TP : Implanter un processeur dans le FPGA.
Programmer ce processeur. Accélérer le code.

0 Présentation du TP :

Pré-requis : ⇒ Avoir suivi les TP_description_schématic_compteur-FPGA et TP_compteur_VHDL_virtual_instruments-FPGA.
⇒ Connaissance du langage C ANSI.
Durée estimée : ⇒ 2 heures.
Objectif : ⇒ Implanter et programmer un **processeur** dans un **FPGA.**

Vocabulaire spécifique au TP :

IP : **I**ntelectual **P**roperties. Les IP sont des descriptions de fonctions à intégrer dans un FPGA.
Cela permet à un développeur un gain de temps considérable, en implantant le code des fonctions IP déjà écrites.

Open Bus : C'est un schéma fonctionnel mettant en œuvre des **IP** sous ALTIUM.

SoC : **S**ystème **o**n **C**hip. C'est le concept d'intégrer une fonction électronique dans un composant programmable.

Avantage d'un SoC:
Accroître les performances d'un système
Réduire le coût d'assemblage
Réduire la puissance consommée
Assurer l'intégrité du signal: limiter la diaphonie, les temps de propagation, la réflexion du signal à haute fréquence
Améliorer les contraintes mécaniques.

Technologies permettant la mise en œuvre d'un SoC :
Full custom
Standard cell
FPGA

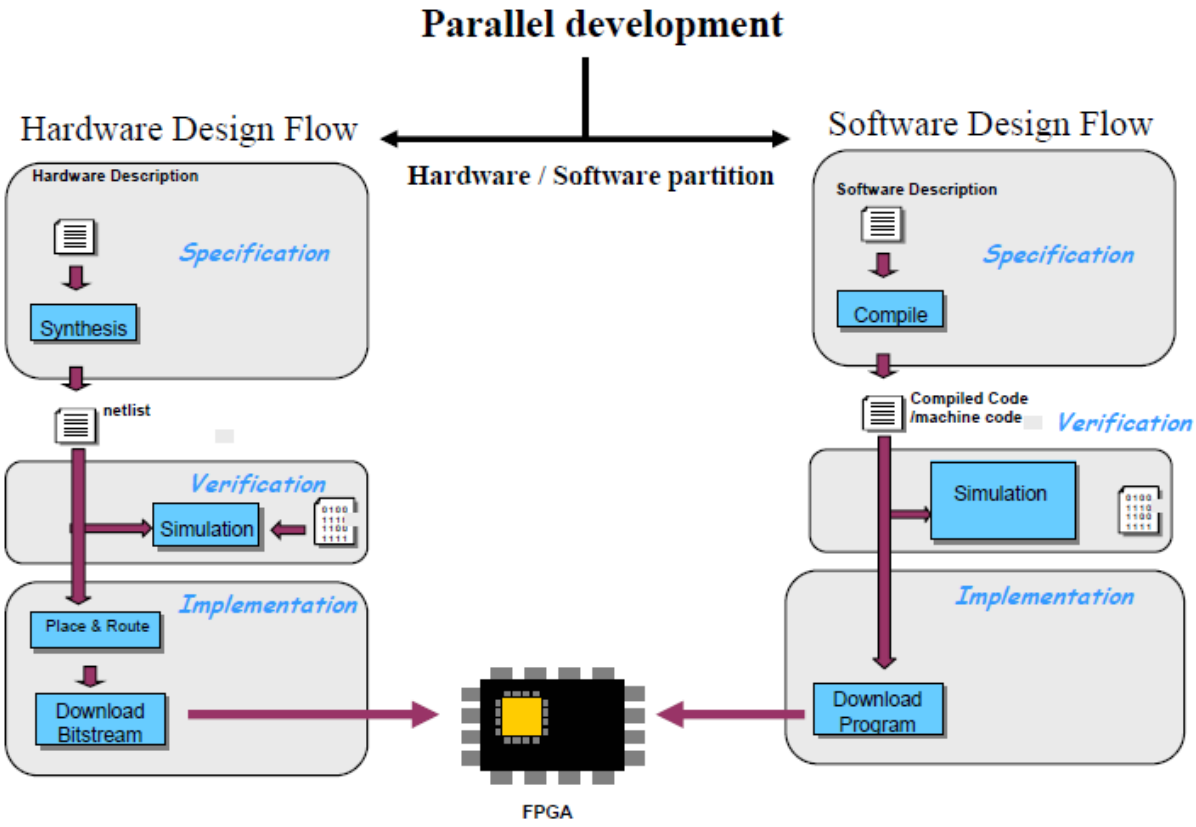
Durant ce TP vous implanterez un **processeur** dans un **FPGA**.
Pour cela vous aller éditer un schéma **Open Bus** mettant en œuvre des **IP**.
Le processeur ainsi décrit est un **SoC**.

Sommaire du TP :

Dans ce TP nous serons amenés à implanter un processeur **SoC**, puis à le programmer en langage C :

Cela implique le développement de deux programmations en parallèles :

⇒ L'implantation dans le FPGA du processeur à partir d'IP : HARDWARE DESIGN FLOW.
⇒ La programmation du code en C : SOFWARE DESIGN FLOW.



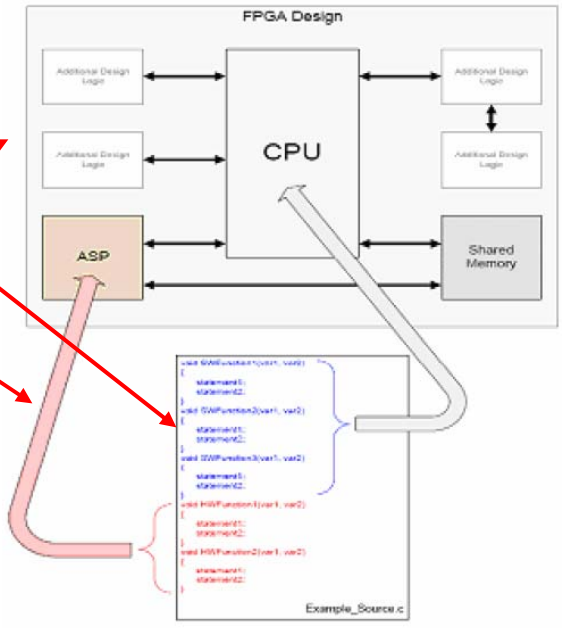
Organisation fonctionnelle de notre projet :

1 : La partie hard est organisée autour du processeur **SoC** implanté dans le **FPGA**.

2 : Le soft, écrit en **langage C**, décrit le programme exécuté par le **FPGA**.

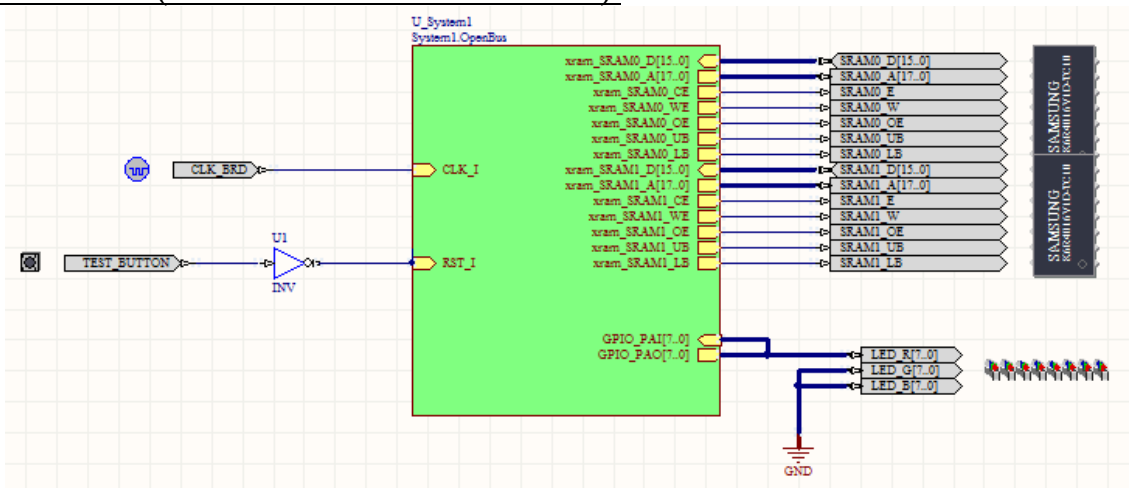
3 : Afin de diminuer le temps d'exécution du programme une partie du code **C** est converti en **VHDL** par la fonction **ASP**

ASP : Application **S**pecific **P**rocessor

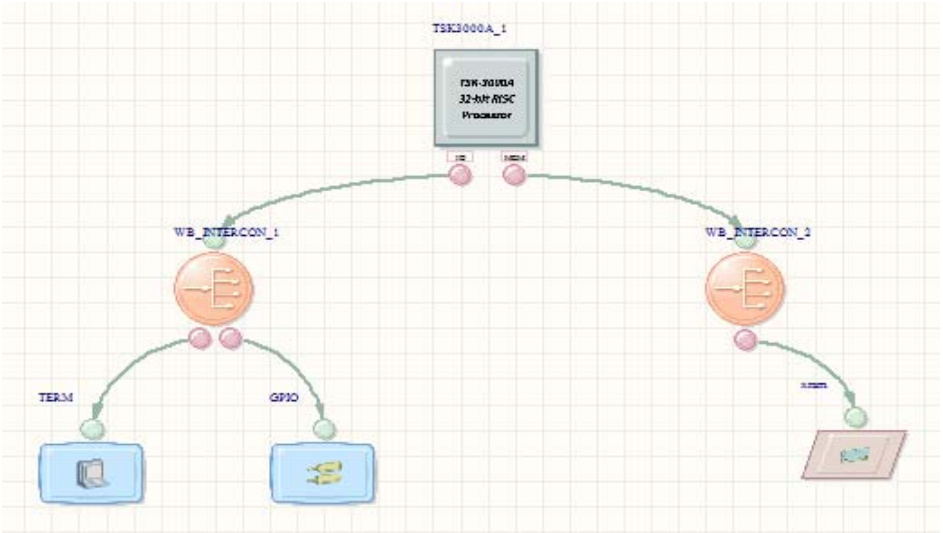


Schémas à dessiner durant ce TP4 :

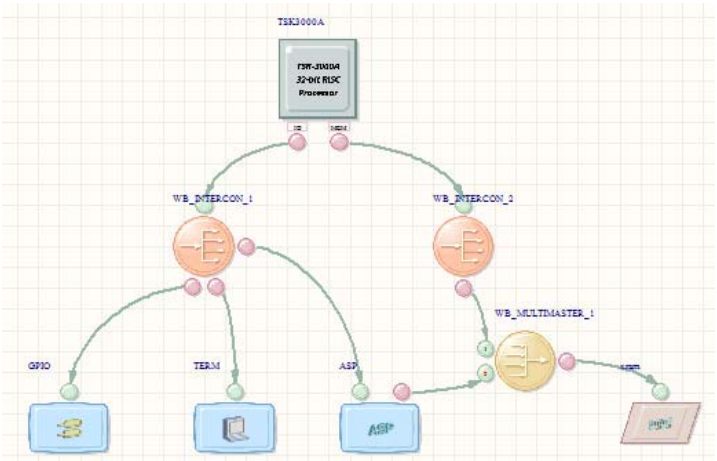
Schéma TOP : (HARDWARE DESIGN FLOW):



1er Schéma Open bus à dessiner: tout le code développé est dédié au processeur :
« SOFWARE DESIGN FLOW »



2éme Schéma Open bus à dessiner: afin d'accélérer l'exécution la fonction CRC16 est traduite du C en une structure intégrée dans le FPGA: « HARDWARE DESIGN FLOW ».



Code C calculant le CRC16 de la mémoire (SOFWARE DESIGN FLOW) :

Fonction main :

```
#include <stdlib.h>
#include <stdint.h>
#include "hardware.h"
#include <stdio.h>



extern uint16_t crc16 ( uint8_t * ptr, size_t count);
volatile uint8_t * const leds = (void *)Base_GPIO;


void main( void )
{
    *leds = 0;  // Initialize the LEDs to all OFF
    for ( ;; )
    {
        (*leds)++;
        crc16((uint8_t *)Base_xram, Size_xram);

    }
}
```

fonction CRC16 :

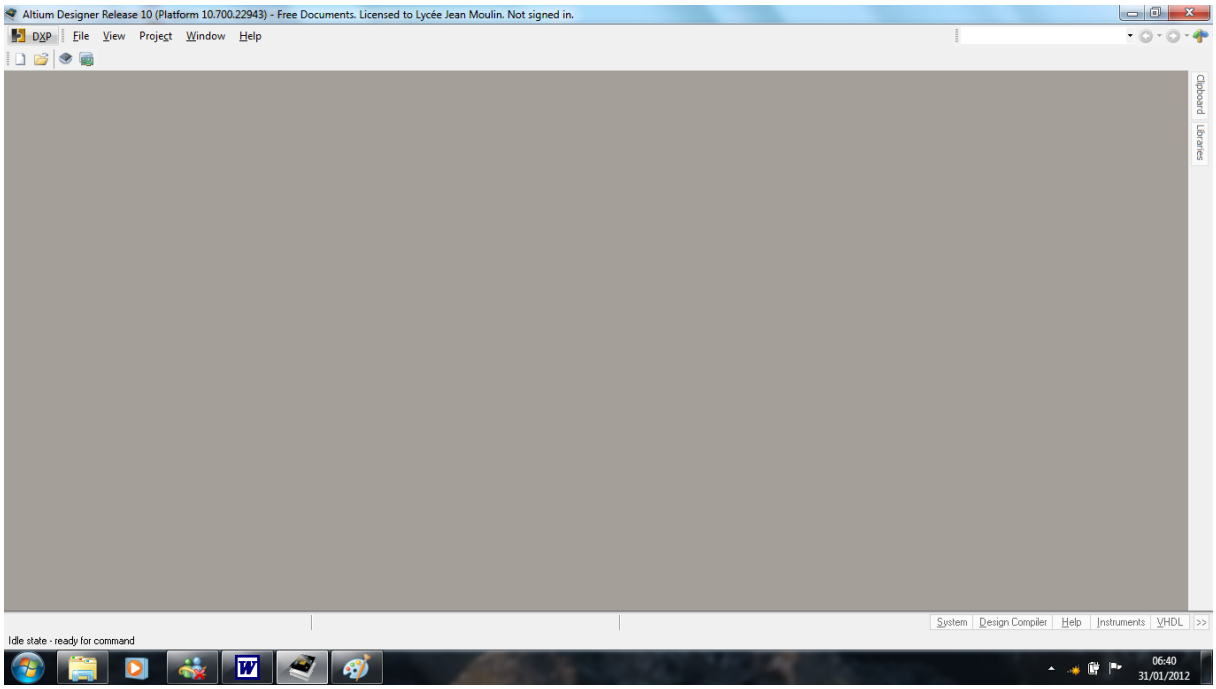
```
uint16_t crc16(uint8_t * ptr, size_t count)
{
         uint16_t crc=0;
         do
         {
            crc=crc ^ (uint16_t)*ptr++ << 8;
            for (int bitcount = 0; bitcount < 8; bitcount++)
            {
                if (crc & 0x8000)
                   crc = crc << 1 ^0x1021;
                else
                    crc <<=1;
            }
         } while(count-- );
         return crc;

}
```

1 Créer un nouveau projet FPGA.

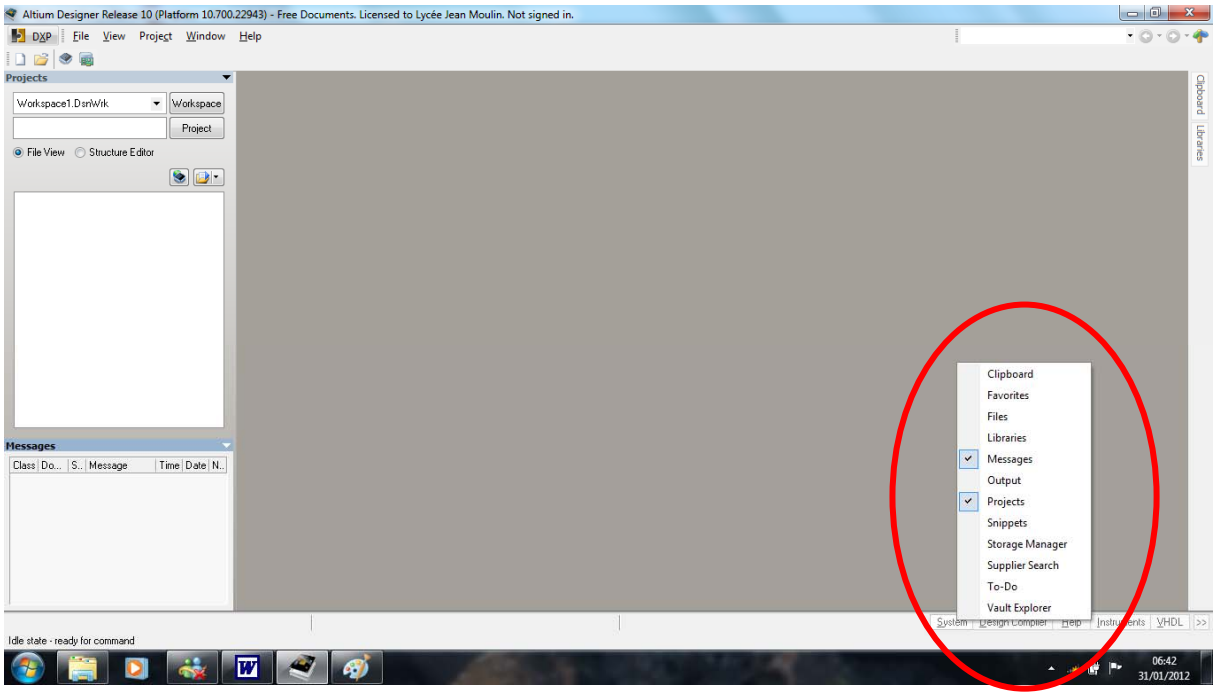
Conseil : créer un nouveau répertoire dans lequel vous placerez tous les éléments du présent projet dont les fichiers main.c et crc16.c

1.1 Repartir d'un environnement vide :



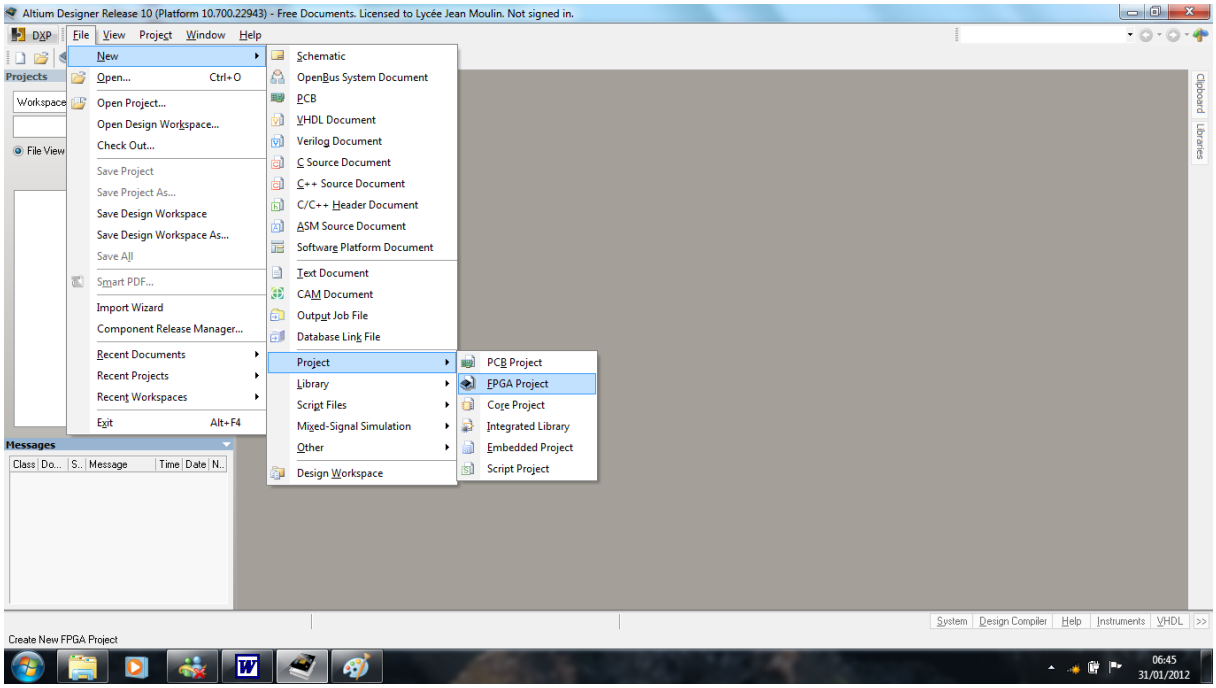
1.2 Ouvrir les fenêtres projet et message :

Paramétrer l'environnement de travail d'Altium Designer :
⇒ Commande ⇒ **System** (en bas à droite) ⇒ **Messages et Projects.**

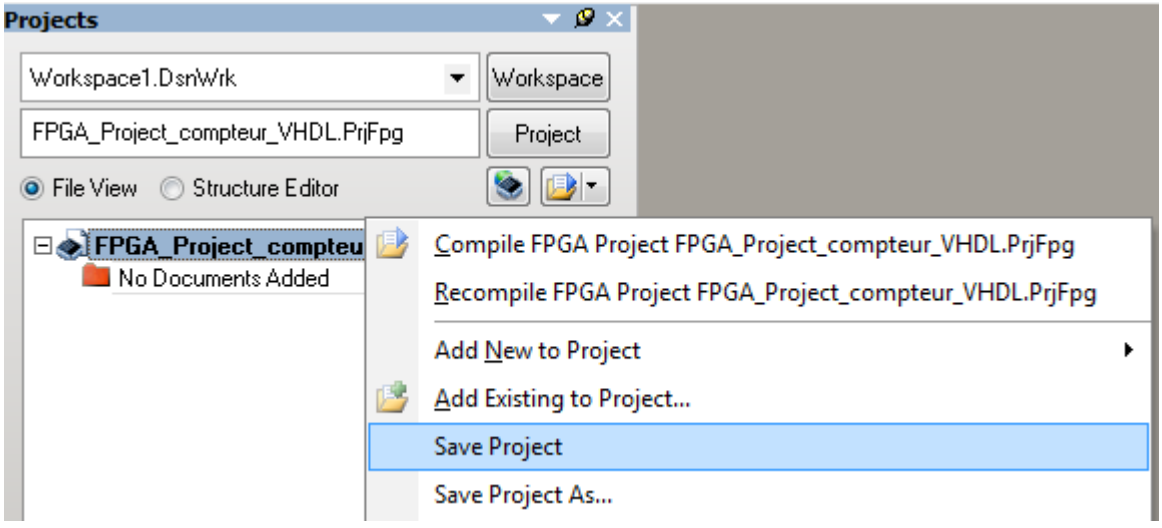


1.3 Créer et renommer le projet :

Créer un nouveau projet en utilisant la commande :
⇒ **File ⇒ New ⇒ FPGA Projet**.



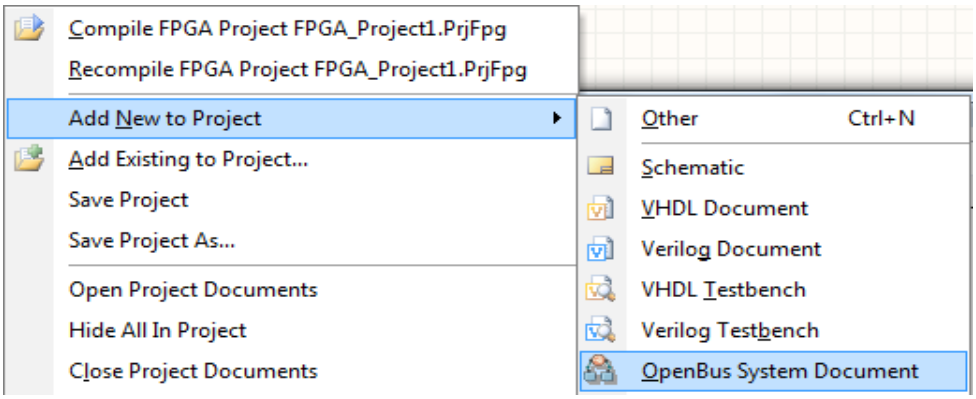
⇒ Cliquer droit sur le nom du nouveau projet.
⇒ Choisir la commande **Save Projet as «FPGA_TP4.PrjFpg»** pour sauvegarder le projet dans le répertoire de travail.



Remarque : Les caractères espace ( ) et/ou tiret (-) ne doivent pas être utilisés dans les noms du projet ou des documents. Le caractère underscore (_) peut être utilisé pour améliorer la lisibilité.

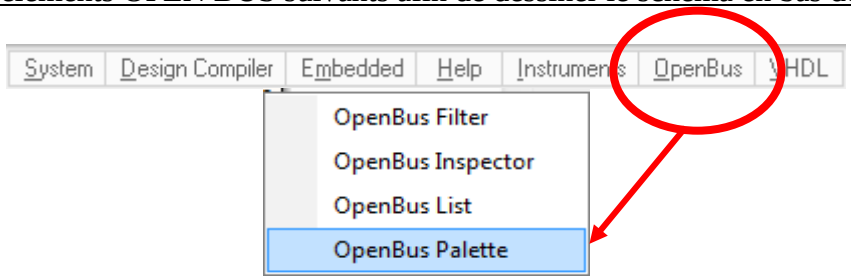
2 Editer le fichier OPEN_BUS.

2.1 Ajouter au projet un fichier OPEN_BUS et sauvegarder ce fichier :
⇒ Clic droit sur le nom du projet FPGA dans l'onglet « **Projets** »
⇒ Commande : **Add New to Project** ⇒ **OpenBus System Document**



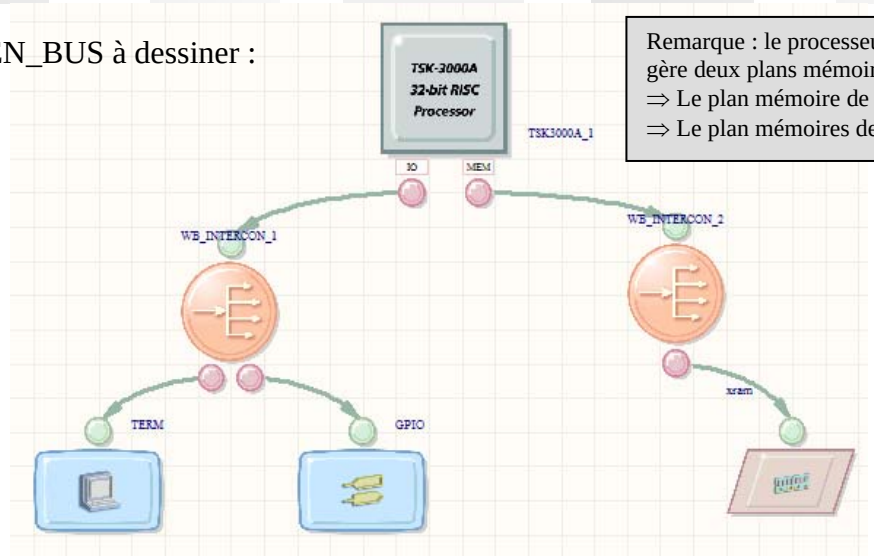
⇒ Clic droit sur le nom « **Sheet1.SchDoc** » dans l'onglet Projets
⇒ Commande ⇒ **Save As** : « **Open_Bus_TP4.OpenBus** » dans le dossier du projet.

2.2 Placer les éléments OPEN BUS suivants afin de dessiner le schéma en bas de page :



⇒ Eléments à placer :

⇒ Le schéma OPEN_BUS à dessiner :

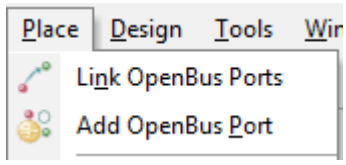


Remarque : le processeur TSK3000A gère deux plans mémoires distincts :
⇒ Le plan mémoire de données (MEM)
⇒ Le plan mémoires des entrées sorties (IO)

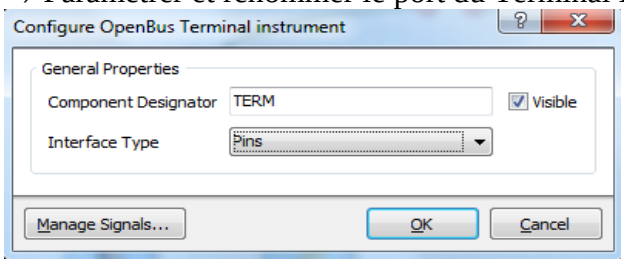
⇒ Dessiner les liens entres les éléments **OPEN_BUS** :
⇒ Clic gauche pour sélectionner le composant **WB_INTERCON1**.
⇒ Menu : **Place ⇒ Add OpenBusPort.** Placer le port sur le composant **WB_INTERCON1.**

Pour dessiner les liens entre les structure **OPEN BUS** utiliser la fonction **LINK**

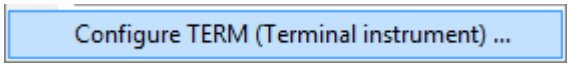
Pour rajouter les ports aux inter connexions utiliser la fonction **ADD open bus port**



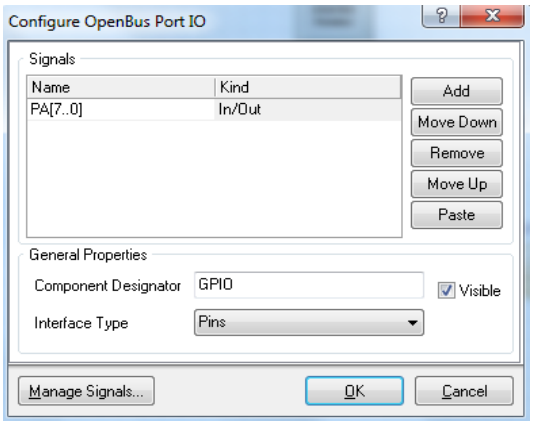
⇒ Paramétrer et renommer le port du Terminal Instrument comme ci-dessous :



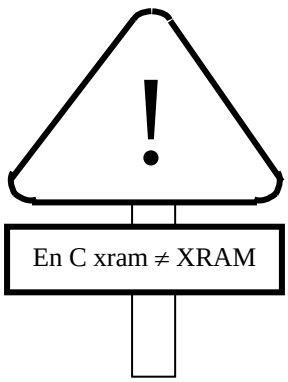
Pour paramétrer un élément OPEN BUS : cliquez droit sur le composant menu CONFIGURE…



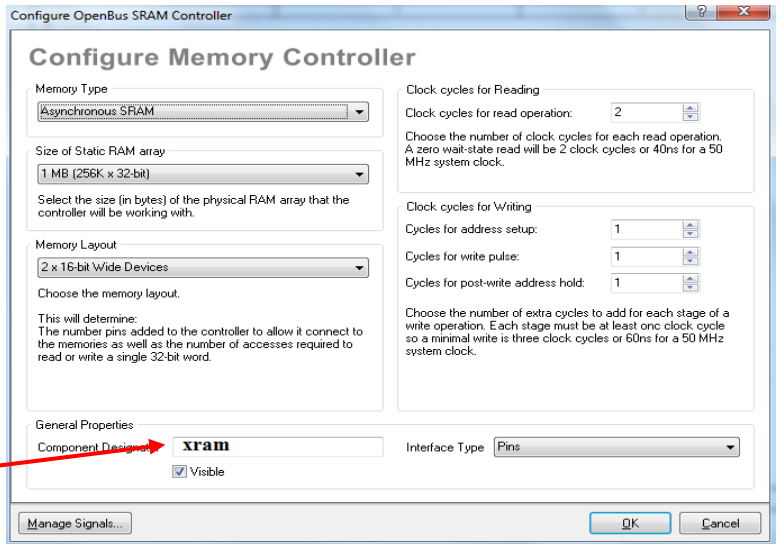
⇒ Paramétrer et renommer le port d'entrées-sorties comme ci-dessous :



⇒ Paramétrer et renommer le port mémoire :

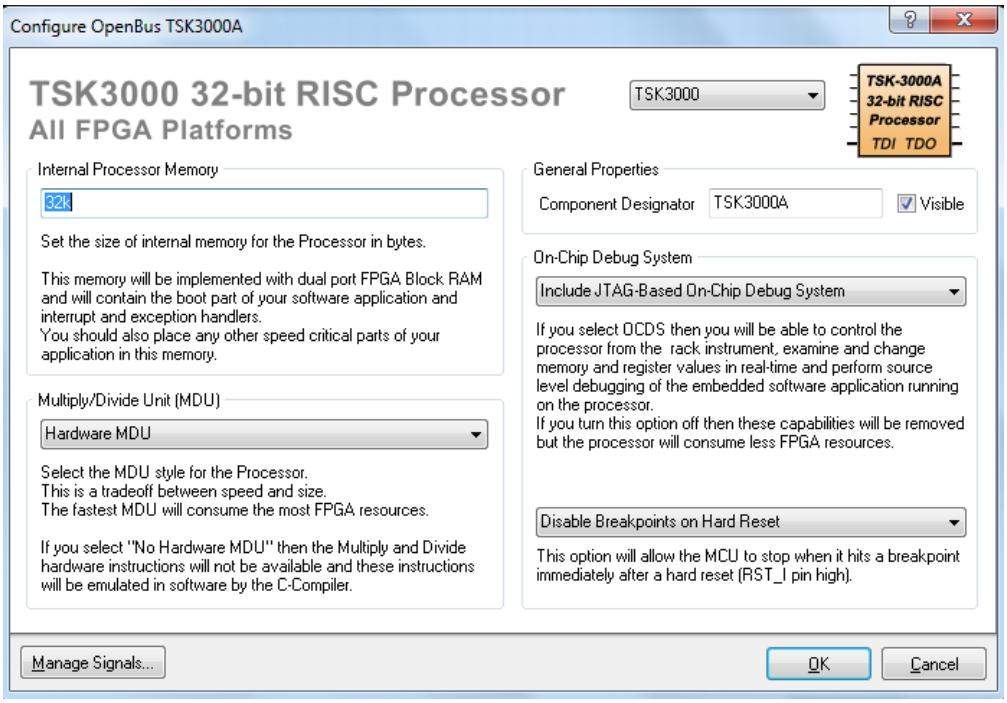


xram en lettres minuscules

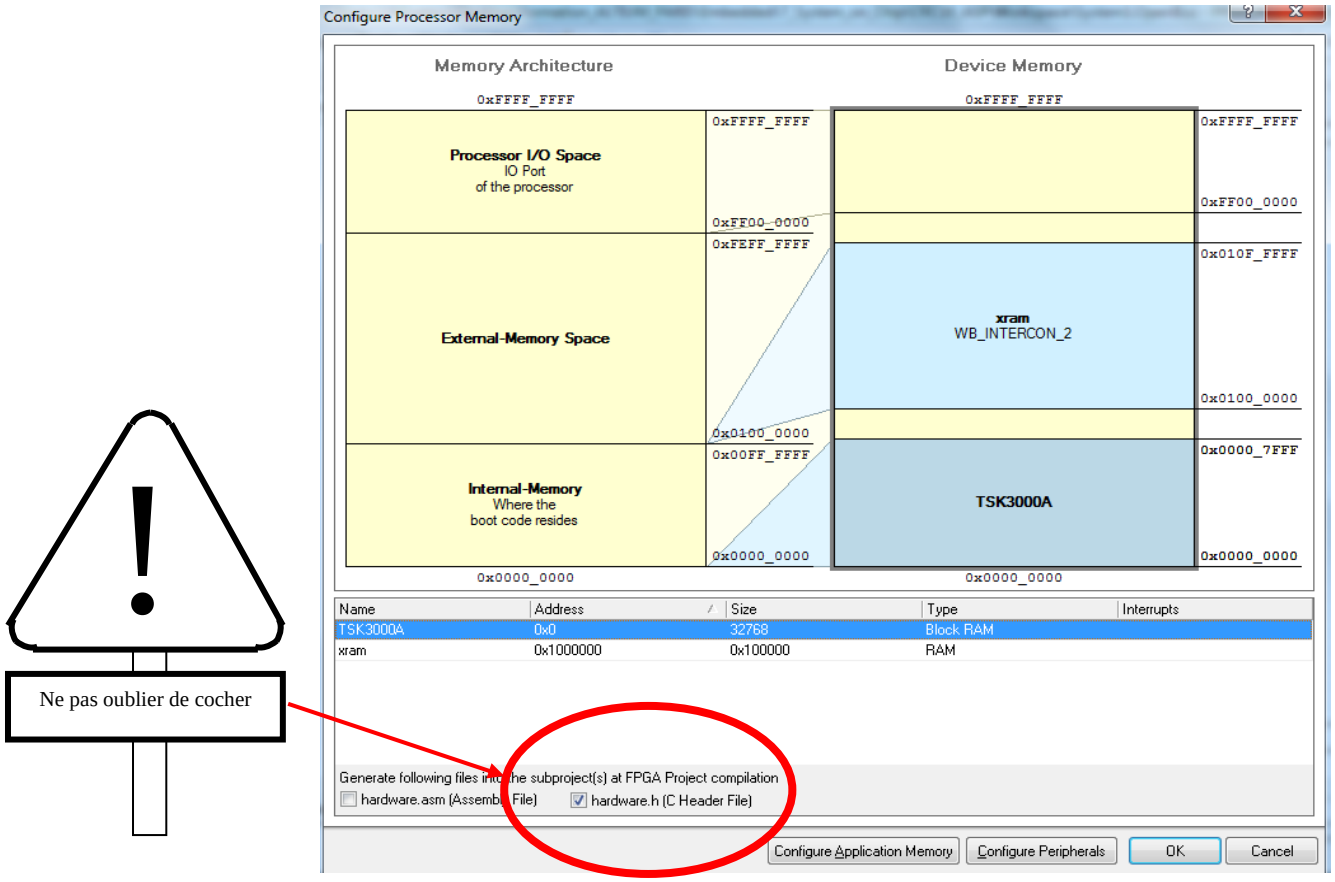


⇒ Paramétrer le processeur comme ci-dessous :

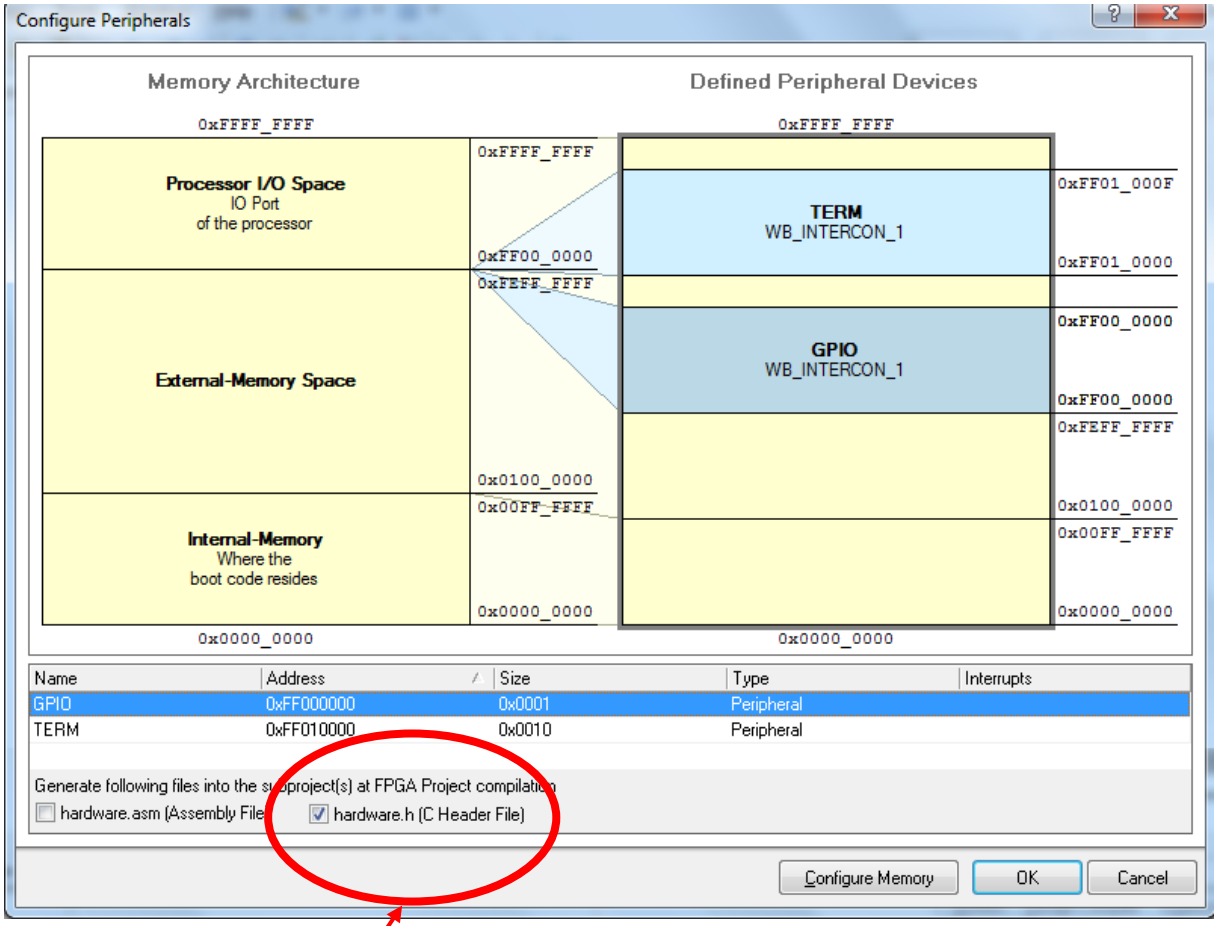
⇒ Clic droit sur le processeur TSK 3000A ⇒ **Configure TSK3000A**



⇒ Clic droit sur le processeur TSK 3000A ⇒ **Configure Processor Memory**

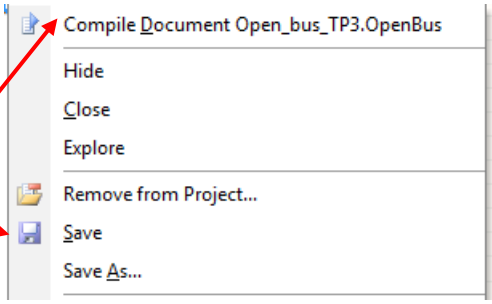


⇒ Clic droit sur le processeur TSK 3000A puis dans l'onglet choisir **Configure Processor Peripherals :**



⇒ Clic droit sur « **Open_Bus_TP4.OpenBus** »

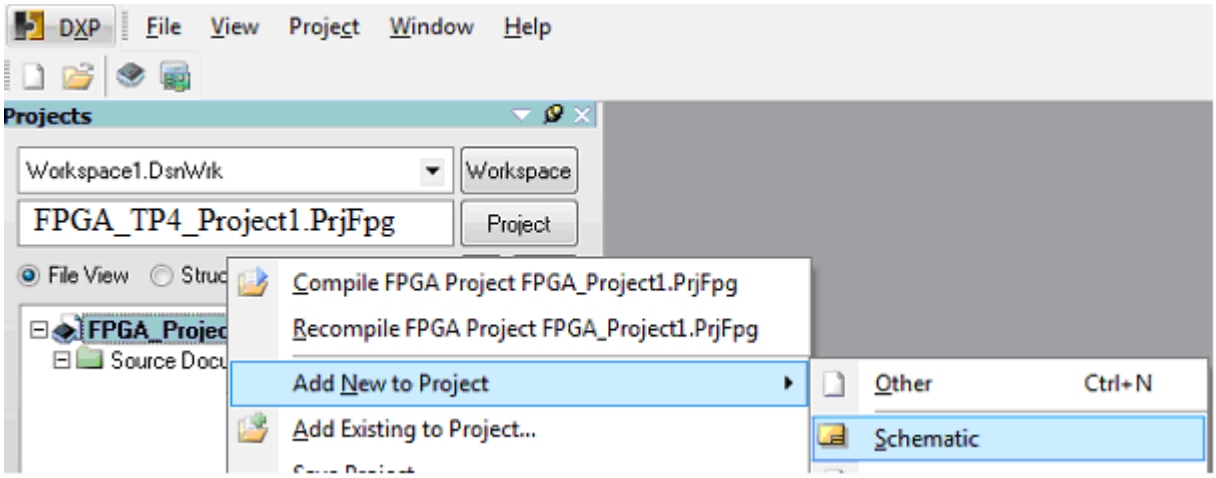
⇒ Compiler et sauvegarder le fichier **:**
« **Open_Bus_TP4.OpenBus** »



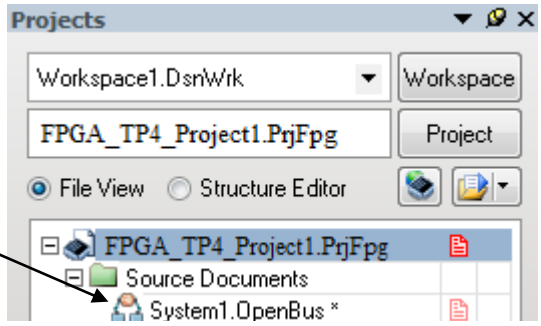
3 Dessin du TOP schéma.

3.1 Création du fichier schéma en tête du projet FPGA :

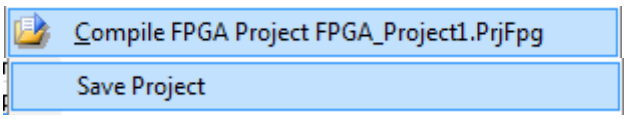
⇒ Ajouter le fichier **« Sheet1.SchDoc »** :
⇒ Clic droit sur **« FPGA_TP4_Project1.PrjFpg »** ⇒ **Add New to Project** ⇒ **Schematic.**



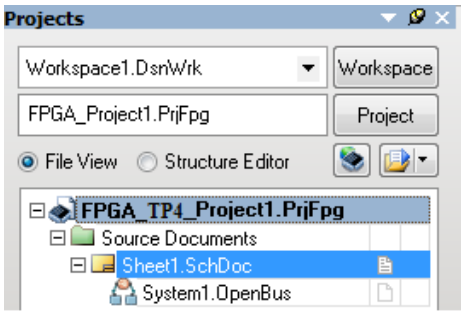
Remarque : à ce stade le schéma Sheet1.SchDoc est hiérarchiquement sous le fichier OPEN_BUS.



⇒ Recompiler et sauver votre projet :



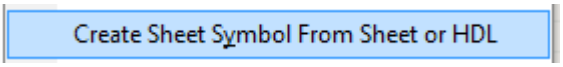
⇒ Ce que vous devez obtenir : suite à la compilation du projet FPGA le schéma est replacé en tête du projet.

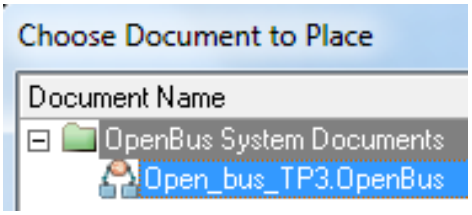


Remarque : si des erreurs apparaissent dans la fenêtre **« message »** suite à la compilation cela est dû au fait que votre schéma OPEN BUS n'est pas complet : rechercher et modifier les erreurs.

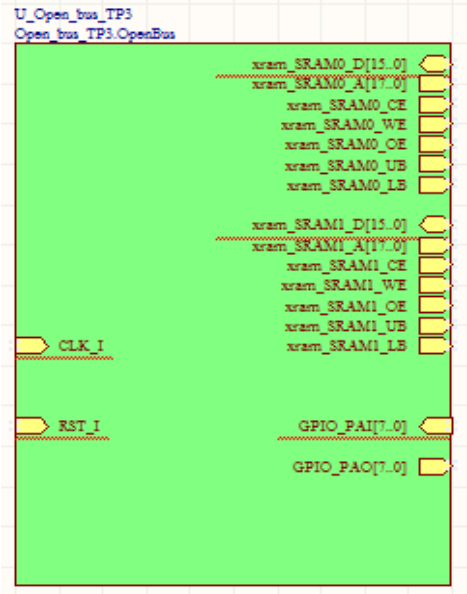
3.2 Placer dans le schéma le symbole créé à partir du fichier OPEN BUS :

Depuis la fenêtre Sheet1.SchDoc
⇒ **Menu :** DESIGN
⇒ **Commande :** Create Sheet Symbol from sheet or HDL





⇒ Organiser le corps du symbole comme ci-dessous en déplaçant les ports en jaune:



3.3 Placer les ports d'entrées sorties du FPGA dans le schéma : voir schéma page 2 :

⇒ Placer dans le schéma les éléments suivants :

Description	Nom de la fonction	Bibliothèque
Entrée de l'horloge paramétrable	CLOCK_BOARD	FPGA NB3000 Port-Plugin.IntLib
Test / Reset Button	TEST_BUTTON	FPGA NB3000 Port-Plugin.IntLib
Commande du Barre-graphe 8 LED 3 couleurs	LEDS RGB	FPGA NB3000 Port-Plugin.IntLib
Port vers la SRAM0 de la Nanoboard	SRAM_DAUGHTER0	FPGA DB Common Port-Plugin.IntLib
Port vers la SRAM0 de la Nanoboard	SRAM_DAUGHTER1	FPGA DB Common Port-Plugin.IntLib
Fonction logique inverseuse	INV	FPGA Generic.IntLib

Pour placer un nouveau composant :

⇒ Cliquez sur **Librairies** sur le bord droit de l'écran
⇒ Sélectionnez la bibliothèque du composant
⇒ Sélectionnez le composants
⇒ Placez le composant

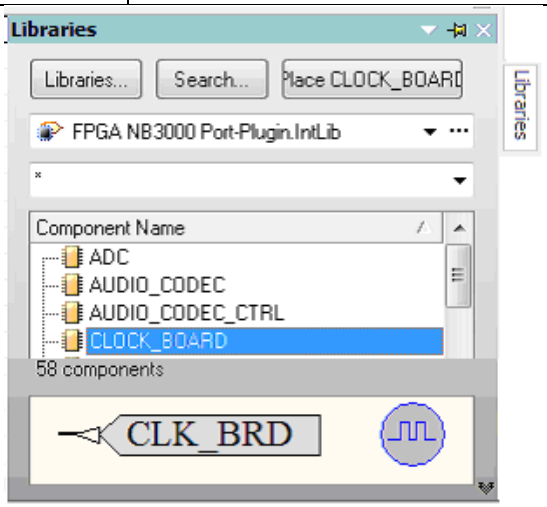
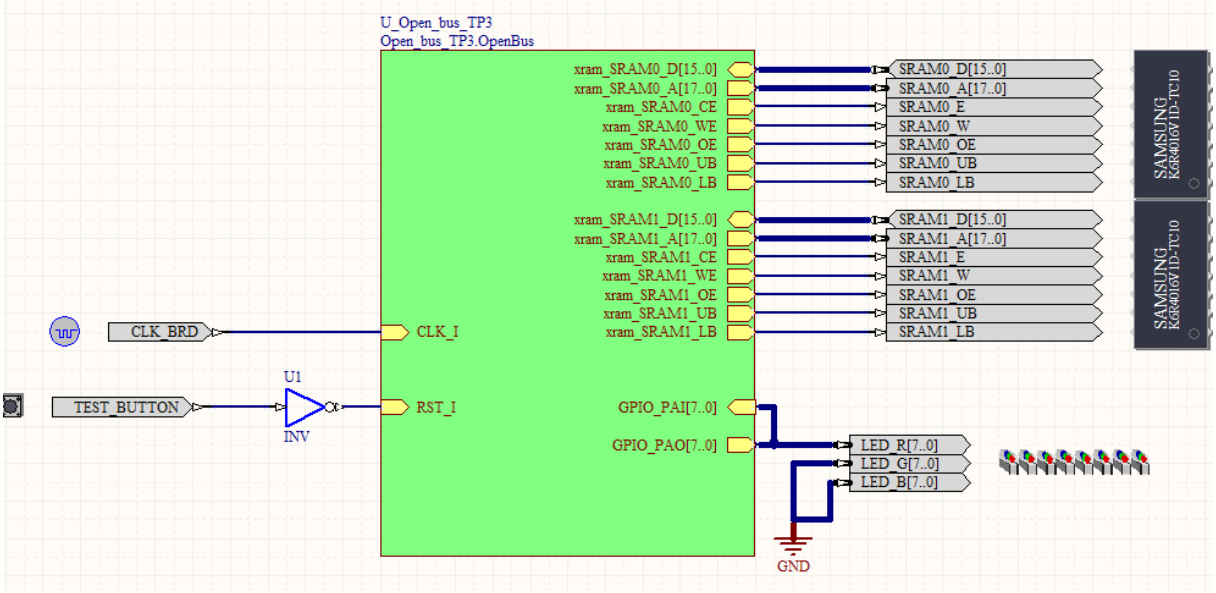
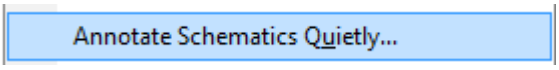


Schéma TOP complet à obtenir :



⇒ Pour attribuer une référence à chaque fonction :
⇒ **Menu** : TOOLS
⇒ **Commande** : Annotate Schématics Quietly

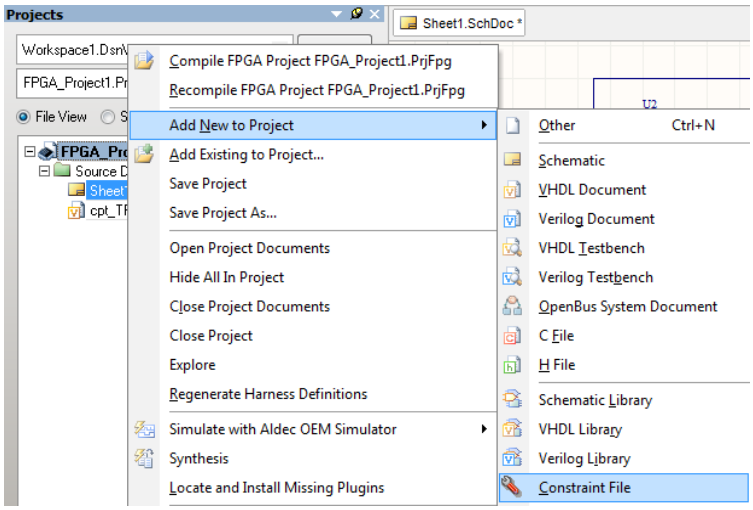


⇒ Lier les fils entre eux
⇒ A ce stade sauver et compiler le projet.

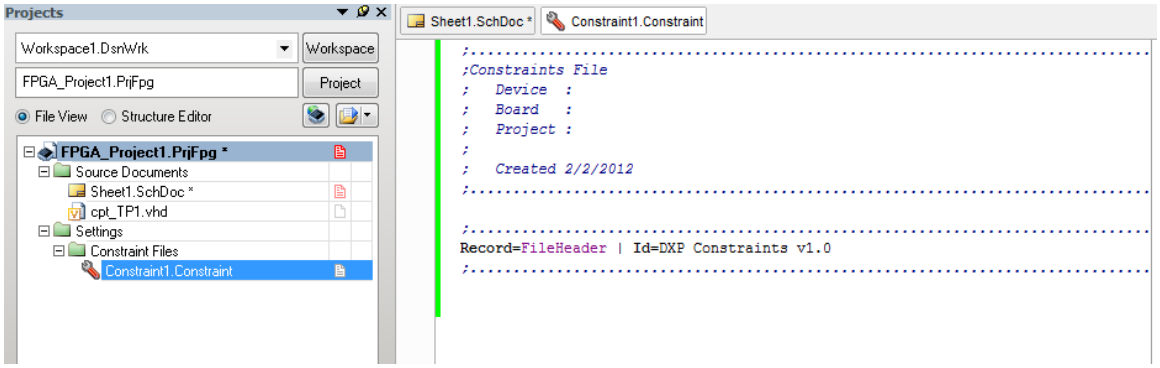
4 Définir les fichiers de contraintes.

4.1 Le fichier contraintes des horloges :

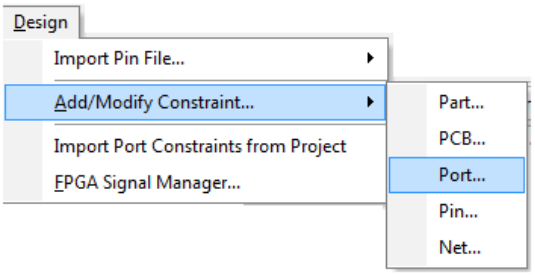
⇒ Ajouter un nouveau fichier contrainte au projet : cliquer avec bouton de droite sur le projet.

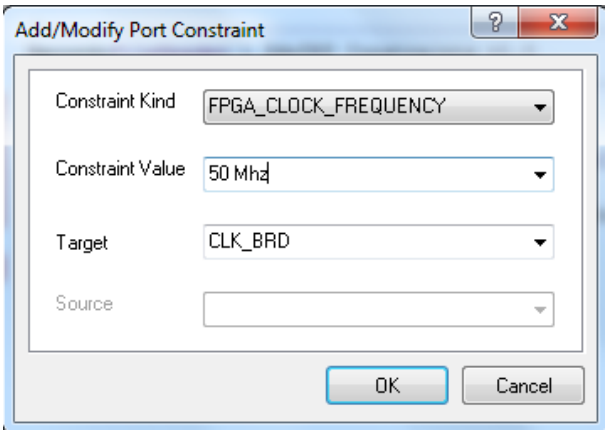


⇒ Il apparaît un fichier texte contrainte à compléter :



⇒ Pour compléter ce fichier contrainte nous allons utiliser des fonctions de génération de code intégrées à ALTIUM :

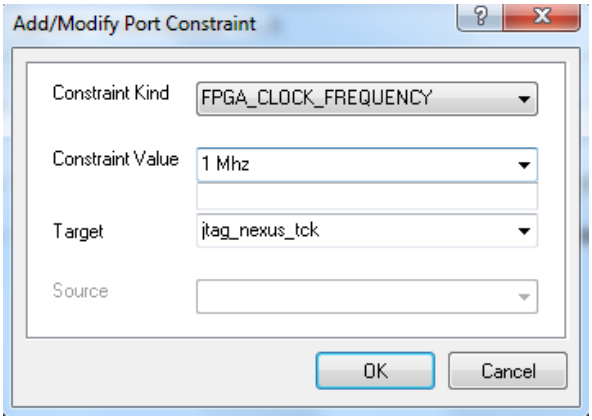




⇒ D'où la ligne jointe au fichier de contraintes :

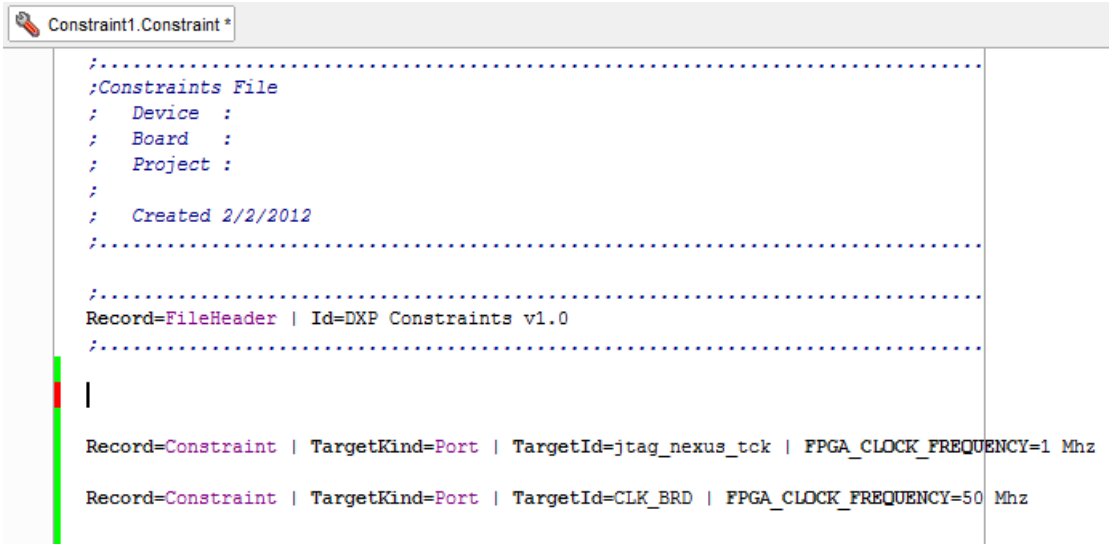
```
Record=Constraint | TargetKind=Port | TargetId=CLK_BRD | FPGA_CLOCK_FREQUENCY=50 Mhz
```

⇒ Faire de même pour les contraintes imposées à l'horloge du JTAG :




```
Record=Constraint | TargetKind=Port | TargetId=jtag_nexus_tck | FPGA_CLOCK_FREQUENCY=1 Mhz
```

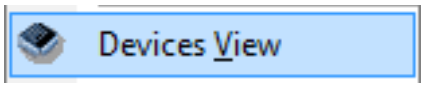
⇒ Vous devez maintenant obtenir le fichier ci-dessous. Sauvegardez ce fichier.



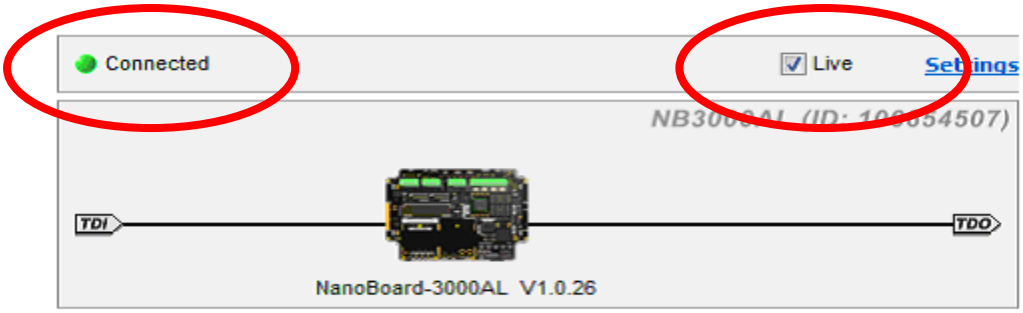
4.2 Appel du fichier contrainte liant les broches du FPGA aux périphériques implantés sur la Nanoboard 3000 :

Faites apparaître la fenêtre de visualisation des composants :

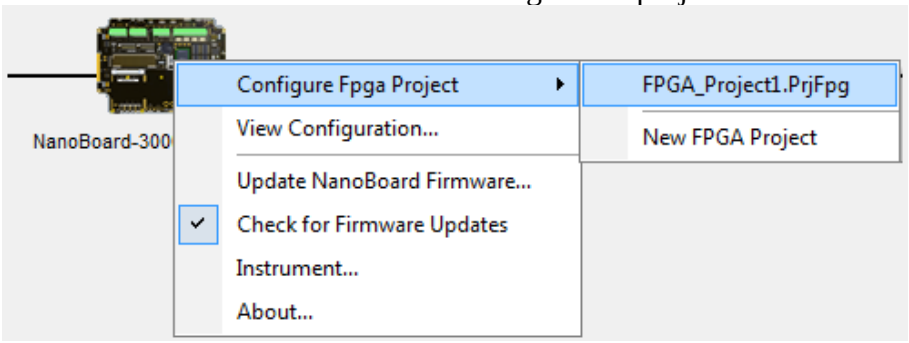
⇒ **Menu** : VIEW
⇒ **Commande** : Device VIEW



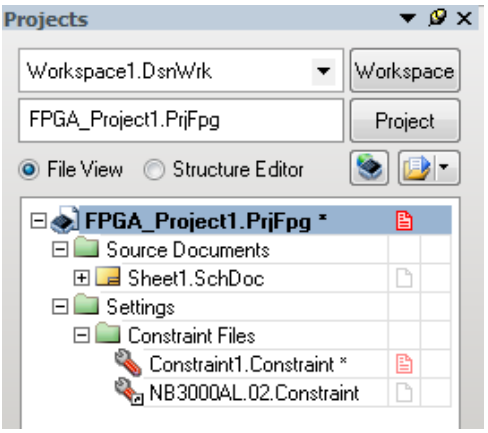
⇒ Assurez-vous que la Nanoboard est bien connectée à votre PC :



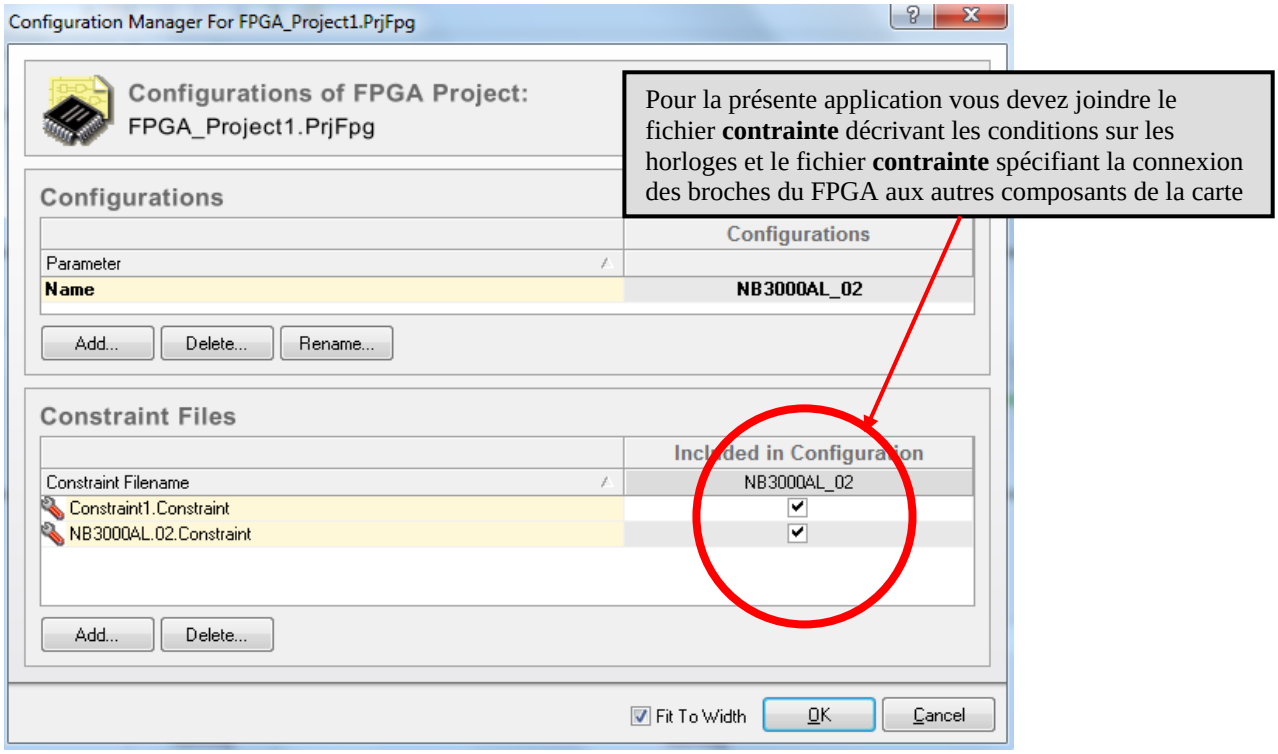
⇒ Cliquez droit sur l'icône de la Nanoboard et configurez le projet FPGA :



⇒ A l'issue de cette étape un deuxième fichier contrainte est lié à votre projet :



⇒ La fenêtre ci-dessous vous invite à spécifier les fichiers contraintes que vous voulez utiliser pour votre prochaine phase de compilation/synthèse/programmation.

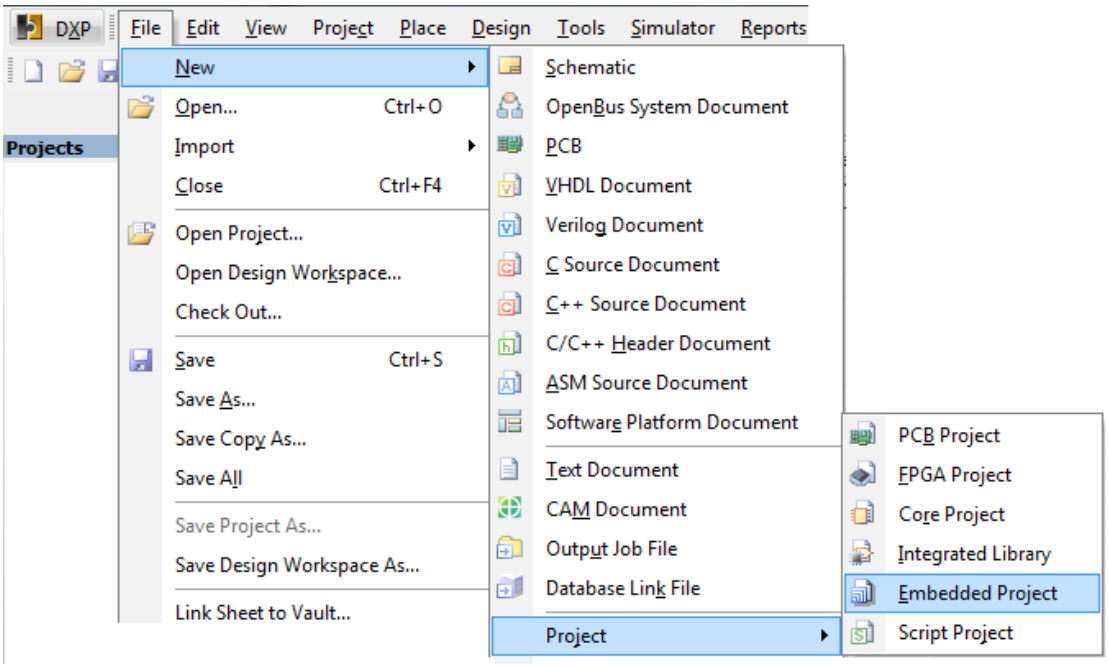


5 Création du projet embarqué.

Programmation du code en C : SOFWARE DESIGN FLOW.

5.1 Ajouter un projet embarqué à votre environnement

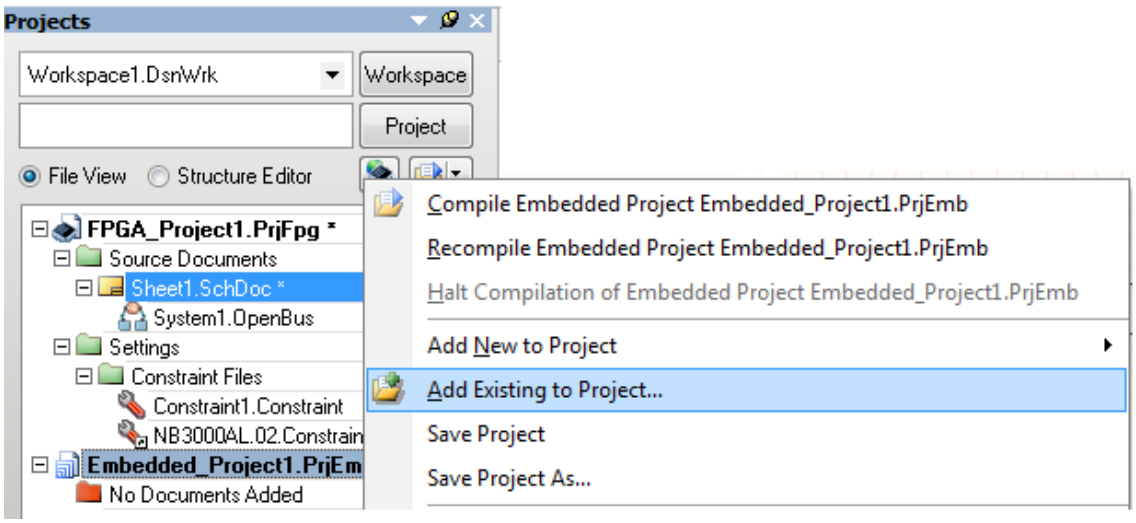
Créer un nouveau projet : Menu ⇒ **File** ⇒ **New** ⇒ **Embedded Projet**.
Un projet nommé « **Embedded_Projet.PrjEmb** » apparaît dans l'onglet **Projects**.



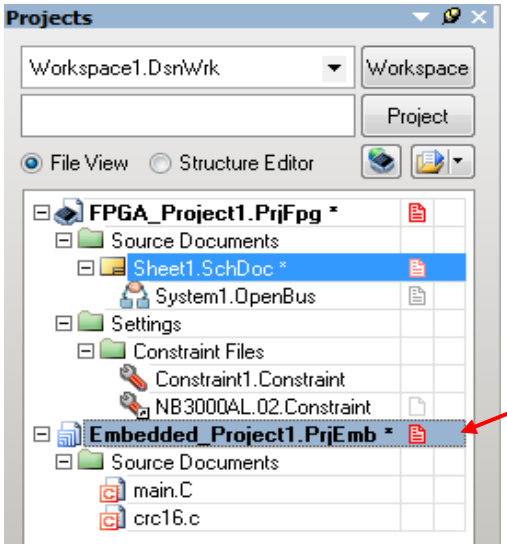
⇒ Clic droit sur le nom du projet sur le projet **« Embedded »**
⇒ Commande **Save Projet as « Embedded_TP4.PrjFpg »** dans le répertoire de travail.

5.2 A ce projet joignez les fonctions C qui seront exécutées par le processeur TASK 3000 :

⇒ Clic droit pour sur **« Embedded_TP4.PrjFpg »** ,
⇒ Choisir la commande ⇒ **Add Existing to project** ⇒ sélectionner les fichiers C.



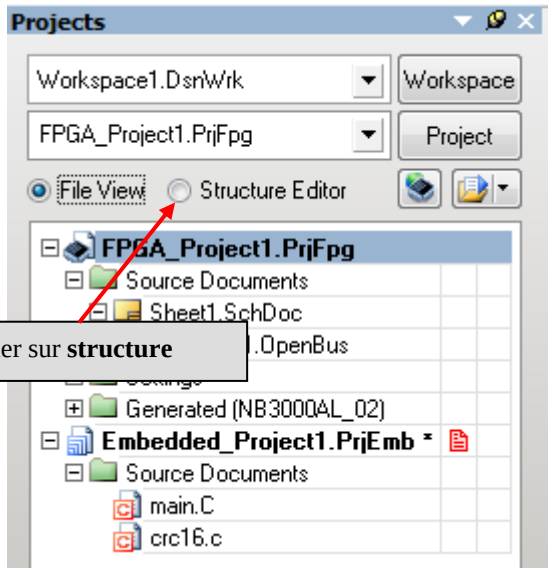
⇒ A joindre : les fonctions **« main.c »** et **« crc16.c »**



A ce stade le projet **FPGA_Project1.PrjFpg** décrivant la programmation du processeur TASK 30000 et le projet embarqué en langage C **Embedded_Project1.PrjEmb** sont indépendants.

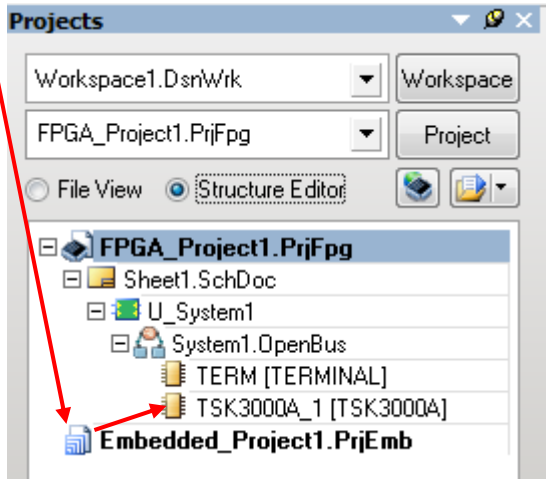
⇒ Sauvegarder le projet embarqué :
⇒ Clic droit sur **« Embedded_TP3.PrjFpg »**
⇒ **Save Project**.

5.3 Intégration du projet embarqué sous le projet FPGA :

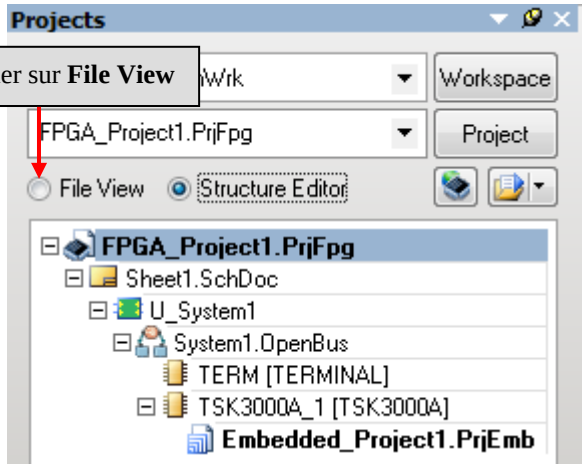


1 : Cliquer sur **structure**

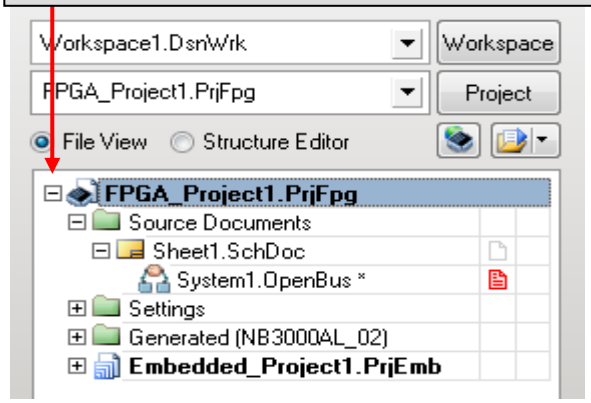
2 : Faites glisser le projet **Embedded_Project1.PrjEmmb** sous le processeur TSK3000A



3 : Cliquer sur **File View**



4 : Le projet Embarqué est maintenant sous le projet FPGA



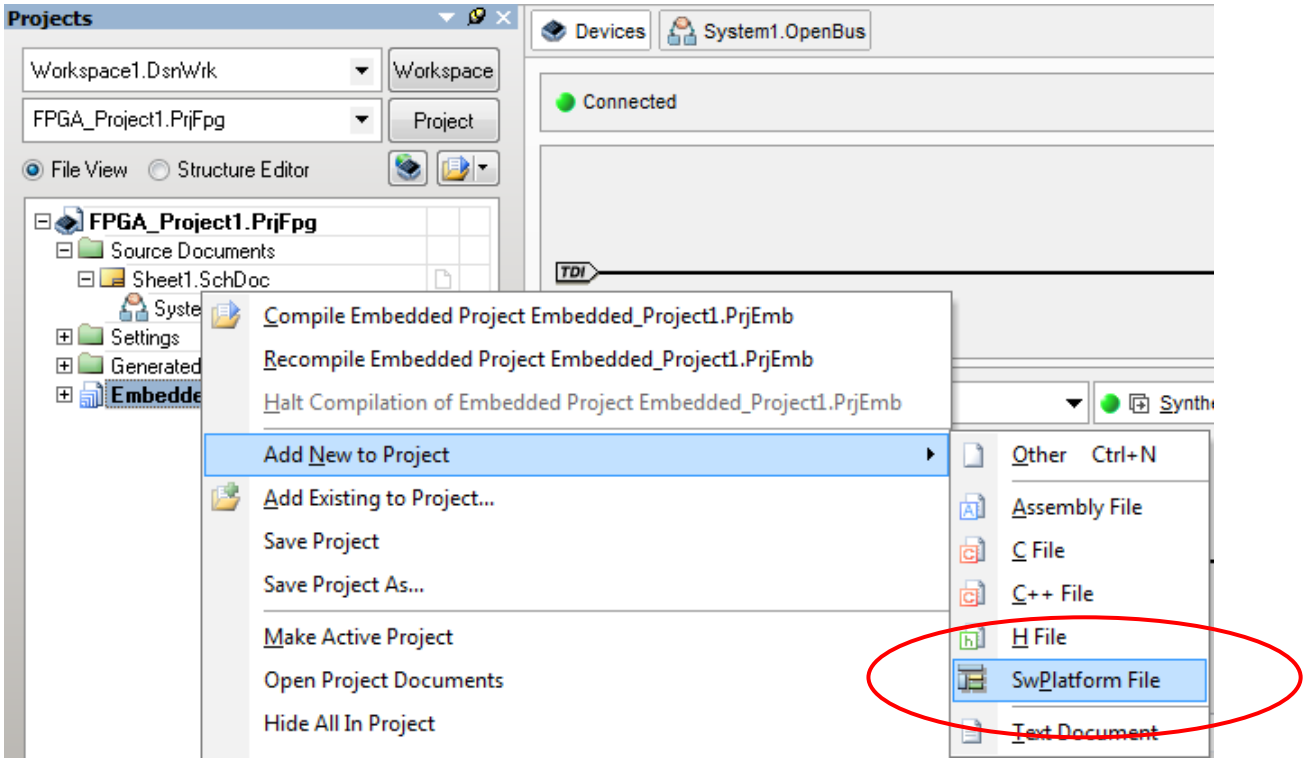
6 Construction du fichier « Softaware platform » : Mise en place des API .

API : **A**pplication **P**rogramming **I**nterface
Une **interface de programmation** est une fonction fournie par un programme informatique. Elle permet l'interaction entre le programme et les couches matérielles de bas niveaux.

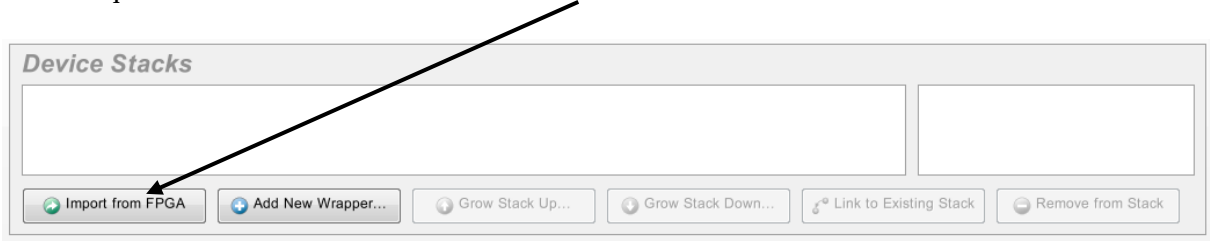
Un exemple, le paramétrage d'une liaison série : débit binaire (9600 bauds), nombre de bits par octet (8) , type de parité (aucune), nombre de bits de stop (1).

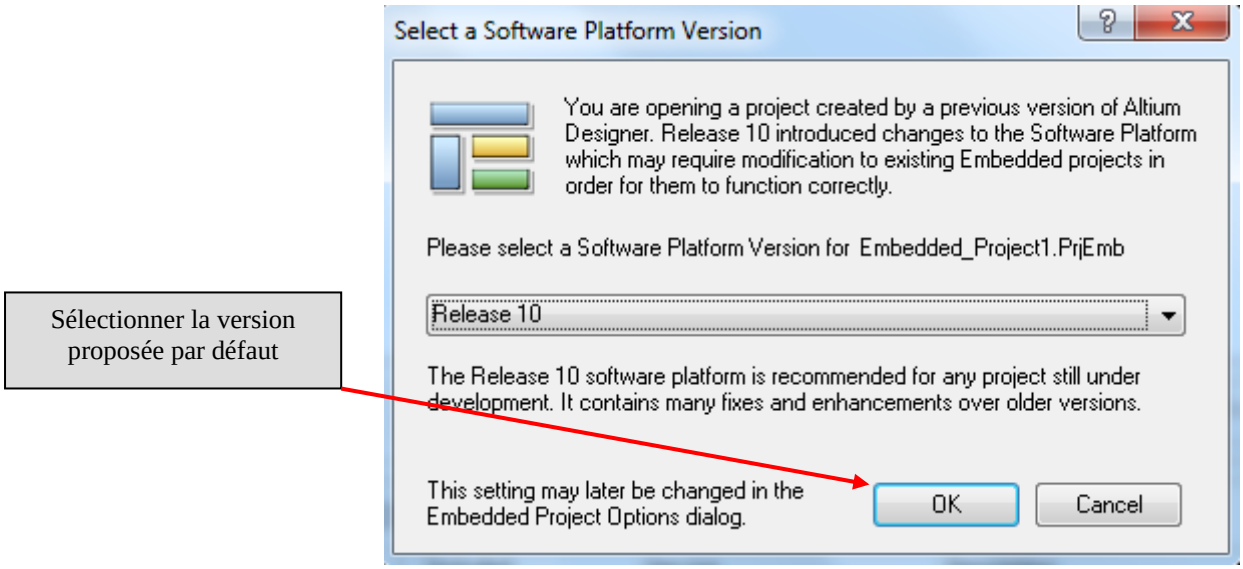
Pour plus d'information lire : Introduction to the Software platform.

⇒ clic droit sur le projet embarqué,
⇒ commande **Add New to Project**,
⇒ **SwPlatform File.**

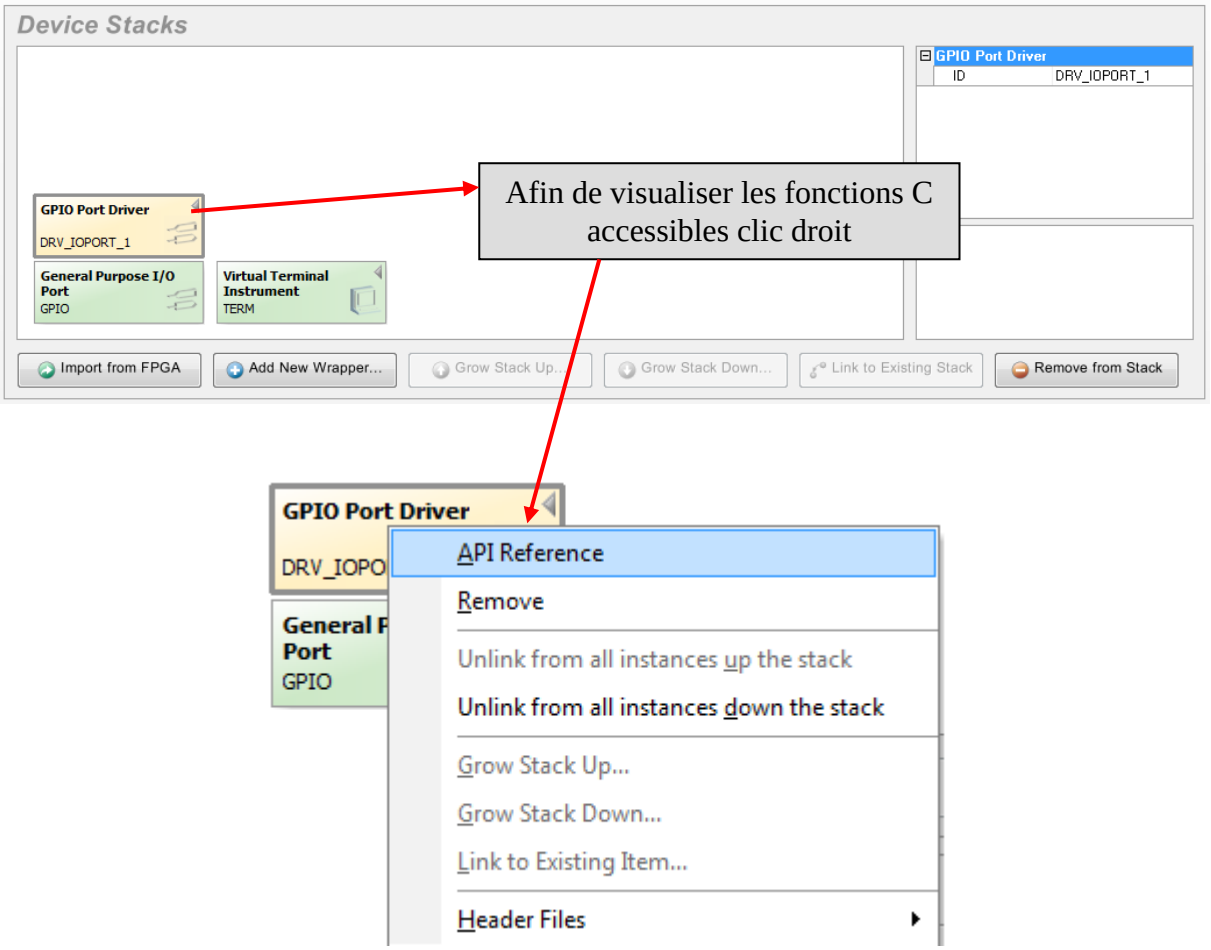


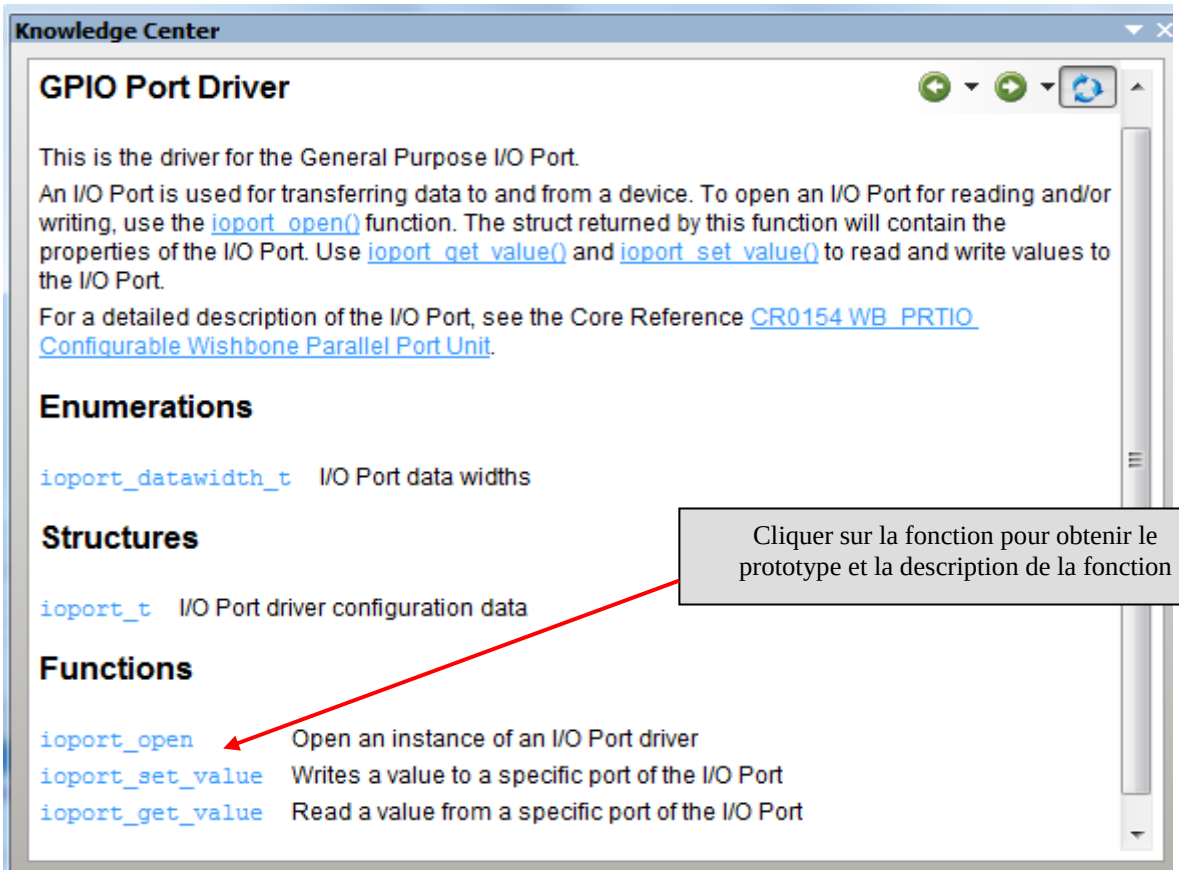
Afin d'effectuer une connexion bas niveau avec les modules décrits dans le fichier OPEN BUS cliquez sur **IMPORT FROM FPGA**





Pour obtenir la couche supérieure du port GPIO :
⇒ Sélectionnez l'icône vert **General Purpose I/O Port**
⇒ Cliquez sur **Grow Stack Up**

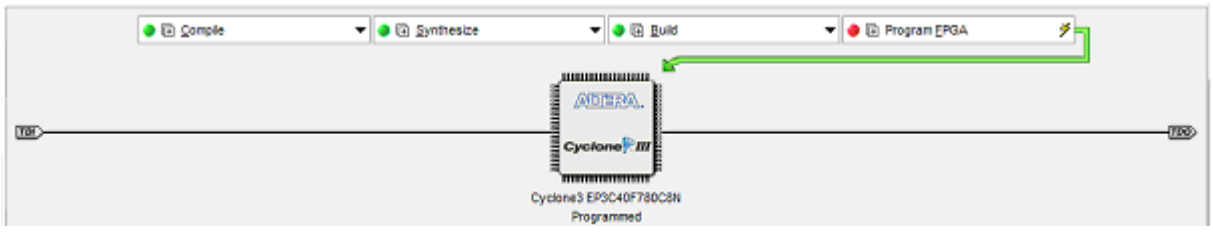




⇒ Développez la pile liée à l'instrument virtuel :

7 Compiler, Synthétiser, construire, Programmer le FPGA.

⇒ Cliquer sur Compile, cliquer sur Synthetize, cliquer sur Build.
⇒ Si une erreur apparaît vous devez la corriger en modifiant le fichier source identifié à partir du message d'erreur.



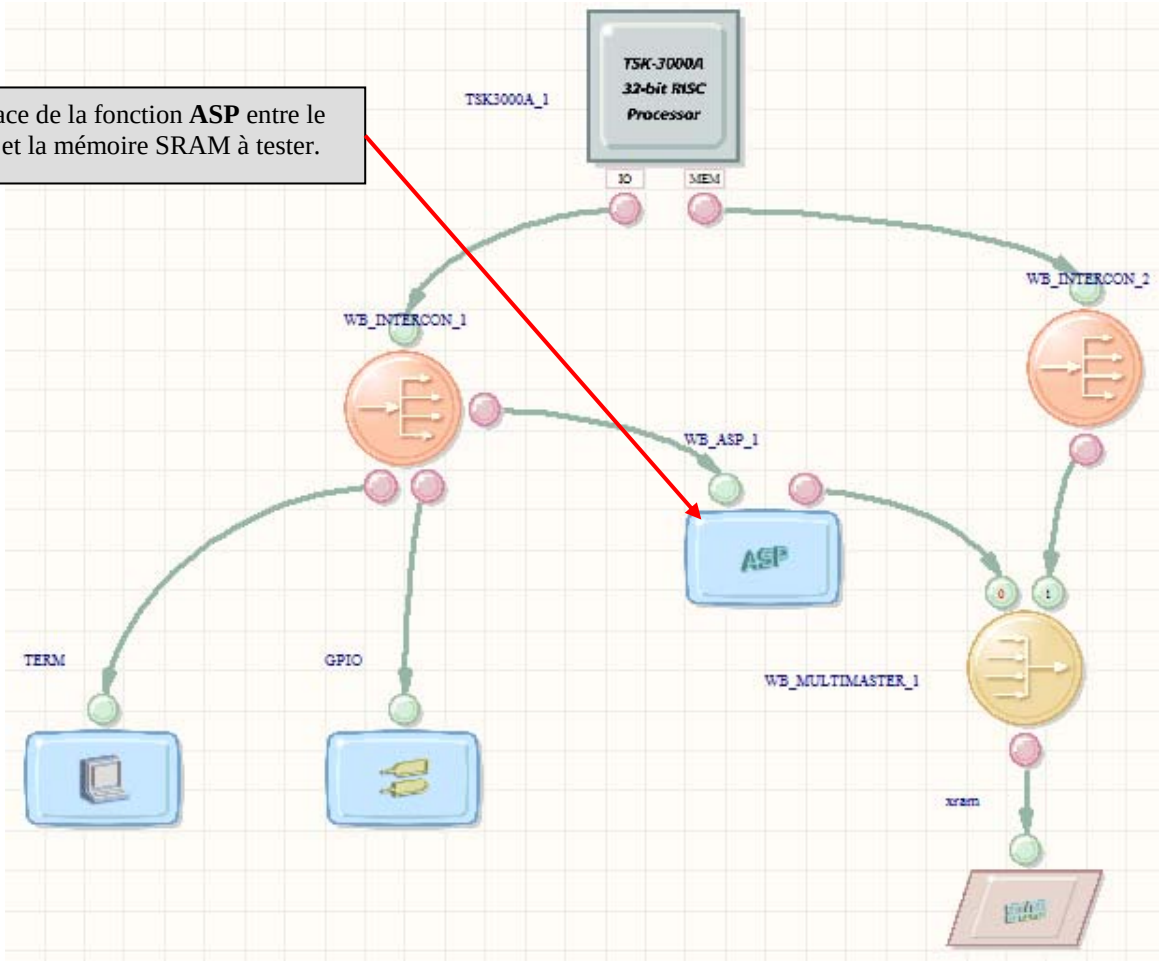
⇒ Appuyer sur la touche TEST/RESET pour lancer le programme.

8 Accélération matérielle : optimisation du code C en code H.

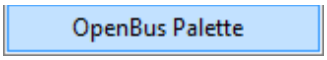
⇒ La durée d'exécution du programme peut être réduite : pour cela il nous faut remplacer la fonction **CRC16.c** par une structure équivalente décrite directement en **VHDL**.
⇒ Nous implanterons pour cela un IP **ASP** dans le fichier **OPEN BUS**.

8.1 Evolution du fichier Open Bus :

Mise en place de la fonction **ASP** entre le processeur et la mémoire SRAM à tester.



⇒ Ouvrir la palette **Open_Bus** (en bas à droite)

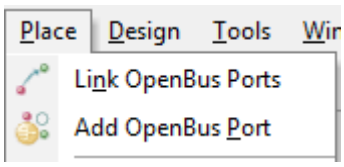


⇒ Placer dans le schéma les éléments **ASP** et **ARBITRER**
⇒ Faites évoluer votre schéma **Open_Bus** comme ci-dessus.

⇒ Rappel de la page 8 : comment dessiner un schéma **Open_Bus** ?

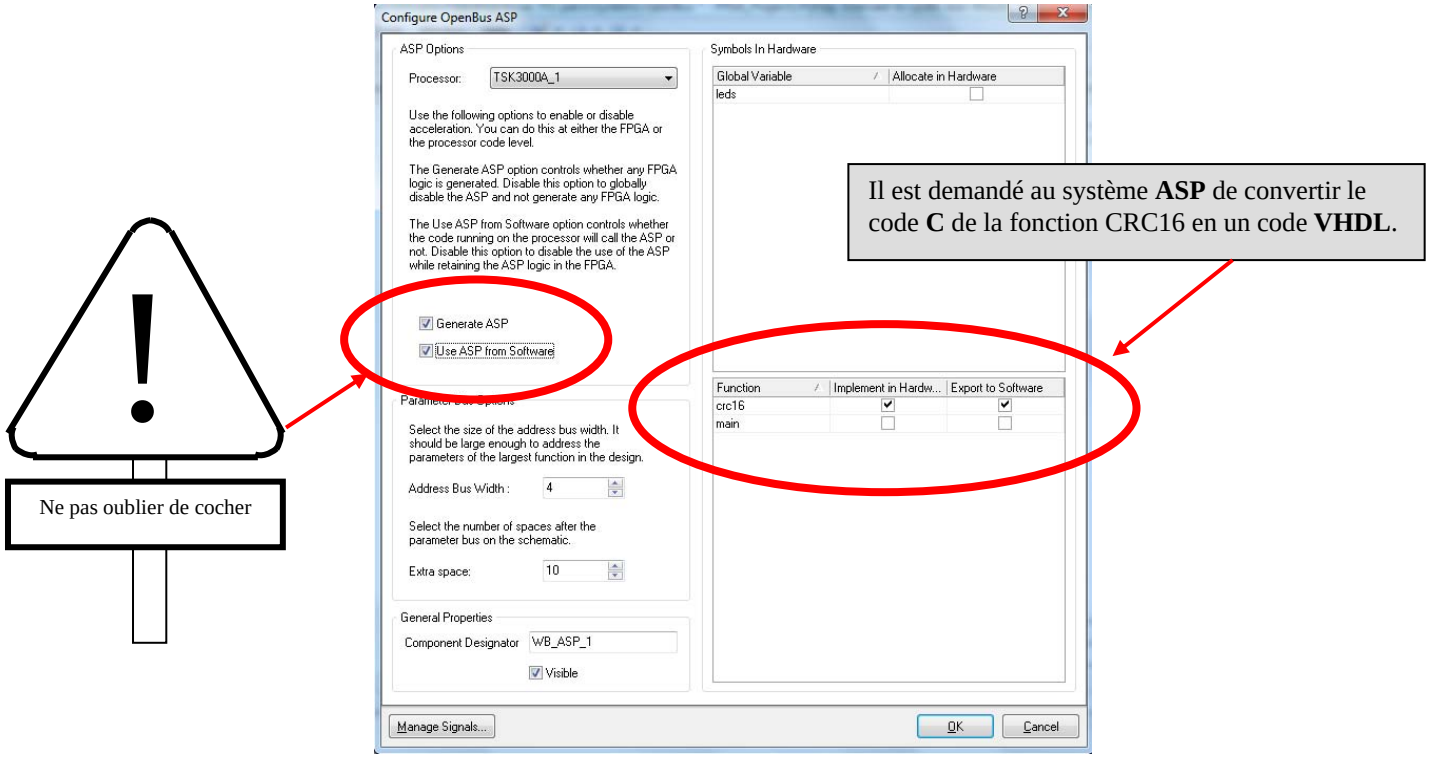
Pour dessiner les liens entre les structure **OPEN BUS** utiliser la fonction **LINK**

Pour rajouter les ports aux inter connexions utiliser la fonction **ADD open bus port**



8.2 Paramétrage de la fonction ASP :

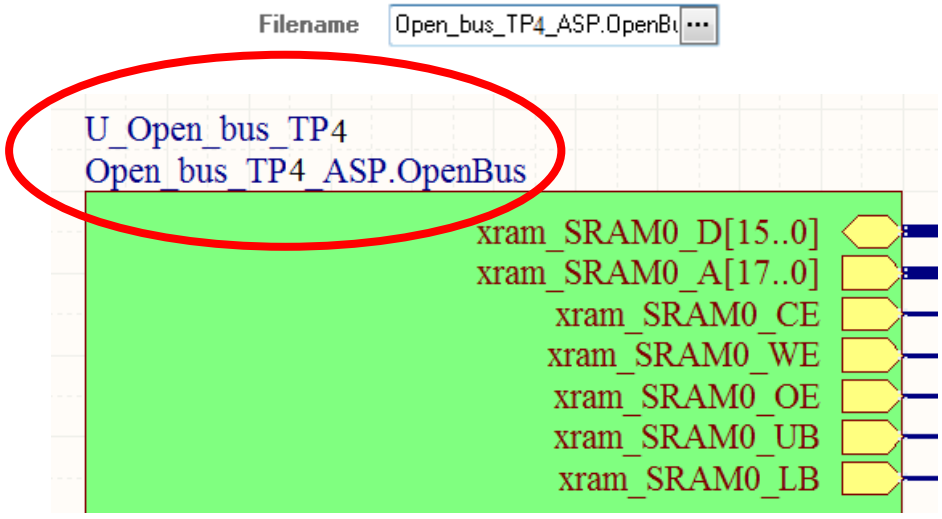
⇒ Double clic sur l'icône ASP
⇒ Sélectionner la fonction CRC16



⇒ Sauvegarder et renommer le fichier **OpenBus** : **« Open_bus_TP4_ASP.OpenBus ».**

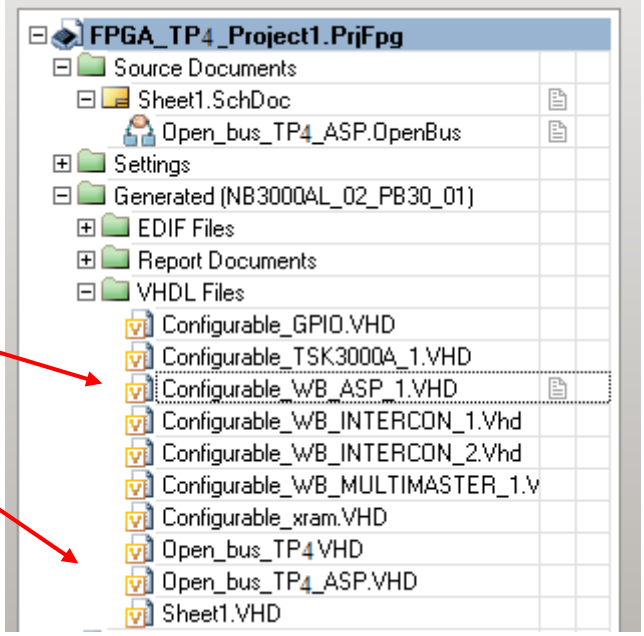
8.3 Lier le fichier TOP Sheet1.SchDoc au nouveau fichier «Open_bus_TP4_ASP.OpenBus».

⇒ Renommez le symbole du nom du nouveau fichier Open Bus :
⇒ Clic droit sur le corps du symbole.
⇒ Cliquez sur PROPERTIES
⇒ Modifiez le nom du symbole :



8.4 Re-compiler, re-synthétiser, re-construire, Re-programmer le FPGA.

Après compilation il est possible de visualiser les fichiers VHDL écrit à partir de la fonction CRC16.C



Comparez les vitesses d'exécution des deux solutions mises en oeuvre.

********** Fin du TP4 **********