Vivado and Vitis Development Flow

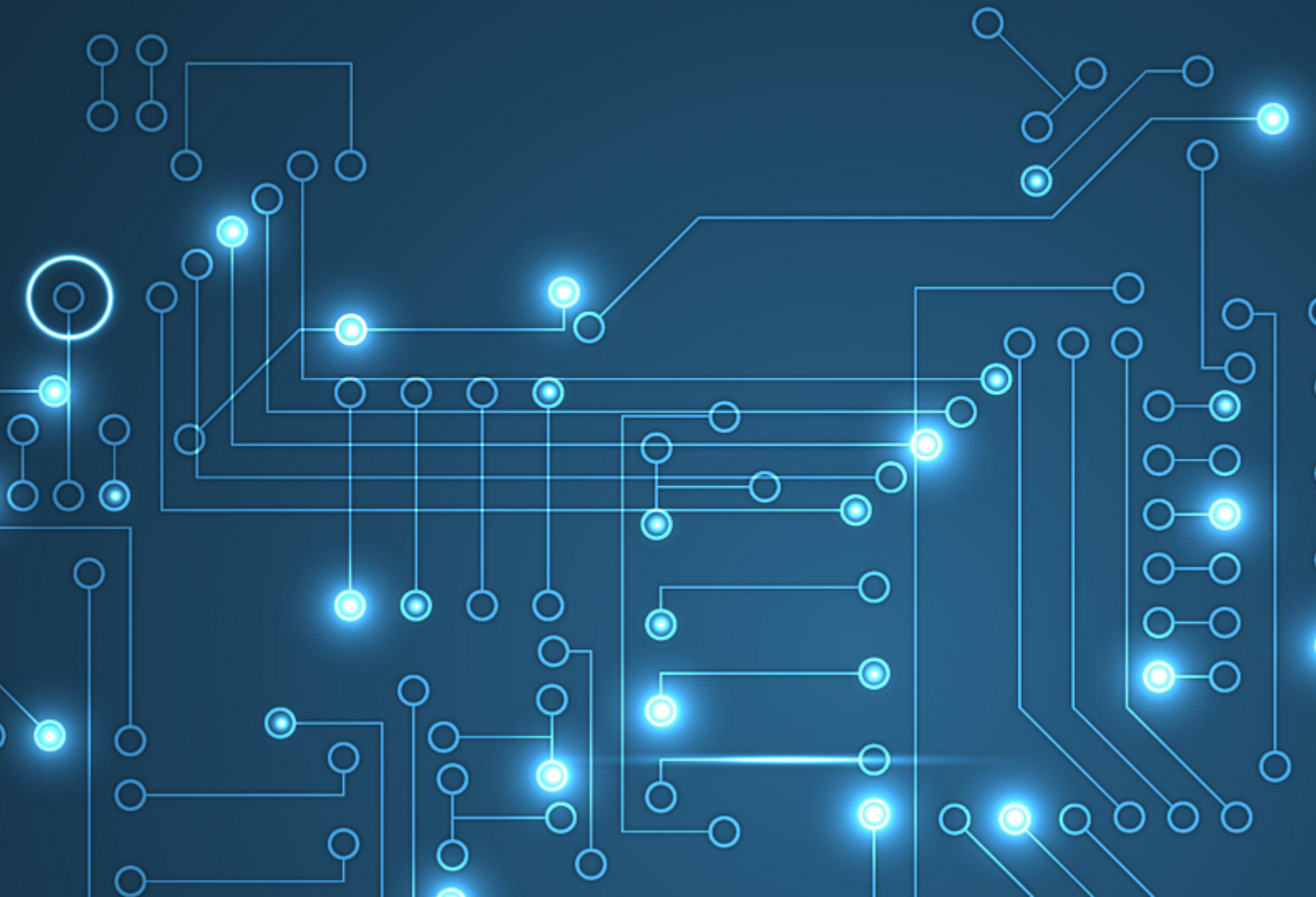
A companion guide to the text book:

A Practical Introduction to the Xilinx Zynq-7000 Adaptive SoC

Author: Derek Murray

Version: 1.0

Date: 29/8/21



Revision History

Version	Date	Comment
1.0	29/8/21	First version.

Table 1. Revision History

Limit of Liability/Disclaimer of Warranty: The author makes no representation or warranty with respect to the accuracy or completeness of the contents of this work and specifically disclaims all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. If improperly wired, circuits described in this work may possibly cause damage to the device and physical injury. The author shall not be liable for damages arising herefrom. The fact that an organisation or website is referred to in this work as a citation and/or a potential source of further information does not mean that the author endorses the information the organisation or website may provide or recommendations it may make. Further, readers should be aware that Internet websites listed in this work may have changed or disappeared between when this work was written and when it is read.

1 Introduction

This is a support document for the text book "A Practical Introduction to the Xilinx Zynq-7000 Adaptive SoC". Both Xilinx SDK and the Vitis unified software platform can be used to develop the projects in the text — this document targets the Vitis IDE, and it can be used with Versions 2020.2 and Version 2021.1. (Note: version 2020.3 is not strictly supported, although most of the steps are compatible. Earlier versions are not recommended.)

1.1 Xilinx Software

At the time of writing, the Vitis Core Development can be downloaded from the following location: https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vitis.html

Make sure to select the ***Vitis (SW) Developer*** tab on the download page. Note that Vivado is included in the core installer, and does not have to be downloaded separately.

1.2 Guide Layout

Section 2 of this document provides a step-by-step guide to building the hardware project in the main book, and the design details can be found in Chapter 6 of the text.

Section 3 of this document shows how to recreate the software projects in the textbook. It does not cover the technical details of each project; the book will need to be referred to for complete details. Table 2 shows the correlation between the software projects in the textbook and the related section in this guide.

Software Project	Book Chapter	Section in this Guide
1	7	Section 3.1
2	8	Section 3.5
3	9	Section 3.6
4	10	Section 3.7
5	11	Section 3.8
6	12	Section 3.9
7	13	Section 3.10
8	14	Section 3.11
9	16	Section 3.12
10	18	Section 3.13

Table 2. Correlation between the software projects in the textbook and the relevant section in this guide.

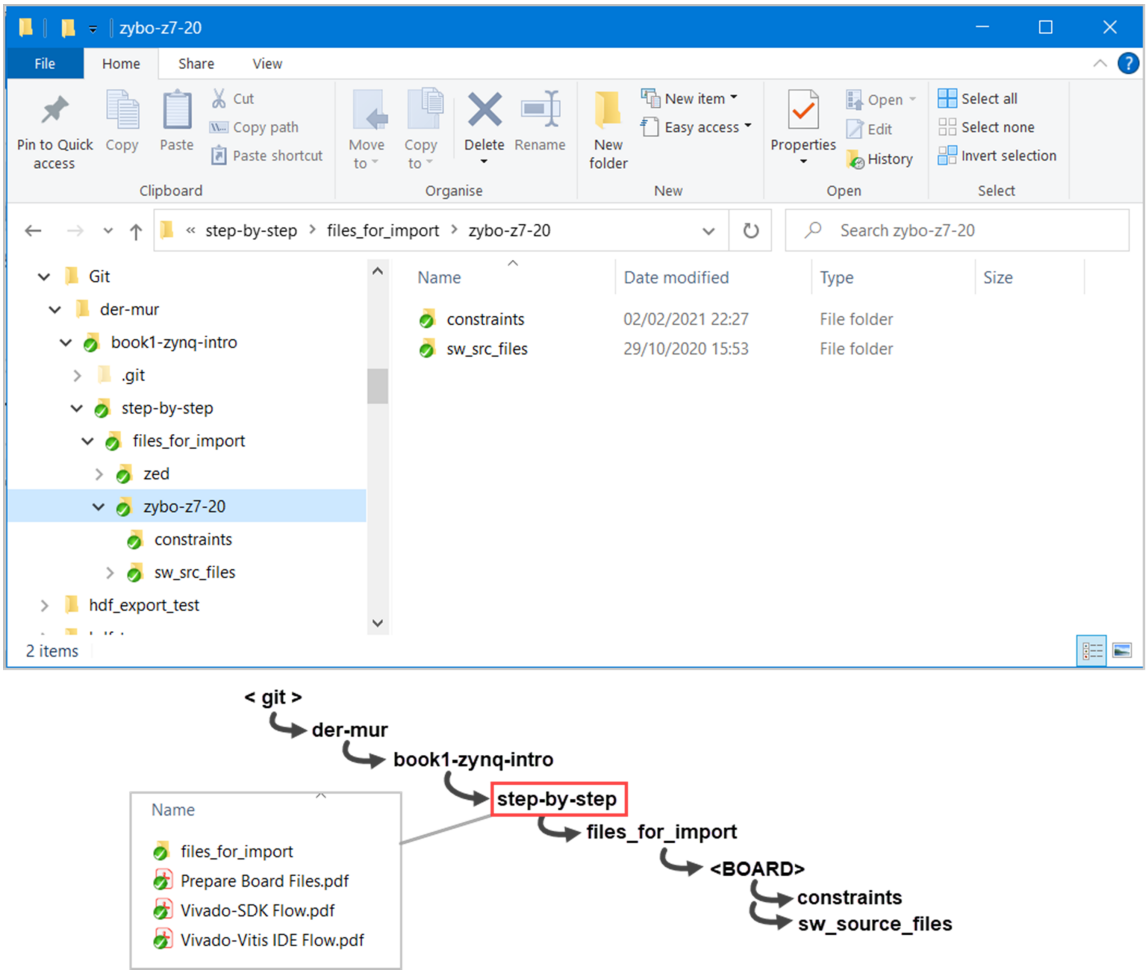


Figure 1. GitHub layout for the support material.

1.3 Accompanying Material

To complete the hardware and software projects in the text book (and in this guide), some additional resources are required. Constraint files are needed for the FPGA design, and C source files are needed for the software projects — currently, these can be found at the authors GitHub repository:

- https://github.com/der-mur/book1-zynq-intro

Figure 1 illustrates the GitHub layout. The Digilent Zybo-Z7-20 is the main developmental platform used in the text book, although the related resources are also directly compatible with the Zybo-Z7-10. In addition, C code and constraint files have been added for the Digilent/Avnet ZedBoard. If using the Zybo-Z7-20 (or -10), the board set-up for each project can be found directly in the textbook. The corresponding ZedBoard details can be seen in Table 3, Table 4, and Table 5.

IP Integrator/ XDC Name	Zybo-Z7-20 Board Interface	Zybo-Z7-20 Pin	ZedBoard Board Interface	ZedBoard Pin
TTC0_WAVE0_OUT	Pmod JC- Pin 1	V15	Pmod JB- Pin 1	W12
TTC0_WAVE1_OUT	Pmod JC- Pin 2	W15	Pmod JB- Pin 2	W11
TTC0_WAVE2_OUT	Pmod JC- Pin 3	T11	Pmod JB- Pin 3	V10
TTC1_WAVE0_OUT	Pmod JC- Pin 7	W14	Pmod JB- Pin 7	V12
TTC1_WAVE1_OUT	Pmod JC- Pin 8	Y14	Pmod JB- Pin 8	W10
TTC1_WAVE2_OUT	Pmod JC- Pin 9	T12	Pmod JB- Pin 9	V9
cs_n	Pmod JD- Pin 1	T14	Pmod JD- Pin 1	V7
mosi	Pmod JD- Pin 2	T15	Pmod JD- Pin 2	W7
miso	Pmod JD- Pin 3	P14	Pmod JD- Pin 3	V5
sck	Pmod JD- Pin 4	R14	Pmod JD- Pin 4	V4
PMOD_ACL_INT2	Pmod JD- Pin 7	U14	Pmod JD- Pin 7	W6
PMOD_ACL_INT1	Pmod JD- Pin 8	U15	Pmod JD- Pin 8	W5
gpio_out[4]	Pmod JE- Pin 1	V12	Pmod JC- Pin 1	AB7
gpio_out[5]	Pmod JE- Pin 2	W16	Pmod JC- Pin 2	AB6
gpio_out[6]	Pmod JE- Pin 3	J15	Pmod JC- Pin 3	Y4
gpio_out[7]	Pmod JE- Pin 4	H15	Pmod JC- Pin 4	AA4
gpio_out[0]	LD0	M14	LD0	T22
gpio_out[1]	LD1	M15	LD1	T21
gpio_out[2]	LD2	G14	LD2	U22
gpio_out[3]	LD3	D18	LD3	U21
gpio_in[8]	Pmod JE- Pin 7	V13	Pmod JC- Pin 7	R6
gpio_in[9]	Pmod JE- Pin 8	U17	Pmod JC- Pin 8	T6
gpio_in[10]	Pmod JE- Pin 9	T17	Pmod JC- Pin 9	T4
gpio_in[11]	Pmod JE- Pin 10	Y17	Pmod JC- Pin 10	U4
gpio_in[4]	SW0	G15	SW0	F22
gpio_in[5]	SW1	P15	SW1	G22
gpio_in[6]	SW2	W13	SW2	H22
gpio_in[7]	SW3	T16	SW3	F21
gpio_in[0]	BTN0	K18	BTNU	T18
gpio_in[1]	BTN1	P16	BTNR	R18
gpio_in[2]	BTN2	K19	BTND	R16
gpio_in[3]	BTN3	Y16	BTNL	N15

Table 3. Comparison of Zybo-Z7-20 and ZedBoard board details

Project Function	Zybo-Z7-20 Board Interface	ZedBoard Board Interface
gpio_out[0-3]	LD0 - LD3	LD0 - LD3
gpio_out[4-7]	Pmod JE [1-4]	Pmod JC [1-4]
gpio_in[0]	BTN0	BTNU
gpio_in[1]	BTN1	BTNR
gpio_in[2]	BTN2	BTND
gpio_in[3]	BTN3	BTNL
gpio_in[4-7]	SW0 - SW3	SW0 - SW3
gpio_in[8-11]	Pmod JE [7-10]	Pmod JC [7-10]
TTC0 WAVE Outputs	Pmod JC	Pmod JB
Pmod ACL	Pmod JD	Pmod JD

Table 4. Summary of Zybo-Z7-20 and ZedBoard interface details

MIO	Zybo-Z7-20 Board Interface	ZedBoard Board Interface
7	LD4	LD9
13	Pmod JF - Pin 1	Pmod JE - Pin 1
10	Pmod JF - Pin 2	Pmod JE - Pin 2
11	Pmod JF - Pin 3	Pmod JE - Pin 3
12	Pmod JF - Pin 4	Pmod JE - Pin 4
0	Pmod JF - Pin 7	Pmod JE - Pin 7
9	Pmod JF - Pin 8	Pmod JE - Pin 8
14	Pmod JF - Pin 9	Pmod JE - Pin 9
15	Pmod JF - Pin 10	Pmod JE - Pin 10
50	BTN4	BTN8
51	BTN5	BTN9

Table 5. MIO interface details for the Zybo-Z7-20 and the ZedBoard

1.4 Required Hardware

- Digilent Zybo-Z7-20 (or Zybo-Z7-10).
- The ZedBoard can also be used, as mentioned earlier.
- Digilent Pmod TPH2. (Ideally, three are required.)

- A 16-channel Logic Analyser. (The Digilent Analog Discovery 2 is used in the text, although any suitable logic analyser can be used.)

1.5 Required Software

A terminal emulator such as Tera Term is required when running the software projects in Section 3. (Note that the Xilinx development tools also have a built-in terminal which can be used.)

1.6 Possible Directory Layout

For readers who do not already have a directory structure in place for project development, Figure 2 shows the layout favoured by the author.

- **Level 1**: A base directory called *zynq_proj*
- **Level 2**: The current tool version (2017_4, 2021_1, etc.)
- **Level 3**: The platform (Zybo-Z7-20, Zybo-Z7-10, ZedBoard, etc.)

The directories for individual projects will then be automatically created in Vivado at the start of development; see Figure 6, for example, where the project for this guide is created as *hw_proj1*.

1.7 Setting Up the Zybo-Z7-20

If using the Zybo-Z7-20 (or Zybo-Z7-10), the reader can optionally follow the steps outlined in an additional document called "*Prepare Board Files.pdf*". (See Figure 1 for the location of this guide in GitHub.) This particular document is not essential, especially if the reader has already set up their platform correctly, although different warnings may appear when working through the steps in the current guide.

1.8 Getting started

1. Download and install Vivado/SDK (see Section 1.1 for software links).
2. Create a directory structure (see Section 1.6 for a possible layout).
3. Optionally, prepare the board files if using the Zybo-Z7-20 or Zybo-Z7-10 (see Section 1.7).
4. Load Vivado and go to Section 2.

A final disclaimer: while every effort has been made to ensure that the steps in this guide are accurate, it is a large document and minor discrepancies may exist (especially as different software versions are supported). If a step is missing, the default option is usually correct!

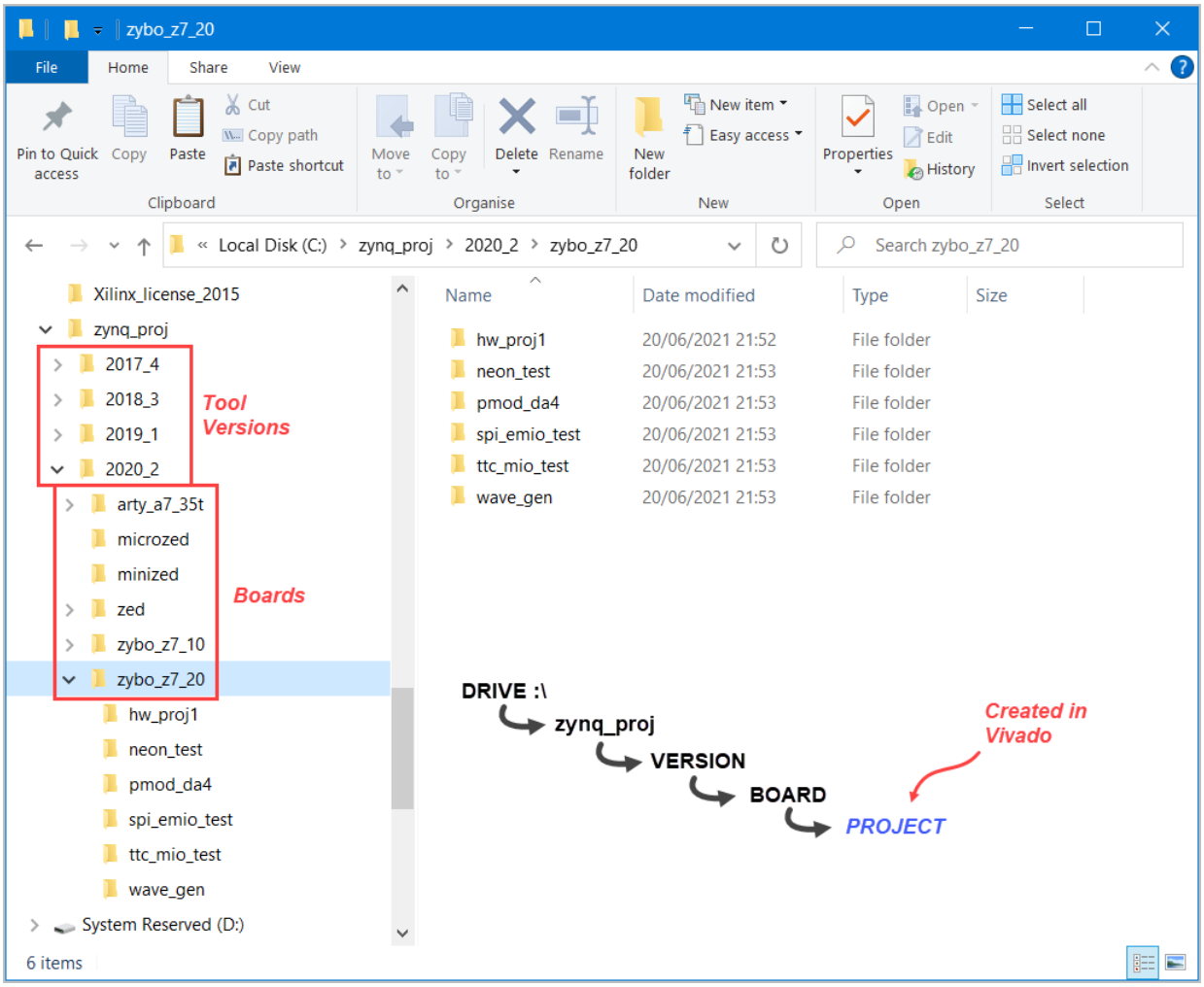


Figure 2. Possible Directory Structure

1.9 Useful Digilent links

- Installing Vivado, Vitis, and Digilent Board Files

1.10 Vitis 2021.1 Error

Note that if using Vitis 2021.1, the error shown in Figure 3 might be seen when building the software projects in Section 3. This occurs if the user downloads a new design to the platform while a previous project is still running. The solution is to power-cycle the board before running each project.

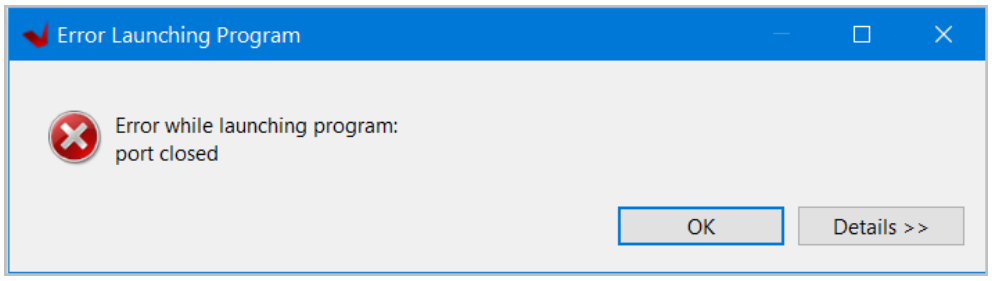


Figure 3. Vitis 2021.1 Error

2 Project Design in Vivado

2.1 Create the Project

Figure 4. When Vivado first launches, select Create Project.

Figure 5. Click Next to continue.

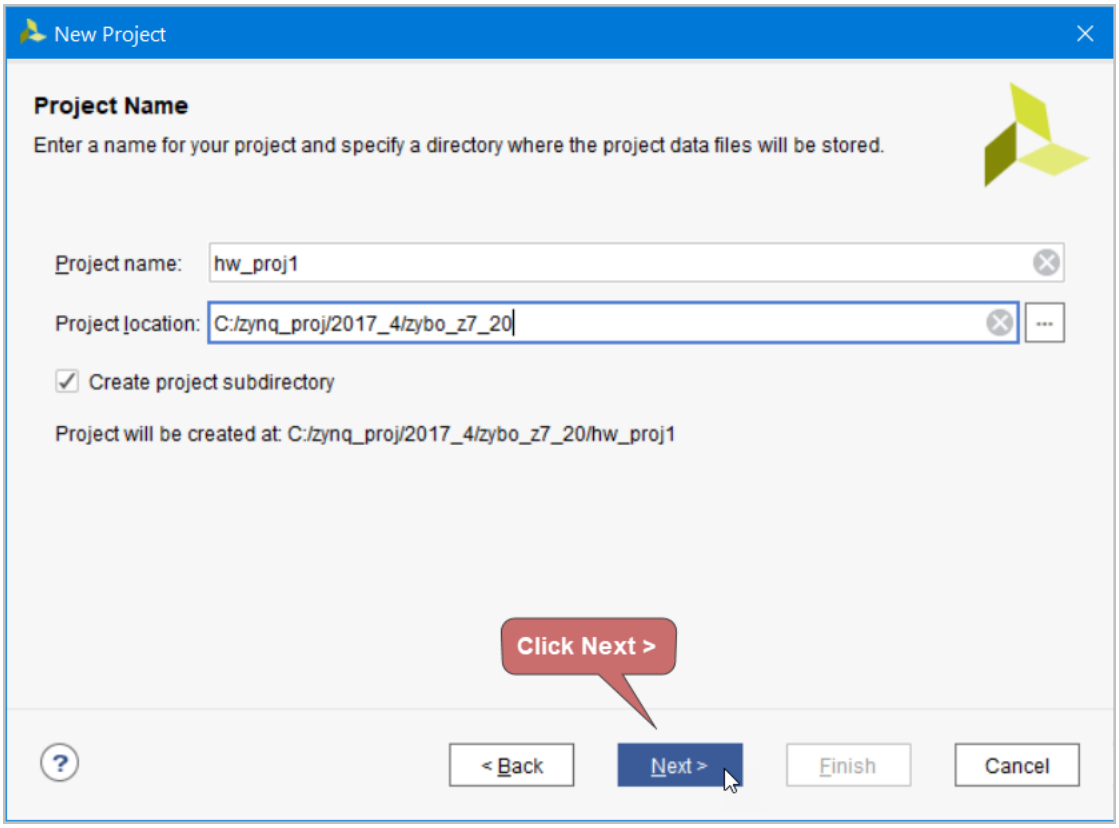


Figure 6. Choose a suitable project location and name ("hw_proj1" is used here).

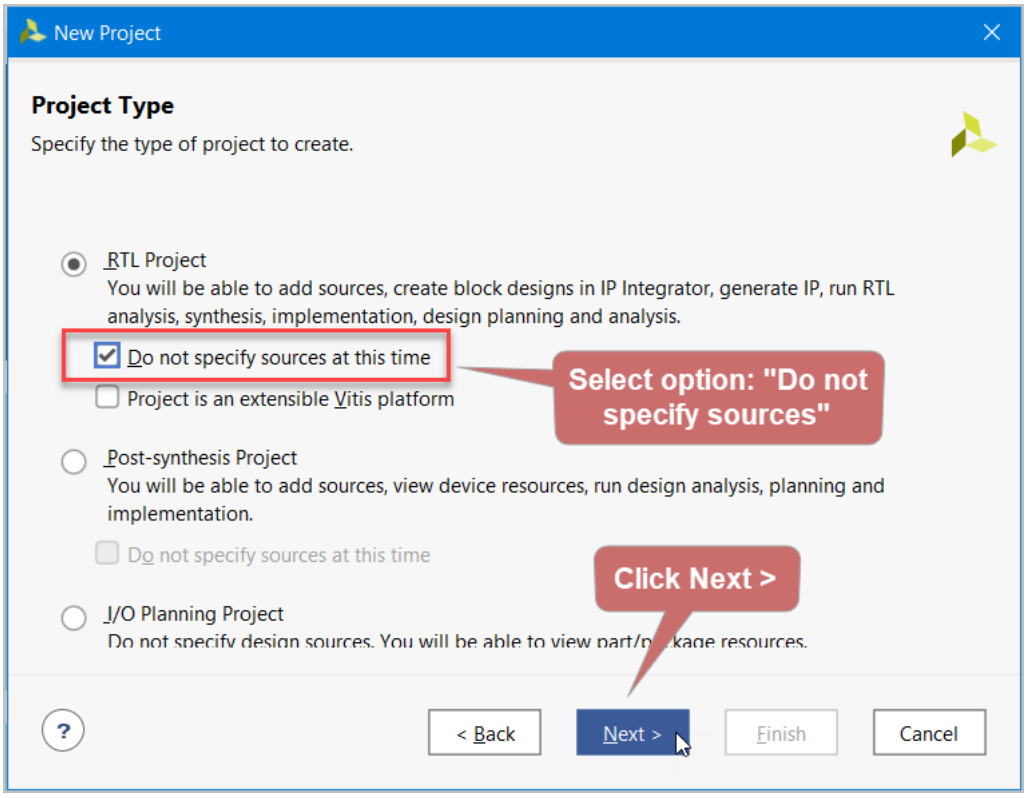


Figure 7. Select "Do not specify sources at this time" and click Next to continue.

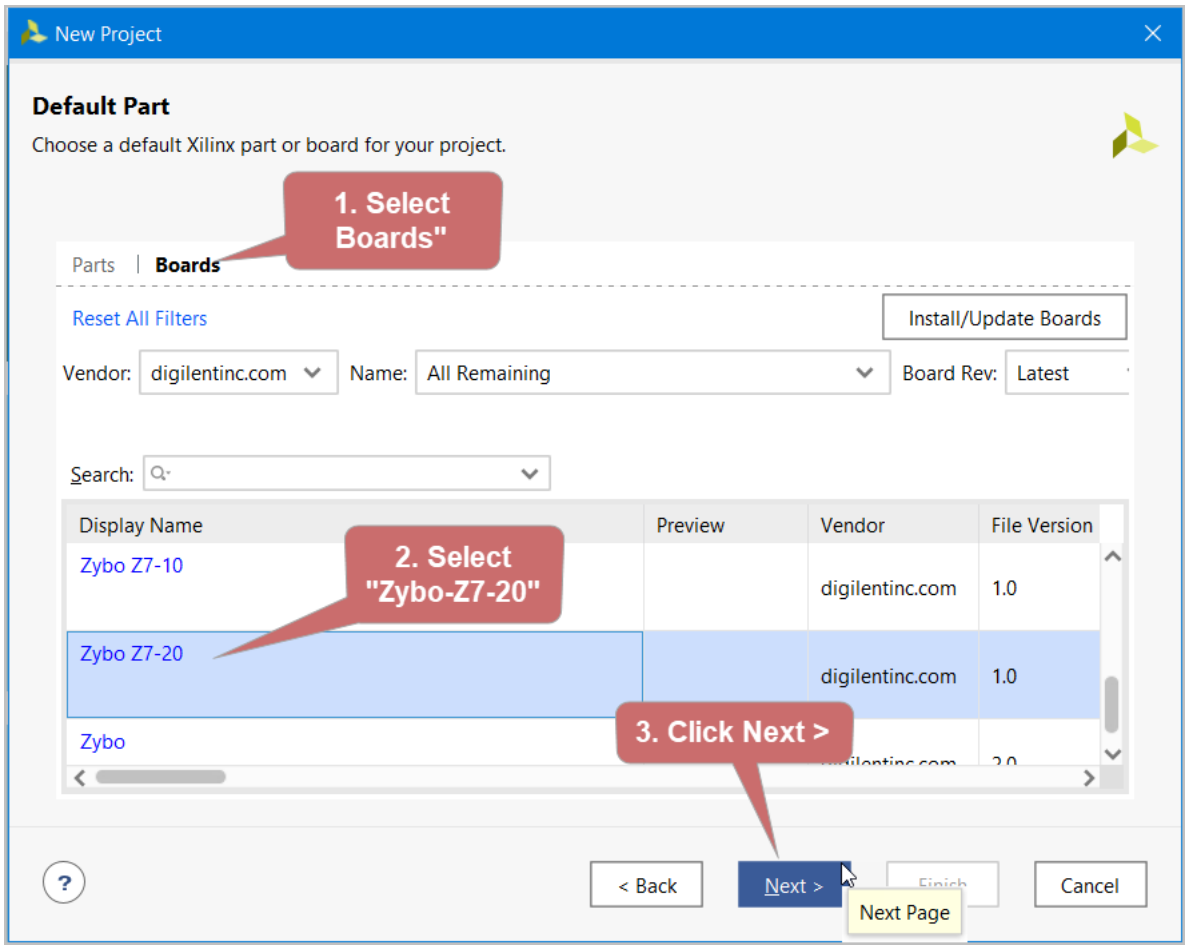


Figure 8. Choose the Digilent Zybo-Z7-20

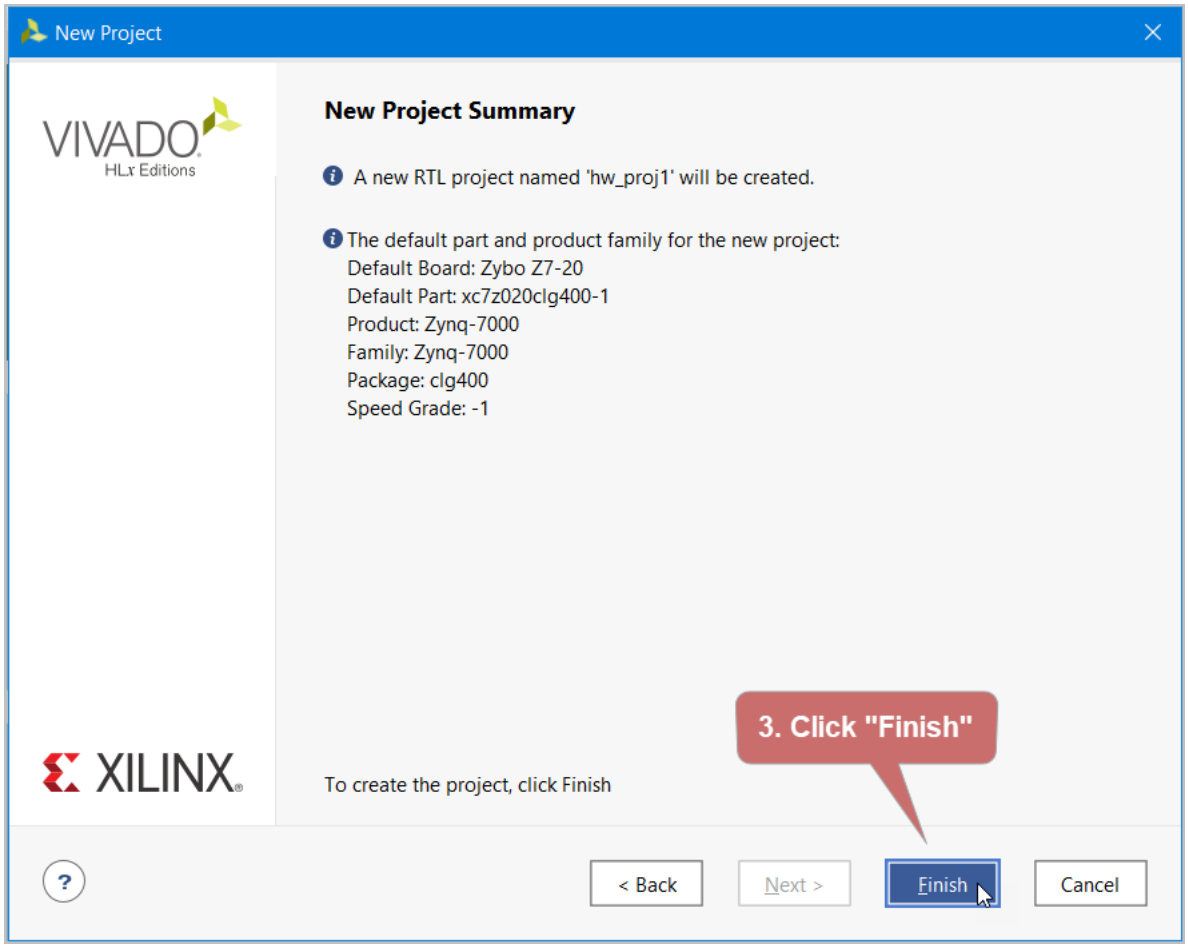


Figure 9. Click Finish.

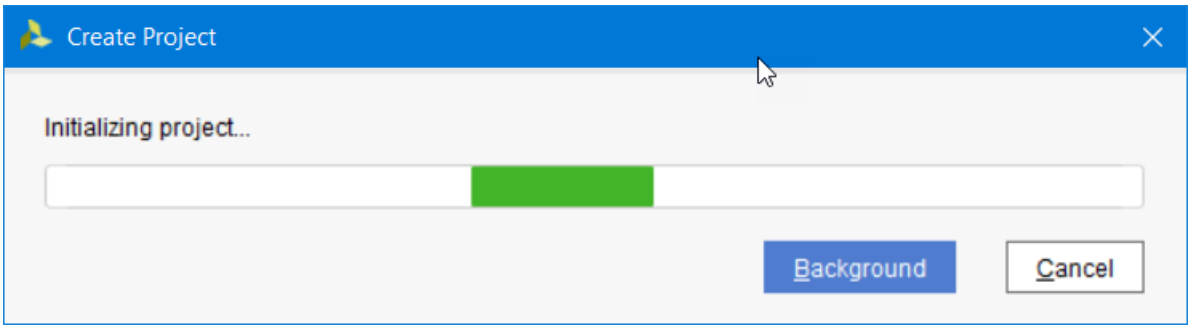


Figure 10. The project will be created...

2.2 Design Entry using IP Integrator

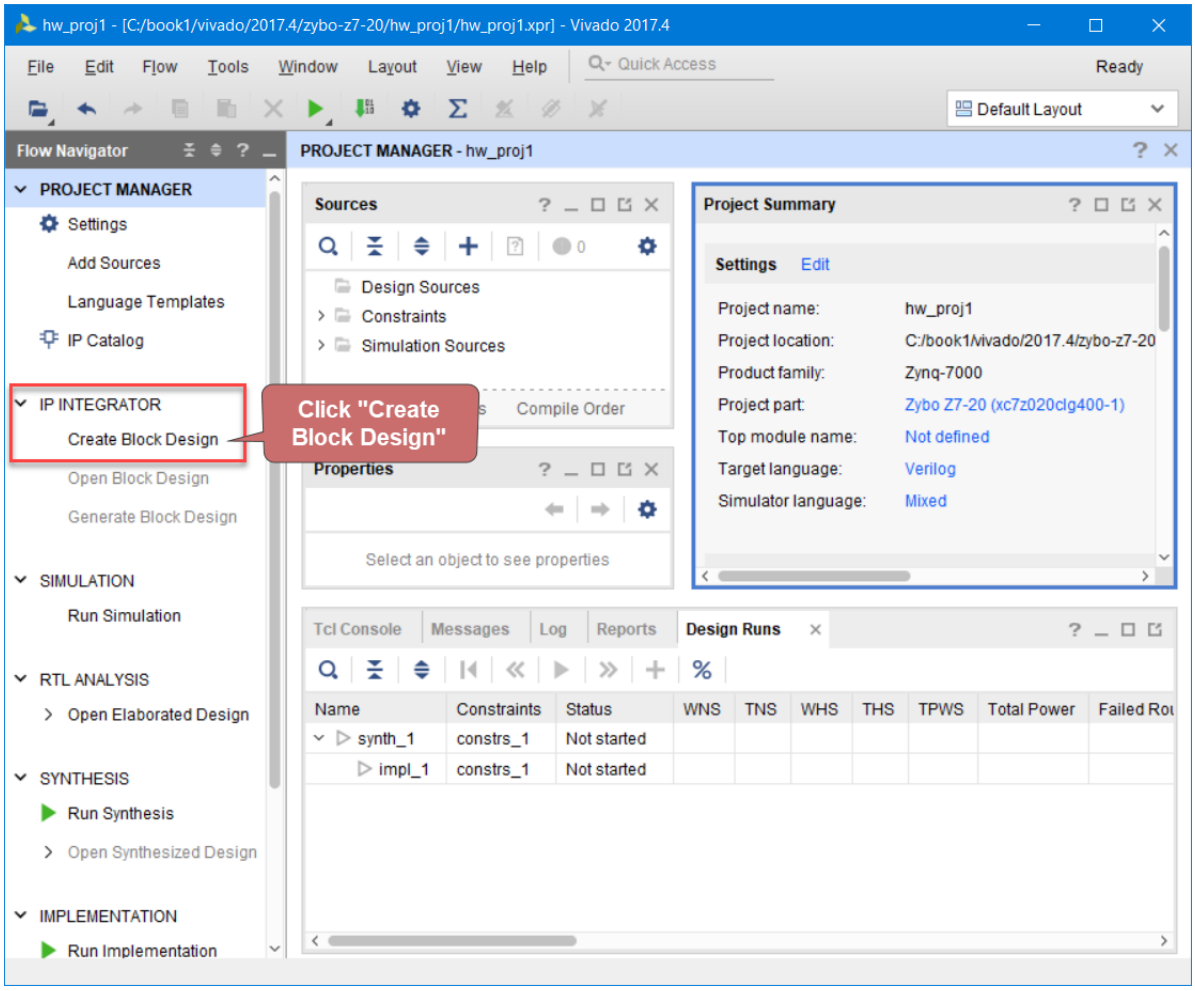


Figure 11. In the Flow Navigator, create the block design canvas.

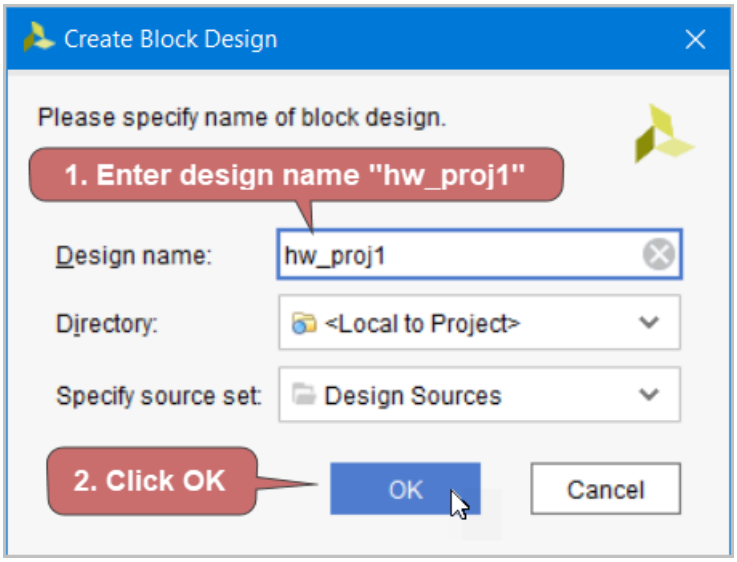


Figure 12. Enter the block design name.

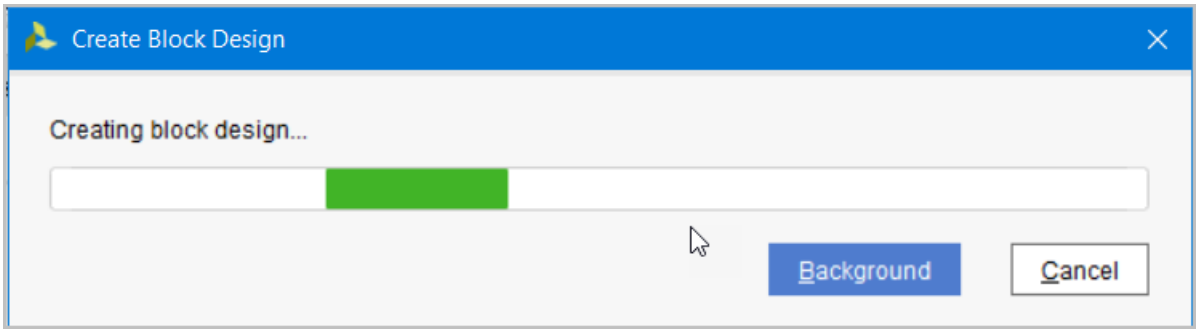


Figure 13. The block design will be created.

2.2.1 Add Zynq-7000 Processing System IP

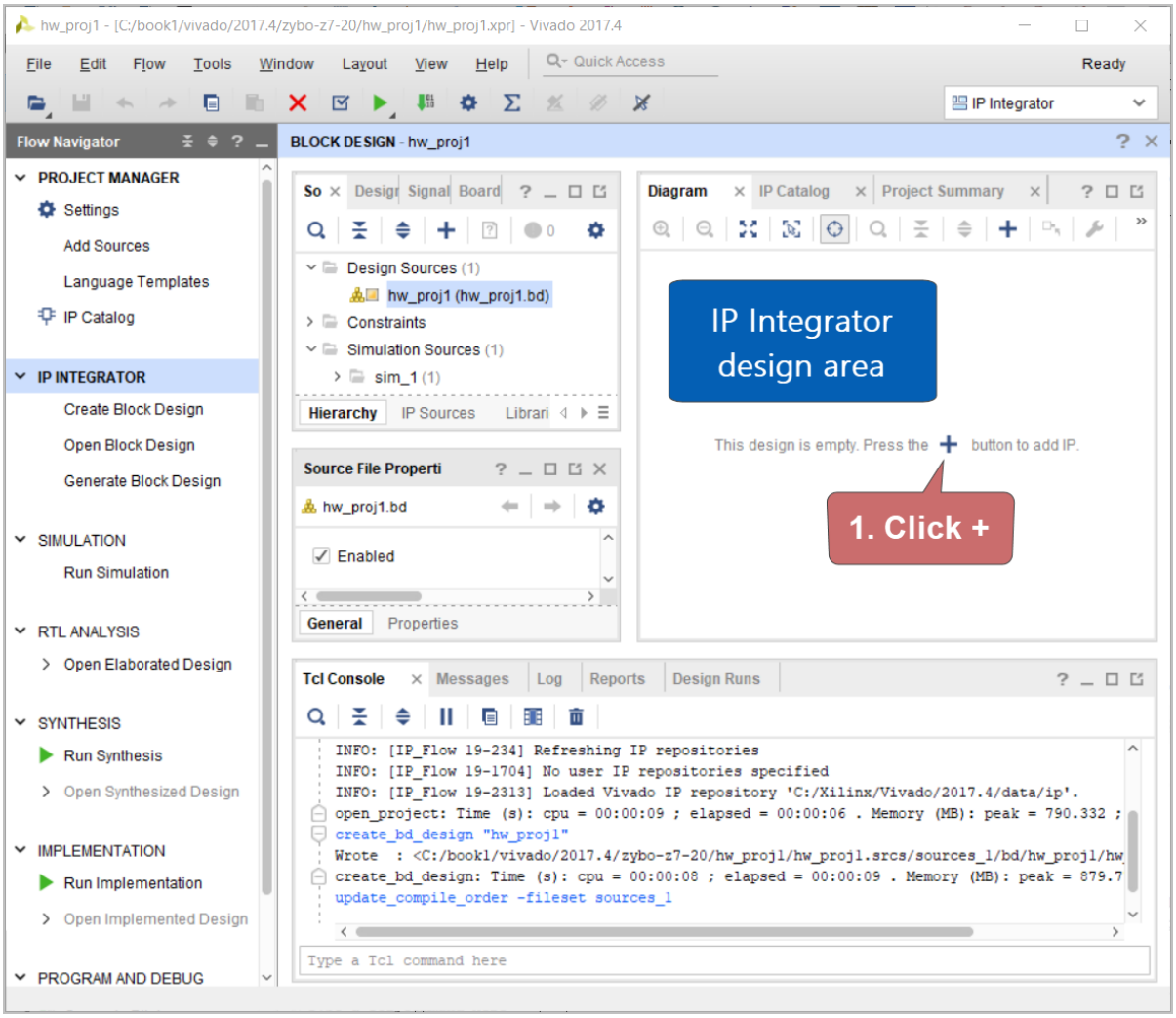


Figure 14. In the design area, click on the "+" icon, or select "CTRL + I"

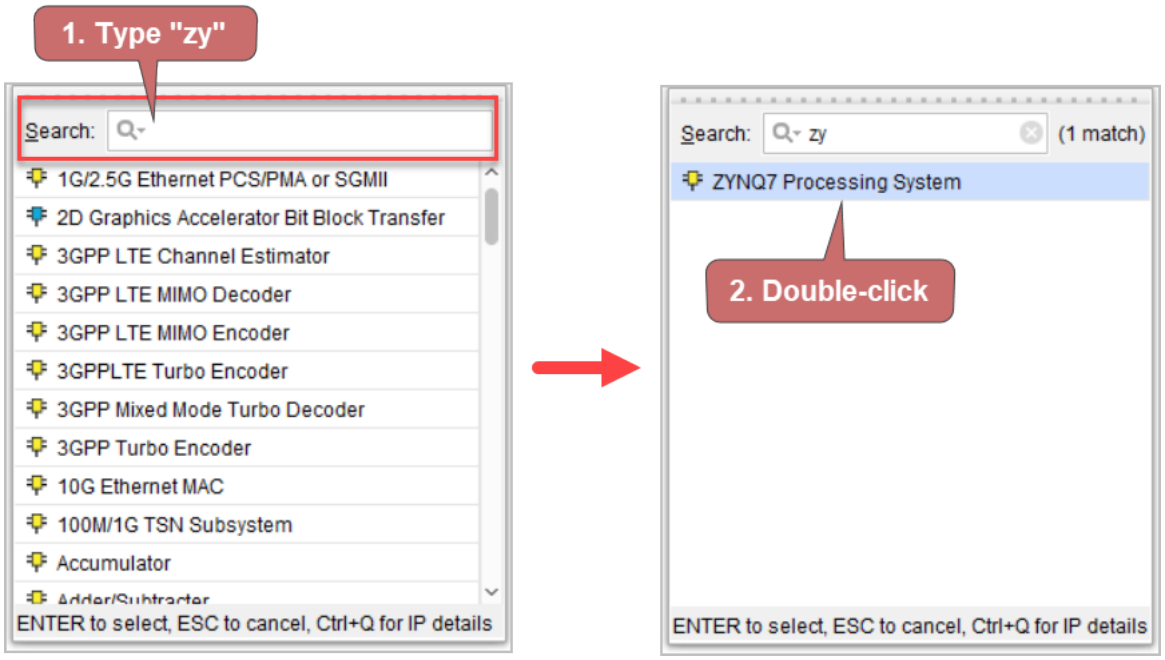


Figure 15. Find the Zynq-7000 Processing System IP

2.2.2 Generate Default Board Connections

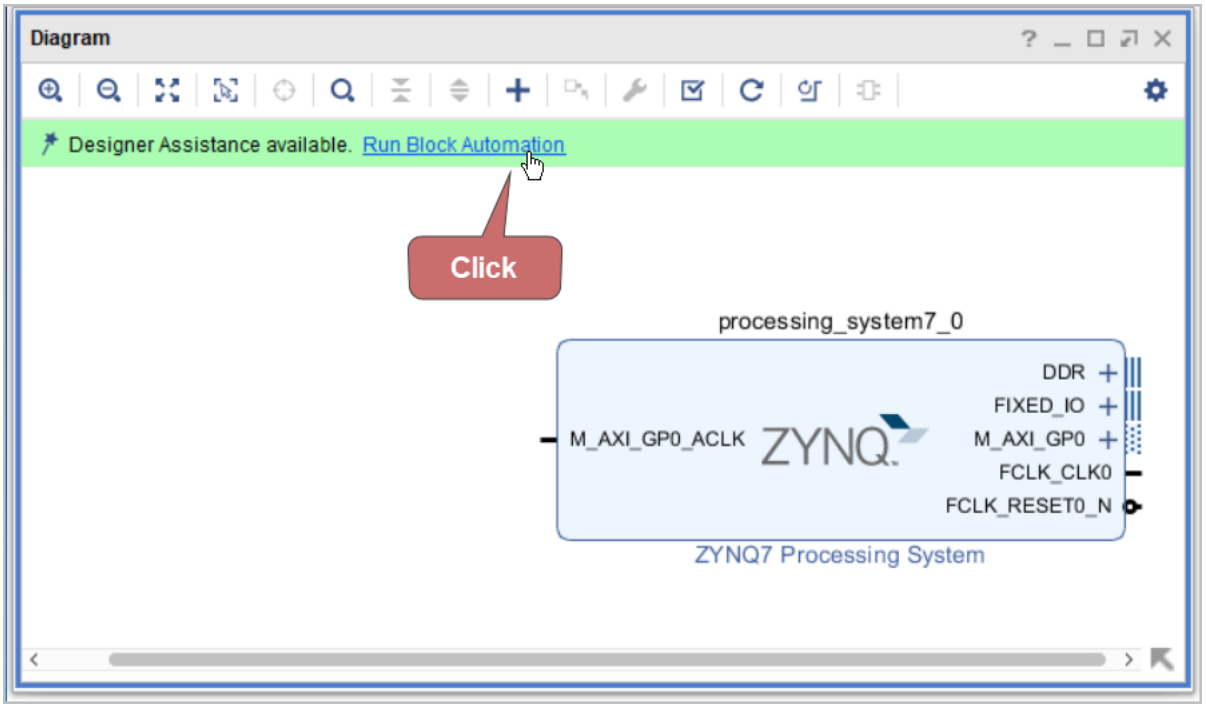


Figure 16. Click on the Run Block Automation option

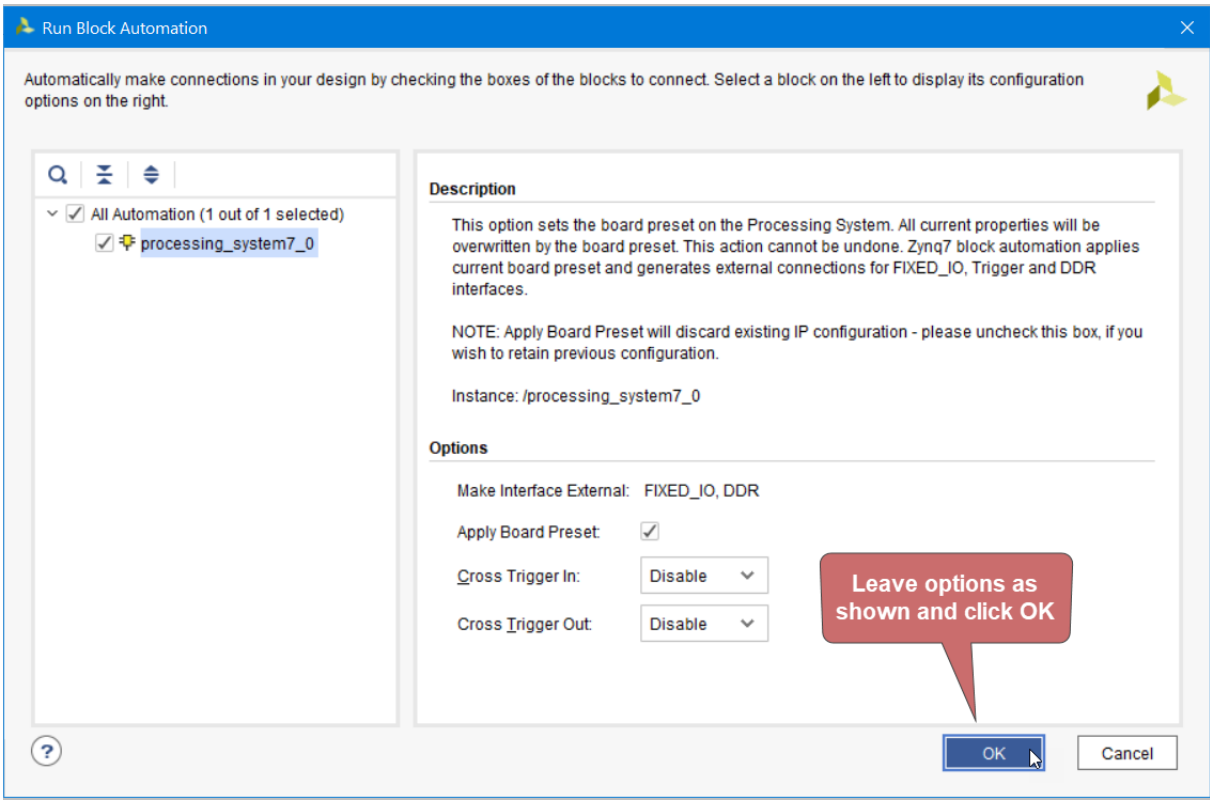


Figure 17. Select the default options and click OK.

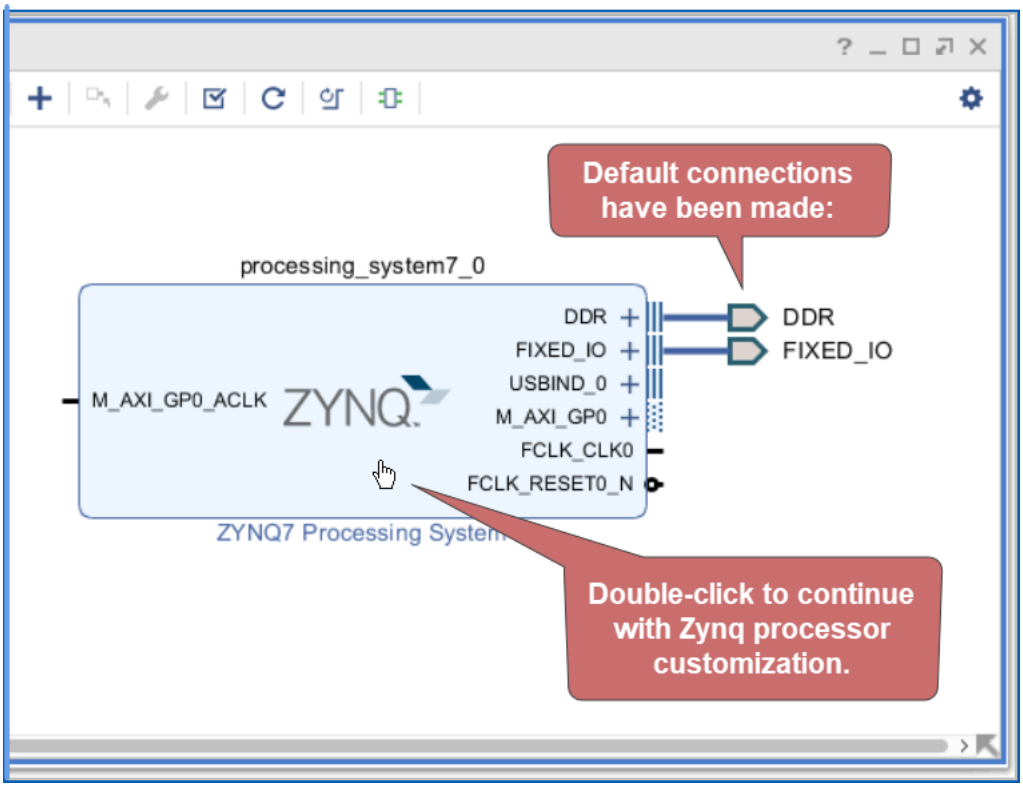


Figure 18. Double-click the Zynq IP to add more configuration options.

2.2.3 Customise the Processing System for our Project

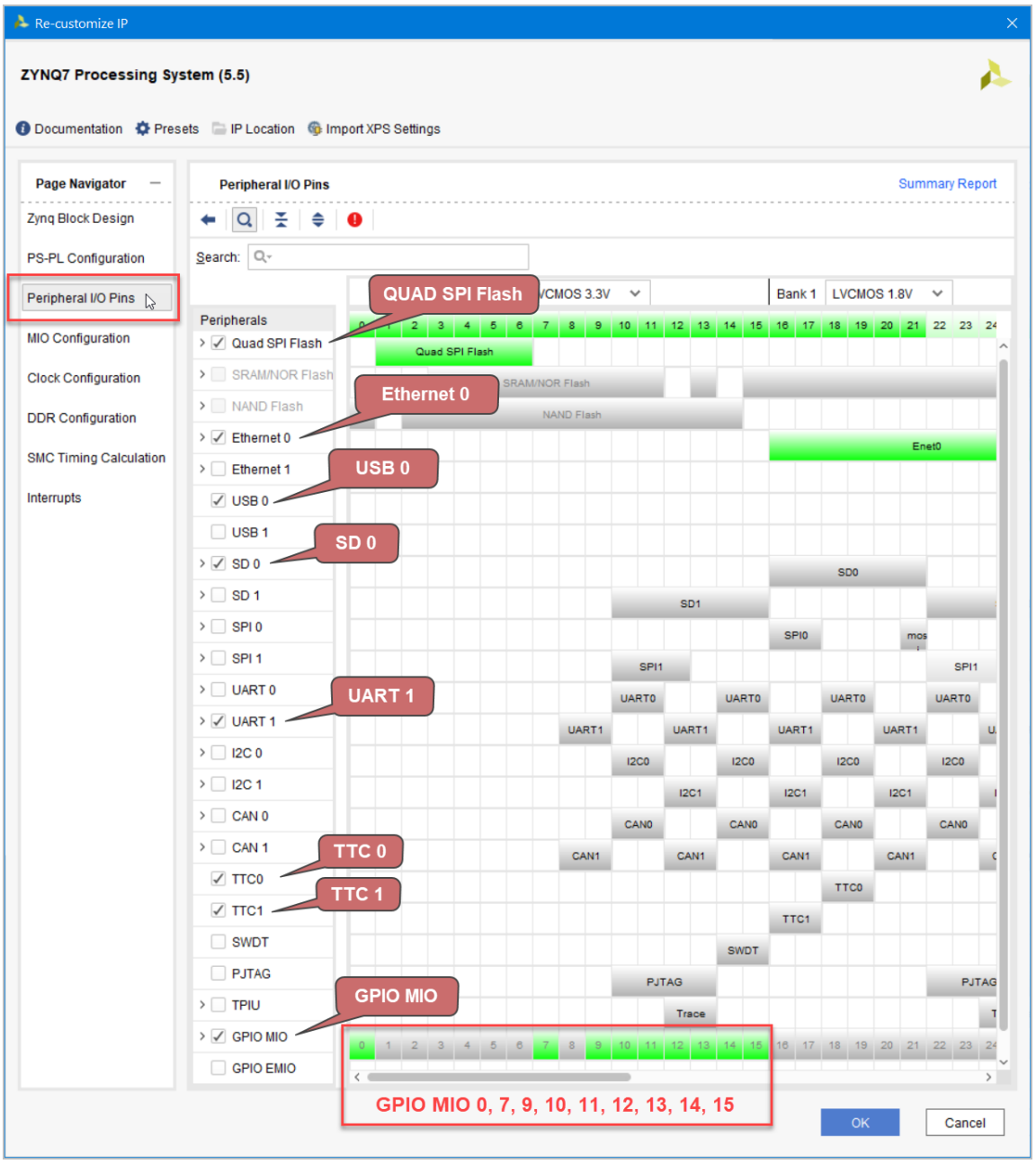


Figure 19. Peripheral I/O Pins (1). The settings should be as shown.

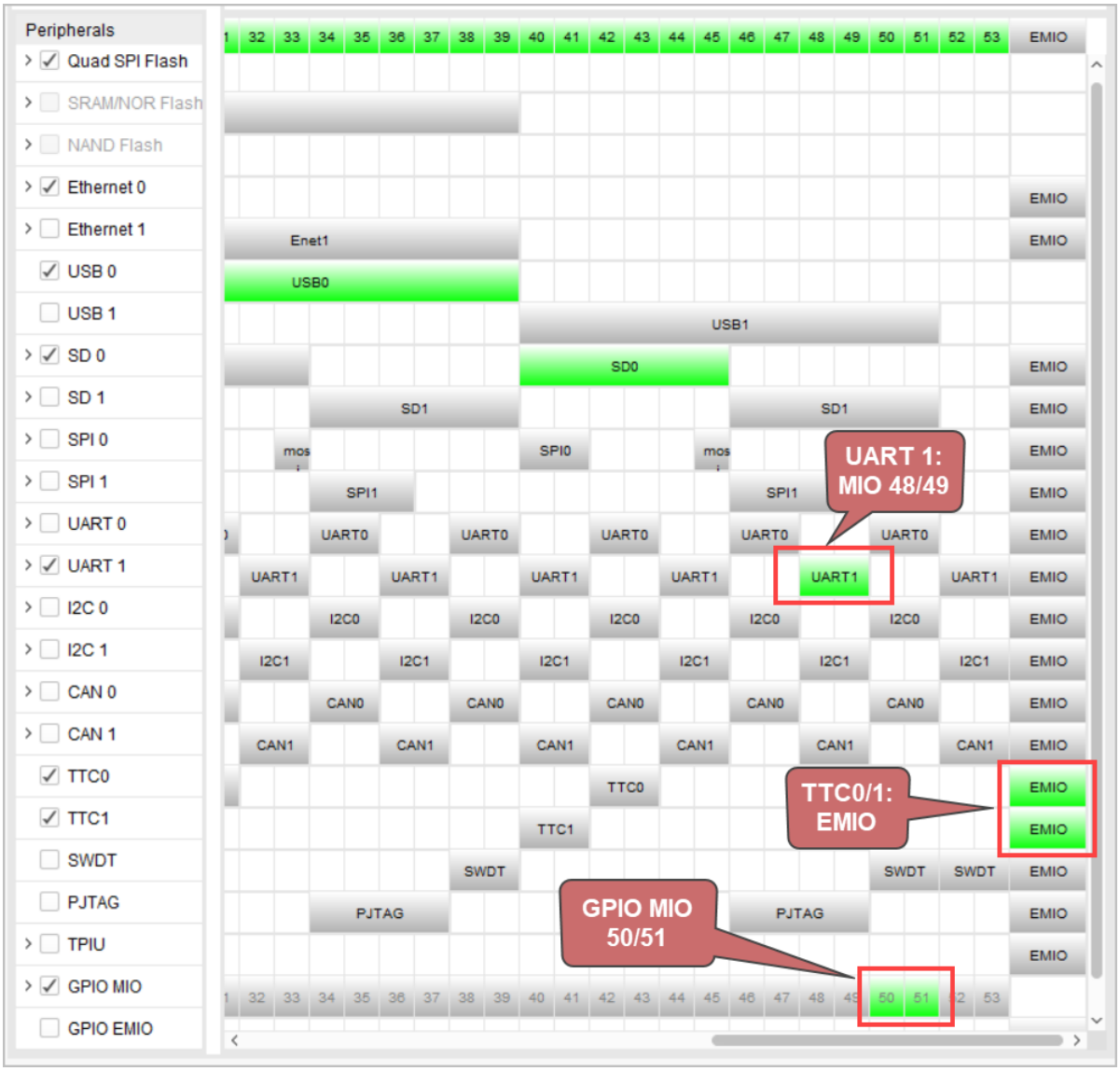


Figure 20. Enable TTC0 and TTC1 (the other settings should already be as shown).

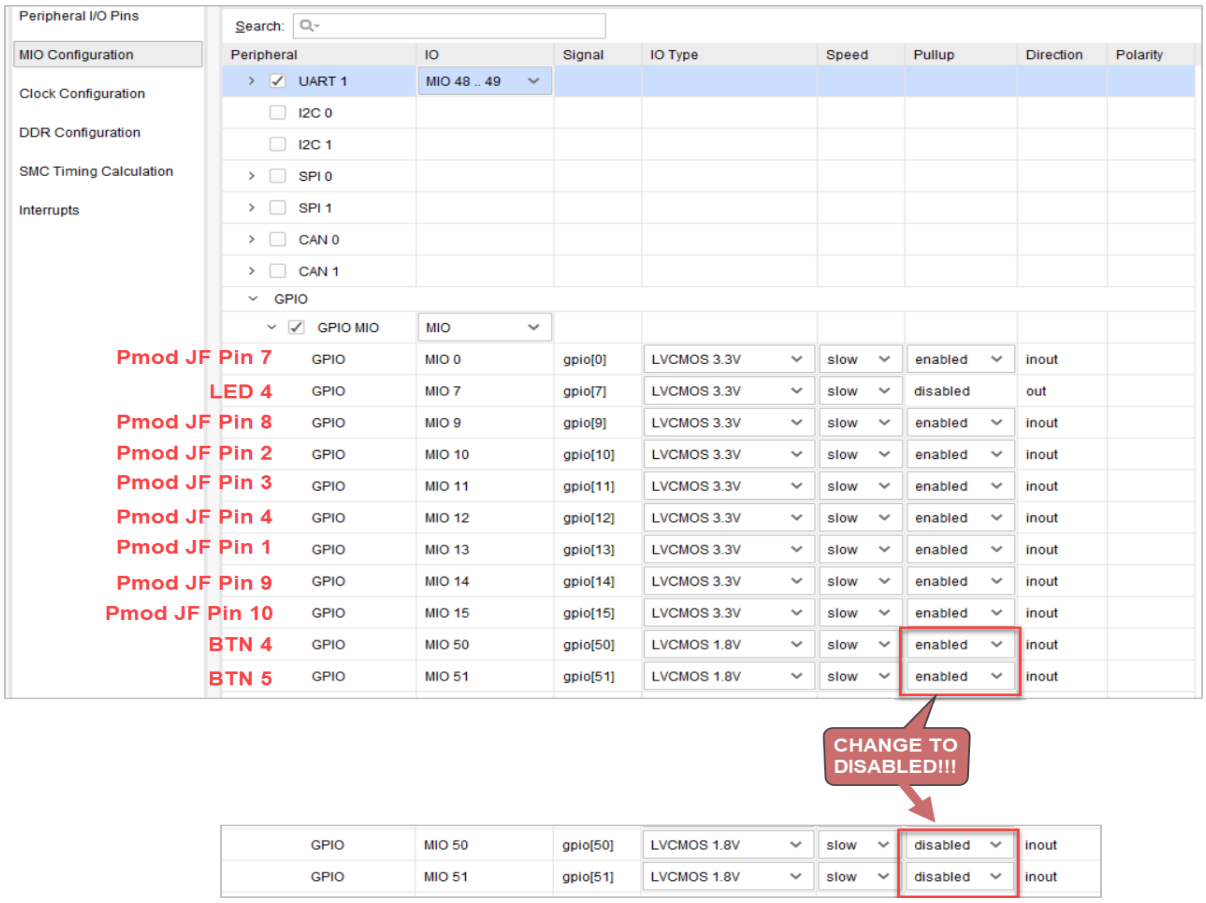


Figure 21. In the MIO Configuration Tab, the pull-ups for MIO50/MIO51 (BTN4/BTN5) can be disabled (if not already configured in the way).

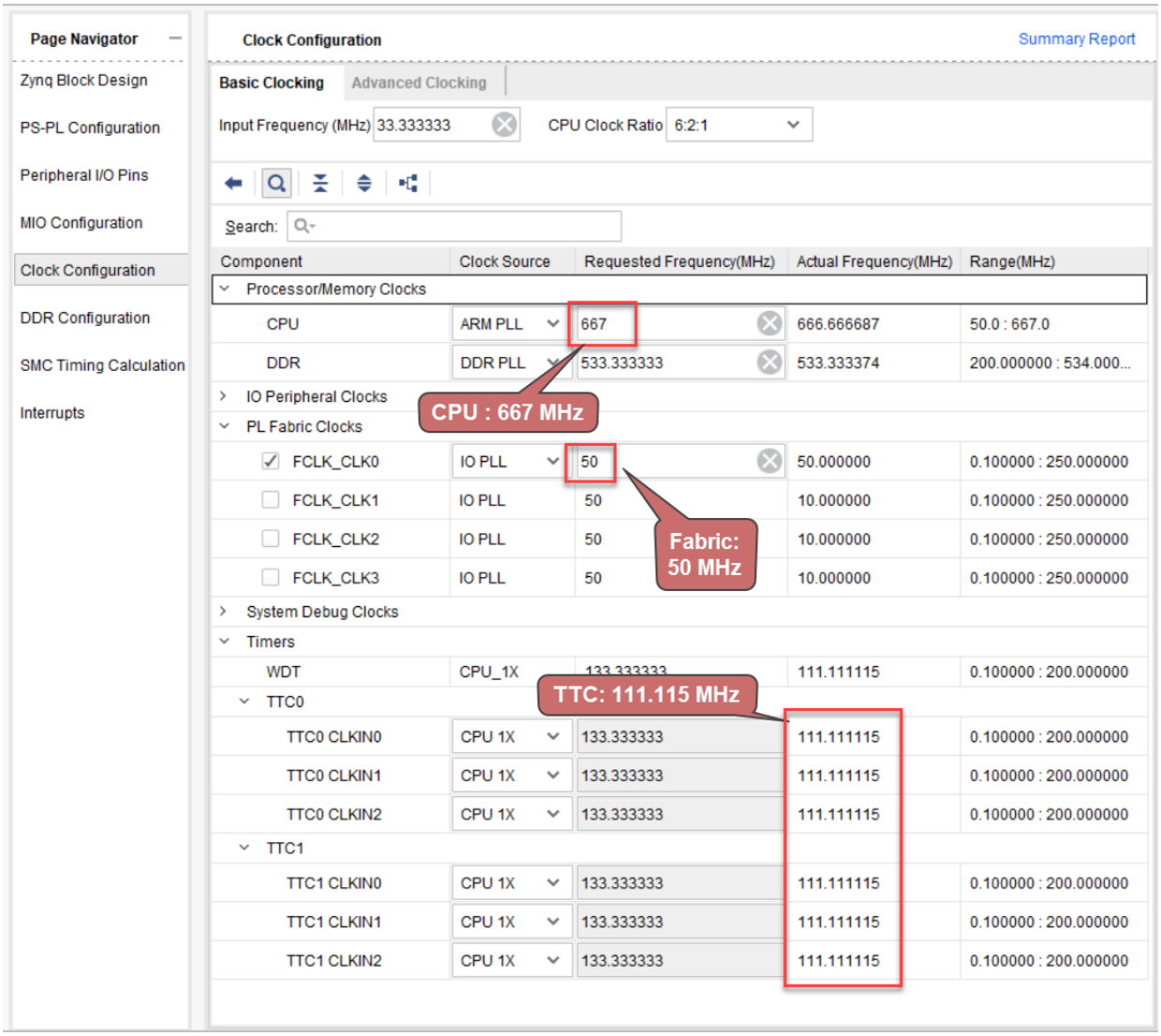


Figure 22. In the Clock Configuration Tab, verify that the clock settings are as shown.

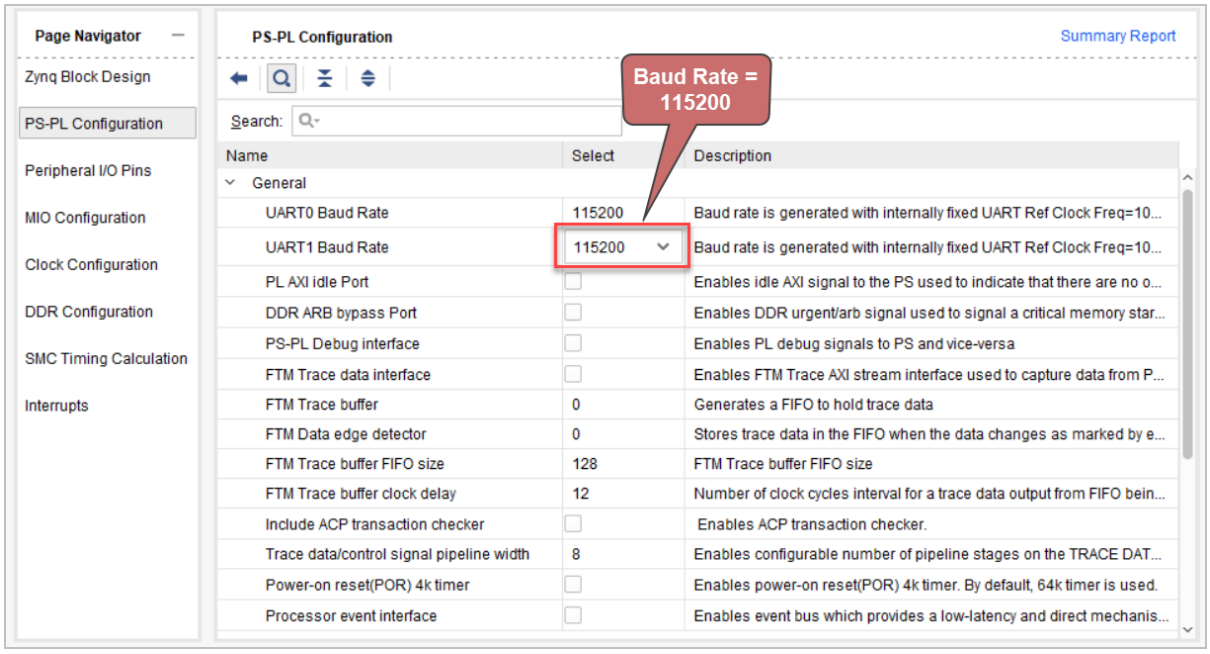


Figure 23. Check that the UART Baud Rate is 115200 baud.

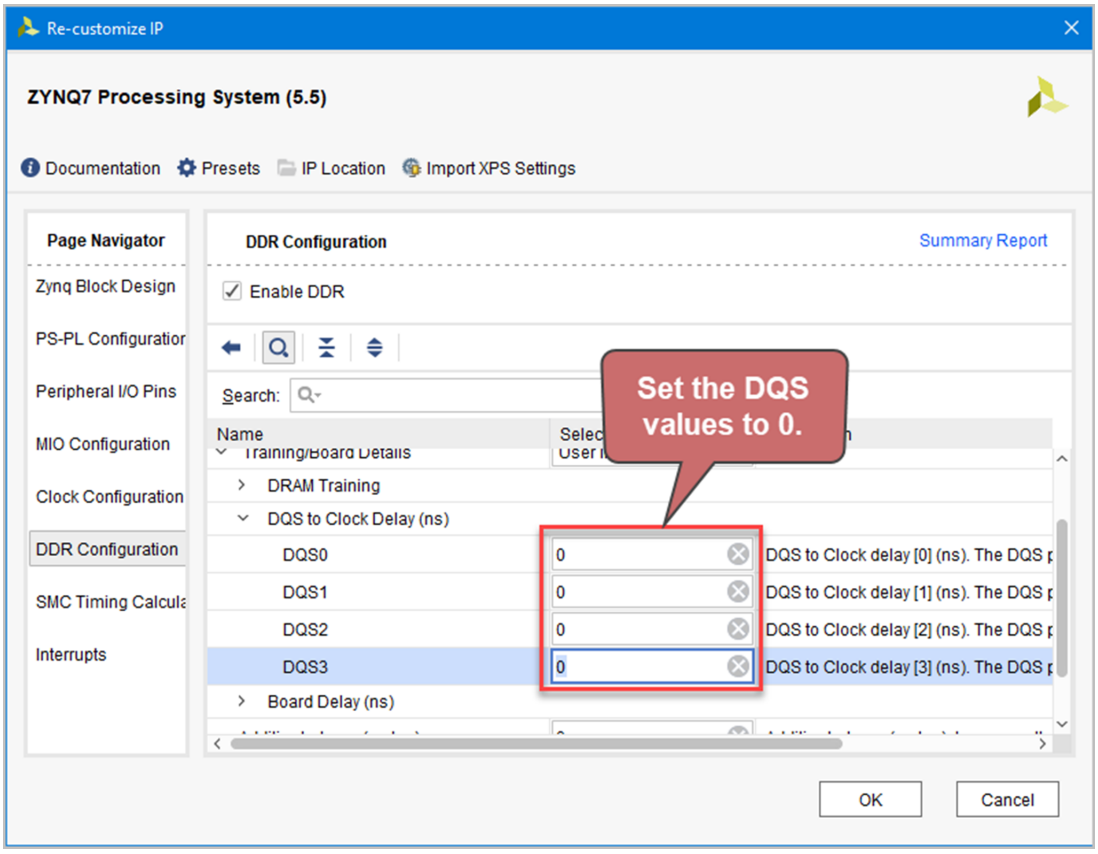


Figure 24. In the DDR Configuration tab, the DQS to Clock Delay settings can all be changed to 0.

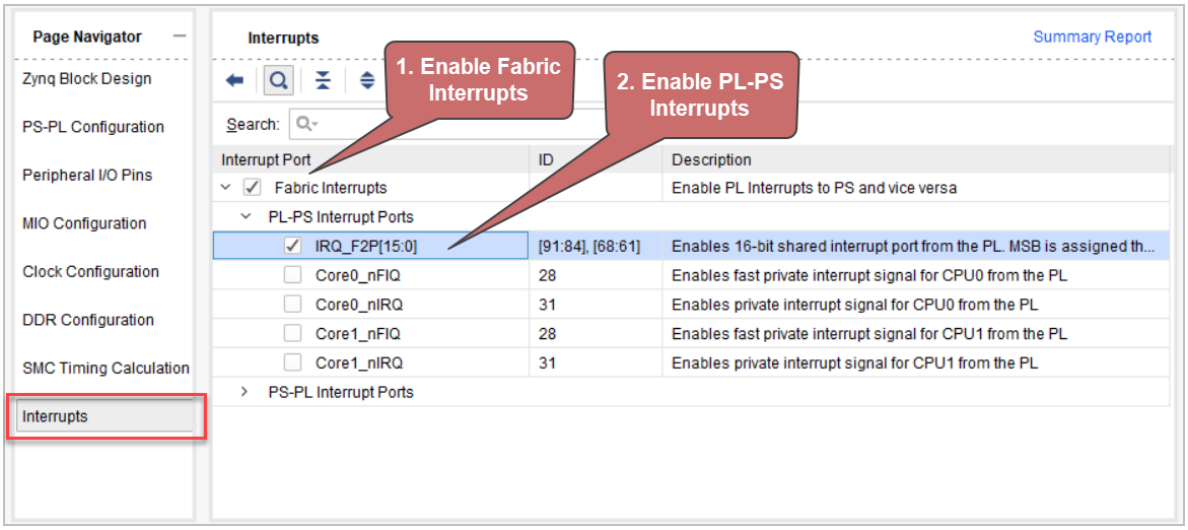


Figure 25. In the Interrupts tab, enable the IRQ_F2P[15:0] interrupts.

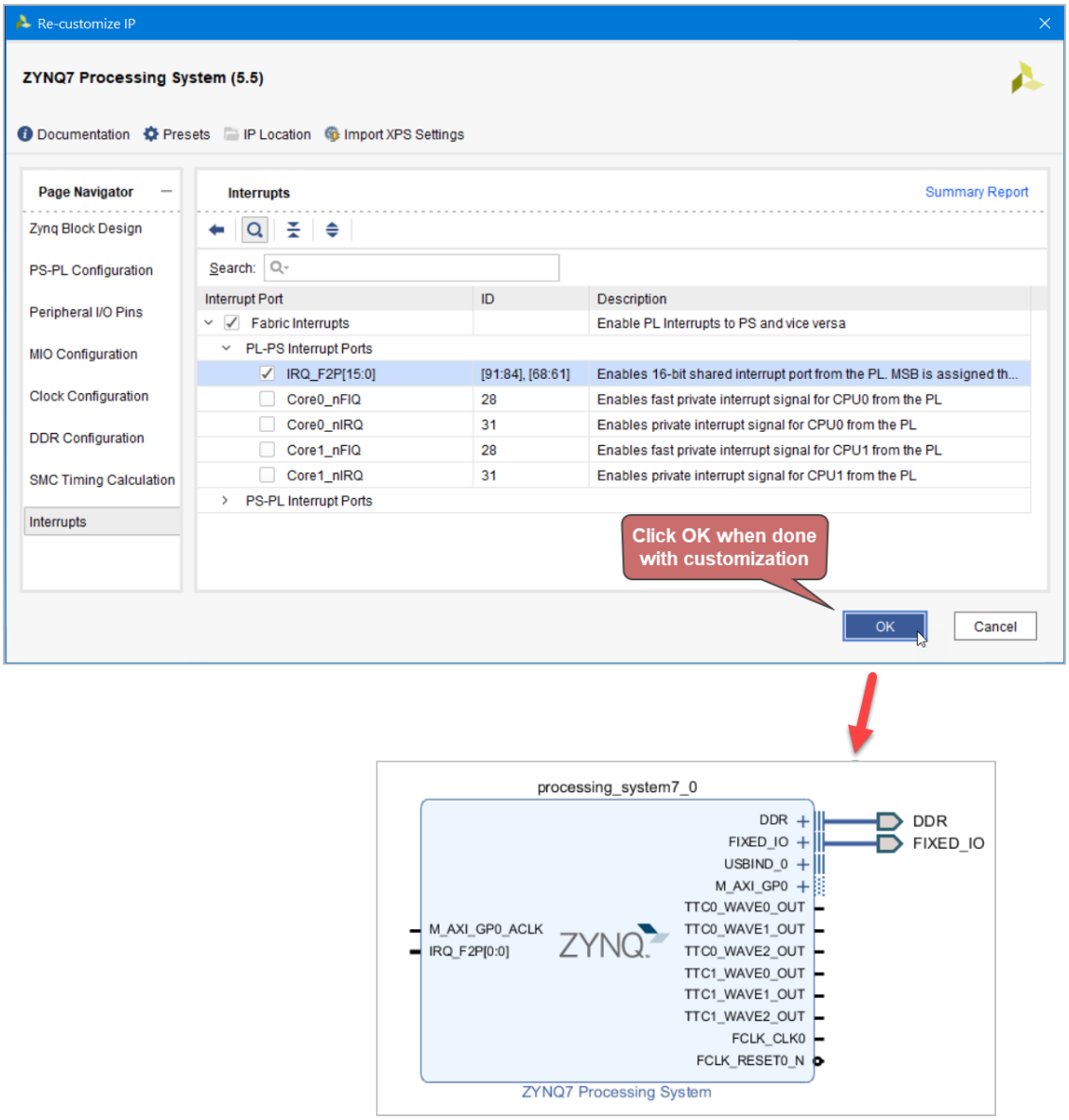


Figure 26. Click OK to apply customisation.

2.2.4 Add TTC Output Pins

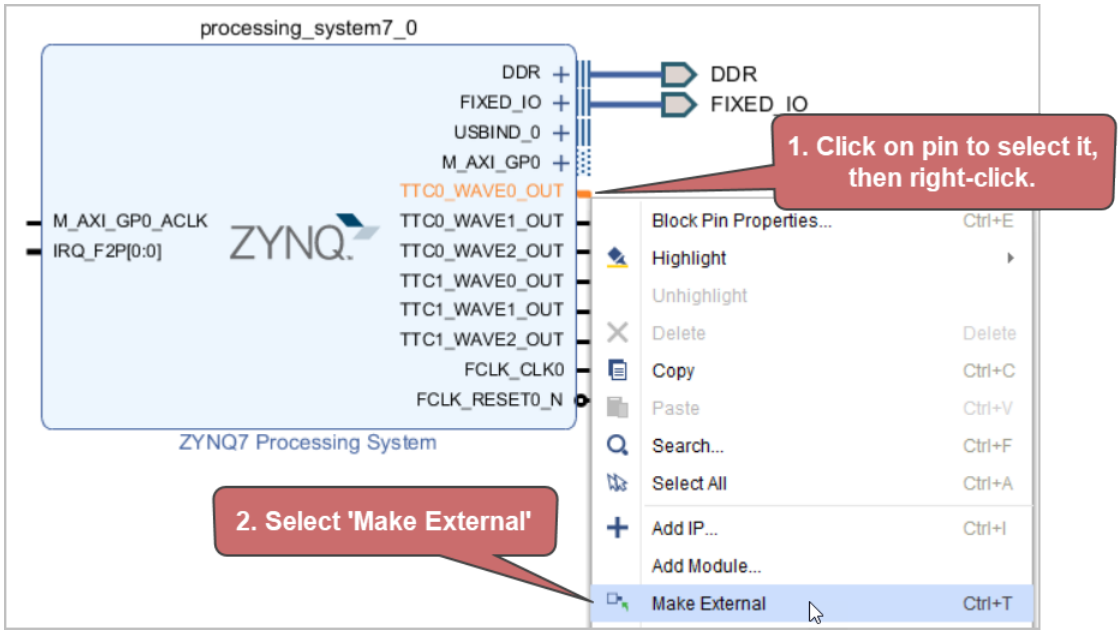


Figure 27. Add TTC pins by right-clicking and selecting "Make External"

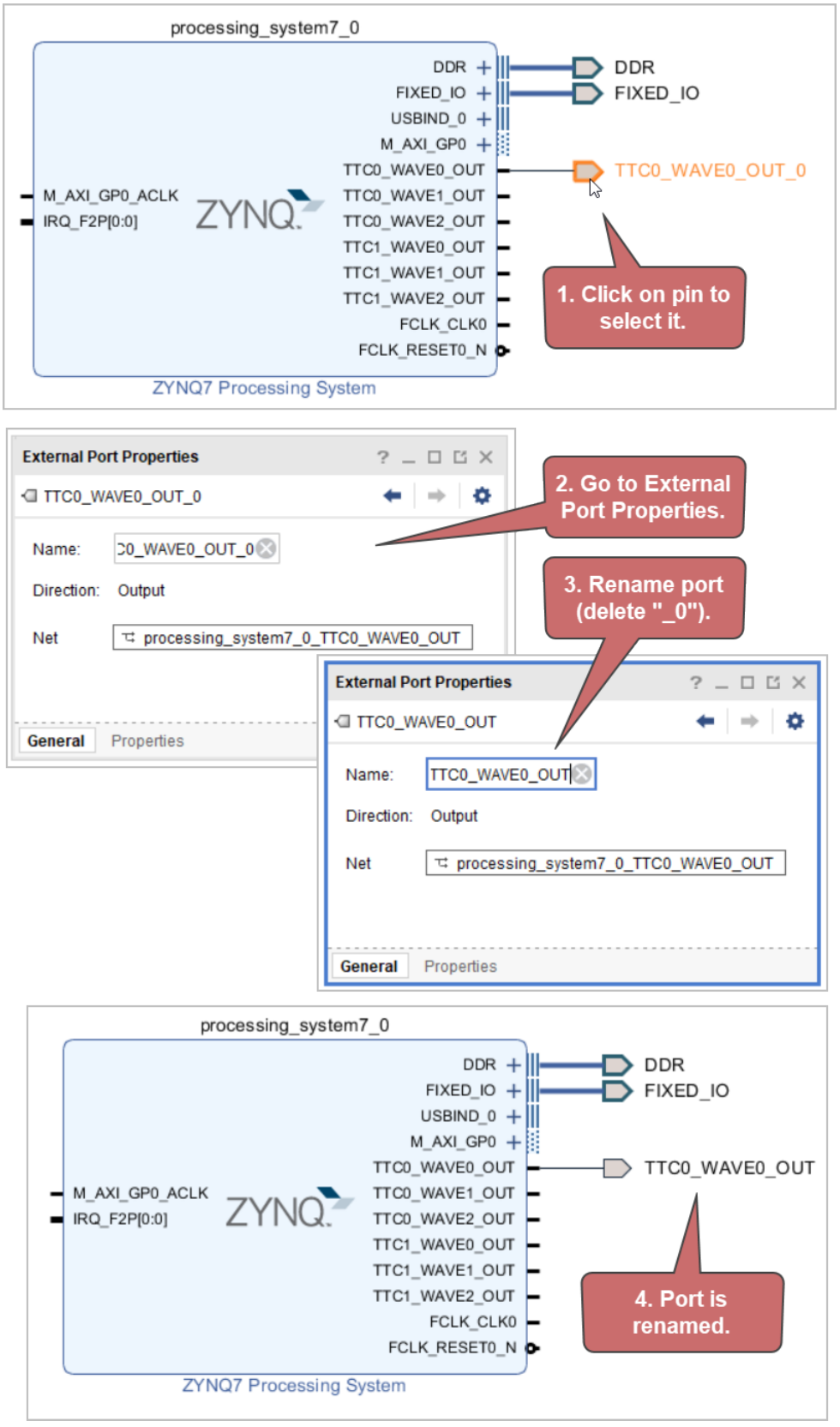


Figure 28. Rename the pins: delete "_0".

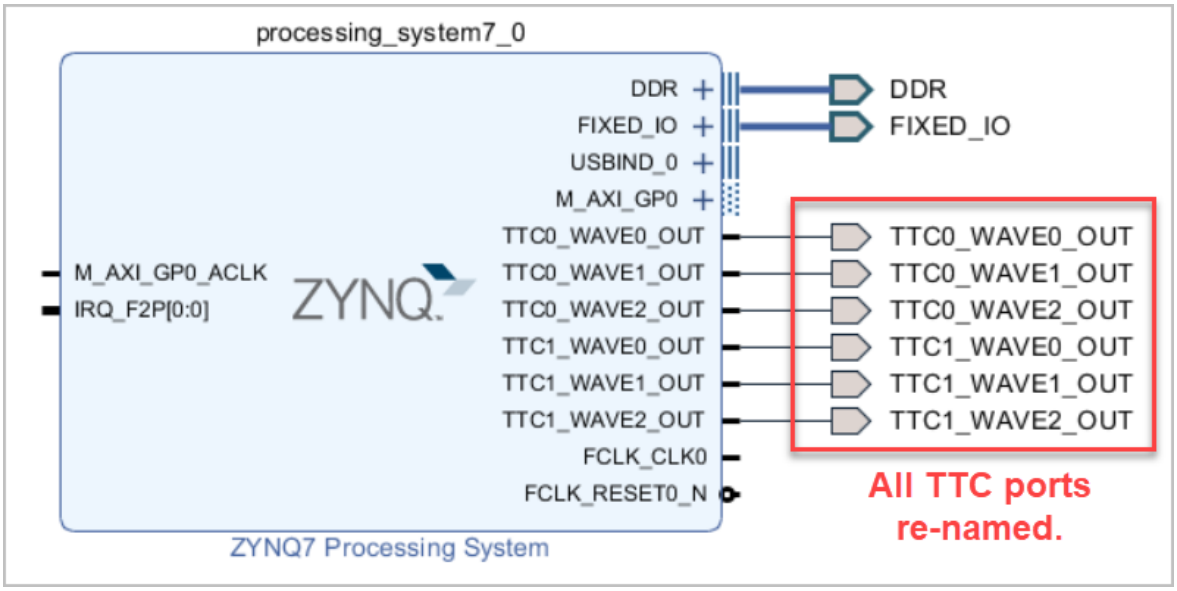


Figure 29. Repeat for all TTC pins

2.2.5 Add AXI GPIO IP

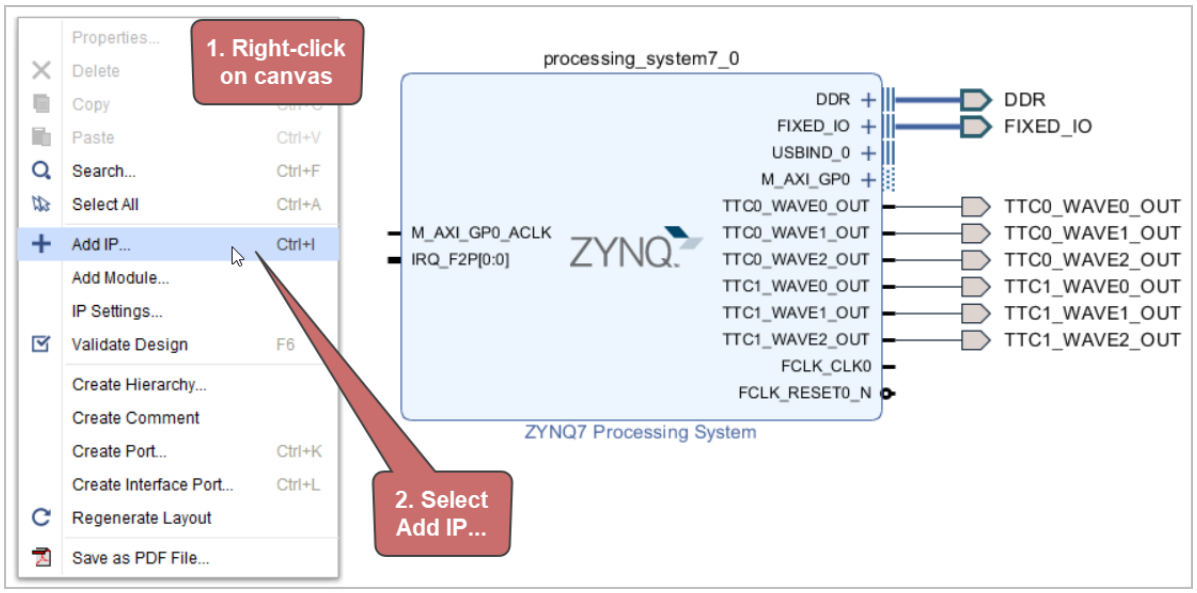


Figure 30. Add the AXI GPIO IP (1).

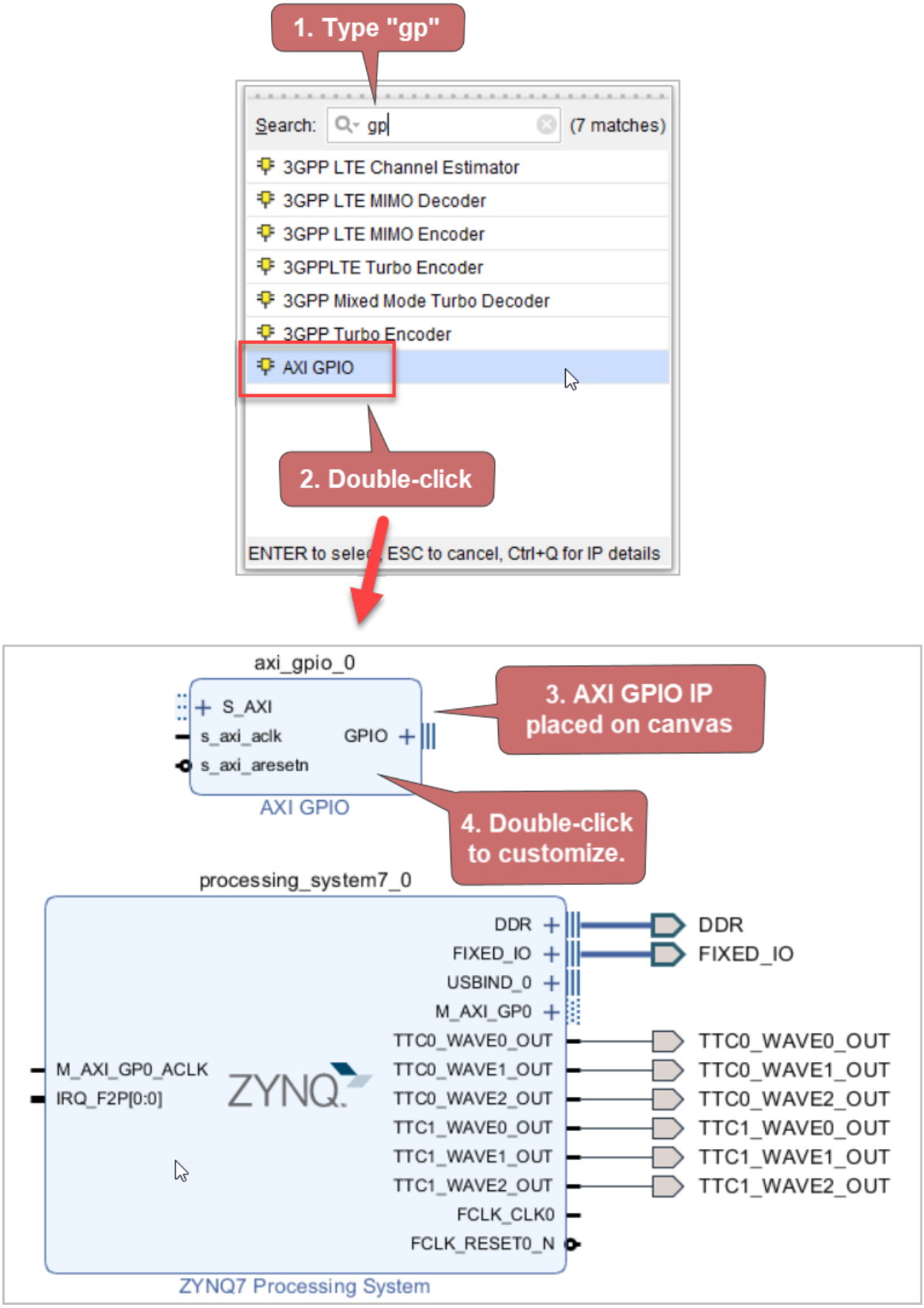


Figure 31. Add the AXI GPIO IP (2).

2.2.5.1 Customise the AXI GPIO IP

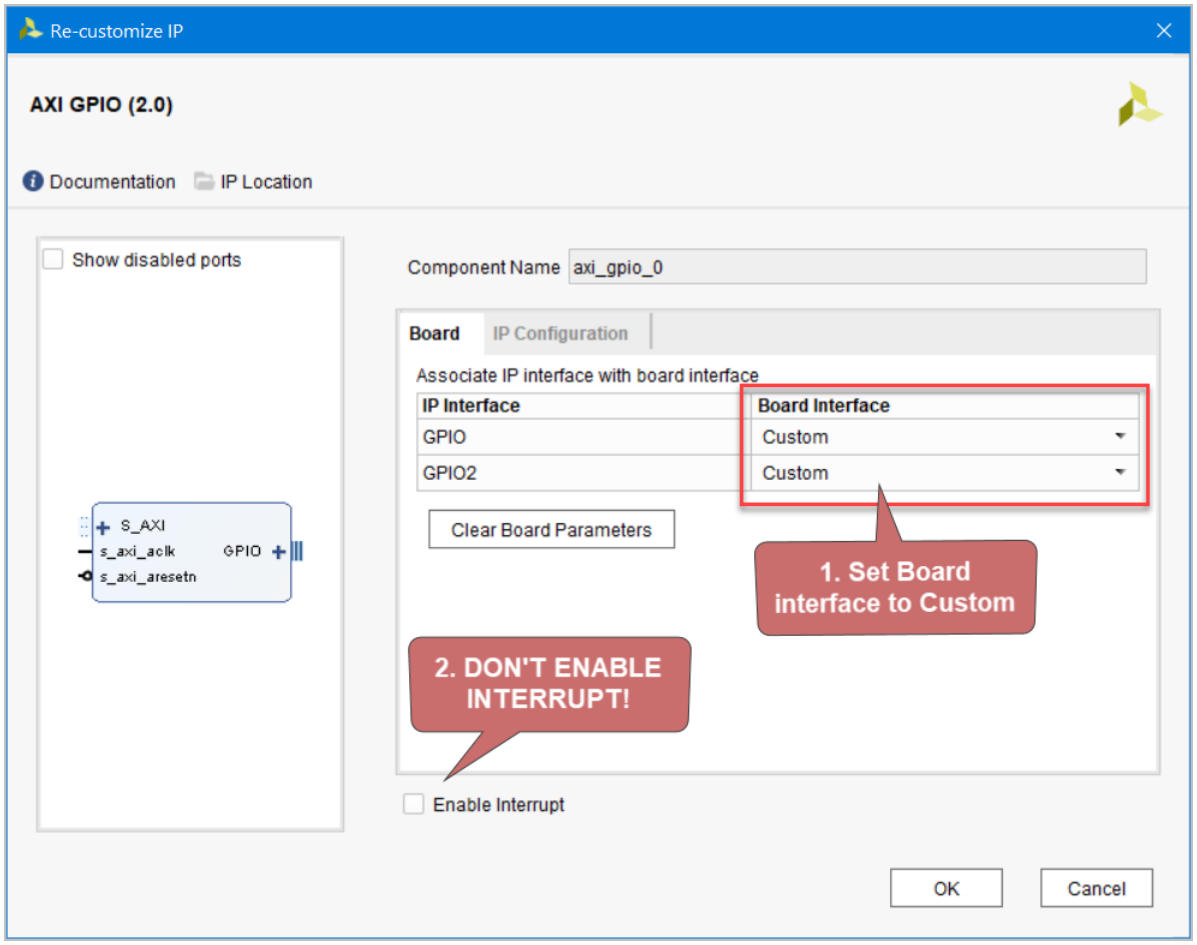


Figure 32. Set the GPIO Board options

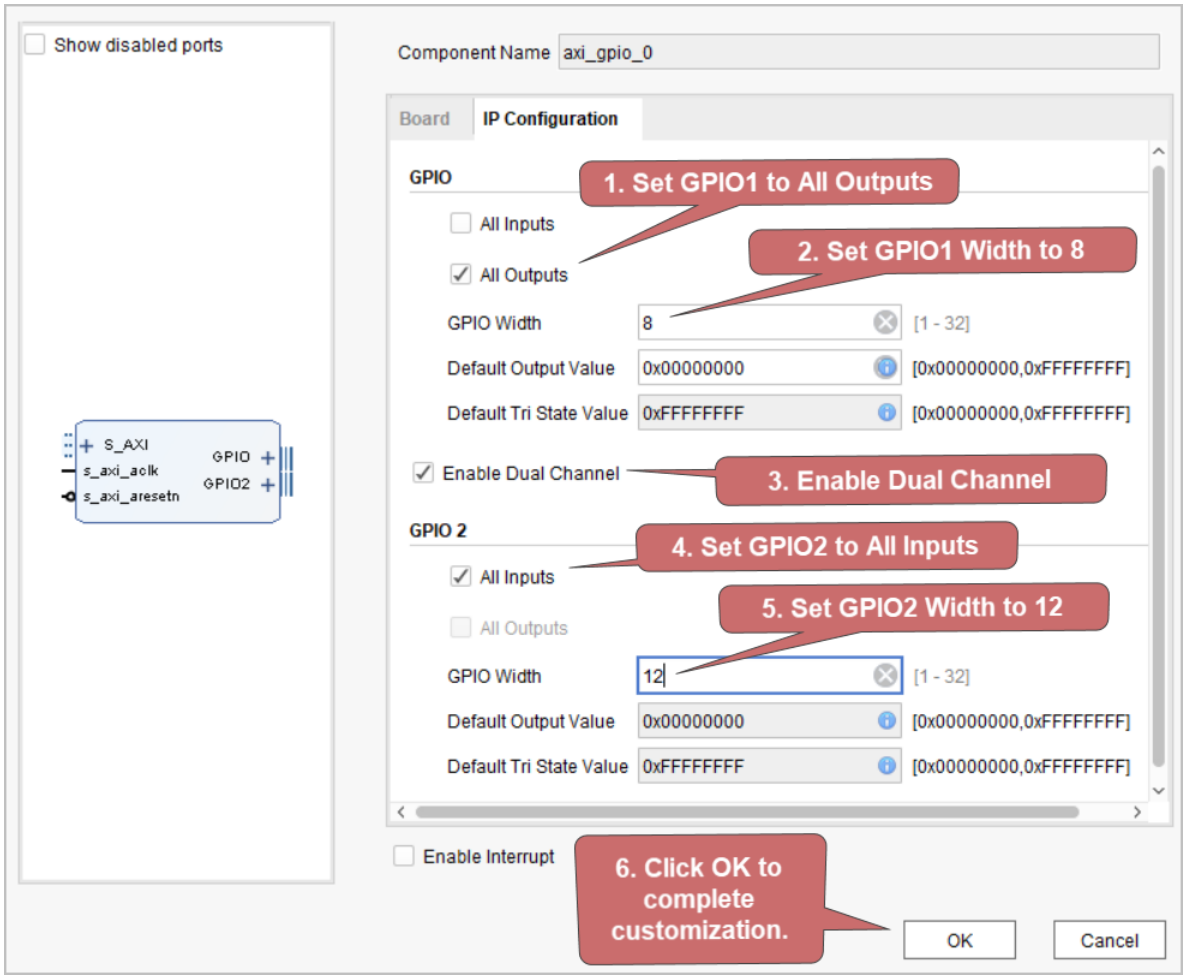


Figure 33. Set GPIO IP Configuration.

2.2.5.2 Add GPIO Input/Output Pins

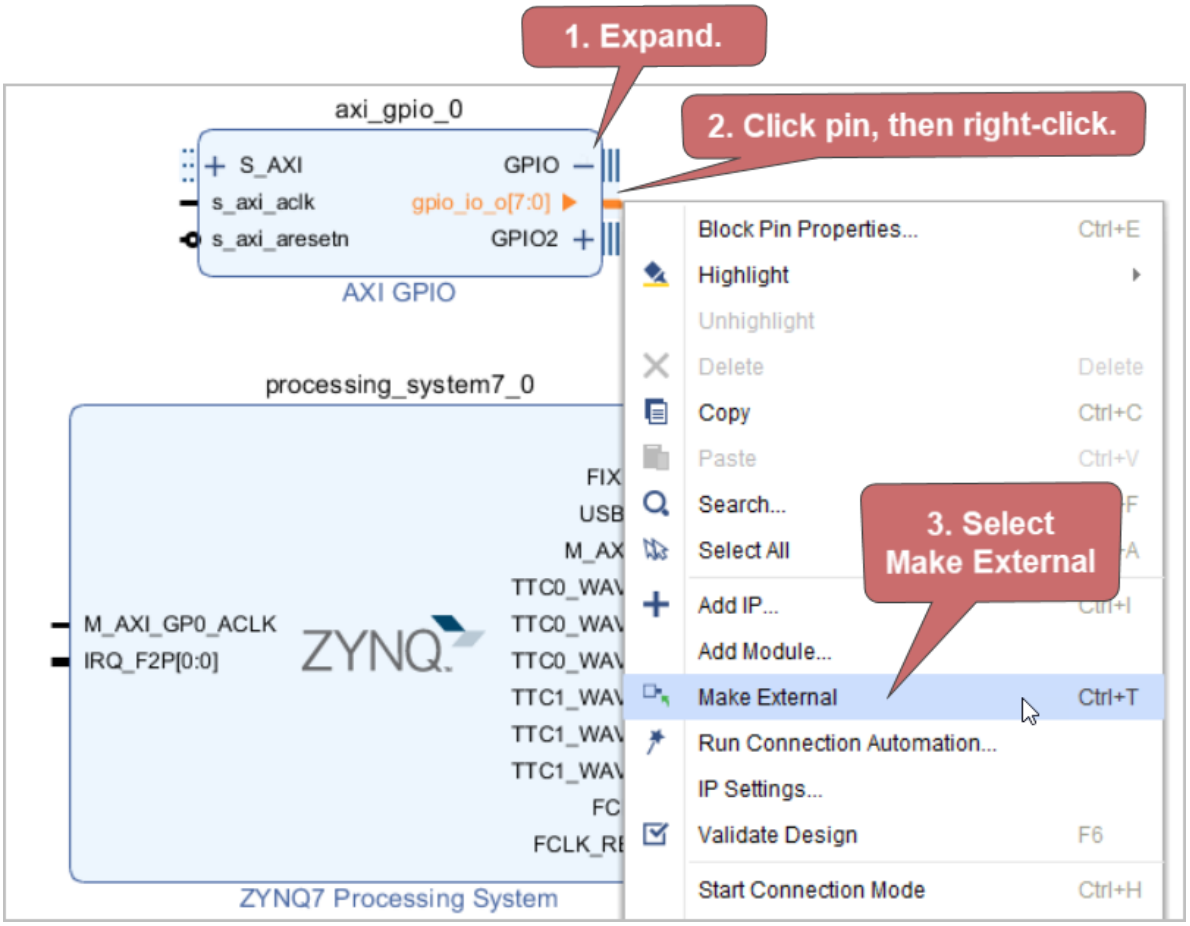


Figure 34. Add the output pins for channel 1.

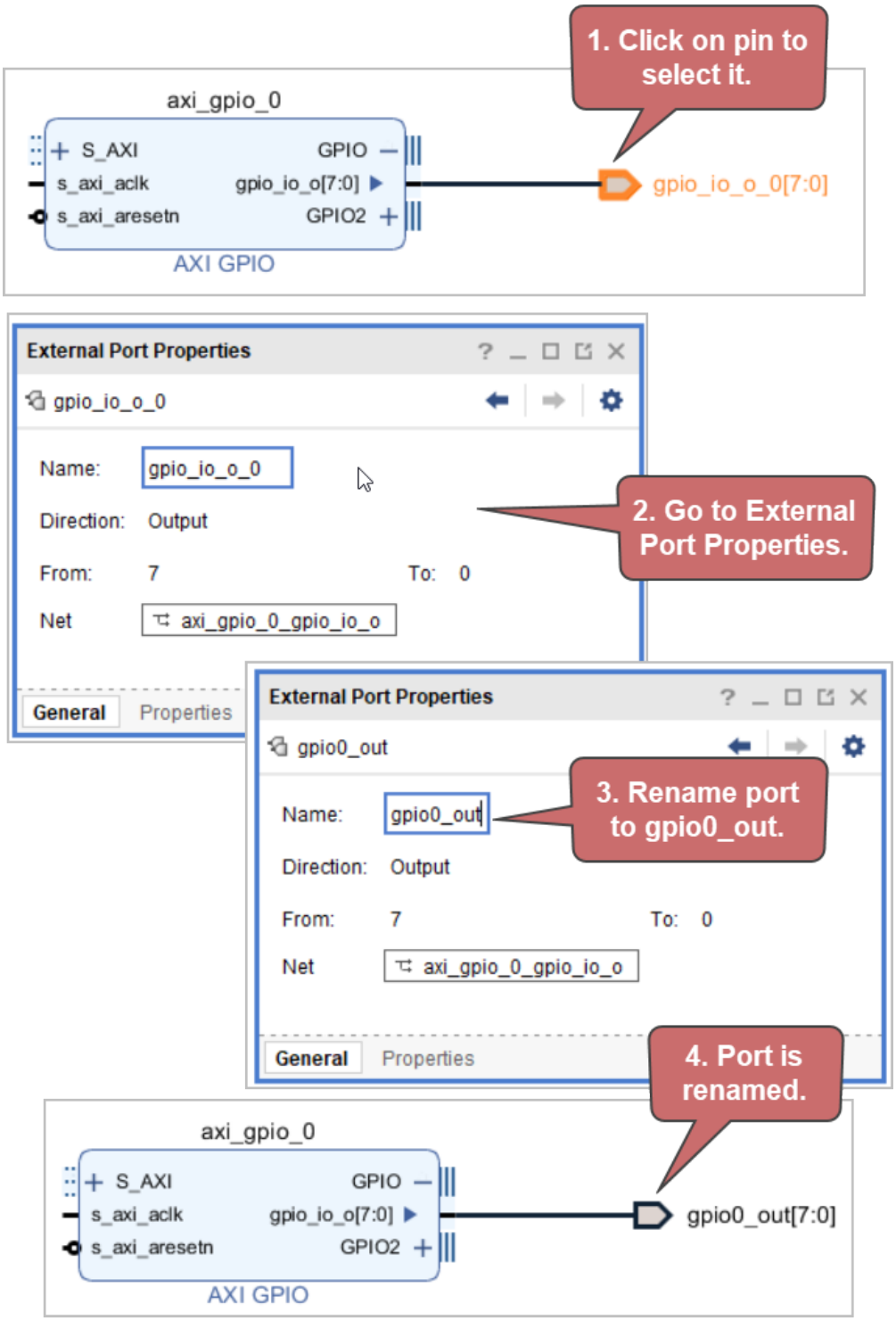


Figure 35. Rename the output pins.

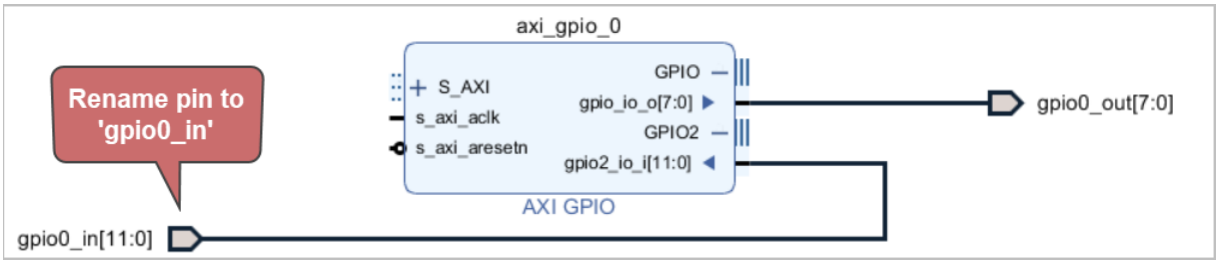


Figure 36. Repeat for the channel 2 input pins.

2.2.5.3 Run Connection Automation to Update Block Design

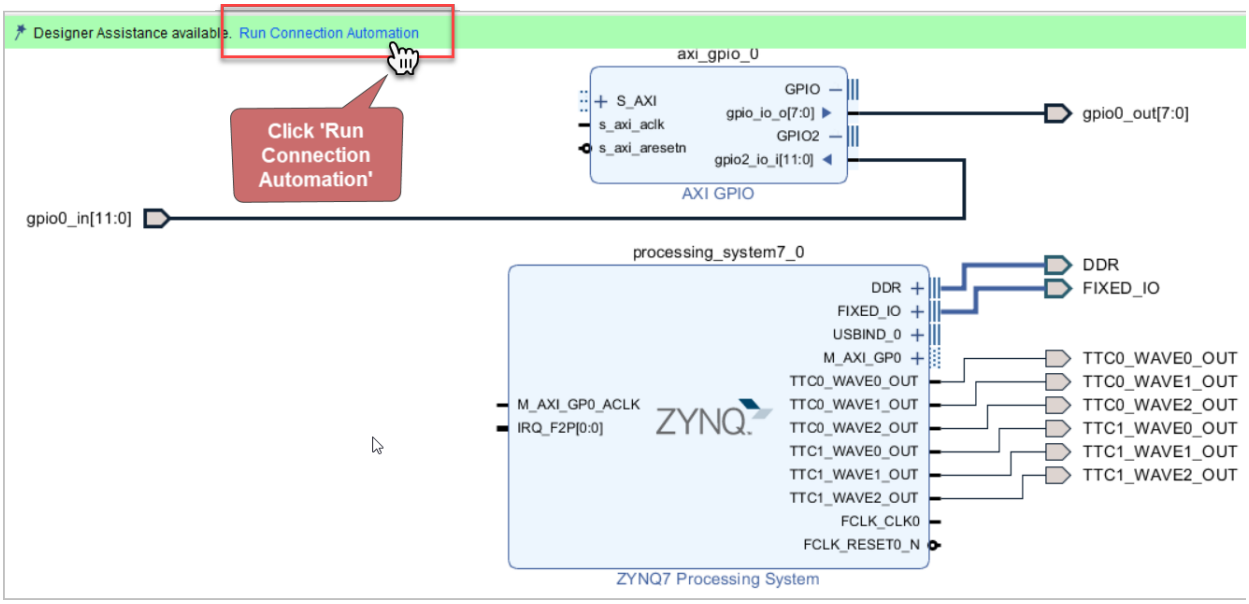


Figure 37. Run Connection Automation.

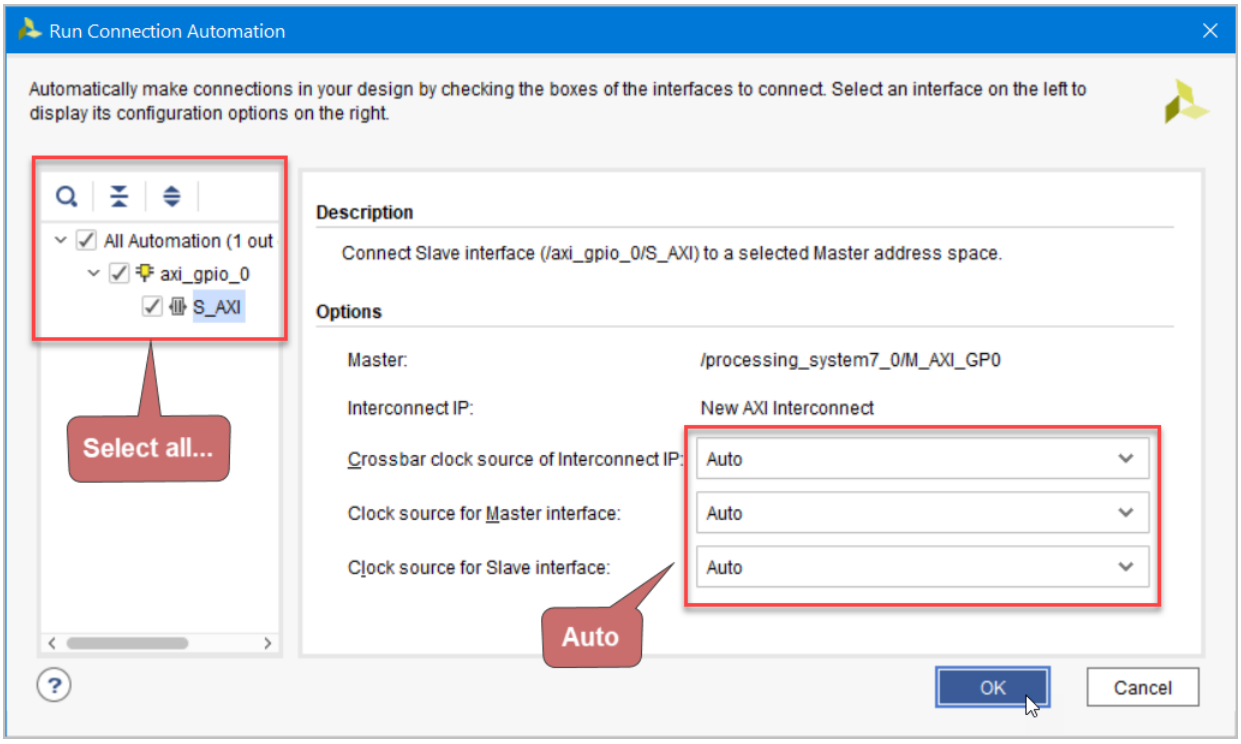


Figure 38. Ensure the options are set to Auto, and click OK.

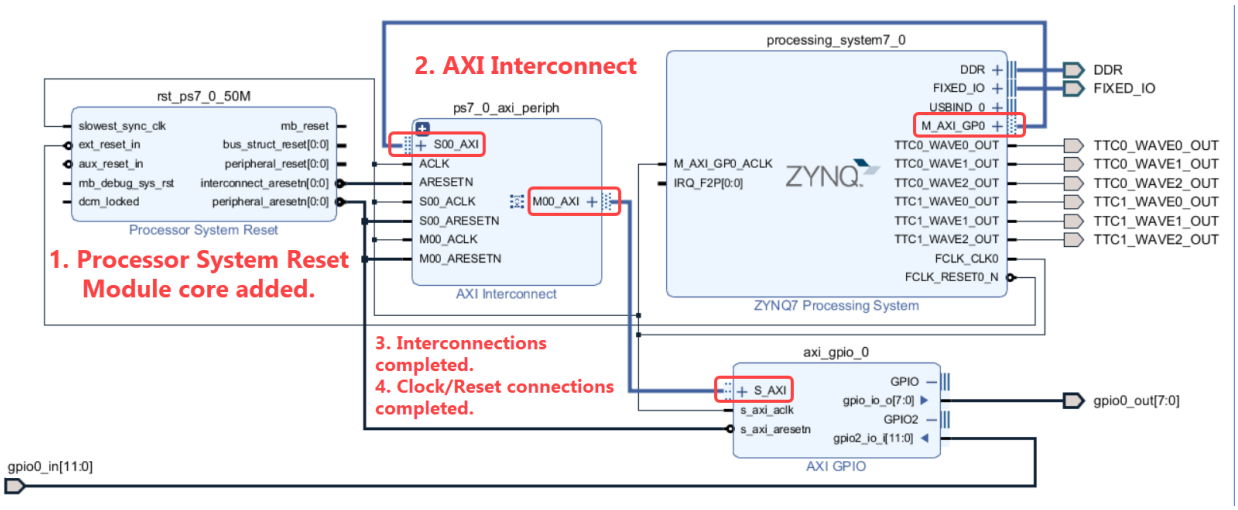


Figure 39. The updated block diagram is as shown.

2.2.5.4 Validate Design

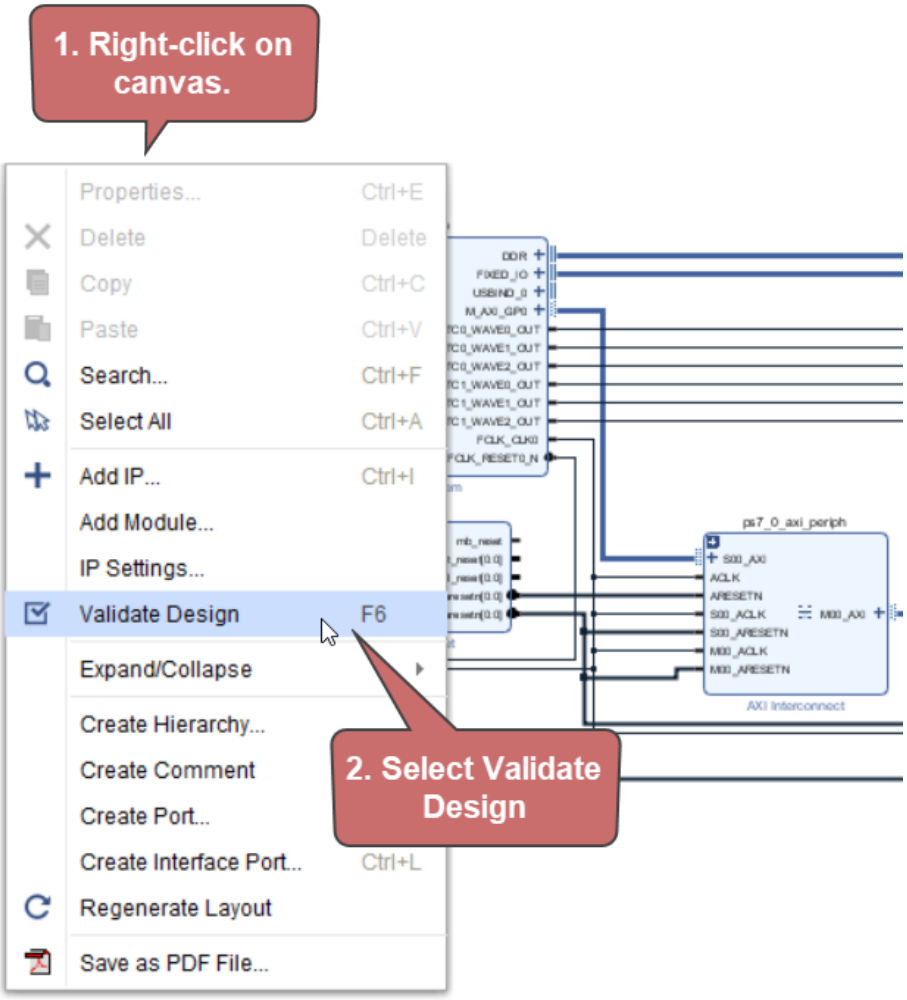


Figure 40. Validate the design.

Figure 41. Click OK to continue.

2.2.6 Add AXI Quad SPI IP

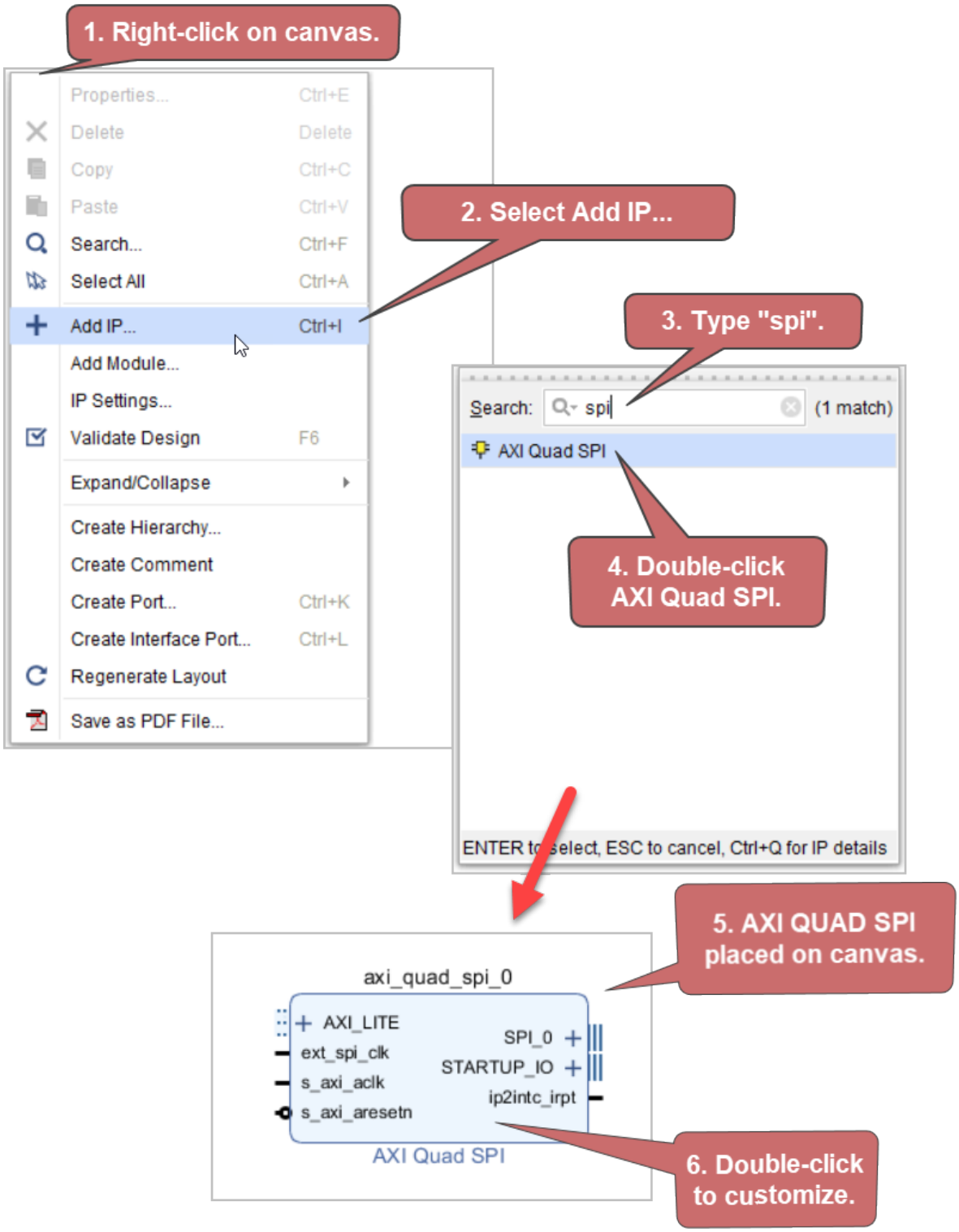


Figure 42. Add the AXI Quad SPI module.

2.2.6.1 Customise AXI Quad SPI IP

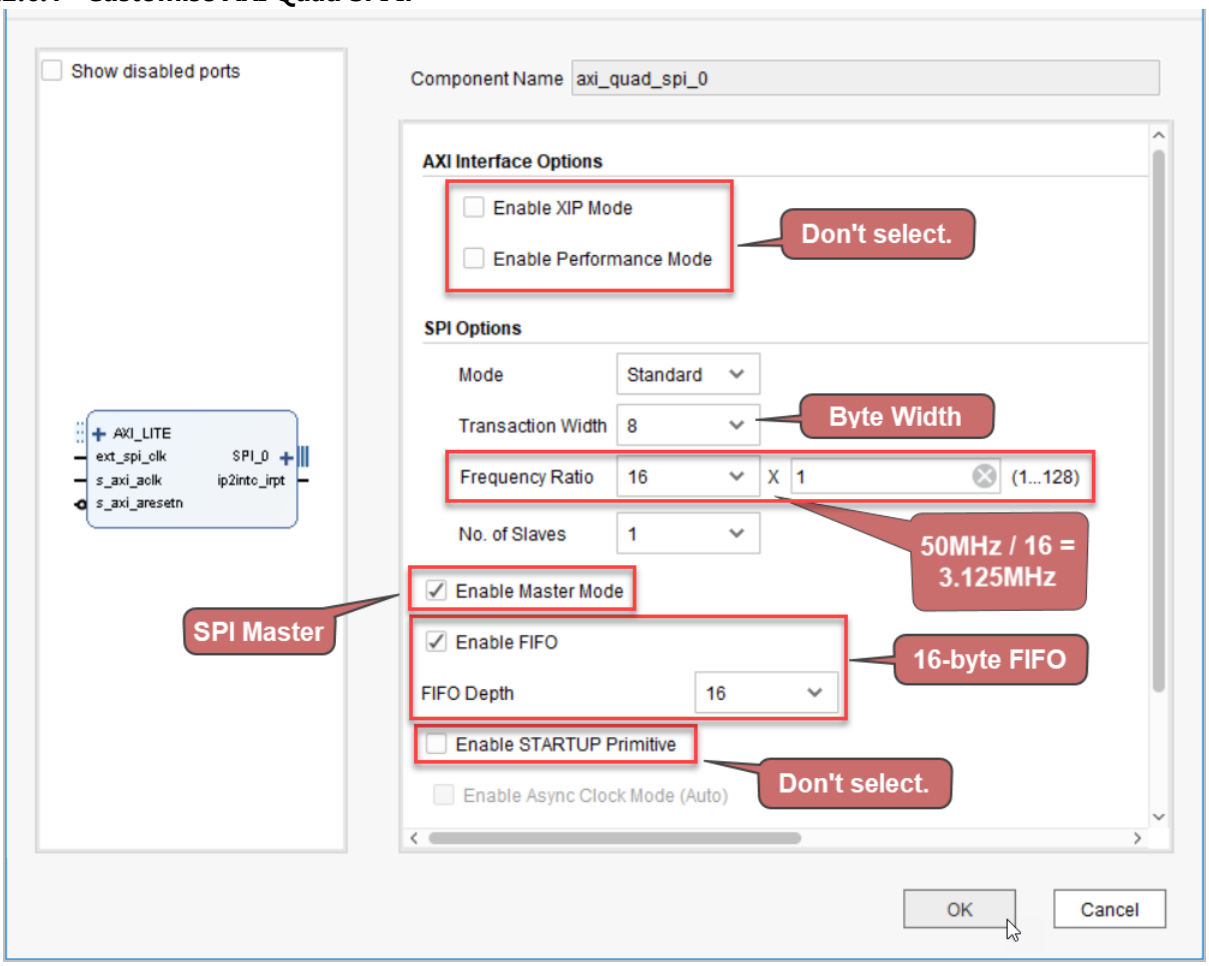


Figure 43. Customise the AXI Quad SPI as shown and click OK.

2.2.6.2 Add SPI Input/Output Pins

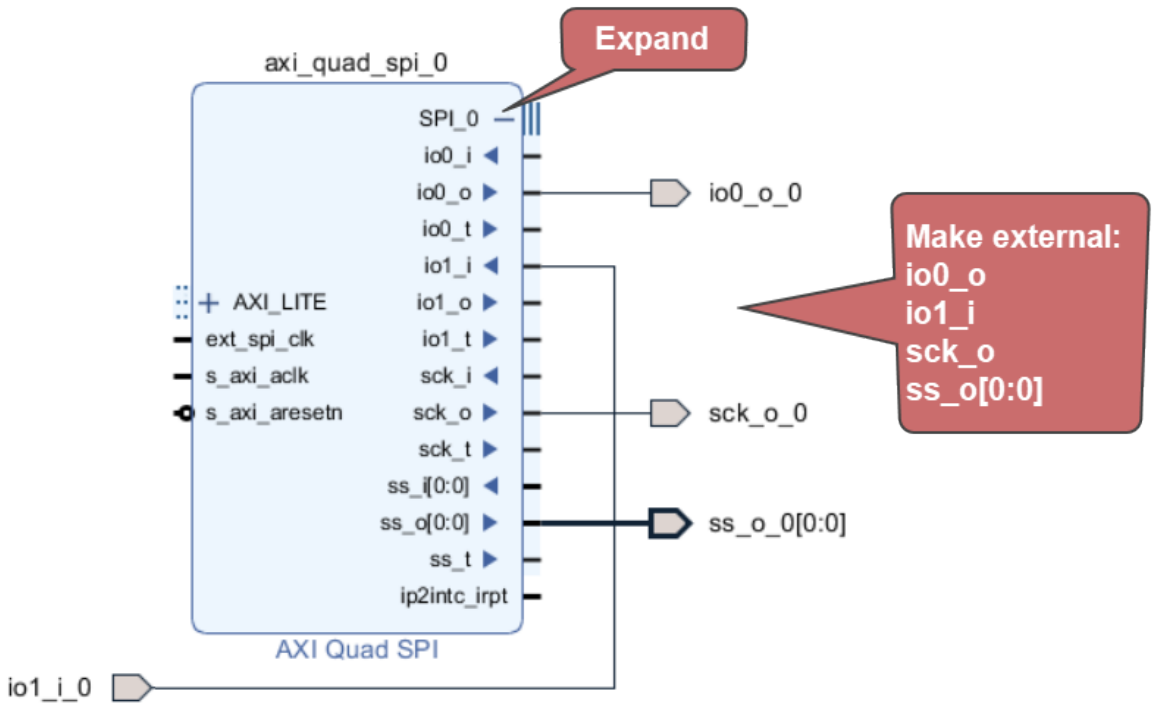


Figure 44. Add individual SPI pins

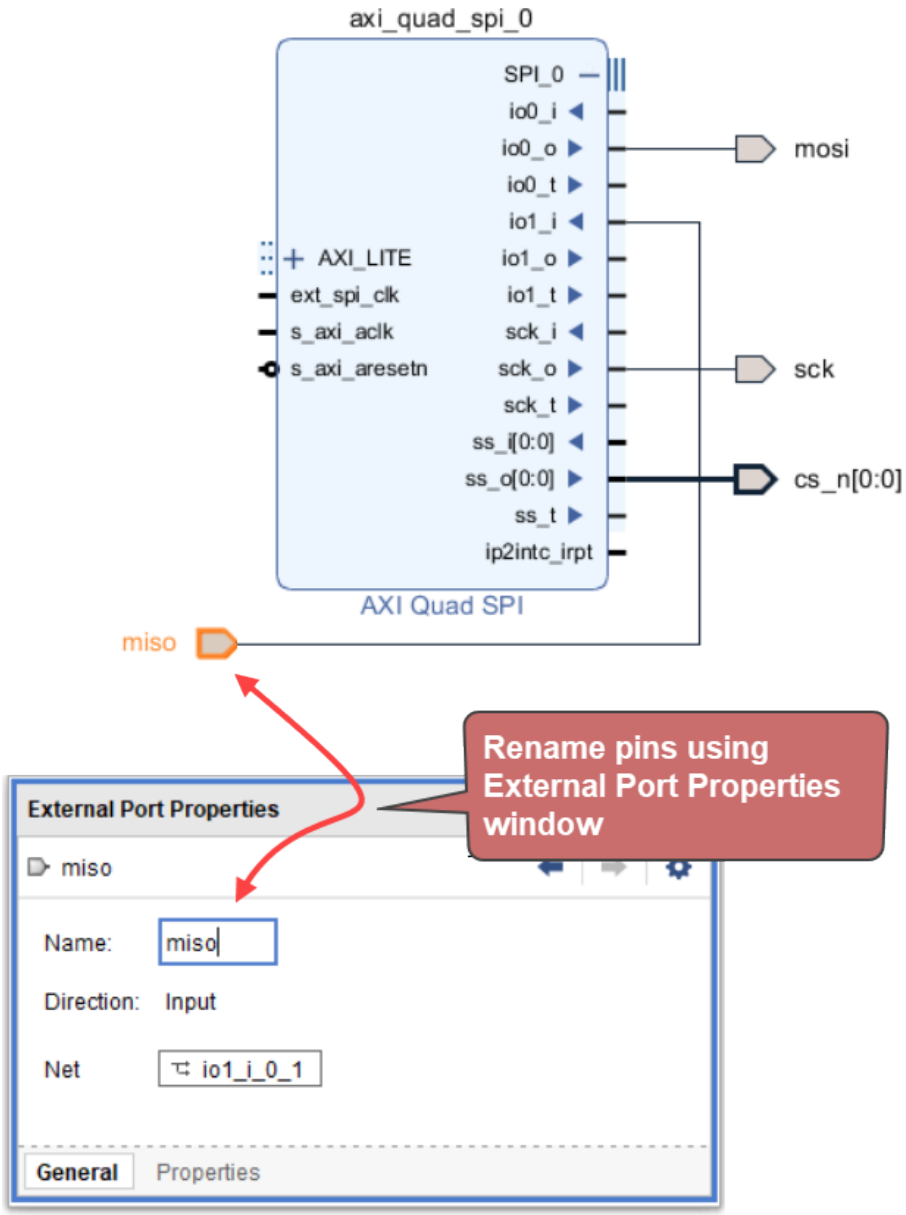


Figure 45. Rename the pins to match standard SPI names.

2.2.6.3 Run Connection Automation to Update Block Design

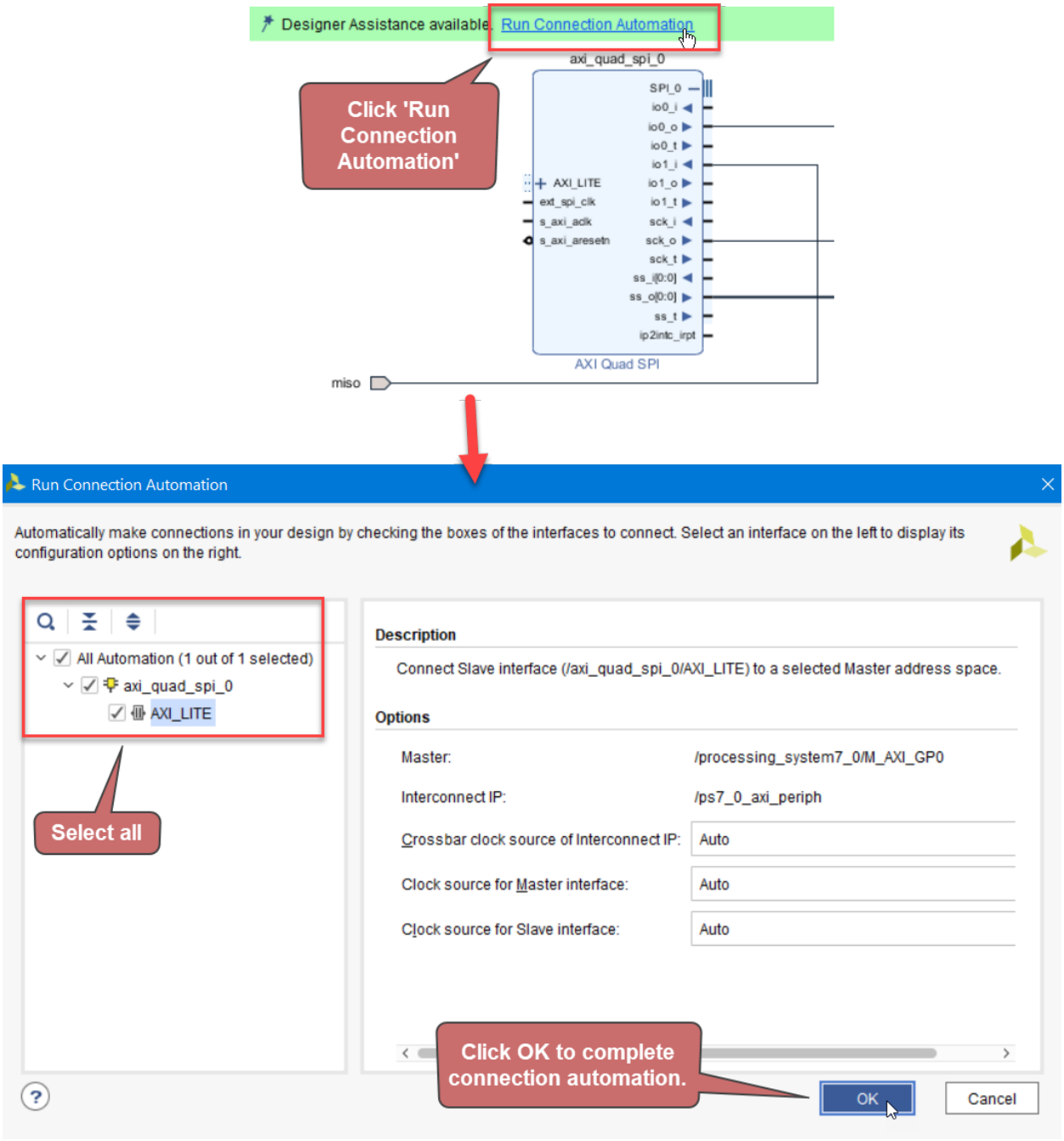


Figure 46. Run Connection Automation again.

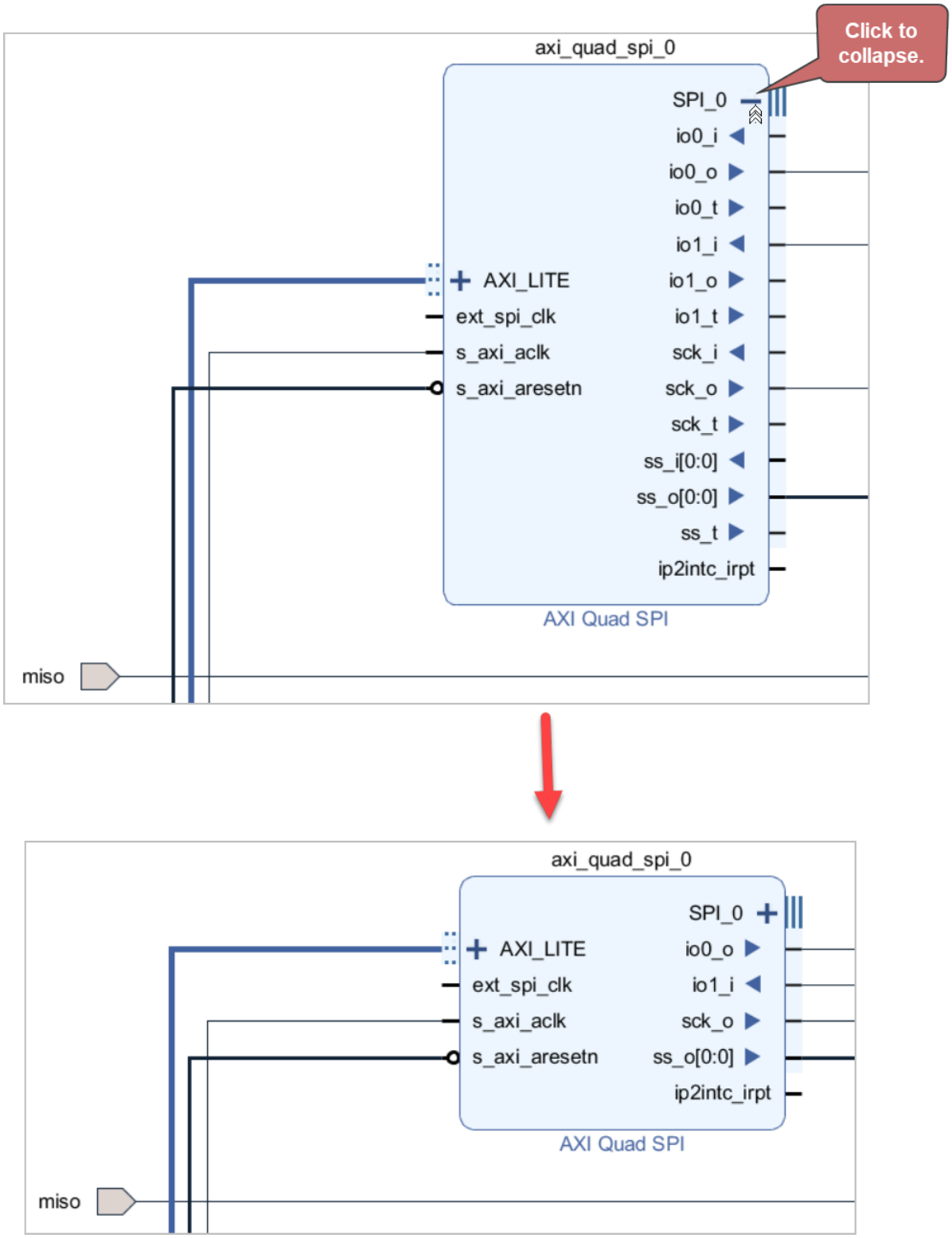


Figure 47. Tidy up block diagram by collapsing SPI_0 port.

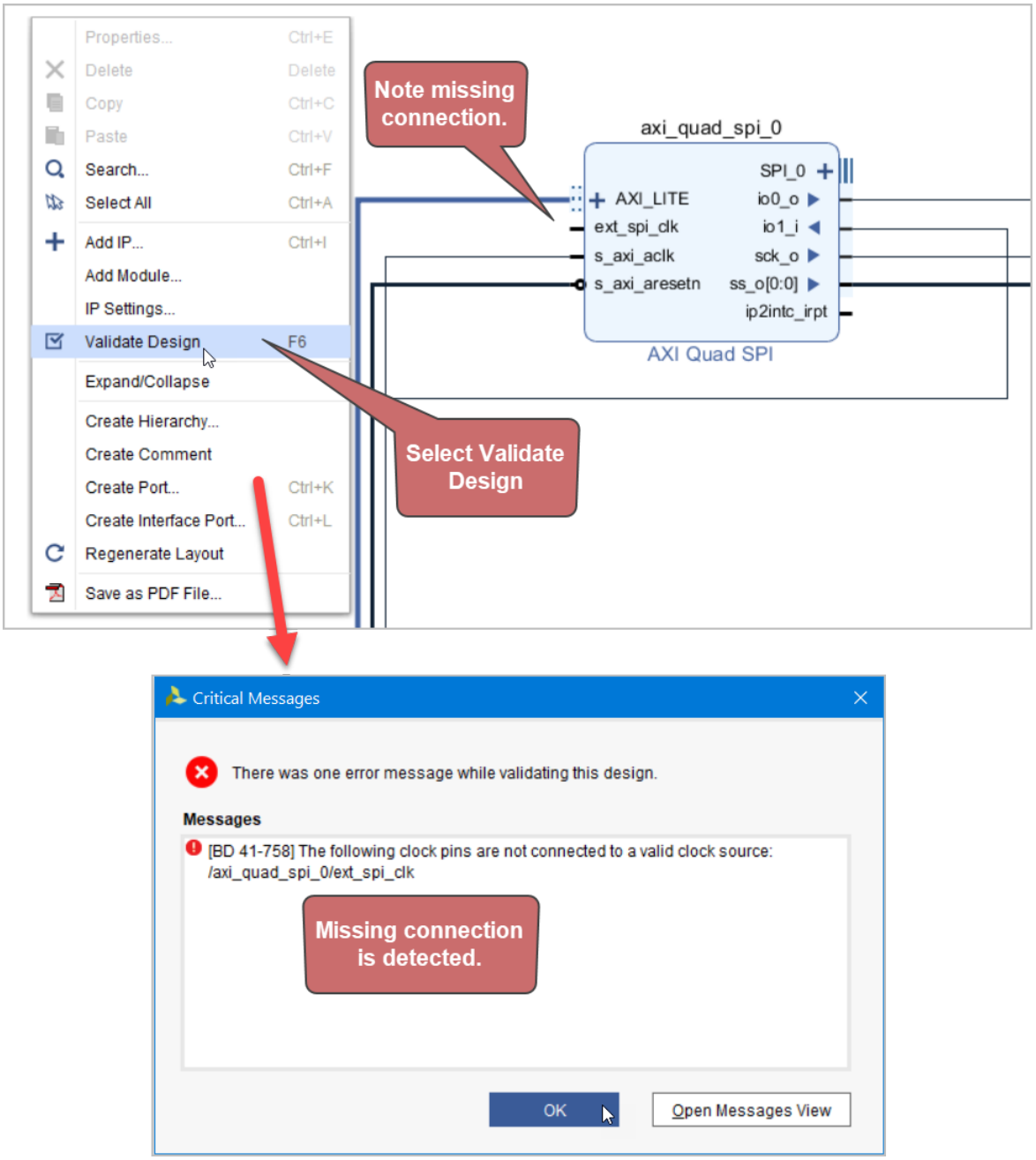


Figure 48. Validate the design; an error should be flagged.

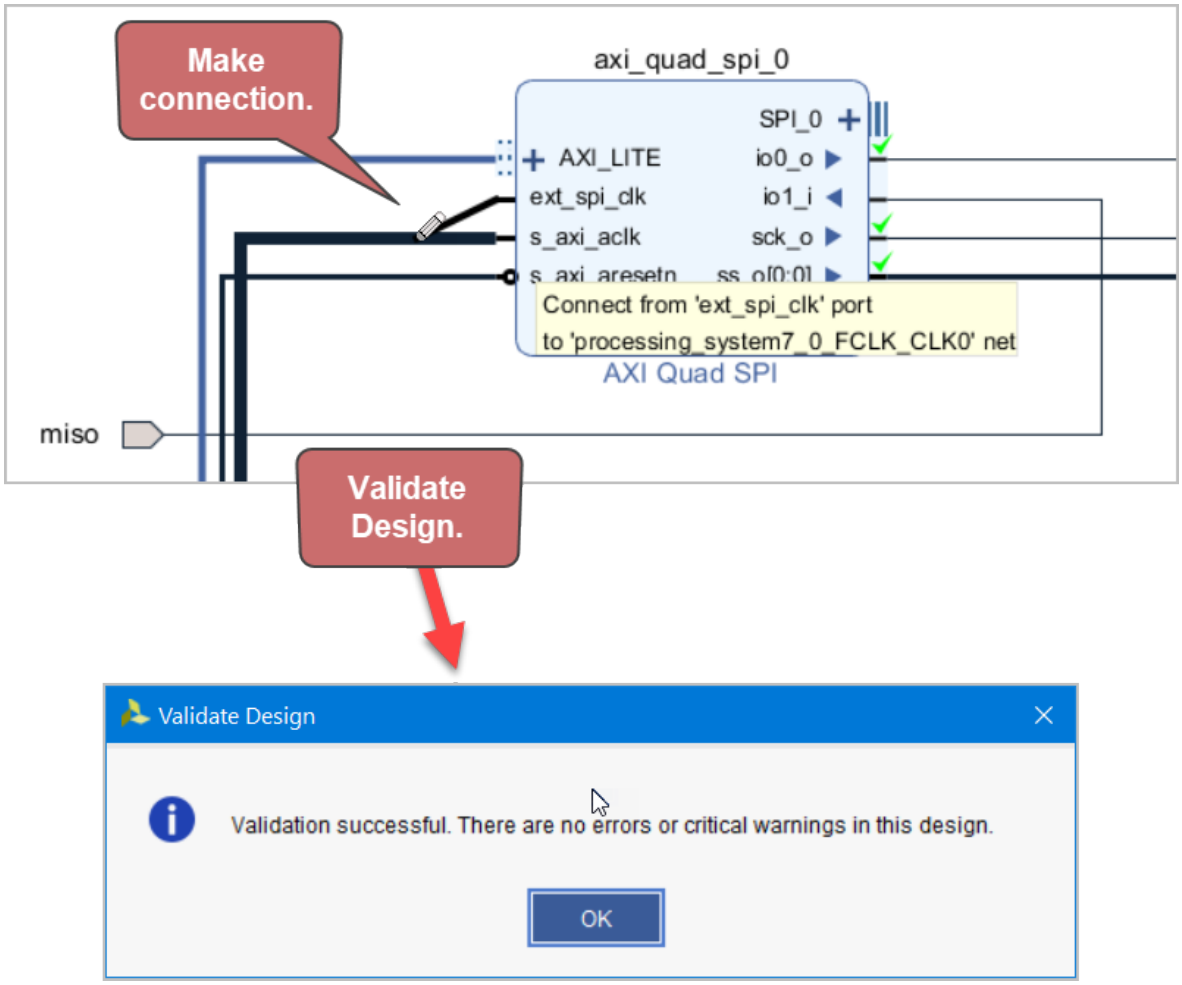


Figure 49. Fix the error by connecting ext_spi_clk to s_axi_aclk.

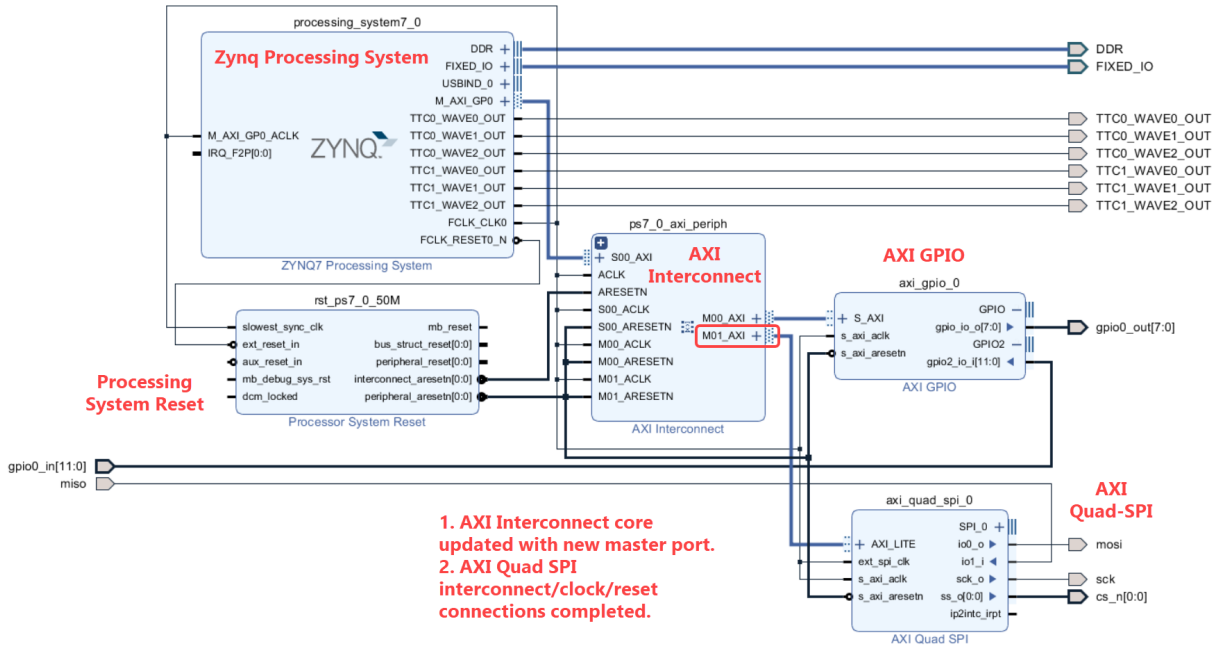


Figure 50. Block diagram at this stage of the design.

2.2.7 Add Concat IP for PmodACL Interrupts

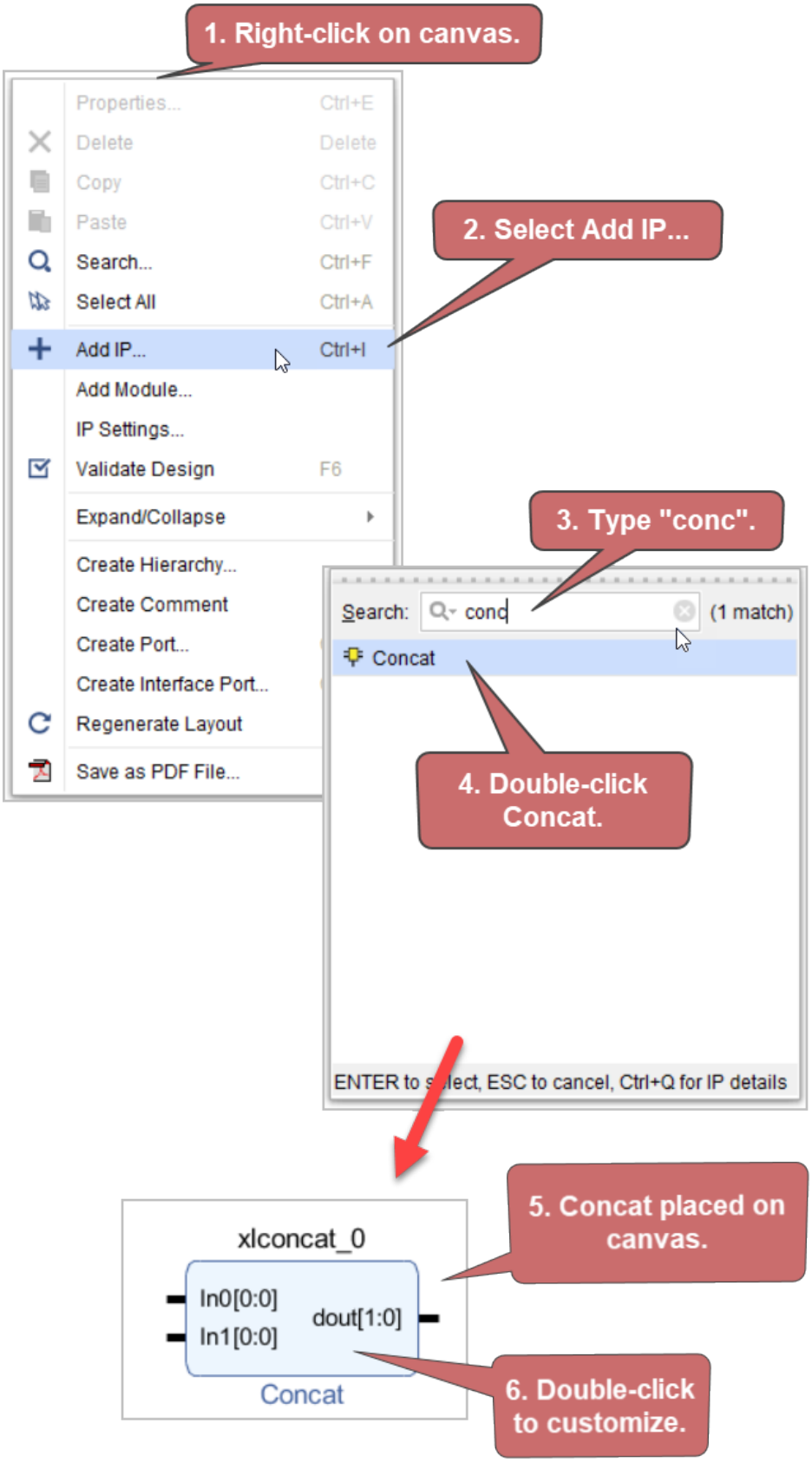


Figure 51. Add the Concat Utility IP for interrupt logic.

2.2.7.1 Customise Concat IP

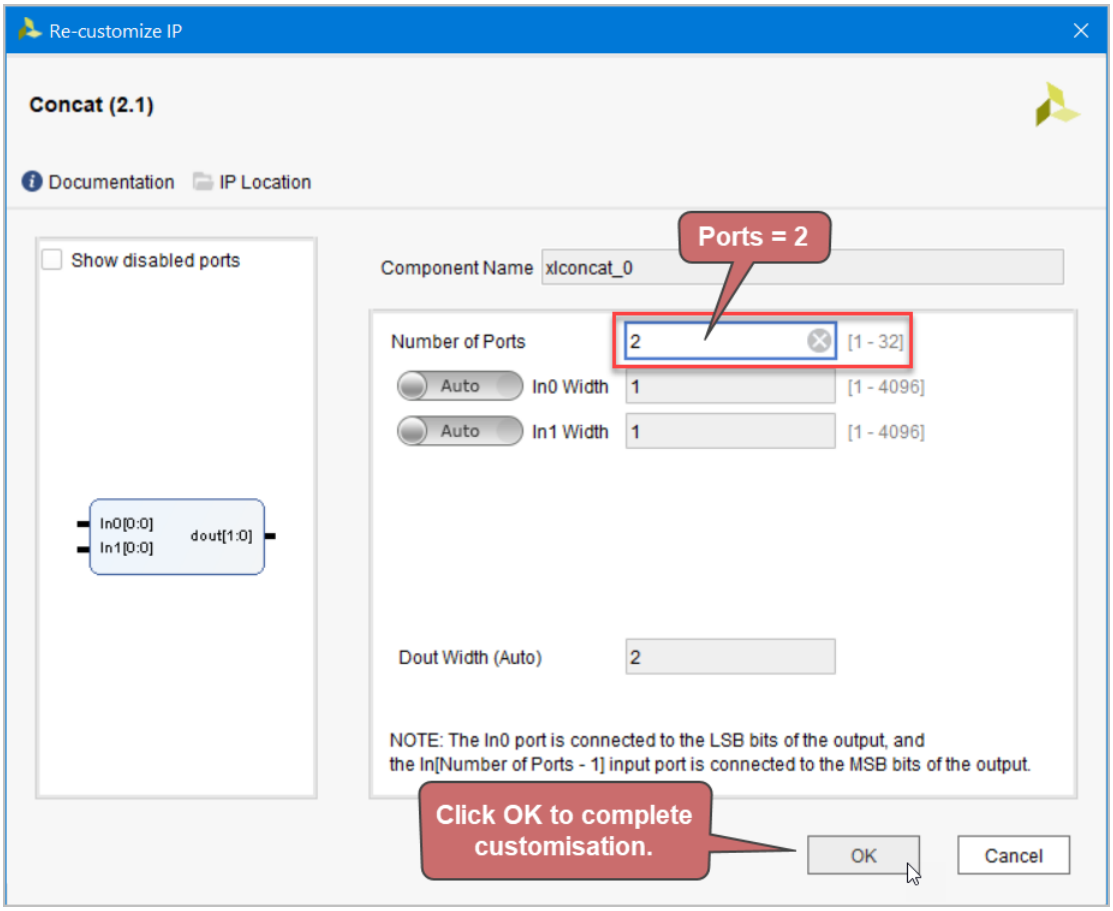


Figure 52. Ensure that two ports are selected and click OK

2.2.7.2 Add PmodACL Interrupt Pins

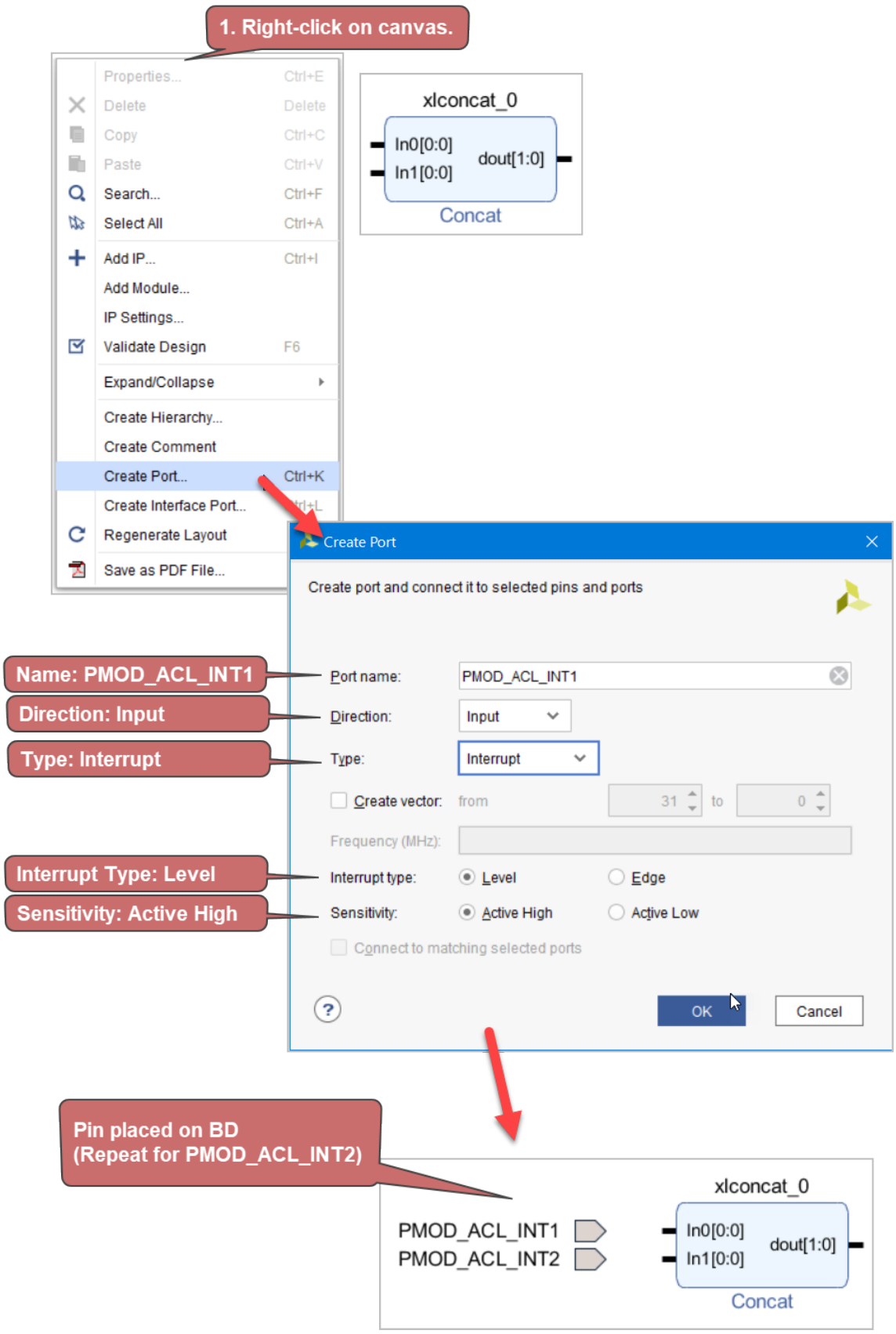


Figure 53. Add the Pmod ACL interrupt pins.

2.2.7.3 Make PmodACL Interrupt Pin Connections

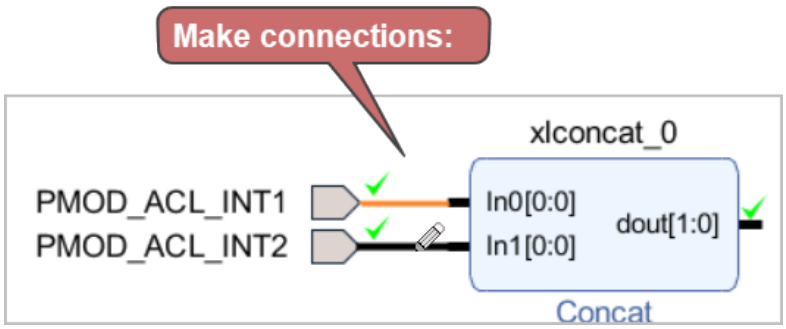


Figure 54. Make the interrupt pin connections.

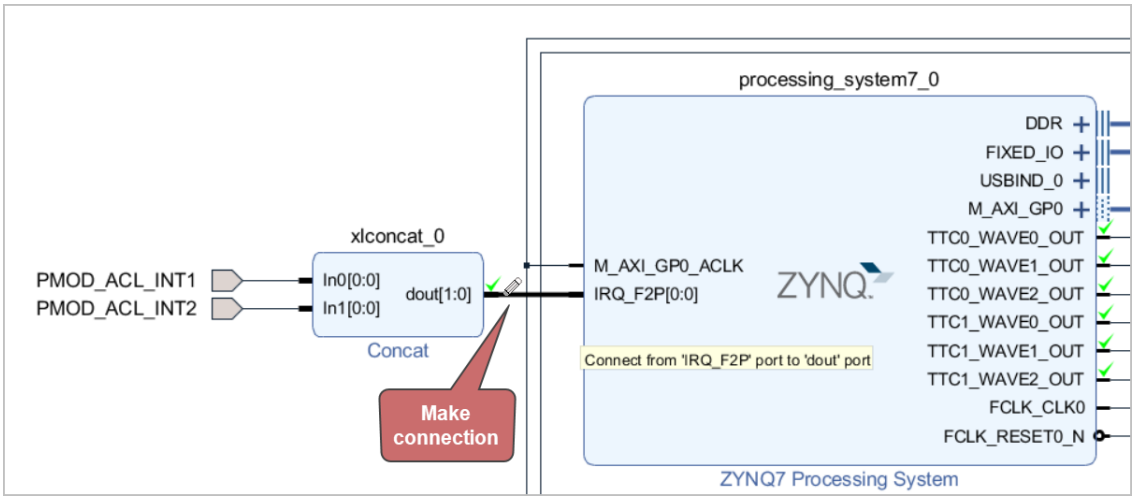


Figure 55. Connect the Concat IP output to IRQ_F2P port on the Zynq IP.

2.2.7.4 Validate Design

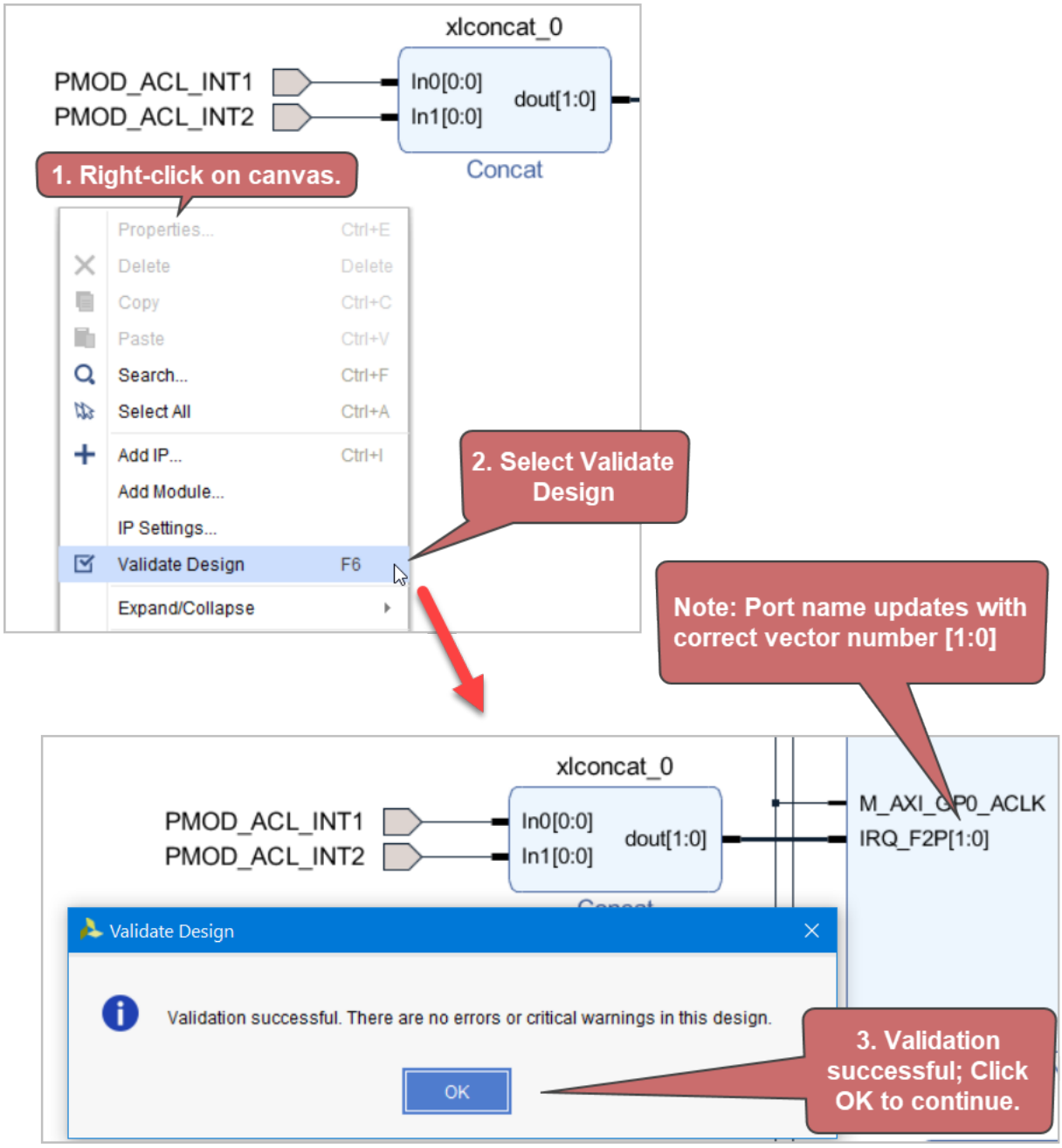


Figure 56. Validate the design for the final time.

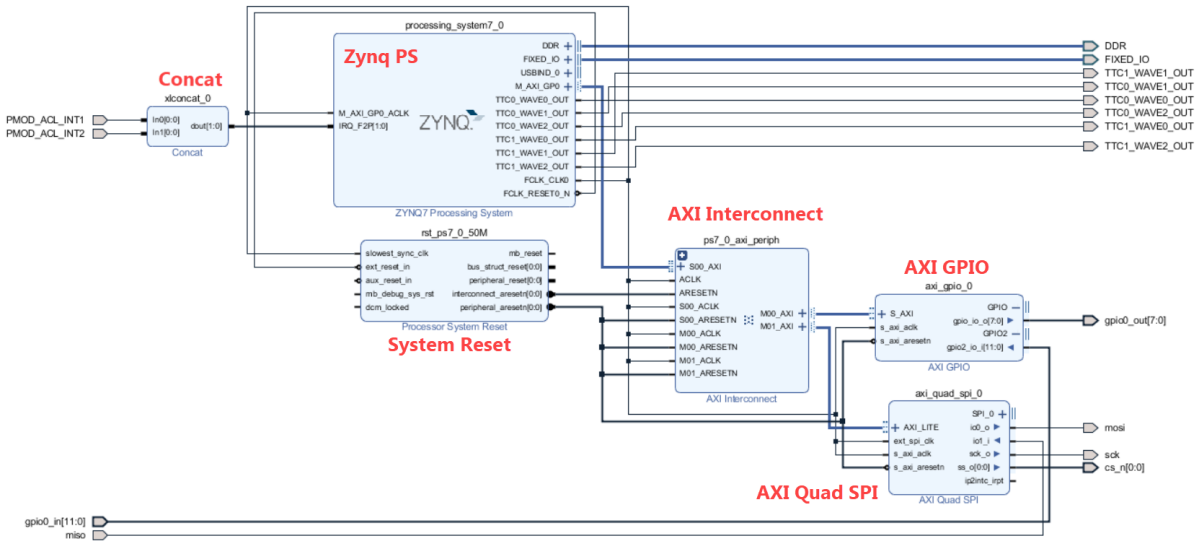


Figure 57. The final block diagram.

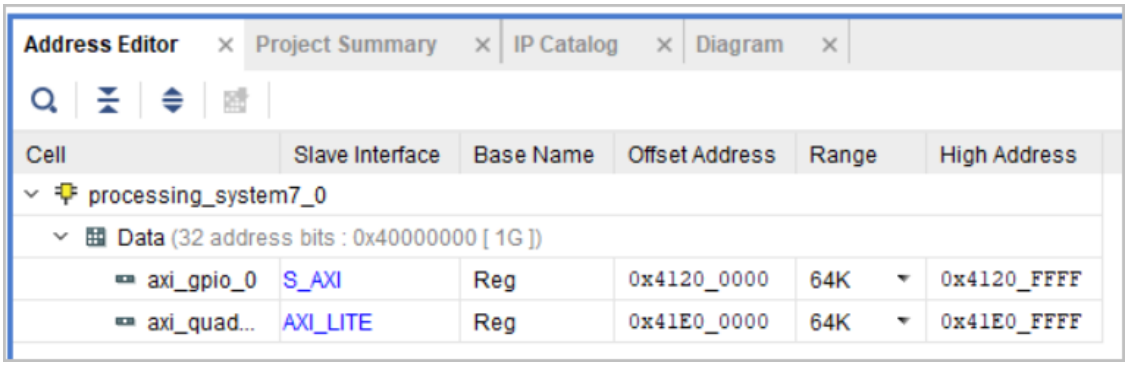


Figure 58. The address information for the AXI GPIO and AXI Quad IP can be viewed in the Address Editor tab.

2.3 Finalize the Design

2.3.1 Generate Output Products

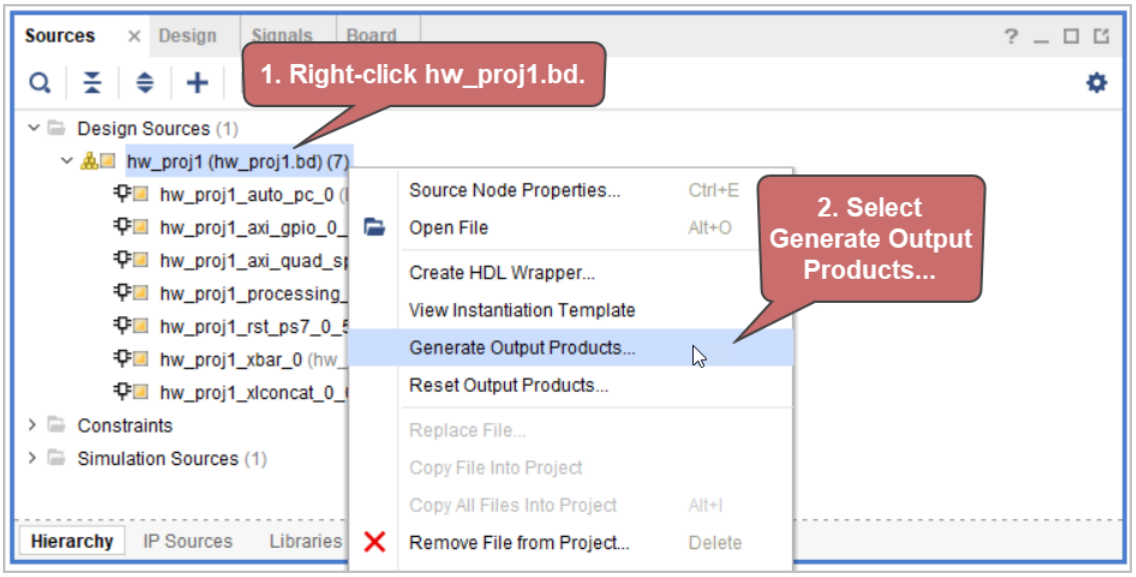


Figure 59. Select Generate Output Products...

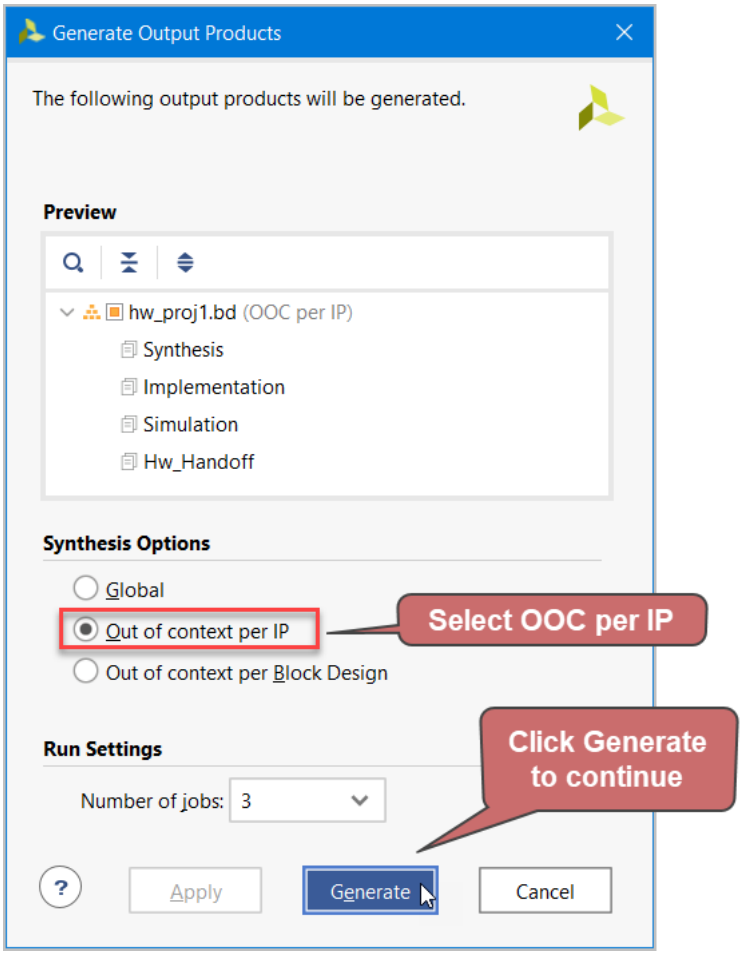


Figure 60. Ensure OOC per IP is selected, and click Generate.

Figure 61. OOC can be set to run in the background.

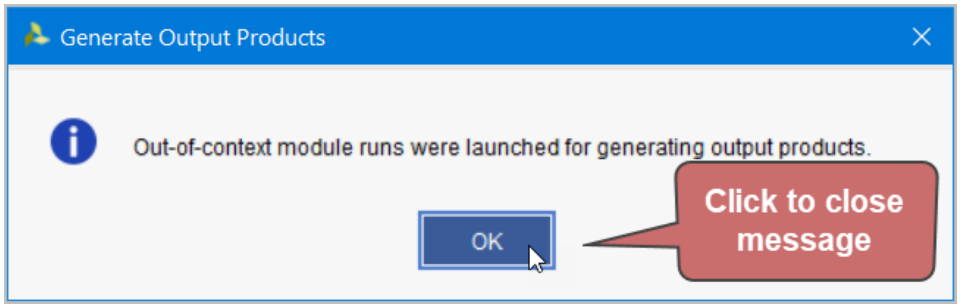


Figure 62. Click OK when the Generate Output Products dialog appears.

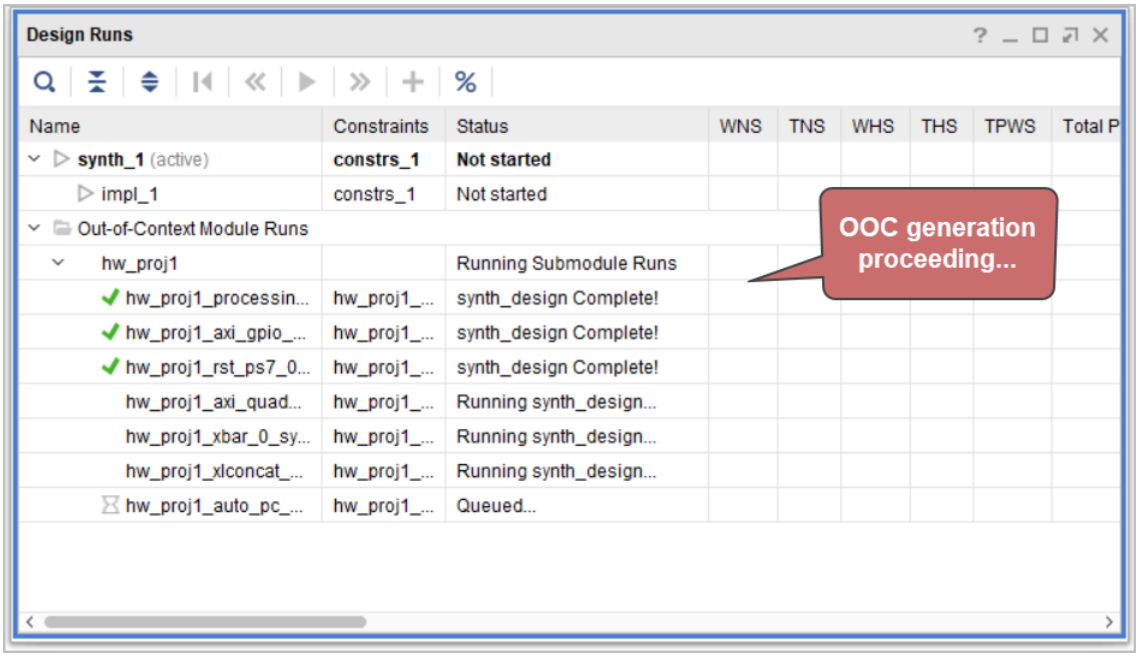


Figure 63. OOC generation runs...

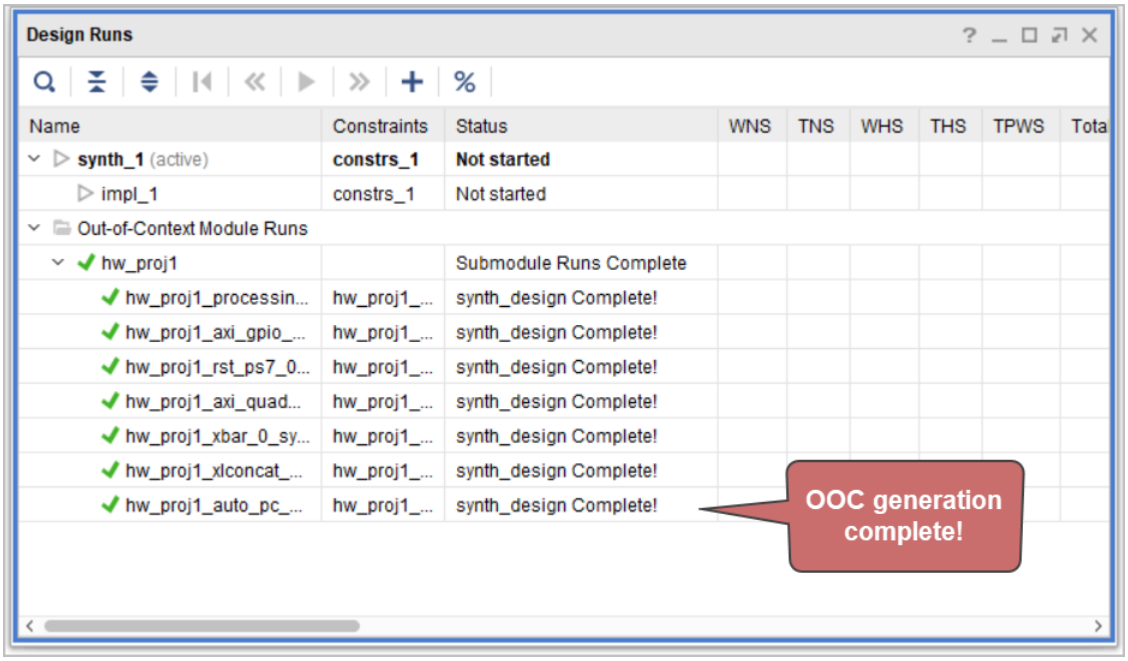


Figure 64. OOC Generation is complete.

2.3.2 Create HDL Wrapper

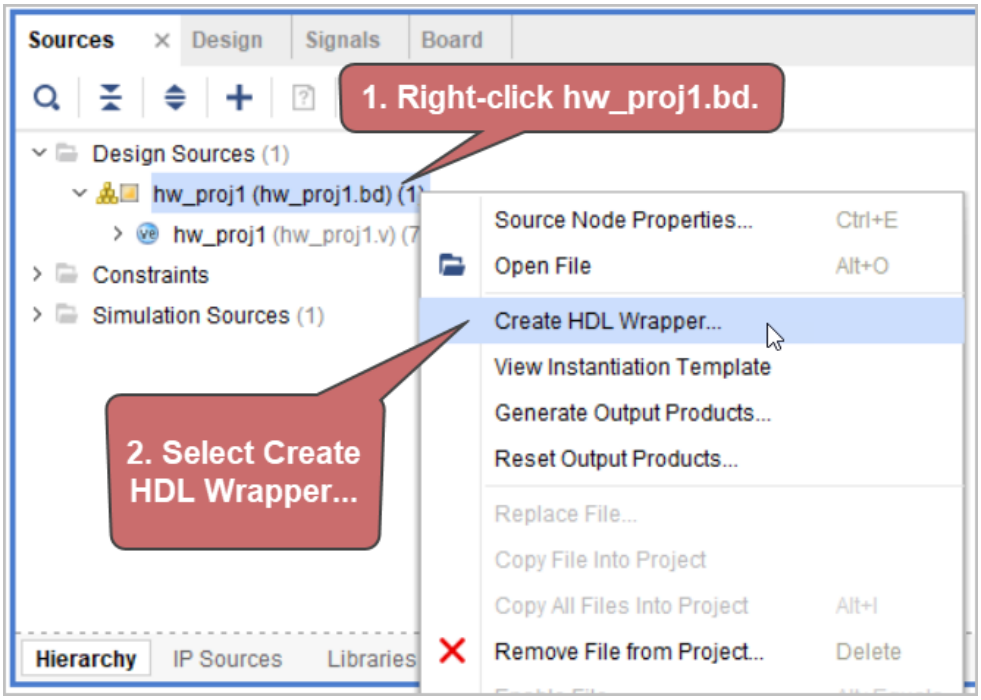


Figure 65. Create the HDL top-level wrapper.

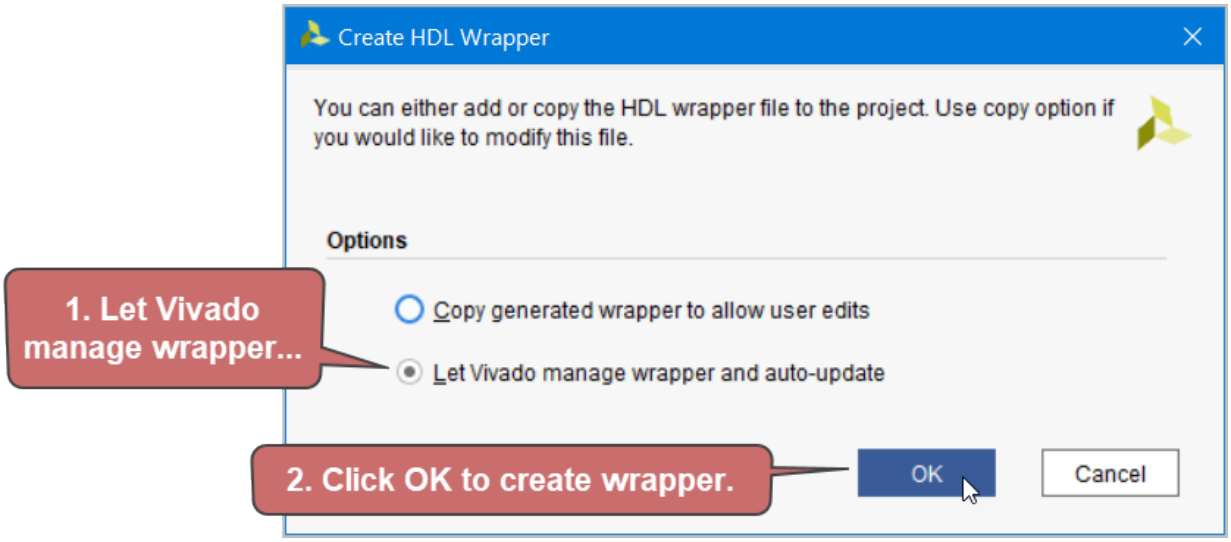


Figure 66. Let Vivado manage the wrapper, and click OK to continue.

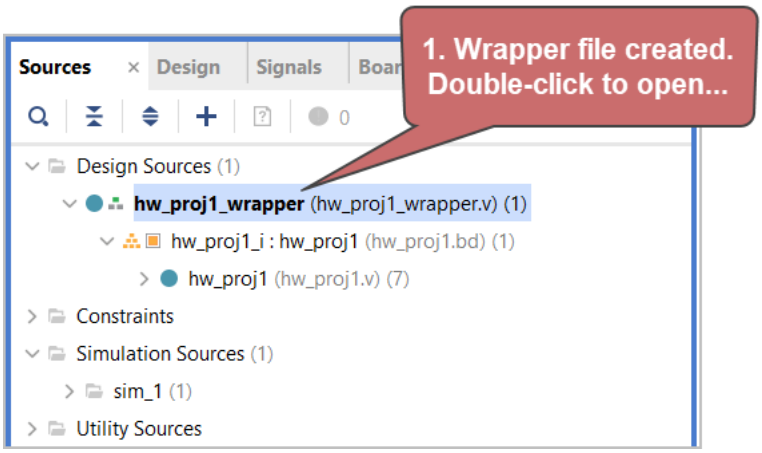


Figure 67. When created, double-click the wrapper to inspect it.

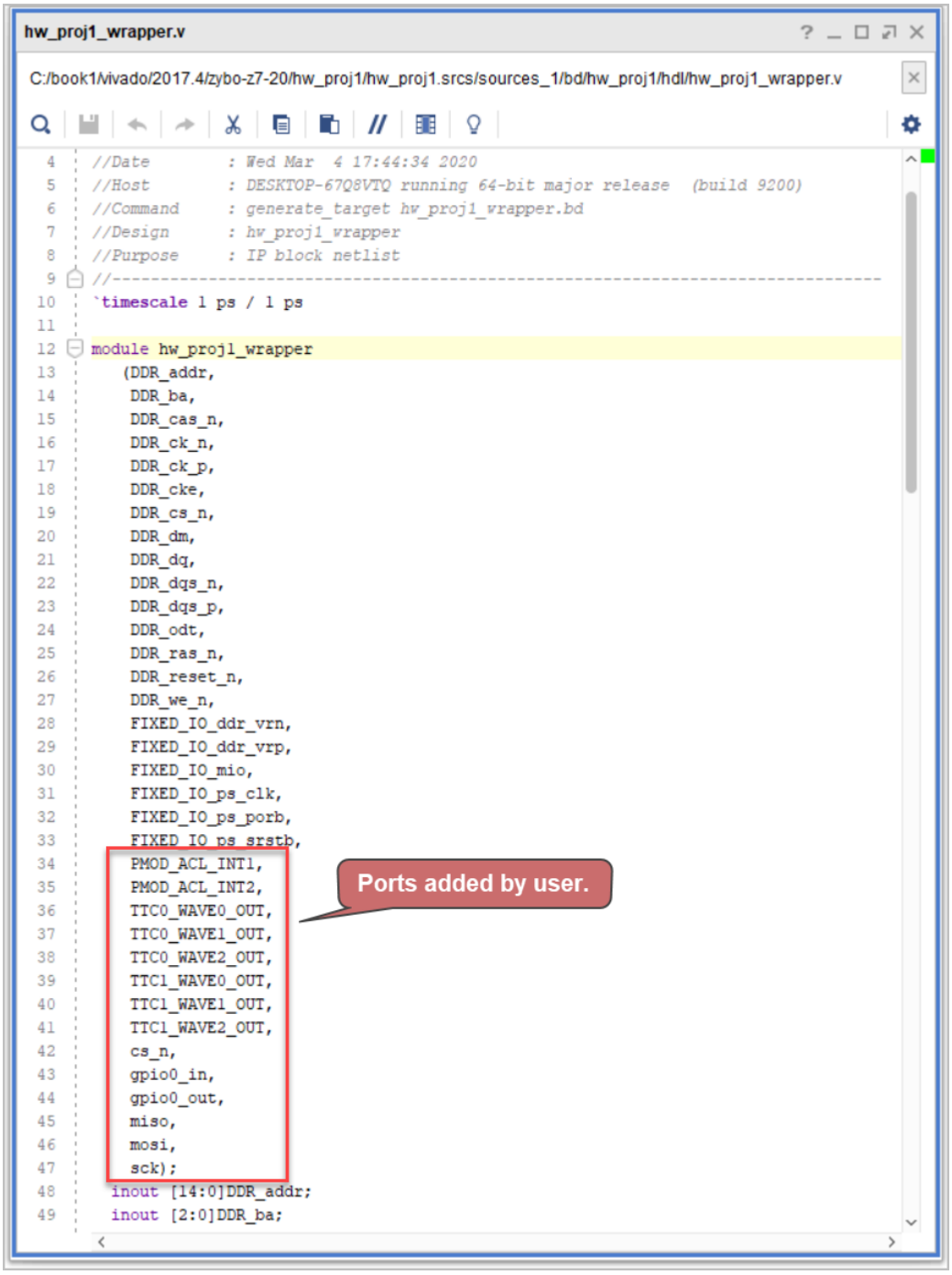


Figure 68. Check that the ports added by the user are present.

2.3.3 Add Constraints for the Project

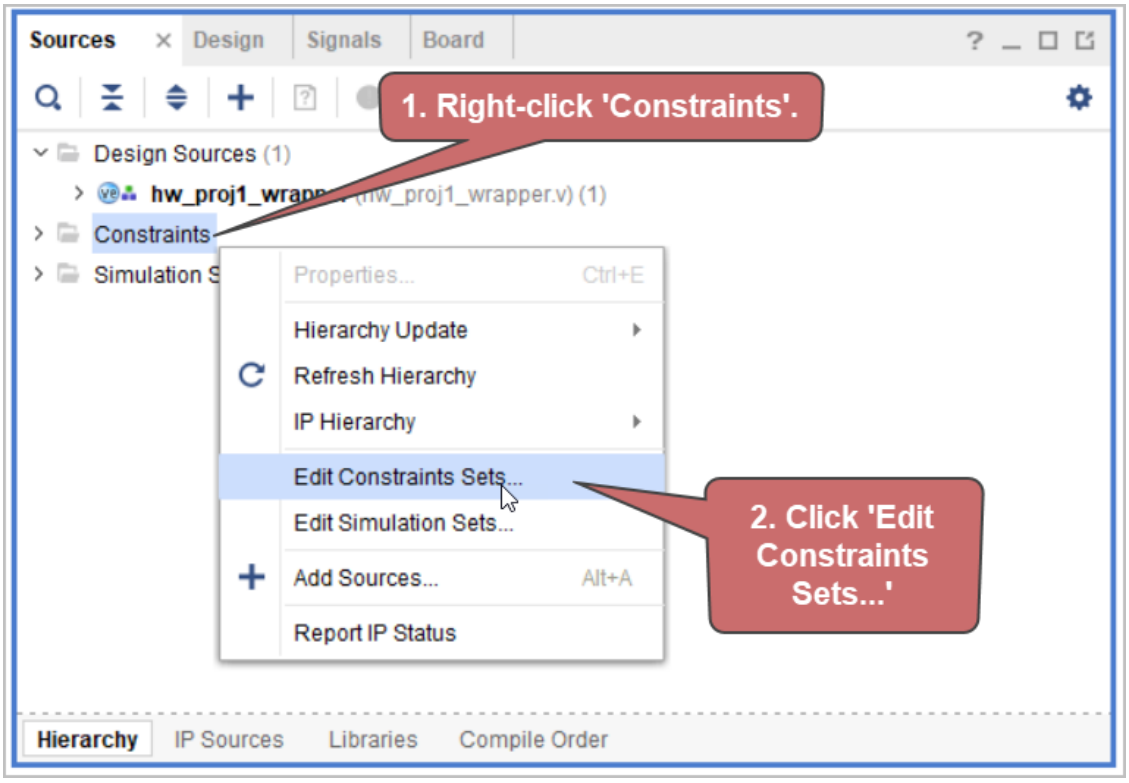


Figure 69. The first step in adding constraints (Right-click, and select "Edit Constraints Sets")

Figure 69 shows the first step in adding constraints for the project. Two options are then available to create the constraints file. The first is to simply to copy the constraints file into the project (Section 2.3.3.1). The second is to create a new file and add the constraints manually (Section 2.3.3.2). Each is covered in the coming pages.

Note that if the ZedBoard is being used, then the first option should be used. Simply choose the constraints file for the ZedBoard instead of the file for the Zybo-Z7 in Figure 71.

(See Section 1.3 for details on where to find the constraint files in GitHub.)

2.3.3.1 Option 1: Copy the constraints into the project

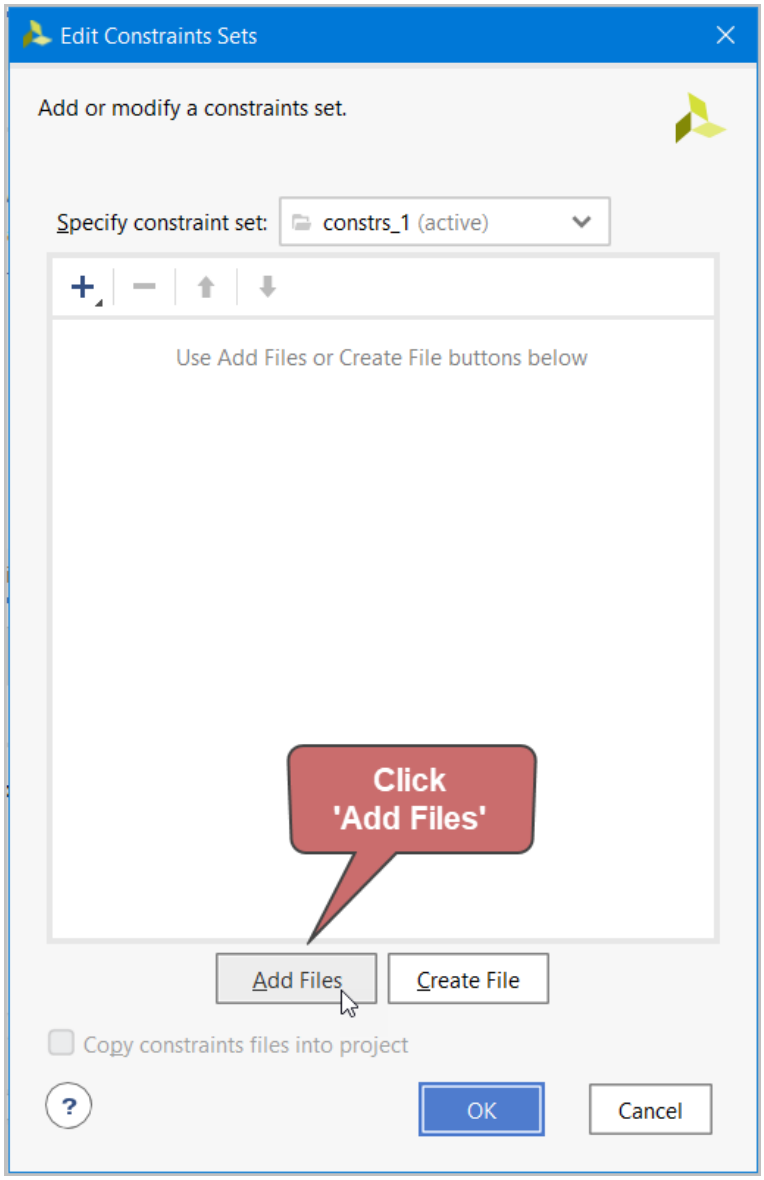


Figure 70. Click Add Files

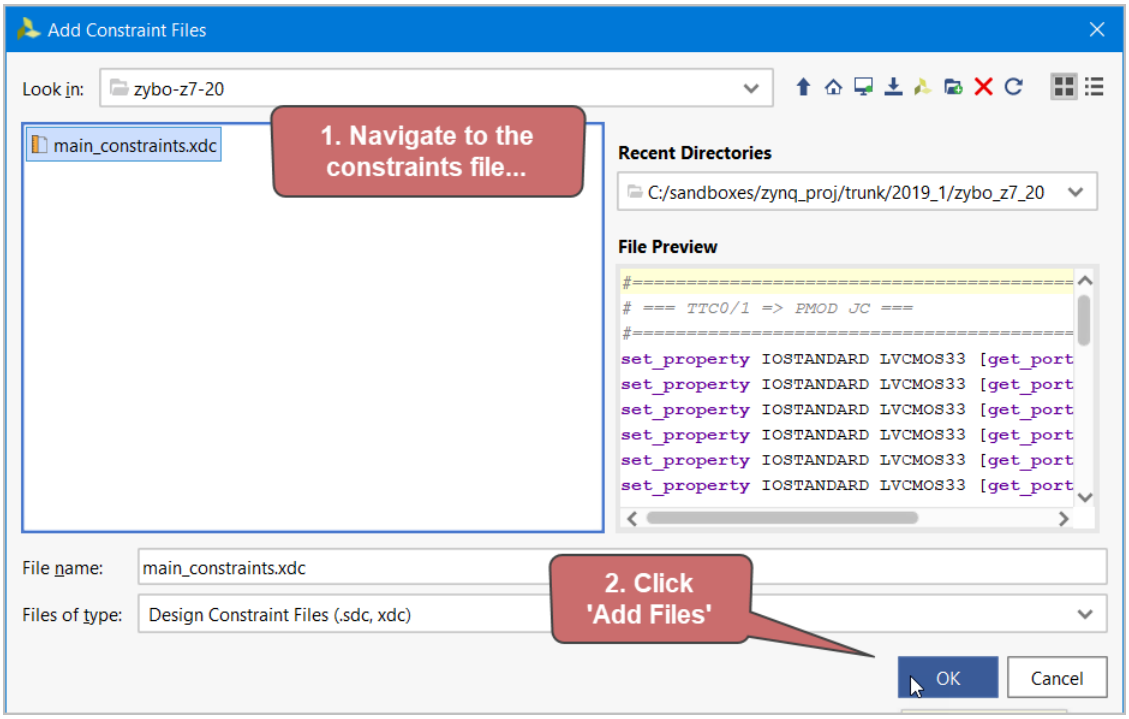


Figure 71. Browse to the supplied constraints file, and click OK to add it. (NOTE: for ZedBoard, browse to the 'zed' directory'_

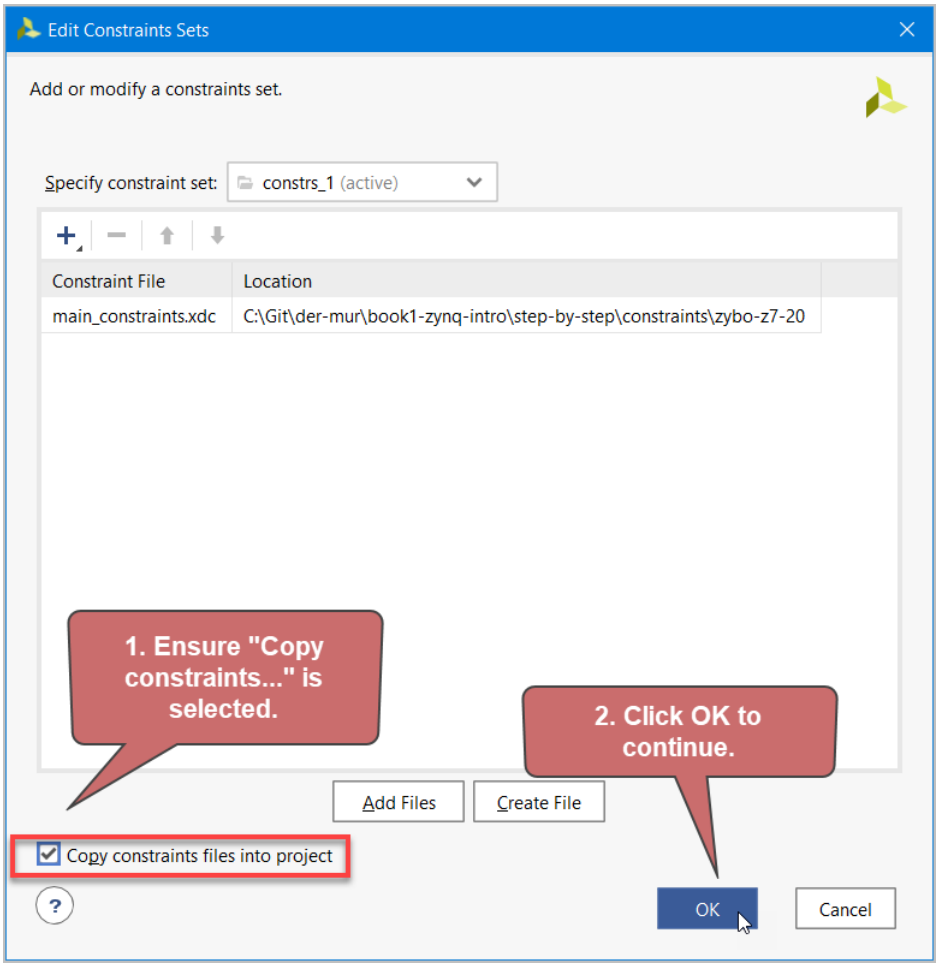


Figure 72. Click OK to continue

2.3.3.2 Option 2: Create the File and Add Constraints

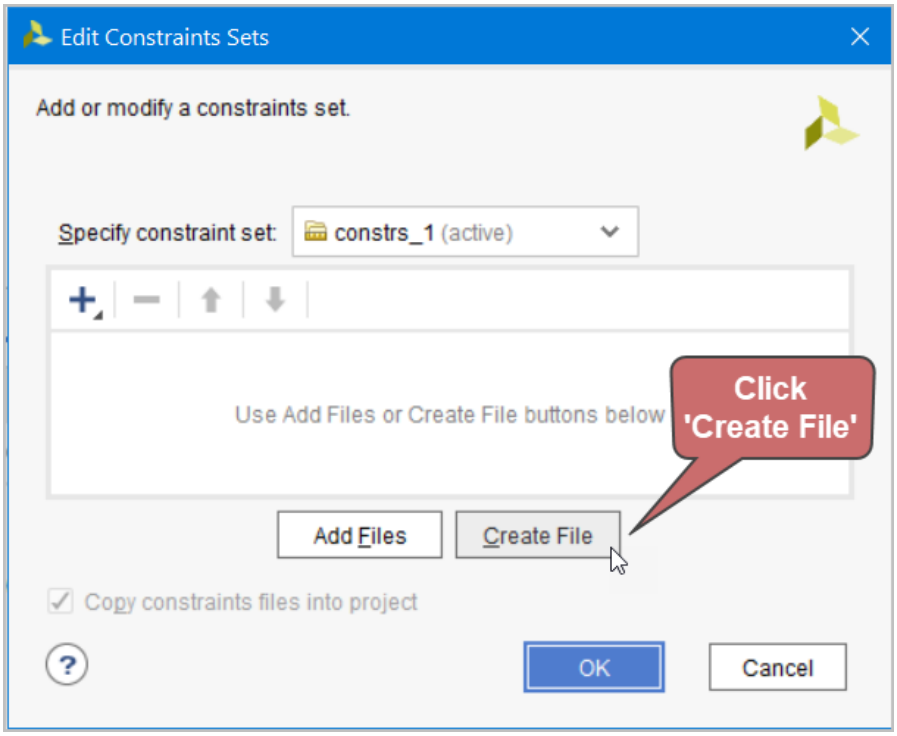


Figure 73. Create the file.

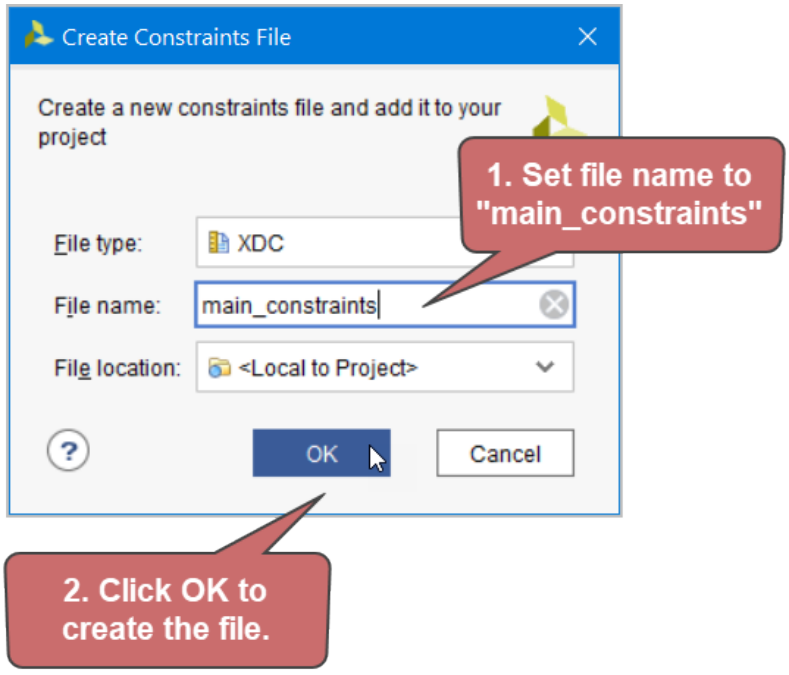


Figure 74. Name the file to "main_constraints" and click OK.

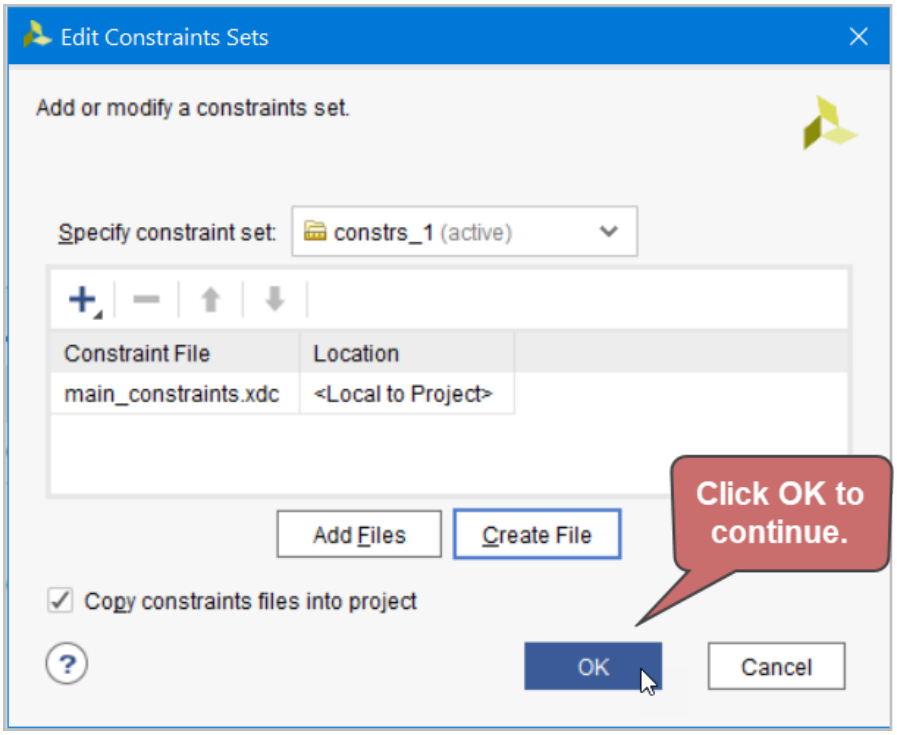


Figure 75. Click OK to add it to the project.

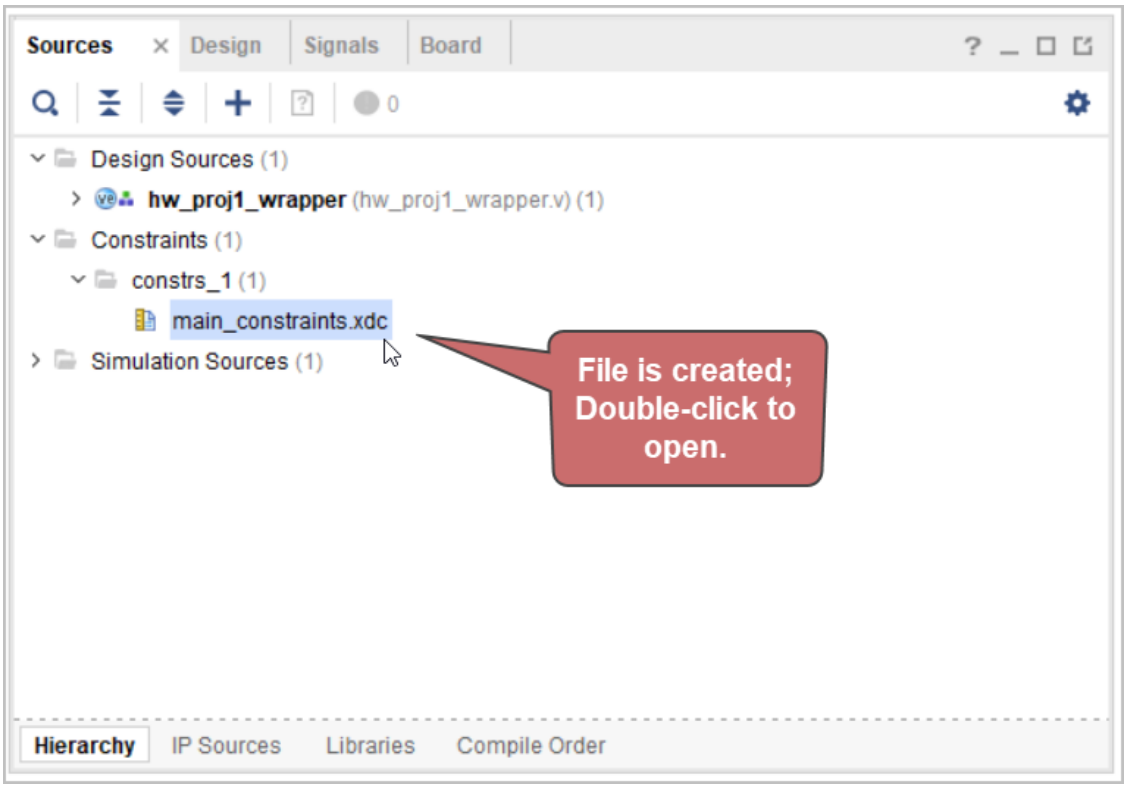


Figure 76. Open the constraints file.

Add (copy) the constraints as shown in the next two pages. (Note that these constraints are specifically for the Zybo-Z7-20 and Zybo-Z7-10 platforms! Do not use for ZedBoard.)

```
#============================================================#
# === AXI GPIO0 ===
#============================================================#
set_property IOSTANDARD LVCMOS33 [get_ports gpio0_out*]
set_property IOSTANDARD LVCMOS33 [get_ports gpio0_in*]

#===============
# gpio0_out[7:0]
#===============
# Board LEDs: LD0 - LD3
set_property PACKAGE_PIN M14 [get_ports {gpio0_out[0]}]
set_property PACKAGE_PIN M15 [get_ports {gpio0_out[1]}]
set_property PACKAGE_PIN G14 [get_ports {gpio0_out[2]}]
set_property PACKAGE_PIN D18 [get_ports {gpio0_out[3]}]

# PMOD JE Pins 1-4
set_property PACKAGE_PIN V12 [get_ports {gpio0_out[4]}]
set_property PACKAGE_PIN W16 [get_ports {gpio0_out[5]}]
set_property PACKAGE_PIN J15 [get_ports {gpio0_out[6]}]
set_property PACKAGE_PIN H15 [get_ports {gpio0_out[7]}]

#===============
# gpio0_in[11:0]
#===============
# Board Push-buttons: BTN0 - BTN3
set_property PACKAGE_PIN K18 [get_ports {gpio0_in[0]}]
set_property PACKAGE_PIN P16 [get_ports {gpio0_in[1]}]
set_property PACKAGE_PIN K19 [get_ports {gpio0_in[2]}]
set_property PACKAGE_PIN Y16 [get_ports {gpio0_in[3]}]

# Board Switches: SW0 - SW3
set_property PACKAGE_PIN G15 [get_ports {gpio0_in[4]}]
set_property PACKAGE_PIN P15 [get_ports {gpio0_in[5]}]
set_property PACKAGE_PIN W13 [get_ports {gpio0_in[6]}]
set_property PACKAGE_PIN T16 [get_ports {gpio0_in[7]}]

# PMOD JE Pins 5-8
set_property PACKAGE_PIN V13 [get_ports {gpio0_in[8]}]
set_property PACKAGE_PIN U17 [get_ports {gpio0_in[9]}]
set_property PACKAGE_PIN T17 [get_ports {gpio0_in[10]}]
set_property PACKAGE_PIN Y17 [get_ports {gpio0_in[11]}]
```

```
#=========================================================#
# === TTC0/1 => PMOD JC ===
#=========================================================#
set_property IOSTANDARD LVCMOS33 [get_ports TTC0_WAVE0_OUT]
set_property IOSTANDARD LVCMOS33 [get_ports TTC0_WAVE1_OUT]
set_property IOSTANDARD LVCMOS33 [get_ports TTC0_WAVE2_OUT]
set_property IOSTANDARD LVCMOS33 [get_ports TTC1_WAVE0_OUT]
set_property IOSTANDARD LVCMOS33 [get_ports TTC1_WAVE1_OUT]
set_property IOSTANDARD LVCMOS33 [get_ports TTC1_WAVE2_OUT]

set_property PACKAGE_PIN V15 [get_ports TTC0_WAVE0_OUT]
set_property PACKAGE_PIN W15 [get_ports TTC0_WAVE1_OUT]
set_property PACKAGE_PIN T11 [get_ports TTC0_WAVE2_OUT]
# PIN 4 (T10) NOT USED
set_property PACKAGE_PIN W14 [get_ports TTC1_WAVE0_OUT]
set_property PACKAGE_PIN Y14 [get_ports TTC1_WAVE1_OUT]
set_property PACKAGE_PIN T12 [get_ports TTC1_WAVE2_OUT]
# PIN 10 (U12) NOT USED

#=========================================================#
# === PmodACL => PMOD JD ===
#=========================================================#
set_property IOSTANDARD LVCMOS33 [get_ports cs_n]
set_property IOSTANDARD LVCMOS33 [get_ports mosi]
set_property IOSTANDARD LVCMOS33 [get_ports miso]
set_property IOSTANDARD LVCMOS33 [get_ports sclk]
set_property IOSTANDARD LVCMOS33 [get_ports PMOD_ACL_INT2]
set_property IOSTANDARD LVCMOS33 [get_ports PMOD_ACL_INT1]

# cs_n => PMOD JD PIN 1 = T14
# mosi => PMOD JD PIN 2 = T15
# miso => PMOD JD PIN 3 = P14
# sclk => PMOD JD PIN 4 = R14
# PMOD_ACL_INT2 => PMOD JD PIN 7 = U14
# PMOD_ACL_INT1 => PMOD JD PIN 8 = U15
set_property PACKAGE_PIN T14 [get_ports cs_n]
set_property PACKAGE_PIN T15 [get_ports mosi]
set_property PACKAGE_PIN P14 [get_ports miso]
set_property PACKAGE_PIN R14 [get_ports sclk]
set_property PACKAGE_PIN U14 [get_ports PMOD_ACL_INT2]
set_property PACKAGE_PIN U15 [get_ports PMOD_ACL_INT1]
```

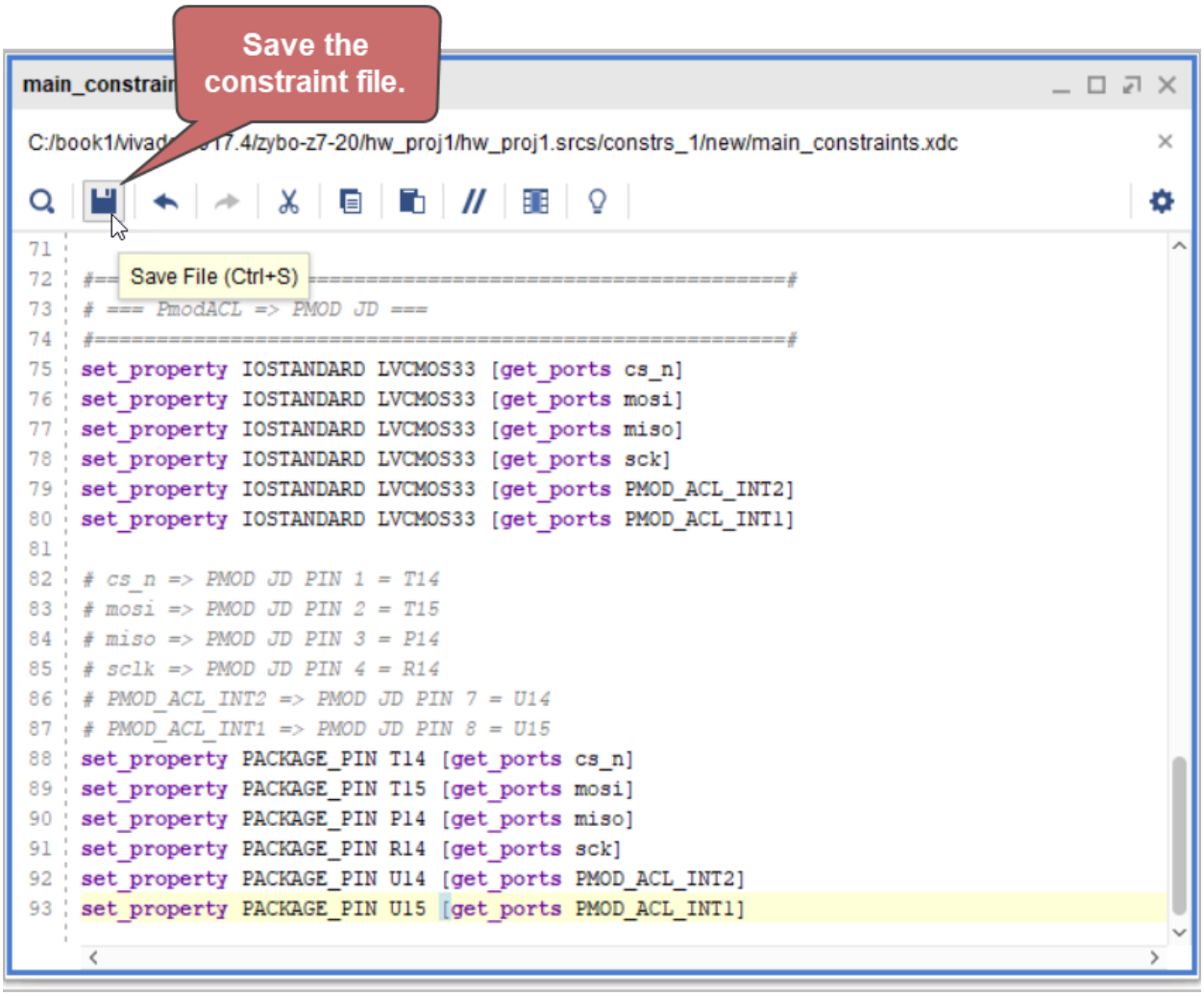


Figure 77. Save the constraints file.

2.4 Design Generation

2.4.1 Synthesize the Design

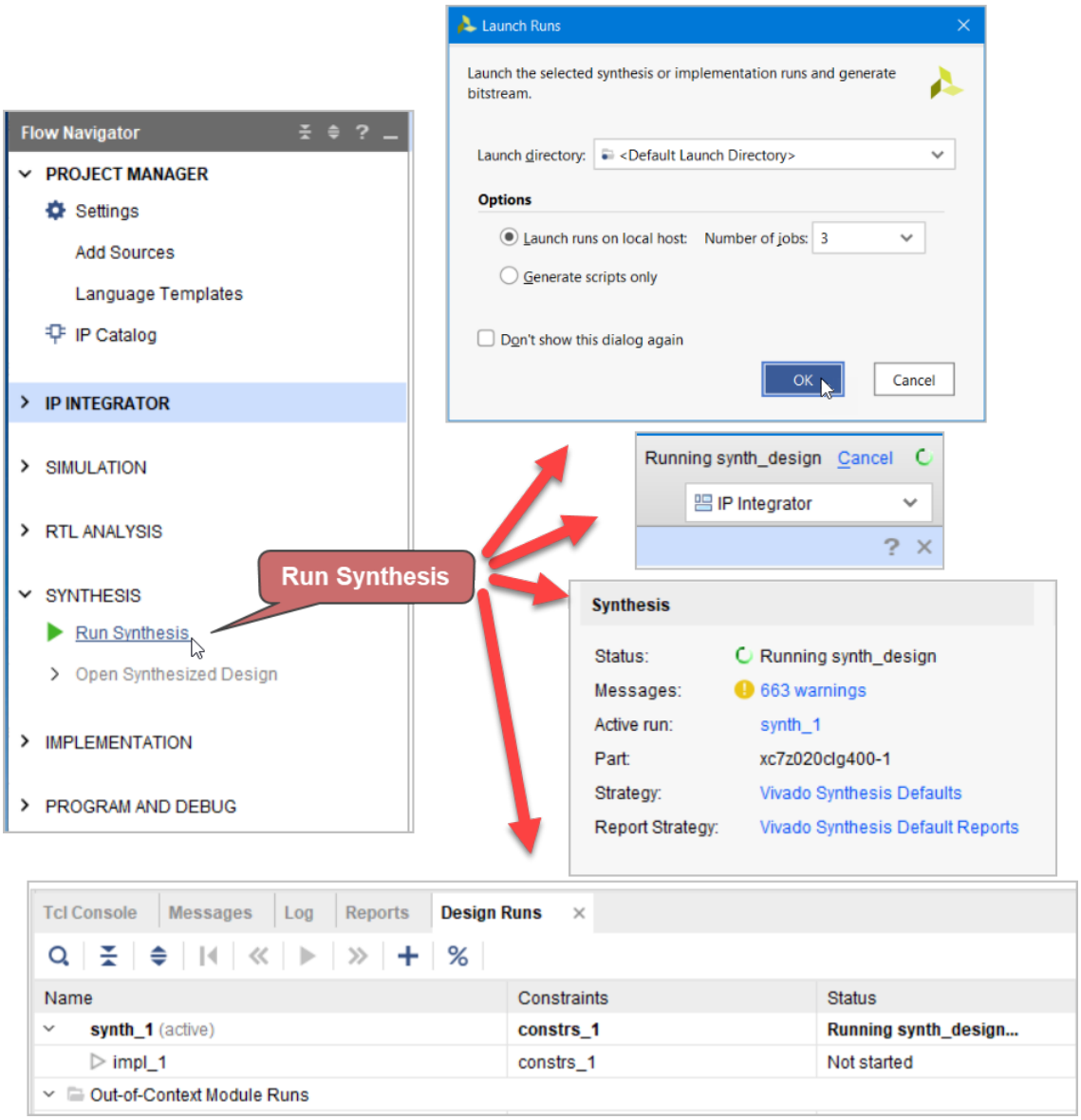


Figure 78. Run design synthesis.

2.4.2 Implement the Design

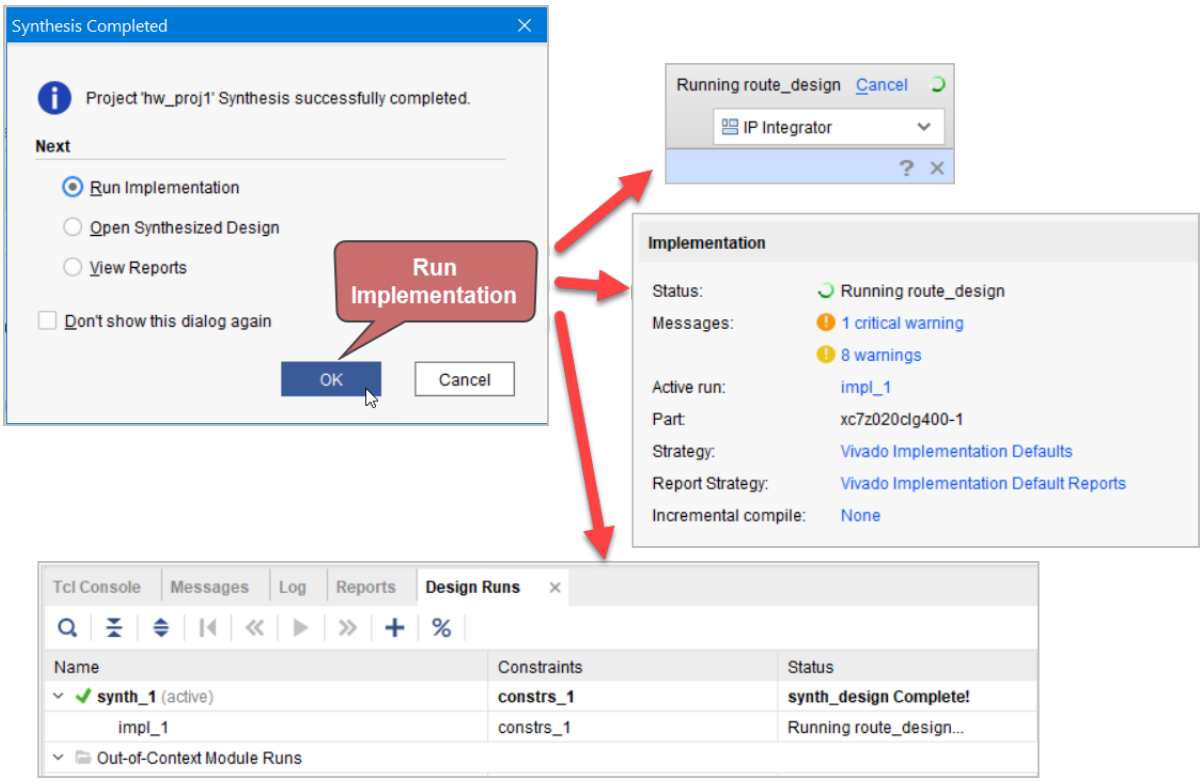


Figure 79. Implement the design.

2.4.3 Generate the Bitstream

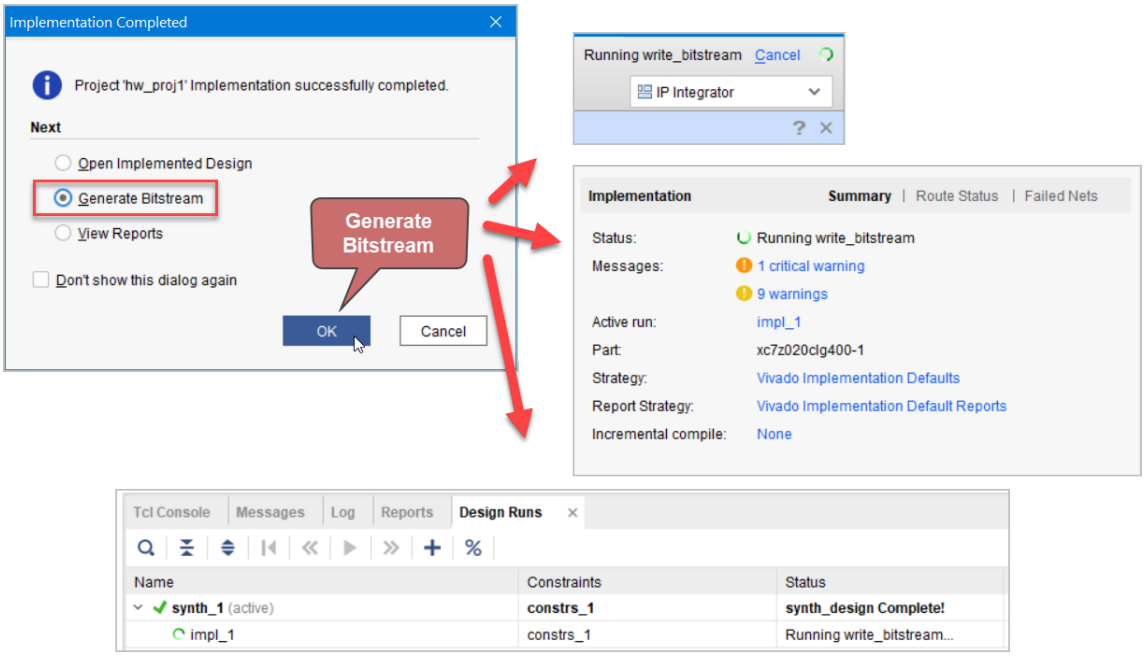


Figure 80. Generate the bitstream.

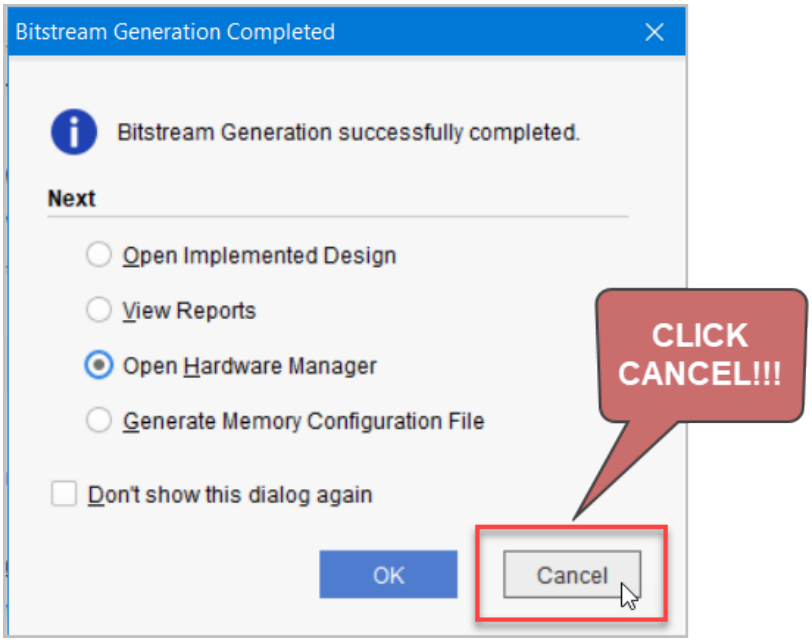


Figure 81. When the bitstream is generated, click Cancel (unless the user wants to open the HW Manager).

2.5 Hardware Hand-off

2.5.1 Export the Design

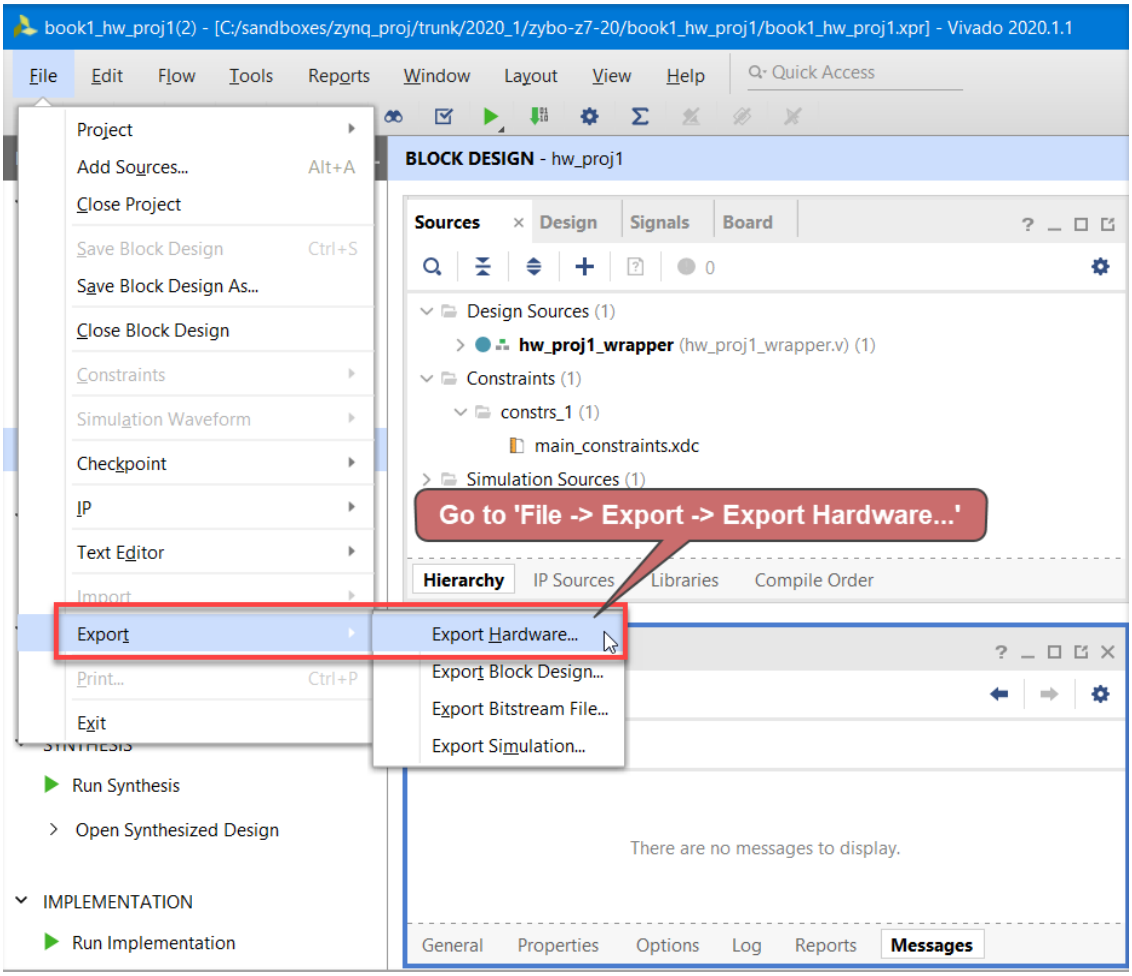


Figure 82. Export the design.

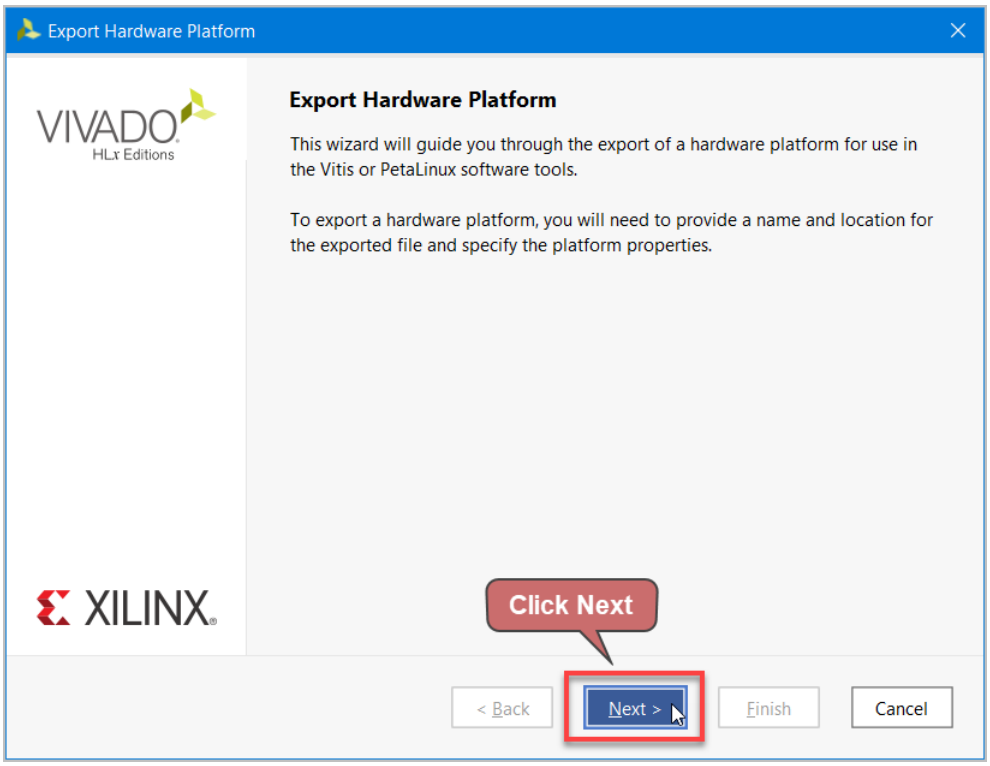


Figure 83. Click Next to continue.

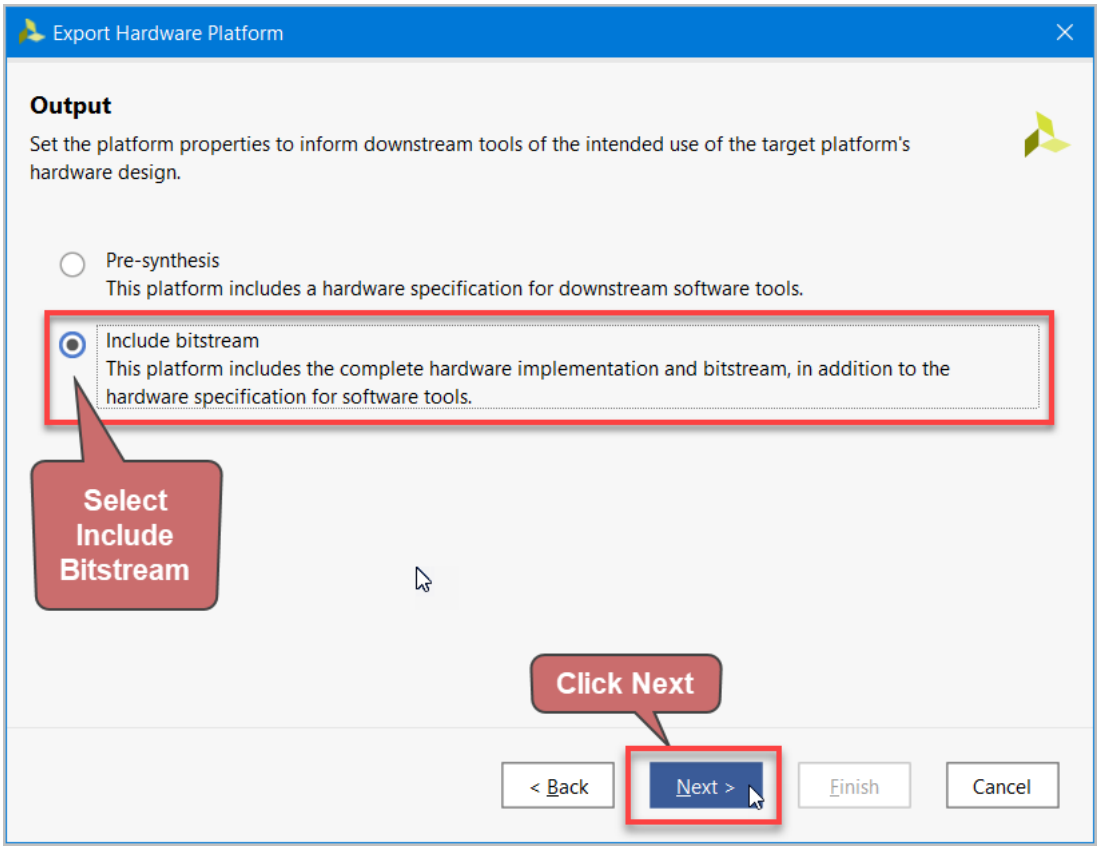


Figure 84. Select *Include bitstream* and then click *Next* to continue.

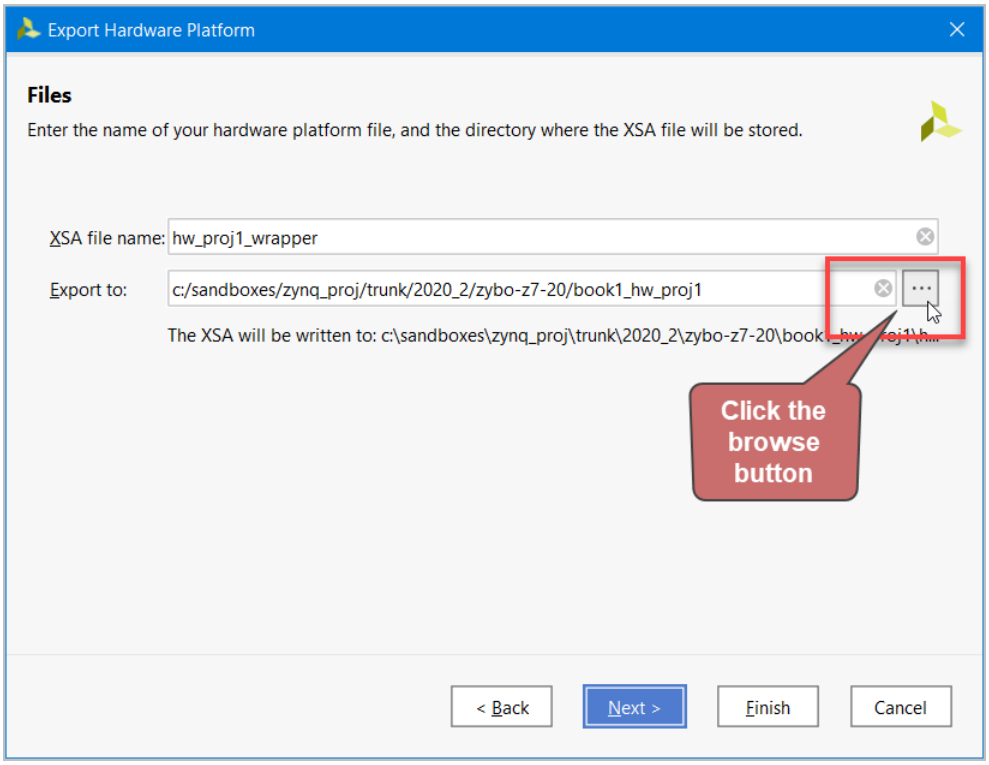


Figure 85. Click the *Browse* button.

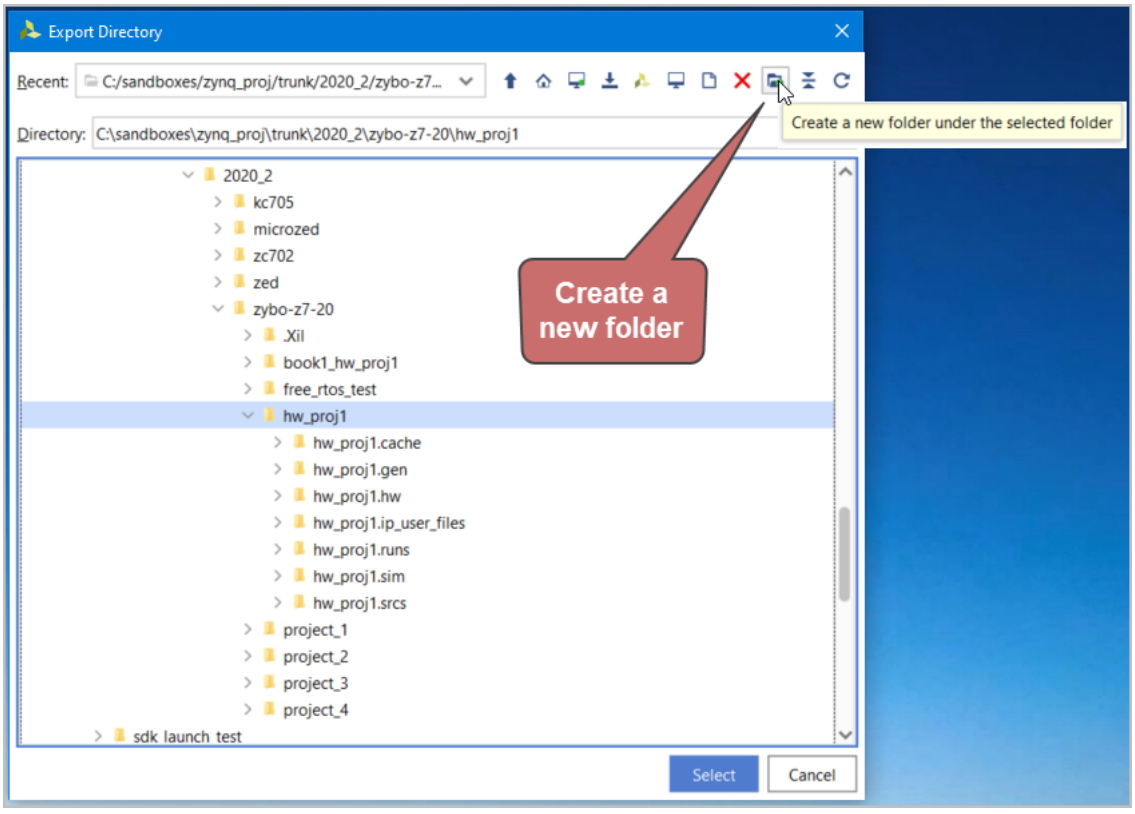


Figure 86. Select the option to create a new folder.

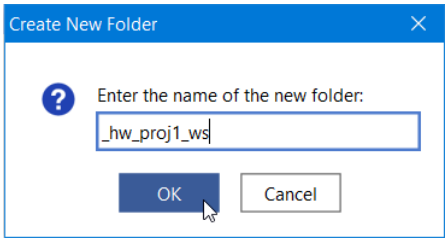


Figure 87. Enter "_hw_proj1_ws" as the folder name and click OK.

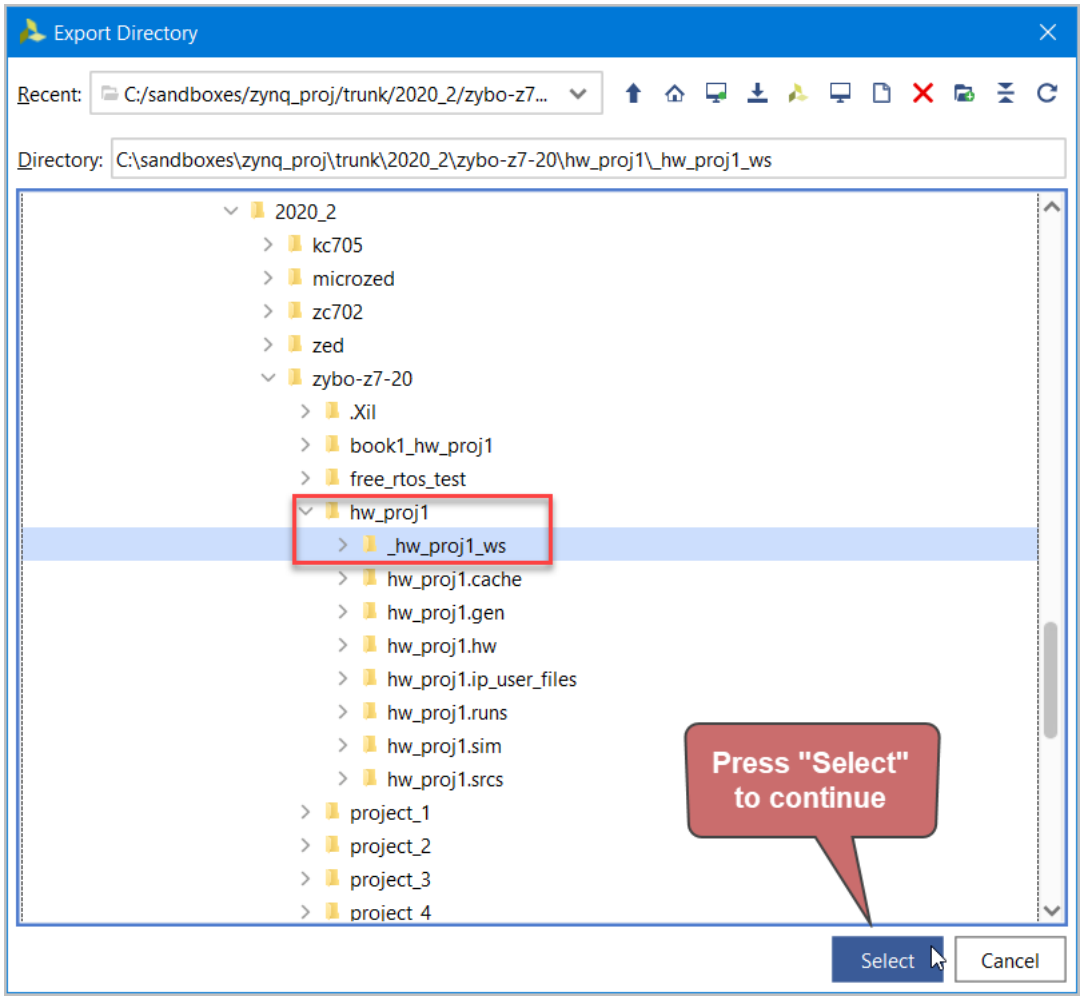


Figure 88. The folder is added. Press *Select* to continue.

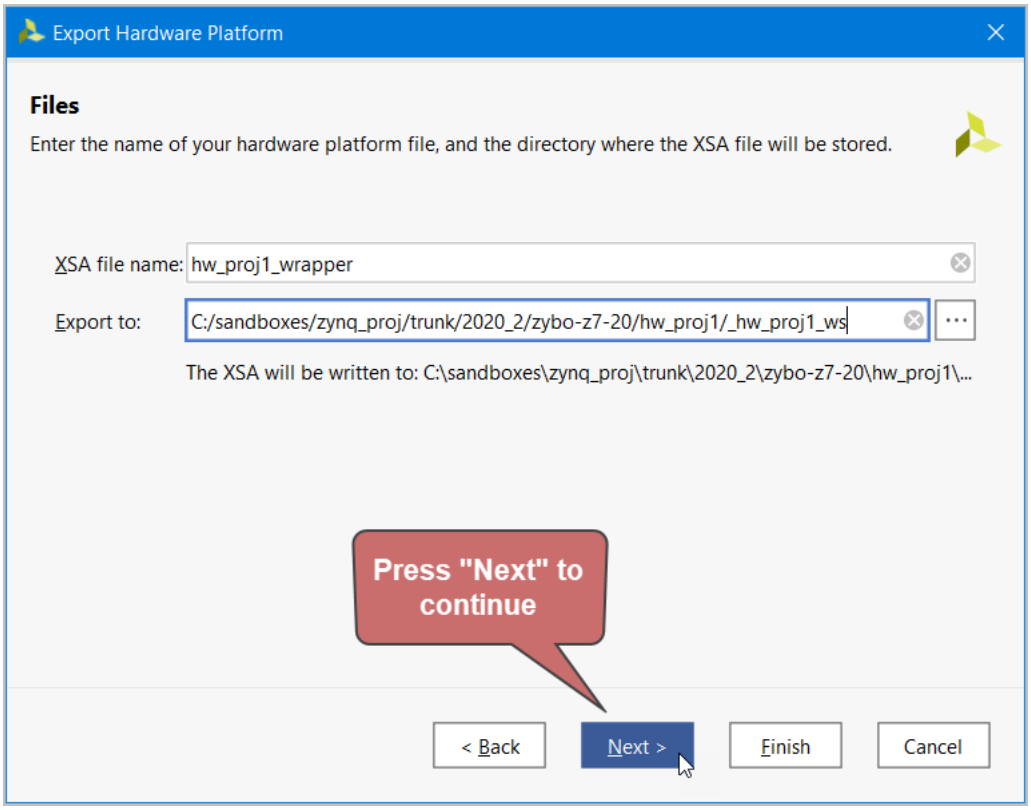


Figure 89. Click *Next* to continue.

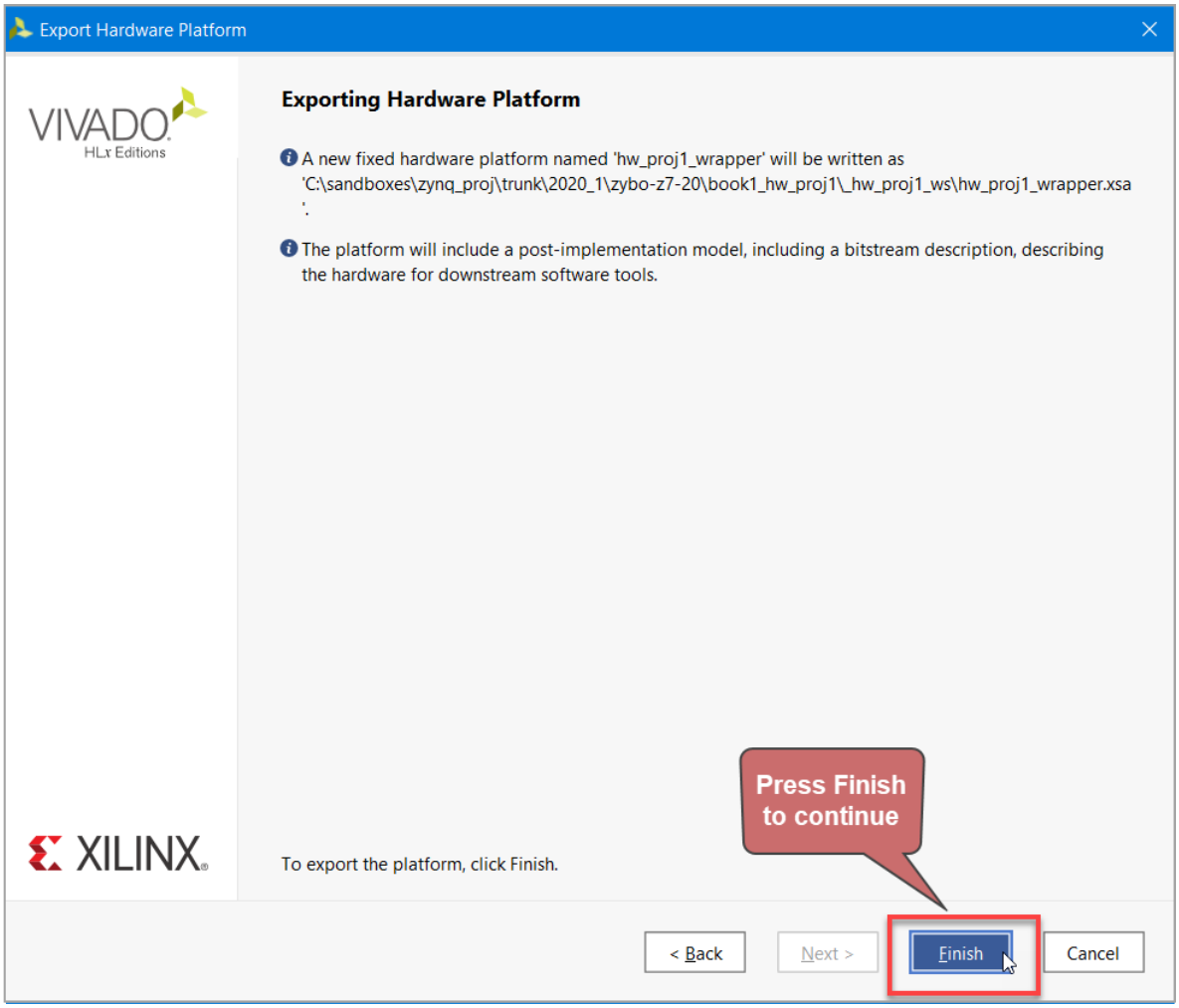


Figure 90. Select *Finish*.

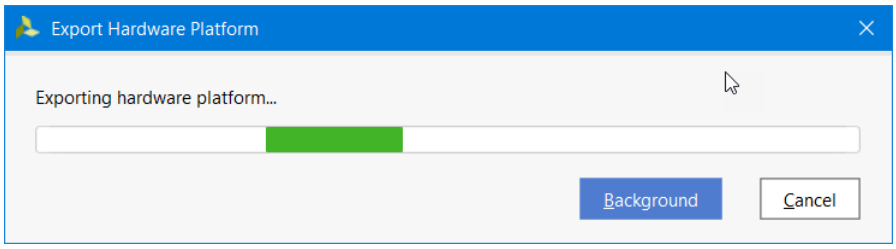


Figure 91. The platform is exported...

2.5.2 Launch Vitis

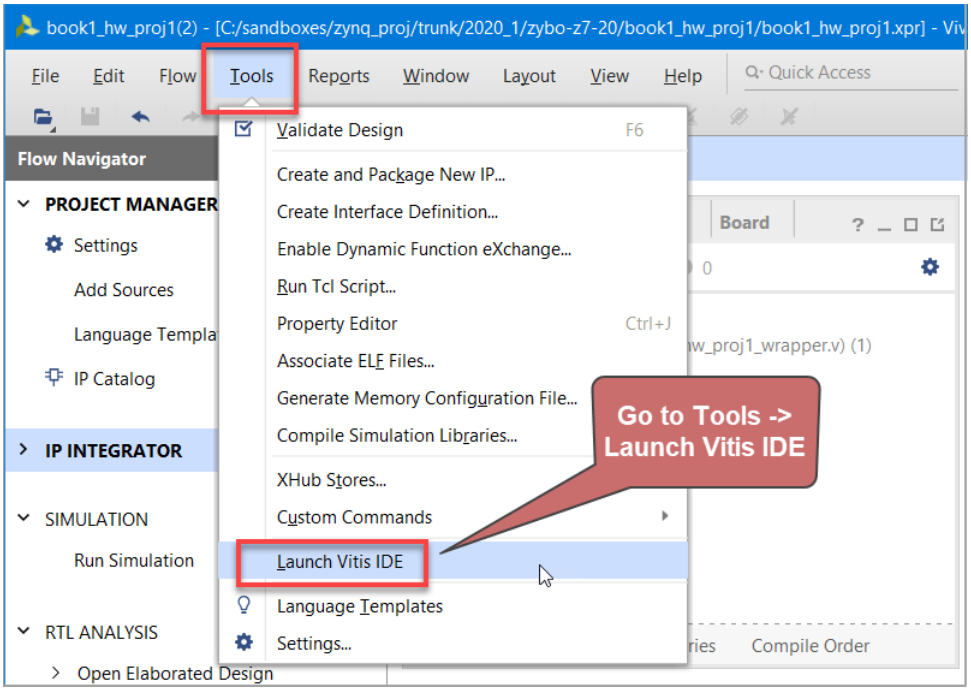


Figure 92. Go to *Tools -> Launch Vitis IDE*.

3 Software Development in Vitis

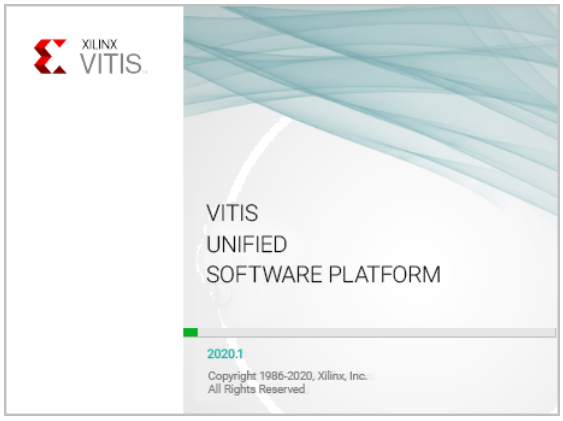


Figure 93. The VITIS splash-screen appears.

3.1 Select a Workspace

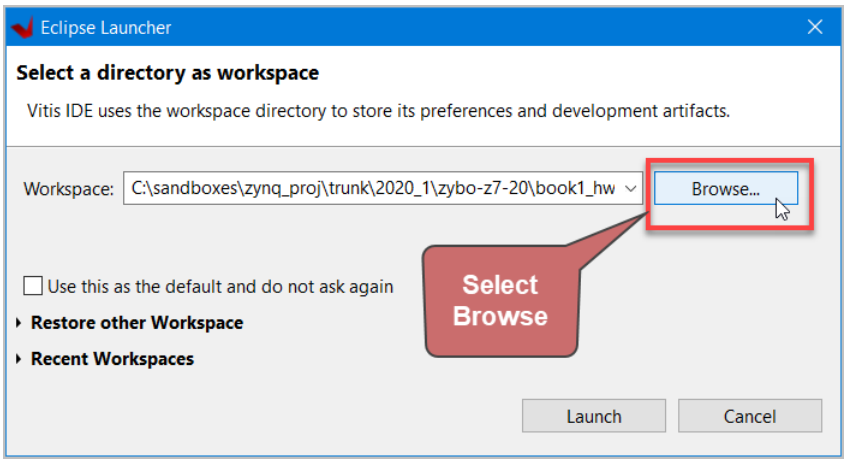


Figure 94. Select Browse in the Launcher dialog.

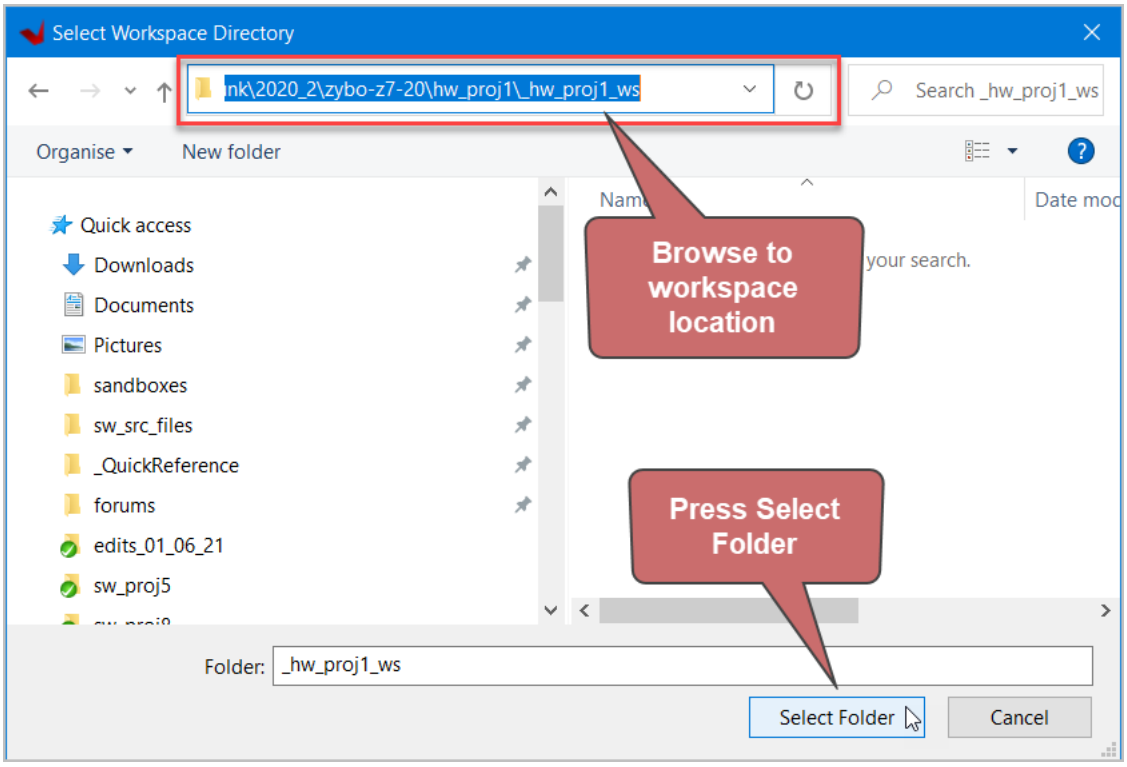


Figure 95. Navigate to the folder created during the hardware hand-off procedure.

3.2 Launch Vitis

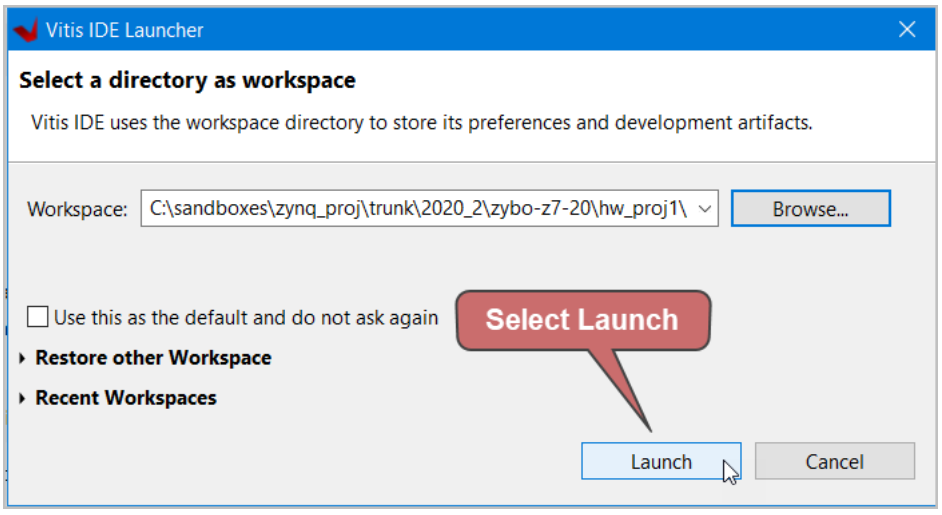


Figure 96. Launch the IDE.

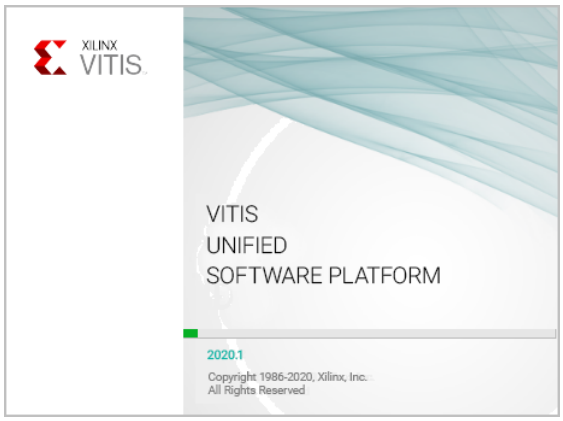


Figure 97. Wait while Vitis continues to launch...

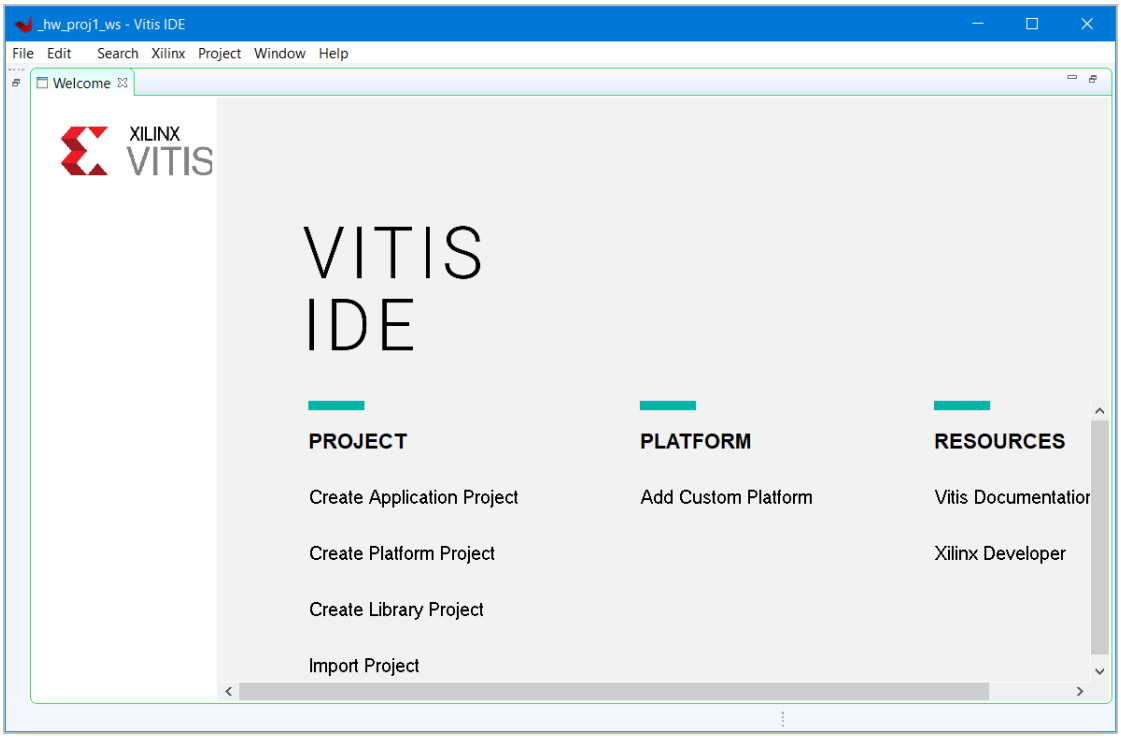


Figure 98. Eventually, the VITIS IDE will be ready.

3.3 Create a Platform Project

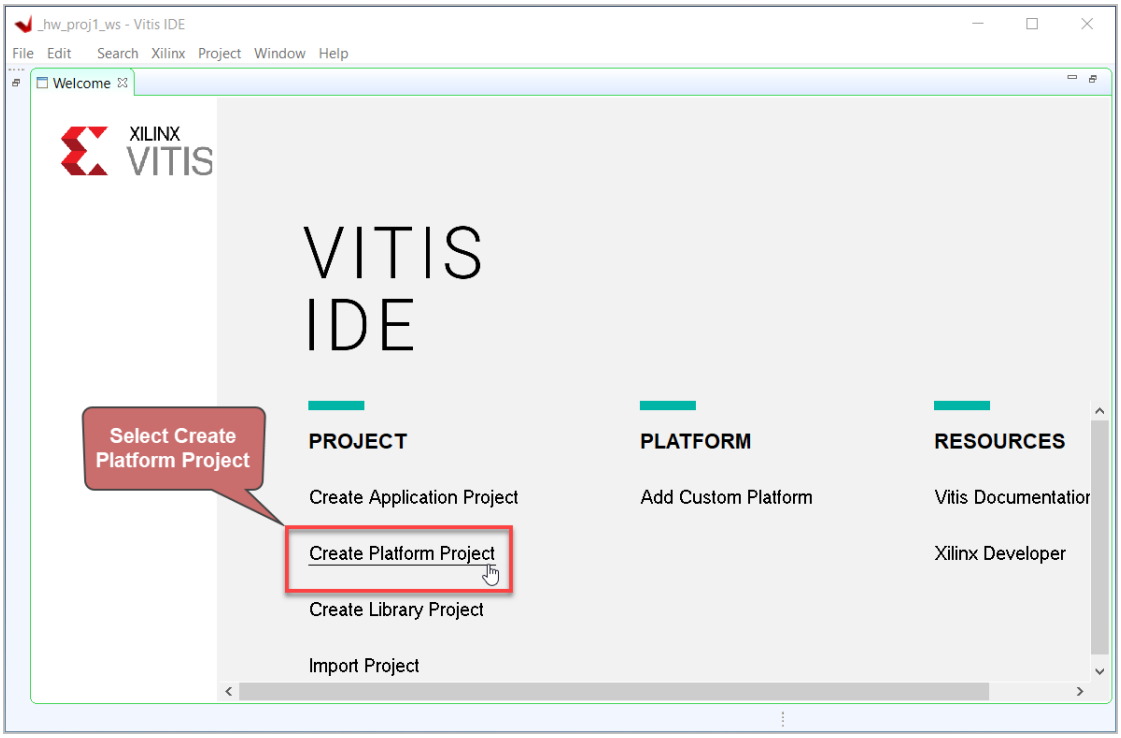


Figure 99. Select *Create Platform Project.*

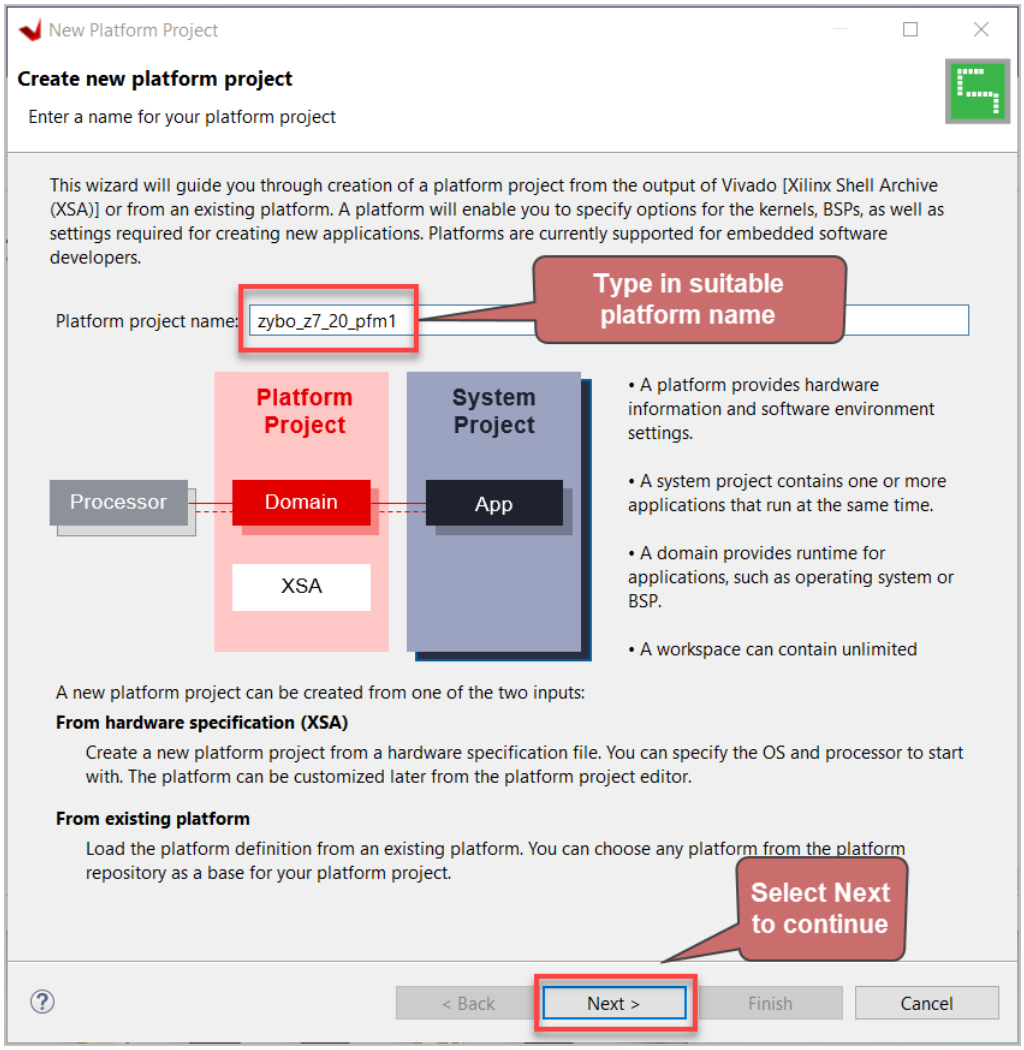


Figure 100. Name the platform. In this case it is called *zybo_z7_20_pfm1*

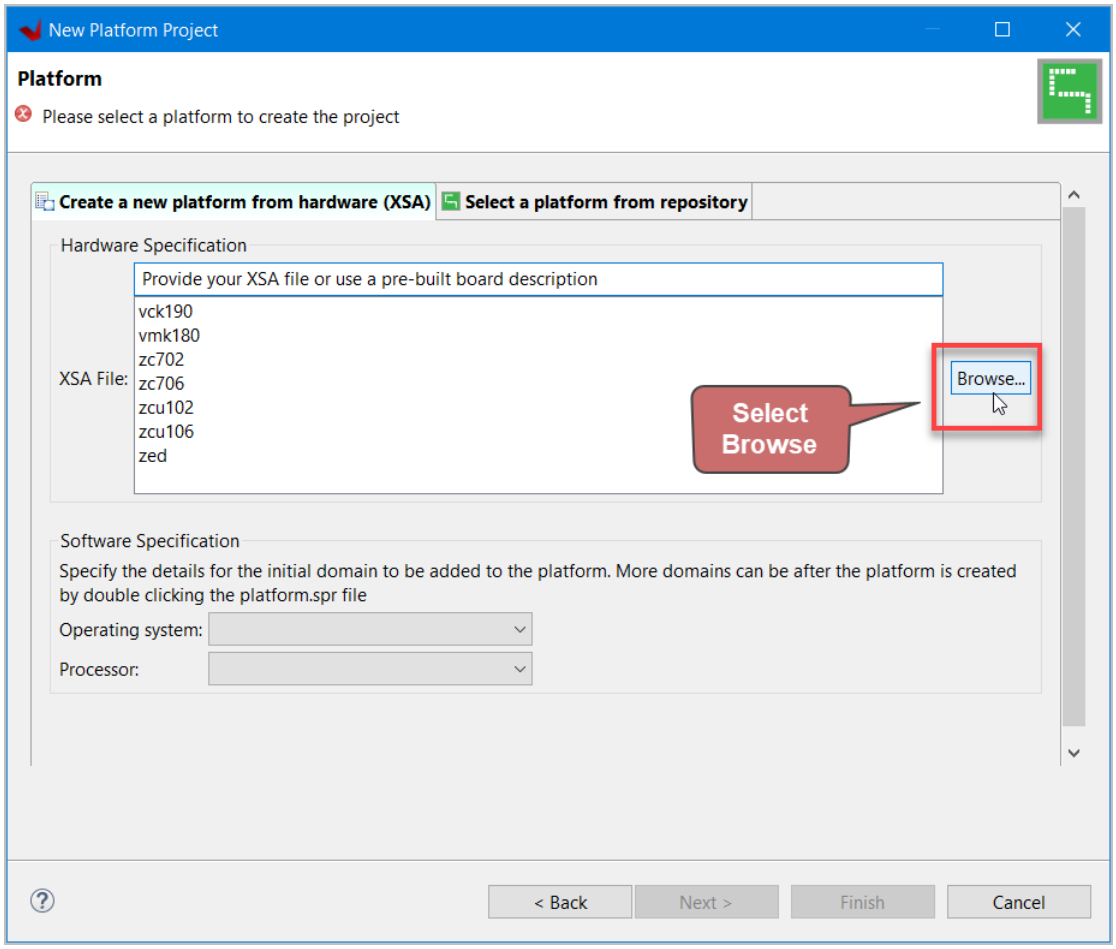


Figure 101. Select *Browse* to navigate to the hardware platform created earlier.

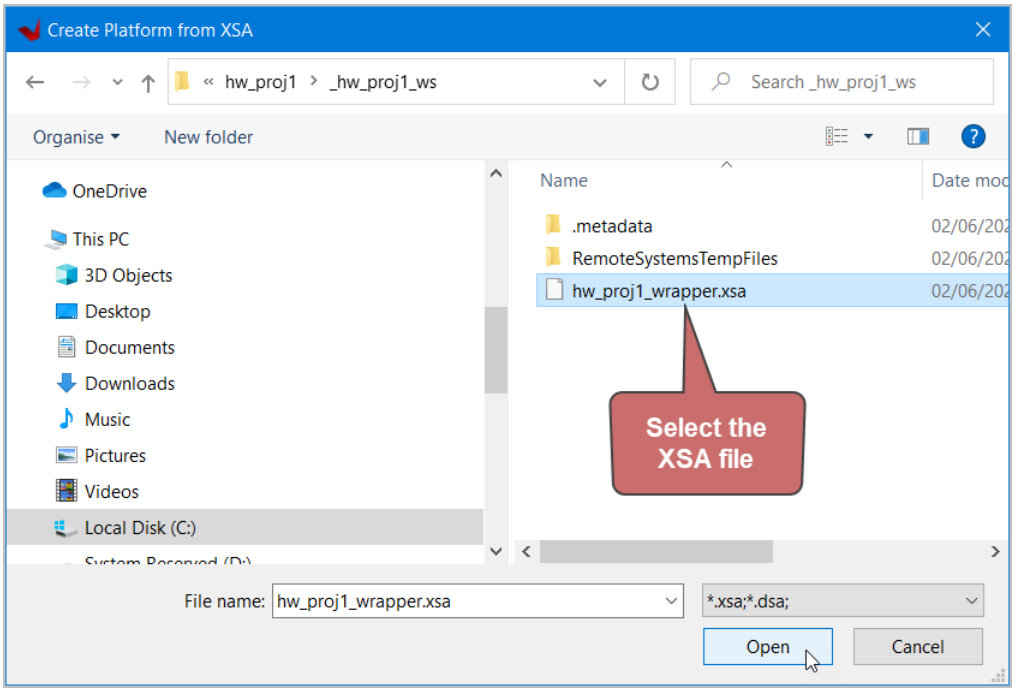


Figure 102. Select the XSA file.

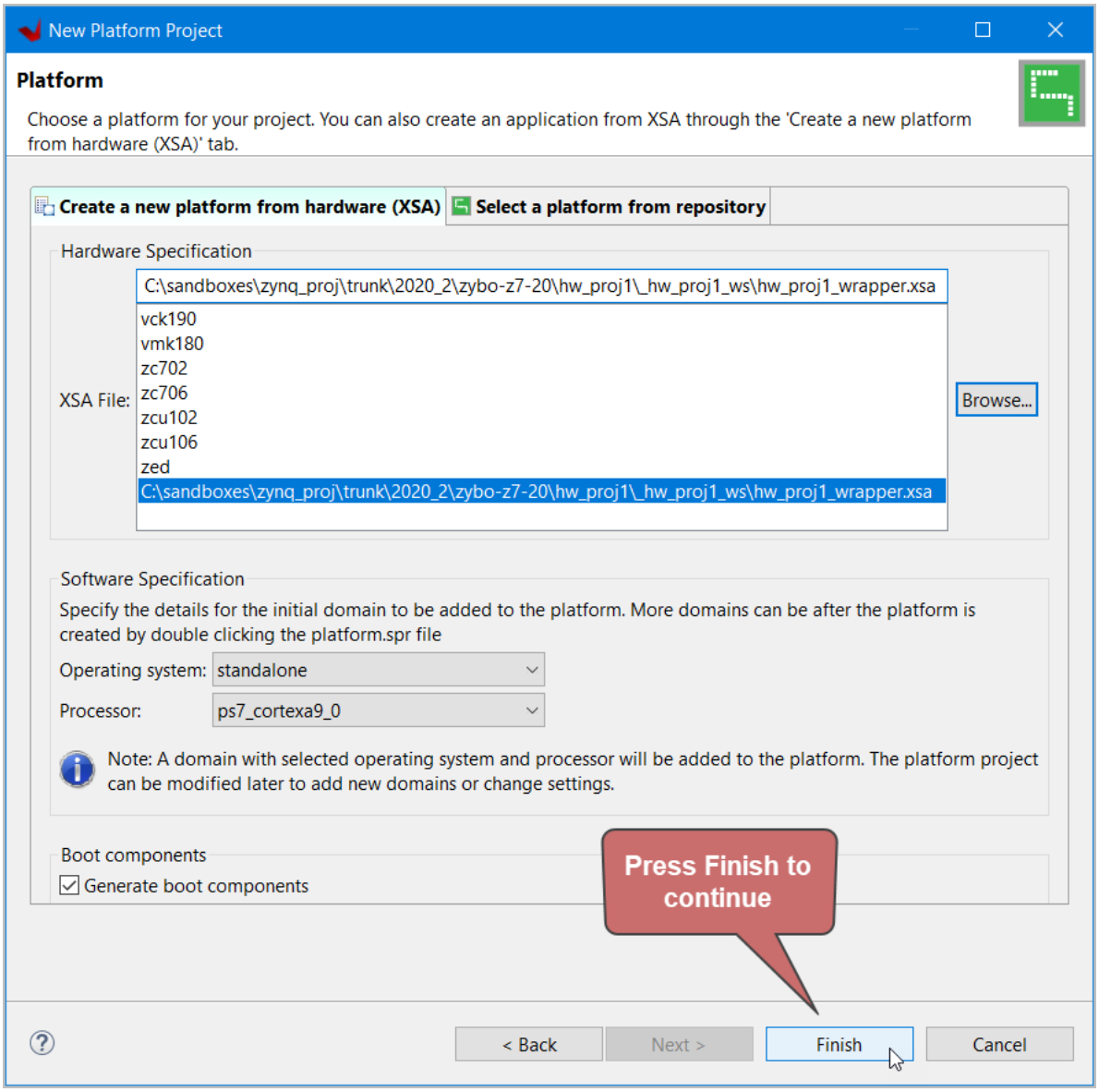


Figure 103. Click *Finish* to continue.

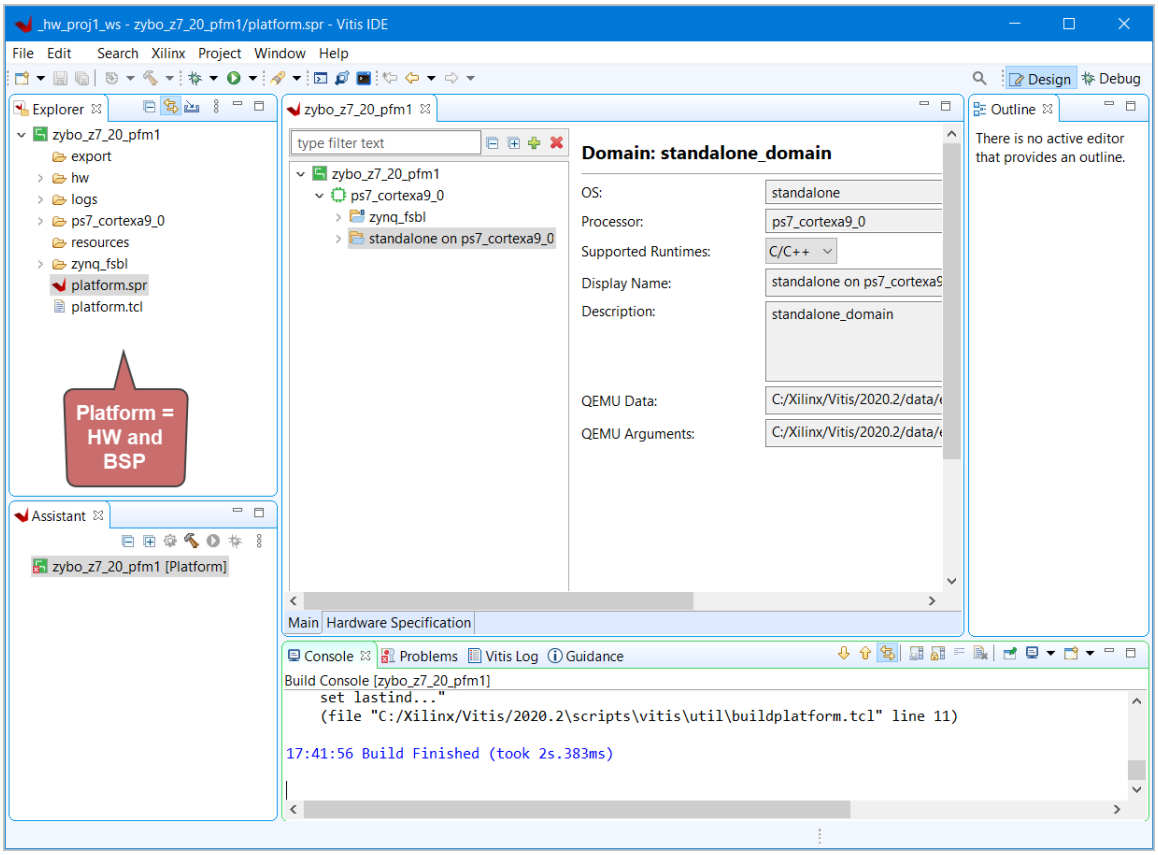


Figure 104. The platform is now in the Vitis IDE workspace.

3.4 System (Software) Project 1: Hello World

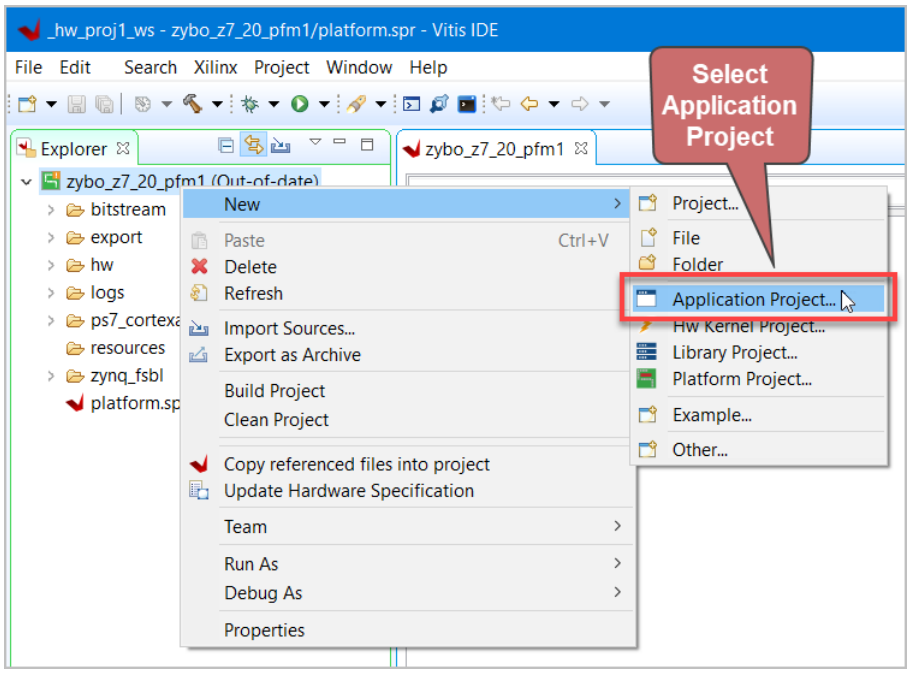


Figure 105. Right-click on the platform, and select New -> Application Project

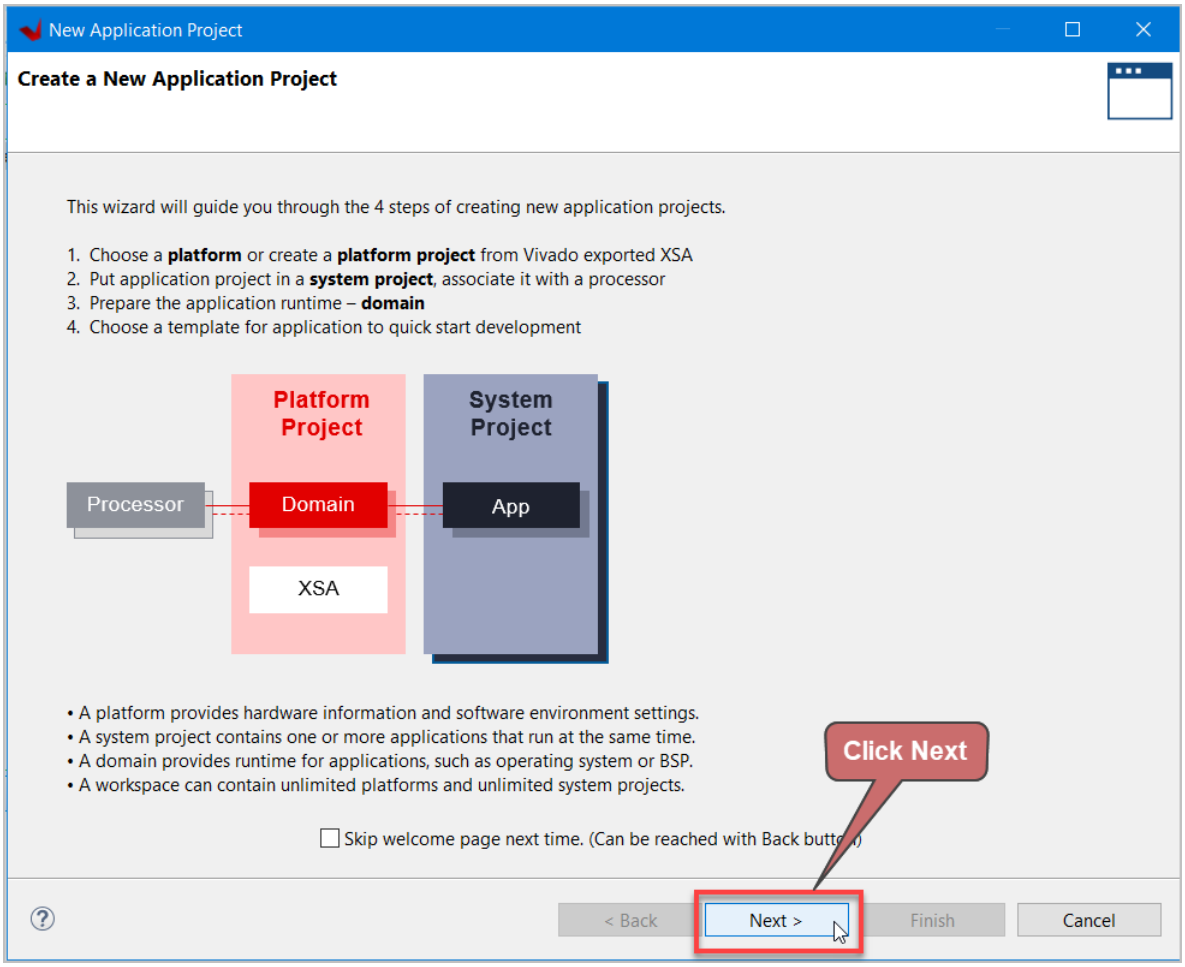


Figure 106. Click Next to start creating the System/Application project

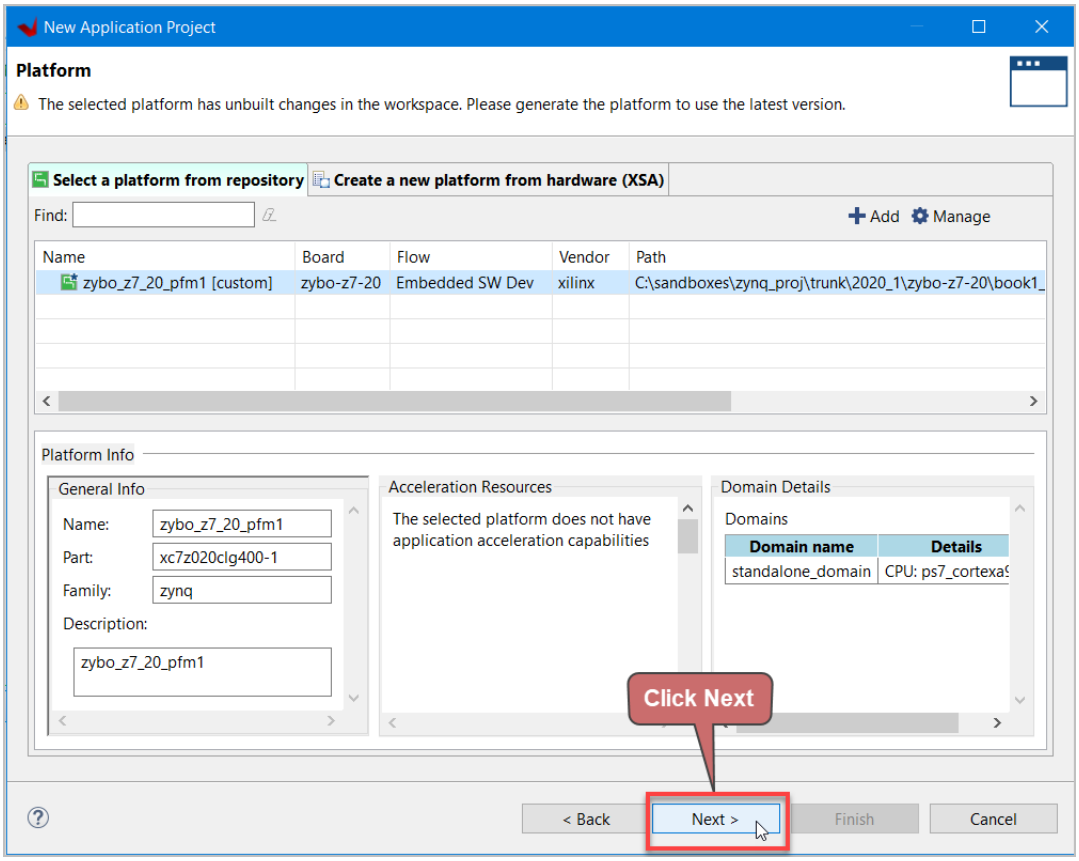


Figure 107. Leave the default settings, and click Next to continue.

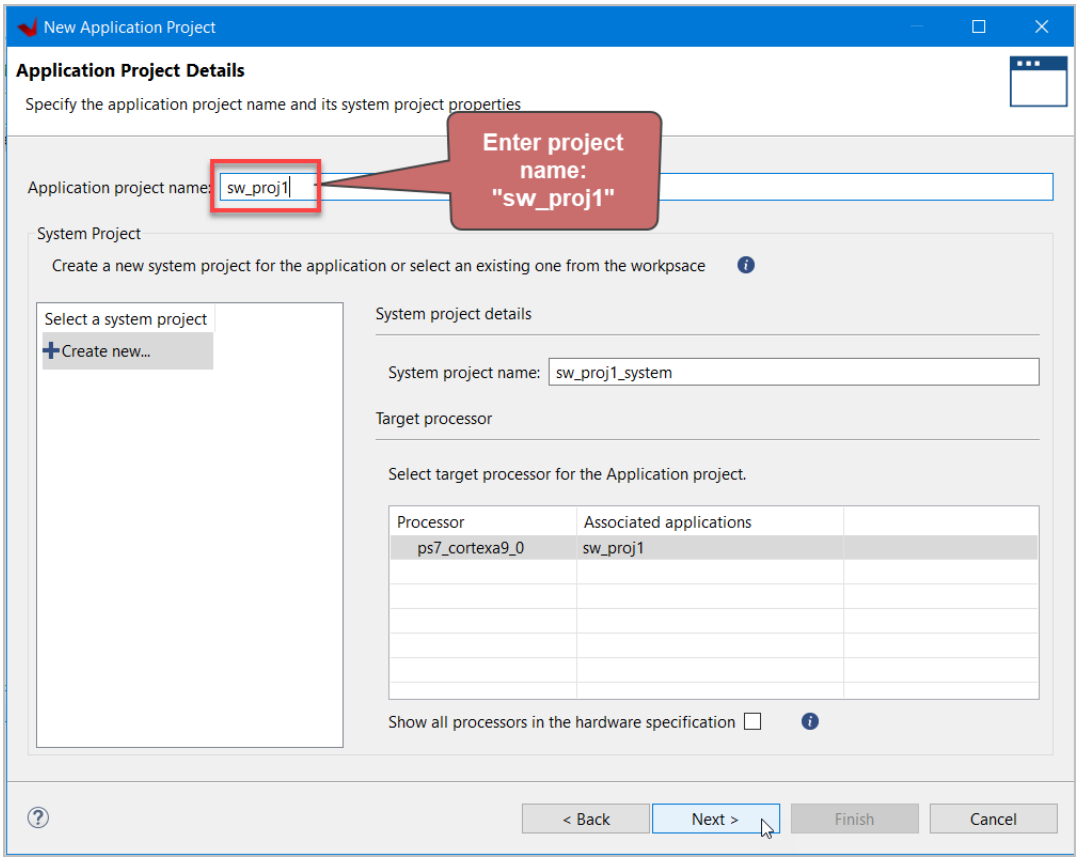


Figure 108. Enter the project name "sw_proj1" and click Next to continue.

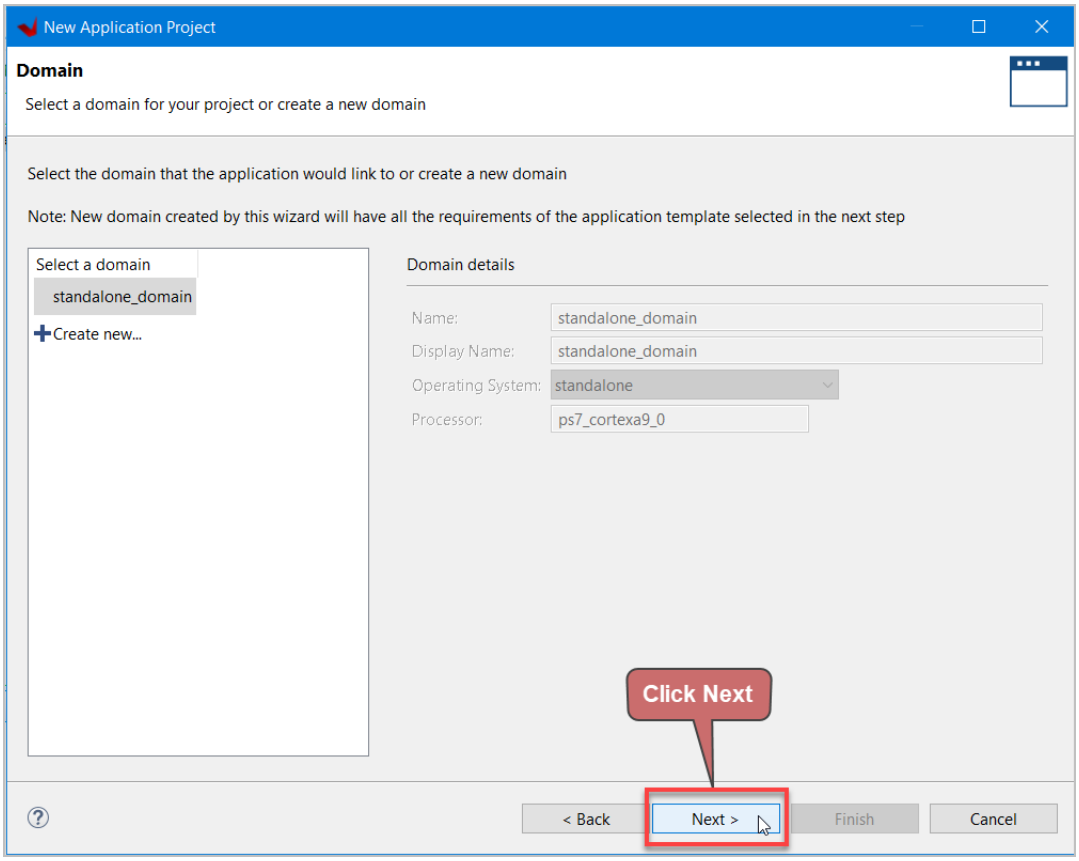


Figure 109. The standalone domain should be selected by default. Click Next to continue.

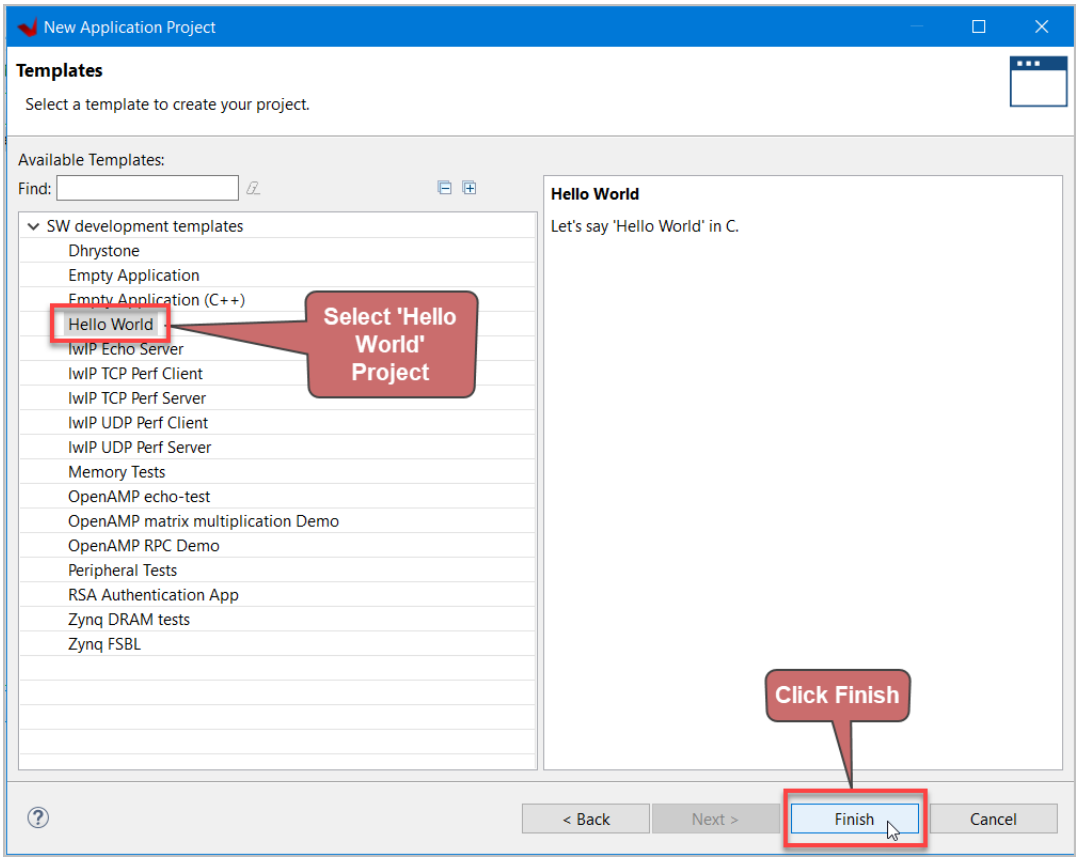


Figure 110. Select the Hello World project, and click Finish to continue.

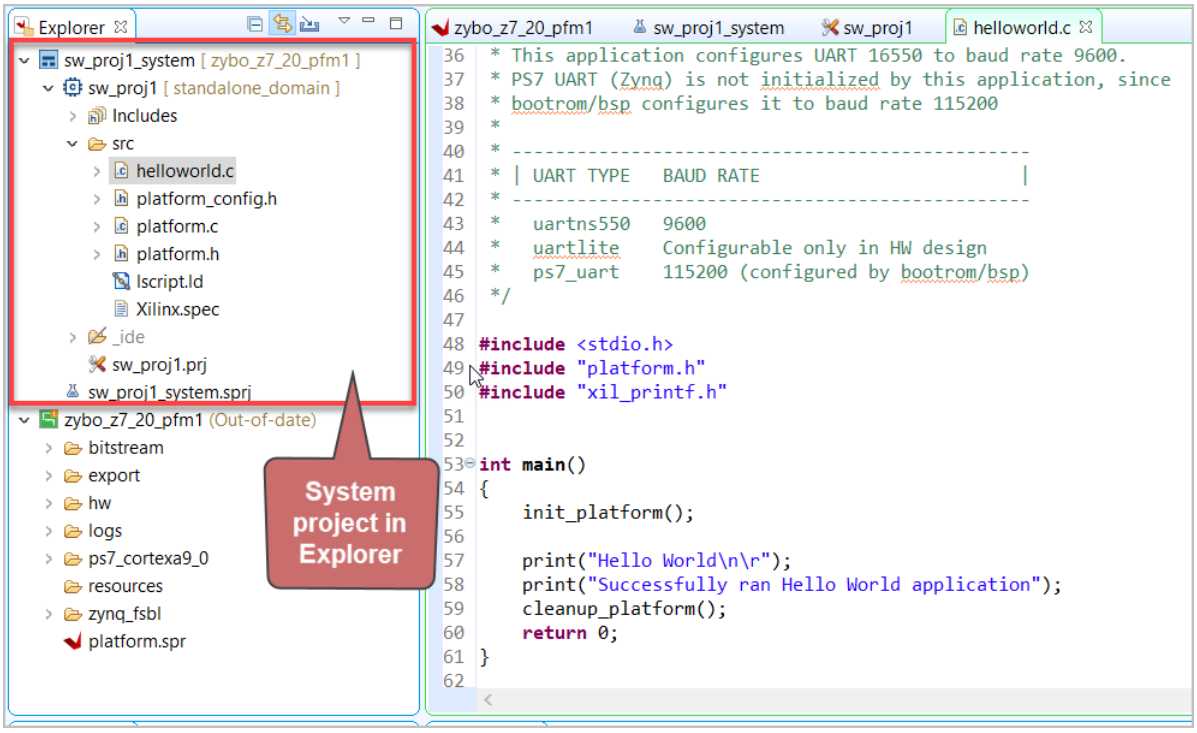


Figure 111. The Hello World application is added to the workspace as part of a System Project.

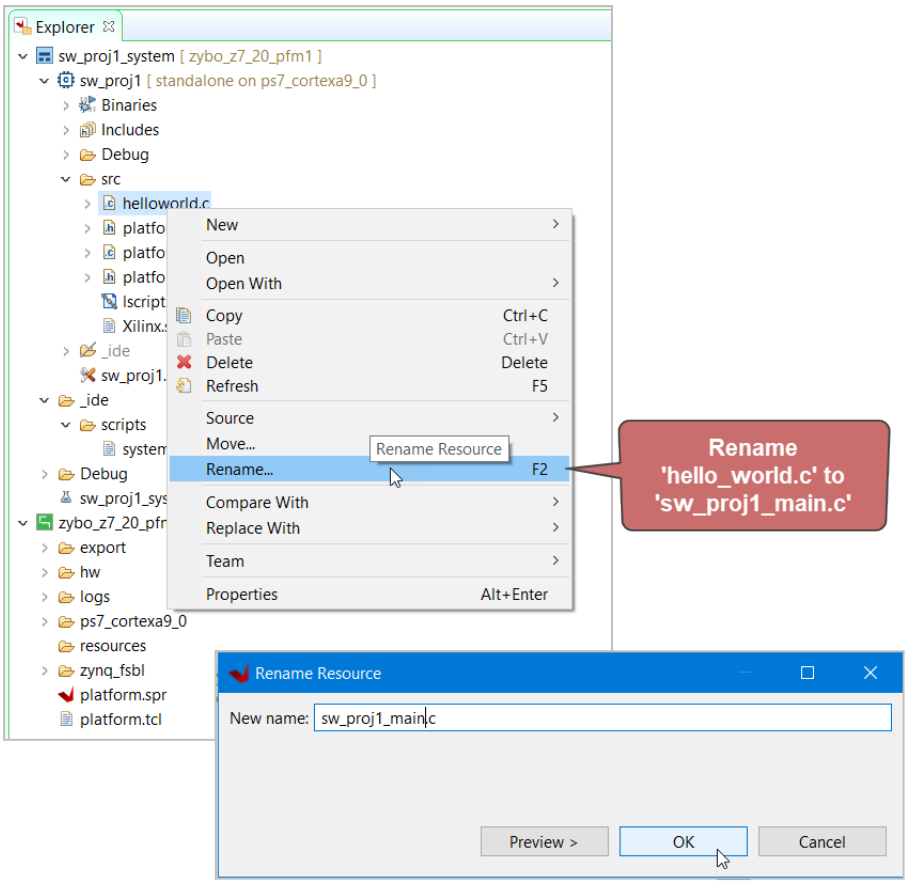


Figure 112. Rename the main source file to "sw_proj1_main.c".

```c
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"

int main()
{
  init_platform();

  print("Hello World\n\r");
  print("Successfully ran Hello World application\n\r");

  cleanup_platform();
  return 0;
}
```

3.4.1 Prepare the Platform (Zybo-Z7-2o or Zybo-Z7-10)

- Ensure that the boot mode is set to JTAG. (Header J5 must have a jumper across pins 1-2.)
- Connect a Micro-B USB cable to J12.
- Power up the platform (SW4).

3.4.2 Open Console Application

Here, Tera Term is used, although the console in the XSDK could also be used (or indeed any other suitable terminal program). Suitable settings are shown in the following figures. The terminal and serial port settings are important, but the window and font settings are optional.

Note that in this particular case, the platform is found on COM Port 6, but this will vary from system to system.

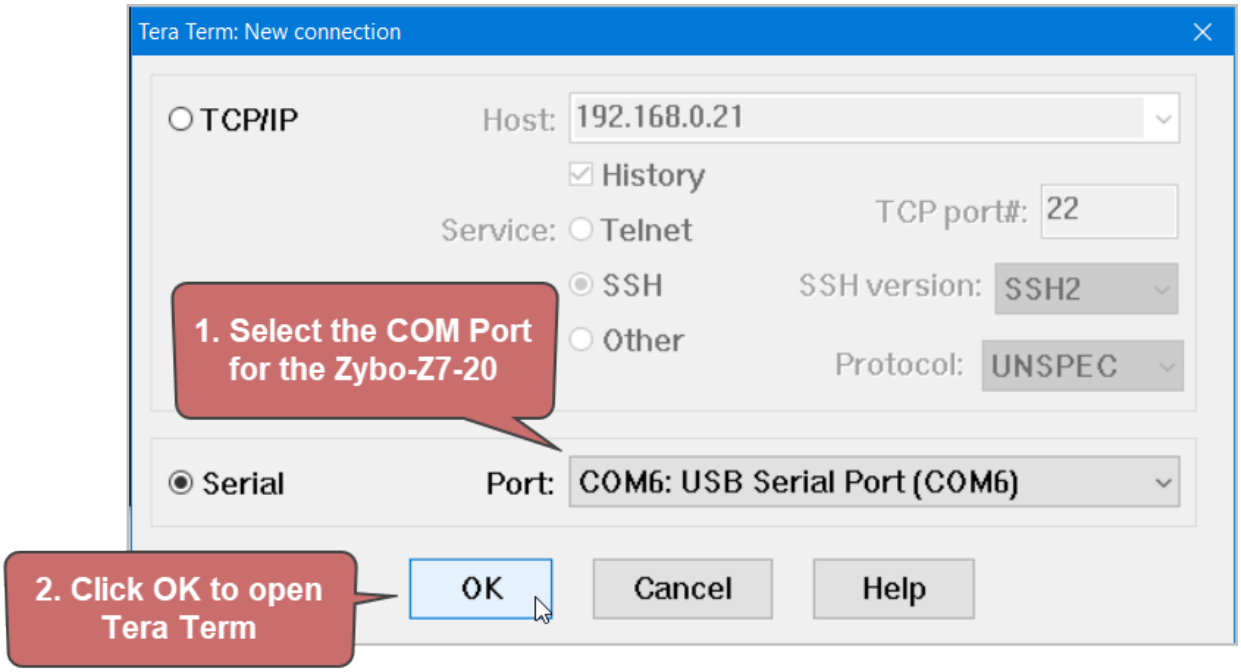


Figure 113. Open the connection to the platform.

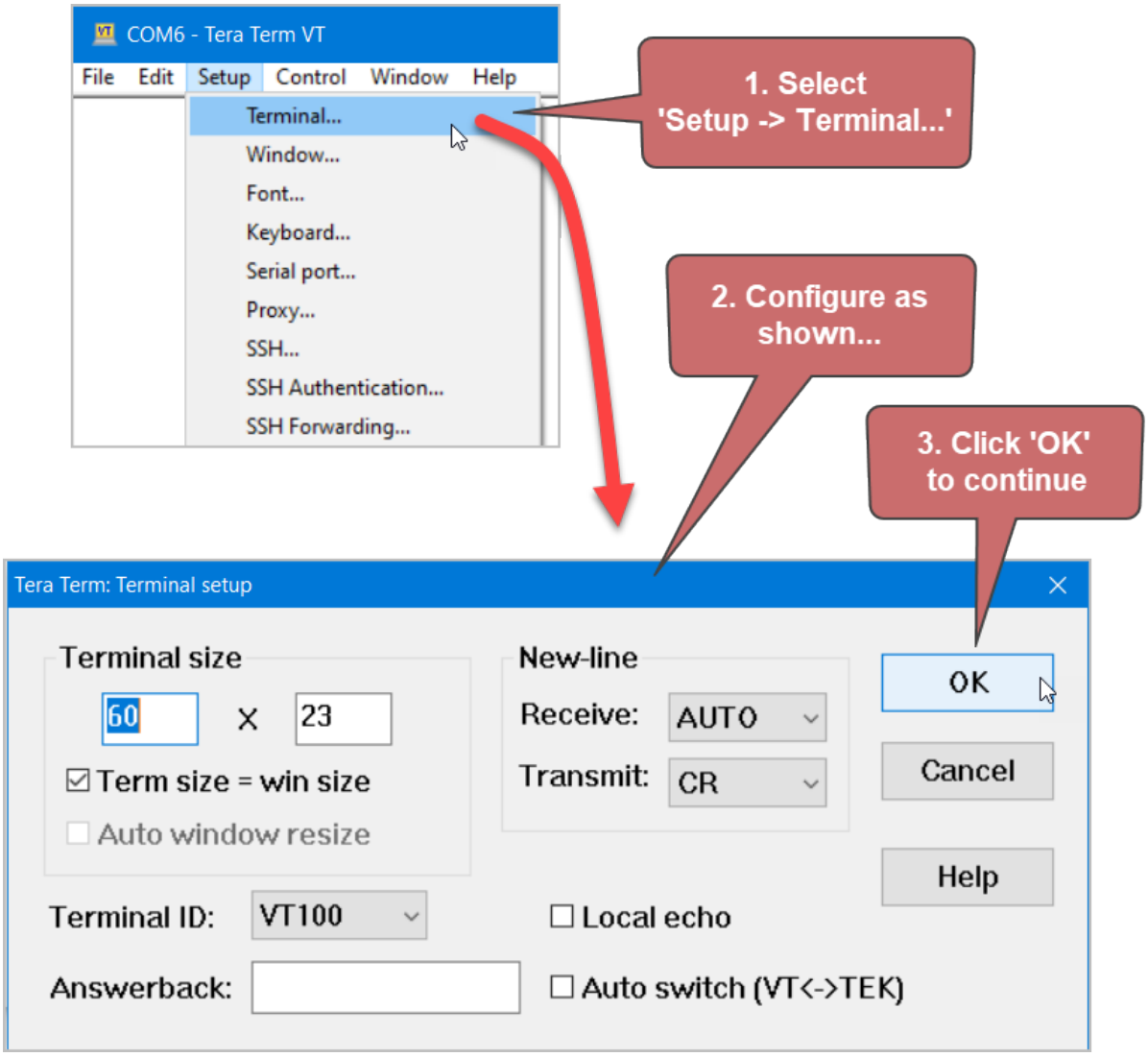


Figure 114. Configure the terminal. The New-line settings in particular are important.

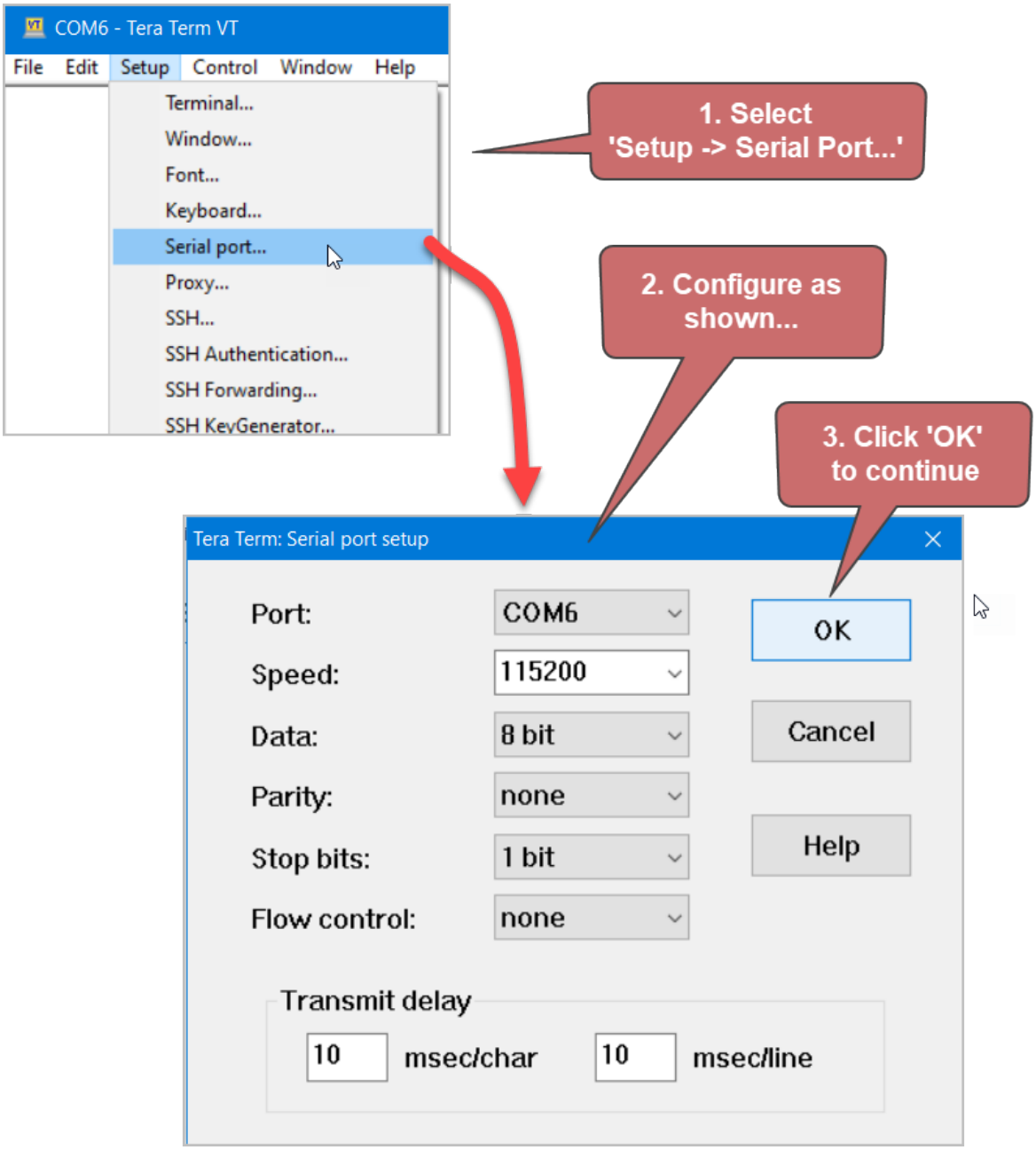


Figure 115. Configure the serial port settings. (Remember, the Port may be different.)

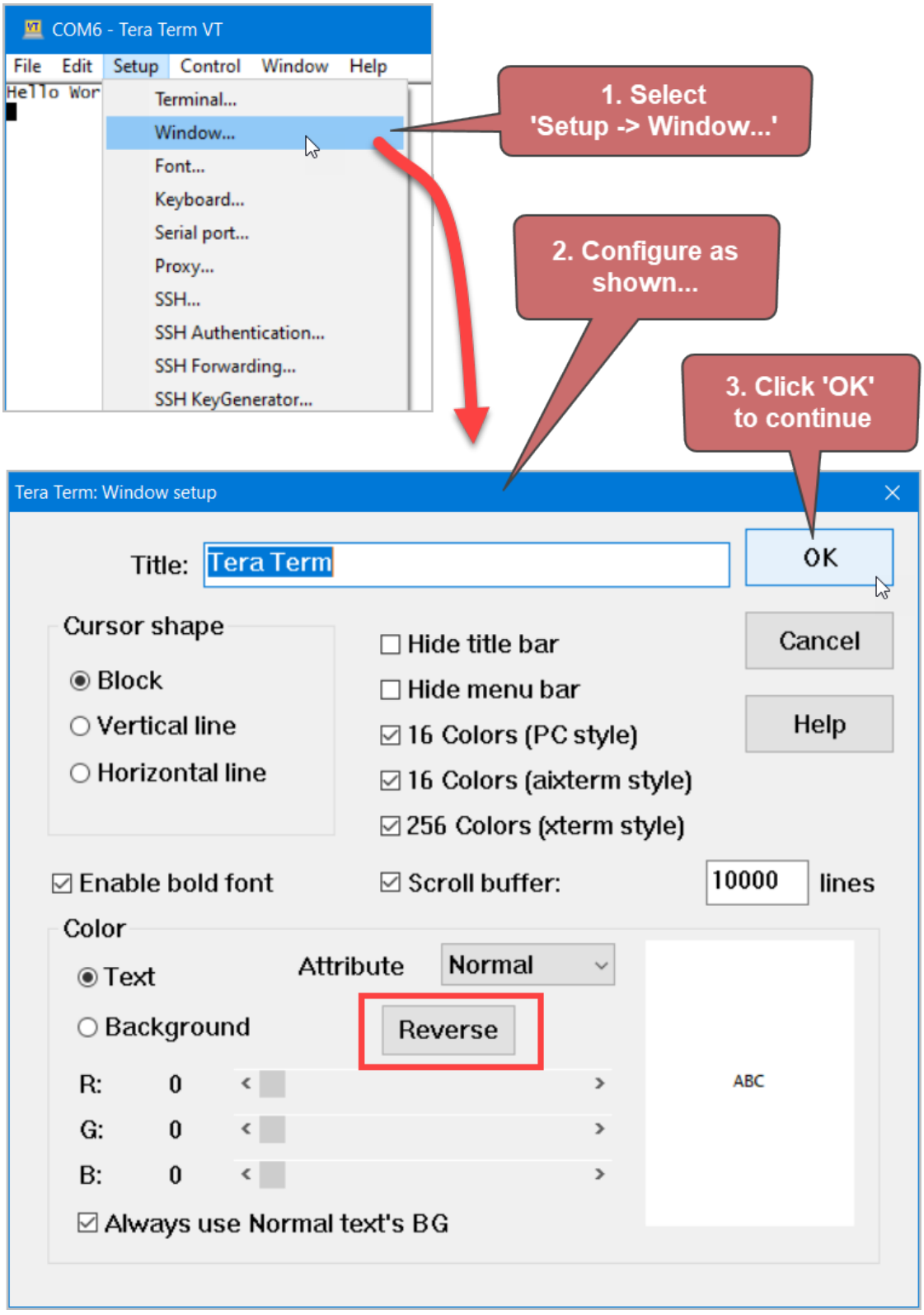


Figure 116. Optionally, reconfigure the window with a white background and black text.

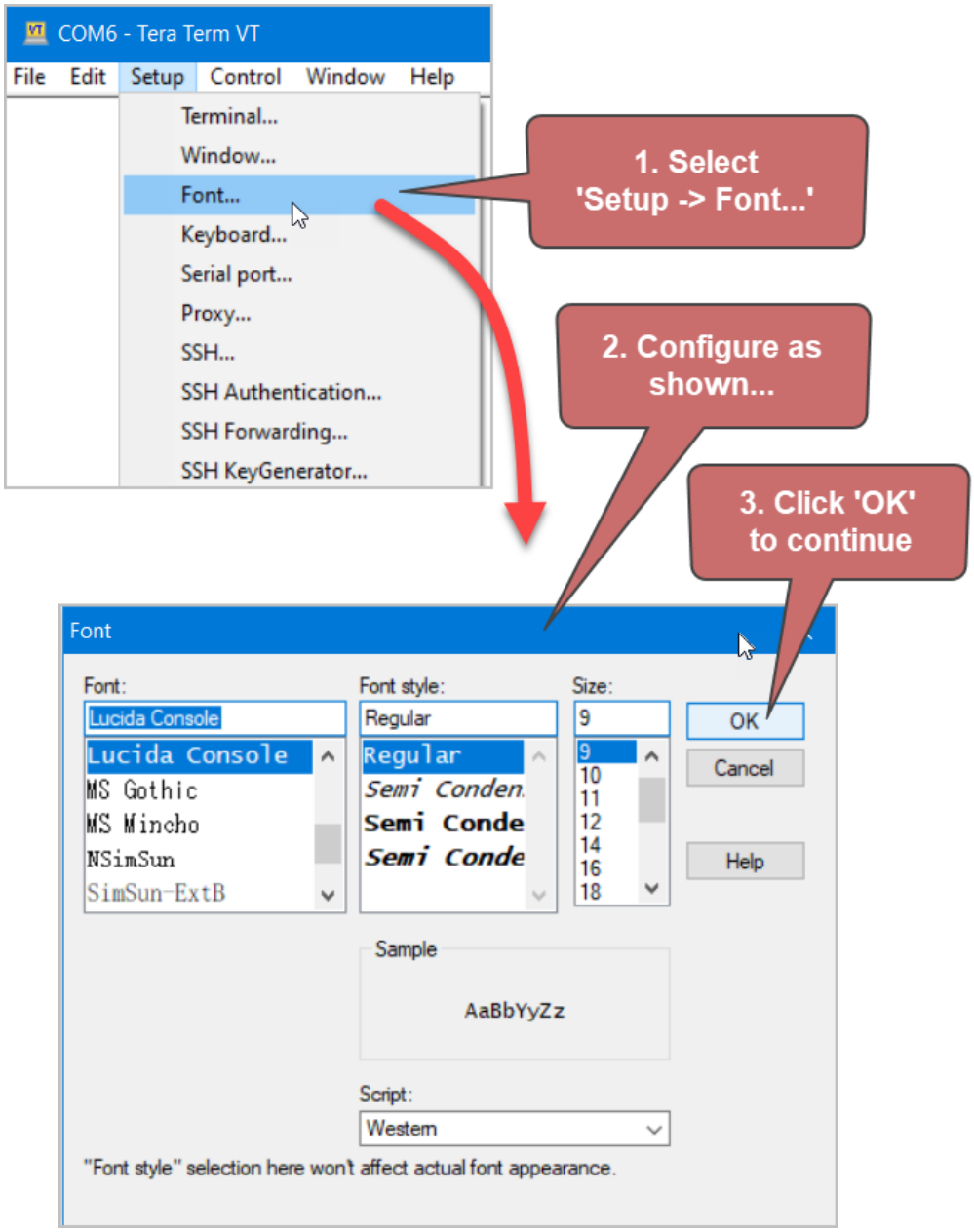


Figure 117. Optionally, change the font.

3.4.3 Build the Project

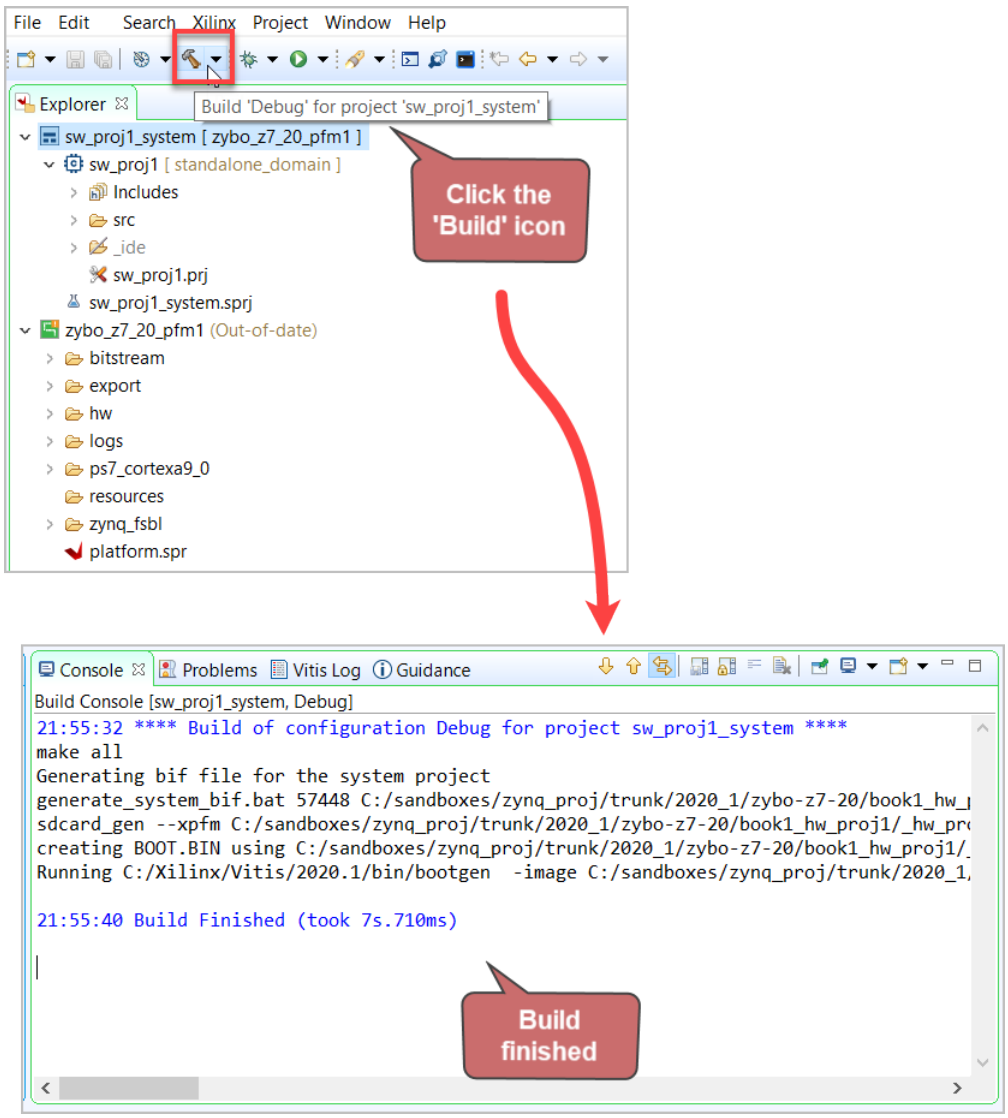


Figure 118. Select the system project and click the "Build" icon..

Build the project by selecting the system project and clicking on the Build icon. The application should be built successfully.

3.4.4 Run the Program

First of all, note that if using Vitis 2021.1, the error shown in Figure 3 might be seen when building the software projects in this section. This has been seen to occur if the user downloads a new design to the platform while a previous project is still running. The solution is to power-cycle the board before running each project.

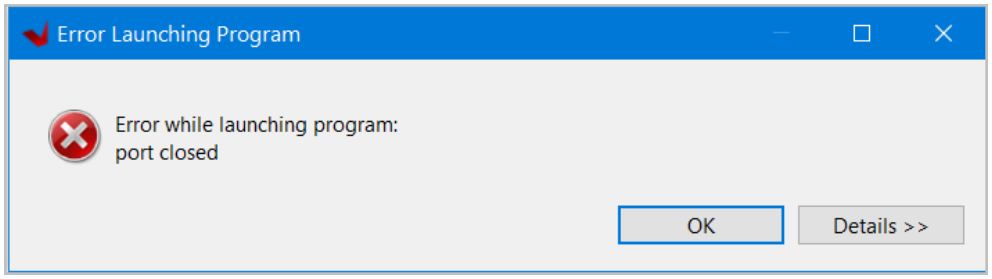


Figure 119. Vitis 2021.1 Error

Next, open the XSCT console. This is not a necessary step, but it does provide useful information on what happens when the program is launched.

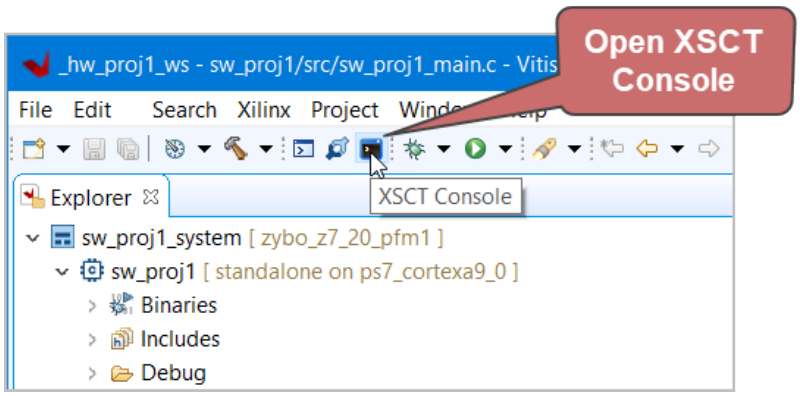


Figure 120. Open the XSCT console

To run the program, the first step is to create a Run configuration. Right-click on the project, and select 'Run As -> Run Configurations...' option.

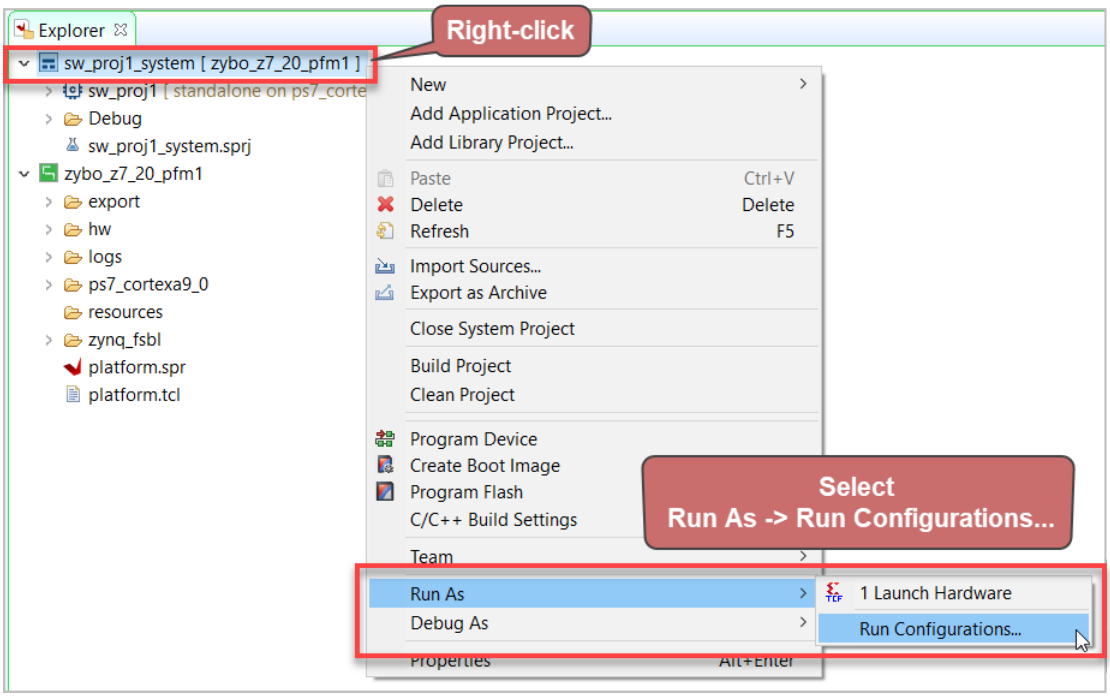


Figure 121. Right-click the software project to create a Run Configuration

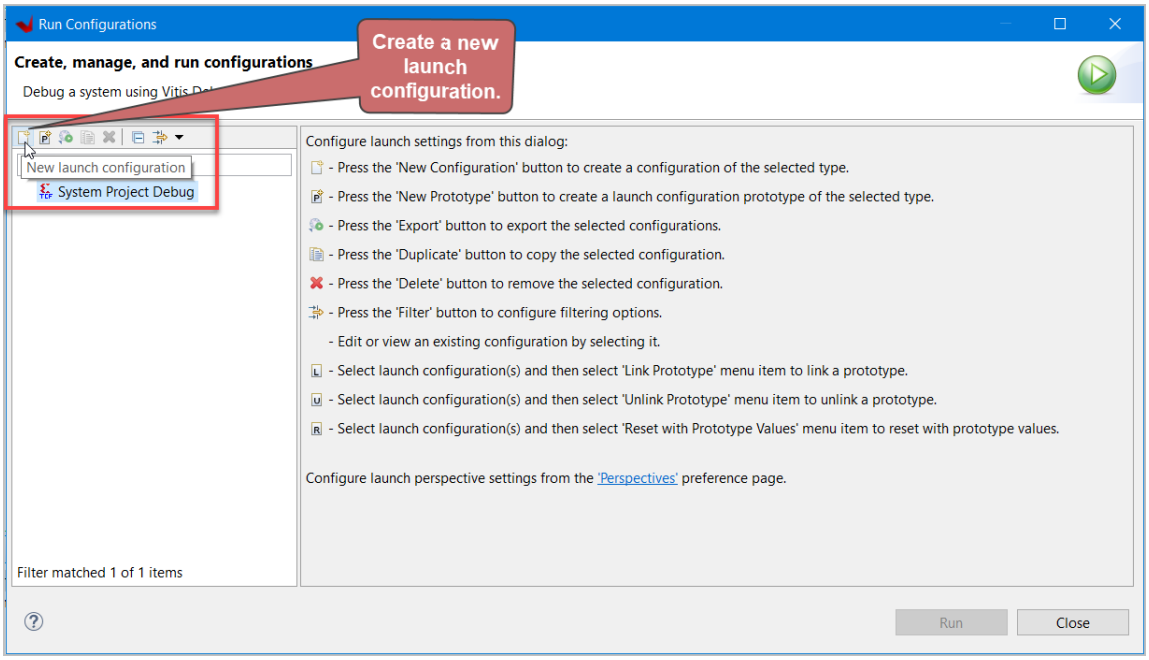


Figure 122. Create a TCF (i.e. System Debugger) configuration.

1. Ensure the System Project Debug option is selected.
2. Click on New Launch Configuration.

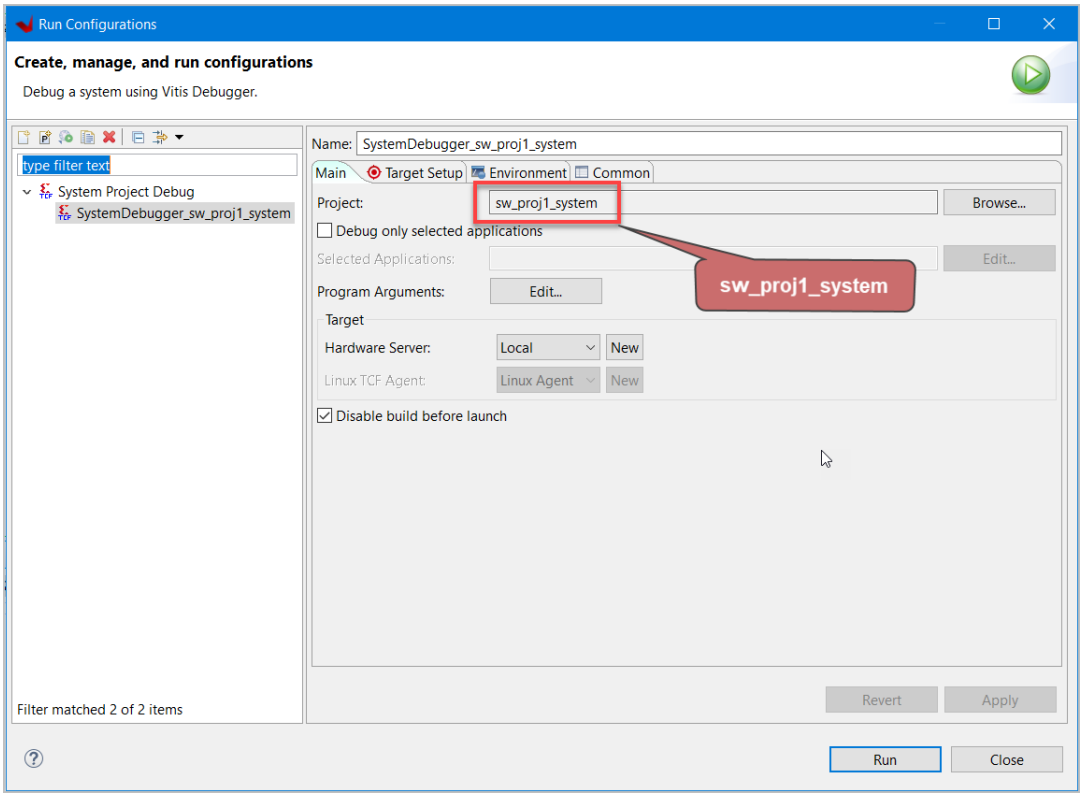


Figure 123. In the Main tab, check that the project is sw_proj1_system.

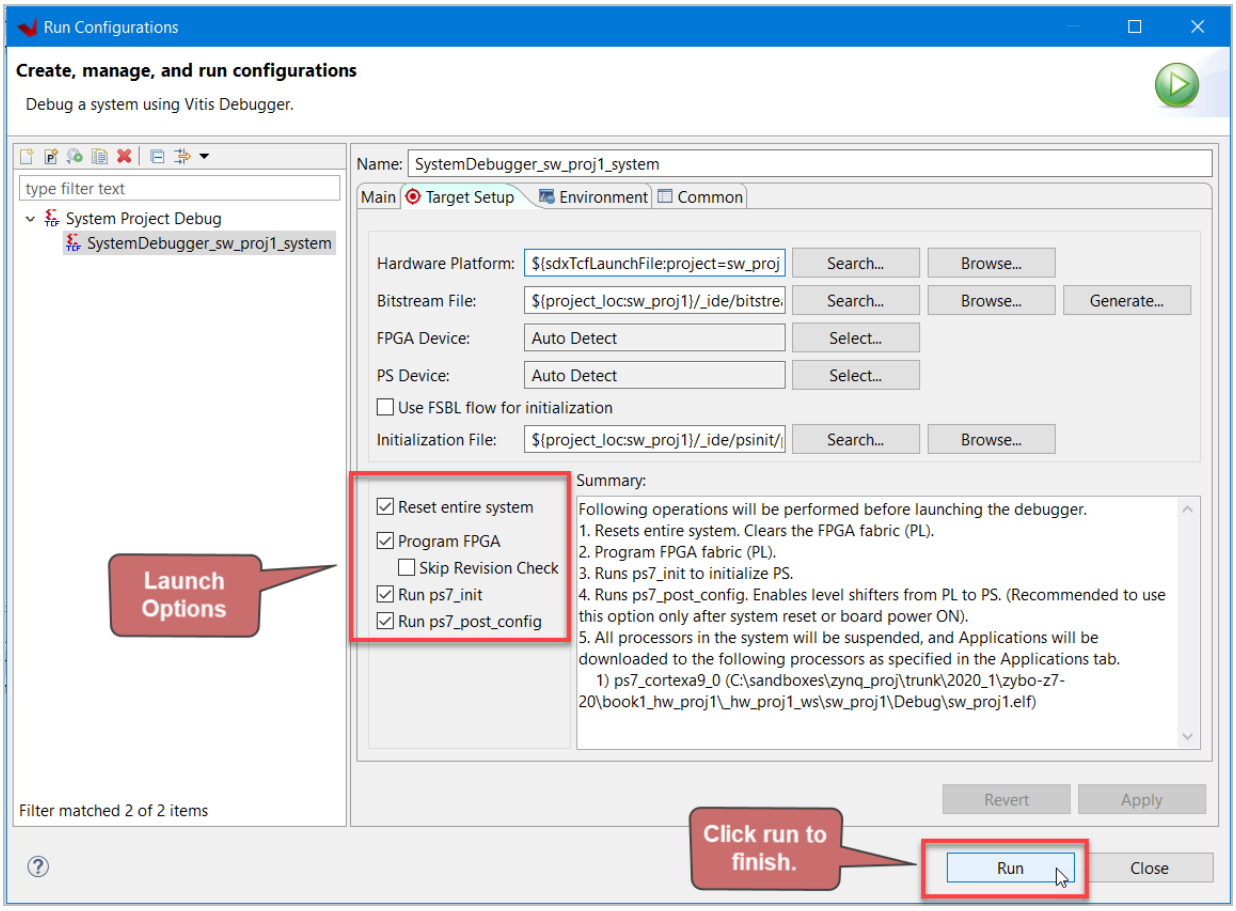


Figure 124. In Target Setup, select the Launch options, and Click on run to execute the program.

1. Select the main options as follows:
   a. Reset entire system
   b. Program FPGA
   c. Run ps7_init
   d. Run ps7_post_config

2. Click on 'Run' to continue. The FPGA should be programmed, and the ELF file should be downloaded to the platform, as shown in the next page.

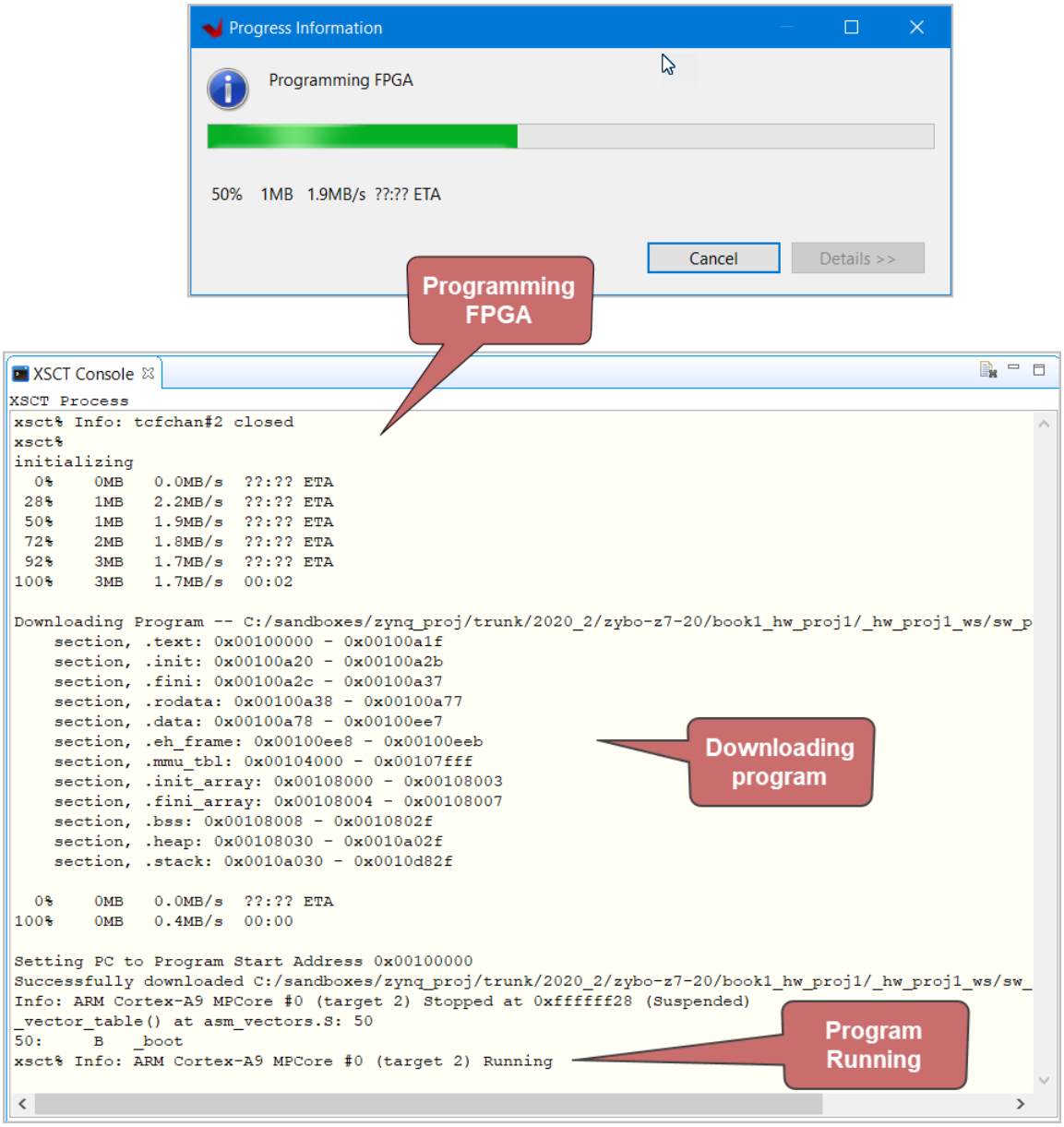


Figure 125. The FPGA is programmed, and the application is downloaded.

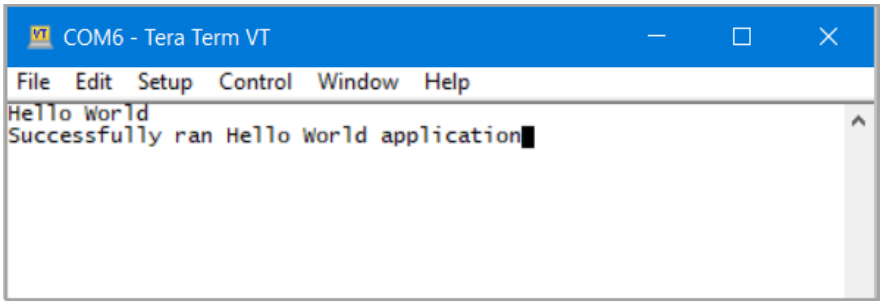


Figure 126. The program output is displayed in the console.

3.5 System (Software) Project 2: Zynq GPIO

3.5.1 Create the Project

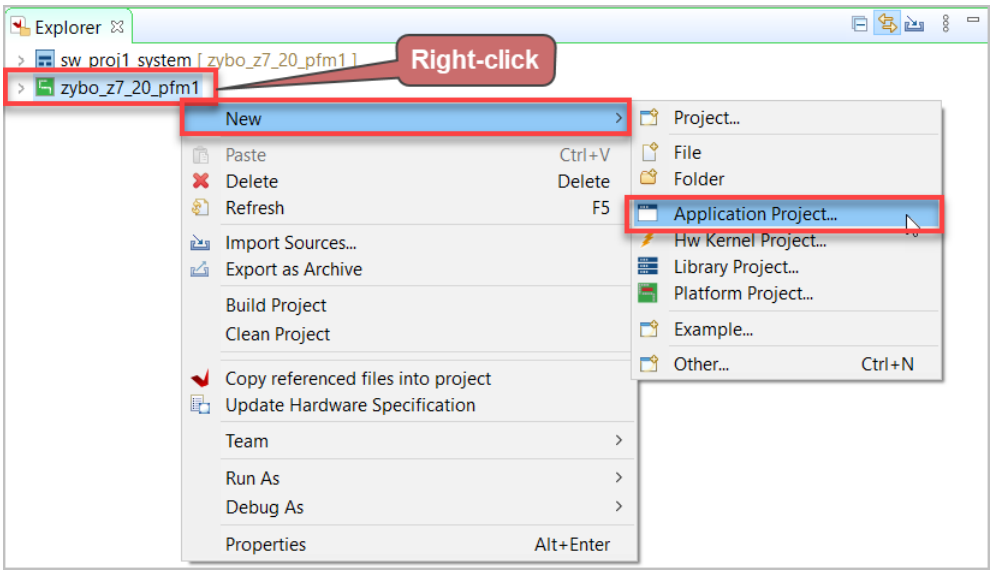


Figure 127. Right-click on the platform and select New-> Application Project.

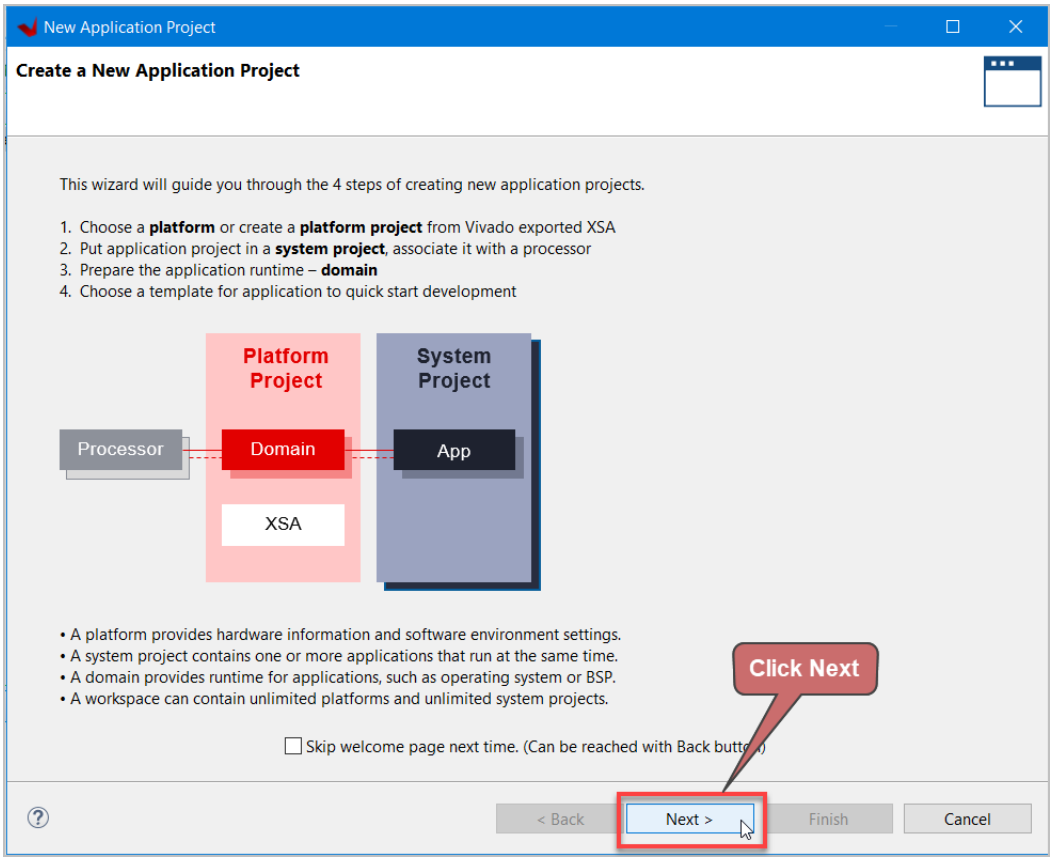


Figure 128. Click Next to continue.

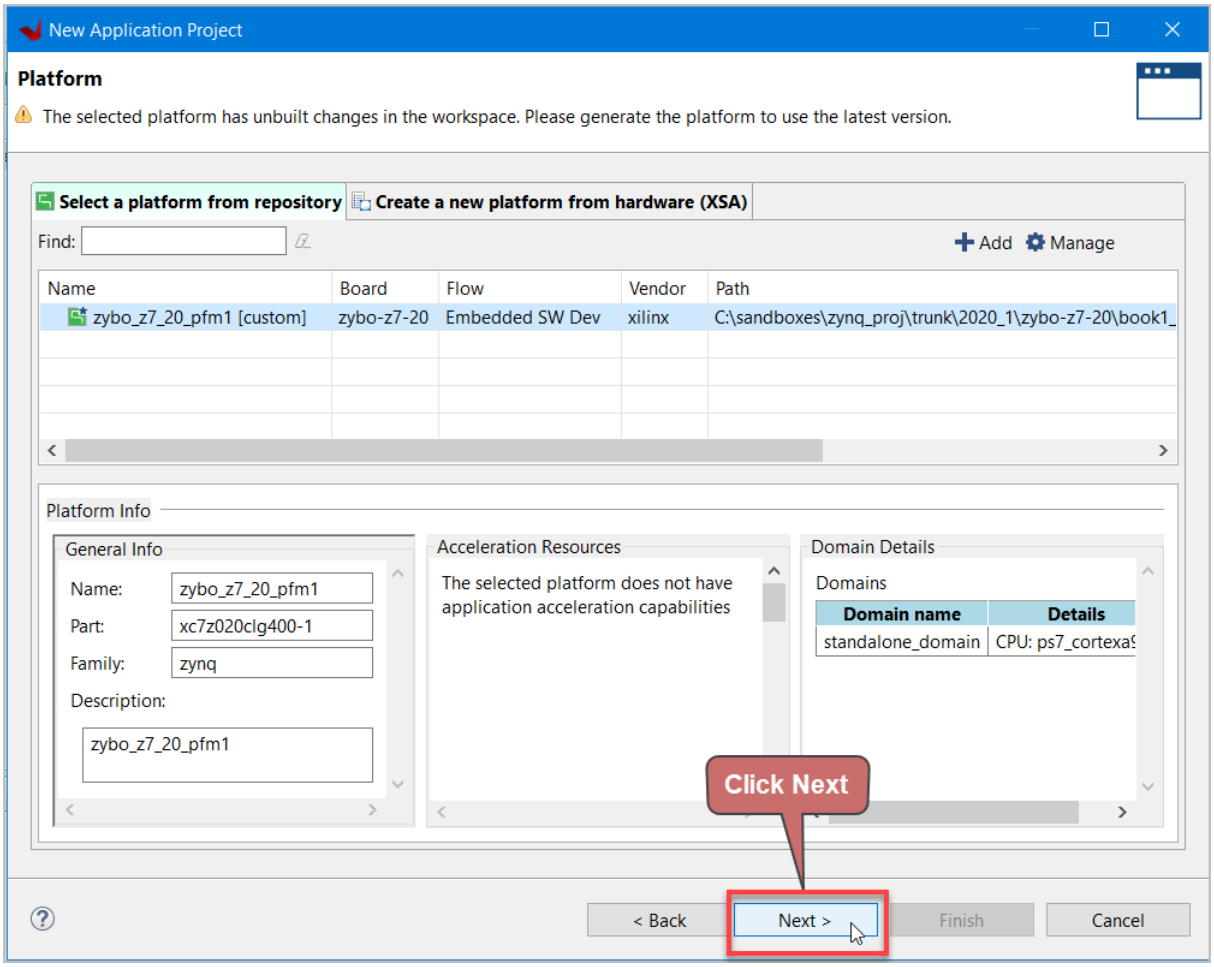


Figure 129. Leave defaults, and click Next to continue.

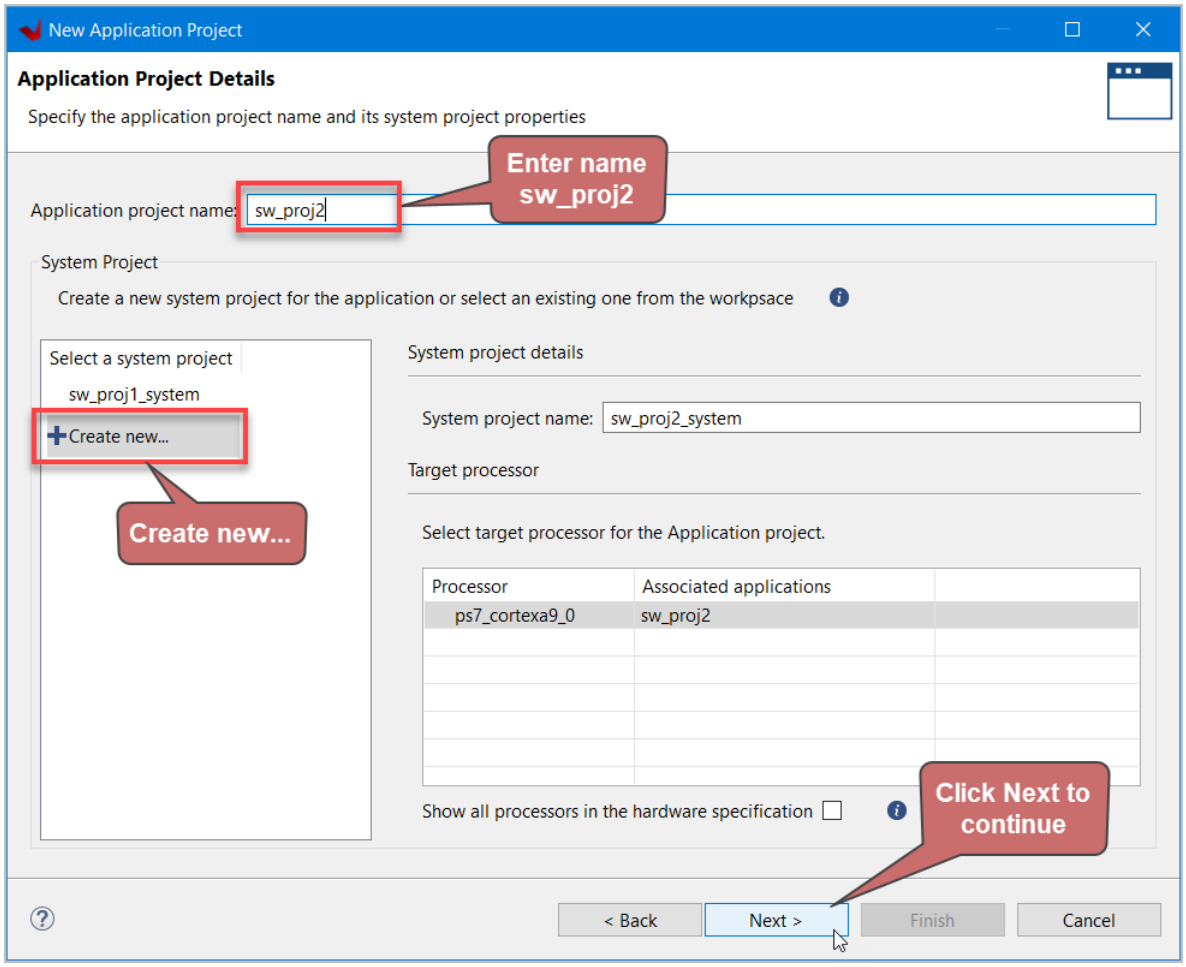


Figure 130. Set the project name to 'sw_proj2', ensure Create New... is selected, and press Next to continue.

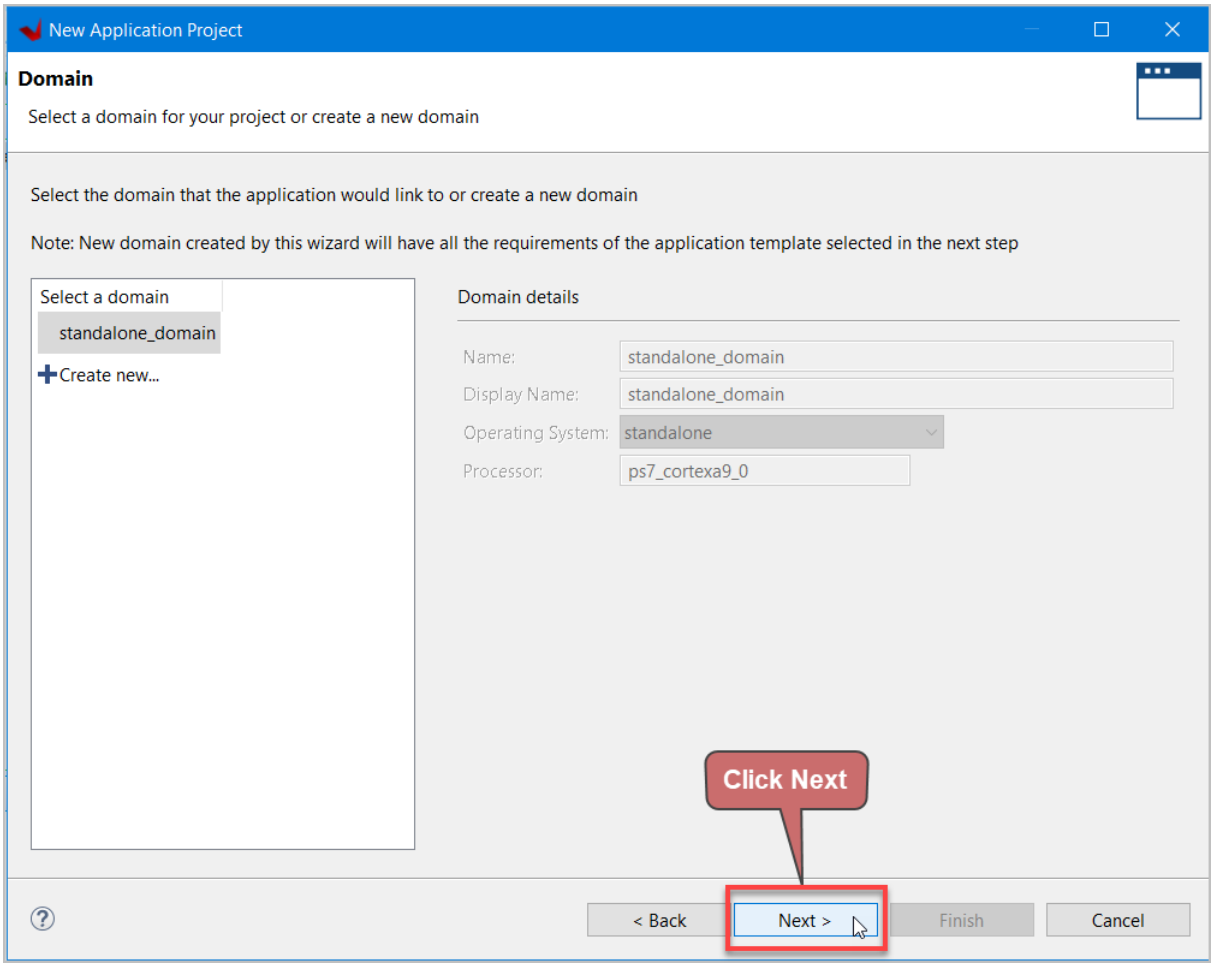


Figure 131. The standalone domain should be selected by default. Click Next to continue.

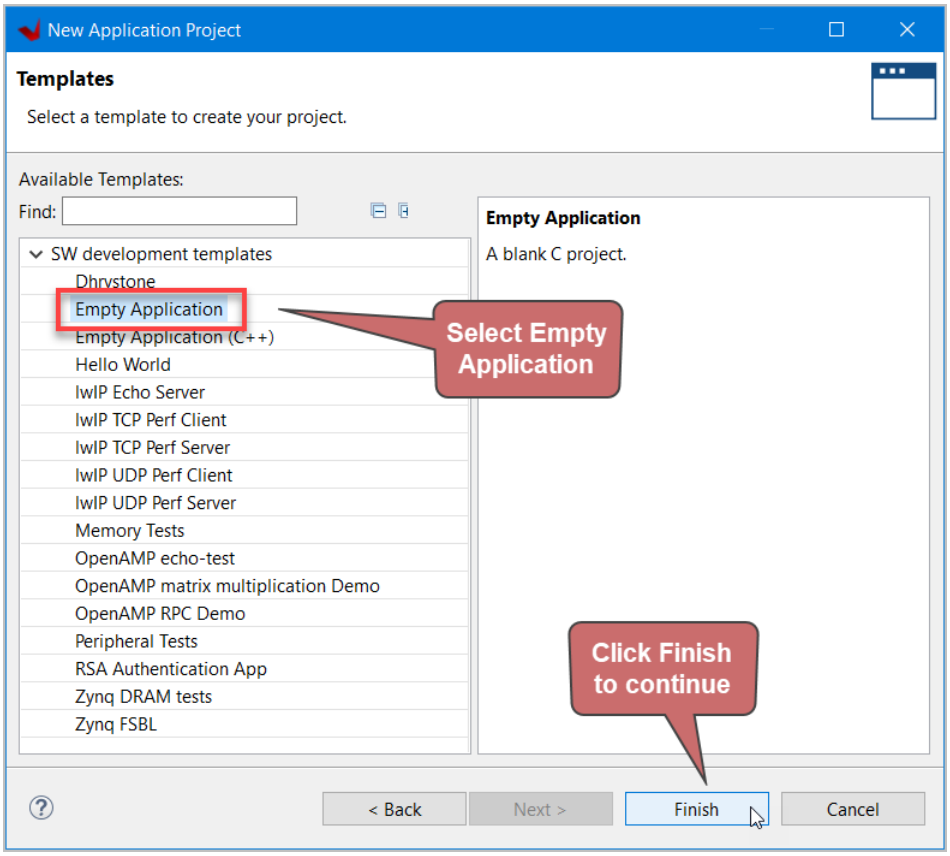


Figure 132. Select Empty Application, and click Finish.

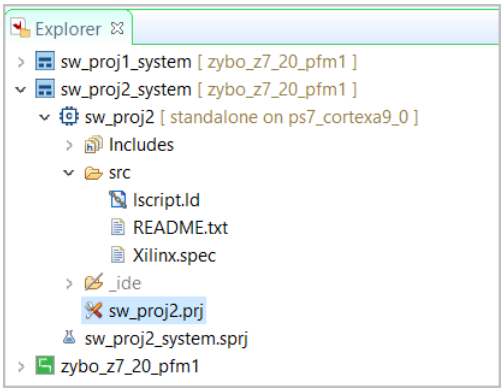


Figure 133. The empty application is created.

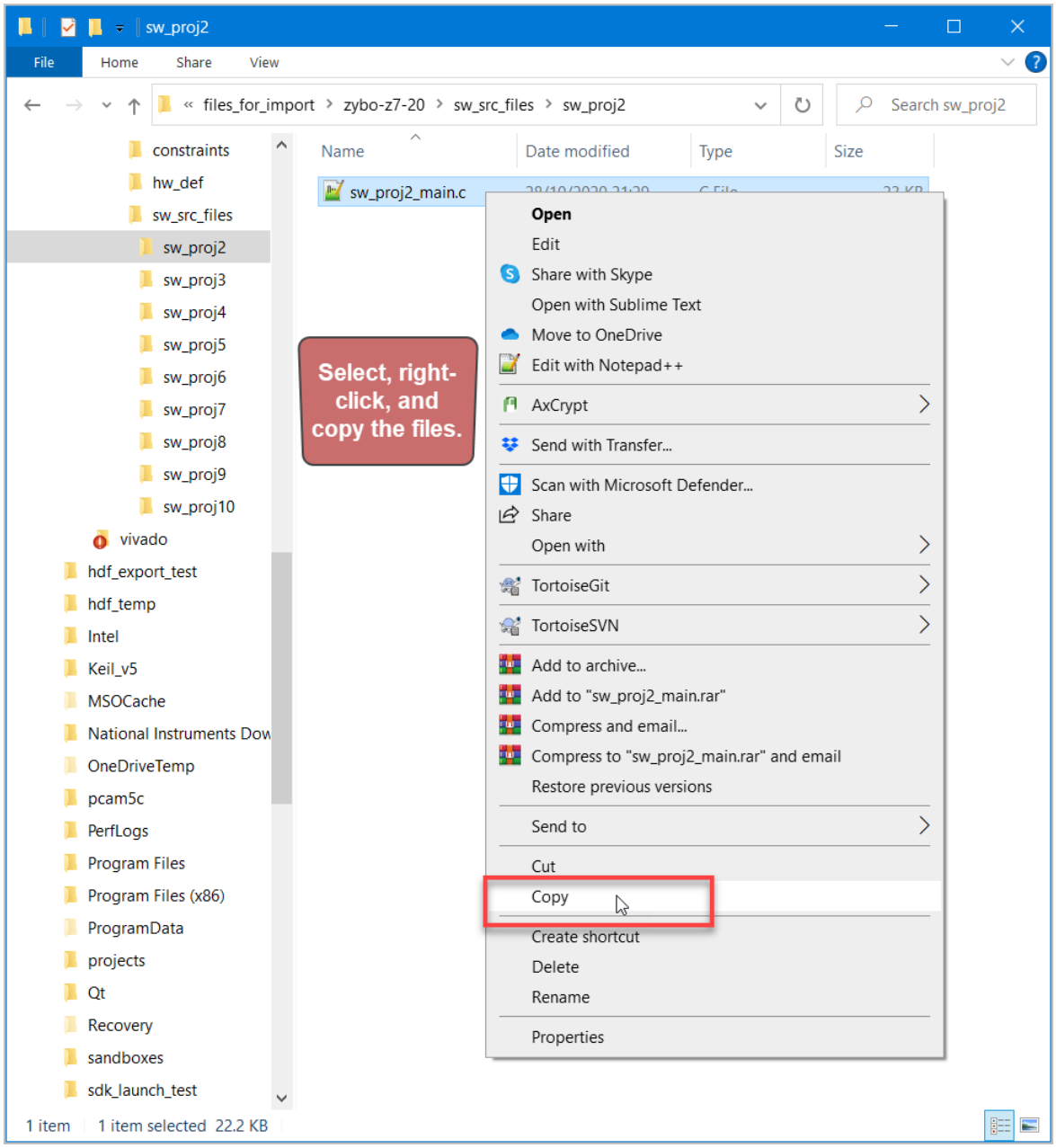


Figure 134. Select the project files, right-click, and select Copy.

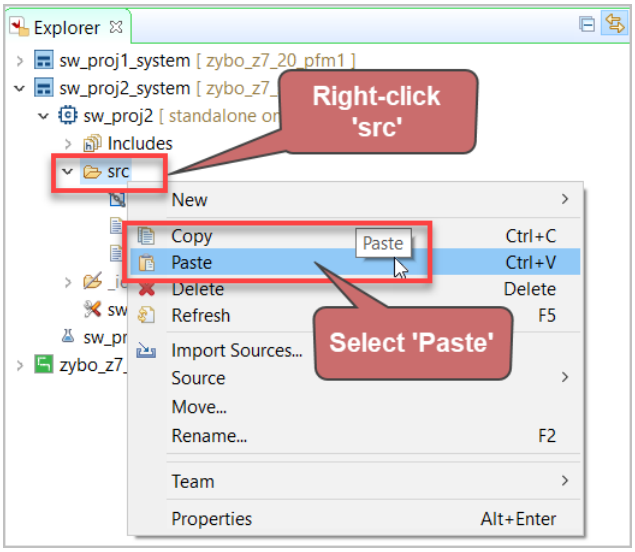


Figure 135. In Vitis, right-click the 'src' directory, and paste the files.

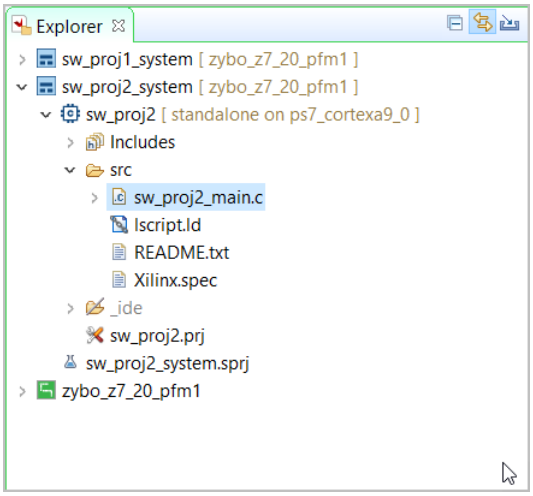


Figure 136. The project file is added.

3.5.2 Build the Project

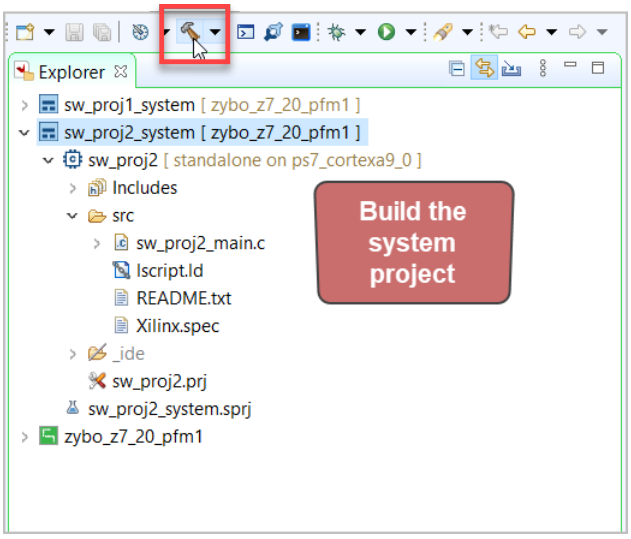


Figure 137. Build the system project

3.5.3 Run the Project

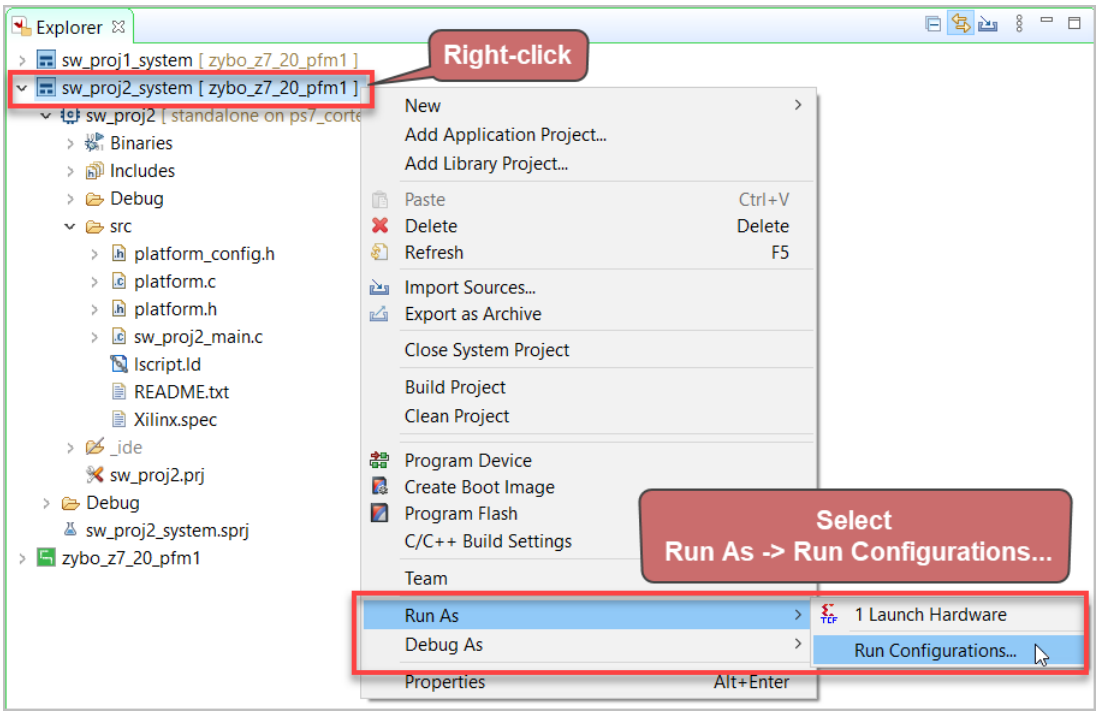


Figure 138. Right-click on the system project and select Run As -> Run Configurations.

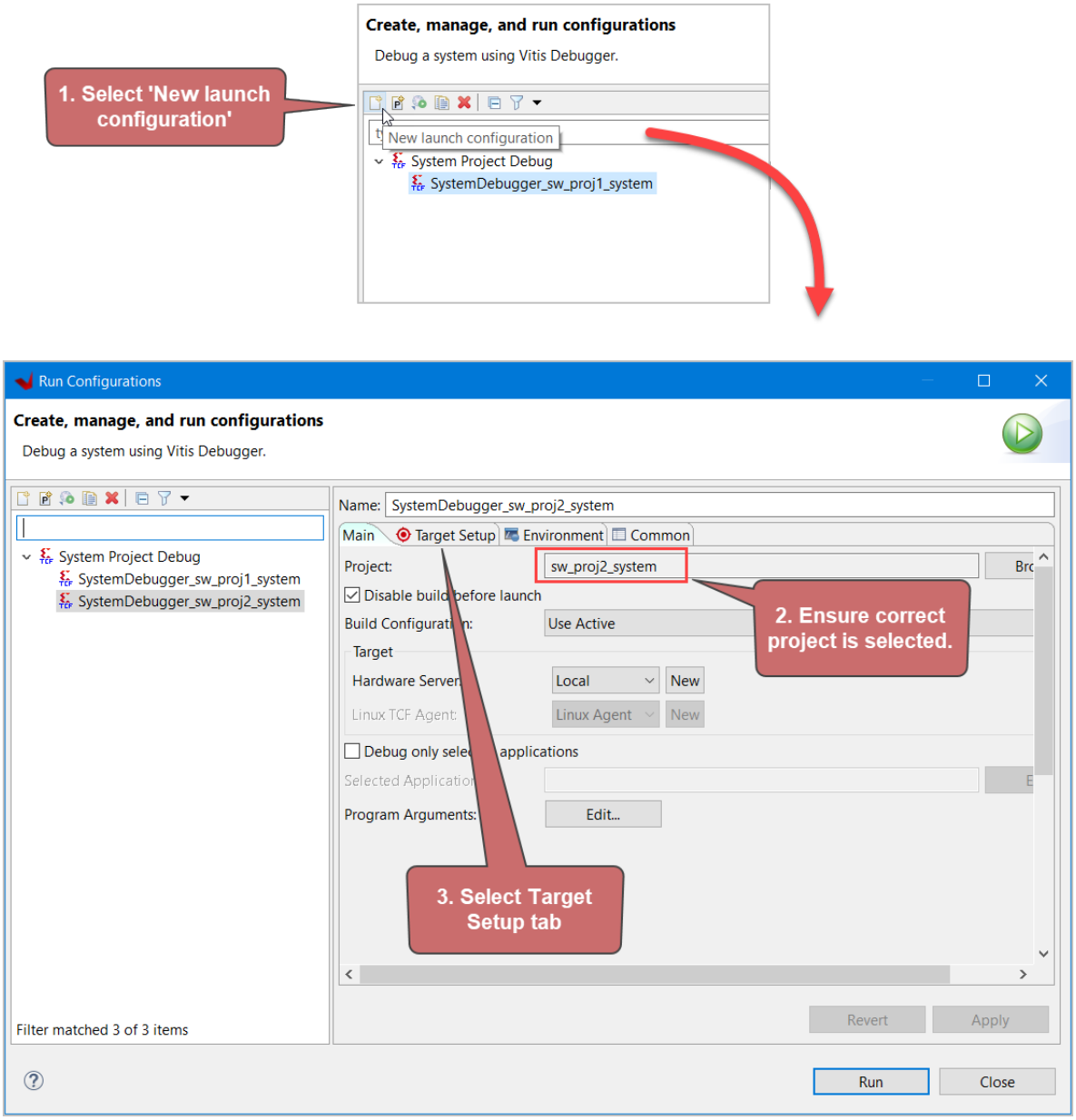


Figure 139. Create a New launch configuration

1. Click on 'New launch configuration'.
2. Ensure that "sw_proj2_system" is selected.
3. Select the Target Setup Tab.

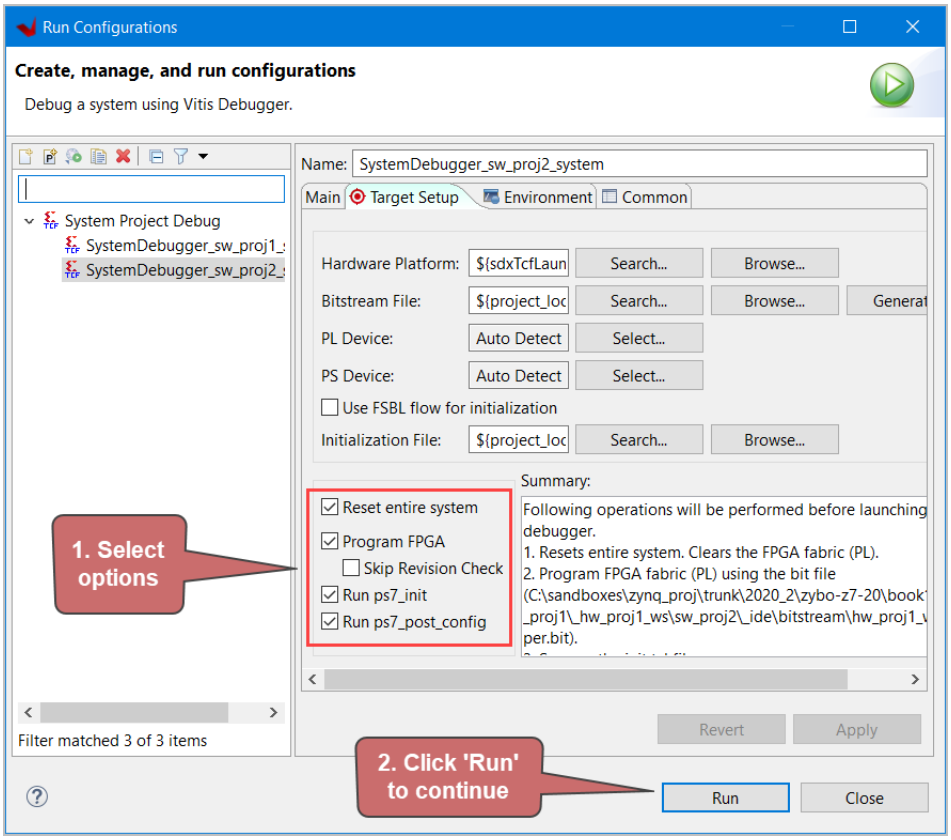


Figure 140. Run the application.

1. Select the main options as follows:
    a. Reset entire system
    b. Program FPGA
    c. Run ps7_init
    d. Run ps7_post_config

Click on 'Run' to continue. The FPGA should be programmed, and the ELF file should be downloaded to the platform, as shown in the next page.

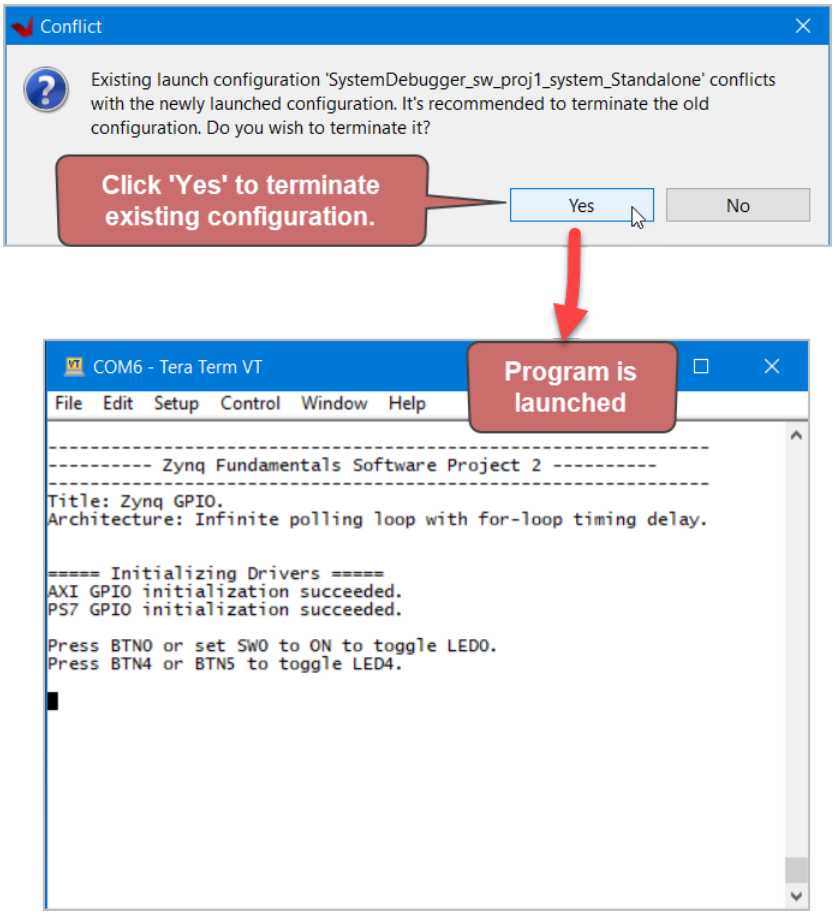


Figure 141. The project details for the software project should appear in the terminal.

If prompted, select 'Yes' to terminate any existing configuration. The FPGA will be programmed and the application will be downloaded to the processor. The terminal program should display the results for launching Software Project 2.

3.6 System (Software) Project 3: Structured Program

3.6.1 Create the Project

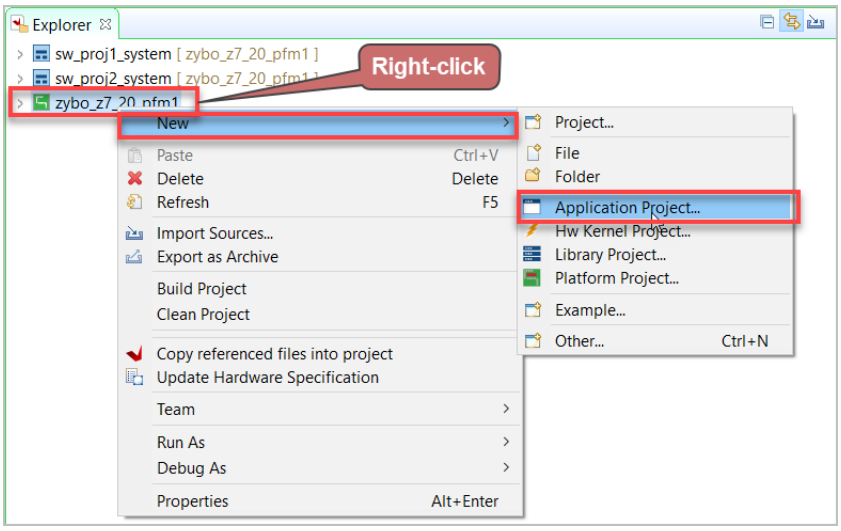


Figure 142. Right-click on the platform and select New-> Application Project.

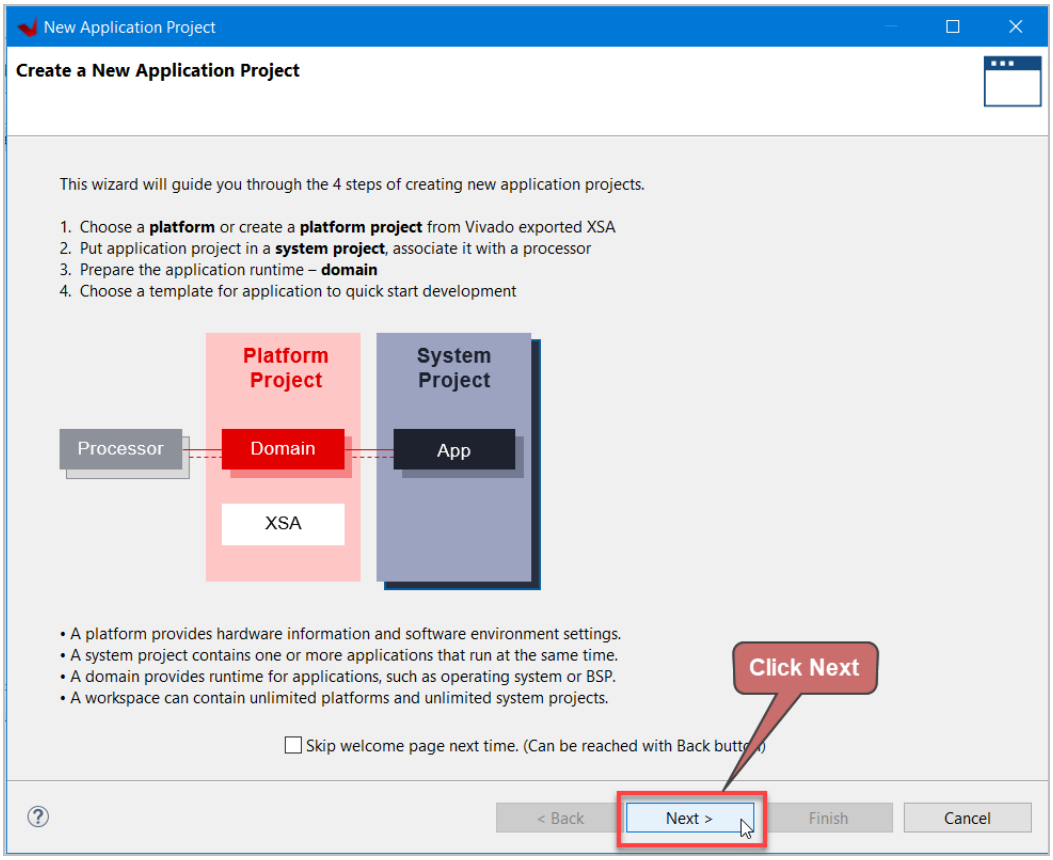


Figure 143. Click Next to continue.

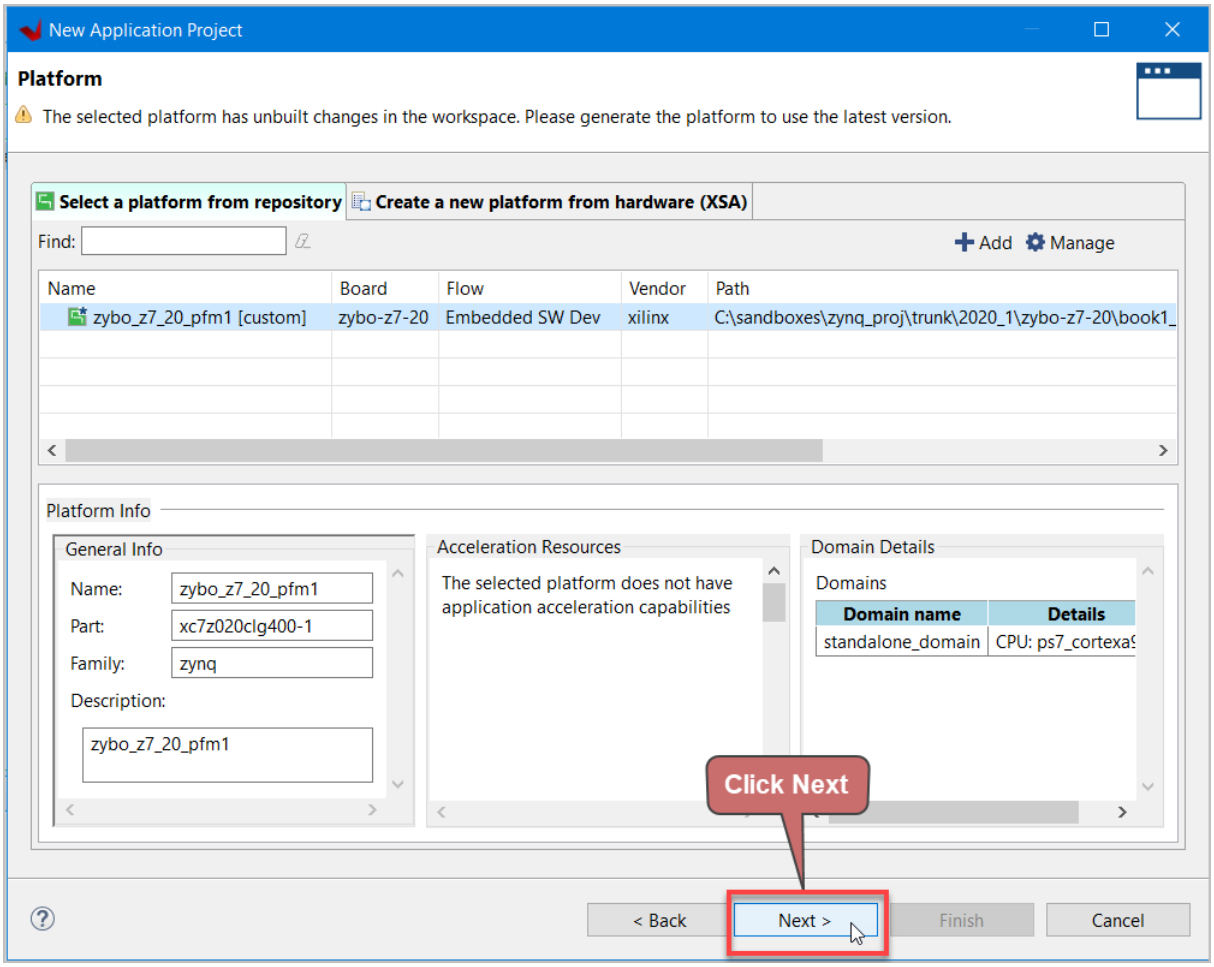


Figure 144. Leave defaults, and click Next to continue.

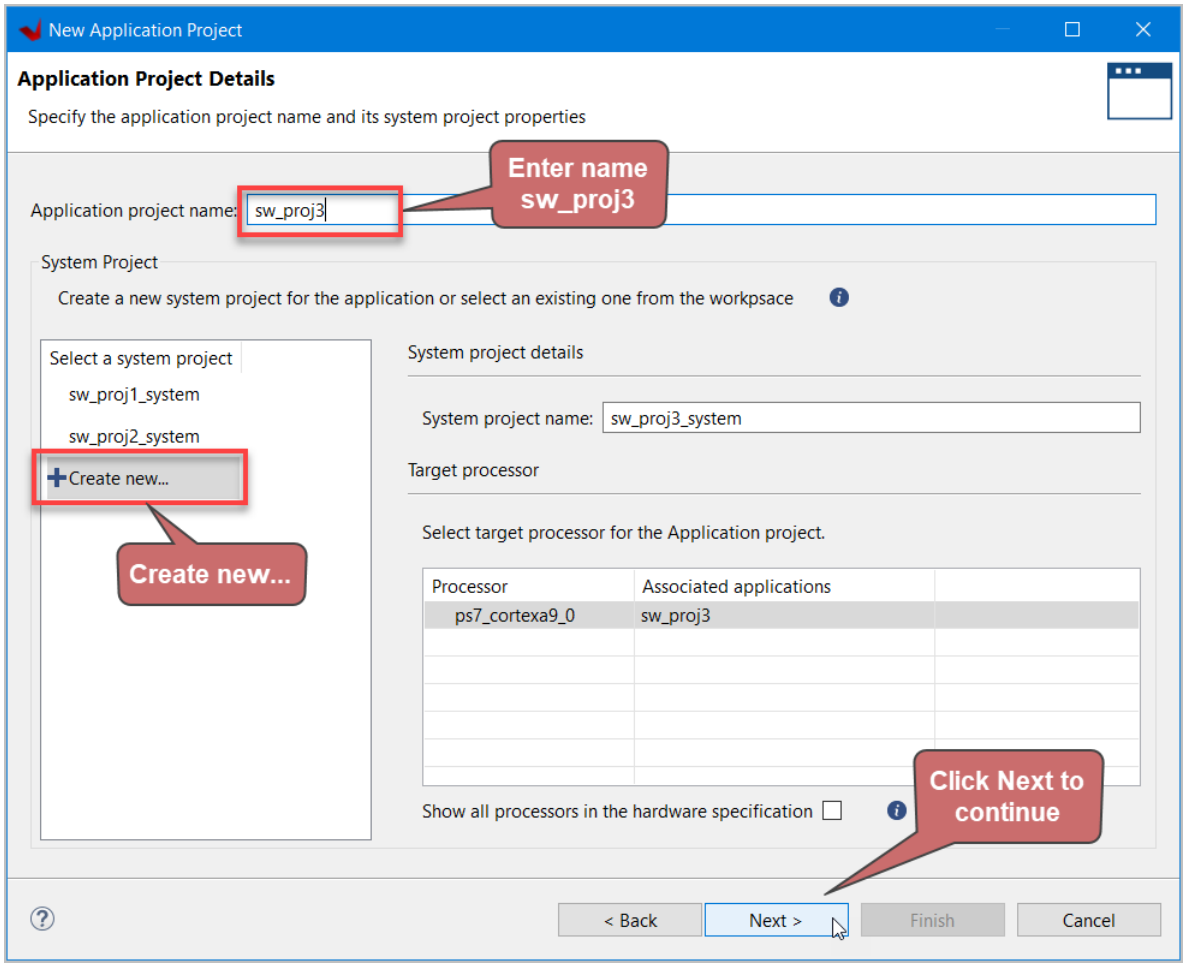


Figure 145. Set the project name to 'sw_proj3', ensure Create New... is selected, and press Next to continue.

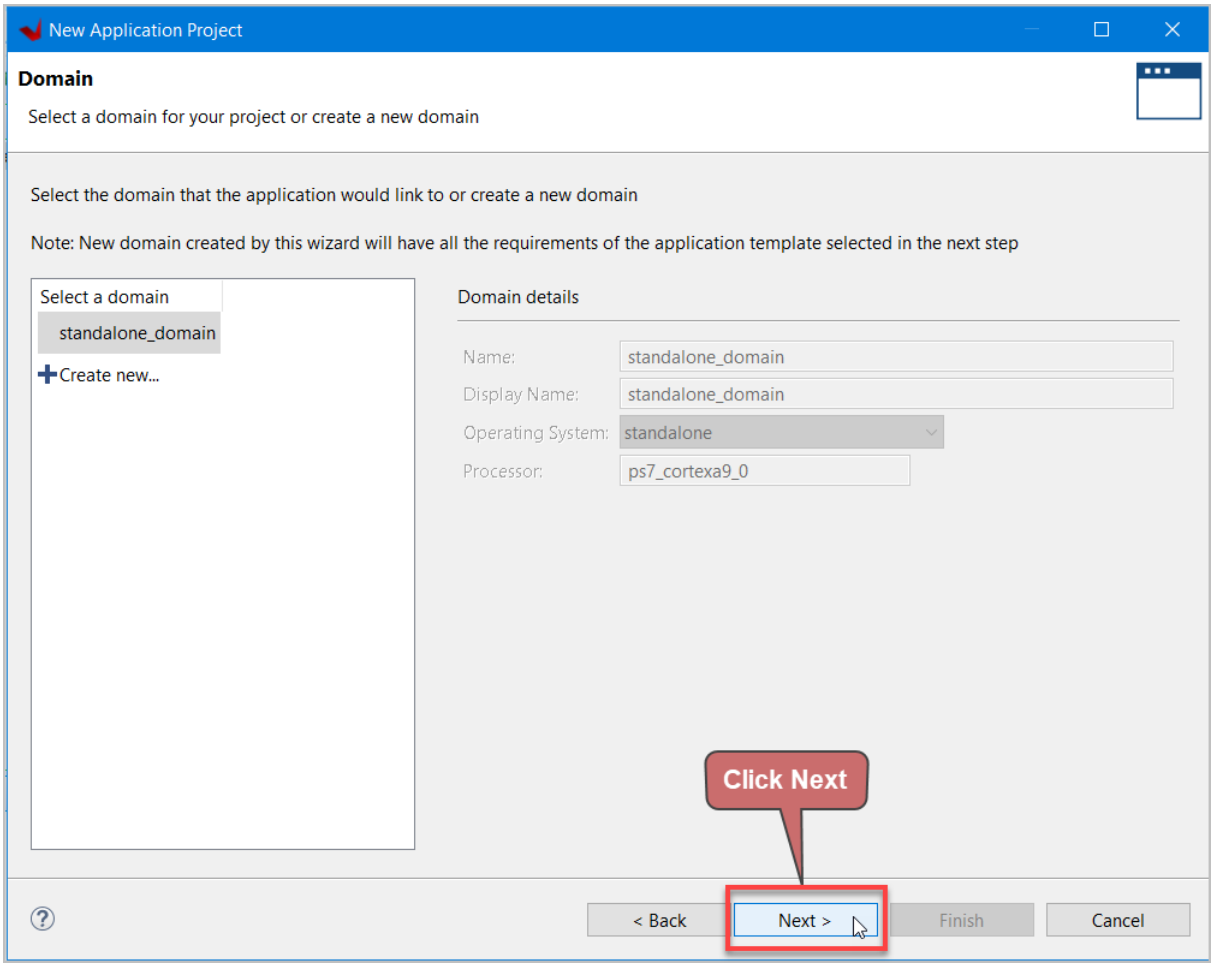


Figure 146. The standalone domain should be selected by default. Click Next to continue.

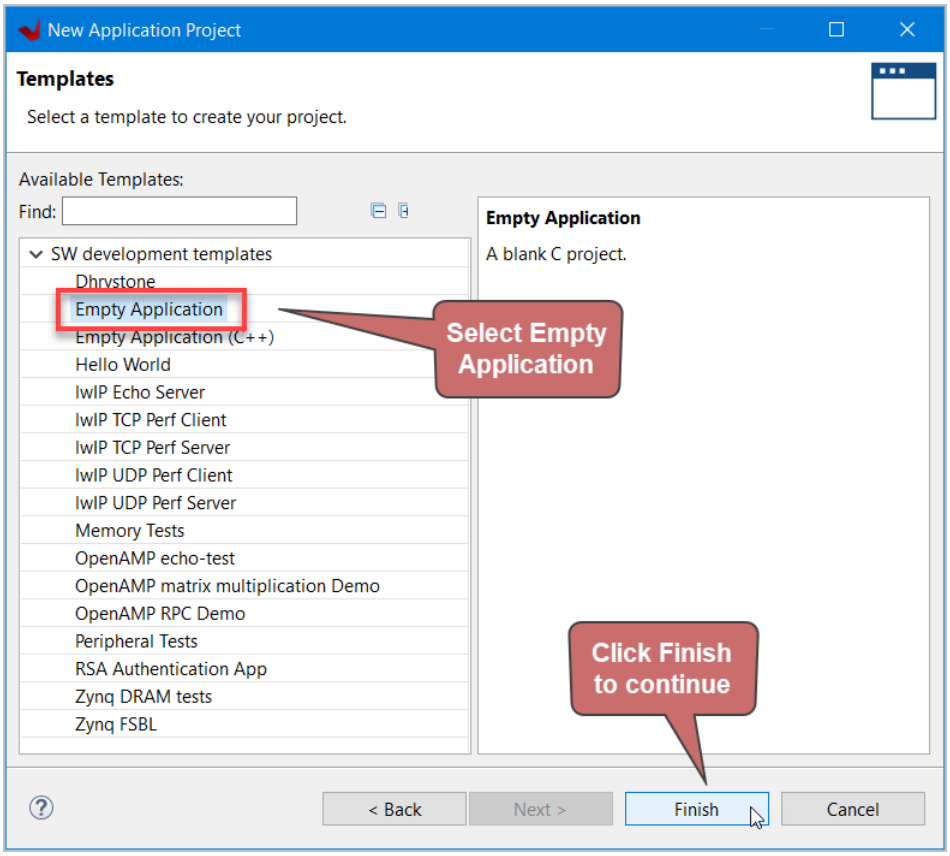


Figure 147. Select Empty Application, and click Finish.

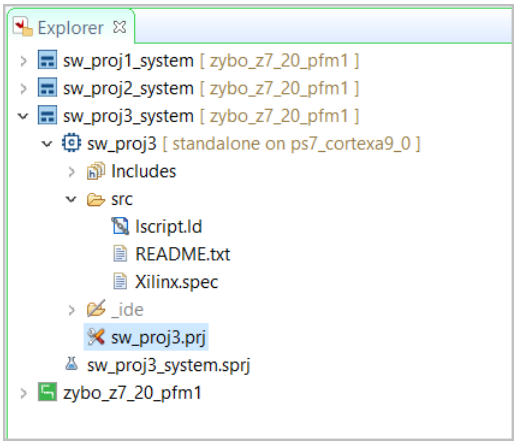


Figure 148. The empty application is created.

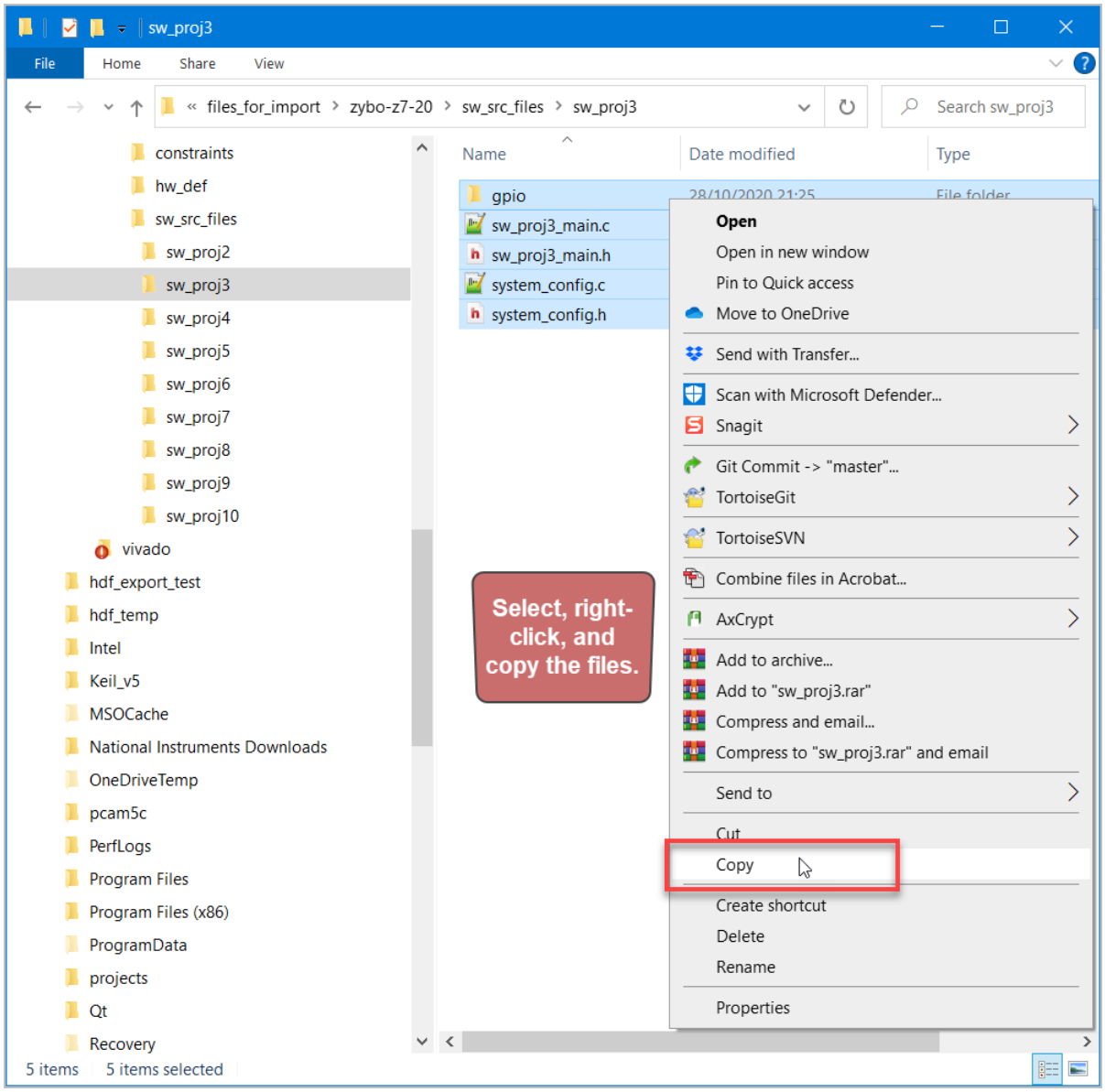


Figure 149. Select the project files, right-click, and select Copy.

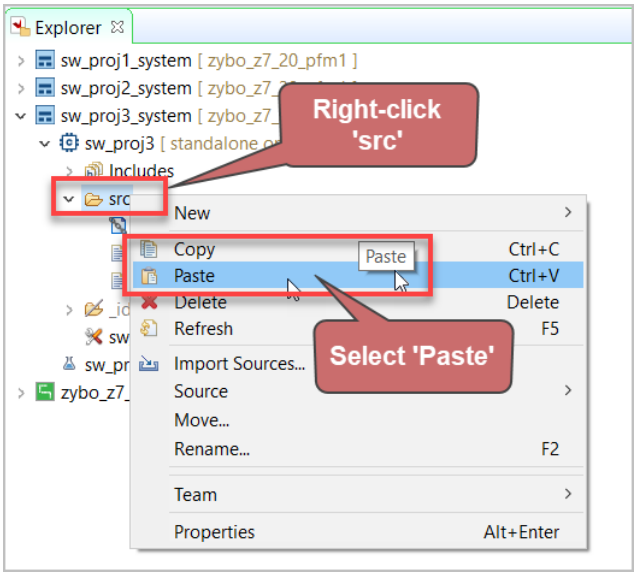


Figure 150. In Vitis, right-click the 'src' directory, and paste the files.

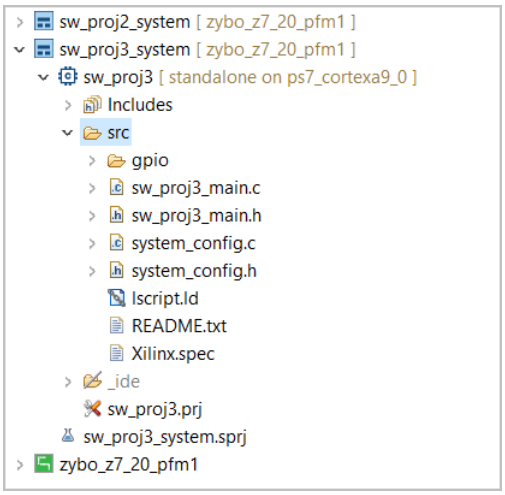


Figure 151. The project files are added.

3.6.2 Build the Project

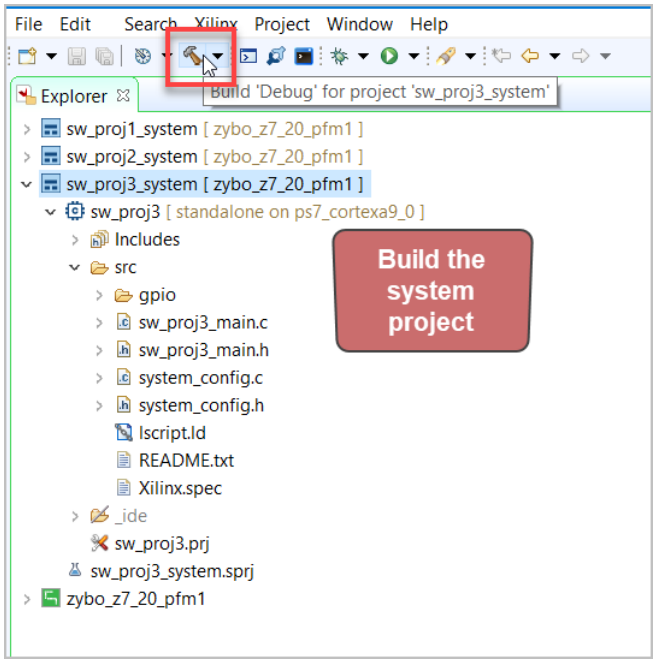


Figure 152. Build the system project

3.6.3 Run the Project

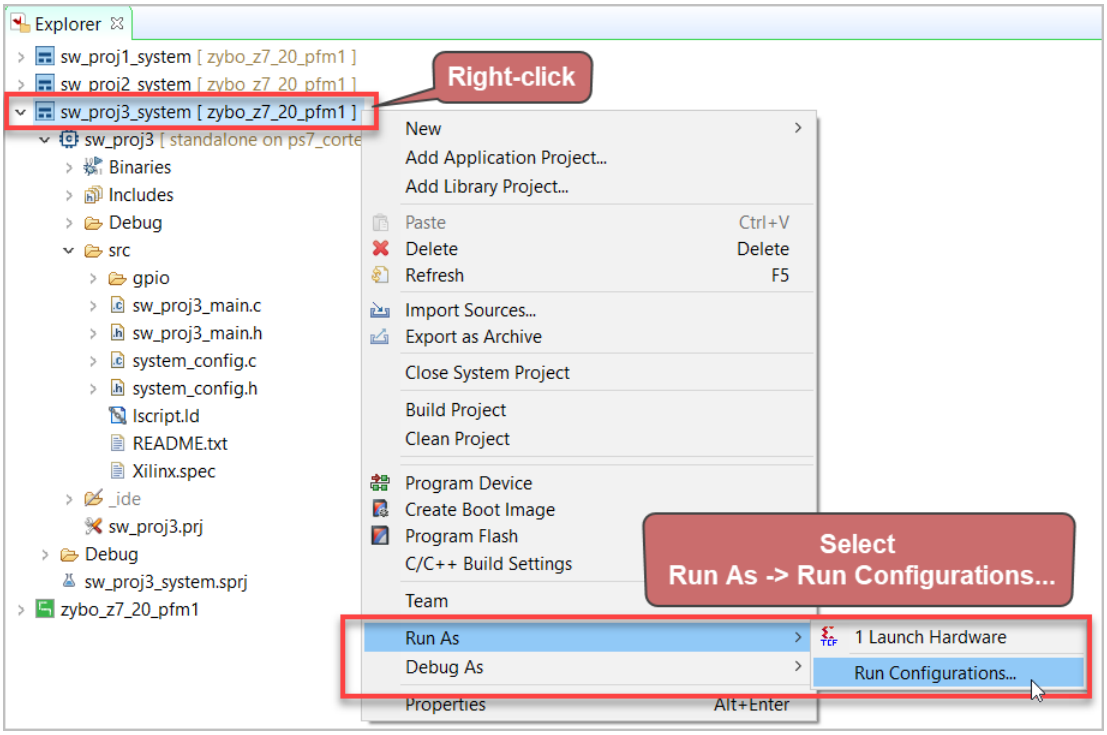


Figure 153. Right-click on the system project and select Run As -> Run Configurations.

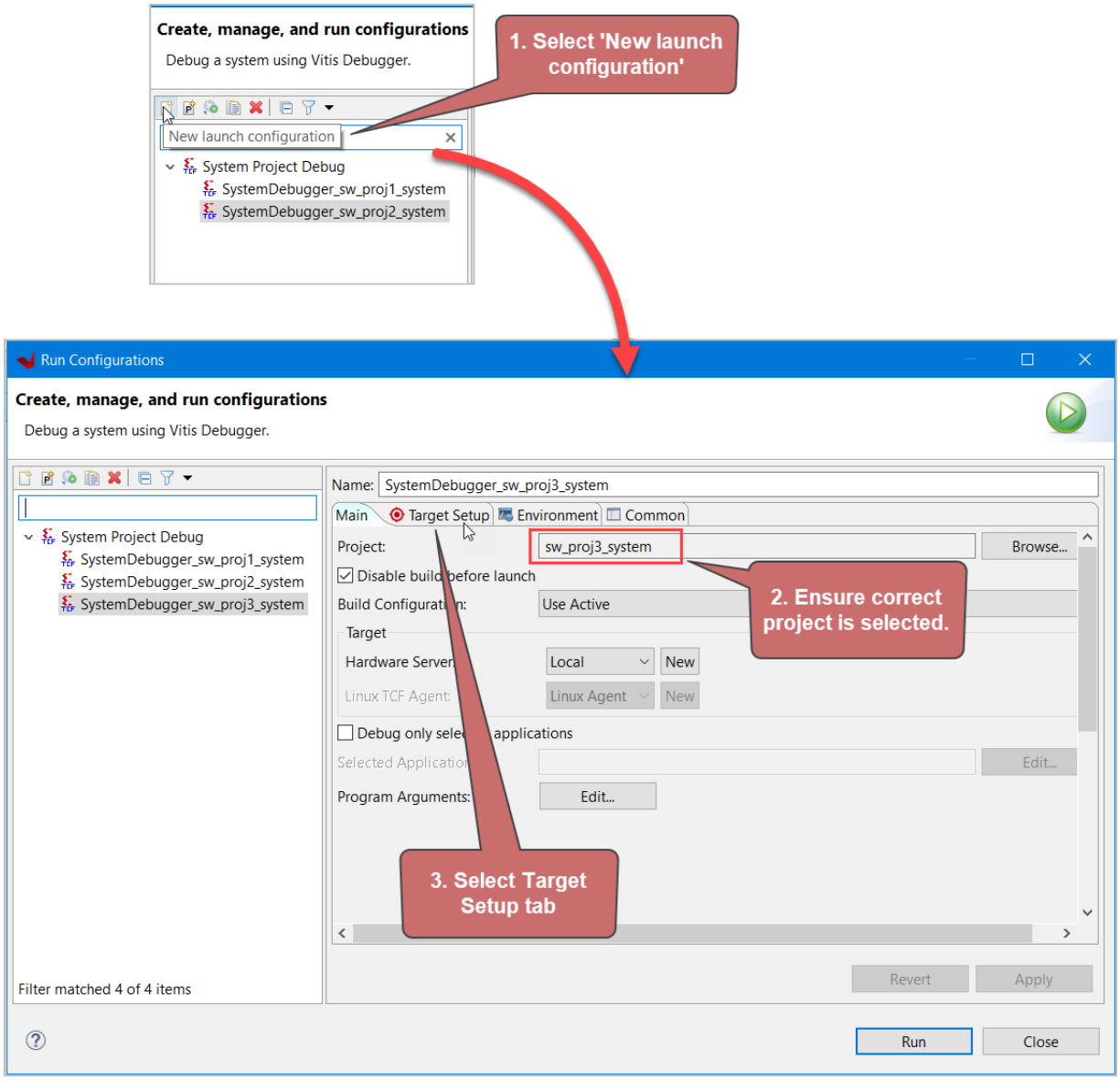


Figure 154. Create a New launch configuration

1. Click on 'New launch configuration'.
2. Ensure that "sw_proj3_system" is selected.
3. Select the Target Setup Tab.

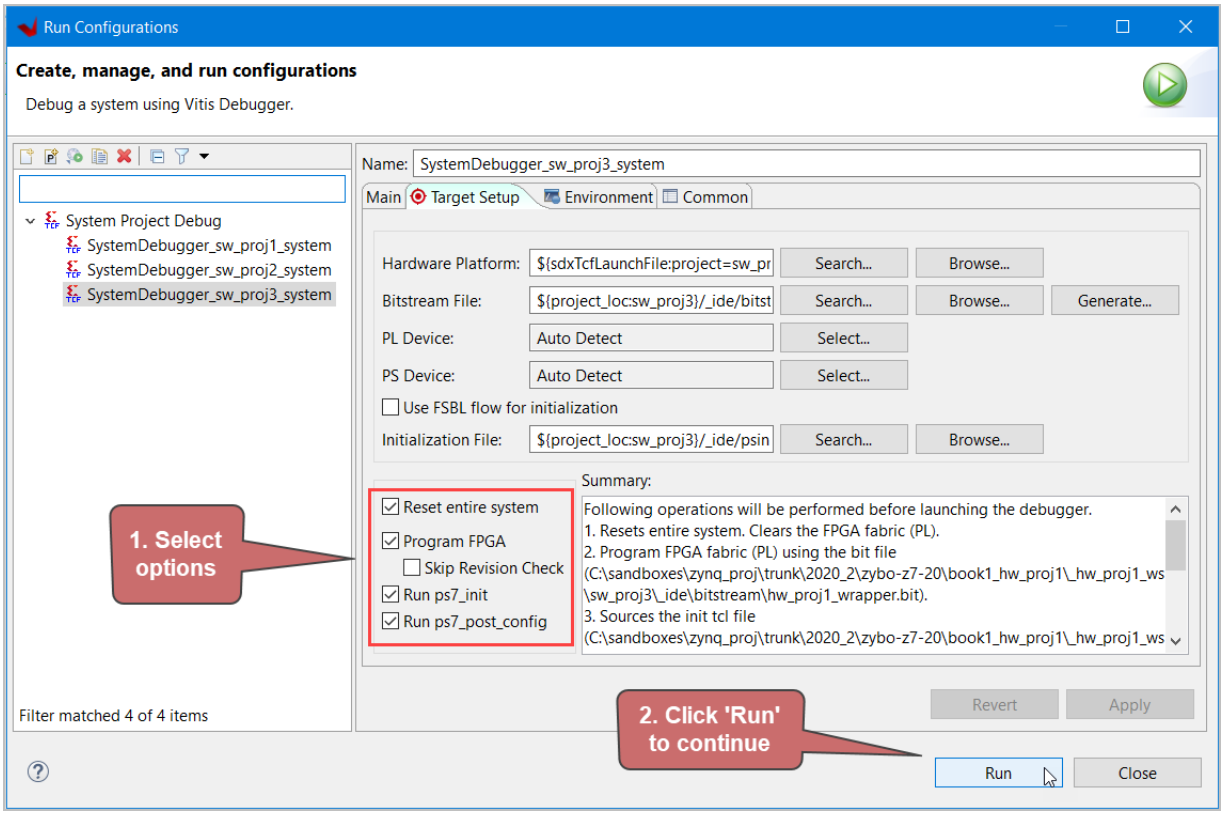


Figure 155. Run the application.

1. Select the main options as follows:
    a. Reset entire system
    b. Program FPGA
    c. Run ps7_init
    d. Run ps7_post_config

Click on 'Run' to continue. The FPGA should be programmed, and the ELF file should be downloaded to the platform, as shown in the next page.

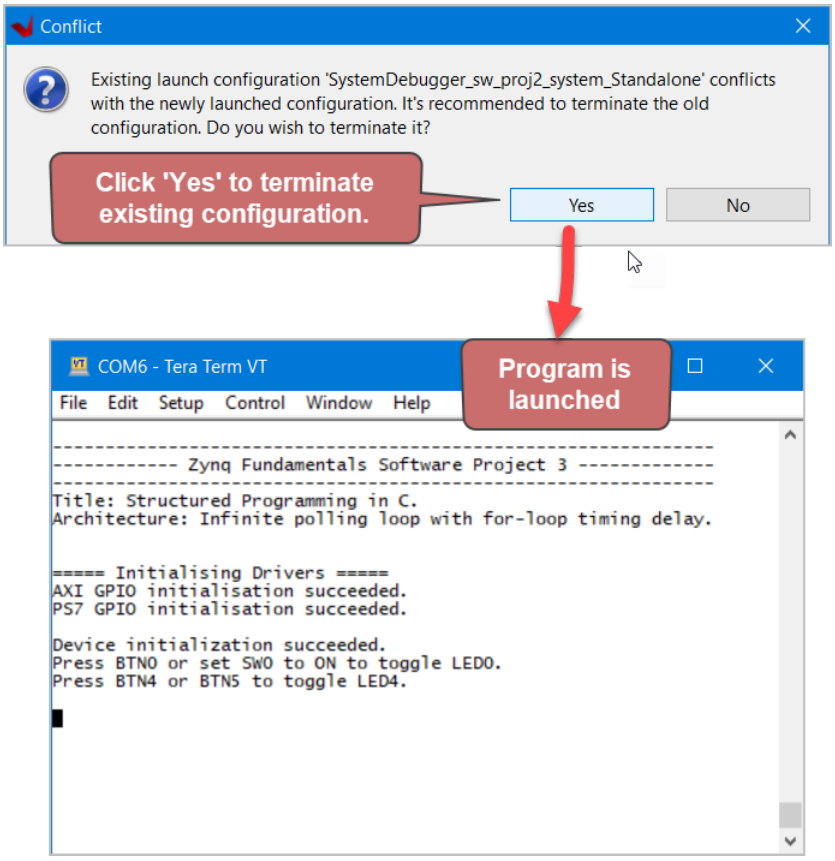


Figure 156. The project details for the software project should appear in the terminal.

If prompted, select 'Yes' to terminate any existing configuration. The FPGA will be programmed and the application will be downloaded to the processor. The terminal program should display the results for launching Software Project 3.

3.7 System (Software) Project 4: SCU Timing

3.7.1 Create the Project

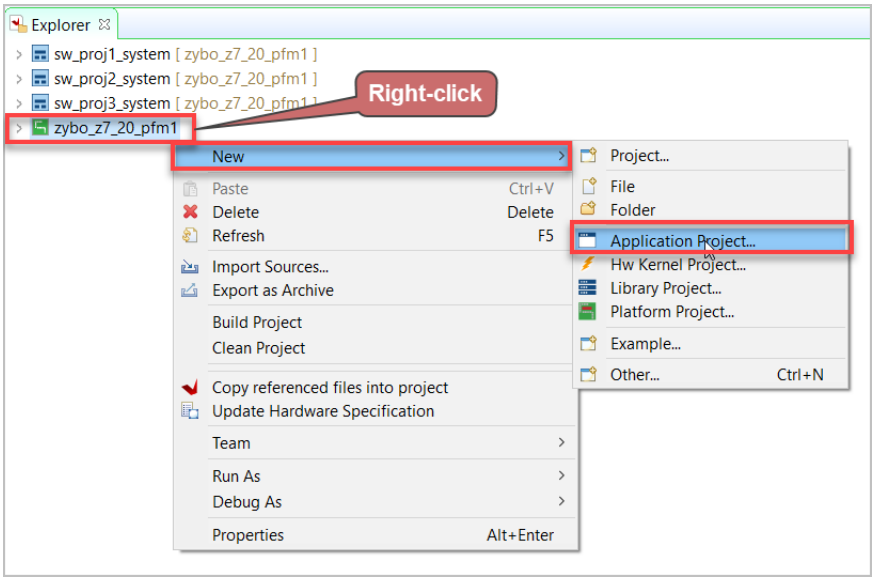


Figure 157. Right-click on the platform and select New-> Application Project.

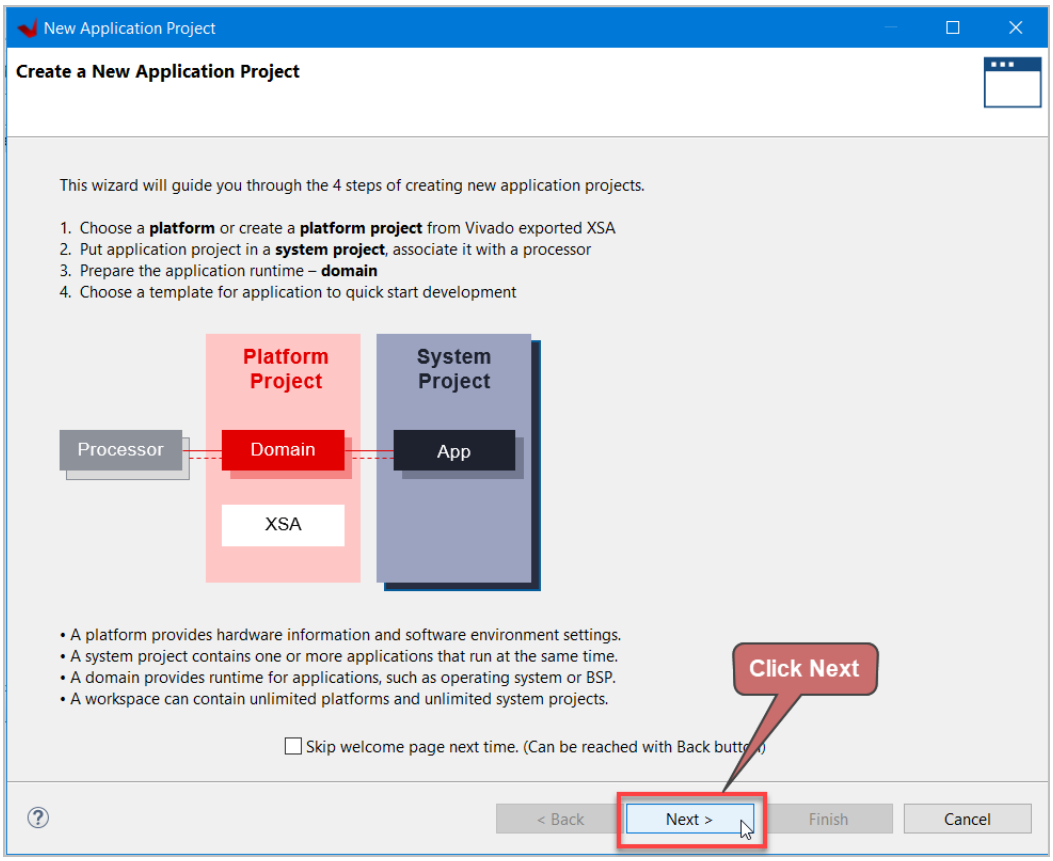


Figure 158. Click Next to continue.

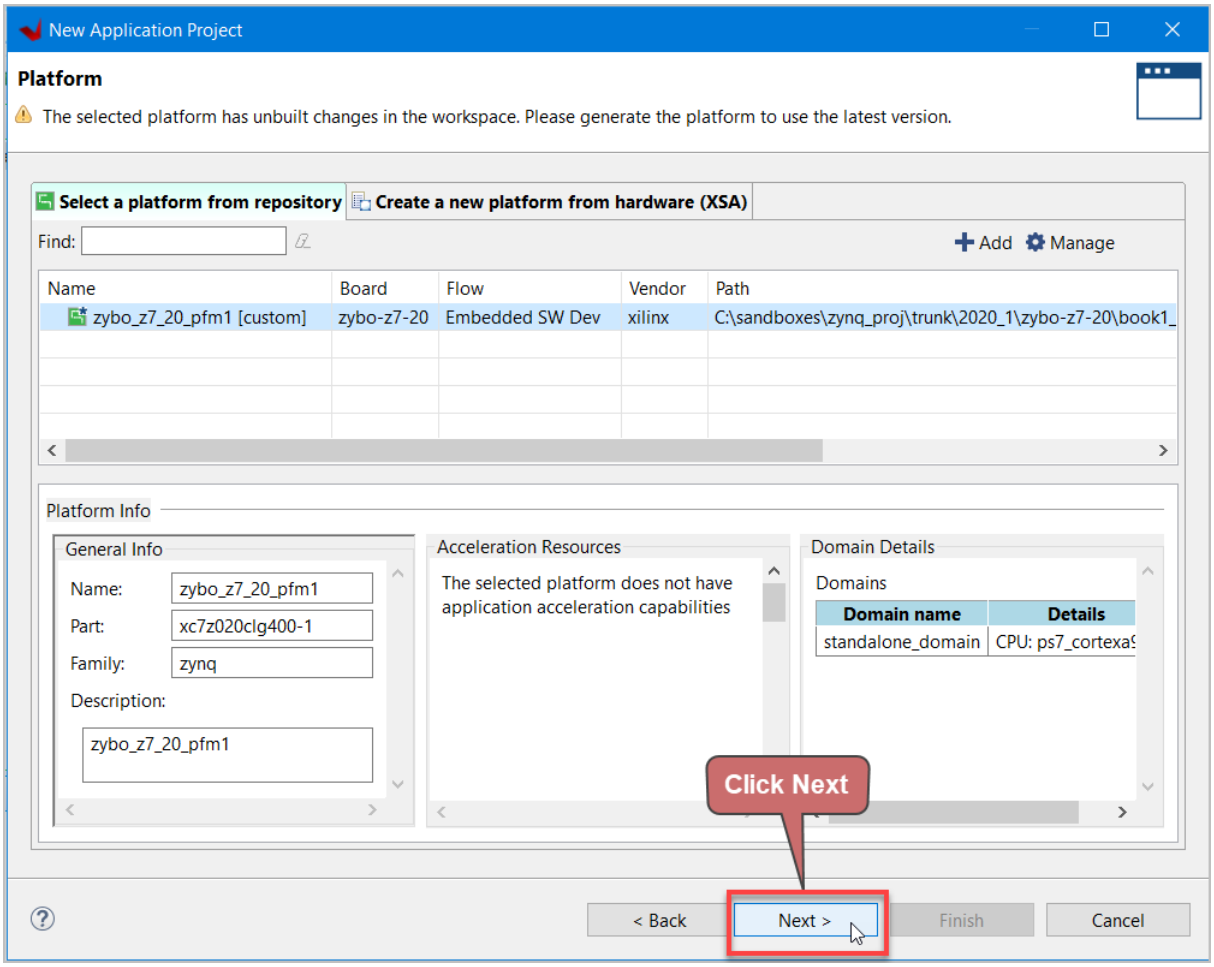


Figure 159. Leave defaults, and click Next to continue.

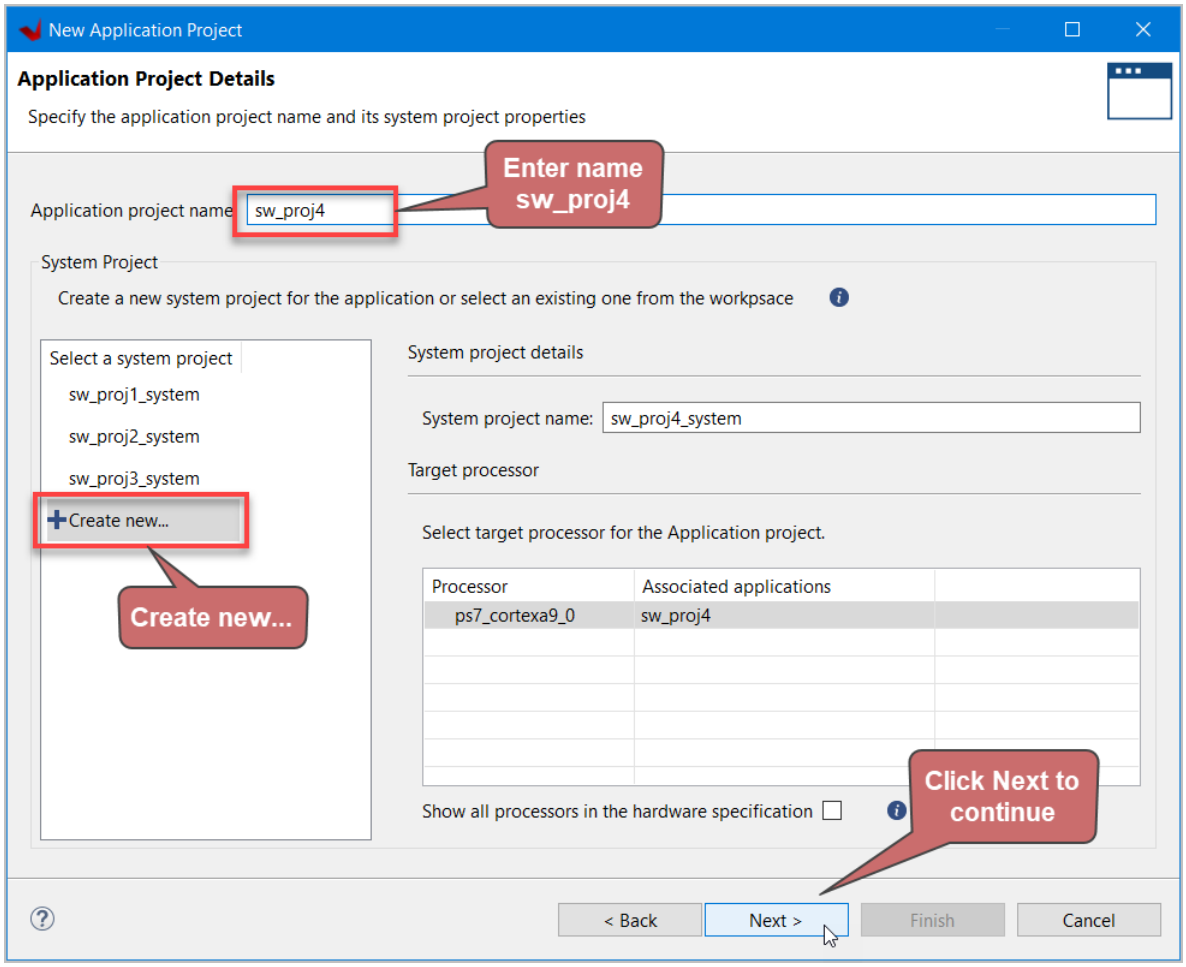


Figure 160. Set the project name to 'sw_proj4', ensure Create New... is selected, and press Next to continue.

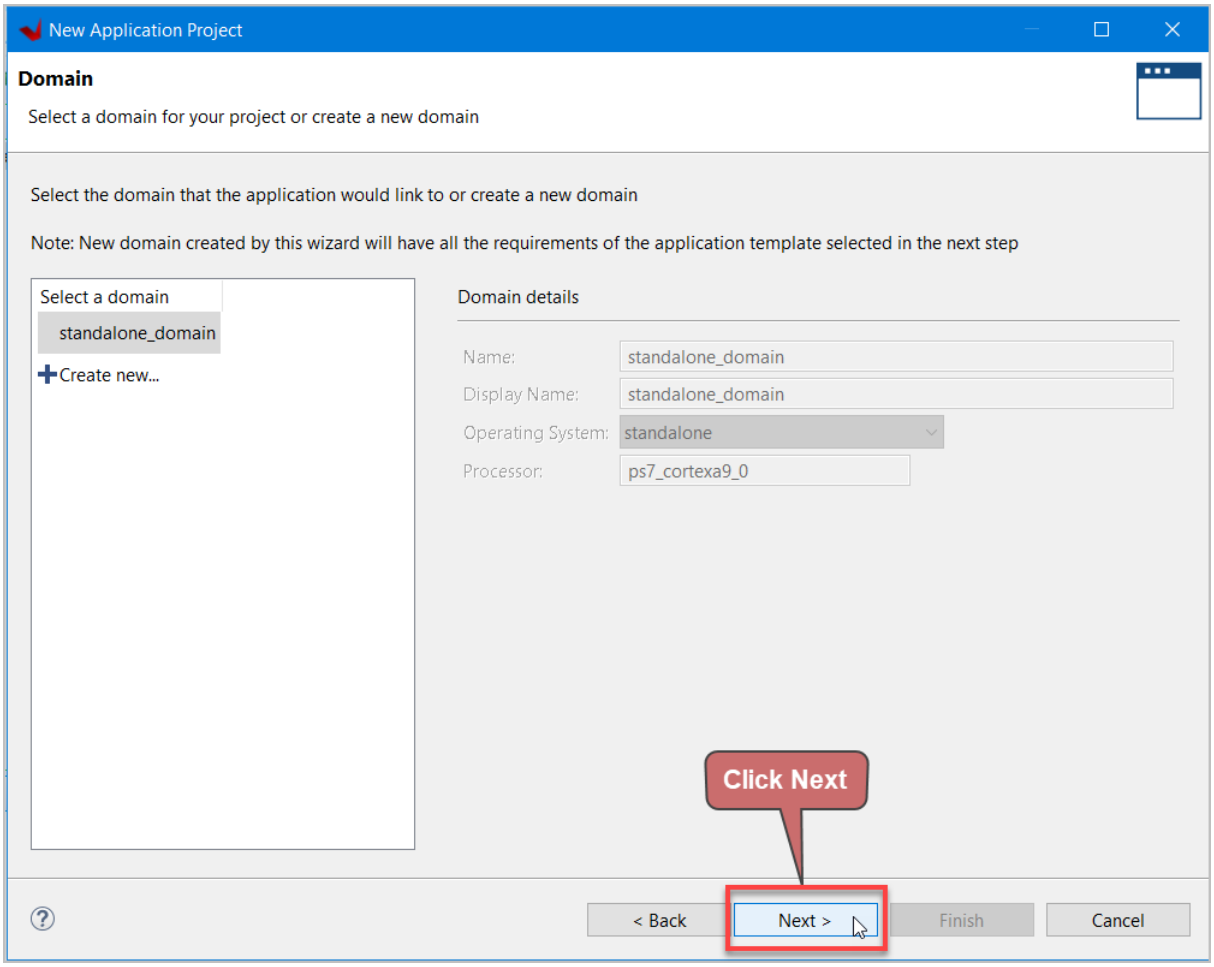


Figure 161. The standalone domain should be selected by default. Click Next to continue.

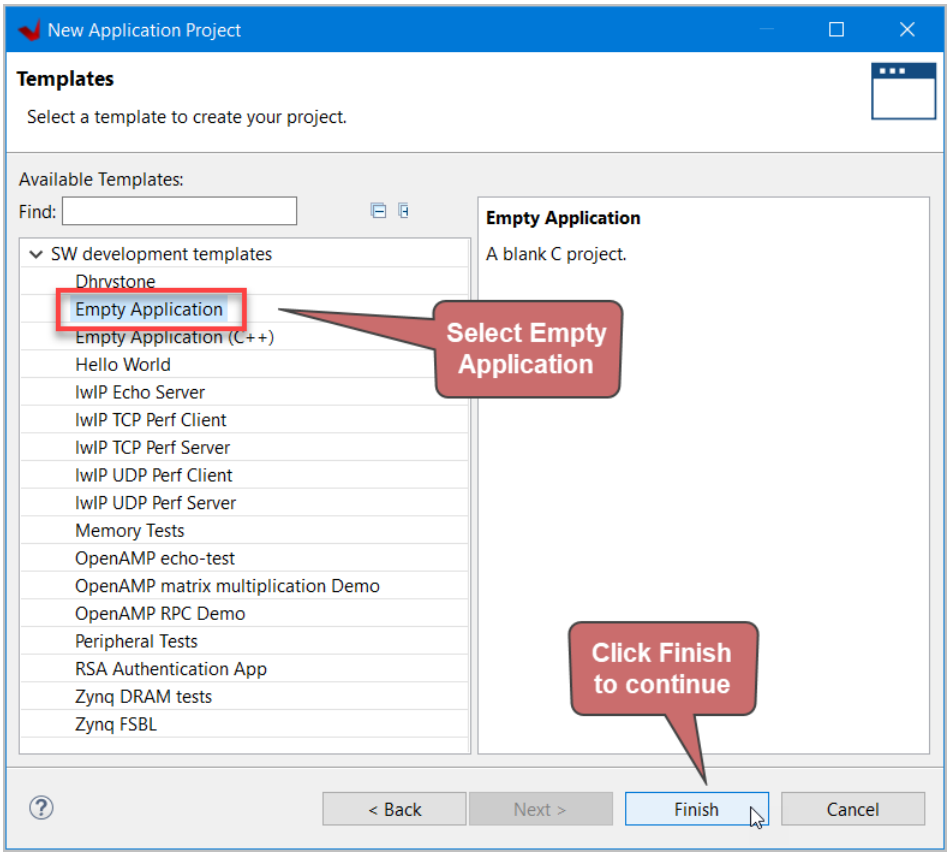


Figure 162. Select Empty Application, and click Finish.

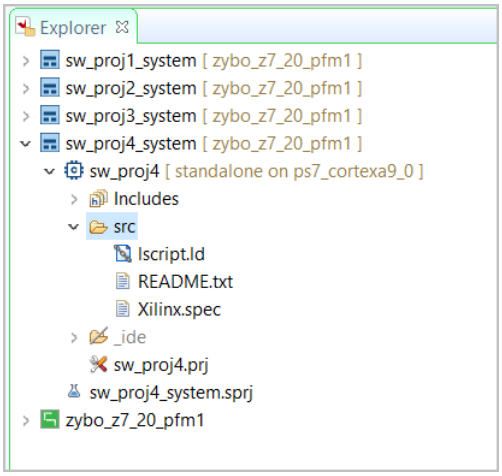


Figure 163. The empty application is created.

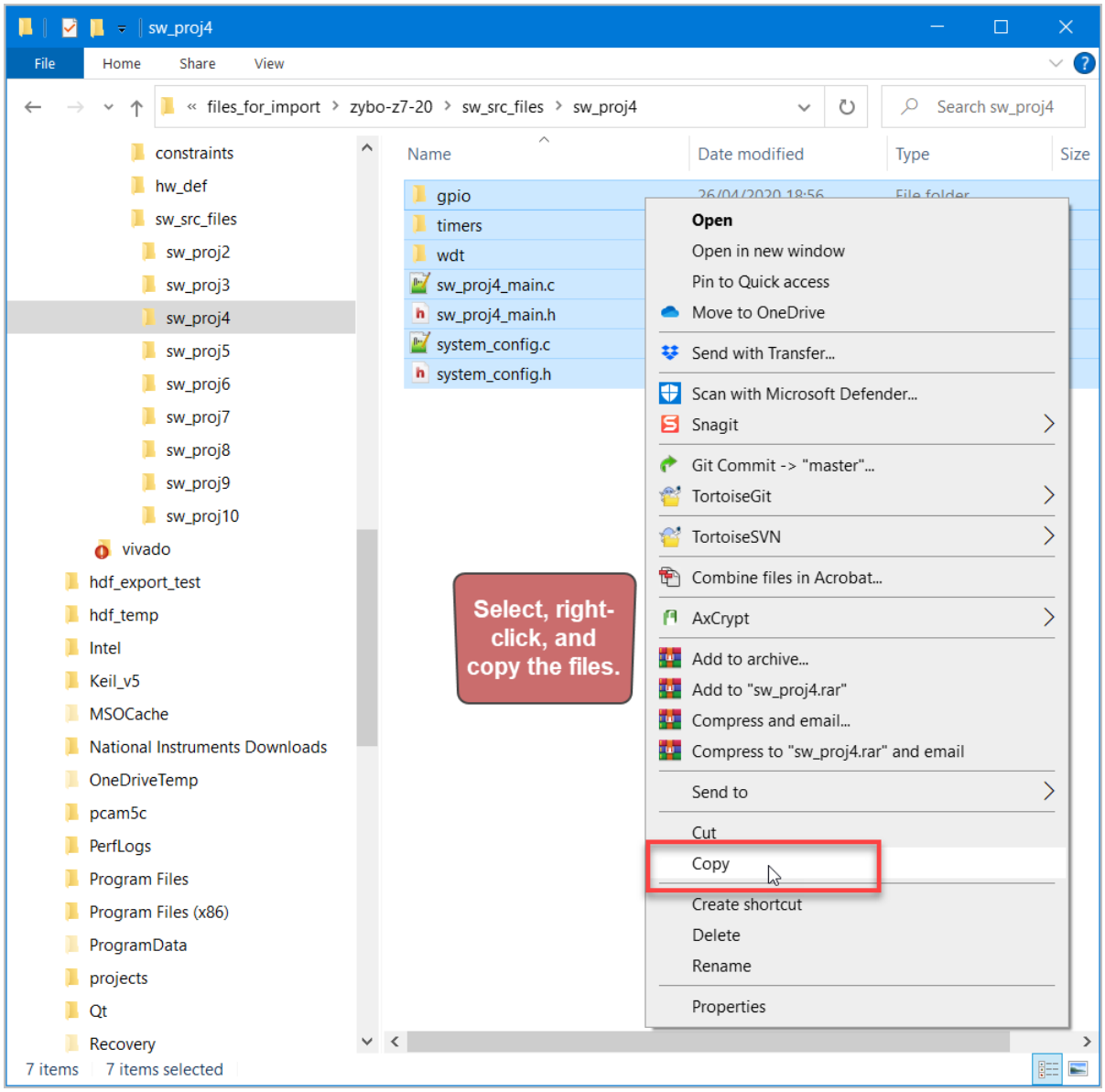


Figure 164. Select the project files, right-click, and select Copy.

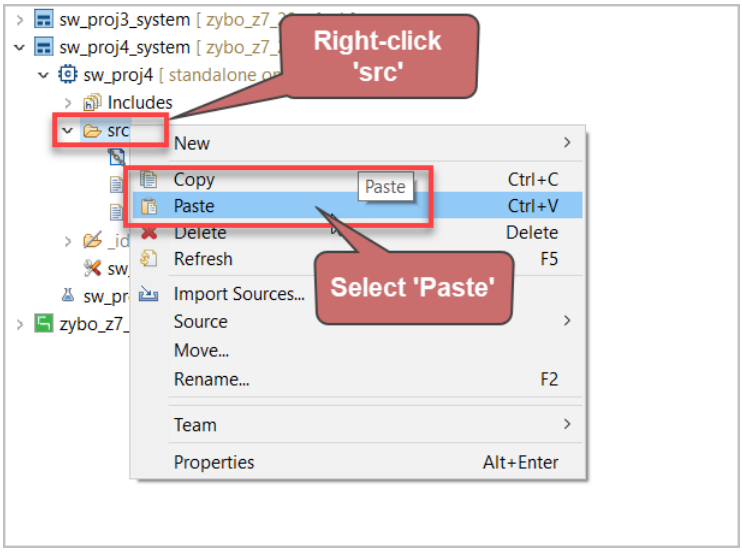


Figure 165. In Vitis, right-click the 'src' directory, and paste the files.

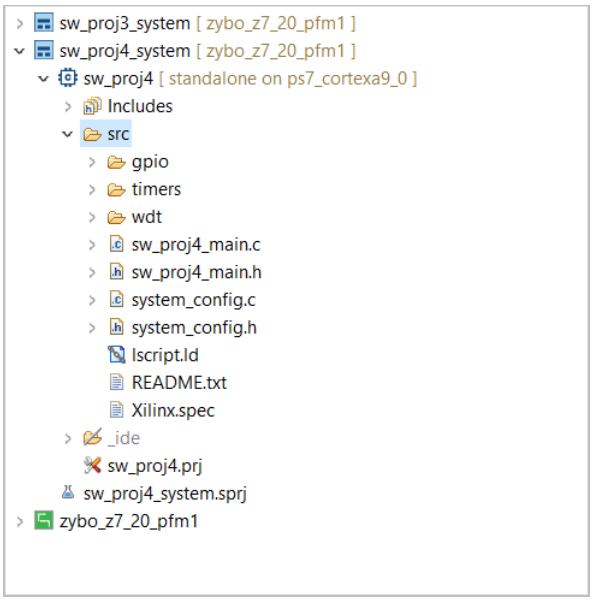


Figure 166. The project files are added.

3.7.2 Build the Project

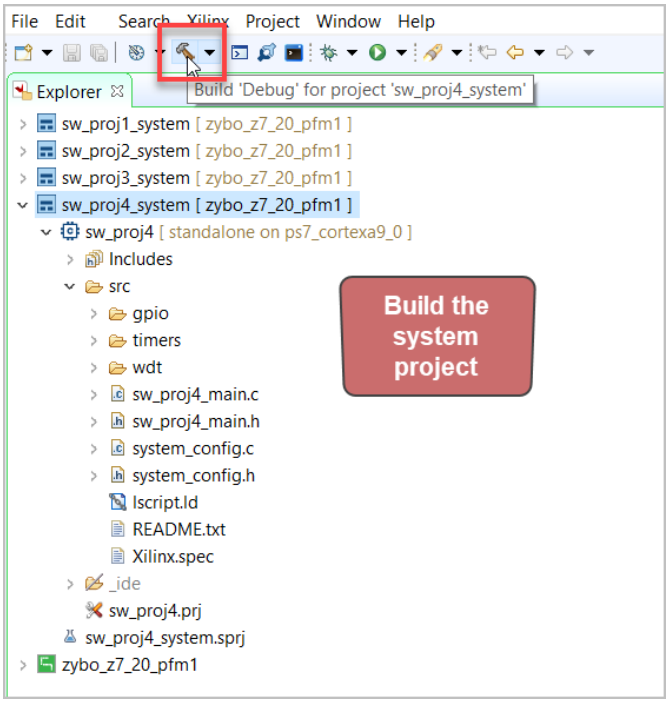


Figure 167. Build the system project

3.7.3 Run the Project

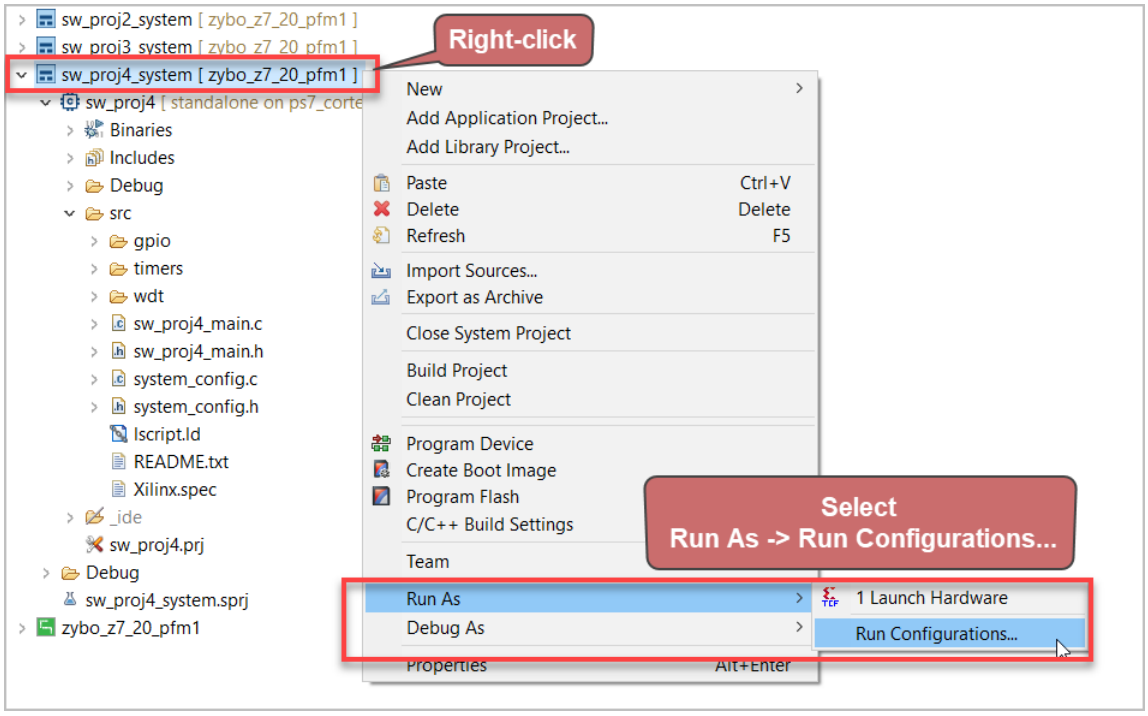


Figure 168. Right-click on the system project and select Run As -> Run Configurations.

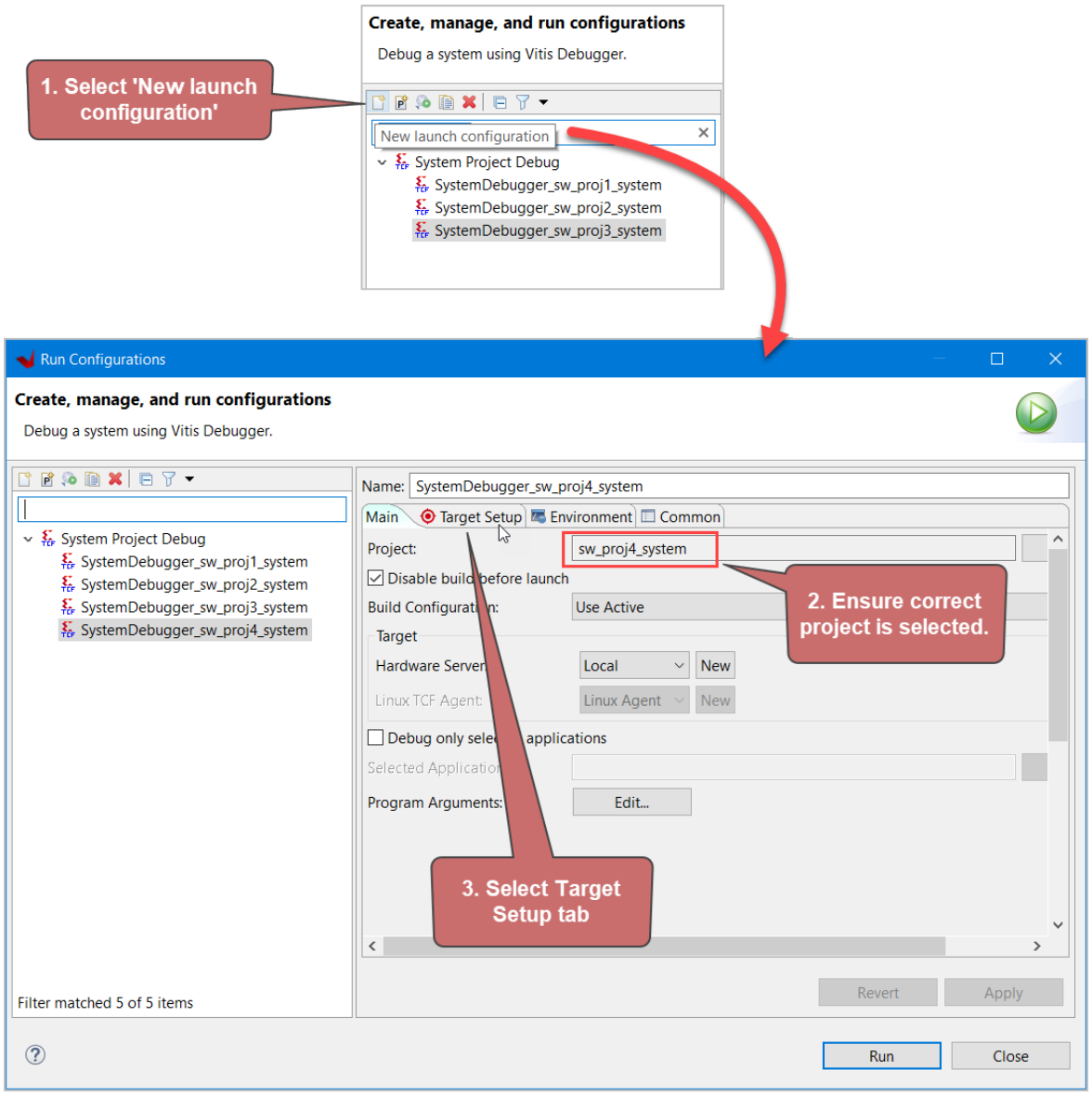


Figure 169. Create a New launch configuration

1. Click on 'New launch configuration'.
2. Ensure that "sw_proj4_system" is selected.
3. Select the Target Setup Tab.

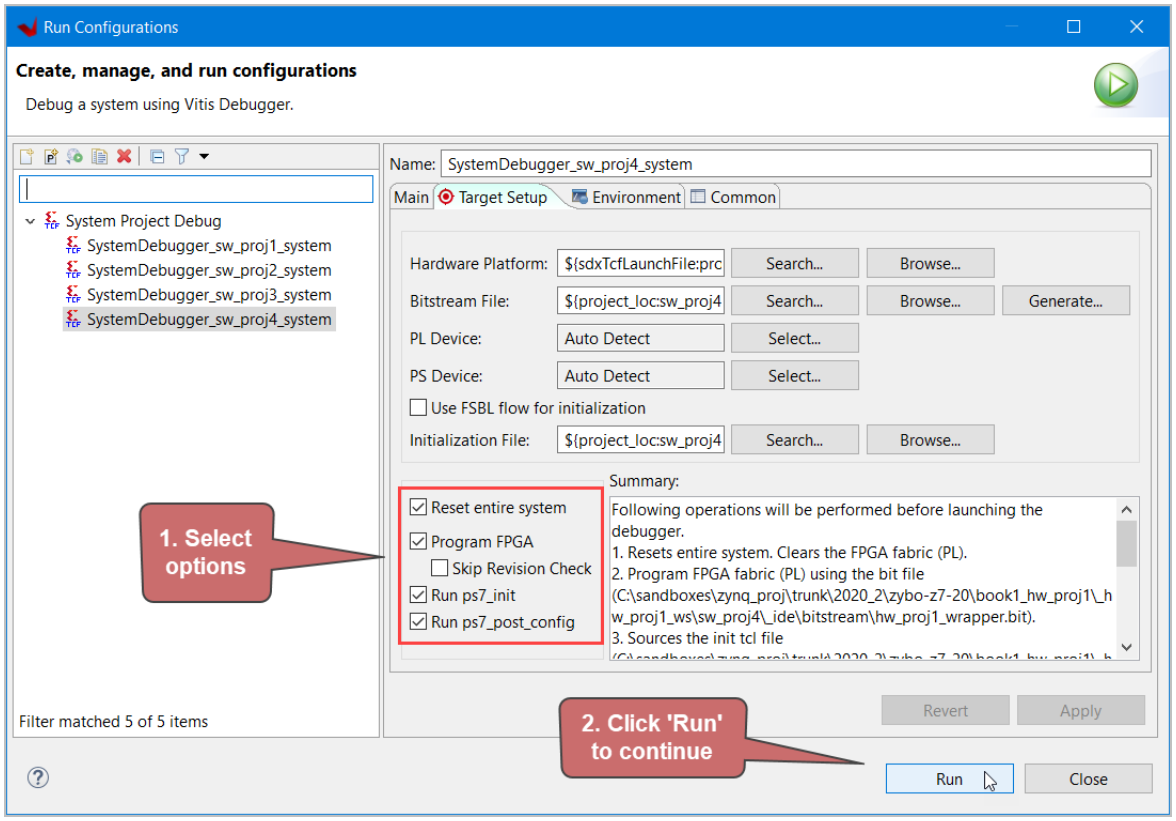


Figure 170. Run the application.

1. Select the main options as follows:
    a. Reset entire system
    b. Program FPGA
    c. Run ps7_init
    d. Run ps7_post_config

Click on 'Run' to continue. The FPGA should be programmed, and the ELF file should be downloaded to the platform, as shown in the next page.

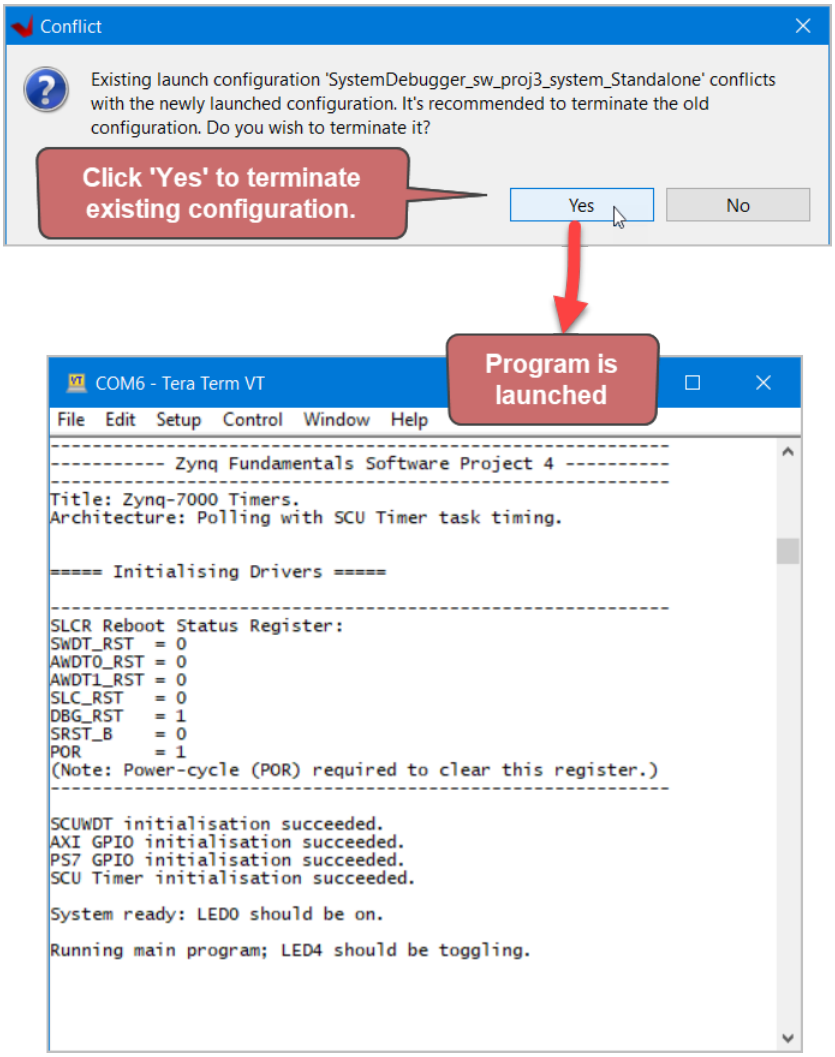


Figure 171. The project details for the software project should appear in the terminal.

If prompted, select 'Yes' to terminate any existing configuration. The FPGA will be programmed and the application will be downloaded to the processor. The terminal program should display the results for launching Software Project 4.

3.8 System (Software) Project 5: Zynq Interrupts

3.8.1 Create the Project

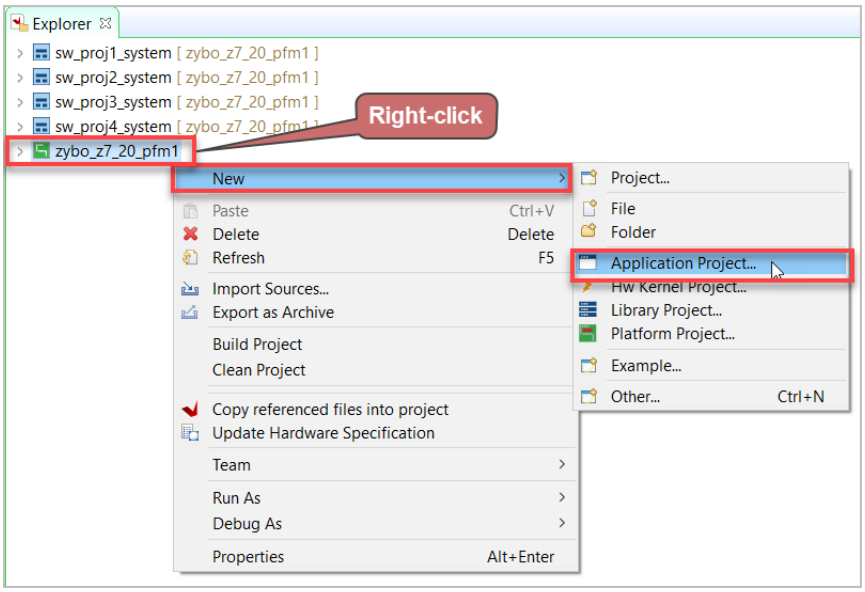


Figure 172. Right-click on the platform and select New-> Application Project.

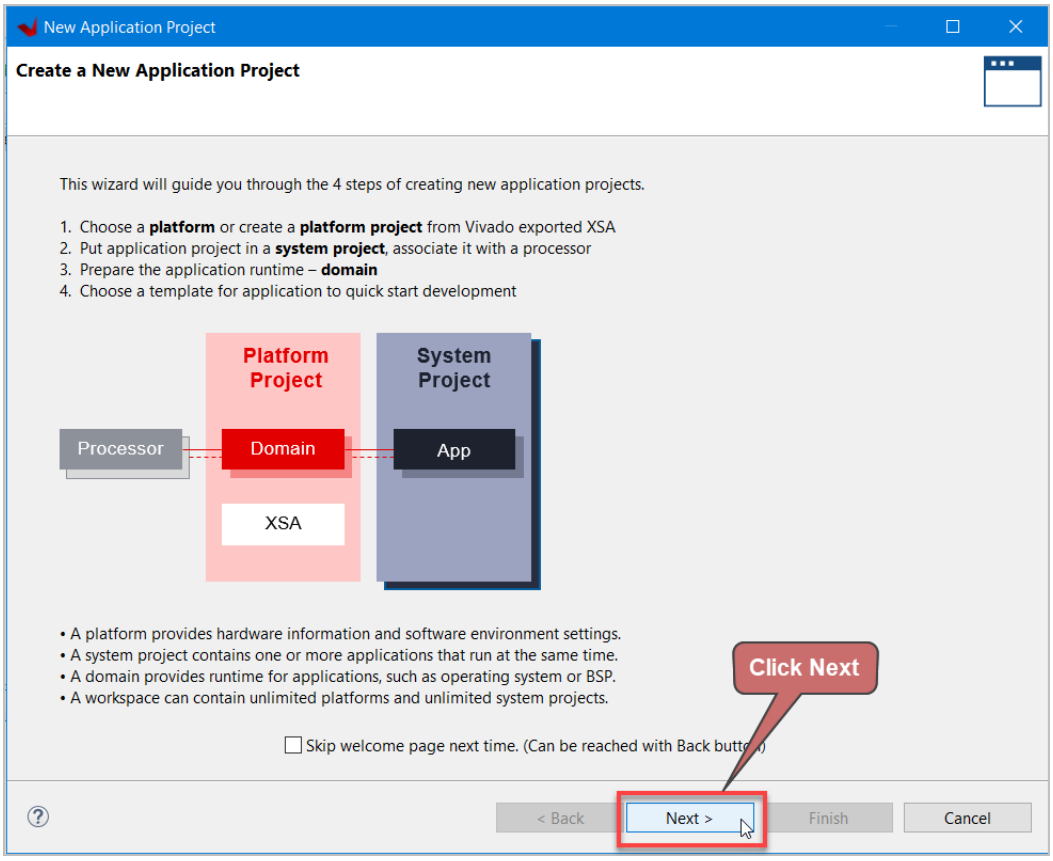


Figure 173. Click Next to continue.

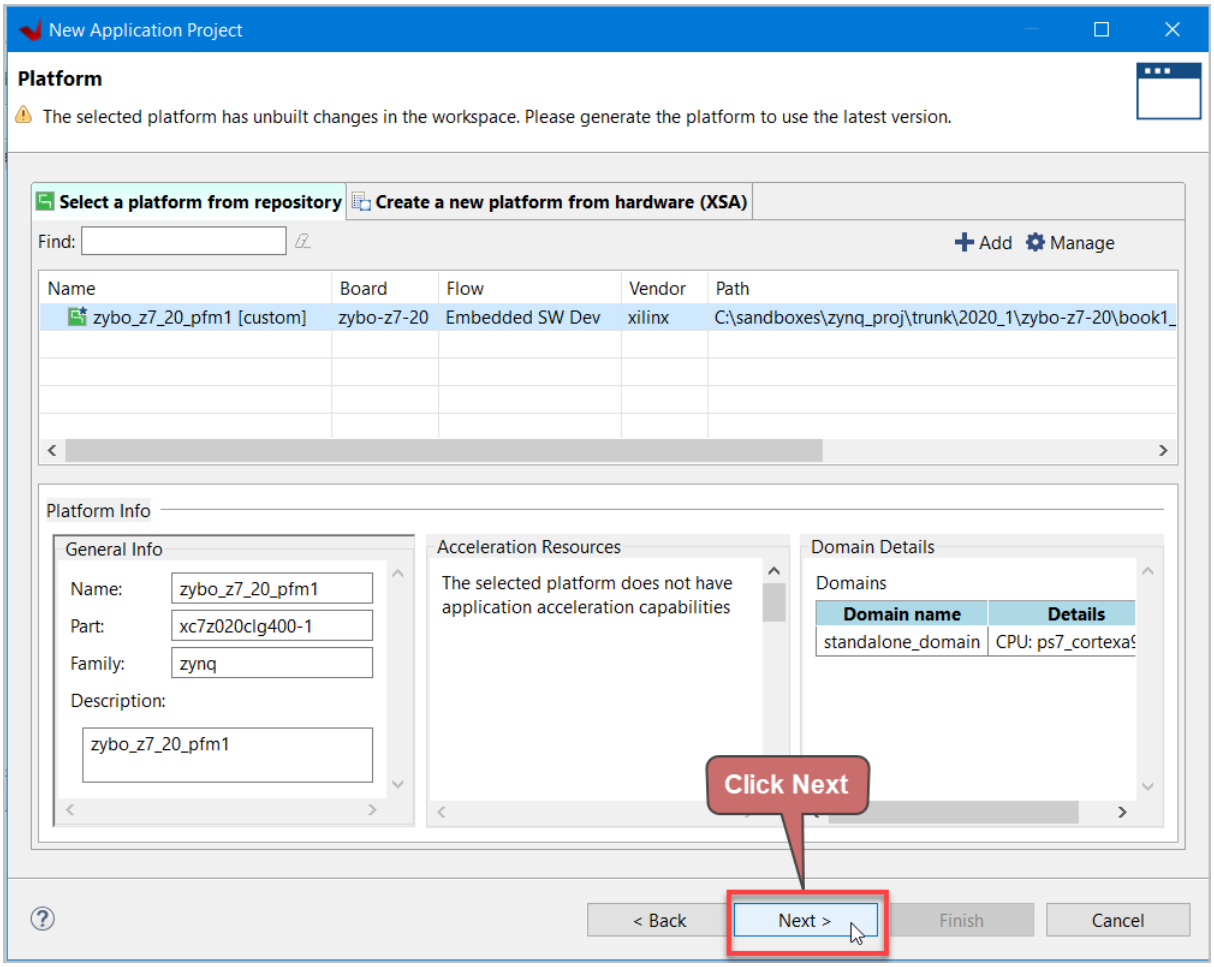


Figure 174. Leave defaults, and click Next to continue.

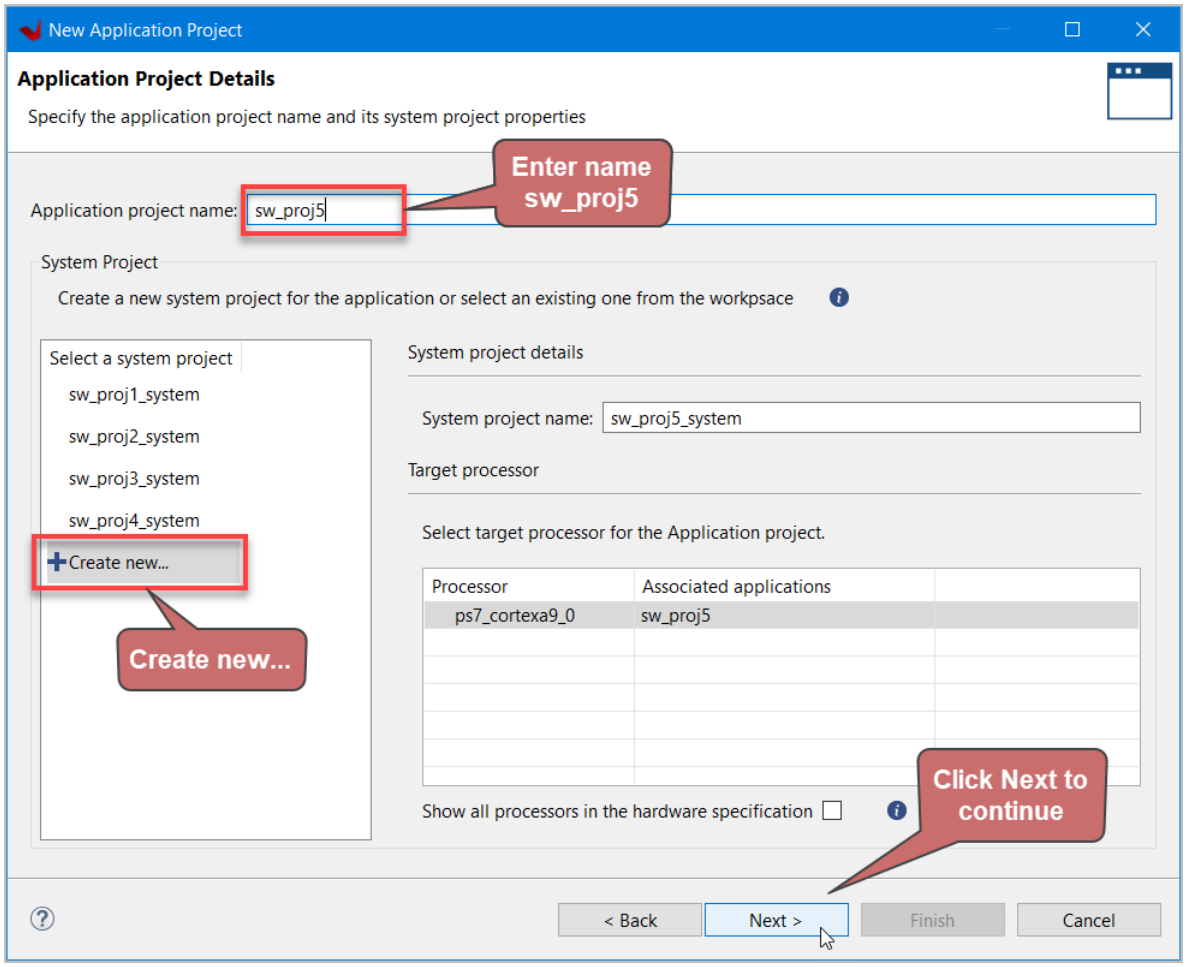


Figure 175. Set the project name to 'sw_proj5', ensure Create New... is selected, and press Next to continue.

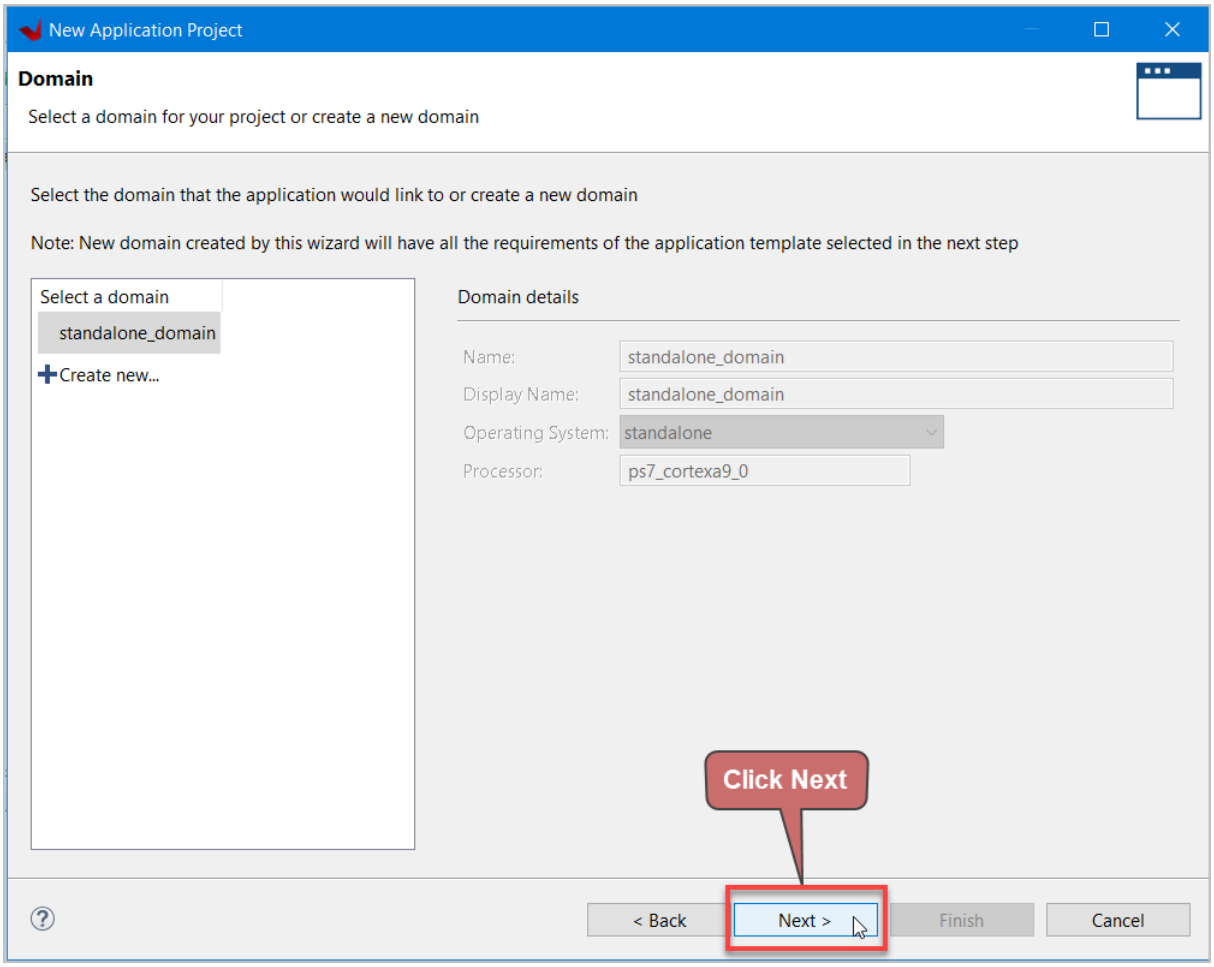


Figure 176. The standalone domain should be selected by default. Click Next to continue.

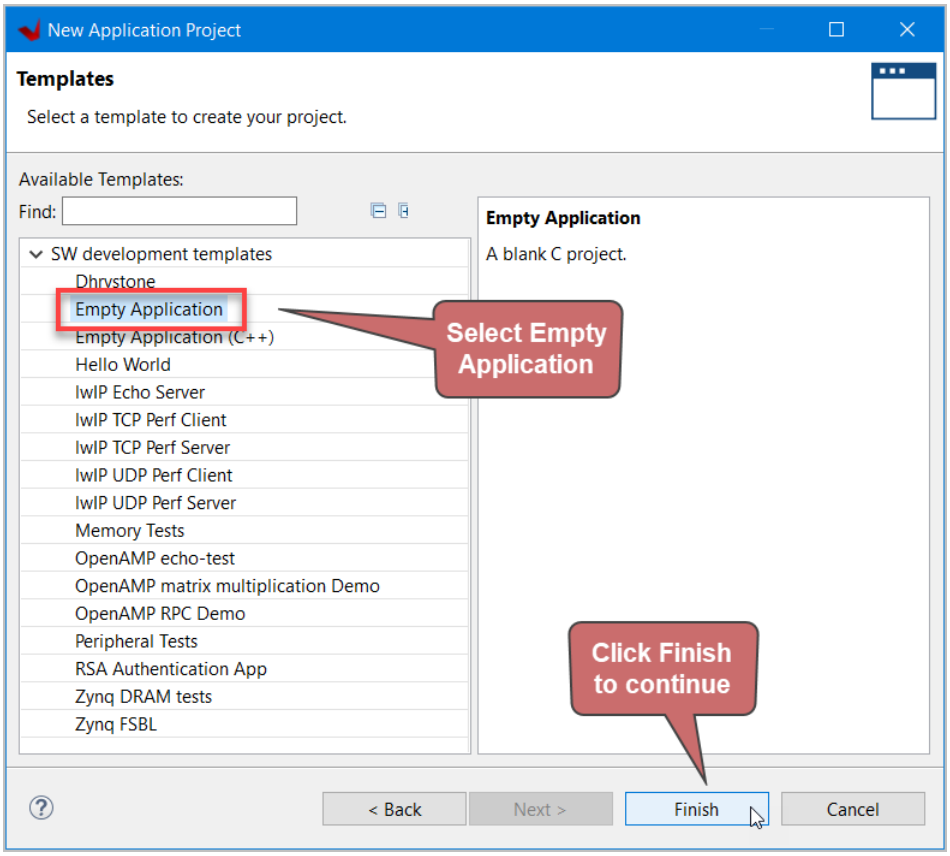


Figure 177. Select Empty Application, and click Finish.

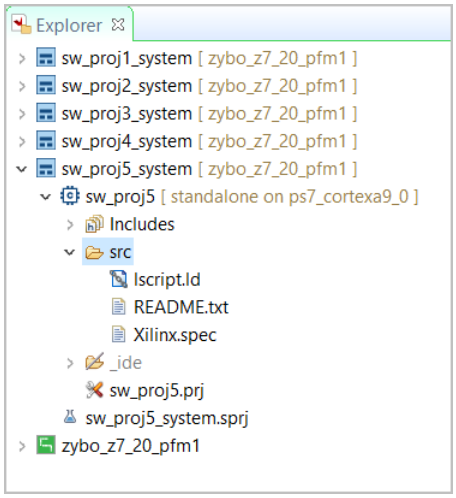


Figure 178. The empty application is created.

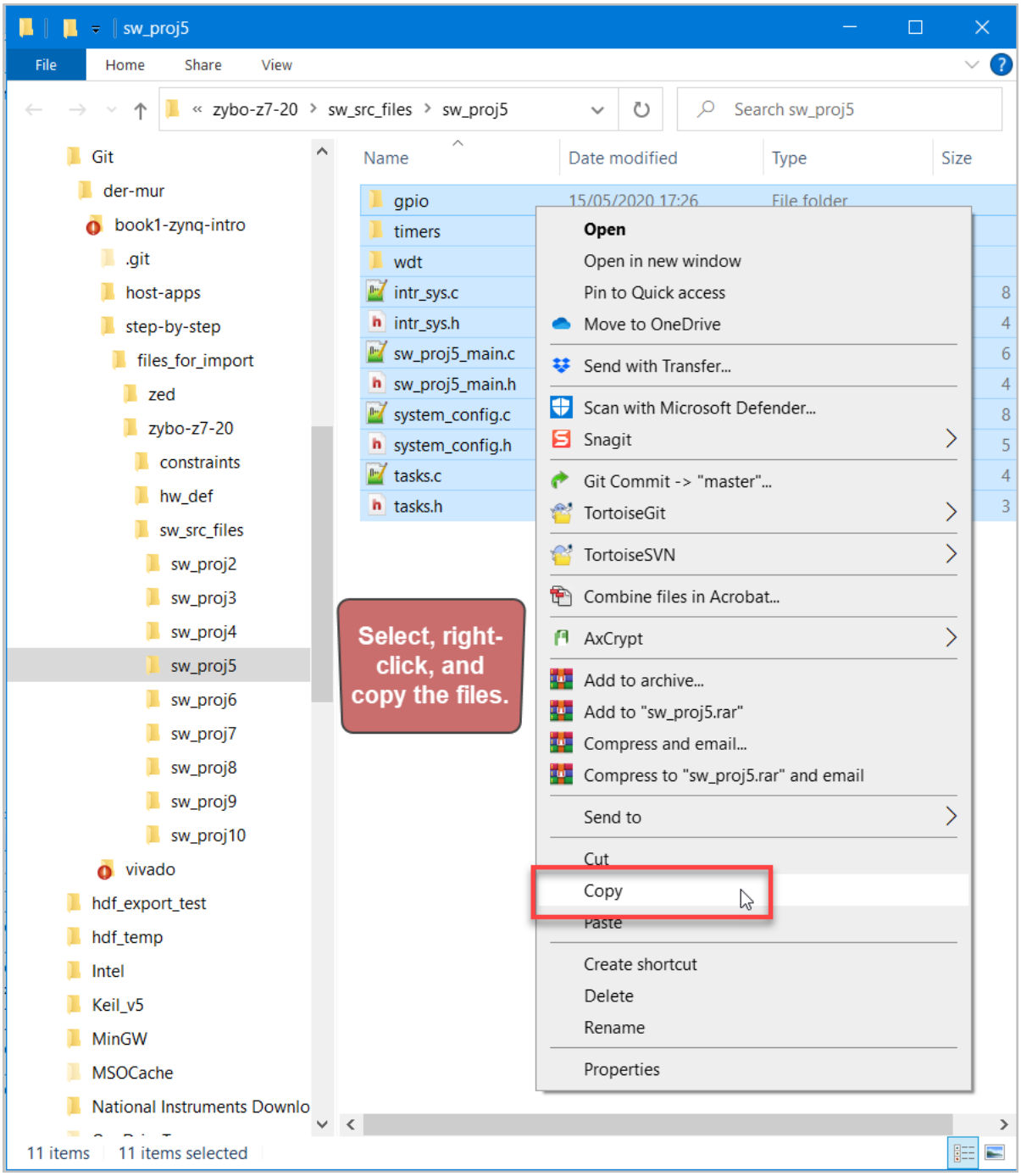


Figure 179. Select the project files, right-click, and select Copy.

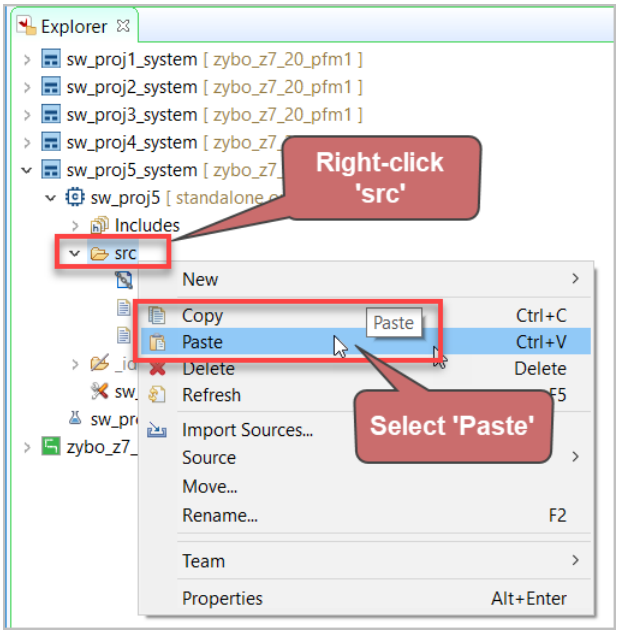


Figure 180. In Vitis, right-click the 'src' directory, and paste the files.

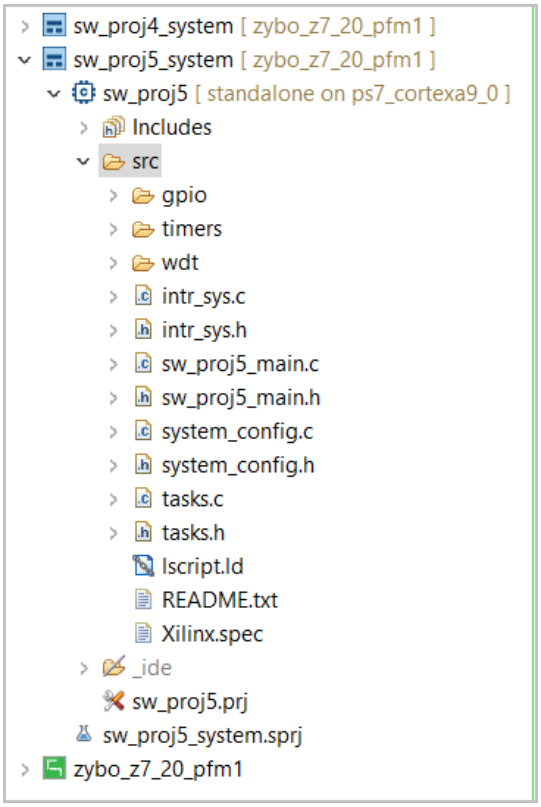


Figure 181. The project files are added.

3.8.2 Build the Project

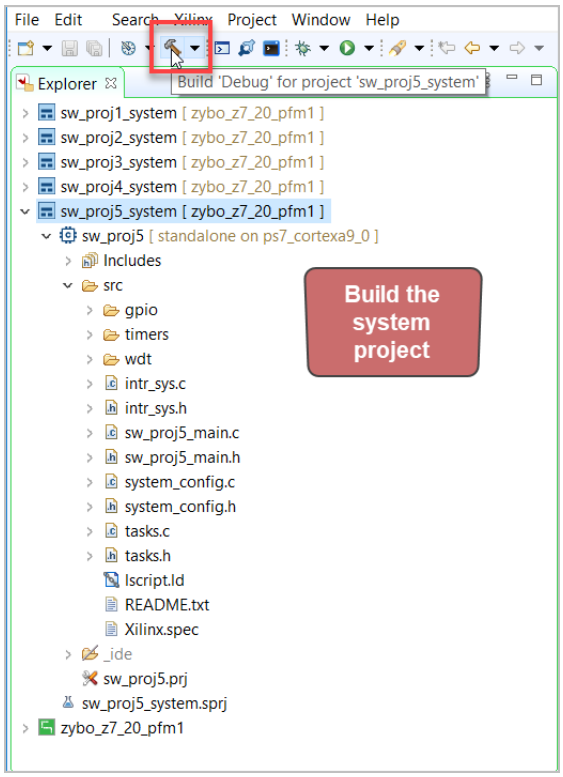


Figure 182. Build the system project

3.8.3 Run the Project

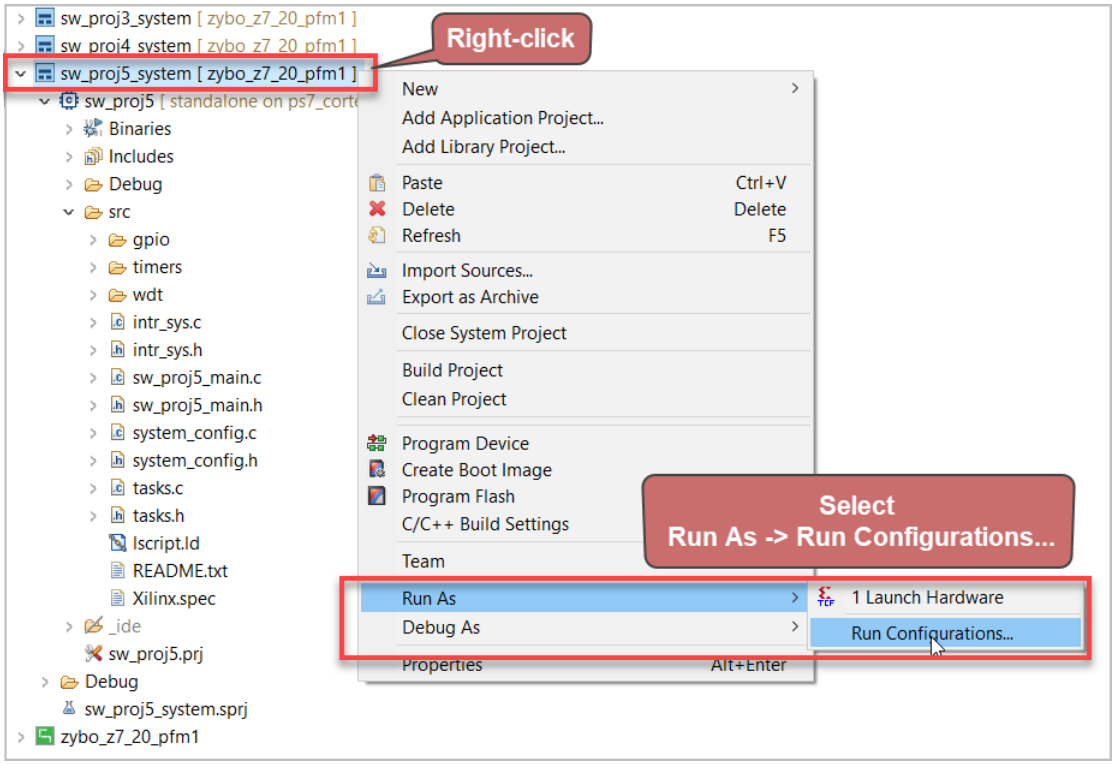


Figure 183. Right-click on the system project and select Run As -> Run Configurations.

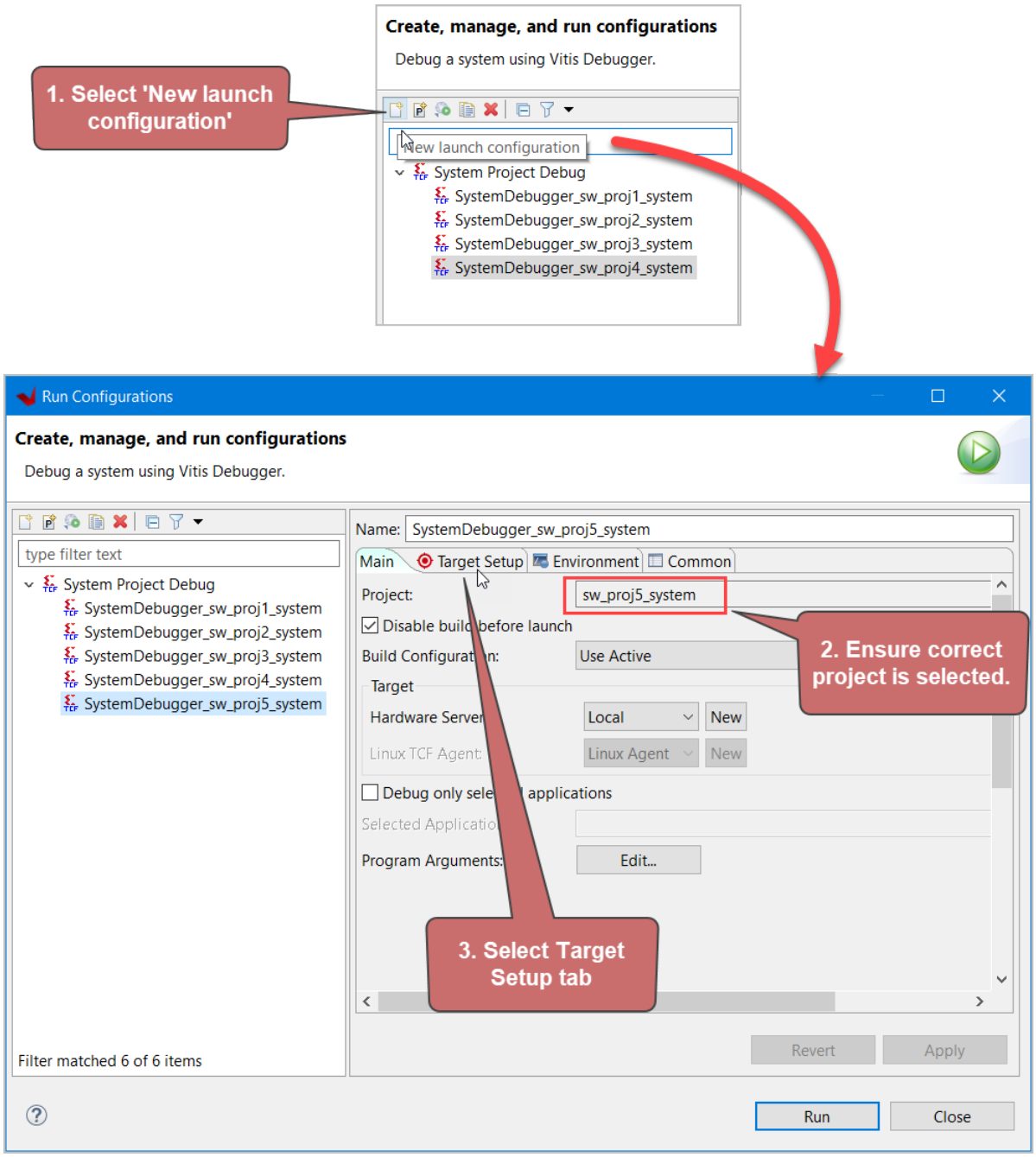


Figure 184. Create a New launch configuration

1. Click on 'New launch configuration'.
2. Ensure that "sw_proj5_system" is selected.
3. Select the Target Setup Tab.

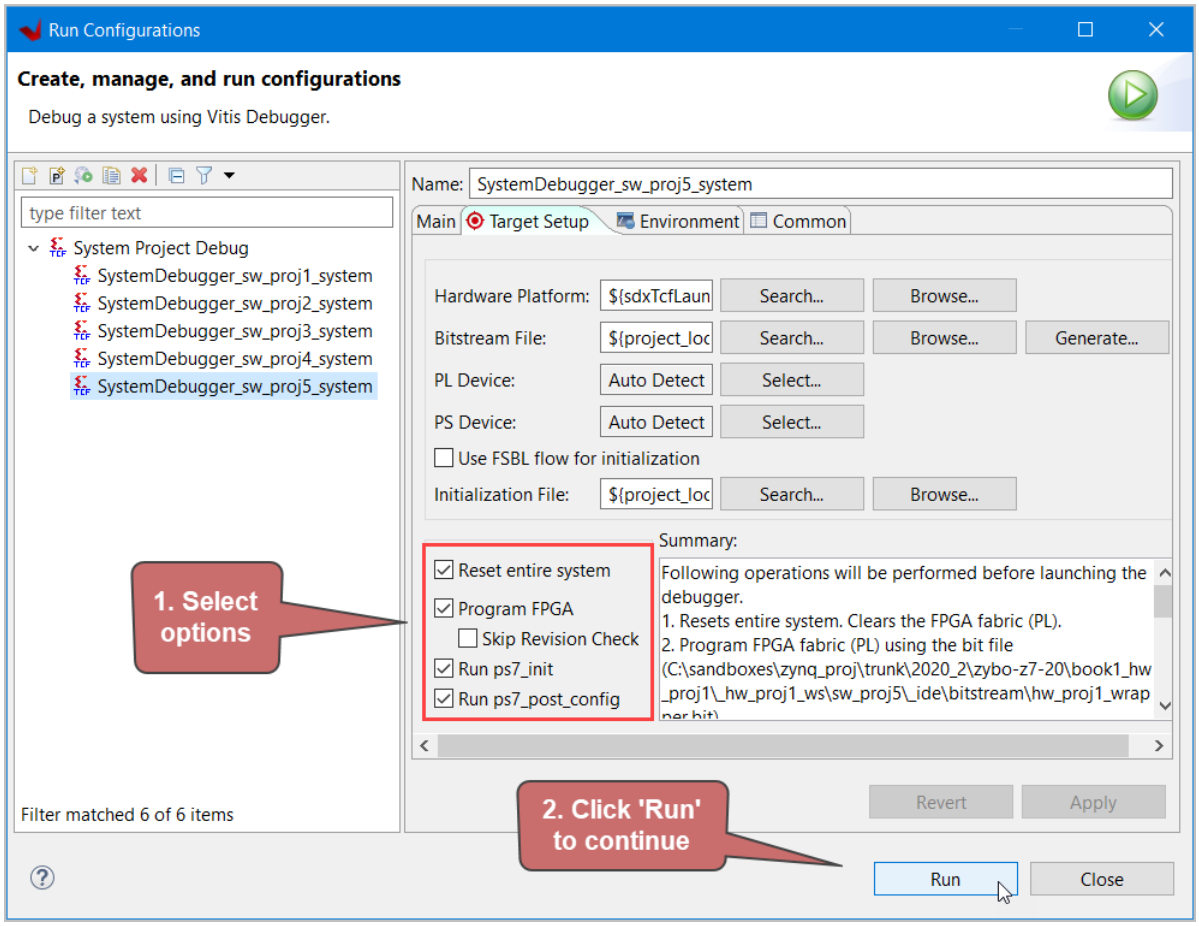


Figure 185. Run the application.

1. Select the main options as follows:
    a. Reset entire system
    b. Program FPGA
    c. Run ps7_init
    d. Run ps7_post_config

Click on 'Run' to continue. The FPGA should be programmed, and the ELF file should be downloaded to the platform, as shown in the next page.

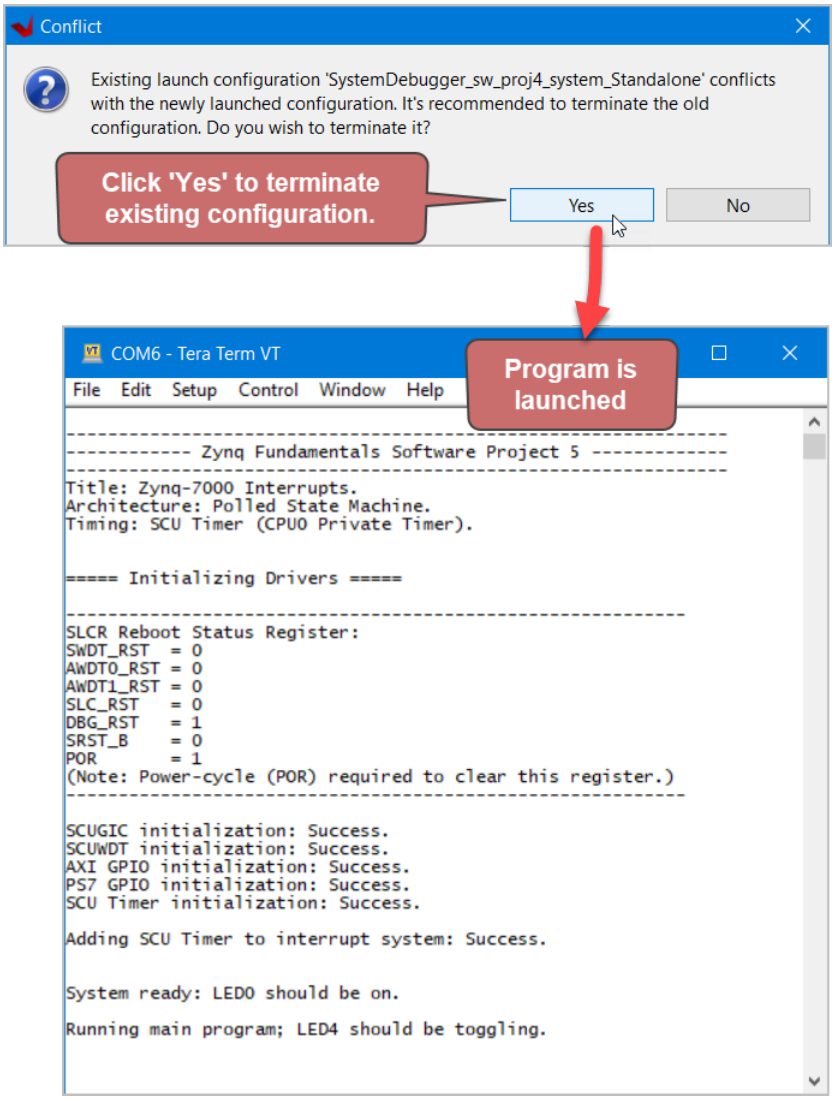


Figure 186. The project details for the software project should appear in the terminal.

If prompted, select 'Yes' to terminate any existing configuration. The FPGA will be programmed and the application will be downloaded to the processor. The terminal program should display the results for launching Software Project 5.

3.9 System (Software) Project 6: TTC Timing

3.9.1 Create the Project

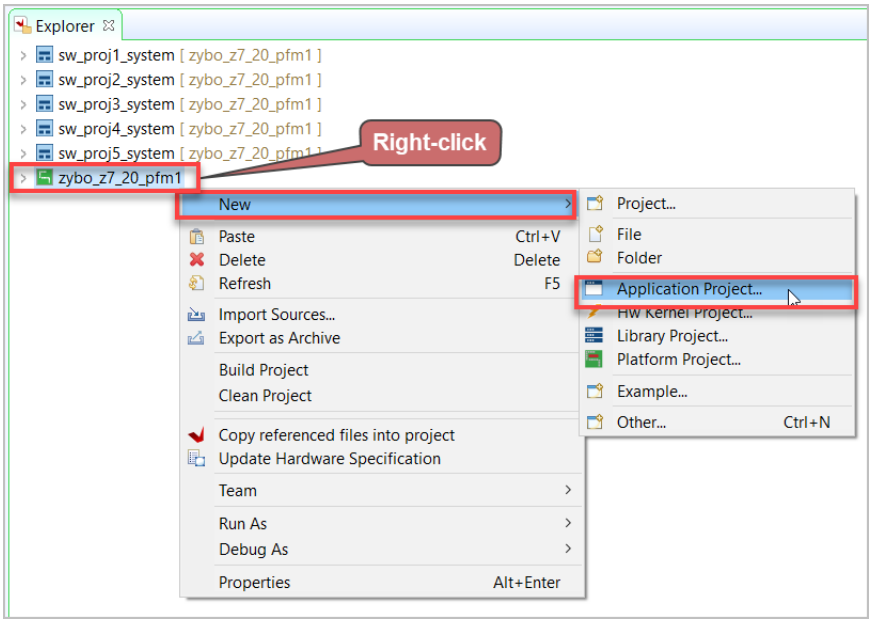


Figure 187. Right-click on the platform and select New-> Application Project.

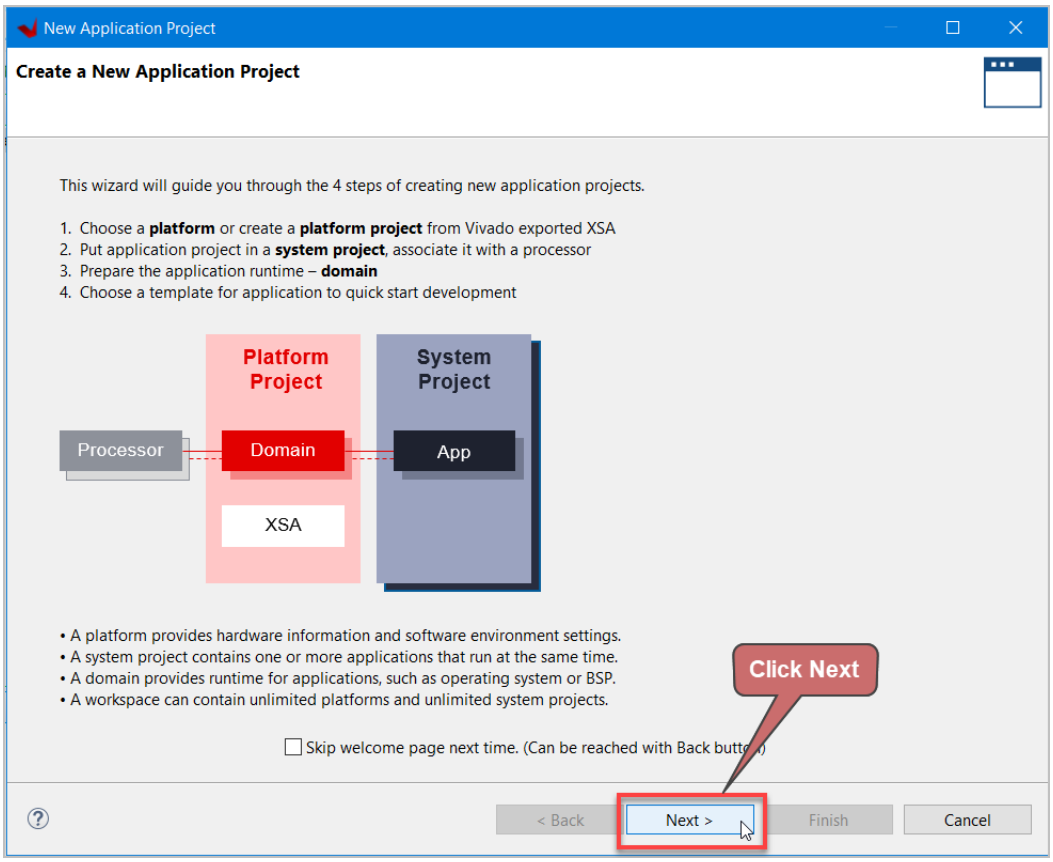


Figure 188. Click Next to continue.

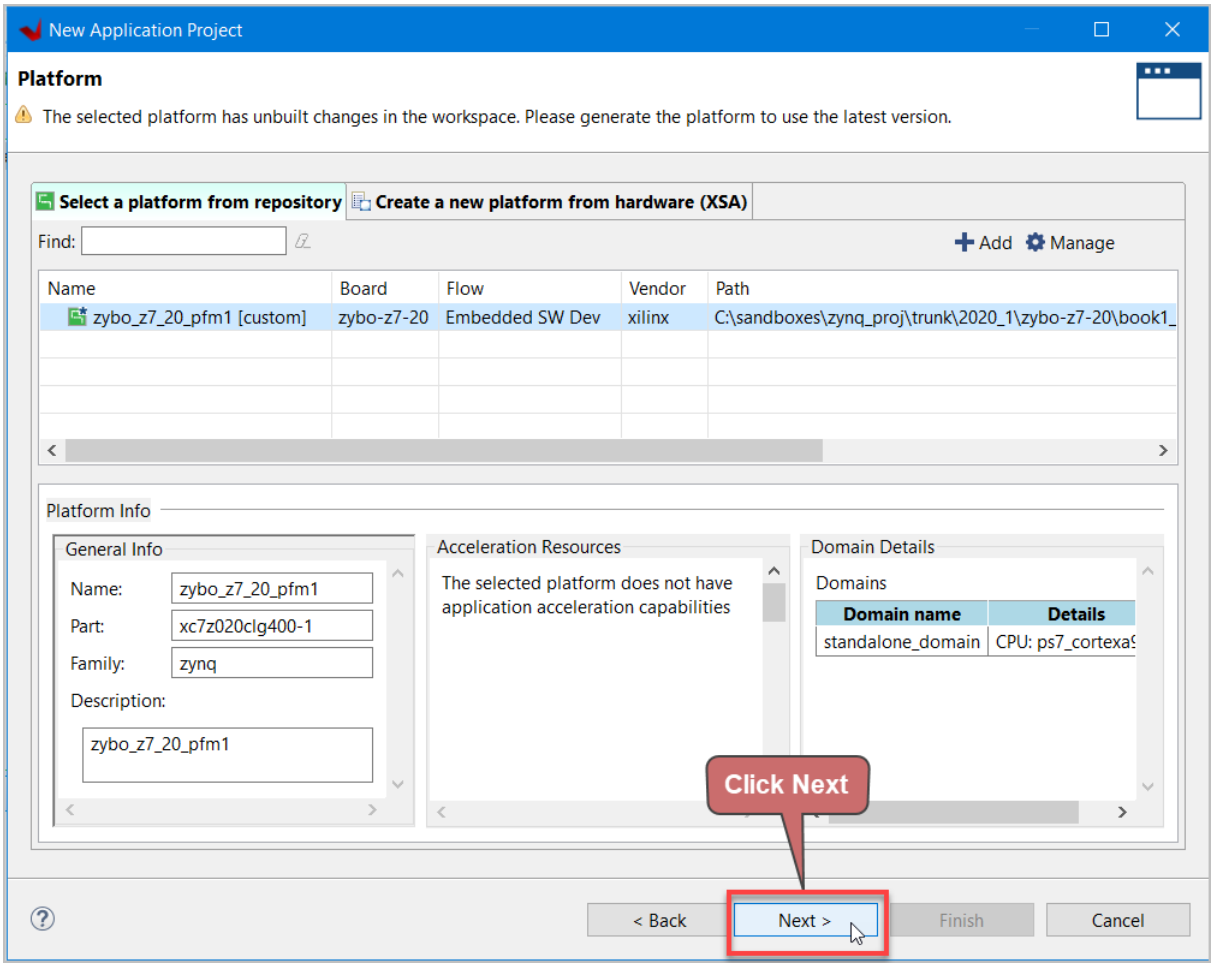


Figure 189. Leave defaults, and click Next to continue.

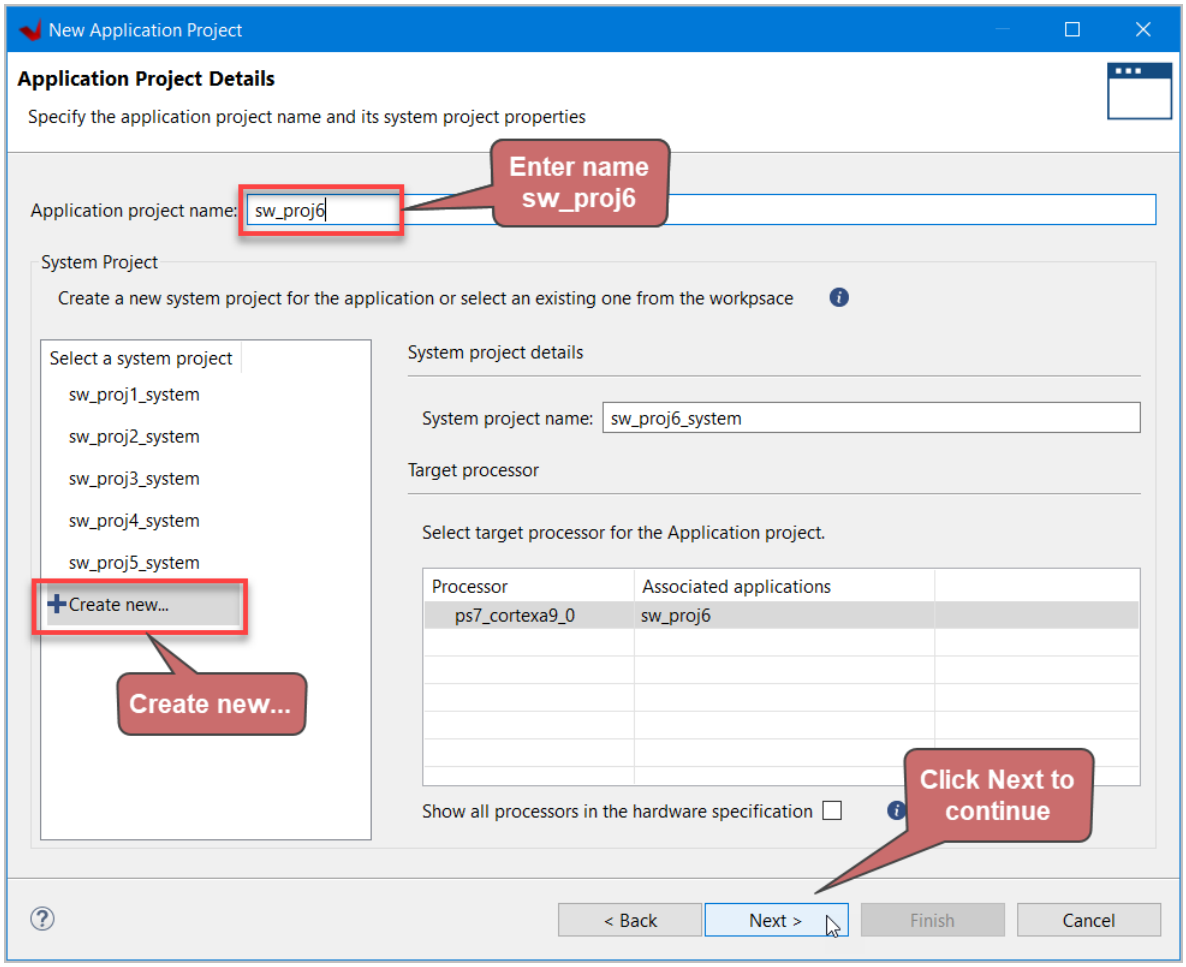


Figure 190. Set the project name to 'sw_proj6', ensure Create New... is selected, and press Next to continue.

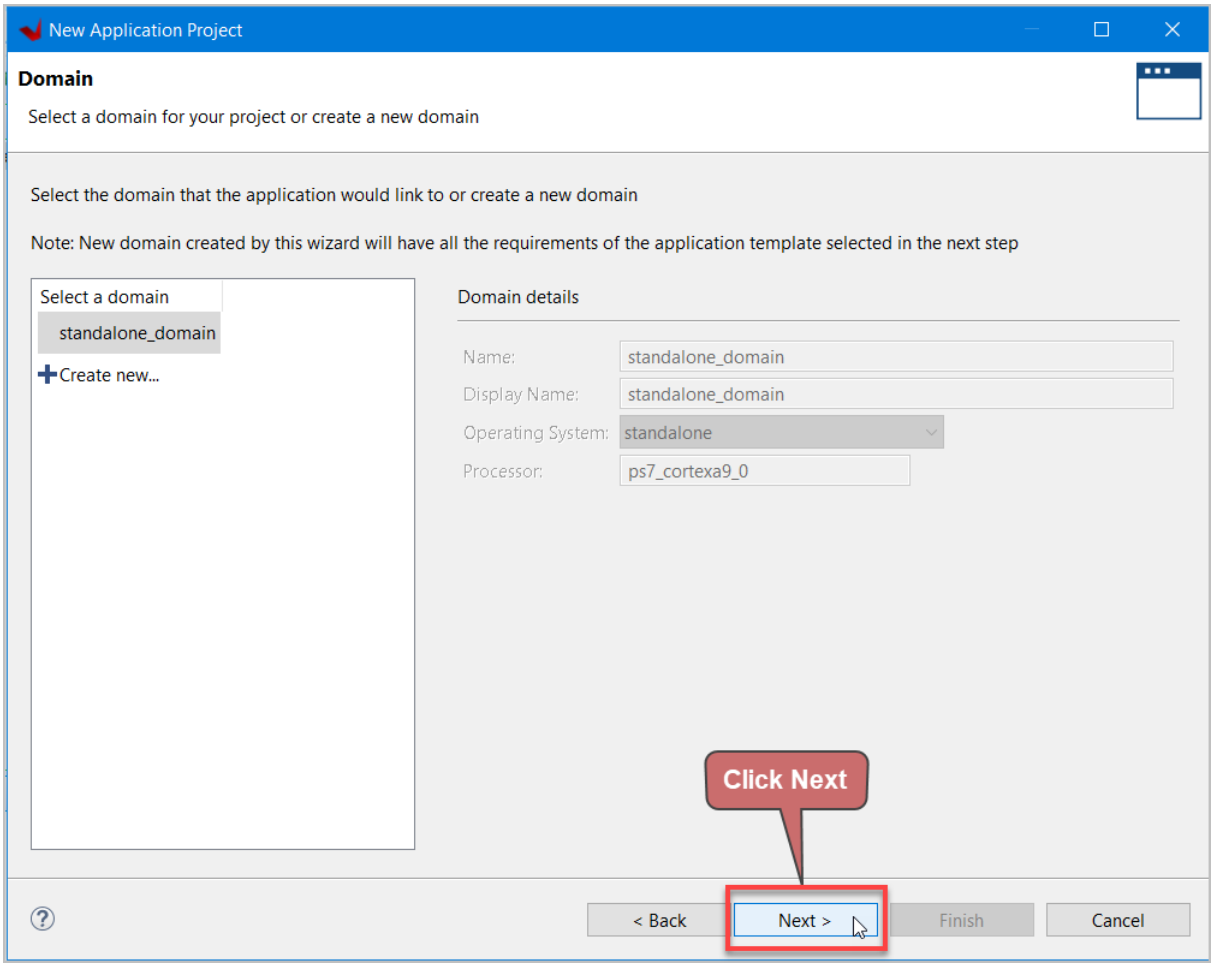


Figure 191. The standalone domain should be selected by default. Click Next to continue.

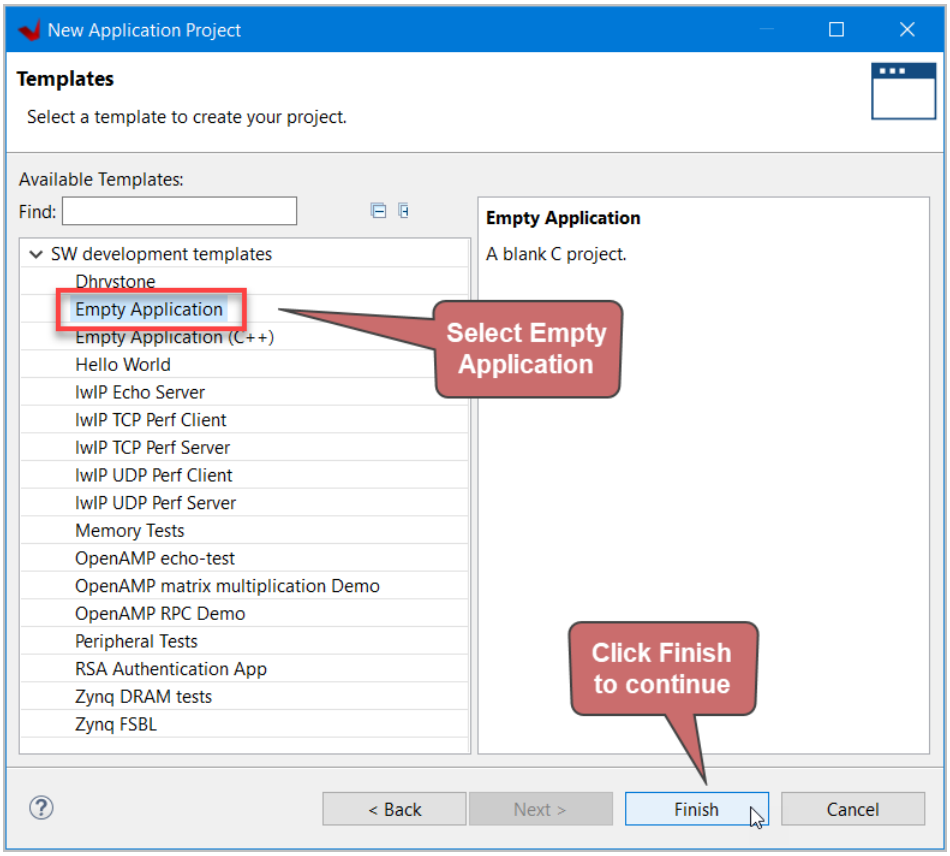


Figure 192. Select Empty Application, and click Finish.

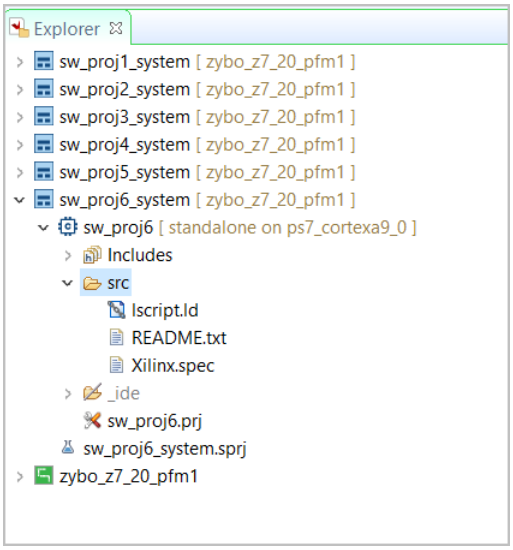


Figure 193. The empty application is created.

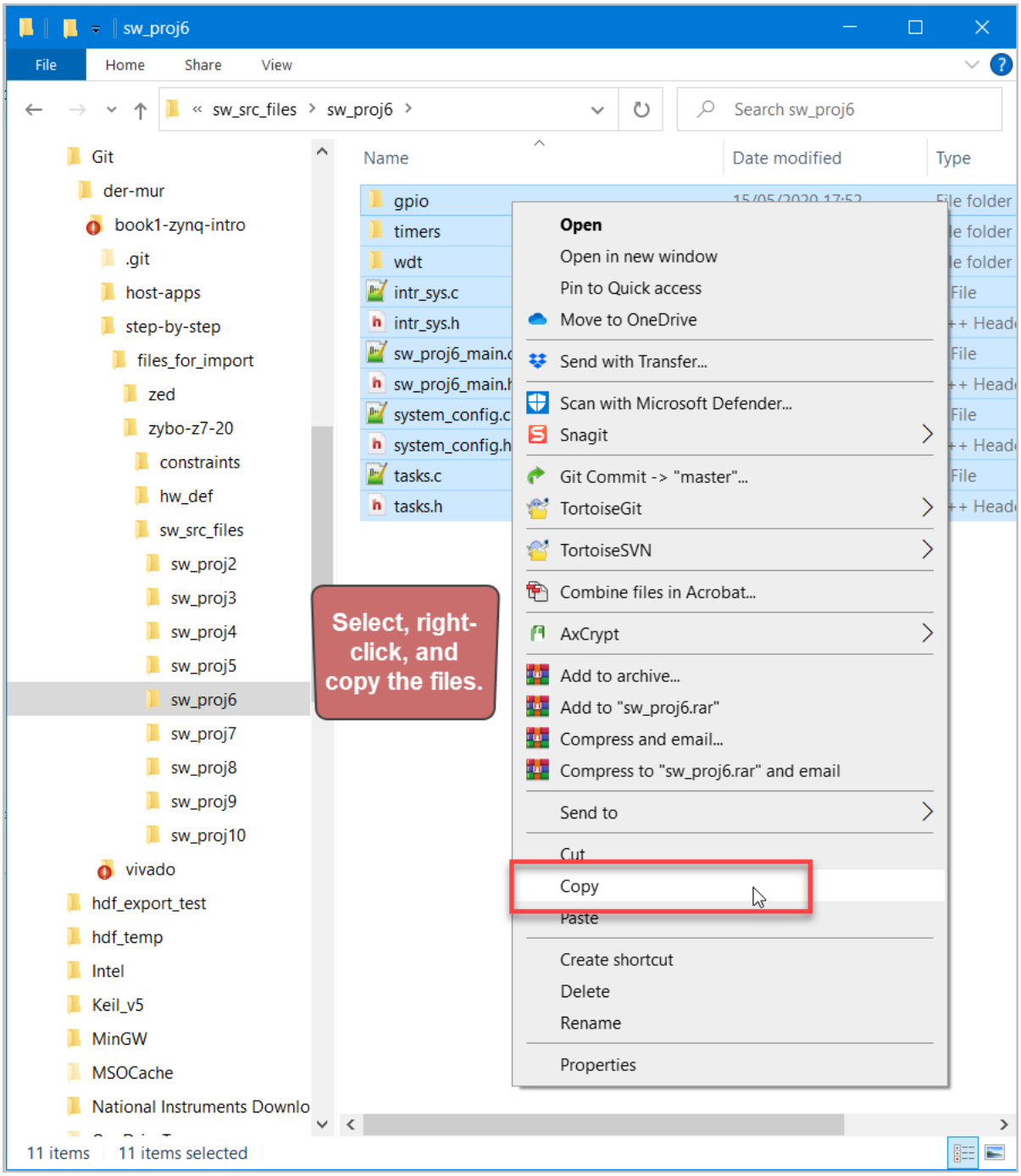


Figure 194. Select the project files, right-click, and select Copy.

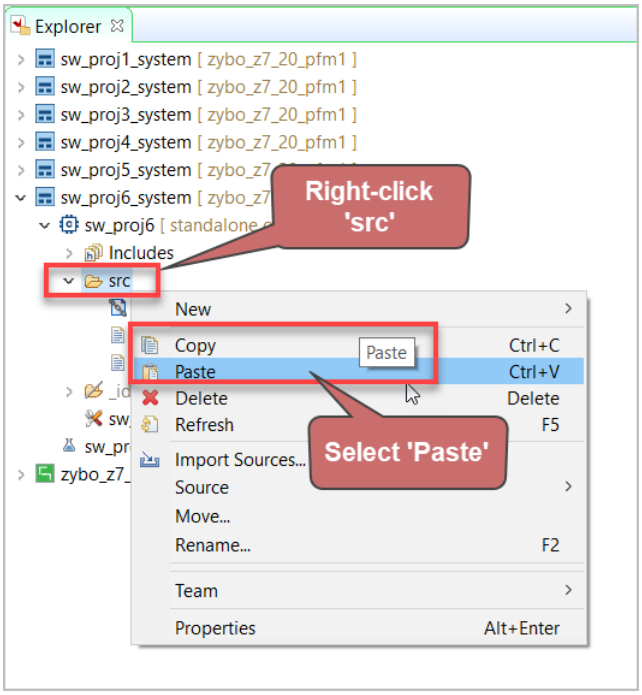


Figure 195. In Vitis, right-click the 'src' directory, and paste the files.

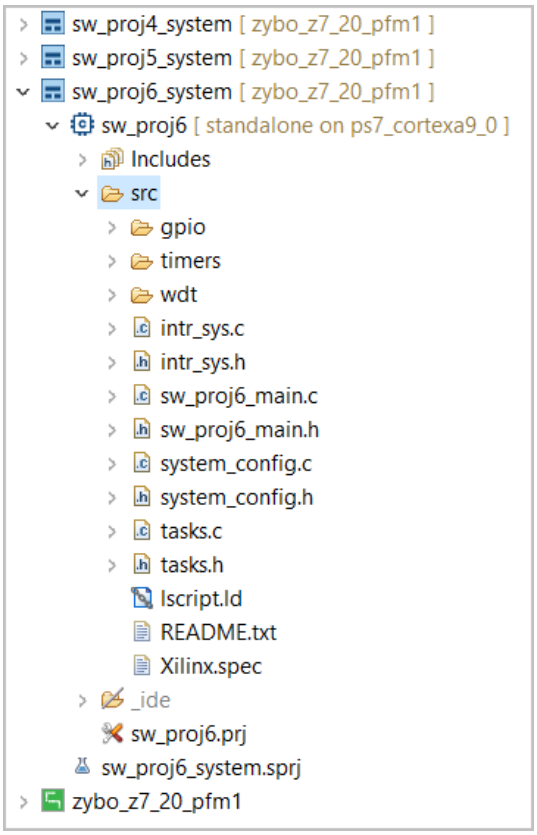


Figure 196. The project files are added.

3.9.2 Build the Project

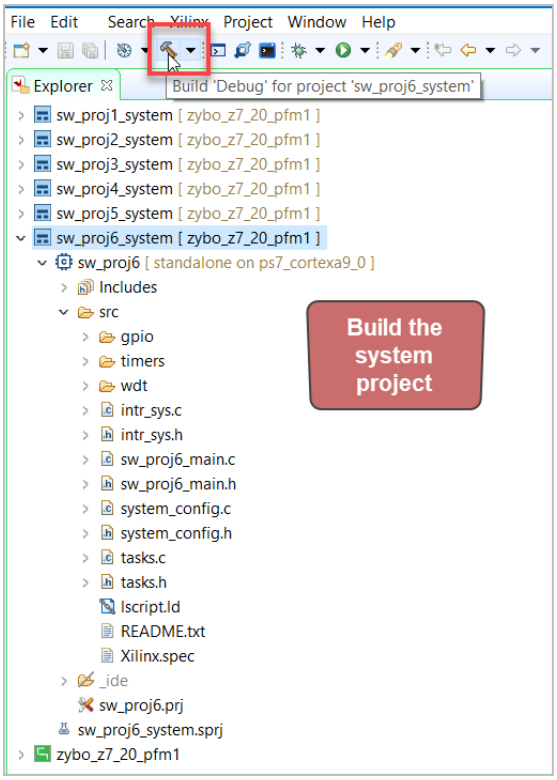


Figure 197. Build the system project

3.9.3 Run the Project

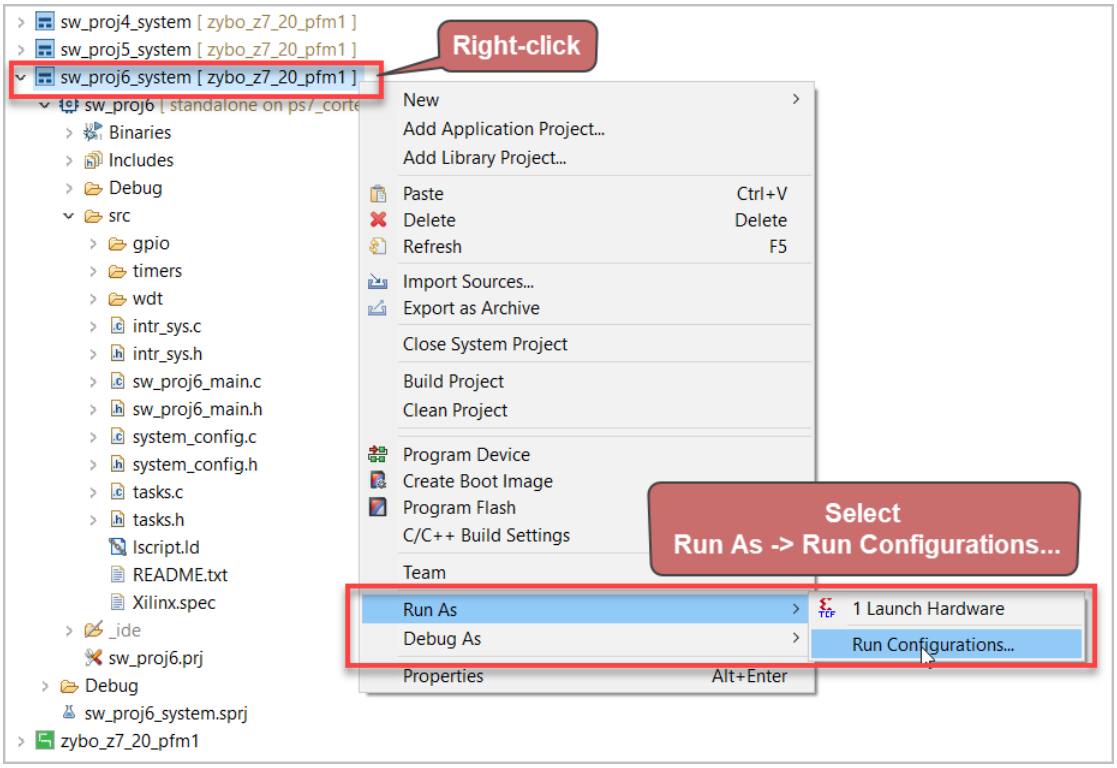


Figure 198. Right-click on the system project and select Run As -> Run Configurations.

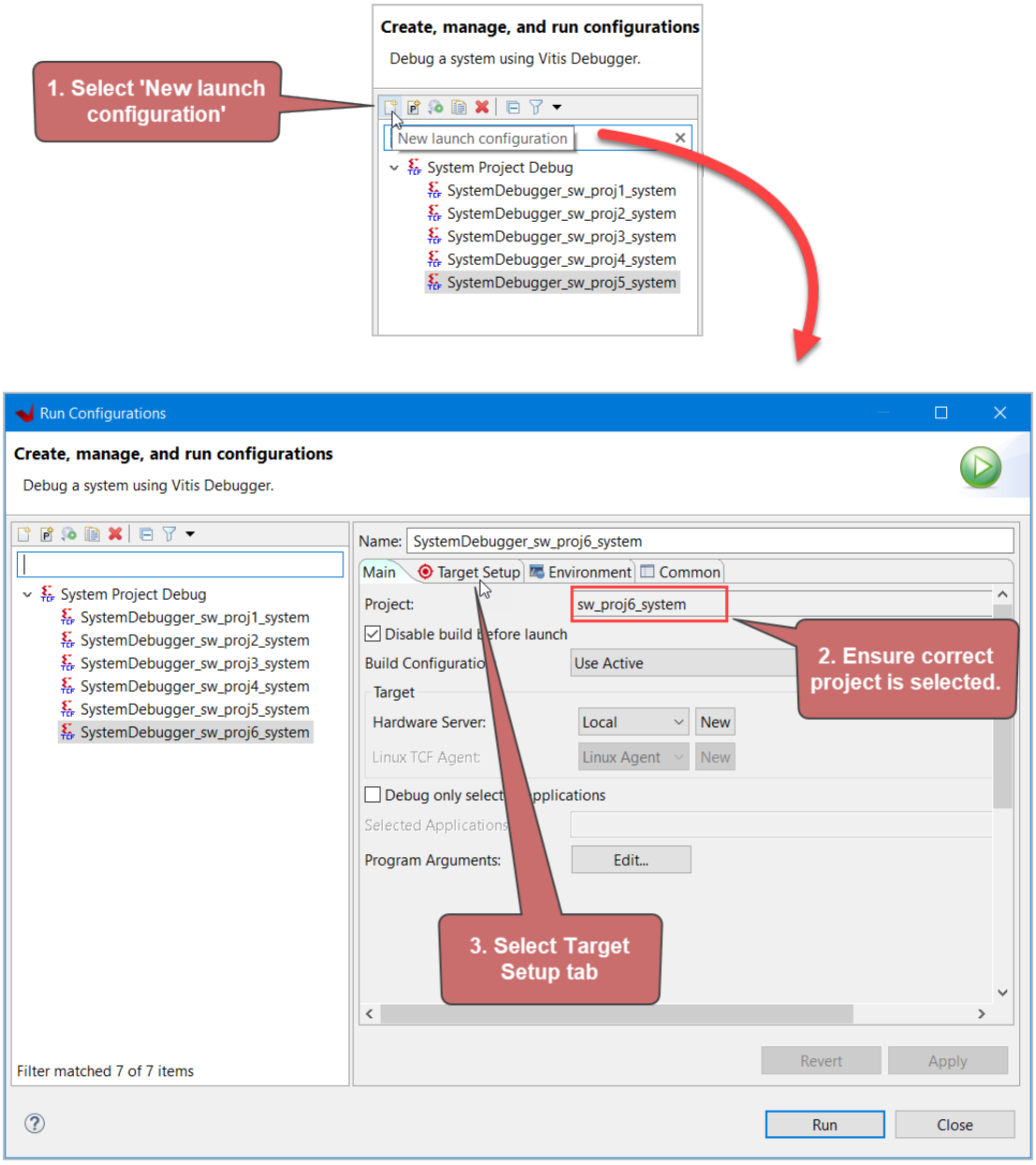


Figure 199. Create a New launch configuration

1. Click on 'New launch configuration'.
2. Ensure that "sw_proj6_system" is selected.
3. Select the Target Setup Tab.

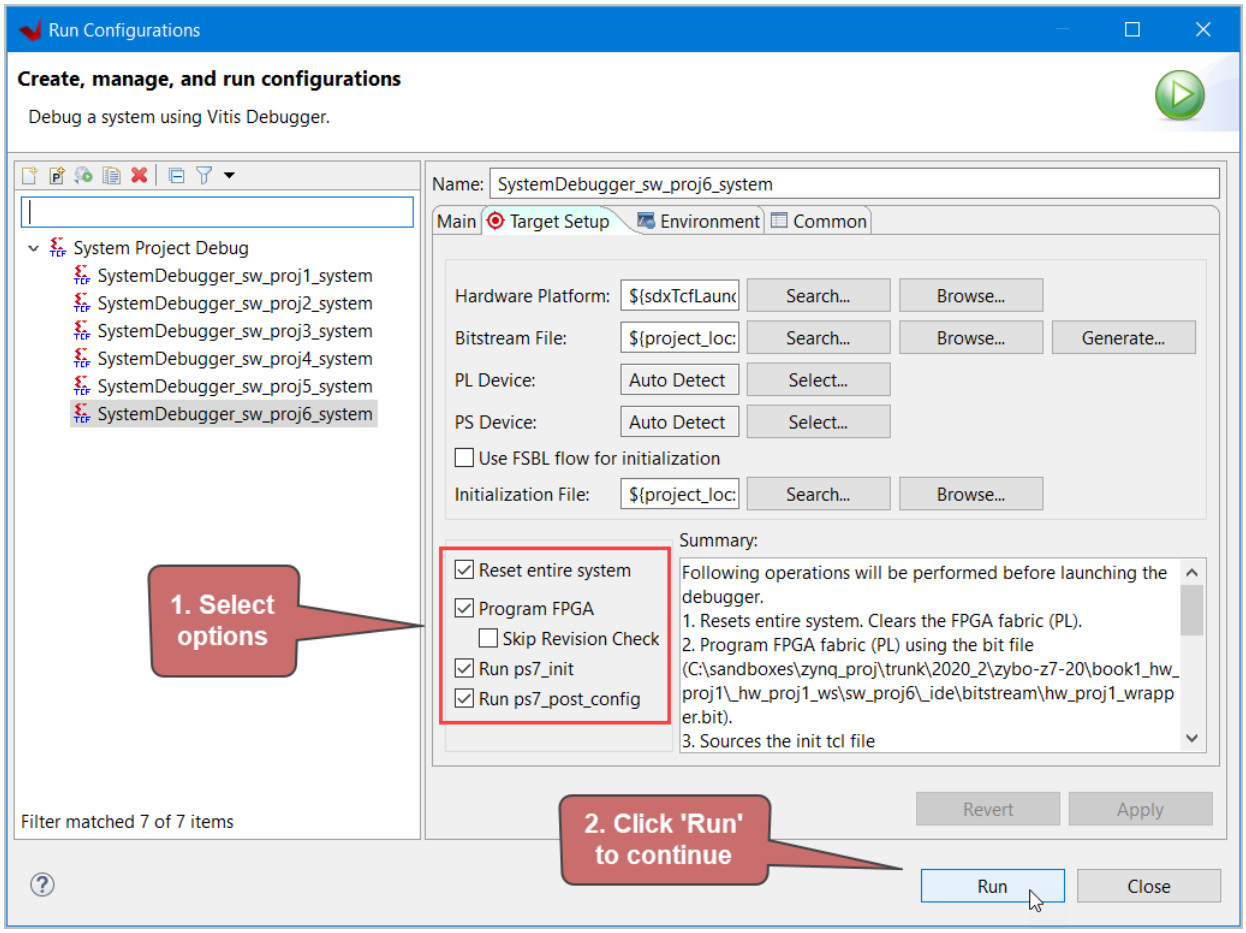


Figure 200. Run the application.

1. Select the main options as follows:
    a. Reset entire system
    b. Program FPGA
    c. Run ps7_init
    d. Run ps7_post_config

Click on 'Run' to continue. The FPGA should be programmed, and the ELF file should be downloaded to the platform, as shown in the next page.

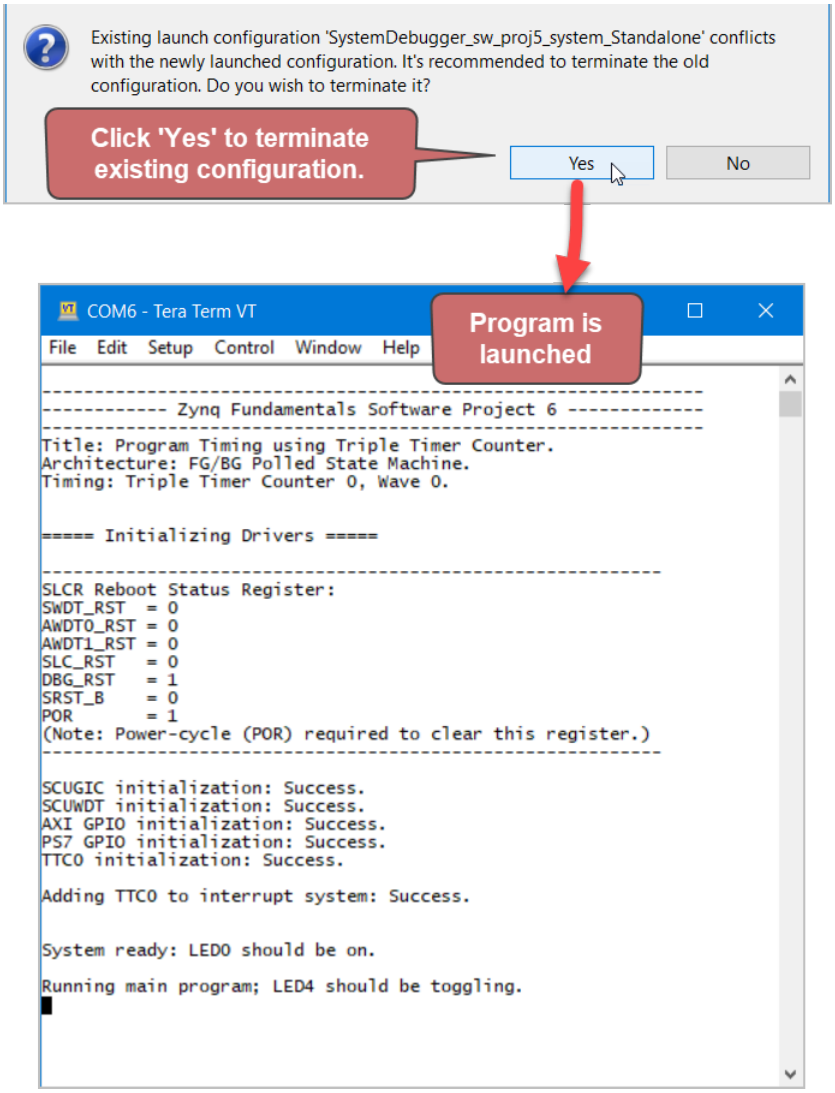


Figure 201. The project details for the software project should appear in the terminal.

If prompted, select 'Yes' to terminate any existing configuration. The FPGA will be programmed and the application will be downloaded to the processor. The terminal program should display the results for launching Software Project 6.

3.10 System (Software) Project 7: Button De-bouncing

3.10.1 Create the Project

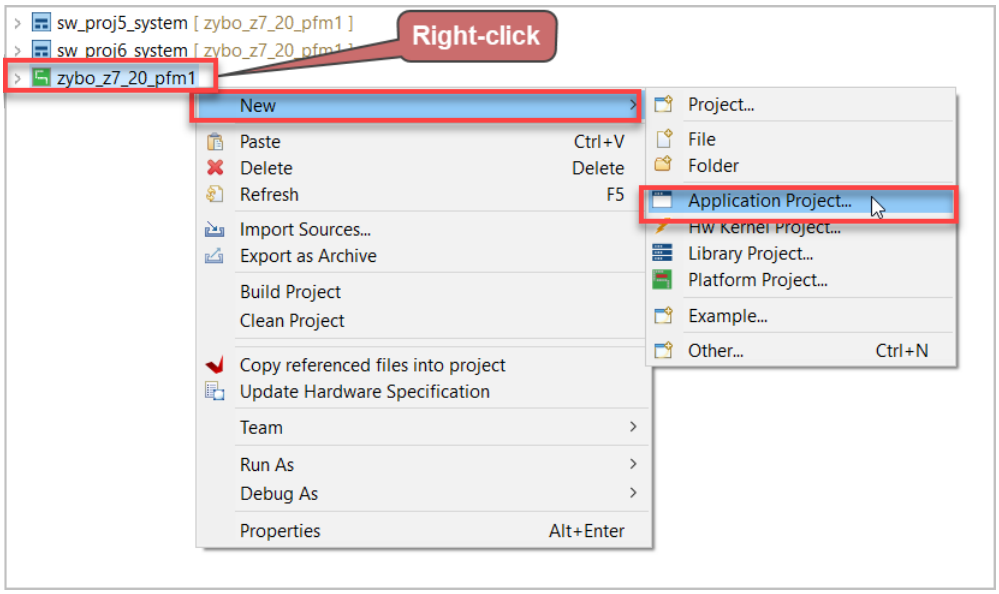


Figure 202. Right-click on the platform and select New-> Application Project.

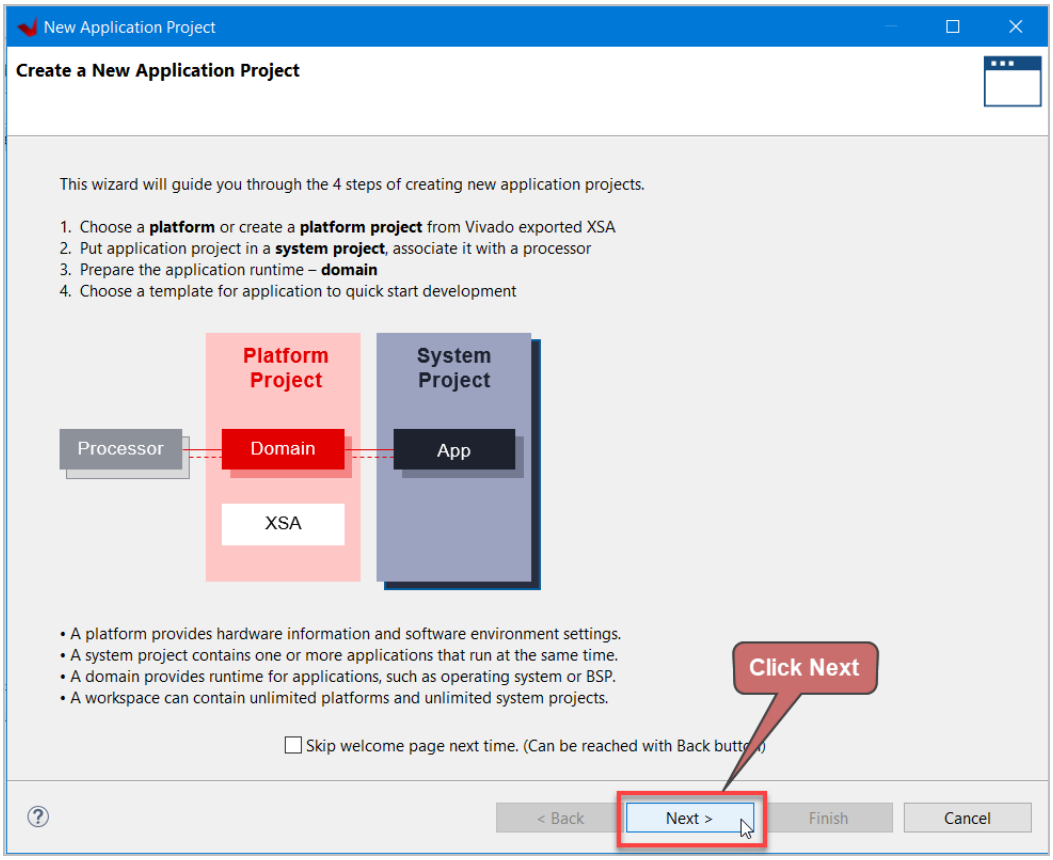


Figure 203. Click Next to continue.

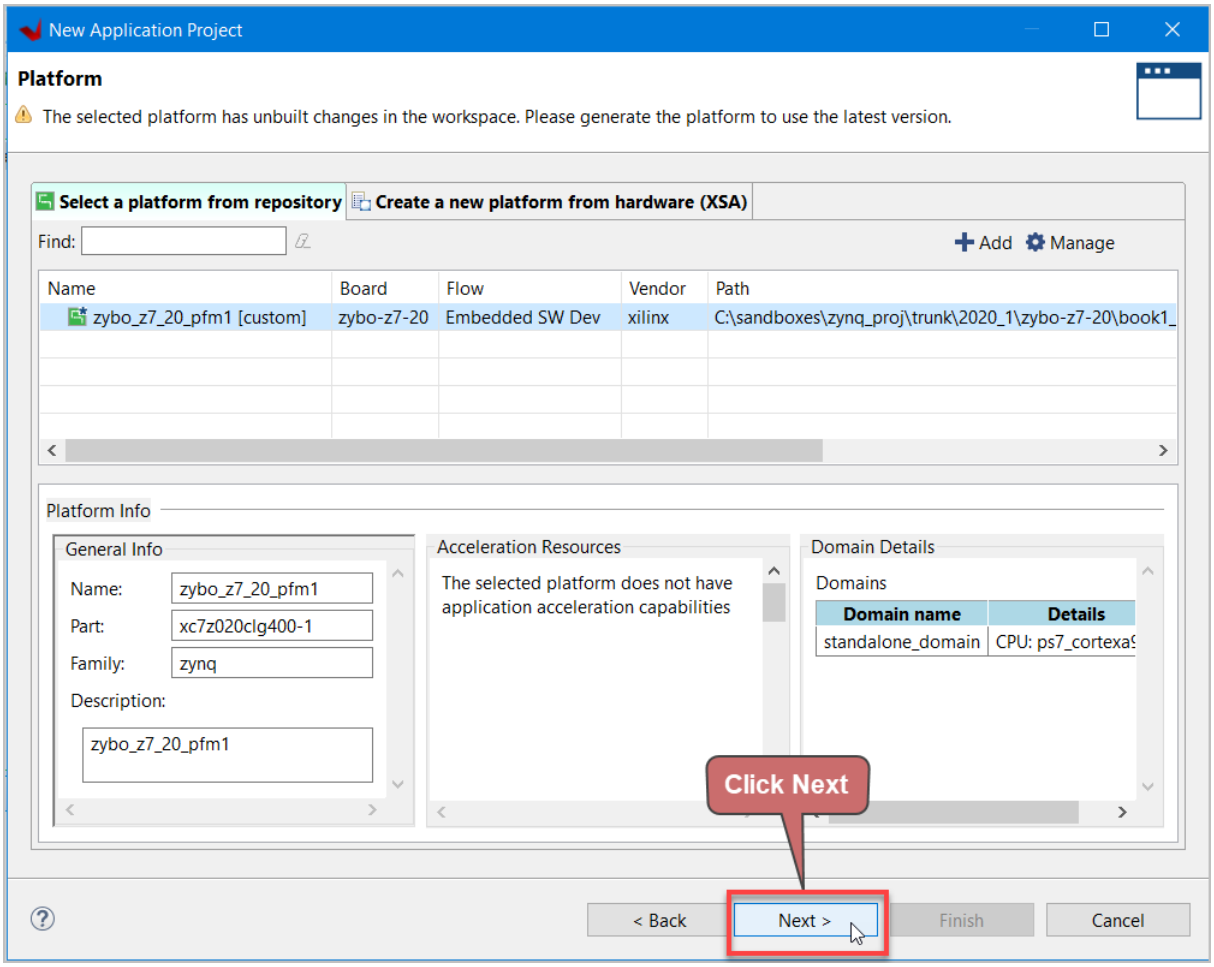


Figure 204. Leave defaults, and click Next to continue.

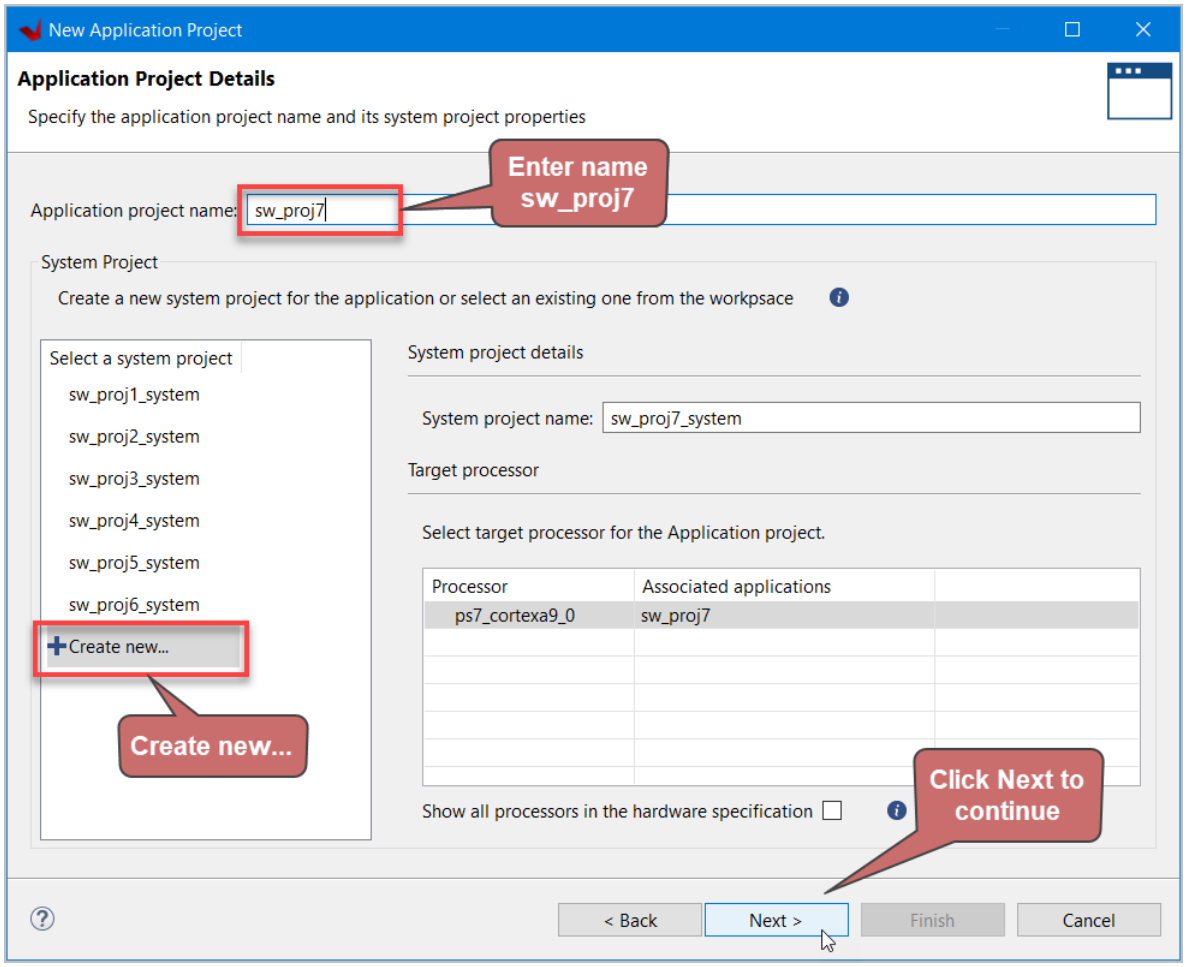


Figure 205. Set the project name to 'sw_proj7', ensure Create New... is selected, and press Next to continue.

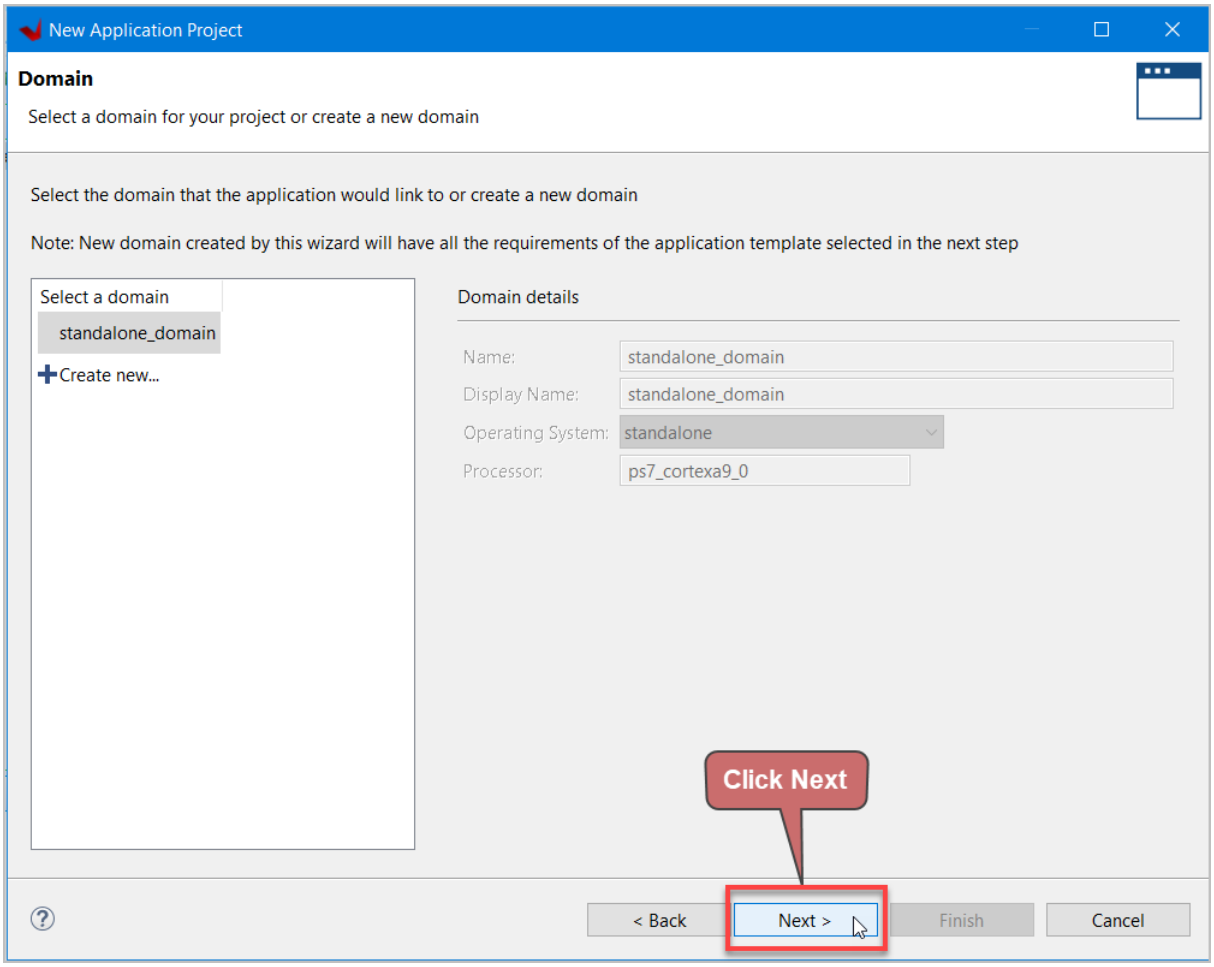


Figure 206. The standalone domain should be selected by default. Click Next to continue.

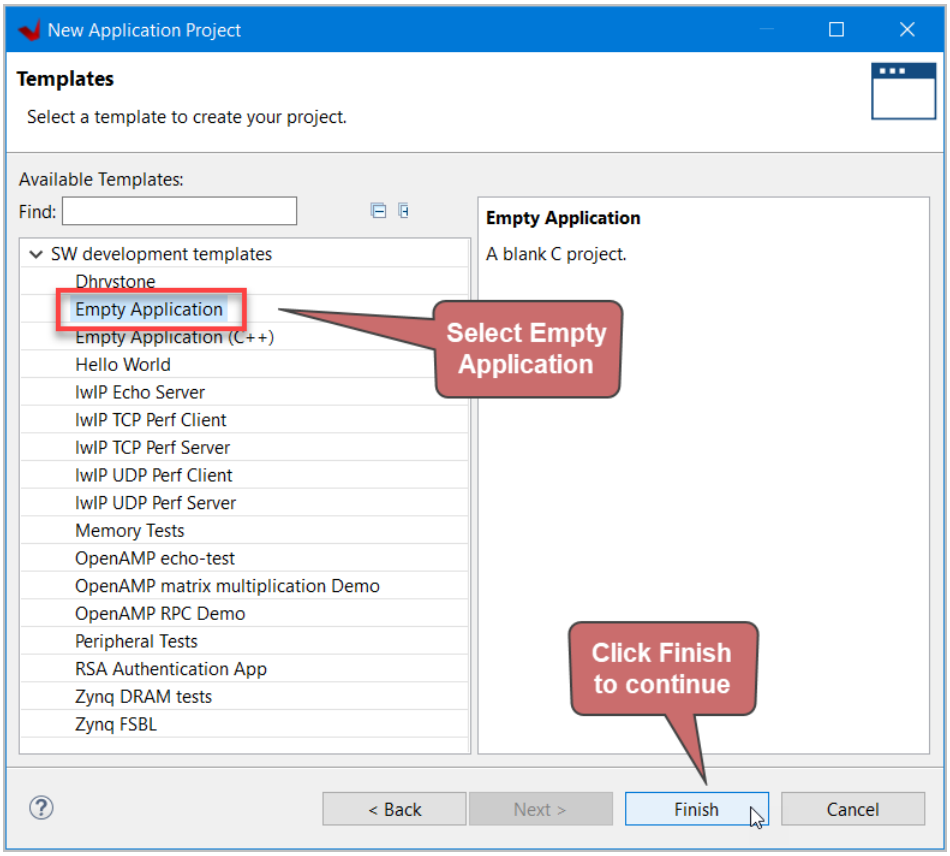


Figure 207. Select Empty Application, and click Finish.

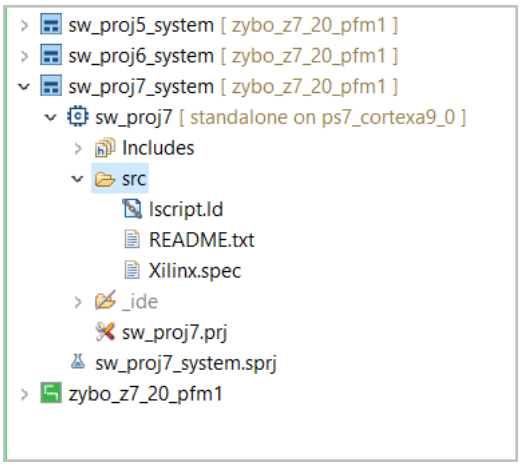


Figure 208. The empty application is created.

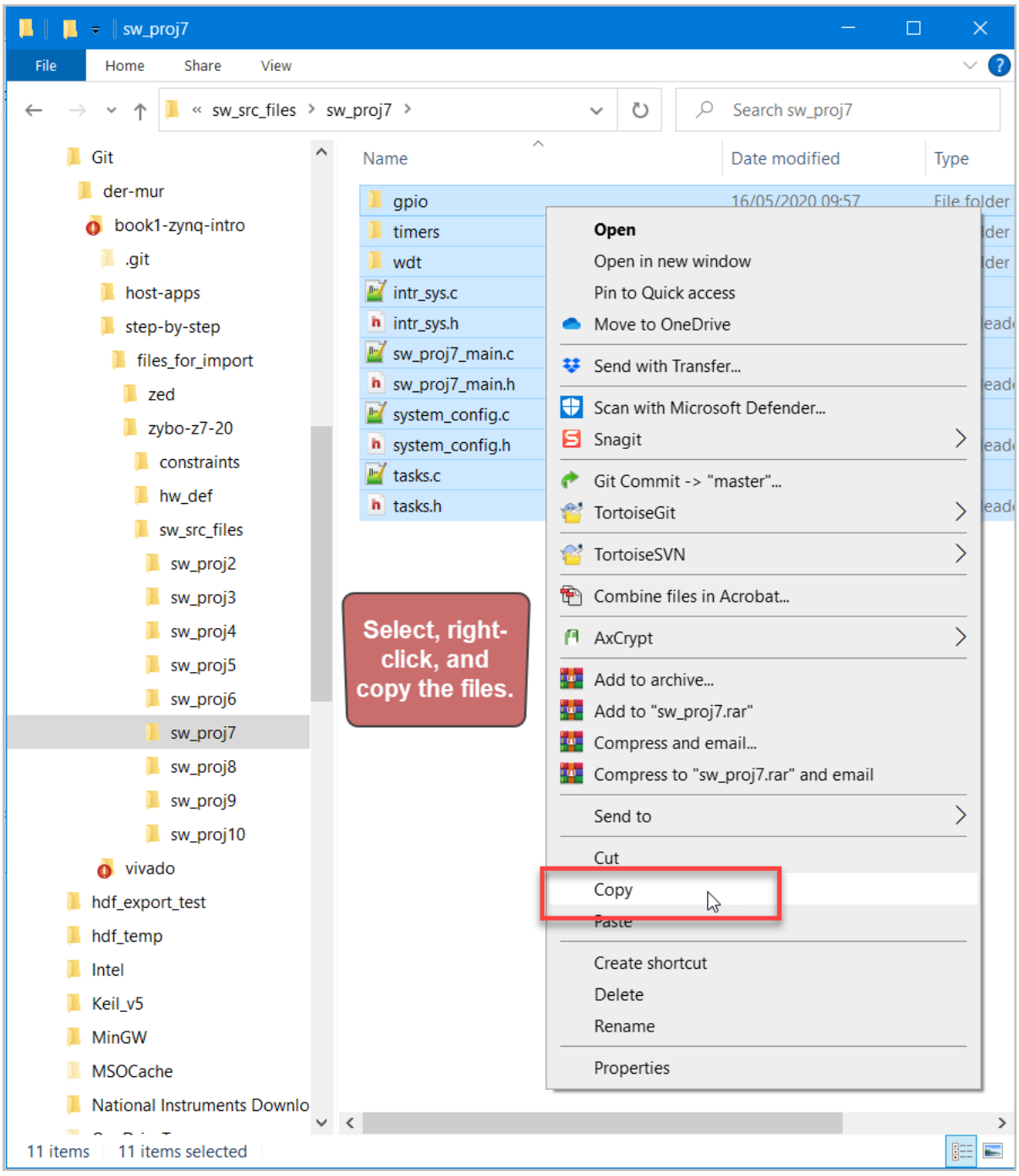


Figure 209. Select the project files, right-click, and select Copy.

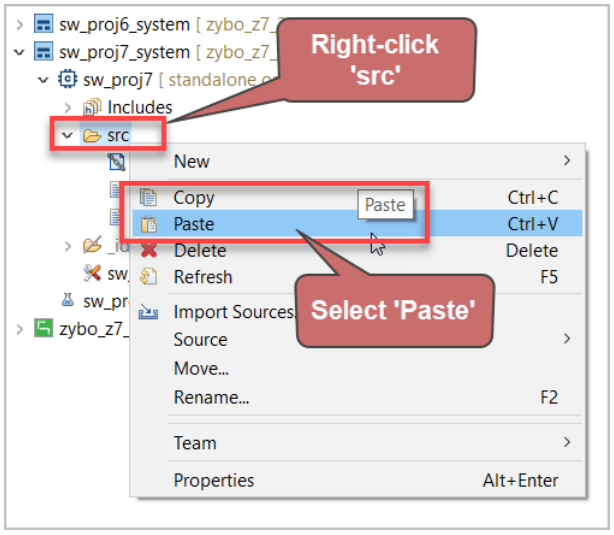


Figure 210. In Vitis, right-click the 'src' directory, and paste the files.

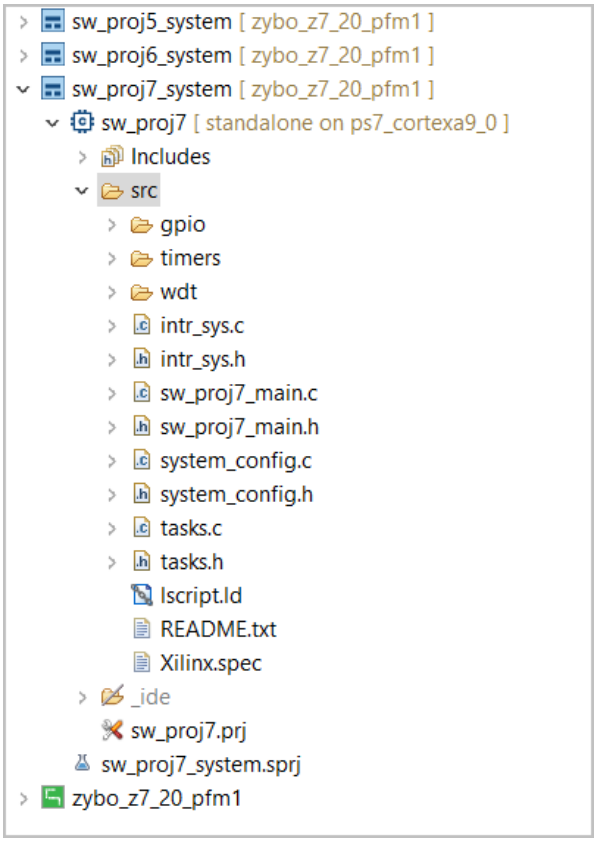


Figure 211. The project files are added.

3.10.2 Build the Project

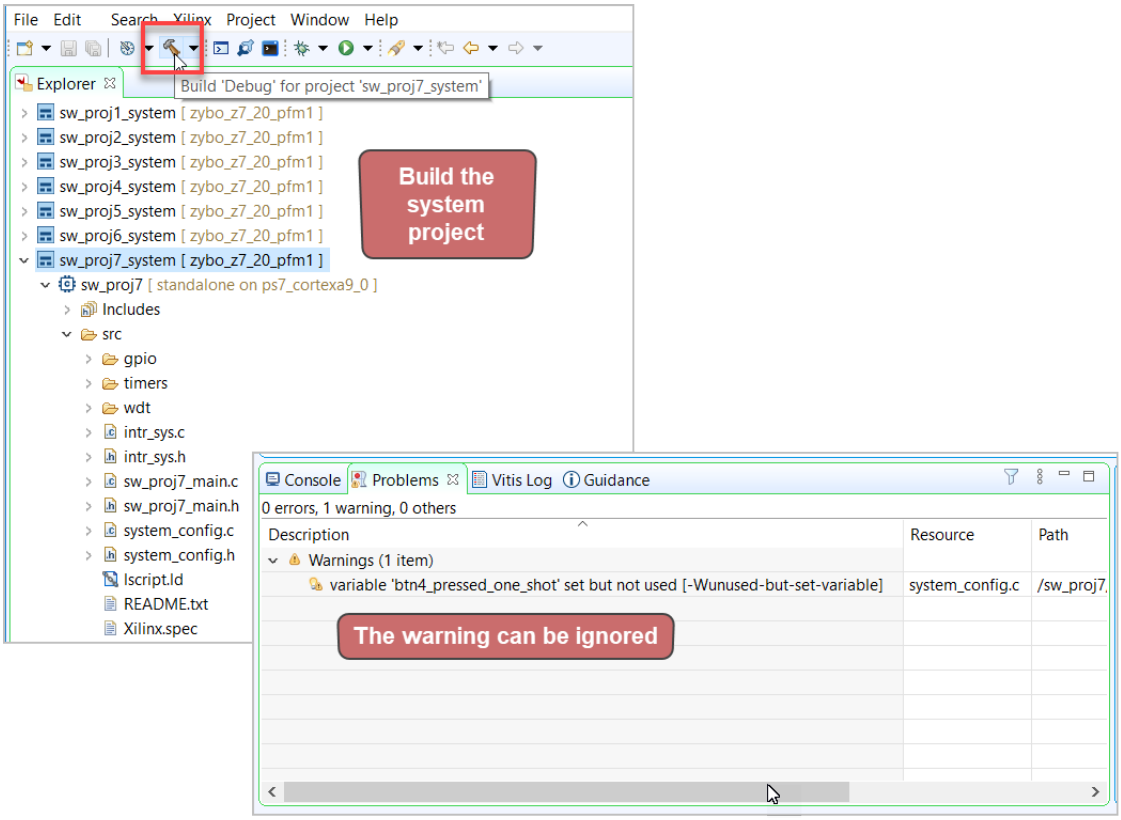


Figure 212. Build the system project

3.10.3 Run the Project

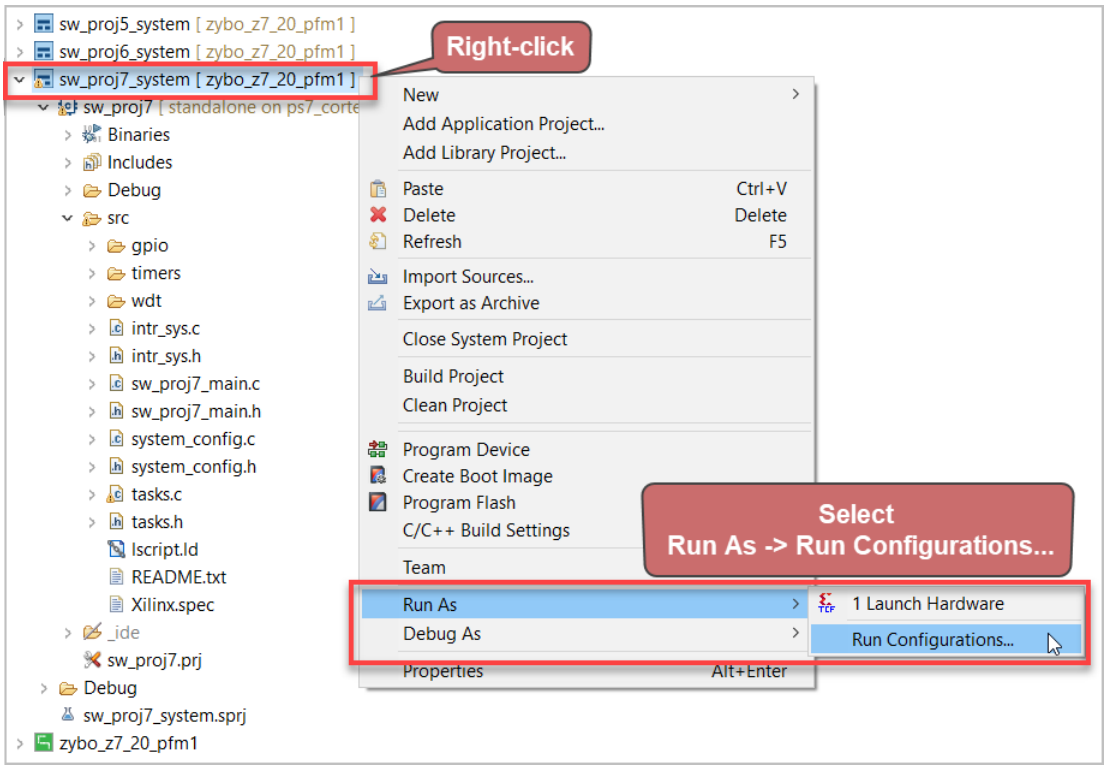


Figure 213. Right-click on the system project and select Run As -> Run Configurations.

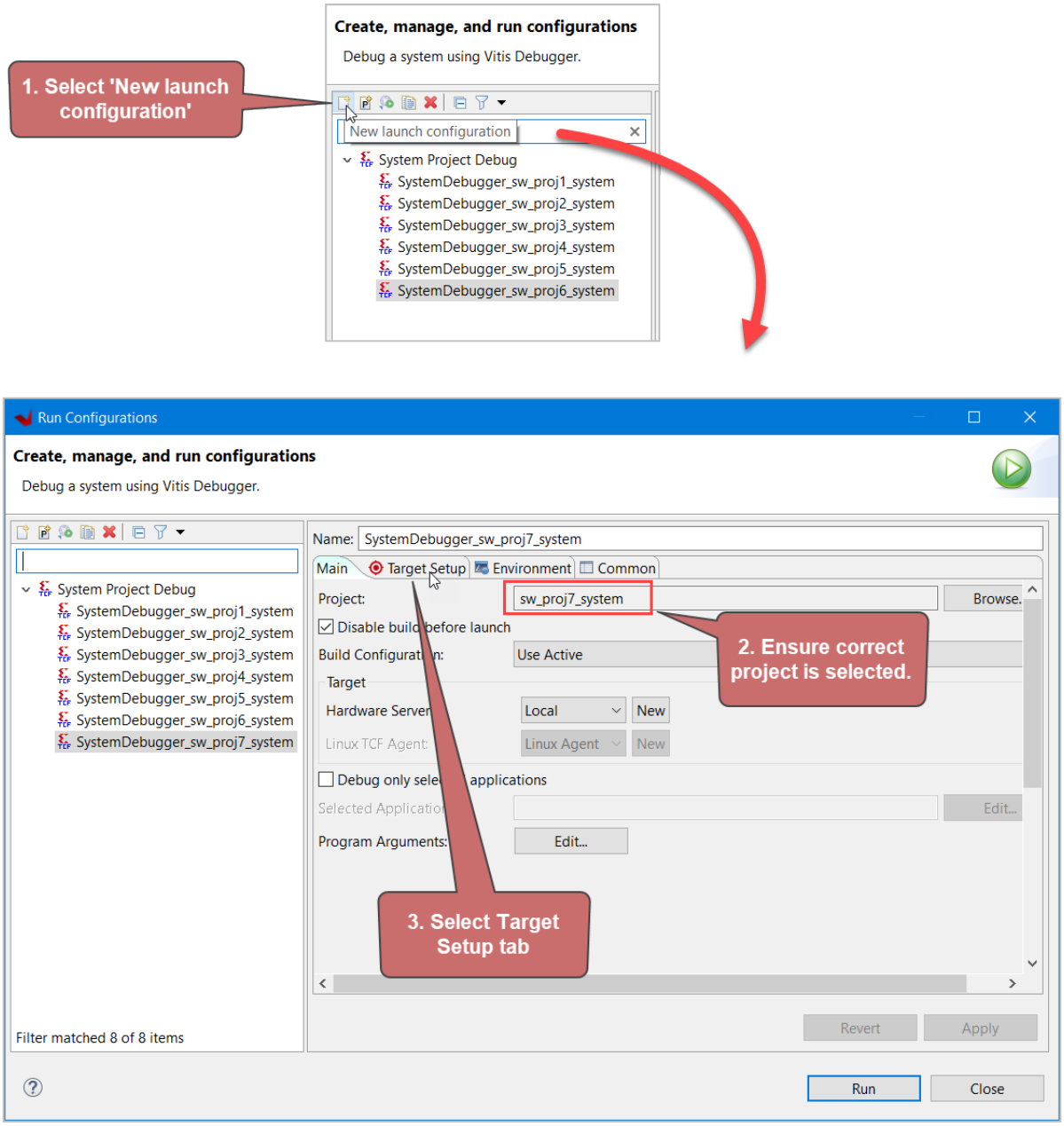


Figure 214. Create a New launch configuration

1. Click on 'New launch configuration'.
2. Ensure that "sw_proj7_system" is selected.
3. Select the Target Setup Tab.

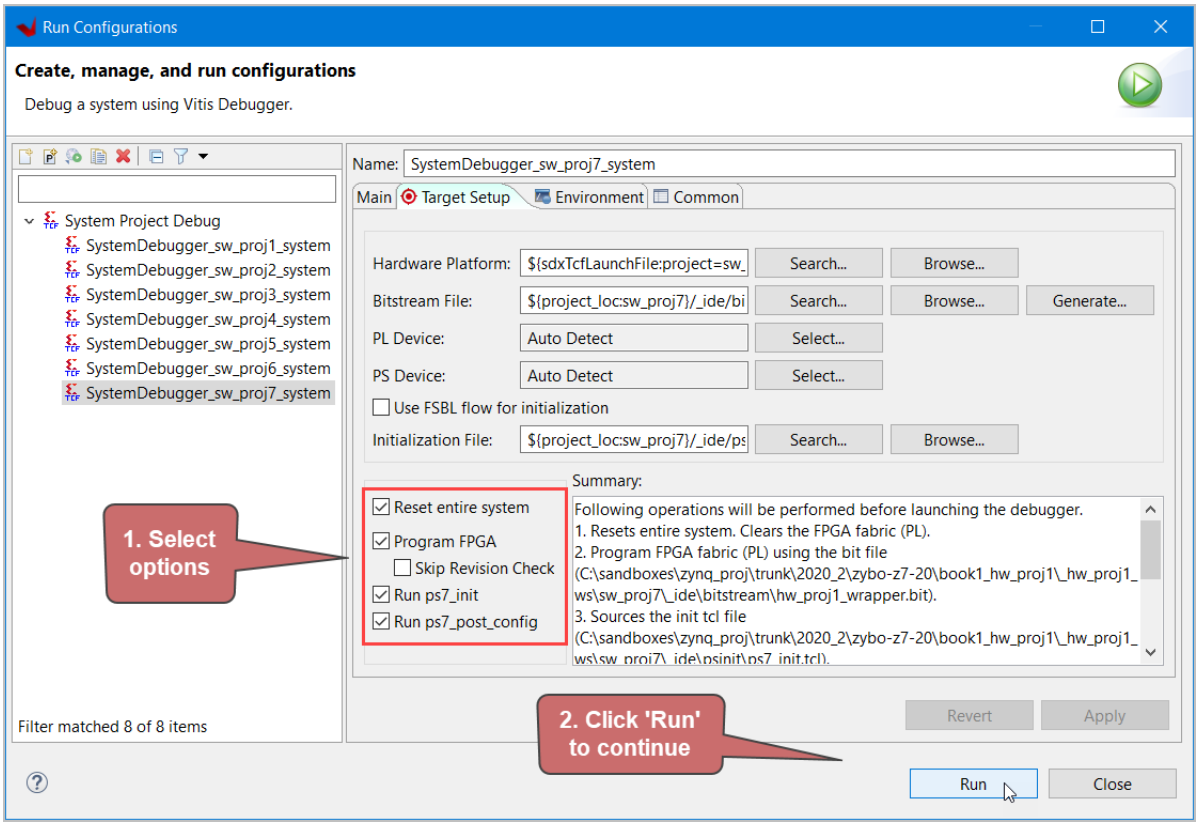


Figure 215. Run the application.

1. Select the main options as follows:
    a. Reset entire system
    b. Program FPGA
    c. Run ps7_init
    d. Run ps7_post_config

Click on 'Run' to continue. The FPGA should be programmed, and the ELF file should be downloaded to the platform, as shown in the next page.

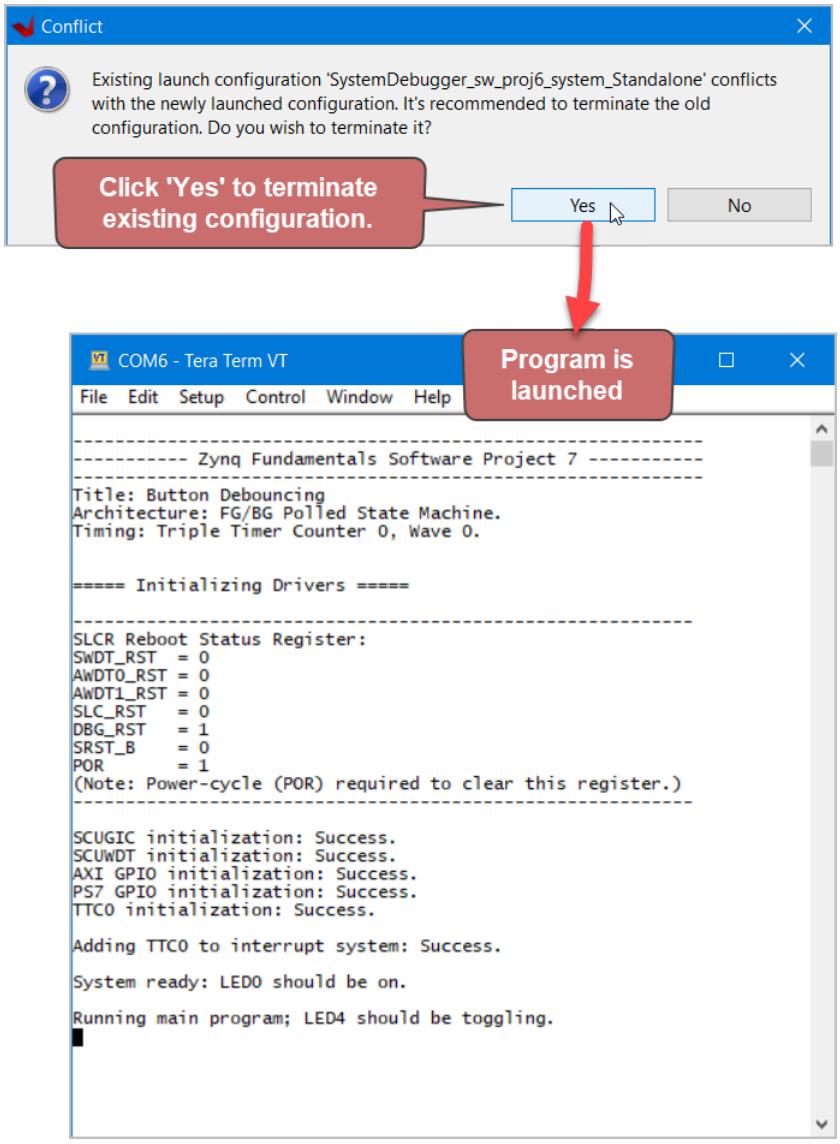


Figure 216. The project details for the software project should appear in the terminal.

If prompted, select 'Yes' to terminate any existing configuration. The FPGA will be programmed and the application will be downloaded to the processor. The terminal program should display the results for launching Software Project 7.

3.11 System (Software) Project 8: UART Command Handler

3.11.1 Create the Project

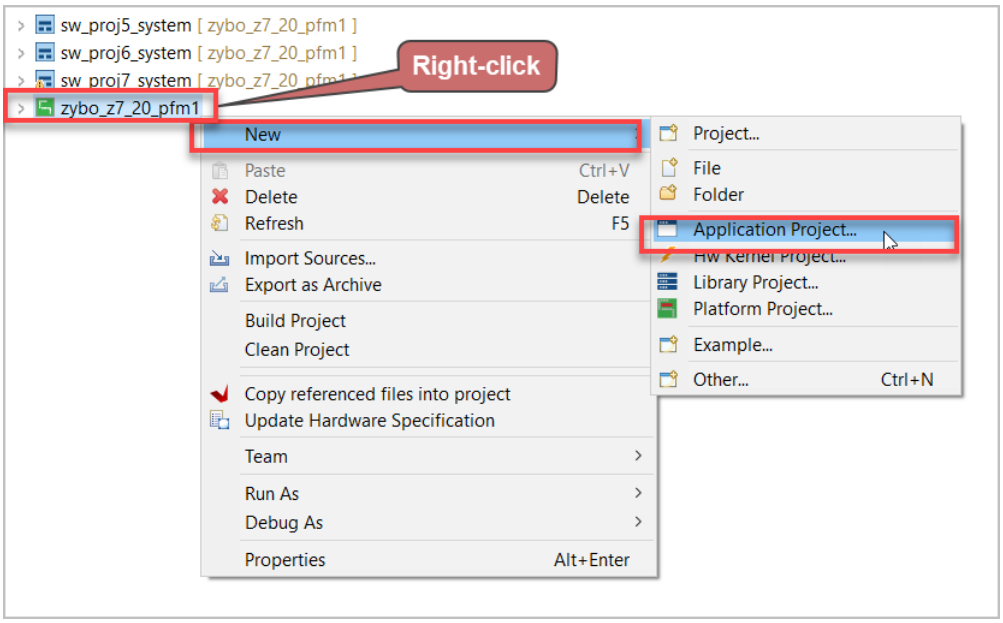


Figure 217. Right-click on the platform and select New-> Application Project.

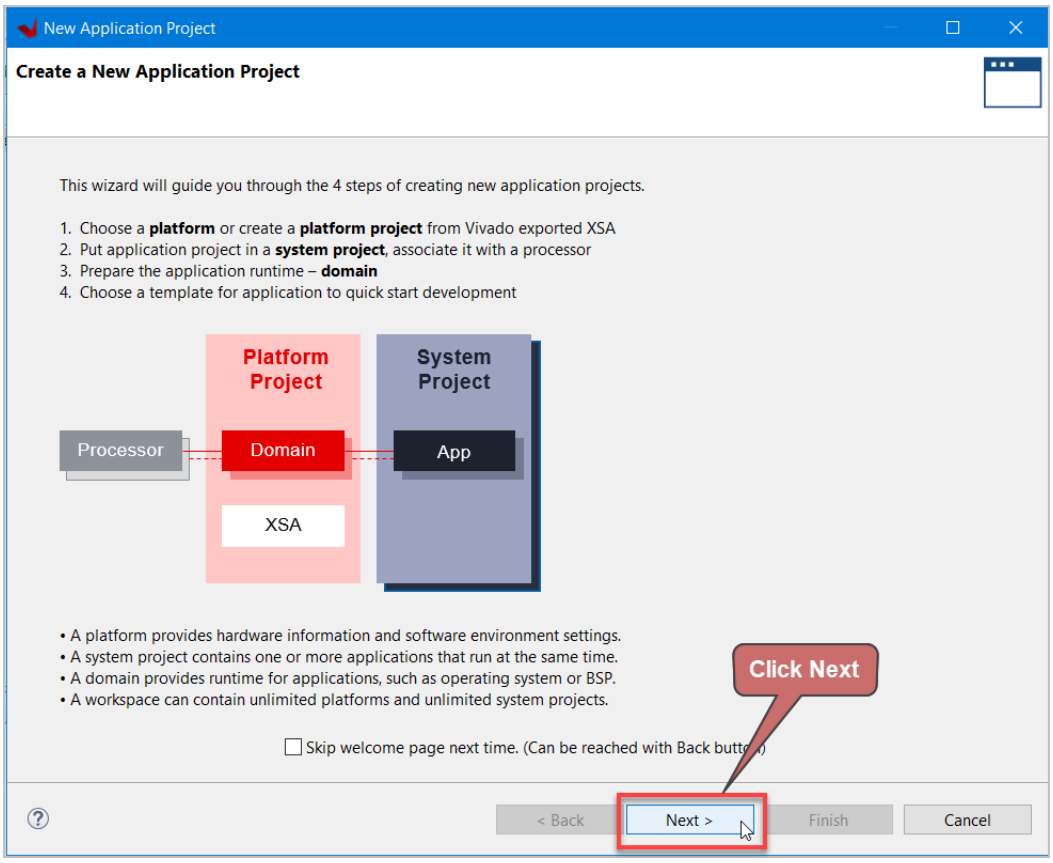


Figure 218. Click Next to continue.

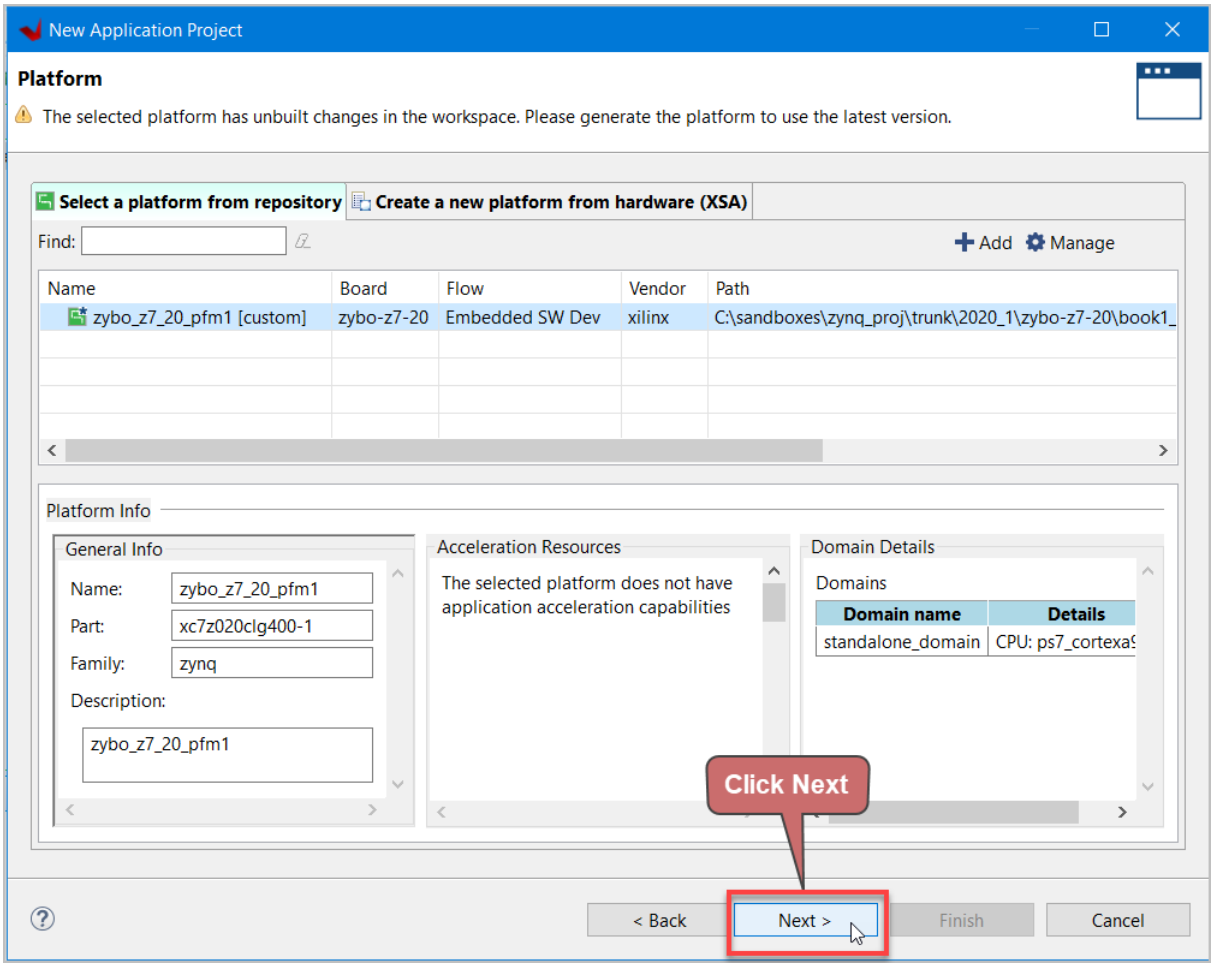


Figure 219. Leave defaults, and click Next to continue.

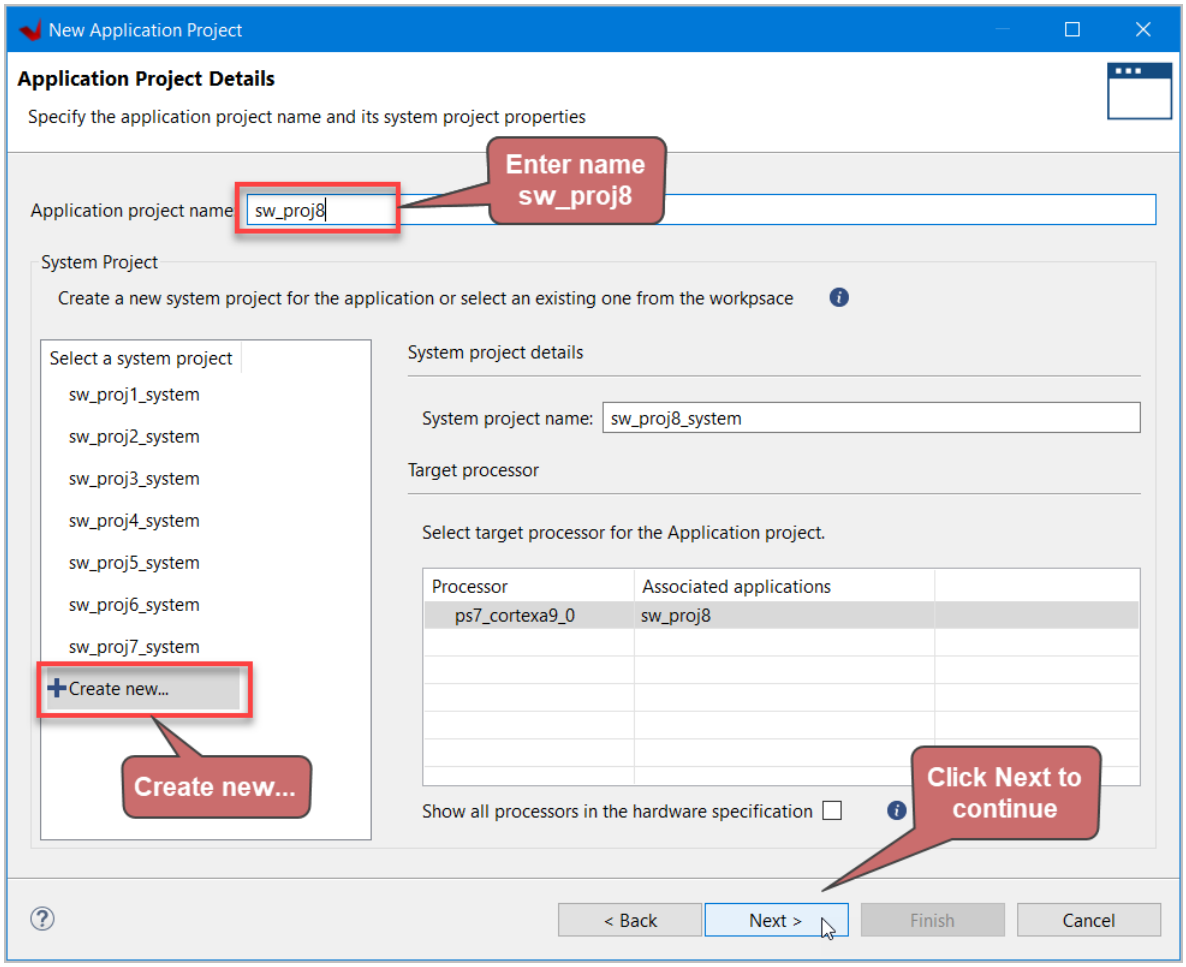


Figure 220. Set the project name to 'sw_proj8', ensure Create New... is selected, and press Next to continue.

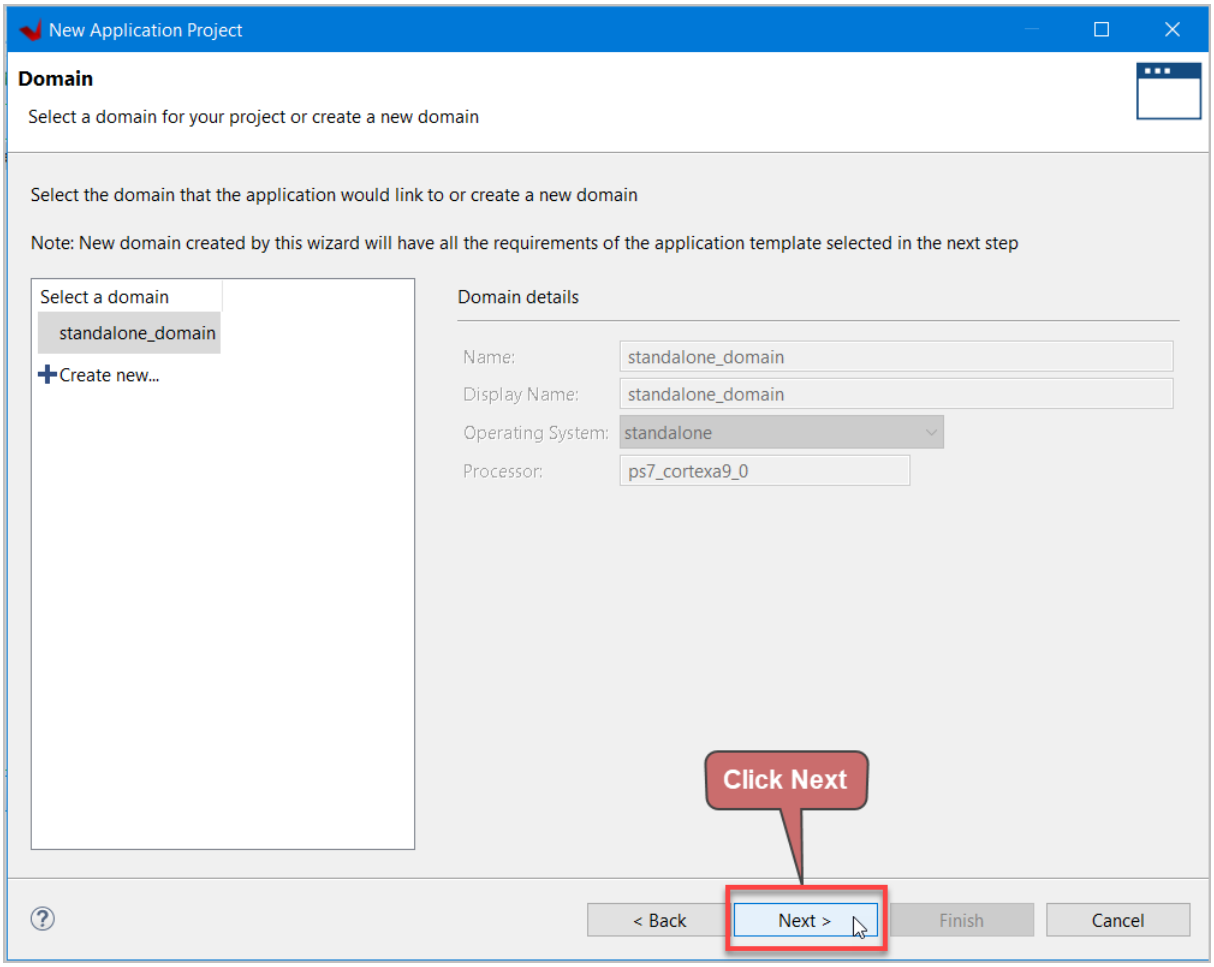


Figure 221. The standalone domain should be selected by default. Click Next to continue.

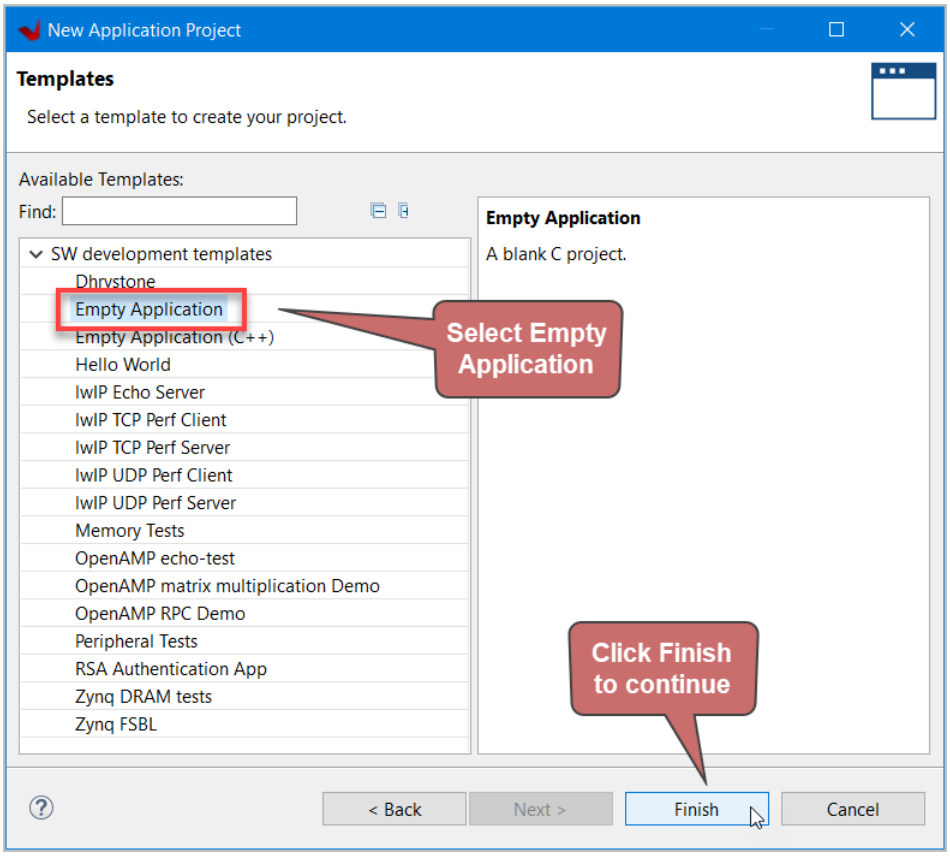


Figure 222. Select Empty Application, and click Finish.

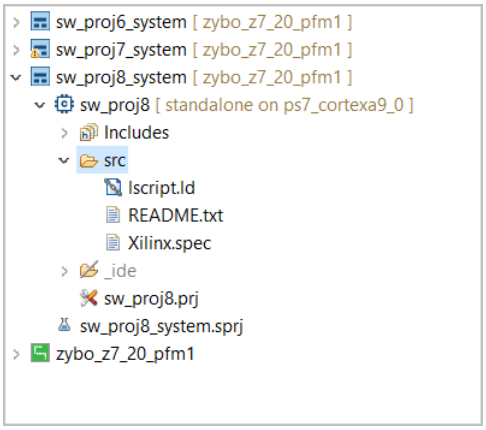


Figure 223. The empty application is created.

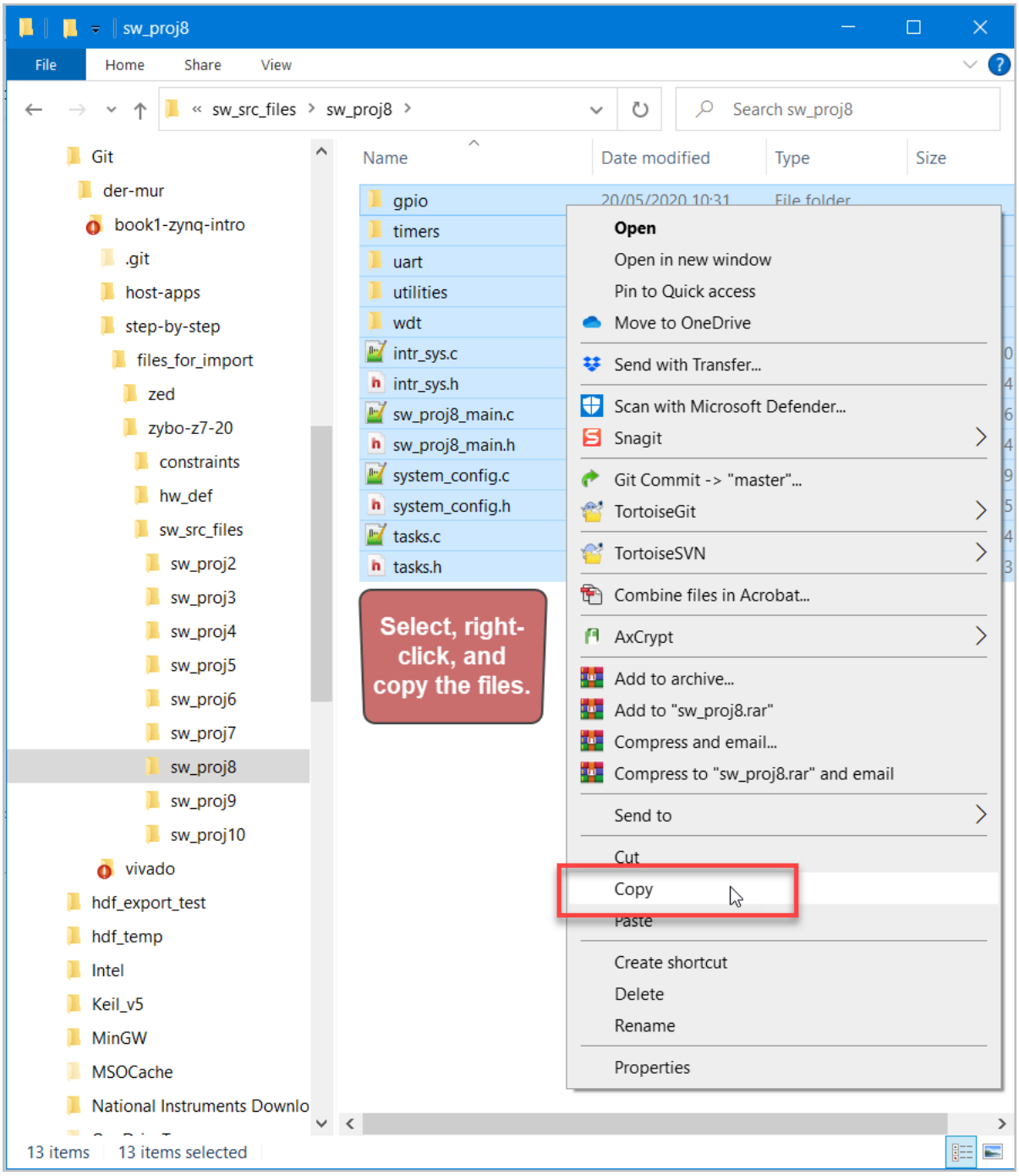


Figure 224. Select the project files, right-click, and select Copy.

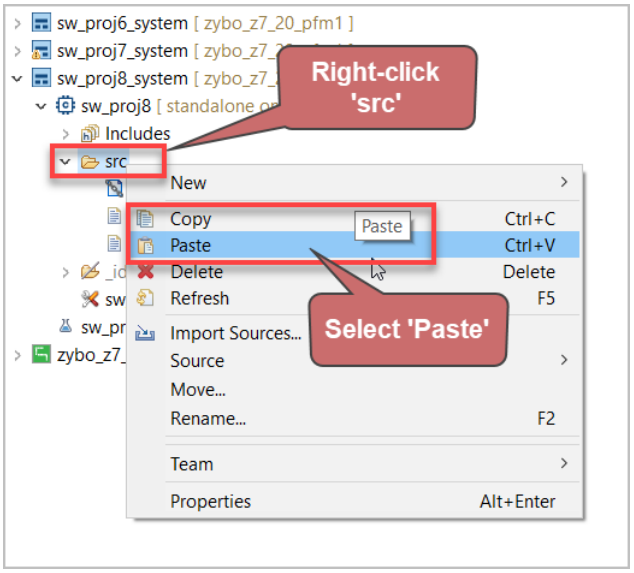


Figure 225. In Vitis, right-click the 'src' directory, and paste the files.

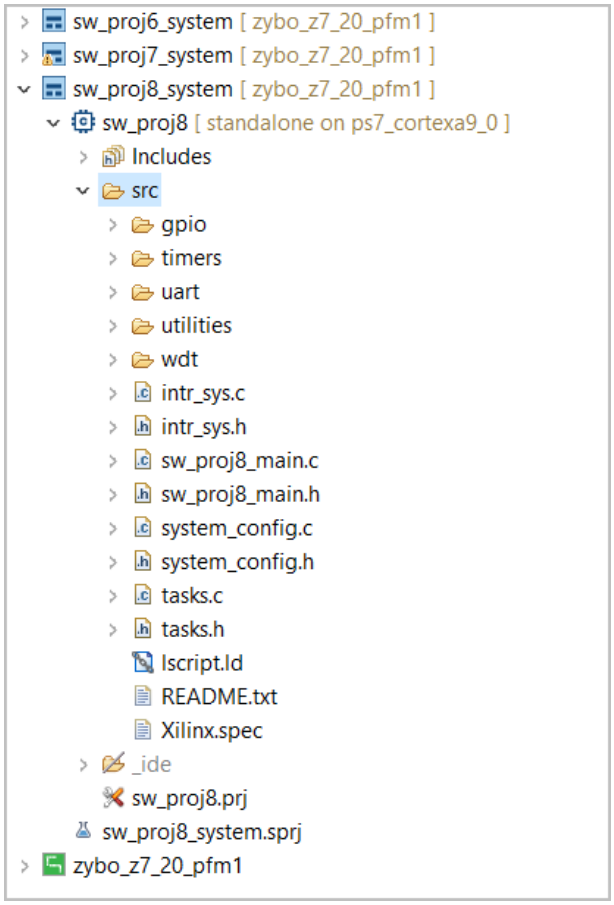


Figure 226. The project files are added.

3.11.2 Build the Project

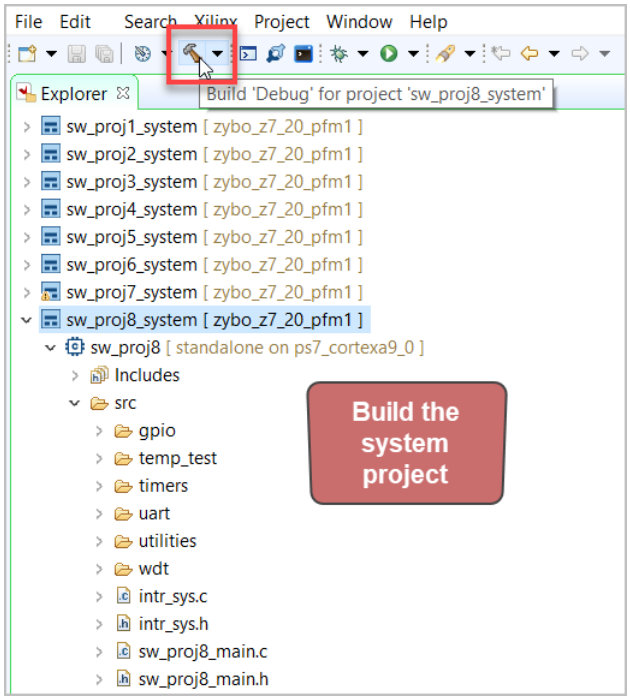


Figure 227. Build the system project

3.11.3 Run the Project

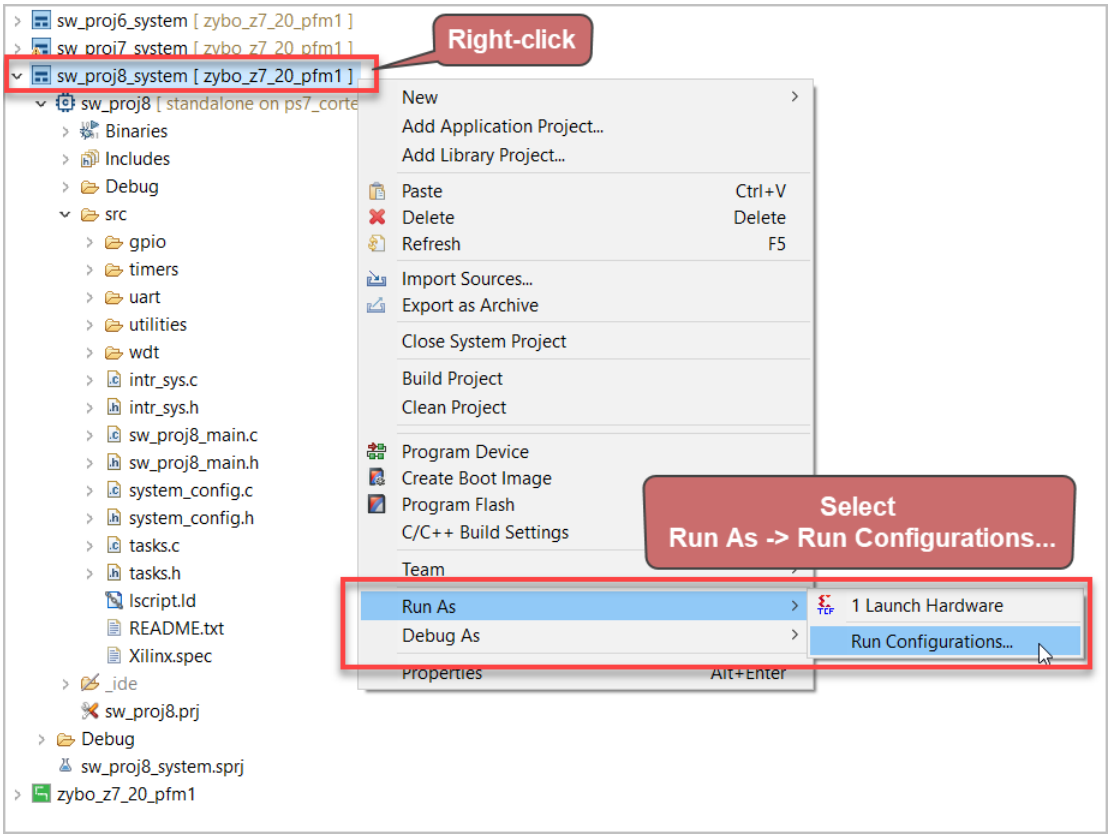


Figure 228. Right-click on the system project and select Run As -> Run Configurations.

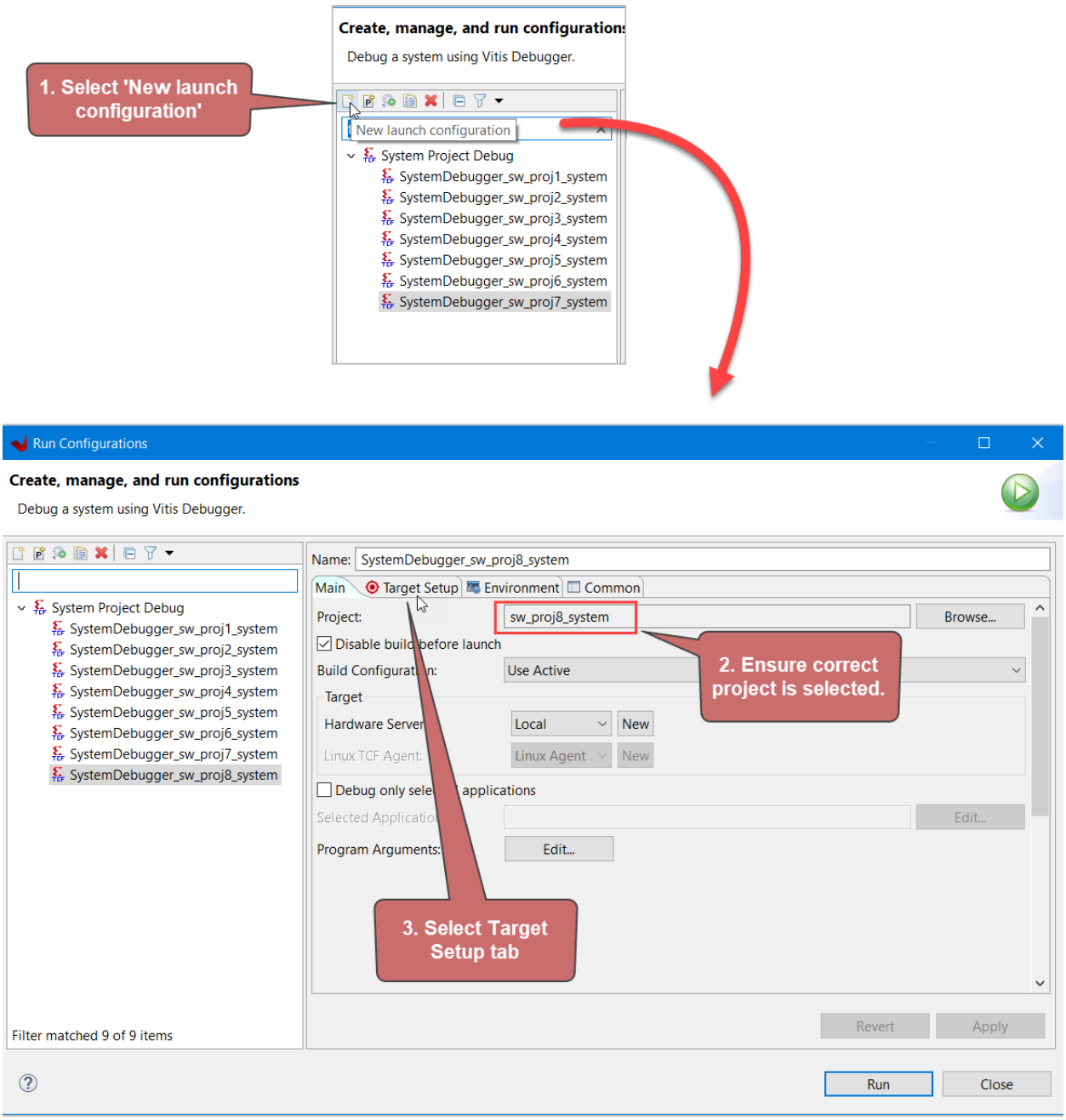


Figure 229. Create a New launch configuration

1. Click on 'New launch configuration'.
2. Ensure that "sw_proj8_system" is selected.
3. Select the Target Setup Tab.

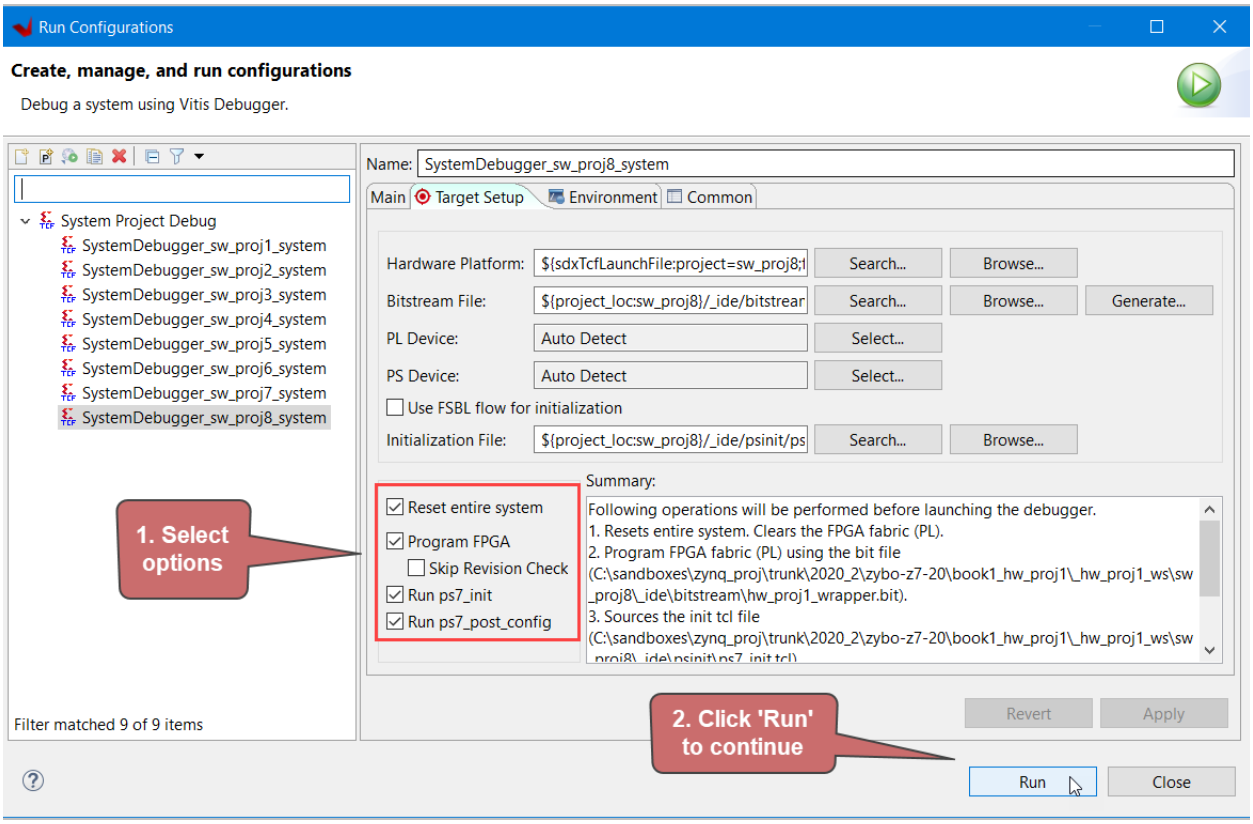


Figure 230. Run the application.

1. Select the main options as follows:
    a. Reset entire system
    b. Program FPGA
    c. Run ps7_init
    d. Run ps7_post_config

Click on 'Run' to continue. The FPGA should be programmed, and the ELF file should be downloaded to the platform, as shown in the next page.

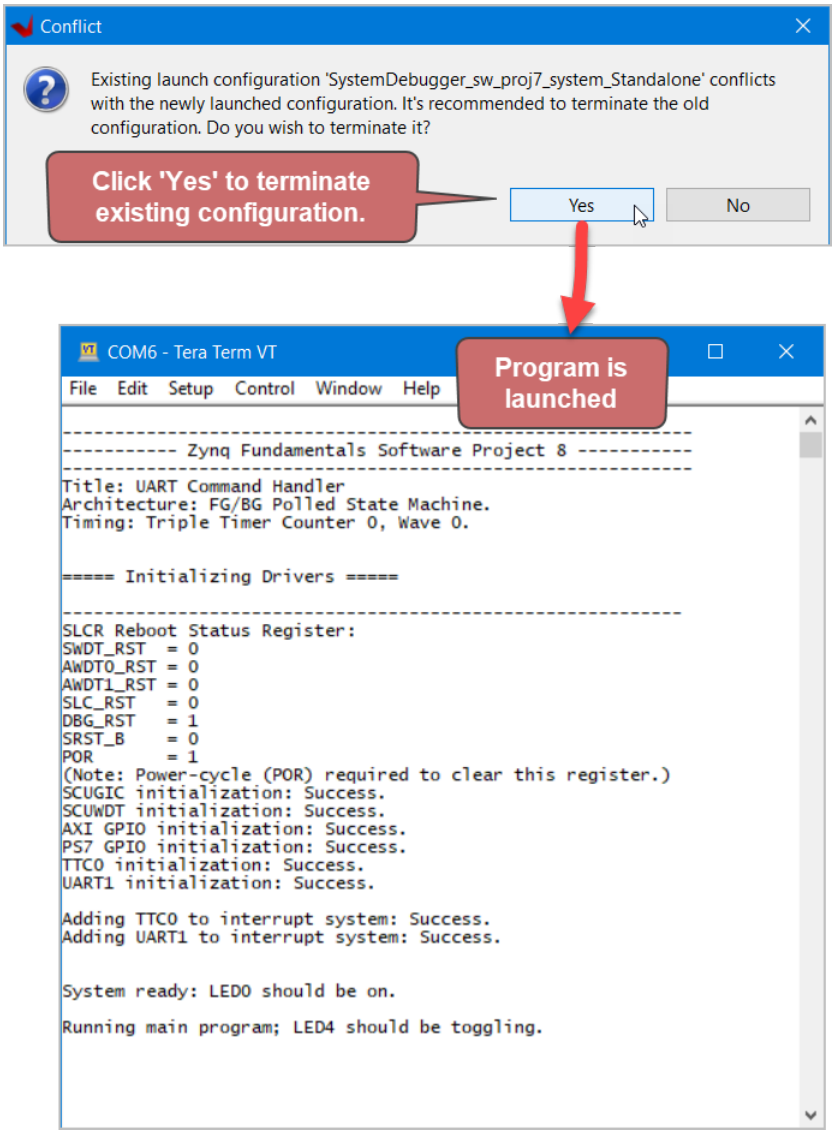


Figure 231. The project details for the software project should appear in the terminal.

If prompted, select 'Yes' to terminate any existing configuration. The FPGA will be programmed and the application will be downloaded to the processor. The terminal program should display the results for launching Software Project 8.

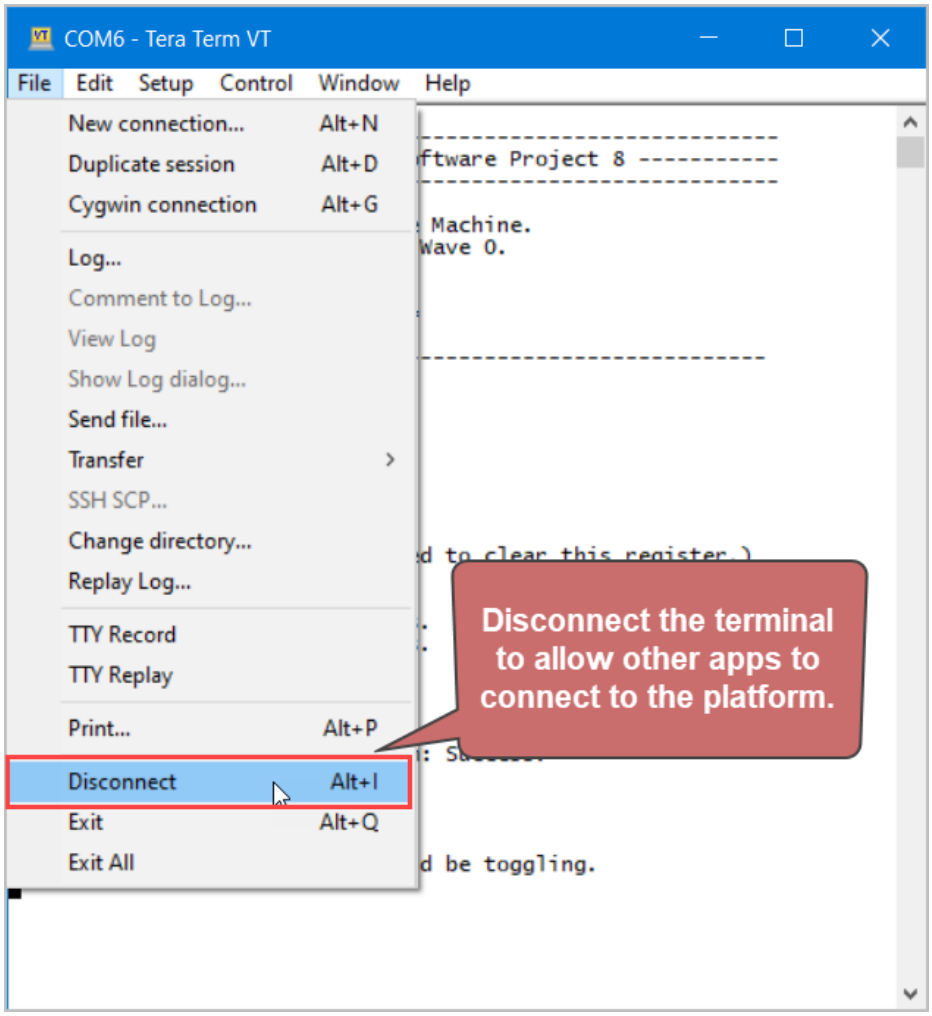


Figure 232. Disconnect the platform.

If the user wants to communicate with the command handler on the platform using a host application, the terminal needs to be disconnected.

3.12 System (Software) Project 9: Programmable Logic Interrupts

3.12.1 Create the Project

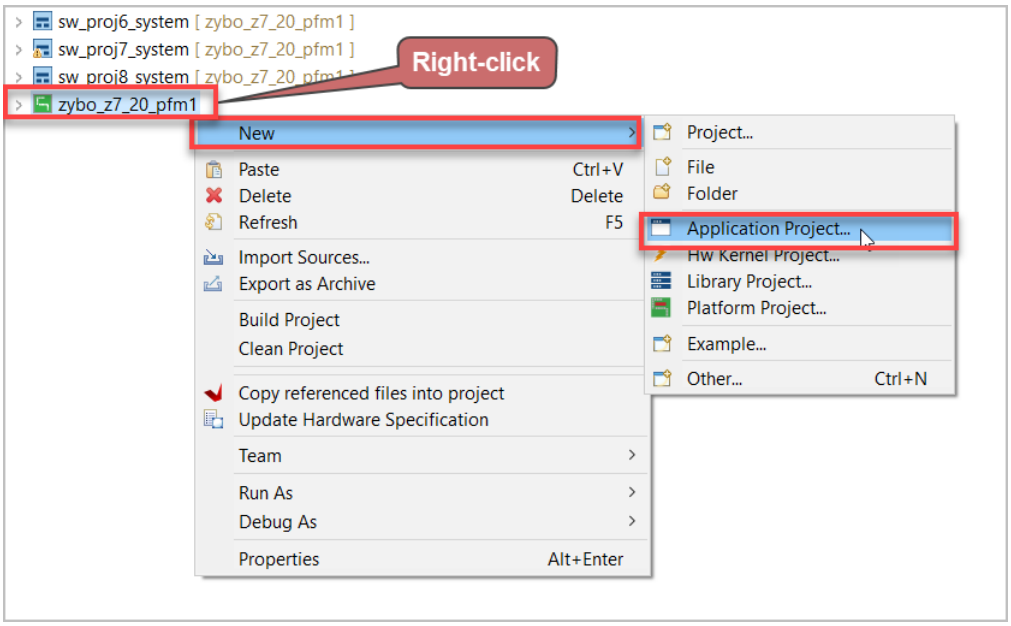


Figure 233. Right-click on the platform and select New-> Application Project.

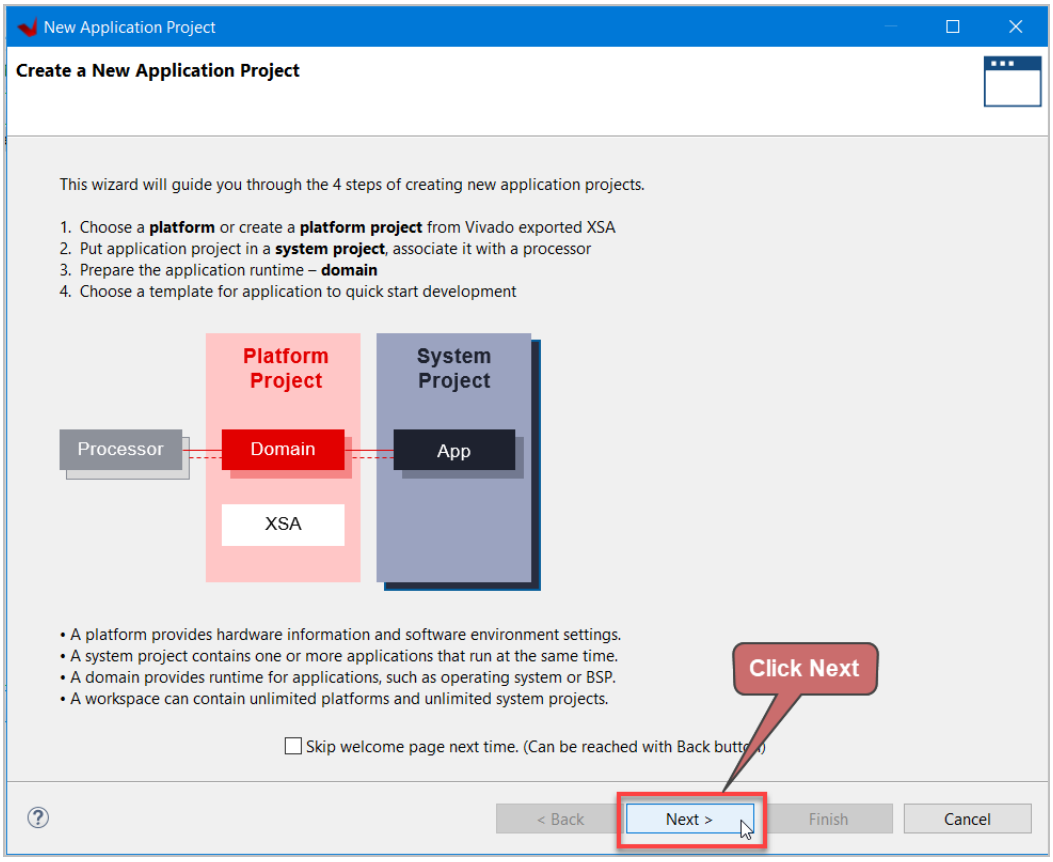


Figure 234. Click Next to continue.

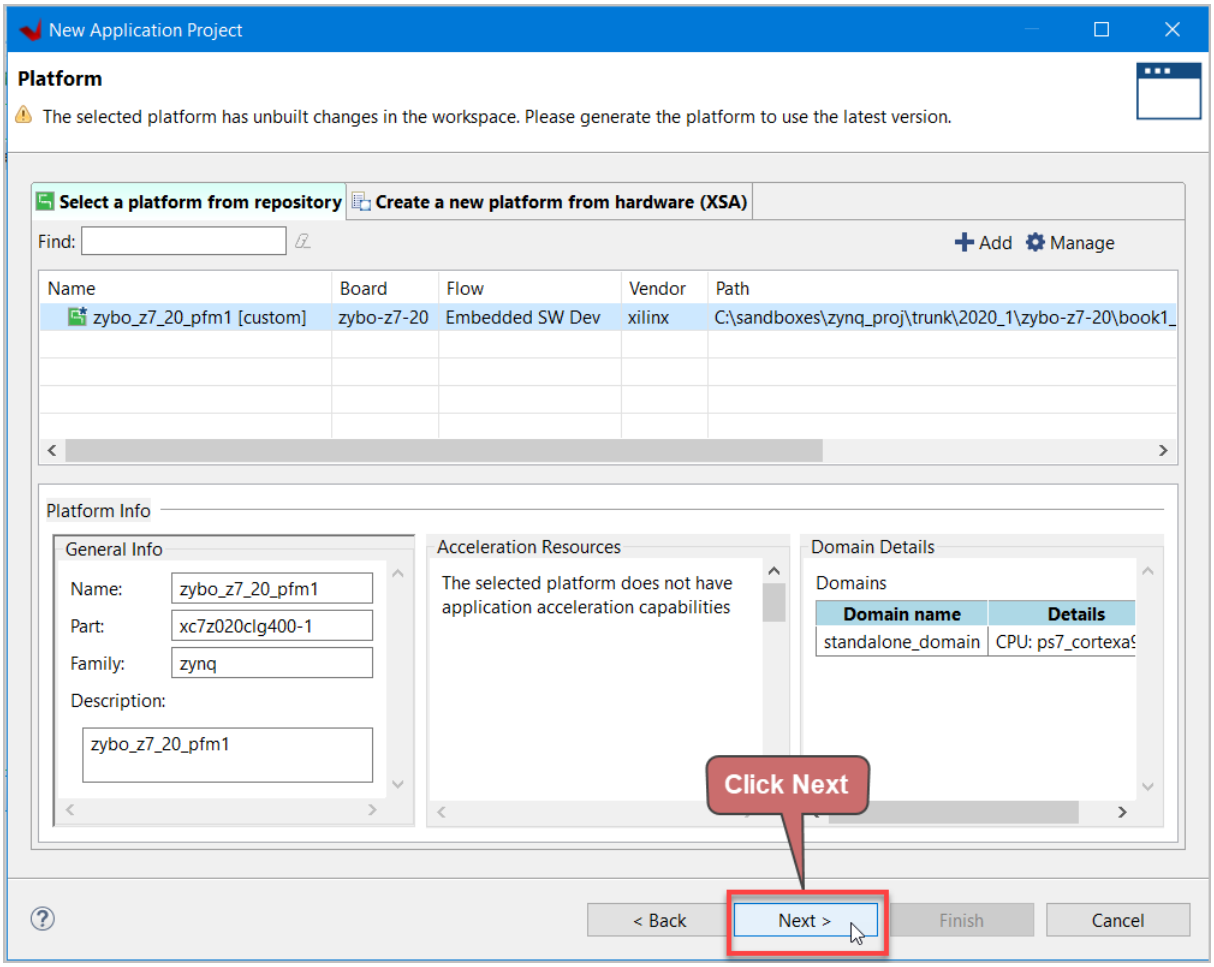


Figure 235. Leave defaults, and click Next to continue.

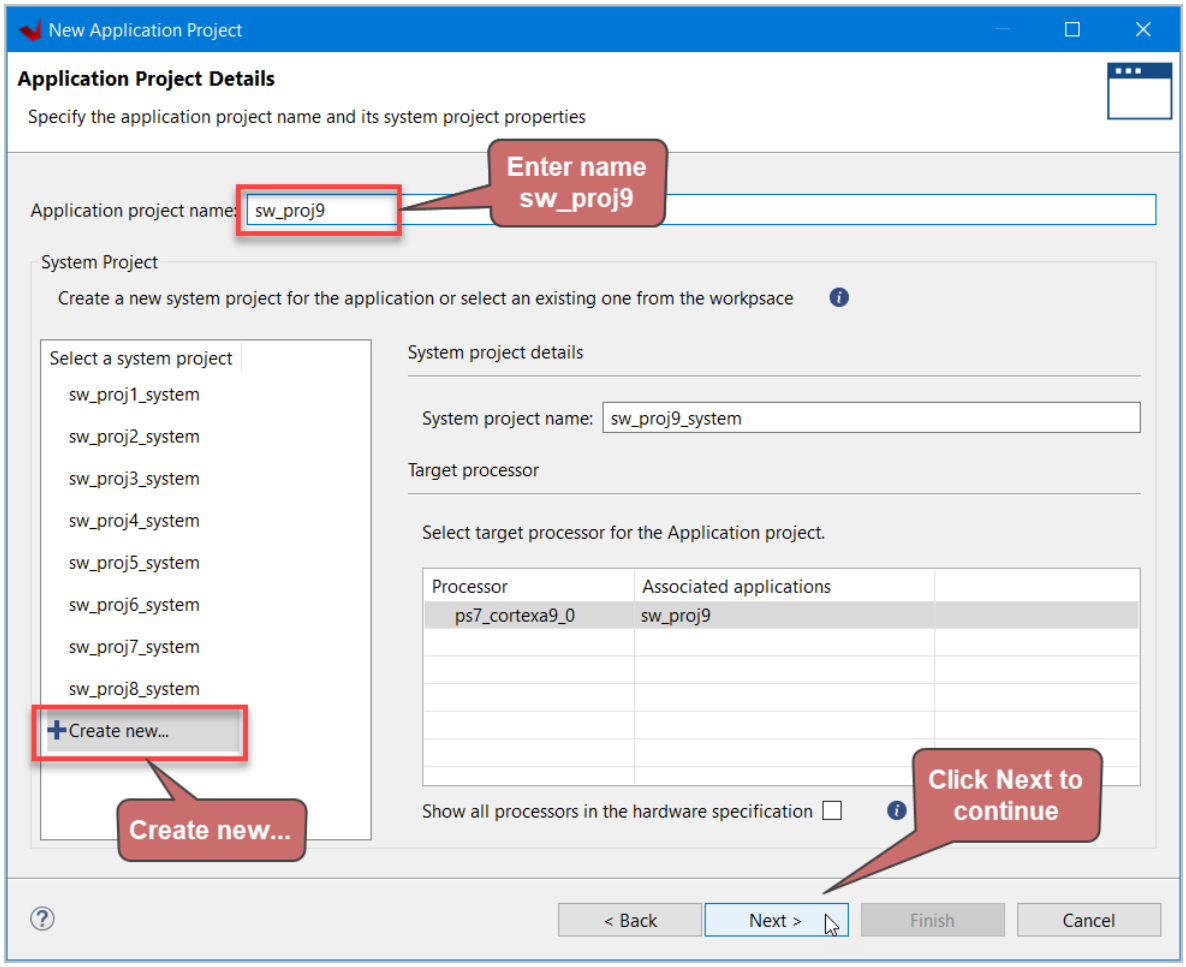


Figure 236. Set the project name to 'sw_proj9', ensure Create New... is selected, and press Next to continue.

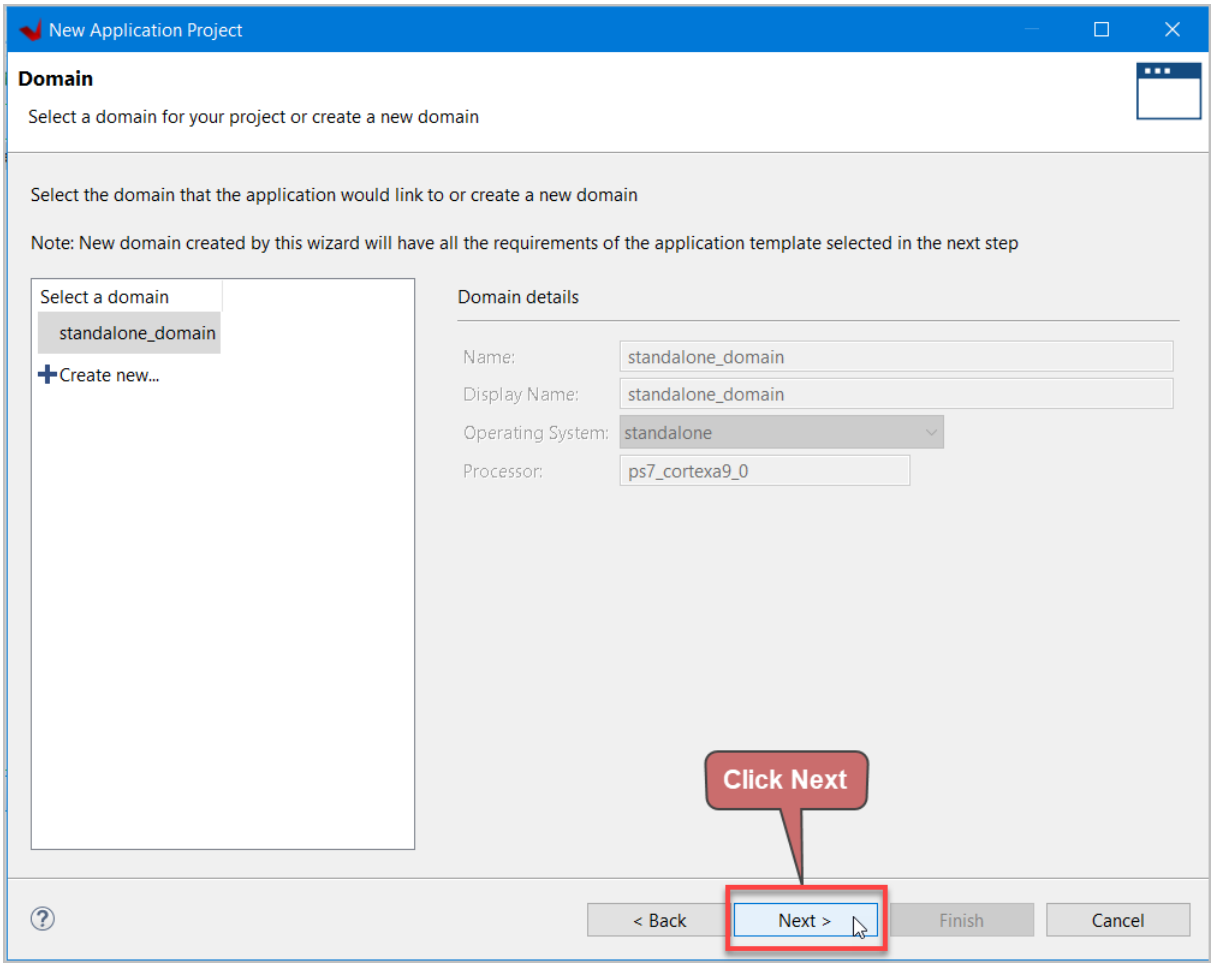


Figure 237. The standalone domain should be selected by default. Click Next to continue.

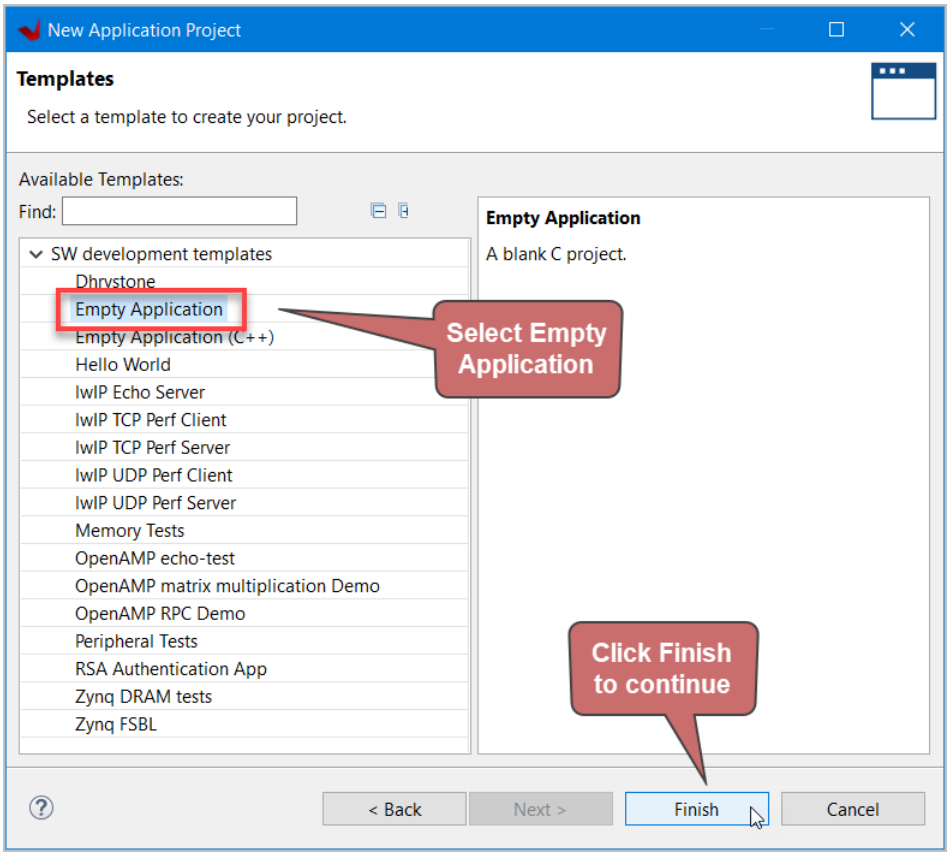


Figure 238. Select Empty Application, and click Finish.

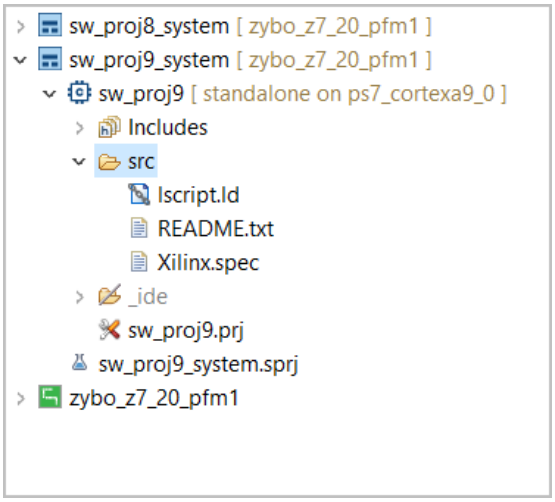


Figure 239. The empty application is created.

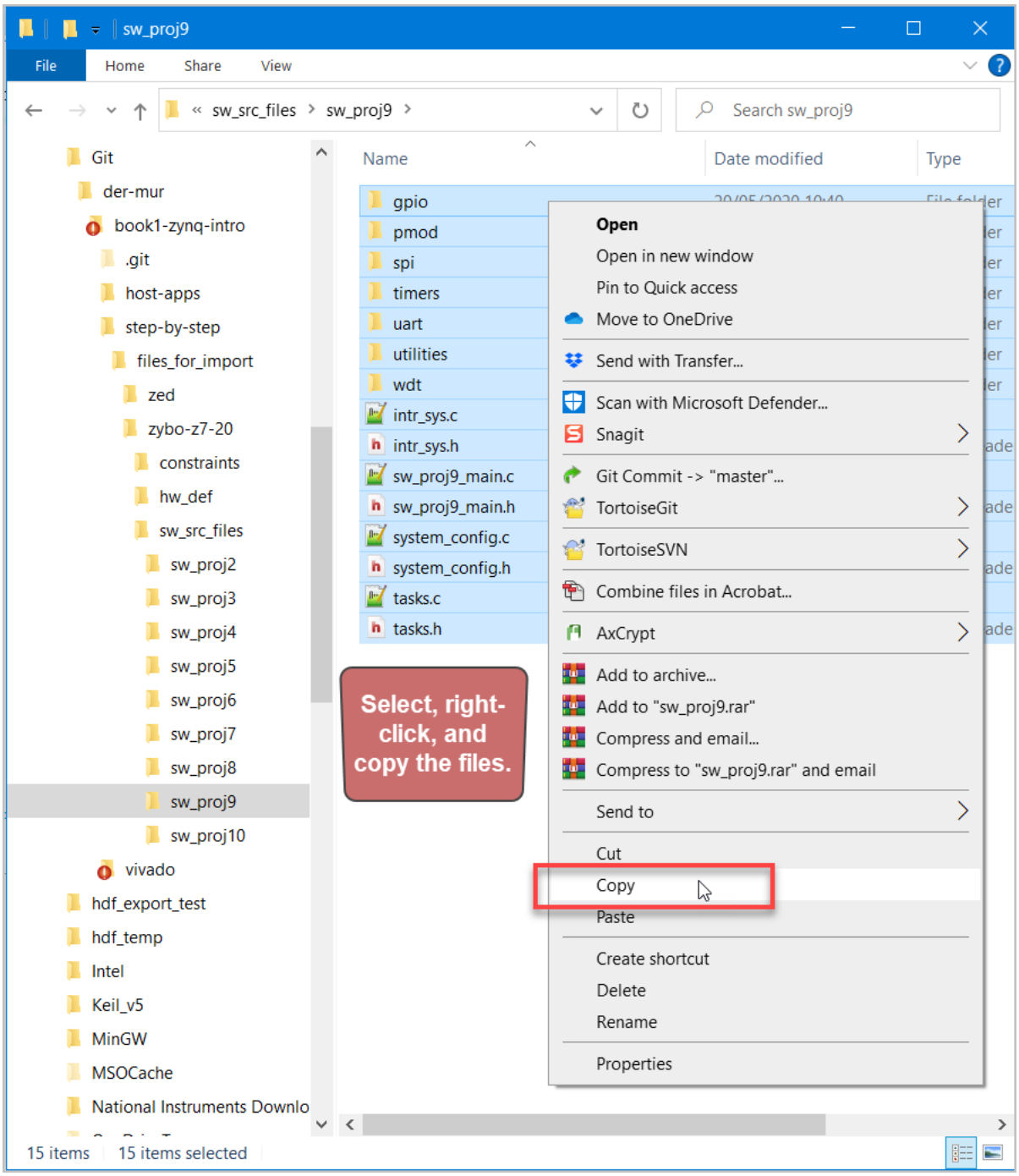


Figure 240. Select the project files, right-click, and select Copy.

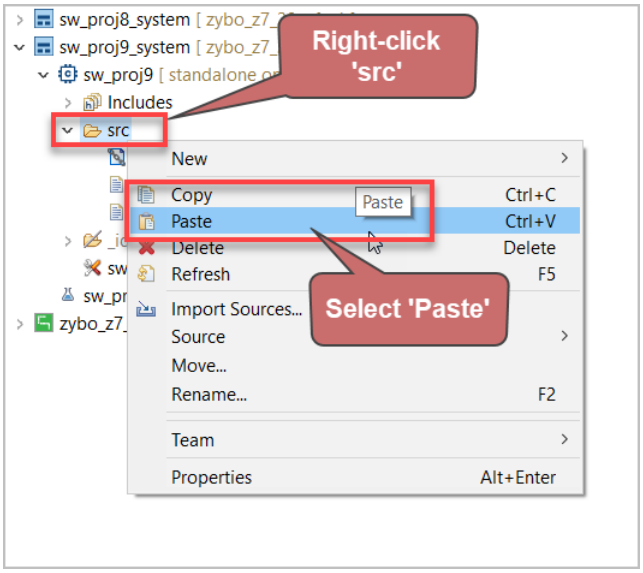


Figure 241. In Vitis, right-click the 'src' directory, and paste the files.

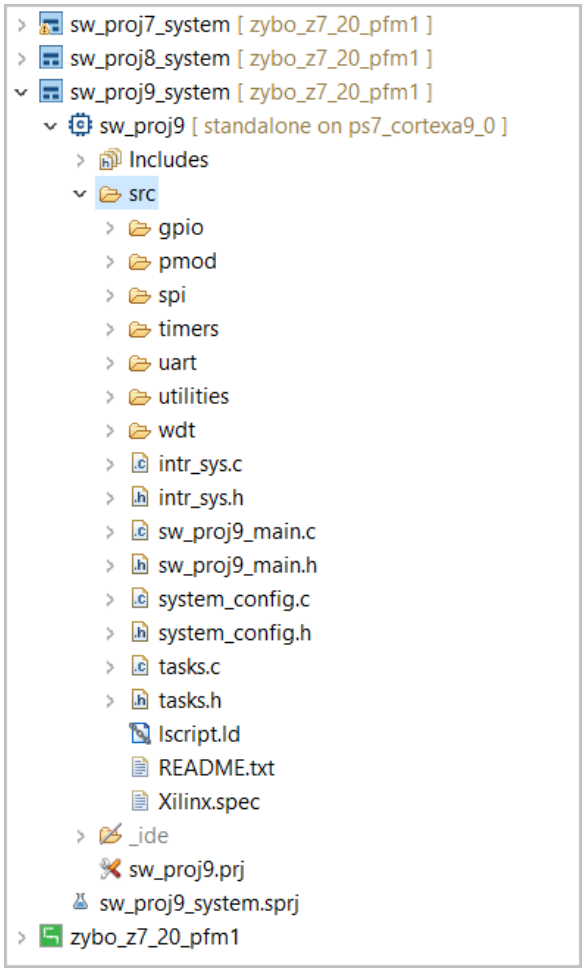


Figure 242. The project files are added.

3.12.2 Build the Project

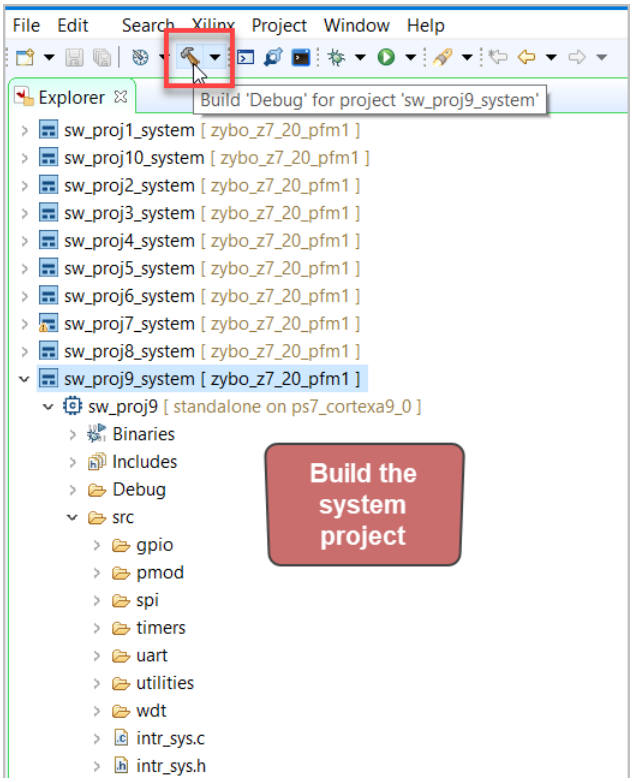


Figure 243. Build the system project

3.12.3 Run the Project

To run the project, start by reconnecting the COM port if it was disconnected after Software Project 8.

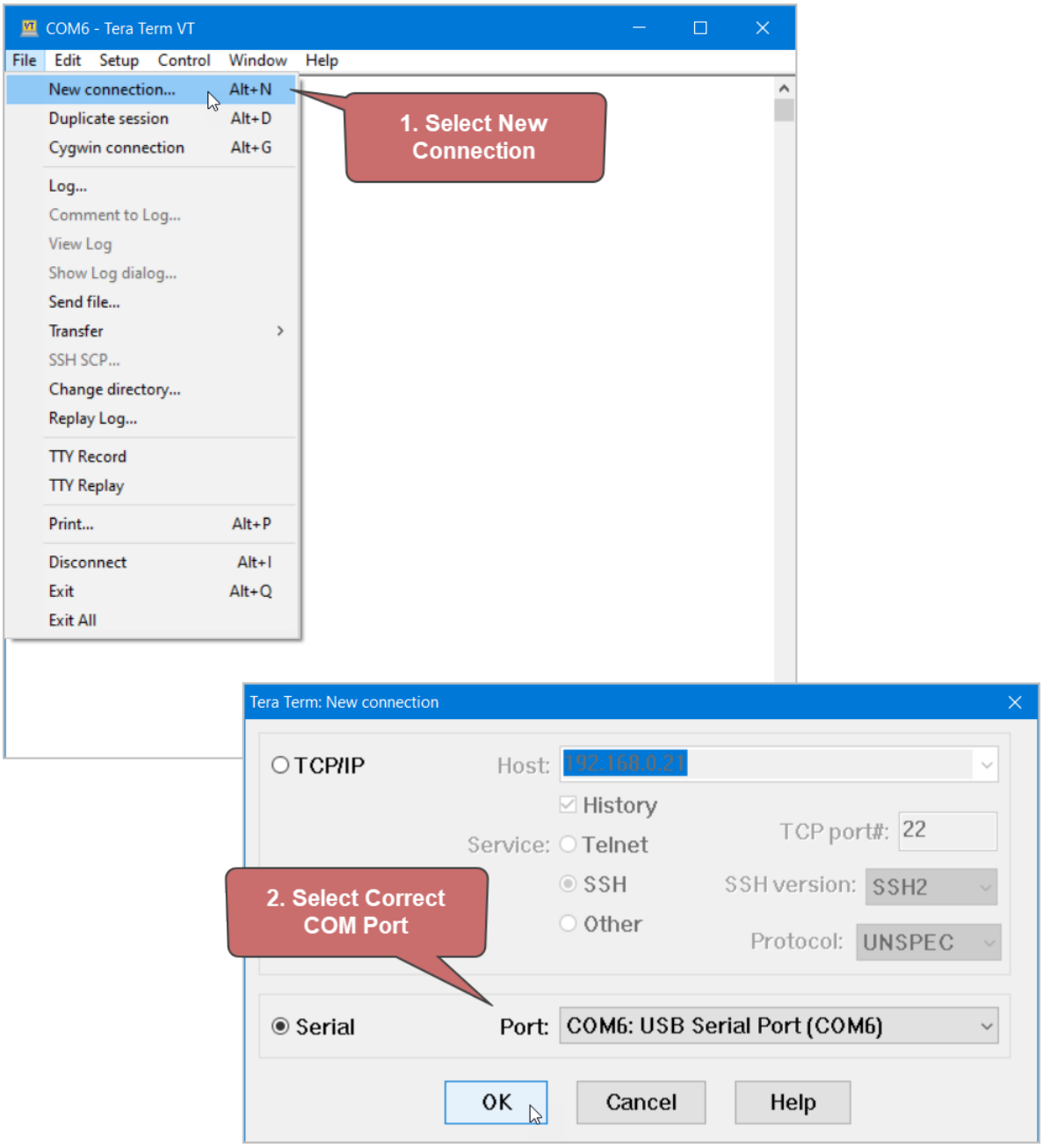


Figure 244. Reconnect terminal to platform if disconnected in last project.

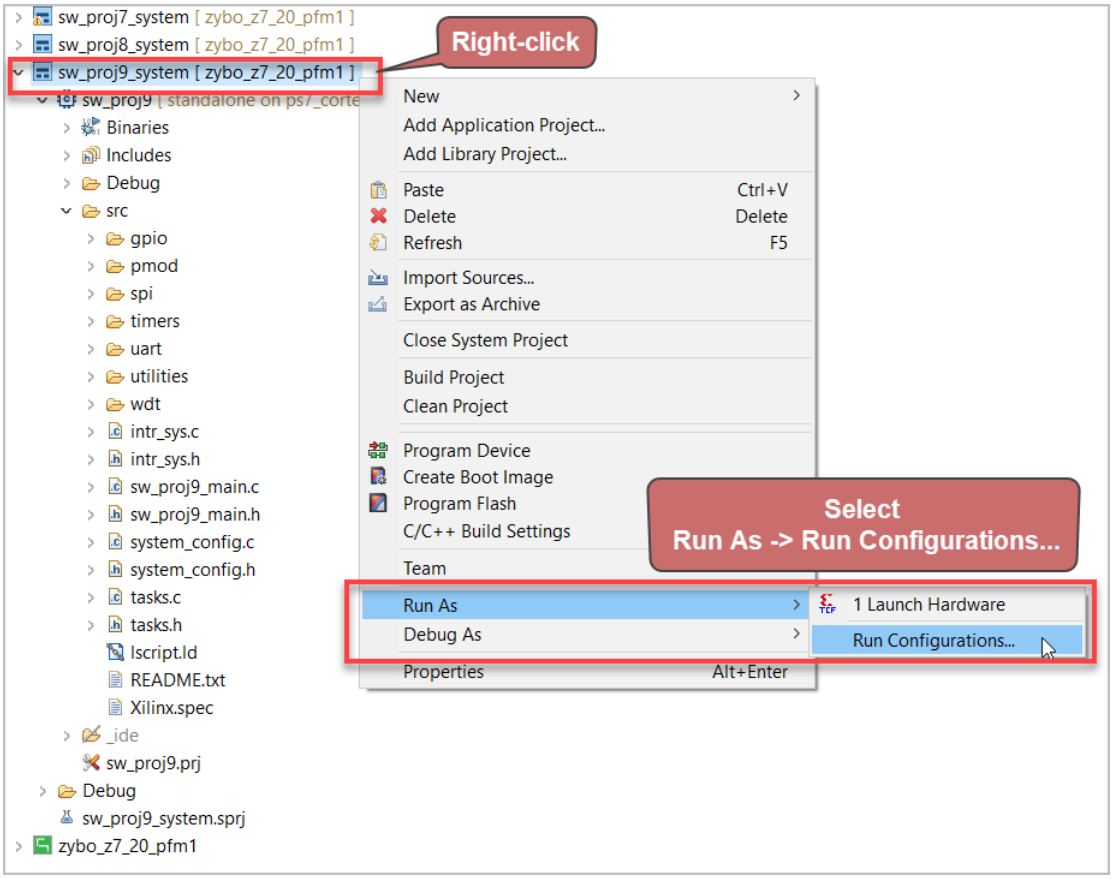


Figure 245. Right-click on the system project and select Run As -> Run Configurations.

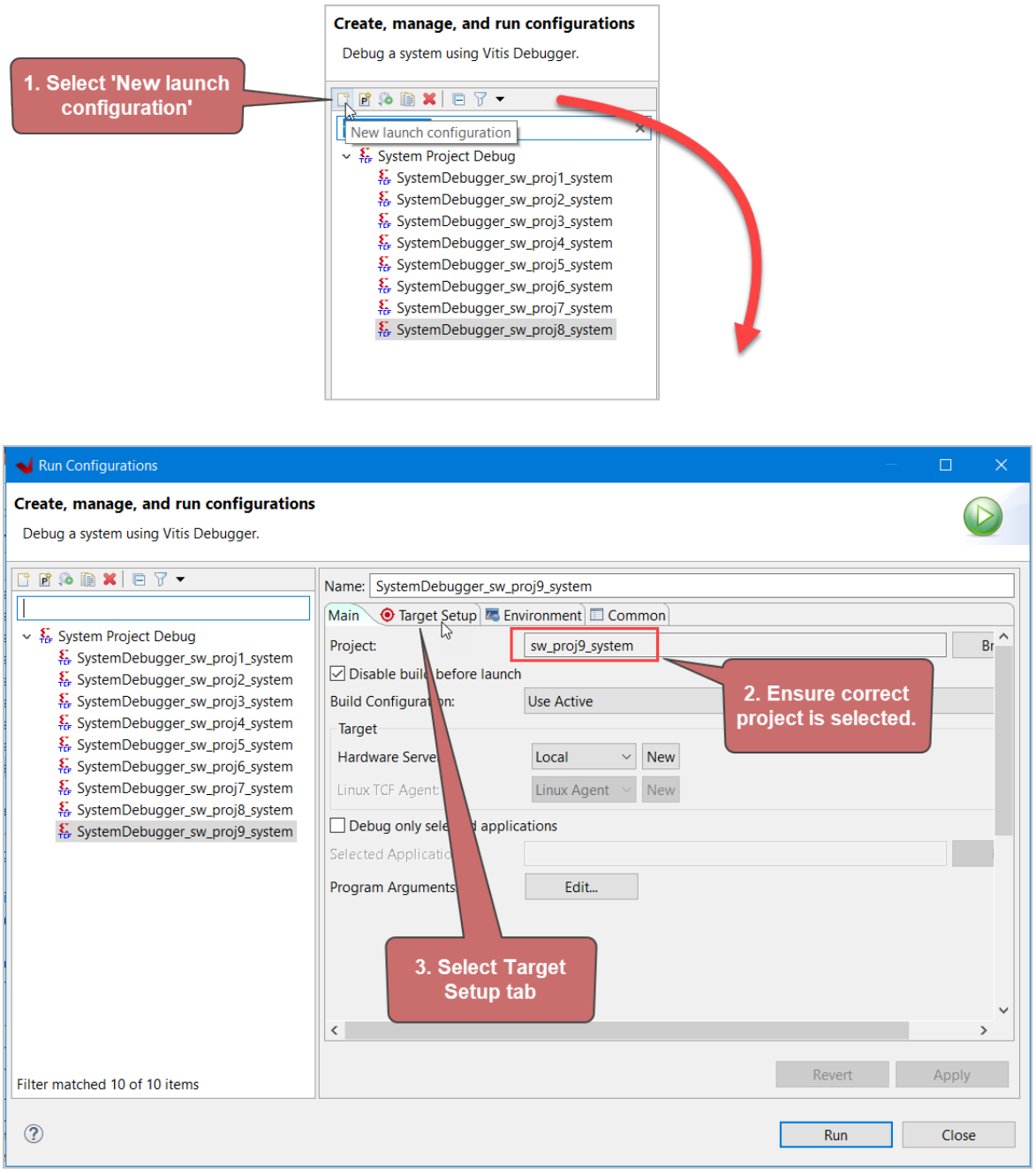


Figure 246. Create a New launch configuration

1. Click on 'New launch configuration'.
2. Ensure that "sw_proj9_system" is selected.
3. Select the Target Setup Tab.

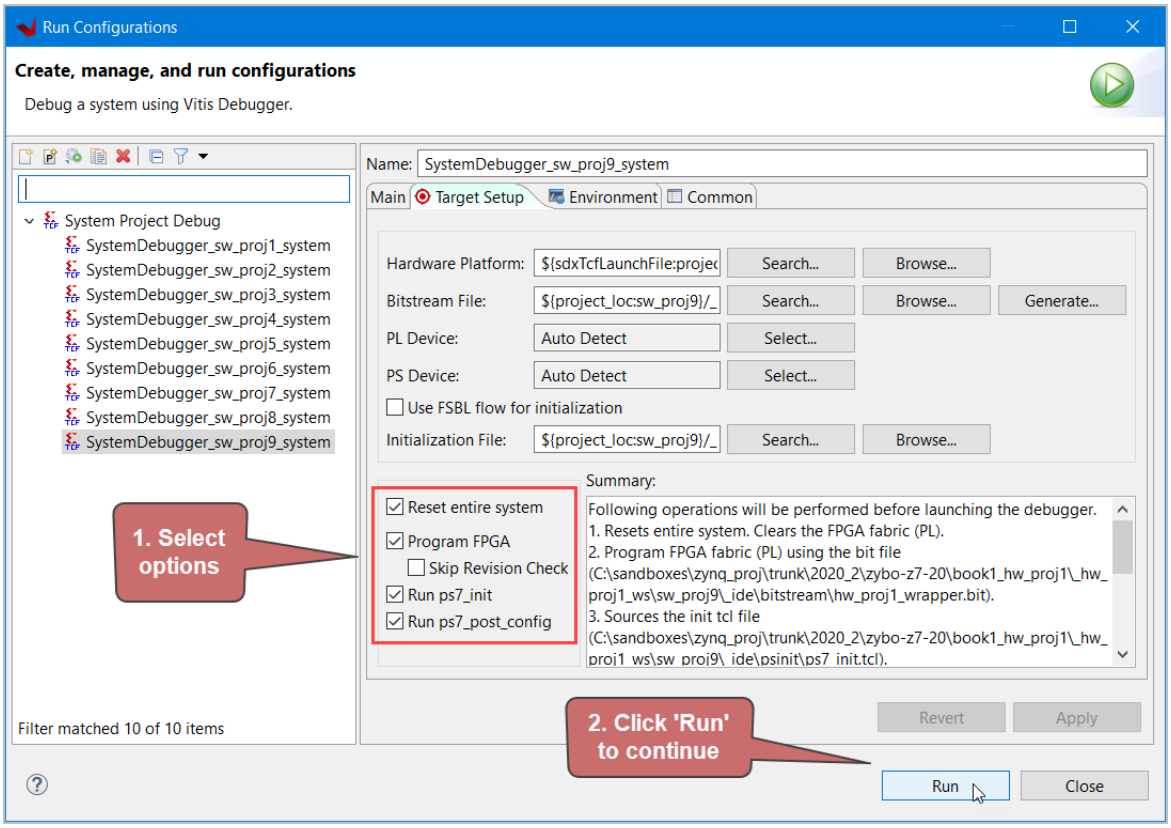


Figure 247. Run the application.

1. Select the main options as follows:
    a. Reset entire system
    b. Program FPGA
    c. Run ps7_init
    d. Run ps7_post_config

Click on 'Run' to continue. The FPGA should be programmed, and the ELF file should be downloaded to the platform, as shown in the next page.

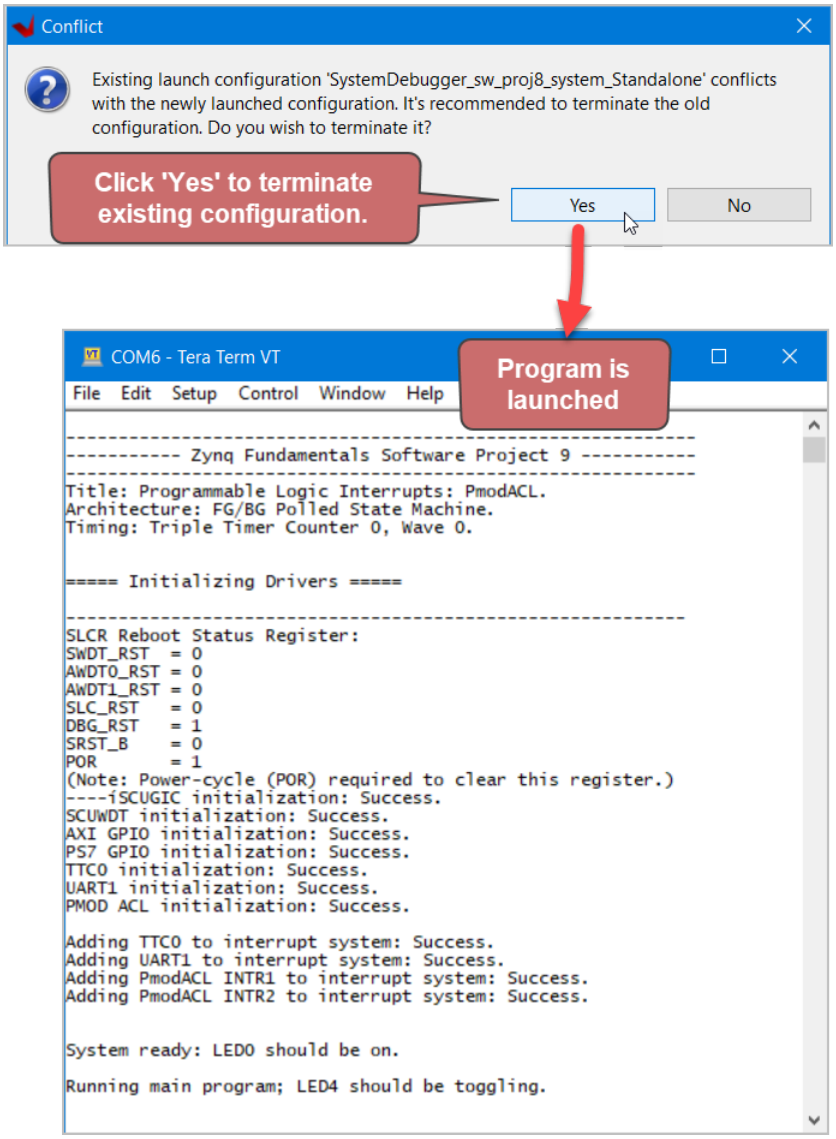


Figure 248. The project details for the software project should appear in the terminal.

If prompted, select 'Yes' to terminate any existing configuration. The FPGA will be programmed and the application will be downloaded to the processor. The terminal program should display the results for launching Software Project 9.

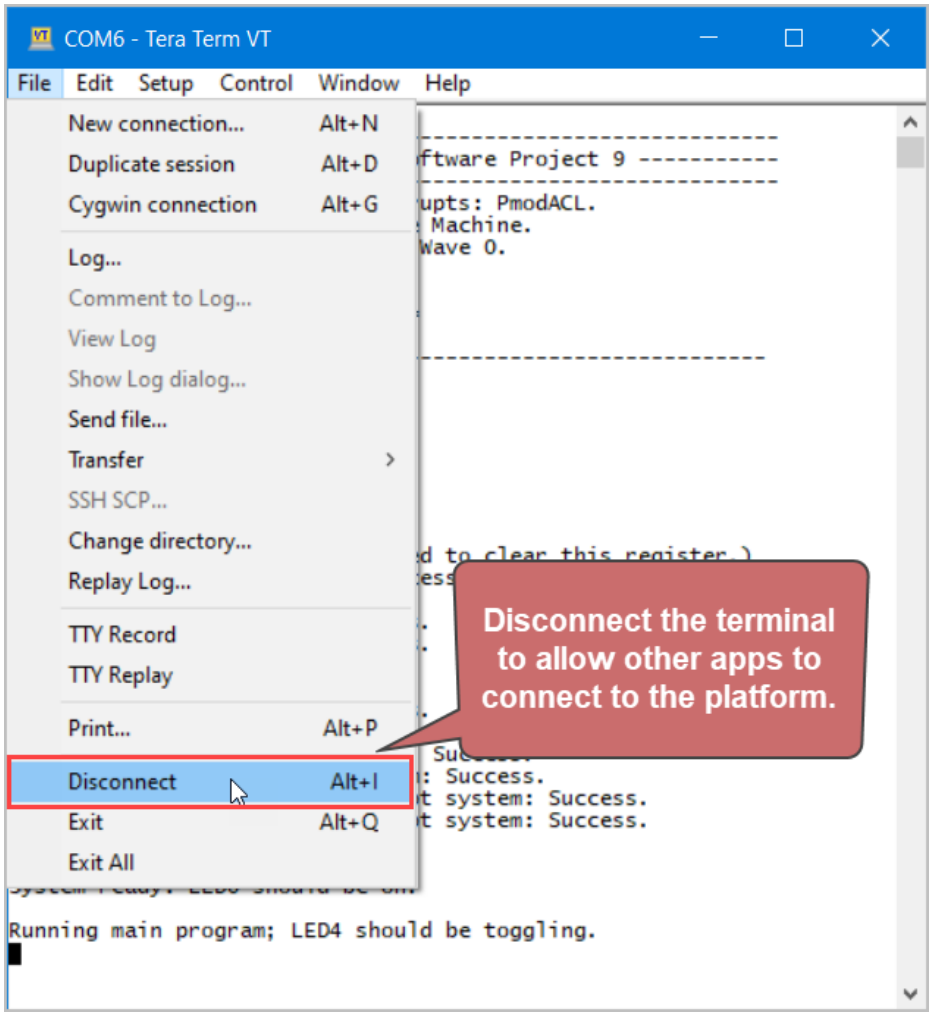


Figure 249. Disconnect the platform.

If the user wants to communicate with the command handler on the platform using a host application, the terminal needs to be disconnected.

3.13 System (Software) Project 10: Additional Interrupt Topics

3.13.1 Create the Project

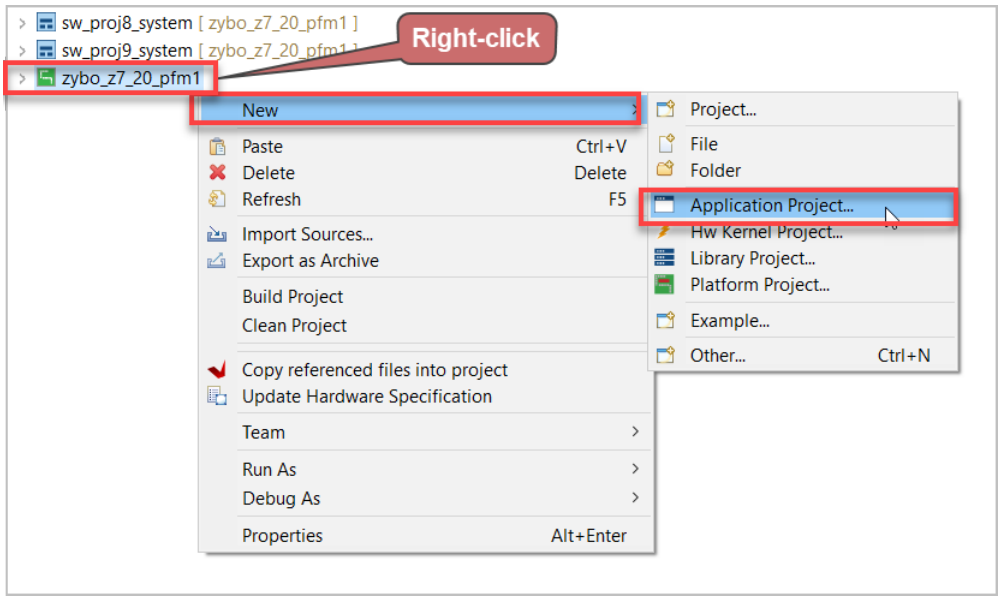


Figure 250. Right-click on the platform and select New-> Application Project.

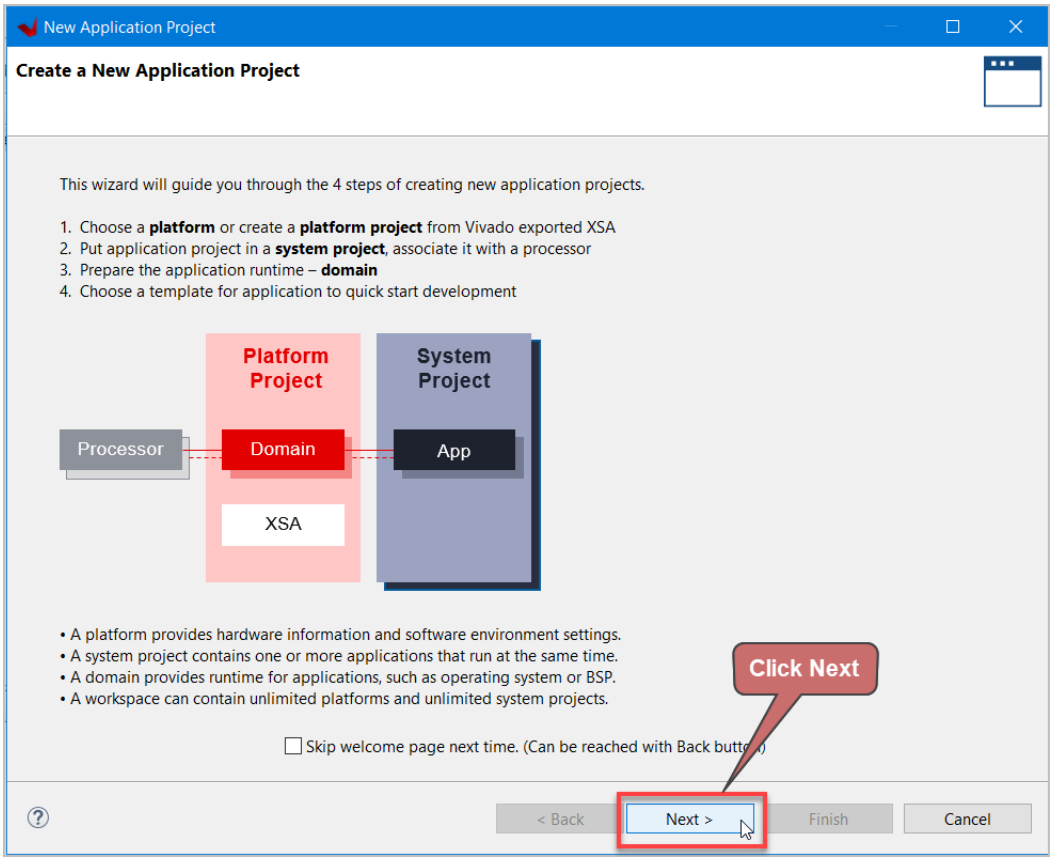


Figure 251. Click Next to continue.

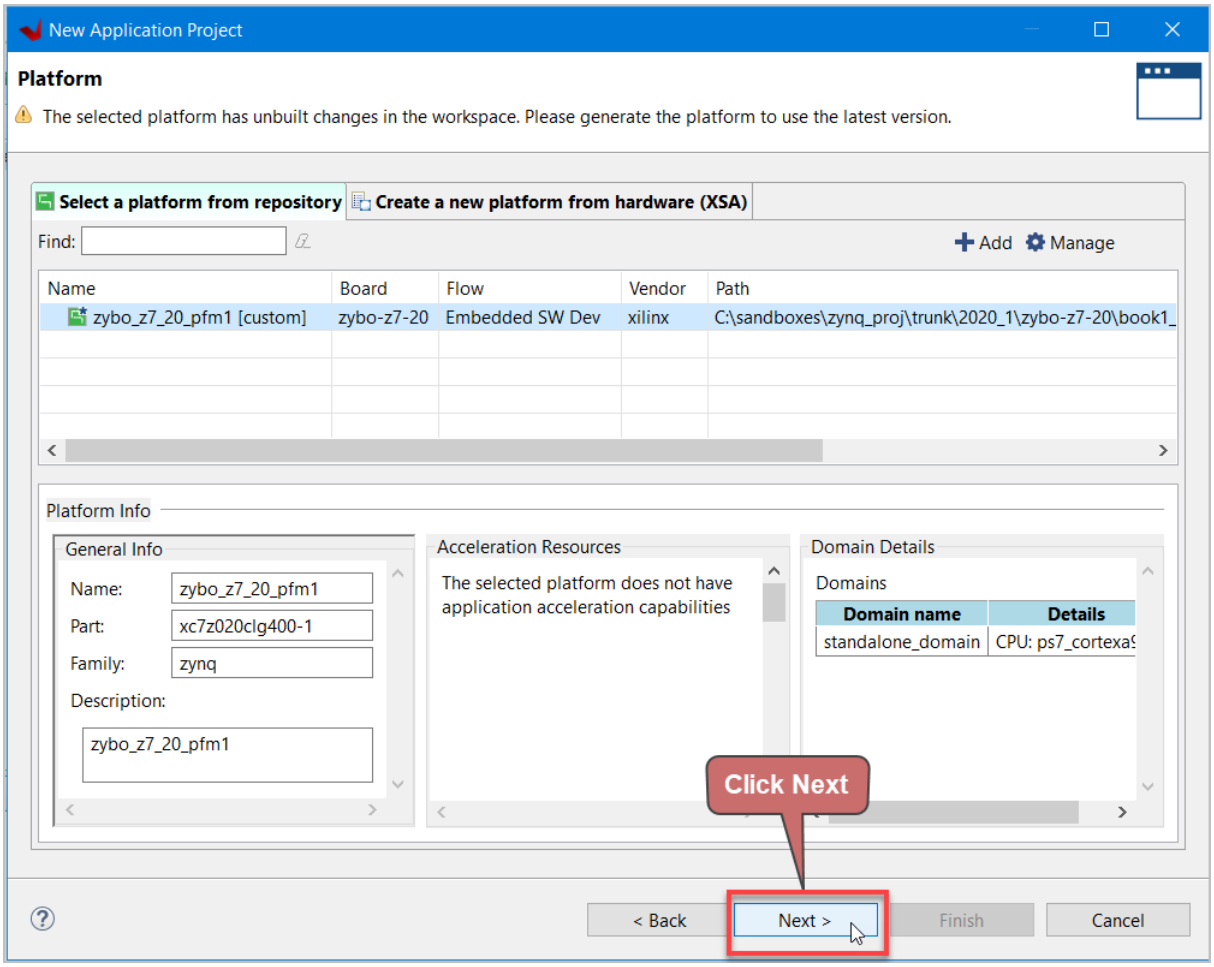


Figure 252. Leave defaults, and click Next to continue.

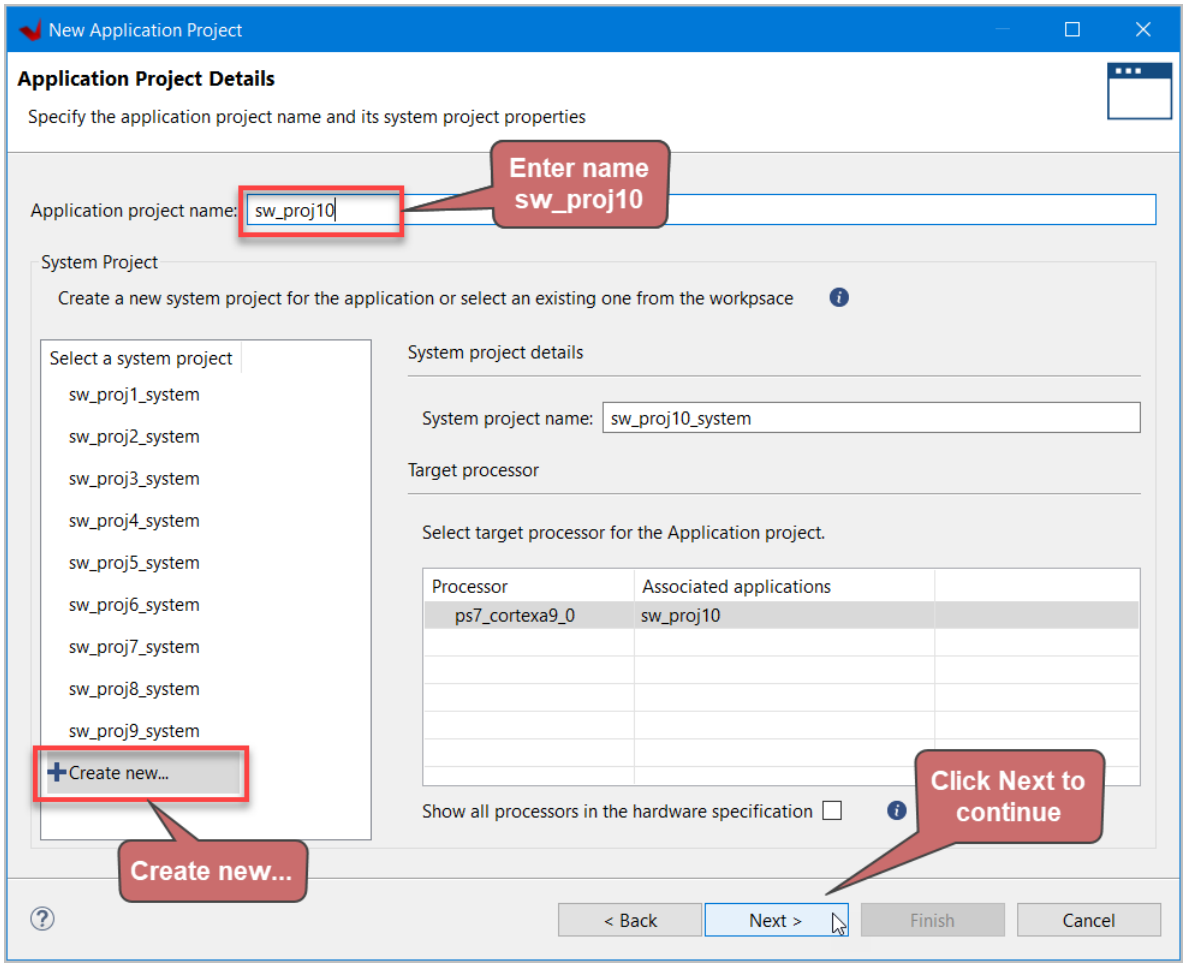


Figure 253. Set the project name to 'sw_proj10', ensure Create New... is selected, and press Next to continue.

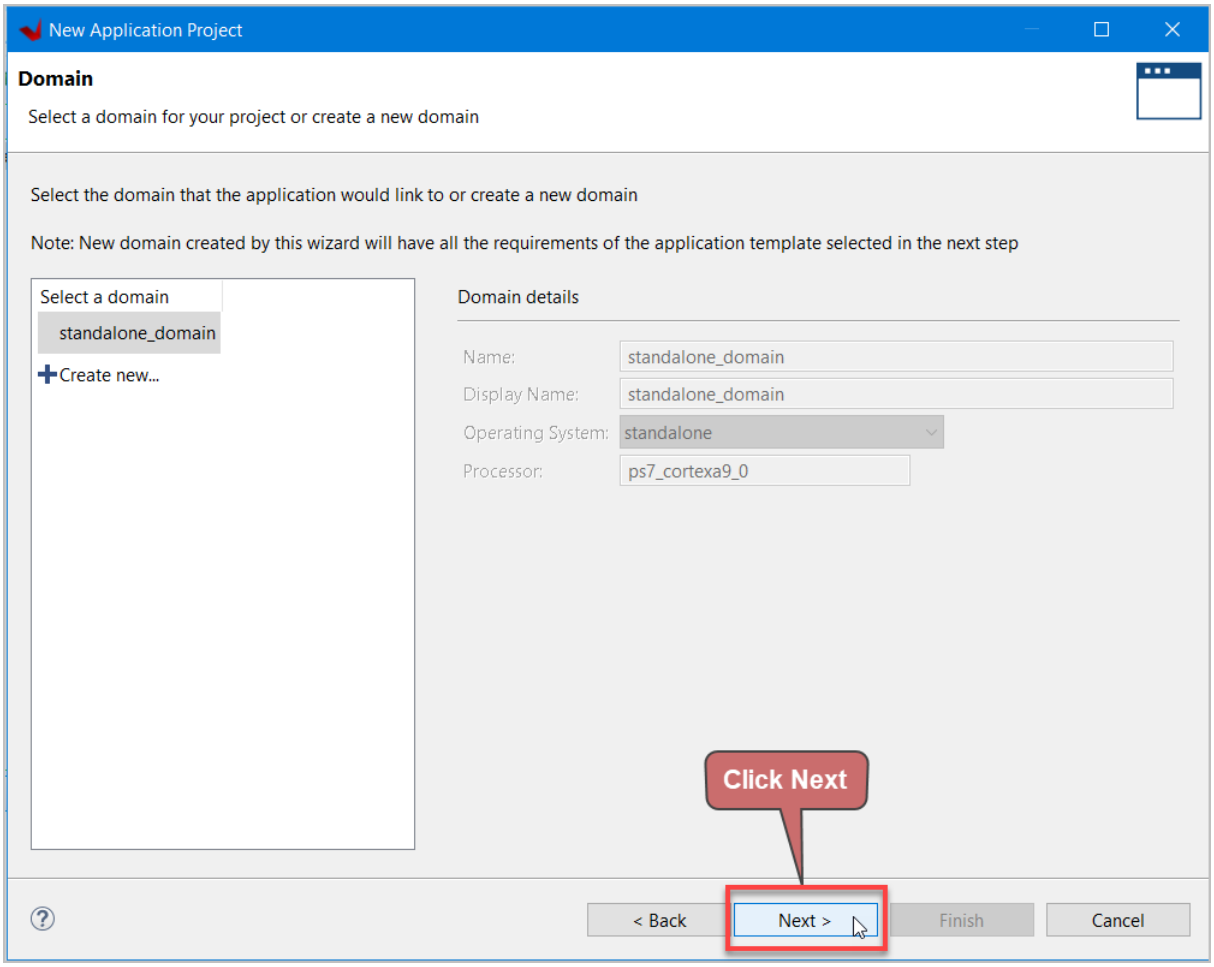


Figure 254. The standalone domain should be selected by default. Click Next to continue.

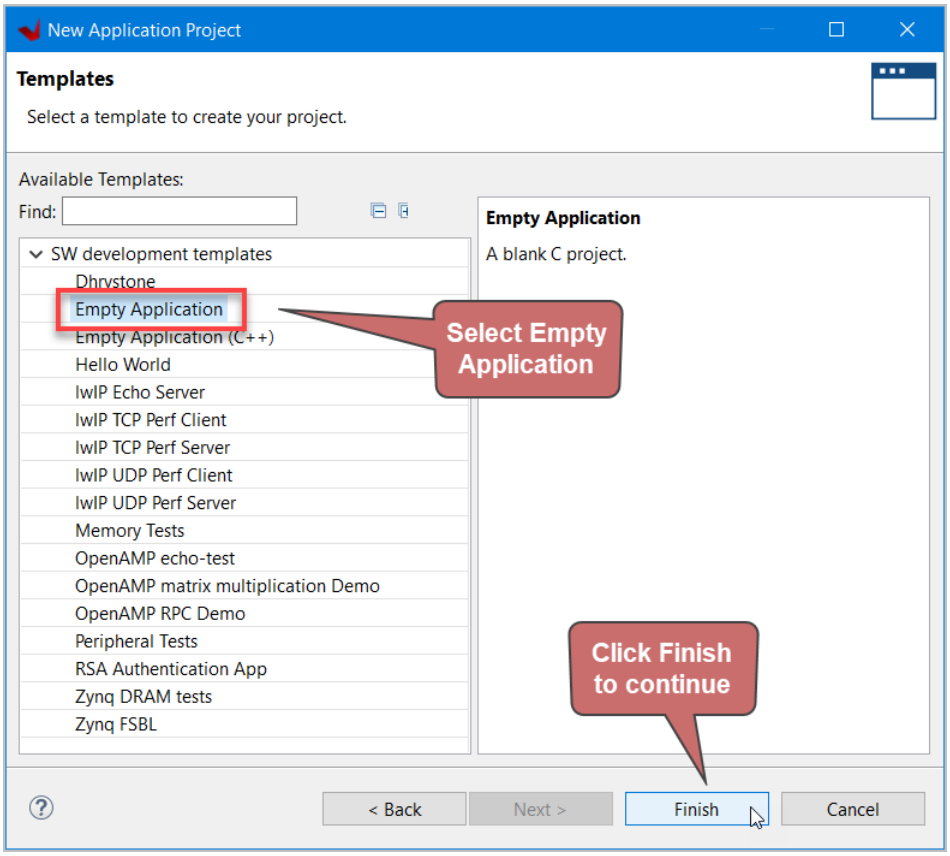


Figure 255. Select Empty Application, and click Finish.

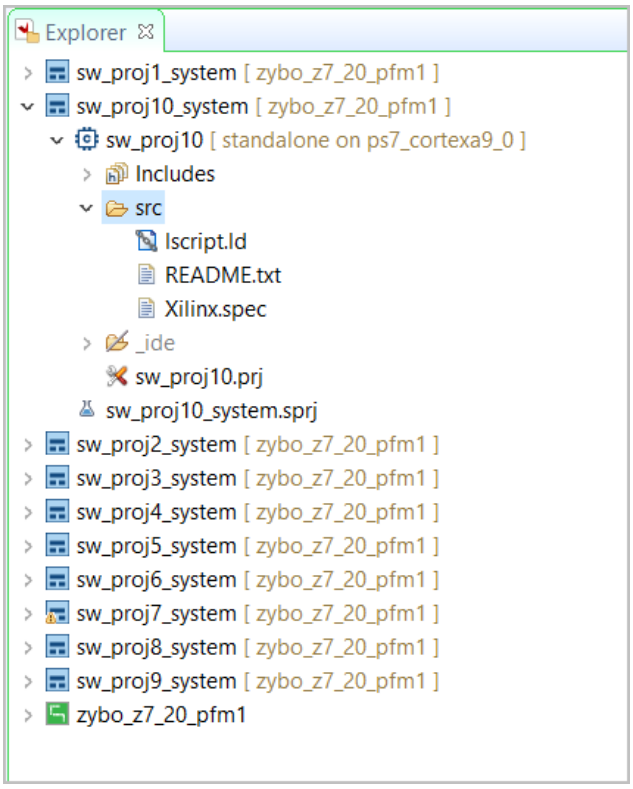


Figure 256. The empty application is created.

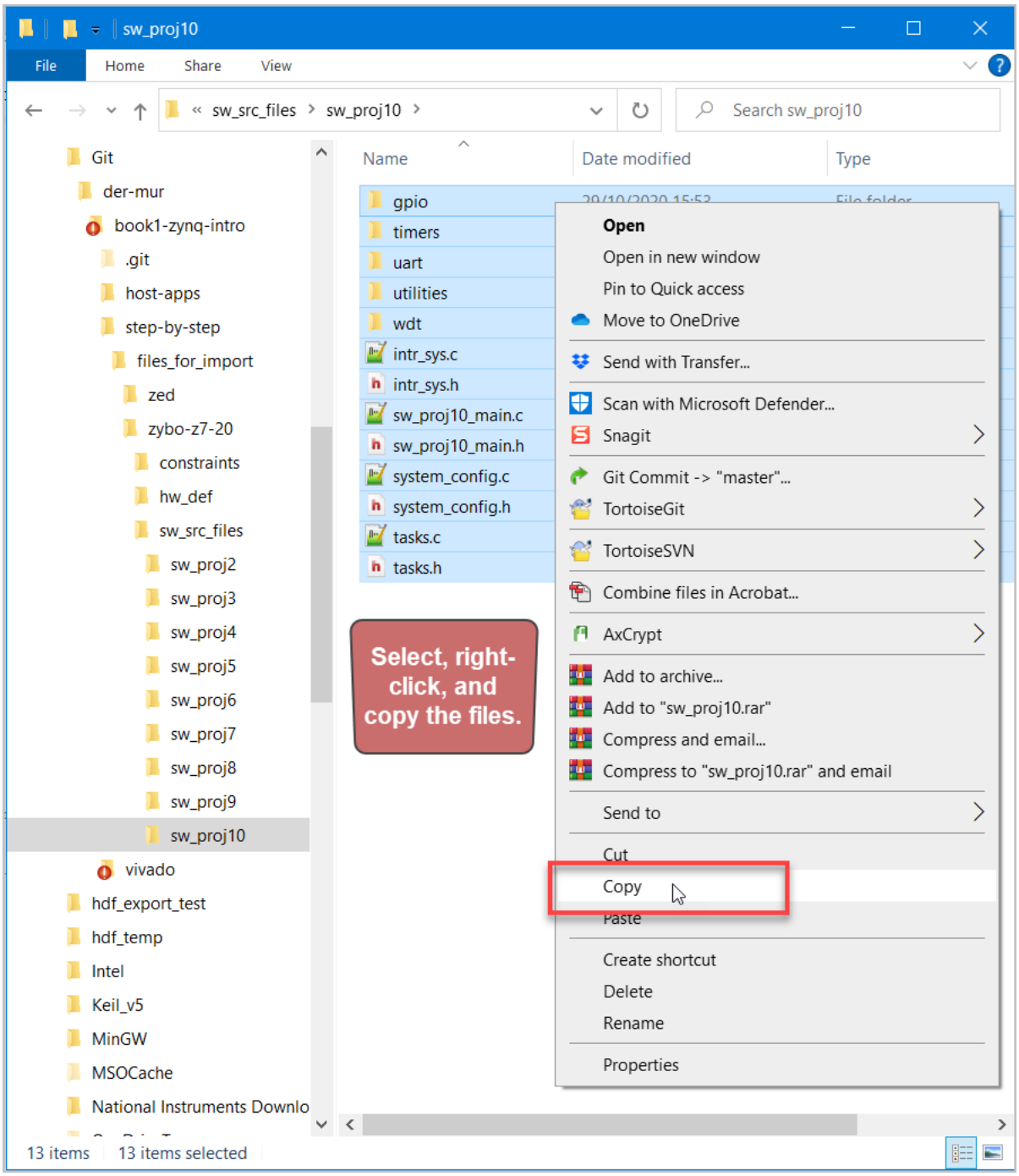


Figure 257. Select the project files, right-click, and select Copy.

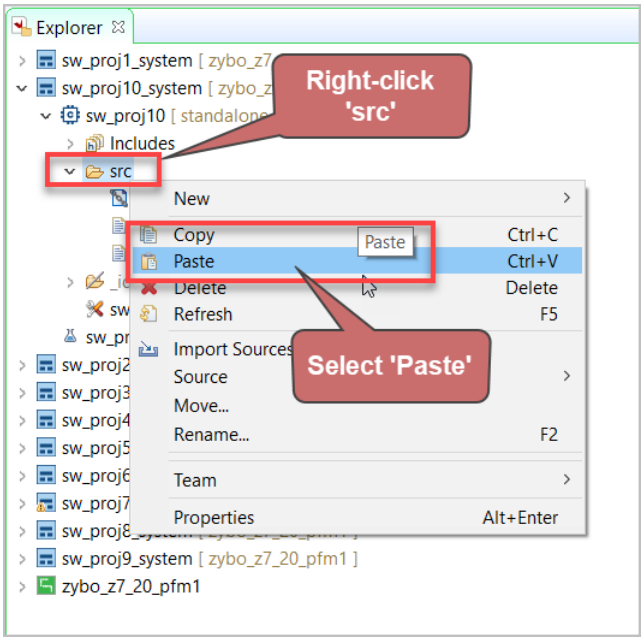


Figure 258. In Vitis, right-click the 'src' directory, and paste the files.

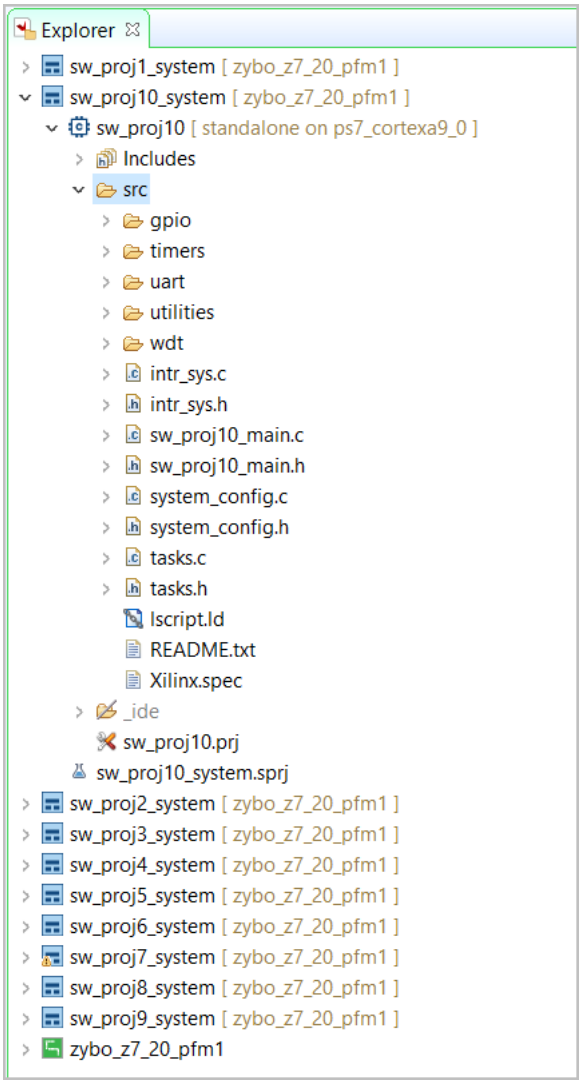


Figure 259. The project files are added.

3.13.2 Build the Project

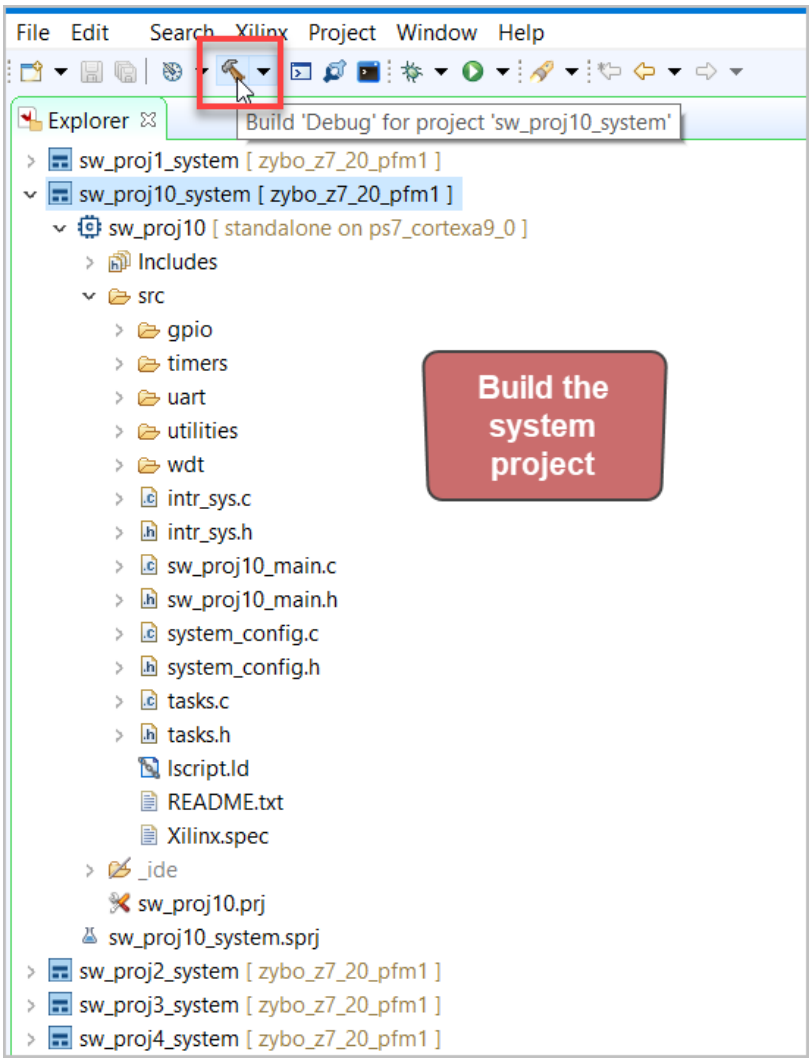


Figure 260. Build the system project

3.13.3 Run the Project

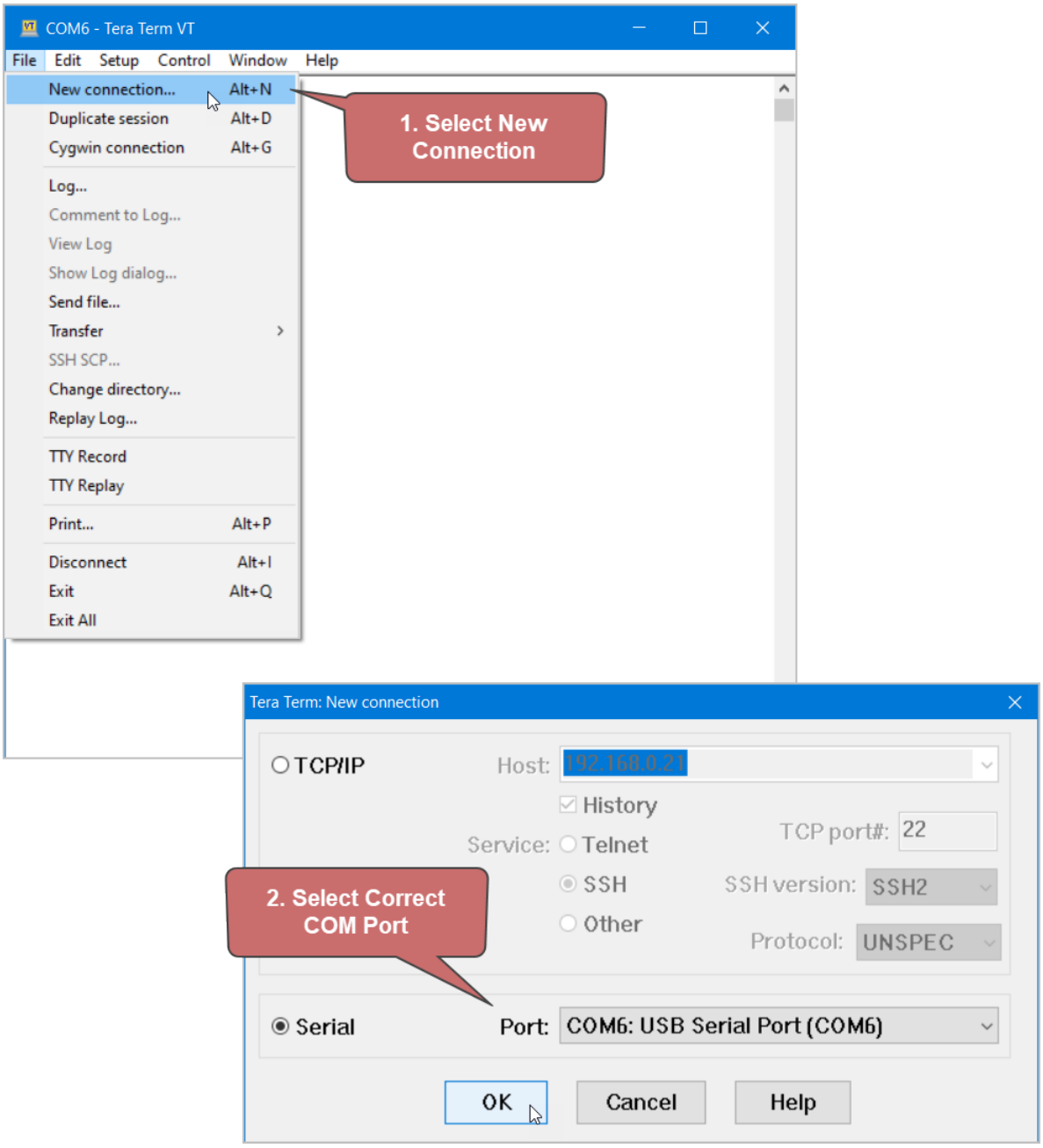


Figure 261. Reconnect terminal to platform if disconnected in last project.

To run the project, start by reconnecting the COM port if it was disconnected after Software Project 9.

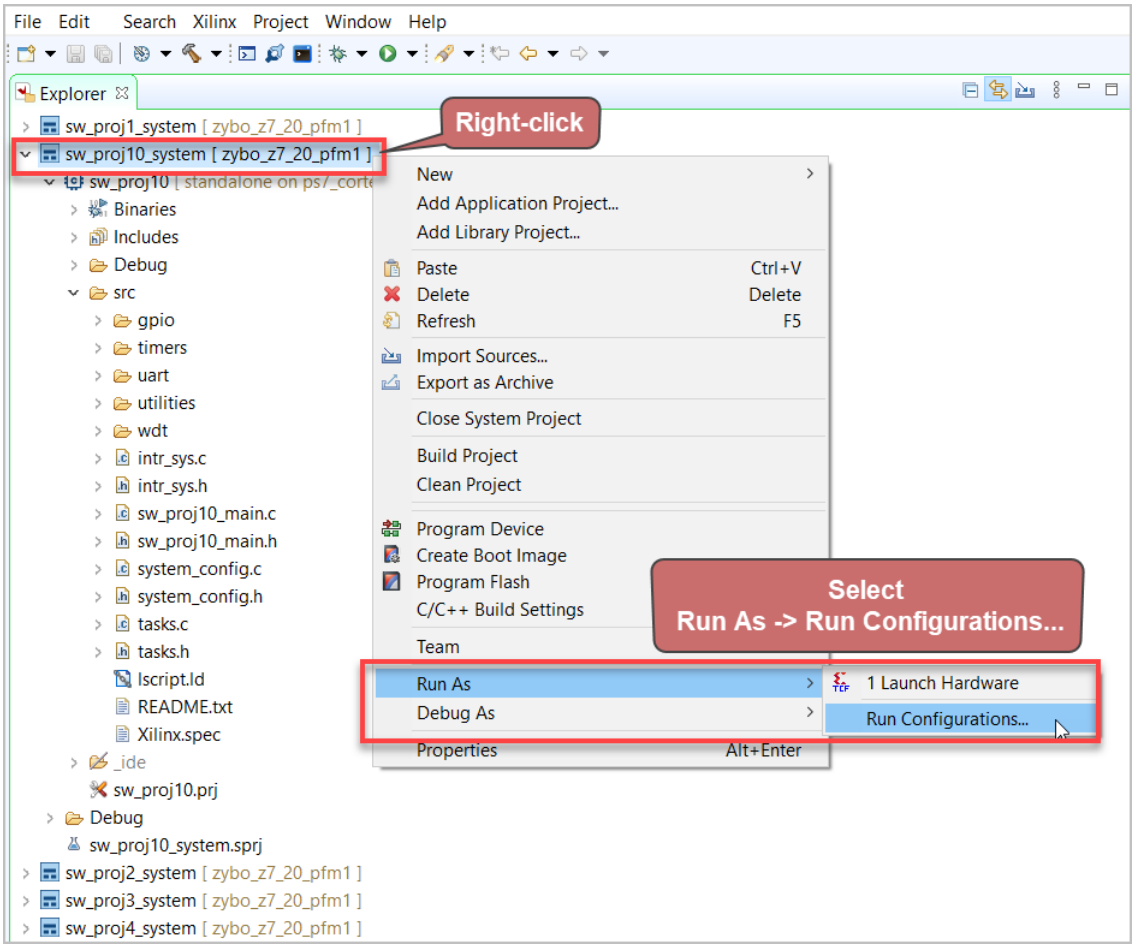


Figure 262. Right-click on the system project and select Run As -> Run Configurations.

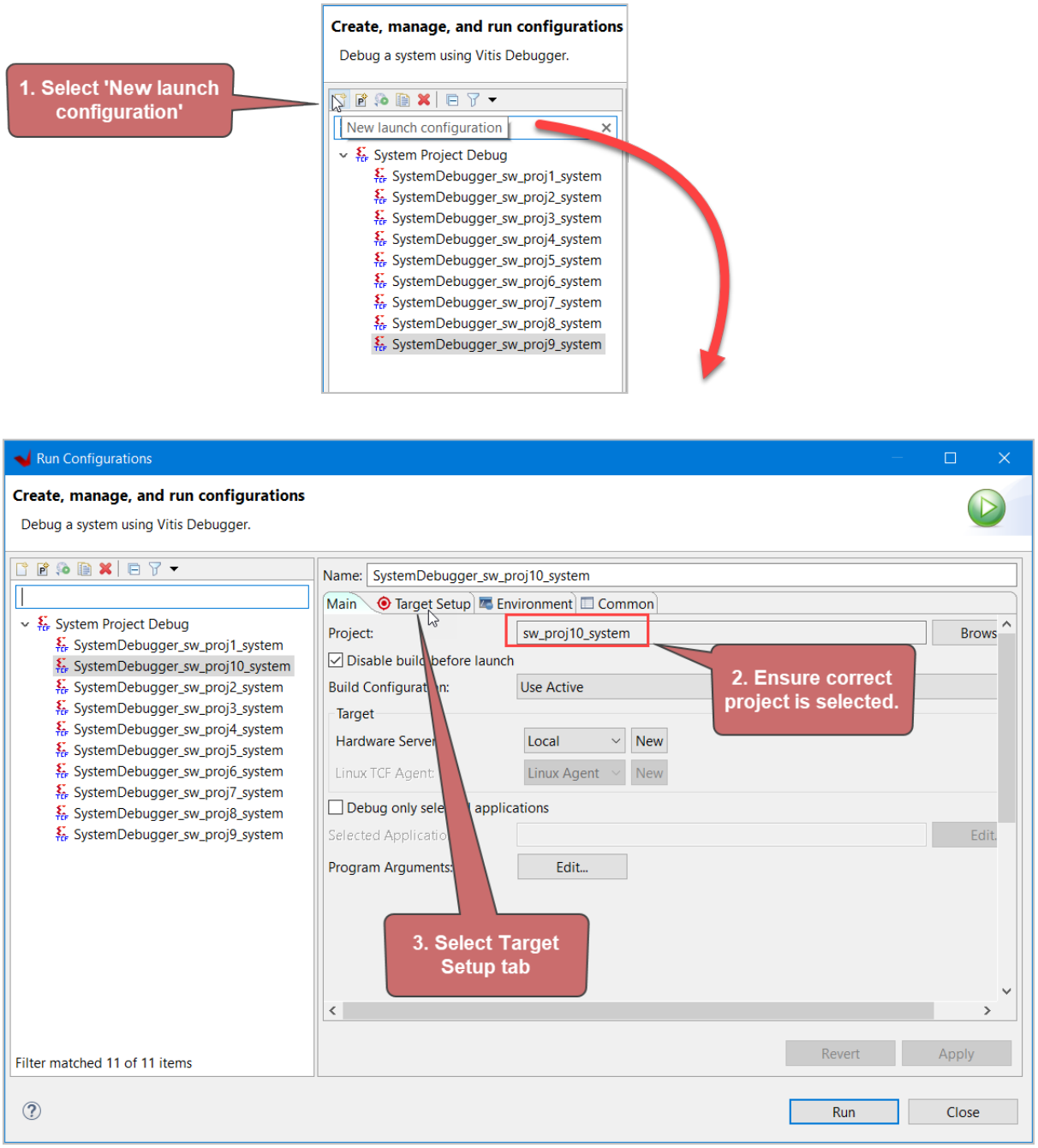


Figure 263. Create a New launch configuration

1. Click on 'New launch configuration'.
2. Ensure that "sw_proj10_system" is selected.
3. Select the Target Setup Tab.

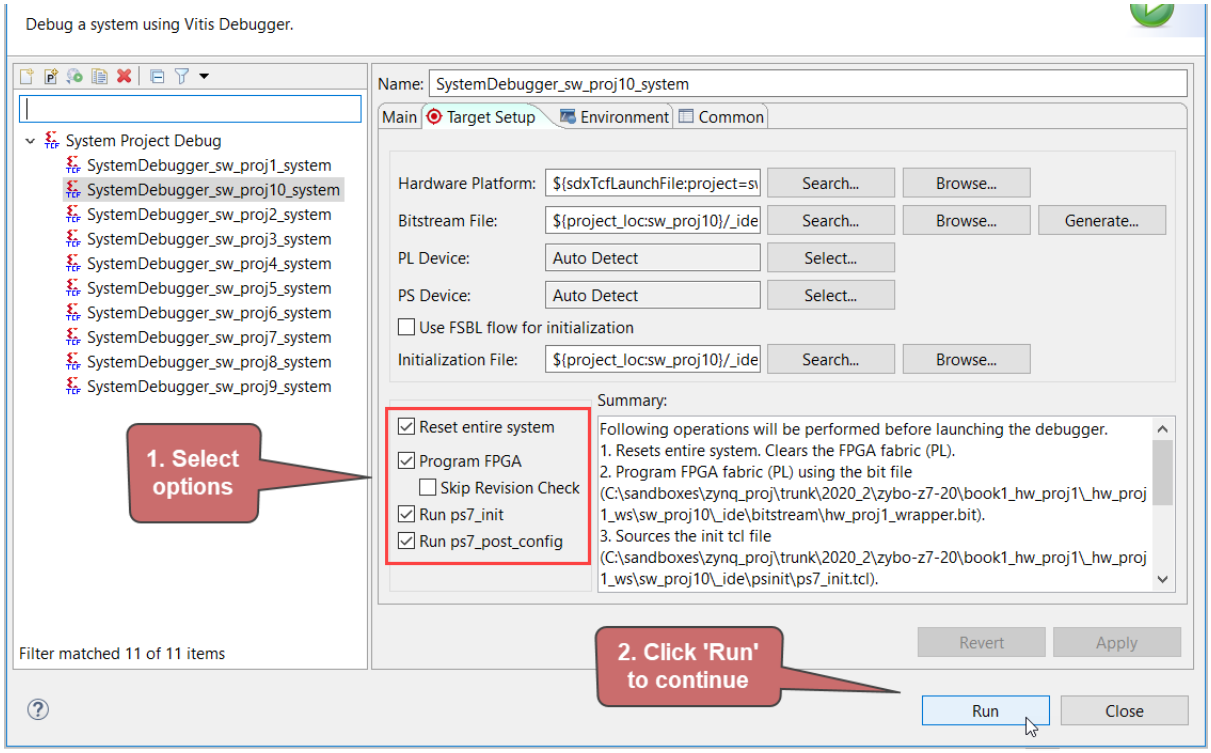


Figure 264. Run the application.

1. Select the main options as follows:
    a. Reset entire system
    b. Program FPGA
    c. Run ps7_init
    d. Run ps7_post_config

Click on 'Run' to continue. The FPGA should be programmed, and the ELF file should be downloaded to the platform, as shown in the next page.

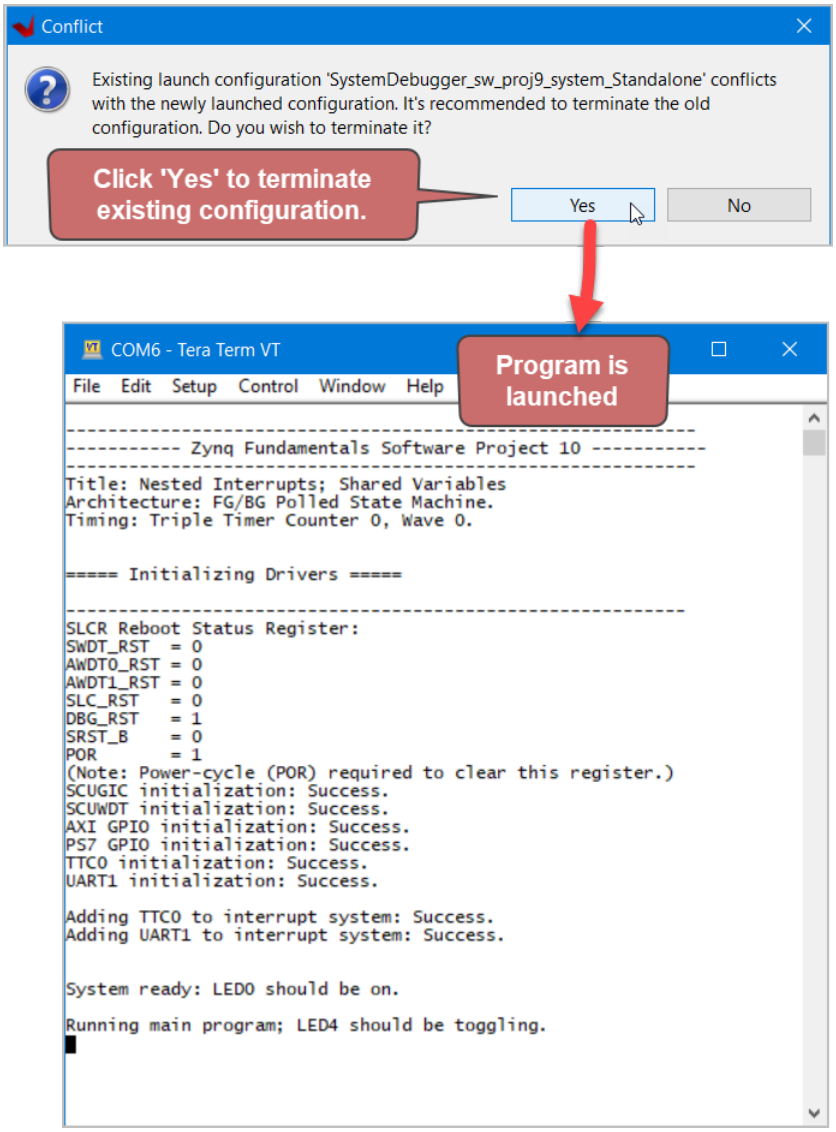


Figure 265. The project details for the software project should appear in the terminal.

If prompted, select 'Yes' to terminate any existing configuration. The FPGA will be programmed and the application will be downloaded to the processor. The terminal program should display the results for launching Software Project 10.

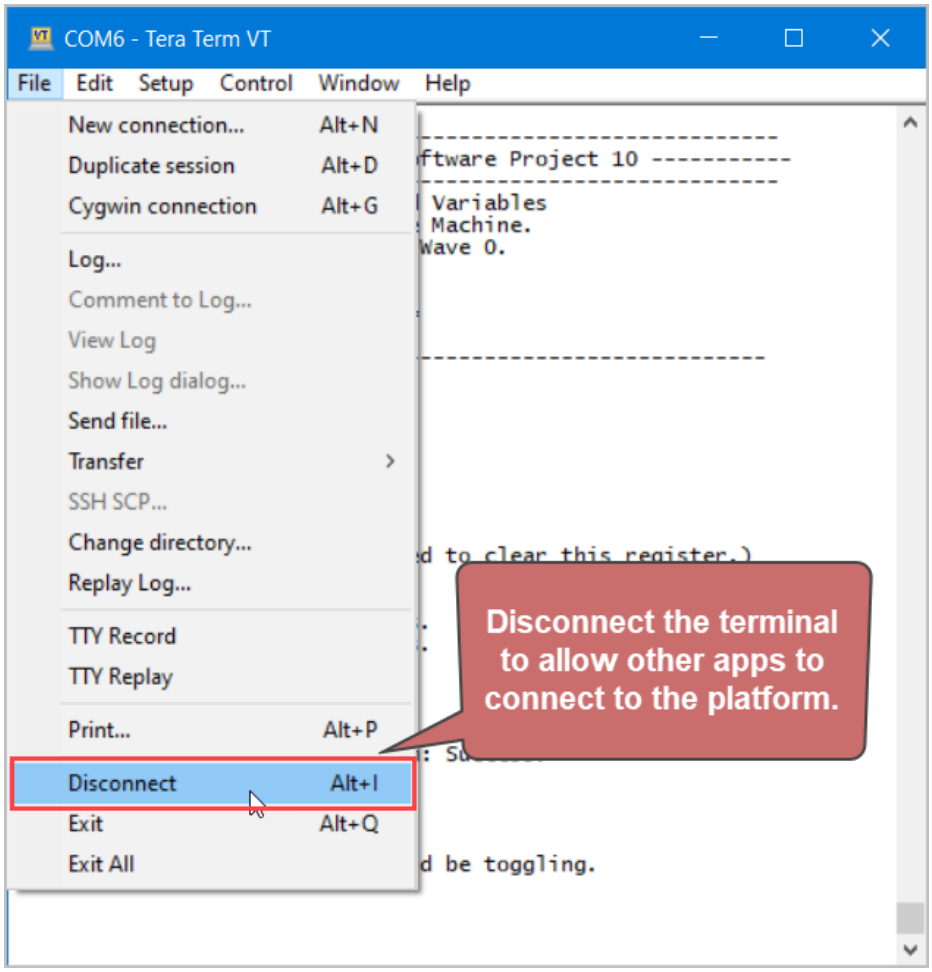


Figure 266. Disconnect the platform.

If the user wants to communicate with the command handler on the platform using a host application, the terminal needs to be disconnected.