# INF1500 – logique des systèmes numériques

## PARTIE I

Simulation avec Vivado

---

# Introduction à l'utilisation de Vivado et à la technologie FPGA

---

*Révisions :*
Hamza - V0.0 (Première version)
Jeferson - V1.0 (Ajout du guide de simulation)
Pierre - V1.1 (Précisions au texte et orthographe)
Imad - V1.2 (Ajout de la conception à base de schéma)
Alexy - V2.0 (Modification pour le cours INF1500)

Septembre 2019

Département de génie informatique et de génie logiciel
École Polytechnique de Montréal

# 1 La technologie FPGA

Un FPGA (pour Field-Programmable Gate Array) est un composant électronique programmable qui peut être reconfiguré (reprogrammé) par l'utilisateur après la fabrication. Le FPGA met à la disposition de l'usager des ressources matérielles où la collectivité des différents éléments est configurable.

Pour les besoins de nos laboratoires en INF1500, le FPGA est considéré comme un "panier" de portes logiques (ET, OU, Non-ET, Non-OU, XOU, etc.) et de modules numériques (compteurs, bascules, etc.) dont l'outil Vivado nous permet de les interconnecter pour former des systèmes numériques plus complexes. En d'autres termes, pour chacun des laboratoires, le circuit/design sera implémenté et testé sur la carte de développement de FPGA.

# 2 La carte de développement de FPGA

La carte FPGA utilisée pour les laboratoires en INF1500 sera celle montrée dans la figure suivante :

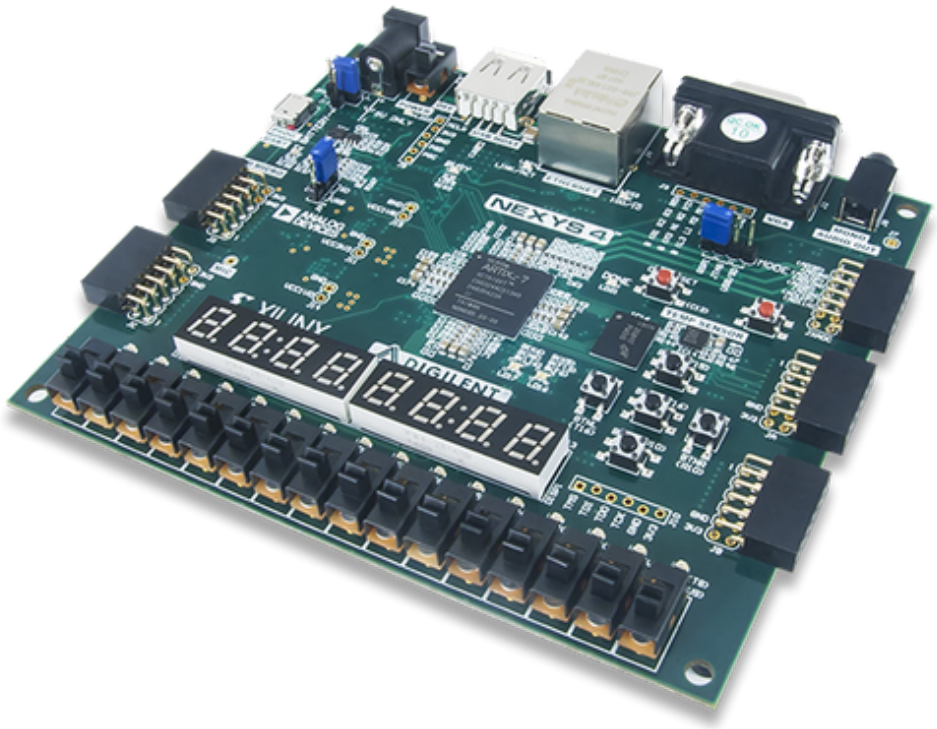


Figure 1 – Carte FPGA Nexys 4

# 3 Utilisation de Xilinx Vivado

Le logiciel Xilinx Vivado est un outil de conception de circuits pour FPGA de Xilinx. Ce logiciel permet essentiellement d'effectuer les différentes étapes de la conception et de la réalisation de circuits numériques sur FPGA. Il permet entre autres de faire la description, la simulation, la synthèse et l'implémentation d'un circuit, puis la programmation d'une puce d'une des différentes familles de FPGA de Xilinx.

## 3.1 Création d'un projet

Lancez Xilinx Vivado en choisissant la commande correspondante dans le menu Démarrer puis Xilinx Design Tools puis Vivado 2018.x (x représentant un chiffre qui peut varier selon la version installée sur votre ordinateur). Un projet regroupe plusieurs fichiers sources pour un laboratoire ou un module en particulier. Après avoir lancé le programme, cliquez sur le bouton "Create Project >" qui se situe dans la section "Quick Start" (ou faites File > Project > New...).


FIGURE 2 – Menu de démarrage Vivado

Cliquez sur **Next** pour passer à la fenêtre suivante. Choisissez un nom représentatif pour votre projet. Il faut aussi spécifier un répertoire où le projet sera sauvegardé. **Il est important que le chemin de ce répertoire ne contienne pas d'espaces**, parce que certains outils invoqués peuvent ne pas les accepter. Dans tous les cas, essayer de toujours respecter cette pratique.

> NOTE : Dans les laboratoires du GIGL, il est recommandé de travailler dans un répertoire personnel sur C :\TEMP\VOTRENOM. Une fois le travail terminé, il est important d'archiver ce répertoire et d'en emporter une copie avec vous. Le répertoire C :\TEMP\ des ordinateurs des laboratoires est régulièrement effacé.

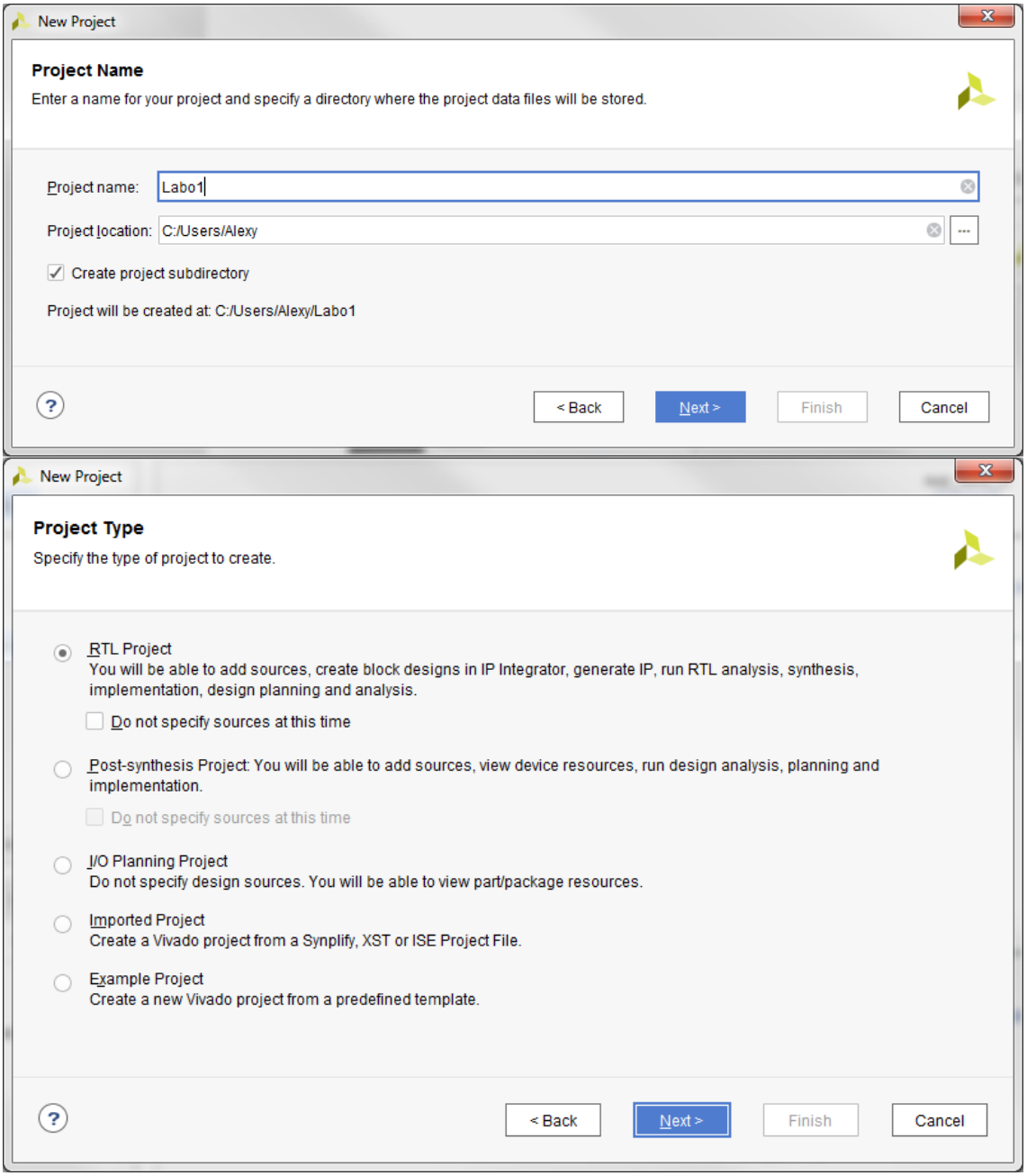


FIGURE 3 – Création du projet

Une fois le nom et le répertoire du projet choisis, cliquez sur **Next** et choisissez un **projet RTL (RTL Project)** dans la fenêtre suivante et décochez **Do not specify sources at this time**. Une nouvelle fenêtre s'ouvre, assurez vous (et changez au besoin) que **Target language** soit séléctionné à **VHDL** et **Simulator language** soit séléctionné à **Mixed**. Cliquez sur **Next** puis **Next**.

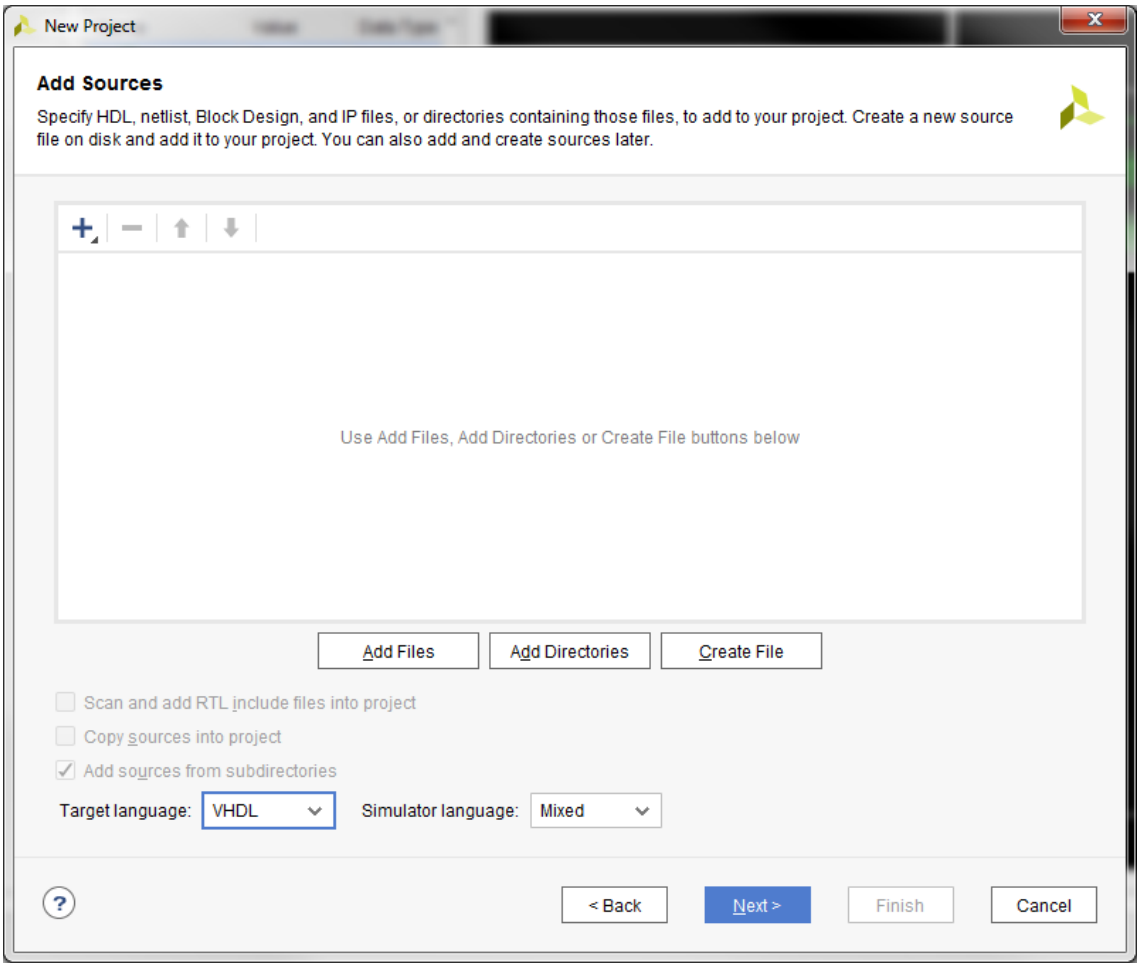


FIGURE 4 – Création du projet

Vous devez maintenant remplir certains champs correspondants au composant FPGA utilisé. Cette puce est celle qui est montée sur la carte FPGA disponible au laboratoire. Reproduisez les choix de la figure suivante (sélectionnez la Part xc7a100tcsg324-1), cliquez sur **Next**, puis sur **Finish**.

Category : General Purpose
Family : Artix-7
Package : CSG324
Speed : -1
Temperature : All Temaining

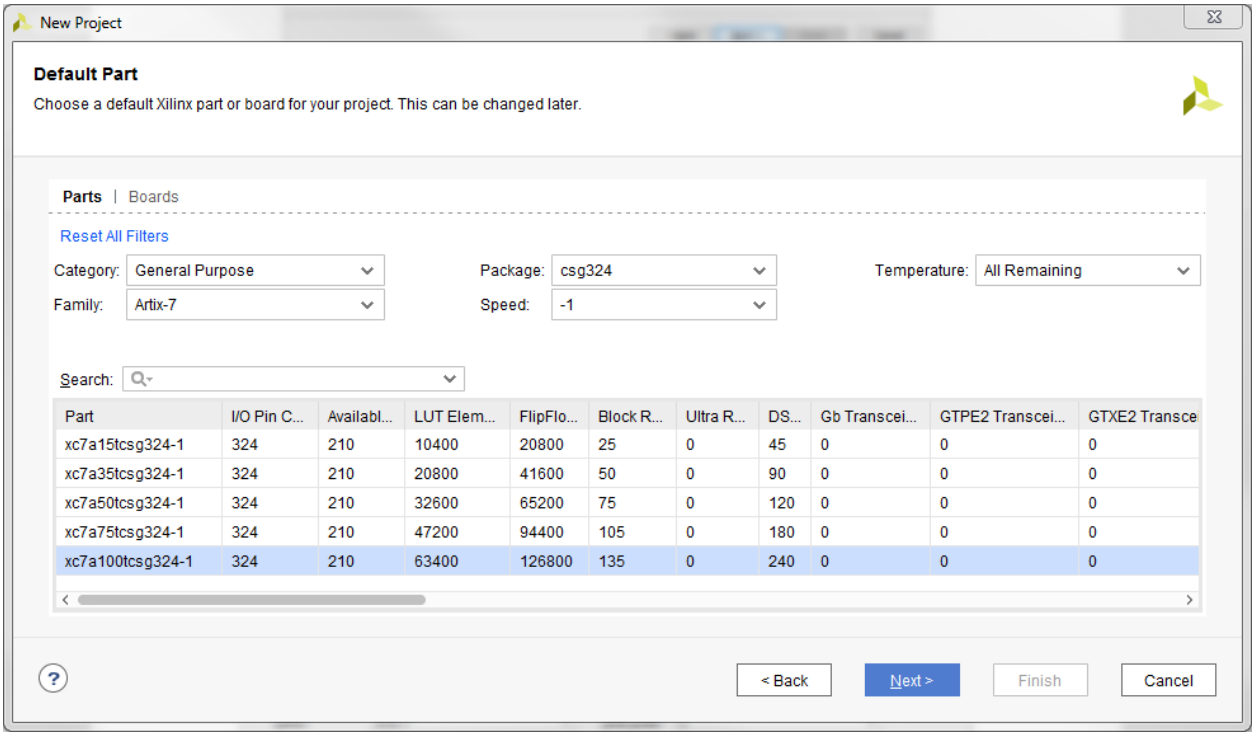


FIGURE 5 – Sélection de la board

## 3.2 Ajout de la librairie de blocs

Pour utiliser les modules de base de porte logique préconçue, il faut ajouter la librairie **XUP_LIB** au répertoire d'IP de Xilinx. Vous devez décompresser le fichier XUP_LIB.zip (présent dans les ressources supplémentaires sur Moodle) et ajouter le chemin au projet. Pour le faire il faut ouvrir l'onglet **Setting** dans **Project Manager** (dans la barre de gauche), ouvrir l'onglet **IP** et cliquer sur **Repository**. Cliquez sur le + puis rajoutez le chemin dans lequel vous avez décompressé l'archive. Une petite boite de dialogue peut apparaître, cliquez sur **Ok**. Une fois fait, cliquez sur **Ok** pour fermer la fenêtre.

Vous devrez répéter l'opération avec la seconde librairie **ATAD_LIB** (présente dans les ressources supplémentaires sur Moodle).

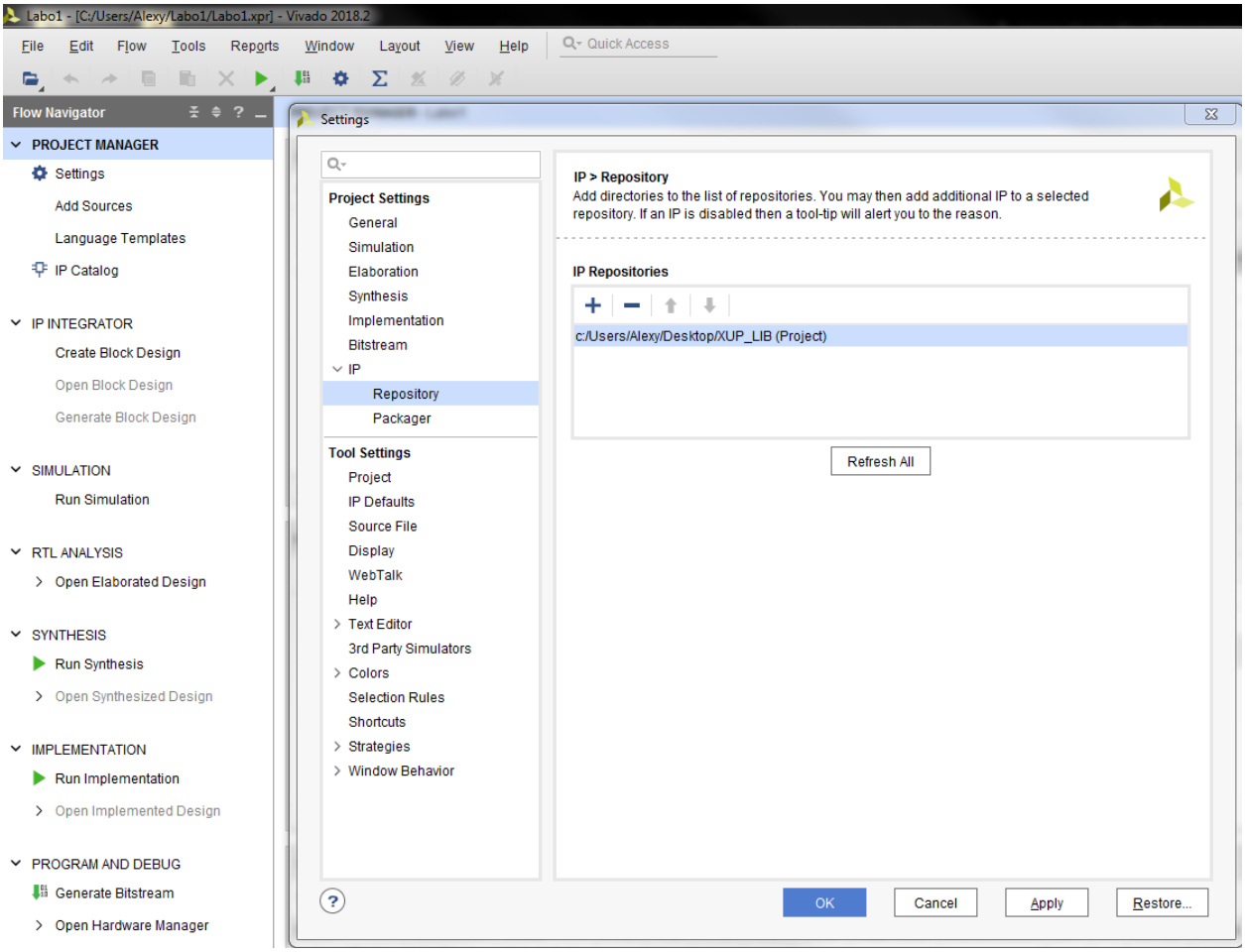


FIGURE 6 – Importation de la librairie

## 3.3 Création d'un bloc

Le principe de design d'un système numérique est toujours le même : nous créons d'abord les plus petits blocs, puis nous pouvons inclure ces blocs dans de plus gros et ainsi de suite. Un bloc, lorsqu'utilisé, ne montre que ses ports et non son fonctionnement interne. Pour cette introduction, vous allez créer un bloc d'additionneur/soustracteur complet de 1 bit. Pour créer un bloc cliquez sur **Create Block Design** dans l'onglet **IP INTEGRATOR** de la barre de gauche. Ici, vous allez décrire ce nouveau bloc de façon schématique. Donnez un nom pertinent à celui-ci, comme **one_bit_full_adder** et cliquer sur**OK**.

Note : ne commencez pas le nom par un chiffre et limitez-vous à des caractères alpha-numériques. Assurez-vous qu'aucun espace existe dans le nom du bloc.

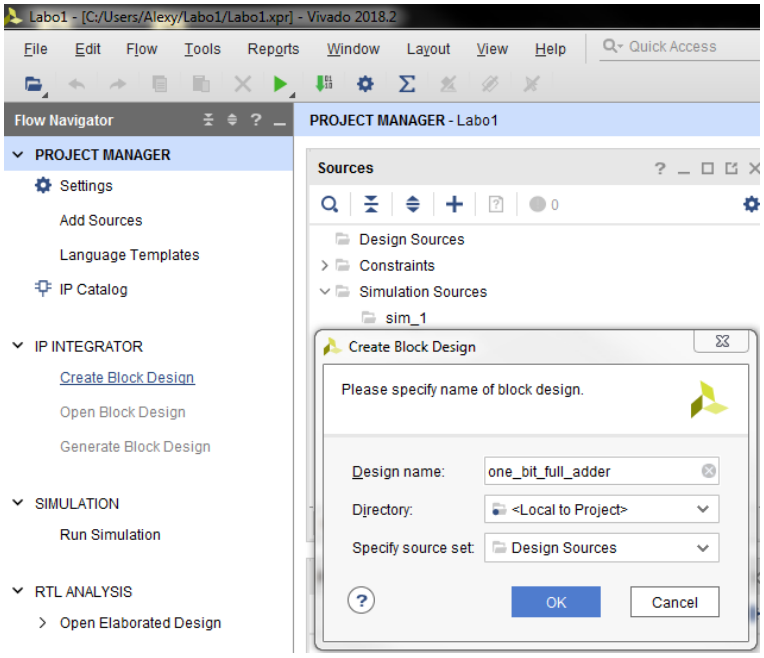


FIGURE 7 – Création d'un bloc

Pour sélectionner une porte logique de la librairie récemment ajoutée, cliquez sur + **(Add IP)** ou faites Ctrl+I, et choisissez l'une des porte sous le nom XUP... comme montré dans la figure suivante :

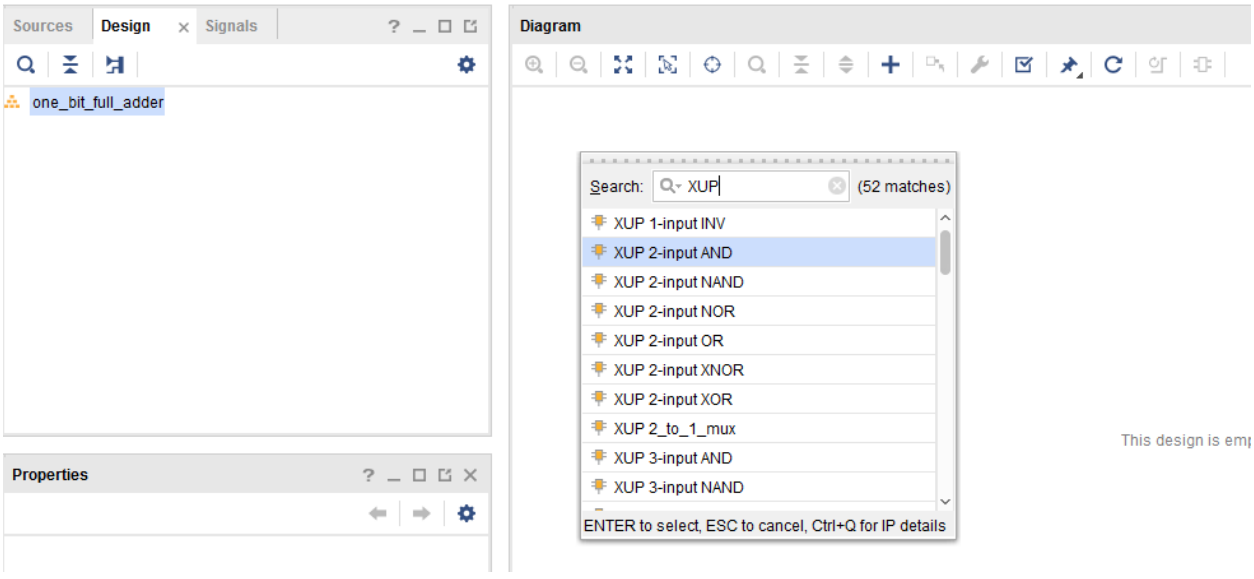


FIGURE 8 – Création d'un bloc

À votre tour de jouer, reproduisez le schéma suivant :

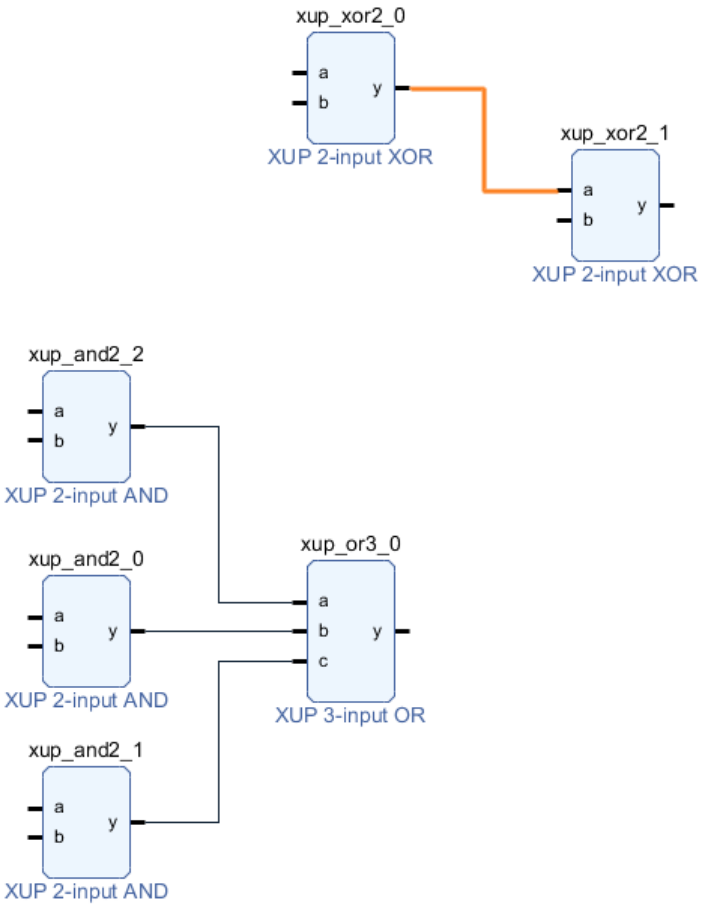


FIGURE 9 – Création de l'additionneur

Maintenant que les éléments de base sont placés, vous ajouterez des entrées/sorties afin de créer les ports de votre bloc. Pour créer un port d'entrée/sortie, faites un clique droit sur une zone blanche dans le fichier de dessin puis cliquez sur **Create Port...**. Créez trois port d'entrée (Input) et deux ports de sortie (Output) comme montré sur la figure suivante. Une fois fait, reliez les entrées/sorties aux IP puis appuyez sur F6 pour valider votre design.

NOTE : Les noms sur les entrées/sortie des portes n'ont pas à correspondre avec les noms des entrées/sorties du circuit.

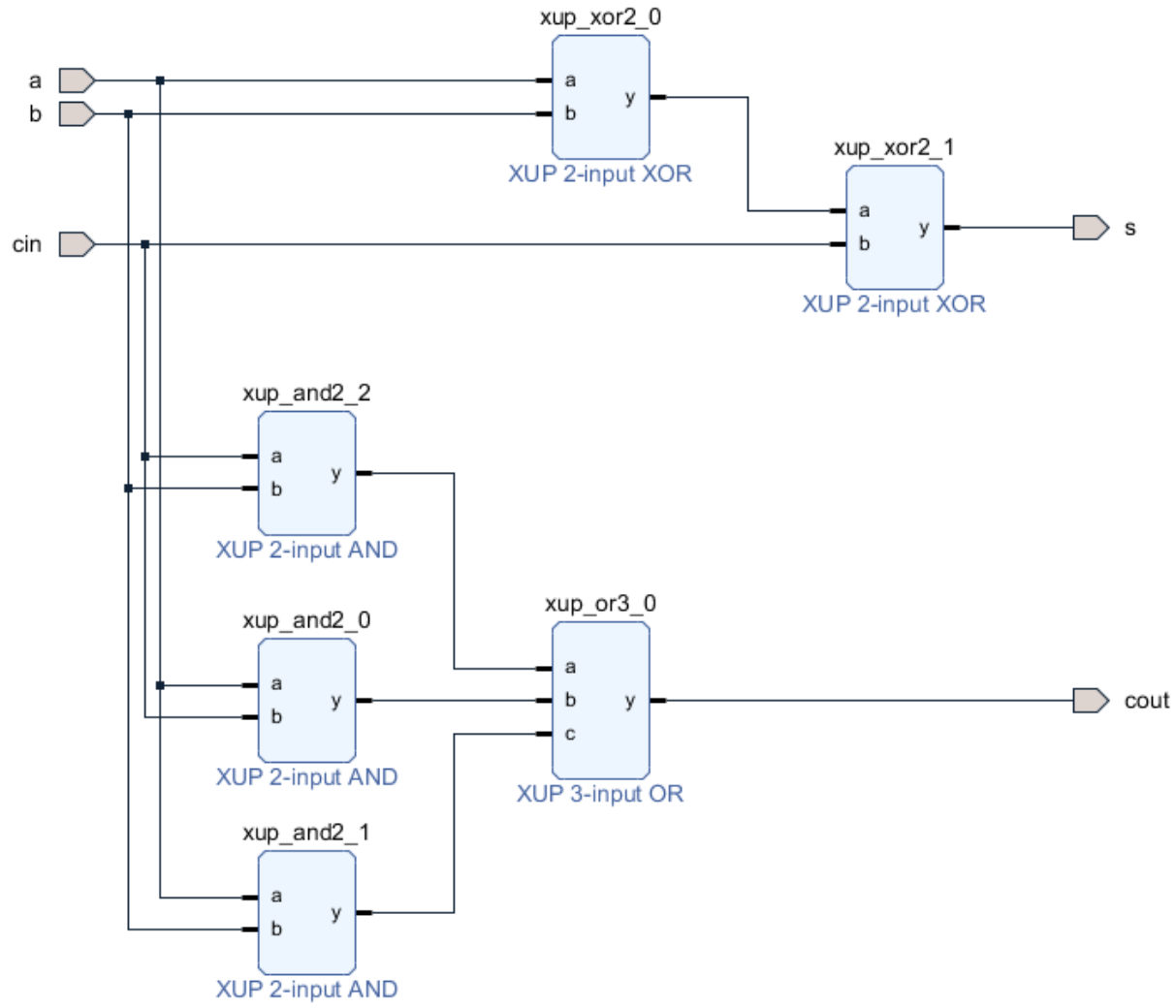


FIGURE 10 – Création de l'additionneur

## 3.4 Simulation d'un bloc

On veut maintenant simuler le design afin de s'assurer de son bon fonctionnement. Pour simuler le design, il faut générer un HDL wrapper. Sélectionnez le fichier de dessin dans l'onglet **Sources** et faites un clique droit puis cliquez sur **Create HDL Wrapper...**. Une fenêtre apparaît, laissez les paramètres par défaut et cliquez sur **OK**.

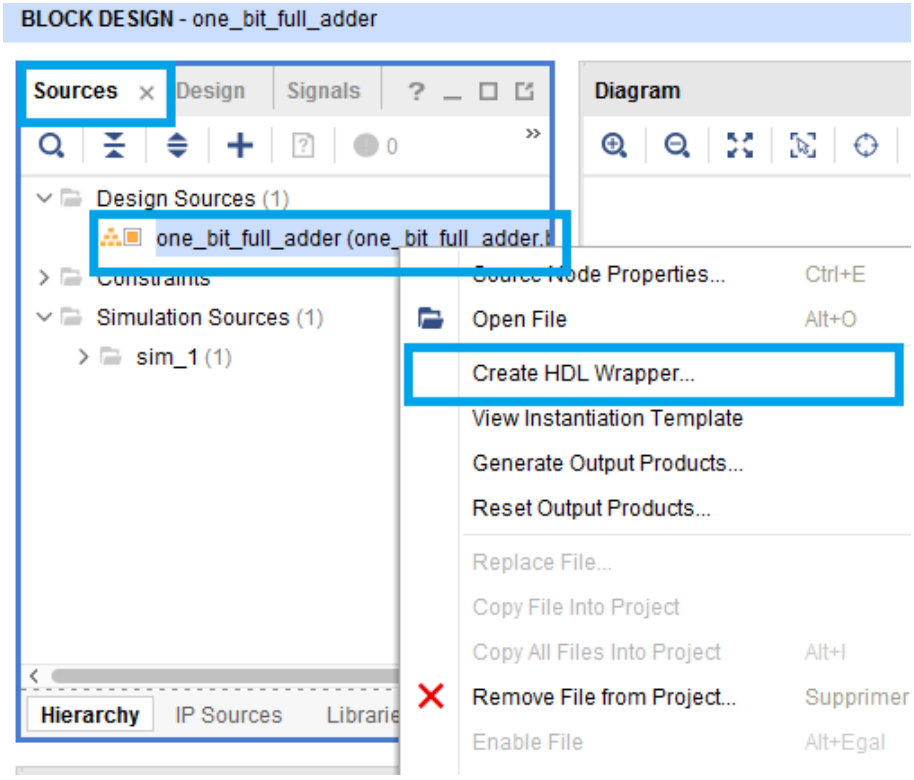


FIGURE 11 – Simulation

Pour lancer la simualtion cliquez sur **Run Simulation** dans **SIMULATION** dans la barre de gauche puis **Run Behavioral Simulation**. Le simulateur Vivado permet aussi de forcer manuellement la valeur des signaux, c'est à dire sans passer par un banc d'essai. Cette fonctionnalité peut être utile pour effectuer un test rapide pour vérifier le comportement de votre circuit. Dans ce cours (INF1500) nous utiliserons toujours cette méthode. Si vous désirez forcer une entrée a une certaine valeur ou une certaine horloge, faites un clique droit dessus (dans la boite **Objects** puis cliquez sur **Force Constant** ou **Force Clock**.

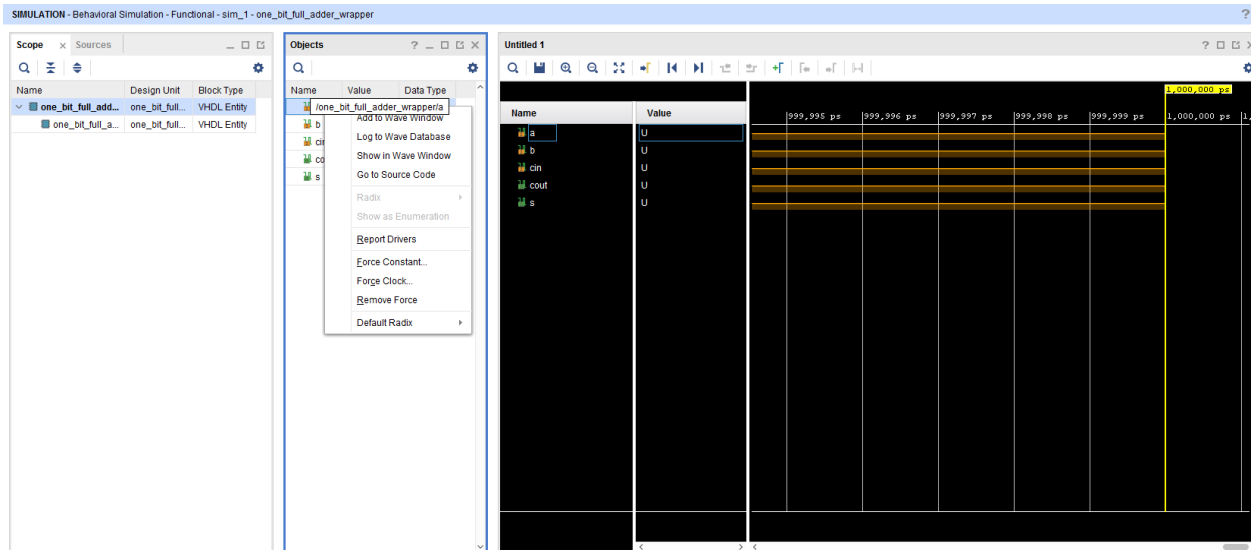


FIGURE 12 – Simulation

A vous de jouer, simulez le circuit en assignat à l'entrée a une horloge de 10MHz (100ns), b une horloge de 20MHz (50ns) et cin une horloge de 40MHz (25ns). Mettez comme paramètre 0 pour **Leading edge value** et 1 pour **Trailing edge value**. **Duty cycle** devrait être réglé à 50.

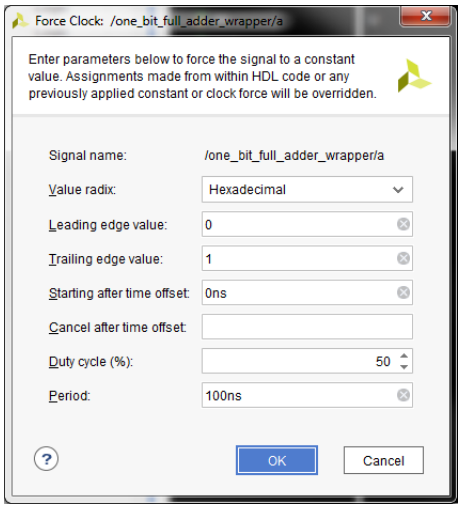


FIGURE 13 – Horloge de 10MHz

Pour rouler la simulation, rentrez le temps de simulation (ici 100ns) dans la barre de simulation (en haut de la fenêtres) puis faite Maj+F2 ou cliquez sur le bouton a coté du temps de simulation pour lancer un pas de simulation. Vous devriez obtenir les mêmes résultats que sur la figure suivante (il faudra peut-être zommer sur la timeline de simulation). Vérifiez

que les résultats sont correctes par vous même en faisant la table de vérité du circuit et comparant les sorties cout et s.

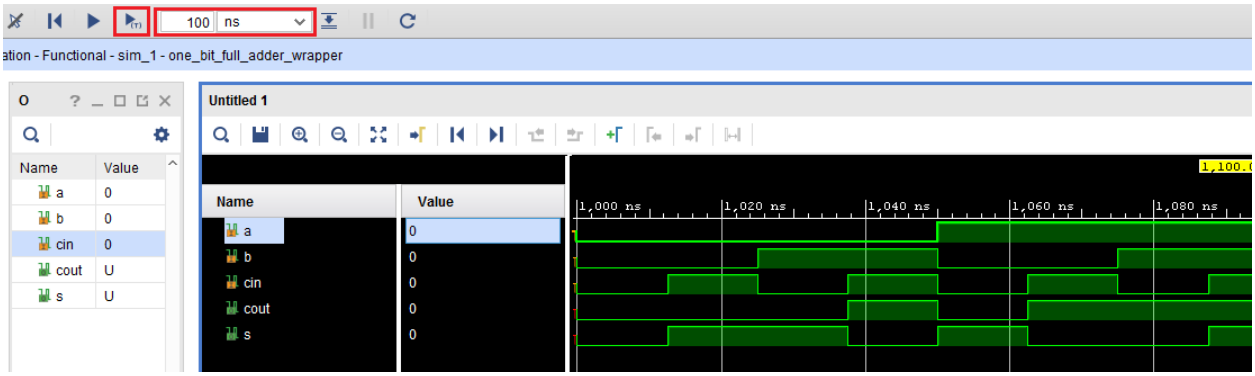


FIGURE 14 – Simulation

Si vous désirez modifier le diagramme, il faudra recréer le Wrapper HDL en effectuant la même manipulation que précédemment avant de re-simuler le circuit.

## 3.5 Réutilisation d'un bloc

Utilisons maintenant l'additionneur complet de 1 bit (le bloc du fichier one_bit_full_adder) que vous venez de créer dans un circuit de plus haut-niveau. Pour ce faire, créez un nouveau fichier de type "Block Diagram" avec le nom four_bit_full_adder. Vous allez effectivement vous servir de votre additionneur simple pour en faire un plus "large" (qui soustrait également). Pour cela suivre les prochaines étapes : dans l'onglet **Tools**, choisir **Create and Package New IP**, puis **Next**.

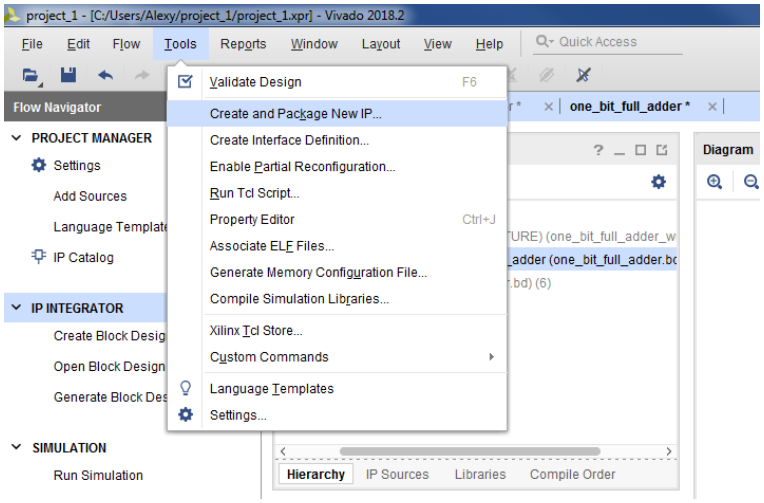


FIGURE 15 – Génération de l'IP

Sélectionnez ensuite **Package a block design from the current project**. Puis dans la liste séléctionnez **one_bit_full_adder** puis appuyez sur **Next**

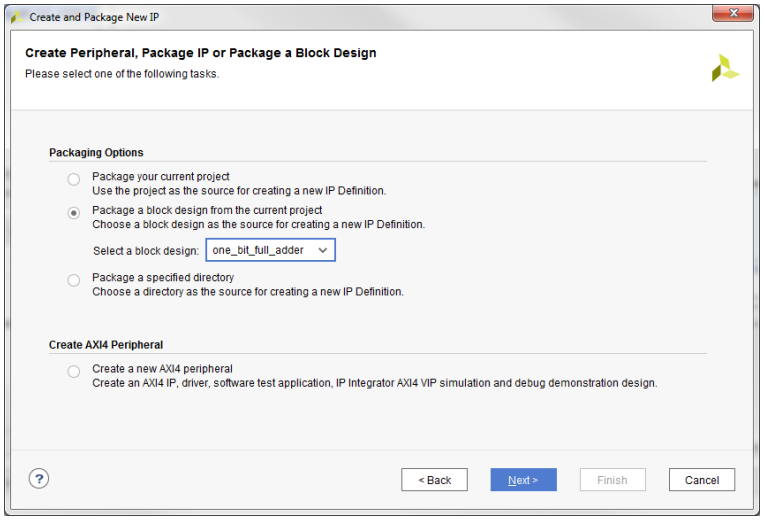


FIGURE 16 – Génération de l'IP

Choisir ensuite la seconde option qui inclut les fichiers .xci générés, modifiez le répertoire de destination de l'IP par votre chemin du projet en gardant comme dossier final ip_repo puis cliquez sur **Next** , **OK**, **OK** et **Finish**.

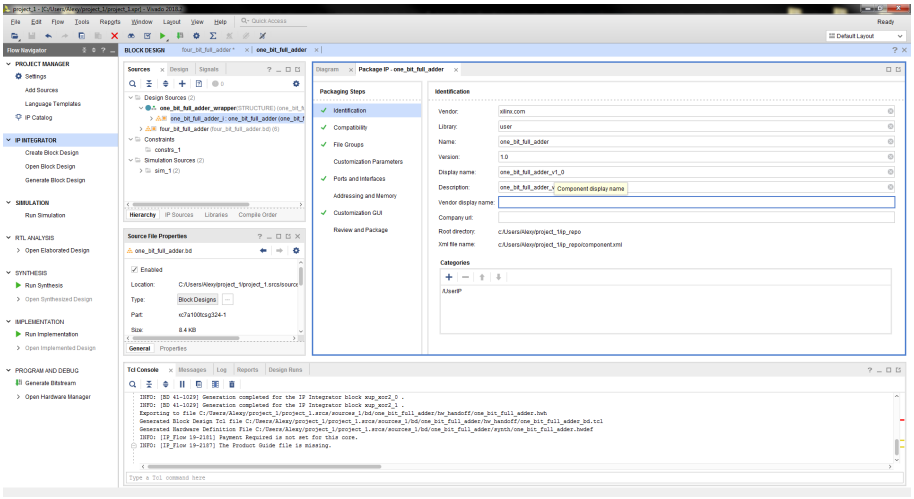

FIGURE 17 – Génération de l'IP

La fenêtre suivante apparaît, donnez un nom à votre IP, puis sélectionnez **Review and Package**. Une fois dans cet onglet, cliquez sur **Package IP**.

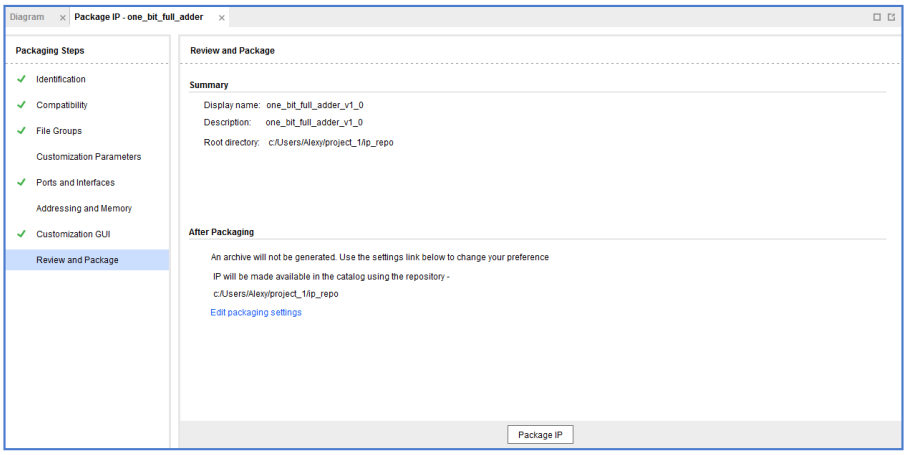


FIGURE 18 – Génération de l'IP

Veiller noter que vous devez créer un nouveau fichier de dessin pour l'utilisation de l'IP généré et non pas modifier le fichier de dessin qui a permis l'élaboration de l'IP.

Votre IP est maintenant packagé et nous pouvons nous en servir dans une nouveau diagramme. Retournez sur votre dessin du four_bit_full_adder. Comme pour une porte, ajoutez un IP et cherchez votre one_bit_full_adder.
Si vous voulez modifier votre IP il faudra le repacketer une fois le design changé.

## 3.6 Creation d'un additionneur soustracteur

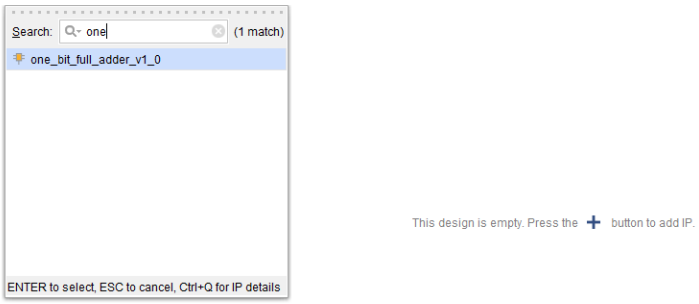


FIGURE 19 – Utilisation de l'IP

Attention, cette fois ci les entrées A B et la sortie S sont des BUS (un ensemble de fils, 3 à 0, se sont les 4 bits de nos entrées). Pour les créer, lorsque vous ajoutez une entrée ou sortie, cochez la case **Create vector :** et réglez la range au besoin.

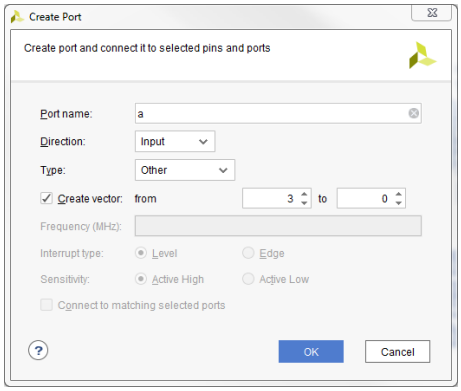


FIGURE 20 – Création d'un BUS

Ajouter quatre instance de votre one bit full adder et complétez le diagramme avec cinq XOR2, un NOR4 et les entrées/sorties comme sur la figure suivante.

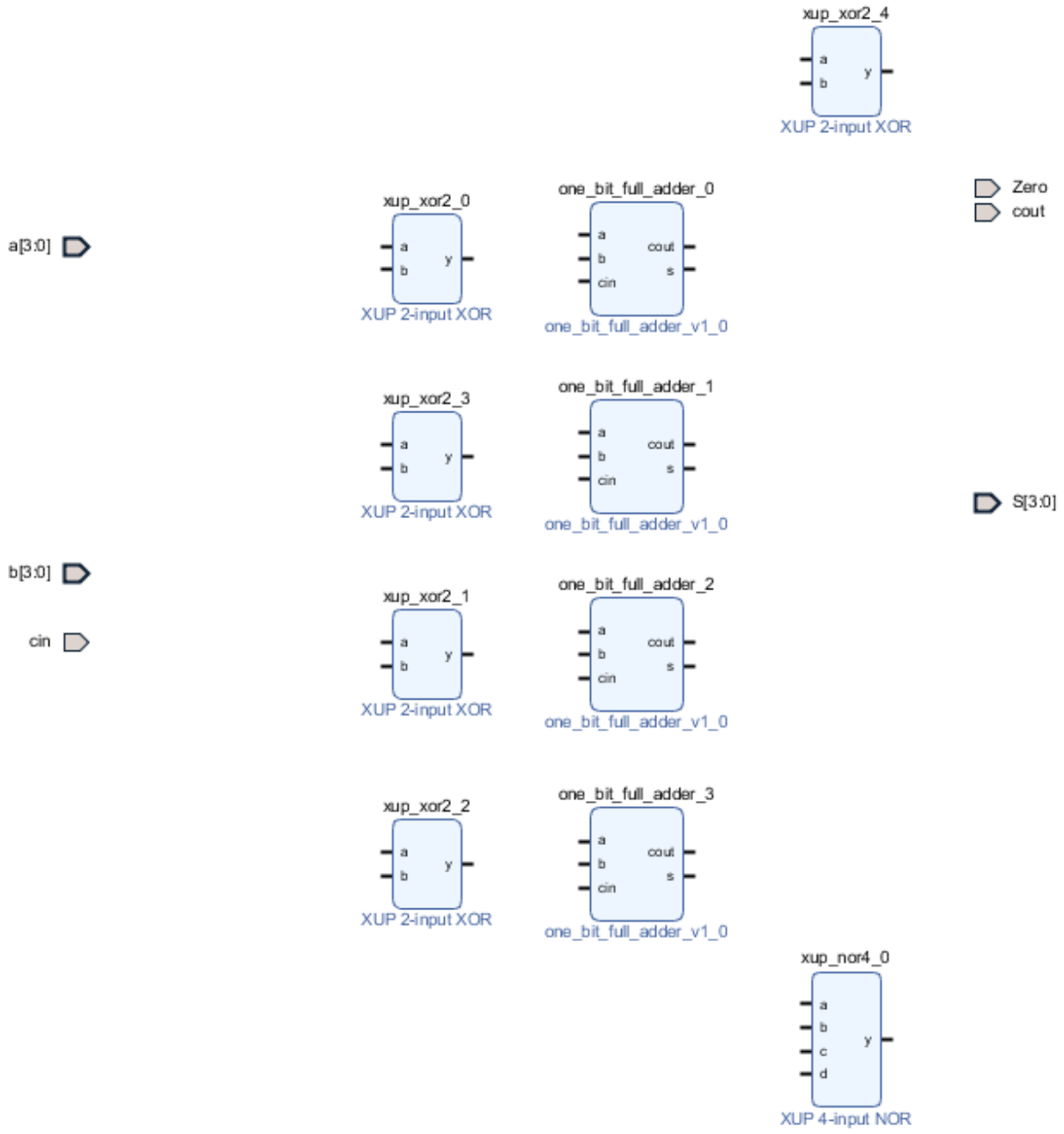


FIGURE 21 – Utilisation de l'IP

Pour relier les BUS à vos entrées de one_bit_full_ adder, il faudra les séparer. Pour cela utilisez l'IP **ATAD_SPLIT_4**. Le 4 designe la taille des données en entrées. Ici notre BUS fait 4 bits. En entrée de l'IP vous mettrez le BUS et vous devrez relier les sortie comme dans la figure ci-dessous.

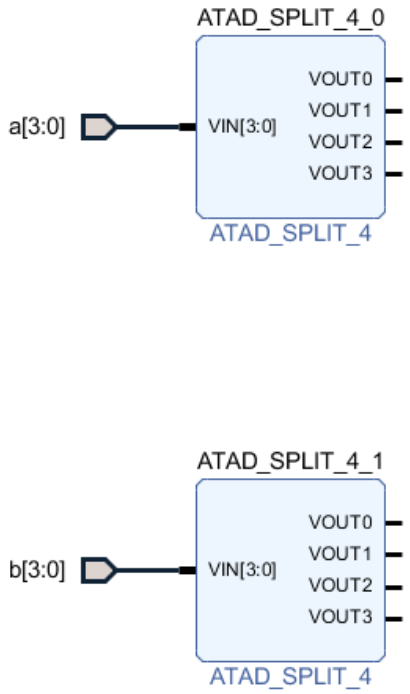


FIGURE 22 – Split du BUS

Pour relier les sorties de vos one bit full adder au BUS s il faudra les fusionner. Pour transformer un ensemble de fils en un BUS, la méthode est la suivante. Ajoutez un IP **Concat**. Double cliquez dessus pour modifier ses paramètres. Dans **Number of Ports** il faut mettre le nombre de fils a fusionner pour créer le BUS. Dans notre cas, nous voulons fusionner les bits de sortie s de nos additionneur donc 4. Le composant se met a jour automatiquement, cliquez sur **OK**. Vous pouvez maintenant relier vos fils.

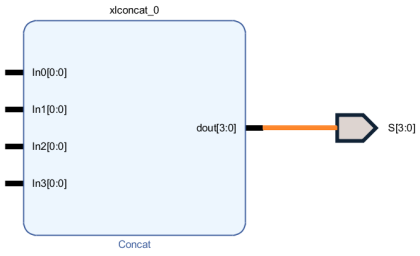


FIGURE 23 – Merge du BUS

Vous pouvez maintenant relier tous les fils entre eux pour reproduire le schema ci-dessous (vous devrez zoomer sur l'image pour mieux voir).

— Relier les bits de a aux entrées a de vos one bit full adder
— Relier les bits de b aux entrée des xor2 en face des one bit full adder
— Relier cin a chaque entrée des xor
— Relier les sortie des xor en face des one bit full adder aux entrés b de ceux-ci
— Relier cin a l'entrée cin du premier one bit full adder (le plus au nord).
— Chaîner les one bit full adder en reliant la sortie cout a l'entrée cin du prochain one bit full adder. Pour le dernier (le plus au sud) relier le cout a la seconde entré du xor seul auquel cin devrai deja etre relié.
— Relier les sorties s des one bit full adder à l'IP **Concat** (faite attention, le one bit full adder le plus au nord doit être connecté a l'entrée 0 du **Concat**, le suivent l'entrée 1, etc...)
— Relier les meme sorties s a la porte NOR4.
— Enfin relier la sortie Zero a la sortie de la porte NOR4, relier cout a la sortie de la porte XOR2 ayant cin et le cout du dernier one bit full adder en entrée.

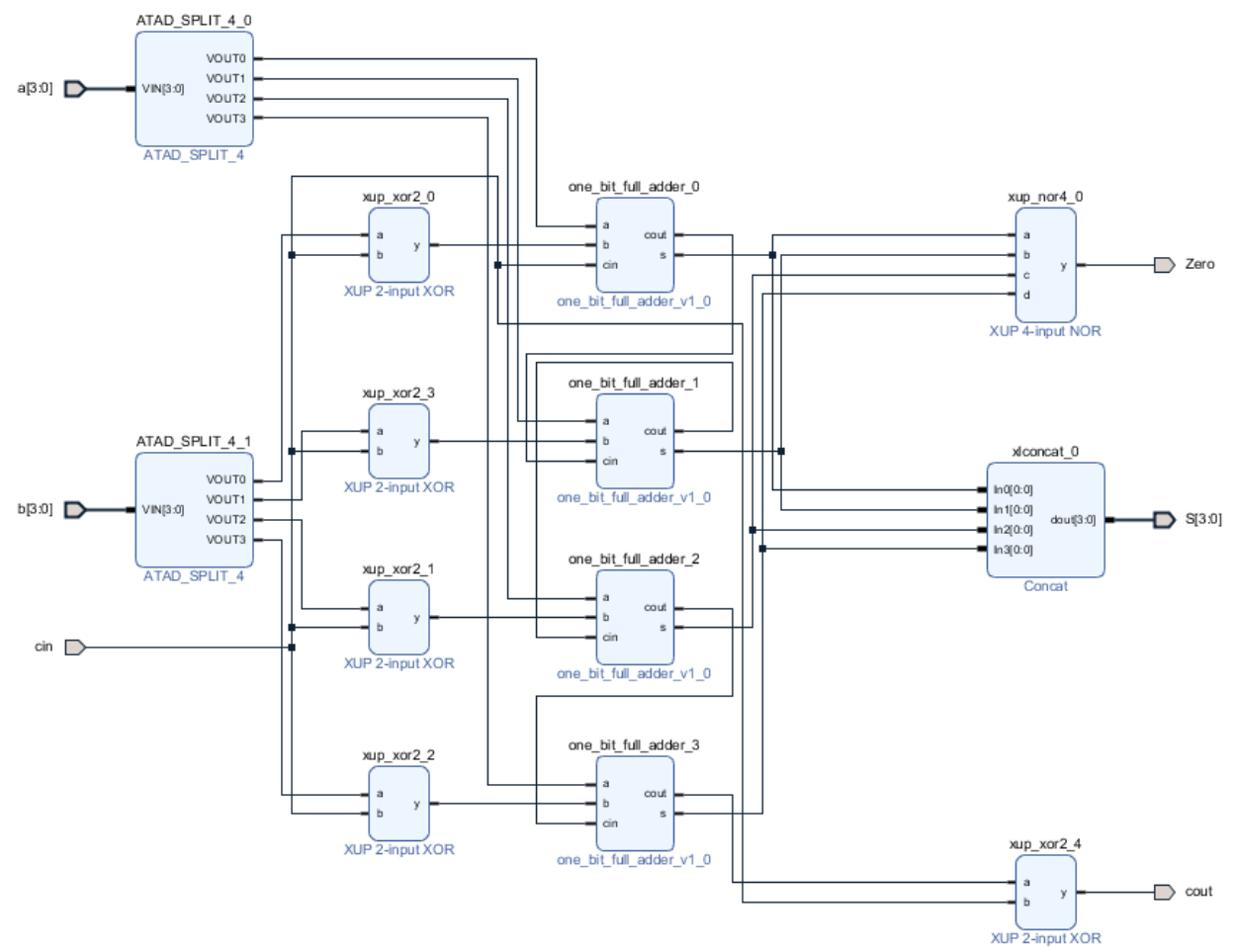


FIGURE 24 – Curcuit final

Pour simuler votre nouveau circuit, vous devez faire une clique droit sur le nouveau HDL wrapper que vous aurez créé et cliquez sur **Set as top**. Faites ensuite un clique droit sur **Run Simulation** dans la barre de gauche puis cliquez sur **Simulation Settings**. Assurez vous que **Simulation top module name** est bien réglé sur four_bit_full_adder. Si ce n'est pas le cas, changez donc cette valeur. Une fois fait, continuez la procédure habituelle.

Bravo, vous venez de finir la partie 1 de l'introduction à Xilinx Vivado ! Il ne vous reste plus qu'a tester votre circuit !