TP1

PRISE EN MAIN DE LA CARTE FPGA DE1 ET DE L'ENVIRONNEMENT DE DEVELOPEMENT ALTERA QUARTUS II

Objectif :

L'objectif de ce TP est de prendre en main les outils de conception Quartus. Vous allez apprendre dans ce TP un flow de conception de type top-down, c'est-à-dire de la spécification à la synthèse de composants, en passant par la simulation.

Le TP est divisé en deux parties, la première porte sur les outils et la carte DE1, tandis que la deuxième partie, porte sur des exercices d'application qui seront notés

Partie I. Prise en main de Quartus II

I. Conception d'un nouveau Projet

Généralement, un système électronique est composé de plusieurs composants. Chaque composant est décrit, compilé et simulé indépendamment des autres avant d'être relié aux autres. Pour cela, il est nécessaire de créer un projet.

a. Lancer Quartus II dans le menu Démarrer\Altera\Quartus II. La fenêtre suivante apparaît:

Nom Prénom :
Groupe :

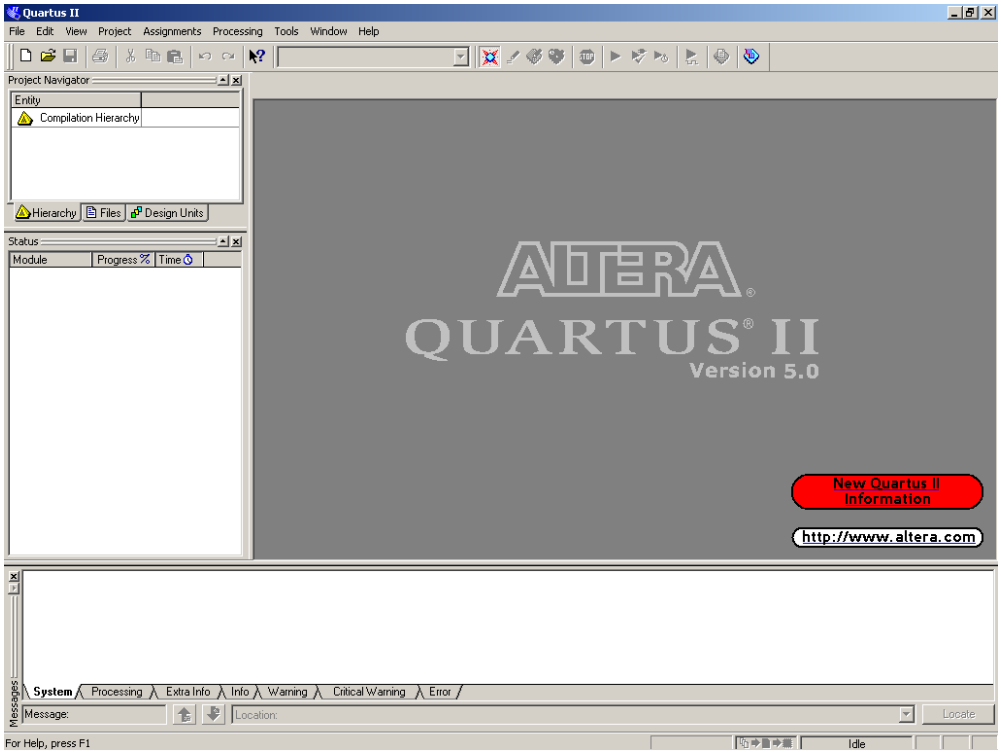


figure 1 : Quartus II

b. Allez dans le menu File et lancer **New Project Wizard**. Cet assistant va vous guider pas à pas pour créer votre projet.

Sous Quartus II, les projets sont liés à des cibles matérielles (FPGA ou CPLD). La principale raison provient du fait que les simulations sont post-synthèse par défaut, c'est-à-dire qu'elles prennent en compte le routage à l'intérieur du circuit (délais dû au routage).

c. Vous allez créer un répertoire TP_NUM sur la racine de votre compte et vous nommerez votre premier projet, TP1. La famille de FPGA que vous sélectionnerez sera la famille Cyclone II avec le circuit EP2C20F484C7. Les autres options seront celles par défaut. Une fois terminé, appuyer sur Finish.

d. Vous pouvez alors observer dans la fenêtre **Project Navigator** la composition de votre projet (entité de haut niveau et circuit utilisé).

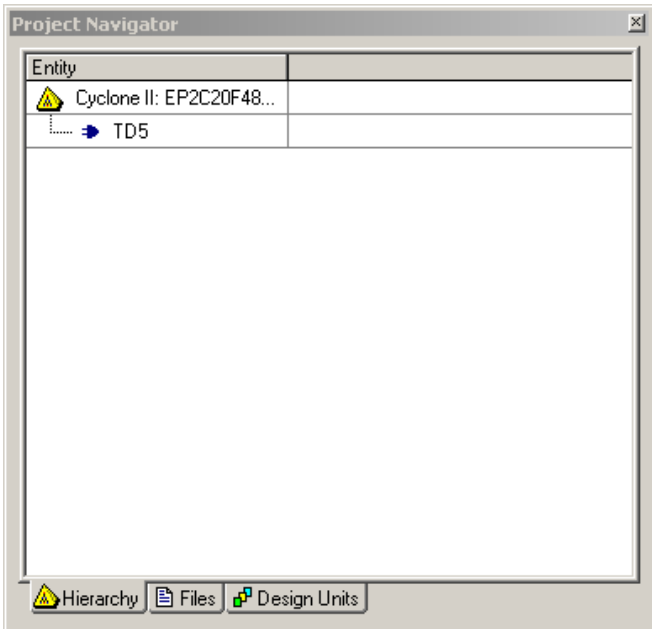


figure 2 : Navigateur de projet

II. Edition et compilation

Une fois le projet créé, il est possible d'y insérer plusieurs fichiers de description de composants : des descriptions structurelles à l'aide de fichier **BDF (Block Diagram File)** ou des descriptions dataflow ou comportementales à l'aide de fichier VHDL, AHDL ou Verilog. Nous nous limiterons ici à des descriptions schématiques. Les descriptions par du VHDL seront étudiés en S3.

a. Description structurelle

i. Dans le menu File\New, sélectionner Block Diagram/Schematic File. Une fenêtre vierge apparaît. Enregistrer votre fichier sous le nom Porte_NAND_SCHEME.bdf

ii. Quartus II vous offre toute une librairie de composant décrit en VHDL que vous pouvez vous servir. Pour cela, appuyer sur le bouton **Symbol Tool** matérialisé par le symbole d'une porte ET.

iii. Une nouvelle fenêtre apparaît. Prenez le composant porte NAND (nand2) et intégré le sur votre feuille vierge.

Nom Prénom :
Groupe :

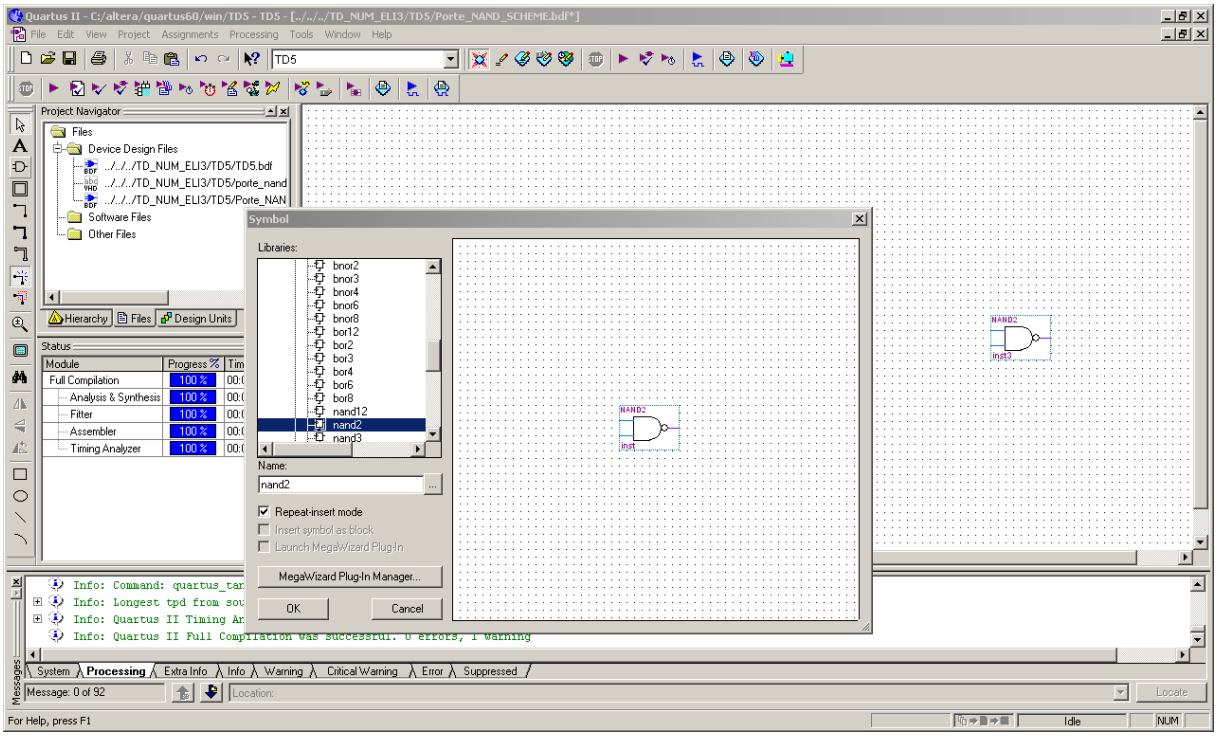


figure 3 : Bibliothèque des composants

Afin de pouvoir être simulé, il est alors nécessaire d'adjoindre à la porte NAND des entrées/sorties physiques.

iv. Pour cela, sélectionner dans la fenêtre symbole, les composants **input** et **output**. Reliez les entrées/sorties du composant avec des fils (signaux 1 bit correspond au bouton « wire »). Enregistrer votre fichier.

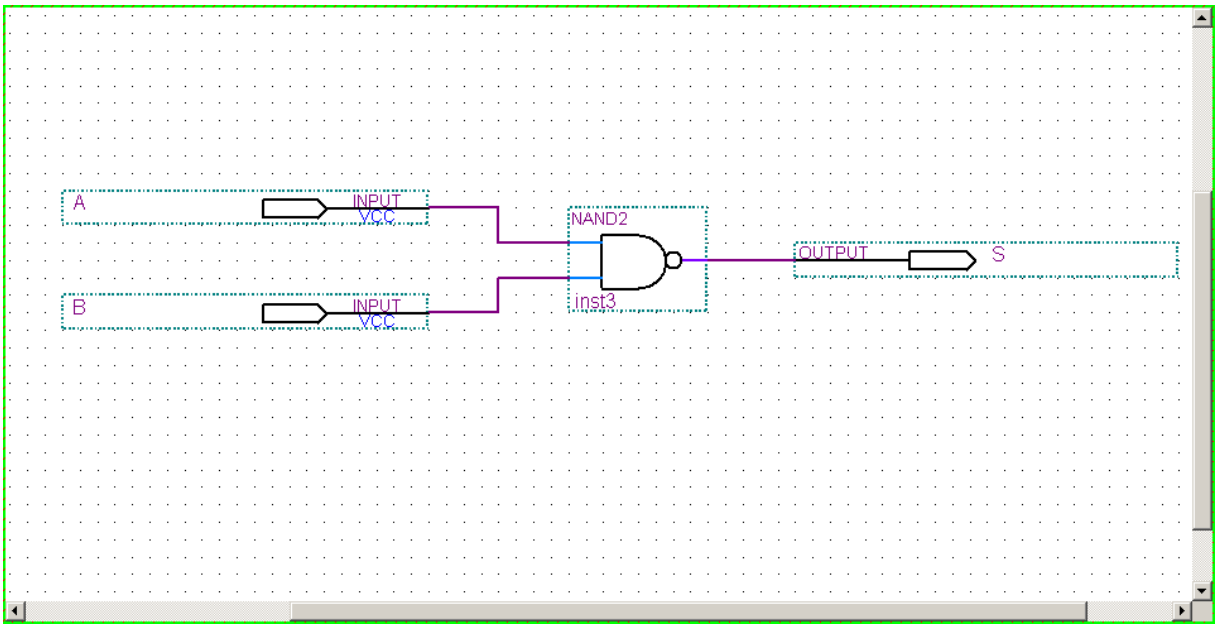


figure 4 : Description structurelle

b. Compilation

Nom Prénom :
Groupe :

Avant de simuler un circuit, il est nécessaire de vérifier qu'il ne comporte pas d'erreur de conception. Pour cela nous allons le compiler. Si votre projet comporte plusieurs descriptions, le compilateur par défaut synthétisera l'entité de haut niveau, celle qui correspond à la description de tout le système.

i. Dans l'onglet **Files** du **Project Navigator** sélectionner le fichier que vous voulez compiler. Appuyer sur le clic droit de la souris et sélectionner **Set as Top-Level Entity**. En réalisant cette opération le compilateur ne synthétisera que ce composant.

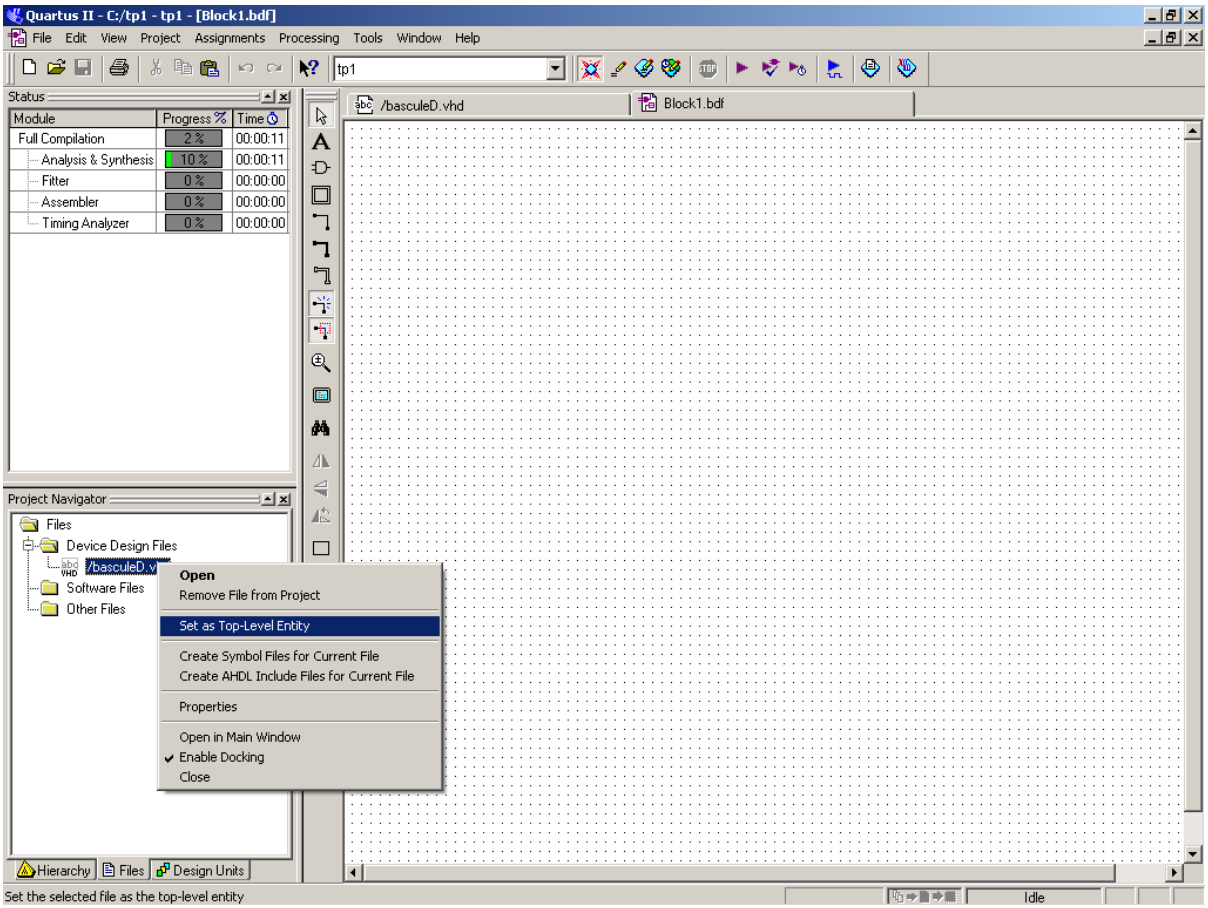


figure 5 : Définition de l'entité de haut niveau

ii. Aller dans le menu **Processing** et appuyer sur **Compiler Tool**. Une nouvelle fenêtre apparaît. Lancer le compilateur. Le compilateur analyse, réalise la synthèse et le placement routage du composant sur le circuit cible. L'ensemble des rapports de compilation et d'analyse des performances sont disponibles à partir de cette fenêtre (compiler tool).

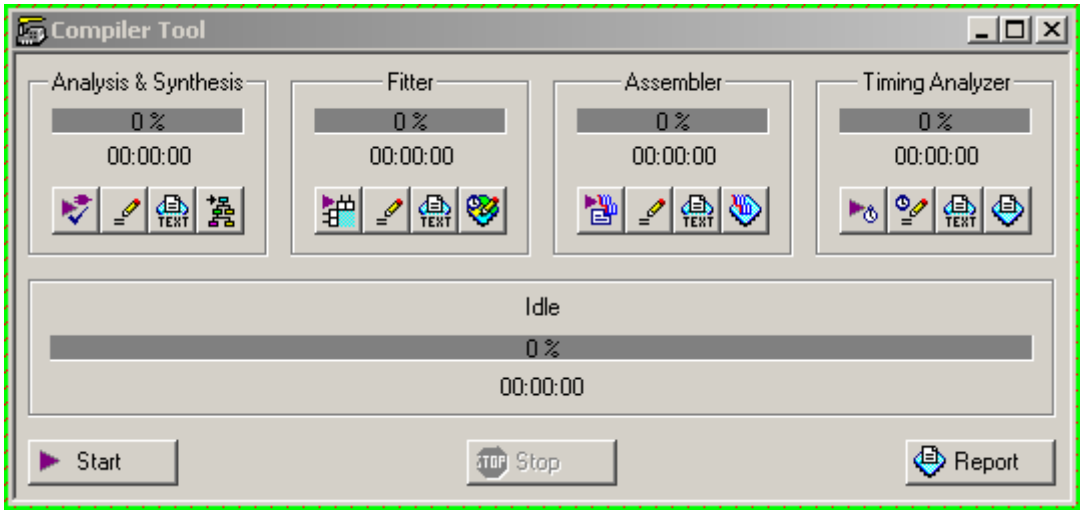


figure 6 : Outils de compilation, synthèse, placement routage et programmation

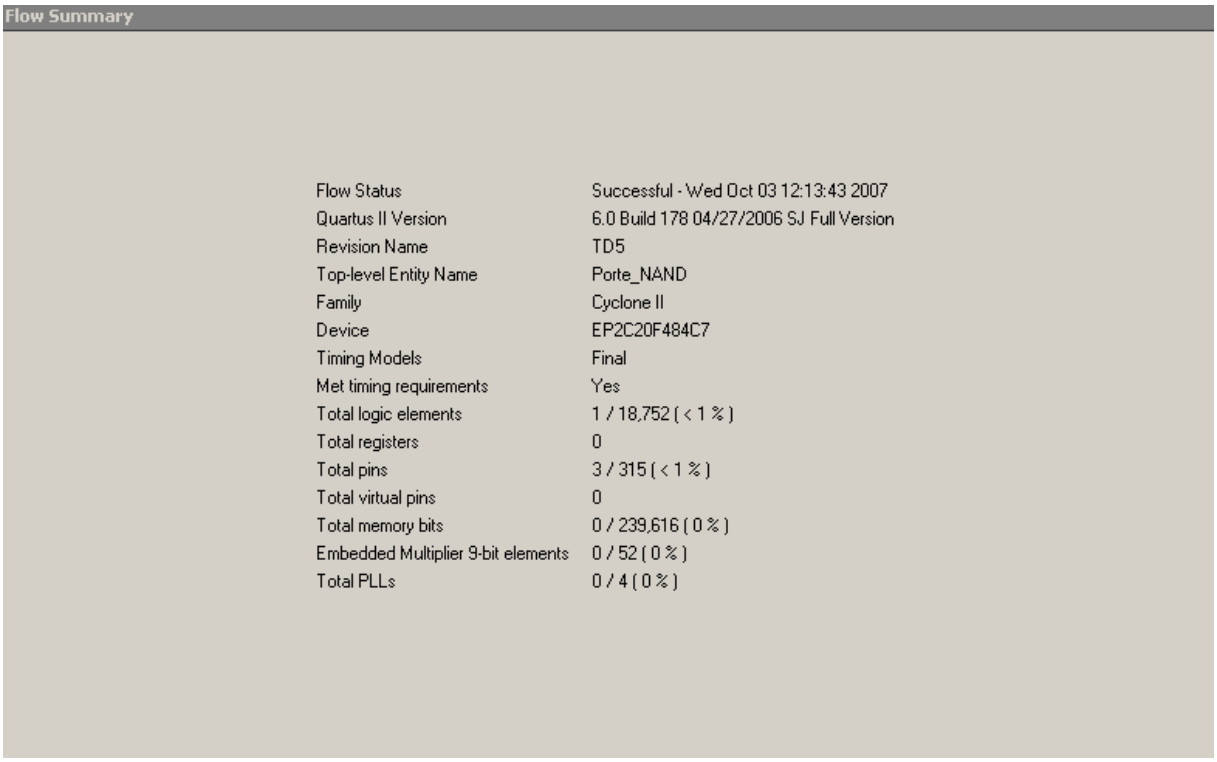


figure 7 : rapport de compilation

III. Simulation

Cette étape consiste à vérifier le comportement du composant crée. Attention, cette étape nécessite d'avoir compiler au préalablement les descriptions de circuit. Le simulateur permet de vérifier les comportements temporels et fonctionnels de descriptions dataflow, structurelles et comportementales.

a. Dans le menu File\New, sélectionner l'onglet **Other files** et **Vector Waveform Files.** Ce fichier permet de décrire visuellement le testbench que vous allez utiliser pour tester votre circuit.

b. Enregistrez le fichier sous **Bench_nom du composant**. Par exemple pour la porte NAND, le nom du test bench sera **Bench_Porte_NAND.vwf**.

i. Le fichier de simulation est divisé en deux parties. Une colonne pour le les broches d'entrées / sorties du composants et une zone graphique munie d'une échelle temporelle. Dans la colonne Name, à l'aide d'un clic droit de la souris, lancez la commande **Insert a node or bus**, puis l'outil **Node Finder.** Cet outil permet de récupérer les noms des entrées/sorties du composant que vous avez créé.

ii. Sélectionner les signaux, **A, B et S**. et terminer l'opération.

iii. **A** et **B** sont des entrées. Nous pouvons leur affecter des valeurs manuellement ou utiliser des stimulis prédéfinis. La barre d'outils **Waveform Editor** prédéfini plusieurs types de signaux (High, Low, Overwrite Clock, etc.). Afin de couvrir l'ensemble des combinaisons de A et B, vous prépositionnerez les valeurs des entrées comme sur la figure ci-dessous.

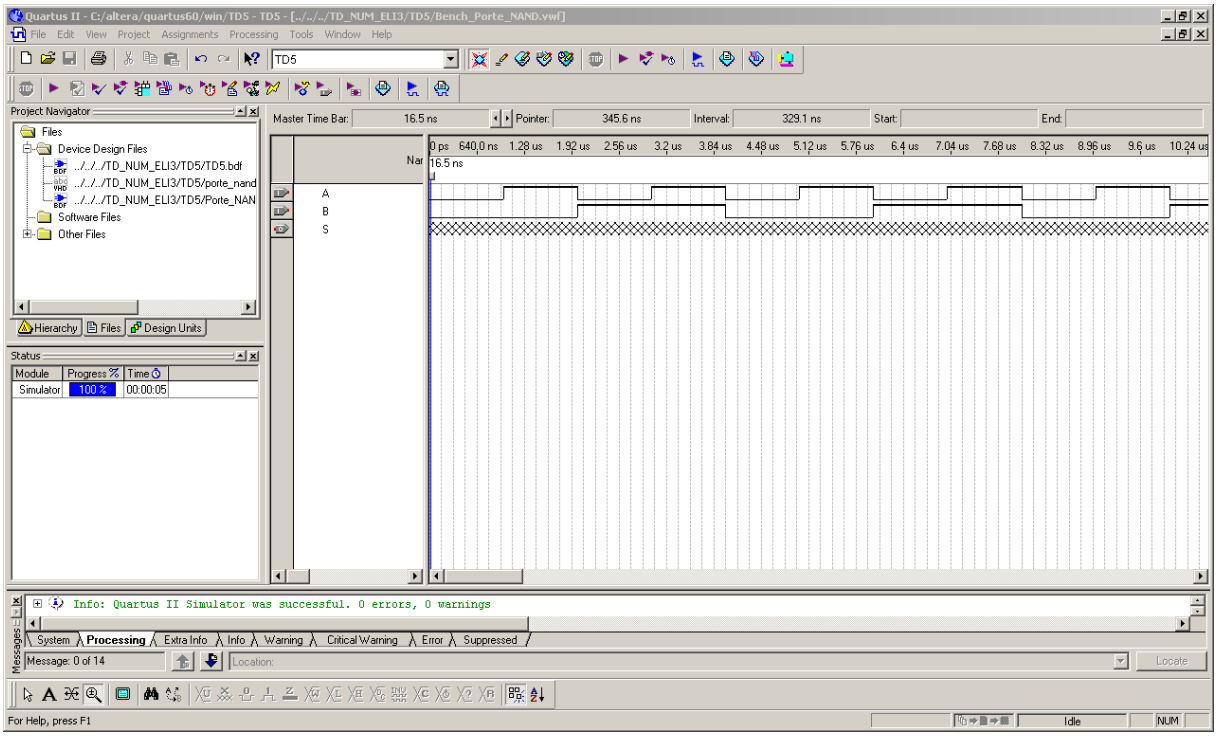


figure 8 : Test Bench

iv. Nous allons simuler le comportement du circuit avec le test_bench que vous venez de réaliser. Pour cela, dans le menu **Processing**, sélectionnez **Simulator Tool**. Réglez les paramètres de simulation

comme sur la figure ci-dessous et lancez la simulation. Vérifiez le résultat obtenu.

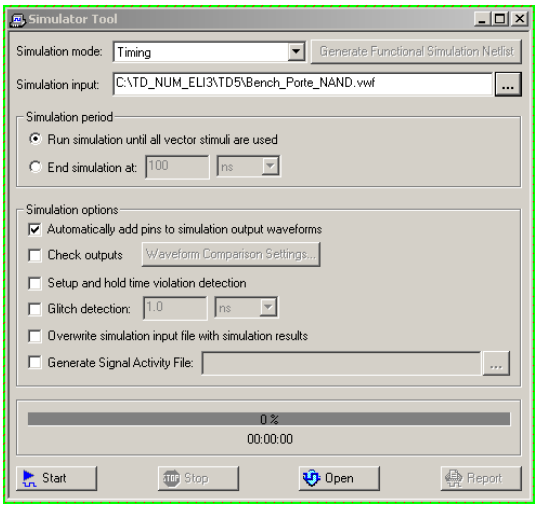


figure 9 : paramètres du simulateur

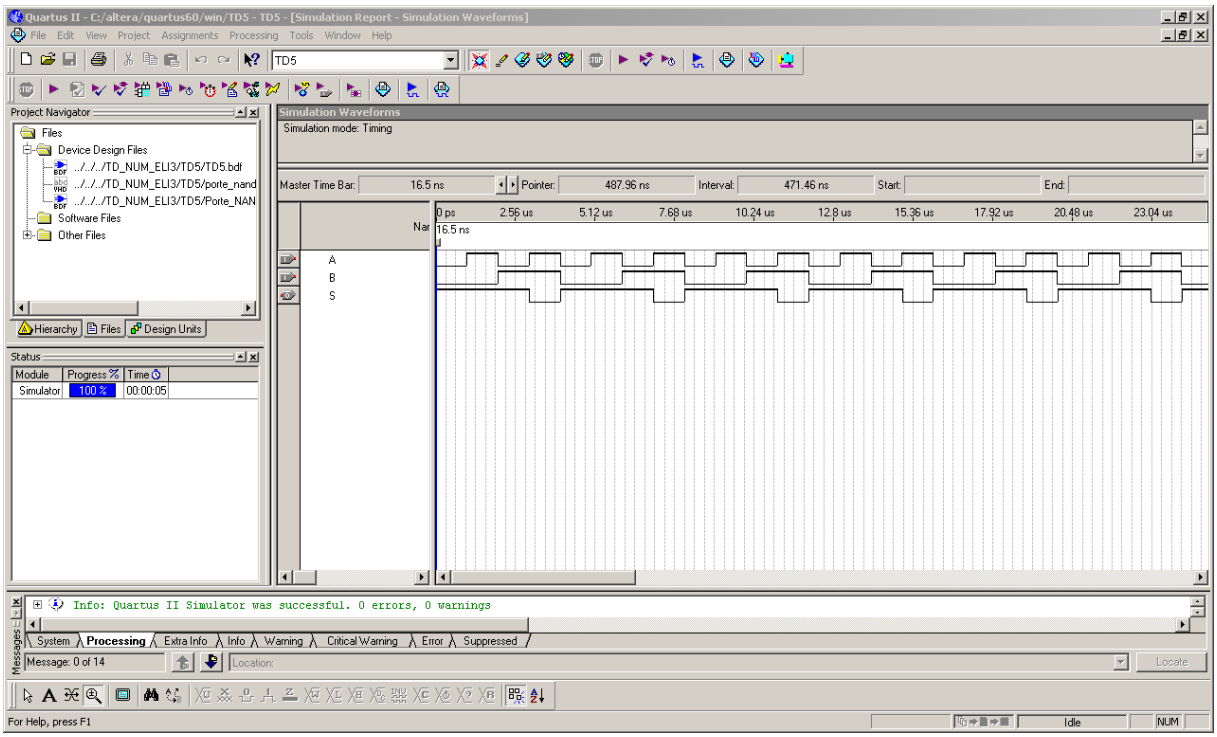


figure 10 : Résultat de simulation

IV. Programmation de la carte

Une fois la conception du système terminé et dans le cas ou le comportement en simulation respecte les spécifications du cahier des charges, Quartus II permet de programmer une carte pour vérifier le système sur un circuit.

Il est important dans un premier temps de relier les entrées / sorties de votre description aux entrées / sorties du circuit FPGA. A l'aide du manuel de la carte Cyclone II FPGA Starter Kit, déterminer quelles sont les ressources que vous voulez utiliser (boutons, switch, led, etc …).

Nom Prénom :
Groupe :

Pour l'exemple de la porte NAND nous utiliserons pour les entrées, les switch SW0 et SW1. Le résultat (sortie S) sera connecté directement à la ledG7. Les broches du FGPA correspondant à ces ressources sont :

SW0 : PIN_L22
SW1 : PIN_L21
LEDG7 : PIN_Y21

Pour affecter une entrée / sortie d'une description VHDL à une entrée / sortie physique, dans le menu **Assignments** sélectionnez **Assignment Editor** (CRTL+SHIFT+A).

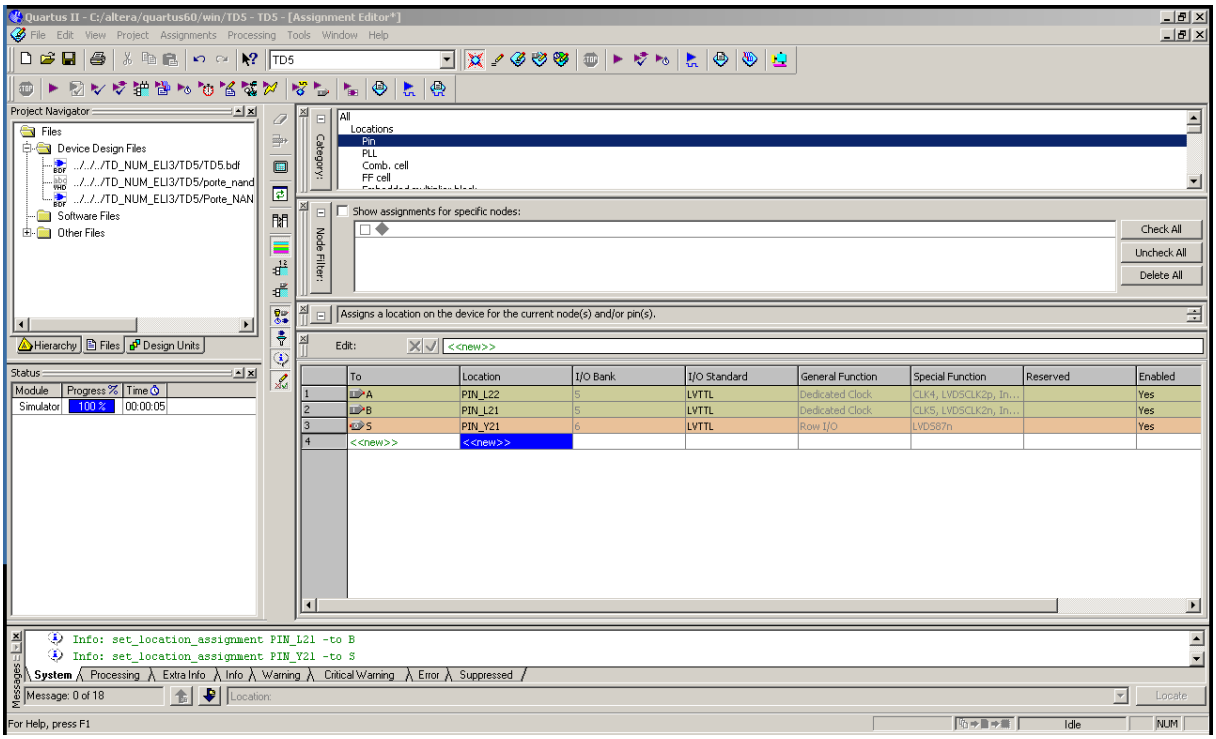
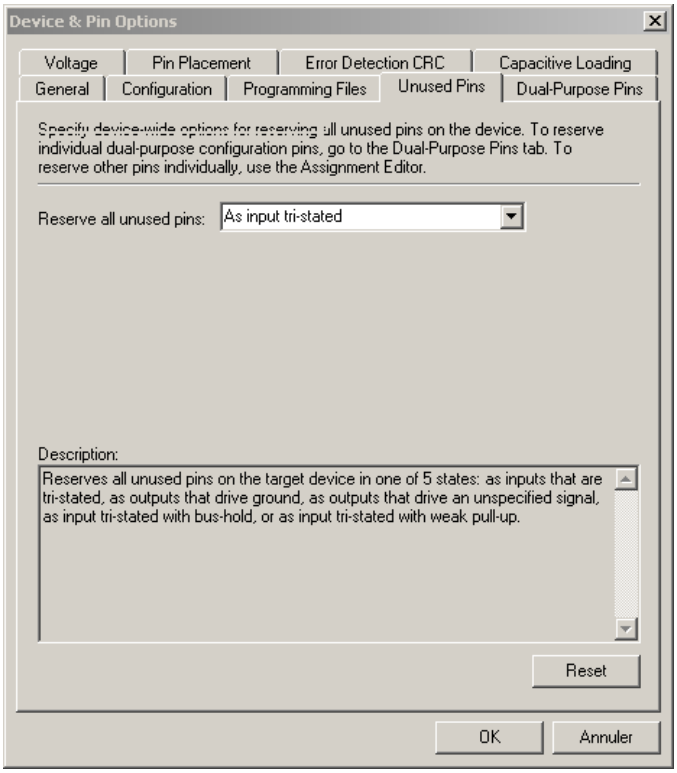


figure 11 : Assignation des broches du FPGA aux I/O du circuit VHDL

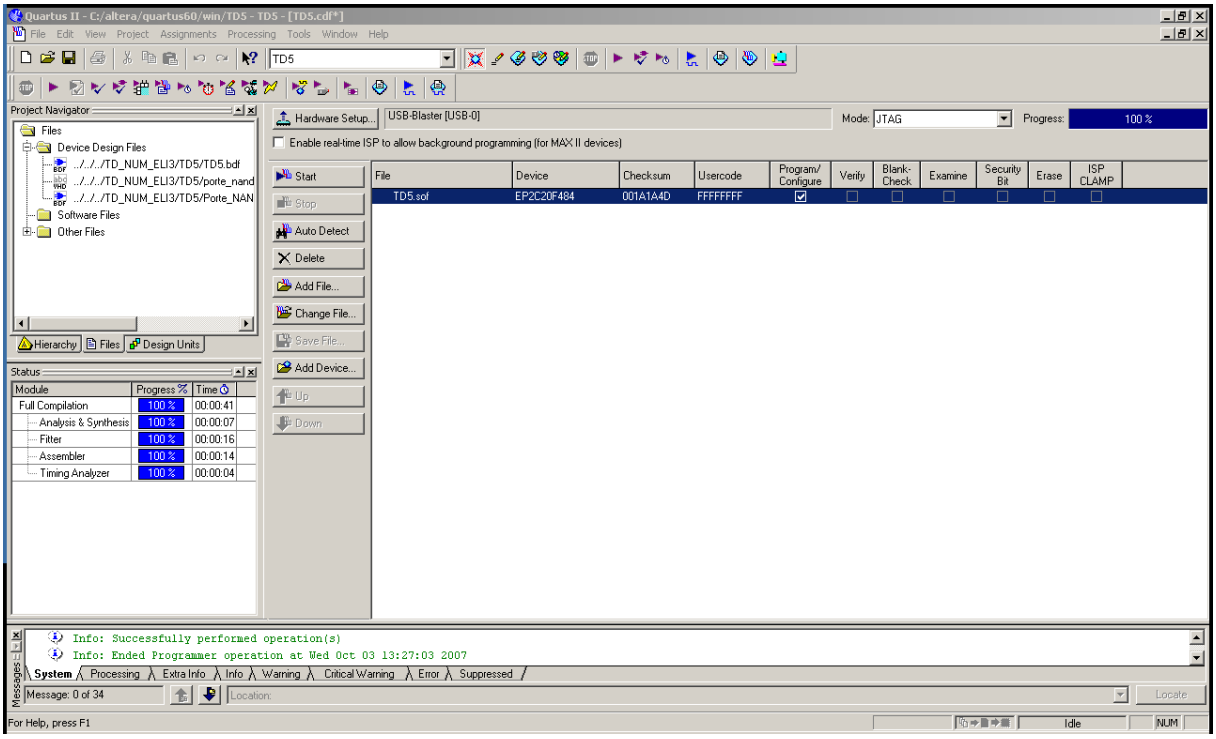
Le FPGA comporte au total 484 broches susceptibles d'être utilisées. Lorsqu'elles ne le sont pas il convient de les programmer en tant qu'entrées 3 états. Pour cela, dans le menu **Assignments**, sélectionnez **Device** et cliquez sur **Device & Pin Options …** Selectionnez l'onglet **Unused Pins**. Dans la cas **Reserved Unused Pins** selectionnez **as Input Tri-Stated**.

Nom Prénom :
Groupe :



Après assignation des entrées/sorties, recompilez votre circuit pour que les modifications soient prises en compte. Après les opérations de synthèse et de placement / routage, un fichier de programmation du circuit avec l'extension .sof

Pour cela, dans le menu **Tools**, sélectionner **Programmer**.



En fonction des switch SW0 et SW1 vérifier le comportement de la porte NAND décrite en VHDL au moyen de la Led G7.

Partie II. Applications

Maintenant que vous maîtrisez l'environnement de développement, c'est à vous de jouer !

V. Exercice n°1 : comparateur à seuils

Soit un nombre binaire sur 4 bits $A=A_3A_2A_1A_0$ codé en binaire naturel. On cherche à concevoir un circuit logique qui donne une sortie S haute quand le nombre binaire est supérieur à 8 et inférieur à 13.

- **a.** A l'aide d'une table de vérité et de tableau de Karnaugh, déterminer l'équation logique de S. Consigner les résultats dans votre compte-rendu
- **b.** Programmer une architecture de manière schématique à l'aide des portes élémentaires disponibles dans la bibliothèque. Consigner dans votre compte-rendu le logigramme.
- **c.** Simuler et consigner le résultat dans votre compte-rendu
- **d.** On affectera les signaux aux I/O du design
  - **i.** A_0 : PIN_L22 (SW0)
  - **ii.** A_1: PIN_L21 (SW1)
  - **iii.** A_2: PIN_M22 (SW2)
  - **iv.** A_3: PIN_V12 (SW3)
  - **v.** S : PIN_W22 (LEDG3)
- **e.** Vérifier expérimentalement la table de vérité
- **f.** **Faite valider par l'enseignant**

VI. Exercice n°2 : transcodeur gray -> binaire

Concevoir un transcodeur 4 bits traduisant une entrée codée en gray vers une sortie exprimée en binaire (table de vérité ci-dessous).

Code Gray	Code binaire
0000	**0000**
0001	**0001**
0010	**0011**
0011	**0010**
0100	**0111**
0101	**0110**
0110	**0100**

0111	**0101**
1000	**1111**
1001	**1110**
1010	**1100**
1011	**1101**
1100	**1000**
1101	**1001**
1110	**1011**
1111	**1010**

a. A l'aide d'une table de vérité et de tableau de Karnaugh, déterminer les équations logiques de B_3 B_2 B_1 B_0 en fonction de G_3 G_2 G_1 G_0 Consigner les résultats dans votre compte-rendu.
b. Programmer une architecture de manière schématique à l'aide des portes élémentaires disponibles dans la bibliothèque. Consigner dans votre compte-rendu le logigramme
c. Simuler et consigner le résultat dans votre compte-rendu
d. On affectera les signaux aux I/O du design
  i. G_0 : PIN_L22 (SW0)
  ii. G_1: PIN_L21 (SW1)
  iii. G_2: PIN_M22 (SW2)
  iv. G_3: PIN_V12 (SW3)
  v. B_0 : PIN_W22 (LEDG3)
  vi. B_1 : PIN_W21 (LEDG4)
  vii. B_2 : PIN_Y22 (LEDG6)
  viii. B_3 : PIN_Y21 (LEDG7)
e. Vérifier expérimentalement la table de vérité
f. **Faite valider par l'enseignant**

VII. Exercice n°4 : Encodeur de priorité

On désire réaliser un encodeur de priorité. On dispose de 4 périphériques A, B, C et D. Ces quatre périphériques peuvent avoir accès à la même ressource partagée S. De manière à éviter les conflits, il est alors nécessaire d'arbitrer les demandes des quatre périphériques. On affectera à A la plus haute priorité et à D la plus faible. A est prioritaire sur B qui est lui-même prioritaire sur C (lui-même prioritaire sur D). La table de vérité de cet encodeur est donnée ci-dessous :

ABCD	S

0000	**0000**
1XXX	**1000**
01XX	**0100**
001X	**0010**
0001	**0001**

a. A l'aide d'un tableau de Karnaugh, déterminer les équations logiques de S_3 S_2 S_1 S_0. Consigner les tableaux et les équations logiques dans votre compte-rendu.
b. Programmer une architecture schématique à l'aide des portes élémentaires disponibles dans la bibliothèque
c. Simuler
d. On affectera les signaux aux I/O du design
  i. A : PIN_L22 (SW0)
  ii. B : PIN_L21 (SW1)
  iii. C : PIN_M22 (SW2)
  iv. D : PIN_V12 (SW3)
  v. S_0 : PIN_W22 (LEDG3)
  vi. S_1 : PIN_W21 (LEDG4)
  vii. S_2 : PIN_Y22 (LEDG6)
  viii. S_3 : PIN_Y21 (LEDG7)
e. Vérifier expérimentalement la correspondance
f. **Faite valider par l'enseignant**