INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES HAUTS-DE-FRANCE

Université Polytechnique HAUTS-DE-FRANCE

TP – Comparaison AES HW et AES SW

Antoine Aubert

Yann Birembaux

28/05/2023

Implémentation

Installation de Quartus

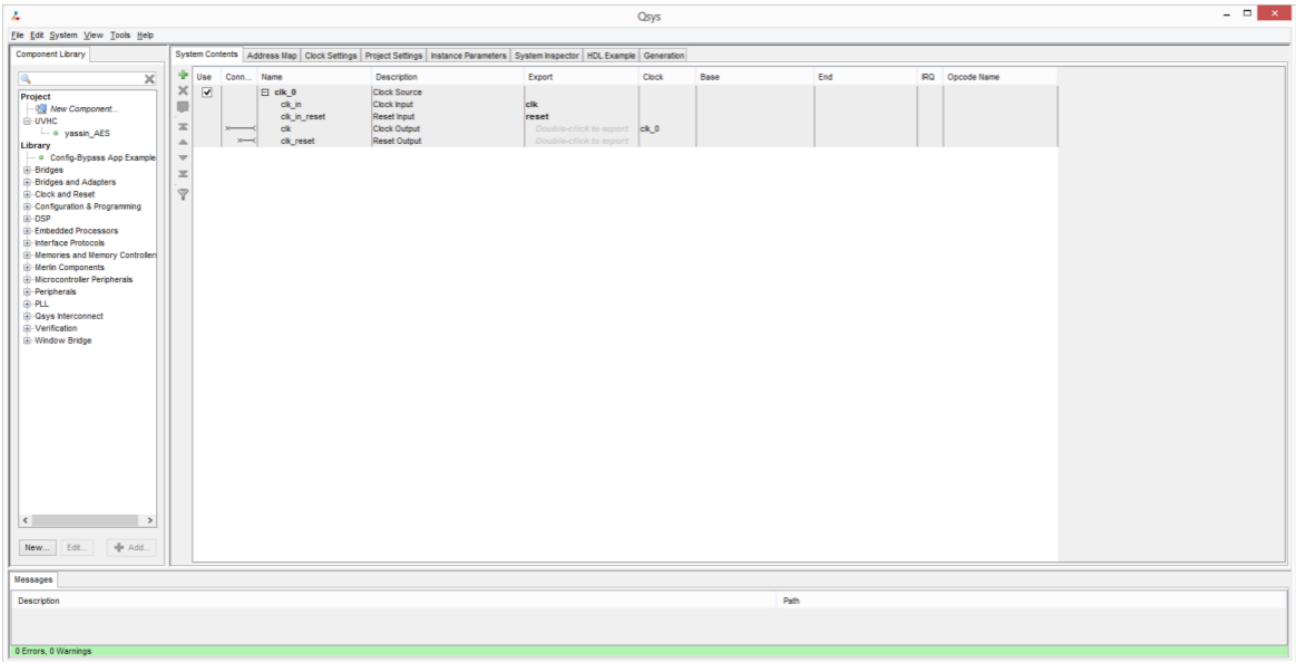
Décompression du fichier « designs_start_0.rar » dans le répertoire C:/altera/13.0sp1/designs/

Copie du contenu du répertoire « copier_dans_HW_SW_AES » dans le répertoire

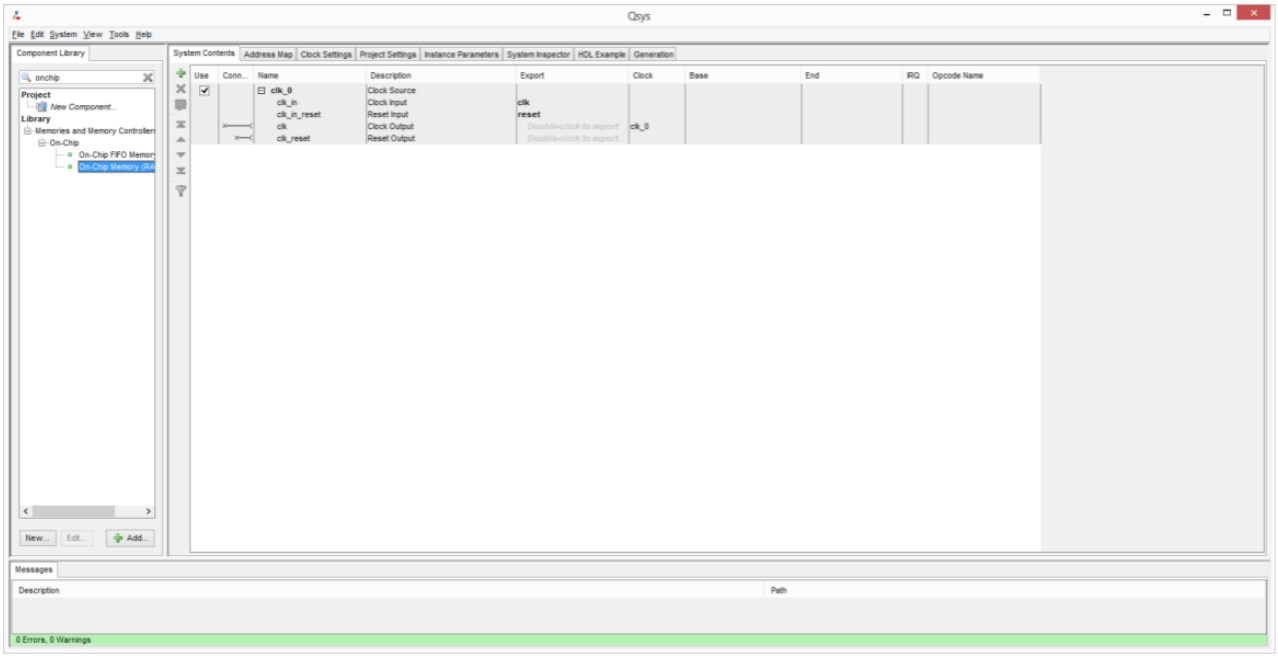
C:/altera/13.0sp1/designs/ HW_SW_AES/

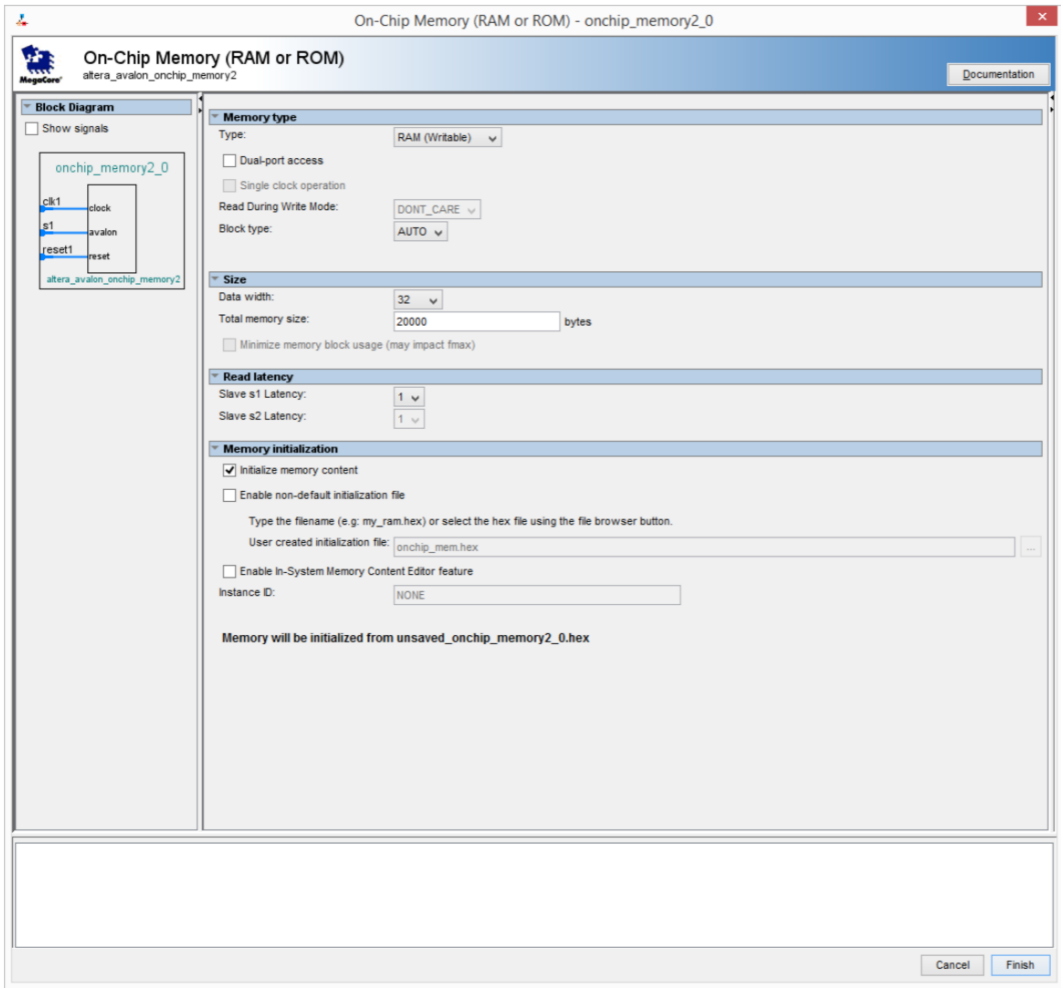
Lancement de Quartus et ouverture du projet HW_SW_AES

Lancement de Qsys

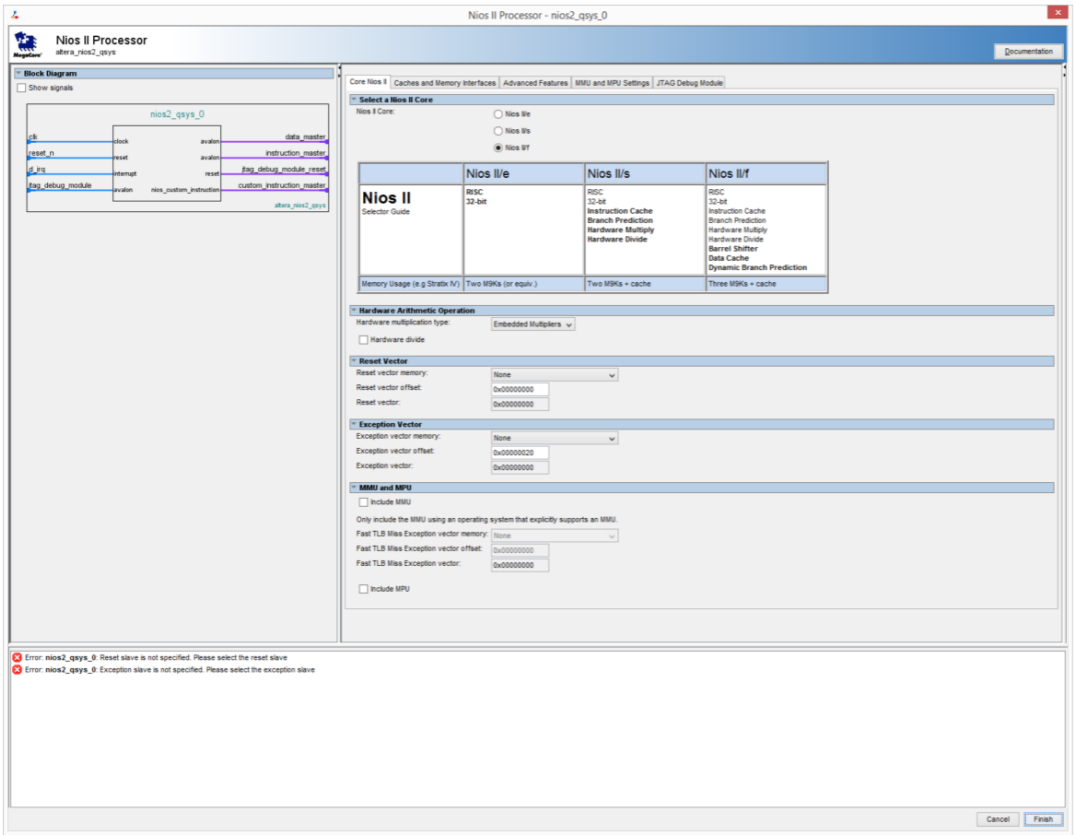


Ajout de onchip_memory2_0

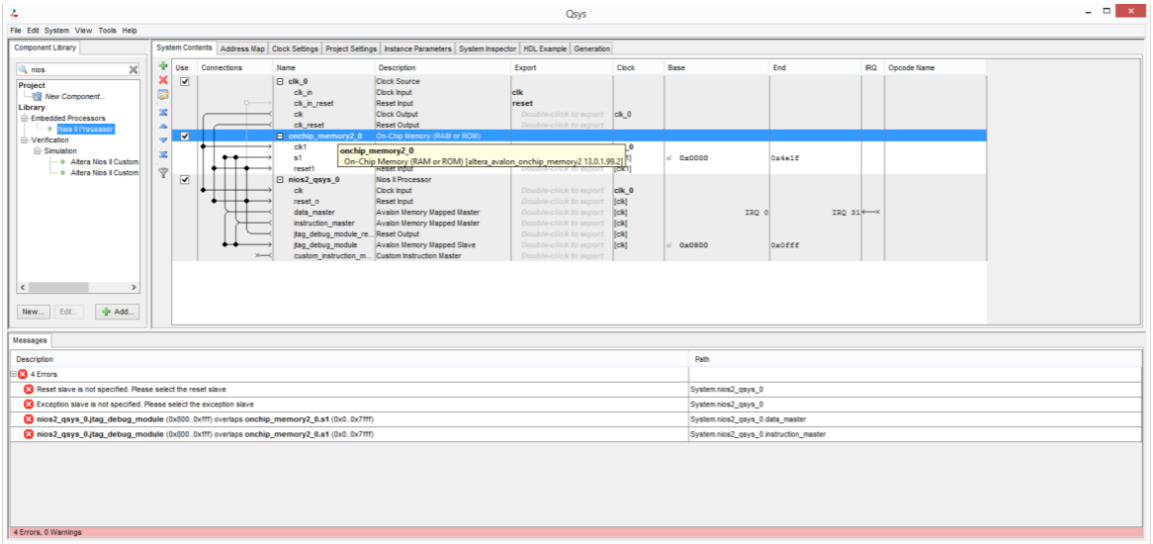

Qsys
File Edit System View Tools Help
Component Library
System Contents
Address Map
Clock Settings
Project Settings
Instance Parameters
System Inspector
HDL Example
Generation
Messages
0 Errors, 0 Warnings


On-Chip Memory (RAM or ROM) - onchip_memory2_0
On-Chip Memory (RAM or ROM)
altera_avalon_onchip_memory2
Documentation
Block Diagram
Show signals
onchip_memory2_0
clk1
s1
reset1
clock
avalon
reset
altera_avalon_onchip_memory2
Memory type
Type:
RAM (Writable)
Dual-port access
Single clock operation
Read During Write Mode:
DONT_CARE
Block type:
AUTO
Size
Data width:
32
Total memory size:
20000
bytes
Minimize memory block usage (may impact fmax)
Read latency
Slave s1 Latency:
Slave s2 Latency:
Memory initialization
Initialize memory content
Enable non-default initialization file
Type the filename (e.g: my_ram.hex) or select the hex file using the file browser button.
User created initialization file:
onchip_mem.hex
Enable In-System Memory Content Editor feature
Instance ID:
NONE
Memory will be initialized from unsaved_onchip_memory2_0.hex
Cancel
Finish

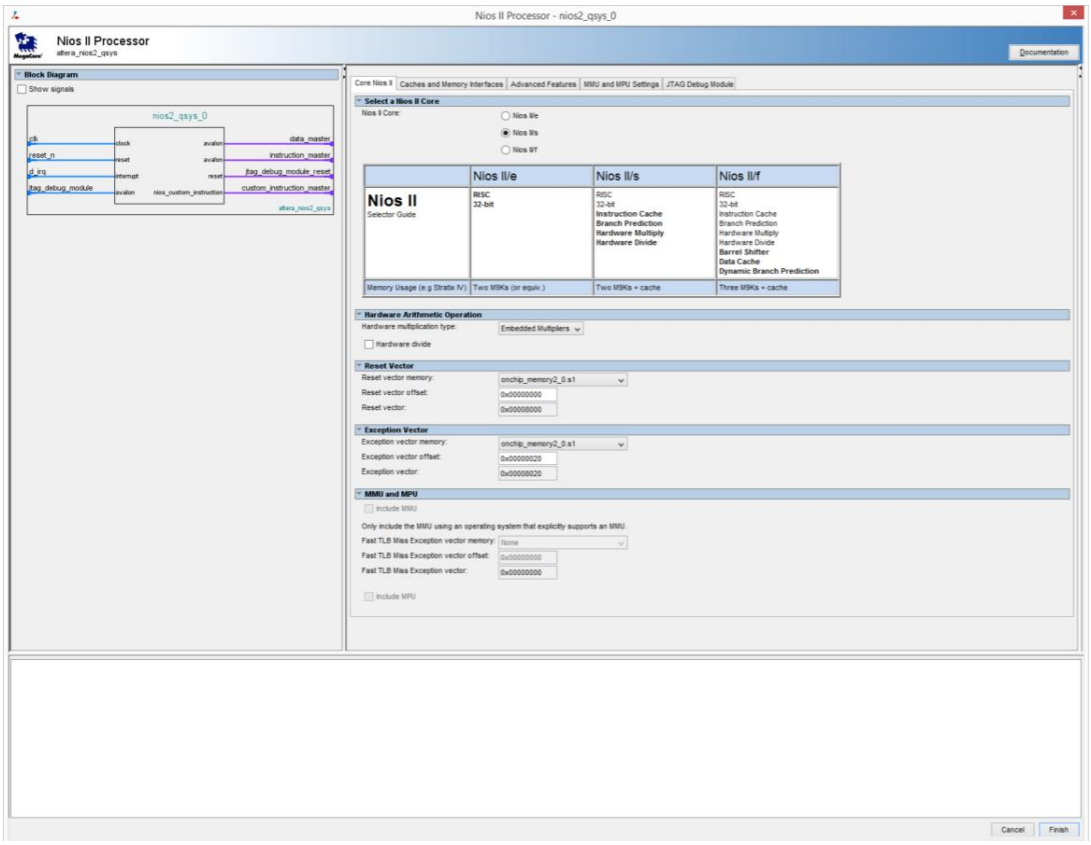
Ajout du CPU



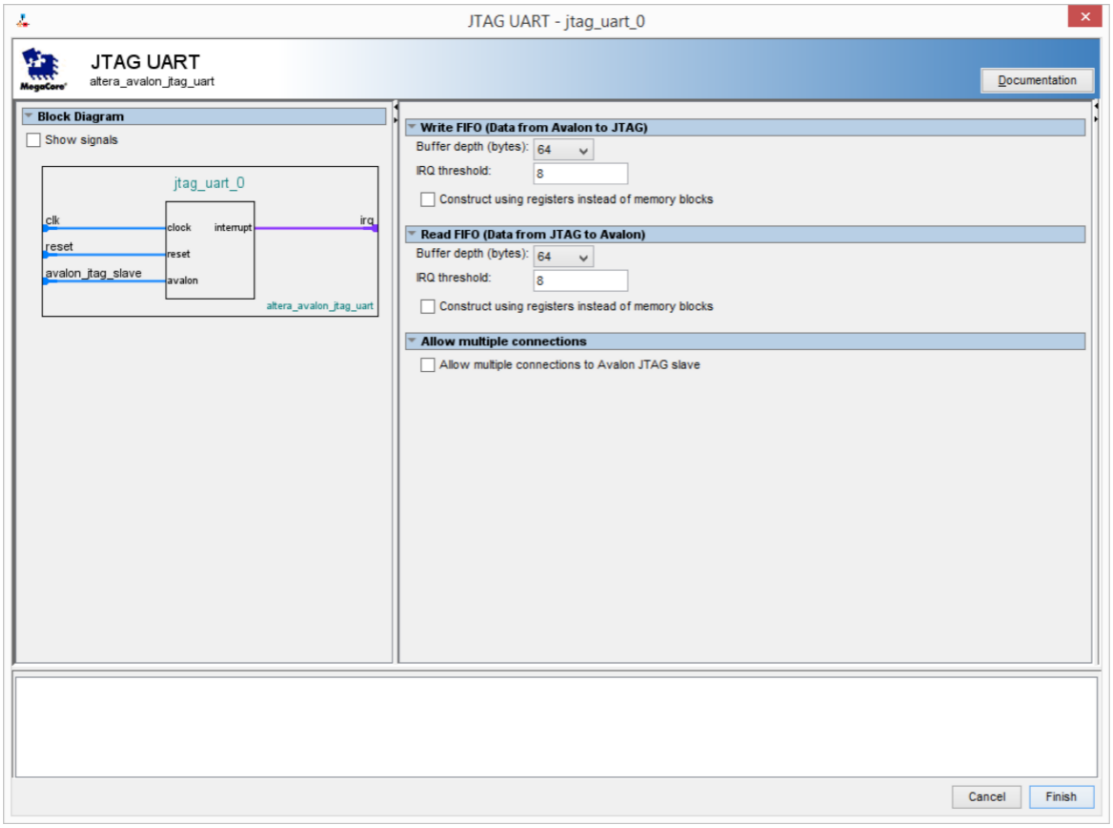
Connection du CPU



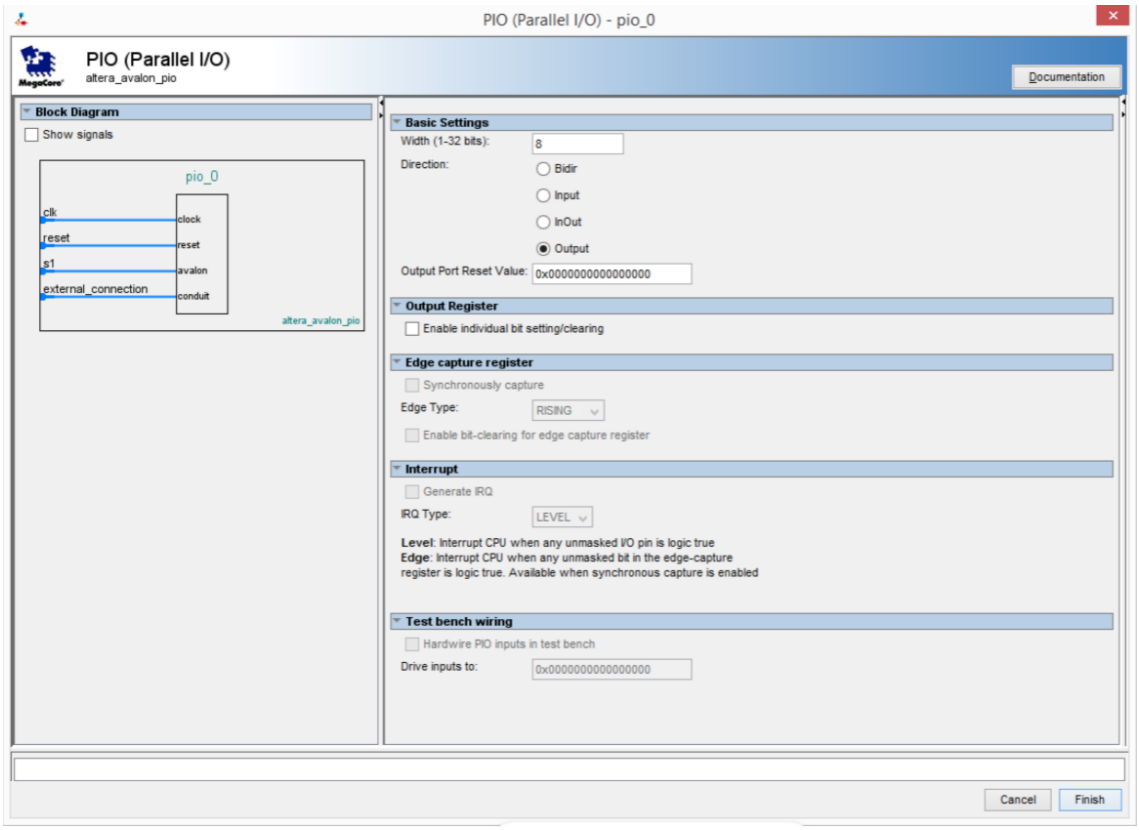
Configuration du CPU



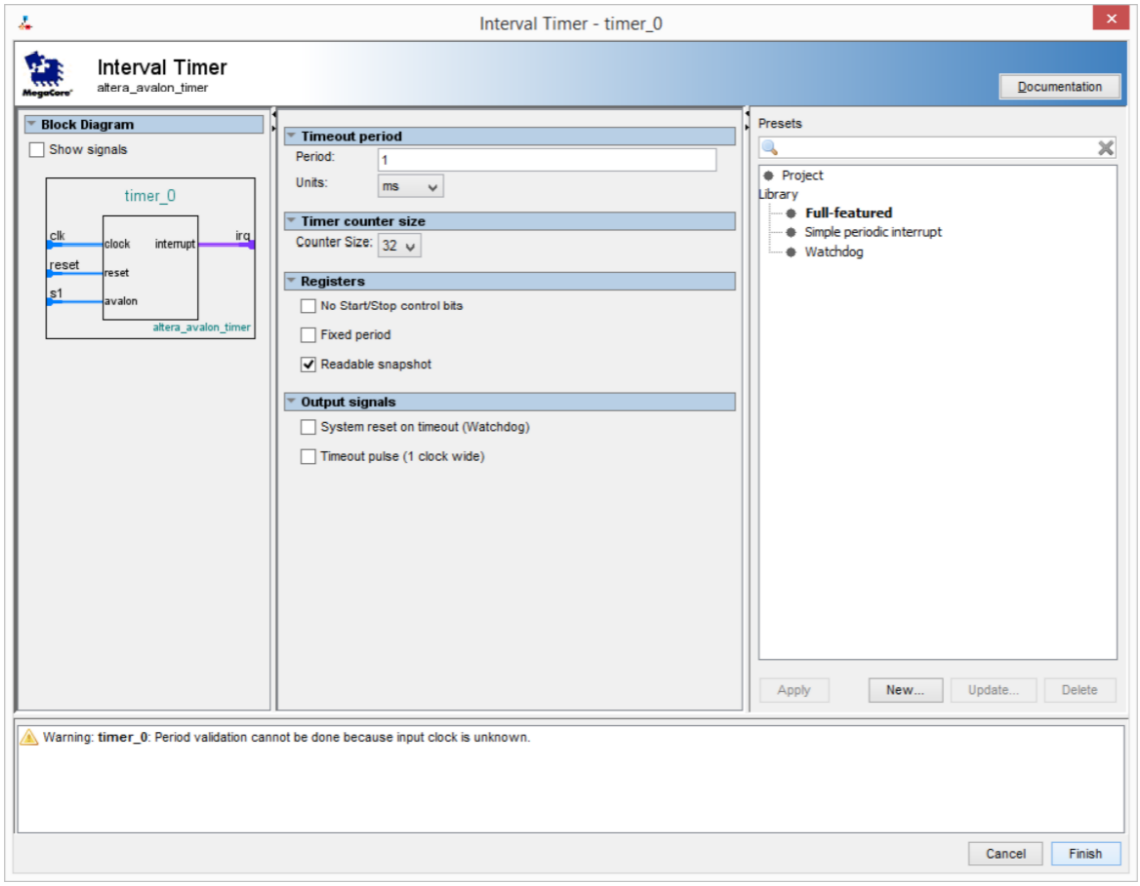
Ajout du module JTAG UART



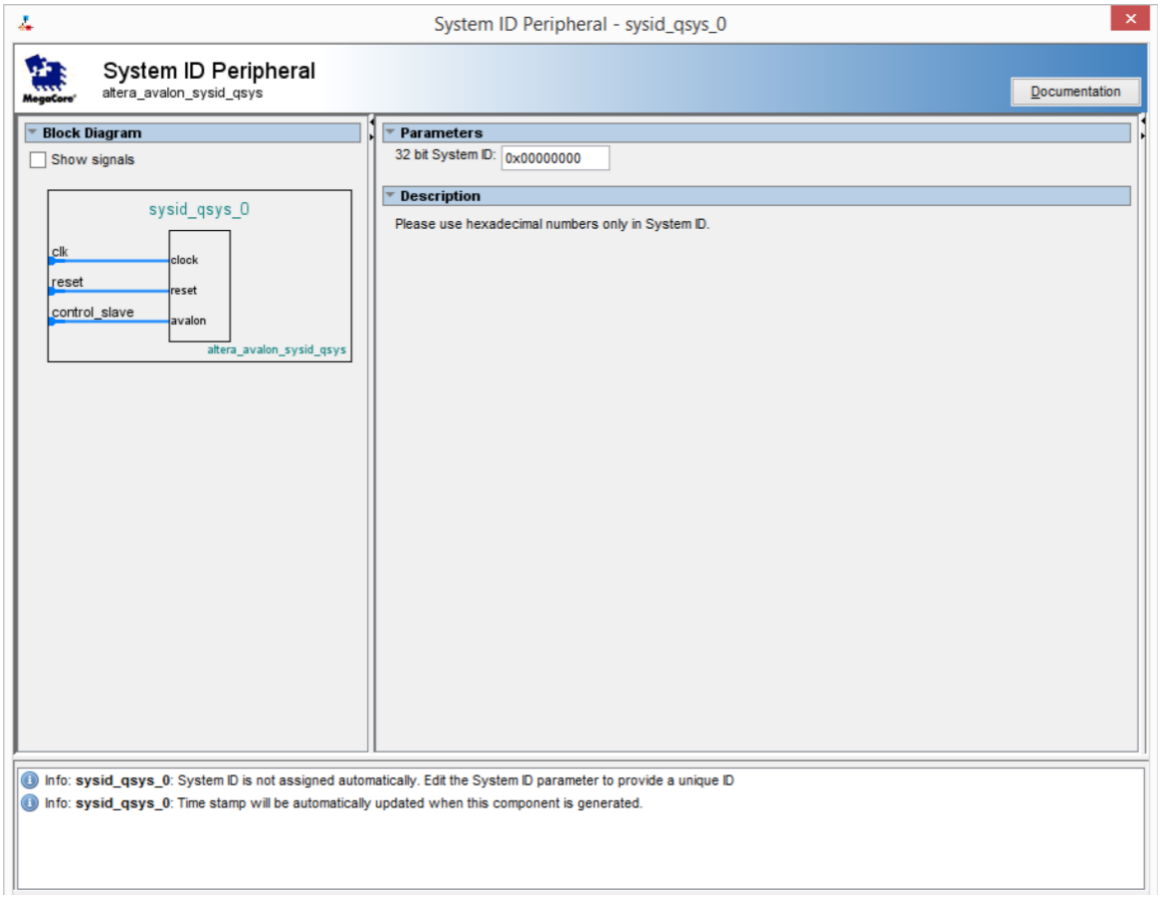
Ajout du module PIO (Parallèle I/O)



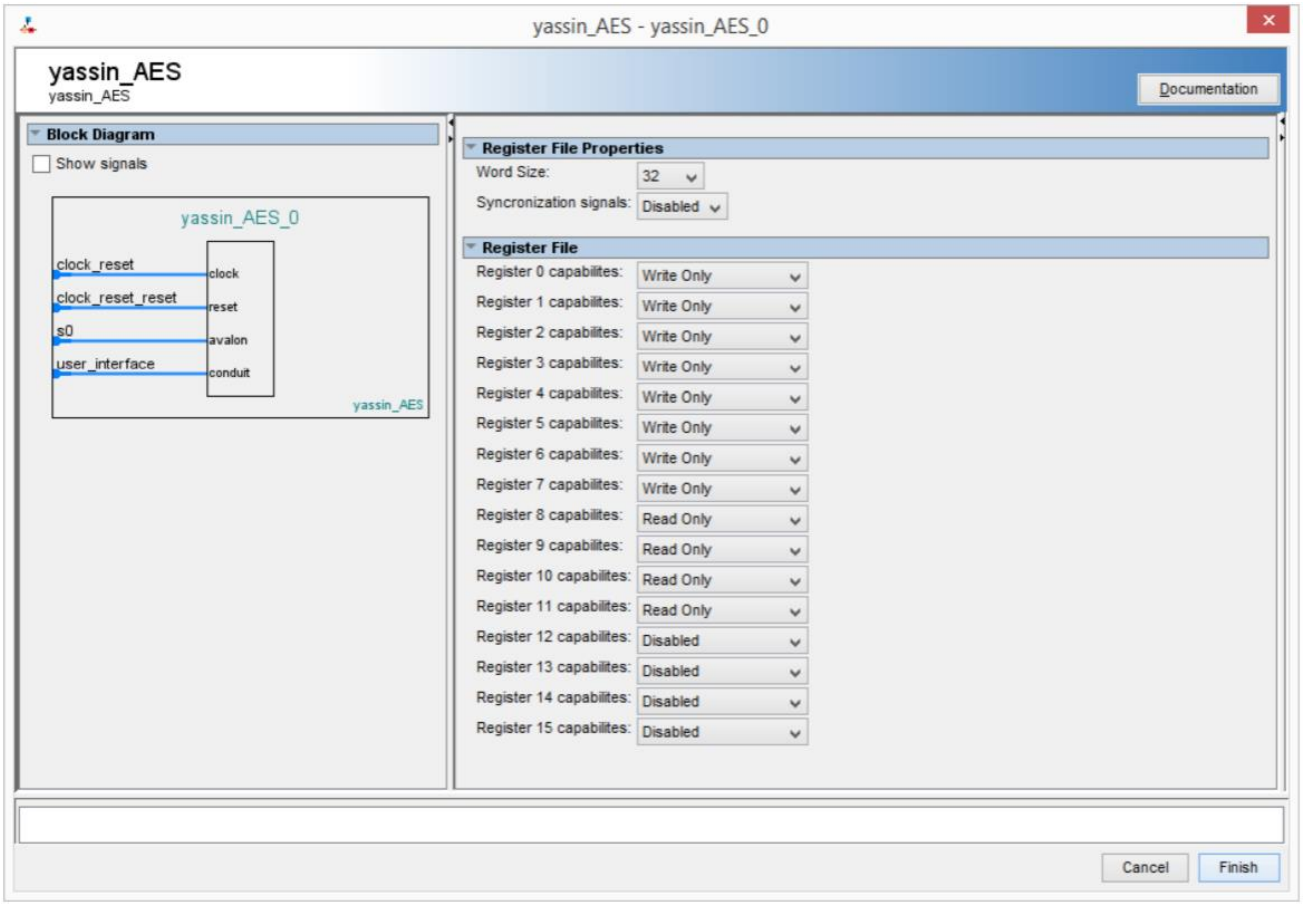
Ajout du module interval Timer



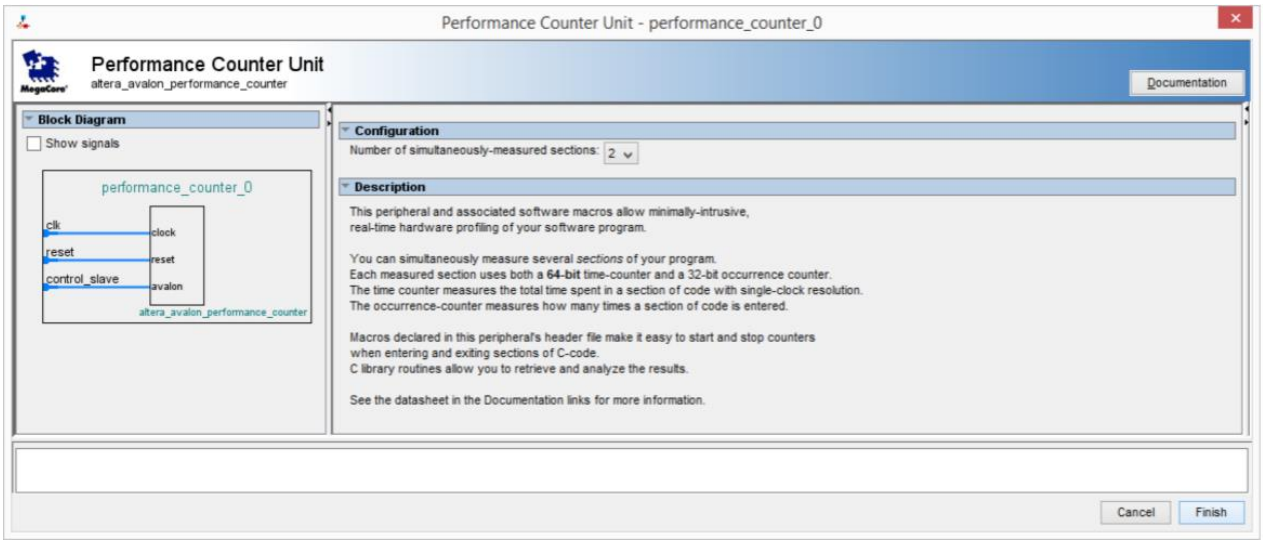
Ajout du module system ID Peripheral



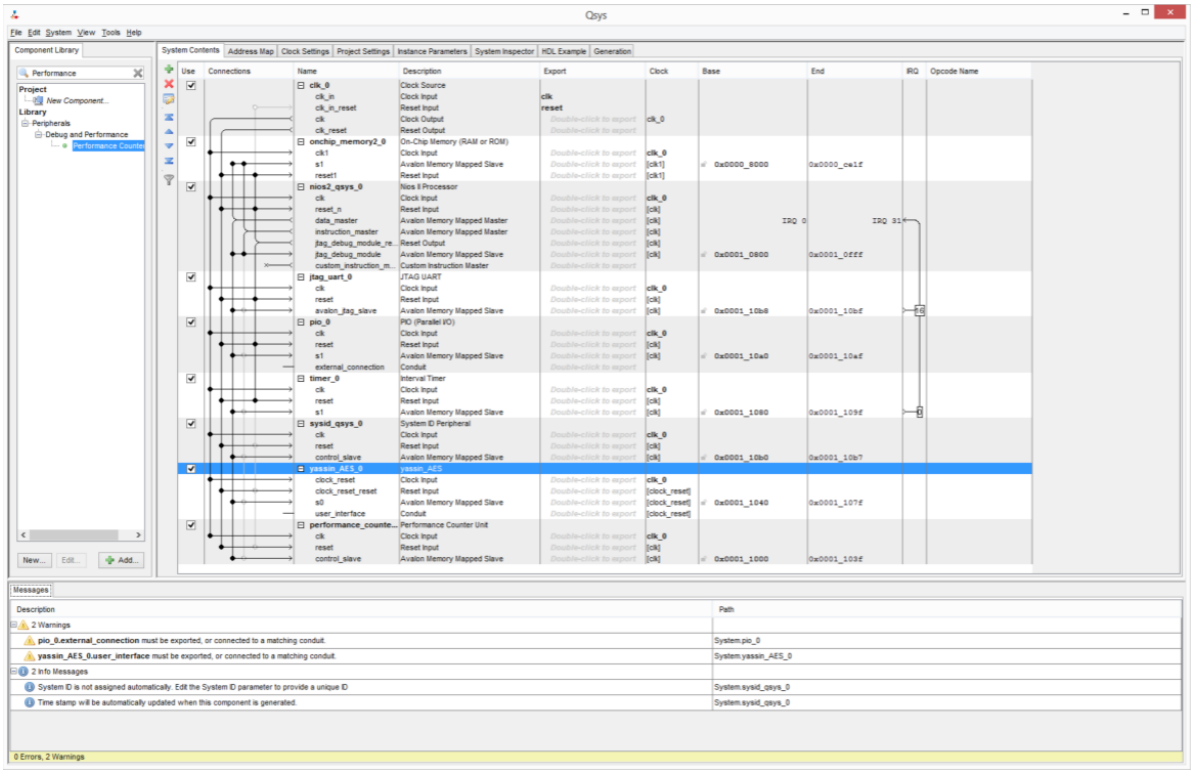
Ajout du module Yassin_AES



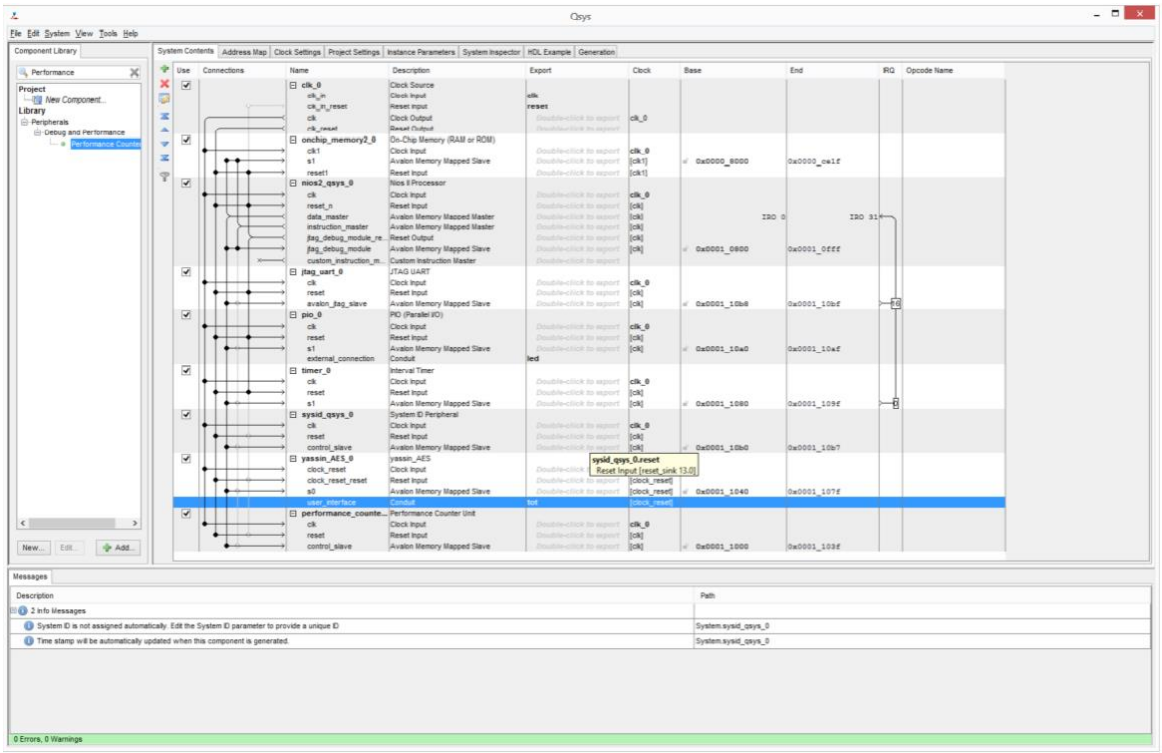
Ajout du module Performance Counter UNIT



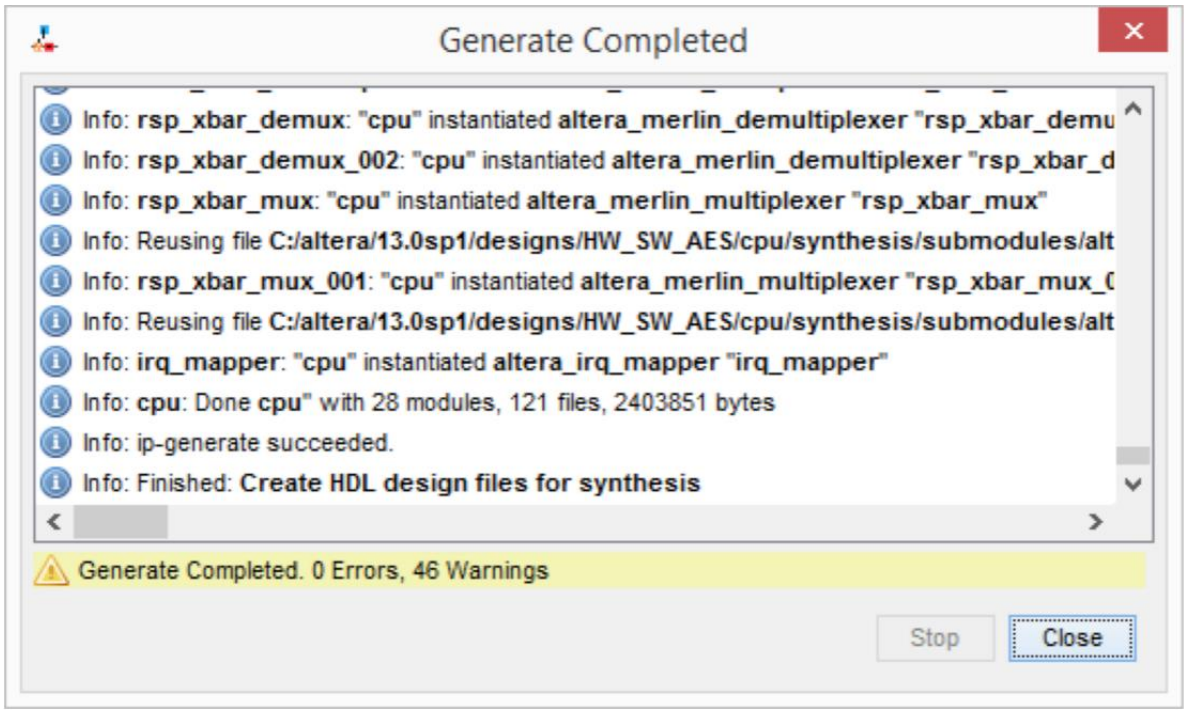
Connexion des différents modules.



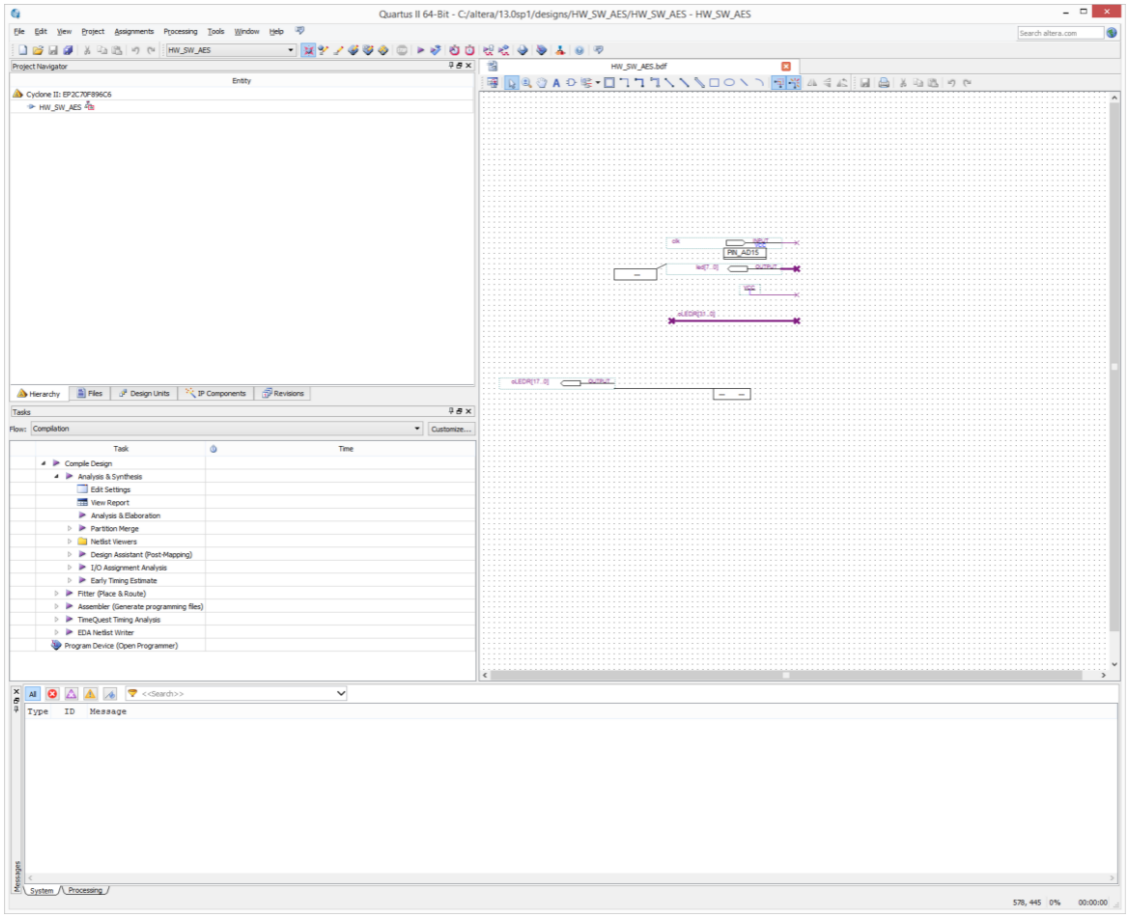
Assignation des adresses de base et exportation des sorties LED et AES



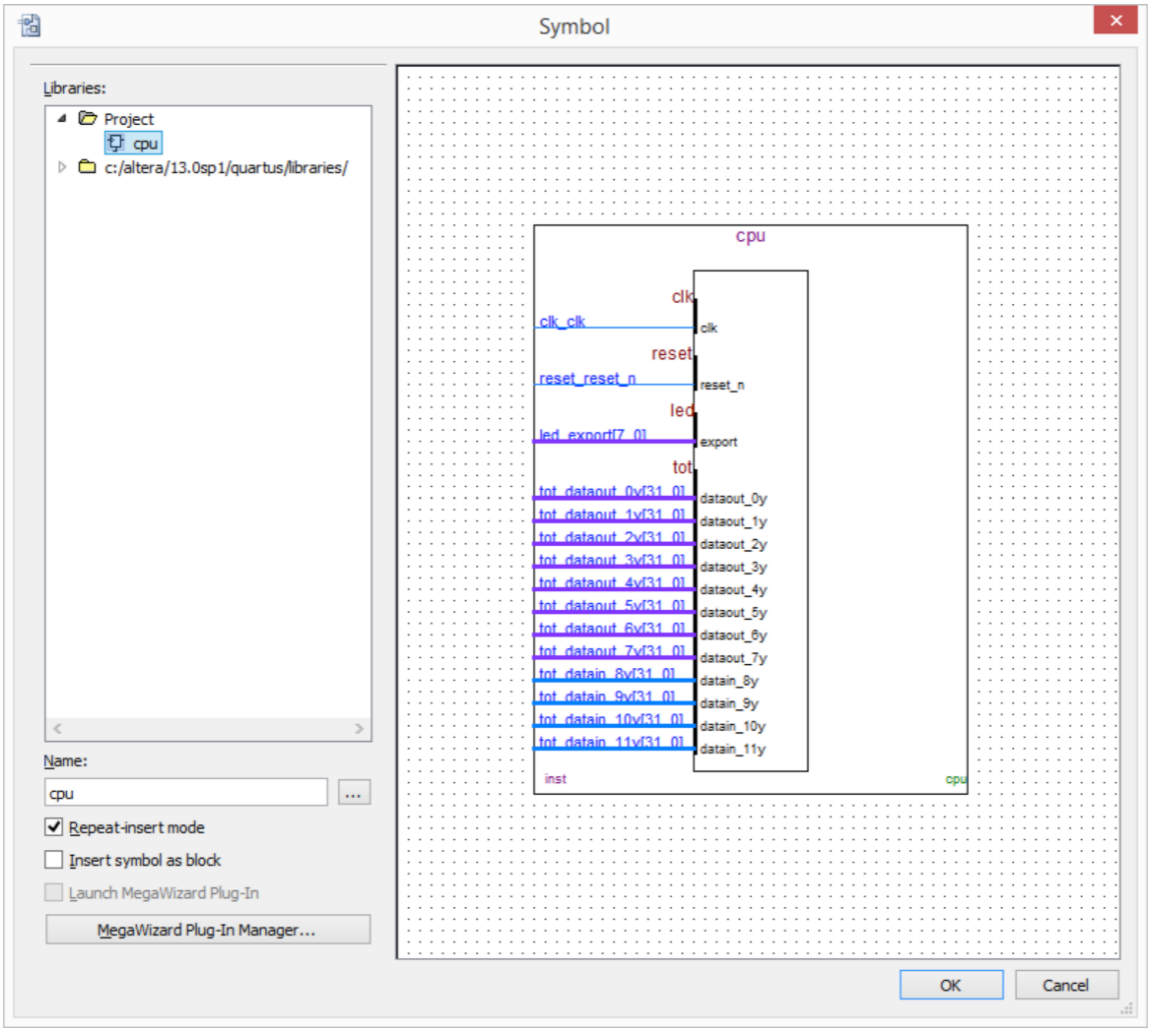
Génération du système



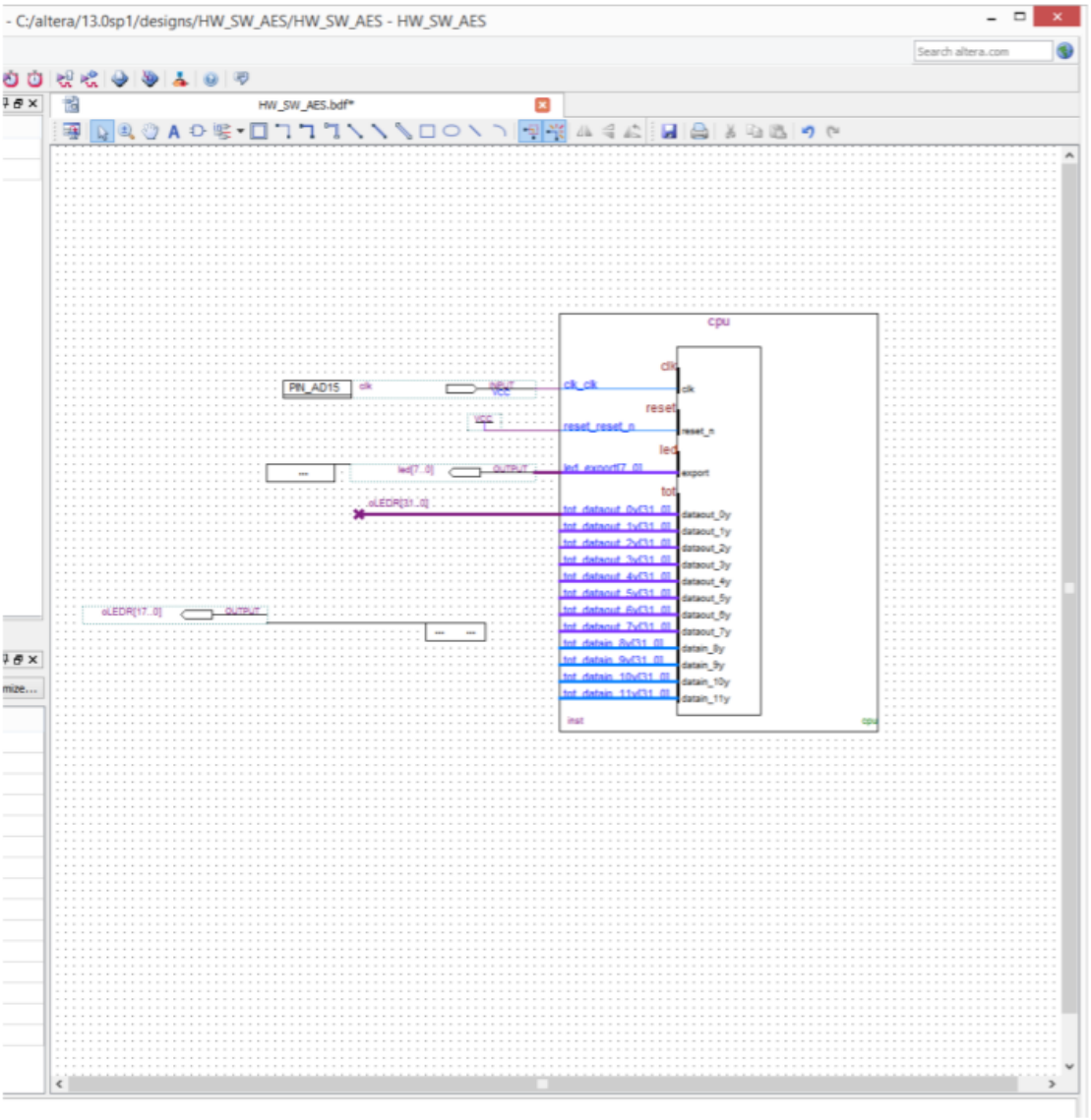
Ouverture du fichier top level file



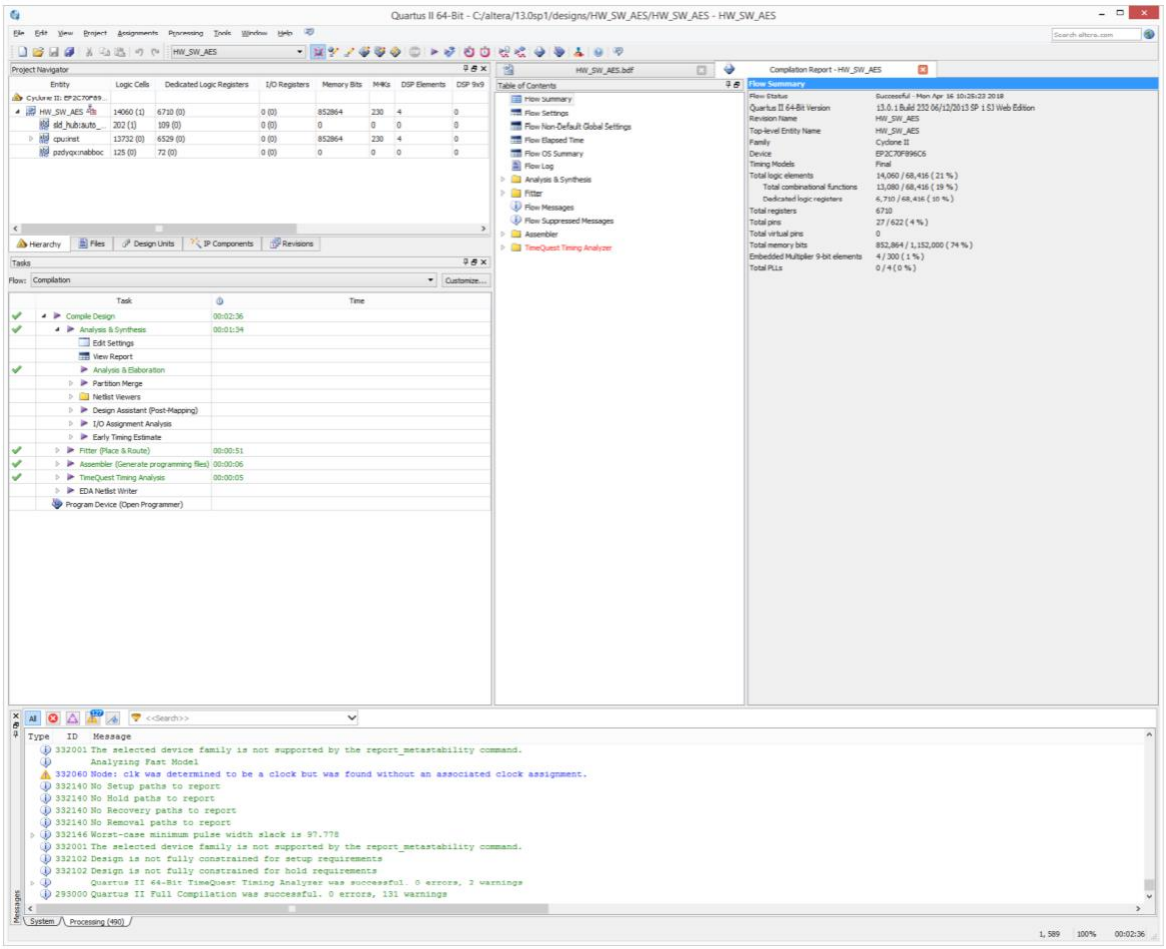
Ajout de notre CPU



Vérification et modification des connections CPU



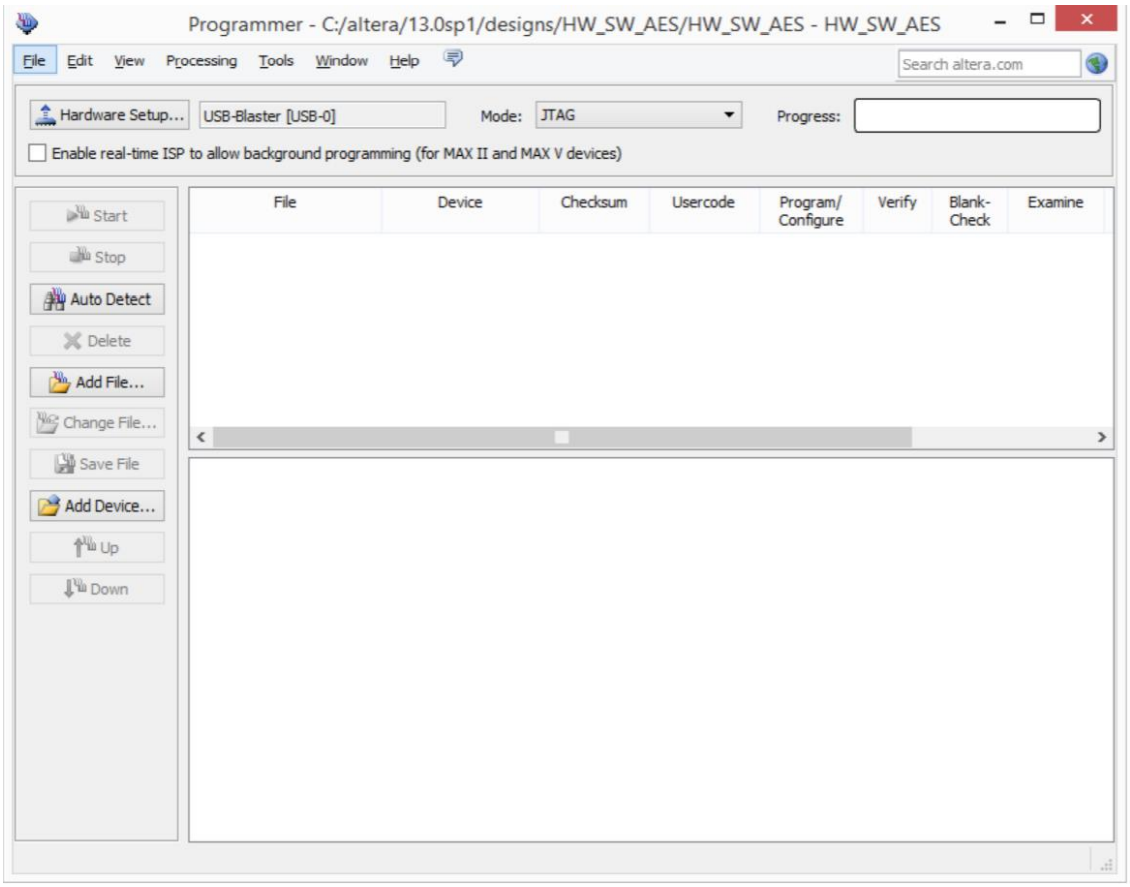
Compilation du fichier et résultat :



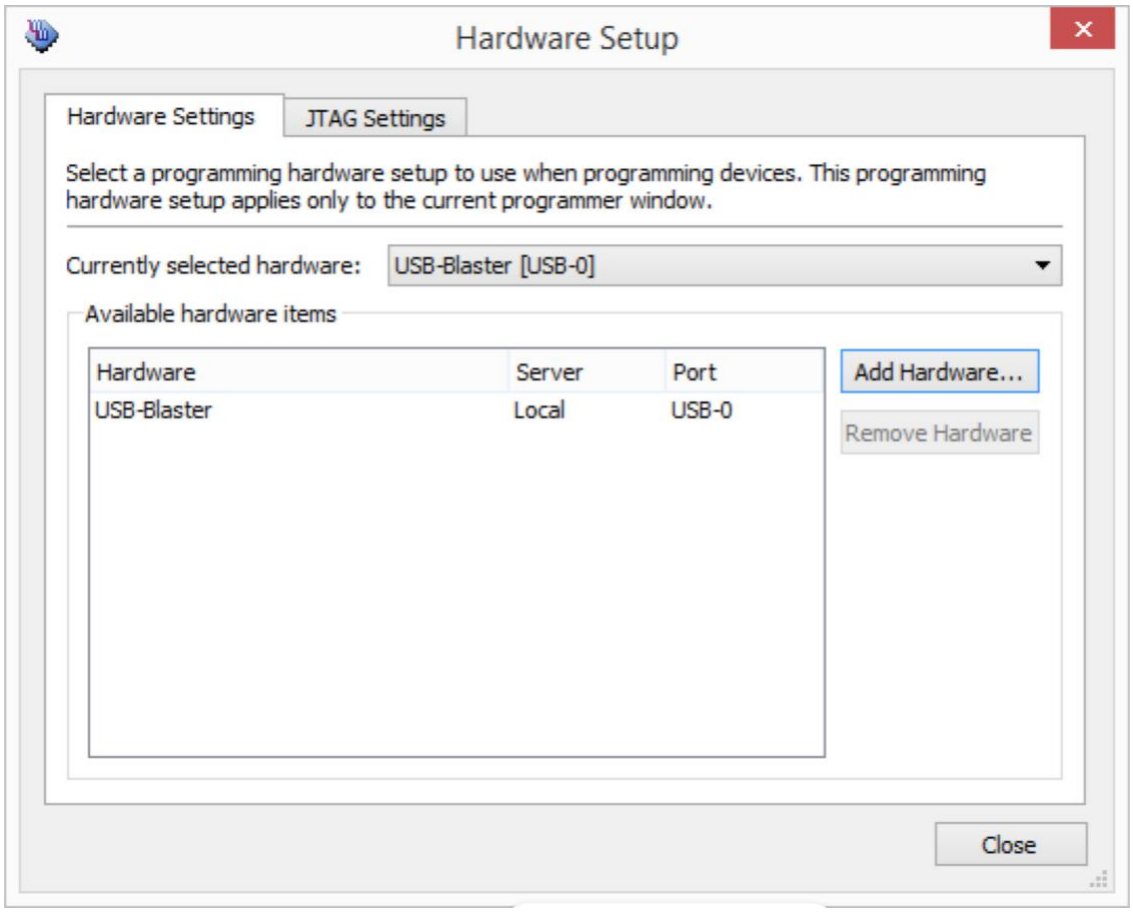
Programmation de la carte

Lancement et configuration

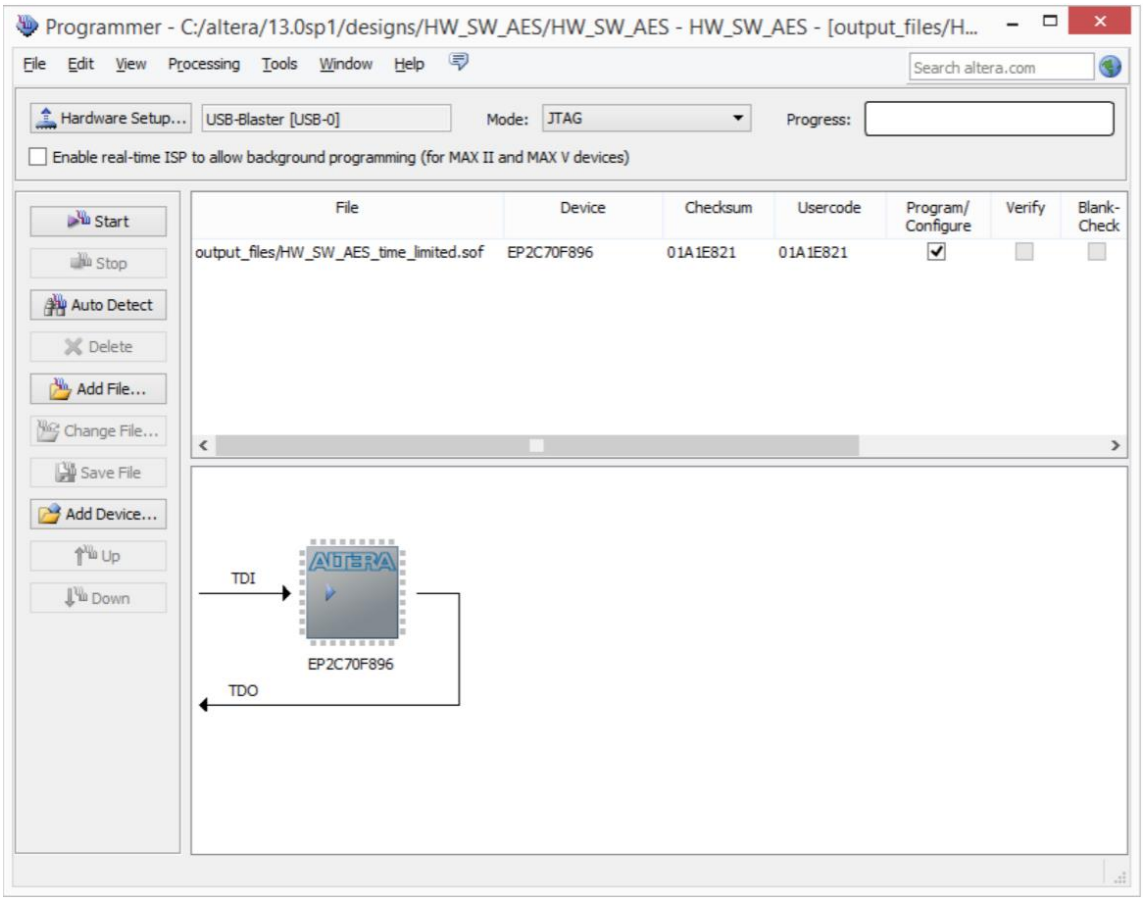
Utilisation de la fonction hardware setup pour connecter la carte



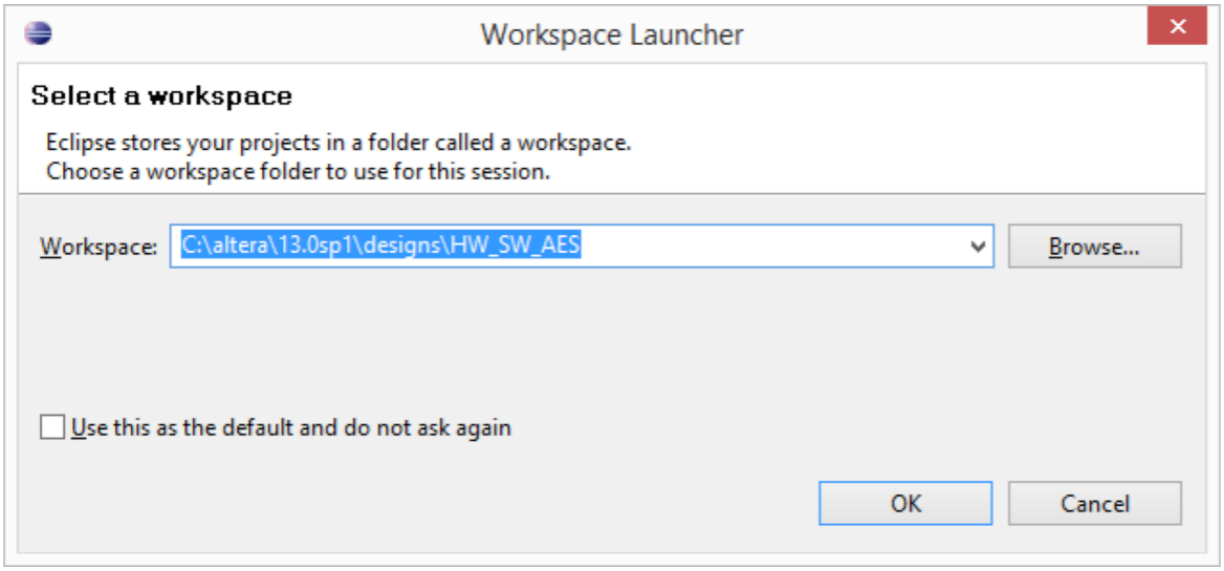
Connection via le port USB-Blaster



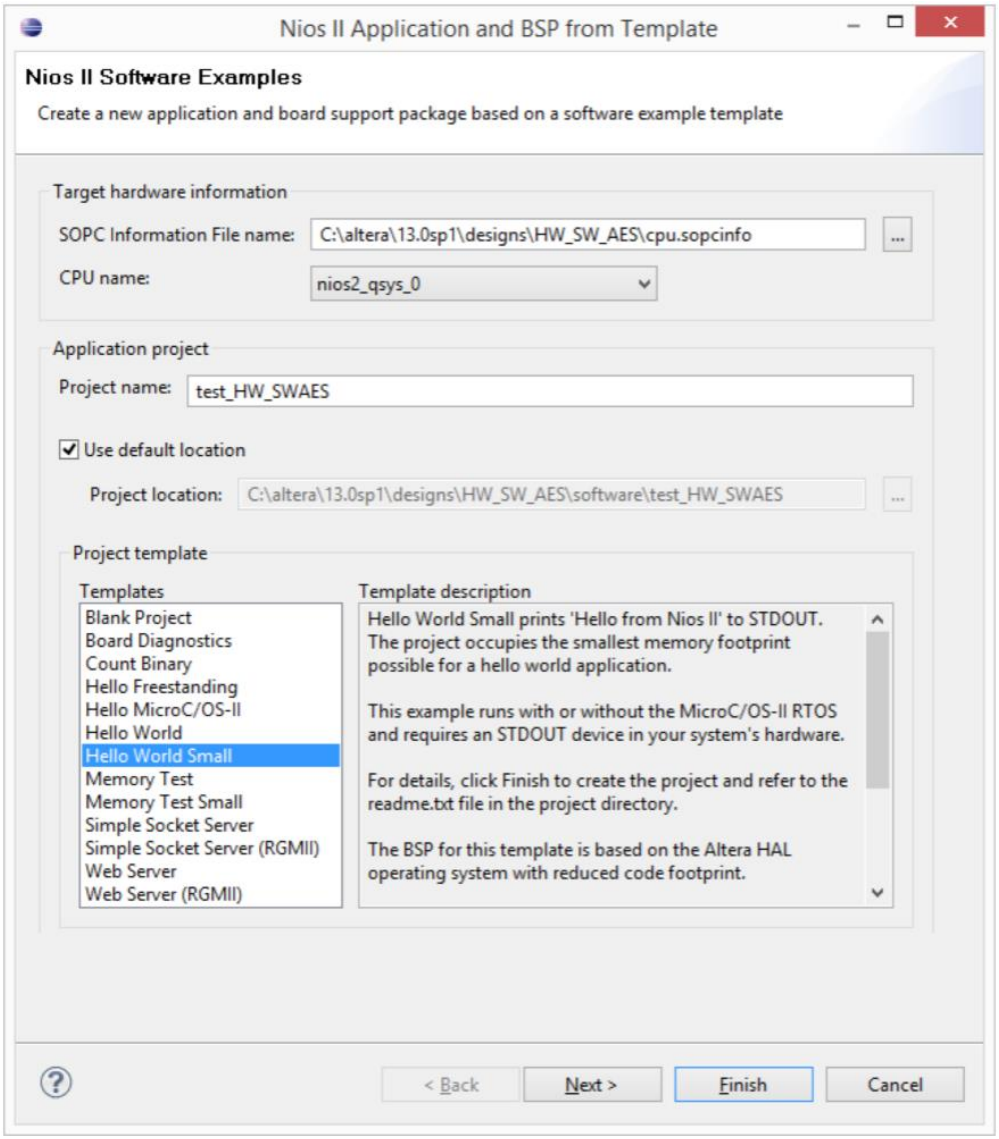
Utilisation de la fonction auto détecte pour trouver le module



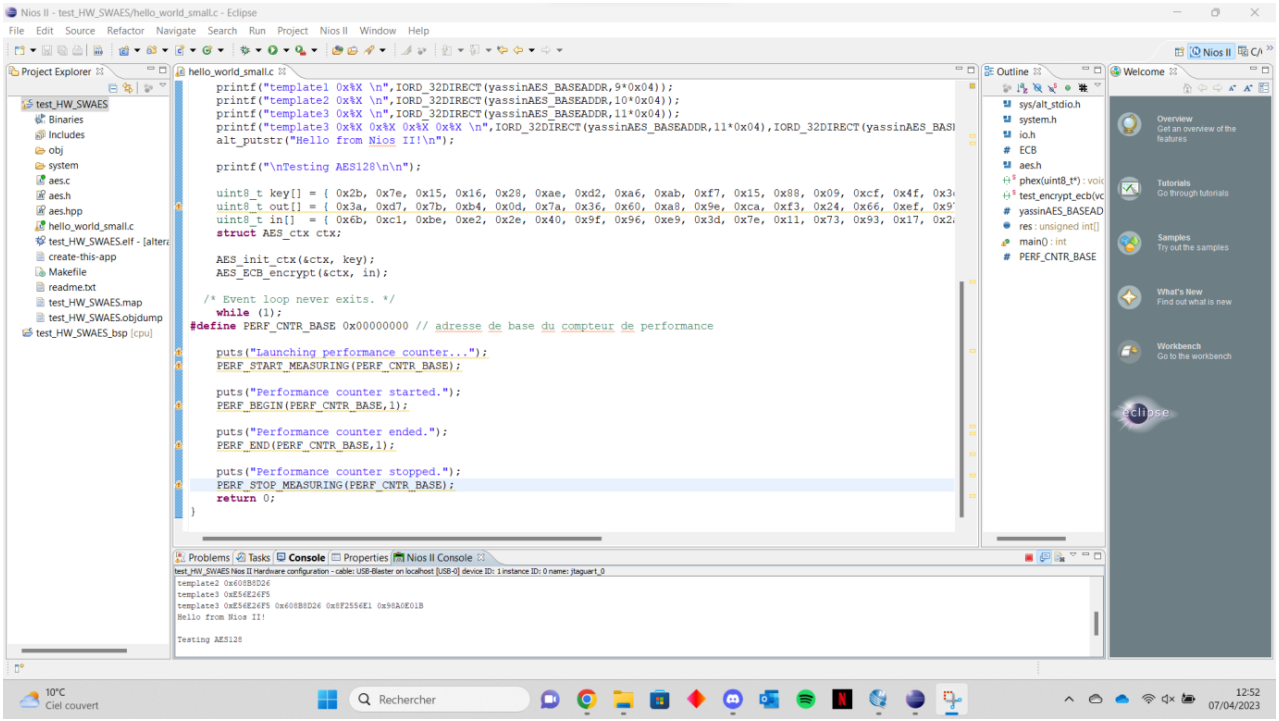
Ouverture de Eclipse et création du Workspace



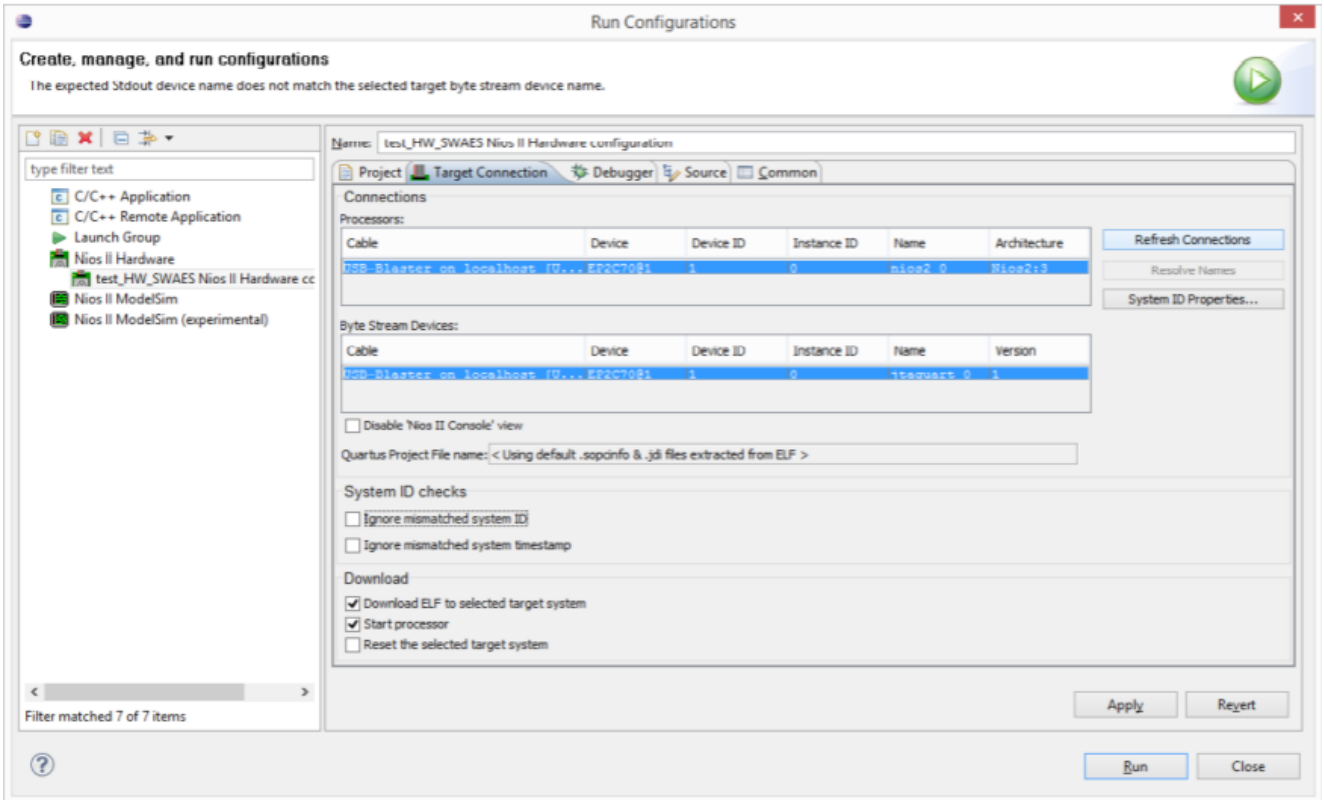
Création du nouveau projet BSP, utilisation de Template par défaut



Injection du code dans l'IDE

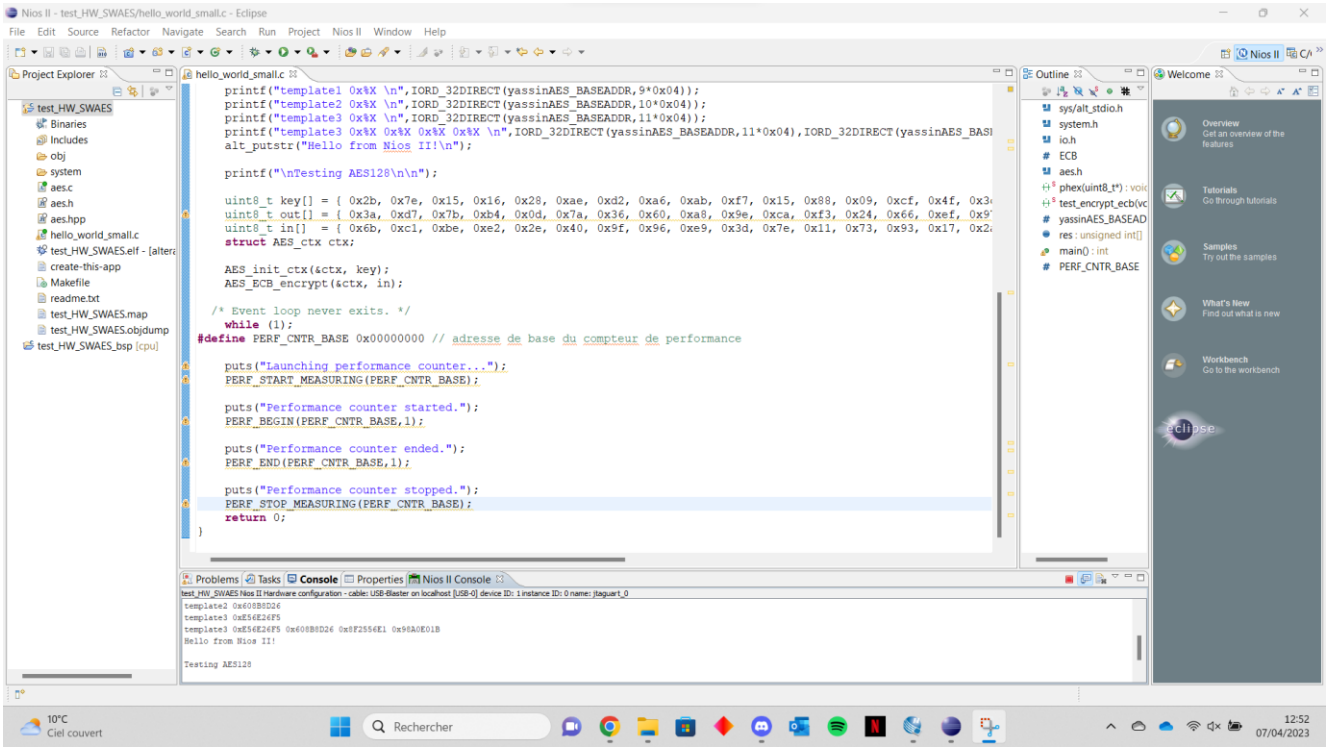


Configuration de Target connections puis Refrech connections



Test de performance et comparaison

Ajout des différentes fonctions puis vérification du Run



Ajout des différentes fonctions il faut aussi ajouter la bibliothèque altera_avalon_performance_counter.h

```c
#include "sys/alt_stdio.h"
#include <system.h>
#include <io.h>

// Definition des adresses de base des composants
#define yassinAES_BASEADDR YASSIN_AES_0_BASE
#define PERF_CNTR_BASE 0x12345678 // Remplacez par l'adresse de base du compteur de performances

// Declaration des variables globales
unsigned int res[4];

// Declaration des fonctions
void PERF_BEGIN(unsigned int base_addr, unsigned int counter);
void PERF_END(unsigned int base_addr, unsigned int counter);
unsigned int perf_get_section_time(unsigned int base_addr, unsigned int counter);

// Fonction principale
int main()
{
    alt_putstr("Hello from Nios II!\n");
    alt_putstr("Load KEY!\n");
    IOWR_32DIRECT(yassinAES_BASEADDR, 0x00, 0x11111111);
    IOWR_32DIRECT(yassinAES_BASEADDR, 1 * 0x04, 0x11111111);
    IOWR_32DIRECT(yassinAES_BASEADDR, 2 * 0x04, 0x11111111);
    IOWR_32DIRECT(yassinAES_BASEADDR, 3 * 0x04, 0x11111111);
    alt_putstr("Load data!\n");
    IOWR_32DIRECT(yassinAES_BASEADDR, 4 * 0x04, 0x11111111);
    IOWR_32DIRECT(yassinAES_BASEADDR, 5 * 0x04, 0x11111111);
    IOWR_32DIRECT(yassinAES_BASEADDR, 6 * 0x04, 0x11111111);
    IOWR_32DIRECT(yassinAES_BASEADDR, 7 * 0x04, 0x11111111);
    alt_putstr("retrive output!\n");

    PERF_BEGIN(PERF_CNTR_BASE, 1); // D�but de la mesure de performance

    printf("template0 0x%X \n", IORD_32DIRECT(yassinAES_BASEADDR, 8 * 0x04));
    printf("template1 0x%X \n", IORD_32DIRECT(yassinAES_BASEADDR, 9 * 0x04));
    printf("template2 0x%X \n", IORD_32DIRECT(yassinAES_BASEADDR, 10 * 0x04));
    printf("template3 0x%X \n", IORD_32DIRECT(yassinAES_BASEADDR, 11 * 0x04));
    printf("template3 0x%X 0x%X 0x%X 0x%X\n",
        IORD_32DIRECT(yassinAES_BASEADDR, 11 * 0x04),
        IORD_32DIRECT(yassinAES_BASEADDR, 10 * 0x04),
        IORD_32DIRECT(yassinAES_BASEADDR, 9 * 0x04),
        IORD_32DIRECT(yassinAES_BASEADDR, 8 * 0x04));

    PERF_END(PERF_CNTR_BASE, 1); // Fin de la mesure de performance

    unsigned int section_time = perf_get_section_time(PERF_CNTR_BASE, 1);
    printf("Temps de la section mesuree : %u cycles\n", section_time);

    alt_putstr("Hello from Nios II!\n");

    while (1)
    {
    }

    return 0;
}

// Fonctions de mesure de performance
void PERF_BEGIN(unsigned int base_addr, unsigned int counter)
{
    // Assurez-vous que le compteur est reinitialise en premier
    IOWR_32DIRECT(base_addr, counter * 0x04, 0);

    // Commencez la mesure de performance en activant le compteur
    IOWR_32DIRECT(base_addr, counter * 0x04 + 1, 1);
}
// Fonctions de mesure de performance
void PERF_END(unsigned int base_addr, unsigned int counter)
{
    // Arretez la mesure de performance en desactivant le compteur
    IOWR_32DIRECT(base_addr, counter * 0x04 + 1, 0);
}
 // Fonctions de mesure de performance
unsigned int perf_get_section_time(unsigned int base_addr, unsigned int counter)
{
    // Lisez la valeur actuelle du compteur
    unsigned int counter_value = IORD_32DIRECT(base_addr, counter * 0x04);

    // Retournez la valeur du compteur, qui represente le temps ecoule pendant la section mesuree
    return counter_value;
}
```