



# Spartan-7 SP701 FPGA Demonstration



# Introduction

This step-by-step guide will show how to rapidly prototype an embedded system, using the Spartan-7 FPGA SP701 evaluation board.

We'll cover two applications using Vivado, a Xilinx tool for implementation and analysis of HDL and IP Integrator designs and Vitis, that enables the development of embedded software and accelerated applications on Xilinx platforms such as FPGAs, SoCs, and ACAPs.

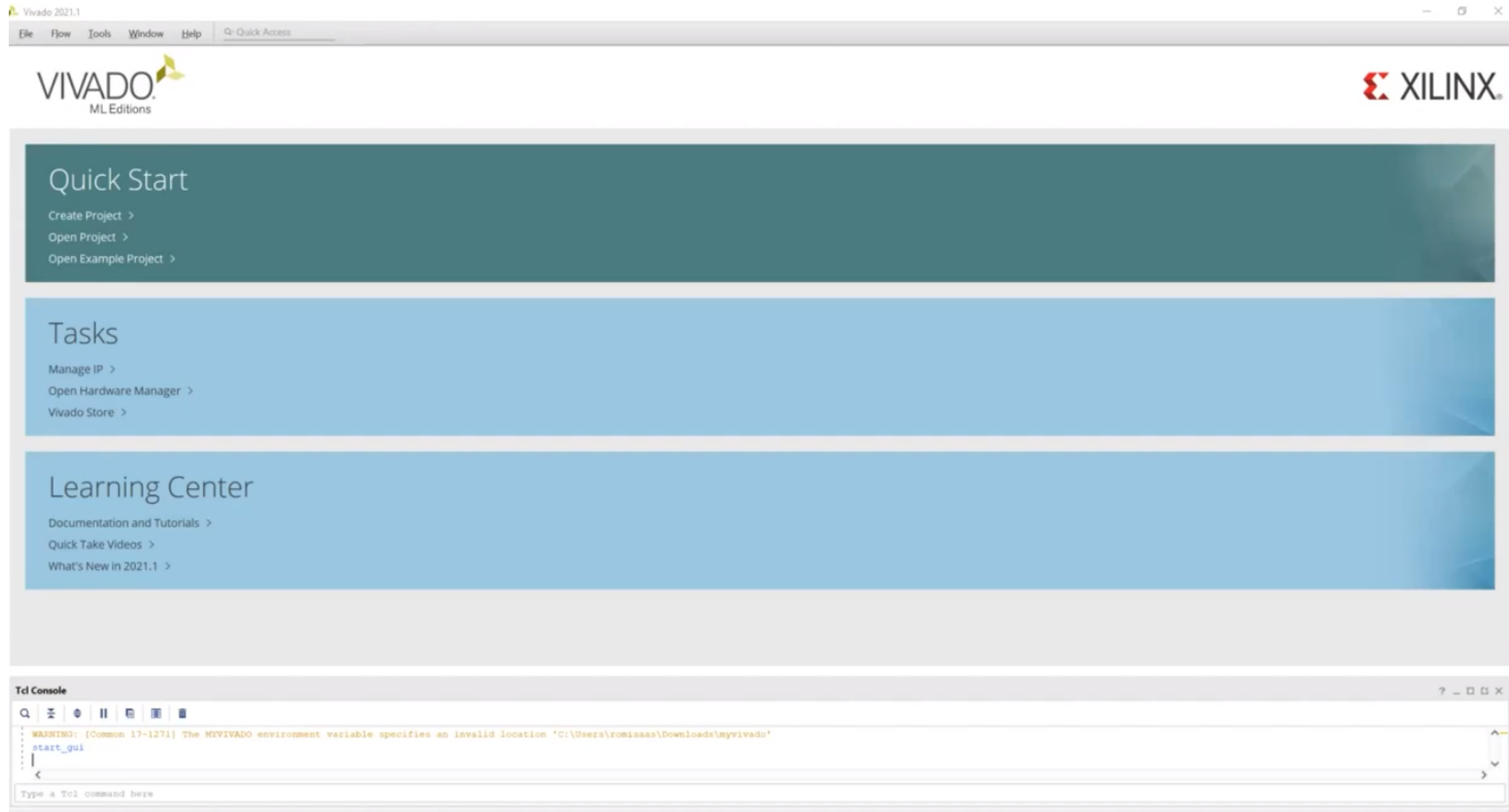
Applications are:

1. Hello World application using MicroBlaze.
2. Pulse Width Modulation (PWM) application.

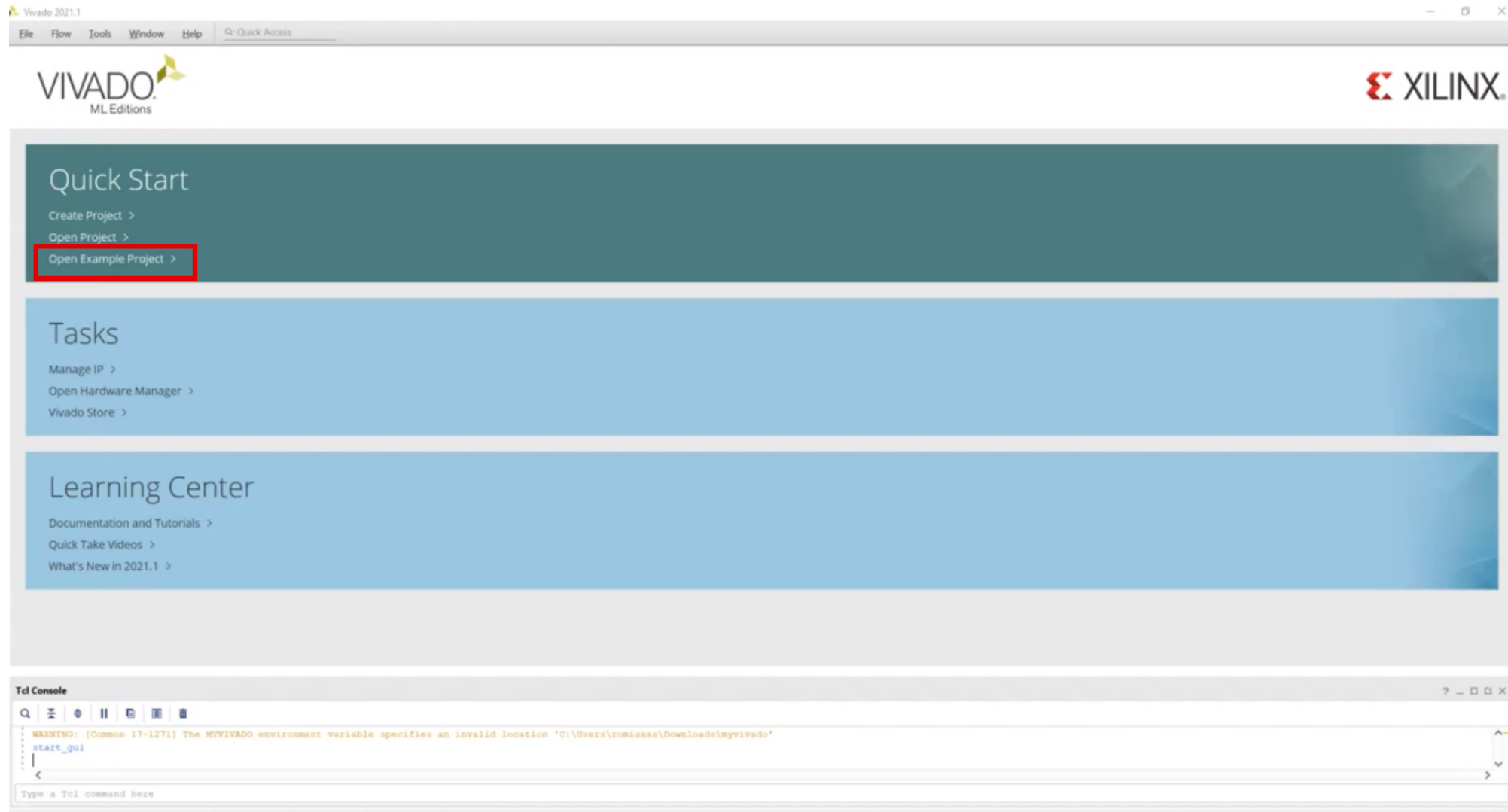


# Hello World application using MicroBlaze.

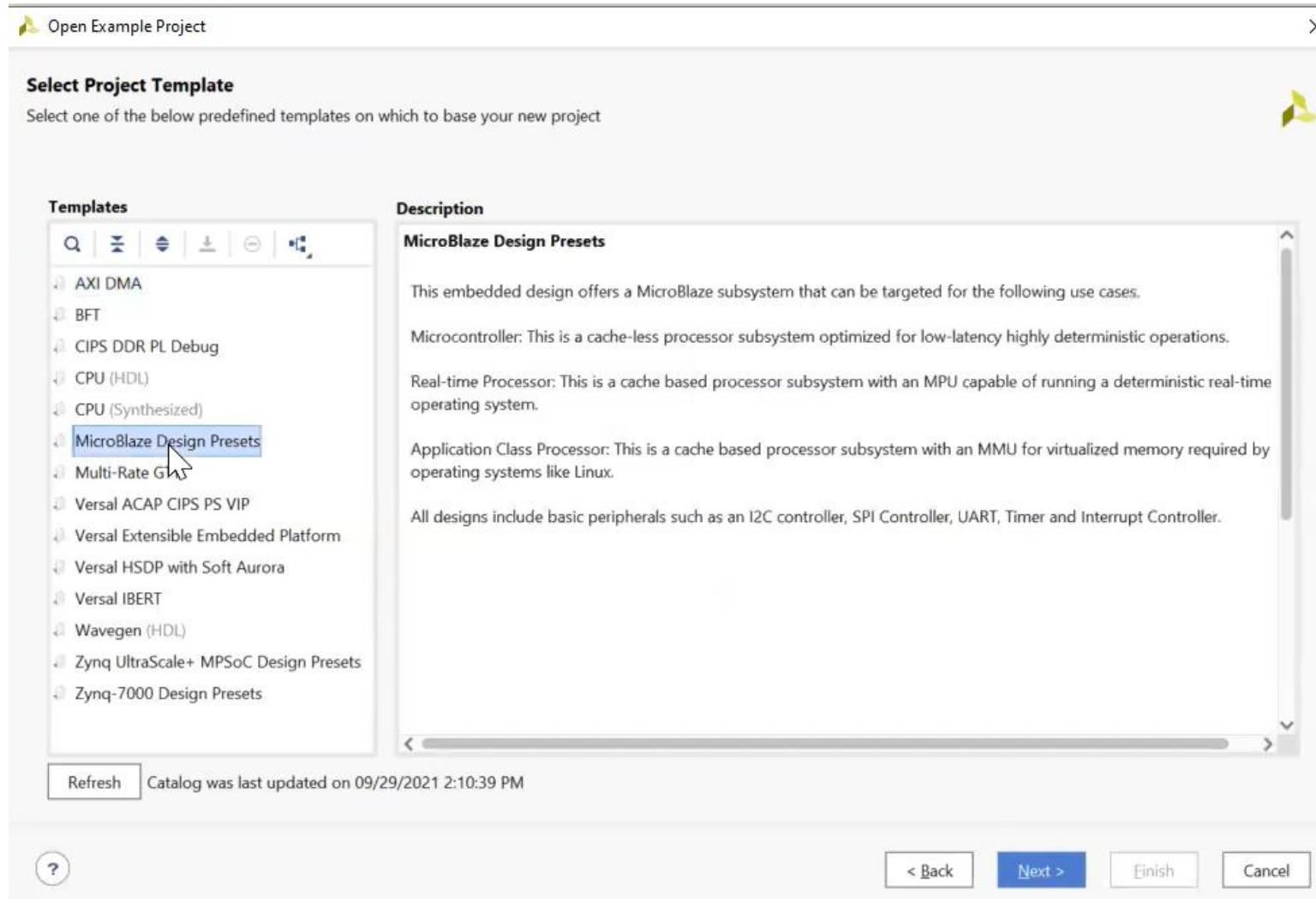
# Launch Vivado 2021.1



# On the Vivado Quick Start page, select “Open an example project”



# From the project templates, select “MicroBlaze Design Presets” and click "Next"



# Then enter the project name and location and click "Next"

Open Example Project

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

Project name: PWM\_MicroBlaze

Project location: C:/Xilinx/Projects

Create project subdirectory

Project will be created at: C:/Xilinx/Projects/PWM\_MicroBlaze









? < Back Next > Finish Cancel

# Select the board to target “SP701”

Open Example Project

**Default Part**  
Choose a default Xilinx board for your project.

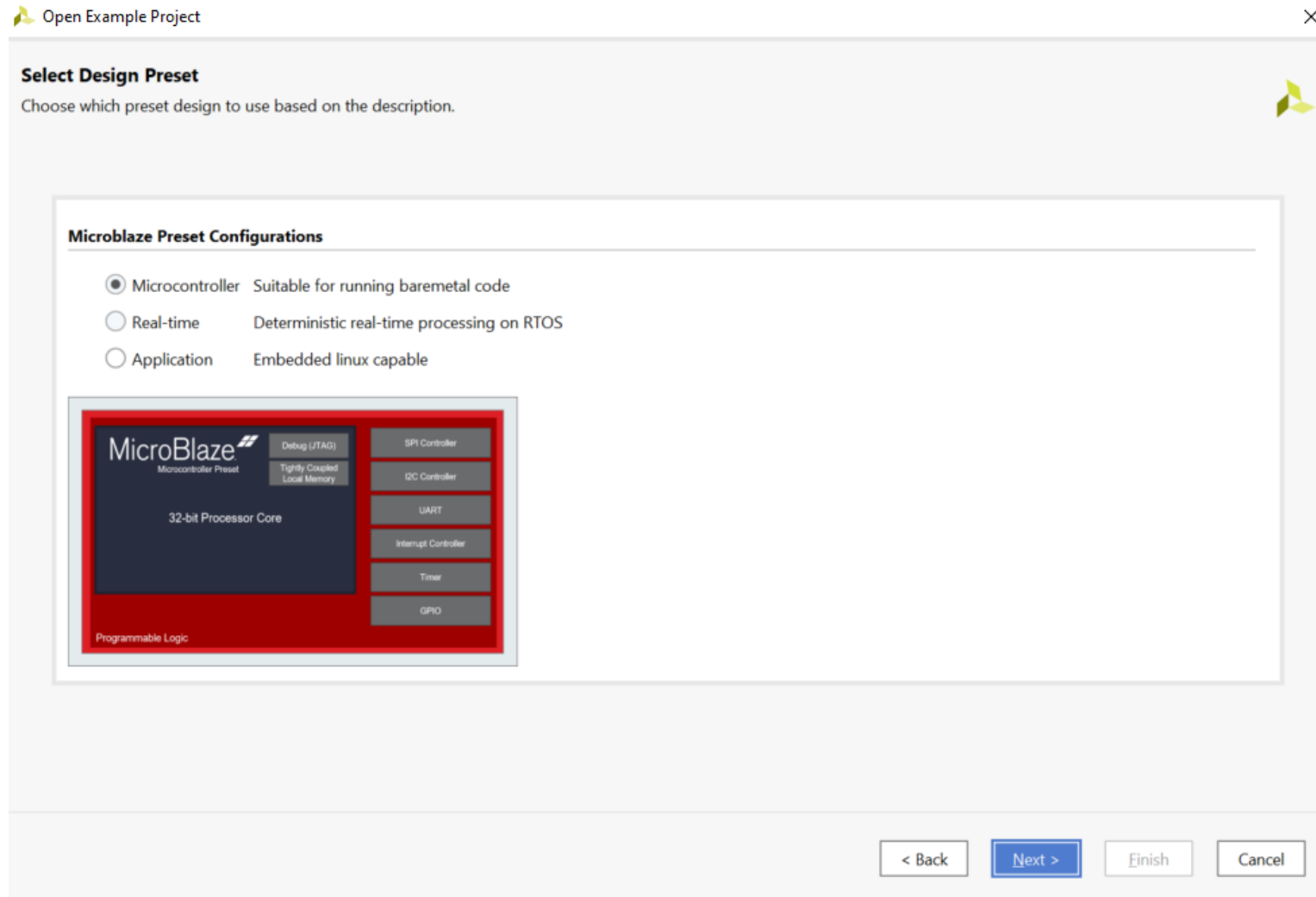
Search:

Display Name	Preview	Vendor	File Version	Part	I/O Pin Count	Board Rev	Available IOBs	LUT Elemer
Kintex-UltraScale KCU105 Evaluation Ple		xilinx.com	1.7	xcku040-ffva115...	1156	1.0	520	242400
Kintex UltraScale+ KCU116 Evaluation P		xilinx.com	1.5	xcku5p-ffvb676-...	676	1.0	280	216960
Virtex-UltraScale VCU108 Evaluation Pla		xilinx.com	1.7	xcvu095-ffva210...	2104	1.0	832	537600
Virtex-UltraScale VCU110 Evaluation Pla		xilinx.com	1.4	xcvu190-flgc210...	2104	1.0	416	1074240
Virtex UltraScale+ VCU118 Evaluation P		xilinx.com	2.4	xcvu9p-flga2104-...	2104	2.0	832	1182240
Versal VCK190 Evaluation Platform		xilinx.com	2.2	xcvc1902-vsva21...	2197	Rev B02	692	899840
Versal VMK180 Evaluation Platform		xilinx.com	2.2	xcvm1802-vsva2...	2197	Rev B02	692	899840
<b>Spartan-7 SP701 Evaluation Platform</b>		xilinx.com	1.1	xc7s100fgga676-2	676	1.0	400	64000

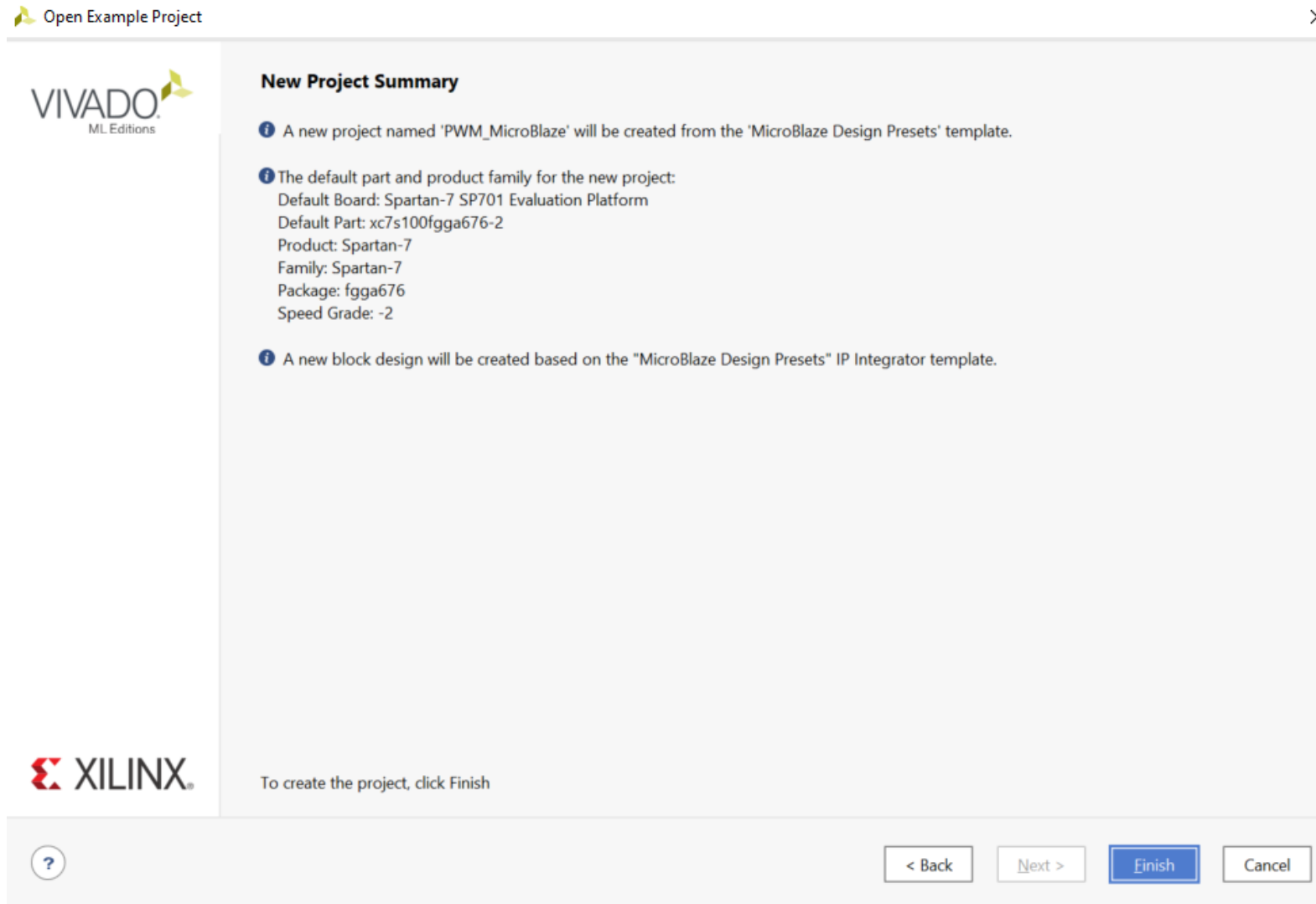
? < Back Next > Finish Cancel



# Select “Microcontroller design preset” in the MicroBlaze Configuration selection



# Review changes in the Project Summary and click "Finish"



The screenshot shows the 'New Project Summary' dialog box in Vivado ML Editions. The window title is 'Open Example Project'. The Vivado ML Editions logo is in the top left. The main content area is titled 'New Project Summary' and contains three informational messages:

- A new project named 'PWM\_MicroBlaze' will be created from the 'MicroBlaze Design Presets' template.
- The default part and product family for the new project:
  - Default Board: Spartan-7 SP701 Evaluation Platform
  - Default Part: xc7s100fgga676-2
  - Product: Spartan-7
  - Family: Spartan-7
  - Package: fgga676
  - Speed Grade: -2
- A new block design will be created based on the "MicroBlaze Design Presets" IP Integrator template.

At the bottom left, the Xilinx logo is displayed. Below it, the text reads 'To create the project, click Finish'. At the bottom right, there are four buttons: '< Back', 'Next >', 'Finish' (highlighted in blue), and 'Cancel'. A help icon (?) is located at the bottom left of the dialog box.

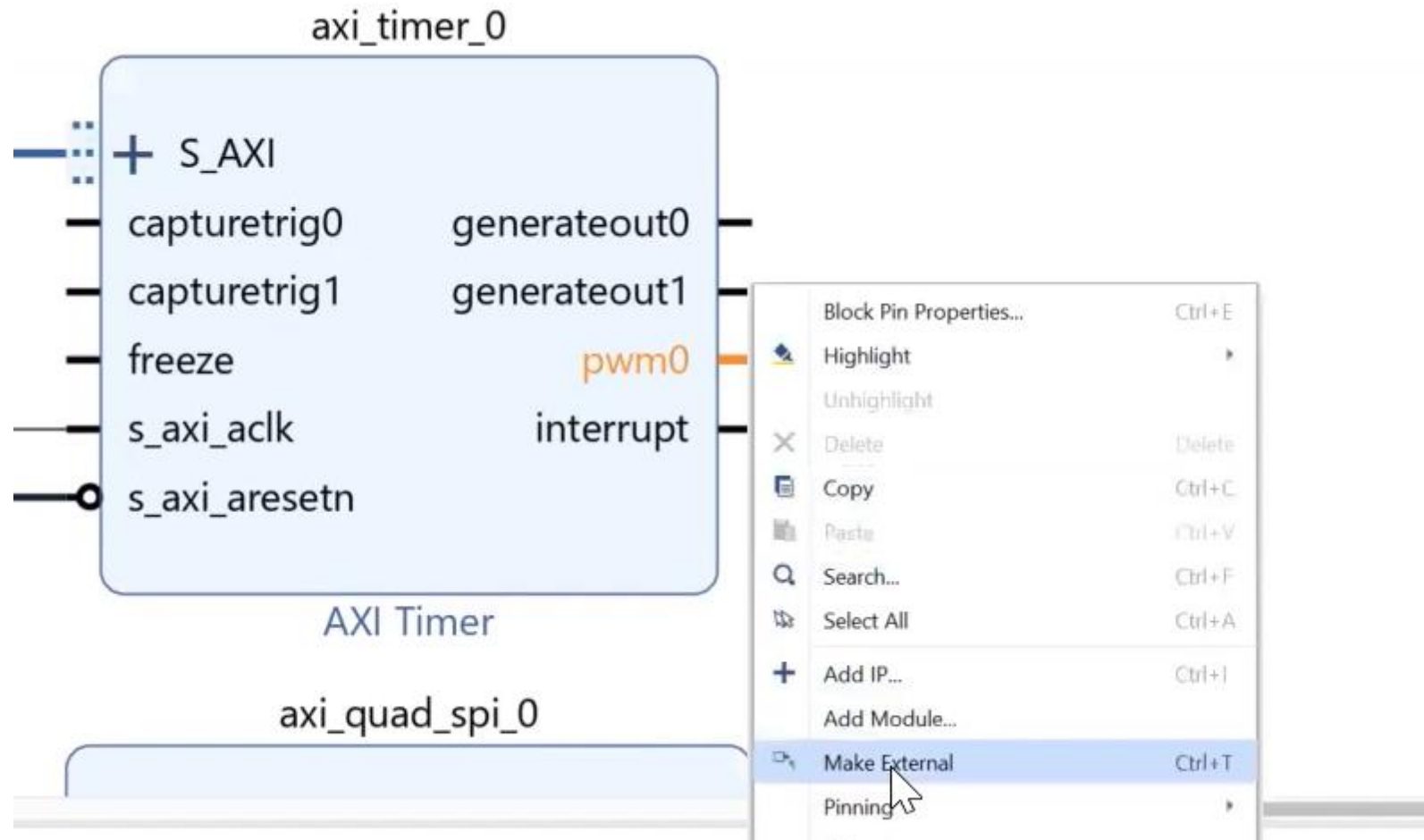
# Windows Layout

The Microblaze design will load, and a Diagram window will appear on the right, where we can see all the IPs in the design including MicroBlaze IP, AXI Uartlite, AXI GPIO, and AXI Timer, which we will use to generate a PWM signal.

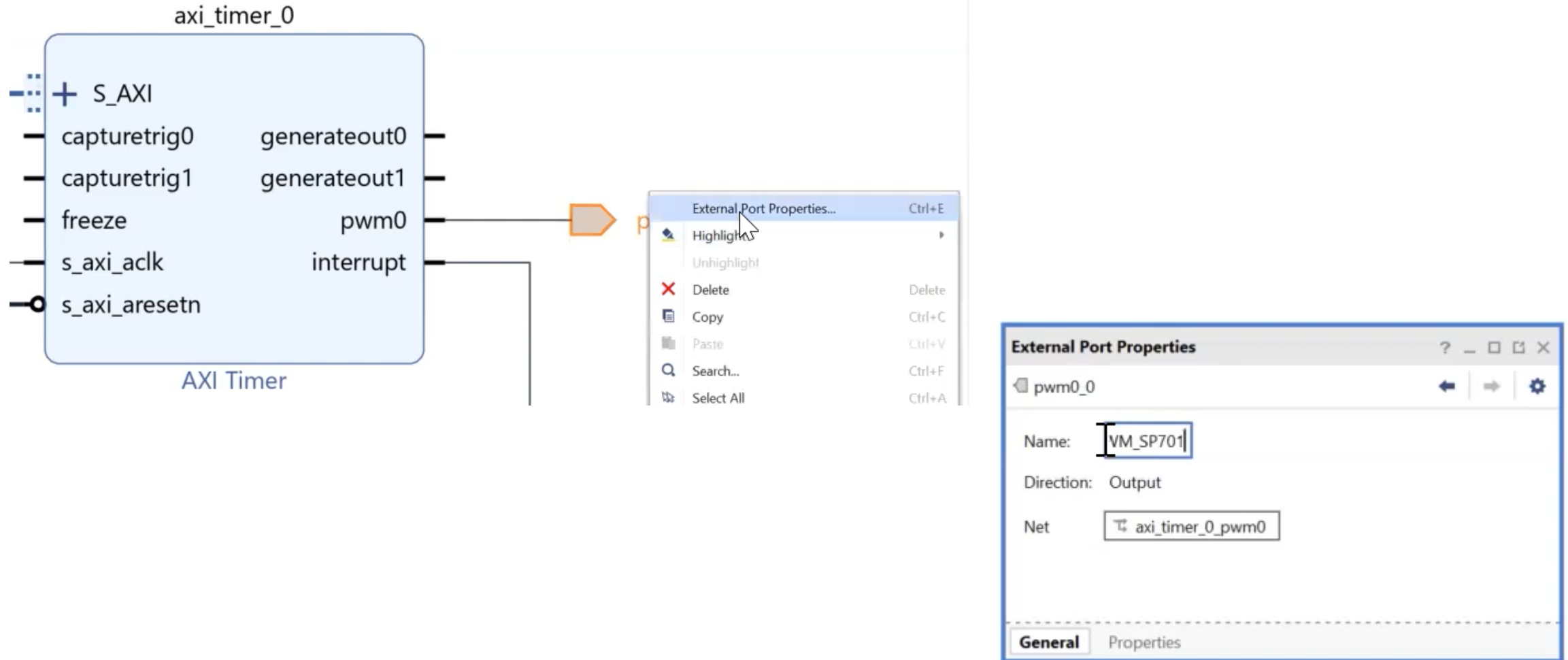
The screenshot displays the Xilinx Vivado IDE interface for a project named "mb\_preset". The left sidebar shows the "IP INTEGRATOR" section with a list of components: axi\_gpio\_0 (AXI GPIO:2.0), axi\_gpio\_1 (AXI GPIO:2.0), axi\_ic\_0 (AXI IIC:2.1), axi\_quad\_spi\_0 (AXI Quad SPI:3.2), axi\_timer\_0 (AXI Timer:2.0), and axi\_uartlite\_0 (AXI Uartlite:2.0). The central "Design" window shows a block diagram of the Microblaze system, including the MicroBlaze processor, local memory, and various peripheral blocks like AXI GPIO, AXI Timer, and AXI Uartlite. The right "Diagram" window provides a detailed view of the MicroBlaze processor and its connections to the peripheral blocks. The bottom "Tcl Console" window shows the following output:

```
slave segment '/axi_gpio_0/s_AXI/Reg' is being assigned into address space '/microblaze_0/Data' at <0x4000_0000 [ 64K ]>.
slave segment '/axi_gpio_1/s_AXI/Reg' is being assigned into address space '/microblaze_0/Data' at <0x4001_0000 [ 64K ]>.
Wrote : <C:\Xilinx\Projects\FW_MicroBlaze.src\sources_1\bd\mb_preset\mb_preset.bd>
VHDL Output written to : C:\Xilinx\Projects\FW_MicroBlaze.gen\sources_1\bd\mb_preset\synth\mb_preset.vhd
VHDL Output written to : C:\Xilinx\Projects\FW_MicroBlaze.gen\sources_1\bd\mb_preset\sim\mb_preset.vhd
make_wrapper: Time (s): cpu = 00:00:04 ; elapsed = 00:00:06 . Memory (MB): peak = 1611.230 ; gain = 169.117
instantiate_example_design: Time (s): cpu = 00:00:24 ; elapsed = 00:00:27 . Memory (MB): peak = 1611.230 ; gain = 350.047
update_compile_order -fileset sources_1
```

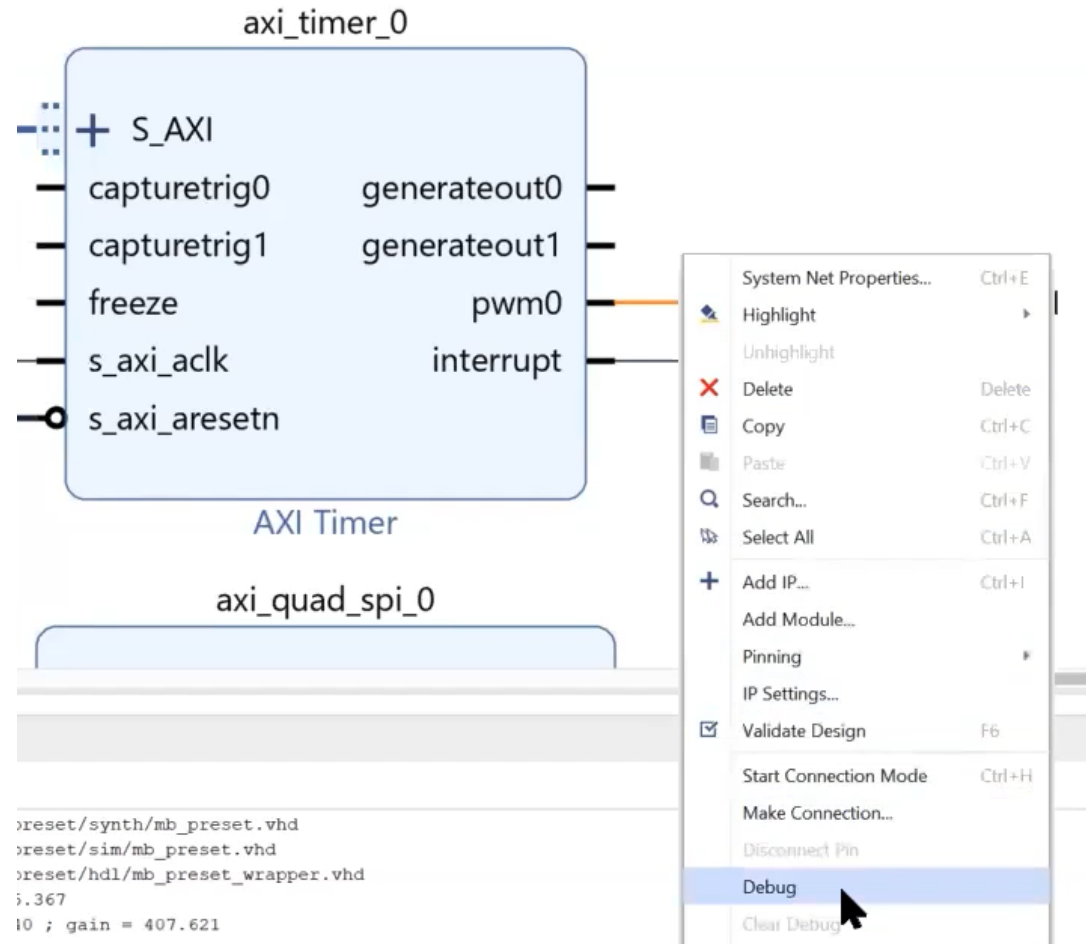
# Connect "PWM" pin to external interface



# Rename the external port “pwm0\_0” by changing the name under the “External Port Properties” to “SP701\_PWM”

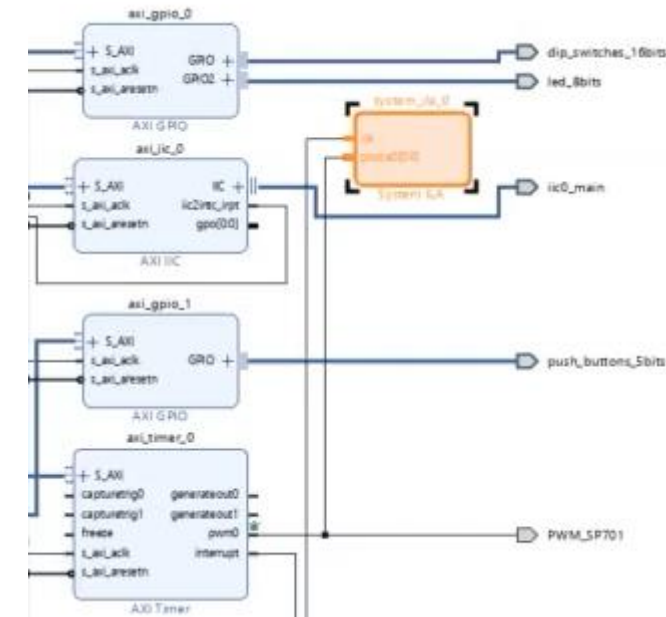
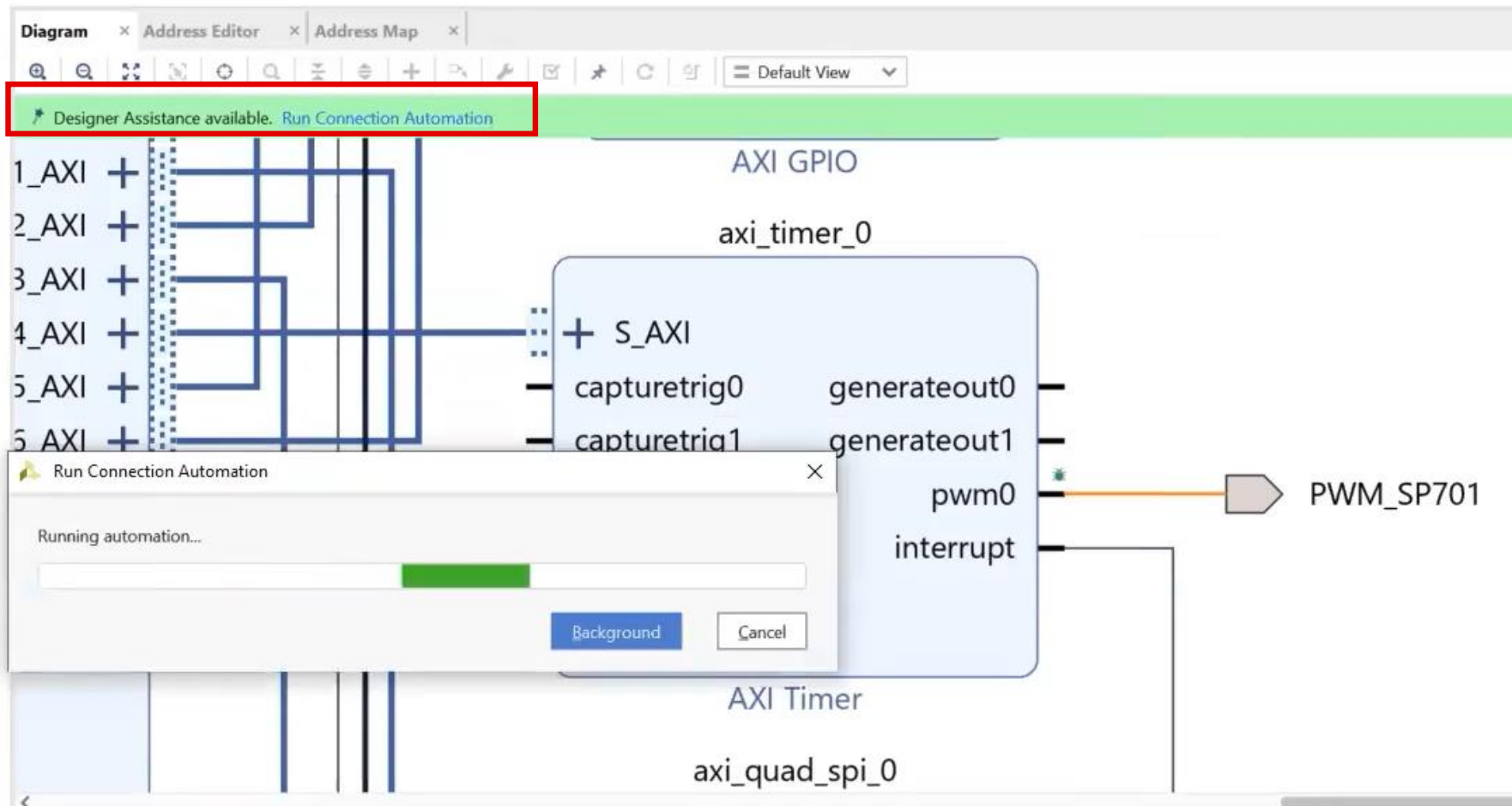


Select the wire between “pwm0” and “SP701\_PWM” by right clicking and selecting “Debug” to monitor the PWM of the Timer

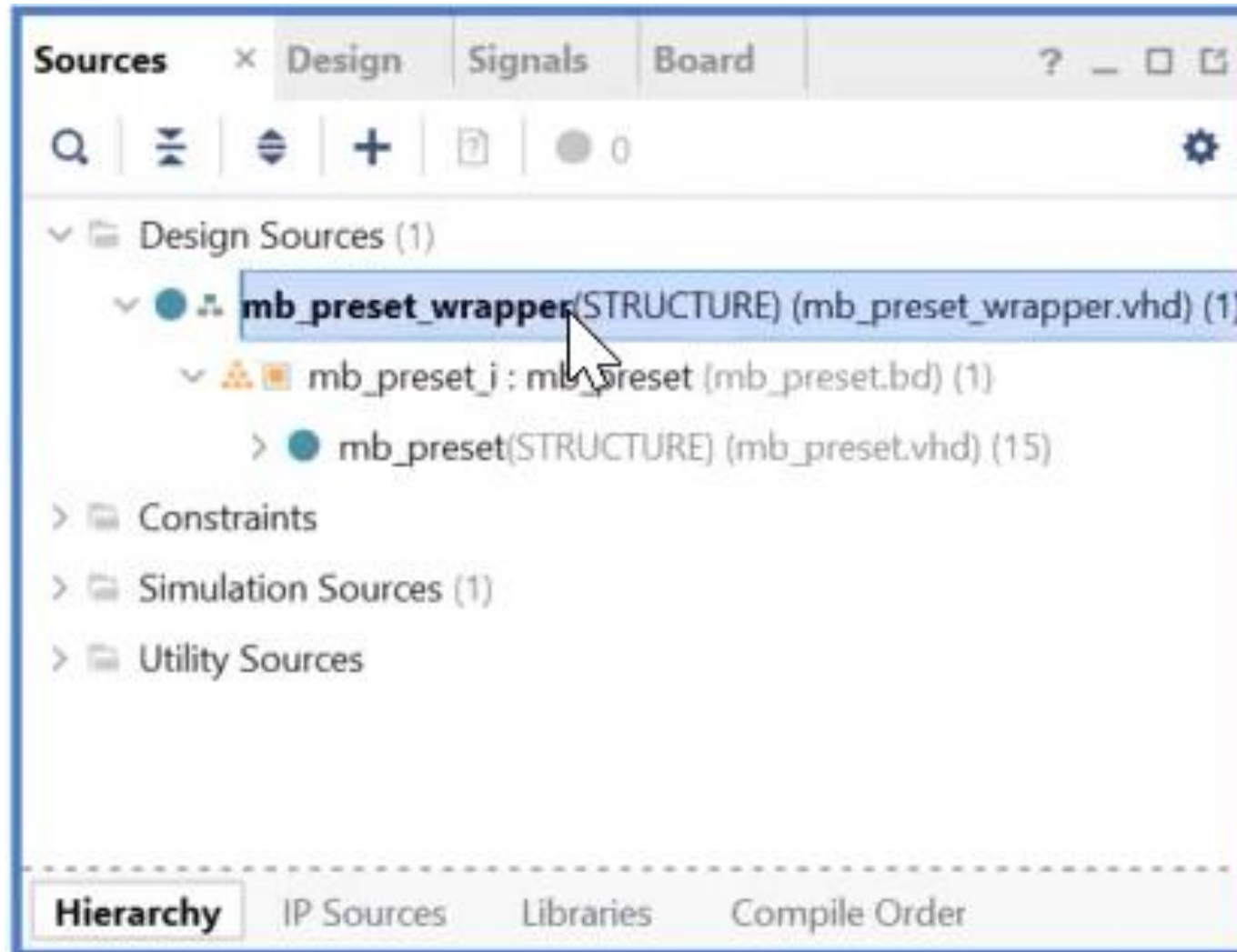


# Use the Designer Assistance and select “Run Connection Automation”, then click “OK”

This step will add System ILA to the design connected to the PWM

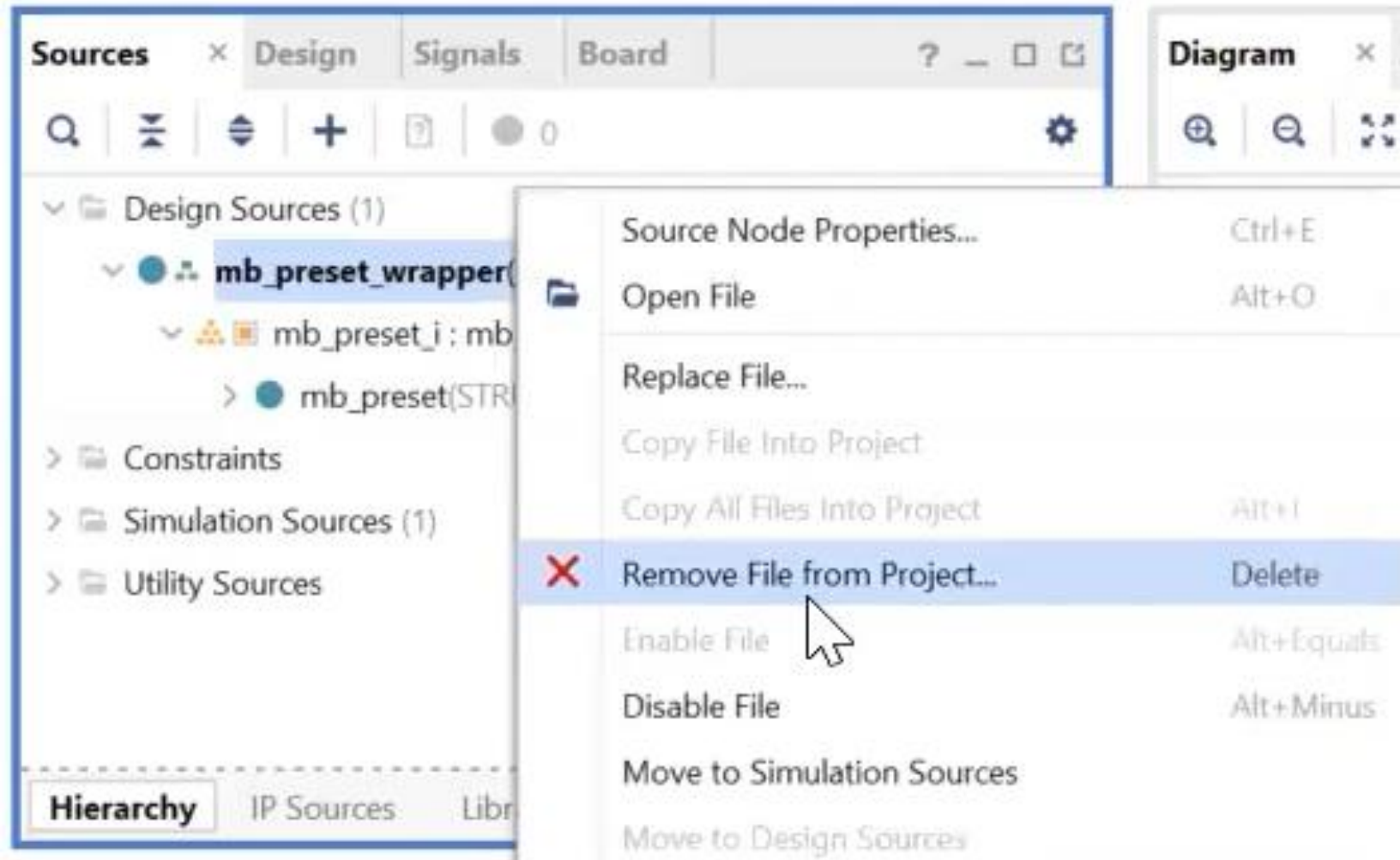


Modify the preset example design by creating a new wrapper, select “mb\_preset\_wrapper” under Design Sources in the Sources window

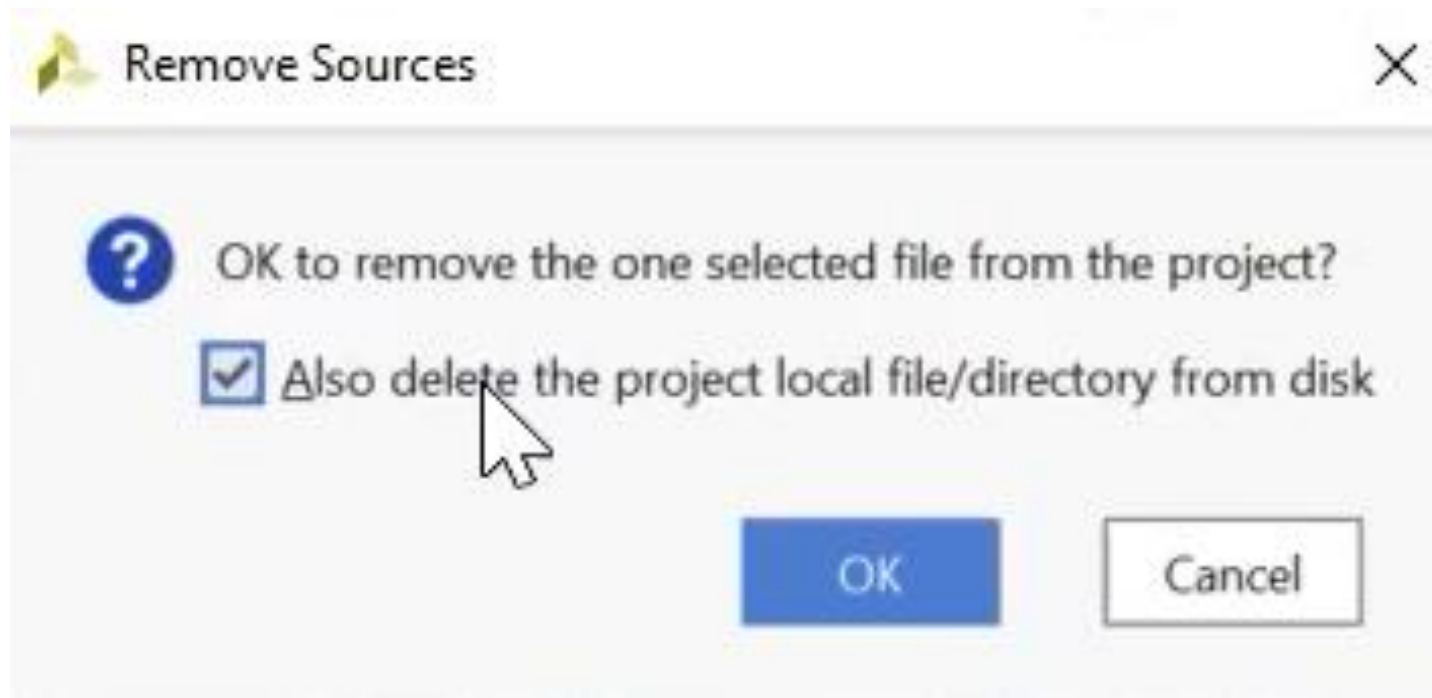




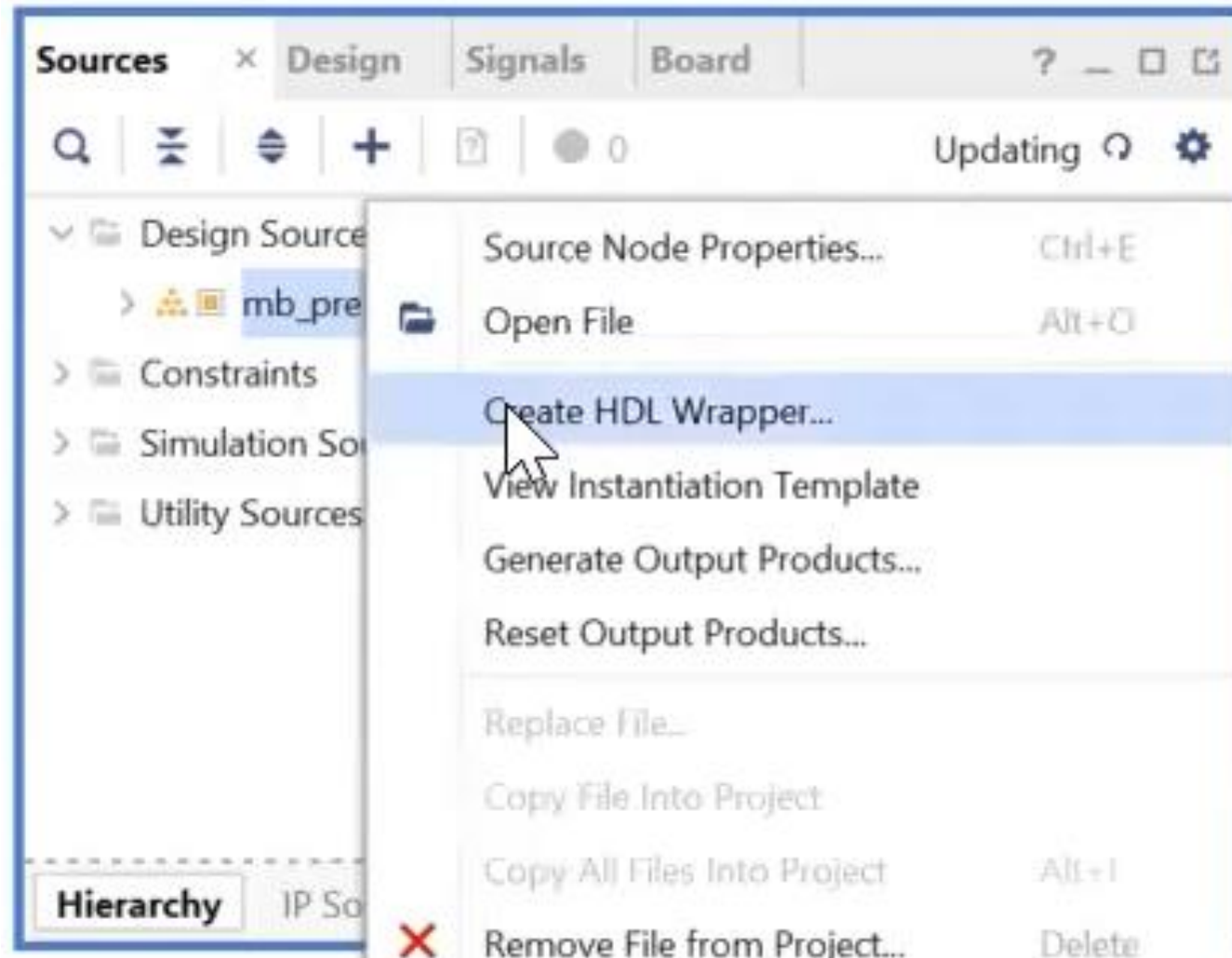
Right-click on it and select “Remove file from the project”



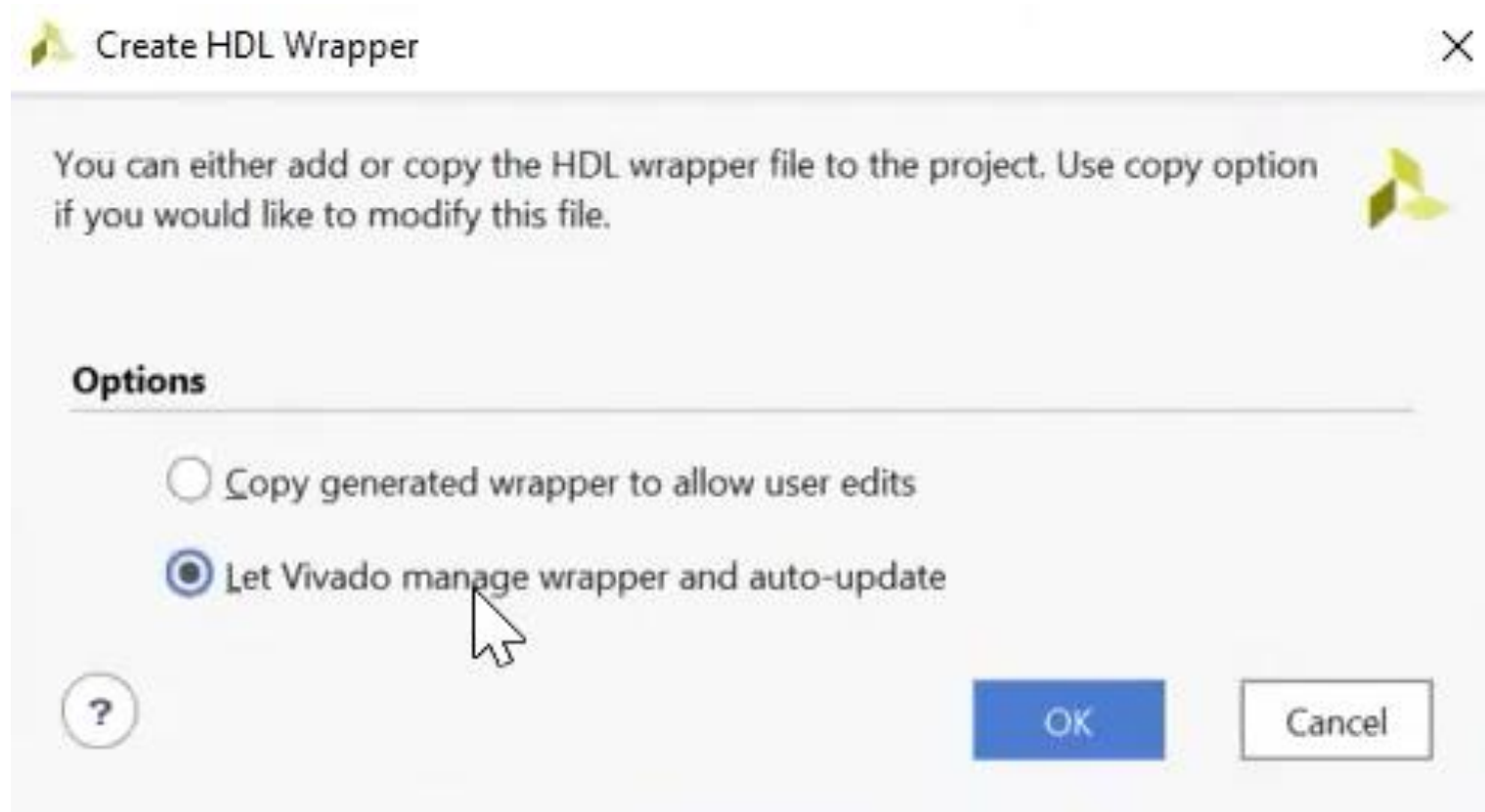
The “Remove Sources” window will appear and check the box for “Also delete the project local file/directory from the disk”



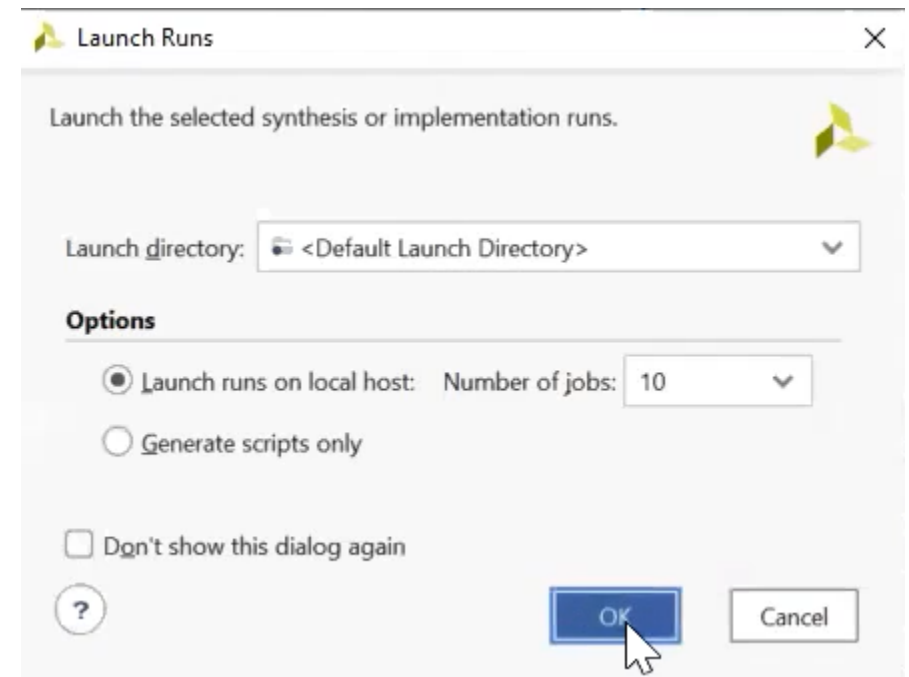
Select “mb\_preset” design under the design sources window and Right-click then select “Create HDL Wrapper”



# Select “Let Vivado manage wrapper and auto-update”

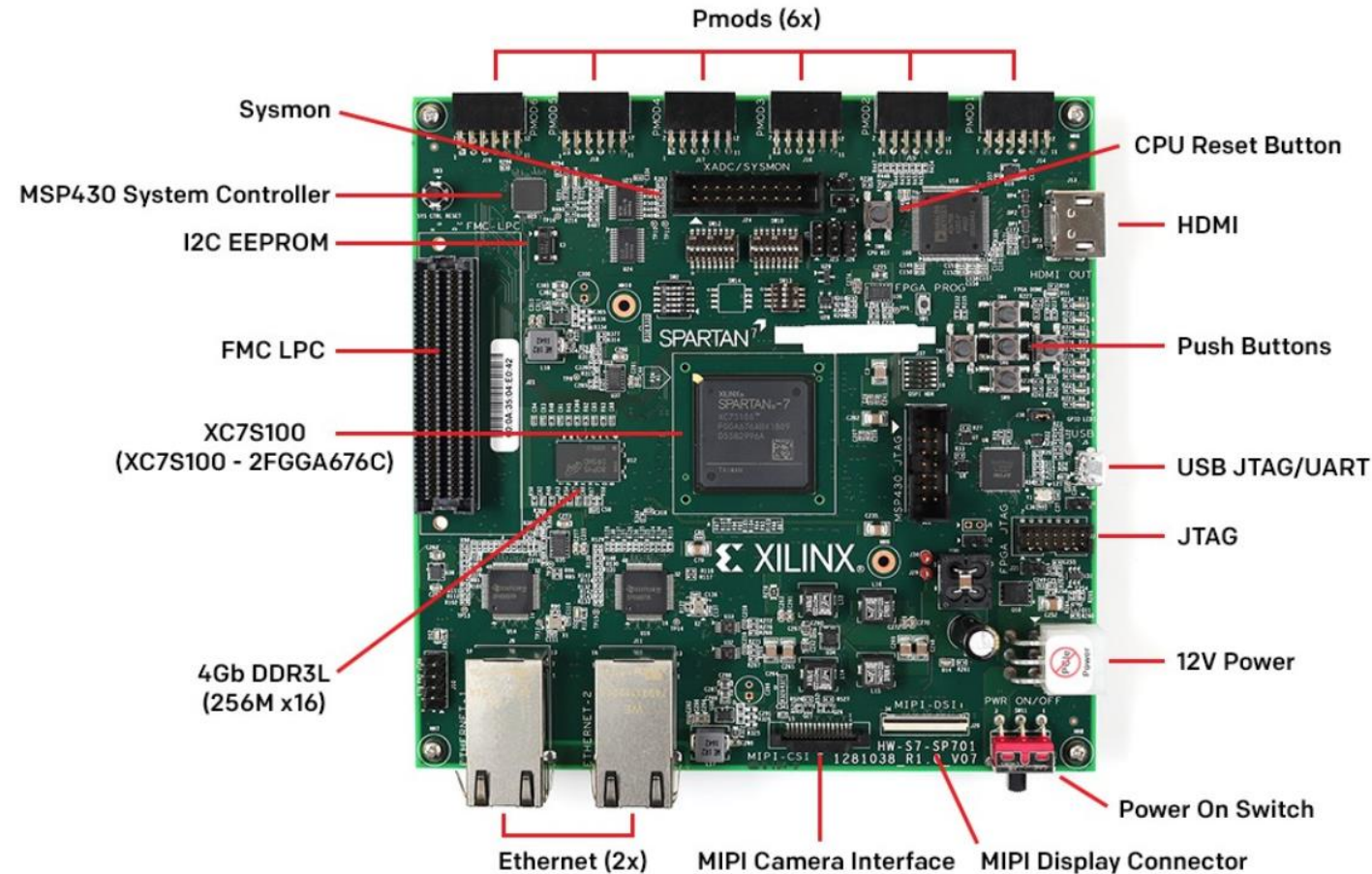


# In the Flow Navigator under Synthesis, click “Run Synthesis” and then Click "OK"



# Connecting “PWM” signal to board interface

The SP701 Board includes 6 Pmods that we can use and assign the PWM signal to



# Connecting “PWM” signal to board interface

1. Go to Spartan-7 [SP701 FPGA Evaluation Kit](#) webpage
2. Under Resources download the [SP701 Schematics](#)
3. Open the xdc file and search for PMOD1\_PIN1
4. You’ll find that it is connected to pin “C13”
5. In the next step we will assign PWM to pin C13



XTP553 - SP701 Schematics (v1.0)

Document Type: Board Files

[See All Versions](#)

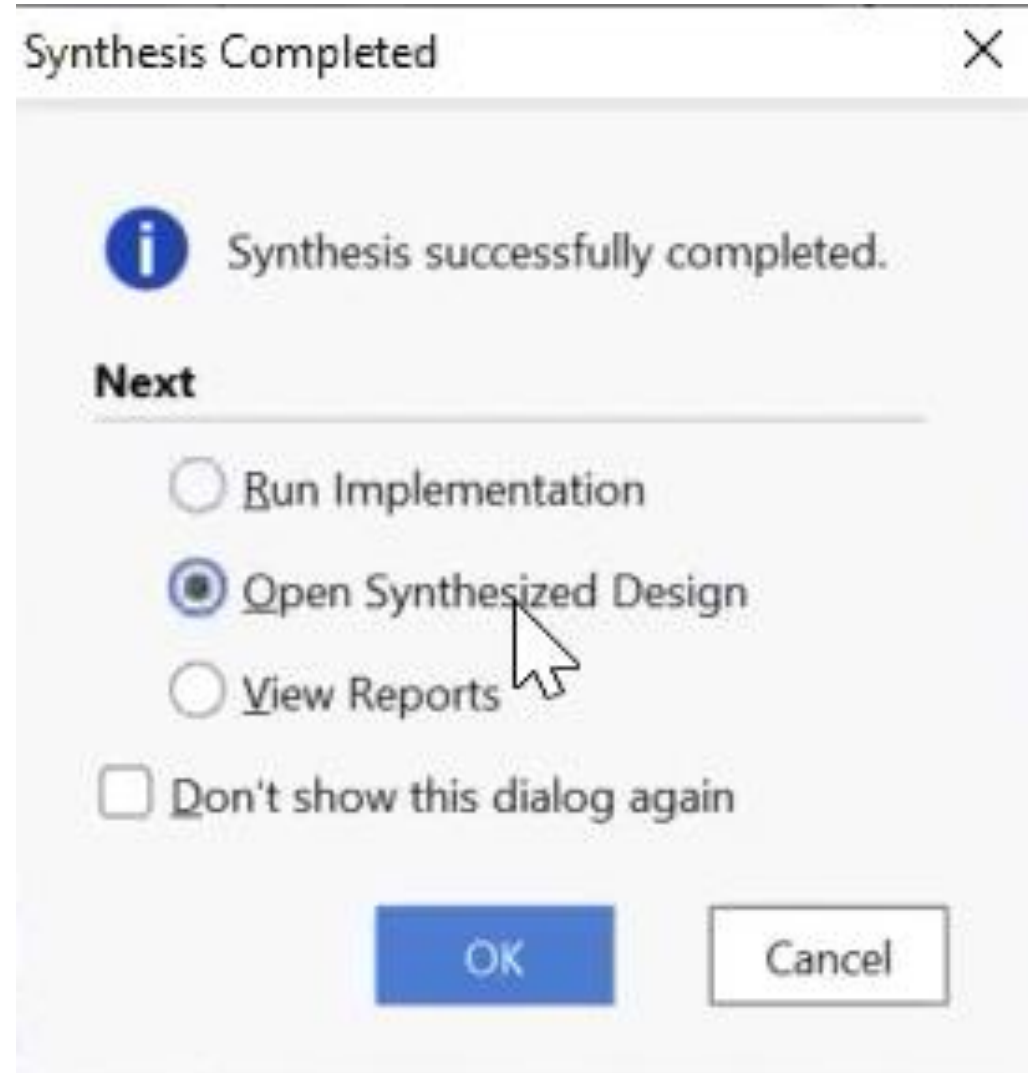
Design File(s):

- [rdf0508-sp701-schematic-source.zip](#)
- [rdf0509-sp701-bom.zip](#)
- [rdf0510-sp701-xdc.zip](#)

```
sp701_rev1.0_U1.xdc
```

346	set_property	PACKAGE_PIN	B22	[get_ports	"PMOD4_PIN7_R"]	;	#	Bank	16	VCCO	-	VCCO_3V3	-	IO_L9N_T1_DQS_16
347	set_property	IOSTANDARD	LVCOS33	[get_ports	"PMOD4_PIN7_R"]	;	#	Bank	16	VCCO	-	VCCO_3V3	-	IO_L9N_T1_DQS_16
348	set_property	PACKAGE_PIN	B24	[get_ports	"PMOD4_PIN3_R"]	;	#	Bank	16	VCCO	-	VCCO_3V3	-	IO_L10P_T1_16
349	set_property	IOSTANDARD	LVCOS33	[get_ports	"PMOD4_PIN3_R"]	;	#	Bank	16	VCCO	-	VCCO_3V3	-	IO_L10P_T1_16
350	set_property	PACKAGE_PIN	C13	[get_ports	"PMOD1_PIN1_R"]	;	#	Bank	16	VCCO	-	VCCO_3V3	-	IO_L10N_T1_16
351	set_property	IOSTANDARD	LVCOS33	[get_ports	"PMOD1_PIN1_R"]	;	#	Bank	16	VCCO	-	VCCO_3V3	-	IO_L10N_T1_16
352	set_property	PACKAGE_PIN	C14	[get_ports	"PMOD1_PIN8_R"]	;	#	Bank	16	VCCO	-	VCCO_3V3	-	IO_L11P_T1_SRCC_16
353	set property	IOSTANDARD	LVCOS33	[get ports	"PMOD1 PIN8 R"]	;	#	Bank	16	VCCO	-	VCCO_3V3	-	IO L11P T1 SRCC 16

After the synthesis is completed that may take up to 10 mins click “Open synthesized design” to assign the PWM signal





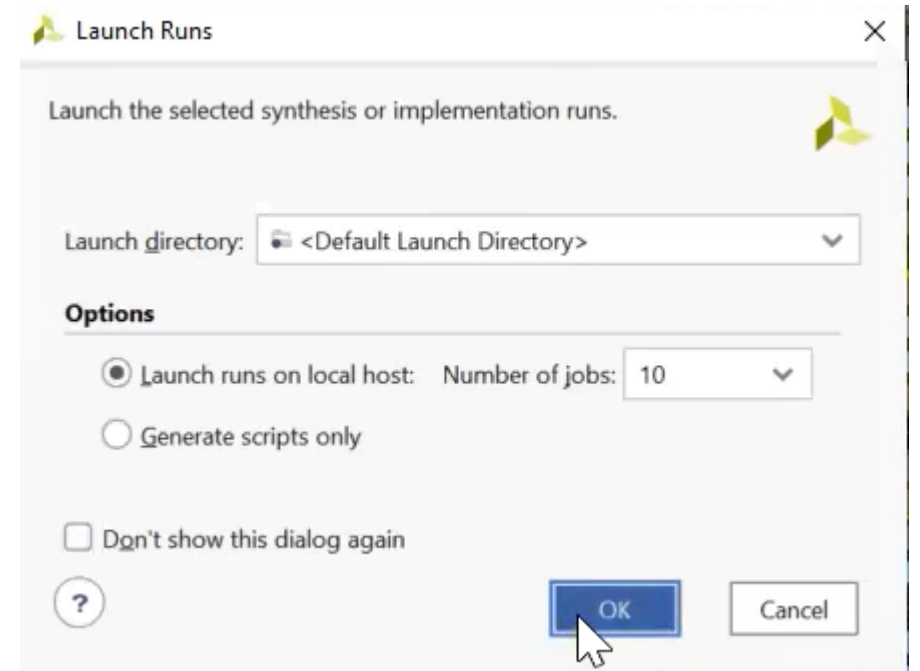
# Open the Scalar ports and assign “C13” to the “PWM signal” and change I/O Std to “LVCMOS33”

The screenshot displays the Xilinx Vivado IDE interface during the I/O Planning phase. The top menu bar includes File, Edit, Flow, Tools, Reports, Window, Layout, View, and Help. The main workspace is divided into several panes:

- Flow Navigator:** Shows the project structure with sections for PROJECT MANAGER, IP INTEGRATOR, SIMULATION, RTL ANALYSIS, and SYNTHESIS.
- Device Constraints:** Lists internal VREF settings (0.6V, 0.675V, 0.75V, 0.9V) and I/O Banks (NONE, I/O Bank 13, 14, 15, 16, 33).
- I/O Port Properties:** Configures the port `PWM_SP701` with Name: PWM\_SP701, Direction: OUT, Package pin: C13, and Fixed checked.
- Package Pins:** A grid showing the physical placement of pins on the device package.
- I/O Ports Table:** A table listing all I/O ports and their configurations.

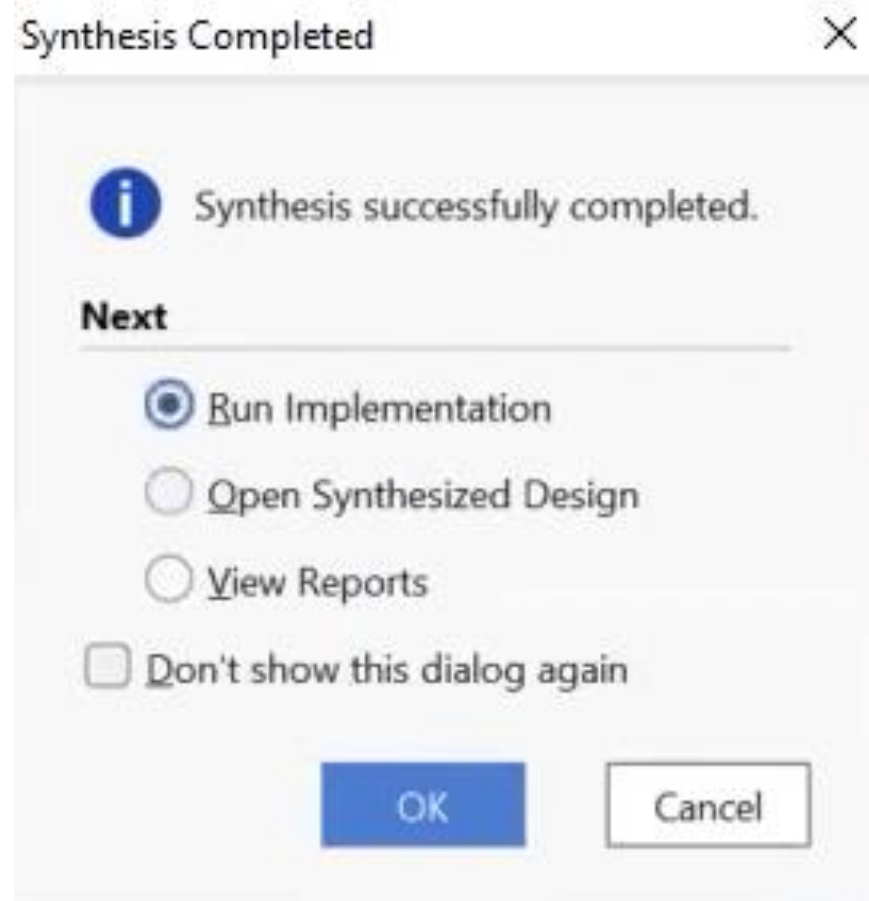
Name	Direction	Board Part Pin	Board Part Interface	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Termination	IN
> push_buttons_5bits_16164 (5)	IN					<input checked="" type="checkbox"/>	13	LVCMOS18	1.800				NONE	NONE	<input type="checkbox"/>
> rs232_uart_16164 (2)	(Multiple)					<input checked="" type="checkbox"/>	14	LVCMOS33*	3.300				NONE	(Multiple)	<input type="checkbox"/>
> RST.RESET_16164 (1)	IN					<input checked="" type="checkbox"/>	13	LVCMOS18	1.800				NONE	NONE	<input type="checkbox"/>
> spi_flash_16164 (5)	INOUT					<input checked="" type="checkbox"/>	14	LVCMOS33*	3.300		12	NONE	NONE	FP_VTT_50	<input type="checkbox"/>
> sys_diff_clock_16164 (2)	IN					<input checked="" type="checkbox"/>	33	LVDS_25*					NONE	NONE	<input type="checkbox"/>
> Scalar ports (1)															
PWM_SP701	OUT				C13	<input checked="" type="checkbox"/>	16	LVCMOS33*	3.300		12		NONE	FP_VTT_50	<input type="checkbox"/>

# Rerun synthesis after the changes

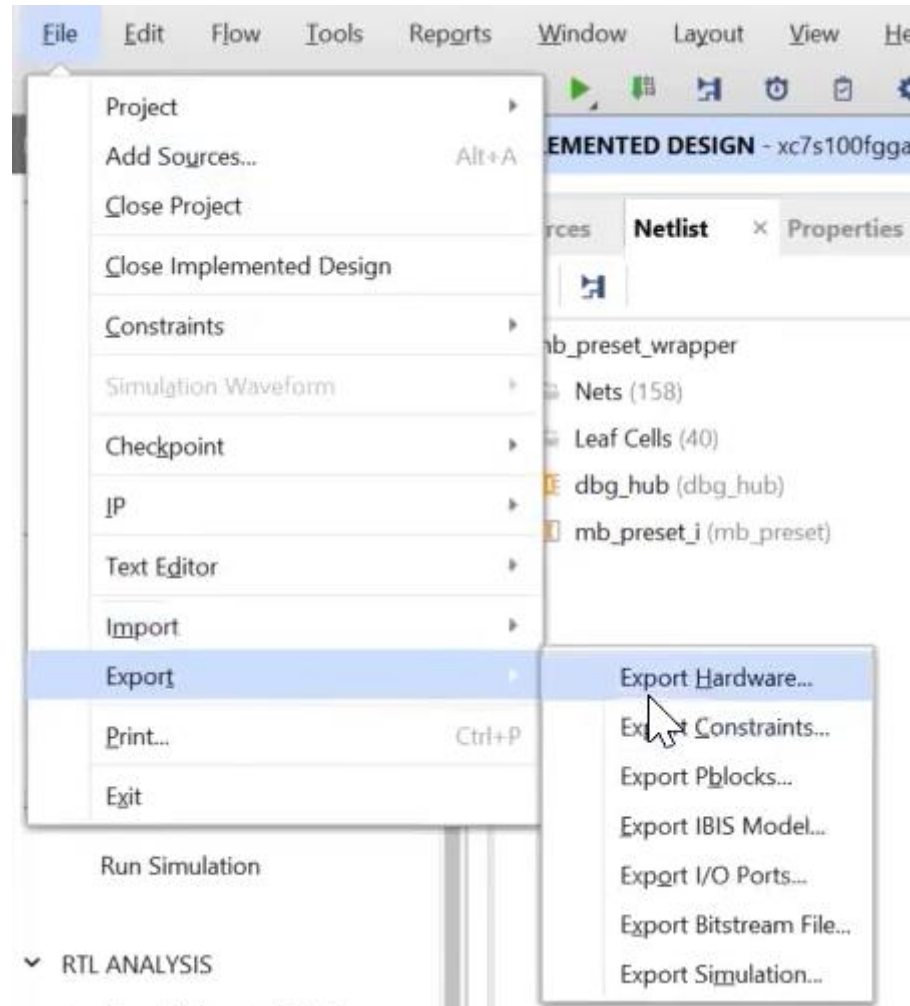


# Run implementation and generate a Bitstream by selecting “Generate Bitstream” under Program and Debug in the flow navigator

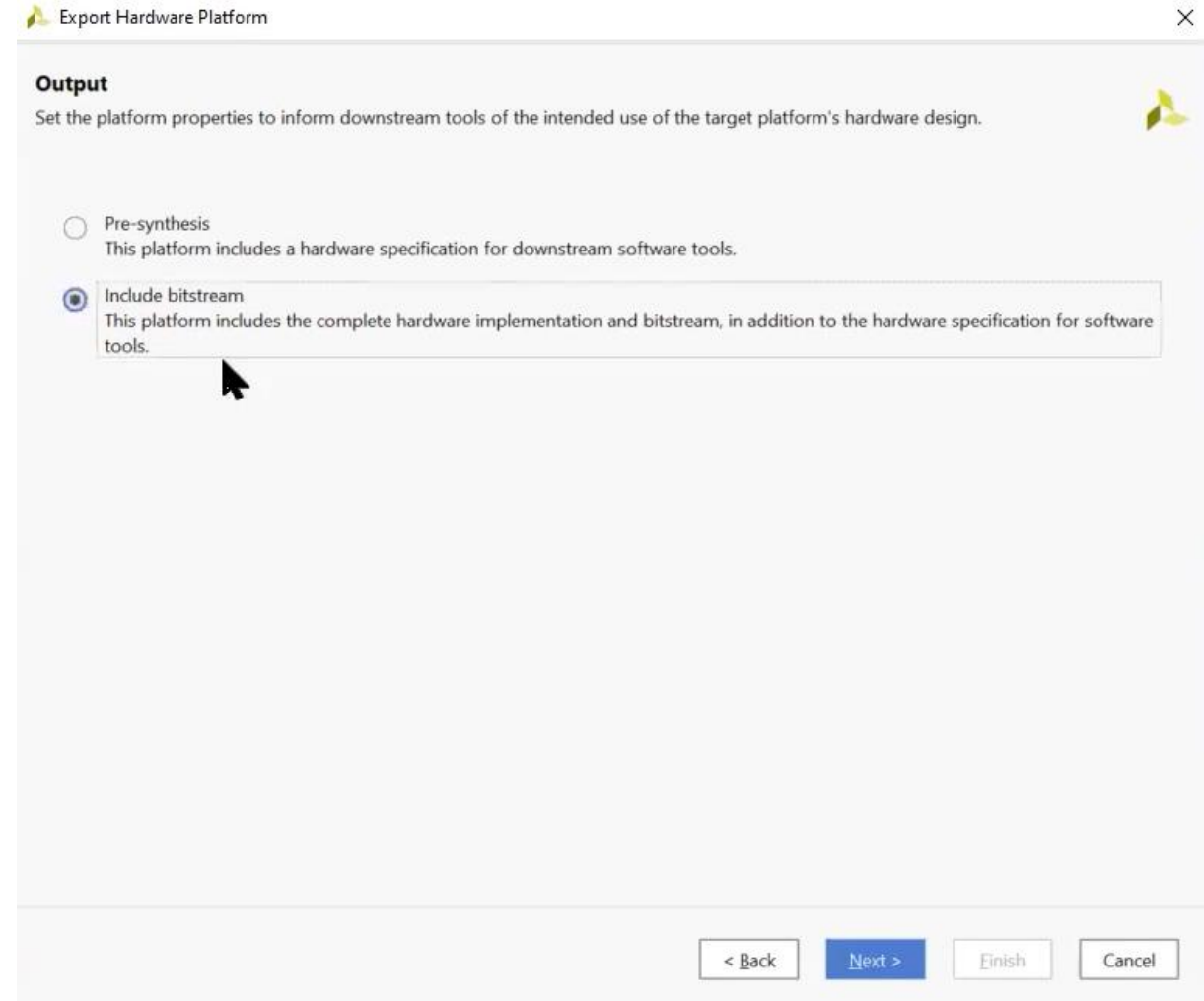
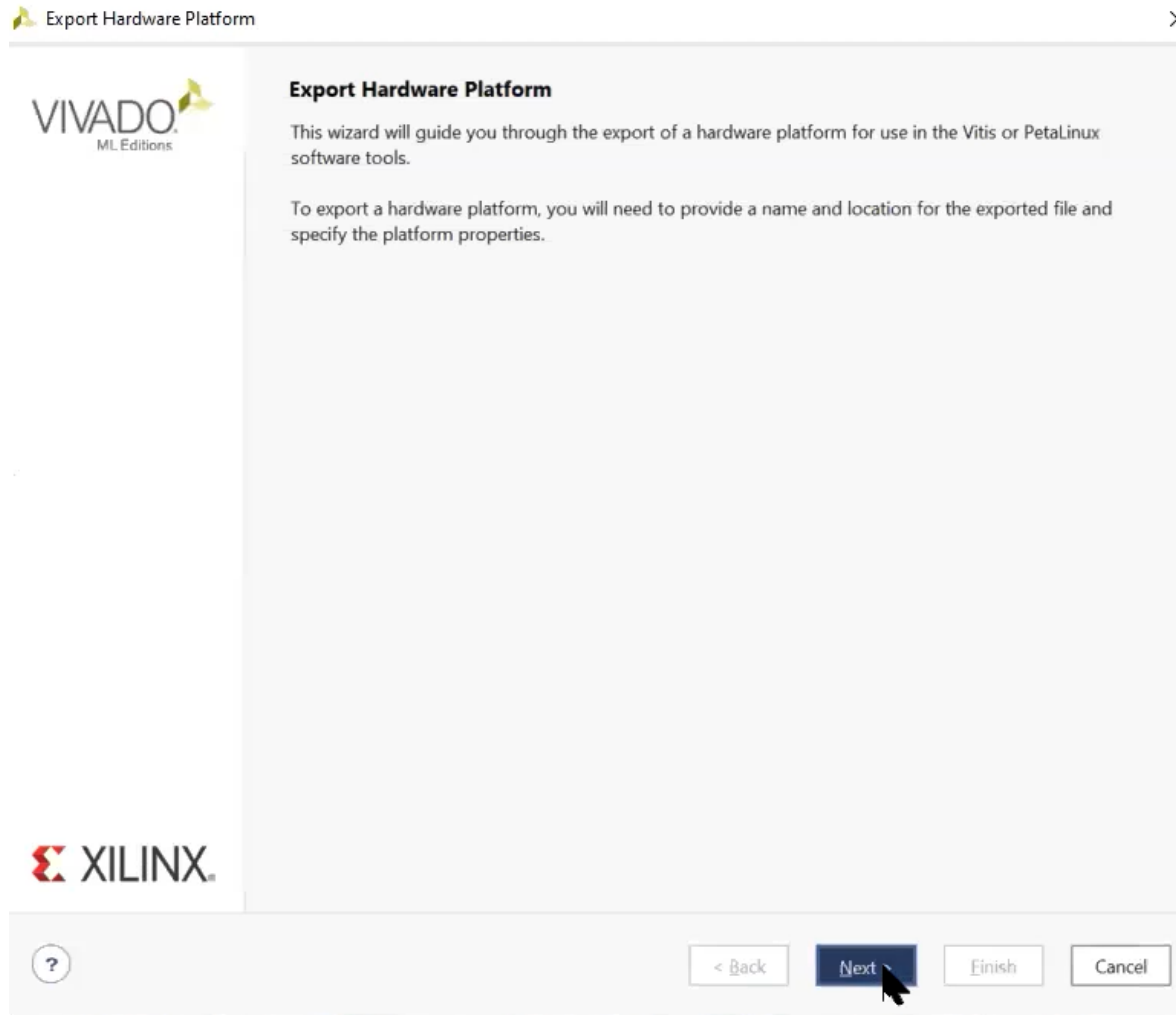
This bitstream generation may take up to 15 mins.



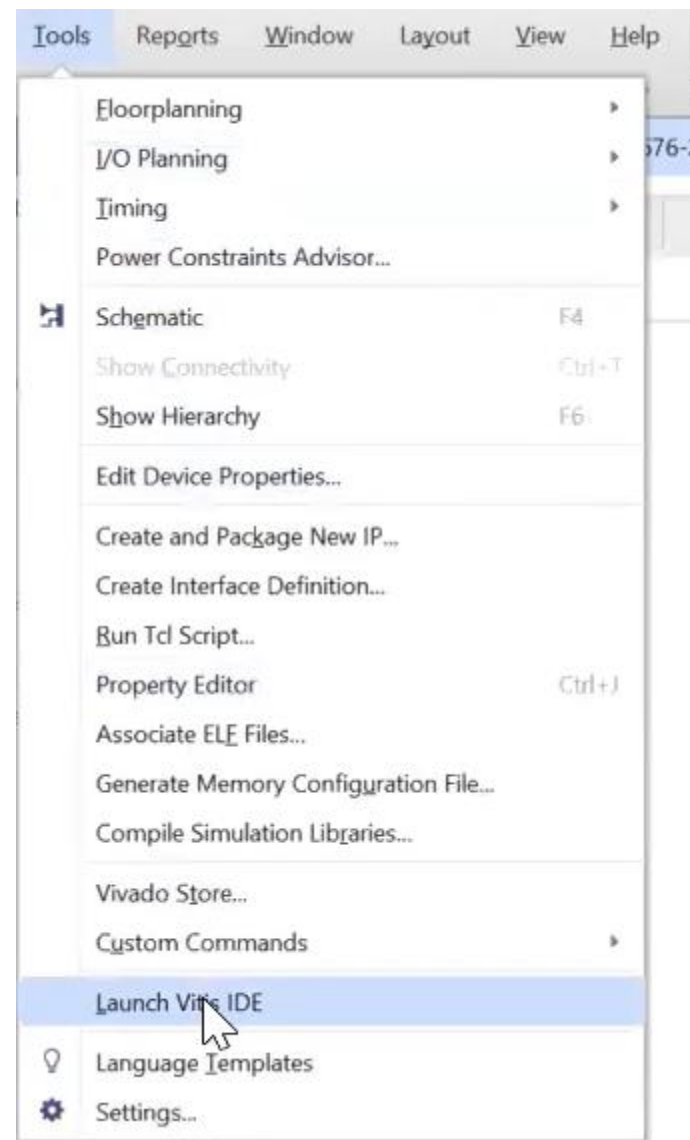
# Export Design select File → Export → Export Hardware in the Vivado Design Suite



# Click "Next" and select "Include Bitstream"



# Under Tools → Select “Launch Vitis”



# In Vitis IDE, go to File → New → Platform Project then click “Next”

The screenshot shows the Vitis IDE interface. On the left, the 'PROJECT' sidebar has 'Create Application Project' highlighted with a red box. On the right, the 'New Application Project' wizard is open. The wizard title is 'New Application Project' and the subtitle is 'Create a New Application Project'. The main content area contains the following text:

This wizard will guide you through the 4 steps of creating new application projects.

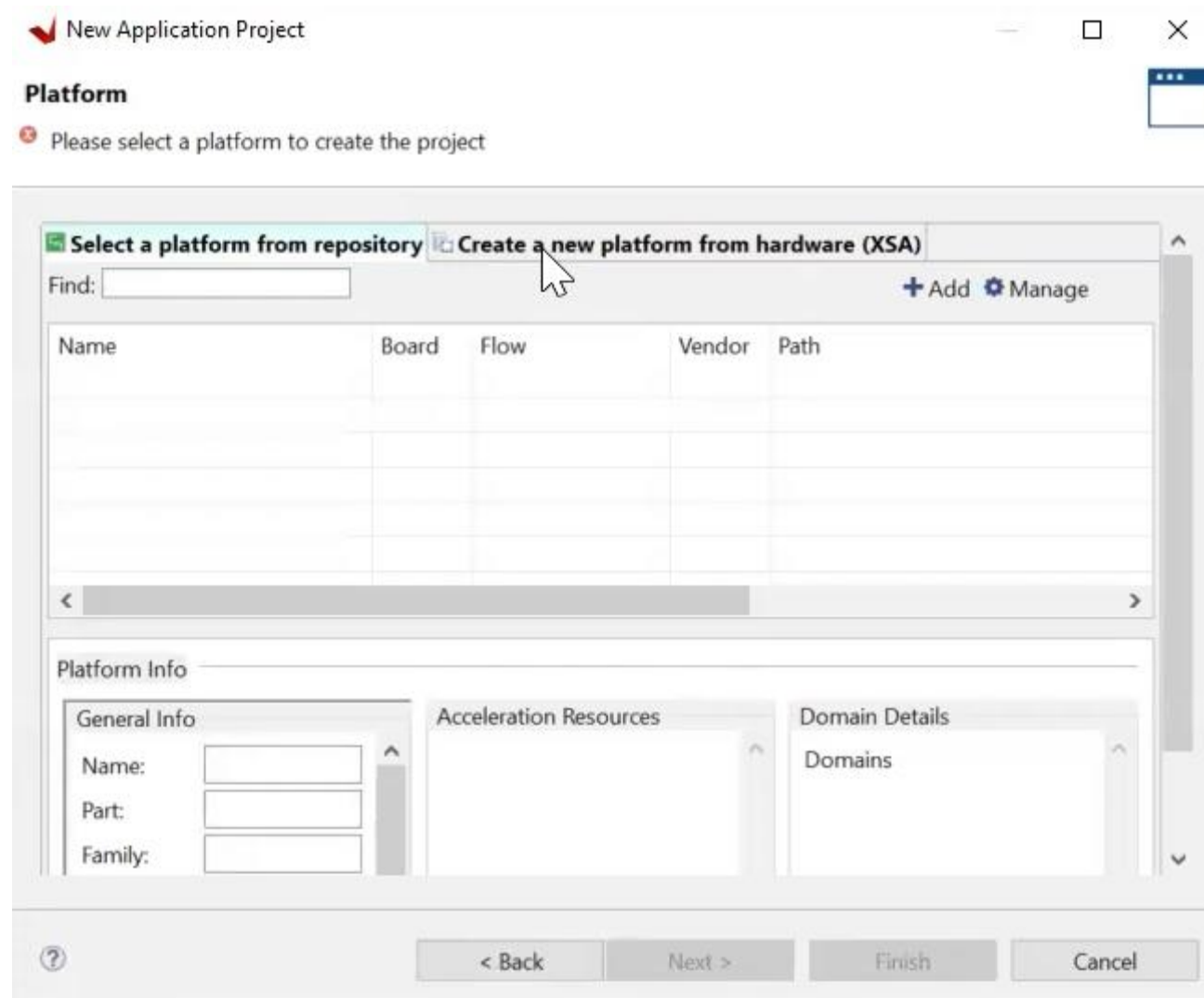
1. Choose a **platform** or create a **platform project** from Vivado exported XSA
2. Put application project in a **system project**, associate it with a processor
3. Prepare the application runtime – **domain**
4. Choose a template for application to quick start development

Below the list is a diagram showing the project structure. A 'Processor' box is connected to a 'Domain' box (part of a 'Platform Project' container) and an 'App' box (part of a 'System Project' container). The 'Domain' box is also connected to an 'XSA' box. The 'Platform Project' container is pink, and the 'System Project' container is blue.

Below the diagram, there is a checkbox:  Skip welcome page next time. (Can be reached with Back button)

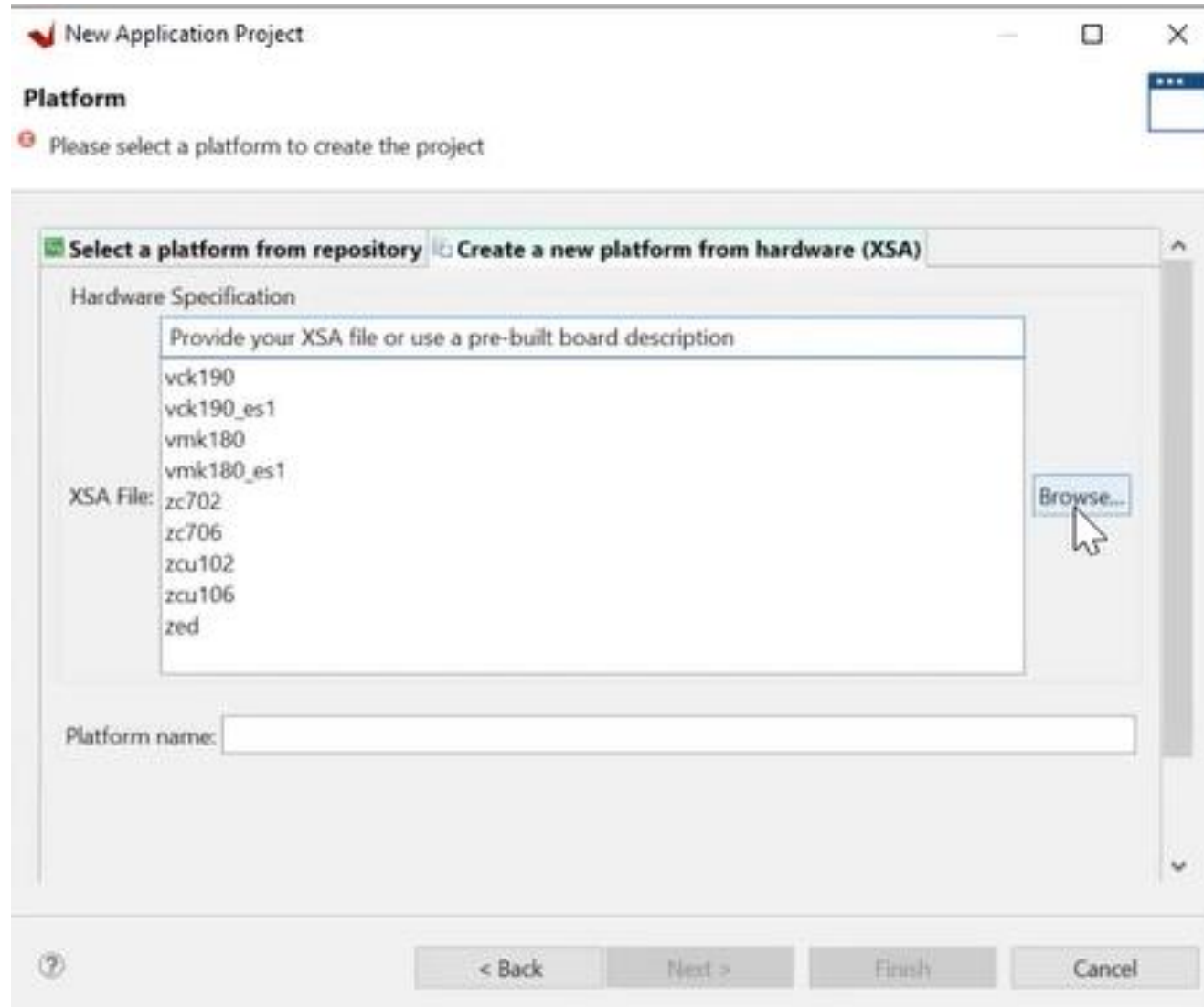
At the bottom of the wizard, there are four buttons: '?', '< Back', 'Next', 'Finish', and 'Cancel'. A mouse cursor is pointing at the 'Next' button.

In the Platform window select “create a new platform from hardware (XSA)”





# Browse the XSA file and click “Next”



Set the project name to SP701\_MicroBlaze, then set the system project name to “your board name” system then click “Next”

New Application Project

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

Application project name: SP701\_MicroBlaze

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

Select a system project  
+ Create new...

System project details

System project name: SP701\_MicroBlaze\_system

Target processor

Select target processor for the Application project.

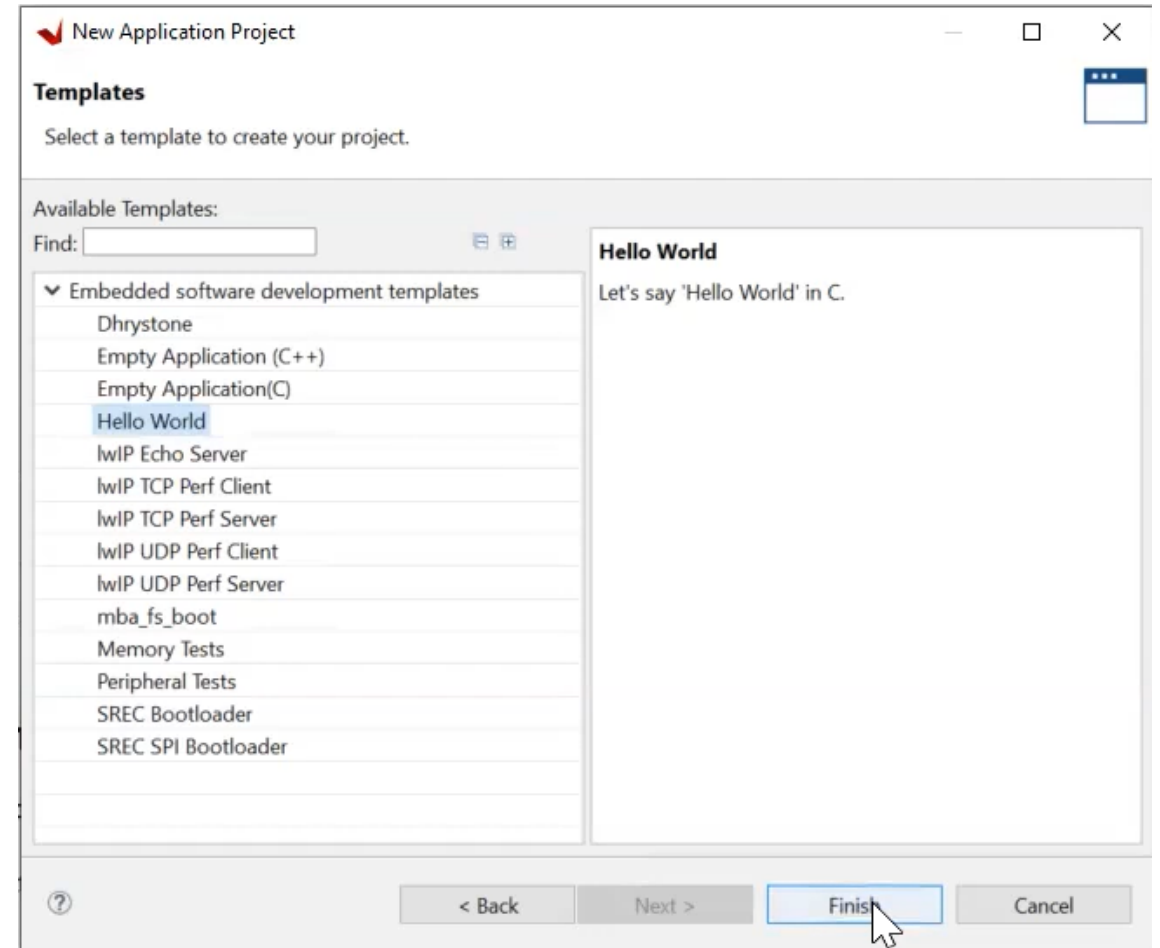
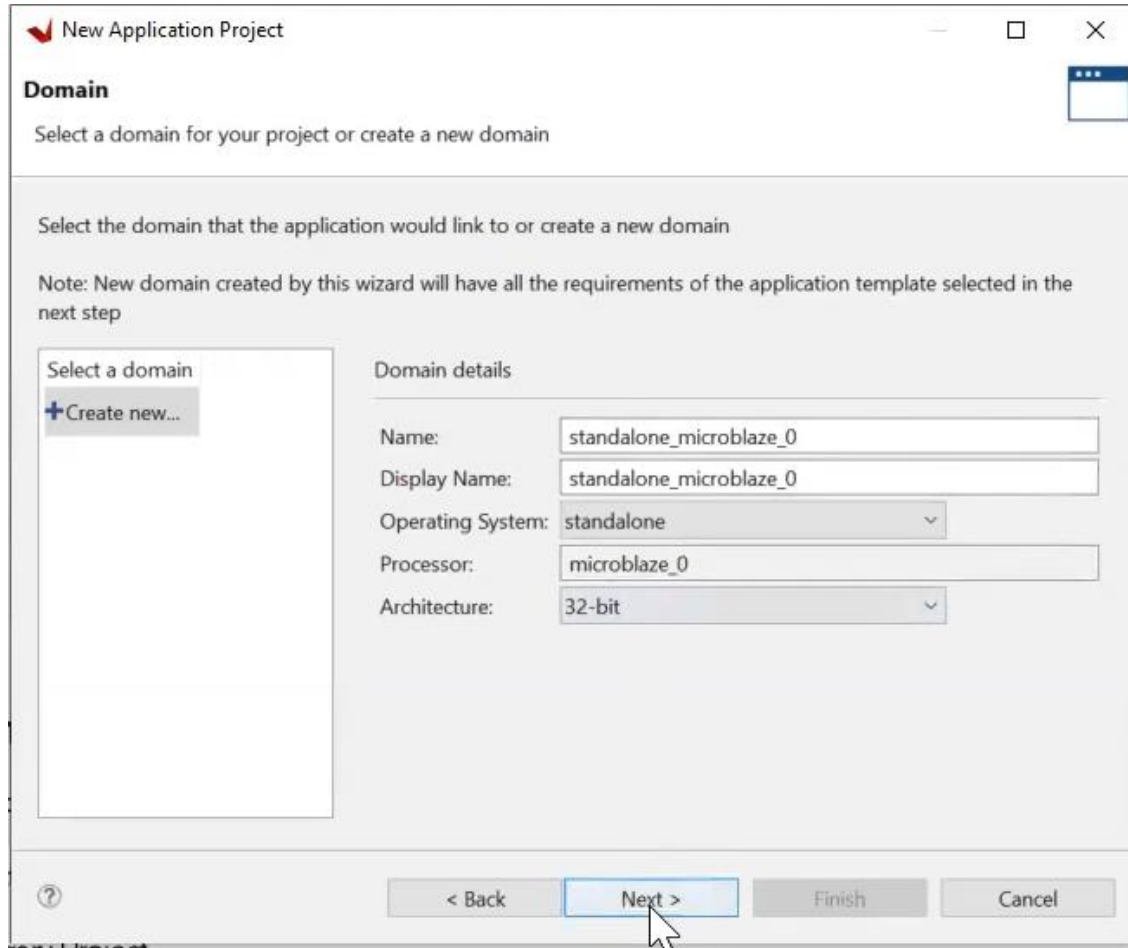
Processor	Associated applications
microblaze_0	SP701_MicroBlaze

Show all processors in the hardware specification

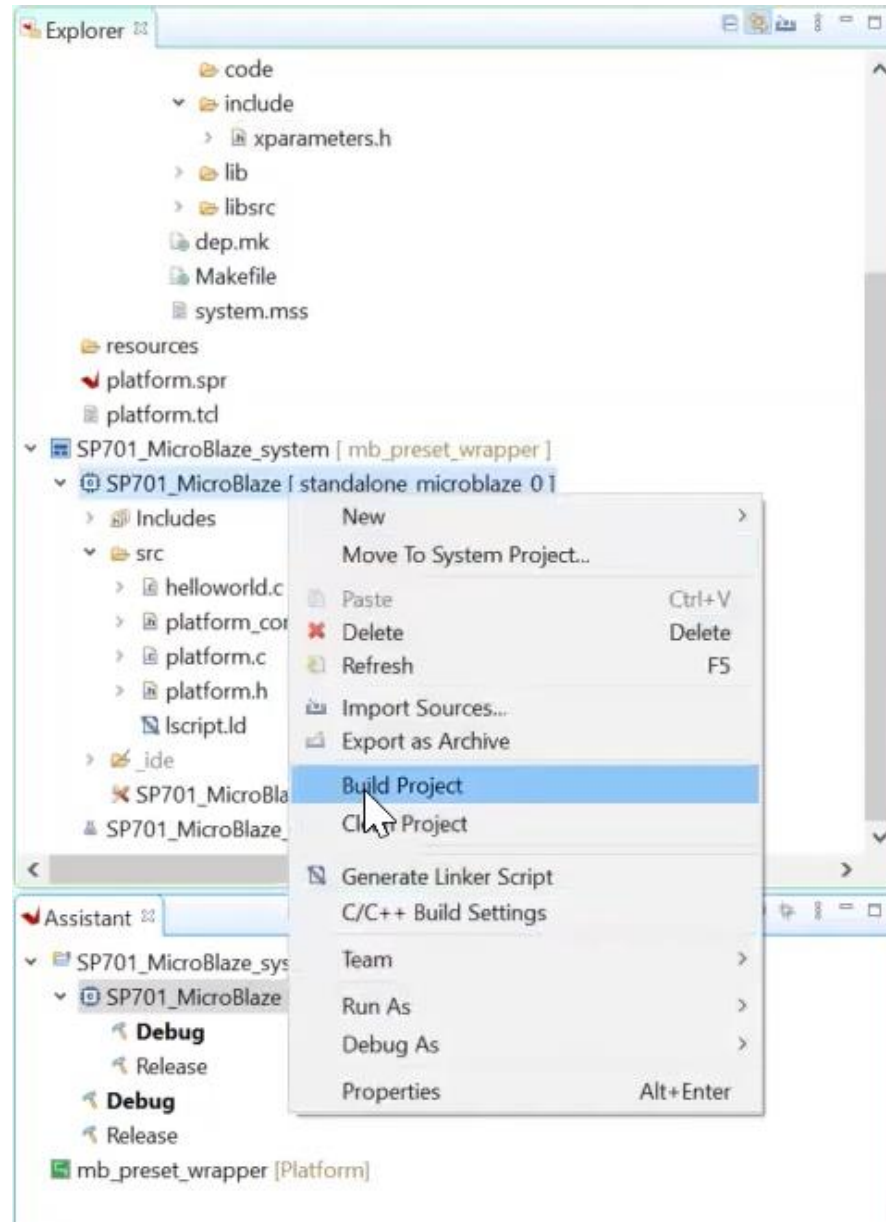
< Back Next > Finish Cancel

rary Project

# Click "Next" again to choose the Hello World template and press "Finish"



# Build the hardware by right-clicking Platform → Build Project



# Connect the board

Connect power cable to the board and connect a USB micro cable from the board's J5 USB JTAG connector to the Windows machine

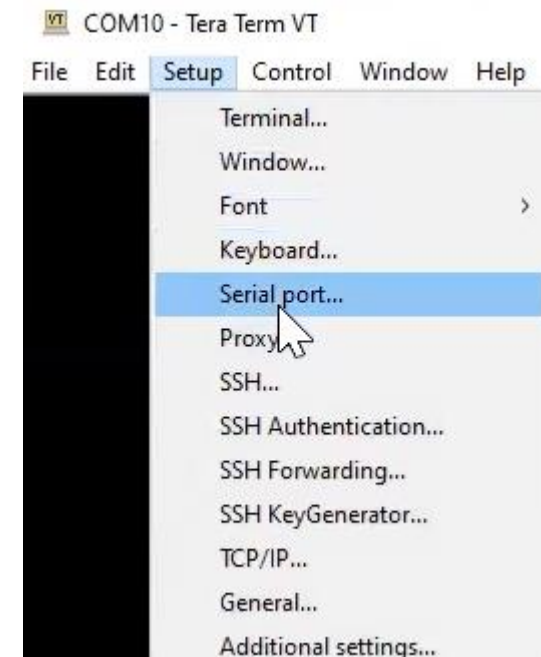
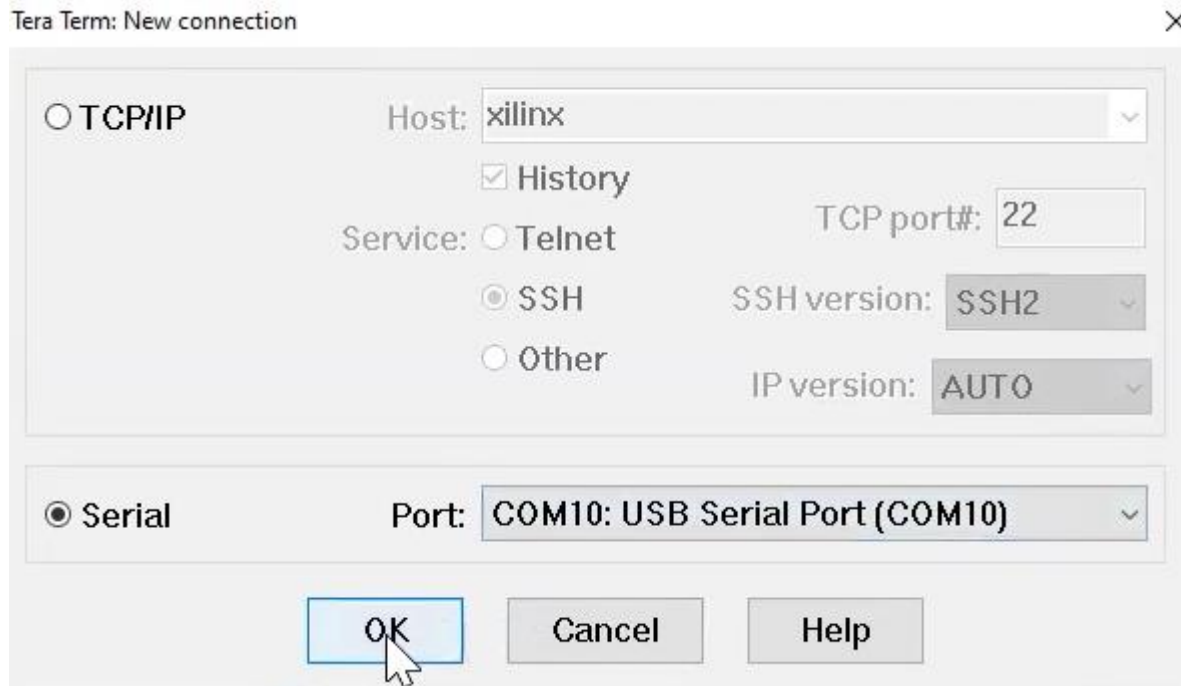
Power the board using the power switch, which will automatically start the Built-in Self-Test



USB connected to PC

Power Cable connected to power outlet

# Set up Tera Term for serial communication. Under set up, select serial port



# Match the UART speed set in the Vivado UART IP setting for this project and save the new setting

Tera Term: Serial port setup and connection

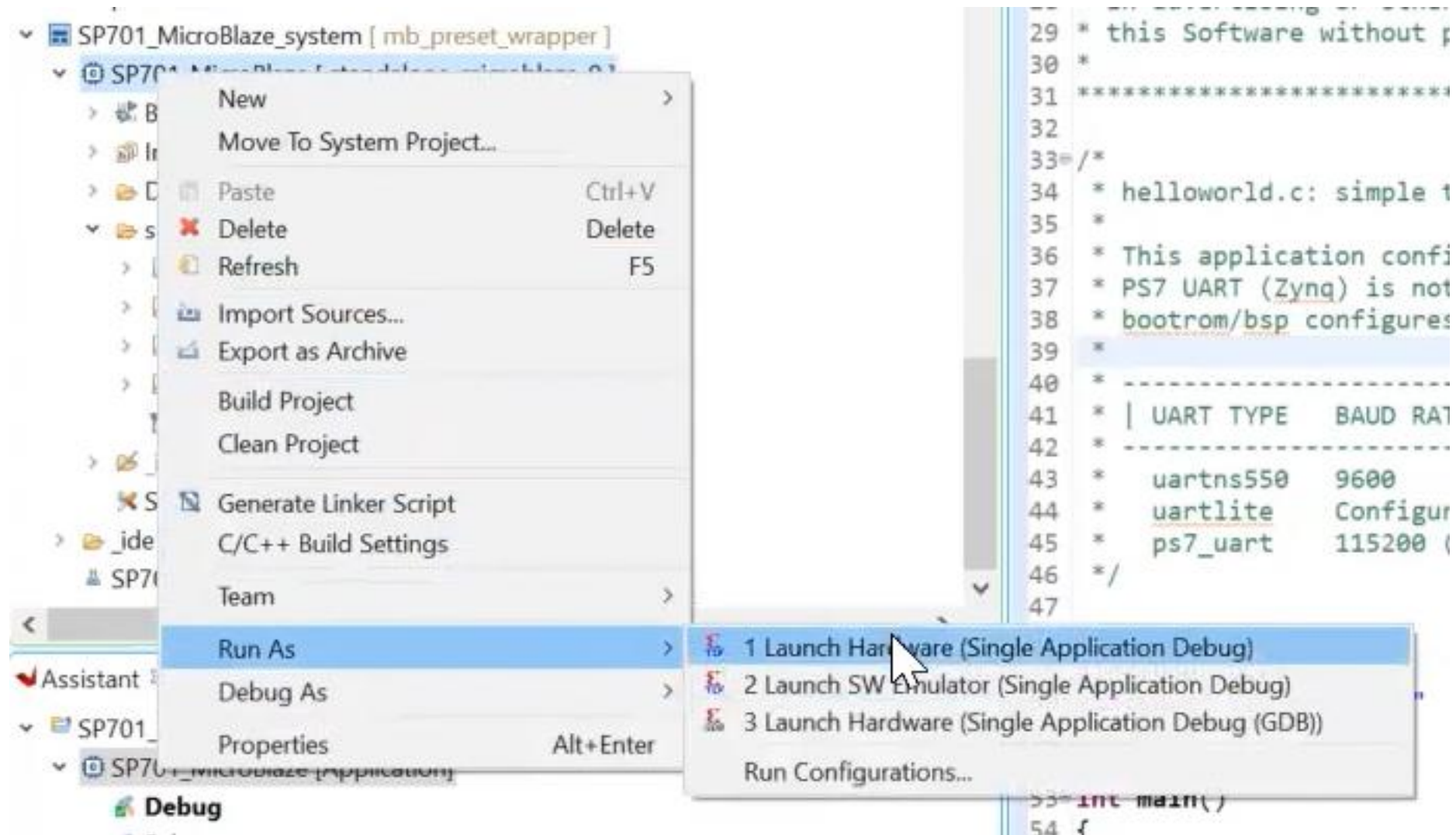
Port:	COM10	<b>New setting</b> Cancel Help
Speed:	115200	
Data:	8 bit	
Parity:	none	
Stop bits:	1 bit	
Flow control:	none	

Transmit delay

0	msec/char	0	msec/line
---	-----------	---	-----------

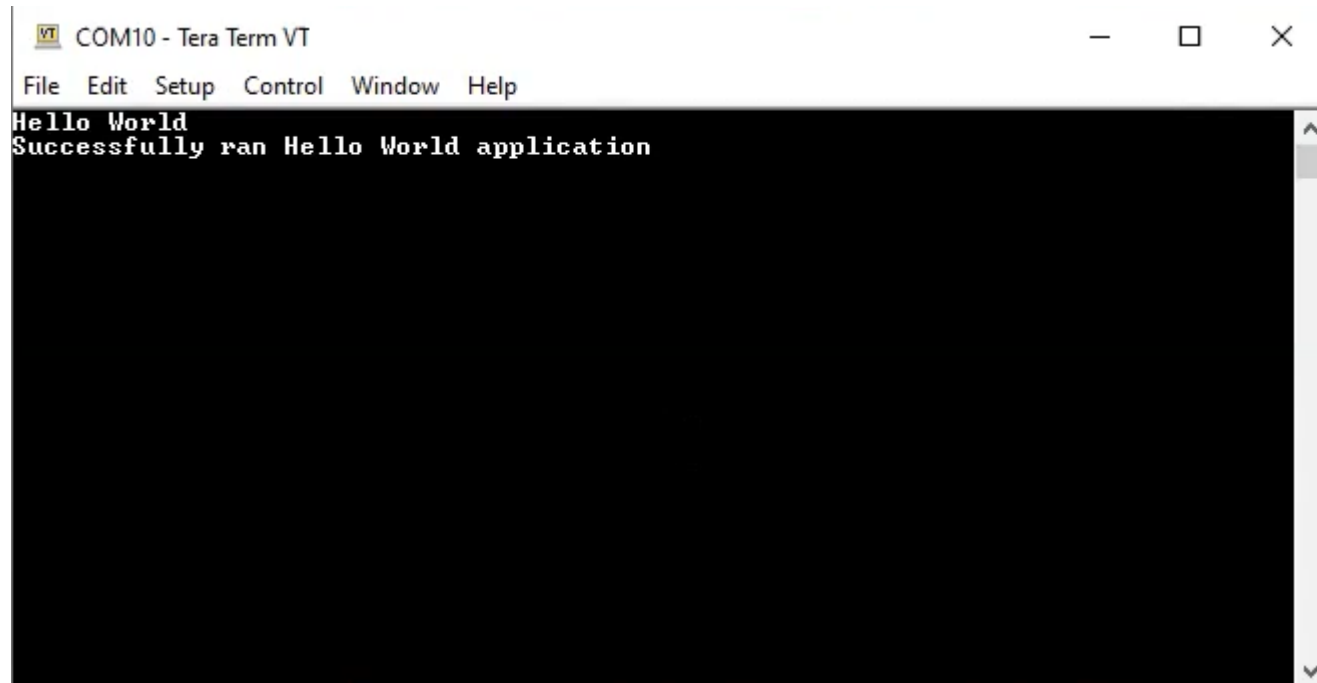
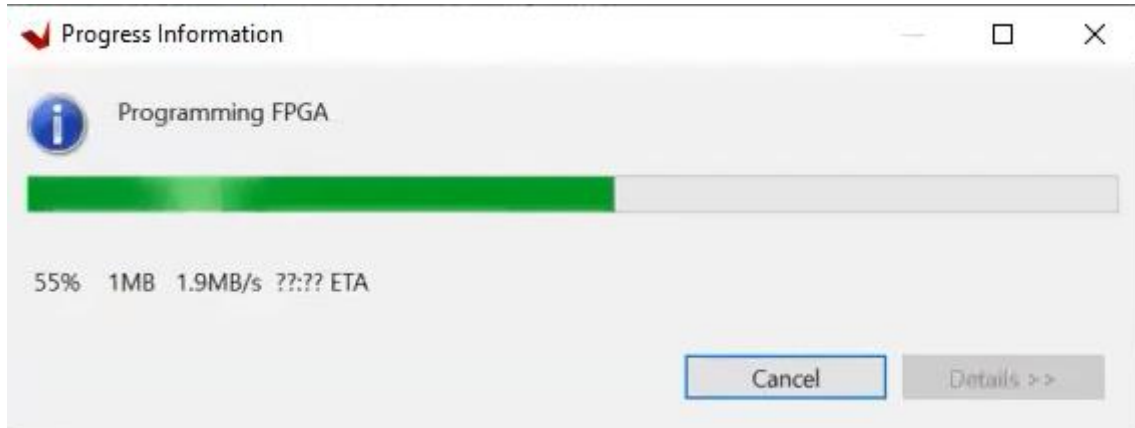
Device Friendly Name: USB Serial Port (COM10)  
Device Instance ID: FTDIBUS\VID\_0403+PID\_6011+31931107268B  
Device Manufacturer: FTDI  
Provider Name: FTDI  
Driver Date: 8-16-2017  
Driver Version: 2.12.28.0

# Launch the hardware by selecting Run as → Launch Hardware (Single application Debug)





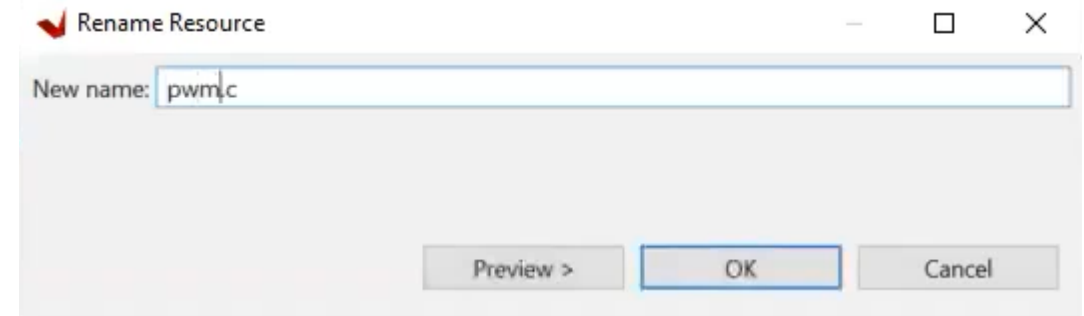
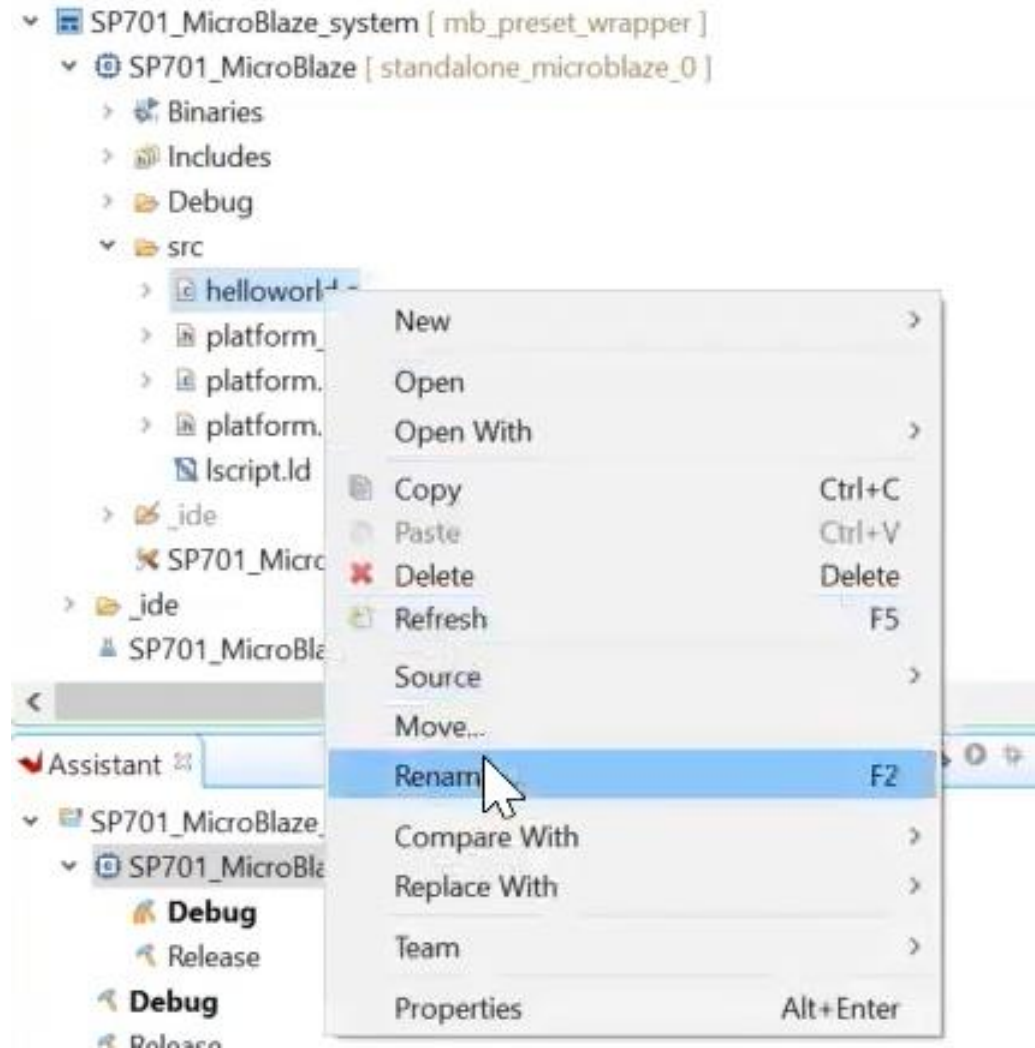
# Hello World will appear on the serial communication in Tera Term





# Pulse Width Modulation (PWM) application

Under the Source folder in the Explorer window, rename the Hello-World.c to “PWM.c”



# Copy and Paste the PWM code to the C file

- ▶ You can access the code under Demonstration steps in:
  - <http://wiki.xilinx.com/Spartan-7+SP701+Evaluation+Kit+PWM+Tutorial>

## Demonstration Steps

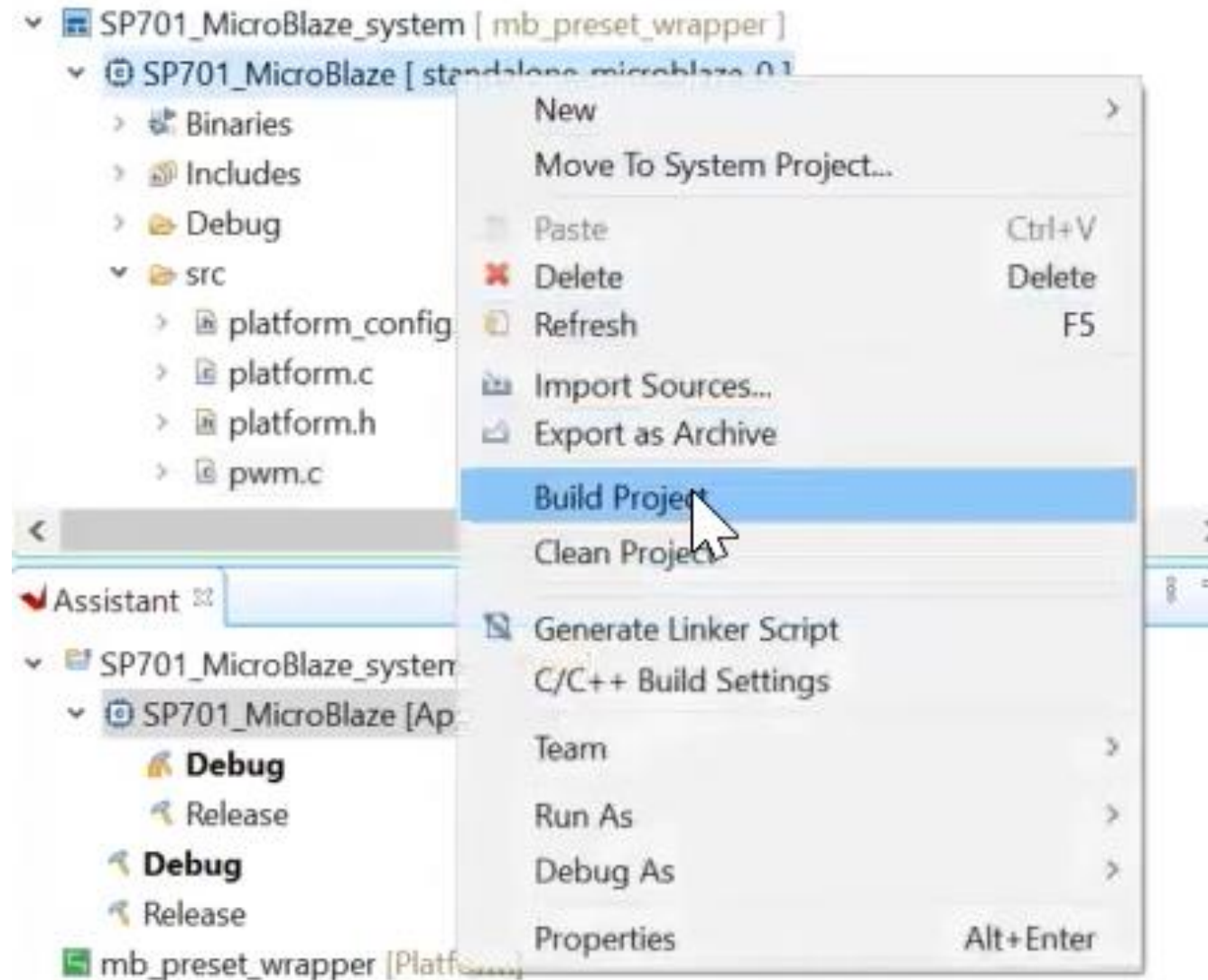
The demonstration video will cover two quick examples of how to build a system from scratch. The first is a Hello World example using MicroBlaze and the second is the PWM application example.

Expand the following section to view the source code for the PWM demo

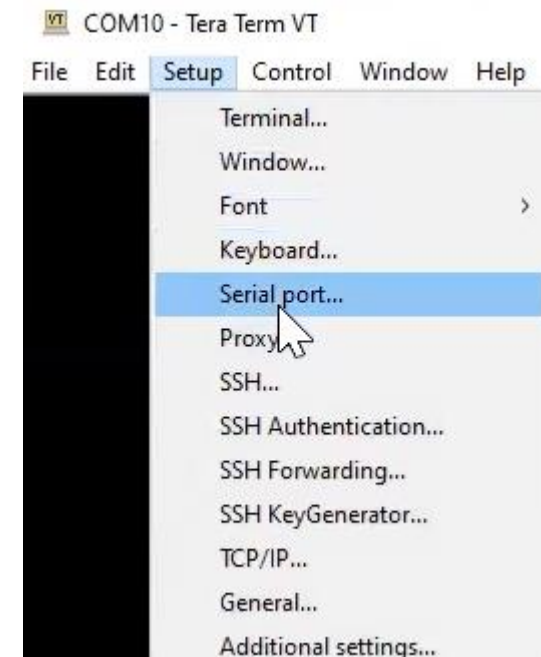
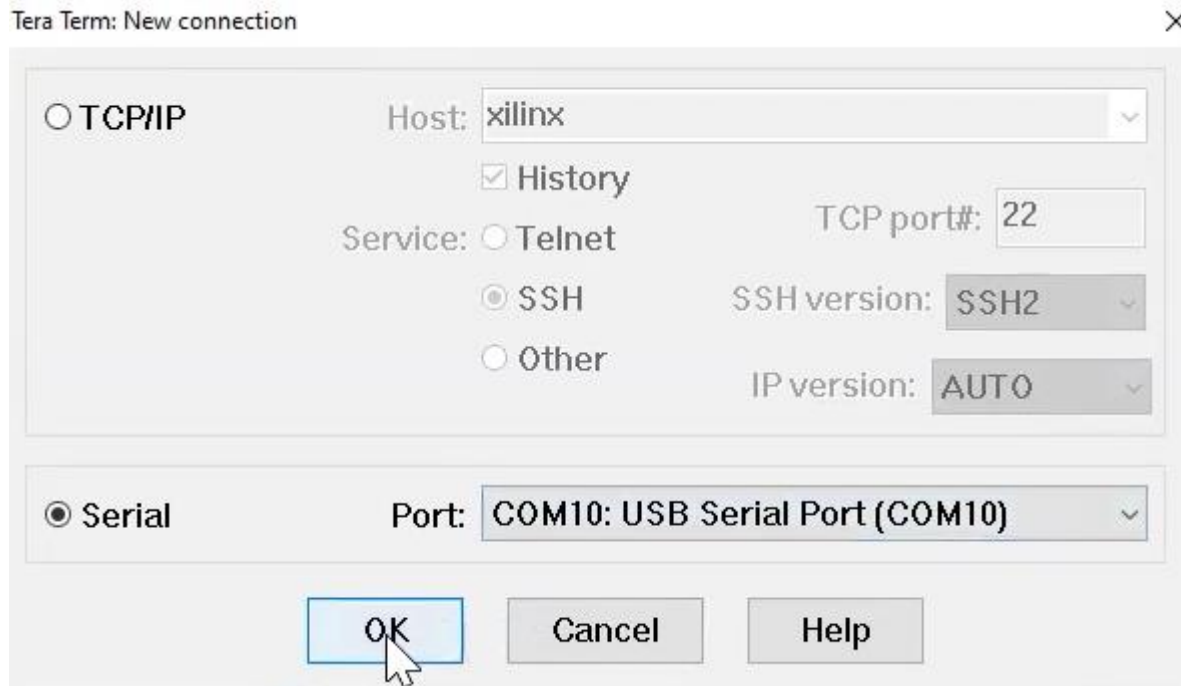
▼ Please copy the code below for the PWM application:

```
1 # Copyright (c) 2021, Xilinx, Inc.
2 # All rights reserved.
3 #
4 # Redistribution and use in source and binary forms, with or without
5 # modification, are permitted provided that the following conditions are met:
6 #
7 # 1. Redistributions of source code must retain the above copyright notice,
8 #    this list of conditions and the following disclaimer.
9 #
10 # 2. Redistributions in binary form must reproduce the above copyright
11 #    notice, this list of conditions and the following disclaimer in the
12 #    documentation and/or other materials provided with the distribution.
13 #
14 # 3. Neither the name of the copyright holder nor the names of its
15 #    contributors may be used to endorse or promote products derived from
16 #    this software without specific prior written permission.
17 #
18 # THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
19 # AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
20 # THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
21 # PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
22 # CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
23 # EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
24 # PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
25 # OR BUSINESS INTERRUPTION). HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
26 # WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
27 # OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
28 # ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
29
30
31 /* Include Header Files and Packages */
32 #include <stdio.h>
33 #include "platform.h"
34 #include "xil_printf.h"
35 #include "xtmrctr.h"
36
37 /* Device IDs Definition */
38 #define GpioLed    XPAR_AXI_GPIO_0_DEVICE_ID
39 #define TMRCTR_ID XPAR_TMRCTR_0_DEVICE_ID
40 #define UARTTraceAddress XPAR_AXI_UART1TTF_0_RACFADDR
```

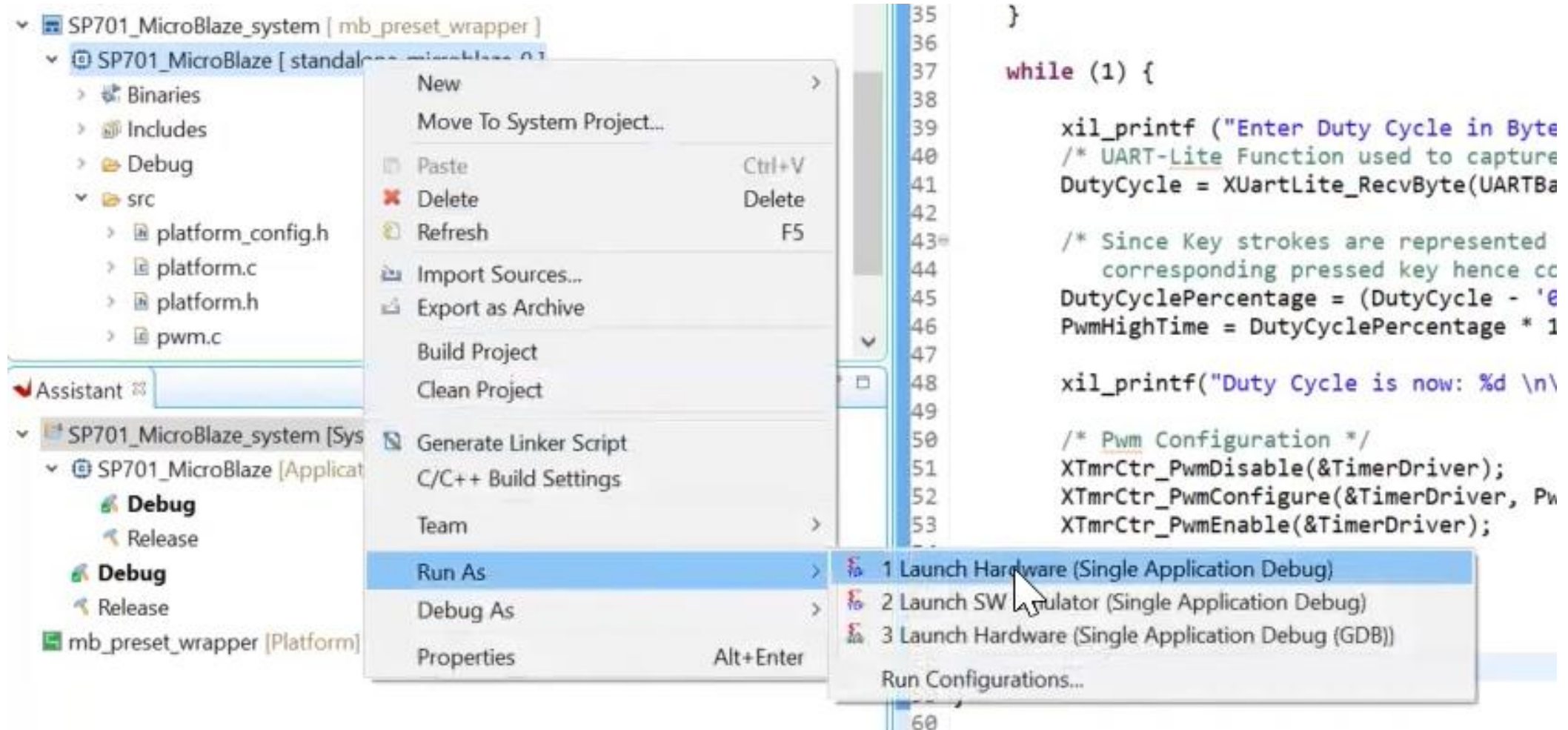
# Right-click on the application project and select build



# Open Tera Term again and set up the serial communication Make sure that the board is connected



# Right-click on the platform and select Run as → Launch Hardware to program the FPGA



The screenshot shows an IDE interface with a project tree on the left, a context menu in the center, and a code editor on the right.

**Project Tree:**

- SP701\_MicroBlaze\_system [ mb\_preset\_wrapper ]
  - SP701\_MicroBlaze [ standard\_microblaze\_0.1 ]
    - Binaries
    - Includes
    - Debug
    - src
      - platform\_config.h
      - platform.c
      - platform.h
      - pwm.c
- Assistant
- SP701\_MicroBlaze\_system [ Sys ]
  - SP701\_MicroBlaze [ Applicat ]
    - Debug
    - Release
  - Debug
  - Release
- mb\_preset\_wrapper [ Platform ]

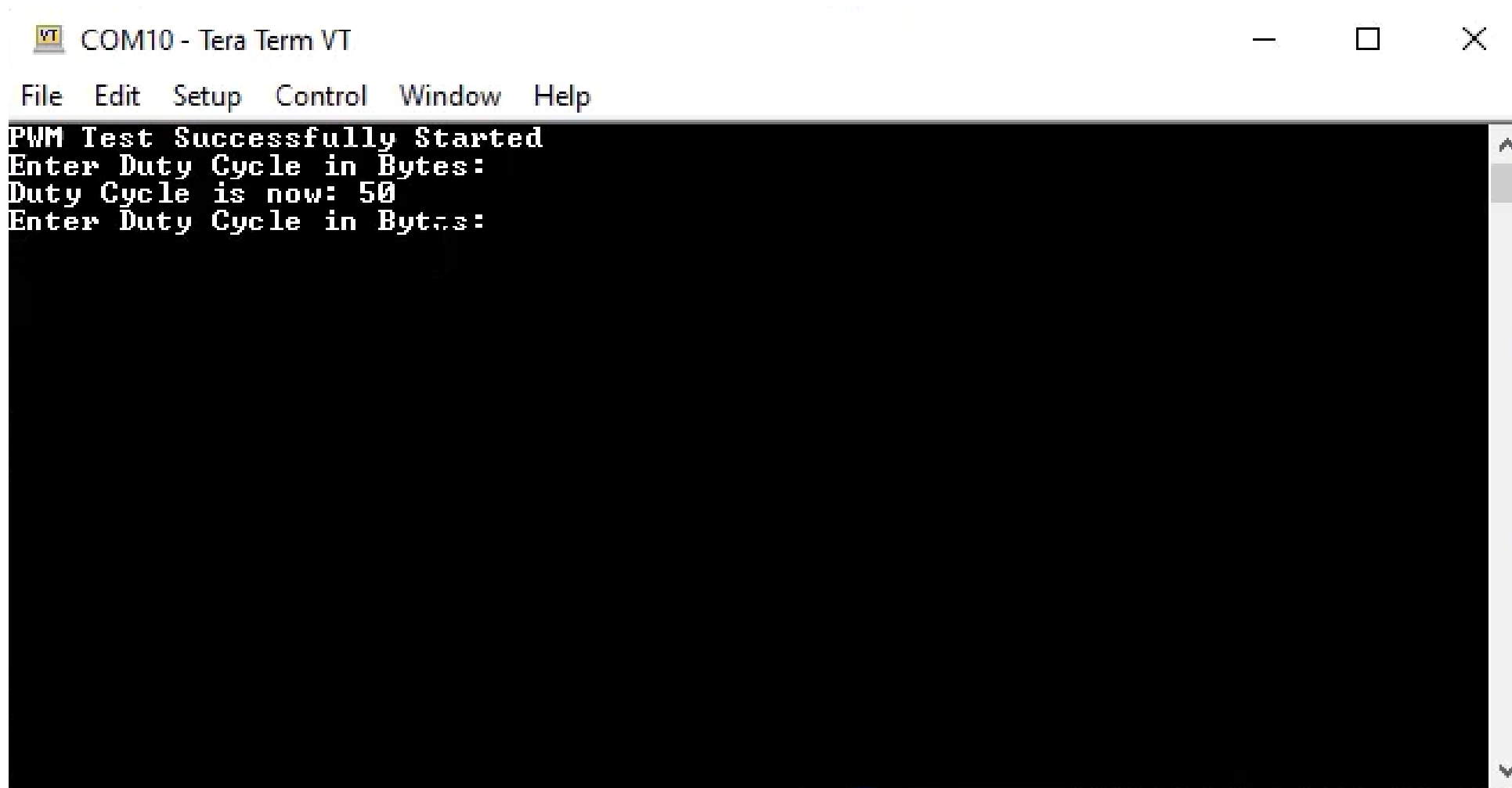
- New
- Move To System Project...
- Paste (Ctrl+V)
- Delete (Delete)
- Refresh (F5)
- Import Sources...
- Export as Archive
- Build Project
- Clean Project
- Generate Linker Script
- C/C++ Build Settings
- Team
- Run As**
- Debug As
- Properties (Alt+Enter)

- 1 Launch Hardware (Single Application Debug)
- 2 Launch SW Simulator (Single Application Debug)
- 3 Launch Hardware (Single Application Debug (GDB))
- Run Configurations...

```
35 }
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
60
```

```
while (1) {
    xil_printf ("Enter Duty Cycle in Byte
/* UART-Lite Function used to capture
DutyCycle = XUartLite_RecvByte(UARTBa
/* Since Key strokes are represented
corresponding pressed key hence cc
DutyCyclePercentage = (DutyCycle - '0
PwmHighTime = DutyCyclePercentage * 1
xil_printf("Duty Cycle is now: %d \n\
/* Pwm Configuration */
XTmrCtr_PwmDisable(&TimerDriver);
XTmrCtr_PwmConfigure(&TimerDriver, Pw
XTmrCtr_PwmEnable(&TimerDriver);
```

# In Terminal, enter duty cycle



The image shows a terminal window titled "COM10 - Tera Term VT". The window has a menu bar with "File", "Edit", "Setup", "Control", "Window", and "Help". The terminal output is as follows:

```
PWM Test Successfully Started  
Enter Duty Cycle in Bytes:  
Duty Cycle is now: 50  
Enter Duty Cycle in Bytes:
```



In Vivado, to view the Integrated Logic Analyzer (ILA), select Hardware Manger then “Open target” and “Auto Connect”



50% duty cycle waveform will be shown



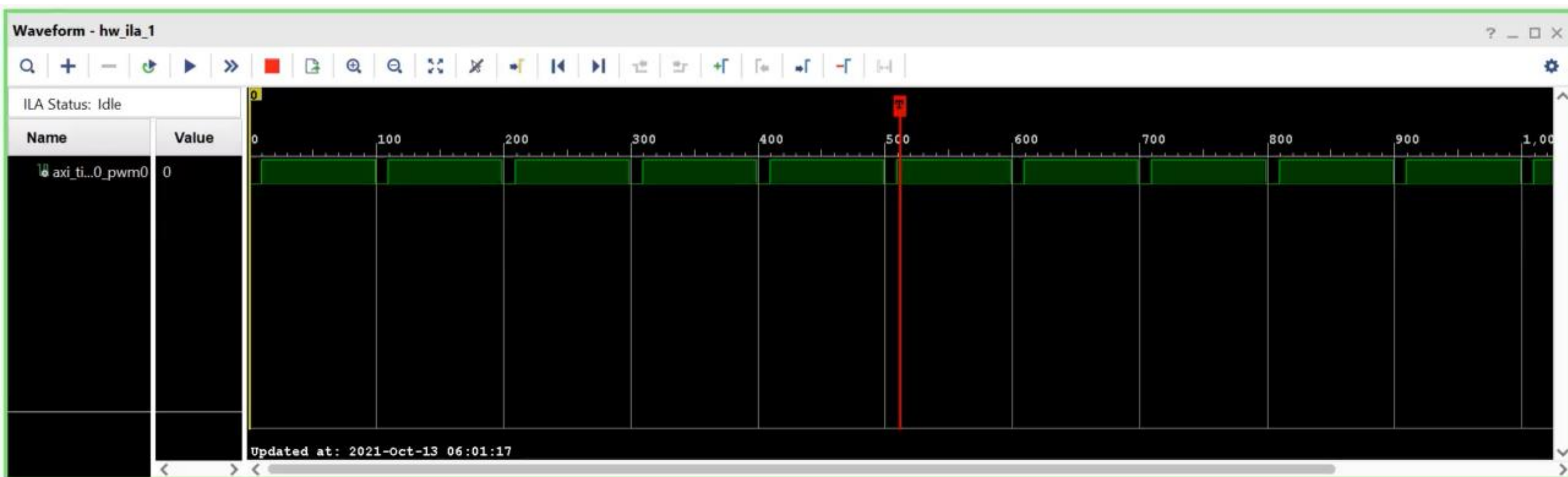
Change the duty cycle to 10%, refresh the waveform and run again the waveform will be updated

```
Duty Cycle is now: 50  
Enter Duty Cycle in Bytes:  
Duty Cycle is now: 10  
Enter Duty Cycle in Bytes:
```



# Apply the same steps for 90% duty cycle

```
Duty Cycle is now: 50
Enter Duty Cycle in Bytes:
Duty Cycle is now: 10
Enter Duty Cycle in Bytes:
Duty Cycle is now: 90
Enter Duty Cycle in Bytes:
```



# Conclusion

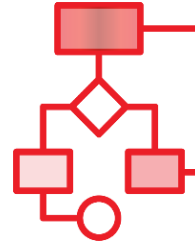
- ▶ In this tutorial:
  - We used Vivado example projects to build a customizable 32-bit embedded processor
  - Used Vivado IP Integrator to quickly add PWM IP block from IP library
  - Built the entire design without writing any RTL code
  - Used Vitis to validate operational Microcontroller by running Hello-World
  - Create and ran PWM code to change pulse-width of output signal
- ▶ PWMs are valuable in many designs as lighting, motor control, power supply control and much more
- ▶ Xilinx enables the ability to connect as many PWMs as I/Os are available
- ▶ This design can be used as a building block for you to start your own design

# Available Resources



## Collateral

- › [MicroBlaze Soft Processor Core Product Page](#)
- › [MicroBlaze Getting Started Wiki Page](#)
- › [Spartan-7 Product Brief](#)
- › [White Paper: Spartan-7 Family](#)
- › [Cost Optimized FPGAs and SoCs](#)
- › [Unboxing SP701 FPGA Evaluation Kit](#)
- › [MicroBlaze Quick Start Video](#)



## Tutorials & Guides

- › [Quick Start Guide: MicroBlaze Soft Processor for Vitis 2021.1](#)
- › [MicroBlaze Processor Reference Guide](#)
- › [Embedded Processor Hardware Design in Vivado Tutorial](#)
- › [Creating and Packaging Custom IP in Vivado](#)



## Workshops & Trainings

- › [No hardware experience? No problem! Xilinx MicroBlaze processors are for everyone.](#)
- › [Arty-S7 Workshops:](#)
  - › [Part 1: Learn about Xilinx FPGAs and Embedded Processing](#)
  - › [Part 2: Building a Custom Microcontroller in Minutes](#)
  - › [Part 3: Rapid Sensor Prototyping with Digilent Peripheral Modules](#)
- › [Embedded System Design Training](#)
- › [Sensor Fusion at the Edge with Spartan-7](#)



---

**Thank You**

