6GEI228 – Systèmes Digitaux

Laboratoire #1

Introduction à Quartus Prime et aux portes logiques

Hiver 2018

1. Objectifs

- Explorer le logiciel Quartus Prime Lite V16.1
- Apprendre à dessiner les circuits
- Apprendre à simuler les circuits
- Apprendre à implémenter sur la plaquette DE10-Lite

2. Méthodologie

Durant le laboratoire, vous aurez à créer des circuits simples avec le logiciel Quartus Prime Lite V16.1. Vous aurez aussi à les simuler en stimulant les entrées et en observant les sorties dans un logiciel. Finalement, vous aurez à implémenter le circuit sur la plaquette DE10-Lite et voir que votre circuit fonctionne réellement. L'ordre des étapes à suivre est illustré à la figure 1.

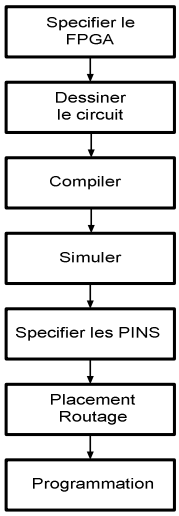

Fig. 1. Étapes à suivre pour l'implémentation FPGA

3. Préparation et Théorie

Un FPGA est une structure qui permet d'implémenter un grand nombre de circuits numériques. Il est composé de plusieurs portes logiques qui ne sont pas connectées ensemble, mais qui peuvent l'être grâce à un réseau d'interconnexion interne programmable. Ceci se fait grâce à un logiciel qui s'appelle Quartus Prime Lite

Dans le logiciel, nous sommes capables faire toutes les étapes qui mènent à la réalisation du prototype : dessiner un circuit, vérifier son fonctionnement avec des simulations et configurer le FPGA.

Il existe plusieurs FPGA et chacun doit être configuré de façons différentes. Il est donc important de spécifier à Quartus Prime Lite quel FPGA on utilise. Le modèle que nous utiliserons au laboratoire est le **10M50DAF484C7G**.

Par la suite, on peut commencer la conception de notre projet. Il existe plusieurs façons de concevoir un système dont deux qui vont nous intéresser particulièrement : graphiquement et en programmation. Dans ce laboratoire, il sera question de seulement utiliser la méthode graphique.

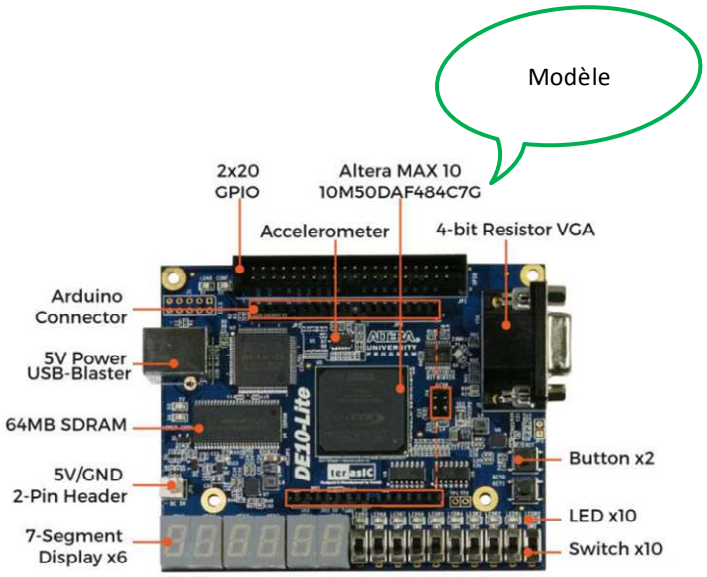

Fig. 2. Plate-forme de développement (vue du dessus).

Vous aurez à dessiner un schéma avec des portes logiques et les interconnecter ensemble. Vous aurez aussi à dire quels signaux doivent provenir de l'extérieur (entrées et sorties) et spécifier où ils sont connectés à l'extérieur.

Une fois que vous avez fini de dessiner, le logiciel doit transformer votre dessin en format qu'il peut comprendre. On parle de la *compilation*. Durant cette partie, il

transforme votre design en portes logiques sous un format compréhensible pour lui. Par la suite, nous avons deux options : 1) Suivre les étapes pour le mettre sur le FPGA ou 2) faire une simulation pour vérifier le comportement de celui-ci. Dans ce laboratoire, nous allons commencer par faire la simulation. La simulation nous permet de mettre des données hypothétiques en entrée et voir si la sortie est bonne. De cette façon, on pourra valider le comportement du système. Pour ce faire, il faut décider de ce qu'on veut mettre à l'entrée et de connaître ce qu'on s'attend de voir à la sortie.

Une fois complété, on peut prendre les étapes pour le mettre sur le FPGA. La compilation prend notre circuit et produit une liste de portes logiques qui sont connectés ensembles. Cependant, le FPGA est énorme et contient une très grande quantité de portes. Où se trouvent ces portes physiquement ? On passe donc par une étape qui s'appelle le *placement et le routage*. De façon purement pratique, ces étapes se font automatiquement quand notre design est simple. Dans certains cas, on va vouloir mettre des contraintes et contrôler le processus un peu plus. Pour ce laboratoire, on va laisser l'outil prendre toutes les décisions : tout se fera automatiquement quand on clique sur le bouton. Avant de cliquer, il ne faut surtout pas oublier **d'assigner les entrées et les sorties**. Certains signaux vont entrer et sortir de la puce, mais par où ? Il y a des centaines de places possibles : certaines pins sont connectées à des boutons d'autres sont connectés à des lumières etc. En consultant le manuel de référence (en ligne : cherchez DE10-Lite manuel sur *Google, cliquez sur Ctrl + clic pour suivre le lien : https://www.google.ca/search?q=DE10+lite+manuel&oq=DE10+lite+manuel&aqs=chrome..69i57.10631j0j8&sourceid=chrome&ie=UTF-8*, on devrait être en mesure de savoir où connecter nos entrées et sorties. Les assignations des numéros des PINS sont dans le manuel DE10 Lite aux pages 24 à 31 inclusivement que vous trouverez sur le web.

Finalement, on fait la programmation en envoyant les données dans le FPGA.

4. Travail demandé

Configuration de base

Ouvrez le logiciel Quartus Prime 16.1.

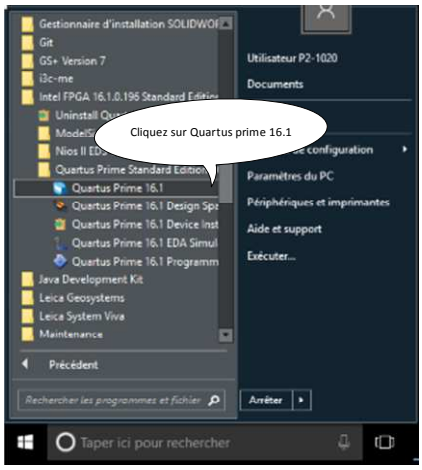

Fig.3. Ouverture du logiciel.

Commençons par se créer un projet :

Cliquez sur la case : *New Project Wizard*

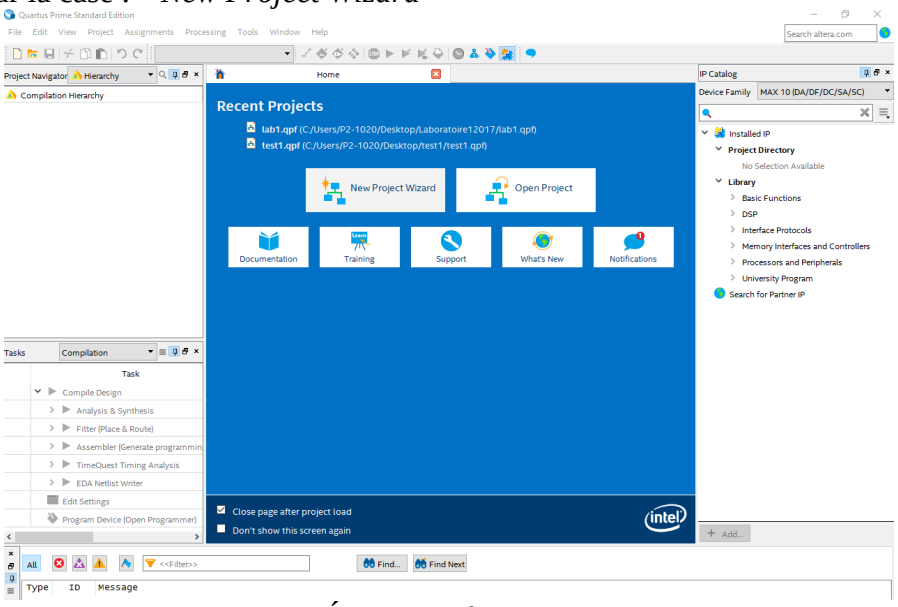

Fig.4. Écran de départ du projet.

Une nouvelle fenêtre apparaîtra et devrait ressembler à celle de la Figure 5. Il faut maintenant se choisir un répertoire où mettre les fichiers. Créez-vous un répertoire *laboratoire_1_prime* sur le bureau, créez un nouveau dossier avec le nom du projet.

Donnez le nom du projet : *lab1*. La dernière case, c'est le nom du module le plus élevé... ce n'est pas nécessaire de comprendre le concept immédiatement. Il faut juste le mettre comme étant le même que le nom du projet (donc, *lab1*). Une fois terminé, pesez sur *Next*.

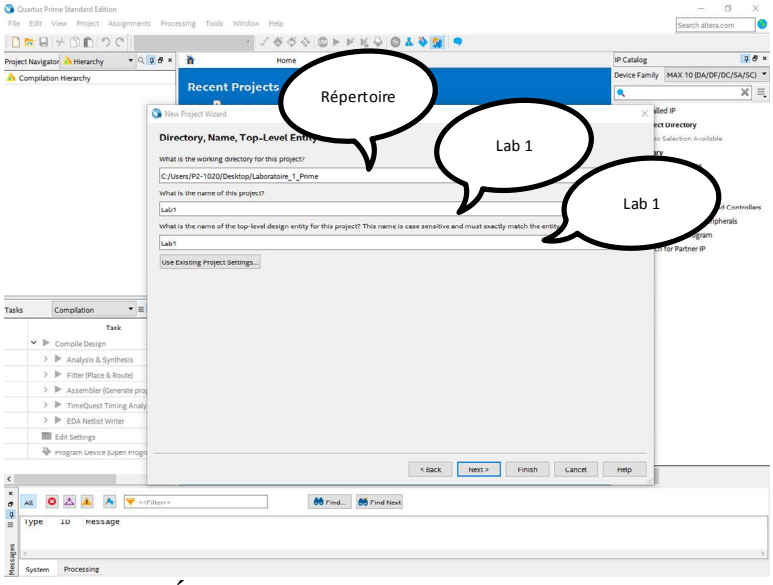

Fig. 5. Écran de spécification de nouveau projet

Dans l'autre fenêtre, on ne fait rien. On passe à la prochaine fenêtre en cliquant sur *Next*. Dans cette autre fenêtre nous cliquons encore sur *Next*. Dans cette fenêtre, nous allons spécifier l'information sur le FPGA que nous utiliserons. C'est de l'information qui se trouve dans la documentation sur le produit mais nous allons le recopier ici pour faciliter le travail (Fig. 3).

Dans *Device* nous allons au bas de la fenêtre et dans *Available devices*, on choisit

10M50DAF484C7G

Cliquez sur *Finish*...

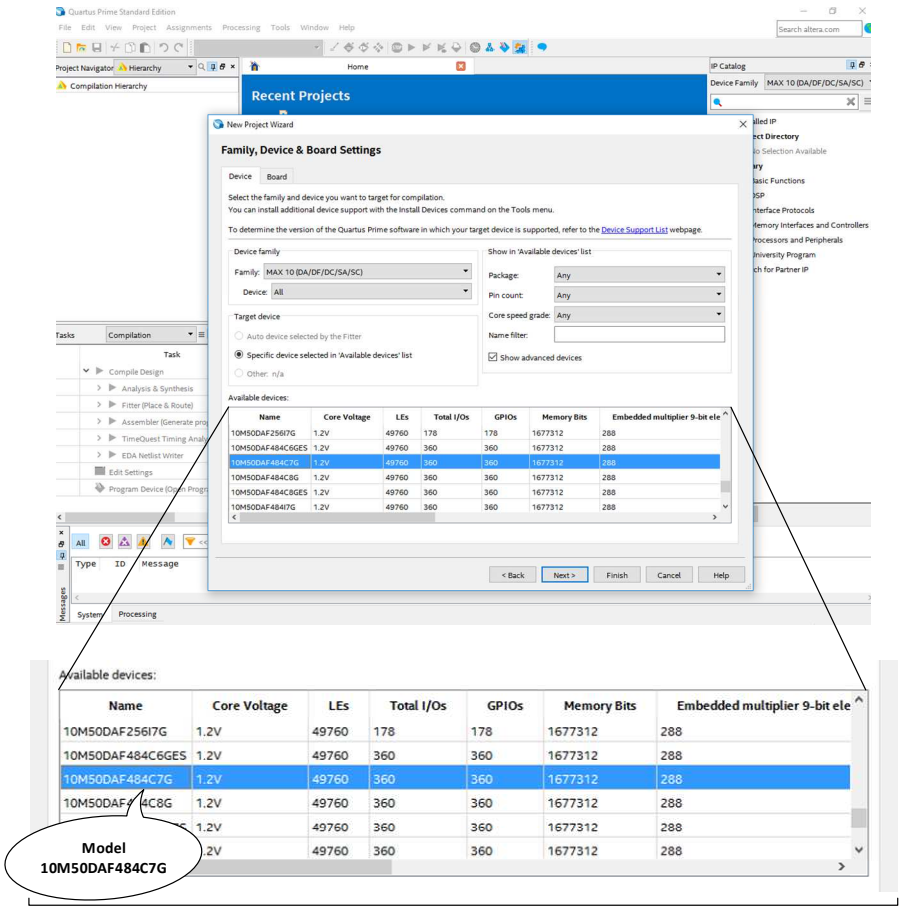

Fig. 6. Spécification du FPGA

Conception du circuit

Vous devriez maintenant avoir une fenêtre avec *lab1* écrit en haut gauche. La prochaine étape est de se créer un circuit en utilisant des dessins de porte logique.

Entrez dans *File→New→Block Diagram/Schematic File*
Cliquez sur *OK*
Une fenêtre semblable à celle de la Fig. 7 devrait apparaître.

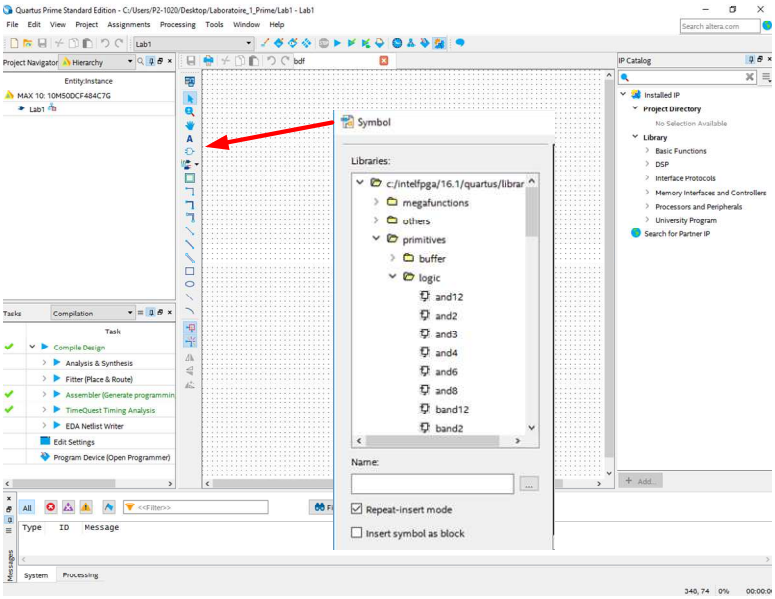

Fig. 7. Écran principal pour le dessin du circuit.

Pour créer votre circuit, il faut ajouter les portes logiques que vous voulez et les connecter ensemble. Allez voir la région à gauche de la région de dessin principale. Vous devriez voir l'icône d'une porte ET (Fig. 7). Cliquez dessus.

Une fenêtre devrait apparaître. En haut à gauche, on voit *Librairies*. Cliquez dessus (ou cliquez sur le >). D'autres options devraient se présenter à vous.

Entrez dans *Primitives→ Logic*

Et vous devriez voir plusieurs noms de portes logiques.

Sélectionnez AND2, qui est une porte ET à 2 entrées et cliquez sur OK. Vous avez spécifié au logiciel que vous vouliez ajouter une porte ET à 2 entrées. Il faut maintenant décider où on veut la placer dans notre schéma. Cliquez une fois sur la région de dessin, quelque part au centre, pour dessiner la porte ET.

On va maintenant spécifier que les entrées de la porte ET sont des ports d'entrée et que la sortie de la porte ET est un port de sortie. Ça veut dire qu'on va dire au logiciel que les signaux à l'entrée de la porte ET proviennent de l'extérieur du FPGA et que la sortie sera, elle aussi, envoyée à l'extérieur. L'ajout de ports d'entrée et de sortie se fait de la même manière qu'avec une porte logique. Cliquez sur la porte logique. Choisissez les primitives mais cette fois-ci, choisissez *PIN*.

Choisissez INPUT et ajoutez-le 2 fois avant de peser sur ESC pour arrêter d'en ajouter. Tracez maintenant des lignes qui vont des *PINs* vers les entrées de votre porte.

Faites les étapes semblables pour insérer un port de sortie (OUTPUT) et connectez la sortie de votre porte ET. Votre design devrait ressembler à celui de la Fig. 8.

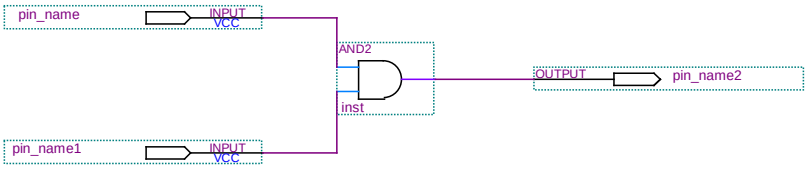

Fig. 8. Porte ET avec les entrées et sorties hors puce

Votre design est maintenant complété.

Une chose qu'on devrait peut-être faire c'est de donner des noms aux entrées et à la sortie. Ce n'est pas obligatoire, mais c'est parfois utile pour des circuits complexes.

Double-cliquez sur *PIN_NAME* qui se situe dans le port d'entrée et changez le nom pour *entree1*. Changez l'autre pour *entree2* et la sortie devrait aussi avoir le nom *sortie*. **Ne pas mettre d'accents**. On a maintenant fini de dessiner notre circuit. On le sauvegarde. Puisque c'est la première fois qu'on le sauvegarde, il va nous demander un nom. On lui donne le même nom que le projet soit : lab1.

Pour voir si tout est bon et pour transformer notre dessin en format compréhensible par le logiciel, on va faire une compilation.

Entrez dans *Processing →Start Compilation*. Il existe deux autres possibilités pour la compilation.

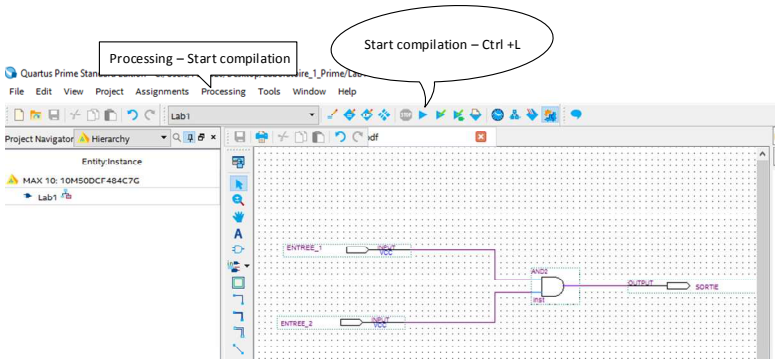

Fig.9. Compilation

En faisant la compilation, on est en train de faire beaucoup plus d'opérations qu'on ne le voudrait. Il passe par des étapes de placement et routage des portes logiques même si on ne voulait pas le faire. Le processus devrait prendre quelques minutes ne devrait théoriquement pas donner d'erreurs. S'il y en a, appelez le chargé de TD.

Simulation

Une fois que le design est terminé, on veut savoir s'il fonctionne avant même de le mettre sur le FPGA. Ceci s'appelle la simulation. Il y a un outil dans Quartus Prime Lite pour faire ça.

Puisque notre design est une porte logique, la meilleure façon de tester son fonctionnement c'est de mettre des 0 et des 1 partout et de voir si la sortie est bonne ou pas. Pour ce faire, on va se créer un fichier qui va contenir toutes la séquence de 0 et de 1 qu'on veut mettre à l'entrée pour le stimuler.

Entrez dans *File→New→University Program VWF* (*Vector Waveform File*).

Une fenêtre devrait apparaître. Dans la fenêtre au milieu (Fig.10), faites un clic du bouton de droite et sélectionnez *Insert →Insert Node or Bus*.

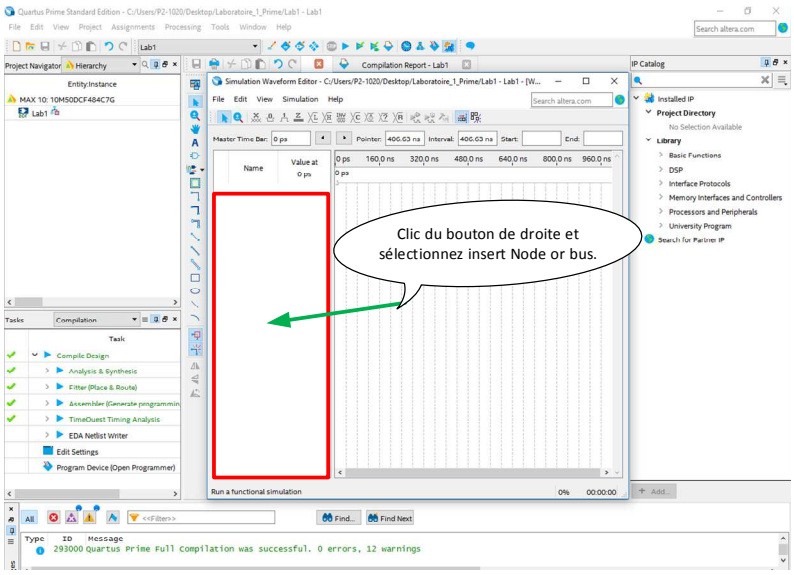

Fig. 10. Fenêtre pour créer un fichier pour la simulation

Une fenêtre devrait apparaitre qui demande une grande quantité d'information. À la place de remplir cette information manuellement, on va aller le remplir automatiquement.

Cliquez sur *Node Finder*. Avec cette option, il peut nous énumérer tous les ports qu'il détecte et ceci se fait en cliquant sur *List* en haut à droite (Fig. 11).

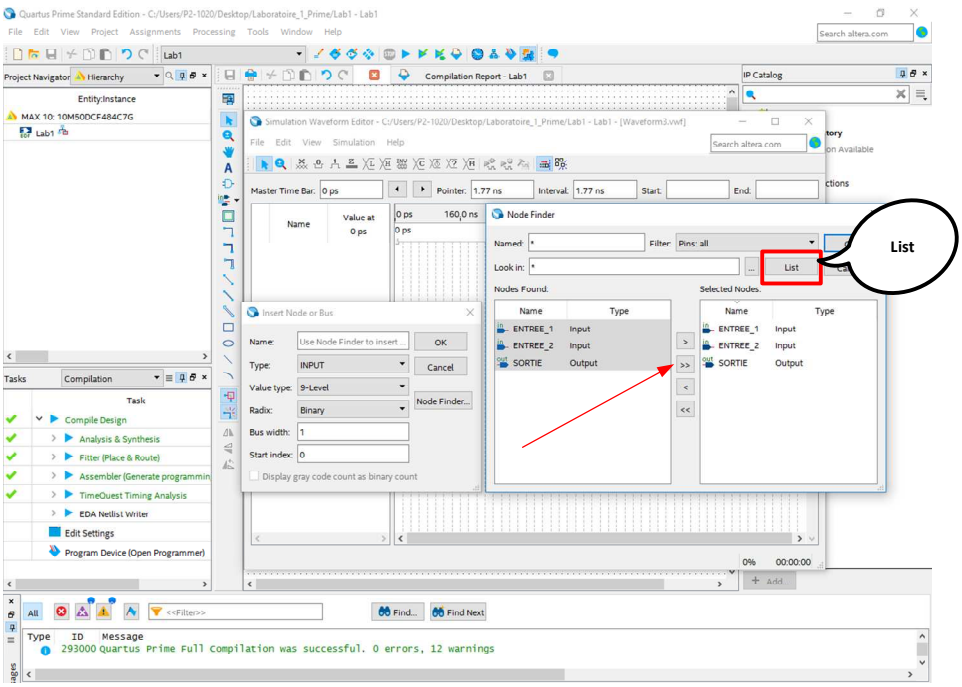

Fig.11. Fenêtre pour ajouter les signaux automatiquement

Les 2 entrées et la sortie devraient apparaitre à gauche. Cliquez sur la double flèche (Fig. 11) qui va de gauche à droite au milieu de la nouvelle fenêtre. Les 3 noms devraient maintenant apparaître à droite. Si c'est bien ça, pesez sur OK et pesez encore sur OK. Sinon, jouez avec les boutons jusqu'à ce que les noms des 3 pins se retrouvent à droite.

Vous devriez maintenant être de retour dans la fenêtre de la Fig. 10 La différence est que vous avez maintenant 3 signaux qui sont énumérés. On va maintenant définir toutes les entrées qu'on veut tester. On veut faire comme un tableau de vérité sauf que le logiciel ne fonctionne pas comme ça. On peut seulement définir un signal qui va changer de valeur avec le temps. La meilleure solution est de mettre une onde carrée de différentes fréquences dans chaque entrée et ça devrait générer toutes les variations possibles pour qu'on puisse l'observer à la sortie.

Cliquez sur *entree1* et la rangée devrait devenir bleue. Faites un clic de droite et sélectionnez *value* et ensuite *overwrite clock*. En bas, modifiez la période pour que ce soit 100ns.

Cliquez sur *entree2*. Faites un clic de droite et sélectionnez value et ensuite *overwrite clock* de nouveau. Modifiez sa période pour avoir 200ns.

On ne spécifie rien pour la sortie, puisque c'est une sortie. C'est le système qui va générer ça quand on va faire la simulation.

Il existe 4 possibilités d'entrée : 00, 01, 10 et 11. En regardant les formes d'ondes à l'entrée, vous voyez que toutes les possibilités sont essayées. Si vous ne voyez pas la forme d'onde en entier, trouvez une manière de faire un zoom (indice : il y a une icône pour une loupe). Sauvegardez le fichier sous le nom *lab1.vwf*.

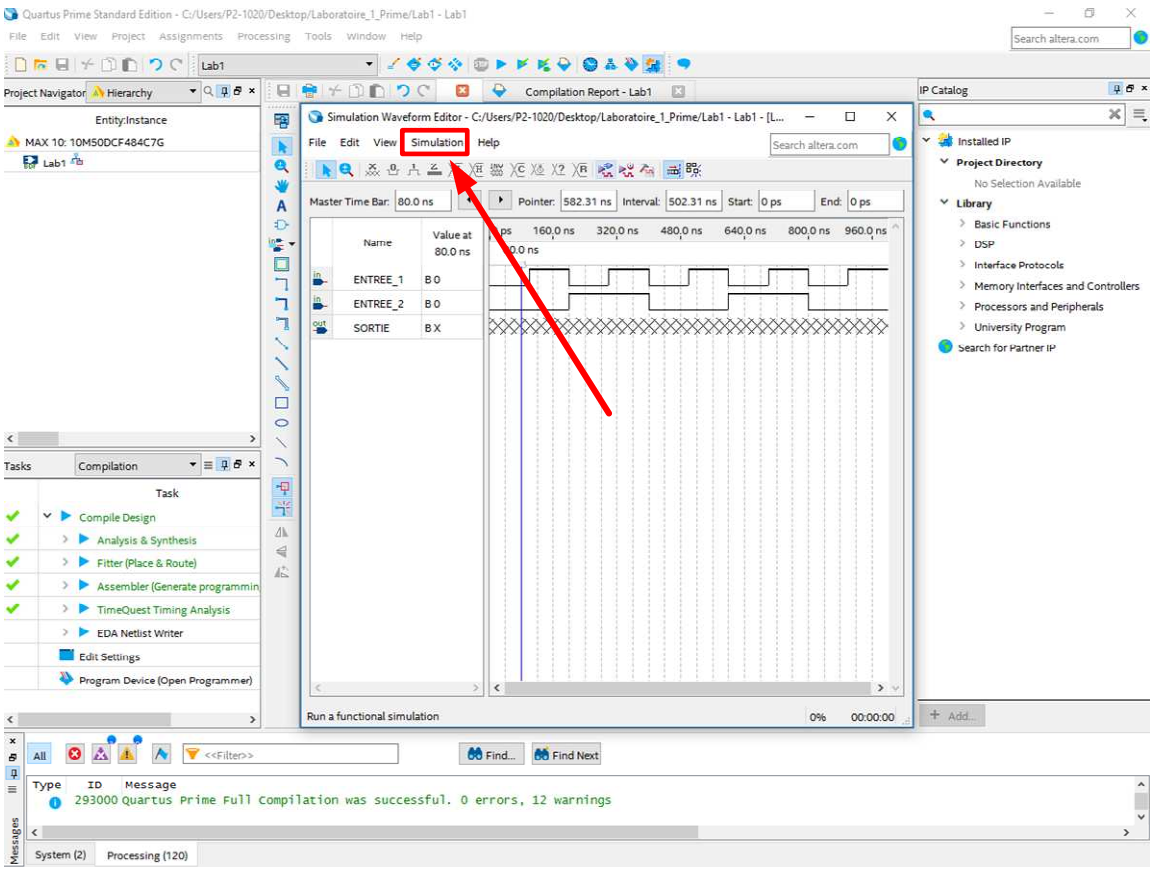

Fig. 12. Écran pour le fichier de simulation

On va maintenant rouler la simulation pour voir comment le système réagit.

Entrez dans *Simulation et choisir →Simulation settings* et choisir VHDL (The language used for testbench and netlist).

Cliquez sur *Simulation* et choisir *Run Fonctional Simulation*

Une fenêtre devrait apparaître montrant les signaux à l'entrée ainsi que le signal à la sortie. On voit que le système fonctionne bien : quand les entrées sont 1 et 1, la sortie est 1. Sinon, c'est 0 (Fig. 13)

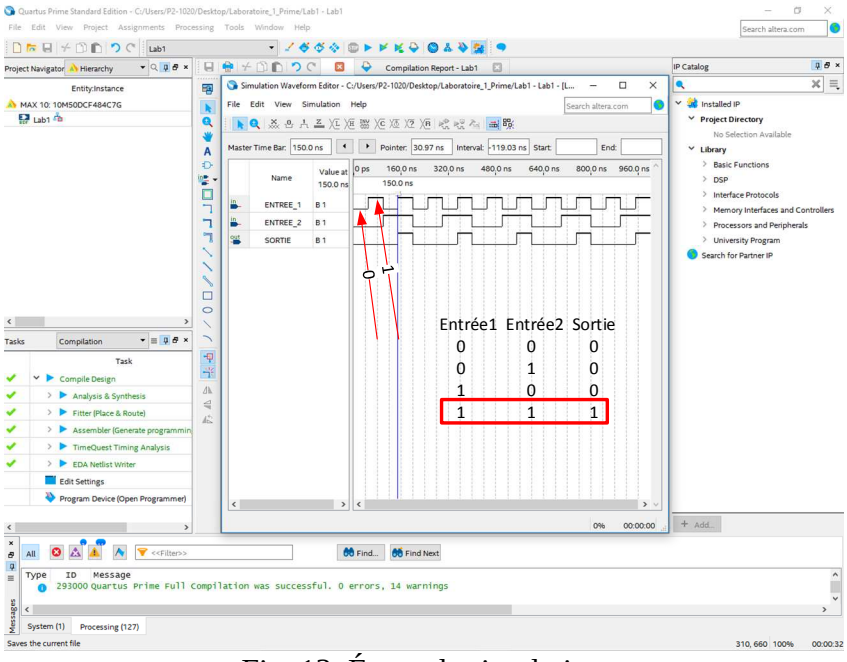

Fig. 13. Écran de simulation

Implémentation sur puce

Une fois que vous aurez confirmé que ça fonctionne, vous pouvez maintenant l'envoyer à la carte. La carte s'appelle le DE10 et offre beaucoup de possibilités. Une des choses qu'on va vouloir spécifier c'est OÙ les entrées et les sorties sont connectées. On a une porte logique à 2 entrées et on aimerait observer concrètement que le circuit fonctionne. Une possibilité c'est de connecter chaque entrée à un commutateur et de connecter la sortie à une lumière (LED, ou diode électroluminescente). Il faut donc spécifier à l'outil quelles sont les connexions qu'on veut faire. Ceci s'appelle *Pin Assignment*.

Entrez dans *Assignment→Pin planner*

Une fenêtre devrait apparaitre. En bas de cette fenêtre vous devriez voir les noms de vos entrées et de votre sortie (Fig. 14). L'endroit où vous voulez connecter vos pins est dans la section *Location*.

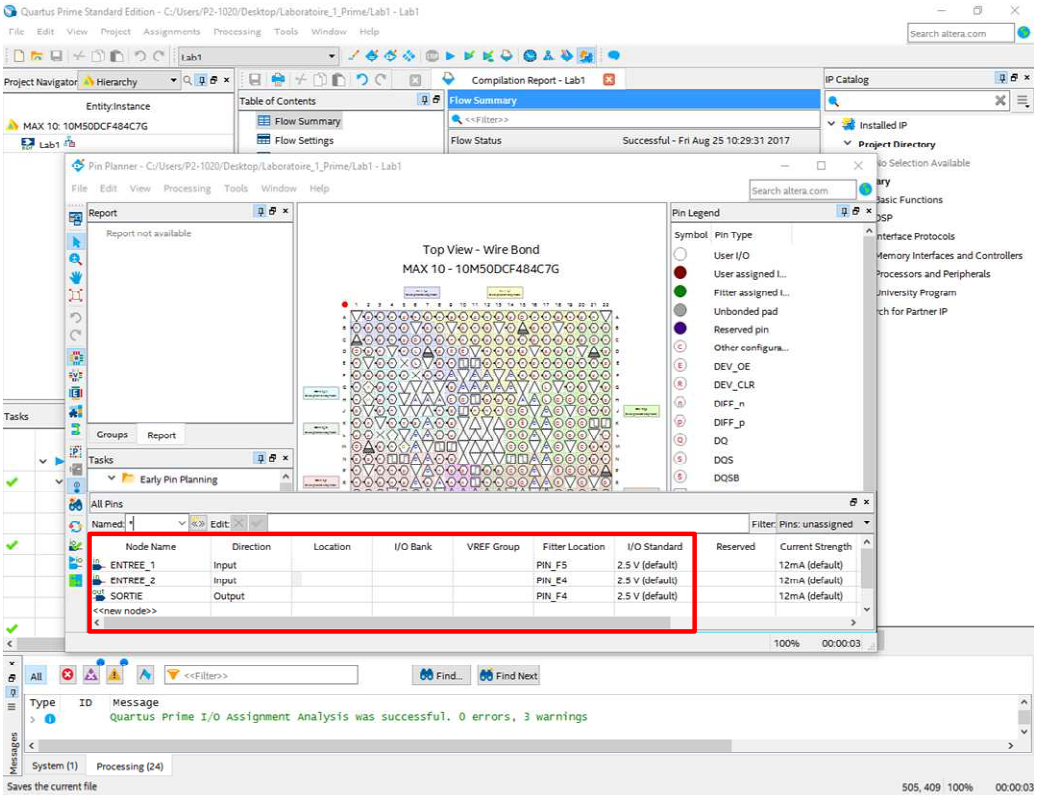

Fig. 14. Fenêtre utilisée pour l'assignation des pins

Dans cette section, vous pouvez spécifier le nom du pin ou vous voulez vous connecter. Pour connaitre les noms, il faut aller dans le manuel de l'usager de DE10 Prime Lite (https://www.google.ca/search?q=DE10+lite+manuel&oq=DE10+lite+manuel&aqs=chrome..69i57.10631j0j8&sourceid=chrome&ie=UTF-8). Par exemple, à la page 26 du manuel (p27 en .PDF), on retrouve les noms de pin des commutateurs (Fig. 15).

Signal Name	FPGA Pin No.	Description
SW0	PIN_C10	Slide Switch[0]
SW1	PIN_C11	Slide Switch[1]
SW2	PIN_D12	Slide Switch[2]
SW3	PIN_C12	Slide Switch[3]
SW4	PIN_A12	Slide Switch[4]
SW5	PIN_B12	Slide Switch[5]
SW6	PIN_A13	Slide Switch[6]
SW7	PIN_A14	Slide Switch[7]
SW8	PIN_B14	Slide Switch[8]
SW9	PIN_F15	Slide Switch[9]

Fig. 15. Extrait du manuel de l'utilisateur qui montre les PINs pour les commutateurs

On voit donc que le commutateur SW [0], le nom du pin à entrer c'est PIN_C10. Assignons les commutateurs SW [0] et SW [1] pour les 2 entrées de notre porte logique. Pour la sortie, assignons-lui une LED. Nous allons donc chercher dans le manuel de l'usager et nous voyons qu'à la page suivante, les LEDs sont énumérés. La première LED

rouge LEDR [0] se trouve à la position PIN_A8. Une fois complété, nous pouvons compiler et ensuite programmer le FPGA.

Pour programmer, entrez dans *Tools→Programmer*. Vous devriez voir apparaître une fenêtre comme à la Fig. 16. Cliquez sur *Hardware Setup* en haut à gauche et sélectionnez l'option qui contient le terme *USB*. Pour que l'option soit présente, votre plaquette doit être connectée et allumée.

Cliquez sur *ADD file…* puis double-cliquez sur le fichier *Lab1.sof*

Vous pouvez ensuite peser sur *Start*.

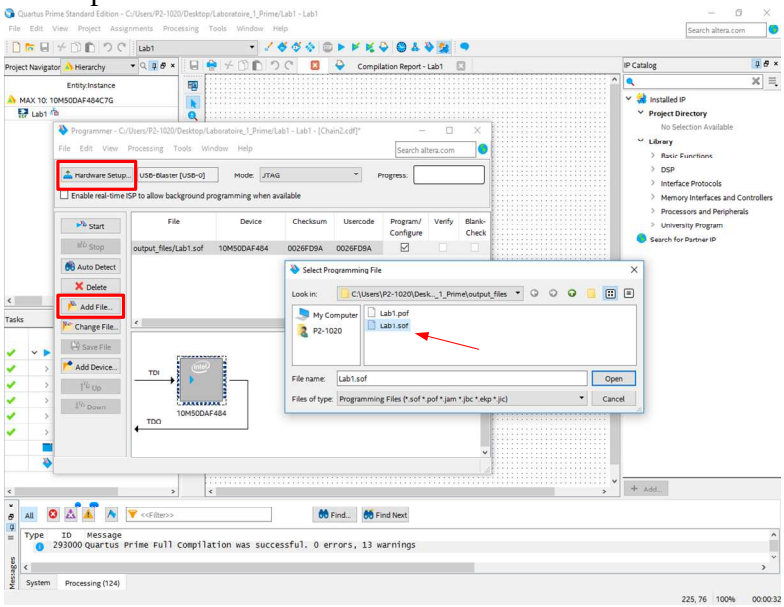

Fig. 16. Fenêtre pour la programmation du FPGA

Manipulez les commutateurs SW0 et SW1 et observez la lumière LEDR0 s'allumer et s'éteindre. Assurez-vous que ça se comporte comme une porte logique ET.

<u>Section à présenter</u>

On aimerait concevoir un système qui détecte les séquences de 3 bits qui ont une quantité impaire de '1'. C'est-à-dire qu'on veut que la sortie donne '1' quand les entrées sont 001, 010, 100 et 111. Dans les autres cas, le système devrait donner '0'.

1. Dessinez son tableau de vérité.
2. Écrivez l'équation somme de produits (somme des *minterms*) de la fonction.
3. Dessinez le circuit dans Quartus 10 Prime Lite et faites une **simulation** qui essaie toutes les possibilités d'entrées.
4. Implémentez le circuit sur FPGA et montrez le bon fonctionnement au chargé de laboratoire.