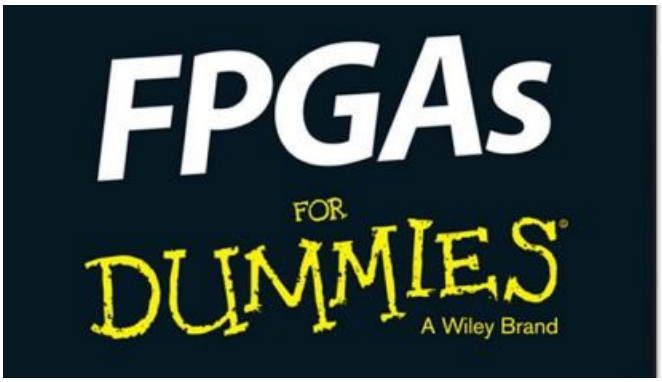
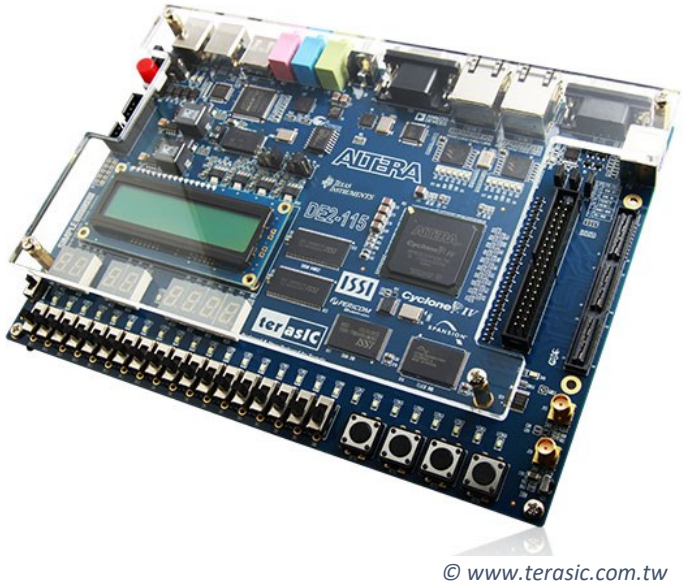


© www.intel.fr

TP FPGA

(Field Programmable Gate Array)



© www.terasic.com.tw

ISEN
ALL IS DIGITAL!
LILLE

yncréa

Novembre 2017

Sommaire

1 – Introduction

L'objectif de ce TP est d'étudier le fonctionnement d'un composant FPGA grâce à un environnement de développement basé sur une carte d'évaluation **DE2-115** et le logiciel de conception **QUARTUS II**. De plus, le logiciel **MODELSIM-ALTERA** sera utilisé pour la simulation des fonctions logiques.

Le TP abordera la conception de fonctions logiques combinatoires et séquentielles et permettra de valider les notions suivantes :

- La création de schémas électriques en utilisant des composants logiques issus de bibliothèques standards,
- La simulation comportementale des circuits réalisés, et la validation ou non des comportements attendus,
- La gestion de l'interface entre les entrées/sorties des composants virtuels et les entrées/sorties physiques du FPGA,
- La validation de la démarche par test sur la carte d'évaluation.

Le TP est à réaliser par binômes sur deux séances de 4 heures environ. Après la fin du TP, un compte-rendu reprenant l'ensemble des démarches effectuées durant le TP (interprétation de l'énoncé, conception des schémas, tests en simulation et éventuellement sur carte, validation de la démarche) devra être rédigé par chaque binôme et envoyé aux moniteurs. Le travail fera l'objet d'une notation aussi bien sur l'avancement du TP que sur la qualité et la clarté du compte-rendu.

2 – Carte d'évaluation

La carte d'évaluation utilisée dans ce TP est la carte **DE2-115** (développée par l'entreprise **terasIC**). Elle est basée sur un FPGA de la famille **Cyclone IV** de chez **ALTERA** (récemment devenu **Intel-FPGA**). La référence exacte du FPGA monté sur la carte d'évaluation utilisée est **EP4CE115F29C7**.

La carte d'évaluation contient également de périphériques d'entrée/sortie variés : boutons poussoir, commutateurs, LEDs, afficheurs 7 segments, codec audio, etc. Ces derniers sont déjà connectés physiquement au FPGA de manière fixe. Pour les utiliser, il faut donc correctement configurer les signaux d'entrée/sortie sur le FPGA de sorte qu'ils soient connectés aux périphériques que l'on veut utiliser.

À cet effet, deux documents sont mis à votre disposition :

- La datasheet de la carte d'évaluation,
- La liste de correspondance entre chaque périphérique de la carte d'évaluation et les entrées/sorties correspondantes du FPGA à configurer. Cette liste est au format .csv et s'ouvre avec un éditeur de texte ou un tableur.

3 – Exercices : réalisation d'un séquenceur

Le but de cette série d'exercices est d'utiliser un FPGA de la famille Cyclone IV ALTERA pour réaliser une séquence qui sera visualisée sur un afficheur à LEDs 7 segments. La Figure 1 montre les composants utilisés, et la Figure 2 la séquence à réaliser. Les segments en périphérie de l'afficheur doivent s'allumer les uns après les autres, un à la fois et dans le sens des aiguilles d'une montre.

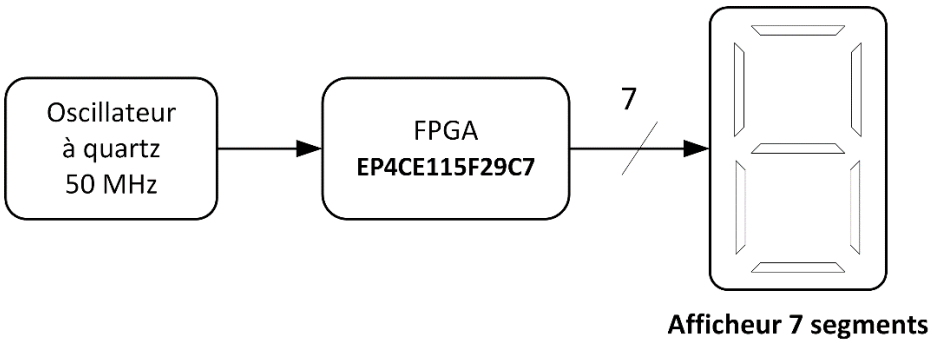


Figure 1 : Schéma du matériel utilisé

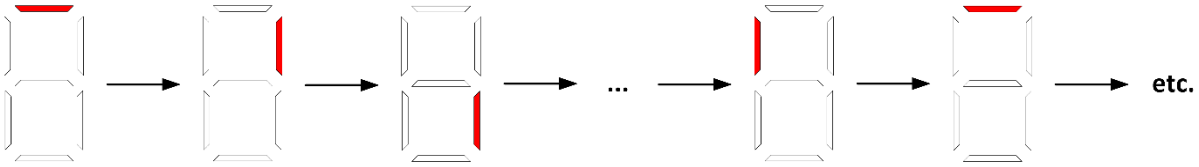


Figure 2 : Séquence à réaliser

La séquence se complexifiera au fur et à mesure des exercices en ajoutant des périphériques à utiliser (bouton poussoir), ou des fonctions (changement du sens de rotation).

La totalité des exercices sera réalisée dans un seul projet, qui sera enrichi au fur et à mesure de l'avancement dans le TP.

3.1 Synthèse d'un compteur modulo 6

Une séquence complète comprend 6 états successifs en sortie. Pour cadencer le système, il faut donc dans un premier temps réaliser un compteur avec un cycle comportant 6 valeurs distinctes. Ceci est obtenu avec un compteur 3 bits (**sans modulo intégré !**) se réinitialisant après que sa sortie ait atteint la valeur 5 (cycle en sortie égal à 0, 1, 2, 3, 4, 5, 0, 1, etc.).

Le schéma de principe d'un tel compteur est donné à la Figure 3.

Décrire et simuler ce compteur sous QUARTUS. Créer un nouveau schéma (avec ***File*** → ***New…*** → ***Block Diagram/Schematic File***) et utiliser les composants suivants :

- compteur → *lpm_counter*
- comparateur → *lpm_compare*
- constante ('101') → *lpm_constant* (si pas intégrée dans le bloc comparateur)
- V_{CC} → *vcc*

Remarque : Le fichier du schéma doit être ajouté au projet à l'enregistrement.

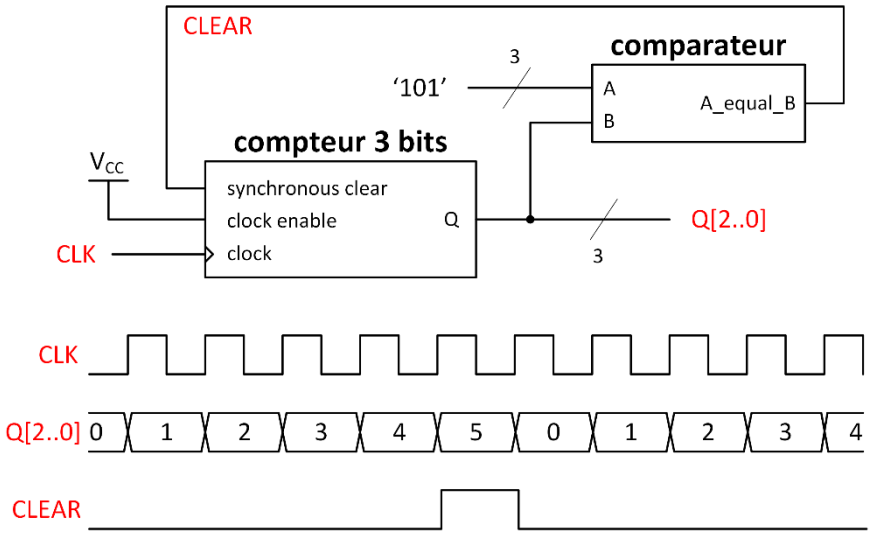


Figure 3 : Schéma de principe du compteur modulo 6

3.2 Synthèse d'un décodeur

La sortie **Q[2..0]** du compteur doit maintenant attaquer un décodeur 3 vers 8 qui permettra de sélectionner le segment d'afficheur à allumer pour chaque état du compteur réalisé ci-dessus. Le principe est donné à la Figure 4 qui indique également les références des segments de l'afficheur dans la liste de correspondance des entrées/sorties des périphériques à configurer.

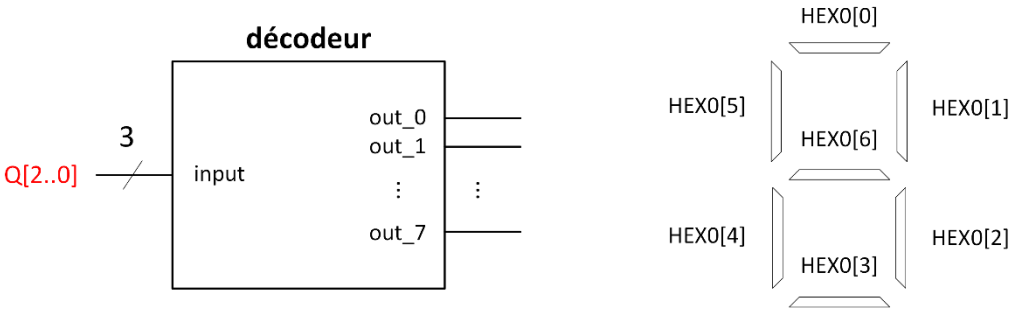


Figure 4 : Schéma de principe du décodeur 3 vers 8 (à gauche), brochage d'un afficheur 7 segments (à droite)

Réaliser ce décodeur et le relier aux sorties du FPGA. Utiliser ici aussi la librairie "*megafunctions*" :

- décodeur → *lpm_decode*

Remarques : **Les segments de l'afficheur sont allumés lorsque la sortie correspondante du FPGA est <u>à l'état bas</u>.**

Le brochage de la figure 4 est donné pour l'afficheur 'HEX0', mais les afficheurs 'HEX1' à 'HEX7' peuvent également être utilisés.

En mettant à la suite le décodeur et le compteur réalisé précédemment, vérifier par simulation que le circuit que vous avez conçu réalise bien la fonction voulue : à chaque incrémentation du compteur, la sortie activée doit se décaler.

3.3 Synthèse d'un diviseur de fréquence

L'oscillateur implanté sur la carte fournit une horloge à 50 MHz. Si celle-ci était appliquée directement sur le compteur décrit ci-dessus, les changements sur l'afficheur 7 segments seraient trop rapides pour pouvoir vérifier visuellement le fonctionnement.

Ce qui suit va permettre de diviser la fréquence d'horloge appliquée en entrée du FPGA, afin de décaler à un rythme beaucoup plus lent les sorties reliées à l'afficheur.

Objectif : Effectuer un cycle complet d'allumage de l'afficheur en exactement une seconde.

Un cycle complet correspond à l'allumage successif de 6 segments, il faut donc décaler la sortie activée tous les $(1/6)^{ème}$ de seconde.

Ceci peut se faire en utilisant l'entrée *clock_enable* du compteur de la figure 3. Cette entrée permet de bloquer le compteur dans un état tant qu'elle est à l'état bas. Ainsi, au lieu de la maintenir comme au préalable à l'état haut, elle ne doit maintenant être activée que 6 fois par seconde, pendant une durée d'un seul coup d'horloge. A vous de concevoir le système qui remplit cette fonction.

La Figure 5 résume l'ensemble du circuit que vous devez avoir au final, ainsi que les chronogrammes des principaux signaux.

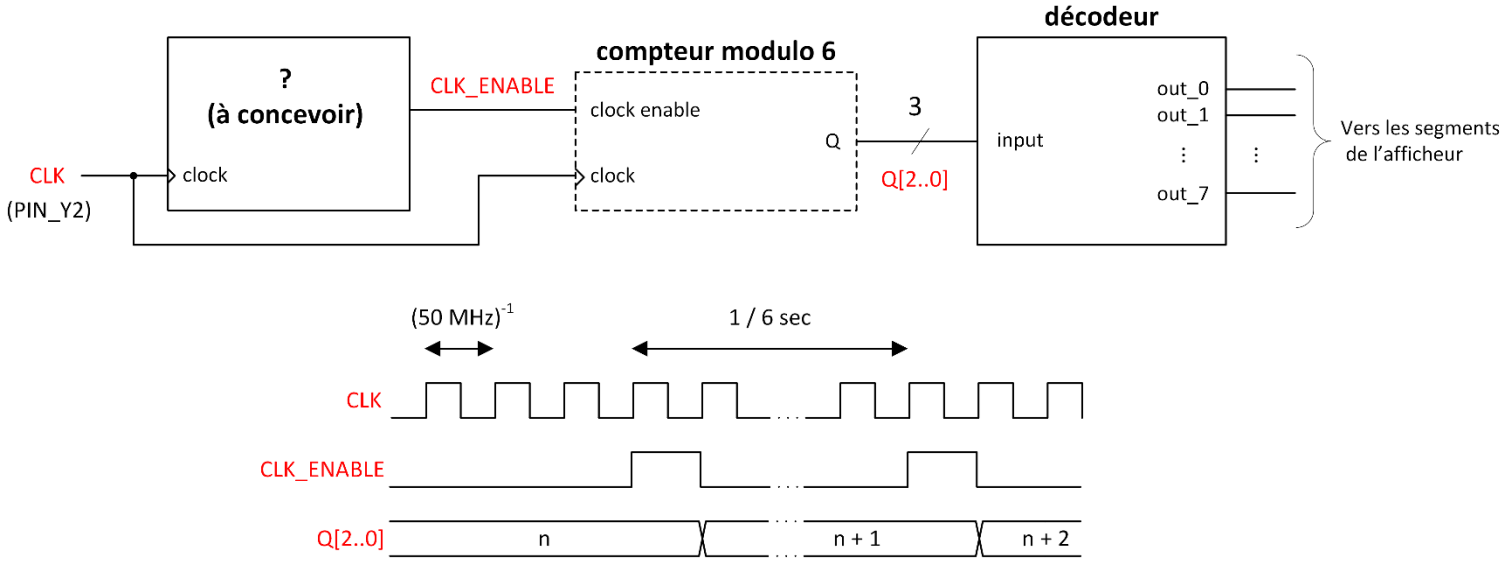


Figure 5 : Schéma et chronogramme du séquenceur complet

Après avoir décrit et synthétisé l'ensemble dans QUARTUS, validez le comportement en simulation tout d'abord. Pour simplifier la lecture et diminuer les temps de simulation, vous pourrez dans un premier temps générer une impulsion de CLK_ENABLE tous les 10 cycles de l'horloge CLK, par exemple. Une fois ceci validé, vous pourrez programmer le composant sur la carte de test avec votre vraie valeur de division de fréquence afin de vérifier le fonctionnement réel.

Remarque : Le signal CLK issu de l'oscillateur à quartz est appliqué sur la patte PIN_Y2 du FPGA (référence CLOCK_50 dans la liste des connexions des périphériques au FPGA).

3.4 Utilisation d'un bouton poussoir

Dans ce qui suit, on désire utiliser un bouton poussoir monté sur la carte de test pour modifier le sens de la séquence et obtenir ainsi un effet de rotation horaire ou anti-horaire du segment allumé. Une action sur le bouton doit inverser le sens de défilement.

Le bouton poussoir est noté KEY0 sur la carte d'évaluation du FPGA. Au repos, le signal sur cette patte est à l'état haut. Il passe à l'état bas lors d'un appui.

3.4.1 Utilisation du bouton seul

Dans un premier temps, ajouter sur le compteur modulo 6 une entrée *updown* qui permet selon son état de le faire compter ou décompter.

Une solution simple pour modifier cette entrée à chaque action du bouton poussoir est d'utiliser le schéma de la Figure 6.

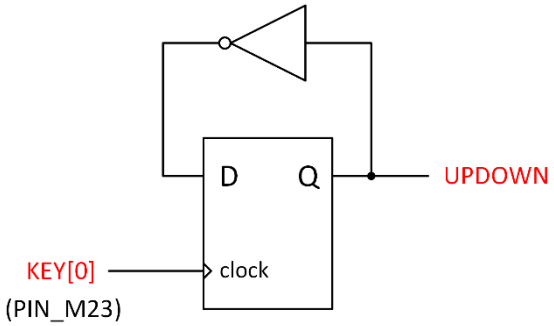


Figure 6 : Schéma de la mise en place la plus simple d'un bouton poussoir

Ajouter cette fonctionnalité à votre séquenceur. Préciser quel type de bascule utiliser et pourquoi. Programmer le composant et vérifier le résultat obtenu.

Tester le fonctionnement sur carte du bouton plusieurs fois. Vous pourrez observer dans certains cas le phénomène de « rebonds » lors d'un appui sur le bouton poussoir. Décrivez-le et proposez dans les grandes lignes un moyen pour résoudre ce problème, que l'on mettra en œuvre sous forme de machine d'états dans la partie suivante.

3.4.2 Système anti-rebond

Le schéma de la Figure 6 va être ici remplacé par une machine d'états afin de s'affranchir des problèmes de rebonds lors de l'ouverture ou la fermeture des contacts.

L'idée est de scruter périodiquement l'entrée reliée à bouton poussoir, à un rythme plus lent que les rebonds. La détection d'un passage d'un état haut à un état bas lors de cette scrutation entraînera un changement en sortie de la machine.

La Figure 7 décrit le principe utilisé, et la Figure 8 donne le diagramme d'état de la machine. Décrire brièvement pourquoi son comportement correspond au fonctionnement attendu.

Pour simplifier le système, on veut utiliser un signal déjà existant dans le système global pour donner la période de scrutation de l'entrée du bouton. Quel signal pourrait convenir ? Justifier.

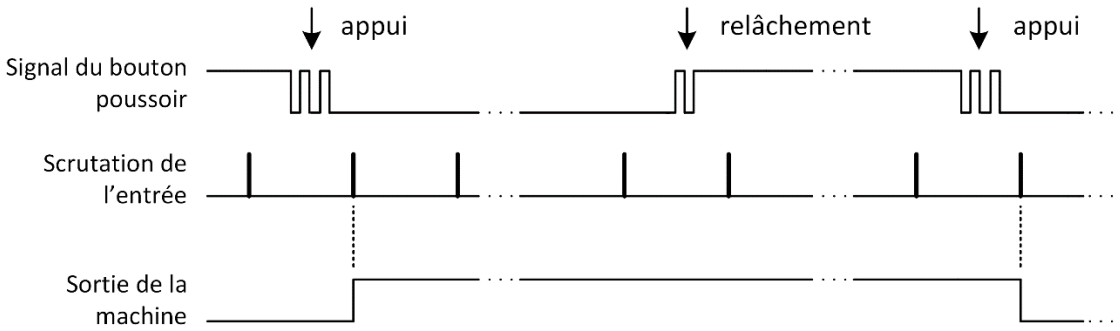


Figure 7 : Principe du système anti-rebond

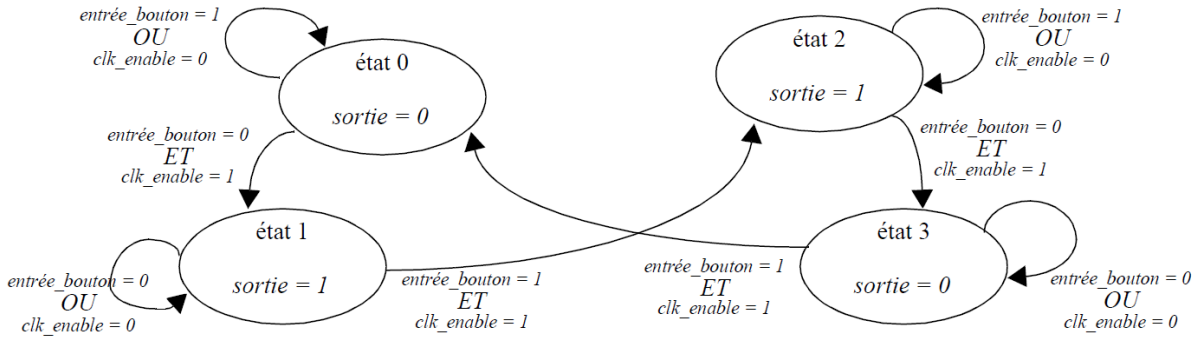


Figure 8 : Diagramme d'états de la machine d'états à mettre en œuvre

Synthétiser la machine ci-dessus. Elle comporte :

- deux entrées : *entrée_bouton* (en provenance de la patte M23),
  un signal de scrutation de l'entrée que l'on nommera *clk_enable*,
- une sortie : *sortie* (qui sera branchée sur le pin *updown* du compteur-décompteur),
- une horloge : utiliser l'horloge principale à 50 MHz.

Décrire la machine sous QUARTUS (dans un nouveau fichier de type ***State Machine File***, que vous associerez à votre projet) et l'exporter en passant par une description en langage VHDL. Vous pourrez remarquer la structure de ce fichier : à quoi correspondent les parties ***entity*** et ***architecture*** ?

Remarques : Lors de l'édition de la machine d'états, ne pas utiliser de caractères accentués, et le nom du fichier doit être différent des autres noms présents dans le projet.

Les transitions utilisent la syntaxe suivante pour les opérateurs logiques :

- A = 1 => A
- A = 0 => ~A (~ : *Alt Gr + 2*)
- A = 1 ET B = 1 => A & B (& : *Alt Gr + 1*)
- A = 1 OU B = 1 => A | B (| : *Alt Gr + 6*)

Créer ensuite un composant à partir de ce fichier, à intégrer dans l'ensemble du projet. Reprogrammer finalement le FPGA avec le système complet. Vérifier le fonctionnement.

3.5 Pour aller plus loin…

Vous pouvez vous inspirer de la structure du fichier VHDL généré automatiquement dans la partie précédente pour créer un composant « inverseur » à partir d'un nouveau fichier VHDL. Vous décrirez tout d'abord la partie ***entity*** du composant puis la partie ***architecture***.

Exporter ce composant dans un nouveau schéma, et le tester en simulation seulement.

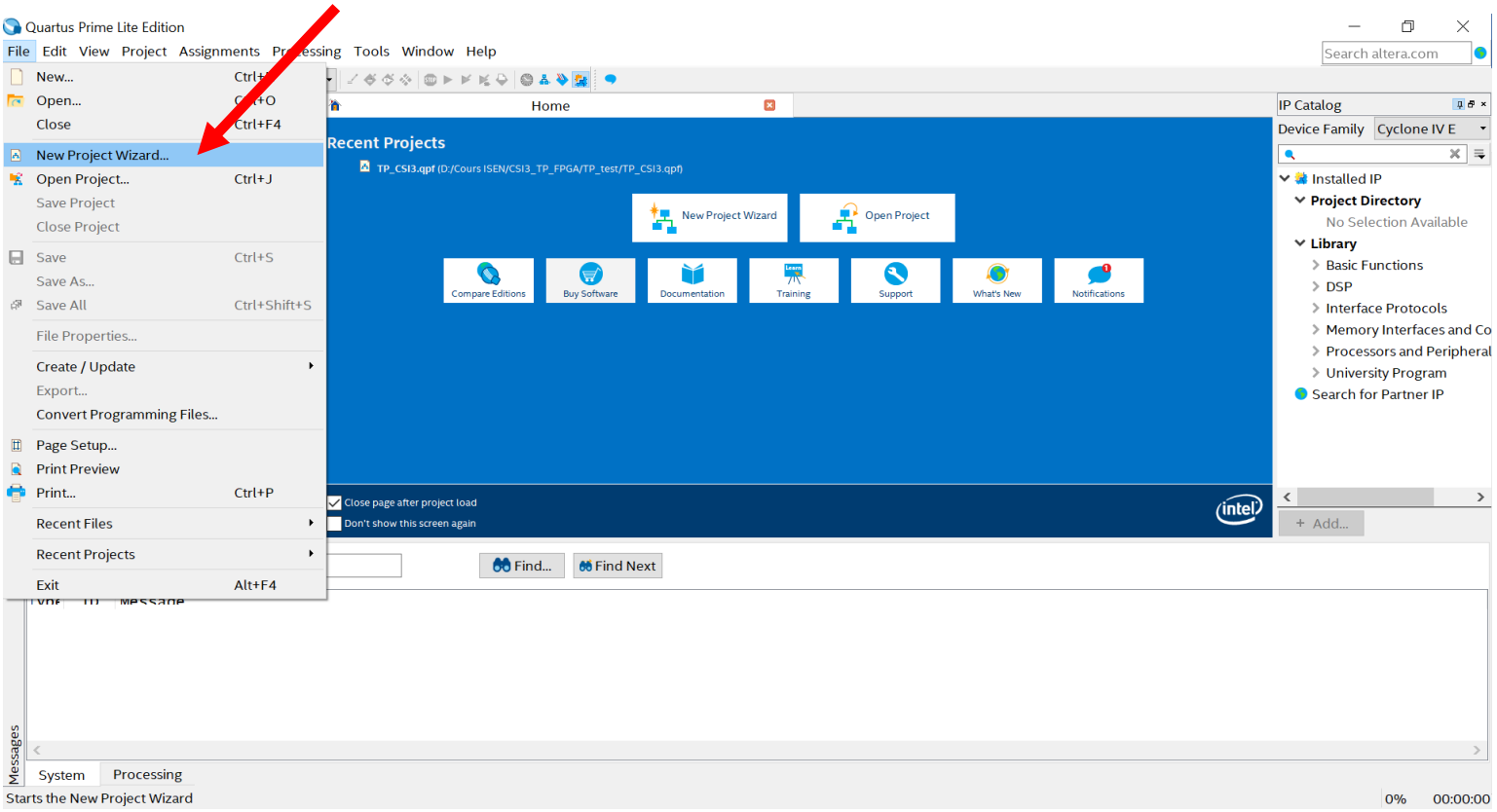
Vous pourrez ensuite essayer de réaliser, toujours en vous inspirant du composant généré automatiquement dans la partie précédente, un composant « compteur modulo 6 » à partir d'un nouveau fichier VHDL.

Exporter ce composant, et le tester à nouveau en simulation dans un schéma à part.

Annexes

1 – Création d'un projet sous QUARTUS

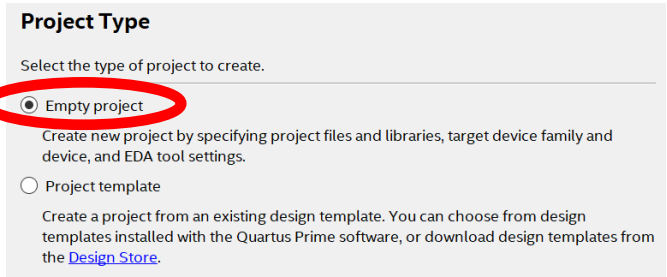
En démarrant QUARTUS, commencer par créer un nouveau projet.
File → New Project Wizard...



Renseigner l'emplacement du projet, ainsi que son nom. Vous remarquerez que le nom de la « top-level design entity » se remplit automatiquement comme étant identique au nom du projet.

Remarque : Pour les noms de projet, ne pas utiliser de caractères accentués.

Dans la fenêtre suivante, sélectionner « ***Empty project*** », puis valider.

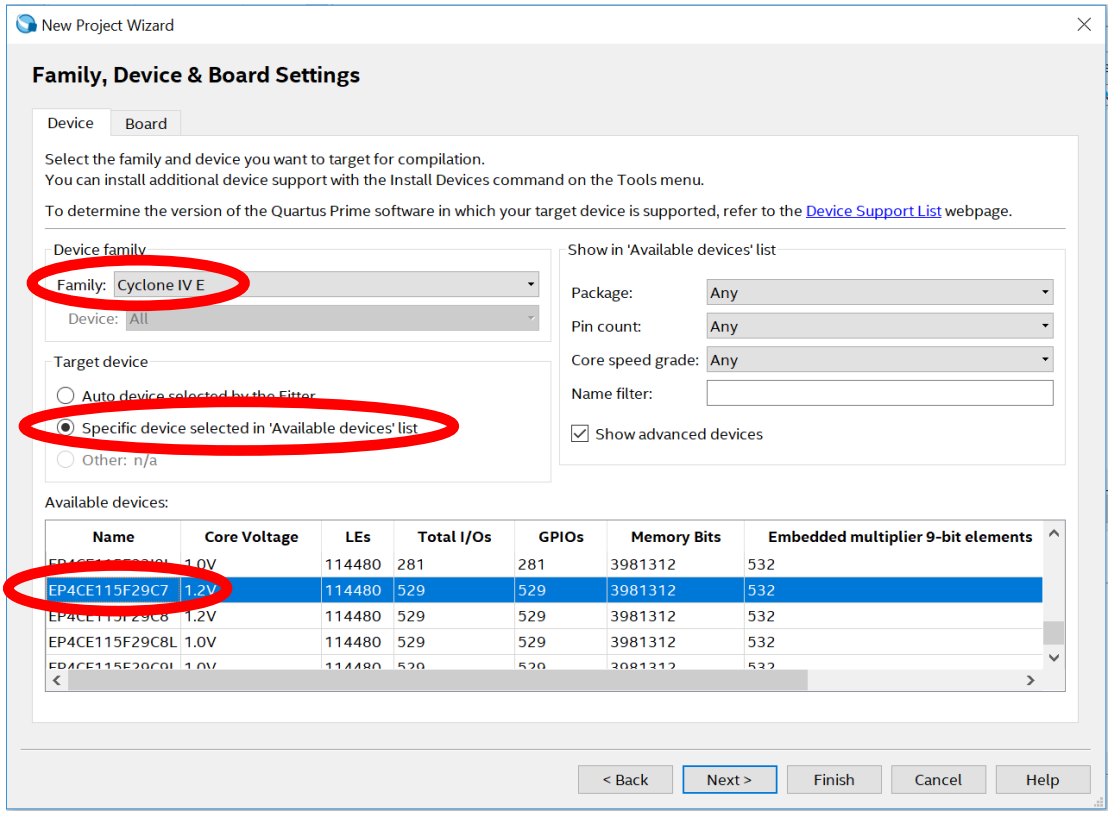


Ne pas ajouter de fichiers dans la fenêtre « ***Add files*** » et passer à l'étape suivante directement.

Dans l'étape « ***Family, Device & Board Settings*** », sélectionner :

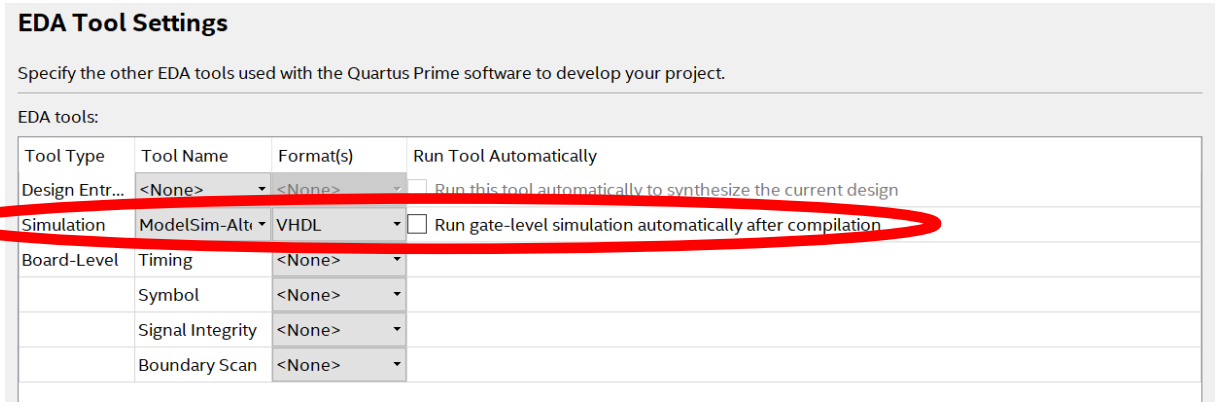
Family : ***Cyclone IV E***
Target device : ***Specific device selected in 'Available devices' list***
Available devices : ***EP4CE115F29C7***

Puis valider.



Dans l'étape « ***EDA Tools Settings*** », à la ligne « ***Simulation*** », sélectionner :

Tool Name : ***ModelSim-Altera***
Format(s) : ***VHDL***

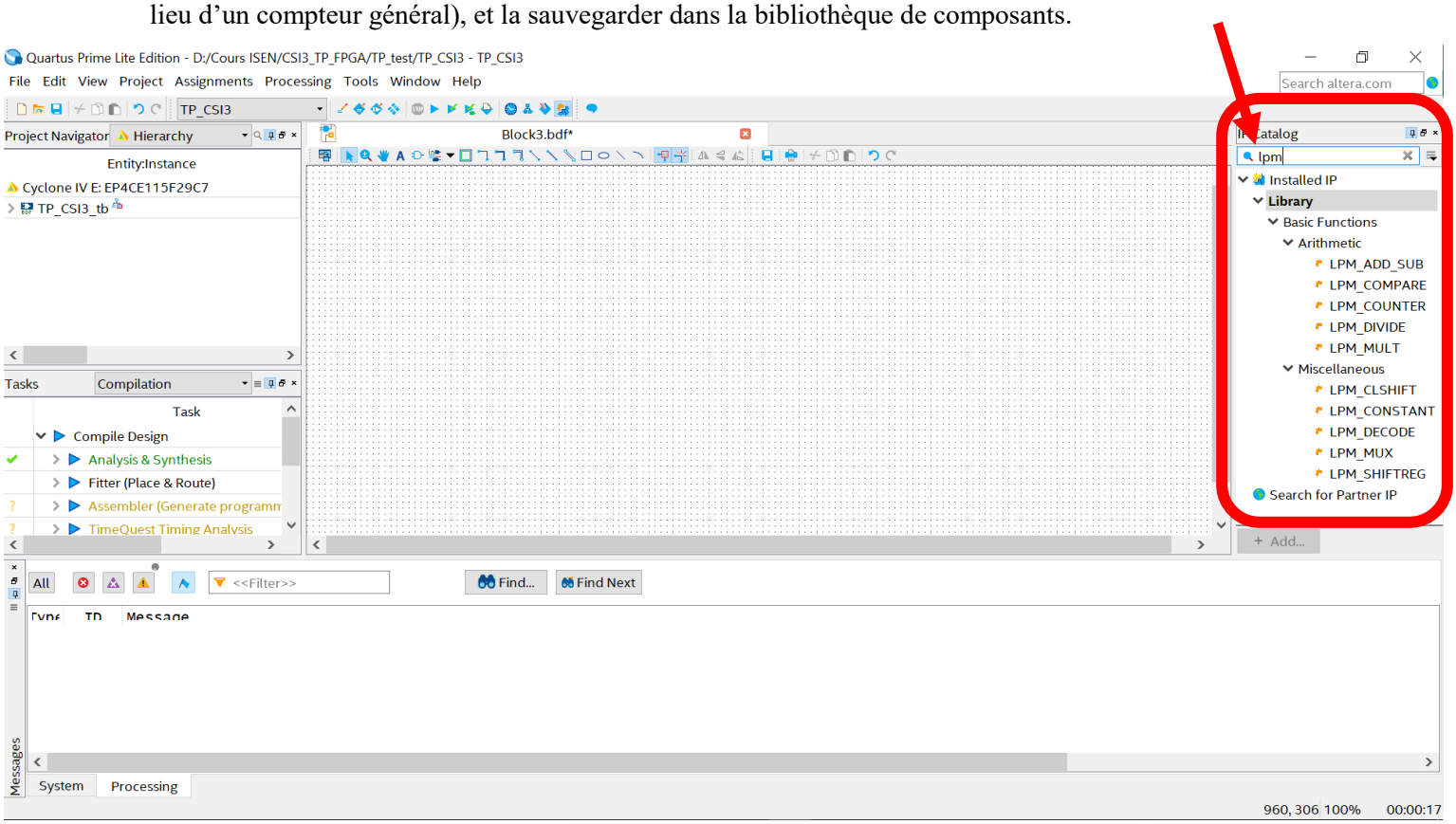


Puis valider et terminer la création de projet.

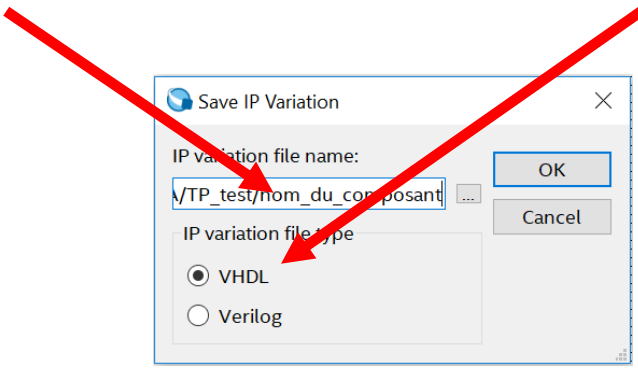
2 – Édition et vérification d'un schéma

À partir d'un fichier de schéma vierge, il faut créer des instances de composants que l'on pourra placer et connecter ensemble pour réaliser le montage désiré.

Pour les composants paramétrables (c.à.d. des composants standards dont on peut choisir quelles entrées/sorties utiliser), utiliser le catalogue de composants en recherchant le composant voulu dans la barre de recherche. Cela va générer une version personnalisée du composant (ex. compteur sur 3 bits au lieu d'un compteur général), et la sauvegarder dans la bibliothèque de composants.

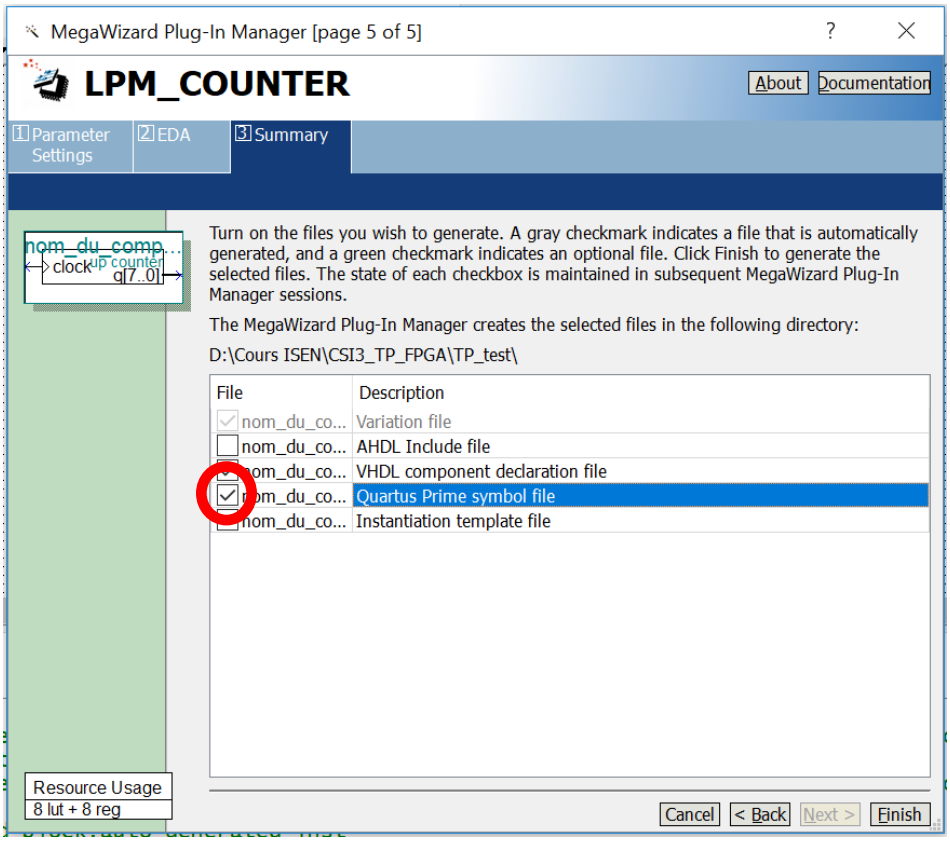


Cela va ouvrir la fenêtre de création de composant personnalisé. Rentrer le nom sous lequel va être enregistré le composant dans la bibliothèque, en sélectionnant l'option ***VHDL***.

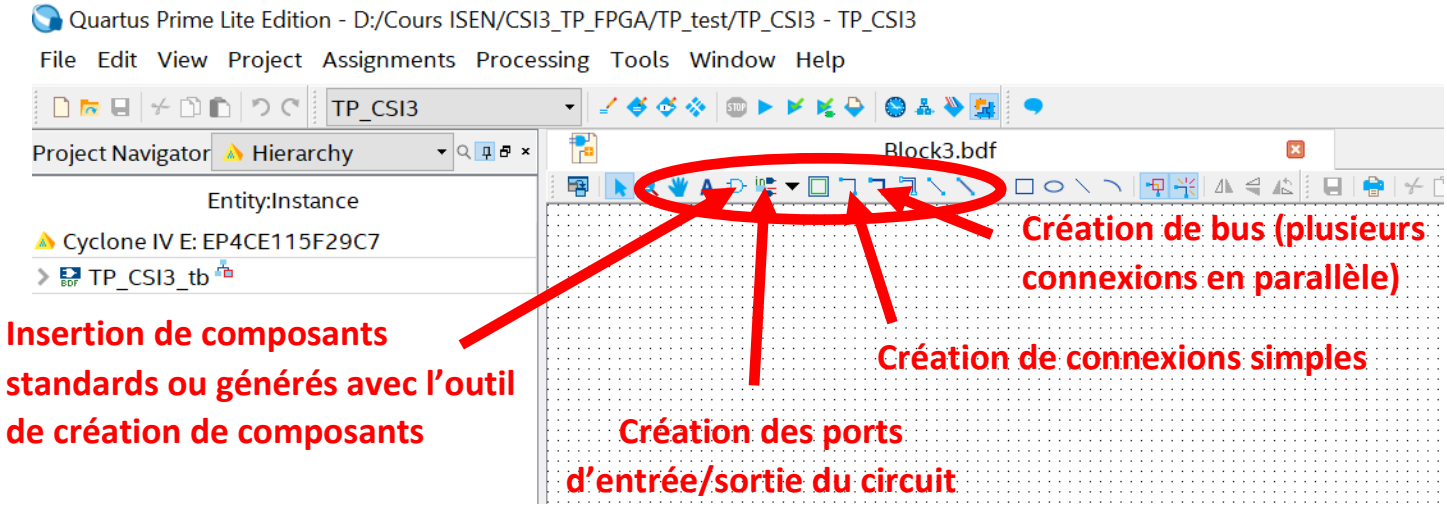


Créer ensuite le composant en suivant les étapes du générateur de composant et en sélectionnant les options et les fonctionnalités désirées.

ATTENTION : A la dernière étape de la création d'un composant, ne pas oublier de cocher la case « *Quartus Prime symbol file* ».



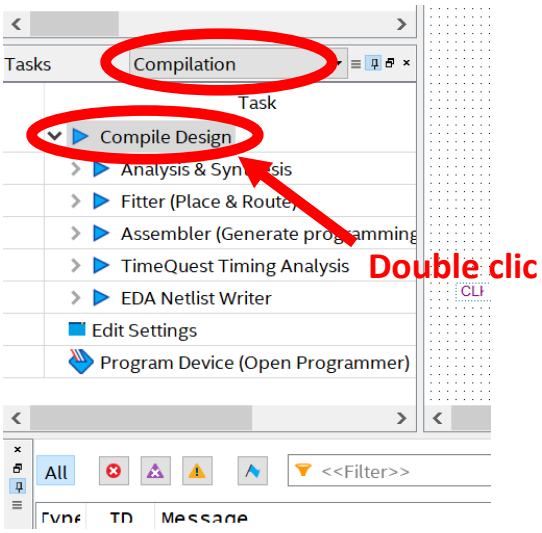
Ceci permettra de rendre le symbole du composant créé accessible depuis la barre d'outils de création d'un schéma électrique. Cette dernière présente les options suivantes :



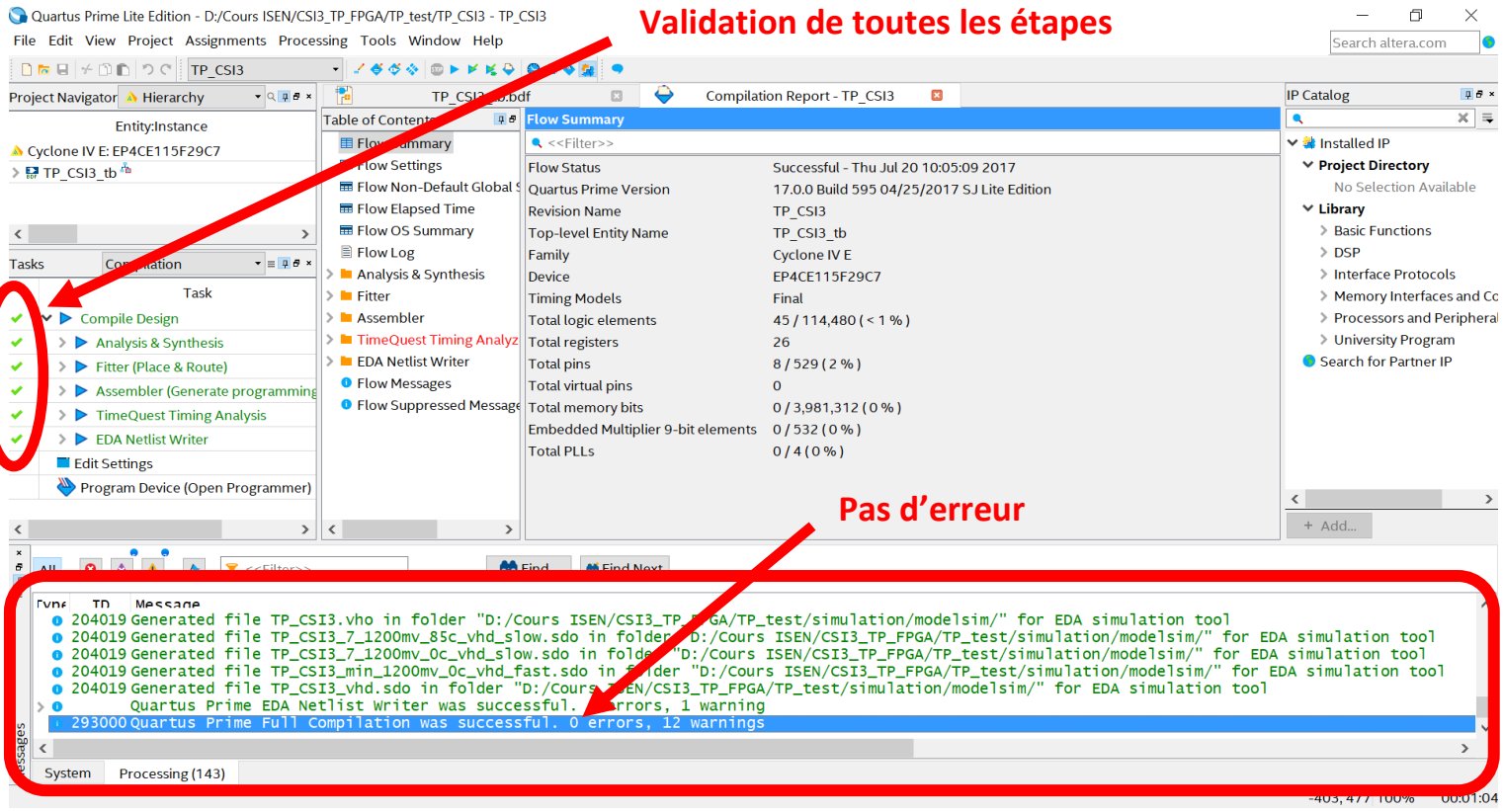
L'ensemble de ces commandes permet à l'utilisateur de réaliser le schéma d'un circuit complet.

Remarque : **Ne pas oublier de placer de matérialiser les entrées et sorties du circuit par des ports d'entrée/sortie !**

Une fois le schéma réalisé, il faut vérifier qu'il n'y a aucune erreur de synthèse. Pour ce faire, double-cliquer sur « ***Compile Design*** » dans la fenêtre ***Compilation***.



S'il n'y a pas d'erreur qui s'affiche dans la Console, on peut passer aux étapes suivantes de la conception.

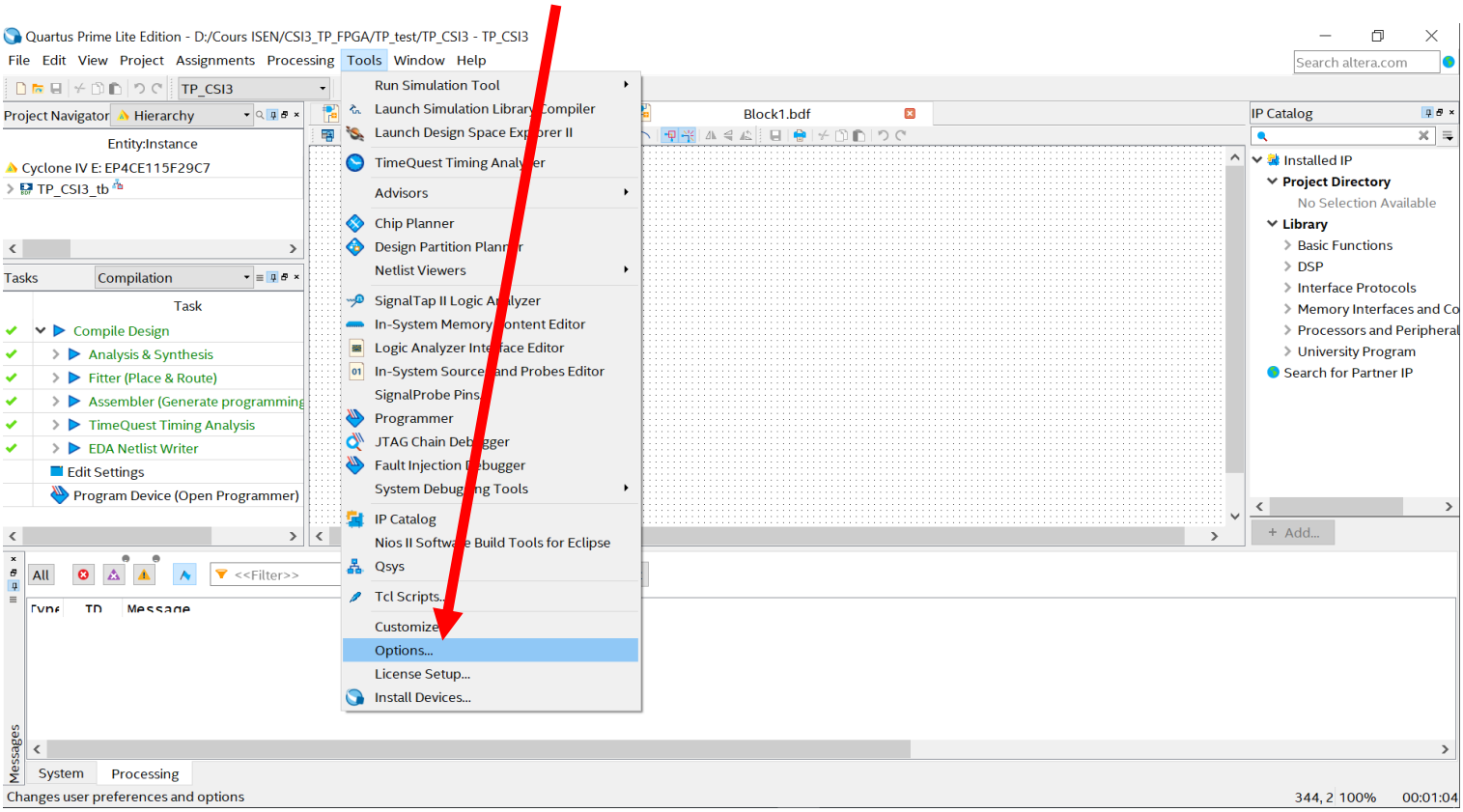


Remarque : **Si plusieurs fichiers sont présents dans le projet, s'assurer que c'est le bon fichier qui est compilé en faisant *Clic droit → Set as Top-Level Entity* sur le fichier à compiler.**

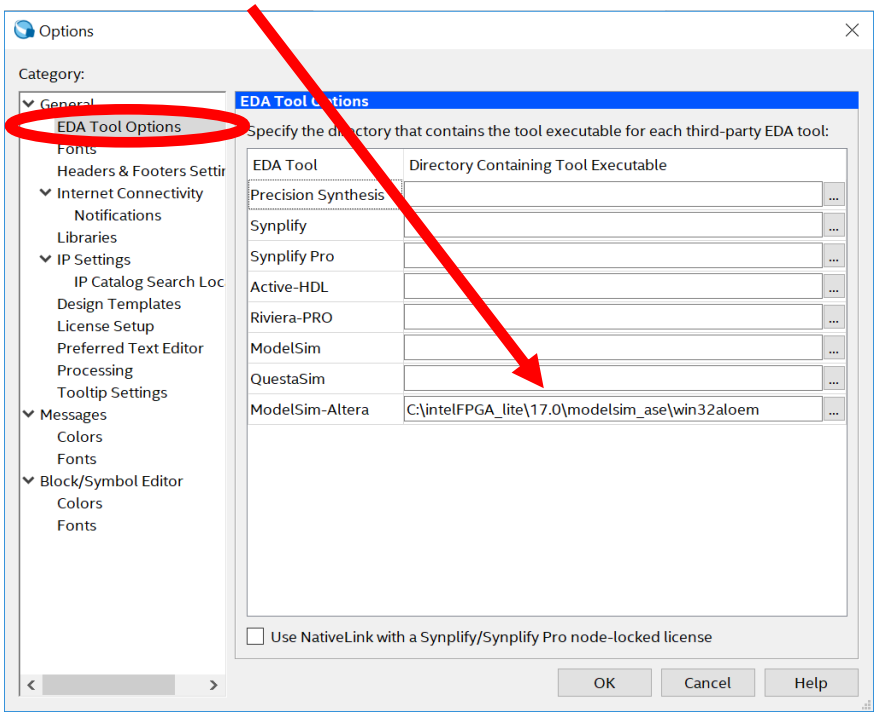
3 - Simulation d'un circuit avec ModelSim-Altera

Avant de commencer la simulation, il faut vérifier que le chemin vers le logiciel ModelSim-Altera est correct.

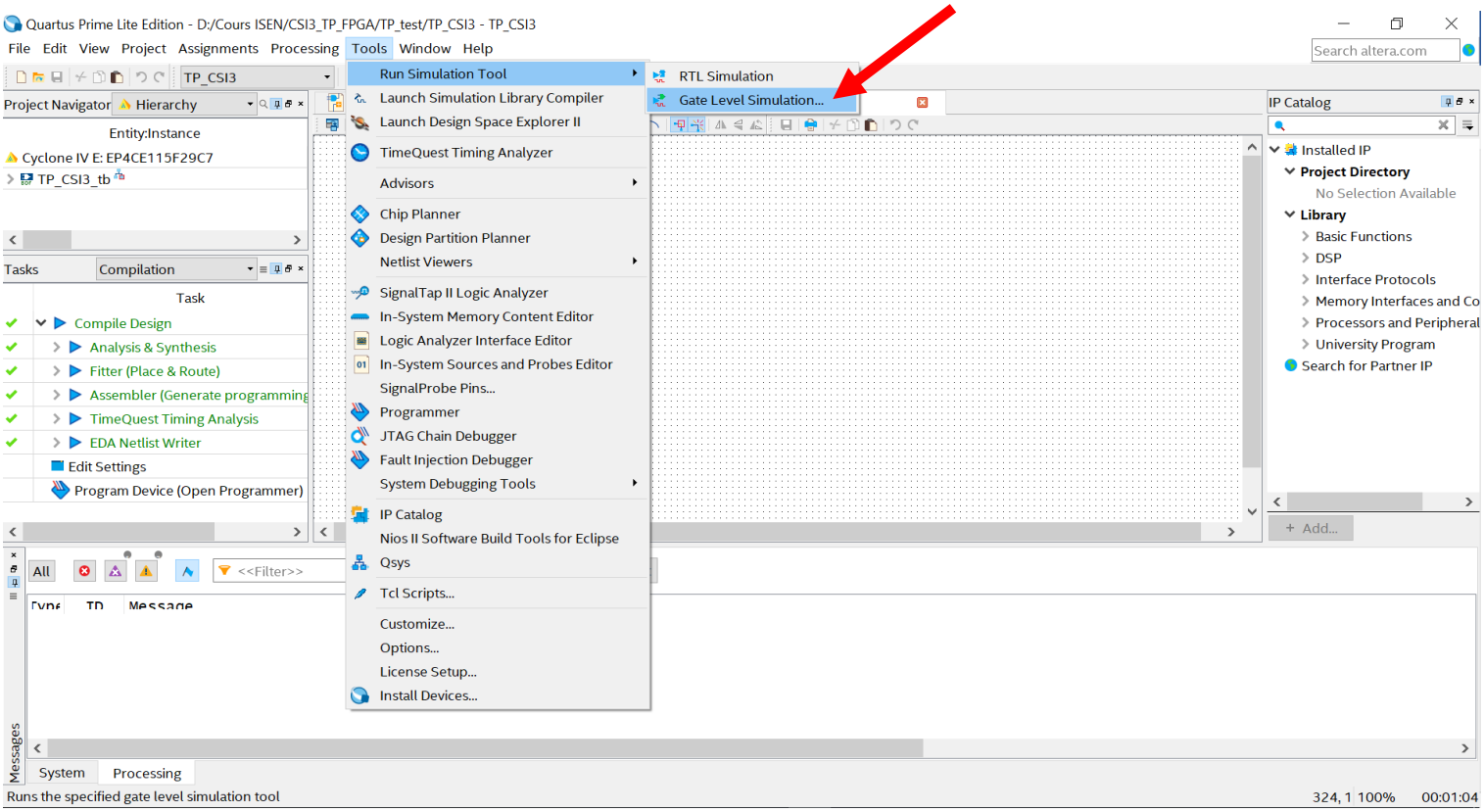
Pour cela, ouvrir le menu ***Tools → Options…***



Dans l'onglet ***EDA Tool Options***, vérifier que le chemin vers le logiciel ModelSim-Altera est correct et mène bien vers **…\modelsim_ase\win32aloem**

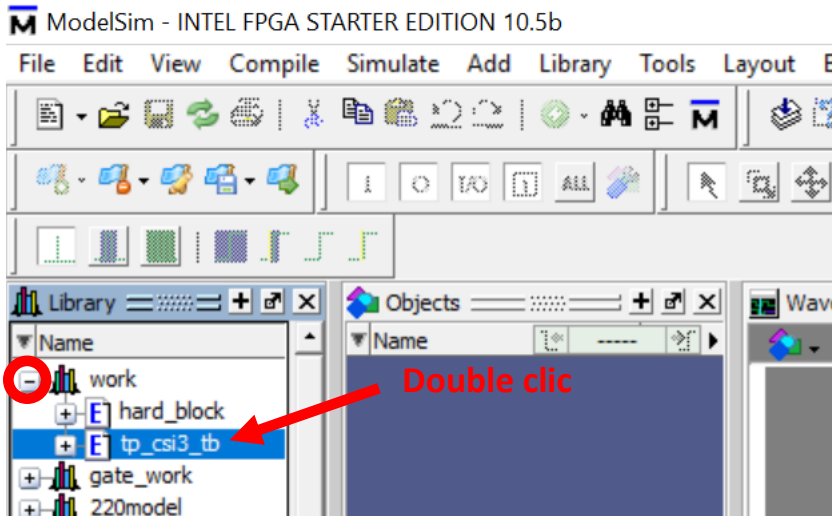


Lancer ensuite le simulateur dans ***Tools → Run Simulation Tool → Gate Level Simulation…***

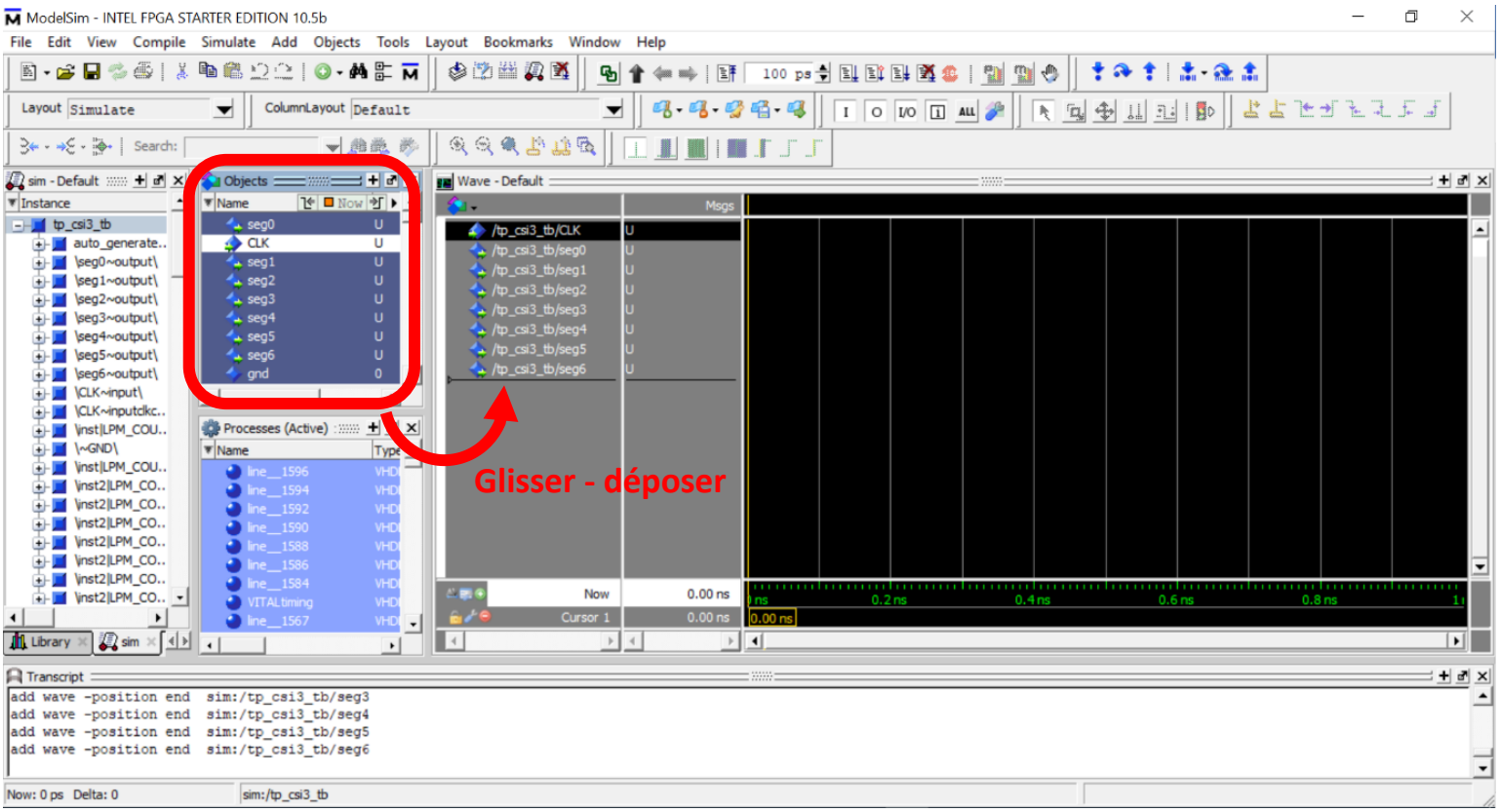


Valider directement la fenêtre qui apparaît, et ModelSim-Altera se lance au bout de quelques secondes.

Dans l'écran qui apparaît, dérouler l'onglet ***work*** dans la fenêtre ***Library***, puis double-cliquer sur le nom de votre schéma à simuler.

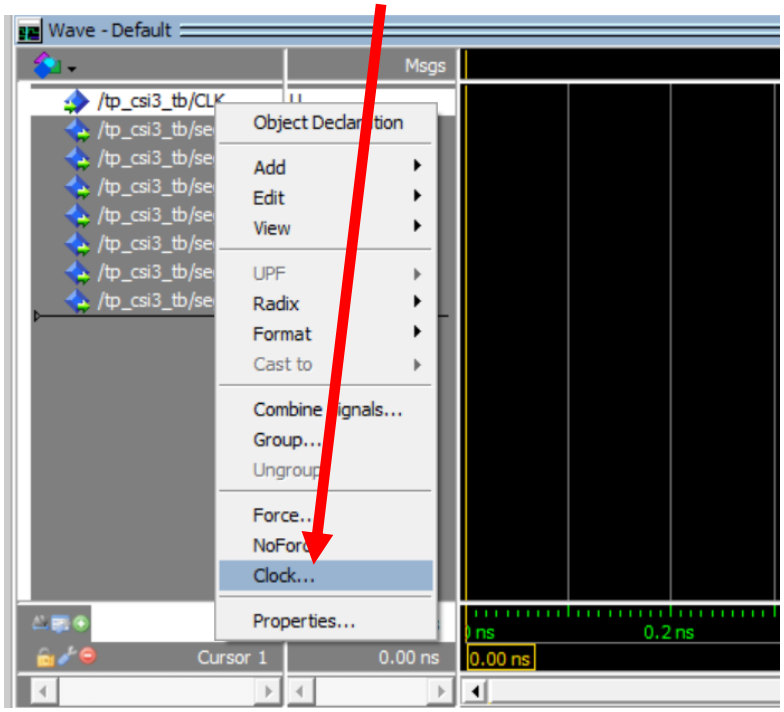


Les différents signaux présents dans le circuit à simuler apparaissent alors dans la fenêtre ***Objects***. Il suffit ensuite de faire glisser les signaux intéressants dans la fenêtre ***Wave - Default*** pour pouvoir visualiser leur comportement.

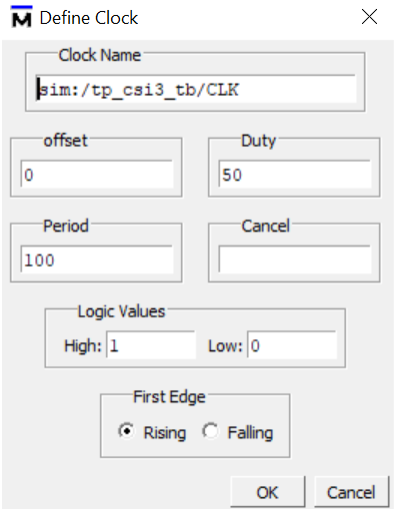


Il faut ensuite paramétrer <u>les signaux d'entrée uniquement</u> pour pouvoir visualiser la réponse des signaux de sortie.

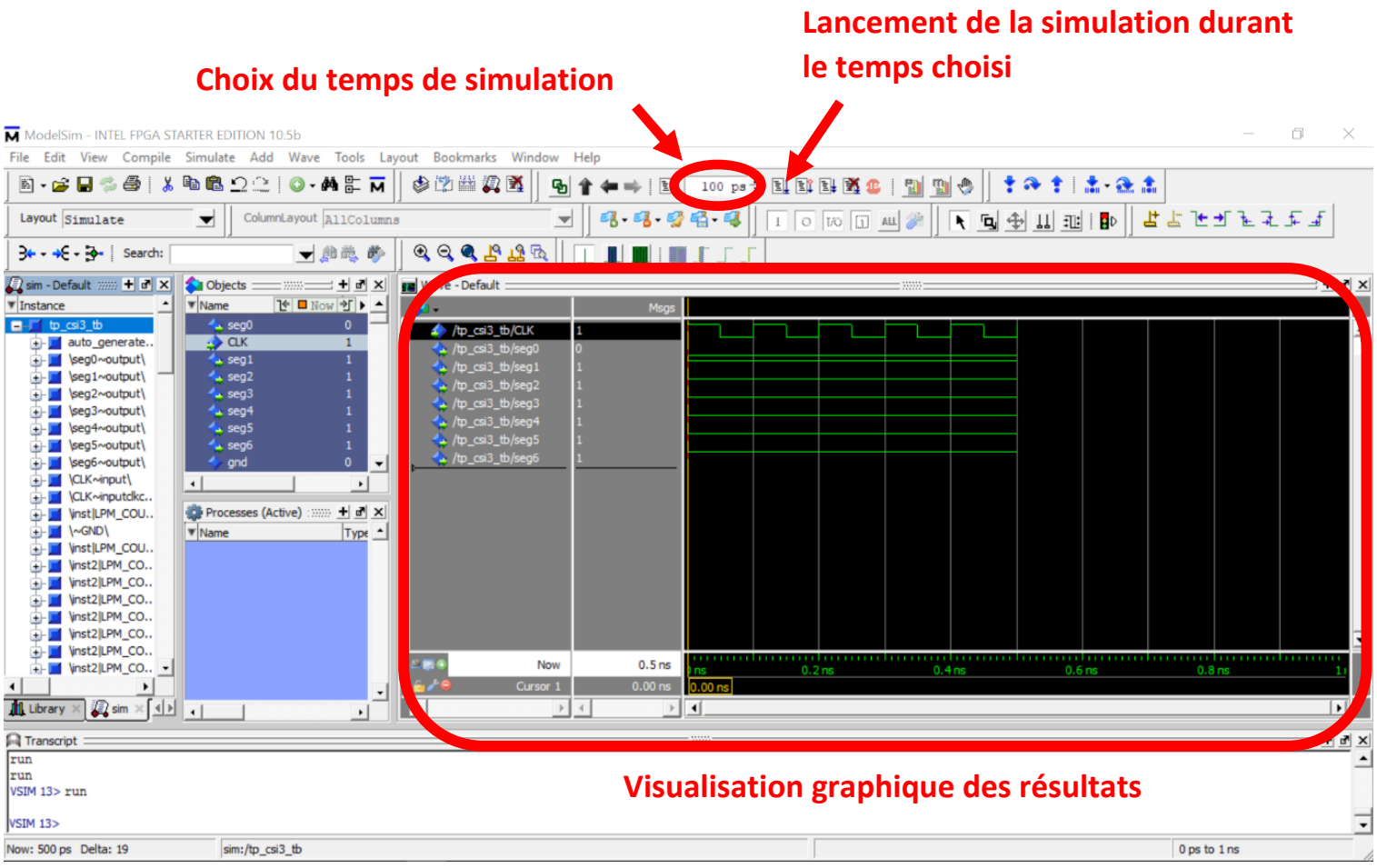
Pour appliquer des signaux simples sur les signaux d'entrée, comme un signal créneau, faire un Clic droit sur les signaux d'entrée et choisir l'option ***Clock…***



Dans la fenêtre qui apparaît alors, on peut paramétrer les différentes caractéristiques du signal : période, phase, rapport cyclique, etc.



Une fois tous les signaux d'entrée paramétrés (pas encore visibles sur le chronogramme), on peut finalement lancer la simulation et visualiser le comportement de l'ensemble des signaux.



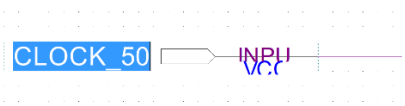
4 – Assignation des pins du FPGA aux entrées/sorties du circuit

Pour cette opération, deux méthodes sont possibles. Ces méthodes sont équivalentes, l'une ou l'autre peut donc être utilisée.

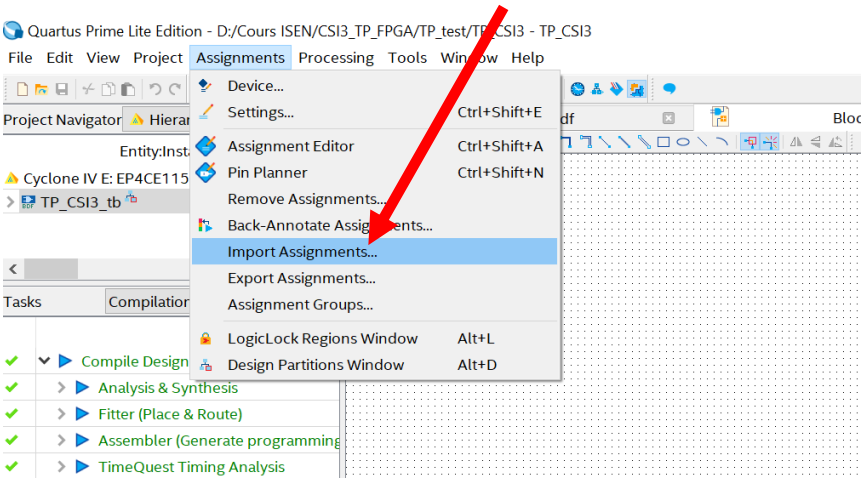
4.1 Affectation à partir du fichier *DE2_115_pin_assignments.csv*

Le fichier ***DE2_115_pin_assignments.csv*** contient les informations (ligne par ligne) sur la correspondance entre les dispositifs présents sur la carte DE2-115 et le FPGA. Chaque ligne est composée ainsi :

Pour faire le lien avec les entrées/sorties du schéma électrique, il faut indiquer à quel dispositif ces dernières correspondent. On va donc renommer les entrées/sorties du schéma avec les noms des dispositifs correspondant dans le fichier ***DE2_115_pin_assignments.csv***. Ainsi, par exemple, on va renommer l'entrée d'horloge sur le schéma « CLOCK_50 ».



Une fois cela fait pour toutes les entrées/sorties, importer le fichier d'affectation ***DE2_115_pin_assignments.csv*** grâce à ***Assignments → Import Assignments…*** :

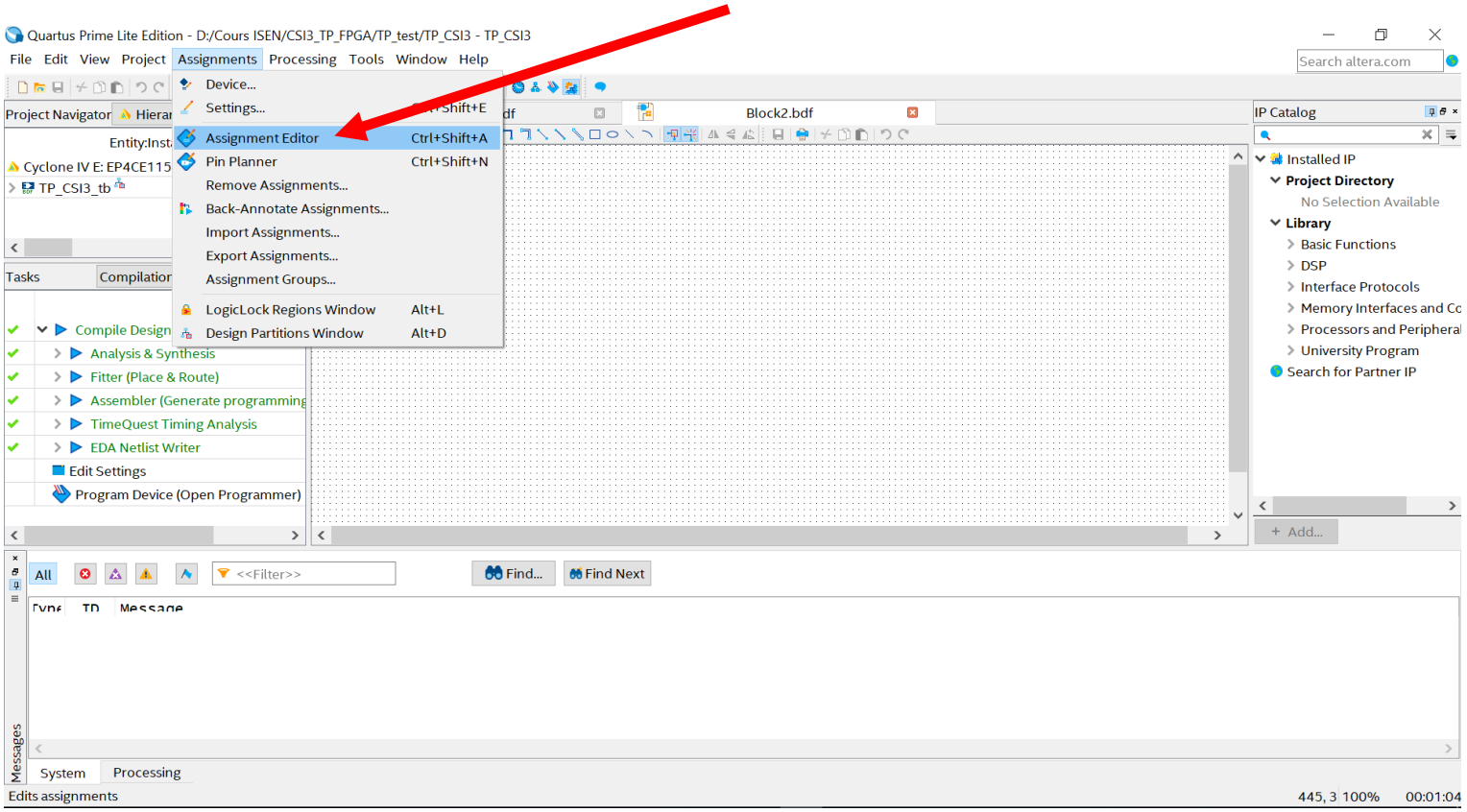


Rechercher l'emplacement du fichier ***DE2_115_pin_assignments.csv***, puis valider.

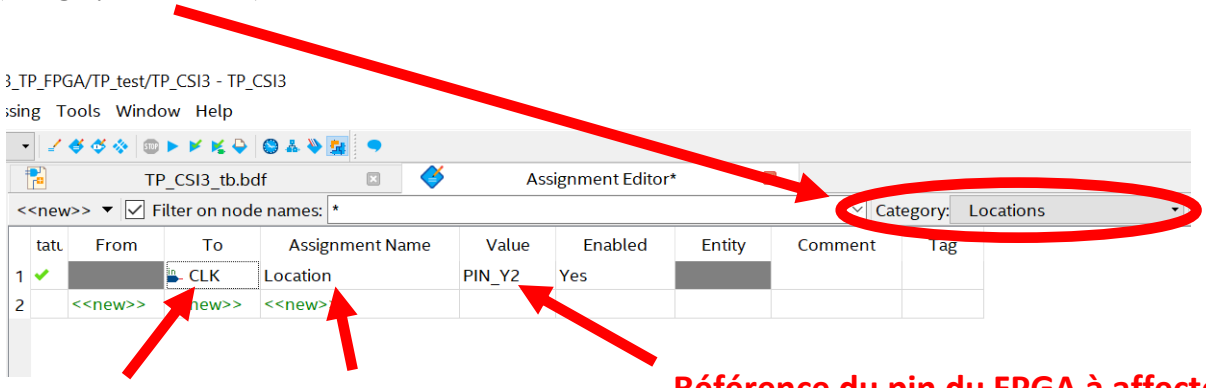
Sauvegarder et recompiler le design. Cela va générer de nombreux warnings, ce qui est normal étant donné que tous les dispositifs présents dans le fichier (et donc déclarés dans le projet) ne le sont pas forcément dans le schéma.

4.2 Utilisation de l'outil graphique

Il est possible de faire la correspondance entre entrées/sorties du schéma électrique et pins du FPGA en démarrant l'outil graphique : ***Assignments → Assignments Editor***



L'interface qui s'affiche permet d'affecter individuellement chaque entrée/sortie à un pin du FPGA (***Category*** : ***Locations***).



Nom (dans le schéma) de l'entrée/sortie à affecter

« Location » sert à indiquer une contrainte physique sur le FPGA

Référence du pin du FPGA à affecter, trouvable dans le fichier *DE2_115_pin_assignments.csv*

Une fois toutes les affectations déclarées, sauvegarder le fichier et recompiler le design.