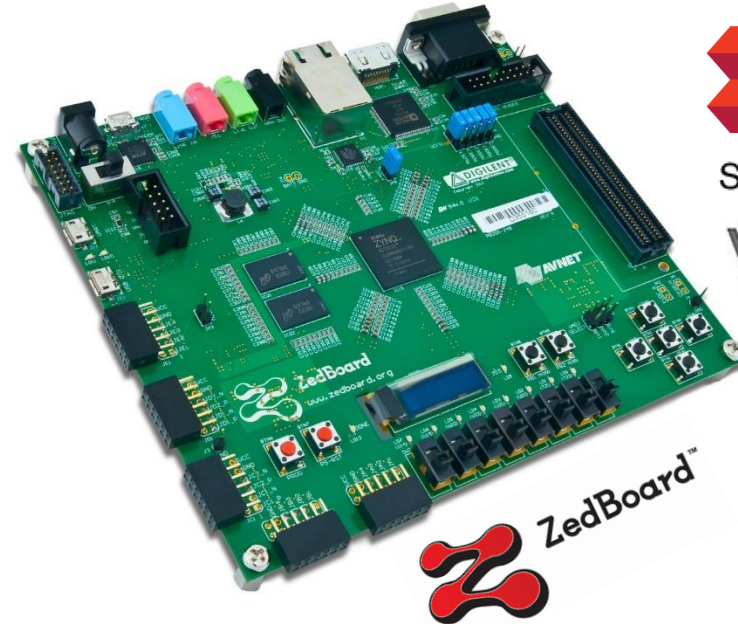




# ECE573

## Advanced Embedded Logic Design (AELD)

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.

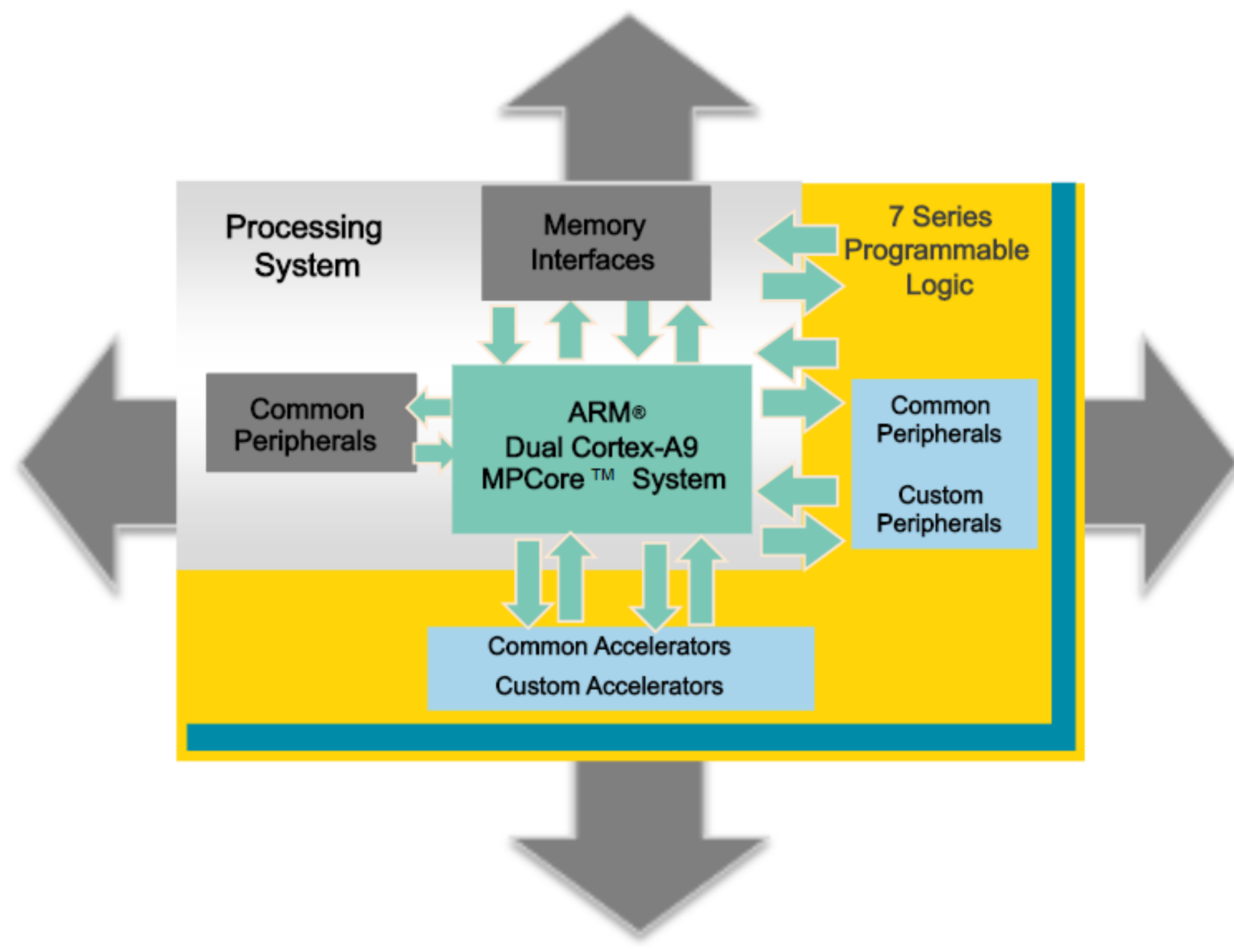
# Lab 1

- 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

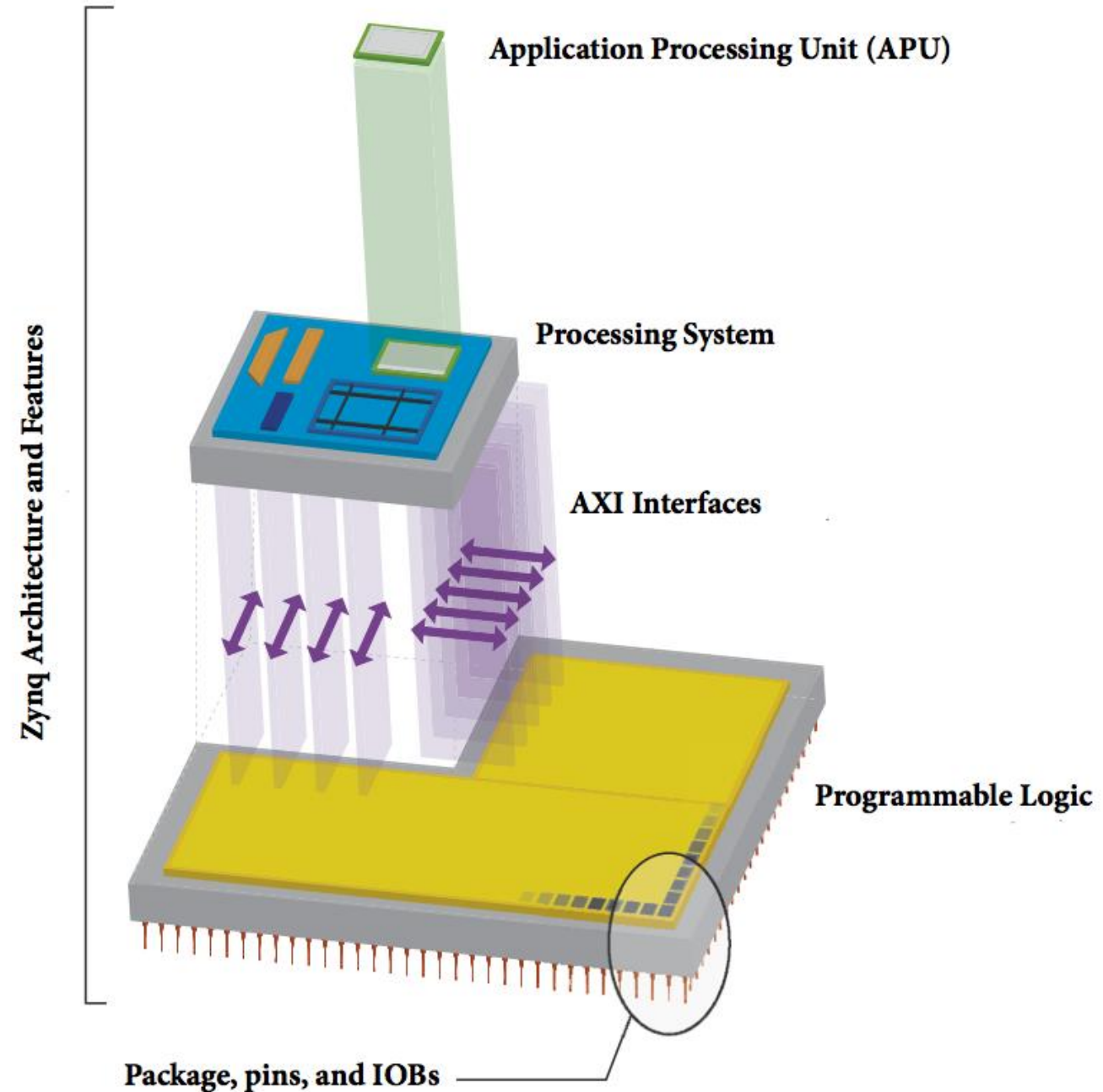
- Vivado 2019.1 with SDK (Check final video for 2021.2)
- Access to local or remote Zedboard (Digilent)
- Add board files in your Vivado catalogue. Please follow this link:  
<https://reference.digilentinc.com/learn/software/tutorials/vivado-board-files/start>

# Zynq SoC

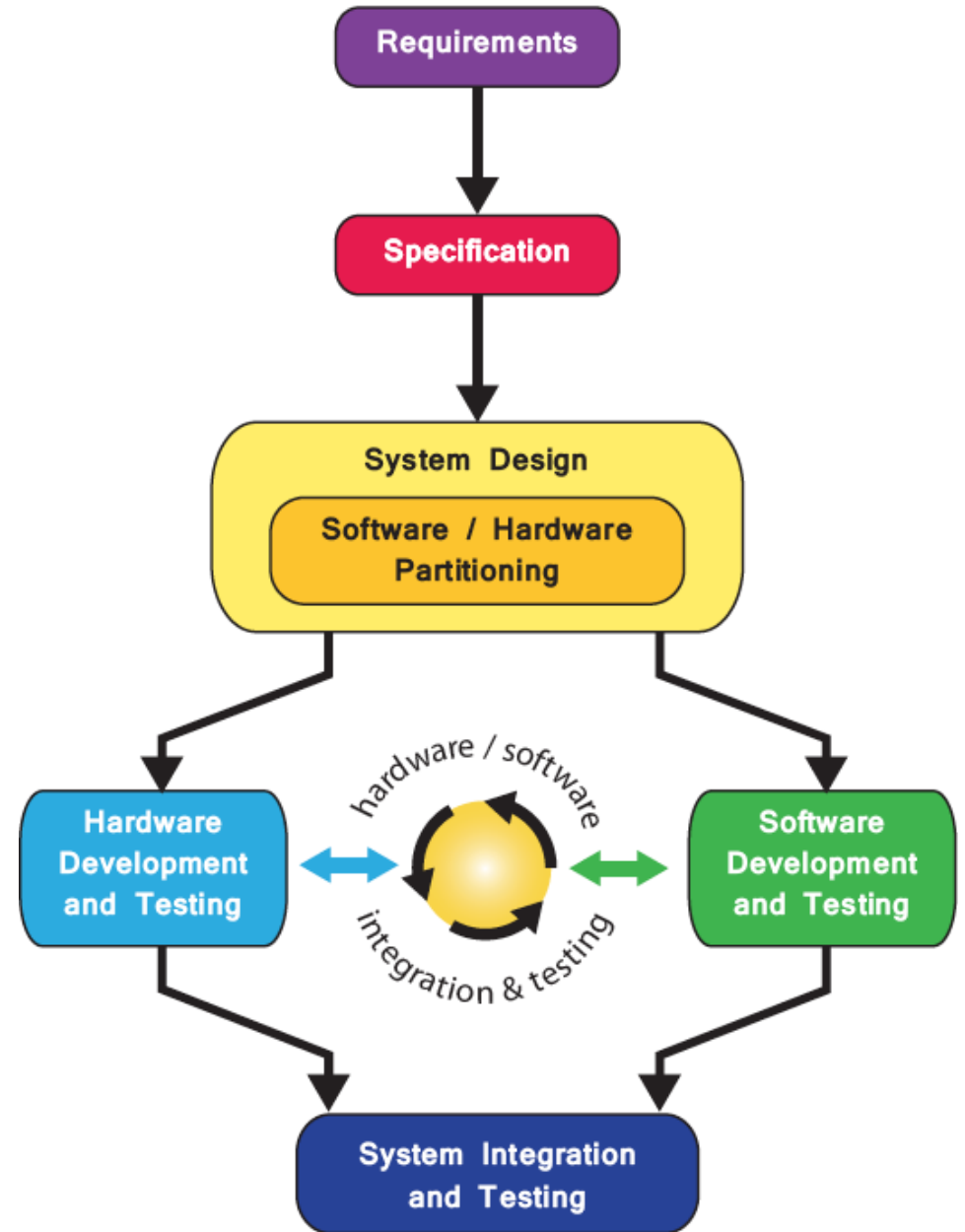


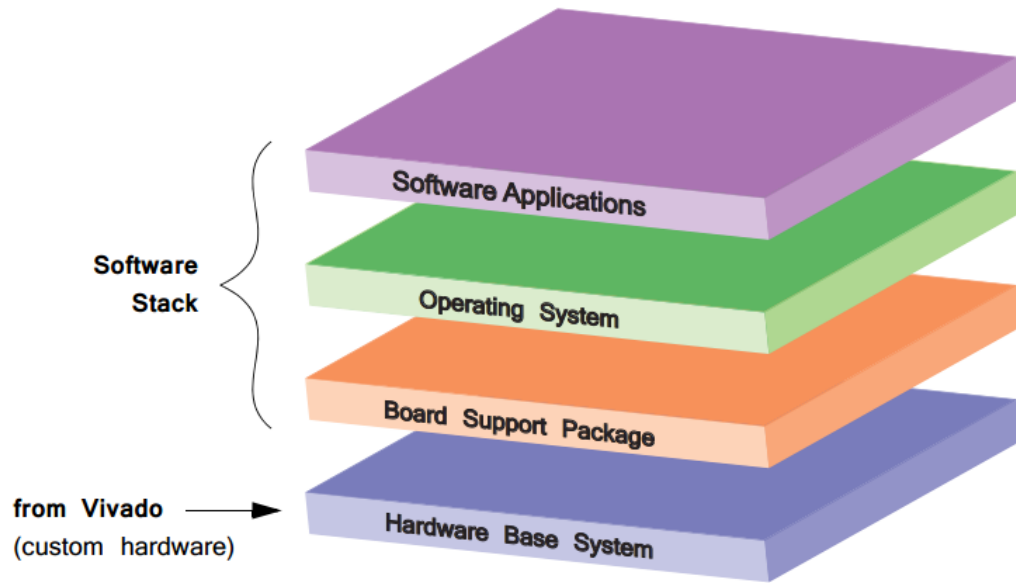
# Zynq

- ❖ 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

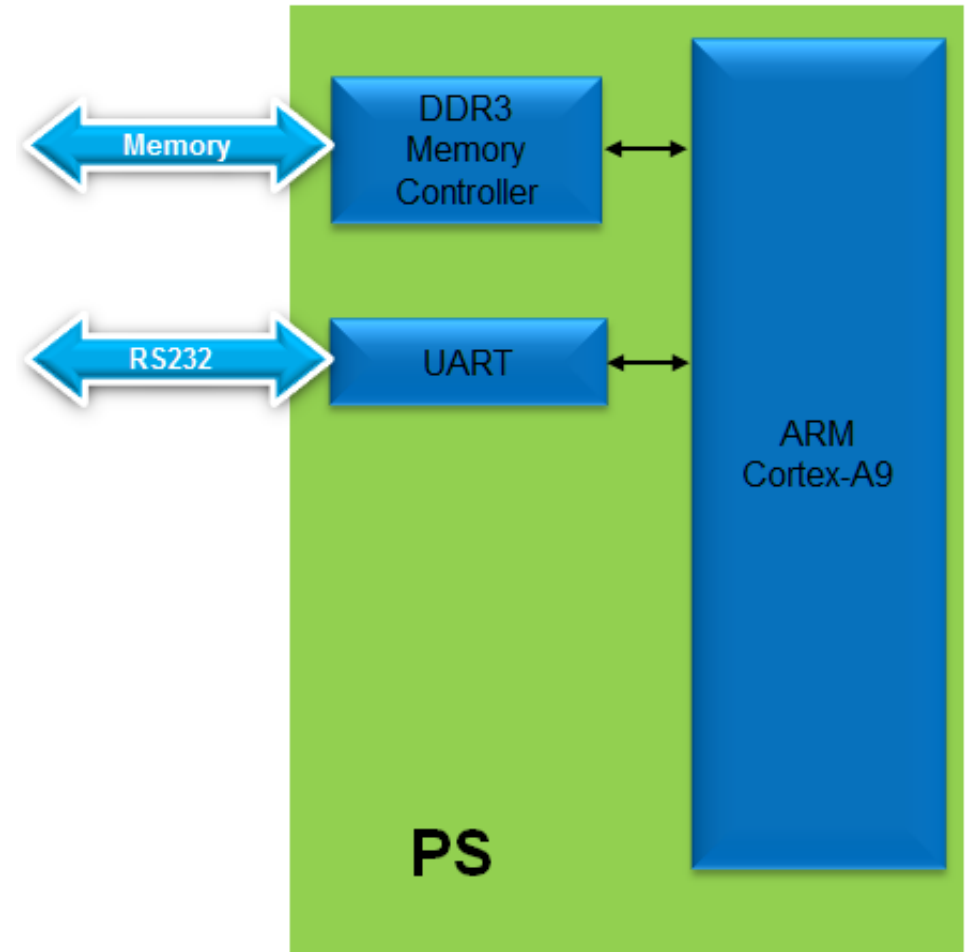
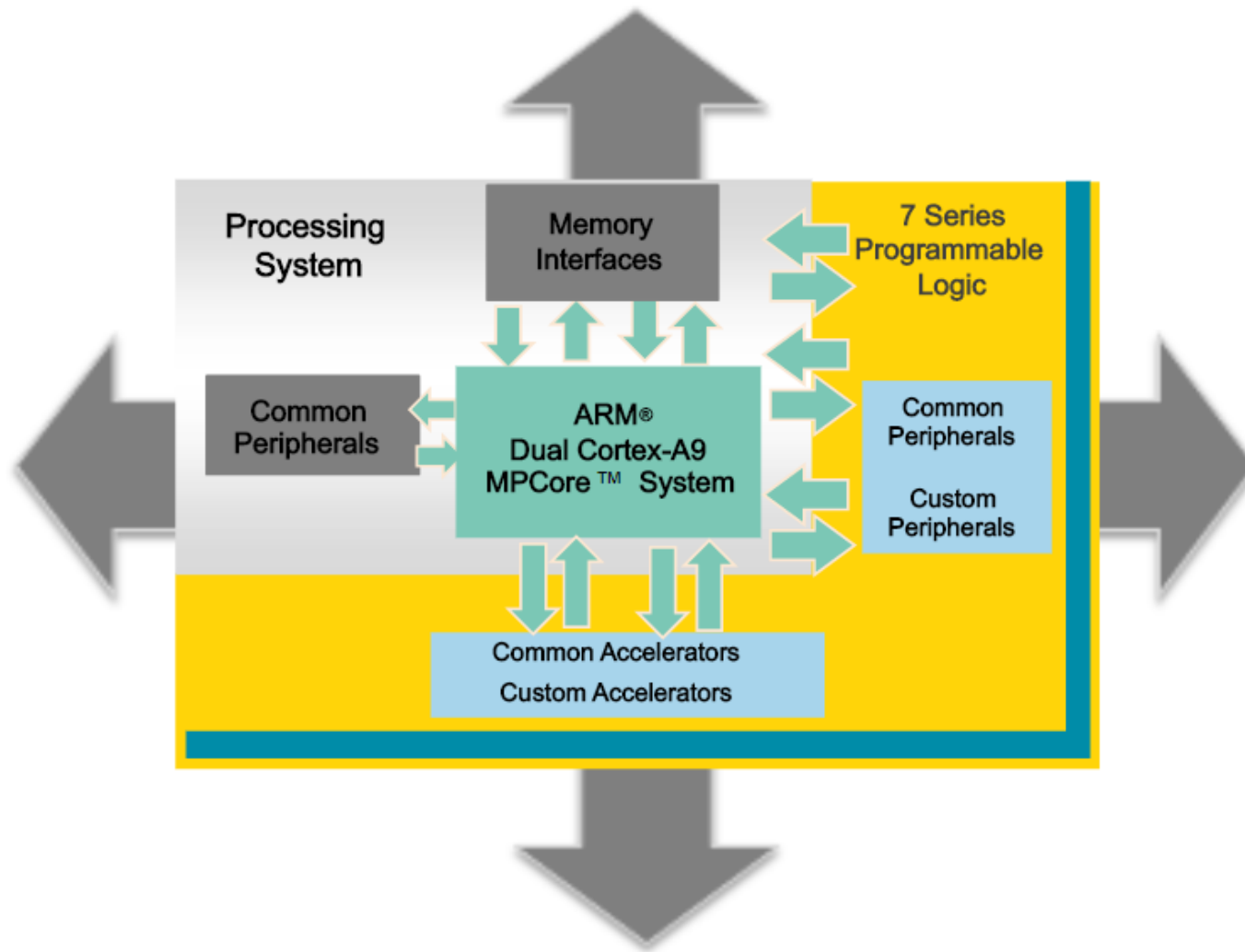




- **Board Support Package (BSP):** Set of low-level drivers and functions that are used by the next layer up (the *Operating System or baremetal*) to **communicate with the hardware**
- **Software Applications** run on top of the Operating System — these collectively represent the uppermost layer in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



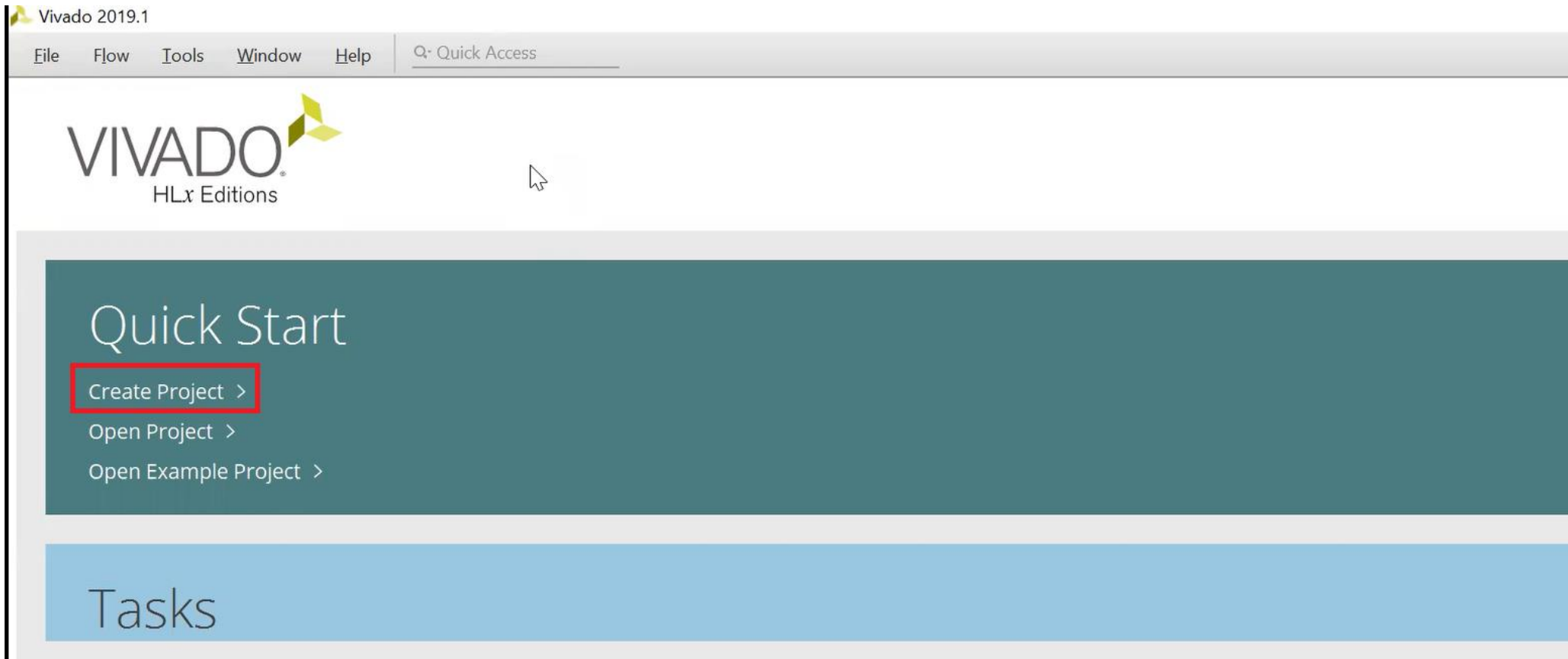


# Lab 1: Introduction to Vivado and SDK

# Vivado Design Flow

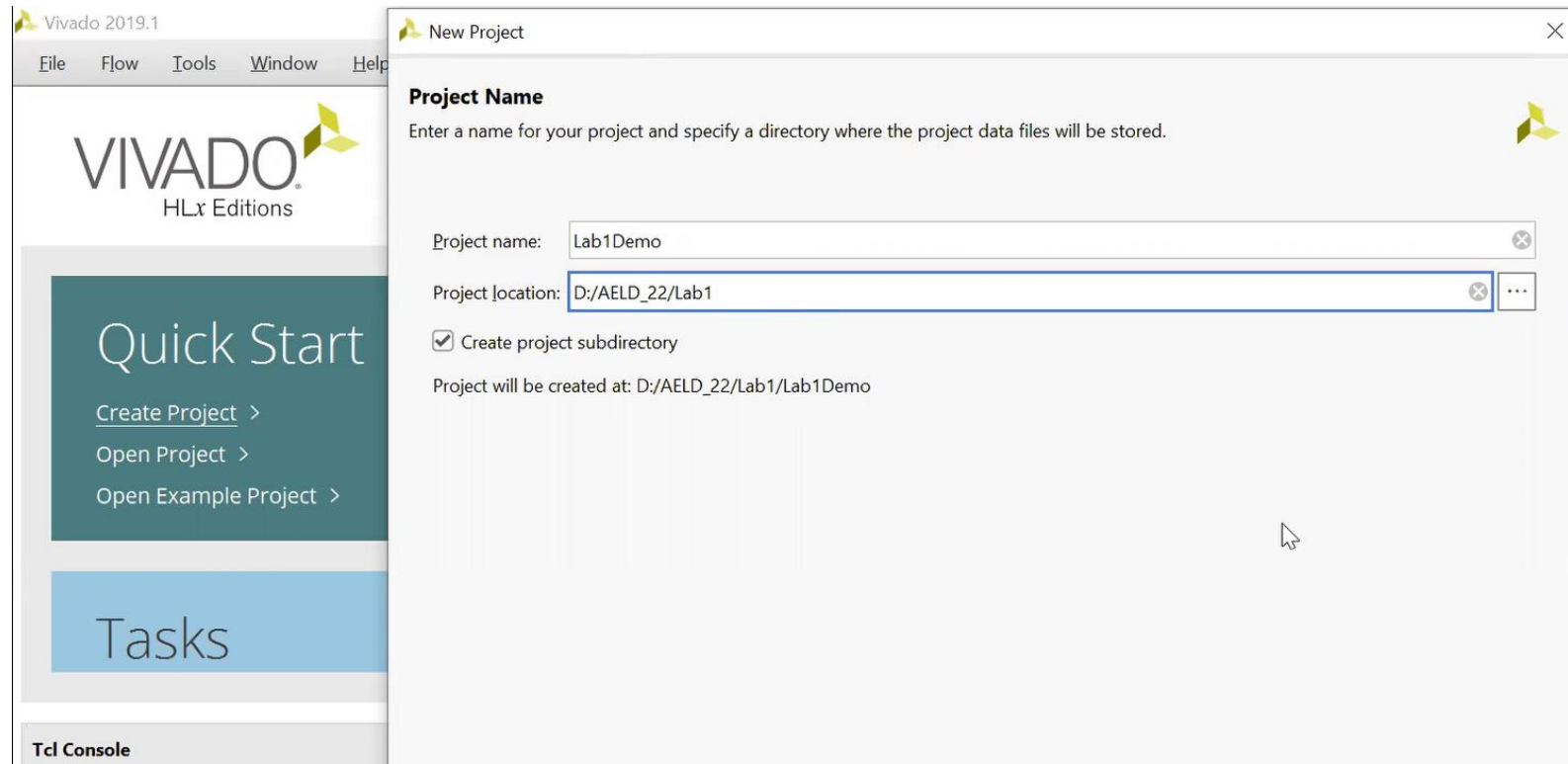
# Vivado Steps: 1

- 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 2019.1

File Flow Tools Window Help

**VIVADO**  
HLx Editions

**Quick Start**

- Create Project >
- Open Project >
- Open Example Project >

**Tasks**

**New Project**



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

Parts | **Boards**

[Reset All Filters](#) [Update Board Repositories](#)

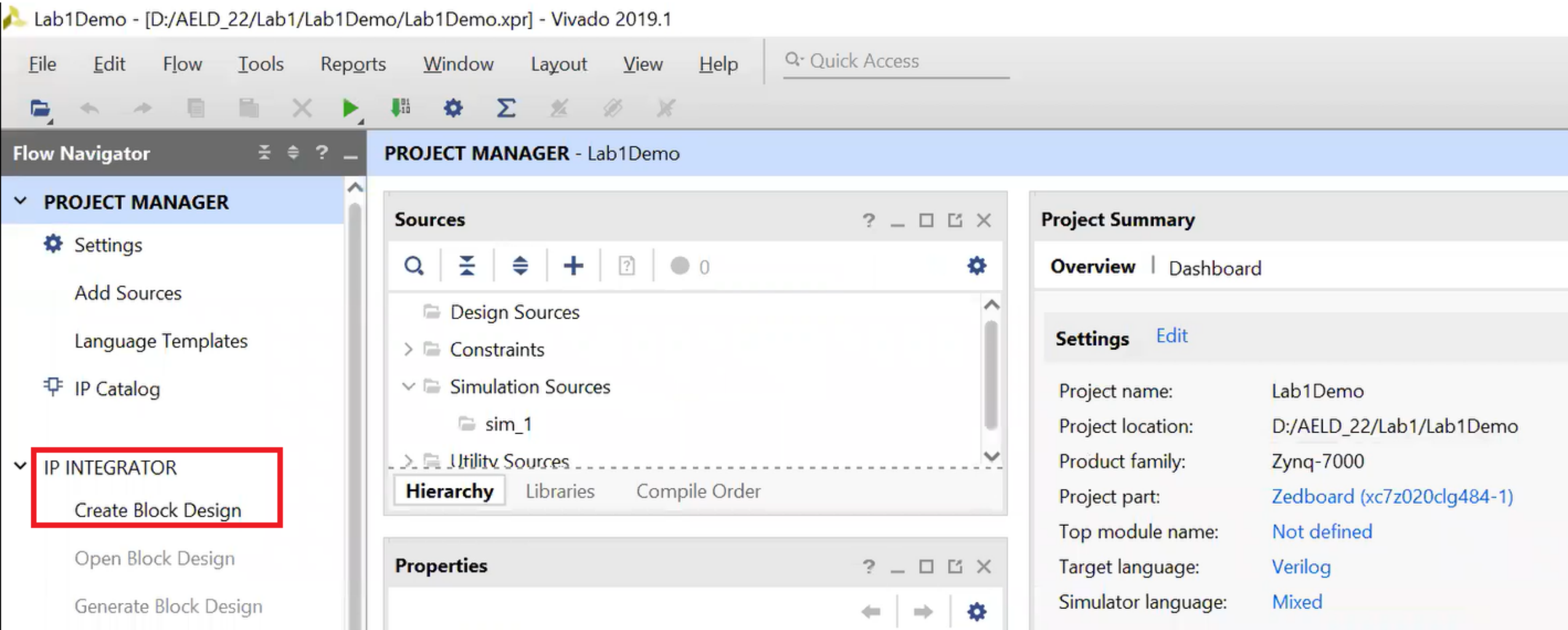
Vendor: All Name: All Board Rev: Latest

Search:  (2 matches)

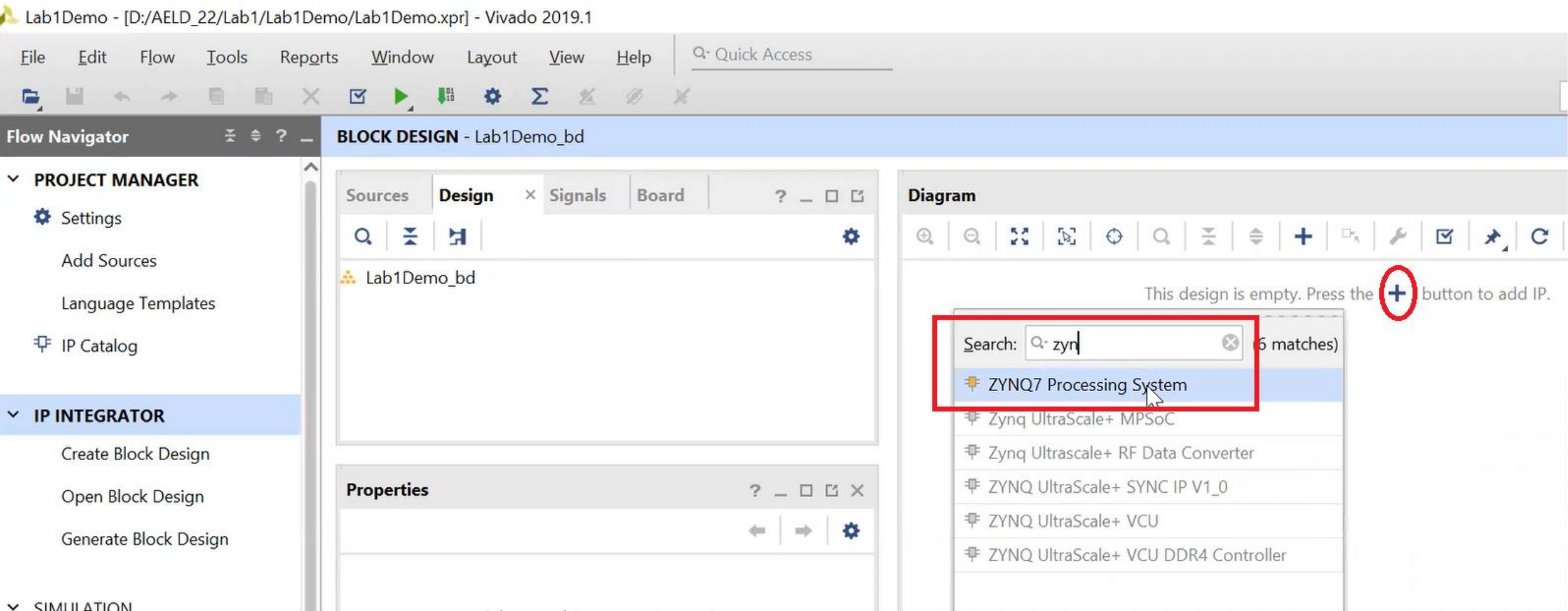
Display Name	Preview	Vendor	File Version	Part
Zedboard		digilentinc.com	1.0	xc7z020clg484-1
ZedBoard Zynq Evaluation and Development Kit		em.avnet.com	1.4	xc7z020clg484-1

Add Daughter Card [Connections](#)

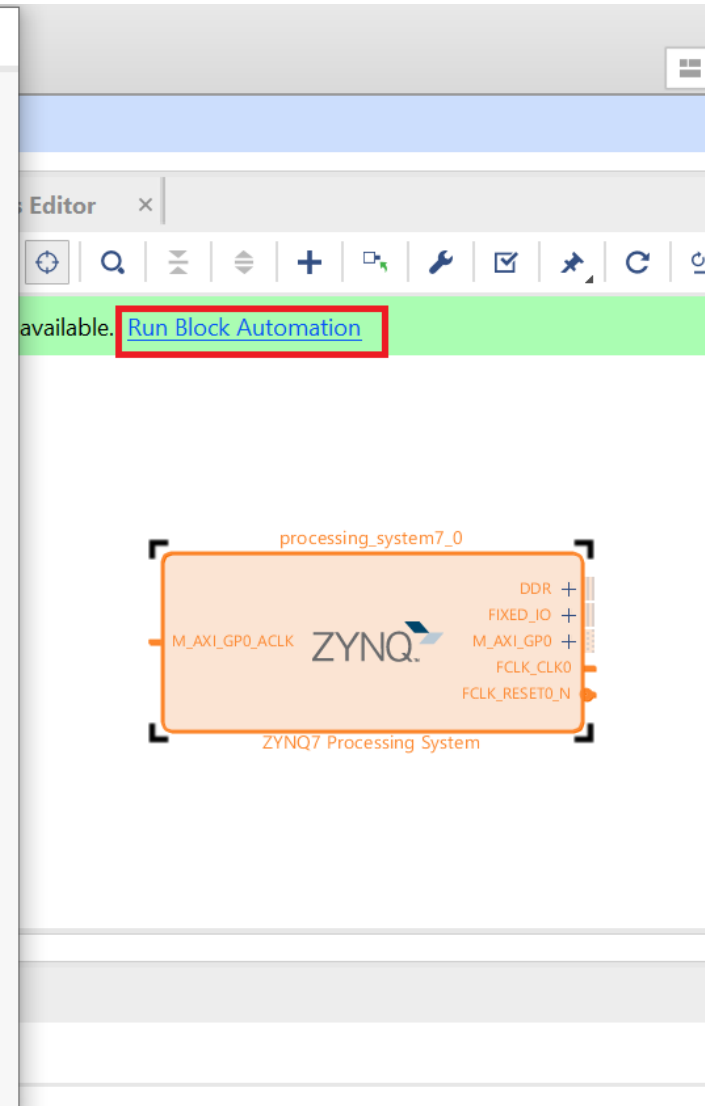
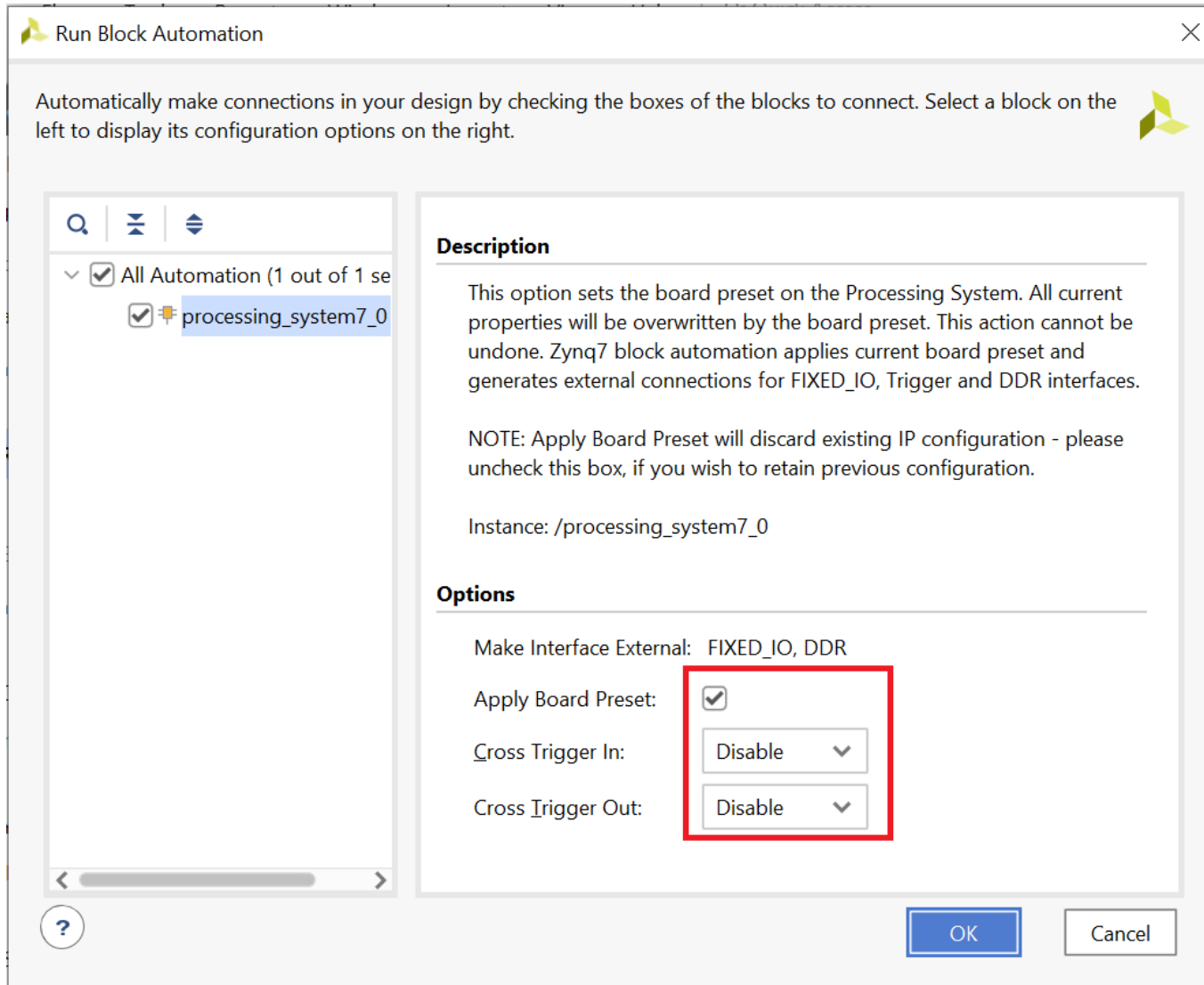
# Vivado Steps: 4



# Vivado Steps: 5



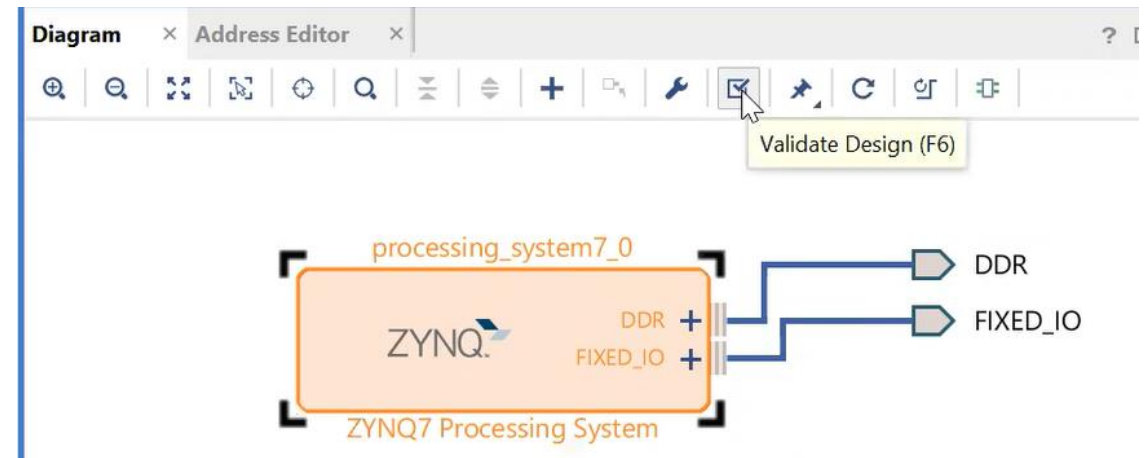
# Vivado Steps: 6





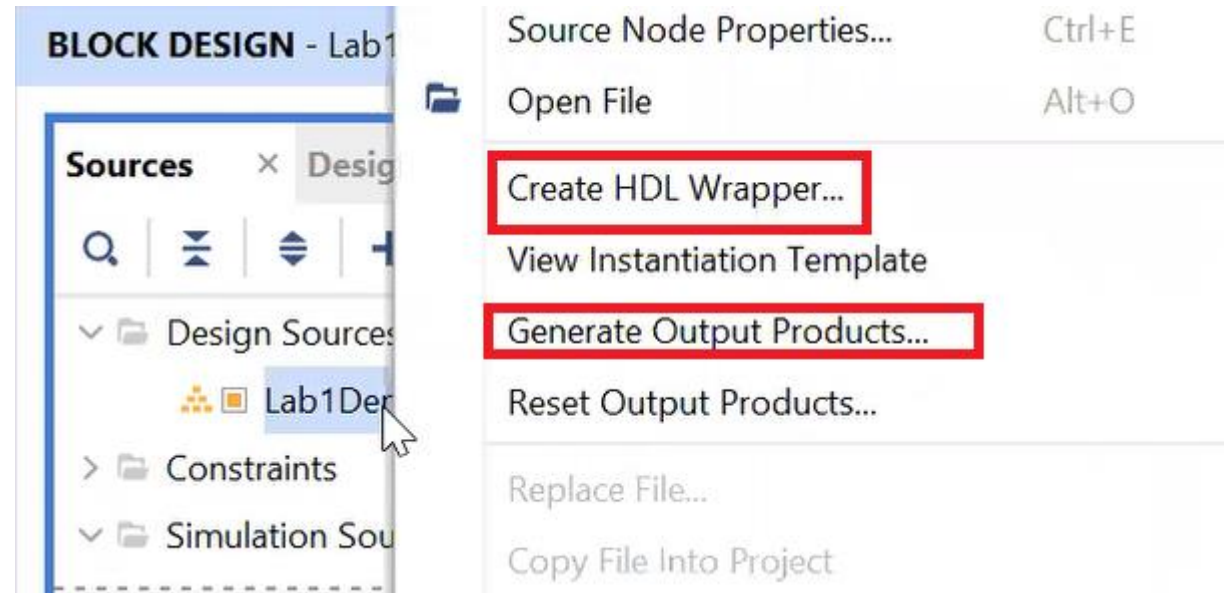
# Vivado Steps: 7

- 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

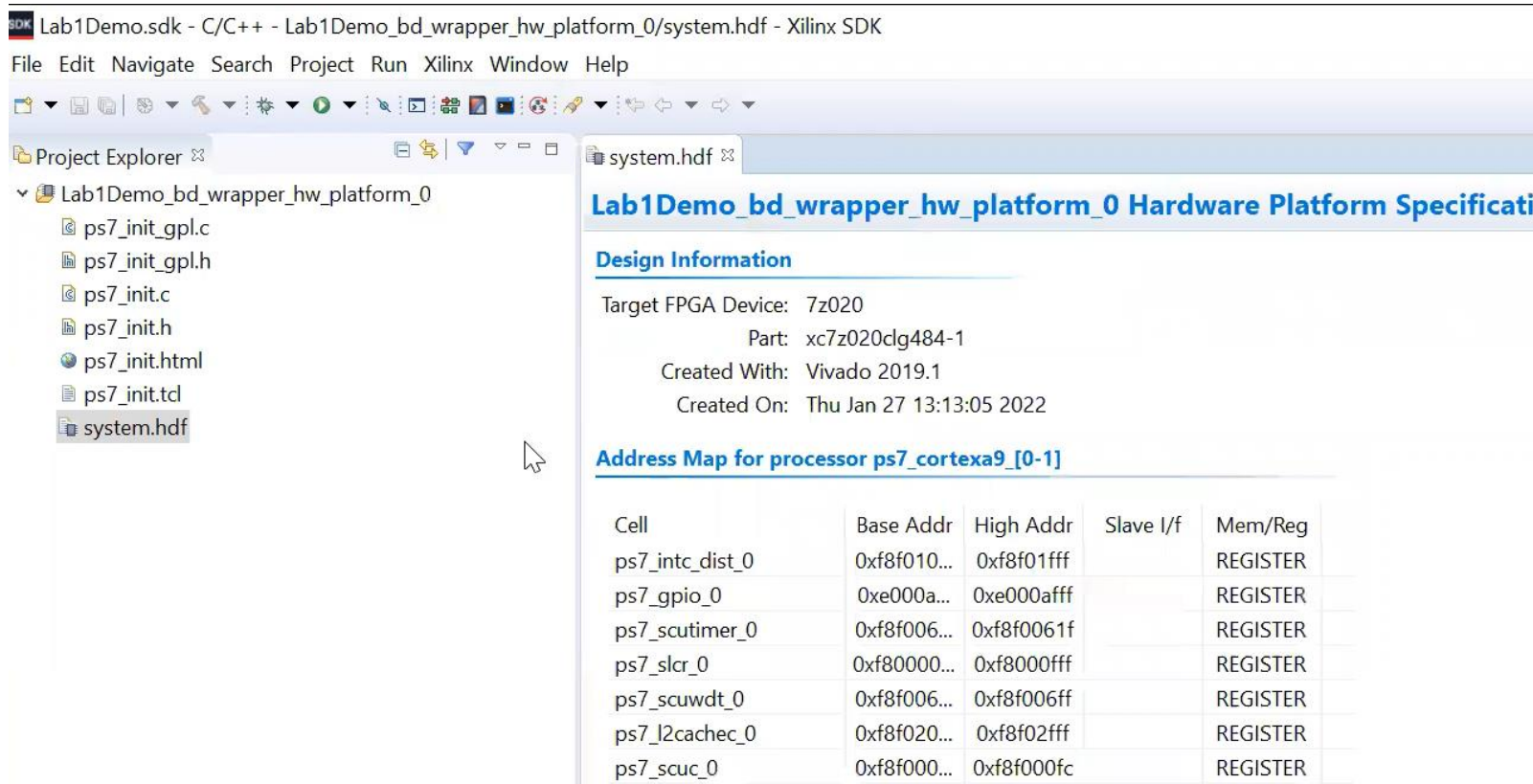
- Generate output products
- Create HDL wrapper
- File -> Export Design
- File -> Launch SDK



# SDK Design Flow

# SDK Steps: 1

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



The screenshot displays the Xilinx SDK environment. The title bar indicates the project is 'Lab1Demo.sdk - C/C++ - Lab1Demo\_bd\_wrapper\_hw\_platform\_0/system.hdf - Xilinx SDK'. The menu bar includes File, Edit, Navigate, Search, Project, Run, Xilinx, Window, and Help. The Project Explorer on the left shows the project structure for 'Lab1Demo\_bd\_wrapper\_hw\_platform\_0', listing files: ps7\_init\_gpl.c, ps7\_init\_gpl.h, ps7\_init.c, ps7\_init.h, ps7\_init.html, ps7\_init.tcl, and system.hdf. The main window displays the 'system.hdf' file, showing the 'Lab1Demo\_bd\_wrapper\_hw\_platform\_0 Hardware Platform Specification'. Under the 'Design Information' section, the following details are provided:

- Target FPGA Device: 7z020
- Part: xc7z020clg484-1
- Created With: Vivado 2019.1
- Created On: Thu Jan 27 13:13:05 2022

Below this, the 'Address Map for processor ps7\_cortexa9 [0-1]' is shown as a table:

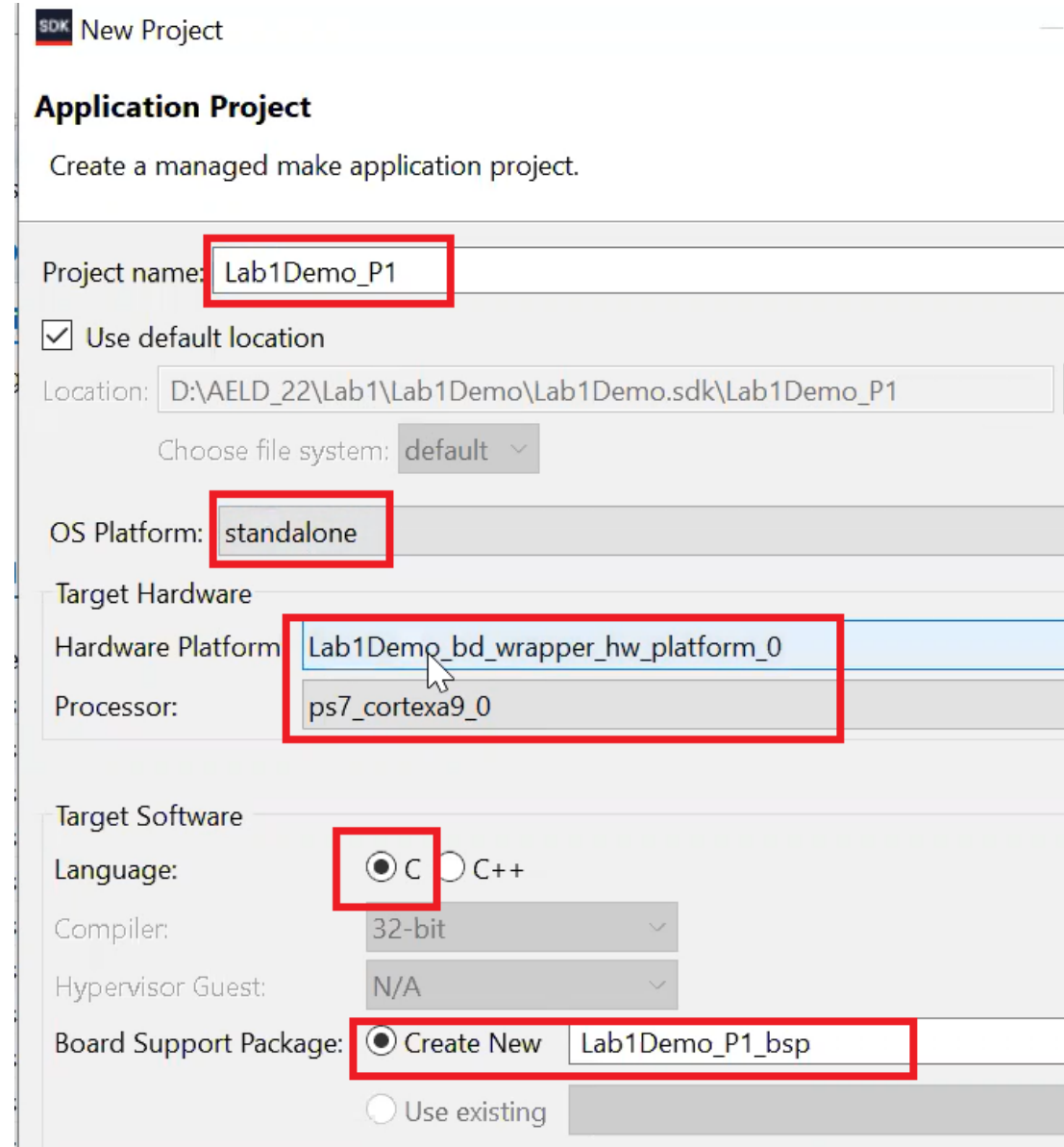
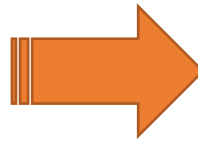
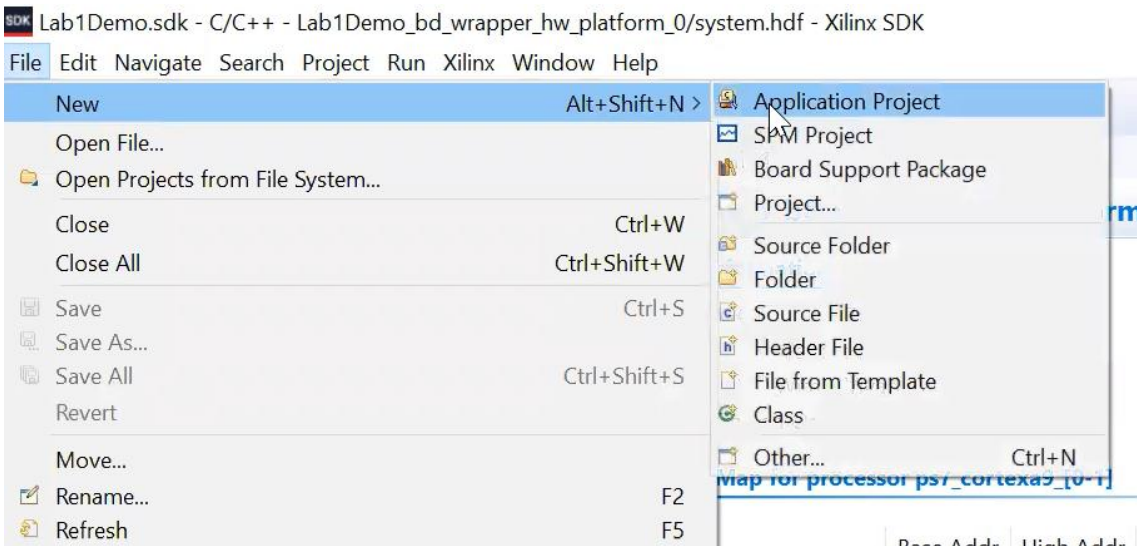
Cell	Base Addr	High Addr	Slave I/f	Mem/Reg
ps7_intc_dist_0	0xf8f010...	0xf8f01fff		REGISTER
ps7_gpio_0	0xe000a...	0xe000afff		REGISTER
ps7_scutimer_0	0xf8f006...	0xf8f0061f		REGISTER
ps7_slcr_0	0xf80000...	0xf8000fff		REGISTER
ps7_scuwdt_0	0xf8f006...	0xf8f006ff		REGISTER
ps7_l2cachec_0	0xf8f020...	0xf8f02fff		REGISTER
ps7_scuc_0	0xf8f000...	0xf8f000fc		REGISTER

# 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.

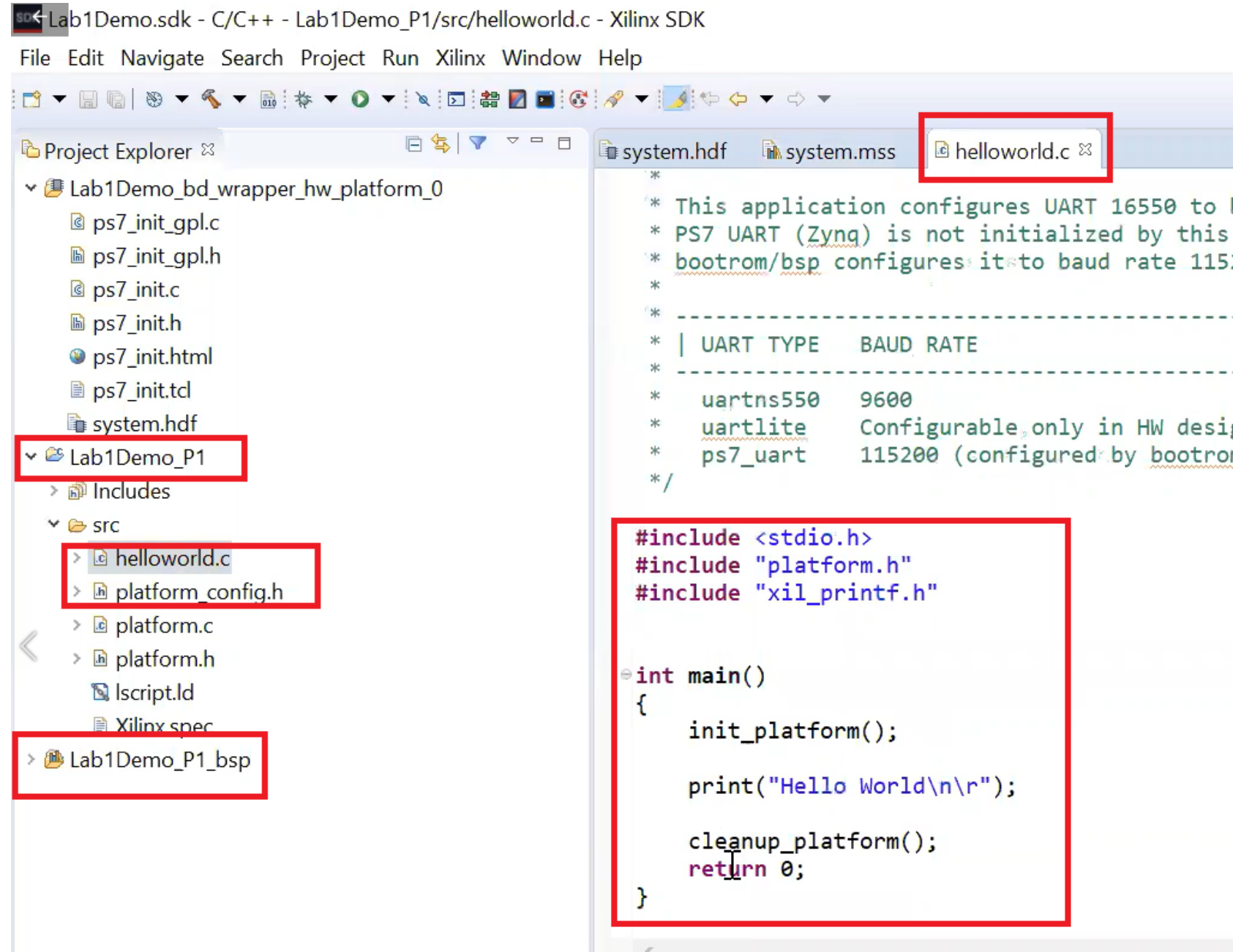
# SDK Steps: 2

- 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)

```
#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

Target Connection Details

**New Target Connection**

Creates new configuration for connecting to a target.

Target Name RemoteBoard

☒ Set as default target

**Specify the connection type and properties**

Type Hardware Server

Host 192.168.33.163

Port 3121

☒ Use Symbol Server

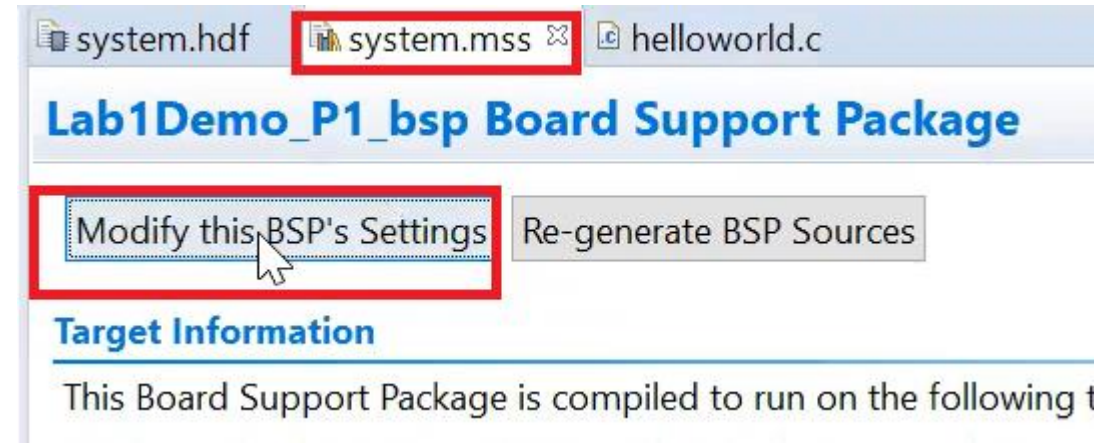
Note: Use Symbol server for source level debugging on remote machine.

Advanced >>

Test Connection OK

# SDK Steps: 7

- Modify BSP settings



## Board Support Package Settings

Control various settings of your Board Support Package.



▼ Overview

standalone

▼ drivers

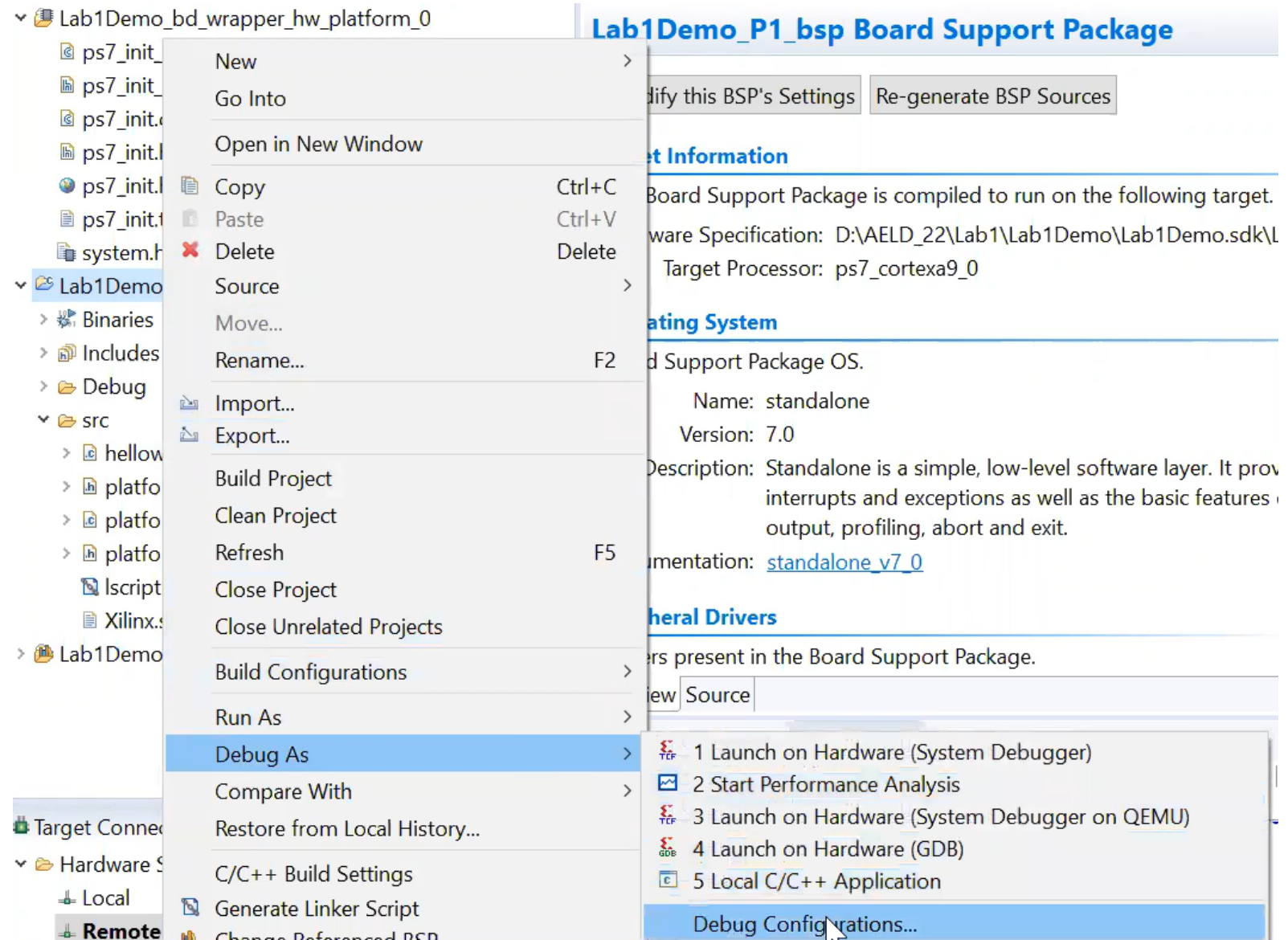
ps7\_cortexa9\_0

Configuration for OS: standalone

Name	Value	Default	Type	Description
hypervisor_guest	false	false	boolean	Enable hypervisor guest
lockstep_mode_debug	false	false	boolean	Enable debug logic in
sleep_timer	none	none	peripheral	This parameter is used
stdin	ps7_coresight_c...	none	peripheral	stdin peripheral
stdout	ps7_coresight	none	peripheral	stdout peripheral
ttc_select_cntr	2	2	enum	Selects the counter to
zynqmp_fsbl_bsp	false	false	boolean	Disable or Enable Opt
> microblaze_exceptions	false	false	boolean	Enable MicroBlaze Exc
> enable_sw_intrusive_pr	false	false	boolean	Enable S/W Intrusive

# SDK Steps: 8

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



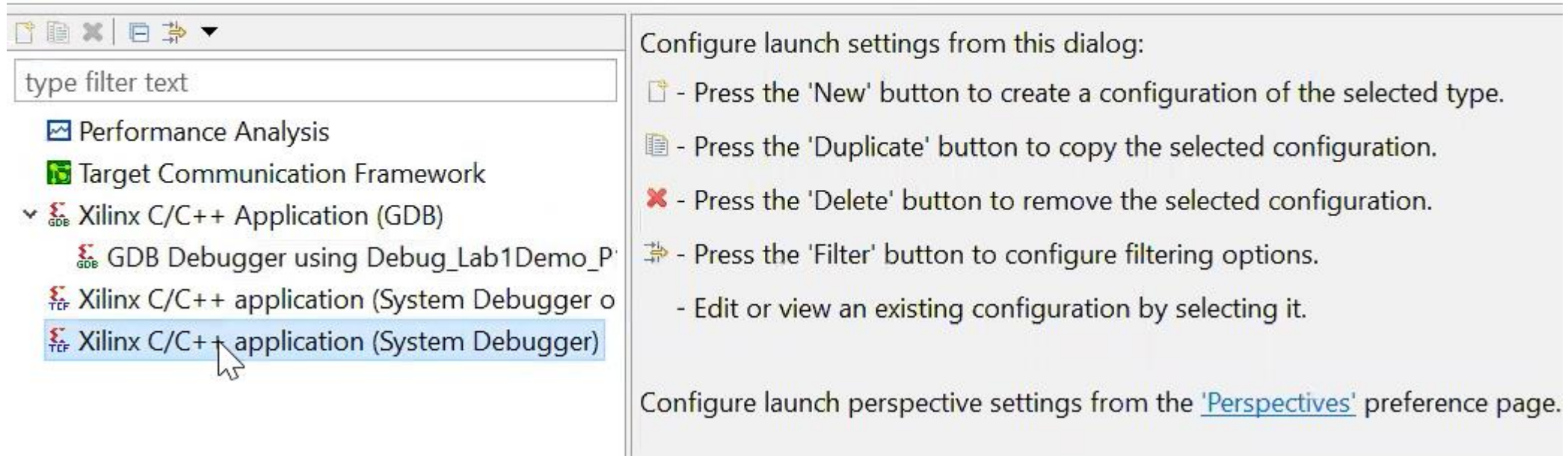


# SDK Steps: 9





- Double click on System Debugger option

## Create, manage, and run configurations

Run or Debug a program using System Debugger.



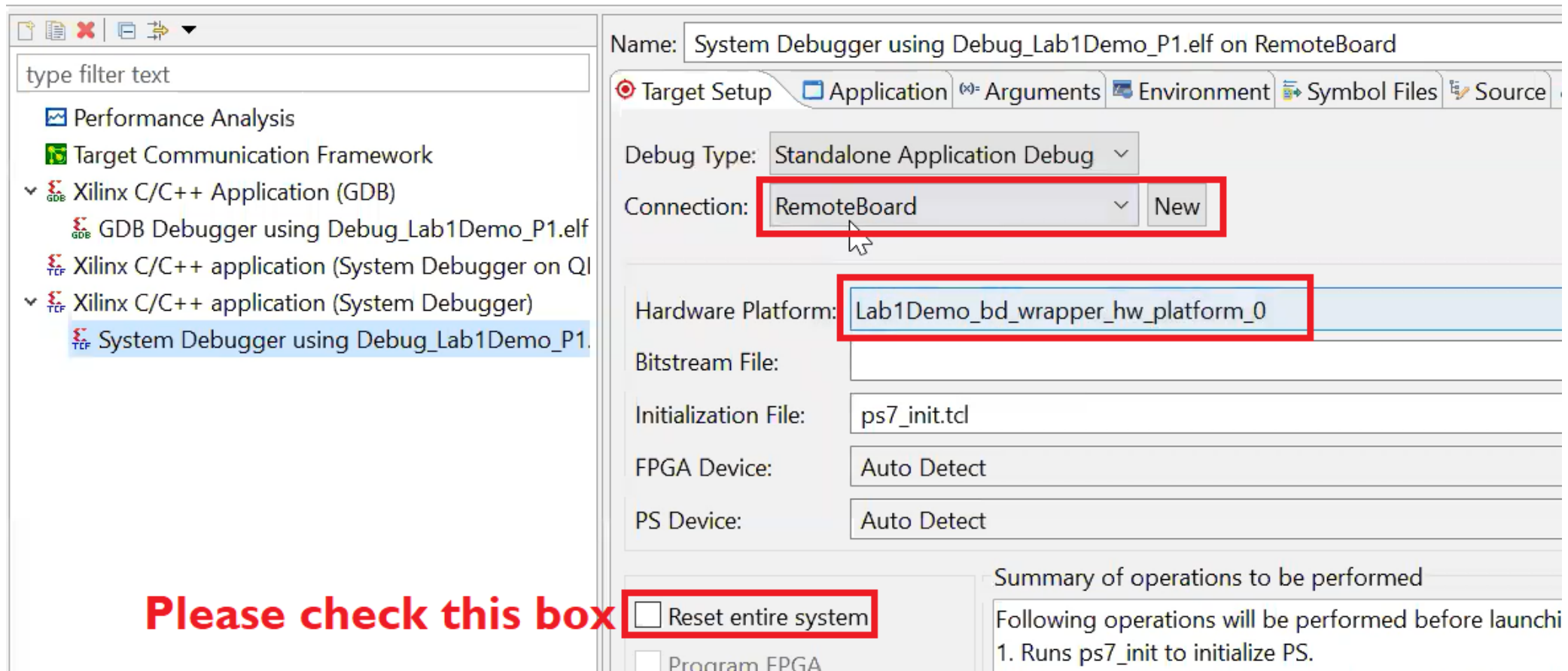
Configure launch settings from this dialog:

-  - Press the 'New' button to create a configuration of the selected type.
-  - Press the 'Duplicate' button to copy the selected configuration.
-  - Press the 'Delete' button to remove the selected configuration.
-  - Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the [Perspectives](#) preference page.

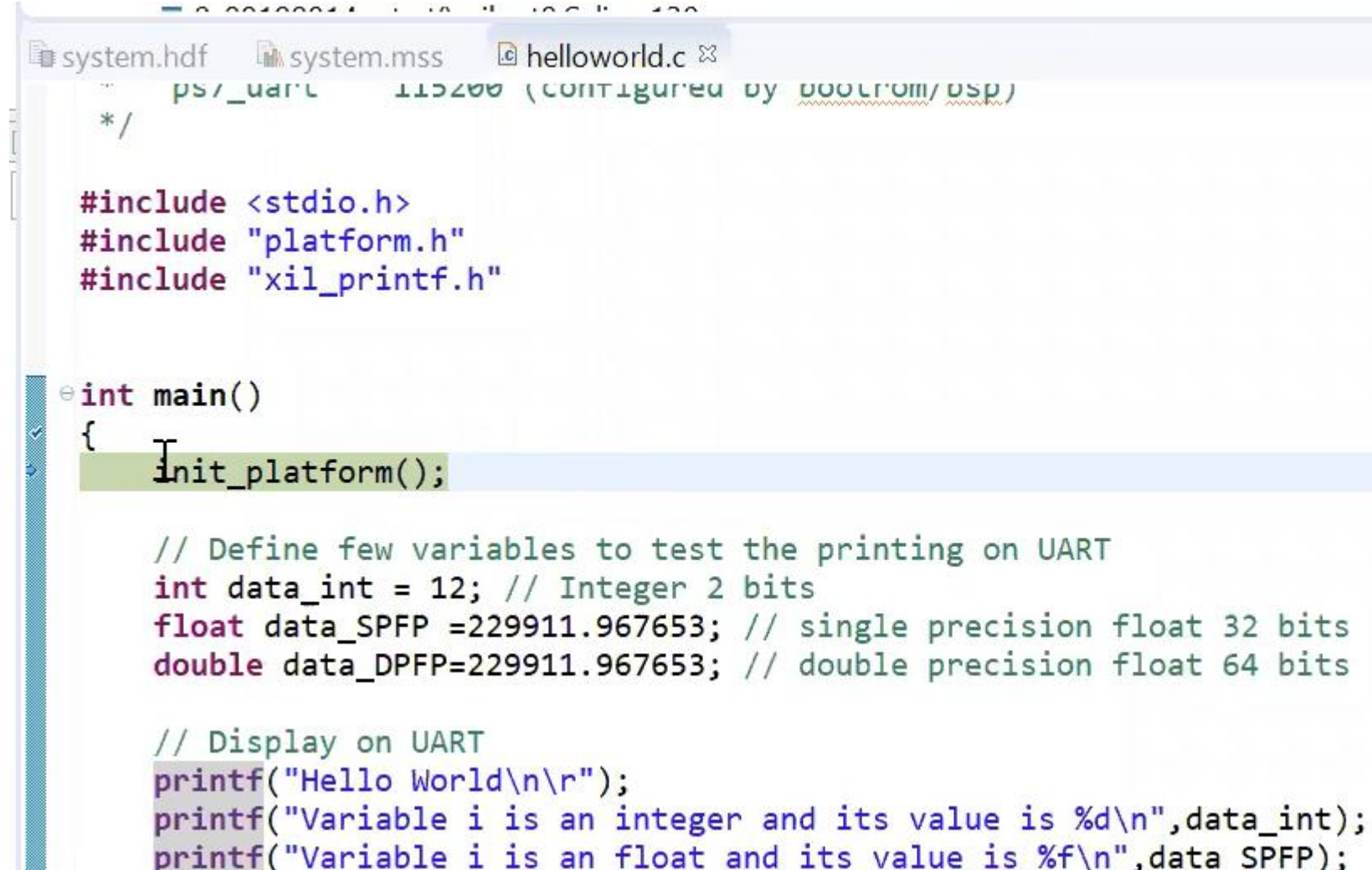
# SDK Steps: 10

- 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



```
system.hdf system.mss helloworld.c x
ps/_uart 115200 (configured by bootrom/bsp)
*/

#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"

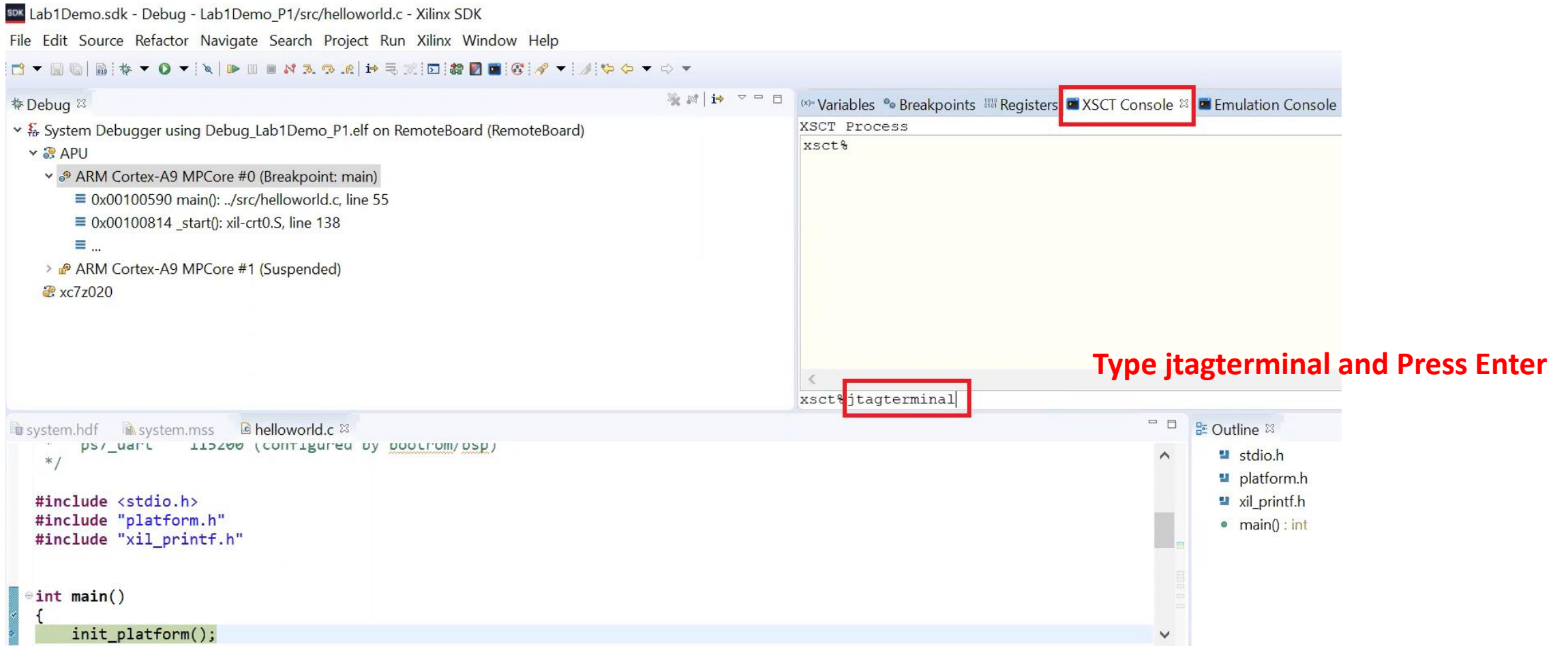
int main()
{
    init_platform();

    // 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 i is an integer and its value is %d\n", data_int);
    printf("Variable i is an float and its value is %f\n", data_SPFP);
}
```

# SDK Steps: 11

- Open Jtagterminal to see UART messages and run the code



The screenshot shows the Xilinx SDK IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Xilinx, Window, and Help. The main window is divided into several panes. On the left, the 'Debug' pane shows the 'System Debugger using Debug\_Lab1Demo\_P1.elf on RemoteBoard (RemoteBoard)'. Under 'APU', there is an 'ARM Cortex-A9 MPCore #0 (Breakpoint: main)' with a list of breakpoints: '0x00100590 main(): ../src/helloworld.c, line 55', '0x00100814 \_start(): xil-crt0.S, line 138', and '...'. Below this, 'ARM Cortex-A9 MPCore #1 (Suspended)' and 'xc7z020' are listed. The right pane is titled 'XSCT Console' and shows 'XSCT Process' with 'xsct%' below it. At the bottom of this pane, the command 'xsct% jtagterminal' is entered. A red box highlights the 'XSCT Console' tab, and another red box highlights the command input area. A red text overlay on the right says 'Type jtagterminal and Press Enter'. The bottom pane shows the source code for 'helloworld.c', which includes 'stdio.h', 'platform.h', and 'xil\_printf.h'. The 'main()' function is shown, starting with 'init\_platform();'.

Lab1Demo.sdk - Debug - Lab1Demo\_P1/src/helloworld.c - Xilinx SDK

File Edit Source Refactor Navigate Search Project Run Xilinx Window Help

Debug

System Debugger using Debug\_Lab1Demo\_P1.elf on RemoteBoard (RemoteBoard)

APU

ARM Cortex-A9 MPCore #0 (Breakpoint: main)

- 0x00100590 main(): ../src/helloworld.c, line 55
- 0x00100814 \_start(): xil-crt0.S, line 138
- ...

ARM Cortex-A9 MPCore #1 (Suspended)

xc7z020

XSCT Console

XSCT Process

xsct%

xsct% jtagterminal

Type jtagterminal and Press Enter

system.hdf system.mss helloworld.c

```
/*  
 *  
 */  
  
#include <stdio.h>  
#include "platform.h"  
#include "xil_printf.h"  
  
int main()  
{  
    init_platform();  
}
```



# 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

```
D:\Xilinx\SDK\2019.1\bin\unwrapped\win64.o\tclsh85t.exe
JTAG-based Hyperterminal.
Connected to JTAG-based Hyperterminal over TCP port : 61572
(using socket : sock668)
Help :
Terminal requirements :
  (i) Processor's STDOUT is redirected to the ARM DCC/MDM UART
  (ii) Processor's STDIN is redirected to the ARM DCC/MDM UART.
      Then, text input from this console will be sent to DCC/MDM's UART port.
      NOTE: This is a line-buffered console and you have to press "Enter"
            to send a string of characters to DCC/MDM.

Hello World

Variable i is an integer and its value is 12
Variable i is an float and its value is 229911.968750
Variable i is an double and its value is 229911.967653
```

# Homework

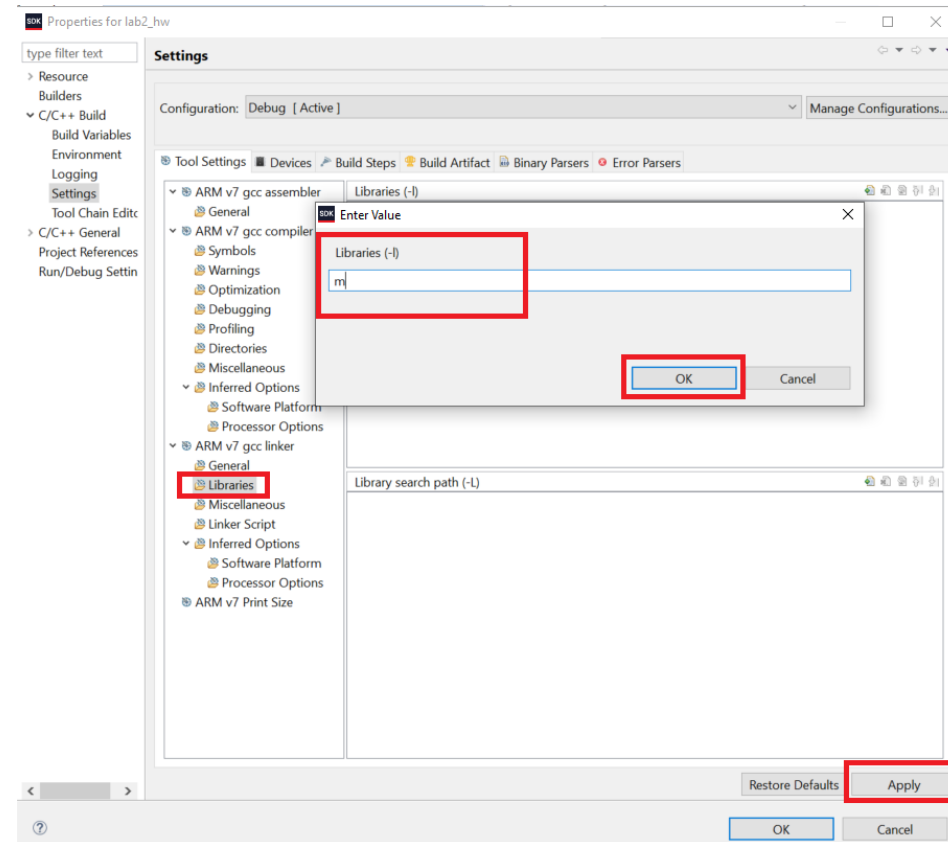
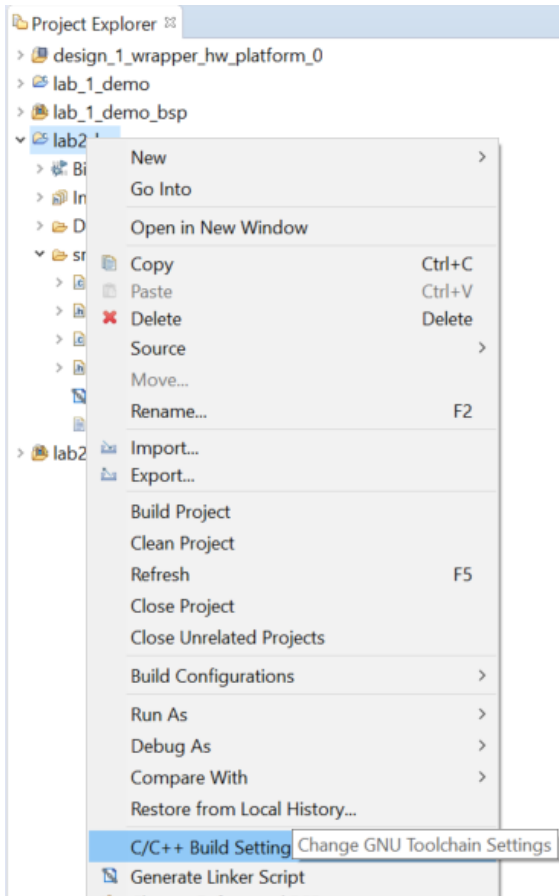
- Realize the following function on PS:

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

- **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