ADIUVO

ENGINEERING AND TRAINING, LTD.

# MicroBlaze Booting From QSPI Flash

Vivado 2020.1

Target Board : Arty S7-50 – Issue B

# Introduction

This document shows how to ensure a MicroBlaze processor can boot from QSPI flash and run an application when the application is executed from DDR.

There are two parts to the document, creation of the MicroBlaze system and the creation of the software.

To save time and prevent the need to create the Vivado design from scratch a TCL script is provided to recreate the project.

A reference BIN file is also provided for demonstration.

# Introduction

The basic flow for this development is to

1. Create the Vivado block diagram
2. Generate the application software in Vitis to run from DDR
3. Generate a QSPI bootloader
4. Convert the application SW to S Record format
5. Merge the QSPI bootloader elf with the bitstream
6. Create a combined bin file which contains the Bitstream & Srecord application
7. Write the combined bin file to the QSPI

# Key Differences

The main difference compared to a program which is loaded from QSPI and executes from the BRAM is that we continue to need access to QSPI post configuration.

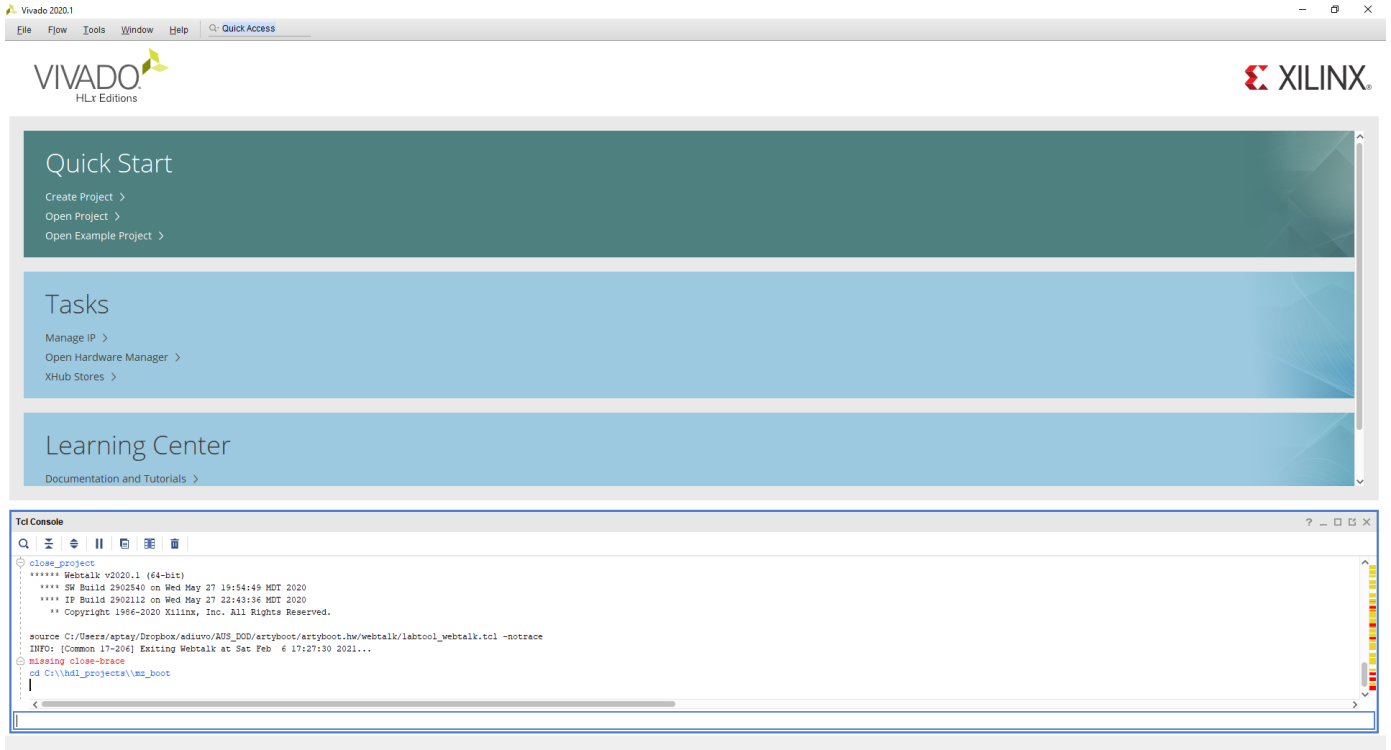
For this reason, the Vivado design must also contain a QSPI interface mapped to the same pins to enable the bootloader to read the application from the QSPI and relocate it to the DDR memory.

# MicroBlaze Creation in Vivado from TCL Script

ADIUVO
ENGINEERING AND TRAINING, LTD.

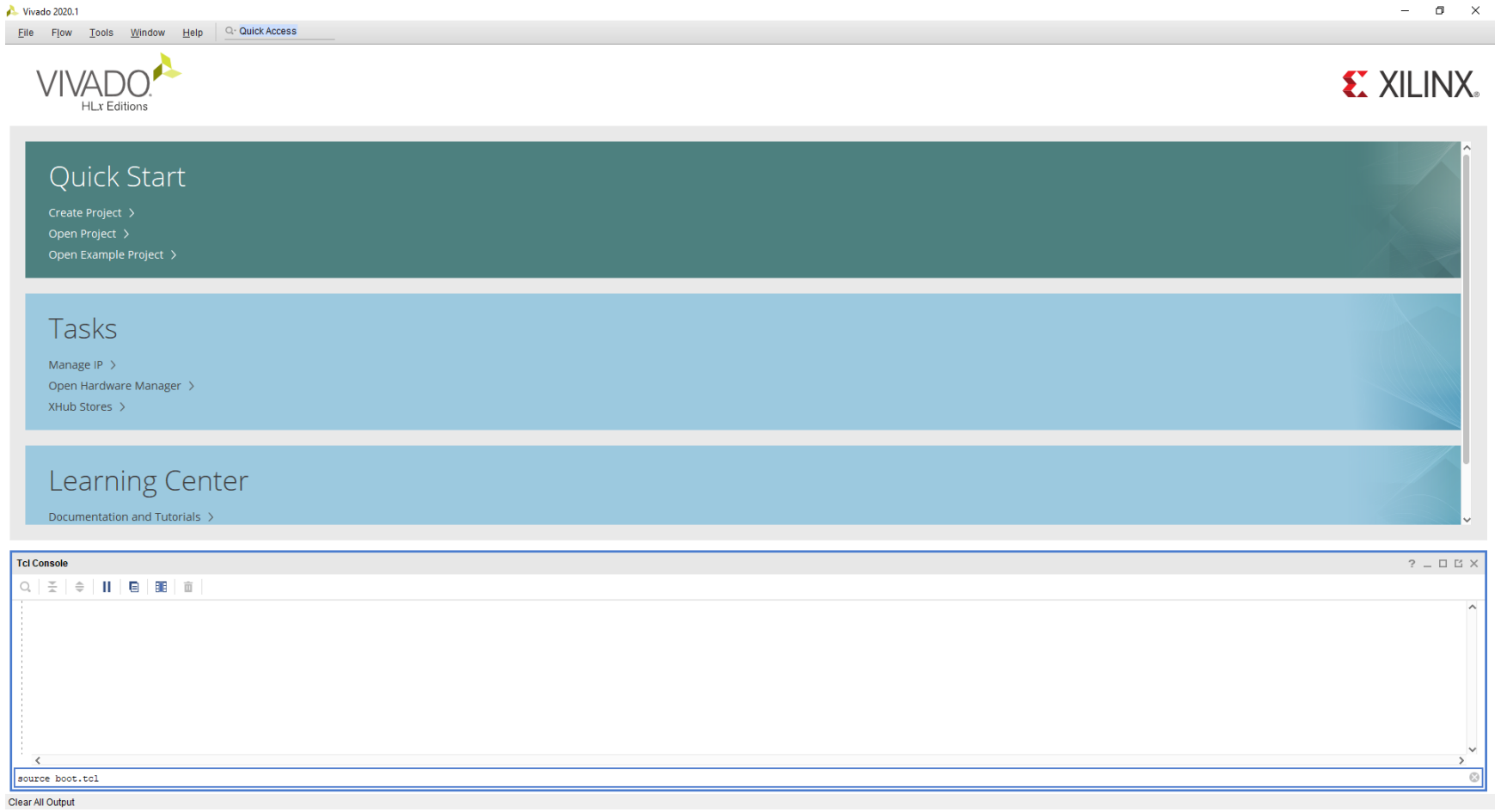
# MicroBlaze Boot From QSPI

In the TCL console change directory one you wish to recreate the project in. Copy in the file boot.tcl into that directory



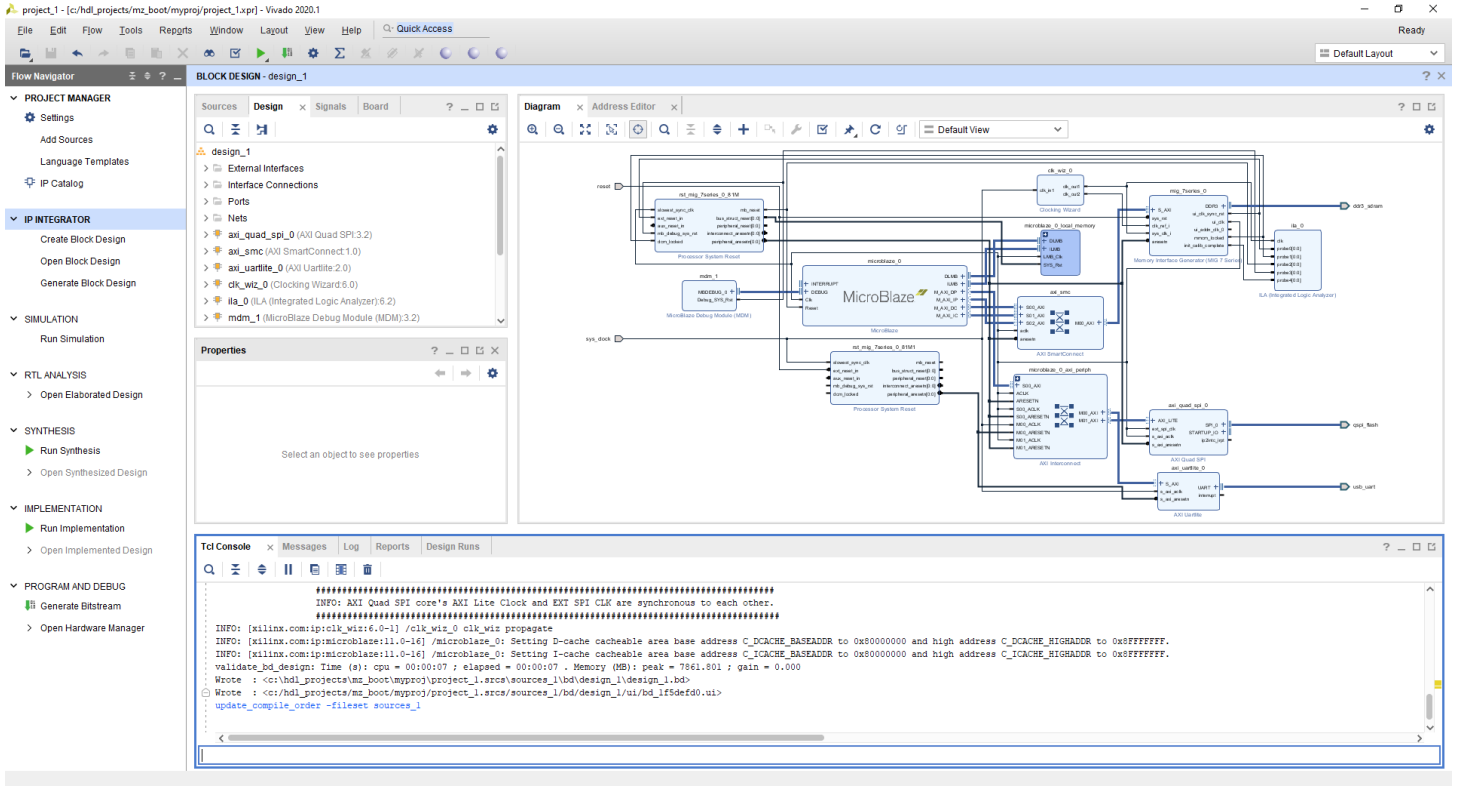
# MicroBlaze Boot From QSPI

Run the command source boot.tcl – from the TCL console



# MicroBlaze Boot From QSPI

This will recreate the block diagram – requiring only the implementation to be able to start development in Vitis.

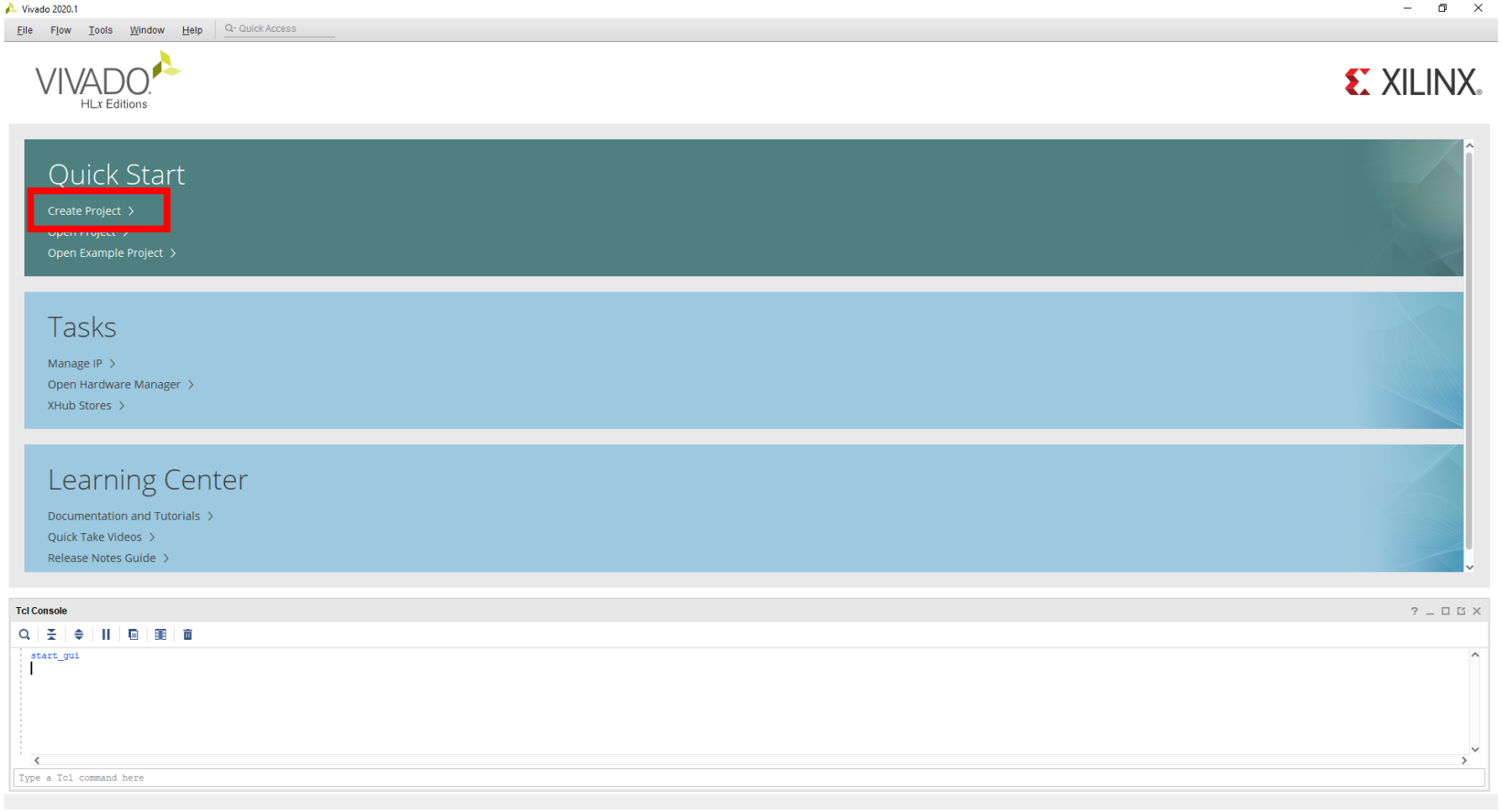


# MicroBlaze Creation in Vivado from Scratch

ADIUVO
ENGINEERING AND TRAINING, LTD.

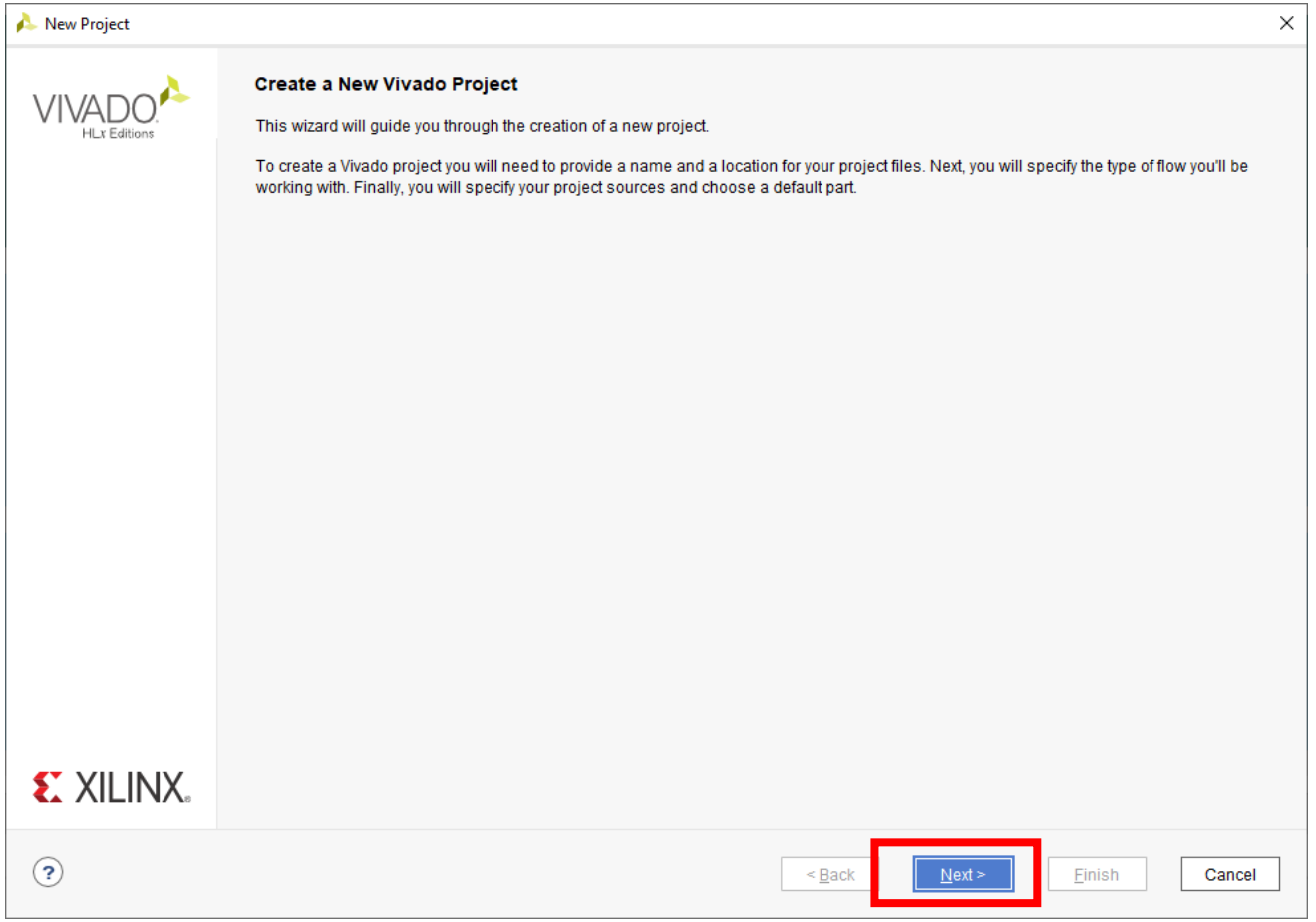
# MicroBlaze Boot From QSPI

## Open a new project in Vivado 2020.1



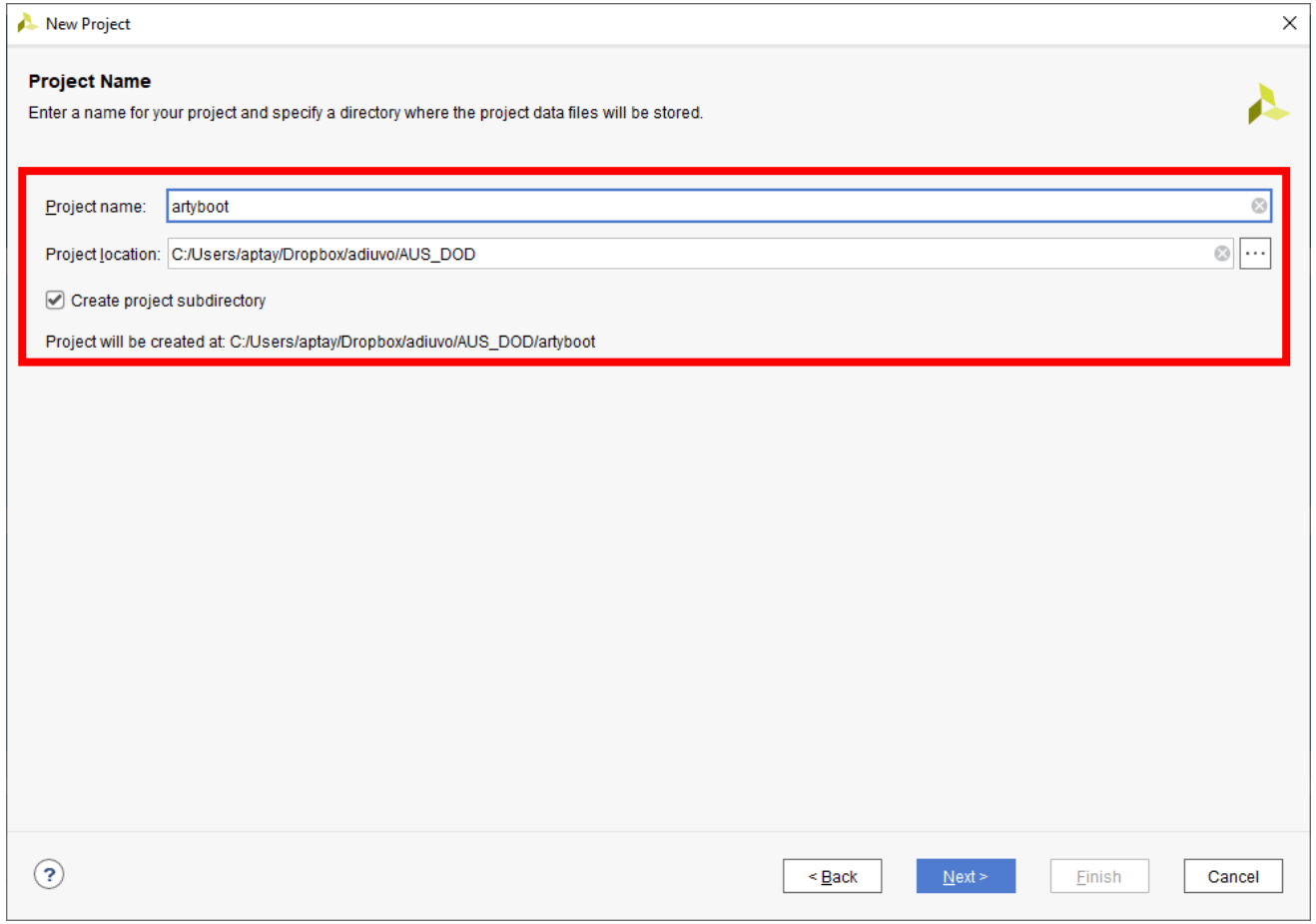
# MicroBlaze Boot From QSPI

Click on next



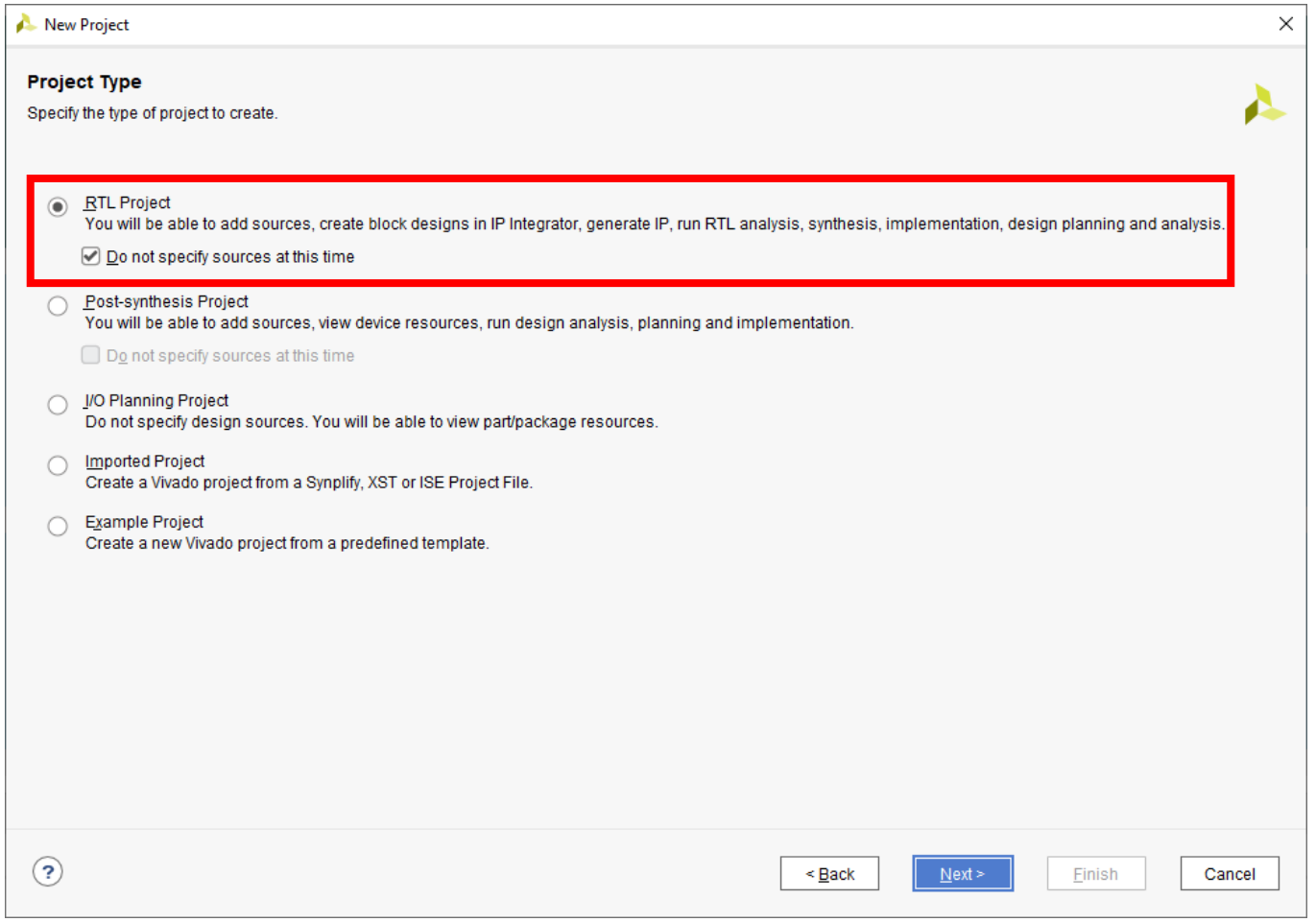
# MicroBlaze Boot From QSPI

Enter a name and project location, click next



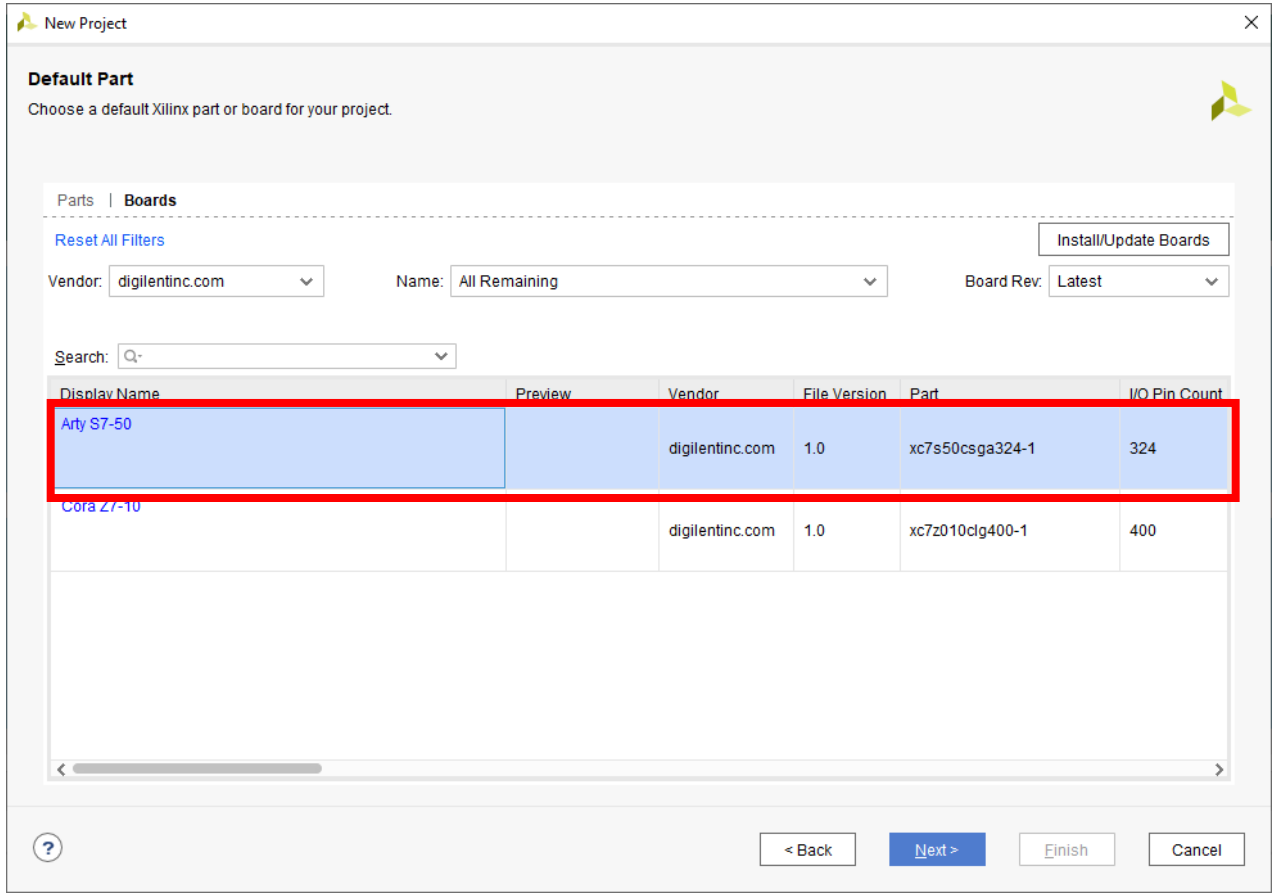
# MicroBlaze Boot From QSPI

Select RTL Project and specify sources at later, click next



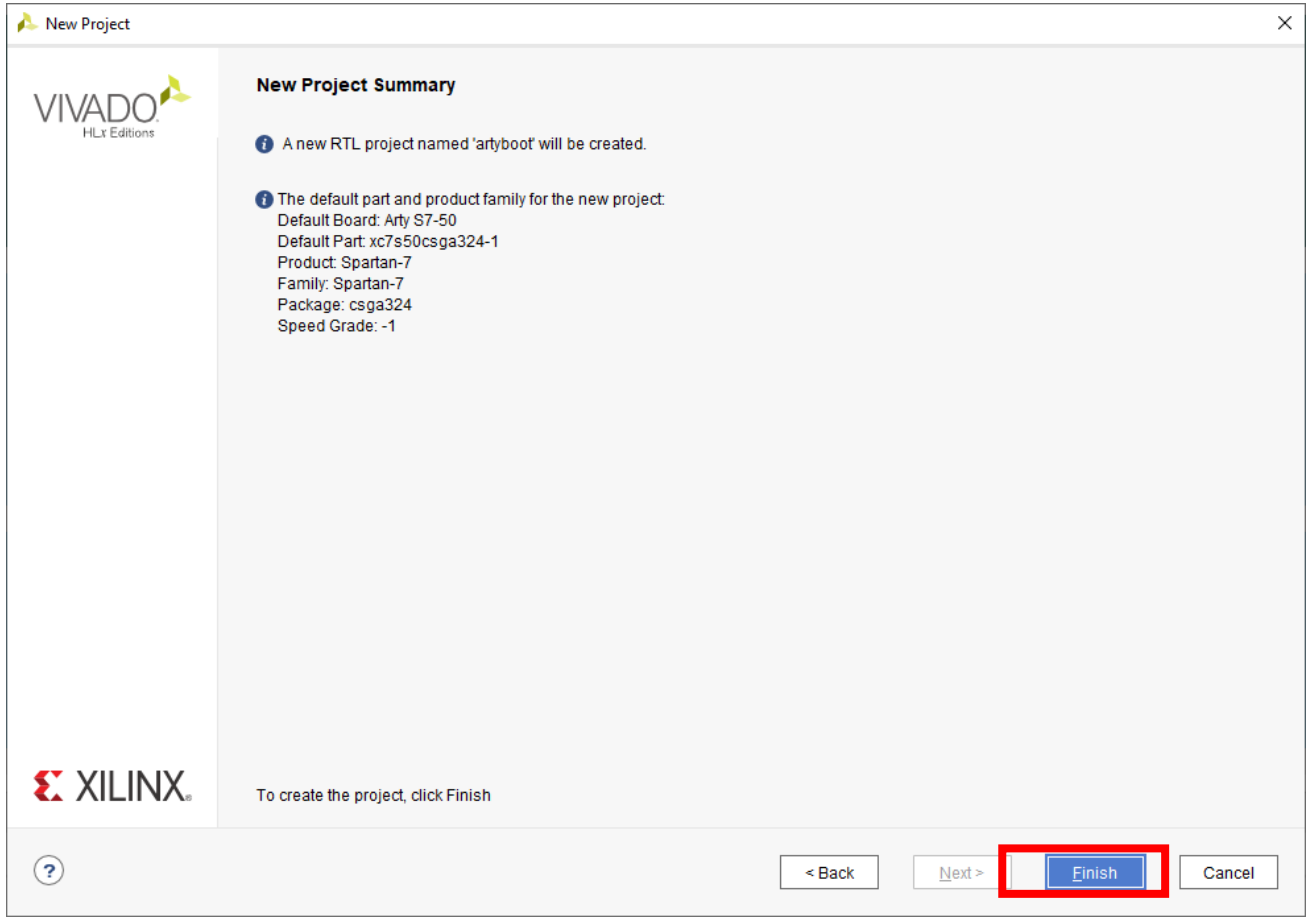
# MicroBlaze Boot From QSPI

Select Boards, and select the Arty S7-50



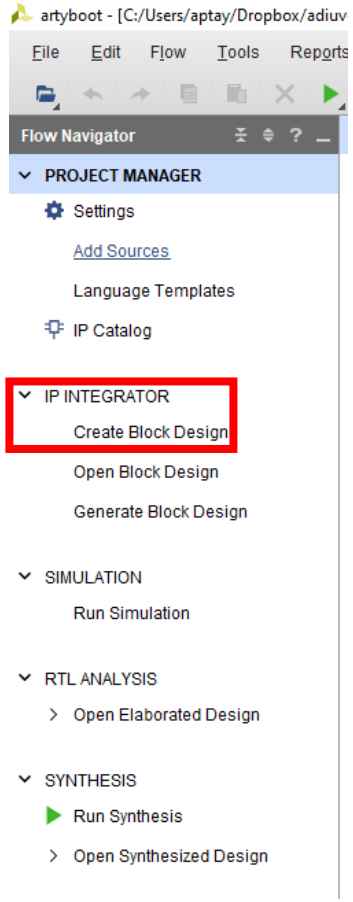
# MicroBlaze Boot From QSPI

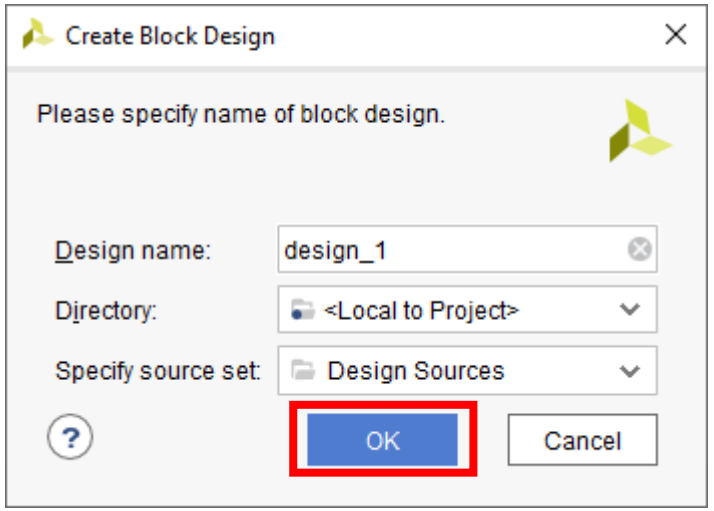
Finish the project creation



# MicroBlaze Boot From QSPI

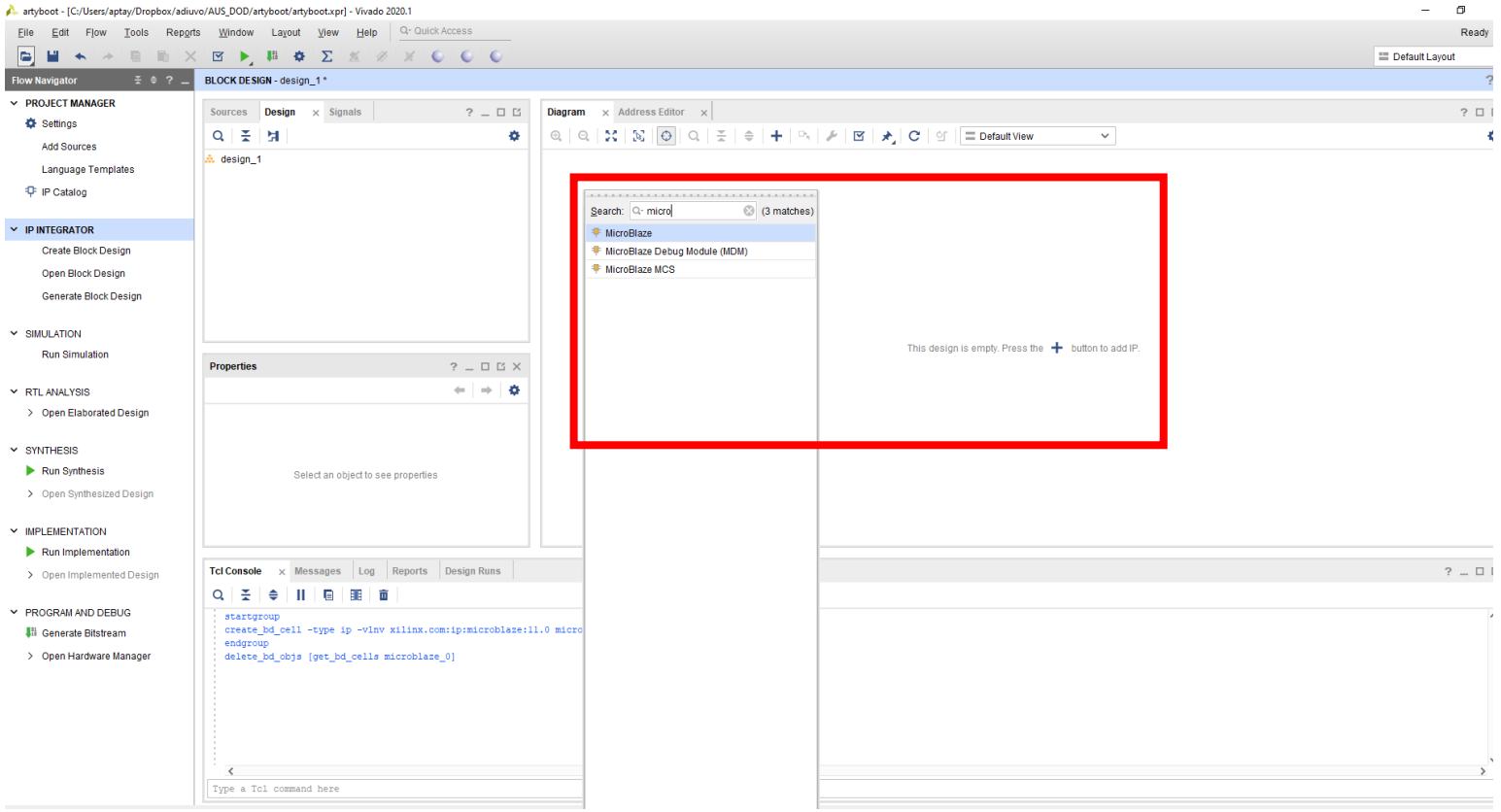
Create a new block design, leave the defaults the same in the dialog



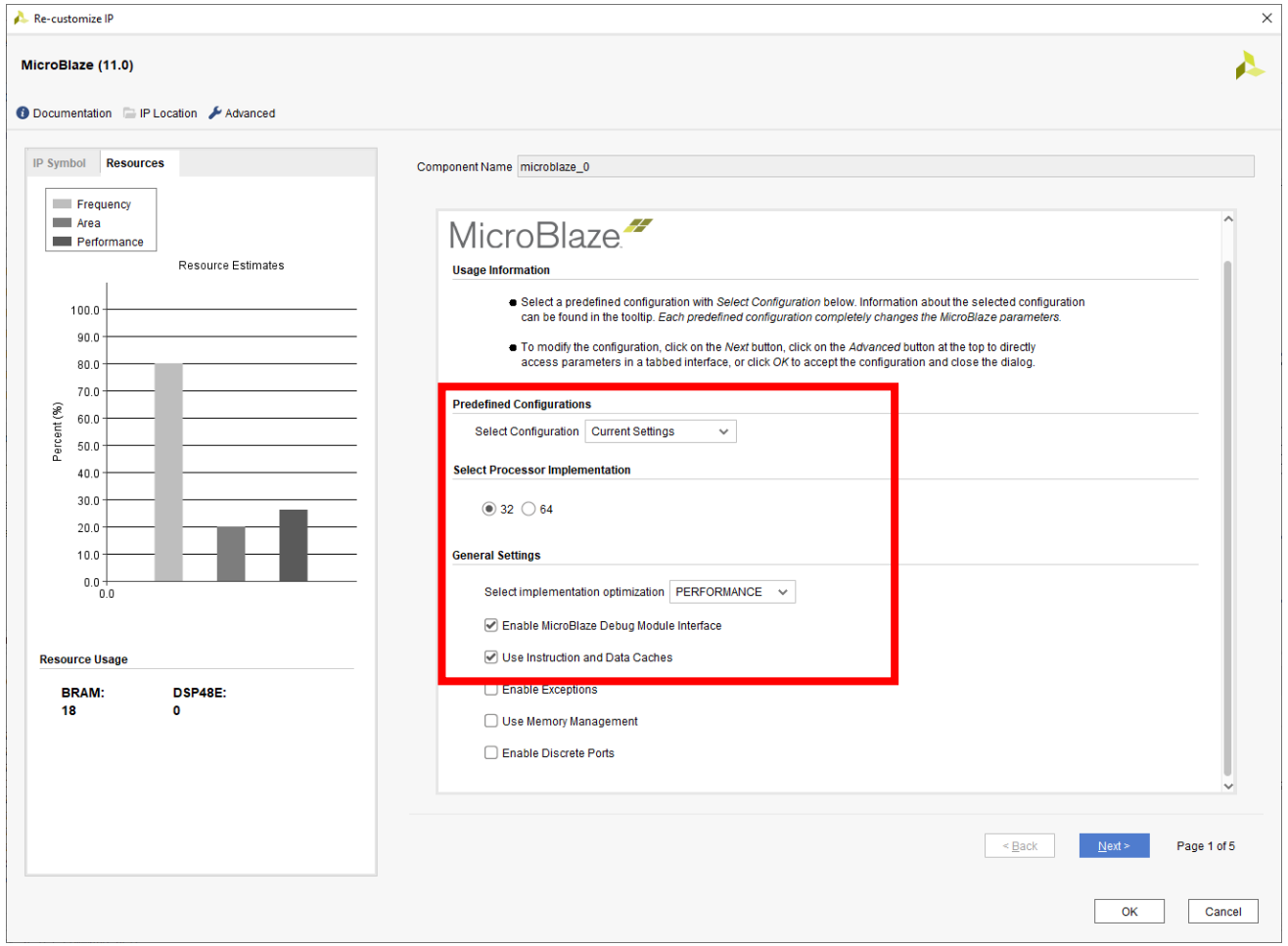
# MicroBlaze Boot From QSPI

Click the + symbol and add in a MicroBlaze Processor



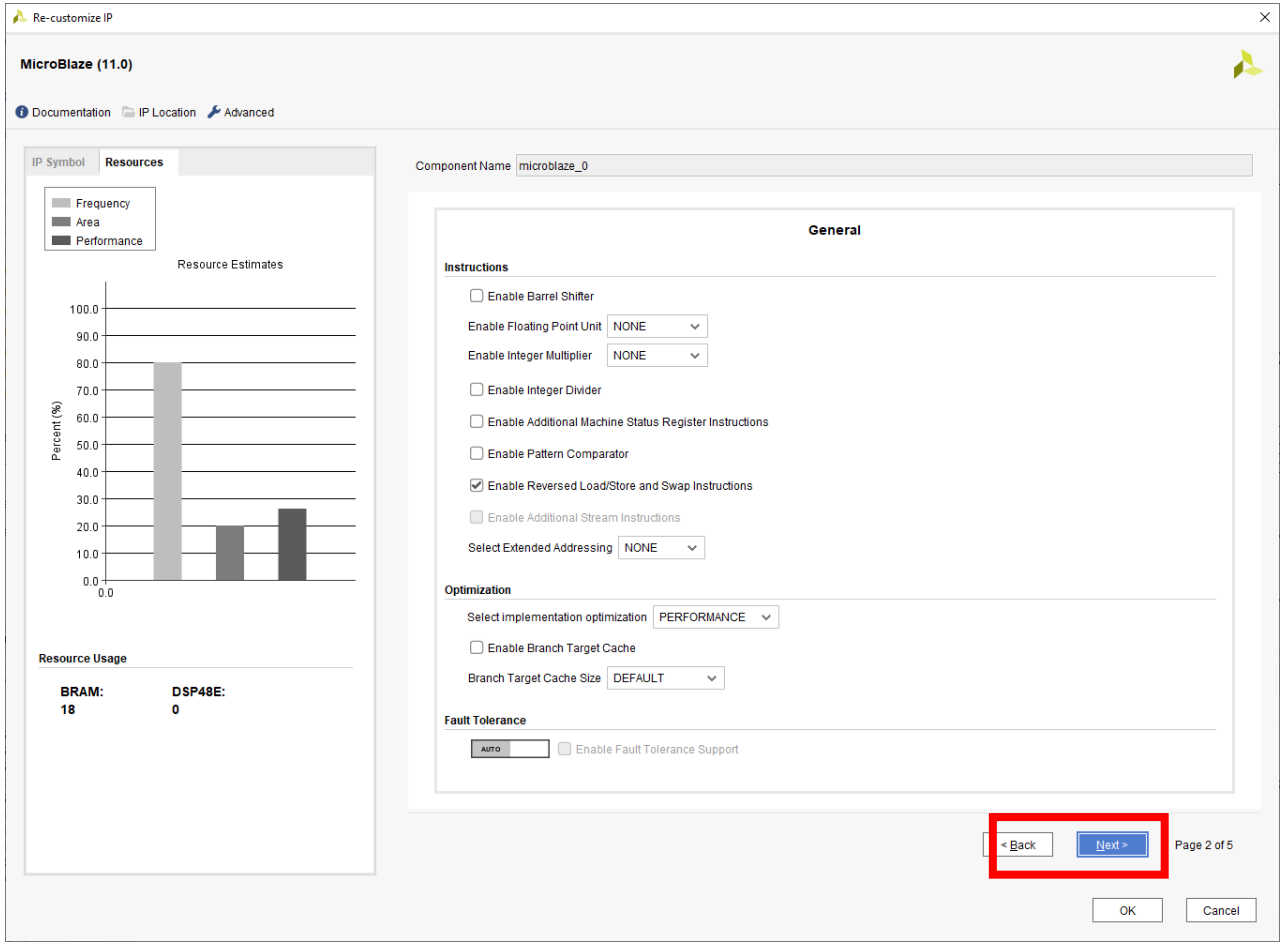
# MicroBlaze Boot From QSPI

Re-customize the MicroBlaze IP – Ensure the Debug and Cache are enabled



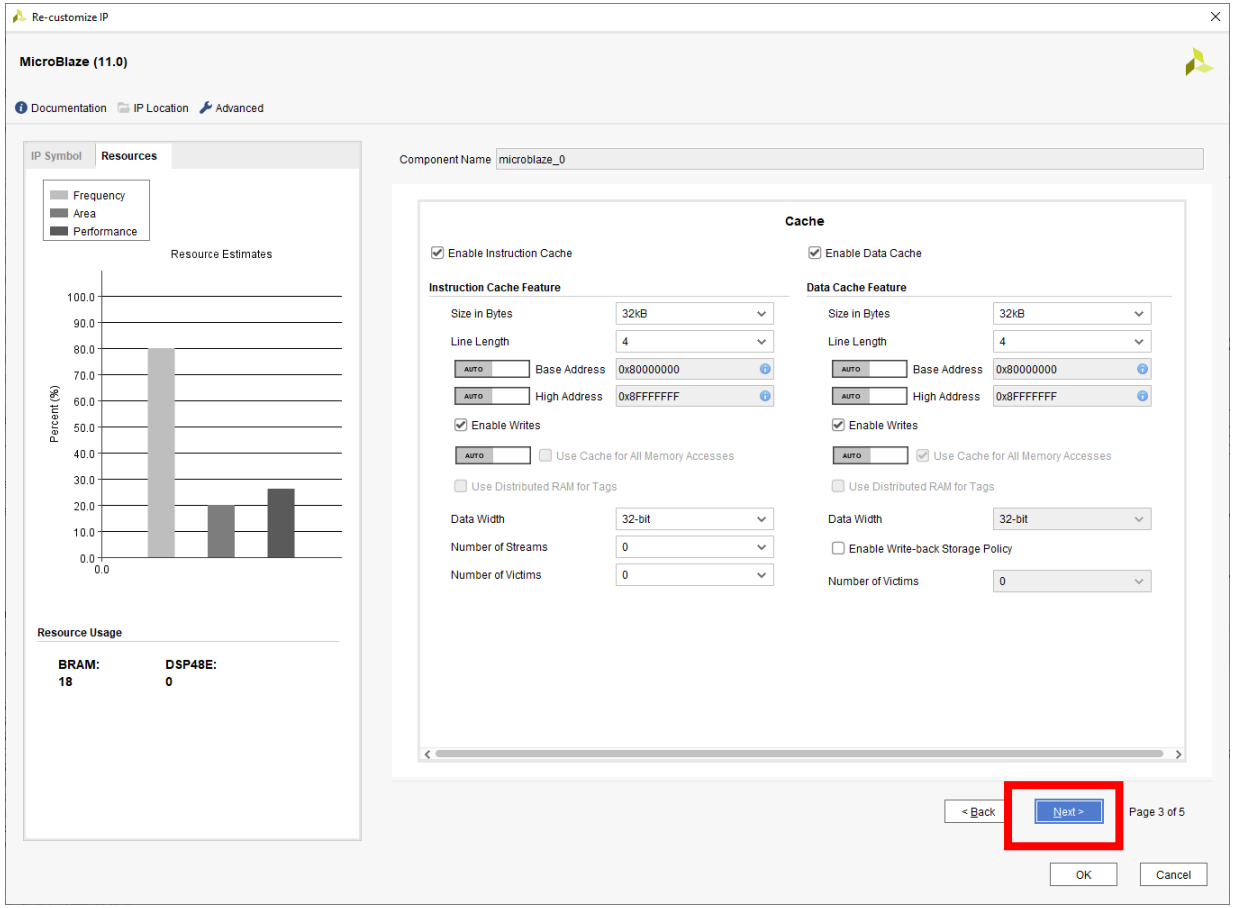
# MicroBlaze Boot From QSPI

On the second page leave unchanged and click next



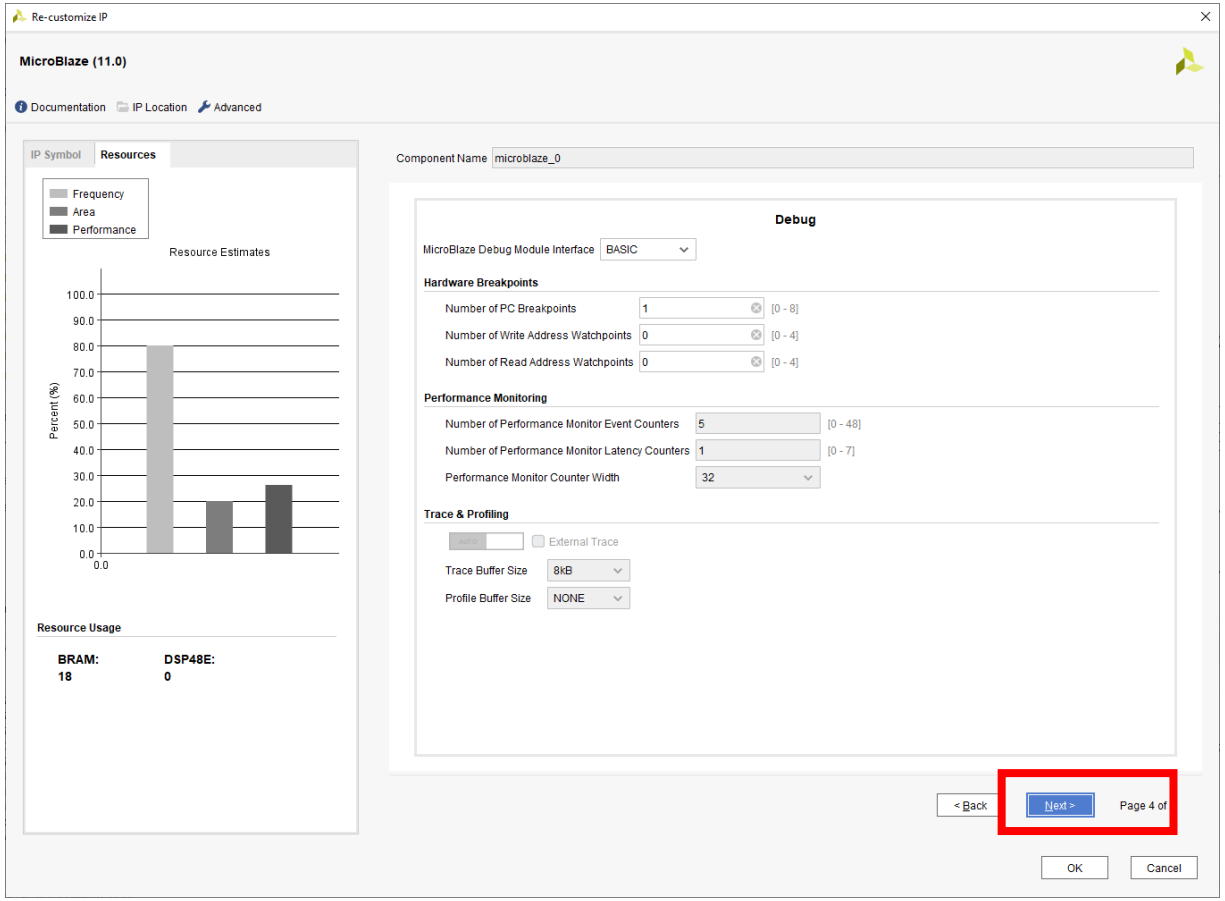
# MicroBlaze Boot From QSPI

Leave the third page unchanged and click next



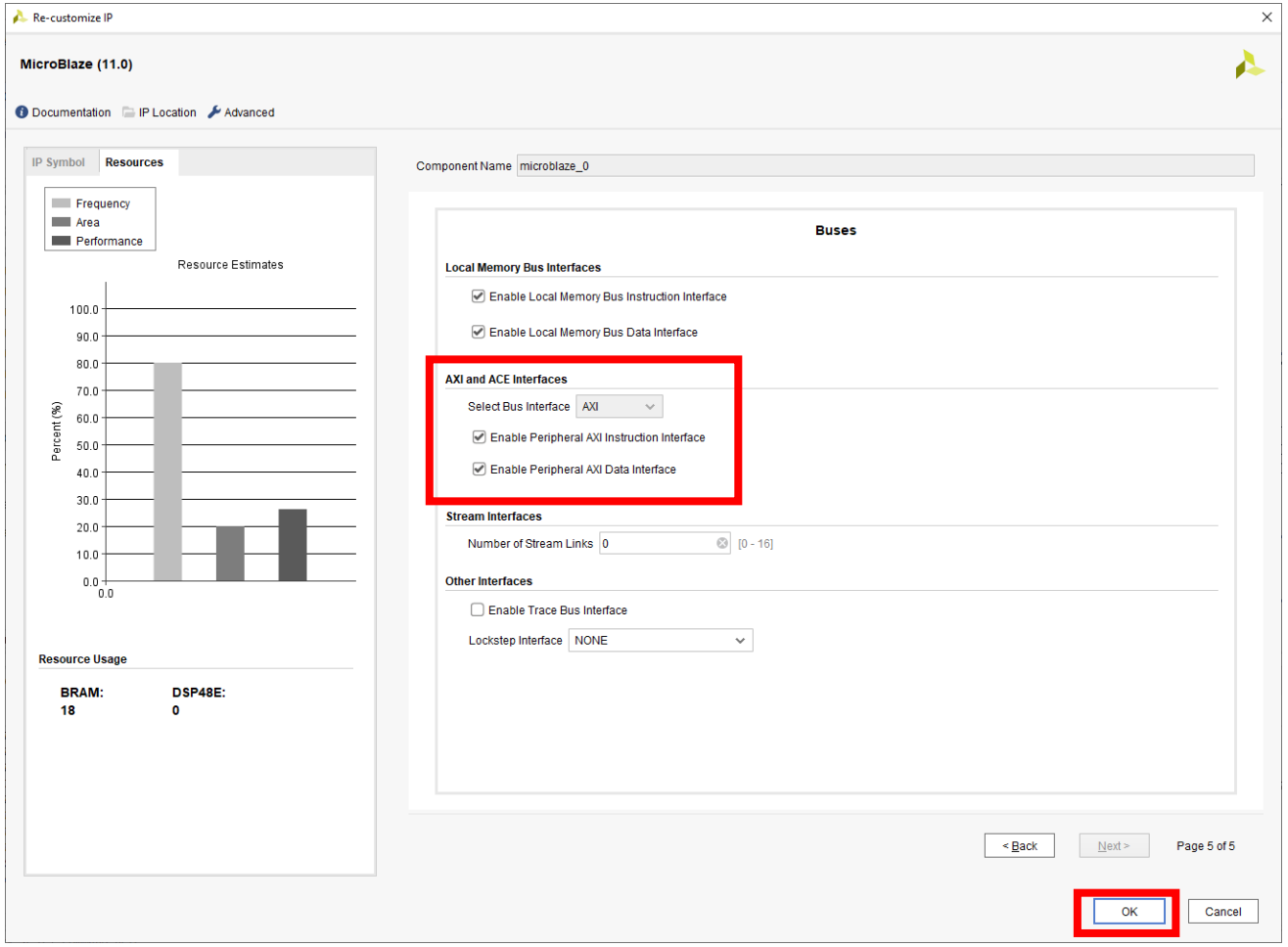
# MicroBlaze Boot From QSPI

Leave the fourth page unchanged and click next



# MicroBlaze Boot From QSPI

Enable the AXI interfaces and click OK

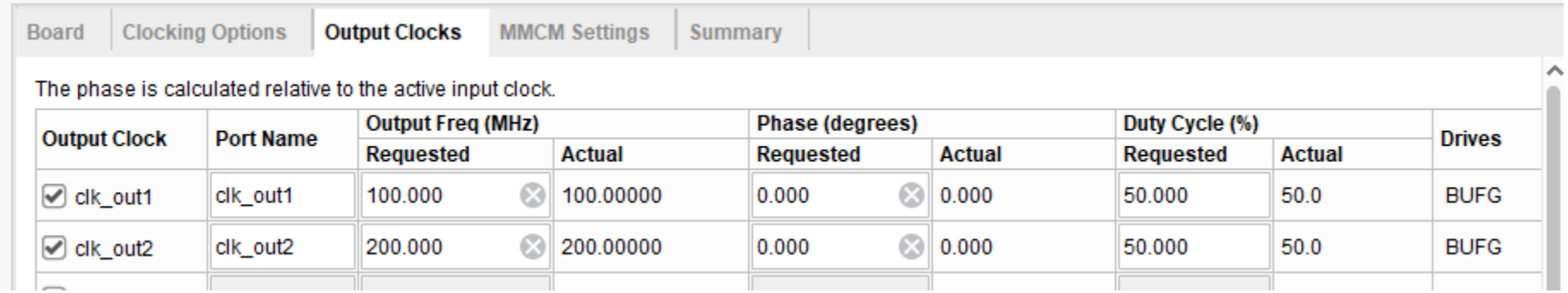


# MicroBlaze Boot From QSPI

From the board tab drag and drop on the DDR3 SDRAM also drag on the System clock from the boards tab.

Delete the reset and DDR clock connected to the MIG

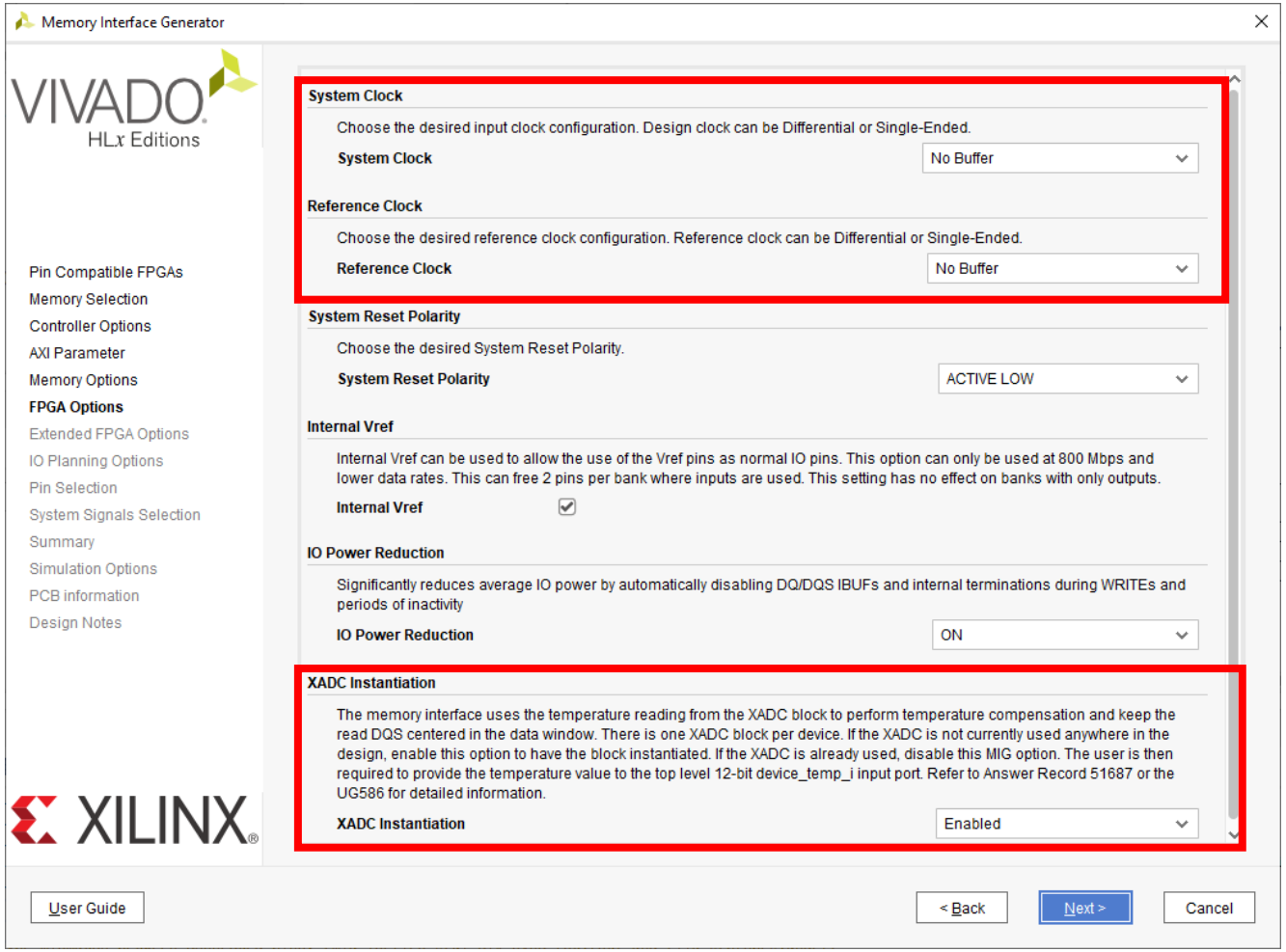
Re-customize the Clock Wizard associated with the System Clock to have no reset input, no locked signal  and to output 100MHz and 200MHz



# MicroBlaze Boot From QSPI

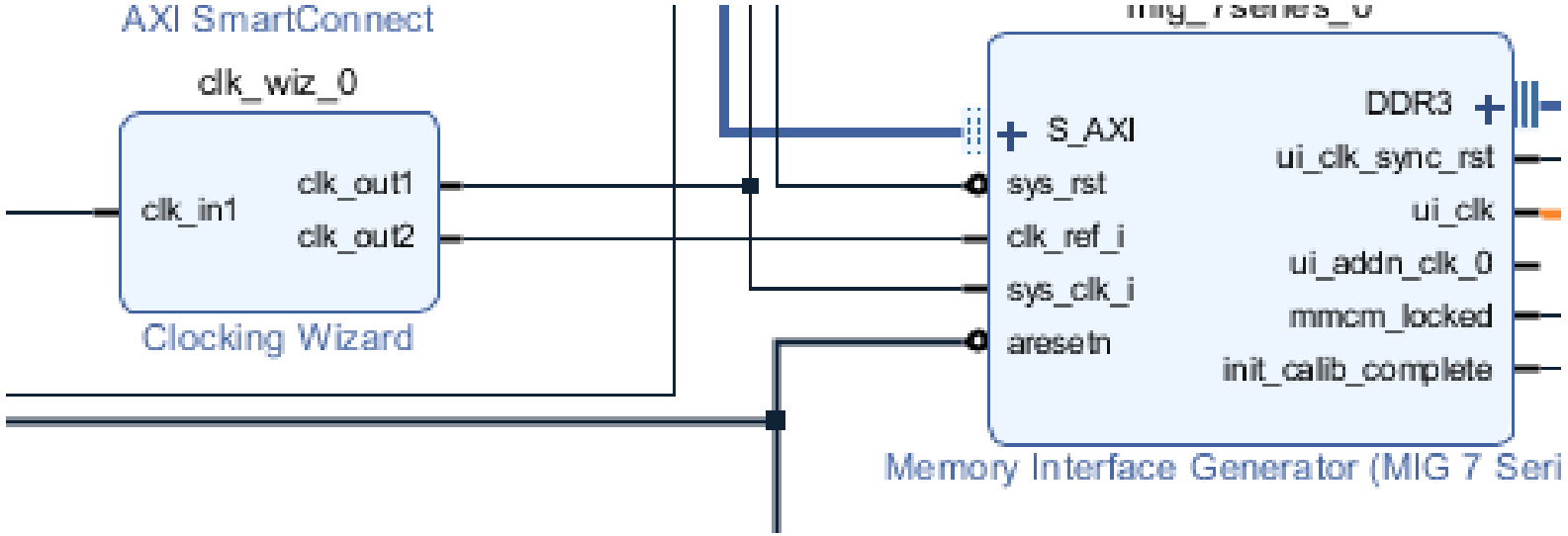
Re Customize the DDR MIG Change

- System Clock No Buffer
- Reference Clock No Buffer
- XADC Instantiation Enabled



# MicroBlaze Boot From QSPI

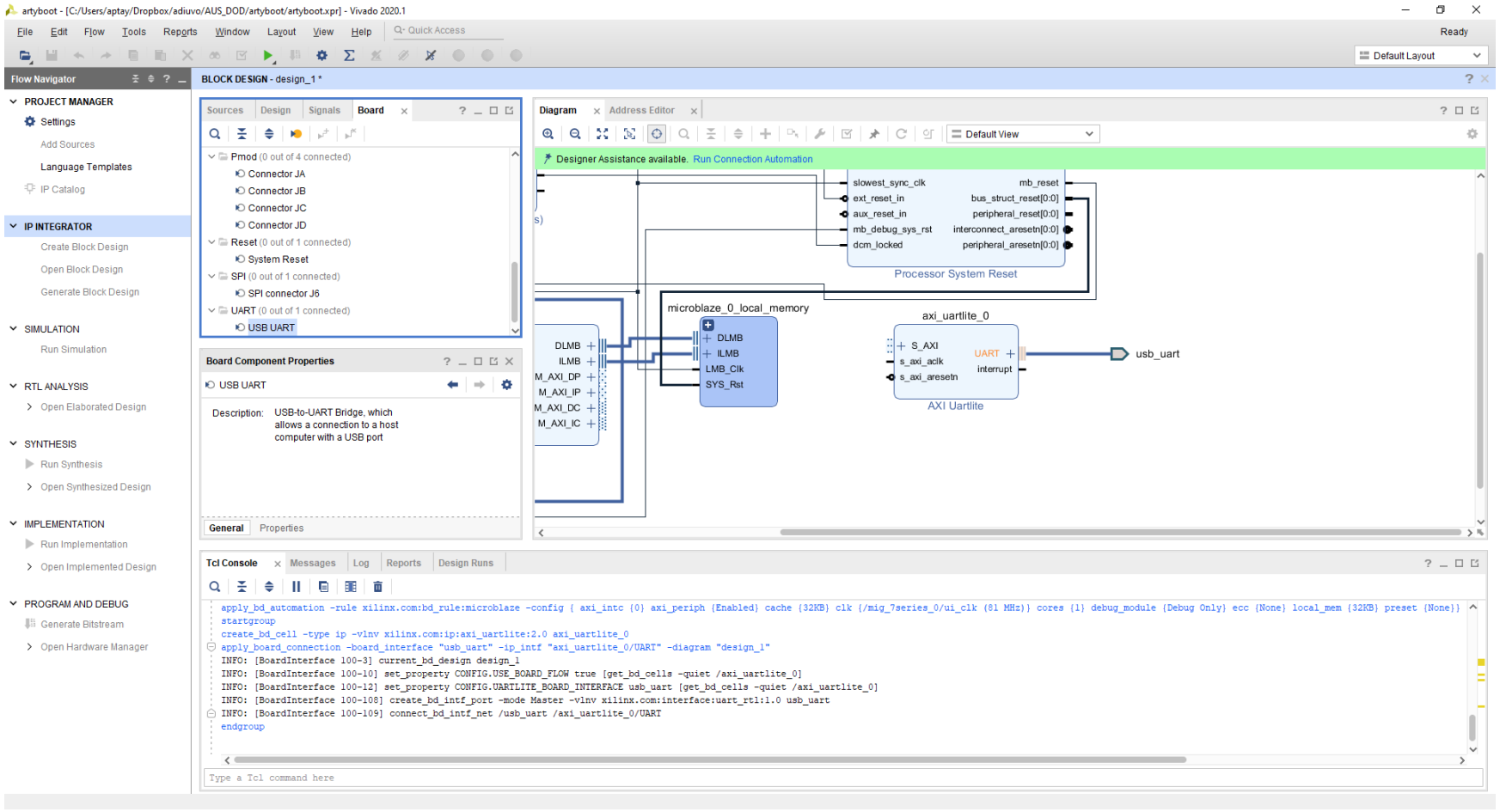
Connect the 100 MHz to the Sys_clk and 200 MHz to the Clk_ref




Note, this is necessary as there is an issue with the DDR CLK on the Rev B version of the Arty S7

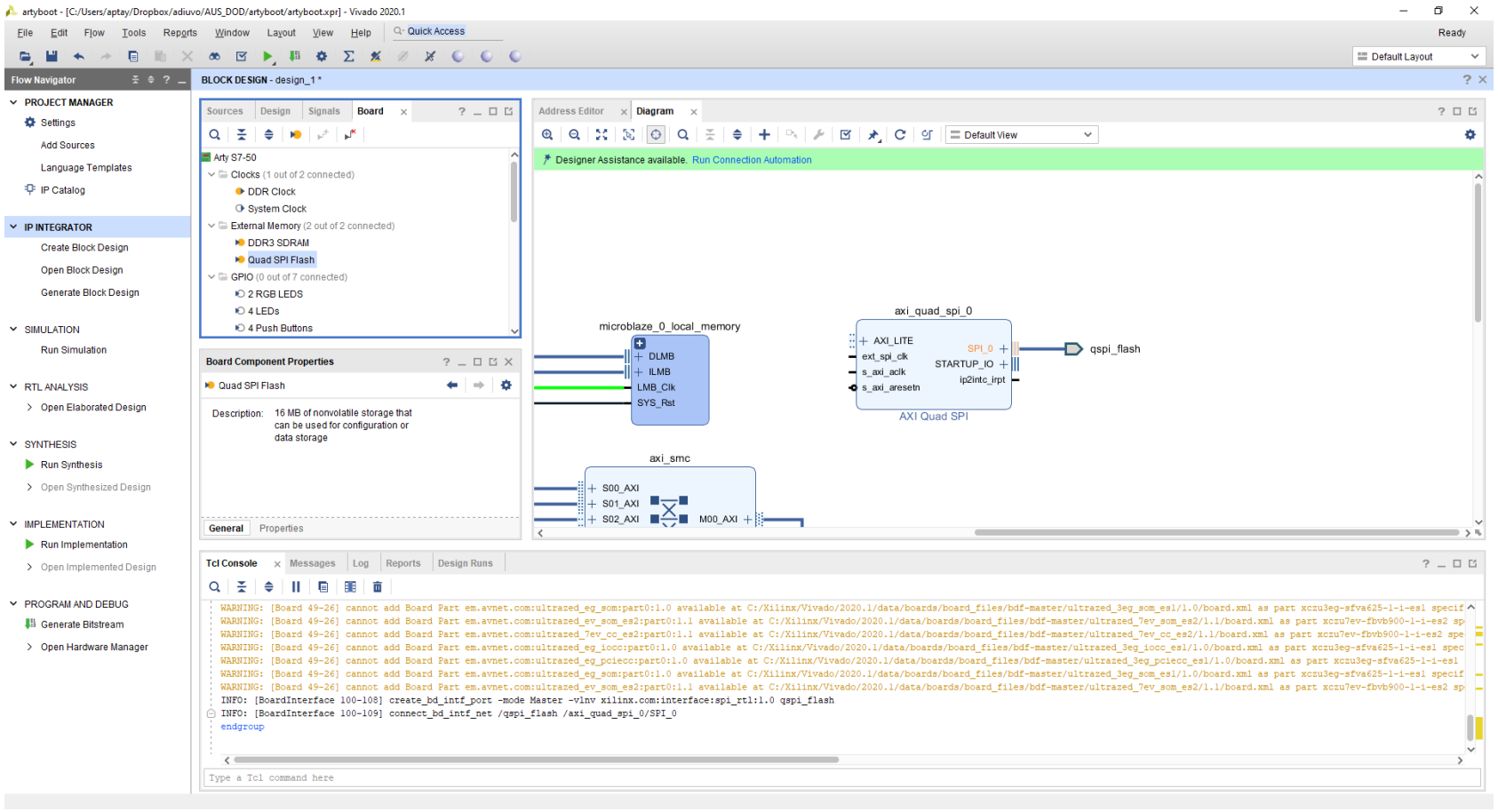
# MicroBlaze Boot From QSPI

Drag Across the USB UART to the block diagram



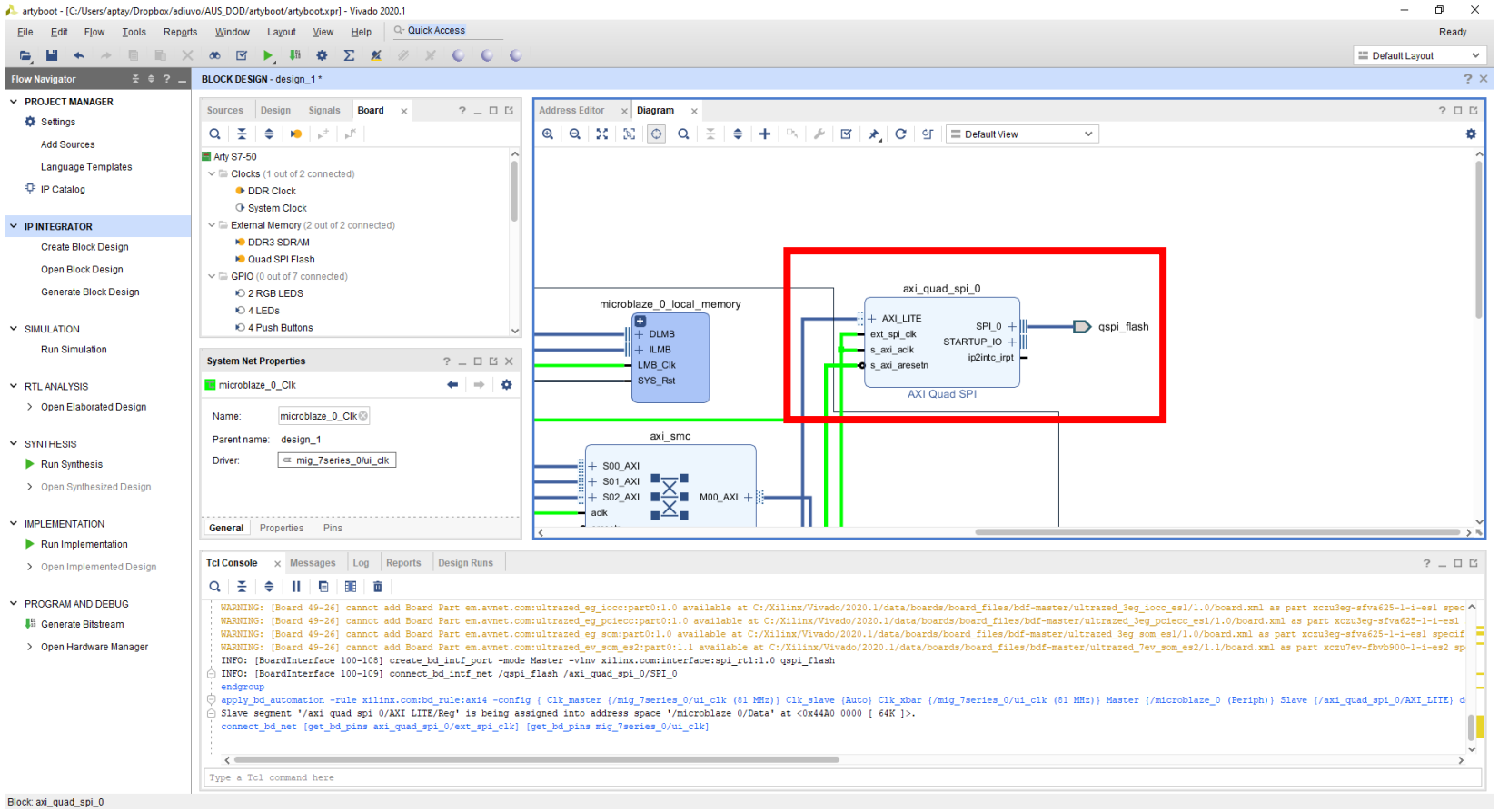
# MicroBlaze Boot From QSPI

Drag the Quad SPI flash onto the Block Diagram



# MicroBlaze Boot From QSPI

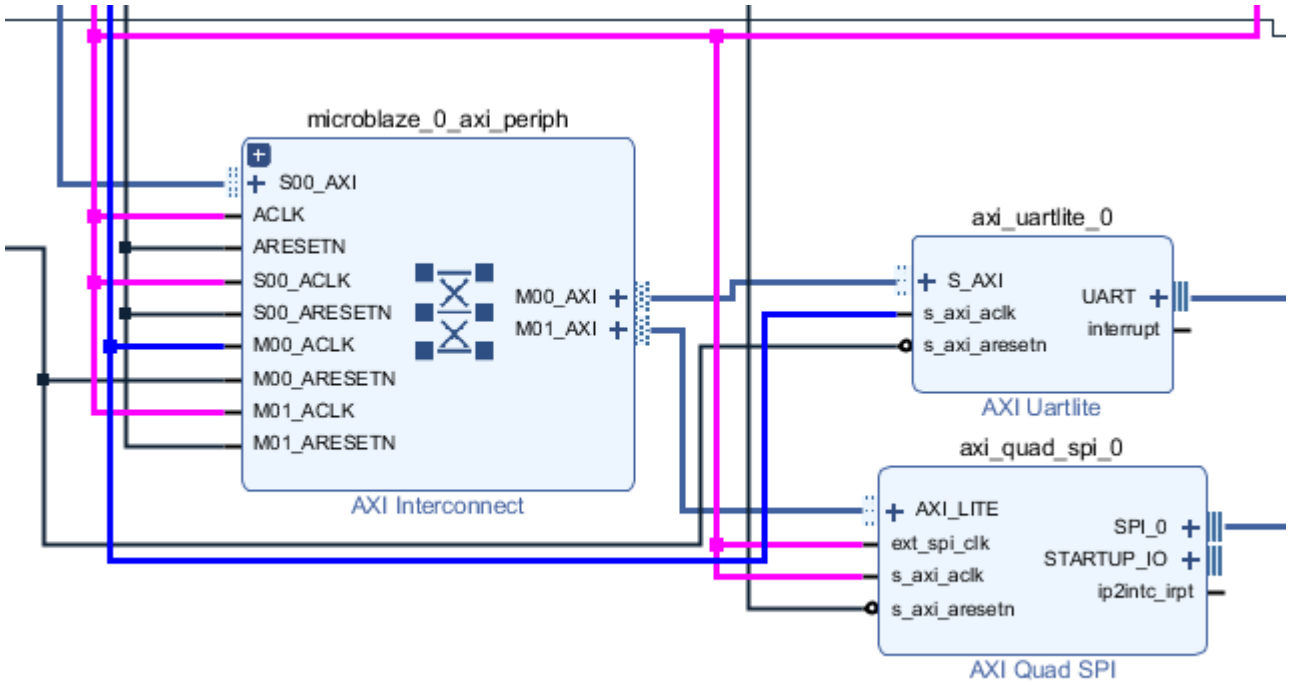
Run the connection automation to create the AXI Network.  On the Quad AXI SPI connect the ext_spi_clk to the s_axi_clk



# MicroBlaze Boot From QSPI

Modify the S_AXI_CLK for the AXI UART and its master AXI port on the AXI Interconnect to run from System_Clk and NOT the UI_CLK

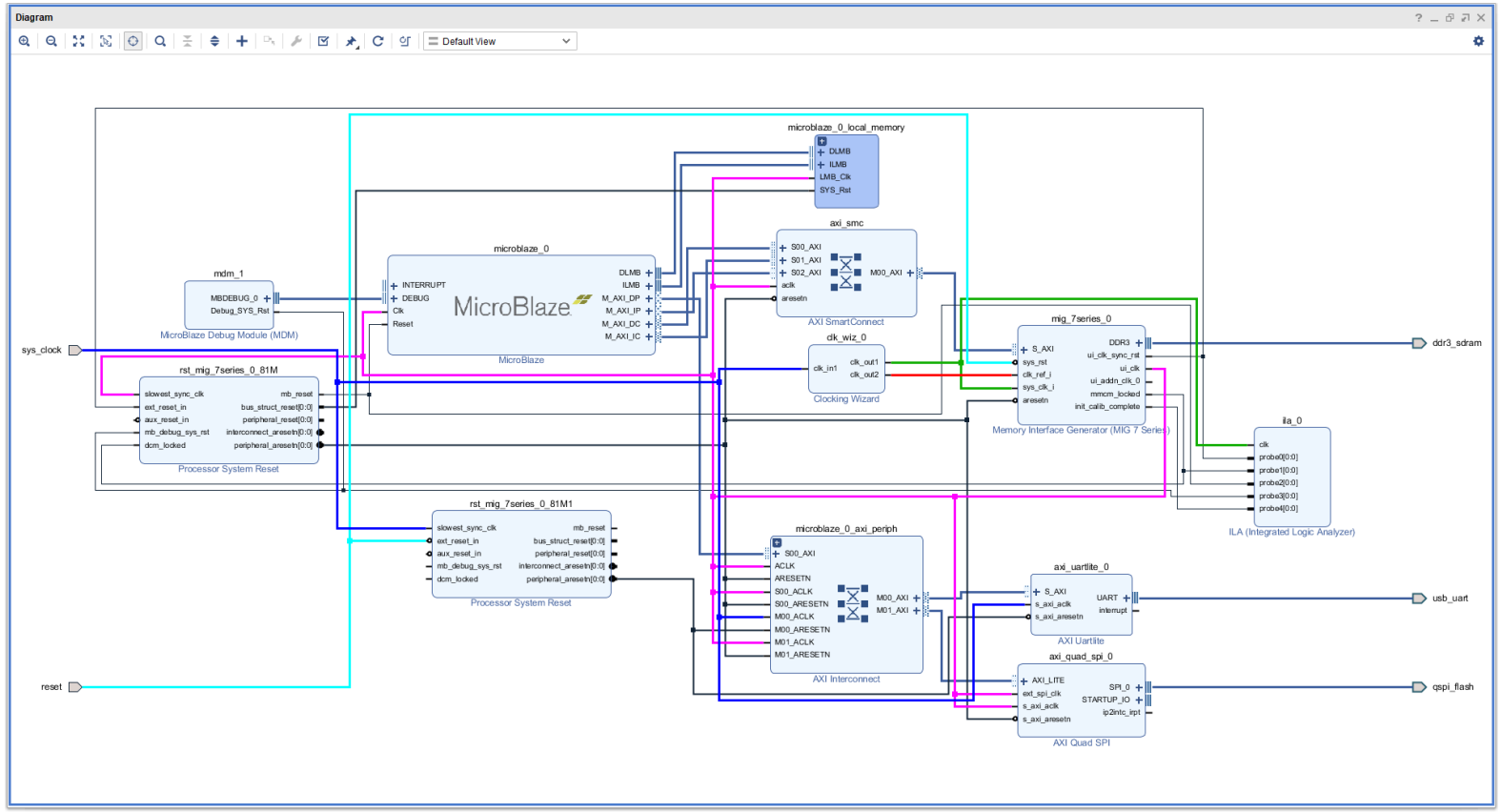
The AXI Interconnect will address the clock domain crossing safely




Note, this is necessary to be able to generate the baud rates for Serial Communication

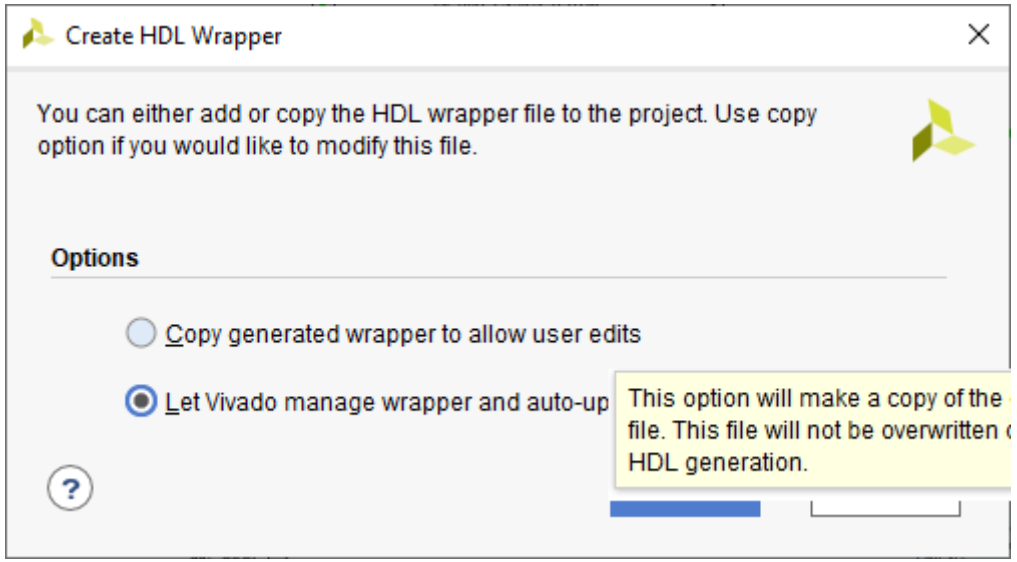
# MicroBlaze Boot From QSPI

The final design in Vivado should be as below - the provided project TCL should recreate this.



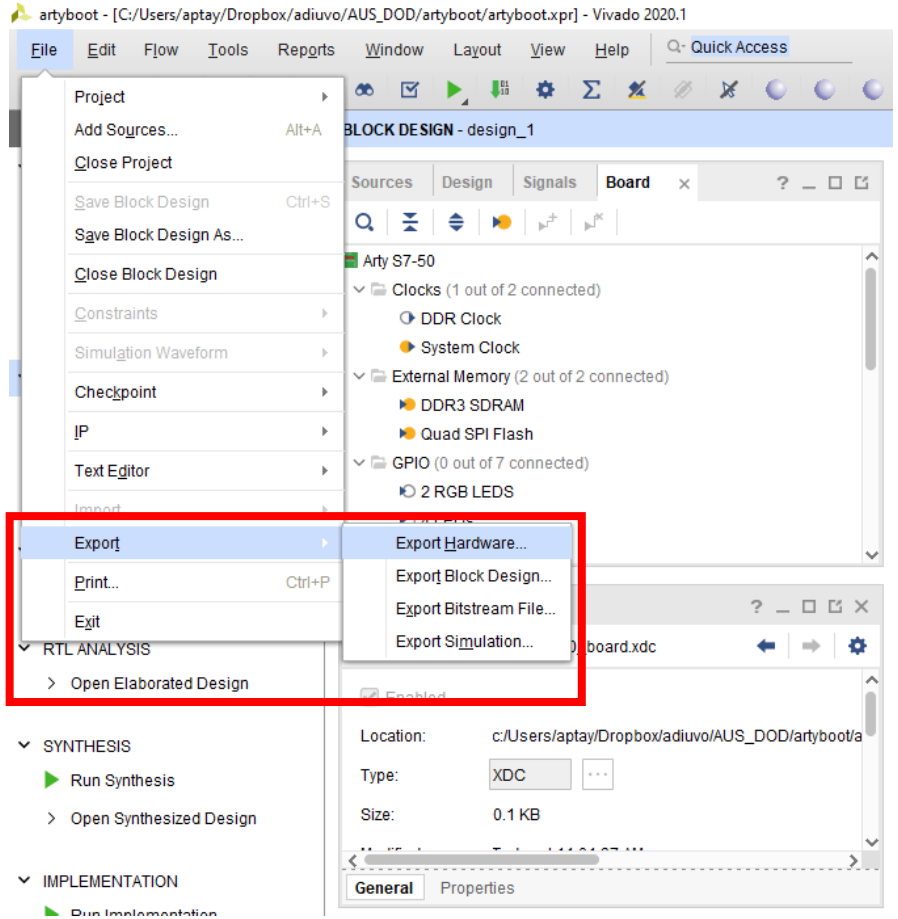
# MicroBlaze Boot From QSPI

Create an HDL Wrapper for the block design and generate the bit stream



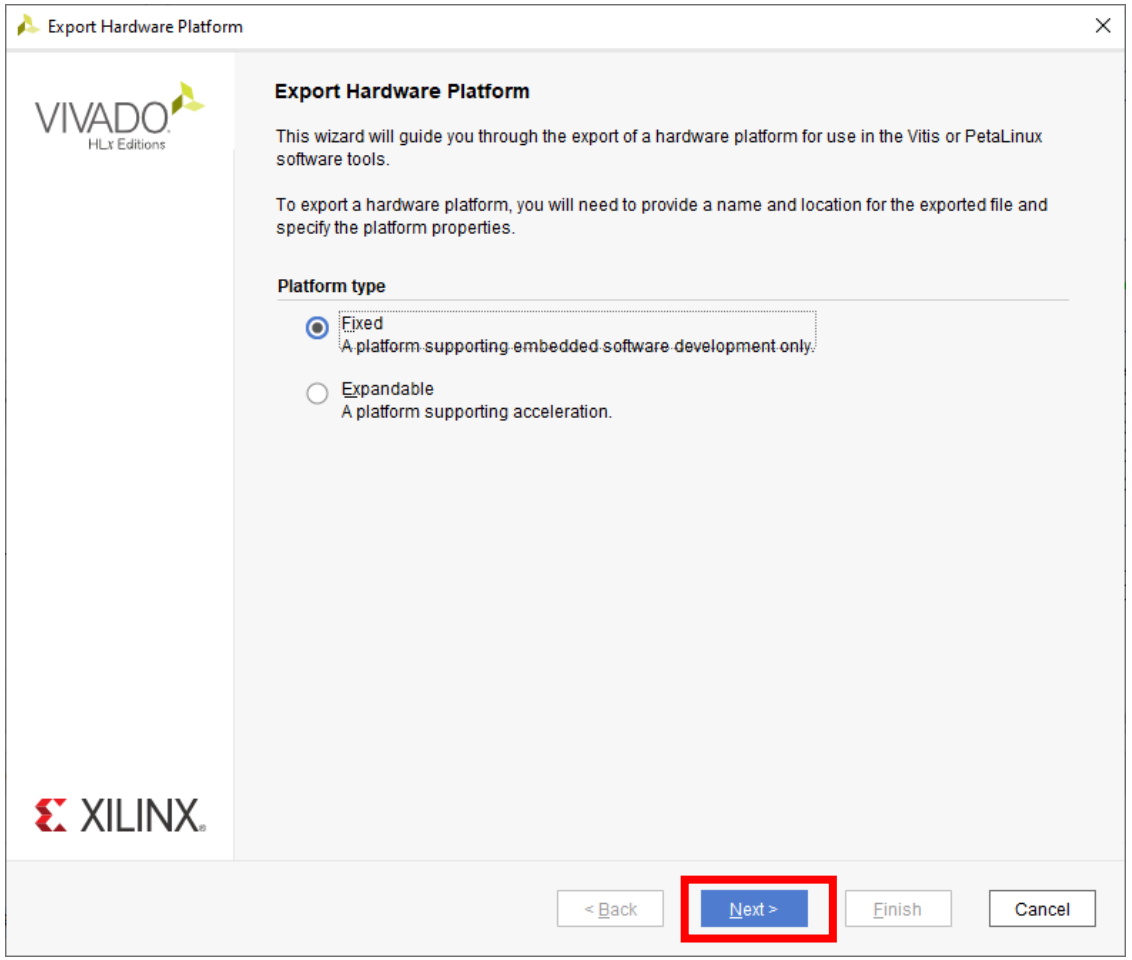
# MicroBlaze Boot From QSPI

Once the bit stream is available export the hardware.



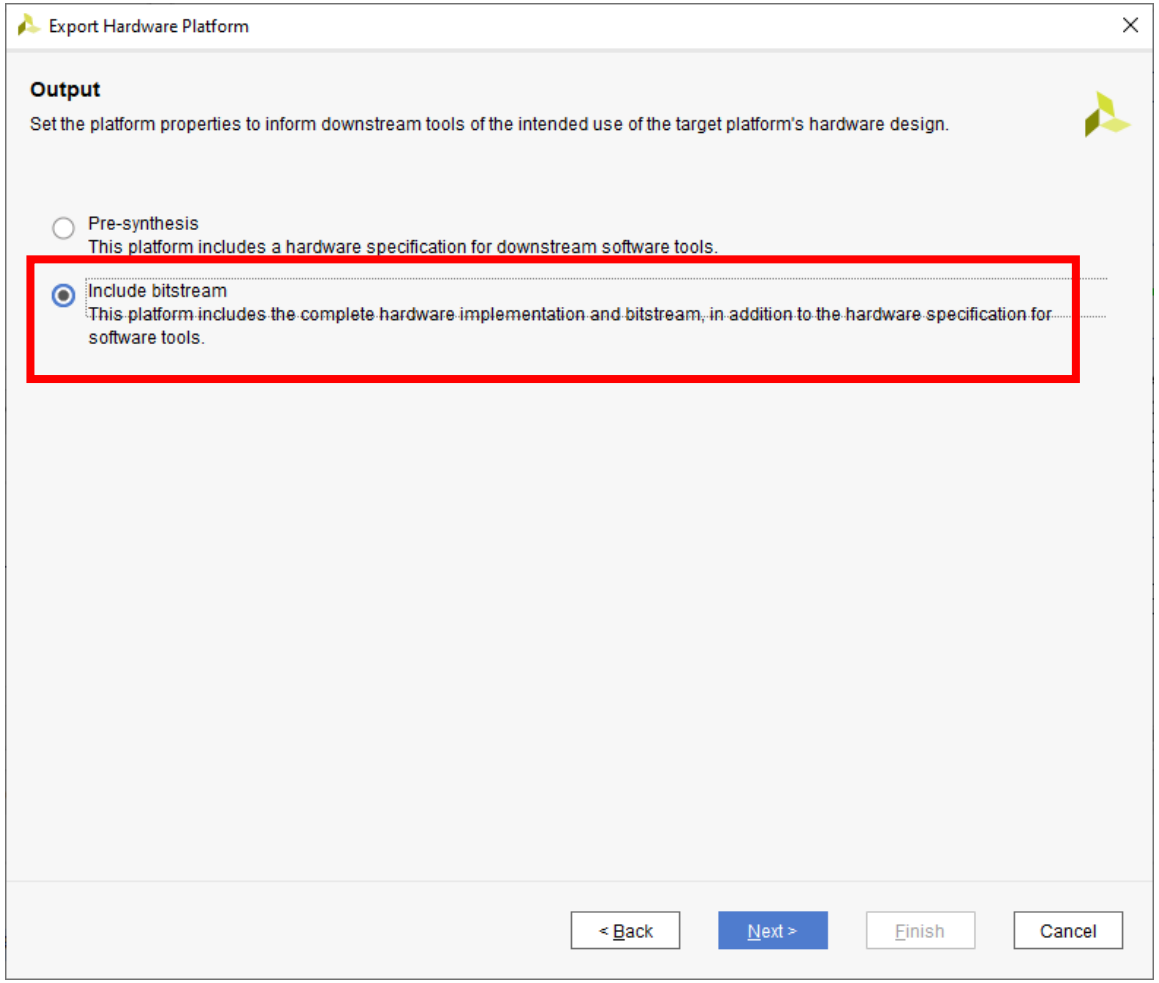
# MicroBlaze Boot From QSPI

Select the Fixed platform type



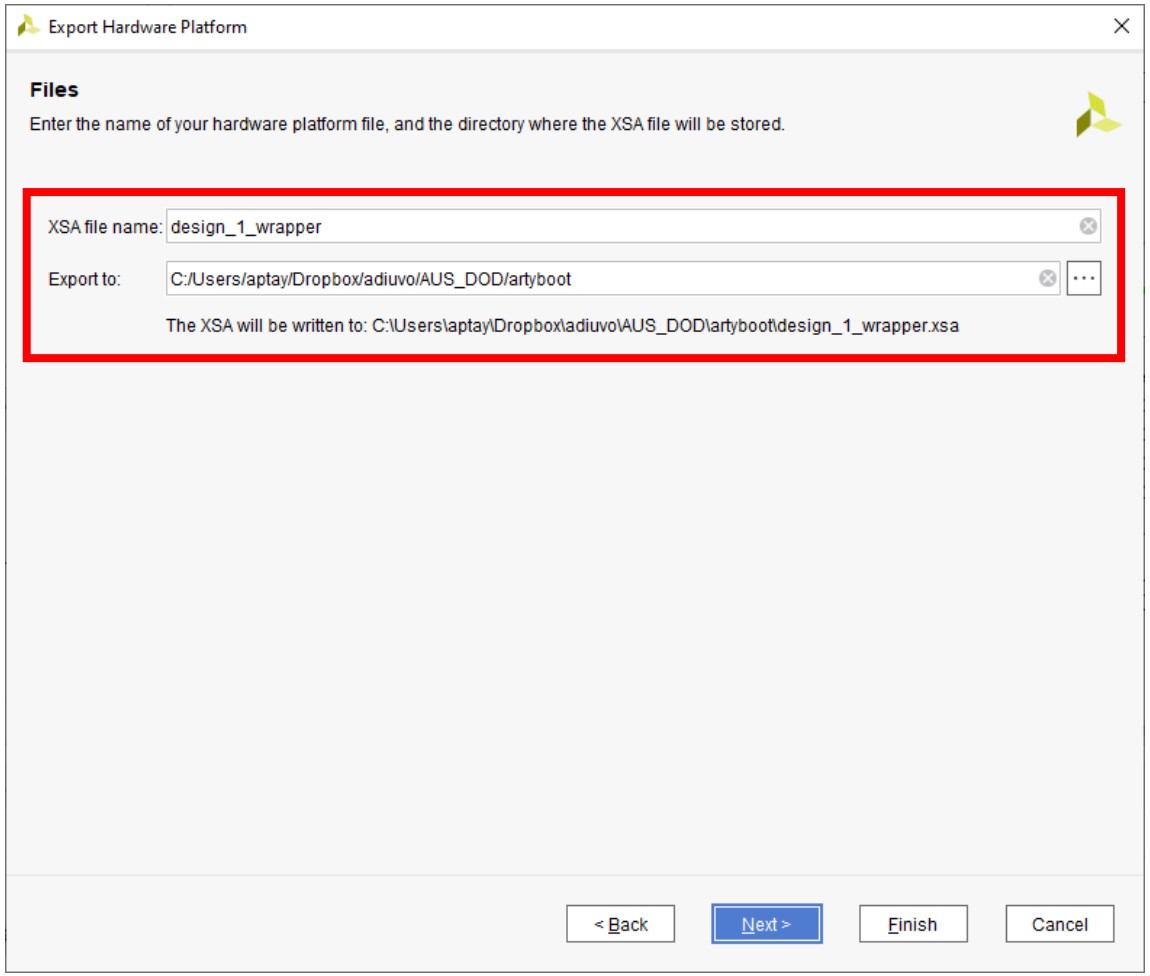
# MicroBlaze Boot From QSPI

Include the bitstream



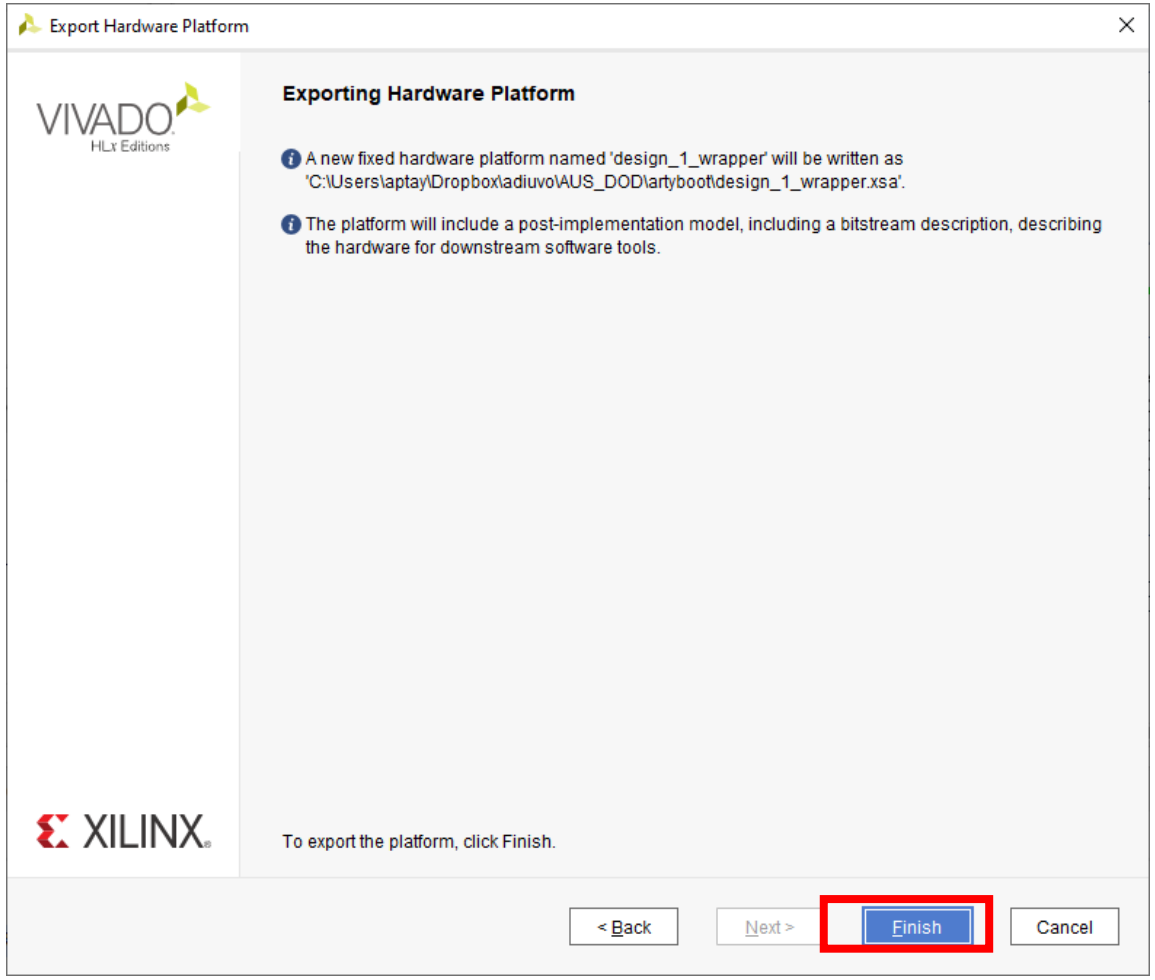
# MicroBlaze Boot From QSPI

Select a name and location for the project



# MicroBlaze Boot From QSPI

Click on Finish to export the hardware

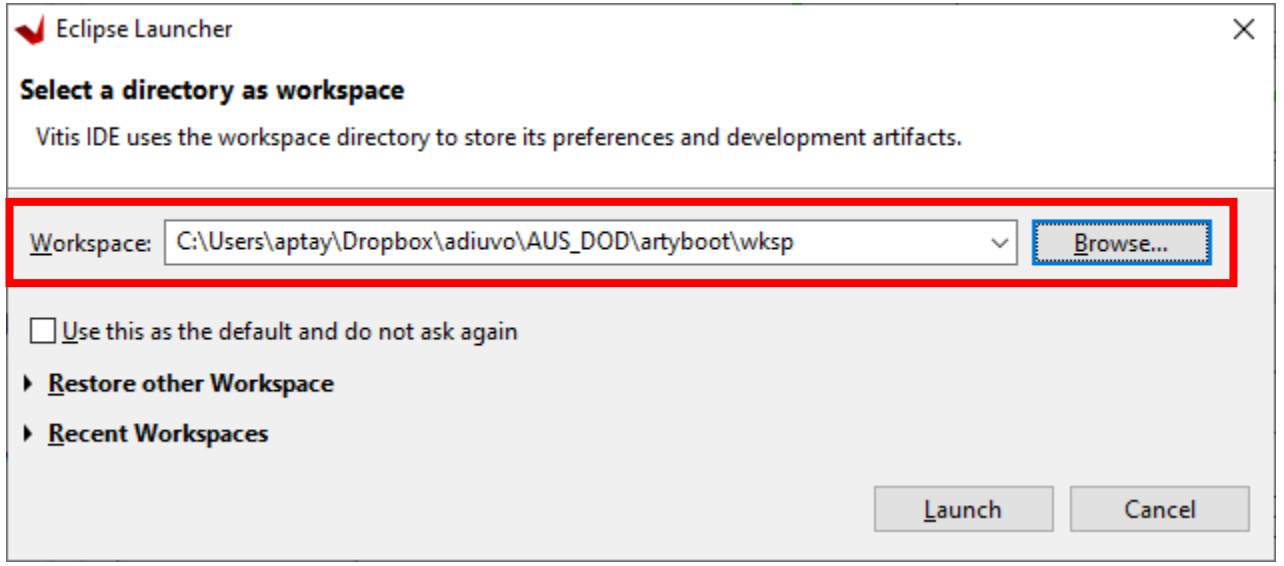


# Software Application Development in Vitis

ADIUVO
ENGINEERING AND TRAINING, LTD.

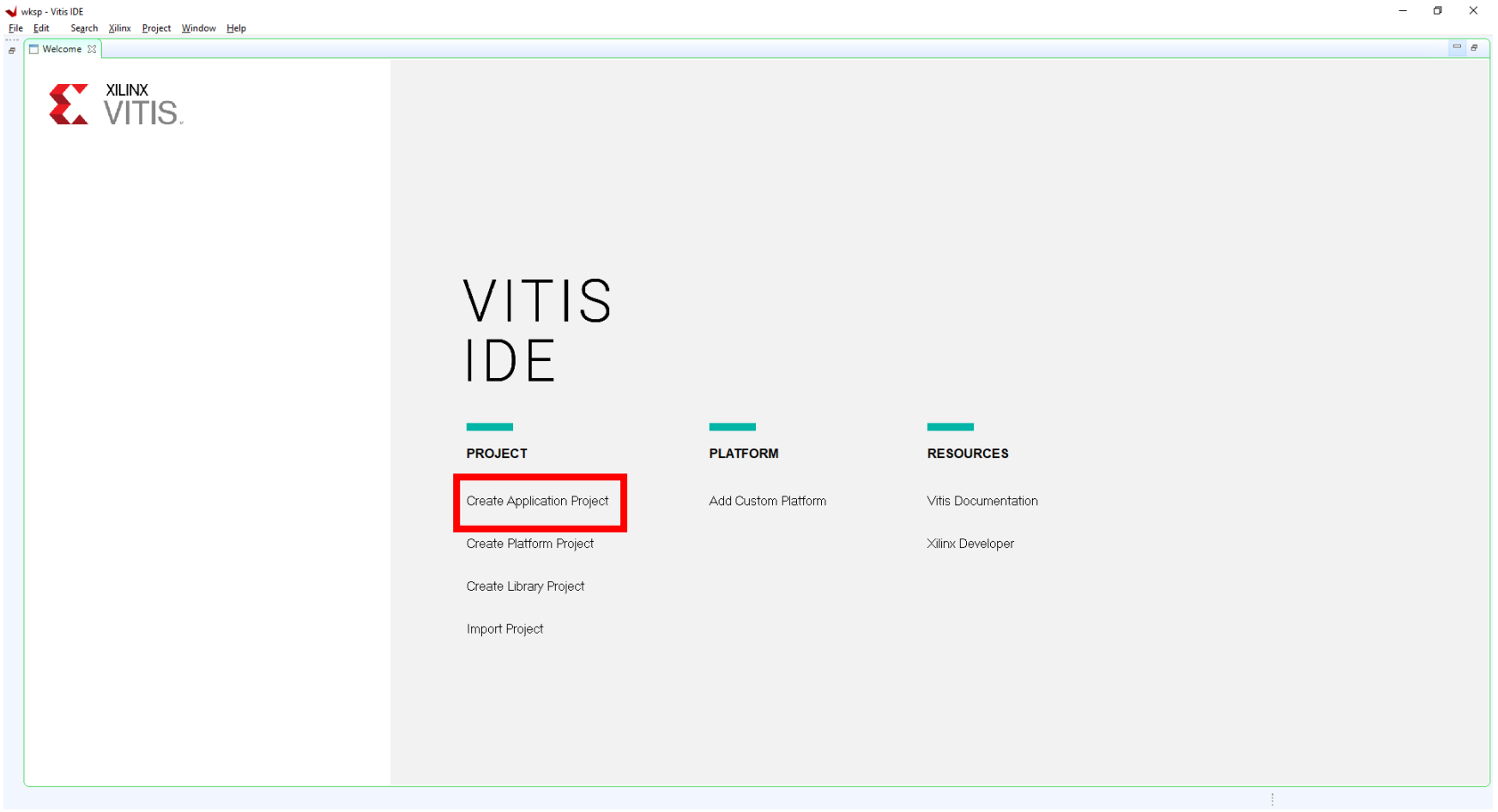
# MicroBlaze Boot From QSPI

From the Vivado tools menu launch Vitis and select a workspace



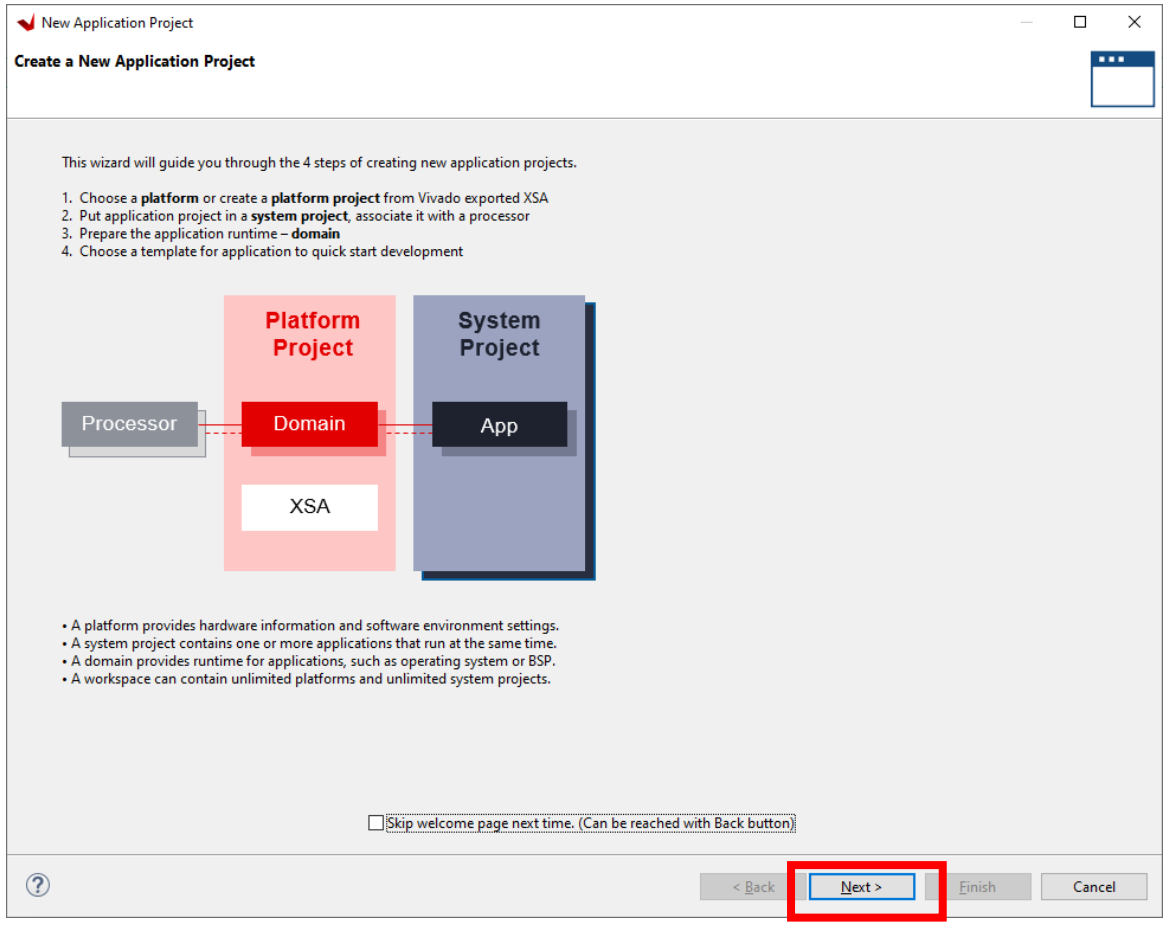
# MicroBlaze Boot From QSPI

Select create application project



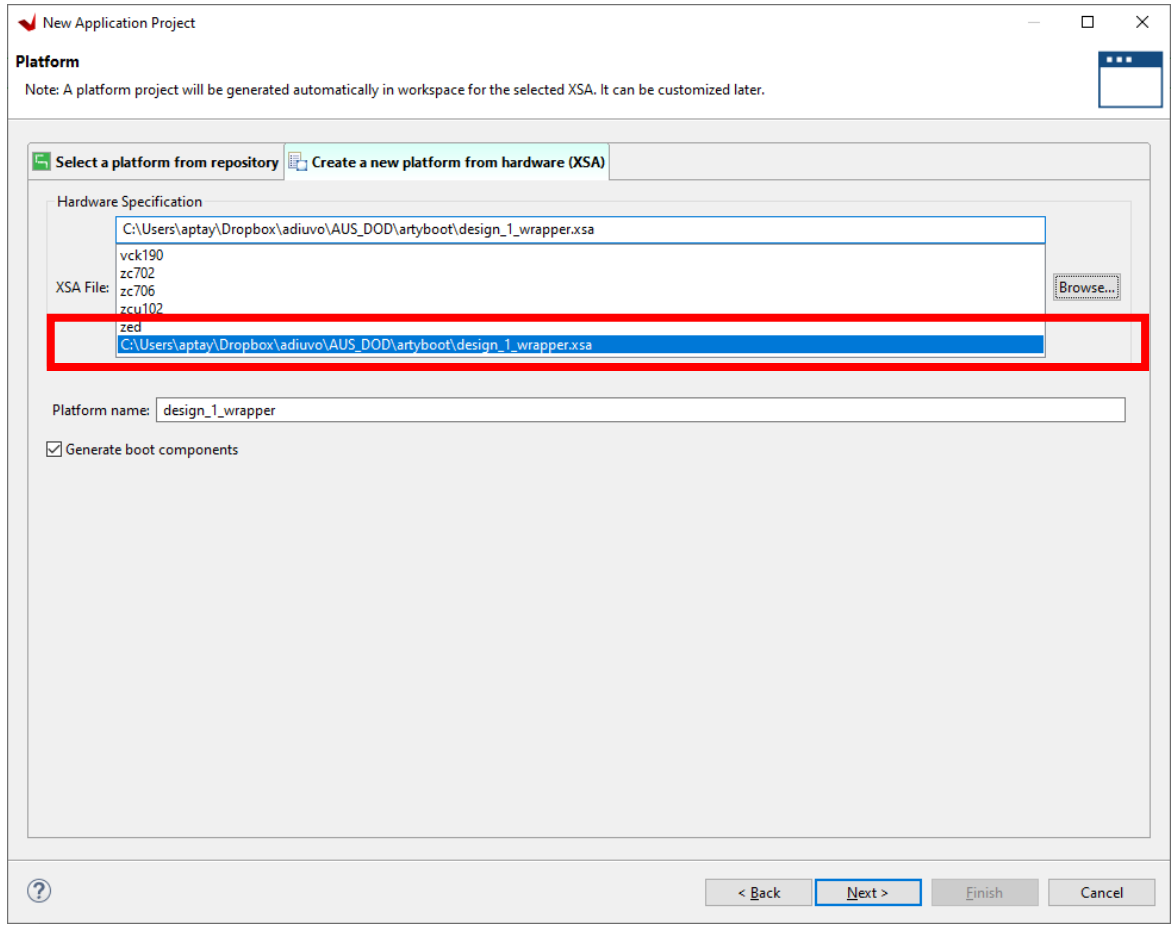
# MicroBlaze Boot From QSPI

Click Next on the dialog



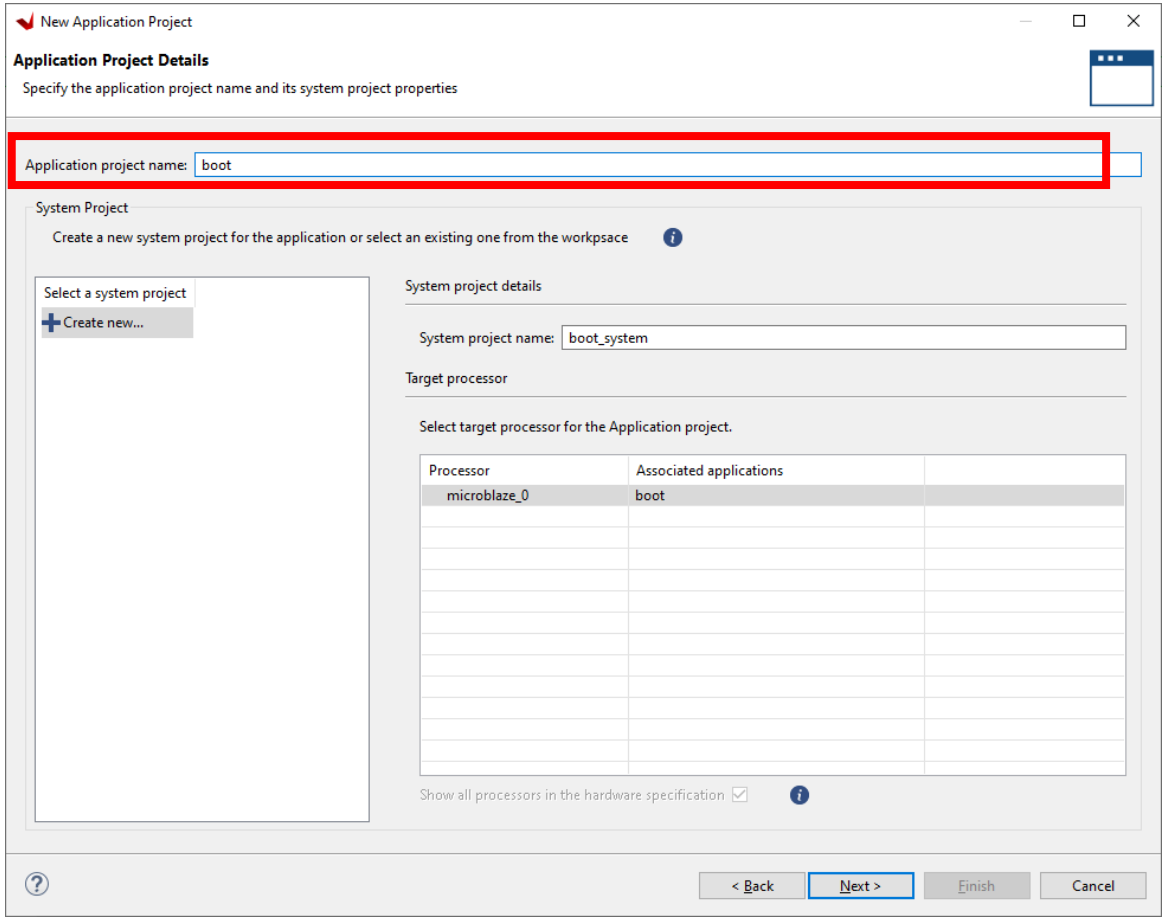
# MicroBlaze Boot From QSPI

Select the Hardware platform just exported



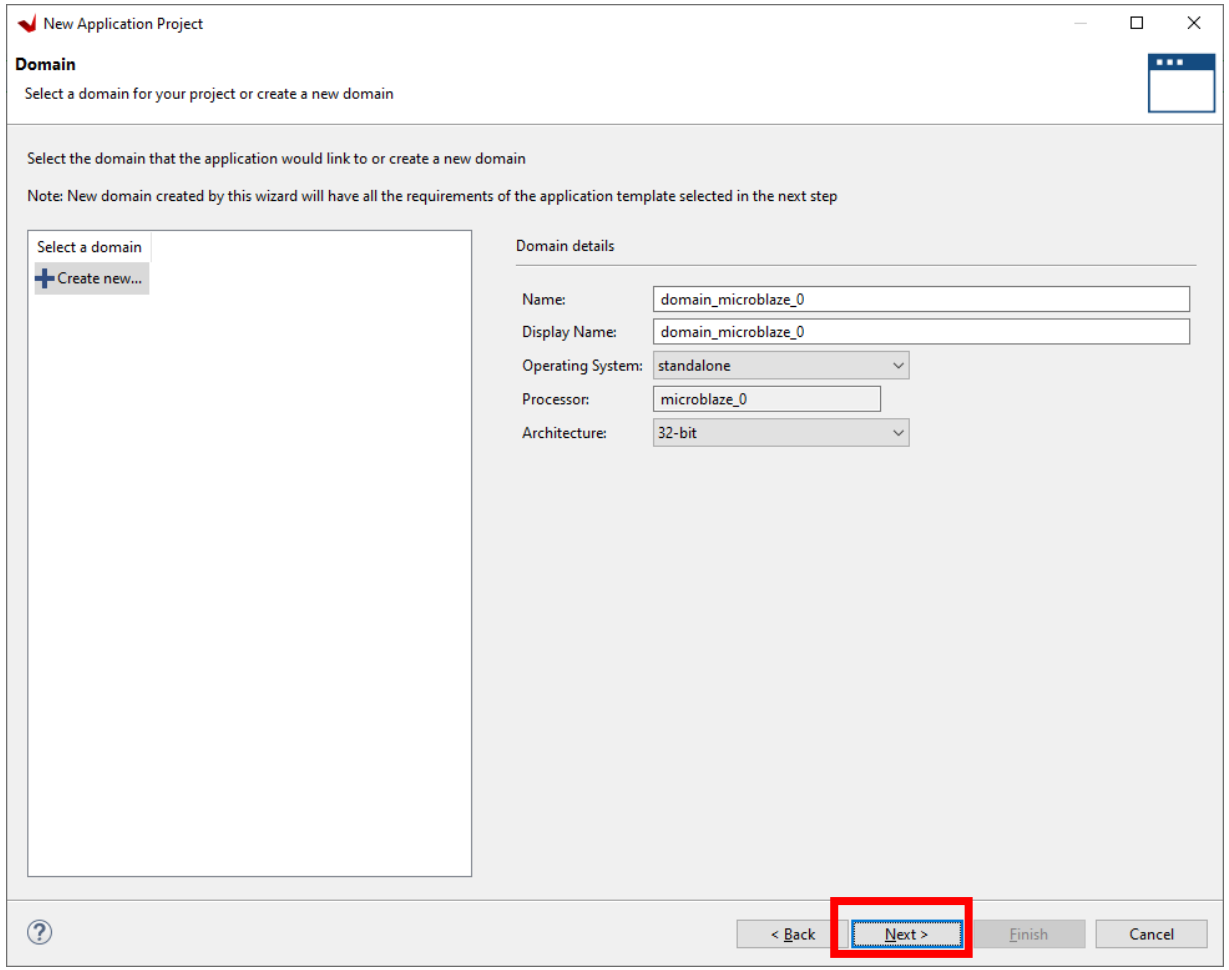
# MicroBlaze Boot From QSPI

Select the name of the project



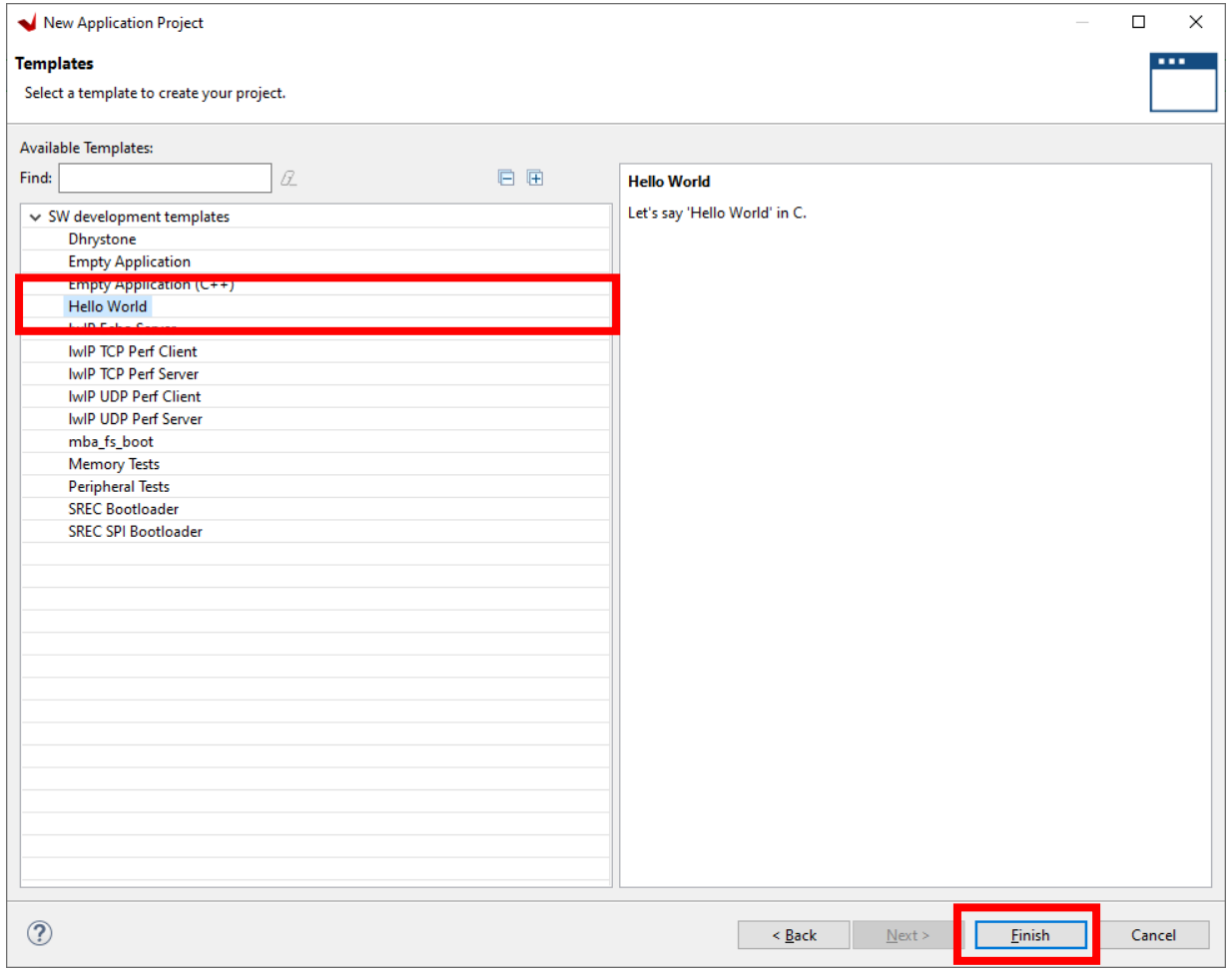
# MicroBlaze Boot From QSPI

Leave the domain unchanged



# MicroBlaze Boot From QSPI

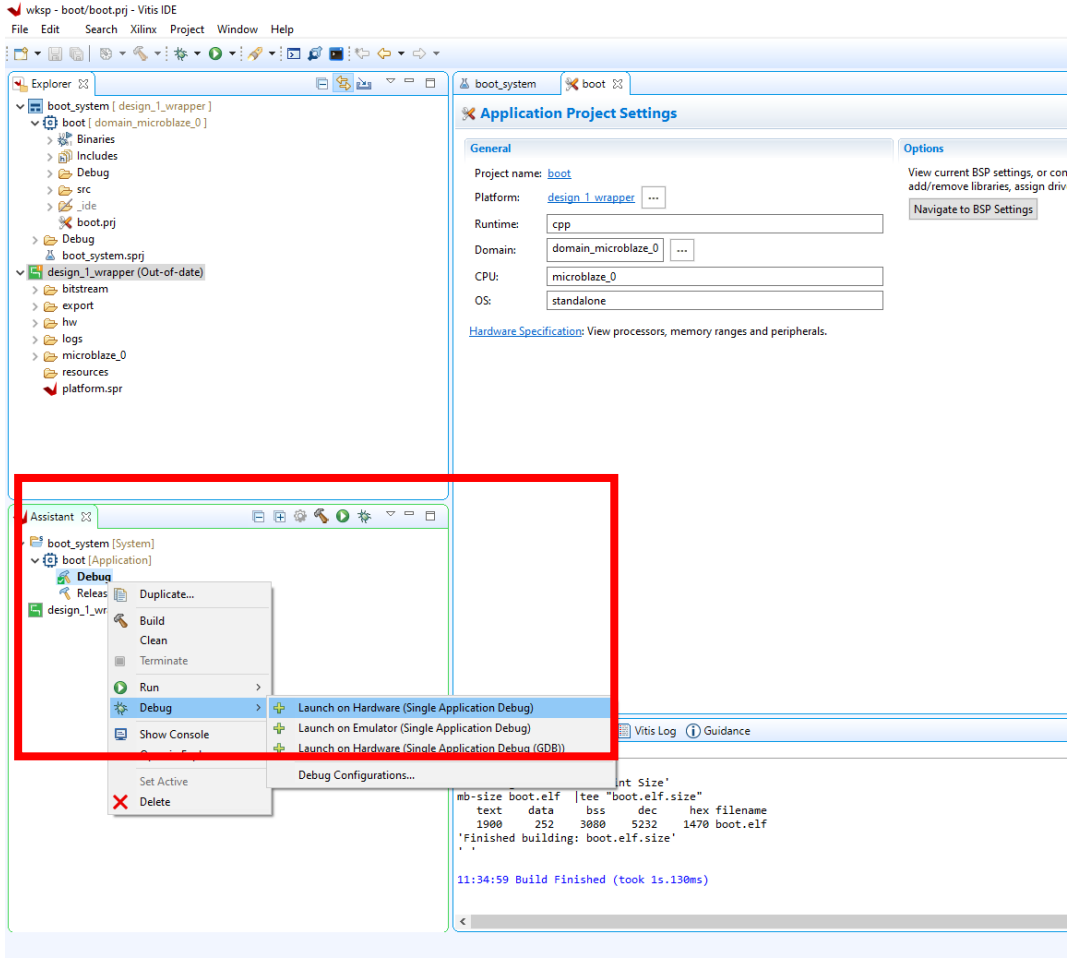
## Select the Hello World Project



# MicroBlaze Boot From QSPI

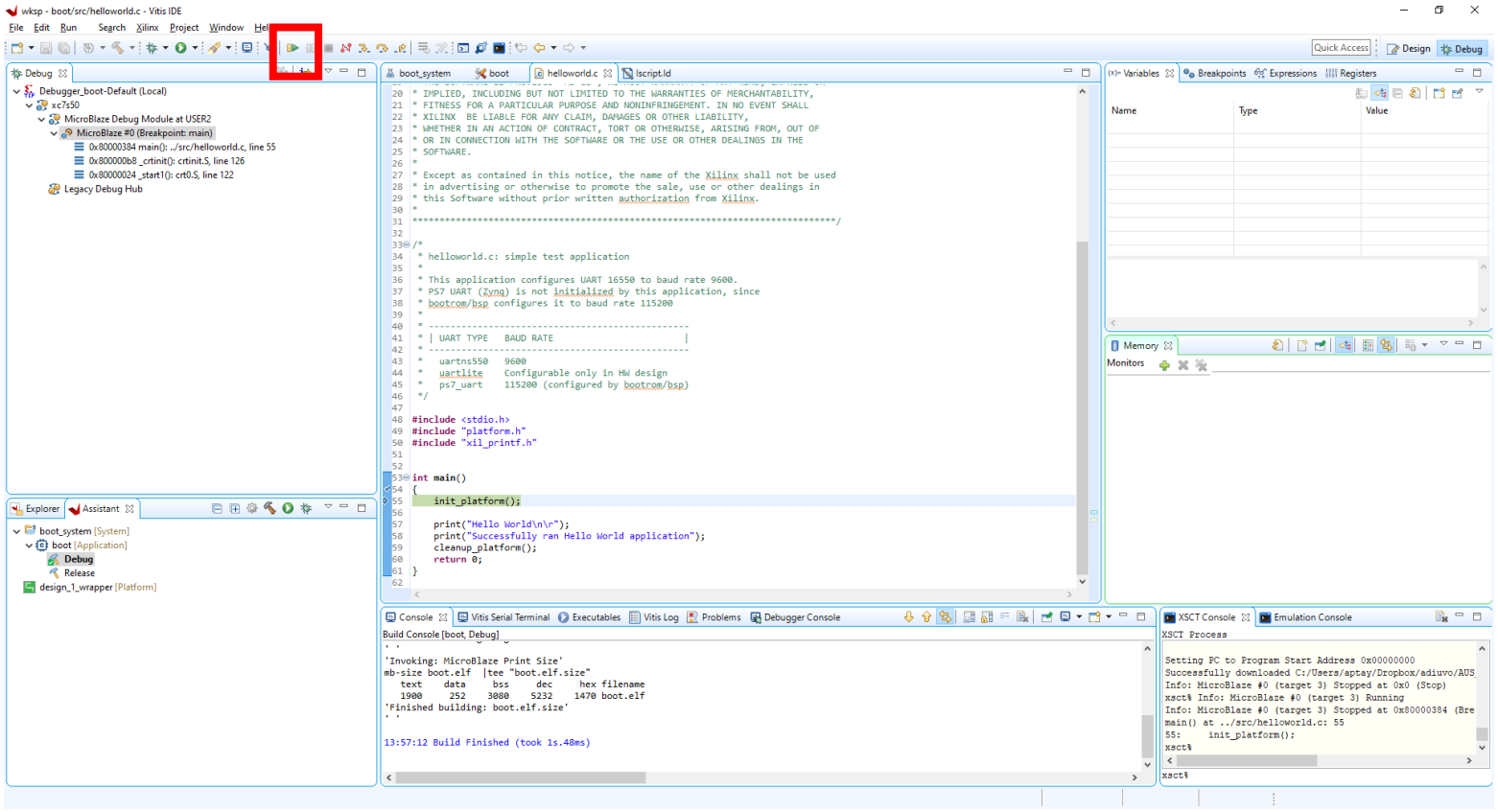
With the project created, debug the application on the hardware to ensure the MicroBlaze is running correctly from DDR before we build the final applications.

In the assistant select <project name> Debug and launch on hardware



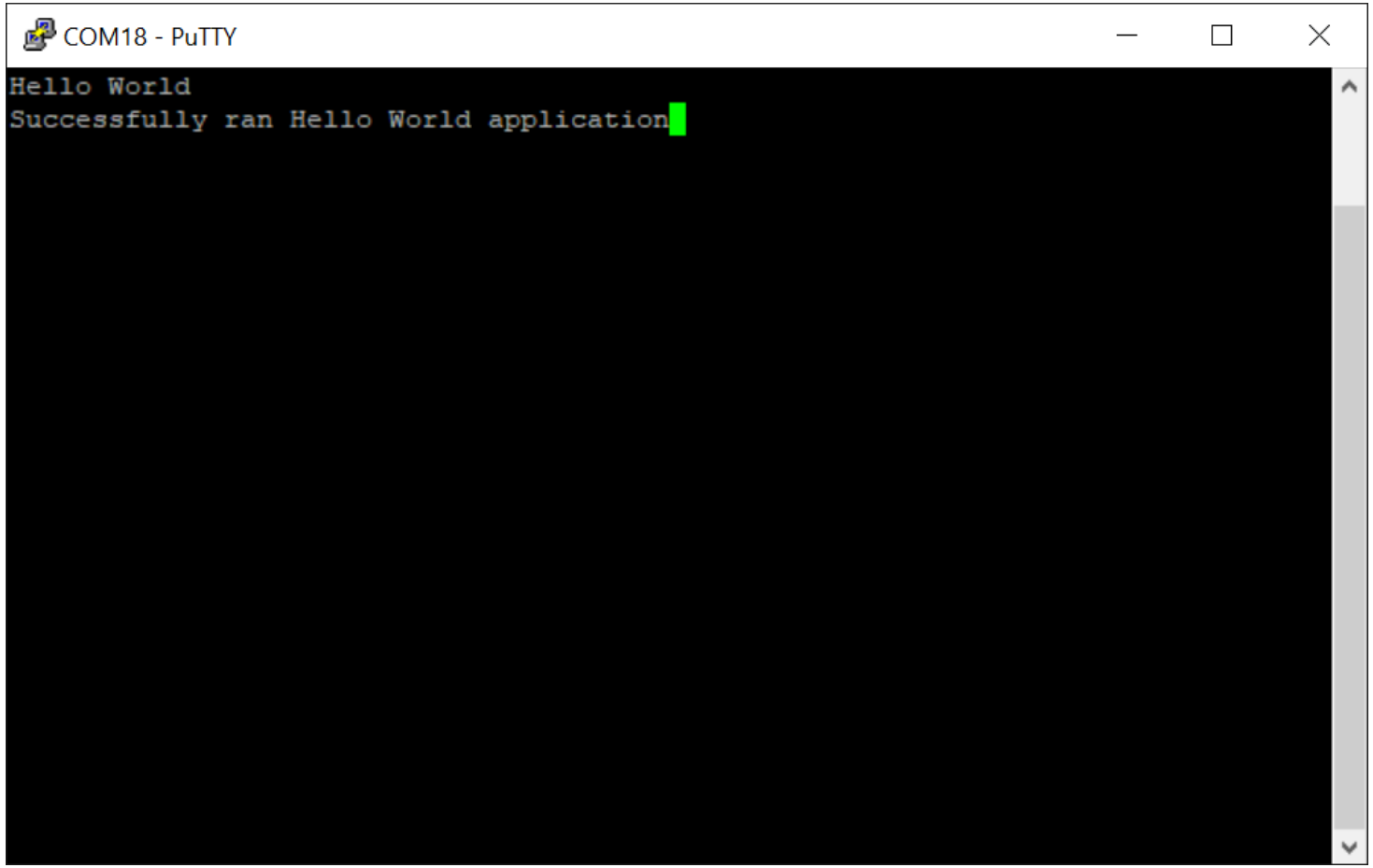
# MicroBlaze Boot From QSPI

The debugger will launch and program the Arty S7-50 board, execution will be paused.  Ensure a serial terminal is connected 9600n1 click run



# MicroBlaze Boot From QSPI

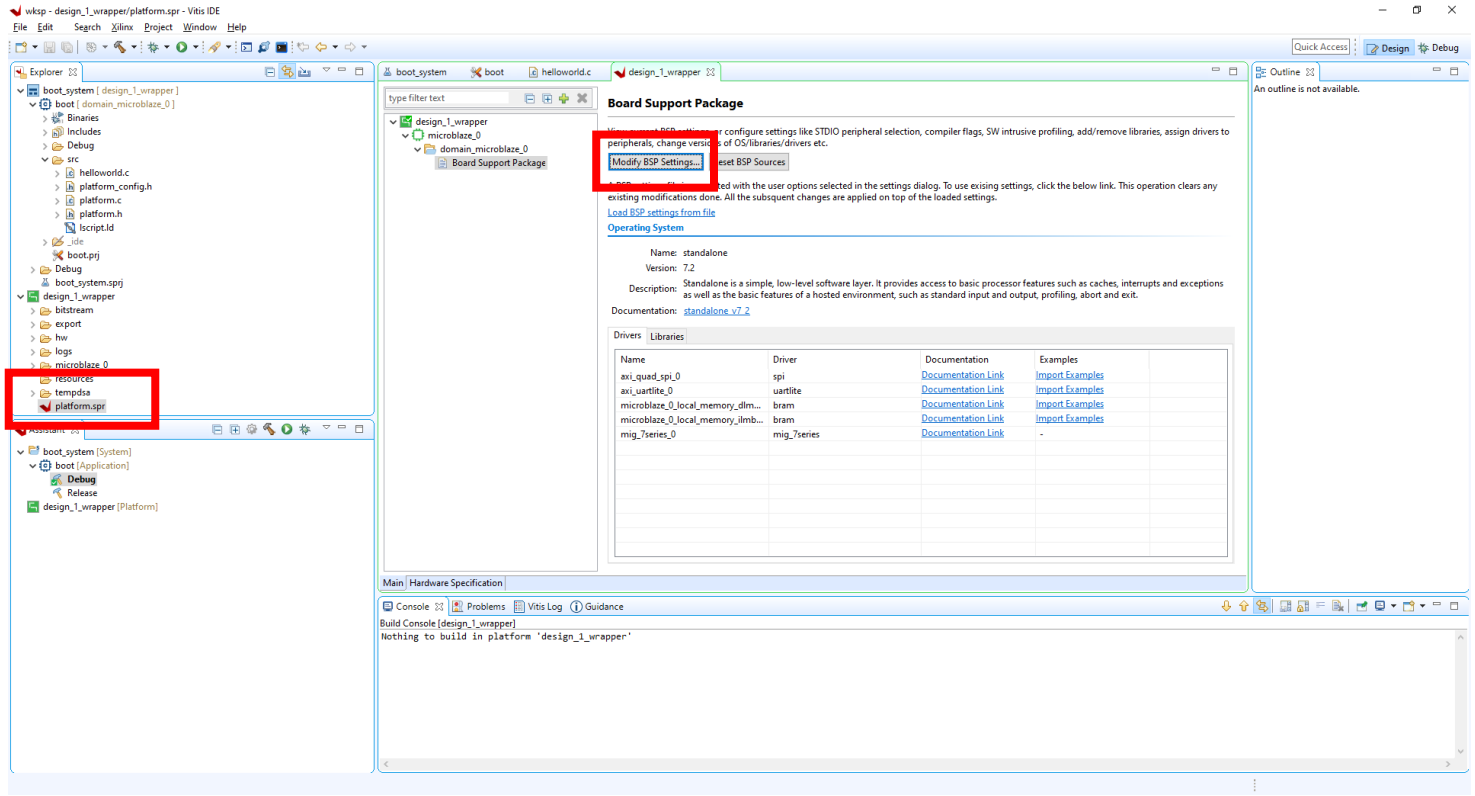
This should show a simple hello world in the terminal if running correctly



# MicroBlaze Boot From QSPI

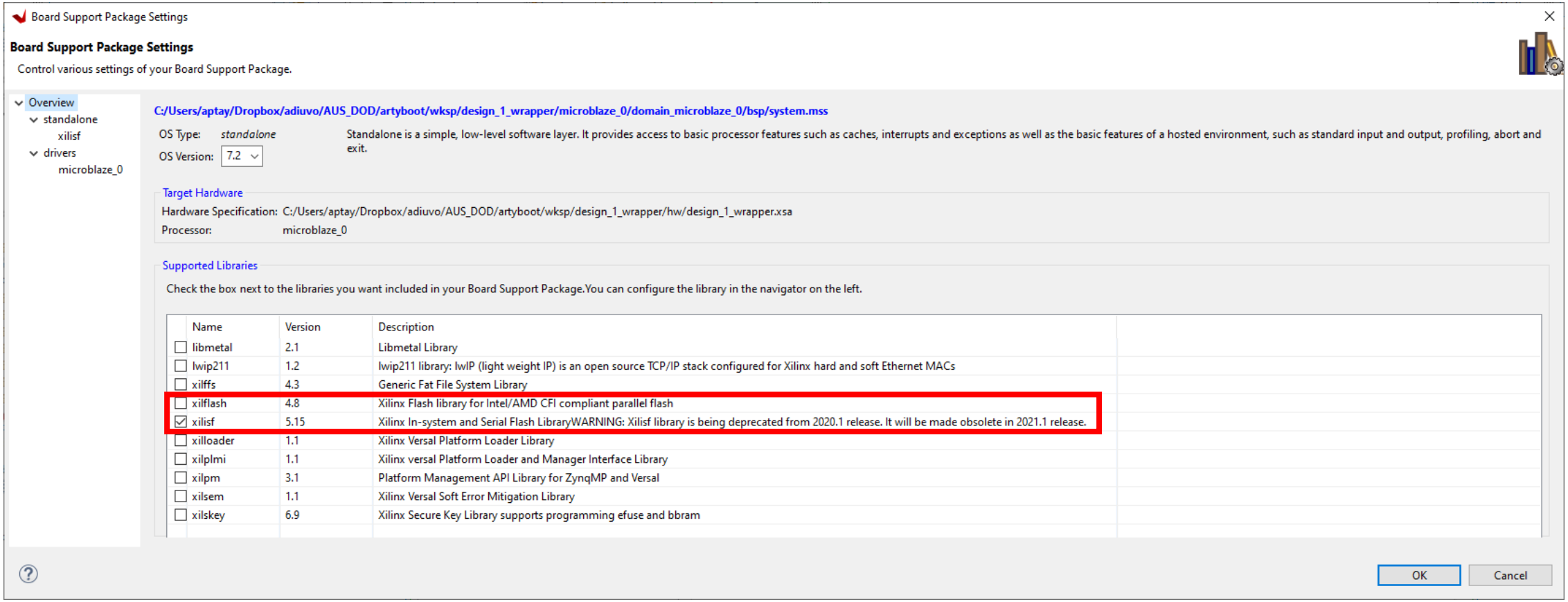
To be able to access the QSPI. We need to add in the Xilinx Serial Flash Library

Double click on the platform.spr file and modify the BSP settings



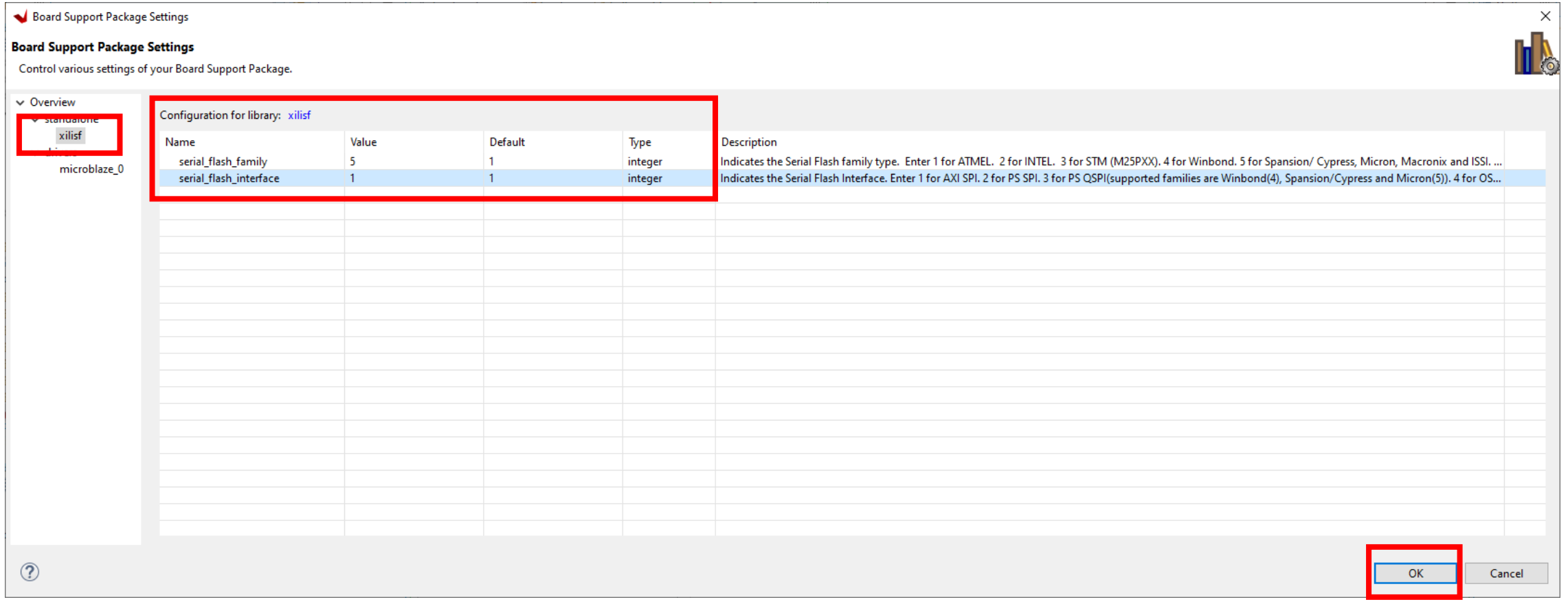
# MicroBlaze Boot From QSPI

In the BSP settings add in the XilSF option



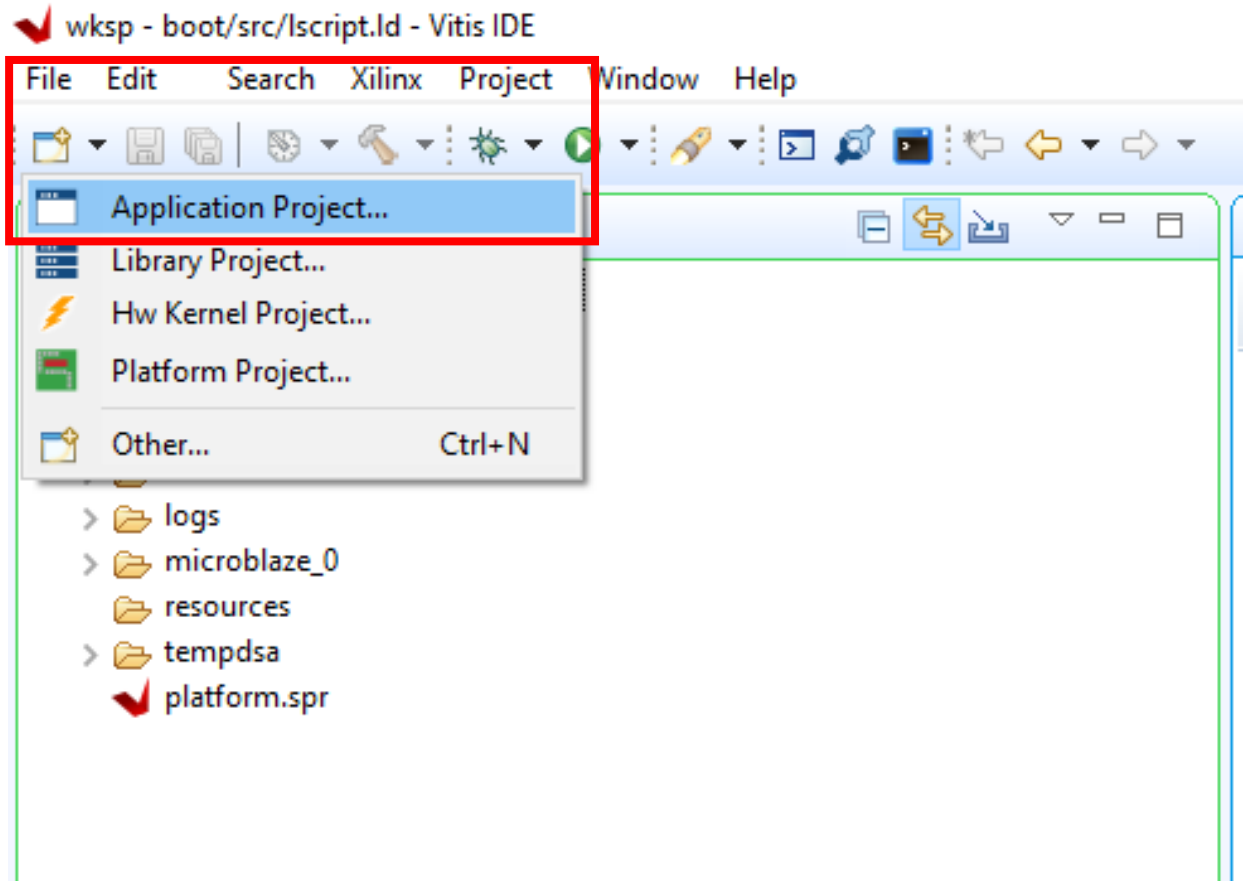
# MicroBlaze Boot From QSPI

Click on xilisf and update the values as necessary for the device and interface. The ArtyS7 uses a spansion device so the type is 5 and interface is 1



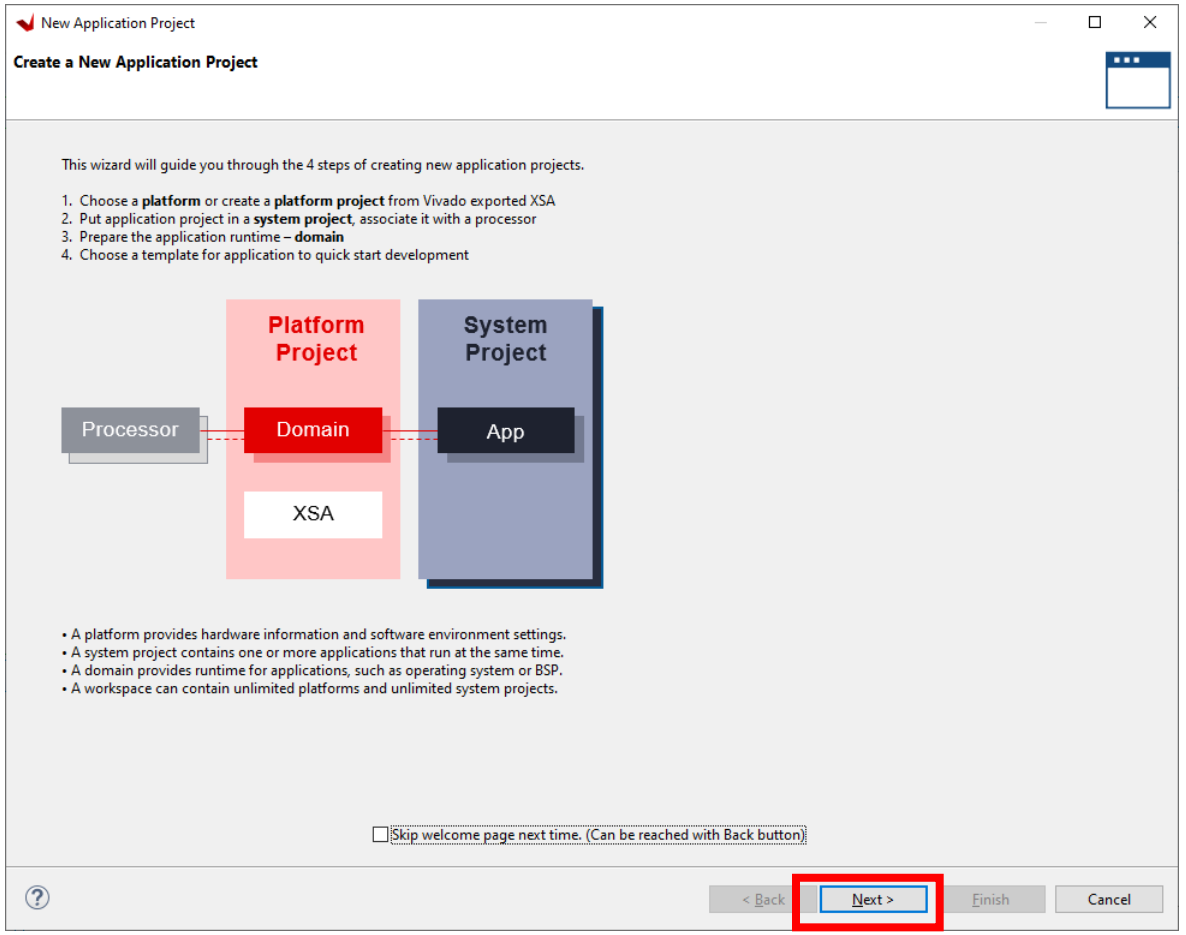
# MicroBlaze Boot From QSPI

Create a new application in Vitis



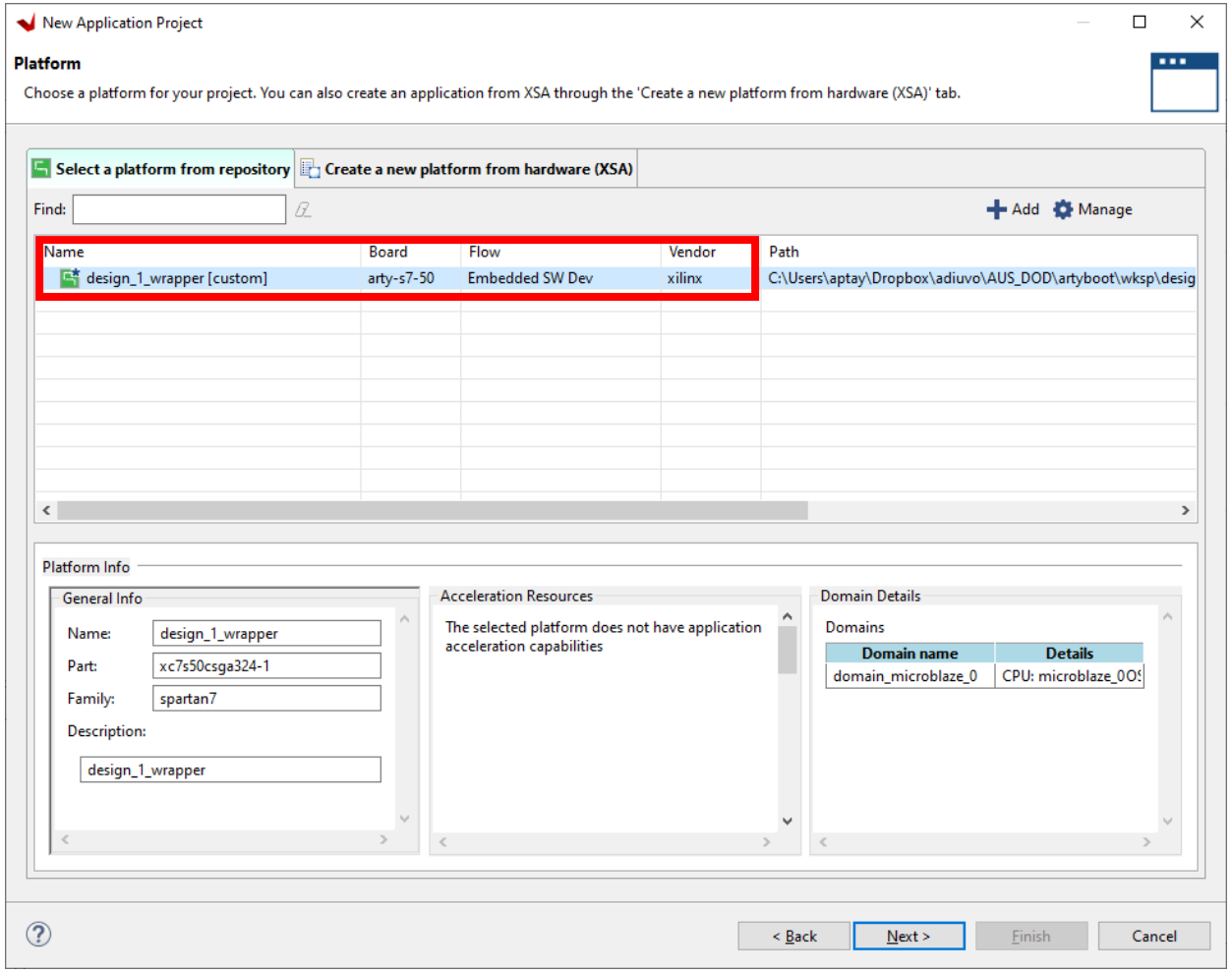
# MicroBlaze Boot From QSPI

Click on next on the application wizard



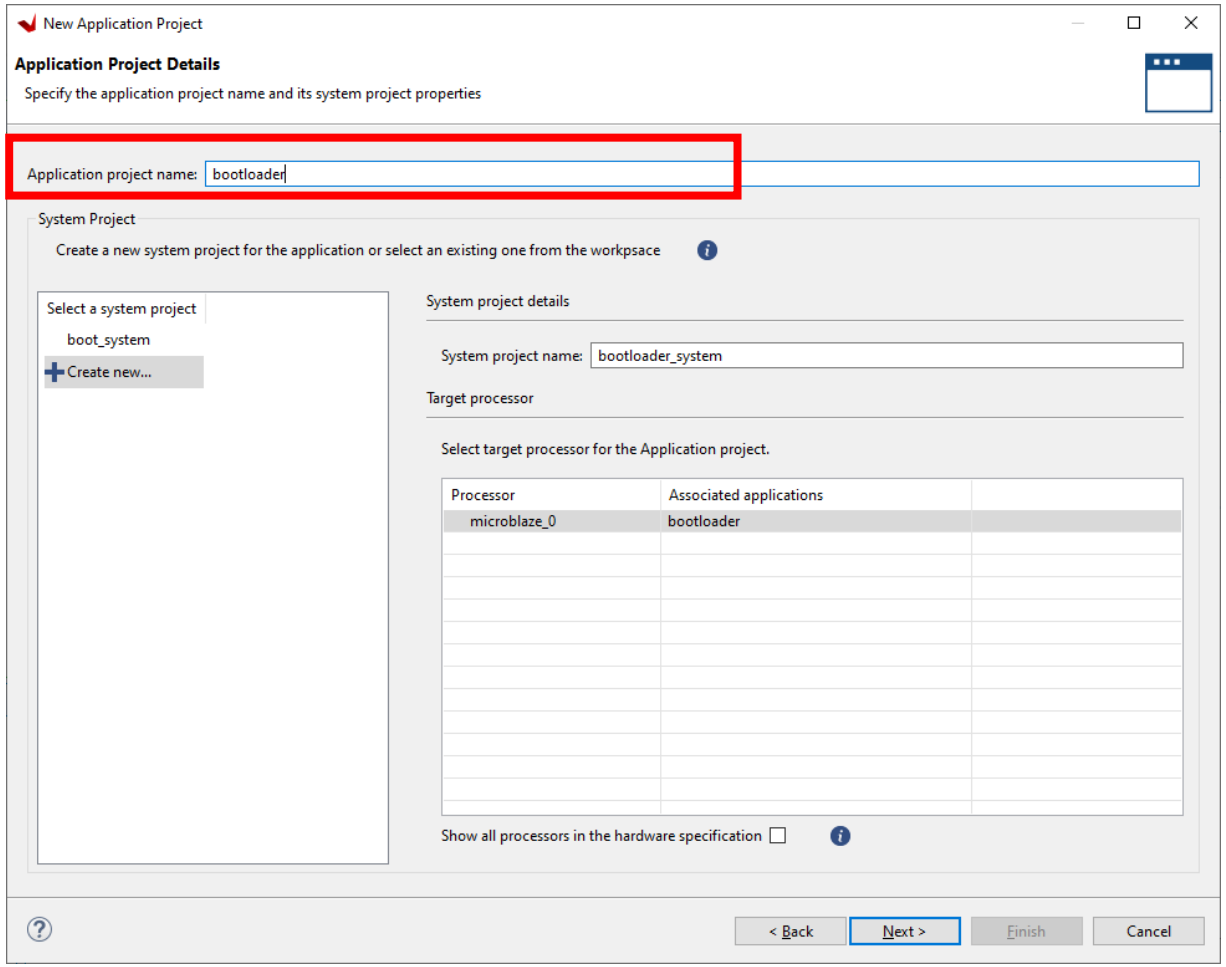
# MicroBlaze Boot From QSPI

Select the existing platform



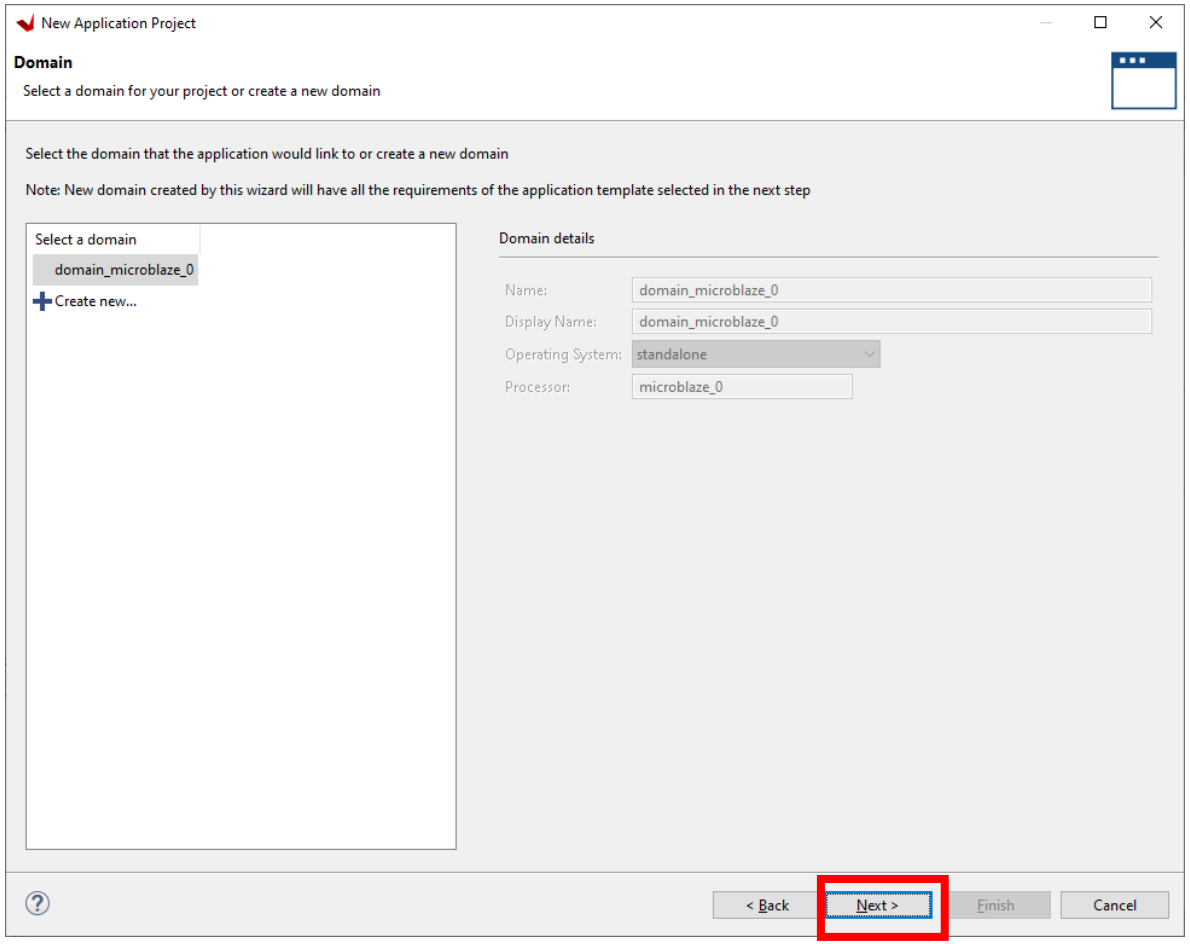
# MicroBlaze Boot From QSPI

Name the project



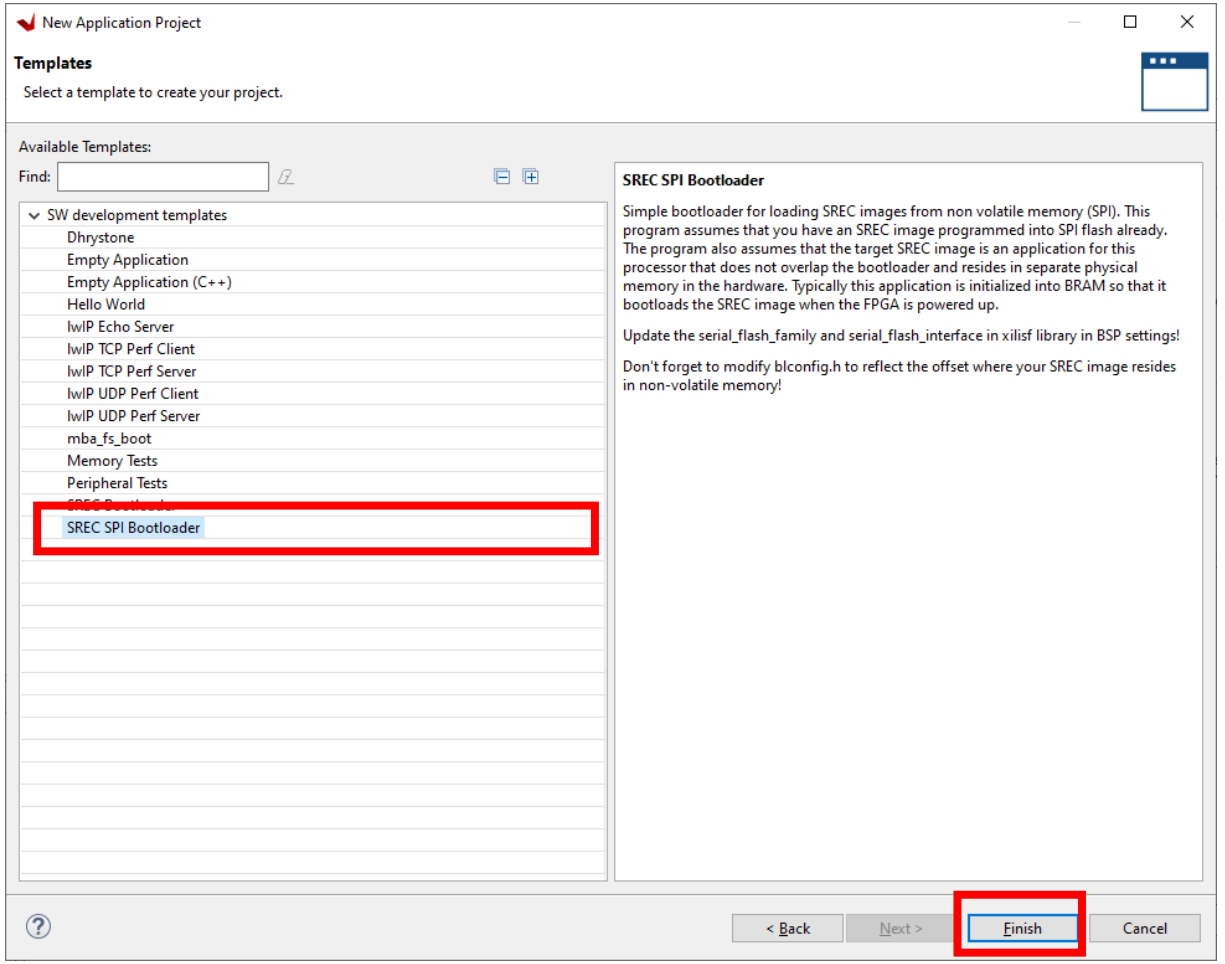
# MicroBlaze Boot From QSPI

Leave the domain unchanged



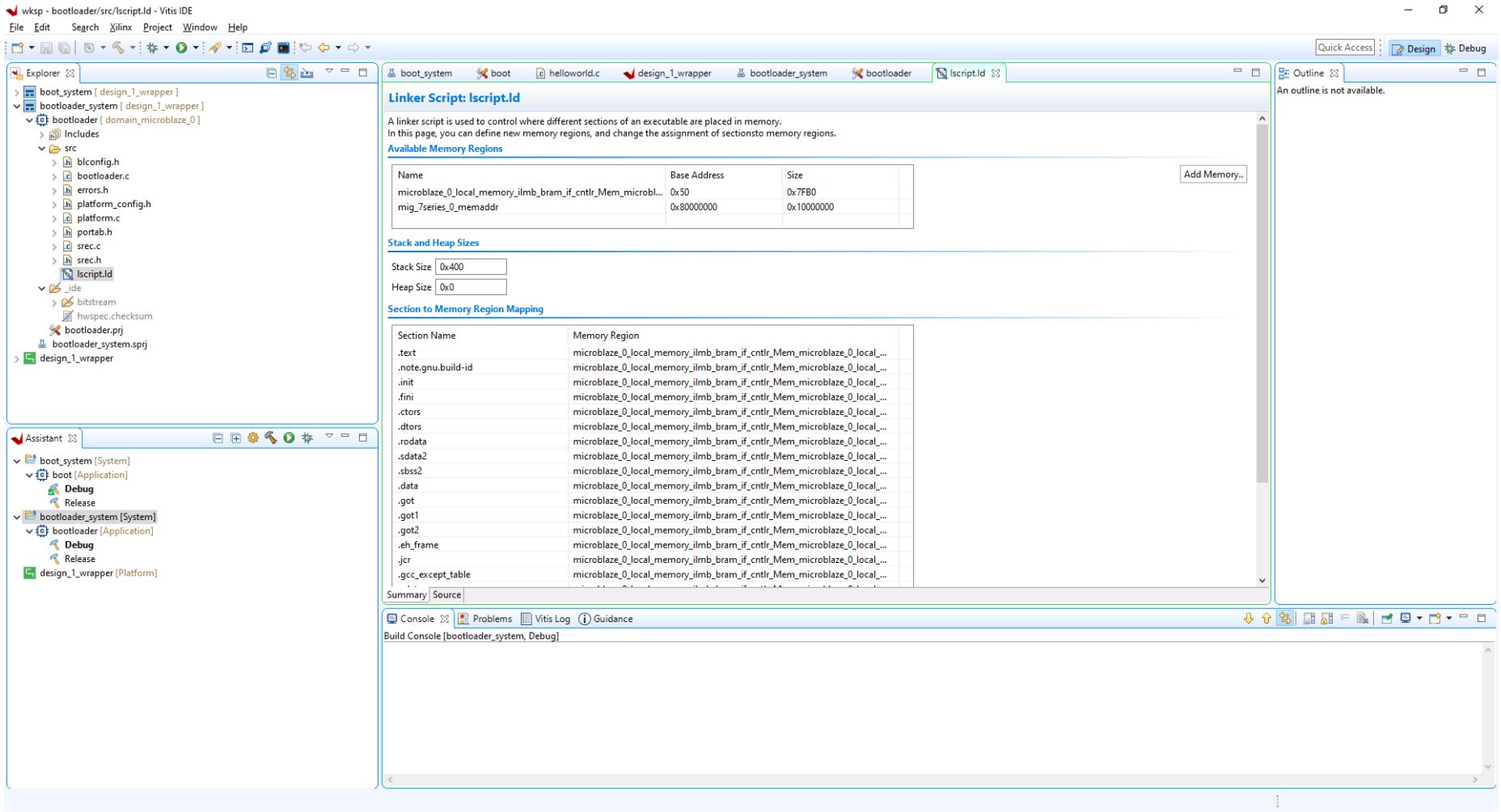
# MicroBlaze Boot From QSPI

Select the SREC SPI Bootloader – click finish



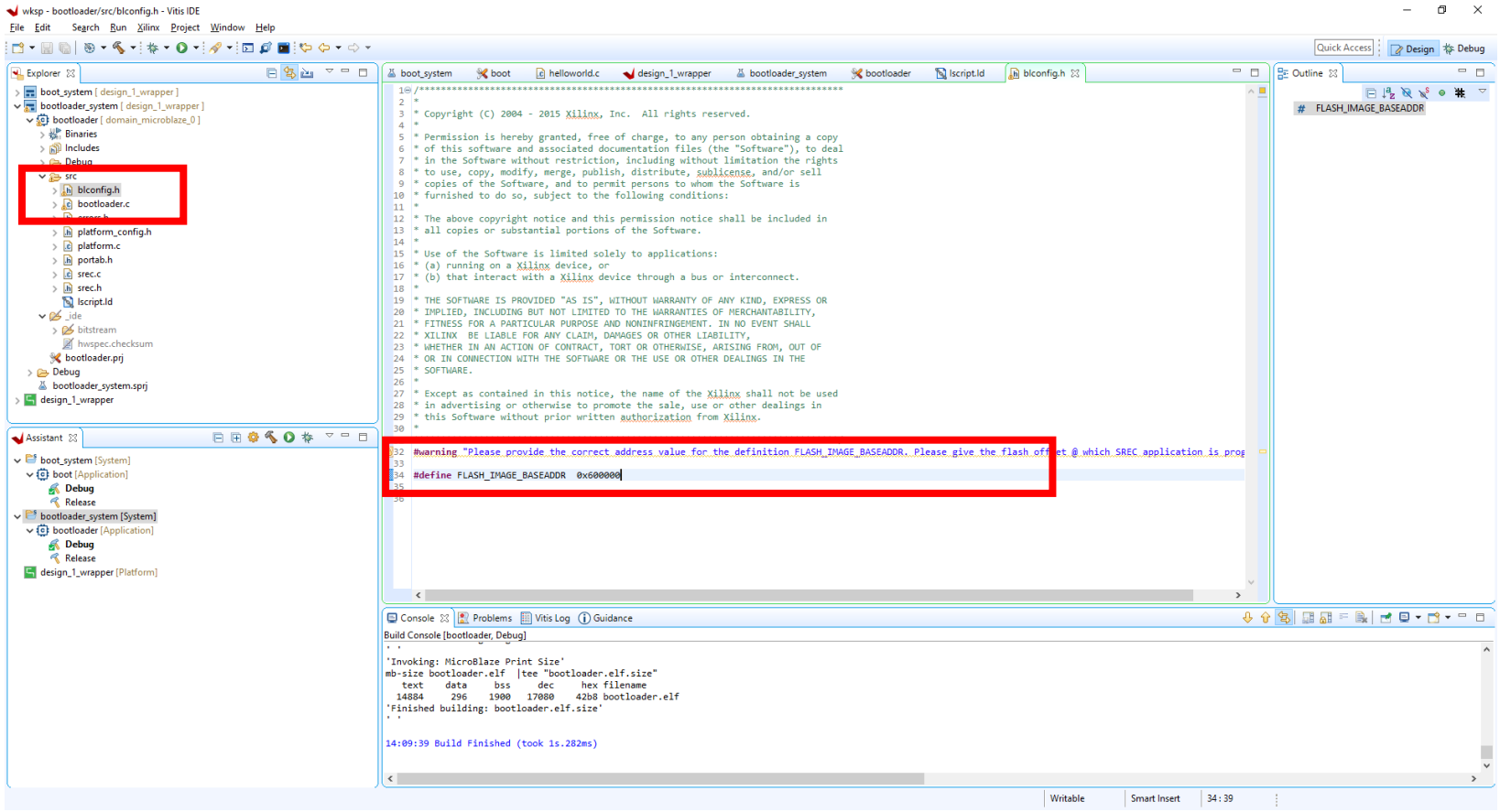
# MicroBlaze Boot From QSPI

Examine the SREC linker script and observe it runs from BRAM



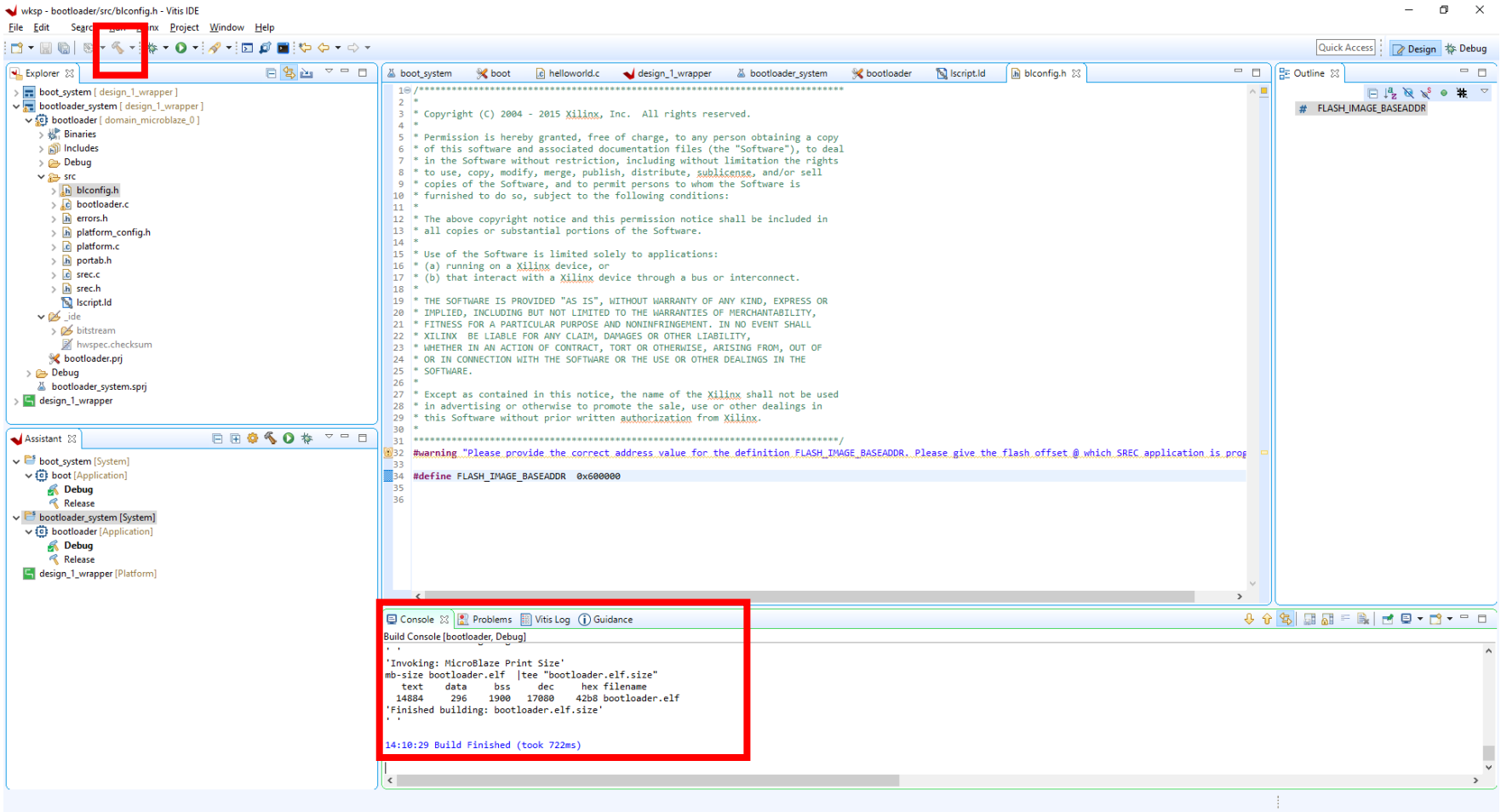
# MicroBlaze Boot From QSPI

Open the blconfig.h and set the location of the program in the QSPI memory to be loaded into DDR.



# MicroBlaze Boot From QSPI

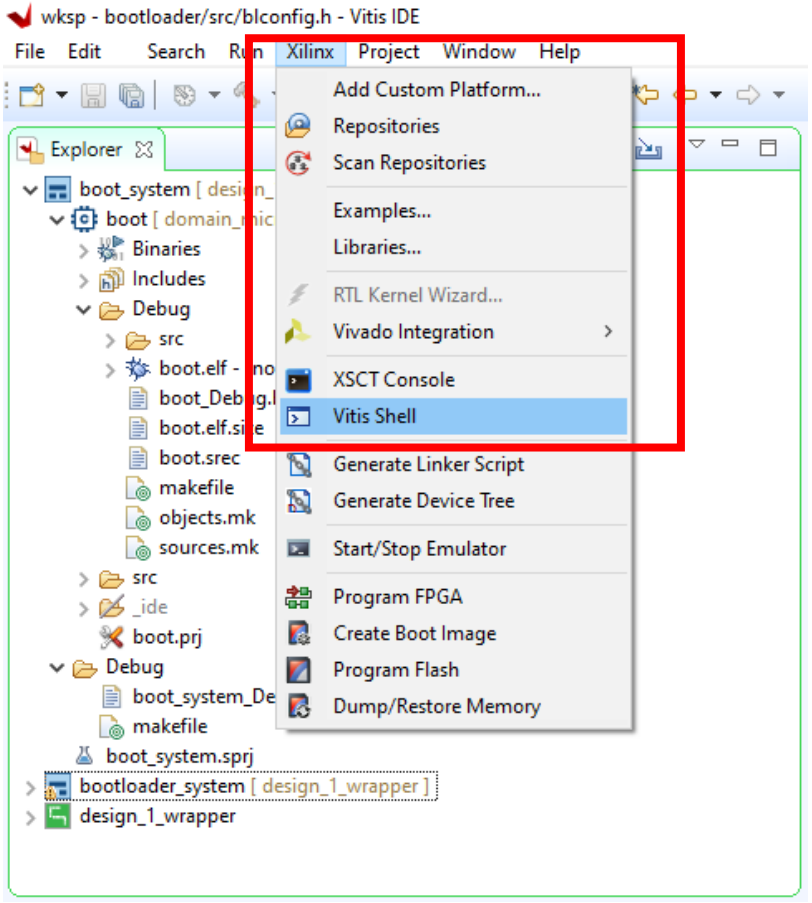
Build the design – we now have ELF for the bootloader and hello world



# MicroBlaze Boot From QSPI

From the Xilinx Menu open the Vitis Shell.

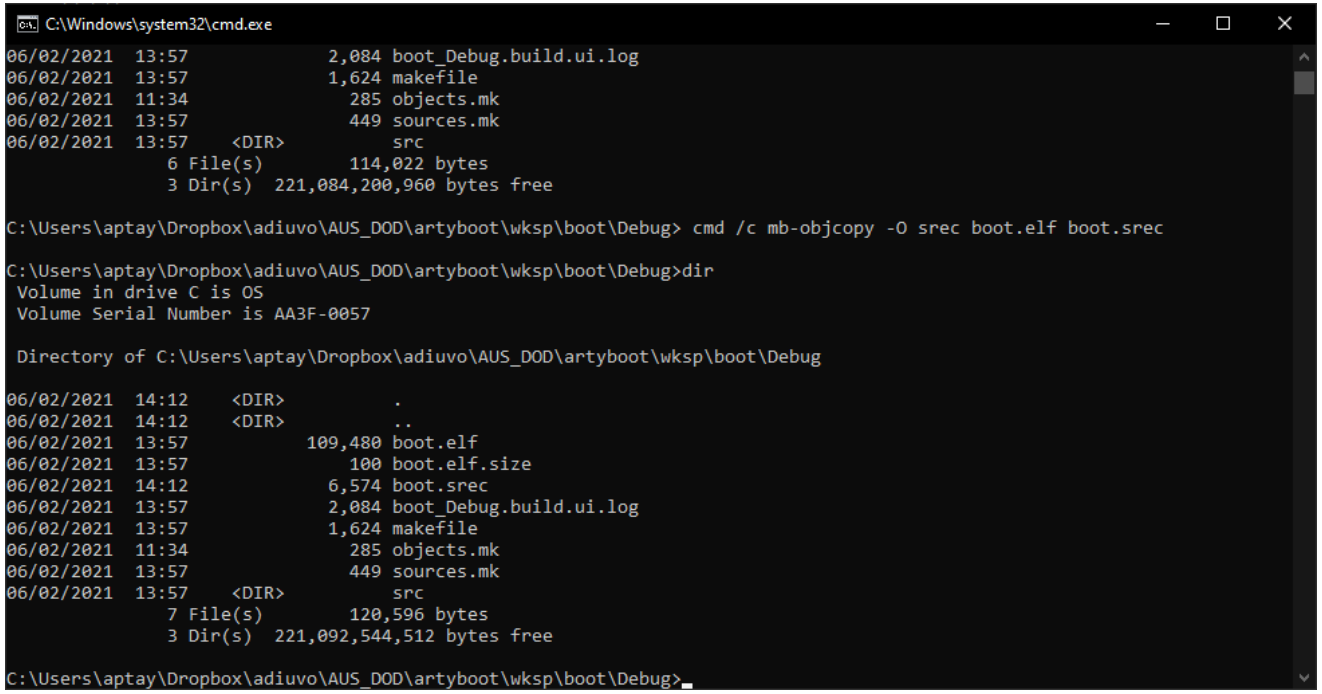
This will allow us to convert the application (hello world) elf into S Record format for loading into QSPI memory



# MicroBlaze Boot From QSPI

In the Vitis Shell change directory into the folder of the application you wish to convert to SREC – This is the application you will run from DDR enter the command

cmd /c mb-objcopy -O srec <app>.elf <app>.srec

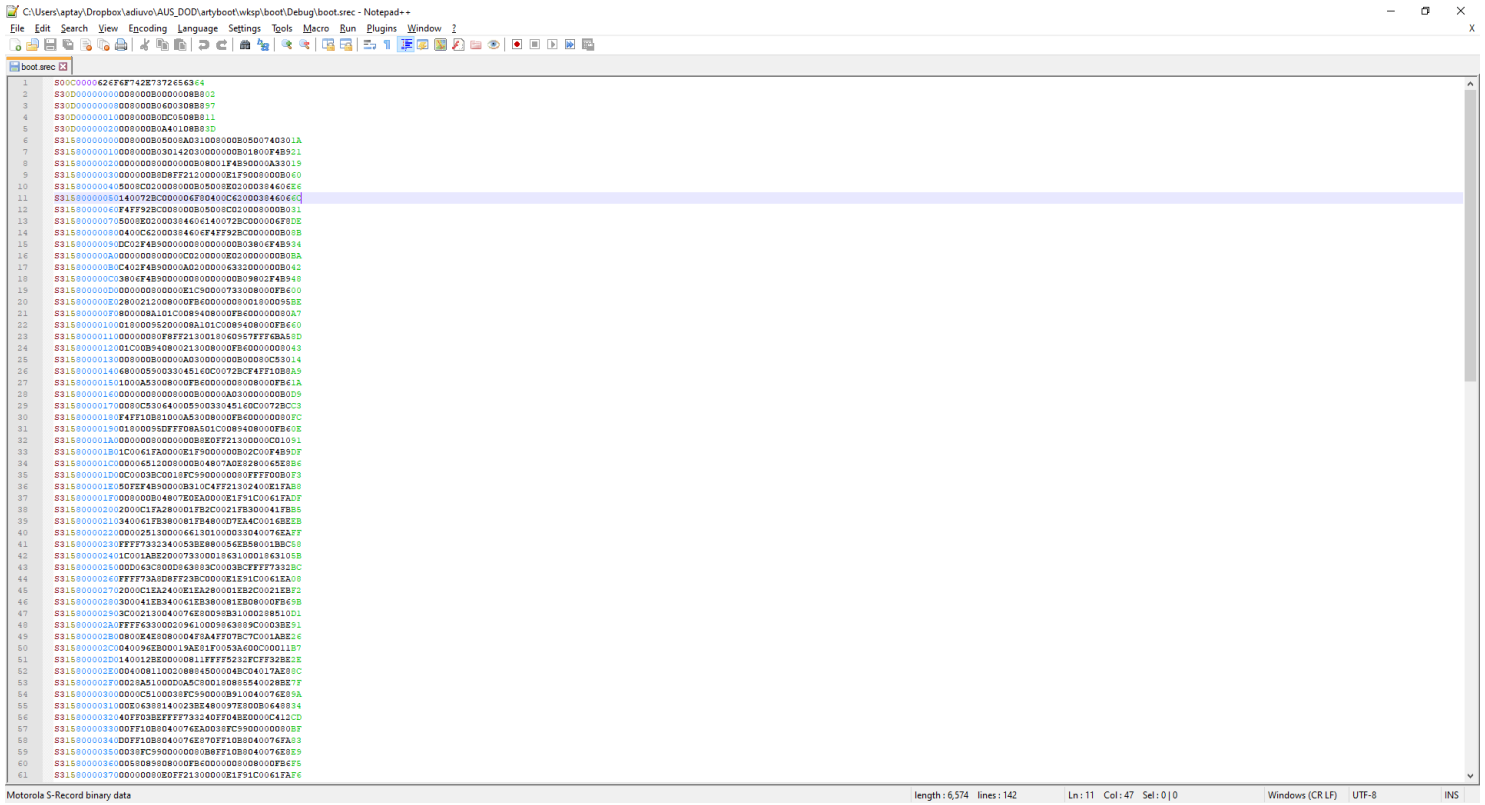


# MicroBlaze Boot From QSPI

The S Record file will be output to the same directory.

Open and observe in a text editor if desired

Do NOT make any changes



# Creating the Programming Files

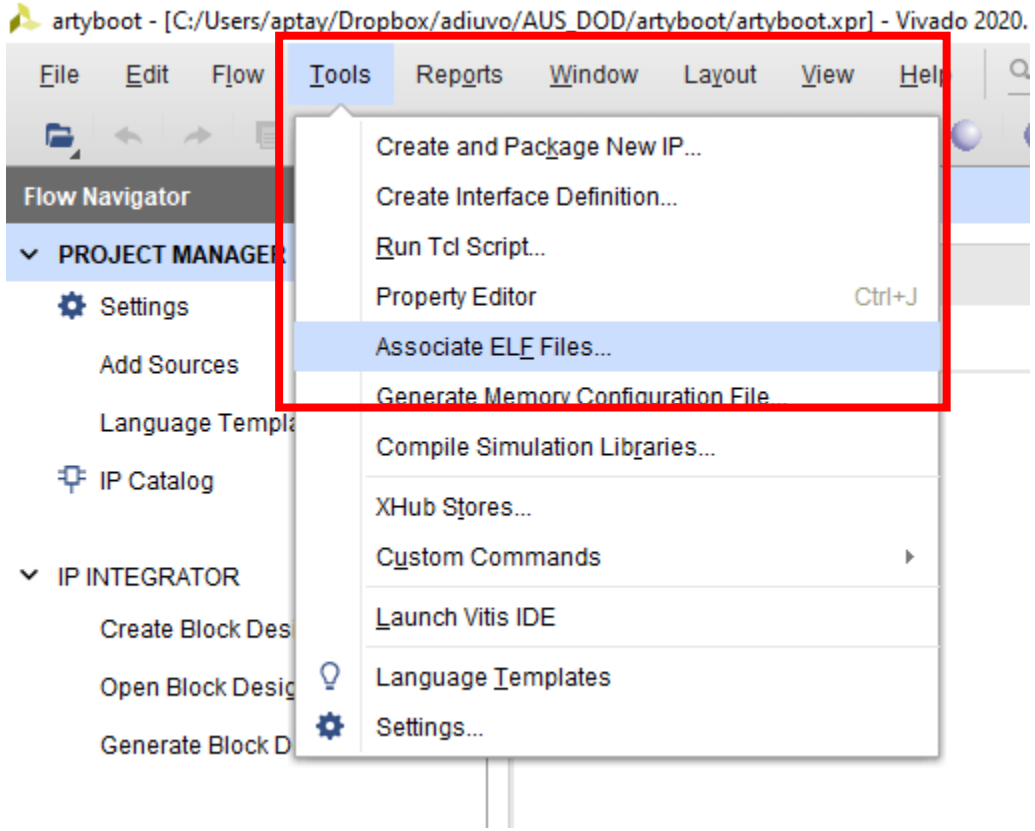
ADIUVO
ENGINEERING AND TRAINING, LTD.

# MicroBlaze Boot From QSPI

Back in Vivado we need to associate the bootloader elf with the bitstream.

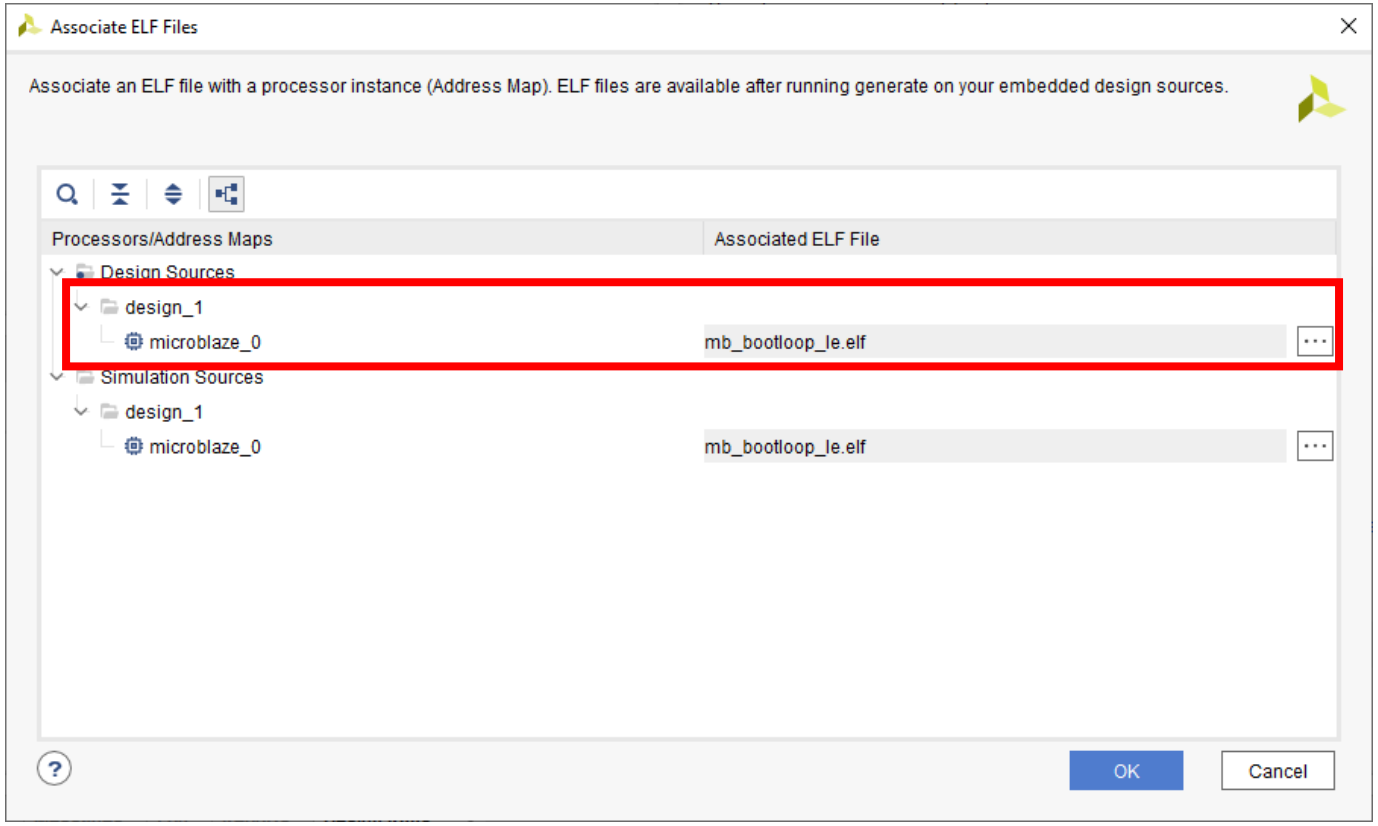
This will ensure the boot loader runs once the FPGA is configured

From the Tools menu select Associate ELF Files



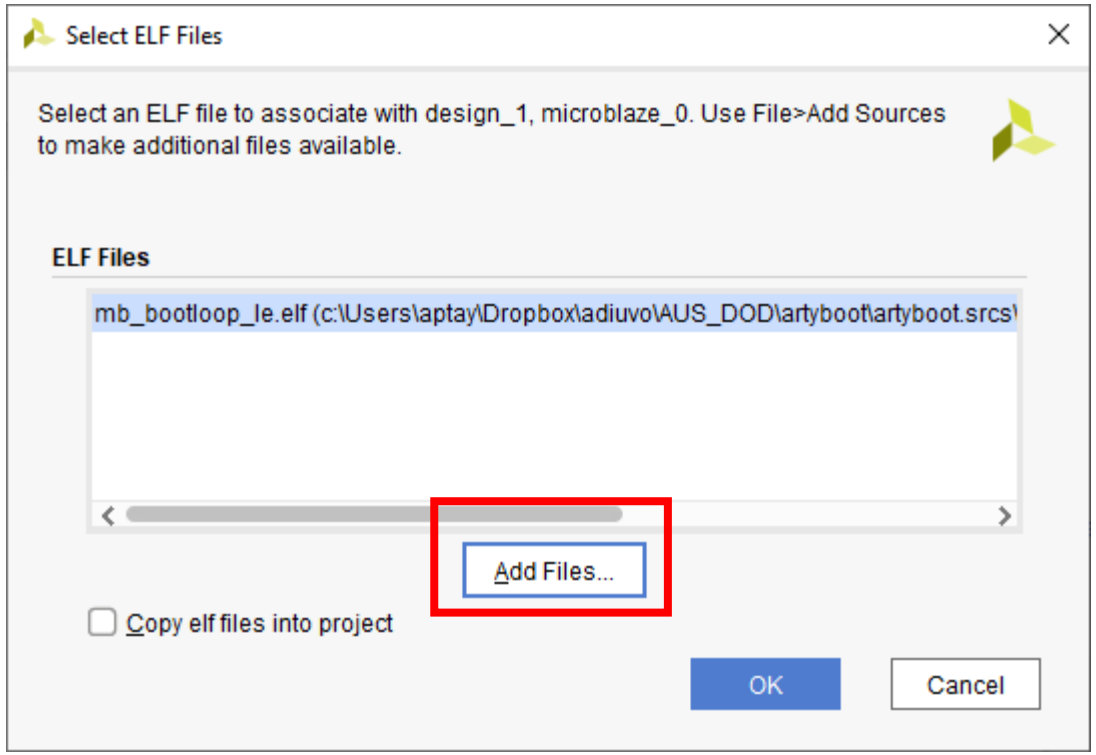
# MicroBlaze Boot From QSPI

In the dialog for design sources click on the … symbol to select a new file



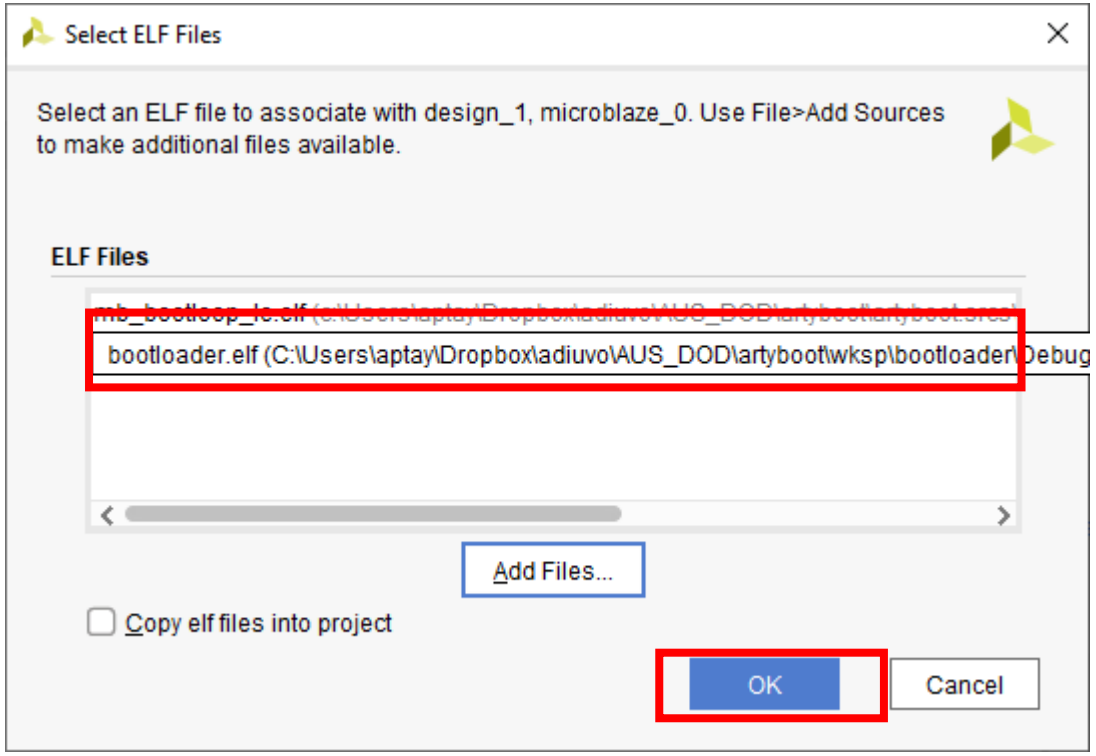
# MicroBlaze Boot From QSPI

Click on Add Files



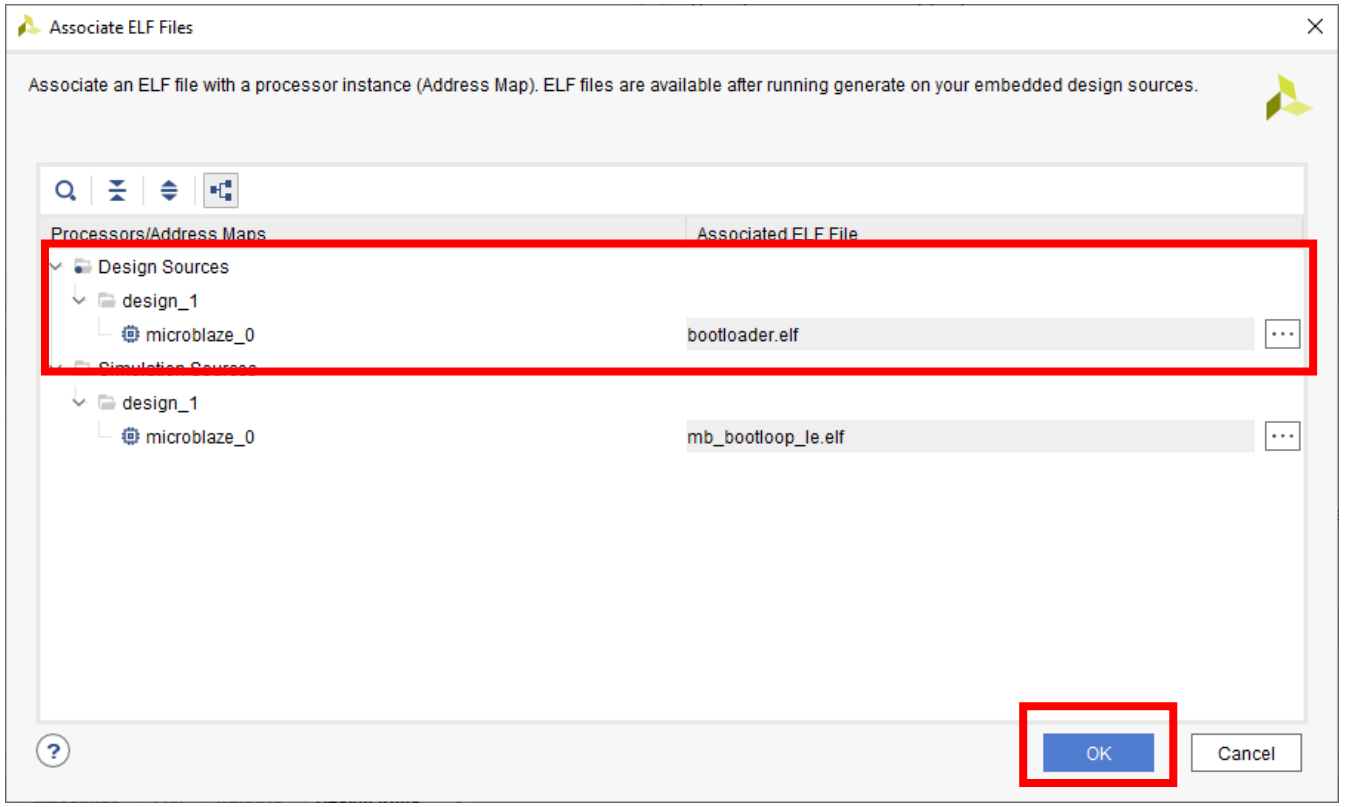
# MicroBlaze Boot From QSPI

Navigate and select the bootloader.elf from the workspace and select that file



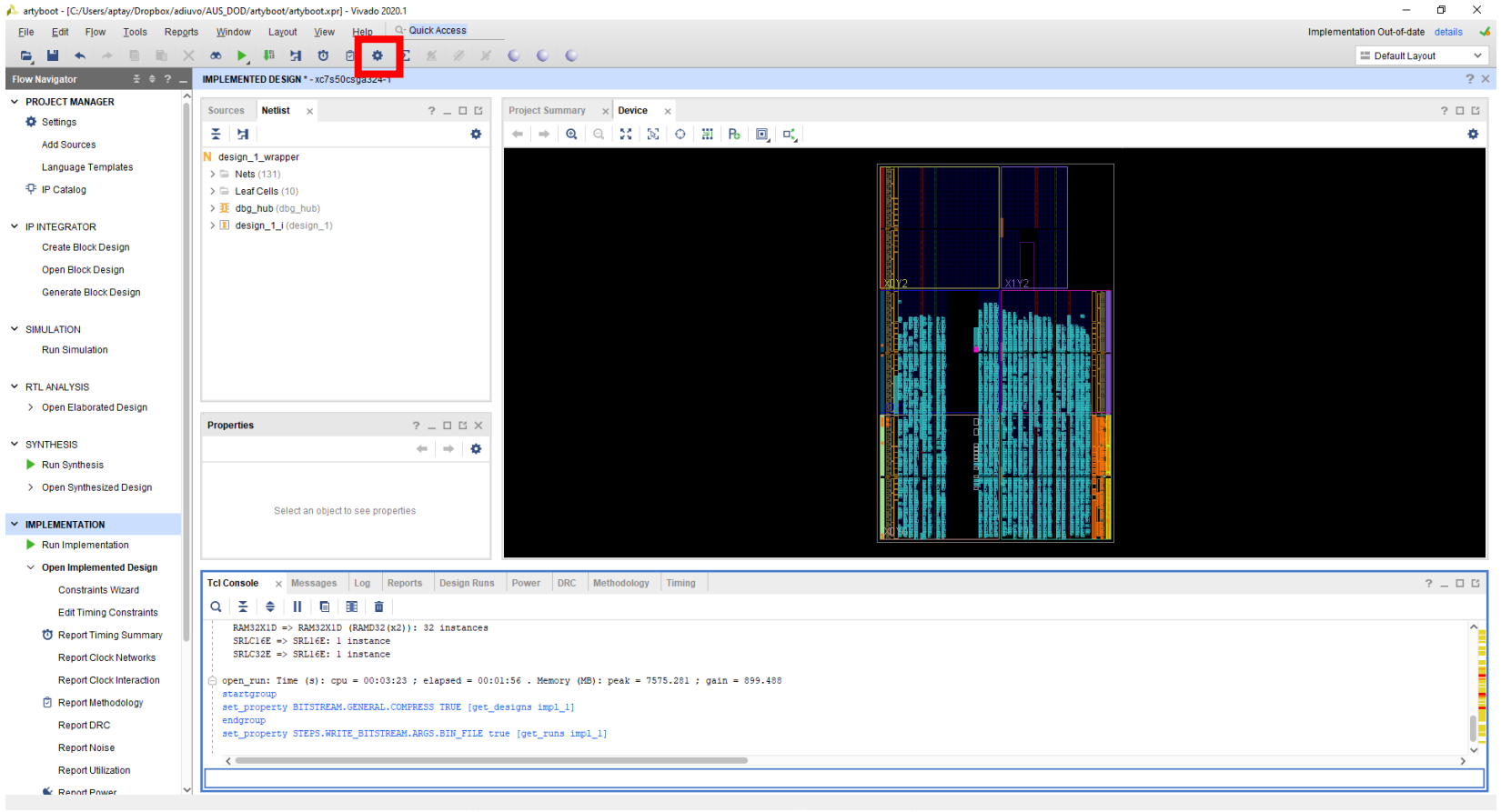
# MicroBlaze Boot From QSPI

This should show the bootloader.elf next to the design sources MicroBlaze. Click OK



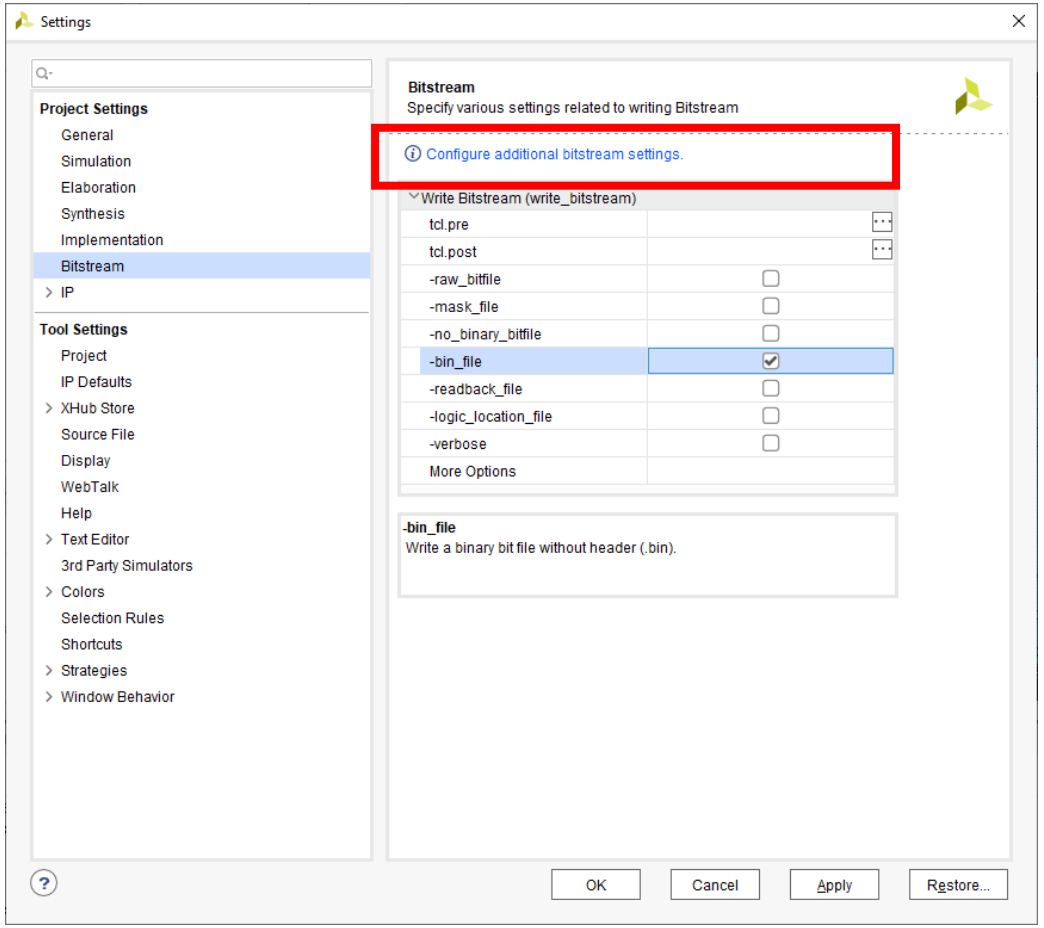
# MicroBlaze Boot From QSPI

Open the Implementation View – Ignore any warning about it being out of date and open the options



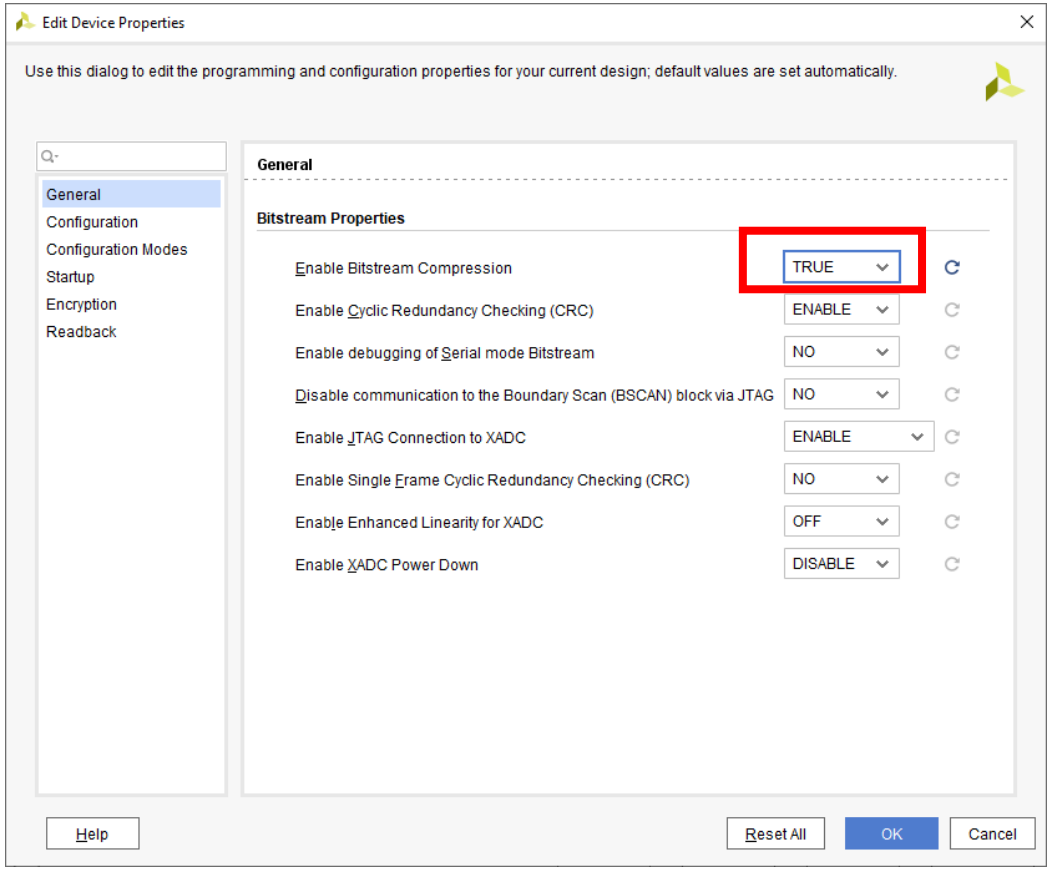
# MicroBlaze Boot From QSPI

Select the configure additional setting



# MicroBlaze Boot From QSPI

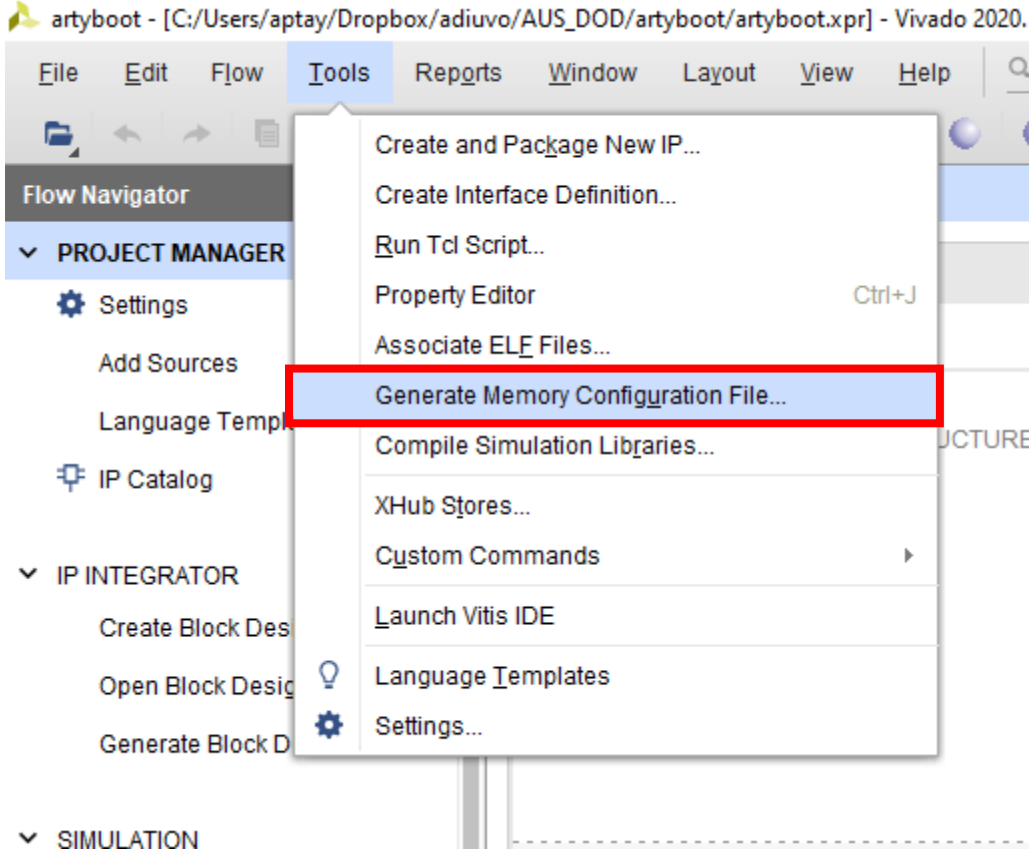
Select bitstream compression – Close the implementation view and re generate the bit stream



# MicroBlaze Boot From QSPI

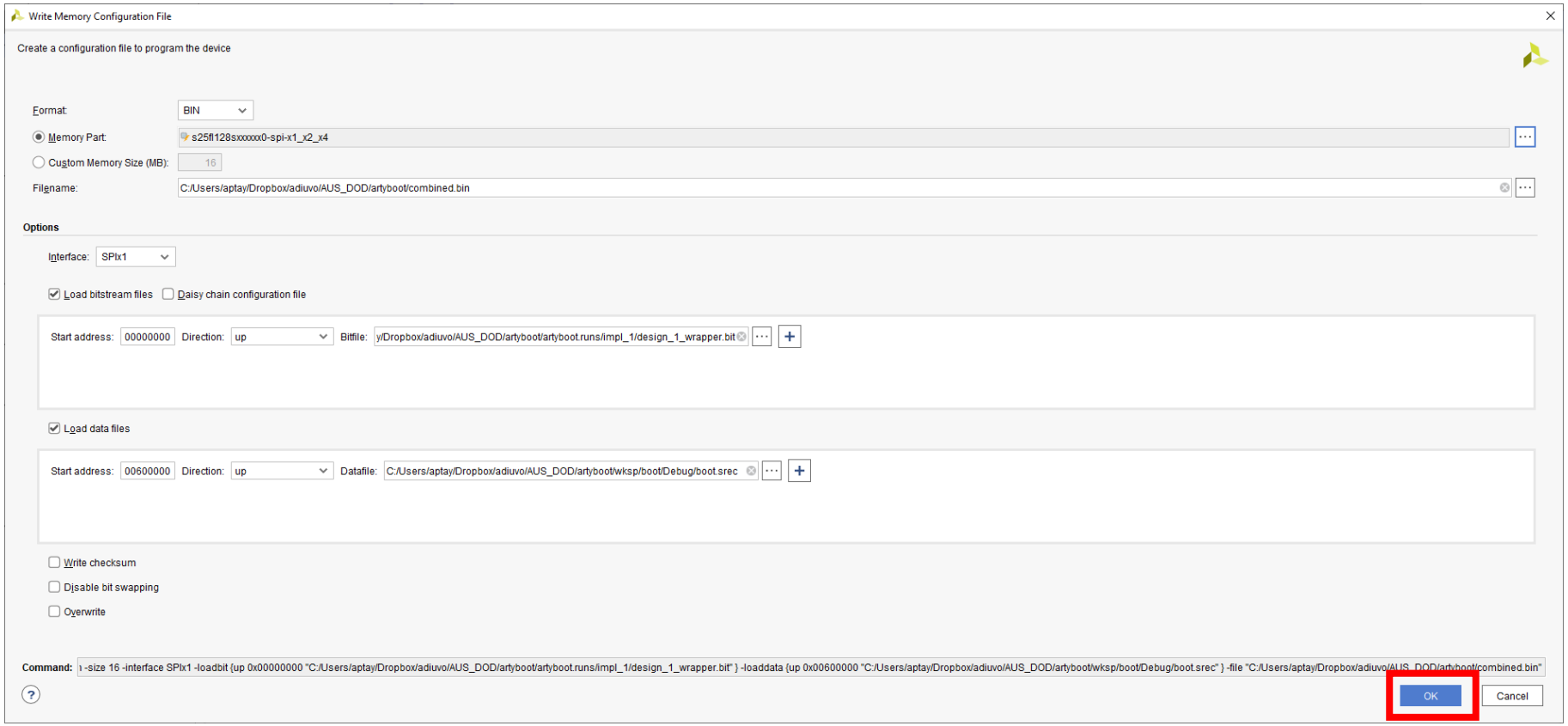
Now we need to create a bin file which contains the bitstream with the bootloader inbuilt and the SREC application

From the Tools menu select Generate Memory Configuration File



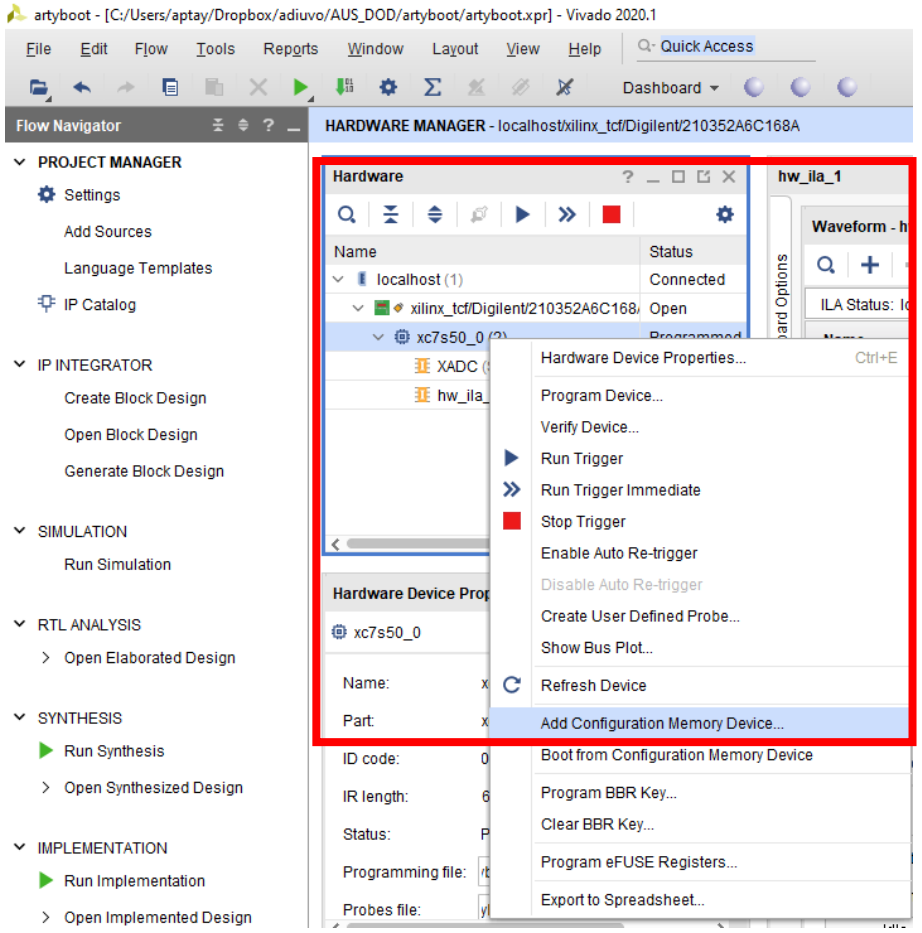
# MicroBlaze Boot From QSPI

Select the Format as BIN, the memory part (S25FL128sxxxxxx0-SPI-X1_X2_X4), the filename of the generated file. Select the bit file created by Vivado, and enable load data file, select the 0x600000 offset and the SREC file created. Then click OK



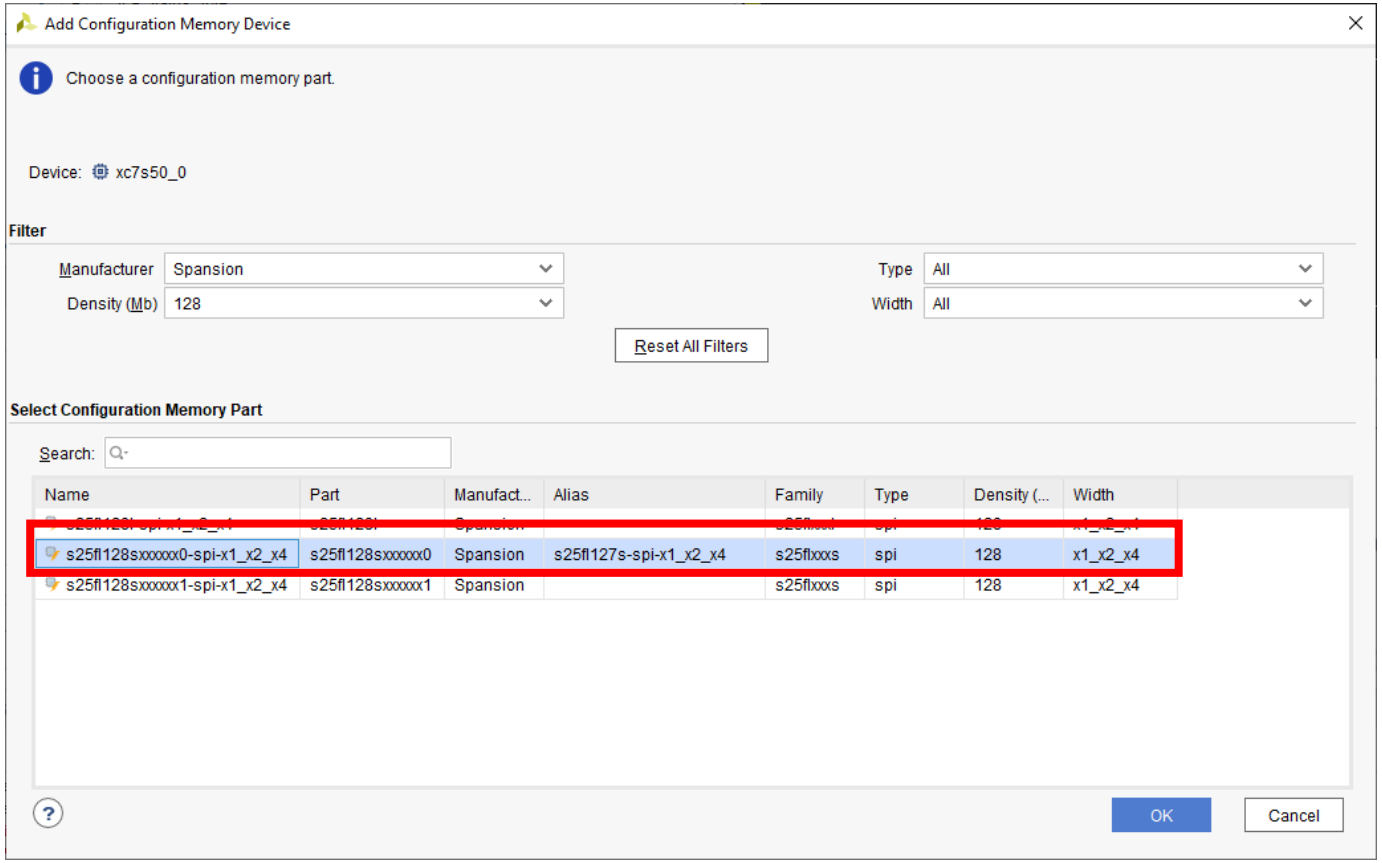
# MicroBlaze Boot From QSPI

Open the Hardware Manager and select the option to add configuration device.



# MicroBlaze Boot From QSPI

Select the Spansion S25FL128sxxxxxx0-SPI-X1_X2_X4

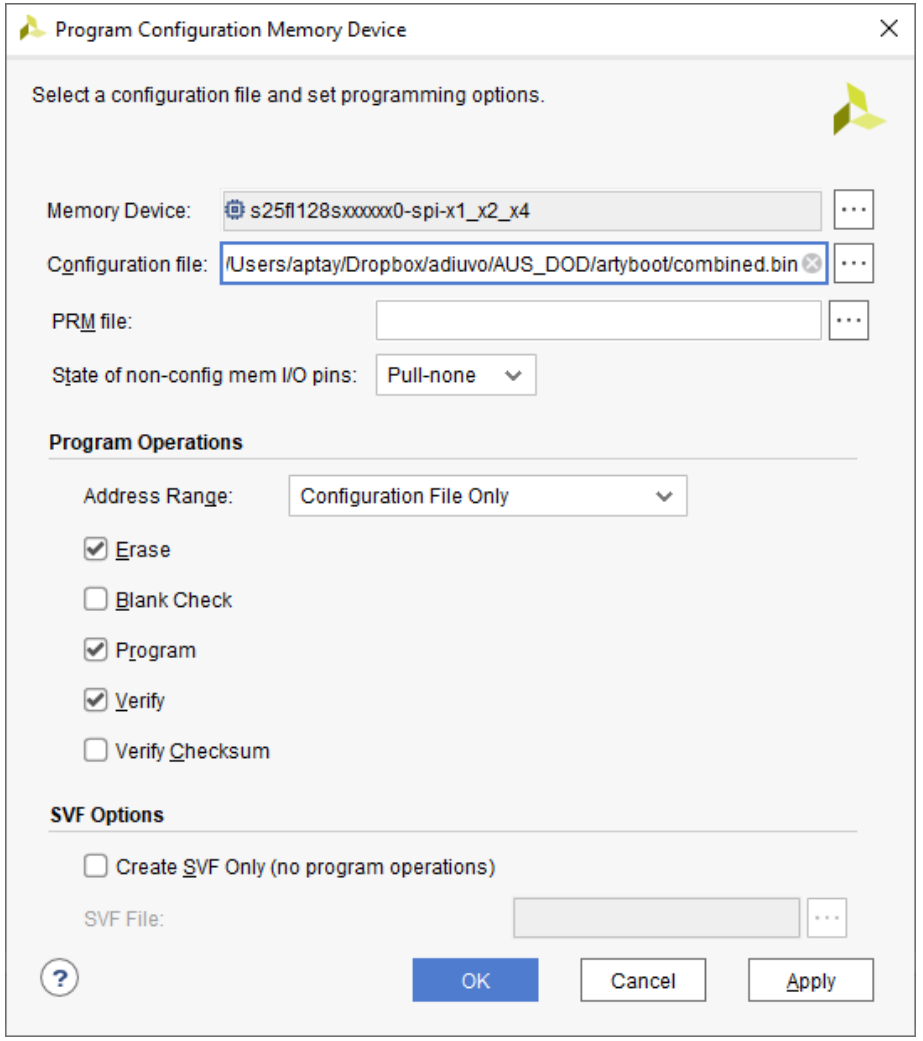


# MicroBlaze Boot From QSPI

Program the flash with the combined Bin file.

Once the programming is complete ensure the Arty S7-50 is configured to boot from the QSPI.

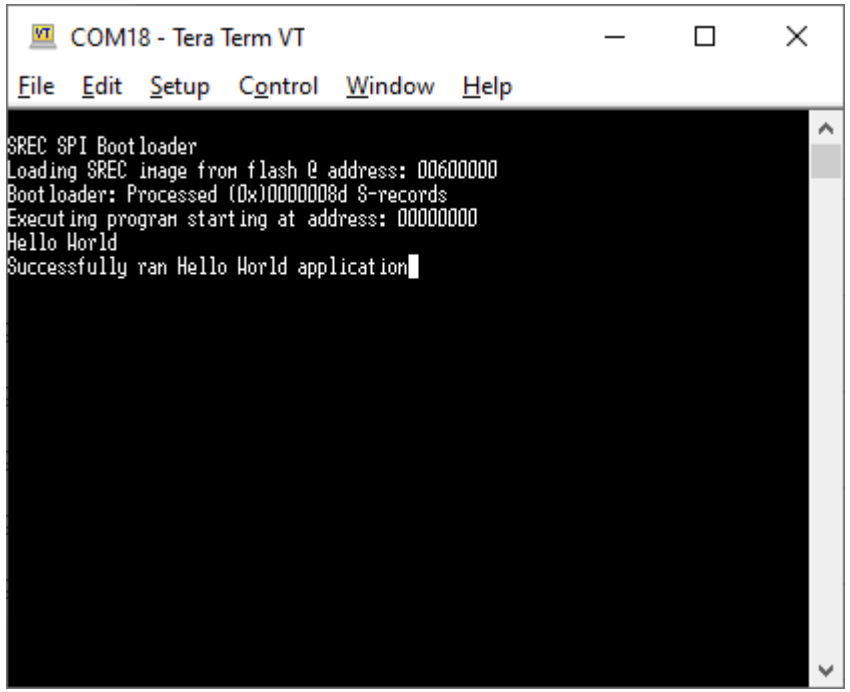
Connect a terminal 9600n1 and power cycle the board.



# MicroBlaze Boot From QSPI

The terminal should show the SREC bootloader running and cross loading into DDR.

Once in DDR the application should then run the hello world program



# ADIUVO

ENGINEERING AND TRAINING, LTD.

www.adiuvoengineering.com

adam@adiuvoengineering.com