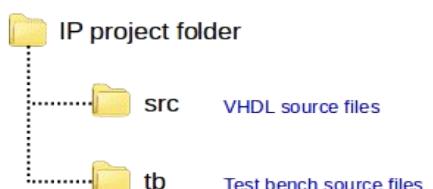
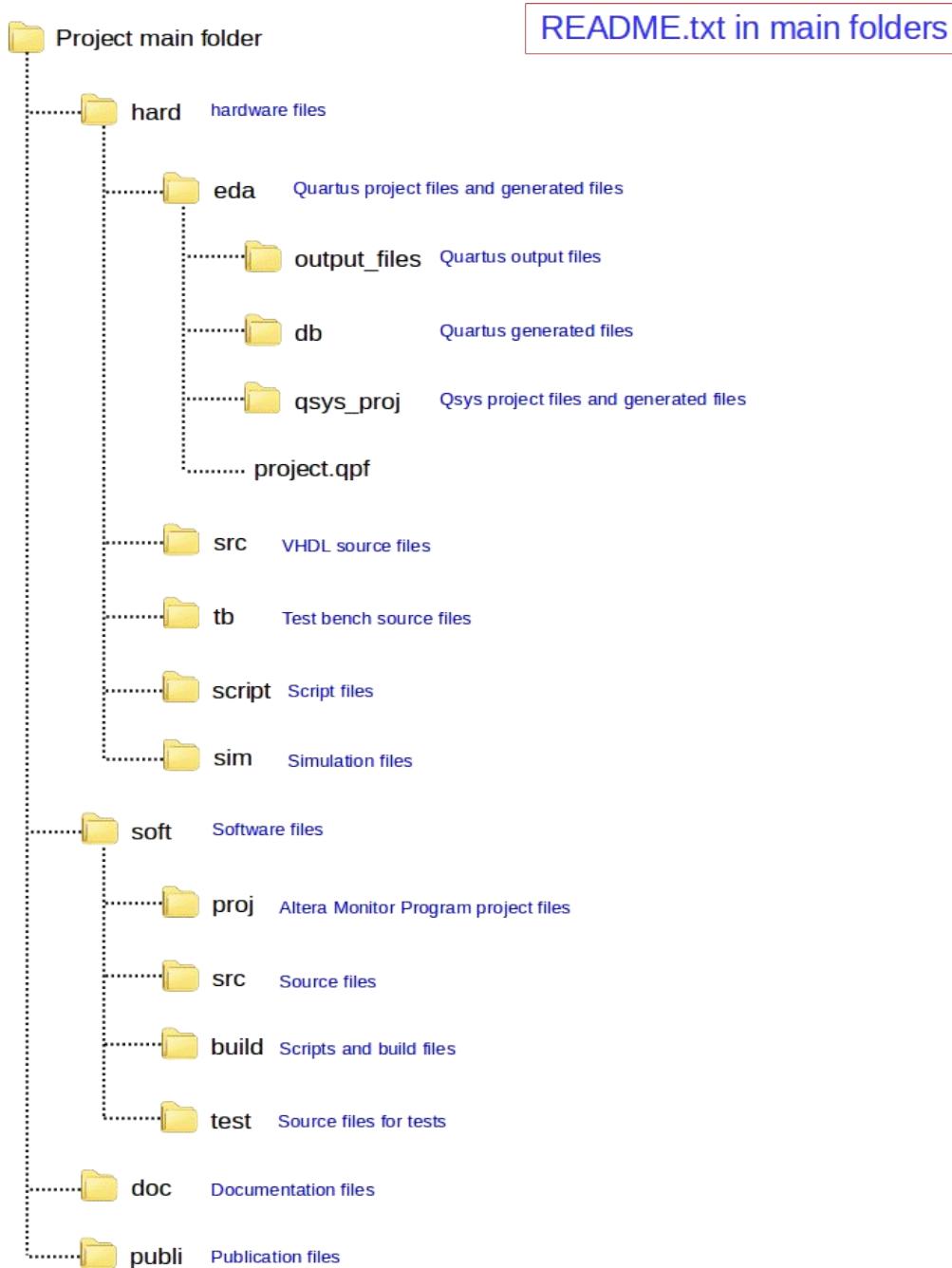


Tutoriel des outils de conception

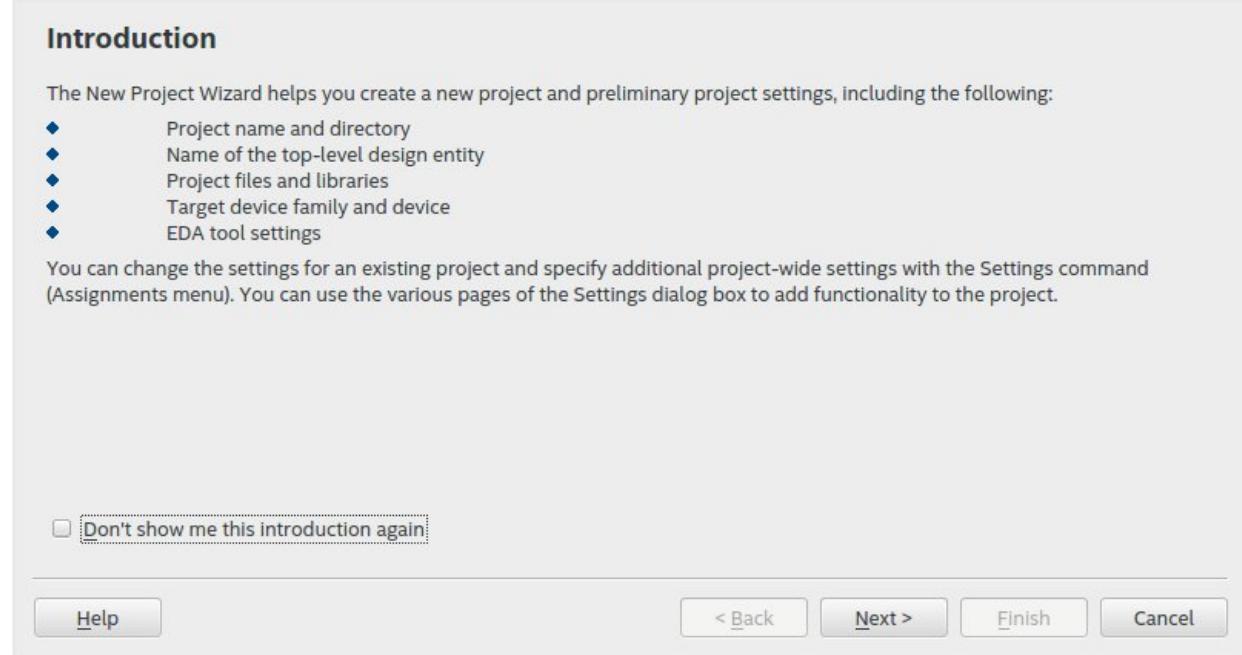
Structure du répertoire projet



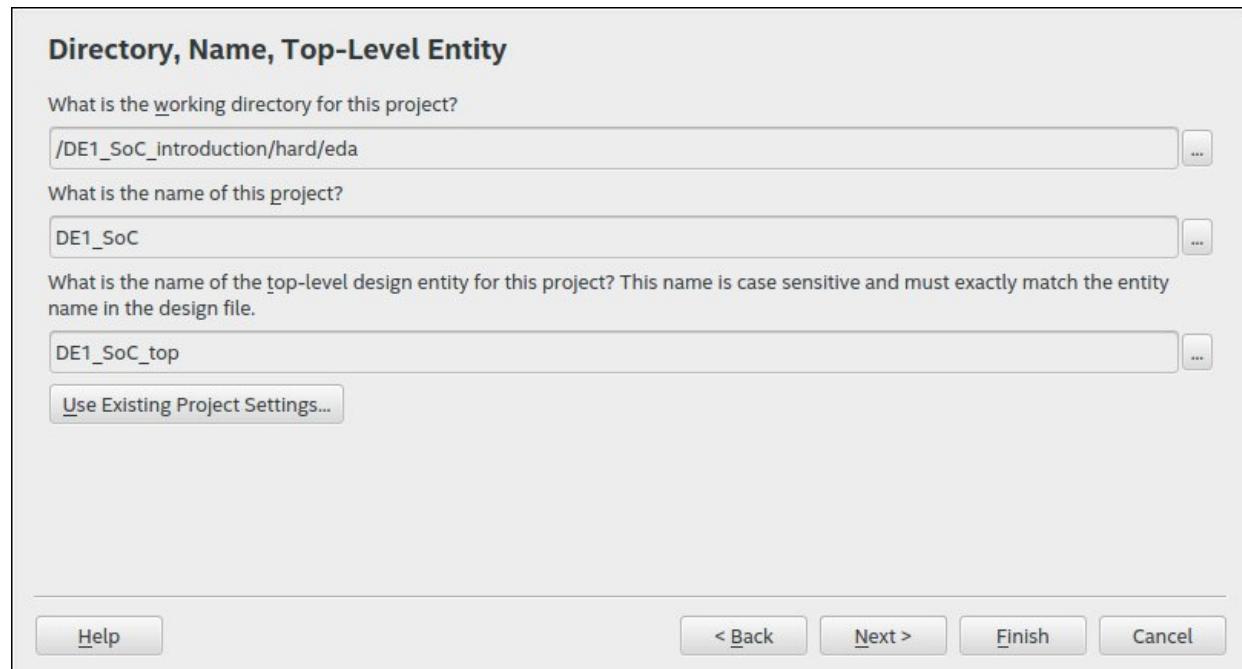
Quartus Prime

Création d'un projet

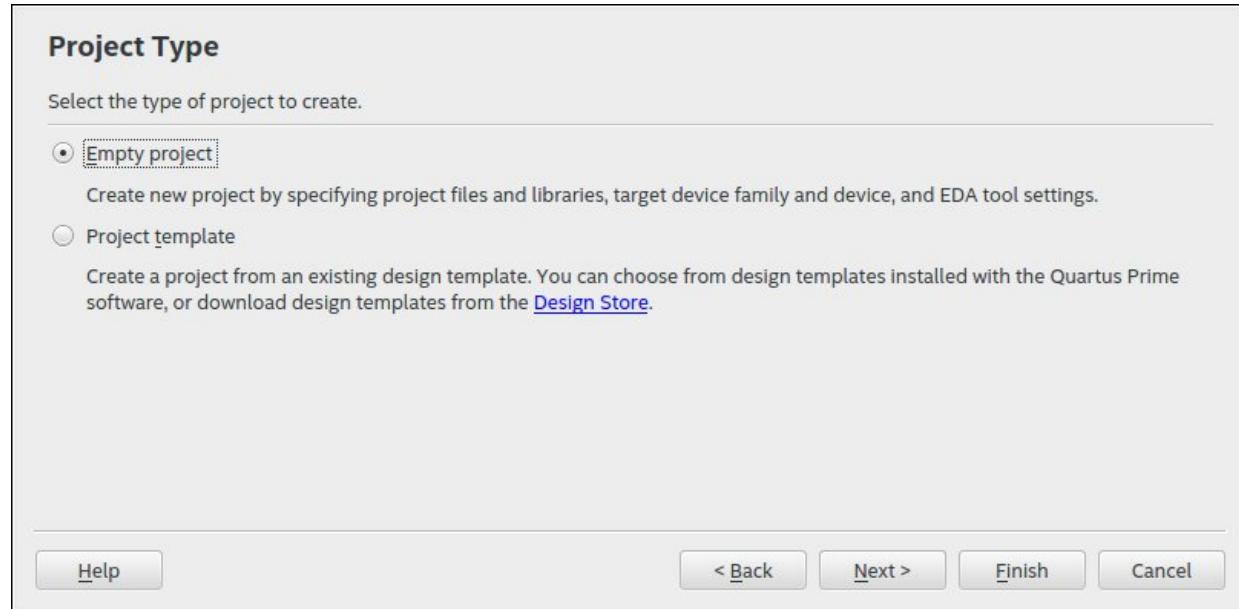
File → New Project Wizard... Next



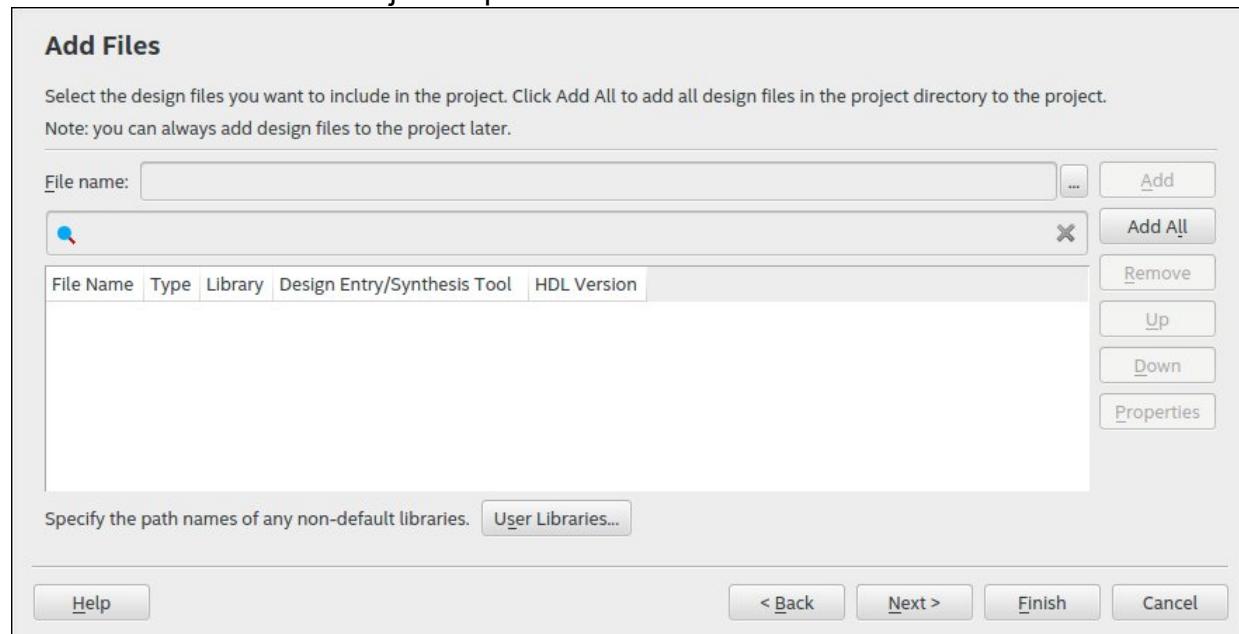
Choisir l'emplacement du projet, son nom, et le nom du top-level design. Next



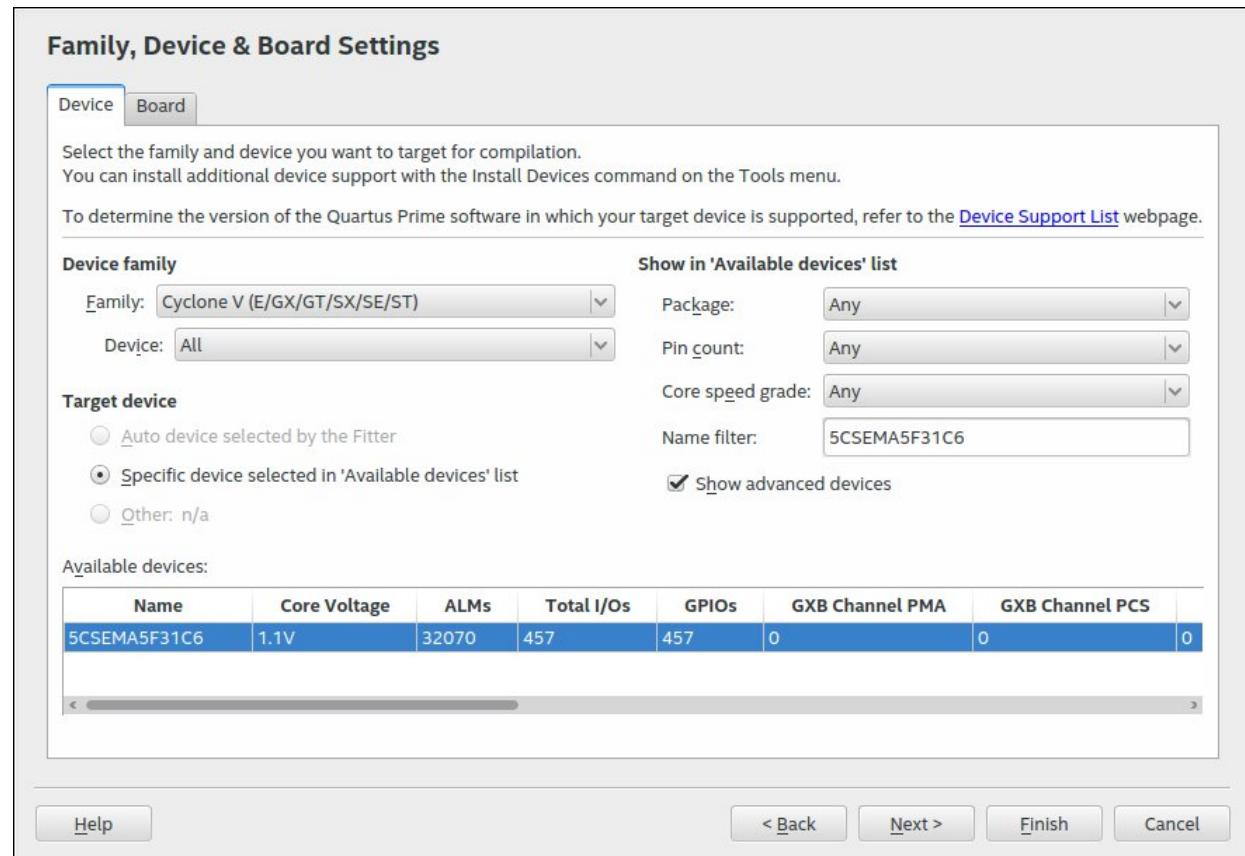
Choisir *Empty project. Next*



Les fichiers sources sont ajoutés plus tard. *Next*



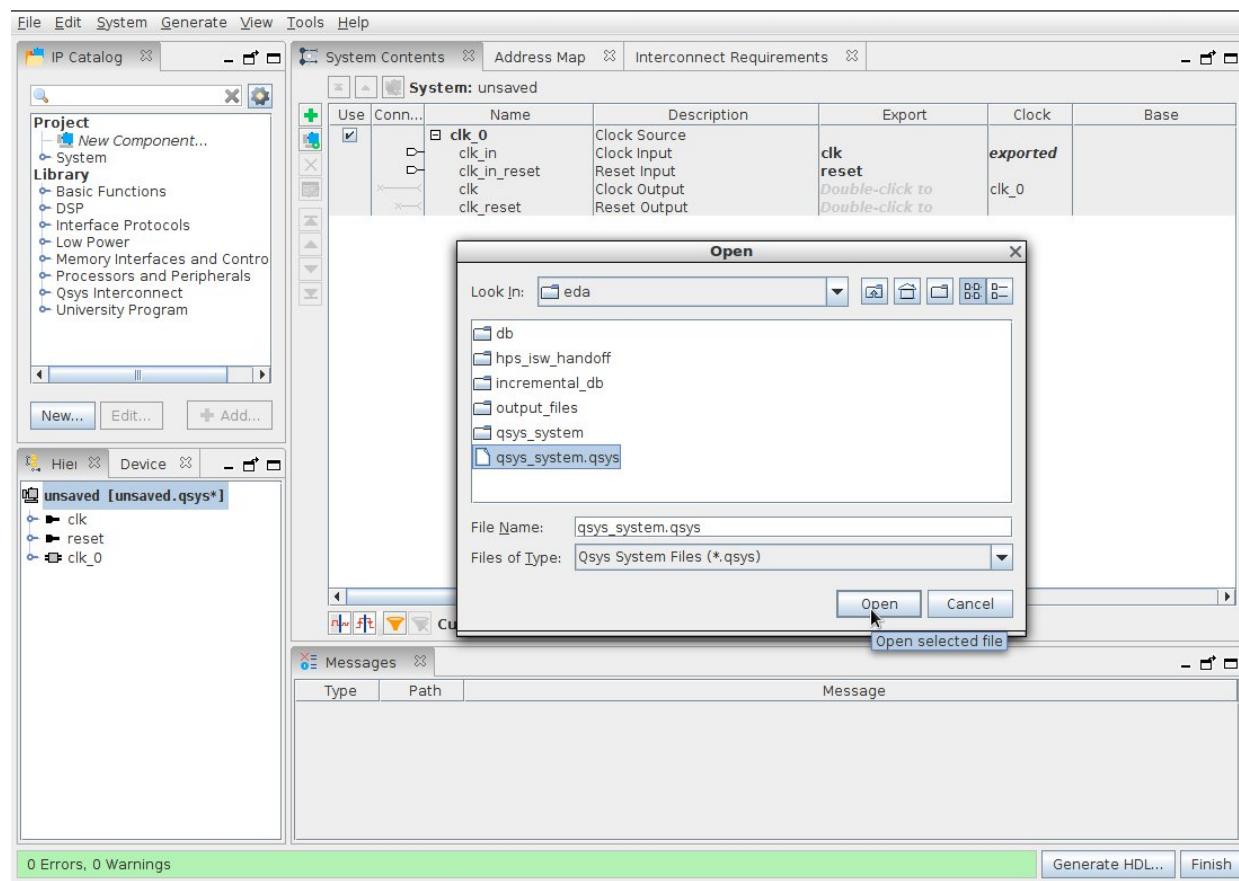
Entrer "5CSEMA5F31C6" dans le champ *Name filter* afin de sélectionner le composant présent sur la carte DE1-SoC. Cliquer sur *Finish*.



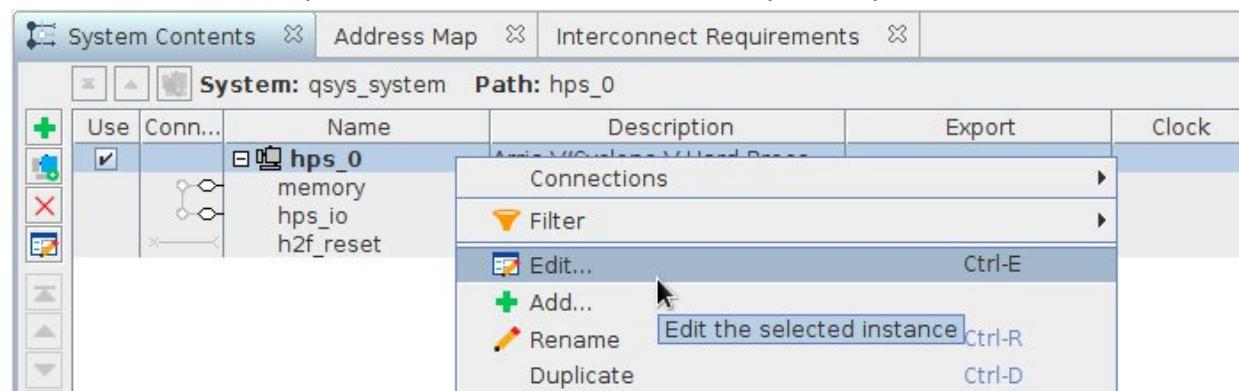
Platform Designer

Pour ouvrir *Platform Designer*, menu *Tools* → *Platform Designer* ou l'icône

Sélectionner le fichier « qsys_system.qsys » puis Open.



Pour modifier le composant HPS, clic droit sur le composant puis *Edit...*

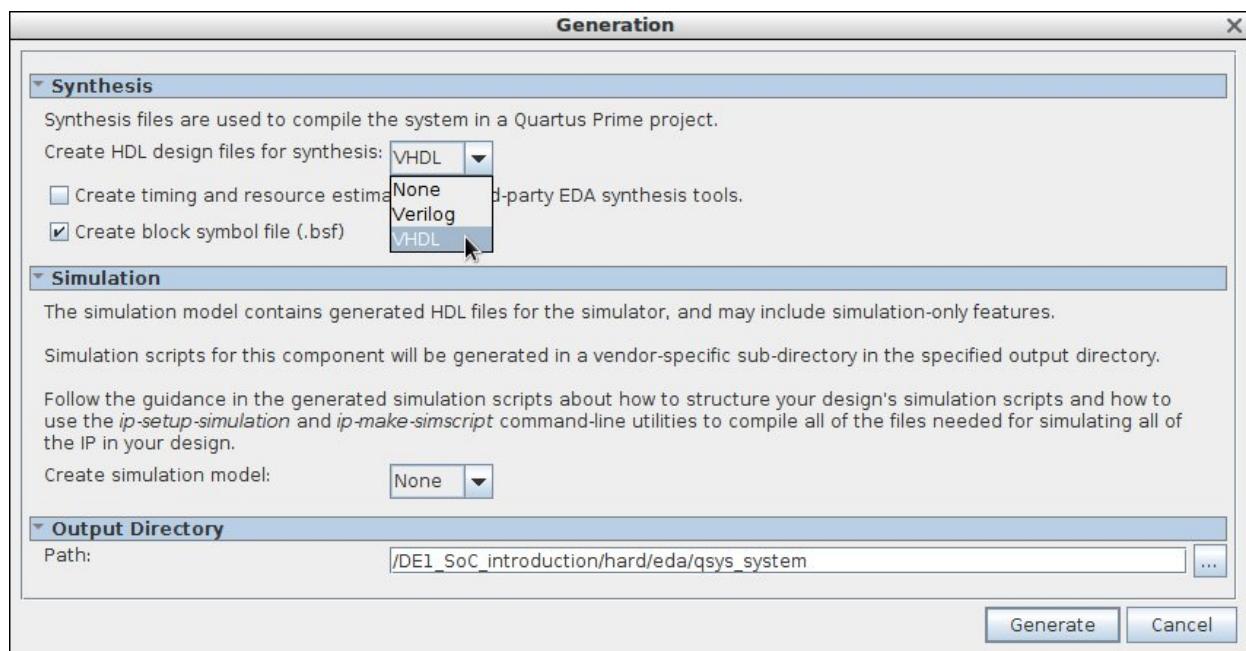


Pour activer ou désactiver les GPIOs du HPS, aller dans l'onglet *Peripheral Pins*, puis tout en bas où se trouve le tableau de configuration. Cliquer ensuite sur les GPIOs pour les activer ou les désactiver selon les besoins. Puis cliquer sur *Finish*.

Peripherals Mux Table		Unused	
TRACE mode:		N/A	
RGMII0_TX_CLK		EMAC0.TX_CLK (Set0)	GPIO00
RGMII0_RXD0		EMAC0.TXD0 (Set0)	GPIO01
RGMII0_RXD1		EMAC0.TXD1 (Set0)	GPIO02
RGMII0_RXD2		EMAC0.TXD2 (Set0)	GPIO03
RGMII0_RXD3		EMAC0.TXD3 (Set0)	GPIO04
RGMII0_RXD4		EMAC0.RXD0 (Set0)	GPIO05
RGMII0_MDC	I2C2.SDA (Set0)	EMAC0.MDC (Set0)	GPIO06
RGMII0_MDO	I2C2.SCL (Set0)	EMAC0.RXD1 (Set0)	GPIO07
RGMII0_RX_CTL		EMAC0.RXD2 (Set0)	GPIO08
RGMII0_RX_CTL		EMAC0.RXD3 (Set0)	GPIO09
RGMII0_RX_CLK		EMAC0.CLK (Set0)	GPIO10
RGMII0_RXD1		EMAC0.STP (Set0)	GPIO11
RGMII0_RXD2		EMAC0.RXD1 (Set0)	GPIO12
RGMII0_RXD3		EMAC0.RXD2 (Set0)	GPIO13
NAND_ALE	QSPI.SS1 (Set1)	EMAC1.TX_CLK (Set0)	NAND_ALE (Set0)
NAND_CE	USB1.D0 (Set0)	EMAC1.TXD0 (Set0)	NAND_CE (Set0)
NAND_CLE	USB1.D1 (Set0)	EMAC1.TXD1 (Set0)	NAND_CLE (Set0)
NAND_RE	USB1.D2 (Set0)	EMAC1.TXD2 (Set0)	NAND.RE (Set0)
NAND_RB	USB1.D3 (Set0)	EMAC1.TXD3 (Set0)	NAND.RB (Set0)
NAND_DQ0		EMAC1.RXD0 (Set0)	NAND.DQ0 (Set0)
NAND_DQ1	I2C3.SDA (Set0)	EMAC1.MDC (Set0)	NAND.DQ1 (Set0)
NAND_DQ2	I2C3.SCL (Set0)	EMAC1.D0 (Set0)	NAND.DQ2 (Set0)
NAND_DQ3	USB1.D4 (Set0)	EMAC1.PX_CTL (Set0)	NAND.DQ3 (Set0)
NAND_DQ4	USB1.D5 (Set0)	EMAC1.TX_CTL (Set0)	NAND.DQ4 (Set0)
NAND_DQ5	USB1.D6 (Set0)	EMAC1.RX_CLK (Set0)	NAND.DQ5 (Set0)
NAND_DQ6	USB1.D7 (Set0)	EMAC1.RXD1 (Set0)	NAND.DQ6 (Set0)
NAND_DQ7		EMAC1.RXD2 (Set0)	NAND.DQ7 (Set0)
NAND_WP	QSPI.SS2 (Set1)	EMAC1.RXD3 (Set0)	NAND_WP (Set0)
NAND_WE		QSPI.SS1 (Set0)	NAND_WE (Set0)
QSPI_I00	USB1.CLK (Set0)		QSPI.I00 (Set0)
QSPI_I01	USB1.STP (Set0)		QSPI.I01 (Set0)
QSPI_I02	USB1.DR (Set0)		QSPI.I02 (Set0)
QSPI_I03	USB1.NXT (Set0)		QSPI.I03 (Set0)
QSPI_S50			QSPI.S50 (Set0)
QSPI_CLK			QSPI.CLK (Set0)
QSPI_S51			QSPI.S51 (Set0)
SDMMC_CMD	USB0.D0 (Set0)	SDIO.CMD (Set0)	GPIO36
SDMMC_CSREN		SDIO.CS (Set0)	GPIO37
SDMMC_D0	USB0.D1 (Set0)	SDIO.D0 (Set0)	GPIO38
SDMMC_D1	USB0.D2 (Set0)	SDIO.D1 (Set0)	GPIO39
SDMMC_D4	USB0.D3 (Set0)	SDIO.D4 (Set0)	GPIO40
SDMMC_D5	USB0.D4 (Set0)	SDIO.D5 (Set0)	GPIO41
SDMMC_D6	USB0.D5 (Set0)	SDIO.D6 (Set0)	GPIO42
SDMMC_D7	USB0.D7 (Set0)	SDIO.D7 (Set0)	GPIO43
HPS_GPI044	USB0.CLK (Set0)		GPIO44
SDMMC_CCLK_OUT	USB0.STP (Set0)	SDIO.CLK (Set0)	GPIO45
SDMMC_D2	USB0.DIR (Set0)	SDIO.D2 (Set0)	GPIO46
SDMMC_D3	USB0.NXT (Set0)	SDIO.D3 (Set0)	GPIO47

Pour générer le code HDL, sélectionner le menu **Generate** → **Generate HDL...**

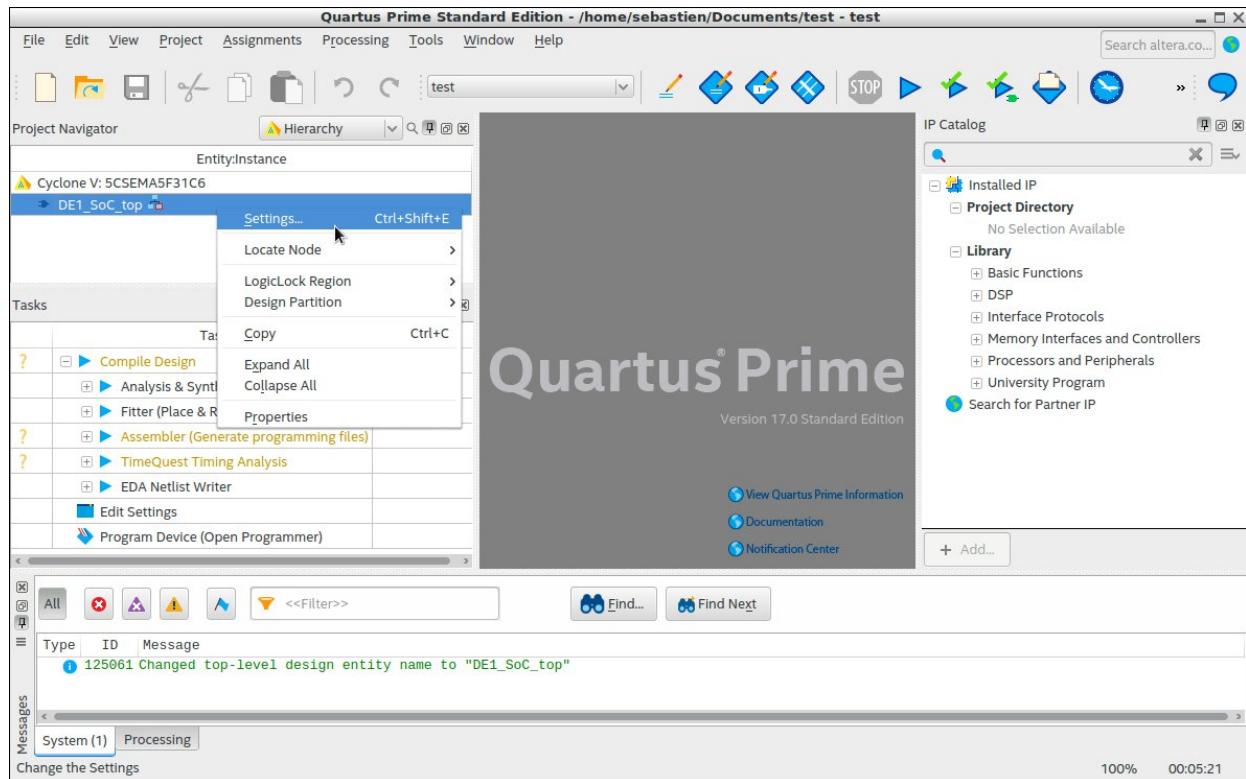
Sélectionner VHDL dans la fenêtre qui s'ouvre, laisser les autres champs par défaut puis cliquer sur le bouton *Generate*.



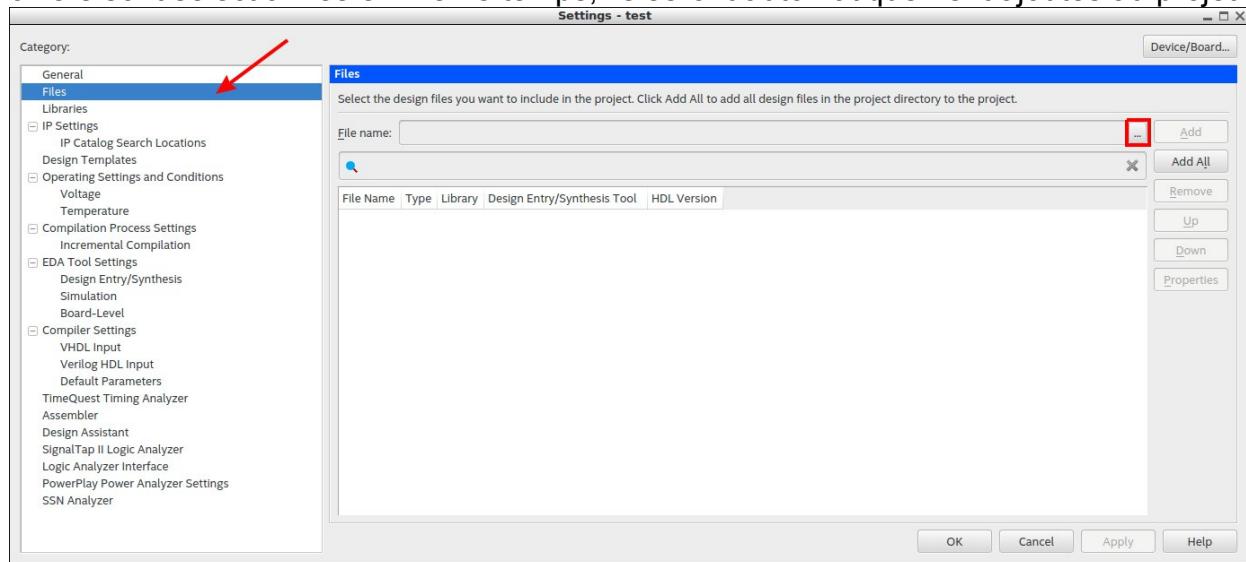
Fermer Platform Designer.

Ajout des sources dans le projet

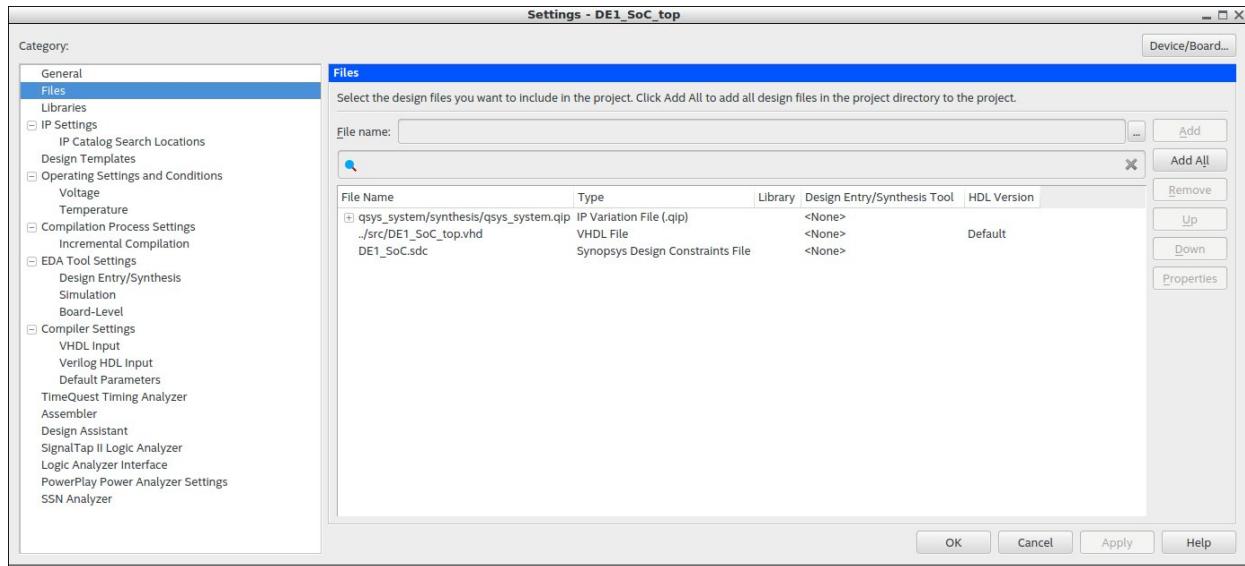
Clic droit sur le nom du projet et sélectionner *Settings...*



Selectionner l'onglet *Files* dans la colonne de gauche. Cliquer sur le bouton "..." dans la partie de droite pour ajouter des fichiers sources. Si un seul fichier est sélectionné, il faut ensuite appuyer sur le bouton *Add* pour qu'il soit ajouté, alors que si plusieurs fichiers sont sélectionnés en même temps, ils seront automatiquement ajoutés au projet.



Cliquer sur *OK* une fois que tous les fichiers sources sont présents dans le projet.



ARM Développement Studio

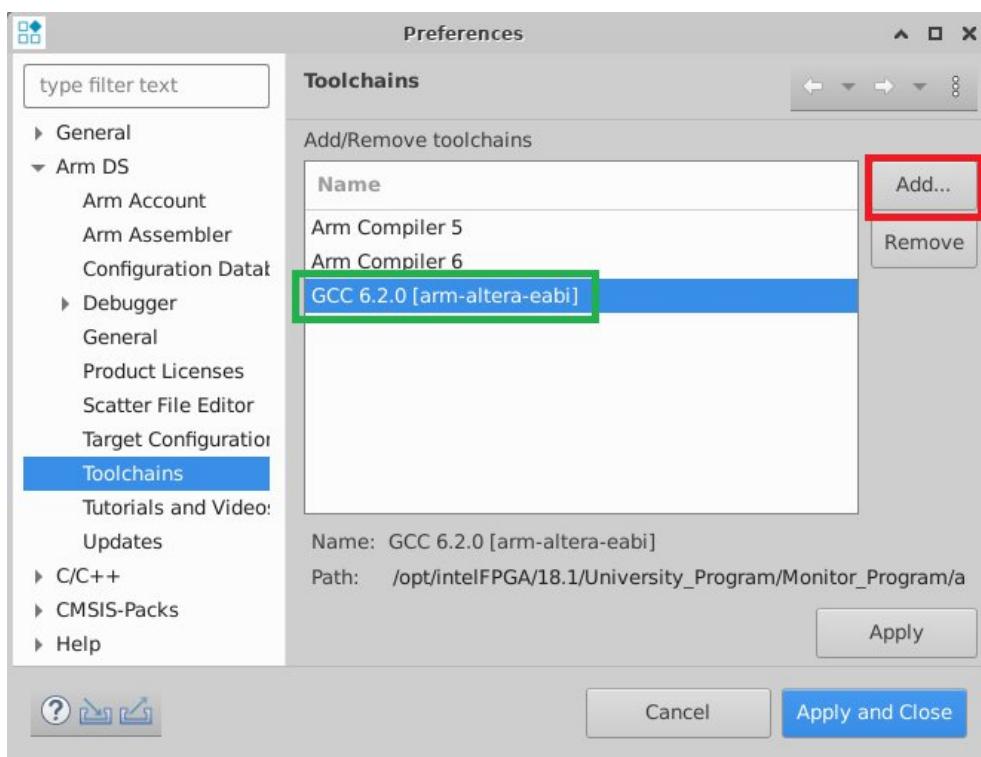
Ajout de la toolchain

Lors du premier démarrage du logiciel ARM-DS dans la VM, il faut ajouter la toolchain dans l'outil. Ensuite, il n'y aura plus besoin de faire cette étape.

Dans l'onglet **Window → Preferences**.

Choisissez dans le menu de gauche **ArmDS → Toolchains**.

Ensuite si la toolchain "**GCC 6.2.0 [arm-altera-eabi]**" n'est pas listé, il faut l'ajouter en cliquant sur **Add...**



Puis pour sélectionner le *Path* cliquez sur **Browse...** et aller chercher le dossier contenant tous les exécutables **arm-altera-eabi-*** et se trouvant dans :

/opt/intelFPGA/18.1/University_Program/Monitor_Program/arm_tools/baremetal/bin

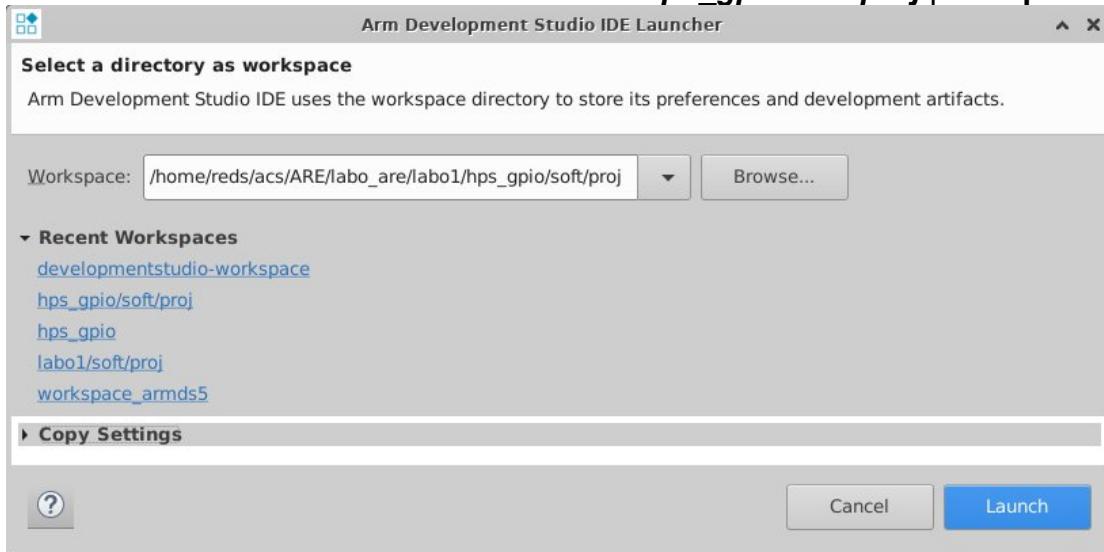
Puis **Open**, et ensuite **Next**.

La toolchain devrait être reconnue, laissez tout par défaut et cliquez sur **Finish**.

Changer le workspace

Dans l'onglet **File** → **Switch Workspace** → **Other...**

Cliquer sur **Browse** et sélectionner le dossier : **.../hps_gpio/soft/proj** puis **Open**

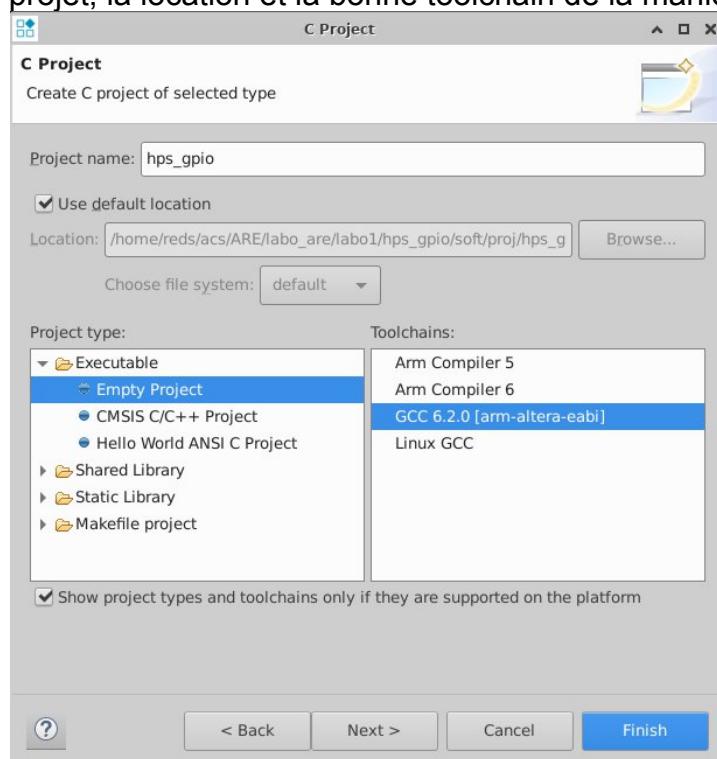


Cliquer ensuite sur **Launch**.

Le logiciel va redémarrer automatiquement avec le nouveau workspace.

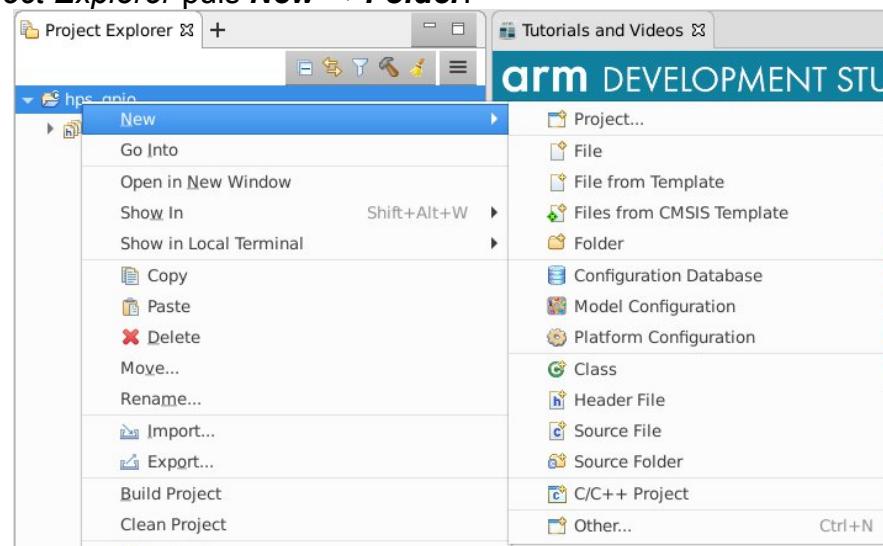
Création d'un projet

Dans l'onglet **File** → **New** → **Project...** choisissez **C/C++** → **C Project** puis **Next >**. Choisir le nom du projet, la location et la bonne toolchain de la manière suivante

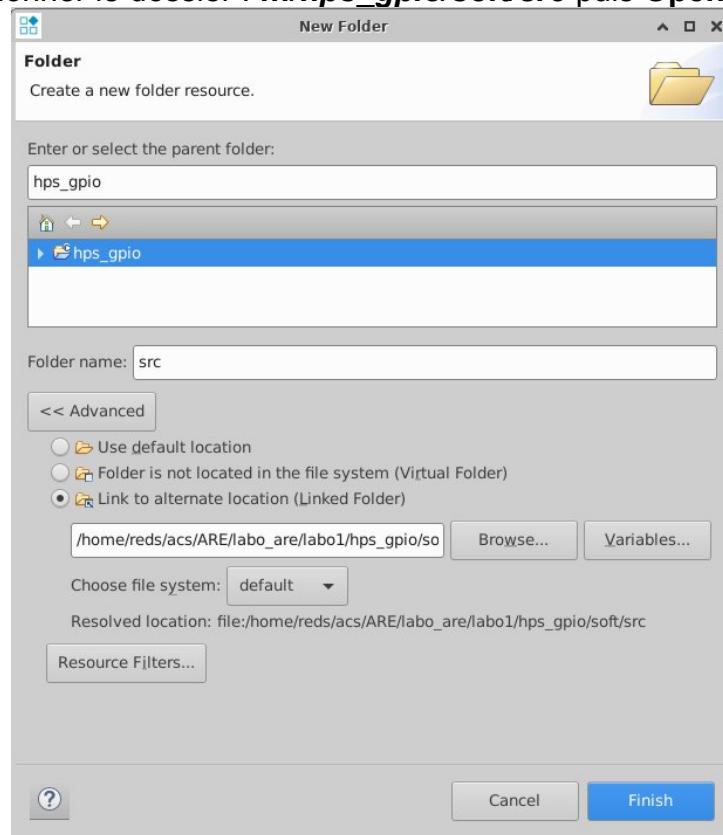


Vous pouvez ensuite tout laisser par défaut et cliquer sur **Next >** puis **Next >** et enfin **Finish**.

Pour ajouter le dossier source au projet, faites un clic droit sur le projet (hps_gpio) dans l'onglet *Project Explorer* puis **New → Folder**.

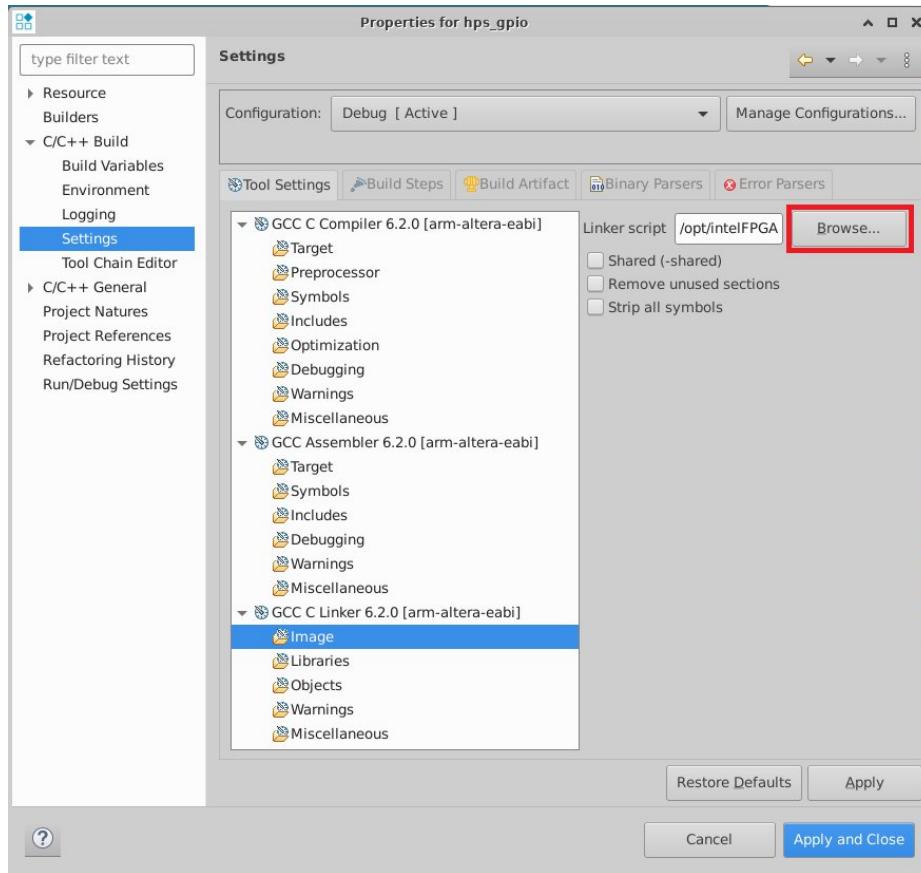


Dans la fenêtre qui s'ouvre, appuyer sur le bouton **Advanced>>**.
Ensuite sélectionner « **Link to alternate location (Linked Folder)** », cliquer sur **Browse** et sélectionner le dossier : **.../hps_gpio/soft/src** puis **Open** et **Finish**.



Les fichiers contenus dans le dossier src seront automatiquement ajouté au projet.

Pour compiler le projet, il est nécessaire d'ajouter le script de linkage lié au HPS cyclone V. Aller dans les propriétés du projet, clic droit sur le projet (hps_gpio) dans l'onglet *Project Explorer* puis **Properties**.



Dans **C/C++ Build** → **Settings**, puis dans l'onglet **Tool Settings** → **GCC C Linker** → **Image**. Cliquer sur **Browse** et aller chercher le fichier se trouvant dans le dossier d'installation d'Altera Monitor Program (attention, le copier-coller prend le retour à la ligne) :

`/opt/intelFPGA/18.1/University_Program/Monitor_Program/arm_tools/baremetal/arm-altera-eabi/lib/cycloneV-dk-ram-hosted.Id`

Puis **Open**, et ensuite **Apply**.

Pour ajouter la génération du code assembleur, dans la même fenêtre, sélectionner l'onglet **Tool Settings** → **GCC C Compiler** → **Miscellaneous**.

Compléter le champ Other flags avec : `-Wa,-adhlns="$@.lst"`

Puis **Apply and Close**.

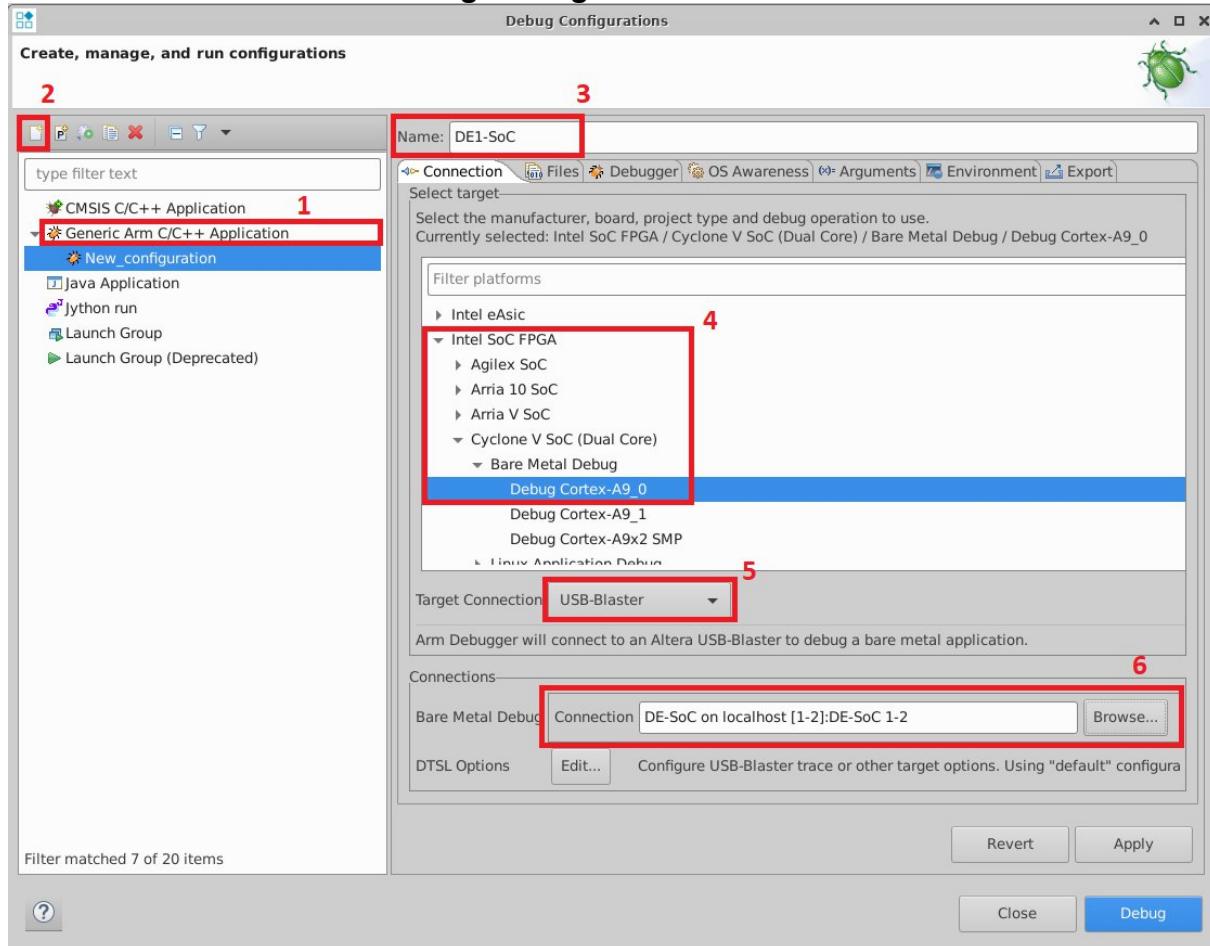
Vous pouvez maintenant compiler votre projet en cliquant sur le petit marteau en haut à gauche dans la partie **Project Explorer**, OU clic droit sur le projet (hps_gpio) dans l'onglet **Project Explorer** puis **Build Project**.

Après la compilation de votre projet, le ou les fichiers *.lst sont générés dans le répertoire Debug/src de votre projet. Ils correspondent à votre code assembleur de vos sources C.

Création d'une connexion de debug

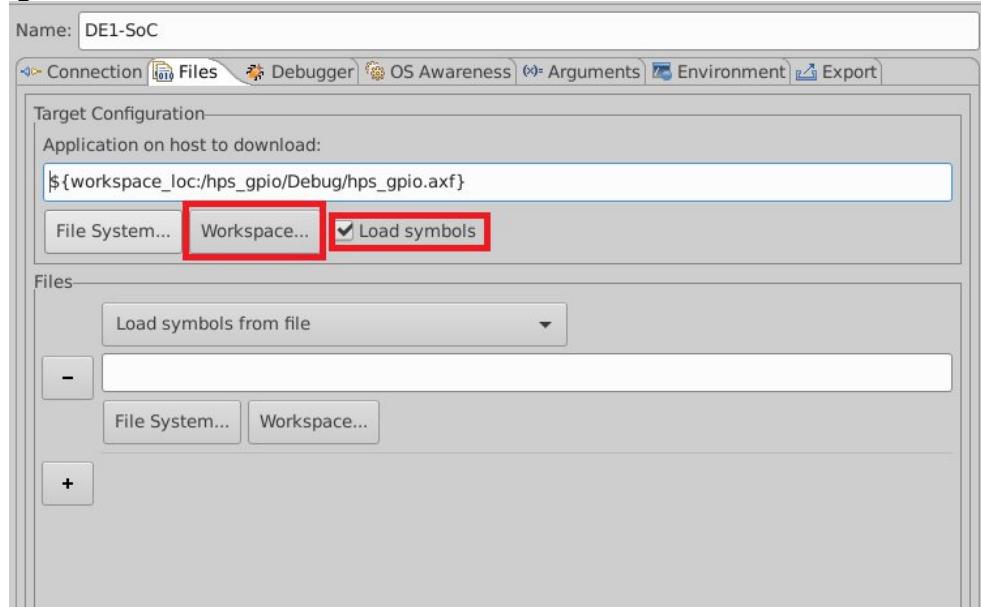
Afin de pouvoir exécuter et debugger votre programme, il est nécessaire de créer une configuration propre à votre cible (processeur).

Aller dans le menu **Run → Debug Configuration...**



1. Sélectionner une configuration de type **Generic Arm C/C++ Application**
2. Appuyer sur **New launch configuration**
3. Donner un nom à votre configuration : **DE1-SoC**
4. Dans l'onglet "Connection" : Choisir la plateforme : Intel SoC FPGA à Cyclone V SoC (Dual Core) à Bare Metal Debug à Debug Cortex-A9_0
5. Sélectionner **USB-Blaster** dans Target Connection.
6. Dans la partie Connections, cliquer sur **Browse...** et sélectionner la connexion vers la carte DE-SoC, puis **Select**. (Il faut attendre quelques secondes pour que la connexion vers la carte apparaisse, la carte doit être branchée au PC, allumée et attachée à la VM. Sur la VM, il est possible qu'il faille réattacher une deuxième fois la carte et réouvrir la fenêtre **Connection browser** pour que la carte apparaisse).

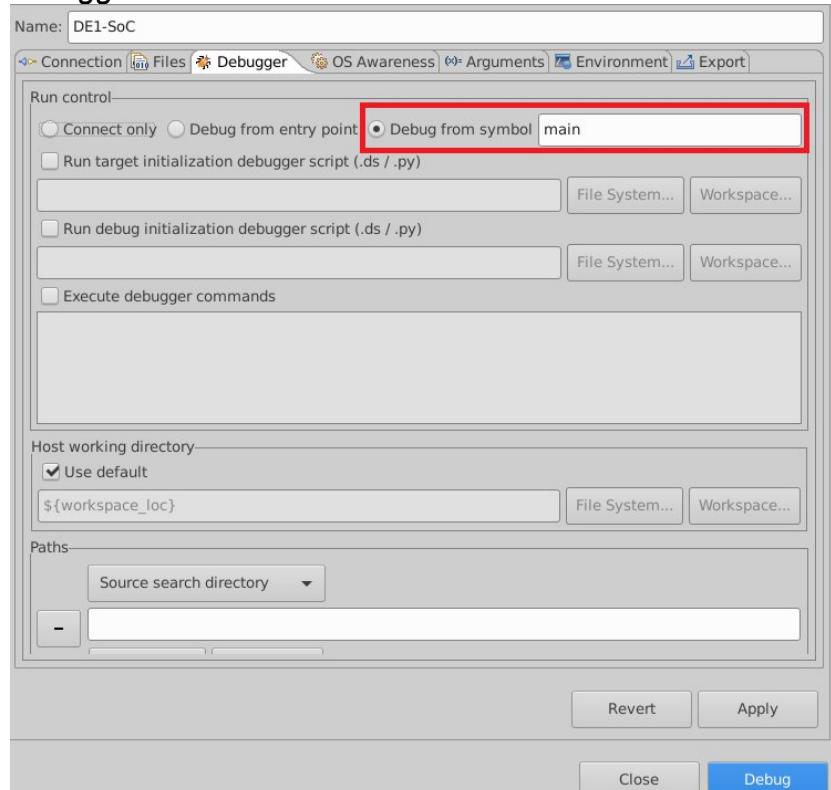
Dans l'onglet *File* :



1. Dans la partie *Target Configuration*, sélectionner le fichier « xxx.axf » générée lors de la compilation du projet en cliquant sur **Workspace...**. Puis **hps_gpio/Debug/hps_gpio.axf**

2. Cocher la case **Load Symbols**.

Dans l'onglet *Debugger* :



1. Cochez **Debug from symbol** et spécifier « main ». Cela mettra le debugger en pause à l'entrée de la fonction main lors de son lancement.

2. Puis cliquer sur **Apply** et **Close**.

Exécuter / debugger le programme C

Programmation DE1-SoC et chargement preloader du HPS

Important : Avant de pouvoir exécuter et debugger un programme C, il est nécessaire de programmer la FPGA de la DE1-SoC et de charger le preloader du HPS. Pour cela, il faut exécuter les scripts python suivant.

Ouvrir un terminal en dehors de ARM-DS. Puis aller dans le répertoire **de1_scripts** à la racine du repo des laboratoires.

Pour programmer la FPGA, exécuter le script python3 avec en paramètre le bitstream à l'aide de la commande suivante :

```
$ python3 pgm_fpga.py -s=<path_to_labs>/hard/eda/output_files/DE1_SoC_top.sof
```

Pour charger le preloader du HPS, exécuter le script python3 suivant :

```
$ python3 upld_hps.py
```

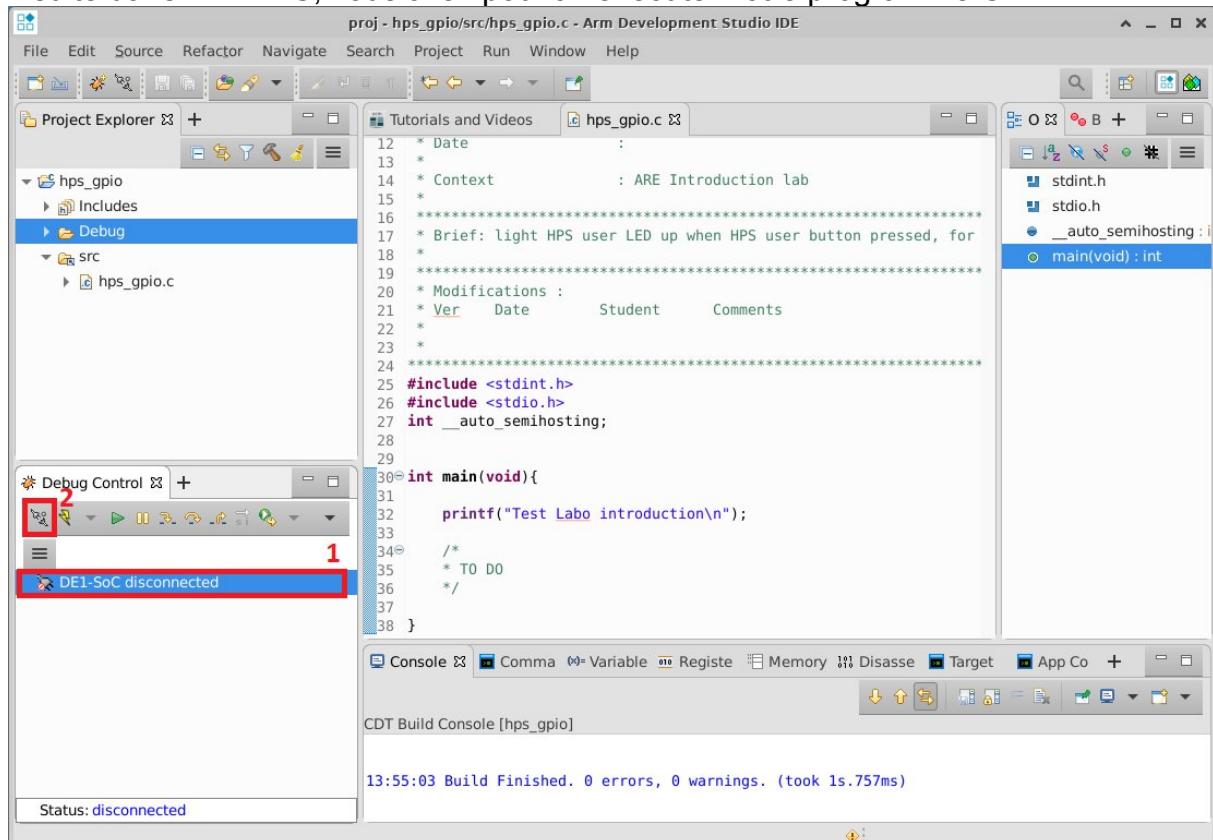
A la fin du log vous devez obtenir le message suivant :

>>Preloader successfully run.

Note : Lors de la génération d'un nouveau bitstream (.sof), il faudra relancer cette programmation avec les scripts python3. Attention, il est nécessaire de déconnecter le debugger ARM-DS de la carte sur pour exécuter les scripts.

Exécution du programme C

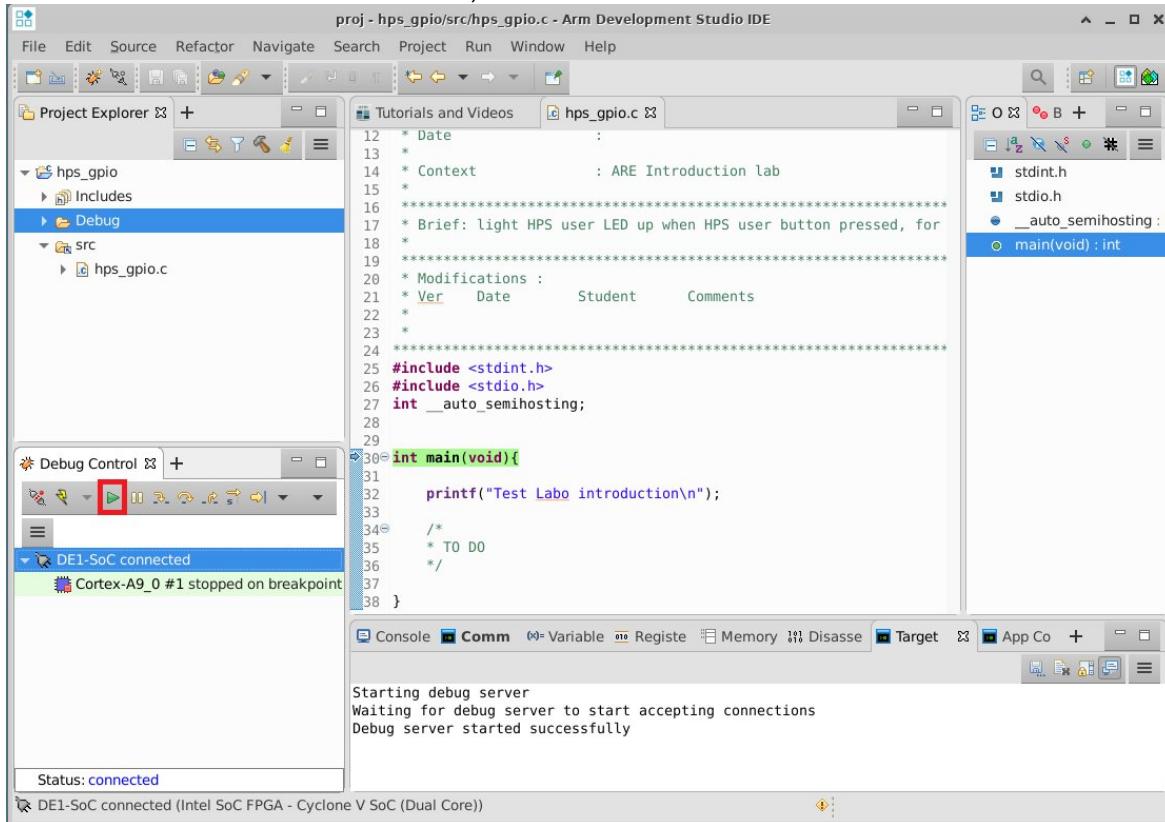
Ensuite dans ARM-DS, vous allez pouvoir exécuter votre programme C :



1. Dans l'onglet *Debug Control*, sélectionner la configuration de debug créée précédemment.

2. Appuyer sur le bouton *Connect to target*.

La connexion avec la carte doit s'établir et l'exécution du programme doit se mettre en pause au début de la fonction *main*, comme sur la fenêtre suivante.



Cliquez sur le bouton *Continue* pour poursuivre l'exécution du programme.

Depuis l'onglet *Debug Control*, vous pouvez effectuer différents types d'actions, en voici certaines :

- Continuer l'exécution du programme jusqu'à la fin ou un breakpoint.
- Mettre le programme en pause.
- Exécuter la ligne suivante.
- Déconnecter le debugger de la carte.

Si vous souhaitez quitter proprement l'exécution du programme pour ensuite le réexécuter plus tard, il faut mettre le programme en pause et déconnecter le debugger de la carte.

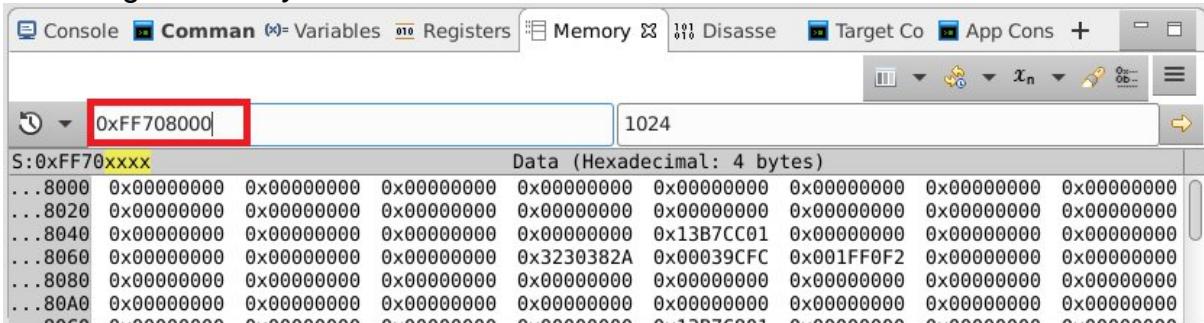
Afin de pouvoir utiliser la commande *printf* dans votre programme C, il est nécessaire d'ajouter les lignes suivantes avant la fonction *main* :

```
#include <stdio.h>
int __auto_semihosting;
```

Ecriture et lecture à une adresse mémoire

Pour exécuter des écritures et lectures à des adresse mémoire, il faut avoir connecté le debugger à la carte et mettre l'exécution du programme en pause.

Dans l'onglet *Memory* :



Entrer l'adresse mémoire complète sur 32bits pour l'écriture ou la lecture, puis appuyer sur la touche « Enter » de votre clavier.

Vous obtiendrez une représentation de la mémoire avec toutes les valeurs lus.

Pour faire une écriture, double-clic sur la valeur dans la représentation de la mémoire. Puis, écrire la nouvelle valeur et appuyer sur la touche « Enter » de votre clavier. L'écriture est exécutée à l'adresse mémoire.

Ajouter un breakpoint

Pour ajouter un breakpoint dans un programme C, ouvrir le code C dans l'éditeur de ARM-DS.

Faire un double-clic sur le numéro de ligne à gauche de la ligne désiré pour mettre le breakpoint.

Un point rouge en face de la ligne doit apparaître, cela signale le breakpoint :

```

30
31 int main(void){
32
33     printf("Test Labo introduction\n");
34

```

Lors de l'exécution du programme, il doit s'arrêter à chaque passage sur un breakpoint.

Pour supprimer un breakpoint, faire également un double-clic sur le numéro de ligne et le point rouge doit s'enlever.

Visualisation du code assembleur du programme

Lorsque l'exécution du programme est en pause, dans l'onglet "Disassembly" vous pouvez voir le programme assembleur.