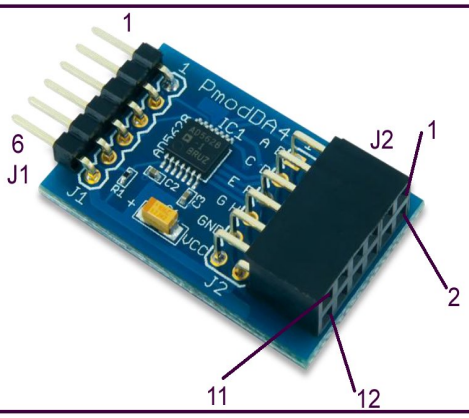
Utilisation d'un module PMOD CAN en mode SPI

1) Présentation

On souhaite lors de ce TP développé un CNA 8 canaux. Le CNA 8 Canaux est intégré dans un module pmodDA4 qui dispose d'un convertisseur numérique analogique de chez analog devices le AD5628.
Ce module pmodDA4 est distribué par Digilent est à le format suivant :



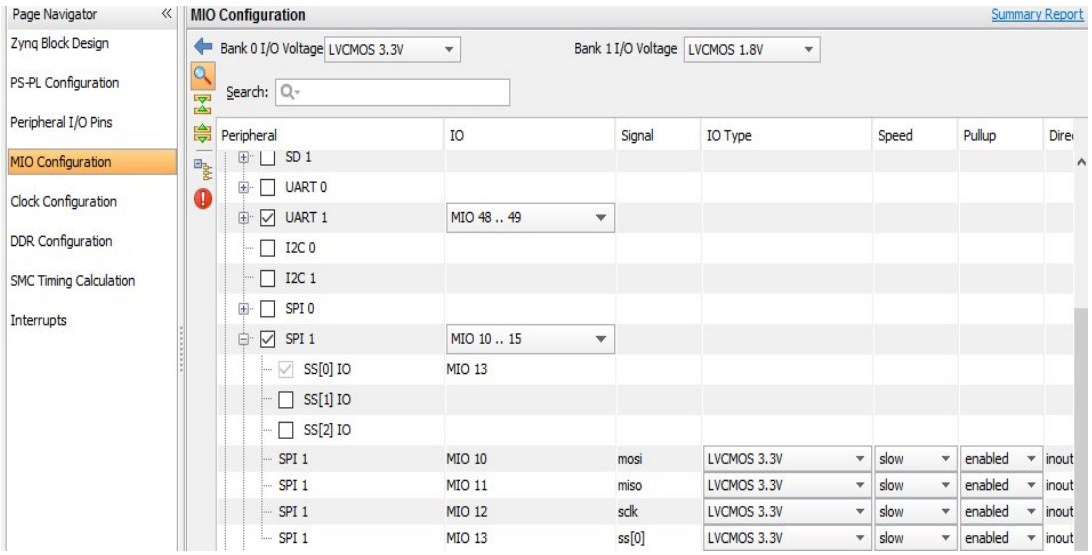
Digital Interface – J1	
1	SYNC
2	DIN
3	Not Connected (NC)
4	SCLK
5	GND
6	VCC

Analog Interface – J2	
1	VOUTA
2	VOUTB
3	VOUTC
4	VOUTD
5	VOUTE
6	VOUTF
7	VOUTG
8	VOUTH
9, 10	GND
11, 12	VCC

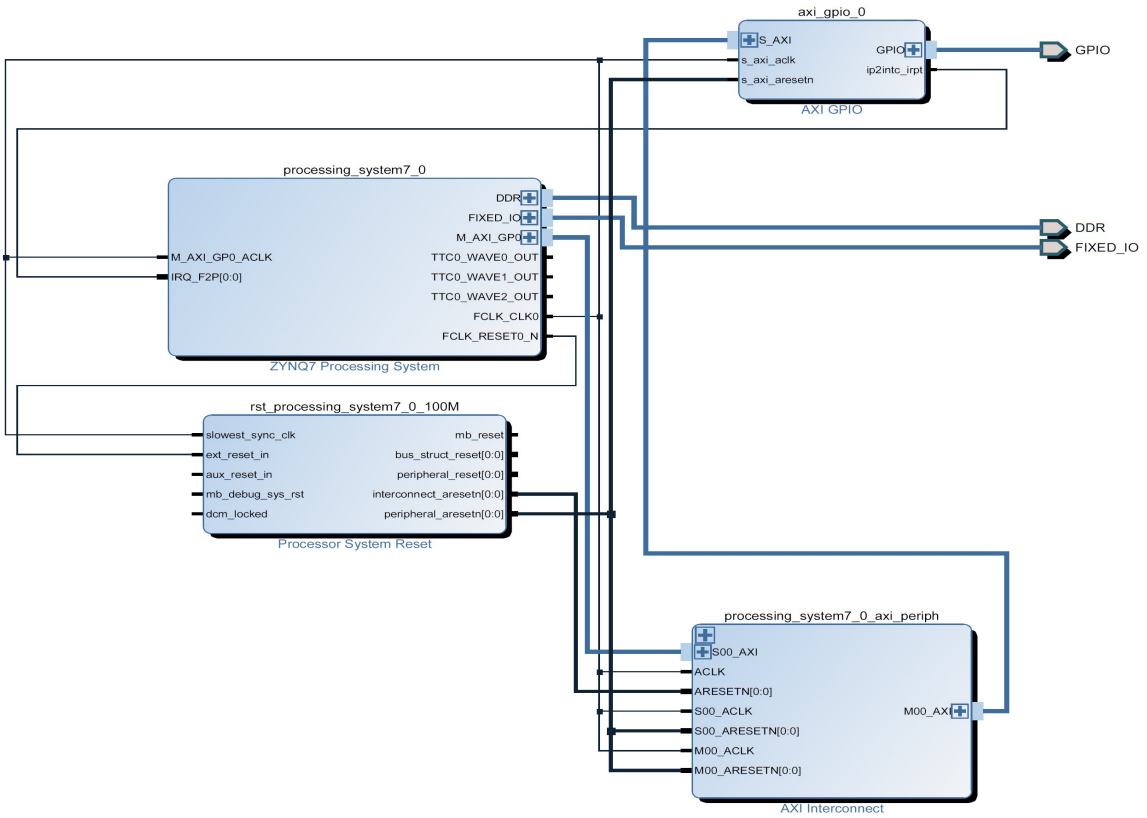
Le connecteur J1 dispose d'une interface SPI afin d'être programmé par le ZYNQ, l'interface J2 nous permet de sortir les 8 canaux numérisés (VOUTA ….. VOUTH).
L'objectif est donc de programmer avec la carte ZYBO et une de ses interfaces SPI, le module pmodDA4 afin de générer les huit sinusoïdes. Le contrôleur SPI fera partie du PS

2) Mise en place matérielle du projet sous Vivado

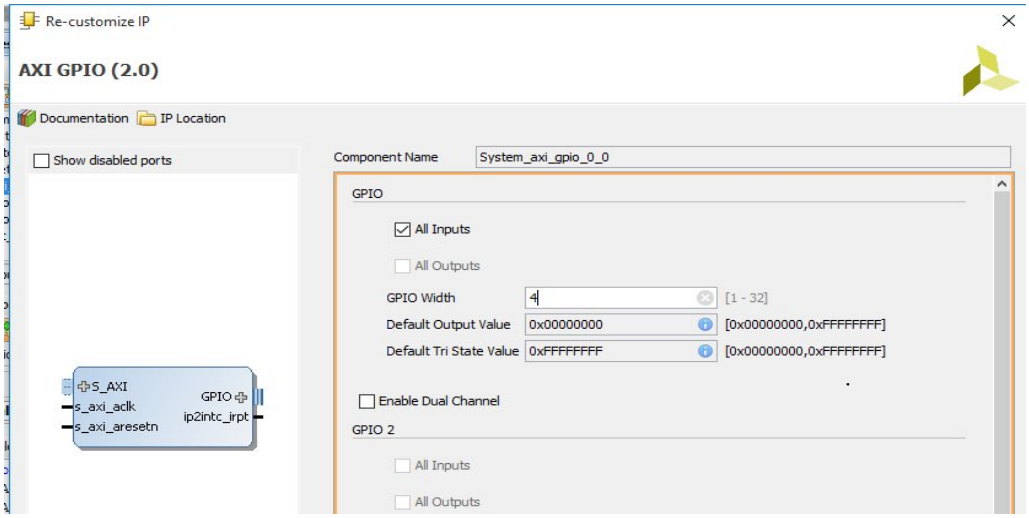
La procédure reste habituelle pour créér un projet et on donne page suivante le schéma à saisir. Lors de la configuration du PS on cochera l'UART1 et le SPI1 du PS



Les broches du SPI1 et de l'UART1 n'apparaissent pas sur le schéma (processing system 7) car elles font partie du MIO



On n'aura pas besoin de modifier le fichier de contrainte pour le SPI1 et l'UART1. Les broches du MIO sont associées sur la carte ZYBO à des broches bien précises du SOC_FPGA
Le GPIO pourra comporter quatres broches en entrée.



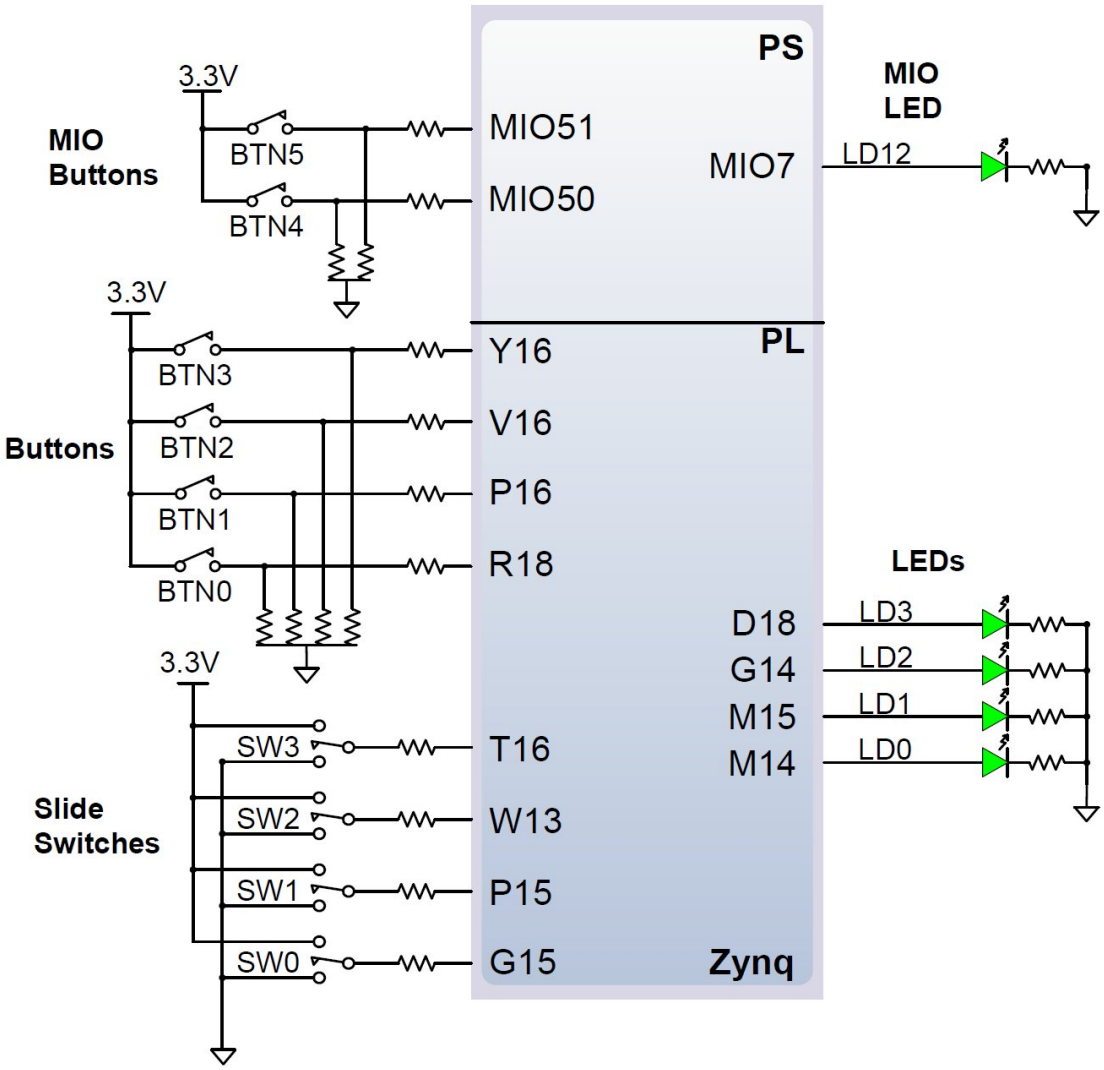
Le fichier de contrainte devra prendre en compte les broches du GPIO, en associant chaque broche du GPIO programmé en entrée aux bouttons de la carte zybo

```
# Buttons
set_property IOSTANDARD LVCMOS33 [get_ports {gpio_tri_i[0]}]
set_property PACKAGE_PIN Y16 [get_ports {gpio_tri_i[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports {gpio_tri_i[1]}]
set_property PACKAGE_PIN V16 [get_ports {gpio_tri_i[1]}]

set_property IOSTANDARD LVCMOS33 [get_ports {gpio_tri_i[2]}]
set_property PACKAGE_PIN P16 [get_ports {gpio_tri_i[2]}]

set_property IOSTANDARD LVCMOS33 [get_ports {gpio_tri_i[3]}]
set_property PACKAGE_PIN R18 [get_ports {gpio_tri_i[3]}]
```

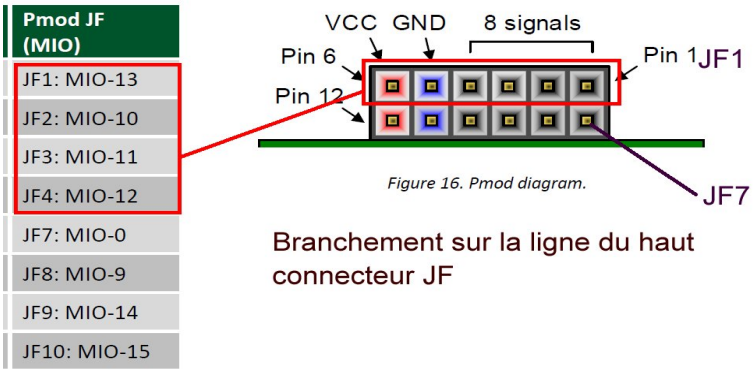


3) Développement logiciel sous SDK

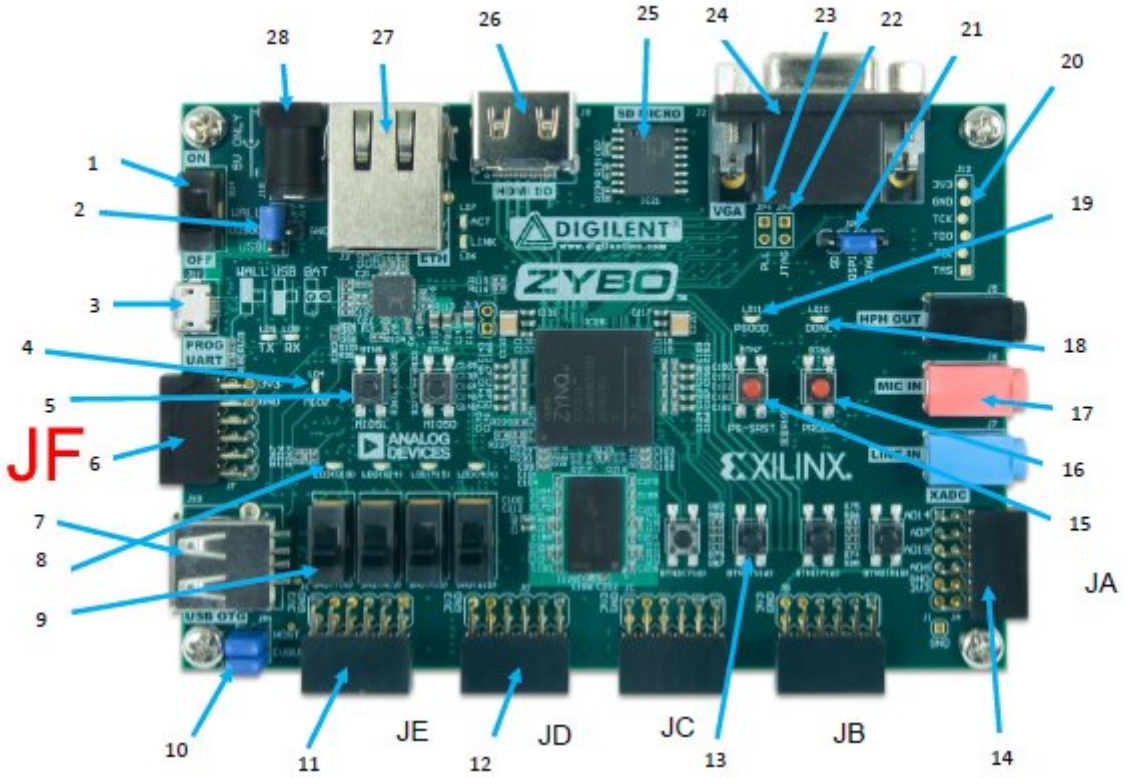
La procédure reste habituelle aux autres projets
Les broches du périphérique SPI du PS sont branchées de la façon suivante :

SPI 1	MIO 10	mosi	LVCMOS 3.3V	slow	enabled	inout
SPI 1	MIO 11	miso	LVCMOS 3.3V	slow	enabled	inout
SPI 1	MIO 12	sclk	LVCMOS 3.3V	slow	enabled	inout
SPI 1	MIO 13	ss[0]	LVCMOS 3.3V	slow	enabled	inout

Le module sera branché de la façon suivante :



Le connecteur JF est sur le coté de la carte ZYBO :



Le développement logiciel découle du fonctionnement interne de CNA AD5628.
Les fonctions de haut niveau (API) du SPI pour le PS (processing system) se trouve sur le WEB :

Xilinx SDK Drivers API Documentation
http://www.wiki.xilinx.com/Standalone+Drivers+and+Libraries

Ou dans un fichier zippé sur github :

https://github.com/Xilinx/embeddedsw

On retiendra les APIs encadrées en rouge :

s32	**XSpiPs_CfgInitialize** (**XSpiPs** *InstancePtr, **XSpiPs_Config** *ConfigPtr, u32 EffectiveAddr)
void	**XSpiPs_Reset** (**XSpiPs** *InstancePtr)
s32	**XSpiPs_Transfer** (**XSpiPs** *InstancePtr, u8 *SendBufPtr, u8 *RecvBufPtr, u32 ByteCount)
s32	**XSpiPs_PolledTransfer** (**XSpiPs** *InstancePtr, u8 *SendBufPtr, u8 *RecvBufPtr, u32 ByteCount)
s32	**XSpiPs_SetSlaveSelect** (**XSpiPs** *InstancePtr, u8 SlaveSel)
u8	**XSpiPs_GetSlaveSelect** (**XSpiPs** *InstancePtr)
void	**XSpiPs_SetStatusHandler** (**XSpiPs** *InstancePtr, void *CallBackRef, **XSpiPs_StatusHandler** FunctionPtr)
void	**XSpiPs_InterruptHandler** (**XSpiPs** *InstancePtr)
void	**XSpiPs_Abort** (**XSpiPs** *InstancePtr)
XSpiPs_Config *	**XSpiPs_LookupConfig** (u16 DeviceId)
s32	**XSpiPs_SelfTest** (**XSpiPs** *InstancePtr)
s32	**XSpiPs_SetOptions** (**XSpiPs** *InstancePtr, u32 Options)
u32	**XSpiPs_GetOptions** (**XSpiPs** *InstancePtr)
s32	**XSpiPs_SetClkPrescaler** (**XSpiPs** *InstancePtr, u8 Prescaler)
u8	**XSpiPs_GetClkPrescaler** (**XSpiPs** *InstancePtr)
s32	**XSpiPs_SetDelays** (**XSpiPs** *InstancePtr, u8 DelayNss, u8 DelayBtwn, u8 DelayAfter, u8 DelayInit)
void	**XSpiPs_GetDelays** (**XSpiPs** *InstancePtr, u8 *DelayNss, u8 *DelayBtwn, u8 *DelayAfter, u8 *DelayInit)
void	**XSpiPs_ResetHw** (u32 BaseAddress)

Le programme est le suivant :

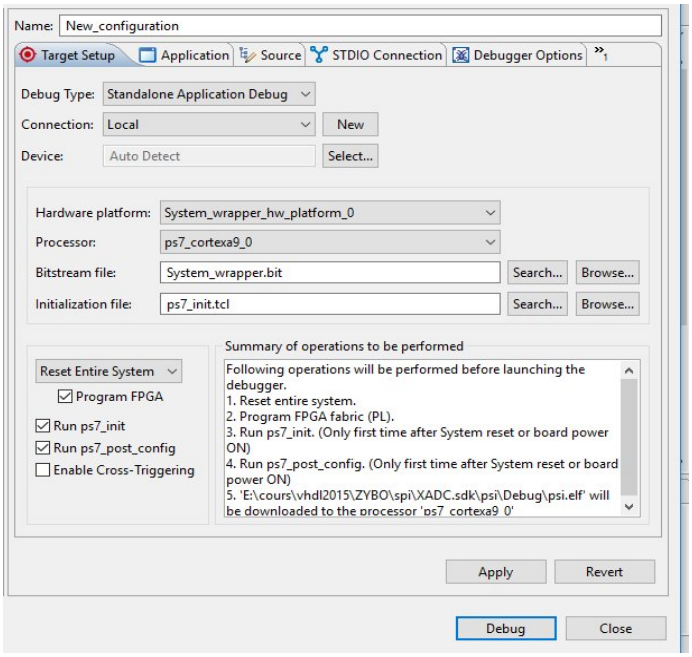
```
#include <stdio.h>
#include "xparameters.h"
#include "xil_cache.h"
#include "xscugic.h"
#include "xil_exception.h"
#include "scugic_header.h"
#include "xdevcfg.h"
#include "devcfg_header.h"
#include "xdmaps.h"
#include "dmaps_header.h"
#include "xgpio.h"
#include "gpio_header.h"
#include "gpio_intr_header.h"
#include "xscutimer.h"
#include "scutimer_header.h"
#include "xscuwdt.h"
```

```
#include "scuwdt_header.h"
#include "xspips.h"
#include "spips_header.h"
#include "xttcps.h"
#include "ttcps_header.h"
#define GPIO_CHANNEL1 1
#define SPI_DEVICE_ID XPAR_XSPIPS_0_DEVICE_ID
static XSpiPs SpiInstancePtr;
XSpiPs_Config *SpiConfigPtr;
//---------------------------------- MIO/EMIO
#define delay_us usleep
const unsigned int sinus_data[32]={
2048,2447,2831,3185,3496,3750,3940,4056,
4095,4056,3940,3750,3496,3185,2831,2447,
2048,1649,1265,911,600,346,156,40,
0,40,156,346,600,911,1265,1649
};
u8 SendBuf[4];

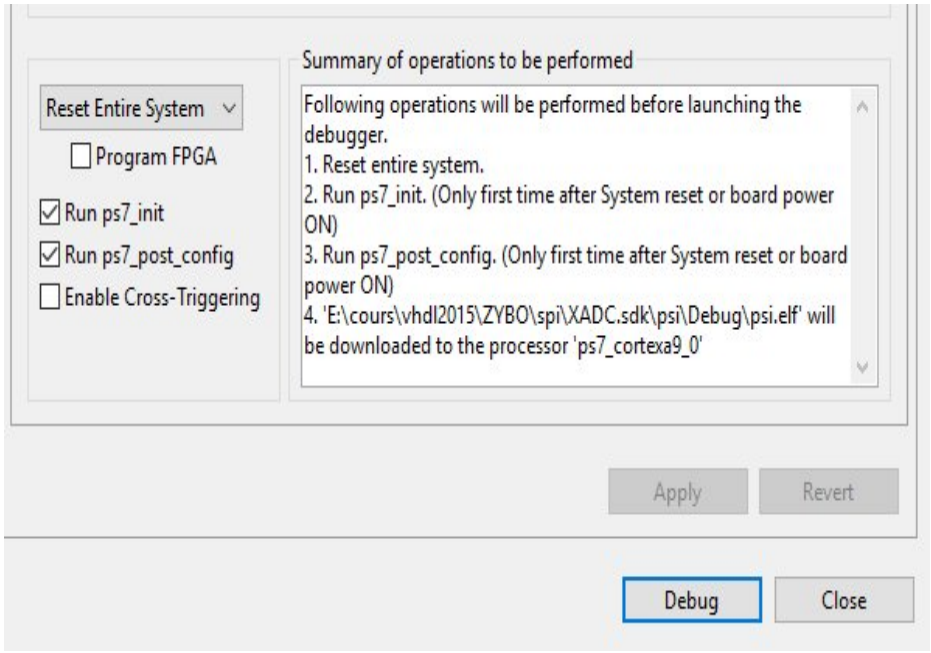

int main()
{
   static XScuGic intc;
   static XGpio axi_gpio_0_Gpio;
   static XScuTimer ps7_scutimer_0;
   static XScuWdt ps7_scuwdt_0;
   static XTtcPs ps7_ttc_0;
   int Status,i;
   Xil_ICacheEnable();
   Xil_DCacheEnable();
   print("---Entering main---\n\r");
   SpiConfigPtr = XSpiPs_LookupConfig(SPI_DEVICE_ID);
   Status = XSpiPs_CfgInitialize(&SpiInstancePtr, SpiConfigPtr, SpiConfigPtr->BaseAddress);
…….. A compléter

}
```

Lors de la mise en place du debugger cochez les options suivantes :



Par la suite à chaque modification logicielle, fermez le debugger et lors de la relance du debugger la programmation du FPGA n'est plus nécessaire :



Ceci évite à chaque modification logicielle sous SDK de reprogrammer le FPGA et de solliciter à nouveau une programmation.
Enfin les résultats sur une des voies ou un canal et le signal MOSI en dessous :

