



Zynq 7000 Soc & Vitis software Introduction

Course Agenda
2023

Agenda

- ▶ **Zynq 7000 Overview**
- ▶ **Processing System (PS) Components**
- ▶ **PS – PL Interfaces**
- ▶ **Zynq Boot and Configuration**

- ▶ **Embedded System Design Flow**
- ▶ **Building and Debugging Linux Applications**
- ▶ **Design MicroBlaze via Vitis**
- ▶ **Using the GP Port in Zynq Devices**



Zynq 7000 Overview

Course Agenda
2023

Zynq-7000 Family Highlights

- ▶ Complete ARM®-based processing system
 - Application Processor Unit (APU)
 - - Dual ARM Cortex™-A9 processors
 - Caches and support blocks
 - Fully integrated memory controllers
 - I/O peripherals
- ▶ Tightly integrated programmable logic
 - Used to extend the processing system
 - Scalable density and performance
- ▶ Flexible array of I/O
 - Wide range of external multi-standard I/O
 - High-performance integrated serial transceivers
 - Analog-to-digital converter inputs

PS and PL

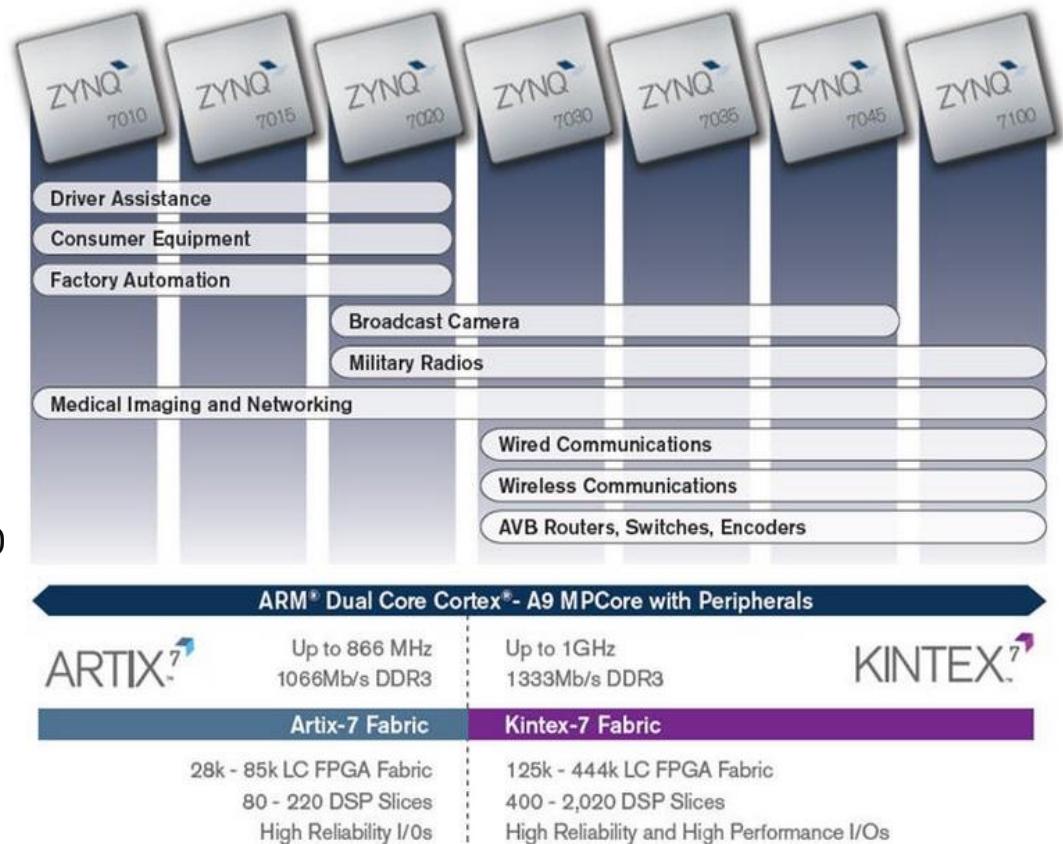
- ▶ The Zynq-7000 AP SoC architecture consists of two major sections

- PS: Processing system

- Dual ARM Cortex-A9 processor based
- Multiple peripherals
- Hard silicon core

- PL: Programmable logic

- Uses the same 7 series programmable logic
 - Artix™-based devices: Z-7010, Z-7015 and Z-7020
(high-range I/O banks only)
 - Kintex™-based devices: Z-7030, Z-7035, Z-7045, and Z-7100
(mix of high-range and high-performance I/O banks)



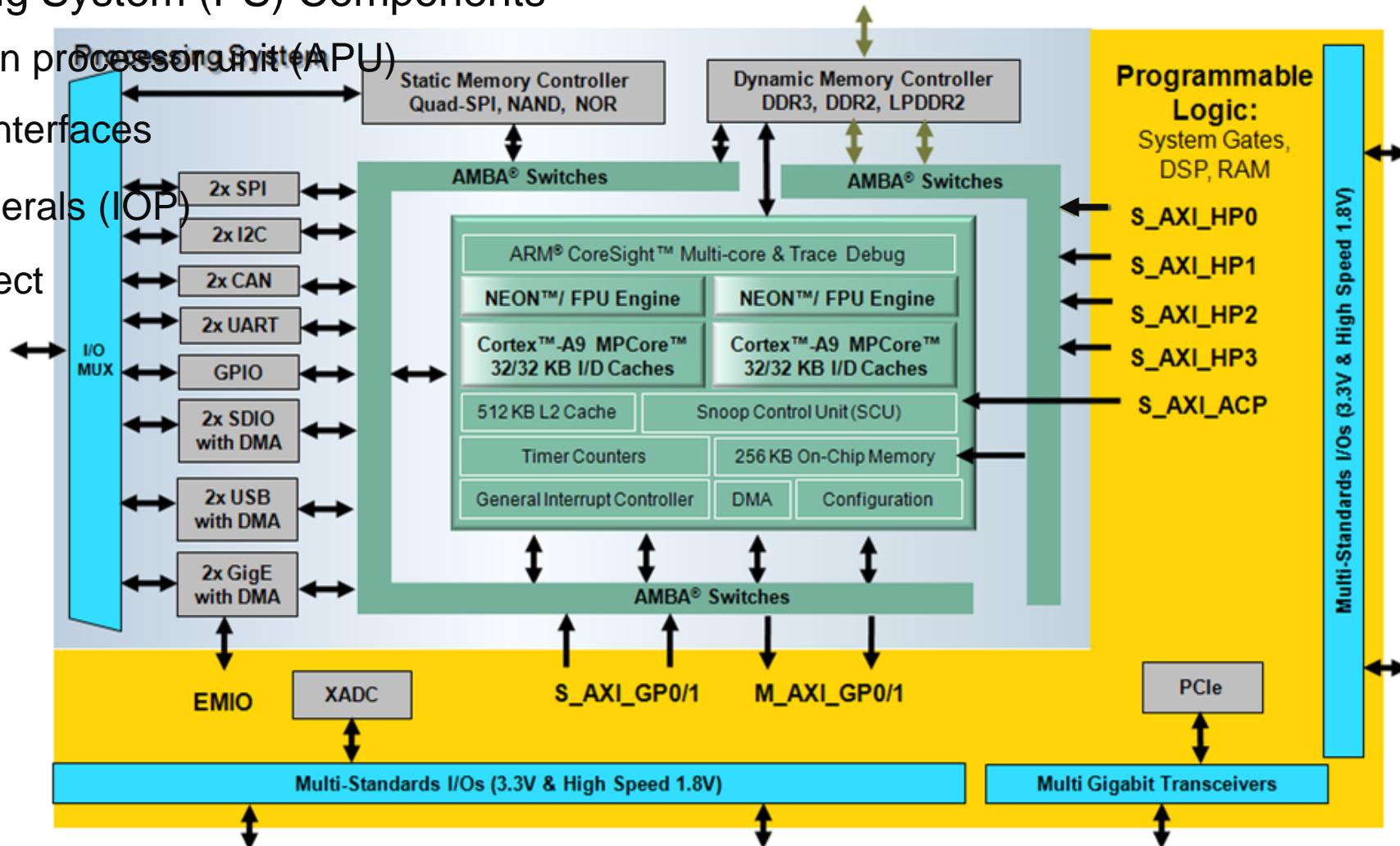


Processing System (PS) Components

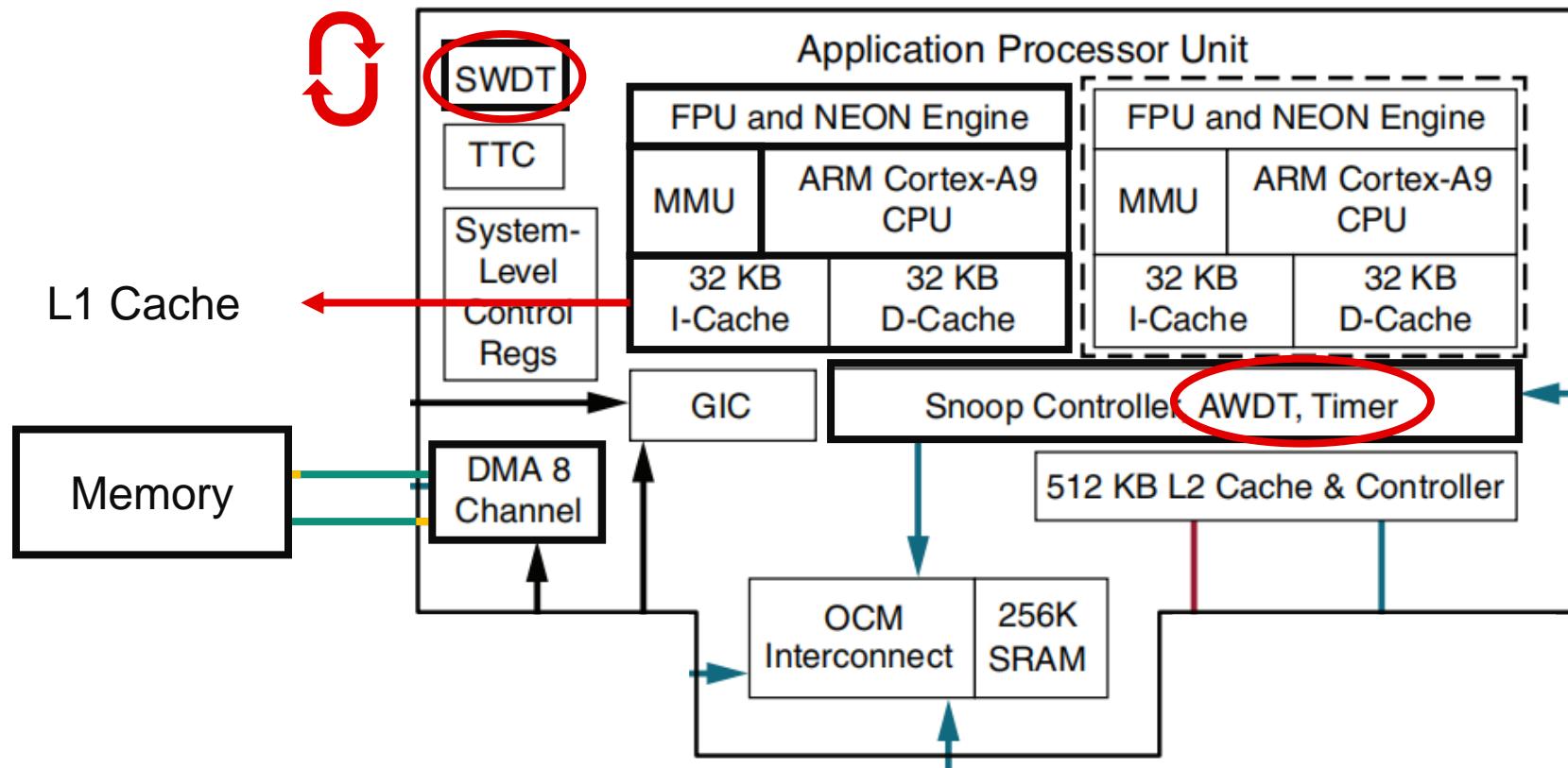
Course Agenda
2023

Zynq-7000 AP SoC Block Diagram

- Processing System (PS) Components
 - Application processor unit (APU)
 - Memory interfaces
 - I/O peripherals (IOP)
 - Interconnect

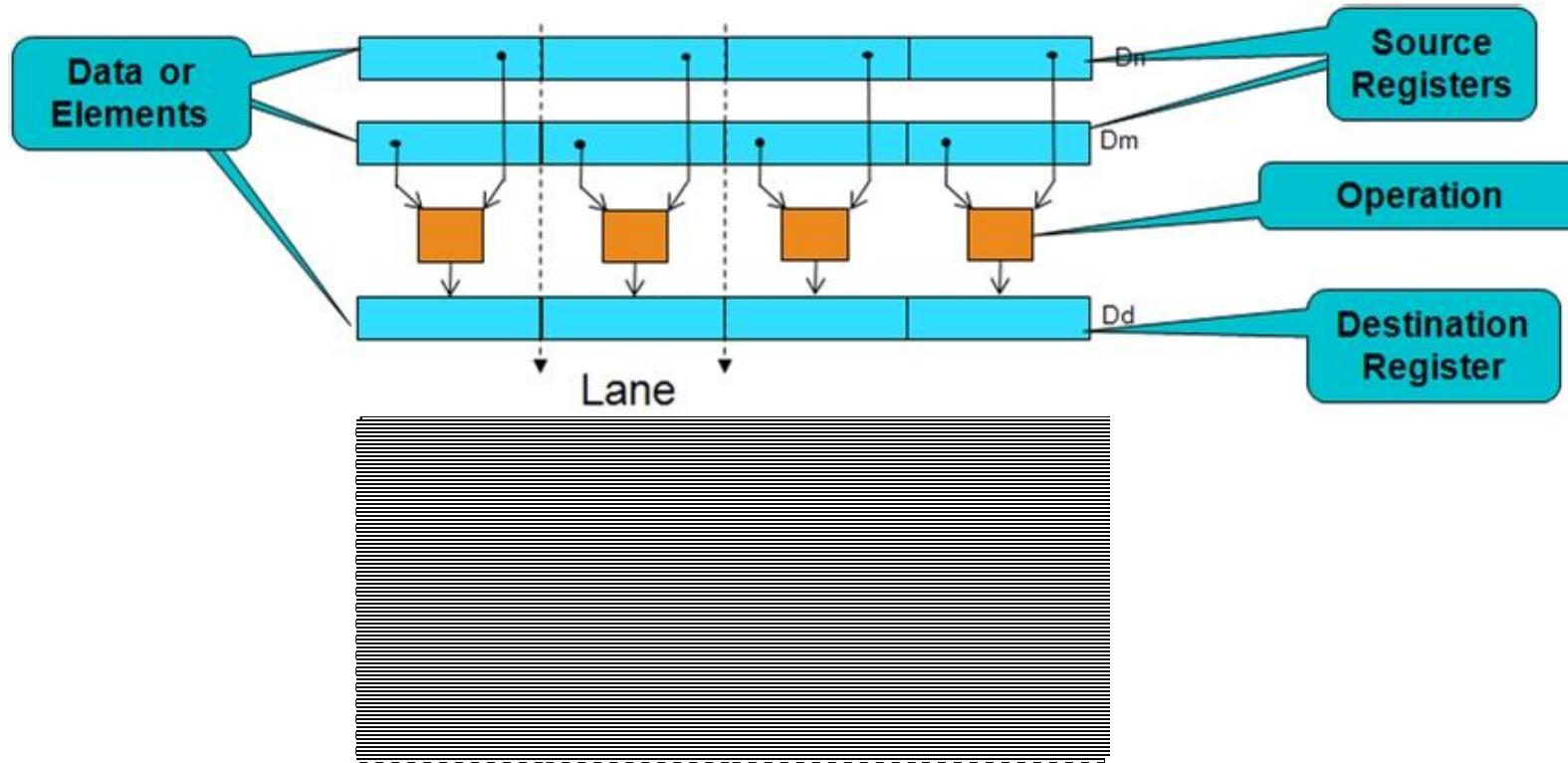


Application processor unit (APU)

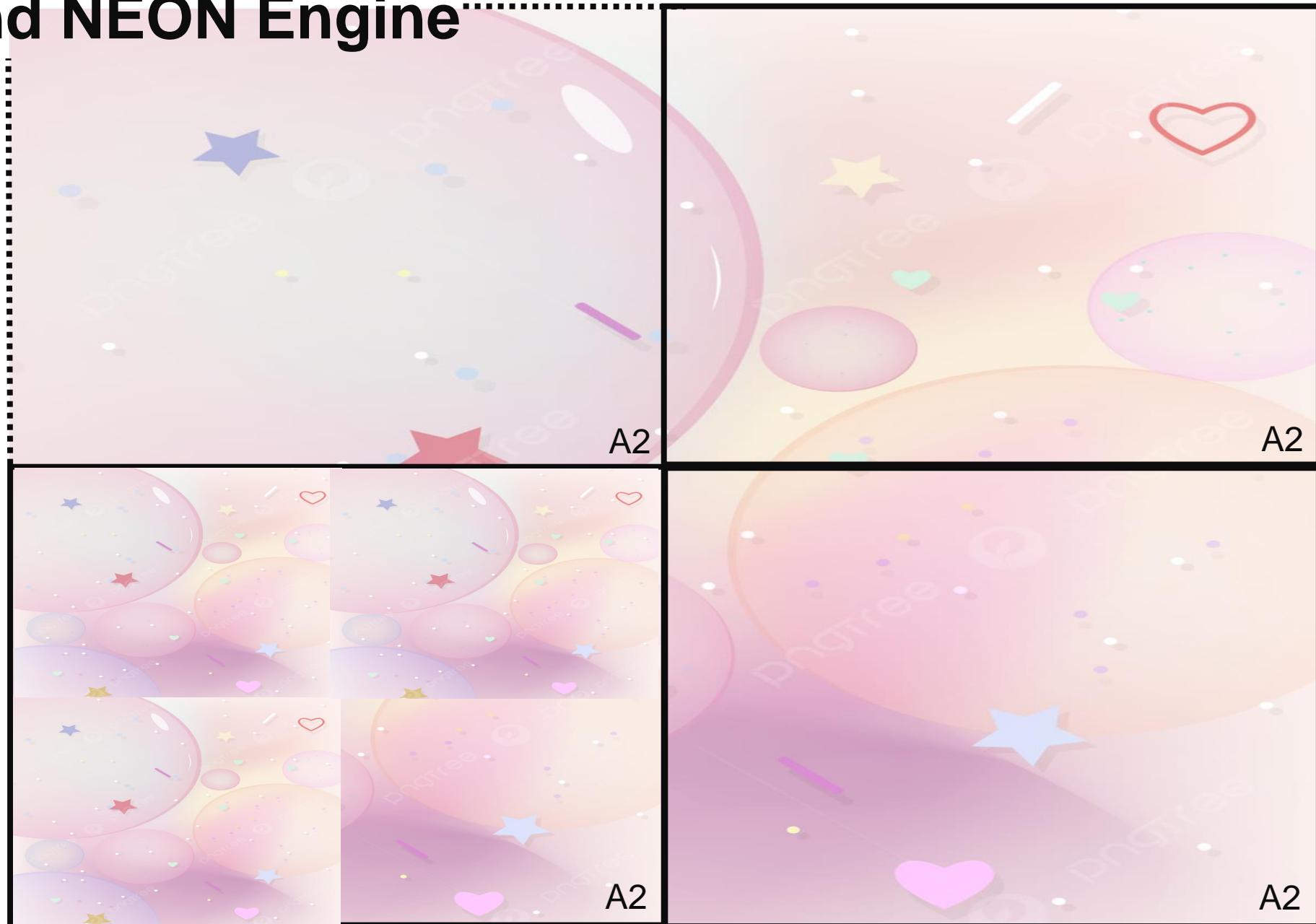


FPU and NEON Engine

- Single Instruction Multiple Data (SIMD)



FPU and NEON Engine



FPU and NEON Engine

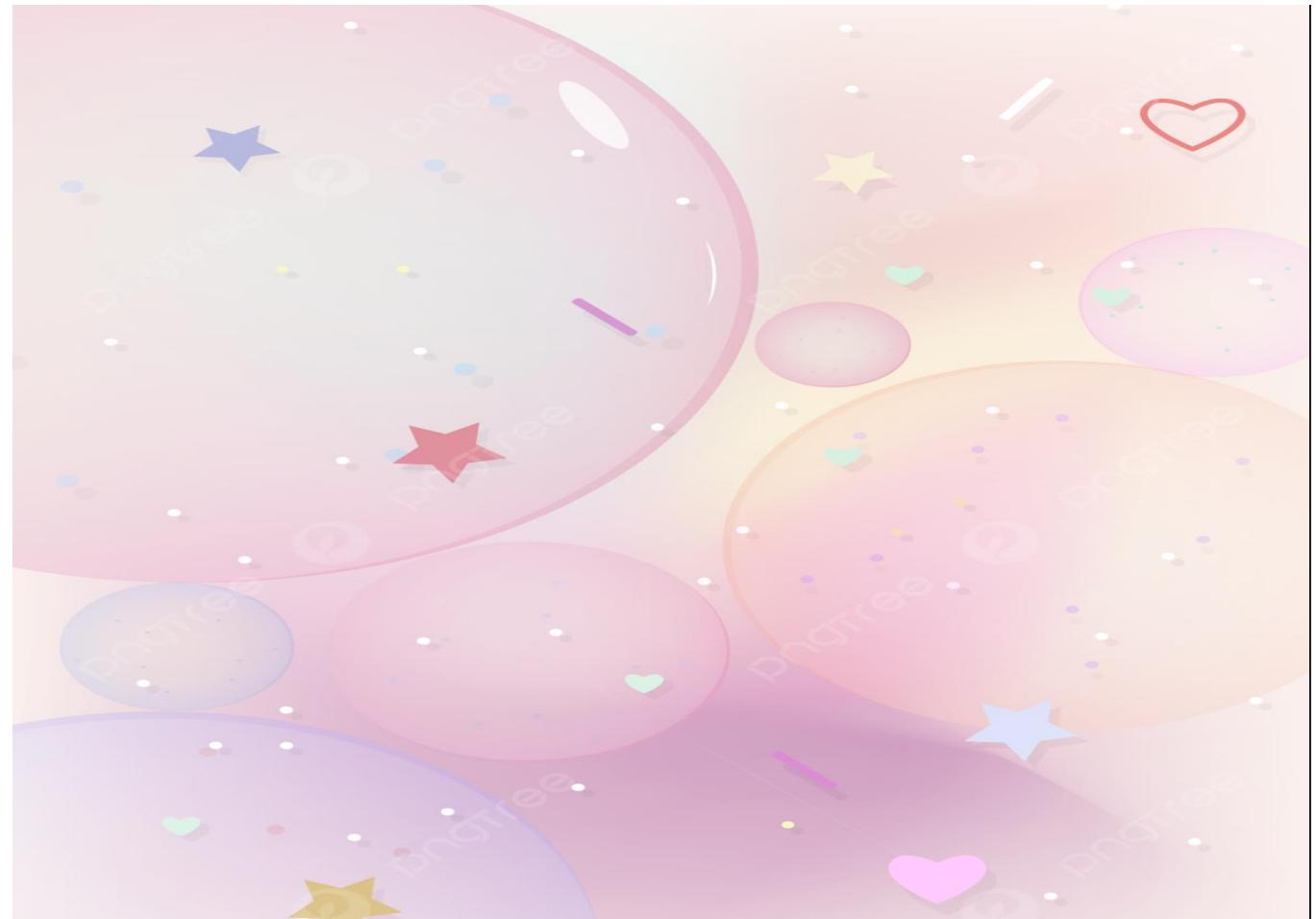
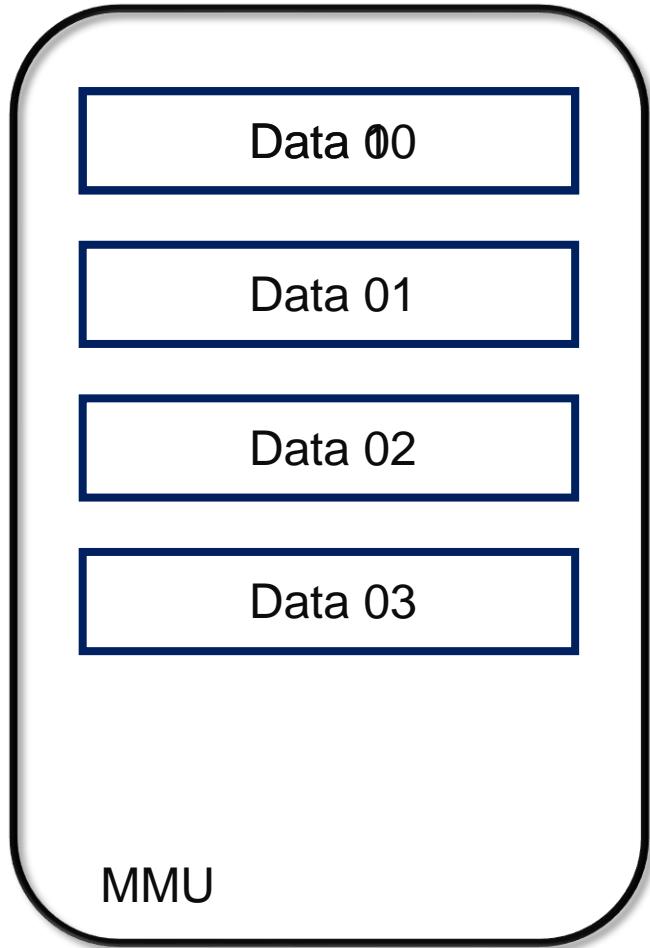
NEON technology is a wide single instruction,
multiple data (SIMD) parallel and co-processing architecture

- 32 registers, 64-bits wide (dual view as 16 registers, 128-bits wide)
- Data types can be: signed/unsigned 8-bit, 16-bit, 32-bit, 64-bit, or 32-bit float

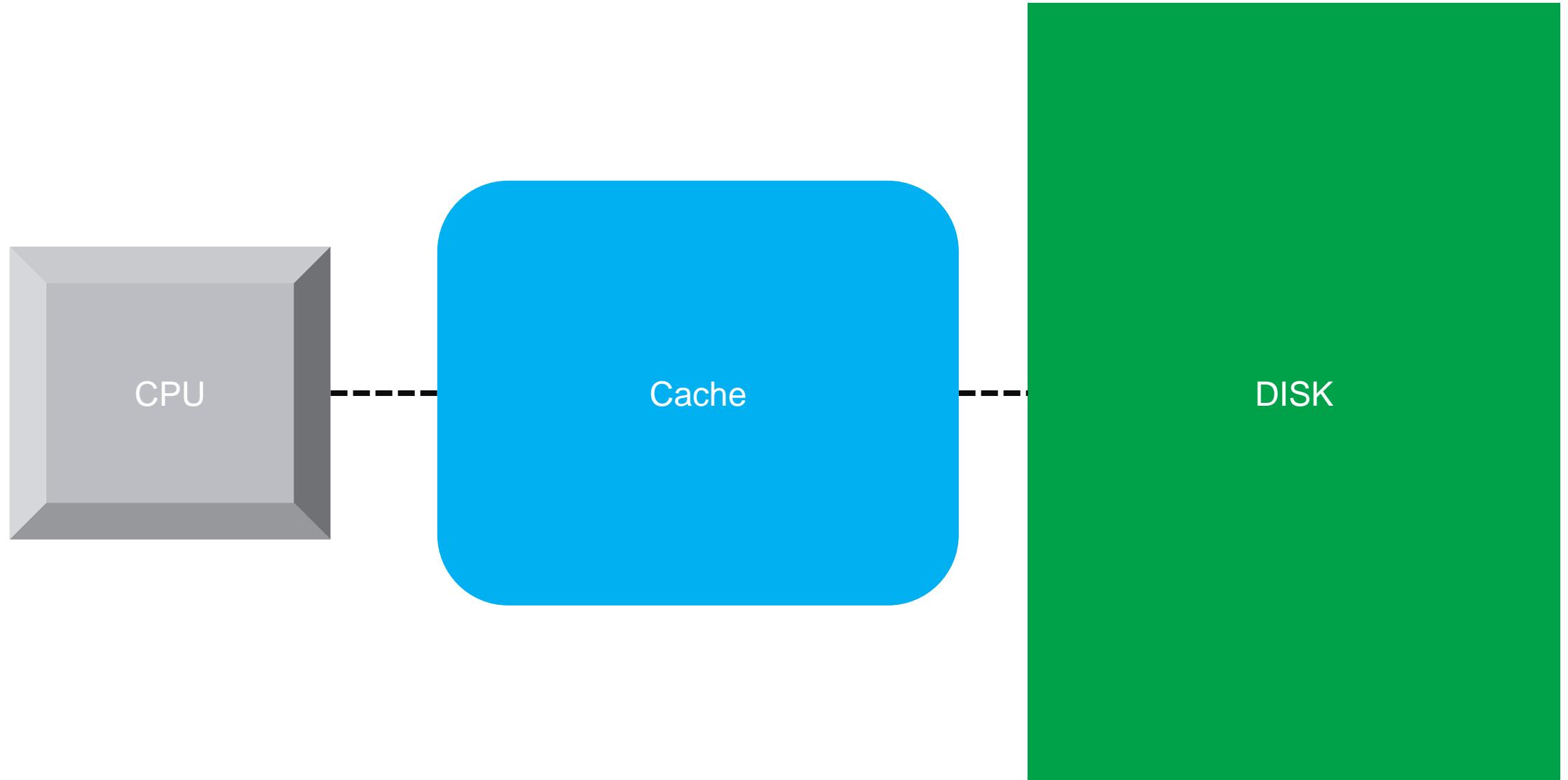


X 32

Memory Management Unit



What is Cache?



Cache Hit rate



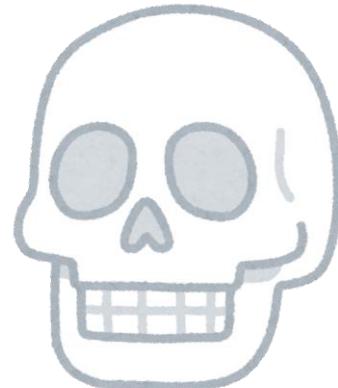
25%



70%



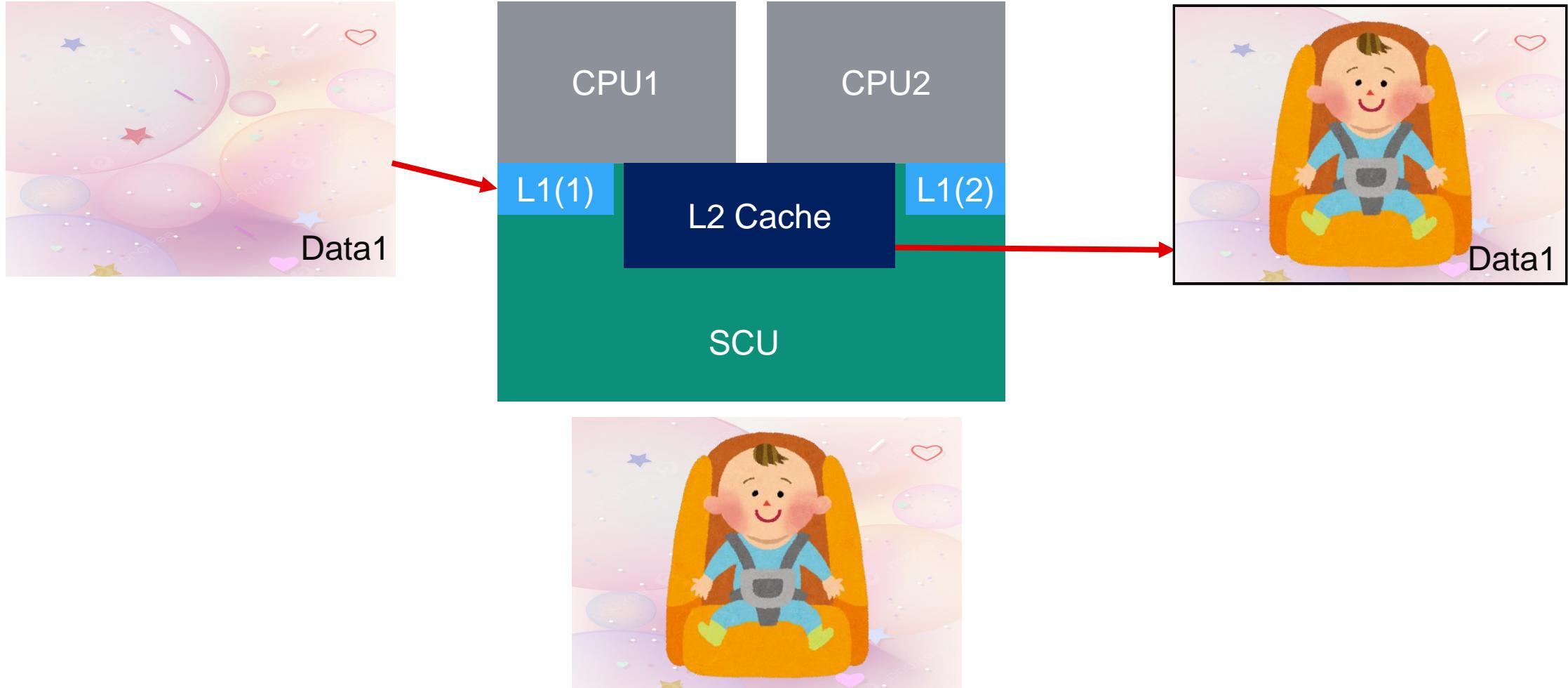
5%



?%

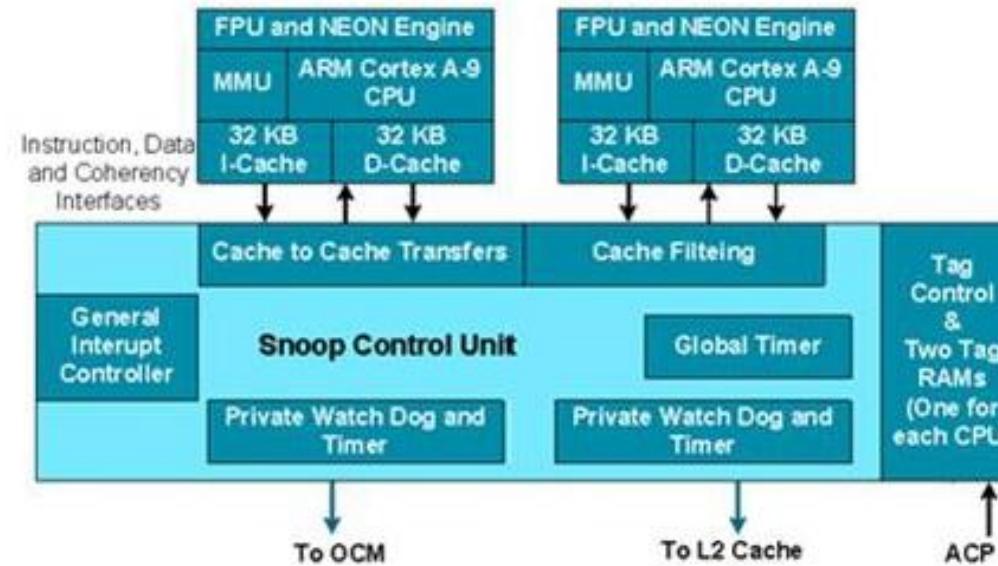


Snoop Control Unit (SCU)



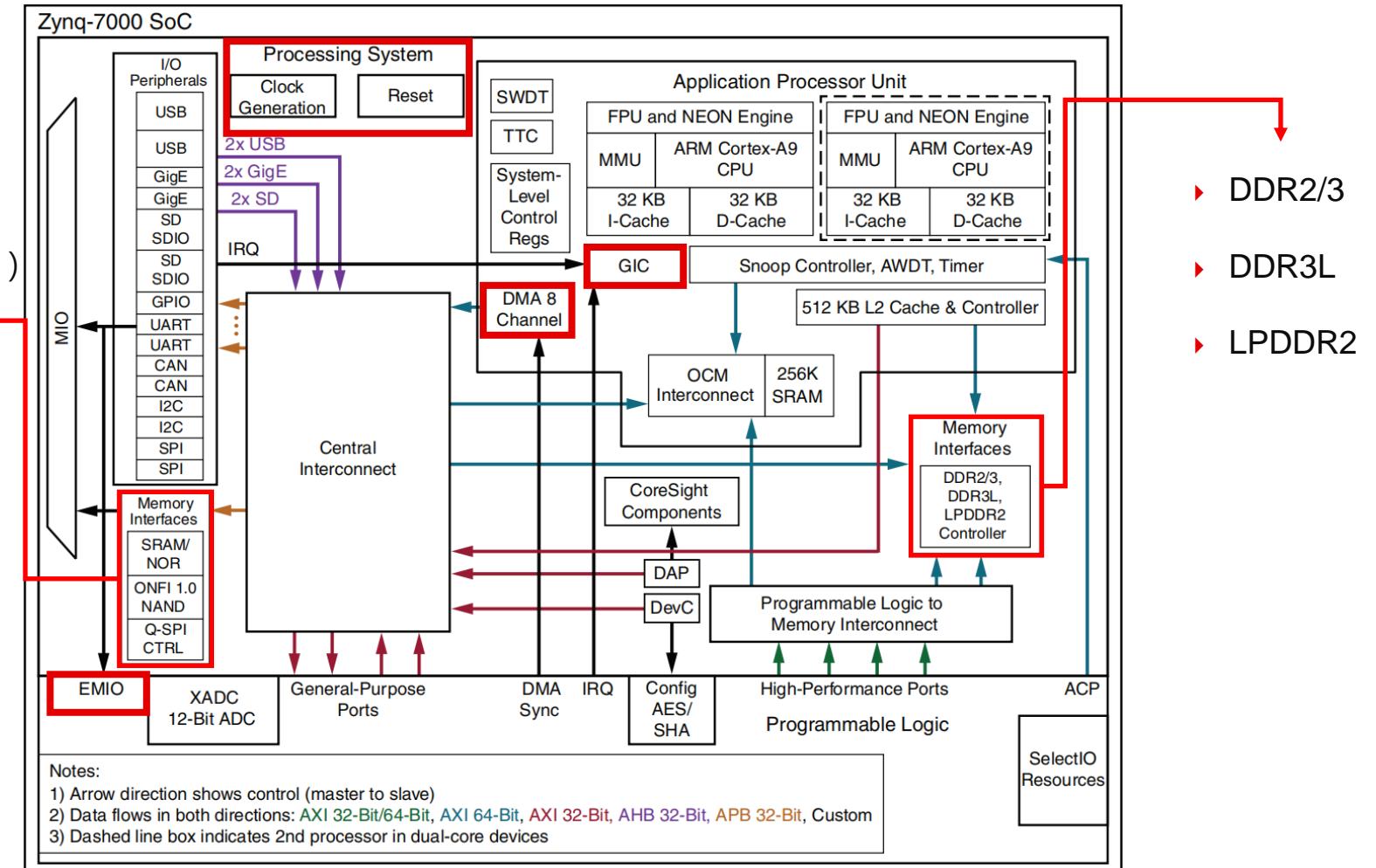
Snoop Control Unit (SCU)

- ▶ Shares and arbitrates functions between the two processor cores
 - Data cache coherency between the processors
 - Initiates L2 AXI memory access
 - Arbitrates between the processors requesting L2 accesses
 - Manages ACP accesses
 - A second master port with programmable address filtering between OCM and L2 memory support



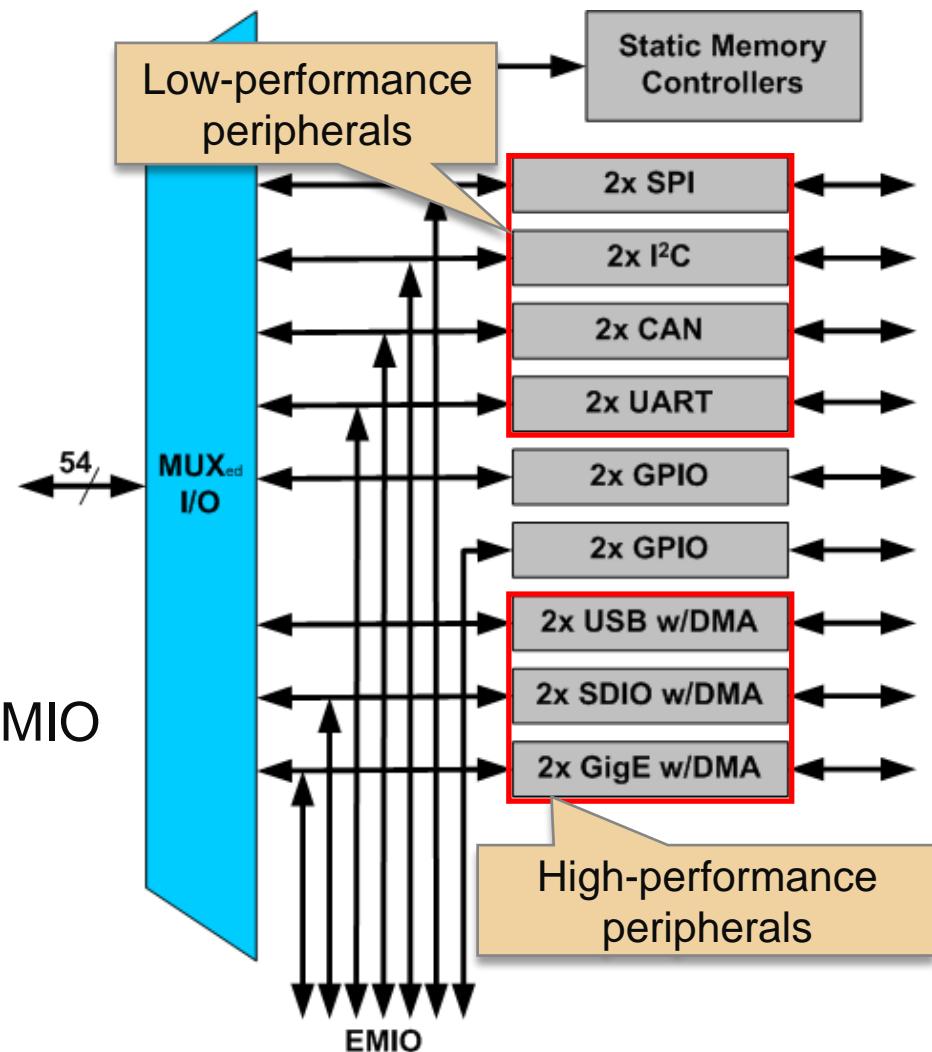
Memory interfaces

- ▶ SRAM/NOR · SRAM interface
- ▶ ONFI 1.0 NAND (Open Nand Flash Interface)
- ▶ Q-SPI CTRL (Qual SPI control · quad-channel SPI FLASH interface)



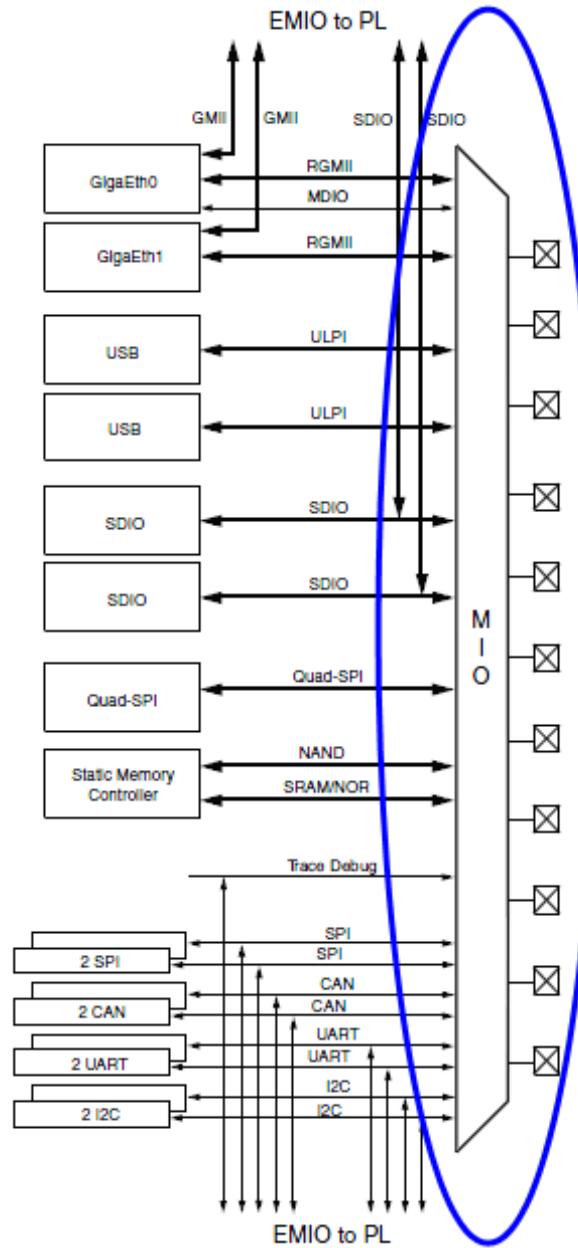
Zynq Built-in Peripherals

- ▶ Two USB 2.0 OTG/device/host
- ▶ Two tri-mode gigabit Ethernet (10/100/1000)
- ▶ Two SD/SDIO interfaces
 - Memory, I/O, and combo cards
- ▶ Two CAN 2.0Bs, SPIs, I2Cs, UARTs
- ▶ Four GPIO 32-bit blocks
 - 54 available through MIO; other 64 available through EMIO

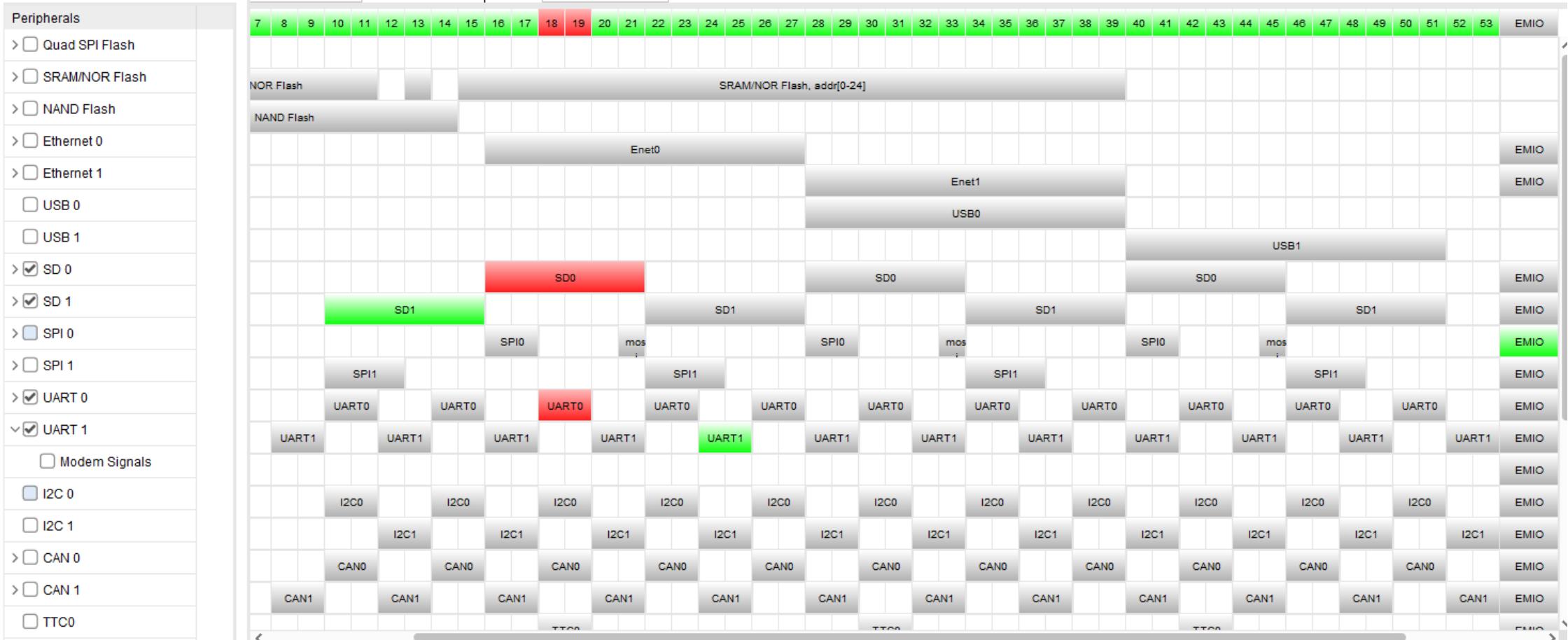


Multiplexed I/O (MIO)

- ▶ External interface to PS I/O peripheral ports
 - 54 dedicated package pins available
 - Software configurable
 - Automatically added to bootloader by tools
 - Not available for all peripheral ports
 - Some ports can only use EMIO



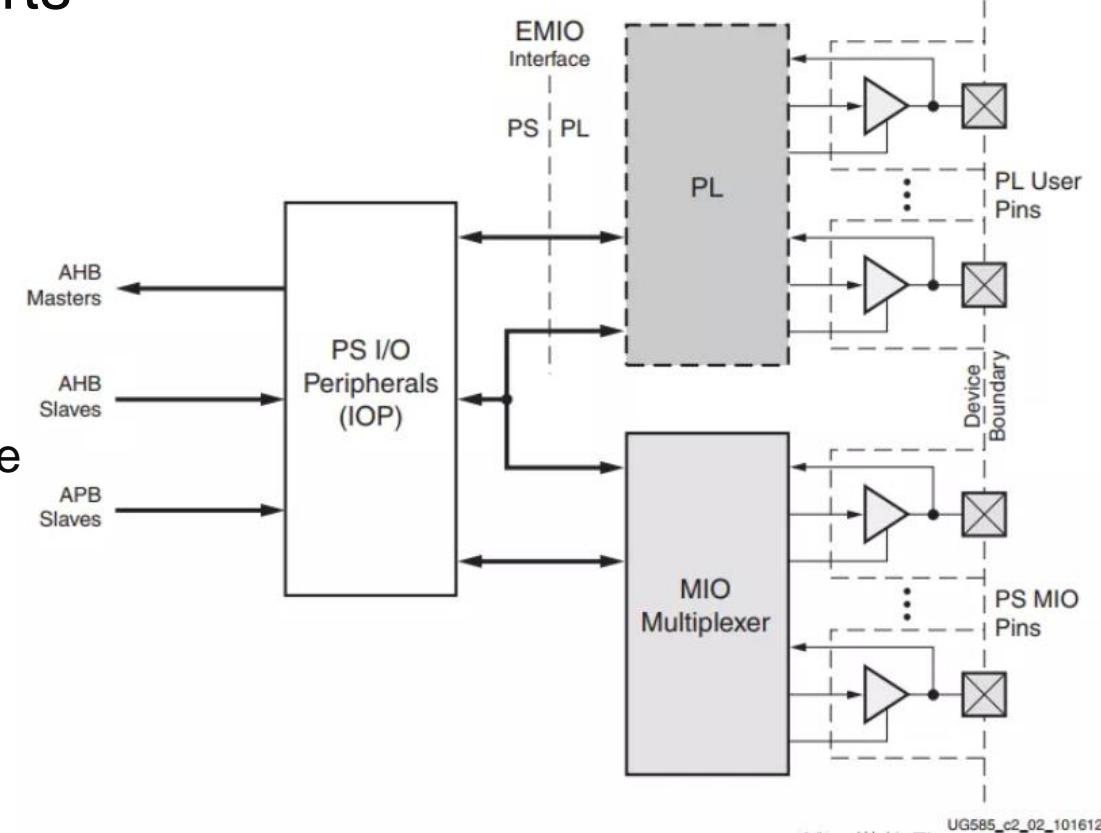
Multiplexed I/O (MIO)



Extended Multiplexed I/O (EMIO)

► Extended interface to PS I/O peripheral ports

- EMIO: Peripheral port to programmable logic
- Alternative to using MIO
- Mandatory for some peripheral ports
- Facilitates
 - Connection to peripheral in programmable logic
 - Use of general I/O pins to supplement MIO pin usage
 - Allows additional signals for many of the peripherals
 - Alleviates competition for MIO pin usage



Extended Multiplexed I/O (EMIO)

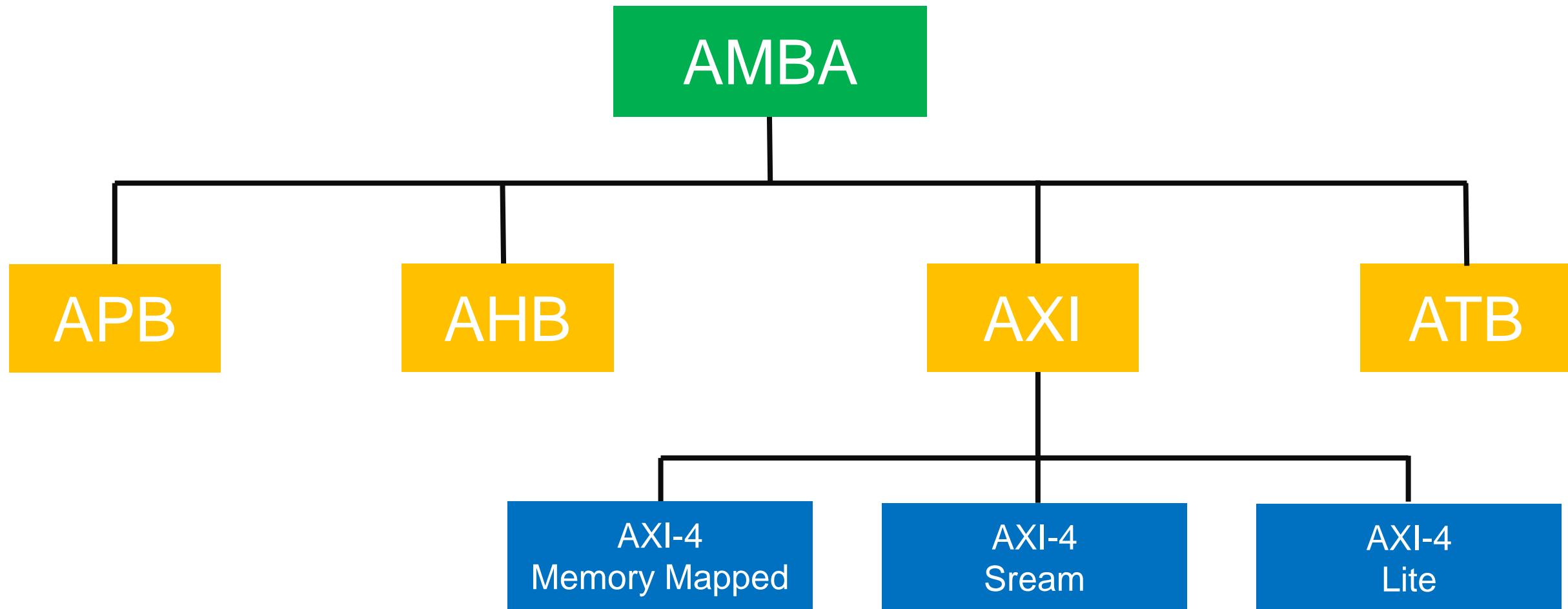
IP	MIO	Extendable MIO in Programmable Logic
QSPI NOR/SRAM NAND	Yes	No
USB:0,1	Yes, Phy off chip	No
SDIO:0,1	Yes – 50MHz	Yes – 25MHz
SPI:0,1 I2C:0,1 CAN:0,1 GPIO	Yes	Yes
GigE:0,1	RGMII v2.0 (HSTL) Phy off chip	Supports GMII, RGMII v2.0 (HSTL), RGMII v1.3 (LVCMOS), RMII, MII, SGMII with wrapper in Programmable Logic
UART:0,1	Simple UART: Only 2 pins (Tx & Rx)	Full UART (Tx, Rx, DTR, DCD, DSR, RI, RTS & CTS) either require: <ul style="list-style-type: none">• 2 Processing System pins (Rx & Tx) through MIO + 6 additional Programmable Logic pins• 8 Programmable Logic pins



PS – PL Interfaces

Course Agenda
2023

AXI-Memory Mapped, Streaming , Lite



AXI-Memory Mapped/Full



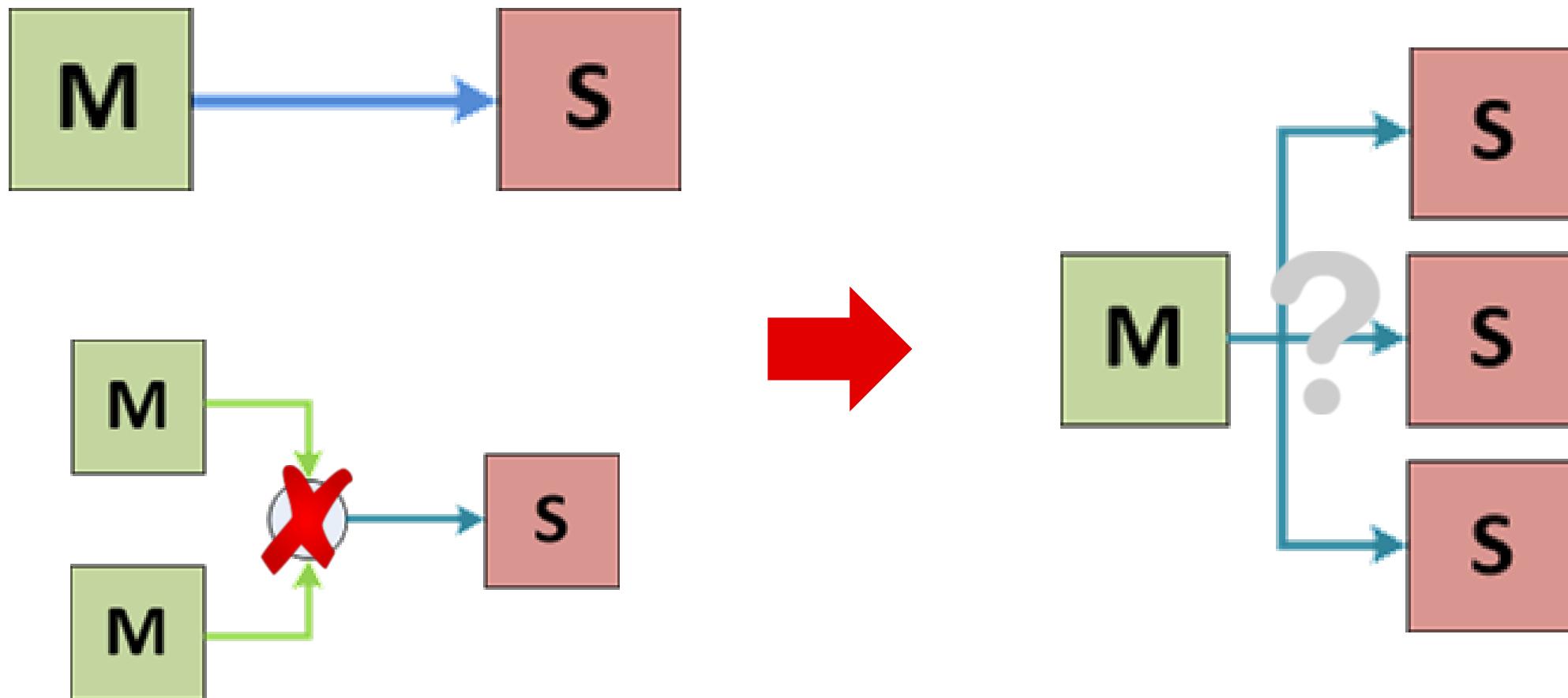
AXI-Memory Lite



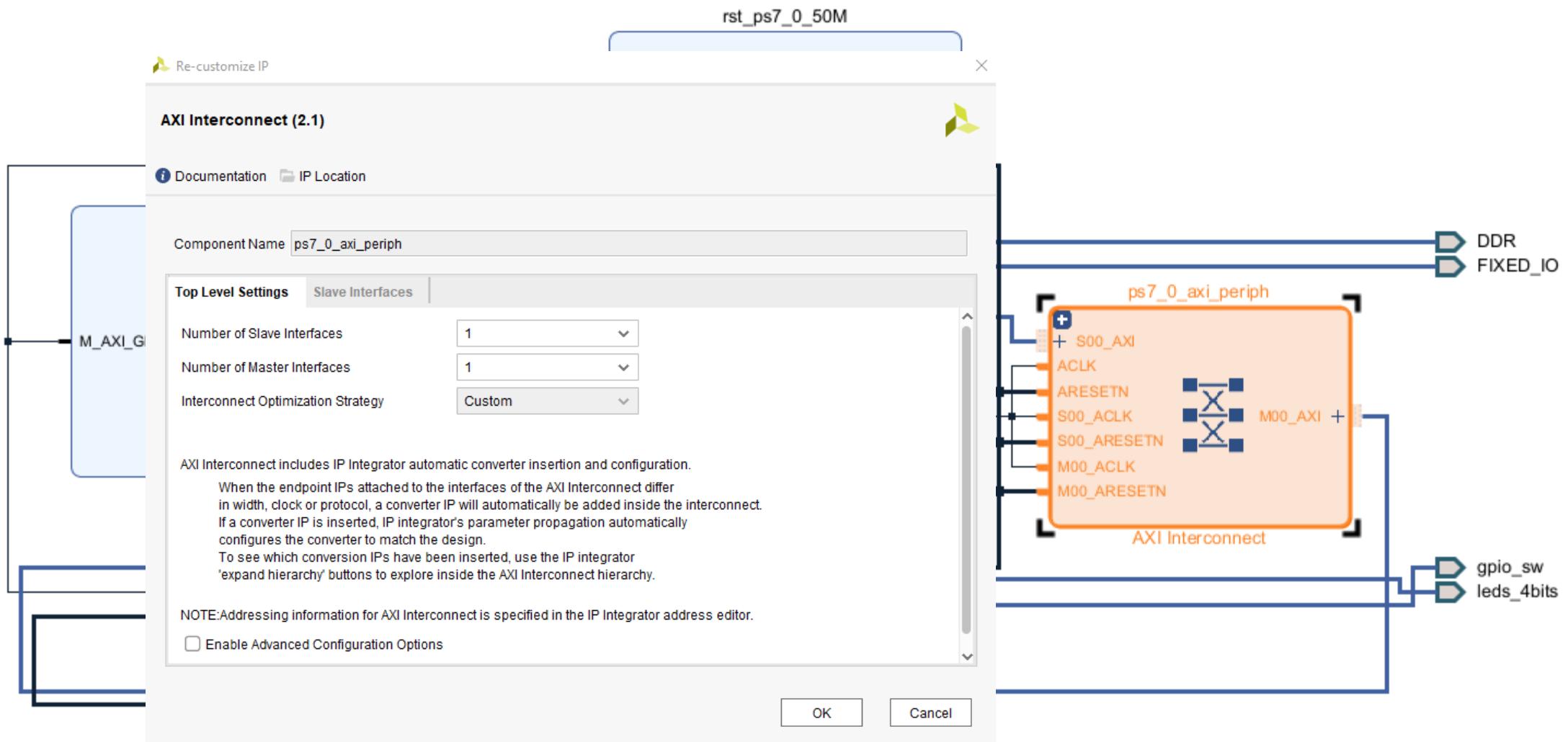
AXI-Streaming



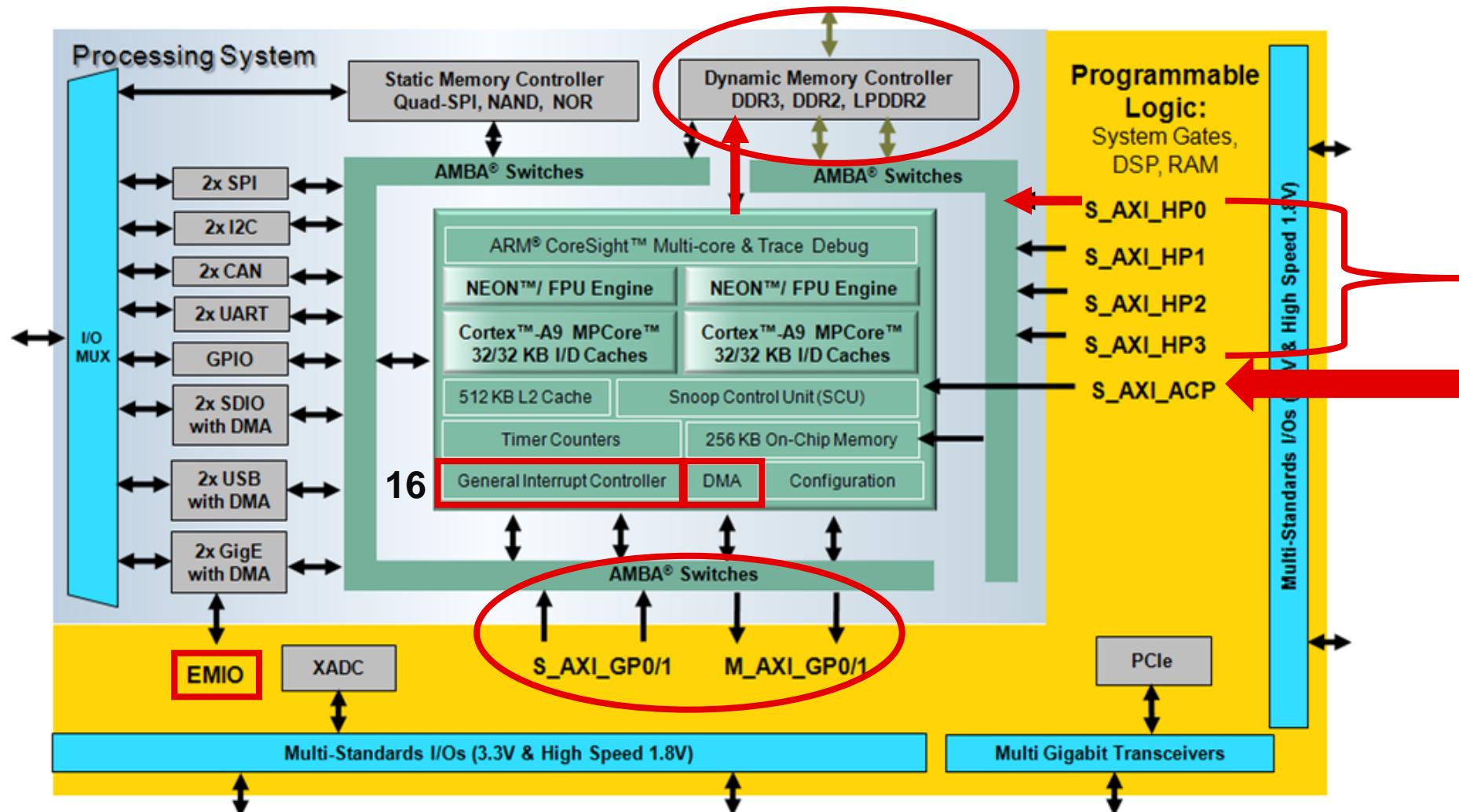
AXI-Interconnect



AXI-Interconnect



PS – PL Interconnect



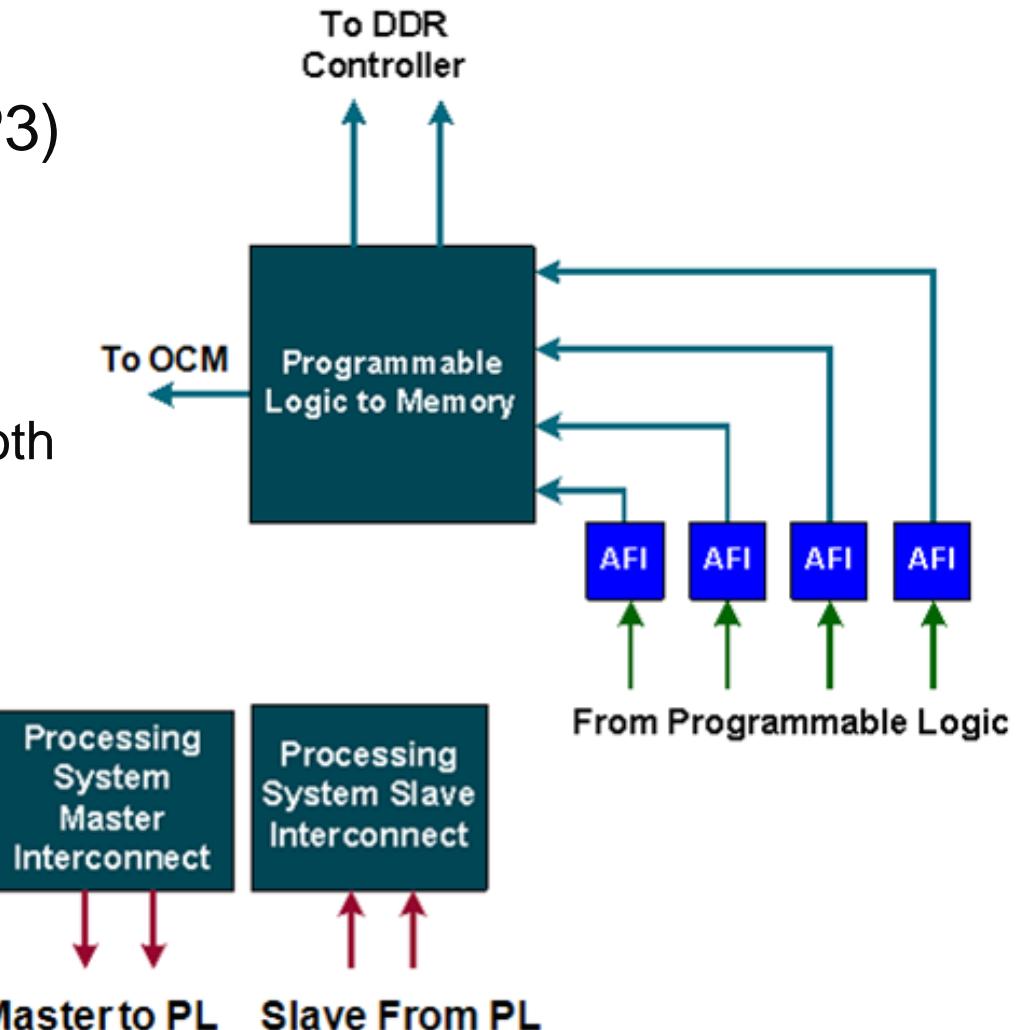
PS – PL Interconnect

▶ AXI high-performance slave ports (HP0-HP3)

- Configurable 32-bit or 64-bit data width
- Access to OCM and DDR only
- Conversion to processing system clock domain
- AXI FIFO Interface (AFI) are FIFOs (1KB) to smooth large data transfers

▶ AXI general-purpose ports (GP0-GP1)

- Two masters from PS to PL
- Two slaves from PL to PS
- 32-bit data width
- Conversation and sync to processing system clock domain



Clock and Reset

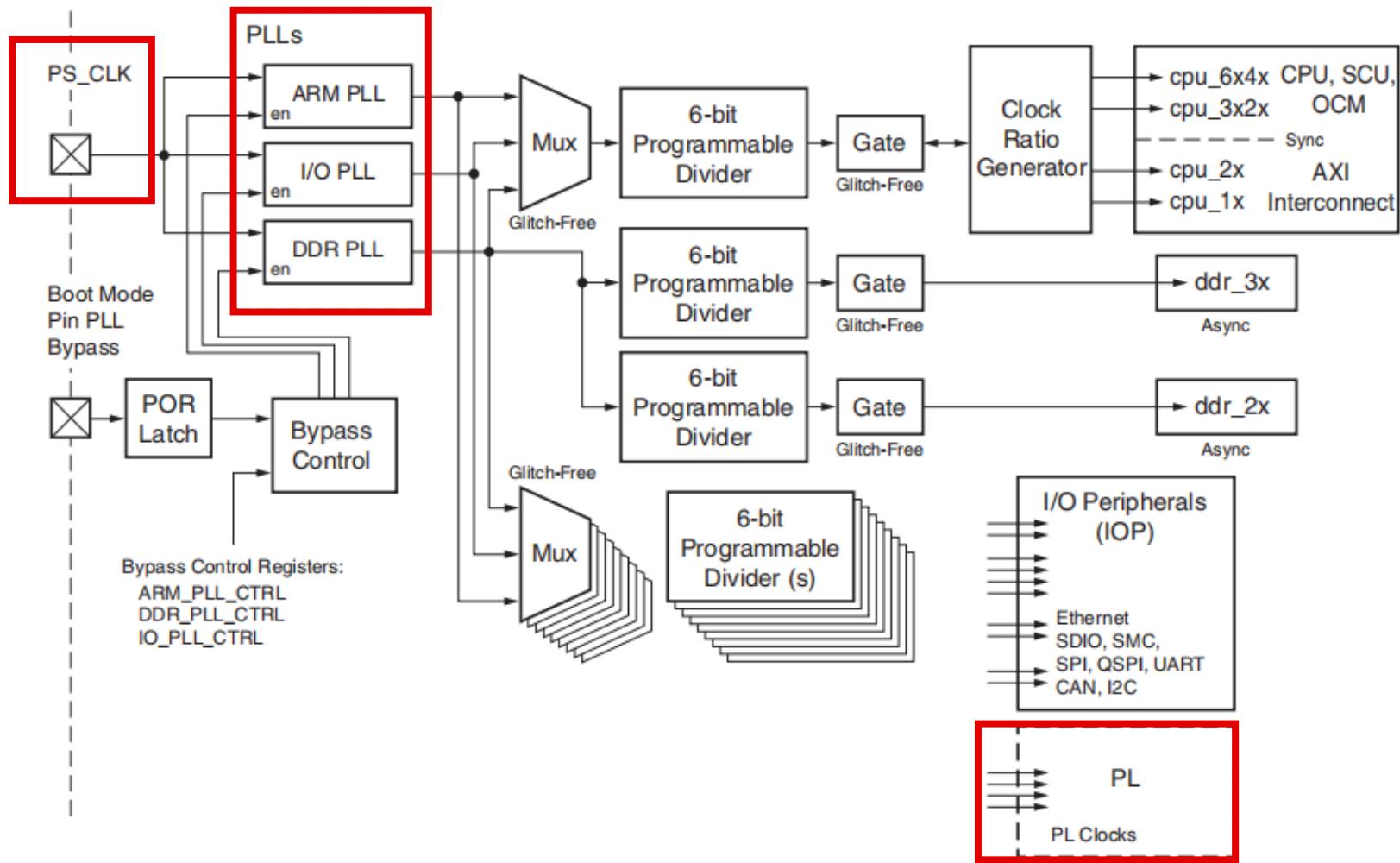


Figure 25-1: PS Clock System Block Diagram

<https://blog.csdn.net/zhoutaopower>

UG585_c25_01_102414

Clock and Reset

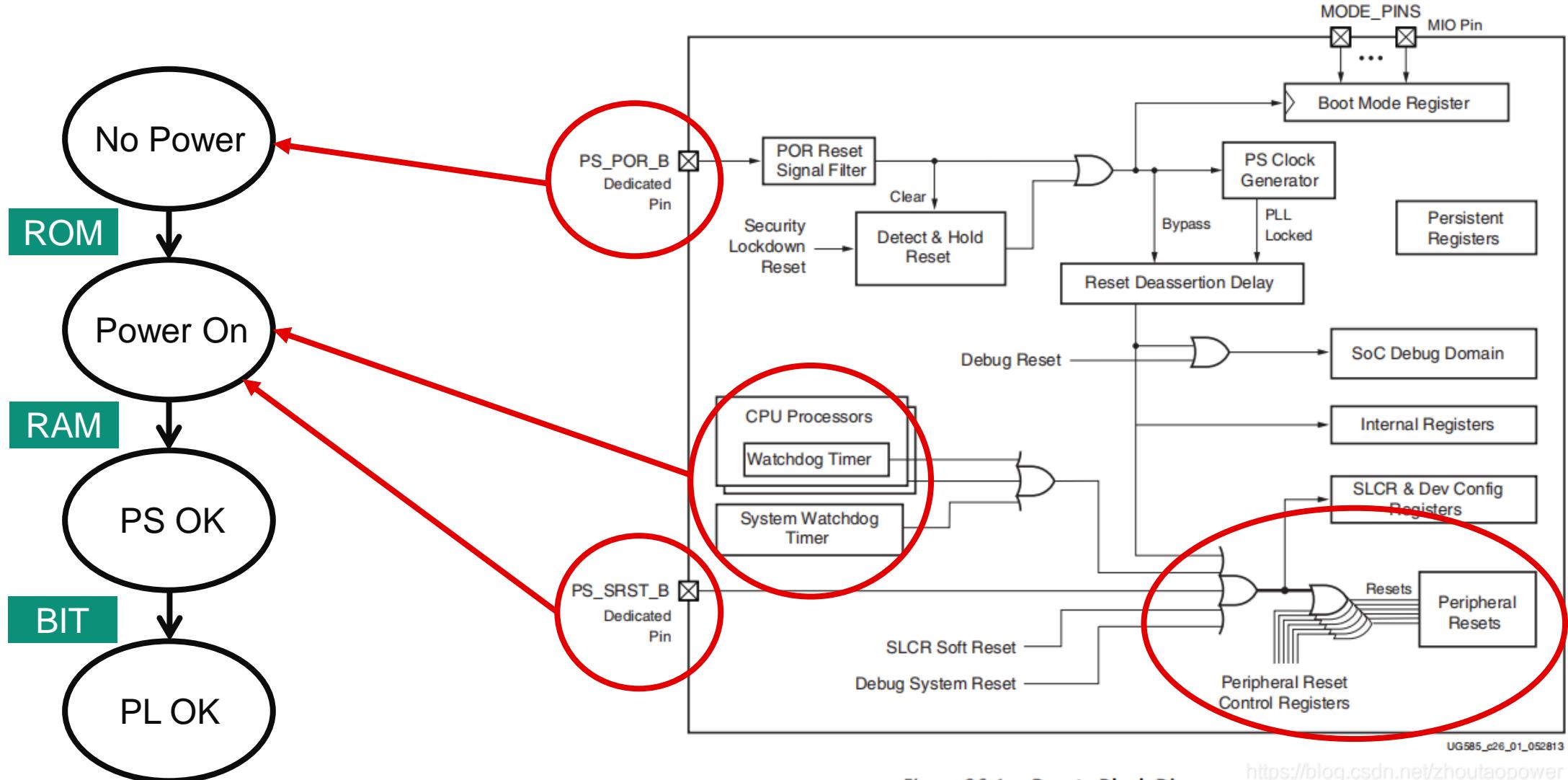


Figure 26-1: Resets Block Diagram

<https://blog.csdn.net/zhoutaopower>



Zynq Boot and Configuration

Course Agenda
2023

Zynq Boot and Configuration

- ▶ Zynq devices can be booted and/or configured in
 - Secure mode via static memories only (JTAG excluded)
 - Ability to have secure software
 - Protects bitstream and IP
 - Non-secure mode via JTAG or static memories (debug and development environment)
 - Standard boot model
- ▶ Four master boot devices

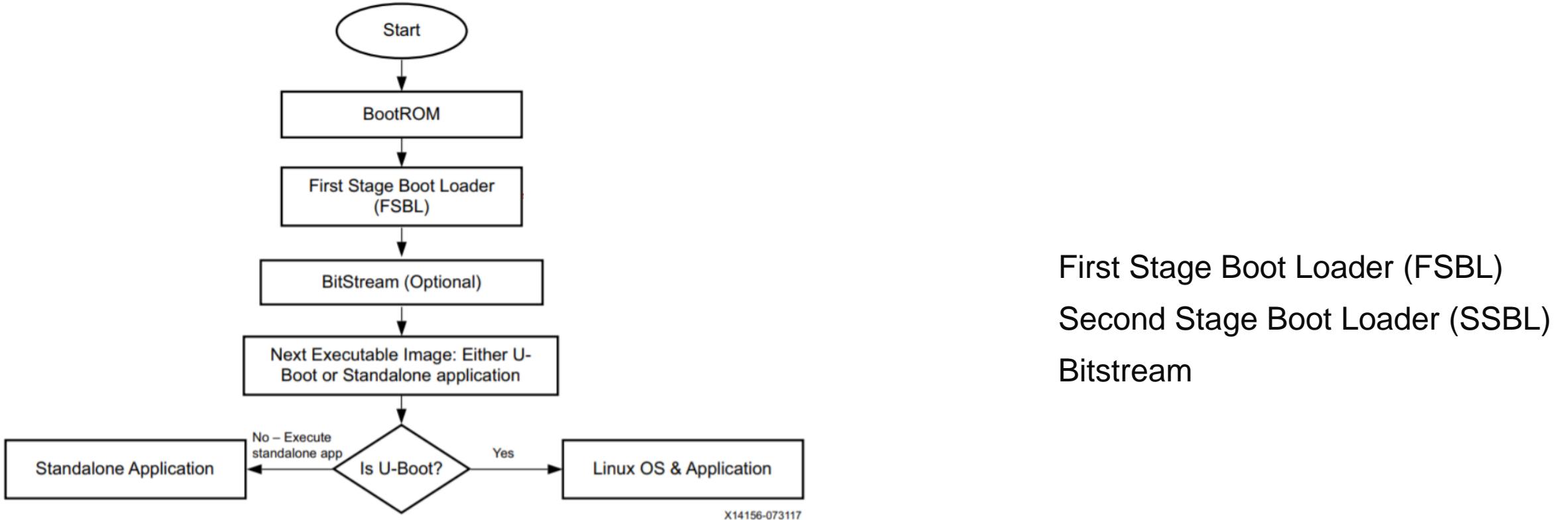
- QSPI: serial memory, linear addressing
 - NAND: complex parallel memory
 - NOR: parallel memory, linear addressing
 - SD: Flash memory card
- ▶ Secondary boot devices
 - USB, Ethernet, and most other peripherals

Standard Boot Model in Zynq AP Soc

► Multi-stage boot process

- Stage 0: Runs from ROM; loads from non-volatile memory to OCM
 - Provided by Xilinx; unmodifiable
- Stage 1: Runs from OCM; loads from non-volatile memory to DDRx memory
 - User developed; Xilinx offers example code through SDK project
 - Initiates PS boot and PL configuration
- Stage 2: Optional; runs from DDR
 - User developed; Xilinx offers example code – Uboot
 - Sourced from flash memory or through common peripherals, programmable logic I/O, etc.
- Programmable logic configuration can be performed in Stage 1 or 2

Standard Boot Model in Zynq AP Soc



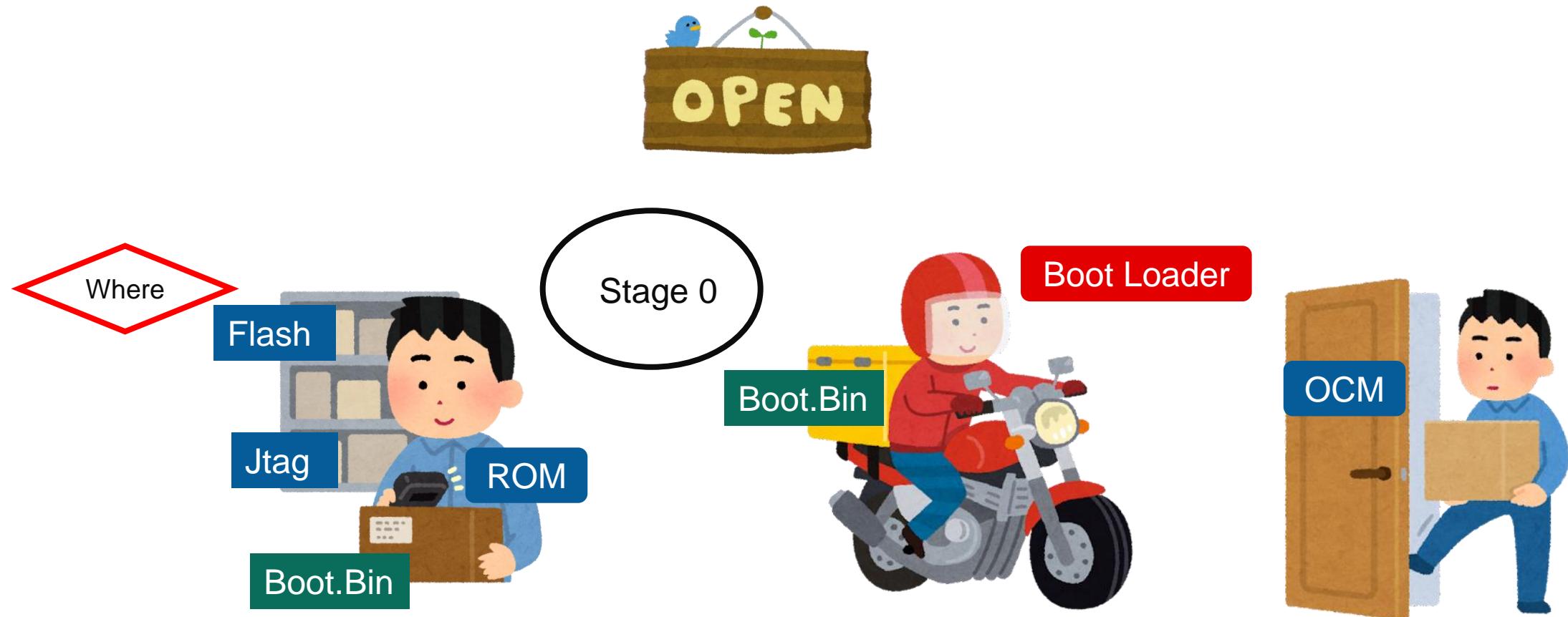
First Stage Boot Loader (FSBL)

- ▶ Example FSBL provided by Xilinx as an SDK example project
 - Otherwise user developed
- ▶ Copies next stage of code into
 - DDRx or static memory (OCM)
 - And/or enables an external device for Stage 2
- ▶ Further initialization of PS components and peripherals
- ▶ Optionally configures programmable logic
- ▶ Upon completion, launches application or Second Stage Boot Load

Second Stage Boot Loader (SSBL)

- ▶ Example U-Boot provided by Xilinx
 - <git://git.xilinx.com/u-boot-xlnx.git>
 - Otherwise user developed
- ▶ Loaded from user-selected external device
- ▶ Flexibility in boot sources
 - Static memory
 - Dynamic memory
 - PS peripherals such as
 - USB, Ethernet, or SD
 - Programmable logic I/Os
- ▶ Initializes rest of PS
- ▶ Optionally configures PL

Zynq Boot and Configuration



Zynq Boot and Configuration



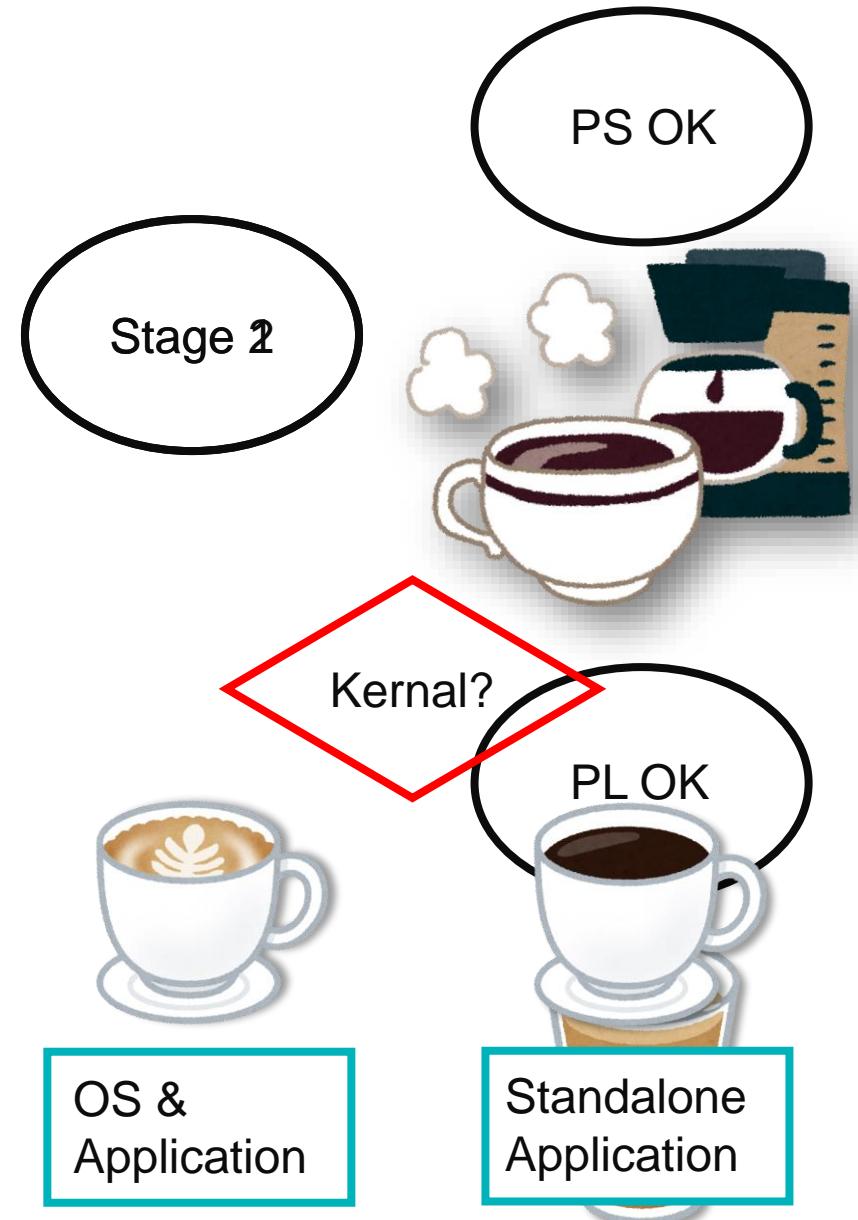
FSBL



SSBL



Bitstream

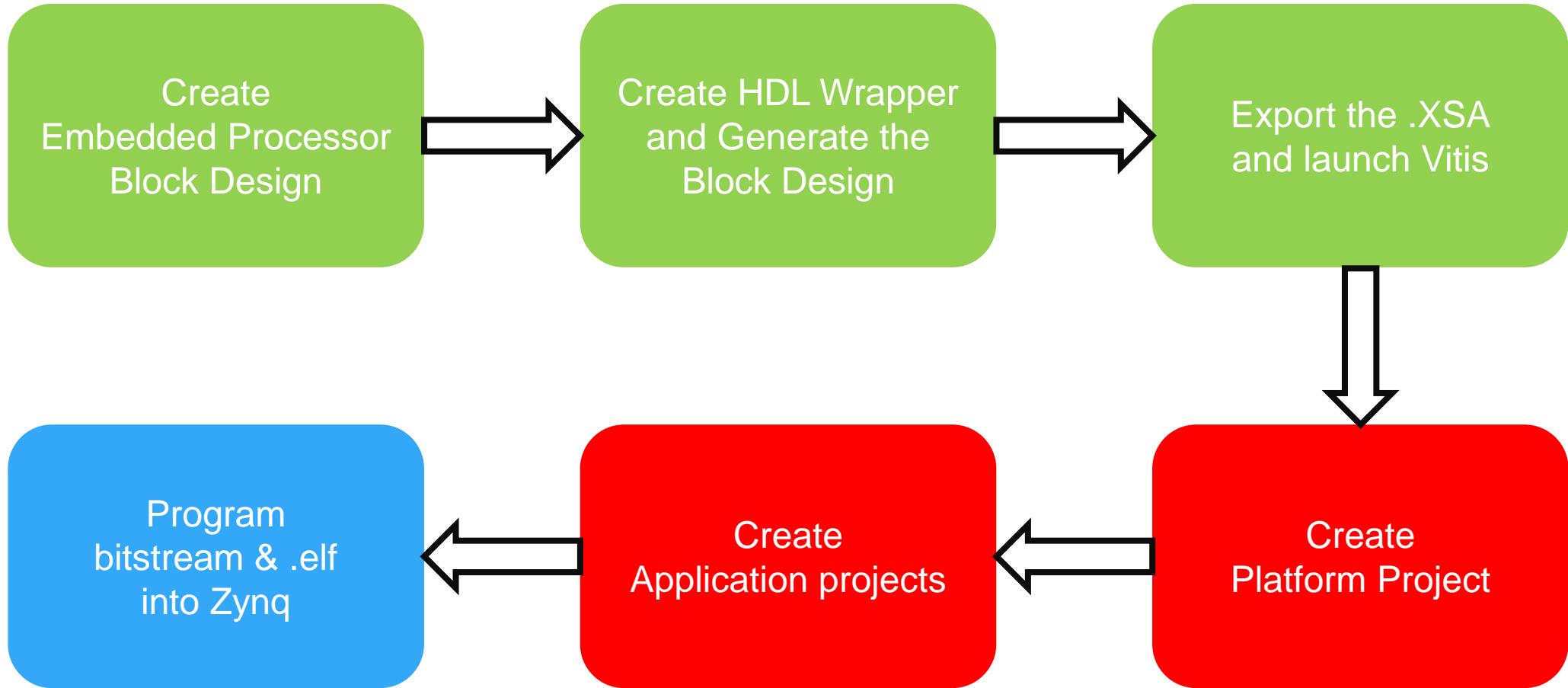




Embedded System Design Flow

Course Agenda
2023

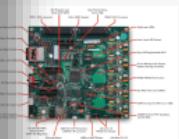
Embedded System Design Flow



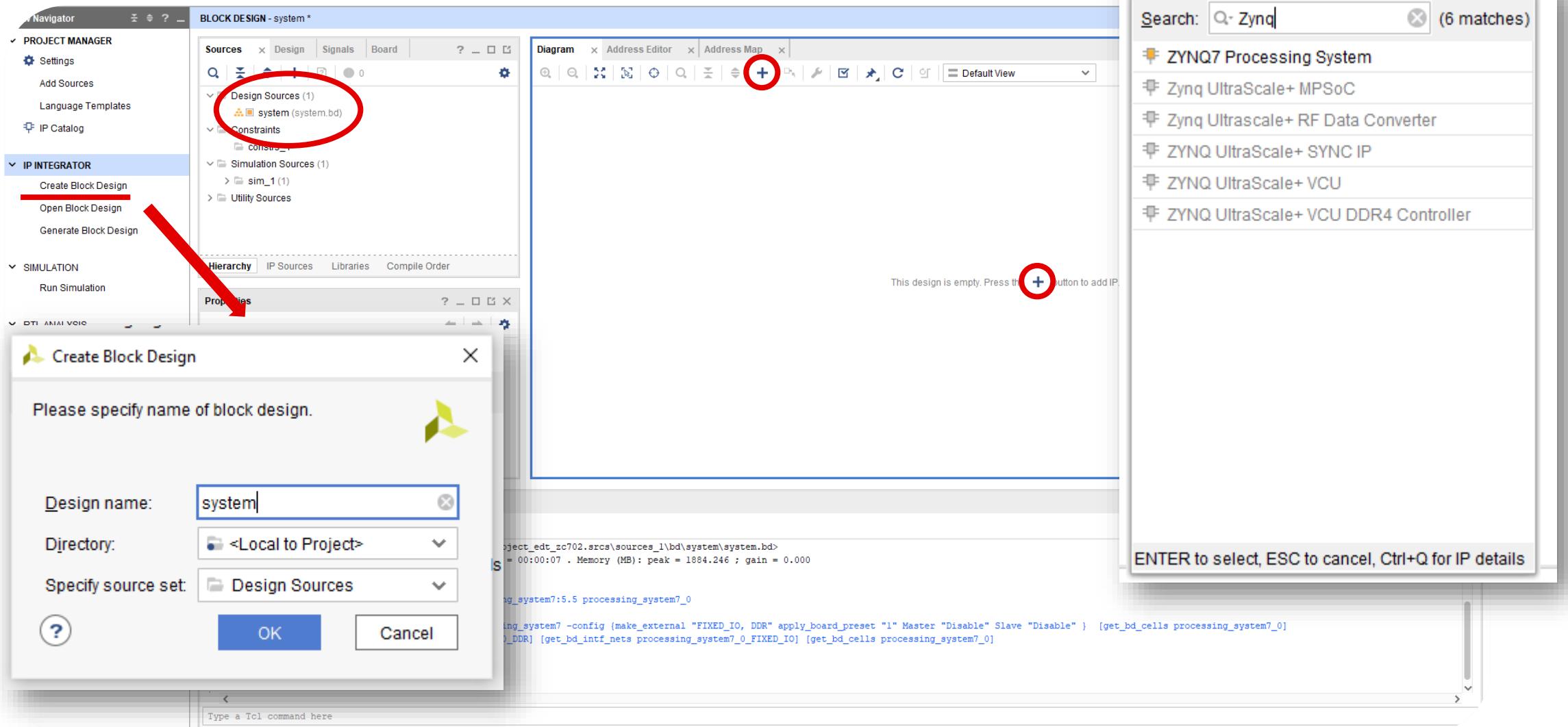
Create Project

Screen	System Property	Setting or Command to Use
Project Summary	Project Name	edt_zc702
	Project Location	C:/edt
	Create Project Subdirectory	Leave this checked.
	Project is an extensible Vitis platform	Leave this unchecked.
Project Type	Specify the type of sources for your design. You can start with RTL or a synthesized EDIF.	RTL Project
	Do not specify sources at this time	Leave this checked.
	Project is a an extensible Vitis platform	Leave this unchecked.
Default Part	Choose a default Xilinx part or board for your project	Select the Boards tab.
	Boards	ZYNQ-7 ZC702 Evaluation Board
New Project Project Summary	Project Summary	Review the project summary.

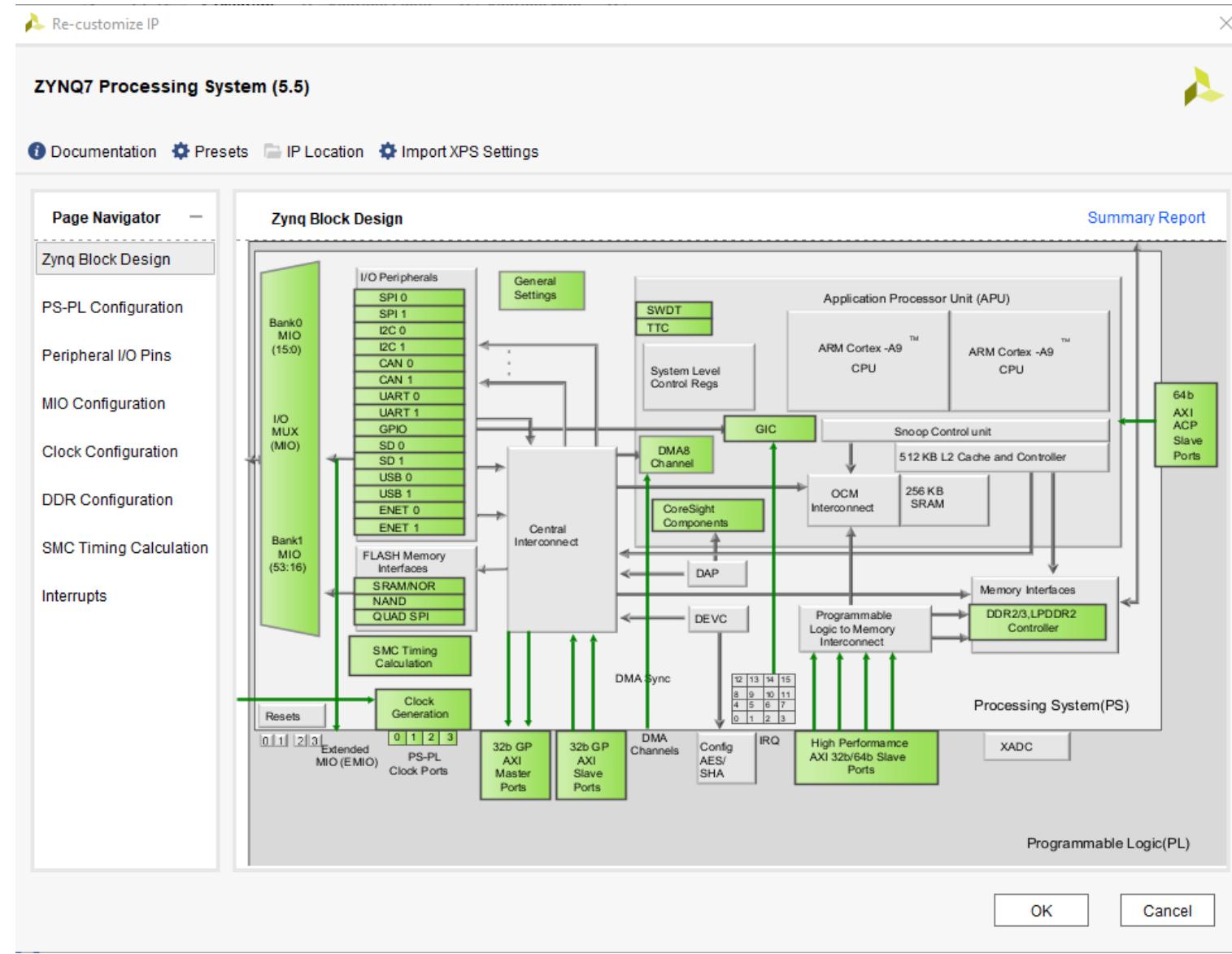
? □ □



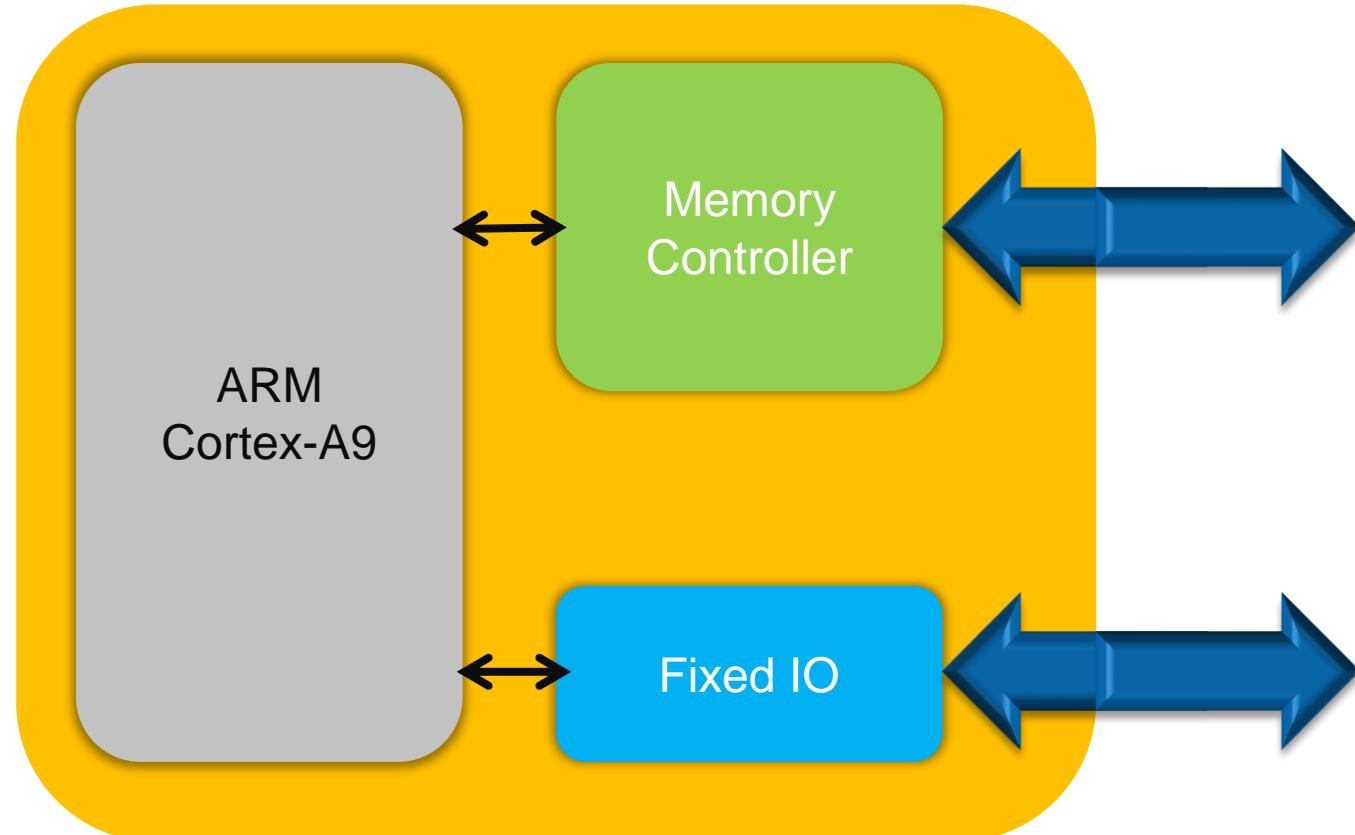
Create Block Design



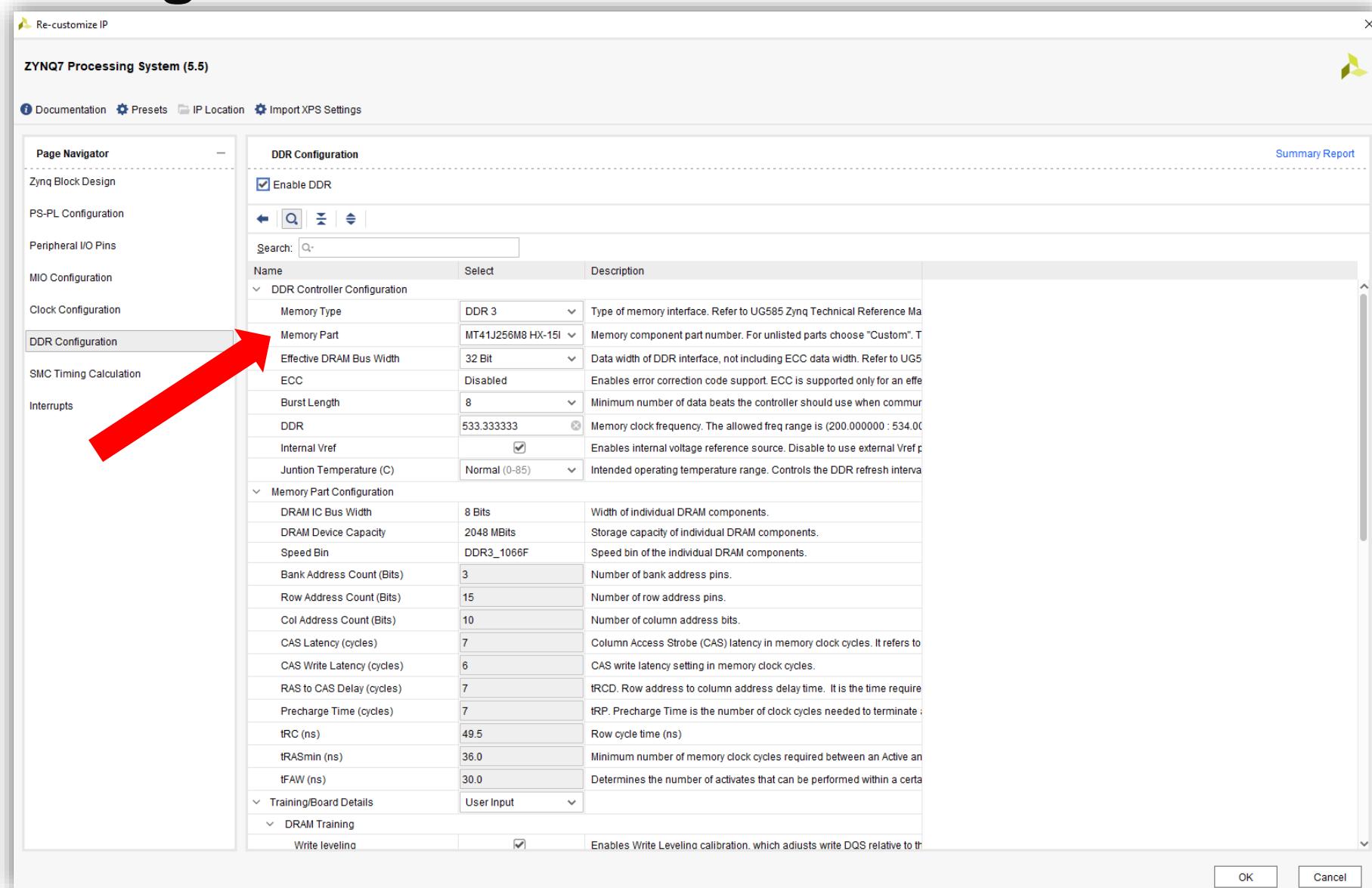
Zynq Block Design



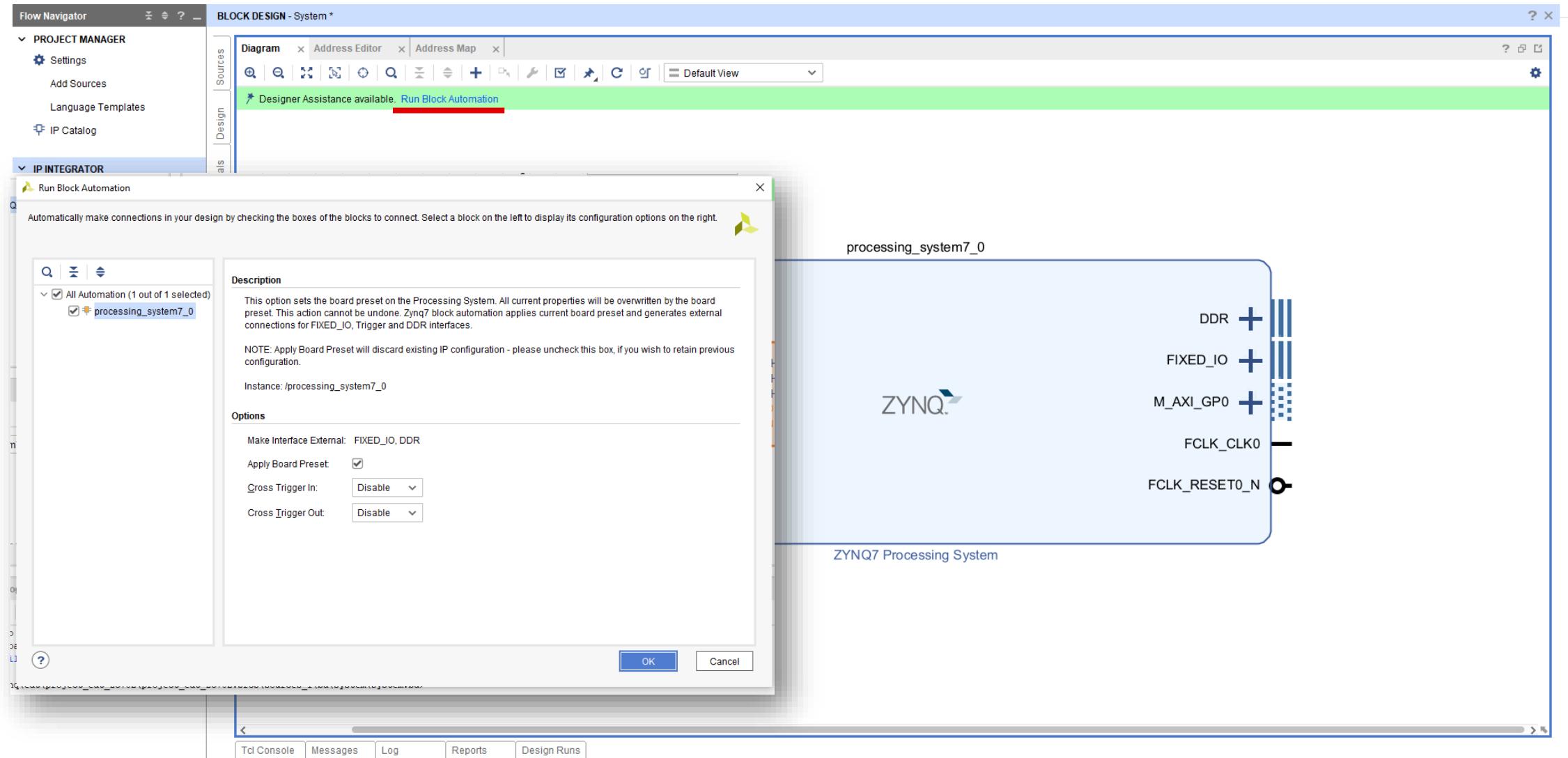
Basic Embedded System



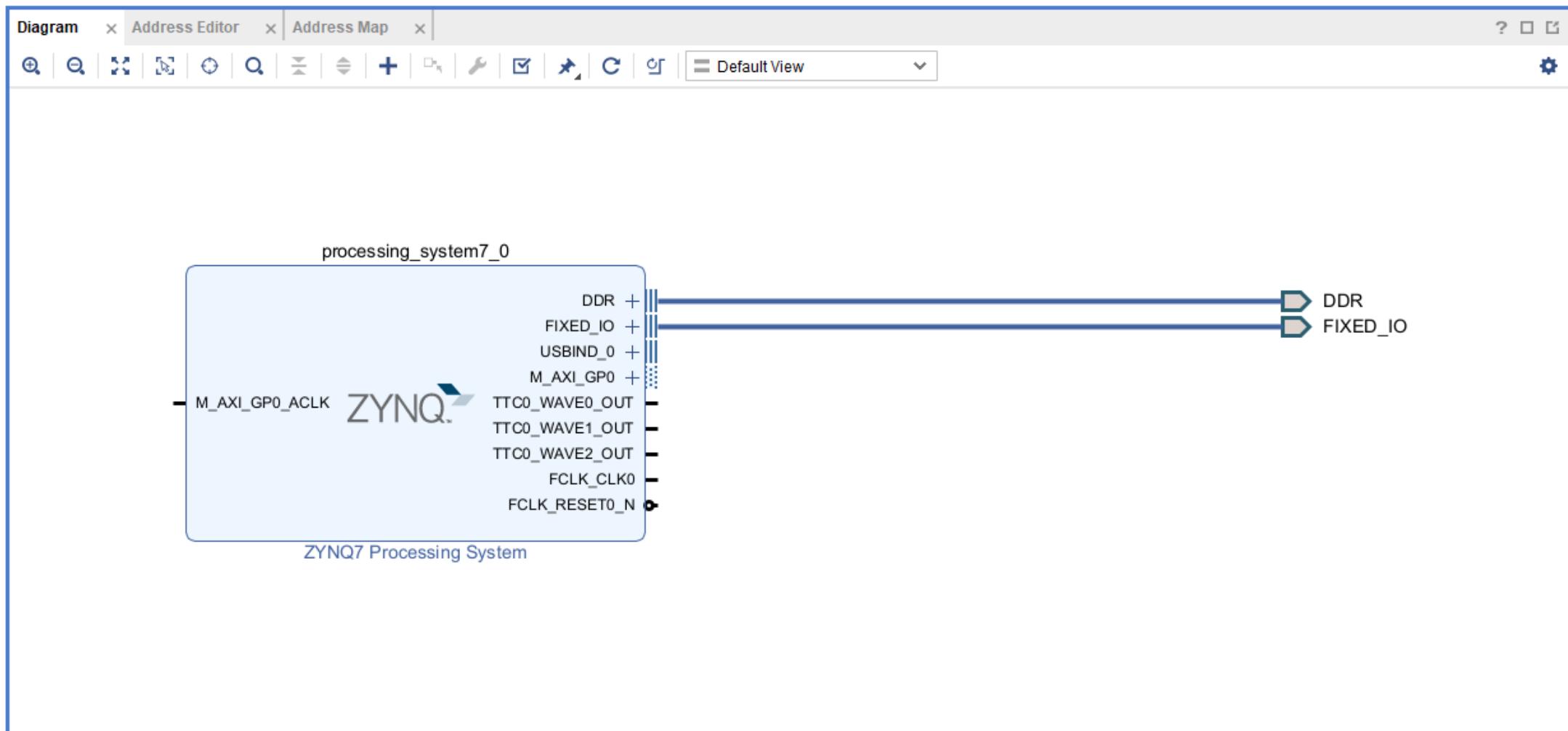
DDR Configuration



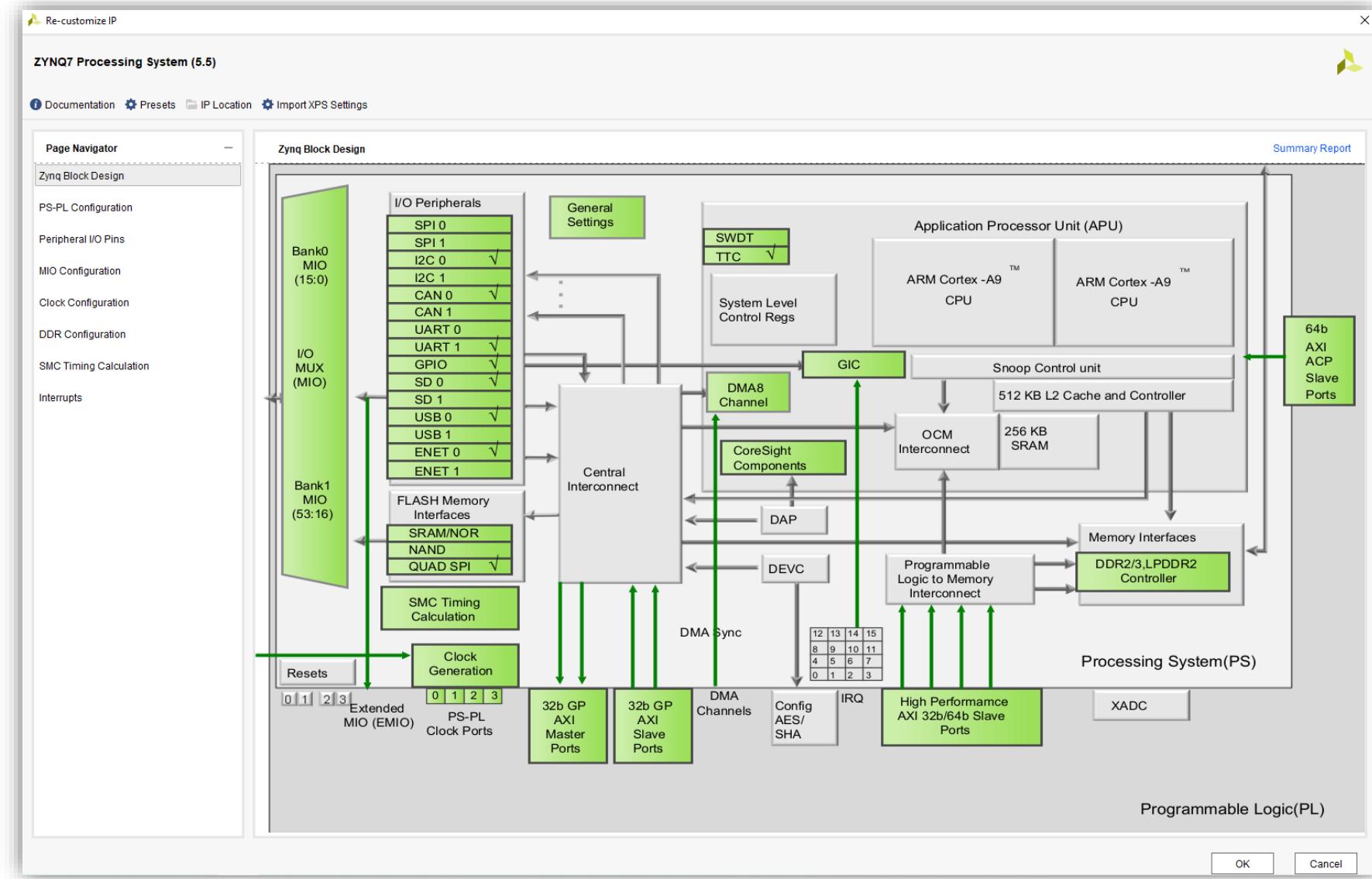
Run Block Automation



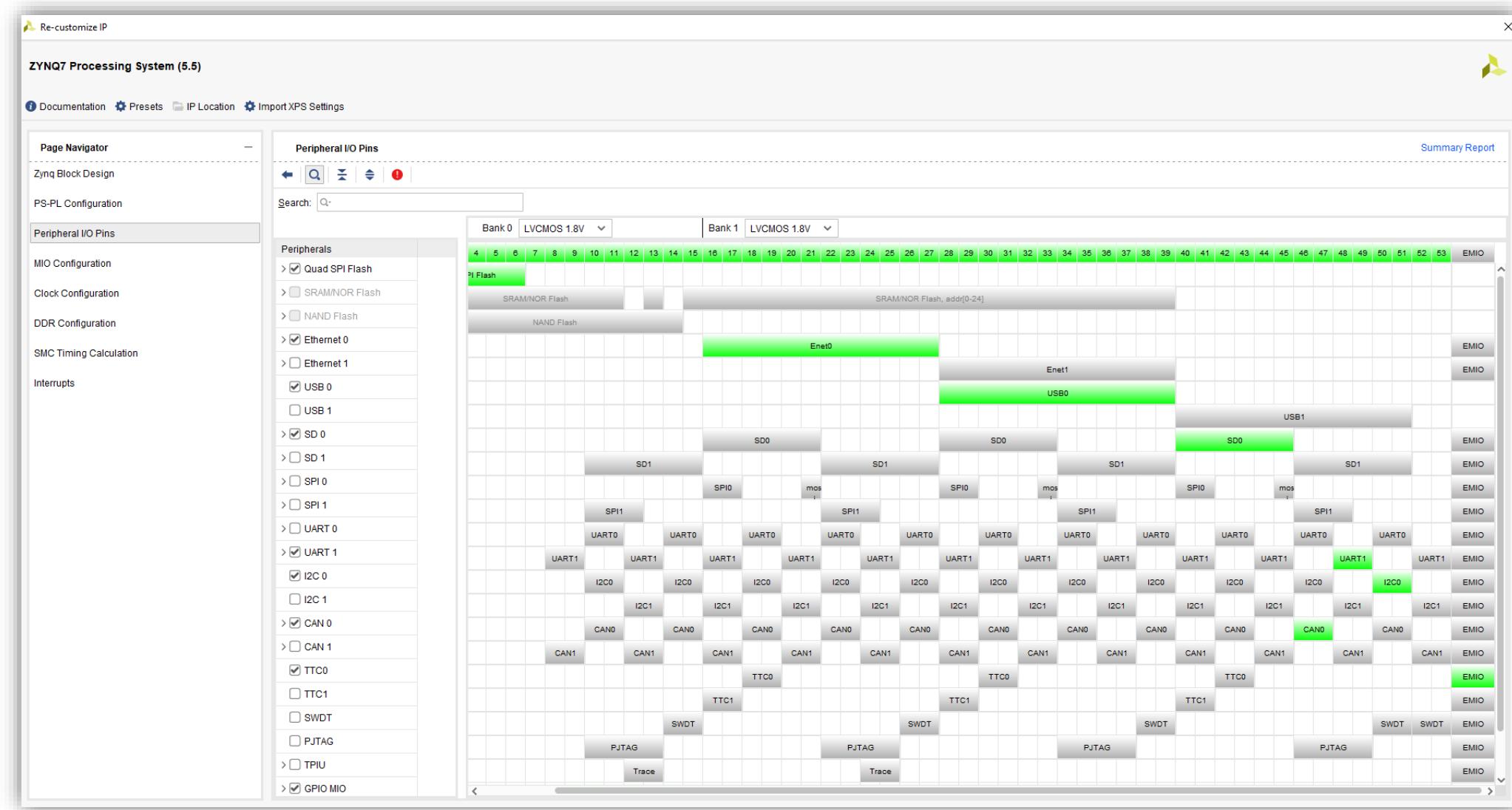
Run Block Automation



Zynq Block Design



Peripheral I/O Pins



PS – PL Configuration & MIO Configuration

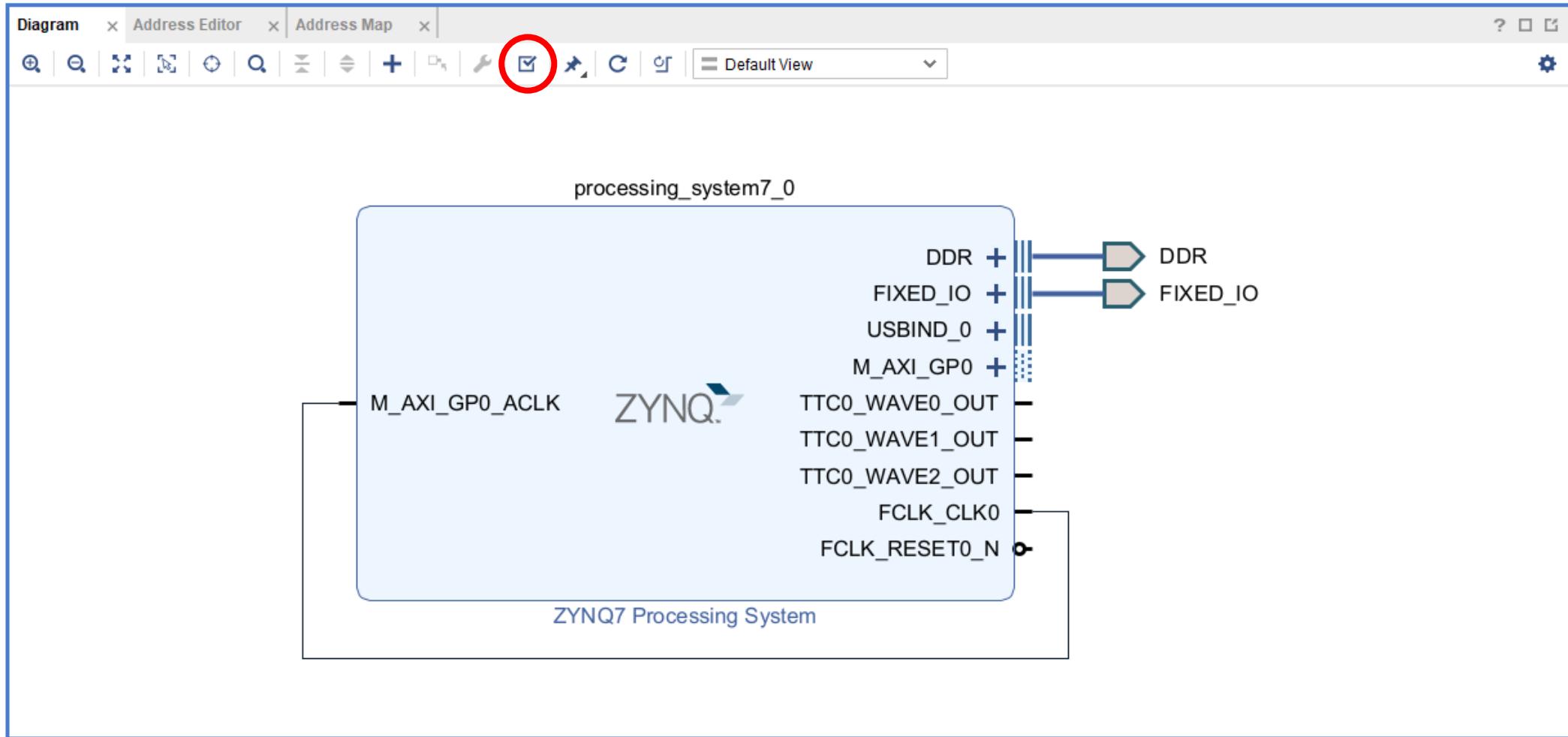
PS-PL Configuration

Name	Select	Description
UART0 Baud Rate	115200	Baud rate is generated with internally fixed UART Ref Clock Freq=1
UART1 Baud Rate	115200	Baud rate is generated with internally fixed UART Ref Clock Freq=1
PL AXI idle Port	<input type="checkbox"/>	Enables idle AXI signal to the PS used to indicate that there are no
DDR ARB bypass Port	<input type="checkbox"/>	Enables DDR urgent/arbitration signal used to signal a critical memory st
PS-PL Debug interface	<input type="checkbox"/>	Enables PL debug signals to PS and vice-versa
FTM Trace data interface	<input type="checkbox"/>	Enables FTM Trace AXI stream interface used to capture data from
FTM Trace buffer	0	Generates a FIFO to hold trace data
FTM Data edge detector	0	Stores trace data in the FIFO when the data changes as marked by
FTM Trace buffer FIFO size	128	FTM Trace buffer FIFO size
FTM Trace buffer clock delay	12	Number of clock cycles interval for a trace data output from FIFO be
Include ACP transaction checker	<input type="checkbox"/>	Enables ACP transaction checker.
Trace data/control signal pipeline width	8	Enables configurable number of pipeline stages on the TRACE DA
Power-on reset(POR) 4k timer	<input type="checkbox"/>	Enables power-on reset(POR) 4k timer. By default, 64k timer is use
Processor event interface	<input type="checkbox"/>	Enables event bus which provides a low-latency and direct mechan
> Address Editor		
> Enable Clock Triggers		
> Enable Clock Resets		
> AXI Non Secure Enablement	0	Enable AXI Non Secure Transaction
> GP Slave AXI Interface		
> HP Slave AXI Interface		
> ACP Slave AXI Interface		
> DMA Controller		
> PS-PL Cross Trigger interface	<input type="checkbox"/>	Enables PL cross trigger signals to PS and vice-versa
> PS-PL Cross Trigger interface	<input type="checkbox"/>	Enables PS-PL cross trigger interface
> DMI Controller		
> Selectable AXI Write Cache		
> Selectable AXI Write Cache		

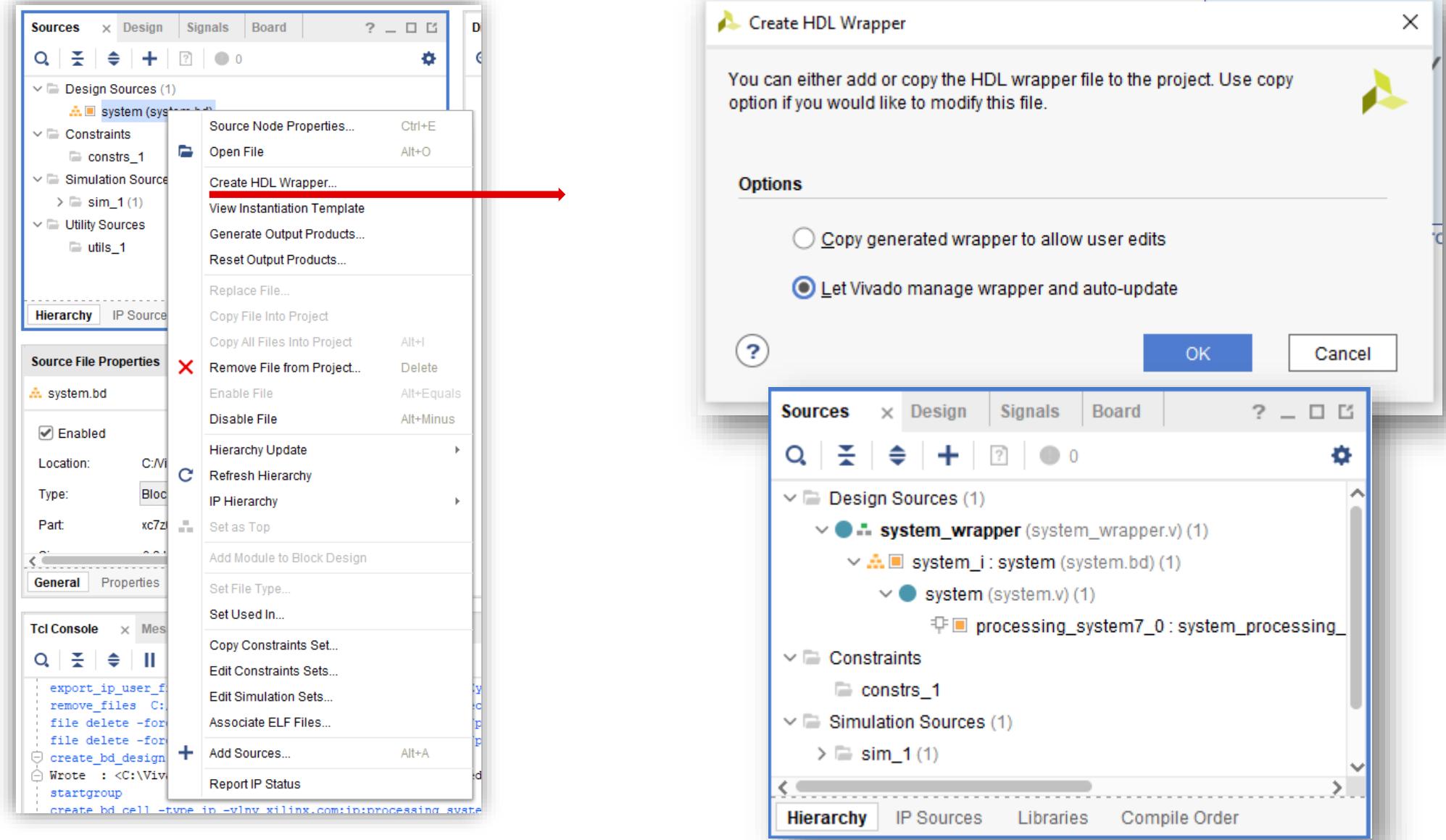
MIO Configuration

Peripheral	IO	Signal	IO Type	Speed	Pullup	Direction	Polarity
Memory Interfaces							
> Quad SPI Flash	MIO 1 .. 6						
> SRAM/NOR Flash							
> NAND Flash							
I/O Peripherals							
> ENET 0	MIO 16 .. 27						
> ENET 1							
> USB 0	MIO 28 .. 39						
> USB 1							
> SD 0	MIO 40 .. 45						
> SD 1							
> UART 0							
> <input checked="" type="checkbox"/> UART 1	MIO 48 .. 49						
Modem signals							
UART 1	MIO 48	tx	LVCMS 1.8V	slow	disable	out	
UART 1	MIO 49	rx	LVCMS 1.8V	slow	disable	in	
> <input checked="" type="checkbox"/> I2C 0	MIO 50 .. 51						
> I2C 1							
> SPI 0							
> SPI 1							
> <input checked="" type="checkbox"/> CAN 0	MIO 46 .. 47						
> CAN 1							
> GPIO							
Application Processor Unit							
> <input checked="" type="checkbox"/> Timer 0	EMIO						
> Timer 1							

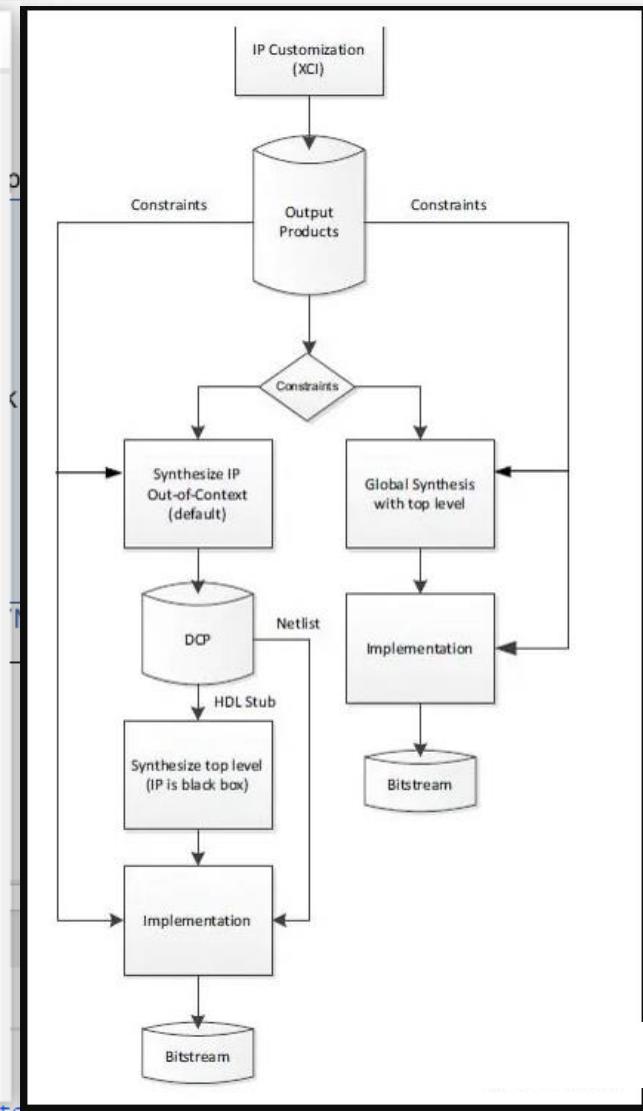
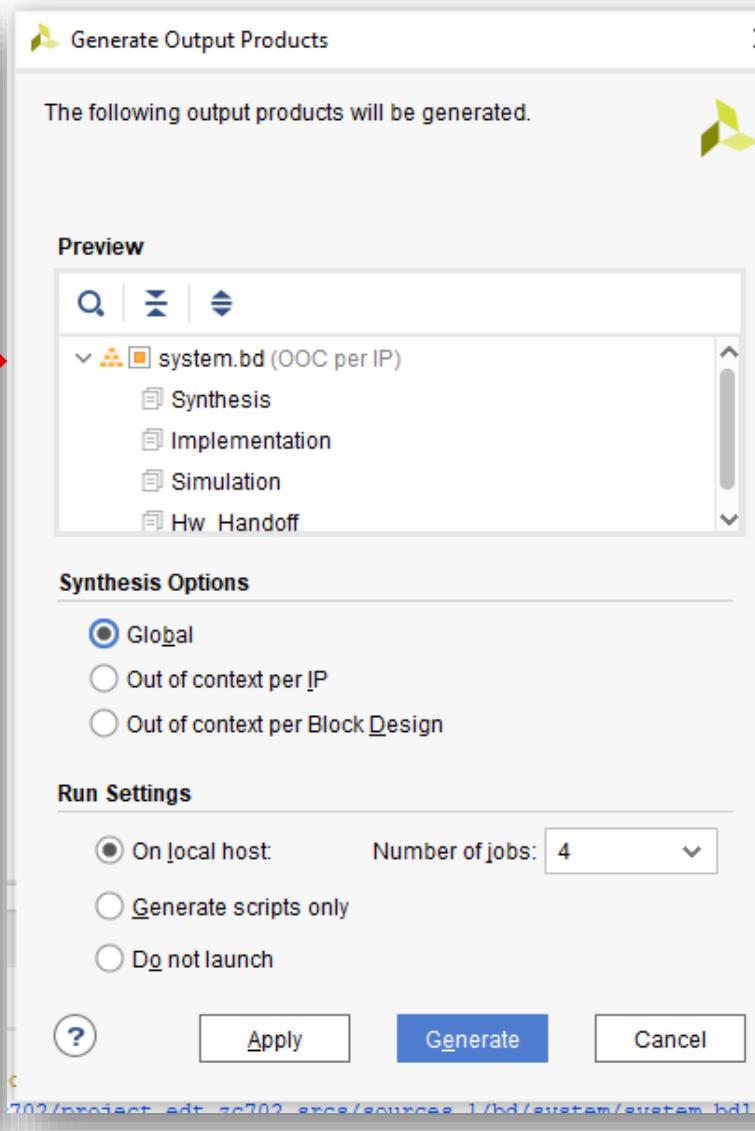
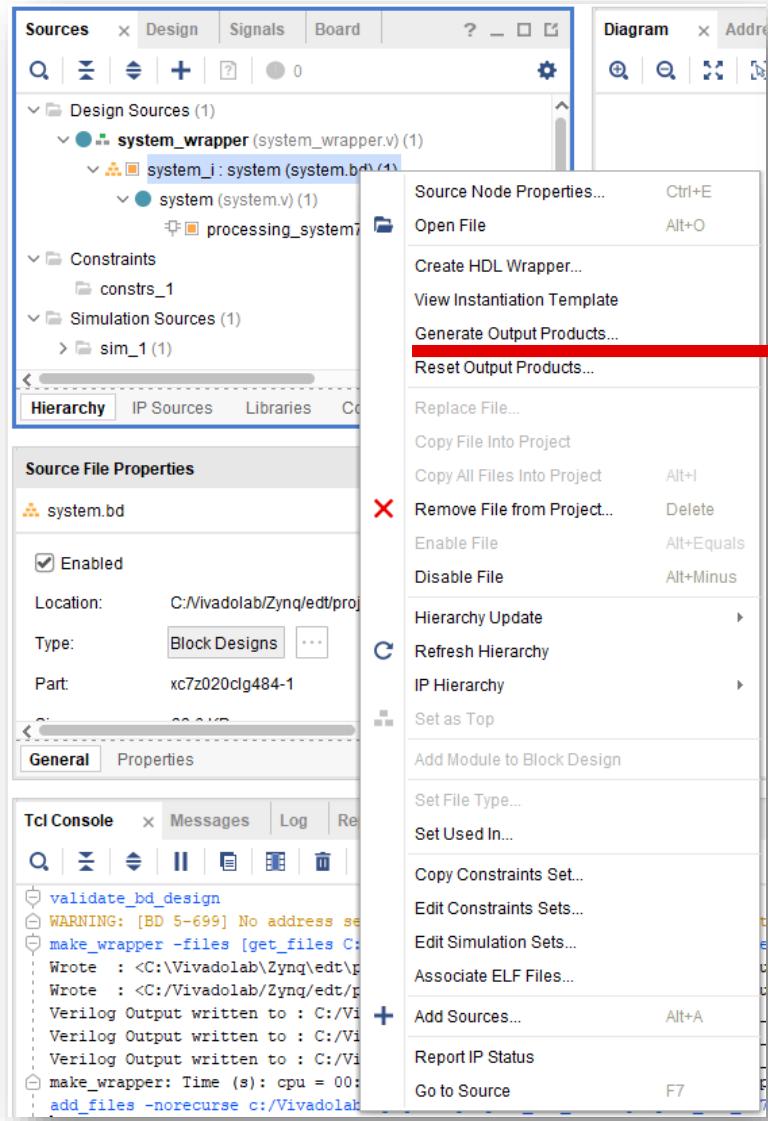
Run Block Automation



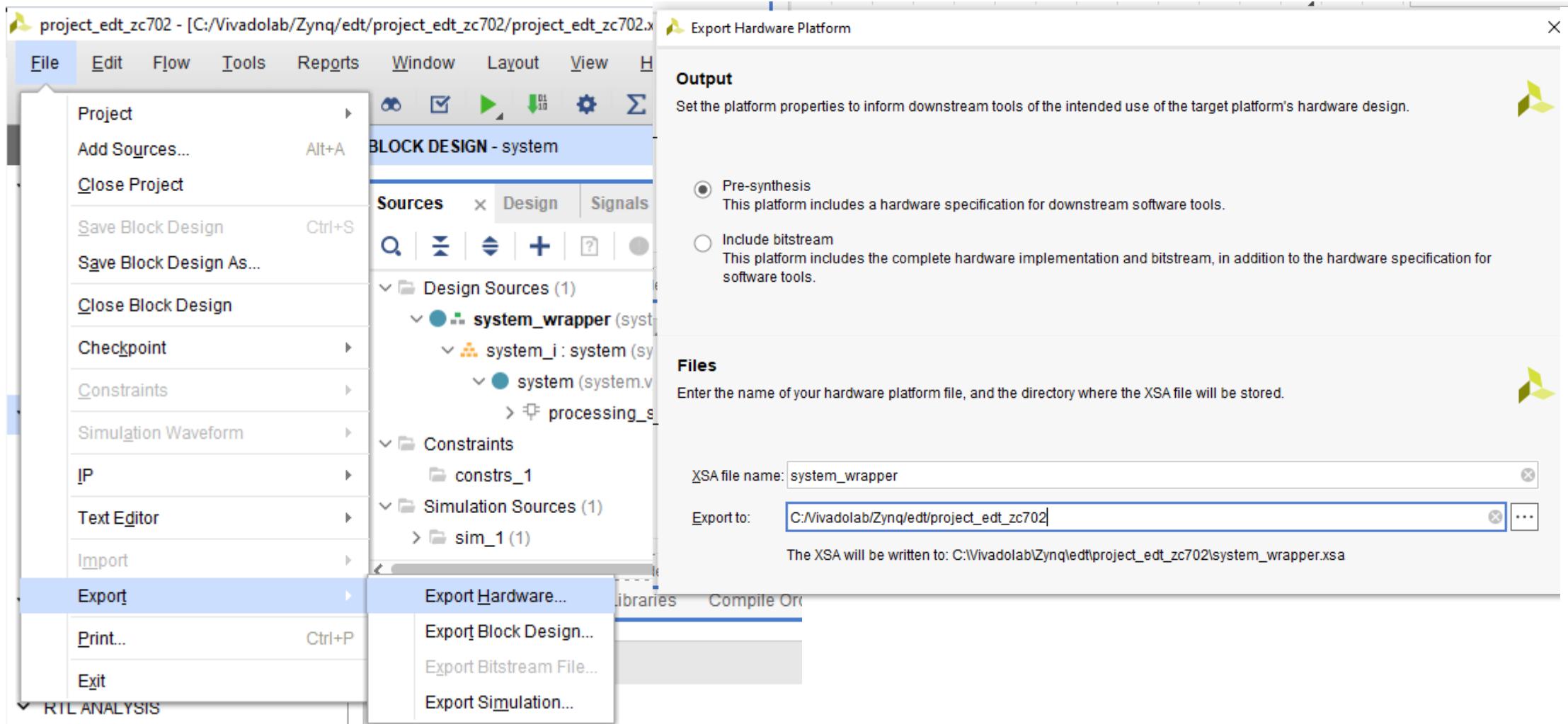
Create HDL Wrapper



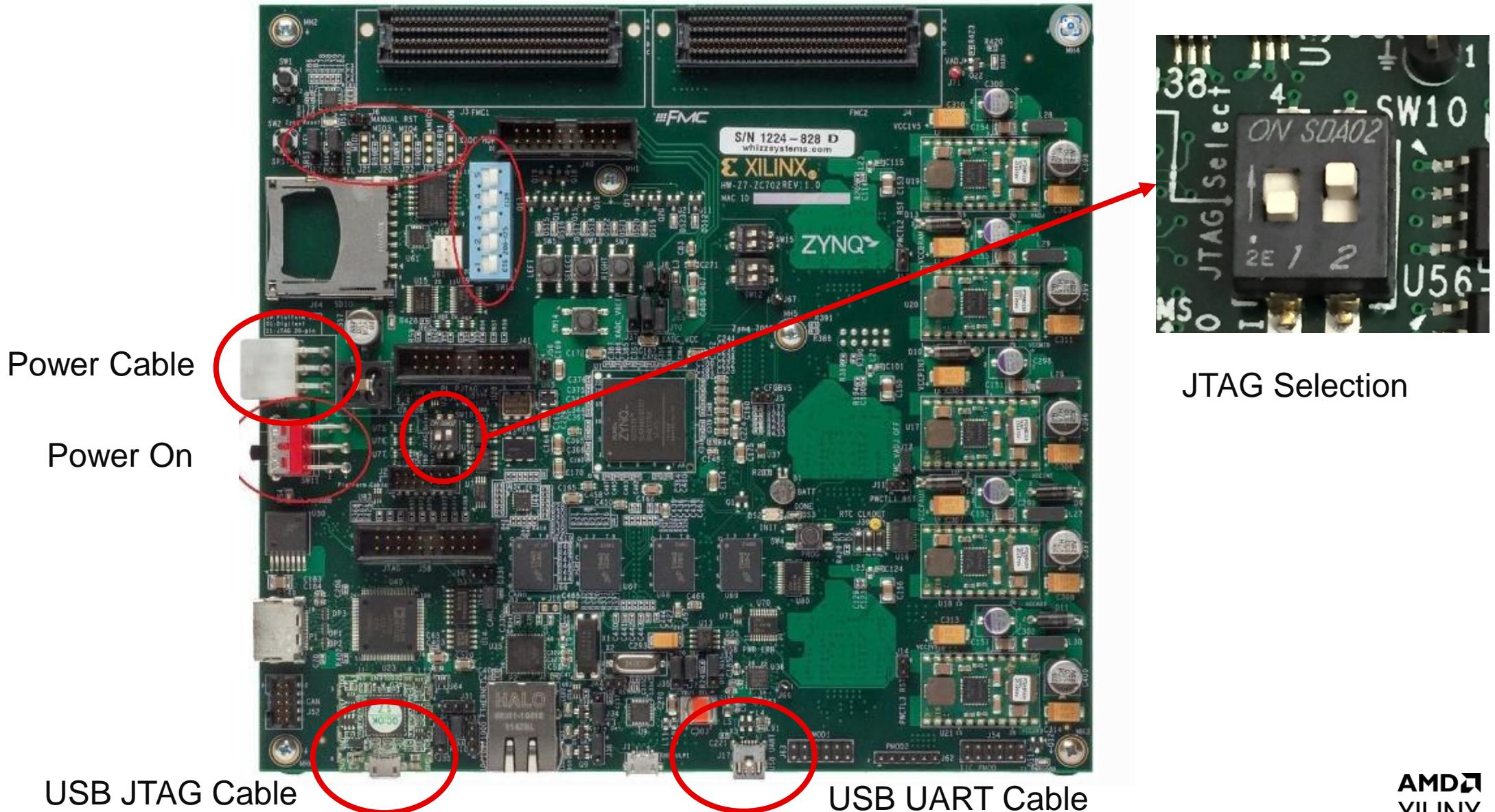
Generate Output Products



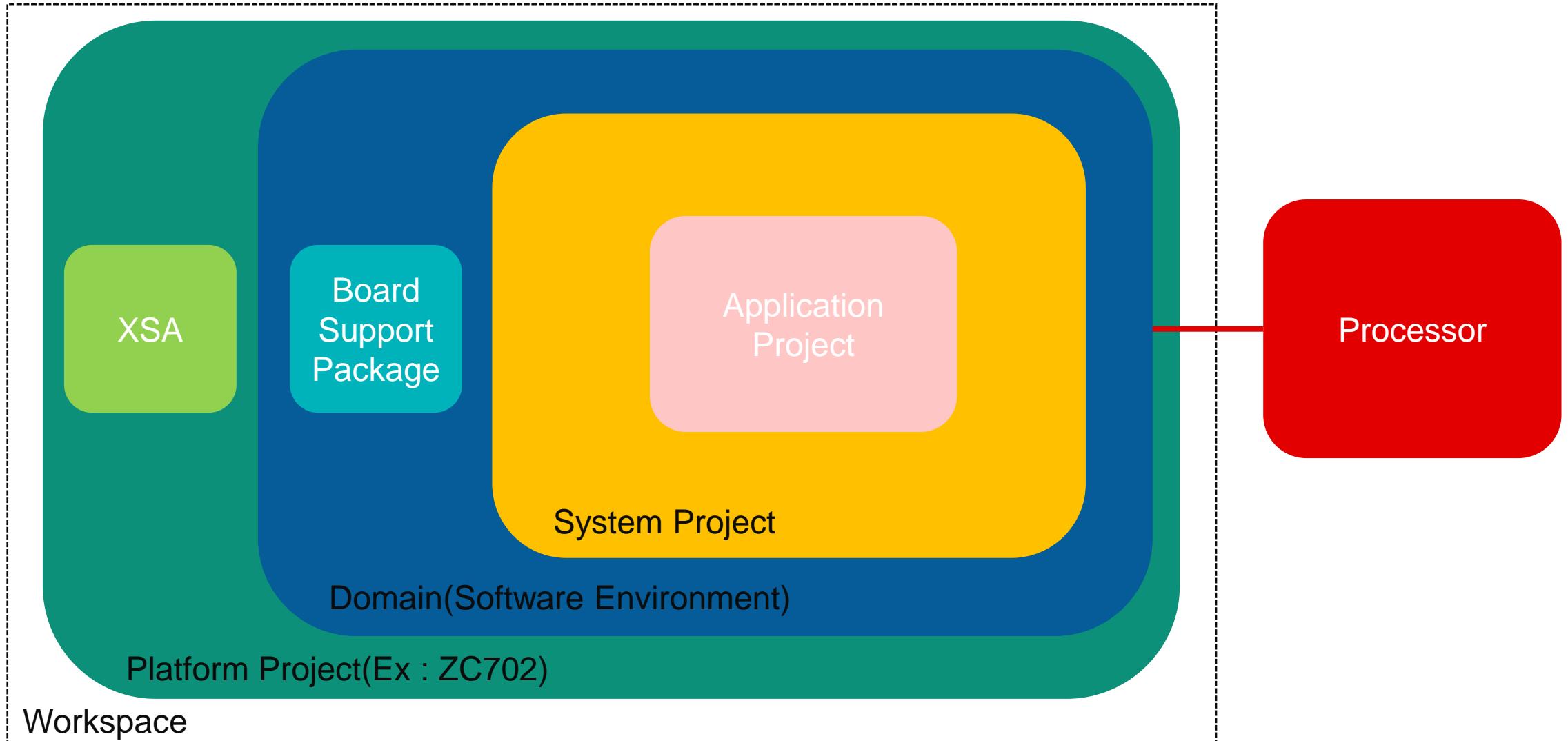
Exporting Hardware



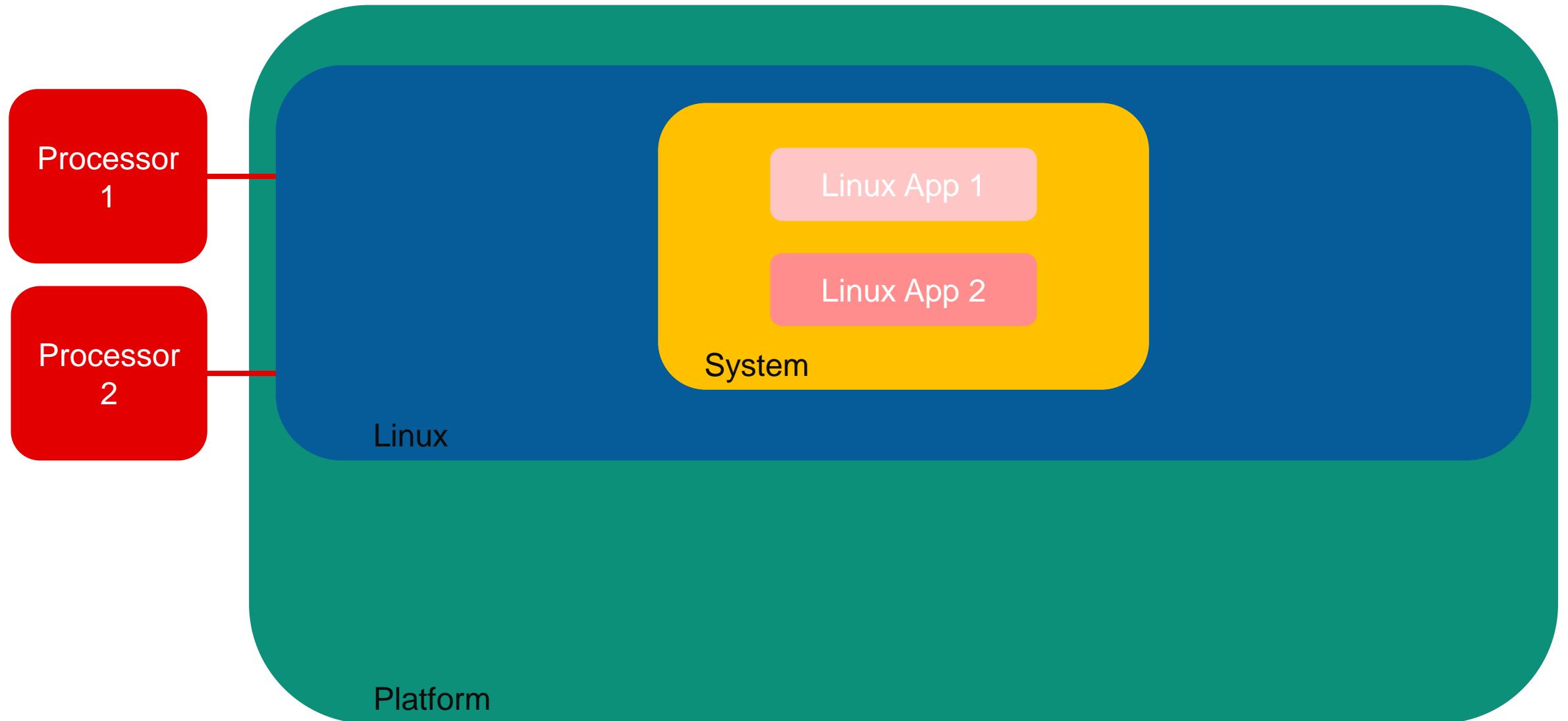
Setting Up the Board



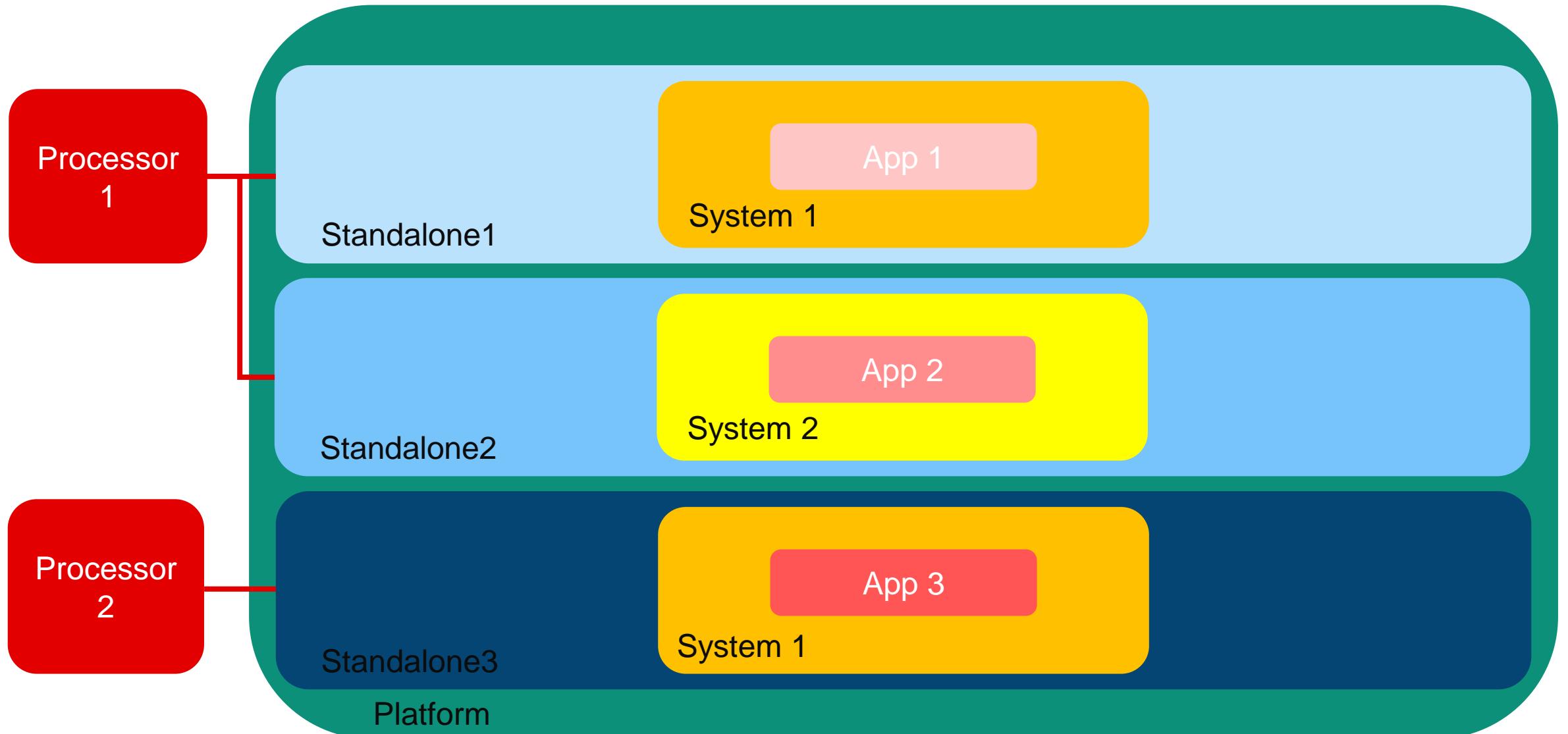
Vitis Workspace Structure



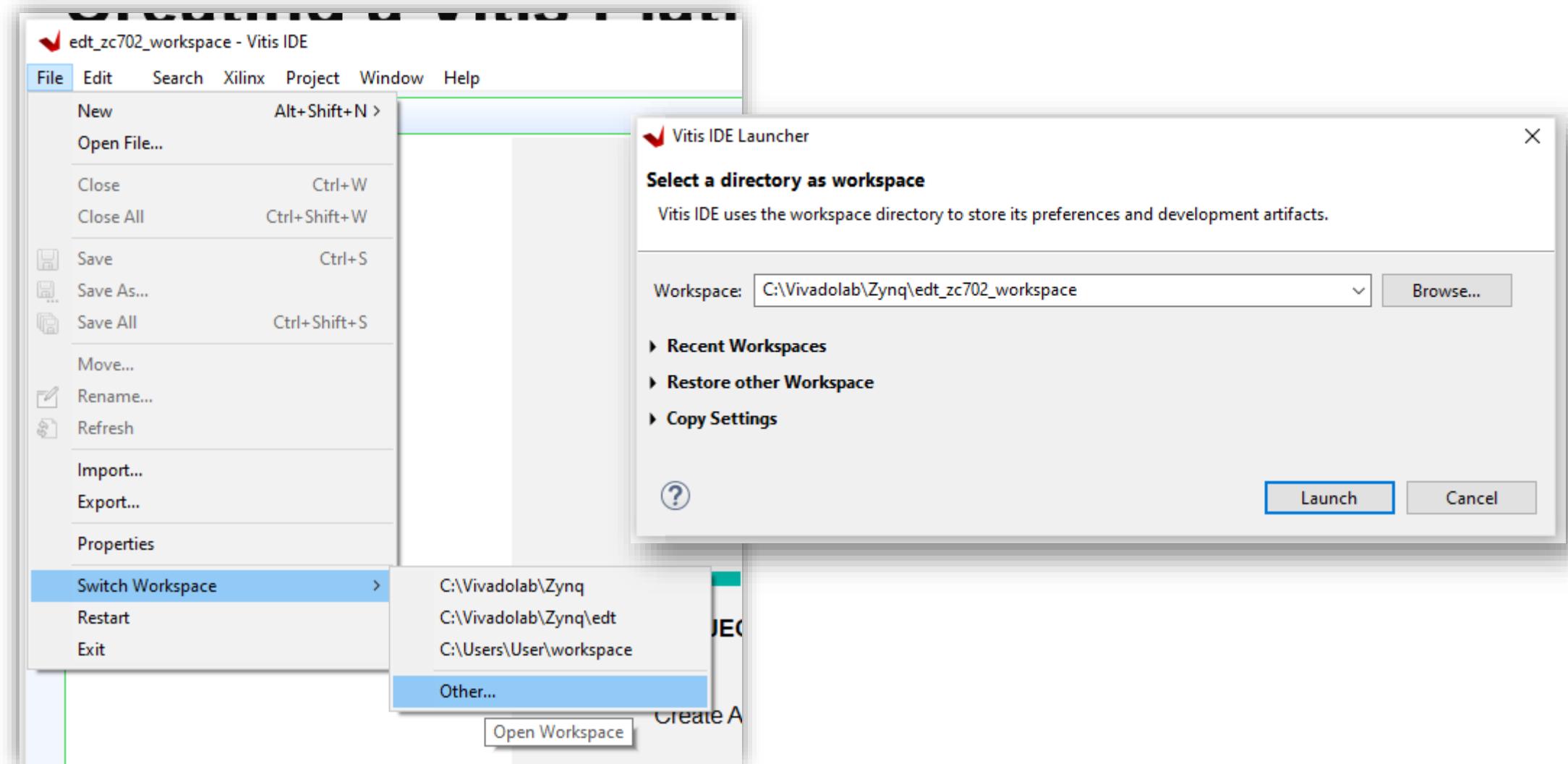
Vitis Workspace Structure



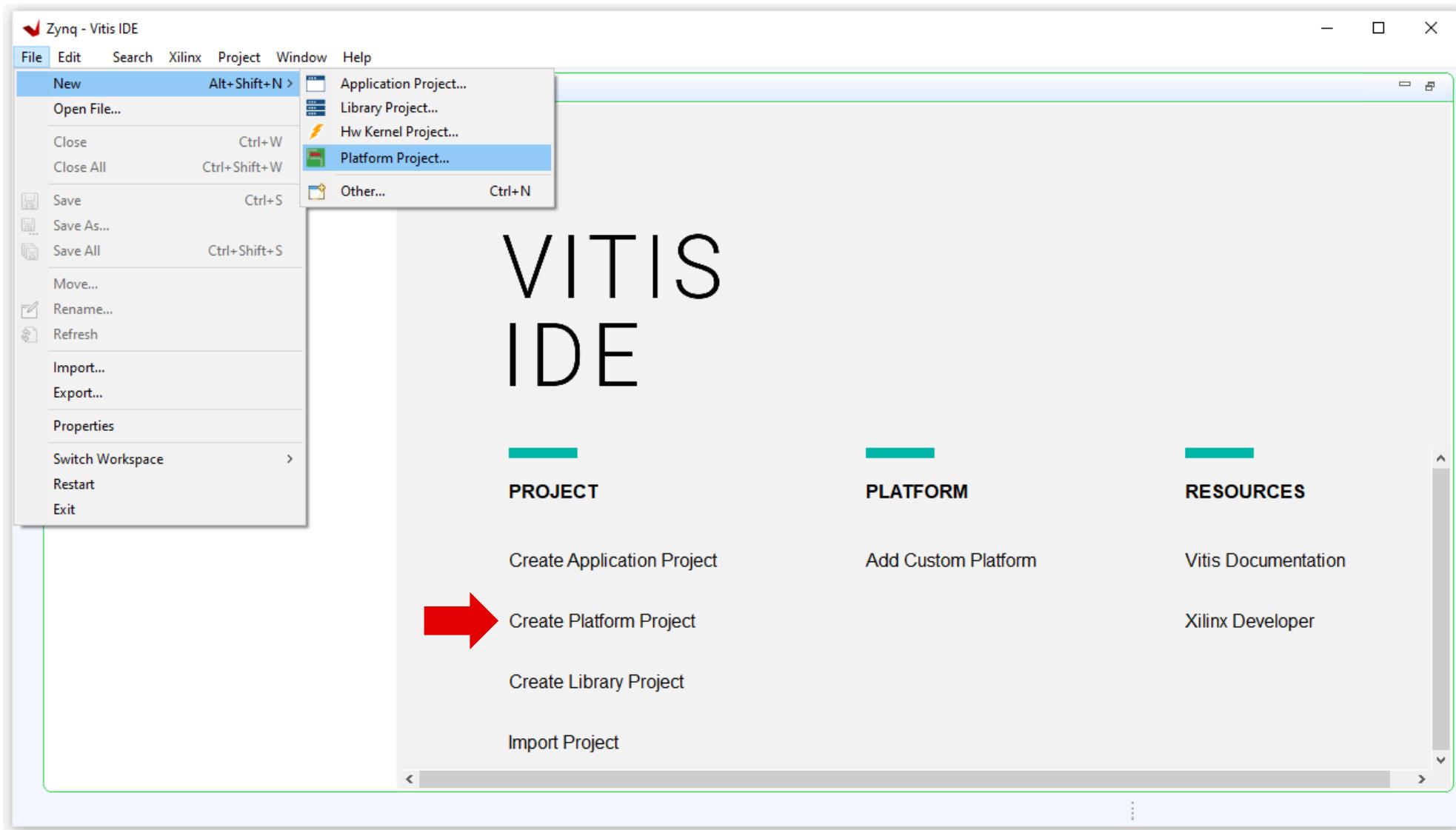
Vitis Workspace Structure



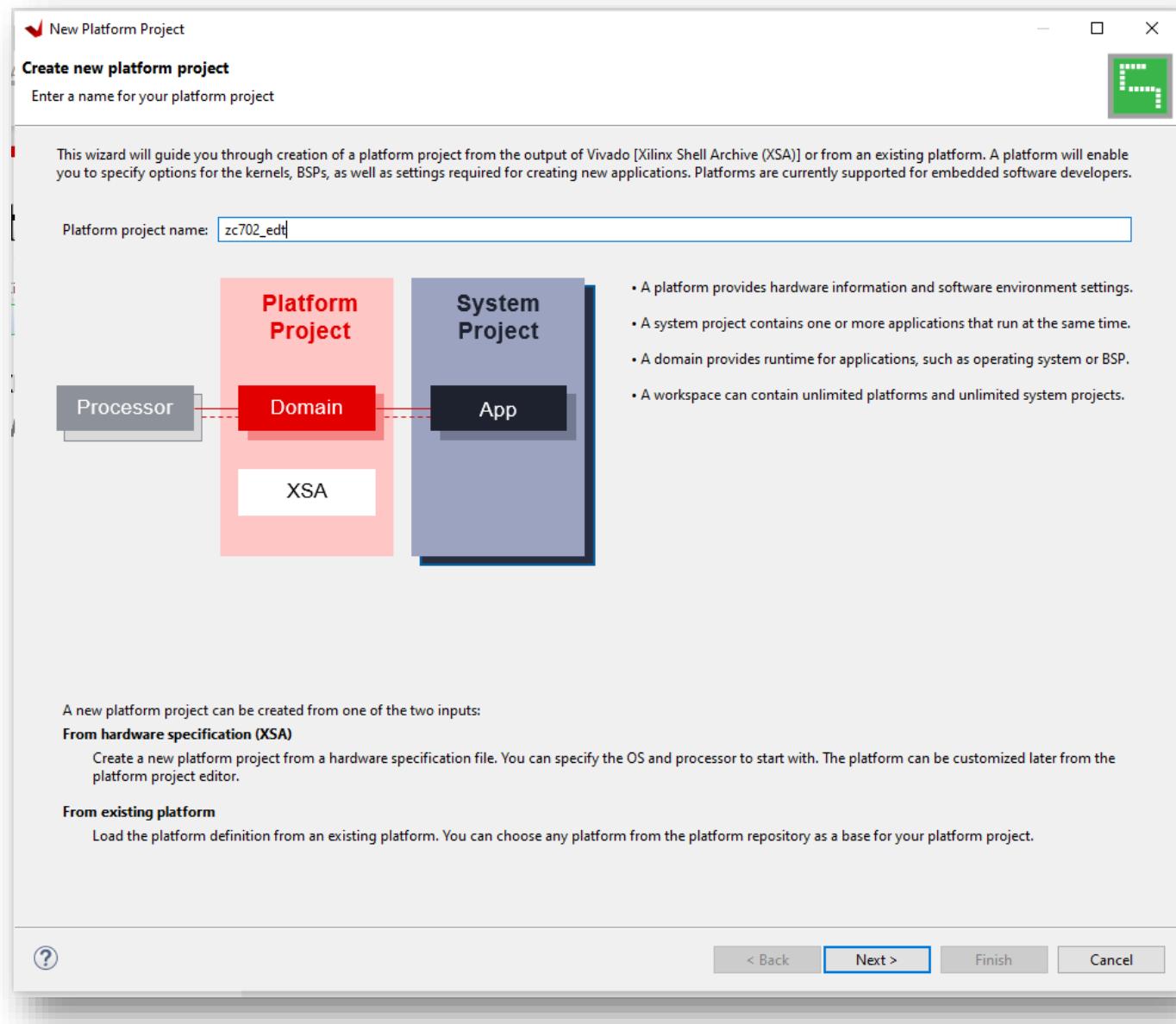
Creating a Vitis Platform Project



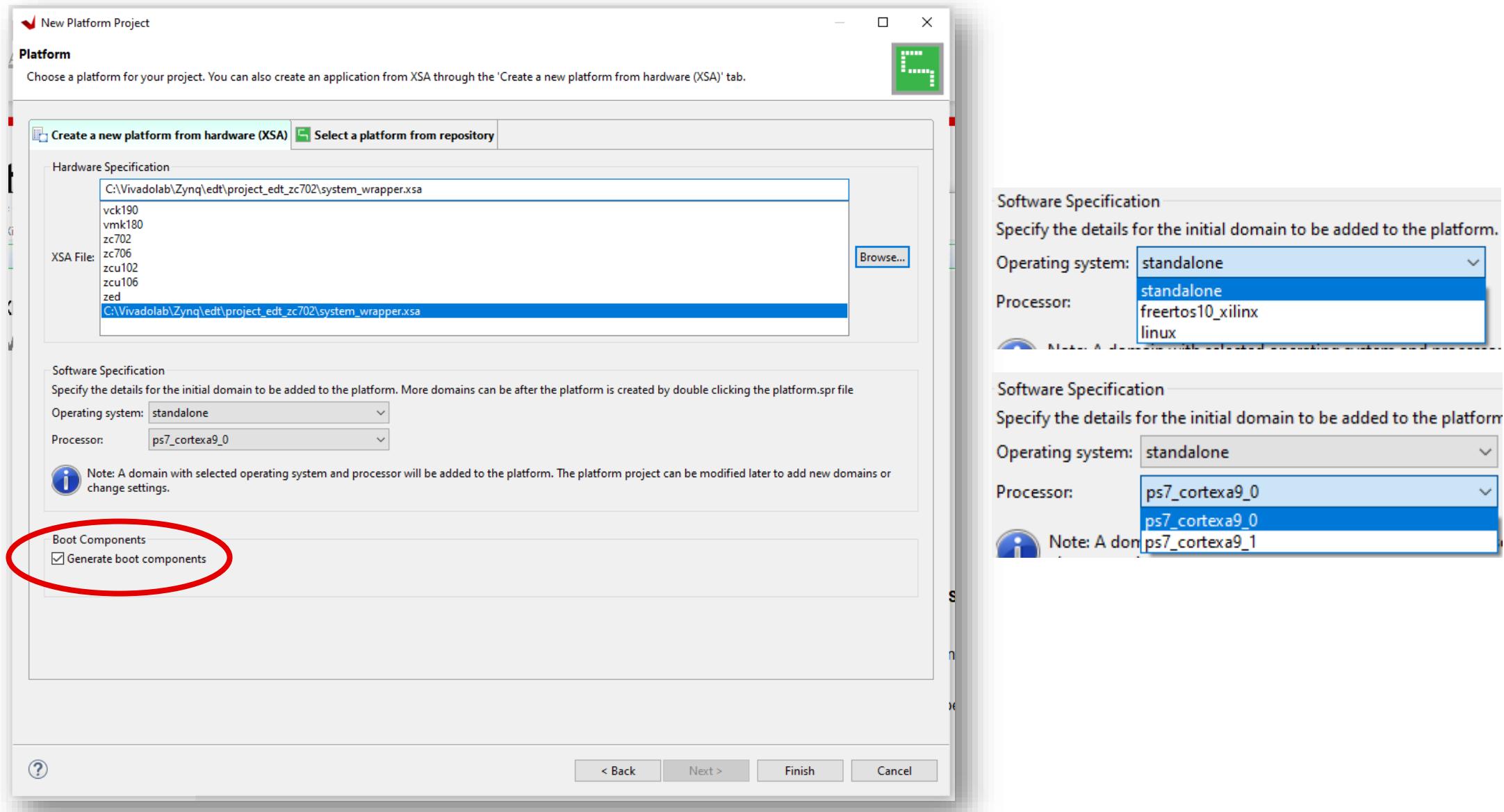
Creating a Vitis Platform Project



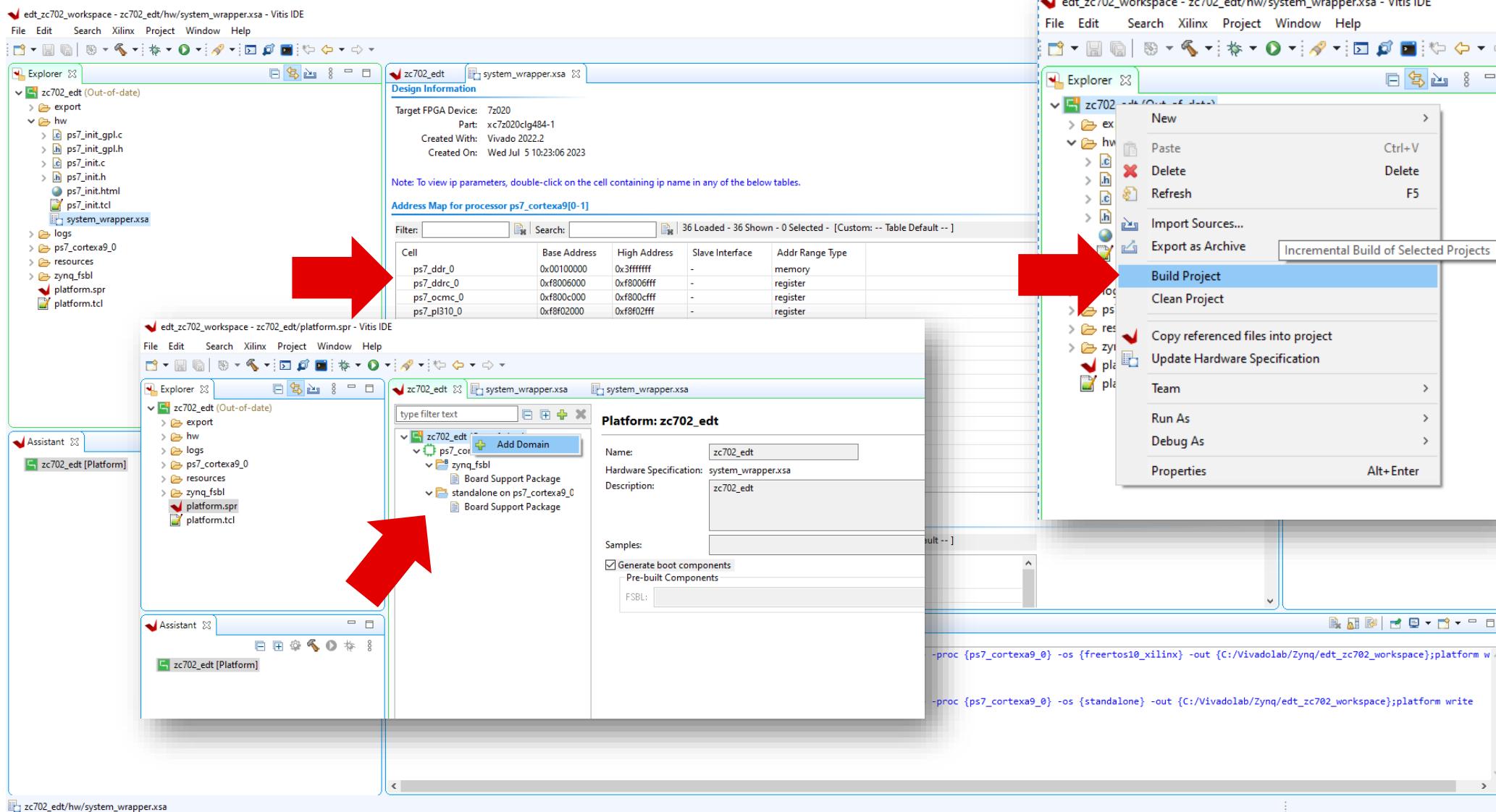
Creating a Vitis Platform Project



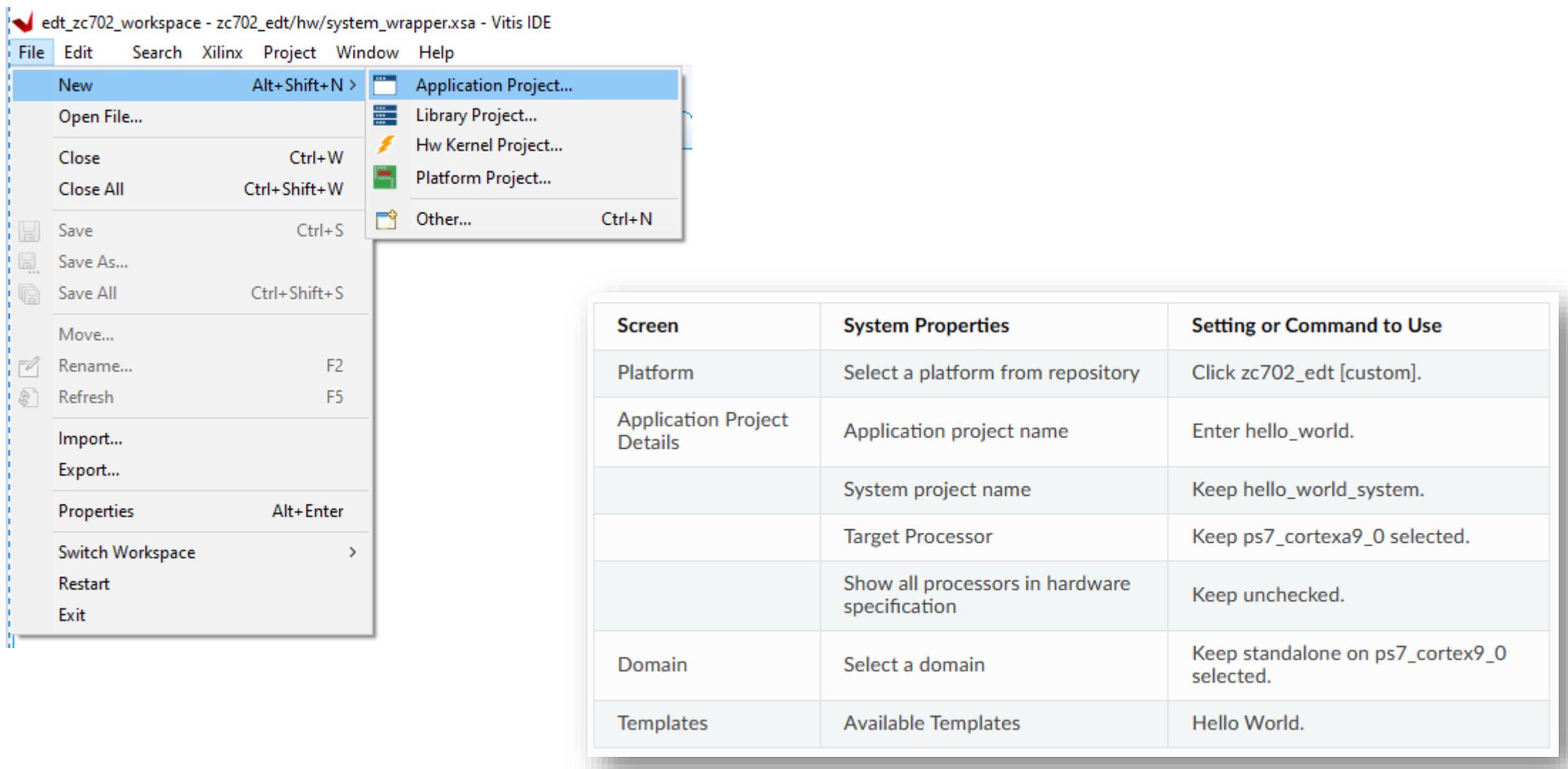
Creating a Vitis Platform Project



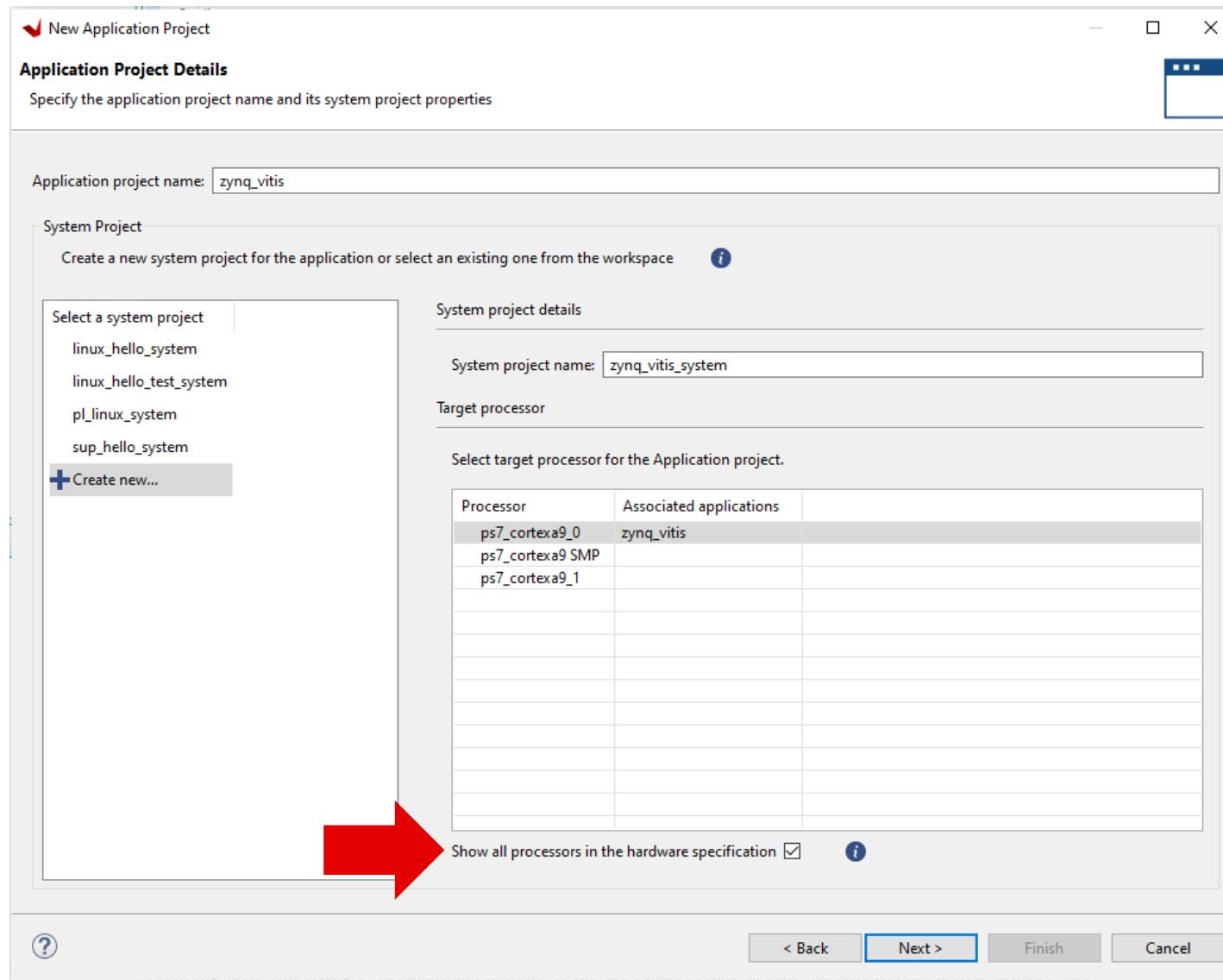
Creating a Vitis Platform Project



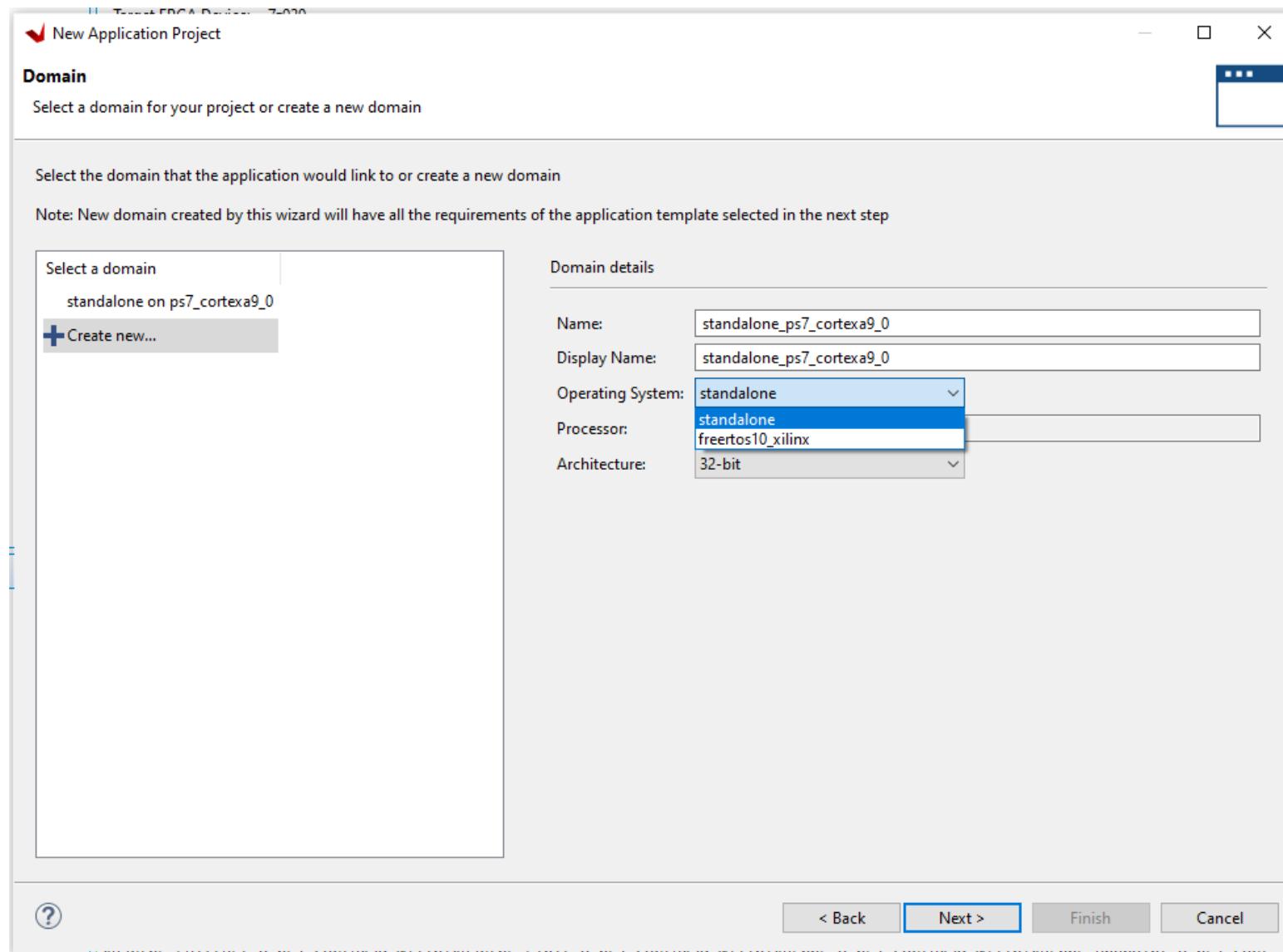
Creating the Hello World Application



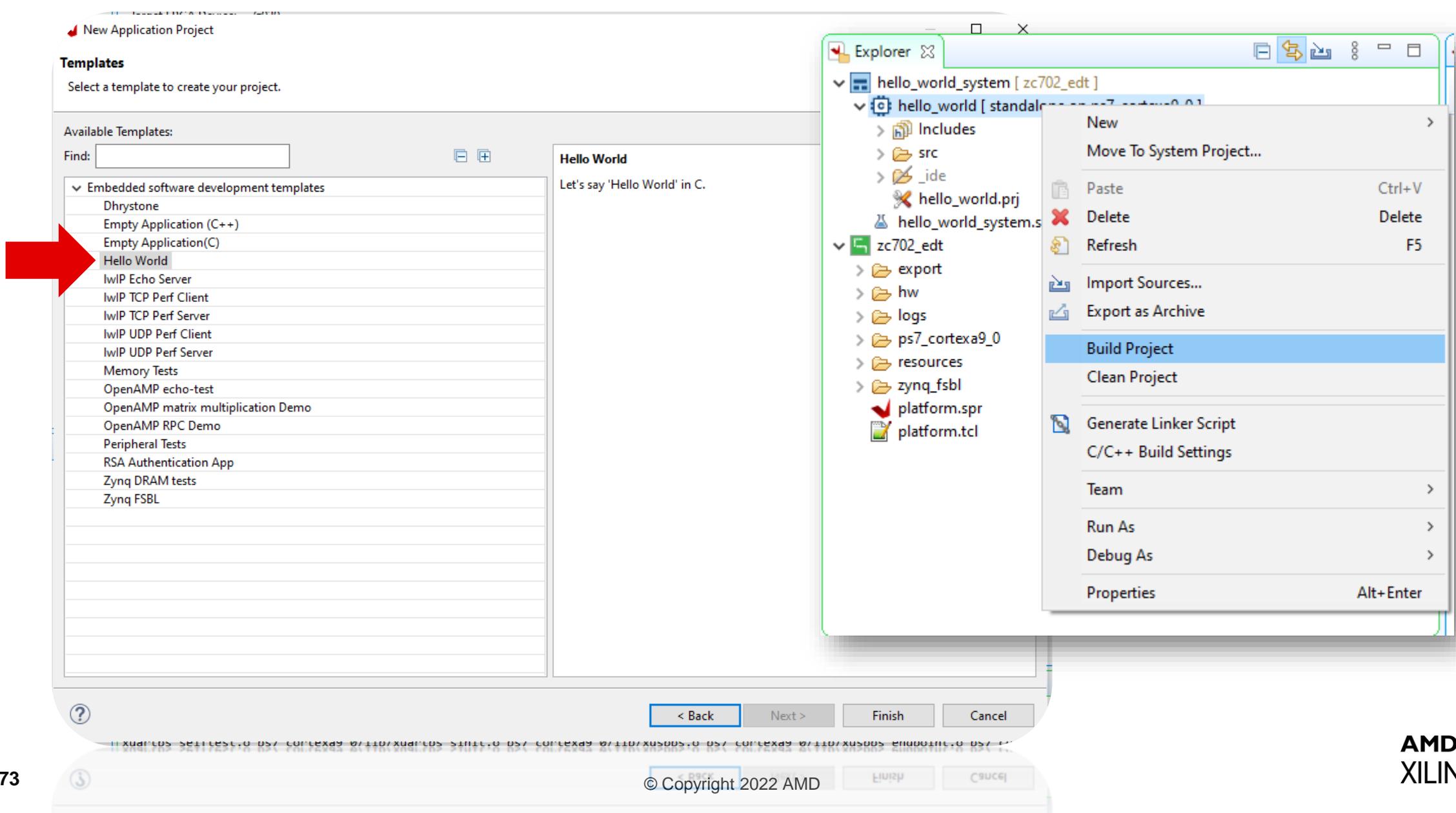
Creating the Hello World Application



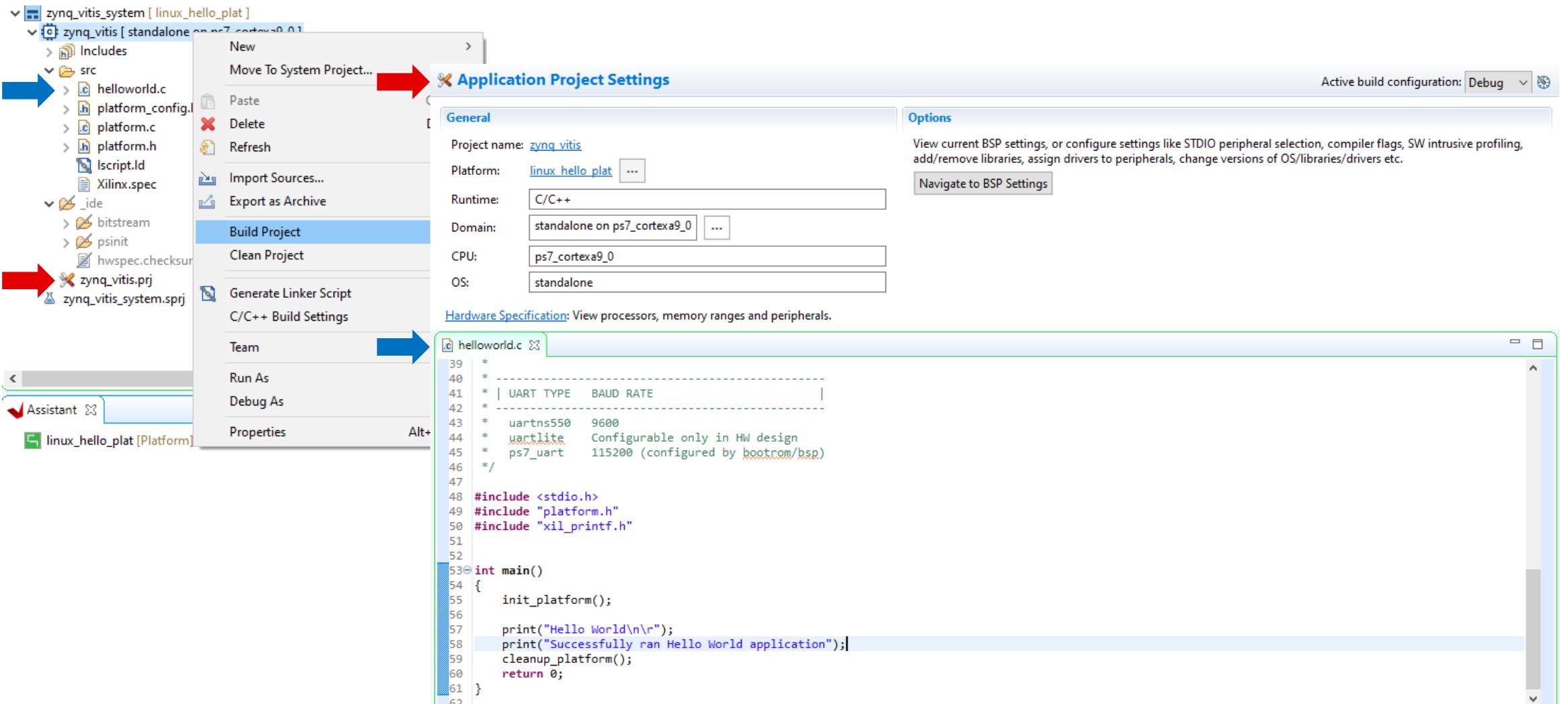
Creating the Hello World Application



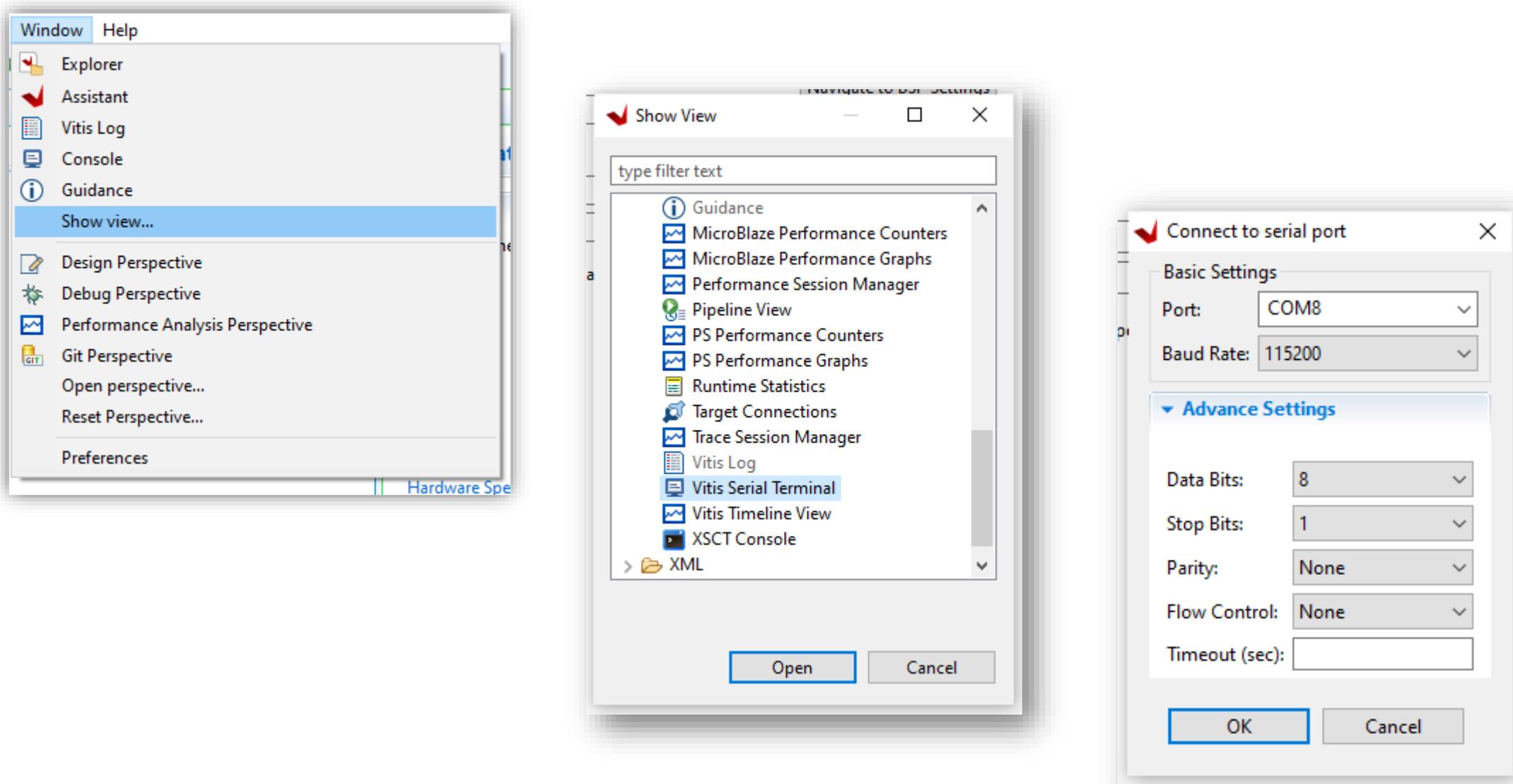
Creating the Hello World Application



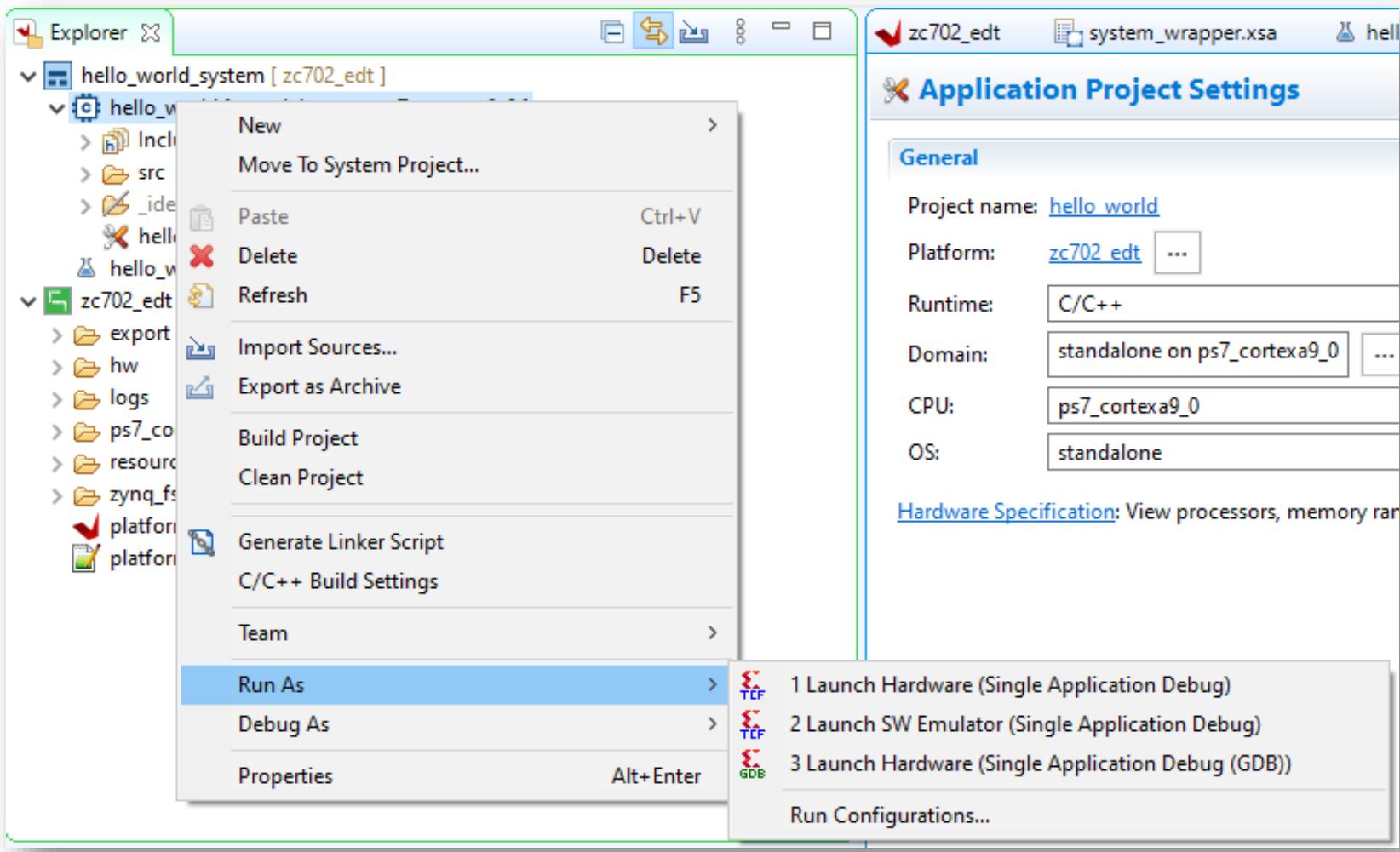
Creating the Hello World Application



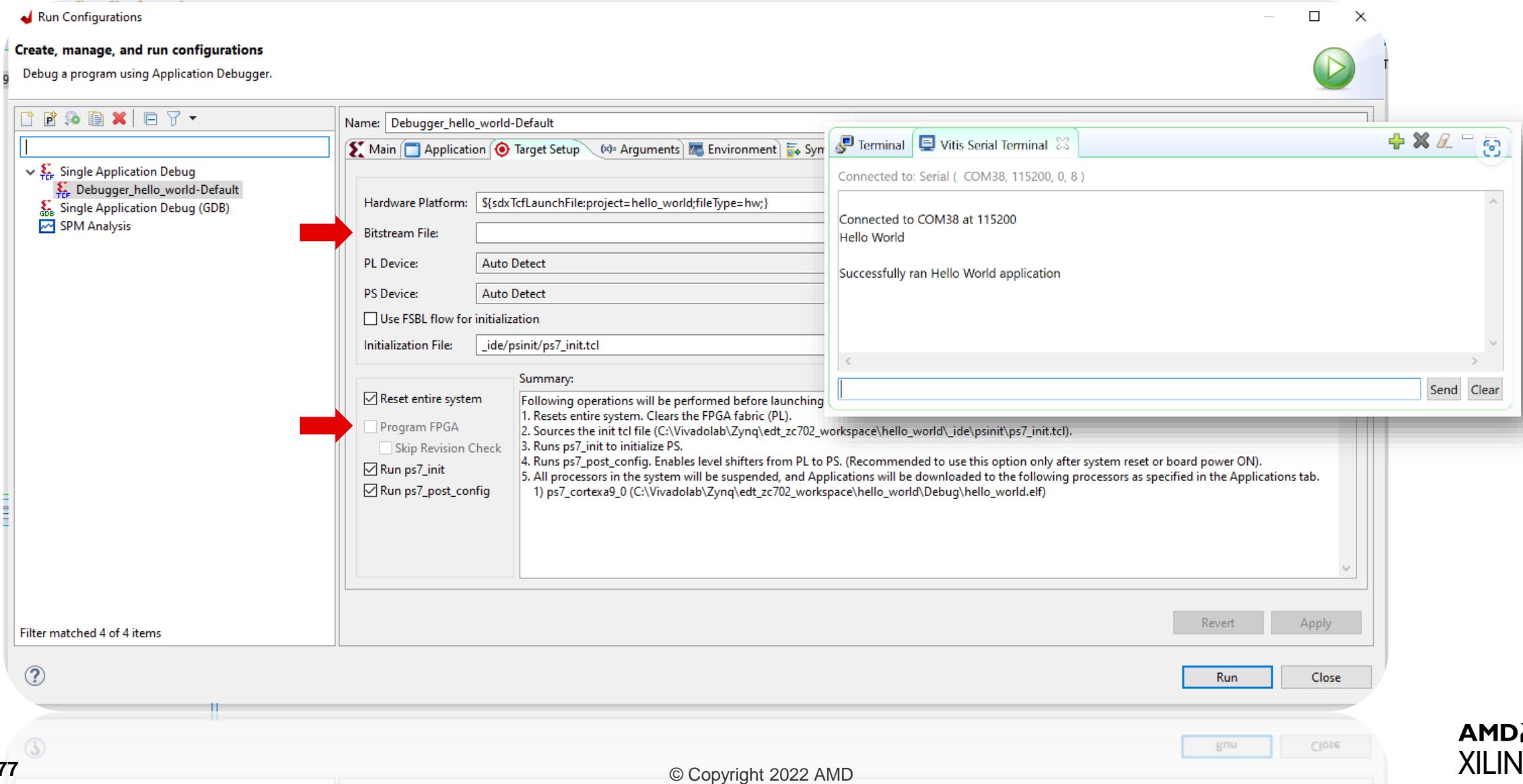
Running the Hello World Application on a ZC702 Board



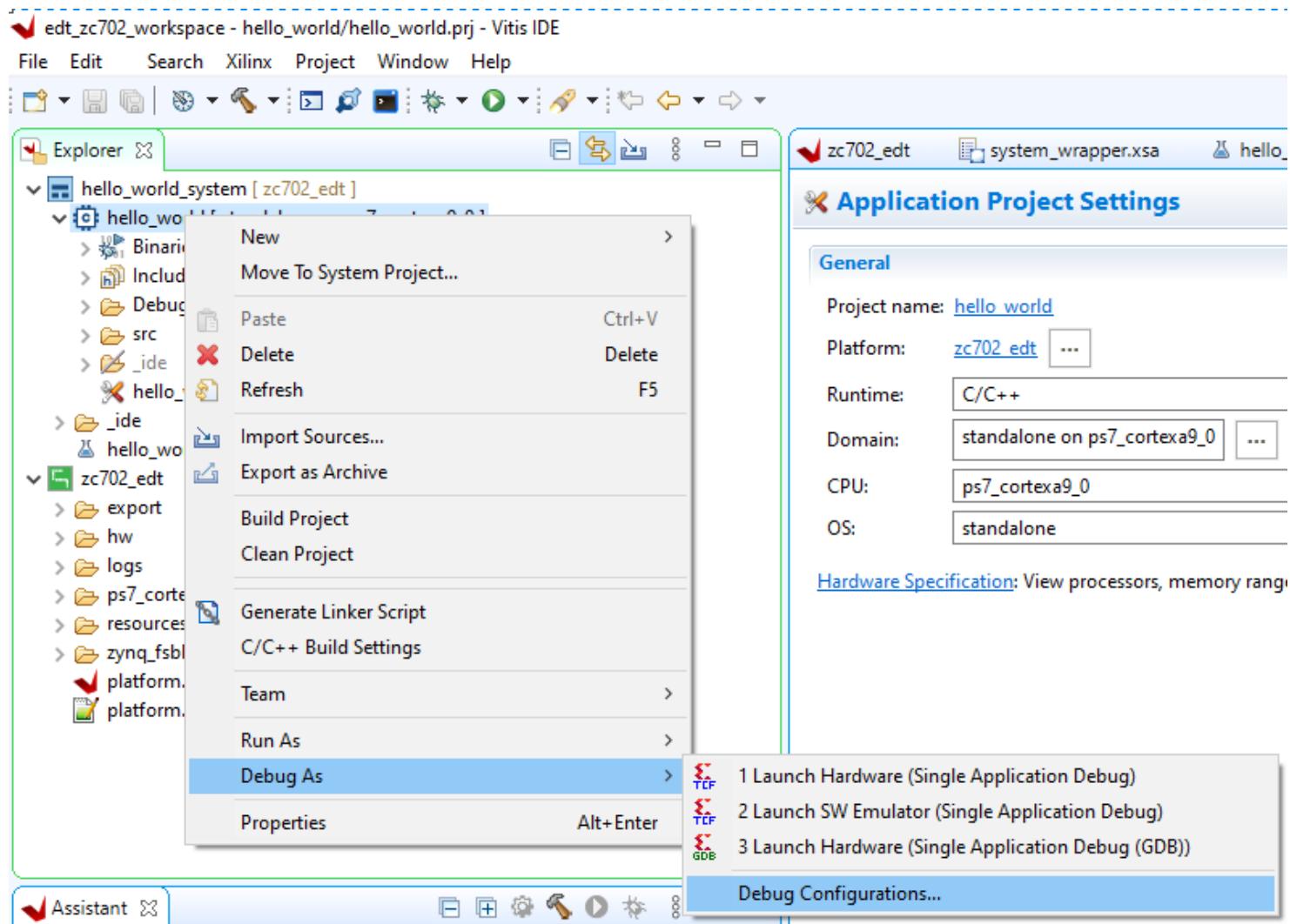
Running the Hello World Application on a ZC702 Board



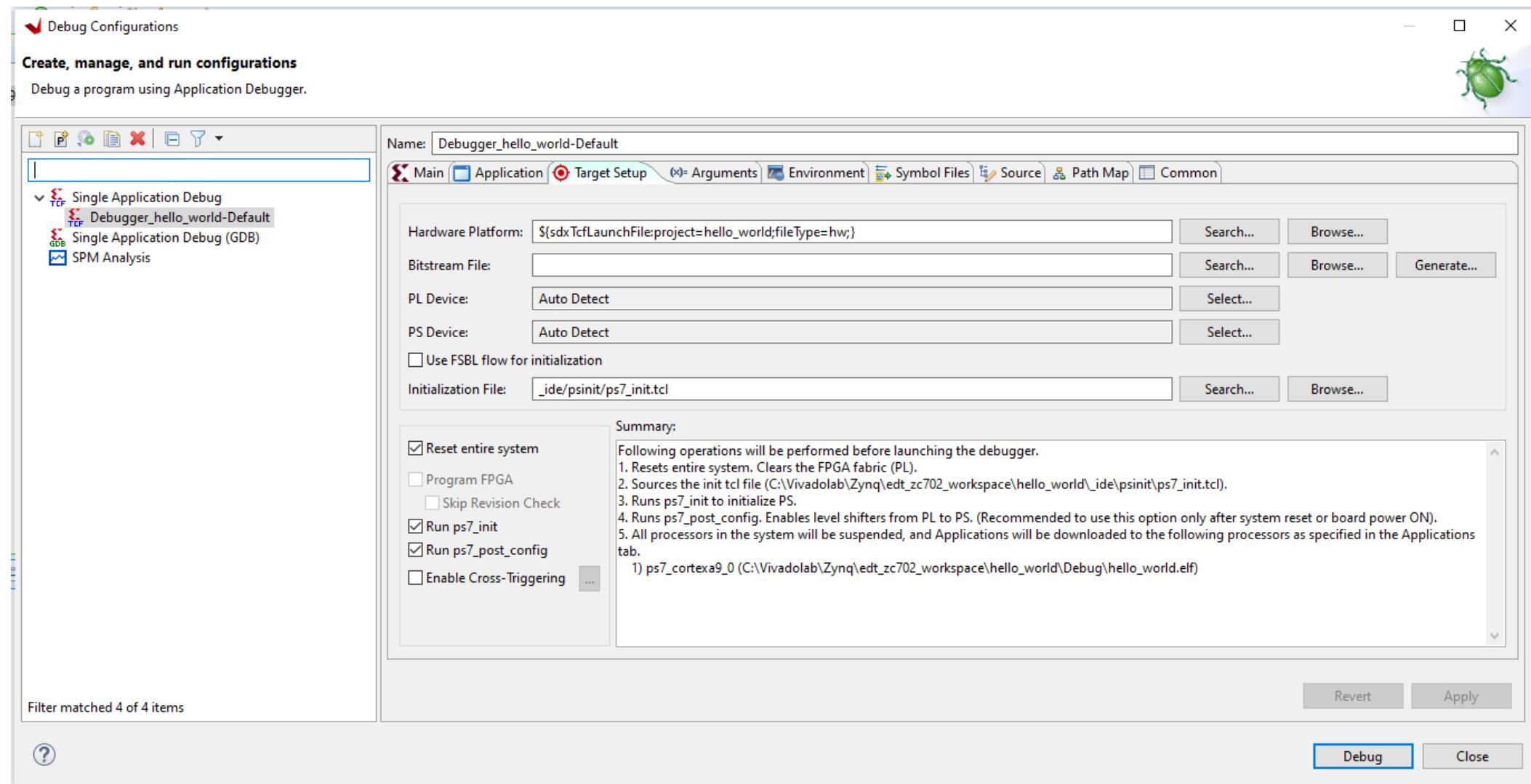
Running the Hello World Application on a ZC702 Board



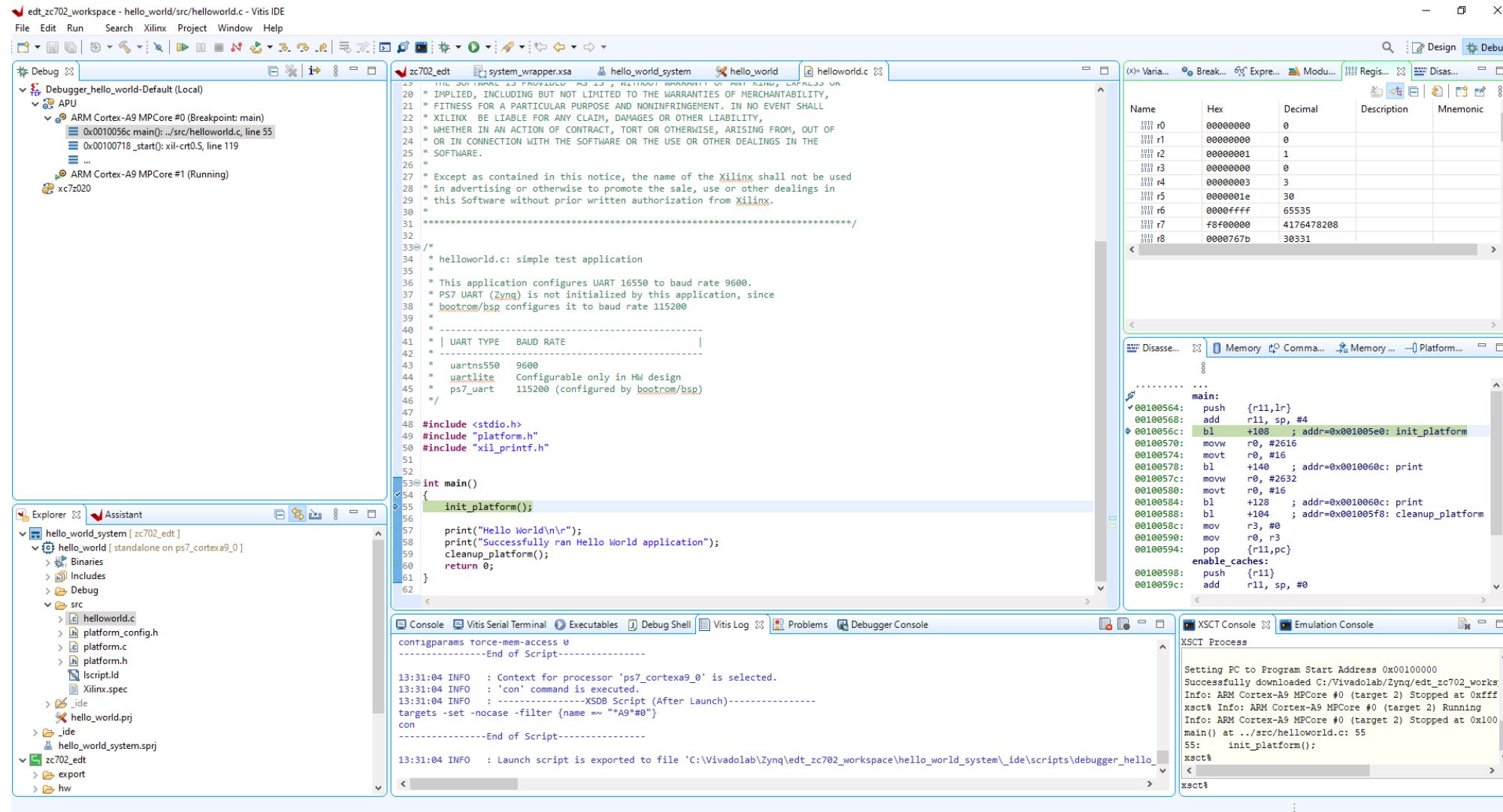
Debugging Standalone Applications in the Vitis IDE



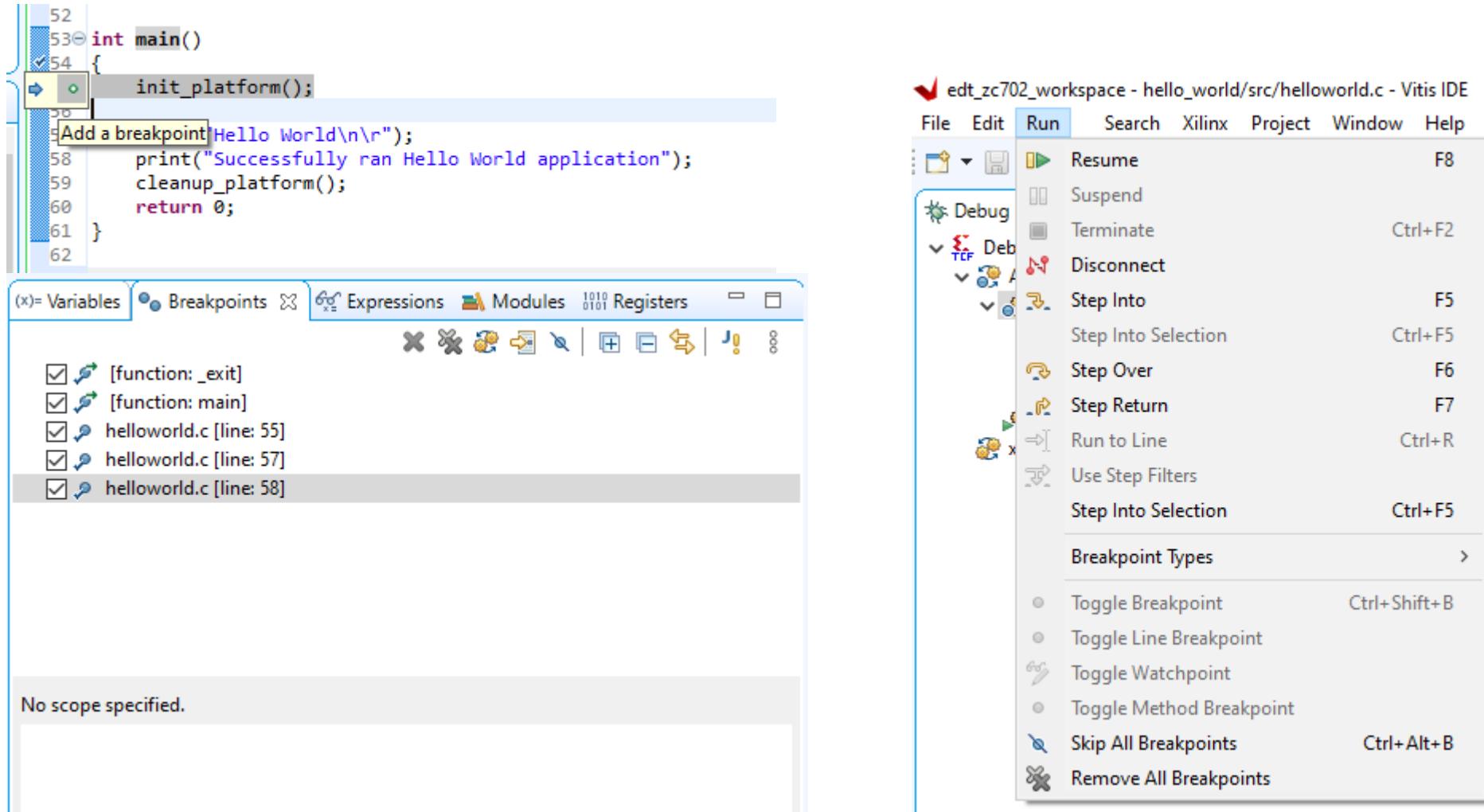
Debugging Standalone Applications in the Vitis IDE



Debugging Standalone Applications in the Vitis IDE



Debugging Standalone Applications in the Vitis IDE

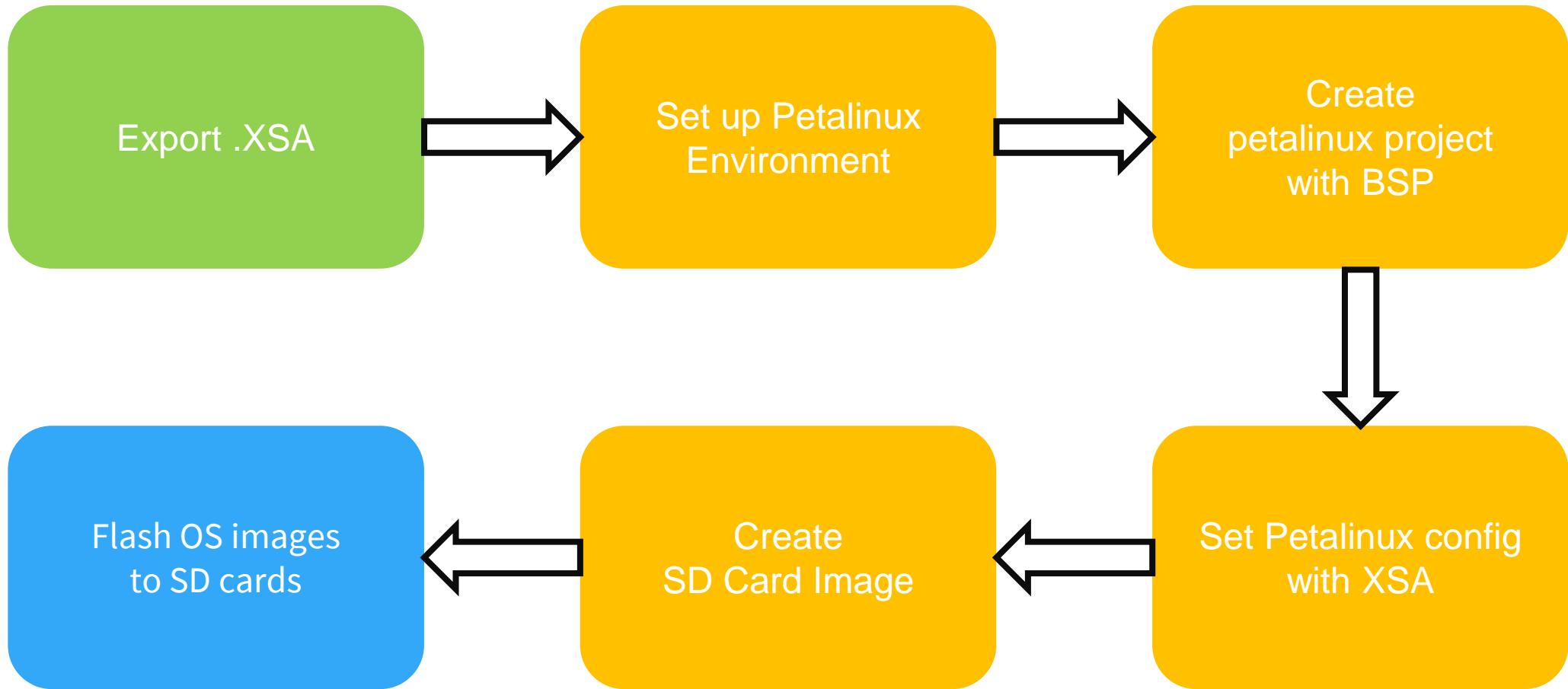




Building and Debugging Linux Applications

Course Agenda
2023

Build Petalinux



Install Petalinux



Products Solutions Resources & Support



[Home](#) / Adaptive Computing Support / Downloads

Downloads

Licensing Help

NIC Software & Drivers

Vivado (HW
Developer)

Vitis (SW
Developer)

Vitis Embedded
Platforms

Power Design
Manager

Alveo Packages

PetaLinux

Device Models

Documentation
Navigator

Version

2023.1

2022.2

2022.1

Archive

PetaLinux Tools - Installer - 2023.1 Full Product Installation

Important Information

The PetaLinux Tools installer is downloaded using the below link. The installer checks for the required host machine package requirements followed by license acceptance from the user. It can be installed in any desired path. Note: All BSPs (located below) require the PetaLinux Tools to be installed first.

[PetaLinux Installer \(TAR/GZIP - 3.18 GB\)](#)

MD5 SUM Value : 78fd08837e2d30541190a7ff20988e0f

Download Type

Full Product Installation

Last Updated

May 17, 2023

Answers

[Release Notes and Known Issues](#)

Documentation

[PetaLinux Tools Documentation](#)
[Installer Information](#)
[What's new in Embedded Software](#)



Install Petalinux

000035006 - PetaLinux 2023.1 - Product Update Release Notes and Known Issues

Aug 24, 2023 • Knowledge

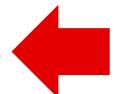
TITLE

000035006 - PetaLinux 2023.1 - Product Update Release Notes and Known Issues

DESCRIPTION

This Answer Record acts as the release notes for PetaLinux 2023.1 and contains links to information about resolved issues and updated collateral contained in this release.

Please find the installation requirements specified in our user guide [here](#).



The packages required for installation per OS can also be found in the attached .xlsx (found at the bottom of this page)

SOLUTION

BSPs supported for the PetaLinux 2023.1 Release

This table contains supported BSPs for Zynq 7000, MicroBlaze, Zynq UltraScale+, MPSoC, and Versal available on the Embedded Development download page.

SNO	Platform	Variant	BSP Name	MicroBlaze	Zynq UltraScale+	MPSoC	Versal	Available on the Embedded Development download page	Linux
1	MicroBlaze	AC701	xilinx-ac701-050802	6	Zynq 7000	ZC702		xilinx-zc702-v2023.1-05080224.bsp	<p>This BSP contains:</p> <ul style="list-style-type: none">Hardware: This design uses Vivado board presets with Zynq-7000 PS block (DDR, UART, SD, QSPI, Ethernet etc.) and AXI GPIO connected with led_4bits.Software: FSBL, U-Boot, Linux, device-tree (includes open-amp), rootfs (minimal packages).Pre-built Images: Ready to test images bitstream, FSBL, U-Boot, Linux and rootfs for booting U-Boot and Linux.
									<p>This BSP contains:</p> <ul style="list-style-type: none">Hardware: This design uses Vivado board presets with Zynq-7000 PS block (DDR, UART, SD, QSPI, Ethernet etc.) and AXI GPIO connected with led_4bits.

Install Petalinux

PetaLinux Tools Documentation: Reference Guide (UG1144) UG1144 2023-05-16 2023.1 English

Search in document

Keywords

+ Overview
– Setting Up Your Environment
 – Installation Steps
 • Installation Requirements
 • Prerequisites
 – Installing the PetaLinux Tool
 • Installing a Preferred eSDK as part of the PetaLinux Tool
 • Troubleshooting
+ PetaLinux Working Environment Setup
• Design Flow Overview
+ Creating a Project
+ Configuring and Building
+ Packaging and Booting
+ Upgrading the Workspace
+ Customizing the Project
+ Customizing the Root File System
+ Debugging
+ Advanced Configurations
+ Yocto Features
+ Technical FAQs
+ Migration

Setting Up Your Environment

Installation Steps

Installation Requirements

The PetaLinux tools installation requirements are:

- Minimum workstation requirements:
 - 8 GB RAM (recommended minimum for AMD tools)
 - 2 GHz CPU clock or equivalent (minimum of eight cores)
 - 100 GB free HDD space
 - Supported OS:
 - Completely removed RHEL and CENTOS to align with upstream Yocto.
 - **Ubuntu** Desktop/Server 18.04.1 LTS, 18.04.2 LTS, 18.04.3 LTS, 18.04.4 LTS, 18.04.5 LTS, 18.04.06 LTS, 20.04 LTS, 20.04.1 LTS, 20.04.2 LTS, 20.04.3 LTS, 20.04.4 LTS, 20.04.5 LTS(64-bit), 22.04 LTS and 22.04.1 LTS
 - OpenSuse Leap 15.3
 - AlmaLinux 8.7
- You need access to install the required packages mentioned in the release notes. The PetaLinux tools need to be installed as non-root user.
- PetaLinux requires some standard development tools and libraries to be installed on your Linux host workstation. Install the libraries and tools listed in the [release notes](#) on the host Linux.
- PetaLinux tools require that your host system `/bin/sh` is 'bash'. If you use **Ubuntu** distribution and your `/bin/sh` is 'dash', consult your system administrator to change your default system shell `/bin/sh` with the `sudo dpkg-reconfigure dash` command.

Note: For package versions, refer to the [PetaLinux 2023.1 Release Notes](#) and Master Answer Record: [000035006](#).

CAUTION! Consult your system administrator if you are unsure about the correct host system package management procedures.

Important: PetaLinux 2023.1 works only with hardware designs exported from AMD Vivado™ Design Suite 2023.1

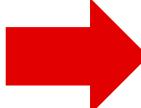
Install Petalinux

Installation Requirements

<command> sudo apt-get install gawk wget git-core diffstat unzip <Tool/Library>

Linux	SSW_AIE	000034487 - 2022.2 Versal : Running AIE2 with PL controller test on v70 causes PetaLinux AIE driver crash	2023.1
Linux	SSW_Drivers	000034517 - 2022.2 Zynq UltraScale+ MPSoC : Suspend/Resume USB wakeup test fails when USB devices connected with USB3.0 Hub	2023.1
Linux	SSW_Hypervisor	000034520 - 2022.2 Zynq UltraScale+ MPSoC : Unable to create DOMU image for VEK280-es1,VPK120, VPK180 and VHk158	2023.1
Linux	SSW_EMBEDDED_Linux	000034519 - 2022.2 Versal: VPK180 Hangs with kernel panic if writing or copying larger than 2.1 GB file	2023.1

Embedded Linux Processor System Design And AXI Embedded Systems PetaLinux Knowledge Base



Files (2)			Download
FILE NAME	SIZE	ACTION	
README_content_v2022_2.txt	1.7 KB	<input type="button" value="▼"/>	
2022.2_Petalinux_Package_List.xlsx	28.03 KB	<input type="button" value="▼"/>	

Community Feedback?

Install Petalinux

Installation zlib1g:i386

```
<command> sudo dpkg --add-architecture i386
```

```
<command> sudo apt-get update
```

```
<command> sudo apt-get install zlib1g:i386
```

```
devin@LAPTOP-90IB0783:$ sudo dpkg --add-architecture i386
devin@LAPTOP-90IB0783:$ sudo apt-get update
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
```

```
Get:21 http://archive.ubuntu.com/ubuntu jammy-backports/main i386 Packages [33.9 kB]
Get:22 http://archive.ubuntu.com/ubuntu jammy-backports/universe i386 Packages [13.4 kB]
Fetched 13.9 MB in 5s (2538 kB/s)
Reading package lists... Done
devin@LAPTOP-90IB0783:$ sudo apt-get install zlib1g:i386
```

Install Petalinux

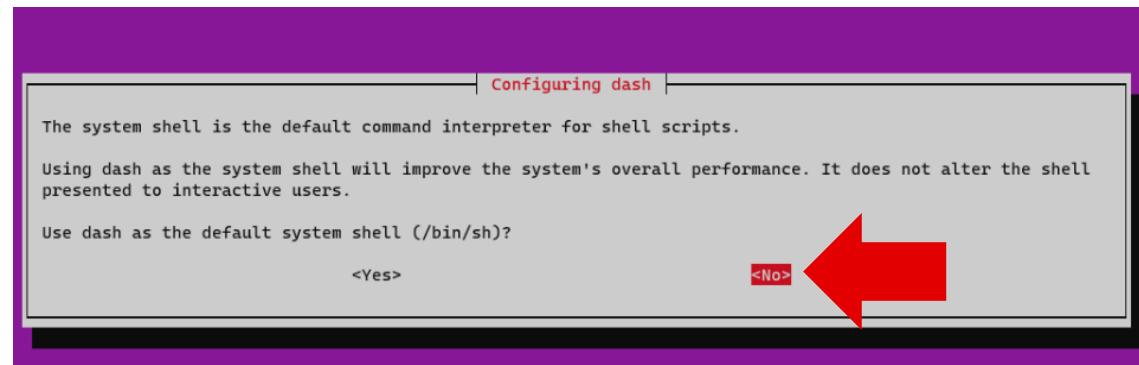
Installation petalinux 2022.2

```
devin@LAPTOP-90IB0783:/opt/pkg/petalinux/2022.2$ ls  
petalinux-v2022.2-10141622-installer.run
```

<command> chmod 755 ./petalinux-v<petalinux-version>-final-installer.run
./petalinux-v<petalinux-version>-final-installer.run

```
devin@LAPTOP-90IB0783:/opt/pkg/petalinux/2022.2$ chmod 755 ./petalinux-v2022.2-10141622-installer.run  
./petalinux-v2022.2-10141622-installer.run
```

<command> sudo dpkg-reconfigure dash



Build Petalinux Project

Setting Up Your Environment

```
<command> source <Petalinux Install Path>/settings.sh
```

Create petalinux project

Already have BSP file

```
petalinux-create -t project -s /<BSP Path>/***.bsp -n proj_name
```

```
components config.project hardware pre-built project-spec README README.hw
```

Do not have BSP file

```
<command> petalinux-create --type project --template <PLATFORM> --name <PROJECT_NAME>
```

--template <PLATFORM> - The following platform types are supported:

zynqMP (for Zynq UltraScale+ MPSoC)

zynq (for Zynq-7000 devices)

microblaze (for MicroBlaze™ processor)

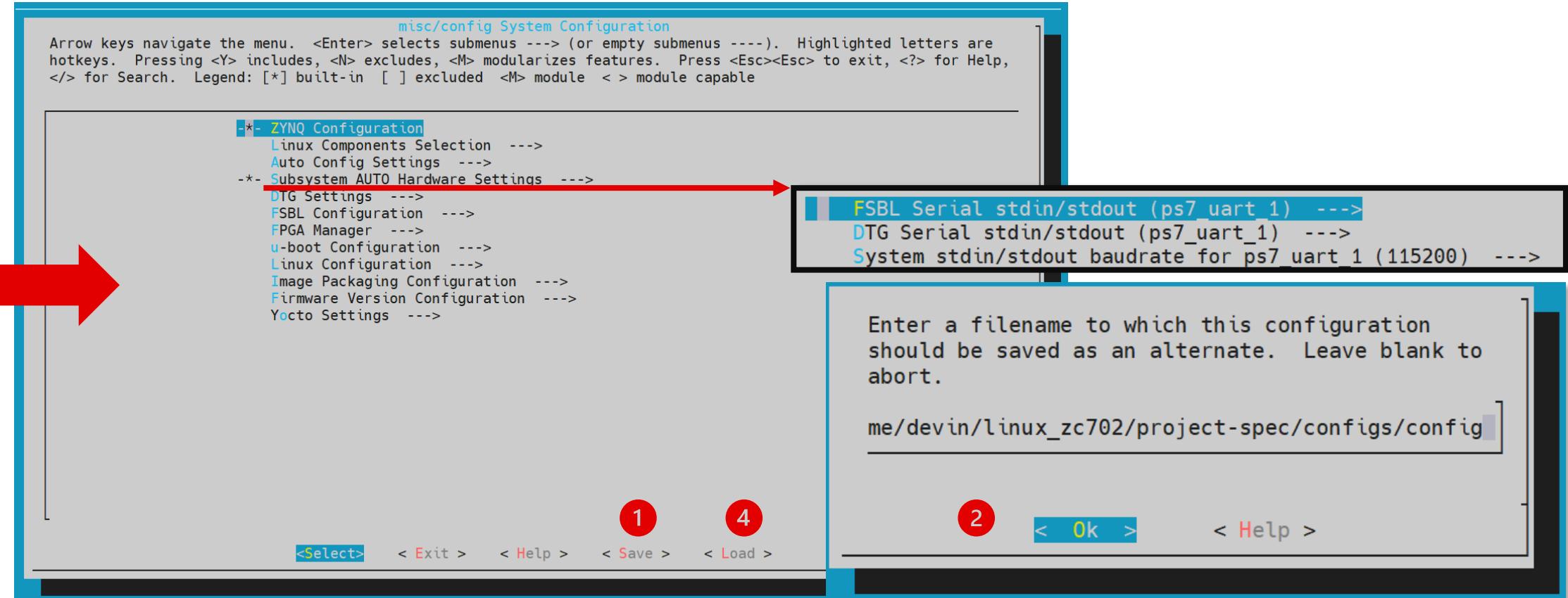
Build Petalinux Project

Import XSA

```
<command> cd <Project Directory>
```

```
<command> petalinux-config --get-hw-description <XSA Directory>
```

③ <command> petalinux-config



Build Petalinux Project

Build Petalinux Image

<command> petalinux-build

boot.scr	pxelinux.cfg	rootfs.cpio.gz.u-boot	rootfs.manifest	system.dtb	u-boot-dtb.elf	vmlinux
config	rootfs.cpio	rootfs.ext4	rootfs.tar.gz	u-boot.bin	u-boot.elf	zImage
image.ub	rootfs.cpio.gz	rootfs.jffs2	system.bit	u-boot-dtb.bin	uImage	zynq_fsbl.elf

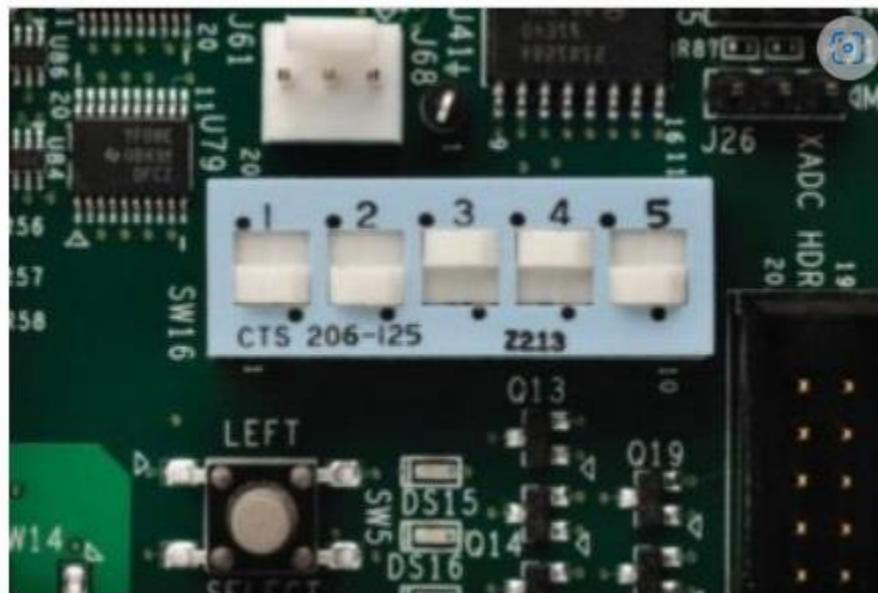
Package Boot.Bin

<command> petalinux-package --boot --fsbl ./zynq_fsbl.elf –fpga ./system.bit –u-boot ./u-boot.elf --force

BOOT.BIN	pxelinux.cfg	rootfs.jffs2	u-boot.bin	vmlinux
bootgen.bif	rootfs.cpio	rootfs.manifest	u-boot-dtb.bin	zImage
boot.scr	rootfs.cpio.gz	rootfs.tar.gz	u-boot-dtb.elf	zynq_fsbl.elf 1
config	rootfs.cpio.gz.u-boot	system.bit 2	u-boot.elf 3	
image.ub	rootfs.ext4	system.dtb	uImage	

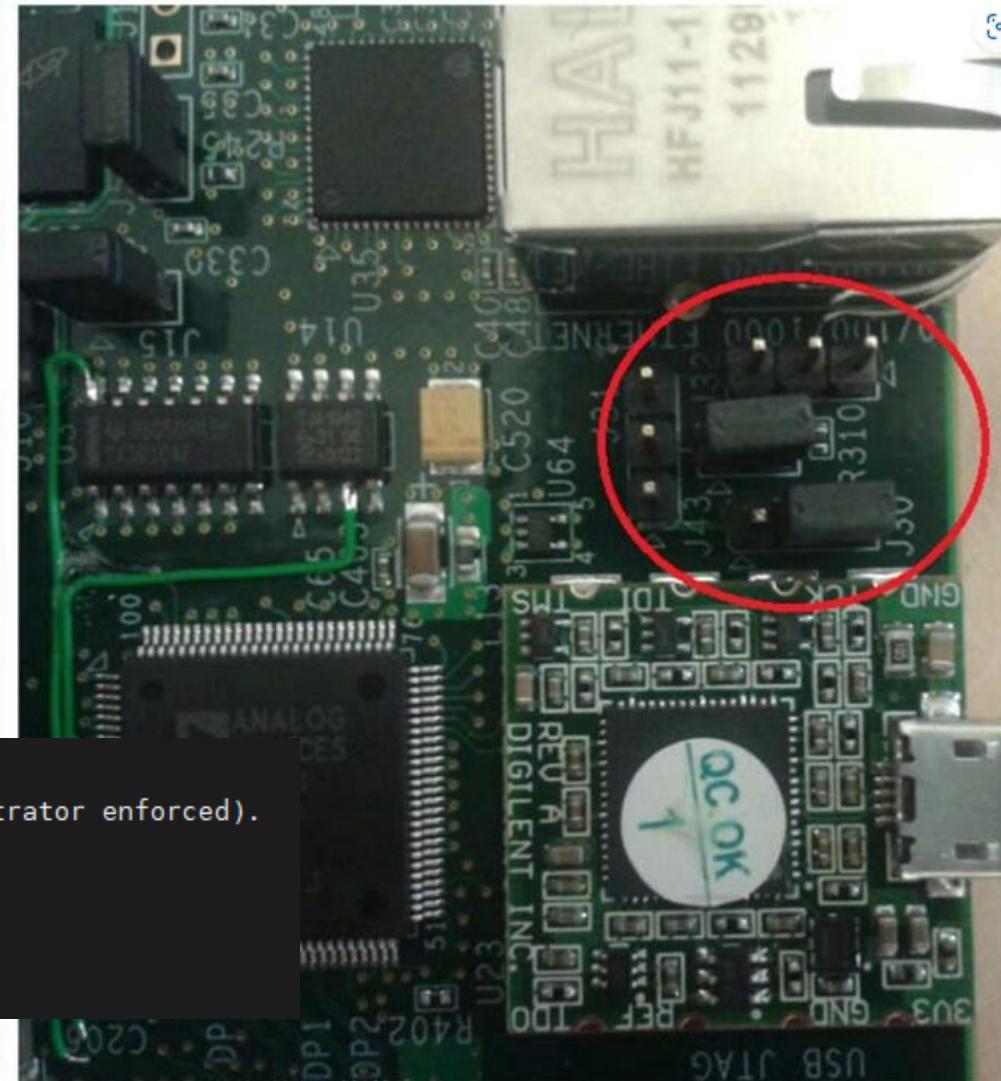
Copy files into SD Card

Build Petalinux Project



SD Boot Mode Setup for SW16

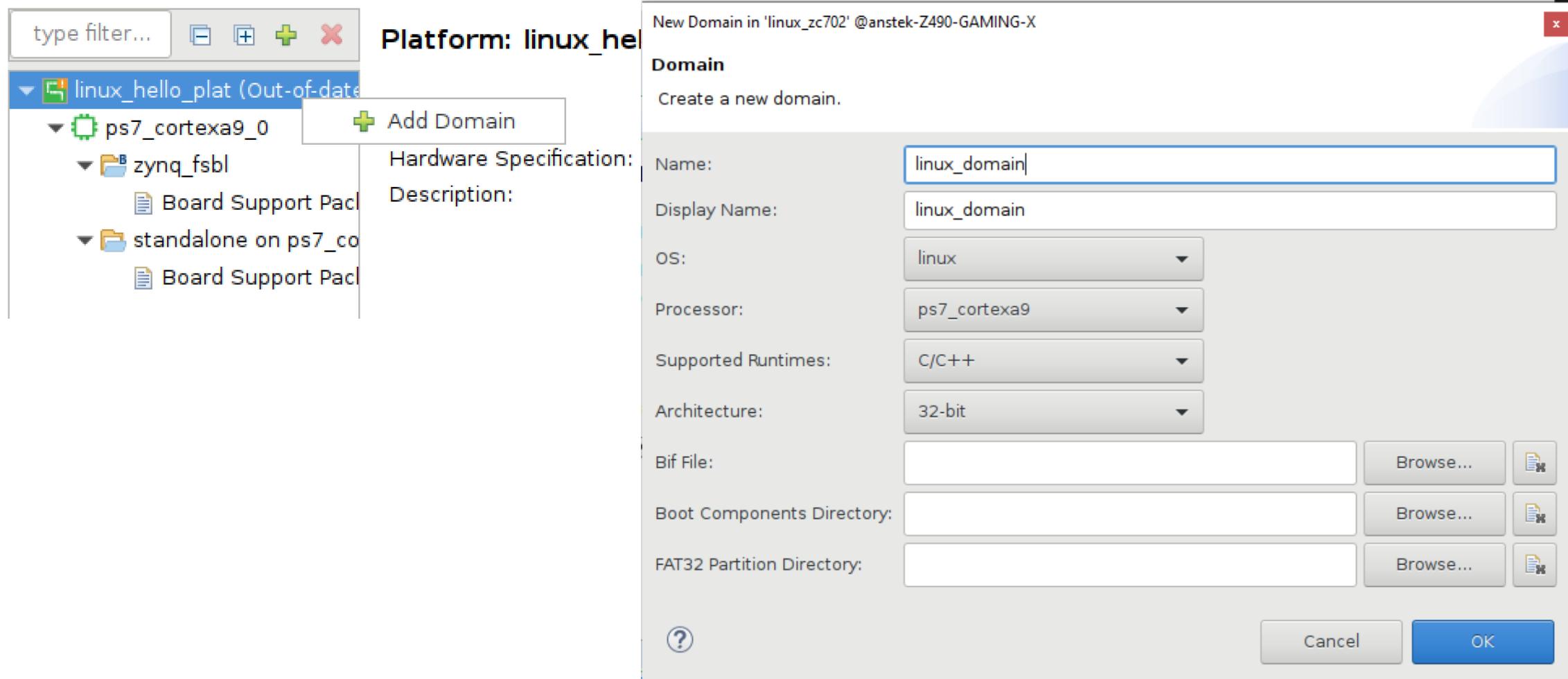
Make sure Ethernet Jumper J30 and J43 are as shown in the following figure.



Ethernet Jumper

© Copyright 2022 AMD

Build Linux Application



Build Linux Application

New Application Project

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

Application project name: linux_hello

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

Select a system project

System project details
System project name: linux_hello_system

Target processor
Select target processor for the Application project.

Processor	Associated applications
ps7_cortexa9_0	
ps7_cortexa9 SMP	linux_hello
ps7_cortexa9_1	

Show all processors in the hardware specification

Linux Kernel in SMP

APU

Arm Cortex-A53 Arm Cortex-A53 Arm Cortex-A53 Arm Cortex-A53

< Back Next > Finish Cancel

Build Linux Application

The screenshot shows two windows side-by-side during the 'New Application Project' process.

Left Window: Domain Selection

New Application Project @ansteck-Z490-GAMING-X

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

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

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

Select a domain

- linux_domain
- + Create new...

Domain details

Name: linux_ps7_cortexa9
Display Name: linux_ps7_cortexa9
Operating System: linux
Processor: ps7_cortexa9
Architecture: 32-bit

Application settings

Sysroot path: Browse...
Root FS: Browse...
Kernel Image: Browse...

Buttons: ? < Back Next > Cancel Finish

Right Window: Template Selection

New Application Project

Templates
Select a template to create your project.

Available Templates:

Find: [x] [+] [?]

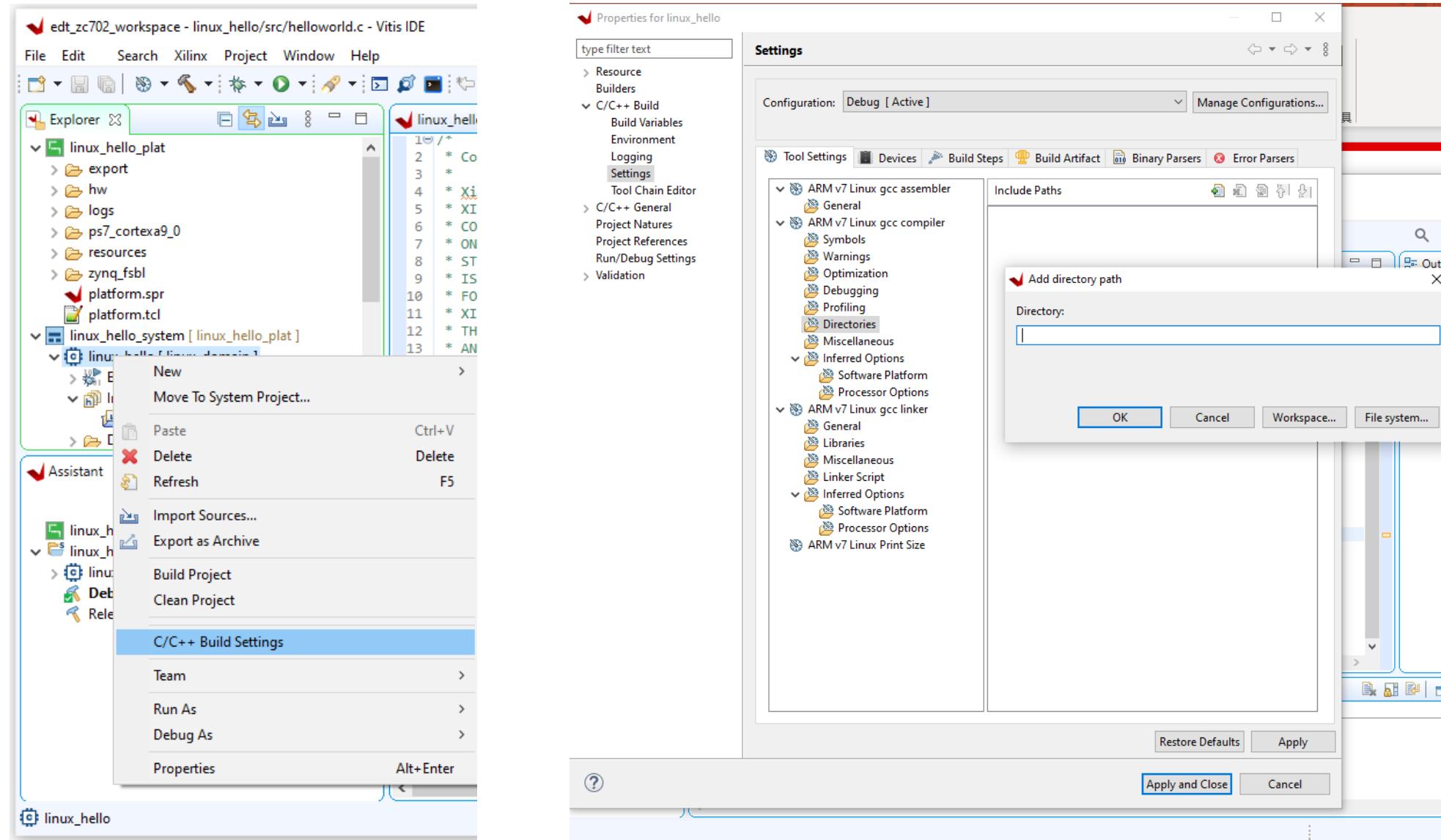
Embedded software development templates

- Empty Application (C++)
- Linux Empty Application
- Linux Hello World

Preview Area

Linux Hello World
Let's say 'Hello World' in C.

Build Linux Application



Build Linux Application

```
xilinx-zc702-2022_2:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 00:0A:35:00:1E:53
          UP BROADCAST MULTICAST MTU:1500 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)
          Interrupt:35 Base address:0xb000

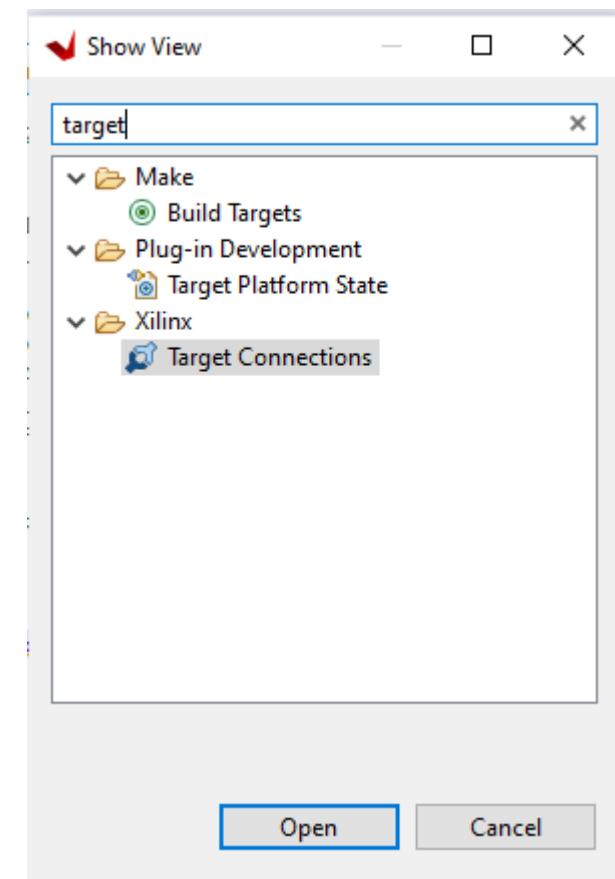
lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:65536 Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:140 (140.0 B) TX bytes:140 (140.0 B)

xilinx-zc702-2022_2:~$ macb e000b000.ethernet eth0: Link is Up - 1Gbps/Full - flow control off
IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready

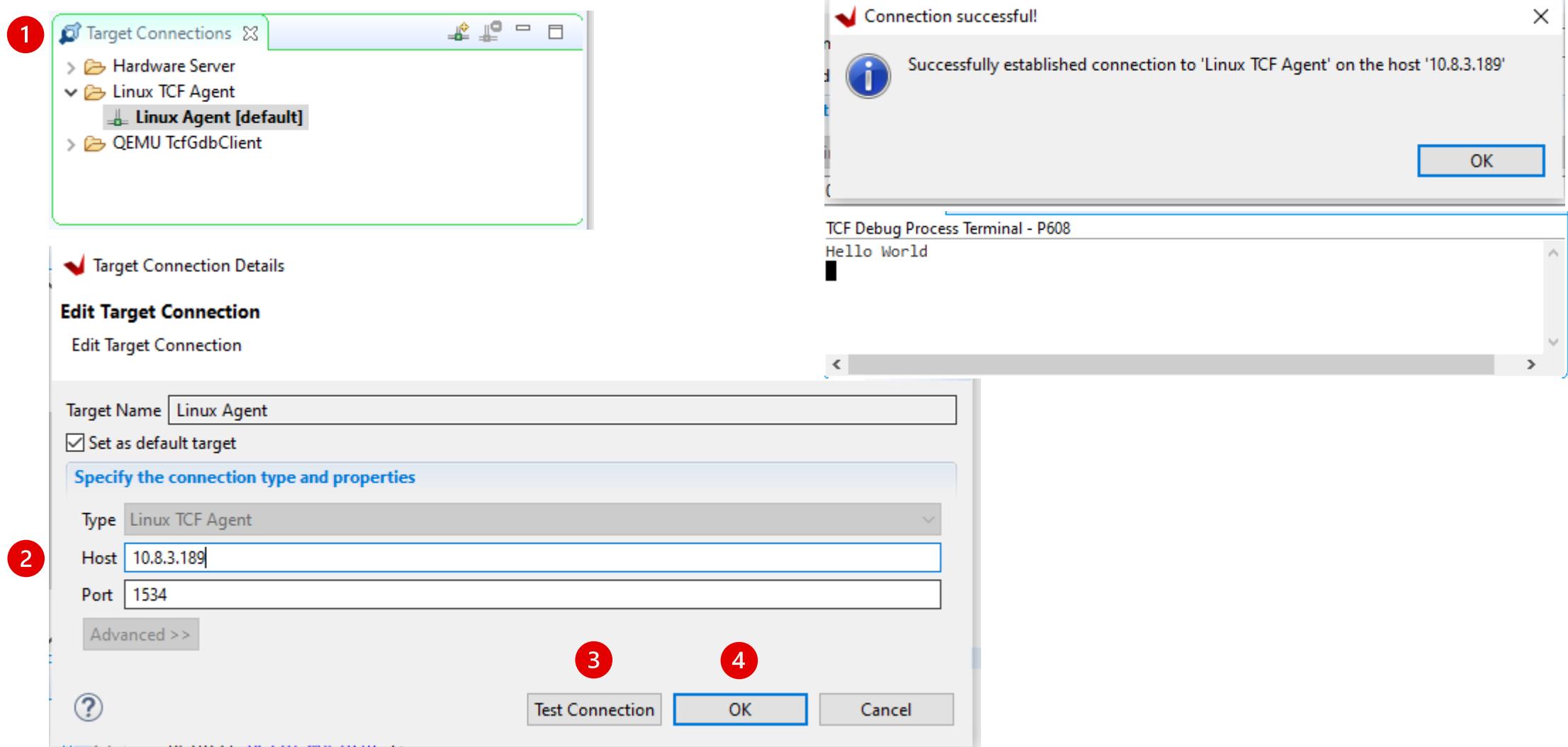
xilinx-zc702-2022_2:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 00:0A:35:00:1E:53
          inet addr:10.8.3.189 Bcast:10.8.3.255 Mask:255.255.255.0
          inet6 addr: fe80::20a:3ff:fe00:1e53/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:11 errors:0 dropped:8 overruns:0 frame:0
          TX packets:11 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7566 (7.3 KiB) TX bytes:1872 (1.8 KiB)
          Interrupt:35 Base address:0xb000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:65536 Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:140 (140.0 B) TX bytes:140 (140.0 B)

xilinx-zc702-2022_2:~$
```



Build Linux Application

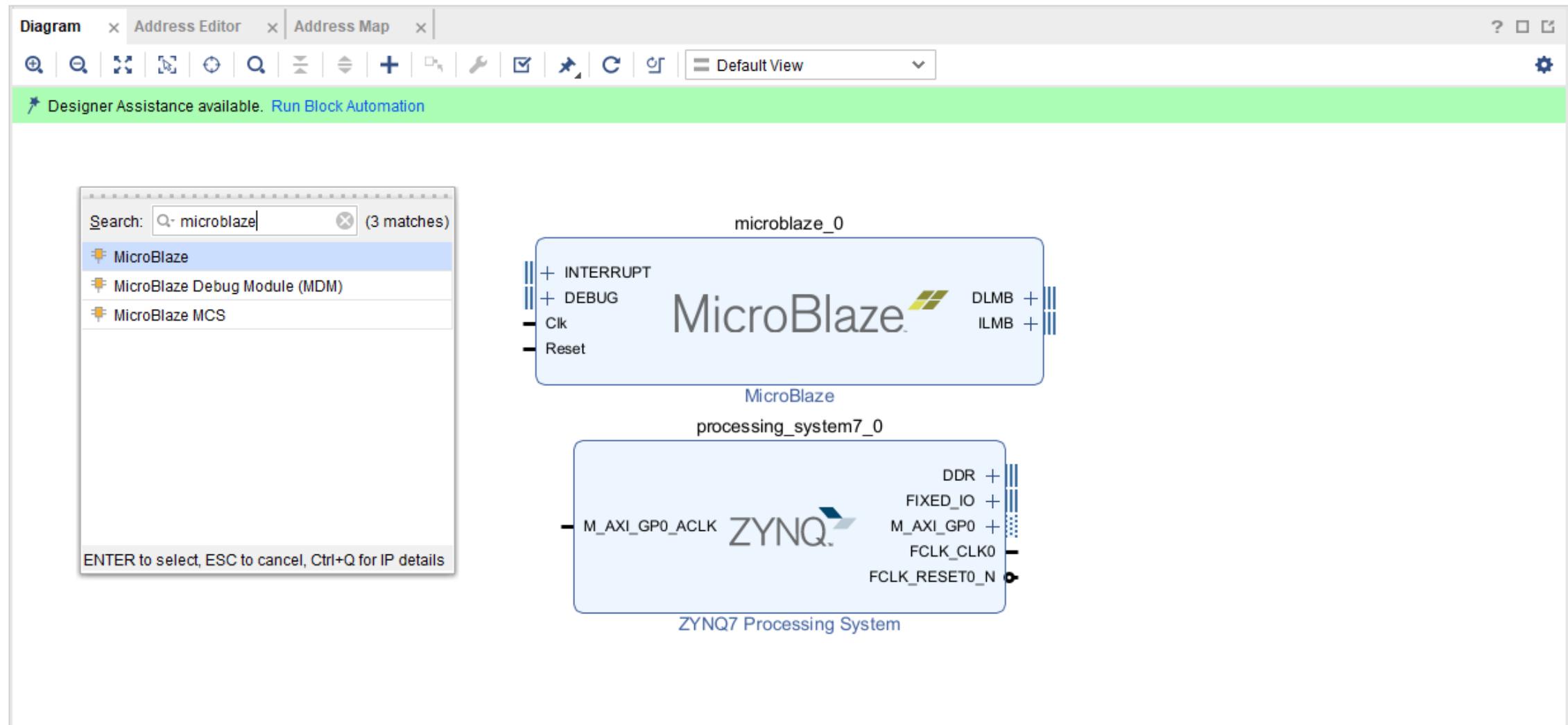




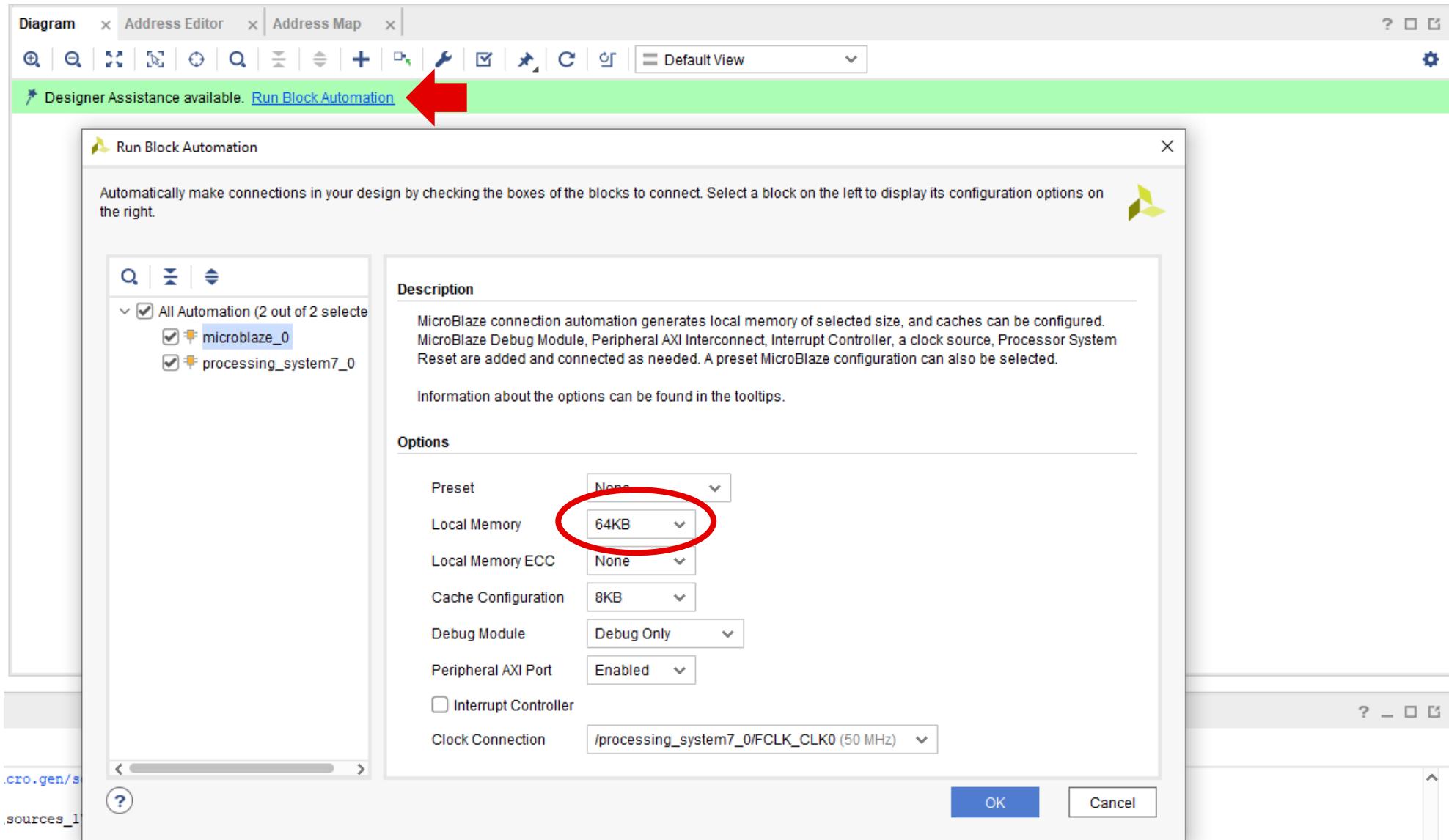
Design Microblaze via Vitis

Course Agenda
2023

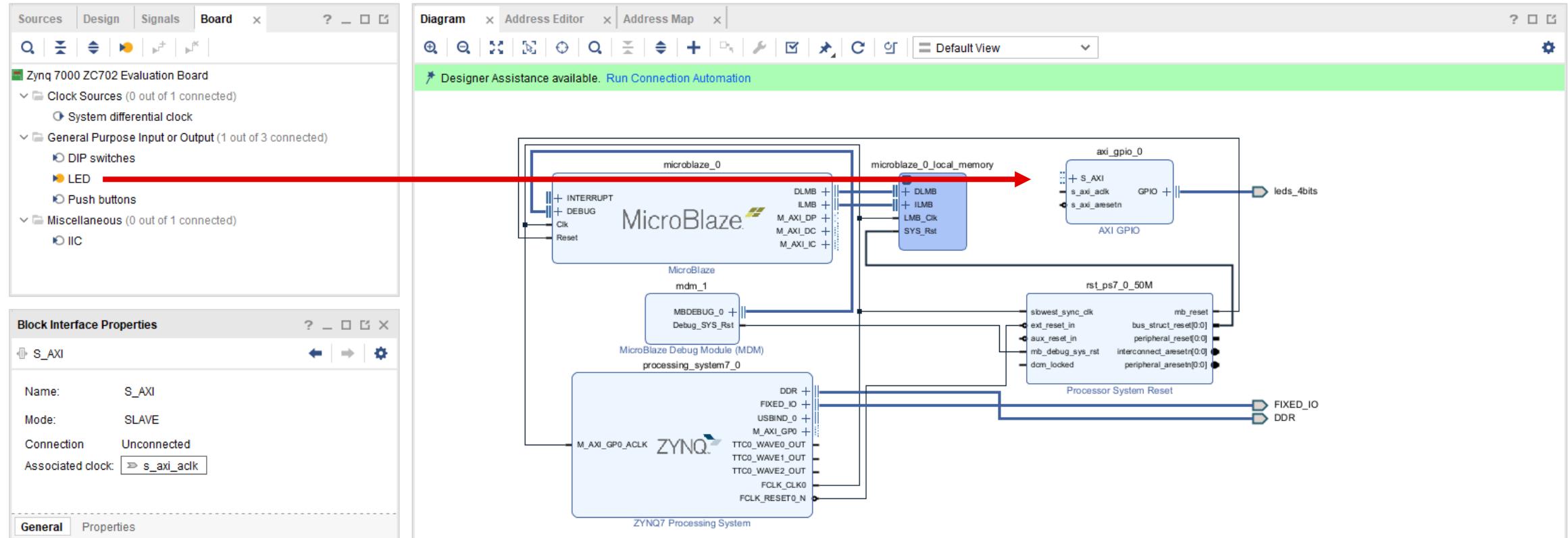
Microblaze Block Design



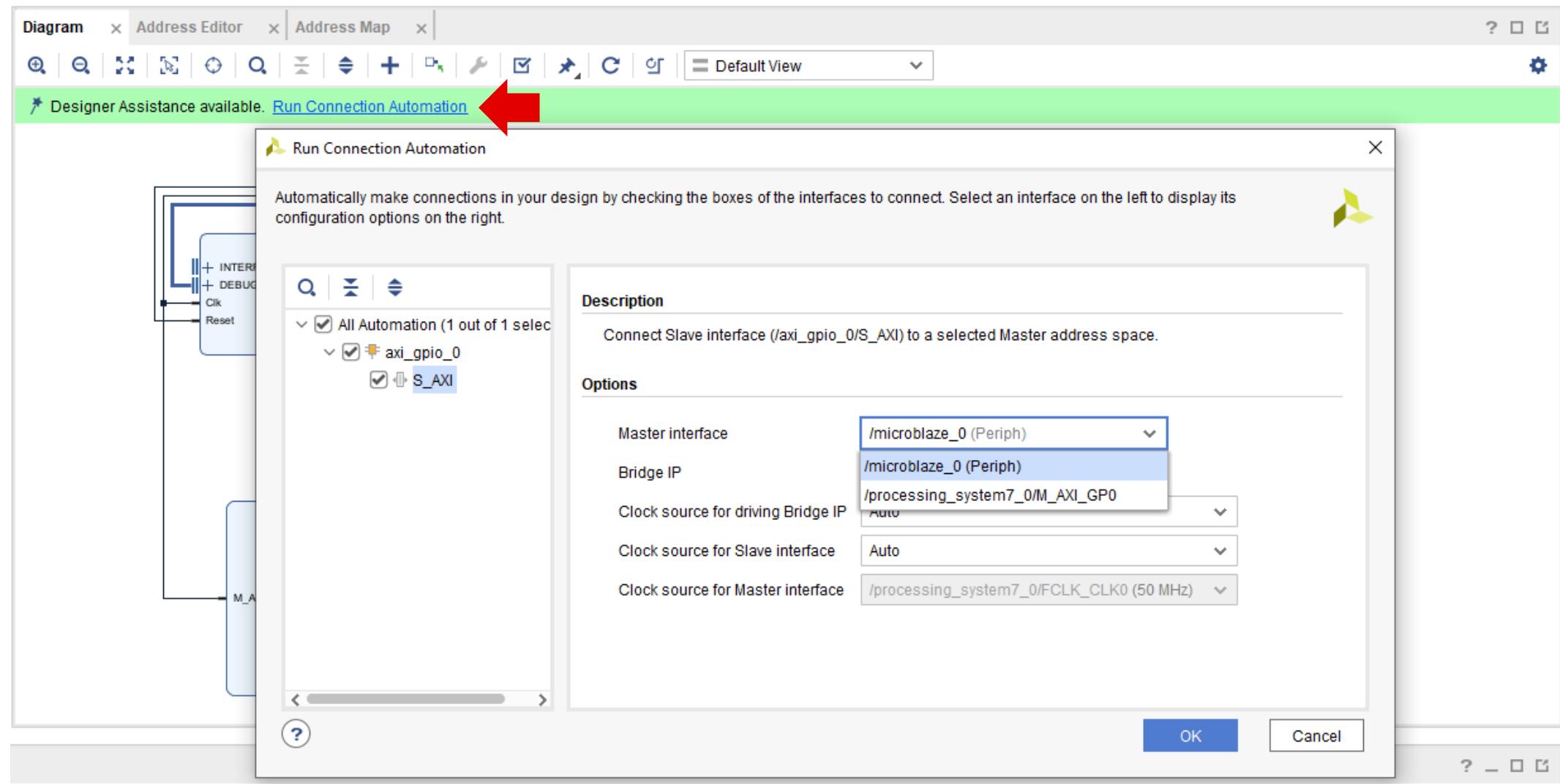
Microblaze Block Design



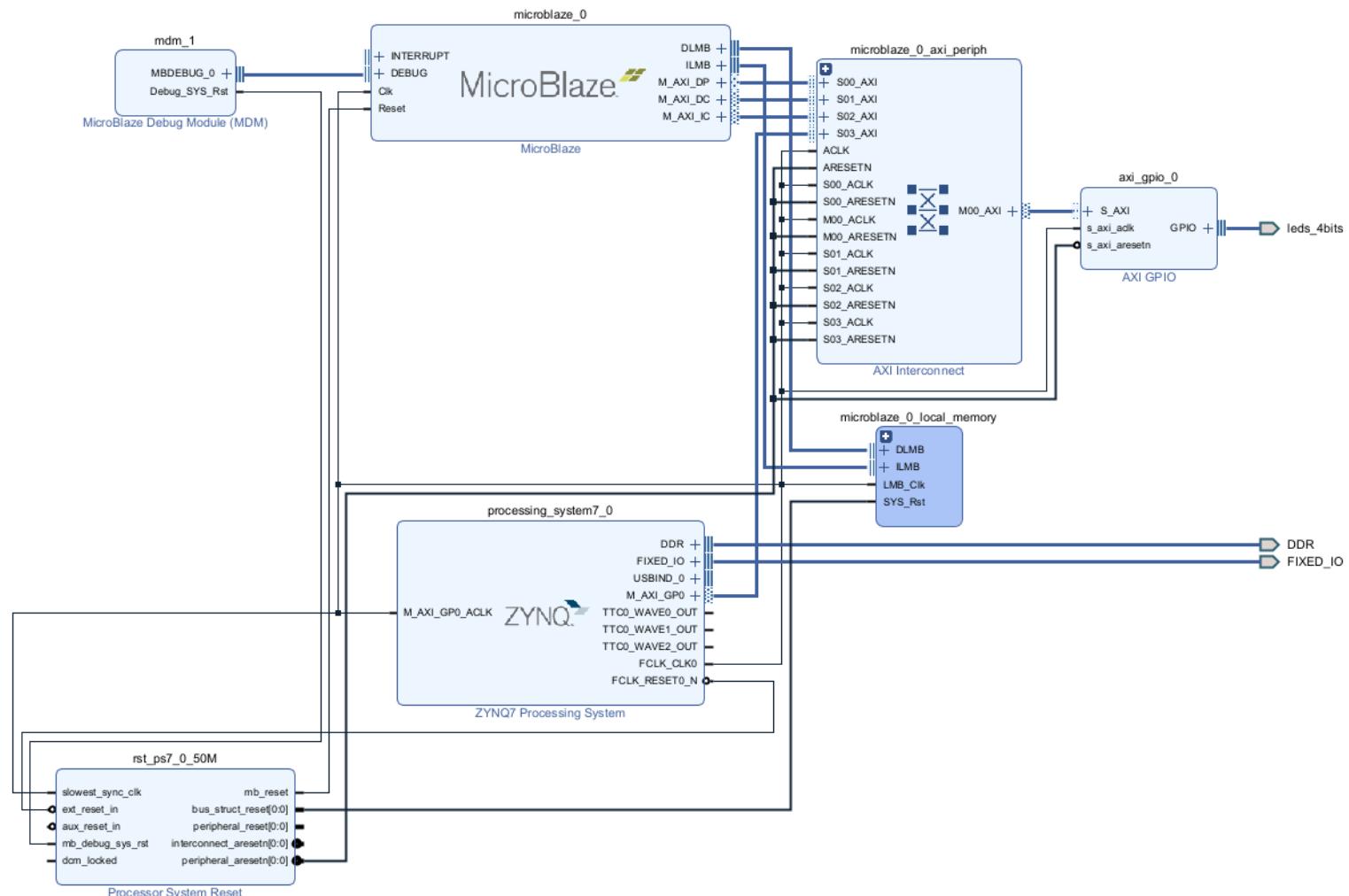
Microblaze Block Design



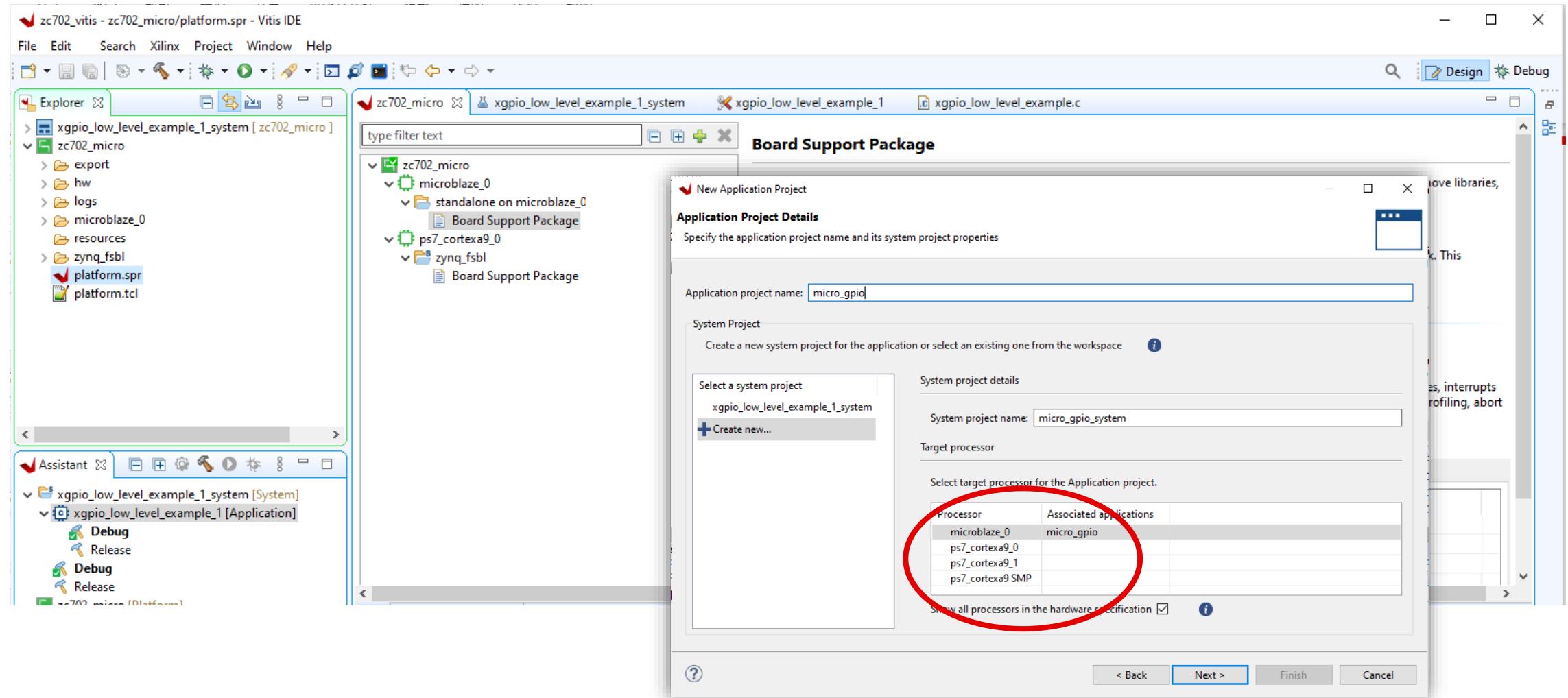
Microblaze Block Design



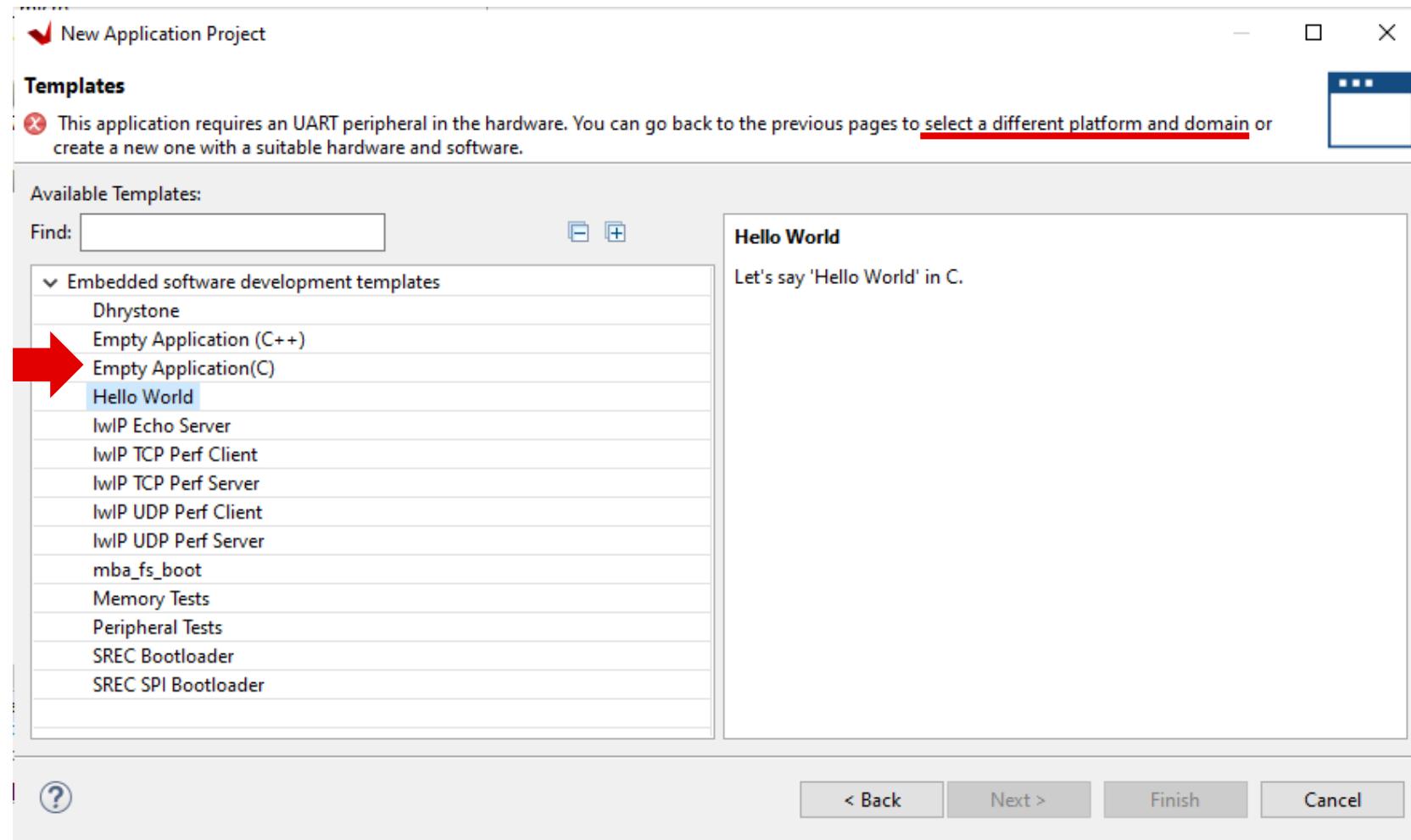
Microblaze Block Design



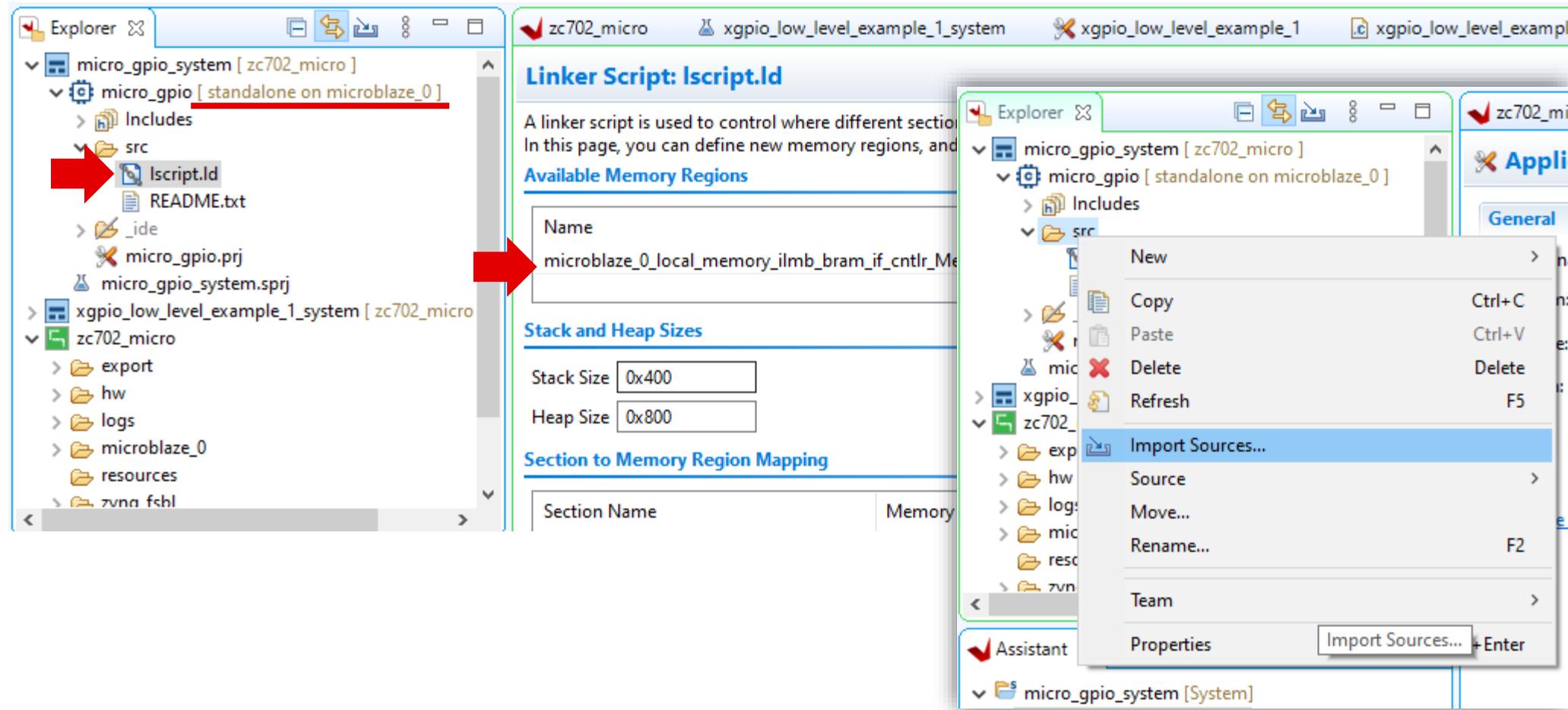
Microblaze in Vitis



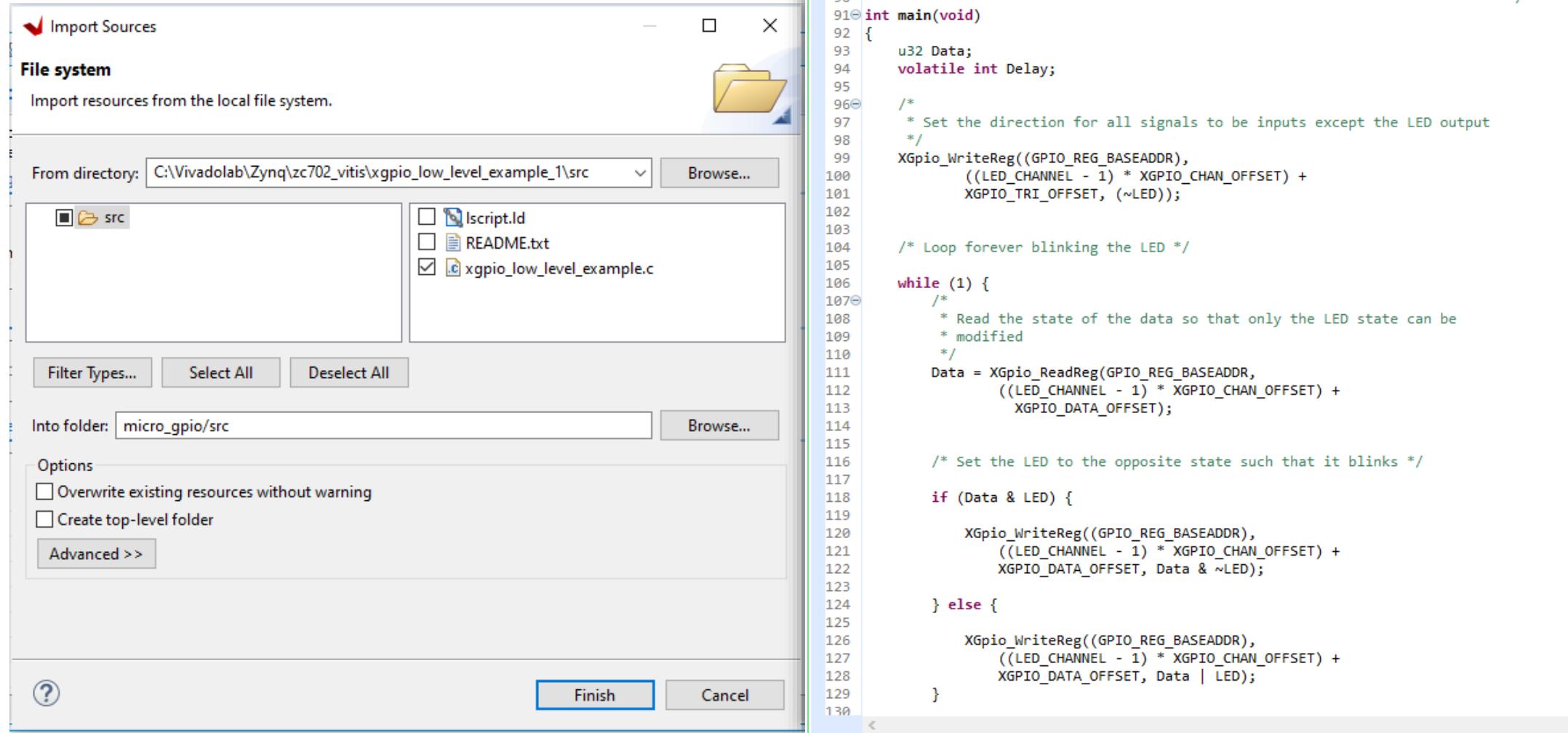
Microblaze in Vitis



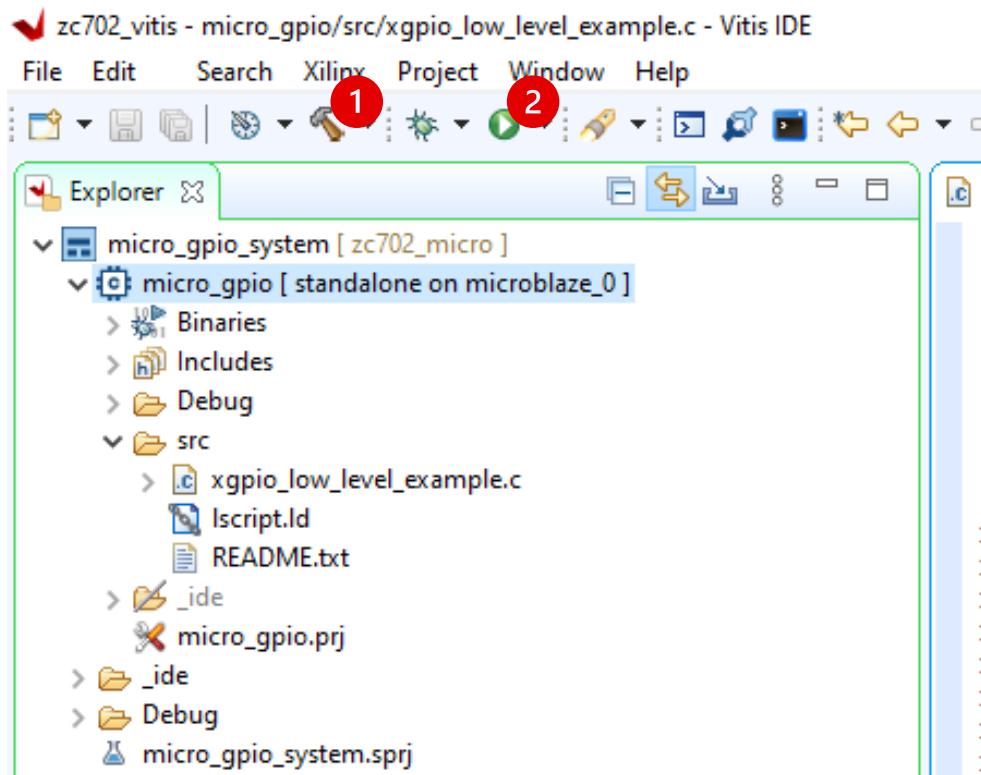
Microblaze in Vitis



Microblaze in Vitis



Microblaze in Vitis





Using the PS GP Port in Zynq Devices

Course Agenda
2023

What is GPIO?

- ▶ General Perpose I/O
 - GPIO peripheral provides software with observation and control of up to 54 device pins via the MIO module.
- ▶ GPIO Can be used to Control or Monitor signal
 - Each GPIO is independently and dynamically programmed as input, output, or interrupt sensing.
 - Software can read all GPIO values within a bank using a single load instruction, or write data to one or more GPIOs
 - The GPIO control and status registers are memory mapped at base address 0xE000_A000.

What is GPIO?

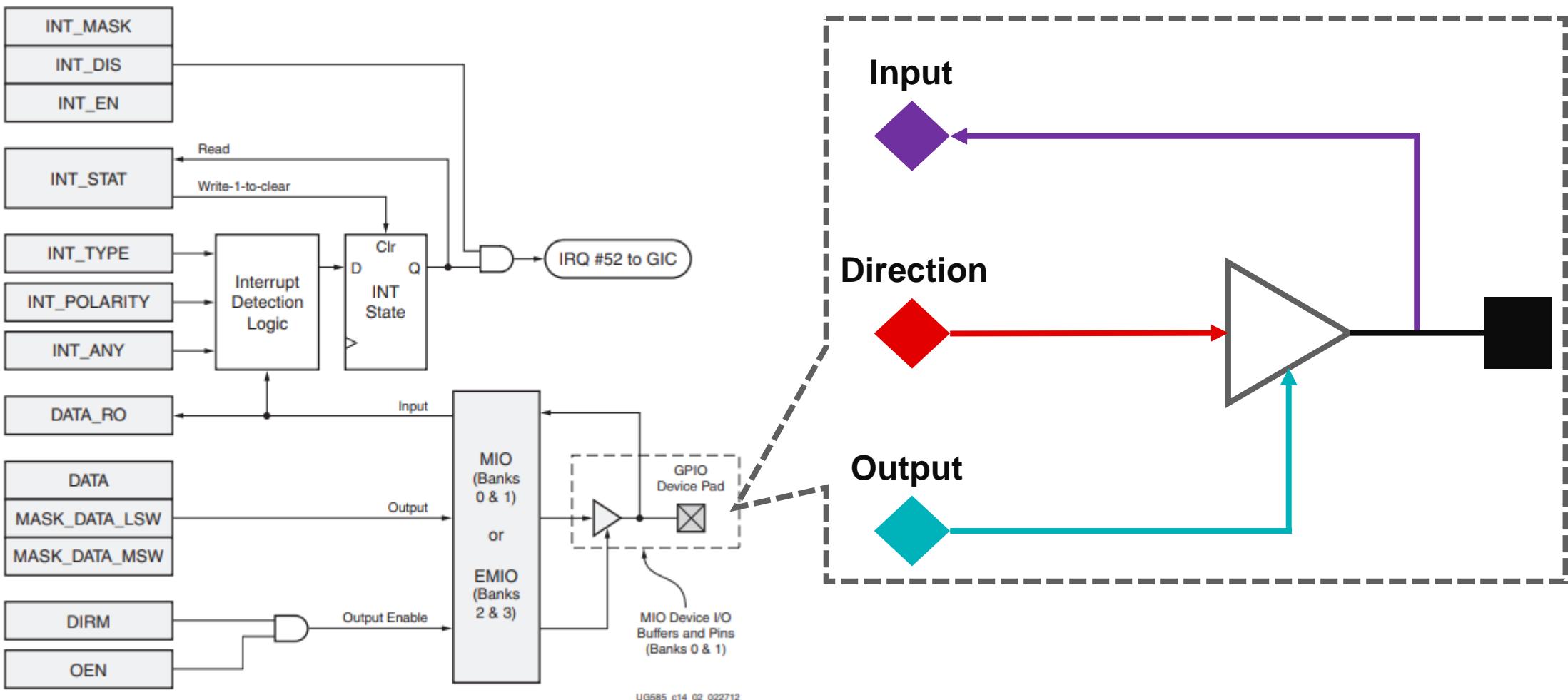
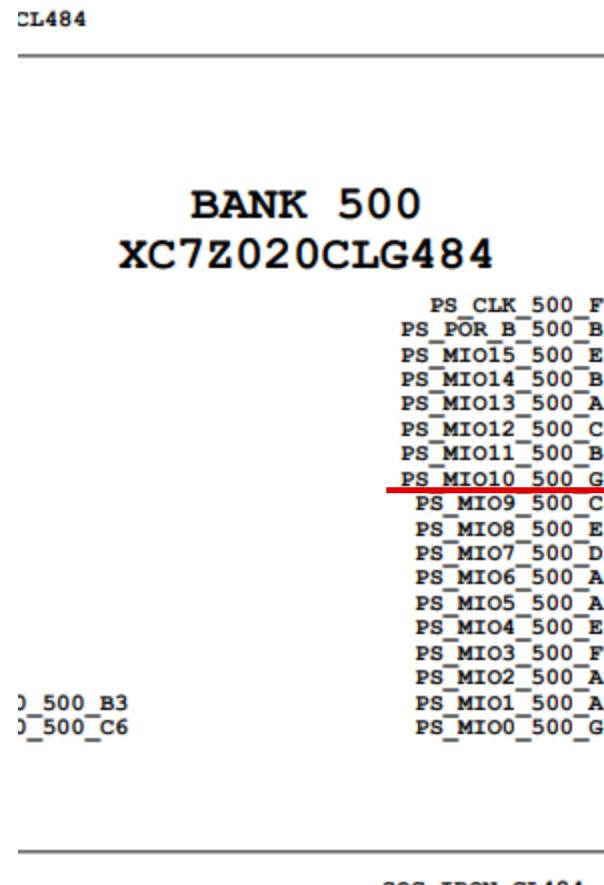
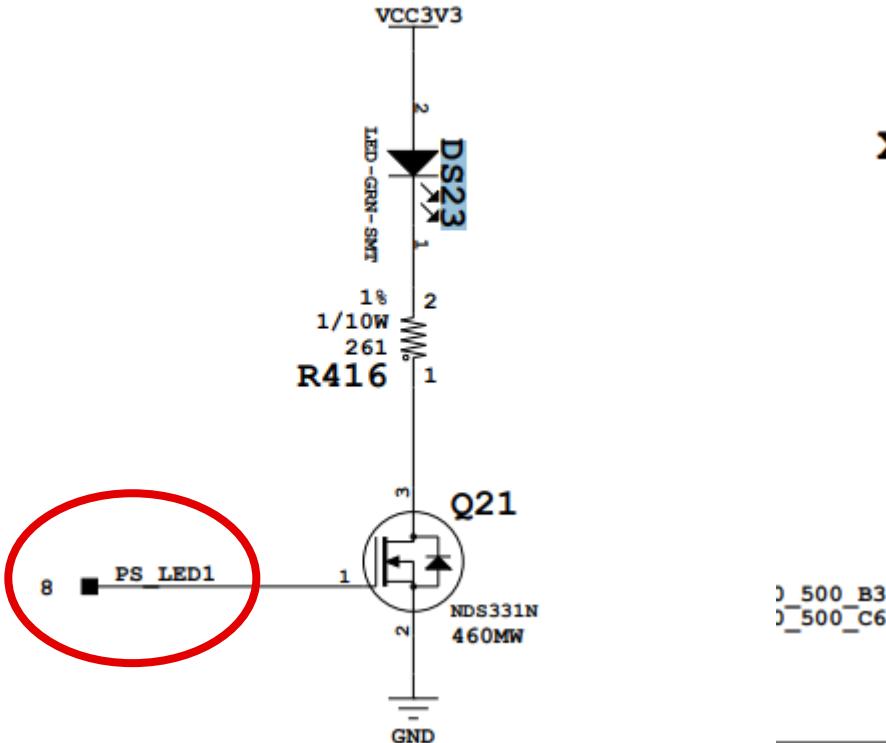
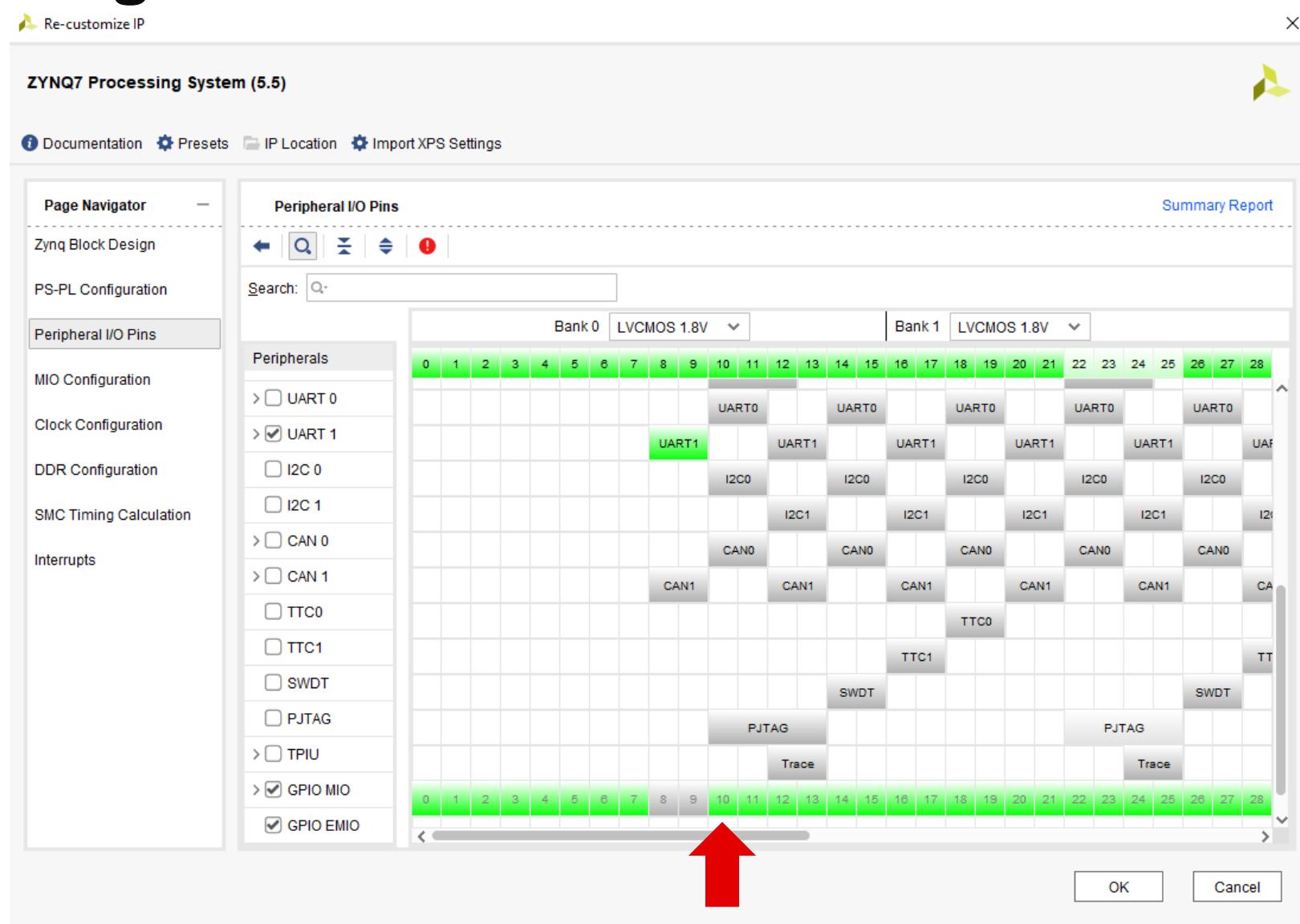


Figure 14-2: GPIO Channel

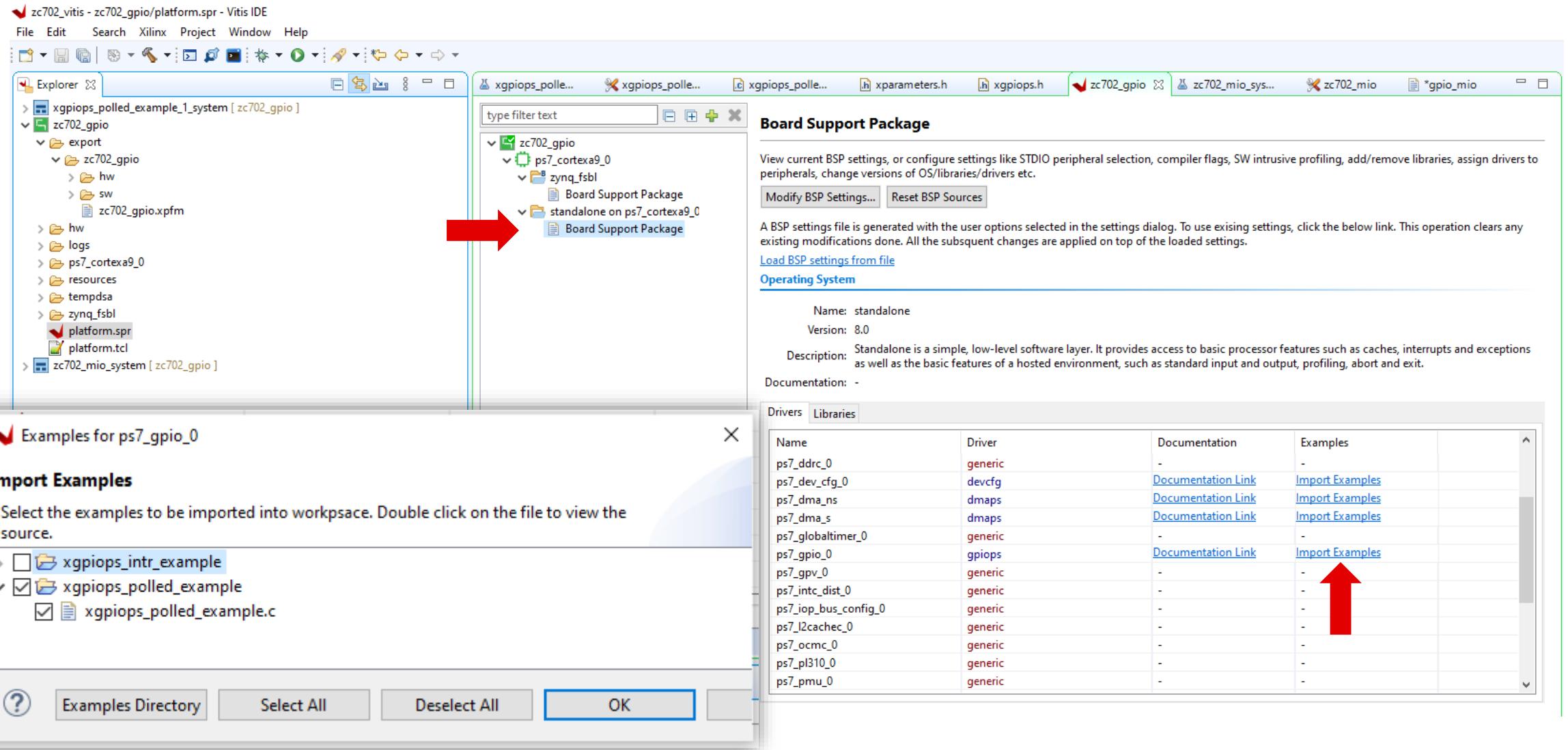
MIO Configuration



MIO Configuration



Vitis Example Design

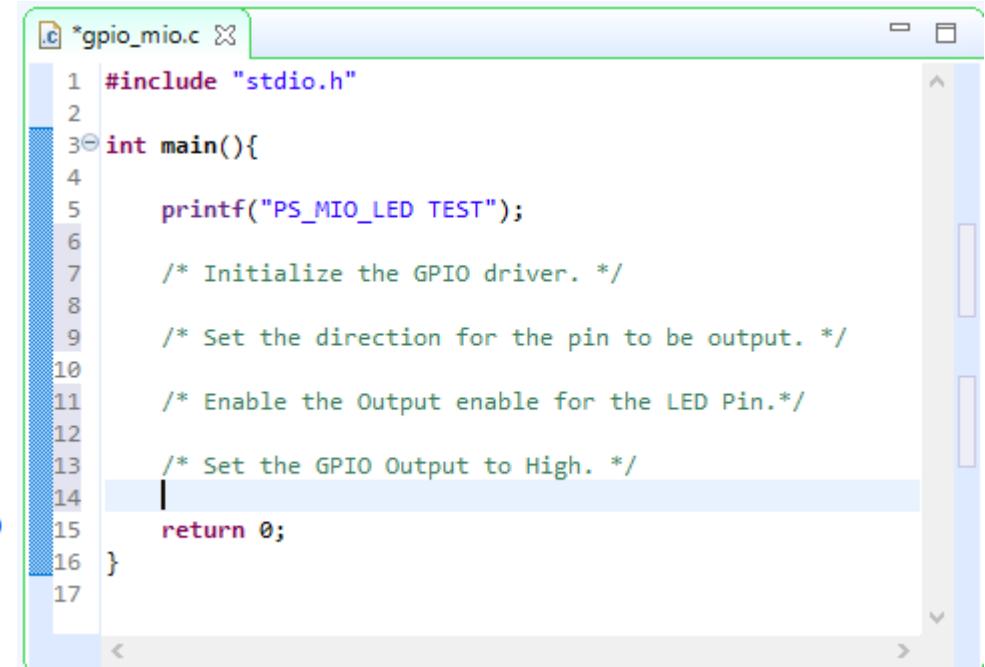


Programming Guide

Start-up Sequence

Main Example: Start-up Sequence

1. **Resets:** The reset options are described in section [Resets](#).
2. **Clocks:** The clocks are described in section [Clocks](#).
3. **GPIO Pin Configurations:** Configure pin as input/output is described in section [GPIO Pin Configurations](#).
4. **Write Data to GPIO Output pin:** Refer to example in section [Writing Data to GPIO Output Pins](#).
5. **Read Data from GPIO Input pin:** Refer to example in section [Reading Data from GPIO Input Pins](#).
6. **Set GPIO pin as wake-up event:** Refer to example in section [GPIO as Wake-up Event](#).



```
1 #include "stdio.h"
2
3 int main(){
4
5     printf("PS_MIO_LED TEST");
6
7     /* Initialize the GPIO driver. */
8
9     /* Set the direction for the pin to be output. */
10
11    /* Enable the Output enable for the LED Pin.*/
12
13    /* Set the GPIO Output to High. */
14
15    return 0;
16
17 }
```

Initialize the GPIO driver

```
119 int main(void)
120 {
121     int Status;
122     u32 InputData;
123
124     printf("GPIO Polled Mode Example Test \r\n");
125     Status = GpioPolledExample(GPIO_DEVICE_ID, &InputData);
126     if (Status != XST_SUCCESS) {
127         printf("GPIO Polled Mode Example Test Failed\r\n");
128         return XST_FAILURE;
129     }
130
131     printf("Data read from GPIO Input is 0x%08x \r\n", (int)InputData);
132     printf("Successfully ran GPIO Polled Mode Example Test\r\n");
133     return XST_SUCCESS;
134 }
135
```



Ctrl + Leftbutton

```
154 int GpioPolledExample(u16 DeviceId, u32 *DataRead)
155 {
156     int Status;
157     XGpioPs_Config *ConfigPtr;
158     int Type_of_board;
159
160     /* Initialize the GPIO driver. */
161     ConfigPtr = XGpioPs_LookupConfig(GPIO_DEVICE_ID);
162     Type_of_board = XGetPlatform_Info();
163     switch (Type_of_board) {
164         case XPLAT_ZYNQ_ULTRA_MP:
165             Input_Pin = 22;
166             Output_Pin = 23;
167             break;
168
169         case XPLAT_ZYNQ:
170             Input_Pin = 14;
171             Output_Pin = 10;
172             break;
173
174 #ifdef versal
175         case XPLAT_VERSAL:
176             /* Accessing PMC GPIO by setting field to 1 */
177             Gpio.PmcGpio = 1;
178             Input_Pin = 56;
179             Output_Pin = 52;
180             break;
181
182 #endif
183     Status = XGpioPs_CfgInitialize(&Gpio, ConfigPtr,
184                                     ConfigPtr->BaseAddr);
185     if (Status != XST_SUCCESS) {
186         return XST_FAILURE;
187     }
```

Initialize the GPIO driver

The screenshot shows a code editor window with four tabs at the top: `xgpiops_polled_example_1_system`, `xgpiops_polled_example_1`, `zc702_mio_system`, and `zc702_mio`. The `zc702_mio` tab is active. The code in the editor is:

```
1 #include "stdio.h"
2
3 int main(){
4
5     printf("PS_MIO_LED TEST");
6
7     /* Initialize the GPIO driver. */
8     ConfigPtr = XGpioPs_LookupConfig(GPIO_DEVICE_ID);
9     Status = XGpioPs_CfgInitialize(&Gpio, ConfigPtr,
10                                    ConfigPtr->BaseAddr);
11    /* Set the direction for the pin to be output. */
12
13    /* Enable the pin. */
14
15    /* Set the GPIO pin direction. */
16
17    return 0;
18 }
```

A tooltip is displayed over the line `Status = XGpioPs_CfgInitialize(&Gpio, ConfigPtr,` with the text: `'ConfigPtr' undeclared (first use in this function)`. Below the tooltip, a context menu is open with the option `Open Declaration` highlighted.

The context menu also includes the instruction `Select one element from the list`. A dropdown list shows five options, all starting with `XGpioPs_Config`:

- XGpioPs_Config - /zc702_gpio/export/zc702_gpio/sw/zc702_gpio/standalone_domain/bspinclude/include/xgpiops.h
- XGpioPs_Config - /zc702_gpio/ps7_cortexa9_0/standalone_domain/bsp/ps7_cortexa9_0/include/xgpiops.h
- XGpioPs_Config - /zc702_gpio/ps7_cortexa9_0/standalone_domain/bsp/ps7_cortexa9_0/libsrc/gpiops_v3_10/src/xgpiops.h
- XGpioPs_Config - /zc702_gpio/zynq_fsbl/zynq_fsbl_bsp/ps7_cortexa9_0/include/xgpiops.h
- XGpioPs_Config - /zc702_gpio/zynq_fsbl/zynq_fsbl_bsp/ps7_cortexa9_0/libsrc/gpiops_v3_10/src/xgpiops.h

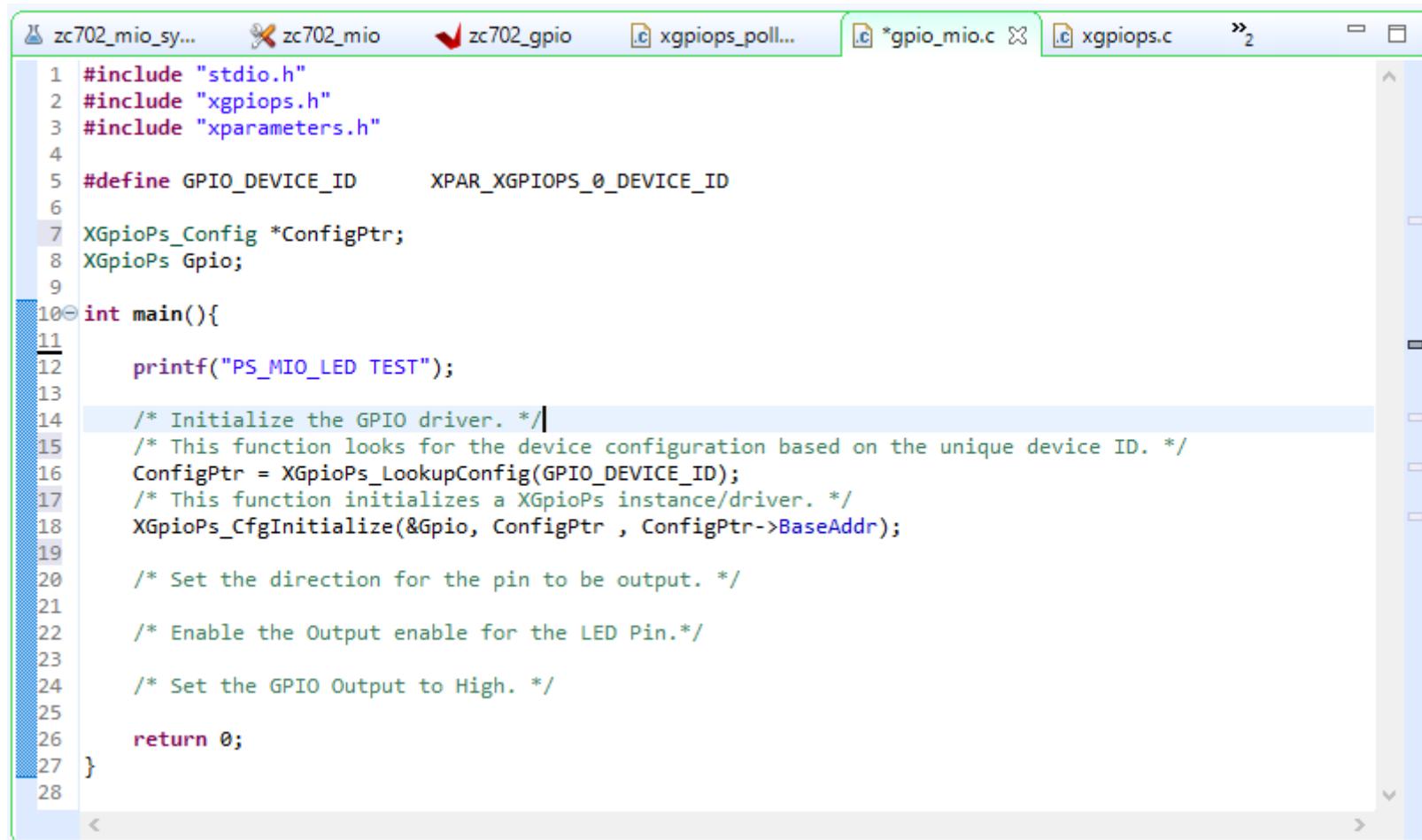
At the bottom right of the context menu are `OK` and `Cancel` buttons.

Initialize the GPIO driver

```
.h xgpiops.h ✘
211 //***** Function Prototypes *****/
212
213 /* Functions in xgpiops.c */
214 s32 XGpioPs_CfgInitialize(XGpioPs *InstancePtr, const XGpioPs_Config *ConfigPtr,
215                           u32 EffectiveAddr);
216
217 /* Bank APIs in xgpiops.c */
218 u32 XGpioPs_Read(const XGpioPs *InstancePtr, u8 Bank);
219 void XGpioPs_Write(const XGpioPs *InstancePtr, u8 Bank, u32 Data);
220 void XGpioPs_SetDirection(const XGpioPs *InstancePtr, u8 Bank, u32 Direction);
221 u32 XGpioPs_GetDirection(const XGpioPs *InstancePtr, u8 Bank);
222 void XGpioPs_SetOutputEnable(const XGpioPs *InstancePtr, u8 Bank, u32 OpEnable);
223 u32 XGpioPs_GetOutputEnable(const XGpioPs *InstancePtr, u8 Bank);
224 #ifdef versal
225 void XGpioPs_GetBankPin(const XGpioPs *InstancePtr, u8 PinNumber, u8 *BankNumber, u8
226 #else
227 void XGpioPs_GetBankPin(u8 PinNumber, u8 *BankNumber, u8 *PinNumberInBank);
228 #endif
229
230 /* Pin APIs in xgpiops.c */
231 u32 XGpioPs_ReadPin(const XGpioPs *InstancePtr, u32 Pin);
232 void XGpioPs_WritePin(const XGpioPs *InstancePtr, u32 Pin, u32 Data);
233 void XGpioPs_SetDirectionPin(const XGpioPs *InstancePtr, u32 Pin, u32 Direction);
234 u32 XGpioPs_GetDirectionPin(const XGpioPs *InstancePtr, u32 Pin);
```

```
.h xparameters.h ✘
290
291
292 //*****
293
294 /* Canonical definitions for peripheral PS7_GPIO_0 */
295 #define XPAR_XGPIOPS_0_DEVICE_ID XPAR_PS7_GPIO_0_DEVICE_ID
296 #define XPAR_XGPIOPS_0_BASEADDR 0xE000A000
297 #define XPAR_XGPIOPS_0_HIGHADDR 0xE000AFFF
298
299 //*****
300 /* Definitions for driver IICPS */
301 #define XPAR_XIICPS_NUM_INSTANCES 1
302
303 /* Definitions for peripheral PS7_I2C_0 */
304 #define XPAR_PS7_I2C_0_DEVICE_ID 0
305 #define XPAR_PS7_I2C_0_BASEADDR 0xE0004000
306 #define XPAR_PS7_I2C_0_HIGHADDR 0xE0004FFF
307 #define XPAR_PS7_I2C_0_I2C_CLK_FREQ_HZ 111111115
308
309 //*****
310
311
312 /* Canonical definitions for peripheral PS7_I2C_0 */
313 #define XPAR_XIICPS_0_DEVICE_ID XPAR_PS7_I2C_0_DEVICE_ID
314 #define XPAR_XIICPS_0_BASEADDR 0xE0004000
315 #define XPAR_XIICPS_0_HIGHADDR 0xE0004FFF
316 #define XPAR_XIICPS_0_I2C_CLK_FREQ_HZ 111111115
317
318 //*****
319
320
321 /* Definition for input Clock */
322
```

Initialize the GPIO driver



The screenshot shows a code editor window with multiple tabs at the top: zc702_mio_sy..., zc702_mio, zc702_gpio, xgpiops_poll..., *gpio_mio.c (selected), xgpiops.c, and »2. The main area displays the following C code:

```
1 #include "stdio.h"
2 #include "xgpiops.h"
3 #include "xparameters.h"
4
5 #define GPIO_DEVICE_ID      XPAR_XGPIOPS_0_DEVICE_ID
6
7 XGpioPs_Config *ConfigPtr;
8 XGpioPs Gpio;
9
10 int main(){
11     printf("PS_MIO_LED TEST");
12
13     /* Initialize the GPIO driver. */
14     /* This function looks for the device configuration based on the unique device ID. */
15     ConfigPtr = XGpioPs_LookupConfig(GPIO_DEVICE_ID);
16     /* This function initializes a XGpioPs instance/driver. */
17     XGpioPs_CfgInitialize(&Gpio, ConfigPtr , ConfigPtr->BaseAddr);
18
19     /* Set the direction for the pin to be output. */
20
21     /* Enable the Output enable for the LED Pin.*/
22
23     /* Set the GPIO Output to High. */
24
25
26     return 0;
27 }
28
```

Pin Configuration and Writing Data

GPIO Pin Configurations

Each individual GPIO pin can be configured as input/output. However, bank0 [8:7] pins must be configured as outputs. Refer to section [Bank0, Bits\[8:7\] are Outputs](#) for further details.

Example: Configure MIO pin 10 as an output

- Set the direction as output:** Write `0x0000_0400` to the `gpio.DIRM_0` register.
- Set the output enable:** Write `0x0000_0400` to the `gpio.OEN_0` register.

Note: The output enable has significance only when the GPIO pin is configured as an output.

Example: Configure MIO pin 10 as an input

- Set the direction as input:** Write `0x0` to the `gpio.DIRM_0` register. This sets `gpio.DIRM_0[10] = 0`.

Writing Data to GPIO Output Pins

For GPIO pins configured as outputs, there are two options to program the desired value.

Option 1: Read, modify, and update the GPIO pin using the `gpio.DATA_0` register.

Example: Set GPIO output pin 10 using the DATA_0 register.

- Read the `gpio.DATA_0` register:** Read `gpio.DATA_0` register to the `reg_val` variable.
- Modify the value:** Set `reg_val[10] = 1`.
- Write updated value to output pin:** Write `reg_val` to the `gpio.DATA_0` register.

Pin Configuration and Writing Data

```
188 /* Run the Output Example. */
189 Status = GpioOutputExample();
190 if (Status != XST_SUCCESS) {
191     return XST_FAILURE;
192 }

216 static int GpioOutputExample(void)
217 {
218     u32 Data;
219     volatile int Delay;
220     u32 LedLoop;
221
222     /*
223      * Set the direction for the pin to be output and
224      * Enable the Output enable for the LED Pin.
225      */
226     XGpioPs_SetDirectionPin(&Gpio, Output_Pin, 1);
227     XGpioPs_SetOutputEnablePin(&Gpio, Output_Pin, 1);
228
229     /* Set the GPIO output to be low. */
230     XGpioPs_WritePin(&Gpio, Output_Pin, 0x0);
231 }

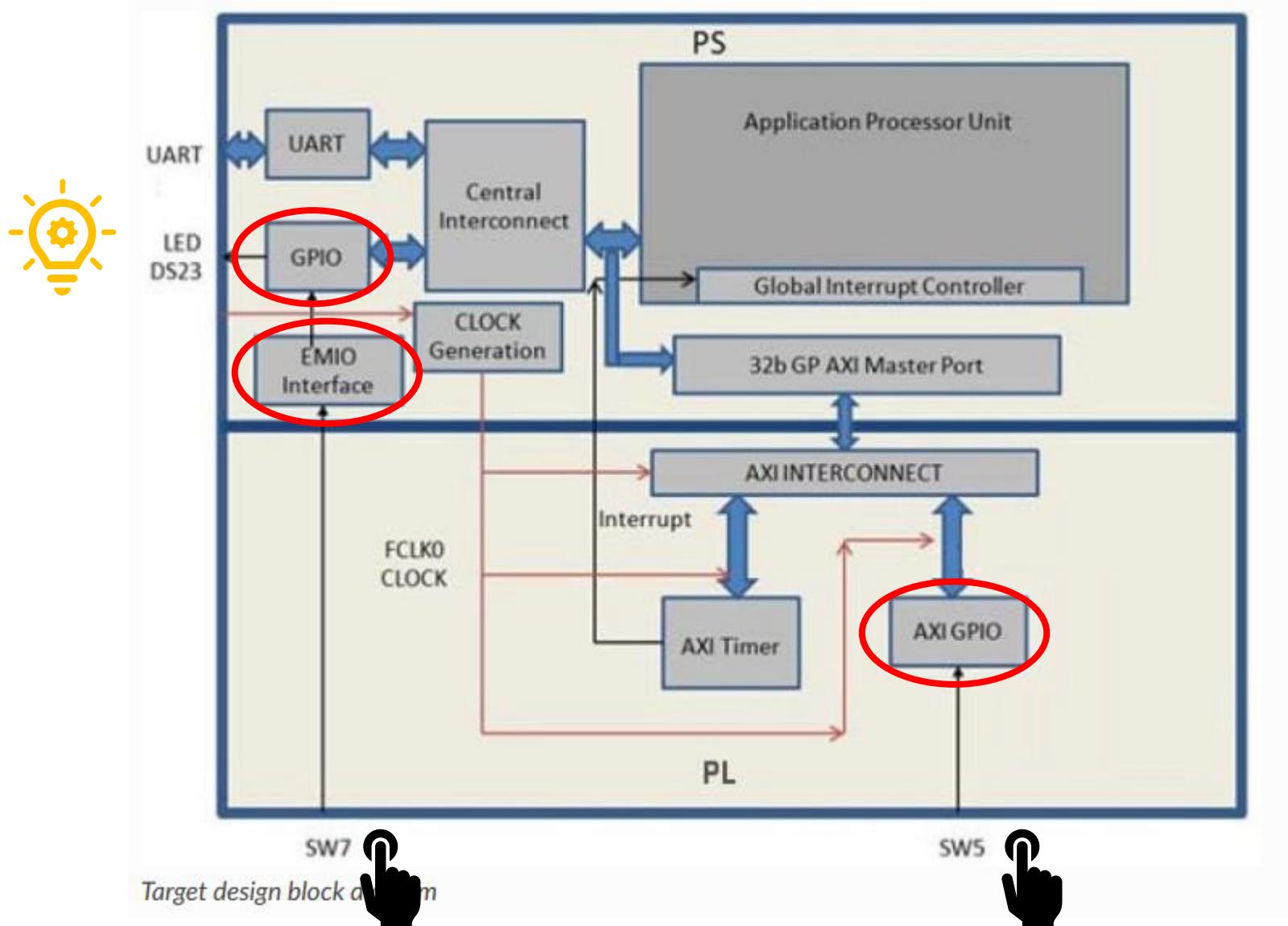
1 #include "stdio.h"
2 #include "xgpiops.h"
3 #include "xparameters.h"
4
5 #define GPIO_DEVICE_ID      XPAR_XGPIOPS_0_DEVICE_ID
6 #define LED_MAX_BLINK      0x10
7 #define LED_DELAY          10000000
8
9 XGpioPs_Config *ConfigPtr;
10 XGpioPs Gpio;
11
12 int main(){
13
14     printf("PS_MIO_LED TEST");
15
16     /* Initialize the GPIO driver. */
17     /* This function looks for the device configuration based on the unique device ID. */
18     ConfigPtr = XGpioPs_LookupConfig(GPIO_DEVICE_ID);
19     /* This function initializes a XGpioPs instance/driver. */
20     XGpioPs_CfgInitialize(&Gpio, ConfigPtr , ConfigPtr->BaseAddr);
21
22     /* Set the direction for the pin to be output.(0 for Input Direction, 1 for Output Direction) */
23     XGpioPs_SetDirectionPin(&Gpio, 10, 1);
24     /* Enable the Output enable for the LED Pin.(0 for Disabling , 1 for Enabling) */
25     XGpioPs_SetOutputEnablePin(&Gpio, 10, 1);
26     /* Set the GPIO Output to High. (0 for Output to Low , 1 for Output to High)*/
27     XGpioPs_WritePin(&Gpio, 10, 1);
28 }
29
```



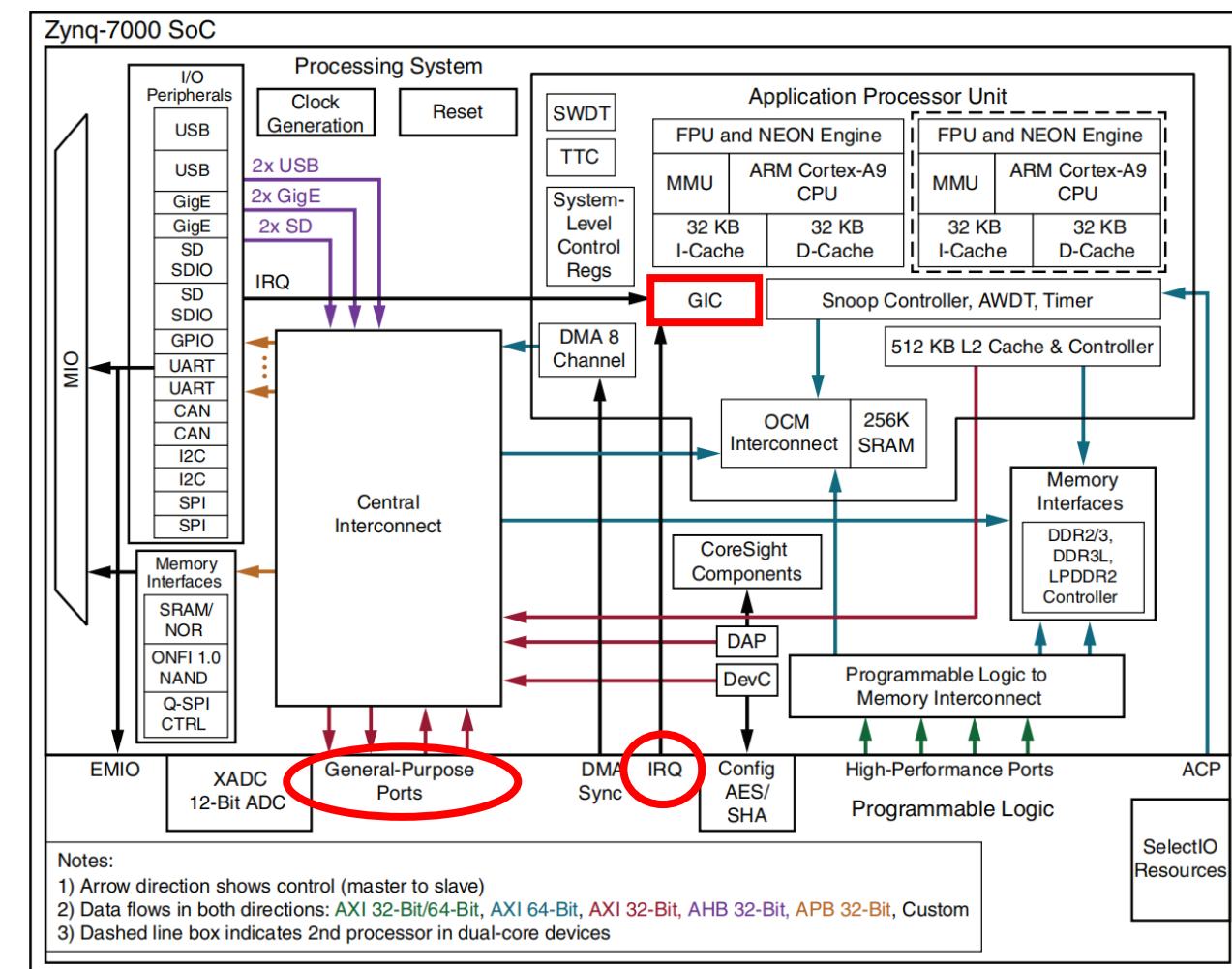
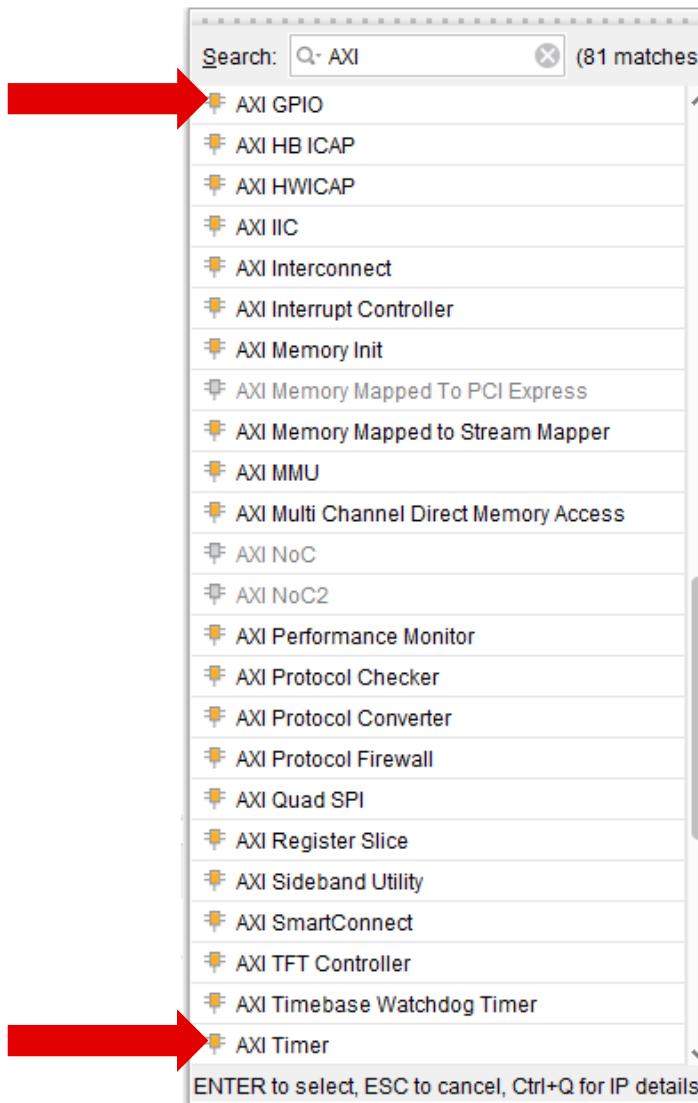
Using the PL GP Port in Zynq Devices

Course Agenda
2023

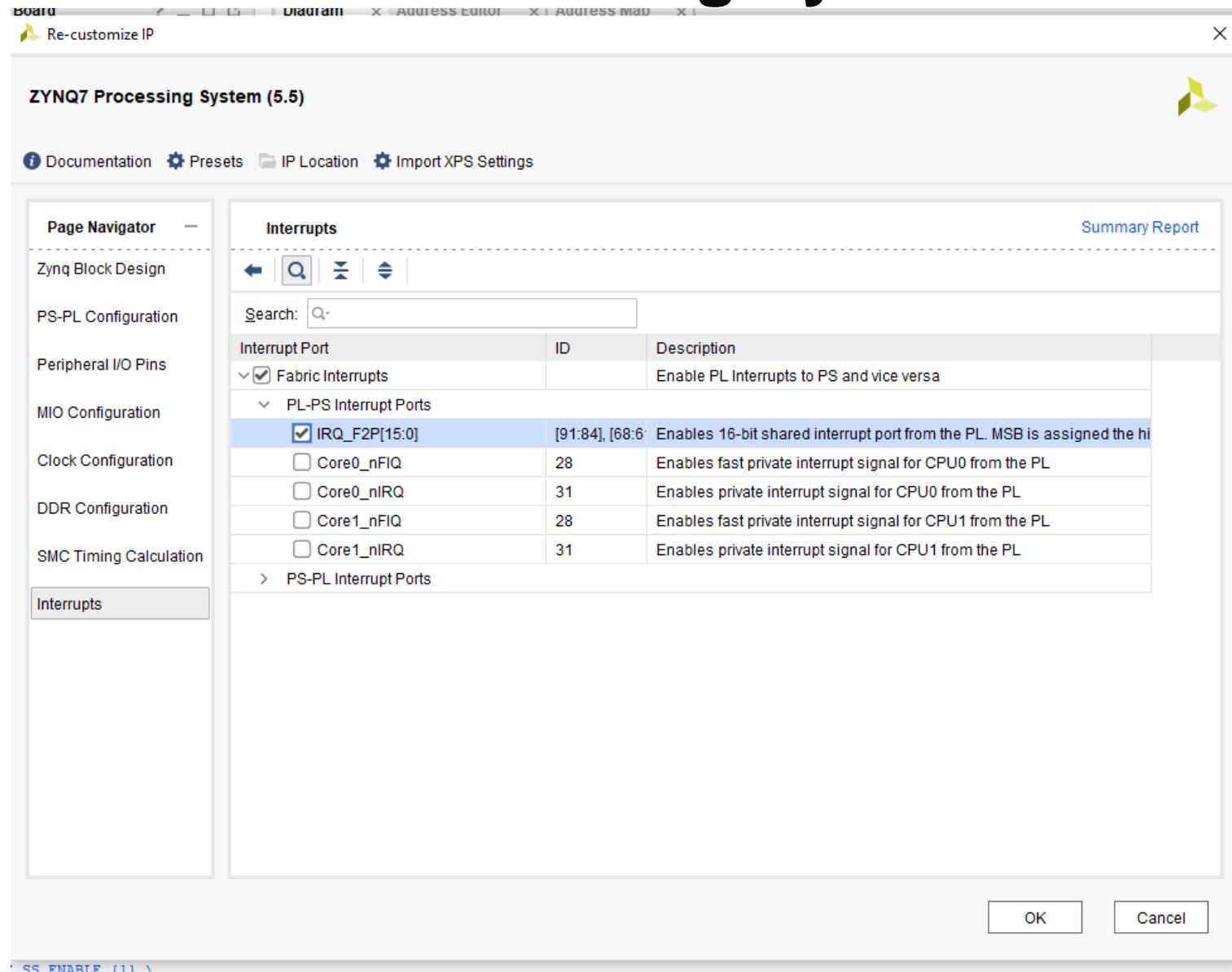
Using the GP Port in Zynq Devices



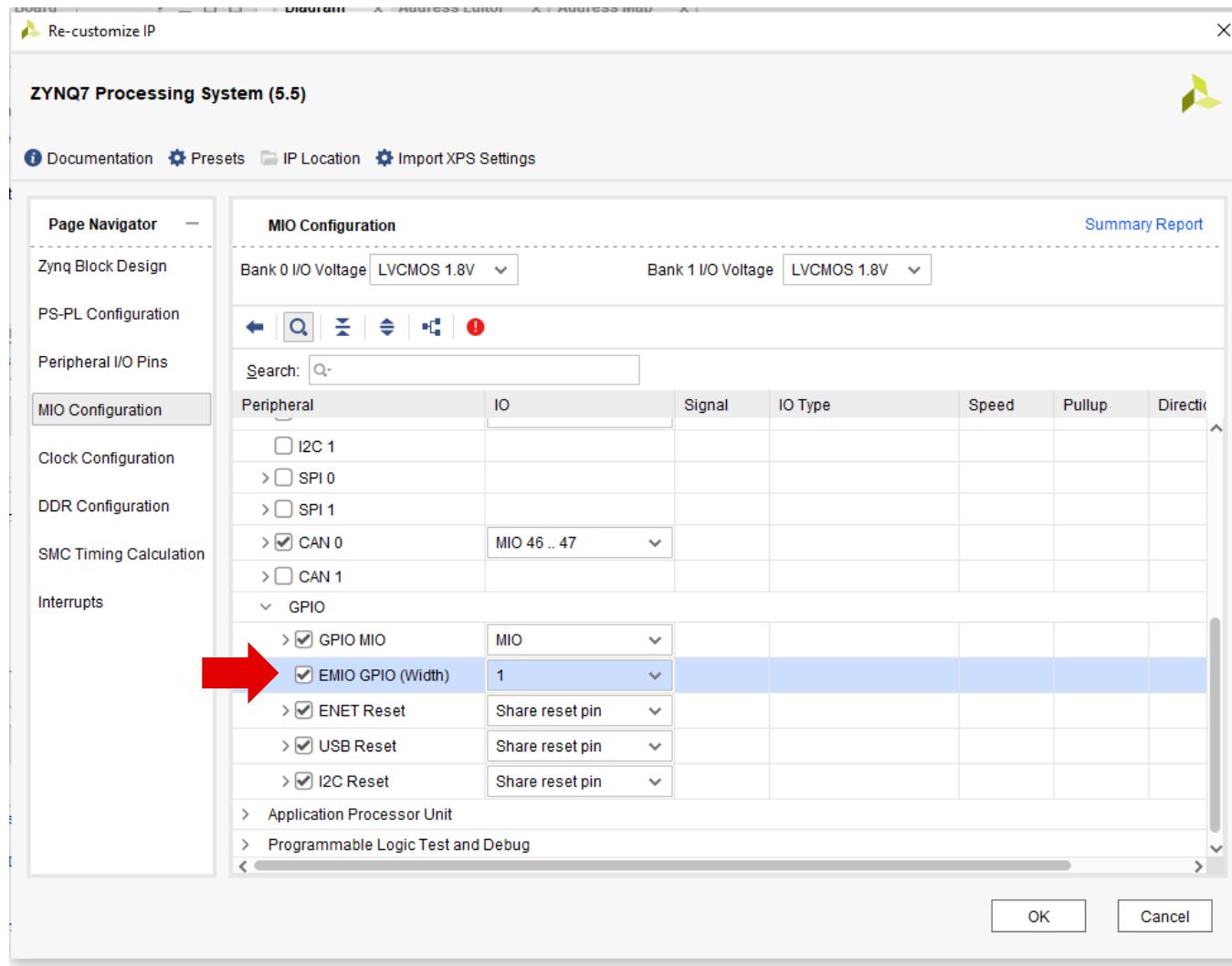
AXI GPIO & AXI Timer



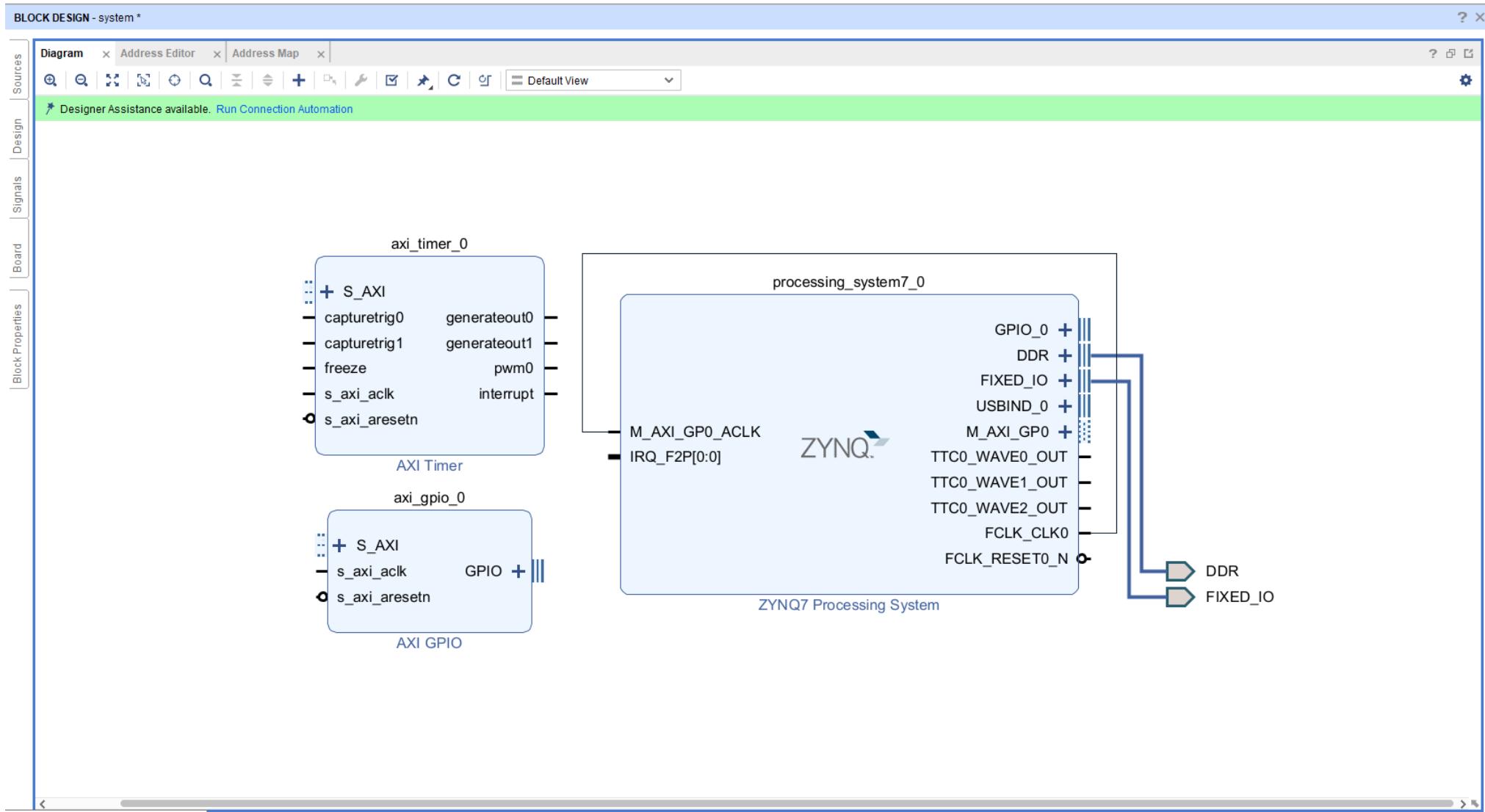
Enable the ZYNQ7 Processing System interrupt



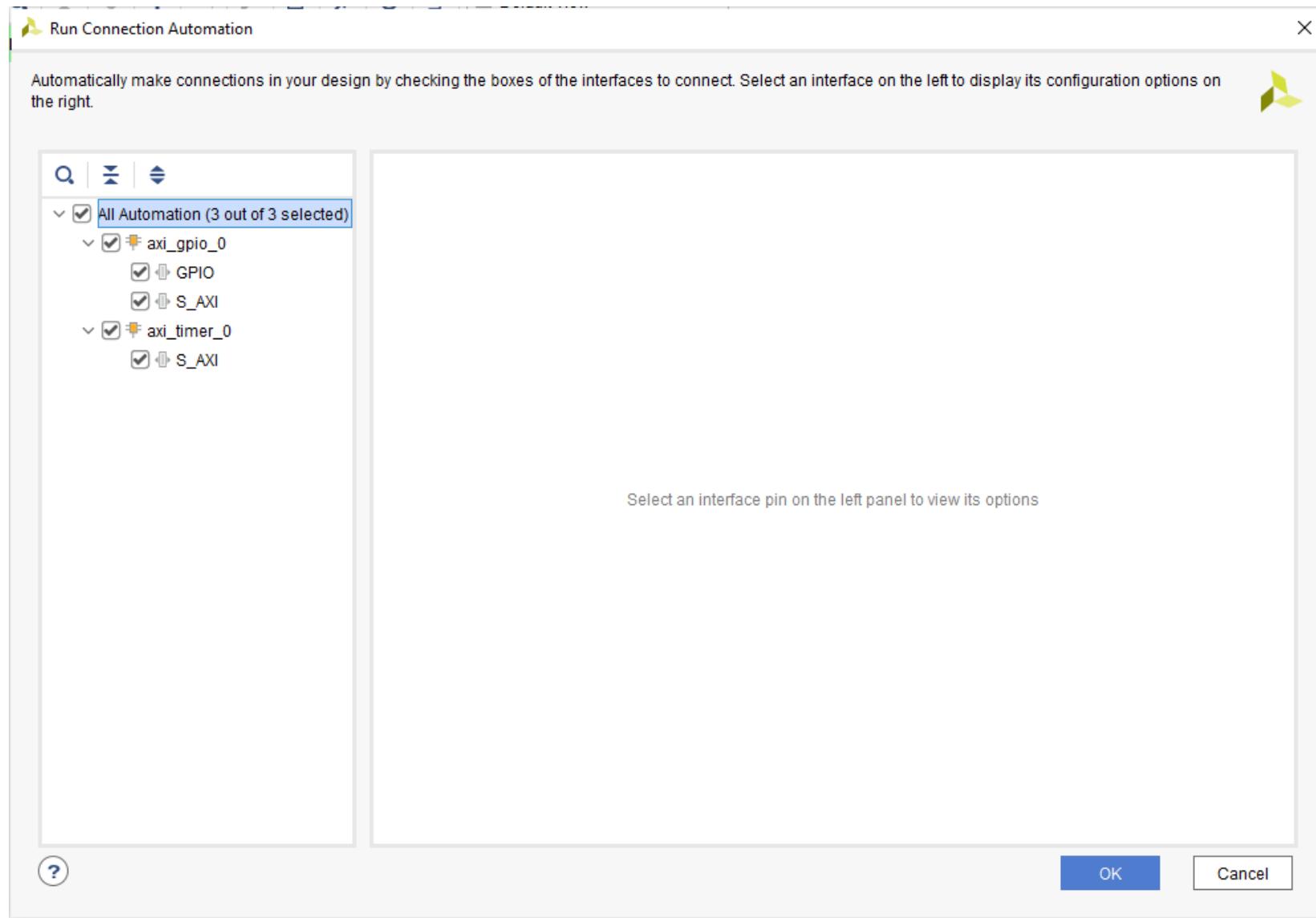
Enable the ZYNQ7 Processing System EMIO GPIO



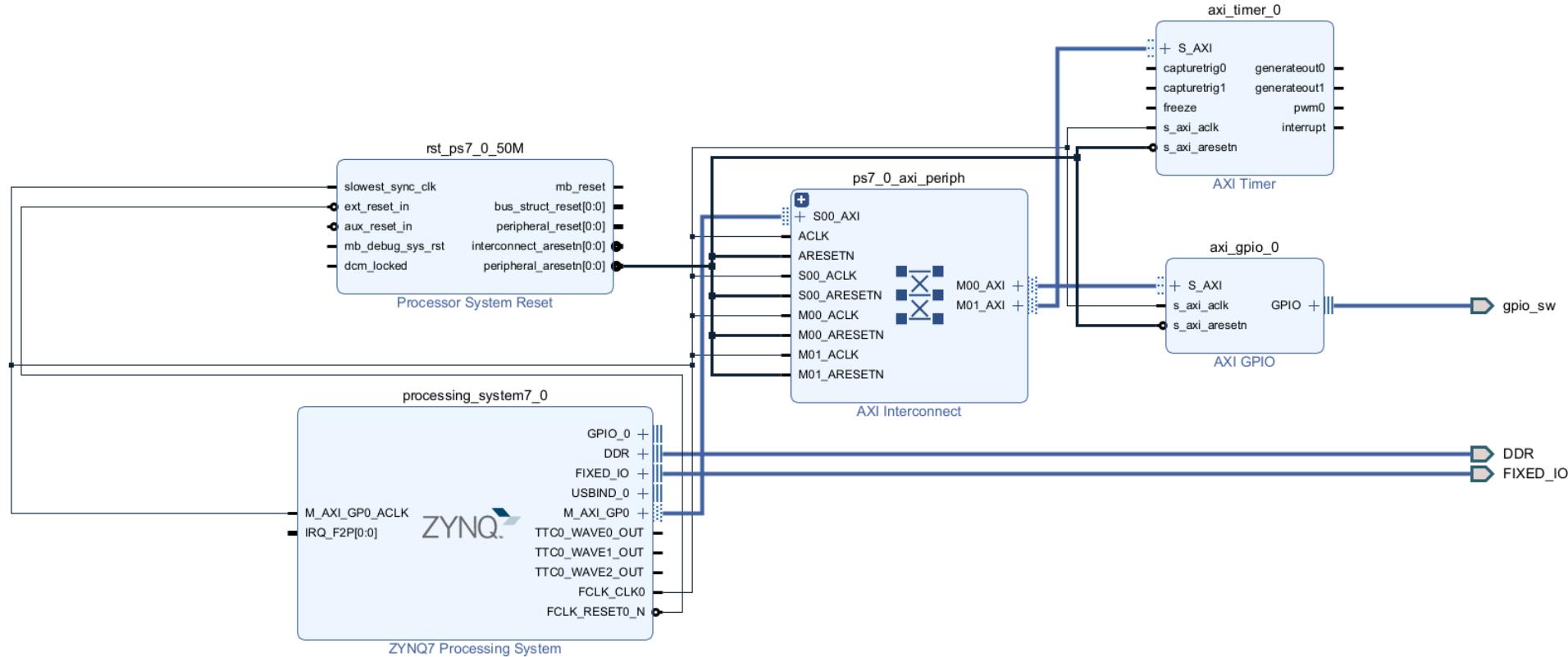
Connect the PL IPs



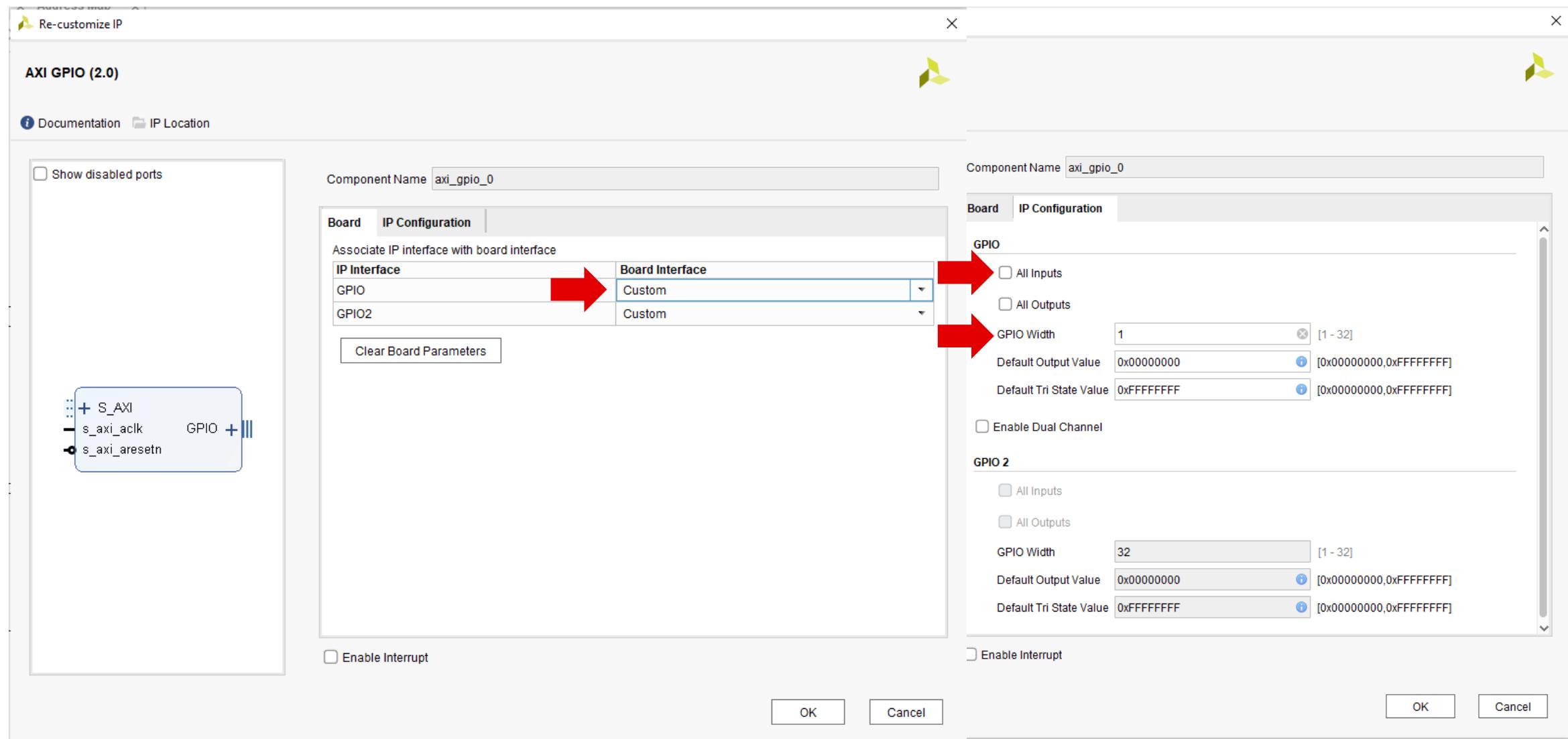
Connect the PL IPs



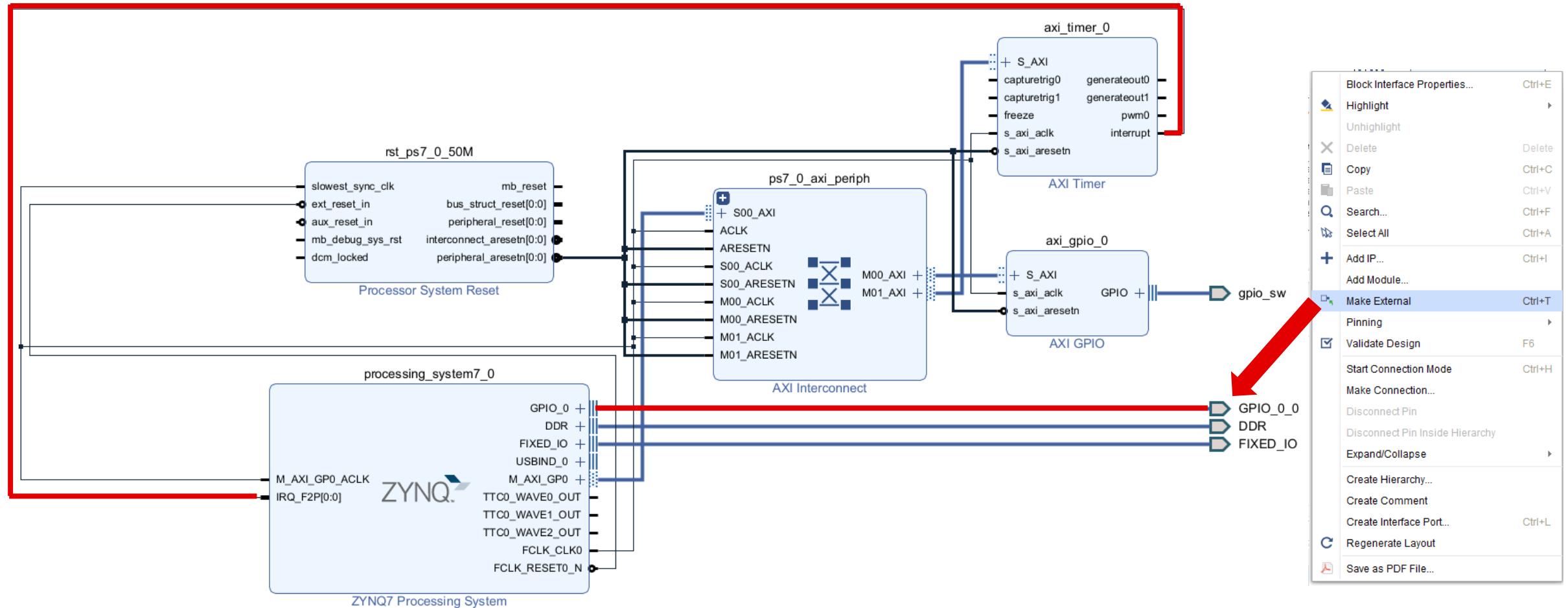
Connect the PL IPs



Customize the AXI GPIO IP block



Connect interrupt signals and Make GPIO output



Assigning Location Constraints to External Pins

Tcl Console | Messages | Log | Reports | Design Runs | Package Pins | **I/O Ports** x

Search | Filter | New | Open | Save | Help | Settings

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
All ports (132)														
> DDR_12642 (71)	INOUT					✓	502	(Multiple)*	1.500	(Multiple)		(Multiple)	NONE	FP_VT
> FIXED_IO_12642 (59)	INOUT					✓	(Multiple)	(Multiple)*	(Multiple)	(Multiple)	(Multiple)	(Multiple)	(Multiple)	(Multiple)
GPIO_0_0_12642 (1)	INOUT					✓	35	LVCMOS25*	2.500	12	✓	NONE	✓	FP_VT
GPIO_0_0_tri_io (1)	INOUT					✓	35	LVCMOS25*	2.500	12	✓	NONE	✓	FP_VT
GPIO_0_0_tri_io[0]	INOUT				F19	✓	35	LVCMOS25*	2.500	12	✓	NONE	✓	FP_VT
Scalar ports (0)														
gpio_sw_12642 (1)	INOUT					✓	35	LVCMOS25*	2.500	12	✓	NONE	✓	FP_VT
gpio_sw_tri_io (1)	INOUT					✓	35	LVCMOS25*	2.500	12	✓	NONE	✓	FP_VT
gpio_sw_tri_io[0]	INOUT				G19	✓	35	LVCMOS25*	2.500	12	✓	NONE	✓	FP_VT
Scalar ports (0)														
Scalar ports (0)														

Save Constraints

Select a target file to write new unsaved constraints to.
Choosing an existing file will update that file with the new constraints.

Create a new file

File type: XDC

File name: constraints

File location: <Local to Project>

Select an existing file

<select a target file>

OK Cancel

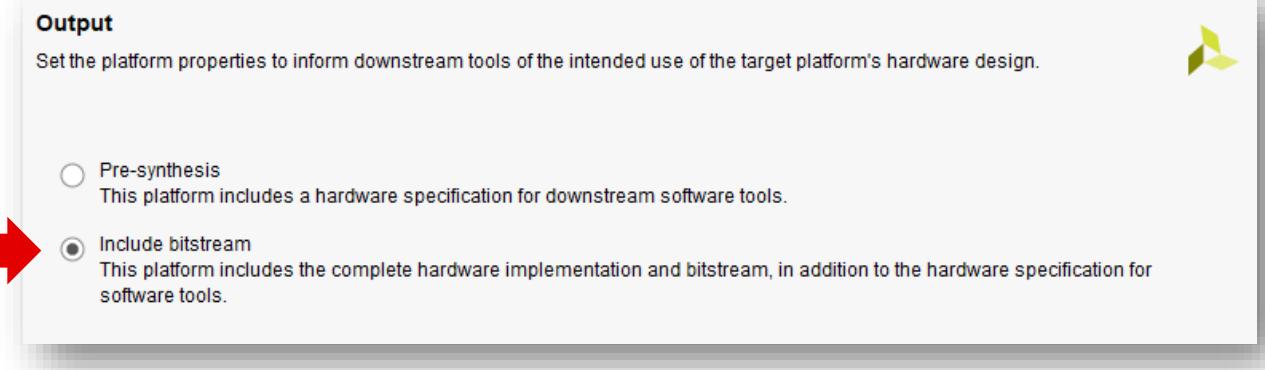
134

Export Hardware Platform

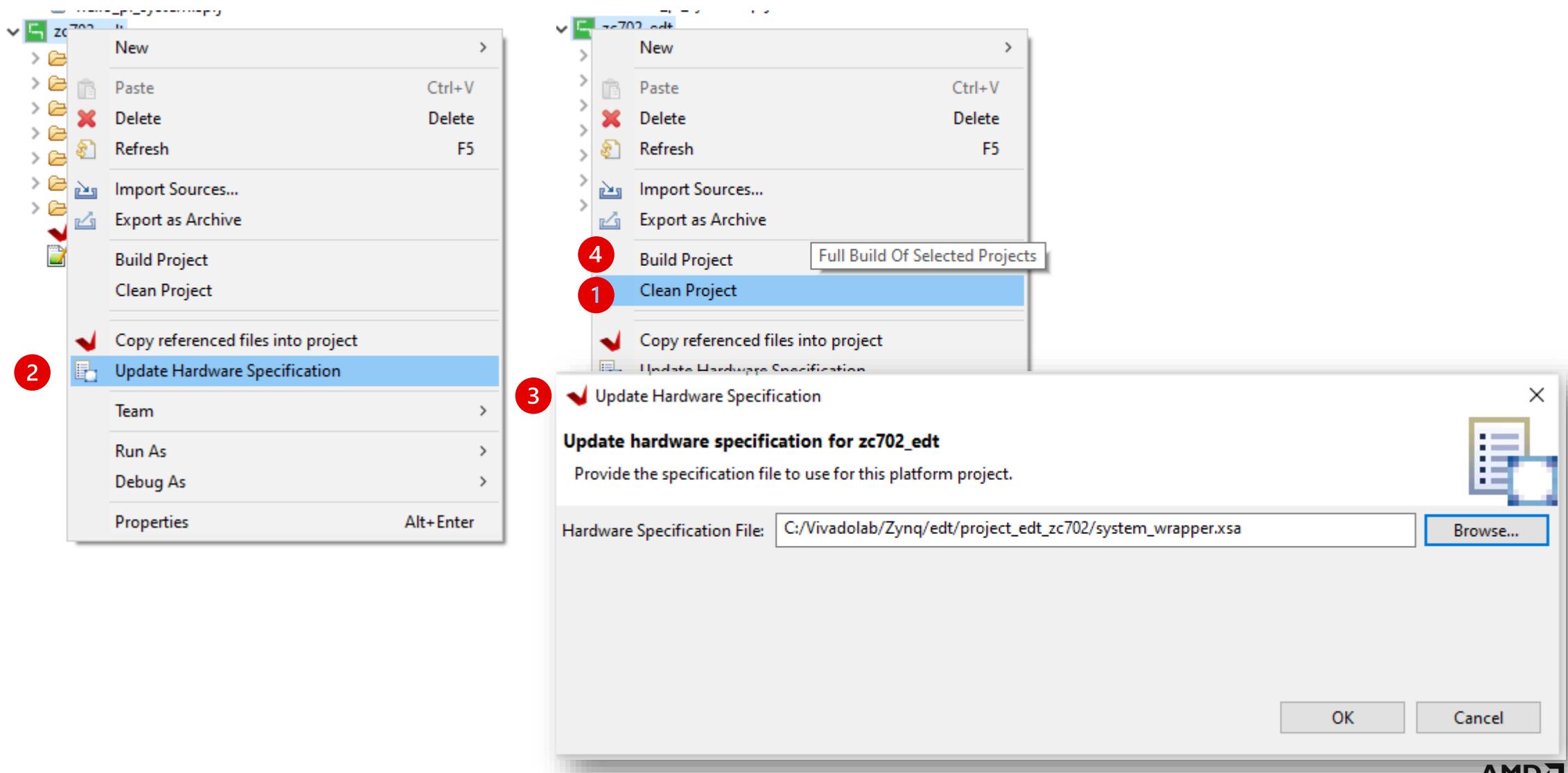
The screenshot shows the Vivado IDE's Sources browser on the left and an output configuration dialog on the right.

Sources Browser:

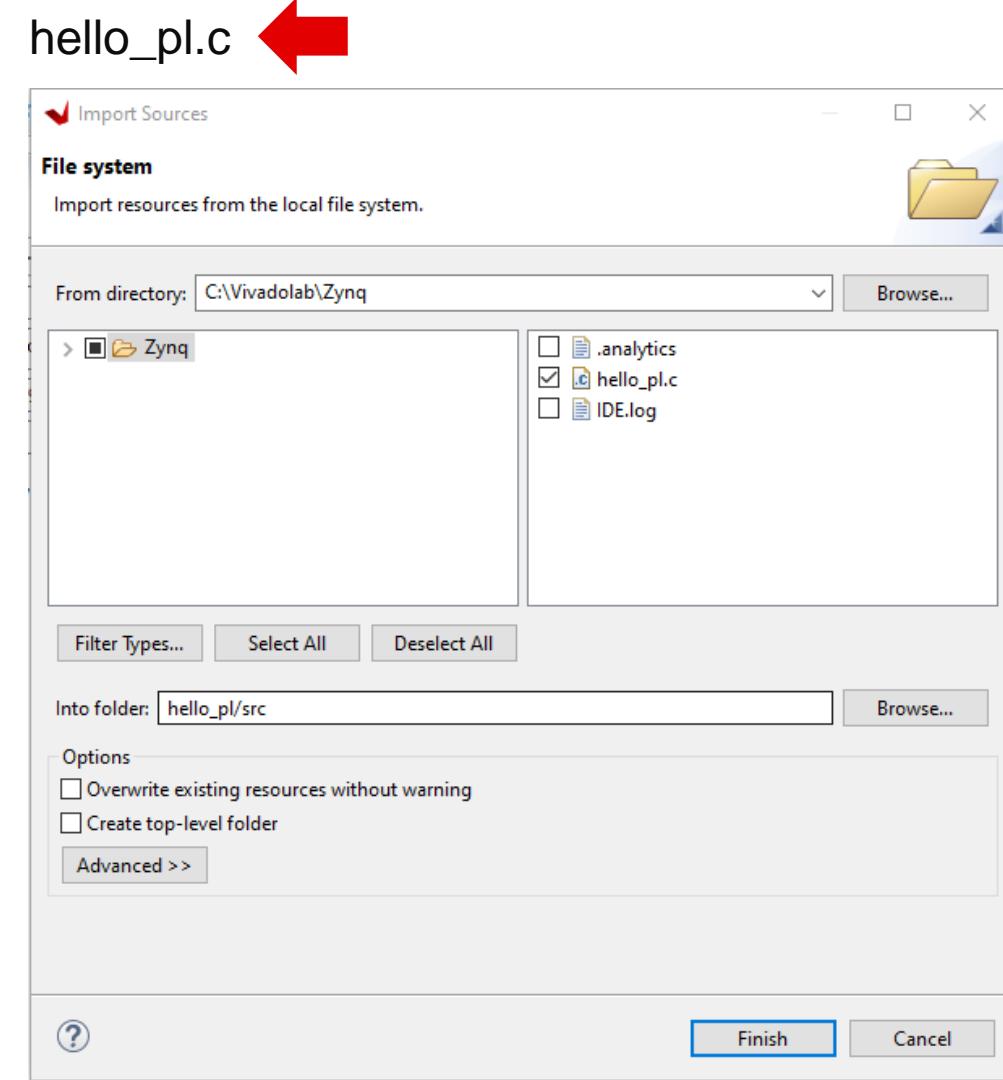
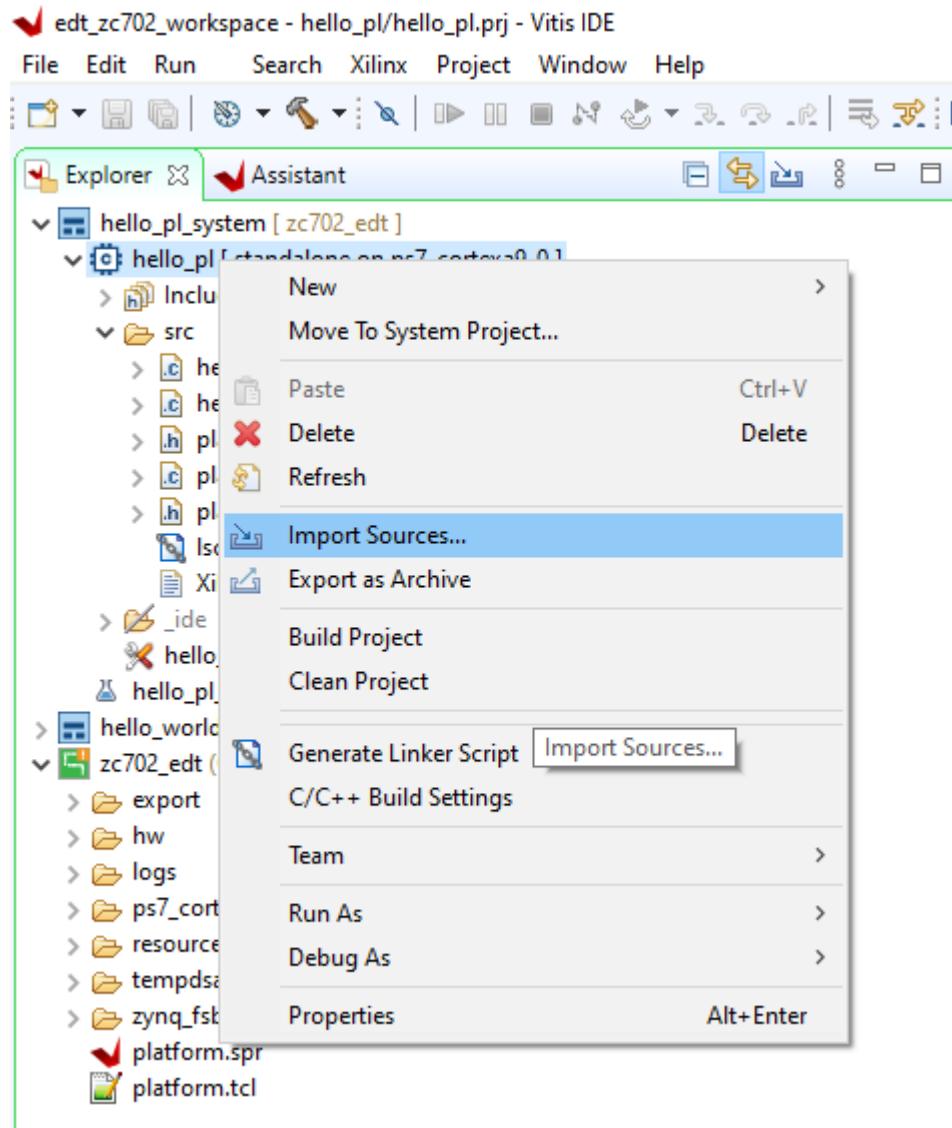
- Sources tab (selected)
- Netlist tab
- Device Constraints tab
- Search and filter icons
- Design Sources (1):
 - system_wrapper (system_wrapper.v) (1)
 - system_i : system (system.bd) (1)
 - system (system.v) (6)
 - axi_gpio_0 : system_axi_gpio_0_1 (system_axi_gpio_0_1.xci)
 - axi_timer_0 : system_axi_timer_0_1 (system_axi_timer_0_1.xci)
 - processing_system7_0 : system_processing_system7_0_0 (system_processing_system7_0_0.xci)
 - ps7_0_axi_periph : system_ps7_0_axi_periph_0 (system.v) (4)
 - ps7_0_axi_periph : system_ps7_0_axi_periph_0
 - rst_ps7_0_50M : system_rst_ps7_0_50M_0 (system_rst_ps7_0_50M_0.xci)
 - Constraints (1):
 - constrs_1 (1)
 - constraints.xdc (target)
 - Simulation Sources (1):
 - sim_1 (1)
 - Utility Sources:
 - utils_1



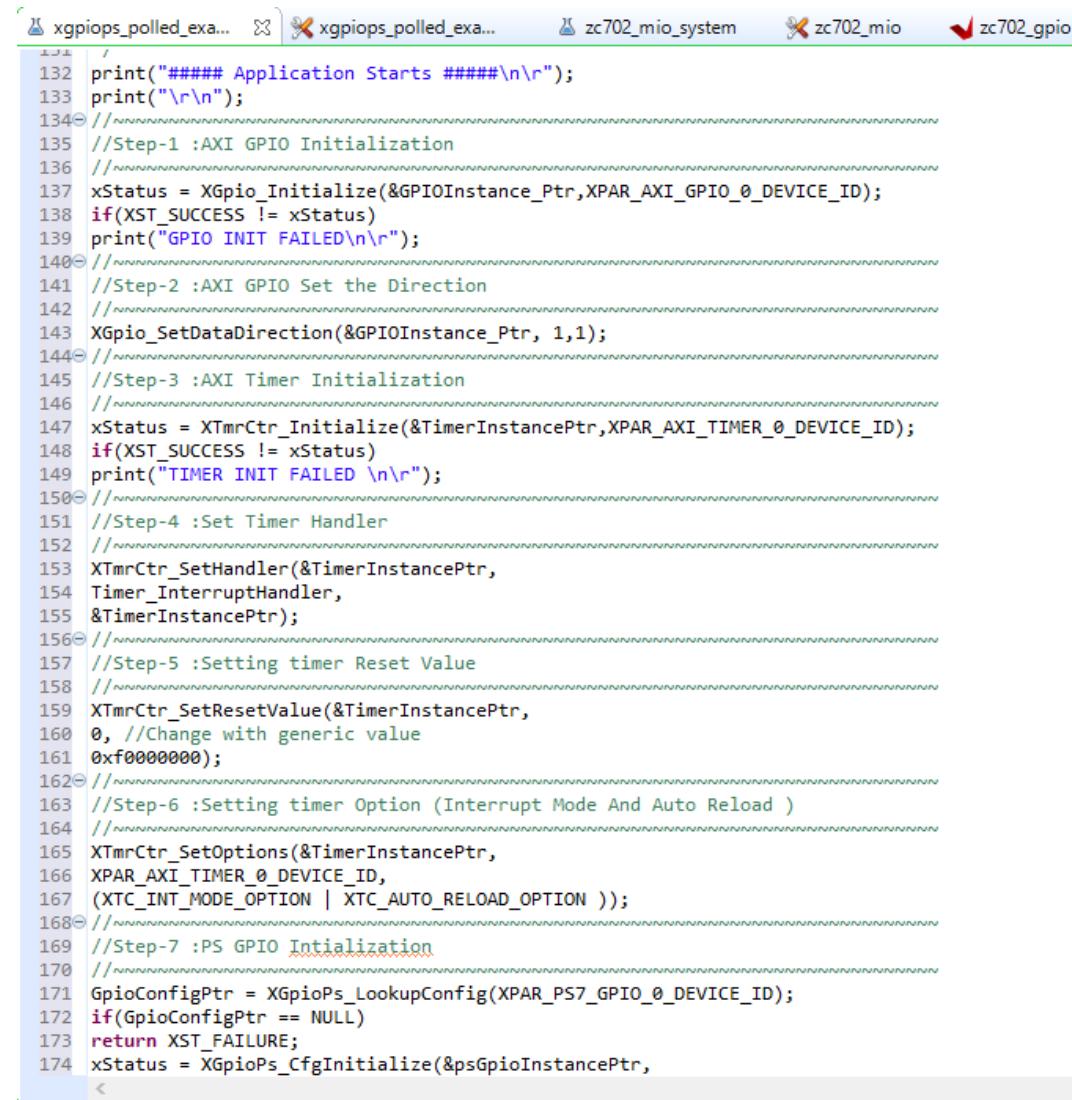
Updating Hardware in the Vitis Software Platform



Testing the PL IP with Prepared Software

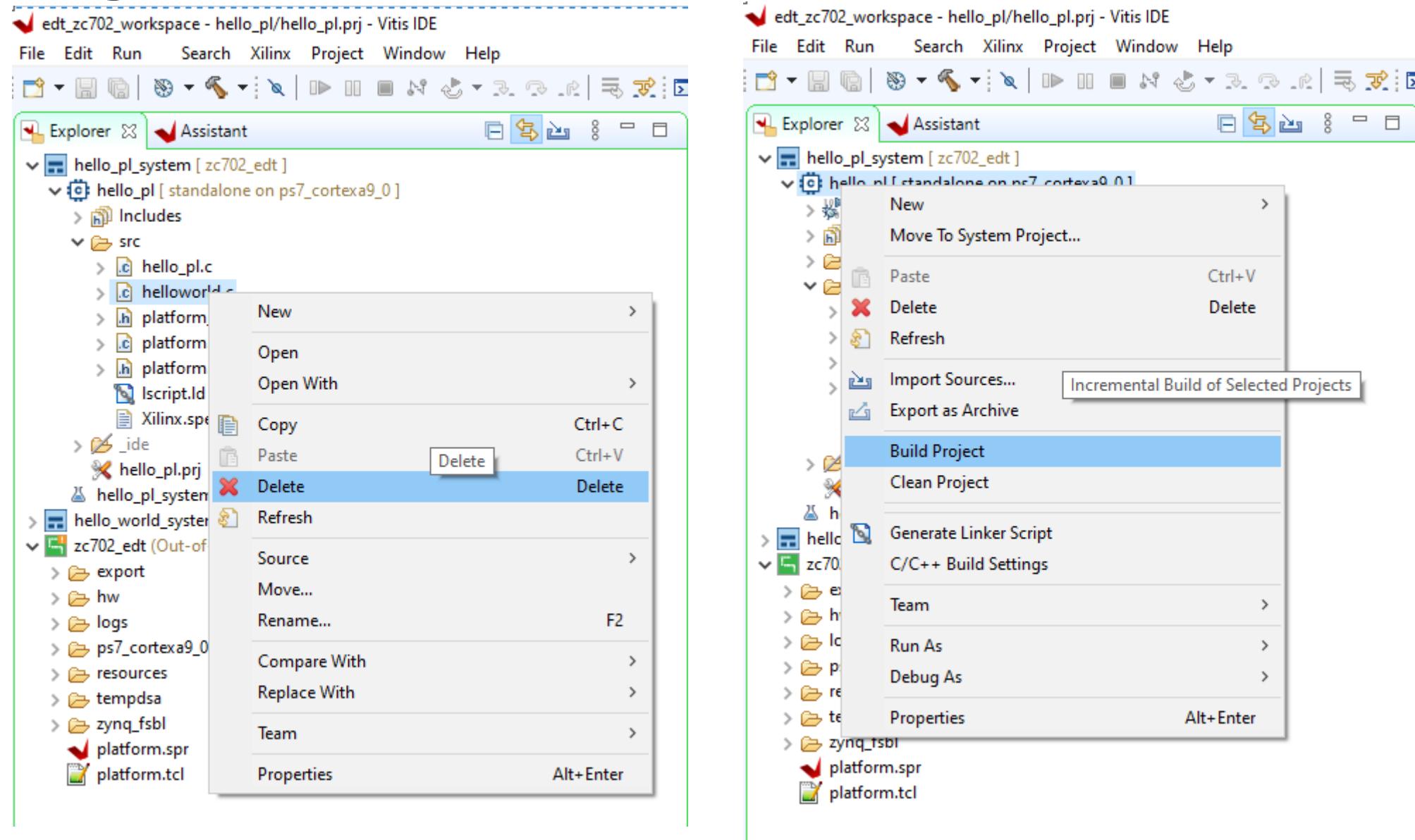


Testing the PL IP with Prepared Software

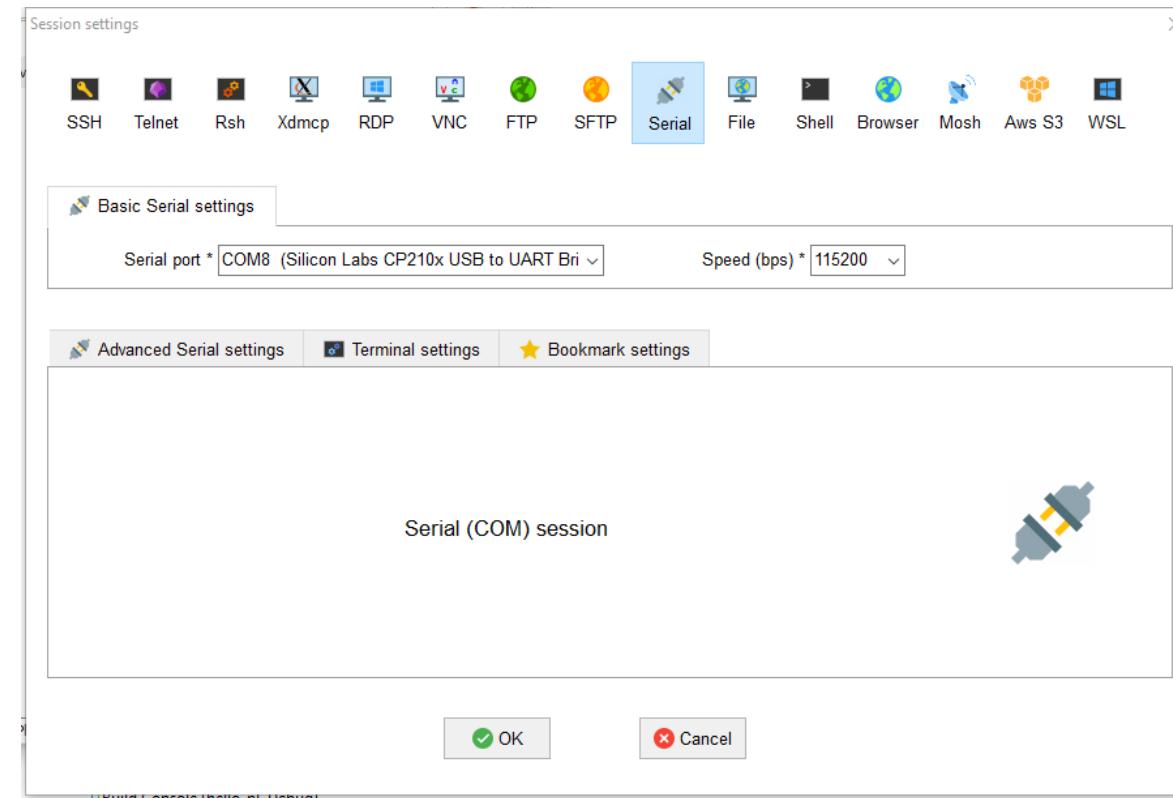


```
 132 //##### Application Starts #####\n\r";
133 print("\r\n");
134 //////////////////////////////////////////////////////////////////
135 //Step-1 :AXI GPIO Initialization
136 //////////////////////////////////////////////////////////////////
137 xStatus = XGpio_Initialize(&GPIOInstance_Ptr,XPAR_AXI_GPIO_0_DEVICE_ID);
138 if(XST_SUCCESS != xStatus)
139 print("GPIO INIT FAILED\r\n");
140 //////////////////////////////////////////////////////////////////
141 //Step-2 :AXI GPIO Set the Direction
142 //////////////////////////////////////////////////////////////////
143 XGpio_SetDataDirection(&GPIOInstance_Ptr, 1,1);
144 //////////////////////////////////////////////////////////////////
145 //Step-3 :AXI Timer Initialization
146 //////////////////////////////////////////////////////////////////
147 xStatus = XTmrCtr_Initialize(&TimerInstancePtr,XPAR_AXI_TIMER_0_DEVICE_ID);
148 if(XST_SUCCESS != xStatus)
149 print("TIMER INIT FAILED \r\n");
150 //////////////////////////////////////////////////////////////////
151 //Step-4 :Set Timer Handler
152 //////////////////////////////////////////////////////////////////
153 XTmrCtr_SetHandler(&TimerInstancePtr,
154 Timer_InterruptHandler,
155 &TimerInstancePtr);
156 //////////////////////////////////////////////////////////////////
157 //Step-5 :Setting timer Reset Value
158 //////////////////////////////////////////////////////////////////
159 XTmrCtr_SetResetValue(&TimerInstancePtr,
160 0, //Change with generic value
161 0xf0000000);
162 //////////////////////////////////////////////////////////////////
163 //Step-6 :Setting timer Option (Interrupt Mode And Auto Reload )
164 //////////////////////////////////////////////////////////////////
165 XTmrCtr_SetOptions(&TimerInstancePtr,
166 XPAR_AXI_TIMER_0_DEVICE_ID,
167 (XTC_INT_MODE_OPTION | XTC_AUTO_RELOAD_OPTION ));
168 //////////////////////////////////////////////////////////////////
169 //Step-7 :PS GPIO Initialization
170 //////////////////////////////////////////////////////////////////
171 GpioConfigPtr = XGpioPs_LookupConfig(XPAR_PS7_GPIO_0_DEVICE_ID);
172 if(GpioConfigPtr == NULL)
173 return XST_FAILURE;
174 xStatus = XGpioPs_CfgInitialize(&psGpioInstancePtr,
```

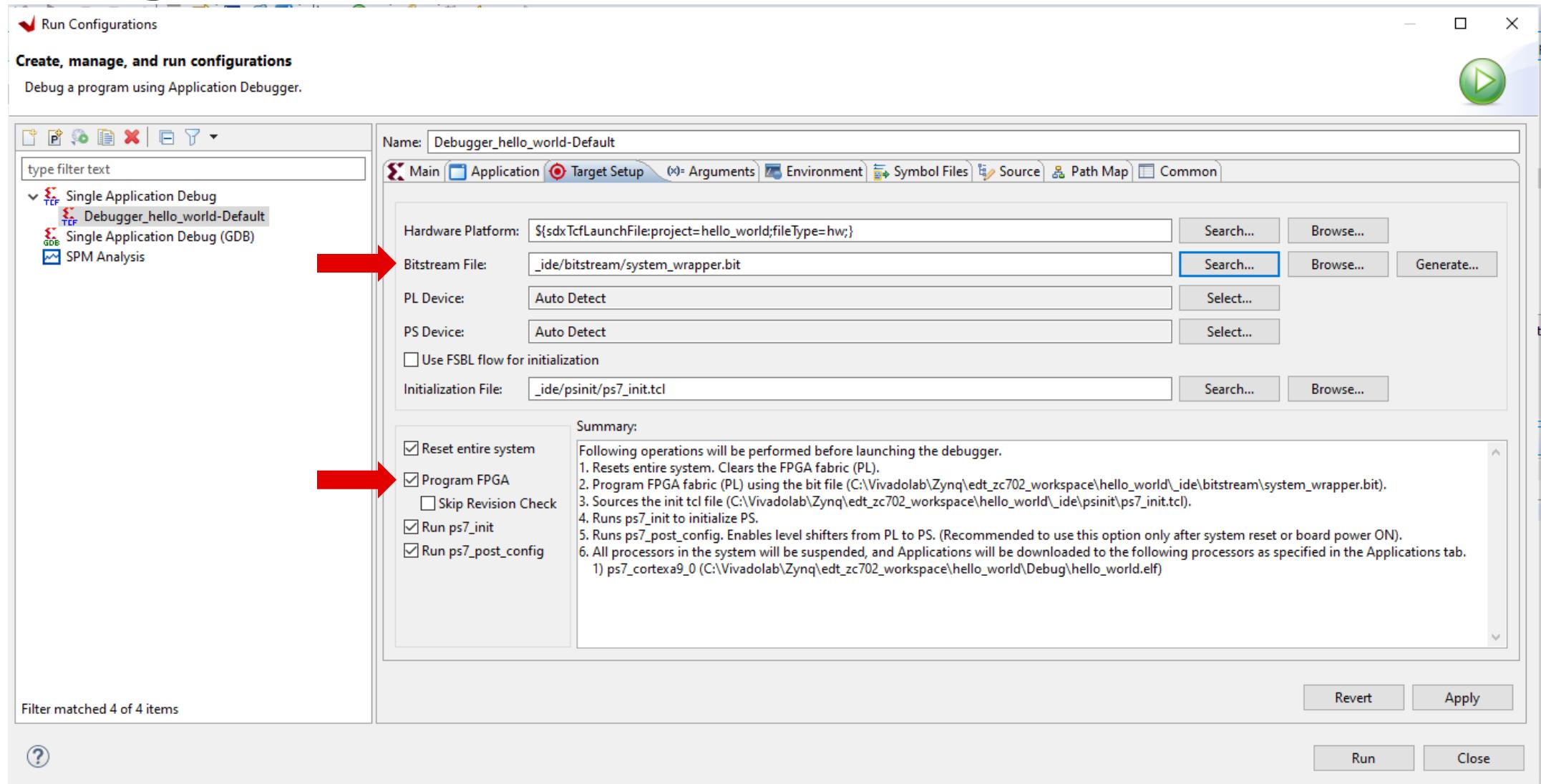
Testing the PL IP with Prepared Software



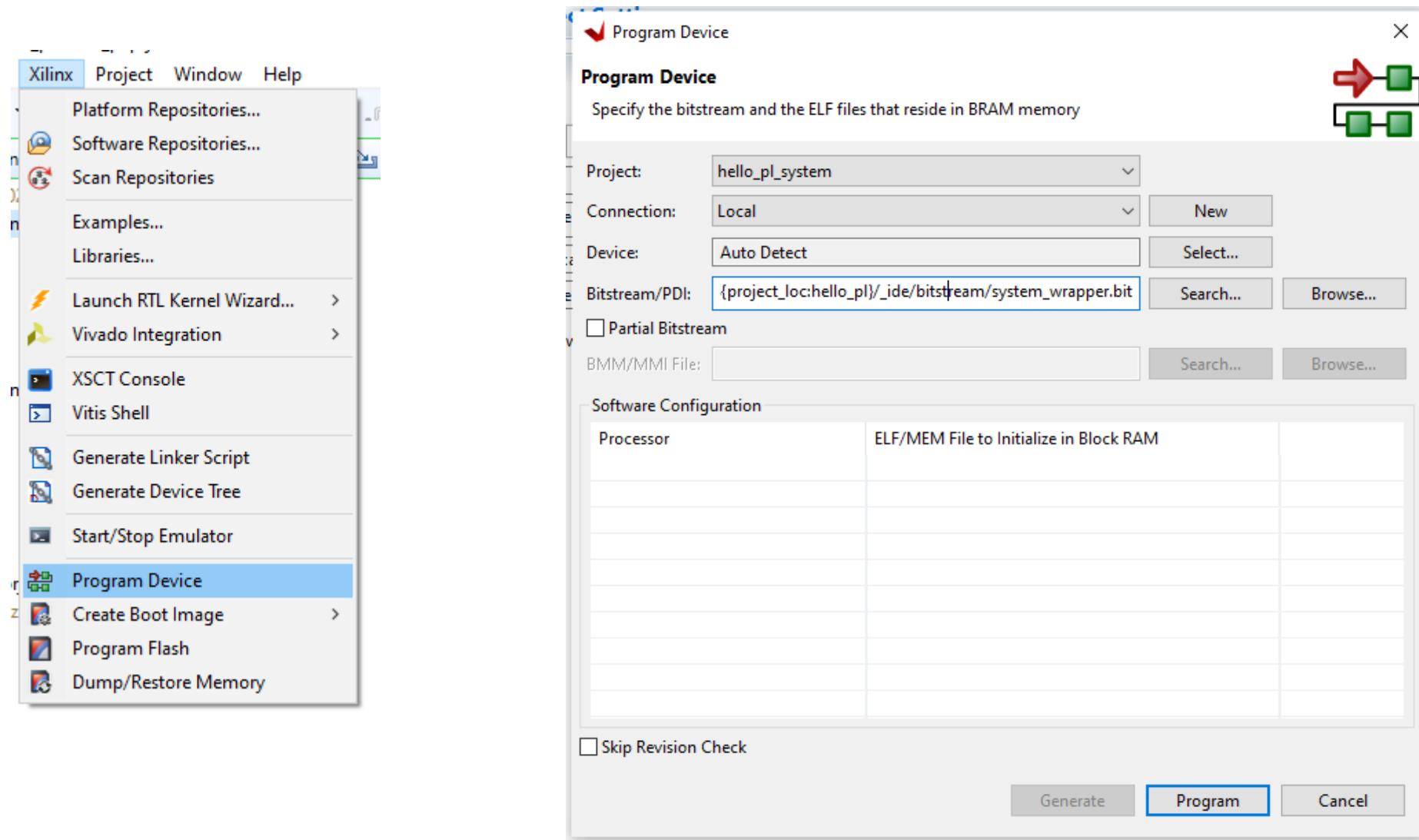
Testing the PL IP with Prepared Software



Testing the PL IP with Prepared Software



Testing the PL IP with Prepared Software



Testing the PL IP with Prepared Software



Reference

Zynq-7000 SoC Family Product Selection Guide

Zynq-7000 SoC Data Sheet: Overview (DS190)

Zynq 7000 SoC Technical Reference Manual(UG585)

AMBA AXI4 Interface Protocol

AXI GPIO v2.0 LogiCORE IP Product Guide (PG144)

Zynq-7000 Embedded Design Tutorial

Embedded Design Documentation GitHub

Vitis Drivers API Documentation - gpio



Thank you very much for your attention!

Course Agenda
2023