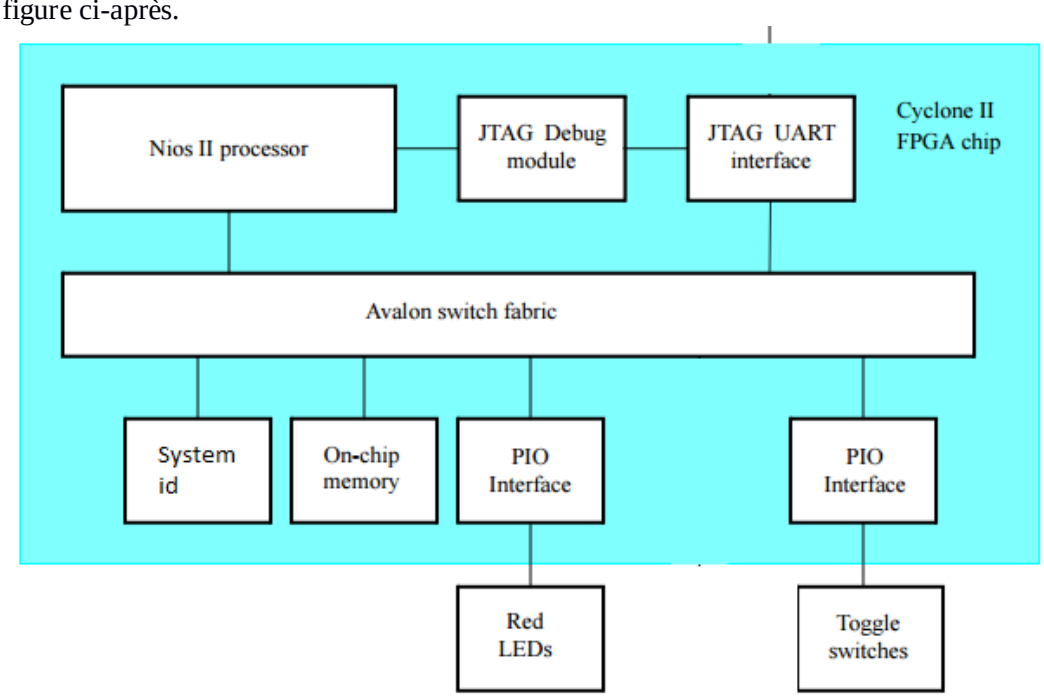
Université Tunis El-Manar	**Année Universitaire** : 2017-2018
Faculté des Sciences de Tunis	**Section :** IF5 Option ISEM
Module : Compléments de IP pour Systèmes Embarqués	
Enseignant : C.A. ABID	

Fiche TP4
Création d'un SoPC basé sur le NIOS II

L'objectif de ce TP est de s'initier à la conception conjointe d'un SoPC avec les outils fournis par Altera.

La création d'une application SOPC comprend les étapes suivantes :
1) Création du composant matériel (processeur + périphériques) dans l'environnement Qsys.
2) Téléchargement dans le composant FPGA (configuration) (environnement Quartus II)
3) Création du logiciel dans l'environnement NIOS2IDE, sa compilation puis son téléchargement dans la mémoire du SoPC.

Pour illustrer la démarche on propose de créer un microcontrôleur disposant de deux entrées connectées à deux boutons (switches) et de deux sorties logiques connectées à des LEDs. Le logiciel, développé en langage C, effectue le changement de l'état de LEDs selon l'état des boutons. Dans cette application la mémoire utilisée est intégrée au composant FPGA. L'architecture du SOPC est donné par la figure ci-après.

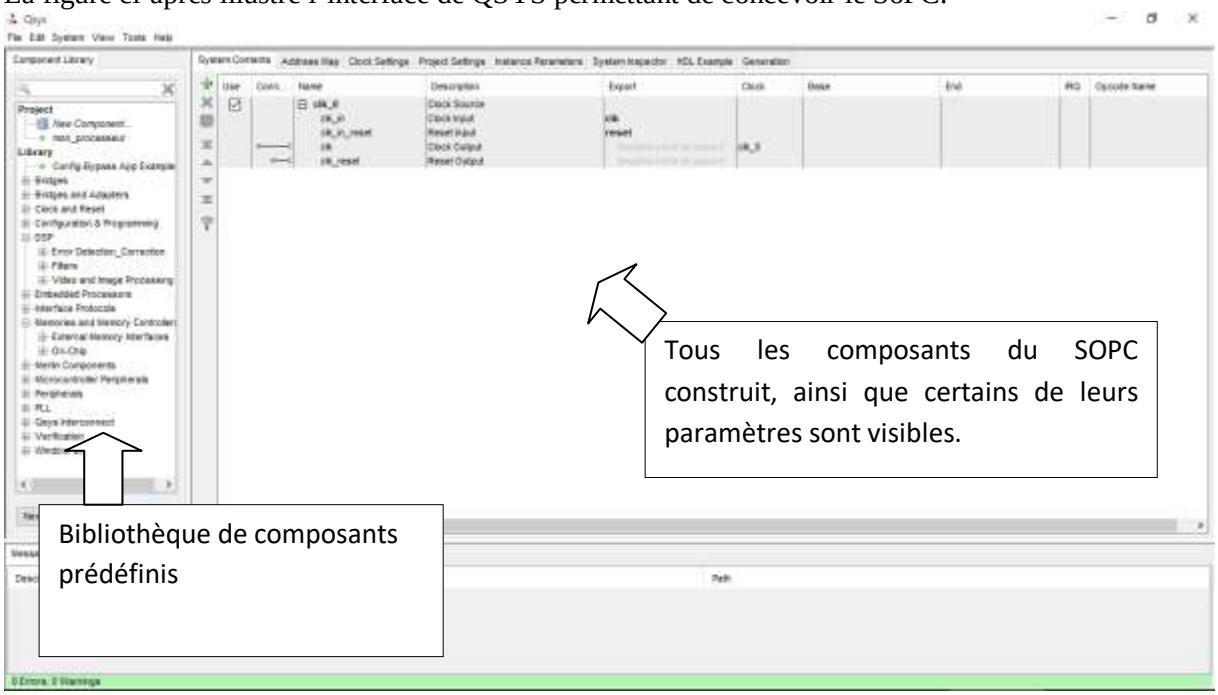


1) Création du composant matériel avec l'outil Qsys

Démarrer Quartus II et créer un projet en spécifiant le répertoire de travail, le nom du projet (TPSOPC par exemple) et le composant FPGA utilisé : ici on utilise une carte Altera DE1 équipée d'un FPGA cyclone II EP2C20F484C7.

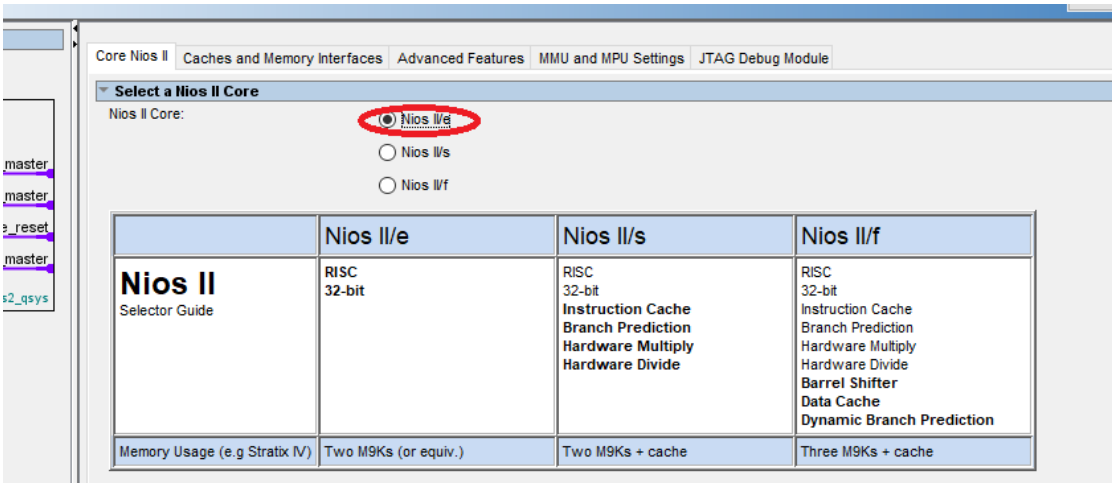
Dans l'environnement Quartus, lancer QSys à partir du menu Tools. L'outil QSys (ou son prédécesseur le SOPC Builder) permet de concevoir des microcontrôleurs spécifiques à une application. Ces microcontrôleurs comportent donc une partie **processeur** à laquelle on associe des périphériques (port PIO, Timers, UART, USB, composants propriétaires, …) et de la mémoire. Cette dernière peut-être embarquée dans le FPGA ou à l'extérieur du composant FPGA. La partie microprocesseur proprement dite est le NIOS2 de ALTERA, processeur de 32 bits.

La figure ci-après illustre l"interface de QSYS permettant de concevoir le SoPC.



On dispose par défaut du signal horloge pour faire fonctionner les différents composants intégrés sur la même puce ainsi que le signal d'initialisation reset actif à l'état bas.

On commence par ajouter le processeur NIOS II. Dans la colonne de gauche (la bibliothèque des composants), sélectionner « Nios II processor ». Il est possible d'effectuer une recherche en tapant 'nios'. Cliquer deux fois pour insérer le composant dans votre réalisation matérielle, puis sélectionner la version économique parmi les trois proposées. Faire « next » et accepter les options par défaut (sélectionner le debugger niveau 1). Renommer le processeur en "mon_processeur" en cliquant sur le bouton gauche de la souris puis en sélectionnant l'item 'rename'.

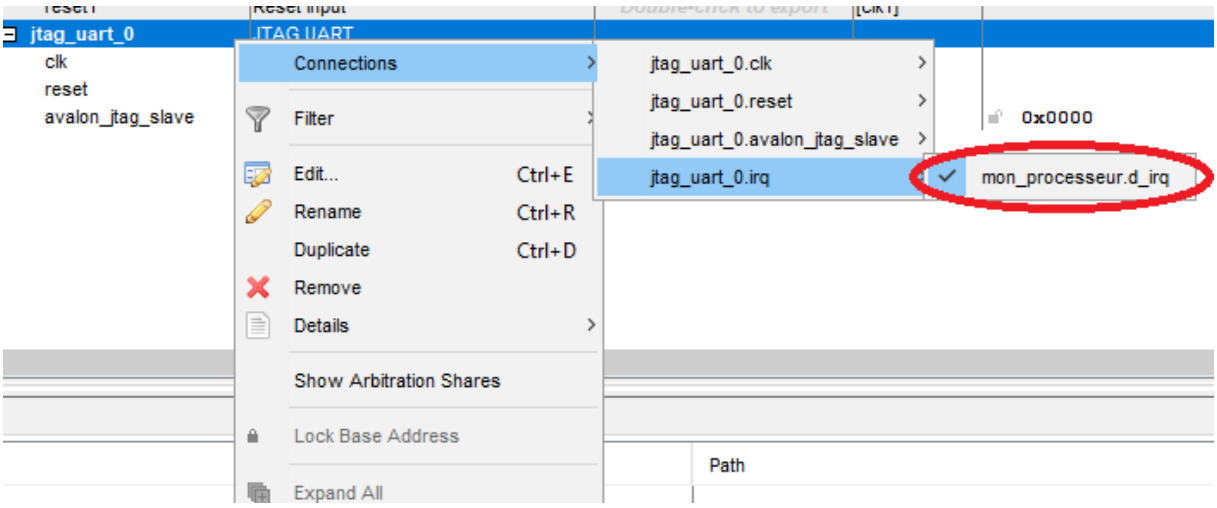


Maintenant, on procède par ajouter la mémoire volatile principale dans laquelle le programme à exécuter sera téléchargé. Pour cela, sélectionner la mémoire 'On-Chip Memory' à partir de la colonne auche, puis faire « add » et spécifier 10240 octets comme taille de mémoire. La limitation de la taille mémoire est due aux ressources mémoires (bistables) du circuit FPGA. Il est possible d'avoir plus d'espace mémoire en intégrant un composant extérieur au chip (des mémoires off-chip).

L'attribut clk1 doit être connecté à la sortie horloge principale.
L'attribut s1 doit être connecté aux attributs data_master et instruction_master du processeur. En effet cette mémoire va être utilisée pour ranger les instructions et les données de l'application logicielle.

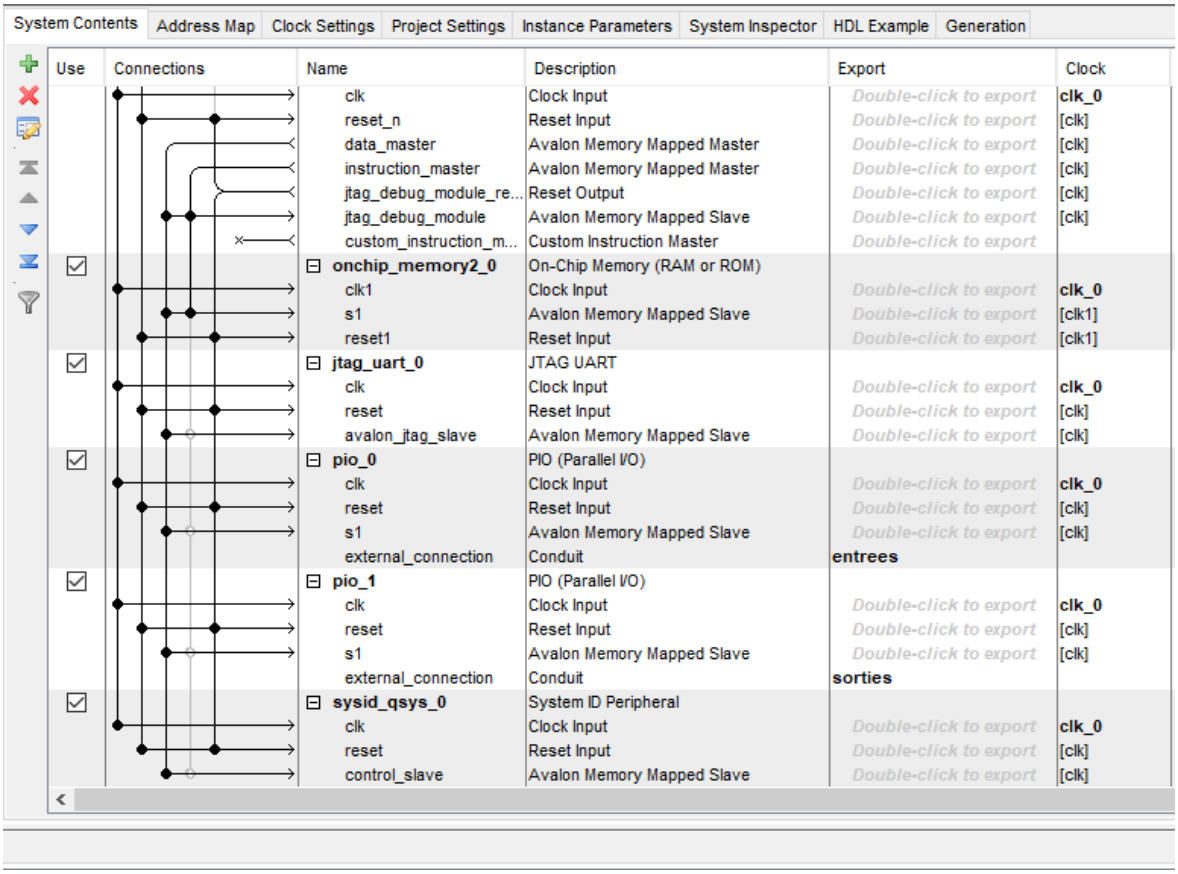
De la même manière, ajouter deux interfaces PIO de deux bits chacunes (Parallel Inputs Outputs) : une en entrée et une en sortie. Nota : on peut les renommer en faisant un clic droit sur le composant. L'interface PIO d'entrées, on l'appelle "switches", alors que l'interface PIO en sortie, on l'appelle "LEDs". À noter que l'attribut s1 de chaque périphérique ajouté doit être lié au bus de données connecté au processeur : s1 => mon_processeur.data_master. De plus, les deux interfaces parallèles devront être utilisées pour connecter, à travers les ports parallèles, des périphériques extérieurs au circuit FPGA. Afin de réaliser cette connexion, on exporte ces entrées et sorties en cliquant deux fois sur la colonne 'export' de l'attribut 'external_connection' de chaque interface.

Rajouter le composant Serial -> JTAG UART qui permettra de communiquer avec le PC hôte, télécharger le logiciel dans le circuit et effectuer le débogage. L'attribut avalon_jtag_slave doit être connecté à data_master du processeur. L'attribut avalon_jtag_slave.irq est connecté à l'attribut irq du processeur. En effet, le JTAG peut générer des interruptions, par exemple lors de débogage.

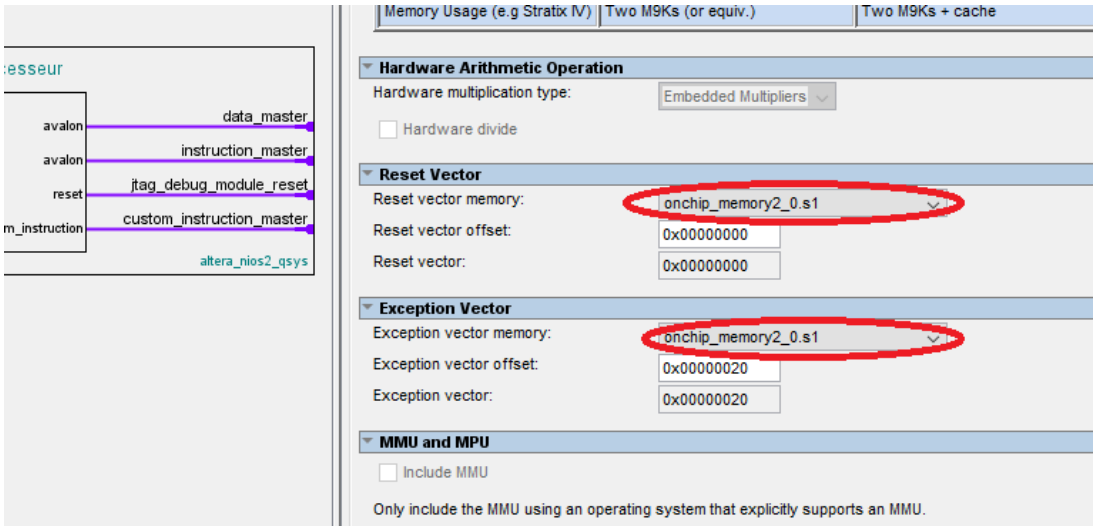


Pour des raisons de sécurité, on rajoute un composant « System ID » : celui-ci a pour rôle de donner un numéro d'identification au système que l'on a conçu et éviter ainsi de télécharger par erreur un programme qui ne correspondrait pas à l'application. L'attribut control_slave doit être connecté à l'attribut data_master du processeur.

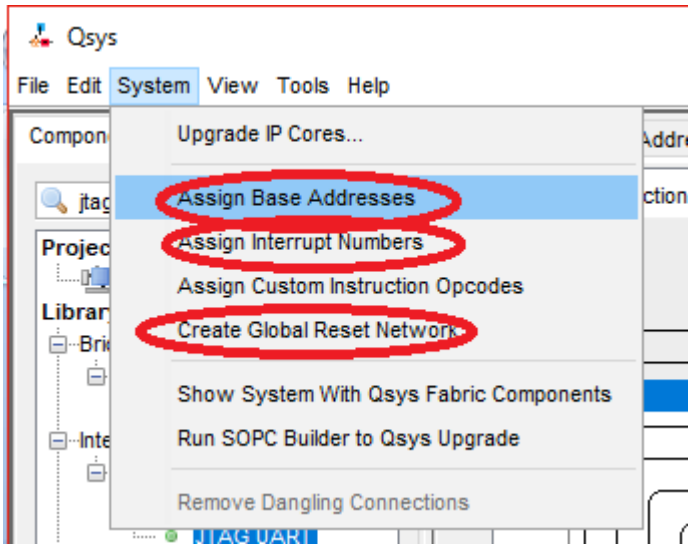
Connecter les entrées horloges des différents composants au signal horloge du système. Puis, réaliser les connexions comme le montre la figure ci-après.



Configurer le processeur pour que le vecteur d'exception et d'initialisation (reset) soient instanciés dans la mémoire utilisée (On-Chip Memory). En effet, la routine de gestion des exceptions et celle de réinitialisation doivent résider dans cette mémoire principale. Pour ce faire, cliquer deux fois sur 'mon_processeur', descendre en bas et choisir 'On-chip Memory' pour l'option 'Reset vector memory' et 'Exception vector memory'..

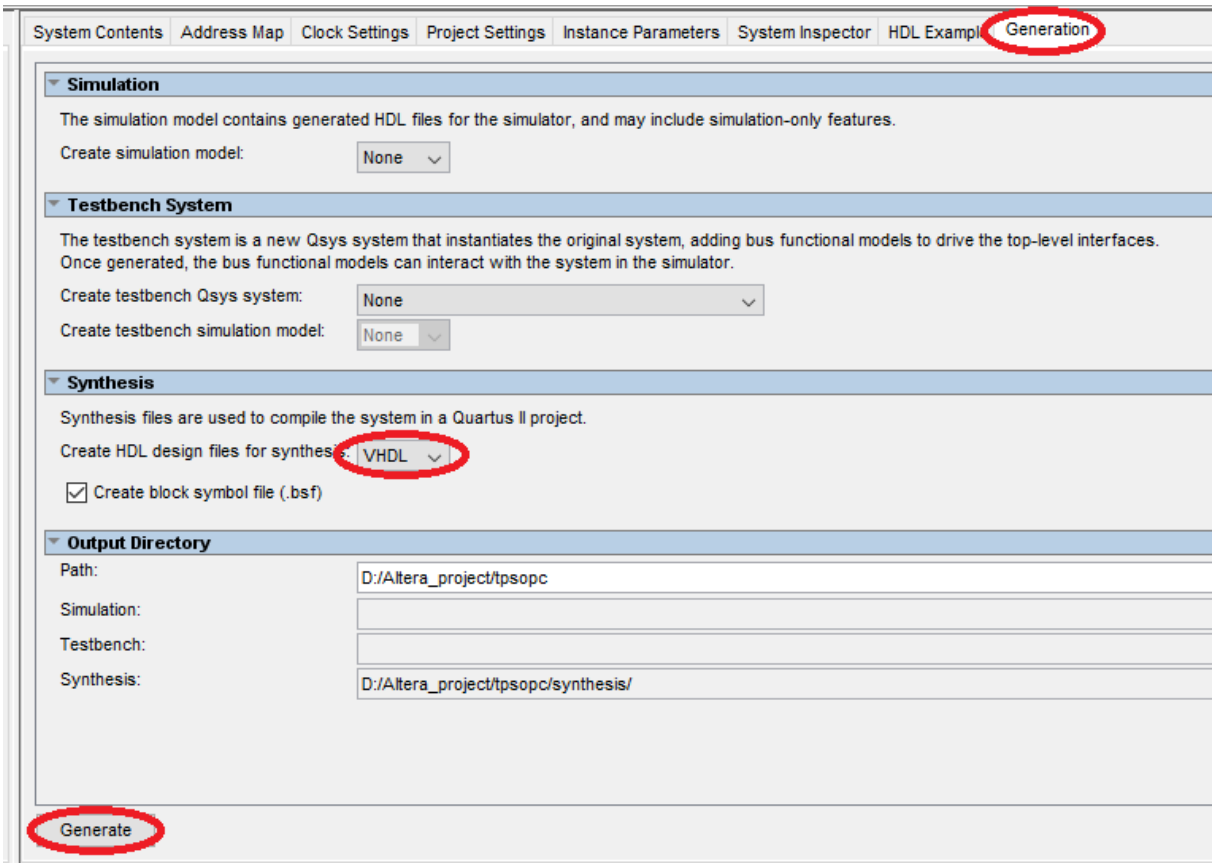


Faire ensuite « System » puis « auto-assign base adress ». Ceci a pour effet d'assigner automatiquement une adresse logique à chaque ressource dans l'espace mémoire adressé par le processeur. De même manière, assigner les interruptions et créer un reset global.

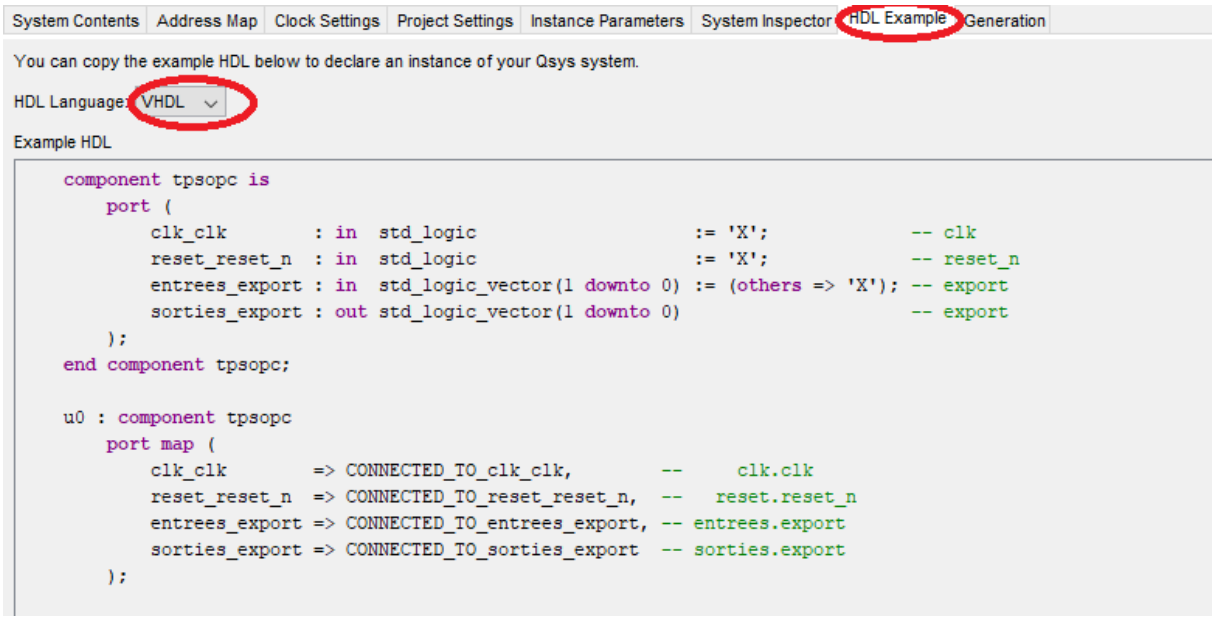


À ce stade le SOPC est défini : type de processeur, horloge, périphériques utilisés, taille mémoire, adresses physiques des composants.

Une fois tous les composants rajoutés, choisir l'onglet 'Generation'. Puis spécifier VHDL comme étant le langage de description utilisé pour la description du SoPC. Finalement, cliquer sur le bouton 'Generate'. Ceci a pour effet de générer les fichiers VHDL et le symbole graphique associés au SoPC que l'on vient de définir. À ce stade le composant a été créé.



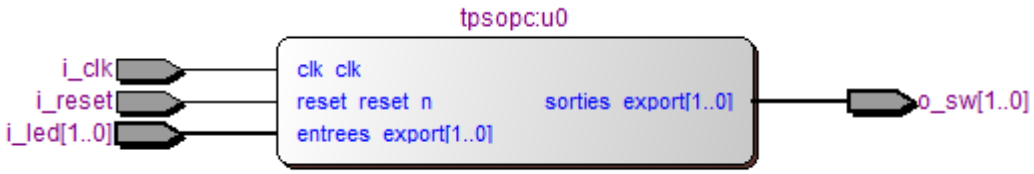
Il est possible d'avoir un exemple de code VHDL permettant l'instanciation du SoPC à partir de l'onglet HDL exemple.



2) Téléchargement dans le composant FPGA (configuration)

On revient à l'outil Altera Quartus II en créant le fichier VHDL principal (top level) dans lequel on instancie le processeur créé tout en assignant les ports d'entrées-sorties. D'abord, il faut inclure tous les fichiers de description relatifs au SoPC créé. Ces fichiers sont localisés dans les sous répertoires './synthesis' et '/synthesis/submodules' dans le répertoire du projet.

La représentation en bloc du module principal est présentée dans la figure ci-après. Dans ce composant, les entrées et sorties de l'instance du SoPC créé doivent être connectées aux entrées et sorties de ce module principal. D'autre part, l'entrée horloge 'i_clk' du module doit être liée à l'horloge fonctionnant avec la fréquence 50Mhz. L'entrée 'i_reset' est assignée à un switcher ou un bouton poussoir. La sortie i_led (vecteur de taille 2) doit être connecté à des leds de votre choix. De manière similaire, la sortie o_sw doit être connecté à des switchers.



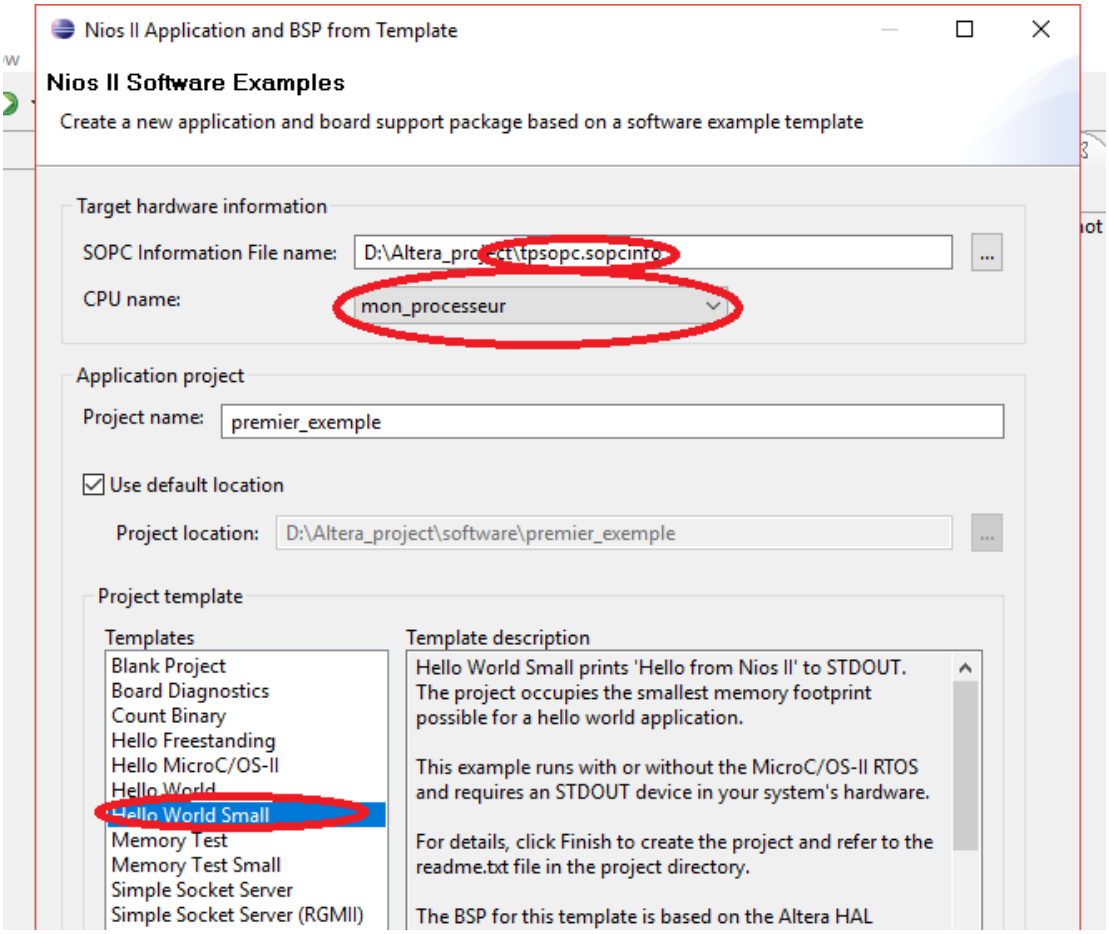
Réaliser la synthèse de la configuration matérielle préparée, puis utiliser l'outil 'programmer' pour configurer le circuit FPGA. S'il n'y a pas de problème alors la SoPC est bien prêt pour exécuter tout programme chargé dans sa mémoire.

3) Développement du logiciel

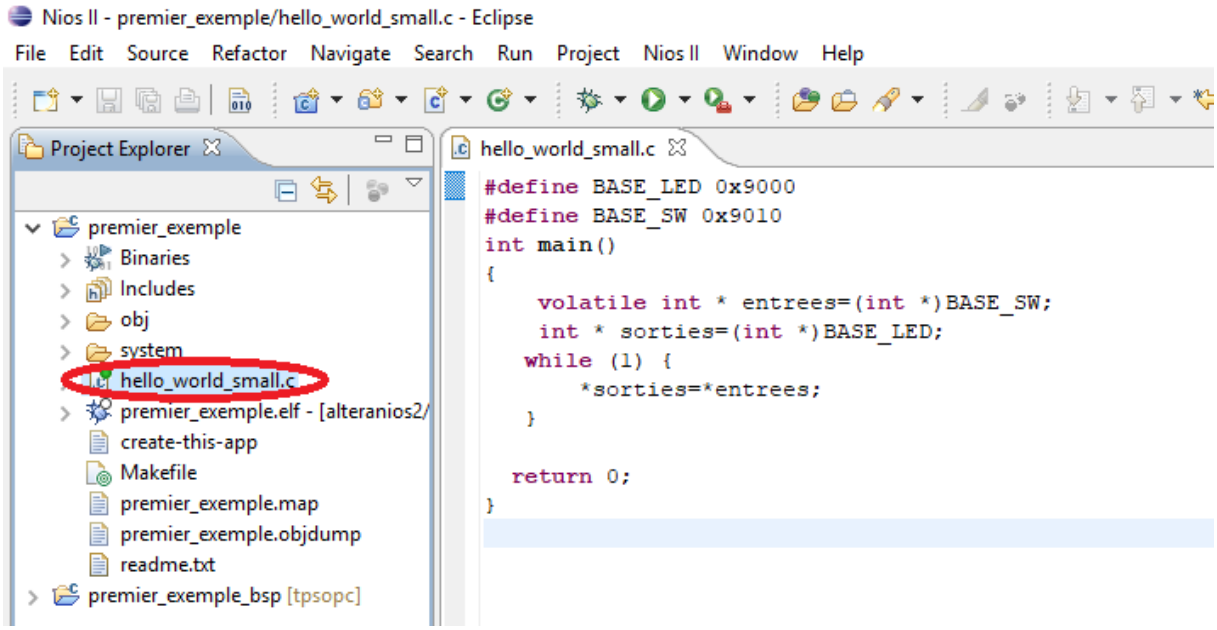
On s'intéresse maintenant à la partie logicielle de l'application. On commence par lancer NIOS2IDE. Il s'agit d'un environnement de développement basé sur Eclipse permettant d'écrire des programmes C destinés à la configuration matérielle préparée. Les étapes pour développer une application se résument comme suit :
- définition d'un espace de travail (Workspace)
- création du projet et de la bibliothèque
- création du programme
- compilation
- téléchargement et exécution

1) Définition de l'espace de travail (Workspace)
Au lancement de NIOS2IDE, un espace de travail est sélectionné par défaut. La fenêtre « Navigator » généralement située à gauche, répertorie tous les projets déjà existant dans cet espace de travail. Si on souhaite changer d'espace faire : **File => Switch Workspace** puis sélectionner le nouvel espace de travail. De préférence sélectionner le dossier du projet créé au début.

2) Création du projet et de la bibliothèque
Faire File => New, cliquer sur **NIOS II application and BSP from template**, localiser et sélectionner le fichier tpsopc.sopcinfo' spécifiant les informations de l'architecture matérielle cible pour le projet. Donner un nom au projet (ex. : premier_exemple). Comme template, choisir 'Hello world small'. Un tel template est utilisé pour avoir une taille minimale du programme exécutable, puisque la taille de mémoire On-chip pour cet exemple est limitée à 10240 octets.



4) Création du programme
Taper le programme illustré dans figure ci-après dans le fichier hello_world_small.c tout en veillant à spécifier les adresses de base correctes assignées aux interfaces PIO.



5) Compilation
Compiler les projets créés.

6) Téléchargement du programme et exécution sur la cible (NIOS II HW configuration)
Le téléchargement et l'exécution du programme se font par la commande « run » ou « debug ». Cliquer droit sur le projet et sélectionner :
- Run as : pour télécharger et exécuter le programme immédiatement.
- Debug as : pour télécharger et exécuter le programme en mode debug.