

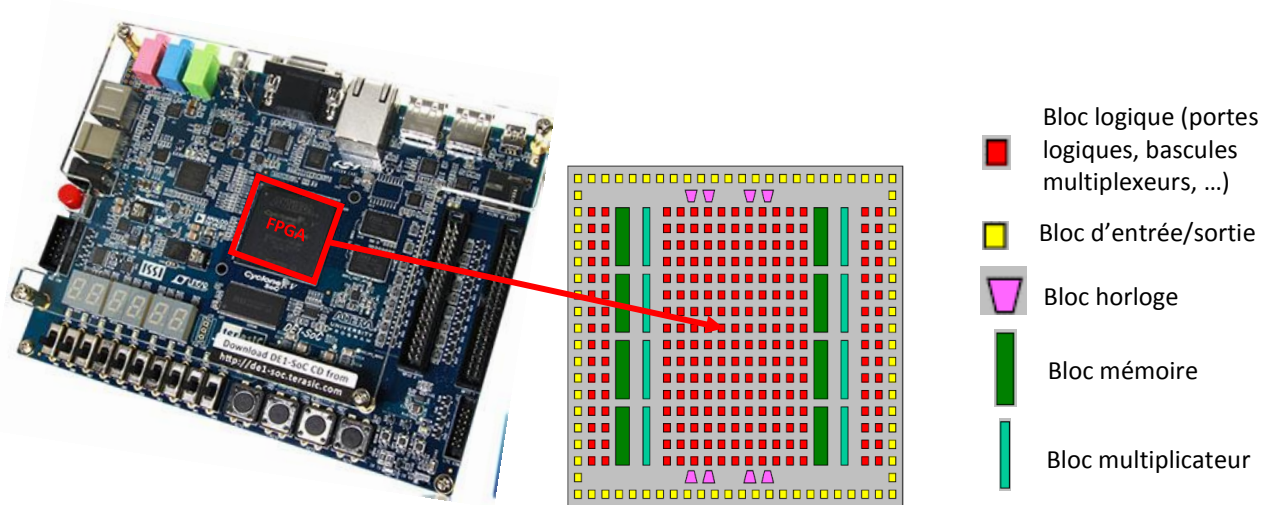
TP1

Conception de machines séquentielles en utilisant des bascules D

CHIRAZ TRABELSI
Et
ALEXANDRE BRIERE

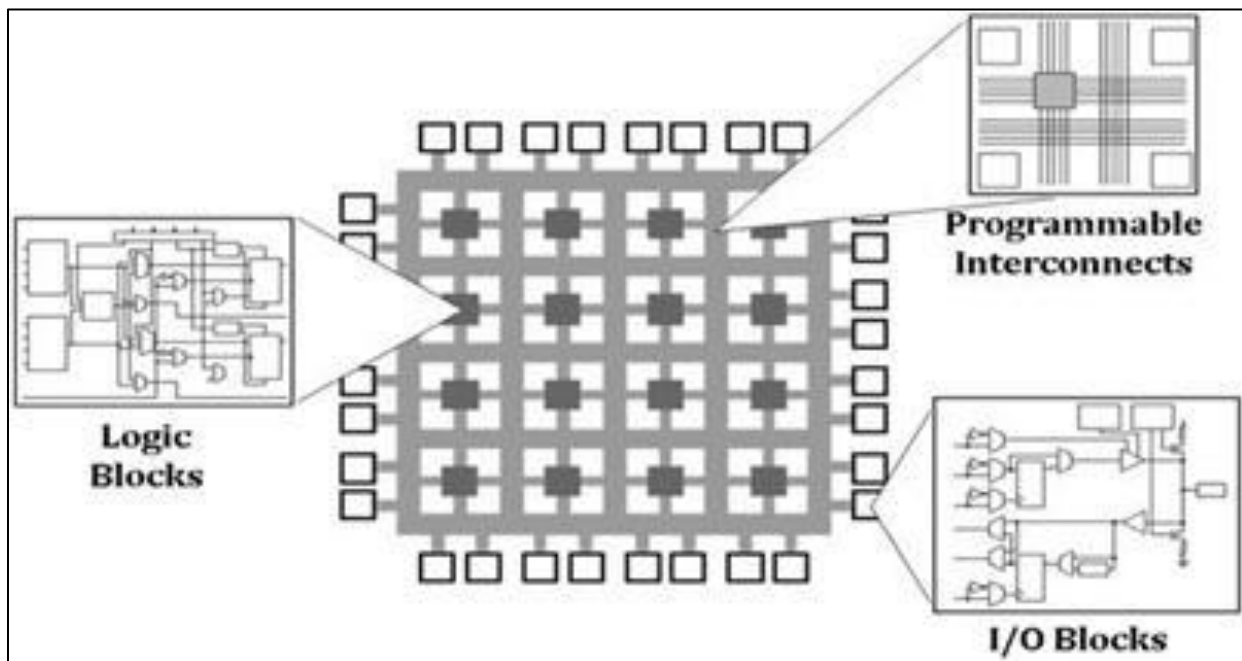
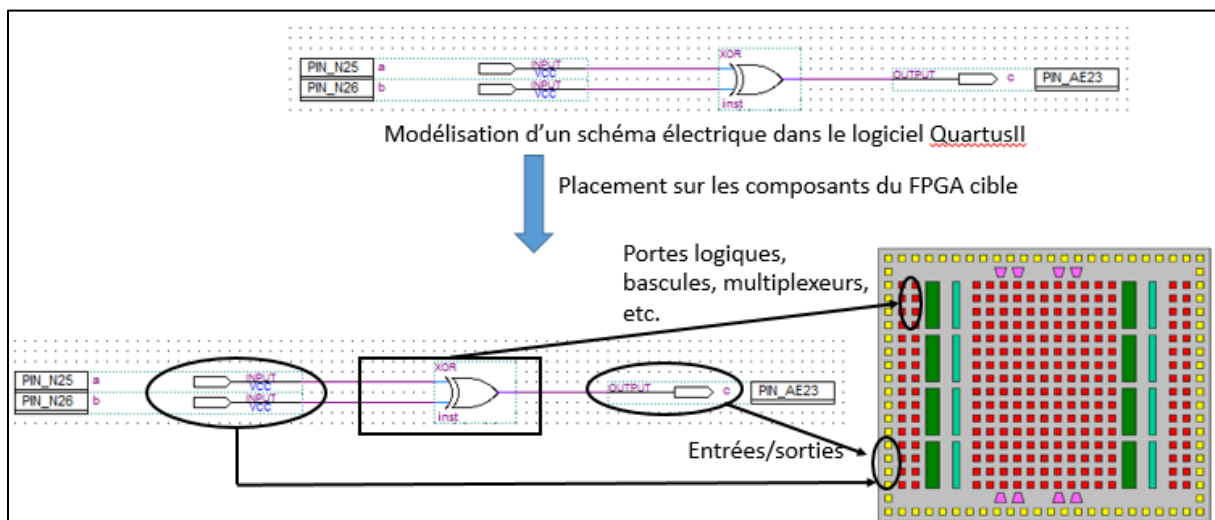


Introduction au FPGA



Le FPGA (Field-Programmable Gate Array) est une puce contenant une matrice de blocs logiques (portes logiques, bascules, multiplexeurs, etc.), mémoires, entrées/sorties et autres. C'est pour cette raison on utilise le terme « Gate Array ». Grâce au grand nombre de blocs contenus dans les FPGAs modernes (des millions de blocs), ils peuvent être utilisés pour implémenter un grand nombre de fonctionnalités. Cette implémentation se fait à l'aide d'un programme logiciel qui permet de charger une configuration donnée sur la puce FPGA. Grâce à cette méthode, on n'a plus besoin d'utiliser des composants logiques et mémoires discrets avec des câblages compliqués, tout est contenu dans une puce de quelque cm².

En ce qui concerne le terme « Field-Programmable », cela signifie que le FPGA est programmable sur le champ. C'est-à-dire que l'utilisateur achète un FPGA vierge (qui ne fait aucune fonctionnalité particulière) et c'est à lui de le programmer. La programmation d'un FPGA permet de lui donner les fonctionnalités qu'on veut. Pour faire des modifications à ces fonctionnalités, il suffit de reprogrammer le FPGA en y chargeant les nouvelles fonctionnalités. Le FPGA peut être programmé et reprogrammé le nombre de fois qu'on veut.

Structure interne des blocs du FPGA*Placement des composants d'un schéma électrique sur les blocs d'un FPGA*

Pour programmer un FPGA, le logiciel place les entrées/sorties du schéma électrique sur les blocs entrées/sorties du FPGA, place les portes logiques sur les blocs logiques du FPGA et fait les connexions entre les blocs utilisés

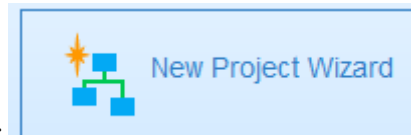
Exercice 1 (Prise en main du logiciel Quartus Prime et du logiciel ModelSim)

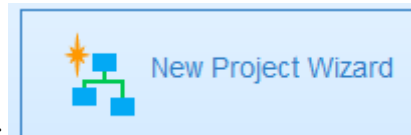
Création du projet

L'outil Quartus Prime ne génère pas un dossier qui englobe tous les fichiers d'un projet. Il génère plusieurs dossiers et fichiers dans le dossier courant. Pour cette raison, **il faut créer un nouveau dossier qui contiendra le projet**. Dans la figure suivante, on a créé par exemple, le dossier TD1 qui va contenir tous les fichiers et dossiers du projet projet_TD1.

Les outils de développement pour FPGA utilisent la notion de projet :

- Définition de l'environnement du projet (Cibles, outils, contraintes et...)
- Travail collaboratif, plusieurs développeurs peuvent intégrer leurs productions dans un même environnement.



Ouvrir QUARTUS Prime 17.0, puis cliquer sur  ou cliquer sur File → New Project Wizard. Le wizard (assistant) va nous guider dans la construction du projet.

Cliquer « Next ».

A screenshot of the "New Project Wizard" dialog box in Quartus Prime. The title bar says "New Project Wizard" with a close button. The main area is titled "Directory, Name, Top-Level Entity". It contains three text input fields, each with a browse button (three dots) to its right. The first field is labeled "What is the working directory for this project?" and contains the text "C:/Users/trabelsi/Desktop/TP1". The second field is labeled "What is the name of this project?" and contains "TP1_exo1". The third field is labeled "What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file." and contains "TP1_exo1". Below these fields is a button labeled "Use Existing Project Settings...". At the bottom of the dialog are five buttons: "< Back", "Next >" (highlighted in blue), "Finish", "Cancel", and "Help".

L'outil ne crée pas un dossier pour le projet mais un ensemble de fichiers dans le dossier spécifié dans le premier champ de la figure ci-dessus, donc il faut spécifier un dossier dans ce champ pour ne pas avoir des fichiers éparpillés.

Exemple: Si le champ contient le chemin vers le bureau, il crée des fichiers qui seront éparpillés sur le bureau. Pour cela, on a ajouté TP1 dans le chemin. Ce dossier sera créé s'il n'existe pas.

Ecrire le nom du projet (par Exemple : TP1_exo1)

Cliquer « Next ».

Dans la fenêtre qui s'affiche par la suite, choisir « Empty projet » puisqu'on va créer un projet vide.

Cliquer « Next ».

La fenêtre qui s'affiche par la suite est la fenêtre « Add Files » qui permet d'ajouter des fichiers sources au projet. Ici il n'y en a pas. Cliquer « Next » pour accéder à la fenêtre suivante.

Dans cette fenêtre, on va choisir le FPGA cible. On va utiliser la carte DE1-SoC, qui est équipée d'un FPGA Cyclone V, 5CSEMA5F31C6 (Vous pouvez taper cette référence dans le champ « Name filter » pour faciliter la recherche.

Family, Device & Board Settings

Device | Board

Select the family and device you want to target for compilation.
You can install additional device support with the Install Devices command on the Tools menu.

To determine the version of the Quartus Prime software in which your target device is supported, refer to the [Device Support List](#) webpage.

Device family

Family: Cyclone V (E/GX/GT/SX/SE/ST)

Devices: All

Target device

☐ Auto device selected by the Filter

☒ Specific device selected in 'Available devices' list

☐ Other: n/a

Show in 'Available devices' list

Package: Any

Pin count: Any

Core Speed grade: Any

Name filter: 5CSEMA5F31C6

☒ Show advanced devices

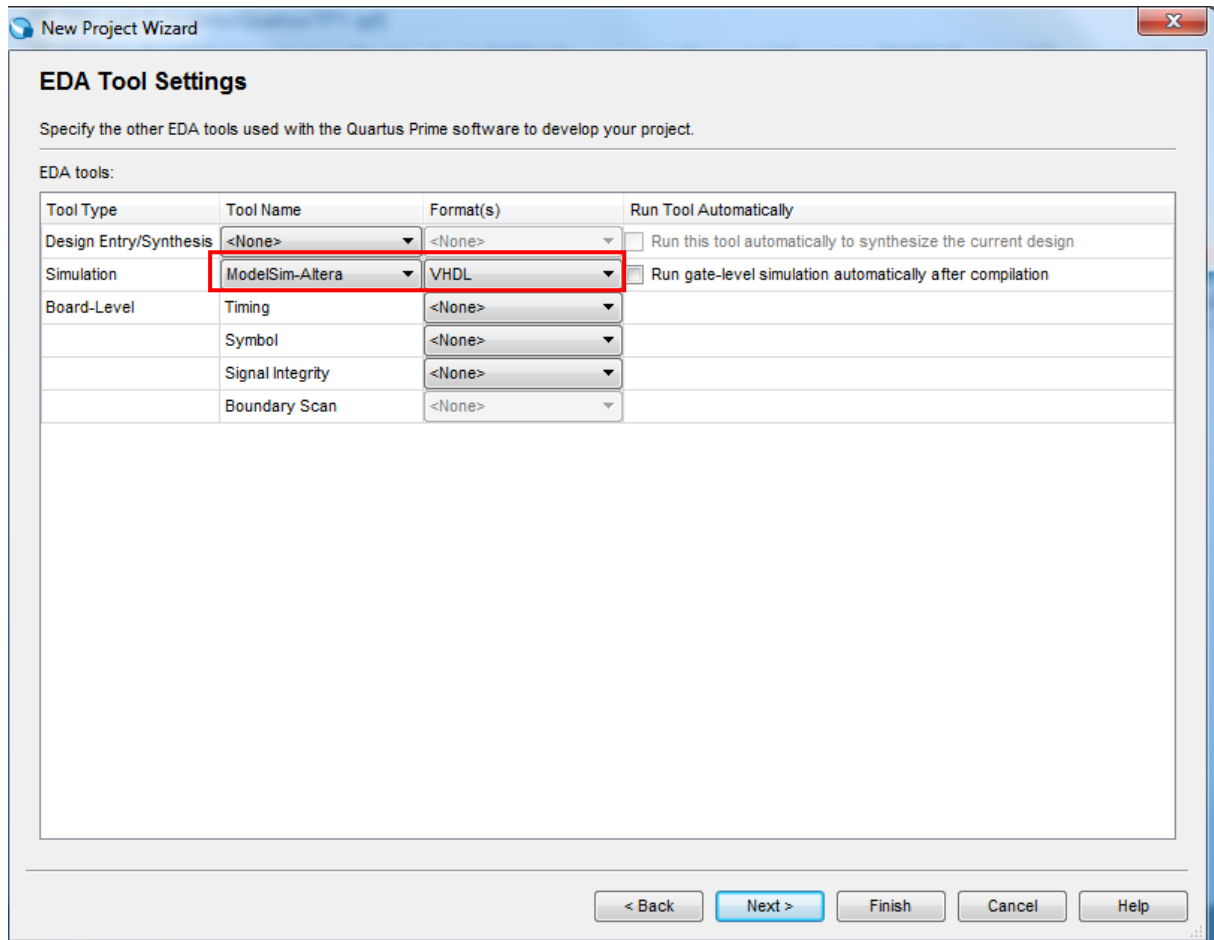
Available devices:

Name	Core Voltage	ALMs	Total I/Os	GPIOs	GXB Channel PMA	GXB Channel PCS	PC
5CSEMA5F31C6	1.1V	32070	457	457	0	0	0

< Back | Next > | Finish | Cancel | Help


Sélectionner cette carte dans le Wizard (voir page suivante) et cliquer « Next ».

Sur la fenêtre suivante, vérifier que le simulateur ModelSim et le langage VHDL sont bien sélectionnés.




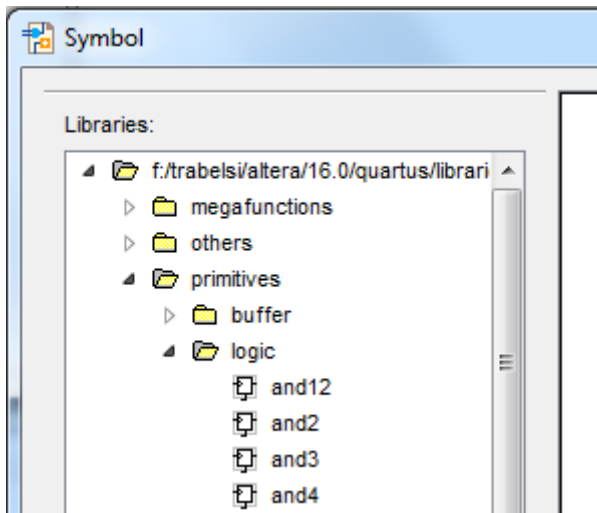
Vérifier la configuration sur la fenêtre suivante puis cliquer « Finish ». Le projet est créé et configuré, il est maintenant possible d'ajouter des fichiers de description schéma, machine à états, etc.

Création d'un schéma

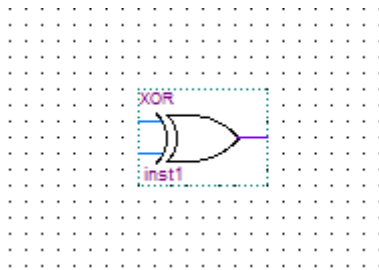
Cliquer « File » → « New » ou  puis sélectionner « Block Diagram/schematic File » dans la liste « Design Files ». L'éditeur de schéma s'ouvre.


L'exercice propose de réaliser un système simple contenant une porte logique XOR (ou exclusif). Le système a deux entrées (a et b) et une sortie (c).

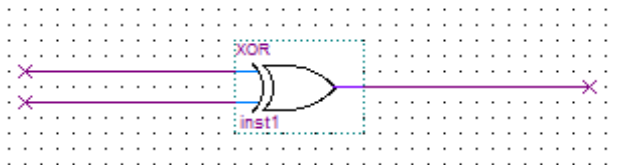
Pour faire le schéma électrique, cliquer  puis choisir « primitives » → « logic » → xor.




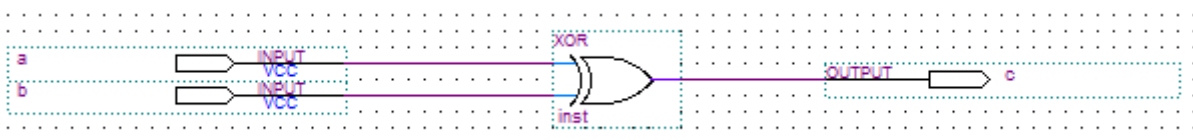
Placer la porte XOR comme ci-dessous.



Cliquer sur  pour ajouter des fils de façon à réaliser le schéma ci-dessous.




Cliquer sur  et placer les entrées et sorties comme le montre la figure ci-dessous, nommer-les (double-clic pour éditer le nom). Attention : vérifier que les entrées et la sortie sont bien connectées à la porte logique (utiliser la flèche  pour sélectionner une entrée/sortie et tirer pour voir si c'est bien connecté).



Enregistrer le schéma (exemple TP1_exo1.bdf) en prenant soin de vérifier le dossier de destination (celui du projet)

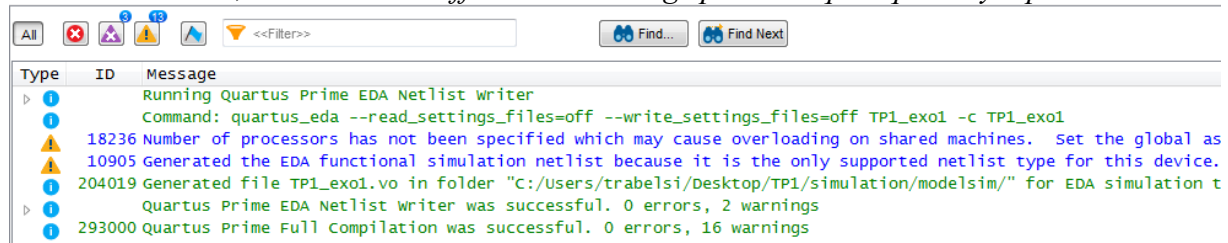
Compilation


Pour avoir une compilation rapide, il faut désactiver l'option d'optimisation avancée du placement du système sur le FPGA. Cette option est activée par défaut mais donne un temps de compilation trop lent. On n'a pas besoin de cette optimisation pour notre projet. Pour désactiver cette optimisation, aller dans Assignements → Settings → Compiler Settings, appuyer sur Advanced Settings (Fitter) et sélectionner off dans le champ « Advanced Physical Optimization ». Valider par OK.

Cliquer sur  pour compiler le projet. (Compter quelques minutes de compilation).

QUARTUS vérifie le schéma. Il crée des rapports pour toutes les étapes de la compilation.

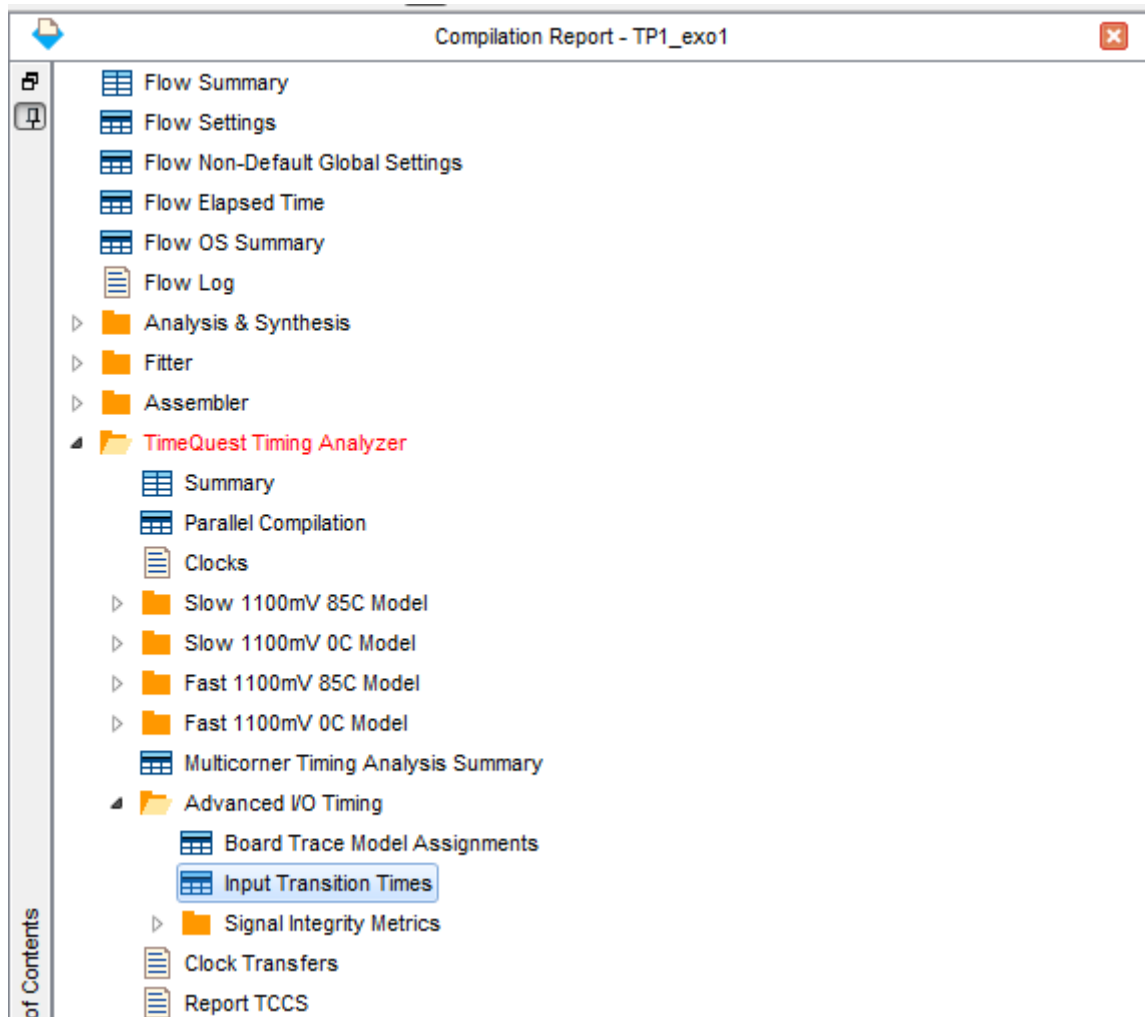
Si tout est correct, La console va afficher un message pour indiquer qu'il n'y a pas d'erreurs.



Les « warnings » indiquent que le compilateur a dû prendre des décisions qui peuvent influencer sur le résultat attendu ou que la description du système est incomplète. Ici, par exemple, aucune horloge n'est utilisée. Cela est indiqué dans le warning «No clocks defined in design » (à visualiser dans la fenêtre des warning  en bas de l'écran). Généralement les « warnings » peuvent être ignorés.

Temps de propagation des signaux

La compilation génère plusieurs rapports à visualiser dans la fenêtre Compilation Report.



Parmi ces rapports, on s'intéresse au rapport temporel qui donne les temps de propagation des signaux. Pour cela, il faut aller dans TimeQuest Timing Analyzer → Advanced I/O Timing → Input Transition Times.

En cliquant dessous, vous allez voir que le signal entrant en (a) a besoin de 2ns pour arriver à la sortie (c) et celui entrant en (b) a besoin de 2ns pour arriver à la sortie (c). Cela sera vérifié par la suite dans la simulation.

Input Transition Times				
	Pin	I/O Standard	10-90 Rise Time	90-10 Fall Time
1	a	2.5 V	2000 ps	2000 ps
2	b	2.5 V	2000 ps	2000 ps

Simulation

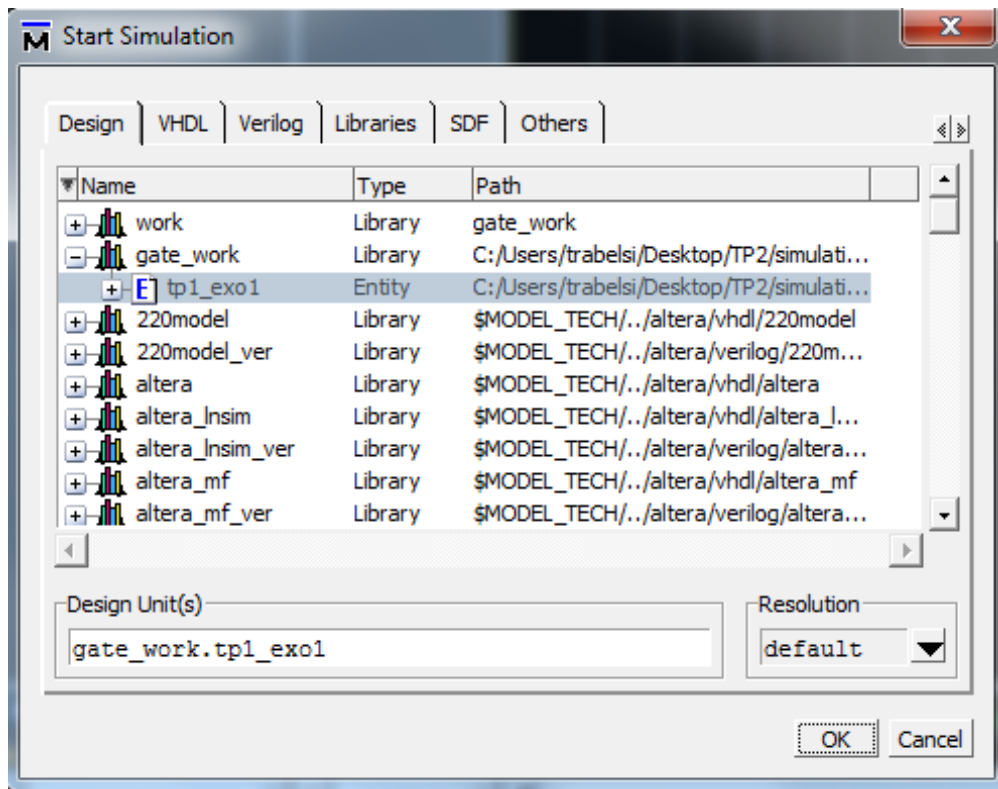
Le test du projet peut être effectué par le simulateur ModelSim.

Cliquer « Tools → Run Simulation Tool → Gate Level Simulation... ».

Le Logiciel ModelSim va s'ouvrir. Si le programme ne se lance pas cela veut dire que le chemin d'accès de ModelSim n'est pas encore enregistré dans Quartus. Pour ce faire, aller dans Tools → Options → EDA Tool Options → Modelsim-Altera et ajouter le chemin d'accès : C:\intelFPGA_lite\17.0\modelsim_ase\win32aloem

Dans ModelSim, cliquer "Simulate → Start Simulation..."

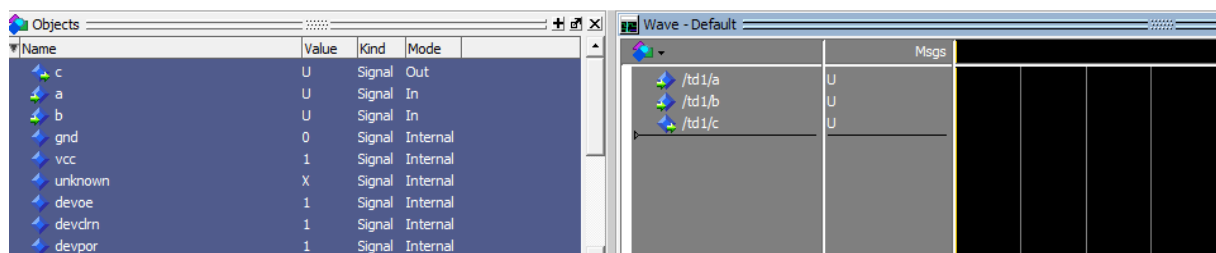
Onglet “Design” → “gate_work” → “structure”



Cliquer « OK » pour fermer la fenêtre « Start Simulation ».

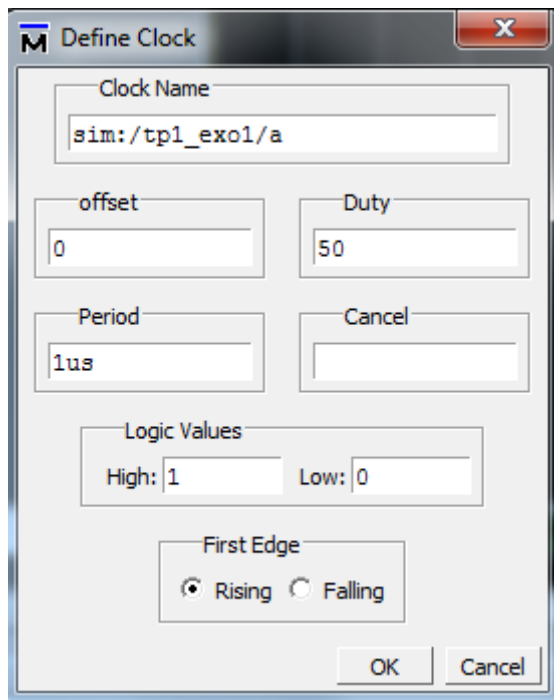
Le simulateur propose dans la fenêtre « Objects » les « éléments » constituant la description. Sélectionner a, b, et c puis clic-droit « add to wave » ou tirer ces trois éléments dans la fenêtre Wave.

Il est possible de déplacer les signaux simplement avec la souris, afin de les visualiser dans un ordre particulier.

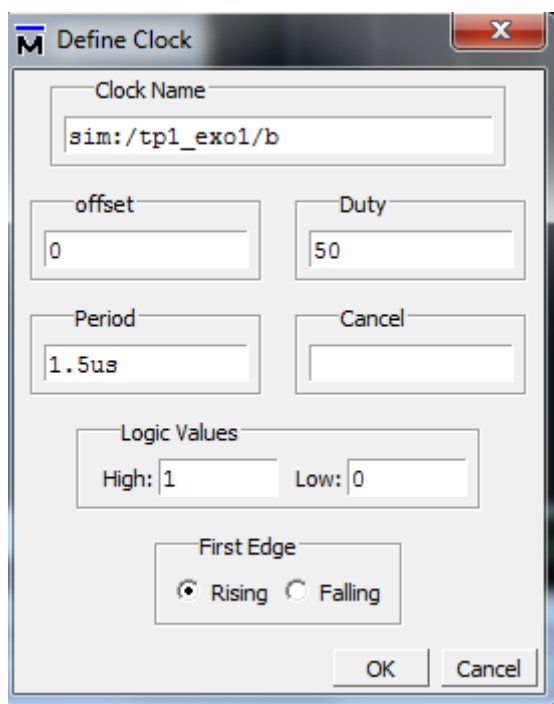


Simulation des signaux d'entrée :

Cliquer- droit sur l'entrée a dans la fenêtre Wave. Cliquer sur « Clock » dans le menu et créer un signal périodique de période de 1us et un rapport cyclique de 50% (c'est-à-dire la période du signal est 50% à 1 et 50% à 0).



Faire la même chose pour pour l'entrée b avec une période de 1.5us.



Noter dans la fenêtre « Transcript » les instructions engendrées

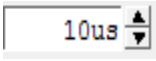


```
add wave -position end sim:/tp1_exo1/a
add wave -position end sim:/tp1_exo1/b
add wave -position end sim:/tp1_exo1/c
force -freeze sim:/tp1_exo1/a 1 0, 0 {500000 ps} -r 1us
force -freeze sim:/ tp1_exo1/b 1 0, 0 {750000 ps} -r 1.5us
```

Pour automatiser le lancement de ces commandes pour la prochaine simulation, vous pouvez copier ces lignes dans un nouveau fichier texte à sauvegarder avec une extension « .do » dans le dossier « simulation/modelsim ». Le lancement de ce fichier s'effectue alors avec la commande « do » dans la fenêtre « Transcript », ex : do test.do

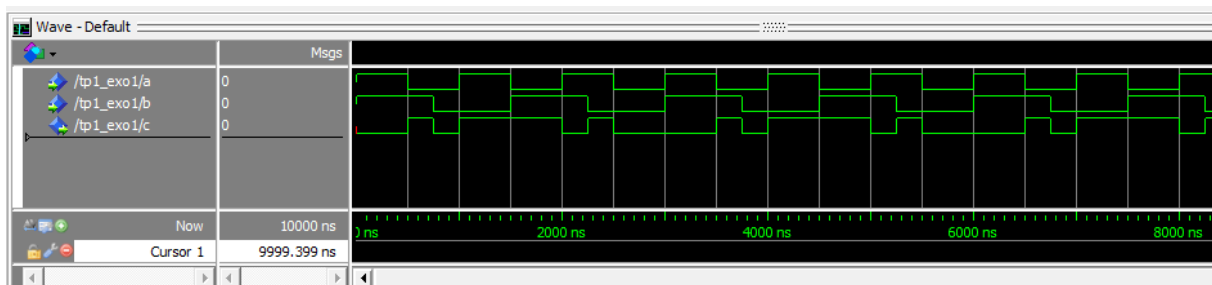
Pour tester cette méthode, créer un script comme décrit ci-dessus. Sélectionner a, b et c and la fenêtre « Wave », faire un clic-droit et sélectionner « Edit → Delete ».

Maintenant, exécuter le script dans la fenêtre « Transcript » en bas de l'écran avec la commande « do test.do » (remplacer le nom « test » avec celui de votre script).


Remarquer que a, b et c réapparaissent de nouveau dans la fenêtre « Wave ».

Ecrire 10us comme durée de simulation  puis cliquer sur run . Le simulateur déroule 10us de simulation. Cliquer dans la fenêtre de simulation puis  pour visualiser l'ensemble.

Vous devez obtenir un résultat qui ressemble à ceci :



Notez que la sortie c correspond bien au résultat attendu.

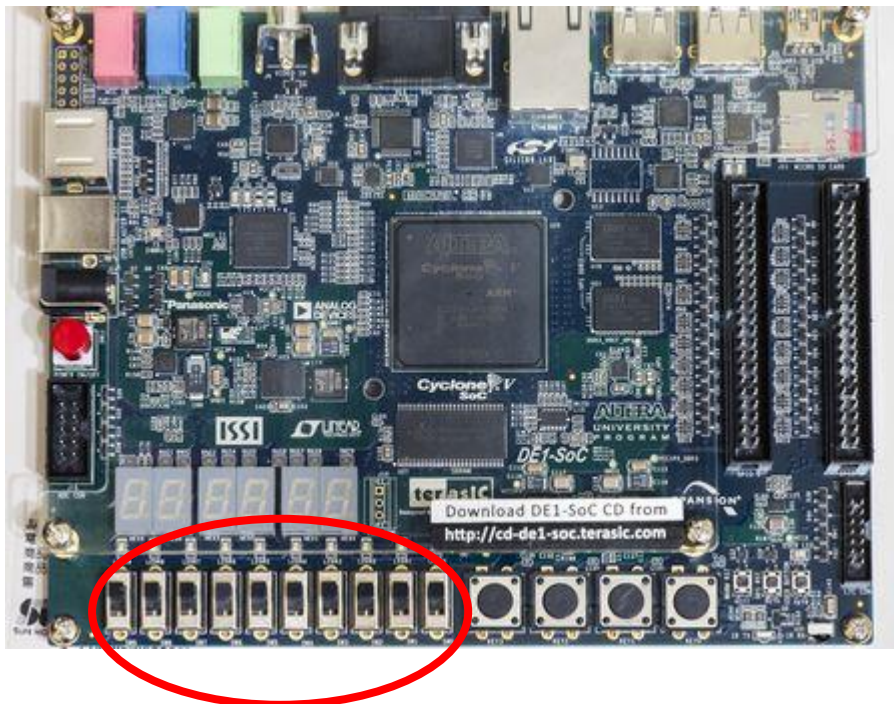
Attention : si vous lancez la simulation pour une durée donnée et vous rendez compte que vous vous êtes trompés sur une valeur d'un signal d'entrée, parfois c'est obligatoire de faire un restart de la simulation (Simulation → Restart), rectifier les valeurs des entrées et click de nouveau sur  pour lancer une nouvelle simulation. En effet, la simulation doit avoir les bonnes valeurs des entrées dès le début (instant t=0).

Appeler l'encadrant pour valider la simulation. Vous êtes censés expliquer à l'encadrant le résultat de la simulation (fonctionnement XOR).

Implémentation physique sur la cible (KIT DE1-SoC)

Le KIT DE1-SoC est équipé d'interrupteurs (switches) et de LEDs qui vont permettre de tester en réel le projet.

On va utiliser deux switches pour implémenter les deux entrées (a et b) et une LED pour modéliser la sortie (c).



Le fichier [DE1-SoC_User_manual.pdf \(Moodle\)](#) contient des tableaux indiquant les numéros des pins des entrées et sorties de la carte DE1-SoC (Tableau 3-6. page 25 pour les switches et Tableau 4 -8 page 26 pour les LED).

Dans cet exercice on va utiliser les pins suivants :

Signal	Interface	Pin
a	SW0	PIN AB12
b	SW1	PIN AC12
c	LED0	PIN V16

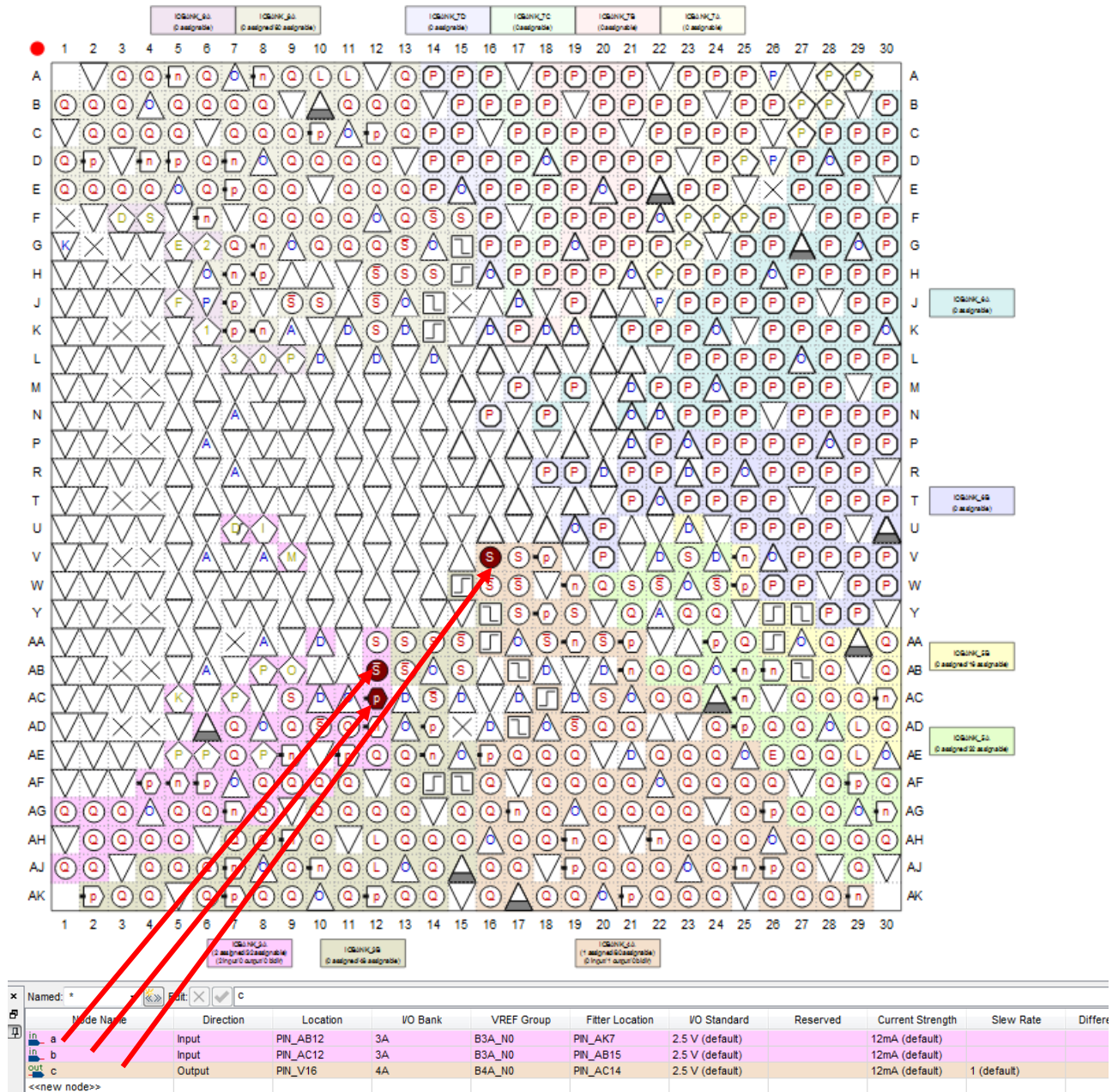
L'outil « PIN PLANER » permet d'attribuer des broches physiques à des entrées/sorties.

Pour lancer Pin Planner, dans le menu « Assignment », cliquer sur « Pin Planer ».

Pin Planer affiche un plan de câblage du FPGA et permet l'affectation des entrées/sorties. Les pins du FPGA sont codés par une lettre et un nombre. Les lettres figurent sur les lignes et les nombres sur les colonnes. Ainsi, la proche AB12 se situe à la ligne AB, colonne 12.

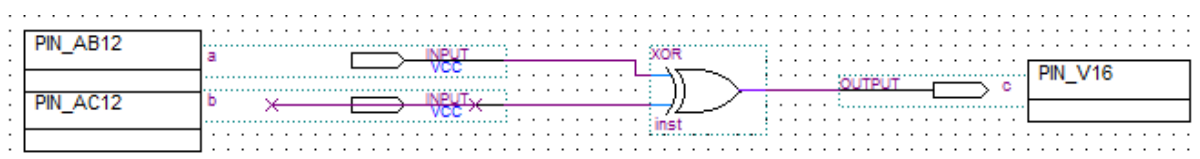
Configurer le brochage de a, b et c comme suit :

Top View - Wire Bond Cyclone V - 5CSEMA5F31C6



Pour associer les entrées-sorties à leurs pins, il y a deux possibilités. La première possibilité est de cliquer sur l'entrée/sortie en bas et de faire un « drag-drop » vers la broche associée dans la carte. La deuxième possibilité est d'écrire le numéro de la broche dans le champ « Location ».

Fermer PIN Planer, les numéros de broches apparaissent sur le schéma.

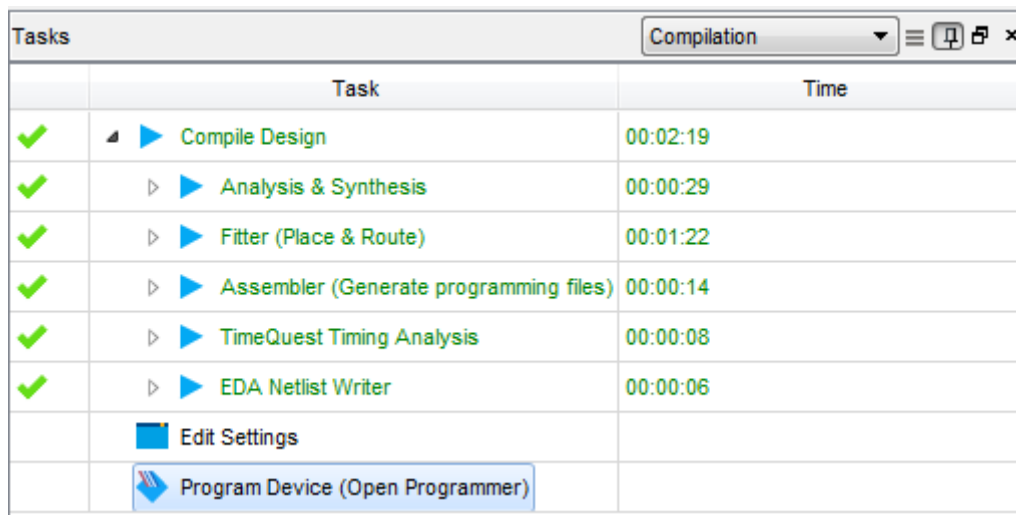


Recompiler le projet en cliquant sur .

Placer un câble entre l'USB **Blaster** Port du KIT DE1-SoC sur un port USB du PC.

Alimenter le KIT et appuyer sur le bouton ON/OFF, (*L'application par défaut du KIT teste les LEDs et les afficheurs*).

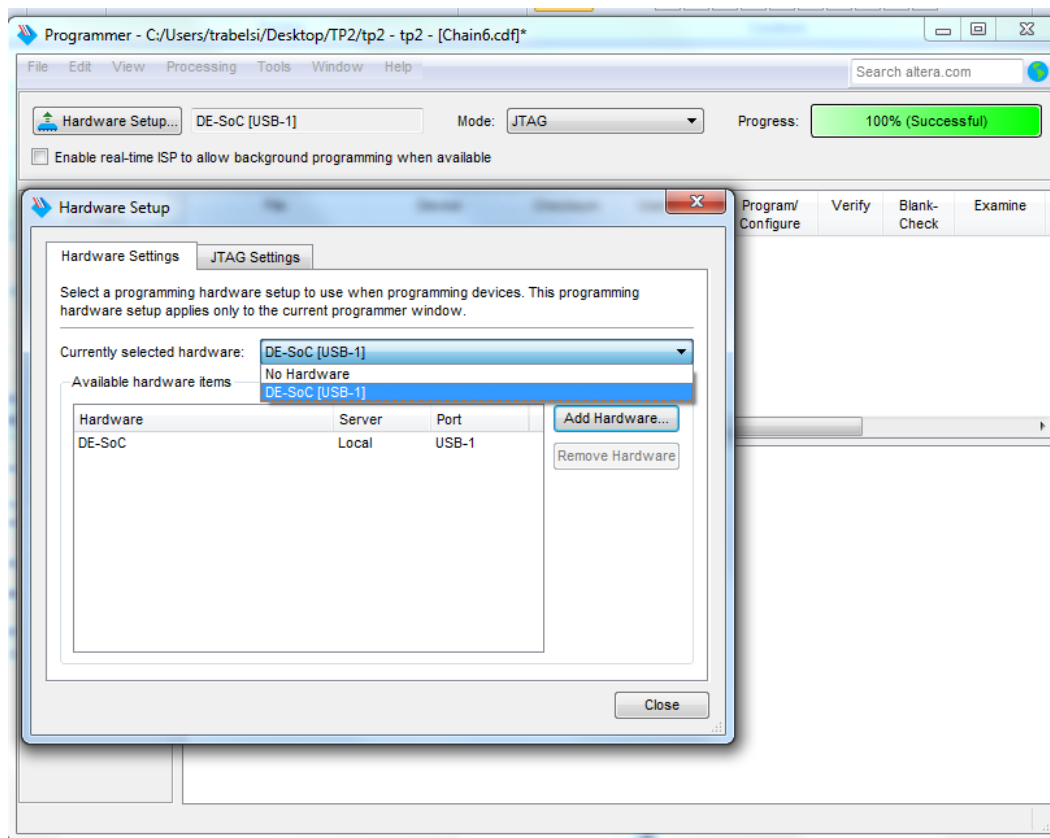
Sur QUARTUS, dans la fenêtre Tasks, double-clique « Program Device »



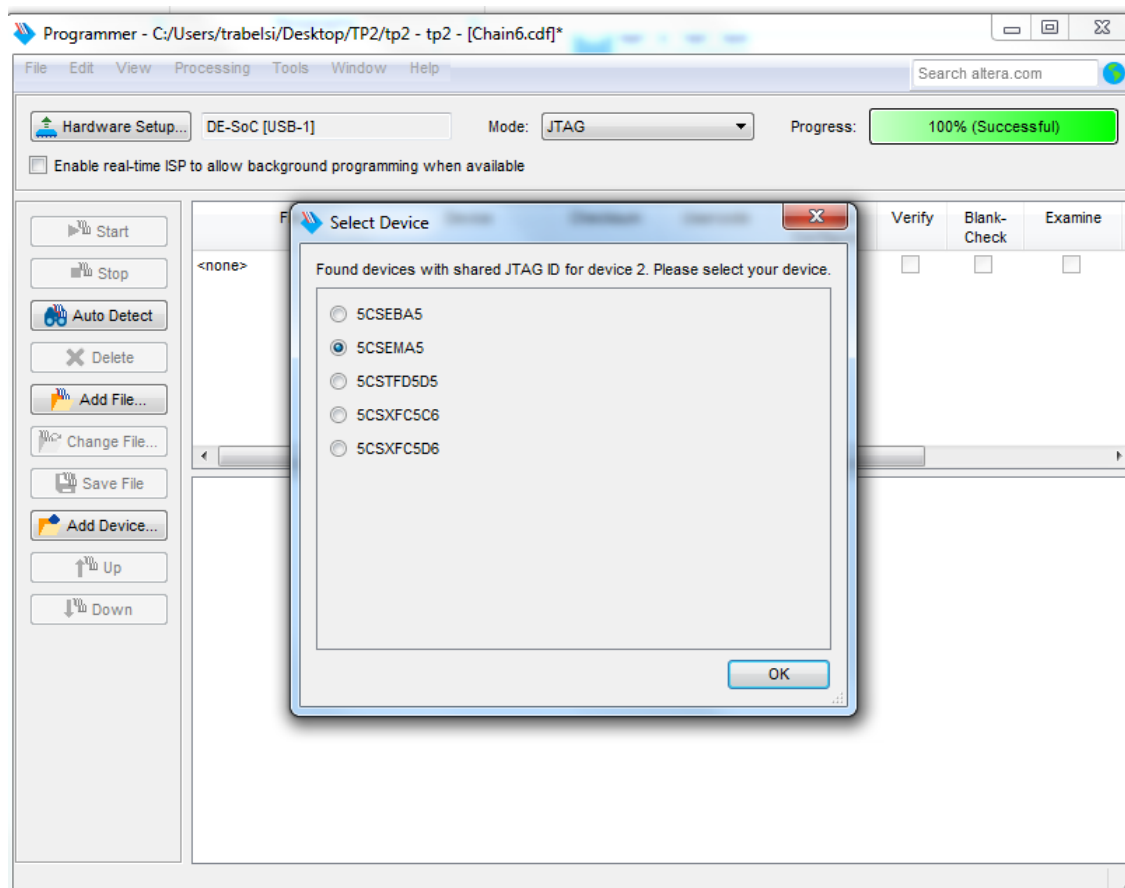
Tasks			Compilation			
	Task		Time			
✓	▶ ▶	Compile Design	00:02:19			
✓	▶ ▶	Analysis & Synthesis	00:00:29			
✓	▶ ▶	Fitter (Place & Route)	00:01:22			
✓	▶ ▶	Assembler (Generate programming files)	00:00:14			
✓	▶ ▶	TimeQuest Timing Analysis	00:00:08			
✓	▶ ▶	EDA Netlist Writer	00:00:06			
		Edit Settings				
		Program Device (Open Programmer)				

Normalement le programmeur « USB-Blaster » est automatiquement détecté.

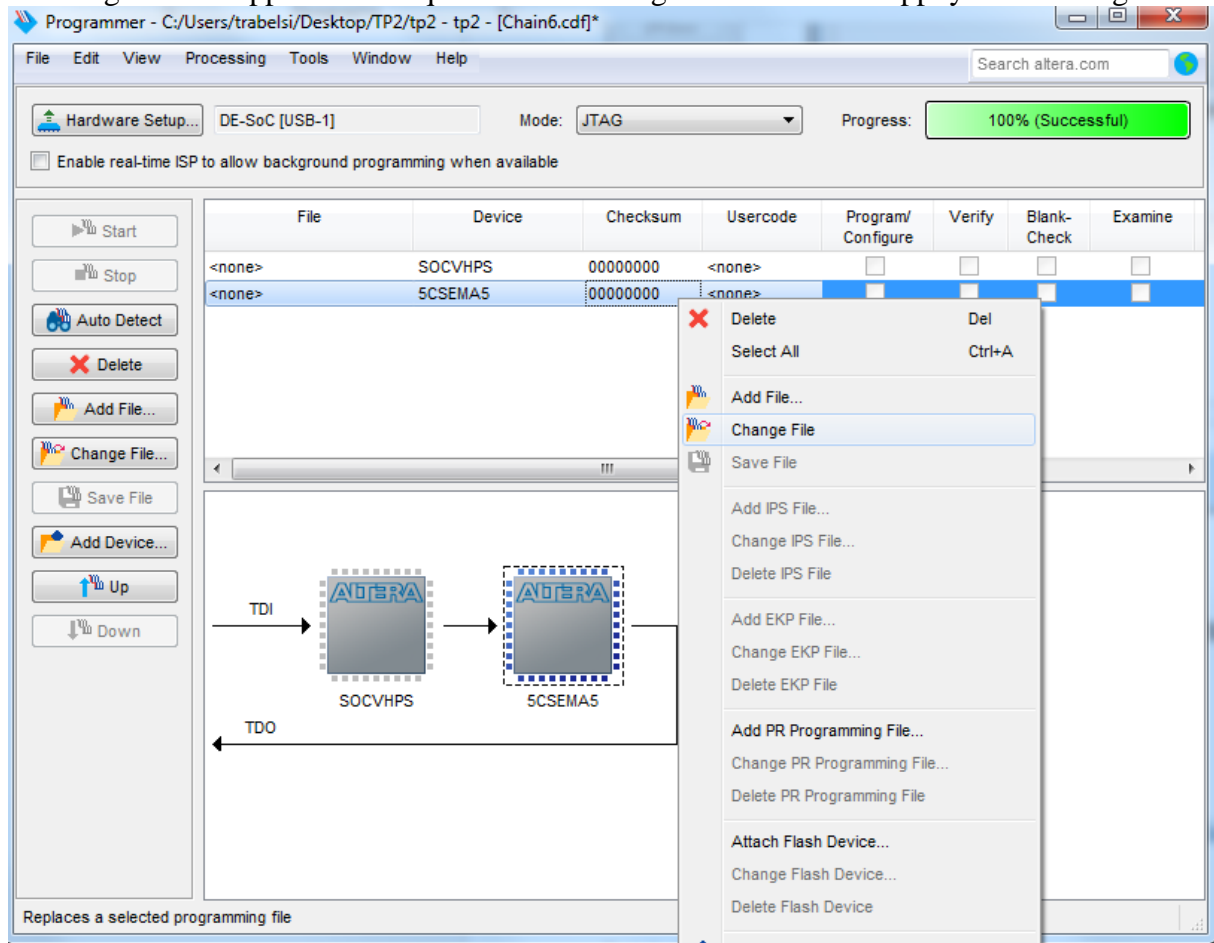
Dans le cas contraire, cliquer sur « Hardware Setup », double-cliquer sur « DE-SoC[USB-1] »



Appuyer sur le bouton « Auto Detect », choisir 5CSEMA5 et cliquer OK.

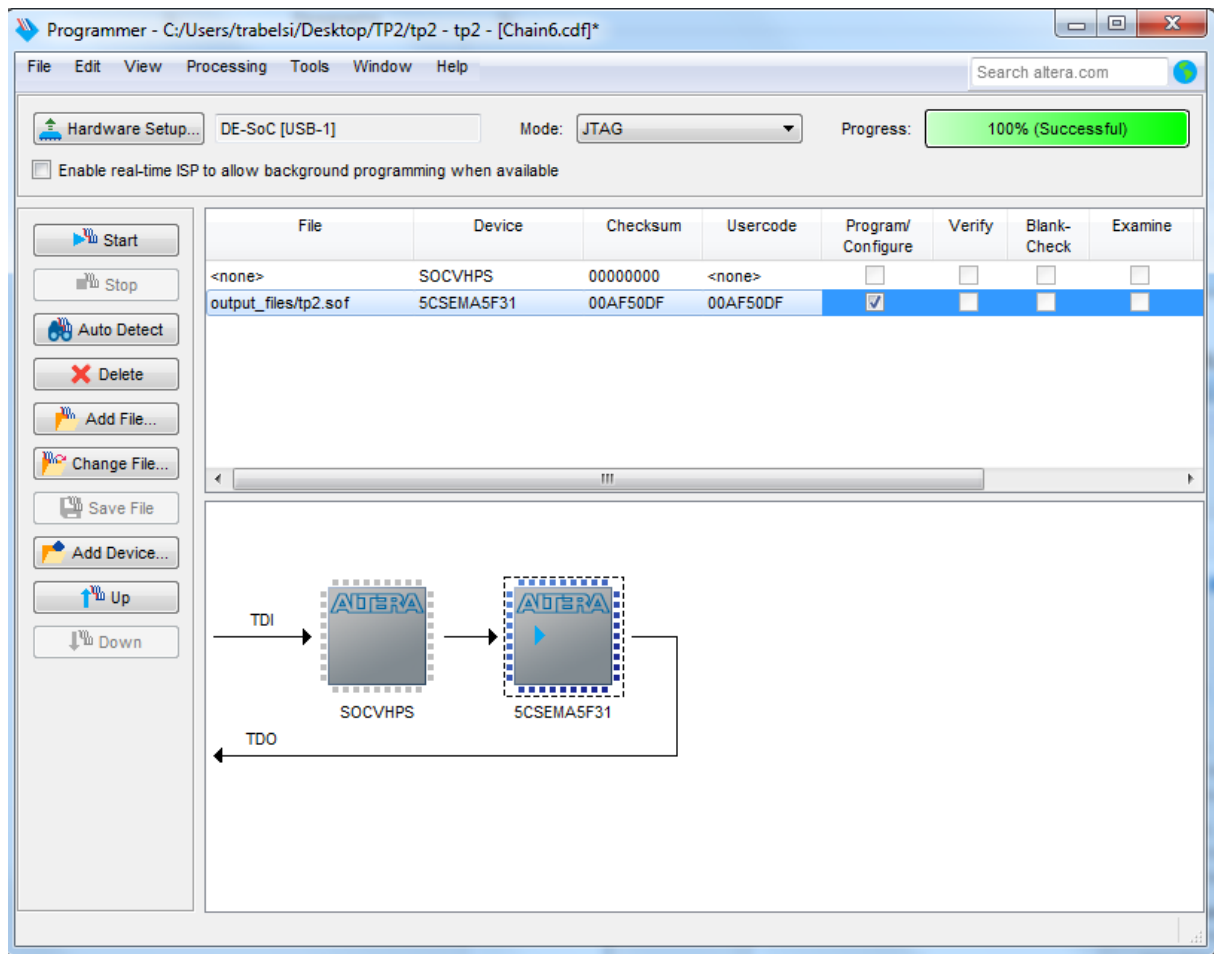


Deux lignes vont apparaître. Cliquer-droit sur la ligne 5CSEMA5 et appuyer sur Change File.



Aller dans le dossier « output_files » et sélectionner le fichier « .sof » qui s'y trouve.

Cocher la case « Program/Configure » et appuyer sur le bouton « Start ». La barre de progress doit afficher « 100% (Successful) » si le programme est bien chargé sur la carte.

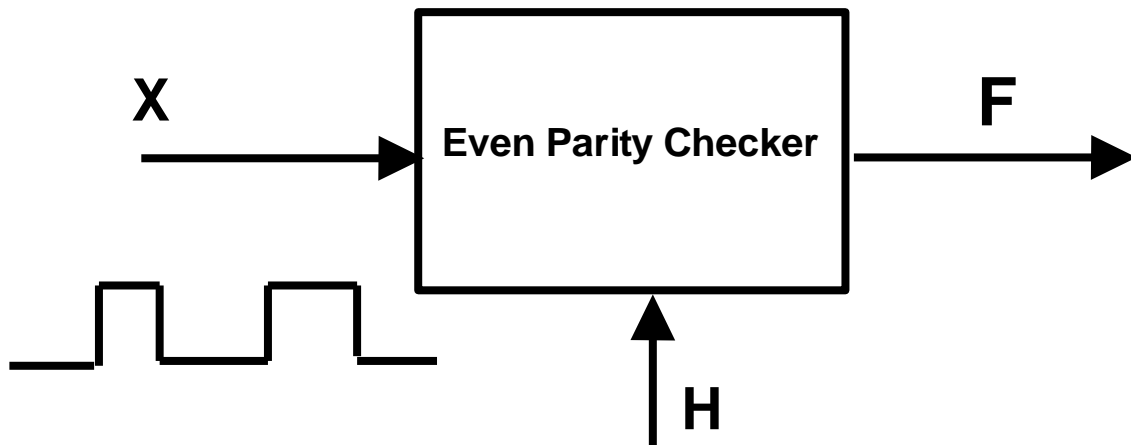


Tester alors la fonction XOR sur les interrupteurs SW0, SW1 et la LED0.

Appeler l'encadrant pour valider l'implémentation.

Exercice 2 (Conception d'une machine séquentielle : vérificateur de parité paire)

Soit à réaliser un circuit vérificateur de parité paire.



Le fonctionnement est le suivant :



A chaque top d'horloge le circuit reçoit une entrée binaire X. Sa sortie F doit passer à '1' si un nombre pair de '1' a été détecté au niveau de l'entrée, sinon elle est au niveau '0'. Au départ, la sortie est à '1' puisqu'au départ on a zéro « 1 » en entrée.

Exemple : Voici les valeurs de sorties F pour une exécution du système pour 8 cycles d'horloge (réception de 8 valeurs sur l'entrée X)

X : 01011001
F : 10010001

1. Ce système a donc deux états: Pair et Impair, l'état Pair étant l'état initial. Modéliser ce fonctionnement par une machine de Moore sur papier. Le codage à utiliser pour les états est le codage binaire. Déterminer les équations des entrées de bascules et l'équation de la sortie F.

Appeler l'encadrant pour valider les équations.

2. Pour réaliser le système avec des bascules D sur le logiciel, suivre ces étapes :
 - a) Créer un nouveau projet Quartus comme vu dans l'exercice 1.
 - b) Cliquer  puis sélectionner « Block Diagram/schematic File » dans la liste « Design Files » pour ajouter un schéma.
 - c) Dessiner votre schéma. Pour ajouter des bascules D, cliquer  puis choisir « primitives » → « storage » → « dff ». L'entrée horloge de la bascule est une entrée ordinaire (comme l'entrée X) à renommer avec le nom « CLK ».
 - d) Sauvegarder votre schéma (exemple Parite.bdf)
3. Désactiver l'optimisation avancée du placement (voir Exercice 1) et compiler le projet
4. Lancer le simulateur (Tools → Run Simulation Tool → Gate Level Simulation)

Pour la simulation, on va vérifier que si on entre une séquence d'entrées « 01011001 », la séquence de sorties correspondantes est bien « 10010001 ».

Pour simuler cette séquence d'entrées, vous pouvez copier les commandes dans le cadre ci-dessous dans la console du simulateur. Une meilleure façon de faire est de copier dans un fichier « run.do » (vous pouvez utiliser n'importe quel autre nom avec l'extension .do) à sauvegarder dans le dossier « **simulation/modelsim** » de votre projet. L'utilisation de ce script se fait en tapant la commande « do run.do » dans la console. Cette méthode vous permet de lancer le script en entier et ne pas être obligés de copier-coller les commandes à chaque fois.

```
#creation de signal horloge de periode 100ns

force sim:/Parite/CLK 0 0ns, 1 50ns -repeat 100ns

#La sequence de x: 01011001, 0 au départ, 1 après 100ns (un cycle d'horloge), .....
force sim:/Parite/X 0 0ns, 1 100ns, 0 200ns, 1 300ns, 1 400ns, 0 500ns, 0 600ns, 1 700ns

#lancer la simulation pour 1us
run 1us
```

Vérifier que la séquence des sorties correspond bien au résultat prévu.

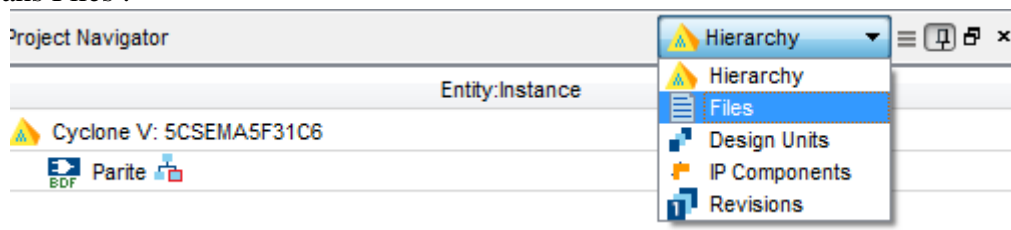
Appeler l'encadrant pour valider la simulation.


Simulation au niveau RTL (simulation fonctionnelle)

RTL (Register Transfer Level) est un niveau de description qui considère le système en termes de transfert de signaux entre registres et ports d'entrées/sorties sans entrer dans les détails d'implémentation physique (traduction en termes de blocs dans le FPGA). **Ce type de simulation est plus rapide que la simulation au niveau porte logique** qu'on a vue précédemment. Il ne contient pas les informations temporelles (exemple temps de propagation). Par contre, il permet de simuler facilement les signaux internes. La simulation RTL se base sur une description du système utilisant un langage de haut niveau appelé VHDL (VHSIC Hardware Description Language). VHSIC est une abbréviation de (Very High Speed Integrated Circuit).

Avant de lancer ce type de simulation, il faut d'abord générer la description VHDL du système. Pour ceci, il faut ouvrir le schéma dans le projet et aller dans « File → Create/Update → Create HDL design from current file », choisir VHDL. Quartus crée un fichier VHDL à partir du schéma. Ayant le même nom que le fichier schéma avec l'extension « .vhd ». Ce fichier n'est pas visible par défaut dans le projet, il faut donc l'ajouter au projet. Puisque ce fichier a le même nom que le fichier schéma, il faut enlever le fichier schéma du projet en suivant la méthode suivante :

Aller dans Files :



Clic-droit sur « Files » → « Add/Remove Files in Projects ... ». Dans la fenêtre qui s'affiche ajouter le fichier .vhd en le cherchant en cliquant sur . Sélectionner le fichier .vhd, puis sur « Add ». Cliquer « OK ».

Afficher le fichier VHDL et examiner sans contenu. Vous pouvez reconnaître les entrées/sortie déclarées au début du fichier.

Le fichier généré ressemble à ceci :

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY work;
ENTITY Parite IS
    PORT
    (
        X : IN STD_LOGIC;
        CLK: IN STD_LOGIC;
        F : OUT STD_LOGIC
    );
END Parite;

ARCHITECTURE bdf_type OF Block2 IS
    .....
END bdf_type;
```

Dans cette description le fichier commence par la déclaration des librairies à utiliser (équivalent des « #include » en langage C)

Par la suite le terme « ENTITY » permet de déclarer une entité, les ports de cette entité sont déclarés avec PORT, en indiquant s'il s'agit d'un port d'entrée (IN) ou de sortie (OUT). Le terme « STD_LOGIC » indique le type des ports qui est un booléen.

Par la suite, la description du système est décrite dans « ARCHITECTURE », où vous pouvez reconnaître l'équivalent en VHDL des bascules et portes logiques que vous avez utilisées. Le code généré n'initialise pas les signaux. Pour cette raison, il faut faire quelques modifications sur les deux lignes commençant par signal en ajoutant := '0' :

```
SIGNAL    SYNTHESIZED_WIRE_0 : STD_LOGIC:= '0';
SIGNAL    SYNTHESIZED_WIRE_1 : STD_LOGIC:= '0';
```

Cela permet d'initialiser les signaux à 0. Si cette initialisation n'est pas faite, la valeur de la sortie qui est liée au signal SYNTHESIZED_WIRE_1 sera inconnue et la simulation ne peut pas se faire correctement.

Pour pouvoir lancer la simulation RTL correspondante au fichier VHDL, il faut le définir en Top-Level. Puisqu'on a un fichier schéma ayant le même nom, si vous lancez la compilation tout de suite, un message indiquant que vous avez un fichier en double va s'afficher. Pour éviter ce problème dans la vue « Files », cliquez-droit sur le fichier schéma et sélectionnez « Remove

File From Project ». Cela ne va pas supprimer le fichier de votre disque dur mais il le détachera de l'arborescence du projet.

Compiler le projet.

Cliquer sur « Tools → Run simulation tools → **RTL Simulation** », ce qui va lancer le simulateur en mode « RTL simulation fonctionnelle ». Procéder ensuite comme précédemment mais en choisissant cette fois ci WORK « bdf_type », et non plus GATE-WORK, puis OK (pas de fichier sdf, le matériel n'étant pas simulé).

Vous pouvez simuler les entrées/sortie ainsi que les signaux internes (entrées et sorties des bascules).



Pour cela, vous pouvez relancer le même fichier « run.do » créé précédemment.

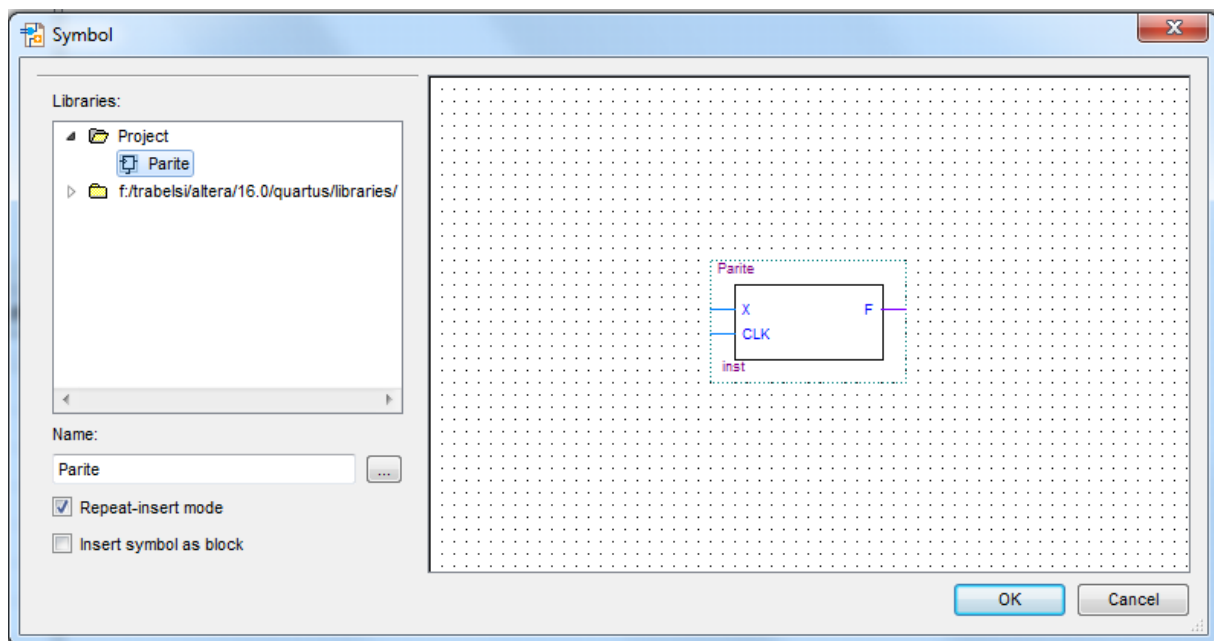
Vérifier qu'on obtient le même résultat que la simulation précédente.

Appeler l'encadrant pour valider la simulation. Vous êtes censés expliquer le résultat de simulation des signaux SYNTHESIZED_WIRE_0 et SYNTHESIZED_WIRE_1 (entre autres).

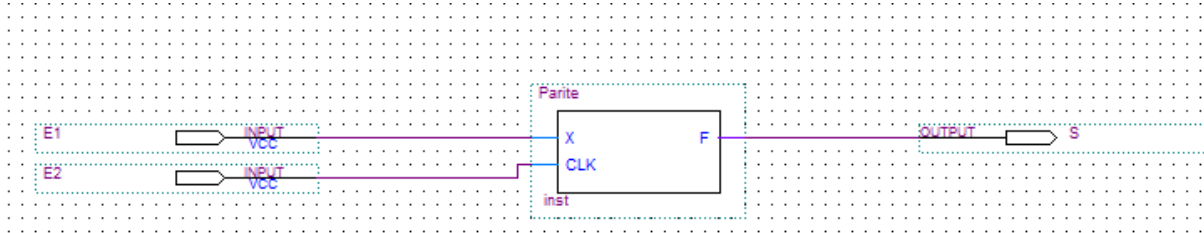
Génération de schéma à partir d'un fichier VHDL

Pour des schémas contenant un grand nombre de composants, la conception d'un schéma électrique devient complexe, pour cette raison il est plus facile de décrire le système à travers le langage VHDL. Le schéma électrique est par la suite généré automatiquement.

Pour tester cette option, il faut ouvrir le fichier .vhd dans le projet puis cliquer sur « File → Create/Update → Create Symbol Files for Current File ». Le fichier Parite.bsf est créé (bsf est l'extension des symboles graphiques des composants) mais il n'est pas visible dans le projet. Pour pouvoir utiliser ce symbole, créer un nouveau fichier schéma (Cliquer « File » → « New » ou  puis sélectionner « Block Diagram/schematic File » dans la liste « Design Files »). Cliquer , vous allez trouver le nouveau Symbole Parite dans le répertoire « Project ».



Ajouter ce symbole au nouveau fichier. Ajouter des ports d'entrée et de sortie pour obtenir le schéma suivant :



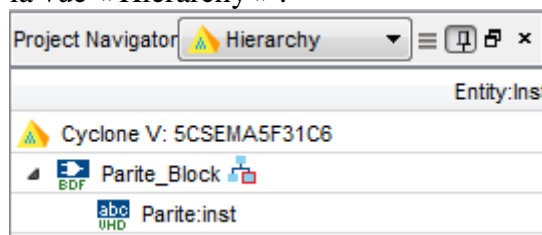
Cette méthode peut être utilisée, par exemple, pour générer plusieurs symboles à partir de fichiers VHDL décrivant chacun un sous-ensemble du système. Par la suite, on peut utiliser un schéma qui connecte tous ces symboles pour modéliser le système final.

Sauvegarder le schéma (exemple : Parite_Block.bdf)

Aller dans la vue « Files », clic-droit sur le schéma et sélectionner « Set As Top-Level Entity ».

Attention, si votre projet contient plus d'un fichier (dans notre cas, le fichier VHDL et le fichier schematic qu'on vient de créer), vous devez sélectionner l'entité "Top-Level". Dans notre cas, c'est le fichier schematic car c'est la racine du système qui contient une référence au fichier VHDL.

La nouvelle racine du projet devient ce nouveau schéma. Vous pouvez le vérifier en allant dans la vue « Hierarchy » :



Recompiler le projet pour mettre à jour. Il faut recompiler le projet à chaque fois que vous changez de fichier racine que cela soit pris en compte dans les étapes suivantes (simulation, chargement sur la carte, etc.).

Schéma RTL

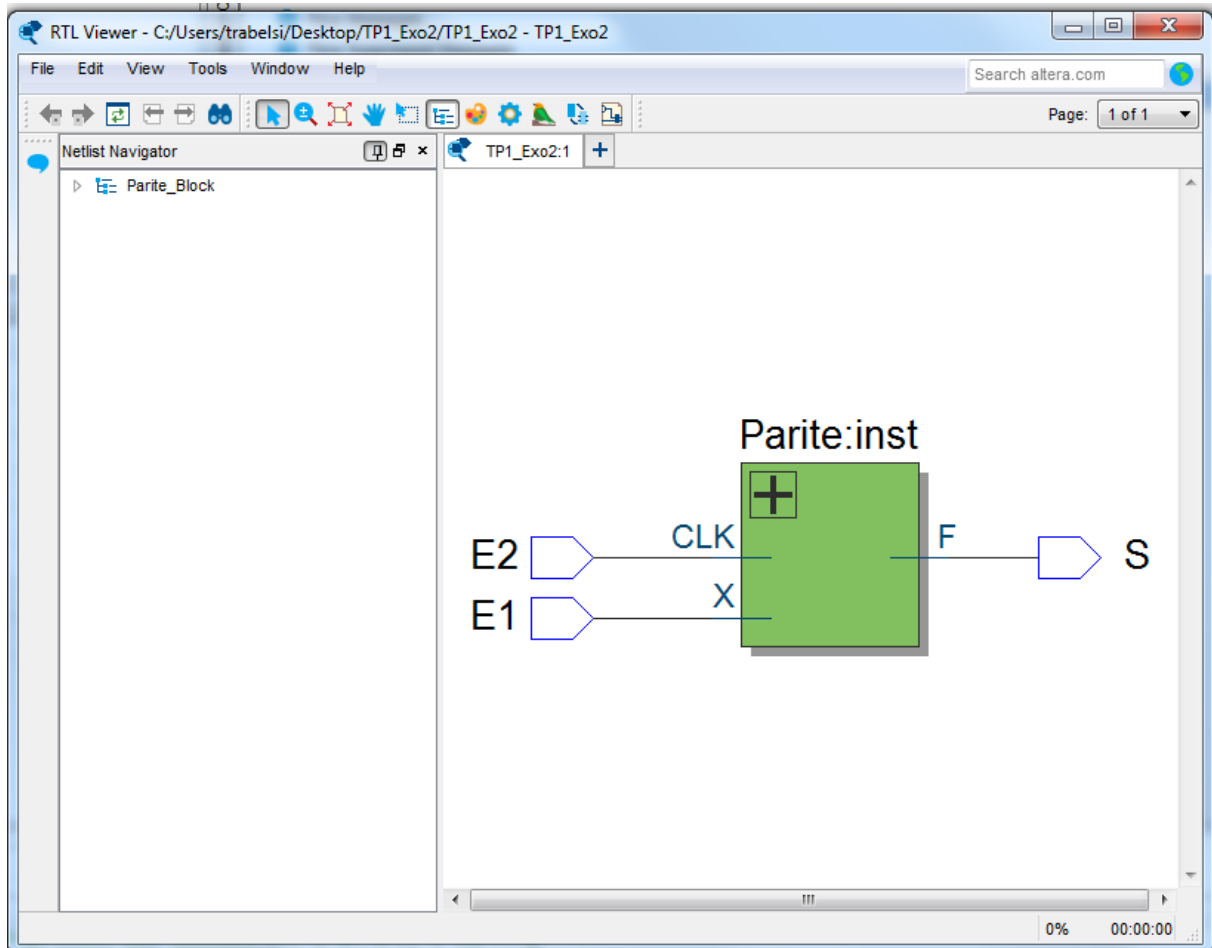
Dans cette section, vous allez voir comment un fichier VHDL est traduit en bascules et portes logiques pour être implémenté physiquement sur FPGA.

A partir d'une description VHDL ou d'un schéma, on peut générer une représentation RTL et une représentation porte-logique (Gate). La représentation RTL est une représentation haut-niveau qui ne prend pas en compte les types de bascules et portes logiques disponibles sur le FPGA cible et qui peut traduire le fichier VHDL en utilisant n'importe quels types de composants qui se trouvent dans la bibliothèque du logiciel (alors que sur un FPGA, les blocs disponibles sont de certains types). Pour cette raison, une simulation RTL n'entre pas dans les détails du FPGA cible, et est plus rapide puisqu'elle n'a l'information sur les temps de propagation, etc.

La représentation porte-logique (Gate) traduit le système en utilisant les blocs qui se trouve sur

le FPGA cible. Une simulation à ce niveau est plus lente que le niveau RTL mais plus précise par rapport aux contraintes de temps.

Pour afficher la représentation RTL, cliquer « Tools → Netlist Viewers → RTL Viewer ». Quartus crée le symbole correspondant à la description fonctionnelle.



Cliquer sur le signe « + » pour visualiser les bascules/portes logiques utilisées.

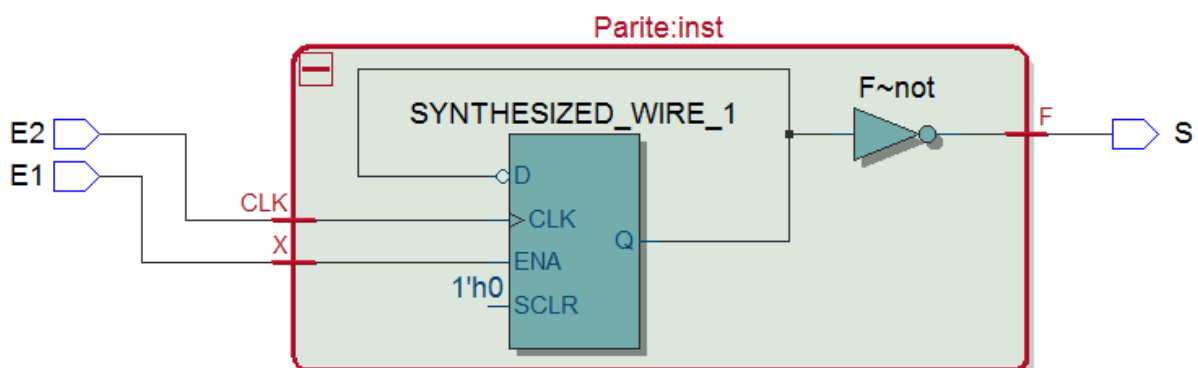
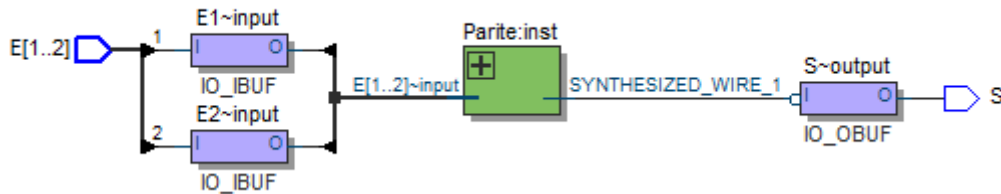


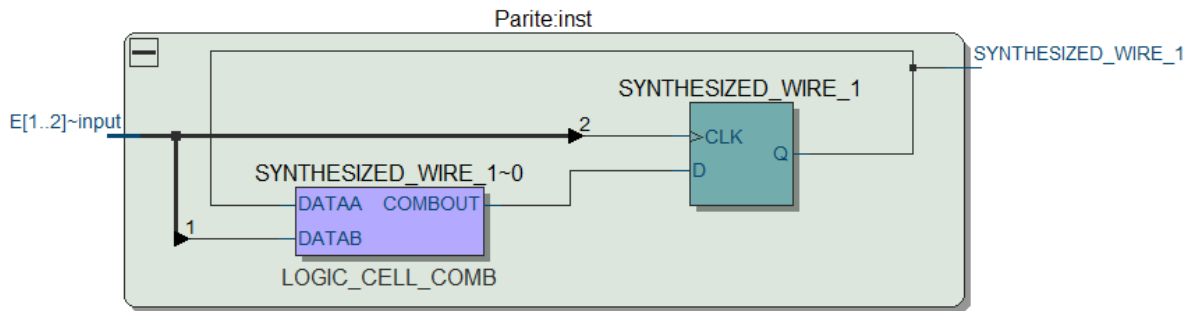
Schéma GATE

Il est possible de voir le schéma après l'intégration dans le FPGA (traduction du système selon les blocs disponibles sur le FPGA cible). Cliquer « Tools → Netlist Viewers → Technologie

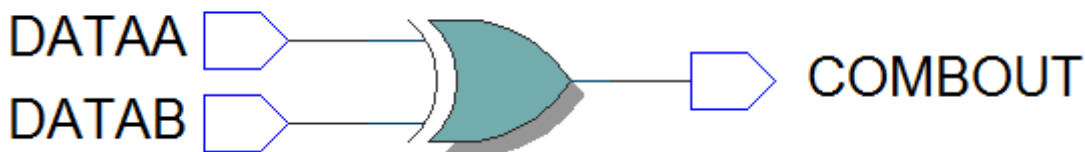
Map Viewer (Post Mapping) ». Ce schéma montre comment le système sera implémenté en utilisant les ressources de la carte cible.



Cliquer sur Parite pour visualiser son contenu.



Le terme « LOGIC_CELL_COMB », cela montre l'utilisation d'un bloc logique pour l'implémentation de l'équation de l'entrée de la bascule. Clic-droit sur le bloc et sélectionner propriétés. Aller dans l'onglet F à droite. Le contenu du bloc s'affiche.



En considérant les trois figures ci-dessus, on reconstruit les équations de $S = F = \text{Not}(Q)$ et de $D = E1 \text{ xor } Q = X \text{ xor } Q$.

Appeler l'encadrant pour valider les schémas obtenus. Vous êtes censés expliquer pourquoi vous avez deux implémentations différentes entre les deux vues.

Essai sur la carte

Tester votre système sur la carte en simulant l'entrée et l'horloge par deux switches.

Appeler l'encadrant pour valider.