

Mastering AMD MicroBlaze™ Processor: The PetaLinux Class

Adam@AduvoEngineering.com

Objective

The objective of this lab are:

1. Create an AMD MicroBlaze™ processor system using AMD Vivado™ software
2. Create a PetaLinux project for the AMD MicroBlaze processor
3. Update the PetaLinux to support SPI and I²C interfaces
4. Develop SW and Debug SW application using AMD Vitis™ software
5. Develop a boot image for the development board

Development Tools

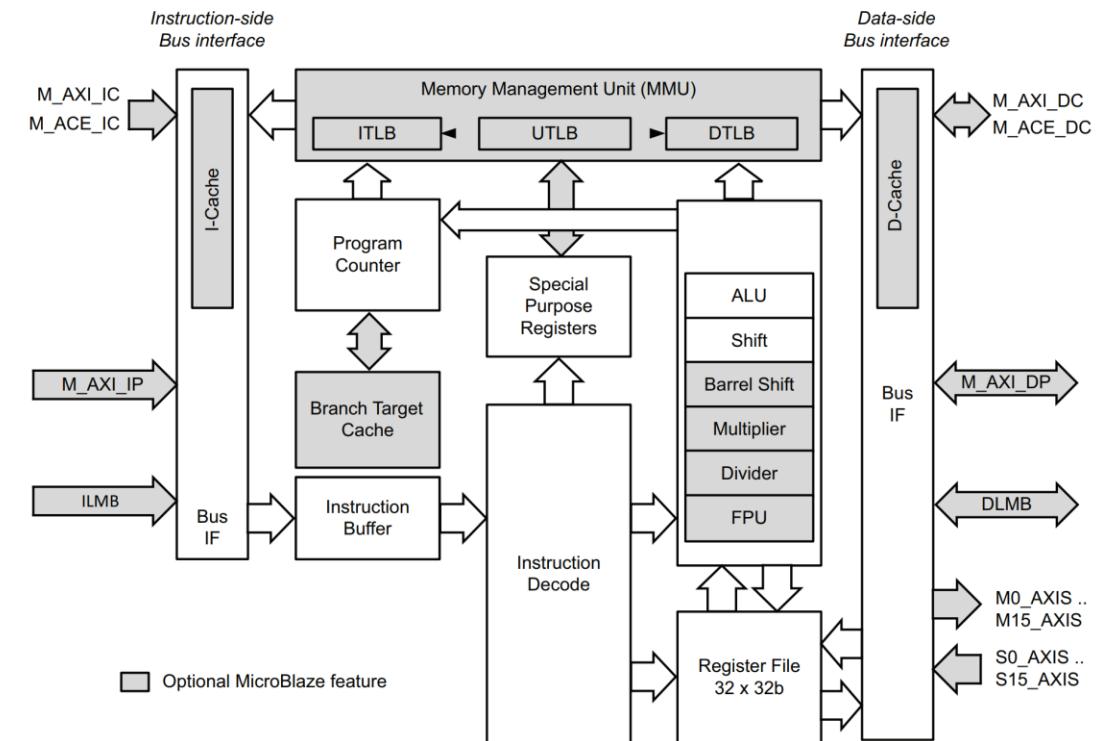
To complete this lab, we will need the following tools installed:

1. AMD Vivado™ software (version 2023.2)
2. AMD Vitis™ software (version 2023.2)
3. PetaLinux 2023.2

You will need either a native Linux development environment or virtual machine for the PetaLinux project creation. See my Hackster project [here](#) for step-by-step instructions on creating a VM.

AMD MicroBlaze™ Processor

- Softcore processor
- Harvard Architecture
- Scalable from small micro to running Linux
- Performance scales with configuration options
- Can be implemented in TMR solution (very useful for high rel)



What is PetaLinux? (A set of tools)

A build system:

- » A set of open-source cross-platform development command line and menu tools for use running under an x86 Linux Host OS
- » Just six PetaLinux commands (with many options):
petalinux-create, **petalinux-config**, **petalinux-build**, **petalinux-boot**, **petalinux-package**,
petalinux-util
- » The build system tailors embedded Linux to run on many different AMD platforms: AMD Zynq™ MPSoC, AMD Zynq 7000 SoCs and the AMD MicroBlaze™ processor. See:
<https://www.xilinx.com/support/answers/71653.html>
- » PetaLinux tools now use the open-source community Yocto system since v2016.3
 - This is very useful for embedded Linux developers!
- » Also includes an emulator (QEMU) that can emulate PetaLinux under x86 Linux
 - Very useful for Kernel and some device driver development

What is PetaLinux? (What comes with it)

Source code, libraries, applications and Yocto recipes

- » Mostly open-source based with a few exceptions for some hardware drivers
- » PetaLinux includes and manages the Linux kernel sources and libraries
- » Various hardware drivers and modules
- » Traditional Linux applications and utilities

Through Yocto, PetaLinux can make use of additional embedded Linux applications beyond what AMD supplies.

- » More on that later!

A more complete list of what is included by AMD can be found here:

<https://www.xilinx.com/support/answers/71653.html>

What is PetaLinux? (What goes in the firmware)

Includes Bootloaders, Trusted Platforms

- » U-Boot, FSBL, Arm® Trusted Firmware,

AMD Vivado™ Software Hardware Designs

- » PetaLinux device drivers work with Vivado IP blocks
- » PetaLinux imports hardware designs into the PL (FPGA)

A Linux OS that runs on the AMD Zynq™ MPSoC, AMD Zynq 7000 SoC (or AMD MicroBlaze™ processor) CPUs

- » 2018.3 uses Linux Kernel version 4.14
- » For single or multiple CPUs

Linux applications, device drivers and lots of configuration files

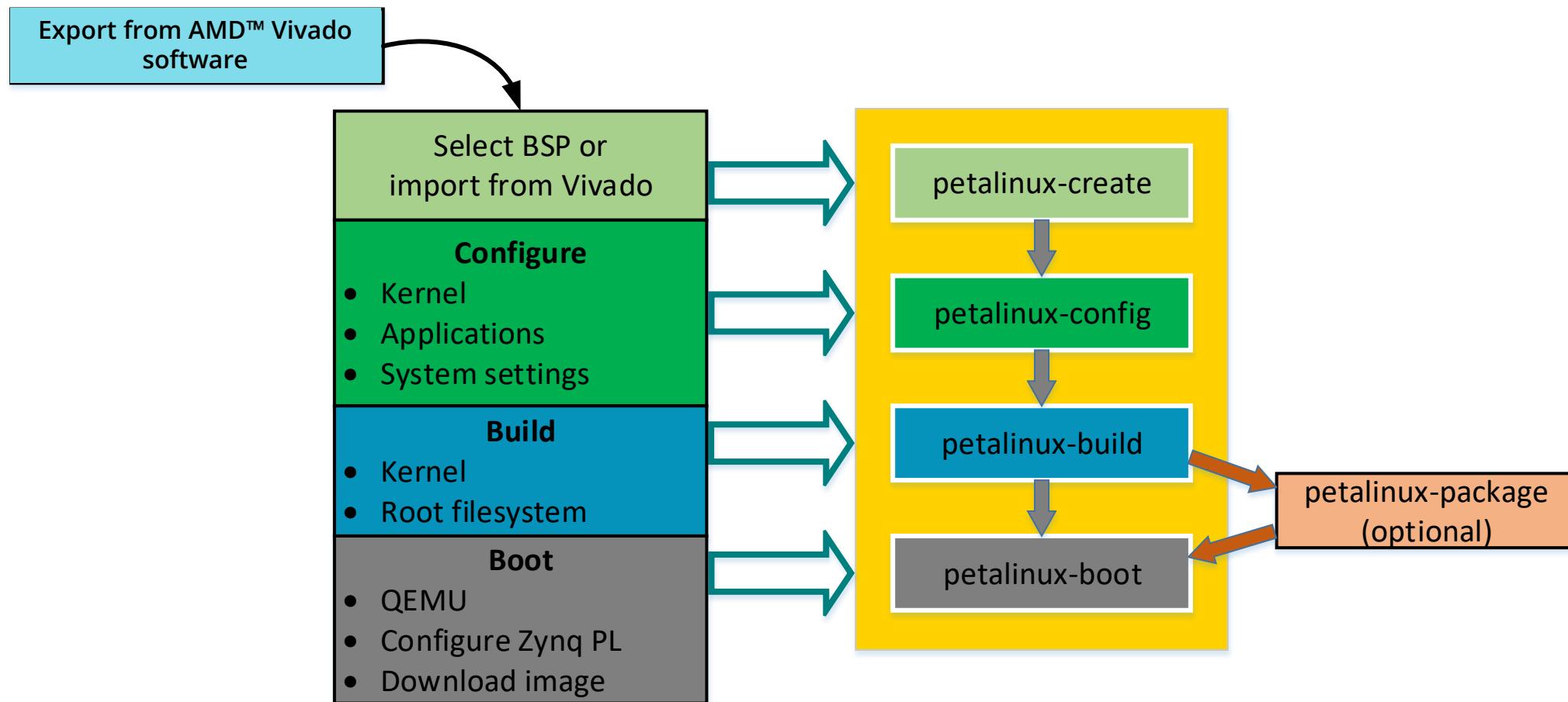
Detailed hardware and memory requirements can be found here: <http://www.xilinx.com/cgi-bin/docs/rdoc?v=latest;d=ug1144-petalinux-tools-reference-guide.pdf>

This is User Guide 1144: it is THE master reference book for PetaLinux!

PetaLinux Tools Testing, Release, and Documentation

- u-boot, Linux kernel, Xen hypervisor
 - Sources:
 - Kernel.org + AMD patches
 - Testing:
 - AMD silicon and AMD reference board features
 - Documentation:
 - Source code
 - Open source communities/forums
 - Answer Records for select interim issues
- Linux file system
 - Sources:
 - Yocto community code base
 - Testing:
 - Build verification
 - Testing of unique software and features from AMD
 - Release strategy:
 - Bug fixes with next Yocto release integration
 - Documentation:
 - Source code
 - Open source communities/forums

PetaLinux Tools Flow



What is The Yocto Project?

The Yocto Project is an open source collaboration project that provides templates, tools and methods to help you create custom Linux-based systems for embedded products regardless of the hardware architecture. It was founded in 2010 as a collaboration among many hardware manufacturers, open-source operating systems vendors, and electronics companies to bring some order to the chaos of embedded Linux development.

It's a framework, not a distribution

- » Allows users to create ***their own*** distributions
- » Contains reference distributions

Commonly used by several organizations:

- » Wind River Linux
- » Mentor Embedded Linux
- » Just about everyone else...

Uses a build tool called **bitbake**

PetaLinux 2018.3 abstracts most of the bitbake details, commands, etc.



Why Yocto?

The challenge...

- » Linux is becoming increasingly popular for embedded systems
- » Non-commercial and commercial embedded Linux has many distros

Leads to the result...

- » Developers spend lots of time porting or making build systems
- » Leaves less time/money to develop interesting software features
- » The industry needs a common build system and core technology
- » Industry leaders joined together in 2010 to form the Yocto Project

With these benefits...

- » Less time spent on things which don't add value (build system, core Linux components)
- » Linux grows more in embedded
- » More time spent on product differentiation and interesting software

Yocto Layers, Recipes, and Bitbake

Yocto builds and assembles Linux images based on *layers*

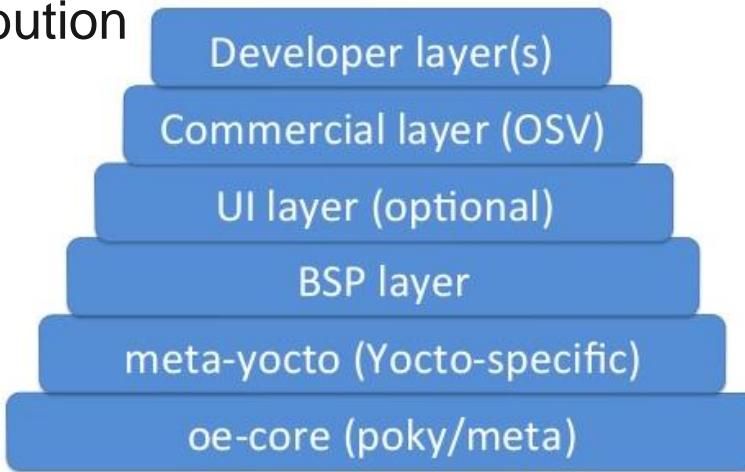
- » Layers are named **meta-something**, such as:
 - **meta-yocto**, **meta-xilinx**, **meta-xilinx-tools**, etc.

Layers consist of *recipes*

- » Recipes describe individual components of a target Linux distribution
- » Can be written to do all sorts of things:
 - Apply source code patches to the kernel
 - Copy files to a target root filesystem
 - Build / install user applications
 - *Build and install kernel drivers for wireless modules*

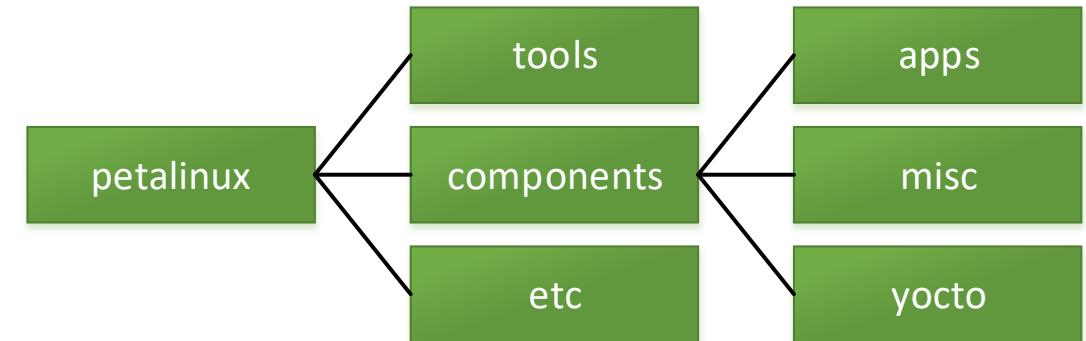
PetaLinux 2018.3 specifics

- » Bitbake is abstracted using familiar commands, e.g. petalinux-build, etc.
- » Layer set aside for user customization: <project>/project-spec/meta-user



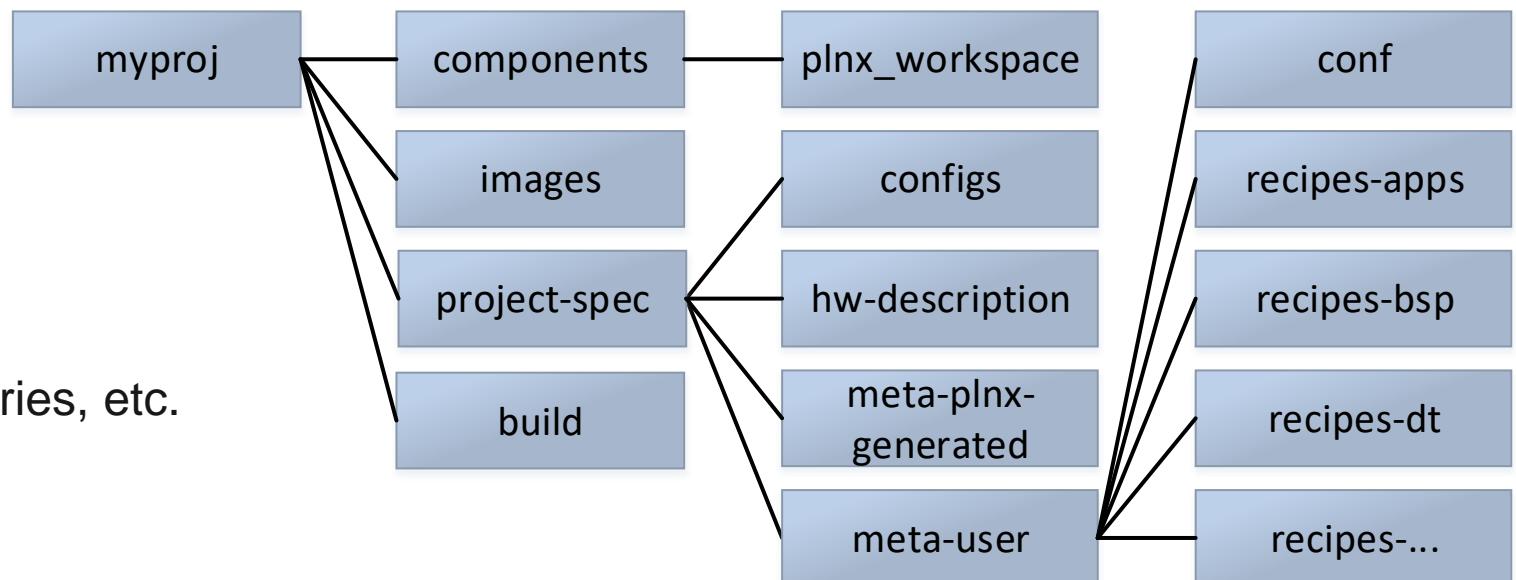
PetaLinux Install and Project Taxonomy

Tools and default source components in install area

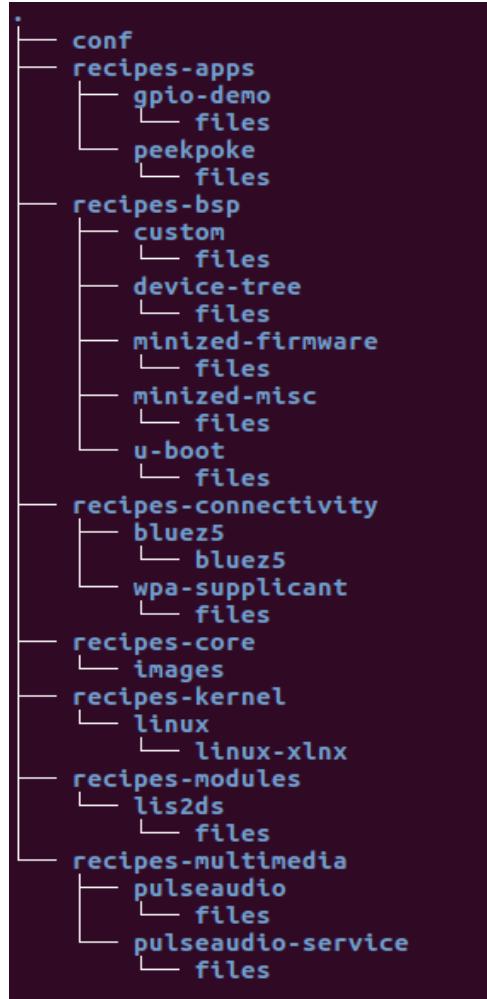


Per-project directory

- » Configuration files
- » Custom recipes
 - Applications, scripts, libraries, etc.
- » Build products



PetaLinux Project Example



meta-user Sub-folder Name	Purpose
recipes-apps	Custom user applications
recipes-bsp	Customizations to the u-boot bootloader, device tree, hardware platform and user file additions to the root file system
recipes-connectivity	Networking protocols and support files
recipes-core	Configuration files to tell PetaLinux what recipes to add to the system
recipes-kernel	Kernel patches to correct, customize or enhance baseline kernel functionality
recipes-modules	Custom device drivers
recipes-multimedia	Custom application and support files for multimedia functionality

Device Tree

DTS file

- » Device Tree Source
- » Textual description of system device tree

DTB

- » Device Tree Blob
- » Compiled, binary representation of the DTS

DTC

- » Device Tree Compiler
- » Provided in PetaLinux installation
- » Converts the DTS to a DTB
- » Can de-compile a DTB back to a DTS

Device Tree Structure

A DTB is provided to the kernel at boot time

- » Kernel startup code parses the tree
- » Instantiates and initializes kernel appropriately
 - System memory location/size
 - Device types, address mappings, configuration options

Kernel infrastructure binds drivers to devices

- » Matched against the "compatible" property in the device node
- » Device driver declares known "compatible" devices

Greatly simplifies the problem of
device ↔ driver binding

Device Tree Source Example (shown on right)

```
40 &sdhci0 {  
41     status = "okay";  
42     bus-width= <4>;  
43     xlnx,has-cd = <0x0>;  
44     xlnx,has-power = <0x0>;  
45     xlnx,has-wp = <0x0>;  
46     non-removeable;  
47     wifi-host; /* adding for compatibility w/ bcmdhd driver */  
48 };  
49  
50 /* Define the external I2C devices and their names and addresses */  
51 &axi_iic_0 {  
52     #address-cells = <0x1>;  
53     #size-cells = <0x0>;  
54     compatible = "xlnx,xps-iic-2.00.a";  
55     i2c5@1d {  
56         #address-cells = <0x1>;  
57         #size-cells = <0x0>;  
58         compatible = "stm,lis2ds12";  
59         reg = <0x1d>;  
60     };  
61 };  
62  
63 /* QSPI addresses are defined with petalinux-config, but here they are overwritten so that one can  
program the flash internally */  
64 &qspi {  
65     #address-cells = <1>;  
66     #size-cells = <0>;  
67     flash0: flash@0 {  
68         compatible = "micron,m25p80";  
69         reg = <0x0>;  
70         #address-cells = <1>;  
71         #size-cells = <1>;  
72         spi-max-frequency = <50000000>;  
73         partition@0x00000000 {  
74             label = "boot";  
75             reg = <0x00000000 0x00ff0000>;  
76         };  
77     };  
78 };
```

Device Tree Creation

PetaLinux includes an automated device tree generator

- » Generates device tree based on hardware design
- » All IP cores and properties exported in DTS

Kernel build process compiles DTS → DTB and links into kernel image

- » Auto-generated DTS files in
`<petalinux_project>/components/plnx_workspace/device-tree-generation/`
 - Do not edit! Auto-generated on import of hardware platform from Vivado
- » User-editable system-top.dts DTS file in
`<petalinux_project>/project-spec/meta-user/recipes-dt/device-tree/files/`
 - User overwrite of default parameters and addition of new parameters
 - Changes to the system-top.dts will not be overwritten by subsequent petalinux-config

Role of the BSP

- Setting up the device tree for a board is important, **as it** can take considerable time.
- It is unlikely there will only ever be one project created
- The Board Support package provides a portable way of configuring Petalinux projects for the board with all the interfaces and applications correctly set up
- Single petalinux command can be used to create the BSP

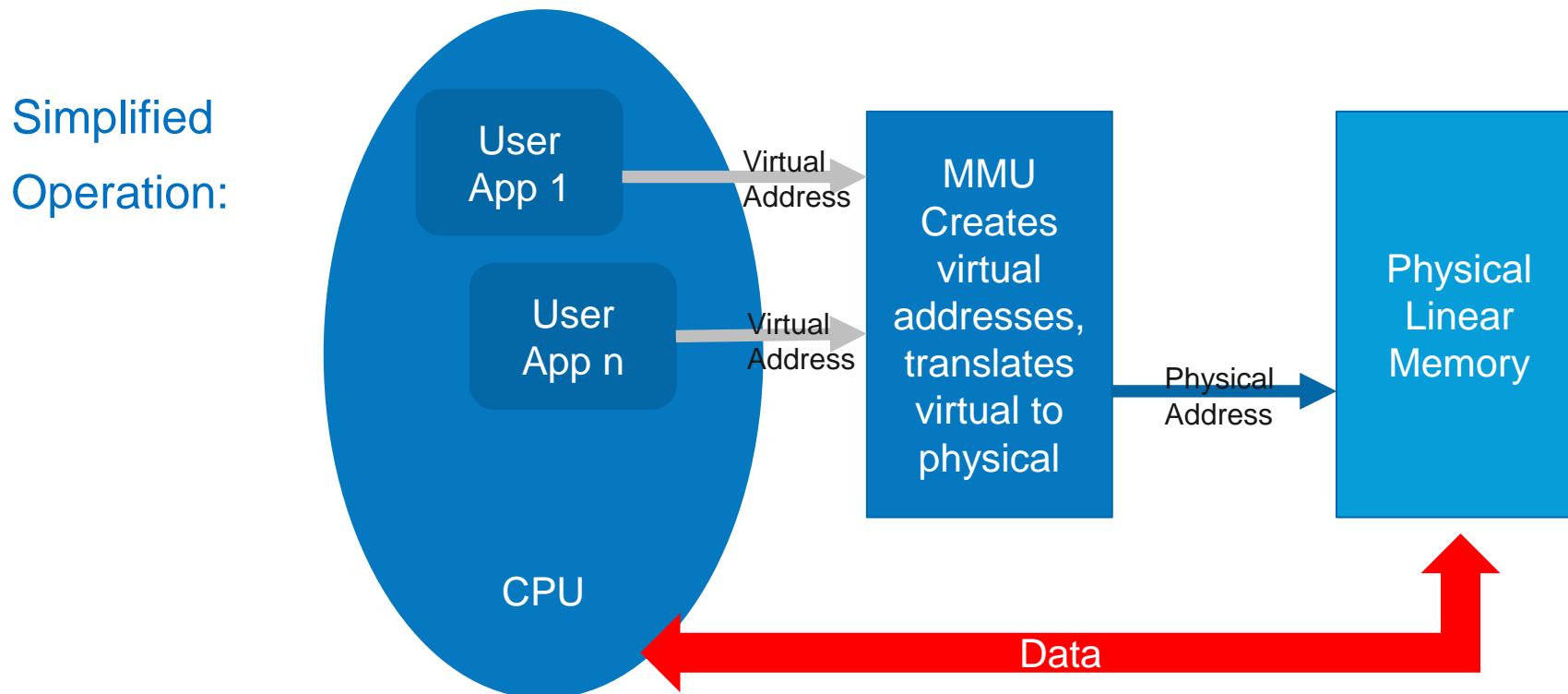
Key Concept: Memory Management Unit

User applications access program and data from virtualized (pseudo) addresses

The MMU unit of the AMD Zynq™ MPSoC is the translator from virtual to physical addresses

Each user program runs as if it is the only program on the CPU

Each user program has its own unique memory mapped address space



PetaLinux and Yocto Bitbake

PetaLinux commands abstract the details of the bitbake commands

- » Creation of new software applications, kernel modules (drivers), scripts, etc.
- » Add new bitbake recipes
- » Additions to configuration menus
- » Build, clean, config
- » Installation onto the root file system

Easy edits to makefile and recipe as application complexity grows

- » Add source files, etc.

PetaLinux command to build the root file system becomes bitbake command...

```
$ petalinux-build -c rootfs  
↓  
$ bitbake petalinux-user-image -c do_rootfs
```

What is in a Bitbake Recipe?

Descriptive information about the package

The version of the recipe

Existing Dependencies

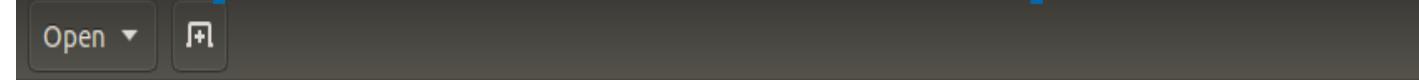
Where the source code resides

Whether the source code requires any patches

How to compile the source code

Where on the target machine to install the package being compiled

Simple Bitbake Recipe



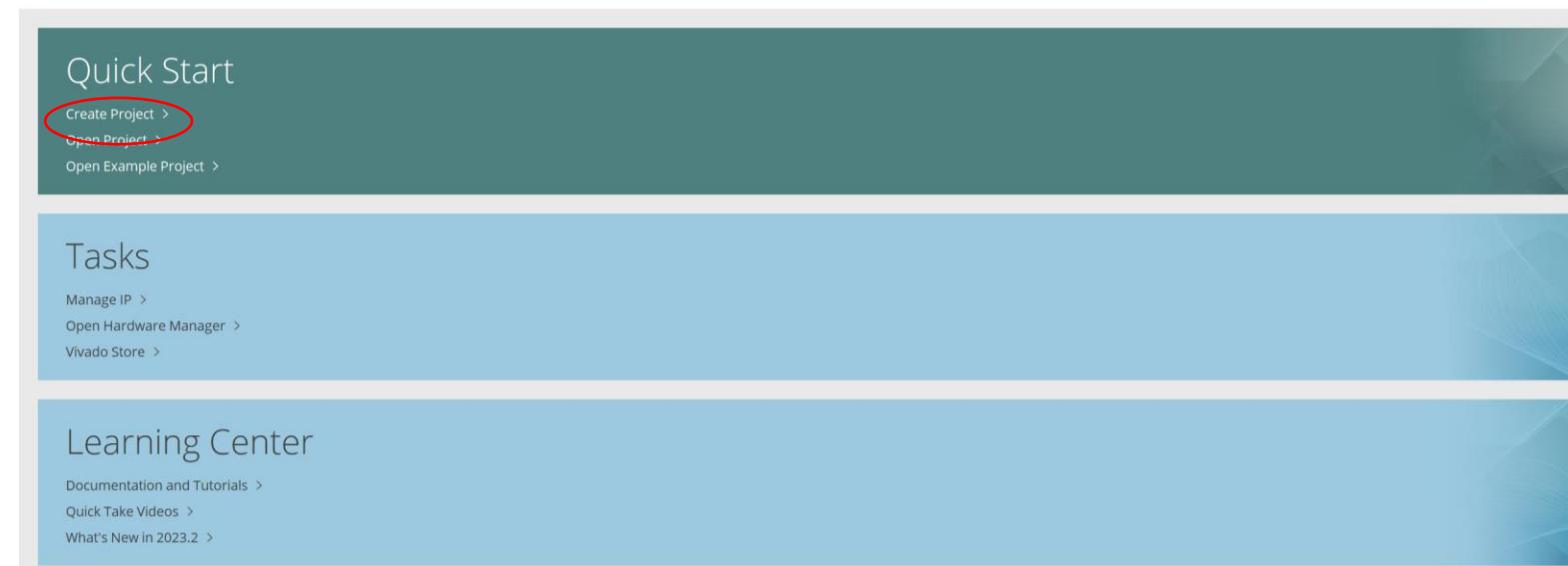
```
#  
# This file is the lsm6dls-sensor recipe.  
  
#  
  
SUMMARY = "Simple lsm6dls-sensor application"  
SECTION = "PETALINUX/apps"  
LICENSE = "MIT"  
LIC_FILES_CHKSUM = "file://${COMMON_LICENSE_DIR}/MIT;md5=0835ade698e0bcf8506ecda2f7b4f302" ← Location and MD5 checksum of the license file for the application  
  
SRC_URI = "file://main.c \  
          file://Makefile \  
          "  
          ← Location of the source file(s) and Makefile  
  
S = "${WORKDIR}" ←  
FILES_${PN} += "/home/root/*" ←  
do_compile() {  
    oe_runmake  
}  
  
do_install() {  
    install -d ${D}${bindir} ←  
    install -m 0755 lsm6dls-sensor ${D}/home/root/ ← Specify where the application should be installed  
}
```

Embedded Linux Lab

Lab: Embedded Linux

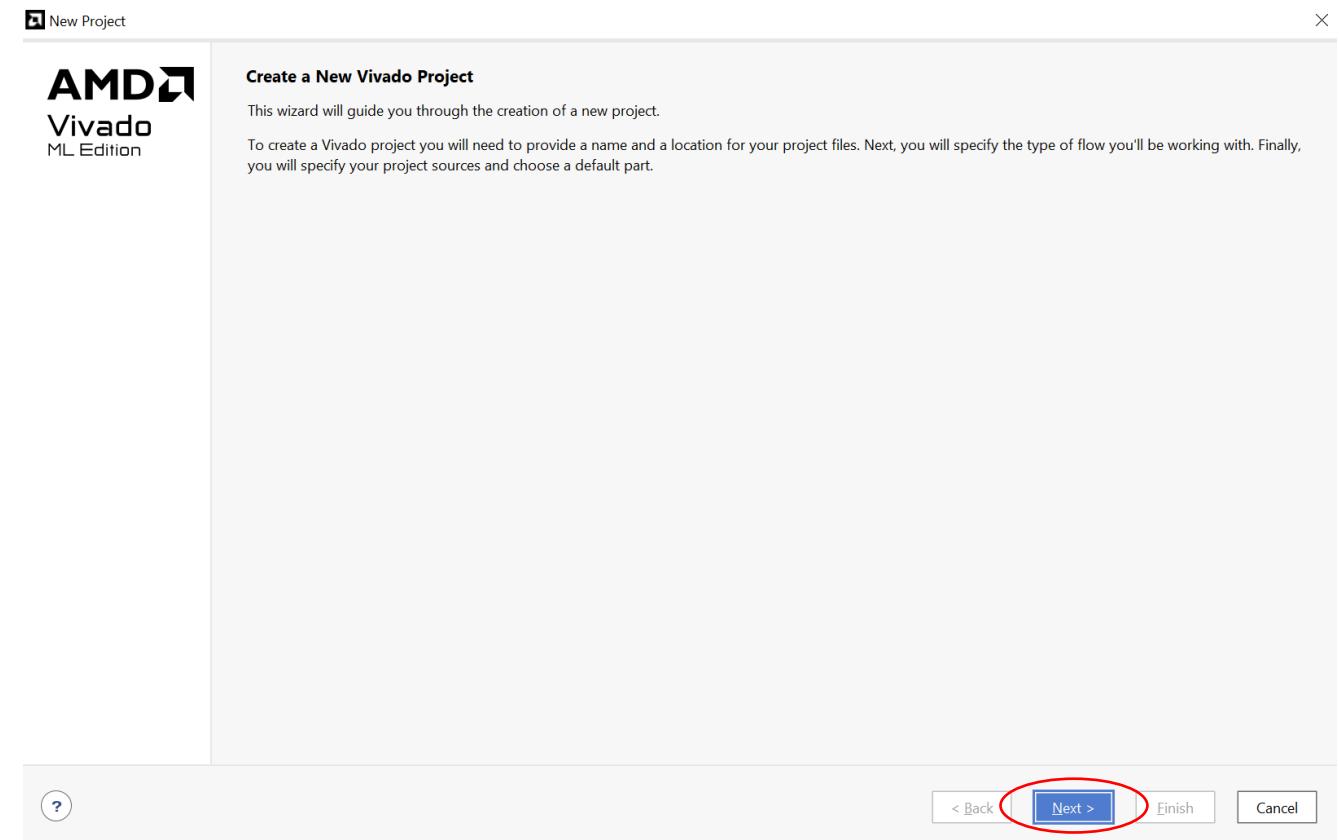
1. Open AMD Vivado™ software (version 2023.2)

2. Click create new project



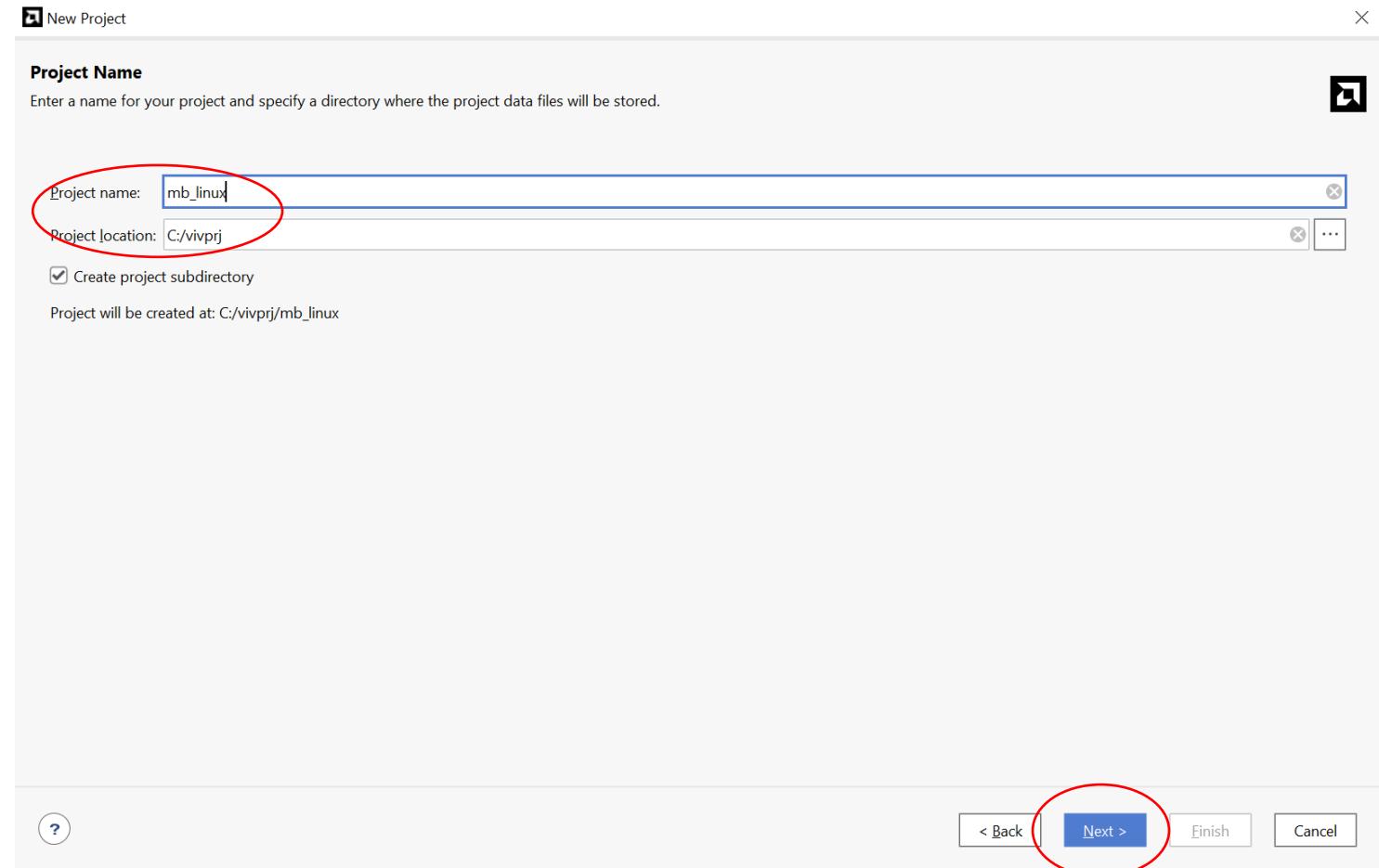
Lab: Embedded Linux

1. Click Next



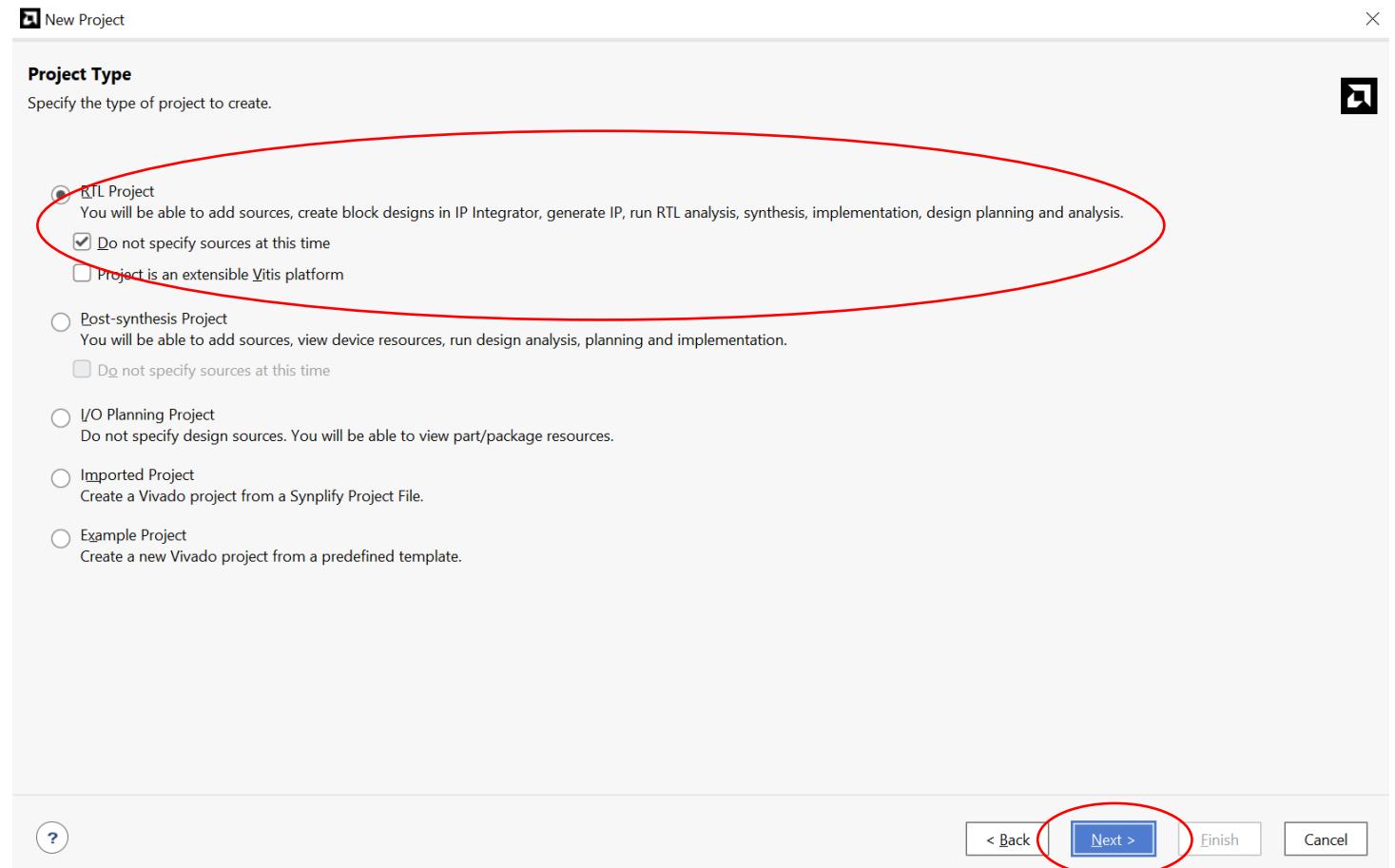
Lab: Embedded Linux

1. Enter a project name
e.g. mb_linux
2. Enter a project location
3. Click next



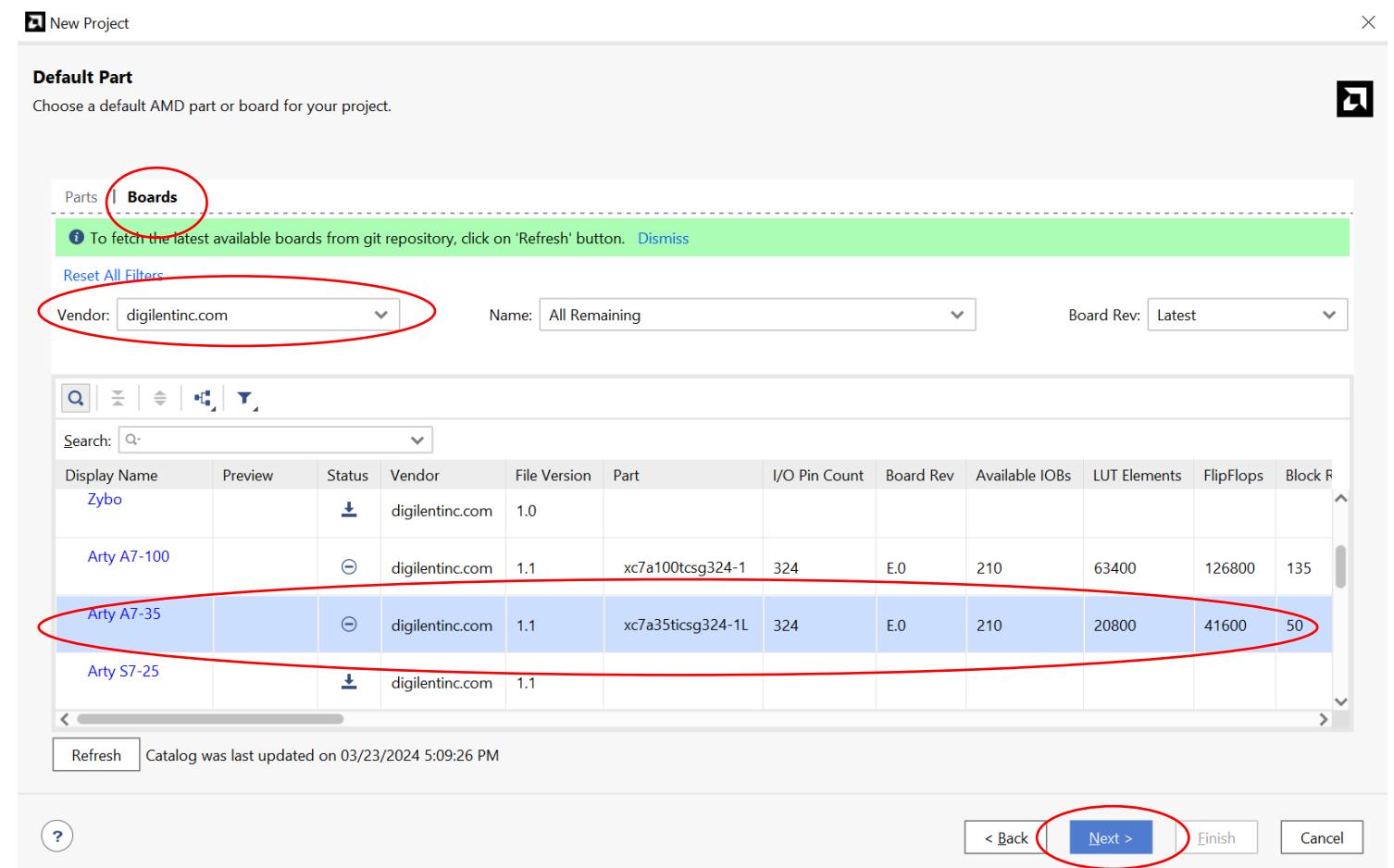
Lab: Embedded Linux

1. Select RTL project and ensure option “*Do not specify sources at this time*” is checked
2. Click Next



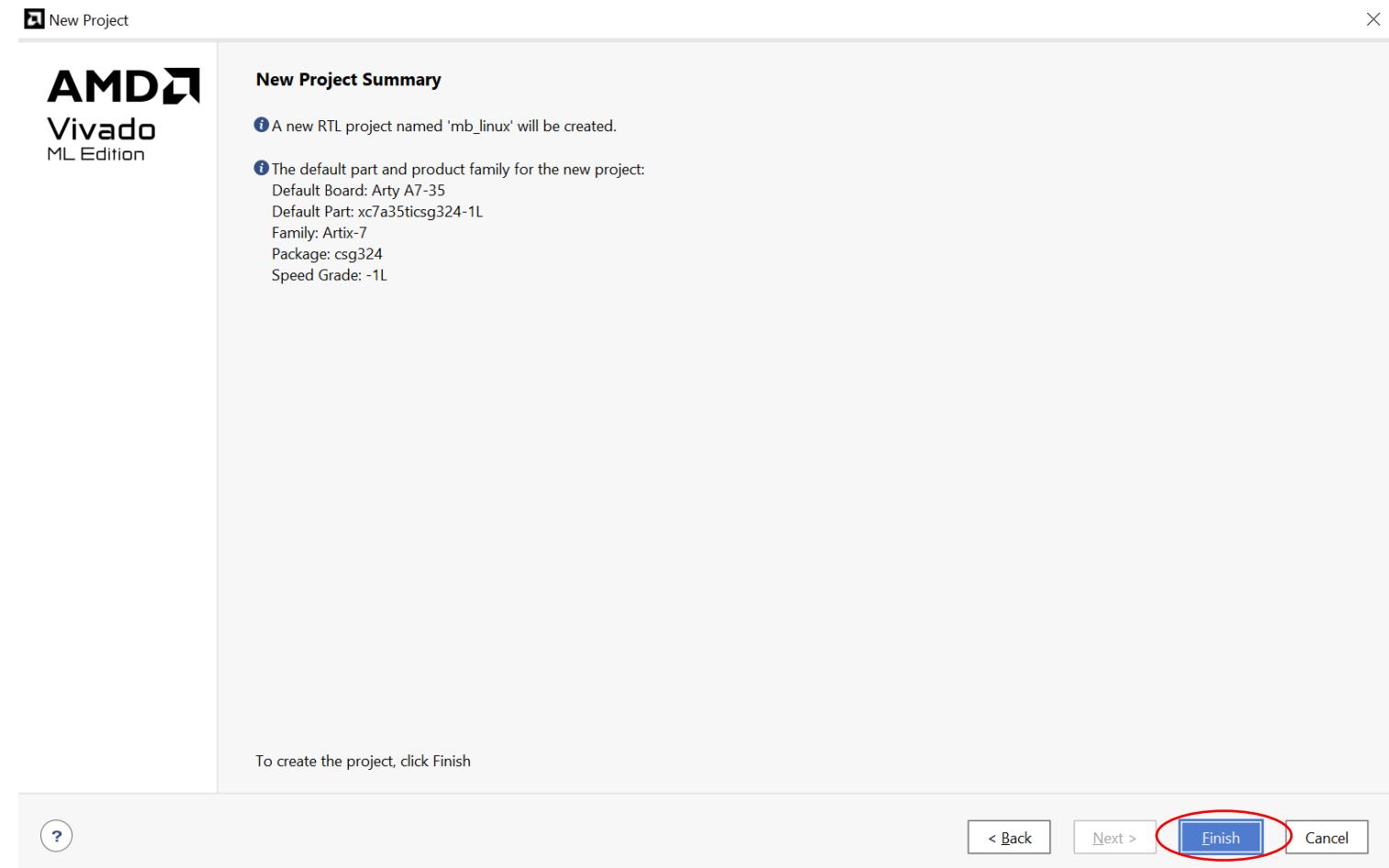
Lab: Embedded Linux

1. Click on boards
2. Select
DiligentInc.com as
the vendor
3. Select the Arty A7-35
or A7-100
4. Click Next



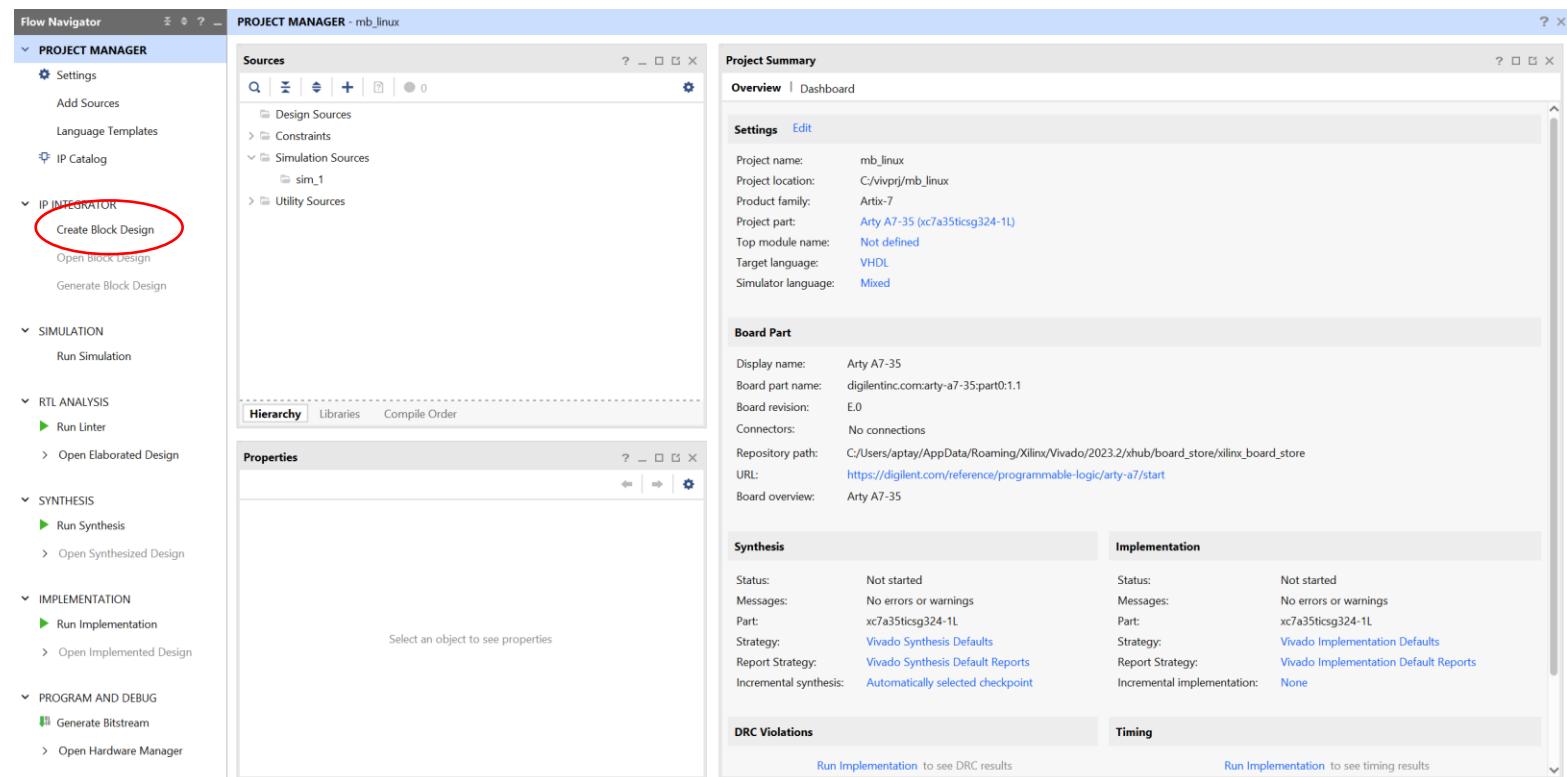
Lab: Embedded Linux

1. Click Finish



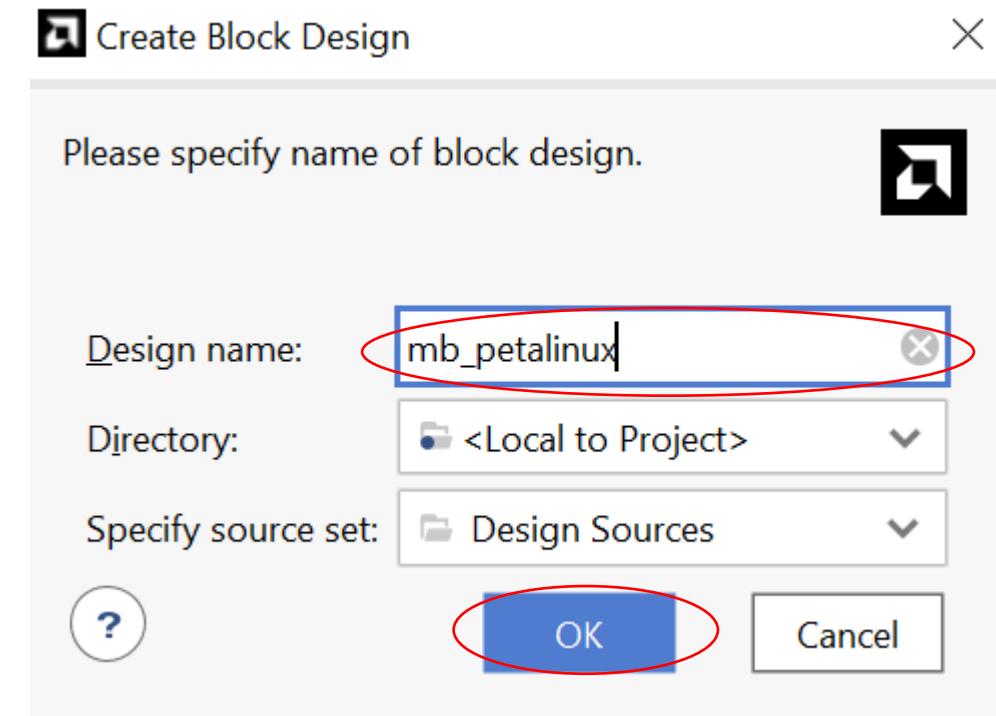
Lab: Embedded Linux

1. Click Create Block Diagram



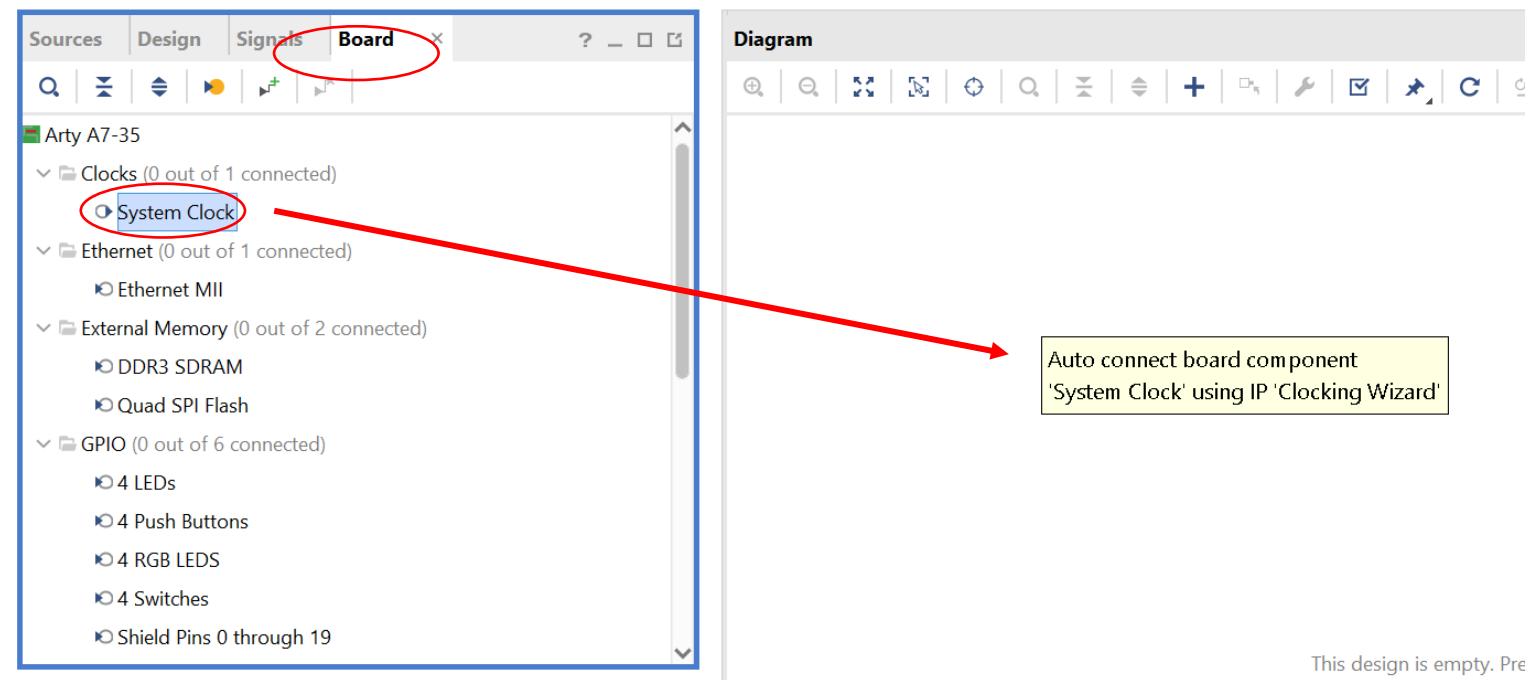
Lab: Embedded Linux

1. Enter the design name mb_petalinux
2. Click OK



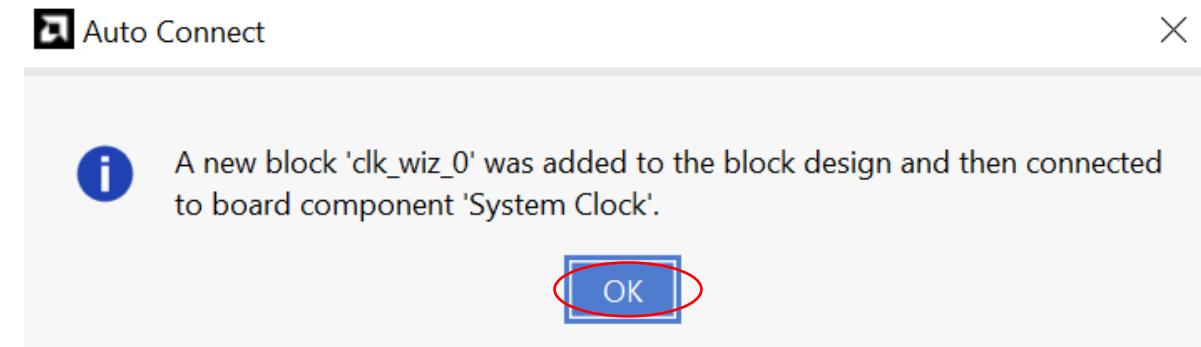
Lab: Embedded Linux

1. On the board tab, select the System Clock left click, hold and drag on to the diagram



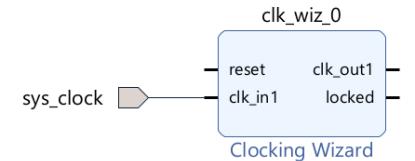
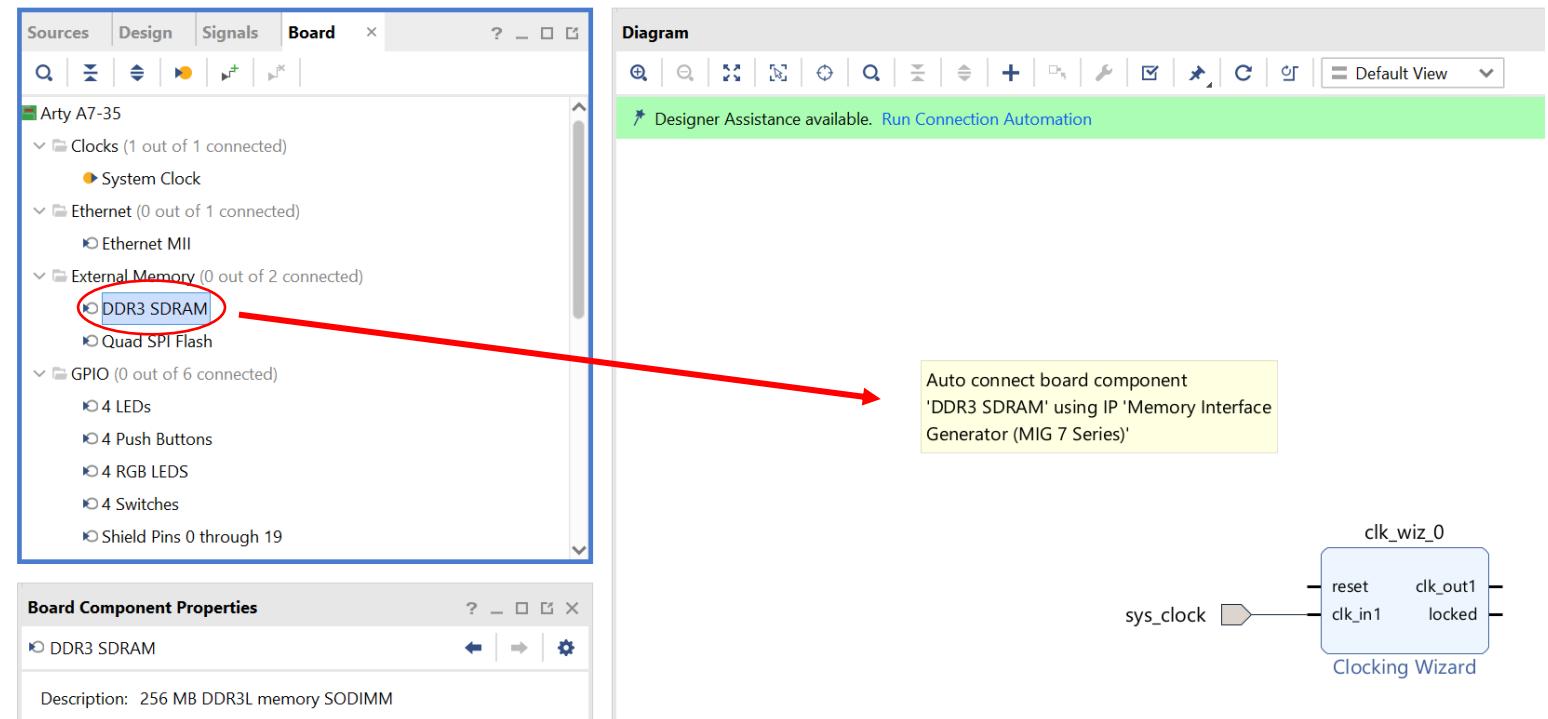
Lab: Embedded Linux

1. Click OK



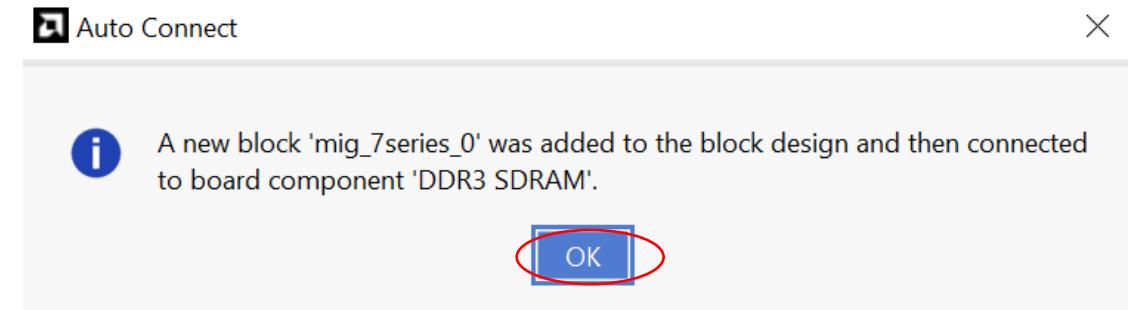
Lab: Embedded Linux

1. On the board tab, select the DDR3 SDRAM left click, hold and drag on to the diagram



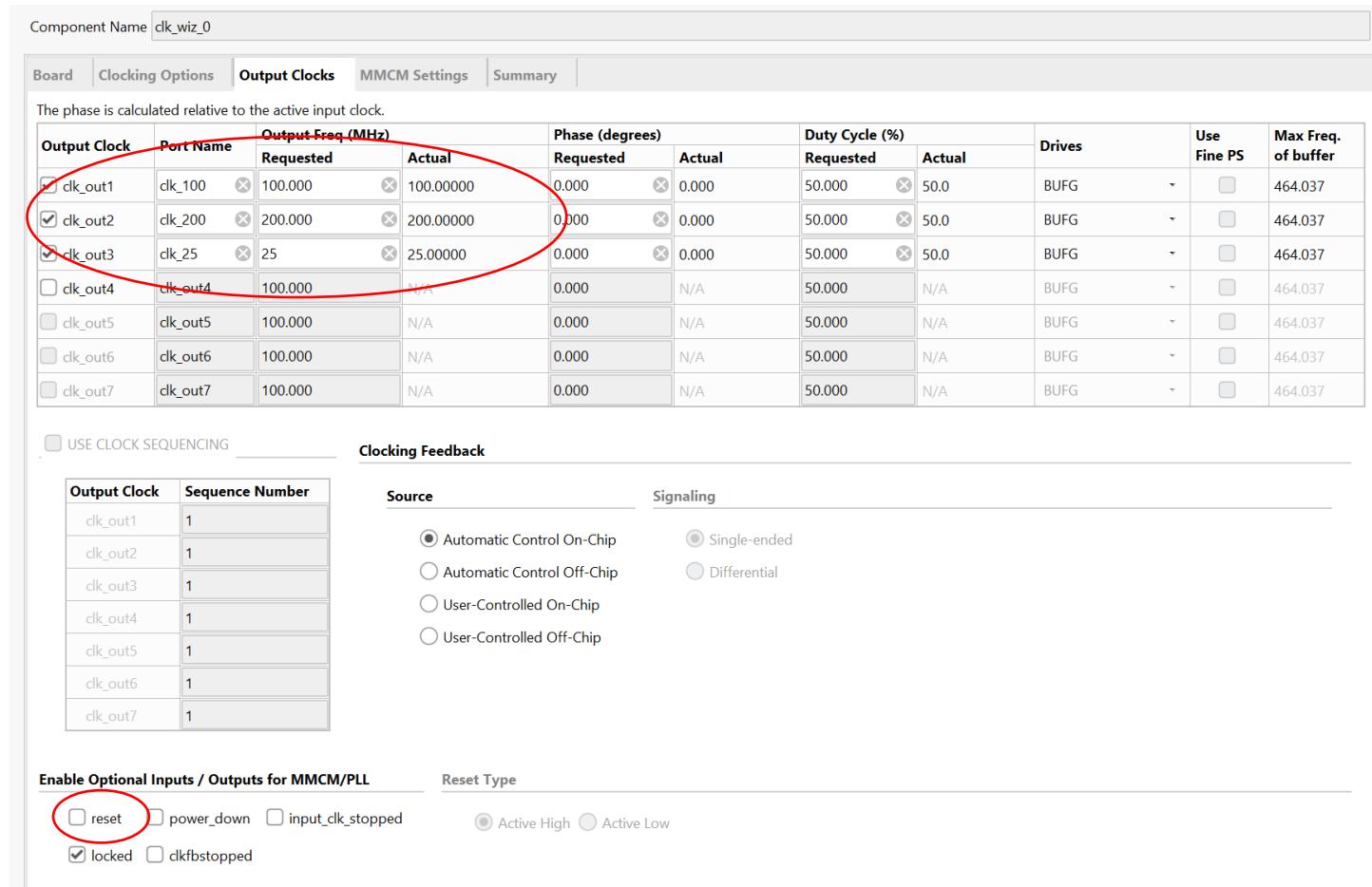
Lab: Embedded Linux

1. Click Ok



Lab: Embedded Linux

1. Re-customize the clock wizard
2. Set clk1 100 MHz
3. Set clk2 200 MHz
4. Set clk3 25 MHz
5. Uncheck the reset
6. Click OK



The screenshot shows the 'clk_wiz_0' component configuration window. The 'Output Clocks' tab is selected, displaying a table of output clocks:

Output Clock	Port Name	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)		Drives	Use Fine PS	Max Freq. of buffer
		Requested	Actual	Requested	Actual	Requested	Actual			
<input checked="" type="checkbox"/> clk_out1	clk_100	100.000	100.00000	0.000	0.000	50.000	50.0	BUFG	<input type="checkbox"/>	464.037
<input checked="" type="checkbox"/> clk_out2	clk_200	200.000	200.00000	0.000	0.000	50.000	50.0	BUFG	<input type="checkbox"/>	464.037
<input checked="" type="checkbox"/> clk_out3	clk_25	25	25.00000	0.000	0.000	50.000	50.0	BUFG	<input type="checkbox"/>	464.037
<input type="checkbox"/> clk_out4	clk_out4	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	<input type="checkbox"/>	464.037
<input type="checkbox"/> clk_out5	clk_out5	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	<input type="checkbox"/>	464.037
<input type="checkbox"/> clk_out6	clk_out6	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	<input type="checkbox"/>	464.037
<input type="checkbox"/> clk_out7	clk_out7	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	<input type="checkbox"/>	464.037

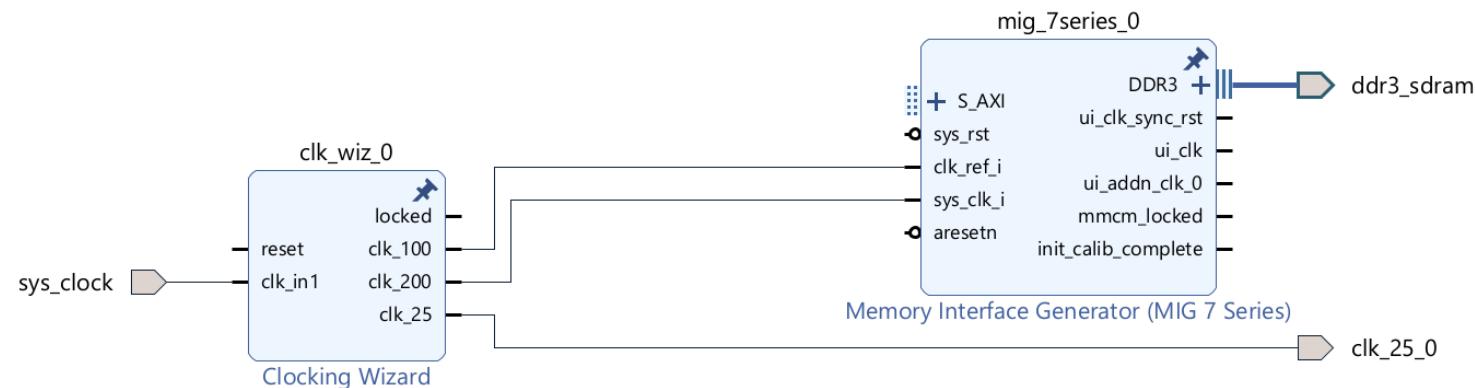
The 'Clocking Options' tab is also visible, showing the following settings:

- USE CLOCK SEQUENCING:
- Clocking Feedback:
 - Source: Automatic Control On-Chip ()
 - Signaling: Single-ended ()
 - Other options: Automatic Control Off-Chip, User-Controlled On-Chip, User-Controlled Off-Chip

In the 'Enable Optional Inputs / Outputs for MMCM/PLL' section, the 'reset' checkbox is circled in red, indicating it should be unselected.

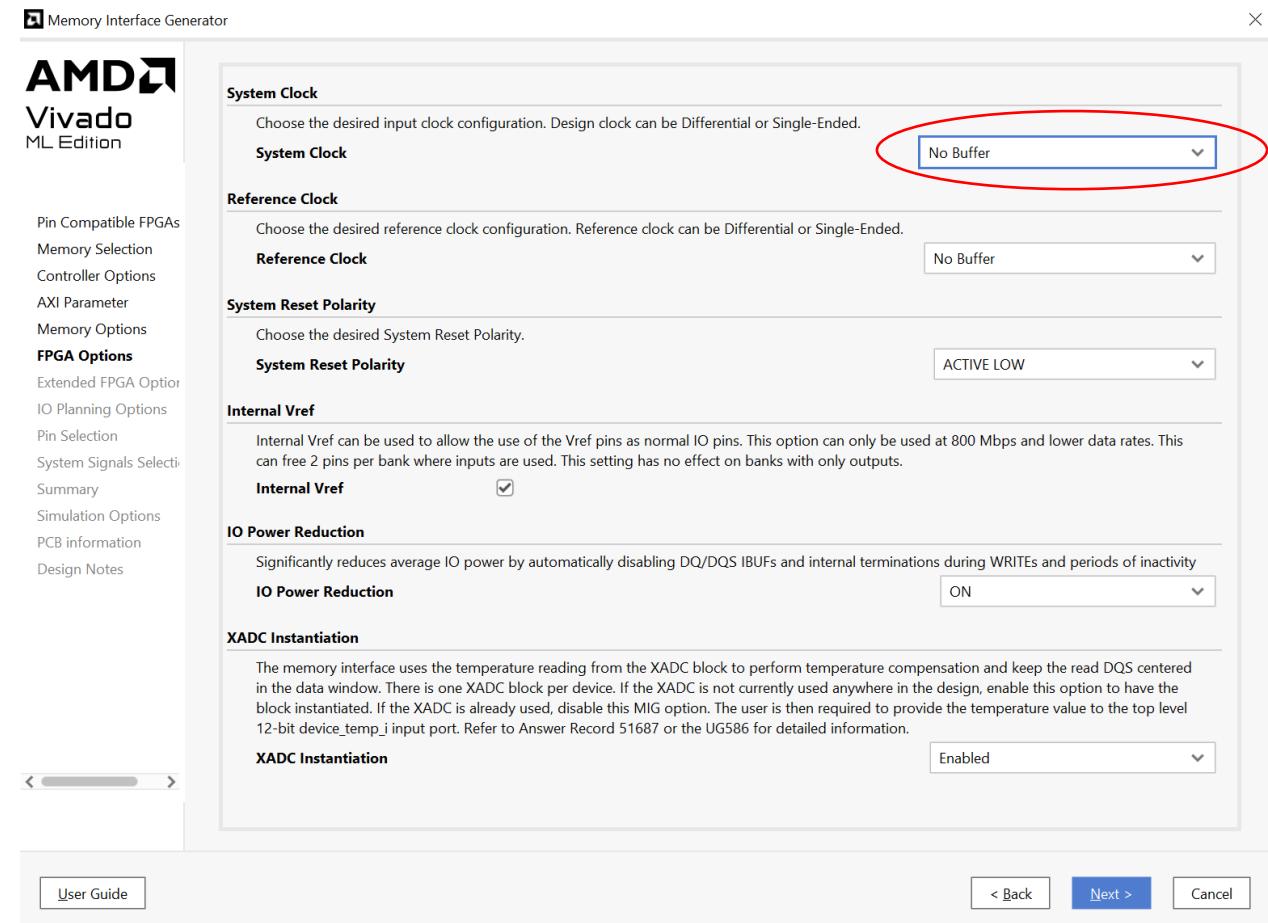
Lab: Embedded Linux

1. Delete the connections on the MIG
2. Connect 100MHz to clk_ref_i
3. Connect 200MHz to sys_clk_i
4. Make Clock 25MHz external



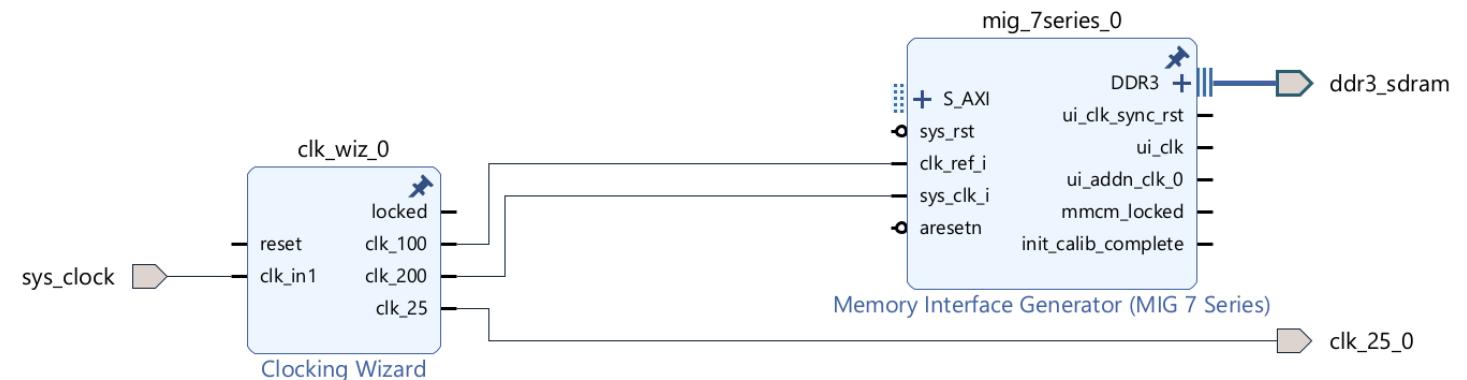
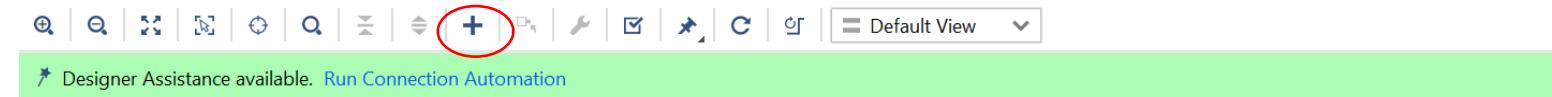
Lab: Embedded Linux

1. Double click on the MIG DDR3 and on the FPGA options page
2. Change the system clock to no buffer



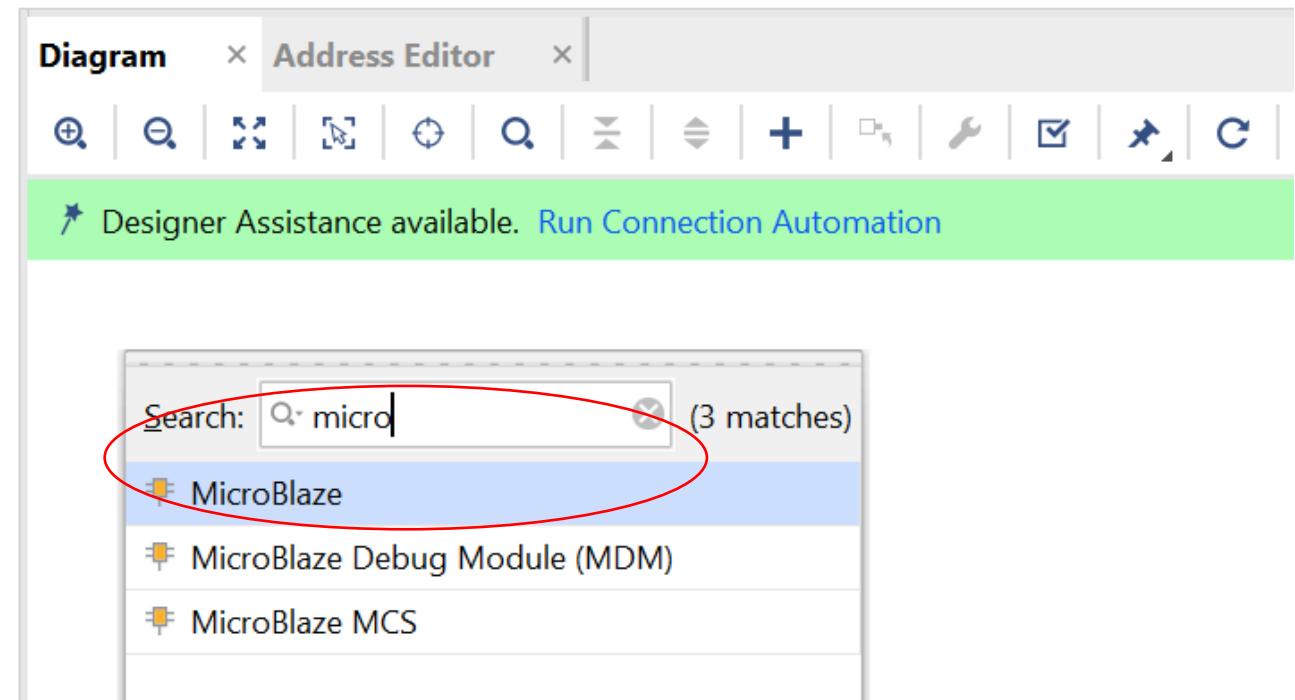
Lab: Embedded Linux

1. Click on the + symbol to open the IP selection library



Lab: Embedded Linux

1. In the search bar enter “Micro”
2. Select MicroBlaze and double click on it.



Lab: Embedded Linux

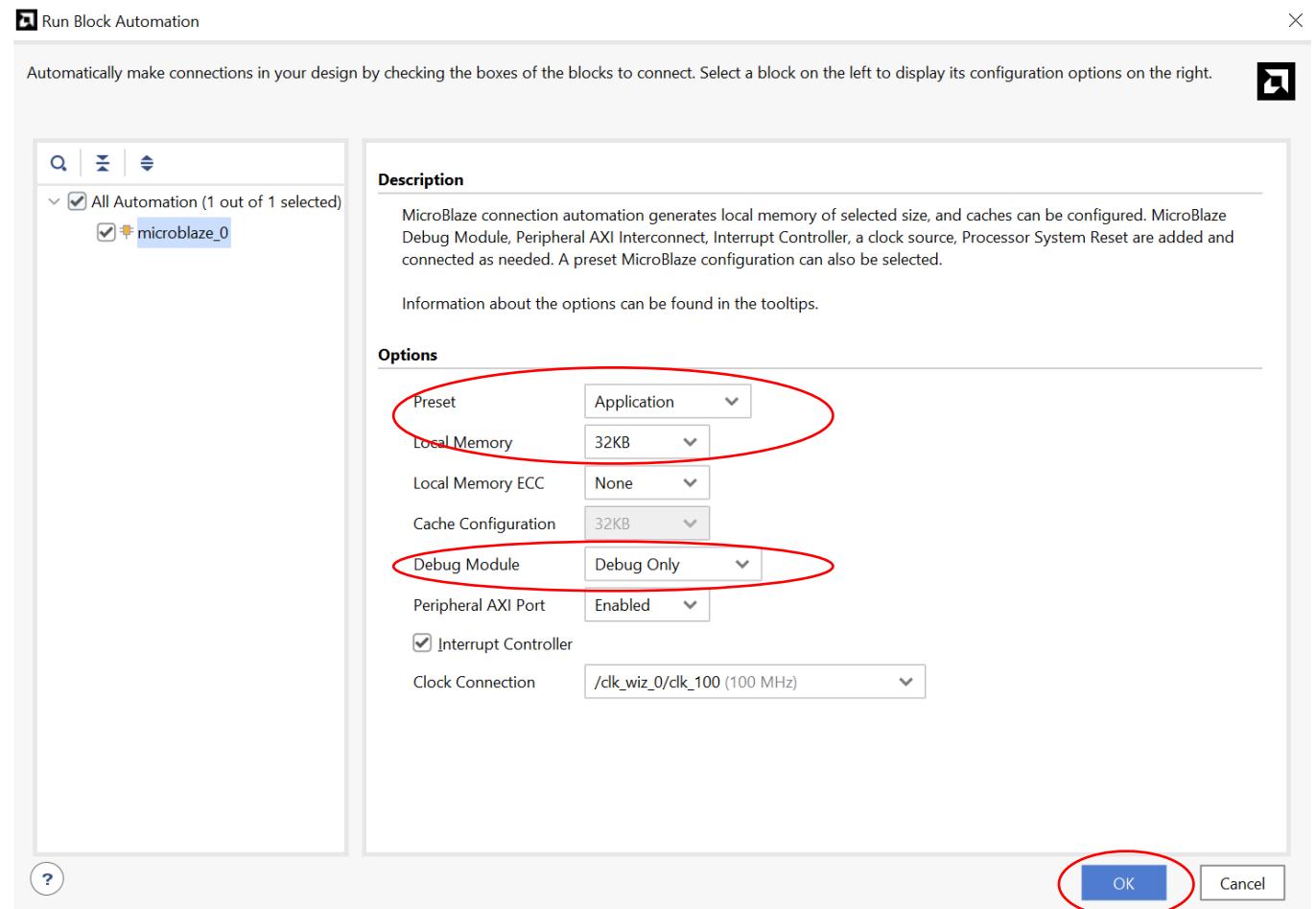
1. Click on Run Block Automation

* Designer Assistance available. [Run Block Automation](#) [Run Connection Automation](#)



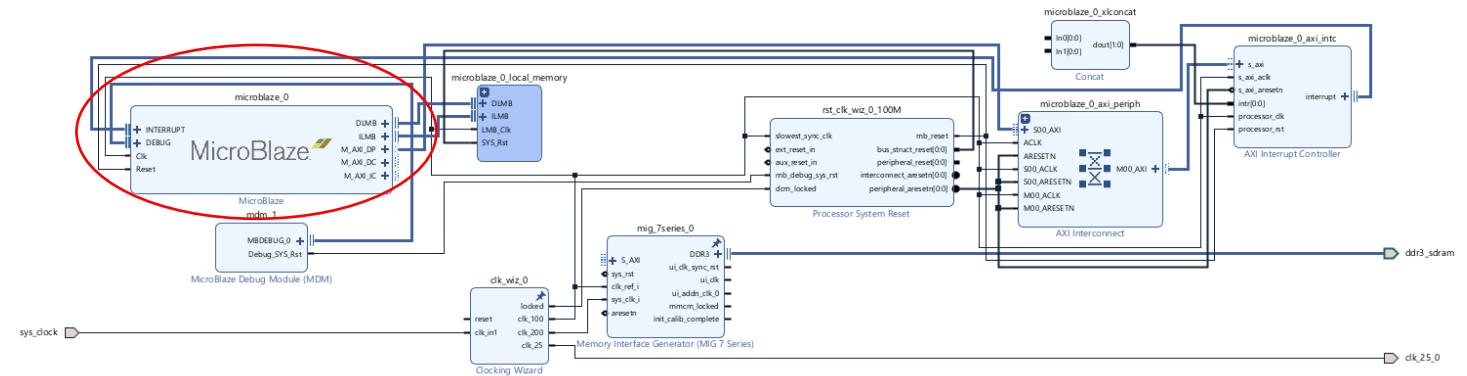
Lab: Embedded Linux

1. Select the preset to Application
2. Set Local Memory to 32KB
3. Set Debug Mode to Debug Only



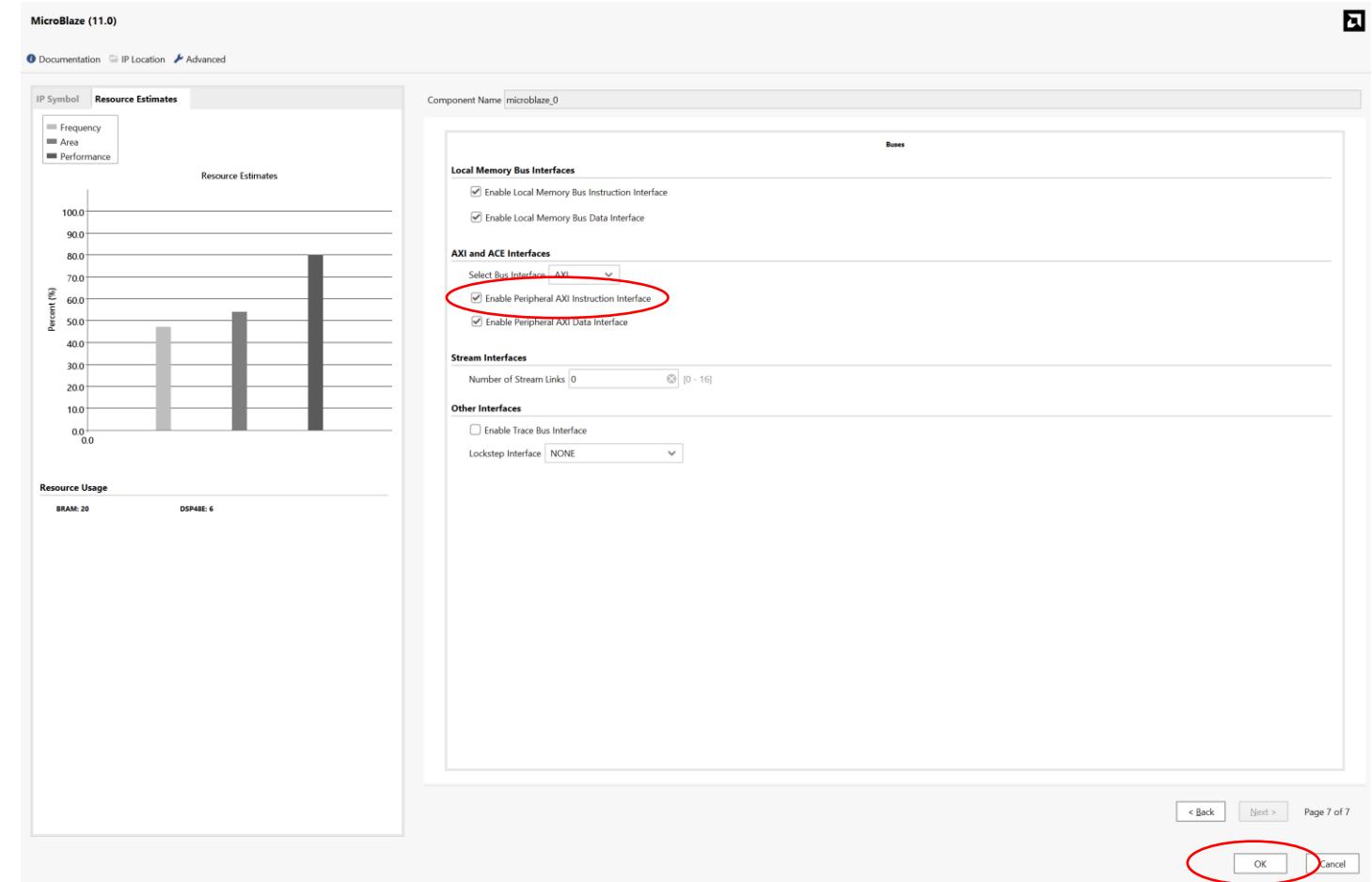
Lab: Embedded Linux

1. Double click on the MicroBlaze to re-customize



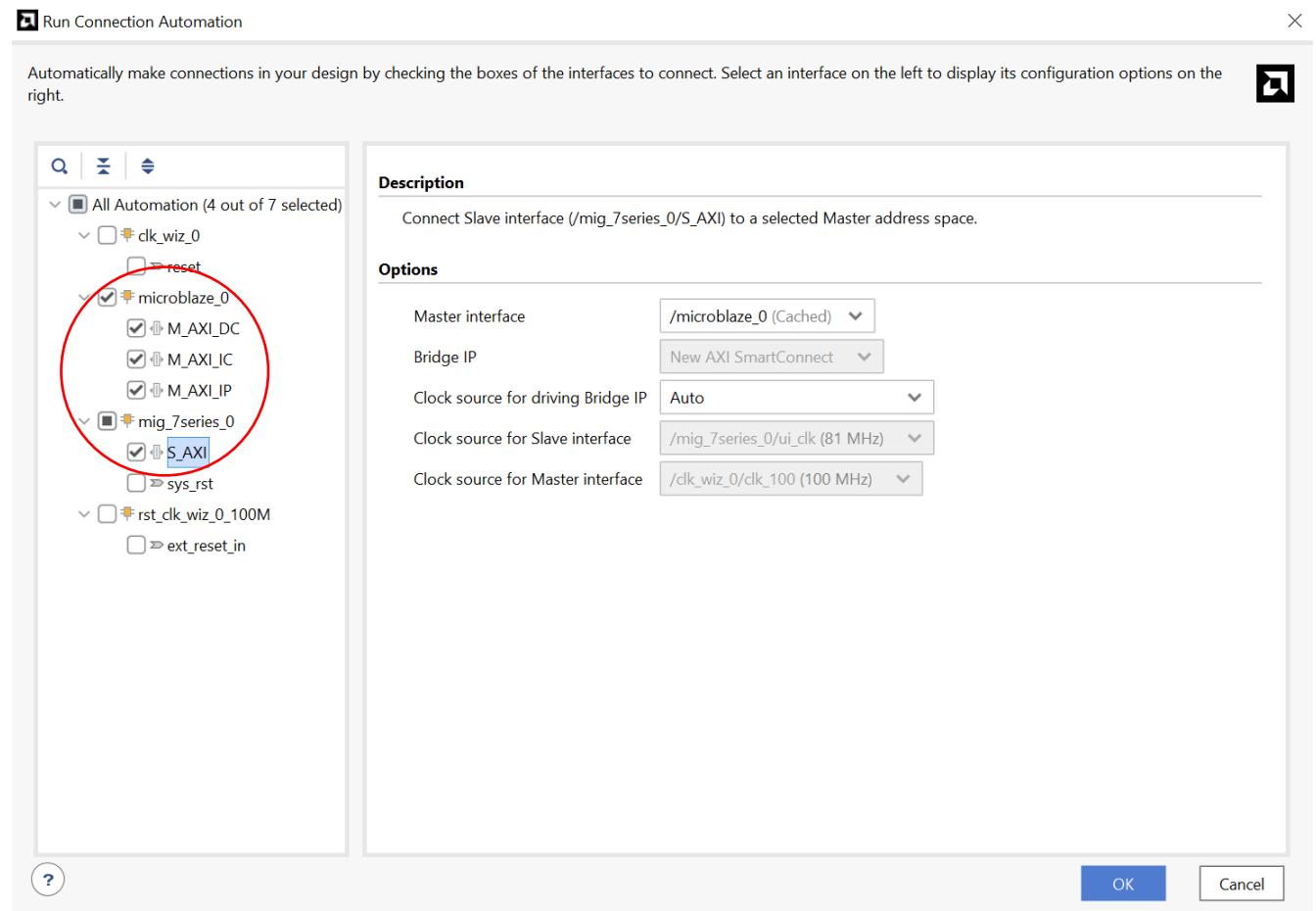
Lab: Embedded Linux

1. On page 7
2. Enable the Peripheral AXI Instruction Interface
3. Click OK



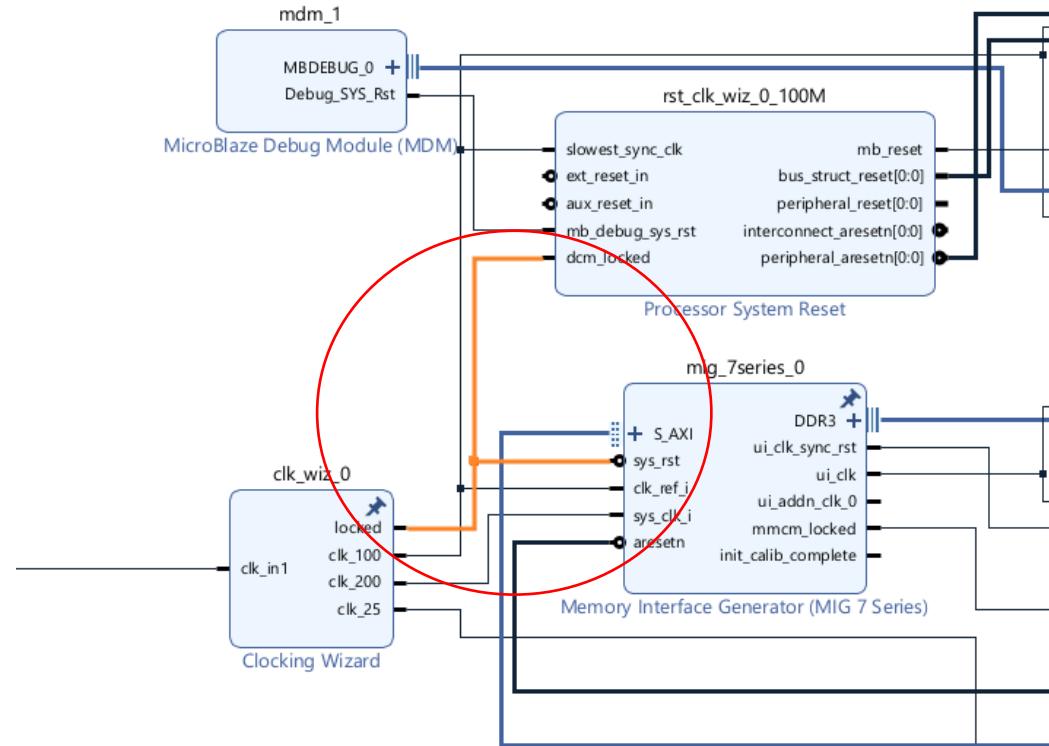
Lab: Embedded Linux

1. Click Next



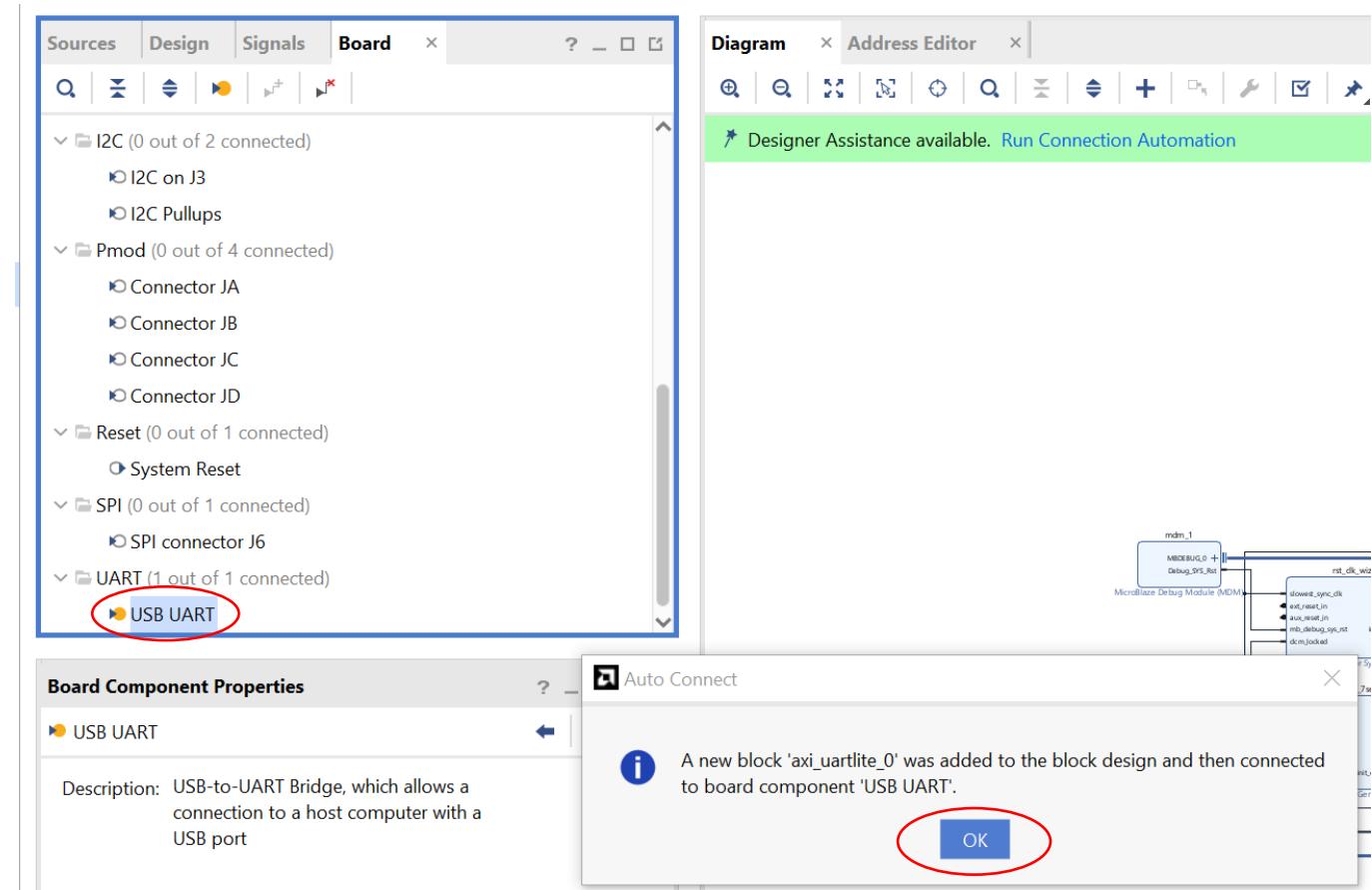
Lab: Embedded Linux

1. Connect the clk wizard locked signal to the sys_rst on the MIG7



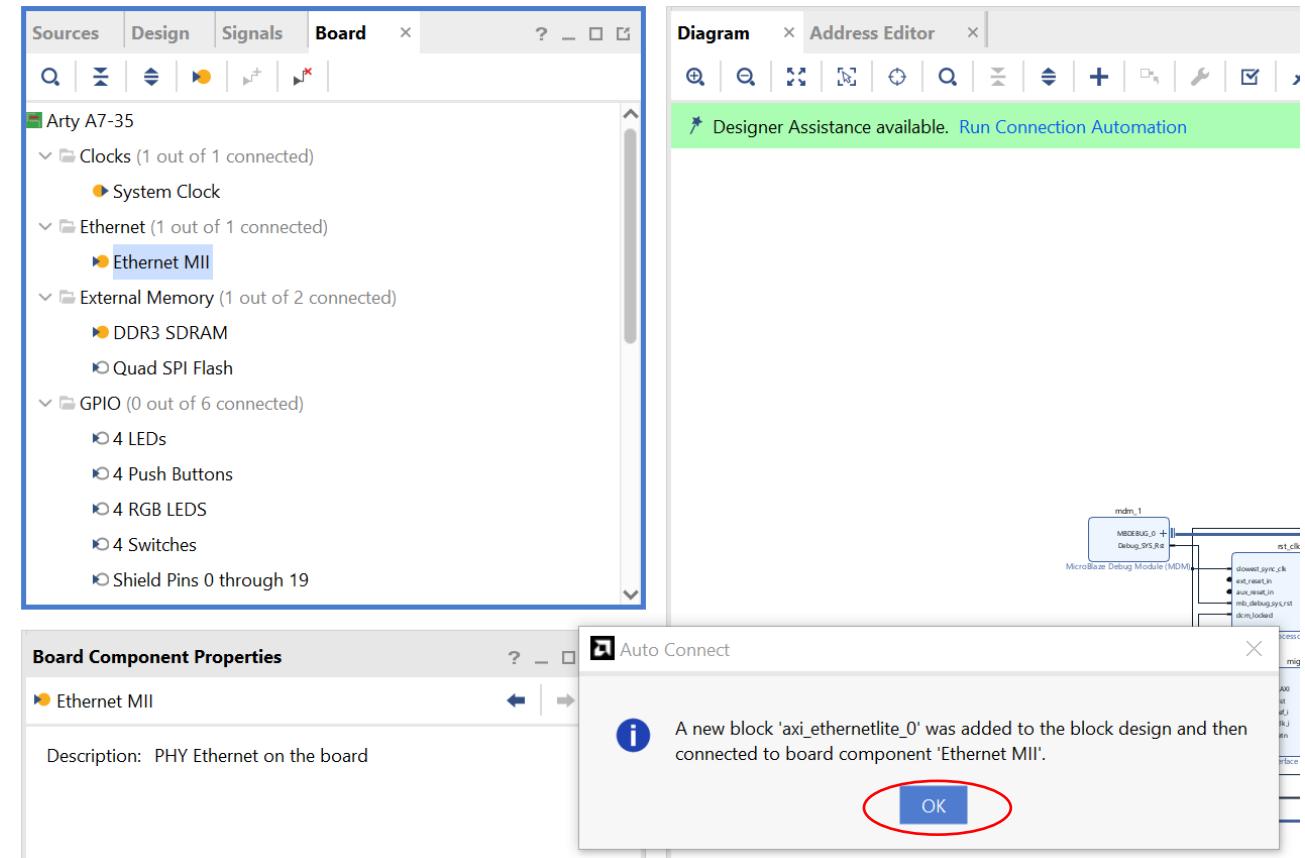
Lab: Embedded Linux

1. From the boards tab add in the USB Uart to the diagram
2. Click OK



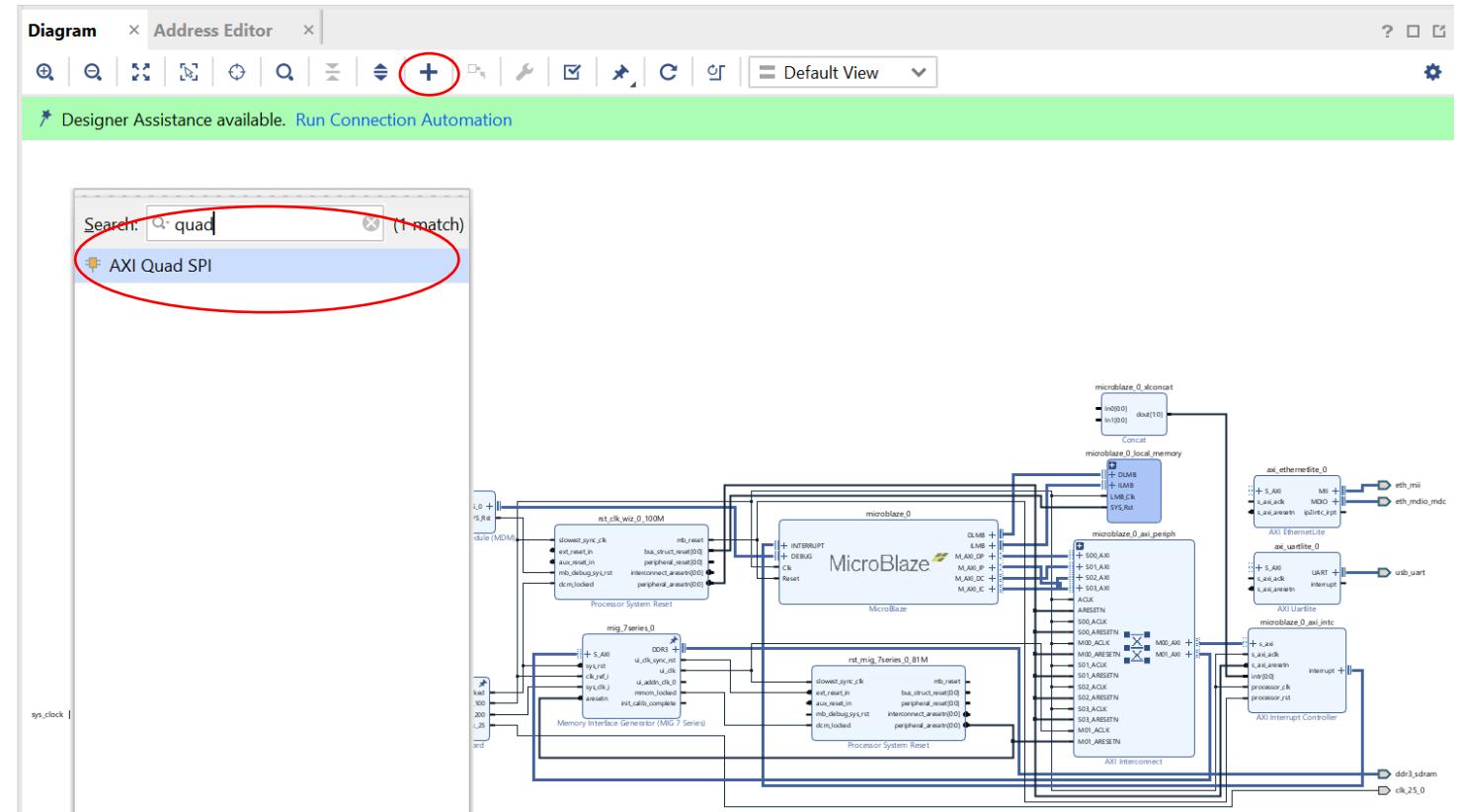
Lab: Embedded Linux

1. From the boards tab drag over the Ethernet MII onto the diagram
2. Click OK



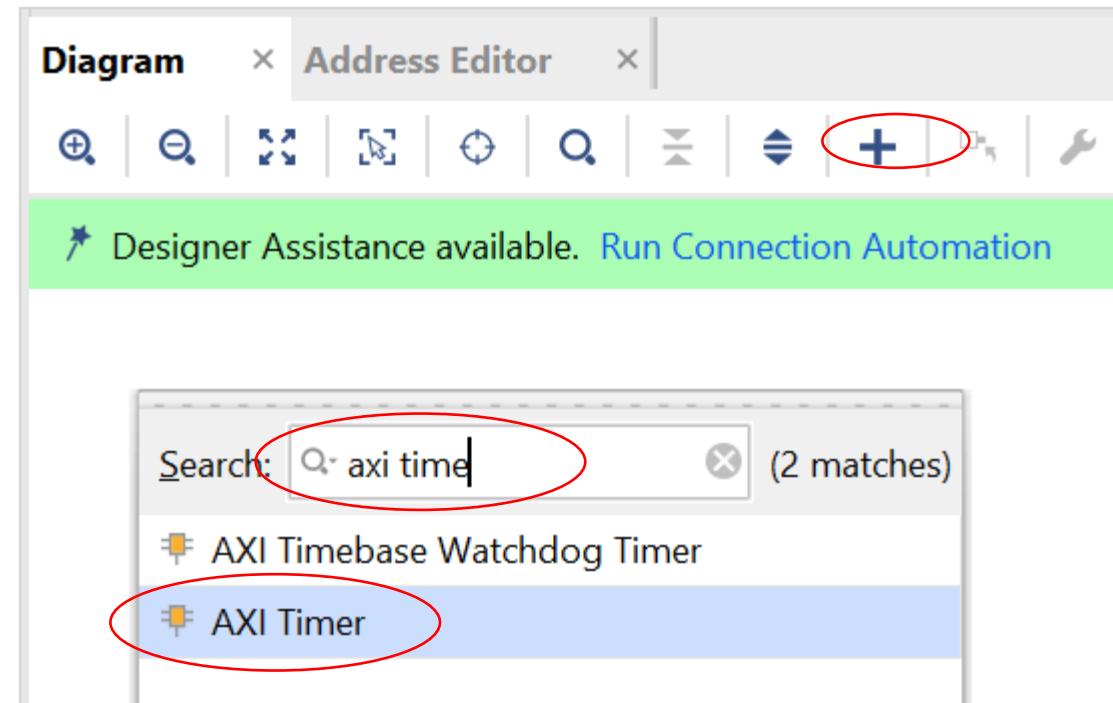
Lab: Embedded Linux

1. Click on the + and in the search box type
2. Quad
3. Select the AXI Quad SPI
4. Double Click on it to add into the diagram



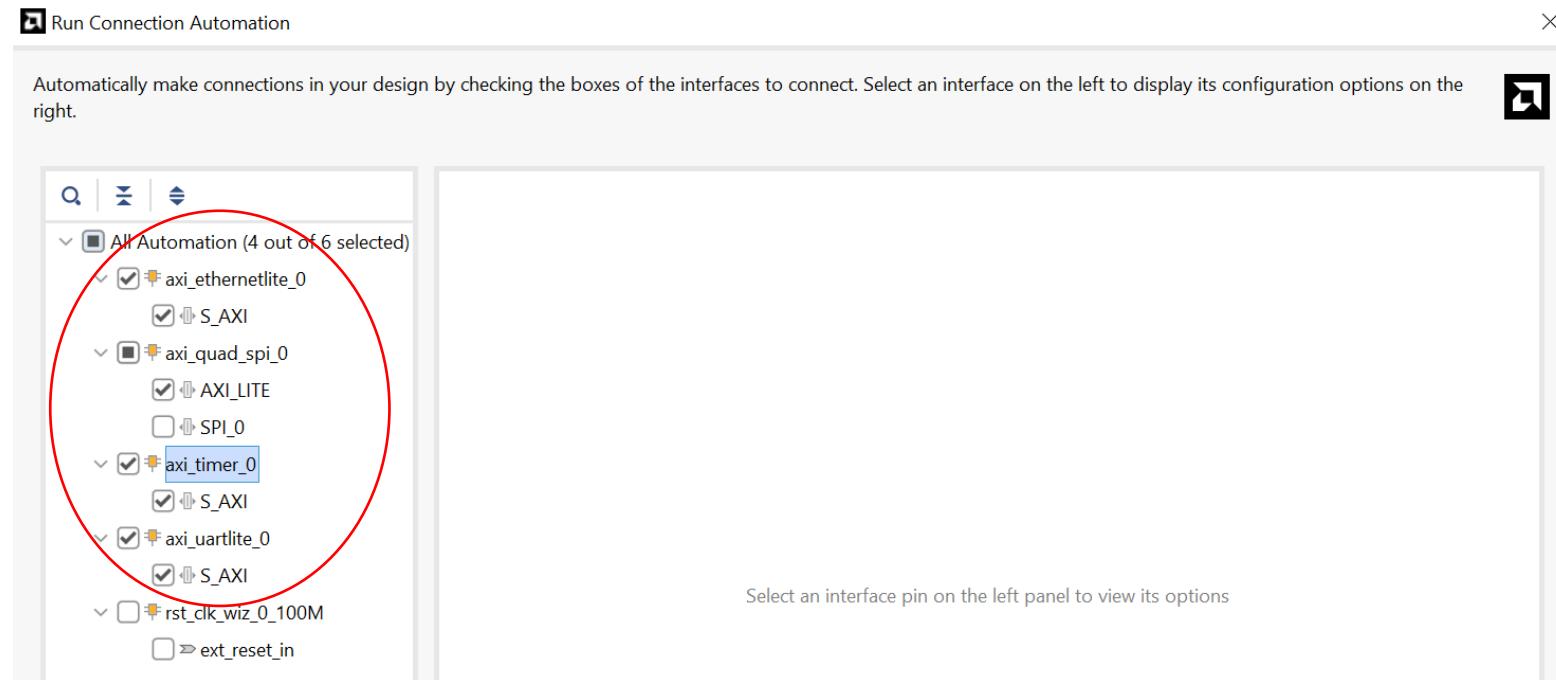
Lab: Embedded Linux

1. Click on the + symbol enter AXI Timer in the search bar
2. Double click on the AXI Timer to add into the diagram



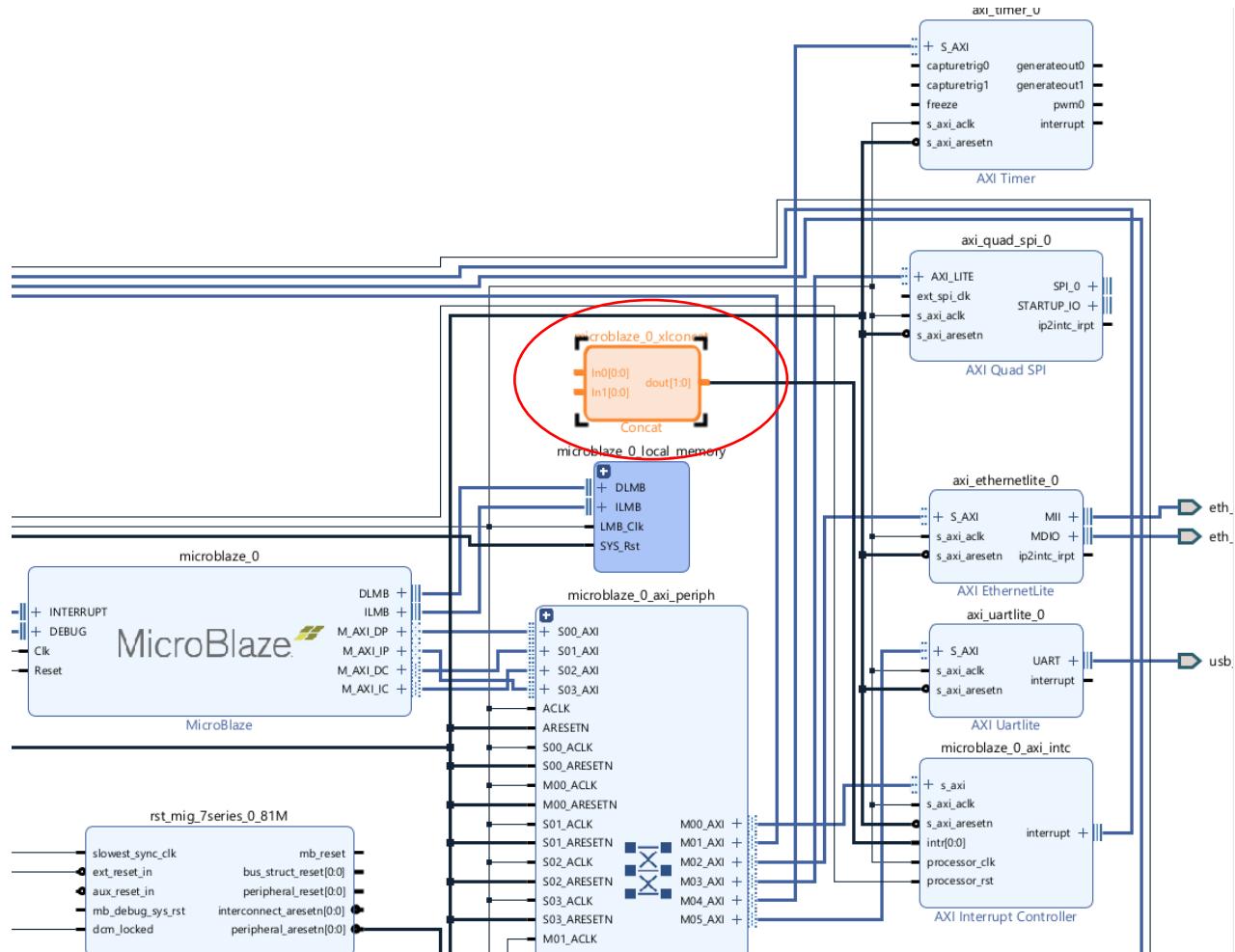
Lab: Embedded Linux

1. Run the block automation
2. Select the AXI Interfaces
3. Click OK



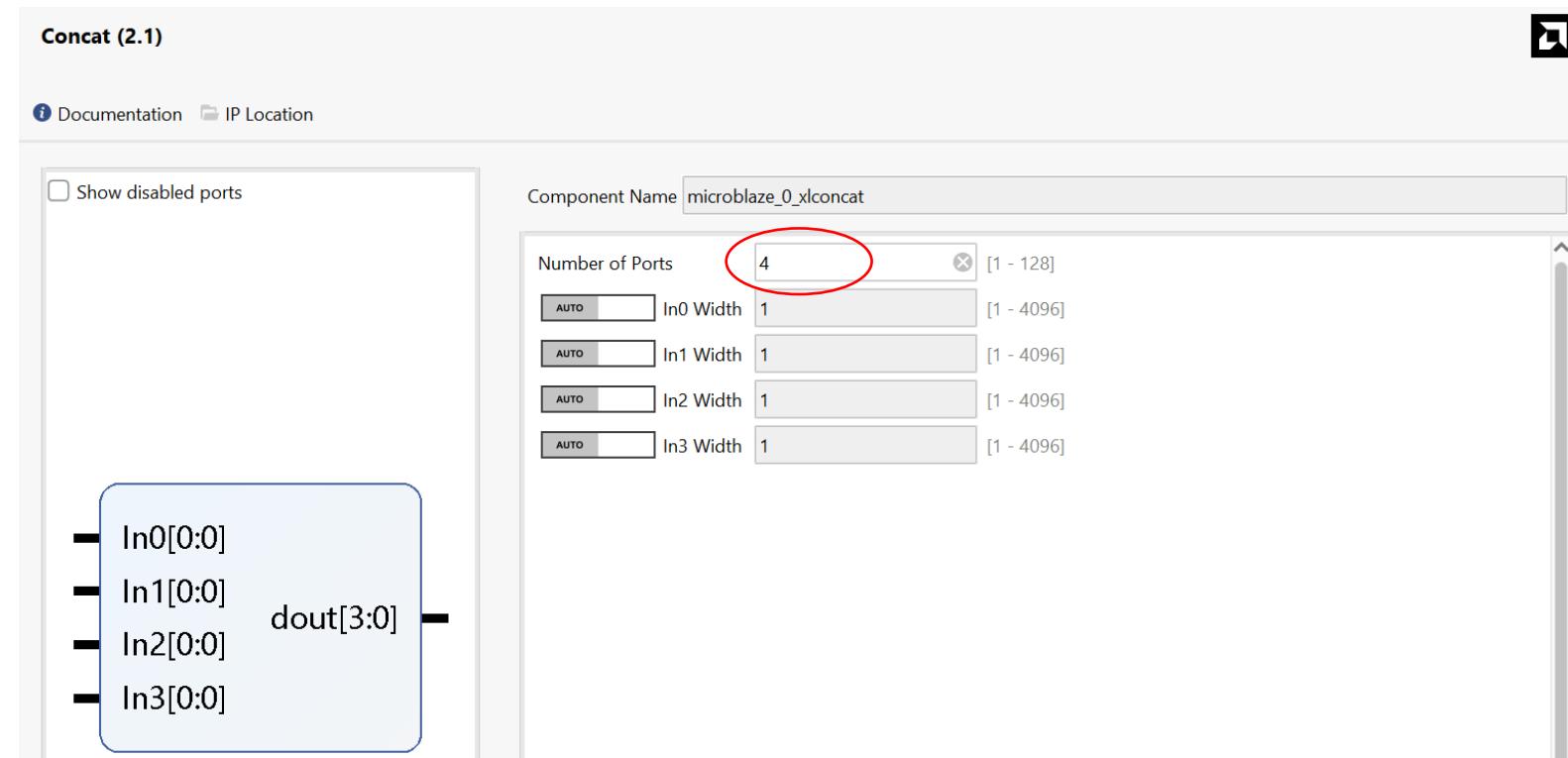
Lab: Embedded Linux

1. Double click on the Concatenate block



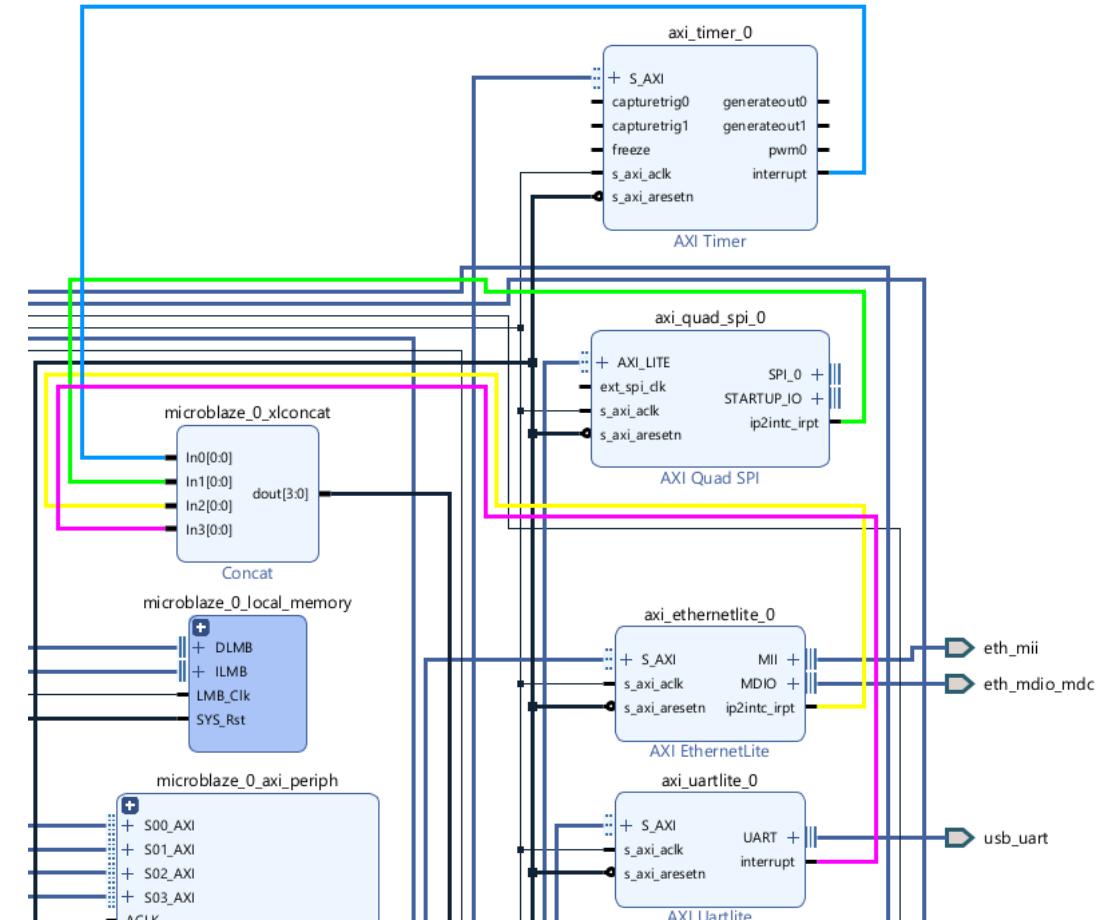
Lab: Embedded Linux

1. Change the number of ports to four



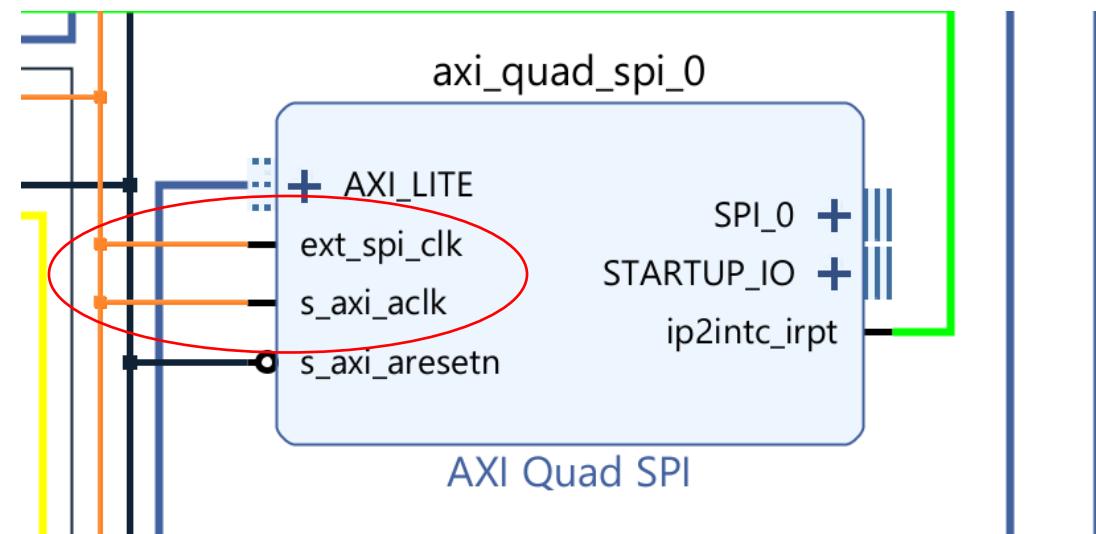
Lab: Embedded Linux

1. From the AXI Timer, AXI Quad SPI, AXI Ethernet and AXI UARTLite connect the interrupts to the concatenate block



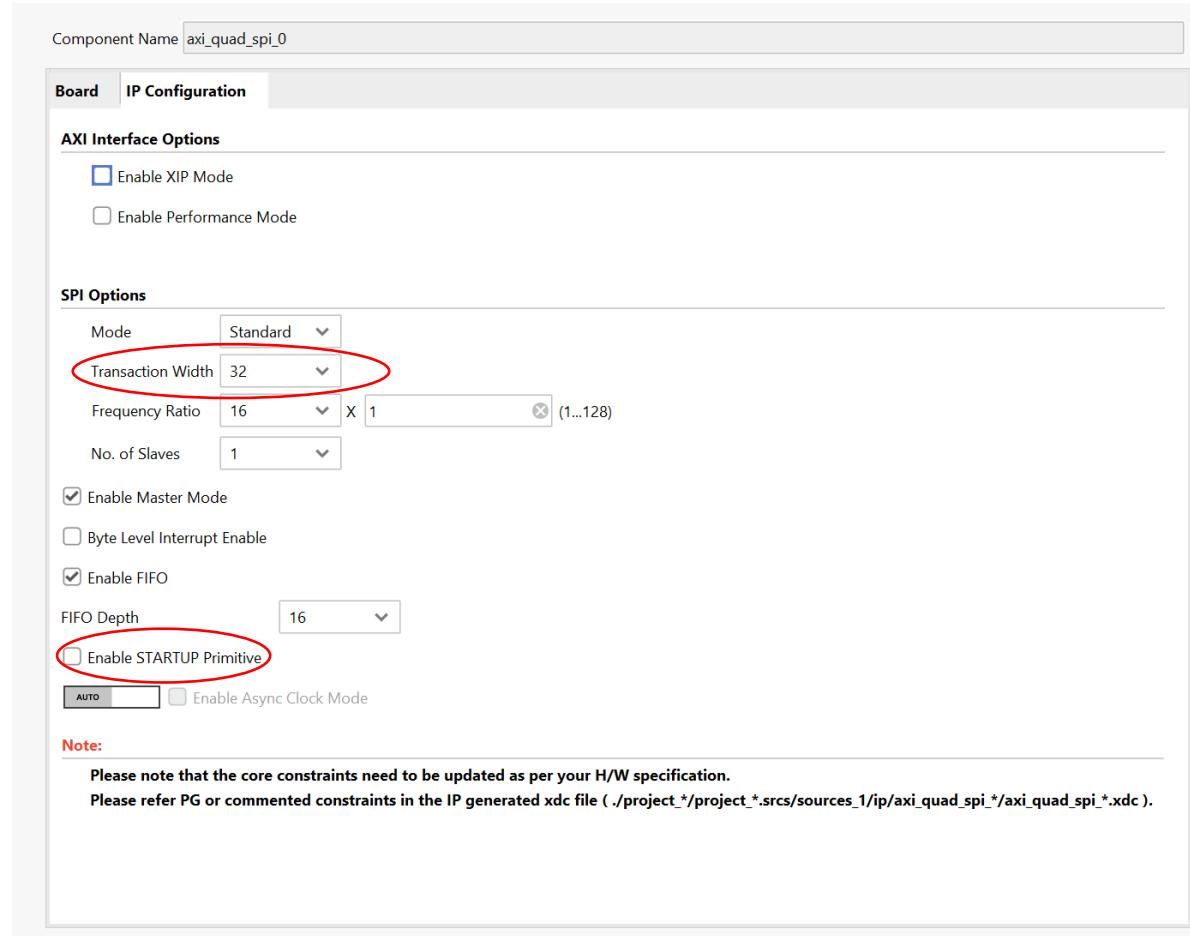
Lab: Embedded Linux

1. On the Quad SPI ensure the ext_spi_clk is connected to the s_axi_clk



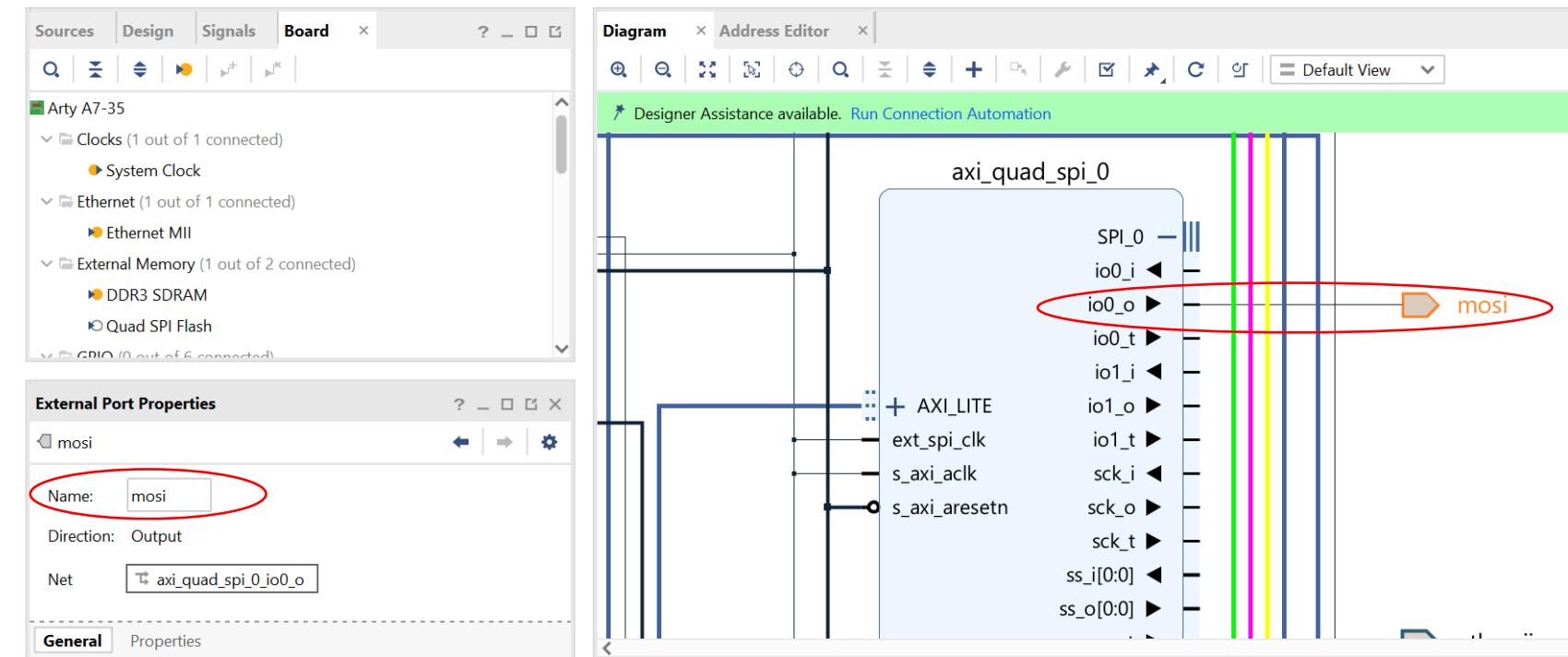
Lab: Embedded Linux

1. Re-customize the AXI Quad SPI to disable the Start up Primitive
2. Set the Transaction width to 32 bits



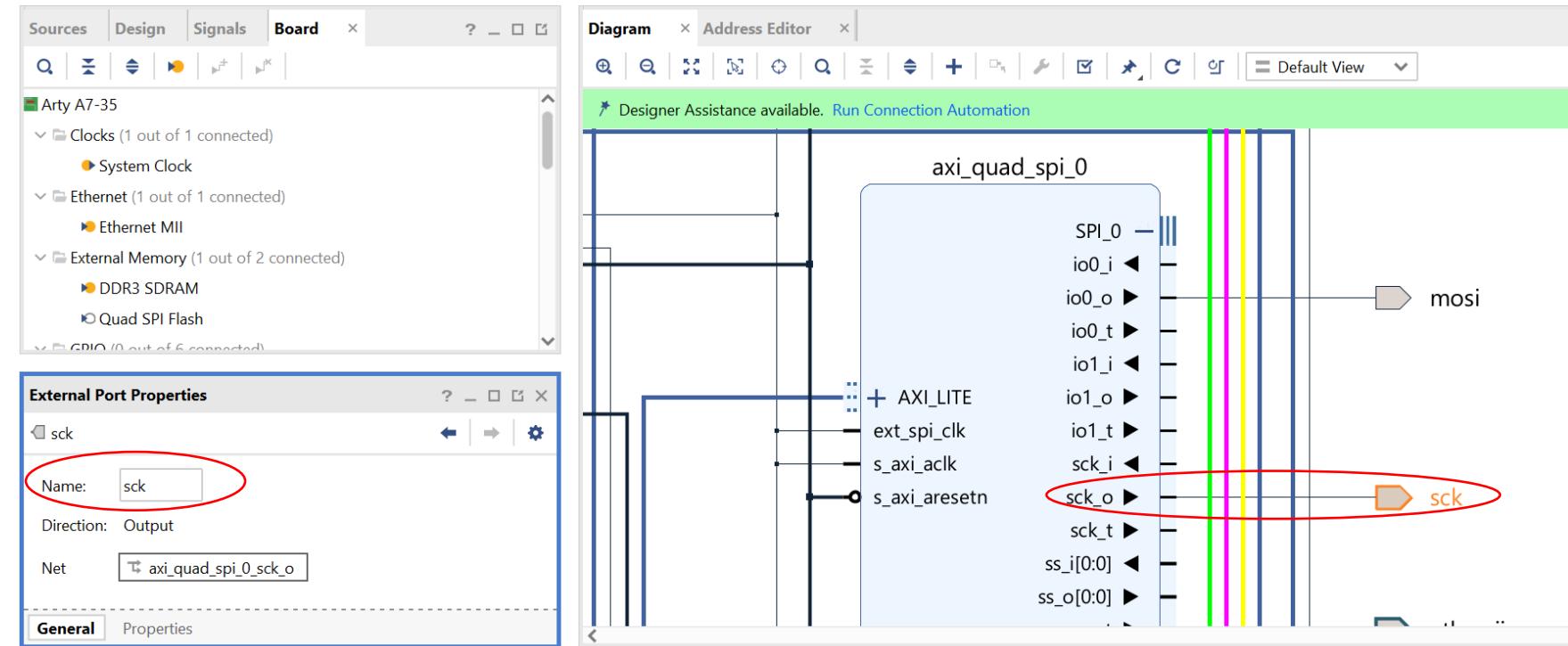
Lab: Embedded Linux

1. Expand the SPI_0 port
2. On io0_o make it external
3. Name it MOSI



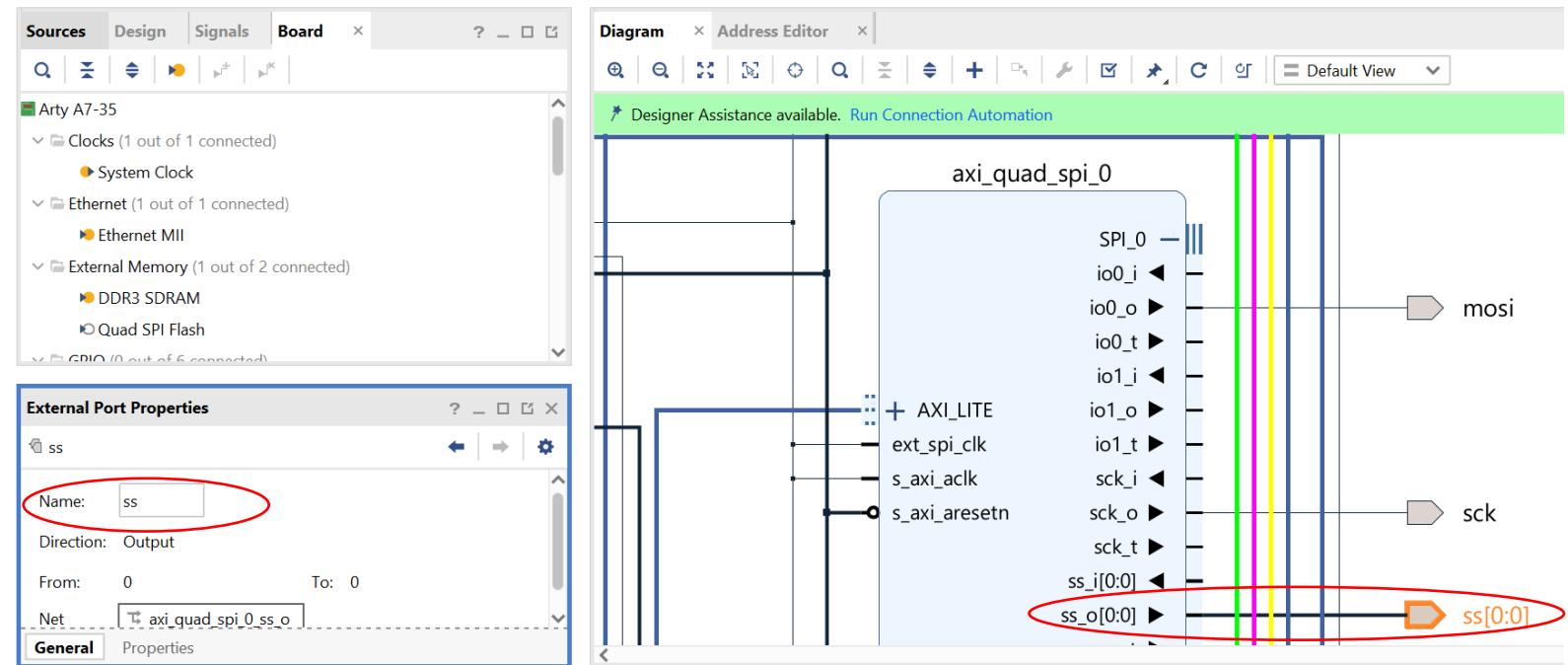
Lab: Embedded Linux

1. Make SCLK_0 external
2. Rename it sck



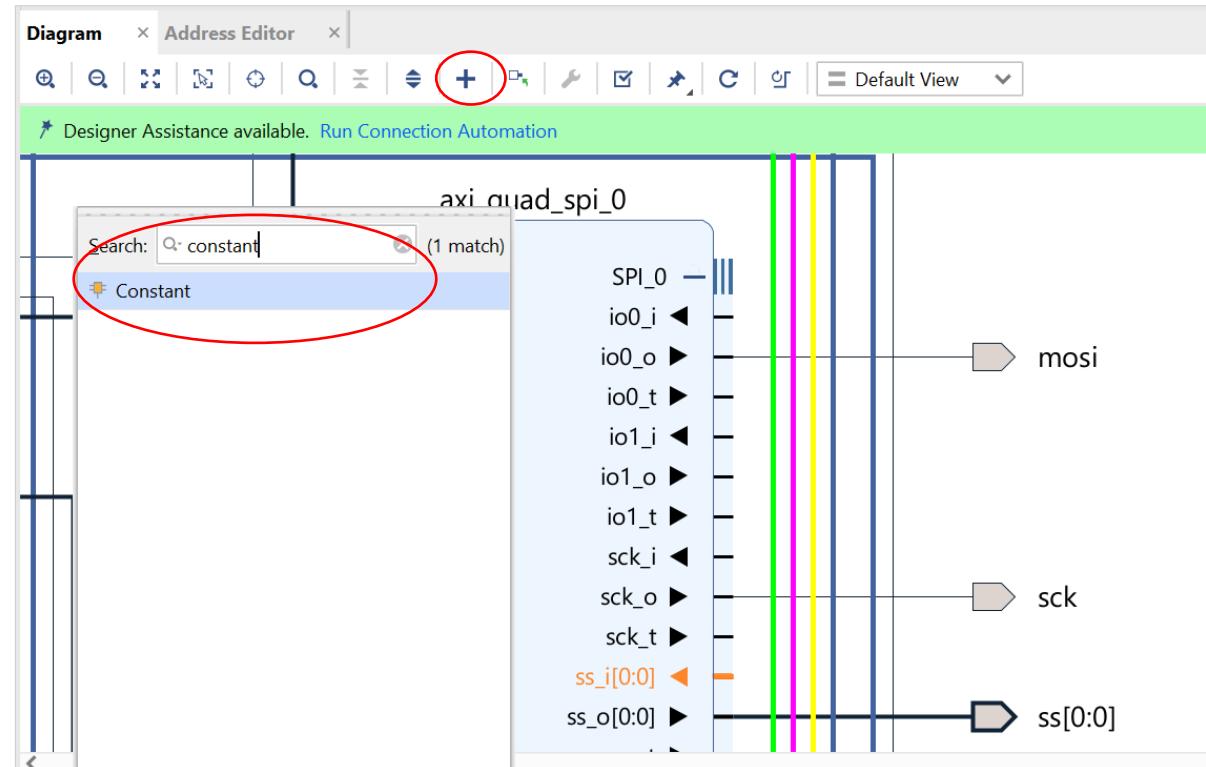
Lab: Embedded Linux

1. Do the same for SS_0
2. Rename it ss



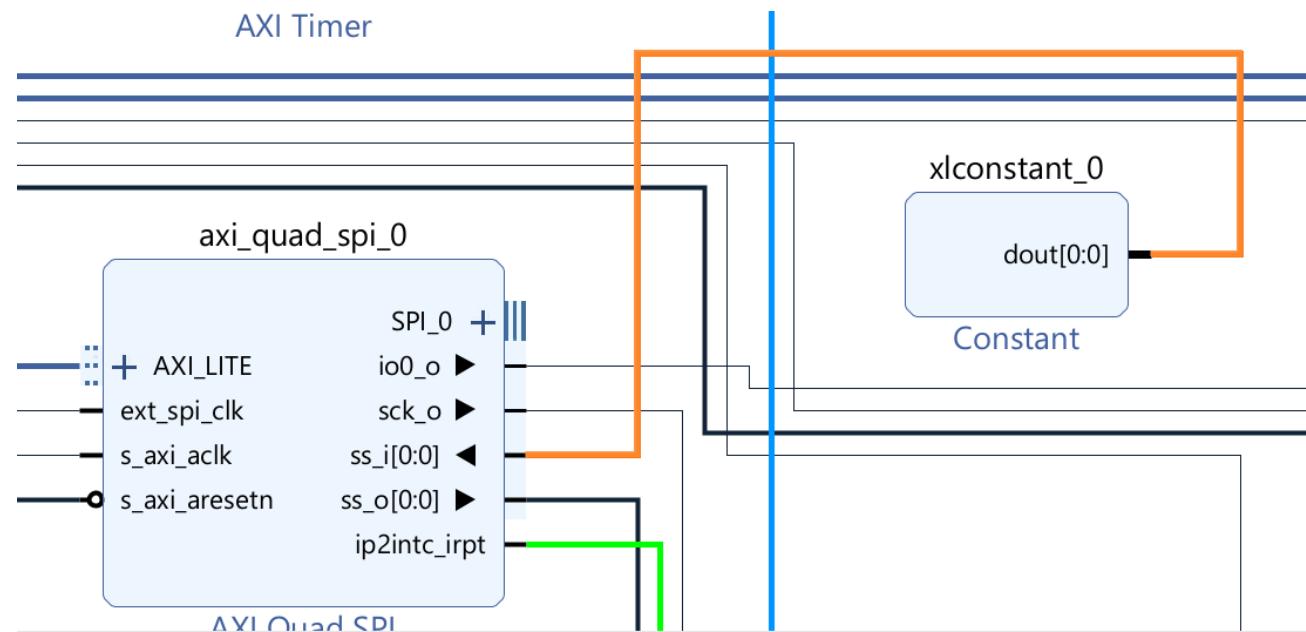
Lab: Embedded Linux

1. Click + and type in constant in the search box
2. Double click on constant and add it into the design
3. Connect it to signal SS_i



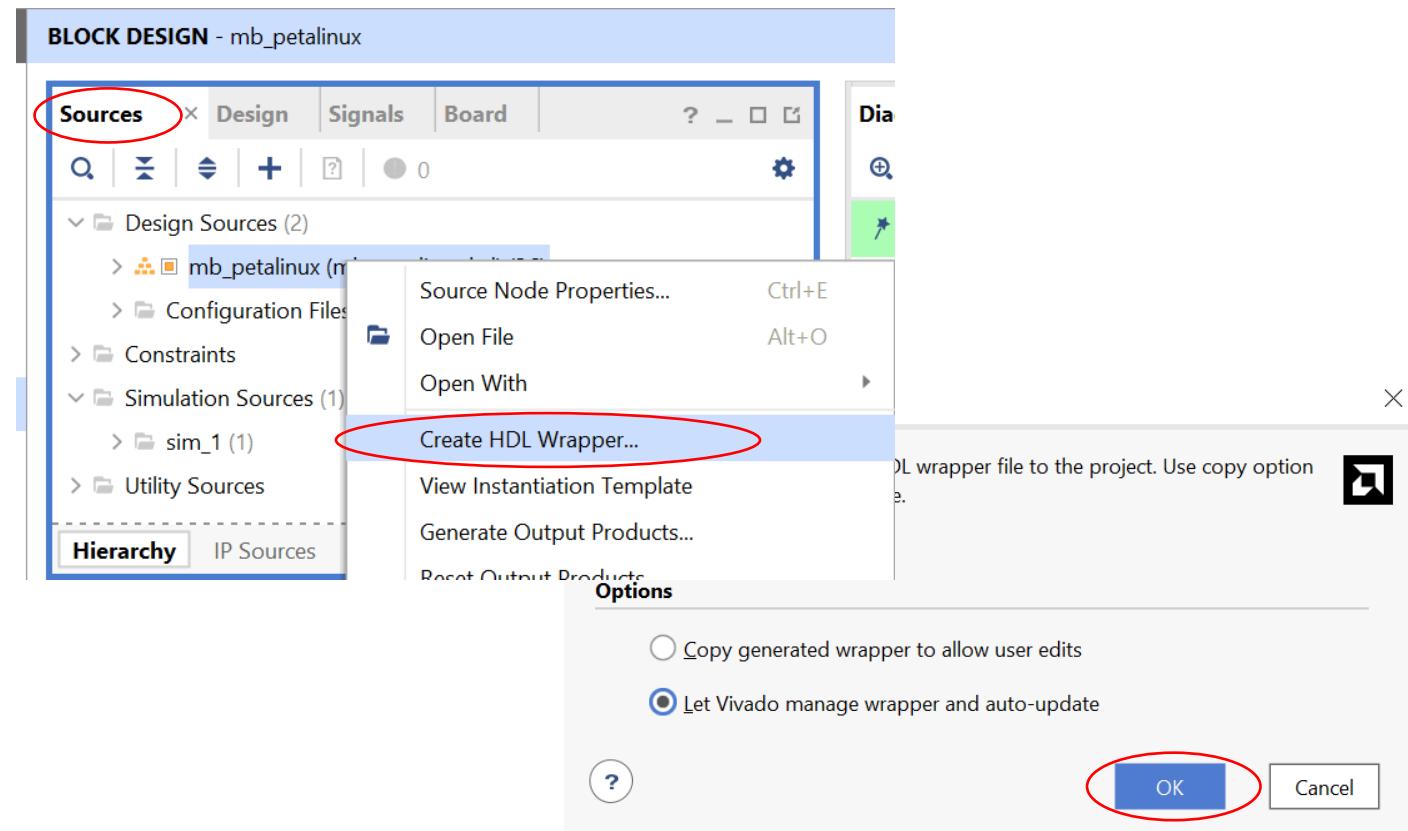
Lab: Embedded Linux

1. The design should look as shown



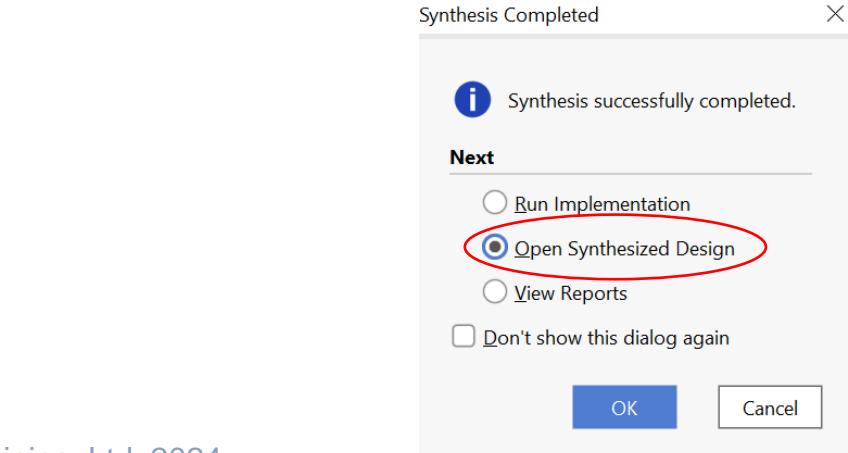
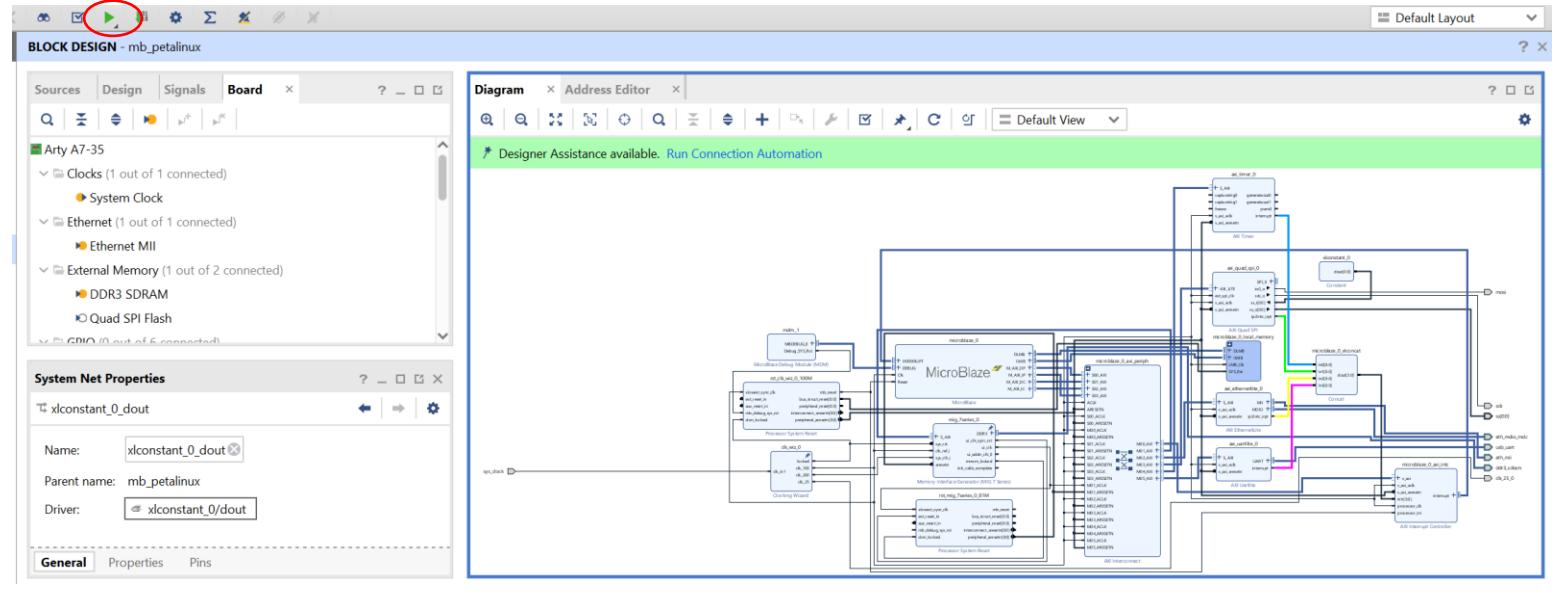
Lab: Embedded Linux

1. Select Sources, right click on the block diagram.
2. Select Create HDL Wrapper
3. When prompted allow the AMD Vivado™ software to manage the file



Lab: Embedded Linux

1. Run the Synthesis
2. Once the synthesis is completed, open the synthesis view



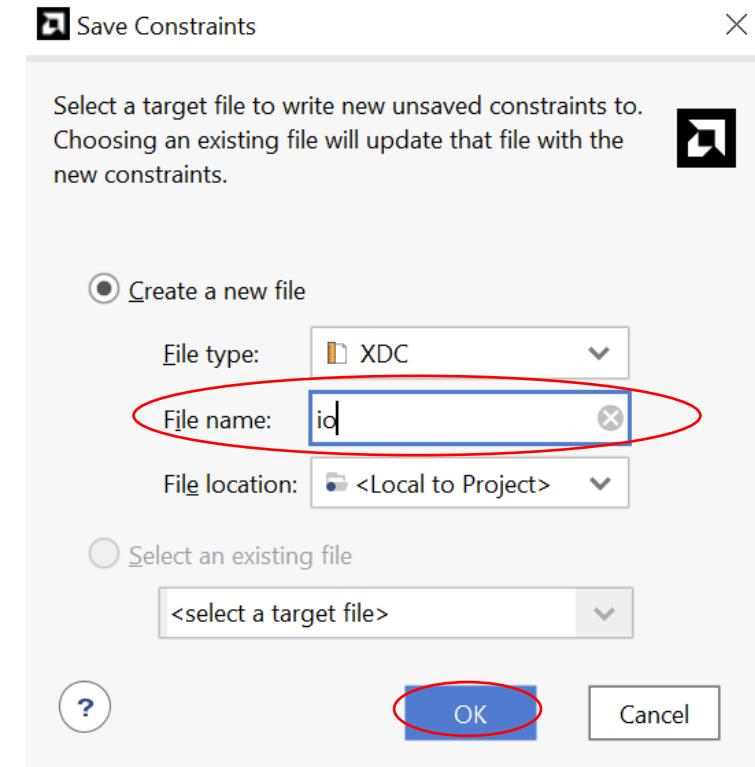
Lab: Embedded Linux

1. Set the IO pin allocation and IO standard as shown opposite

Name	Direction	Board Part Pin	Board Part Interface	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std
▼ All ports (73)								
CLK.CLK_25_0_12088 (1)	OUT					<input checked="" type="checkbox"/>	15	LVC MOS33*
▼ Scalar ports (1)								
clk 25 0	OUT				G18	<input checked="" type="checkbox"/>	15	LVC MOS33*
CLK.SYS_CLOCK_12088 (1)	IN					<input checked="" type="checkbox"/>	35	LVC MOS33*
ddr3_sdram_12088 (48)	(Multiple)					<input checked="" type="checkbox"/>	34	(Multiple)*
eth_mdio_mdc_12088 (2)	(Multiple)					<input checked="" type="checkbox"/>	15	LVC MOS33*
eth_mii_12088 (16)	(Multiple)					<input checked="" type="checkbox"/>	15	LVC MOS33*
▼ SPI_0_45829 (3)								
ss (1)	OUT					<input checked="" type="checkbox"/>	15	LVC MOS33*
ss[0]	OUT				G13	<input checked="" type="checkbox"/>	15	LVC MOS33*
▼ Scalar ports (2)								
mosi	OUT				B11	<input checked="" type="checkbox"/>	15	LVC MOS33*
sck	OUT				D12	<input checked="" type="checkbox"/>	15	LVC MOS33*
usb_uart_12088 (2)	(Multiple)					<input checked="" type="checkbox"/>	16	LVC MOS33*

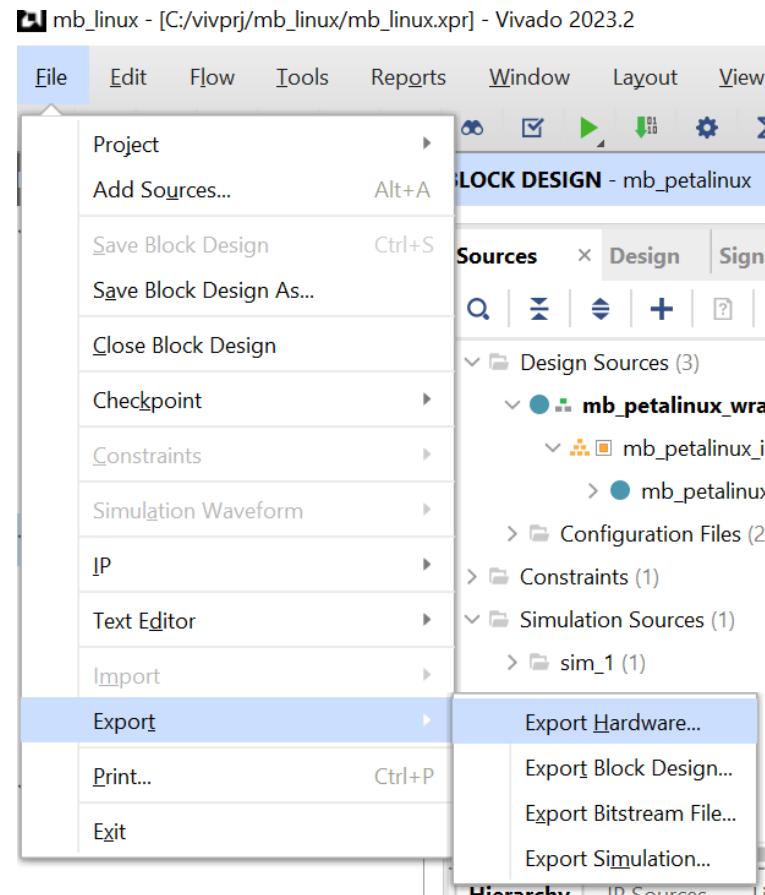
Lab: Embedded Linux

1. Click save and in the prompt add in the file name IO and save it.



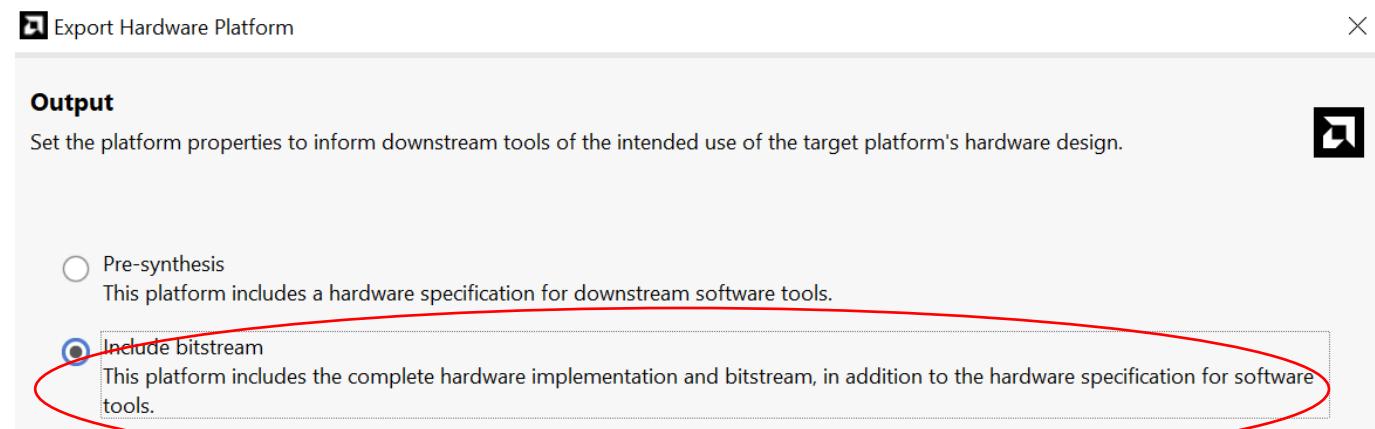
Lab: Embedded Linux

1. Once the bitstream is generated. Export the XSA.



Lab: Embedded Linux

1. Check the option to include the bitstream



Lab: Embedded Linux

1. Open a terminal in the folder you want to create the project on the Linux machine
2. Source the settings script to set up the PetaLinux 2023.2 settings

```
adiuvo@atlas:~/petalinux_projects/microblaze_petalinux$ source ~/petalinux_2023.2/settings.sh
PetaLinux environment set to '/home/adiuvo/petalinux_2023.2'
WARNING: /bin/sh is not bash!
bash is PetaLinux recommended shell. Please set your default shell to bash.
INFO: Checking free disk space
INFO: Checking installed tools
INFO: Checking installed development libraries
INFO: Checking network and other services
WARNING: No tftp server found - please refer to "UG1144 2023.2 PetaLinux Tools Documentation Reference Guide" for its impact and solution
adiuvo@atlas:~/petalinux_projects/microblaze_petalinux$
```

Lab: Embedded Linux

1. Create a new project targeting the AMD MicroBlaze™ processor in PetaLinux

```
adiuvo@atlas:~/petalinux_projects/microblaze_petalinux$ petalinux-create --type project --template microblaze --name mb_linux
INFO: Create project: mb_linux
INFO: New project successfully created in /home/adiuvo/petalinux_projects/microblaze_petalinux(mb_linux
adiuvo@atlas:~/petalinux_projects/microblaze_petalinux$ █
```

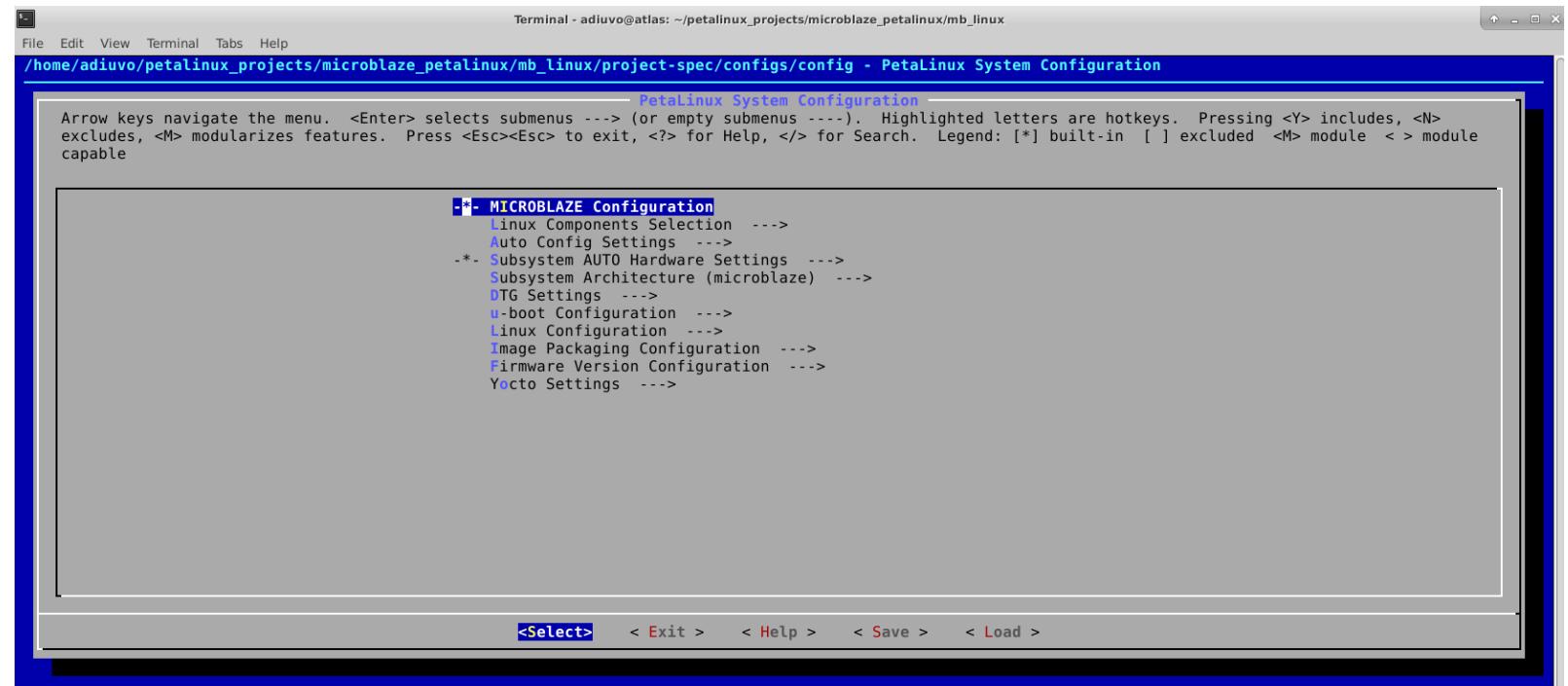
Lab: Embedded Linux

1. Load in the configuration from the XSA which has just been created in the AMD Vivado™ software

```
adiuvo@atlas:~/petalinux_projects/microblaze_petalinux(mb_linux$ petalinux-config --get-hw-description=.../  
[INFO] Sourcing buildtools  
[INFO] Getting hardware description...  
INFO: Renaming mb_petalinux_wrapper.xsa to system.xsa  
[INFO] Extracting yocto SDK to components/yocto. This may take time!  
[INFO] Generating Kconfig for project  
[INFO] Menuconfig project  
[INFO] Generating kconfig for rootfs  
[INFO] Silentconfig rootfs  
[INFO] Adding user layers  
[INFO] Generating machine conf file  
[INFO] Generating plnxtool conf file  
[INFO] Generating kernel configuration files  
[INFO] Generating u-boot configuration files  
[INFO] Generating workspace directory  
[INFO] Successfully configured project
```

Lab: Embedded Linux

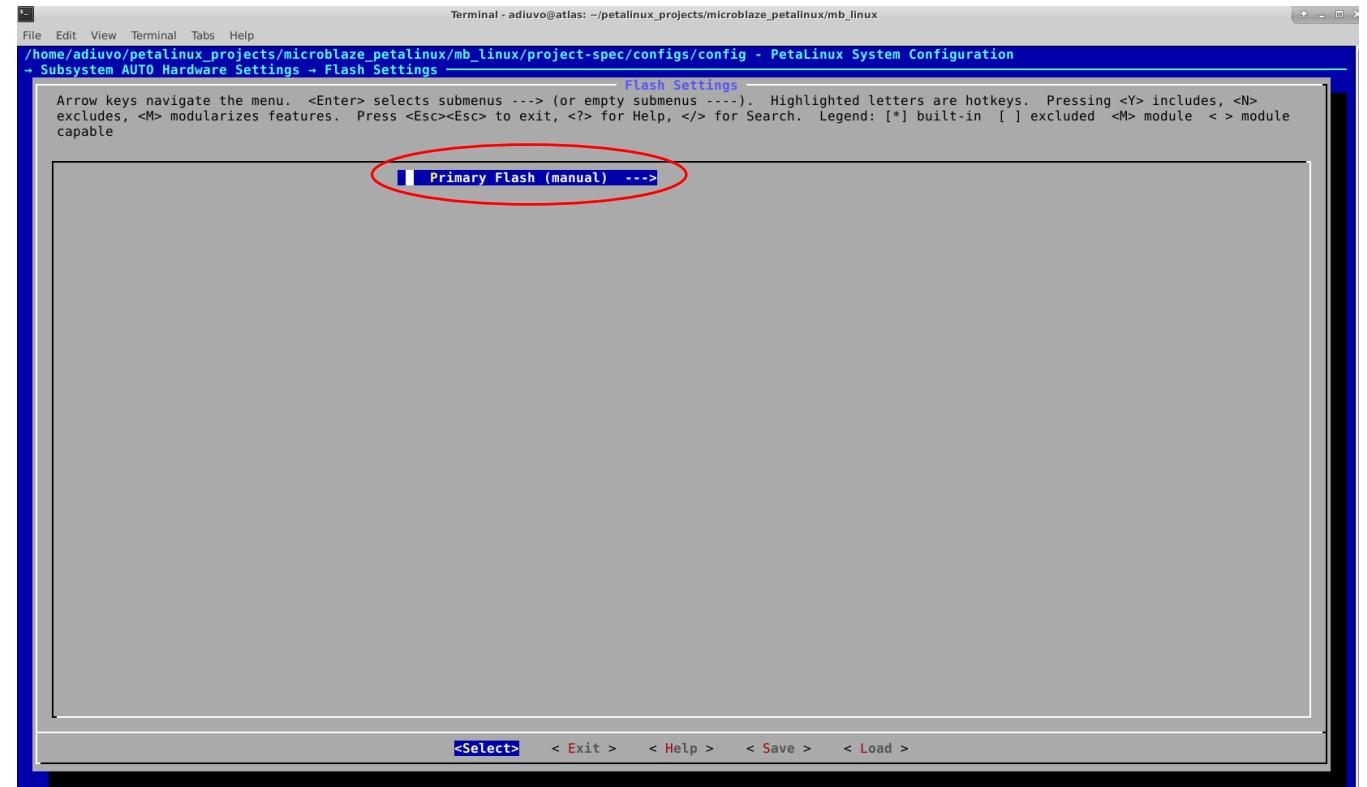
1. This will open the project settings



Lab: Embedded Linux

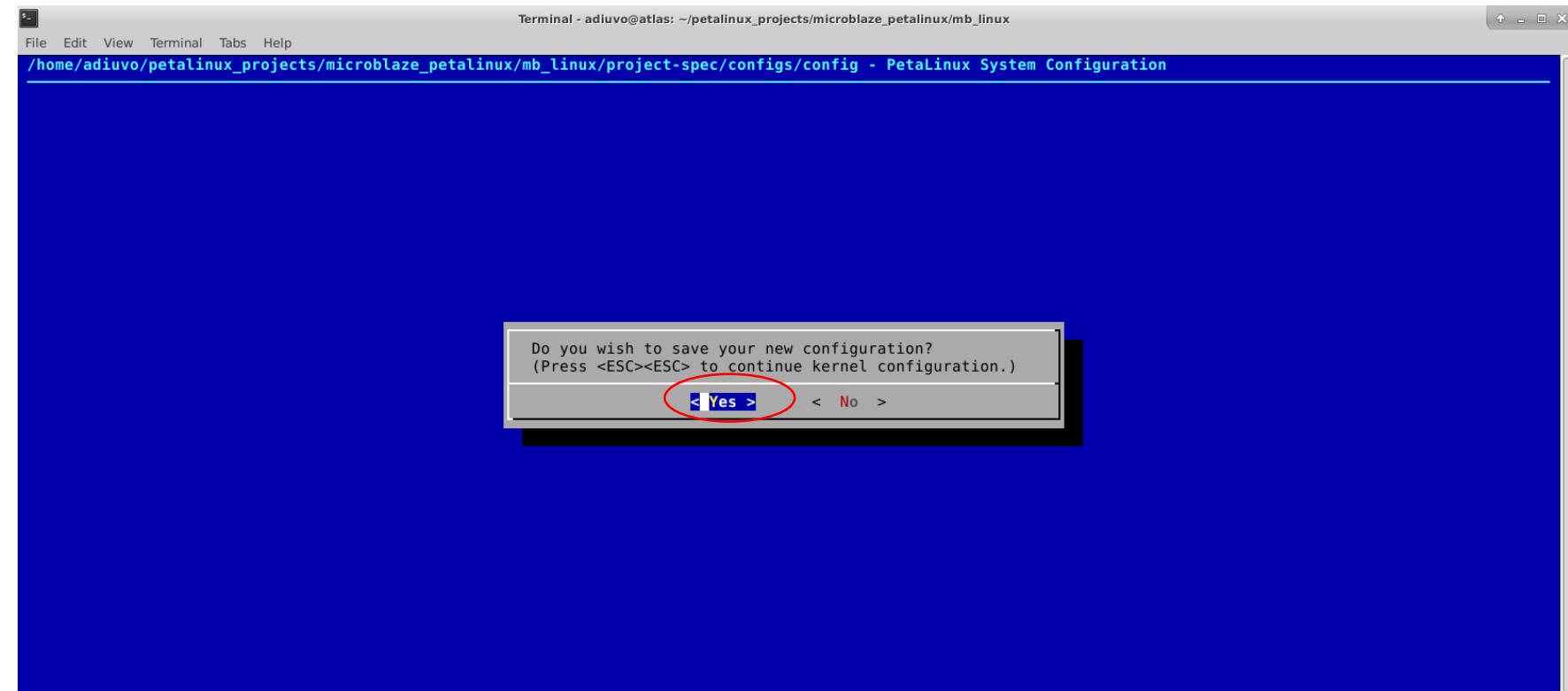
1. Under Subsystem Auto Hardware Settings, change the primary flash settings to manual

2. This stops it trying to use the AXI QSPI flash for boot



Lab: Embedded Linux

1. Navigate to the exit and save the changes



Lab: Embedded Linux

1. Enter the command below to customize the kernel (this might take several minutes to complete)

```
adiuvo@atlas:~/petalinux_projects/microblaze_petalinux(mb_linux)$ petalinux-config -c kernel
```

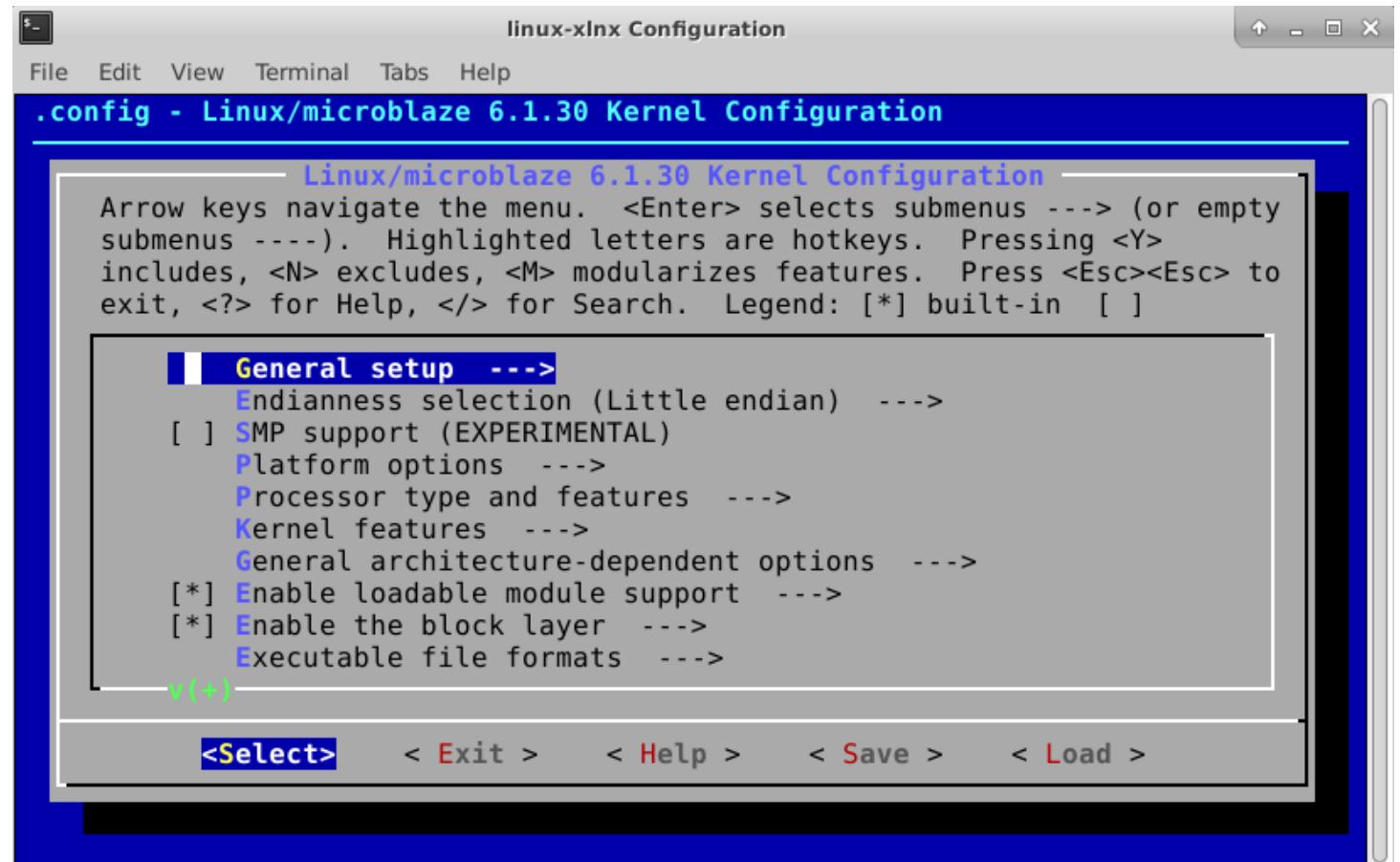
Lab: Embedded Linux

1. You will see lots of information downloaded

```
adiuvo@atlas:~/petalinux_projects/microblaze_petalinux(mb_linux$ petalinux-config -c kernel
[INFO] Sourcing buildtools
[INFO] Silentconfig project
[INFO] Silentconfig rootfs
[INFO] Generating machine conf file
[INFO] Generating plnxtool conf file
[INFO] Generating kernel configuration files
[INFO] Generating u-boot configuration files
[INFO] Generating workspace directory
[INFO] Configuring: kernel
[INFO] bitbake virtual/kernel -c cleansstate
NOTE: Started PRServer with DBfile: /home/adiuvo/petalinux_projects/microblaze_petalinux(mb_linux/build/cache/prserv.sqlite3, Address: 127.0.0.1:38731, PID: 20686
Loading cache: 100% | ETA: ---:--
Loaded 0 entries from dependency cache.
Parsing recipes: 100% |#####| Time: 0:00:35
Parsing of 4400 .bb files complete (0 cached, 4400 parsed). 6339 targets, 892 skipped, 1 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies
NOTE: Fetching uninative binary shim file:///home/adiuvo/petalinux_projects/microblaze_petalinux(mb_linux/components/yocto/downloads/uninative/3dd82c3fbdb59e87bf091c
3eef55a05fae528eeda3083828f76cd4deacea8b/x86_64-nativesdk-libc-3.9.tar.xz;sha256sum=3dd82c3fbdb59e87bf091c3eef55a05fae528eeda3083828f76cd4deacea8b (will check PR
EMIRROS first)
Initialising tasks: 100% |#####| Time: 0:00:00
Sstate summary: Wanted 0 Local 0 Mirrors 0 Missed 0 Current 0 (0% match, 0% complete)
NOTE: No setscene tasks
NOTE: Executing Tasks
NOTE: Tasks Summary: Attempted 3 tasks of which 0 didn't need to be rerun and all succeeded.
[INFO] bitbake virtual/kernel -c menuconfig
NOTE: Started PRServer with DBfile: /home/adiuvo/petalinux_projects/microblaze_petalinux(mb_linux/build/cache/prserv.sqlite3, Address: 127.0.0.1:45095, PID: 21951
Loading cache: 100% |#####| Time: 0:00:01
Loaded 6338 entries from dependency cache.
Parsing recipes: 100% |#####| Time: 0:00:02
Parsing of 4400 .bb files complete (4399 cached, 1 parsed). 6339 targets, 892 skipped, 1 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies
Initialising tasks: 100% |#####| Time: 0:00:01
Checking sstate mirror object availability: 100% |#####| Time: 0:00:02
Sstate summary: Wanted 137 Local 0 Mirrors 136 Missed 1 Current 0 (99% match, 0% complete)
NOTE: Executing Tasks
Setscene tasks: 137 of 137
Currently 1 running tasks (619 of 620) 99% |#####| Time: 0:00:03
0: linux-xlnx-6.1.30-xilinx-v2023.2+gitAUTOINC+a19da02cf5-r0 do_menuconfig - 5m35s (pid 3363)
```

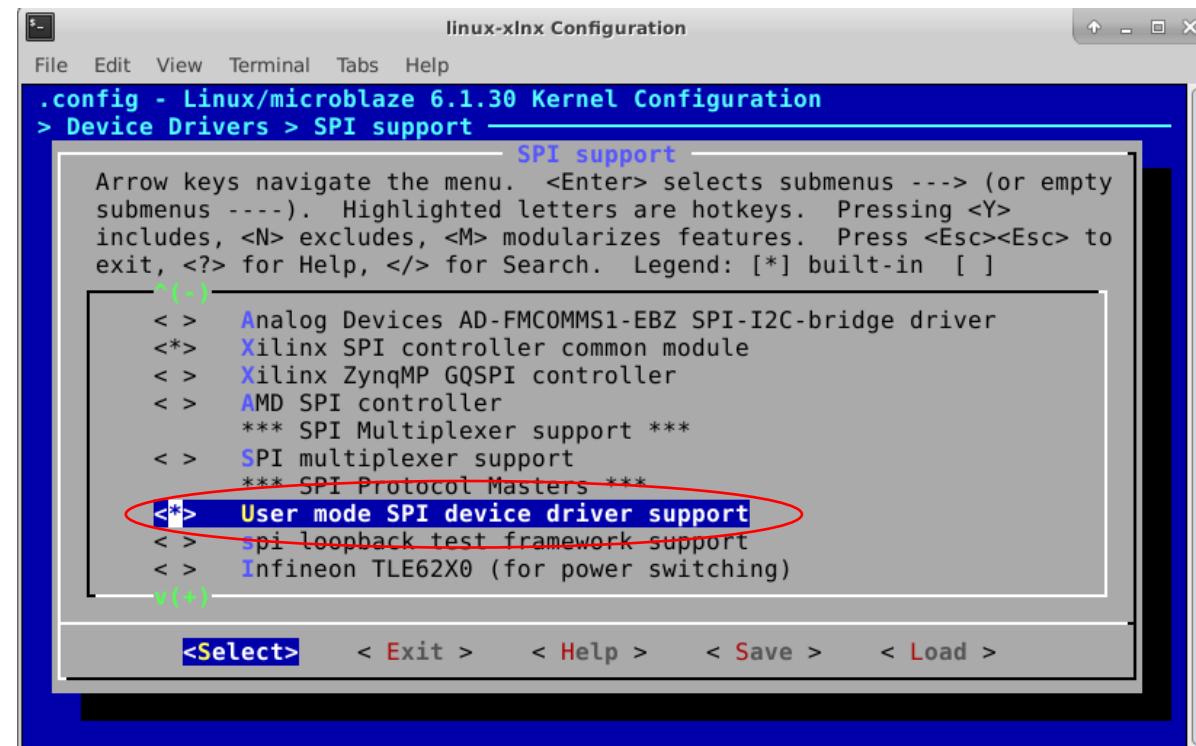
Lab: Embedded Linux

1. The kernel configuration will open



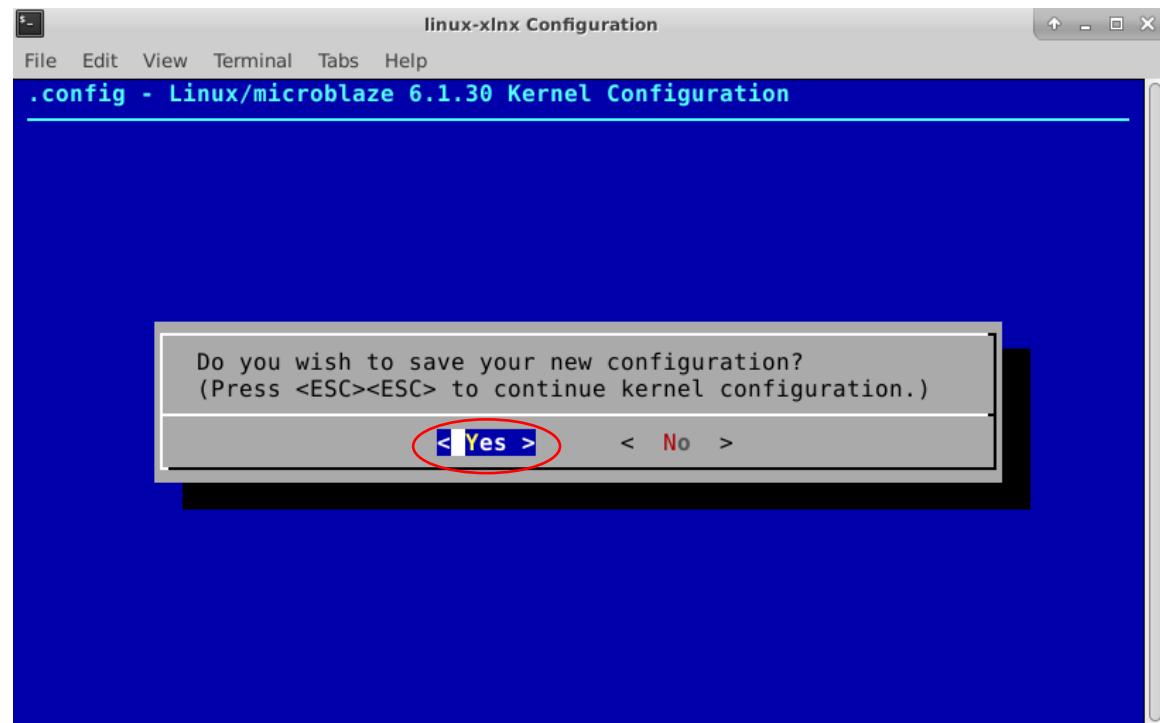
Lab: Embedded Linux

1. Navigate to the Device Drivers SPI Support and enable the user mode SPI device driver support



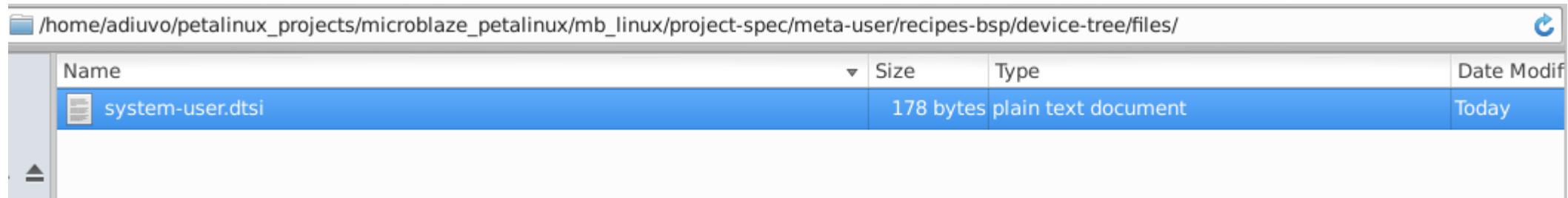
Lab: Embedded Linux

1. Exit and save the configuration



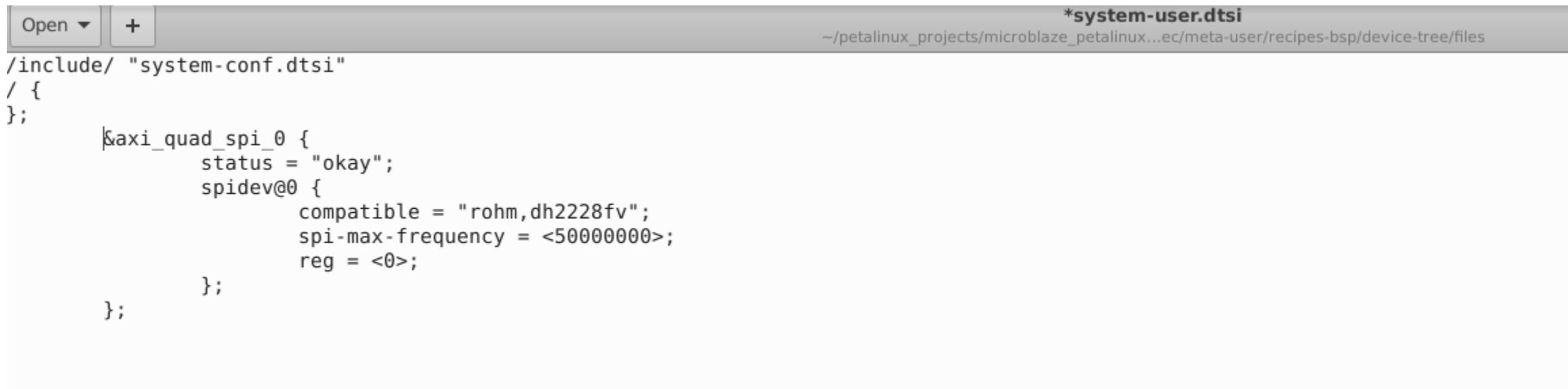
Lab: Embedded Linux

1. Open the system-user.dtsi file under the meta-user layer – only files under here can be manipulated by the user



Lab: Embedded Linux

1. Add in the text below to enable the SPI Dev on device 0, Chip Select 0



```
Open + *system-user.dtsi
~/petalinux_projects/microblaze_petalinux...ec/meta-user/recipes-bsp/device-tree/files

/include/ "system-conf.dtsi"
{
};

&xaxi_quad_spi_0 {
    status = "okay";
    spidev@0 {
        compatible = "rohm,dh2228fv";
        spi-max-frequency = <50000000>;
        reg = <0>;
    };
};
```

Lab: Embedded Linux

1. Build the system

```
adiuvo@atlas:~/petalinux_projects/microblaze_petalinux/mb_linux$ petalinux-build
[INFO] Sourcing buildtools
[INFO] Building project
[INFO] Silentconfig project
[INFO] Silentconfig rootfs
[INFO] Generating workspace directory
INFO: bitbake petalinux-image-minimal
NOTE: Started PRServer with DBfile: /home/adiuvo/petalinux_projects/microblaze_petalinux/mb_linux/build/cache/prserv.sqlite3, Address: 127.0.0.1:38441, PID: 9314
Loading cache: 100% |#####| Time: 0:00:01
Loaded 6338 entries from dependency cache.
Parsing recipes: 100% |#####| Time: 0:00:02
Parsing of 4400 .bb files complete (4399 cached, 1 parsed). 6339 targets, 892 skipped, 1 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies
Initialising tasks: 100% |#####| Time: 0:00:02
Checking sstate mirror object availability: 100% |#####| Time: 0:00:21
Sstate summary: Wanted 1127 Local 0 Mirrors 997 Missed 130 Current 54 (88% match, 88% complete)
NOTE: Executing Tasks
WARNING: fs-boot-git+gitAUTOINC+c9a0ee31b2-r0 do_package_qa: QA Issue: File /boot/fs-boot-AUTOINC+c9a0ee31b2.elf in package fs-boot contains reference to TMPDIR [buildpaths]
NOTE: Tasks Summary: Attempted 3246 tasks of which 2710 didn't need to be rerun and all succeeded.

Summary: There was 1 WARNING message.
INFO: Failed to copy built images to tftp dir: /tftpboot
[INFO] Successfully built project
adiuvo@atlas:~/petalinux_projects/microblaze_petalinux/mb_linux$
```

Lab: Embedded Linux

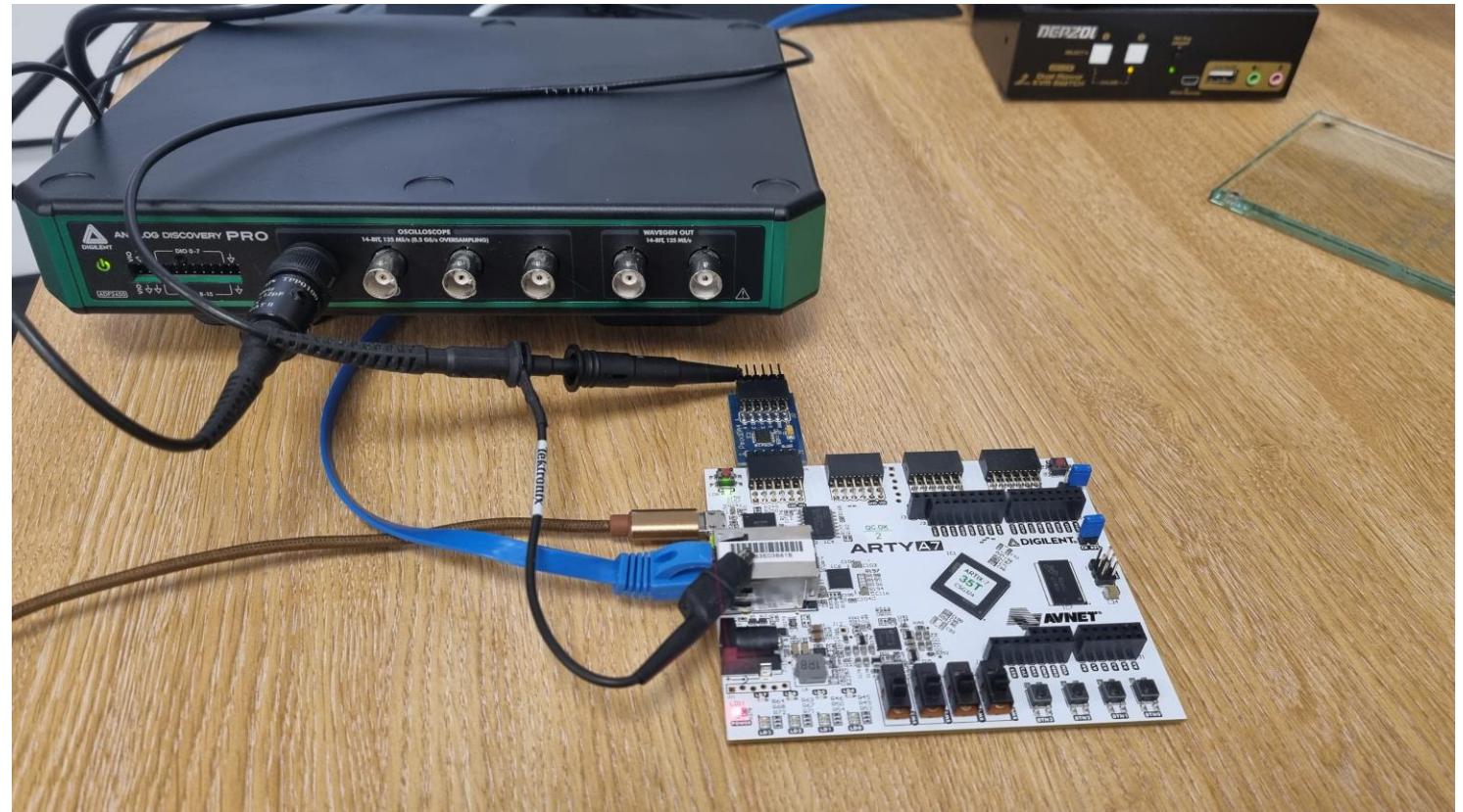
1. Connect the board over the USB Uart and Connect the Ethernet, etc
2. Download the FPGA over JTAG
3. Download the Kernel over JTAG

```
adiuvo@atlas:~/petalinux_projects/microblaze_petalinux(mb_linux$ petalinux-boot --jtag --fpga
[INFO] Sourcing buildtools
INFO: Use bitstream: "/home/adiuvo/petalinux_projects/microblaze_petalinux(mb_linux/images/linux/system.bit".
INFO: Please use --fpga --bitstream <BITSTREAM> to specify a bitstream if you want to use other bitstream.
INFO: Launching XSDB for file download and boot.
INFO: This may take a few minutes, depending on the size of your image.
rlwrap: warning: your $TERM is 'xterm-256color' but rlwrap couldn't find it in the terminfo database. Expect some problems.: Inappropriate ioctl for device
INFO: Configuring the FPGA...
INFO: Downloading bitstream: /home/adiuvo/petalinux_projects/microblaze_petalinux(mb_linux/images/linux/system.bit to the target.

adiuvo@atlas:~/petalinux_projects/microblaze_petalinux(mb_linux$ petalinux-boot --jtag --kernel
[INFO] Sourcing buildtools
INFO: Launching XSDB for file download and boot.
INFO: This may take a few minutes, depending on the size of your image.
rlwrap: warning: your $TERM is 'xterm-256color' but rlwrap couldn't find it in the terminfo database. Expect some problems.: Inappropriate ioctl for device
INFO: Downloading ELF file: /home/adiuvo/petalinux_projects/microblaze_petalinux(mb_linux/images/linux/u-boot.elf to the target.
INFO: Loading image: /home/adiuvo/petalinux_projects/microblaze_petalinux(mb_linux/images/linux/linux.bin.ub at 0x80000000
INFO: Loading image: /home/adiuvo/petalinux_projects/microblaze_petalinux(mb_linux/images/linux/system.dtb at 0x81e00000
INFO: Loading image: /home/adiuvo/petalinux_projects/microblaze_petalinux(mb_linux/images/linux/rootfs.cpio.gz.u-boot at 0x82e00000
INFO: Loading image: /home/adiuvo/petalinux_projects/microblaze_petalinux(mb_linux/images/linux/boot.scr at 0x8f200000
adiuvo@atlas:~/petalinux_projects/microblaze_petalinux(mb_linux$
```

Lab: Embedded Linux

1. The configuration should be similar to as shown



Lab: Embedded Linux

1. In a terminal window set for 9600:N:1 wait for the Linux OS to boot and sign in

2. View the list of devices

```
/init
with environment:
  HOME=/
  TERM=linux
INIT: version 3.04 booting
Starting mdev... OK
Fri Mar  9 12:34:56 UTC 2018
random: crng init done
Configuring packages on first boot....
  (This may take several minutes. Please do not power off the machine.)
Running postinst /etc/rpm-postinsts/100-sysvinit-inittab...
update-rc.d: /etc/init.d/run-postinsts exists during rc.d purge (continuing)
  Removing any system startup links for run-postinsts ...
  /etc/rcS.d/S99run-postinsts
INIT: Entering runlevel: 5
Configuring network interfaces... udhcpc: started, v1.35.0
udhcpc: broadcasting discover
udhcpc: broadcasting discover
udhcpc: broadcasting discover
udhcpc: no lease, forking to background
done.
Starting Dropbear SSH server: Generating 2048 bit rsa key, this may take a while...
Public key portion is:
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQCuDpf5Lt1zuzxA69xsHoEcRlt9BgaGegJU7ywXzIIHZ7u4C6sAZohzt22+WK6A6wWs0kNXFcaeNuxFGjAwhb6yyPYSpf2a2+nz6Ub2zoZGjU5qa+B3m
+AIz4akwRWwE3LVdQBJD0neehsu1j6pTiqP/uP0/SoS63sxSylh62GB+PiI40ggcZwYRw2W1cRpHzuHKn0+cj4jxv2v6e3aY3v6NESF1Frvt8oiaHku9np6Wn/gKzGUQ7YpF0pMj9oocufsfJT3lyww
G4CIMNr9LHGehkMsRyqKTW2wXaoRe9g+dB+5EI/zXm+mi4+I128yetM5PTIAuWwtTnmaPlkDnsxF root@mb_linux
Fingerprint: SHA256:vQlu+UA4zXoHyutZwn1+rDmVBABZo078z7jsaN9XzE4
dropbear.
Starting internet superserver: inetd.
Starting syslogd/klogd: done
Starting tcf-agent: OK

PetaLinux 2023.2+release-S10121051 mb_linux ttyUL0

mb_linux login: petalinux
You are required to change your password immediately (administrator enforced).
New password:
Retype new password:
mb_linux:~$ ls /dev
```

Lab: Embedded Linux

1. You should observe spidev0.0 as present

```
mb_1@linux:~$ ls /dev
console  ptyc7  ptyp9  ptypsb  ptyvd  ptyyf  ttya9  ttydb  ttyqd  ttytf  ttx1
full     ptyc8  ptypa  ptypsc  ptyve  ptyz0  ttyaa  ttdc   ttyqe  ttx0  ttx2
initctl  ptyc9  ptypb  ptypsd  ptyvf  ptyz1  ttyab  ttdd   ttyqf  ttx1  ttx3
kmsg     ptyca  ptypc  ptyse   ptyw0  ptyz2  ttyac  ttdde  ttxr0  ttx2  ttx4
log      ptycb  ptypd  ptysf   ptyw1  ptyz3  ttyad  ttdf   ttxr1  ttx3  ttx5
mem      ptycc  ptype  ptyt0   ptyw2  ptyz4  ttyae  tte0   ttxr2  ttx4  ttx6
mtab    ptycd  ptypf  ptyt1   ptyw3  ptyz5  ttyaf  ttxel  ttxr3  ttx5  ttx7
null    ptyce  ptyq0  ptyt2   ptyw4  ptyz6  ttyb0  ttxe2  ttxr4  ttx6  ttx8
port    ptycf  ptyq1  ptyt3   ptyw5  ptyz7  ttyb1  ttxe3  ttxr5  ttx7  ttx9
ptmx    ptyd0  ptyq2  ptyt4   ptyw6  ptyz8  ttyb2  ttxe4  ttxr6  ttx8  ttxa
pts     ptyd1  ptyq3  ptyt5   ptyw7  ptyz9  ttyb3  ttxe5  ttxr7  ttx9  ttxb
ptya0   ptyd2  ptyq4  ptyt6   ptyw8  ptyza  ttyb4  ttxe6  ttxr8  ttxa  ttxc
ptya1   ptyd3  ptyq5  ptyt7   ptyw9  ptyzb  ttyb5  ttxe7  ttxr9  ttxb  ttxd
ptya2   ptyd4  ptyq6  ptyt8   ptywa  ptyzc  ttyb6  ttxe8  ttxra  ttxc  ttxe
ptya3   ptyd5  ptyq7  ptyt9   ptywb  ptyzd  ttyb7  ttxe9  ttxrb  ttxd  ttxf
ptya4   ptyd6  ptyq8  ptyta   ptywc  ptyze  ttyb8  ttxea  ttxrc  ttxe  ttx0
ptya5   ptyd7  ptyq9  ptytb   ptywd  ptyzf  ttyb9  ttxeb  ttxrd  ttxuf  ttx1
ptya6   ptyd8  ptyqa  ptytc   ptywe  ram0   ttyba  ttxec  ttxre  ttx0  ttx2
ptya7   ptyd9  ptyqb  ptytd   ptywf  ram1   ttybb  ttxed  ttxrf  ttx1  ttx3
ptya8   ptyda  ptyqc  ptyte   ptyx0  ram10  ttybc  ttxee  ttxs0  ttx2  ttx4
ptya9   ptydb  ptyqd  ptytf   ptyx1  ram11  ttybd  ttxef  ttxs1  ttx3  ttx5
ptyaa   ptydc  ptyqe  ptyu0   ptyx2  ram12  ttybe  ttxfp  ttxs2  ttx4  ttx6
ptyab   ptydd  ptyqf  ptyu1   ptyx3  ram13  ttybf  ttxp1  ttxs3  ttx5  ttx7
ptyac   ptyde  ptyr0  ptyu2   ptyx4  ram14  ttyc0  ttxp2  ttxs4  ttx6  ttx8
ptyad   ptydf  ptyr1  ptyu3   ptyx5  ram15  ttyc1  ttxp3  ttxs5  ttx7  ttx9
ptyae   ptye0  ptyr2  ptyu4   ptyx6  ram2   ttyc2  ttxp4  ttxs6  ttx8  ttxa
ptyaf   ptye1  ptyr3  ptyu5   ptyx7  ram3   ttyc3  ttxp5  ttxs7  ttx9  ttxb
ptyb0   ptye2  ptyr4  ptyu6   ptyx8  ram4   ttyc4  ttxp6  ttxs8  ttxa  ttxc
ptyb1   ptye3  ptyr5  ptyu7   ptyx9  ram5   ttyc5  ttxp7  ttxs9  ttxb  ttxd
ptyb2   ptye4  ptyr6  ptyu8   ptyxa  ram6   ttyc6  ttxp8  ttxsa  ttxvc  ttxe
ptyb3   ptye5  ptyr7  ptyu9   ptyxb  ram7   ttyc7  ttxp9  ttxsb  ttxvd  ttxf
ptyb4   ptye6  ptyr8  ptyua  ptyxc  ram8   ttyc8  ttxpa  ttxsc  ttxve  ttxz0
ptyb5   ptye7  ptyr9  ptyub  ptyxd  ram9   ttyc9  ttxpb  ttxsd  ttxvf  ttxz1
ptyb6   ptye8  ptyra  ptyuc  ptyxe  random  ttyca  ttxpc  ttxse  ttxw0  ttxz2
ptyb7   ptye9  ptyrb  ptyud  ptyxf  spidev0.0  ttycb  ttxpd  ttxsf  ttxw1  ttxz3
ptyb8   ptyea  ptyrc  ptyue  pty0   ttxy  ttxcc  ttxpe  ttxt0  ttxw2  ttxz4
ptyb9   ptyeb  ptyrd  ptyuf  pty1   ttxy0  ttxcd  ttxpf  ttxt1  ttxw3  ttxz5
ptyba   ptyec  ptyre  ptyv0  pty2   ttxy1  ttxce  ttxq0  ttxt2  ttxw4  ttxz6
ptybb   ptyed  ptyrf  ptyv1  pty3   ttxy2  ttxcf  ttxq1  ttxt3  ttxw5  ttxz7
ptybc   ptyee  ptyso  ptyv2  pty4   ttxy3  ttxq2  ttxt4  ttxw6  ttxz8
ptybd   ptyef  ptysl  ptyv3  pty5   ttxy4  ttxq3  ttxt5  ttxw7  ttxz9
ptybe   ptyp0  ptyss  ptyv4  pty6   ttxy5  ttxq4  ttxt6  ttxw8  ttxza
ptybf   ptyp1  ptyss  ptyv5  pty7   ttxy6  ttxq5  ttxt7  ttxw9  ttxzb
ptyc0   ptyp2  ptyss  ptyv6  pty8   ttxy7  ttxq6  ttxt8  ttxw8  ttxzc
ptyc1   ptyss  ptyss  ptyv7  pty9   ttxy8  ttxq7  ttxt9  ttxw9  ttxzd
ptyc2   ptyp4  ptyss  ptyv8  ptya   ttxy9  ttxq8  ttxta  ttxwc  ttxze
ptyc3   ptyss  ptyss  ptyv9  ptyb   ttxy0  ttxq9  ttxtb  ttxwd  ttxzf
```

Lab: Embedded Linux

1. Get the IP address of the board

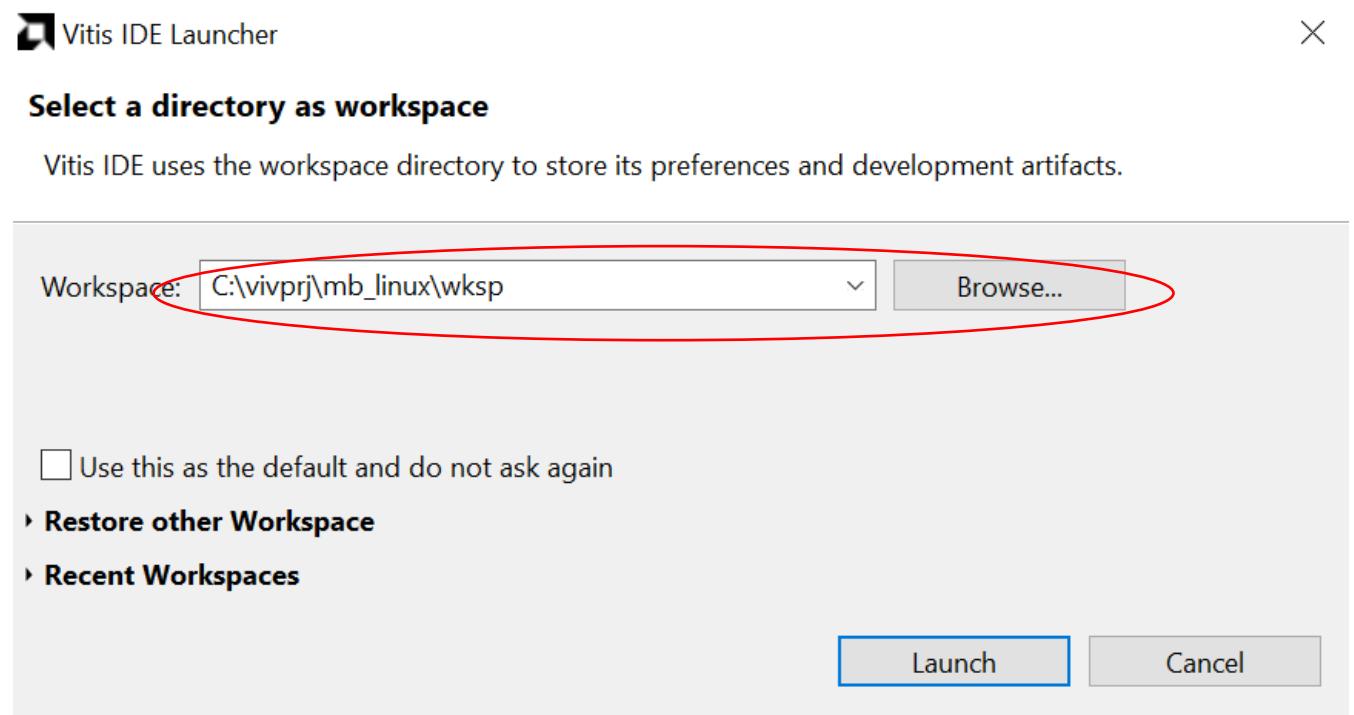
```
mb_linux:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0A:35:00:84:79
          inet addr:10.10.30.13 Bcast:10.10.30.255  Mask:255.255.255.0
                  UP BROADCAST RUNNING MTU:1500  Metric:1
                  RX packets:990 errors:0 dropped:607 overruns:0 frame:0
                  TX packets:64 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:1000
                  RX bytes:1066065 (1.0 MiB)  TX bytes:7468 (7.2 KiB)
                  Interrupt:4 Memory:40e00000-40e0ffff

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
                  UP LOOPBACK RUNNING MTU:65536  Metric:1
                  RX packets:0 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:1000
                  RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

```
mb_linux:~$ █
```

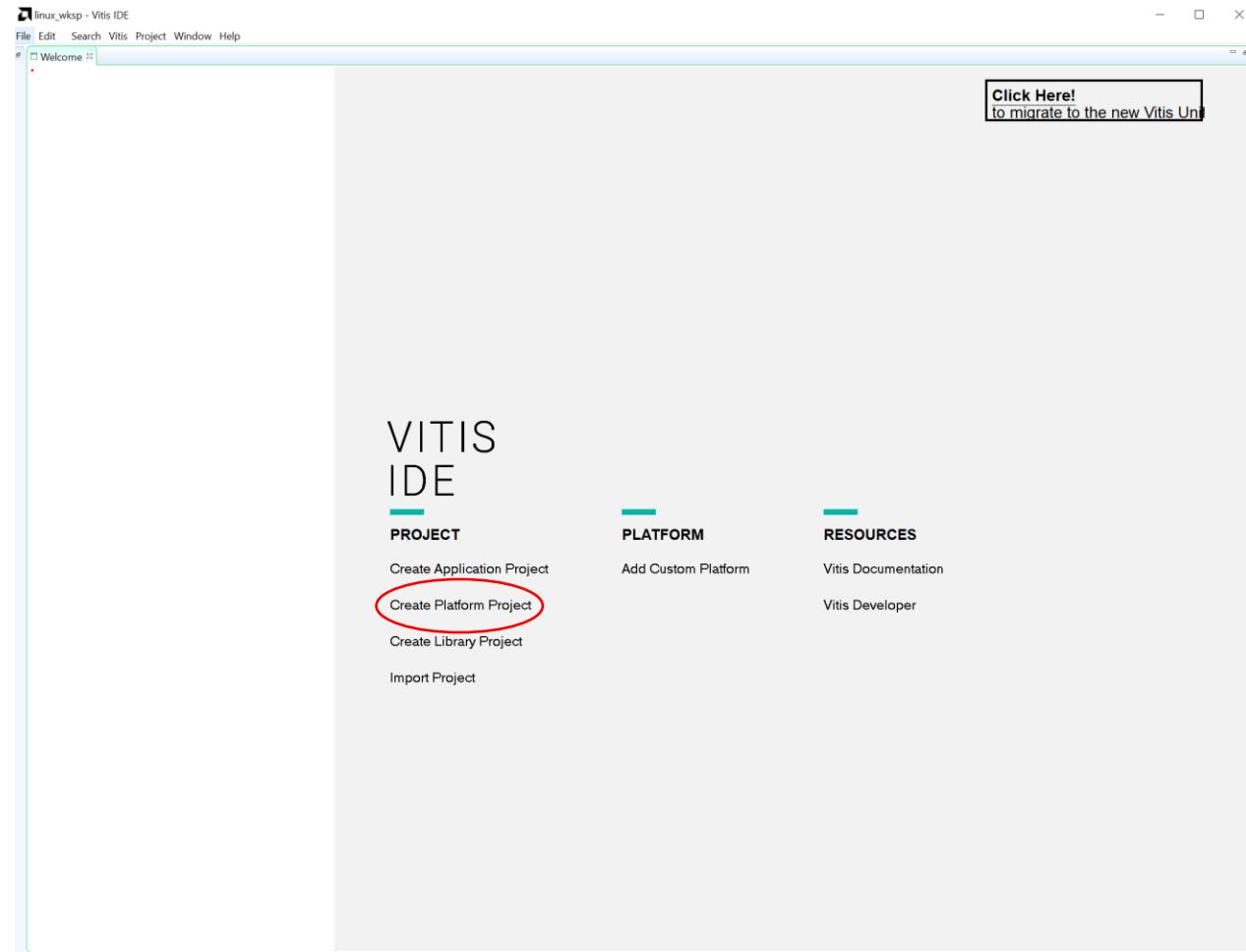
Lab: Embedded Linux

1. Open the AMD Vitis™ software and select the workspace you want to work with



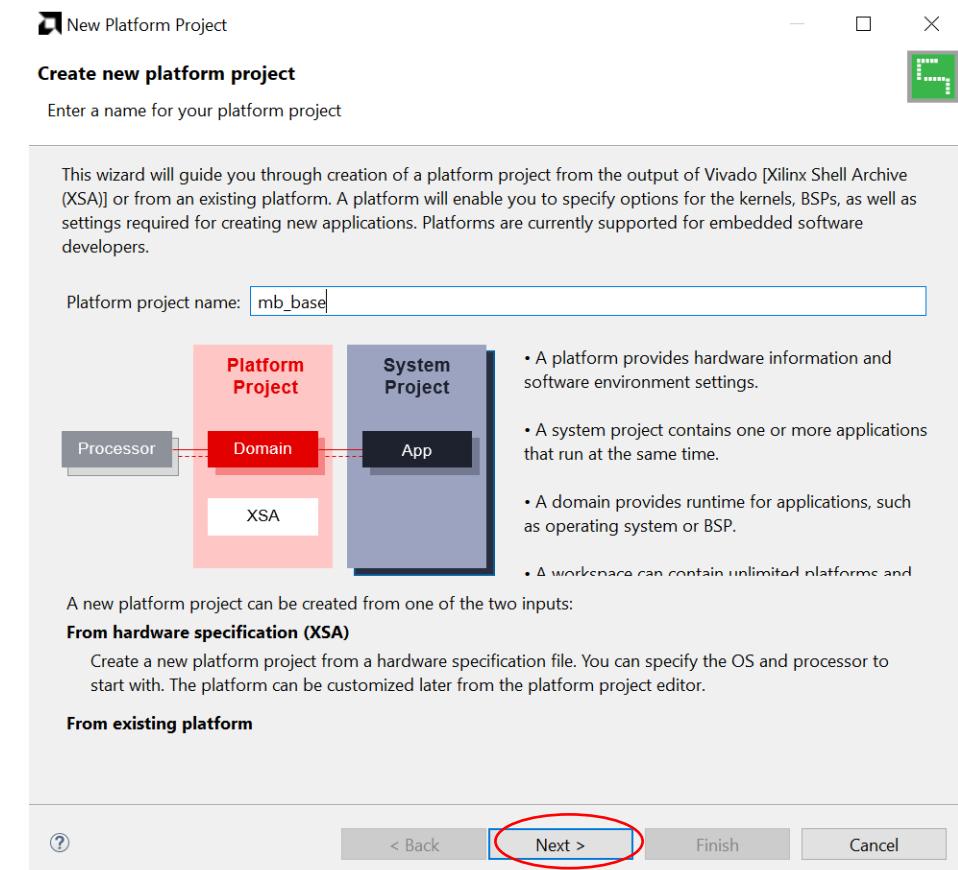
Lab: Embedded Linux

1. Create a Platform Project



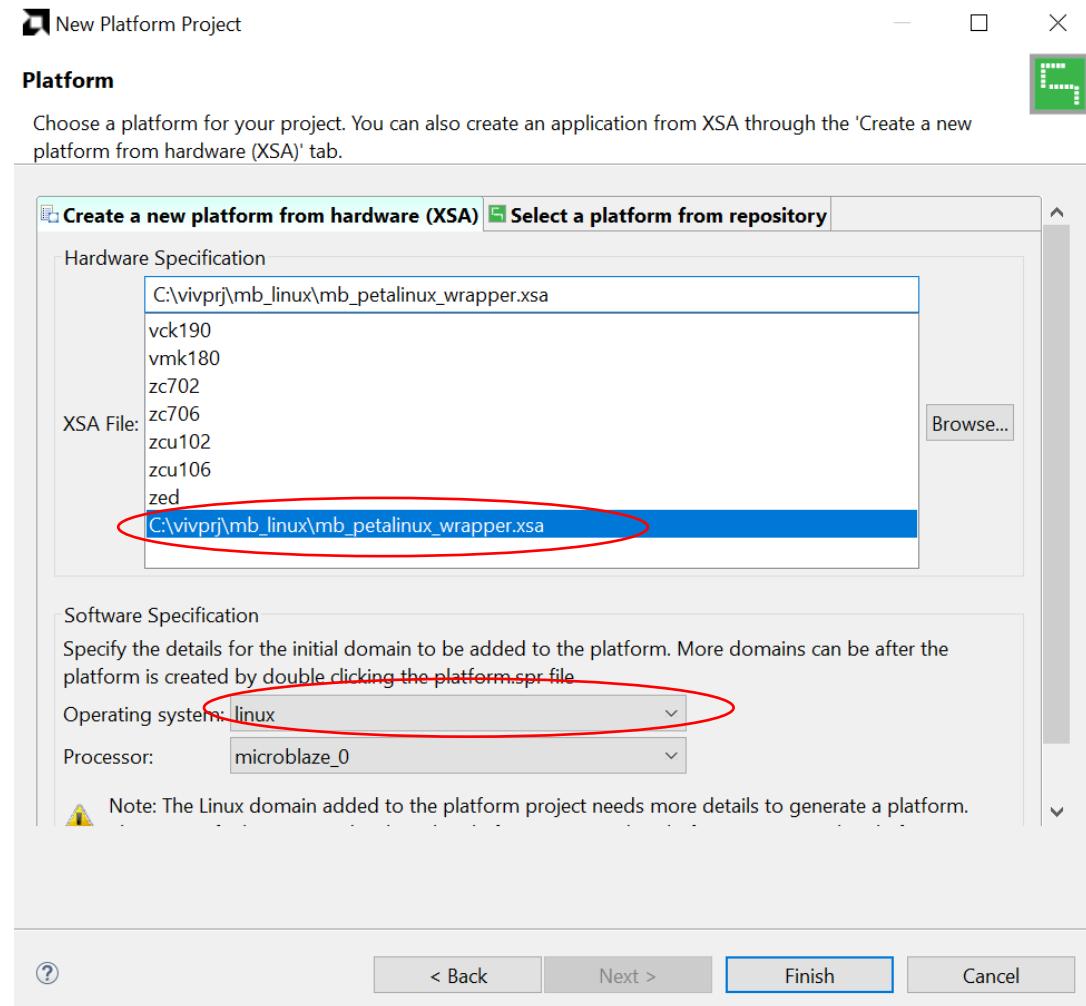
Lab: Embedded Linux

1. Click Next



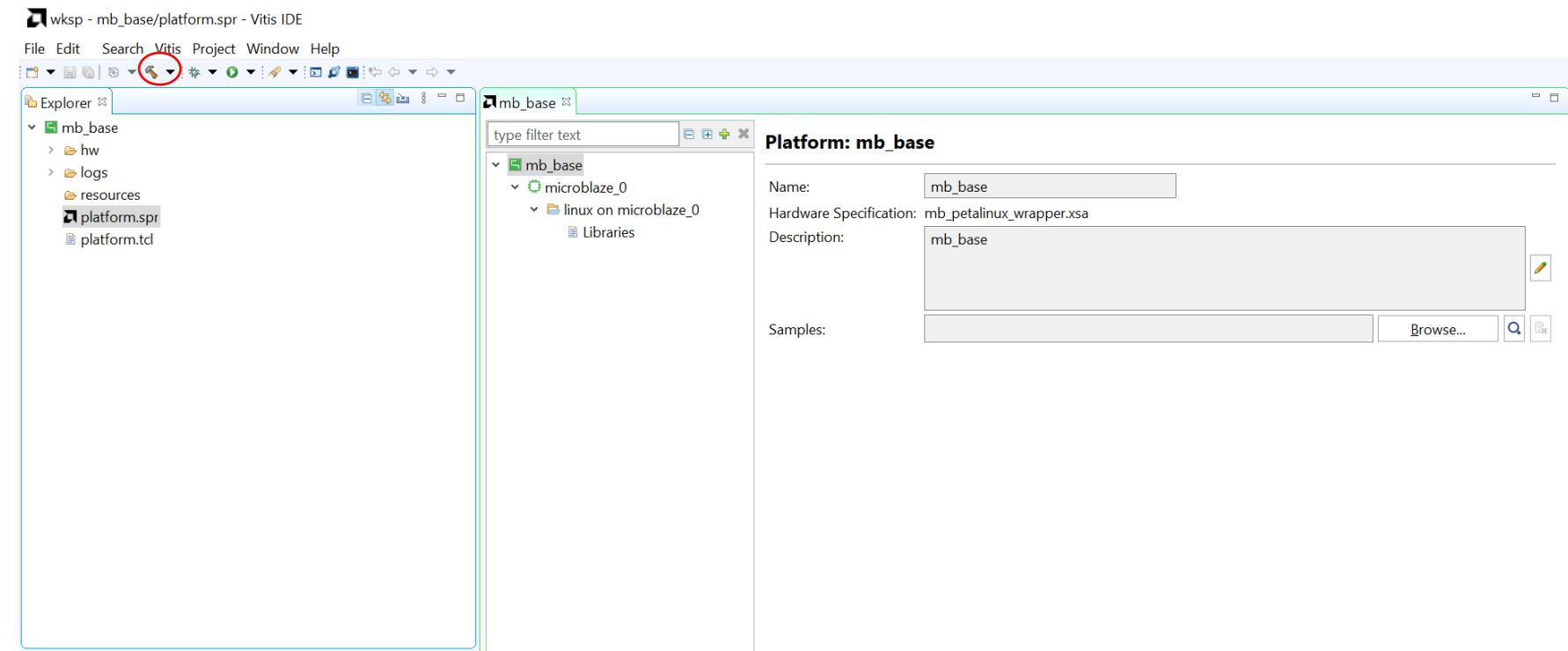
Lab: Embedded Linux

1. Select the XSA exported from the AMD Vivado™ software
2. Select the operating system as Linux



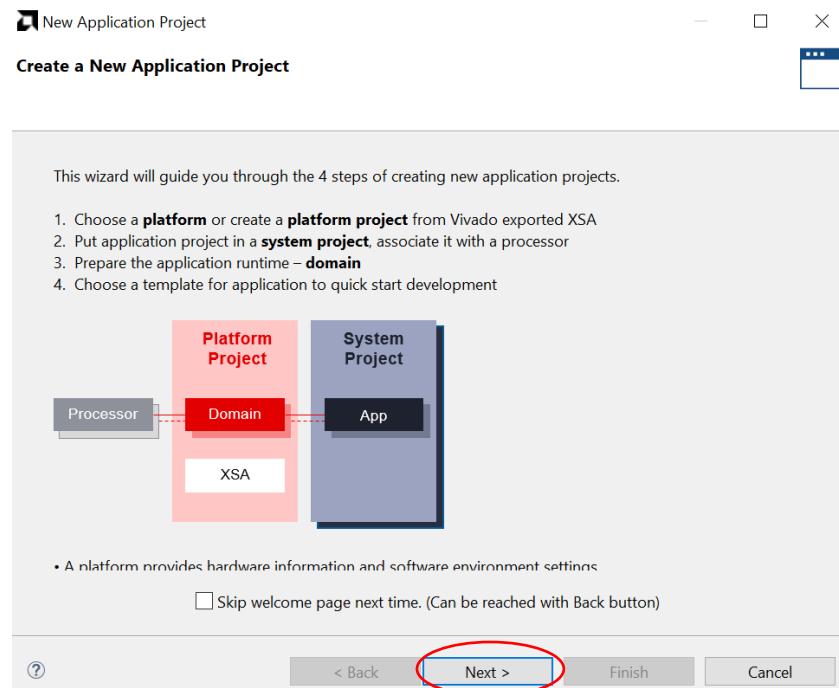
Lab: Embedded Linux

1. When the platform opens click on the hammer to compile it



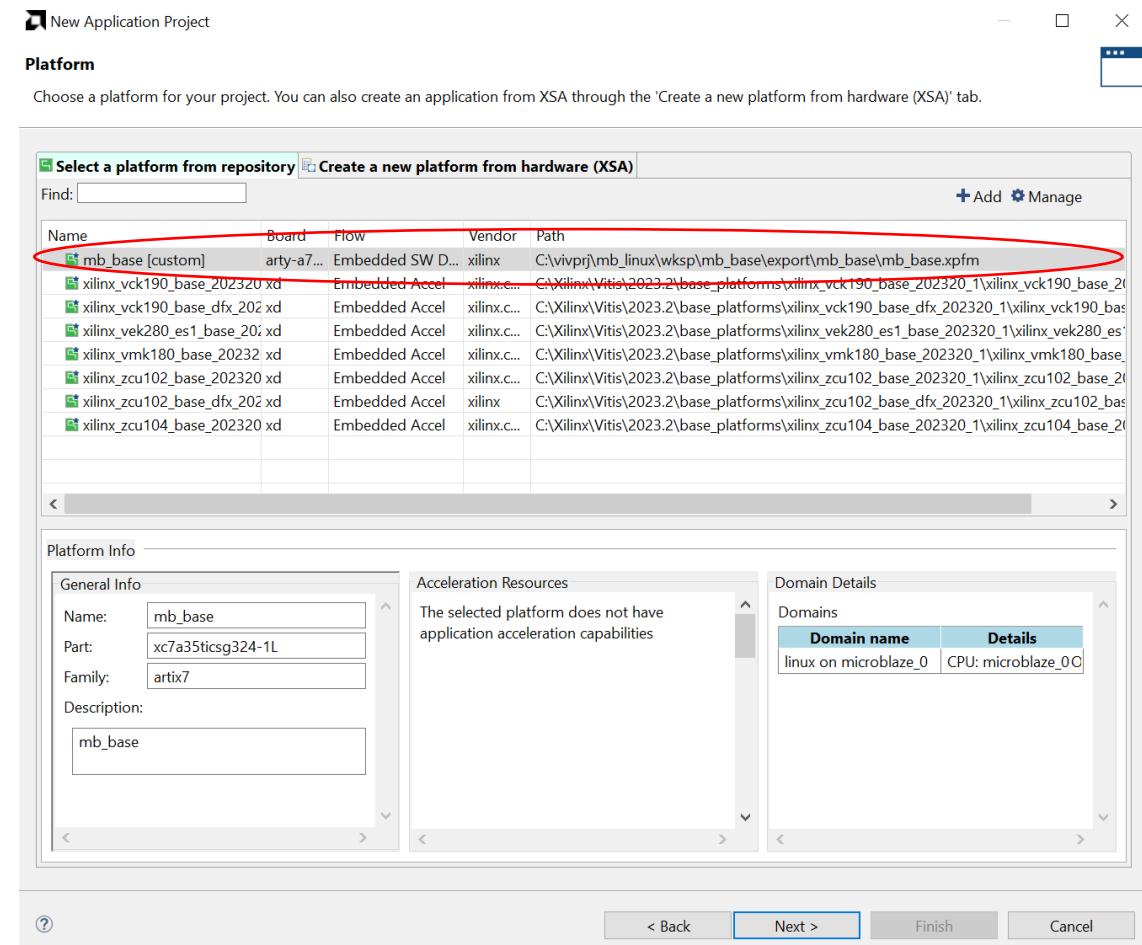
Lab: Embedded Linux

1. Create a new application project
2. Click Next



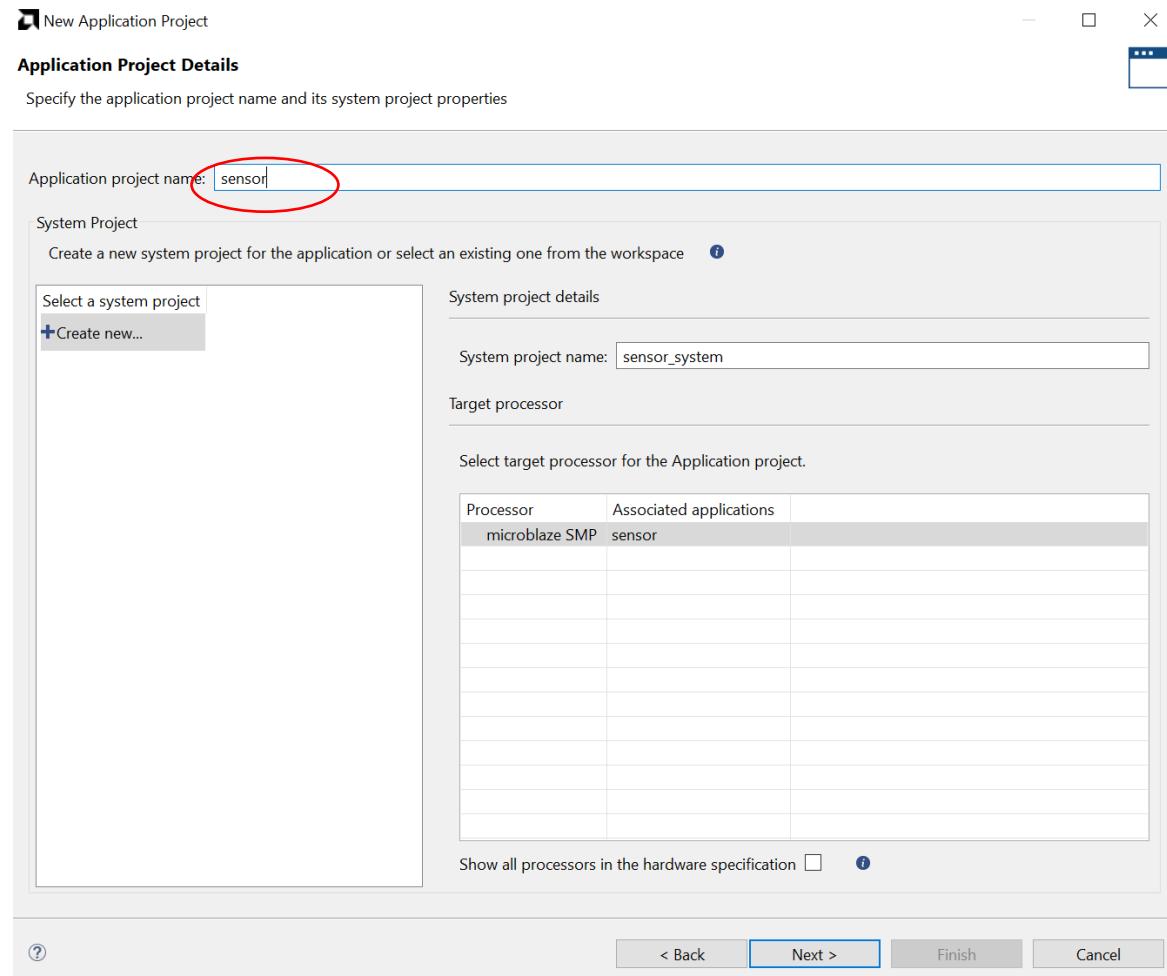
Lab: Embedded Linux

1. Select the platform we just created



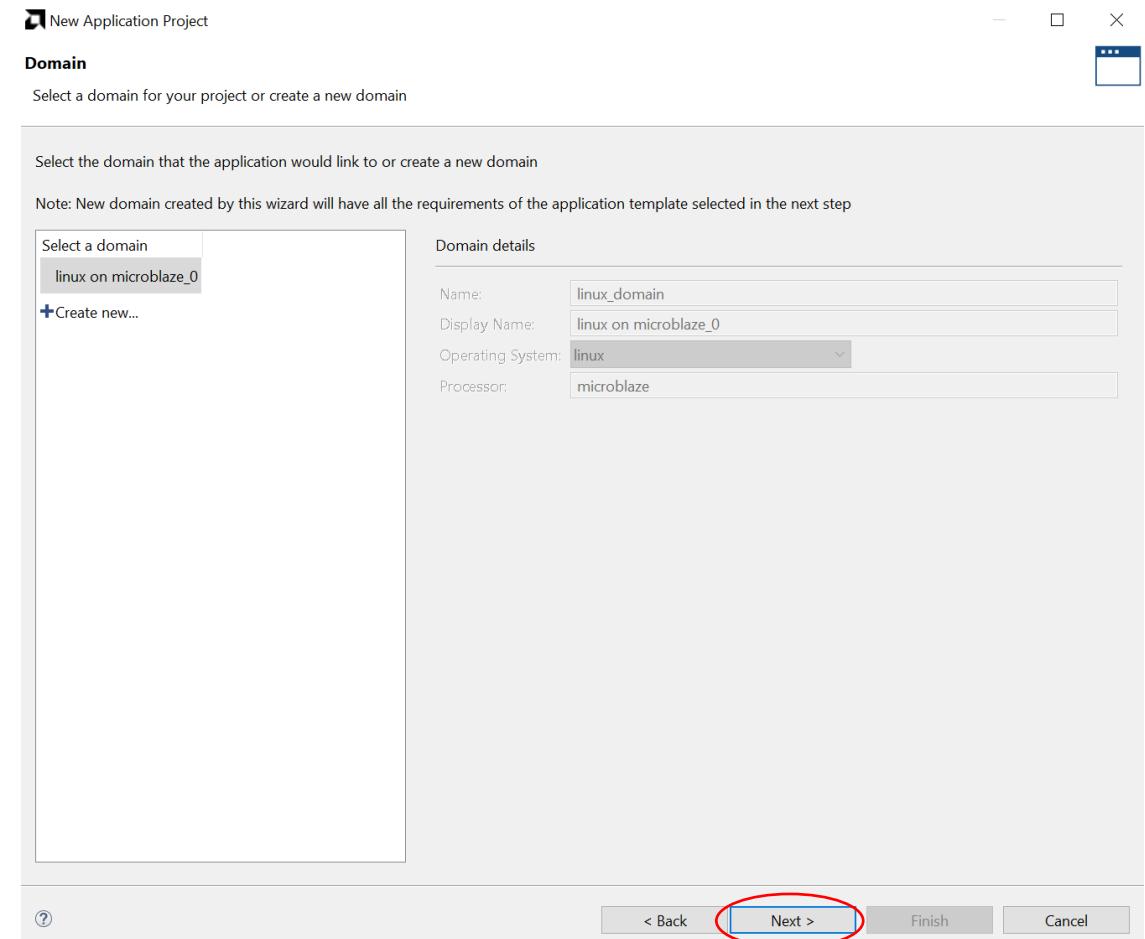
Lab: Embedded Linux

1. Name the project and click next



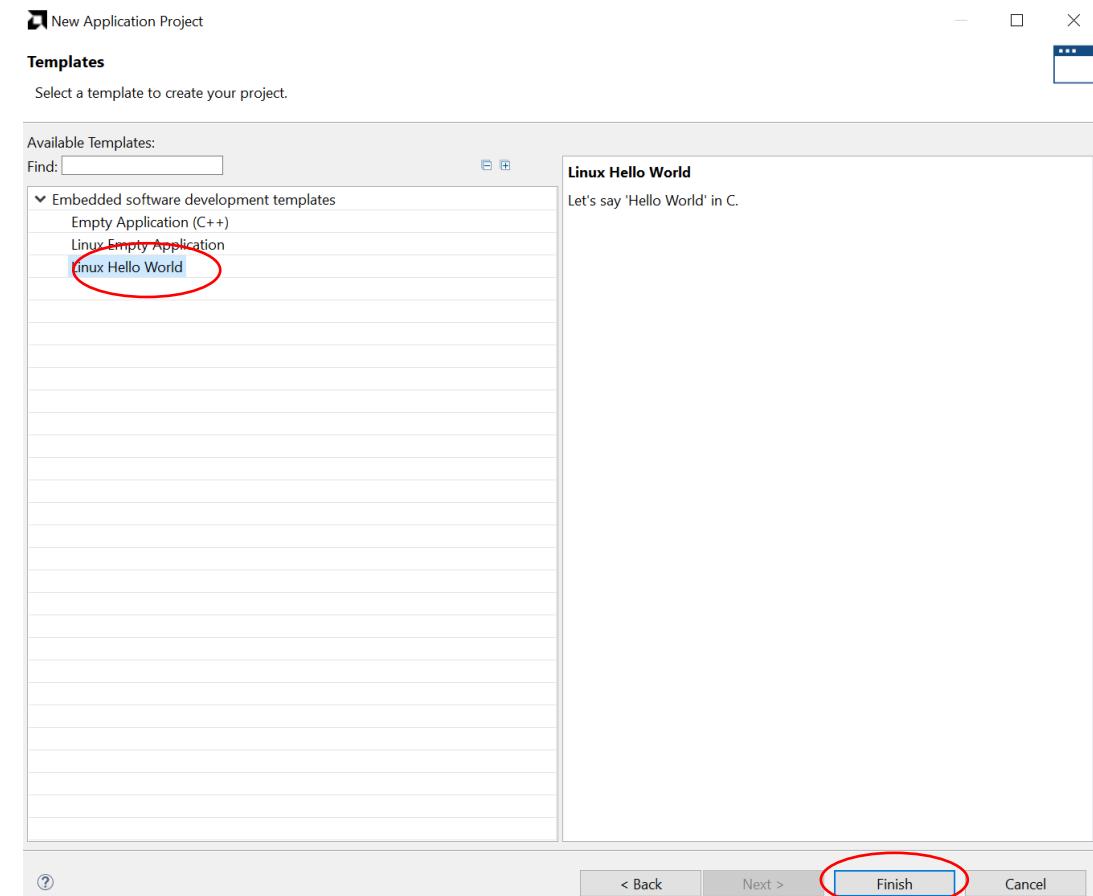
Lab: Embedded Linux

1. Click Next



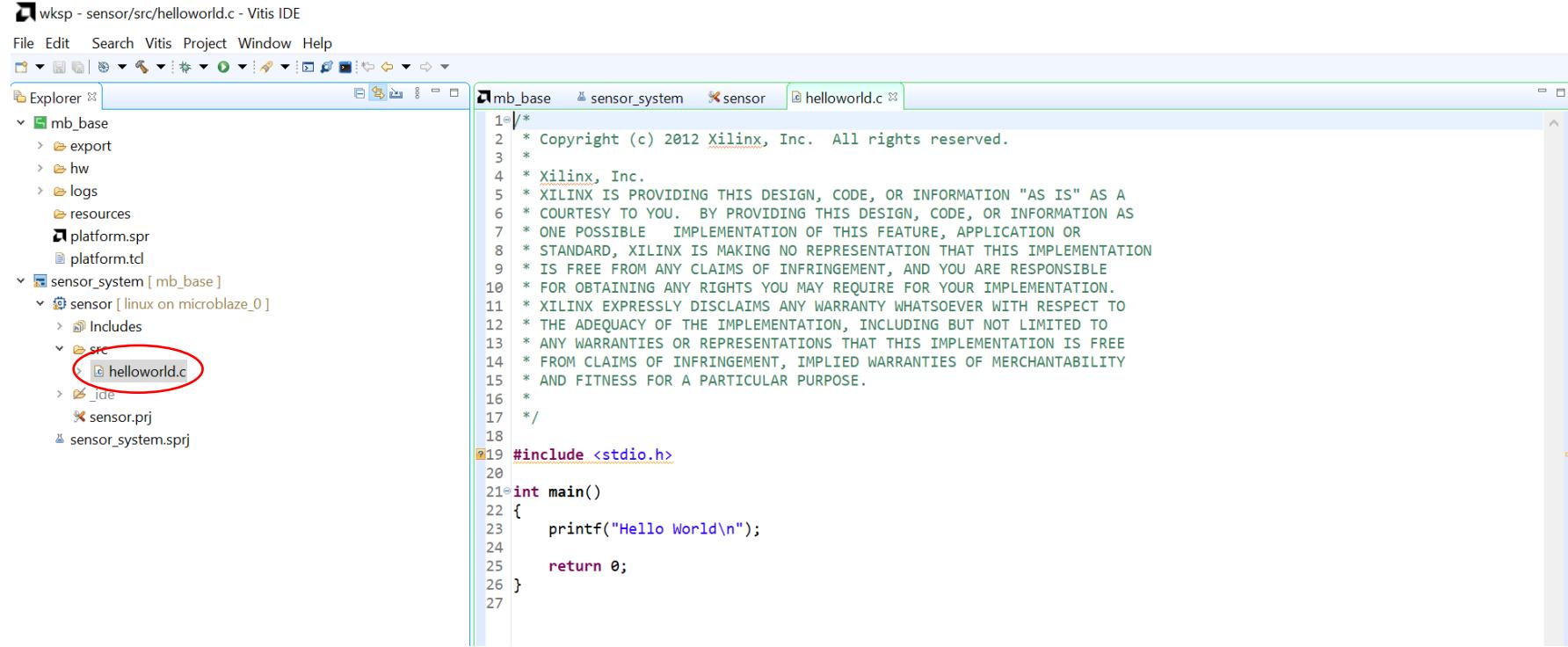
Lab: Embedded Linux

1. Select the Linux hello world
2. Click Next



Lab: Embedded Linux

1. Open the helloworld.c source and delete the contents

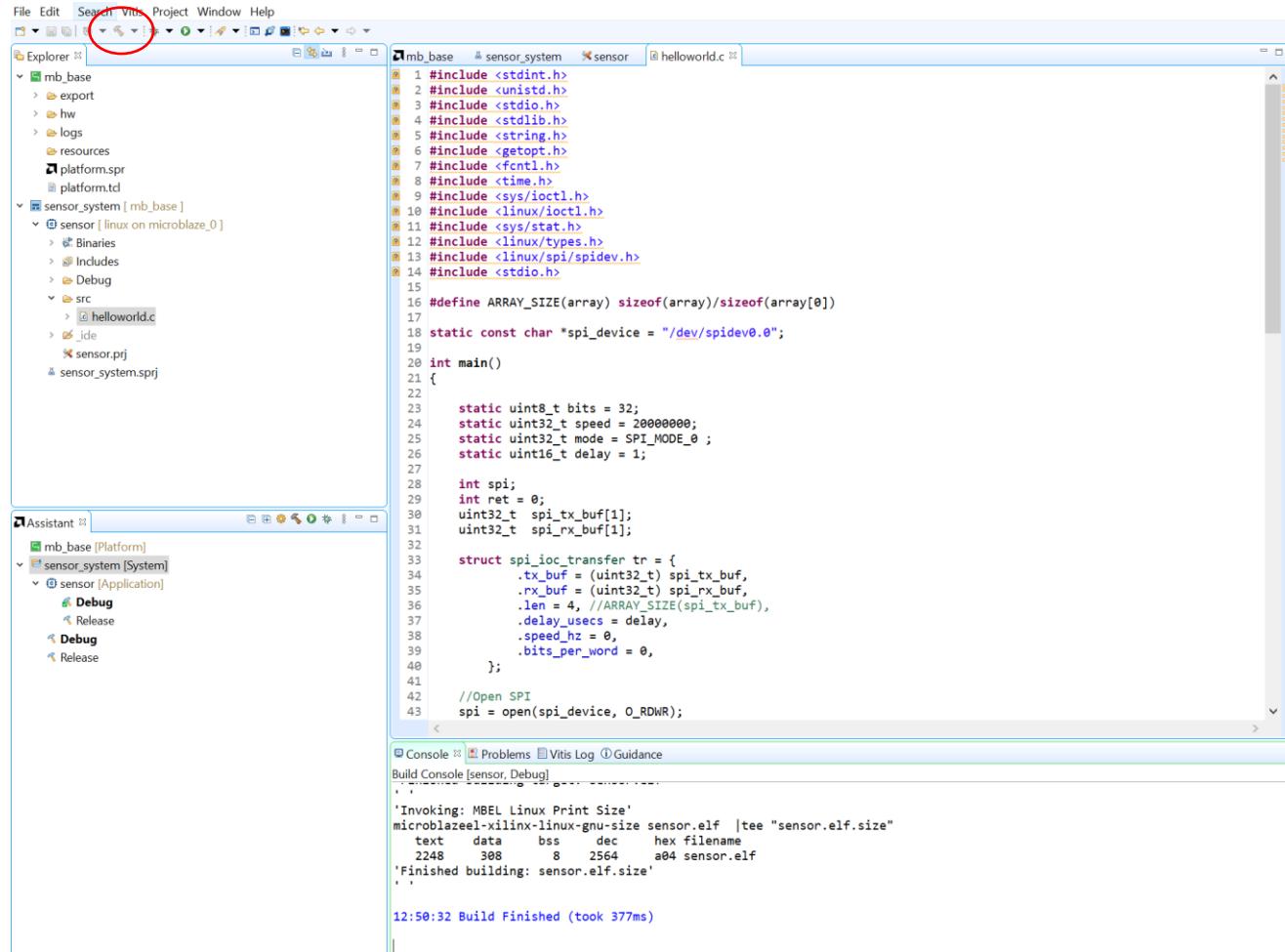


The screenshot shows the Vitis IDE interface. On the left, the Explorer view displays a project structure under 'sensor_system [mb_base]'. Inside 'src', the file 'helloworld.c' is highlighted with a red circle. On the right, the code editor shows the contents of 'helloworld.c'. The code includes a standard C header and a main function that prints "Hello World\n".

```
1/*  
2 * Copyright (c) 2012 Xilinx, Inc. All rights reserved.  
3 *  
4 * Xilinx, Inc.  
5 * XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS" AS A  
6 * COURTESY TO YOU. BY PROVIDING THIS DESIGN, CODE, OR INFORMATION AS  
7 * ONE POSSIBLE IMPLEMENTATION OF THIS FEATURE, APPLICATION OR  
8 * STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THIS IMPLEMENTATION  
9 * IS FREE FROM ANY CLAIMS OF INFRINGEMENT, AND YOU ARE RESPONSIBLE  
10 * FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE FOR YOUR IMPLEMENTATION.  
11 * XILINX EXPRESSLY DISCLAIMS ANY WARRANTY WHATSOEVER WITH RESPECT TO  
12 * THE ADEQUACY OF THE IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO  
13 * ANY WARRANTIES OR REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE  
14 * FROM CLAIMS OF INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY  
15 * AND FITNESS FOR A PARTICULAR PURPOSE.  
16 *  
17 */  
18  
19 #include <stdio.h>  
20  
21 int main()  
22 {  
23     printf("Hello World\n");  
24  
25     return 0;  
26 }  
27
```

Lab: Embedded Linux

1. Copy in the source from the provided git repo
2. Click on the hammer to build the application



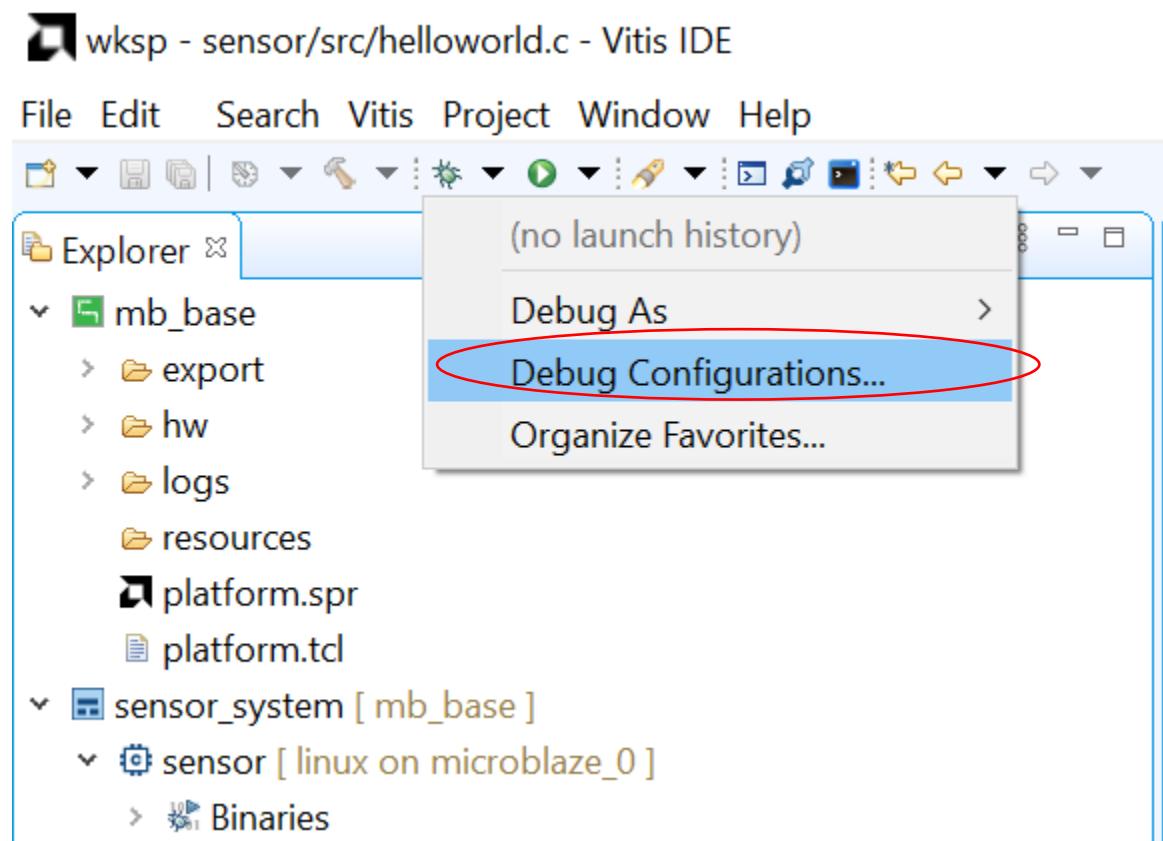
The screenshot shows the Vitis IDE interface. The top menu bar includes File, Edit, Search, Vitis, Project, Window, and Help. The toolbar contains various icons, with the hammer icon circled in red. The Explorer view on the left shows the project structure under mb_base and sensor_system. The code editor view in the center displays helloworld.c, which includes headers like stdint.h, unistd.h, stdio.h, stdlib.h, string.h, getopt.h, fcntl.h, time.h, sys/ioctl.h, linux/ioctl.h, sys/stat.h, sys/types.h, and linux/spi/spidev.h. It defines a SPI interface with static variables for device, speed, mode, and delay, and a main function that initializes the SPI and performs a transfer. The Assistant view at the bottom shows build configurations for Debug and Release. The bottom right corner shows a terminal window with a build log:

```
'Invoking: MBEL Linux Print Size'
microblazeel-xilinx-linux-gnu-size sensor.elf |tee "sensor.elf.size"
    text    data     bss   dec   hex filename
  2248     308       8   2564   a04 sensor.elf
'Finished building: sensor.elf.size'

12:50:32 Build Finished (took 377ms)
```

Lab: Embedded Linux

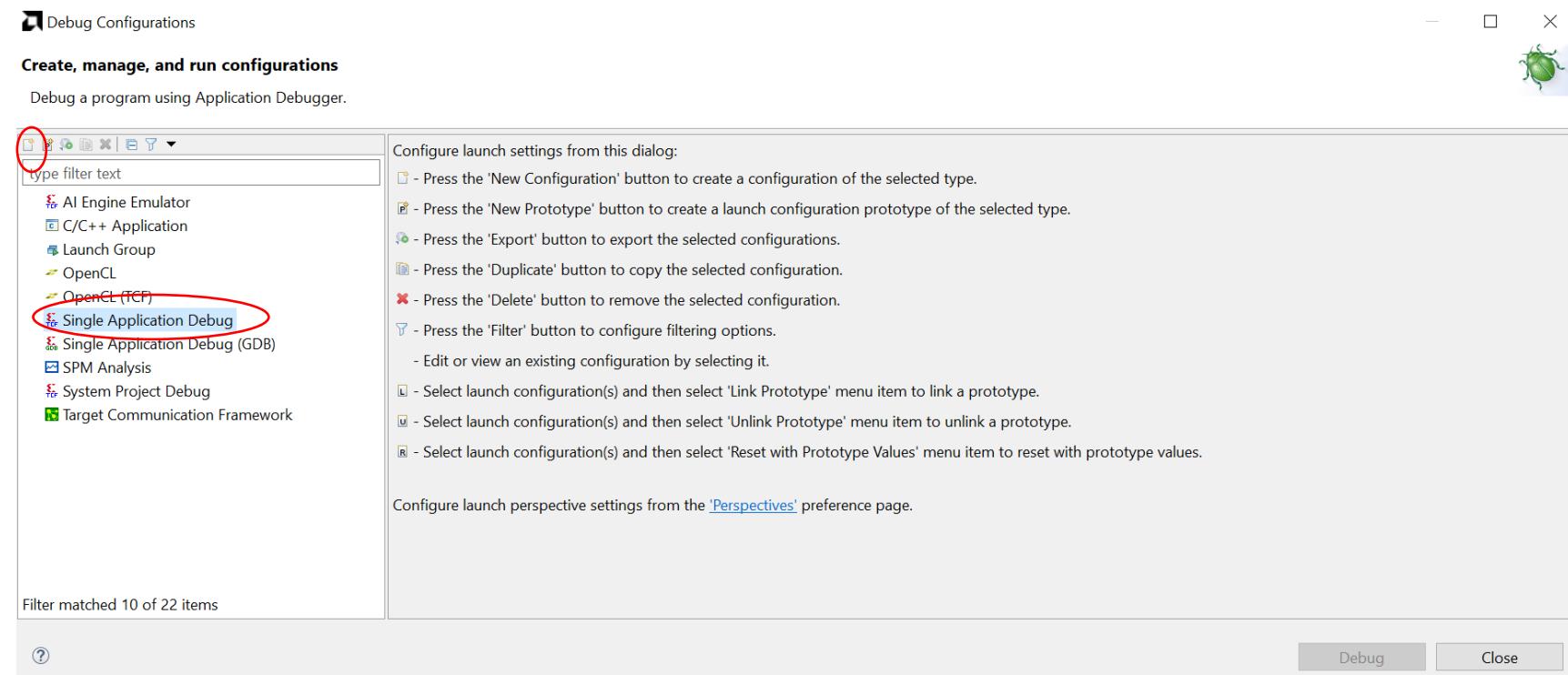
1. From the debug menu select Debug Configurations



Lab: Embedded Linux

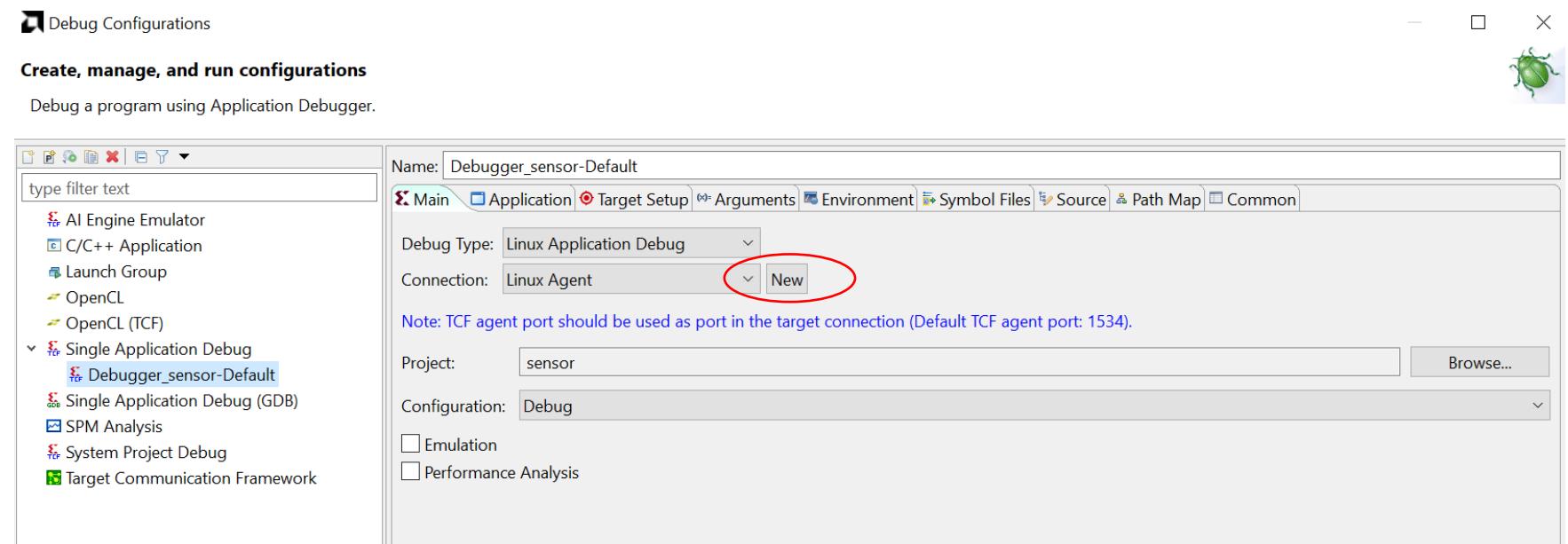
1. Select Single application debug

2. Click new



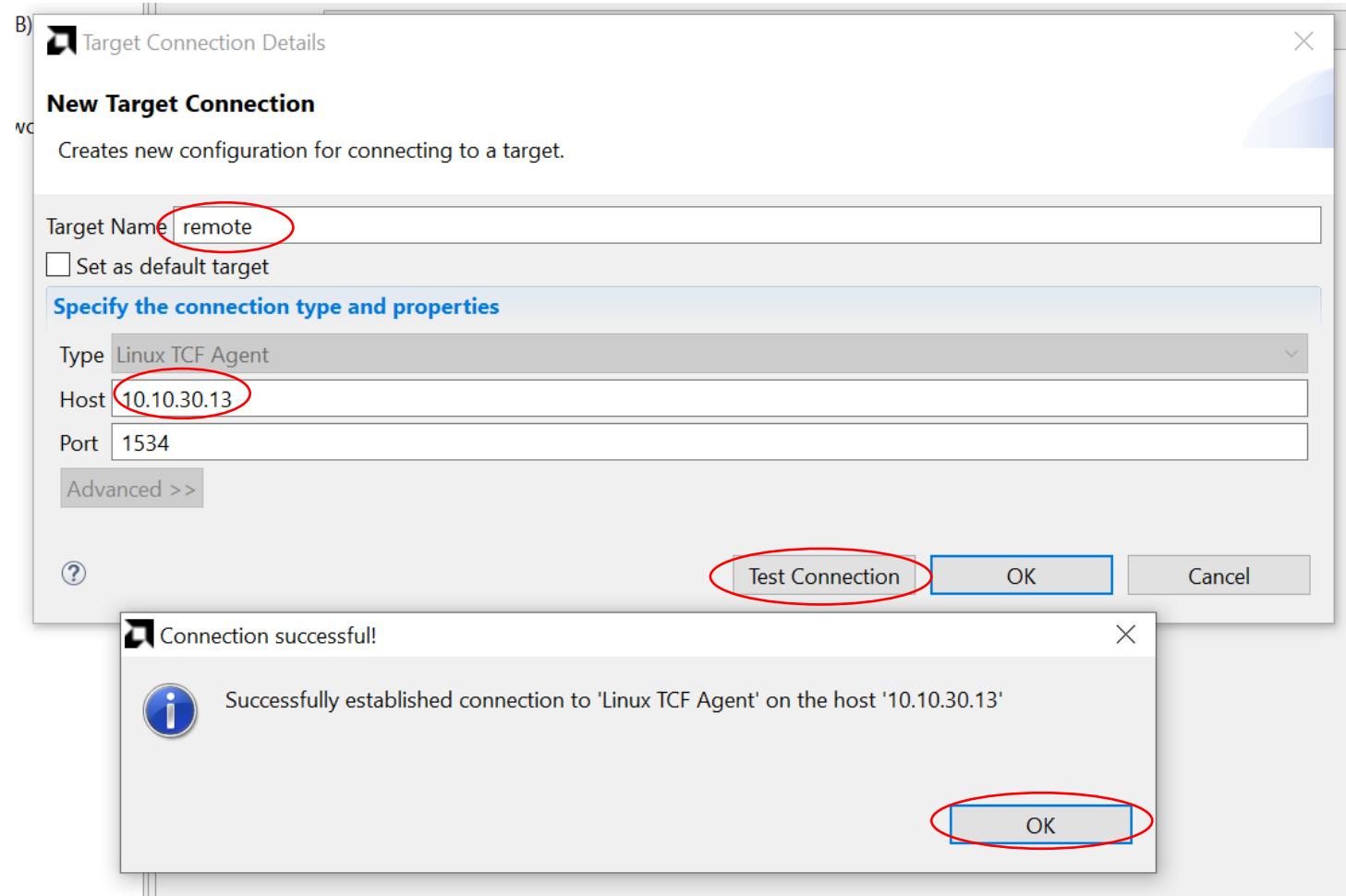
Lab: Embedded Linux

1. Click new on the connection type



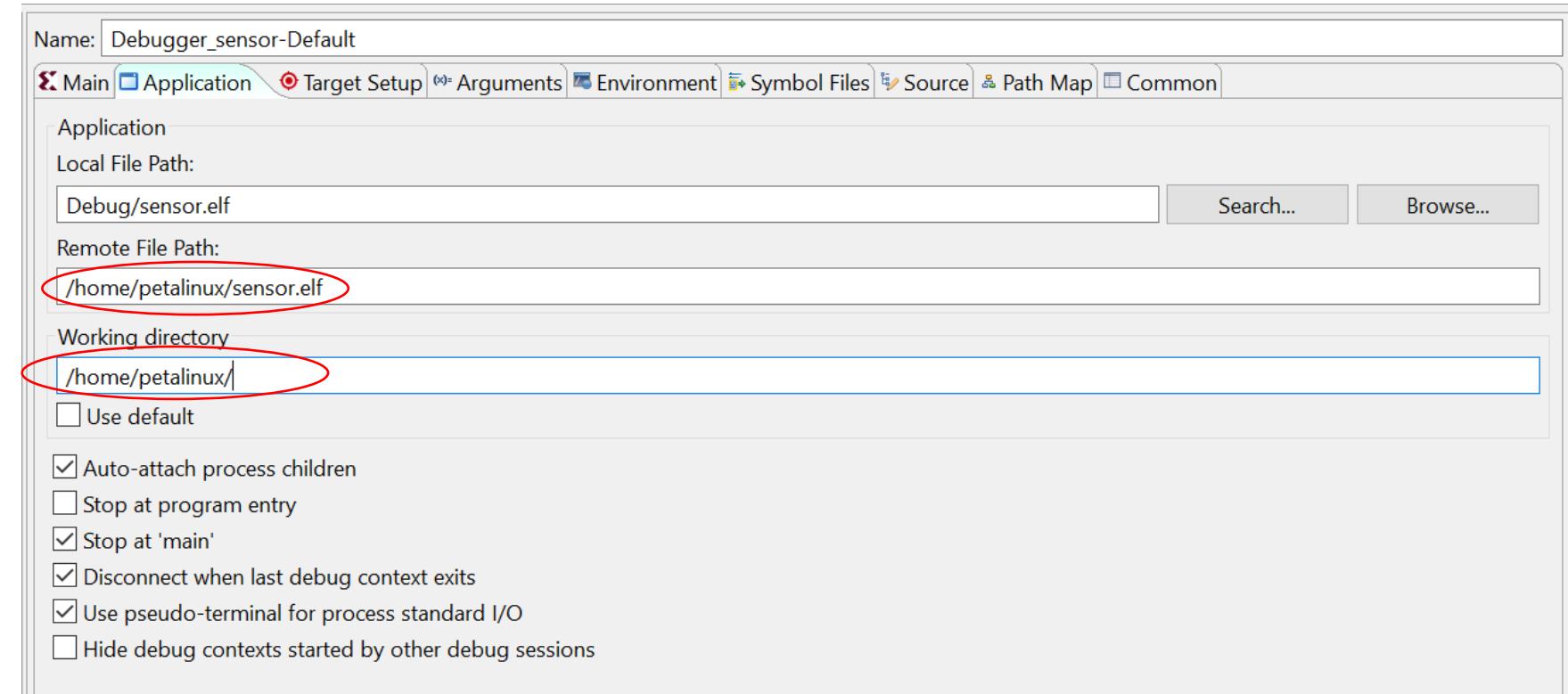
Lab: Embedded Linux

1. Enter a target name
2. Enter the board IP address
3. Click Test Connection
4. Click Ok



Lab: Embedded Linux

1. Change the remote file path and working directory to be /home/petalinux/



2. Click Debug

Lab: Embedded Linux

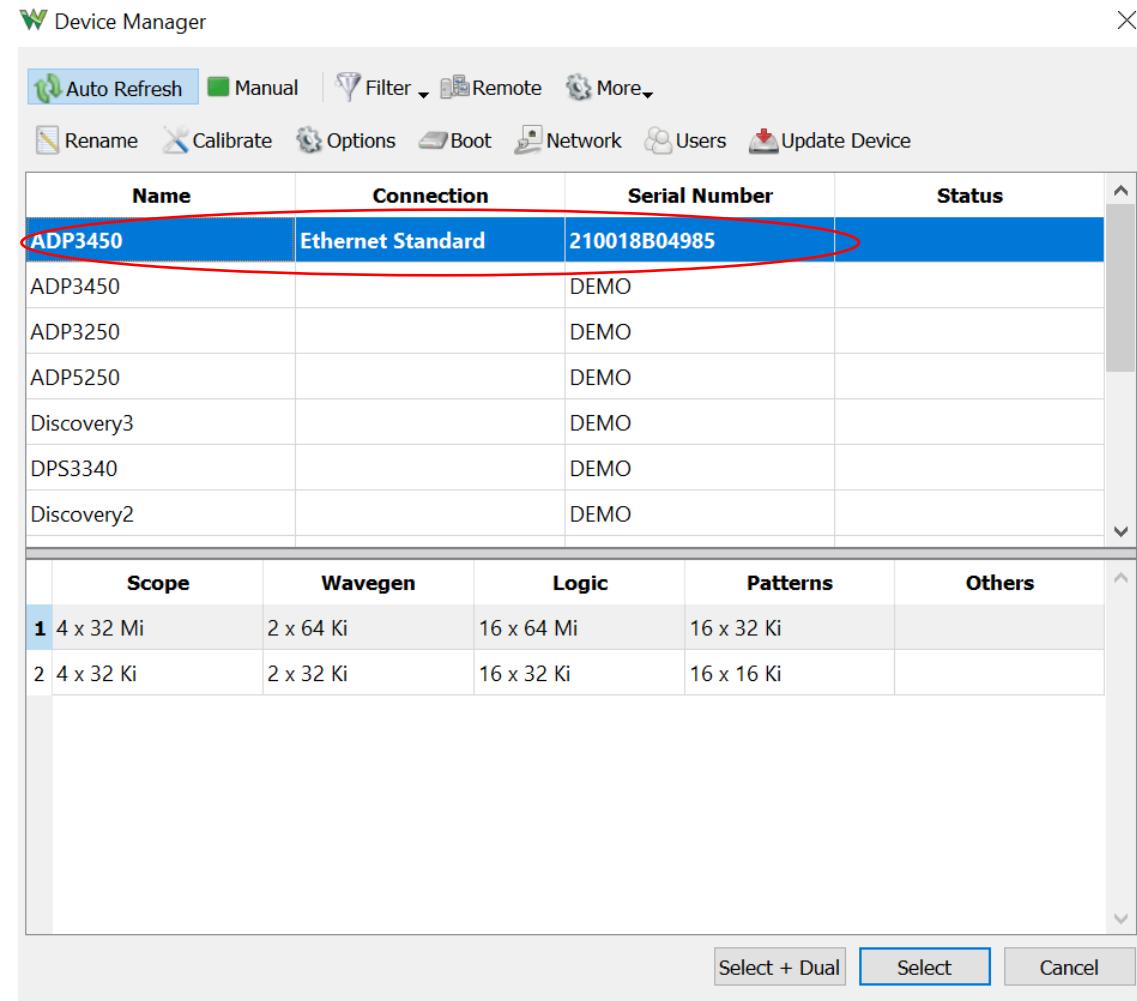
1. This will download the application on to the target board and pause for execution

The screenshot shows the Vitis IDE interface with several windows open:

- Top Left:** A file browser window titled "Explorer" showing project structure under "mb_base" and "sensor_system".
- Top Right:** The main code editor window titled "helloworld.c" showing C code for a SPI transfer. A line of code `int ret = 0;` is highlighted and circled in red.
- Middle Right:** A "Variables" window showing memory values for variables like bits, speed, mode, delay, spi, ret, and pointers to buffers (spi_tx_buf, spi_rx_buf).
- Bottom Right:** A "XSCT Console" window showing the command-line interface with the message: "xsct% Info: 327 (target 2) Stopped at 0x100 ^ main() at ../src/helloworld.c: 29 29: int ret = 0; xsct%".
- Bottom Center:** A "Console" window titled "Vitis Serial Terminal" showing the same stopped state information.

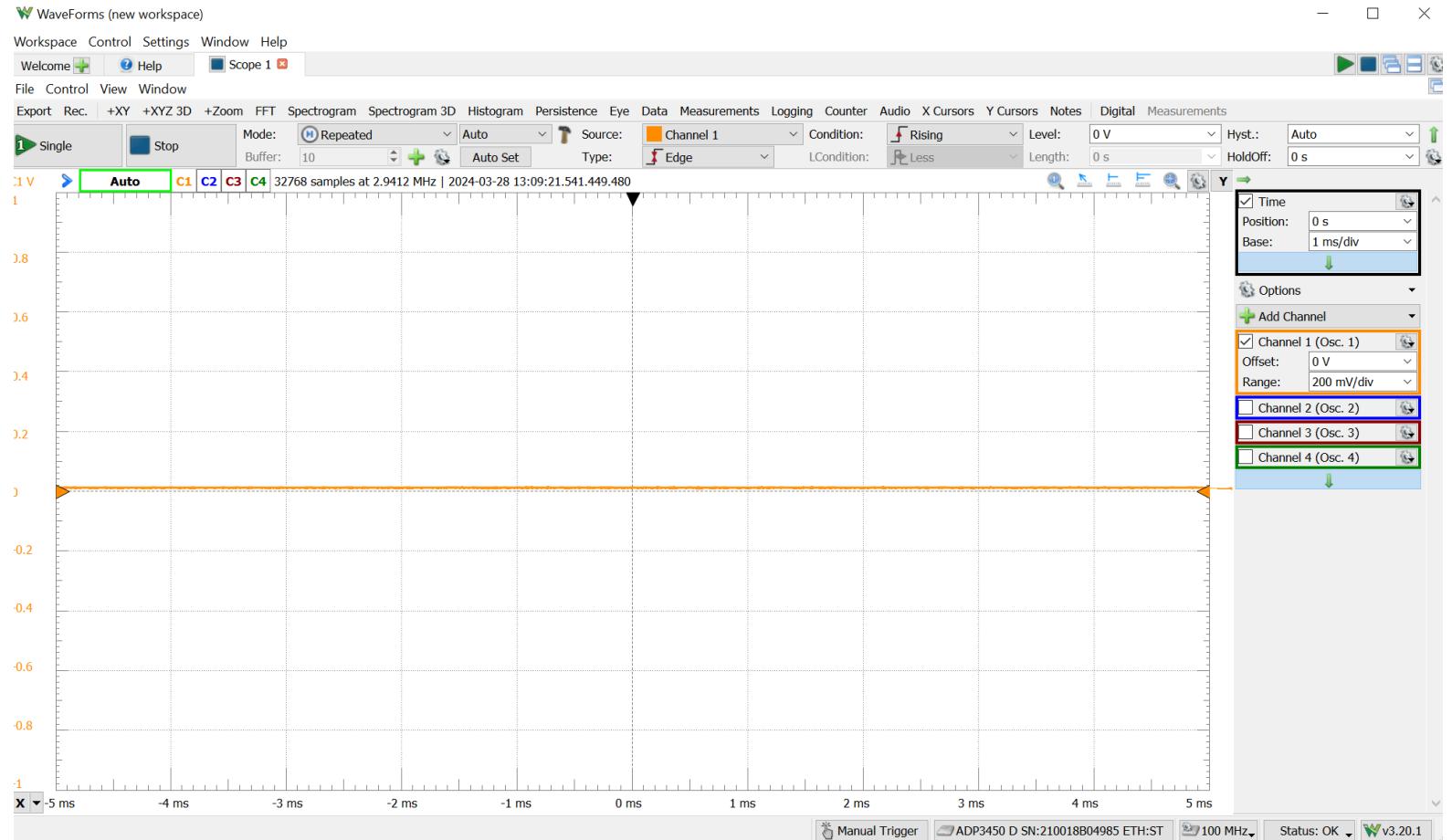
Lab: Embedded Linux

1. Open Waveforms and connect to the oscilloscope



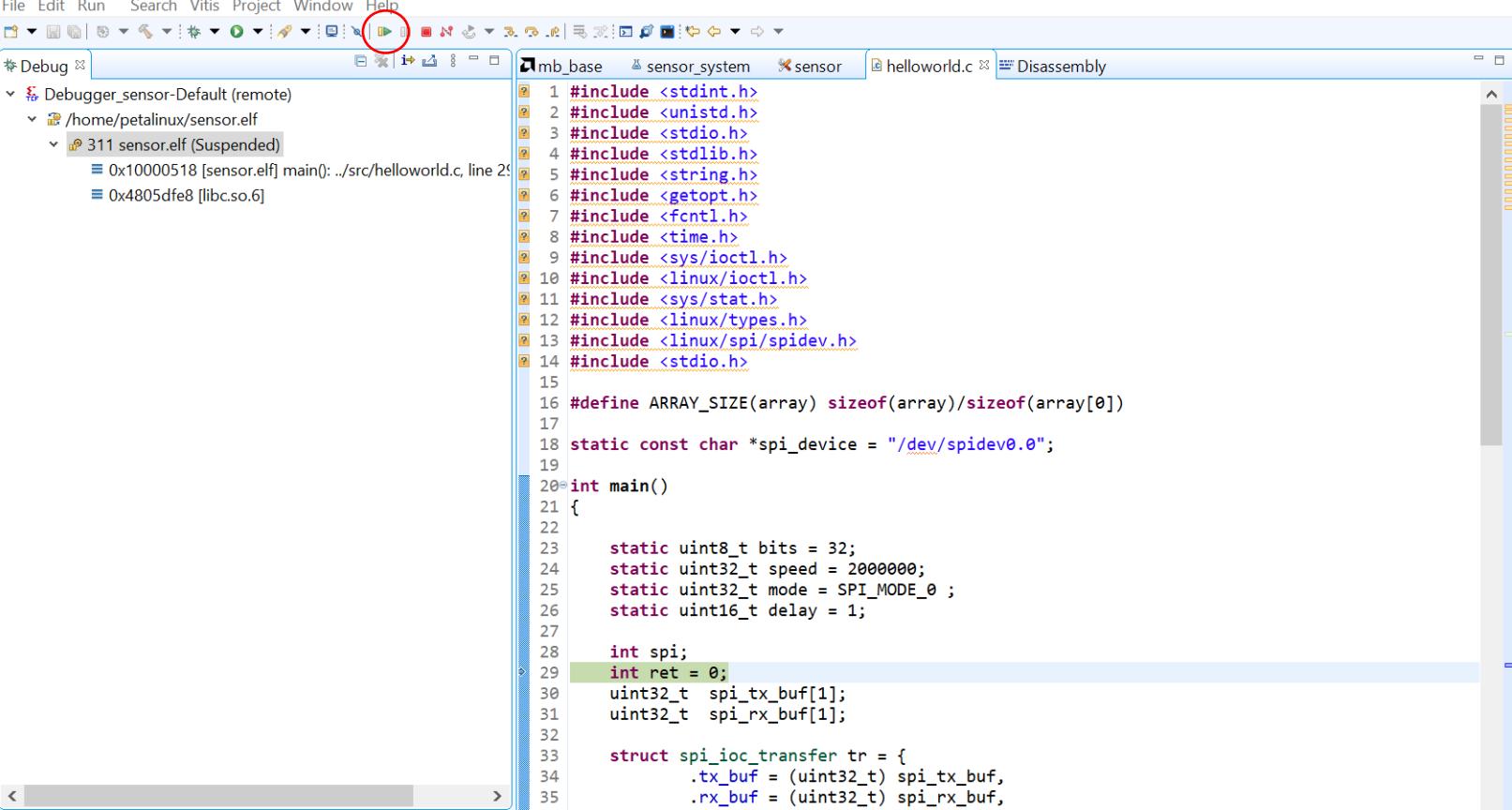
Lab: Embedded Linux

1. Open the scope view and run the scope



Lab: Embedded Linux

1. Click Run



The screenshot shows the Vitis IDE interface in the Debug perspective. The toolbar at the top has a green play button icon highlighted with a red circle. The left pane displays the debugger stack trace for 'sensor.elf' with a suspended thread at line 29. The right pane shows the source code for 'helloworld.c' with syntax highlighting for C code. The code includes standard library includes and a main() function that initializes SPI parameters and defines transfer structures.

```
#include <stdint.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <getopt.h>
#include <fcntl.h>
#include <time.h>
#include <sys/ioctl.h>
#include <linux/ioctl.h>
#include <sys/stat.h>
#include <linux/types.h>
#include <linux/spi/spidev.h>
#include <stdio.h>

#define ARRAY_SIZE(array) sizeof(array)/sizeof(array[0])

static const char *spi_device = "/dev/spidev0.0";

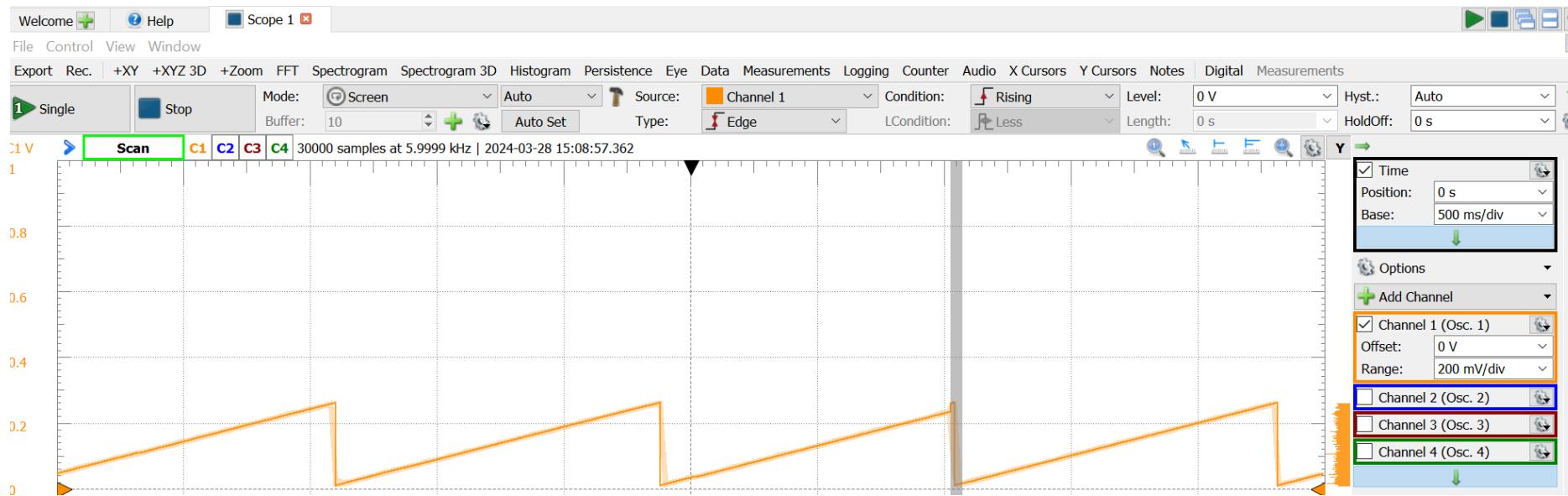
int main()
{
    static uint8_t bits = 32;
    static uint32_t speed = 2000000;
    static uint32_t mode = SPI_MODE_0 ;
    static uint16_t delay = 1;

    int spi;
    int ret = 0;
    uint32_t spi_tx_buf[1];
    uint32_t spi_rx_buf[1];

    struct spi_ioc_transfer tr = {
        .tx_buf = (uint32_t) spi_tx_buf,
        .rx_buf = (uint32_t) spi_rx_buf,
```

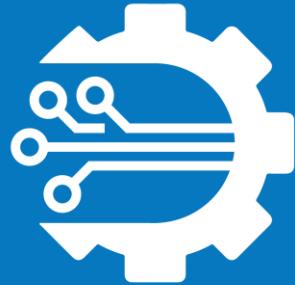
Lab: Embedded Linux

1. Observe the output in the scope





AMD sponsored this workshop, including engineering hours. AMD, and the AMD Arrow logo, MicroBlaze, Vitis, Vivado, Zynq, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.



ADIUVO

ENGINEERING AND TRAINING, LTD.

www.adiuvoengineering.com



info@adiuvoengineering.com