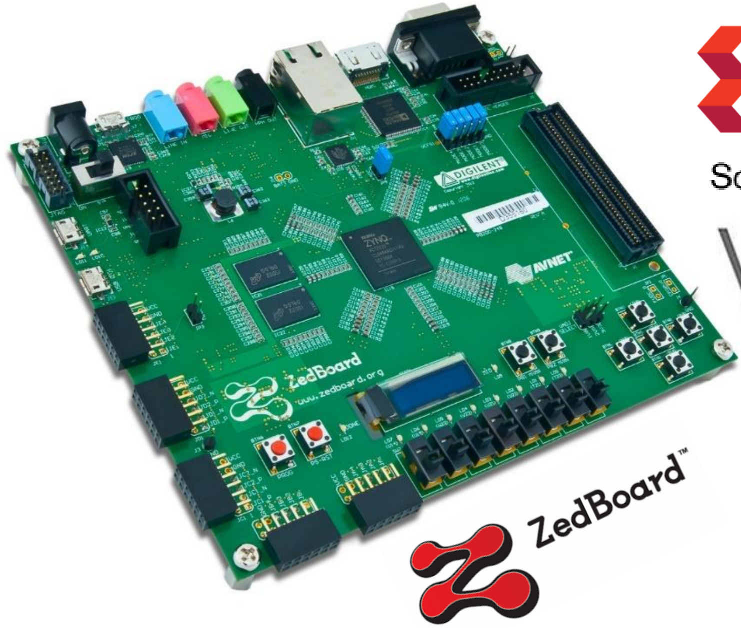
DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

A2A Algorithms to Architecture

ECE573
Advanced Embedded Logic Design (AELD)



SDK Software Development Kit

VIVADO

Dr. Sumit J Darak
Algorithms to Architectures Lab
Associate Professor, ECE Department
IIIT Delhi
http://faculty.iiitd.ac.in/~sumit/

- The material for this presentation is taken from various books, courses and Xilinx/ARM XUP resources. The instructor does not claim ownership of the material presented in this class.

Lab 1

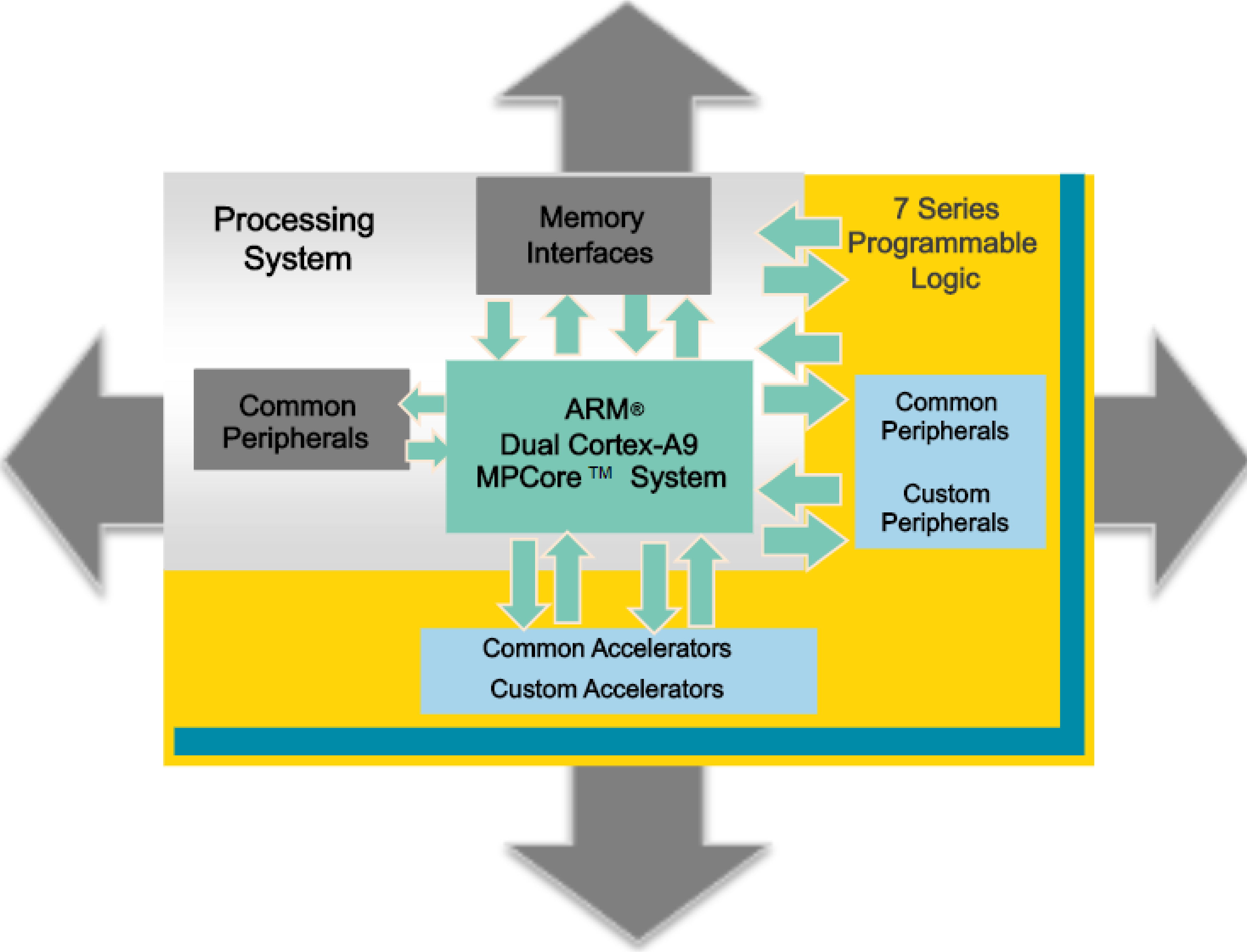
- In this lab, we will discuss the architecture of Zynq SoC and learn how to program ARM processor to communicate over UART (or JTAG) using Hello World and Floating point examples.
- We will use Vivado (2019.1) to create the hardware system and SDK (Software Development Kit) to create an example application to verify the hardware functionality.
- **Topics to explore:** 1) Zynq Architecture, 2) Zynq configuration, 3) Create a Vivado project for a Zynq system, 4) Use SDK to create C-based applications for ARM, 5) Verification on the local and remoted hardware, 6) Debugging, and 6) Extension on Vivado 2021.2

Pre-requisites

- Vivado 2019.1 with SDK (Check final video for 2021.2)
- Access to local or remote Zedboard (Digilent)
- Add board files in your Vivado catalogue. Please follow this link:

https://reference.digilentinc.com/learn/software/tutorials/vivado-board-files/start

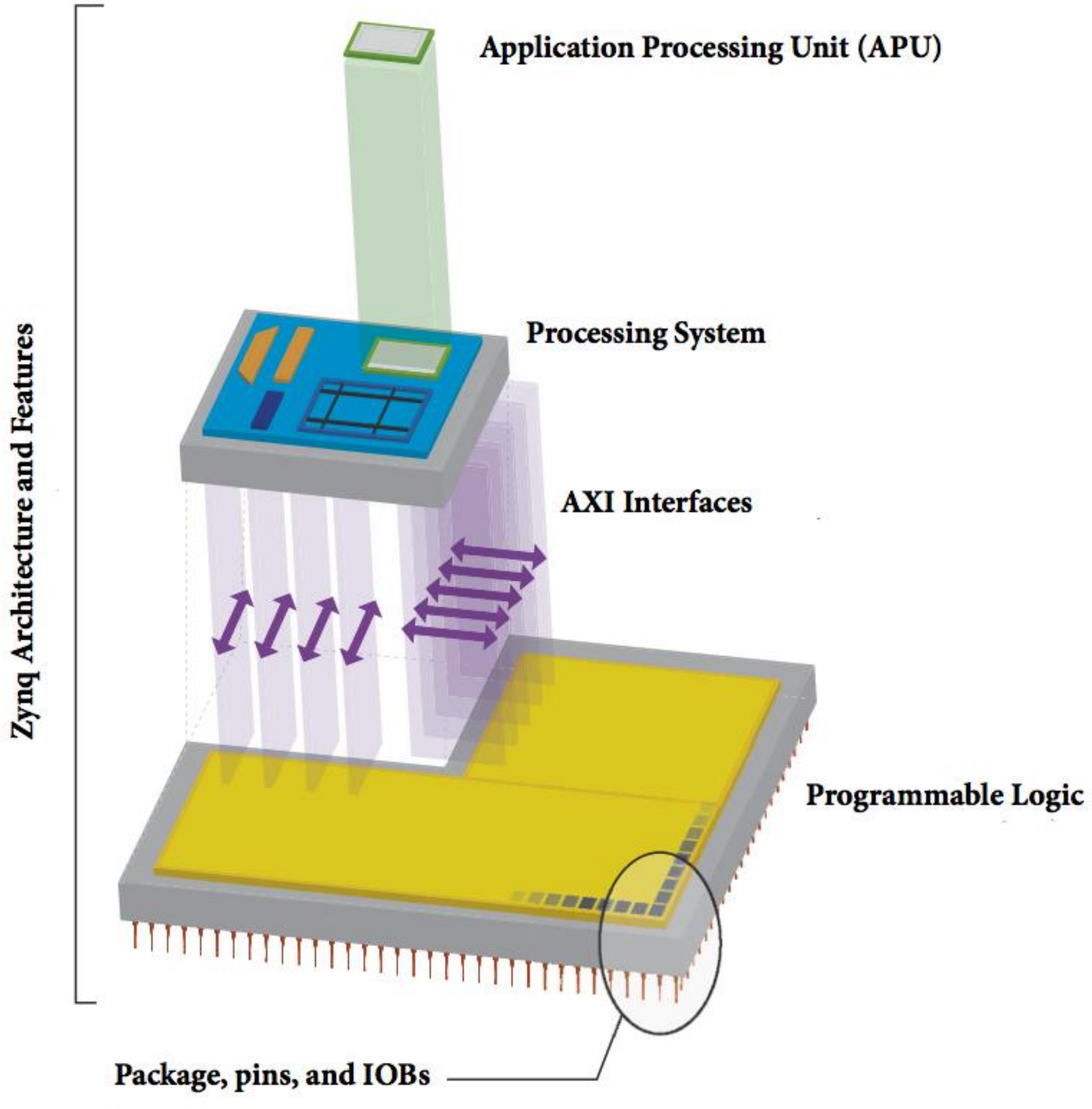
Zynq SoC



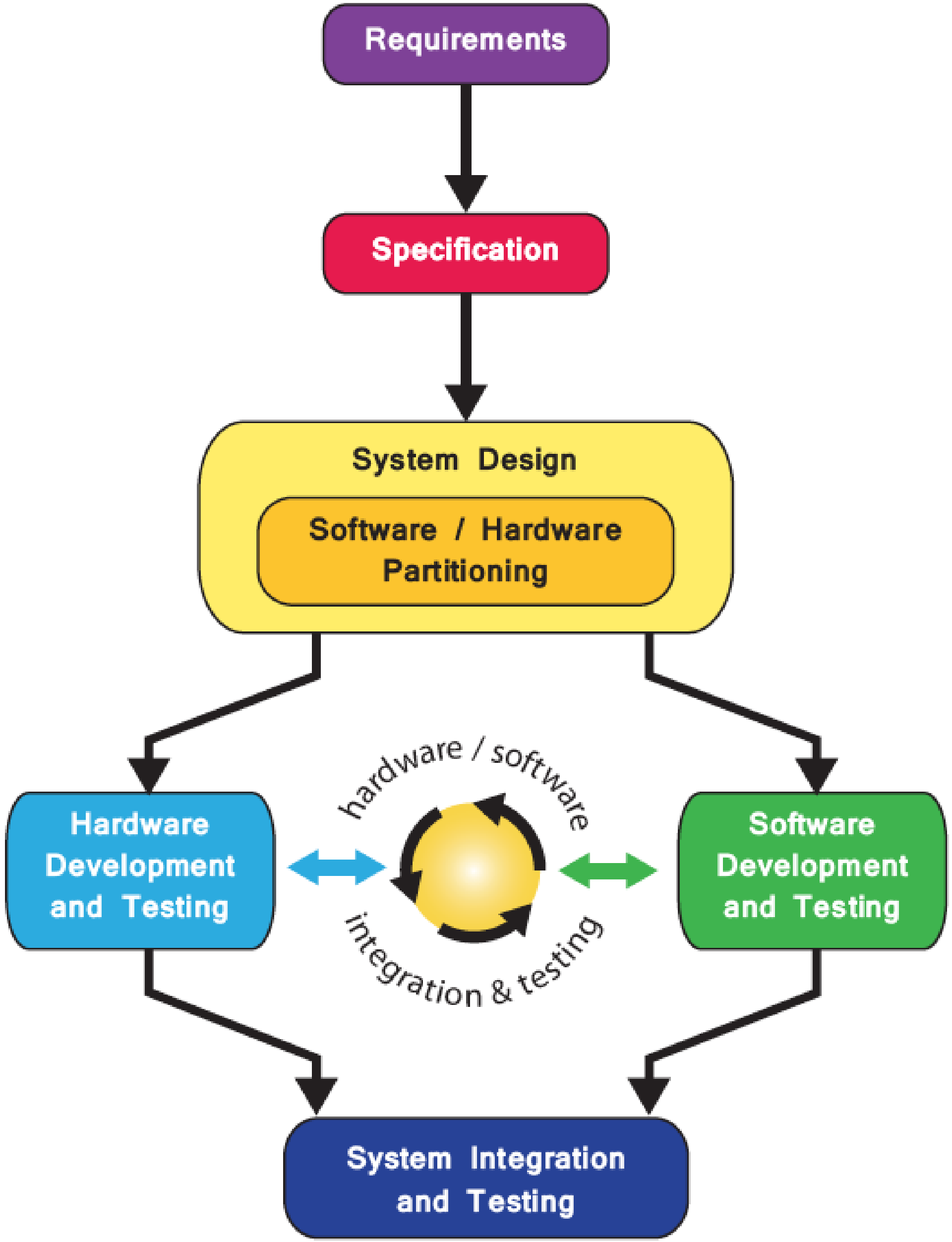
Processing System
Memory Interfaces
7 Series Programmable Logic
Common Peripherals
ARM®
Dual Cortex-A9
MPCore ™ System
Common Peripherals
Custom Peripherals
Common Accelerators
Custom Accelerators

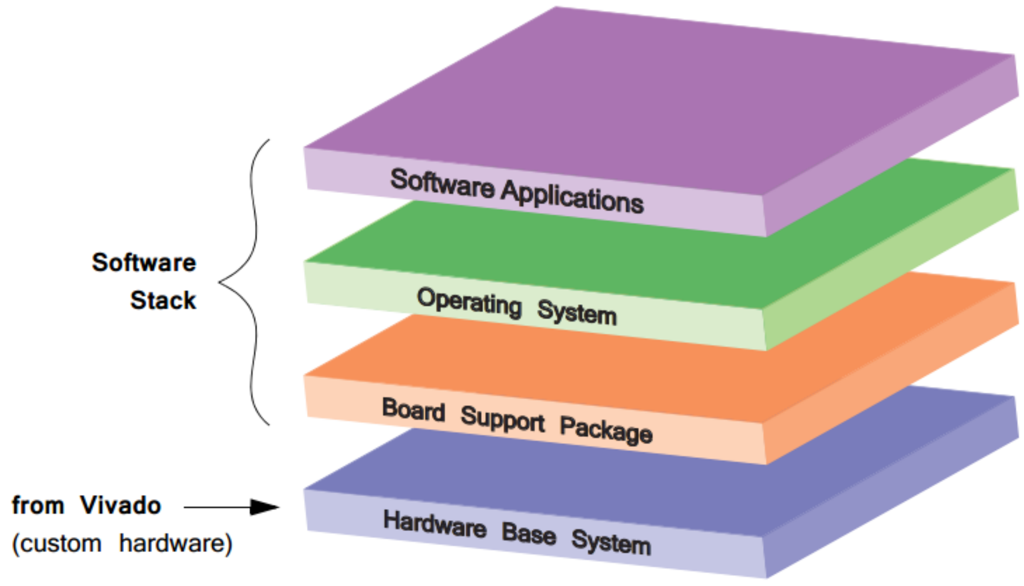
Zynq

- ❖ Not an ordinary FPGA
- ❖ Not an ordinary microprocessor
- ❖ Unique blend of two technologies with powerful interconnect
- ❖ In Zynq, the ARM Cortex-A9 is an application grade processor, capable of running full operating systems such as Linux, while the programmable logic is based on Xilinx 7-series FPGA architecture



Zynq Design Flow

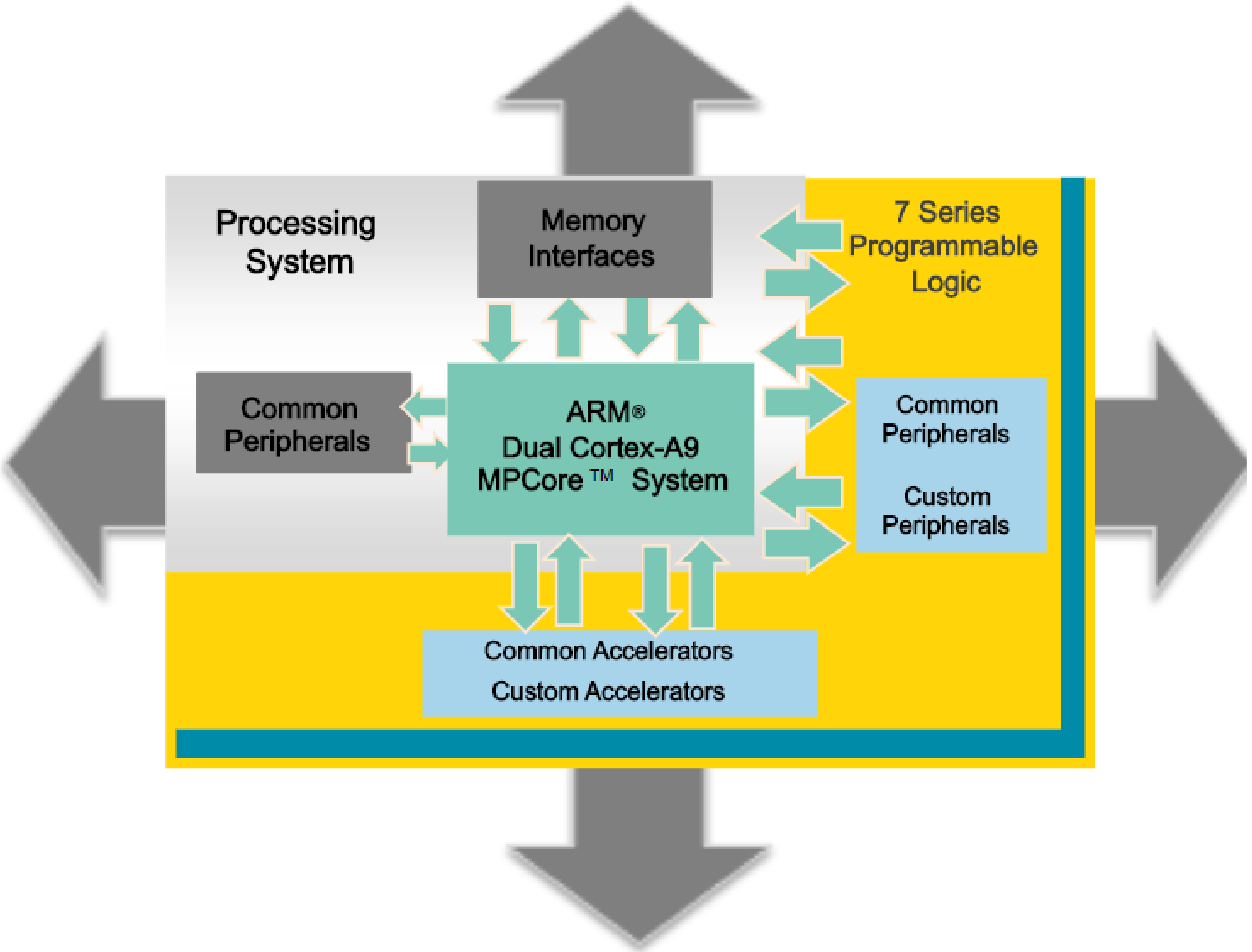


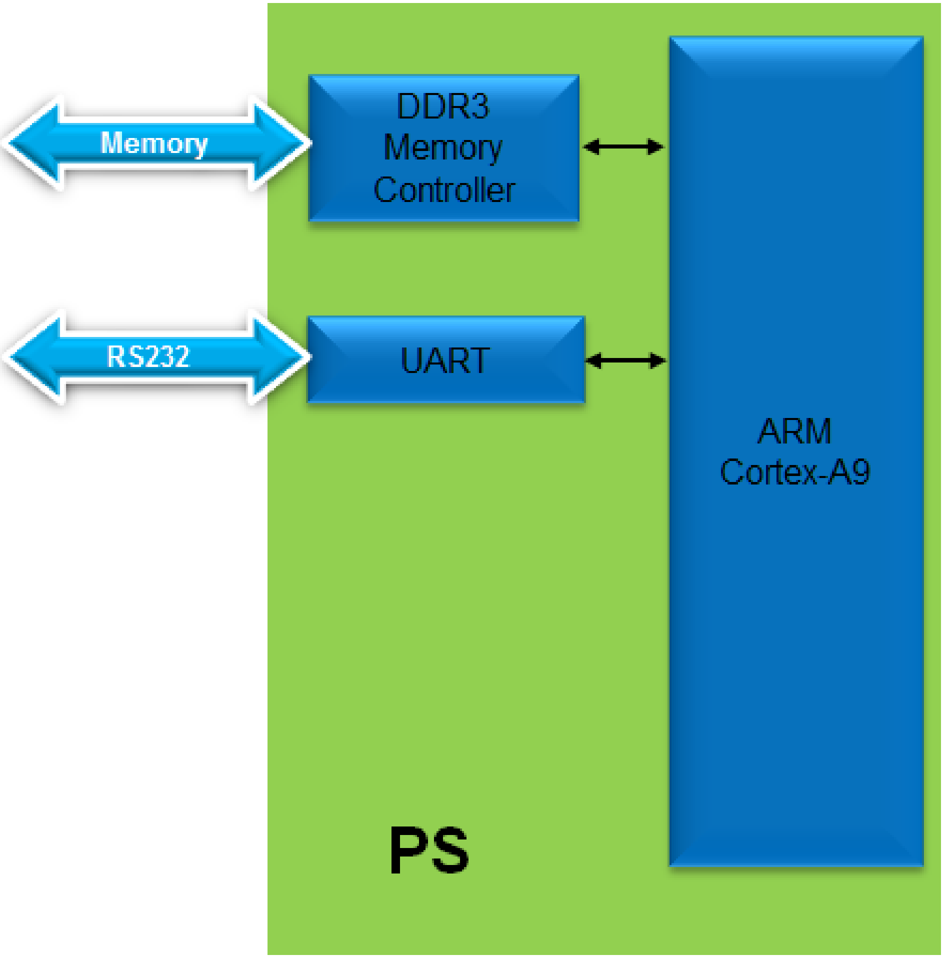


- ***Board Support Package (BSP)*:** Set of low-level drivers and functions that are used by the next layer up (the *Operating System or baremetal*) to communicate with the hardware
- **Software Applications** run on top of the Operating System — these collectively represent the uppermost layer in the software stack

- SDK provides the environment for creating BSPs, and developing and testing software for deployment in the upper layers.
- BSP (includes hardware parameters, device drivers, and low-level OS functions) should be refreshed if changes are made to the hardware base system.
- The purpose of ELF files is to program the PS, while BIT files are used to program the PL.
- Not all designs require both an Executable Linkable Format (ELF, *.elf) file and a BIT (*.bit) file to configure the device.
- If only one part of the Zynq device is used (PS or PL), then only the corresponding file type is needed.

Lab 1: Introduction to Vivado and SDK



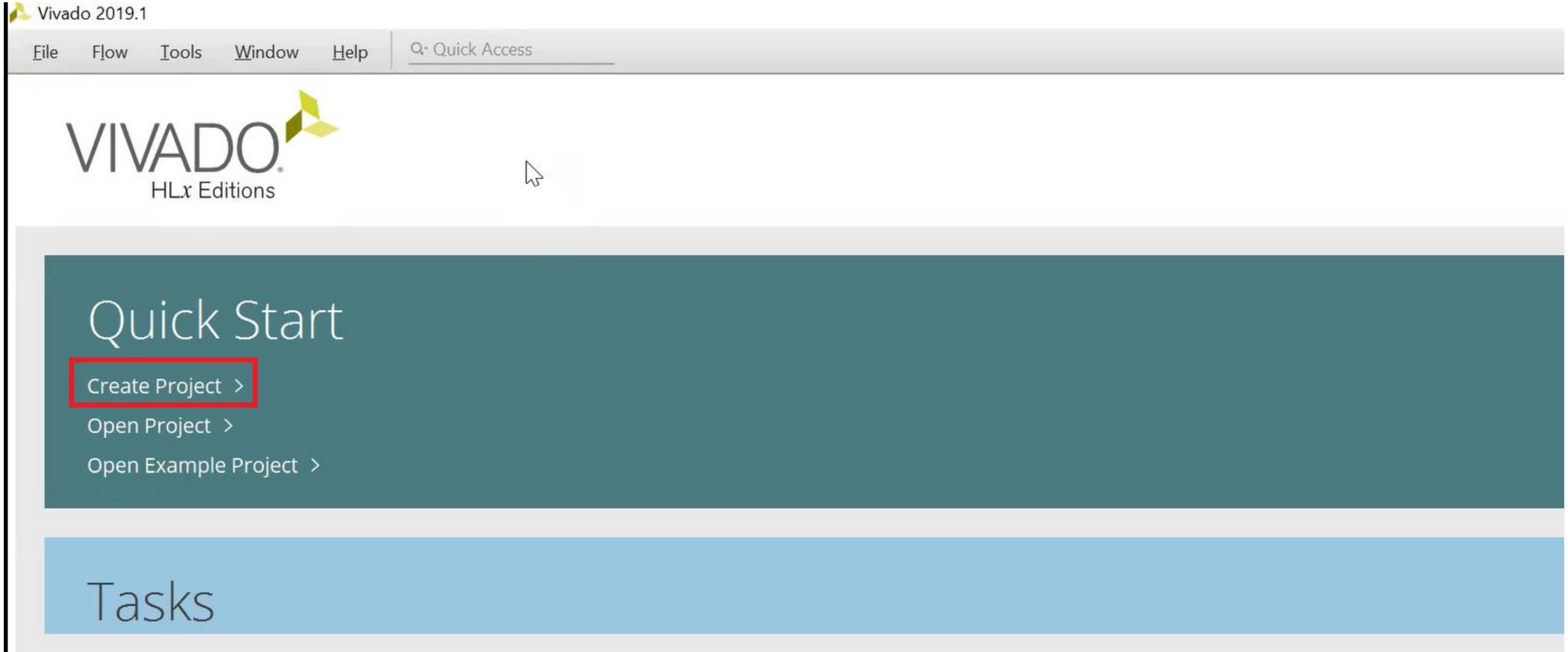


Lab 1: Introduction to Vivado and SDK

Vivado Design Flow

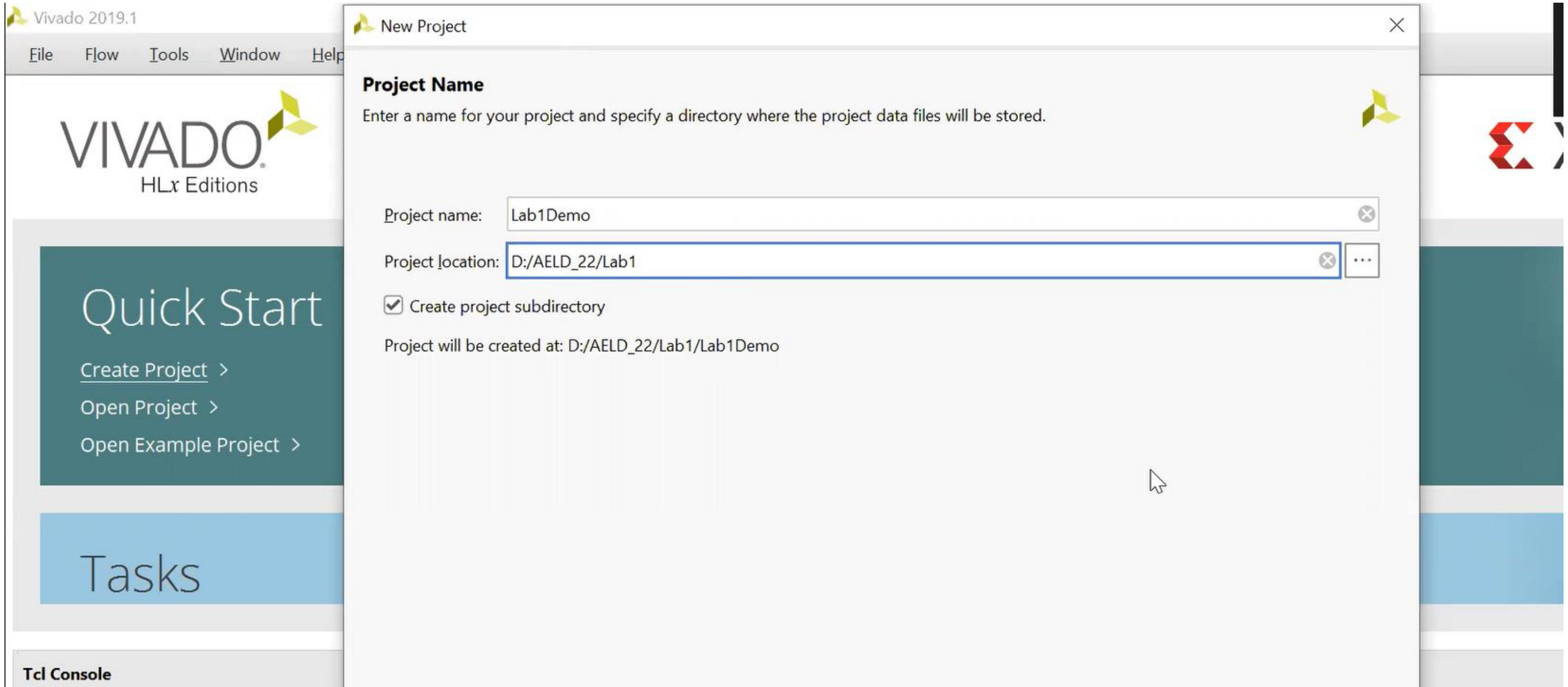
Vivado Steps: 1

- Open Vivado 2019.1 and click on Create Project
- Click Next in the next window.

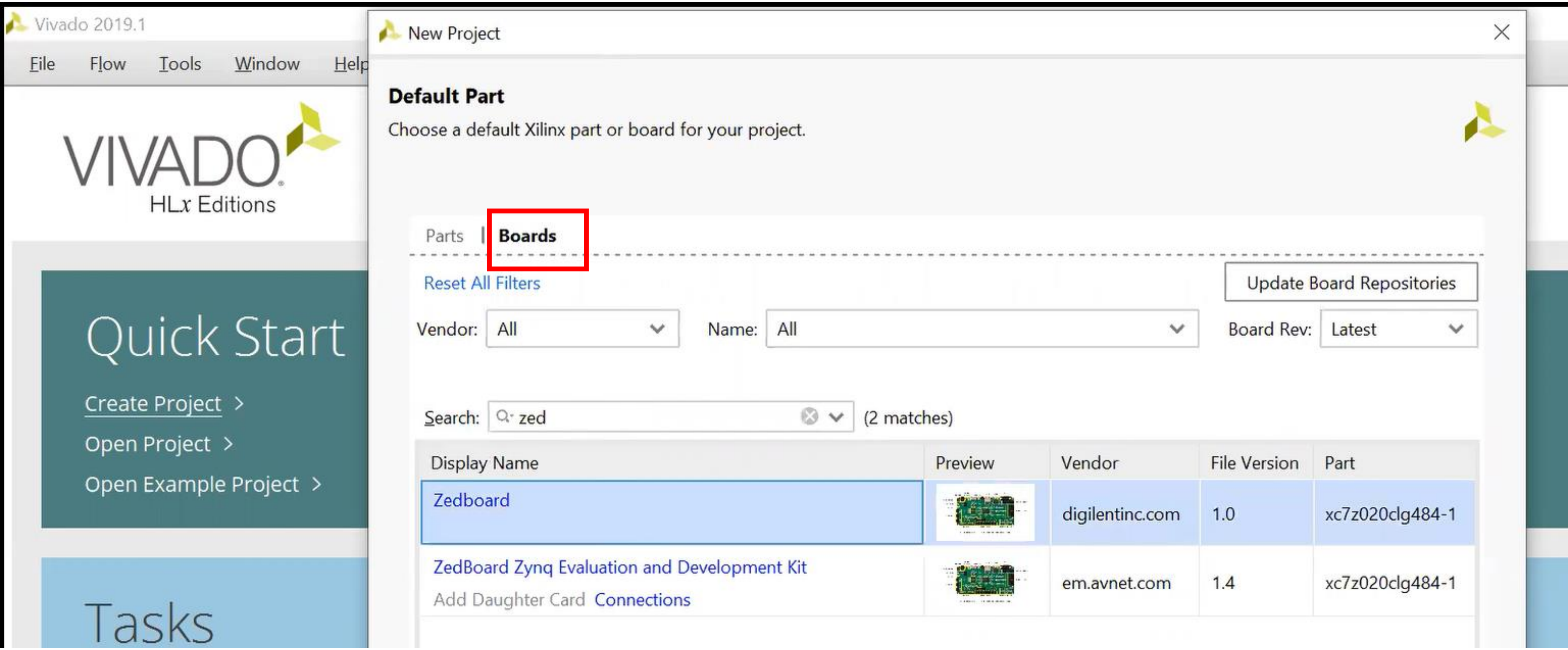


Vivado Steps: 2

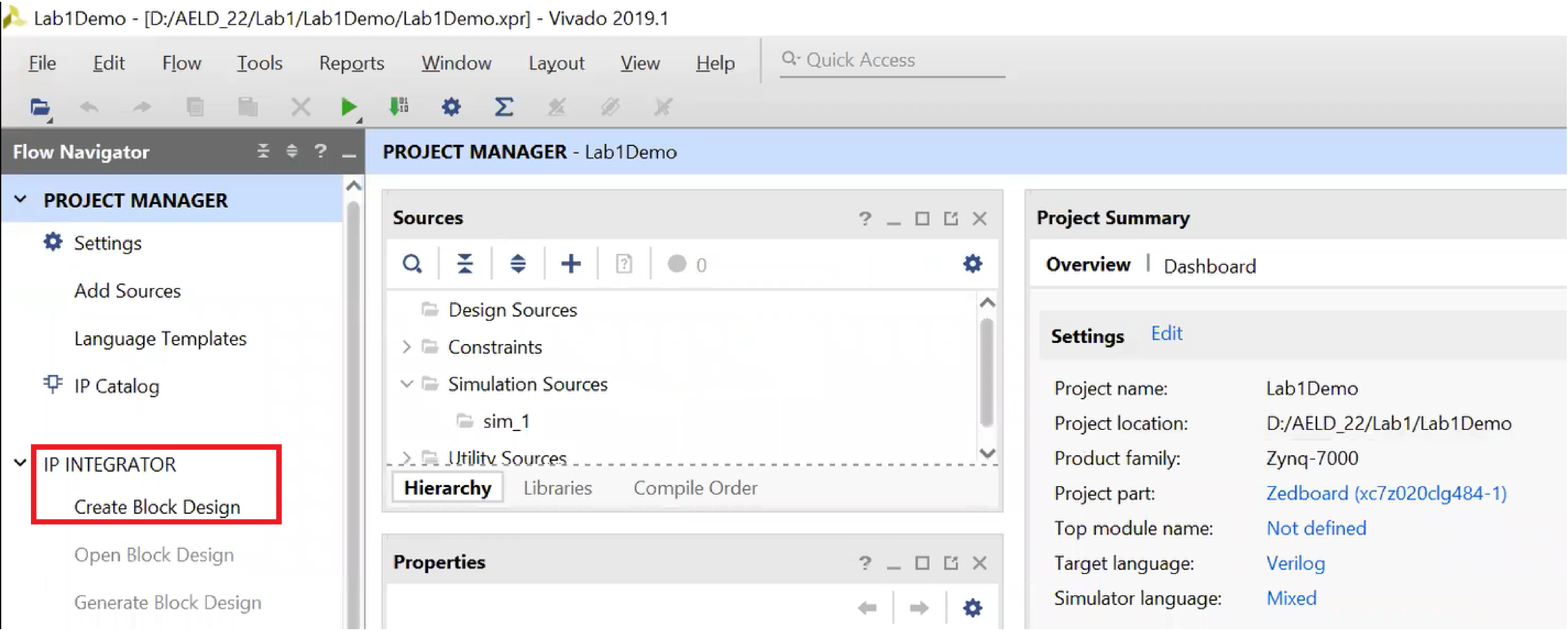
- Use the folder without any space in the address.
- On the next screen, select “Do not specify sources at this time” under RTL Project tab.



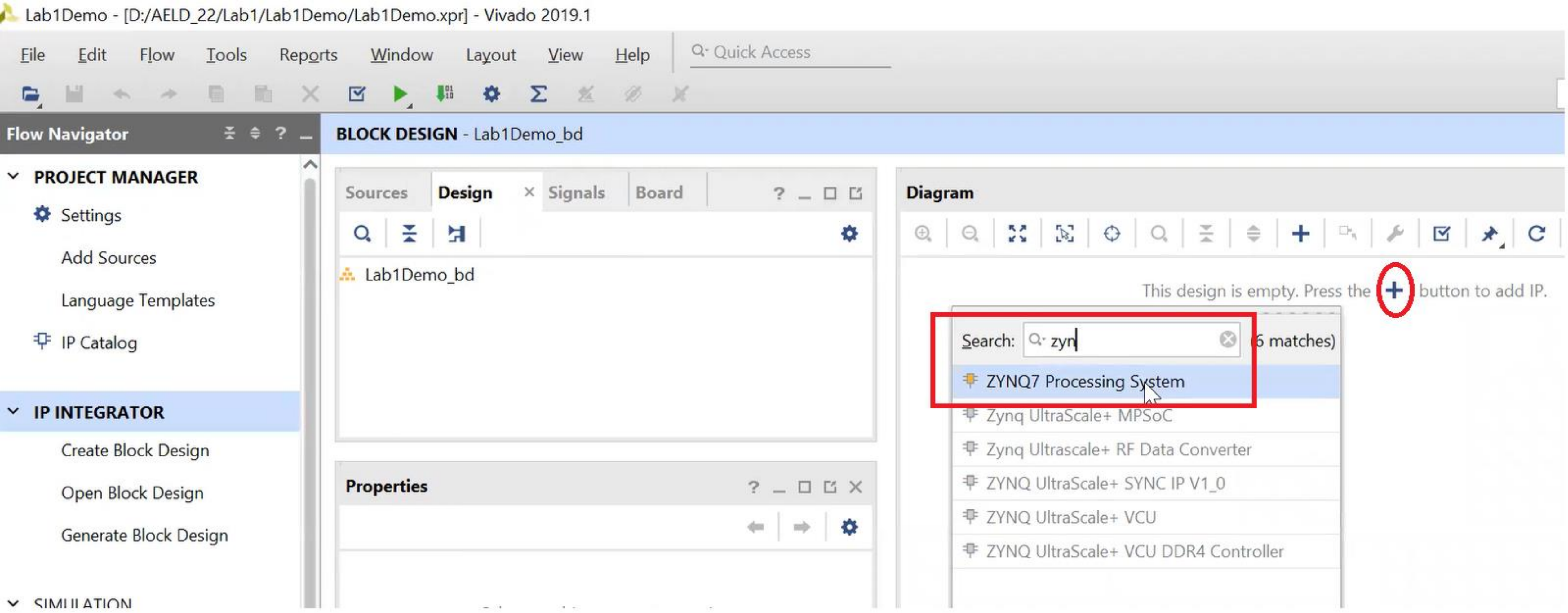
Vivado Steps:3 -> Select Digilent Zedboard



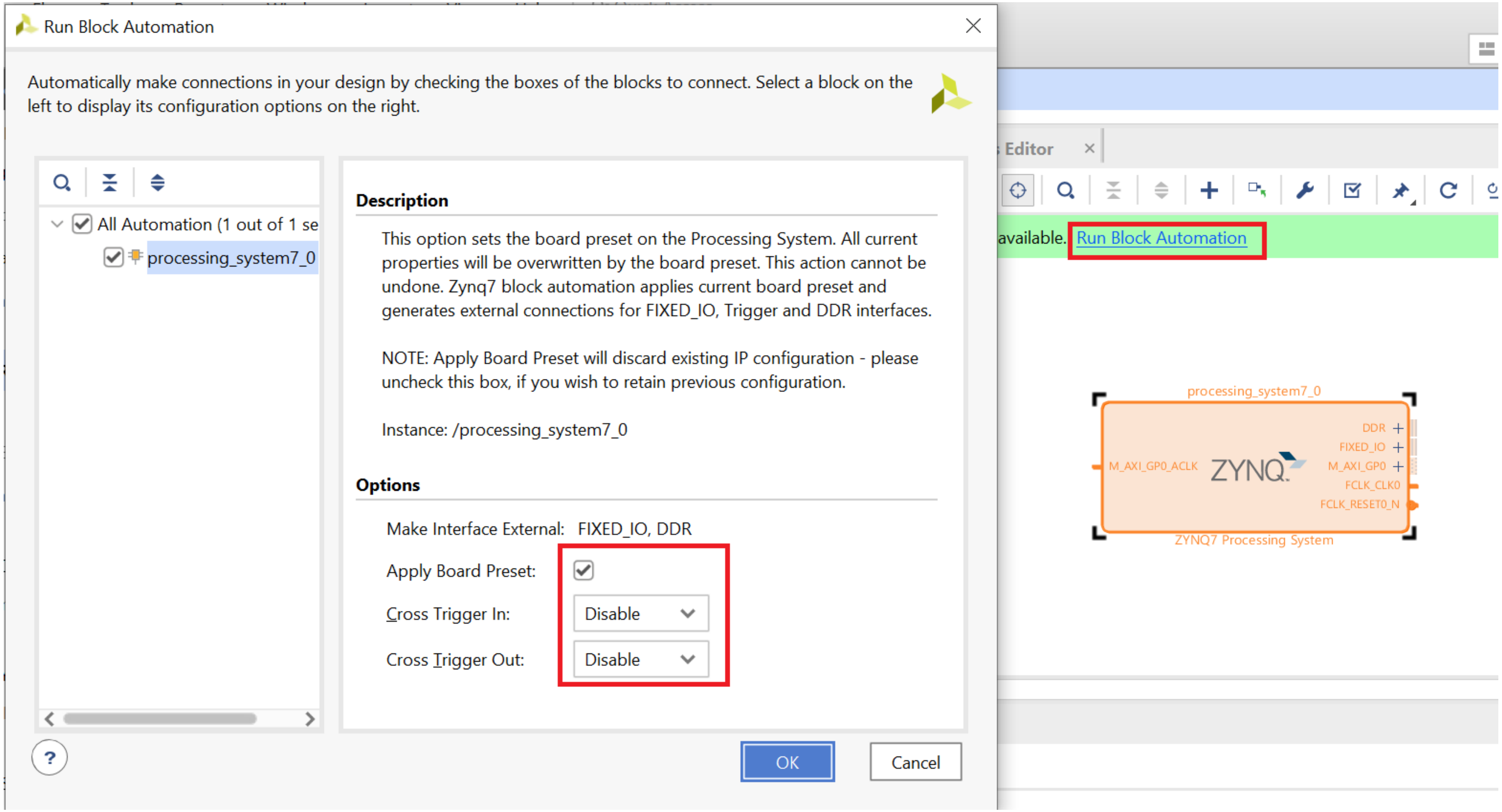
Vivado Steps: 4



Vivado Steps: 5

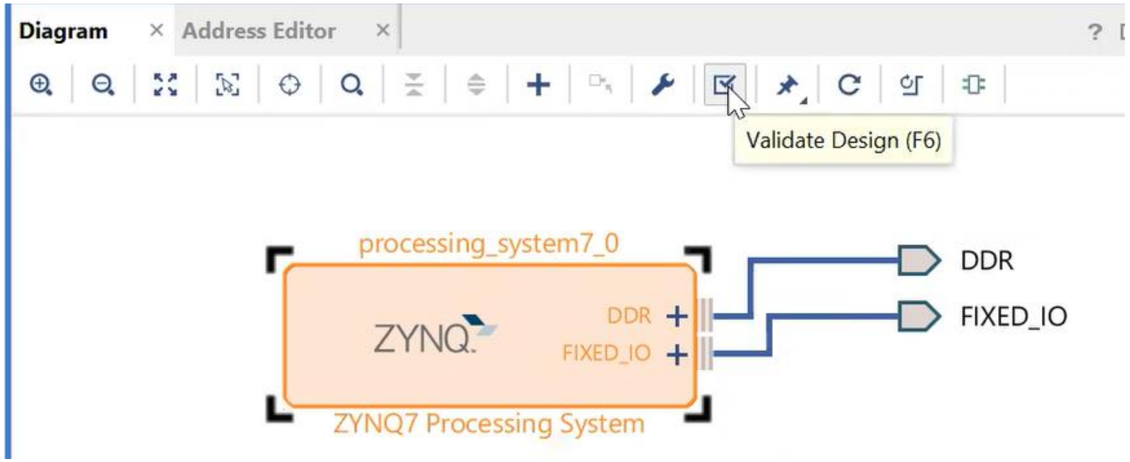


Vivado Steps: 6



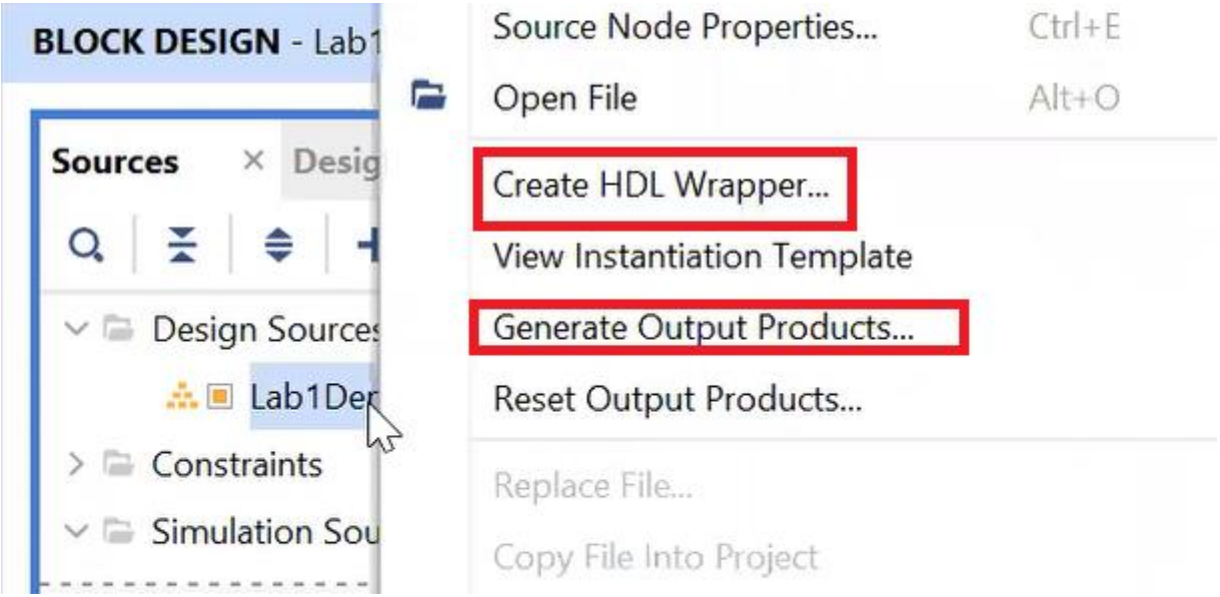
Vivado Steps: 7

- Double click on Zynq IP and configure it as per the application requirements.
- For Lab 1: Disable M_AXI_GP0 (PS-PL Configuration -> AXI Non Secure Enablement -> GP Master AXI Interface), Timer 0 (MIO Configuration -> Application Processor Unit), All peripherals except UART1, Clock Reset (PS-PL configuration -> General)
- Validate the design



Vivado Steps: 8

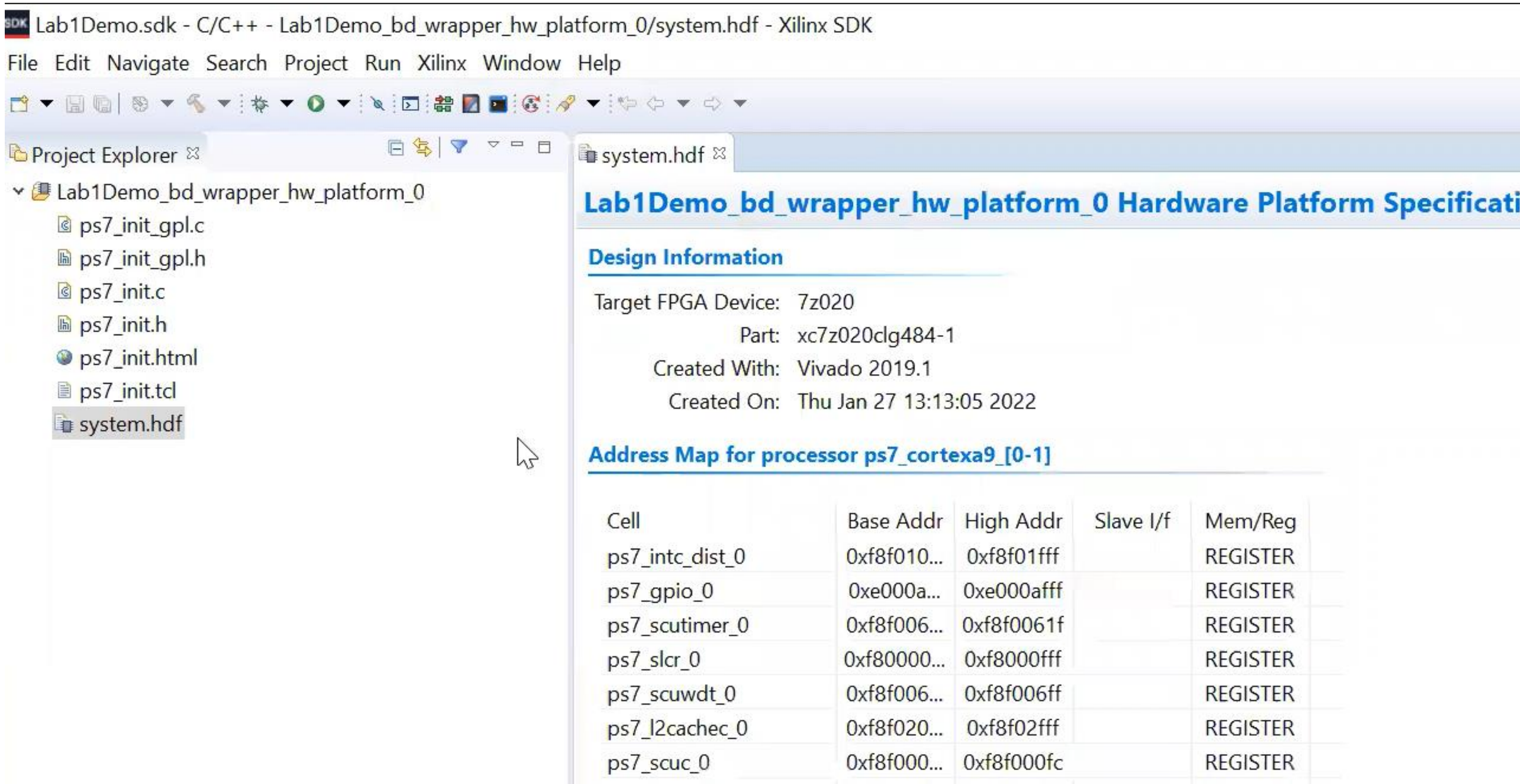
- Generate output products
- Create HDL wrapper
- File -> Export Design
- File -> Launch SDK



SDK Design Flow

SDK Steps: 1

- After Launch SDK from Vivado, do not click on anything till you see the screen as below:

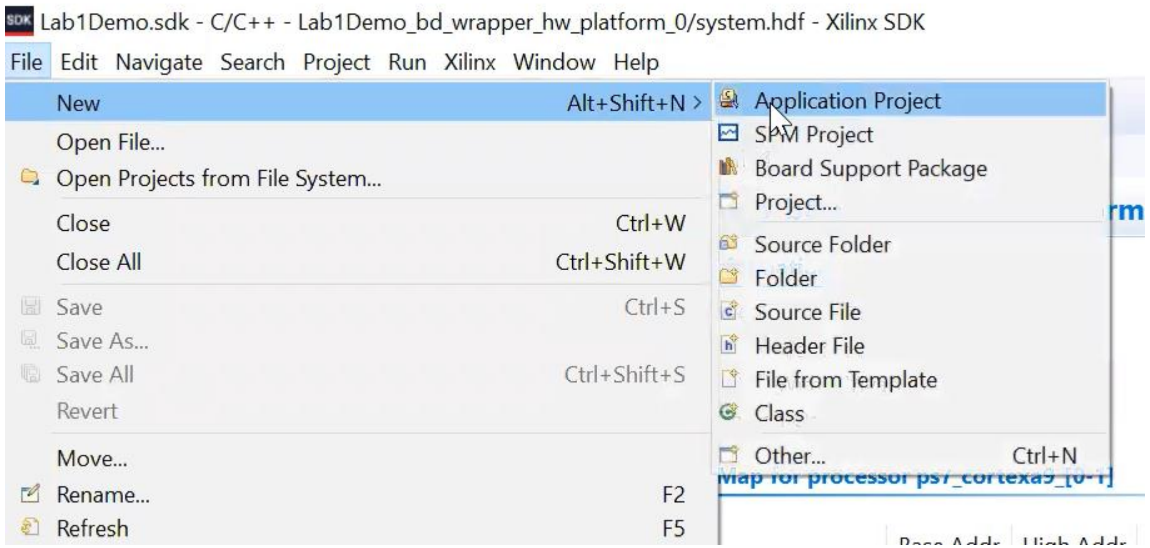


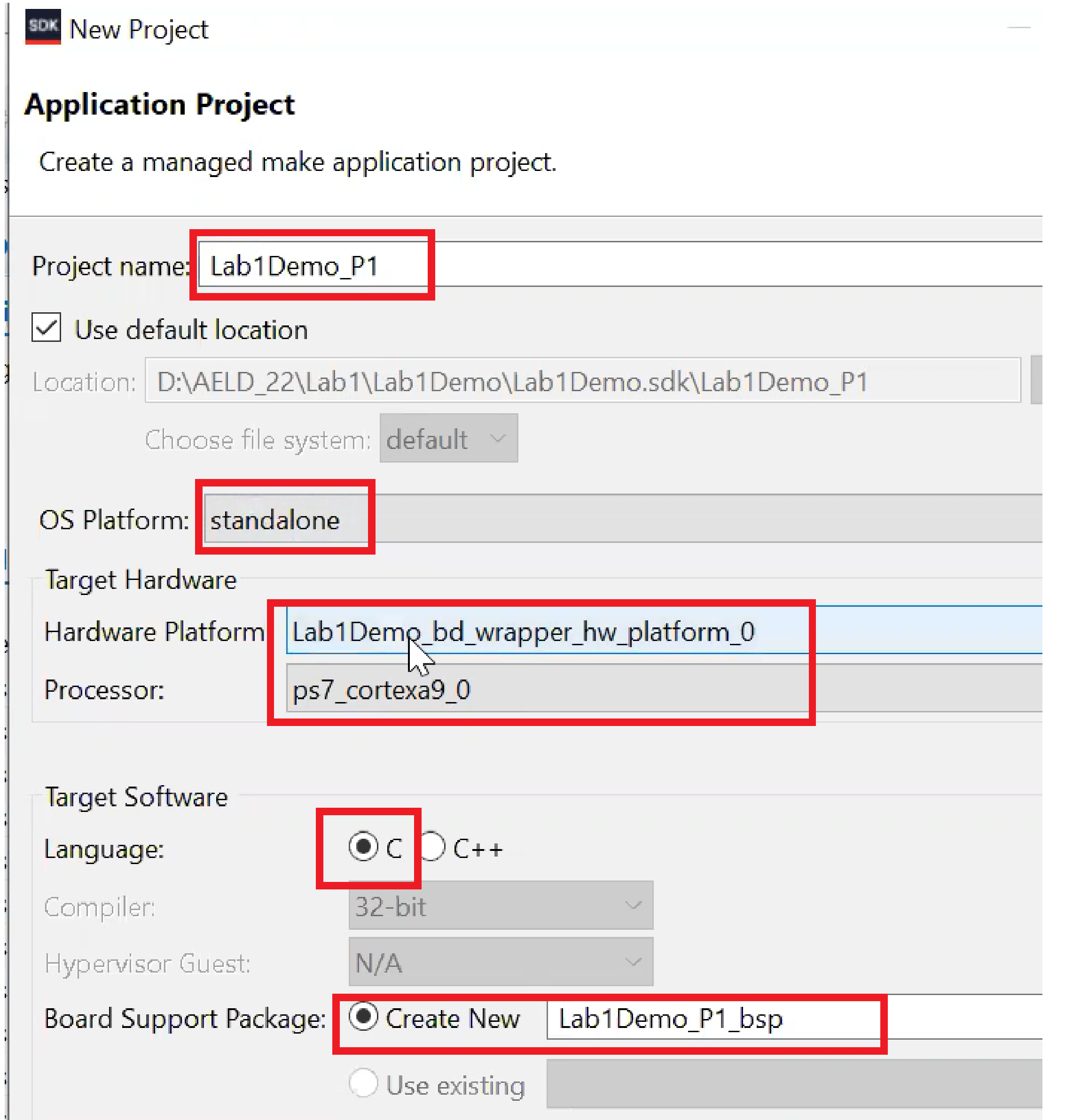
SDK Steps: 1

- The system.hdf file (Hardware Description File) for the Hardware platform should open in the preview pane.
- Double click system.hdf to open it if it is not.
- Basic information about the hardware configuration of the project can be found in the .hdf file, along with the Address maps for the PS systems, and driver information.
- The .hdf file is used in the software environment to determine the peripherals available in the system, and their location in the address map.

SDK Steps: 2

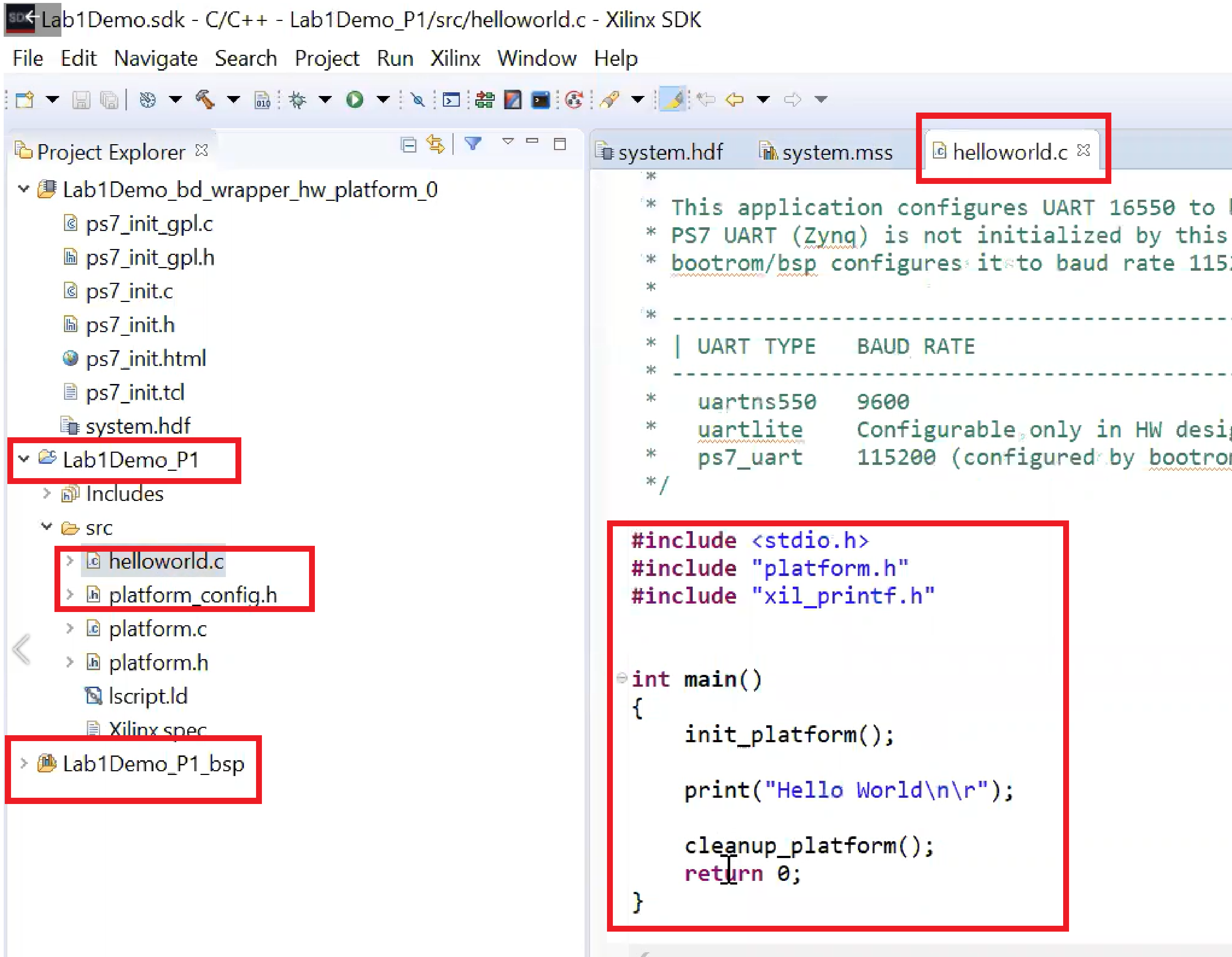
- Enter appropriate name for the SDK project and click Next (Do not click on Finish)
- **Select Hello World Project**





SDK Steps: 3

- Check out the newly created application project



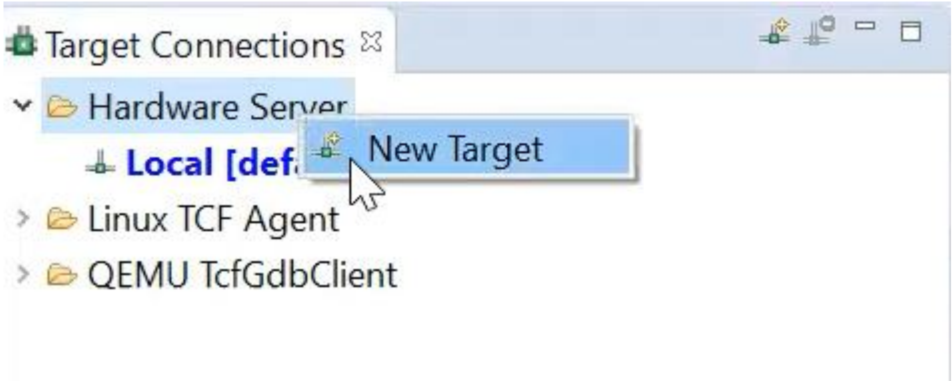
SDK Steps: 4

- Replace the main function in HelloWorld.c with the code shown here.
- Understand the functionality of this code
- Note that the project is automatically build (See the Console window at the bottom)

```c
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
int main()
{
    init_platform();
    // Define few variables to test the printing on UART
    int data_int = 12; // Integer 2 bits
    float data_SPFP =229911.967653; // single precision float 32 bits
    double data_DPFP=229911.967653; // double precision float 64 bits
    // Display on UART
    printf("Hello World\n\r");
    printf("Variable data_int is an integer and its value is %d\n",data_int);
    printf("Variable data_SPFP is an float and its value is %f\n",data_SPFP);
    // Notice difference between single and double precision float
    printf("Variable data_DPFP is an double and its value is %lf\n",data_DPFP);
    // Except the data from UART Terminal
    printf("Enter the new value of data_int \n");
    scanf("%d",&data_int);
    // Display the received value on UART
    printf("The updated value of data_int is %d\n",data_int);
    cleanup_platform();
    return 0;
}
```

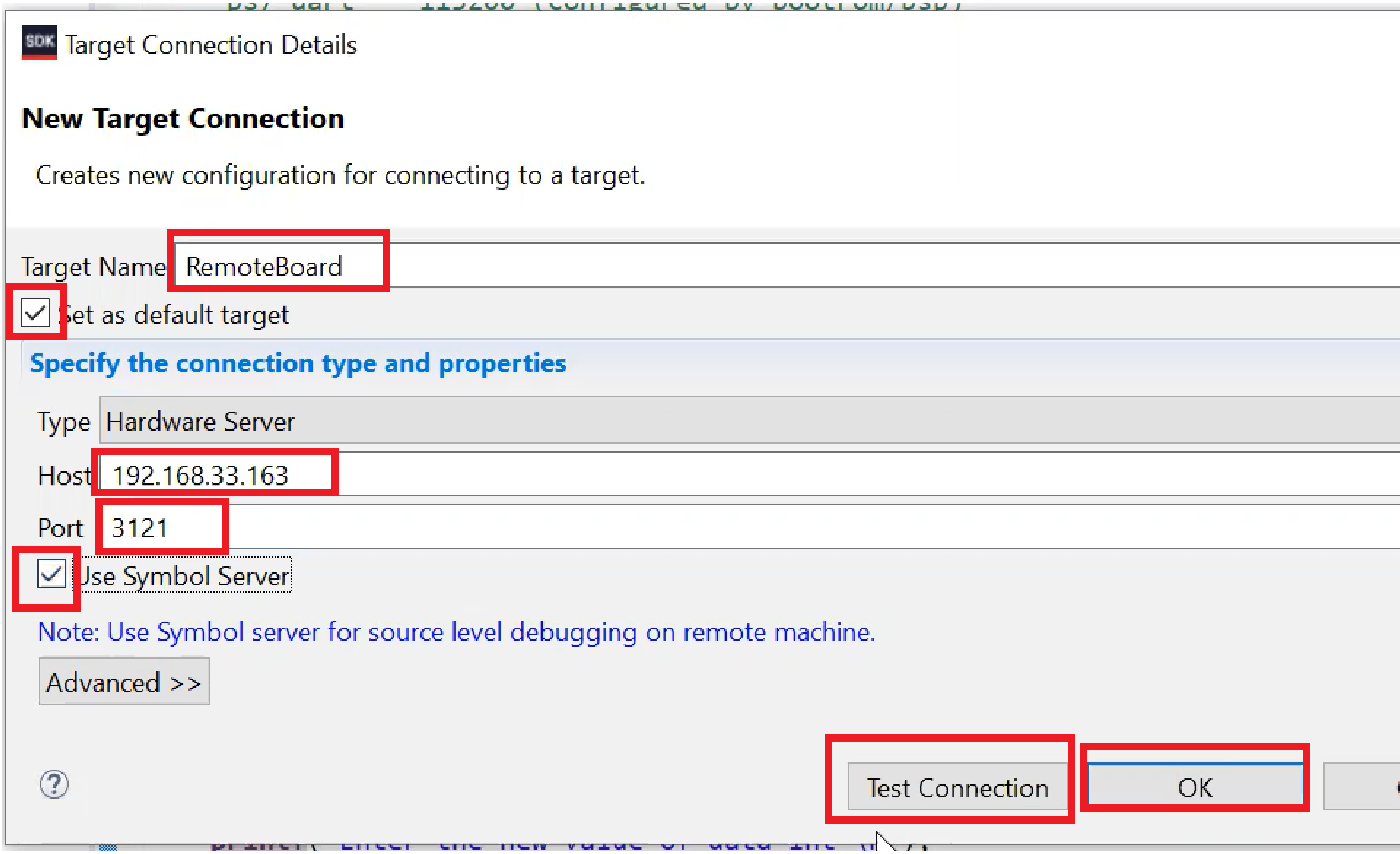
SDK Steps: 5

- Please make sure you have booked the slot for the board and you have IP address and port number with you. Note that they are valid only for the duration of slot.
- During live lab sessions, please check with TAs on how to access the board. **Please coordinate among yourselves and TAs.**
- Next step is to add the board details:
- Check Hardware Server tab at bottom left.



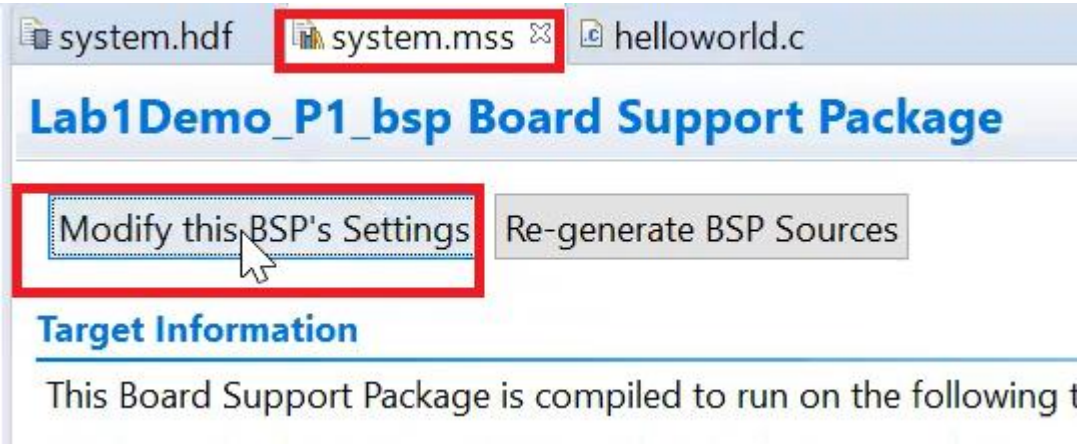
SDK Steps: 6

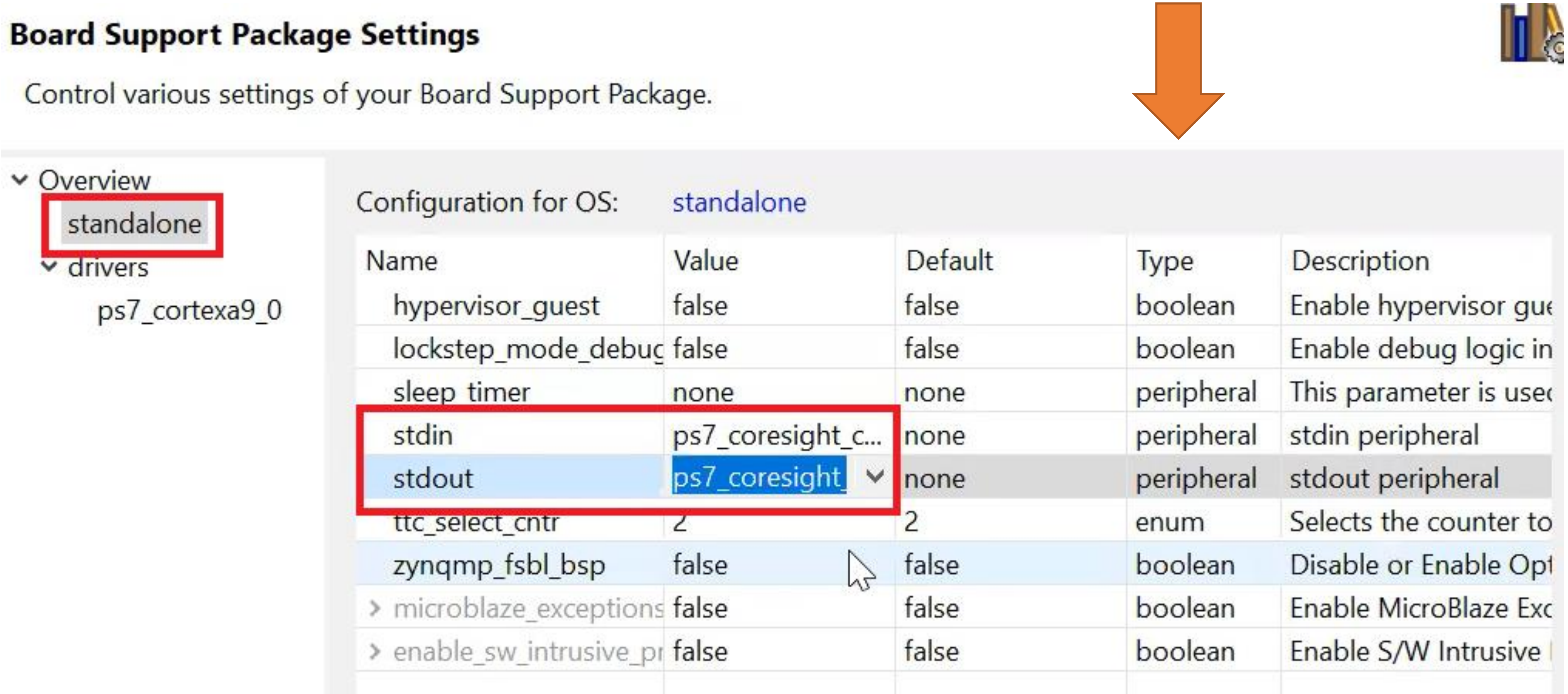
- Please enter the details as received over email. You can give any Target Name.
- Test your connection and then click on OK



SDK Steps: 7

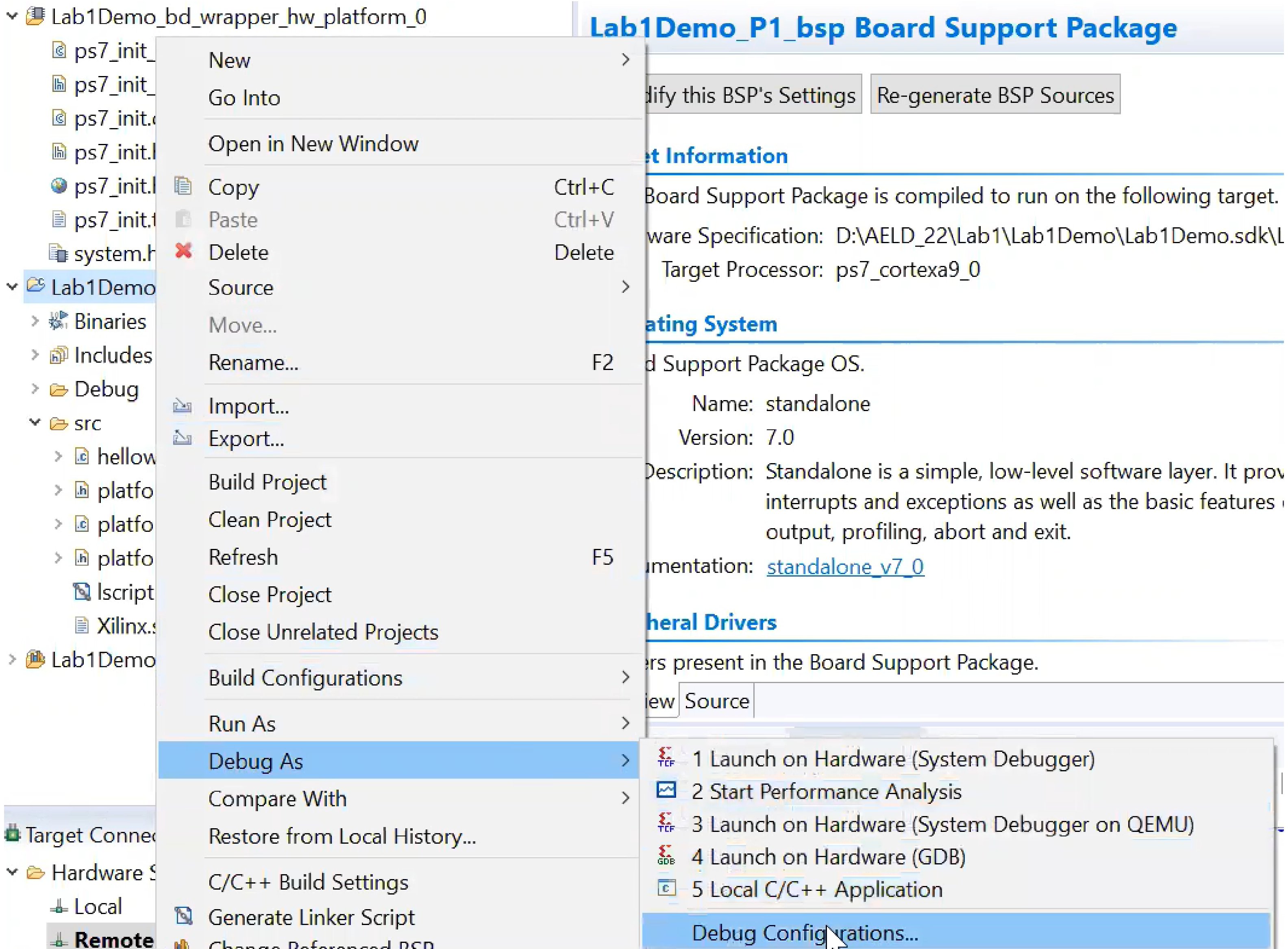
- Modify BSP settings





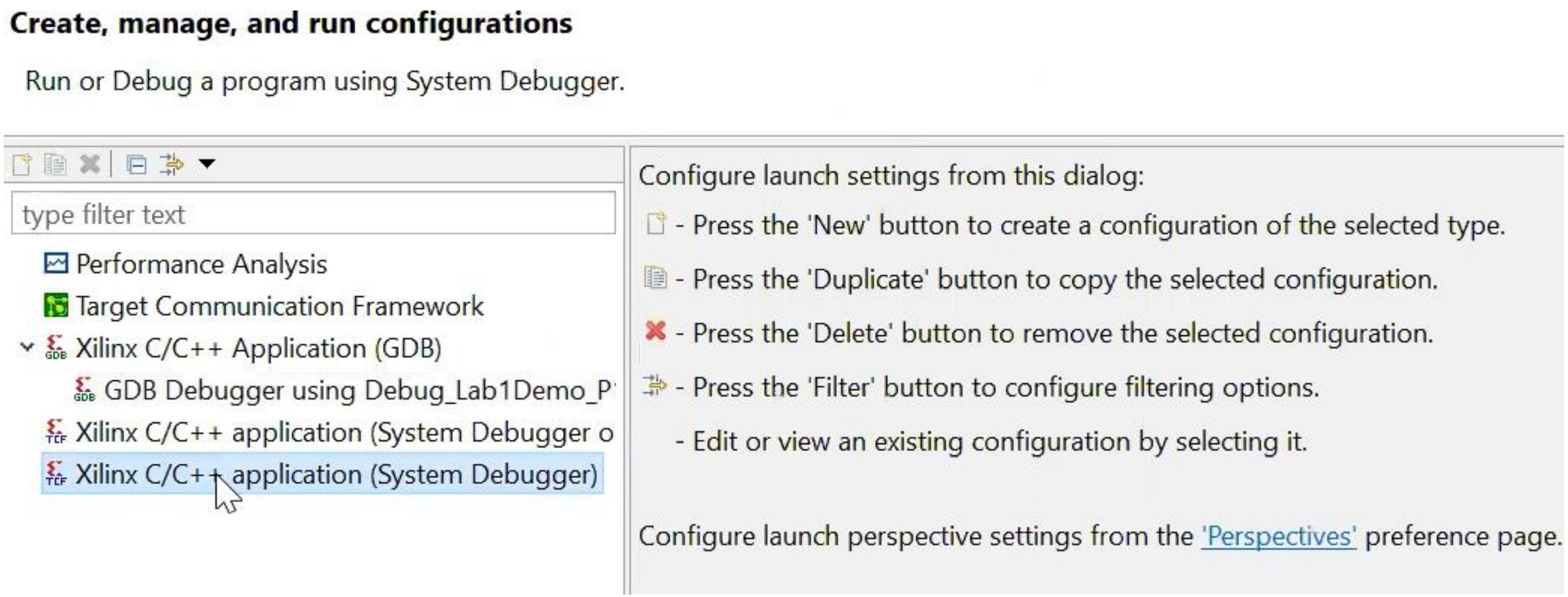
SDK Steps: 8

- Note that remote hardware can be used only in debug mode



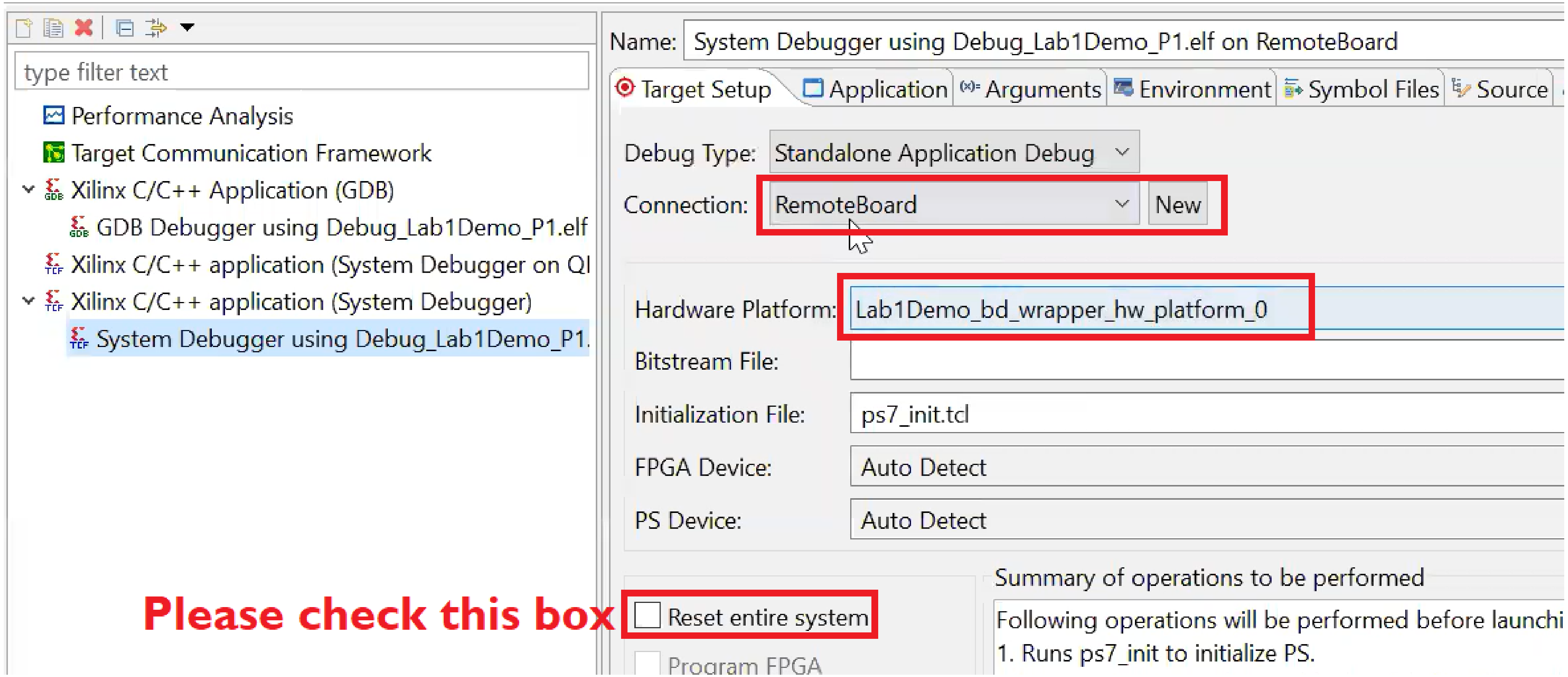
SDK Steps: 9

- Double clock on System Debugger option



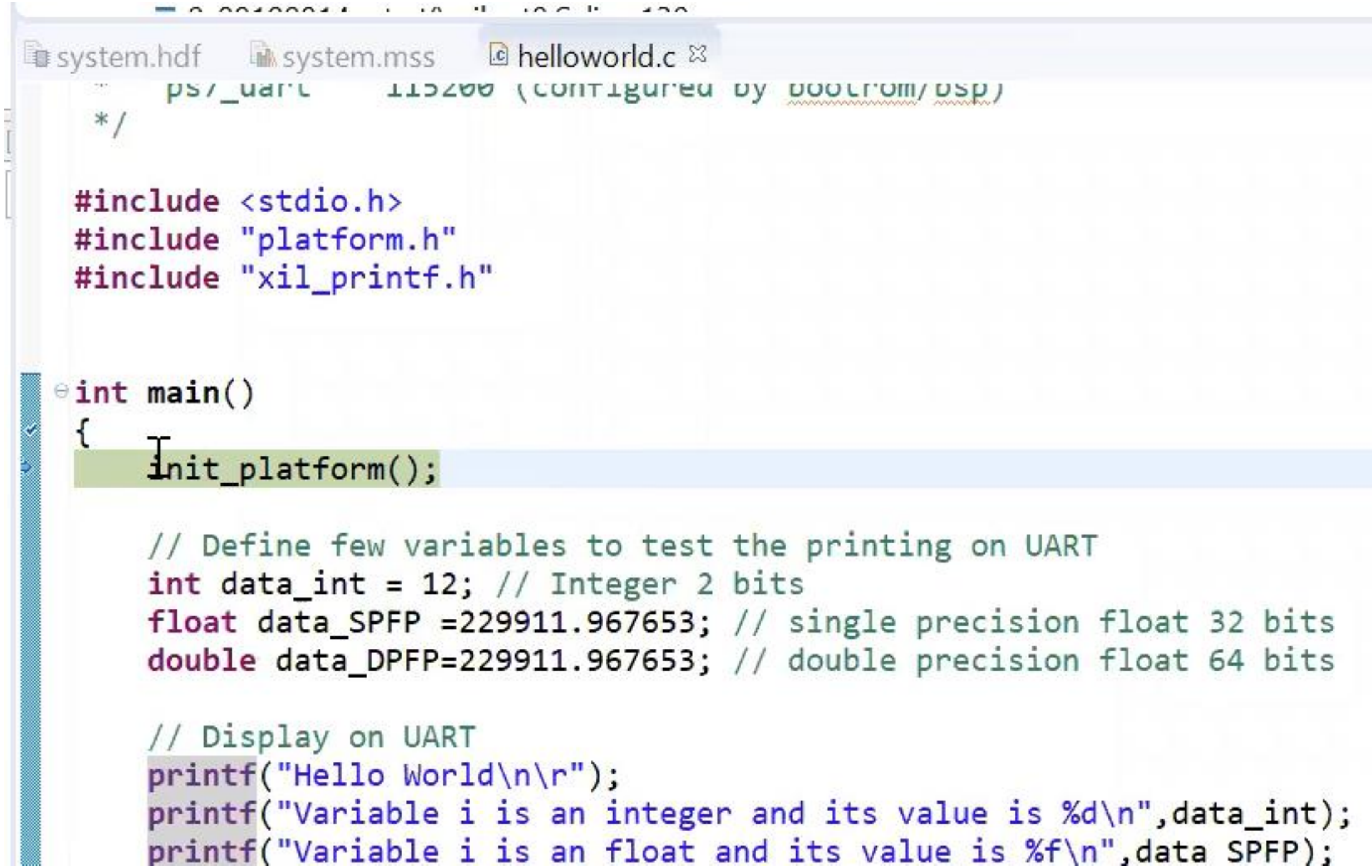
SDK Steps: 10

- After verifying all settings, click on Debug



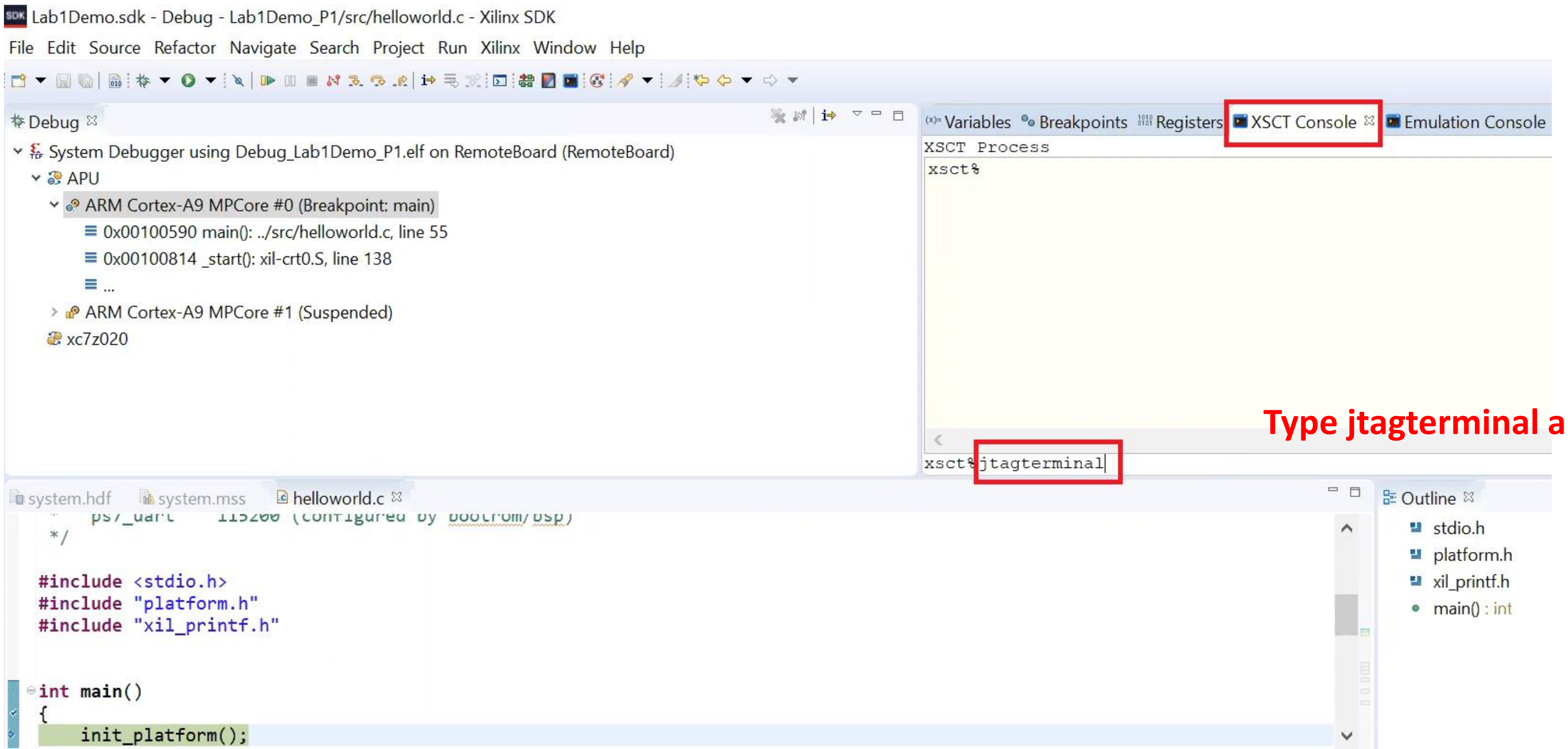
SDK Steps: 11

- After you click on Debug, you code will be downloaded on remoted board and program starts executing on ARM Core0.
- In debug mode, it will stop at first line of the main function



SDK Steps: 11

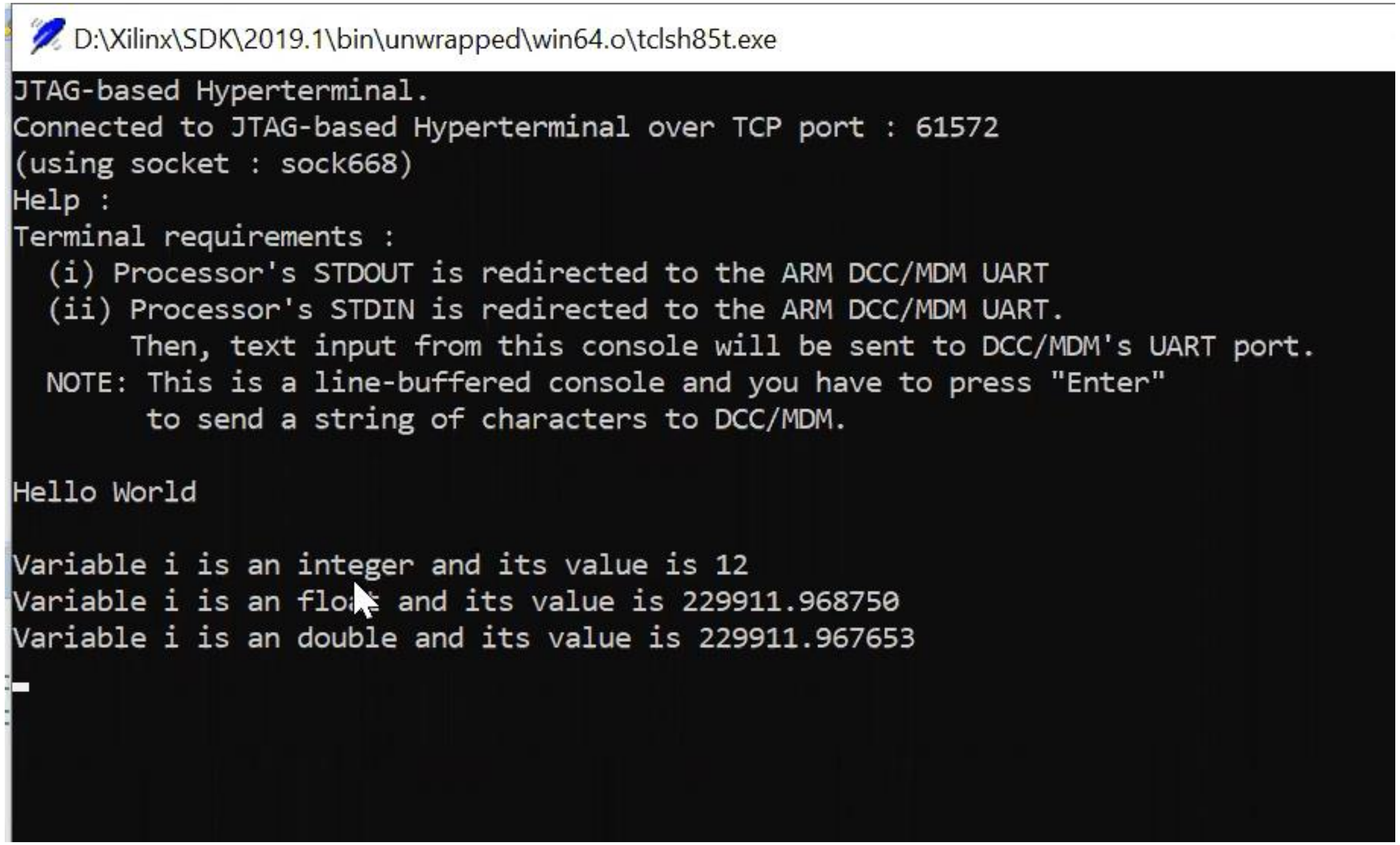
- Open Jtagterminal to see UART messages and run the code



Type jtagterminal and Press Enter

SDK Steps: 12

- Verify the print statements on the JTAG terminal
- Notice the difference in the values of floating point numbers
- Execute rest of the code



Homework

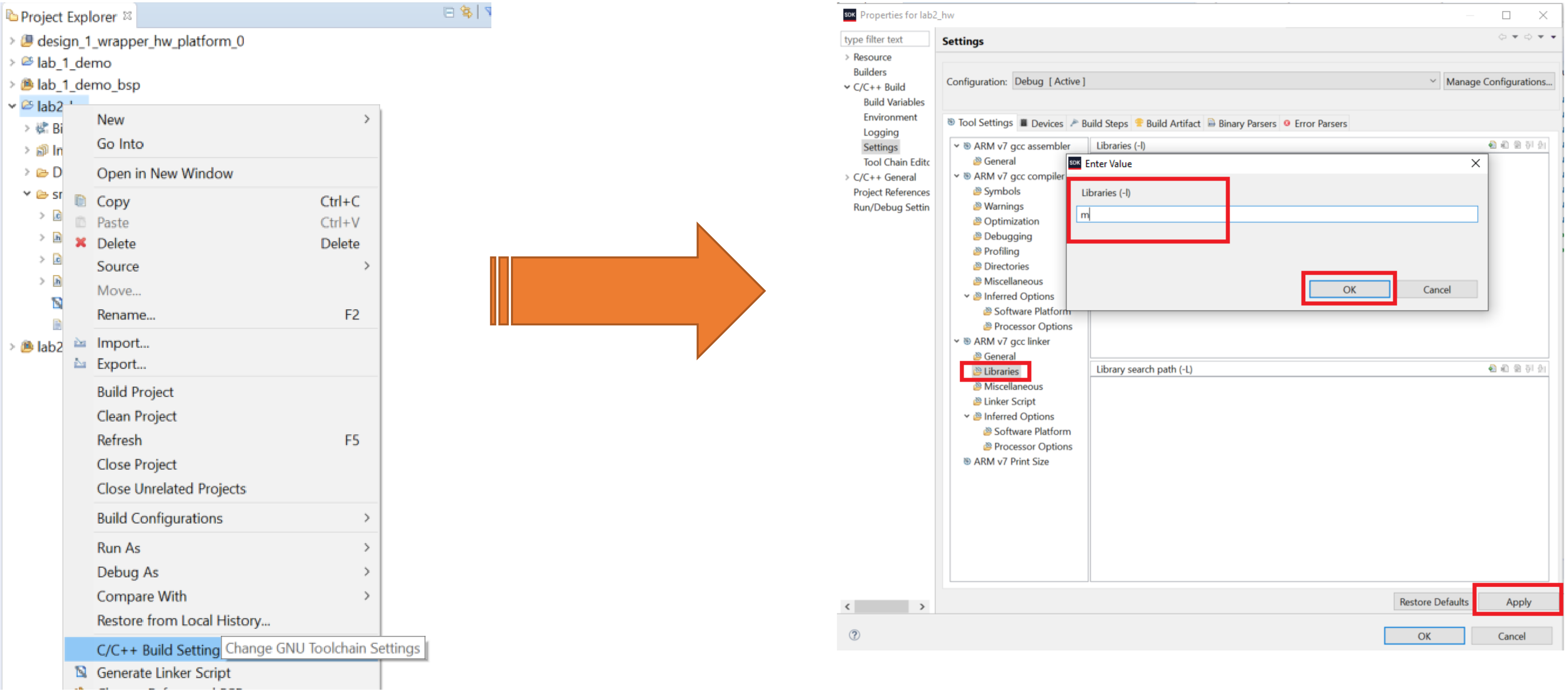
- Realize the following function on PS:

$$Q = \frac{X}{T} + \sqrt{\frac{2 * N}{T}}$$

  - **Assume X,T and N are vectors of size 3 and take these values from user. Show the output on UART.**
  - Do explore various arithmetic functions and on your own.

Math Library

- If math.h is not available, add it using following steps:



SDK Debugging

Check the Handout