHDL

Dans cet exercice, on réutilisera le demi-additionneur précédemment développé pour réaliser un additionneur complet. Le bloc réalisé peut être décrit comme suit :

Bloc additionneur 1b complet			
	Nom	Taille	Description
Entrées	A	1 bit	Premier opérande
	B	1 bit	Deuxième opérande
	Cin	1 bit	Retenue à l'entrée
Sorties	S	1 bit	Somme A + B + Cin
	Cout	1 bit	Retenue à la sortie

Travail demandé :

1. Rappelez comment réaliser un additionneur complet à partir de deux demi-additionneurs
2. Ajoutez à votre projet un nouveau fichier `full_adder.vhd`, dans lequel vous décrirez dans une *entity* nommée `full_adder` le fonctionnement de l'additionneur complet. Pour ce faire, vous instancierez deux demi-additionneurs précédemment décrits.
3. Ajoutez à votre projet un nouveau fichier de *testbench* `tb_full_adder.vhd`, dans lequel vous décrirez les entrées à appliquer à l'additionneur complet pour tester son bon fonctionnement
4. Synthétisez et simulez votre additionneur complet

II.a.3 Additionneur complet 4-bits – Instanciation d'un composant en interface graphique

Dans cet exercice, on réutilisera l'additionneur complet 1b précédemment développé pour réaliser un additionneur complet 4b. Cette fois-ci vous réaliserez la description de l'additionneur en interface graphique, comme un schéma-blocs.

Le bloc réalisé peut être décrit comme suit :

Bloc additionneur 4b complet			
	Nom	Taille	Description
Entrées	A	4 bits	Premier opérande
	B	4 bits	Deuxième opérande
	Cin	1 bit	Retenue à l'entrée
Sorties	S	4 bits	Somme A + B + Cin
	Cout	1 bit	Retenue à la sortie

Travail demandé :

1. Rappelez comment réaliser un additionneur à retenue propagée à partir d'additionneurs complets
2. Ajoutez à votre projet un nouveau fichier `full_adder_4b.bdf1`, dans lequel vous dessinerez en interface graphique le schéma de l'additionneur 4b complet. Pour ce faire, vous utiliserez des instances de l'additionneur complet précédemment décrit².
3. Exportez votre schéma-blocs en un fichier VHDL³. Ensuite, retirez le fichier bdf du projet, et ajoutez le fichier vhd généré à sa place.
4. Ajoutez à votre projet un nouveau fichier de *testbench* `tb_full_adder_4b.vhd`, dans lequel vous décrirez les entrées à appliquer à l'additionneur 4b complet pour tester son bon fonctionnement
5. Synthétisez et simulez votre additionneur 4b complet

II.a.4 Additionneur complet 4-bits – interaction avec les entrées/sorties de la carte

On souhaite implémenter sur la carte Cyclone V GX l'additionneur réalisé, afin d'en démontrer la fonctionnalité en pratique. On utilisera alors :

- Les interrupteurs comme entrées (A : SW0-3 ; B : SW4-7 ; Cin : SW9)
- Les LEDs rouges comme affichage des entrées (A : LEDR0-3 ; B : LEDR0-3 ; Cin : LEDR9)
- Les LEDs vertes comme affichage de la sortie (S : LEDG0-3 ; Cout : LEDG7)

D'abord, vous devez décrire comment votre module s'interface avec les entrées/sorties de la carte de développement. On appellera communément cette description le **Top Level**, puisqu'il se trouvera au sommet de la hiérarchie de votre solution.

Les entrées/sorties de cette *entity* décrivent les interfaces matérielles de la carte avec lesquelles vos modules s'interfaceront. Vous pouvez lire l'intégralité de ces mots-clés dans le fichier `baseline_c5gx.v`, ajouté automatiquement à vos projets lorsque vous les créez. Ici, vous utiliserez les mots-clés réservés

¹ File > New > Block Diagram / Schematic file

² Vous devez au préalable créer une vue symbolique du composant à instancier par clic droit sur la source > Create Symbol Files for Current File

³ File > Create / Update > Create HDL Design File from Current File.

LEDR (sorties vers les 10 LEDs rouges), LEDG (sorties vers les 8 LEDs vertes), et SW (entrées vers les 10 switches).

Travail demandé

1. Ajoutez à votre projet un fichier toplevel.vhd
2. Décrivez l'entité toplevel, dont l'instruction port sera :

```
entity toplevel is
  port (
    LEDR : out std_logic_vector(9 downto 0);
    LEDG : out std_logic_vector(7 downto 0);
    SW    : in  std_logic_vector(9 downto 0)
  );
end entity;
```

3. Instancier dans l'architecture de toplevel votre additionneur, en connectant adéquatement ses entrées/sorties à LEDR, LEDG, et SW.
4. Implémentez votre additionneur sur la carte en utilisant le bouton program device⁴ et vérifier qu'il se comporte comme attendu. Félicitations, vous avez implémenté votre première fonction combinatoire sur FPGA.

⁴ Voir annexe en fin de document

II.b Synthèse d'un transcodeur pour afficheur 7 segments (optionnel)

Pour rendre la démonstration de votre sommateur plus convaincante et pour vous faire construire un bloc combinatoire plus réaliste, on se propose d'afficher les entrées-sorties du sommateur sur les afficheurs 7 segments⁵ :

- L'afficheur HEX3 affichera l'opérande A sous forme hexadécimale
- L'afficheur HEX2 affichera l'opérande B sous forme hexadécimale
- L'afficheur HEX0 affichera le résultat B sous forme hexadécimale

Travail demandé

1. Créer une entity dans un nouveau fichier *transcodeur_7seg.vhd* permettant de transcoder un nombre représenté sur 4 bits en sa représentation hexadécimale sur afficheur 7 segments, définie ainsi :

Transcodeur 7 segments			
	Nom	Taille	Description
Entrées	BIN	4 bits	Valeur d'entrée encodée en binaire
Sorties	SEG	7 bits	Valeur mise en forme pour les afficheurs 7 segments

2. Instancier dans votre toplevel trois transcodeurs afin d'afficher les entrées et sorties de votre additionneur (vous utiliserez les mots-clés réservés HEX3, HEX2, et HEX0 pour accéder aux afficheurs)
3. Implémentez cette architecture sur le FPGA

⁵ Les afficheurs 7 segments de la carte Cyclone V GX utilisent des entrées HEXn sur 7 bits (1 bit par segment) ; les segments sont notés a,b,c,d,e,f,g, avec a le bit de poids faible du mot HEXn et g le bit de poids fort. Chaque bit représente l'état allumé/éteint du segment, en logique inverse (1 : éteint, 0 : allumé)

```

      a
+----+
f|    |b
+-g-+
e|    |c
+----+
      d

```

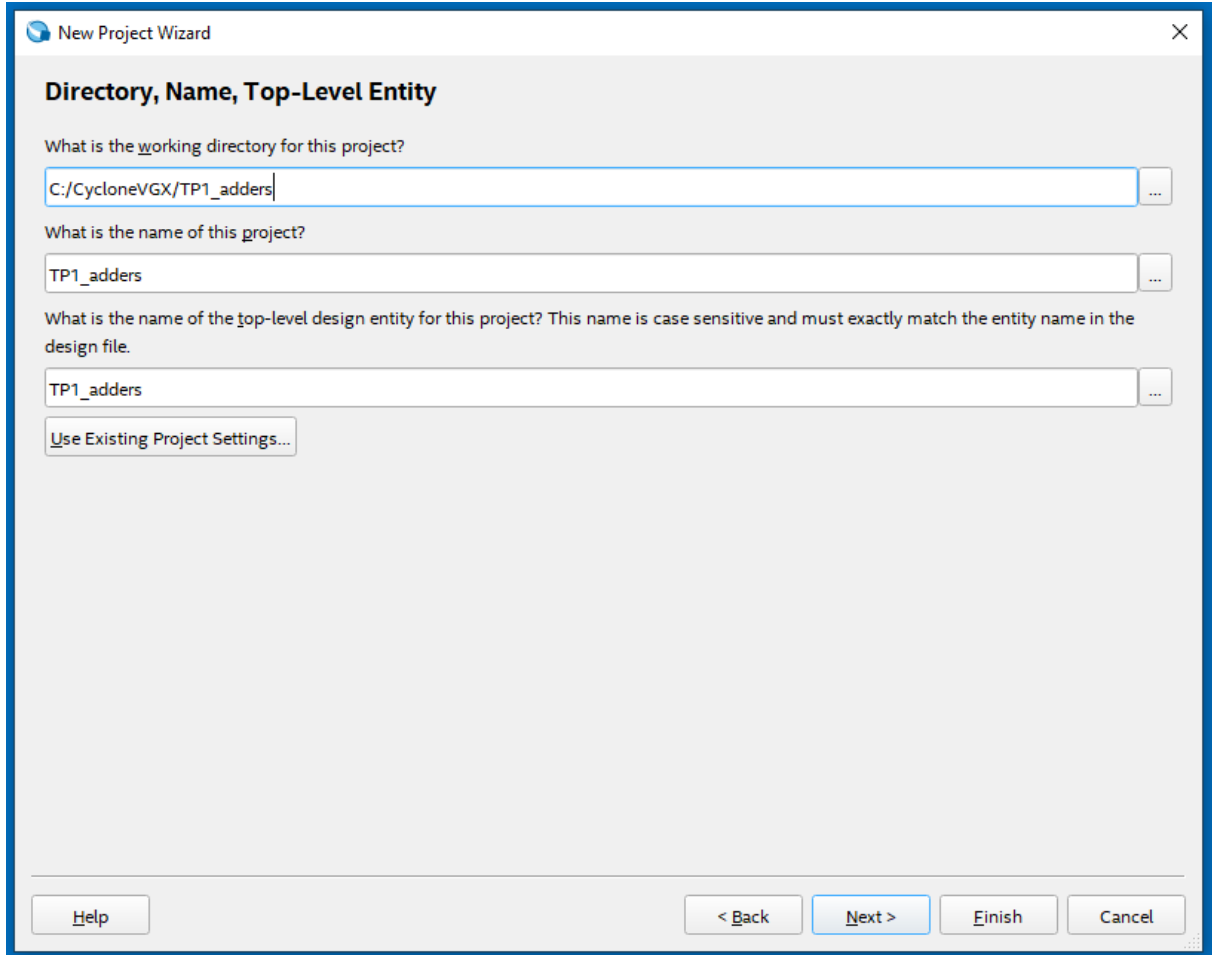
HEXn <= "0100100" -- Représentation de « 2 »

gfedcba

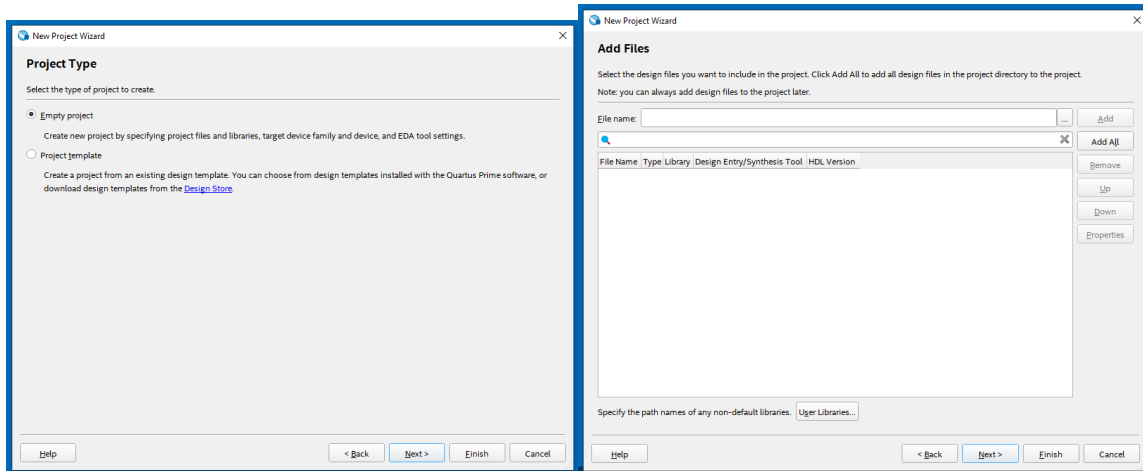
Annexes

Créer un nouveau projet sous Quartus Prime :

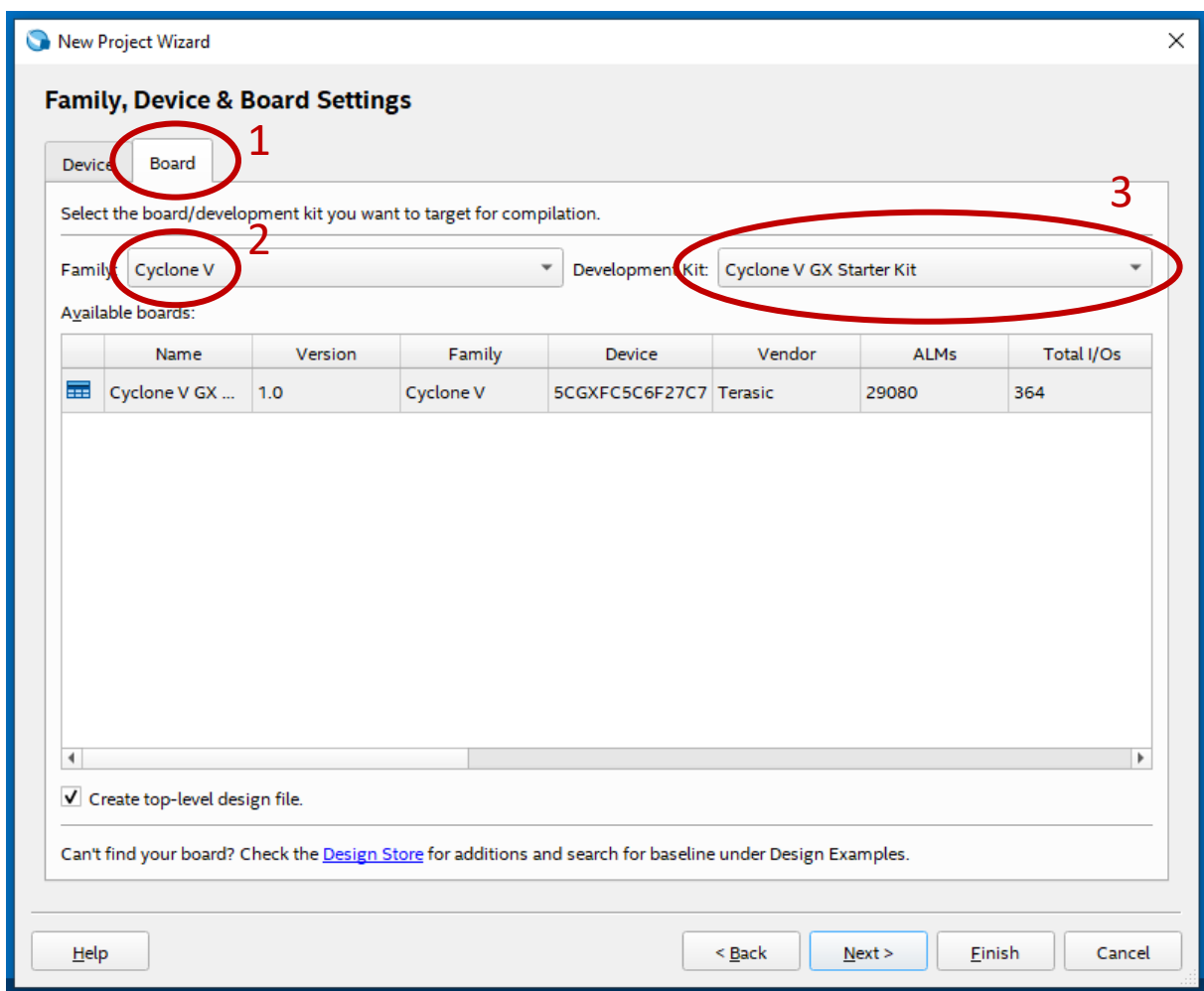
Allez dans File > New Project Wizard... et nommez adéquatement votre nouveau projet.
Cliquez sur Next.



Poursuivez avec les fenêtres «**Project Type** » et «**Add Files** » sans rien modifier.
Cliquez sur Next.



Dans la fenêtre **Family, Device & Board**, sélectionnez Board, puis la famille Cyclone V, puis le kit Cyclone V GX Starter Kit, qui correspond à la carte de développement qui sera utilisée par la suite. Cliquez sur Next.



Enfin, dans **EDA Tool Settings**, renseignez les champs Simulation :

Tool : Questa Intel FPGA et Format : VHDL

Cliquez sur Next, puis terminez.

New Project Wizard

EDA Tool Settings

Specify the other EDA tools used with the Quartus Prime software to develop your project.

EDA tools:

Tool Type	Tool Name	Format(s)	Run Tool Automatically
Design Entry/Synthesis	<None>	<None>	<input type="checkbox"/> Run this tool automatically to synthesize the current design
Simulation	Questa Intel FPGA	VHDL	<input type="checkbox"/> Run gate-level simulation automatically after compilation
Board-Level	Timing	<None>	
	Symbol	<None>	
	Signal Integrity	<None>	
	Boundary Scan	<None>	

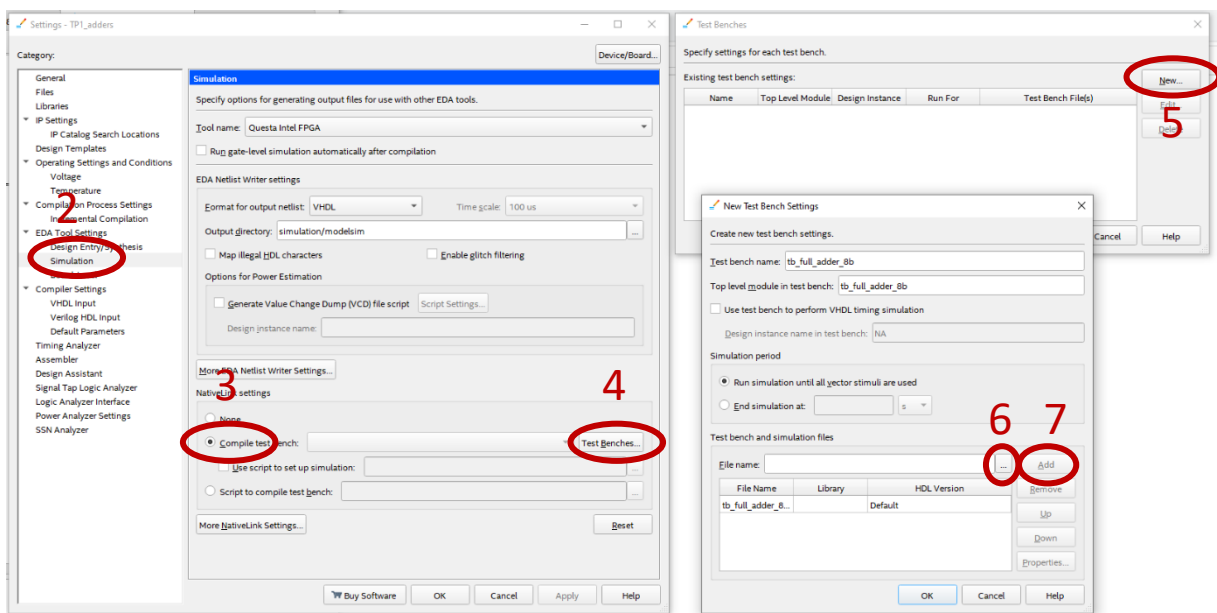
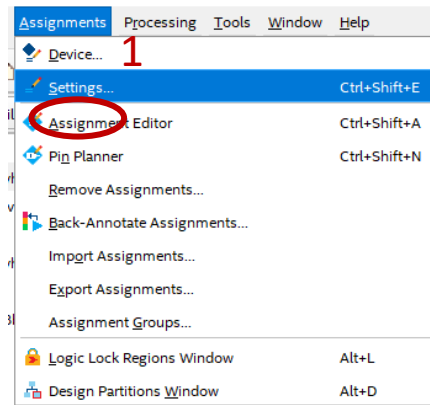
Help < Back Next > Finish Cancel

Simuler un testbench

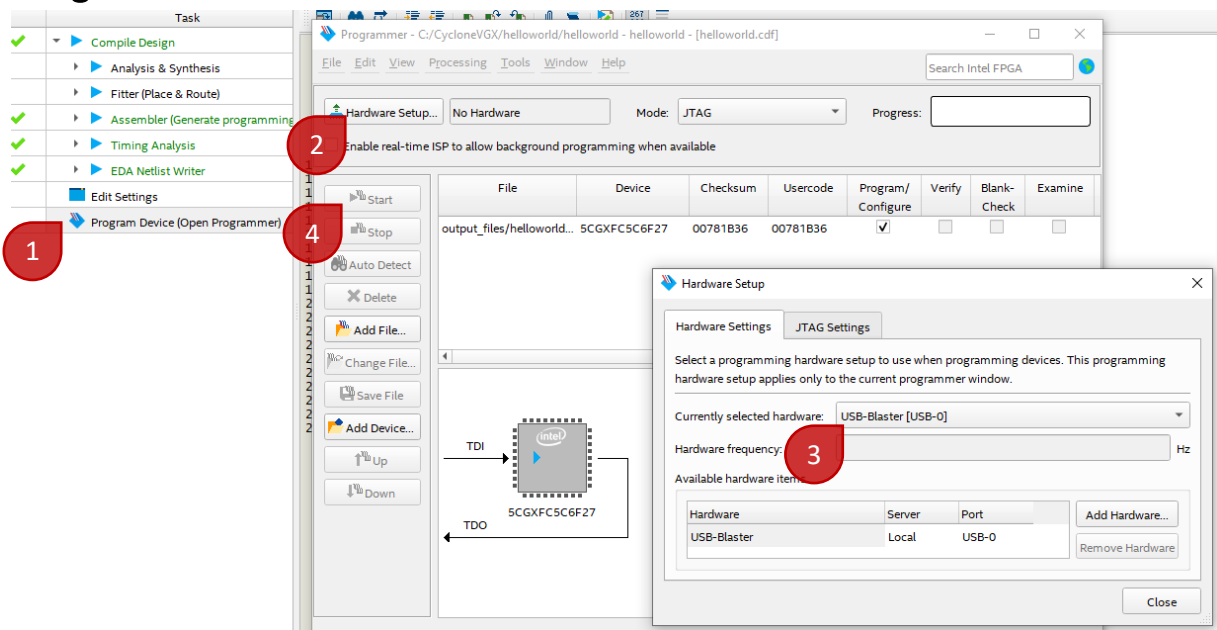
Lorsque vous avez créé un testbench, suivez la procédure suivante pour le simuler :

1. Allez dans Assignments > Settings ...
2. Cliquez sur Simulation
3. Cliquez sur Compile testbench
4. Ajoutez le fichier de testbench en cliquant sur Test Benches
5. Cliquez sur New ...
6. Nommez le testbench à ajouter, puis ajoutez le fichier .vhd correspondant en cliquant sur « ... »
7. Validez l'ajout du fichier vhd en cliquant sur « Add », puis validez toutes les fenêtres.

Une fois cette procédure suivie, vous pourrez lancer une simulation de votre module top-level en allant dans Tools > Run Simulation Tool > ...



Programmation de la carte



Si vous ne trouvez pas l'USB Blaster, référez-vous au document « Annexe Quartus Prime » sur junia-learning.