



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.

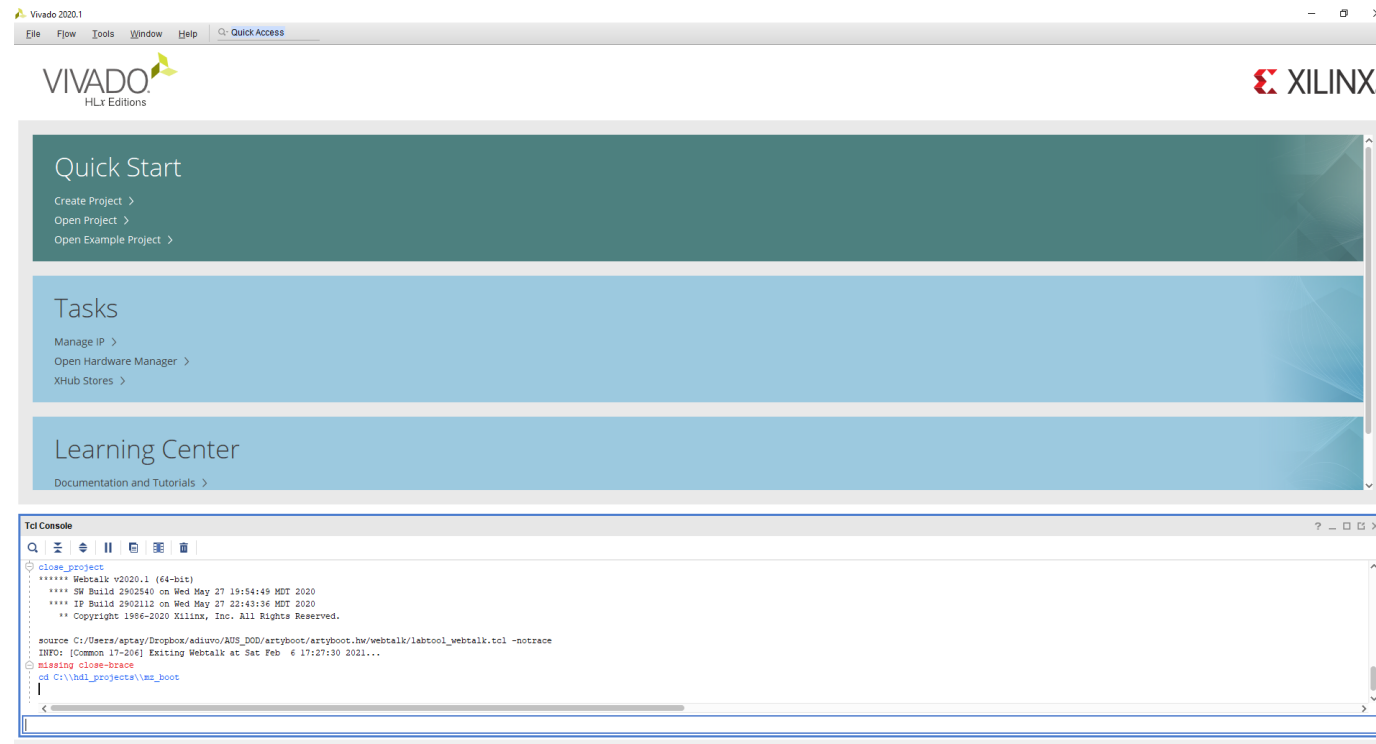
For this reason, the Vivado design must also contain a QSPI interface mapped to the same pins to enable the bootloader to read the application from the QSPI and relocate it to the DDR memory.



MicroBlaze Creation in Vivado from TCL Script

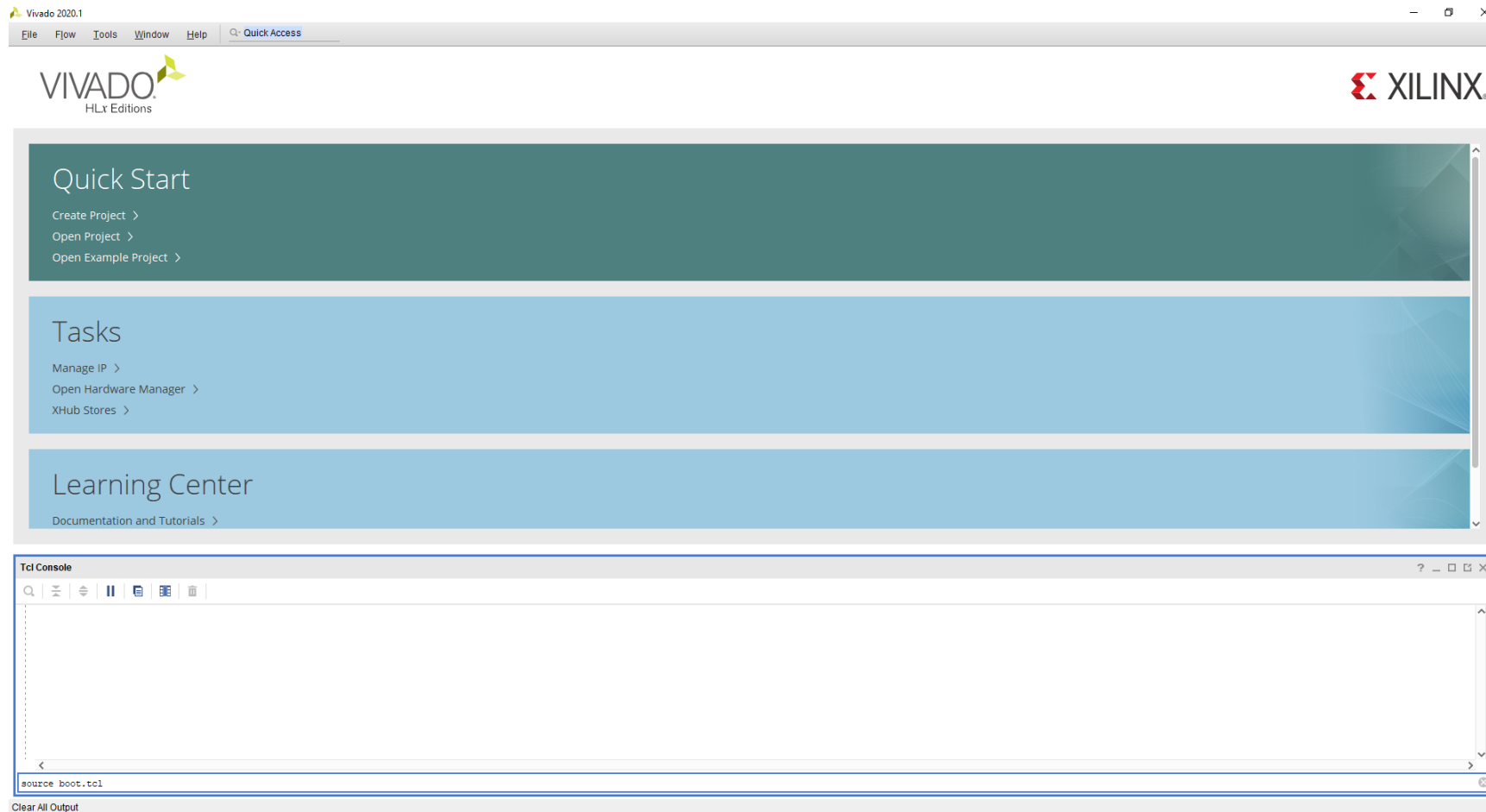
MicroBlaze Boot From QSPI

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

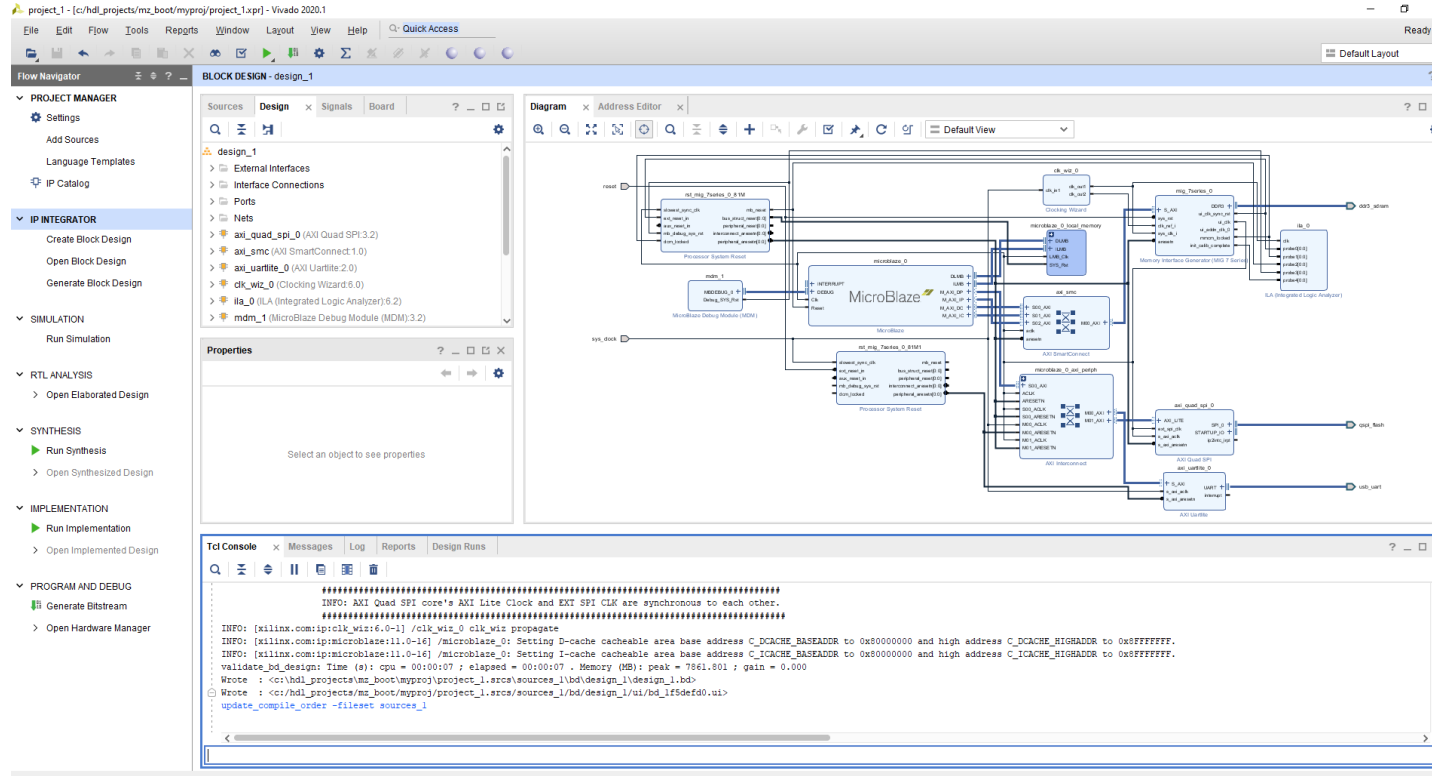
Run the command `source boot.tcl` – from the TCL console



Clear All Output

MicroBlaze Boot From QSPI

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



The screenshot displays the Vivado 2020.1 interface for a project named 'project_1'. The main window shows a block diagram of a MicroBlaze processor system. Key components include:

- MicroBlaze Processor:** The central component, configured with various parameters like 'MIPS_ARCH', 'MIPS_DSP', and 'MIPS_ICACHE'.
- AXI Quad SPI (0.81M):** An external interface connected to the processor's system bus.
- AXI SmartConnect:** A component that manages the system bus connections between the processor and other peripherals.
- AXI Quad SPI (0.81M):** Another instance of the SPI interface, connected to the system bus.
- AXI Quad SPI (0.81M):** A third instance of the SPI interface, connected to the system bus.
- AXI Quad SPI (0.81M):** A fourth instance of the SPI interface, connected to the system bus.
- AXI Quad SPI (0.81M):** A fifth instance of the SPI interface, connected to the system bus.
- AXI Quad SPI (0.81M):** A sixth instance of the SPI interface, connected to the system bus.
- AXI Quad SPI (0.81M):** A seventh instance of the SPI interface, connected to the system bus.
- AXI Quad SPI (0.81M):** An eighth instance of the SPI interface, connected to the system bus.
- AXI Quad SPI (0.81M):** A ninth instance of the SPI interface, connected to the system bus.
- AXI Quad SPI (0.81M):** A tenth instance of the SPI interface, connected to the system bus.

The console window at the bottom shows the following synthesis logs:

```

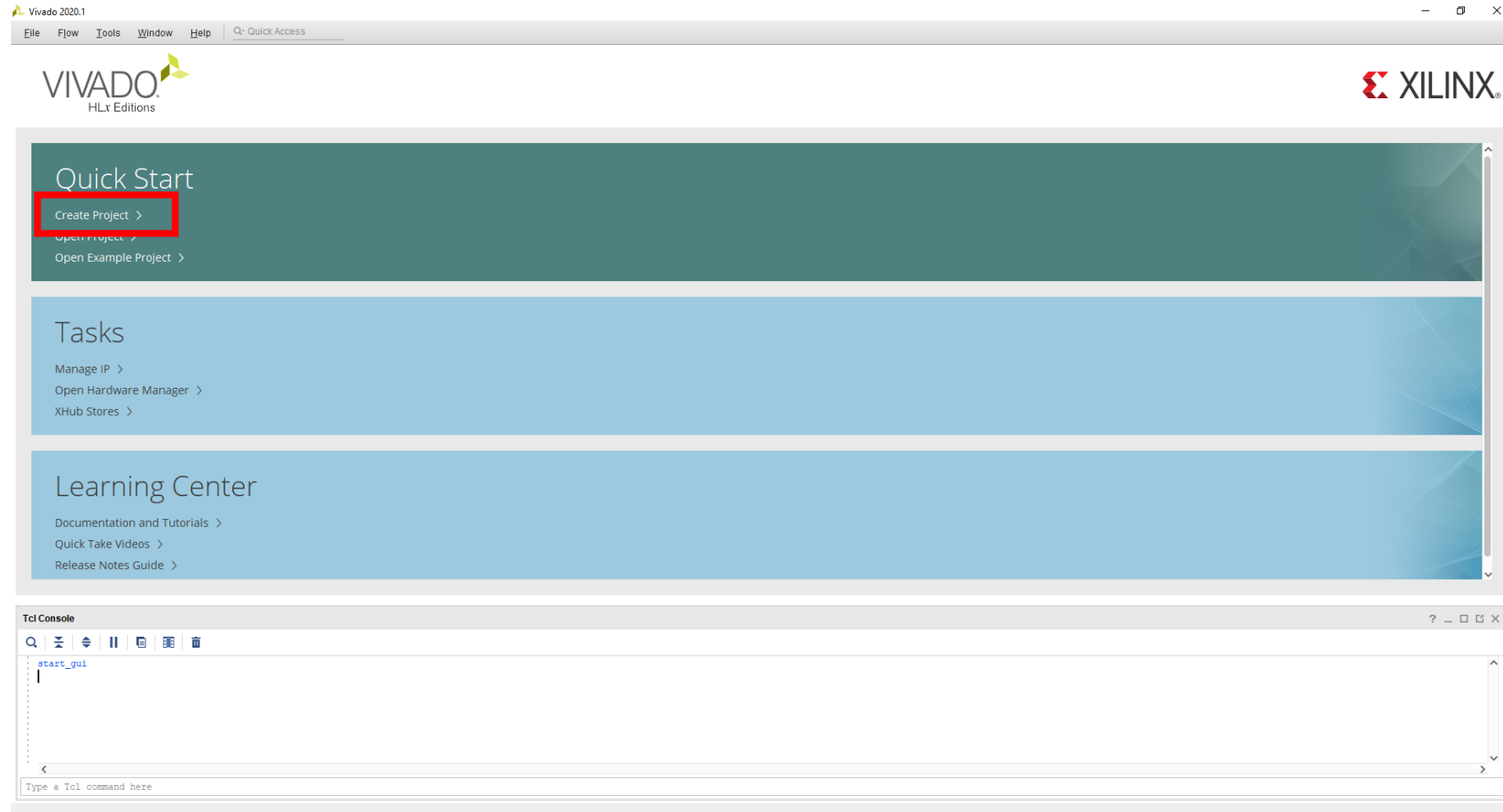
INFO: AXI Quad SPI core's AXI Lite Clock and EXT SPI CLK are synchronous to each other.
INFO: [xilix.com:ip:microblaze:11.0-16] /microblaze_0: Setting D-cache cacheable area base address C_DCACHE_BASEADDR to 0x90000000 and high address C_DCACHE_HIGHADDR to 0x8FFFFFFF.
INFO: [xilix.com:ip:microblaze:11.0-16] /microblaze_0: Setting I-cache cacheable area base address C_ICACHE_BASEADDR to 0x90000000 and high address C_ICACHE_HIGHADDR to 0x8FFFFFFF.
validate_bd_design: Time (s): cpu = 00:00:07 ; elapsed = 00:00:07 . Memory (MB): peak = 7861.801 ; gain = 0.000
Wrote : <ci:\hdl_projects\ms_boot\myproj\project_1.srcs\sources_1\bd\design_1\design_1.bd>
Wrote : <ci:\hdl_projects\ms_boot\myproj\project_1.srcs\sources_1\bd\design_1\ui\bd_defd0.u1>
update_compile_order -fileset sources_1
  
```




MicroBlaze Creation in Vivado from Scratch

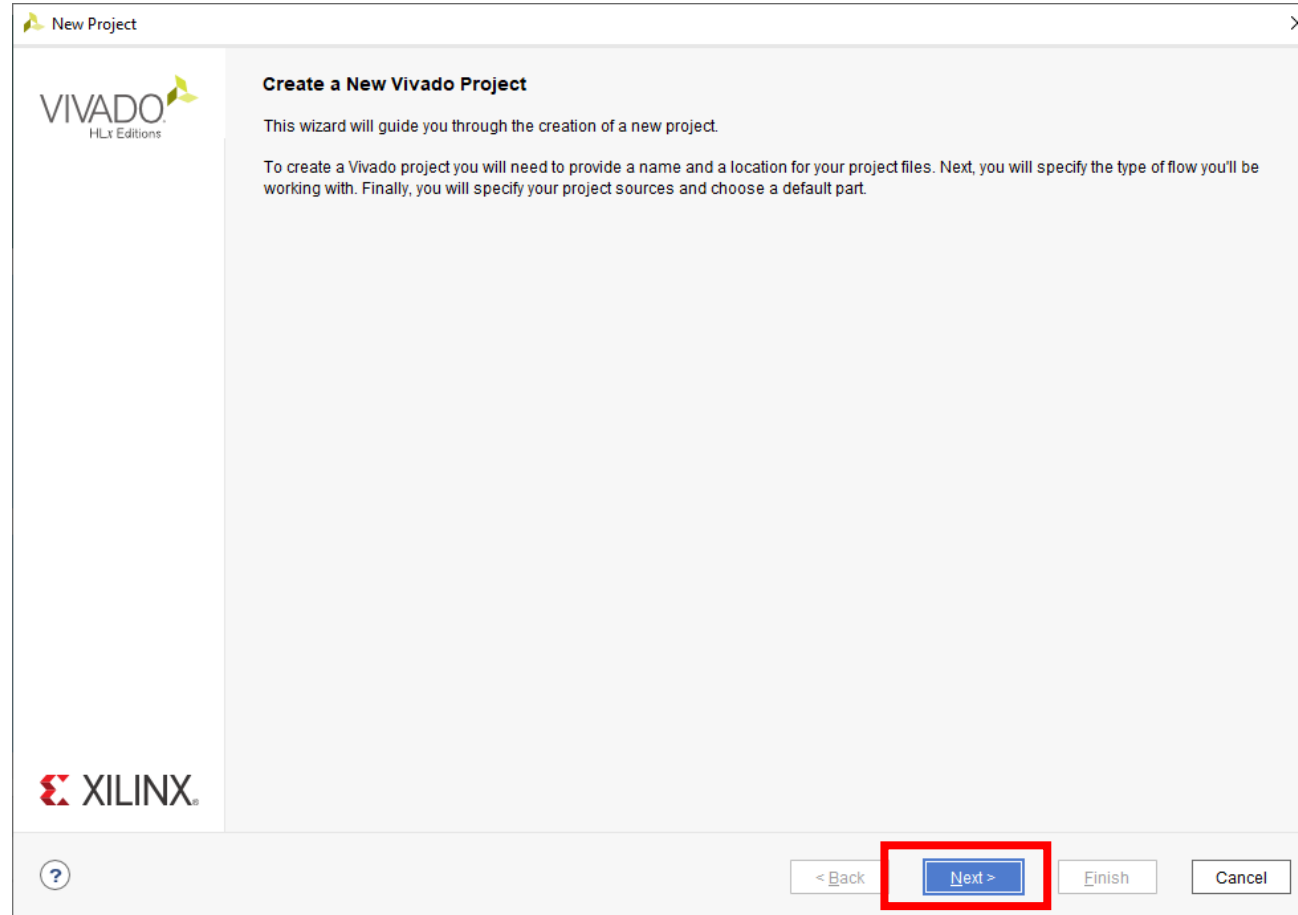
MicroBlaze Boot From QSPI

Open a new project in Vivado 2020.1



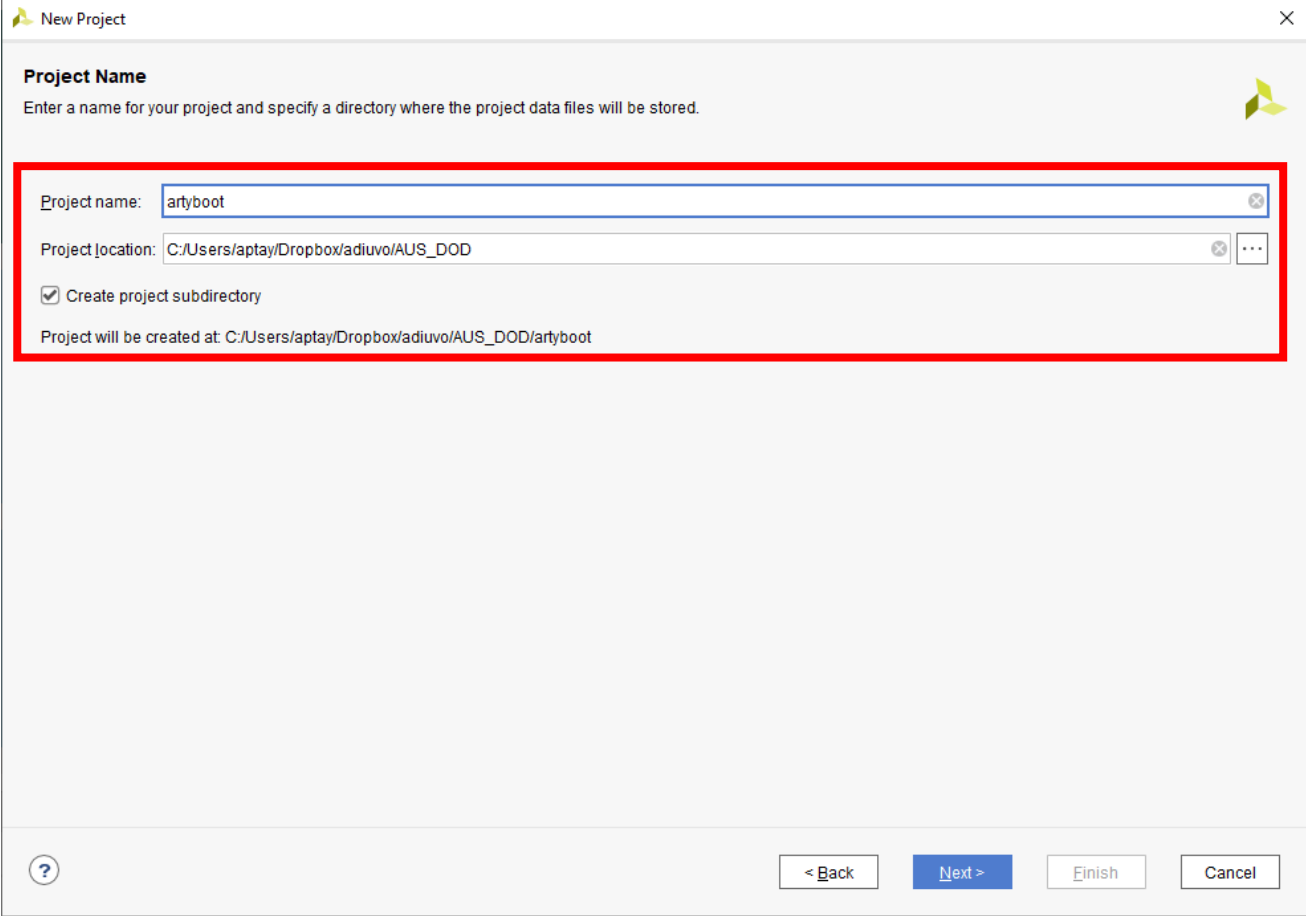
MicroBlaze Boot From QSPI

Click on next



MicroBlaze Boot From QSPI

Enter a name and project location, click next



New Project

Project Name
Enter a name for your project and specify a directory where the project data files will be stored.

Project name:

Project location:

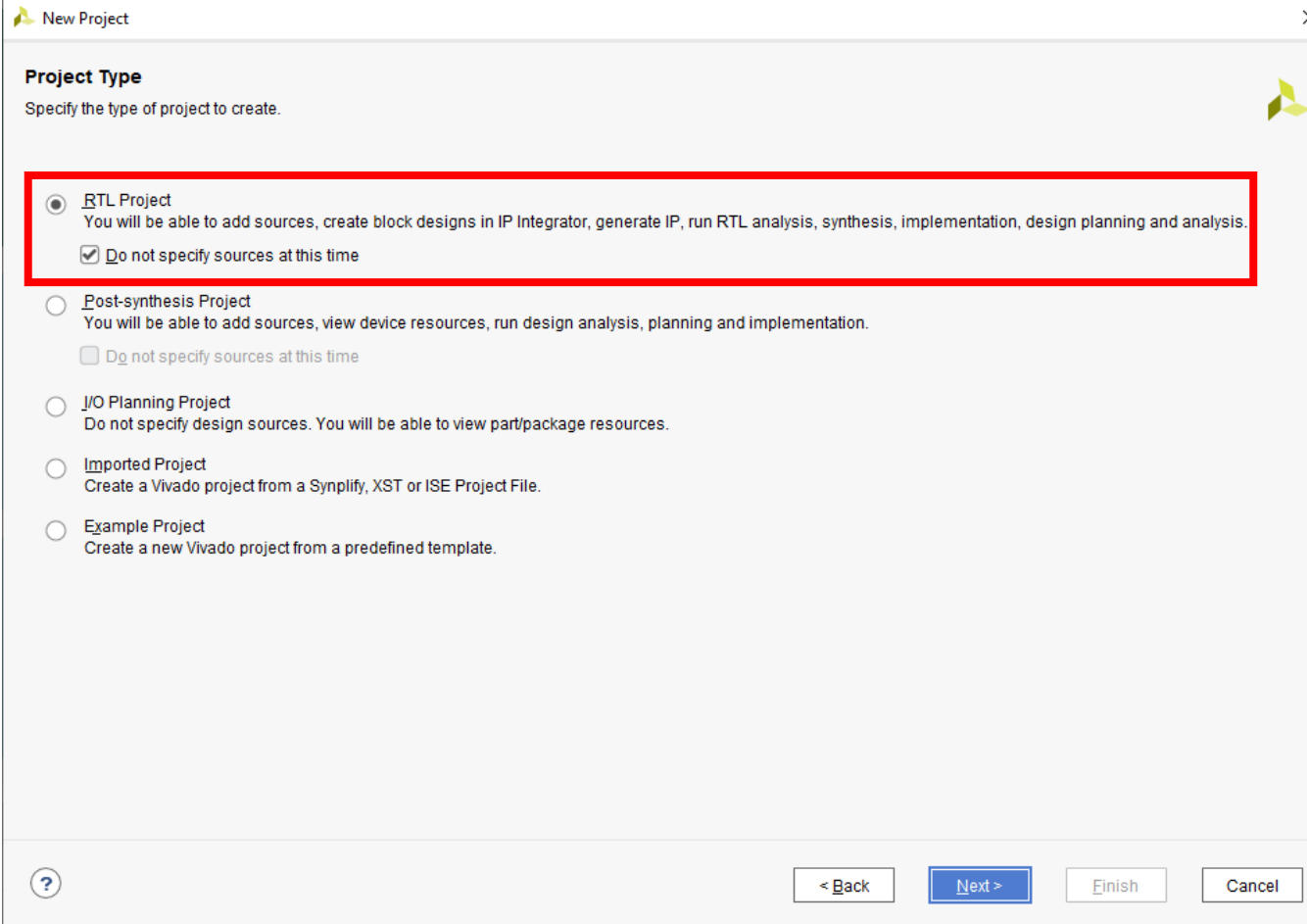
Create project subdirectory

Project will be created at: C:/Users/aptay/Dropbox/adiuvo/AUS_DOD/artyboot

? < Back Next > Finish Cancel

MicroBlaze Boot From QSPI

Select RTL Project and specify sources at later, click next



New Project

Project Type
Specify the type of project to create.

RTL Project
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
 Do not specify sources at this time

Post-synthesis Project
You will be able to add sources, view device resources, run design analysis, planning and implementation.
 Do not specify sources at this time

I/O Planning Project
Do not specify design sources. You will be able to view part/package resources.

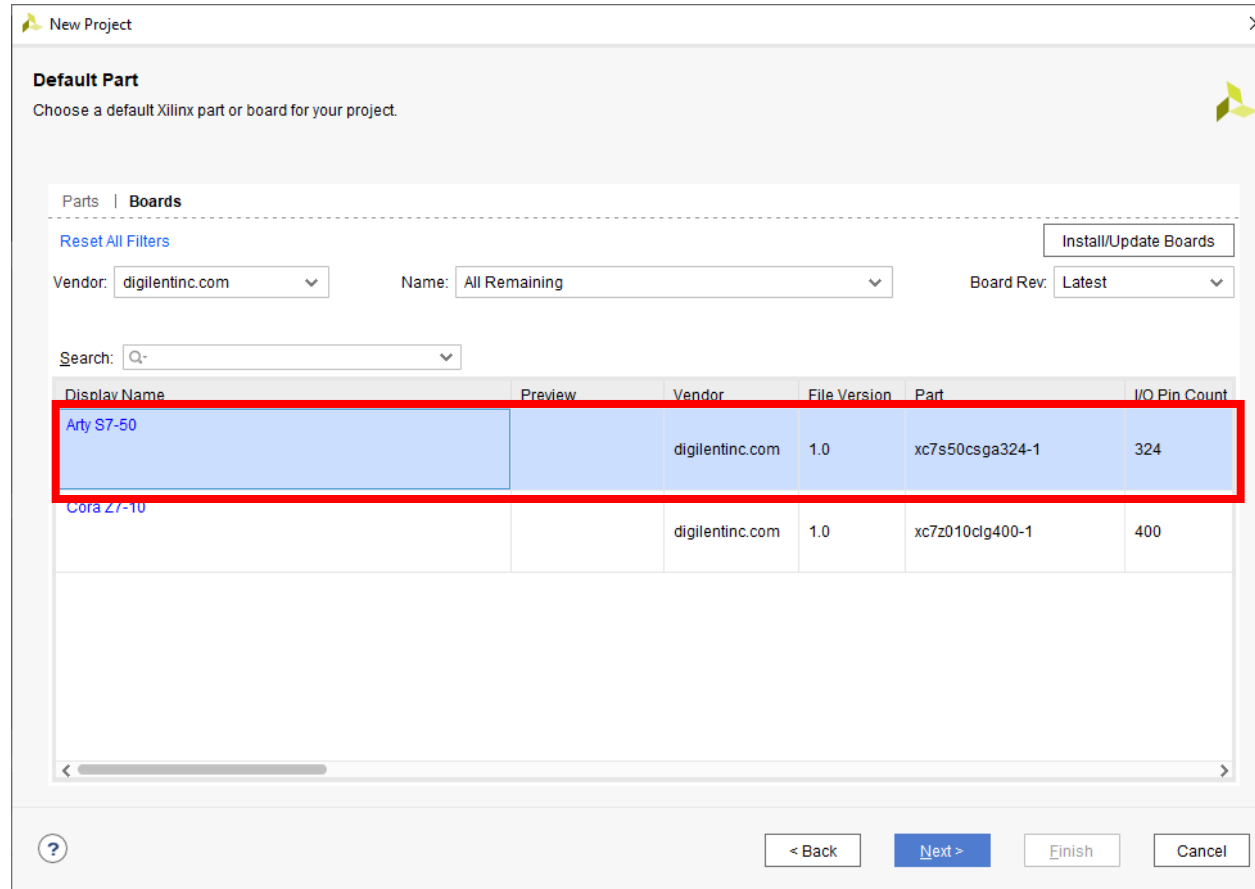
Imported Project
Create a Vivado project from a Synplify, XST or ISE Project File.

Example Project
Create a new Vivado project from a predefined template.

? < Back Next > Finish Cancel

MicroBlaze Boot From QSPI

Select Boards, and select the Arty S7-50



New Project

Default Part
Choose a default Xilinx part or board for your project.

Parts | **Boards**

[Reset All Filters](#) Install/Update Boards

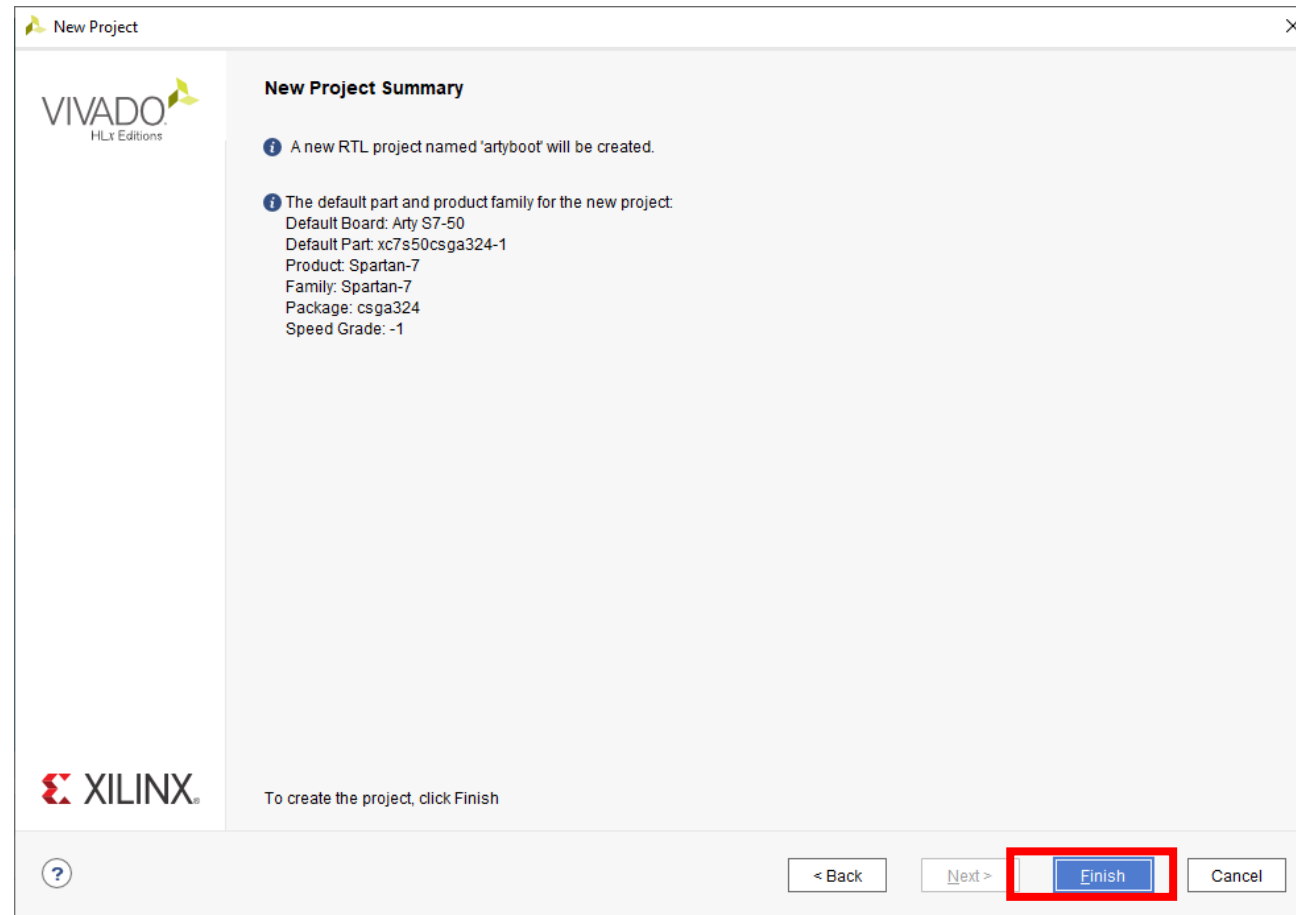
Vendor: Name: Board Rev:

Search:

Display Name	Preview	Vendor	File Version	Part	I/O Pin Count
Arty S7-50		digilentinc.com	1.0	xc7s50csga324-1	324
Cora Z7-10		digilentinc.com	1.0	xc7z010clg400-1	400

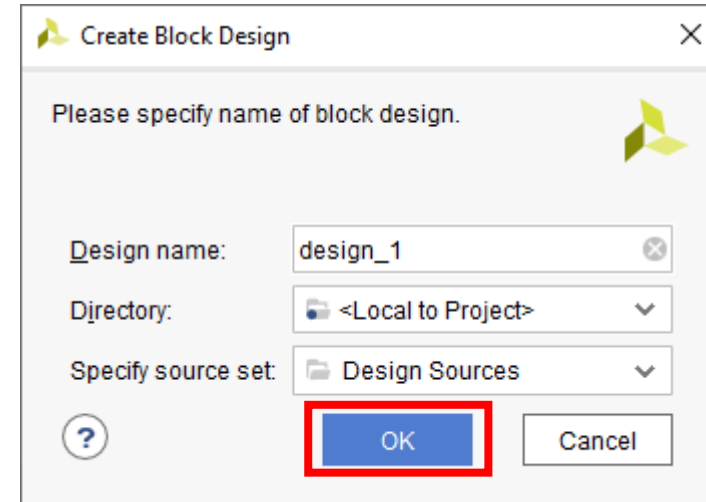
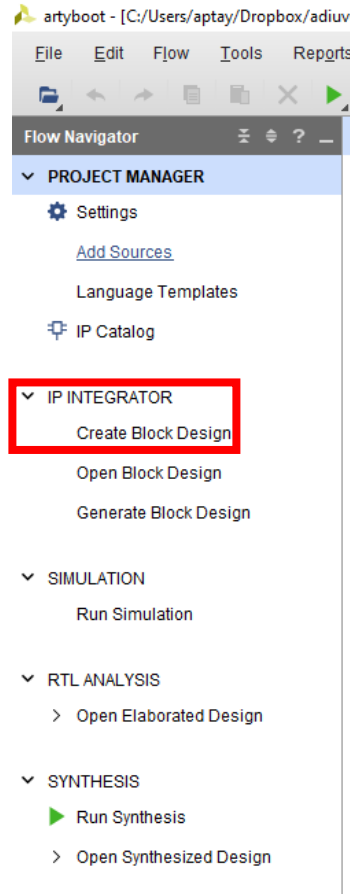
MicroBlaze Boot From QSPI

Finish the project creation



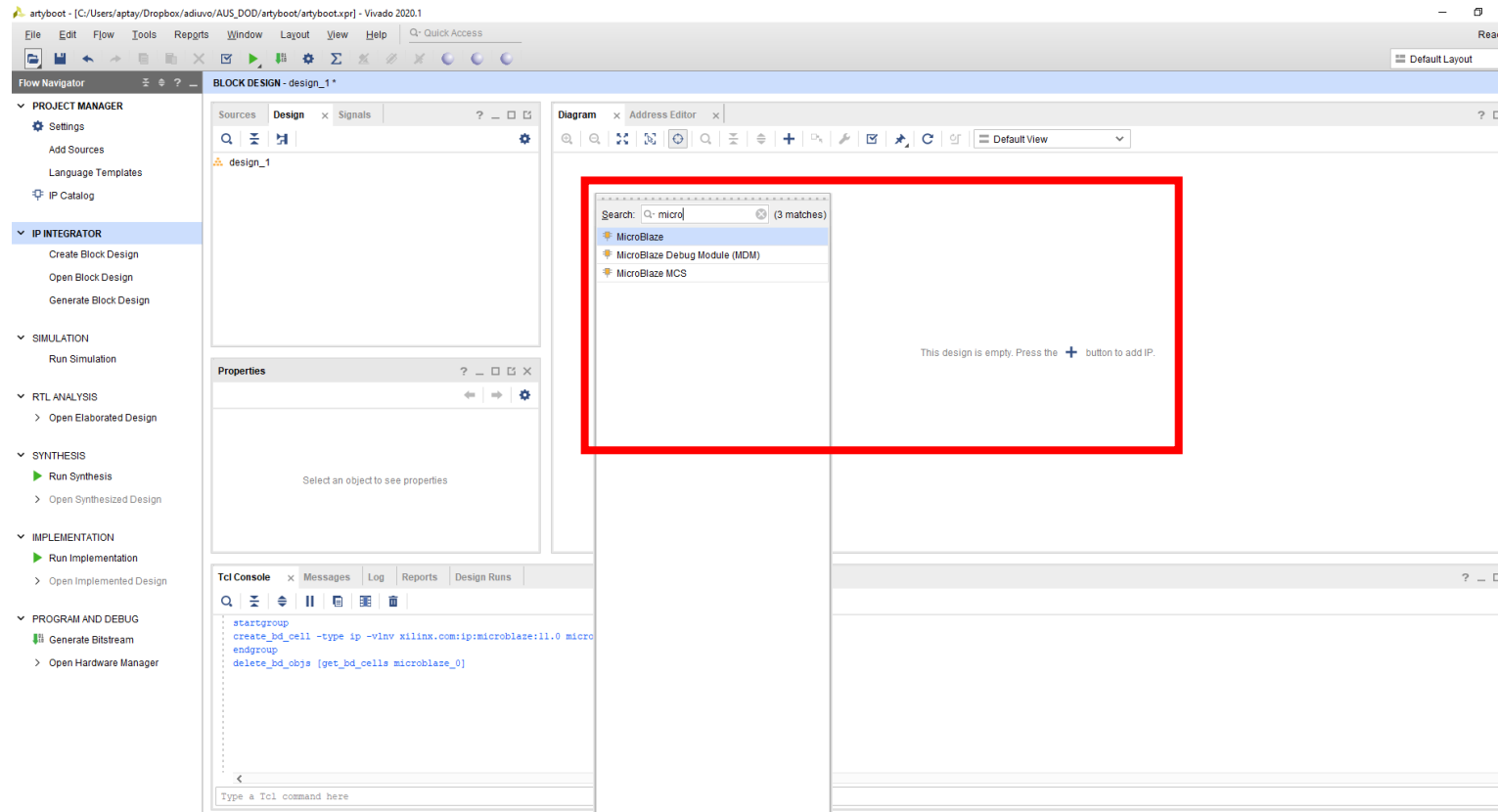
MicroBlaze Boot From QSPI

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 (11.0)

Documentation IP Location Advanced

IP Symbol Resources

Component Name: microblaze_0

Resource Estimates

Category	Value
Frequency	~80%
Area	~20%
Performance	~25%

Resource Usage

BRAM:	DSP48E:
18	0

Usage Information

- Select a predefined configuration with *Select Configuration* below. Information about the selected configuration can be found in the tooltip. *Each predefined configuration completely changes the MicroBlaze parameters.*
- To modify the configuration, click on the *Next* button, click on the *Advanced* button at the top to directly access parameters in a tabbed interface, or click *OK* to accept the configuration and close the dialog.

Predefined Configurations

Select Configuration: Current Settings

Select Processor Implementation

32 64

General Settings

Select implementation optimization: PERFORMANCE

Enable MicroBlaze Debug Module Interface

Use Instruction and Data Caches

Enable Exceptions

Use Memory Management

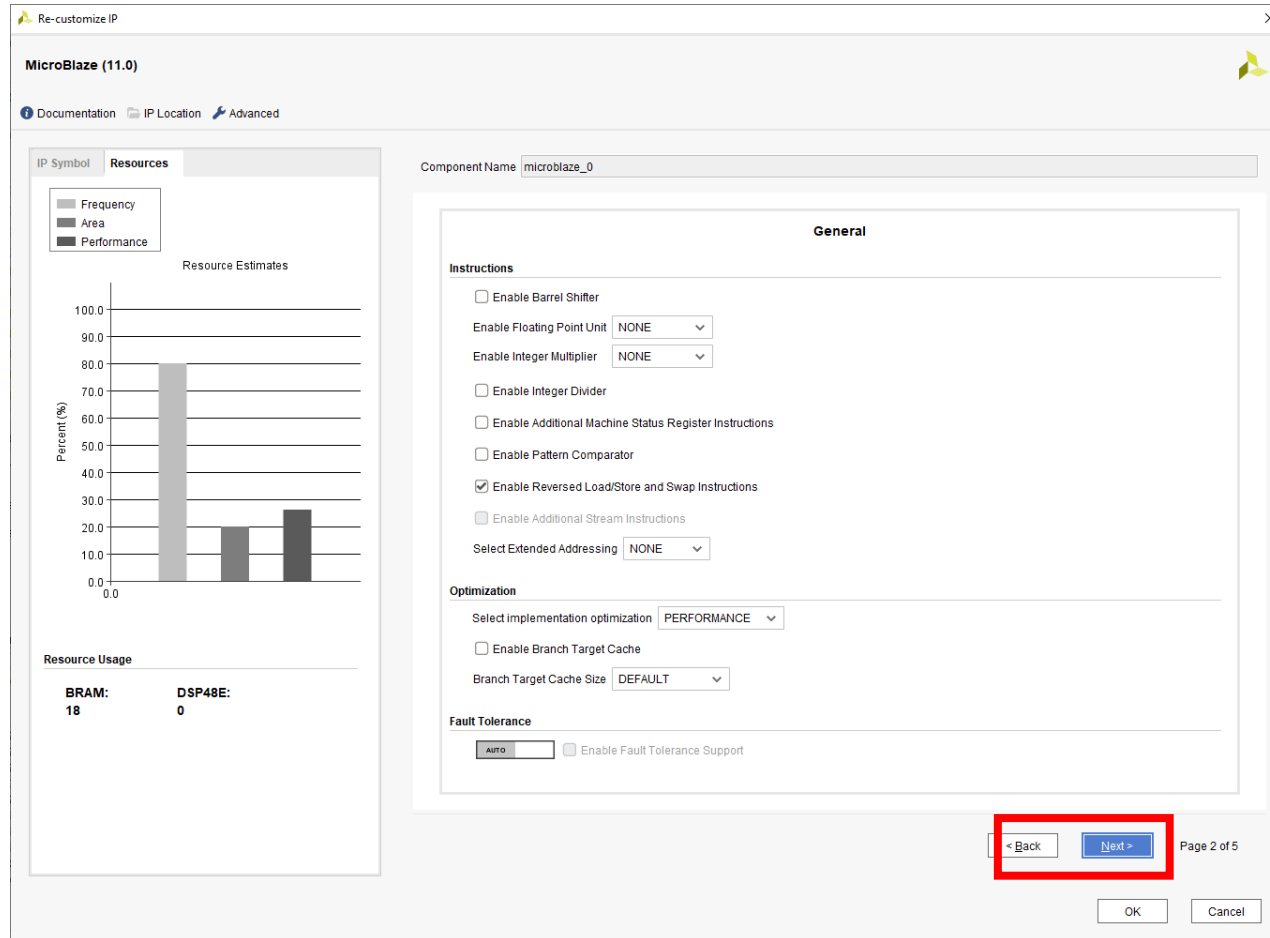
Enable Discrete Ports

< Back Next > Page 1 of 5

OK Cancel

MicroBlaze Boot From QSPI

On the second page leave unchanged and click next



MicroBlaze (11.0)

Documentation IP Location Advanced

IP Symbol **Resources** Component Name: `microblaze_0`

Resource Estimates

Category	Frequency (%)	Area (%)	Performance (%)
Frequency	~80	~20	~28

Resource Usage

BRAM:	DSP48E:
18	0

General

Instructions

- Enable Barrel Shifter
- Enable Floating Point Unit: NONE
- Enable Integer Multiplier: NONE
- Enable Integer Divider
- Enable Additional Machine Status Register Instructions
- Enable Pattern Comparator
- Enable Reversed Load/Store and Swap Instructions
- Enable Additional Stream Instructions
- Select Extended Addressing: NONE

Optimization

- Select implementation optimization: PERFORMANCE
- Enable Branch Target Cache
- Branch Target Cache Size: DEFAULT

Fault Tolerance

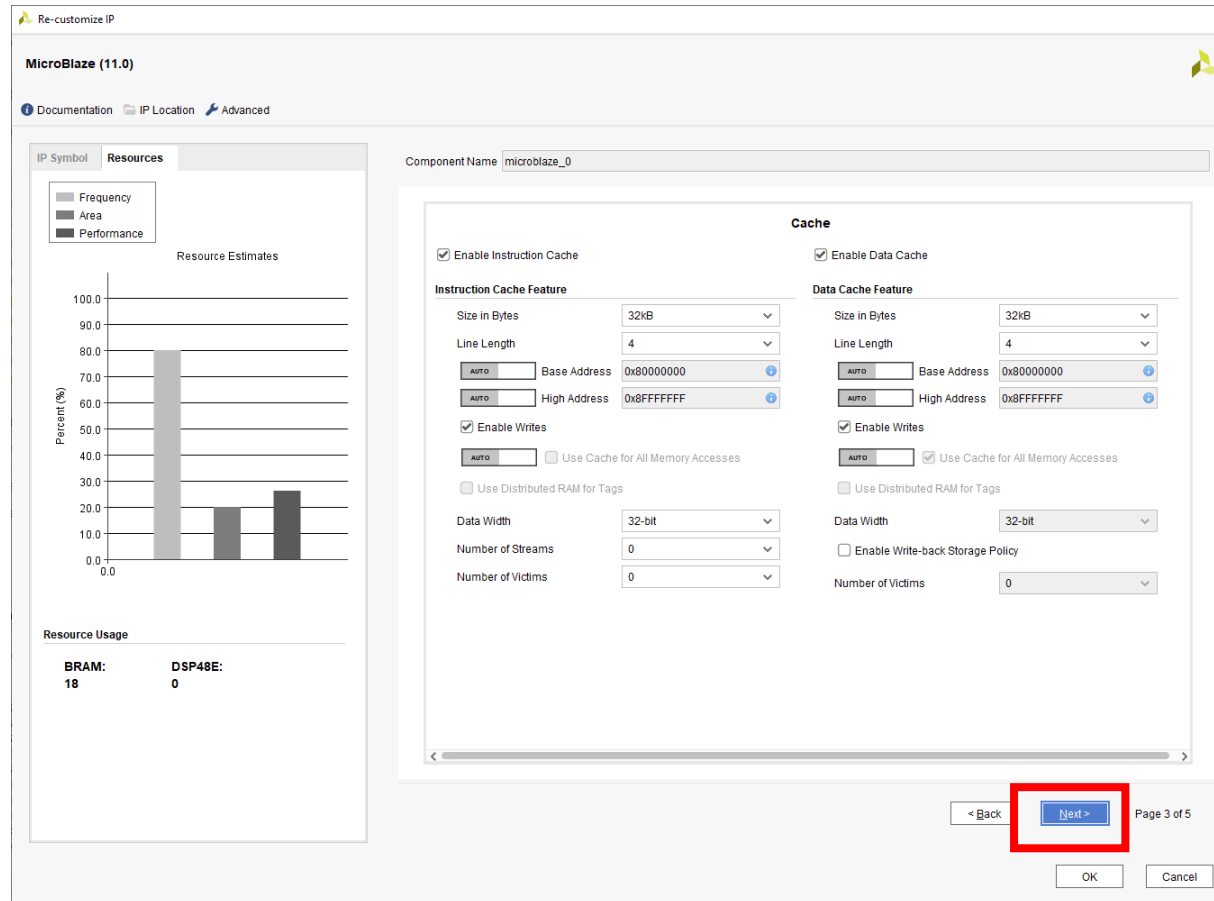
- Auto Enable Fault Tolerance Support

< Back Next > Page 2 of 5

OK Cancel

MicroBlaze Boot From QSPI

Leave the third page unchanged and click next



The screenshot shows the 'Re-customize IP' window for MicroBlaze (11.0). The 'Cache' configuration page is active, showing settings for the Instruction Cache and Data Cache. The 'Next >' button is highlighted with a red box.

Cache Configuration:

- Enable Instruction Cache
- Enable Data Cache
- Instruction Cache Feature:**
 - Size in Bytes: 32kB
 - Line Length: 4
 - Base Address: 0x80000000
 - High Address: 0x8FFFFFFF
 - Enable Writes
 - Use Cache for All Memory Accesses
 - Use Distributed RAM for Tags
 - Data Width: 32-bit
 - Number of Streams: 0
 - Number of Victims: 0
- Data Cache Feature:**
 - Size in Bytes: 32kB
 - Line Length: 4
 - Base Address: 0x80000000
 - High Address: 0x8FFFFFFF
 - Enable Writes
 - Use Cache for All Memory Accesses
 - Use Distributed RAM for Tags
 - Data Width: 32-bit
 - Enable Write-back Storage Policy
 - Number of Victims: 0

Resource Estimates:

Category	Value (%)
Frequency	~80
Area	~20
Performance	~25

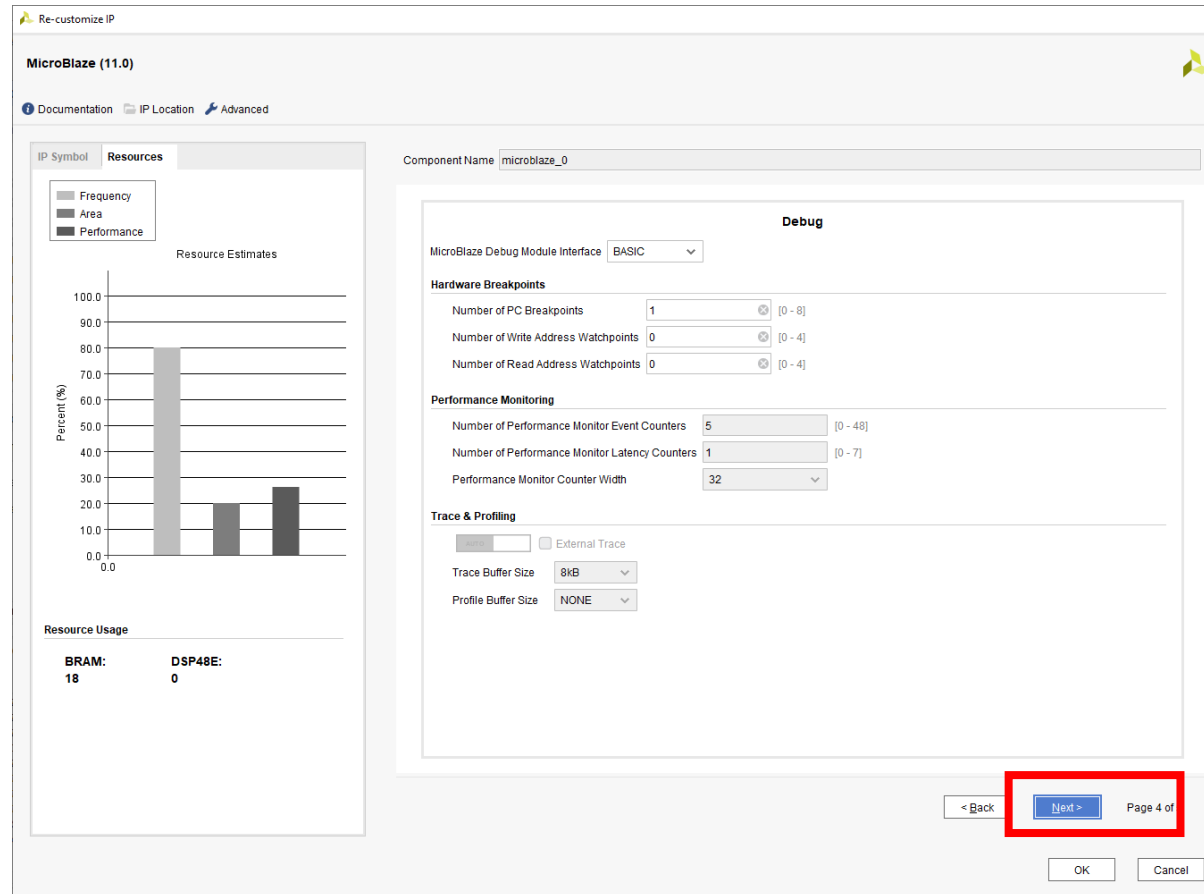
Resource Usage:

Resource	Usage
BRAM:	18
DSP48E:	0

Navigation: < Back, **Next >**, Page 3 of 5, OK, Cancel

MicroBlaze Boot From QSPI

Leave the fourth page unchanged and click next



Re-customize IP

MicroBlaze (11.0)

Documentation IP Location Advanced

IP Symbol Resources

Frequency Area Performance

Resource Estimates

Percent (%)

Resource Usage

BRAM: 18 DSP48E: 0

Component Name: microblaze_0

Debug

MicroBlaze Debug Module Interface: BASIC

Hardware Breakpoints

Number of PC Breakpoints: 1 [0 - 8]

Number of Write Address Watchpoints: 0 [0 - 4]

Number of Read Address Watchpoints: 0 [0 - 4]

Performance Monitoring

Number of Performance Monitor Event Counters: 5 [0 - 48]

Number of Performance Monitor Latency Counters: 1 [0 - 7]

Performance Monitor Counter Width: 32

Trace & Profiling

External Trace

Trace Buffer Size: 8kB

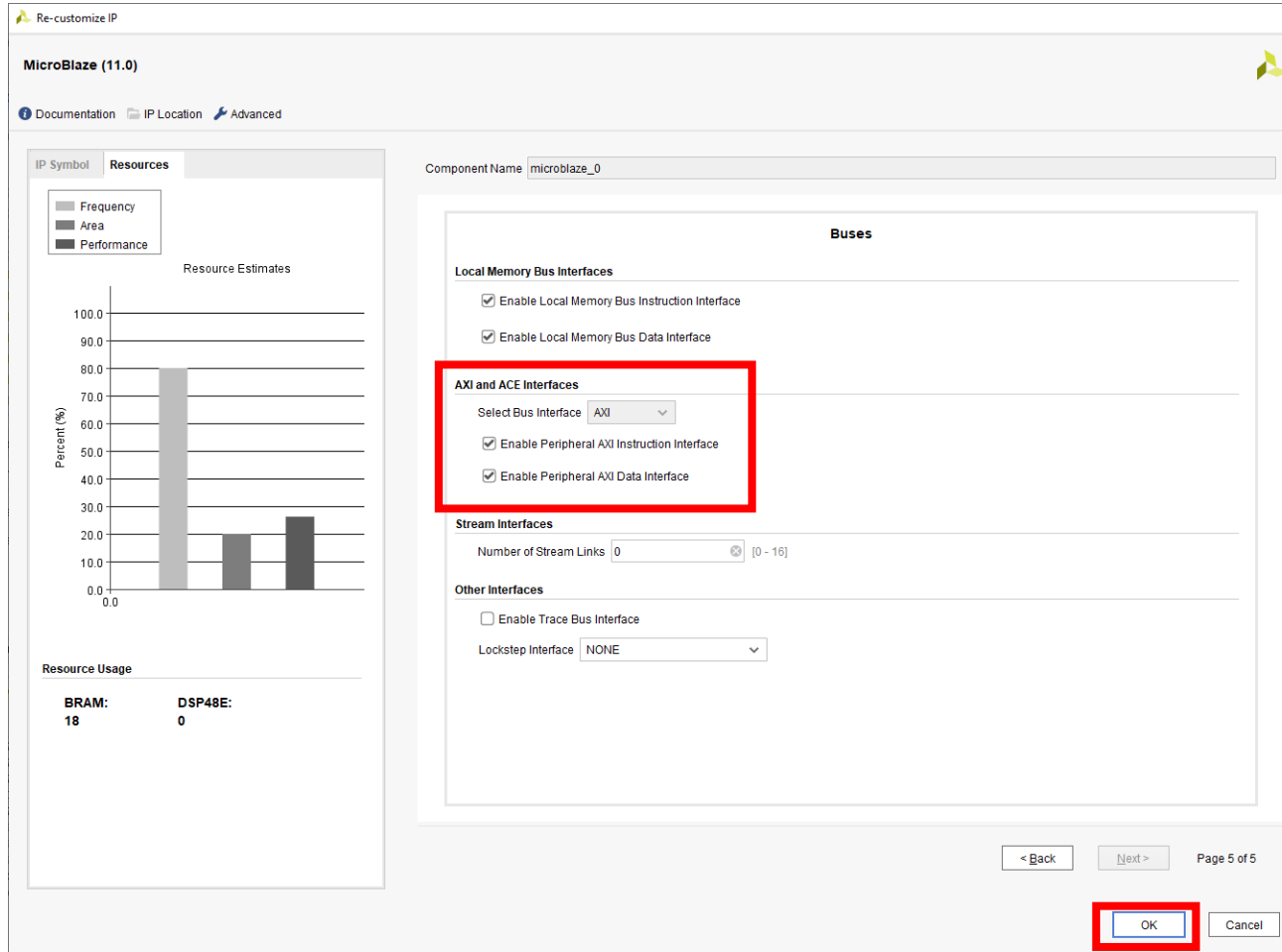
Profile Buffer Size: NONE

< Back Next > Page 4 of

OK Cancel

MicroBlaze Boot From QSPI

Enable the AXI interfaces and click OK



Re-customize IP

MicroBlaze (11.0)

Documentation IP Location Advanced

IP Symbol Resources

Component Name: microblaze_0

Buses

Local Memory Bus Interfaces

- Enable Local Memory Bus Instruction Interface
- Enable Local Memory Bus Data Interface

AXI and ACE Interfaces

Select Bus Interface: AXI

- Enable Peripheral AXI Instruction Interface
- Enable Peripheral AXI Data Interface

Stream Interfaces

Number of Stream Links: 0 [0 - 16]

Other Interfaces

- Enable Trace Bus Interface
- Lockstep Interface: NONE

Resource Estimates

Resource Usage

BRAM:	DSP48E:
18	0

< Back Next > Page 5 of 5

OK Cancel

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

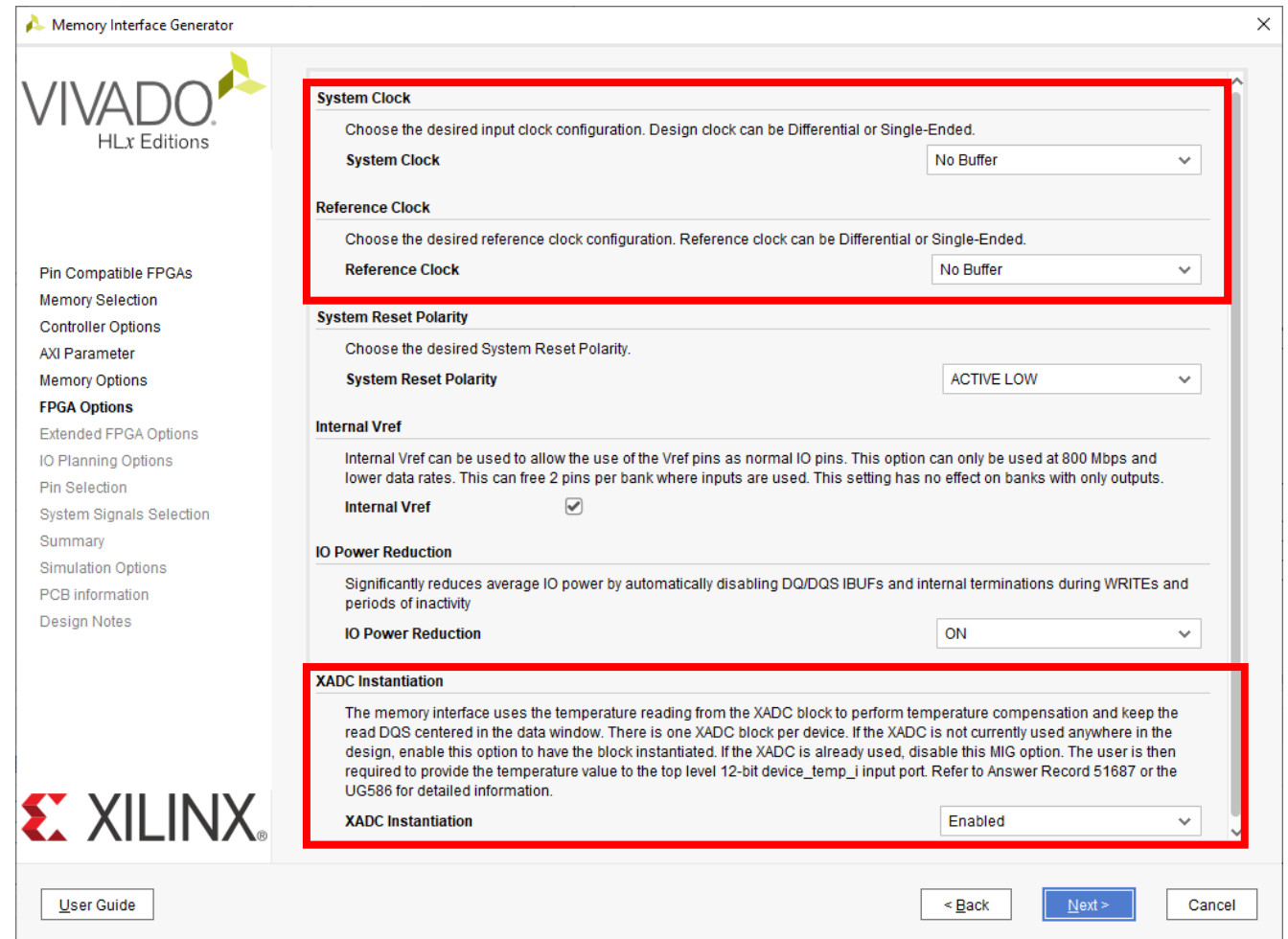
Re-customize the Clock Wizard associated with the System Clock to have no reset input, no locked signal and to output 100MHz and 200MHz

Board Clocking Options Output Clocks MMCM Settings Summary									
The phase is calculated relative to the active input clock.									
Output Clock	Port Name	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)		Drives	
		Requested	Actual	Requested	Actual	Requested	Actual		
<input checked="" type="checkbox"/> clk_out1	clk_out1	100.000	100.000000	0.000	0.000	50.000	50.0	BUFG	
<input checked="" type="checkbox"/> clk_out2	clk_out2	200.000	200.000000	0.000	0.000	50.000	50.0	BUFG	

MicroBlaze Boot From QSPI

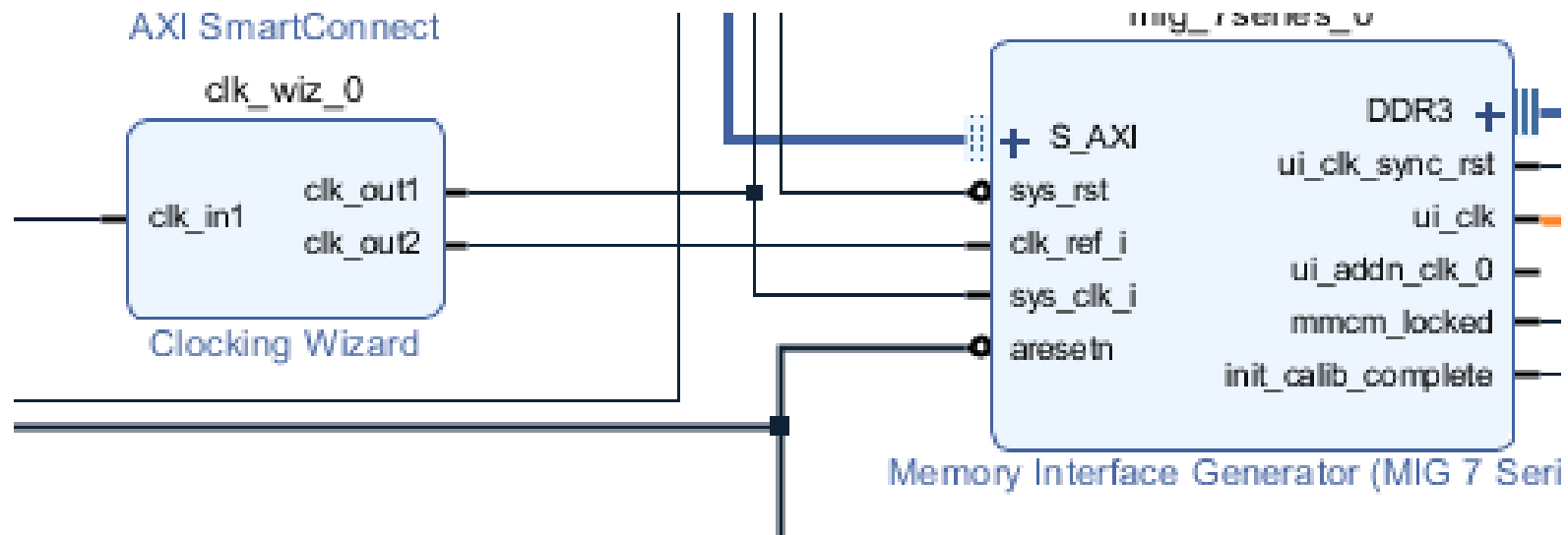
Re Customize the DDR MIG
Change

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



MicroBlaze Boot From QSPI

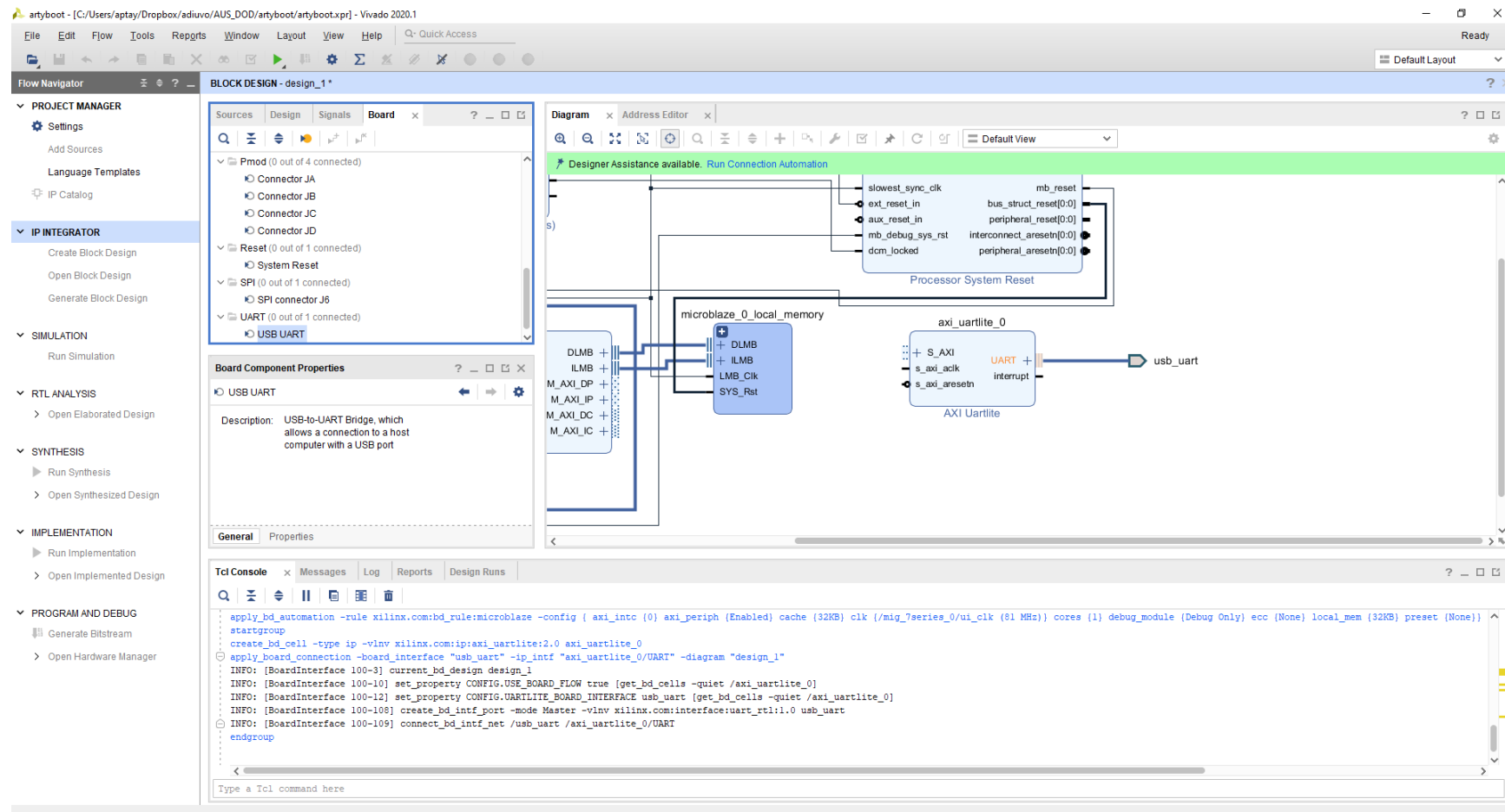
Connect the 100 MHz to the Sys_clk and 200 MHz to the Clk_ref



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



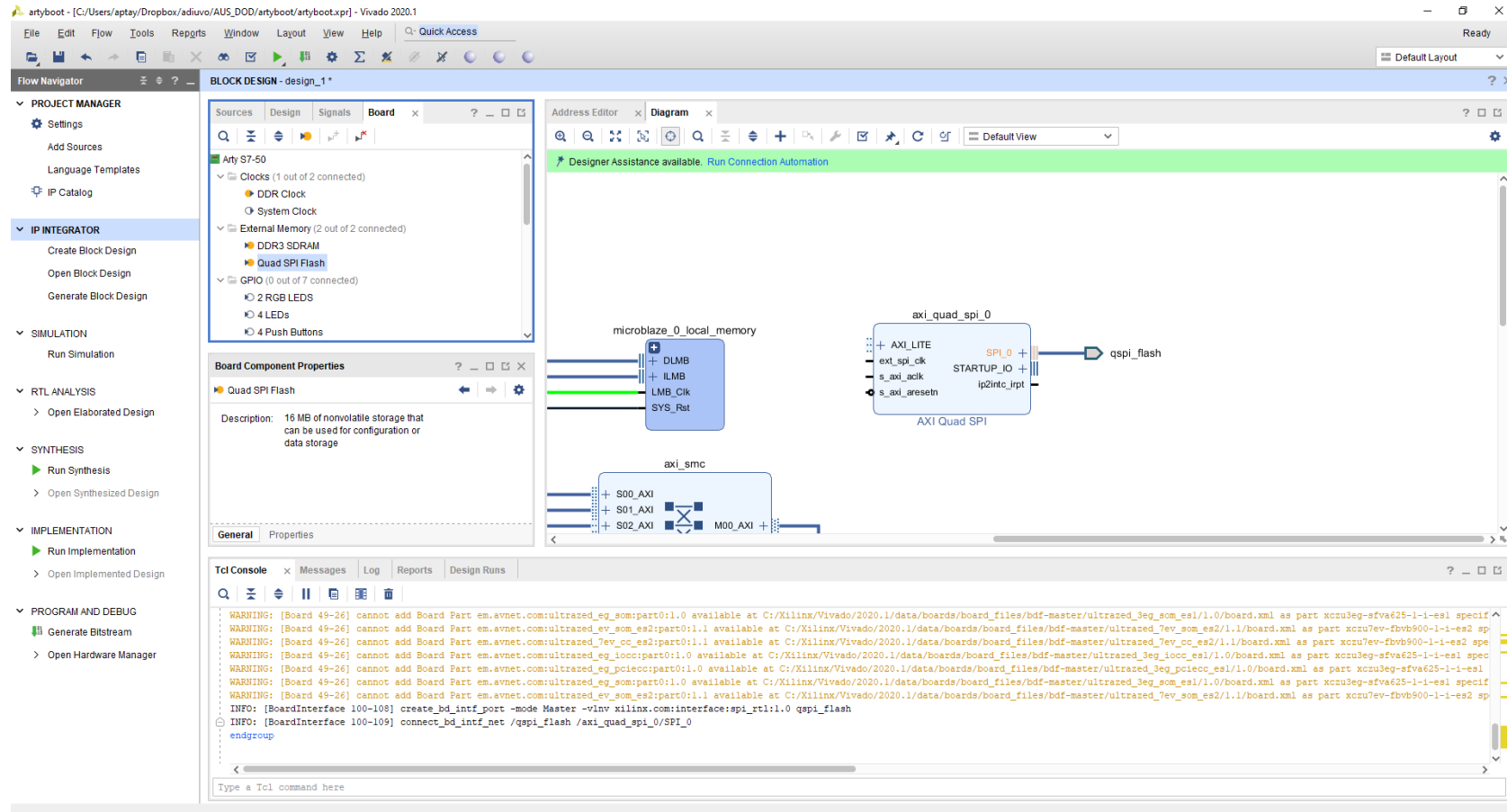
The screenshot displays the Vivado 2020.1 interface for a project named 'artyboot'. The 'Sources' pane on the left shows the 'USB UART' component under the 'UART' category. The 'Diagram' pane shows a block diagram with 'microblaze_0_local_memory' and 'axi_uartlite_0' blocks. The 'Tcl Console' at the bottom shows the commands used to create the board interface and connect the USB UART component.

```

apply_bd_automation -rule xilinx.com:bd_rule:microblaze -config { axi_intc (0) axi_periph [Enabled] cache (32KB) clk (/mig_7series_0/ui_clk (81 MHz)) cores (1) debug_module (Debug Only) ecc (None) local_mem (32KB) preset (None) }
startgroup
create_bd_cell -type ip -vlnv xilinx.com:ip:axi_uartlite:2.0 axi_uartlite_0
apply_board_connection -board_interface "usb_uart" -ip_intf "axi_uartlite_0/UART" -diagram "design_1"
INFO: [BoardInterface 100-3] current_bd_design design_1
INFO: [BoardInterface 100-10] set_property CONFIG.USE_BOARD_FLOW true [get_bd_cells -quiet /axi_uartlite_0]
INFO: [BoardInterface 100-12] set_property CONFIG.UARTLITE_BOARD_INTERFACE usb_uart [get_bd_cells -quiet /axi_uartlite_0]
INFO: [BoardInterface 100-108] create_bd_intf_port -mode Master -vlnv xilinx.com:interface:uart_rtl:1.0 usb_uart
INFO: [BoardInterface 100-109] connect_bd_intf_net /usb_uart /axi_uartlite_0/UART
endgroup
  
```

MicroBlaze Boot From QSPI

Drag the Quad SPI flash onto the Block Diagram



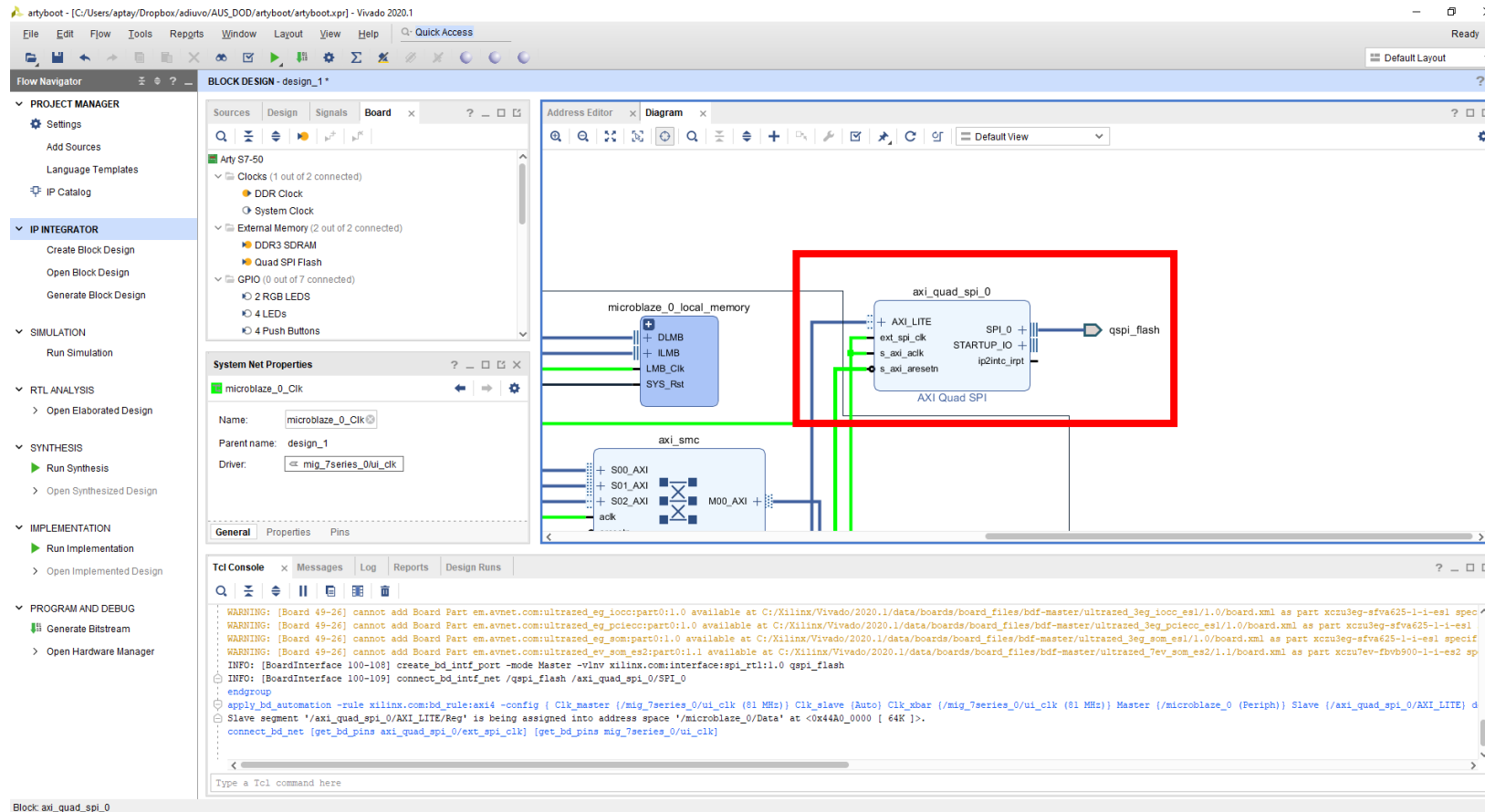
The screenshot displays the Vivado 2020.1 interface for a project named 'artyboot'. The 'Sources' pane on the left shows the 'Board' view for 'Arty S7-50', with 'External Memory' containing 'Quad SPI Flash'. The 'Diagram' pane shows a block diagram with 'microblaze_0_local_memory' connected to 'axi_quad_spi_0' (AXI Quad SPI) and 'axi_smc'. The 'Tcl Console' at the bottom shows the following command and output:

```

INFO: [BoardInterface 100-108] create_bd_intf_port -mode Master -vlnv xilinx.com:interface:spi_rtl:1.0 qspi_flash
INFO: [BoardInterface 100-109] connect_bd_intf_net /qspi_flash /axi_quad_spi_0/SPI_0
endgroup
  
```

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`



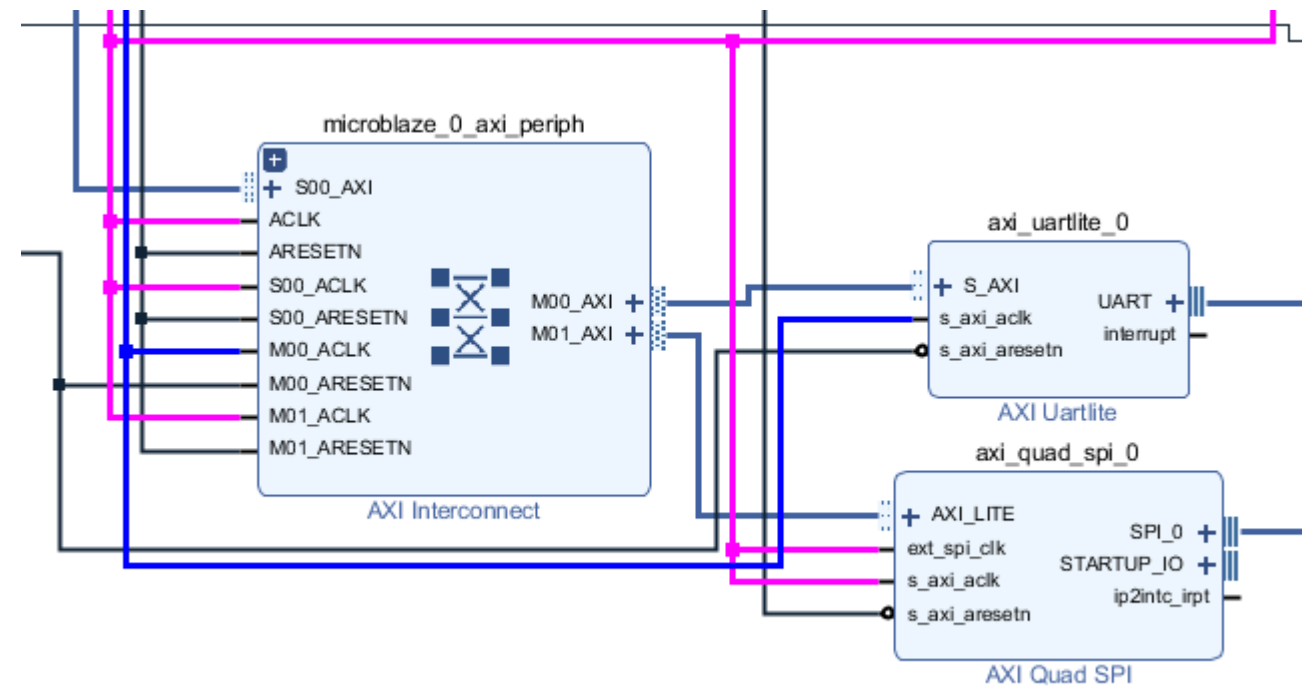
The screenshot displays the Vivado 2020.1 Block Design environment. The main diagram shows the 'axi_quad_spi_0' block (AXI Quad SPI) connected to the 'qspi_flash' block. The 's_axi_clk' input of the 'axi_quad_spi_0' block is connected to the 'ext_spi_clk' output of the 'microblaze_0_local_memory' block. The 'axi_quad_spi_0' block is highlighted with a red box. The 'Tcl Console' at the bottom shows the following command:

```
connect_bd_net [get_bd_pins axi_quad_spi_0/ext_spi_clk] [get_bd_pins mig_7series_0/ui_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

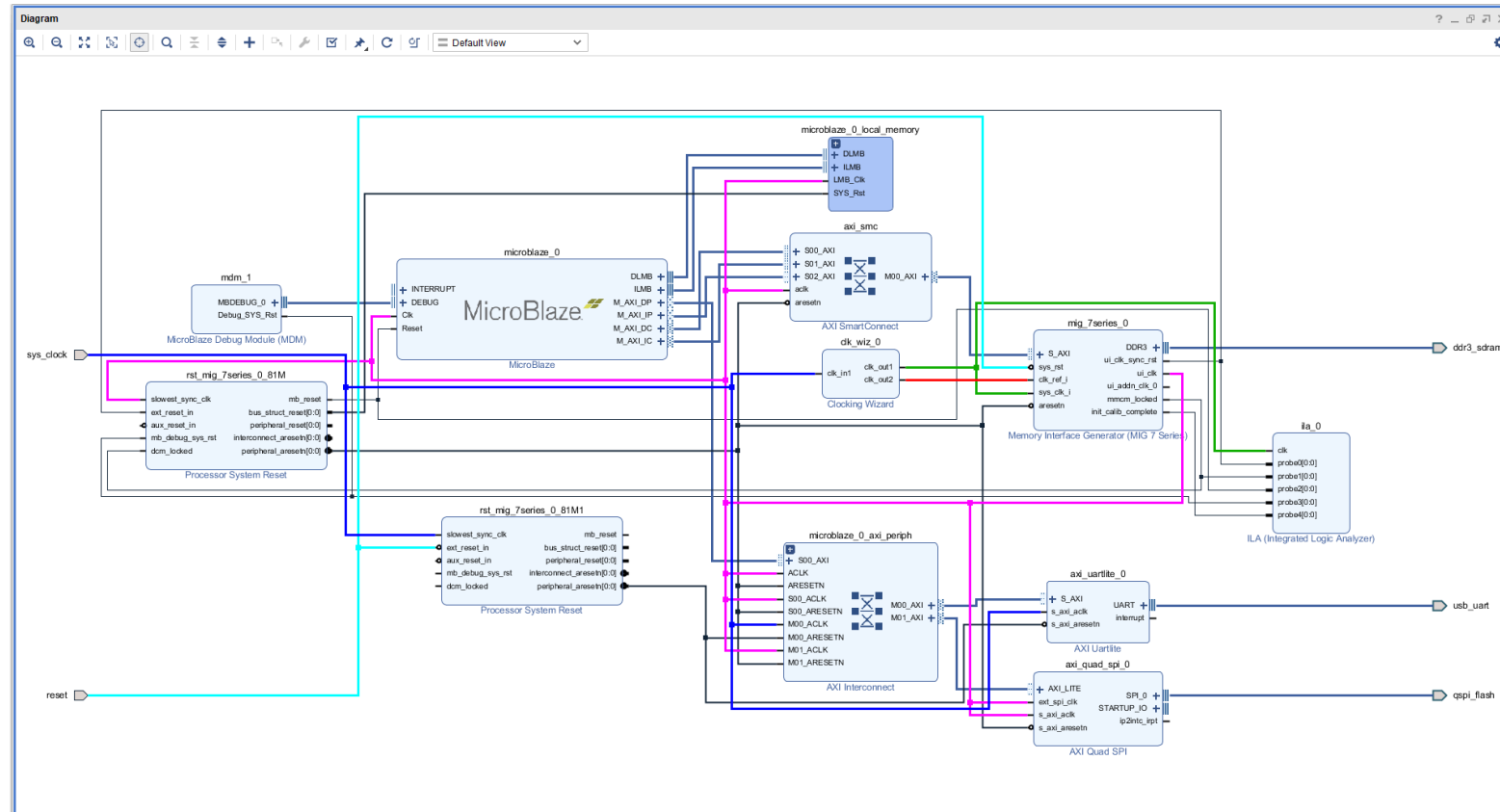
The AXI Interconnect will address the clock domain crossing safely



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

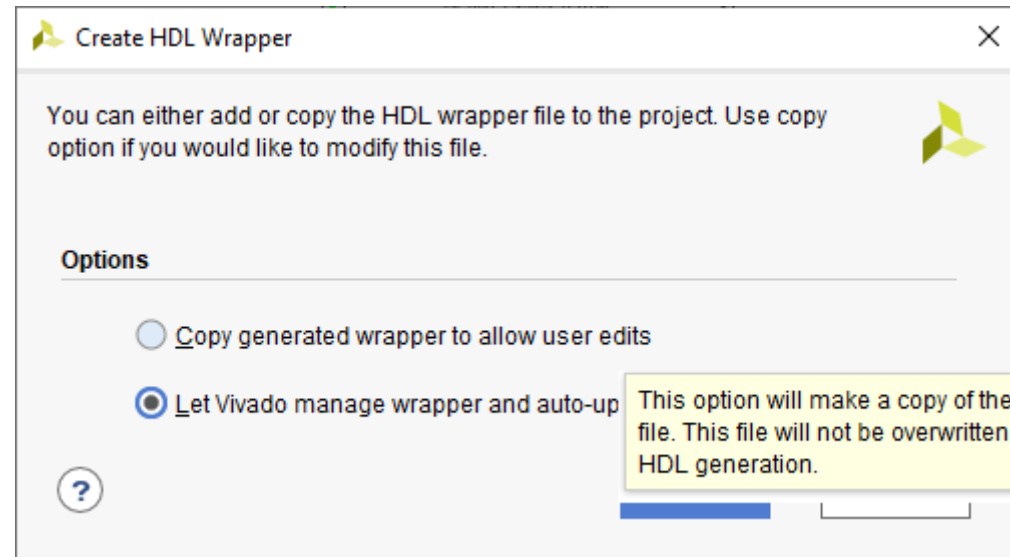
MicroBlaze Boot From QSPI

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



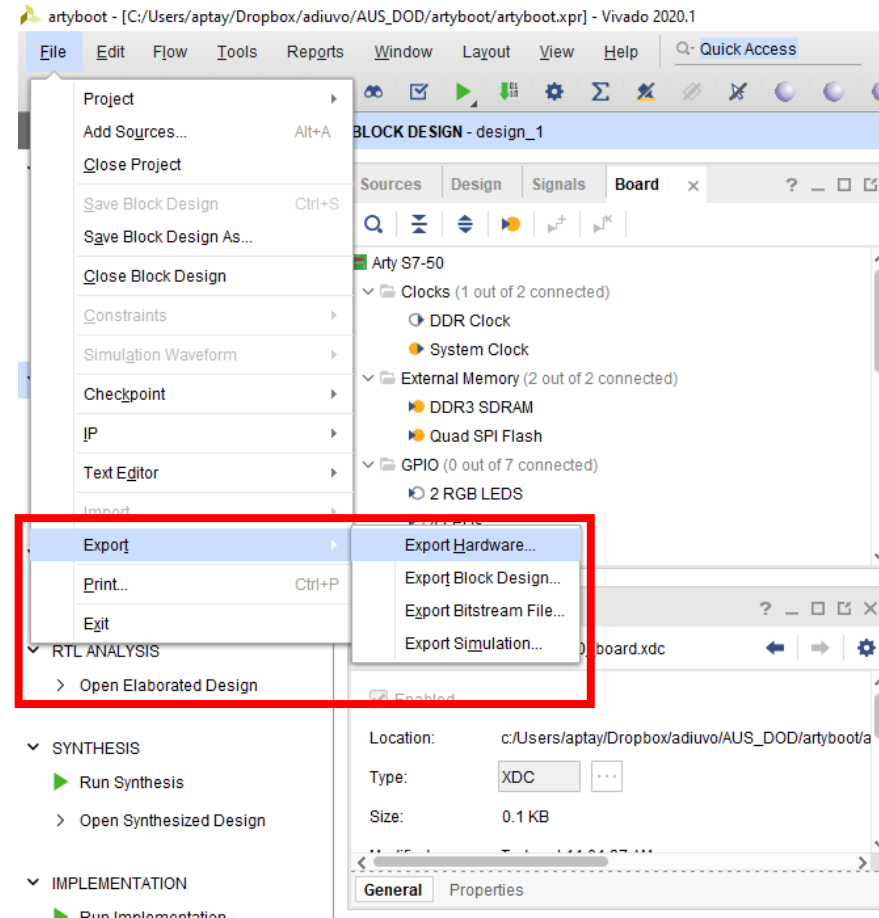
MicroBlaze Boot From QSPI

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



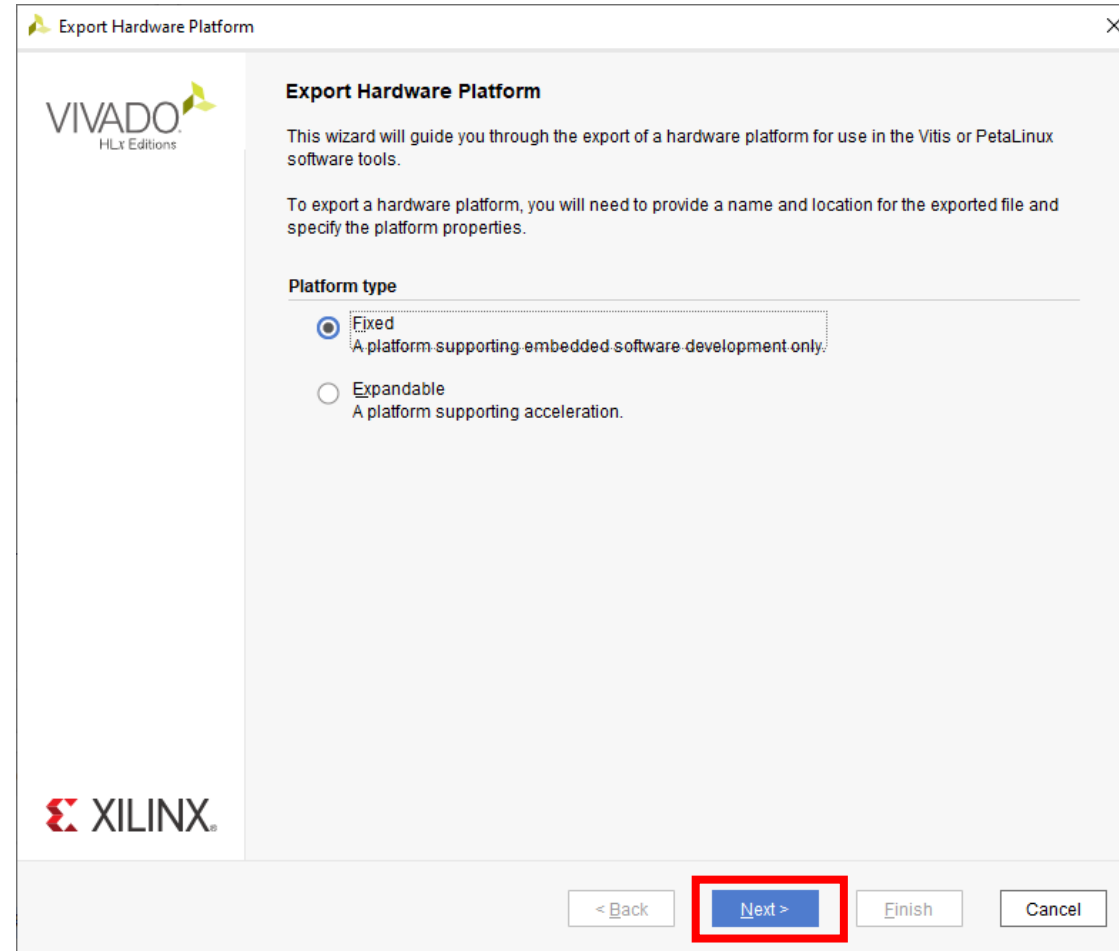
MicroBlaze Boot From QSPI

Once the bit stream is available export the hardware.



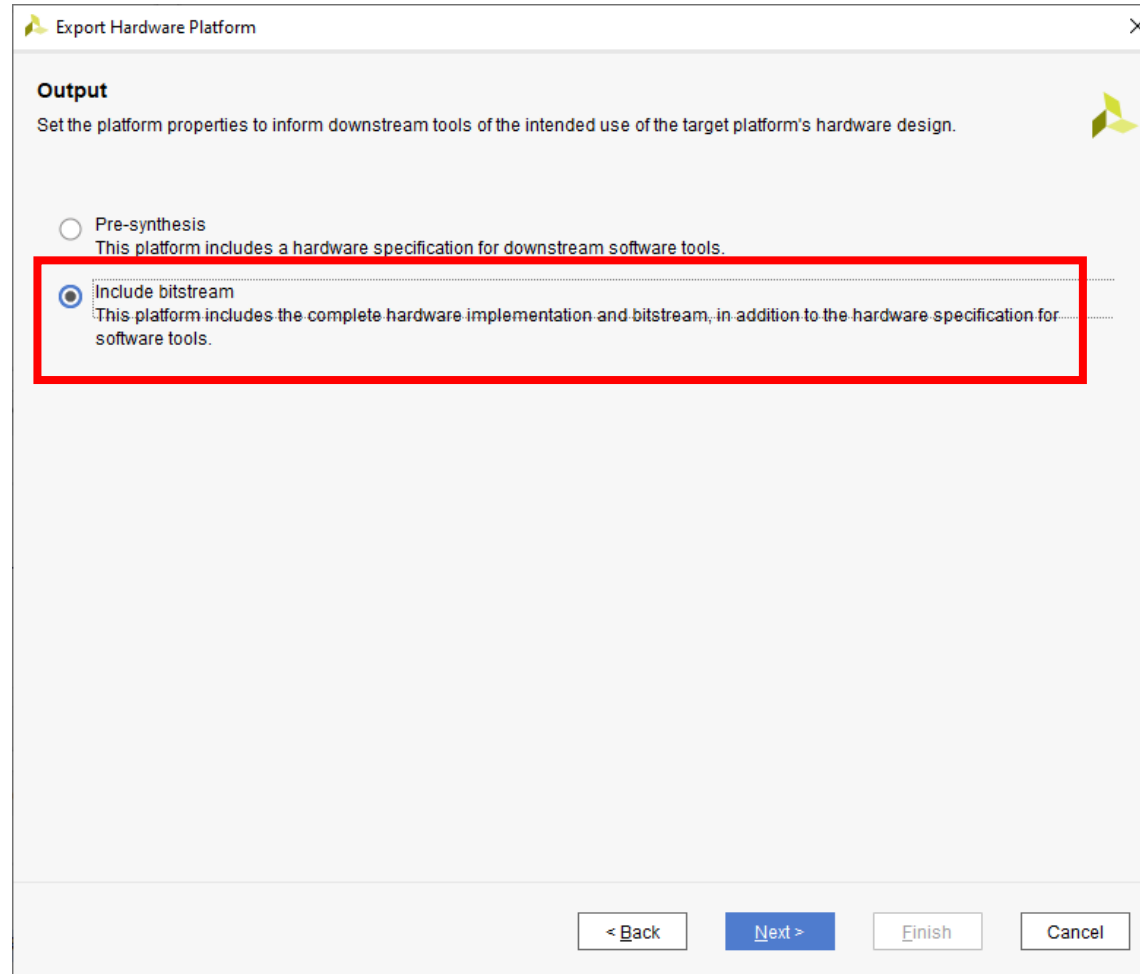
MicroBlaze Boot From QSPI

Select the Fixed platform type



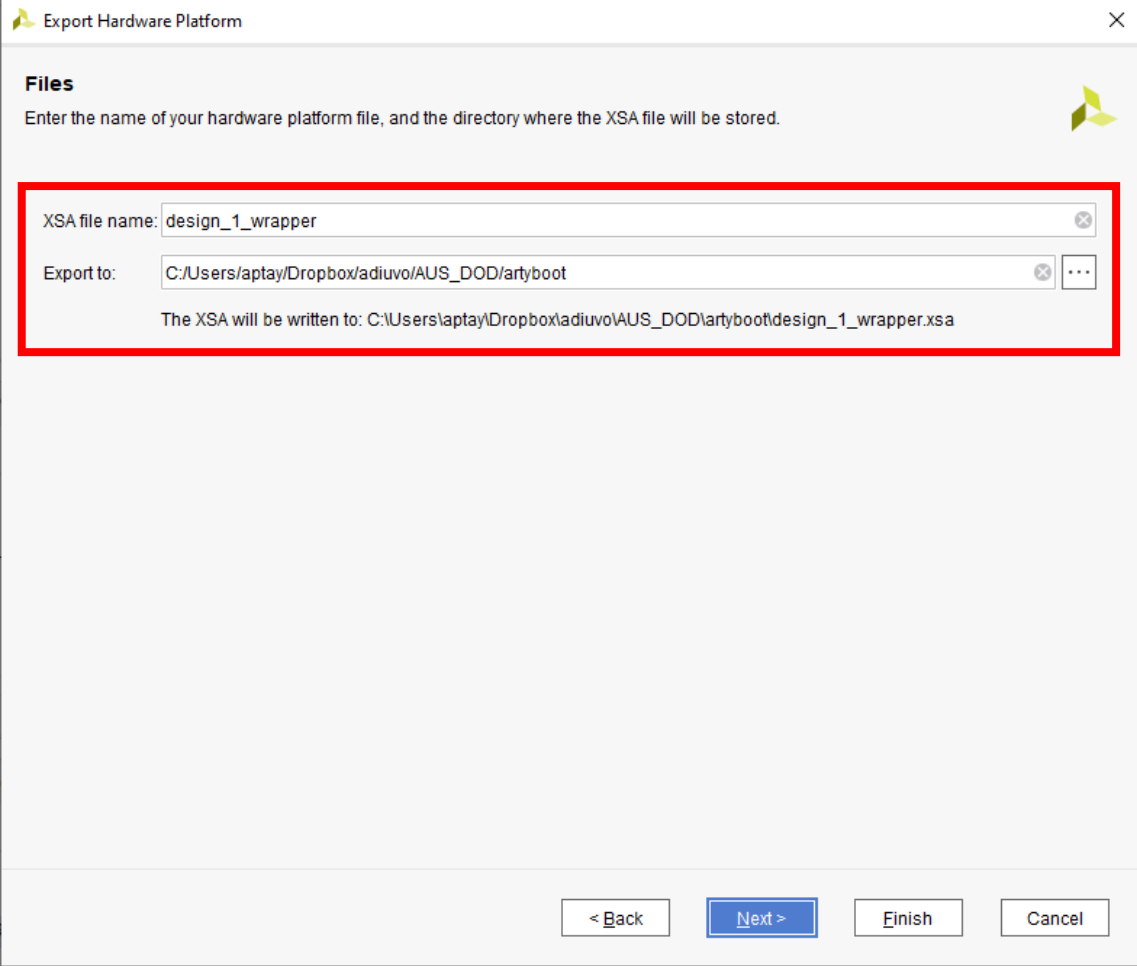
MicroBlaze Boot From QSPI

Include the bitstream



MicroBlaze Boot From QSPI

Select a name and location for the project



Export Hardware Platform

Files

Enter the name of your hardware platform file, and the directory where the XSA file will be stored.

XSA file name: design_1_wrapper

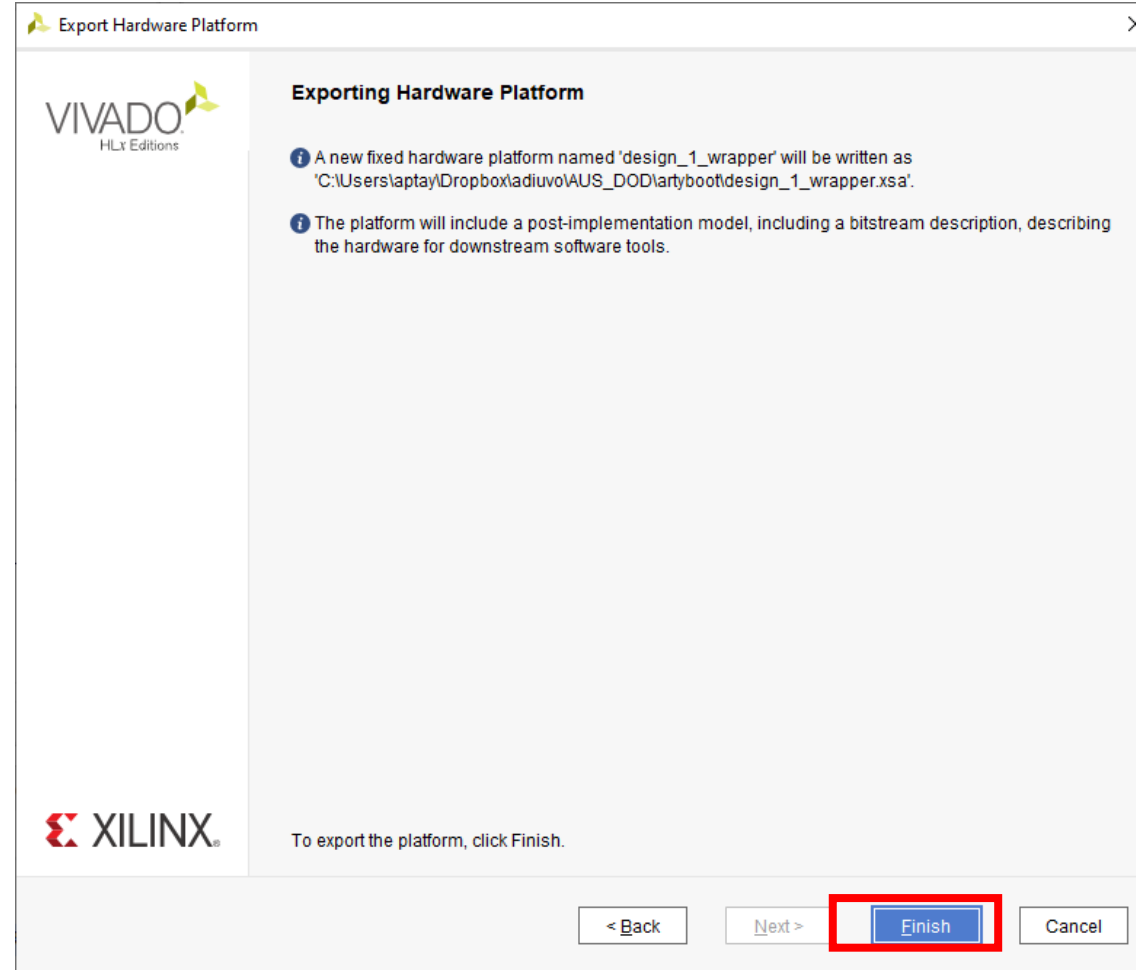
Export to: C:/Users/aptay/Dropbox/adiuvo/AUS_DOD/artyboot

The XSA will be written to: C:\Users\aptay\Dropbox\adiuvo\AUS_DOD\artyboot\design_1_wrapper.xsa

< Back Next > Finish Cancel

MicroBlaze Boot From QSPI

Click on Finish to export the hardware

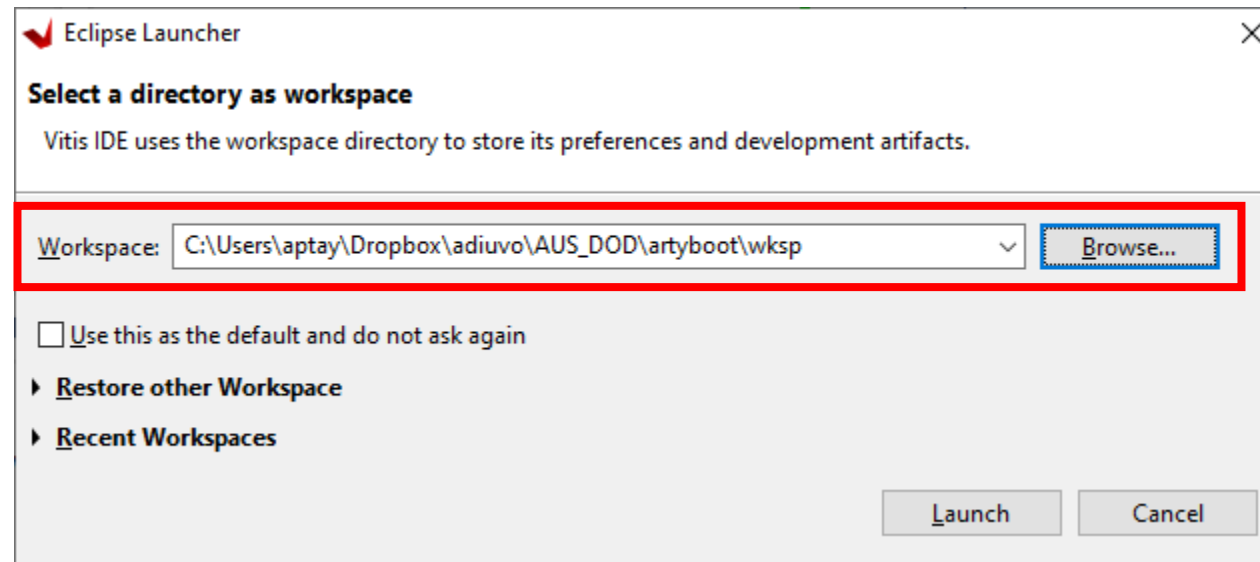




Software Application Development in Vitis

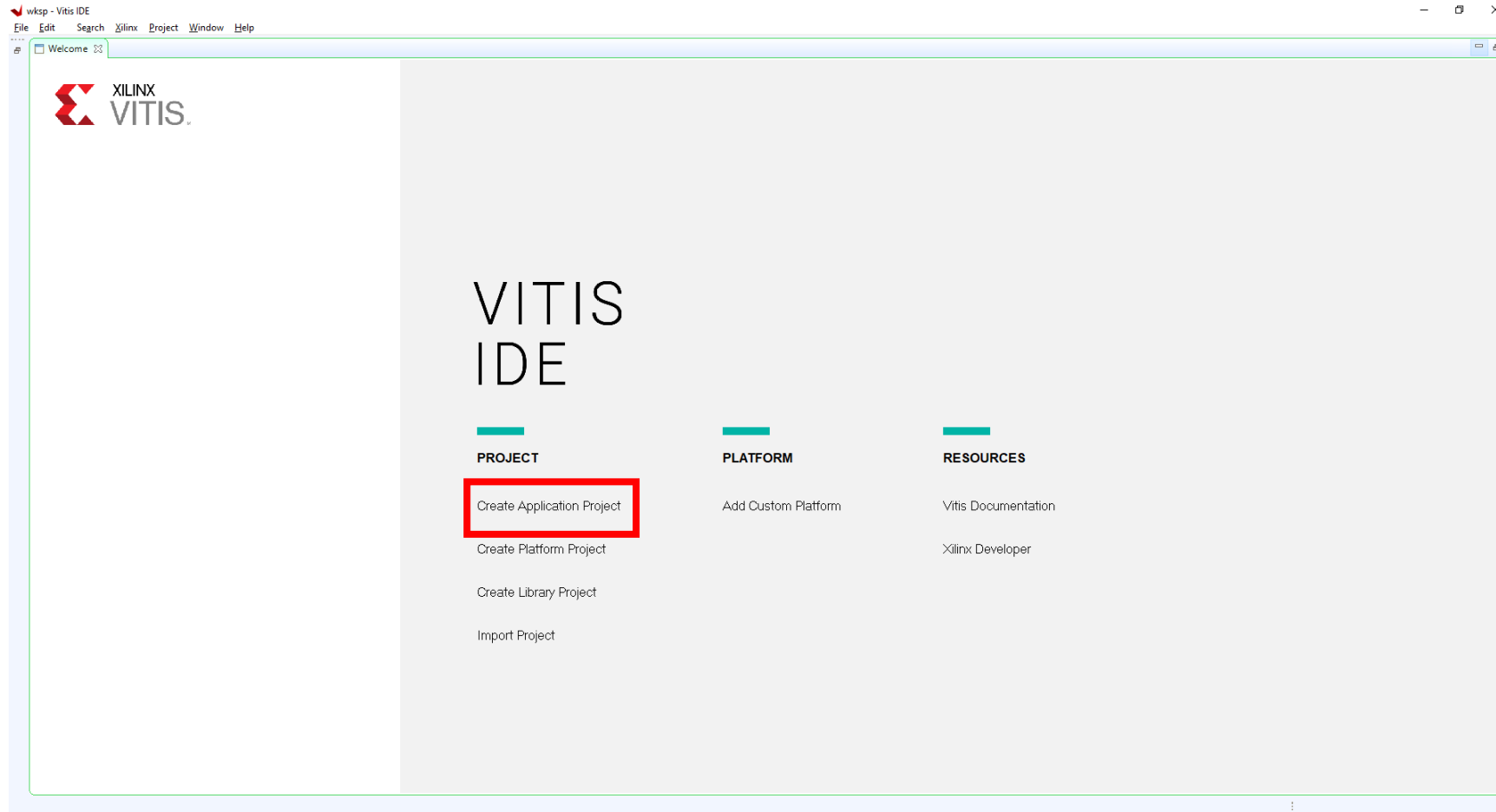
MicroBlaze Boot From QSPI

From the Vivado tools menu launch Vitis and select a workspace



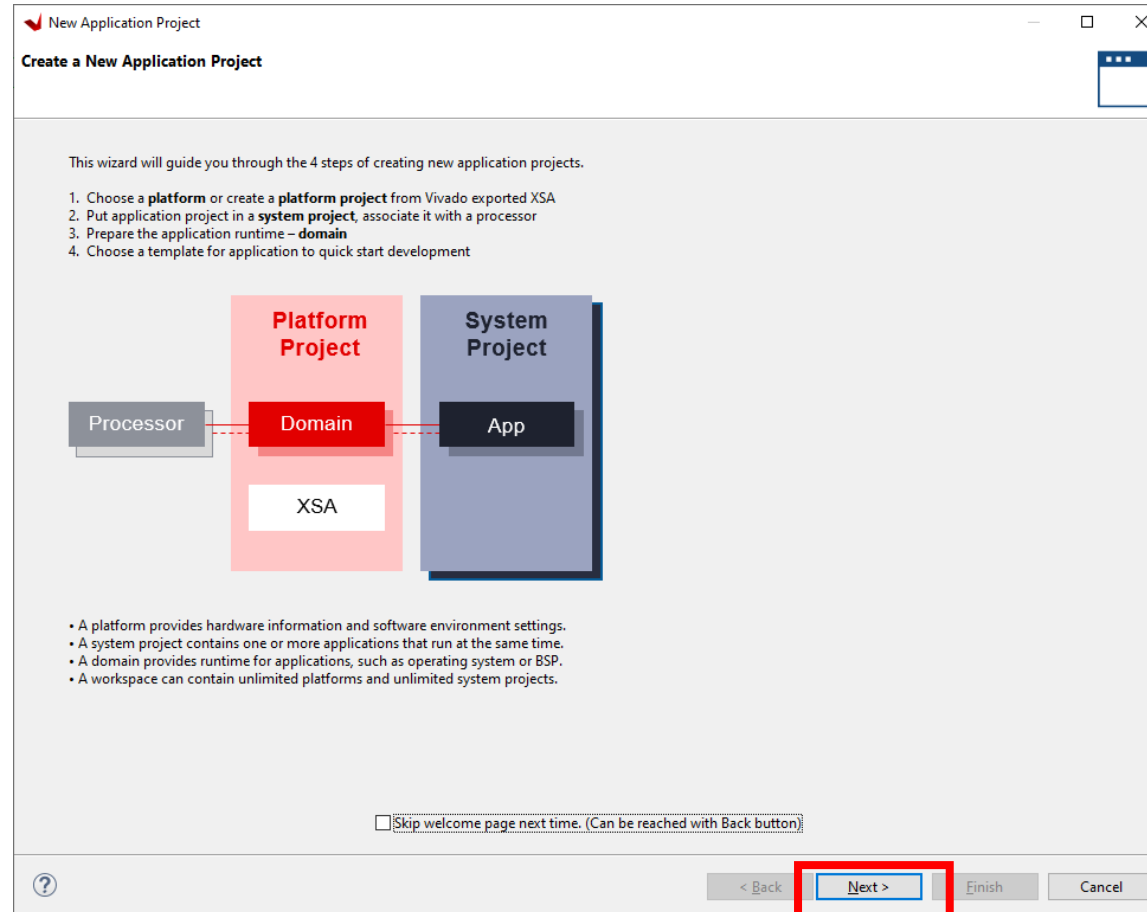
MicroBlaze Boot From QSPI

Select create application project



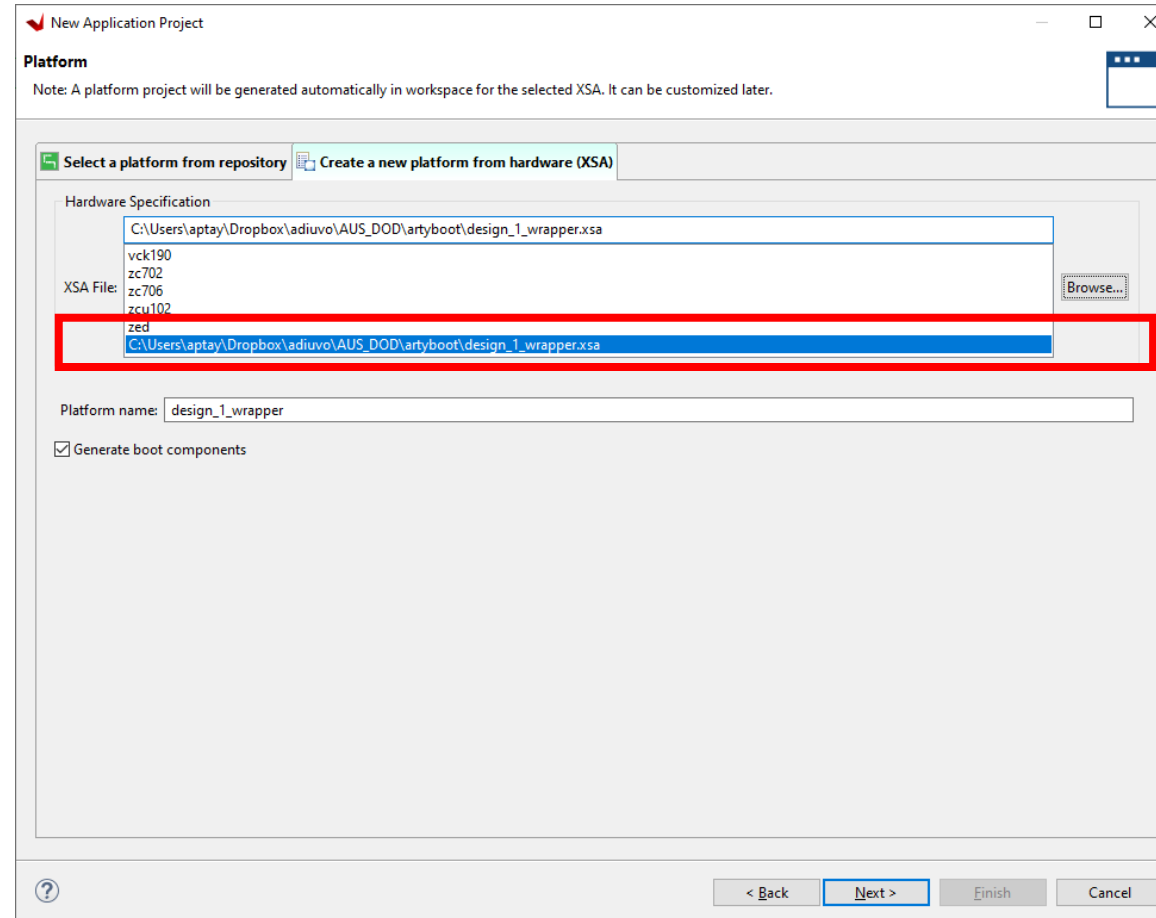
MicroBlaze Boot From QSPI

Click Next on the dialog



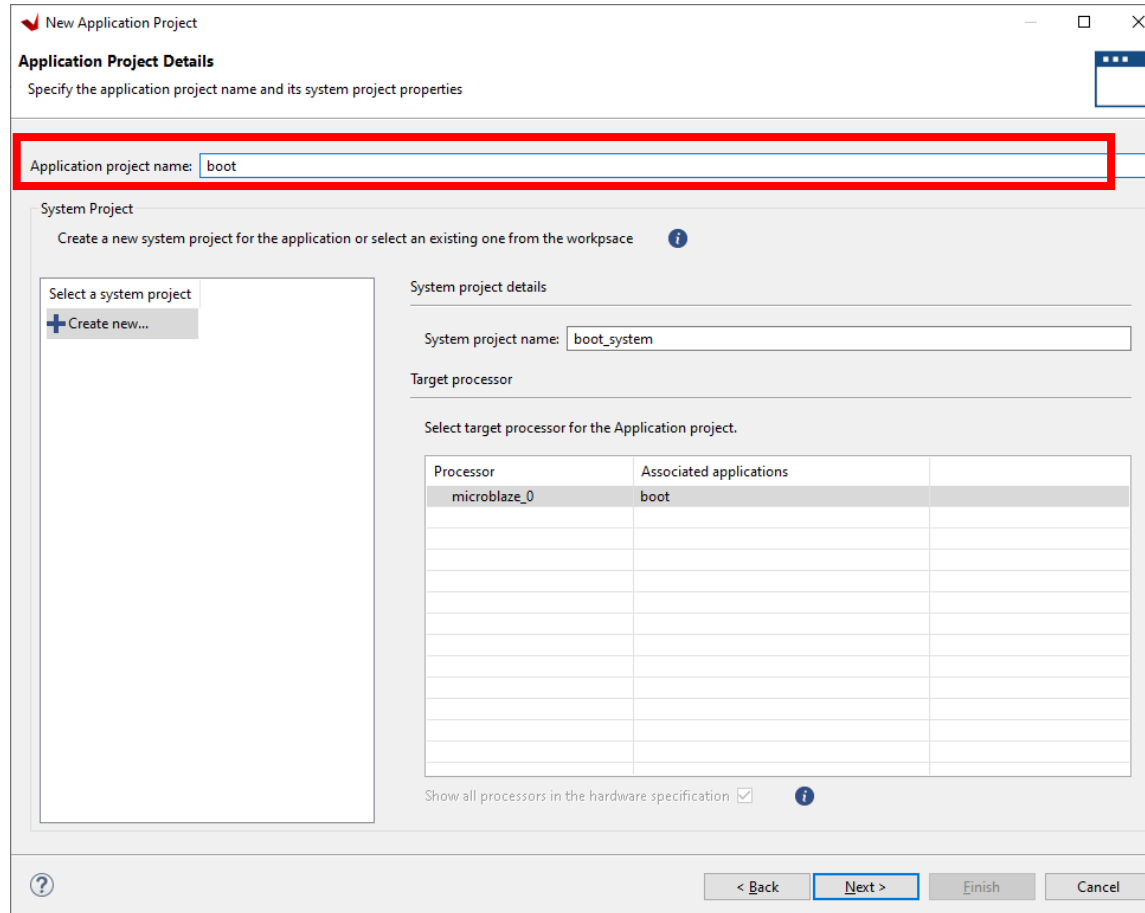
MicroBlaze Boot From QSPI

Select the Hardware platform just exported



MicroBlaze Boot From QSPI

Select the name of the project



New Application Project

Application Project Details
Specify the application project name and its system project properties

Application project name:

System Project

Create a new system project for the application or select an existing one from the workspace

Select a system project
[+ Create new...](#)

System project details

System project name:

Target processor

Select target processor for the Application project.

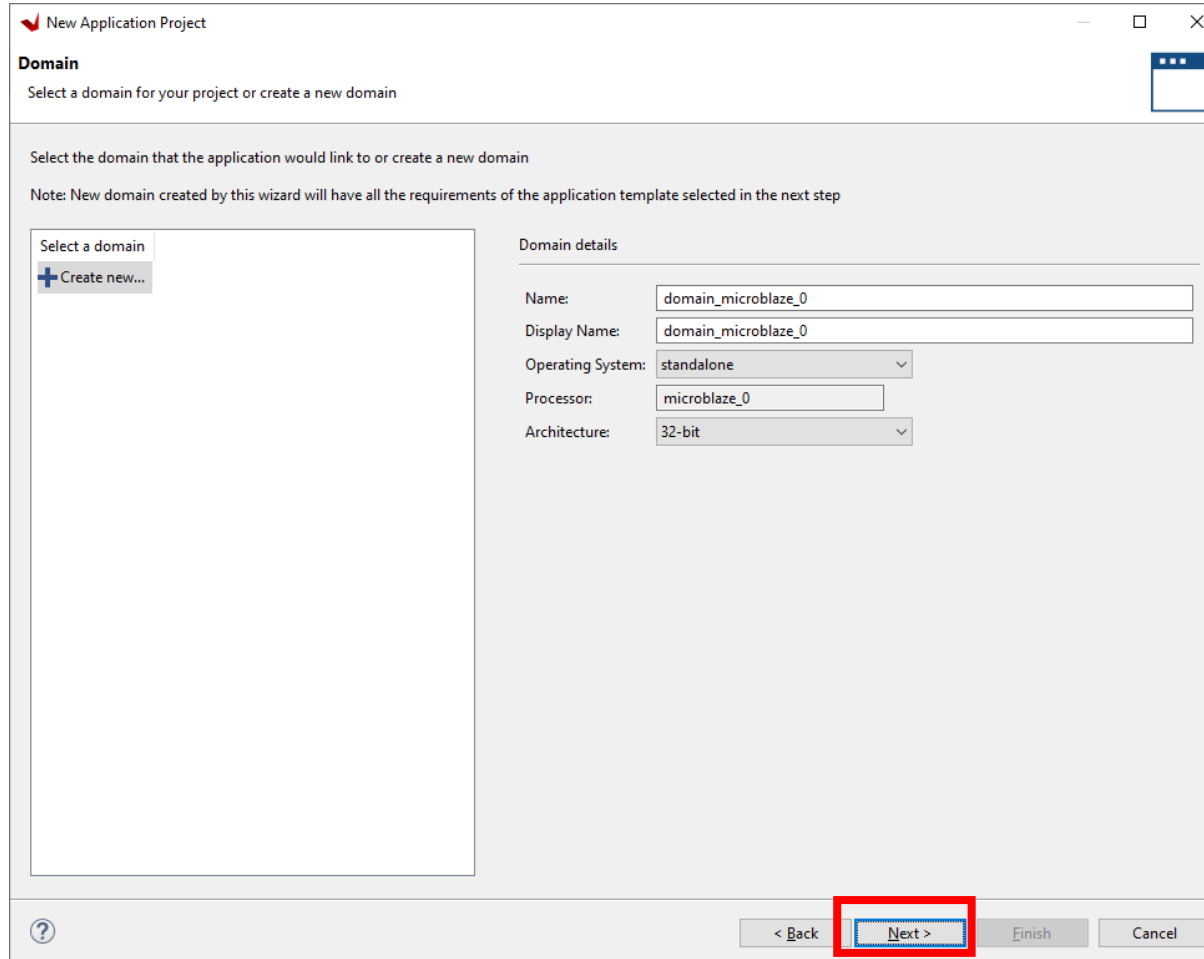
Processor	Associated applications
microblaze_0	boot

Show all processors in the hardware specification

< Back Next > Finish Cancel

MicroBlaze Boot From QSPI

Leave the domain unchanged



New Application Project

Domain
Select a domain for your project or create a new domain

Select the domain that the application would link to or create a new domain

Note: New domain created by this wizard will have all the requirements of the application template selected in the next step

Select a domain
[+ Create new...](#)

Domain details

Name:

Display Name:

Operating System: standalone

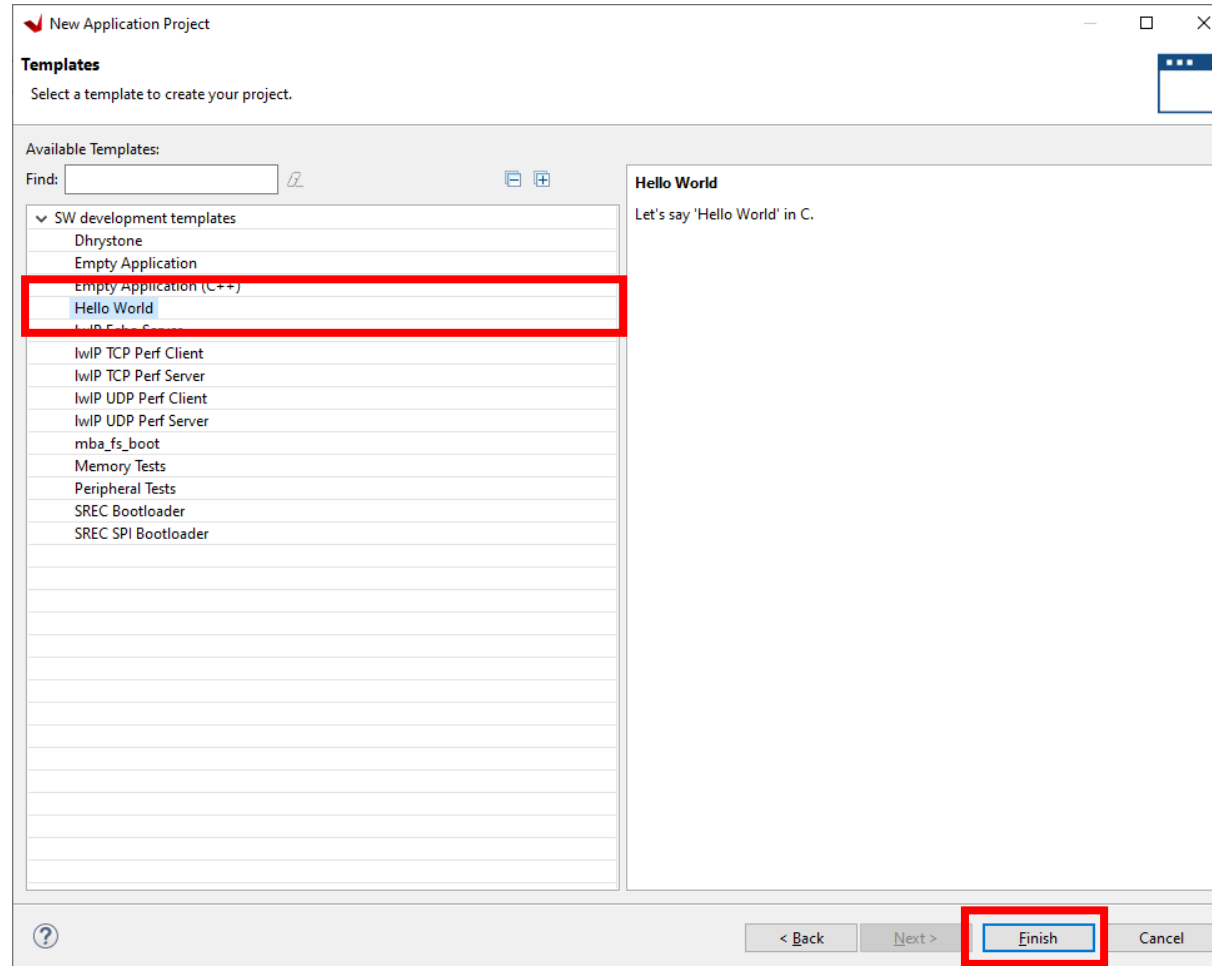
Processor:

Architecture: 32-bit

[? < Back](#) [Next >](#) [Finish](#) [Cancel](#)

MicroBlaze Boot From QSPI

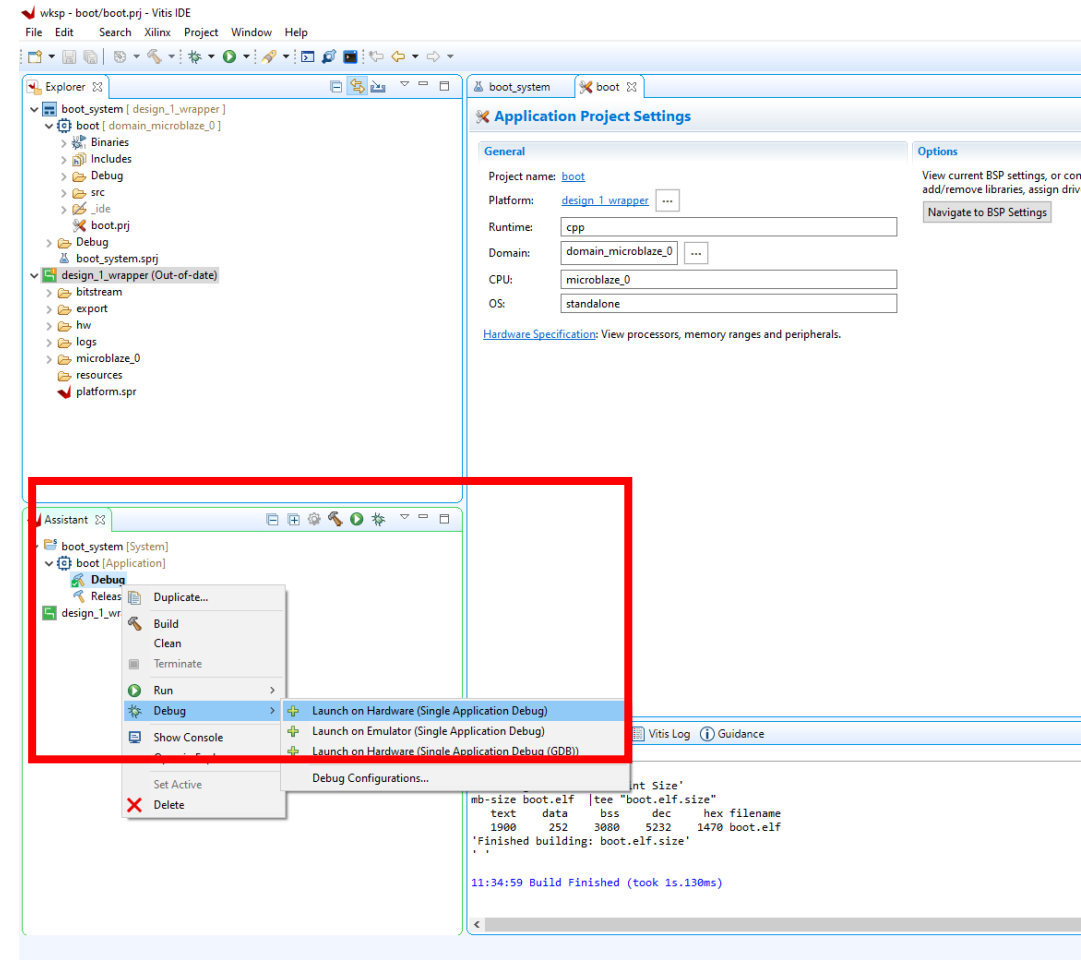
Select the Hello World Project



MicroBlaze Boot From QSPI

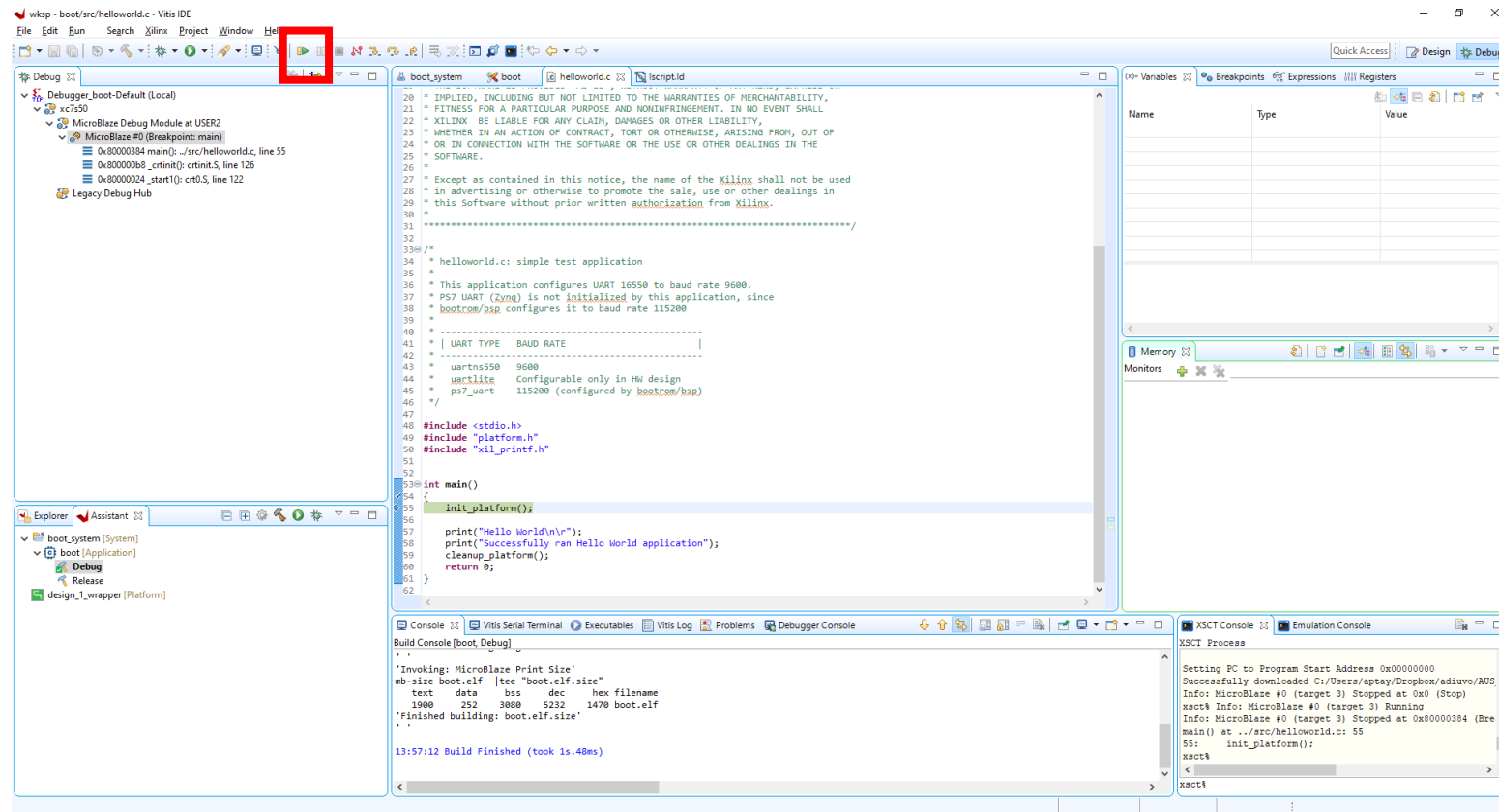
With the project created, debug the application on the hardware to ensure the MicroBlaze is running correctly from DDR before we build the final applications.

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



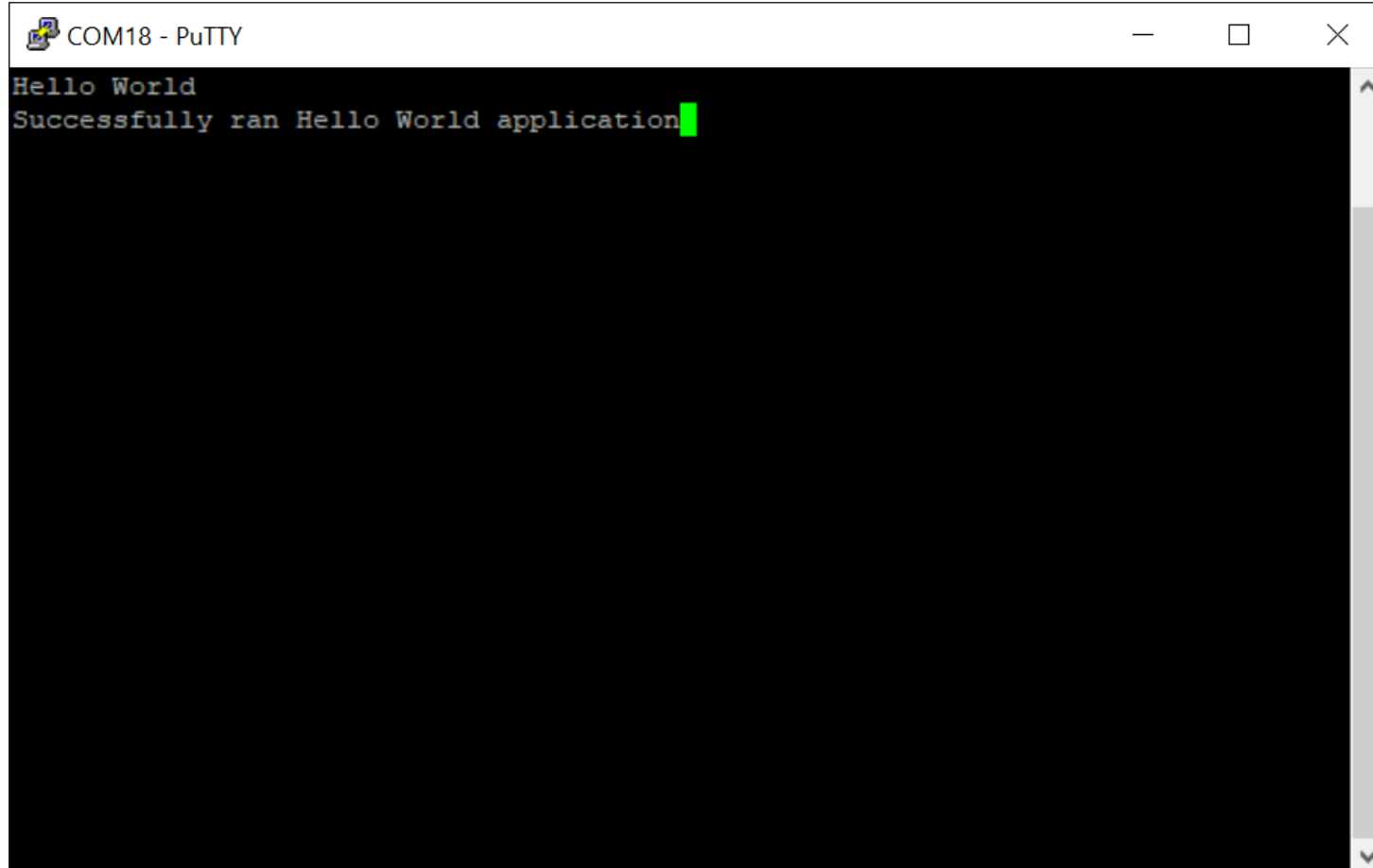
MicroBlaze Boot From QSPI

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

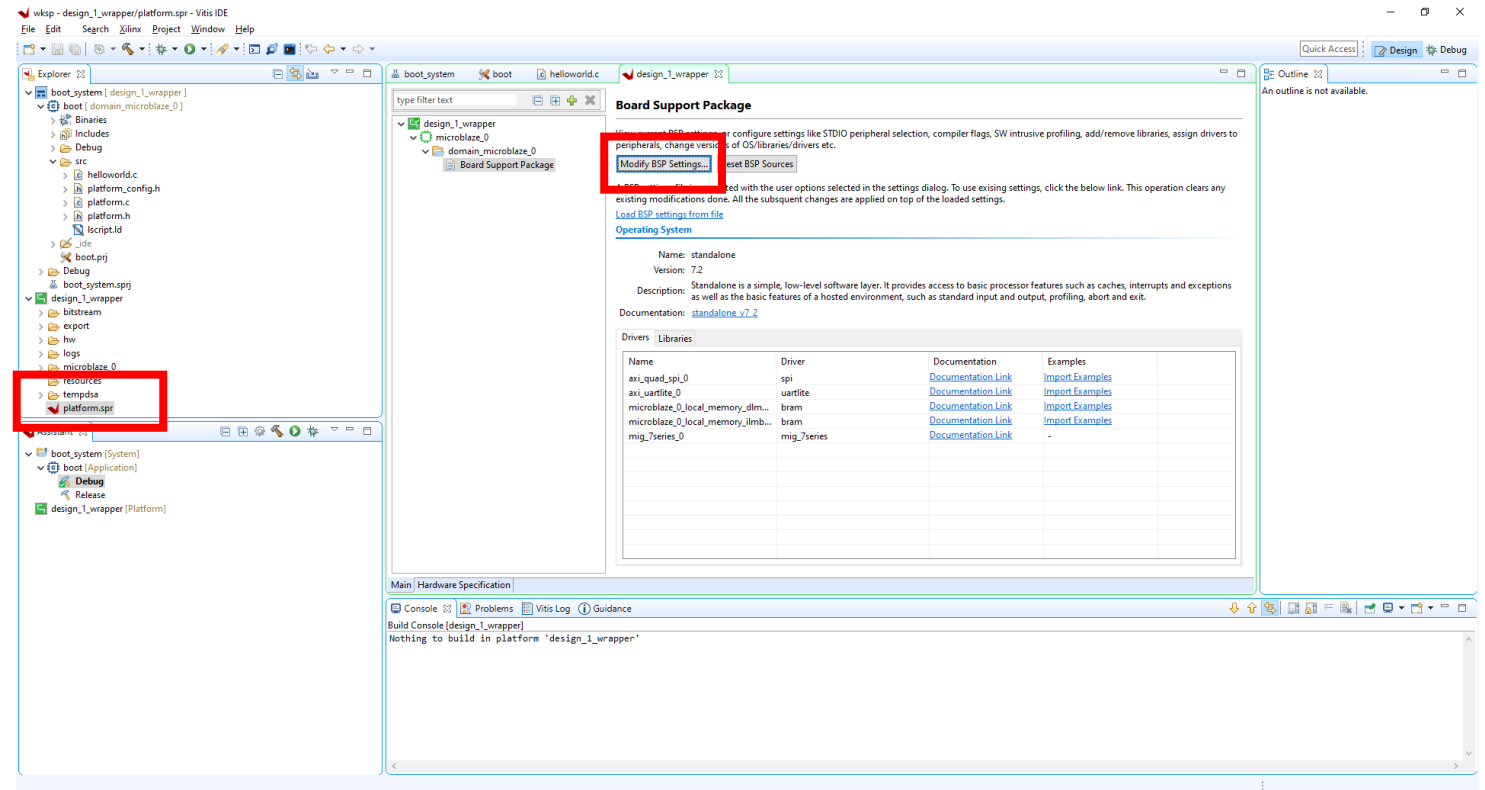


```
COM18 - PuTTY
Hello World
Successfully ran Hello World application
```

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

Board Support Package Settings

Board Support Package Settings
Control various settings of your Board Support Package.

Overview

- standalone
 - xilif
- drivers
 - microblaze_0

C:/Users/aptay/Dropbox/adiuvo/AUS_DOD/artyboot/wksp/design_1_wrapper/microblaze_0/domain_microblaze_0/bsp/system.mss

OS Type: *standalone* Standalone is a simple, low-level software layer. It provides access to basic processor features such as caches, interrupts and exceptions as well as the basic features of a hosted environment, such as standard input and output, profiling, abort and exit.

OS Version: 7.2

Target Hardware

Hardware Specification: C:/Users/aptay/Dropbox/adiuvo/AUS_DOD/artyboot/wksp/design_1_wrapper/hw/design_1_wrapper.xsa

Processor: microblaze_0

Supported Libraries

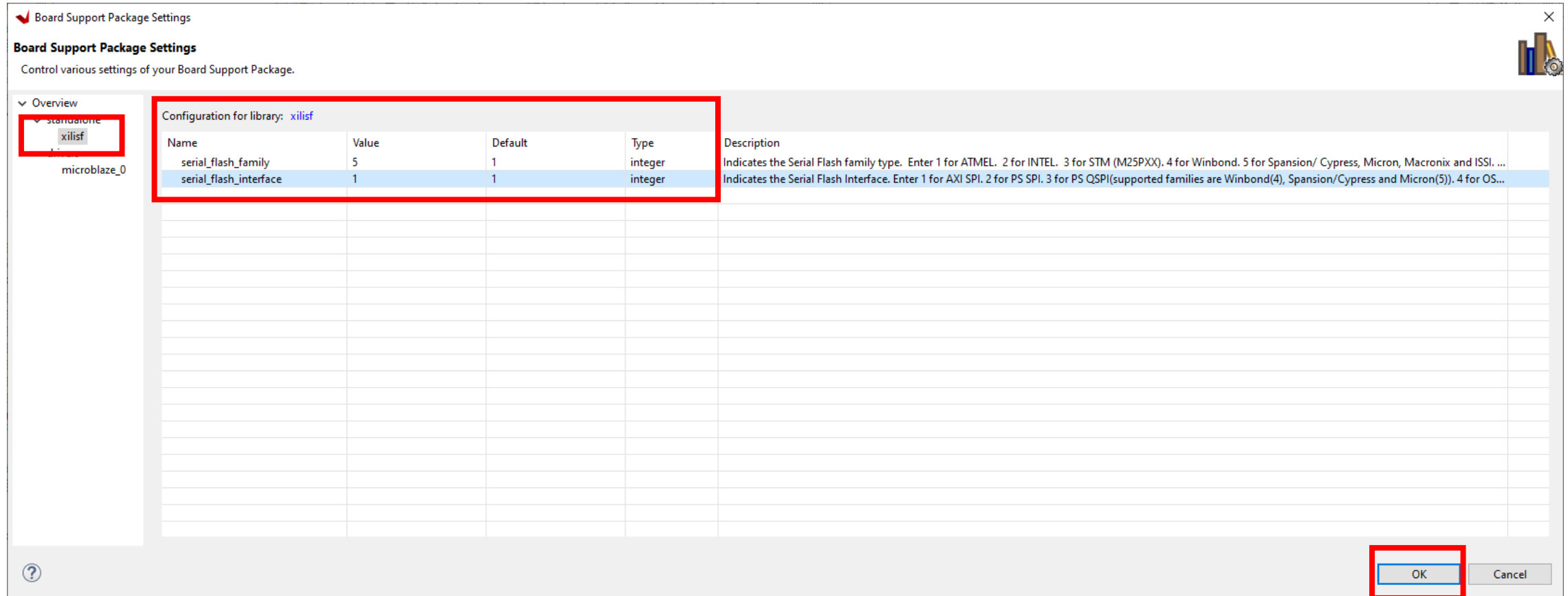
Check the box next to the libraries you want included in your Board Support Package. You can configure the library in the navigator on the left.

Name	Version	Description
<input type="checkbox"/> libmetal	2.1	Libmetal Library
<input type="checkbox"/> lwip211	1.2	Lwip211 library: lwIP (light weight IP) is an open source TCP/IP stack configured for Xilinx hard and soft Ethernet MACs
<input type="checkbox"/> xilffs	4.3	Generic Fat File System Library
<input type="checkbox"/> xilflash	4.8	Xilinx Flash library for Intel/AMD CFI compliant parallel flash
<input checked="" type="checkbox"/> xilif	5.15	Xilinx In-system and Serial Flash Library WARNING: Xilif library is being deprecated from 2020.1 release. It will be made obsolete in 2021.1 release.
<input type="checkbox"/> xilloader	1.1	Xilinx Versal Platform Loader Library
<input type="checkbox"/> xilplmi	1.1	Xilinx versal Platform Loader and Manager Interface Library
<input type="checkbox"/> xilpm	3.1	Platform Management API Library for ZynqMP and Versal
<input type="checkbox"/> xilsem	1.1	Xilinx Versal Soft Error Mitigation Library
<input type="checkbox"/> xilskey	6.9	Xilinx Secure Key Library supports programming efuse and bbram

OK Cancel

MicroBlaze Boot From QSPI

Click on xilif 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



Board Support Package Settings

Board Support Package Settings
Control various settings of your Board Support Package.

Overview

- standalone
 - xilif**
 - microblaze_0

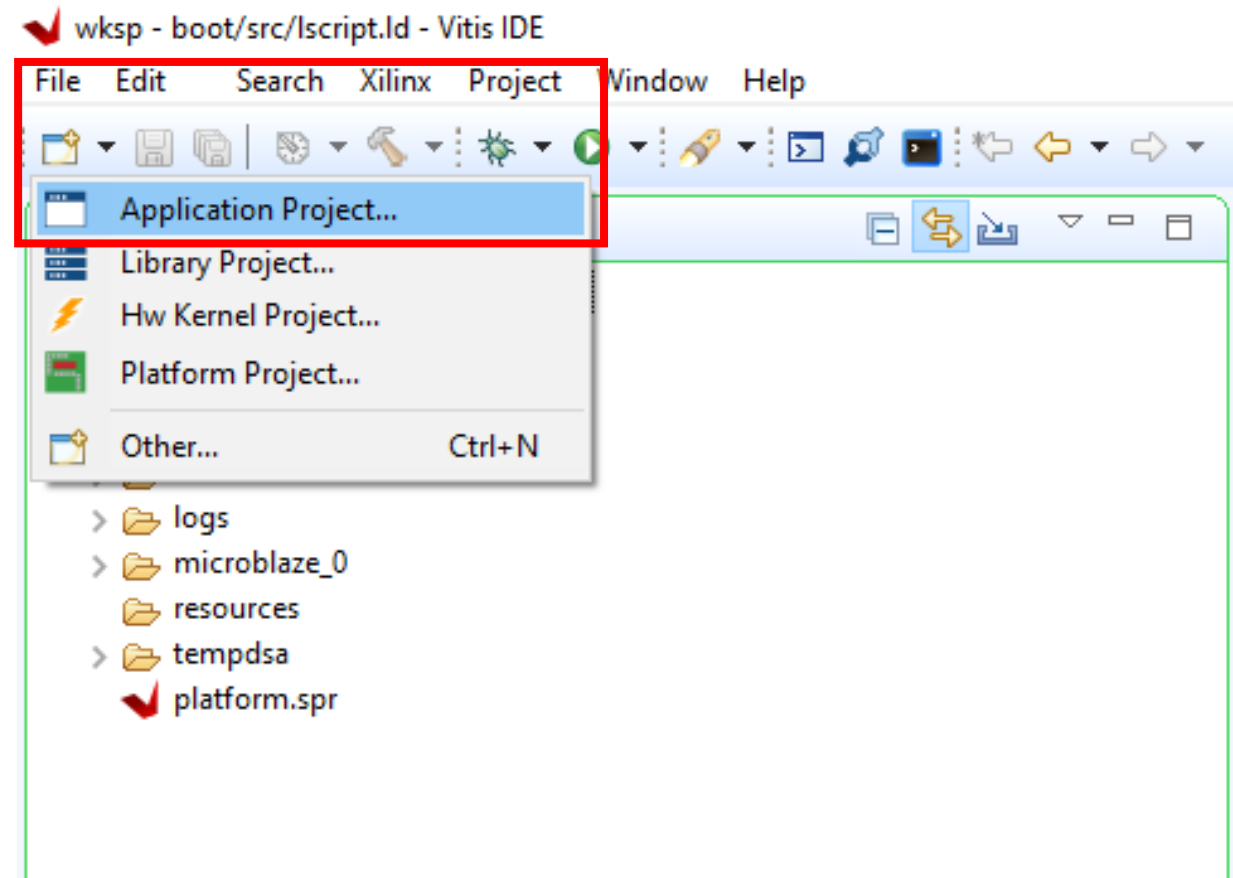
Configuration for library: xilif

Name	Value	Default	Type	Description
serial_flash_family	5	1	integer	Indicates the Serial Flash family type. Enter 1 for ATMEL. 2 for INTEL. 3 for STM (M25PXX). 4 for Winbond. 5 for Spansion/ Cypress, Micron, Macronix and ISSI. ...
serial_flash_interface	1	1	integer	Indicates the Serial Flash Interface. Enter 1 for AXI SPI. 2 for PS SPI. 3 for PS QSPI(supported families are Winbond(4), Spansion/Cypress and Micron(5)). 4 for OS...

OK Cancel

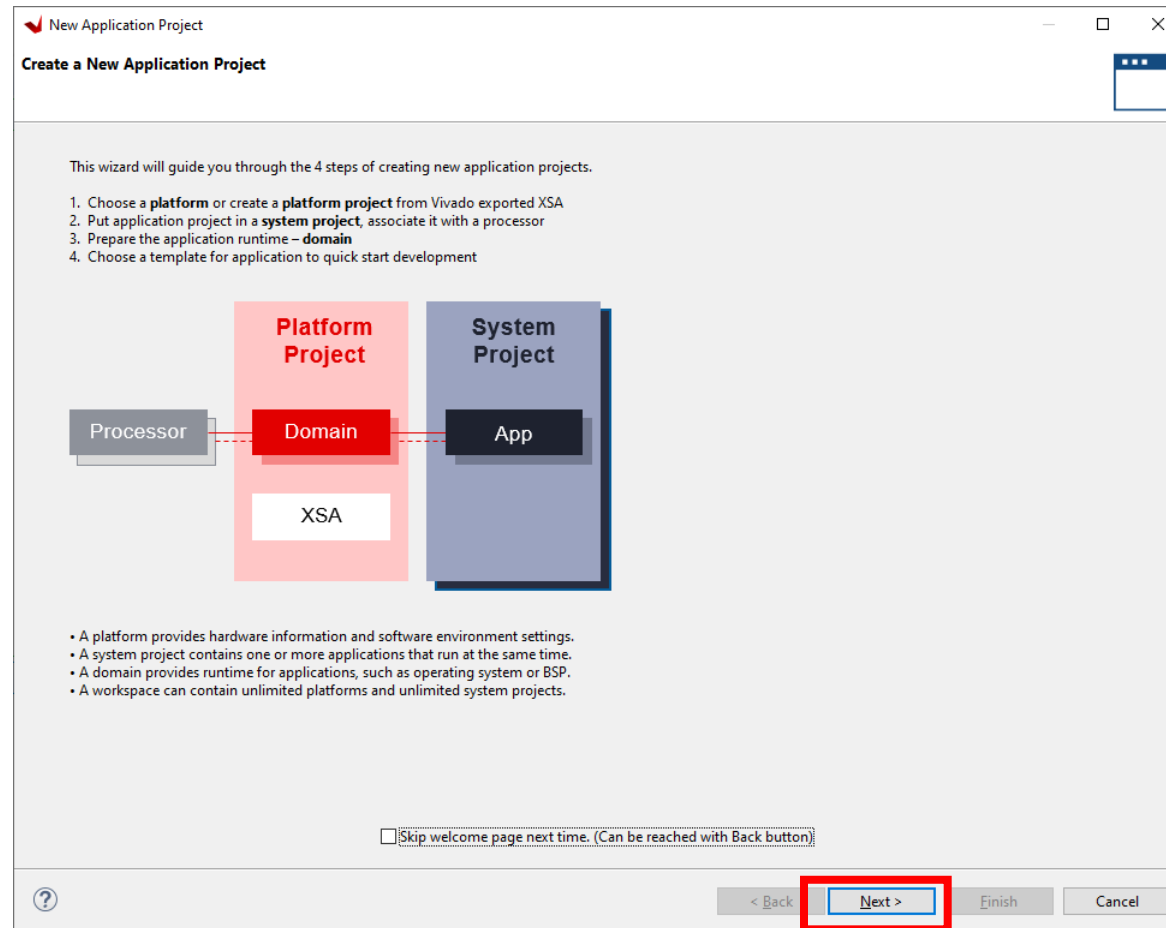
MicroBlaze Boot From QSPI

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

New Application Project

Platform
Choose a platform for your project. You can also create an application from XSA through the 'Create a new platform from hardware (XSA)' tab.

Select a platform from repository | Create a new platform from hardware (XSA)

Find:

Name	Board	Flow	Vendor	Path
 design_1_wrapper [custom]	arty-s7-50	Embedded SW Dev	xilinx	C:\Users\aptay\Dropbox\adiuvo\AUS_DOD\artyboot\wksp\desig

Platform Info

General Info

Name:

Part:

Family:

Description:

Acceleration Resources

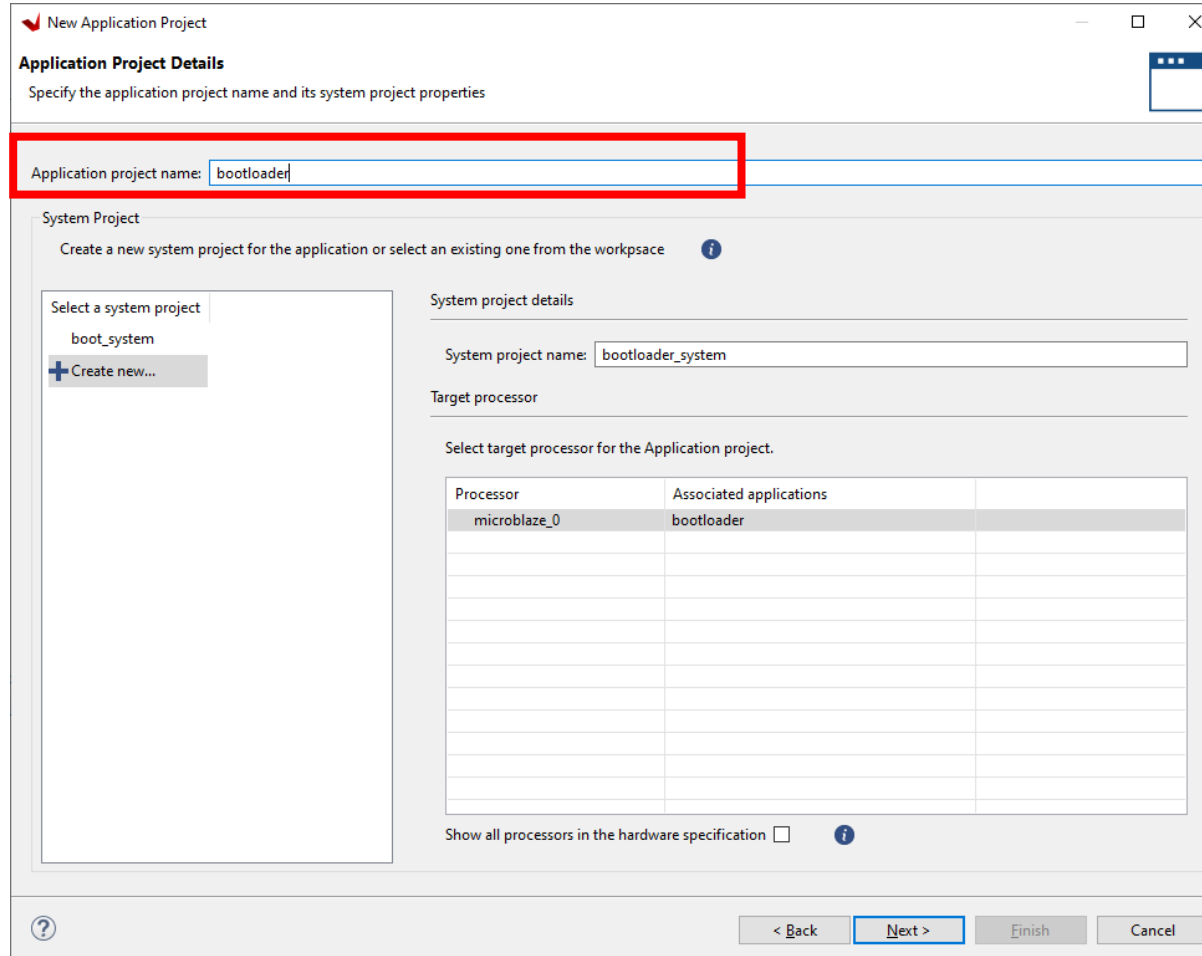
The selected platform does not have application acceleration capabilities

Domain Details

Domain name	Details
domain_microblaze_0	CPU: microblaze_00!

MicroBlaze Boot From QSPI

Name the project



New Application Project

Application Project Details
Specify the application project name and its system project properties

Application project name:

System Project
Create a new system project for the application or select an existing one from the workspace

Select a system project

- boot_system
- + Create new...

System project details

System project name:

Target processor

Select target processor for the Application project.

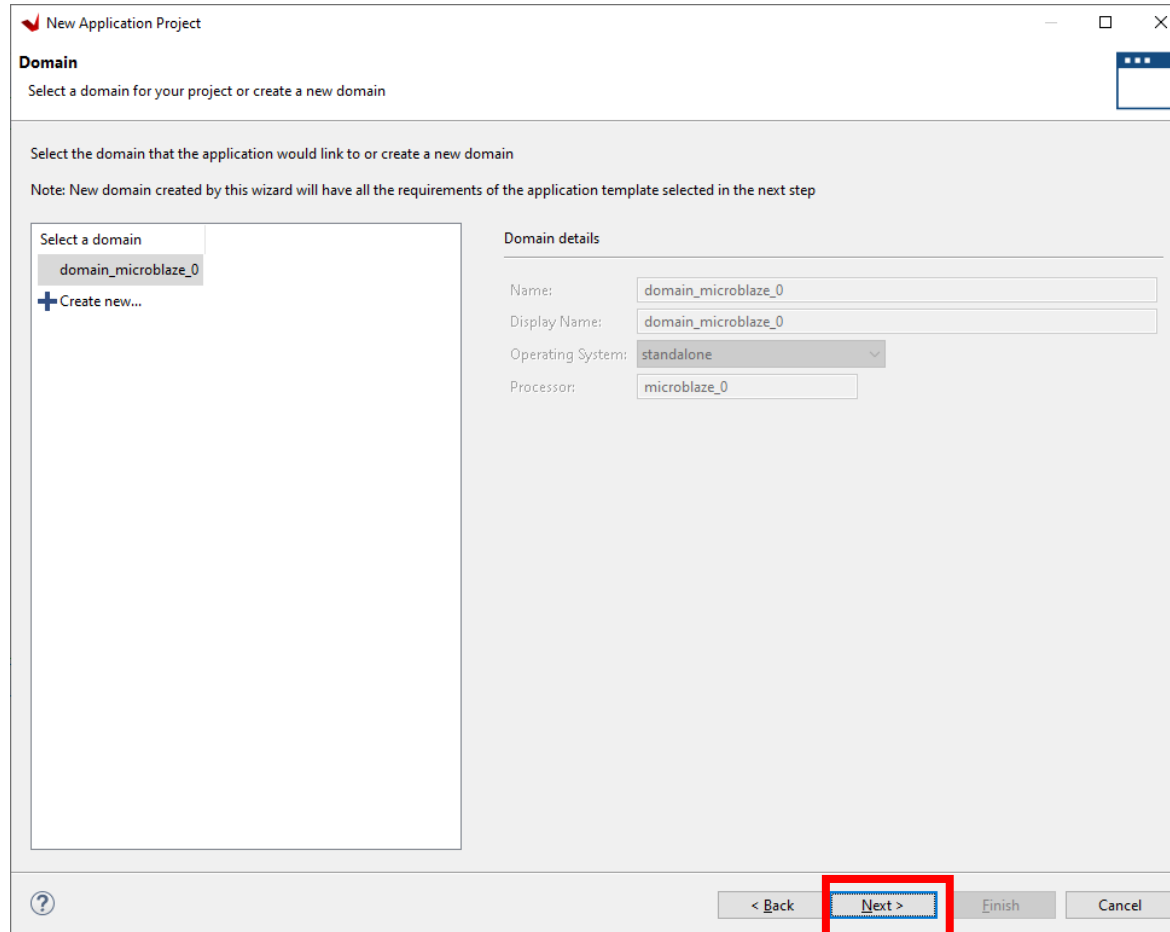
Processor	Associated applications
microblaze_0	bootloader

Show all processors in the hardware specification

< Back Next > Finish Cancel

MicroBlaze Boot From QSPI

Leave the domain unchanged



New Application Project

Domain
Select a domain for your project or create a new domain

Select the domain that the application would link to or create a new domain

Note: New domain created by this wizard will have all the requirements of the application template selected in the next step

Select a domain

- domain_microblaze_0
- + Create new...

Domain details

Name: domain_microblaze_0

Display Name: domain_microblaze_0

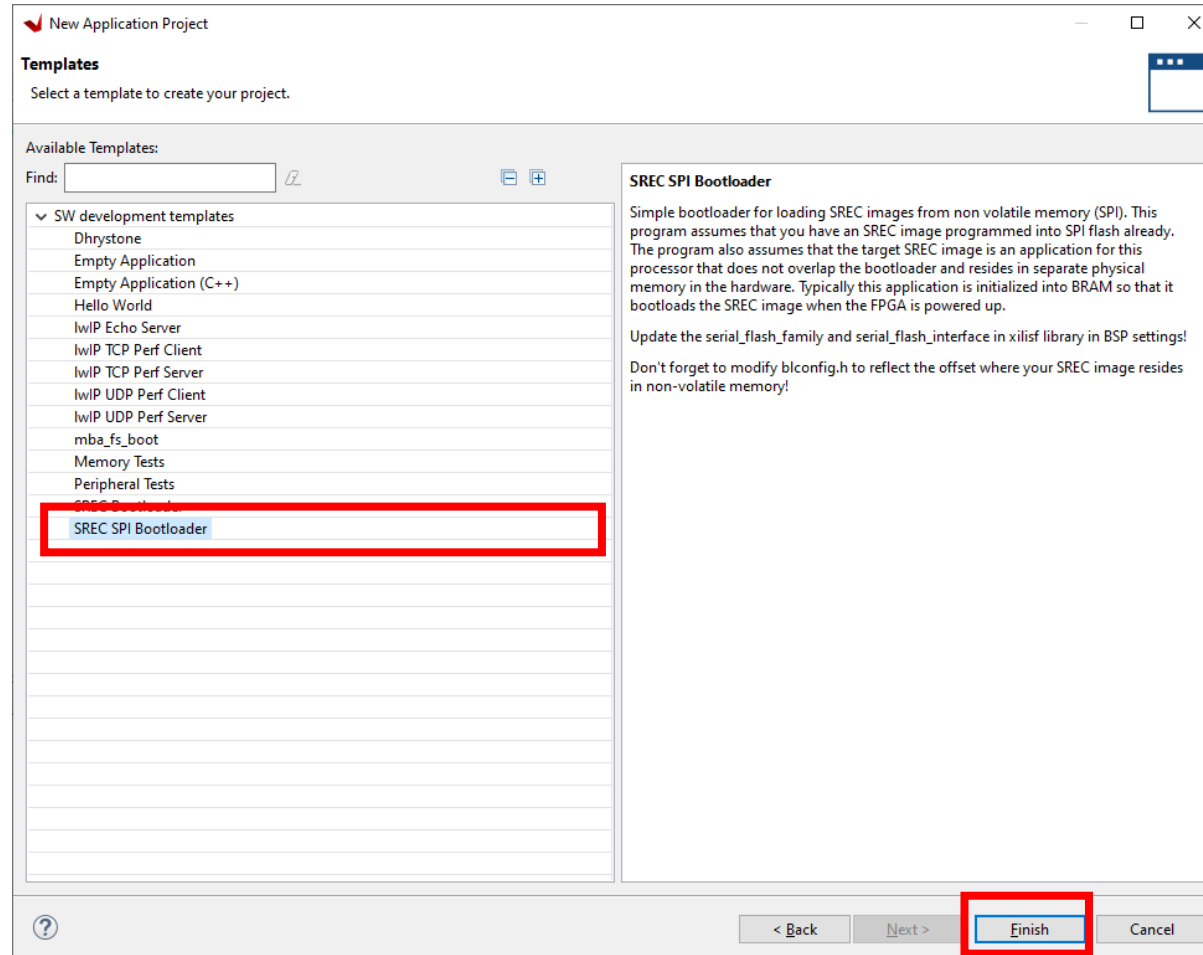
Operating System: standalone

Processor: microblaze_0

< Back **Next >** Finish Cancel

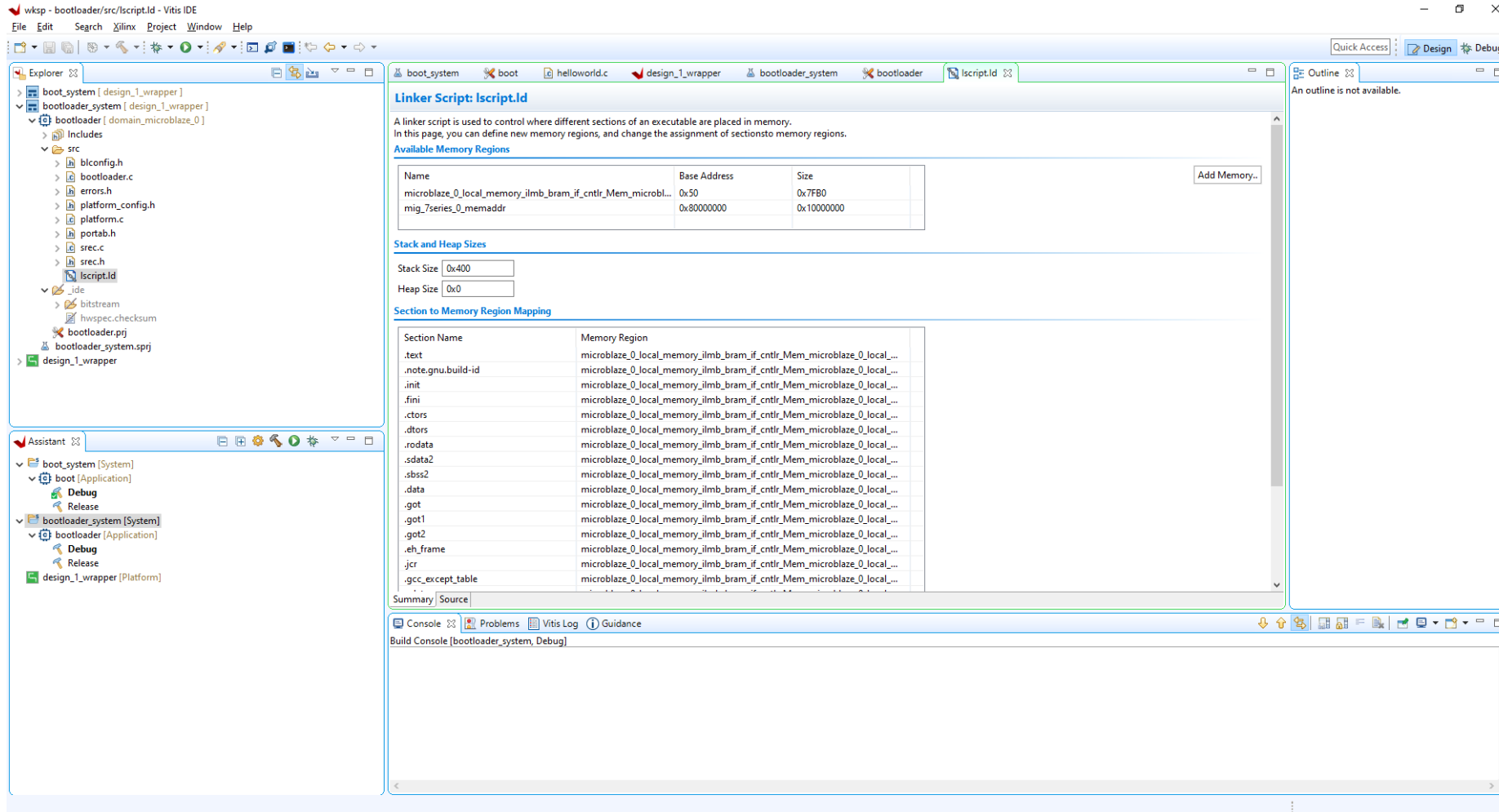
MicroBlaze Boot From QSPI

Select the SREC SPI Bootloader – click finish



MicroBlaze Boot From QSPI

Examine the SREC linker script and observe it runs from BRAM



Linker Script: Iscript.ld

A linker script is used to control where different sections of an executable are placed in memory. In this page, you can define new memory regions, and change the assignment of sections to memory regions.

Available Memory Regions

Name	Base Address	Size
microblaze_0_local_memory_ilm_bram_if_cntrl_Mem_microblaze_0...	0x50	0x7F80
mig_7series_0_memaddr	0x8000000	0x1000000

Stack and Heap Sizes

Stack Size:

Heap Size:

Section to Memory Region Mapping

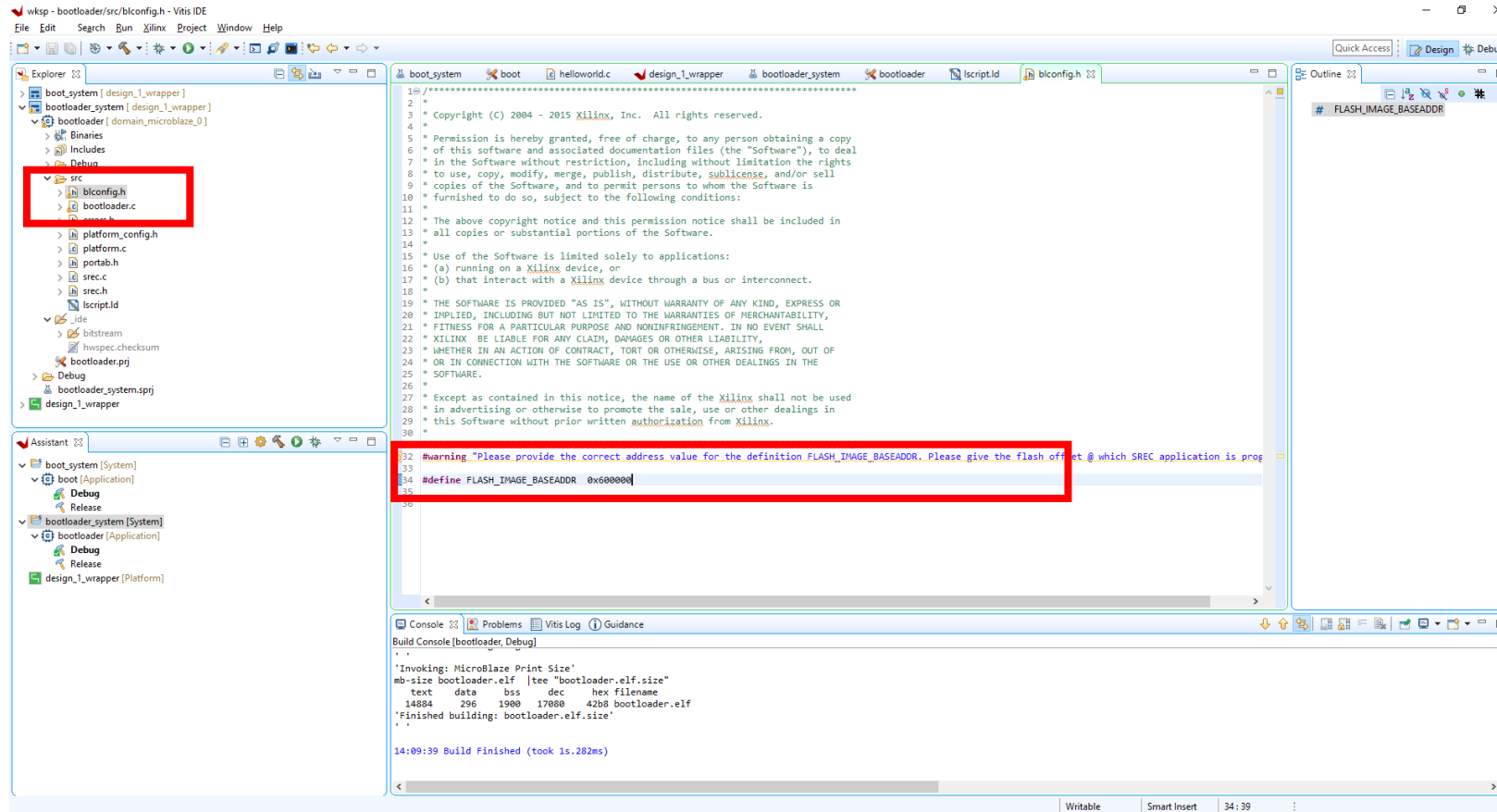
Section Name	Memory Region
.text	microblaze_0_local_memory_ilm_bram_if_cntrl_Mem_microblaze_0_local_...
.note.gnu.build-id	microblaze_0_local_memory_ilm_bram_if_cntrl_Mem_microblaze_0_local_...
.init	microblaze_0_local_memory_ilm_bram_if_cntrl_Mem_microblaze_0_local_...
.fini	microblaze_0_local_memory_ilm_bram_if_cntrl_Mem_microblaze_0_local_...
.ctors	microblaze_0_local_memory_ilm_bram_if_cntrl_Mem_microblaze_0_local_...
.dtors	microblaze_0_local_memory_ilm_bram_if_cntrl_Mem_microblaze_0_local_...
.rodata	microblaze_0_local_memory_ilm_bram_if_cntrl_Mem_microblaze_0_local_...
.sdata2	microblaze_0_local_memory_ilm_bram_if_cntrl_Mem_microblaze_0_local_...
.sbss2	microblaze_0_local_memory_ilm_bram_if_cntrl_Mem_microblaze_0_local_...
.data	microblaze_0_local_memory_ilm_bram_if_cntrl_Mem_microblaze_0_local_...
.got	microblaze_0_local_memory_ilm_bram_if_cntrl_Mem_microblaze_0_local_...
.got1	microblaze_0_local_memory_ilm_bram_if_cntrl_Mem_microblaze_0_local_...
.got2	microblaze_0_local_memory_ilm_bram_if_cntrl_Mem_microblaze_0_local_...
.eh_frame	microblaze_0_local_memory_ilm_bram_if_cntrl_Mem_microblaze_0_local_...
.jcr	microblaze_0_local_memory_ilm_bram_if_cntrl_Mem_microblaze_0_local_...
.gcc_except_table	microblaze_0_local_memory_ilm_bram_if_cntrl_Mem_microblaze_0_local_...

Summary | Source

Console: Build Console [bootloader_system, Debug]

MicroBlaze Boot From QSPI

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



The screenshot shows the Vitis IDE interface. The Explorer window on the left shows the project structure, with the `blconfig.h` file highlighted in the `src` directory. The main editor window displays the contents of `blconfig.h`, which includes a copyright notice and a warning message. A red box highlights the warning message and the definition of `FLASH_IMAGE_BASEADDR`.

```

1 //
2 *
3 * Copyright (C) 2004 - 2015 Xilinx, Inc. All rights reserved.
4 *
5 * Permission is hereby granted, free of charge, to any person obtaining a copy
6 * of this software and associated documentation files (the "Software"), to deal
7 * in the Software without restriction, including without limitation the rights
8 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
9 * copies of the Software, and to permit persons to whom the Software is
10 * furnished to do so, subject to the following conditions:
11 *
12 * The above copyright notice and this permission notice shall be included in
13 * all copies or substantial portions of the Software.
14 *
15 * Use of the Software is limited solely to applications:
16 * (a) running on a Xilinx device, or
17 * (b) that interact with a Xilinx device through a bus or interconnect.
18 *
19 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
20 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
21 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
22 * XILINX BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
23 * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF
24 * OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
25 * SOFTWARE.
26 *
27 * Except as contained in this notice, the name of the Xilinx shall not be used
28 * in advertising or otherwise to promote the sale, use or other dealings in
29 * this Software without prior written authorization from Xilinx.
30 *
31
32 #warning "Please provide the correct address value for the definition FLASH_IMAGE_BASEADDR. Please give the flash offset @ which SREC application is prog
33
34 #define FLASH_IMAGE_BASEADDR 0x600000
35
36

```

The Assistant window on the left shows the project hierarchy, including the `bootloader_system` application. The Console window at the bottom shows the build output for the `bootloader` application, indicating that the build was successful.

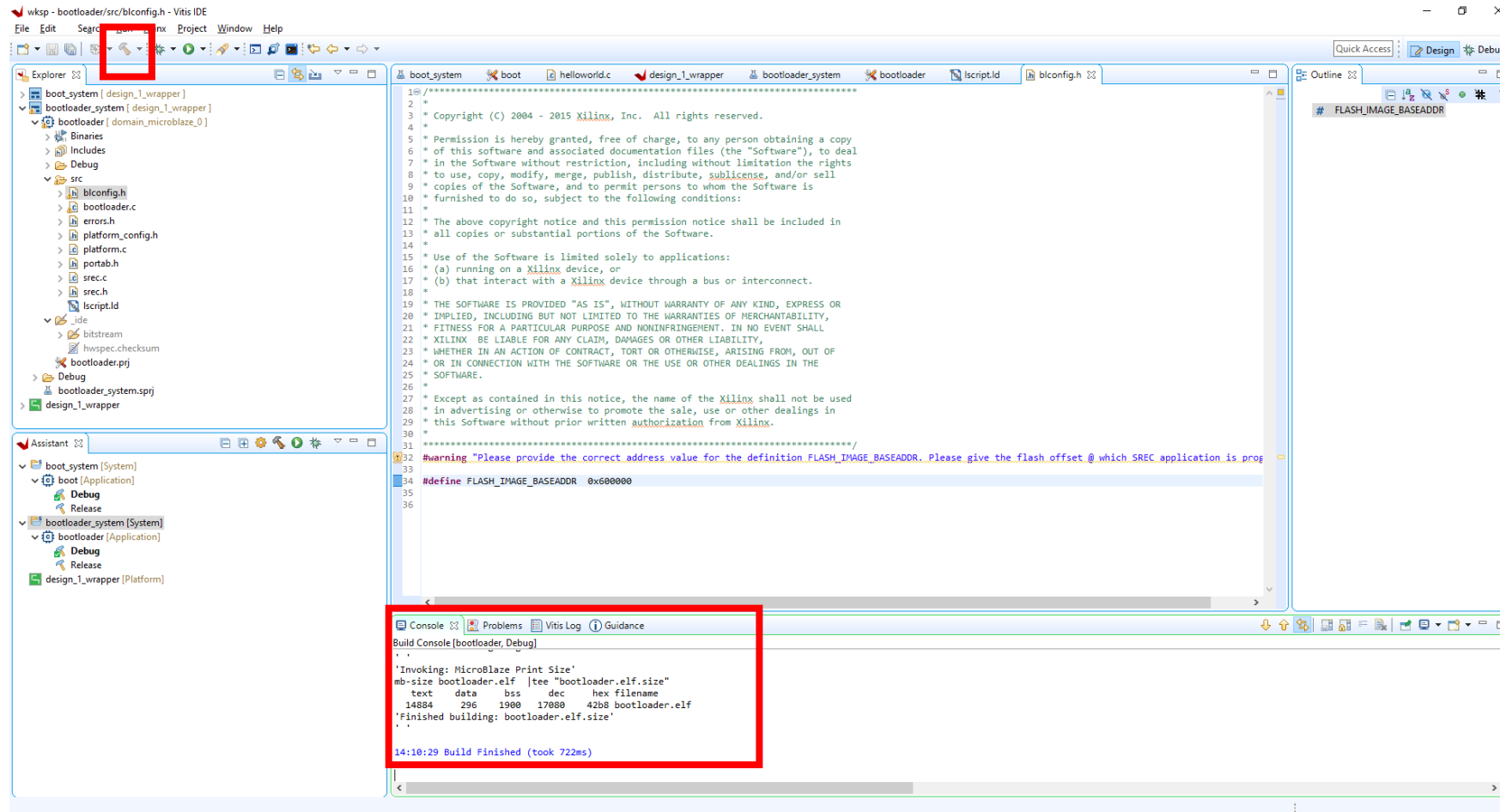
```

Build Console [bootloader, Debug]
..
Invoking: MicroBlaze Print Size
mb-size bootloader.elf |tee "bootloader.elf.size"
text data bss dec hex filename
14884 296 1900 17080 4208 bootloader.elf
Finished building: bootloader.elf.size
..
14:09:39 Build Finished (took 1s.282ms)

```

MicroBlaze Boot From QSPI

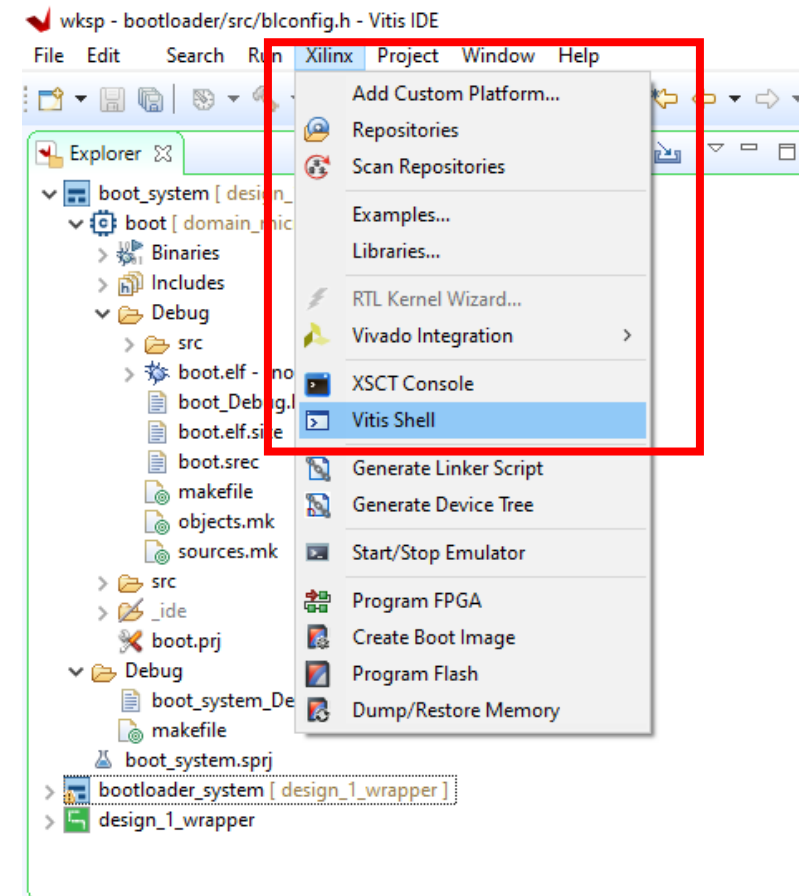
Build the design – we now have ELF for the bootloader and hello world



MicroBlaze Boot From QSPI

From the Xilinx Menu open the Vitis Shell.

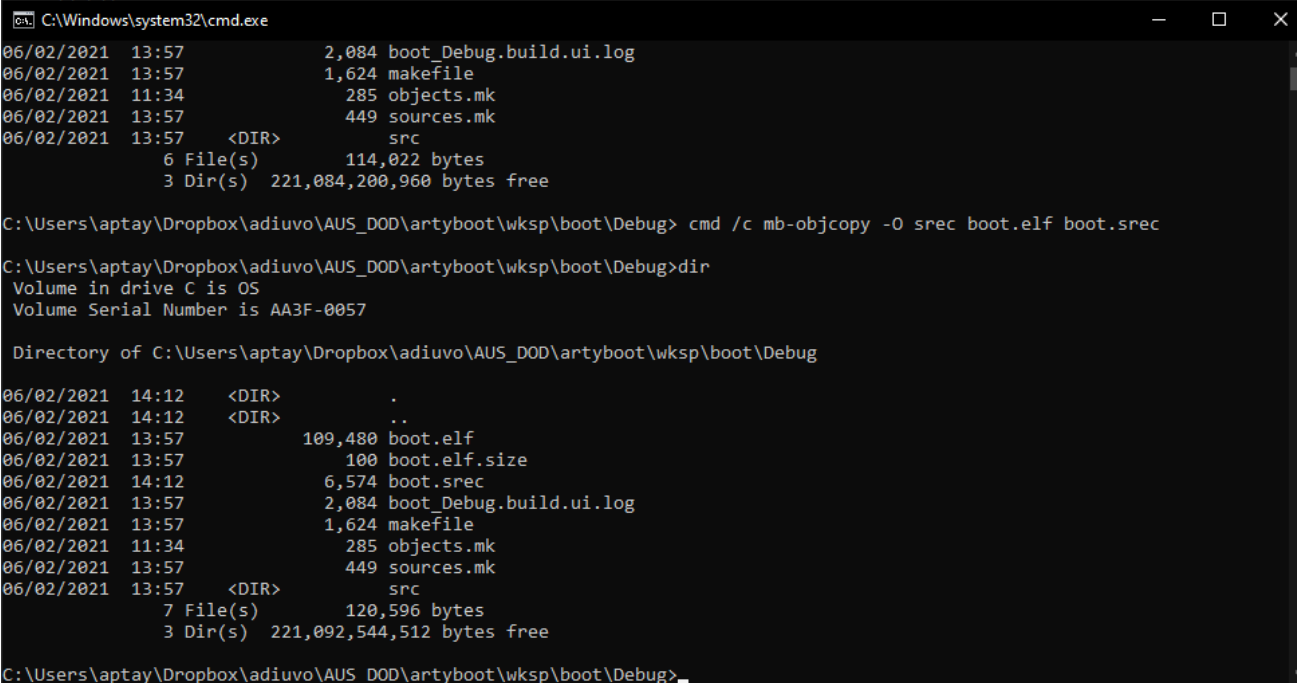
This will allow us to convert the application (hello world) elf into S Record format for loading into QSPI memory



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
```



```
C:\Windows\system32\cmd.exe
06/02/2021 13:57      2,084 boot_Debug.build.ui.log
06/02/2021 13:57      1,624 makefile
06/02/2021 11:34      285 objects.mk
06/02/2021 13:57      449 sources.mk
06/02/2021 13:57    <DIR>      src
          6 File(s)      114,022 bytes
          3 Dir(s)    221,084,200,960 bytes free

C:\Users\aptay\Dropbox\adiuvo\AUS_DOD\artyboot\wksp\boot\Debug> cmd /c mb-objcopy -O srec boot.elf boot.srec

C:\Users\aptay\Dropbox\adiuvo\AUS_DOD\artyboot\wksp\boot\Debug>dir
Volume in drive C is OS
Volume Serial Number is AA3F-0057

Directory of C:\Users\aptay\Dropbox\adiuvo\AUS_DOD\artyboot\wksp\boot\Debug

06/02/2021 14:12    <DIR>      .
06/02/2021 14:12    <DIR>      ..
06/02/2021 13:57     109,480 boot.elf
06/02/2021 13:57       100 boot.elf.size
06/02/2021 14:12     6,574 boot.srec
06/02/2021 13:57      2,084 boot_Debug.build.ui.log
06/02/2021 13:57      1,624 makefile
06/02/2021 11:34      285 objects.mk
06/02/2021 13:57      449 sources.mk
06/02/2021 13:57    <DIR>      src
          7 File(s)      120,596 bytes
          3 Dir(s)    221,092,544,512 bytes free

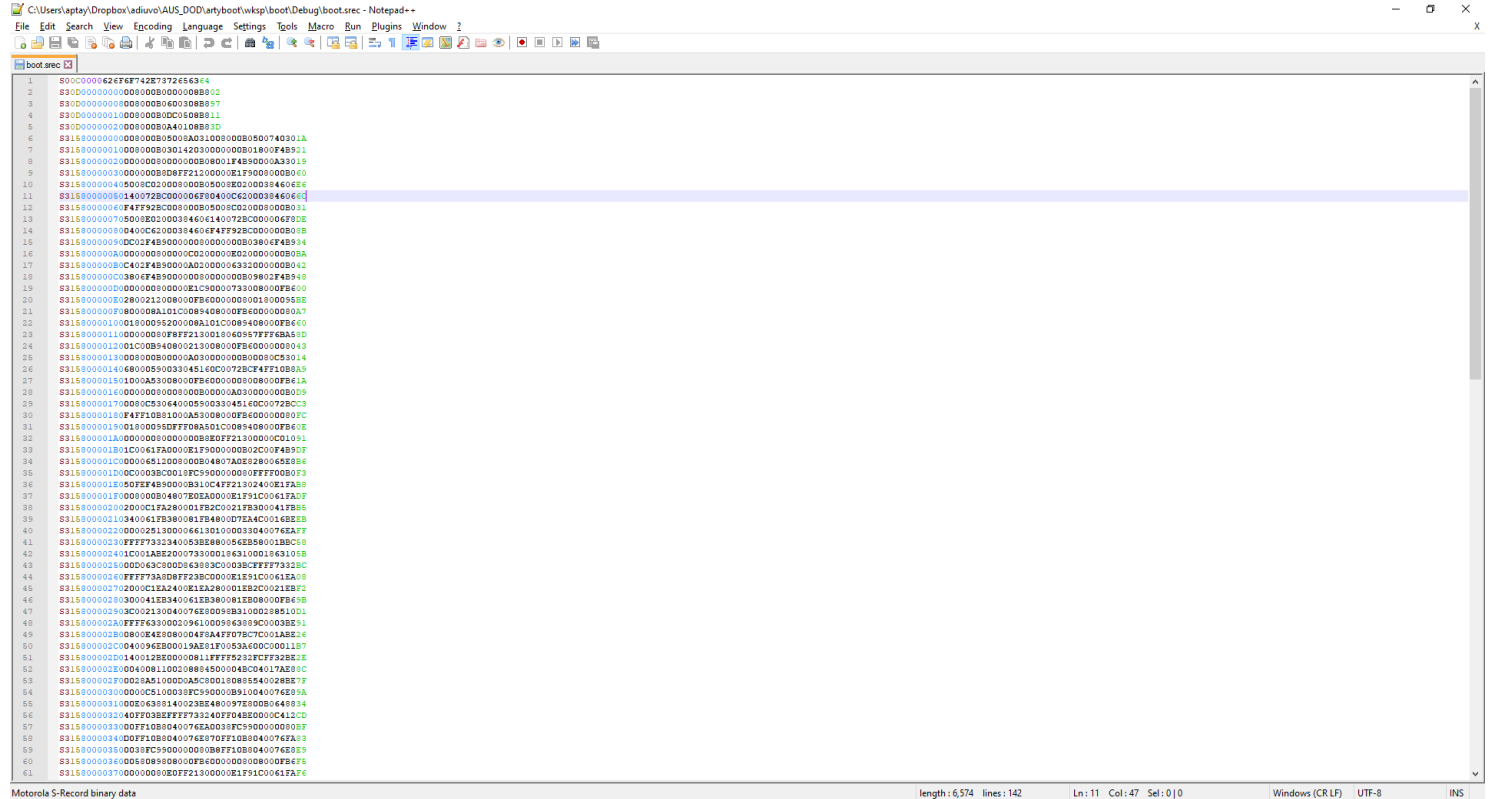
C:\Users\aptay\Dropbox\adiuvo\AUS_DOD\artyboot\wksp\boot\Debug>
```

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



```

1  S00000062E6F742E7372E563E4
2  S3D0000000000000000000000
3  S3D0000000000000000000000
4  S3D0000001000000000000000
5  S3D00000020000000A401000000
6  S31E0000000000000000000000
7  S31E0000011000000000000000
8  S31E0000021000000000000000
9  S31E0000031000000000000000
10 S31E0000041000000000000000
11 S31E0000051400720C00000000
12 S31E000006074FF92BC00000000
13 S31E00000705000E02000394604
14 S31E00000804000C10003946044
15 S31E0000090DC02F4B900000000
16 S31E00000A0000000000000000
17 S31E00000B0C402F4B90000A0200
18 S31E00000C08074B900000000000
19 S31E00000D0000000000000000
20 S31E00000E0200021200000000
21 S31E00000F04000A101C00000000
22 S31E0000100100005200000000
23 S31E0000110000000000000000
24 S31E00001201C00B94000221300
25 S31E0000130000000000000000
26 S31E00001406000050030451600
27 S31E0000150100A5300000000000
28 S31E0000160000000000000000
29 S31E00001700000C506400000000
30 S31E000018074FF1080100A5300
31 S31E0000190100005000000000
32 S31E00001A0000000000000000
33 S31E00001B1C0061FA0000E1F00
34 S31E00001C0000651200000000
35 S31E00001D0C0103BC0010FC900
36 S31E00001E0000000000000000
37 S31E00001F0000000048070E0A
38 S31E0000200200C1FA20001FB2C
39 S31E000021040041FB90001FB40
40 S31E0000220000025130000661
41 S31E0000230FFF7332340053B
42 S31E0000240000000000000000
43 S31E0000250000000000000000
44 S31E0000260FFF73A000000000E
45 S31E00002702000C1A24000000E
46 S31E0000280004A00000000000
47 S31E0000290C02130040076E00
48 S31E00002A0FFF6330002096100
49 S31E00002B0000000000000000
50 S31E00002C0400000000000000
51 S31E00002D0140012E00000811
52 S31E00002E0004008110010000
53 S31E00002F0003A51000000000
54 S31E0000300001000000000000
55 S31E000031000E6300140023E4
56 S31E00003240FF03E000000000
57 S31E00003300FF108040076E00
58 S31E00003400FF108040076E00
59 S31E0000350030FC5000000000
60 S31E0000360050080000000000
61 S31E0000370000000000000000

```



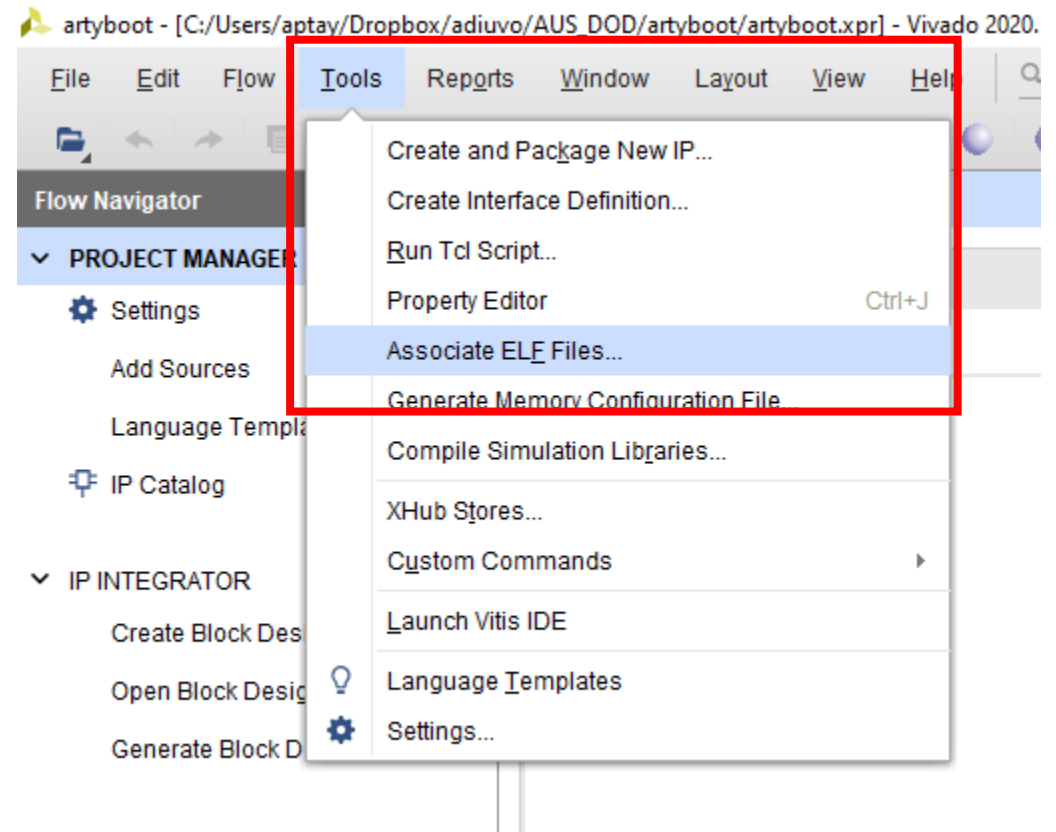
Creating the Programming Files

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



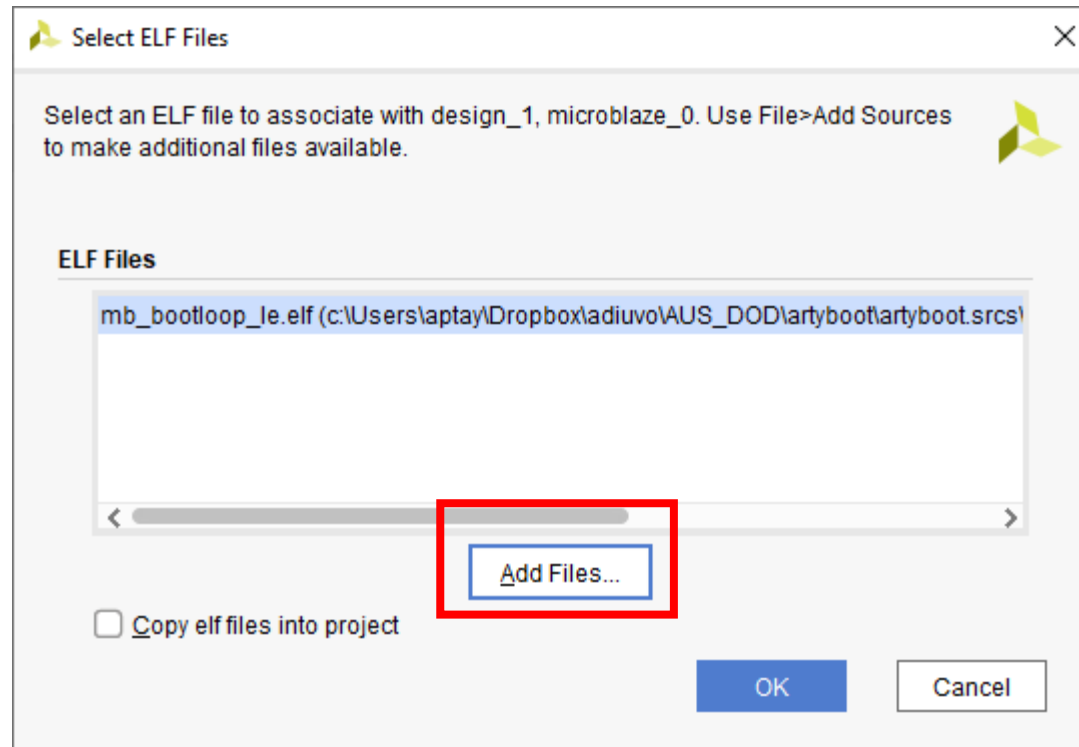
MicroBlaze Boot From QSPI

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



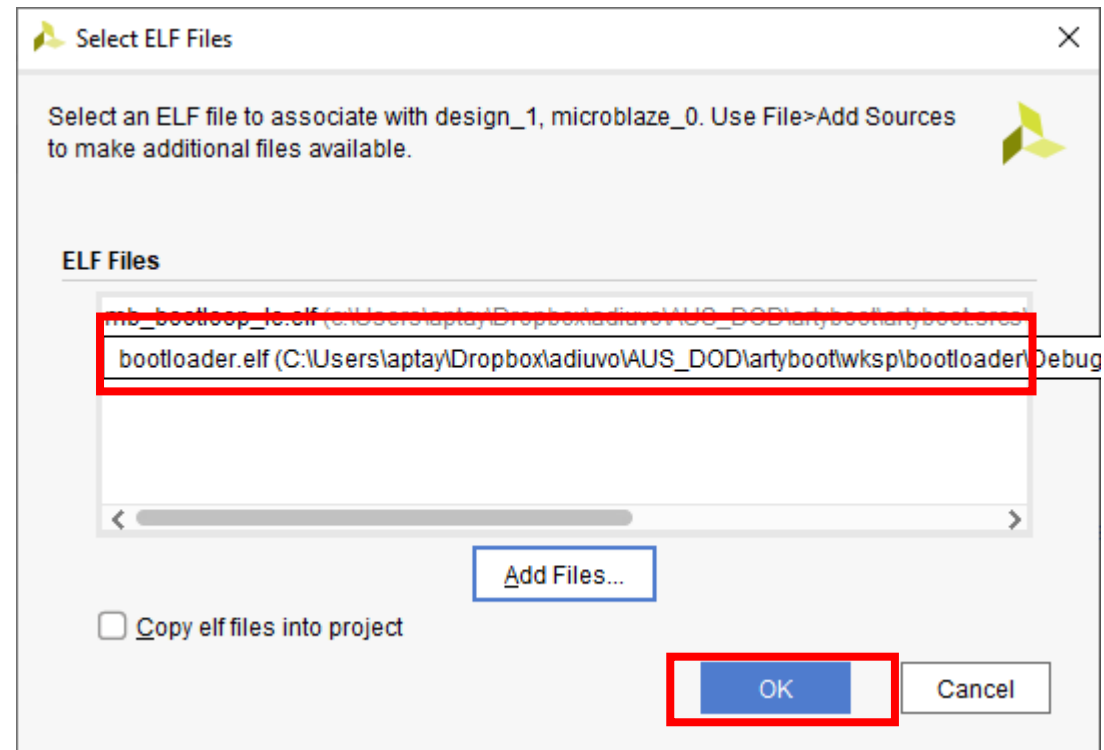
MicroBlaze Boot From QSPI

Click on Add Files



MicroBlaze Boot From QSPI

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



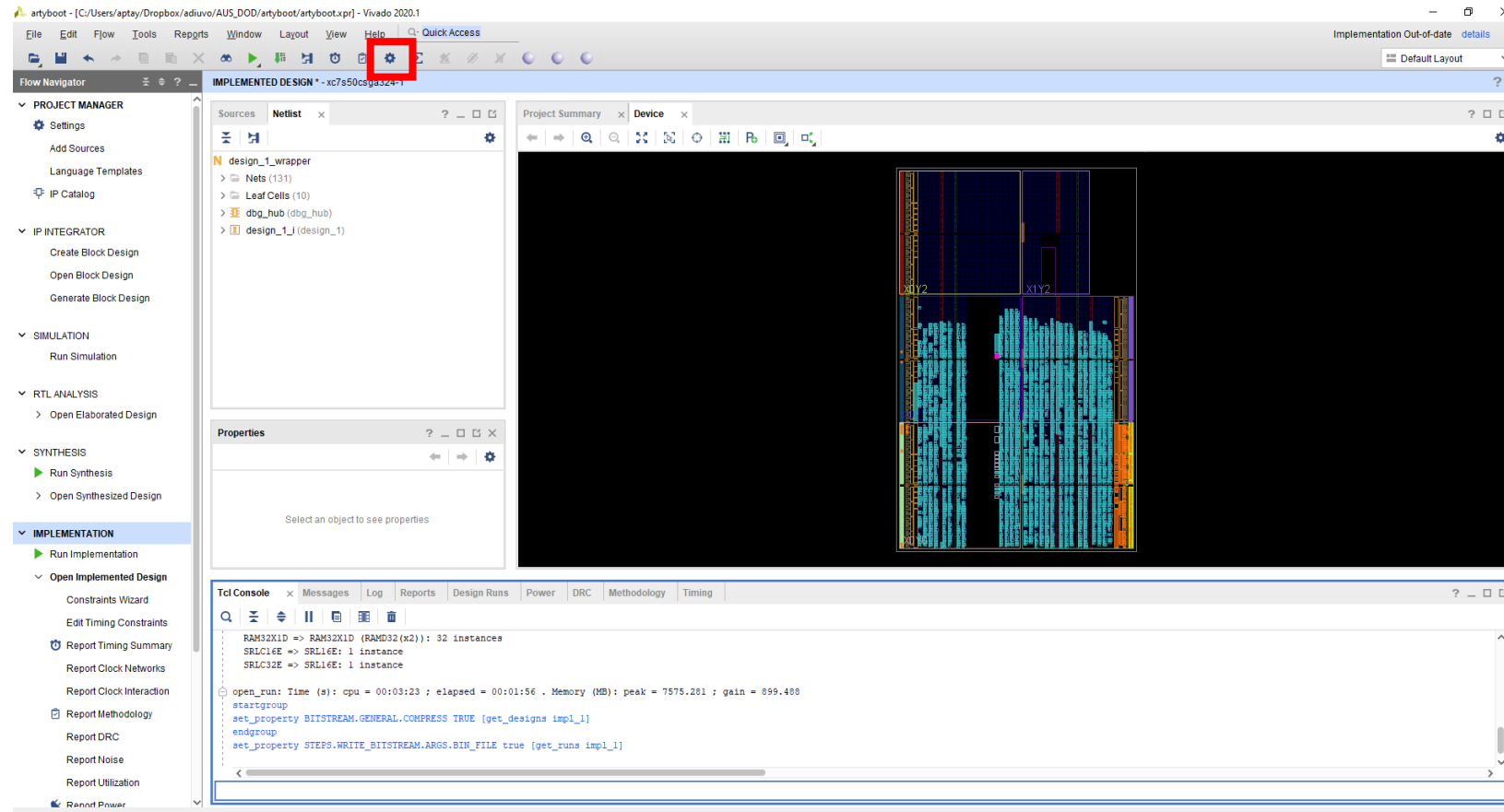
MicroBlaze Boot From QSPI

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



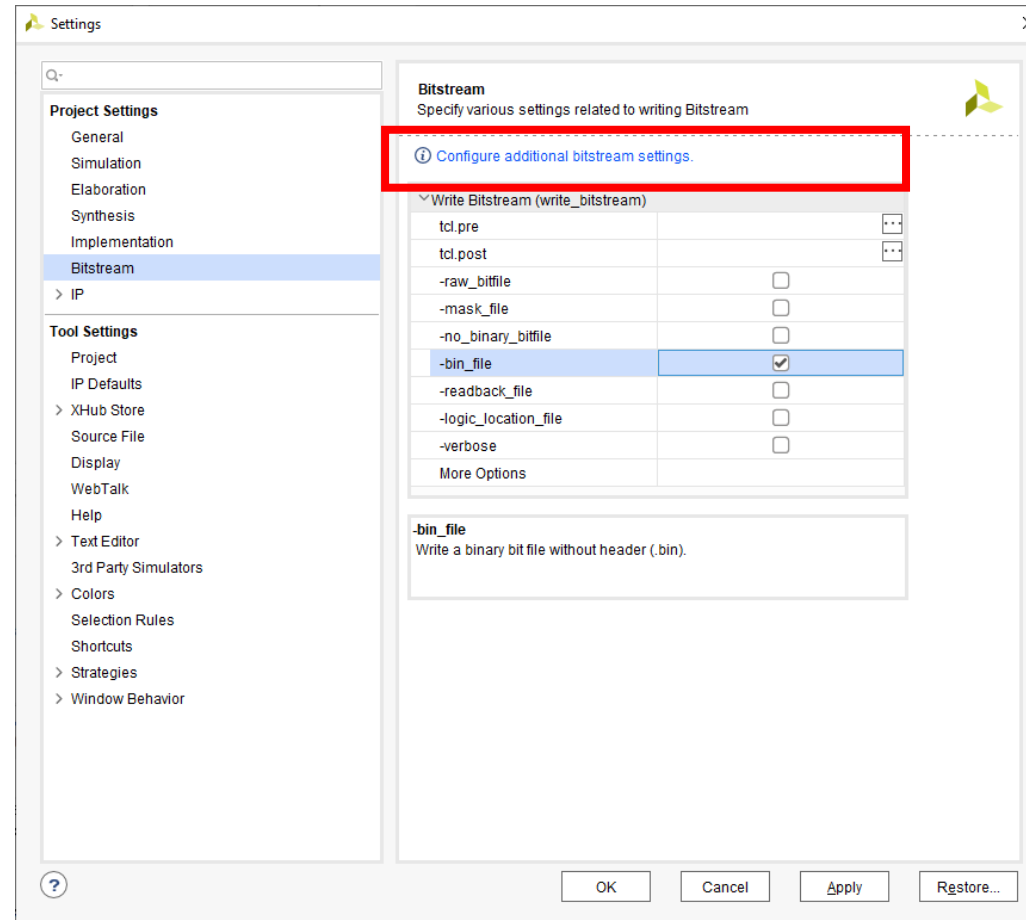
MicroBlaze Boot From QSPI

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



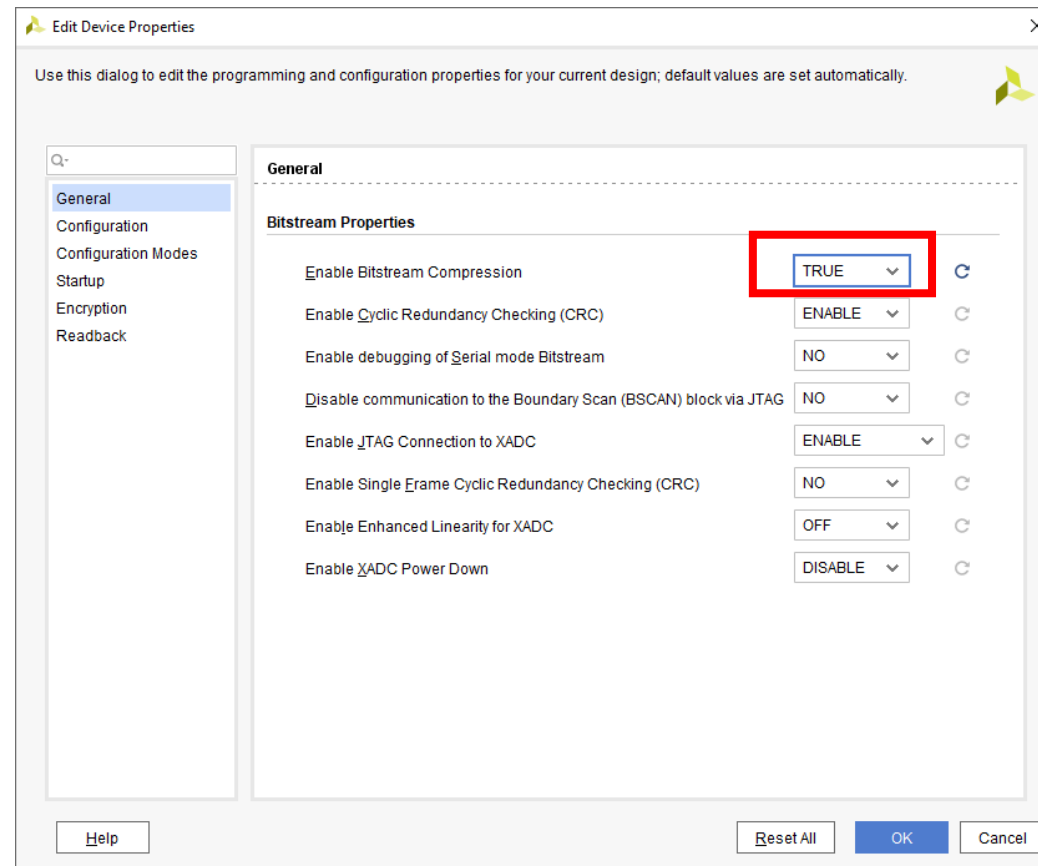
MicroBlaze Boot From QSPI

Select the configure additional setting



MicroBlaze Boot From QSPI

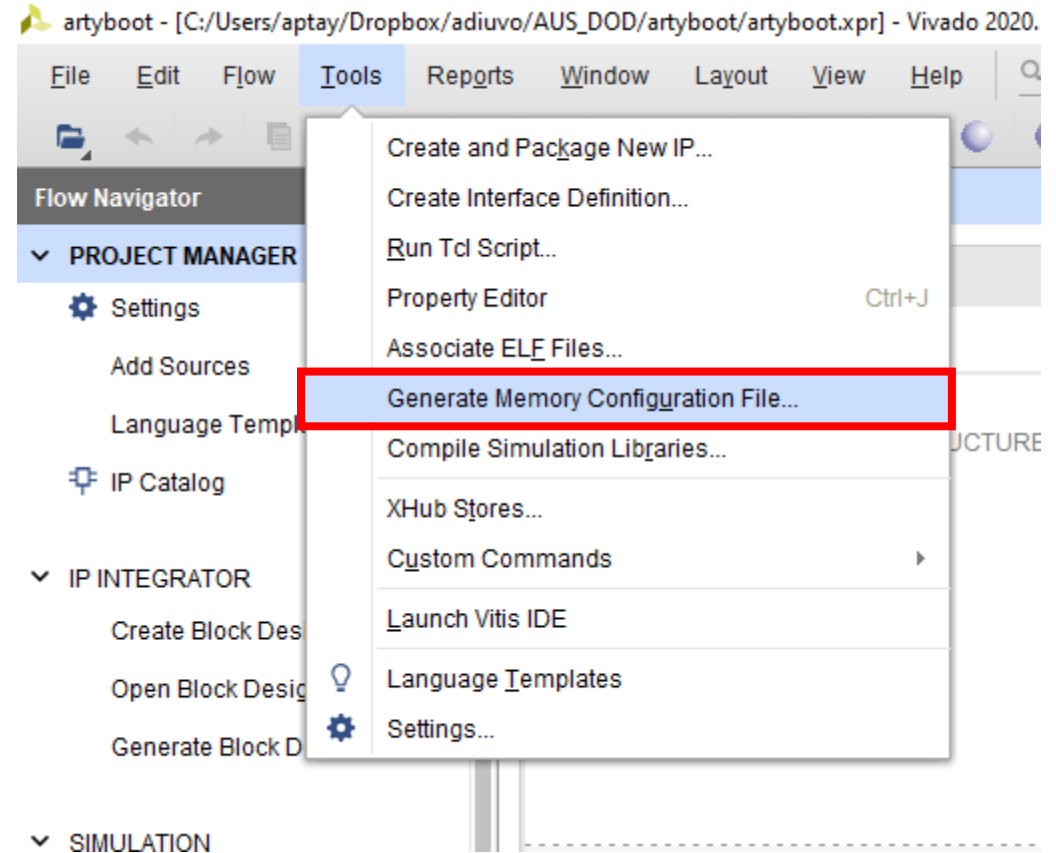
Select bitstream compression – Close the implementation view and re-generate the bit stream



MicroBlaze Boot From QSPI

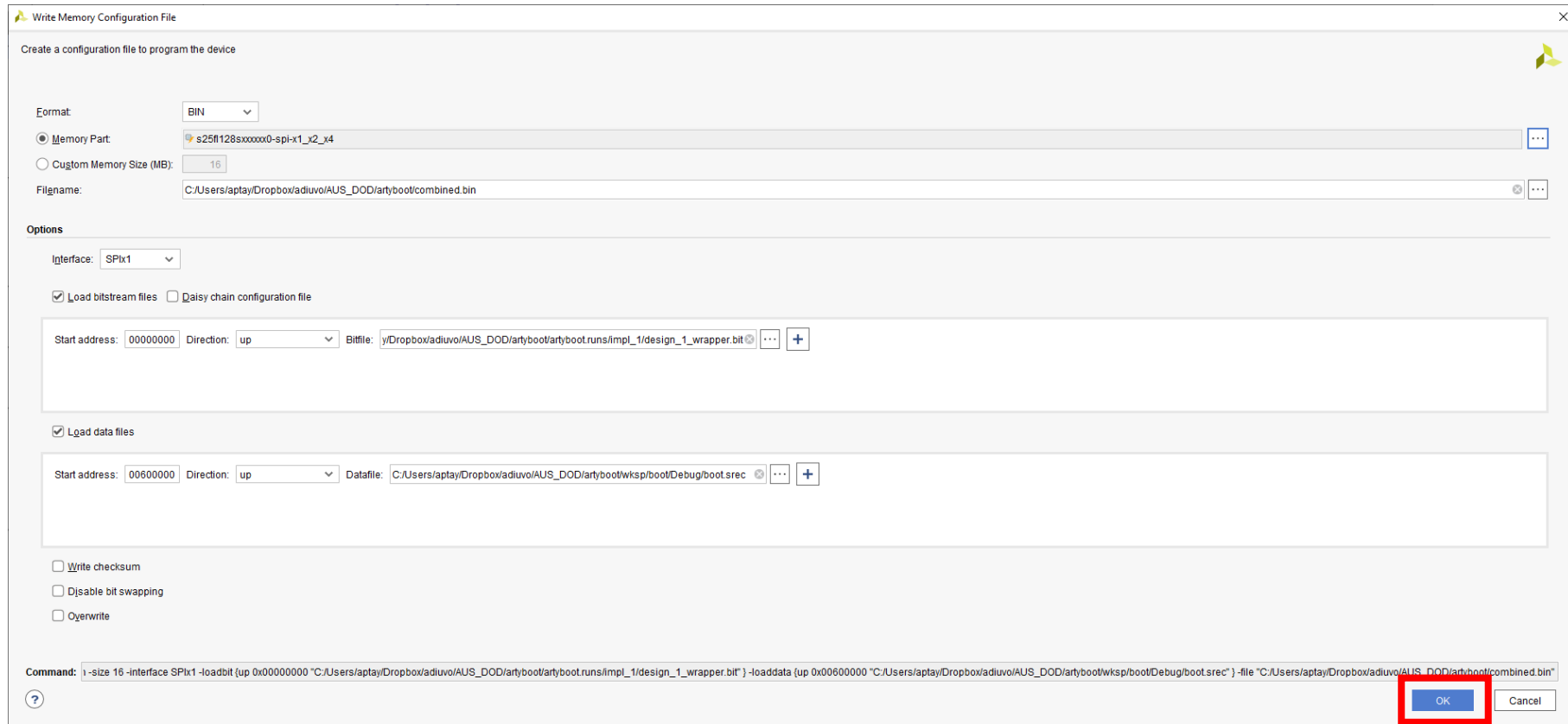
Now we need to create a bin file which contains the bitstream with the bootloader inbuilt and the SREC application

From the Tools menu select Generate Memory Configuration File



MicroBlaze Boot From QSPI

Select the Format as BIN, the memory part (S25FL128sxxxxx0-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



Write Memory Configuration File

Create a configuration file to program the device

Format:

Memory Part:

Custom Memory Size (MB):

Filename:

Options

Interface:

Load bitstream files Daisy chain configuration file

Start address: Direction: Bitfile:

Load data files

Start address: Direction: Datafile:

Write checksum

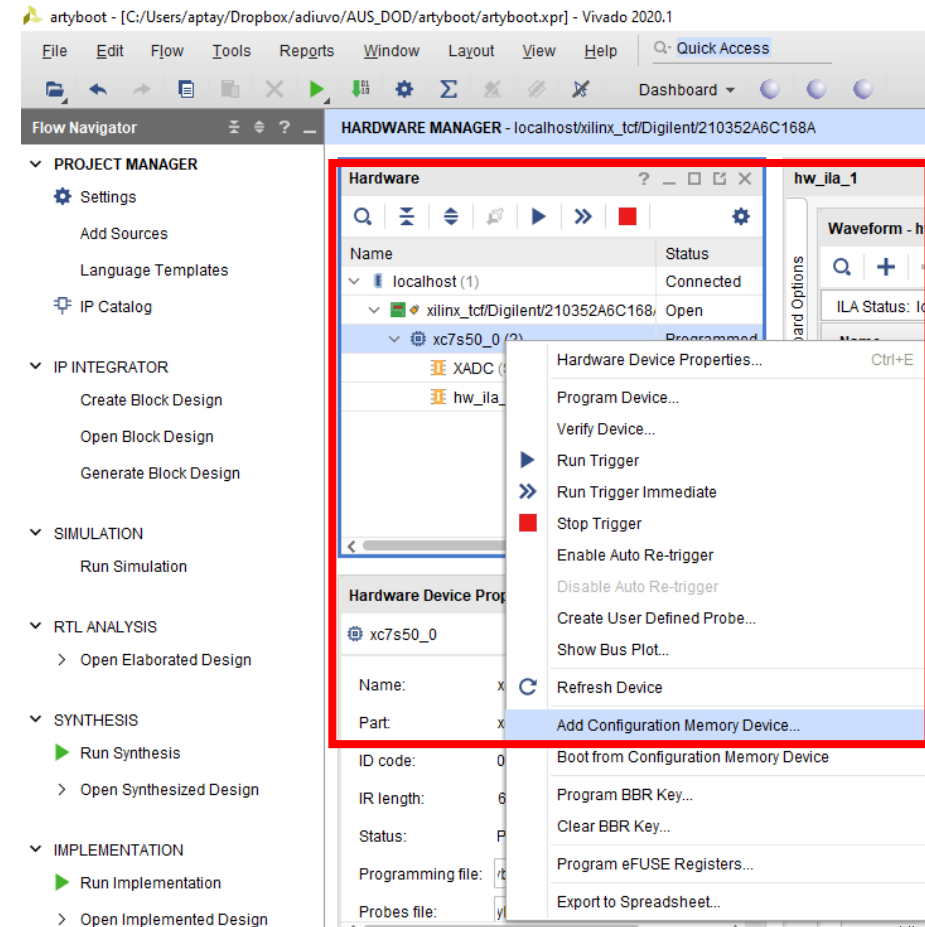
Disable bit swapping

Overwrite

Command:

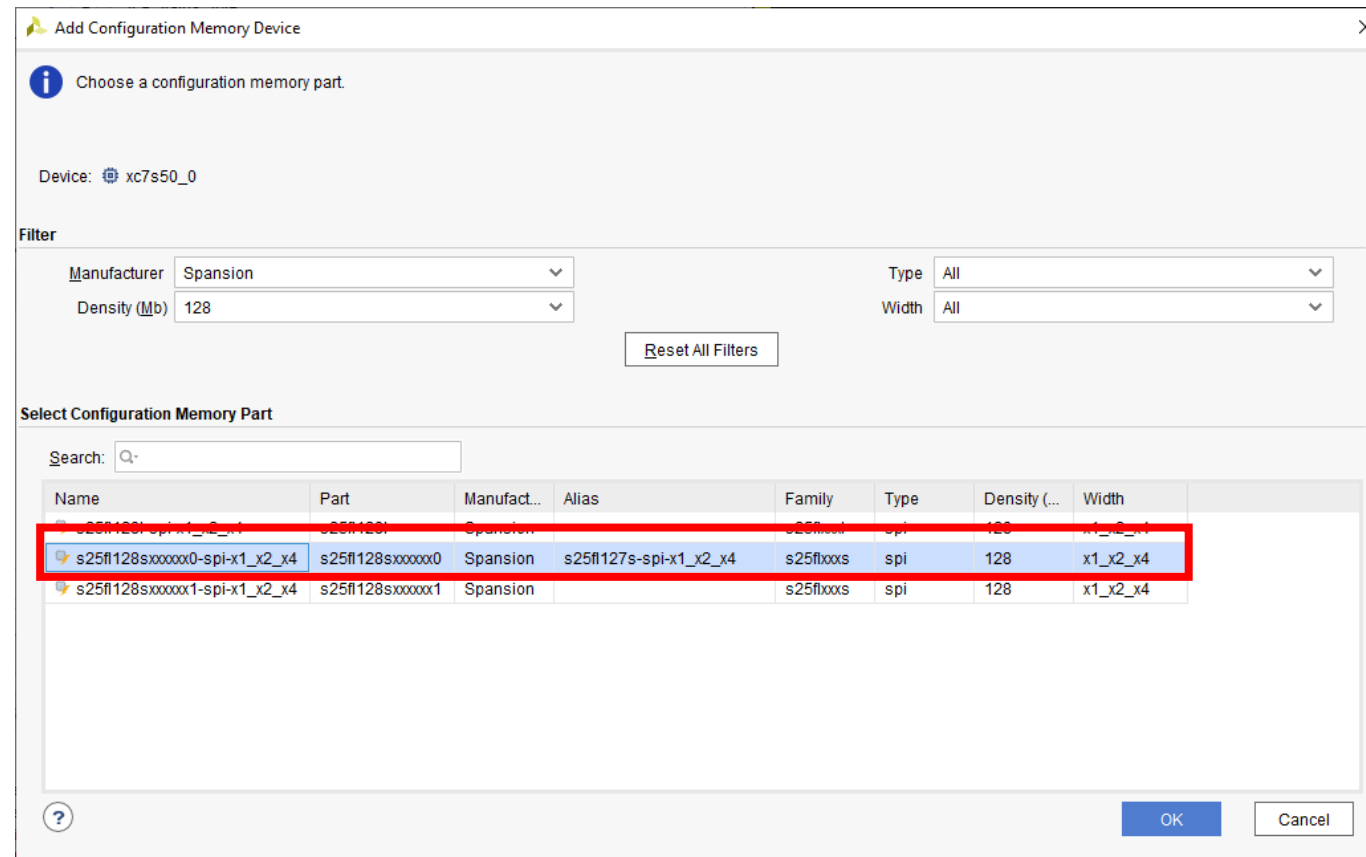
MicroBlaze Boot From QSPI

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



MicroBlaze Boot From QSPI

Select the Spansion S25FL128sxxxxx0-SPI-X1_X2_X4

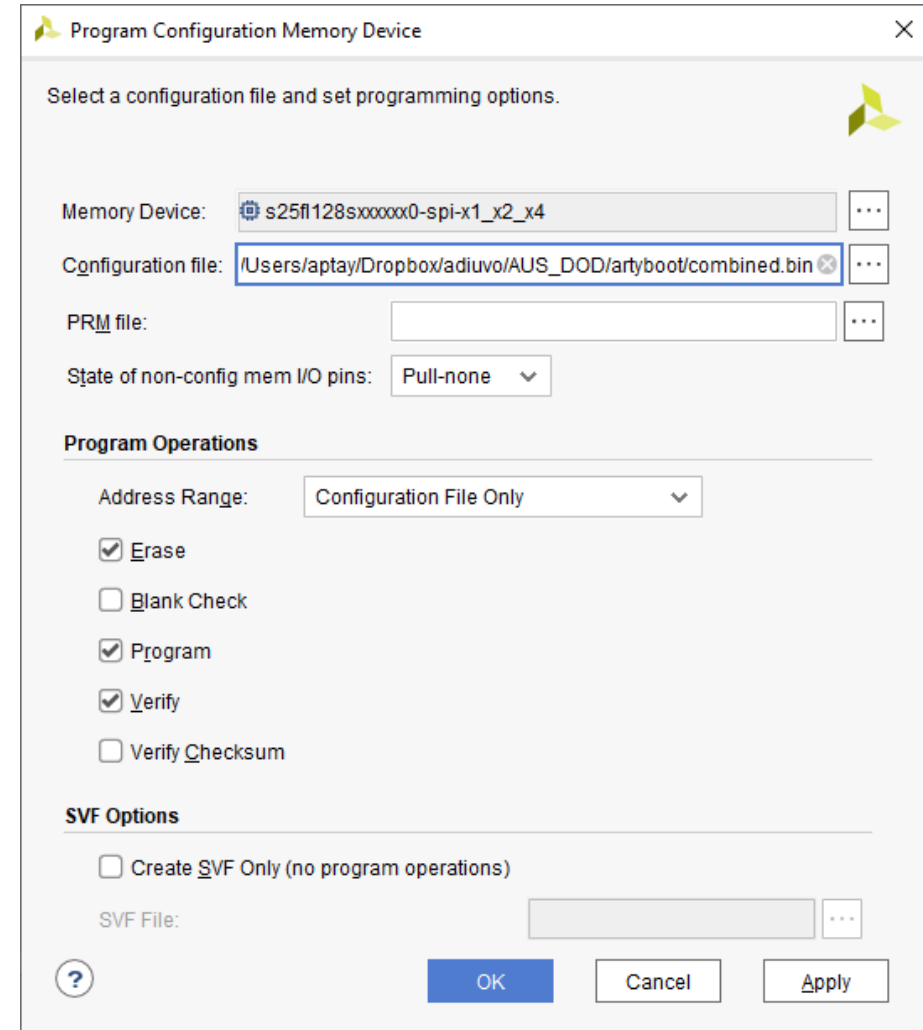


MicroBlaze Boot From QSPI

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.

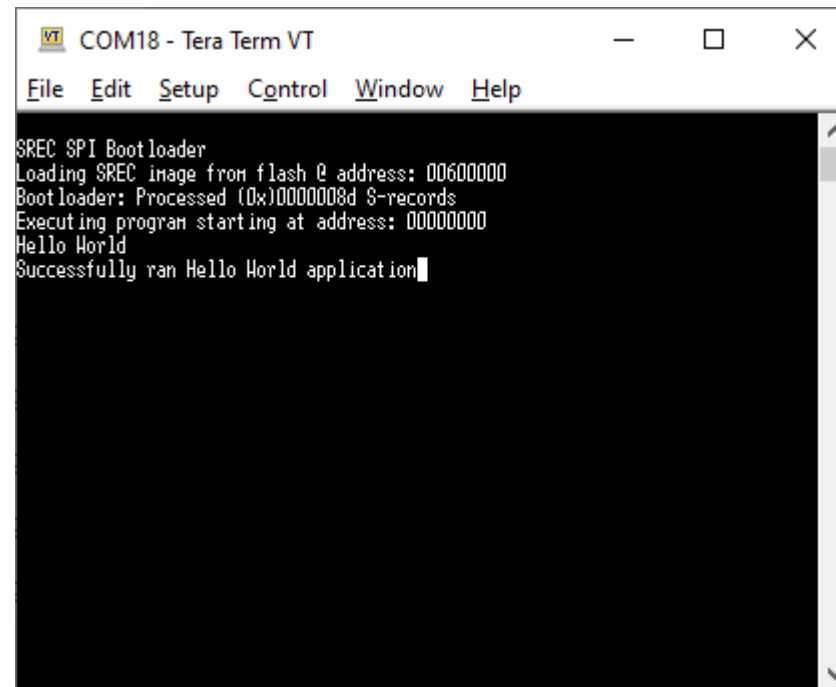
Connect a terminal 9600n1 and power cycle the board.



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



```
COM18 - Tera Term VT
File Edit Setup Control Window Help
SREC SPI Bootloader
Loading SREC image from flash @ address: 00600000
Bootloader: Processed (0x)0000008d 8-records
Executing program starting at address: 00000000
Hello World
Successfully ran Hello World application
```



ADIUVO

ENGINEERING AND TRAINING, LTD.

www.adiuvoengineering.com



adam@adiuvoengineering.com