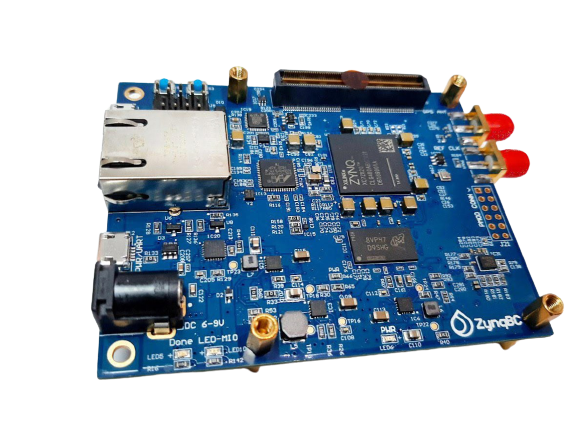
**Using the DAC on ConnectBOB mezzanine board of ZynqBC**



# Overview

This guide will provide a step-by-step walk-through of creating a hardware design using the Vivado IP Integrator for the zynqBC and running a BME280 sensor.

At the end of this tutorial, you will have:

* Created a hardware design for BME sensor.
* Created a .C project in XIlinx Vivado SDK and running the BME functionality using the given functions in C

## **Prerequisites**

#### **Hardware**

* **zynqBC Development Board and JTAG programmer for programming ZYNQ board**

#### **Software**

* **Xilinx Vivado 2020.2 with the vitis.**

#### **Board Support Files**

* **zynqBC Support Files**
  + Follow this link to download the board files from the internet and copy them in the board files folder i.e.
    - *C:\Xilinx\Vivado\2020.2\data\boards\board\_files*

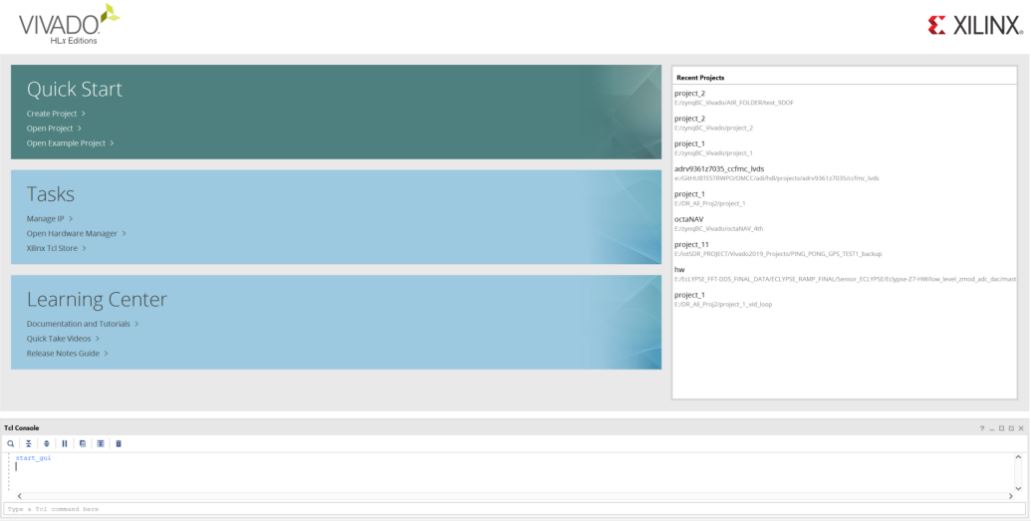
*or*

* + - *E:\Xilinx\Vivado\2020.2\data\boards\board\_files*

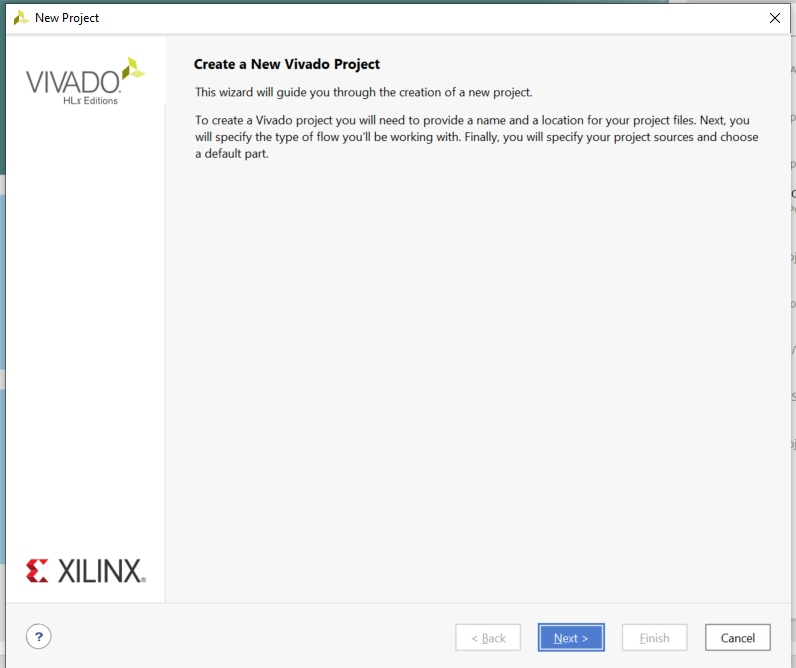
### **1. Creating a New Project**

When you first run Vivado this will be the main start window where you can create a new project or open a recent one.

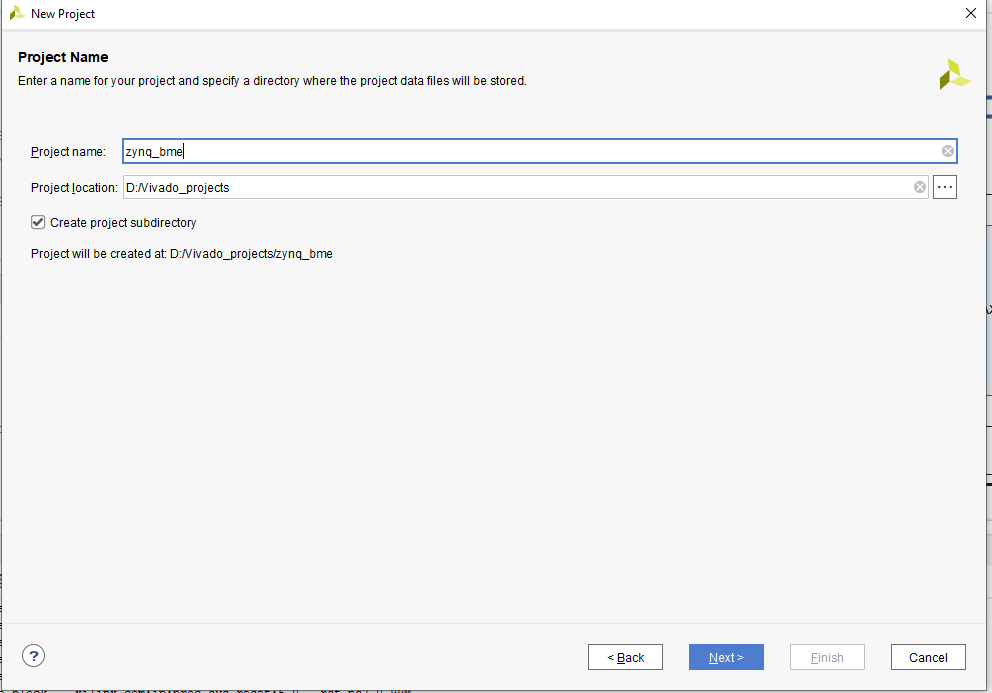
* 1. Click on Create New Project.



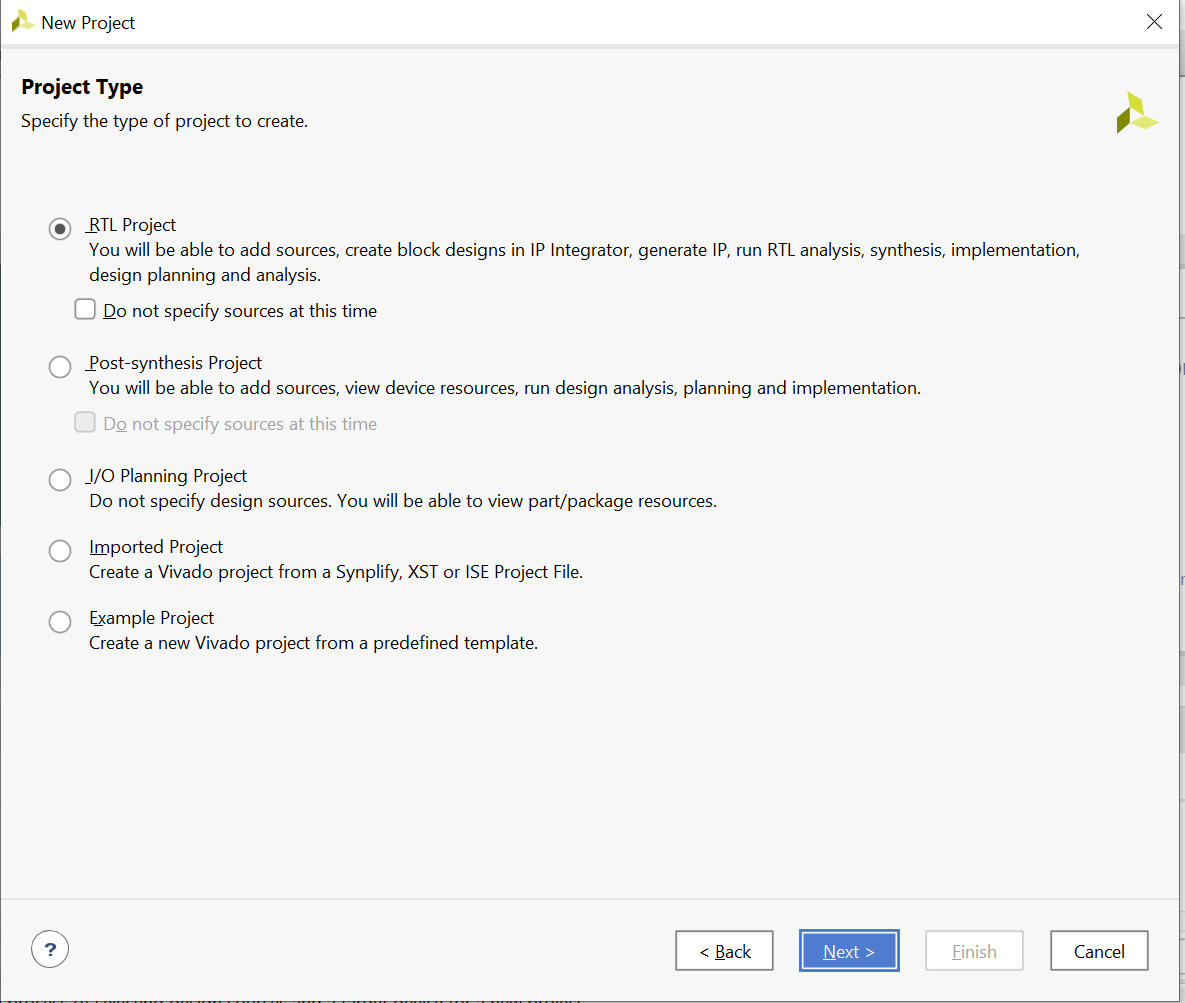
* 1. You will be presented with the project creation wizard. Click Next.



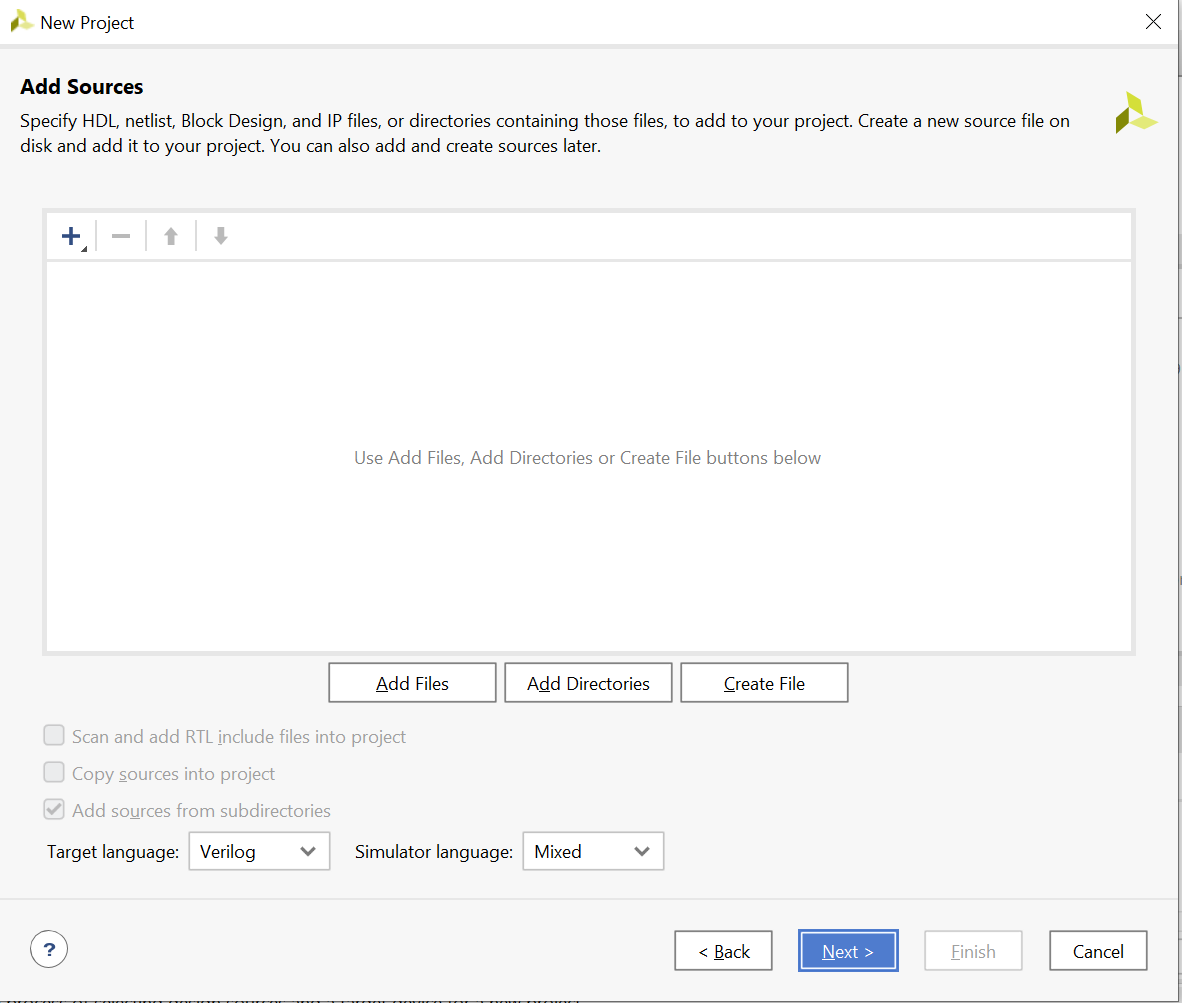
* 1. Enter a project name and location the click **Next**.



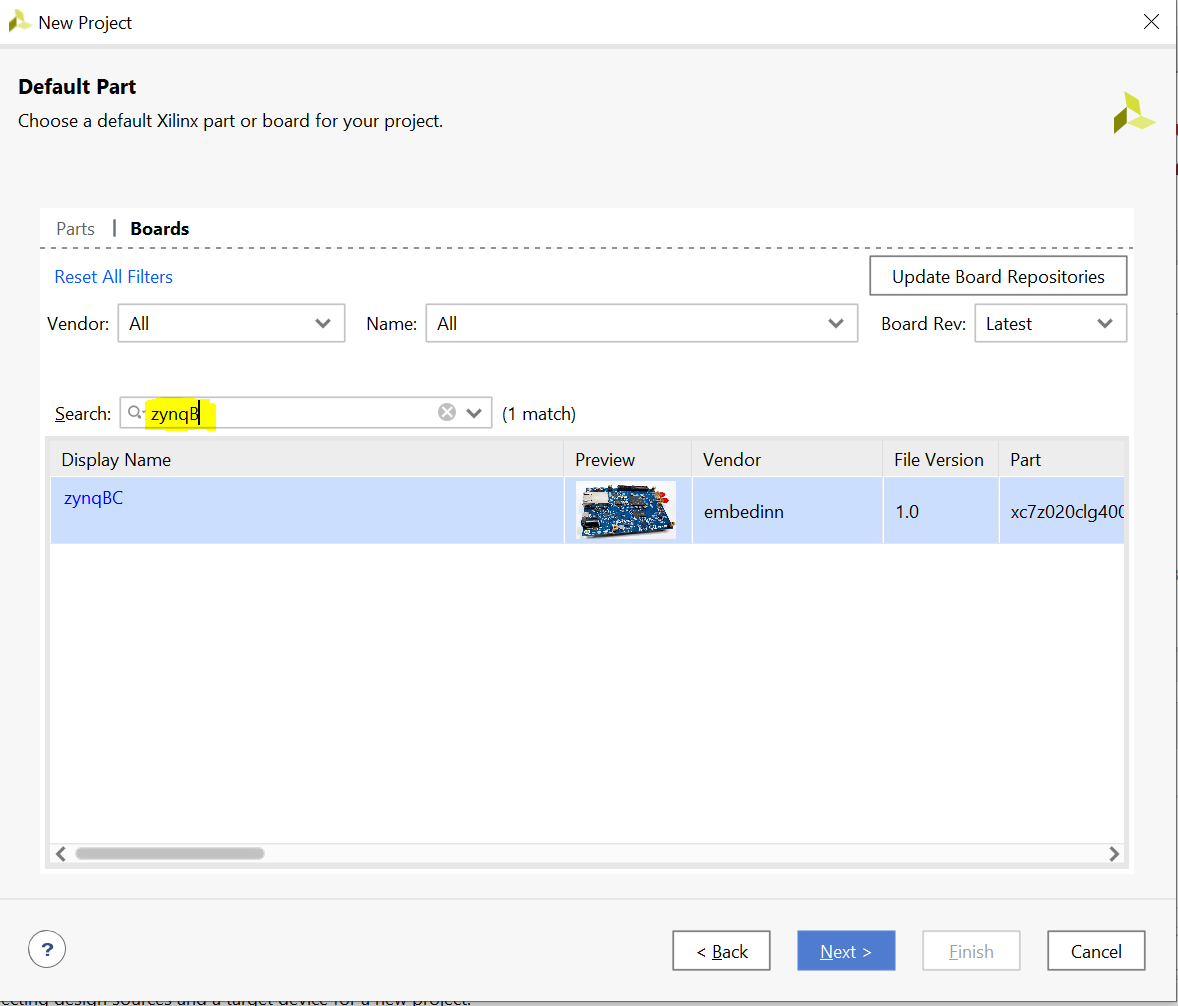
* 1. Select **RTL Project** and click **Next**.



* 1. This demo does not use any existing sources, existing IP or constraints. Click through the next three screens.

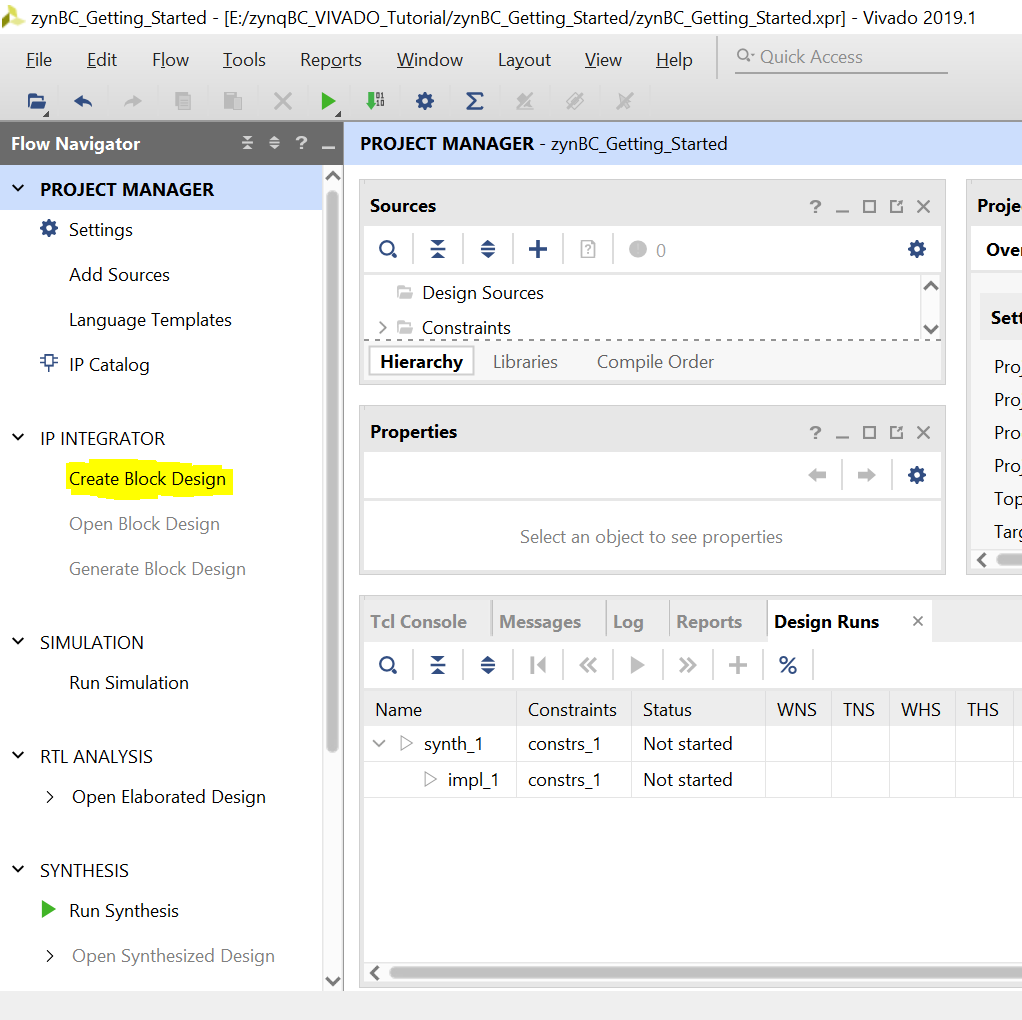


* 1. Select Boards and select the zynqBC board file. Click Next and then Finish. Just make sure before this step that board files are placed already.

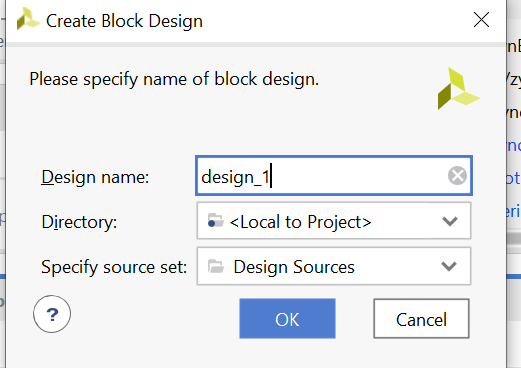


### **Creating a New Block Design**

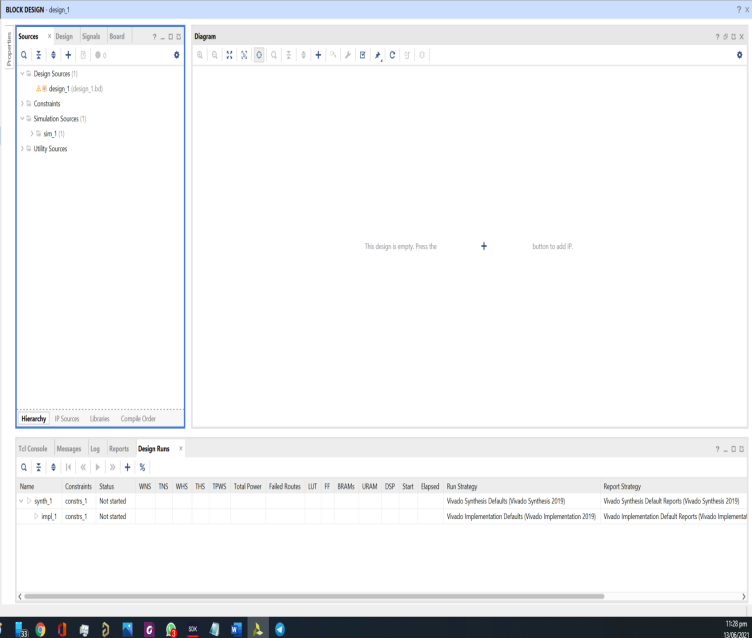
* 1. Once the process has completed, click **Create Block Design** in the flow navigator.



* 1. Click Ok

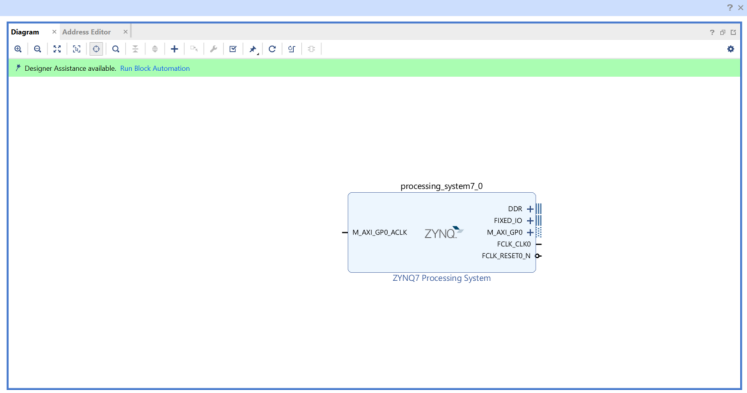


* 1. A blank Block Design will open up.

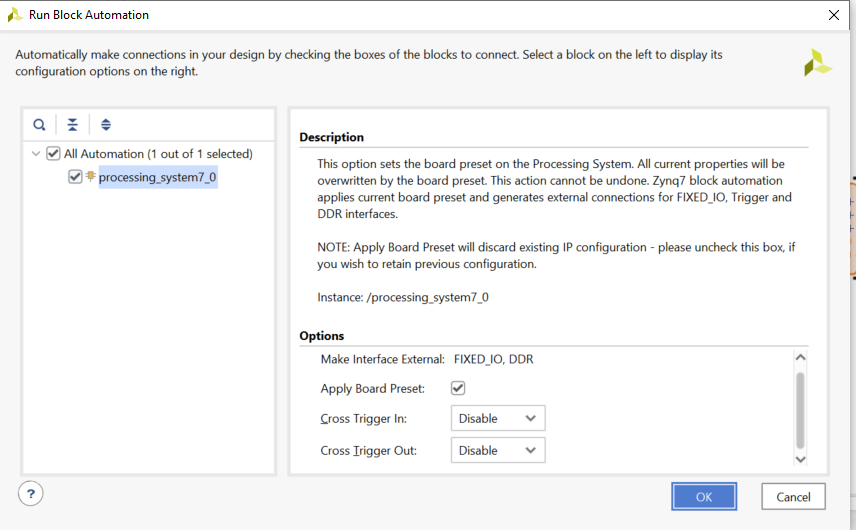


### **Add the Zynq IP & SPI Blocks**

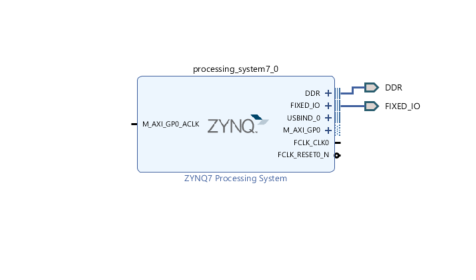
* 1. Click the  **Add IP** button and search for ZYNQ. Double click on **ZYNQ 7 Processing System** to place the bare Zynq block.



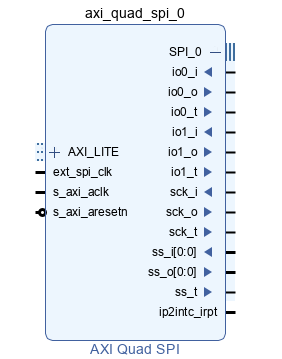
* 1. Click the **Run Block Automation** link



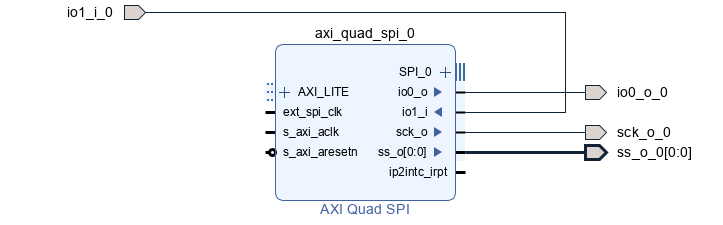
Your Zynq block should now look like the picture below.



* 1. Click on **ADD IP** button and search for axi\_quad\_spi to place it in design. In SPI block expand SPI\_0. The block diagram look like this.

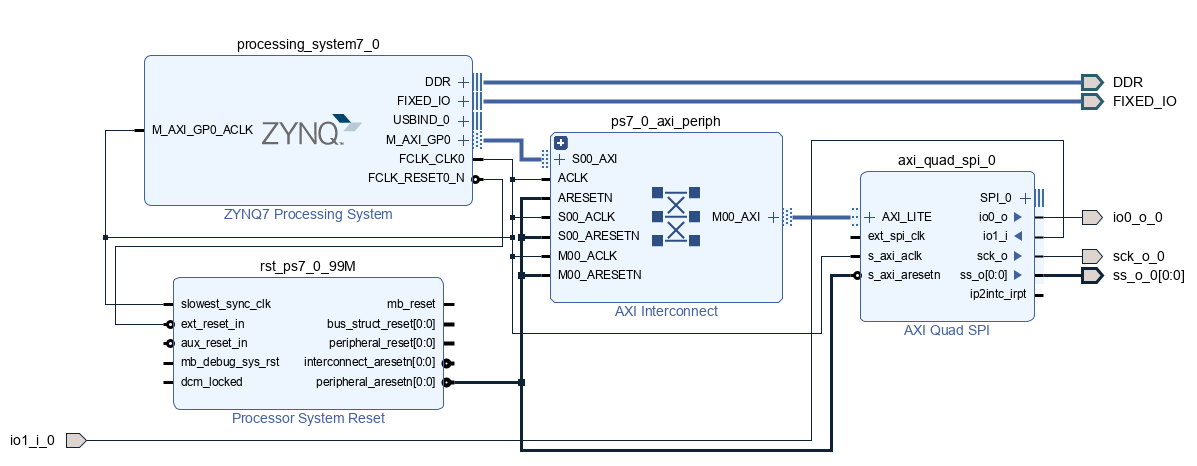


* 1. In this SPI block right click on io0\_o and click on make it external. Similarly do this for io1\_i, sck\_o and ss\_o[0:0]. Unexpand SPI\_0, block diagram look like this.



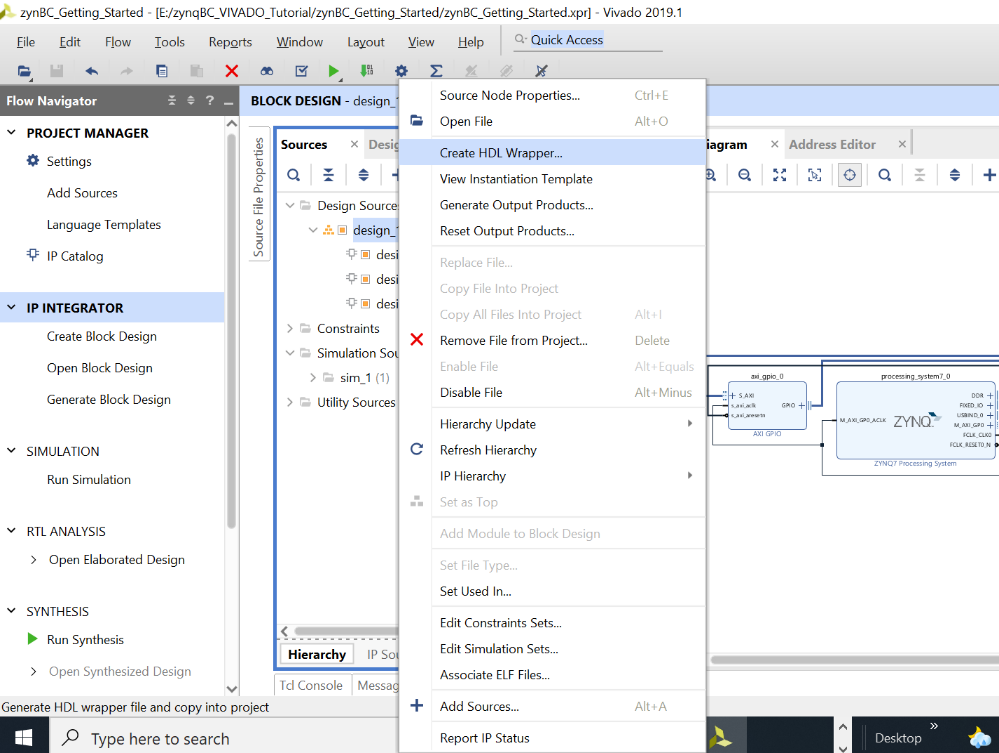
Now click on **Run Connection Automation** again and make sure to check All automation box. **Press OK**

* 1. Your new block design must look like this



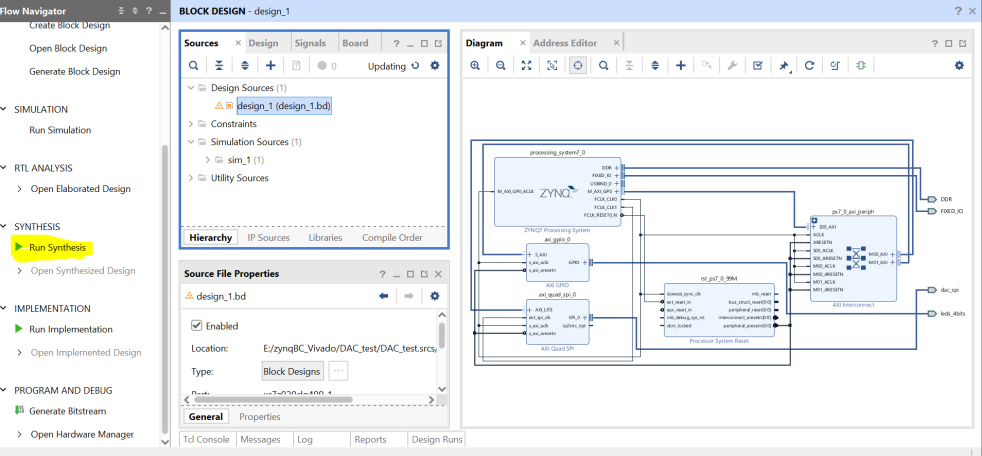
### **Generate HDL Wrapper and Validate Design**

* 1. Select  Validate Design. This will check for design and connection errors.
  2. After the design validation step, we will proceed with creating a HDL System Wrapper. In the block design window, under the **Design Sources** tab, right-click on the block diagram file. We labelled it “design\_1.bd” and select **Create HDL Wrapper**.

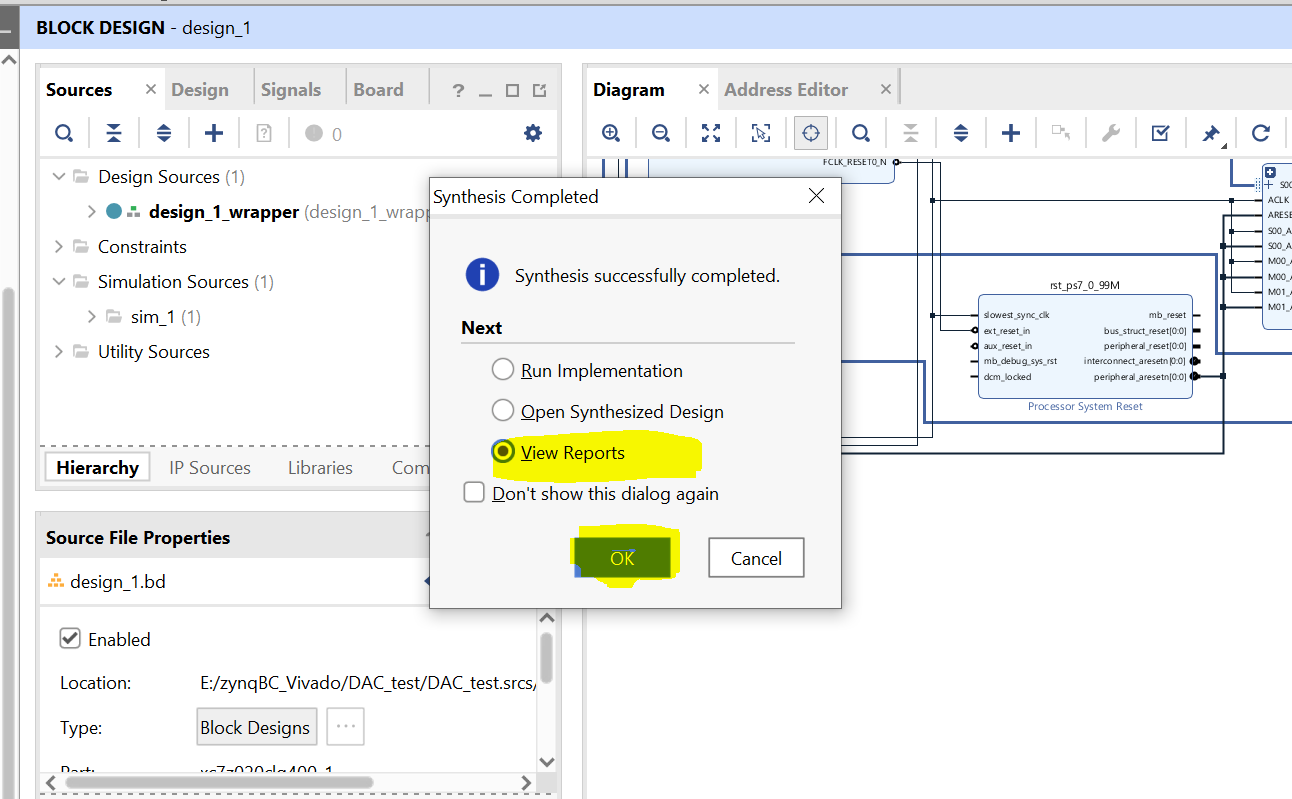


This will create a top module in Verilog and will allow you to generate a bitstream. But before that we need to set the XDC file for the GPIO which is added for LDAC

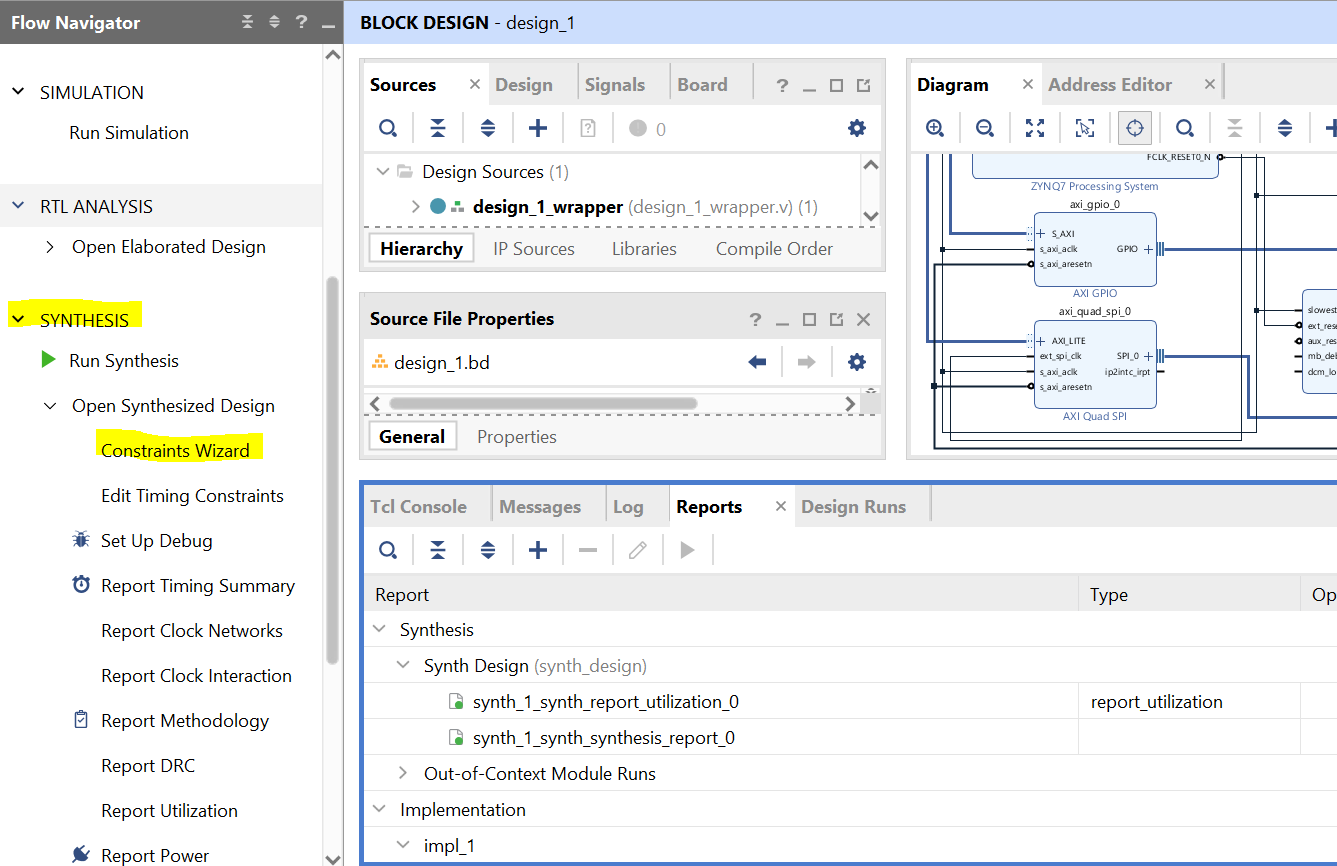
1. **Generate Synthesis**
   1. Click on Run Synthesis



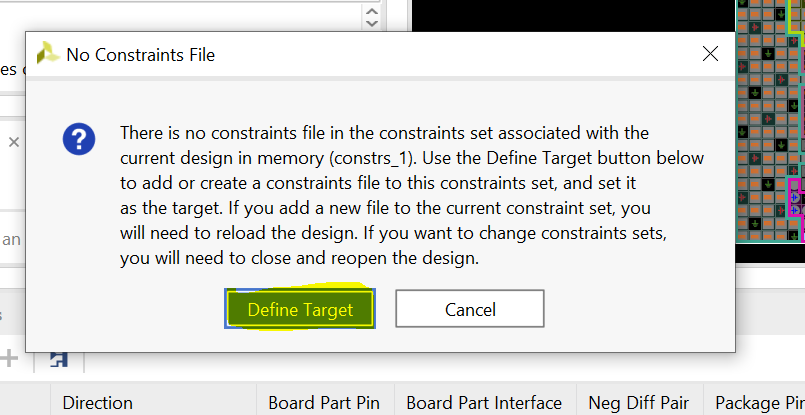
* 1. Once Completed, a message box will appear. Check **View Report** and Press OK

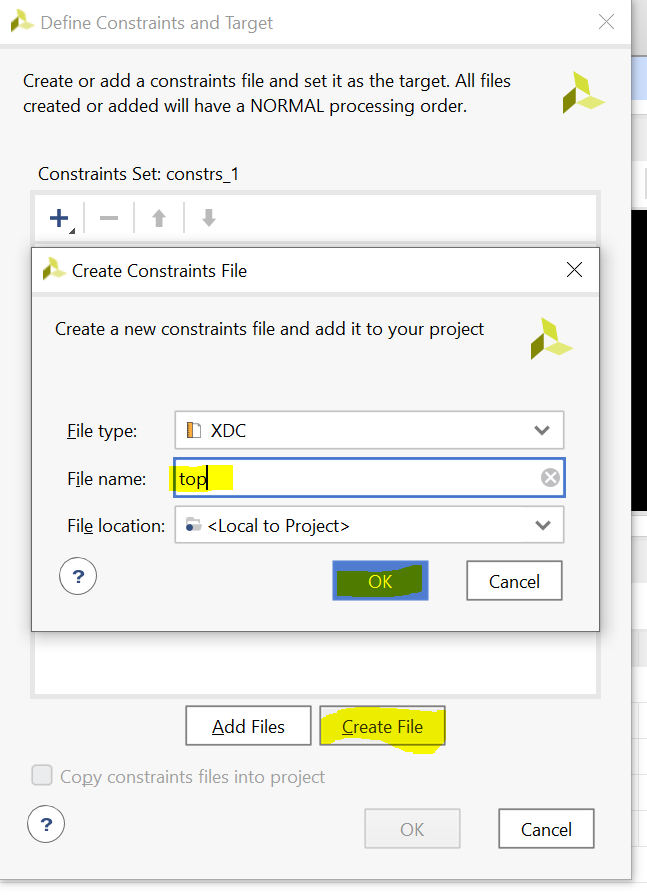


* 1. Click on **Constraints Wizards** under Synthesis

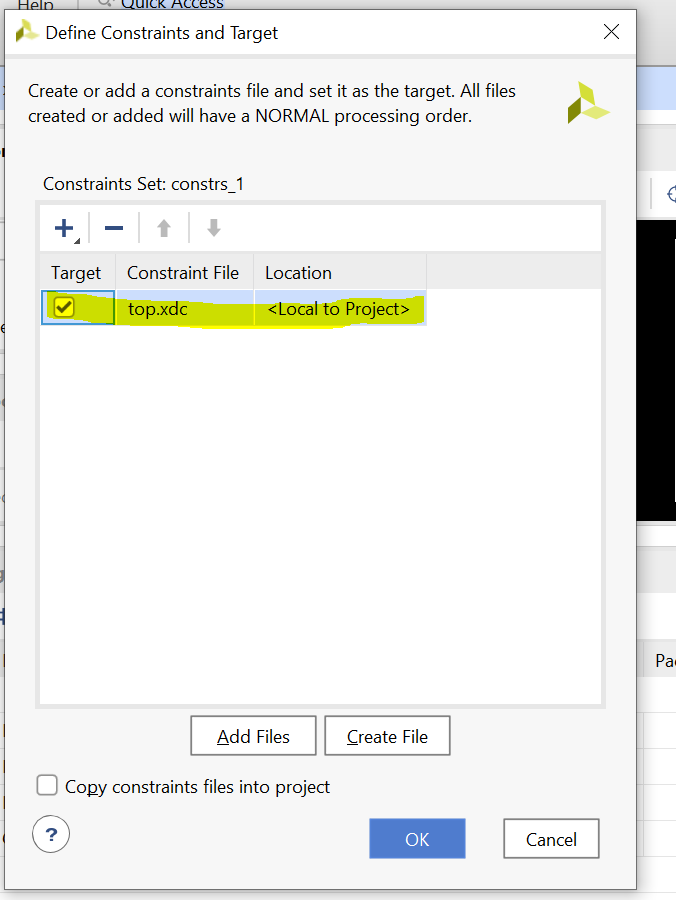


* 1. Pop up will appear with No Constraints file. Click Define Target 🡪 Create File and give top as file name

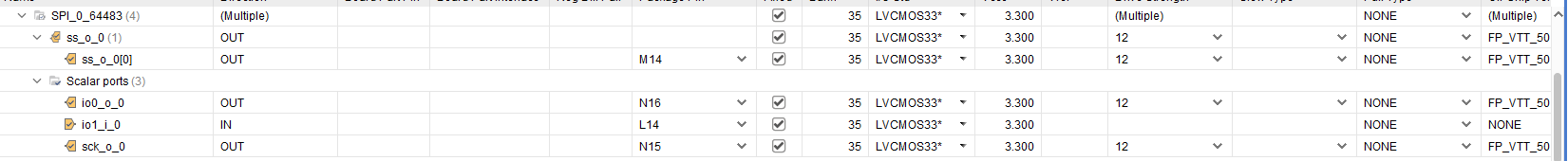




* 1. Check the top.xdc block and Press **OK**

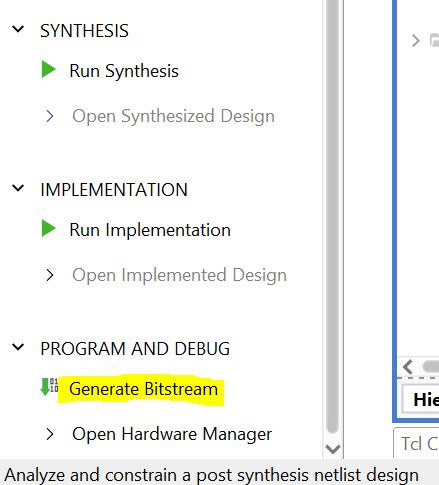


* 1. Under IO Ports, select the pin for SPI. The voltage standard will be LVCMOS33 and pins configuration is shown in figure below.

****

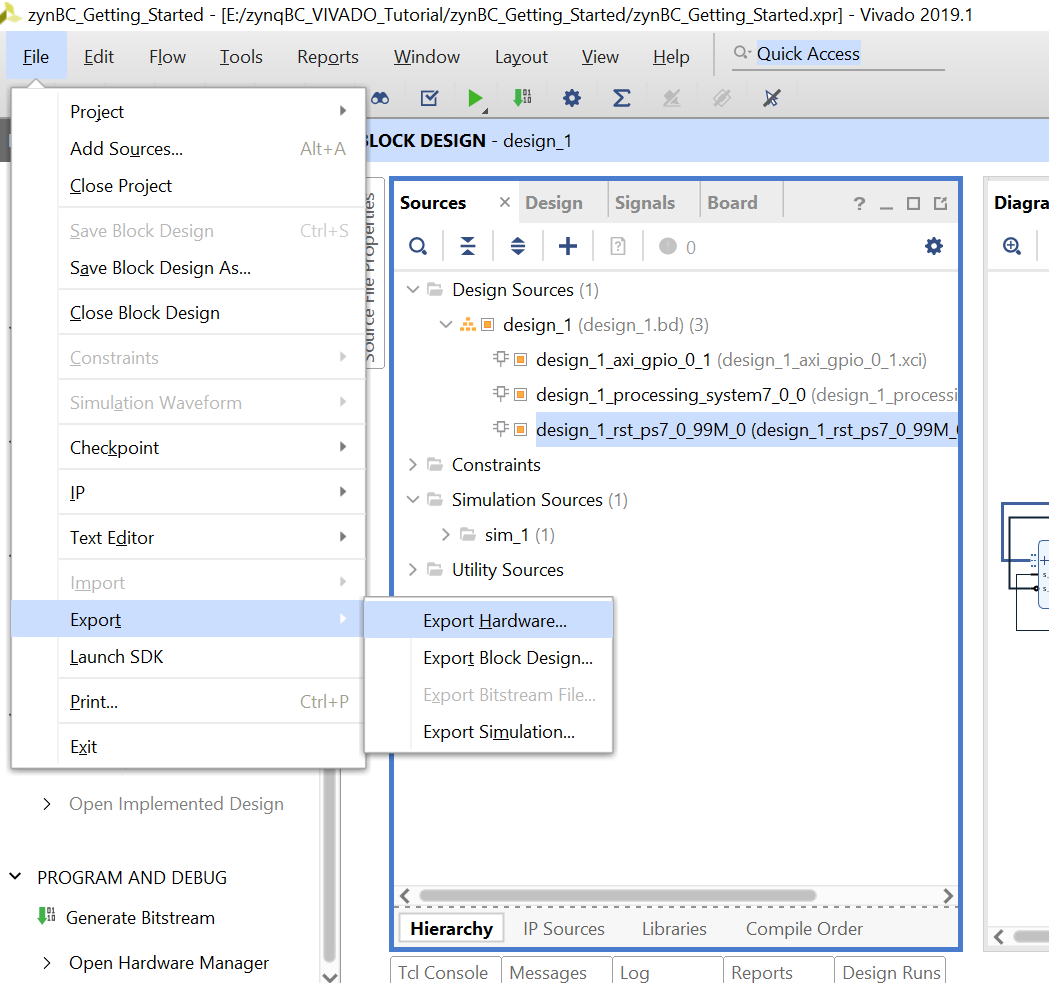
* 1. Press CTRL+S to save the xdc file settings and press **Ok**

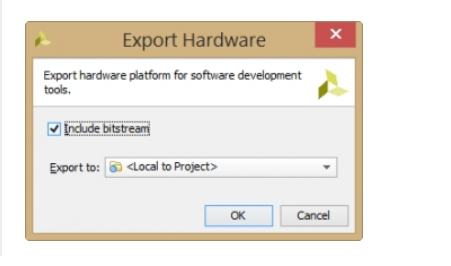
1. **Generate the Bitstream**
   1. Click on **Generate Bitstream** at the bottom of the Flow Navigator. Wait for the process to complete and click OK.



### **Export hardware files for SDK**

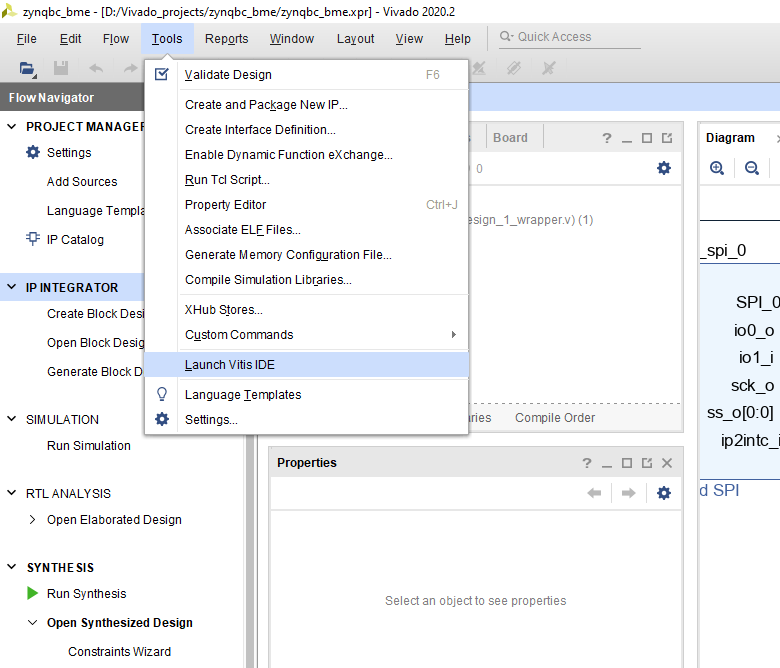
* 1. Go to file→Export→Export Hardware… Make sure to check the box for **Include bitstream** then click **OK**.



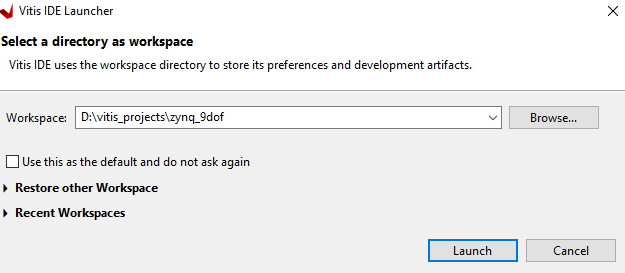


### **Launch Vitis Ide**

* 1. Once bitstream is generated, Go to **Tools→Launch Vitis** IDE and click **OK**.

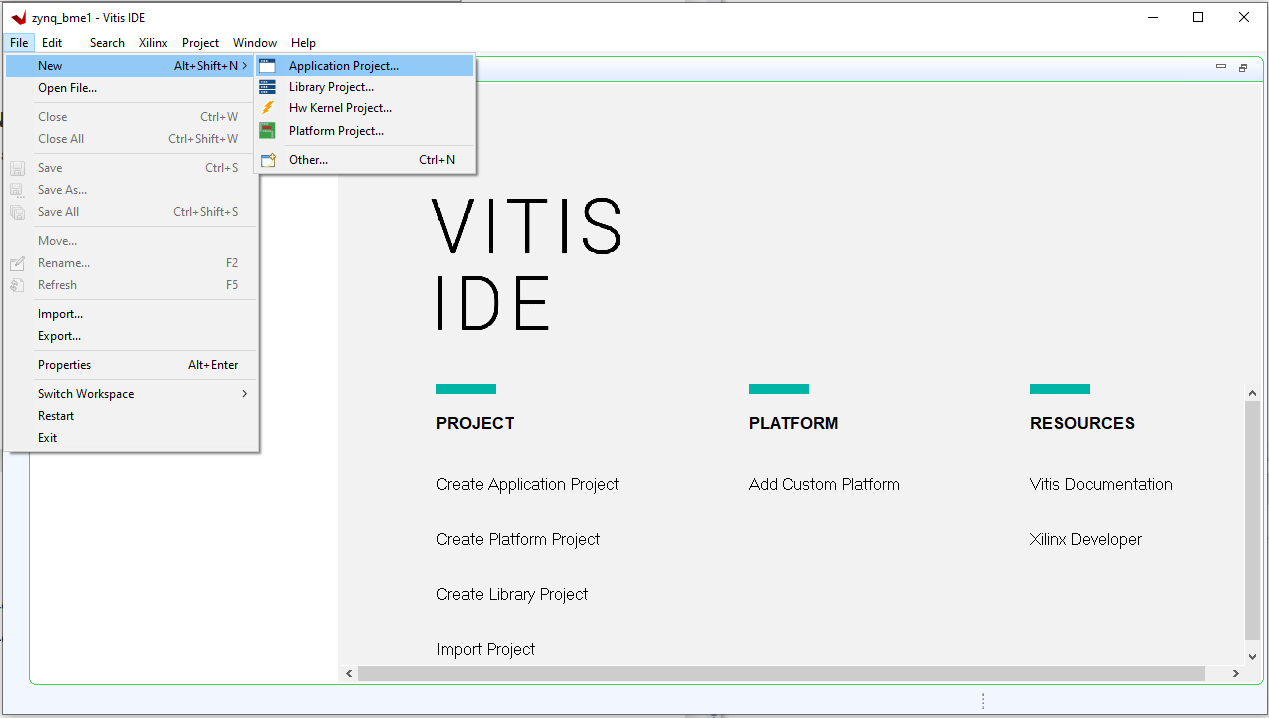


* 1. Set your worksapce. And click on Launch.

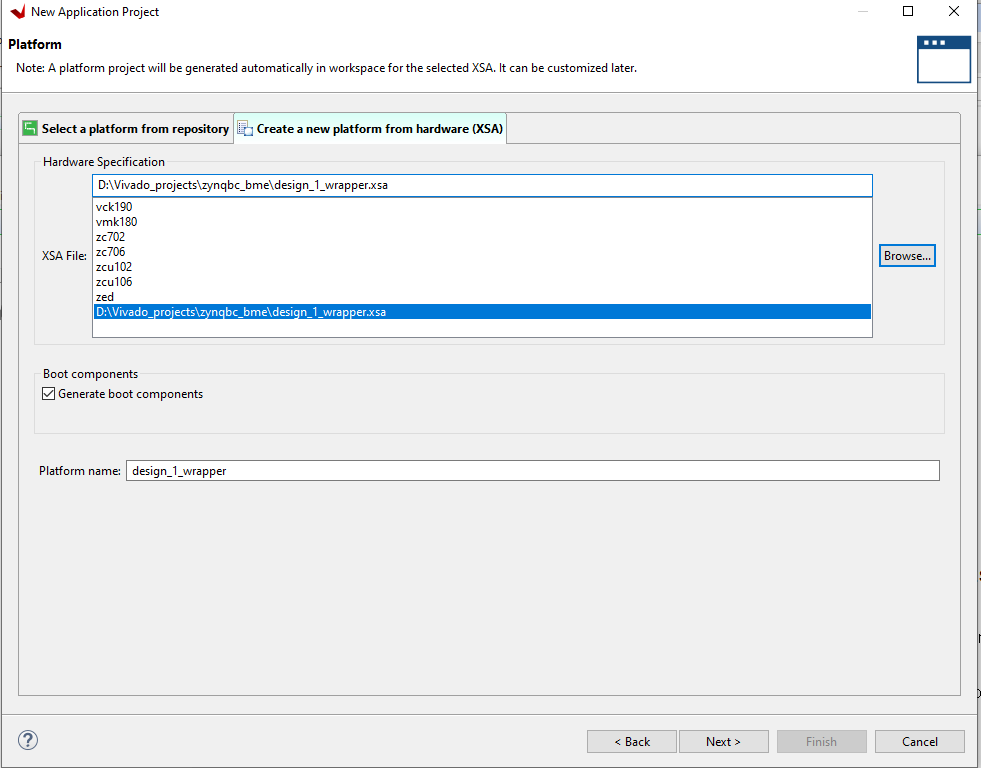


### **Create an Empty Hello World Application on Vitis Ide**

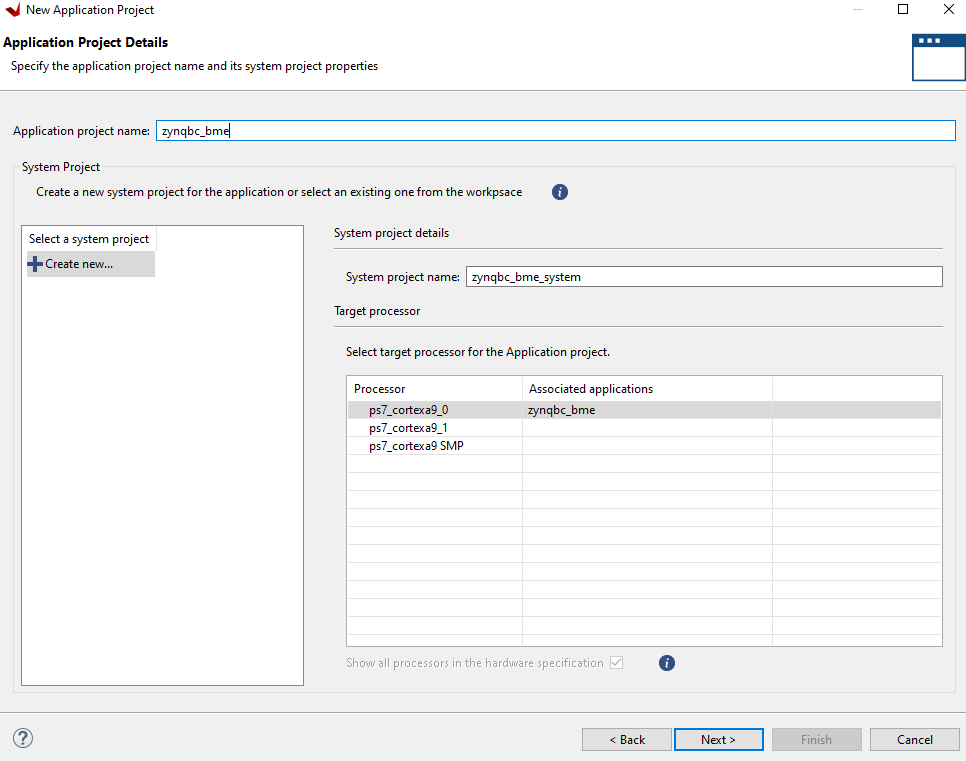
* 1. Click on File and Create an Application Project. Click on next.



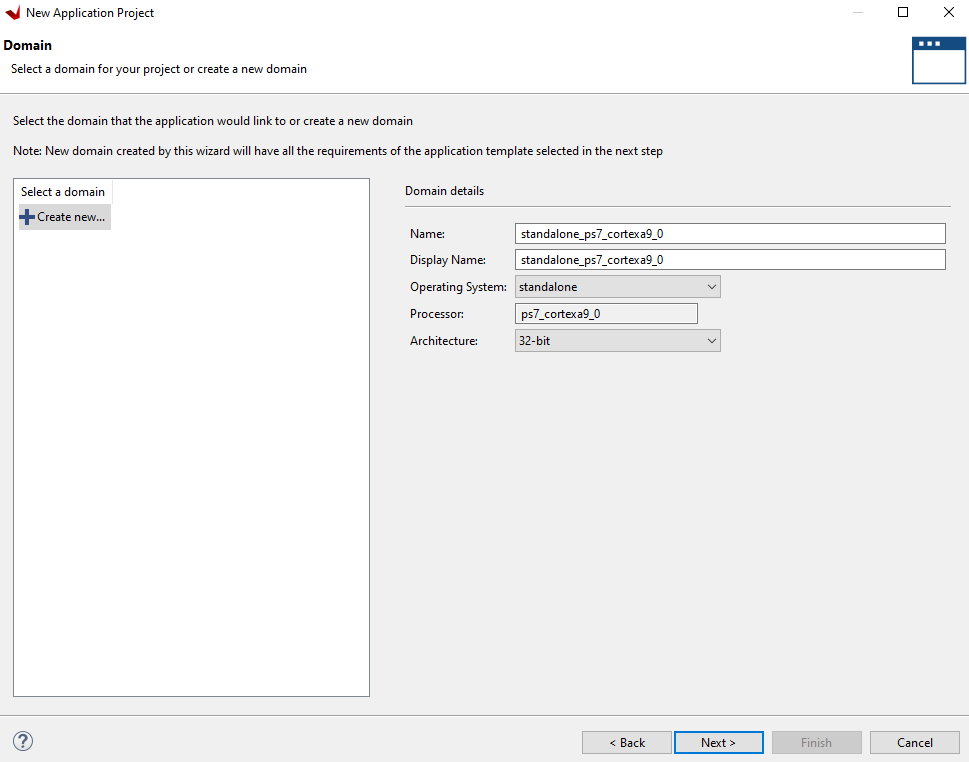
* 1. Go to **create a new platform hardware(XSA)** tab and browse for your design file. Then click Next.



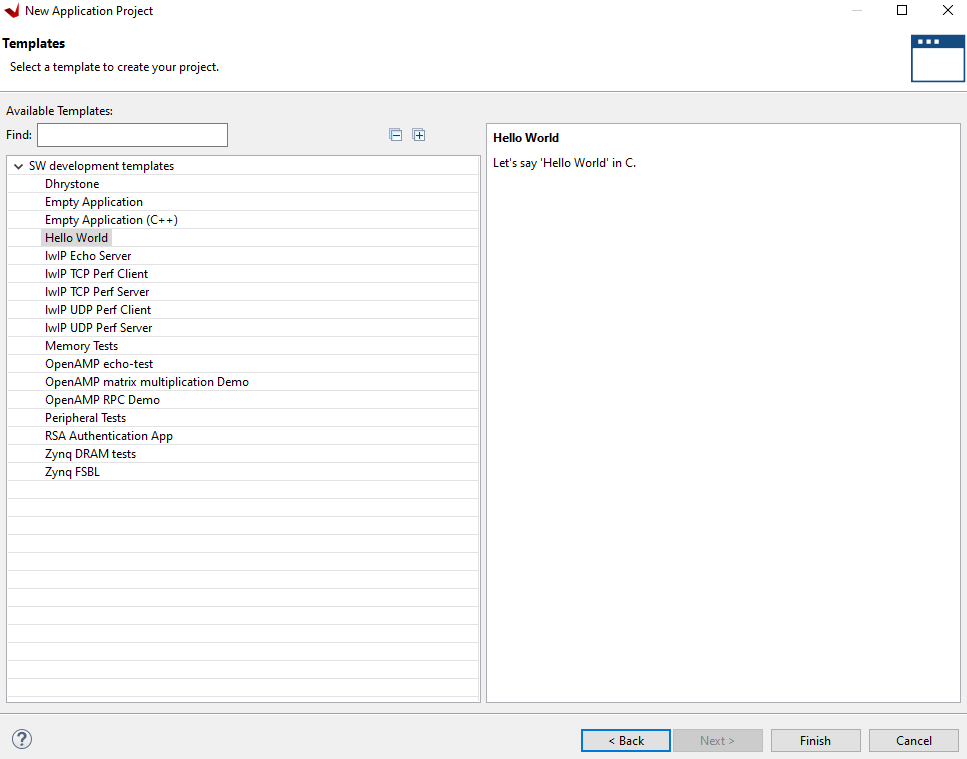
* 1. Give the name of your project and click Next.



* 1. Again click Next.



* 1. Select Hello World and click Finish.



* 1. Copy the following code in helloworld.c

**#include** <stdio.h>

**#include** "platform.h"

**#include** "xil\_printf.h"

**#include** "xgpio.h"

**#include** "xstatus.h"

**#include** "xspi.h"

**#include** "Xil\_io.h"

XSpi\_Config \*ConfigPtr; /\* Pointer to Configuration data \*/

XSpi Spi;

**int** **spiInit**(**void**);

//function to read from 9dof accelerometer/gyro registers using spi

**void** **dof\_ag\_Read**(u8 Add, u8 \* buffer, u8 N){

u8 Write[N+1];

Write[0] = Add | 0x80; //R/W = 1

XSpi\_Transfer(&Spi, Write, buffer, N+1);

}

//function to write in 9dof accelerometer/gyro registers using spi

**void** **dof\_ag\_Write**(u8 Add, u8 val){

u8 Write[2], Read[2];

Write[0] = Add & 0x7F; //R/W = 0

Write[1] = val;

XSpi\_Transfer(&Spi, Write, Read, 2);

}

**void** **ag\_check**( **void**){

u8 readbuffcheck[2];

dof\_ag\_Read(0x0F,readbuffcheck,2);

**if** (readbuffcheck[1]==0x68){

print("AG Spi is working\n\r");

}

**else** {

print("AG Spi is not working\n\r");

}

}

**void** **initGyro**(){

dof\_ag\_Write(0x10,0xC0);

dof\_ag\_Write(0x11,0x00);

dof\_ag\_Write(0x12,0x00);

dof\_ag\_Write(0x1E,0x38);

}

**void** **initAccel**(){

dof\_ag\_Write(0x1F,0x38);

dof\_ag\_Write(0x20,0x00);

dof\_ag\_Write(0x21,0x00);

}

**float** **get\_acceleration**(int16\_t\* axaxis,int16\_t\* ayaxis,int16\_t\* azaxis){

int16\_t ax1,ax2,ay1,ay2,az1,az2;

u8 accelerometerbuff[10]={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};

//dof\_ag\_Write(0x20,0x48); //setting accelerometer settings

dof\_ag\_Read(0x27,accelerometerbuff,7); //reading accelerometer values from register

ax1=(int16\_t)accelerometerbuff[2];

ax2=(int16\_t)accelerometerbuff[3];

ay1=(int16\_t)accelerometerbuff[4];

ay2=(int16\_t)accelerometerbuff[5];

az1=(int16\_t)accelerometerbuff[6];

az2=(int16\_t)accelerometerbuff[7];

axaxis=(ax2<<8)|ax1;

ayaxis=(ay2<<8)|ay1;

azaxis=(az2<<8)|az1;

//return xaxis,yaxis,zaxis;

}

**float** **get\_gyro**(){

int16\_t gx1,gx2,gy1,gy2,gz1,gz2;

u8 gyrobuff[10]={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};

//dof\_ag\_Write(0x10,0x20); //setting gyro

dof\_ag\_Read(0x17,gyrobuff,7); //reading gyro values from register

gx1=(int16\_t)gyrobuff[2];

gx2=(int16\_t)gyrobuff[3];

gy1=(int16\_t)gyrobuff[4];

gy2=(int16\_t)gyrobuff[5];

gz1=(int16\_t)gyrobuff[6];

gz2=(int16\_t)gyrobuff[7];

int16\_t gxaxis=(gx2<<8)|gx1;

int16\_t gyaxis=(gy2<<8)|gy1;

int16\_t gzaxis=(gz2<<8)|gz1;

}

**int** **main**()

{

init\_platform();

spiInit();

ag\_check();

initAccel();

initGyro();

**while**(1){

// u8 accelbuff[10]={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};

// dof\_ag\_Write(0x20,0x48);

// dof\_ag\_Read(0x28,accelbuff,6);

int16\_t axaxis,ayaxis,azaxis,gxaxis,gyaxis,gzaxis;

get\_acceleration(&axaxis,&ayaxis,&azaxis);

//for(int i=0; i<10;i++){};

get\_gyro();

//for(int i=0; i<10;i++){};

}

cleanup\_platform();

**return** 0;

}

**int** **spiInit**(**void**){

**int** Status;

ConfigPtr = XSpi\_LookupConfig(0U);

**if** (ConfigPtr == NULL) {

**return** XST\_DEVICE\_NOT\_FOUND;

}

Status = XSpi\_CfgInitialize(&Spi, ConfigPtr,ConfigPtr->BaseAddress);

**if** (Status != XST\_SUCCESS) {

**return** XST\_FAILURE;

}

Status = XSpi\_SetOptions(&Spi, XSP\_MASTER\_OPTION );

**if** (Status != XST\_SUCCESS) {

**return** XST\_FAILURE;

}

XSpi\_Start(&Spi);

XSpi\_IntrGlobalDisable(&Spi);

XSpi\_SetSlaveSelect(&Spi, 0x01);

}