AMD
XILINX

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

AMD
XILINX

Zynq 7000 Overview

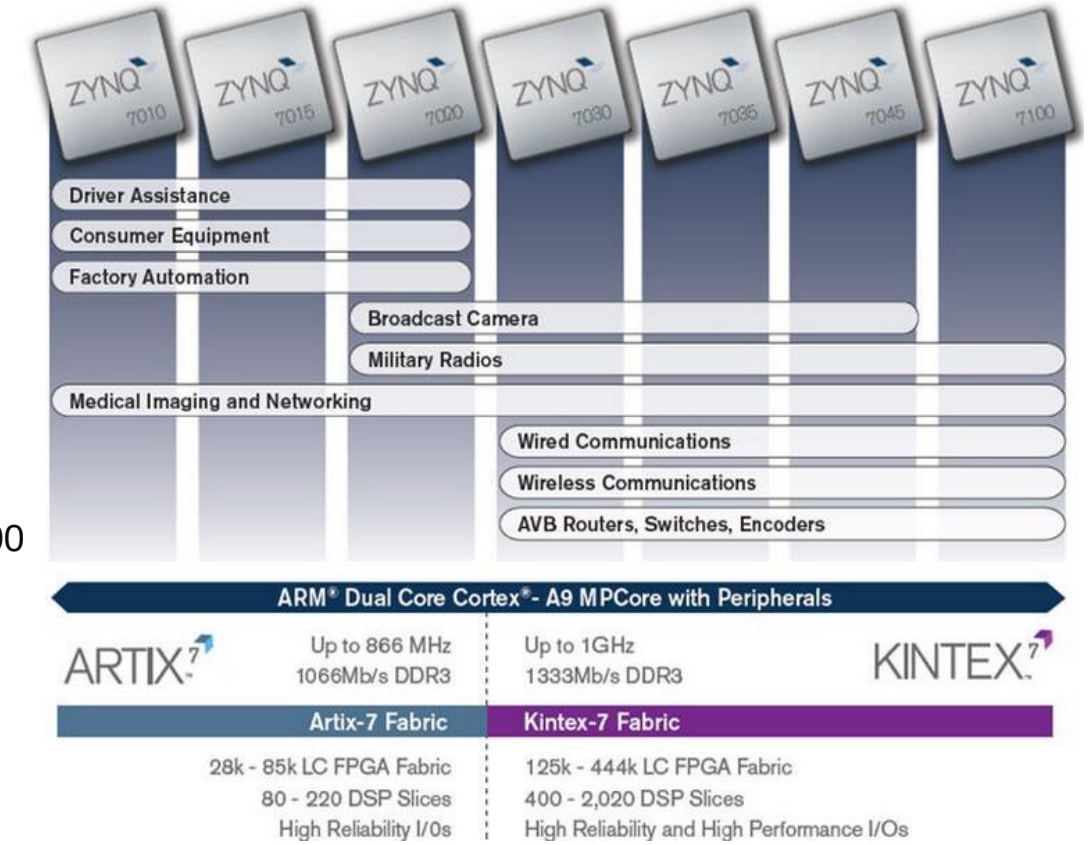
Course Agenda
2023

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)



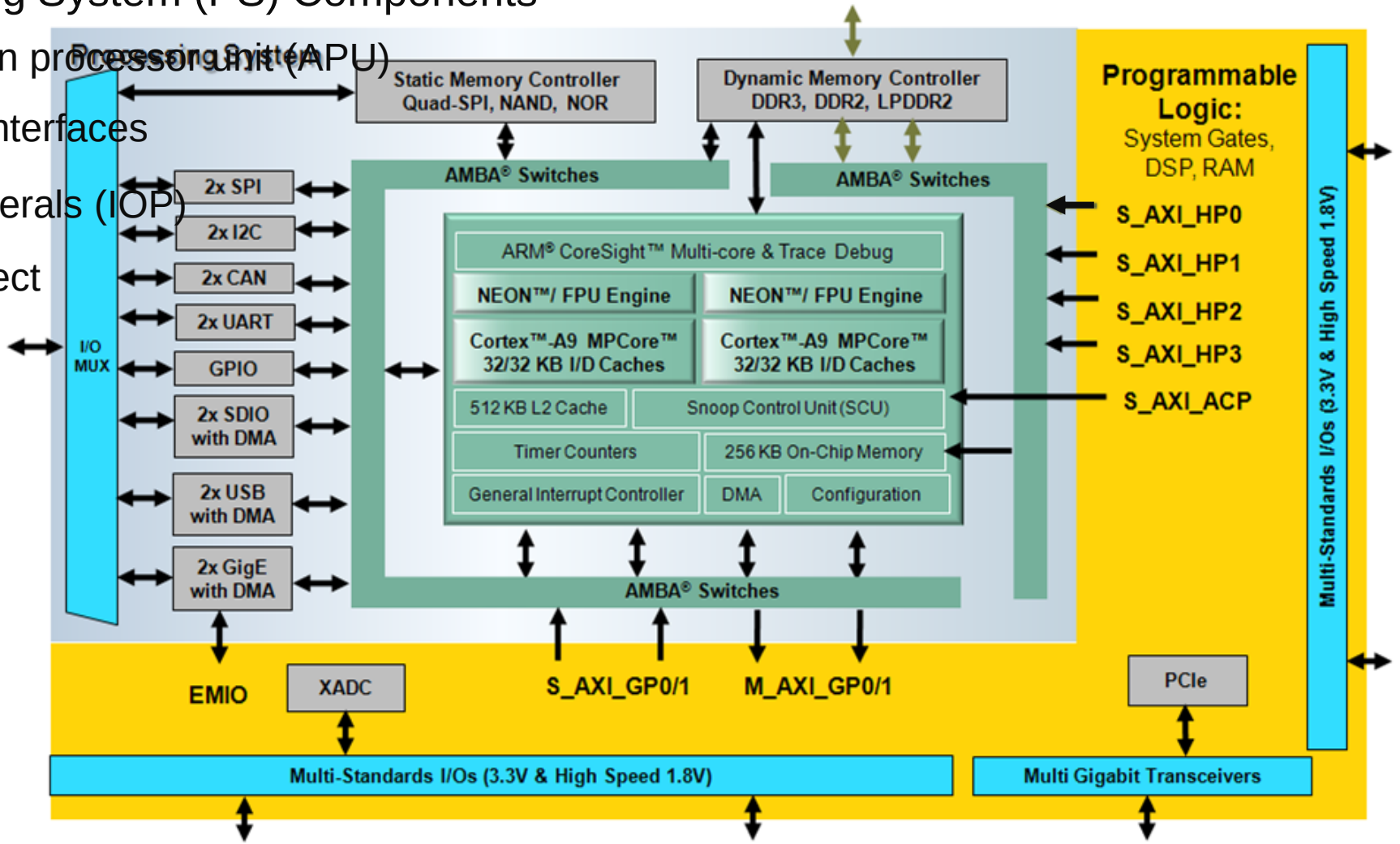
AMD
XILINX

Processing System (PS) Components

Course Agenda
2023

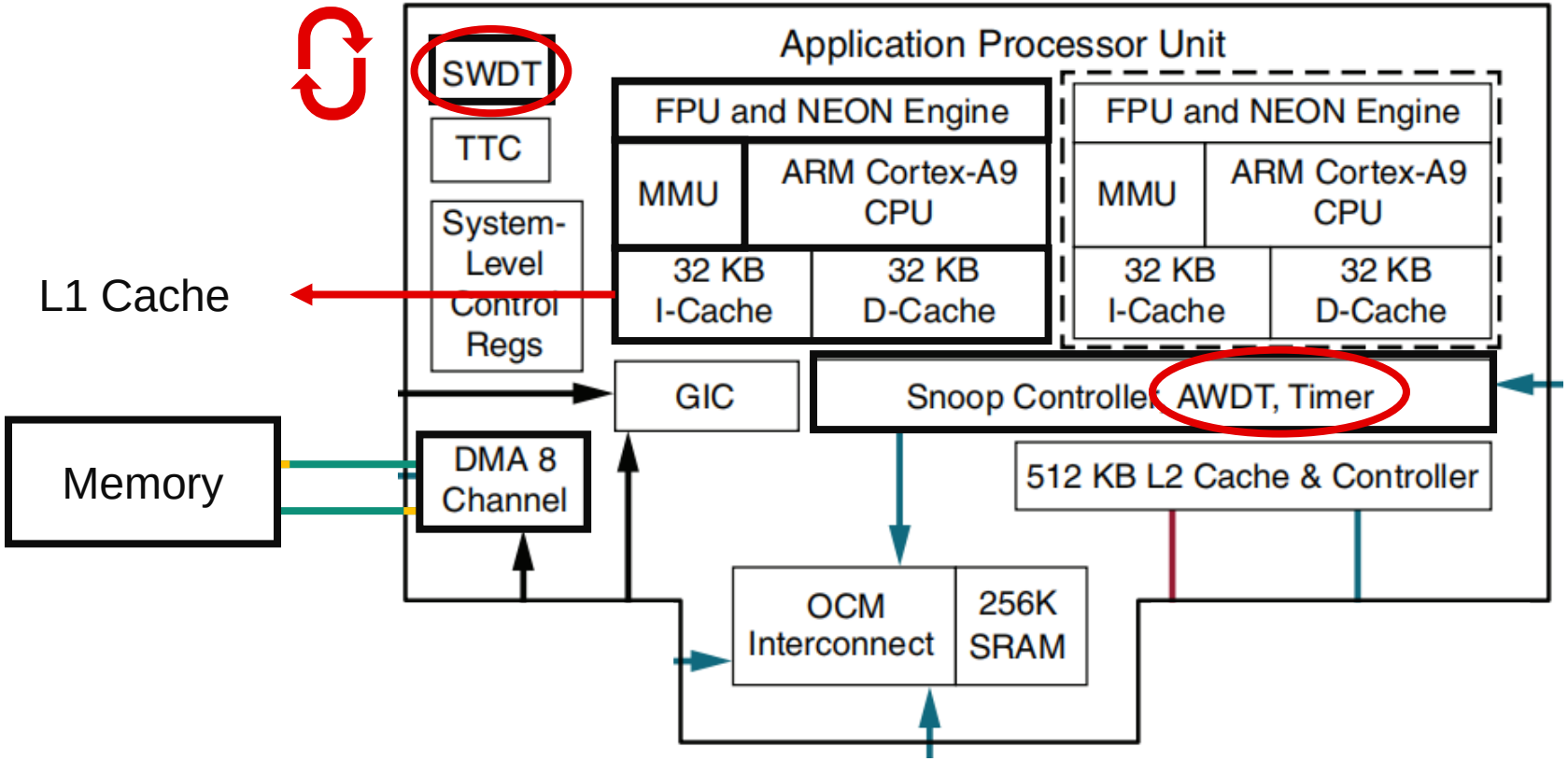
Zynq-7000 AP SoC Block Diagram

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



AMD XILINX

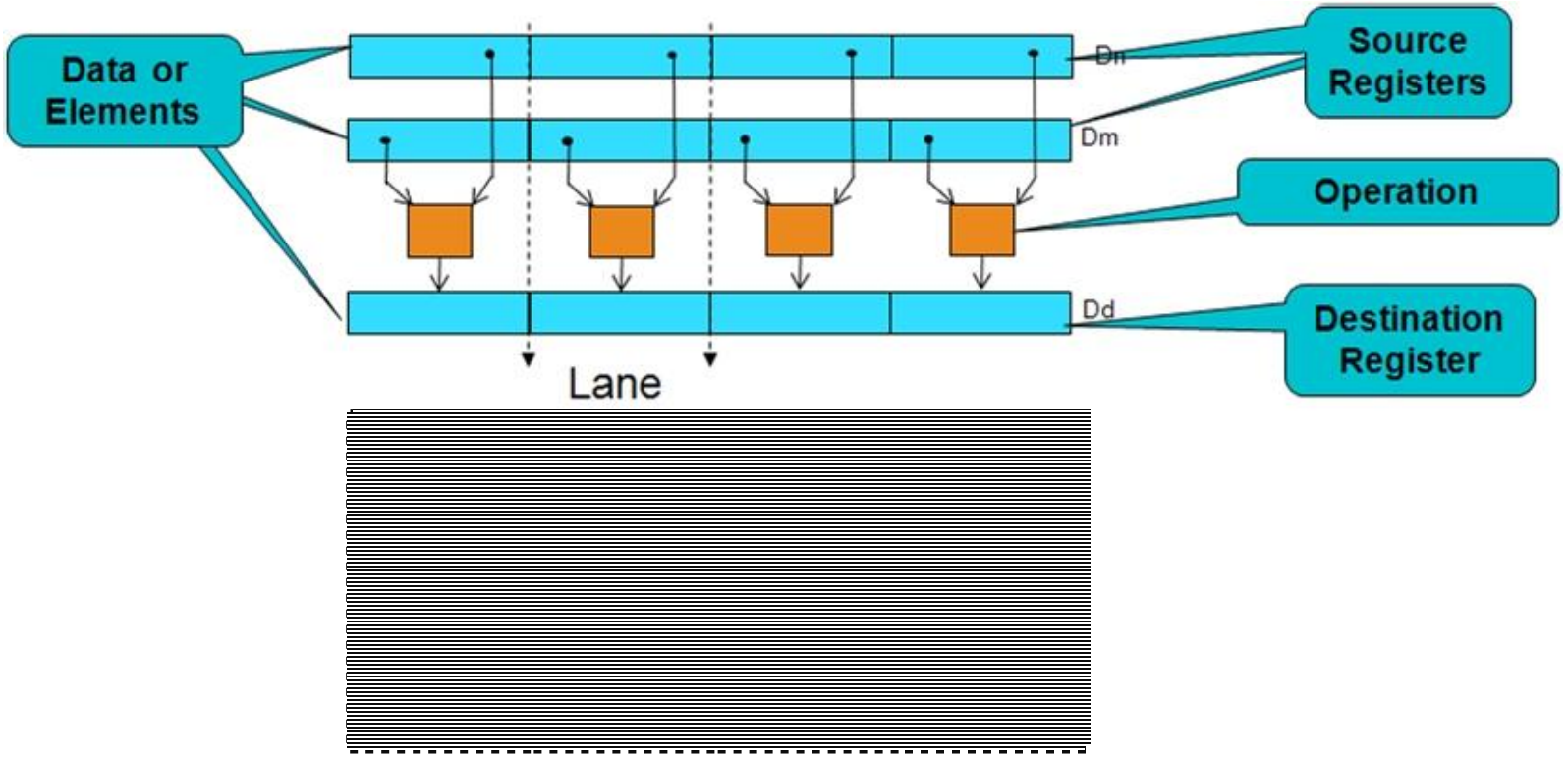
Application processor unit (APU)



AMD XILINX

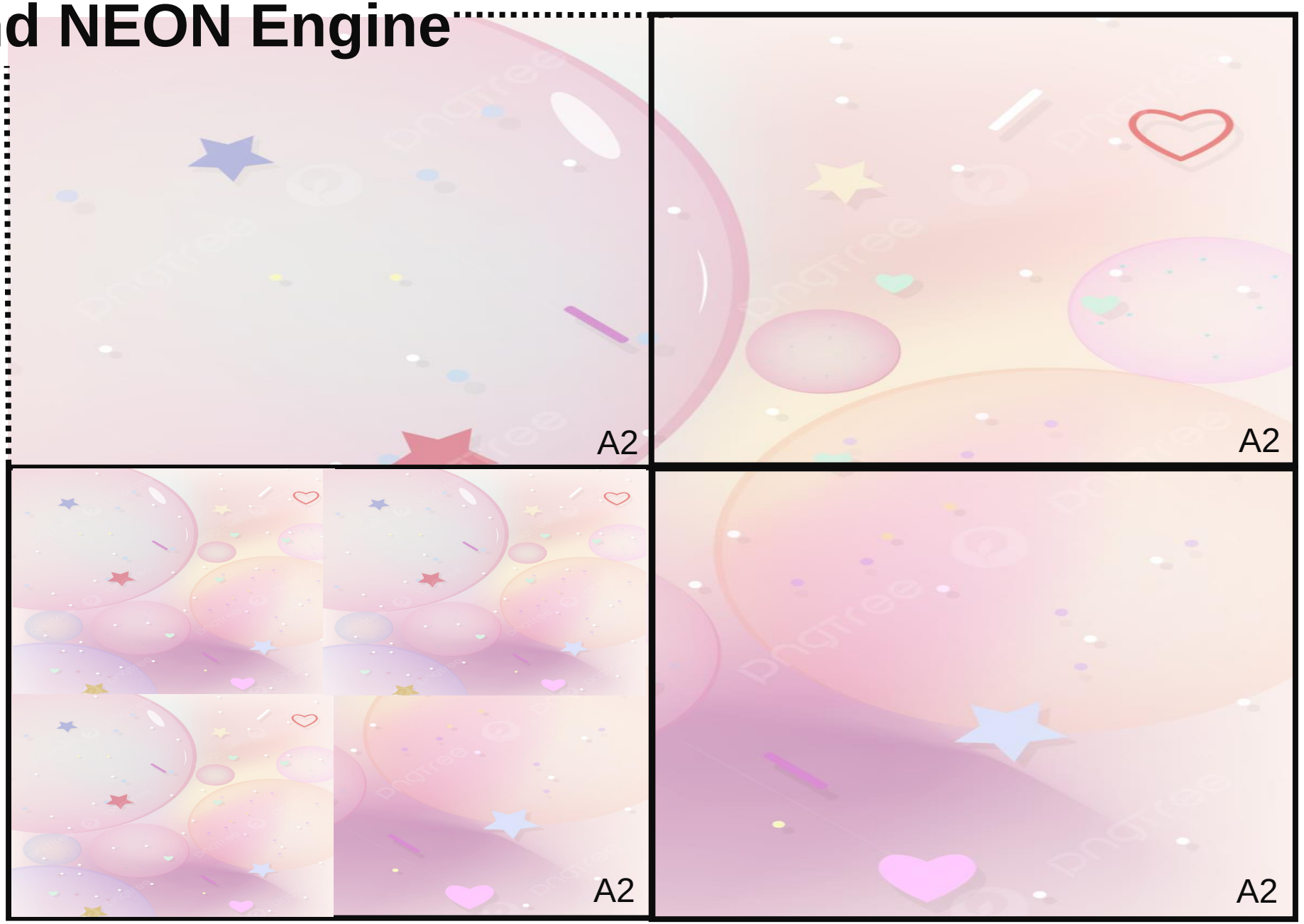
FPU and NEON Engine

- Single Instruction Multiple Data (SIMD)



AMD XILINX

FPU and NEON Engine



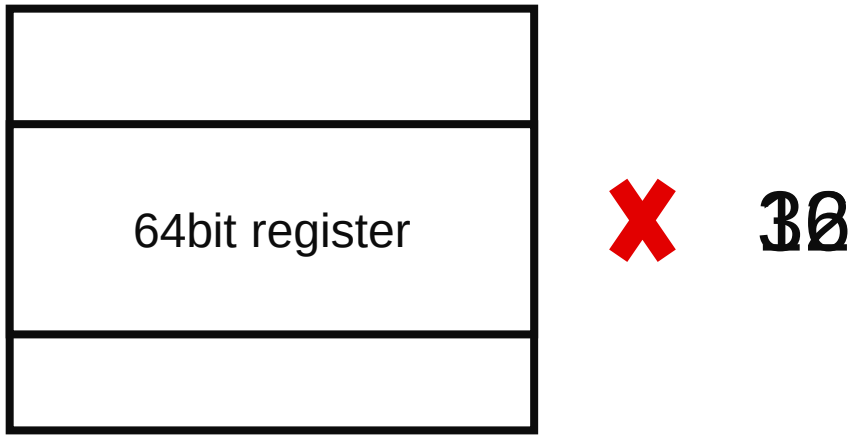
AMD
XILINX

FPU and NEON Engine

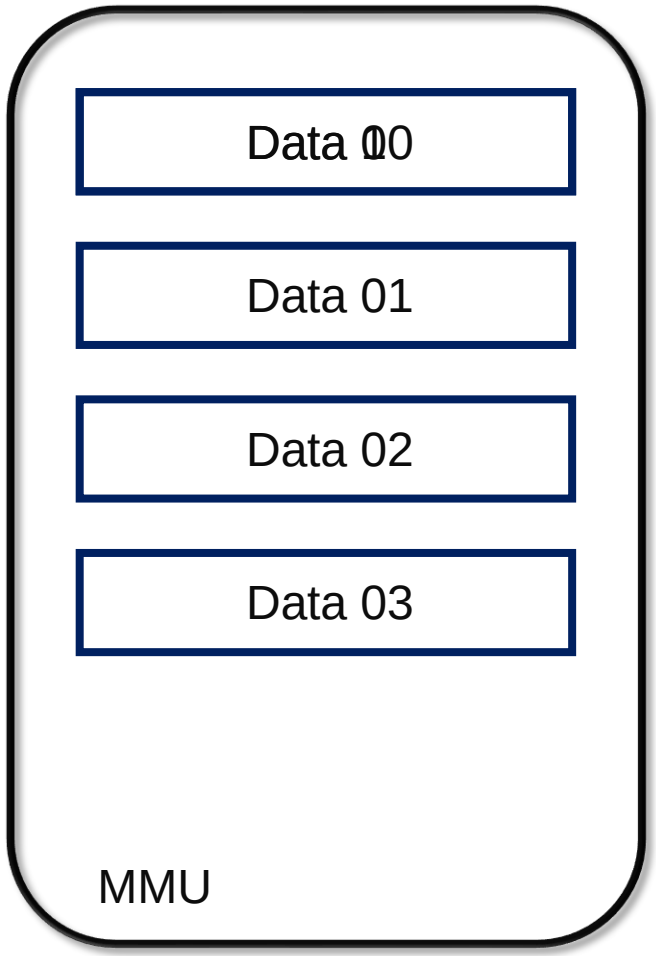
NEON technology is a wide single instruction,

multiple data (SIMD) parallel and co-processing architecture

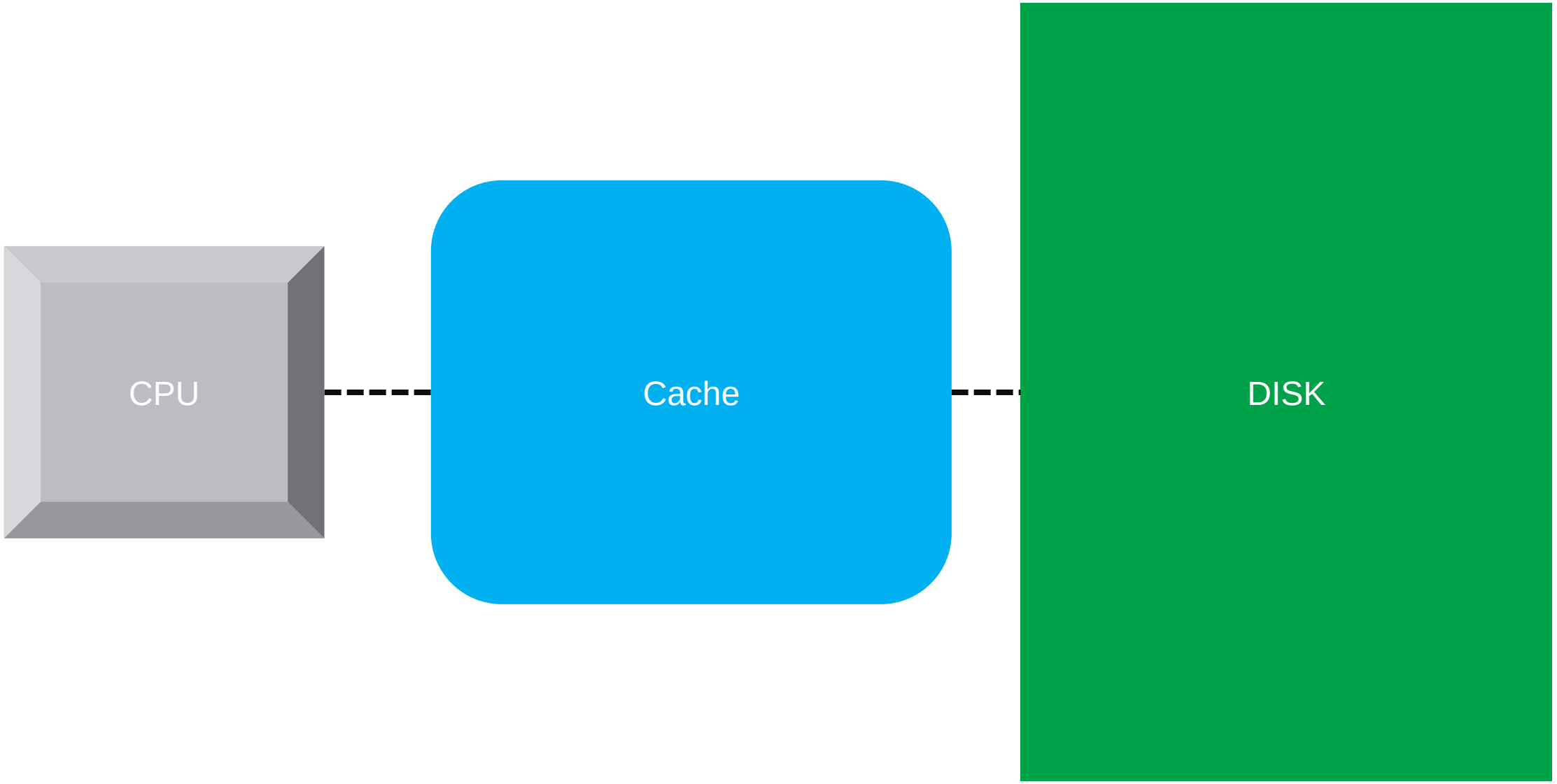
- 32 registers, 64-bits wide (dual view as 16 registers, 128-bits wide)
- Data types can be: signed/unsigned 8-bit, 16-bit, 32-bit, 64-bit, or 32-bit float



Memory Management Unit



What is Cache?



Cache Hit rate

25%

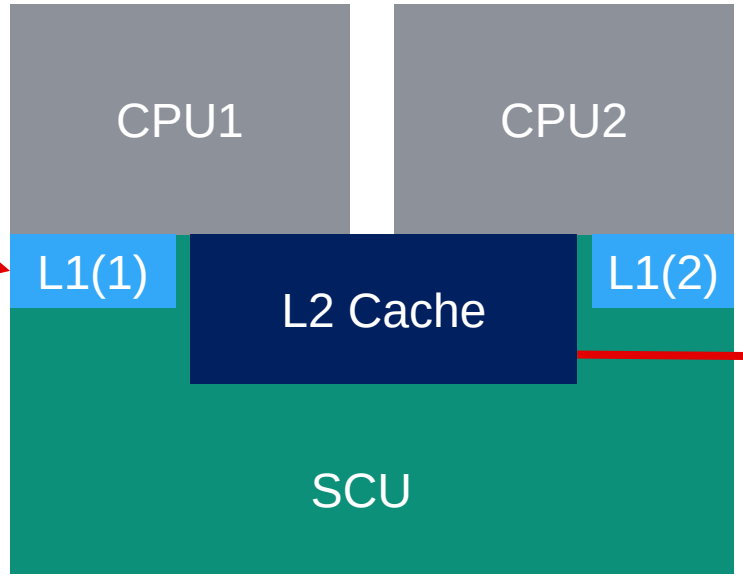
70%

5%

?%

AMD
XILINX

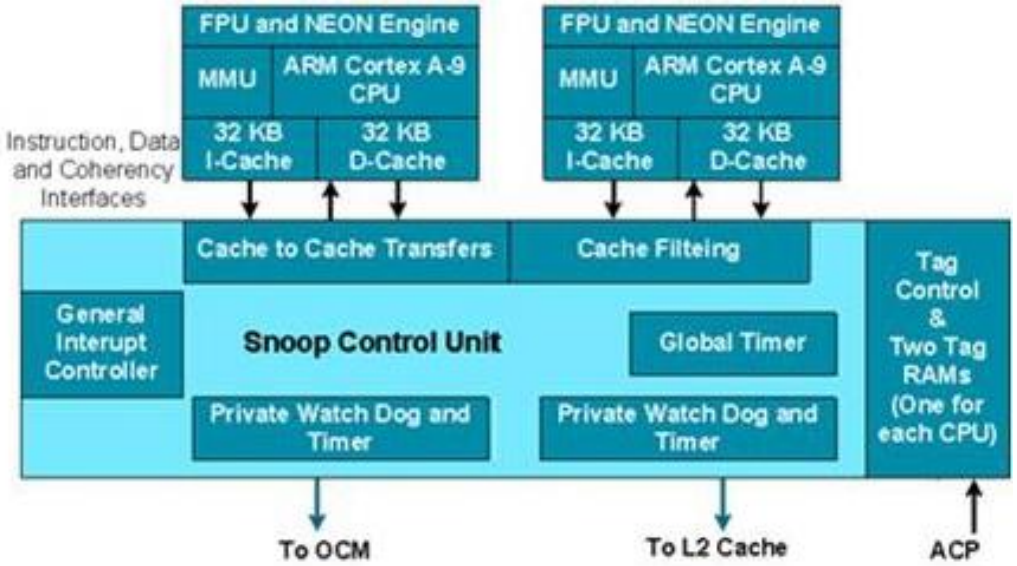
Snoop Control Unit (SCU)



AMD
XILINX

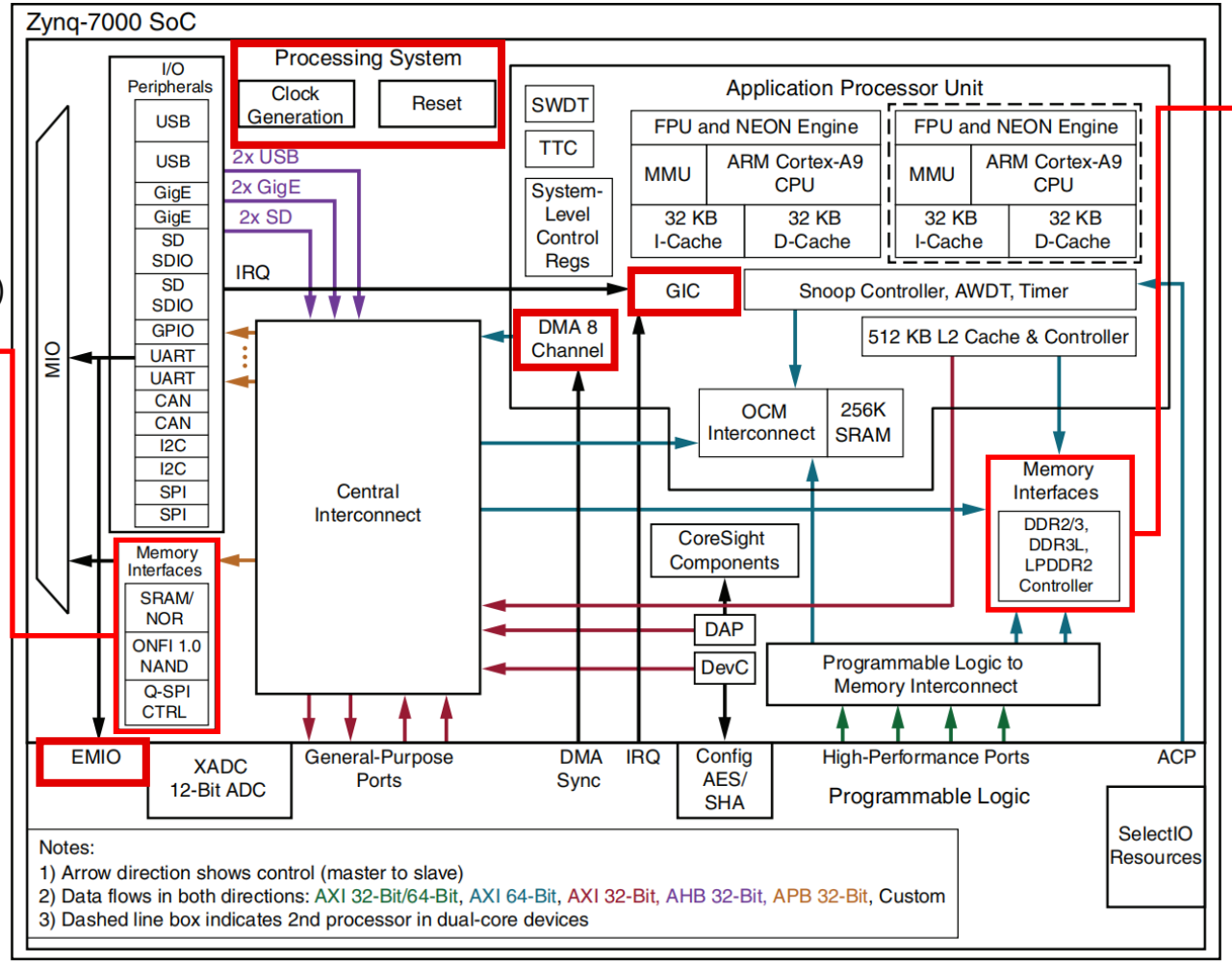
Snoop Control Unit (SCU)

- 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）

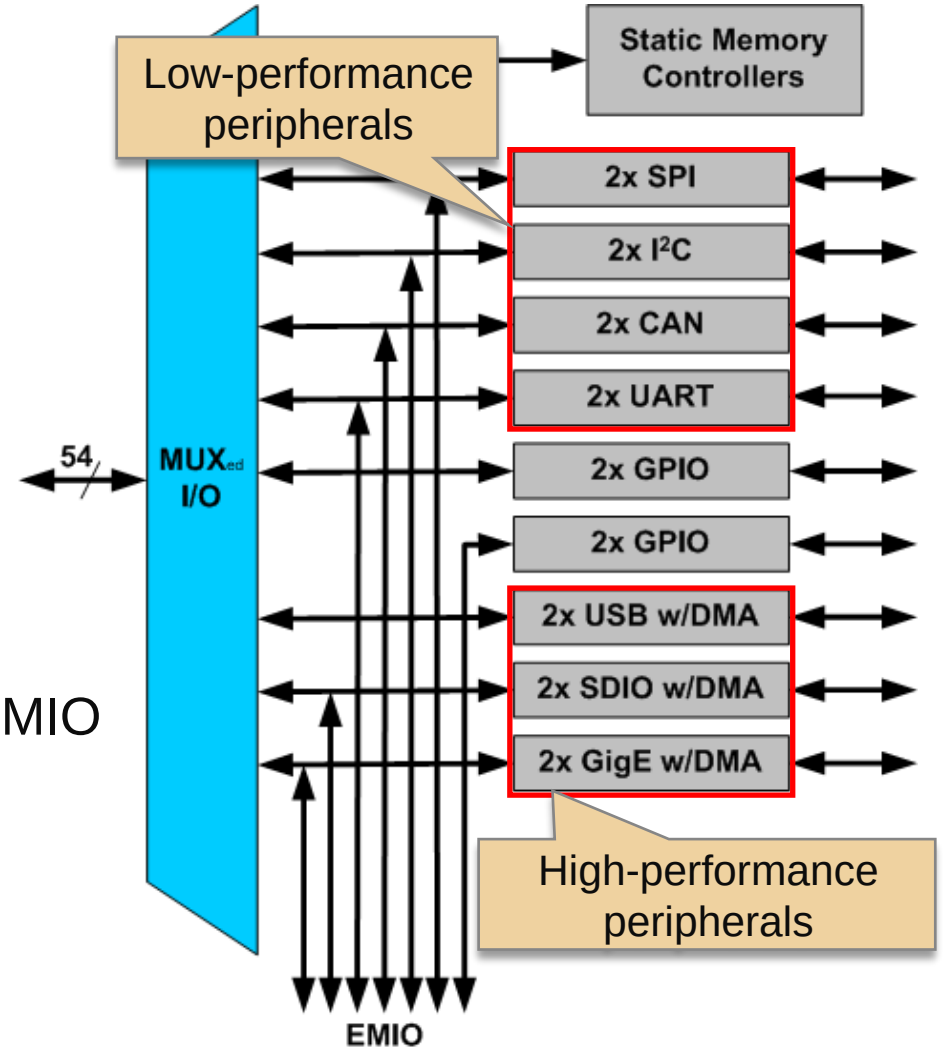


- DDR2/3
- DDR3L
- LPDDR2

AMD XILINX

Zynq Built-in Peripherals

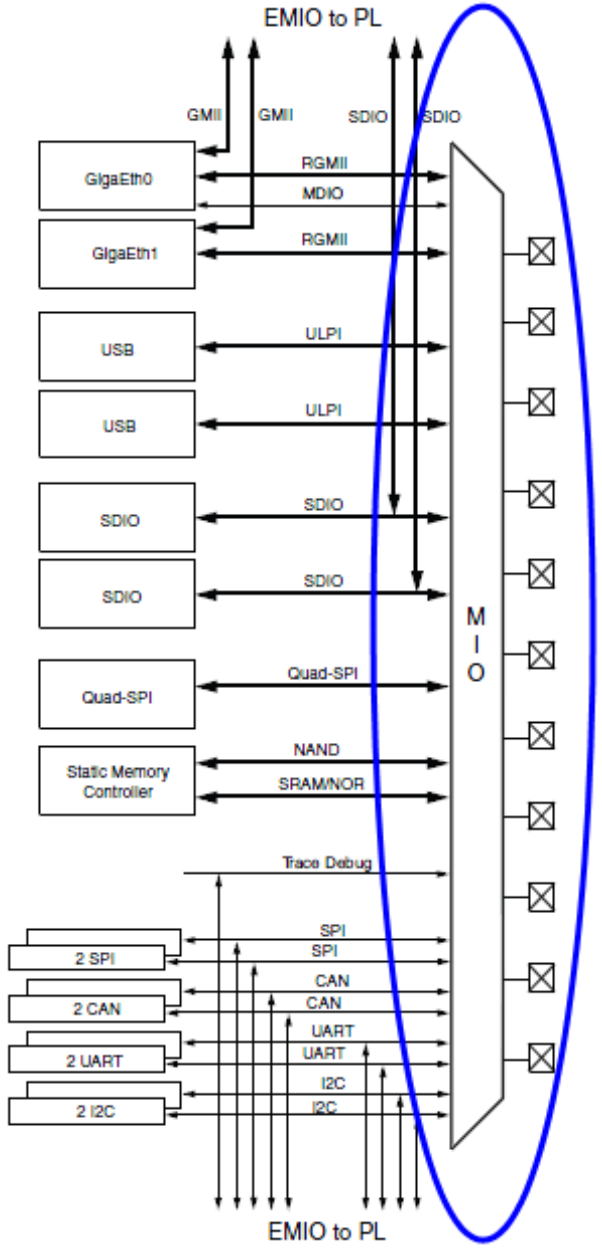
- Two USB 2.0 OTG/device/host
- Two tri-mode gigabit Ethernet (10/100/1000)
- Two SD/SDIO interfaces
  - Memory, I/O, and combo cards
- Two CAN 2.0Bs, SPIs, I2Cs, UARTs
- Four GPIO 32-bit blocks
  - 54 available through MIO; other 64 available through EMIO



AMD XILINX

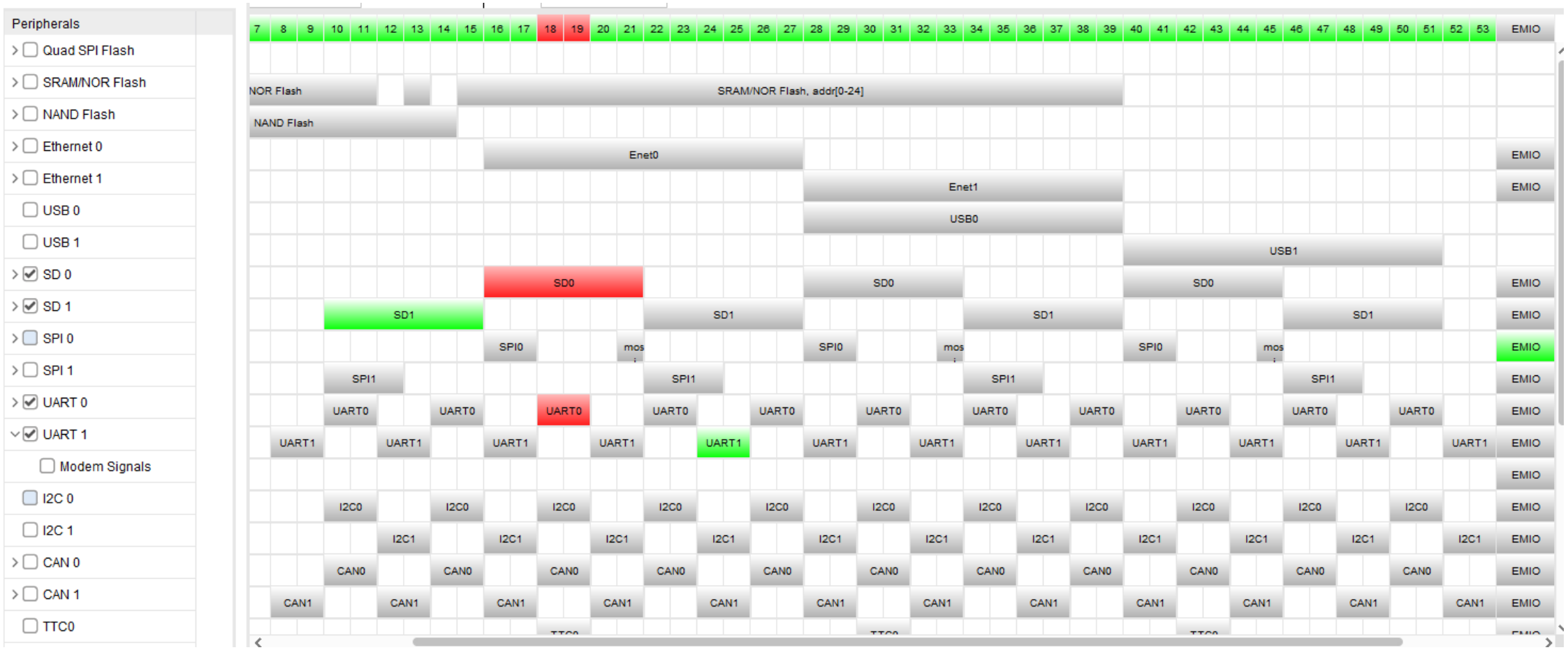
Multiplexed I/O (MIO)

- 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



AMD XILINX

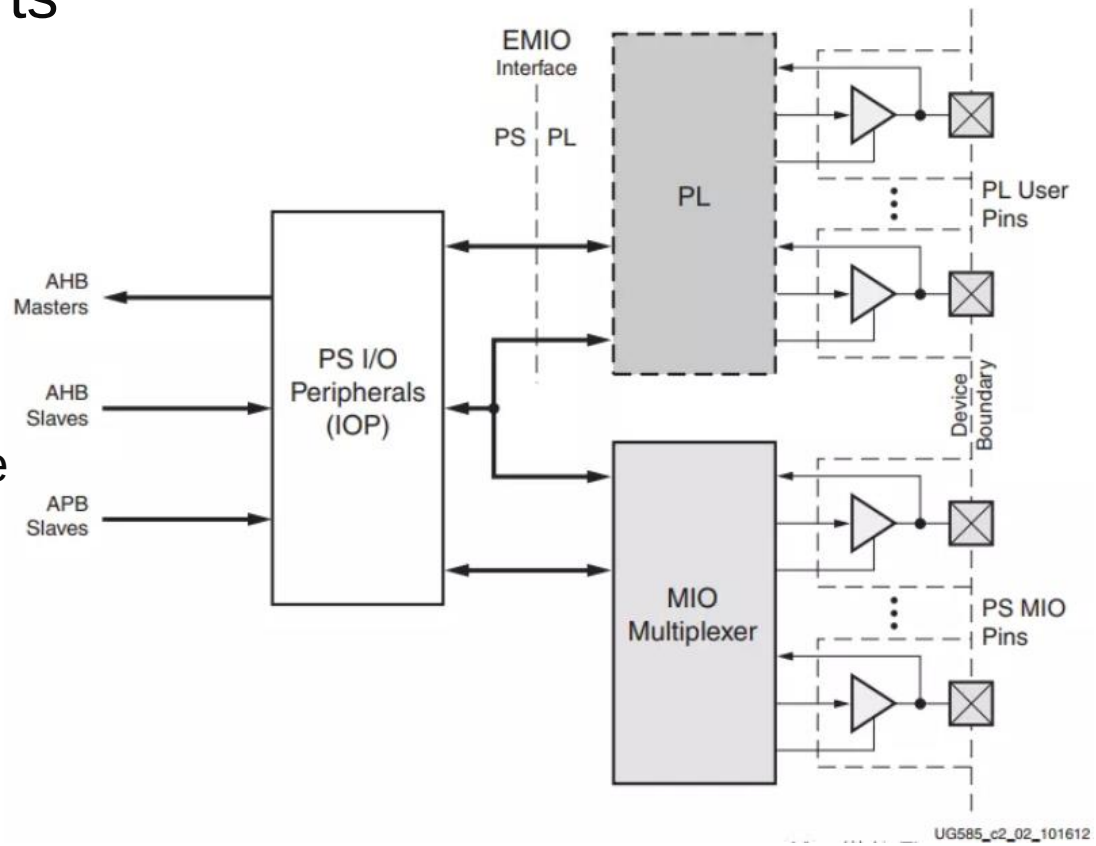
Multiplexed I/O (MIO)



AMD XILINX

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)

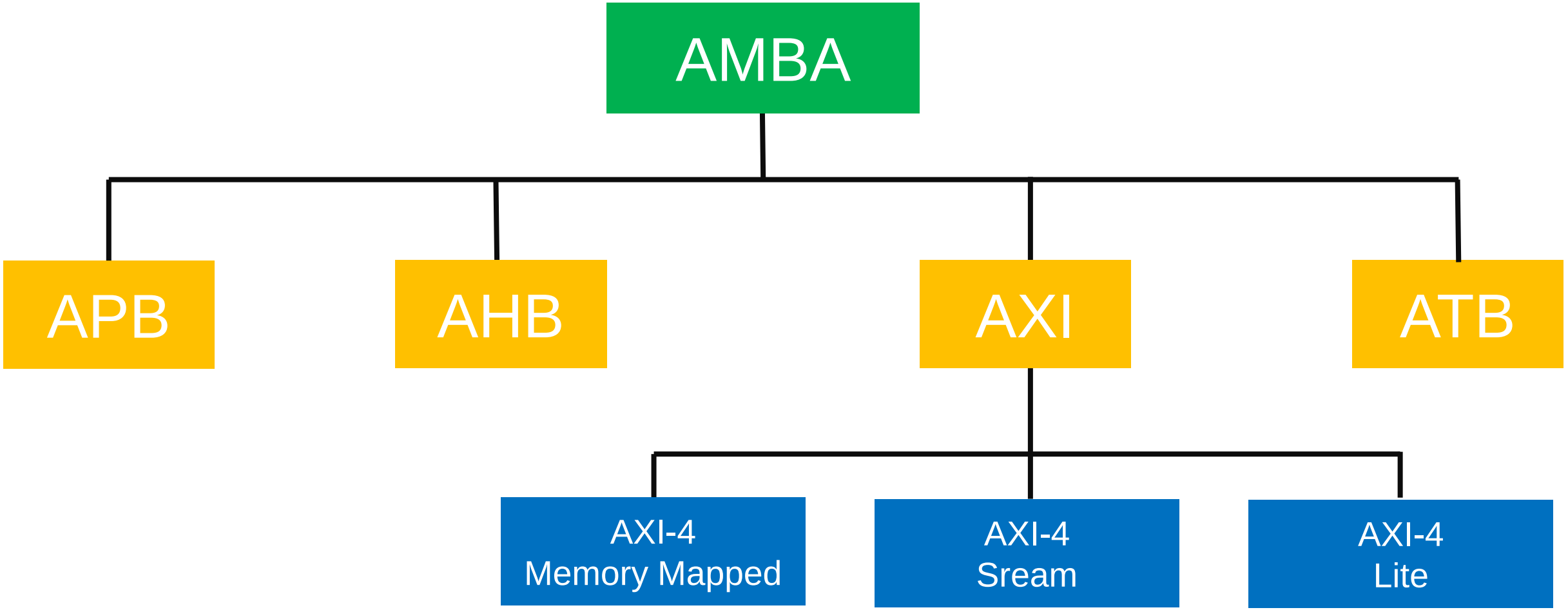
IP	MIO	Extendable MIO in Programmable Logic
QSPI NOR/SRAM NAND	Yes	No
USB: 0,1	Yes, Phy off chip	No
SDIO: 0,1	Yes – 50MHz	Yes – 25MHz
SPI: 0,1 I2C: 0,1 CAN: 0,1 GPIO	Yes	Yes
GigE: 0,1	RGMII v2.0 (HSTL) Phy off chip	Supports GMII, RGMII v2.0 (HSTL), RGMII v1.3 (LVCMOS), RMII, MII, SGMII with wrapper in Programmable Logic
UART: 0,1	Simple UART: Only 2 pins (Tx & Rx)	Full UART (Tx, Rx, DTR, DCD, DSR, RI, RTS & CTS) either require: • 2 Processing System pins (Rx & Tx) through MIO + 6 additional Programmable Logic pins • 8 Programmable Logic pins

AMD
XILINX

PS – PL Interfaces

Course Agenda
2023

AXI-Memory Mapped, Streaming , Lite

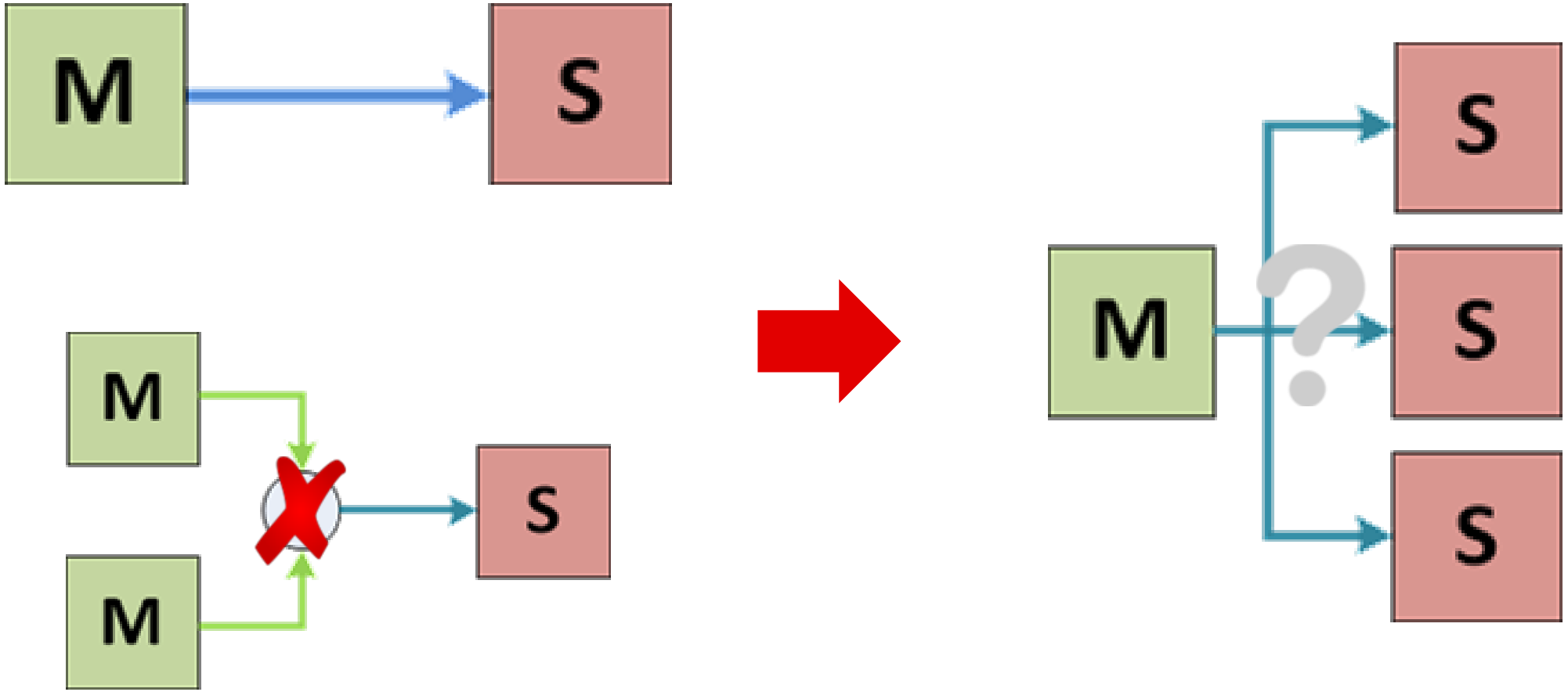


AXI-Memory Mapped/Full

AXI-Memory Lite

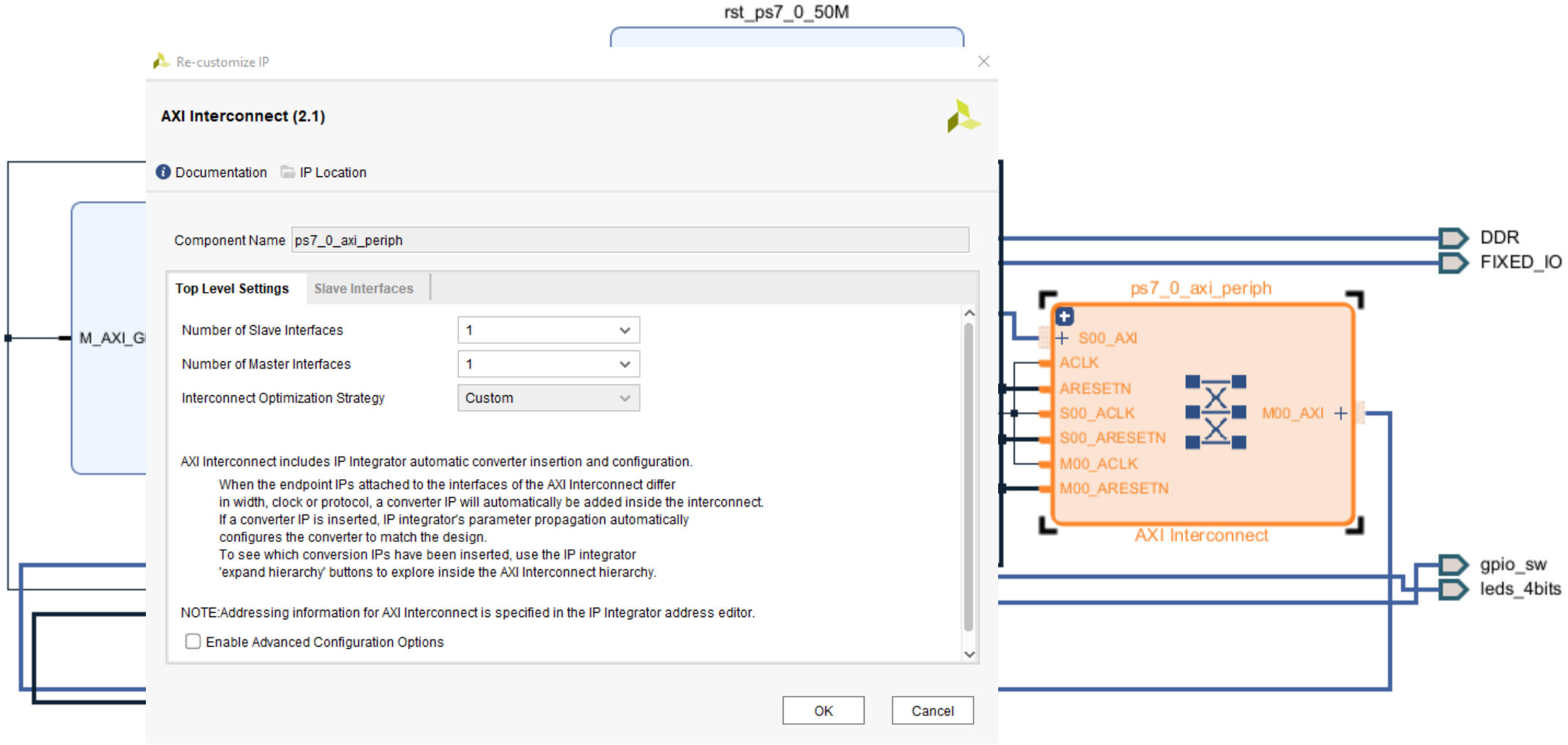
AXI-Streaming

AXI-Interconnect



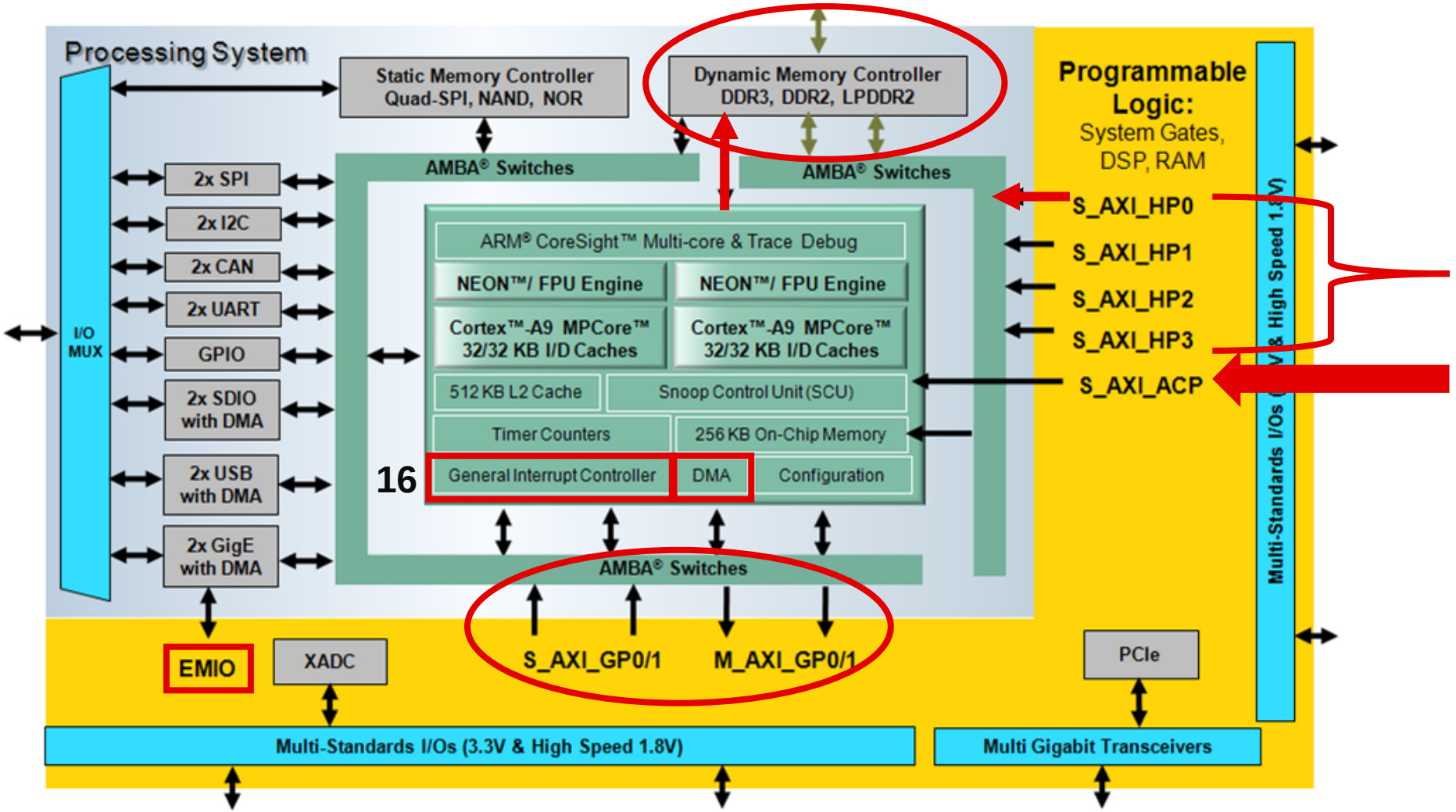
AMD
XILINX

AXI-Interconnect



AMD XILINX

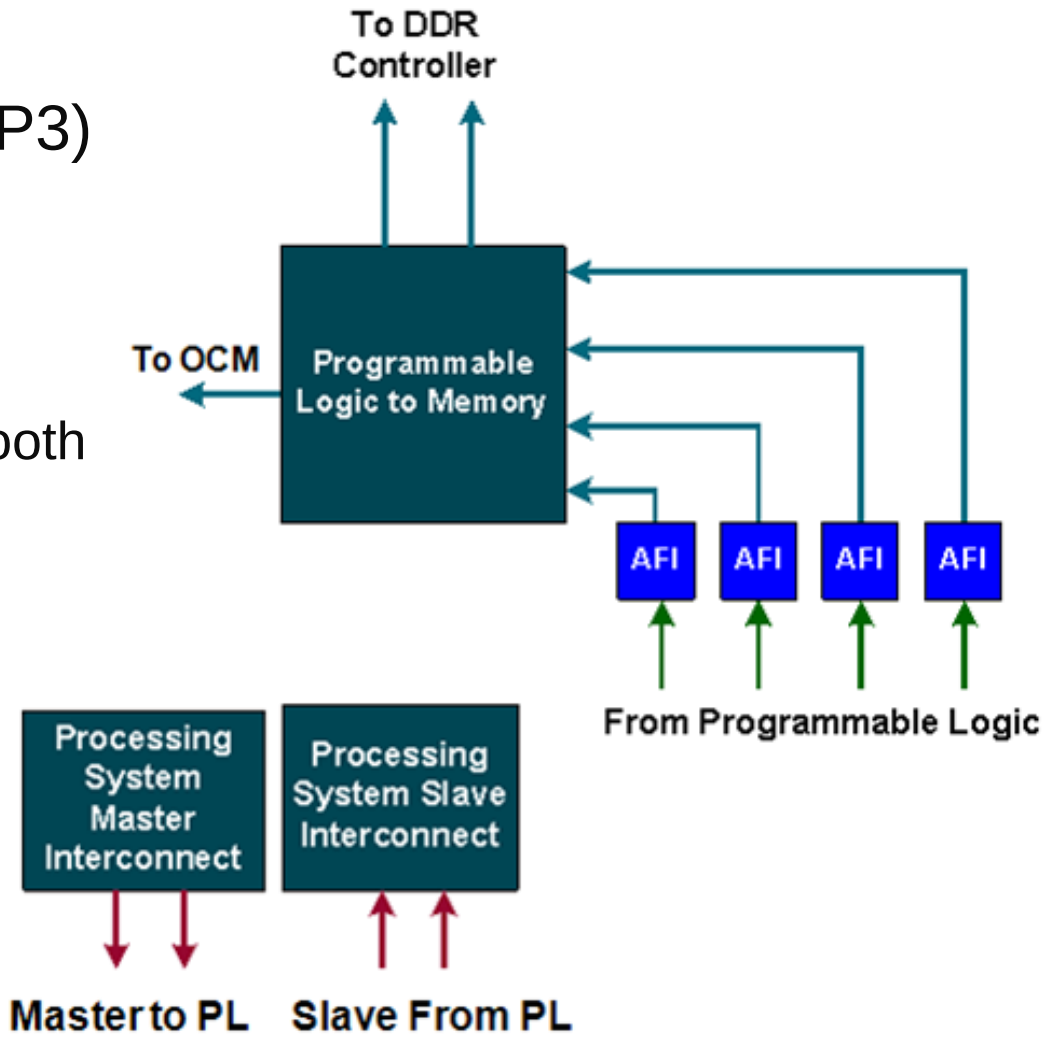
PS – PL Interconnect



AMD XILINX

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

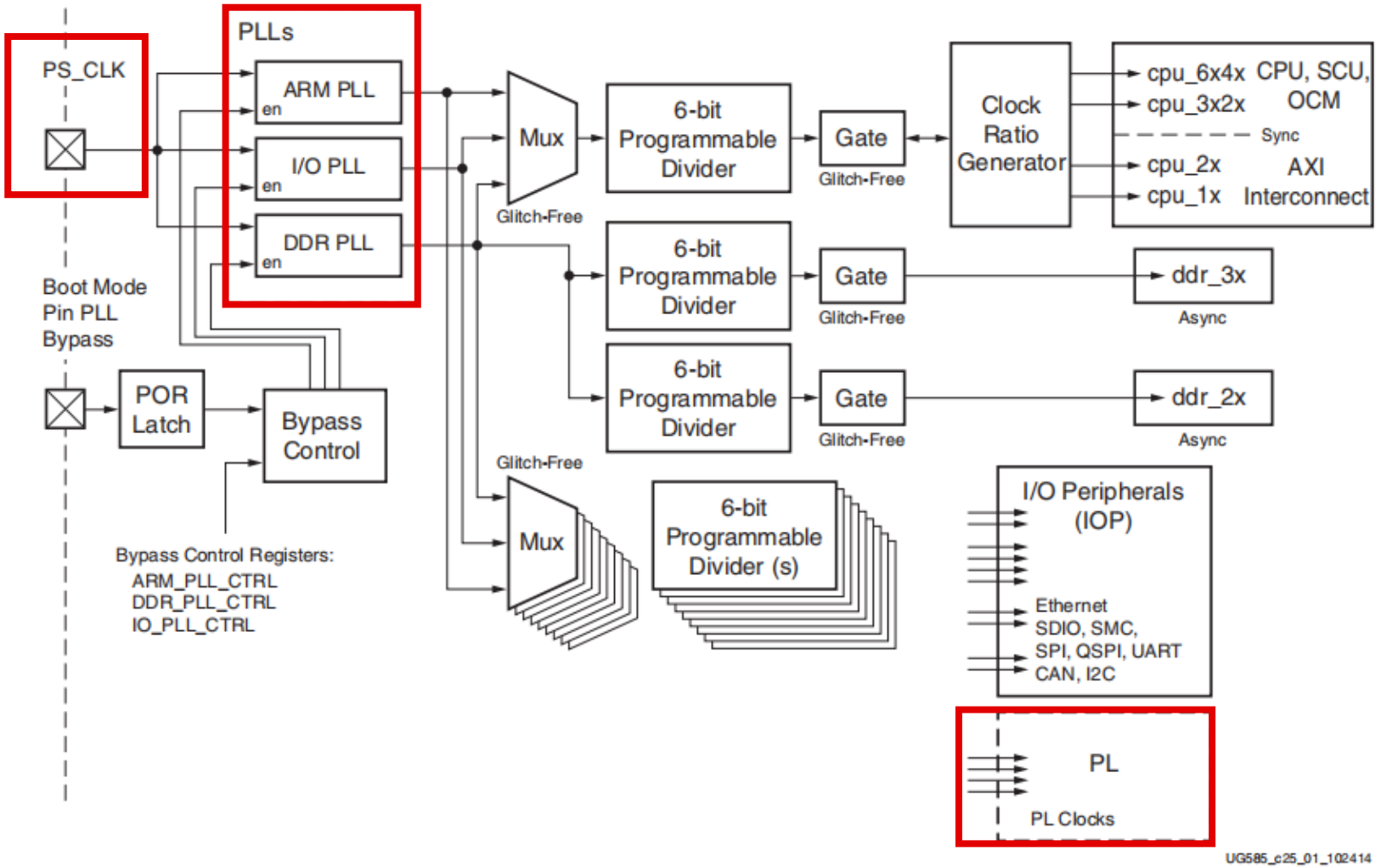


Figure 25-1: **PS Clock System Block Diagram**

AMD
XILINX

Clock and Reset

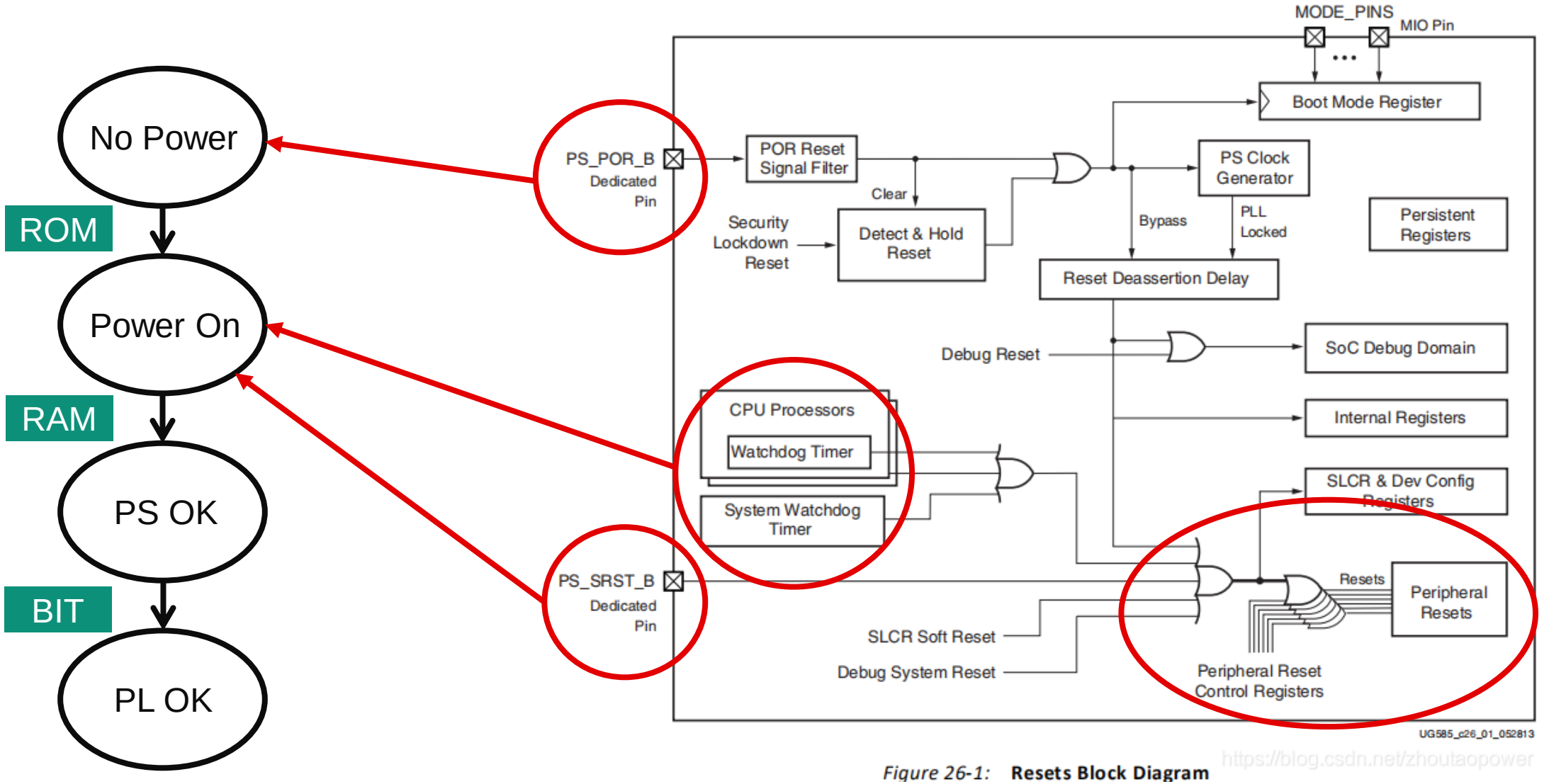


Figure 26-1: **Resets Block Diagram**

AMD XILINX

AMD
XILINX

Zynq Boot and Configuration

Course Agenda
2023

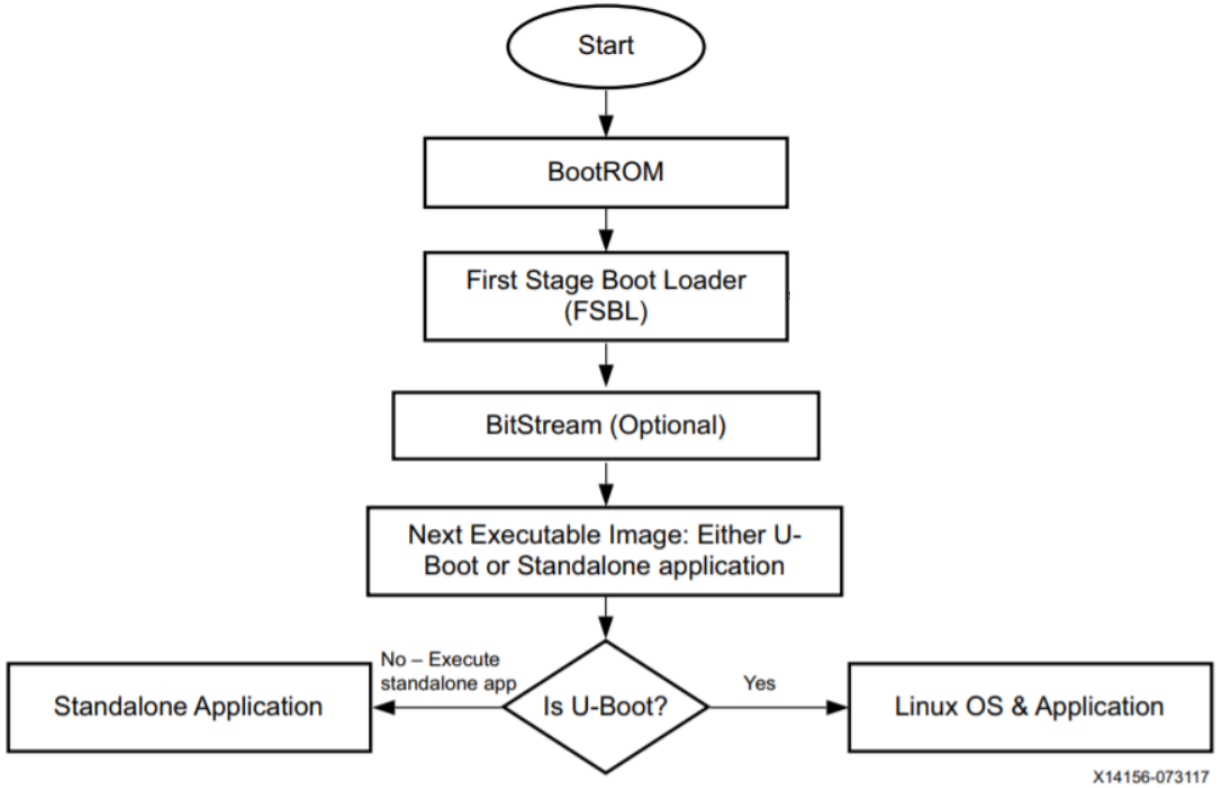
Zynq Boot and Configuration

- 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)

Second Stage Boot Loader (SSBL)

Bitstream

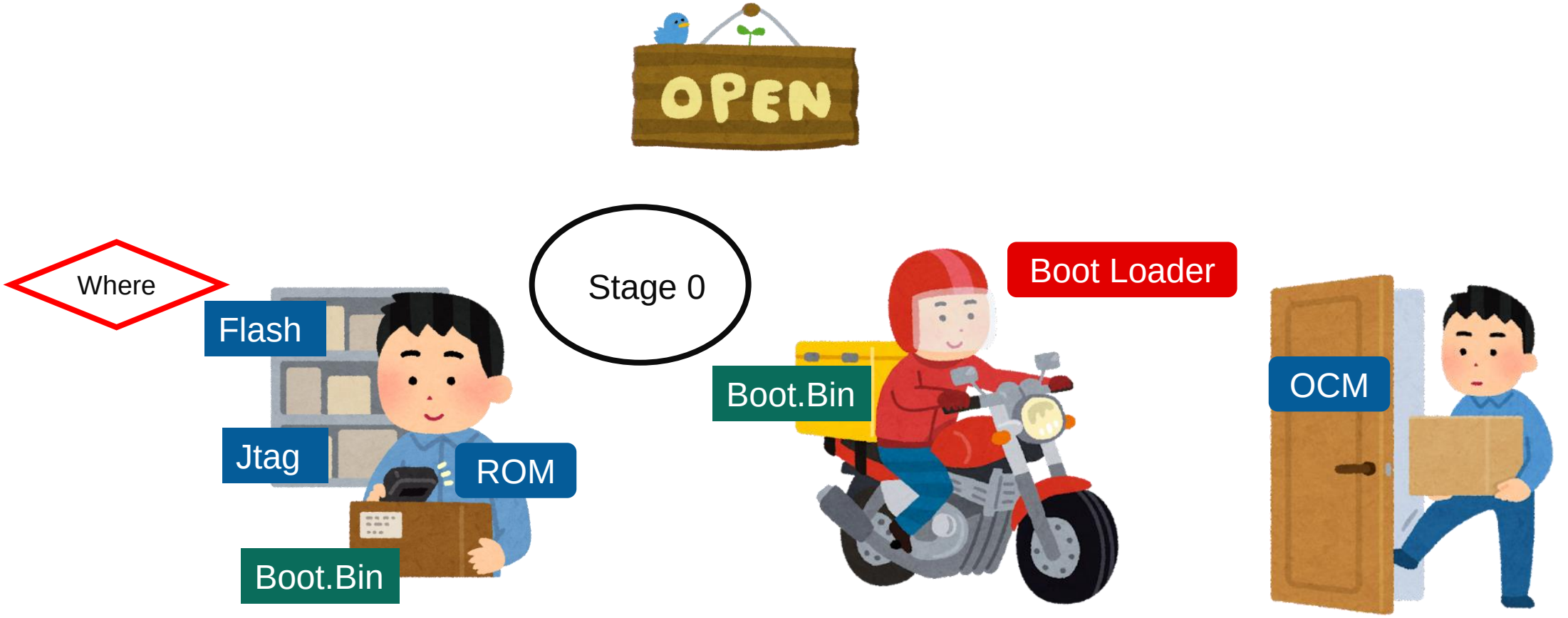
First Stage Boot Loader (FSBL)

- 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

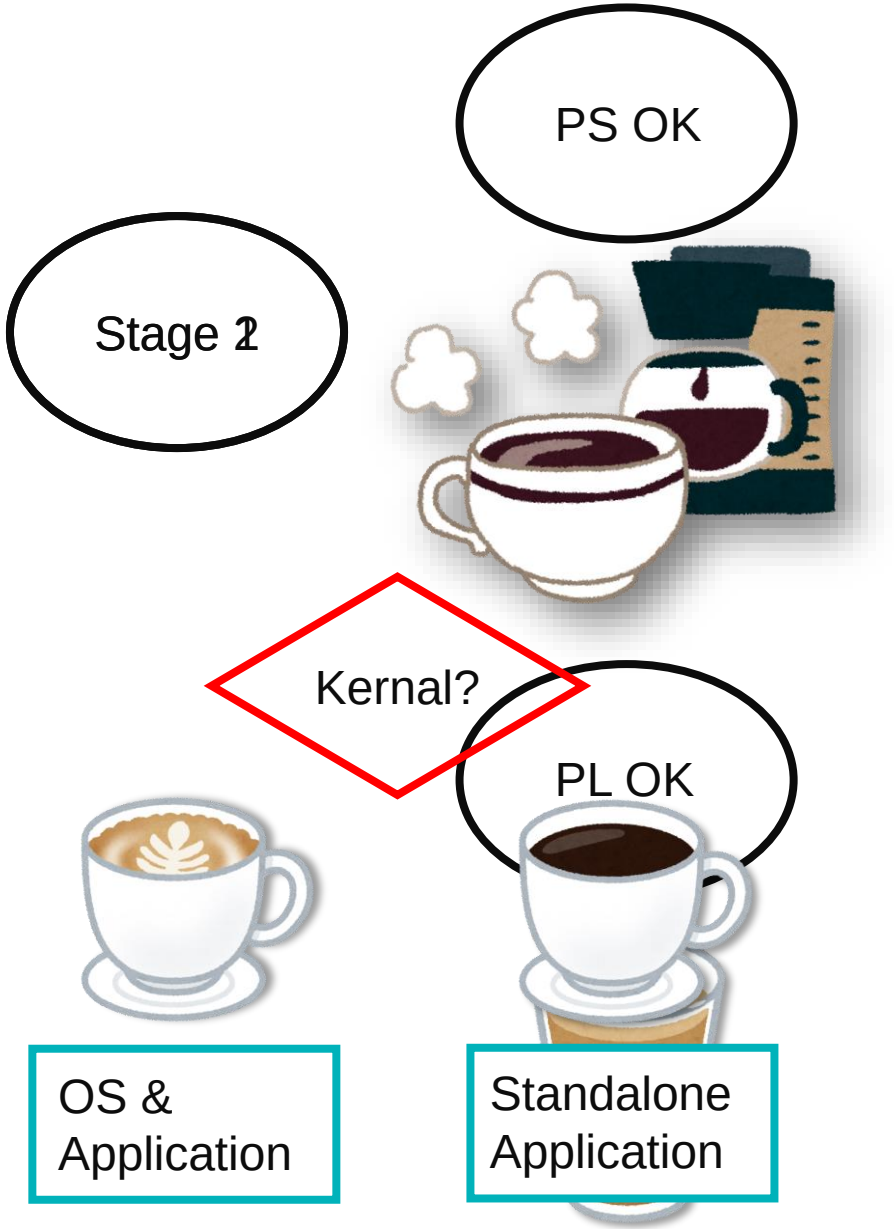


Zynq Boot and Configuration

FSBL

SSBL

Bitstream



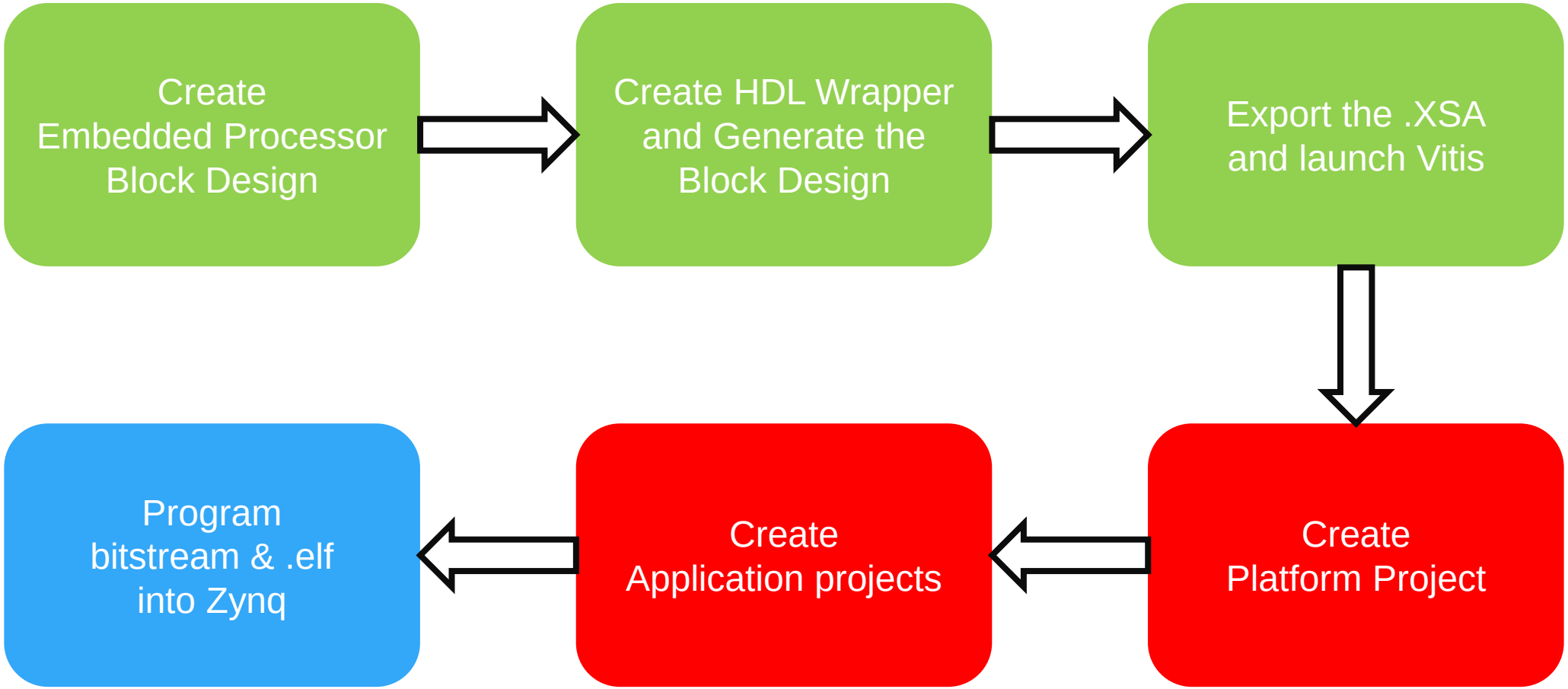
AMD
XILINX

AMD
XILINX

Embedded System Design Flow

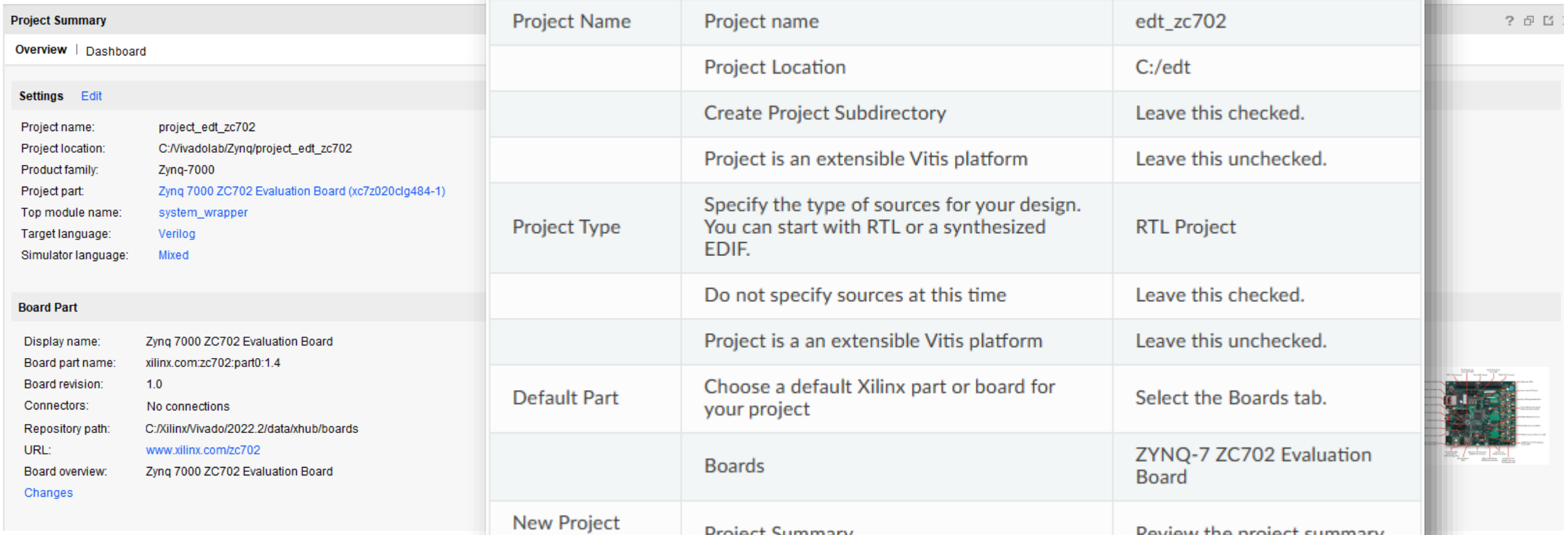
Course Agenda
2023

Embedded System Design Flow



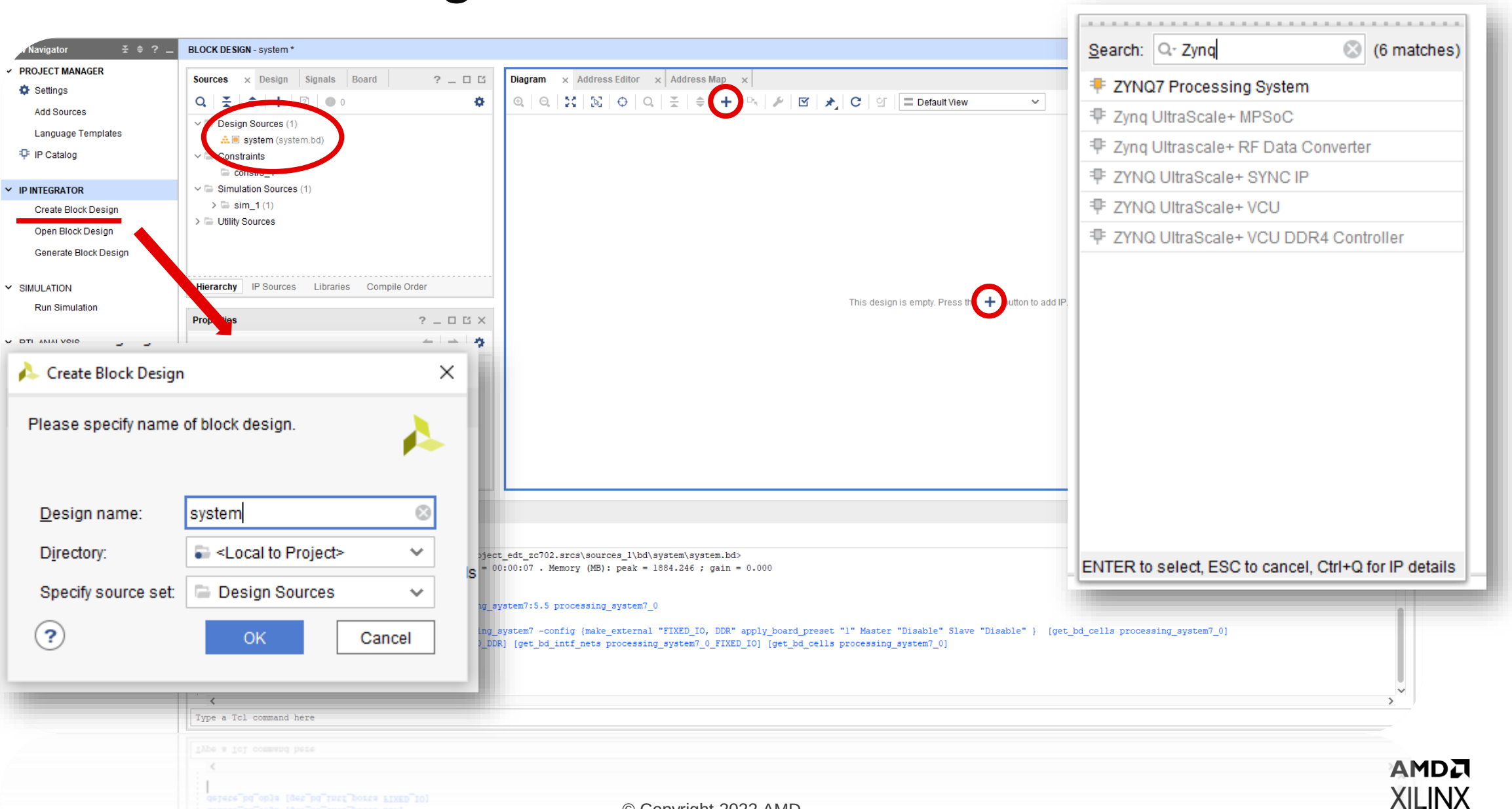
AMD
XILINX

Create Project



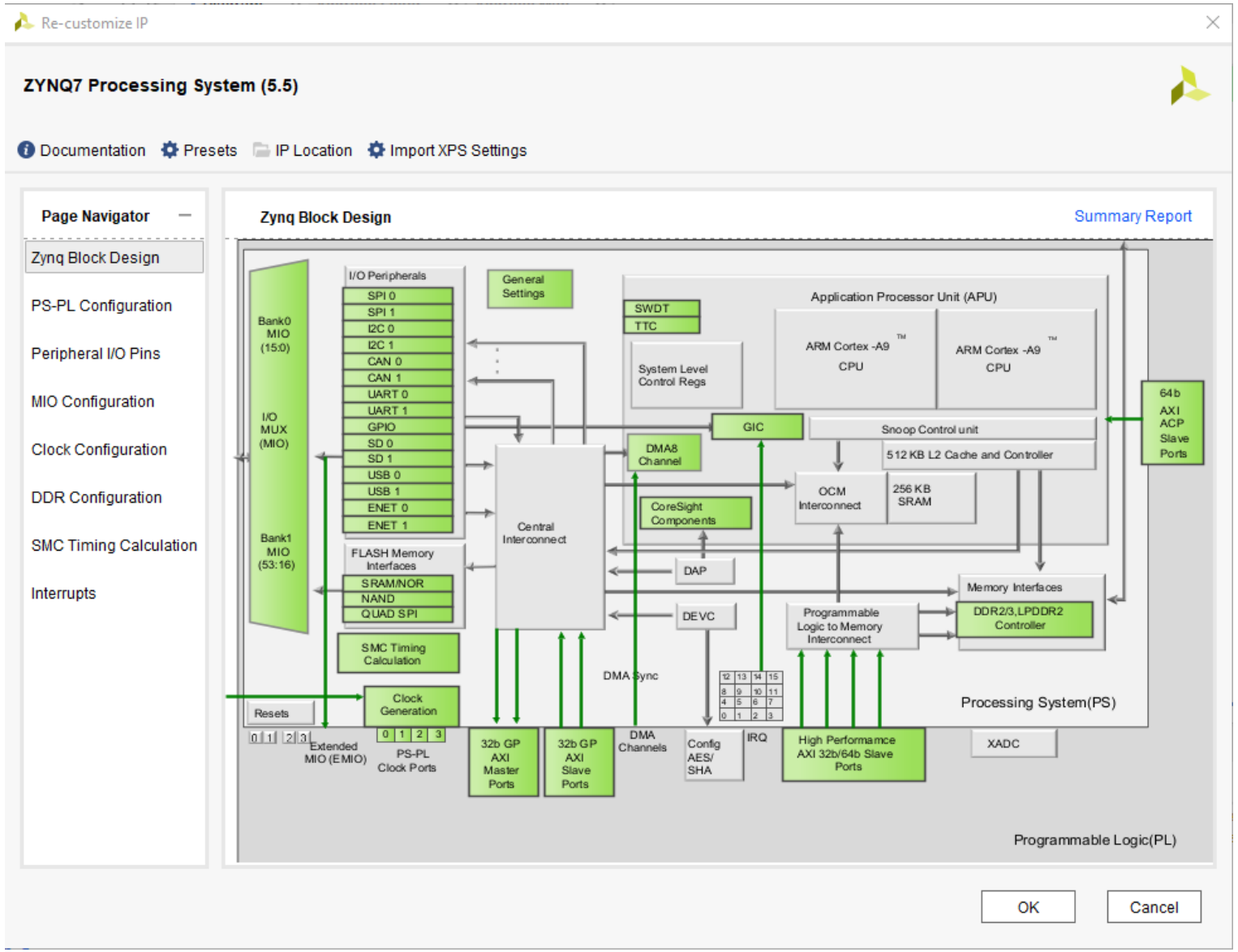
Screen	System Property	Setting or Command to Use
Project Name	Project name	edt_zc702
	Project Location	C:/edt
	Create Project Subdirectory	Leave this checked.
	Project is an extensible Vitis platform	Leave this unchecked.
Project Type	Specify the type of sources for your design. You can start with RTL 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 roject Summary	Project Summary	Review the project summary.

Create Block Design



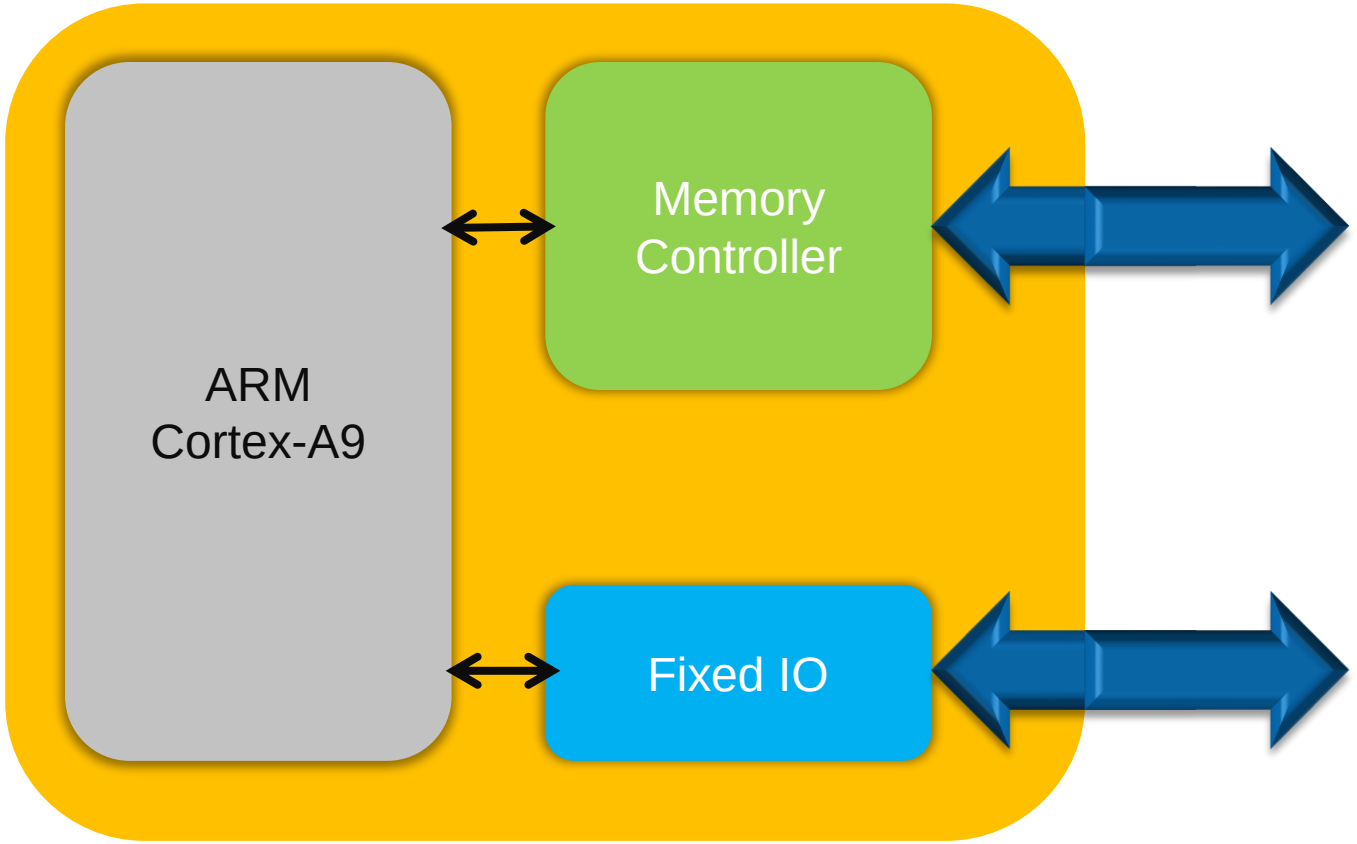
AMD XILINX

Zynq Block Design

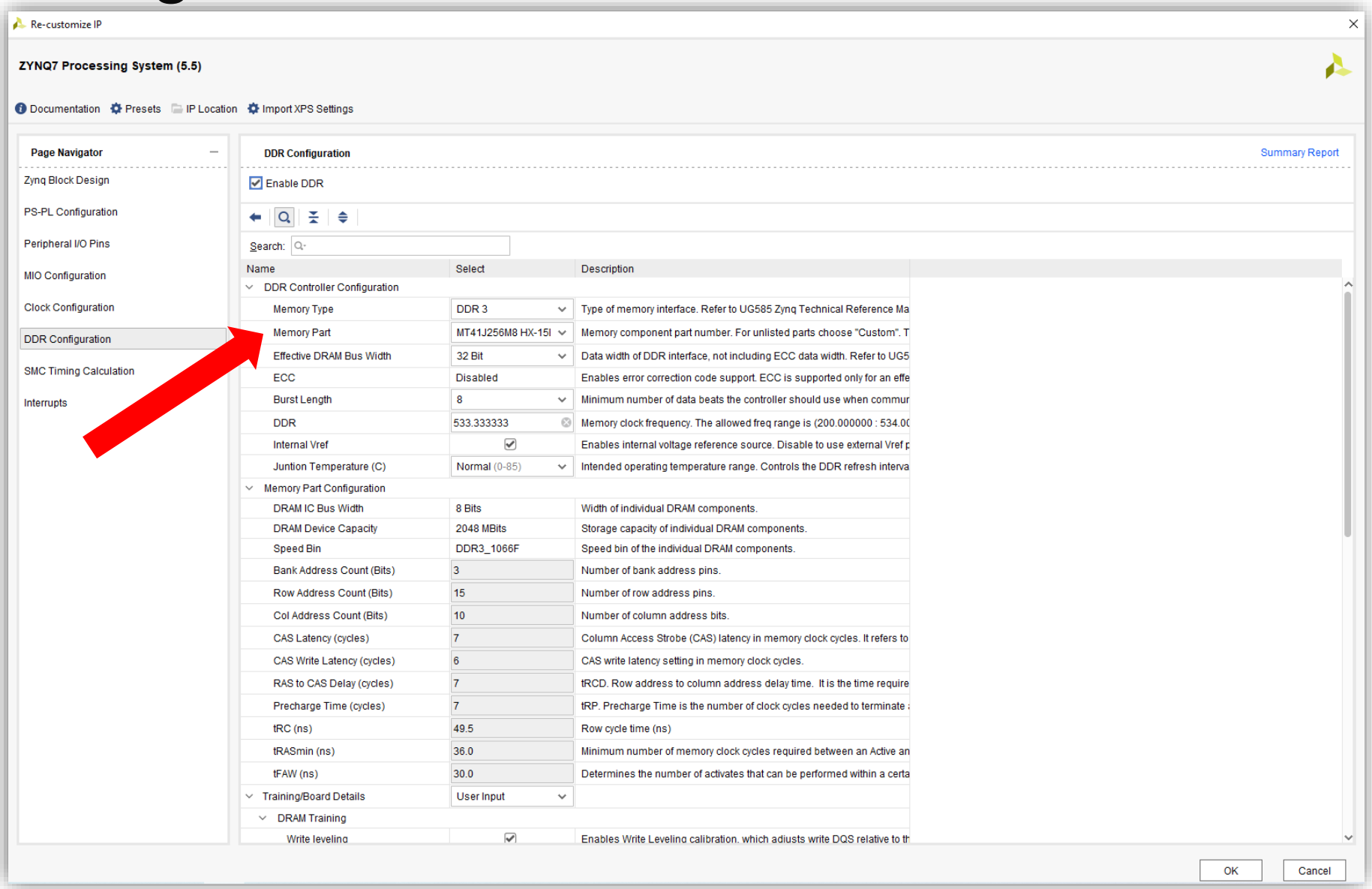


AMD XILINX

Basic Embedded System

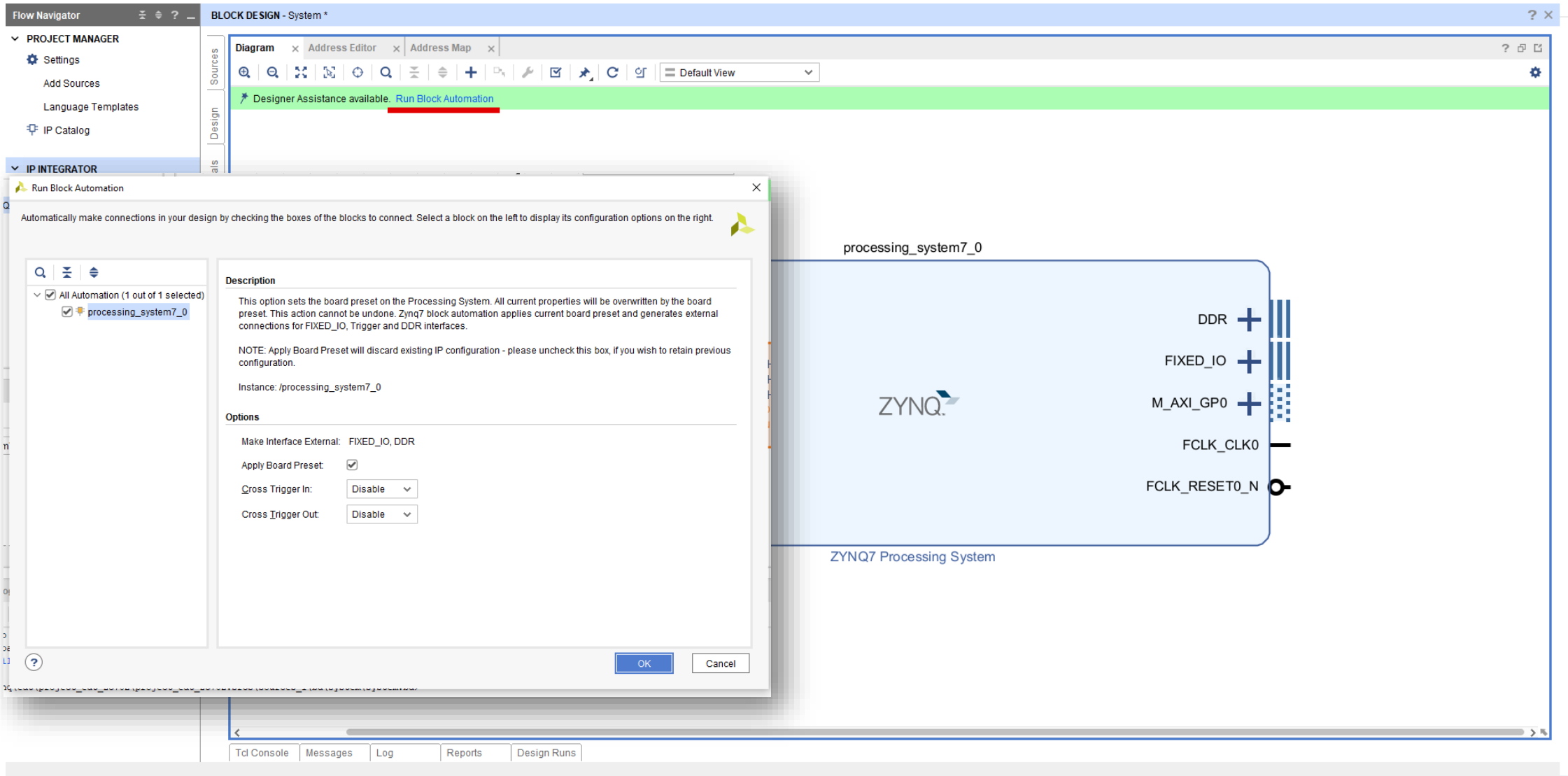


DDR Configuration

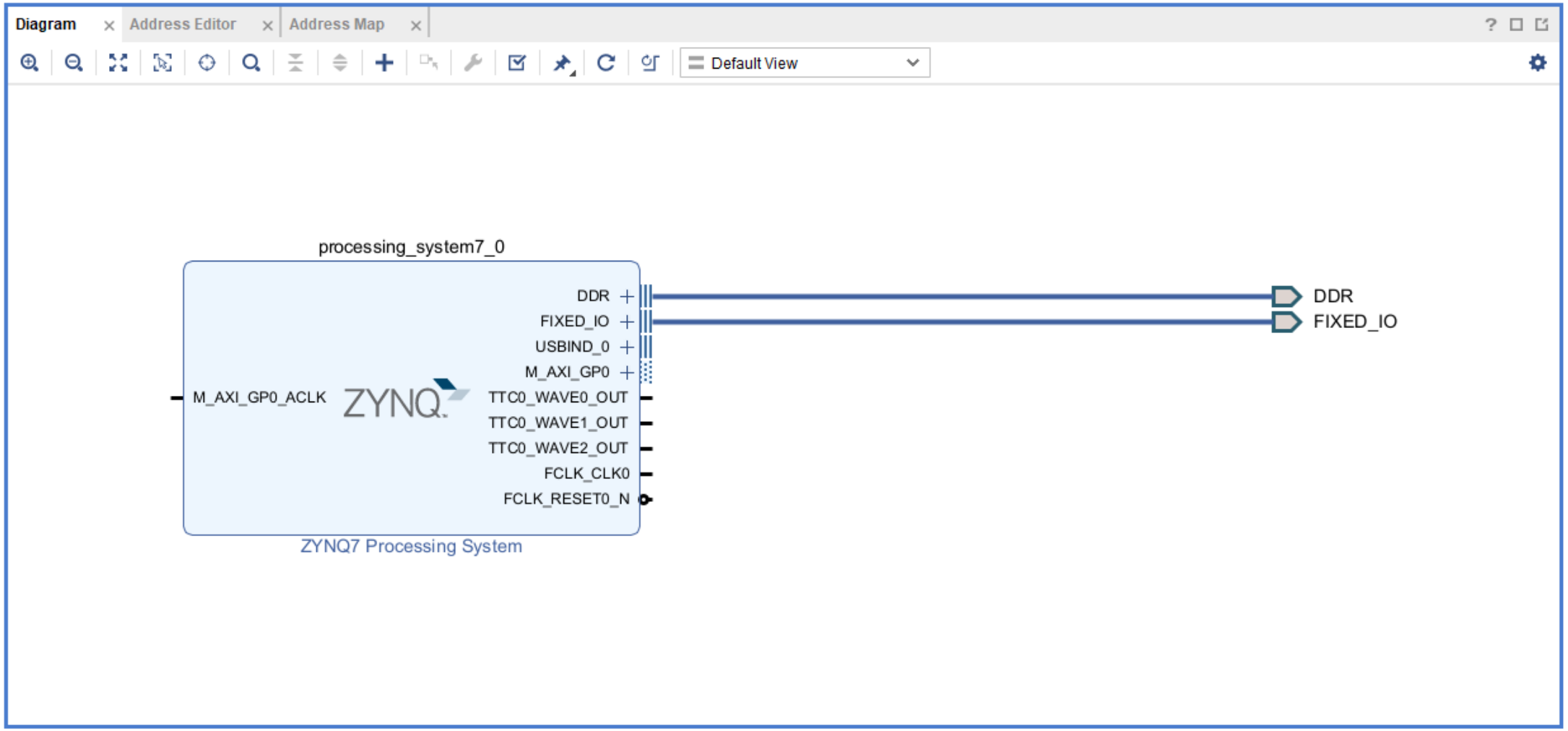


AMD
XILINX

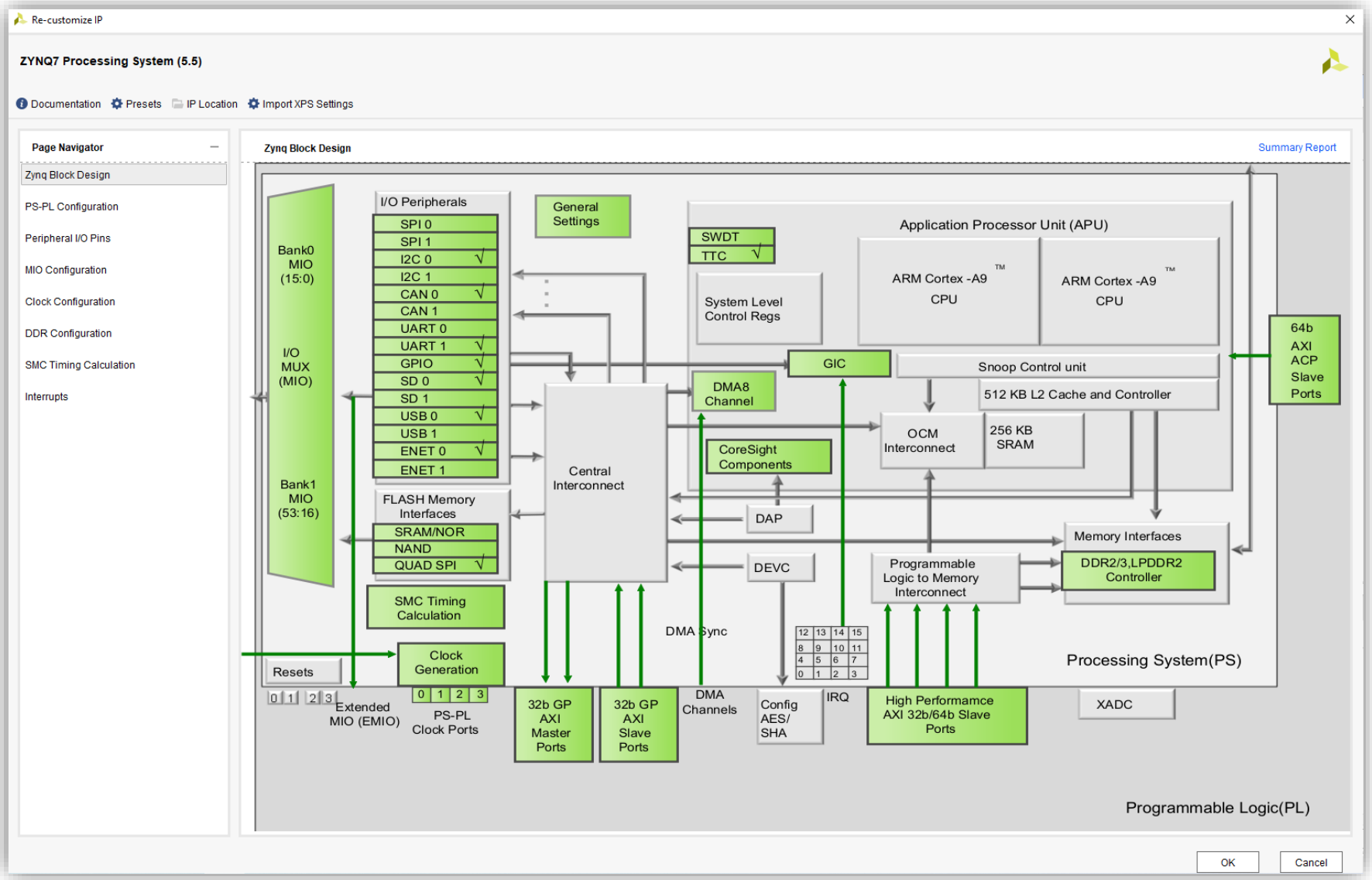
Run Block Automation



Run Block Automation

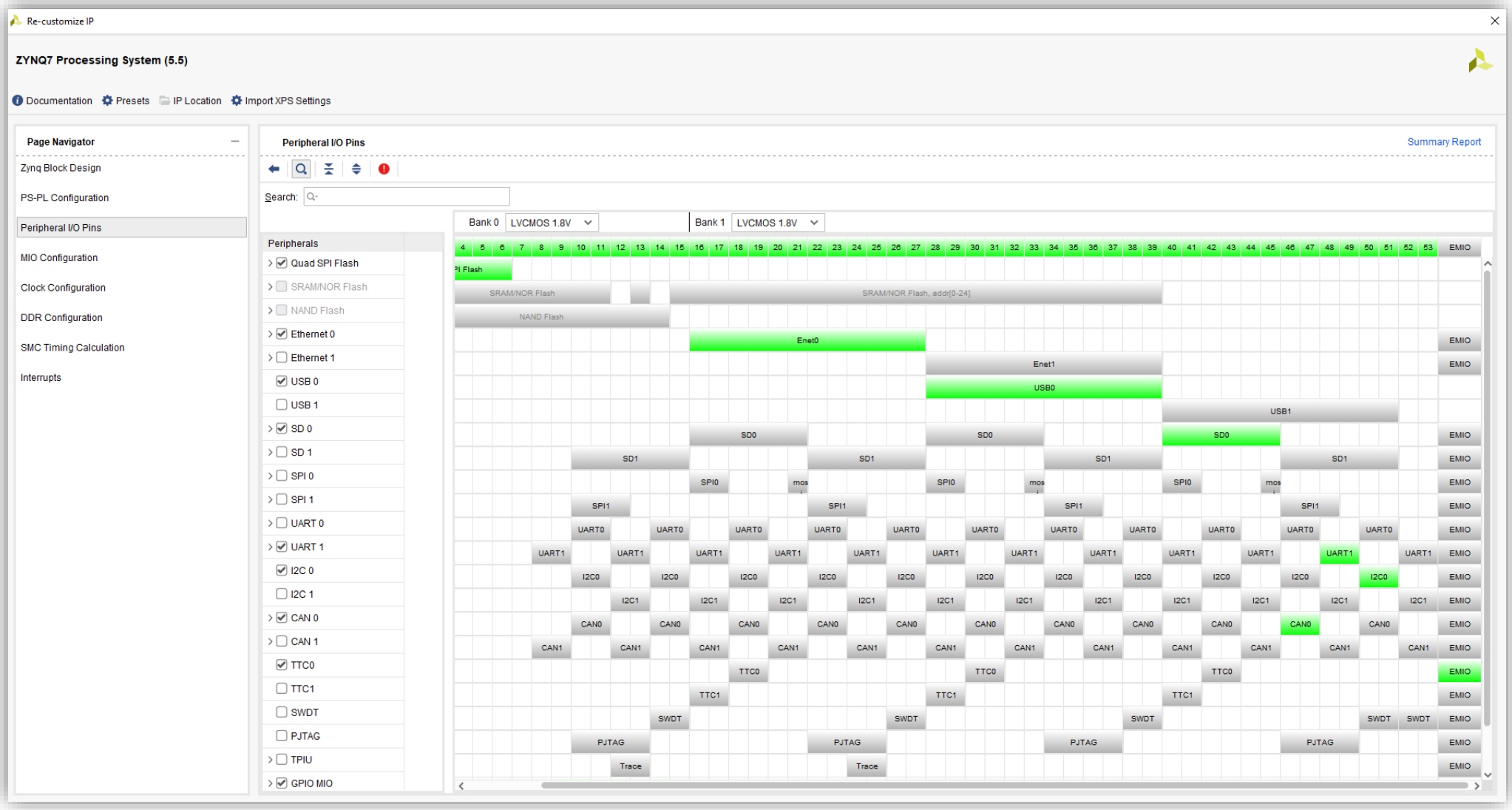


Zynq Block Design



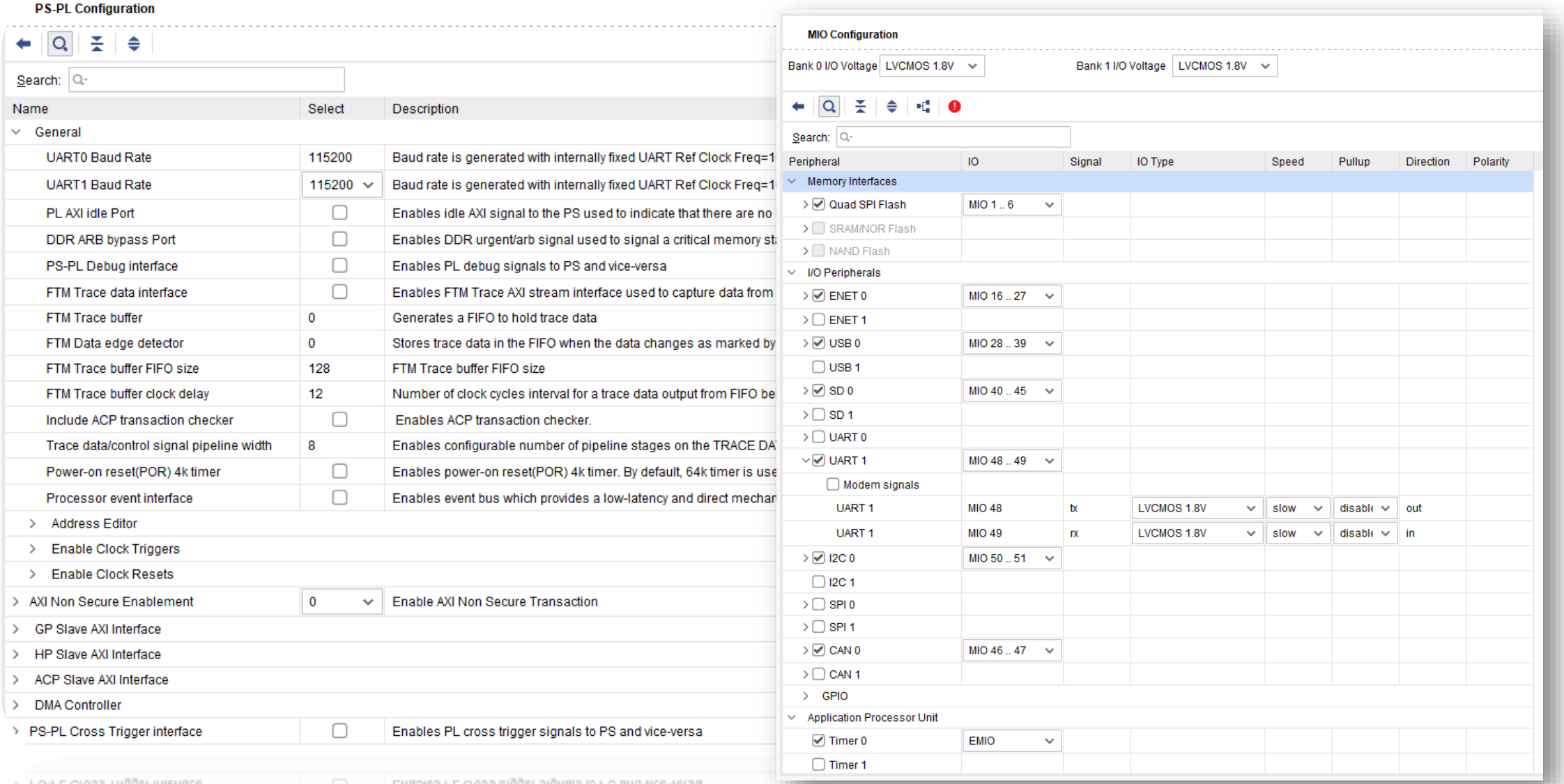
AMD XILINX

Peripheral I/O Pins



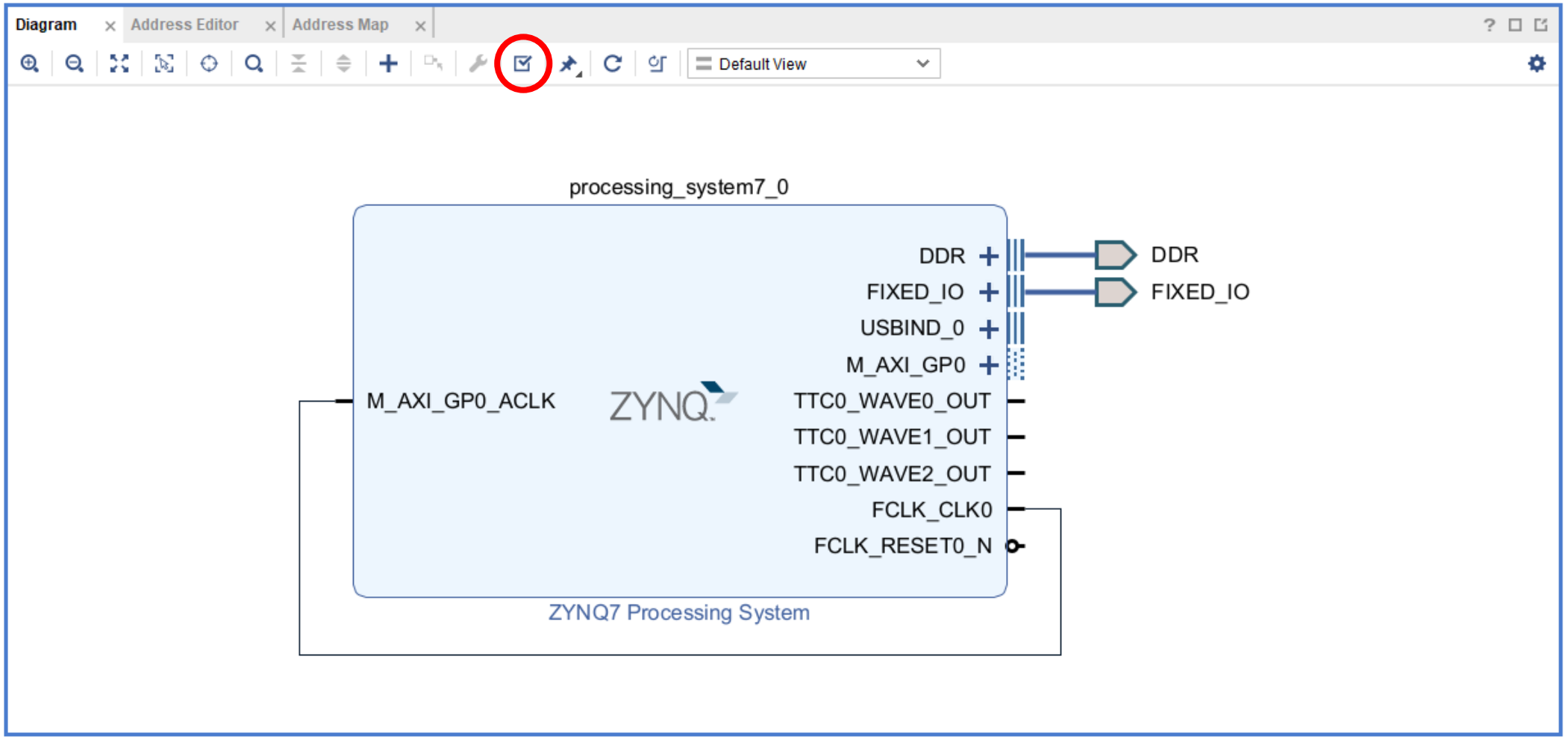
AMD XILINX

PS – PL Configuration & MIO Configuration



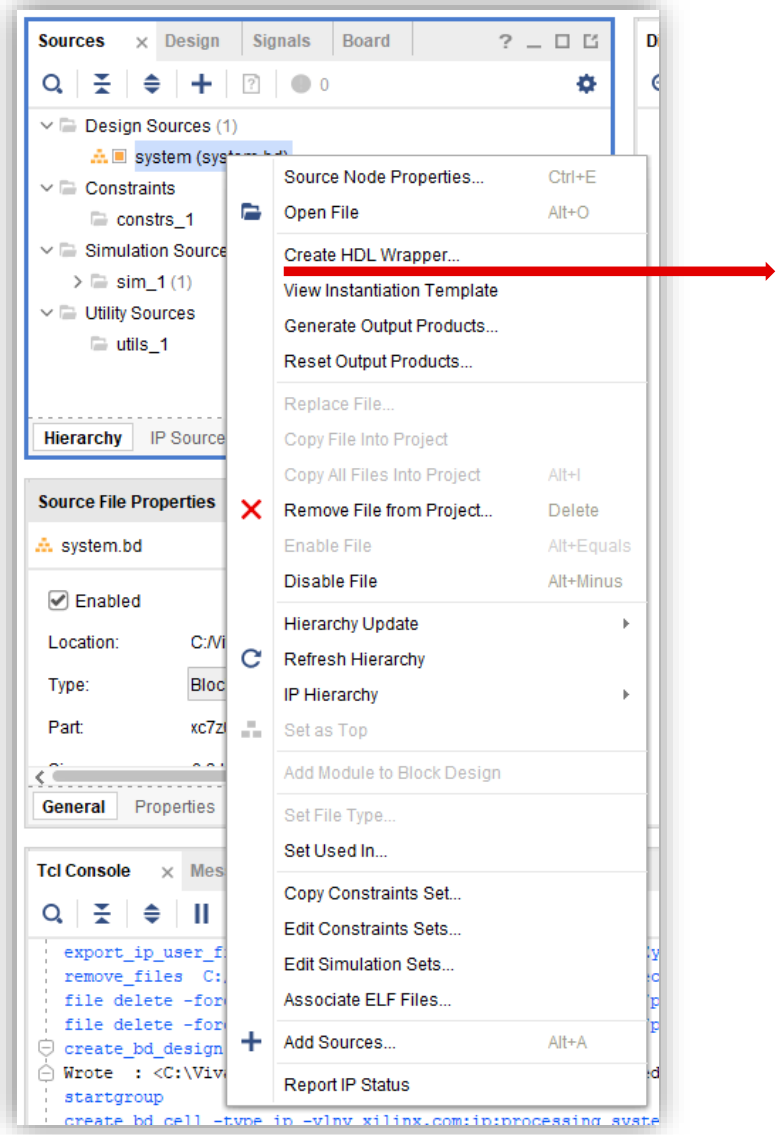
AMD XILINX

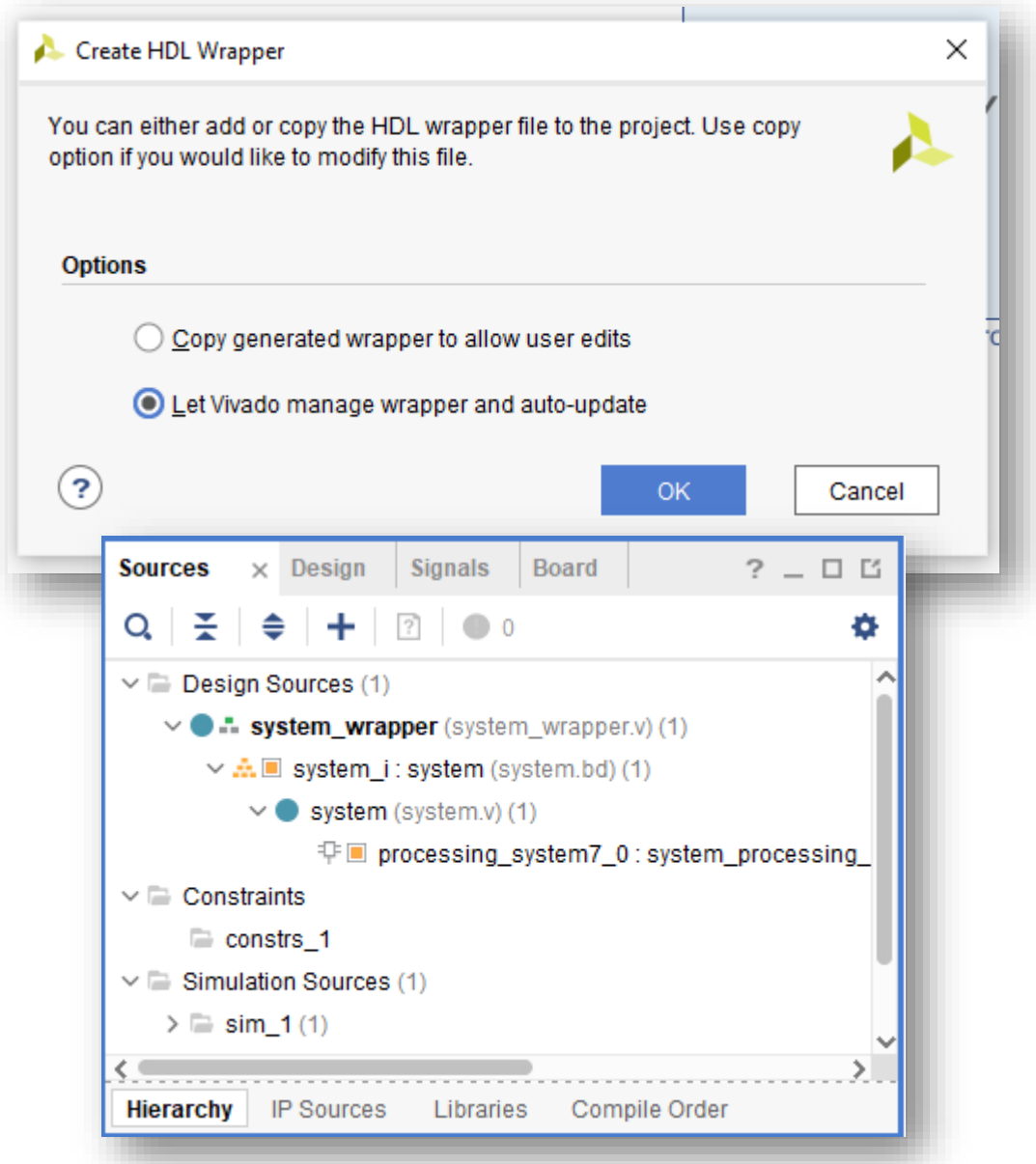
Run Block Automation



AMD XILINX

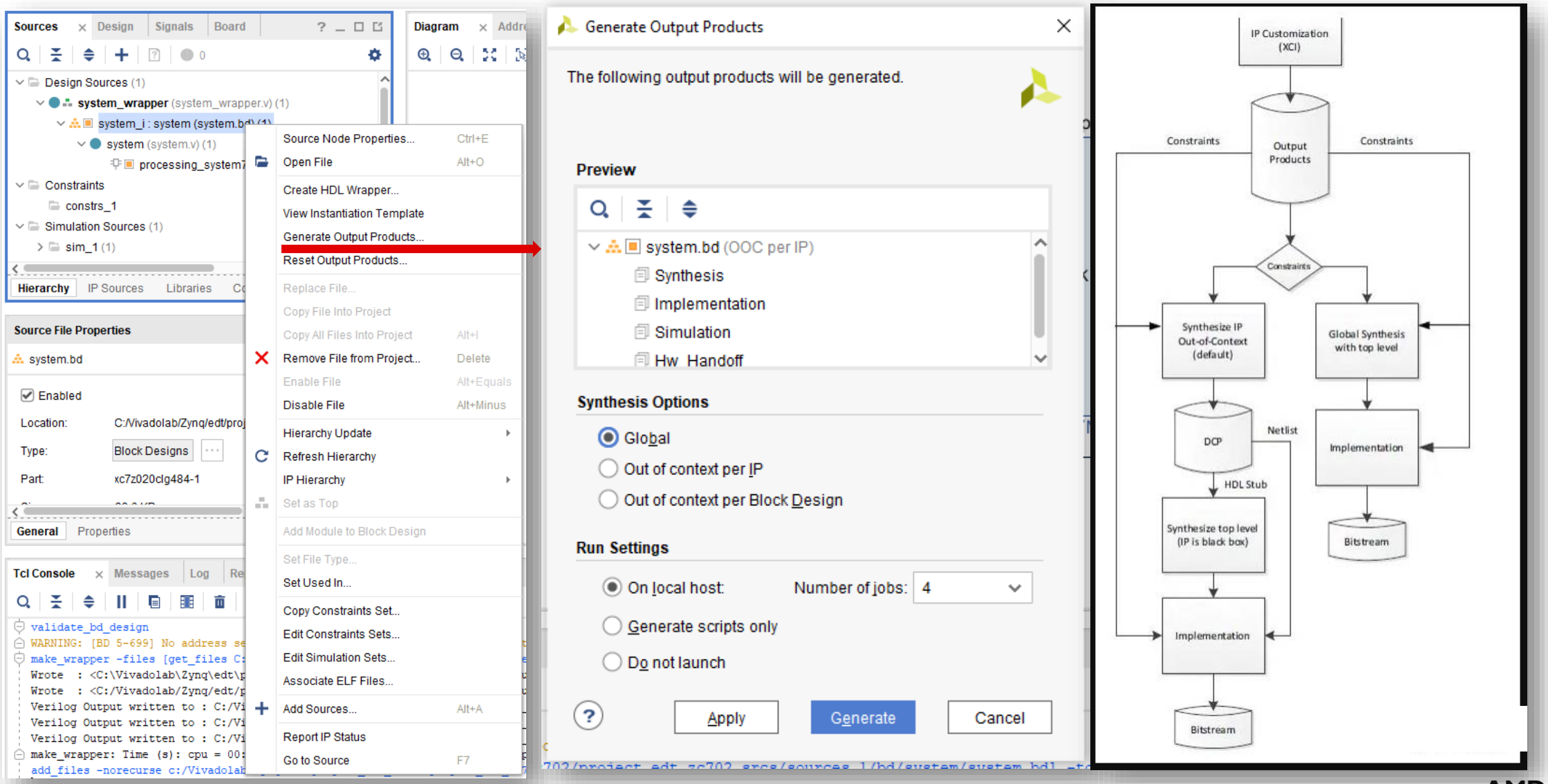
Create HDL Wrapper





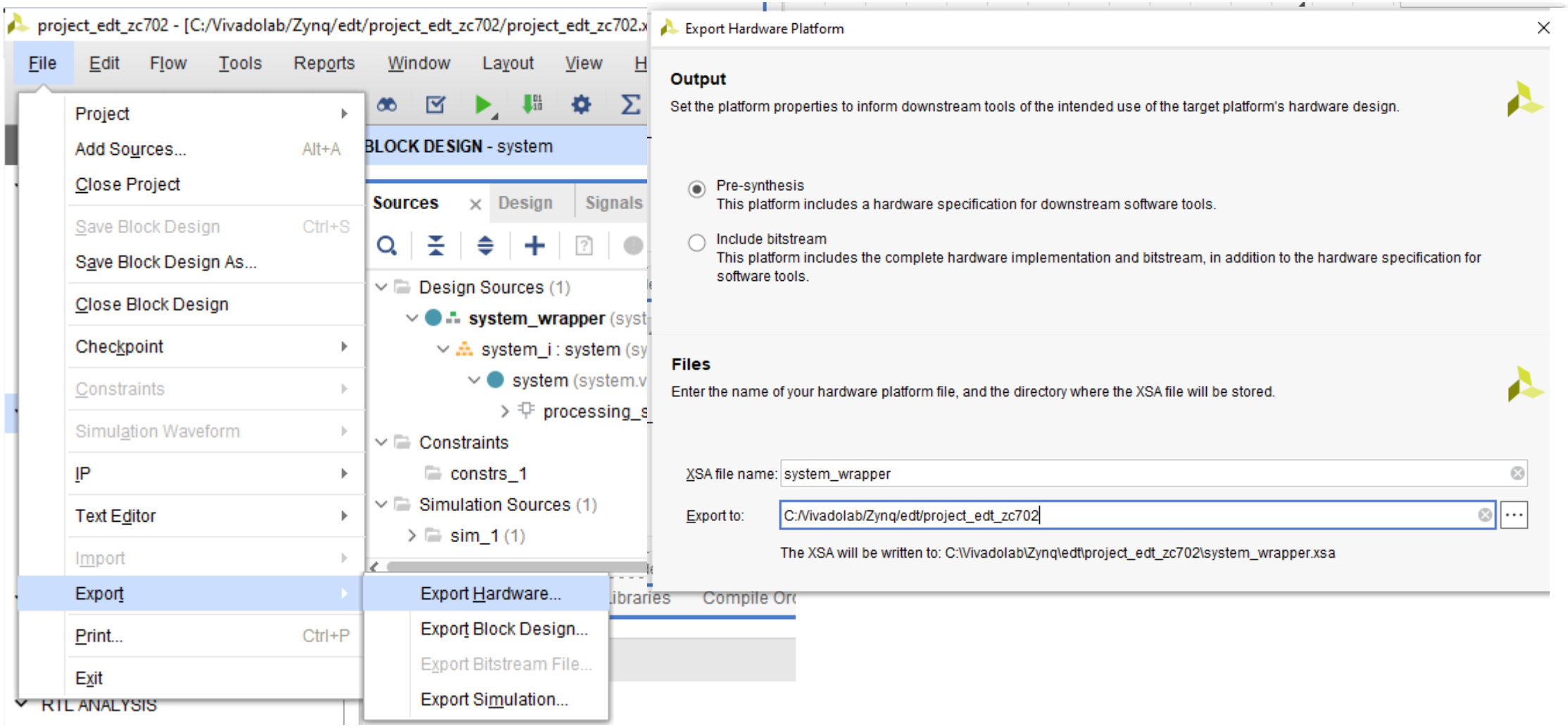
AMD XILINX

Generate Output Products

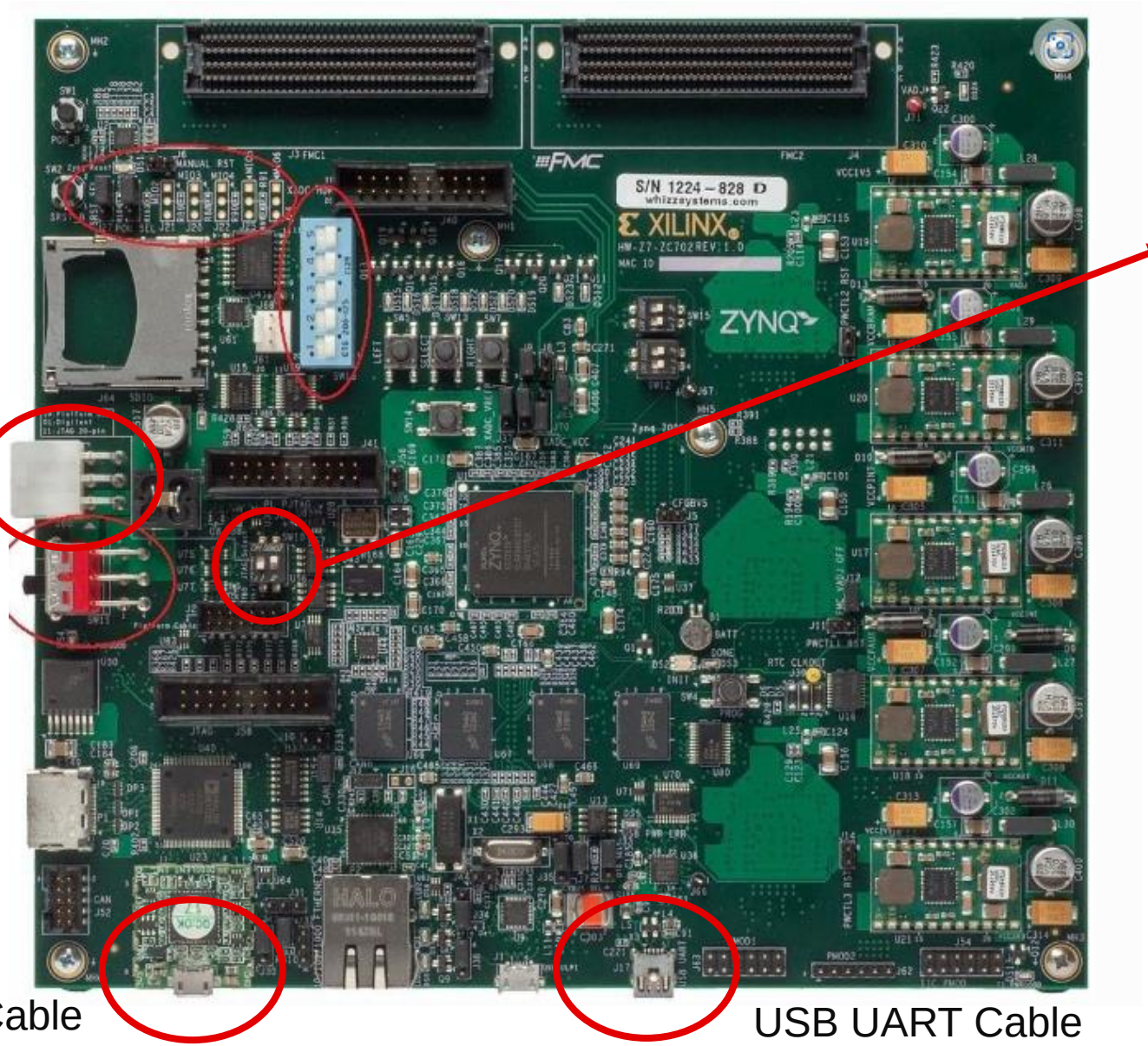


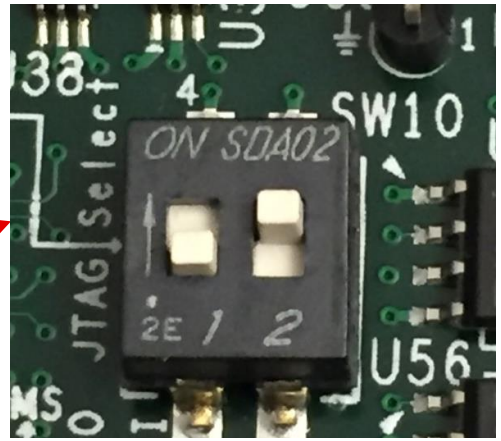
AMD XILINX

Exporting Hardware



Setting Up the Board

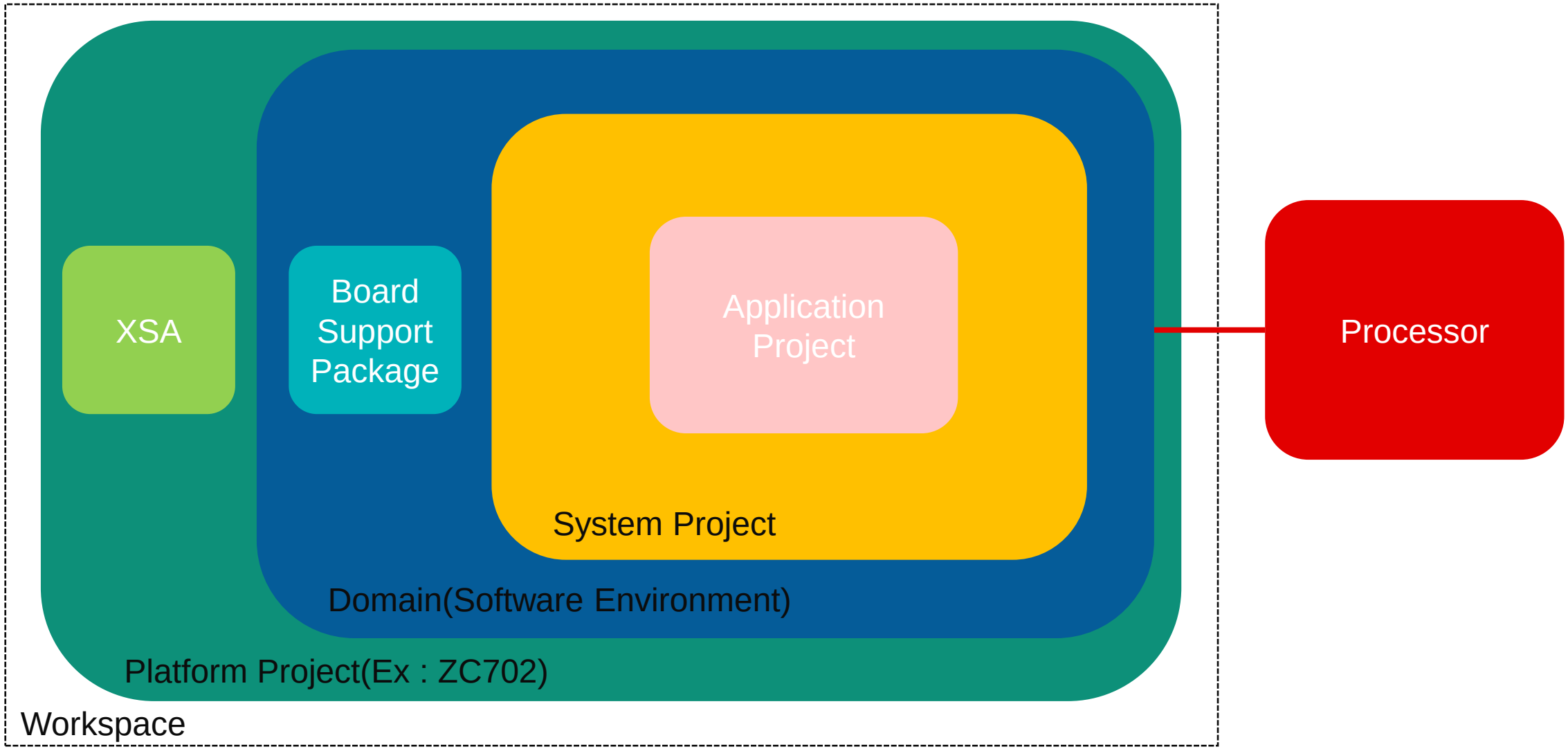




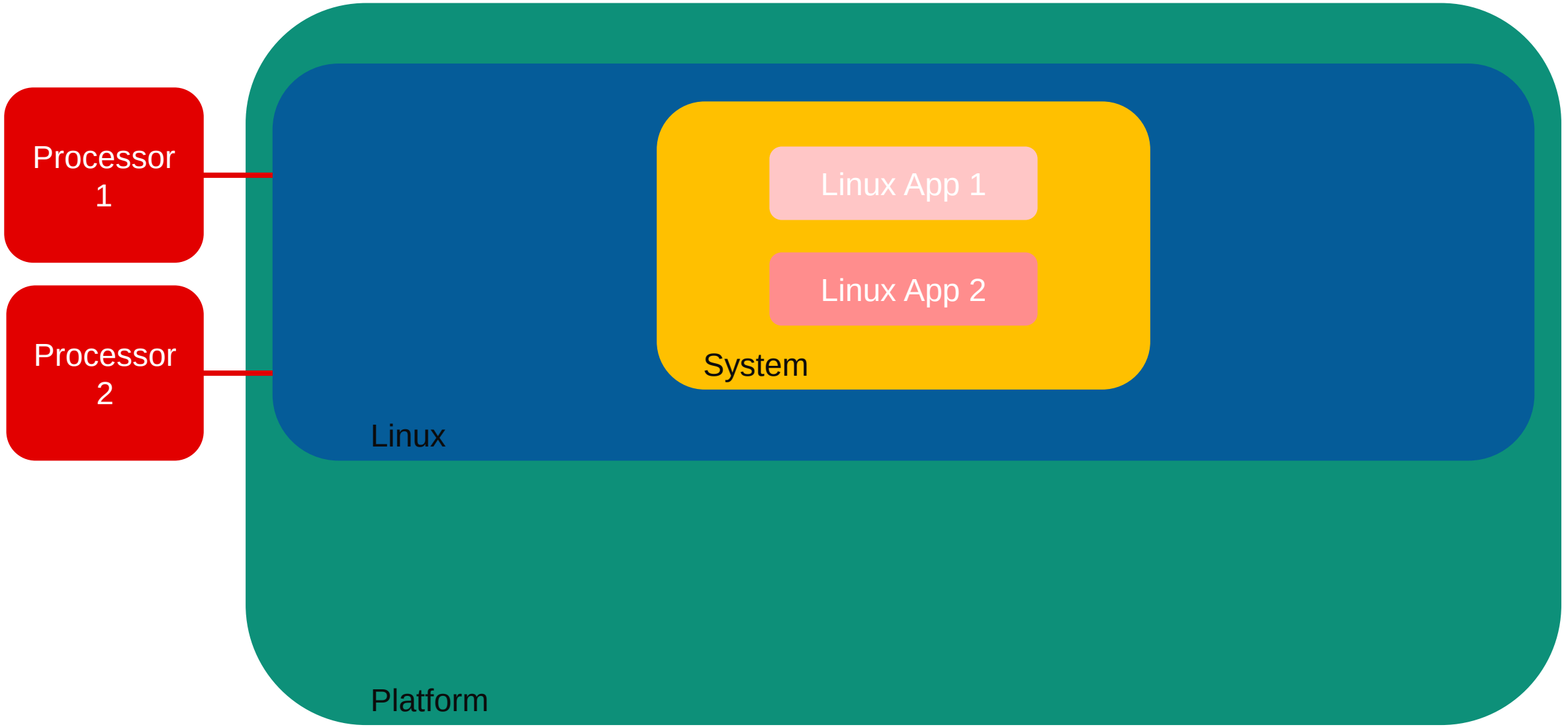
JTAG Selection

AMD XILINX

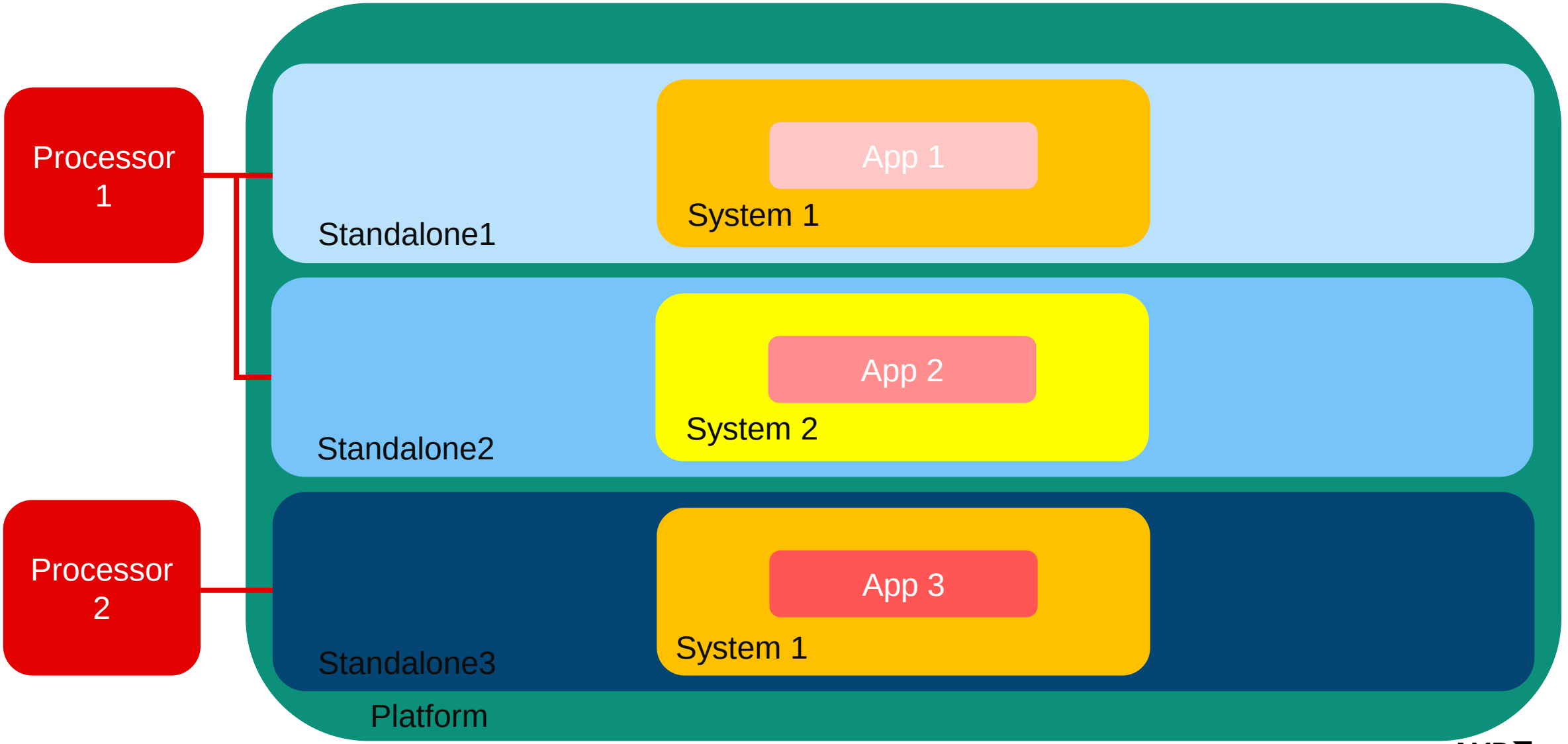
Vitis Workspace Structure



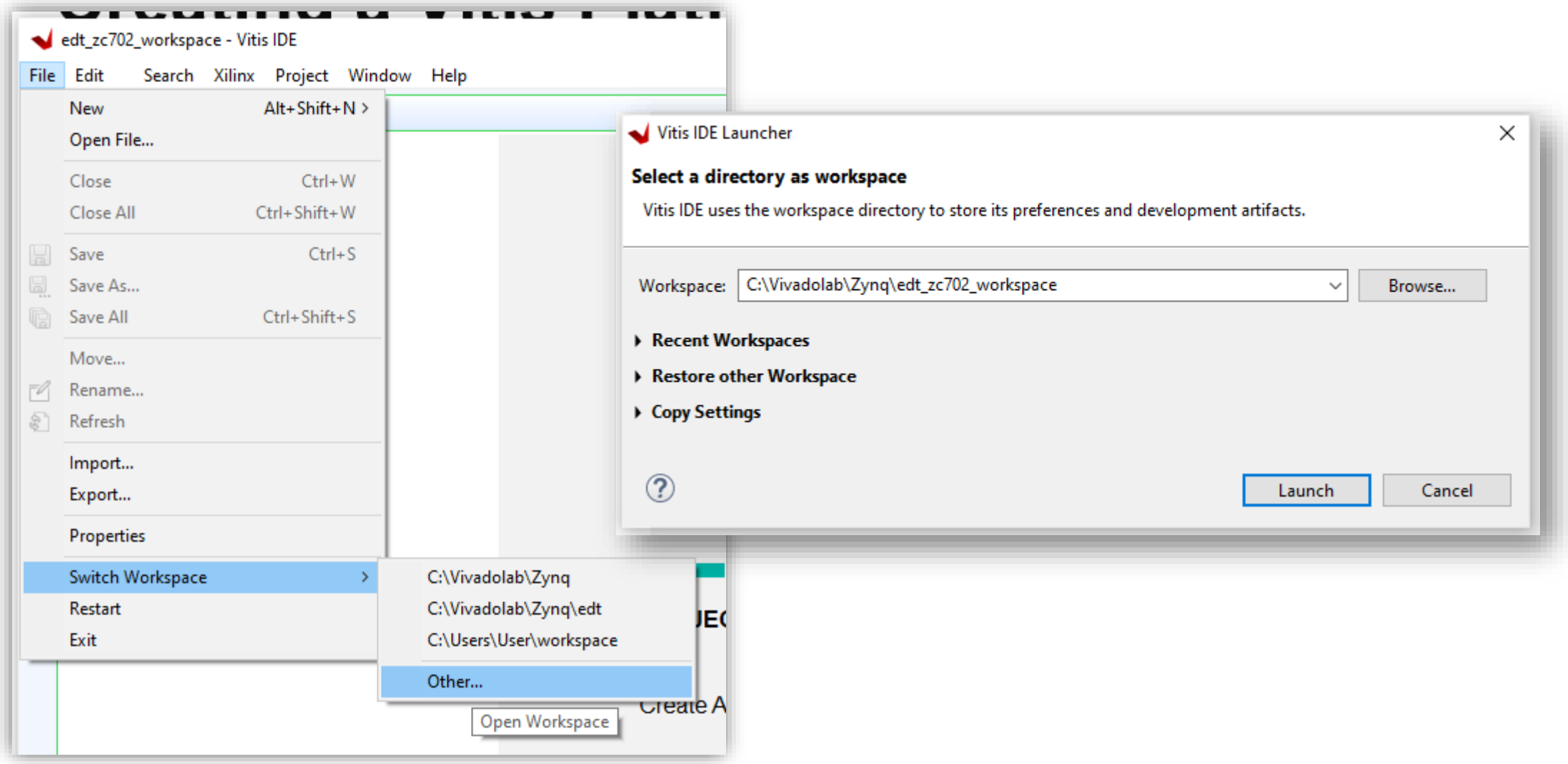
Vitis Workspace Structure



Vitis Workspace Structure

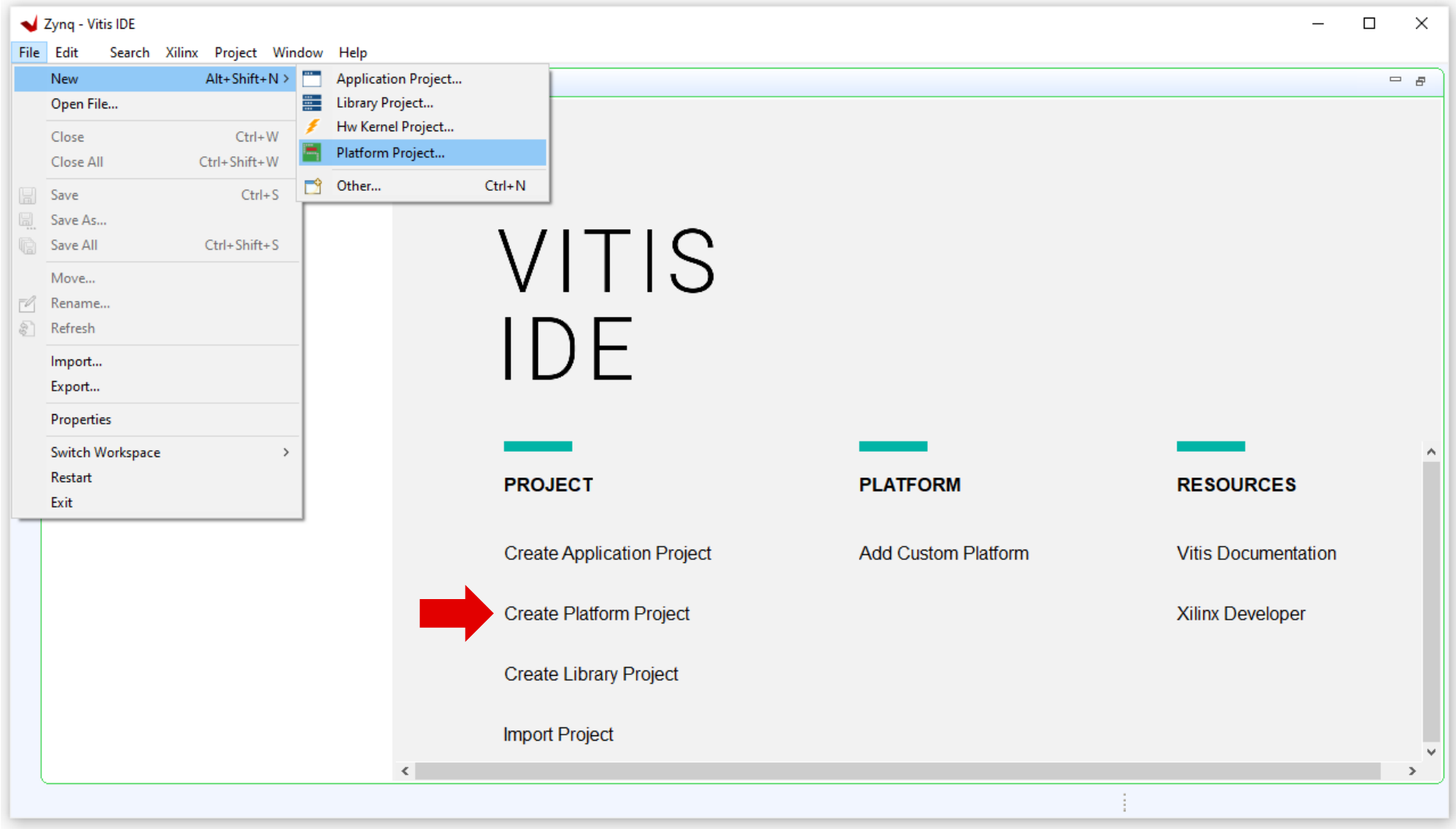


Creating a Vitis Platform Project



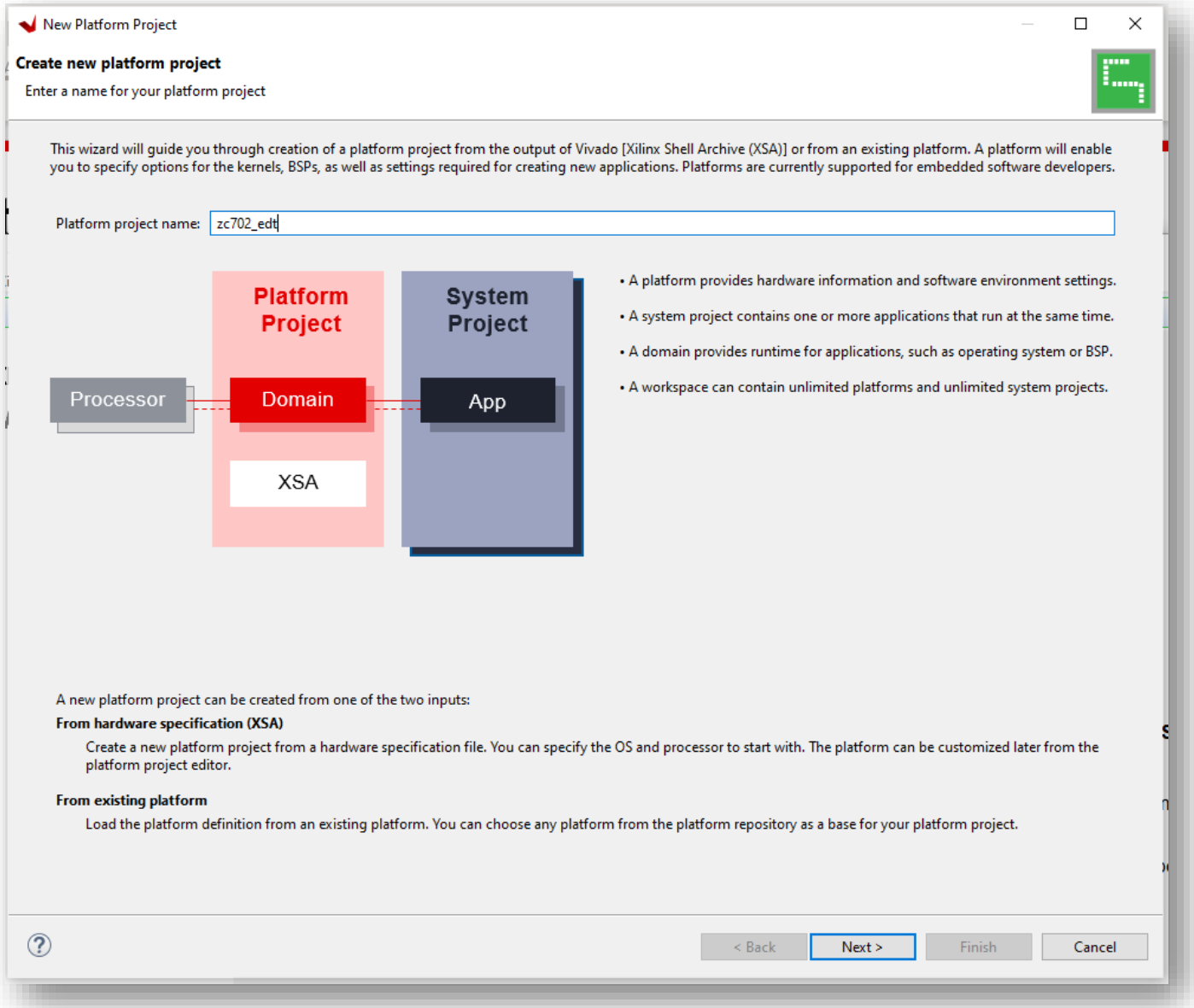
AMD
XILINX

Creating a Vitis Platform Project



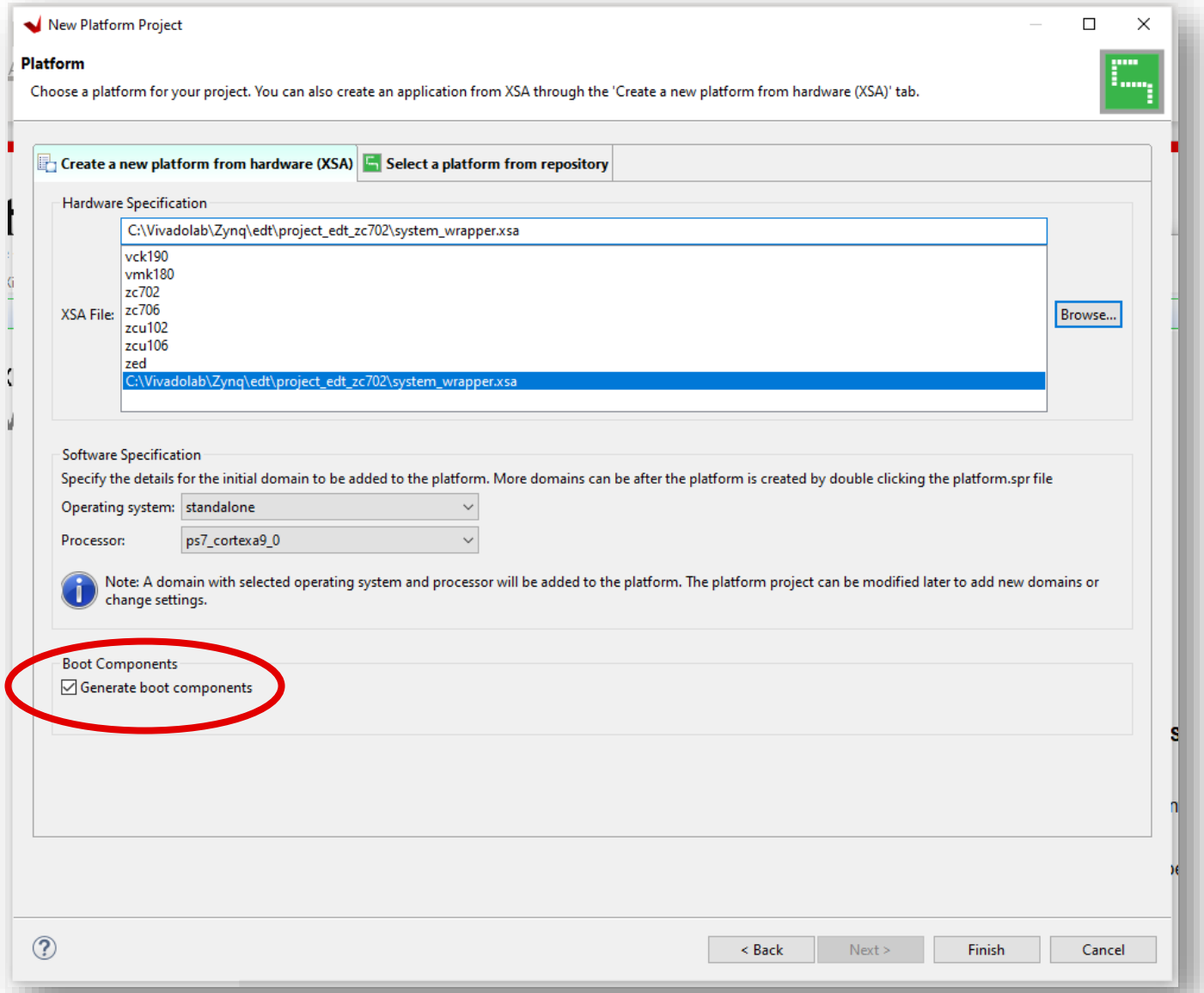
AMD XILINX

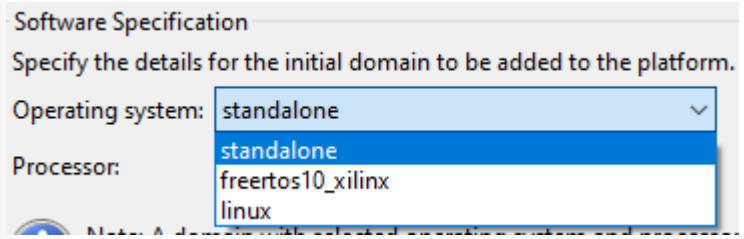
Creating a Vitis Platform Project

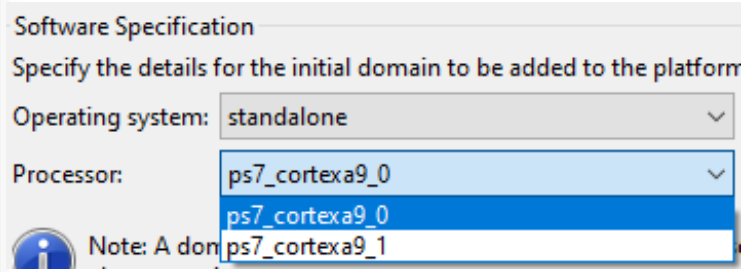


AMD XILINX

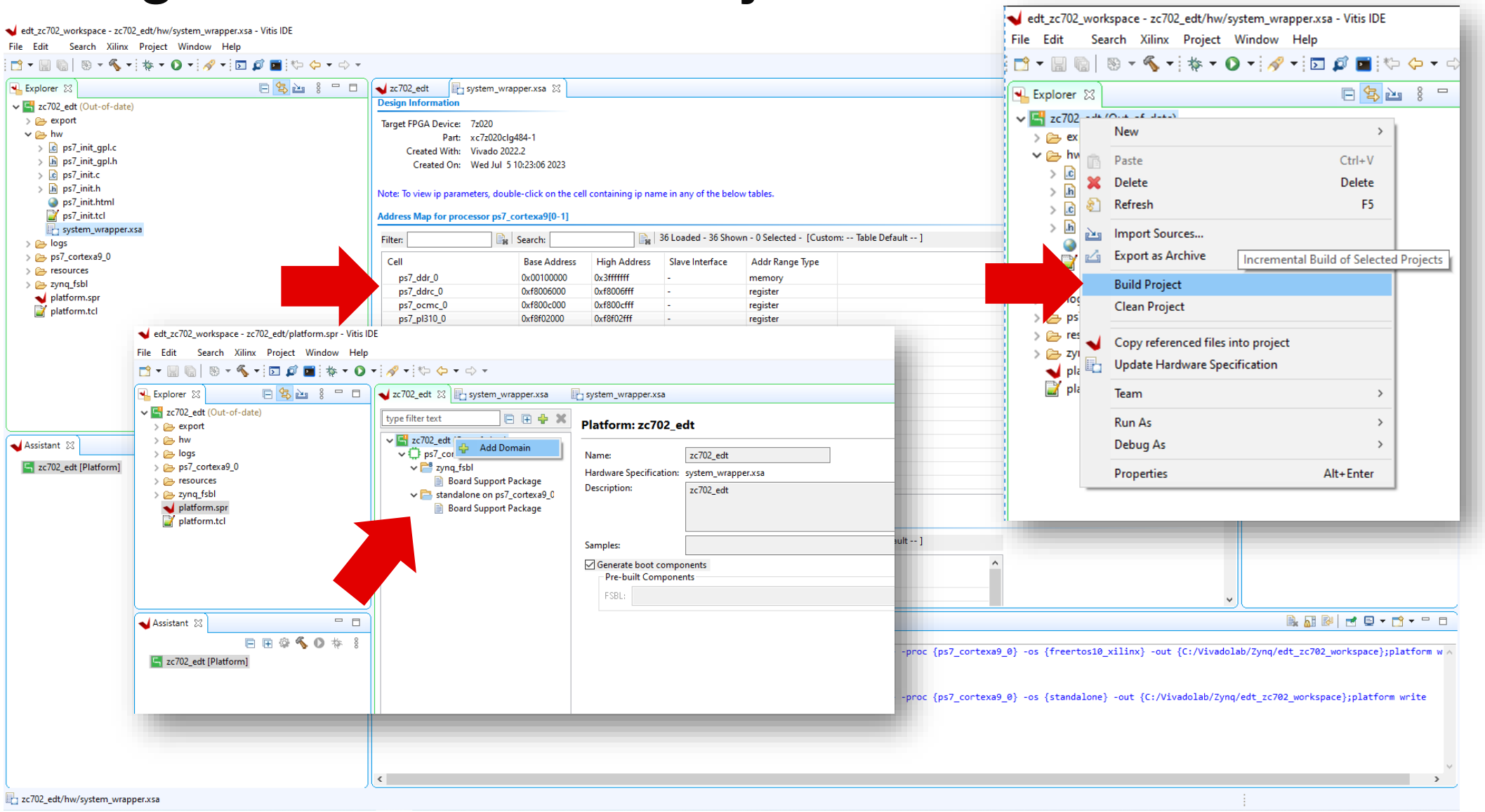
Creating a Vitis Platform Project





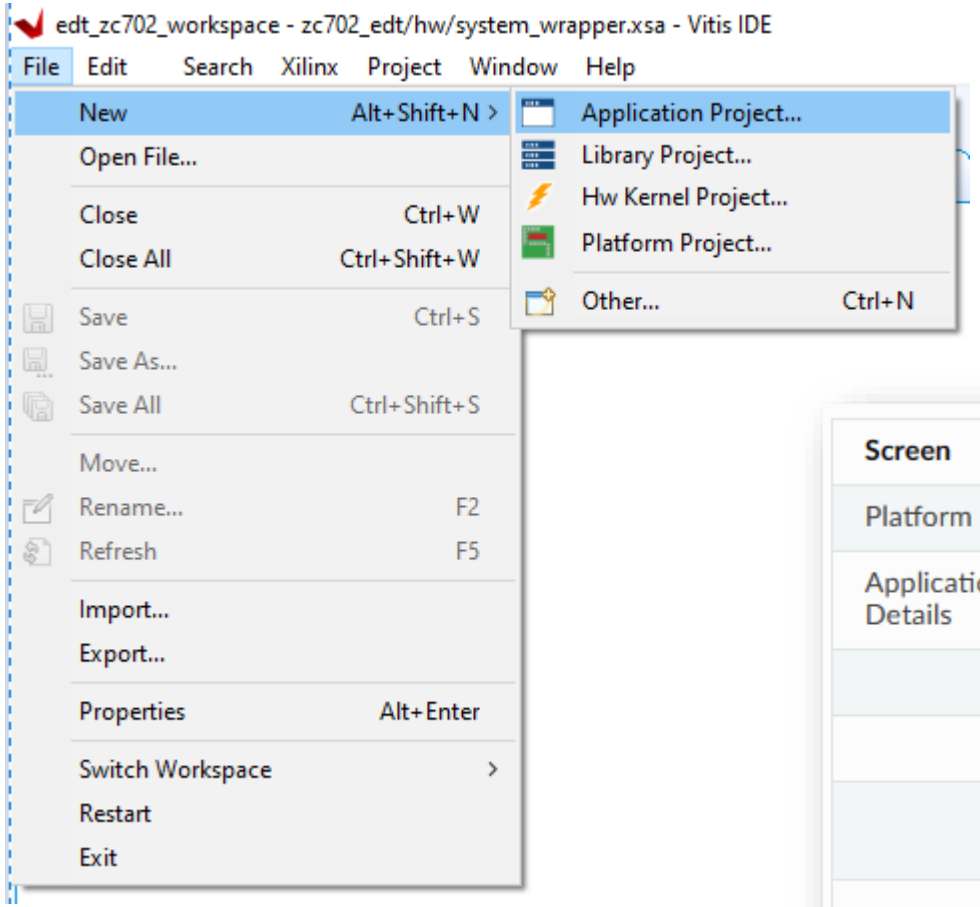


Creating a Vitis Platform Project



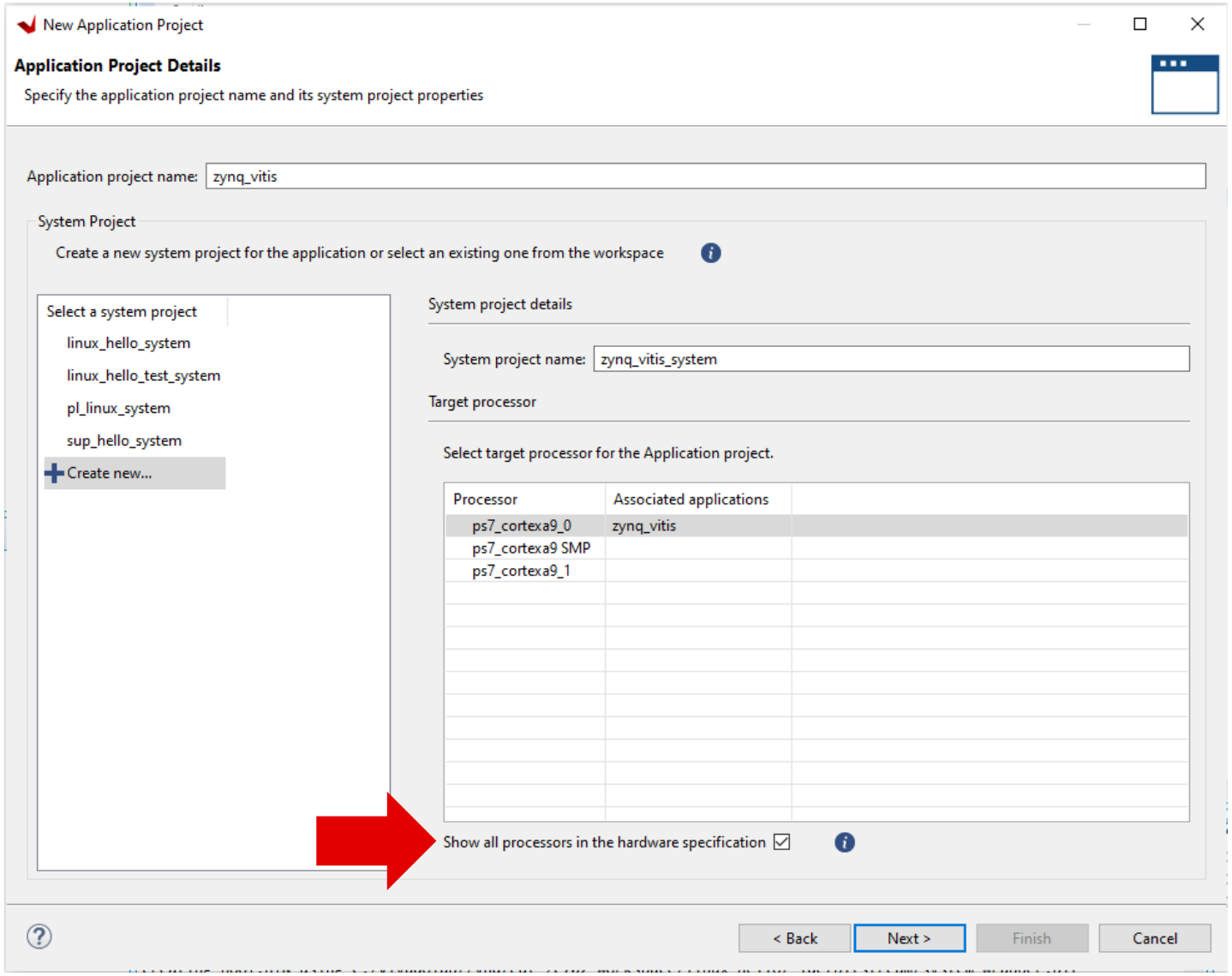
AMD XILINX

Creating the Hello World Application



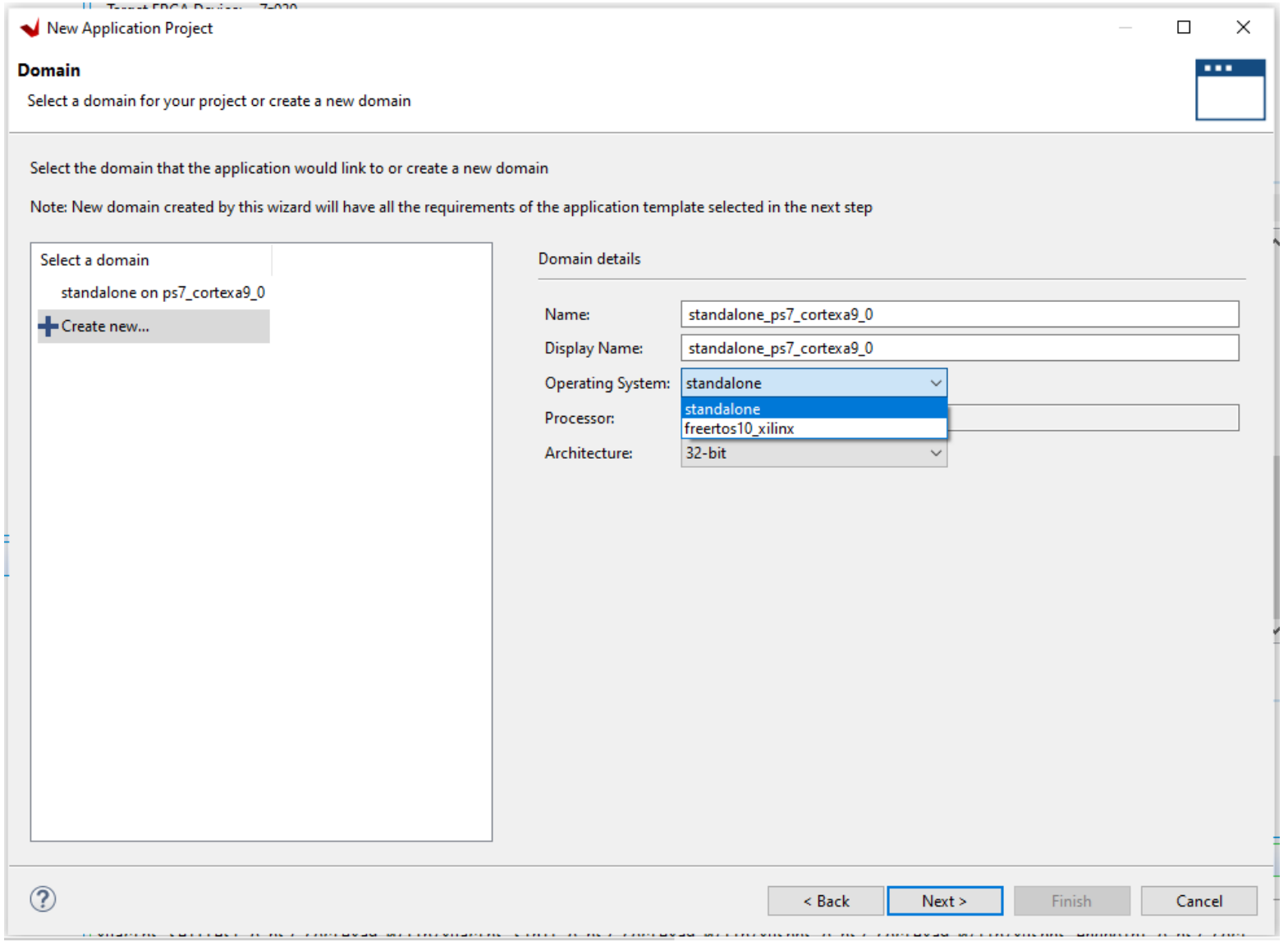
Screen	System Properties	Setting or Command to Use
Platform	Select a platform from repository	Click zc702_edt [custom].
Application Project Details	Application project name	Enter hello_world.
	System project name	Keep hello_world_system.
	Target Processor	Keep ps7_cortexa9_0 selected.
	Show all processors in hardware specification	Keep unchecked.
Domain	Select a domain	Keep standalone on ps7_cortex9_0 selected.
Templates	Available Templates	Hello World.

Creating the Hello World Application

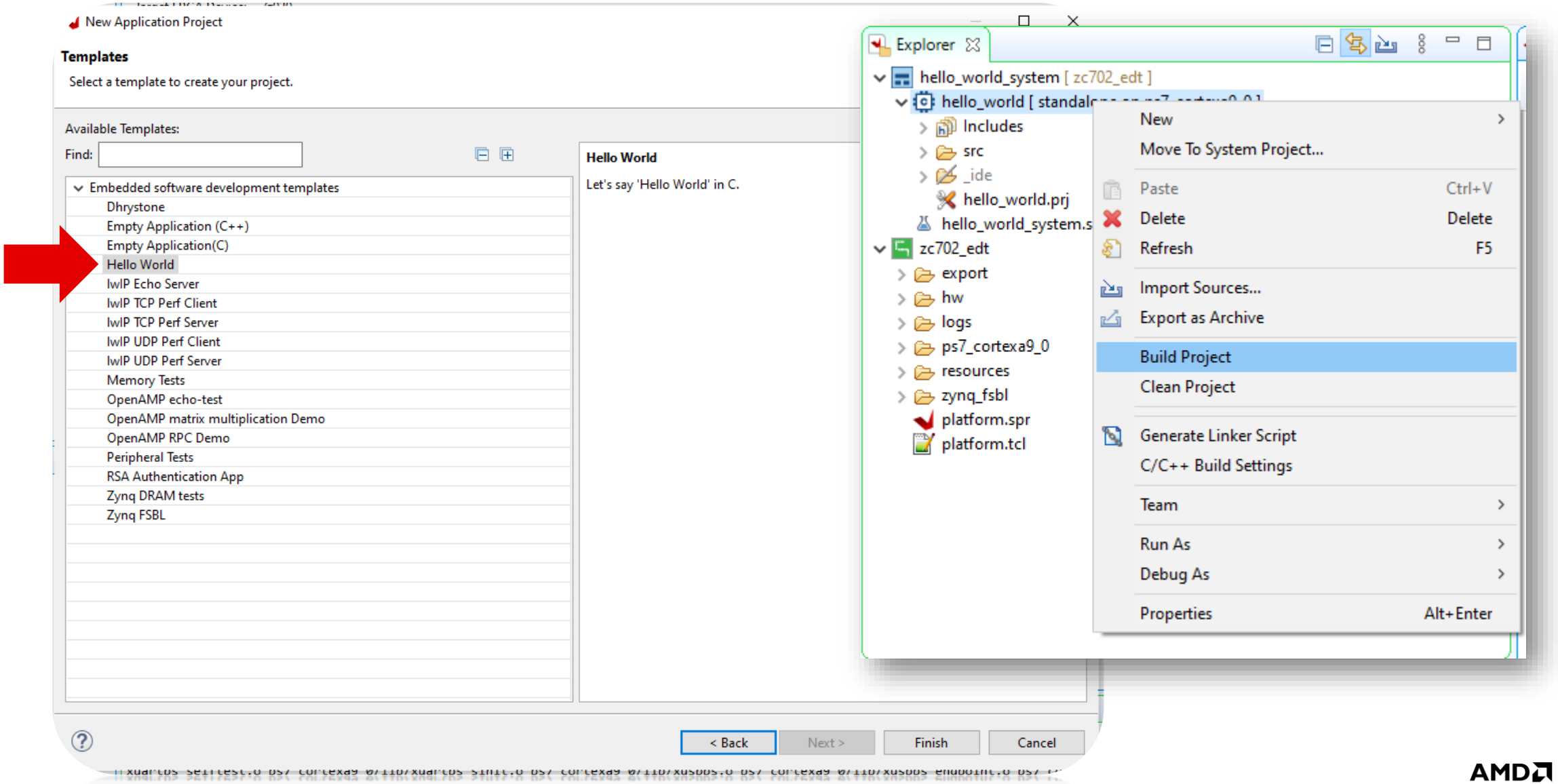


AMD XILINX

Creating the Hello World Application

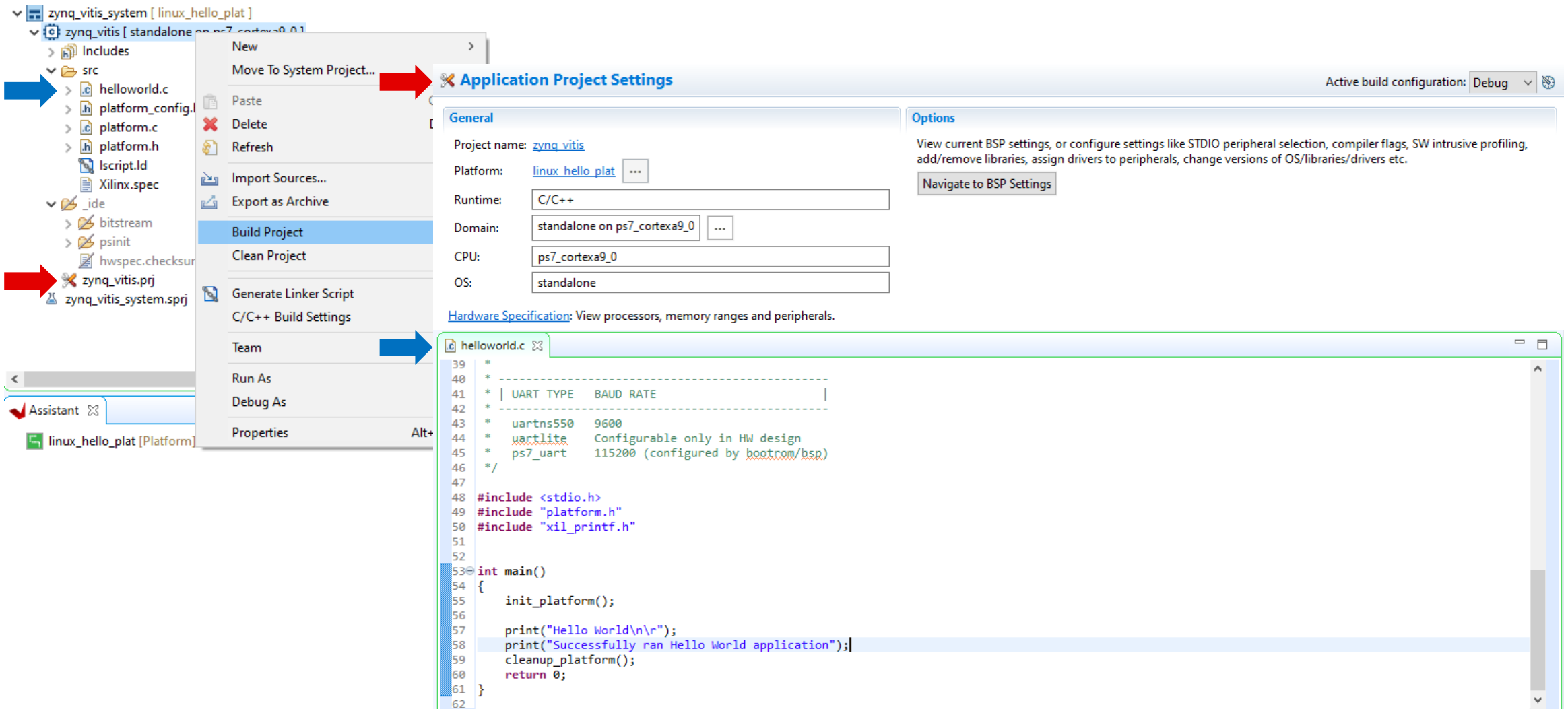


Creating the Hello World Application



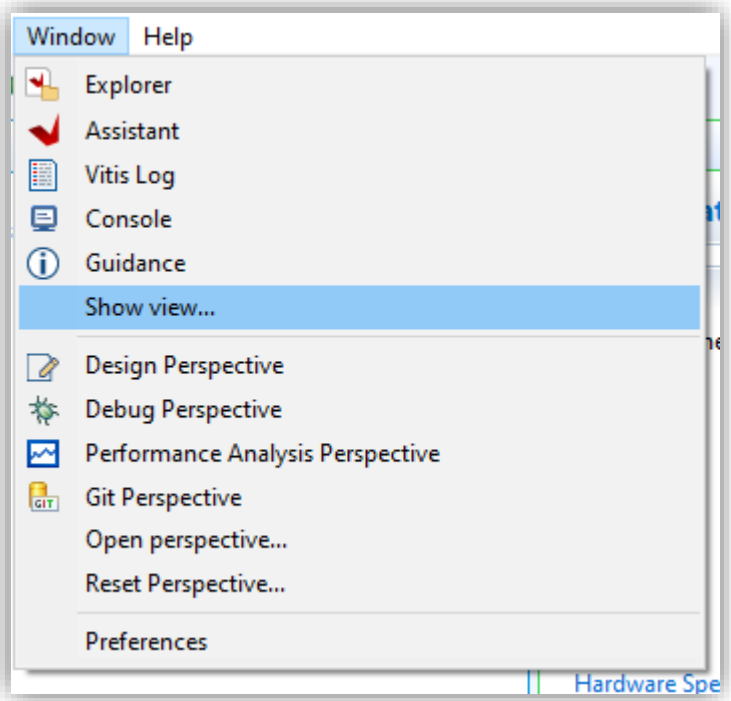
AMD XILINX

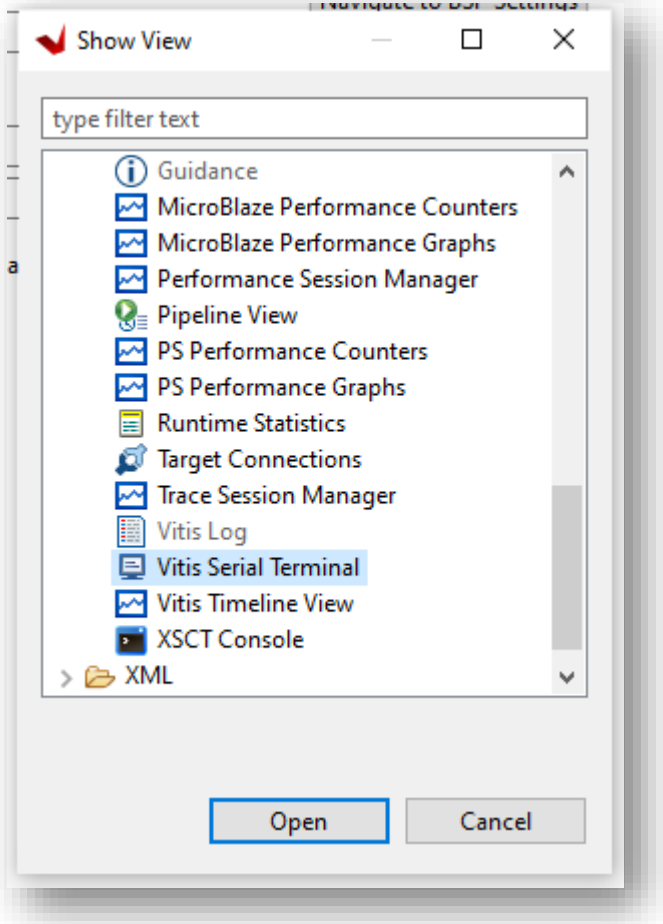
Creating the Hello World Application

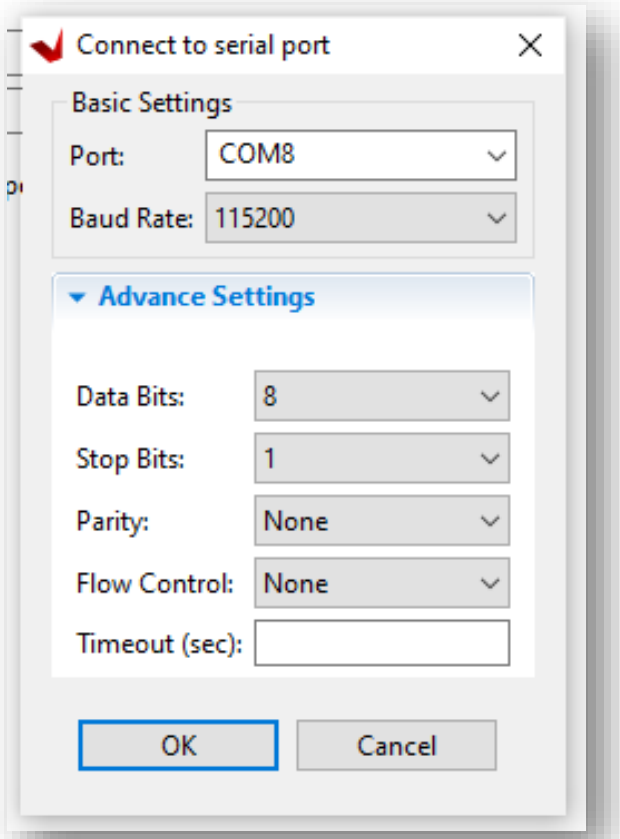


AMD XILINX

Running the Hello World Application on a ZC702 Board

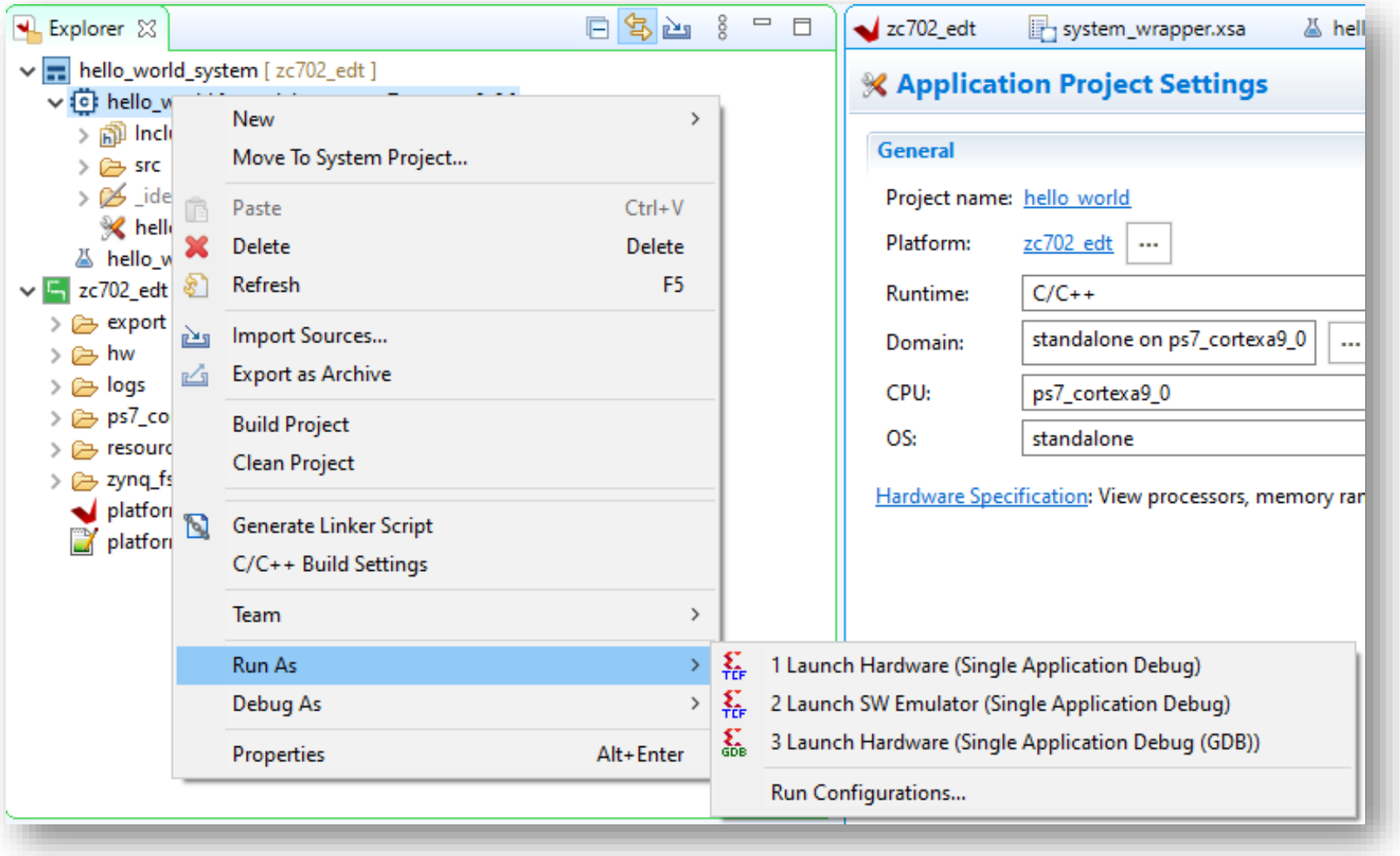






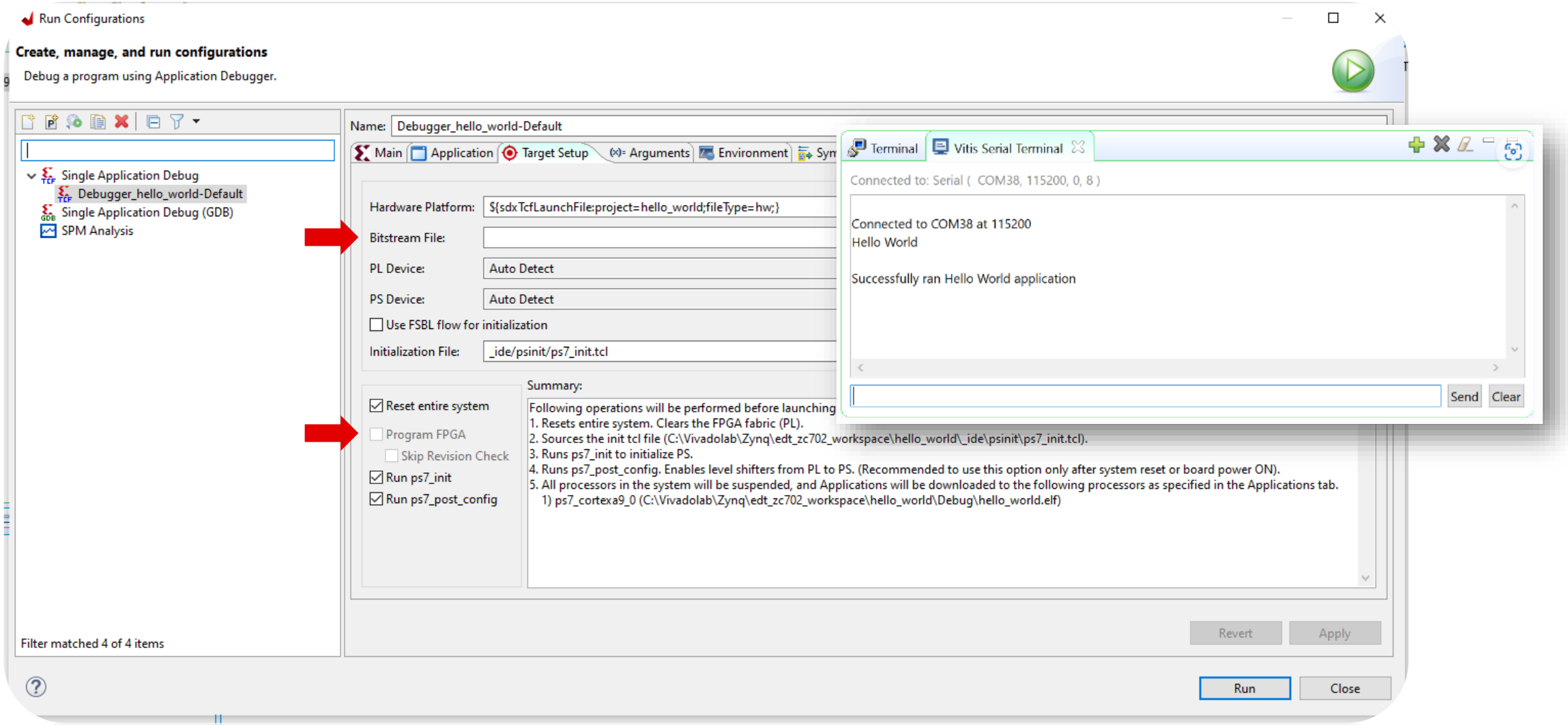
AMD
XILINX

Running the Hello World Application on a ZC702 Board



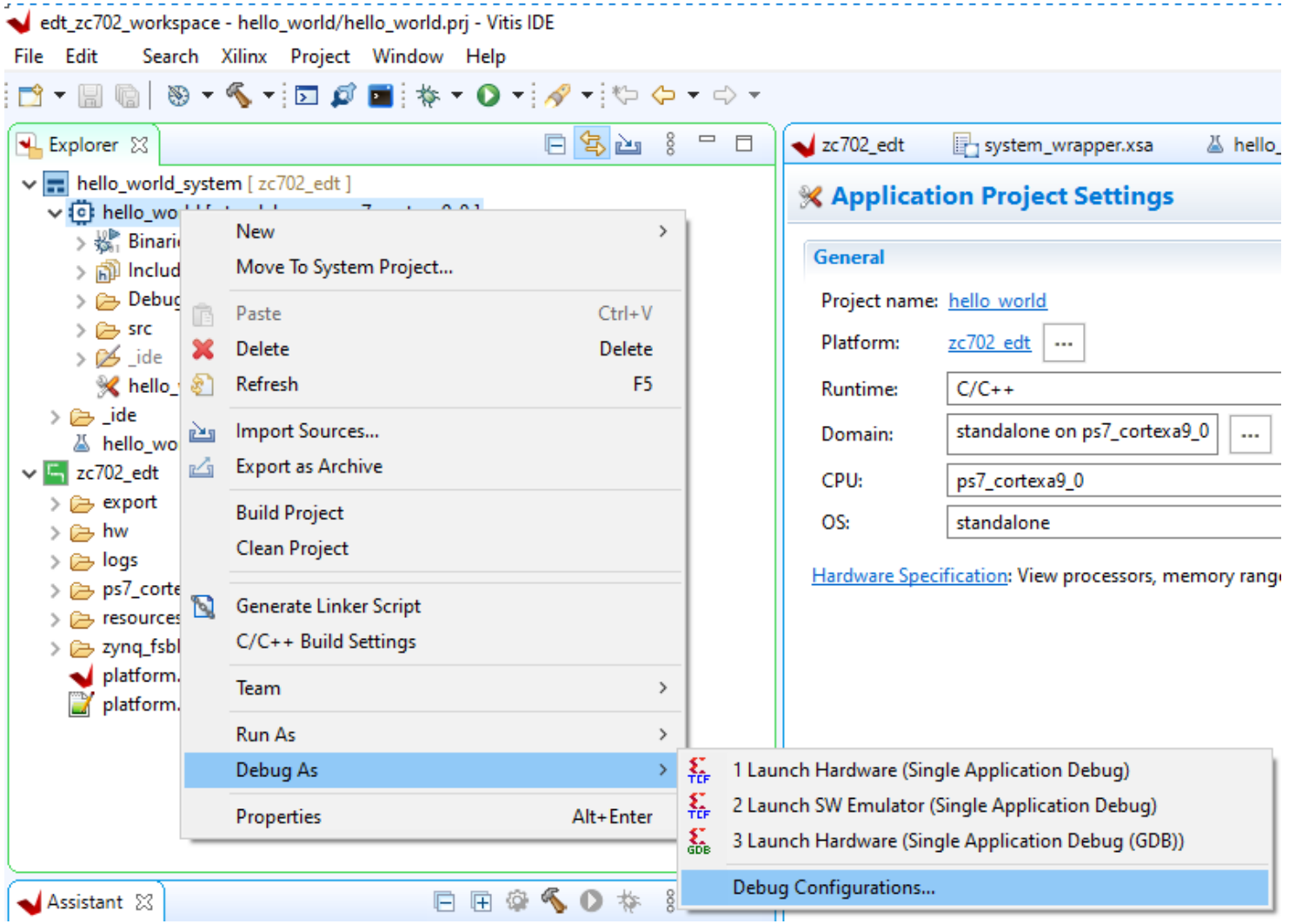
AMD XILINX

Running the Hello World Application on a ZC702 Board



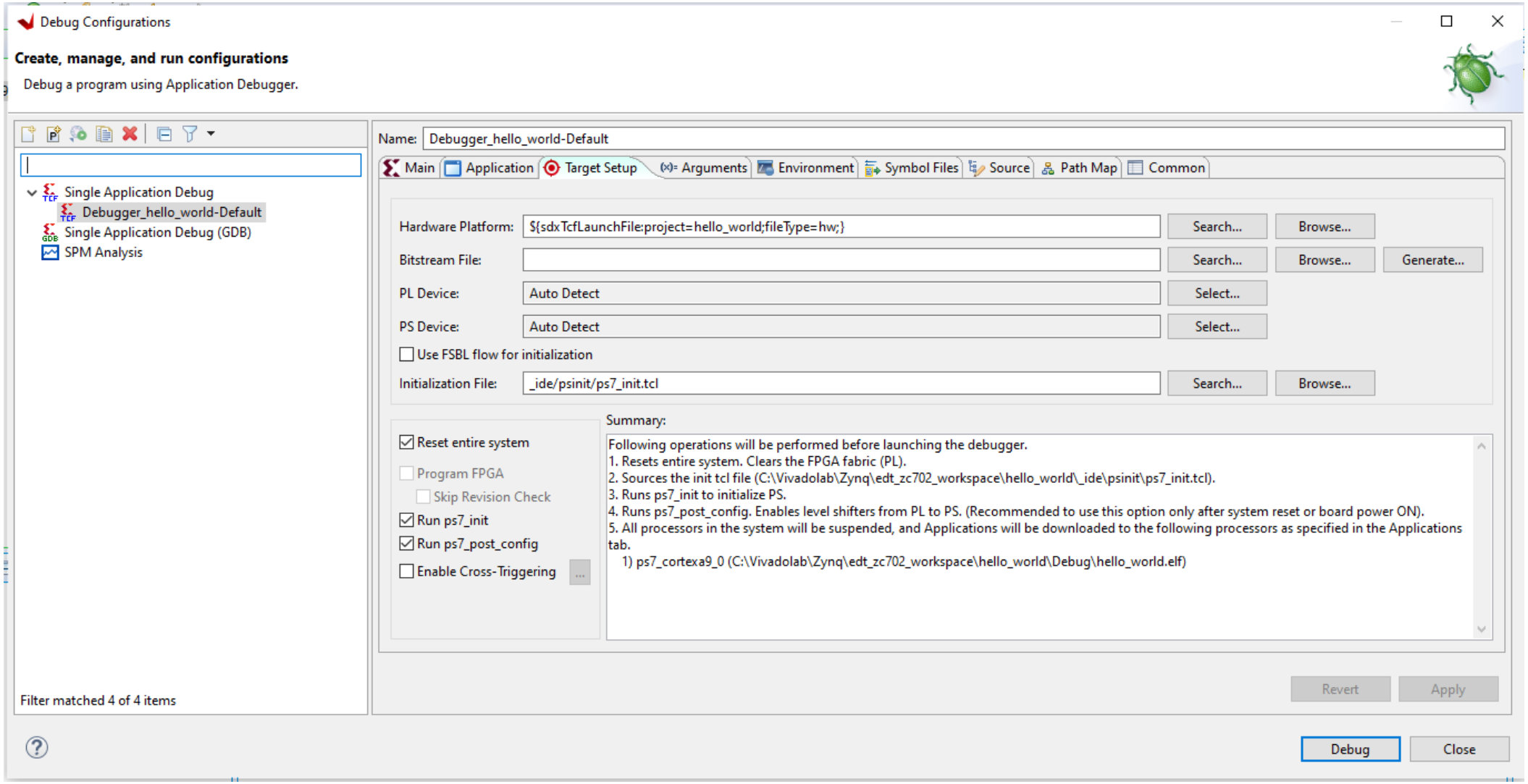
AMD XILINX

Debugging Standalone Applications in the Vitis IDE



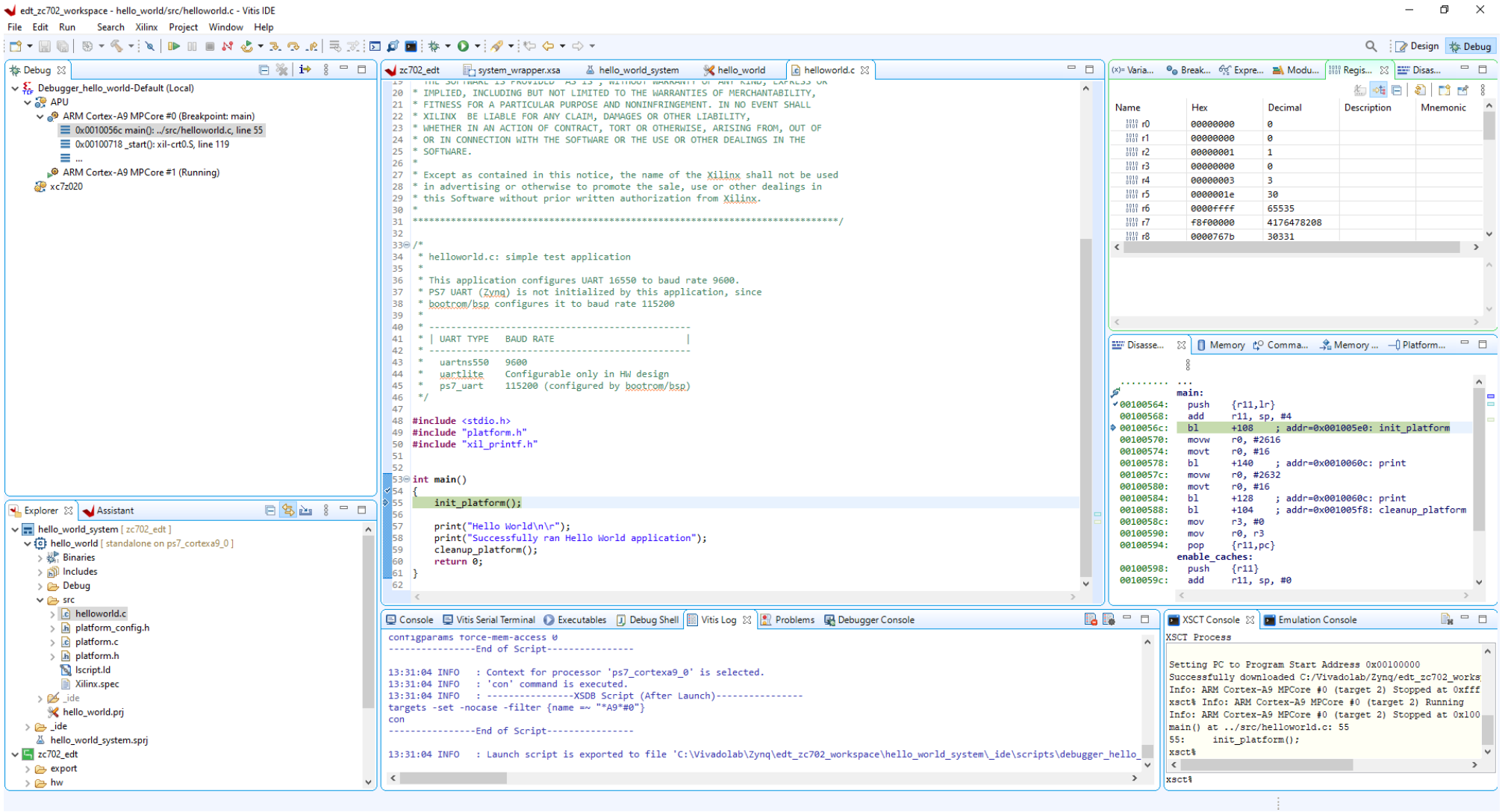
AMD XILINX

Debugging Standalone Applications in the Vitis IDE



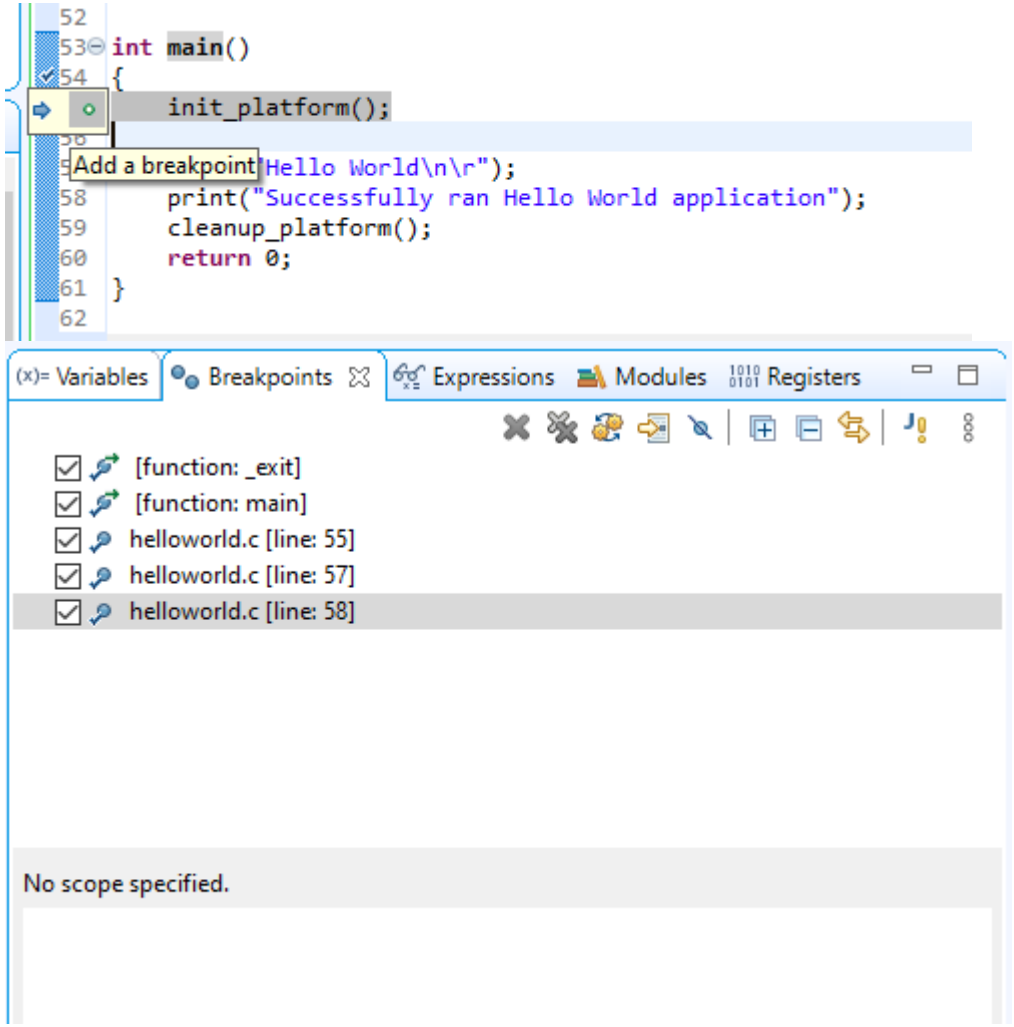
AMD XILINX

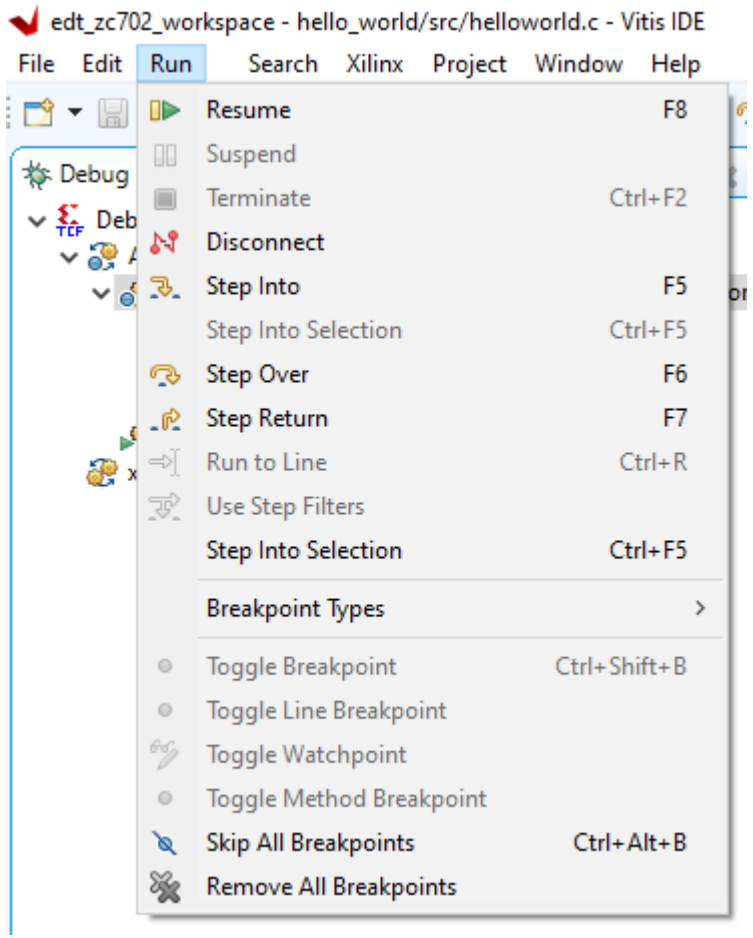
Debugging Standalone Applications in the Vitis IDE



AMD XILINX

Debugging Standalone Applications in the Vitis IDE



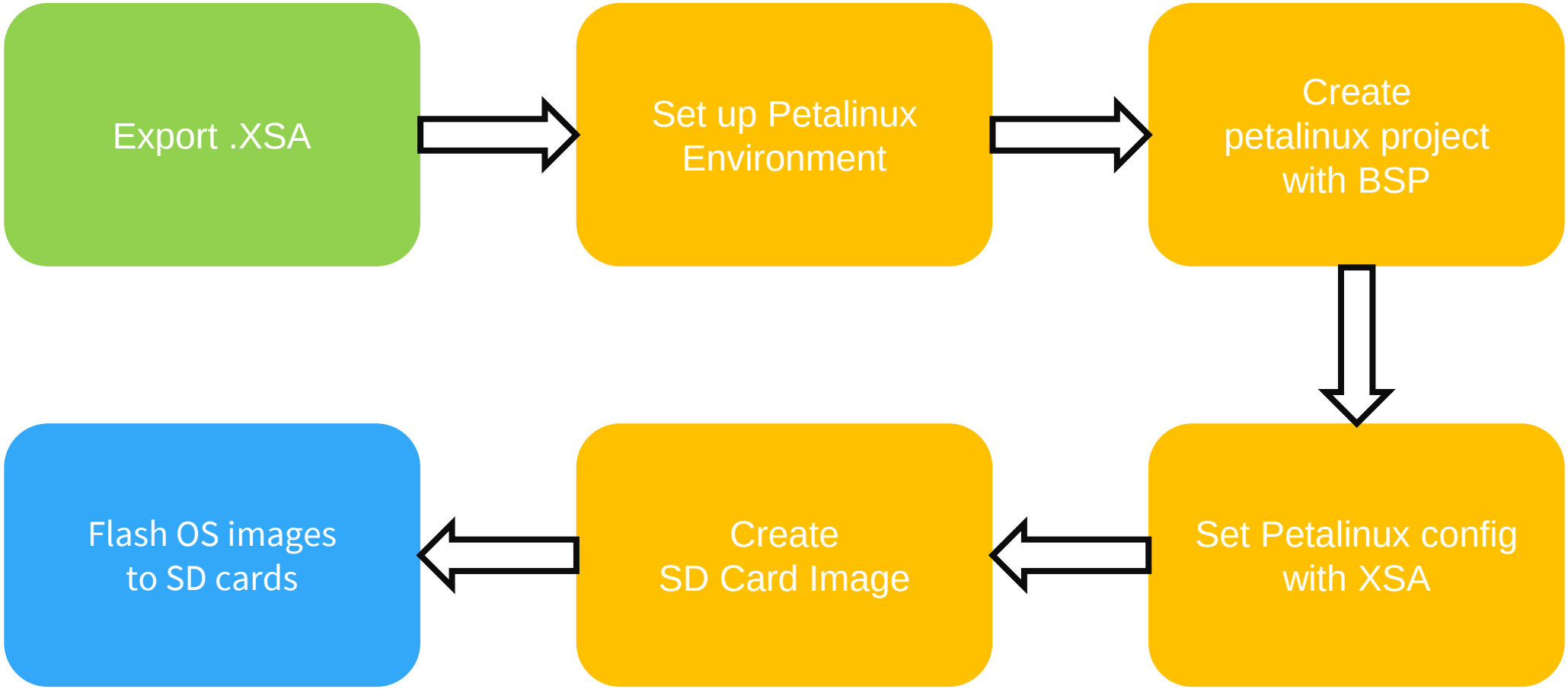


AMD
XILINX

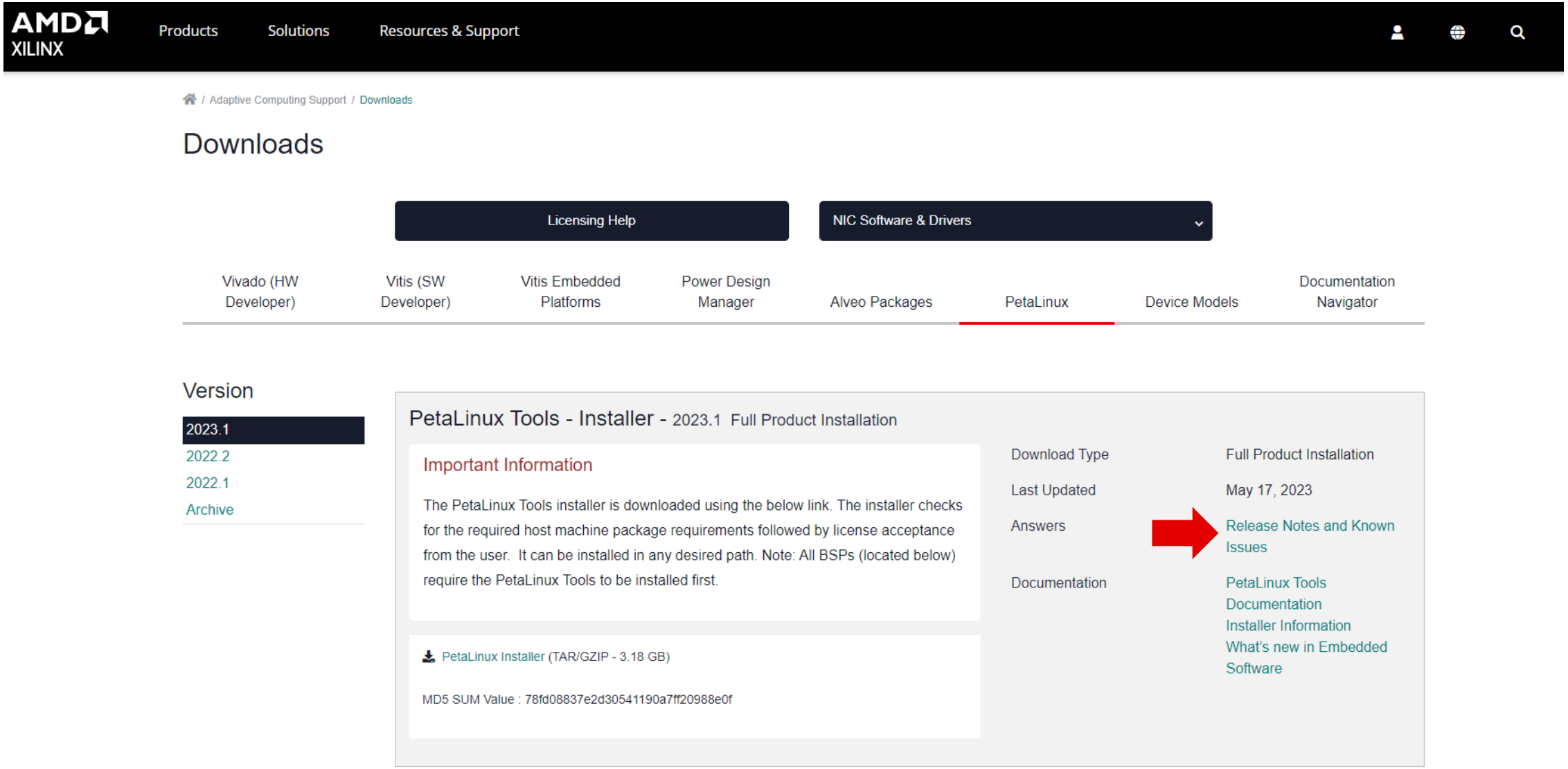
Building and Debugging Linux Applications

Course Agenda
2023

Build Petalinux



Install Petalinux



AMD XILINX

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

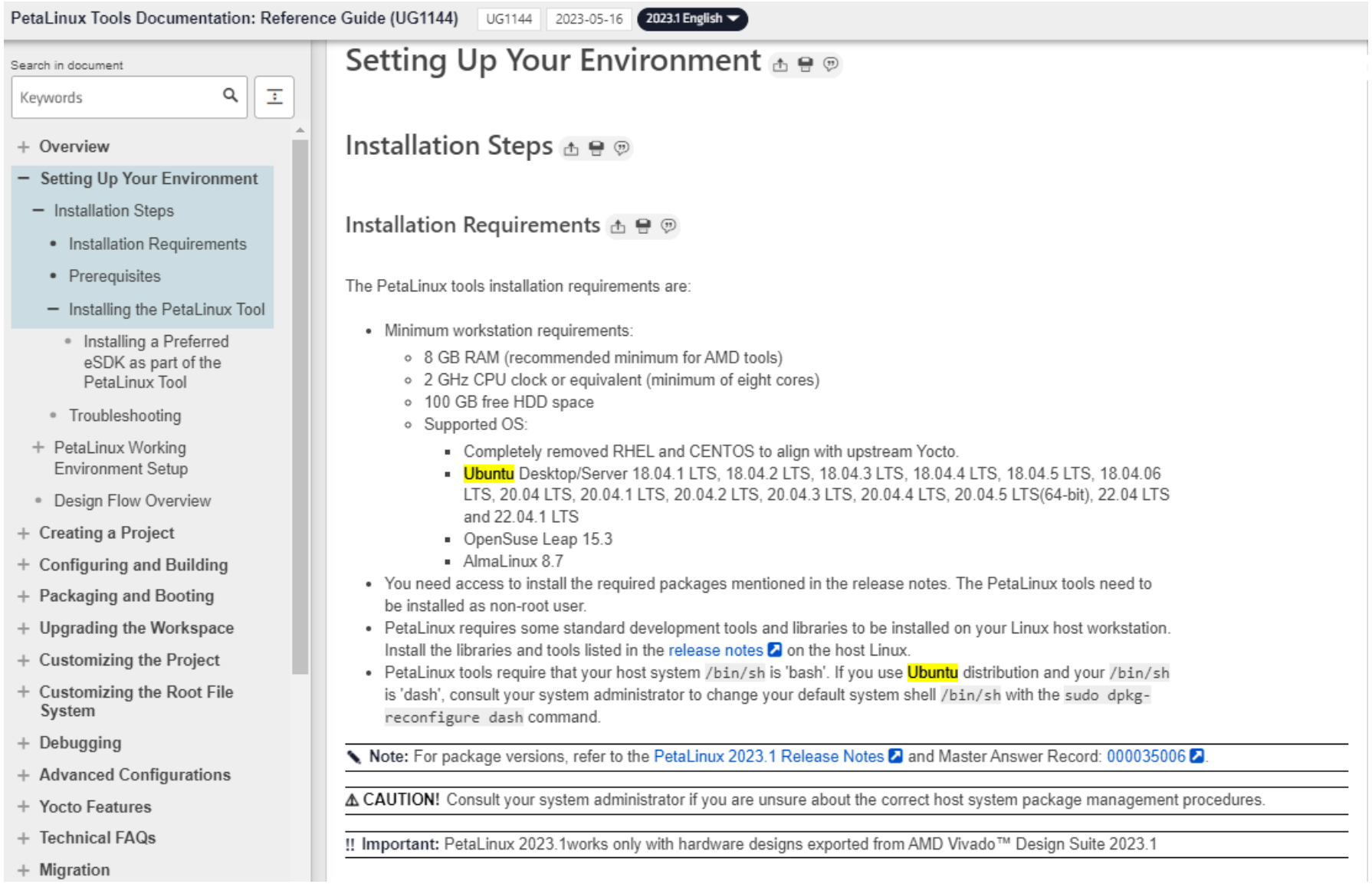
BSPs supported for the PetaLinux 2023.1 Release

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
1	MicroBlaze	AC701	xilinx-a... 050802...
			Linux.
6	Zynq 7000	ZC702	xilinx-zc702-v2023.1-05080224.bsp	This BSP contains: • **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.
				This BSP contains: • **Hardware:** This design uses Vivado board presets with Zynq-7000 PS block (DDR, UART, SD, QSPI, Ethernet

AMD XILINX

Install Petalinux



AMD XILINX

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	
2022.2_PetaLinux_Package_List.xlsx	28.03 KB	

Community Feedback?

AMD XILINX

Install Petalinux

Installation zlib1g:i386

<command> sudo dpkg –add-architecture i386

<command> sudo apt-get update

<command> sudo apt-get install zlib1gLi386

```
devin@LAPTOP-90IBO783:/$ sudo dpkg --add-architecture i386
devin@LAPTOP-90IBO783:/$ 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]
```

```
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-90IBO783:/$ sudo apt-get install zlib1g:i386
```


AMD XILINX

Install Petalinux

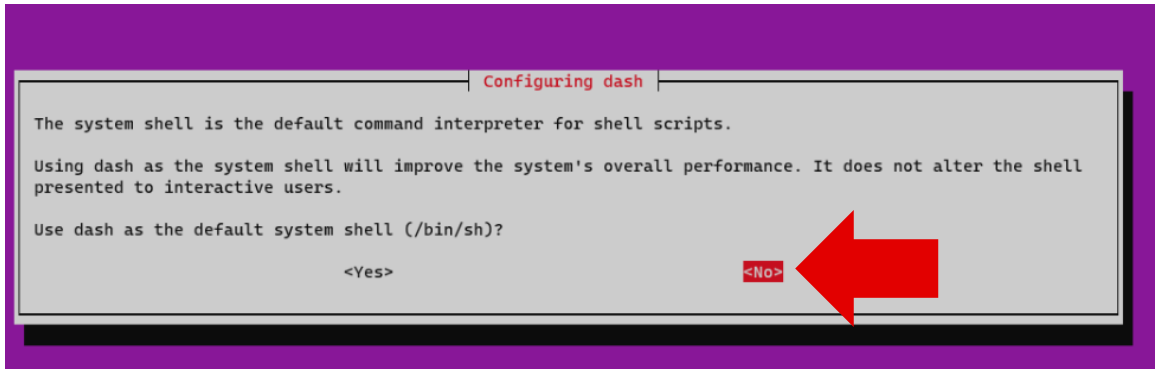
Installation petalinux 2022.2

```
devin@LAPTOP-90IBO783:/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-90IBO783:/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)

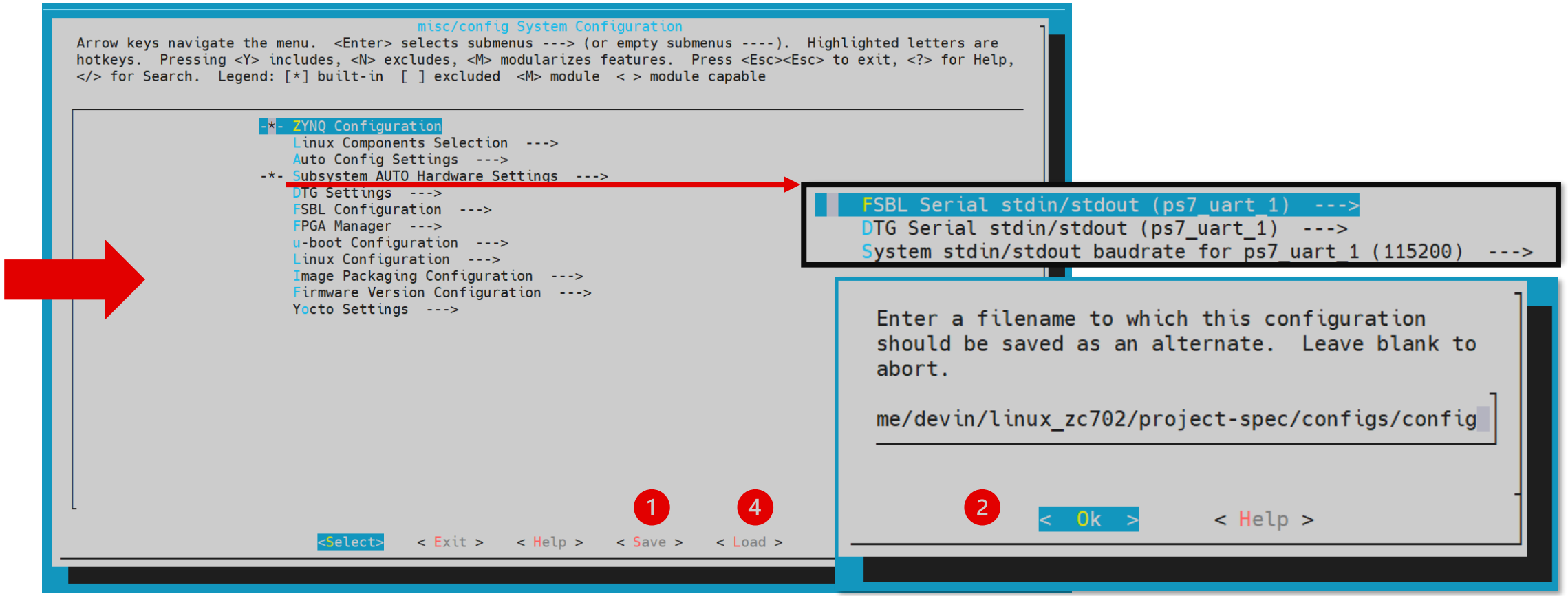
Build Petalinux Project

Import XSA

<command> cd <Project Directory>

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

3 **<command> petalinux-config**



AMD XILINX

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

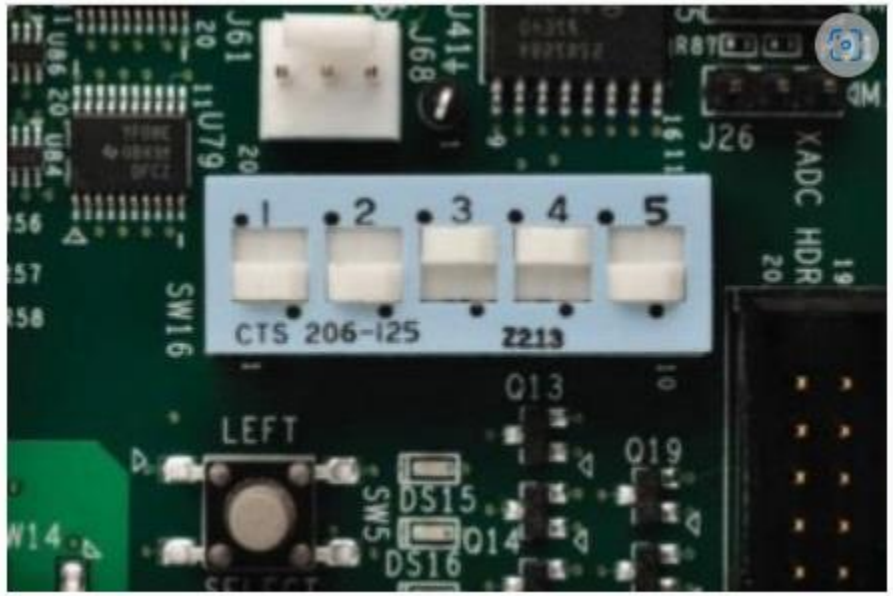
<command> petalinux-package --boot --fsbl ./zynq_fsbl.elf –fpga ./system.bit –u-boot ./u-boot.elf --force

```
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

AMD XILINX

Build Petalinux Project



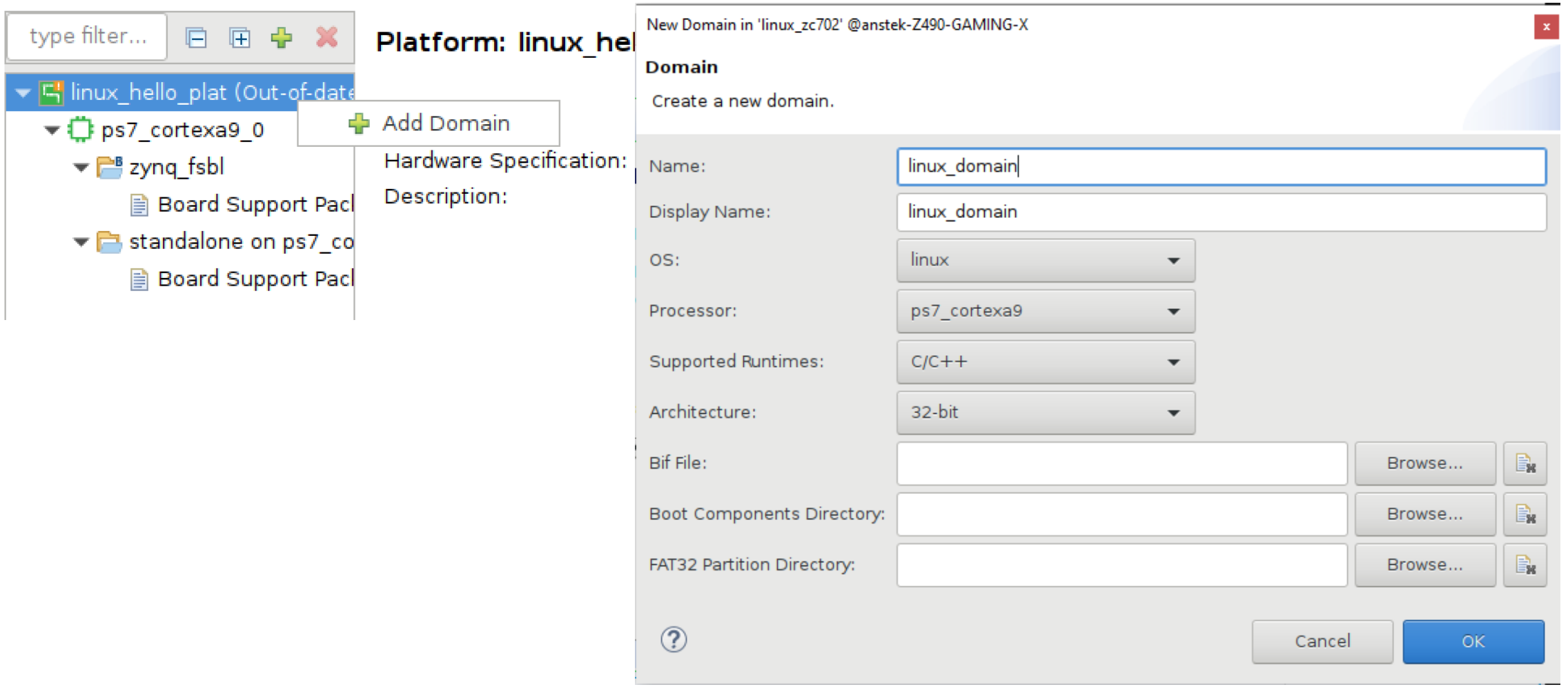
SD Boot Mode Setup for SW16

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

Ethernet Jumper

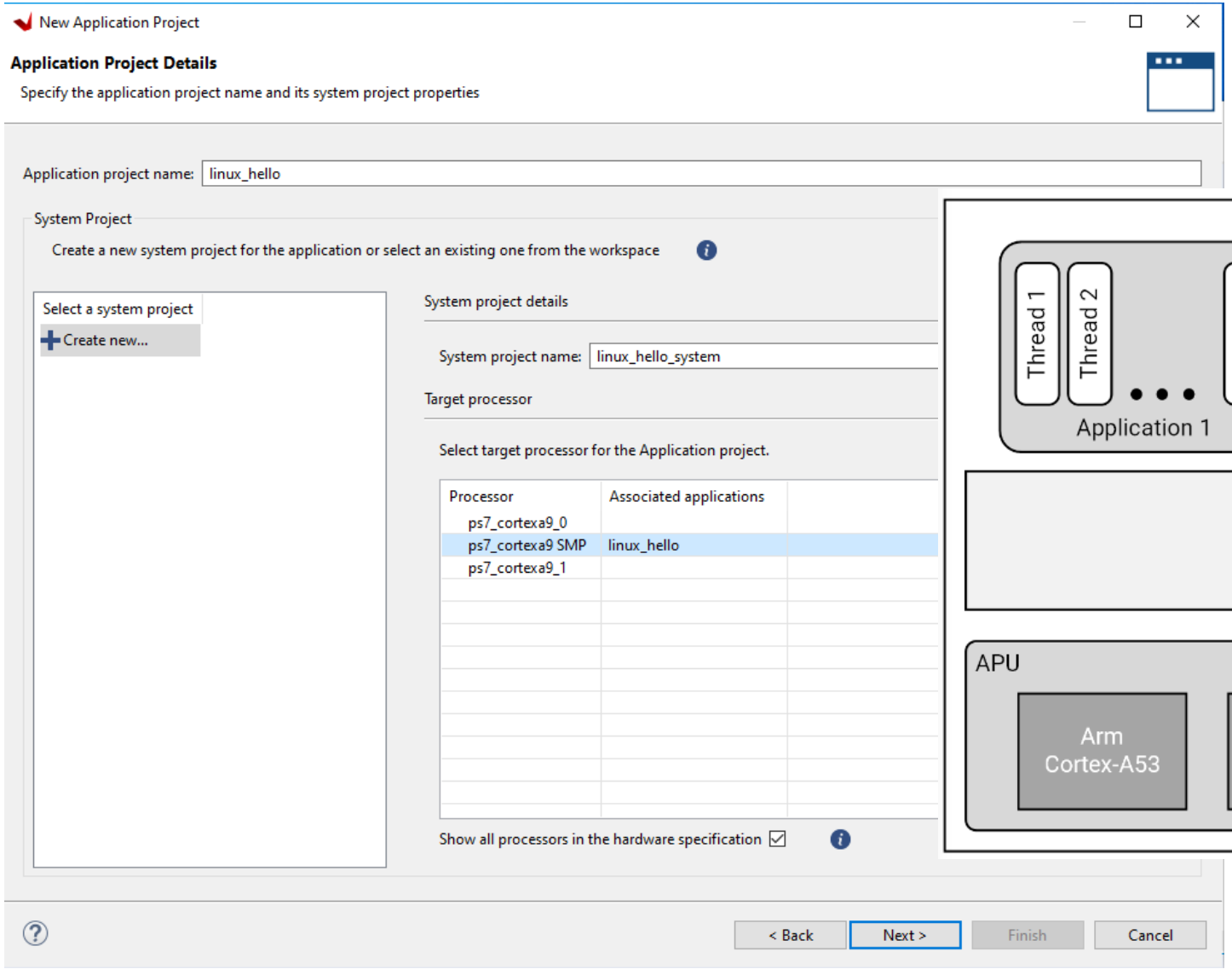
```
xilinx-zc702-2022_2 login: petalinux
You are required to change your password immediately (administrator enforced).
New password:
Retype new password:
xilinx-zc702-2022_2:~$
xilinx-zc702-2022_2:~$
xilinx-zc702-2022_2:~$
xilinx-zc702-2022_2:~$
```

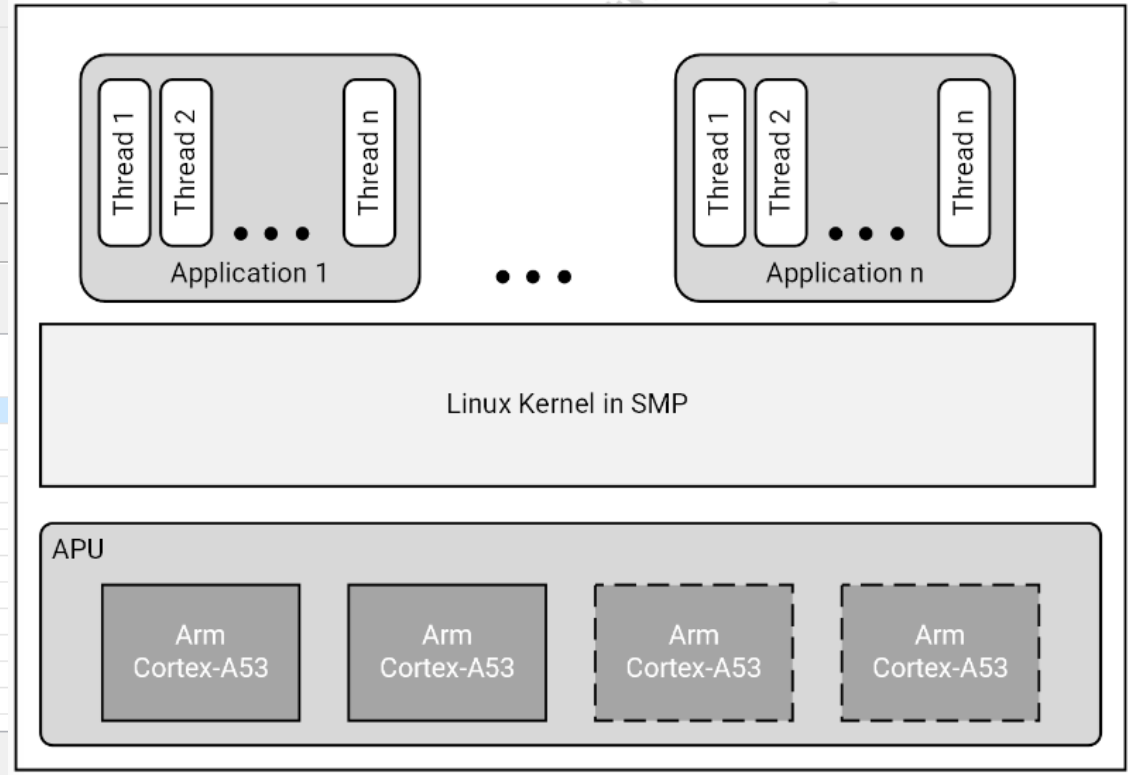
Build Linux Application



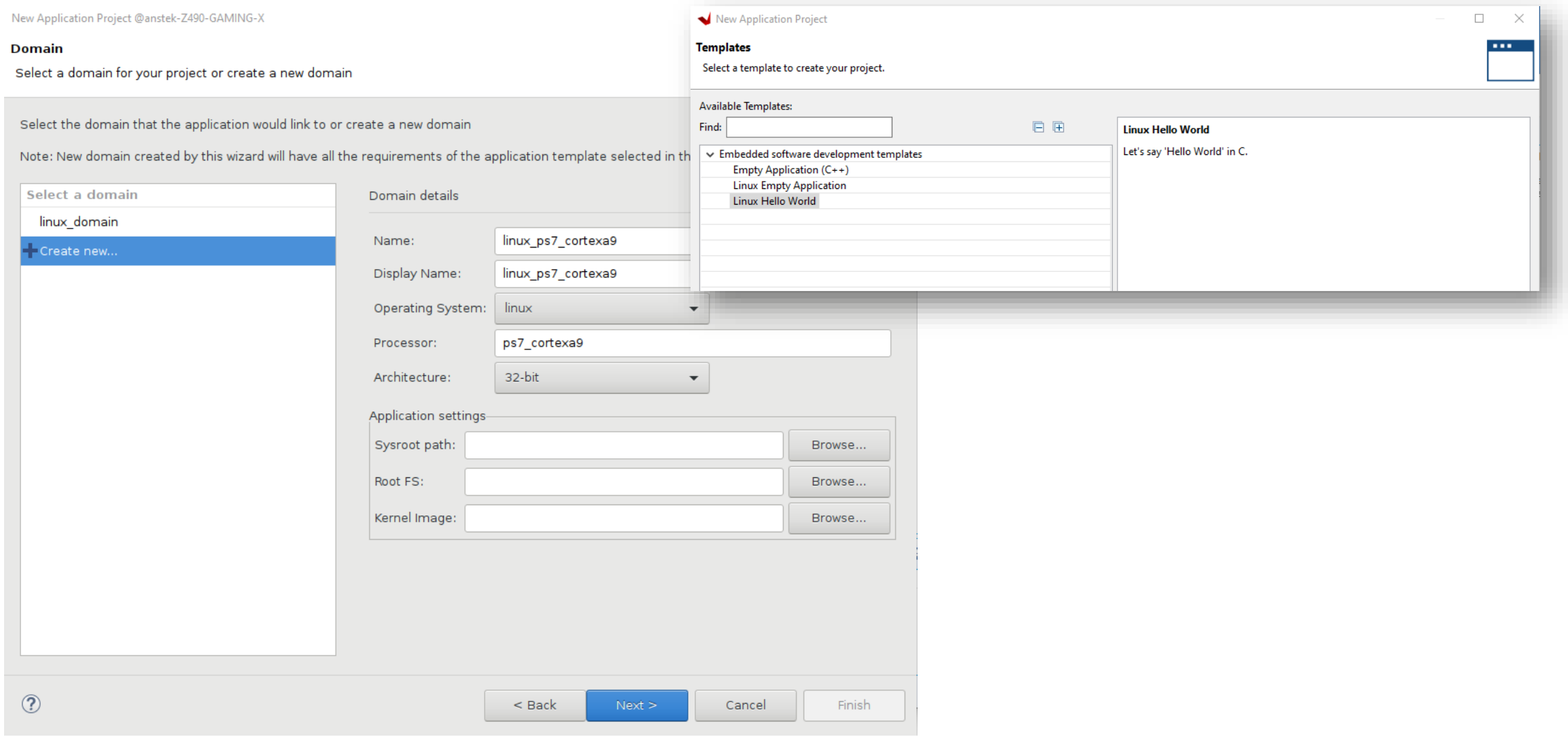
AMD XILINX

Build Linux Application



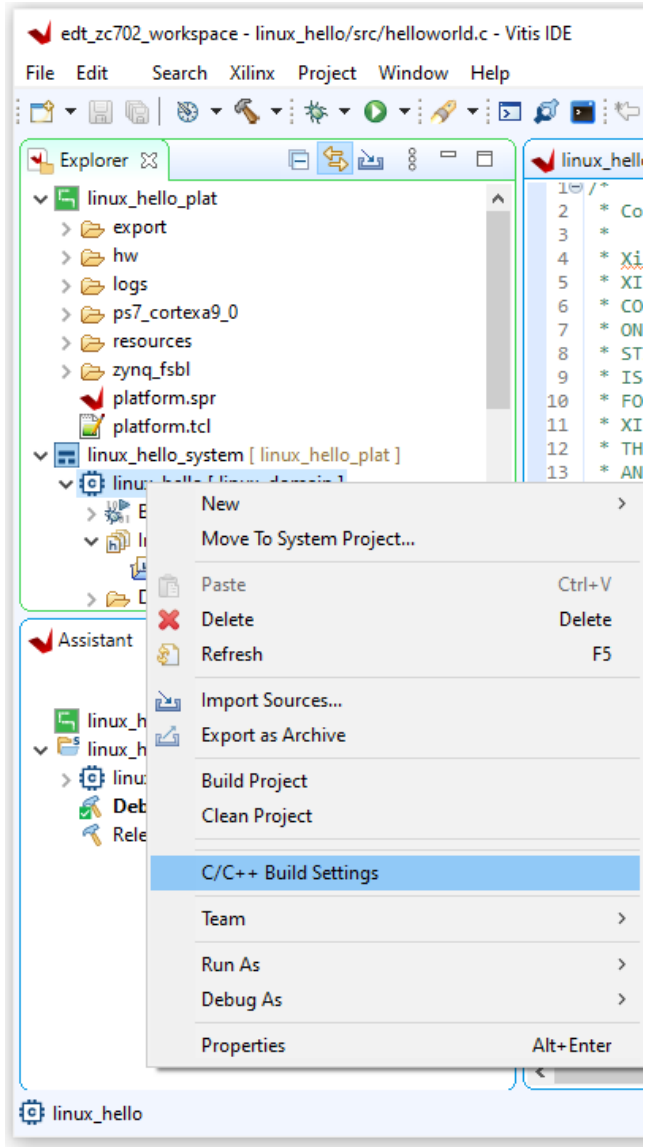


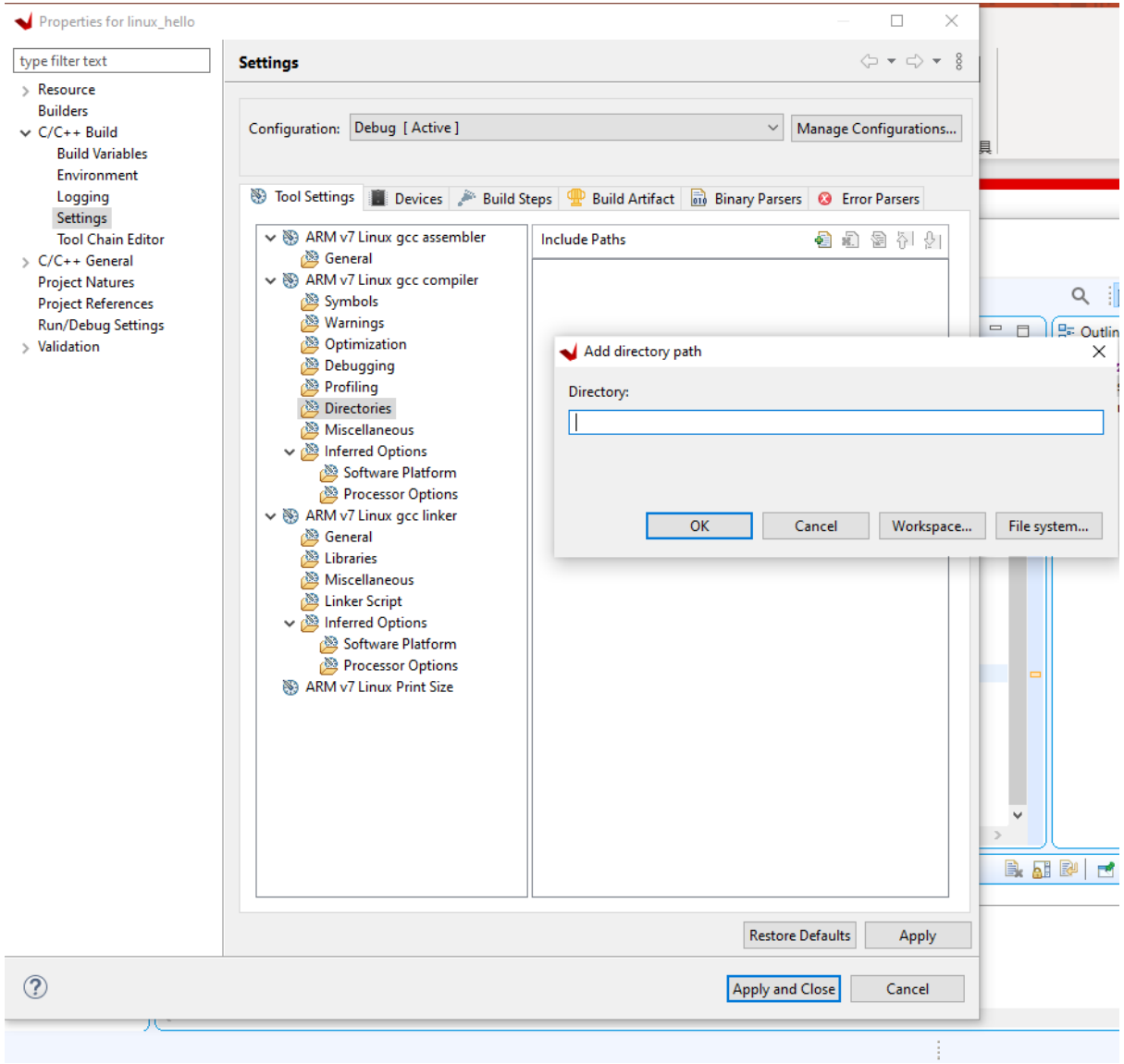
Build Linux Application



AMD
XILINX

Build Linux Application

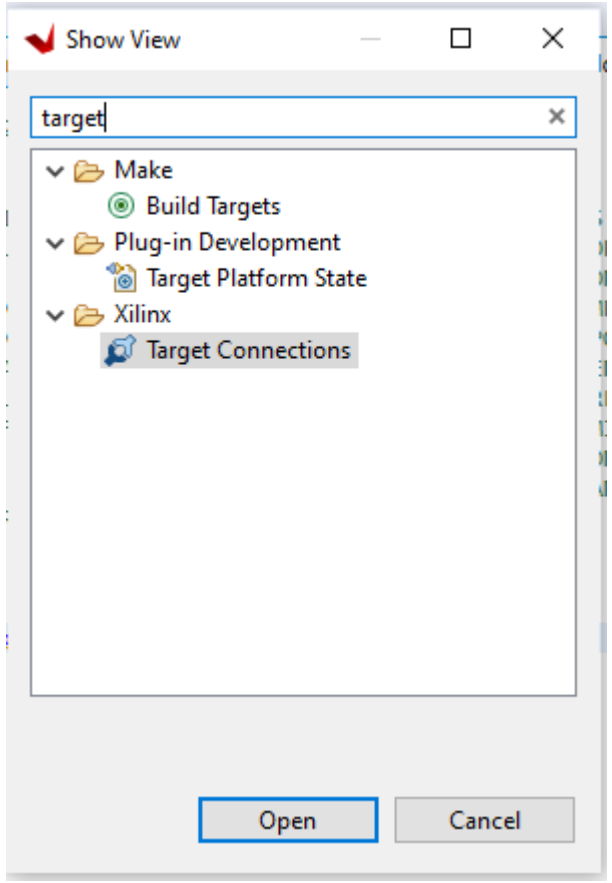




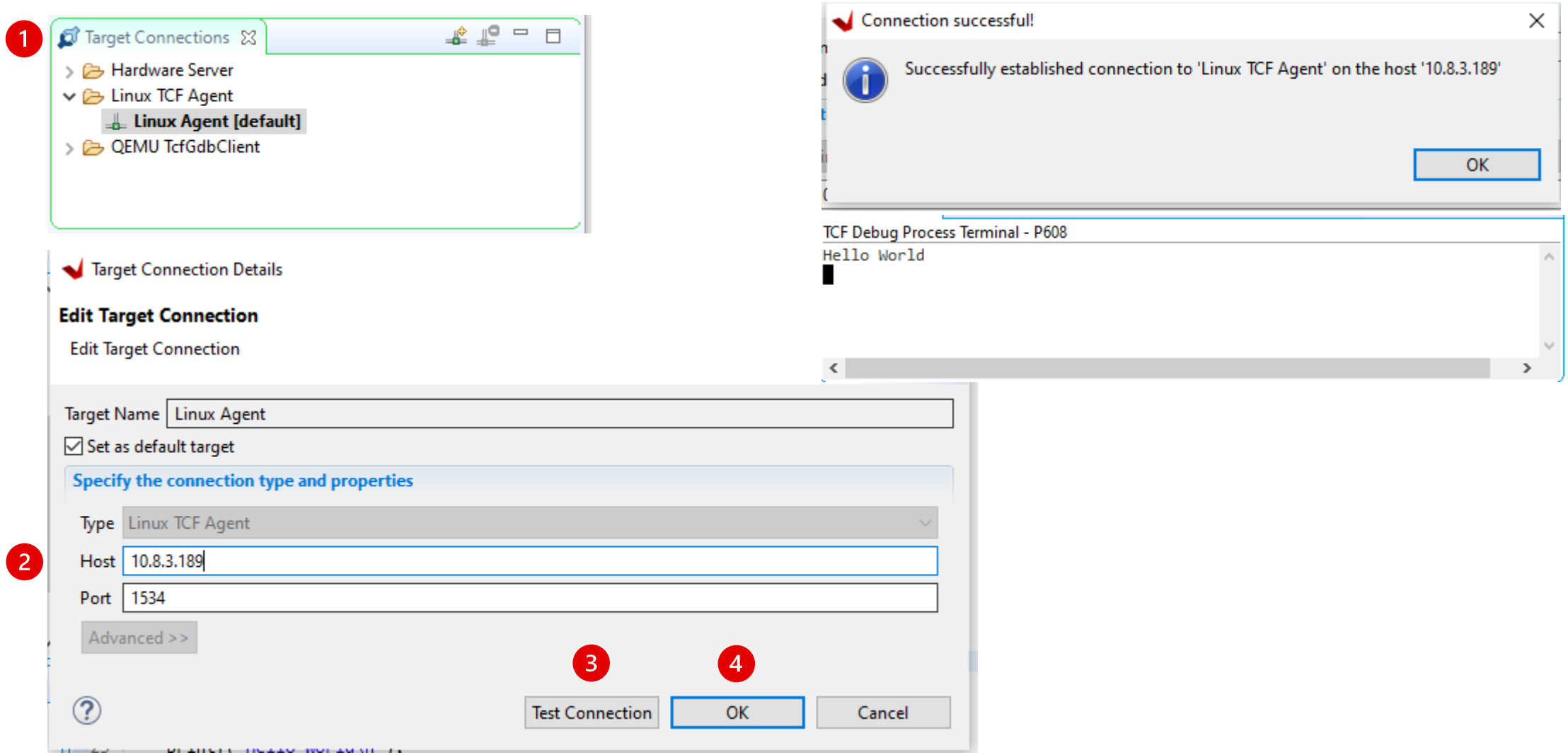
AMD XILINX

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:35ff: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



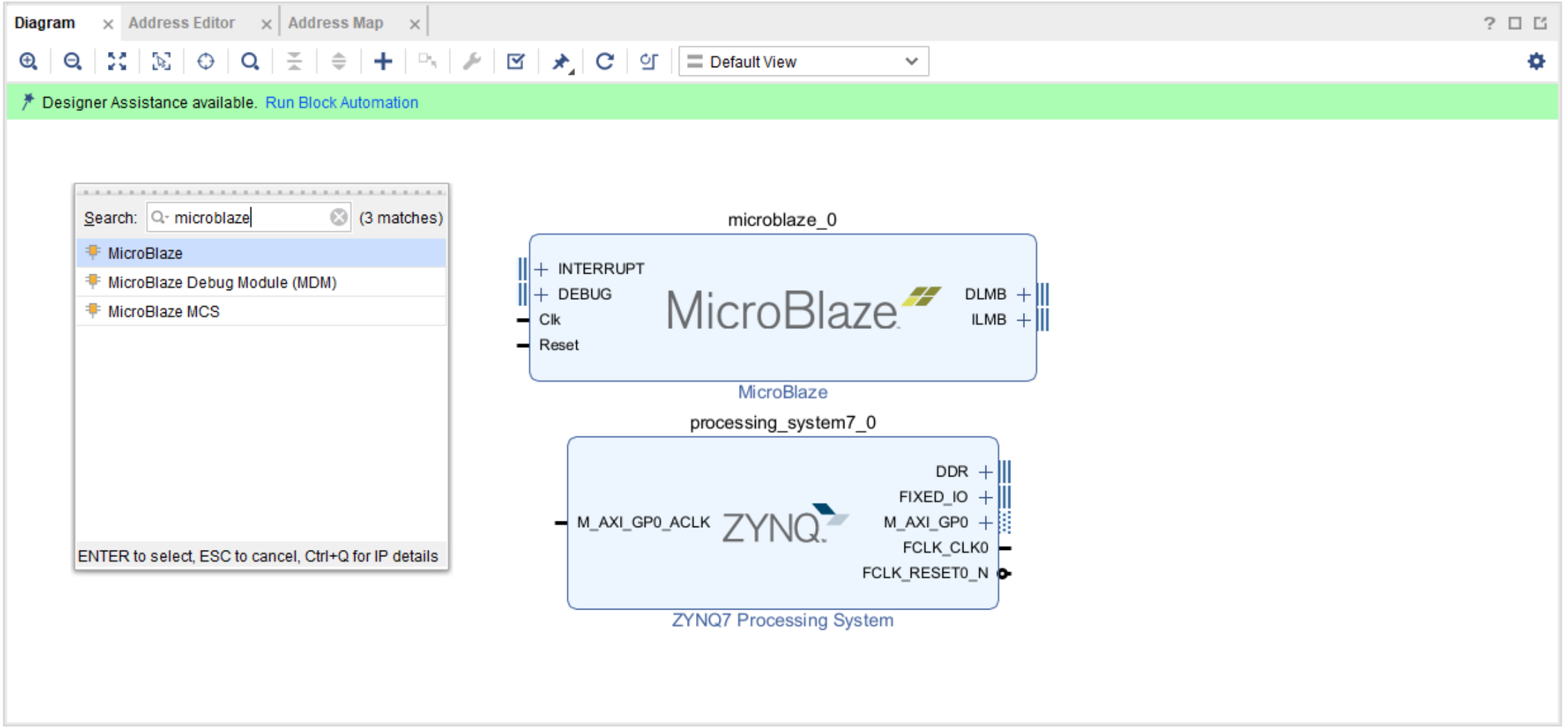
AMD XILINX

AMD
XILINX

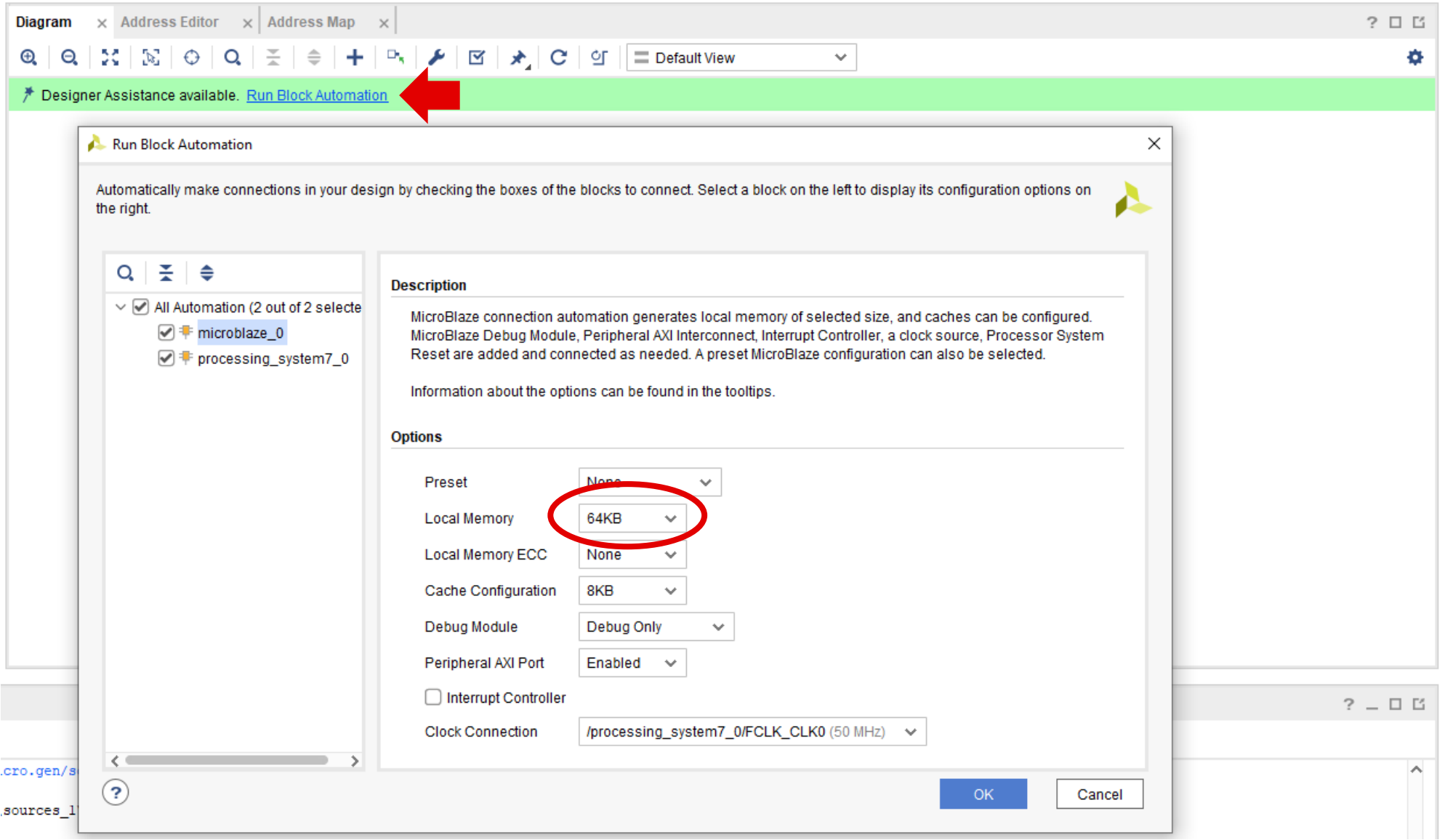
Design Microblaze via Vitis

Course Agenda
2023

Microblze Block Design

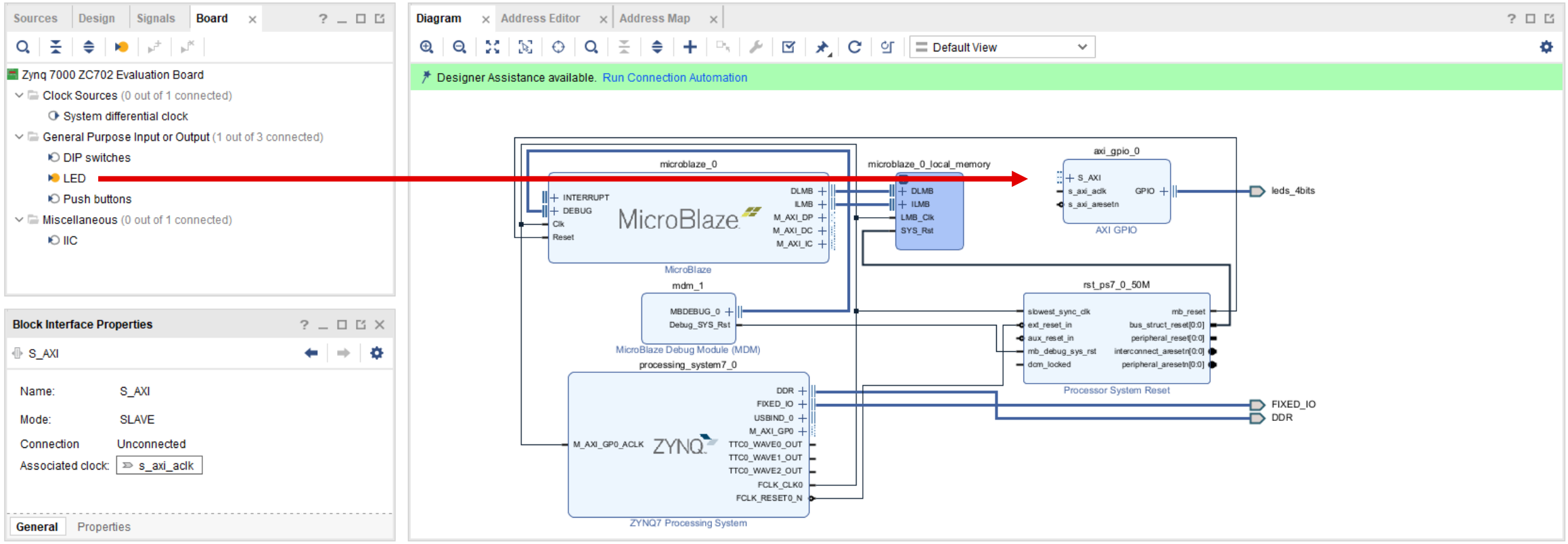


Microblze Block Design



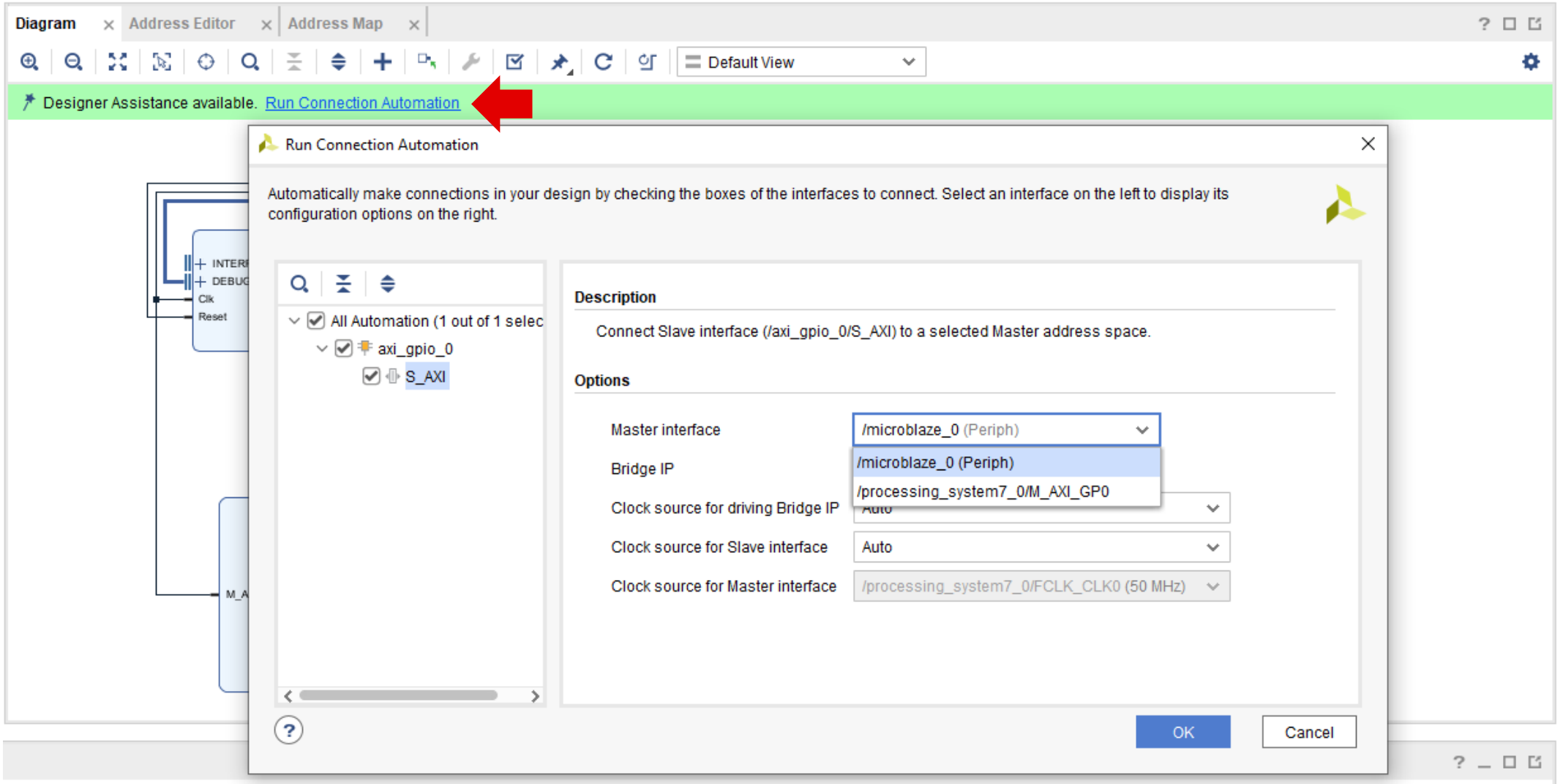
AMD XILINX

Microblze Block Design



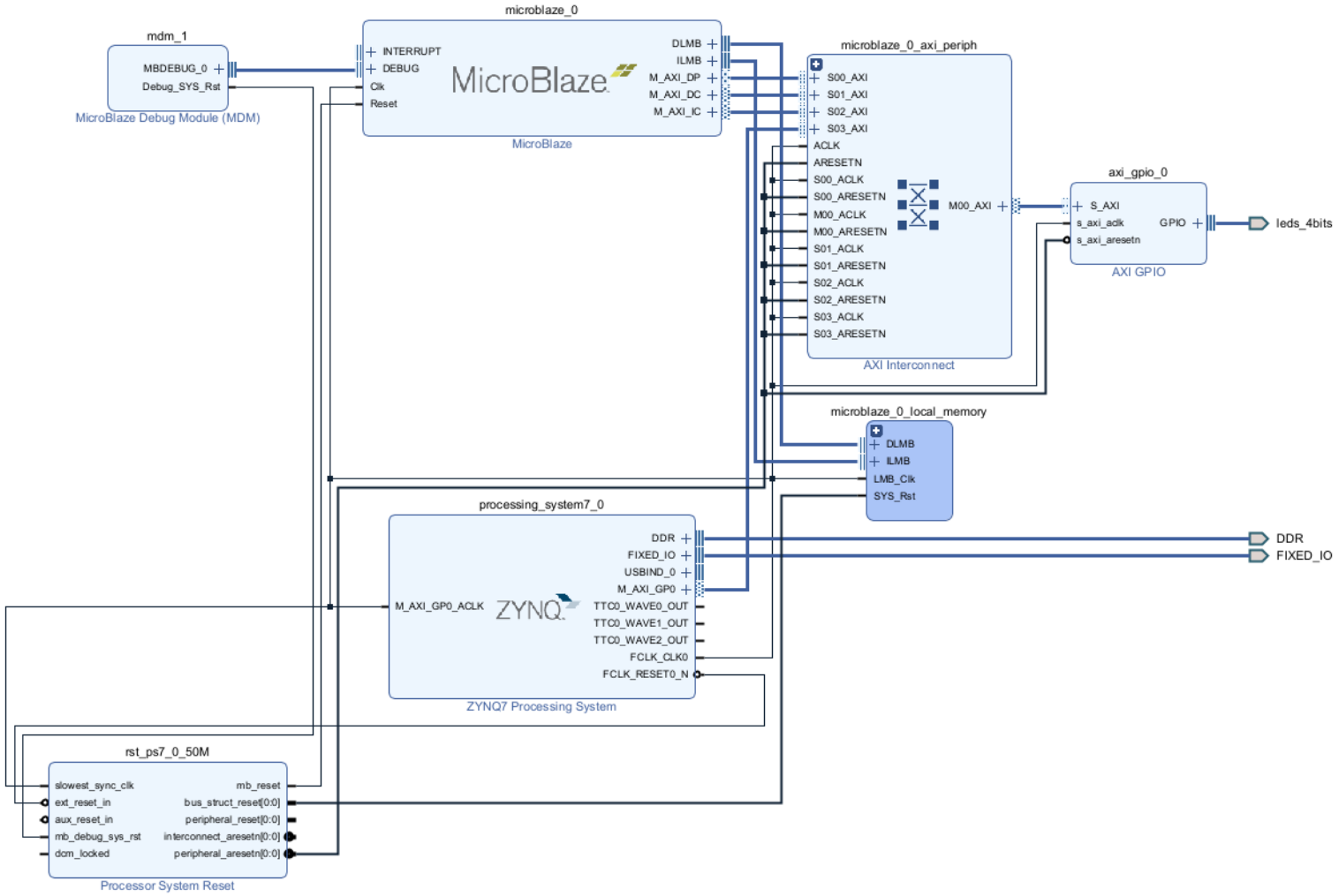
AMD XILINX

Microblze Block Design



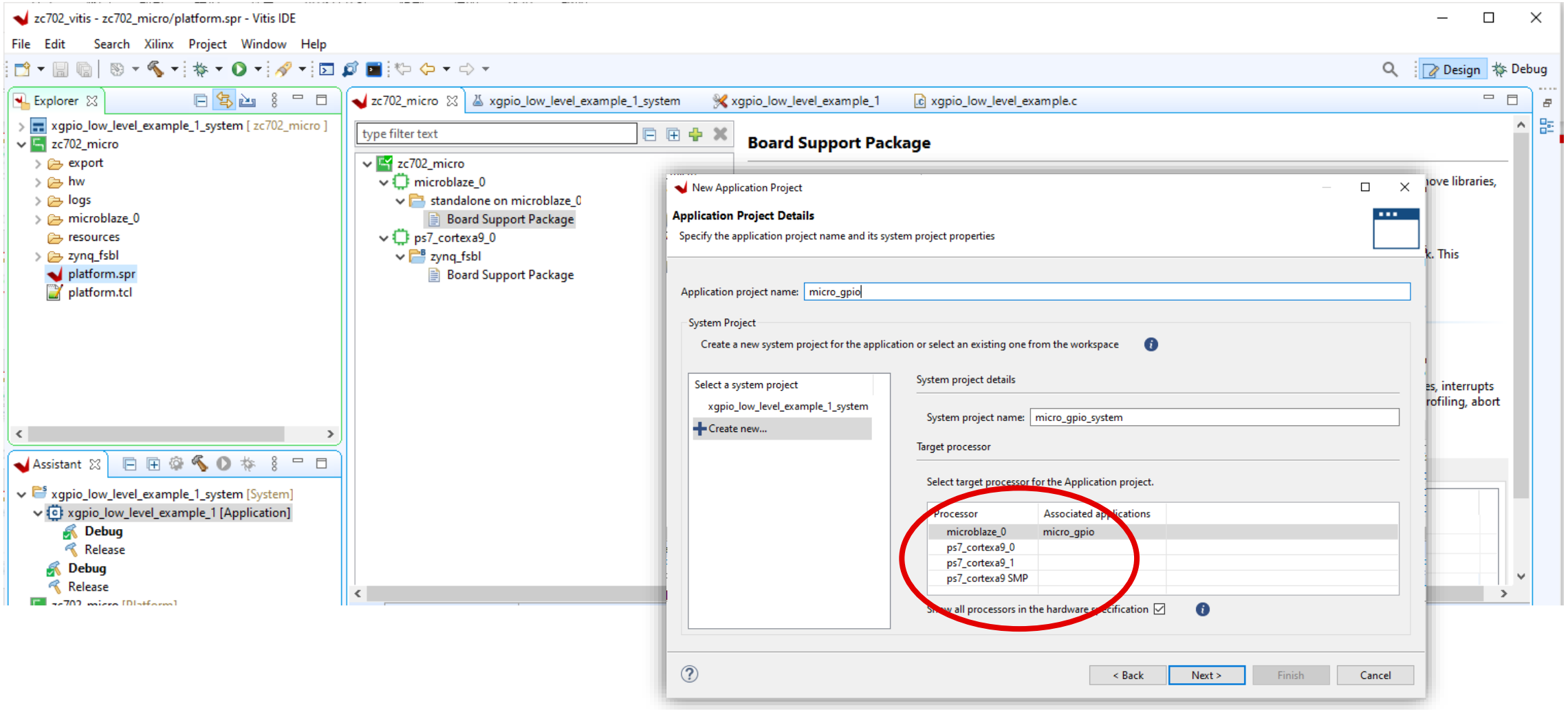
AMD XILINX

Microblze Block Design



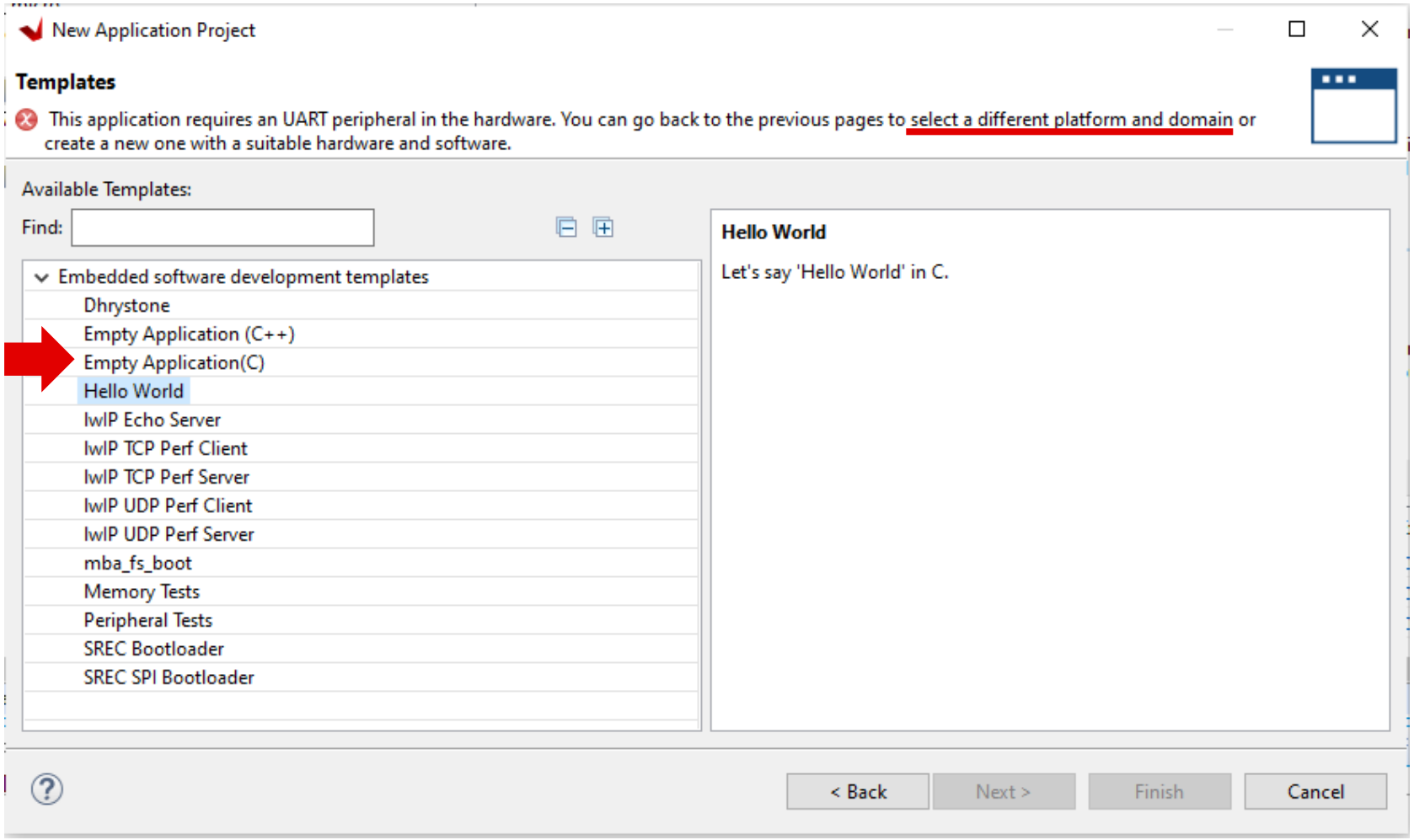
AMD XILINX

Microblze in Vitis

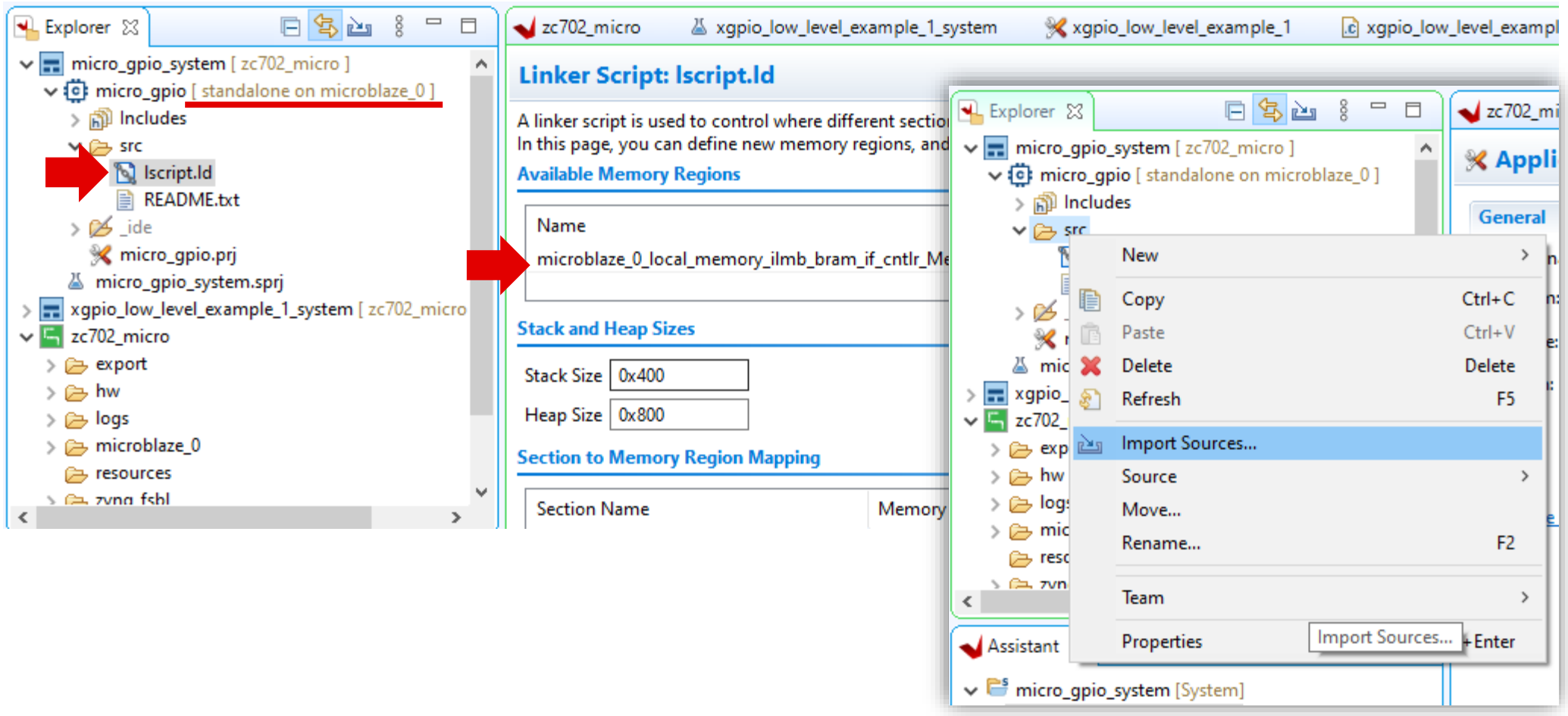


AMD XILINX

Microblze in Vitis

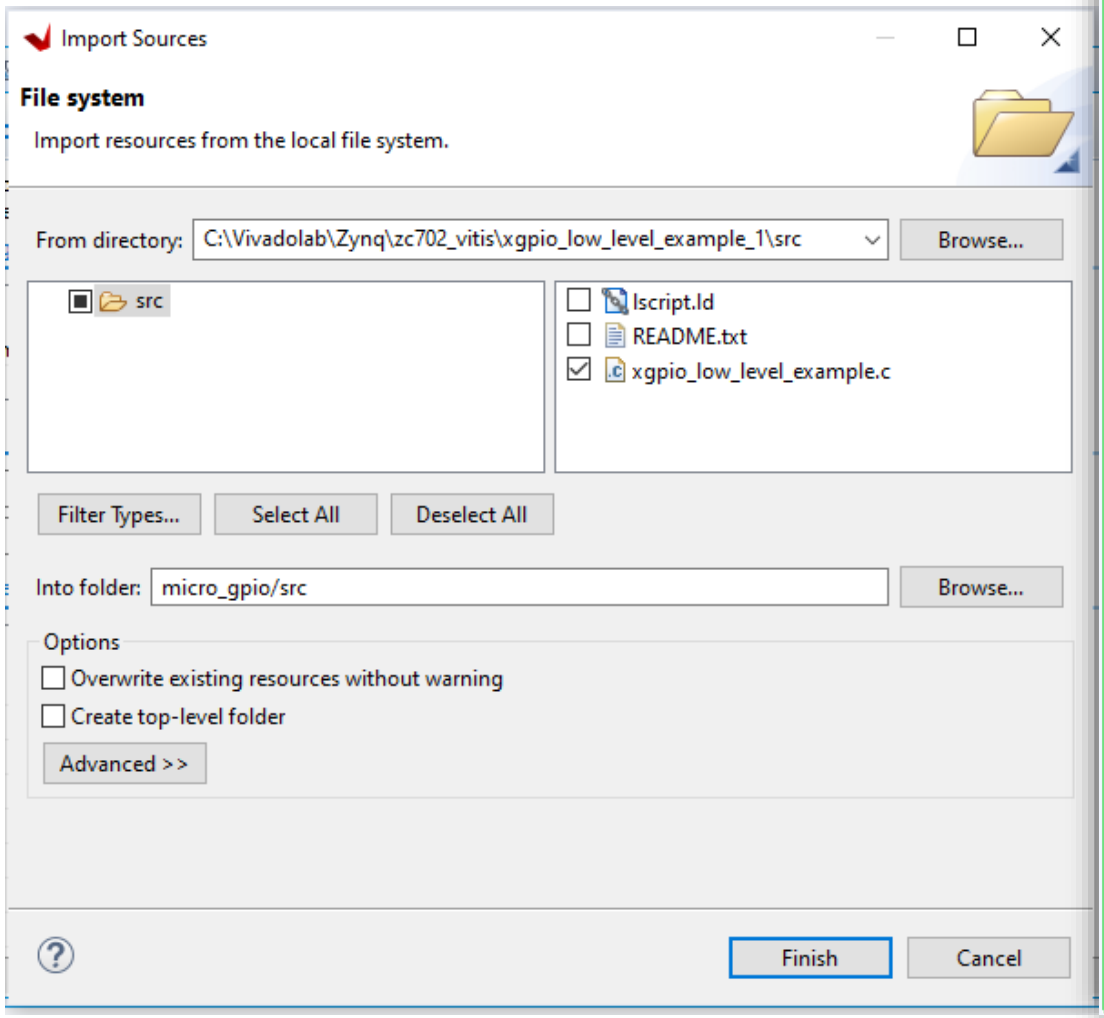


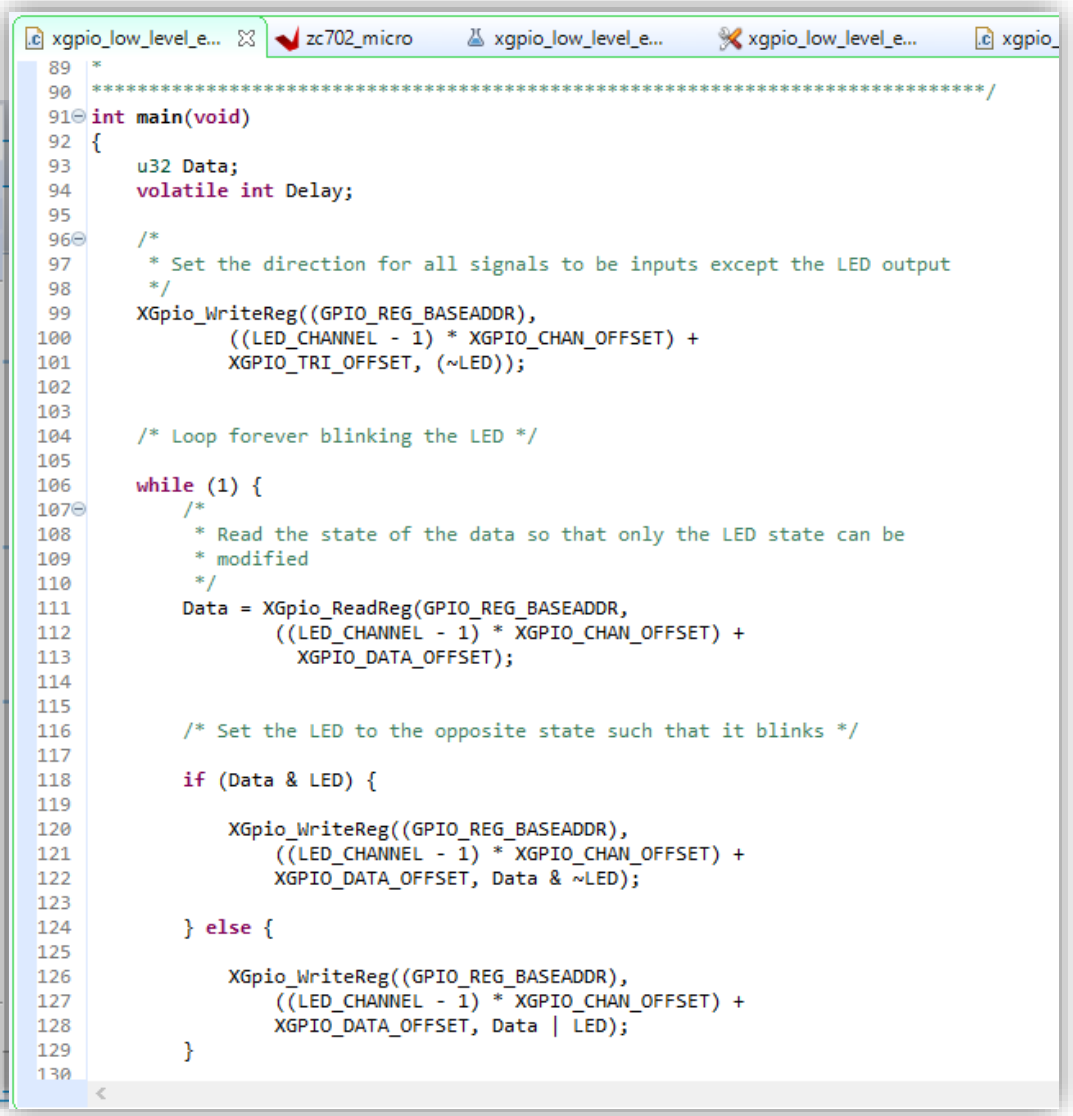
Microblze in Vitis



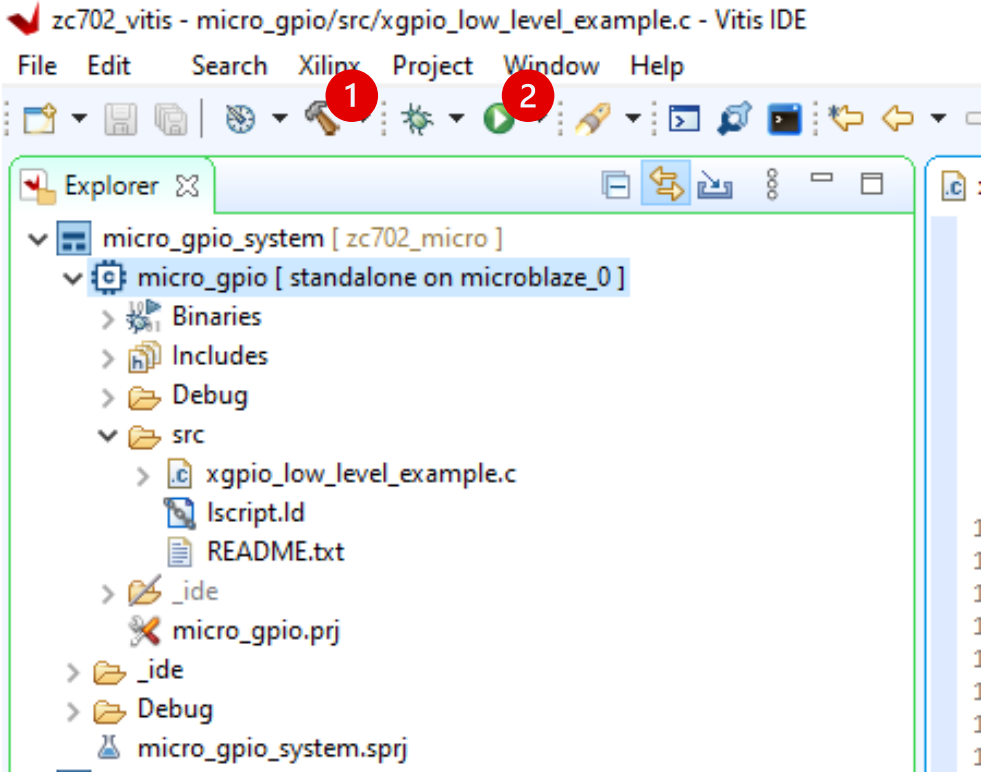
AMD XILINX

Microblze in Vitis





Microblze in Vitis



AMD
XILINX

AMD
XILINX

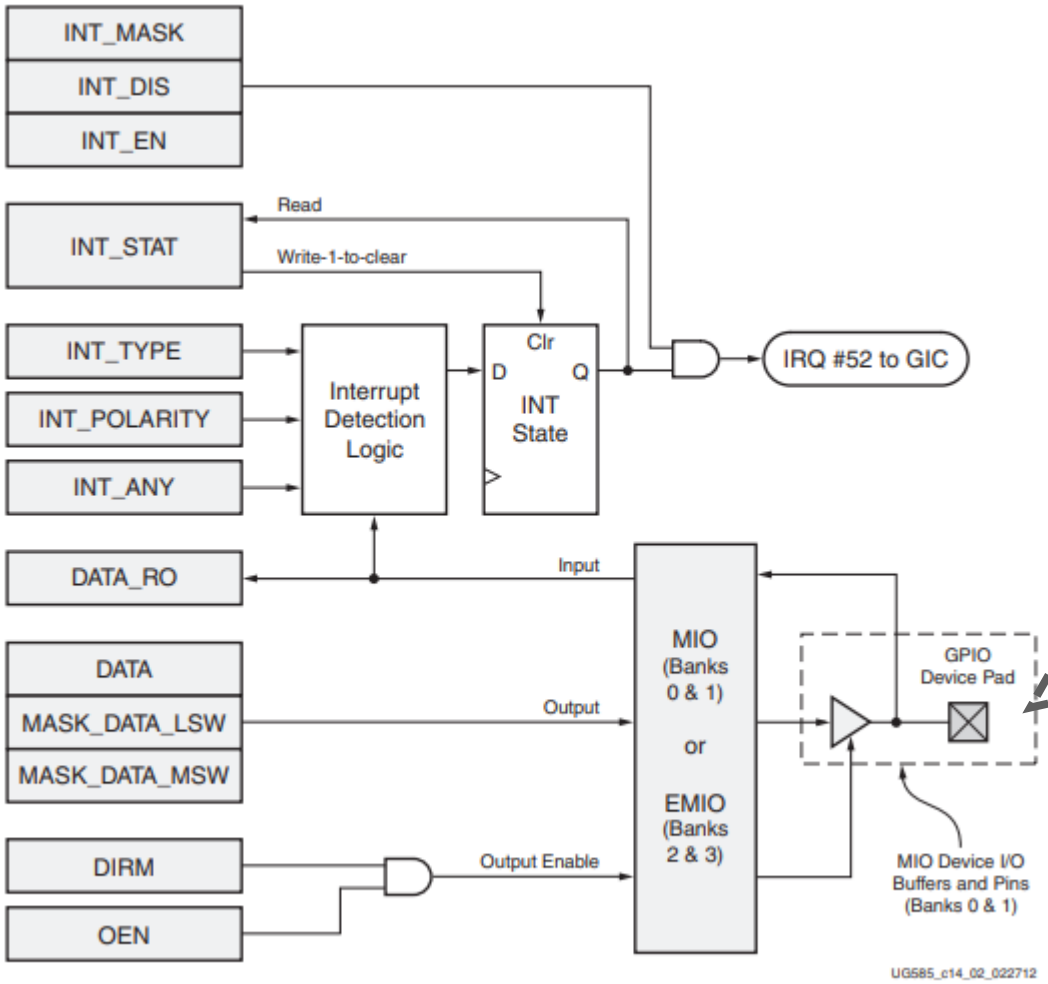
Using the PS GP Port in Zynq Devices

Course Agenda
2023

What is GPIO?

- Genral 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?



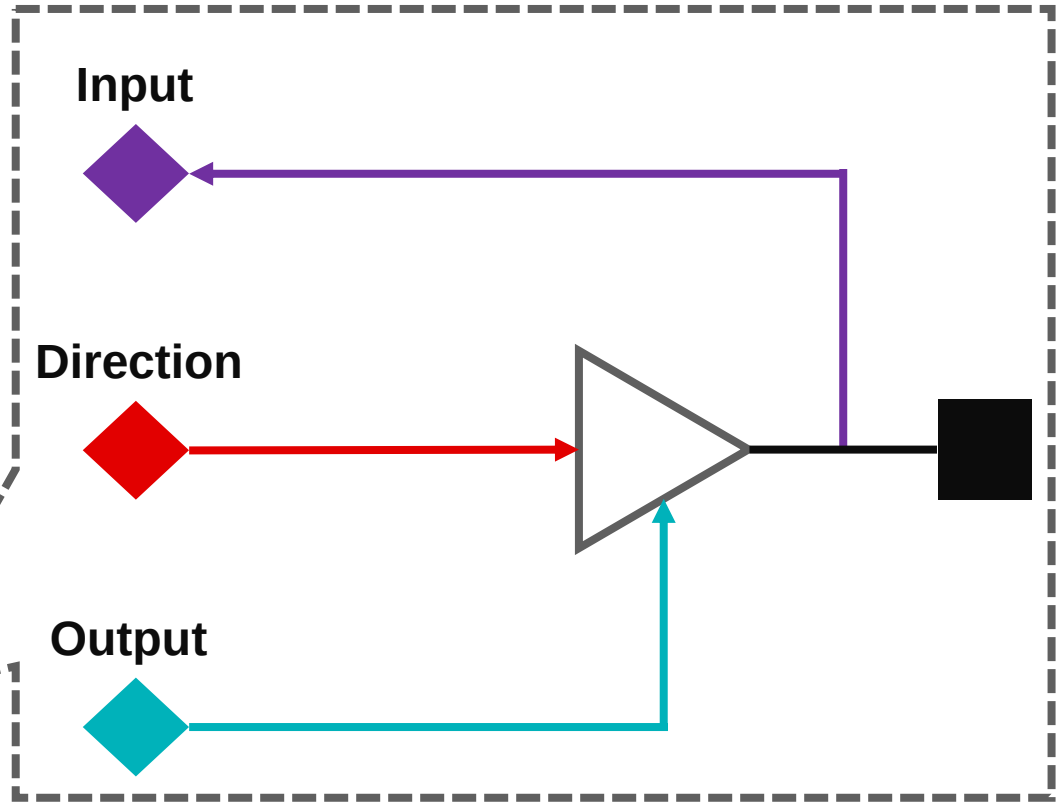
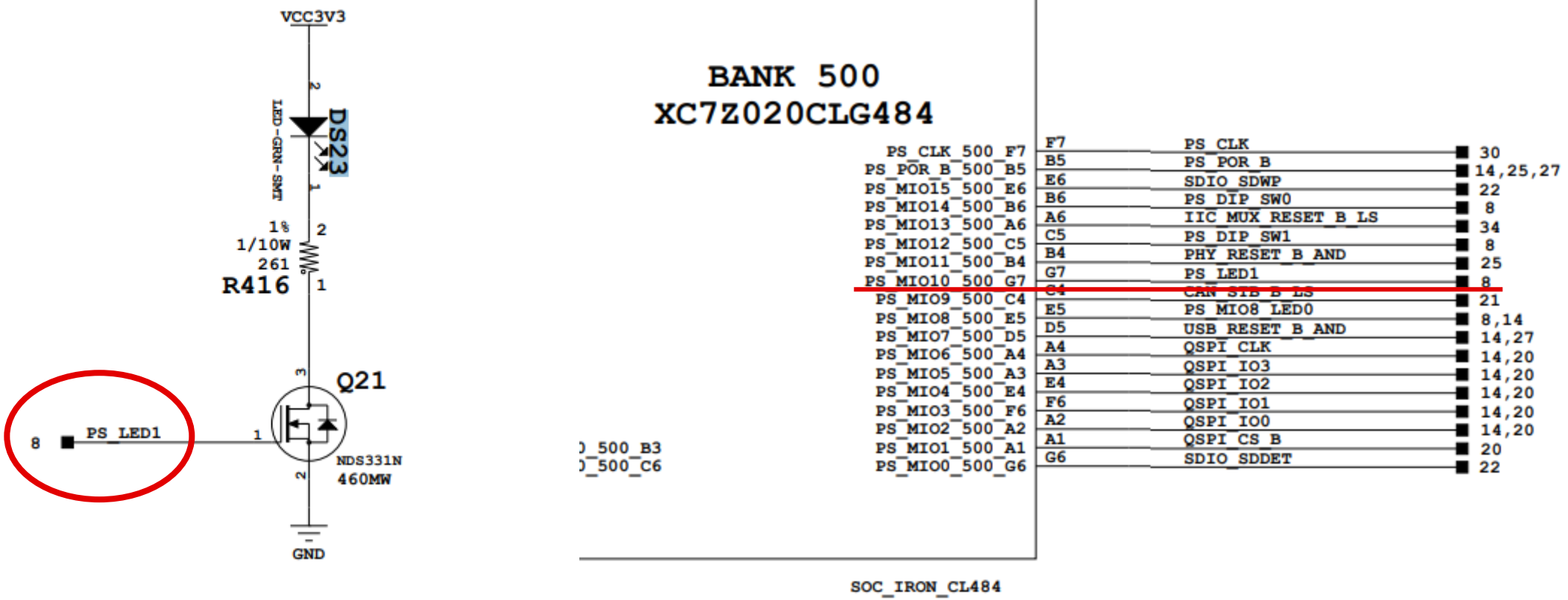


Figure 14-2: **GPIO Channel**

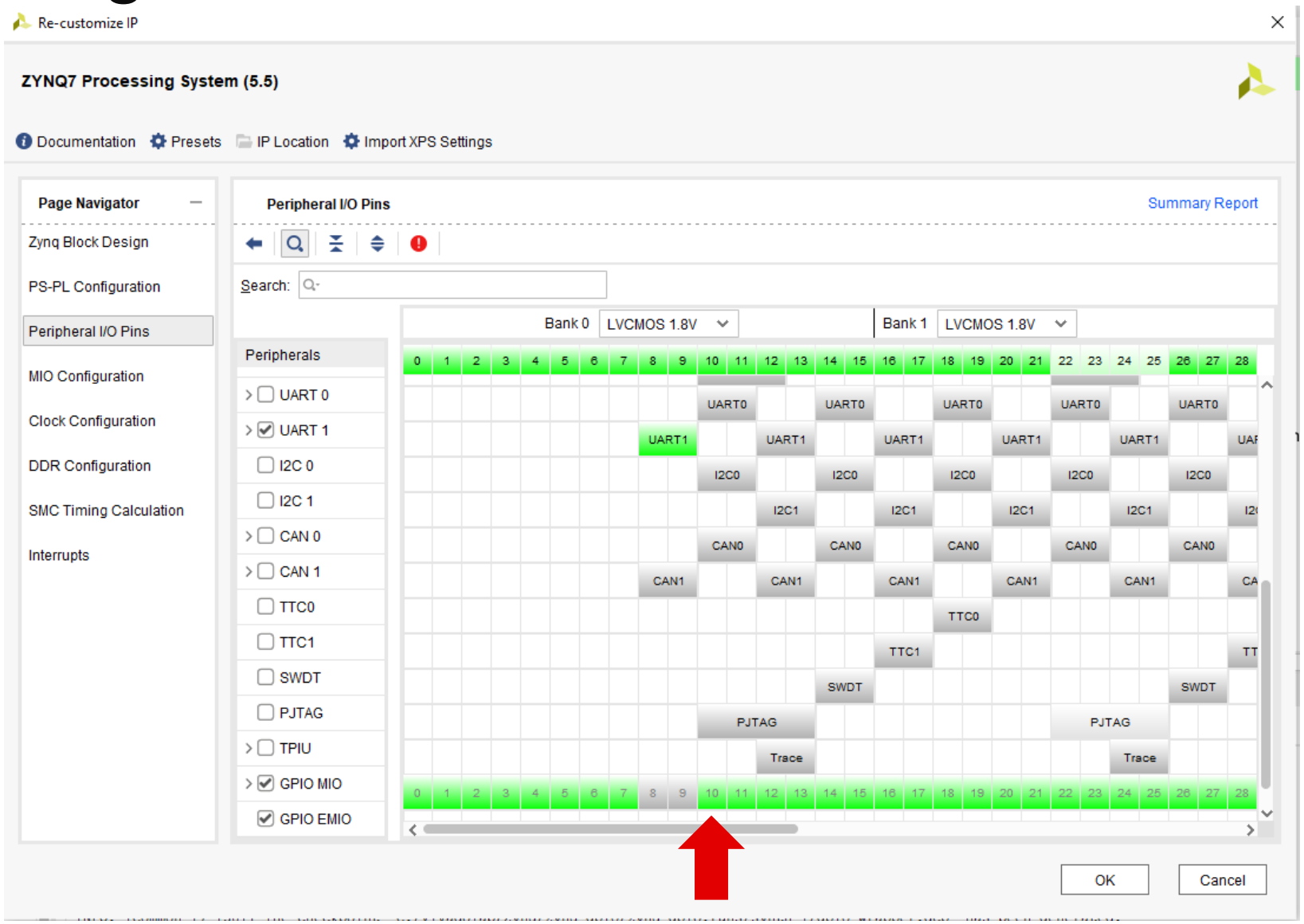
AMD XILINX

MIO Configuration



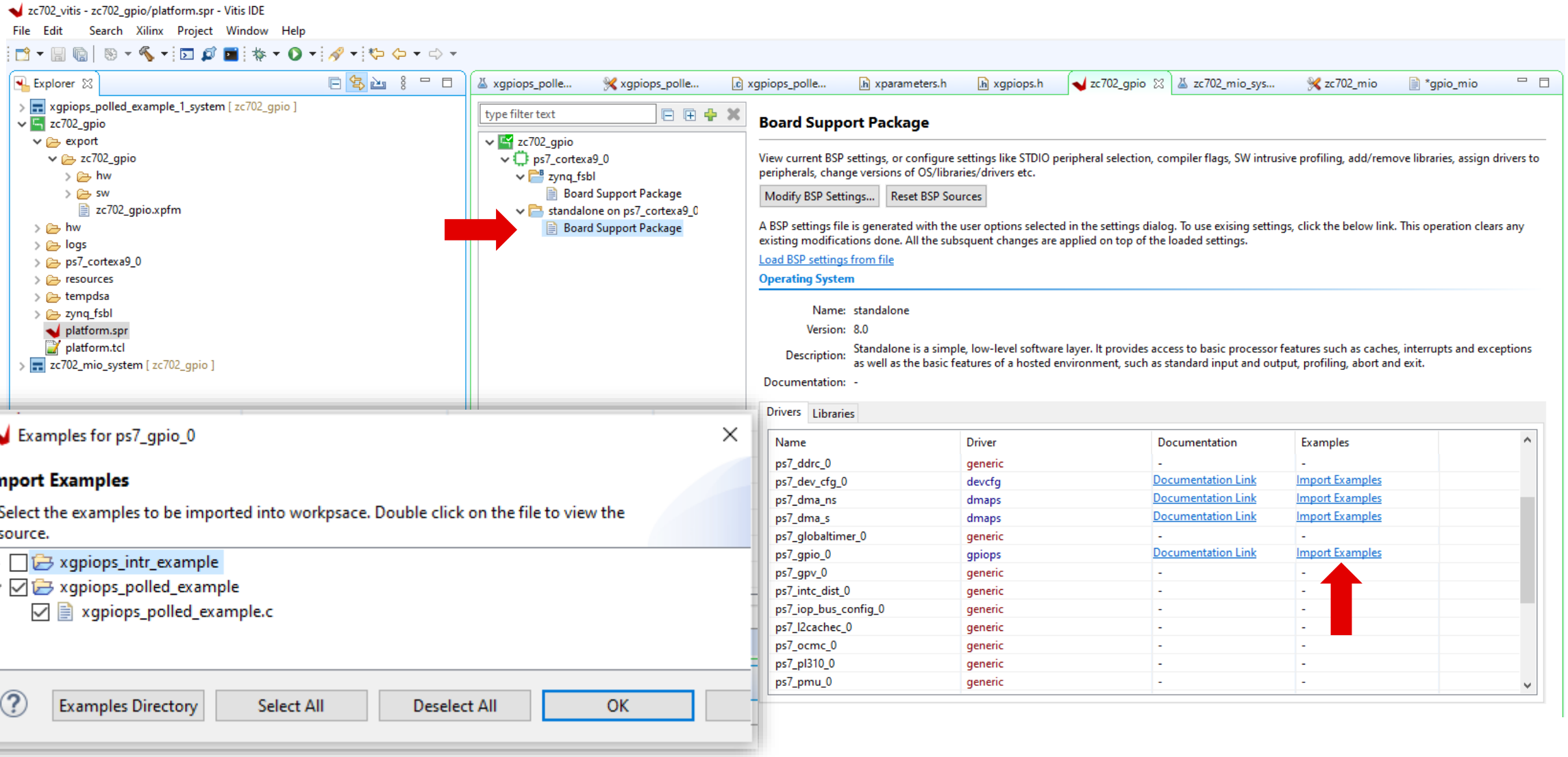
AMD XILINX

MIO Configuration



AMD XILINX

Vitis Example Design



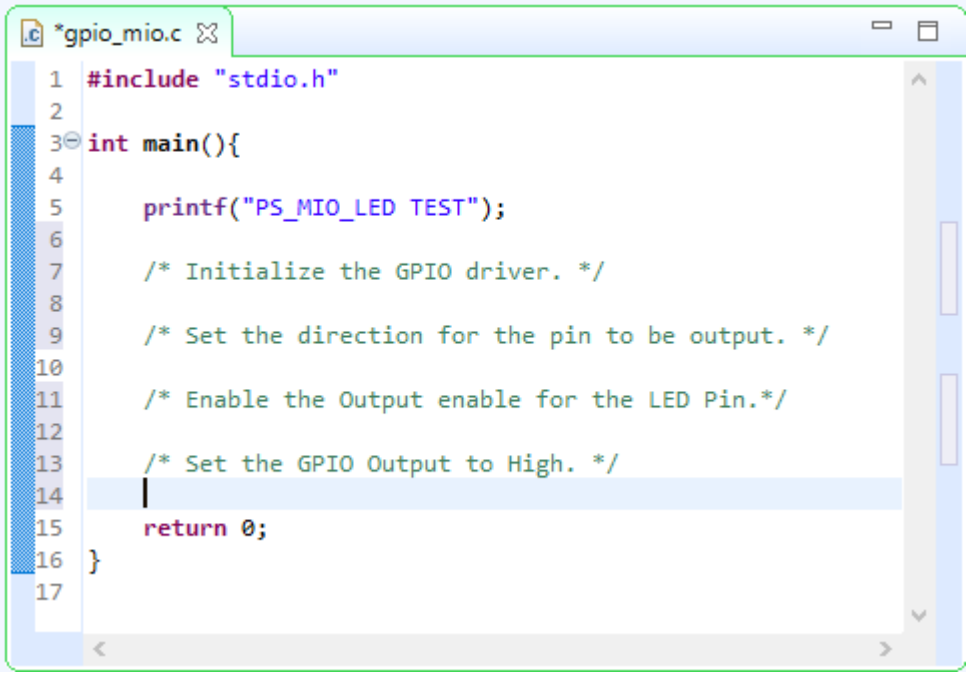
AMD XILINX

Programming Guide

Start-up Sequence

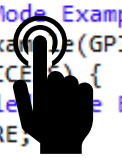
Main Example: Start-up Sequence

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.



Initialize the GPIO driver

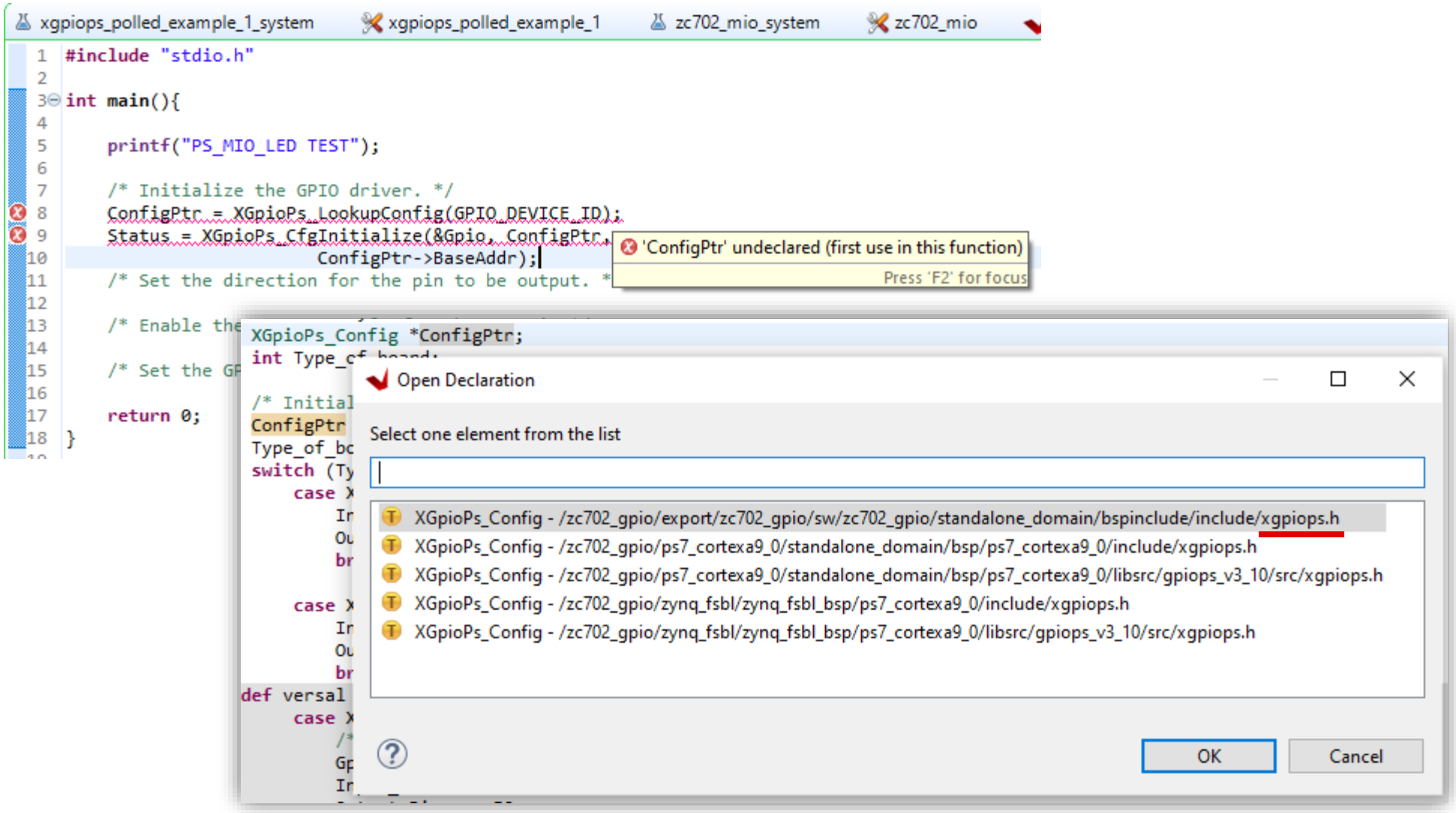
```c
int main(void)
{
    int Status;
    u32 InputData;

    printf("GPIO Polled Mode Example Test \r\n");
    Status = GpioPolledExample(GPIO_DEVICE_ID, &InputData);
    if (Status != XST_SUCCESS) {
        printf("GPIO Polled Mode Example Test Failed\r\n");
        return XST_FAILURE;
    }
    printf("Data read from GPIO Input is  0x%x \n\r", (int)InputData);
    printf("Successfully ran GPIO Polled Mode Example Test\r\n");
    return XST_SUCCESS;
}
```



Ctrl + Leftbutton

```c
int GpioPolledExample(u16 DeviceId, u32 *DataRead)
{
    int Status;
    XGpioPs_Config *ConfigPtr;
    int Type_of_board;

    /* Initialize the GPIO driver. */
    ConfigPtr = XGpioPs_LookupConfig(GPIO_DEVICE_ID);
    Type_of_board = XGetPlatform_Info();
    switch (Type_of_board) {
        case XPLAT_ZYNQ_ULTRA_MP:
            Input_Pin = 22;
            Output_Pin = 23;
            break;

        case XPLAT_ZYNQ:
            Input_Pin = 14;
            Output_Pin = 10;
            break;
#ifdef versal
        case XPLAT_VERSAL:
            /* Accessing PMC GPIO by setting field to 1 */
            Gpio.PmcGpio =  1;
            Input_Pin    = 56;
            Output_Pin   = 52;
            break;
#endif
    }

    Status = XGpioPs_CfgInitialize(&Gpio, ConfigPtr,
                    ConfigPtr->BaseAddr);
    if (Status != XST_SUCCESS) {
        return XST_FAILURE;
    }
```

Initialize the GPIO driver



Initialize the GPIO driver

xgpiops.h

```
/************************** Function Prototypes ******************************/

/* Functions in xgpiops.c */
s32 XGpioPs_CfgInitialize(XGpioPs *InstancePtr, const XGpioPs_Config *ConfigPtr,
                u32 EffectiveAddr);

/* Bank APIs in xgpiops.c */
u32 XGpioPs_Read(const XGpioPs *InstancePtr, u8 Bank);
void XGpioPs_Write(const XGpioPs *InstancePtr, u8 Bank, u32 Data);
void XGpioPs_SetDirection(const XGpioPs *InstancePtr, u8 Bank, u32 Direction);
u32 XGpioPs_GetDirection(const XGpioPs *InstancePtr, u8 Bank);
void XGpioPs_SetOutputEnable(const XGpioPs *InstancePtr, u8 Bank, u32 OpEnable);
u32 XGpioPs_GetOutputEnable(const XGpioPs *InstancePtr, u8 Bank);
#ifdef versal
void XGpioPs_GetBankPin(const XGpioPs *InstancePtr,u8 PinNumber,u8 *BankNumber, u8
#else
void XGpioPs_GetBankPin(u8 PinNumber,u8 *BankNumber, u8 *PinNumberInBank);
#endif

/* Pin APIs in xgpiops.c */
u32 XGpioPs_ReadPin(const XGpioPs *InstancePtr, u32 Pin);
void XGpioPs_WritePin(const XGpioPs *InstancePtr, u32 Pin, u32 Data);
void XGpioPs_SetDirectionPin(const XGpioPs *InstancePtr, u32 Pin, u32 Direction);
u32 XGpioPs_GetDirectionPin(const XGpioPs *InstancePtr, u32 Pin);
```

xparameters.h

```
/******************************************************************/

/* Canonical definitions for peripheral PS7_GPIO_0 */
#define XPAR_XGPIOPS_0_DEVICE_ID XPAR_PS7_GPIO_0_DEVICE_ID
#define XPAR_XGPIOPS_0_BASEADDR 0xE000A000
#define XPAR_XGPIOPS_0_HIGHADDR 0xE000AFFF


/******************************************************************/

/* Definitions for driver IICPS */
#define XPAR_XIICPS_NUM_INSTANCES 1

/* Definitions for peripheral PS7_I2C_0 */
#define XPAR_PS7_I2C_0_DEVICE_ID 0
#define XPAR_PS7_I2C_0_BASEADDR 0xE0004000
#define XPAR_PS7_I2C_0_HIGHADDR 0xE0004FFF
#define XPAR_PS7_I2C_0_I2C_CLK_FREQ_HZ 111111115


/******************************************************************/

/* Canonical definitions for peripheral PS7_I2C_0 */
#define XPAR_XIICPS_0_DEVICE_ID XPAR_PS7_I2C_0_DEVICE_ID
#define XPAR_XIICPS_0_BASEADDR 0xE0004000
#define XPAR_XIICPS_0_HIGHADDR 0xE0004FFF
#define XPAR_XIICPS_0_I2C_CLK_FREQ_HZ 111111115


/******************************************************************/

/* Definition for input Clock */
```


AMD XILINX

Initialize the GPIO driver

zc702_mio_sy... | zc702_mio | zc702_gpio | xgpiops_poll... | *gpio_mio.c | xgpiops.c

```
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
12     printf("PS_MIO_LED TEST");
13
14     /* Initialize the GPIO driver. */
15     /* This function looks for the device configuration based on the unique device ID. */
16     ConfigPtr = XGpioPs_LookupConfig(GPIO_DEVICE_ID);
17     /* This function initializes a XGpioPs instance/driver. */
18     XGpioPs_CfgInitialize(&Gpio, ConfigPtr , ConfigPtr->BaseAddr);
19
20     /* Set the direction for the pin to be output. */
21
22     /* Enable the Output enable for the LED Pin.*/
23
24     /* Set the GPIO Output to High. */
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

1. **Set the direction as output:** Write `0x0000_0400` to the gpio.DIRM_0 register.
2. **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

1. **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.

1. **Read the gpio.DATA_0 register:** Read gpio.DATA_0 register to the reg_val variable.
2. **Modify the value:** Set reg_val [10] =`1`.
3. **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);
```

xgpiops_poll... | xgpiops_poll... | zc702_mio_sy... | xgpiops_poll... | gpio_mio.c | xgpiops.c

```
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 }
```

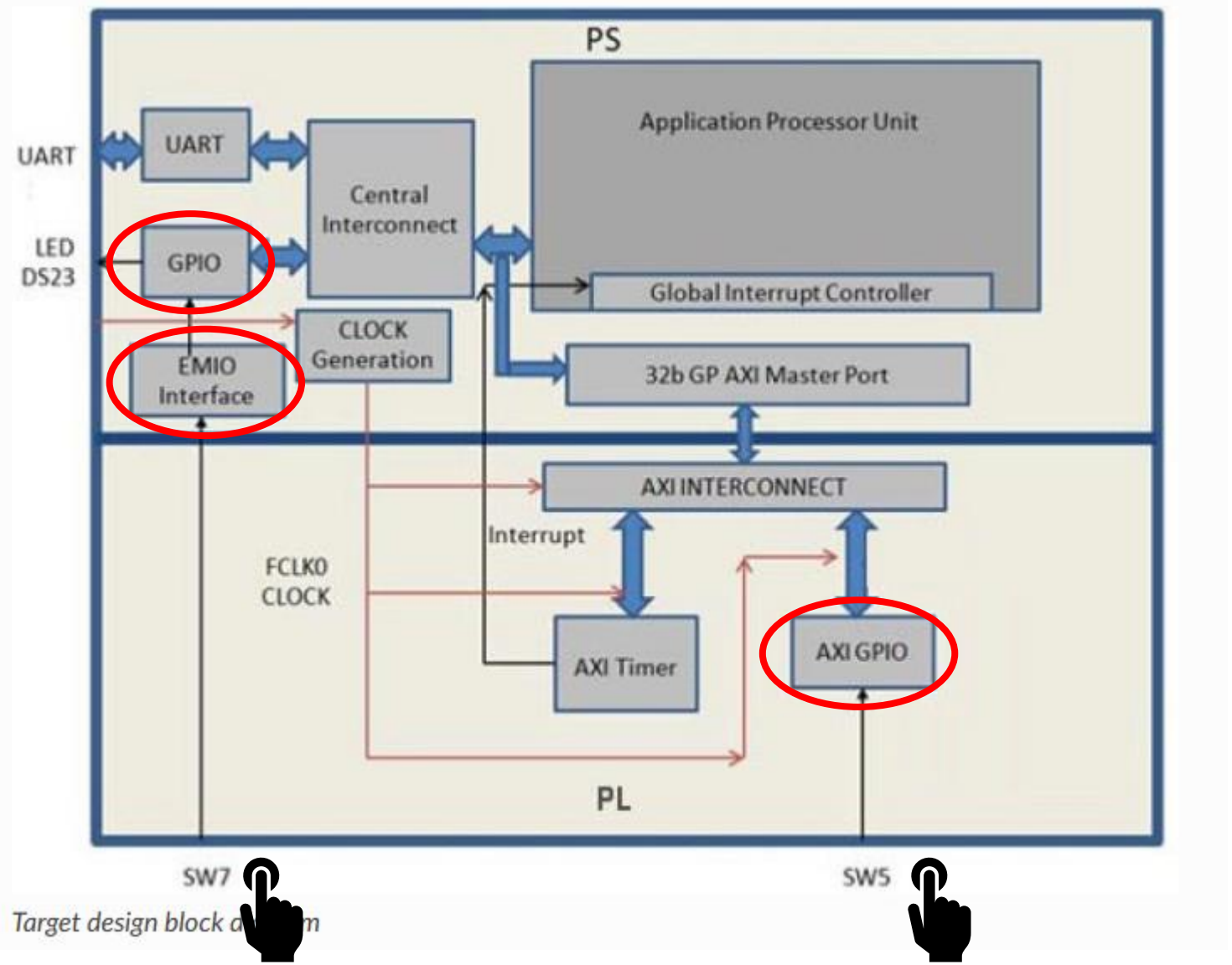

AMD XILINX

AMD
XILINX

Using the PL GP Port in Zynq Devices

Course Agenda
2023

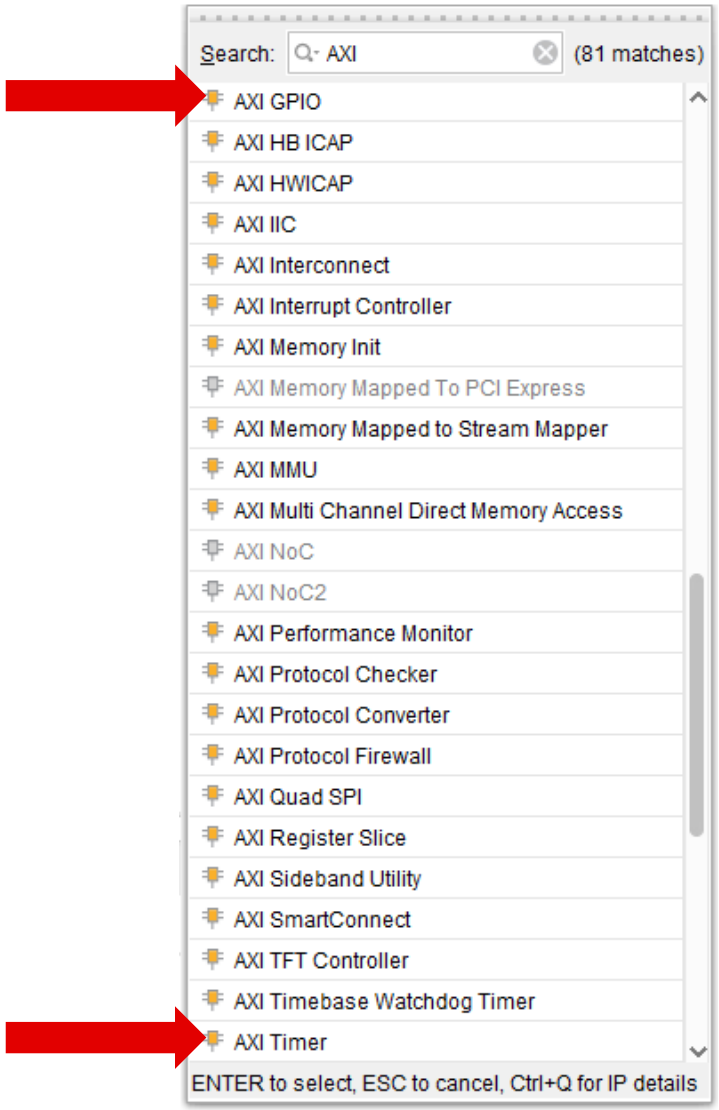
Using the GP Port in Zynq Devices

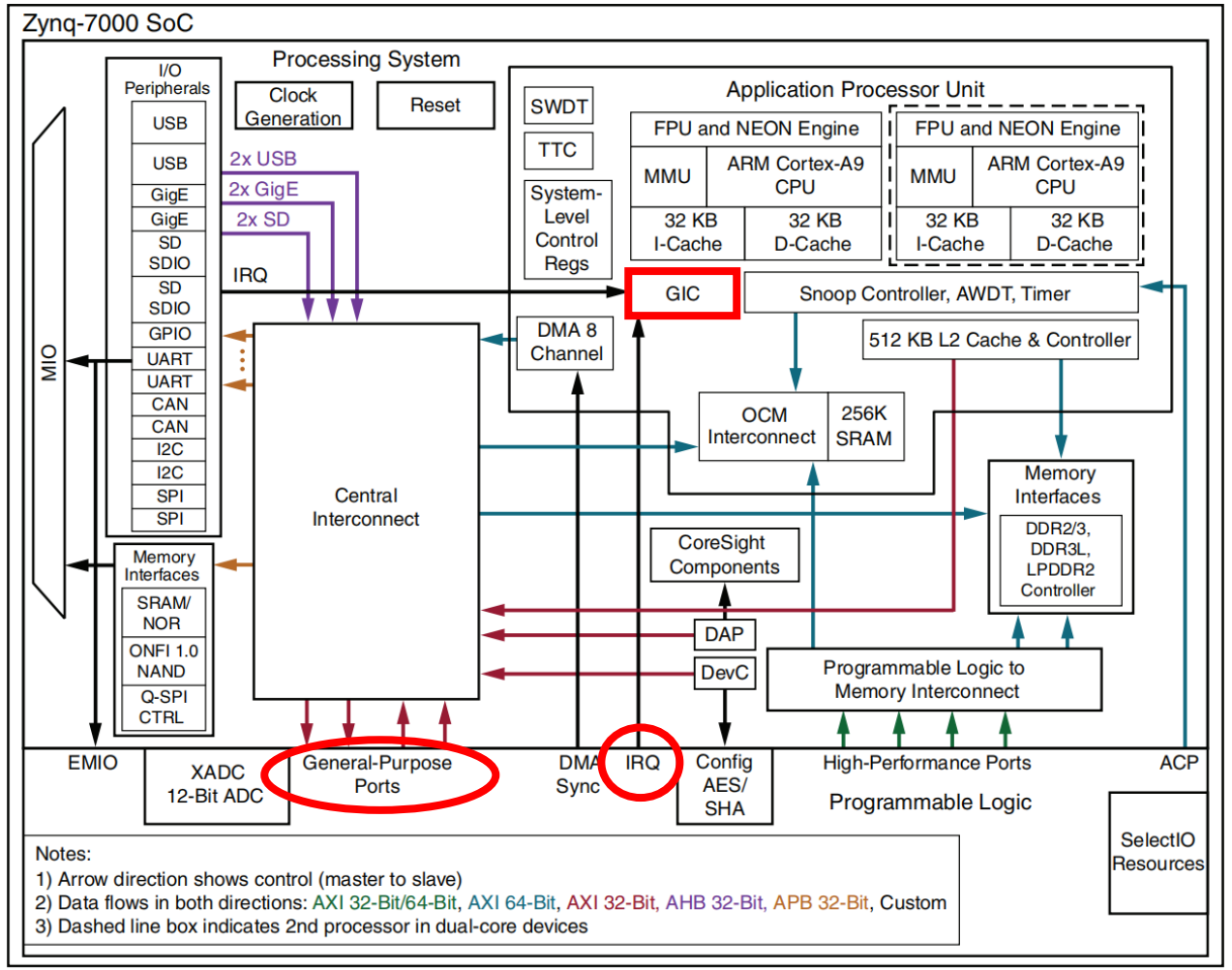


Target design block d m

AMD XILINX

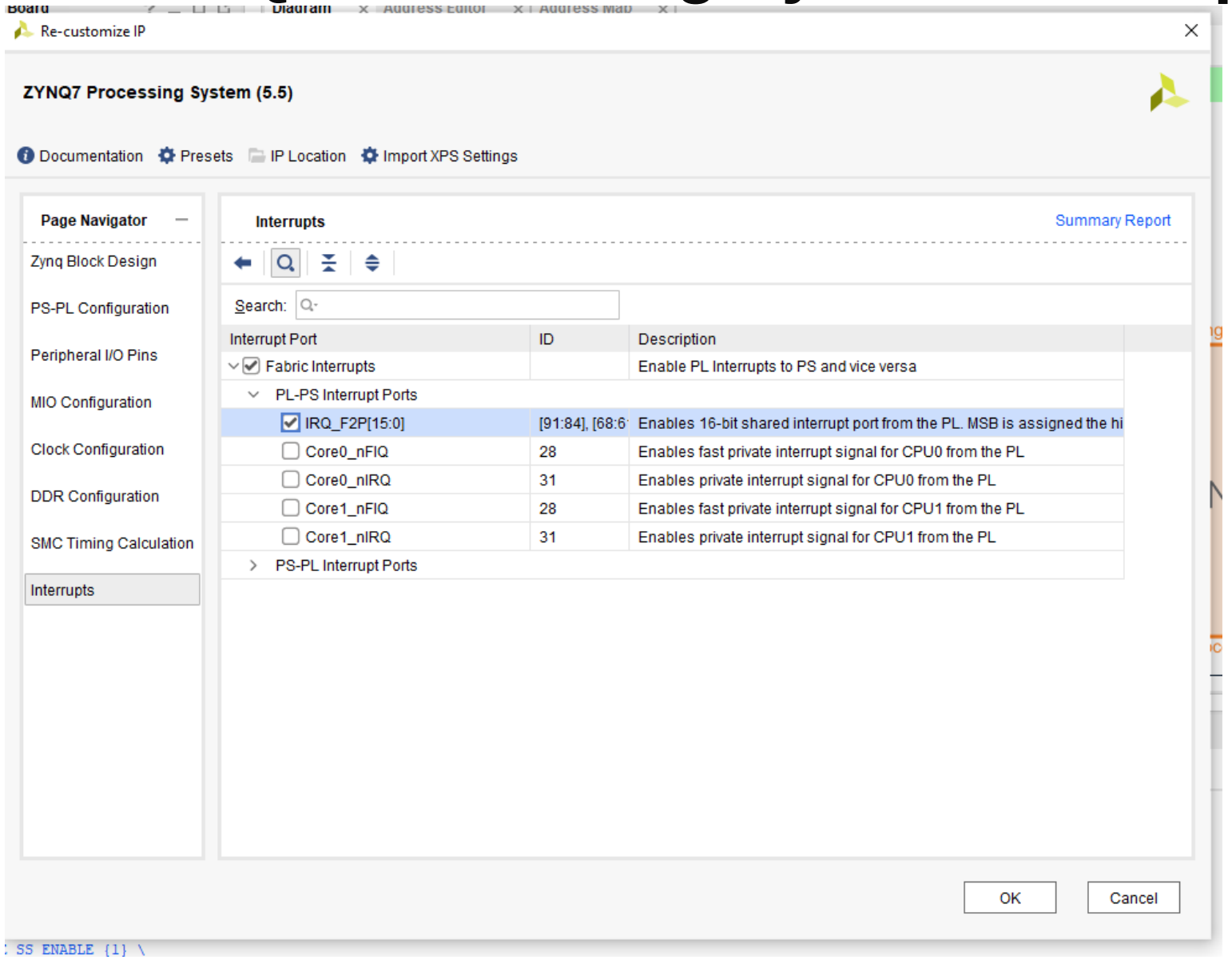
AXI GPIO & AXI Timer





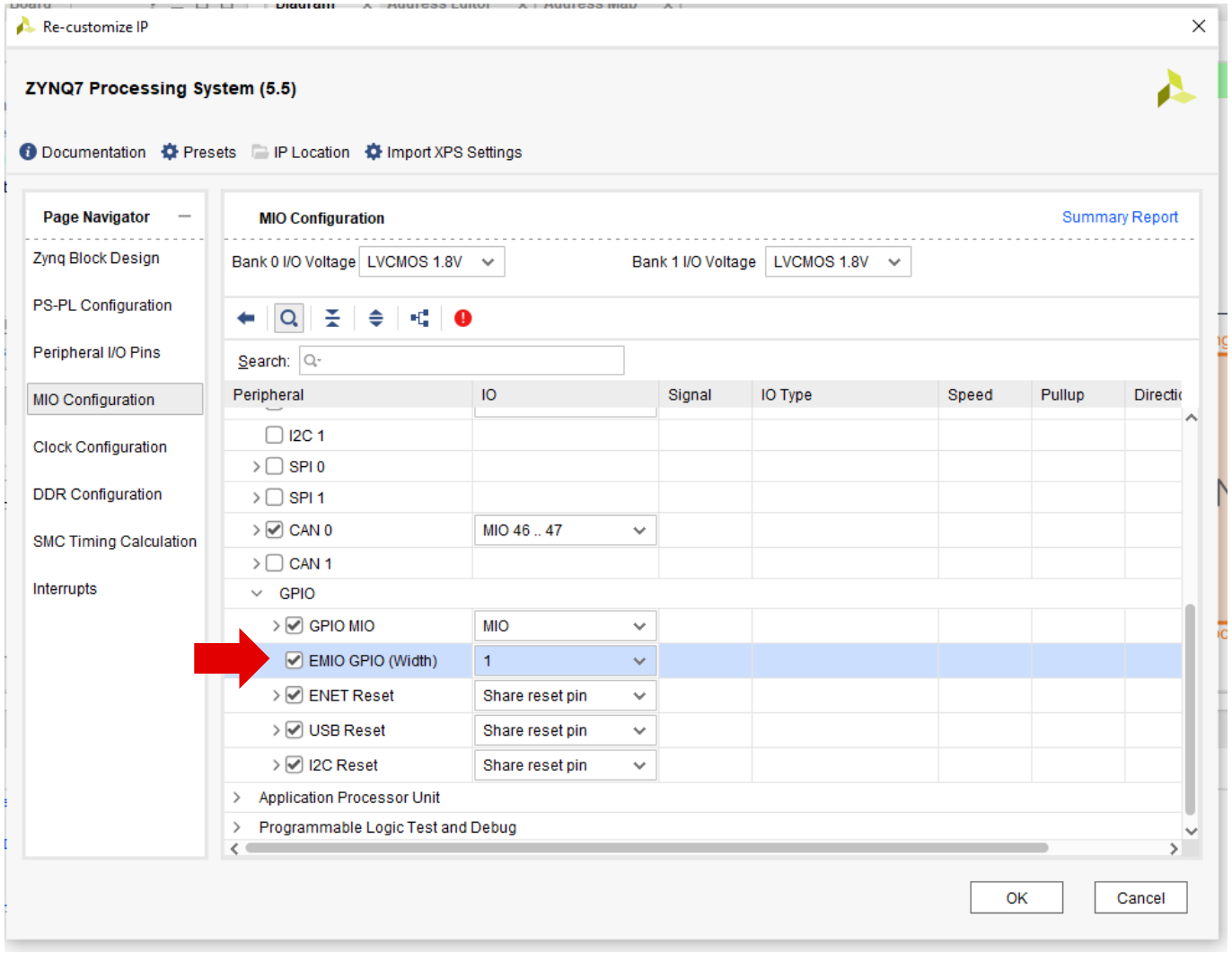
AMD XILINX

Enable the ZYNQ7 Processing System interrupt



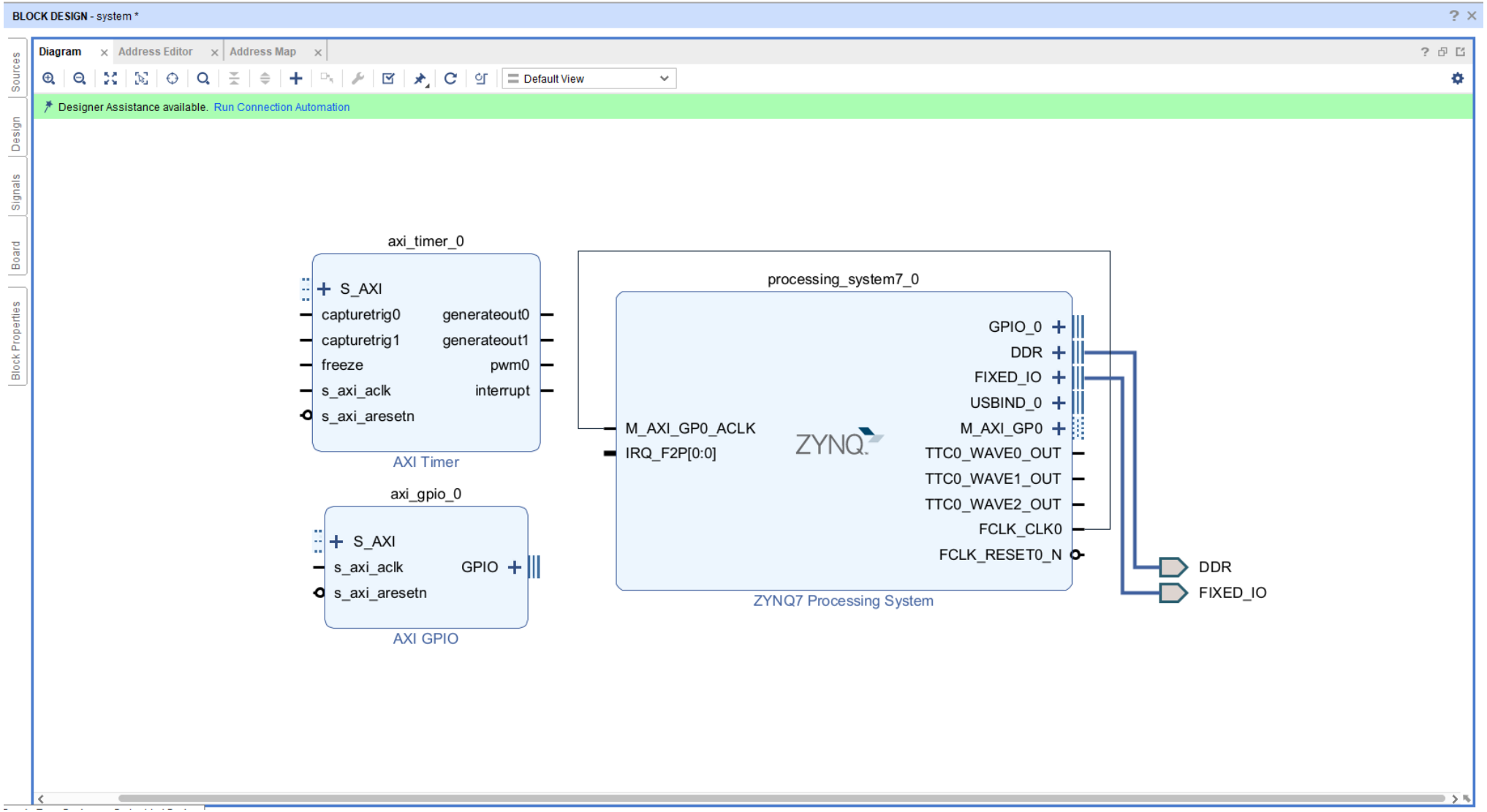
AMD XILINX

Enable the ZYNQ7 Processing System EMIO GPIO



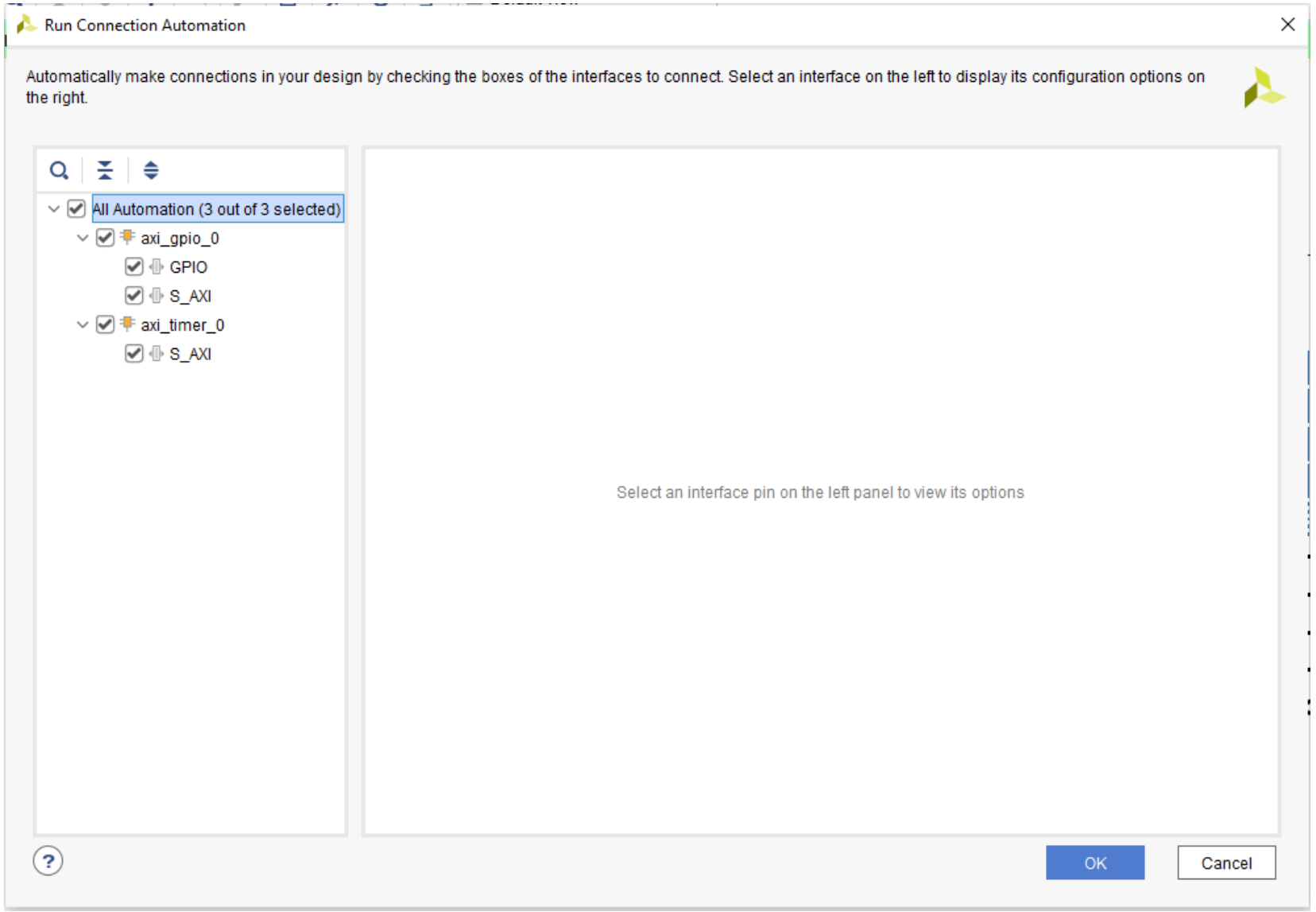
AMD XILINX

Connect the PL IPs



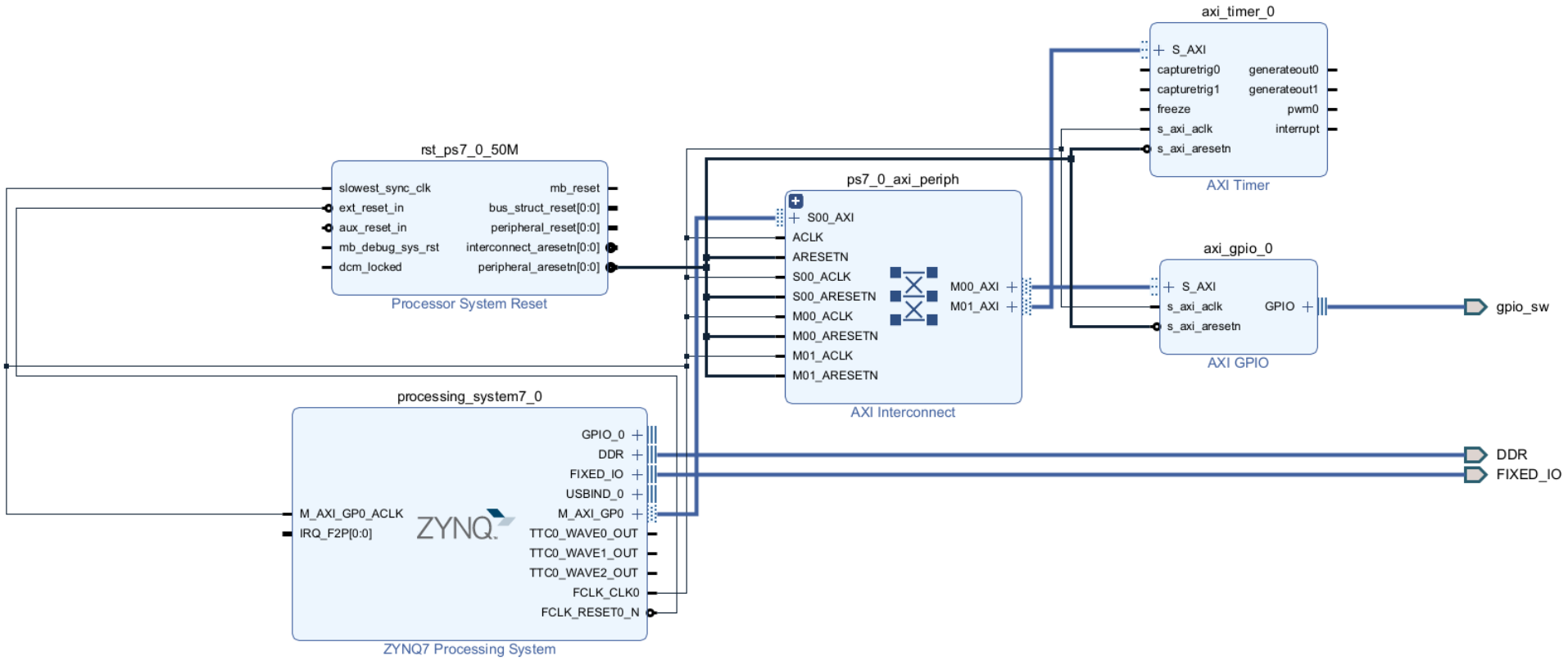
AMD XILINX

Connect the PL IPs



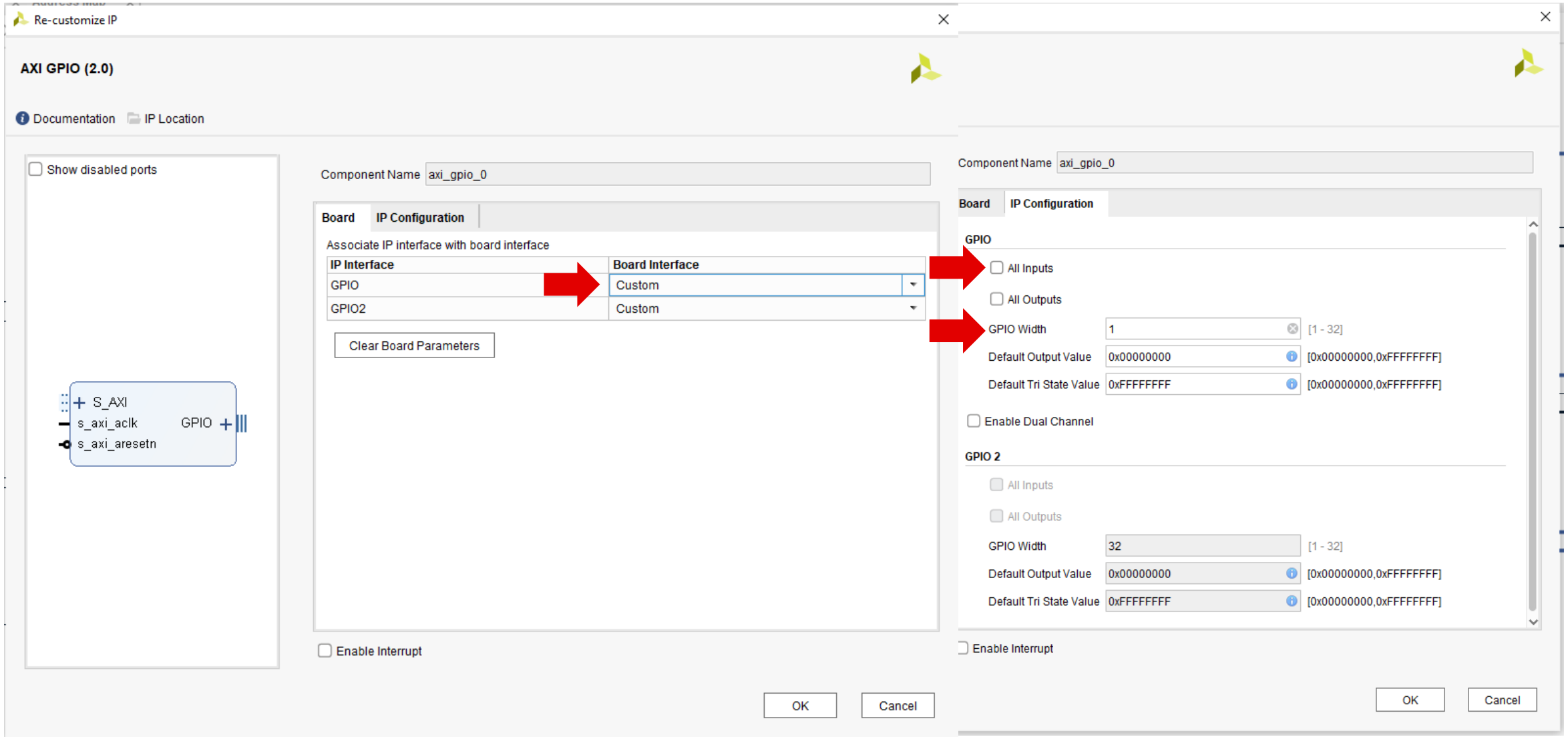
AMD XILINX

Connect the PL IPs



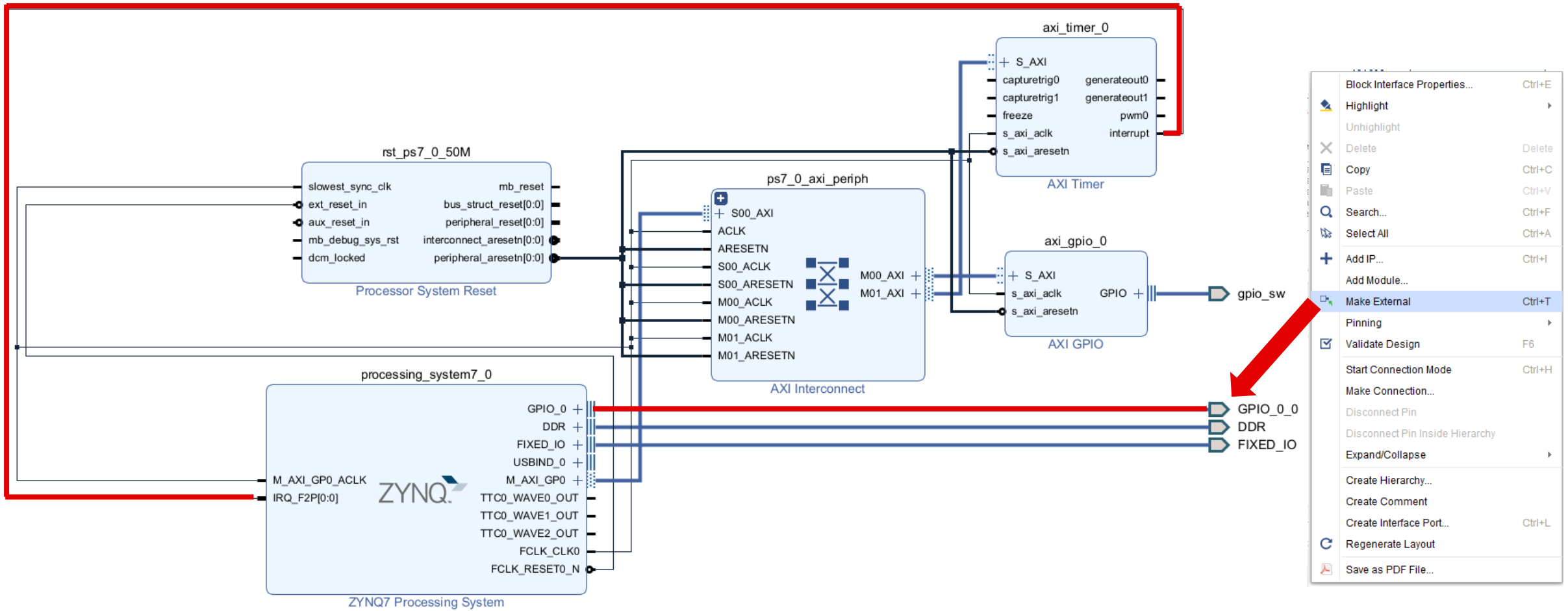
AMD XILINX

Customize the AXI GPIO IP block



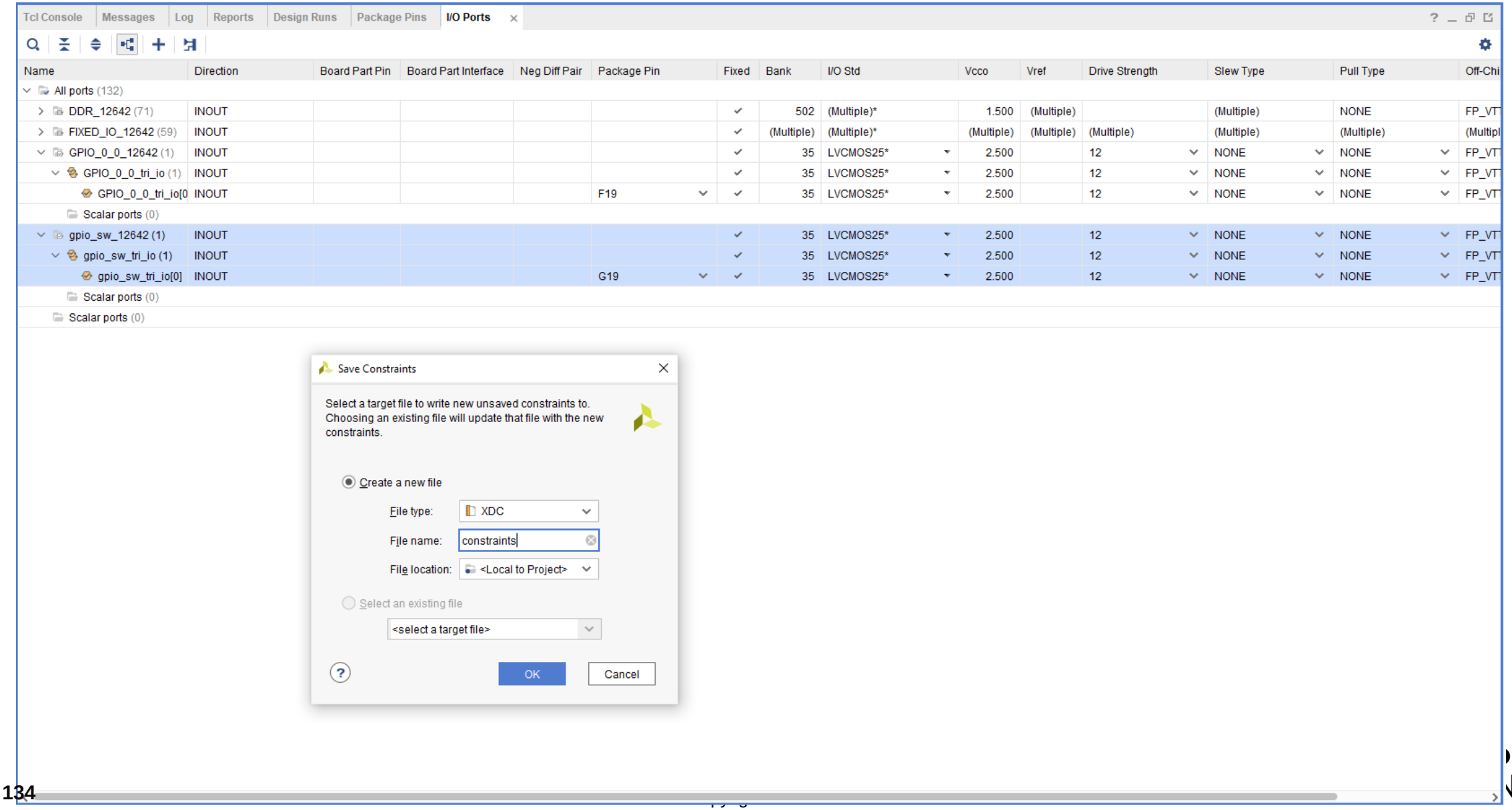
AMD XILINX

Connect interrupt signals and Make GPIO output

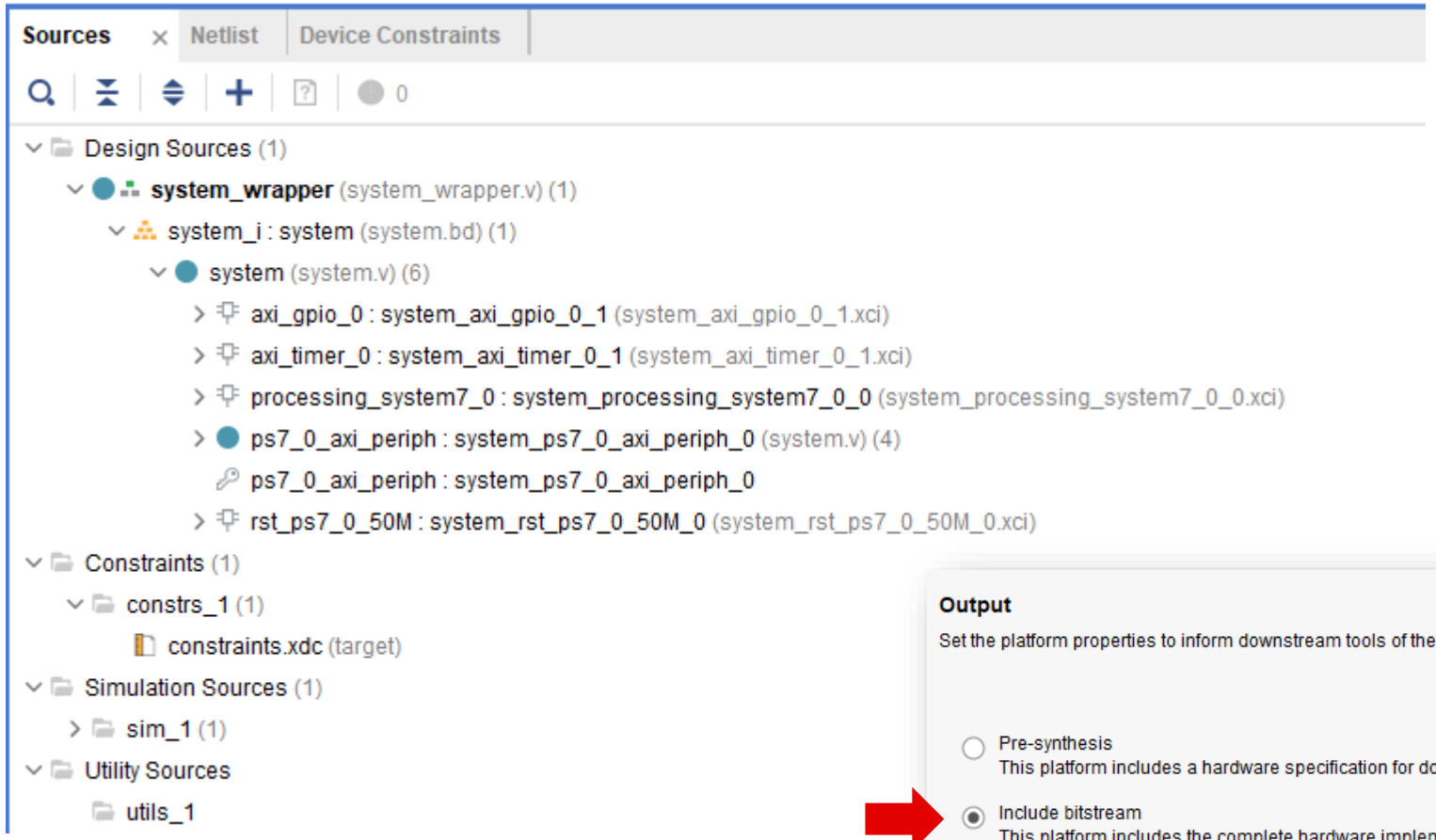


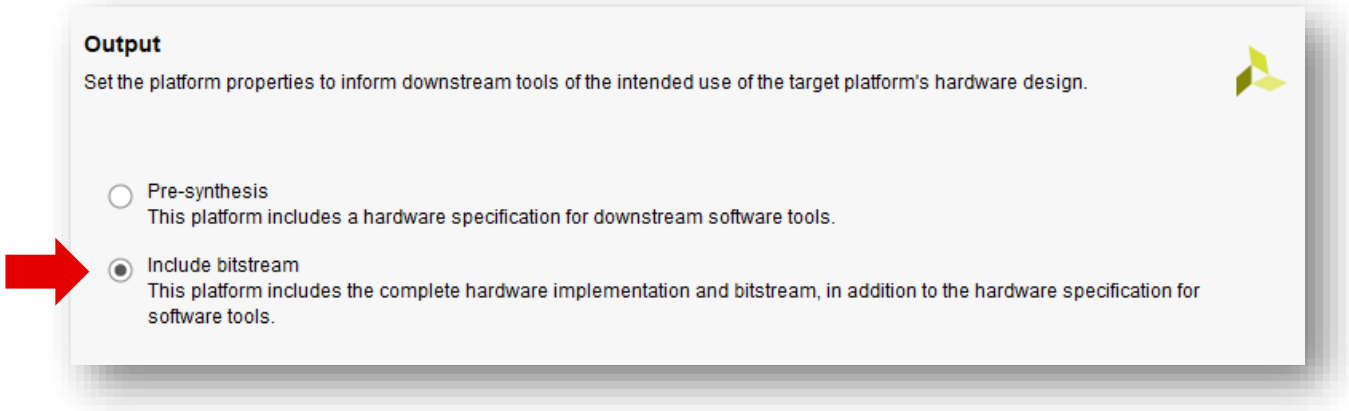
AMD XILINX

Assigning Location Constraints to External Pins



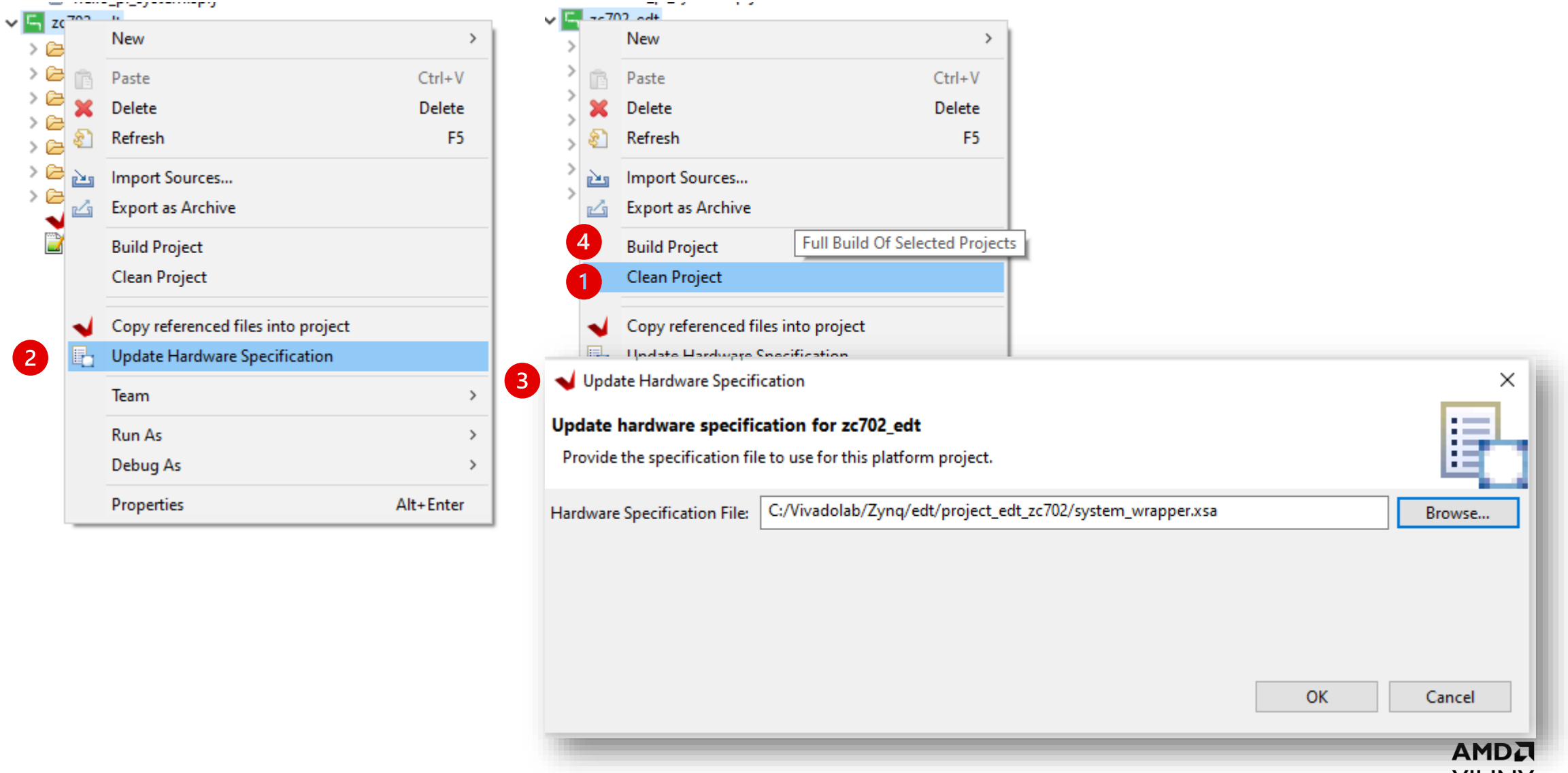
Export Hardware Platform





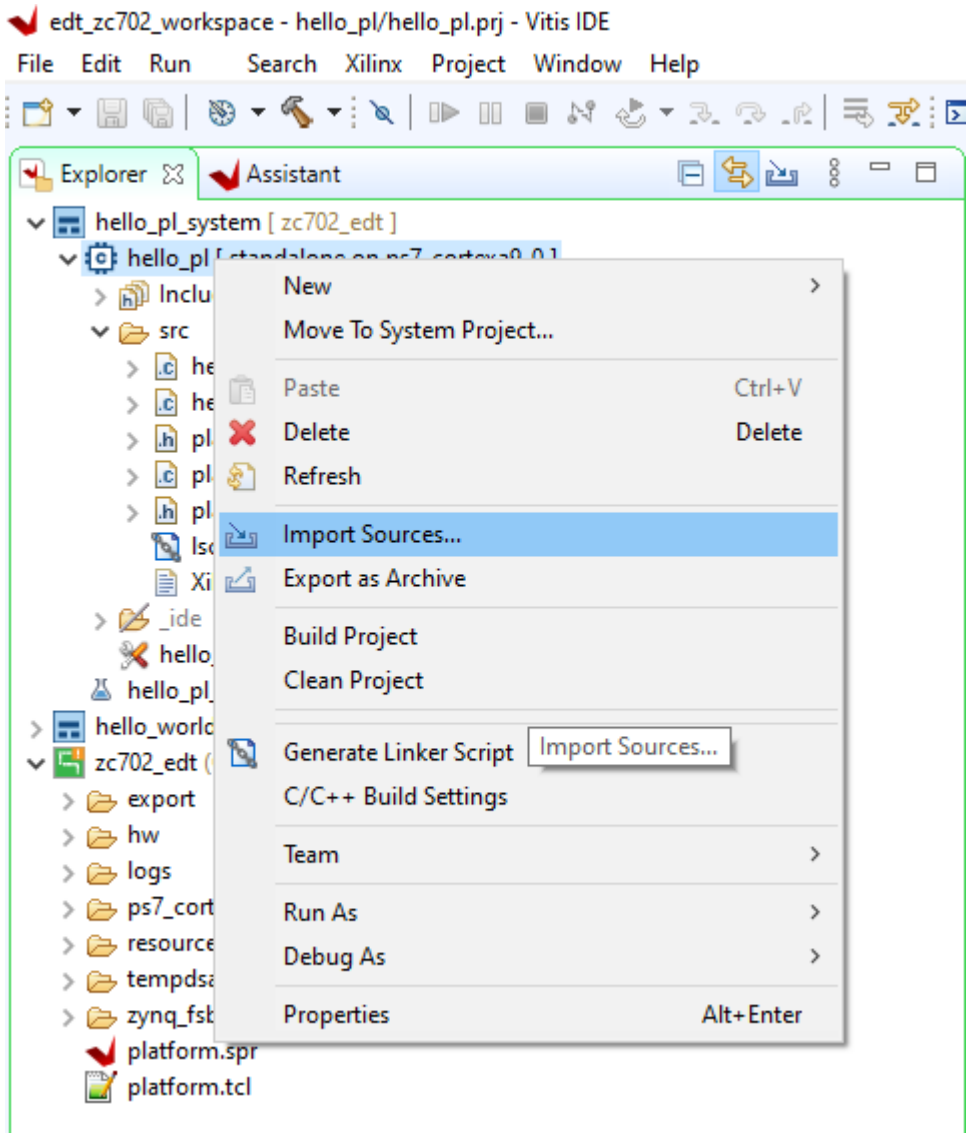
AMD XILINX

Updating Hardware in the Vitis Software Platform

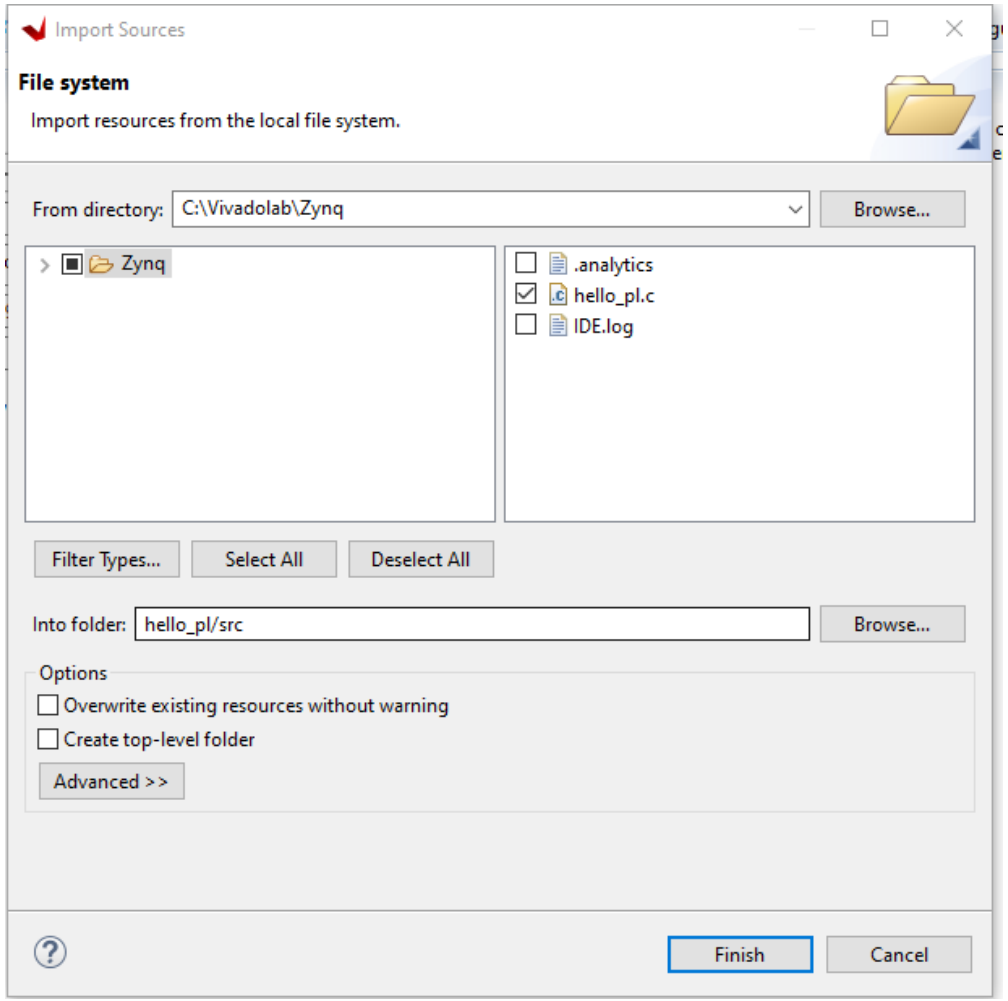


AMD XILINX

Testing the PL IP with Prepared Software



hello_pl.c



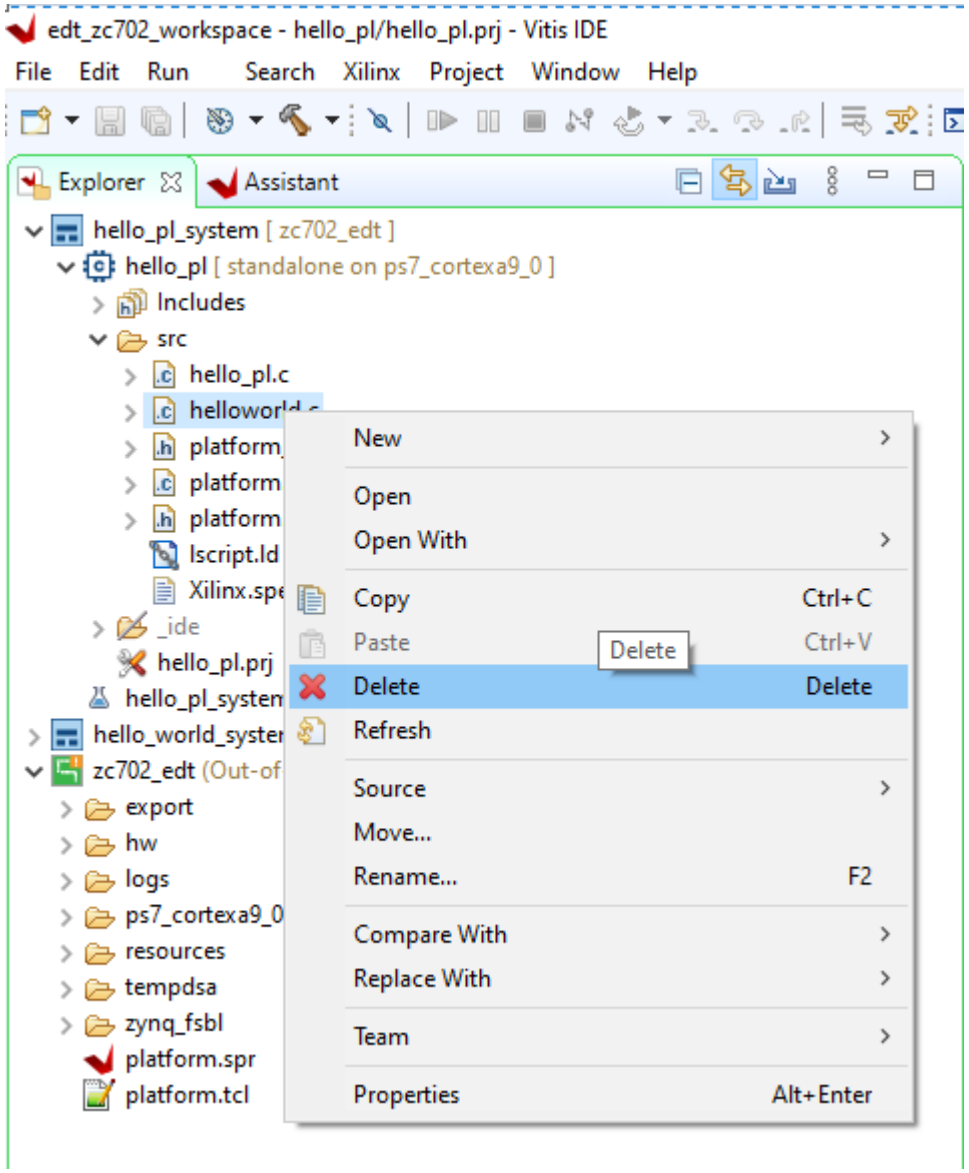
AMD XILINX

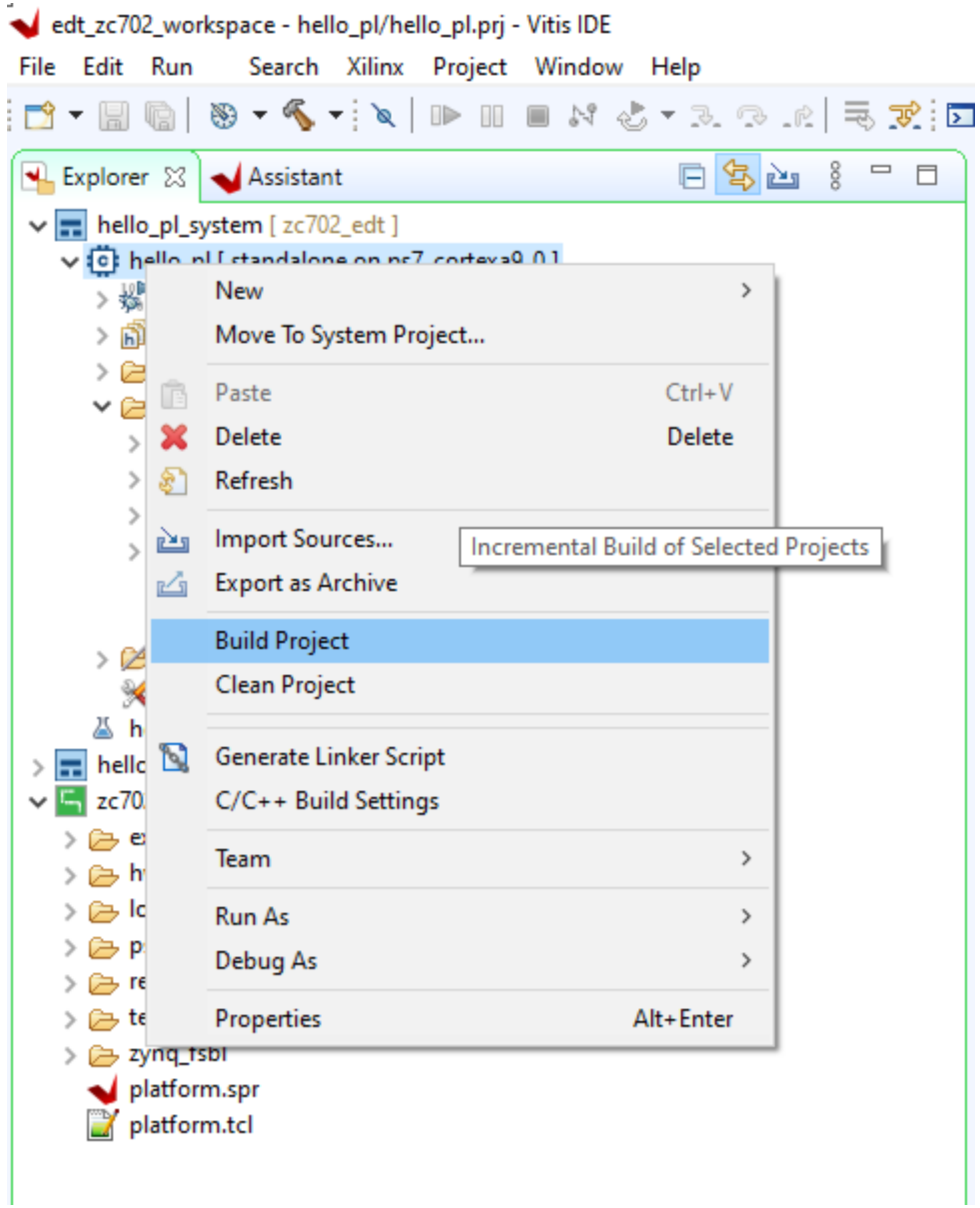
Testing the PL IP with Prepared Software

xgpiops_polled_exa... | xgpiops_polled_exa... | zc702_mio_system | zc702_mio | zc702_gpio

```
print("##### Application Starts #####\n\r");
print("\r\n");
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//Step-1 :AXI GPIO Initialization
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
xStatus = XGpio_Initialize(&GPIOInstance_Ptr,XPAR_AXI_GPIO_0_DEVICE_ID);
if(XST_SUCCESS != xStatus)
print("GPIO INIT FAILED\n\r");
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//Step-2 :AXI GPIO Set the Direction
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
XGpio_SetDataDirection(&GPIOInstance_Ptr, 1,1);
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//Step-3 :AXI Timer Initialization
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
xStatus = XTmrCtr_Initialize(&TimerInstancePtr,XPAR_AXI_TIMER_0_DEVICE_ID);
if(XST_SUCCESS != xStatus)
print("TIMER INIT FAILED \n\r");
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//Step-4 :Set Timer Handler
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
XTmrCtr_SetHandler(&TimerInstancePtr,
Timer_InterruptHandler,
&TimerInstancePtr);
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//Step-5 :Setting timer Reset Value
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
XTmrCtr_SetResetValue(&TimerInstancePtr,
0, //Change with generic value
0xf0000000);
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//Step-6 :Setting timer Option (Interrupt Mode And Auto Reload )
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
XTmrCtr_SetOptions(&TimerInstancePtr,
XPAR_AXI_TIMER_0_DEVICE_ID,
(XTC_INT_MODE_OPTION | XTC_AUTO_RELOAD_OPTION ));
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//Step-7 :PS GPIO Intialization
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
GpioConfigPtr = XGpioPs_LookupConfig(XPAR_PS7_GPIO_0_DEVICE_ID);
if(GpioConfigPtr == NULL)
return XST_FAILURE;
xStatus = XGpioPs_CfgInitialize(&psGpioInstancePtr,
```

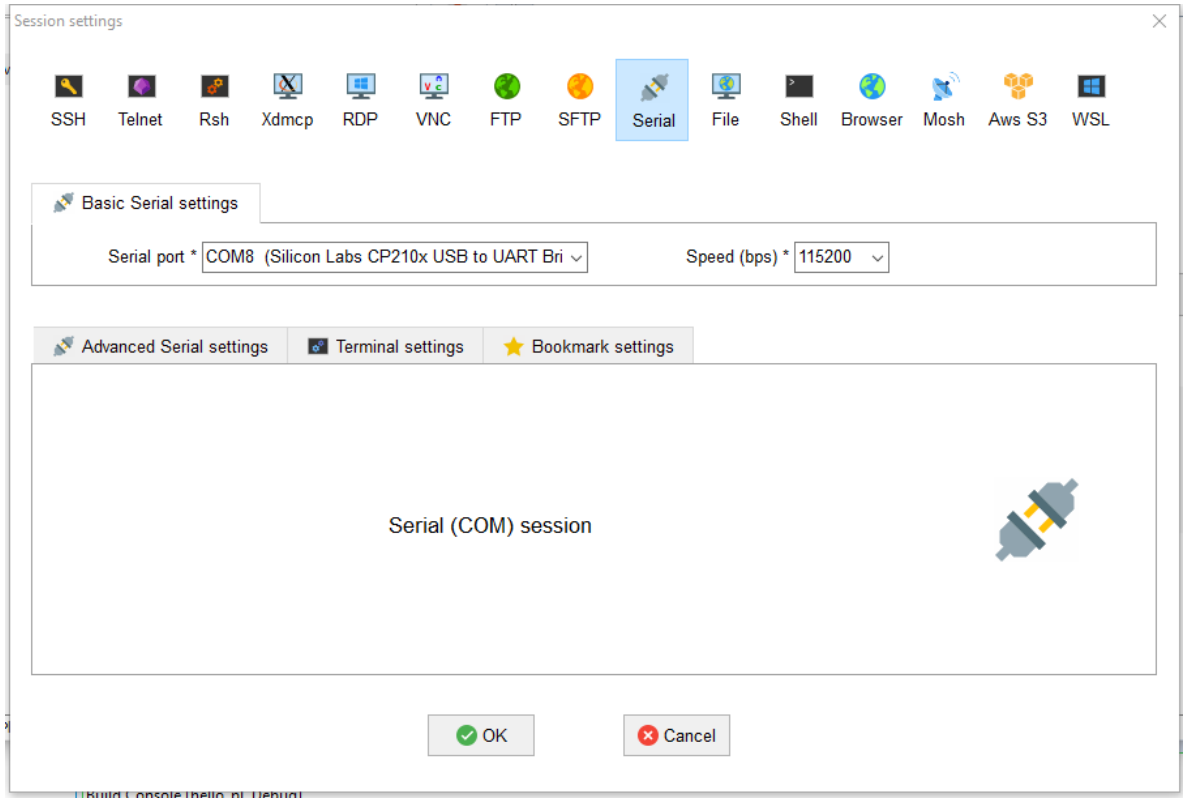
Testing the PL IP with Prepared Software





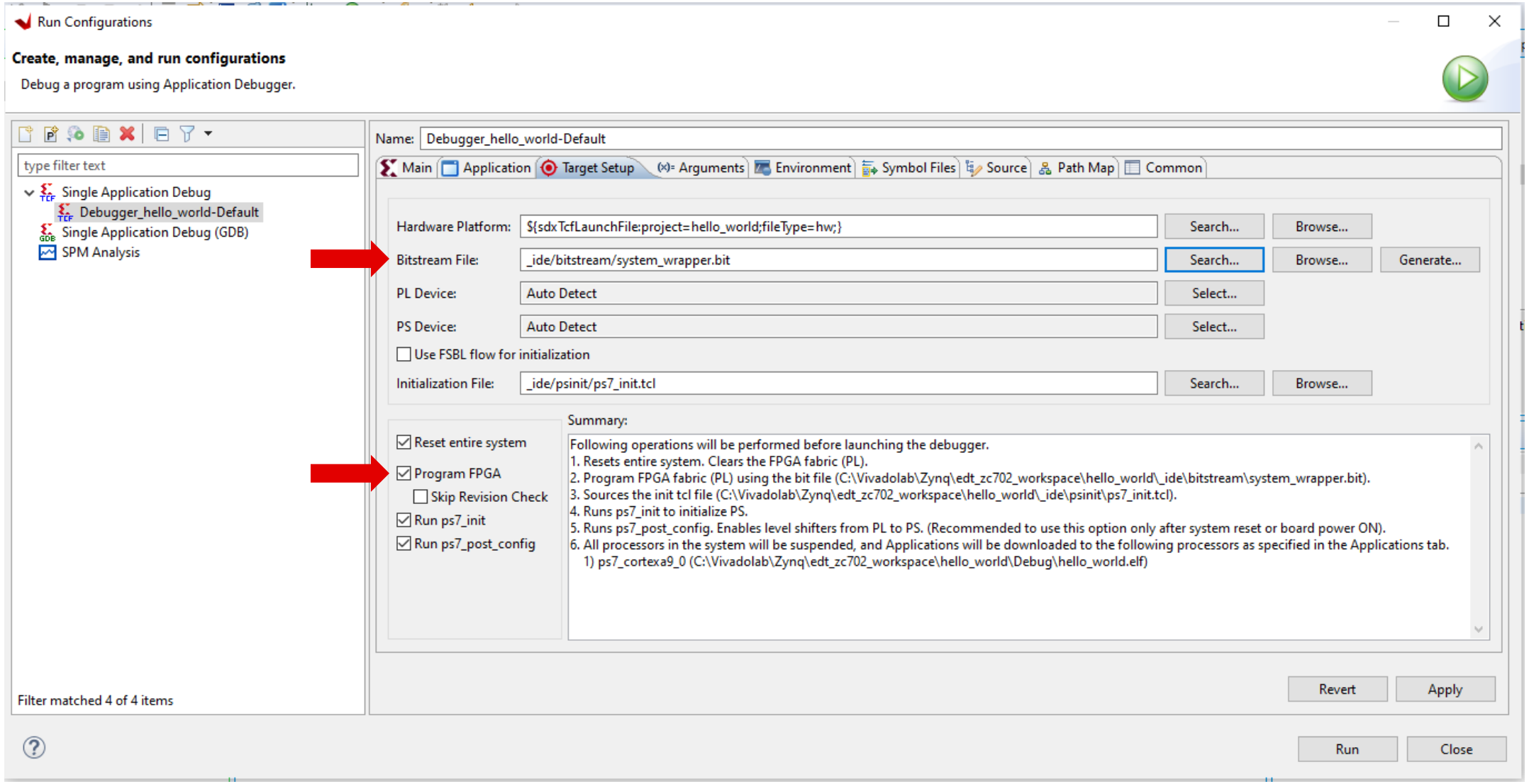
AMD XILINX

Testing the PL IP with Prepared Software



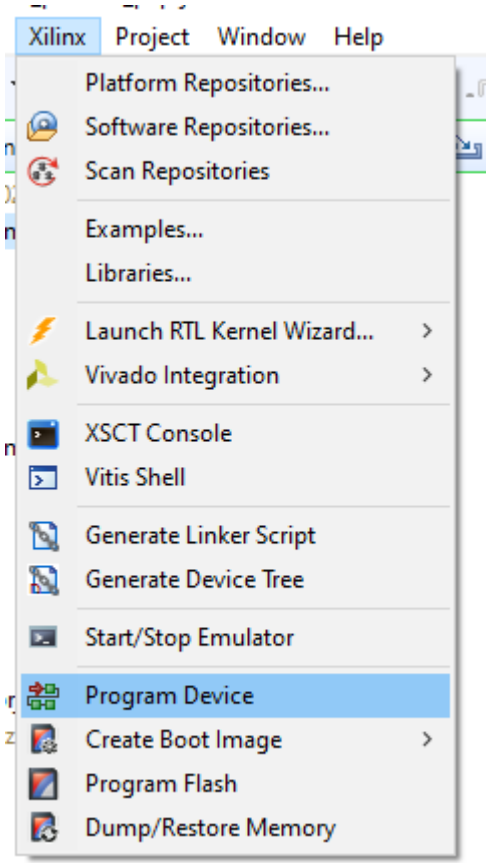
AMD XILINX

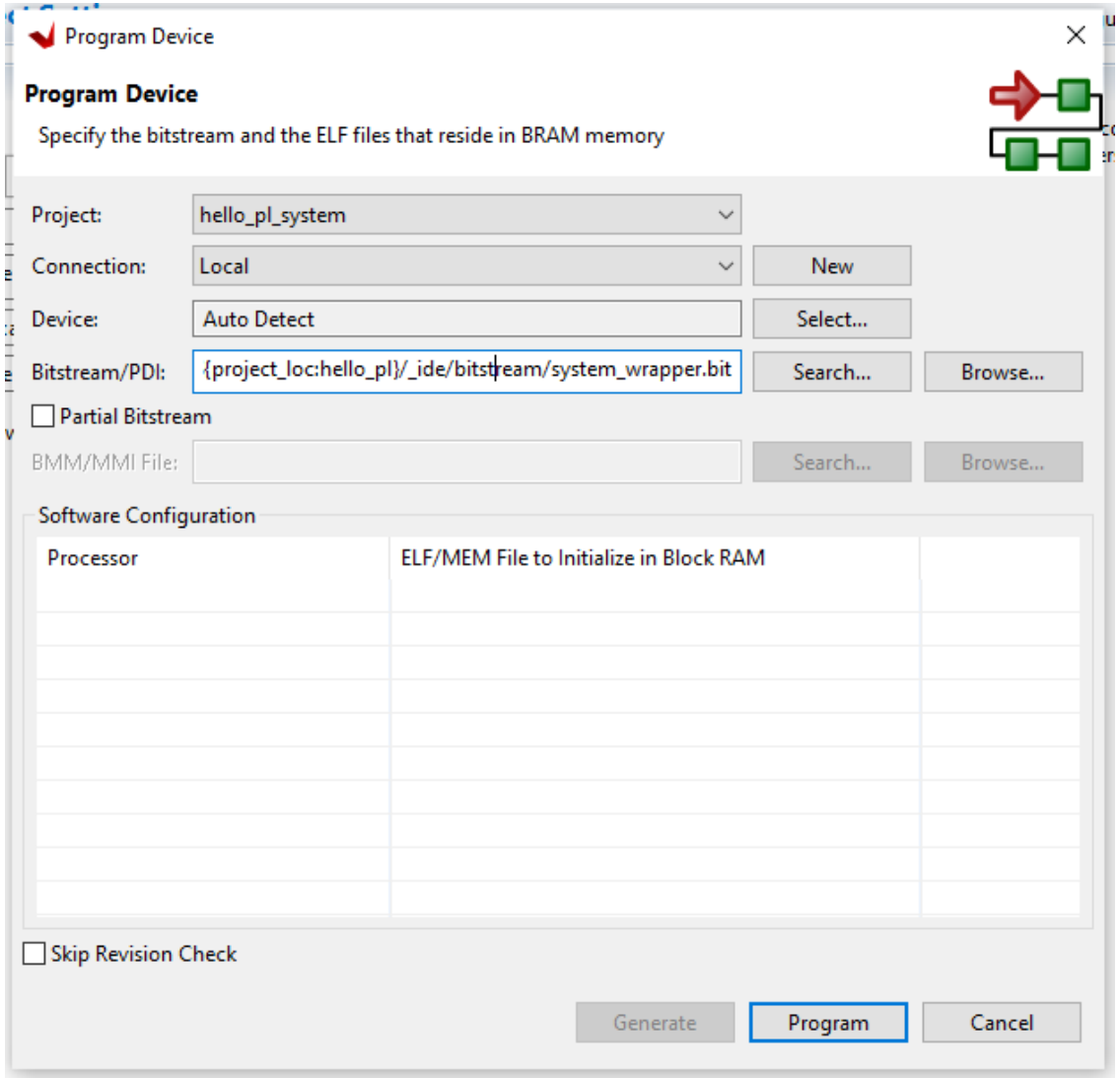
Testing the PL IP with Prepared Software



AMD XILINX

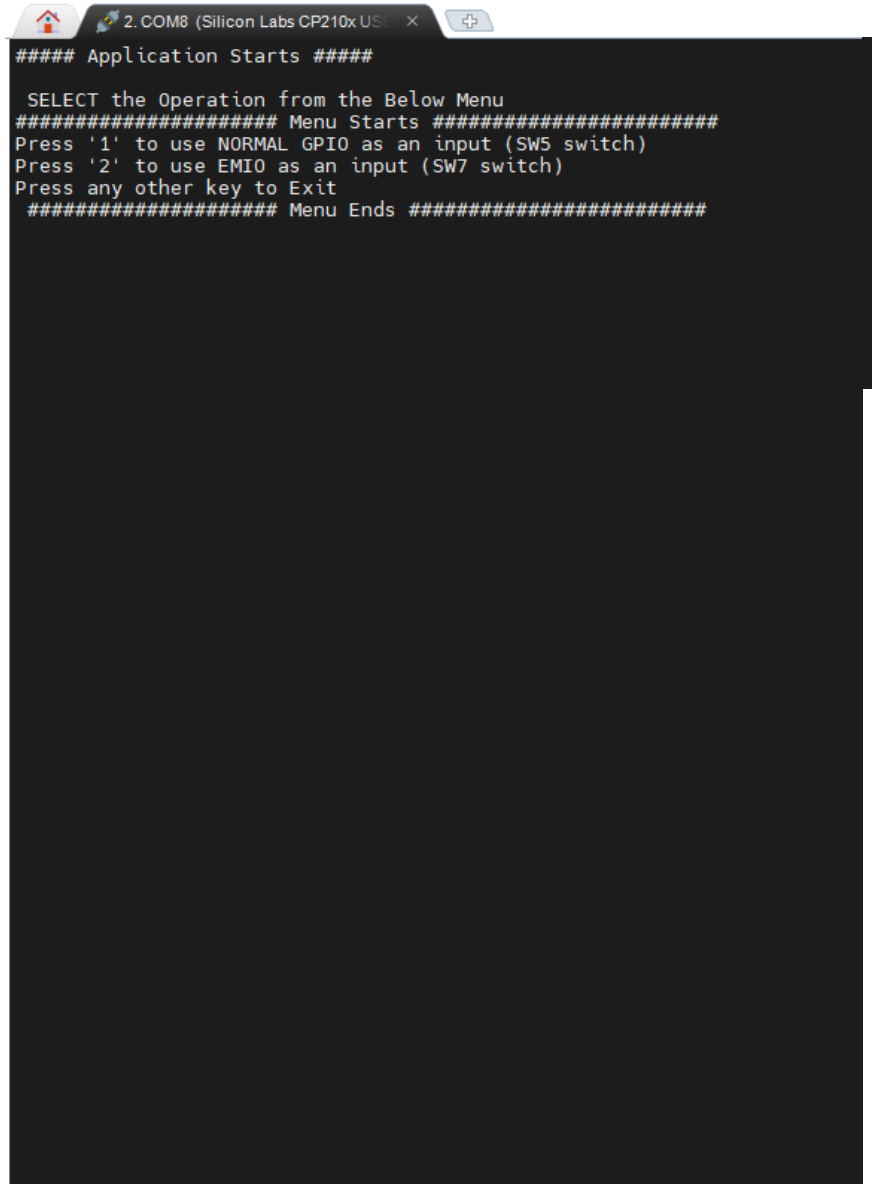
Testing the PL IP with Prepared Software





AMD XILINX

Testing the PL IP with Prepared Software



AMD XILINX

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 GuitHub

Vitis Drivers API Documentation - gpio

AMD
XILINX

Thank you very much for your attention!

Course Agenda
2023