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:

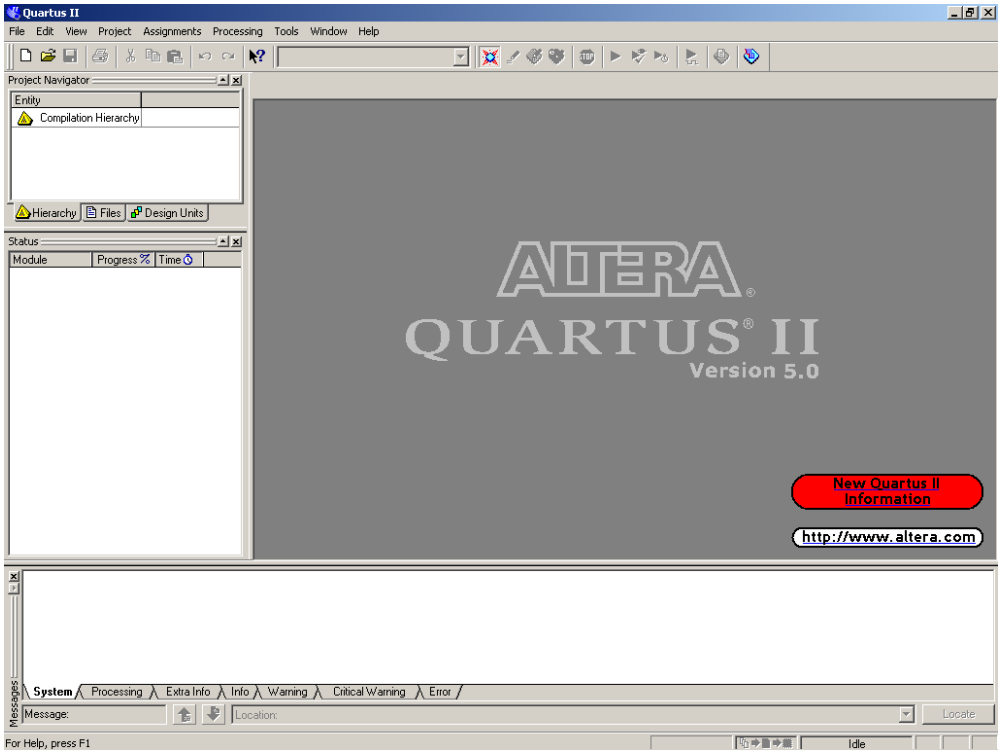


figure 1 : Quartus II

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).

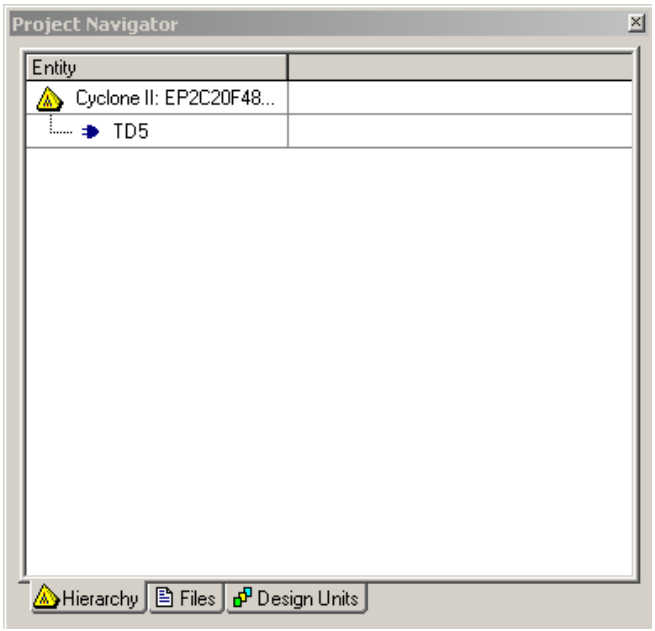


figure 2 : Navigateur de projet

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.

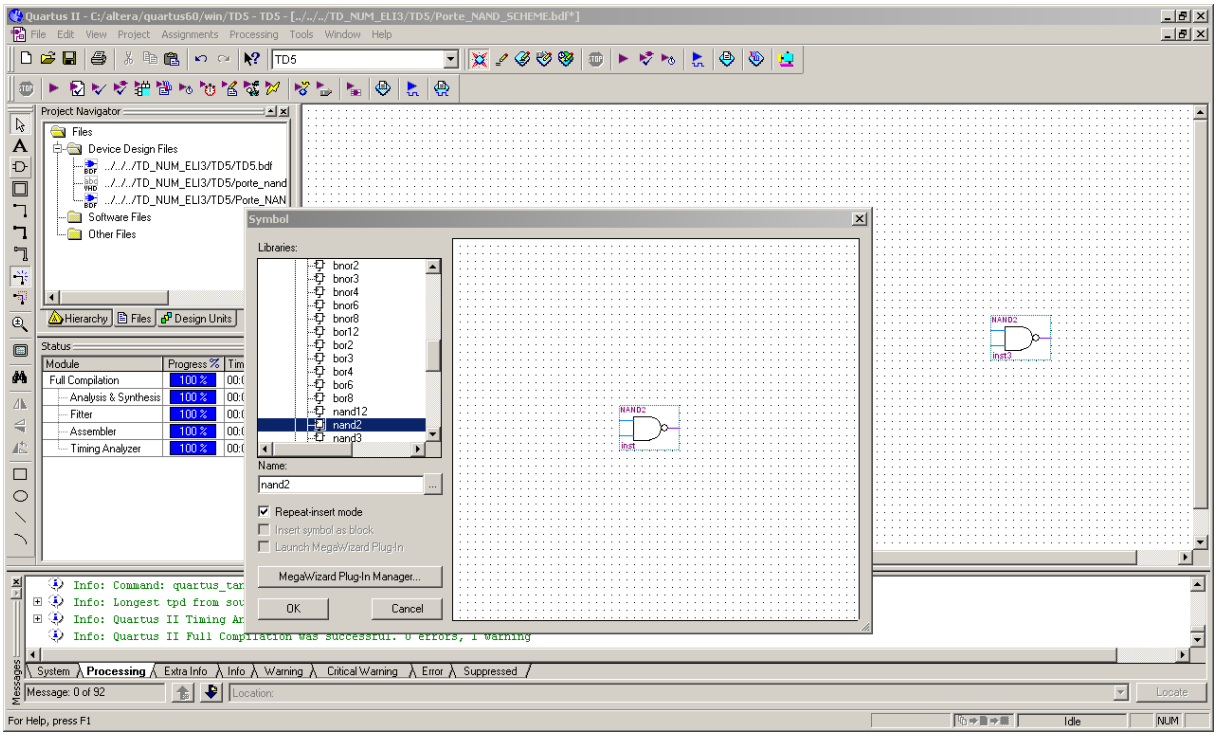


figure 3 : Bibliothèque des composants

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.

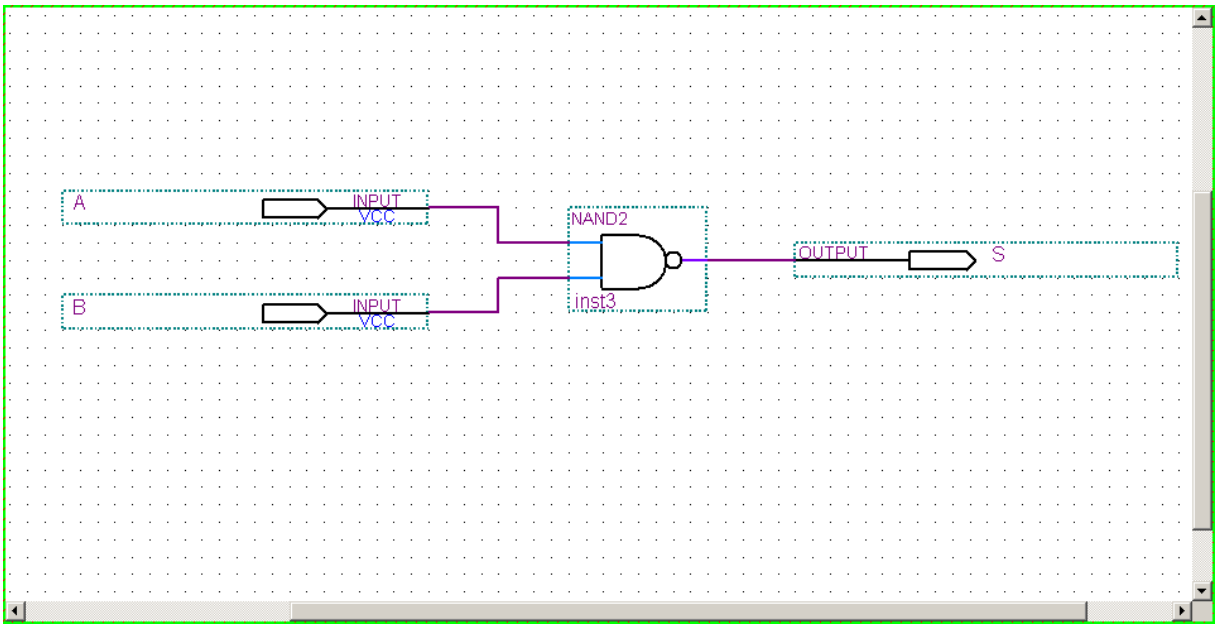


figure 4 : Description structurelle

b. Compilation

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.

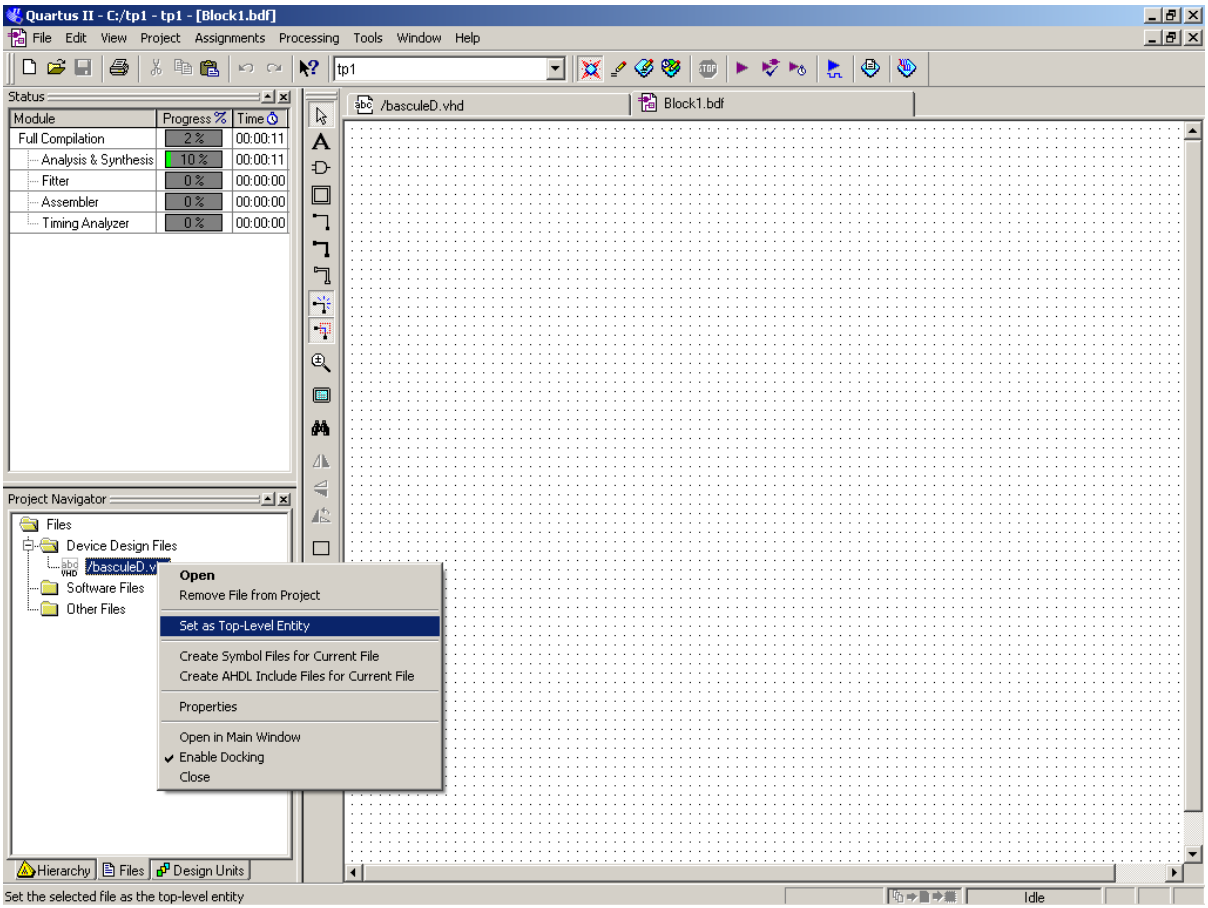


figure 5 : Définition de l'entité de haut niveau

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).

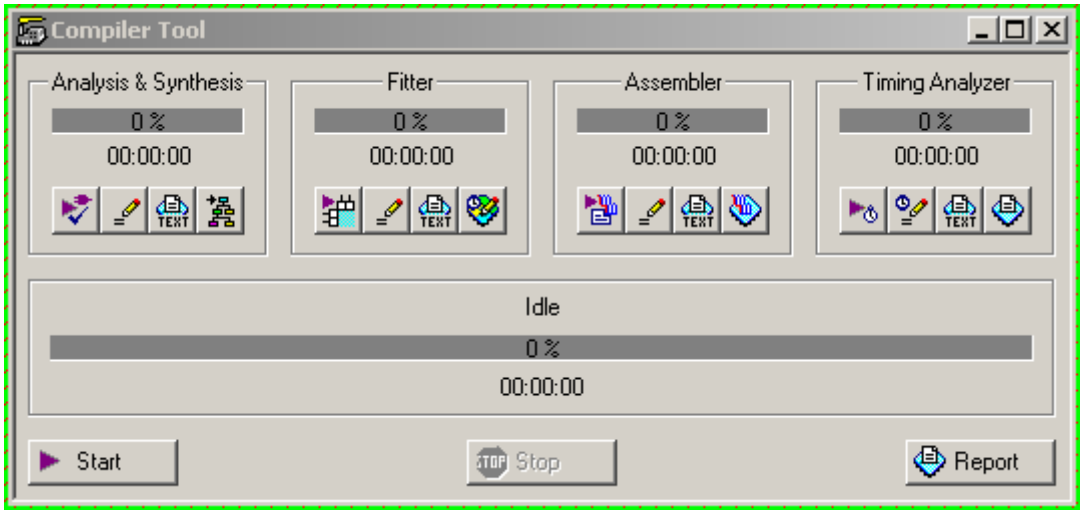


figure 6 : Outils de compilation, synthèse, placement routage et programmation

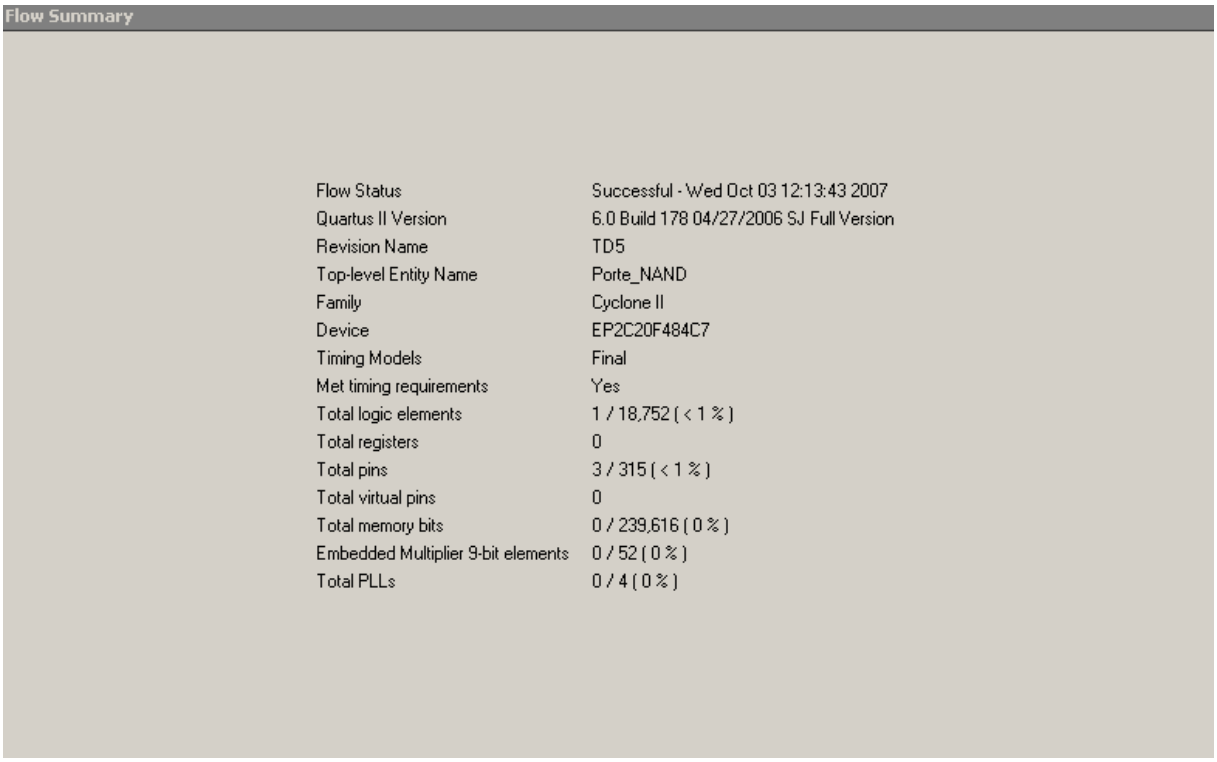


figure 7 : rapport de compilation

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.

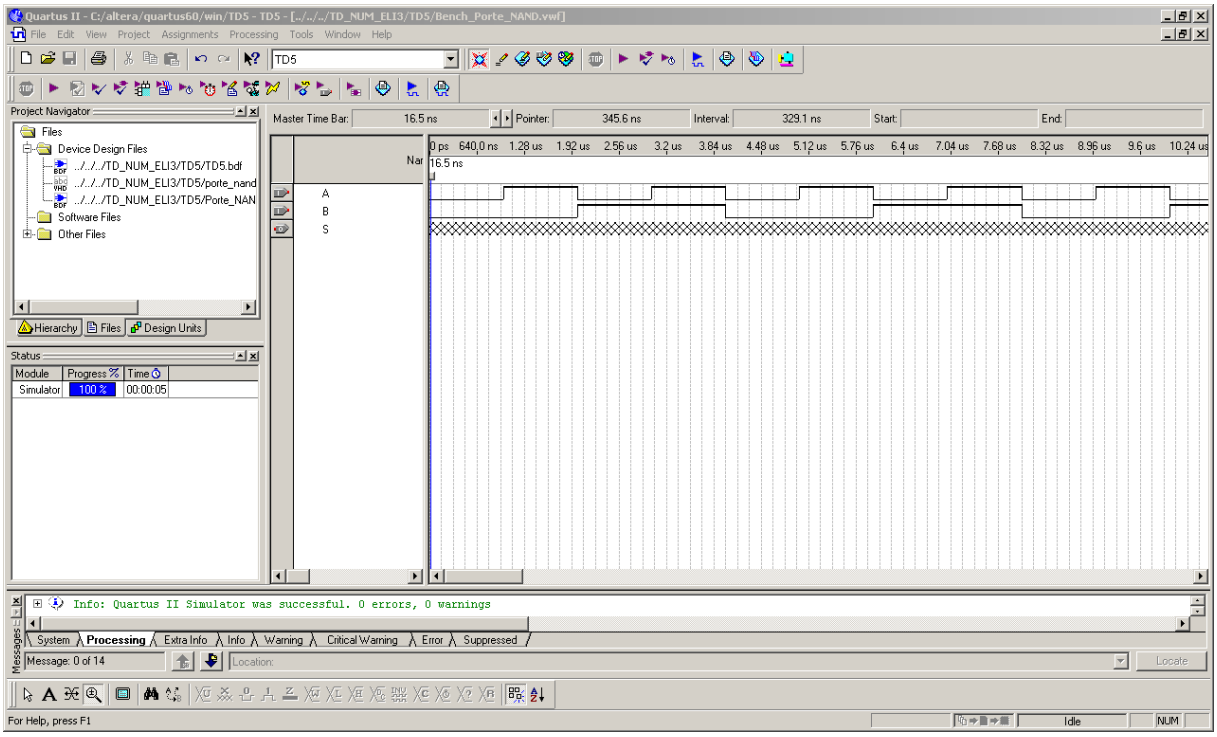


figure 8 : Test Bench

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.

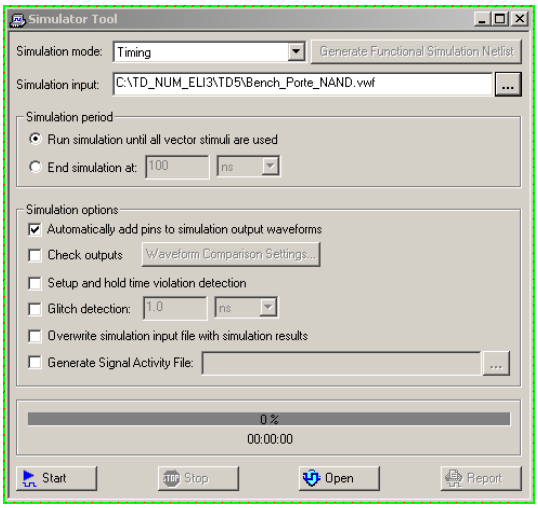


figure 9 : paramètres du simulateur

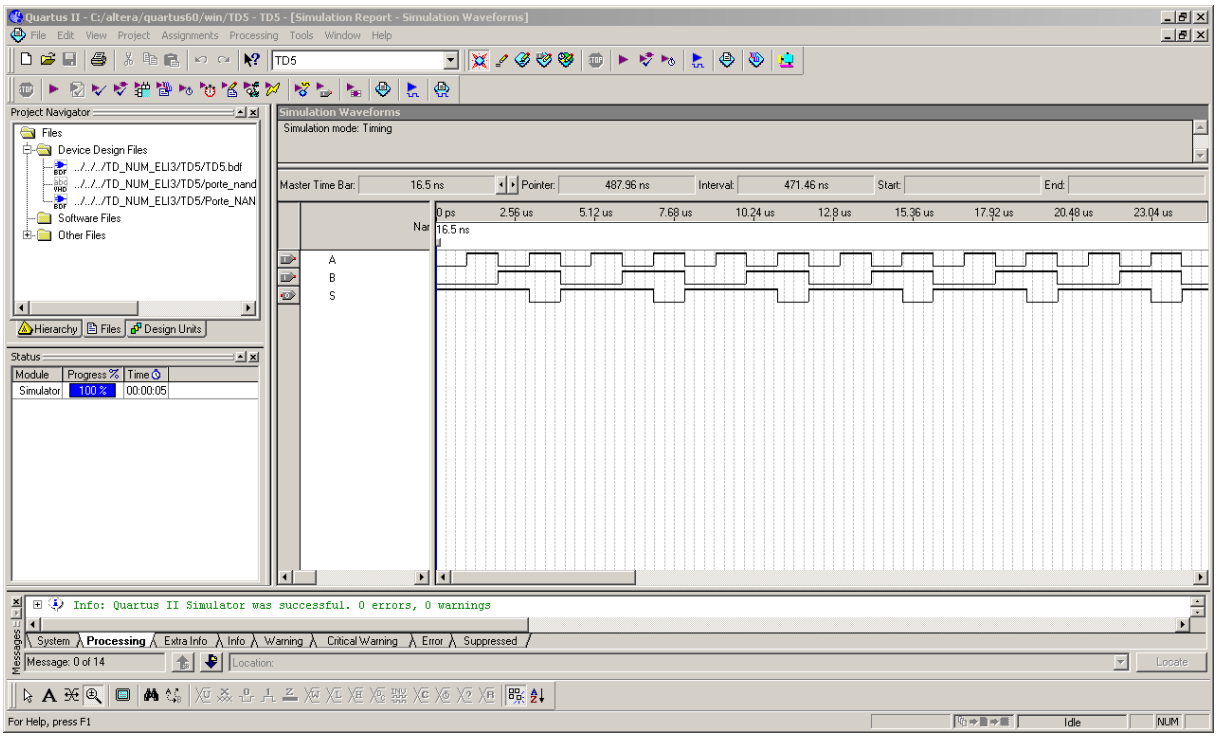


figure 10 : Résultat de simulation

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 …).

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).

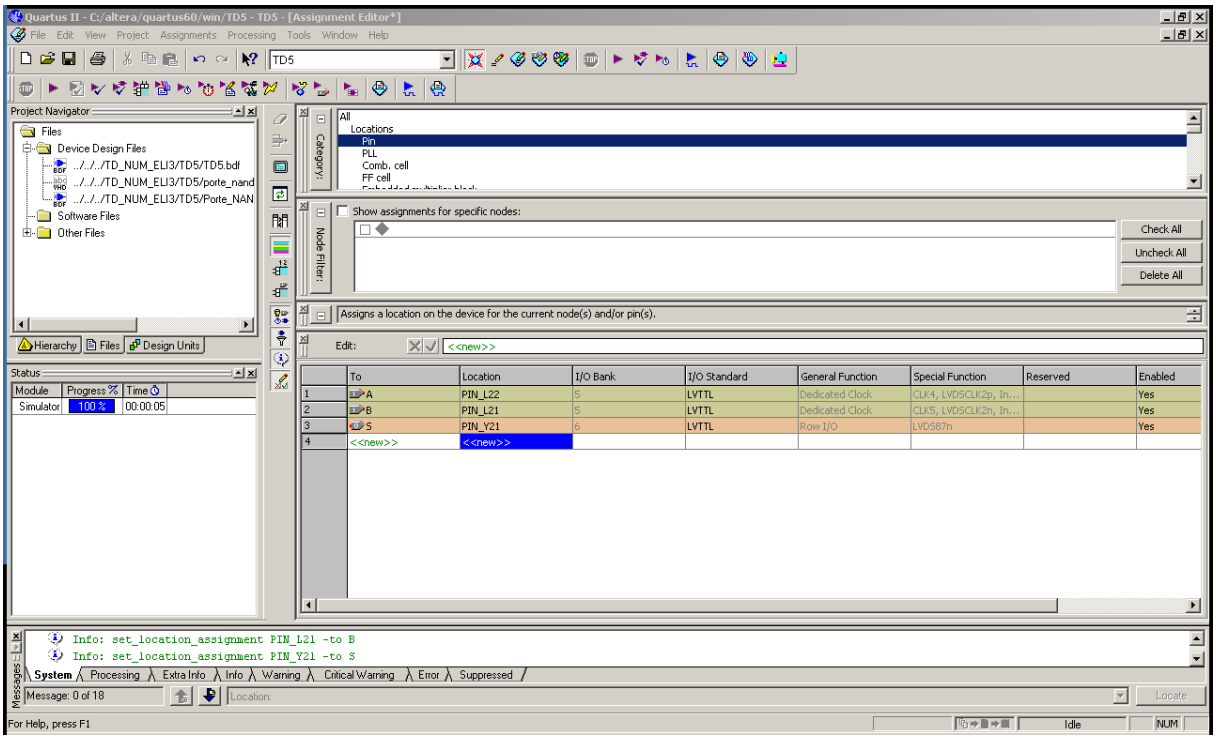
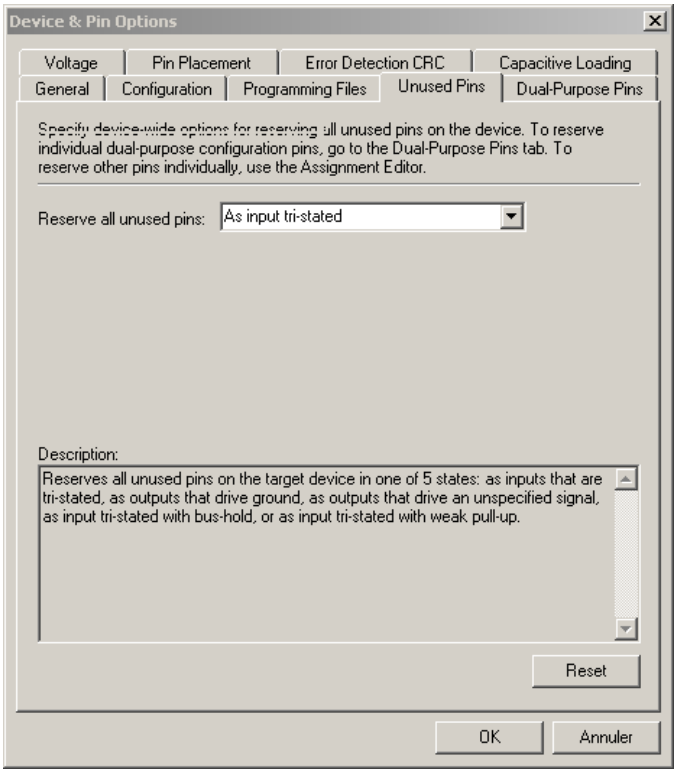


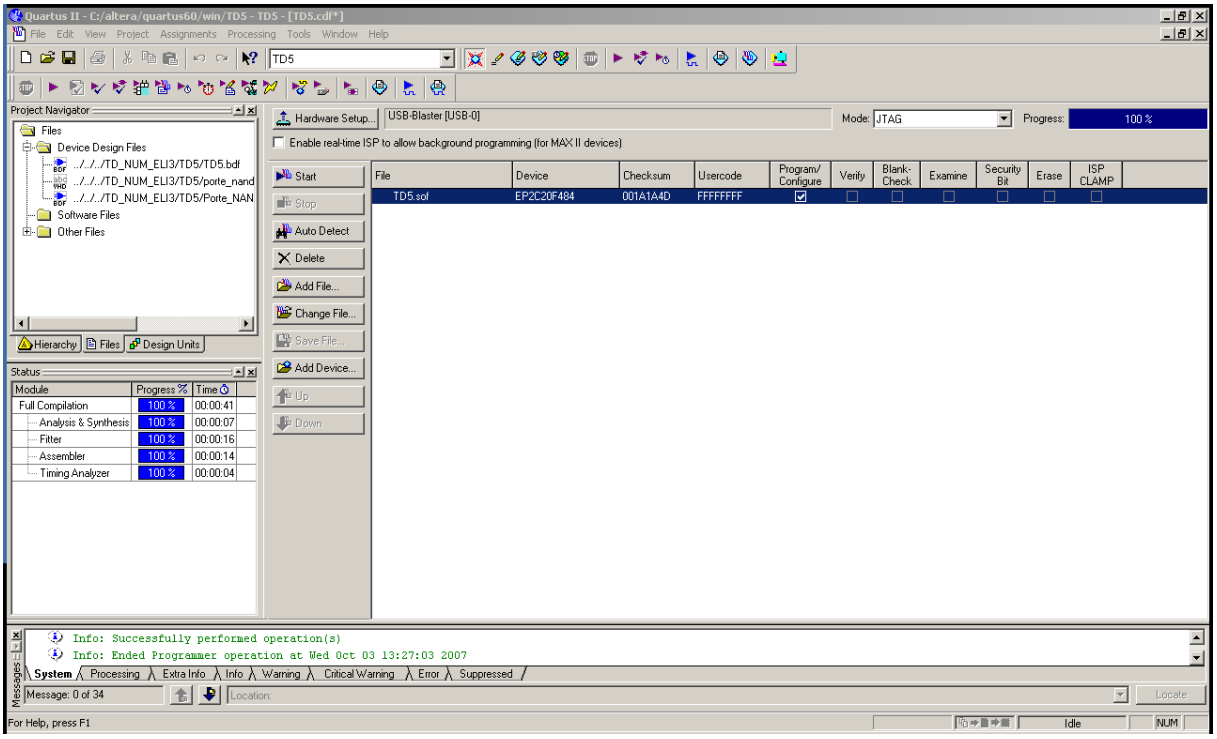
figure 11 : Assignation des broches du FPGA aux I/O du circuit VHDL

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**.



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**

Nom Prénom :
Groupe :

TP2

CIRCUITS ARITHMETIQUES

Préliminaires

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.
b. Allez dans le menu File et lancer **New Project Wizard**. Cet assistant va vous guidez pas à pas pour créer votre projet.
c. Vous allez créer un répertoire TP_NUM/TP2 sur la racine de votre compte et vous nommerez ce TP, TP2. La famille de FPGA que vous utiliserez est indiquée sur le circuit programmable (EP2C20F484C7) de le carte de prototypage. 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).

Exercice 1 : Additionneur 1 bit

a. Donner la table de vérité de l'additionneur 1 bit. Consignez la table dans votre compte-rendu.
b. Déterminer les équations logiques de S et C_{OUT} en fonction de A, B et C_{IN}. Consignez les équations dans votre CR.
c. Créer un nouveau fichier bdf et renommer le adder1bit
d. Afin que l'outil puisse le compiler, déclarer le en tant que « top level entity ». Clique droit de la souris sur le fichier sélectionné dans la fenêtre de gauche de l'environnement.
e. Programmer une architecture de manière schématique à l'aide des portes élémentaires disponibles dans la bibliothèque. Consigner dans votre compte-rendu le logigramme.
f. Simuler et consigner le résultat dans votre compte-rendu
g. On affectera les signaux aux I/O du design. Utiliser l'assignment editor.
A : PIN_L22 (SW0)
B : PIN_L21 (SW1)
C_{in} : PIN_M22 (SW2)
S : PIN_W22 (LEDG3)
C_{out} : PIN_W21 (LEDG4)

h. Vérifier expérimentalement la table de vérité
i. Créer un symbole graphique pour ce composant adder1bit. Menu file/create-update et selectionner create symbol files for current file. Votre composant est maintenant accessible dans la bibliothèque des composants de votre projet.
j. **Faite valider par l'enseignant**

Exercice 2 : Additionneur 4 bits

La conception d'un additionneur 4 bits peut être réalisé en cascadant 4 additionneurs 1 bits et en propageant la retenue d'un étage à l'autre.

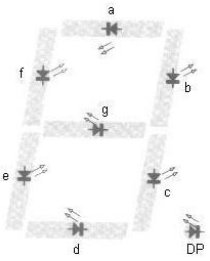
a. Créer un nouveau fichier bdf et renommer le adder4bits
b. Afin que l'outil puisse le compiler, déclarer le en tant que « top level entity ». Clique droit de la souris sur le fichier sélectionné dans la fenêtre de gauche de l'environnement.
c. Proposer un logigramme pour l'additionneur 4 bits à partir de 4 additionneur 1 bit
d. **Faite valider par l'enseignant**
e. Programmer une architecture schématique à l'aide des portes élémentaires disponibles dans la bibliothèque.
f. Simuler et consigner le résultat dans votre compte-rendu
g. On affectera les signaux aux I/O du design. Utiliser l'assignment editor.

A(0) : PIN_L22 (SW0)
A(1) : PIN_L21 (SW1)
A(2) : PIN_M22 (SW2)
A(3) : PIN_V12 (SW3)
B(0) : PIN_W12 (SW4)
B(1) : PIN_U12 (SW5)
B(2) : PIN_U11 (SW6)
B(3) : PIN_M2 (SW7)
Cin: PIN_M1 (SW8)
S(0) : PIN_U22 (LEDG0)
S(1) : PIN_U21 (LEDG1)
S(2) : PIN_V22 (LEDG2)
S(3) : PIN_V21 (LEDG3)
Cout : PIN_W22 (LEDG4)

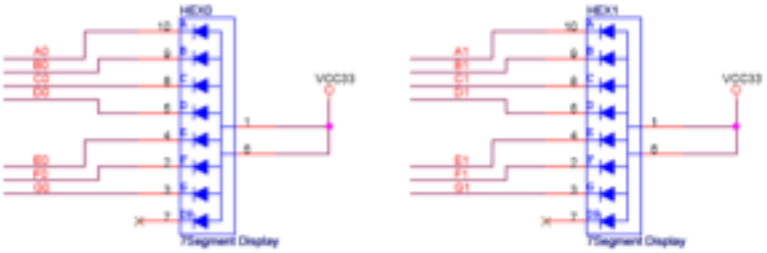
h. Vérifier expérimentalement la table de vérité
i. Créer un symbole graphique pour ce composant adder3bit. Menu file/create-update et selectionner create symbol files for current file. Votre composant est maintenant accessible dans la bibliothèque des composants de votre projet.
j. **Faite valider par l'enseignant**

Exercice 3 : Transcodeur Binaire / 7 segments

On désire afficher le résultat de l'addition de l'opérande A avec l'opérande B. Le résultat de l'addition est sur 4 bits (retenue sortante et sortie S). Pour afficher le résultat, on va utiliser un des quatre afficheurs 7 segments de la carte (Hex0). Un afficheur 7 segments comme son nom l'indique comporte 7 leds, référencées a, b, c, d, e, f et g. Deux types d'afficheurs existent, les afficheurs à anode ou à cathode commune. Dans le premier cas, un 0 allume le segment tandis que dans le second cas, c'est un 1 logique.



a. Le schéma électrique ci-dessous montre comment sont connectés les deux afficheurs 7 segments (HEX0 et HEX1) aux broches du FPGA. Déterminer s'ils sont à anode ou à cathode commune.



b. Créer un nouveau fichier bdf et renommer le 7 segments
c. Afin que l'outil puisse le compiler, déclarer le en tant que « top level entity ». Clique droit de la souris sur le fichier sélectionné dans la fenêtre de gauche de l'environnement.
d. A partir de tableaux de Karnaugh, déterminer les équations logiques des segments a, b, c, d, e, f et g en fonction du mot binaire ($A_3A_2A_1A_0$) appliqué en entrée du transcodeur.
e. Proposer un logigramme pour chaque segment
f. **Faite valider par l'enseignant**
g. Programmer une architecture schématique à l'aide des portes élémentaires disponibles dans la bibliothèque.
h. Simuler et consigner le résultat dans votre compte-rendu
i. On affectera aux signaux aux I/O du design

A(0) : PIN_L22 (SW0)
A(1) : PIN_L21 (SW1)
A(2) : PIN_M22 (SW2)
A(3) : PIN_V12 (SW3)
a : PIN_J2
b : PIN_J1
c : PIN_F2
d : PIN_F1
e : PIN_H2
f : PIN_H1
g : PIN_E2

j. Programmer la carte Altera et vérifier le comportement du transcodeur 7 segments
k. **Faite valider par l'enseignant**
l. Créer un symbole pour ce composant

Exercice 4 : Additionneur 4 bits avec affichage du résultat

a. Ouvrir le fichier bdf adder4bits
b. On souhaite afficher le résultat de l'addition des deux opérandes. Combien faut il d'afficheur 7 segments ?
c. Modifier le logigramme et insérer le composant transcodeur de l'exercice 3 pour réaliser cet affichage
d. **Faite valider par l'enseignant**

e. Simuler et consigner le résultat dans votre compte-rendu

f. On affectera les signaux aux I/O du design. Utiliser l'assignment editor.

A(0) : PIN_L22 (SW0)
A(1) : PIN_L21 (SW1)
A(2) : PIN_M22 (SW2)
A(3) : PIN_V12 (SW3)
B(0) : PIN_W12 (SW4)
B(1) : PIN_U12 (SW5)
B(2) : PIN_U11 (SW6)
B(3) : PIN_M2 (SW7)
Cin: PIN_M1 (SW8)

La figure ci-dessous donne les affectations des broches du FPGA / afficheurs 7 segments

Signal Name	FPGA Pin No.	Description
HEX0[0]	PIN_J2	Seven Segment Digit 0[0]
HEX0[1]	PIN_J1	Seven Segment Digit 0[1]
HEX0[2]	PIN_H2	Seven Segment Digit 0[2]
HEX0[3]	PIN_H1	Seven Segment Digit 0[3]
HEX0[4]	PIN_F2	Seven Segment Digit 0[4]
HEX0[5]	PIN_F1	Seven Segment Digit 0[5]
HEX0[6]	PIN_E2	Seven Segment Digit 0[6]
HEX1[0]	PIN_E1	Seven Segment Digit 1[0]
HEX1[1]	PIN_H6	Seven Segment Digit 1[1]
HEX1[2]	PIN_H5	Seven Segment Digit 1[2]
HEX1[3]	PIN_H4	Seven Segment Digit 1[3]
HEX1[4]	PIN_G3	Seven Segment Digit 1[4]
HEX1[5]	PIN_D2	Seven Segment Digit 1[5]
HEX1[6]	PIN_D1	Seven Segment Digit 1[6]
HEX2[0]	PIN_G5	Seven Segment Digit 2[0]
HEX2[1]	PIN_G6	Seven Segment Digit 2[1]
HEX2[2]	PIN_C2	Seven Segment Digit 2[2]
HEX2[3]	PIN_C1	Seven Segment Digit 2[3]
HEX2[4]	PIN_E3	Seven Segment Digit 2[4]
HEX2[5]	PIN_E4	Seven Segment Digit 2[5]
HEX2[6]	PIN_D3	Seven Segment Digit 2[6]
HEX3[0]	PIN_F4	Seven Segment Digit 3[0]
HEX3[1]	PIN_D5	Seven Segment Digit 3[1]
HEX3[2]	PIN_D6	Seven Segment Digit 3[2]
HEX3[3]	PIN_J4	Seven Segment Digit 3[3]
HEX3[4]	PIN_L8	Seven Segment Digit 3[4]
HEX3[5]	PIN_F3	Seven Segment Digit 3[5]
HEX3[6]	PIN_D4	Seven Segment Digit 3[6]

g. Vérifier expérimentalement le comportement de l'additionneur 4 bits.

h. Faite valider par l'enseignant

Exercice 5 : Soustracteur 4 bits avec affichage du résultat

i. Créer un fichier bdf sub4bits
j. Comment peut on passer d'un additionneur à un soustracteur 4 bits ?
k. Proposer un logigramme
l. **Faite valider par l'enseignant**
m. Programmer une architecture schématique
n. Simuler et consigner le résultat dans votre compte-rendu
o. On affectera aux signaux aux I/O du design. Utiliser l'assignment editor.

A(0) : PIN_L22 (SW0)
A(1) : PIN_L21 (SW1)
A(2) : PIN_M22 (SW2)
A(3) : PIN_V12 (SW3)
B(0) : PIN_W12 (SW4)
B(1) : PIN_U12 (SW5)
B(2) : PIN_U11 (SW6)
B(3) : PIN_M2 (SW7)
Cin: PIN_M1 (SW8)

p. Programmer la carte Altera et vérifier le comportement du soustracteur 4 bits
q. **Faite valider par l'enseignant**
r. Créer un symbole pour ce composant

Exercice 6 : Additionneur/Soustracteur 4 bits avec affichage du résultat

a. Créer un fichier bdf addsub4bits
b. On désire réaliser un additionneur / soustracteur 4 bits. Pour cela, le signal Add/Sub permettra de choisir l'opération. Si Add/Sub est à 0, l'opération de soustraction est choisie, sinon c'est une addition.
c. Proposer un logigramme
d. **Faite valider par l'enseignant**
e. Programmer une architecture schématique
f. Simuler et consigner le résultat dans votre compte-rendu
g. On affectera aux signaux aux I/O du design. Utiliser l'assignment editor.

A(0) : PIN_L22 (SW0)
A(1) : PIN_L21 (SW1)
A(2) : PIN_M22 (SW2)
A(3) : PIN_V12 (SW3)
B(0) : PIN_W12 (SW4)
B(1) : PIN_U12 (SW5)
B(2) : PIN_U11 (SW6)
B(3) : PIN_M2 (SW7)
Cin: PIN_M1 (SW8)

h. Programmer la carte Altera et vérifier le comportement de l'additionneur/soustracteur 4 bits
i. **Faite valider par l'enseignant**
j. Créer un symbole pour ce composant

TP3

COMPTEURS

Préliminaires

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.
b. Allez dans le menu File et lancer **New Project Wizard**. Cet assistant va vous guidez pas à pas pour créer votre projet.
c. Vous allez créer un répertoire TP_NUM/TP3 sur la racine de votre compte et vous nommerez ce TP, TP3. La famille de FPGA que vous utiliserez est indiquée sur le circuit programmable (EP2C20F484C7) de le carte de prototypage. 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).

Exercice 1 : Compteur BCD synchrone

a. Créer un fichier BDF et renommer le counter10
b. Donner la table de vérité d'un compteur binaire qui compte de 0 à 9
c. En utilisant la méthode de Marcus, déterminer les équations des entrées des 4 bascules D. Consignez les équations dans votre CR.
d. Programmer une architecture de manière schématique à l'aide des portes élémentaires et des bascules d (dff) disponibles dans la bibliothèque. Consigner dans votre compte-rendu le logigramme.
e. Simuler et consigner le résultat dans votre compte-rendu
f. **Faite valider par l'enseignant**
g. Créer un symbole pour ce circuit

Exercice 2 : Compteur quinaire synchrone

a. Créer un fichier BDF et renommer le counter5

La table de vérité d'un compteur quinaire est donnée ci-dessous :

N	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	1	1	1
4	1	1	0

b. En utilisant la méthode de Marcus, déterminer les équations des entrées des 3 bascules D. Consignez les équations dans votre CR.

c. Programmer une architecture de manière schématique à l'aide des portes élémentaires et des bascules d (dff) disponibles dans la bibliothèque. Consigner dans votre compte-rendu le logigramme.
d. Simuler et consigner le résultat dans votre compte-rendu
e. **Faite valider par l'enseignant**
f. Créer un symbole pour ce circuit

Exercice 3 : Horloge 1 Hz

La carte FPGA est cadencée par un quartz à 50MHz. On désire réaliser une horloge à 1 Hz à partir du quartz.

a. Quel doit être le rapport de division ?
b. Proposer à partir des circuits des exercices 1 et 2, une solution asynchrone. L'entrée du circuit sera le quartz et la sortie sera nommée H1Hz.
c. On affectera aux signaux aux I/O du design. Utiliser l'assignment editor.
50MHz : PIN_L1
H1Hz : PIN_W21 (LEDG4)

d. Vérifier expérimentalement que la ledg4 clignote toutes les secondes.
e. Créer un symbole graphique pour ce composant. Menu file/create-update et selectionner create symbol files for current file. Votre composant est maintenant accessible dans le bibliothèque des composants de votre projet.
f. **Faite valider par l'enseignant**

Exercice 4 : Chronomètre

a. Créer un nouveau fichier bdf et renommer le chronomètre.bdf
b. Afin que l'outil puisse le compiler, déclarer le en tant que « top level entity ». Clique droit de la souris sur le fichier sélectionné dans la fenêtre de gauche de l'environnement.

On souhaite réaliser un chronomètre qui s'incrémente toutes les secondes et qui affiche le résultat sur 4 afficheurs 7 segments. Pour cela on utilisera les blocks définis dans les exercices précédents.

c. Proposer un logigramme pour réaliser le chronogramme
d. **Faite valider par l'enseignant**
e. Programmer l'architecture en utilisant les blocks de ce TP et des TPs précédents.
f. Simuler et consigner le résultat dans votre compte-rendu
g. On affectera aux signaux aux I/O du design. Utiliser l'assignment editor.

Signal Name	FPGA Pin No.	Description
HEX0[0]	PIN_J2	Seven Segment Digit 0[0]
HEX0[1]	PIN_J1	Seven Segment Digit 0[1]
HEX0[2]	PIN_H2	Seven Segment Digit 0[2]
HEX0[3]	PIN_H1	Seven Segment Digit 0[3]
HEX0[4]	PIN_F2	Seven Segment Digit 0[4]
HEX0[5]	PIN_F1	Seven Segment Digit 0[5]
HEX0[6]	PIN_E2	Seven Segment Digit 0[6]
HEX1[0]	PIN_E1	Seven Segment Digit 1[0]
HEX1[1]	PIN_H6	Seven Segment Digit 1[1]
HEX1[2]	PIN_H5	Seven Segment Digit 1[2]
HEX1[3]	PIN_H4	Seven Segment Digit 1[3]
HEX1[4]	PIN_G3	Seven Segment Digit 1[4]
HEX1[5]	PIN_D2	Seven Segment Digit 1[5]
HEX1[6]	PIN_D1	Seven Segment Digit 1[6]
HEX2[0]	PIN_G5	Seven Segment Digit 2[0]
HEX2[1]	PIN_G6	Seven Segment Digit 2[1]
HEX2[2]	PIN_C2	Seven Segment Digit 2[2]
HEX2[3]	PIN_C1	Seven Segment Digit 2[3]
HEX2[4]	PIN_E3	Seven Segment Digit 2[4]
HEX2[5]	PIN_E4	Seven Segment Digit 2[5]
HEX2[6]	PIN_D3	Seven Segment Digit 2[6]
HEX3[0]	PIN_F4	Seven Segment Digit 3[0]
HEX3[1]	PIN_D5	Seven Segment Digit 3[1]
HEX3[2]	PIN_D6	Seven Segment Digit 3[2]
HEX3[3]	PIN_J4	Seven Segment Digit 3[3]
HEX3[4]	PIN_L8	Seven Segment Digit 3[4]
HEX3[5]	PIN_F3	Seven Segment Digit 3[5]
HEX3[6]	PIN_D4	Seven Segment Digit 3[6]

h. Vérifier expérimentalement le fonctionnement du chronomètre

i. Faite valider par l'enseignant