User's Guide

AFE79xx SPI Bringup Guide With Xilinx FPGAs

Thyagaraju Naidu and Dhruvil Solanki

ABSTRACT

This tutorial guides through the process of using Xilinx Vivado and Vitis development environments to bring up Serial Peripheral Interface (SPI) and non-timing critical General-Purpose Outputs (GPOs) for Texas Instruments AFE79xx EVM along with the companion LMK series clocking chip, thereby enabling an easier integration of the AFE79xx device into a system design. This guide will demonstrate how to use a Xilinx ZCU102 setup as an example.

Table of Contents

List of Figures

List of Tables

Trademarks

All trademarks are the property of their respective owners.

1 Introduction

This user guide is a walk-through of complete hardware and software flow to bring up SPI and GPO in a AFE79xx system along with a Xilinx FPGA. The hardware in this case refers to a Xilinx Microblaze processor based block design along with AXI SPI, AXI GPIO and other required peripherals.

The specific step-wise objectives are as follows:

- Instantiate a block design with SPI IP in a Vivado project.
- Map the required signals of block design to FPGA IOs.
- Import hardware design and building a new Vitis application project for software development.
- Compile, link, and download C program to processor along with bit file for FPGA.

2 Prerequisites

For effective use of this documentation, ensure to have the following prerequisites:

- Xilinx Vitis IDE v2020.1.0 (or higher)
- Xilinx Vivado v2020.1.0 (or higher)
- Xilinx FPGA board along with TI AFE EVM
- FPGA bit file download/debug programmer
- USB-UART cable for debug terminal
- TI supplied C-APIs

Table 2-1. Prerequisites

TI AFE	AFE79xx
Sample Configuration	2T-2R-1FB
Lanes	2 RX lanes (1RX, 1FB) and 2 TX lanes at 5Gbps
AFE EVM	AFE79xx EVM
FPGA Board	Xilinx ZCU102 EVM

3 Typical Bare-Metal Design Flow

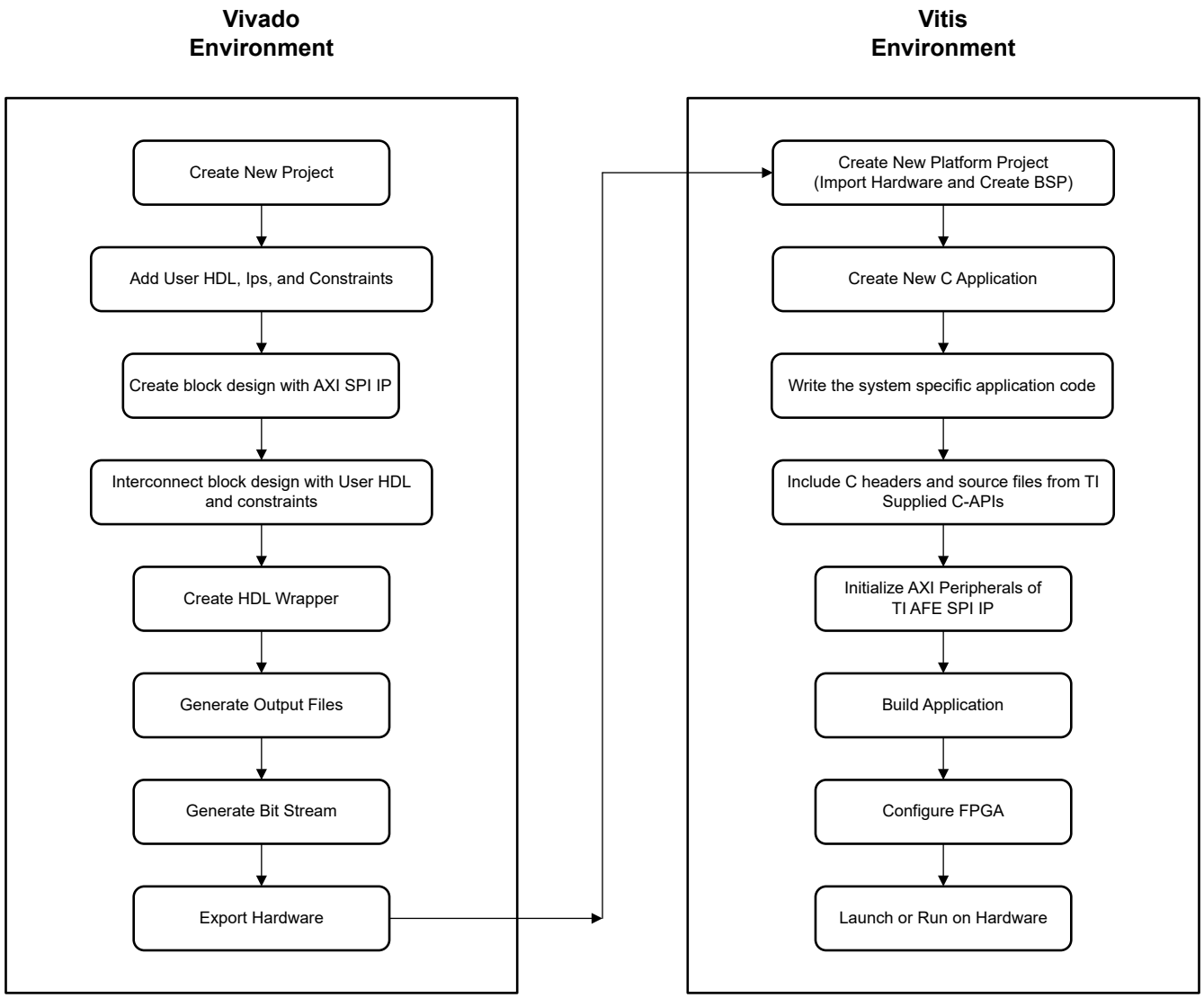


Figure 3-1. Bare-Metal Design Flow

4 Background

This example uses a soft core Microblaze because the Microblaze can be instantiated in most of the Xilinx FPGA families. The SPI, UART, and GPIO AXI blocks run on relatively lower frequency AXI clocks. As seen in Figure 4-1, the AXI peripherals are controlled by a Microblaze block through smart interconnects.

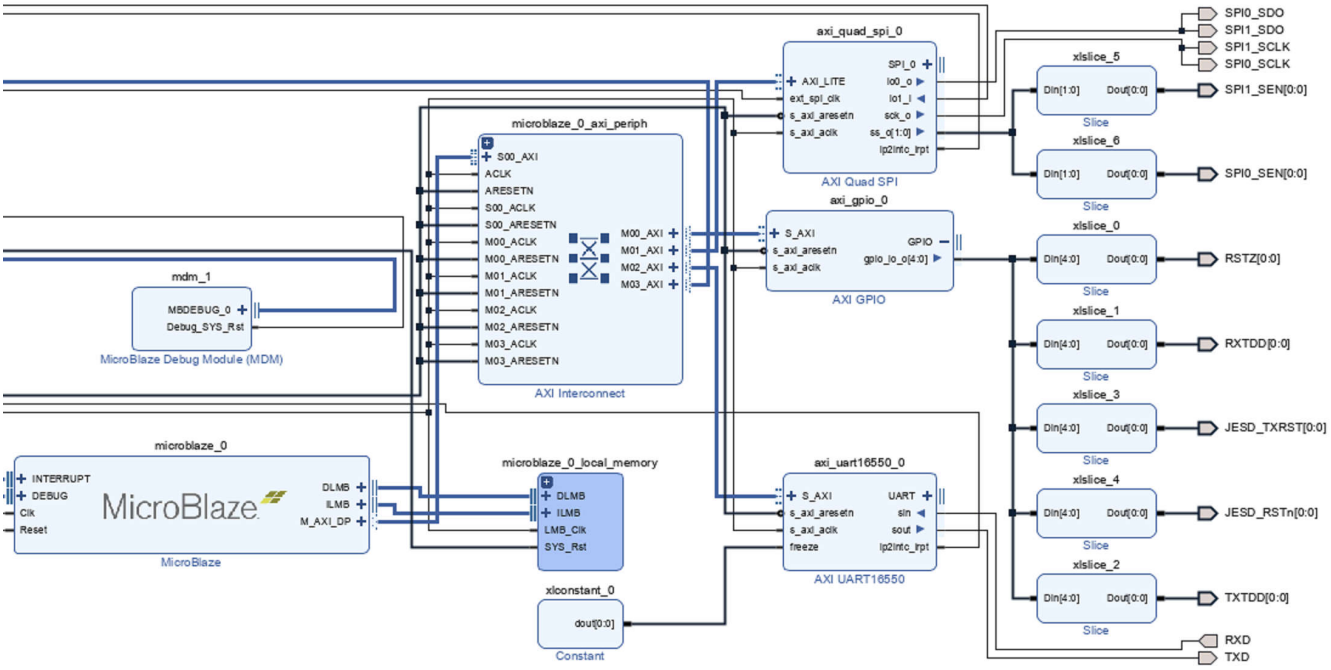


Figure 4-1. Typical Block Design With Microblaze and AXI Peripherals

The HP port of the Microblaze drives AXI peripherals block design. The clocking of the entire IP is expected from a 100MHz differential clock source. This example uses a 100MHz differential clock source because this clock is typically available as ‘user clock’ in most FPGA EVMs. All other clock frequencies are derived internally through a clocking wizard. Depending on the number of independent SPI buses required in the system, more AXI SPI IPs can be added to the block design.

5 Add Microblaze and SPI IP for Use in Vitis for Embedded Development

1. Open an existing Vivado project or create a new one.
2. Under the ‘IP Integrator’ in the left pane, click ‘Create Block Design’.

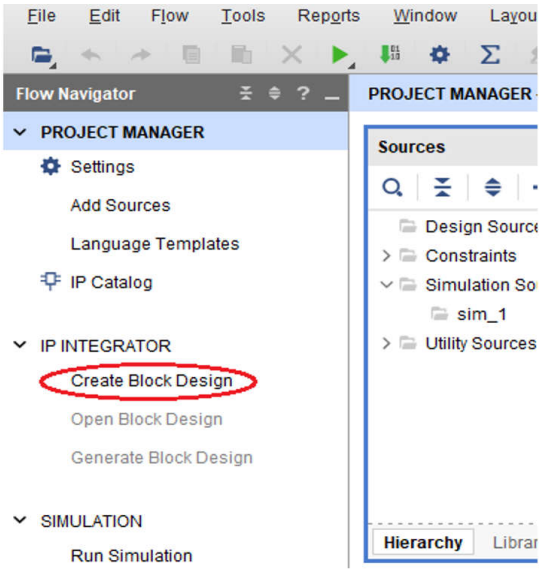


Figure 5-1. Creating Block Design

3. Give a name to the block design and click ‘OK’.

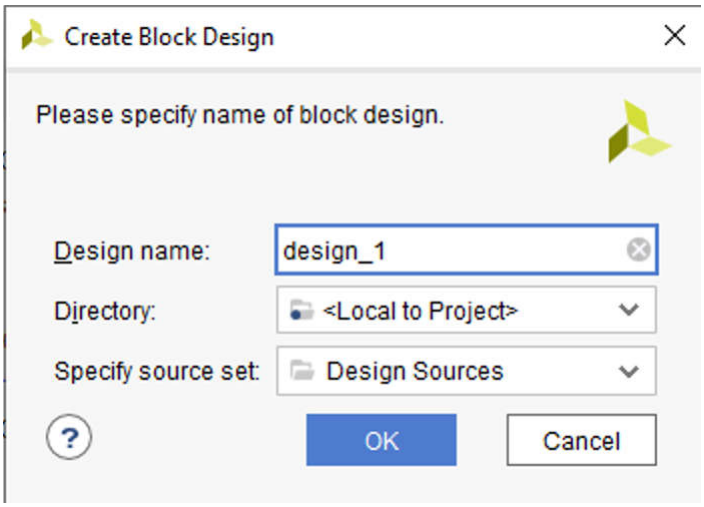


Figure 5-2. Name the Block Design

4. In the newly created block design, click ‘+’ to add IP.

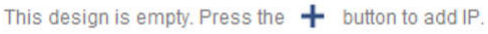


Figure 5-3. Adding IP to Block Design

5. Search for Microblaze and add ‘Microblaze’ to block design.

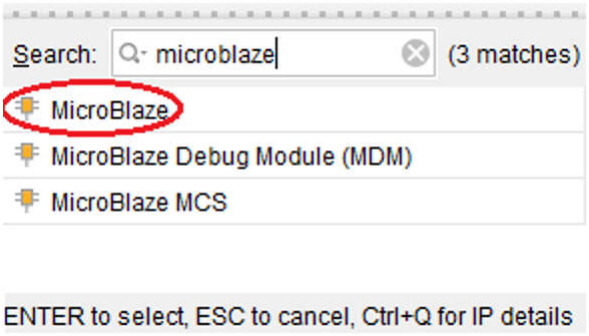


Figure 5-4. Adding Microblaze to Block Design

6. Click ‘Run Block automation’ and then ‘OK’.

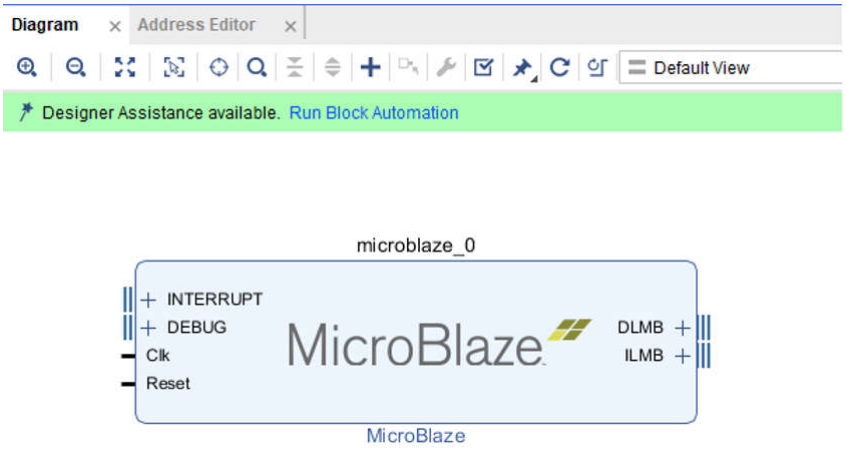


Figure 5-5. Run Block Automation for Microblaze

7. Several IP blocks companions to Microblaze are automatically added by Vivado.
8. Click ‘Run Connection automation’.
9. In the Connection Automation pop-up, select ‘CLK_IN1_D’, map it to ‘user_si570_sysclk’ and click ‘OK’.

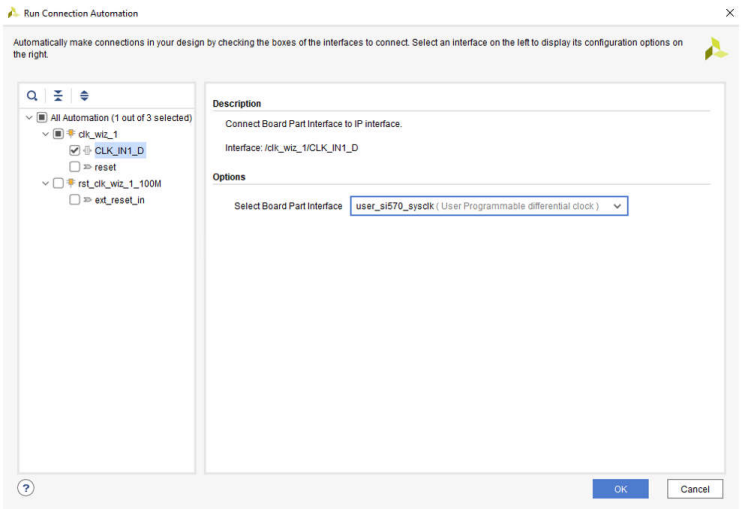


Figure 5-6. CLKIN for Microblaze

10. Click ‘Run Connection automation’ again.
11. In the Connection Automation pop-up, select ‘reset’, ‘ext_reset_in’ and map them to ‘reset (FPGA_reset)’ and click ‘OK’.

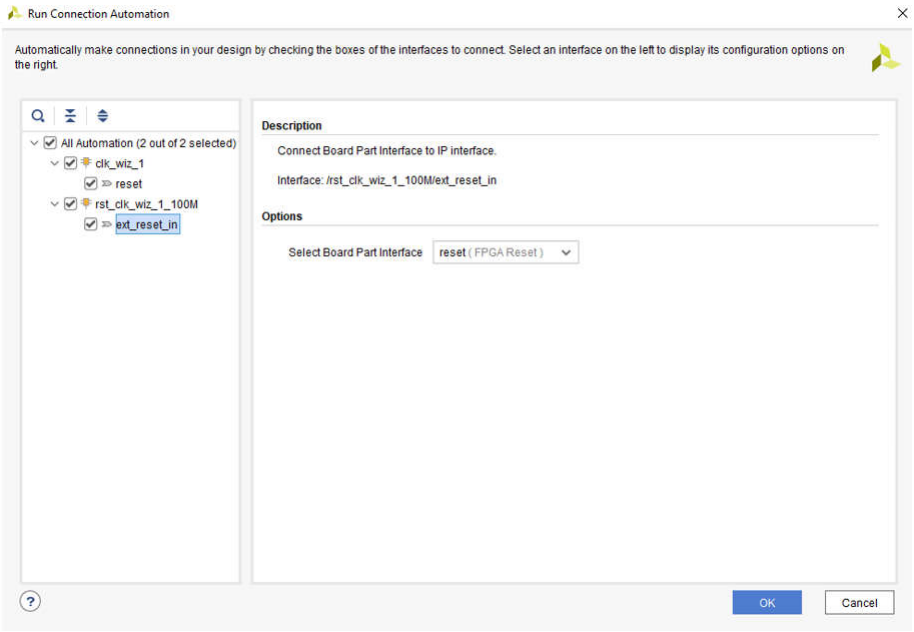


Figure 5-7. Reset Connection for Microblaze

12. Right click block design and add ‘AXI Quad SPI’ as shown in Figure 5-8 and Figure 5-9.

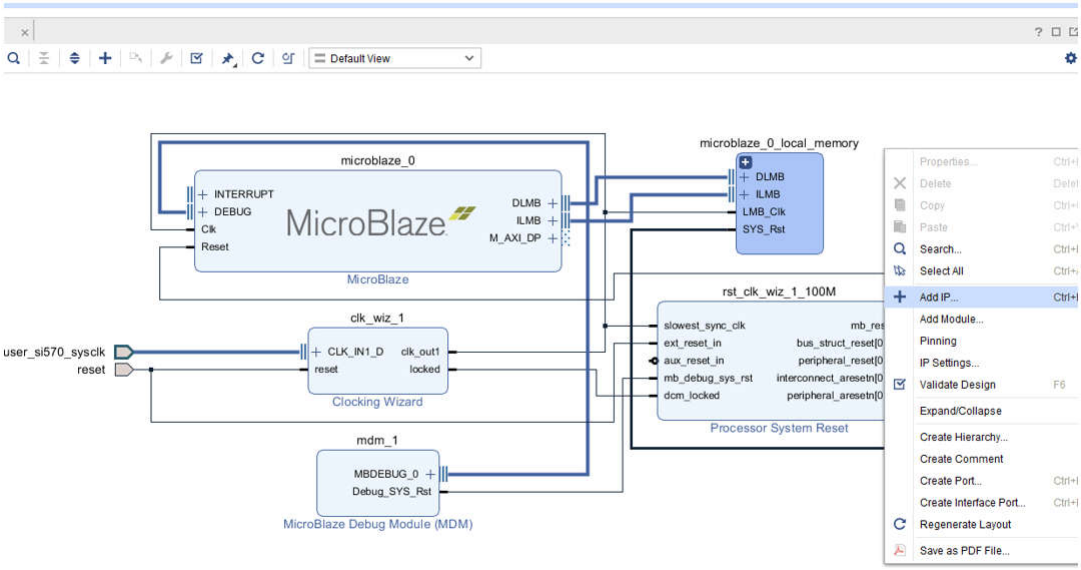


Figure 5-8. Adding IP to Block Design

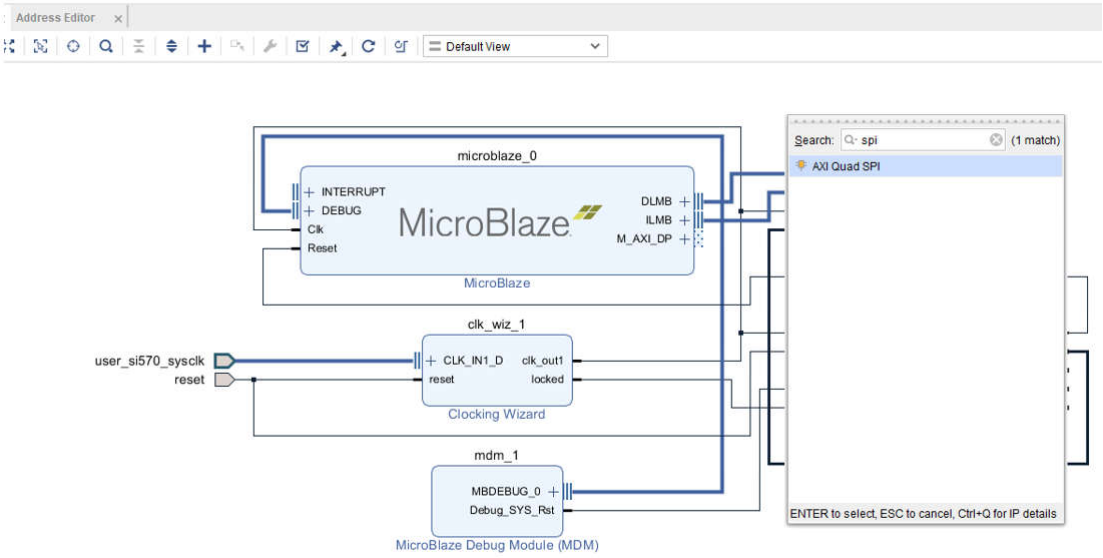


Figure 5-9. Adding 'AXI QUAD SPI' IP to Block Design

13. Click 'Run Connection automation'.

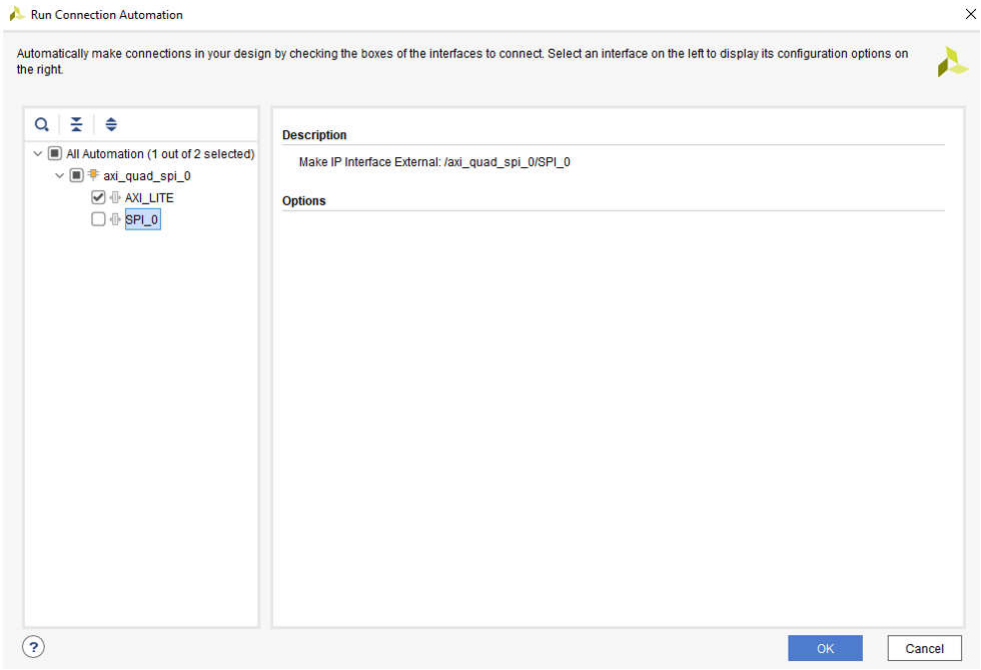


Figure 5-10. Running Connection Automation for 'AXI_LITE'

14. Select ‘AXI_LITE’ and click ‘OK’.

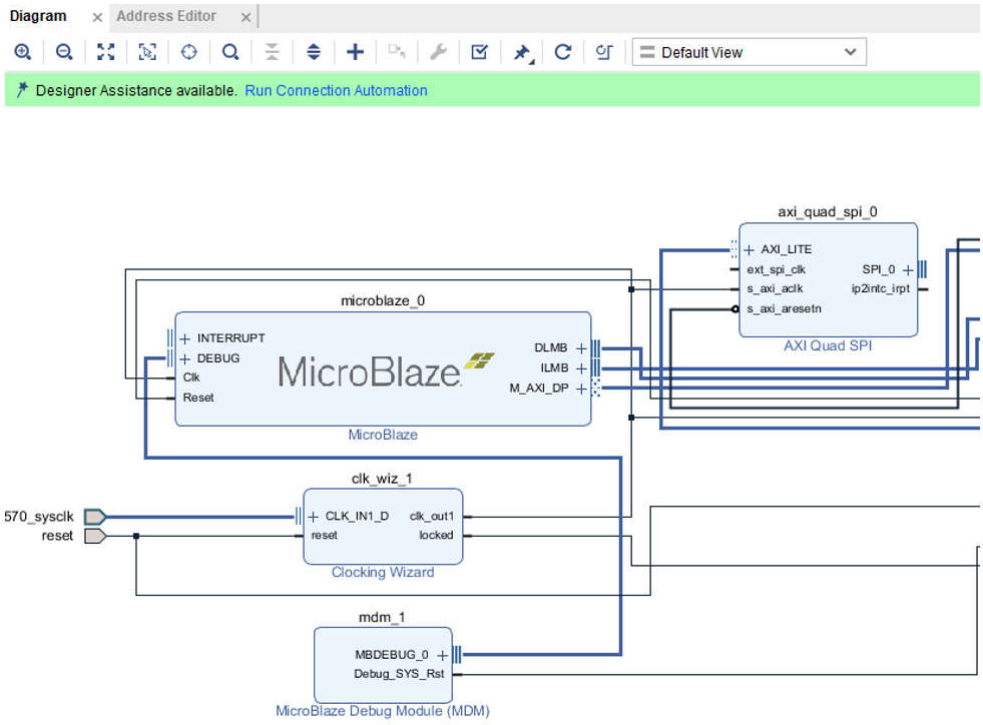


Figure 5-11. ‘ext_spi_clk’ Shows No Connection in ‘AXI QUAD SPI’

15. Connect ‘s_axi_aclk’ to ‘ext_spi_clk’.

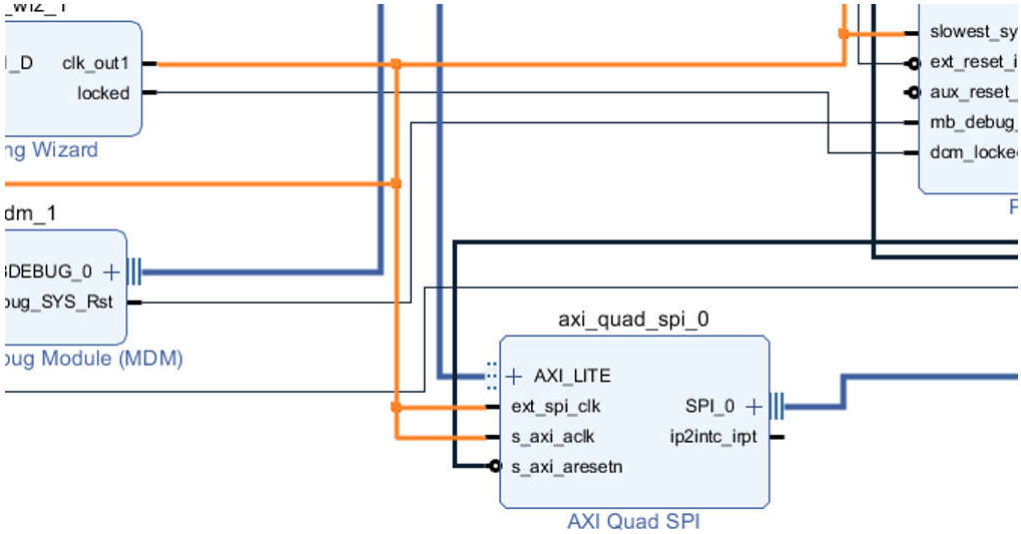


Figure 5-12. ‘ext_spi_clk’ Connected to ‘s_axi_aclk’

16. Double click ‘AXI Quad SPI’ -> select No. of slaves and then click ‘OK’.

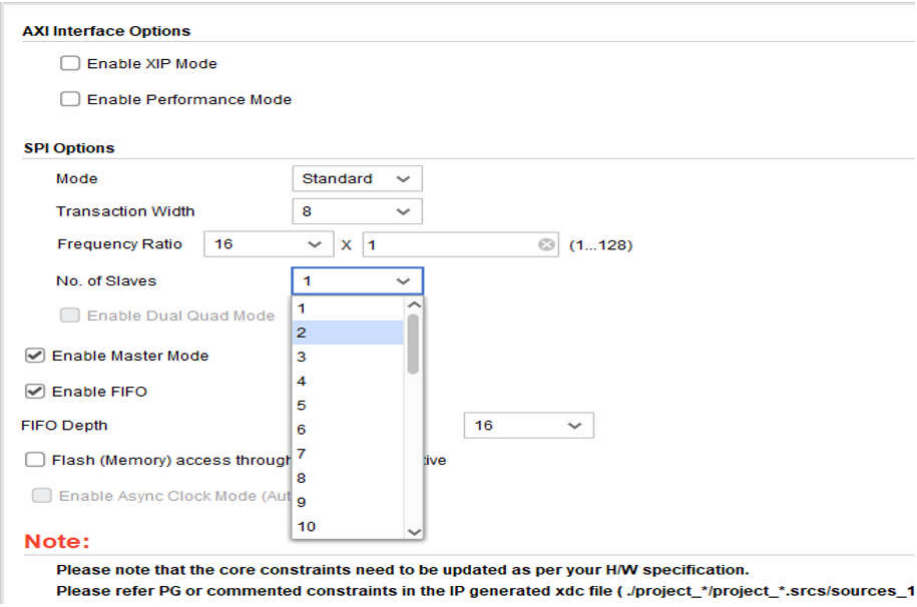


Figure 5-13. Select Number of SPI Slaves in ‘AXI QUAD SPI’

17. From the Quad SPI IP, map the signals as following:
    a. ‘io0_o’ -> SPI_SDO
    b. ‘Io1_i’ <- SPI_SDI
    c. ‘sck_o’ -> SPI_SCL
    d. ‘ss_o[1:0] -> SPI_SEN0, SPI_SEN1
18. The ‘ss_o’ bit width will be based on No. of slaves selected in step:16

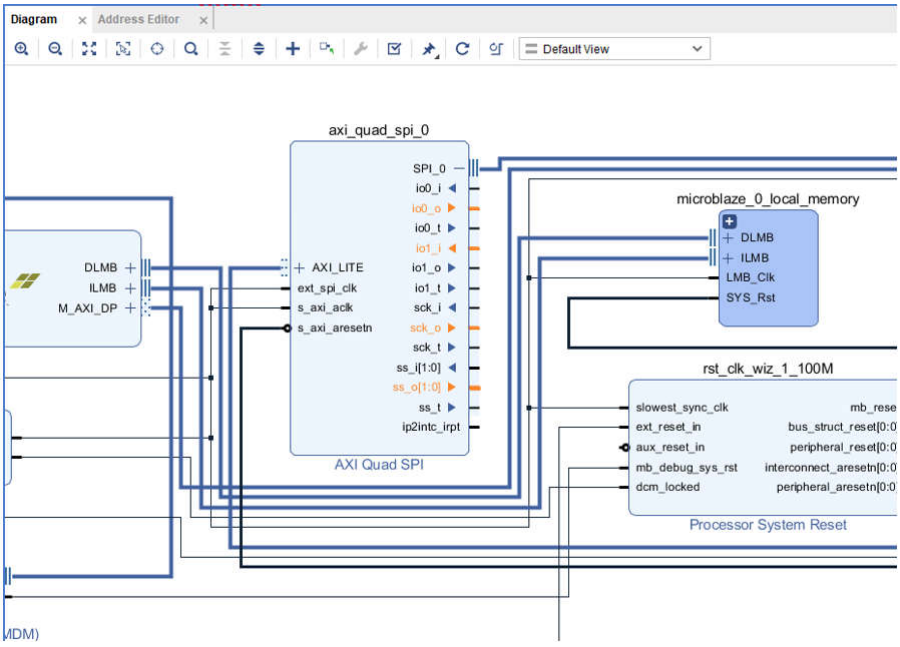


Figure 5-14. Highlighting Ports for External Connections in ‘AXI QUAD SPI’

19. Validate the design to ensure no errors as shown in Figure 5-15.

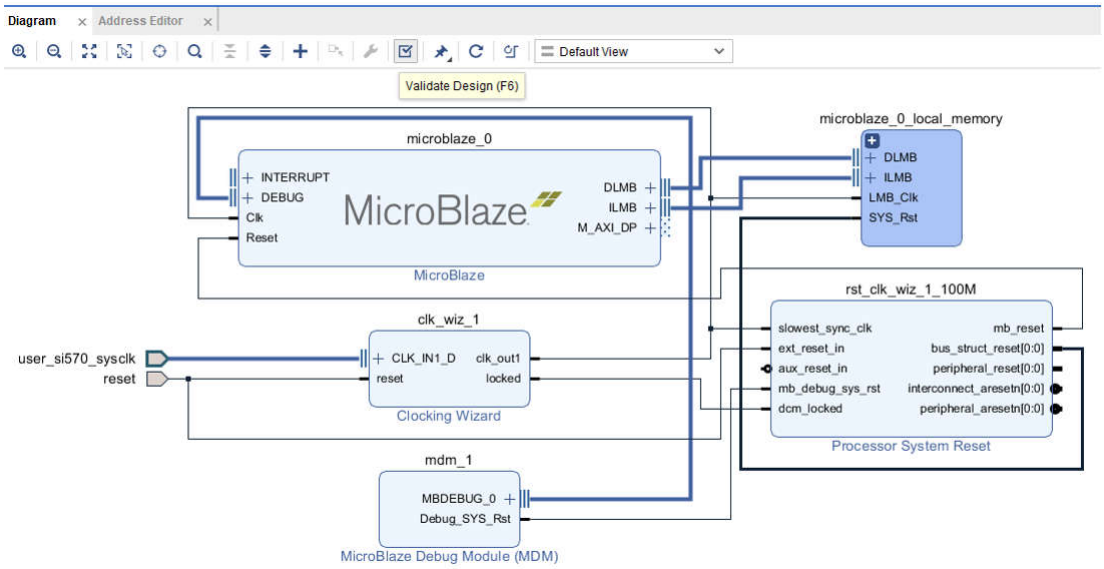


Figure 5-15. Validating Block Design

20. To add GPIOs, add 'AXI GPIO' from catalog and repeat similar steps as above.

6 Create New Platforms in Vitis

To create a new platform in Vitis, follow these steps below:

1. Open the *File* menu, go to *New*, and then click *Platform Project* (see Figure 6-1).

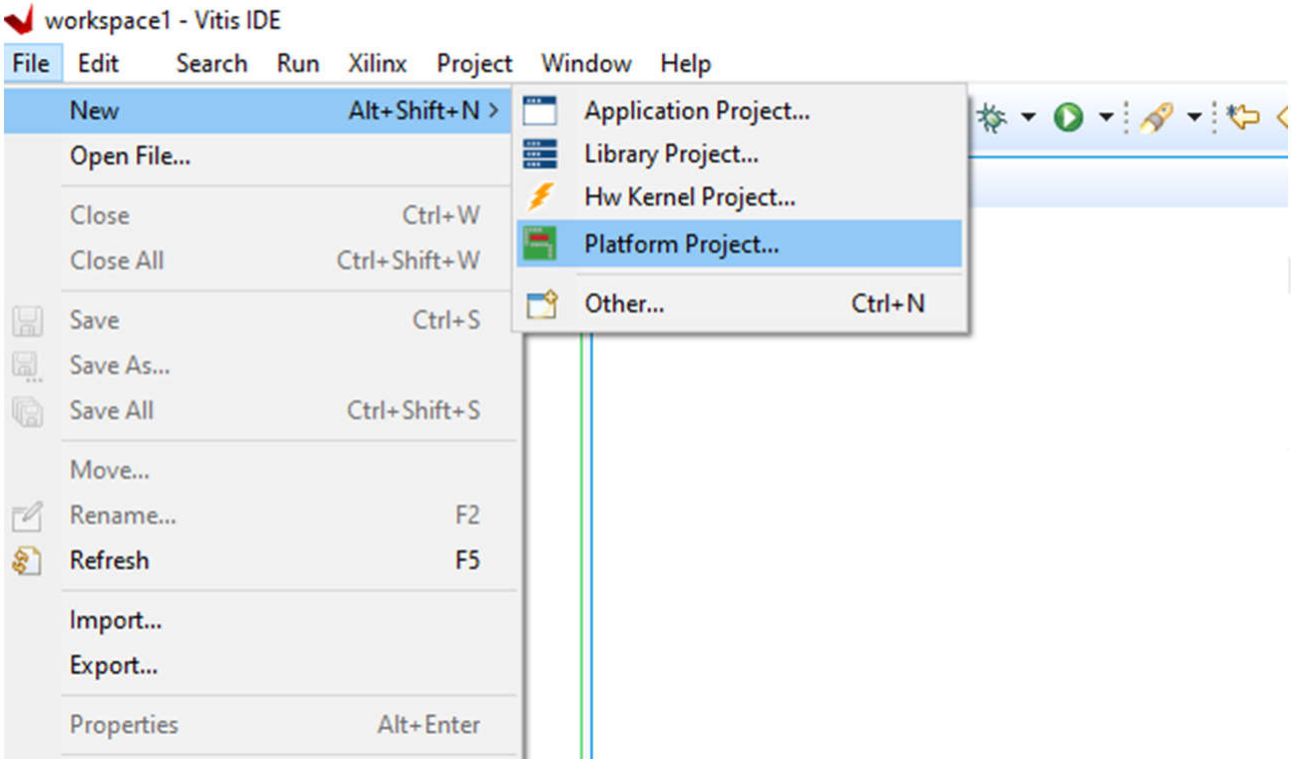


Figure 6-1. New Platform Project

2. Enter the desired platform name. The name *ZCU102ps* was used as an example (see Figure 6-2).

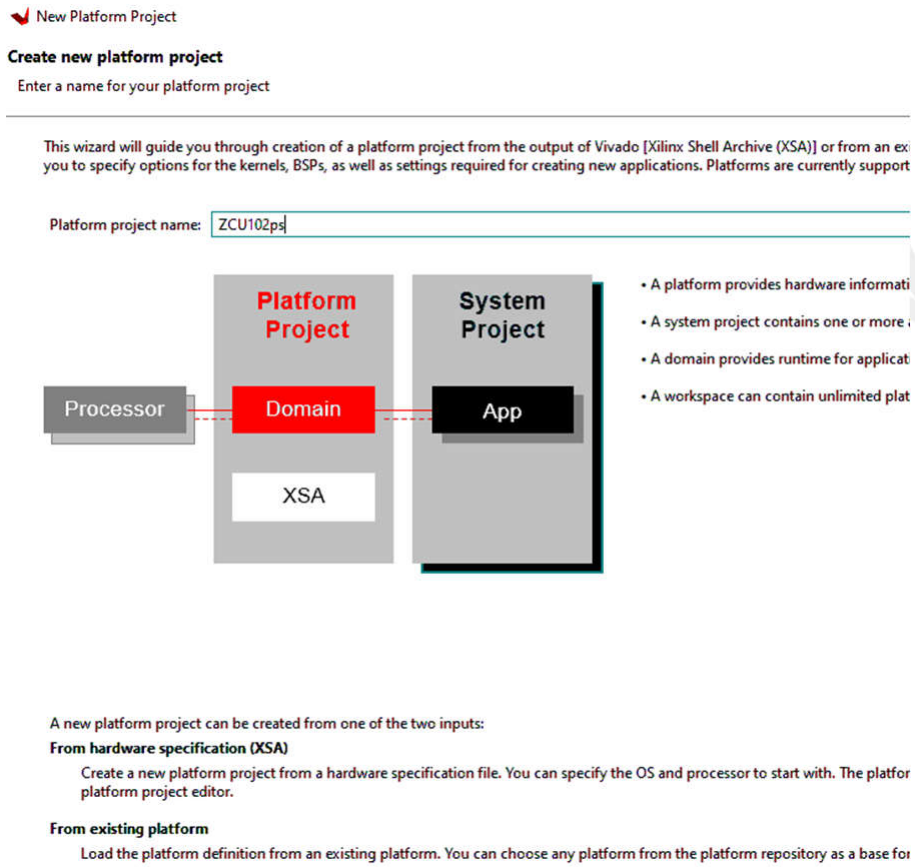


Figure 6-2. Naming of Platform Project

3. After the new platform is named, a menu appears (see Figure 6-3). Select the *XSA* (Xilinx Support Archive) file.

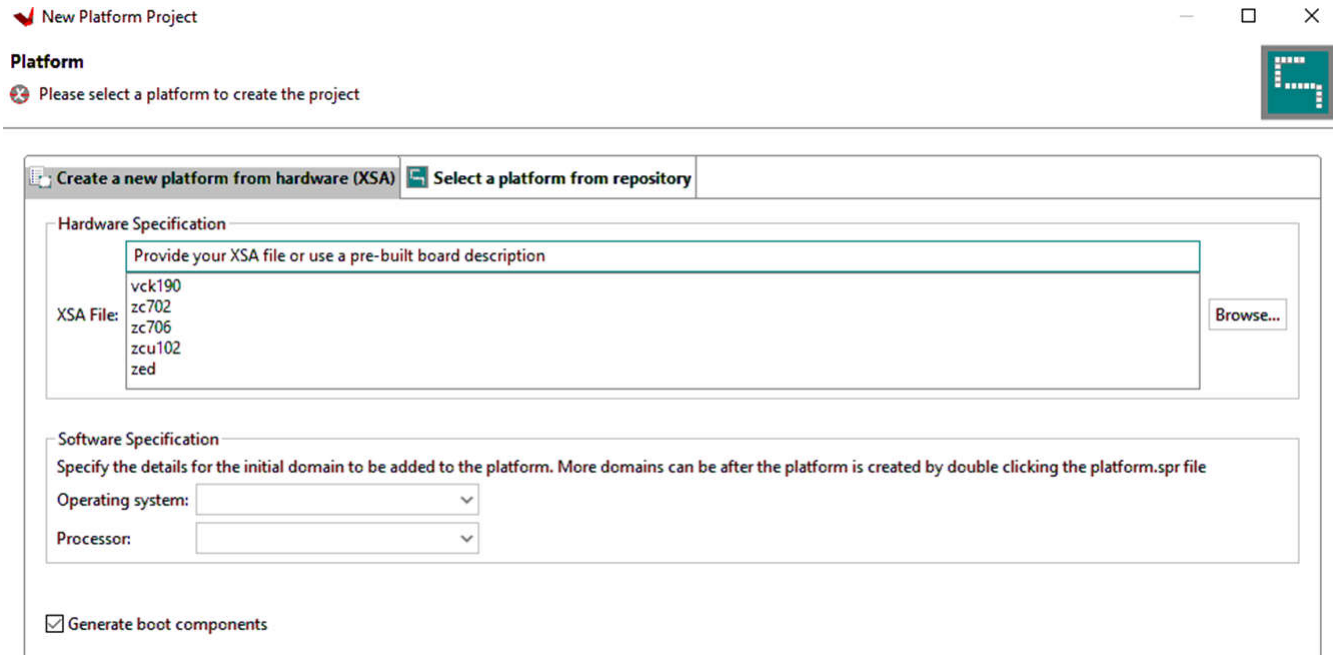


Figure 6-3. Hardware Specification

4. Browse and select the *.XSA* file from the FPGA folder shared along with this document (see Figure 6-4).

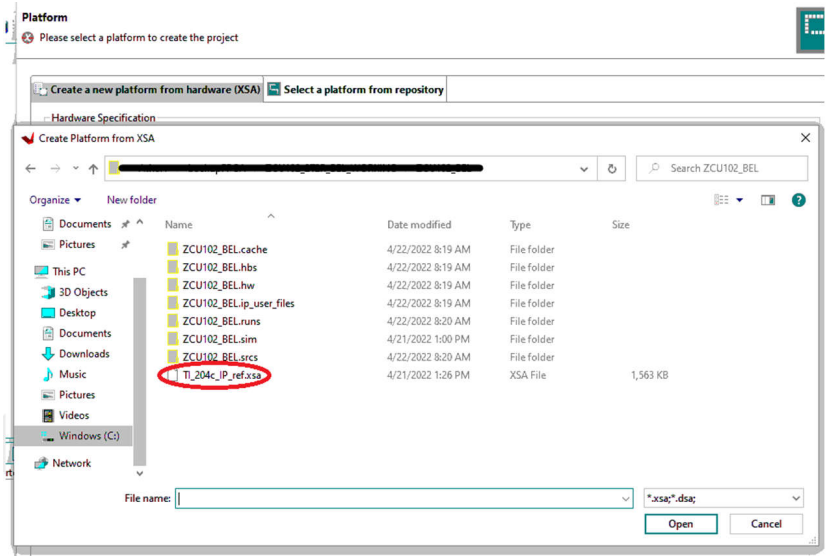


Figure 6-4. Selecting the Platform

5. Right-click the new platform project to open the drop-down menu. Click *Build Project* to start the build (see Figure 6-5). This can take some time to complete the build.

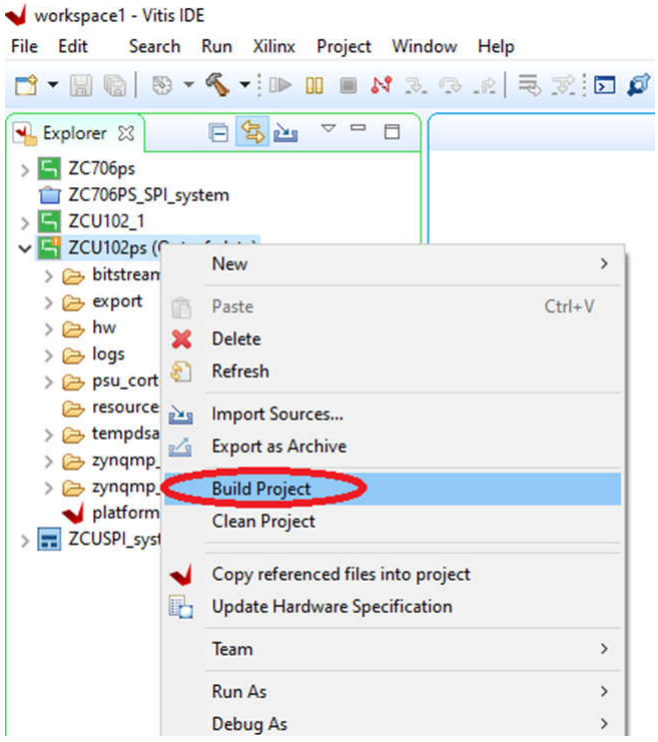


Figure 6-5. Building the New Project

7 Create New Application Projects in Vitis

After the build is complete, create a new application project in Vitis. To create a new project, follow these steps:

1. Right-click the *Platform* project, go to *New*, then click *Application Project* (see Figure 7-1).

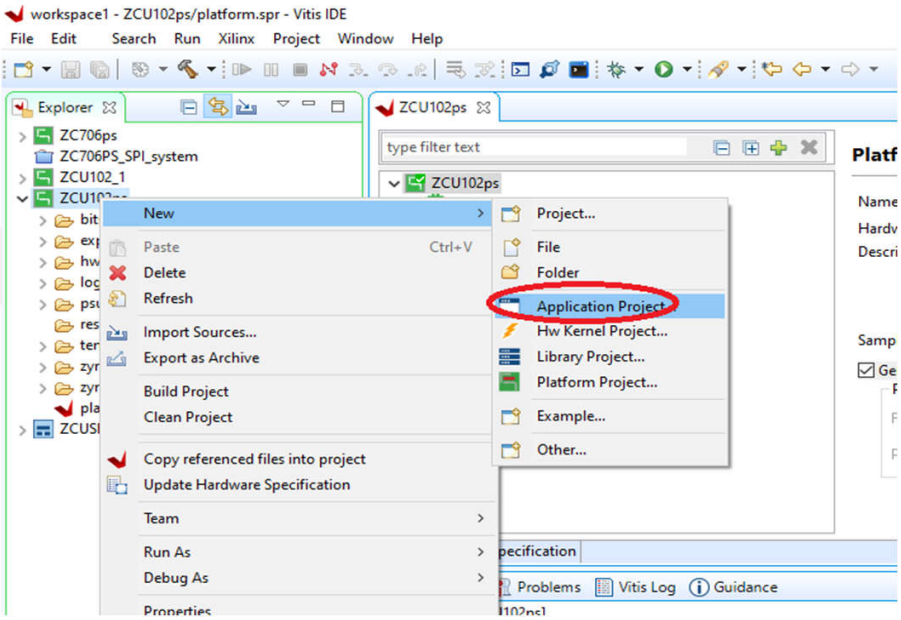


Figure 7-1. Creating New Application Project

2. When the *New Application Project* window appears (see Figure 7-2), click *Next*.

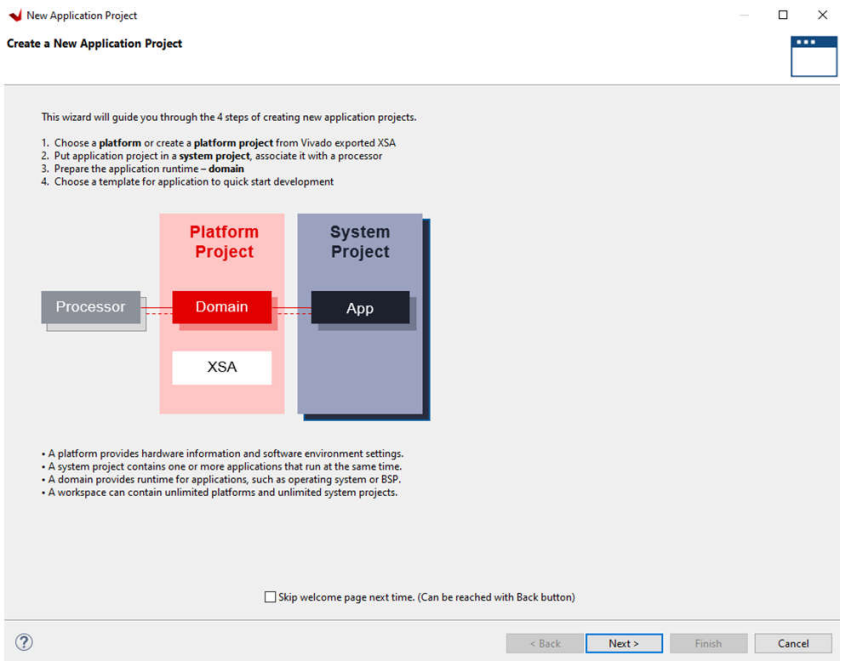


Figure 7-2. New Application Project

3. Select the newly created platform *ZCU102ps* and click *Next* (see Figure 7-3).

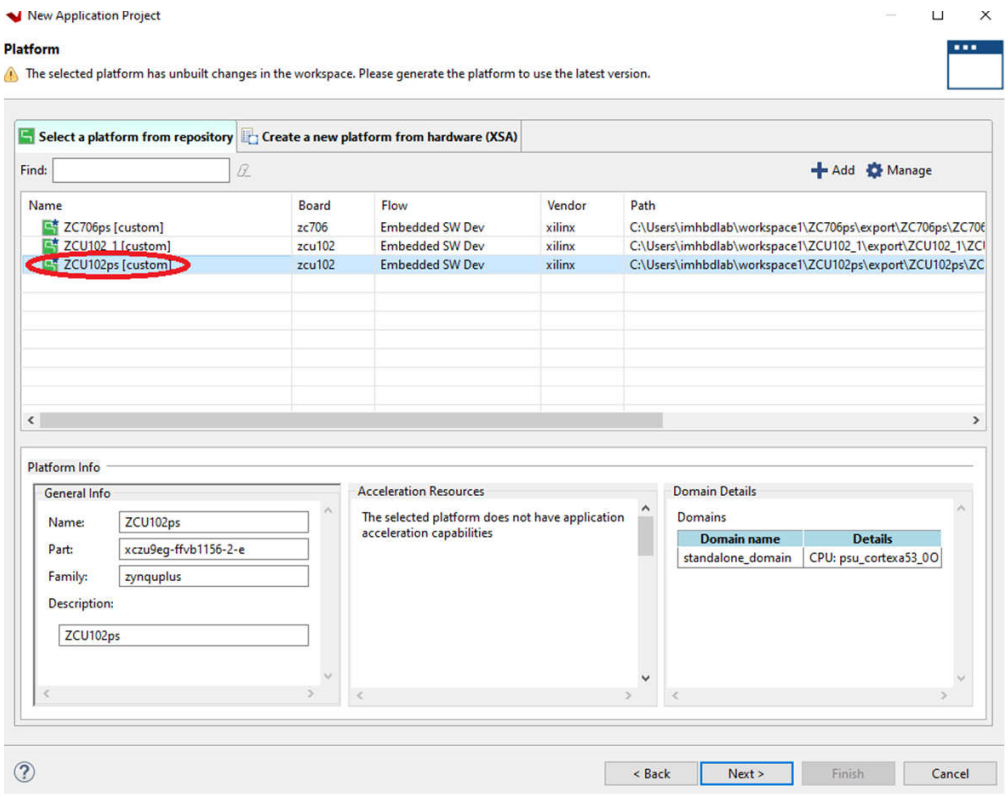


Figure 7-3. Selecting the Application Project

4. Type a new application name. *ZCU102ps_SPI* was used as an example.

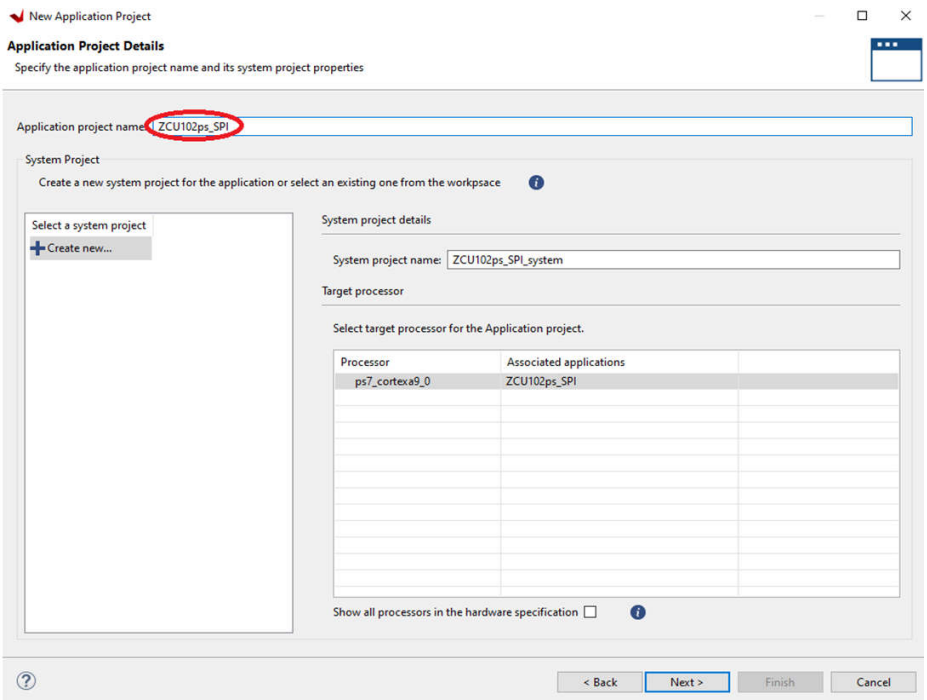


Figure 7-4. New Application Project Name

5. Select *standalone on microblaze_0* and click *Next* (see Figure 7-5).

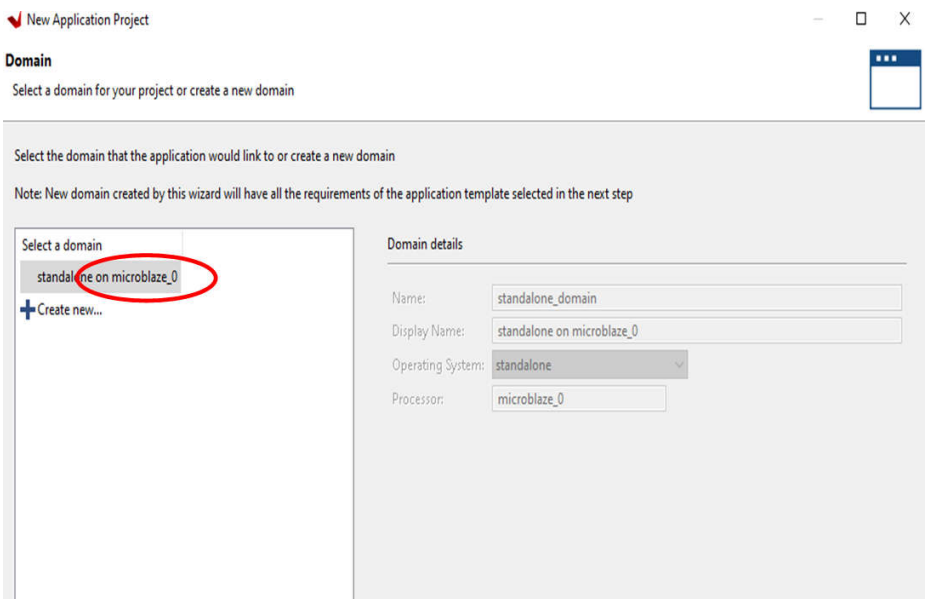


Figure 7-5. Application Project Name

6. Select *Hello World* from the list of templates and click *Finish* (see Figure 7-6).

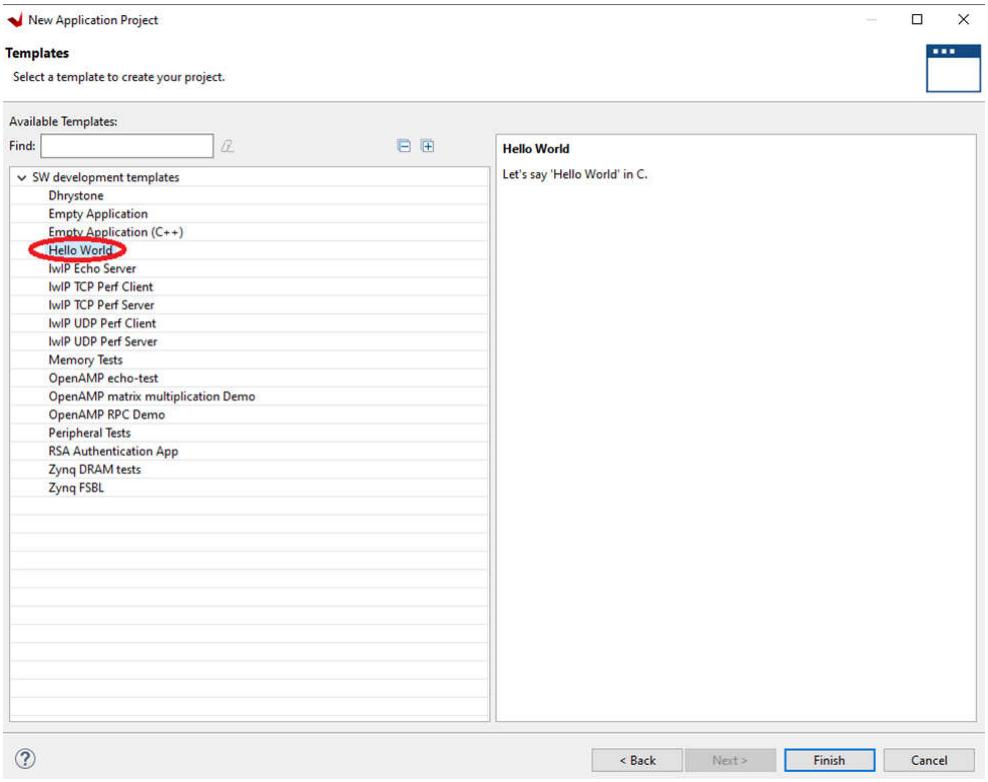


Figure 7-6. Selecting Template

7. A fresh C project appears on top where the actual application development can start.

8 Build Application Projects

To build an application project, follow these steps:

1. Right-click the application name and select *Build Project* (see Figure 8-1).

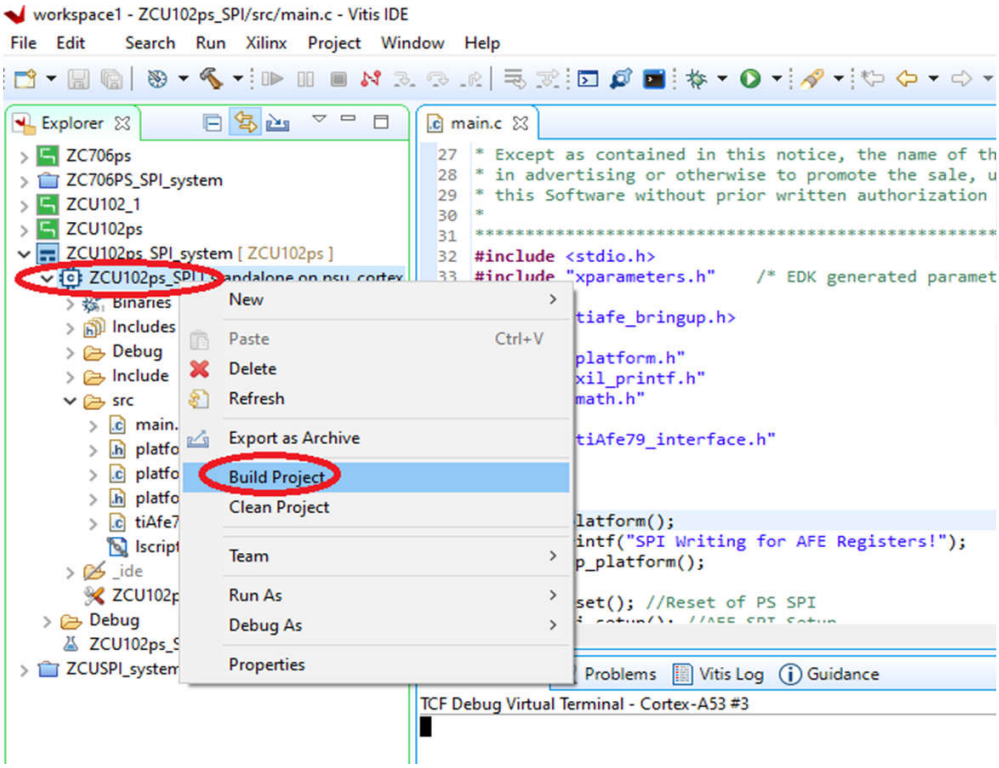


Figure 8-1. Building Project

2. Ensure that the project builds without any errors.

9 Generate SPI Log for AFE79xx EVM

The SPI log generation is split into three different parts:

1. Use the AFE79xx GUI to generate the LMK SPI Log
2. Use the AFE79xx GUI to generate the AFE SPI Log
3. Convert the generated SPI Logs to the format used in Vitis

9.1 Generating the LMK SPI Log

In the AFE79xx GUI the below lines can be added in order to enable the logging of the LMK SPI writes. These should be added before AFE.LMK.lmkConfig() line or AFE.deviceBringup().

```
lmklogDumpInst=mLogDump.logDump(ASTERIX_DIR+DEVICES_DIR+r"\Afe79xxPg1_LMK.txt")
lmklogDumpInst.logFormat=0x1
lmk.logClassInst = lmklogDumpInst
lmk.rawWriteLogEn=1
```

After running the script in the AFE79xxGUI a file, named “Afe79xxPg1_LMK.txt", containing the registers writes for the LMK will be generated in the following folder: ‘C:\Users\<UserName>\Documents\Texas Instruments\Afe79xxLatte\lib’

9.2 Generating the AFE SPI Log

In the AFE79xx bringup script the parameter below can be updated in order to enable the logging of the AFE SPI writes in the two formats, Format 1 and Format 5.

- logDumpInst.logFormat to ‘0x21’

After running the script in the AFE79xxGUI a file, named “Afe79xxPg1Format5’.txt", containing the registers writes for the AFE will be generated in the following folder: ‘C:\Users\<UserName>\Documents\Texas Instruments\Afe79xxLatte\lib’.

9.3 Converting SPI Logs to Format for Vitis

In the AFE79xx Secure folder you can find a script called ‘SPI_Convert.py’, which is used to convert the generated SPI logs into the correct format. This script should be opened in the AFE79xx GUI and after running the script two files called ‘AFEspiwrites.c’ and ‘LMKspiwrites.c’ will be generated.

All the commands in these spiwrites files can be copied and pasted into the tiafe_bringup.h file in the Vitis project.

1. The contents of the ‘AFEspiwrites.c’ file must be copied into the bringupafe() function. In the figure below the function that contains the commands is shown.

```
52 #include <stdio.h>
53 #include <stdint.h>
54 #include "xil_printf.h"
55 #include <tiAfe79_interface.h>
56
57 void bringupafe()
58 {
59     int rdVal=0;
60     int pollIter=0;
61     int pollVal=0;
62
63 }
64
```

2. The contents of the ‘LMKspiwrites.c’ file must be copied into the lmk_config() function. In the figure below the function that contains the commands is shown.

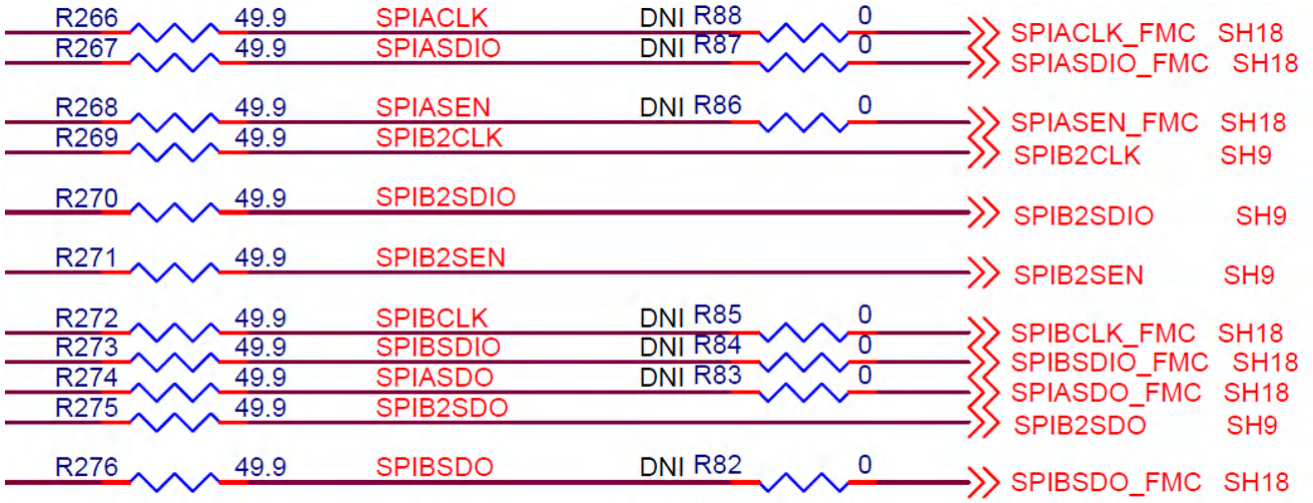
```
66 void lmk_config()
67 {
68
69
70 }
```

After updating the tiafe_bringup.h file in Vitis the application project should be built again, following the steps outlined in Section 8 Build Application Projects.

10 AFE79xxEVM Board Modifications

To connect AFE SPI to the FPGA, the modifications below must be made:

1. Remove R266, R267, R268, and R274 (Found on the bottom of the EVM).
2. Install 0Ω resistors for R88, R87, R86, and R83 (Found on the top of the EVM).



The LMK SPI has no connection to the FMC connector, so for the FPGA to control the LMK SPI the following board modifications must be made.

1. Connect LMK_SCK to SPIACLK (Wire from R88 to R248).
2. Connect LMK_SDIO to SPIASDIO (Wire from R87 to R249).
3. Connect LMK_CS to pin D26 of the of the FMC connector.
   a. To do this a wire can be connected from R242 to the open pad of R9 that is connected to the FMC.

11 Configure the AXI GPIO

Sequence for AXI GPIO configuration should be:

1. Initialize GPIO module
2. Set Direction
3. Set High or Low for corresponding bits

The above sequence must be completed before initializing AFE and LMK devices.

11.1 Initializing the GPIO

The C syntax for initializing the AXI GPIO is in two steps:

1. XGpio GPOs: Initialize a pointer (GPOs) to the GPIO configuration register.
2. XGpio_Initialize(&GPOs, XPAR_AXI_GPIO_0_DEVICE_ID): Refer to *Xparameters.h* to find the correct AXI GPIO DEVICE ID.

11.2 Setting the Direction

XGpio_SetDataDirection(&GPOs, 1, 0);

First argument *&GPOs* point to the GPIO instance initialized in previous command.

Second argument 1 indicates the GPIO bank. In this example, only the first bank is used.

Third argument 0 indicates all GPIO bits are set for outputs.

11.3 Setting High or Low for Corresponding Bits

XGpio_DiscreteWrite(&GPOs, 1, regval);

First argument *&GPOs* point to the GPIO instance initialized in previous command.

Second argument 1 indicates the GPIO bank. In this example, only the first bank is used.

Table 11-1. Bit Mapping of IP I/Os

Bit Description	Bit Position
JESD RSTn	4
JESD TXRST	3
RSTn	2
RXTDD	1
TXTDD	0

For Example:

XGpio_DiscreteWrite(&GPOs, 1, 0x14);

This command sets *JESD RSTn* and *RSTn* to 1, sets all other bits to 0

12 Configure the AXI SPI

The AXI SPI instance in TI AFE SPI IP is used in *Standard Mode*.

Peripherals *select 0* and *select 1* are used as chip selects for AFE and LMK clocking device, respectively

SCL frequency is hard coded to 10 MHz within TI IP

Key commands for SPI initialization and usage in Vitis are as explained below:

1. *XSpi_Config *ConfigPtr*;

   Initialize a pointer (ConfigPtr).
2. *ConfigPtr = XSpi_LookupConfig(XPAR_AXI_QUAD_SPI_0_DEVICE_ID)*;

   Refer ‘Xparameters.h’ to find the correct AXI QUAD SPI DEVICE ID.
3. *XSpi_CfgInitialize(&Spidev, ConfigPtr, ConfigPtr->BaseAddress)*;

   Initialize a new instance of SPI (Spidev).
4. *XSpi_SetOptions(&Spidev, XSP_MASTER_OPTION)*;

   Set the Spidev instance to be in Controller mode.
5. *XSpi_Start(&Spidev)*;
6. *XSpi_SetSlaveSelect(&Spidev, 1)*;

   Select Peripheral: AFE.
7. *XSpi_SetSlaveSelect(&Spidev, 2)*;

   Select Peripheral: LMK.
8. *XSpi_Transfer(&Spidev, WrBufdev, RdBufdev, 3);*

   Second argument WrBufdev is an array with 3 bytes (24-bit data to be transmitted on SPI).

   Third argument RdBufdev is an array with 3 bytes, the last byte has the SPI read value.

   Fourth argument is number of bytes to be transmitted/received...3 in our case.

The D23 is the MSB bit of the 24-bit data because the D23 indicates whether it is a Read or a Write SPI operation:

- If D23 is set to 1, then it is a read operation and RdBufdev[2] stores the read back address contents
- If D23 is set to 0, then it is a write operation and RdBufdev[2] has no significance

13 Set Up and Power on Hardware

To set up and power on the hardware, follow these steps:

1. Dock the AFE EVM to J5 (HPC0) FMC on the ZCU102 board.
2. Connect a 1.5-GHz signal through a clock source to RFROM EVM at J14.
3. Connect J2 (JTAG) and J83 (UART) USB connectors from ZCU102 FPGA board to the computer.
4. Connect the 12-V Xilinx EVM Adapter for the ZCU102 at J52.
5. After all the above connections are made, power up the setup. Note that the AFE EVM in this example is completely powered by the ZCU102 FMC interface.

14 Set up ZCU102 Board Interface for VADJ_FMC

To set the ZCU102 board interface for VADJ_FMC, follow these steps:

1. Execute the *ZCU102-Board User Interface* software (available for download from Xilinx.com).
2. Select the appropriate COM Port to enable communication between the onboard MSP430 of the ZCU102 and the PC. This software is required to turn on the FMC_AUX supply of 1.8V for the FMC bank of FPGA (see Figure 14-1).

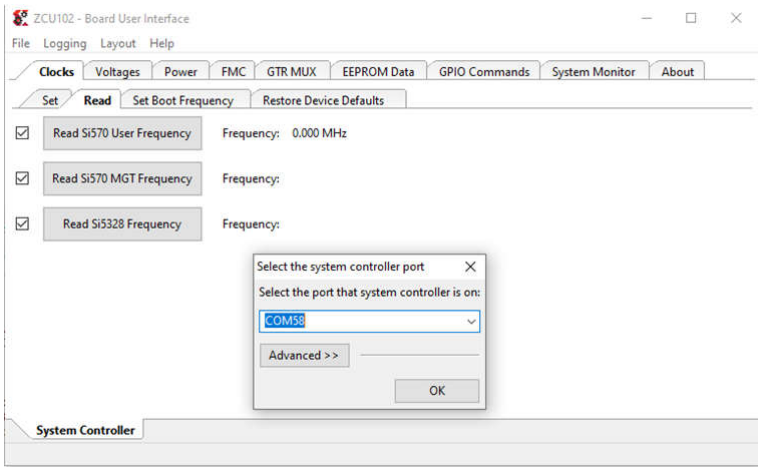


Figure 14-1. ZCU102 Board User Interface

3. Select the *Set VADJ to 1.8V* check box (see Figure 14-2).

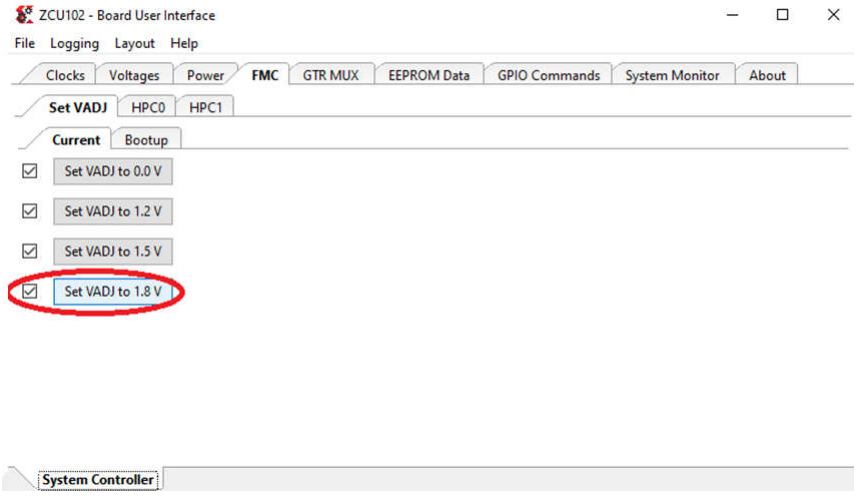


Figure 14-2. Setting VADJ

4. Confirm the same by reading the VADJ_FMC voltage. The voltage value must be 1.80V (see Figure 14-3).

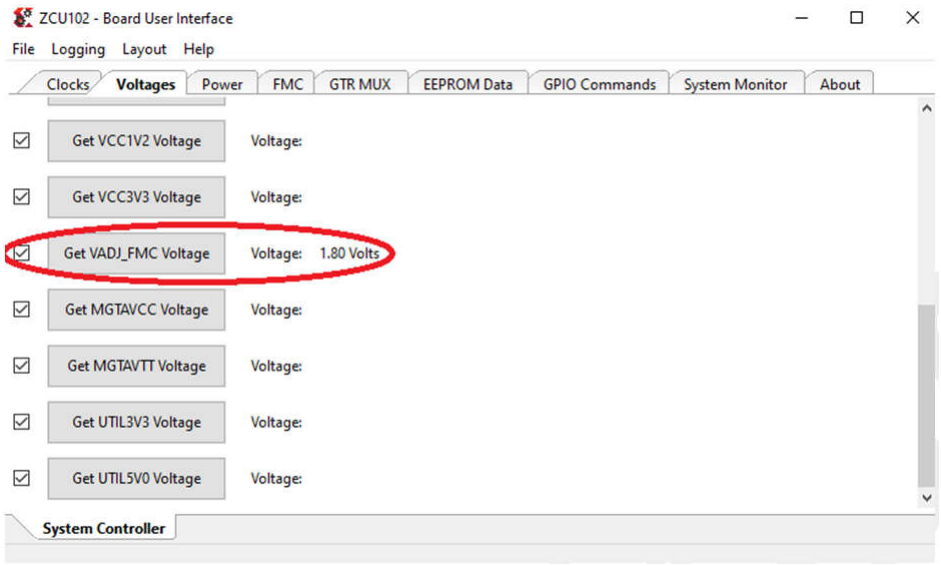


Figure 14-3. VADJ_FMC Voltage

15 Debug Application Projects and Set up Vitis Serial Terminal

To debug the application project and set up the Vitis serial terminal. follow these steps:

1. Right-click the project name and go to *Debug As* from the drop-down menu. Click *Launch on Hardware (Single Application Bug)* to run the debug (see Figure 15-1).

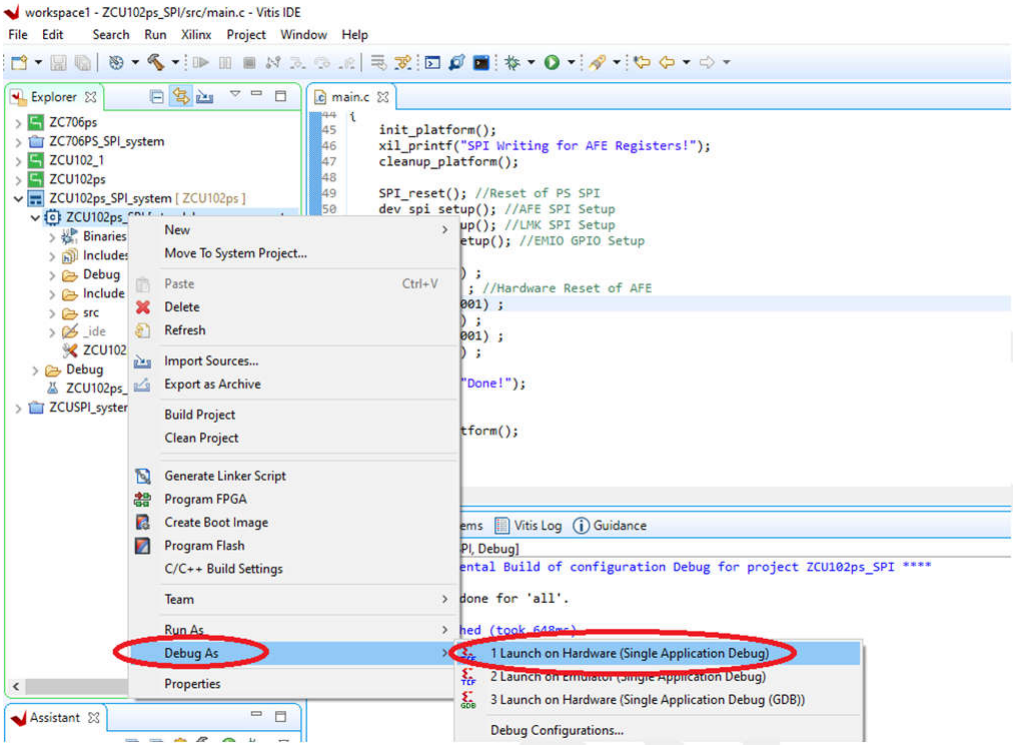


Figure 15-1. Debugging Application Project

2. Connect the Vitis Serial terminal (see Figure 15-2) with baudrate 115200 (this can be used to see SPI write or read status).

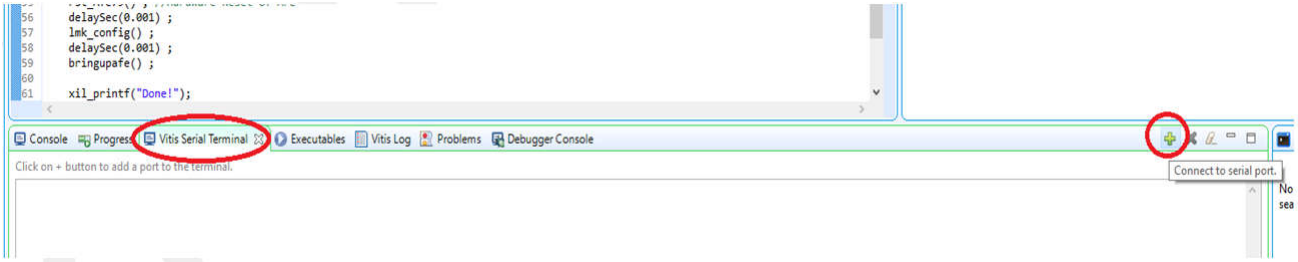


Figure 15-2. Vitis Serial Terminal

16 Execute the Application

To execute an application run, follow these steps:

1. Click the right arrow button as shown in Figure 16-1.

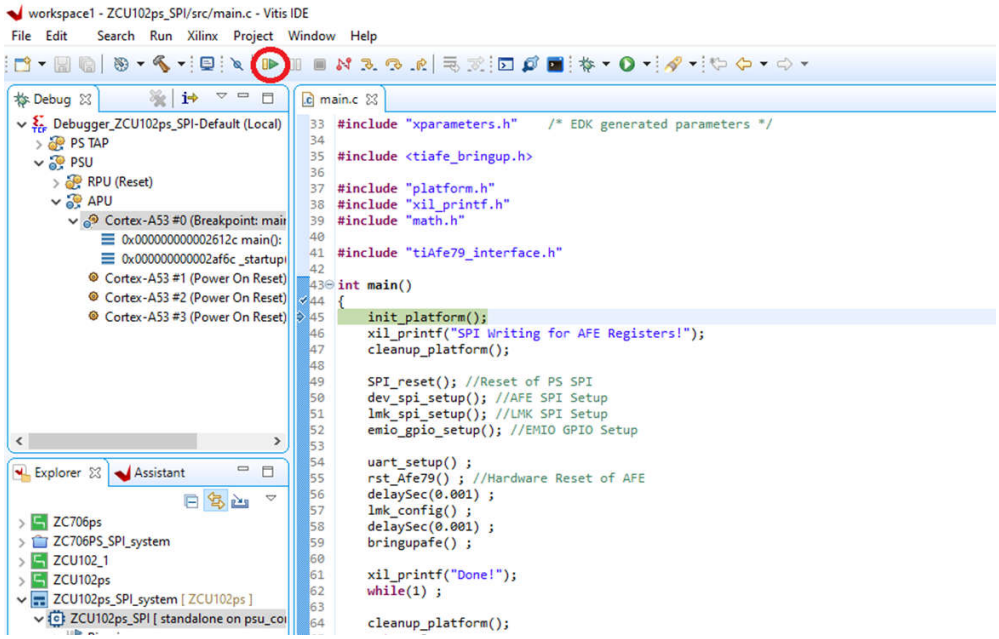
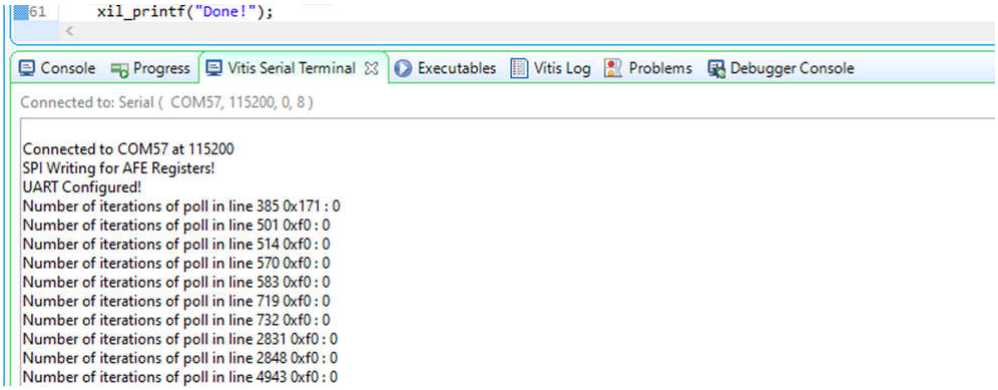


Figure 16-1. Executing Application

2. Notice the SPI logs being printed on the UART terminal:



17 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Revision * (November 2022) to Revision A (May 2024)	Page
• Changed *Abstract* section	1
• Updated the numbering format for tables, figures and cross-references throughout the document	2
• Deleted *AFE SPI IP Container Pinout*, *TI AFE SPI IP Container* and *Create Block Designs With TI AFE SPI IP* sections	2
• Updated Section 1	2
• Deleted *TI supplied AFE SPI IP* from Section 2	2
• Updated Section 3	3
• Updated Section 4	4
• Added Section 5	5
• Deleted *AFE SPI IP Container Pinout*, *TI AFE SPI IP Container* and *Create Block Designs With TI AFE SPI IP* sections	11
• Added Section 9	17
• Added Section 10	19
• Updated Section 11.2	19
• Updated Section 11.3.	20
• Deleted the *Create Boot Images to Run on SD Card* section	21

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265