# Lab 9

**Tasks to be done in this Lab:**

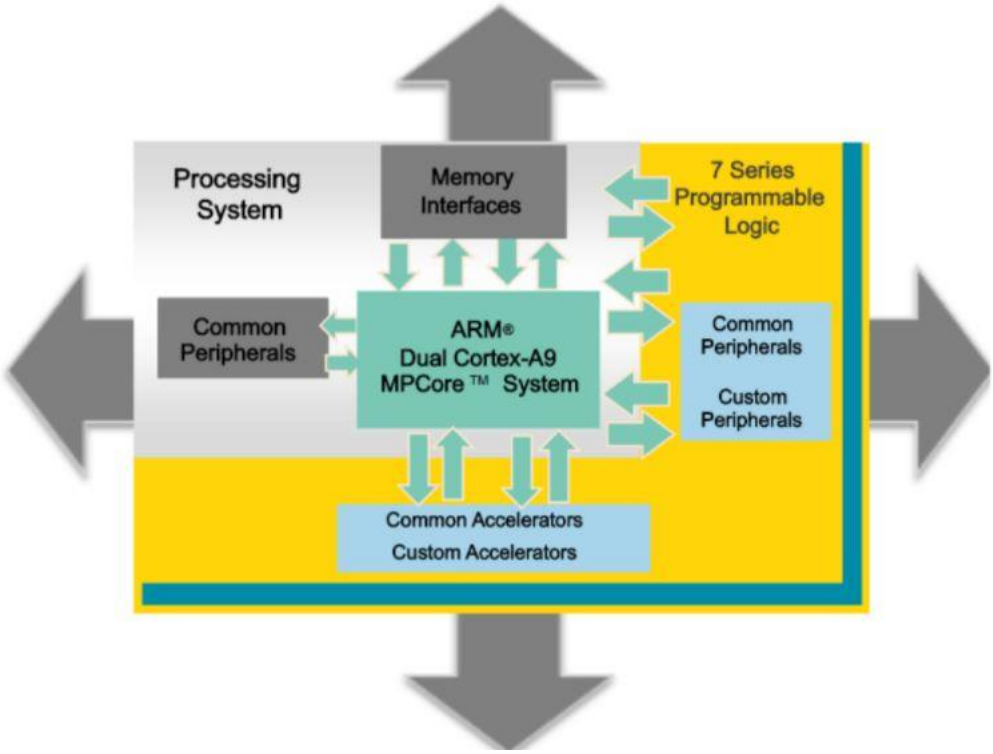
1. Create block design in Vivado using the IP Integrator and configure the Zynq IP according to our needs.
2. Create a Hello World application in Xilinx SDK for the ARM processor and learn how to display different kinds of data and take user input using JTAG Terminal.
3. Write a C application to compute the following expression.

$$X/T + SQRT(2*LogN/T)$$

**Topics to explore:** 1) Zynq Architecture, 2) Zynq Configuration, 3) Creating Project in Vivado using IP Integrator, 4) Using SDK to create C-based Application for ARM

## Part -1

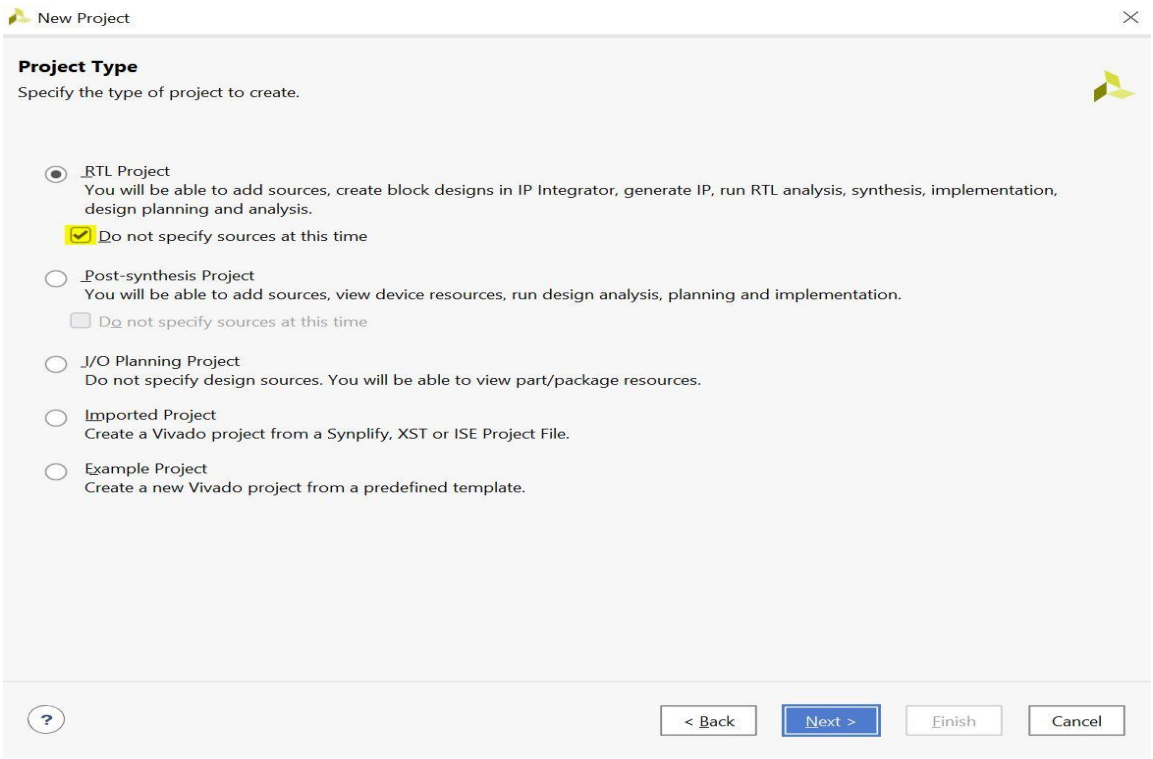
In this part, we will learn how to use IP Integrator to create a processing system-based design consisting of ARM cortex A9 cores. An abstract view of the Zynq architecture is given below. We will need the DDR3 controller for external DDR3 memory. As we are accessing the Zybo board remotely, we will be using the JTAG terminal instead of UART for the STDIN and STDO.



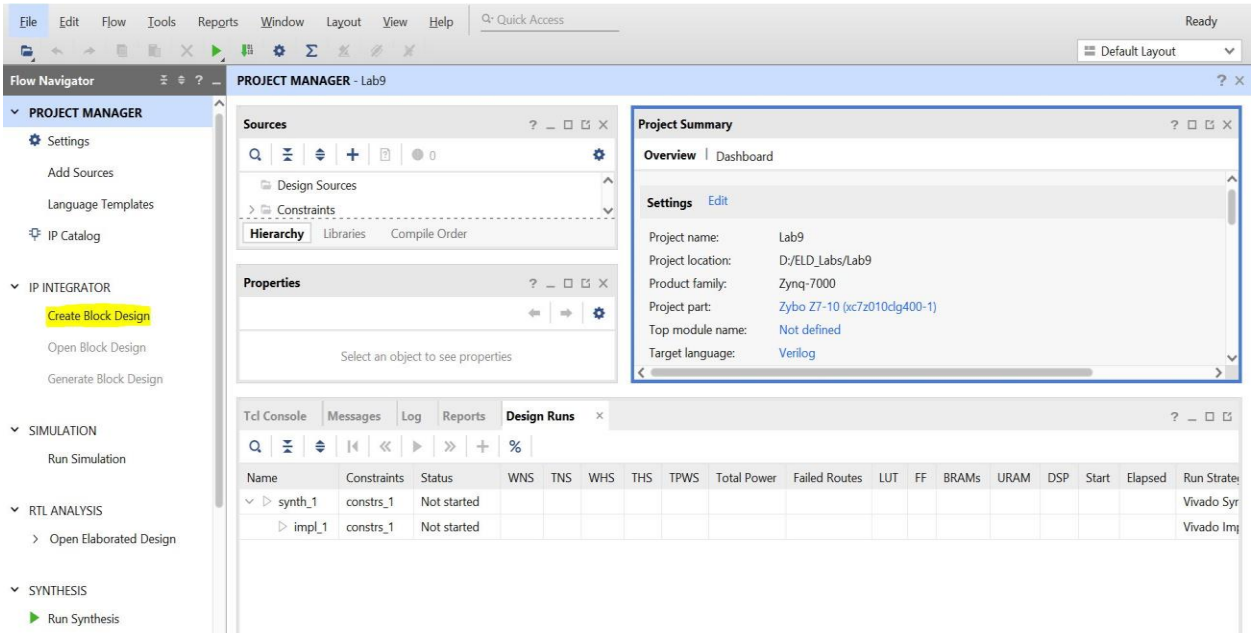

**Zynq Architecture**

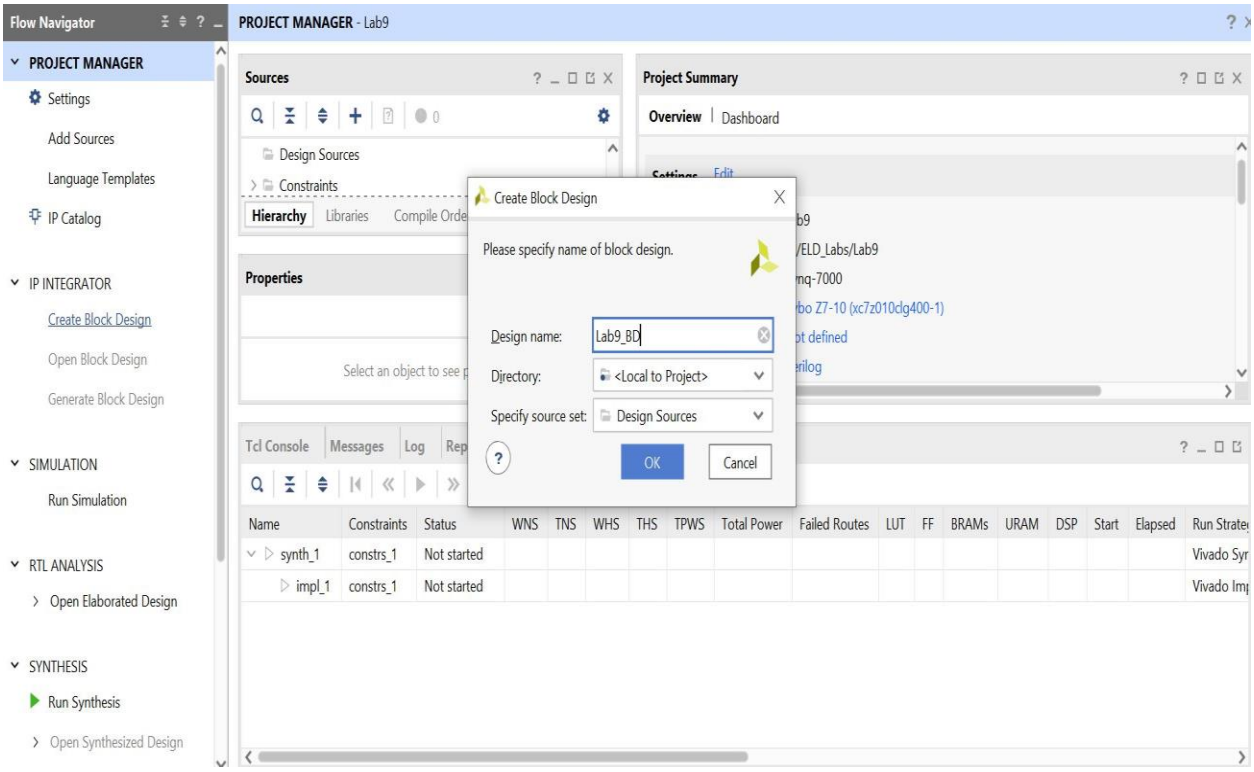
Please follow the following steps to create the block design:

1. Create a new Vivado Project.
2. At the next window, tick "Do not specify sources at this time".

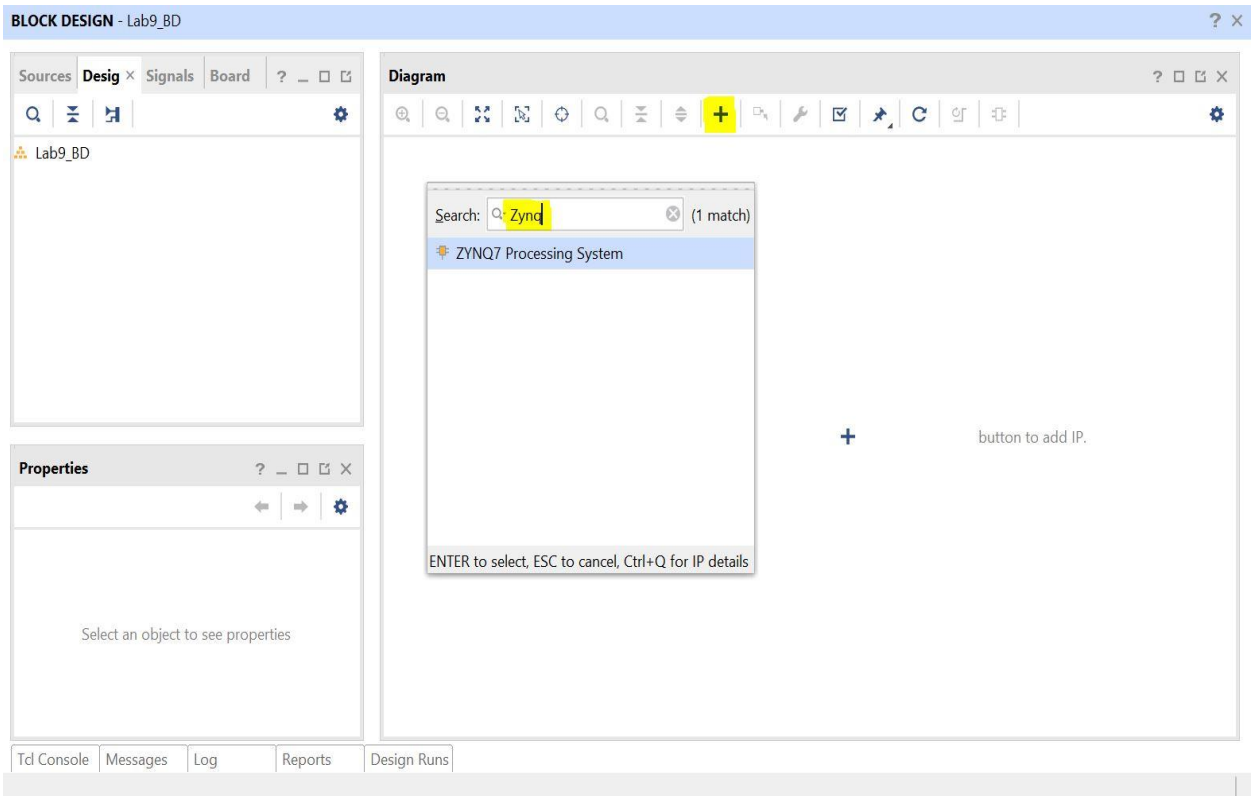



3. In the board selection window, select Zybo Z7-10 and click on Finish in the next window.
4. Under the IP Integrator dropdown, click on Create Block Design.

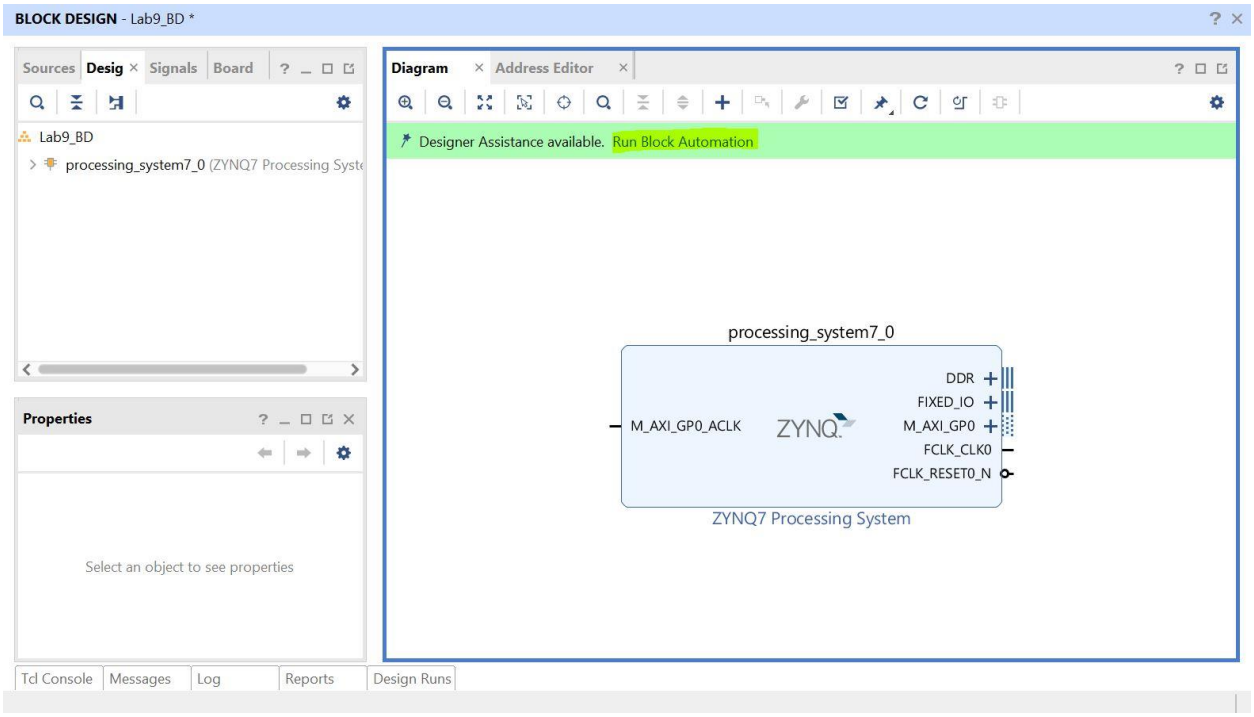




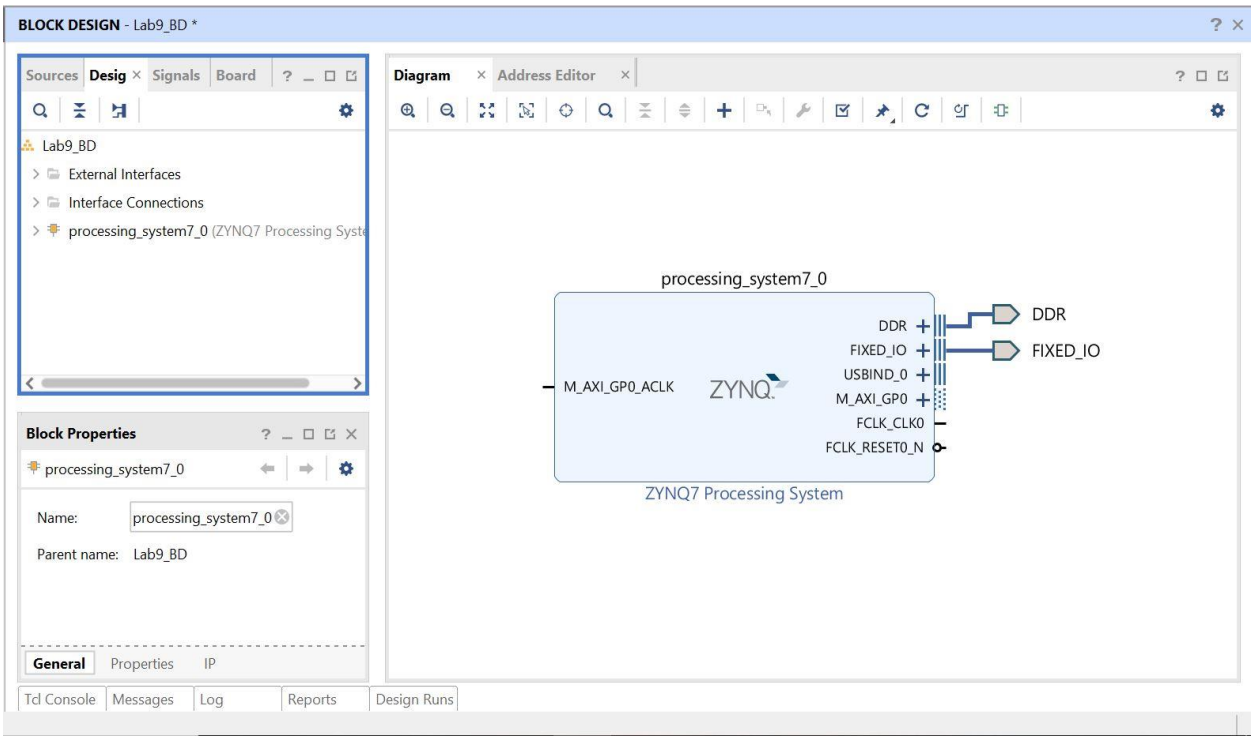

5. In the diagram, window click on +(Add IP sign) and search for Zynq PS, and double click on it.



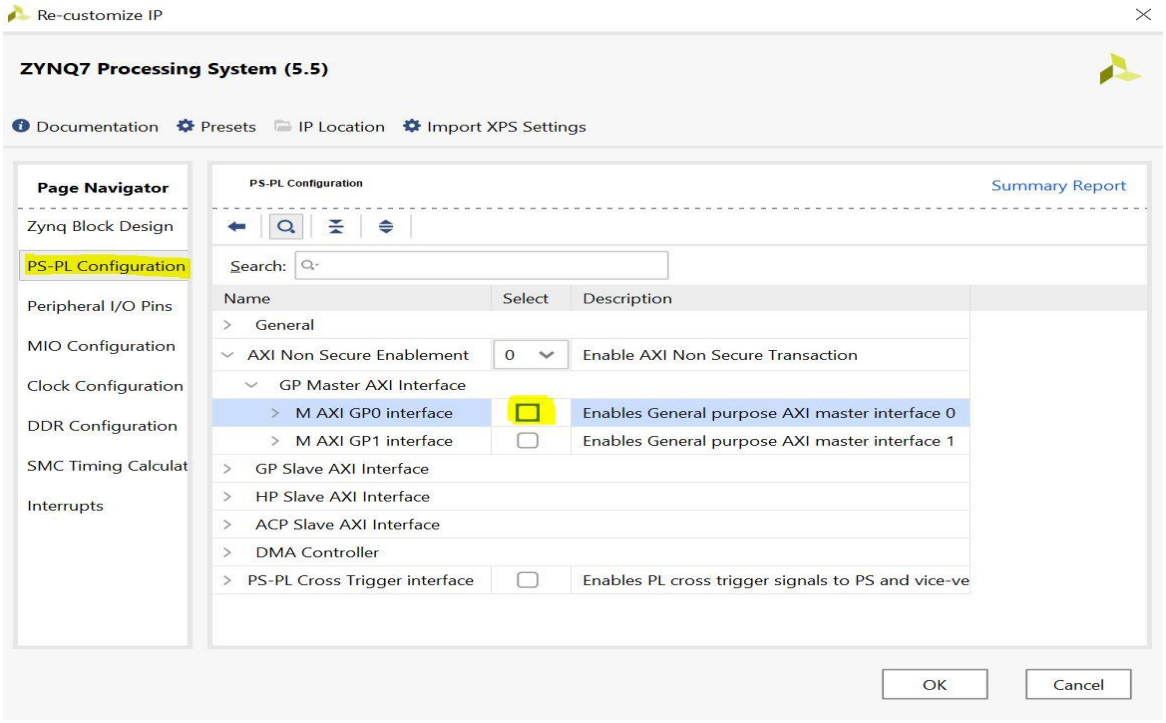
6. Once the IP got placed in the diagram window, "Run Block Automation." This step needs to be done every time you add an IP.

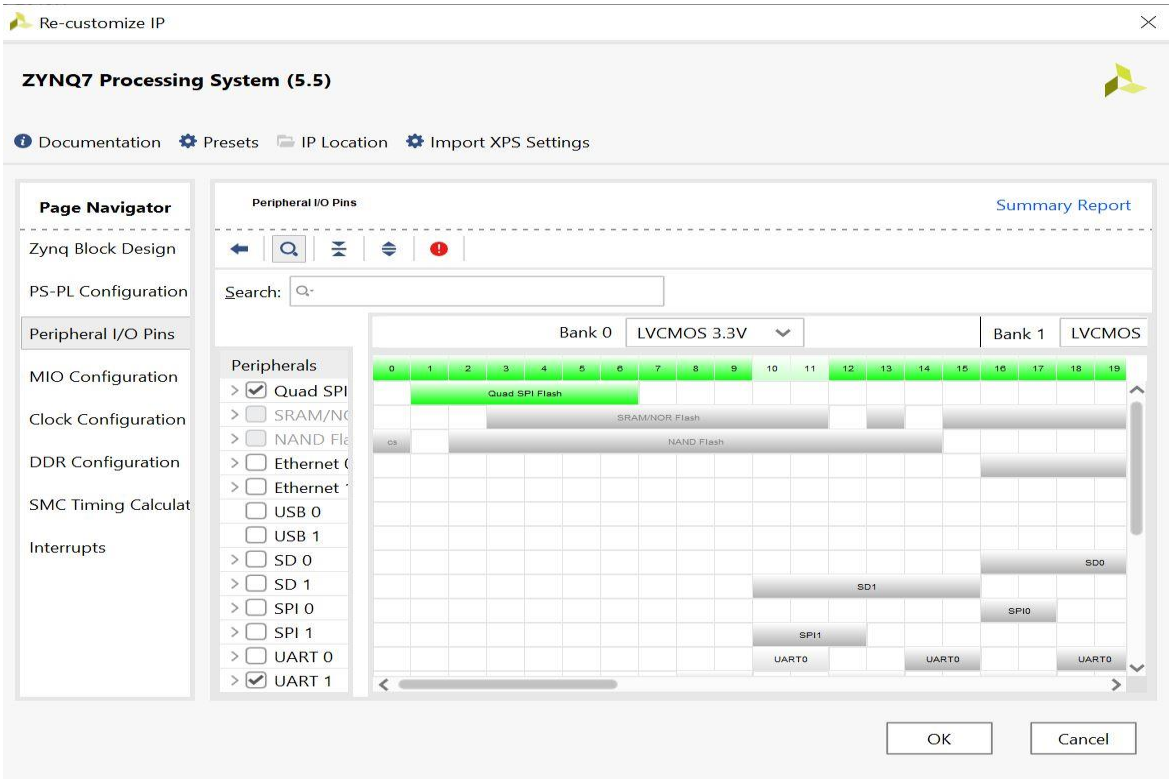


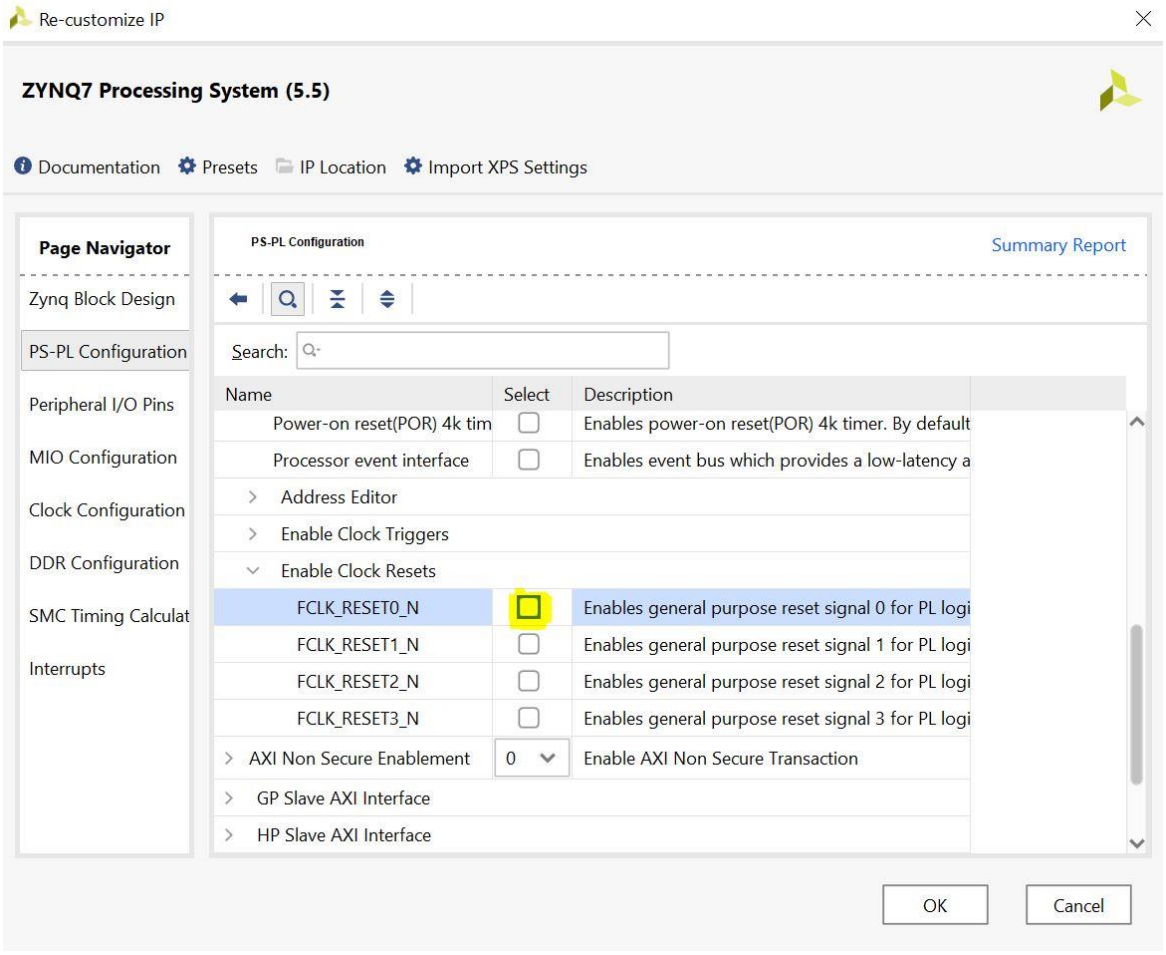

7. After running the block automation, your block diagram should look like the one given below:

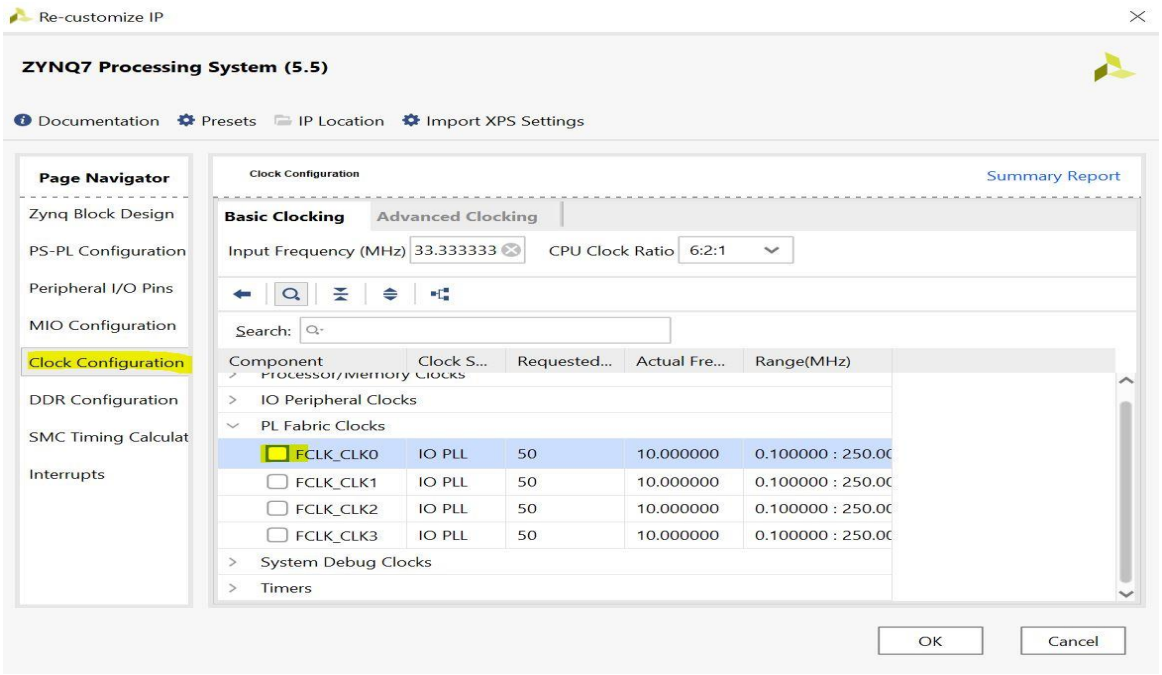


8. Next, we will customize the IP according to our needs. For this Lab, we need only Fixed_IO and DDR memory controller. Double click on the Zynq IP and follow the following steps to customize.

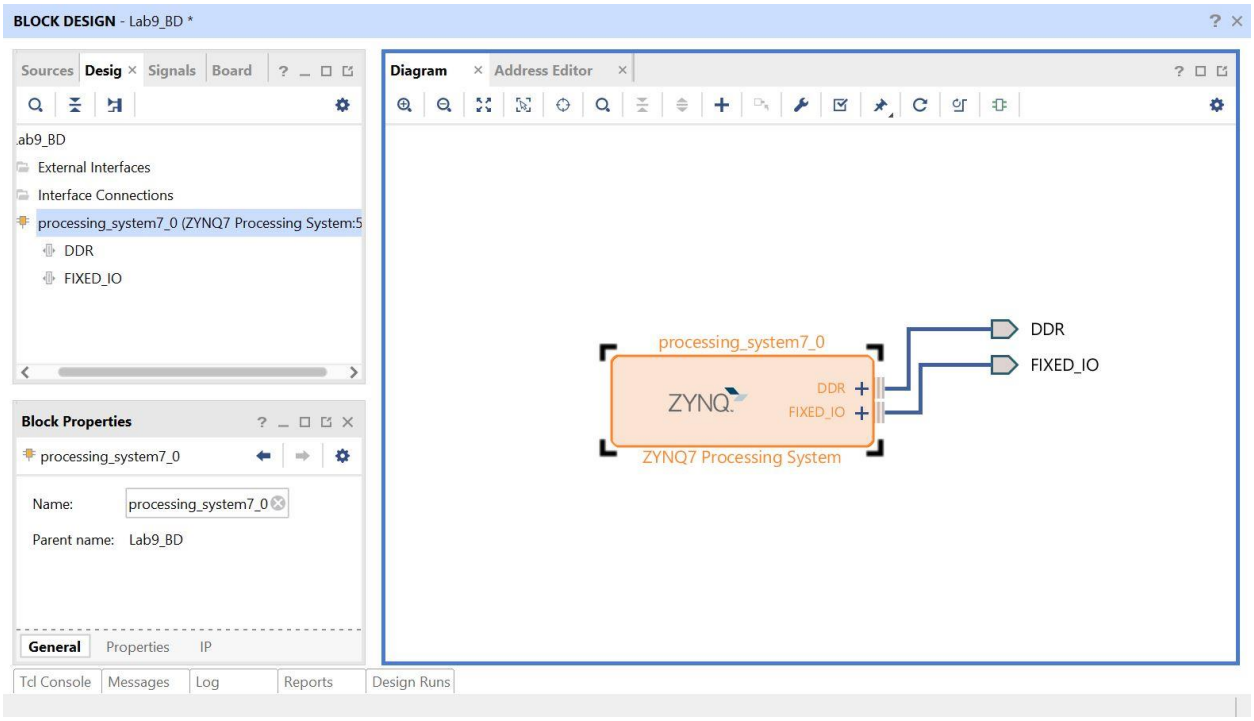


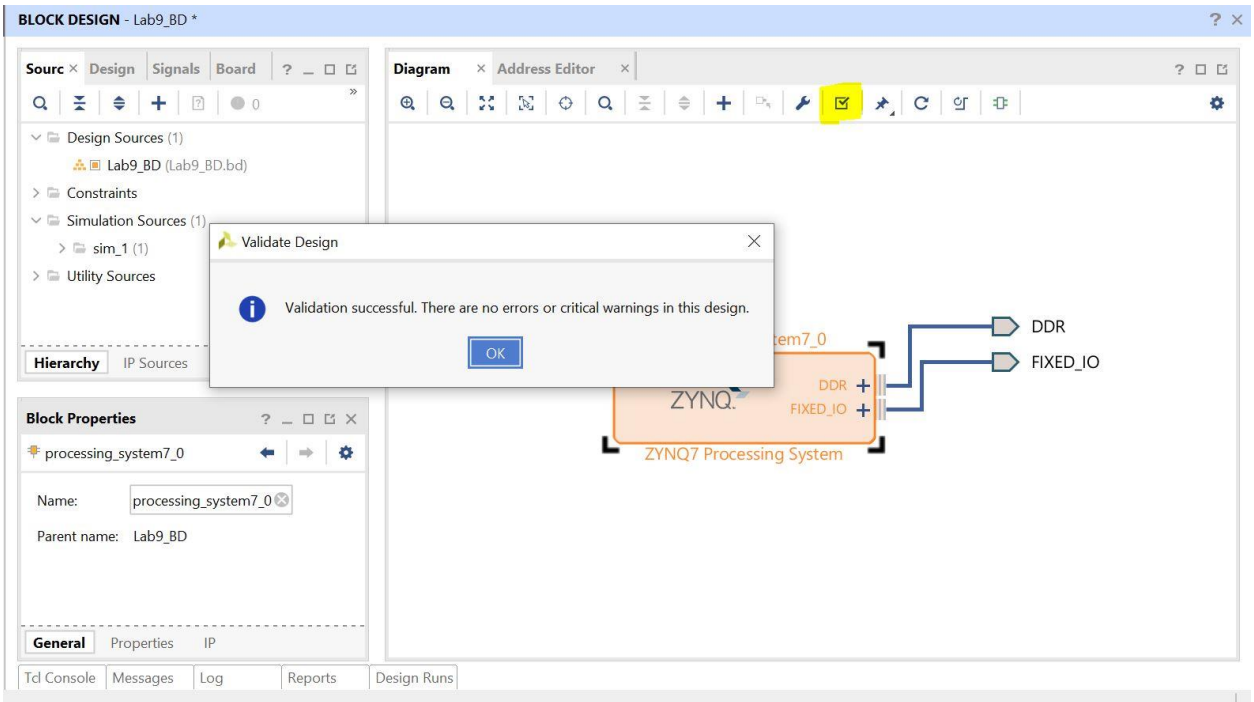

Re-customize IP
ZYNQ7 Processing System (5.5)
Documentation
Presets
IP Location
Import XPS Settings
Page Navigator
Zynq Block Design
PS-PL Configuration
Peripheral I/O Pins
MIO Configuration
Clock Configuration
DDR Configuration
SMC Timing Calculat
Interrupts
PS-PL Configuration
Summary Report
Search:
Name
Select
Description
Power-on reset(POR) 4k tim
Enables power-on reset(POR) 4k timer. By default
Processor event interface
Enables event bus which provides a low-latency a
Address Editor
Enable Clock Triggers
Enable Clock Resets
FCLK_RESET0_N
Enables general purpose reset signal 0 for PL logi
FCLK_RESET1_N
Enables general purpose reset signal 1 for PL logi
FCLK_RESET2_N
Enables general purpose reset signal 2 for PL logi
FCLK_RESET3_N
Enables general purpose reset signal 3 for PL logi
AXI Non Secure Enablement
0
Enable AXI Non Secure Transaction
GP Slave AXI Interface
HP Slave AXI Interface
OK
Cancel

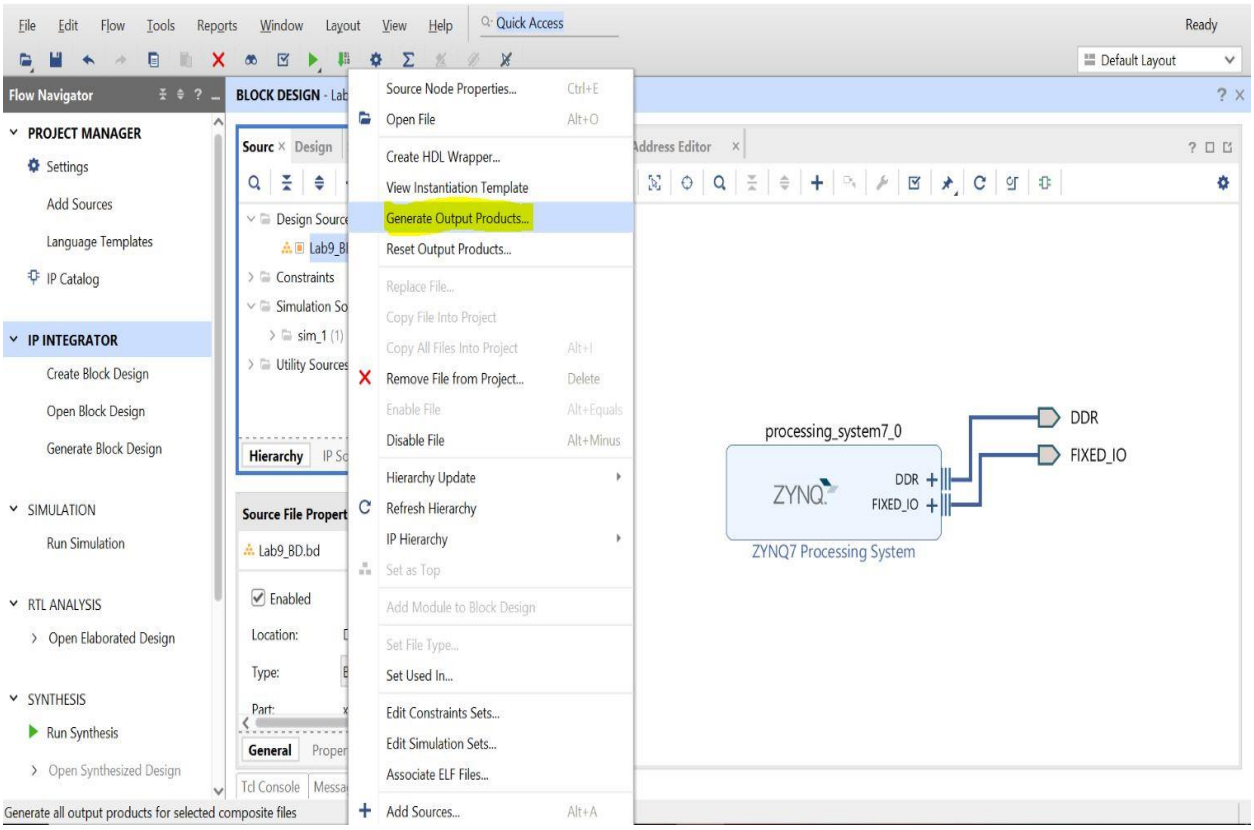

Re-customize IP
ZYNQ7 Processing System (5.5)
Documentation
Presets
IP Location
Import XPS Settings
Page Navigator
Zynq Block Design
PS-PL Configuration
Peripheral I/O Pins
MIO Configuration
Clock Configuration
DDR Configuration
SMC Timing Calculat
Interrupts
Clock Configuration
Summary Report
Basic Clocking
Advanced Clocking
Input Frequency (MHz)
33.333333
CPU Clock Ratio
6:2:1
Search:
Component
Clock S...
Requested...
Actual Fre...
Range(MHz)
IO Peripheral Clocks
PL Fabric Clocks
FCLK_CLK0
IO PLL
50
10.000000
0.100000 : 250.00
FCLK_CLK1
IO PLL
50
10.000000
0.100000 : 250.00
FCLK_CLK2
IO PLL
50
10.000000
0.100000 : 250.00
FCLK_CLK3
IO PLL
50
10.000000
0.100000 : 250.00
System Debug Clocks
Timers
OK
Cancel

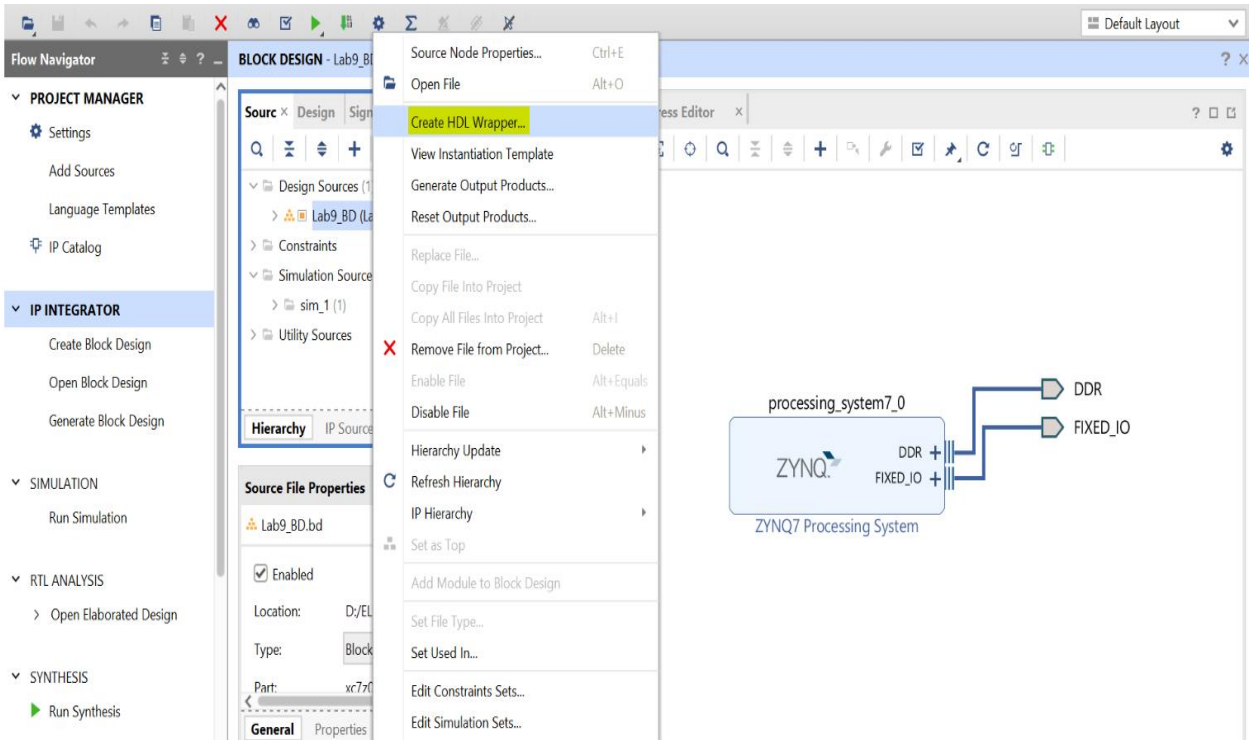
9. After the customization, your block diagram should look like the one shown below.



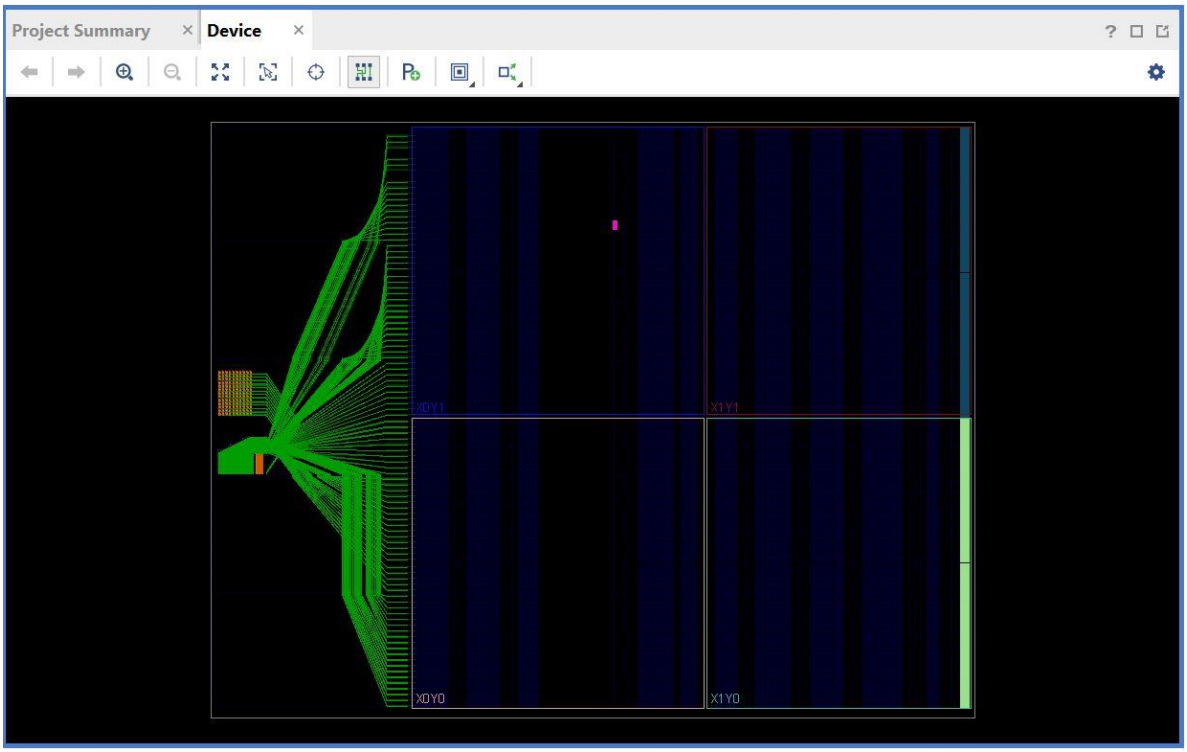

10. The next three steps are important and need to be done in the same order. Validate the design, generate output product, and create HDL wrapper. DO NOT ever miss these steps.
For validation, click on the tick button, as shown below:




File Edit Flow Tools Reports Window Layout View Help
Quick Access
Ready
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
BLOCK DESIGN - Lab
Source Node Properties... Ctrl+E
Open File Alt+O
Create HDL Wrapper...
View Instantiation Template
Generate Output Products...
Reset Output Products...
Replace File...
Copy File Into Project
Copy All Files Into Project Alt+I
Remove File from Project... Delete
Enable File Alt+Equals
Disable File Alt+Minus
Hierarchy Update
Refresh Hierarchy
IP Hierarchy
Set as Top
Add Module to Block Design
Set File Type...
Set Used In...
Edit Constraints Sets...
Edit Simulation Sets...
Associate ELF Files...
Add Sources... Alt+A
Design Source
Constraints
Simulation So
sim_1 (1)
Utility Sources
Hierarchy
Source File Propert
Lab9_BD.bd
Enabled
Location:
Type:
Part:
General
Tcl Console
Generate all output products for selected composite files
Address Editor
processing_system7_0
ZYNQ
DDR
FIXED_IO
ZYNQ7 Processing System

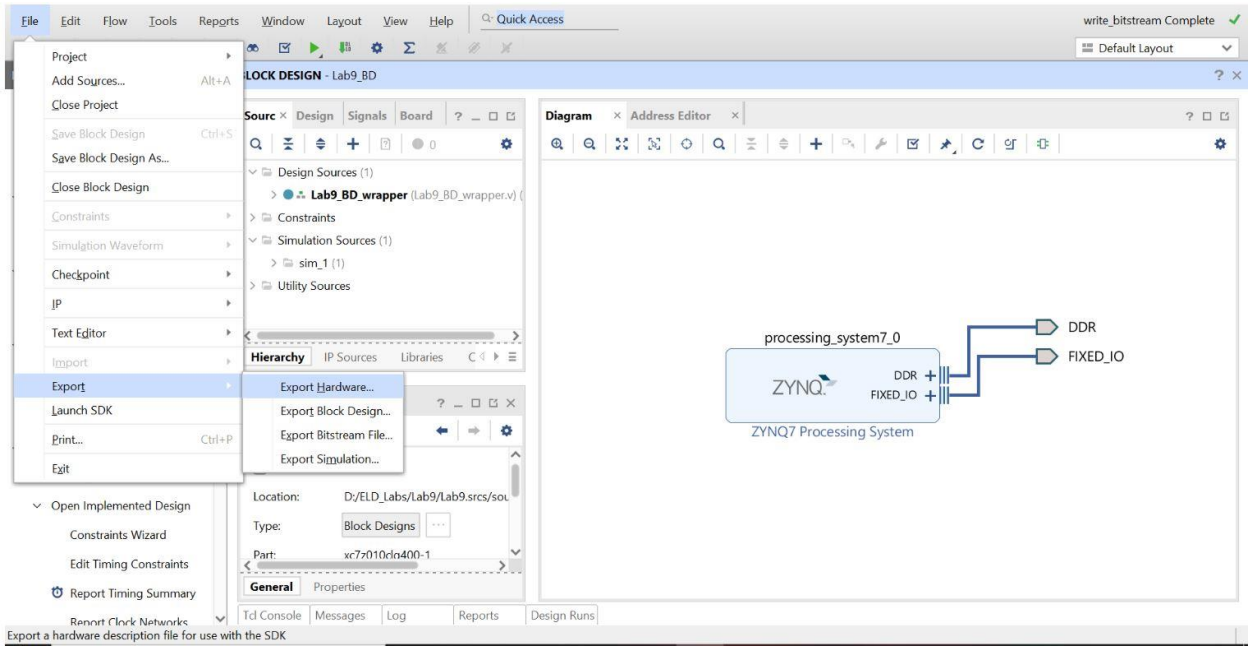

Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Source Node Properties... Ctrl+E
Open File Alt+O
Create HDL Wrapper...
View Instantiation Template
Generate Output Products...
Reset Output Products...
Replace File...
Copy File Into Project
Copy All Files Into Project Alt+I
Remove File from Project... Delete
Enable File Alt+Equals
Disable File Alt+Minus
Hierarchy Update
Refresh Hierarchy
IP Hierarchy
Set as Top
Add Module to Block Design
Set File Type...
Set Used In...
Edit Constraints Sets...
Edit Simulation Sets...
Design Sources (1)
Constraints
Simulation Source
sim_1 (1)
Utility Sources
Hierarchy
Source File Properties
Lab9_BD.bd
Enabled
Location:
Type:
Part:
General
Properties
processing_system7_0
ZYNQ
DDR
FIXED_IO
ZYNQ7 Processing System

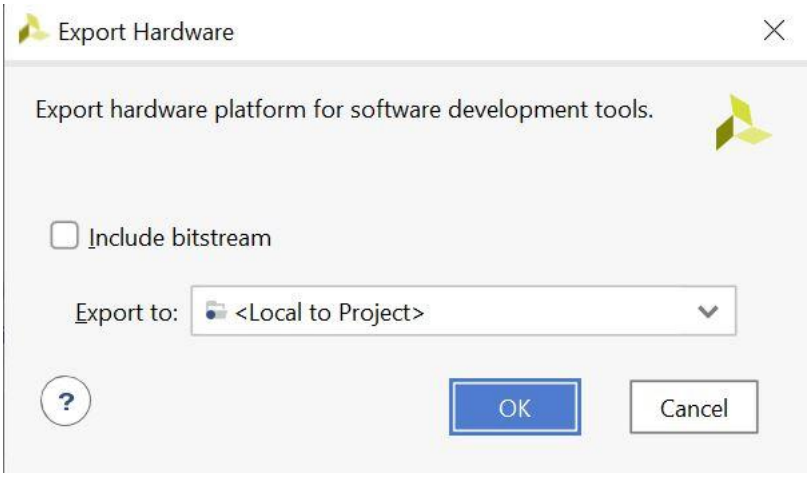
11. Next, we will generate the bitstream and look at the synthesized design.



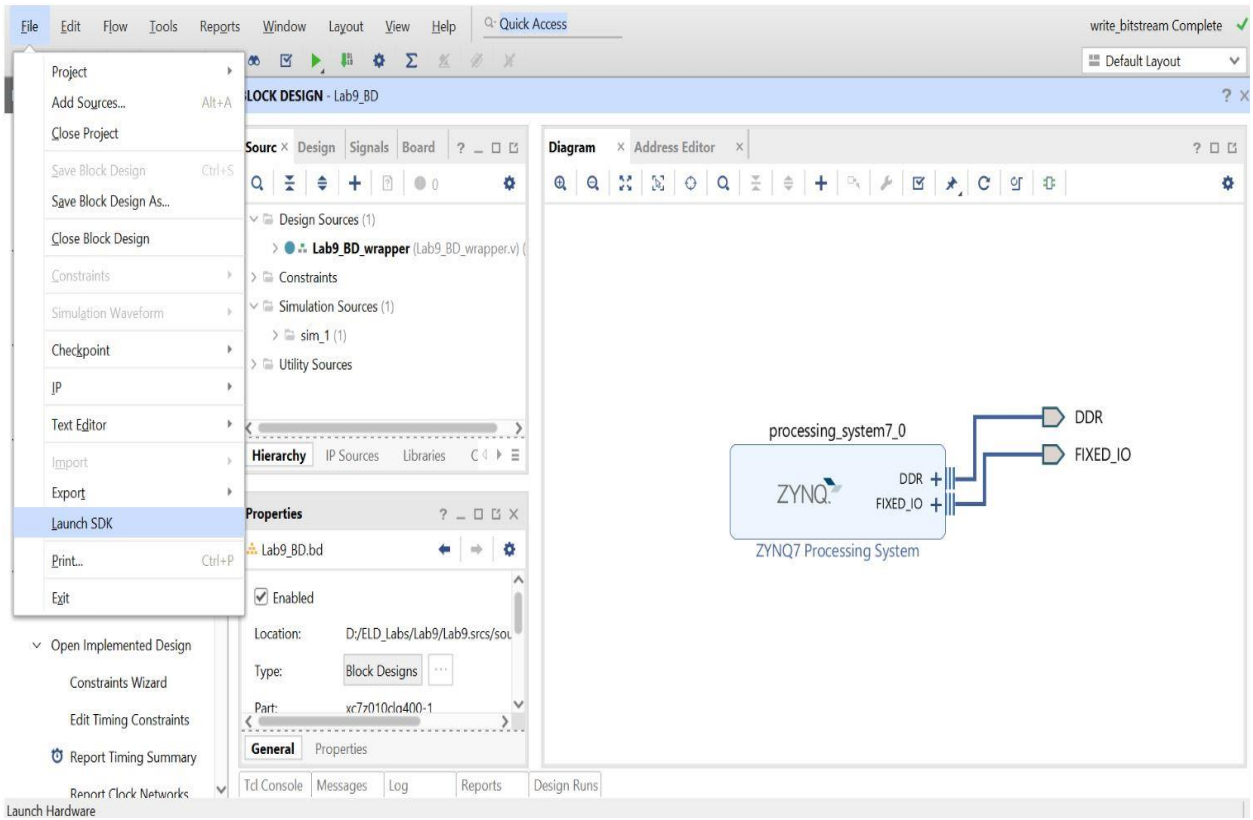

As you can see that there is routing in the PL part(shown on the right) as we have not used the FPGA in this Lab.

12. The final step is to export hardware and Launch SDK. Follow the following screenshots to do so.



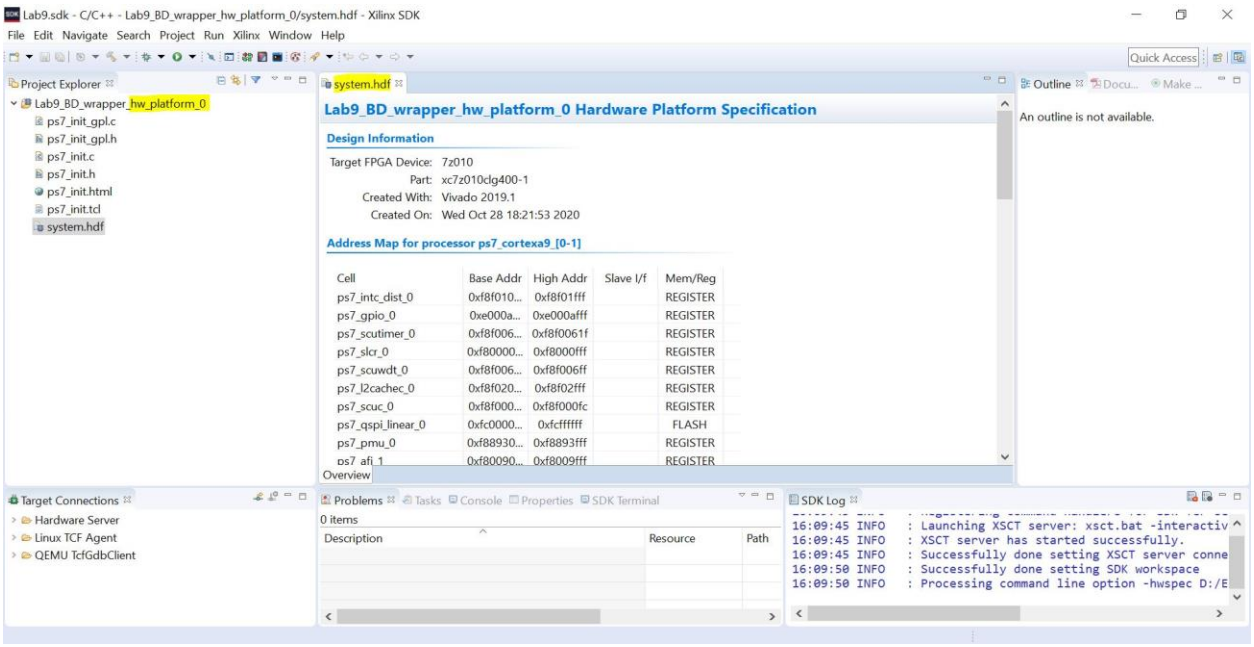



No need to include bitstream this time because we have not used the FPGA.
Finally, Launch SDK.

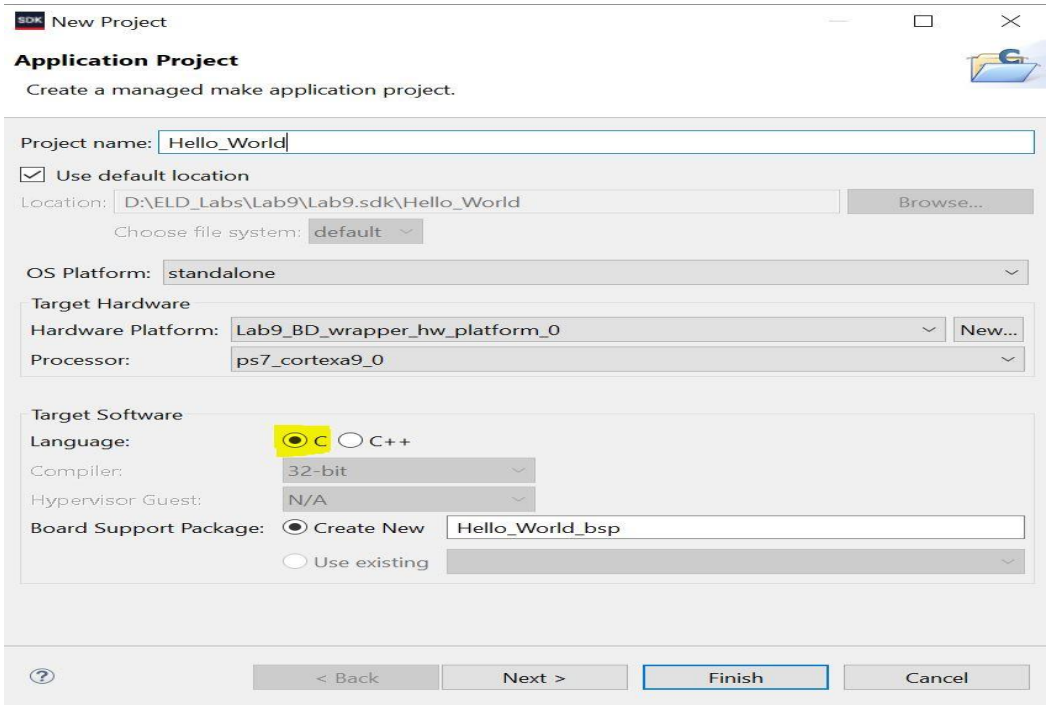


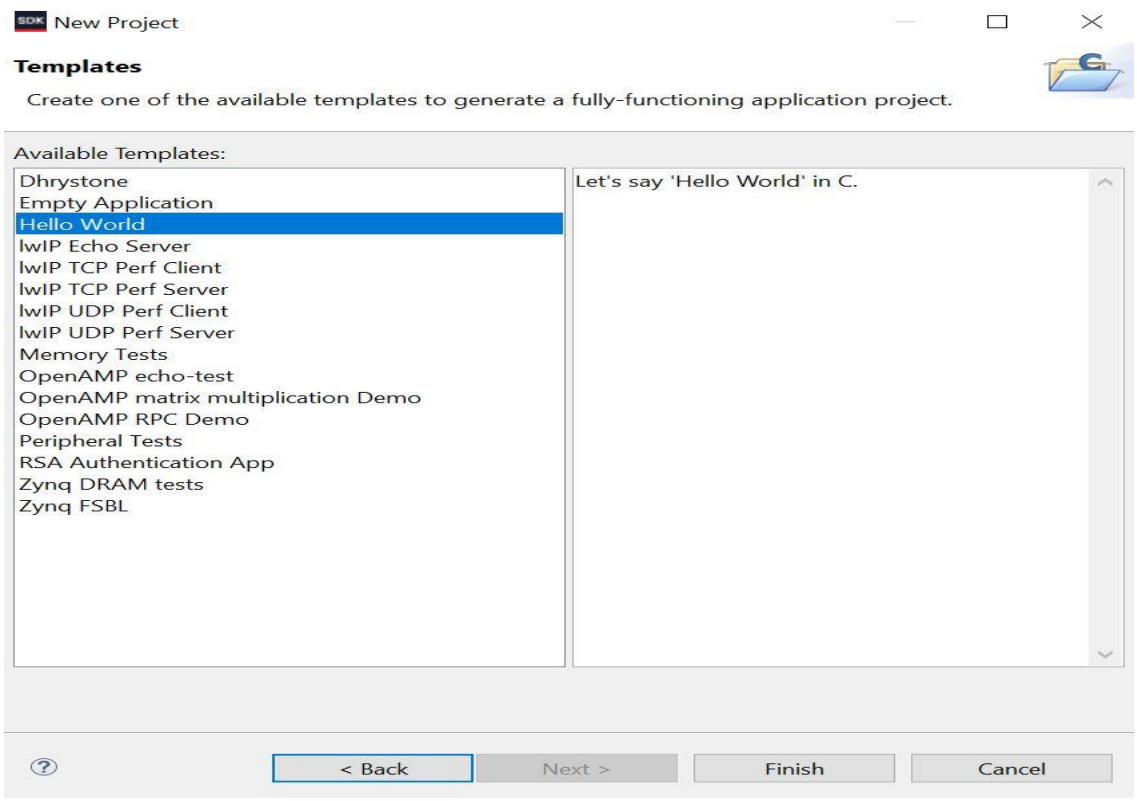
## Part-2

1. Once the SDK is launched, you can see the hardware platform under the Project Explorer Tab, and by default, system.hdf will be open as shown below.

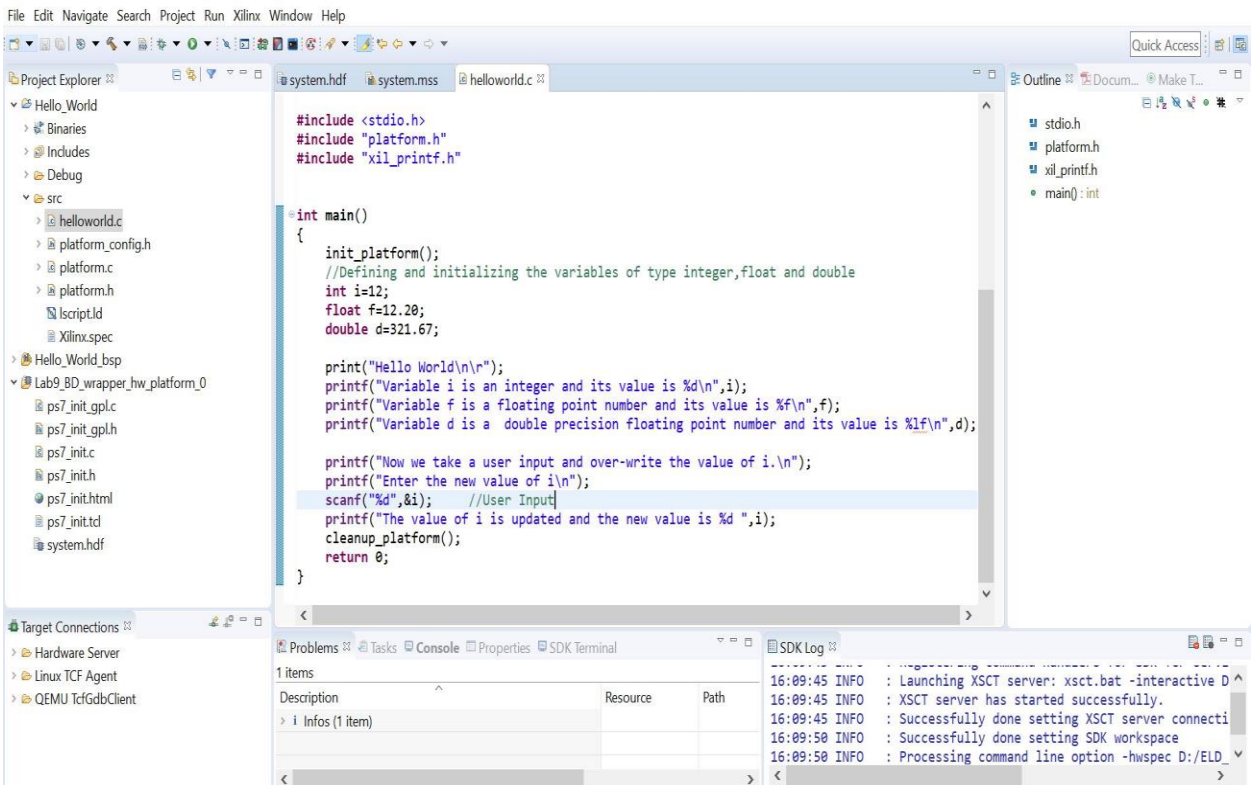



2. Next, we will create an application project. For that, go to File->New->Application Project. Select the project's name, keep the target language as C. Don't click on Finish instead, click on next, and select the Hello World Project.



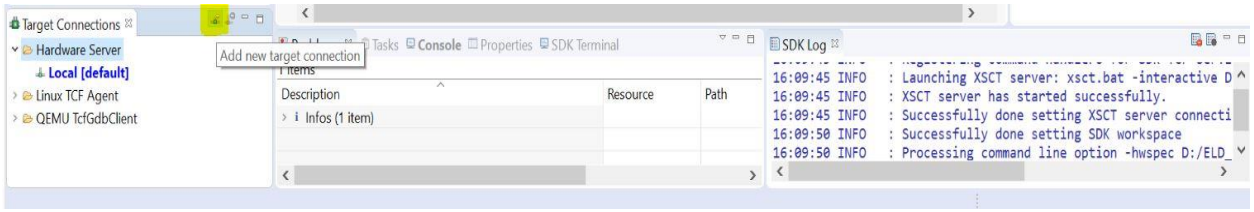


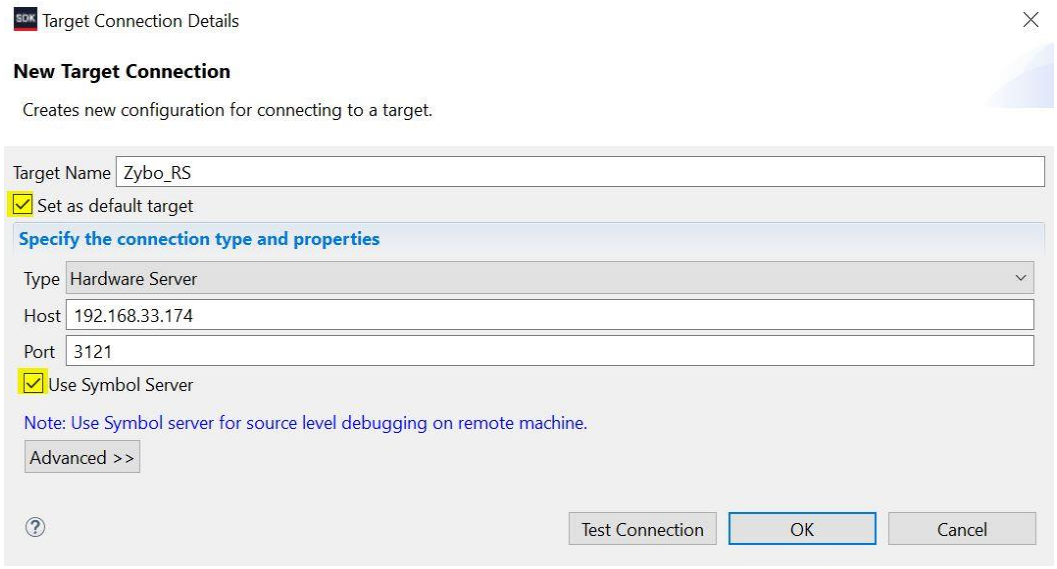

3. Navigate to the source code(.c file) of the application and make the following changes in the default code.



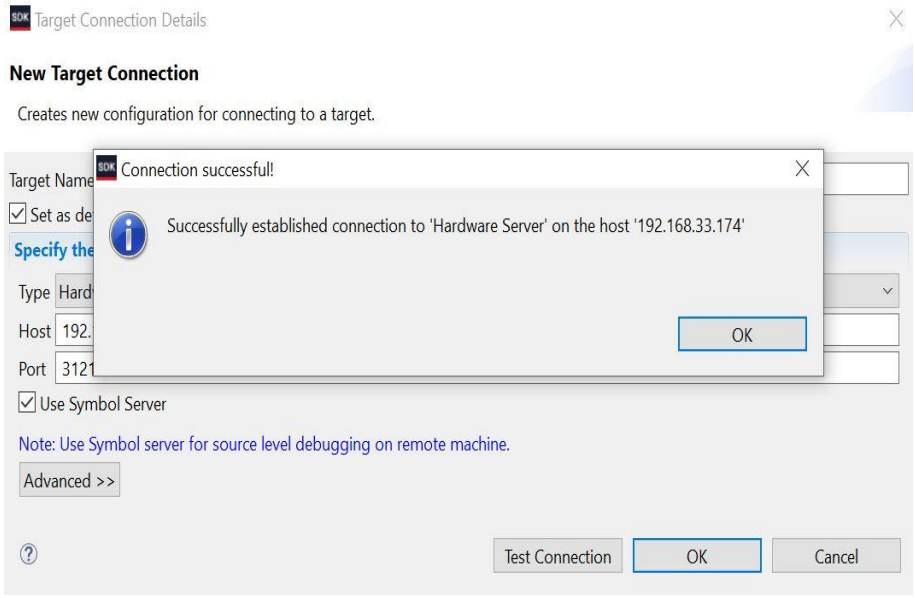
4. Since we are remotely accessing the board, we will use JTAG Terminal for standard Input and Output. To configure the remote access, follow the following steps.

   ➔ In the Target Connection Tab, Add New target Connection

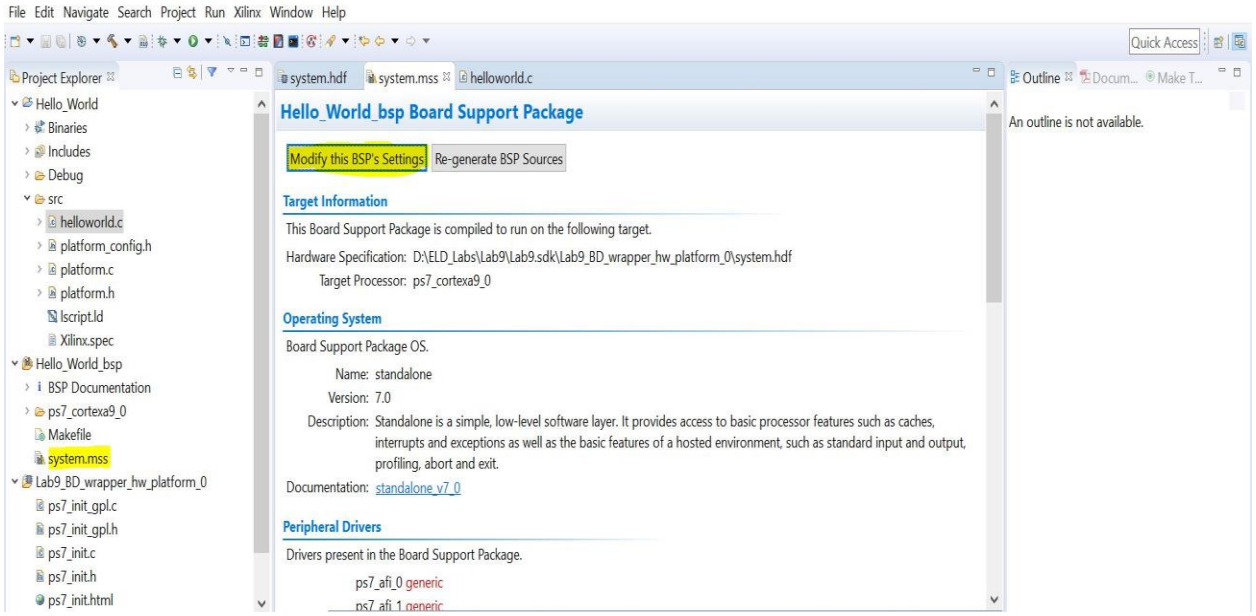


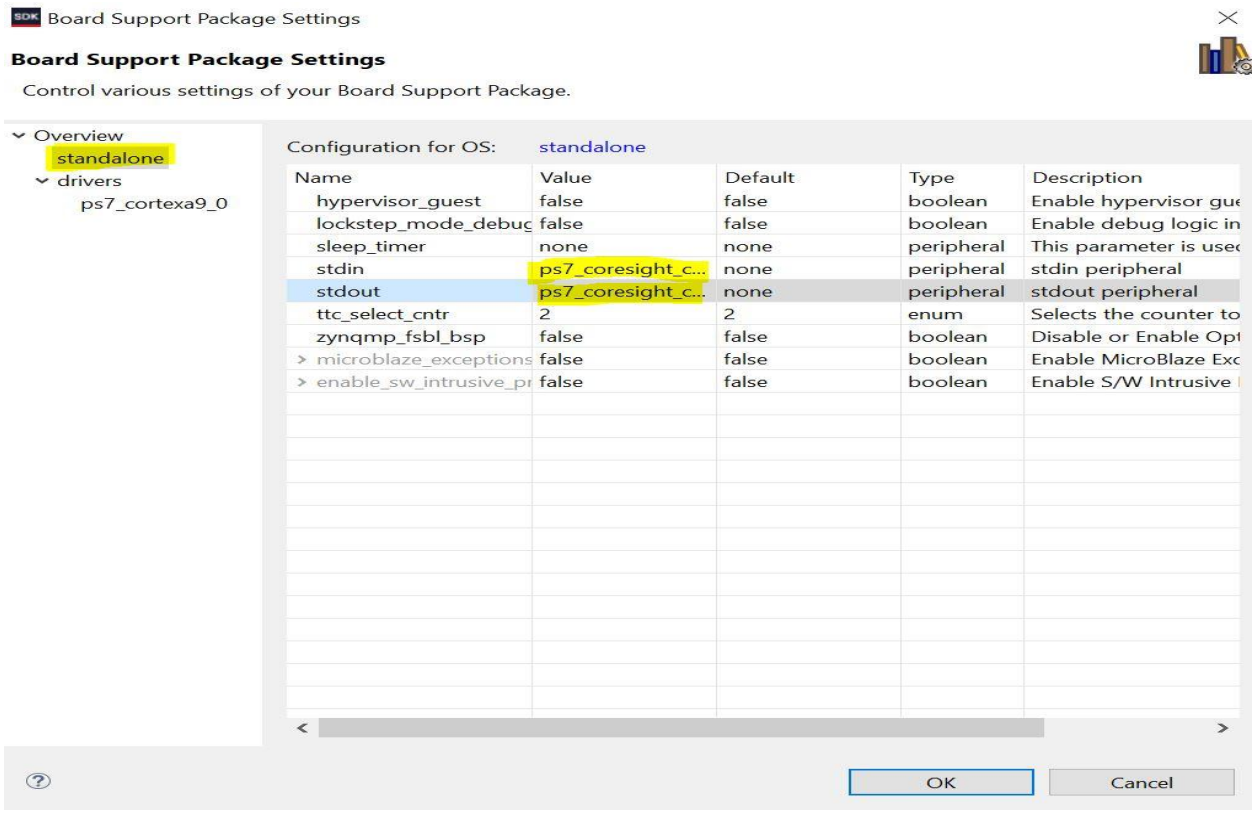




Test the connection to check if the connection is established correctly.

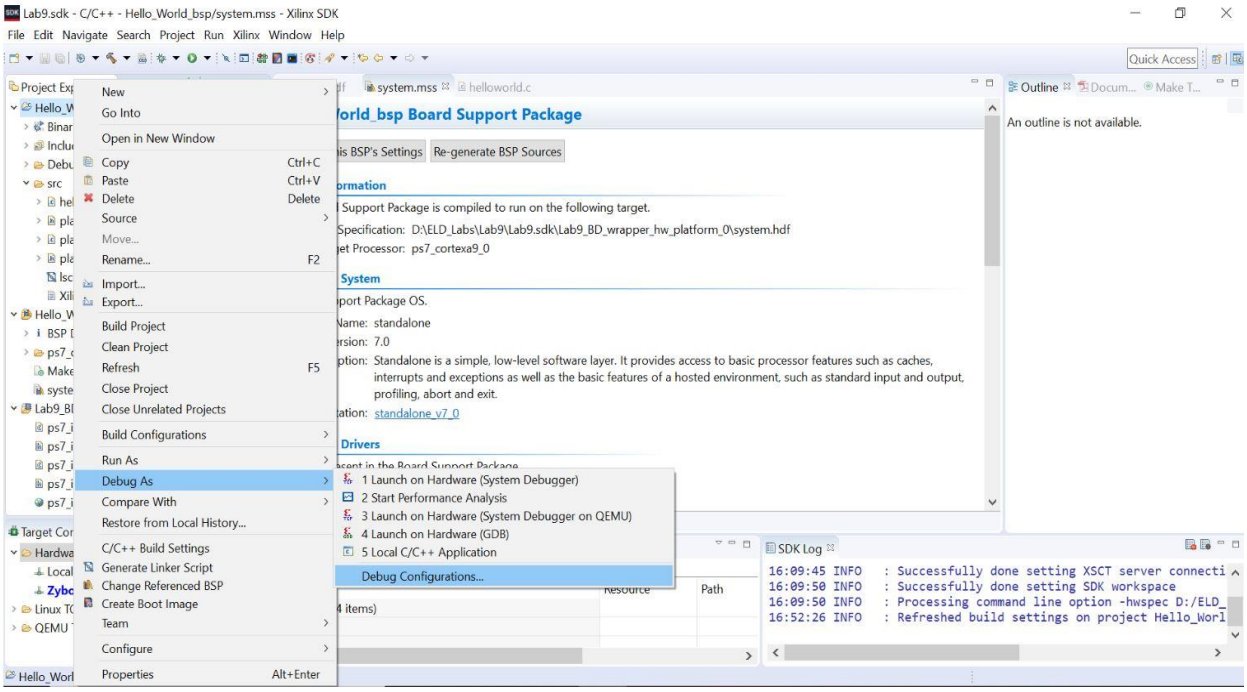


➔ Next, we have to make changes in the Board Support Package and select Coresight for stdin and stdout instead of UART. For that open the system.mss file and click on Modify the BSP's Settings.

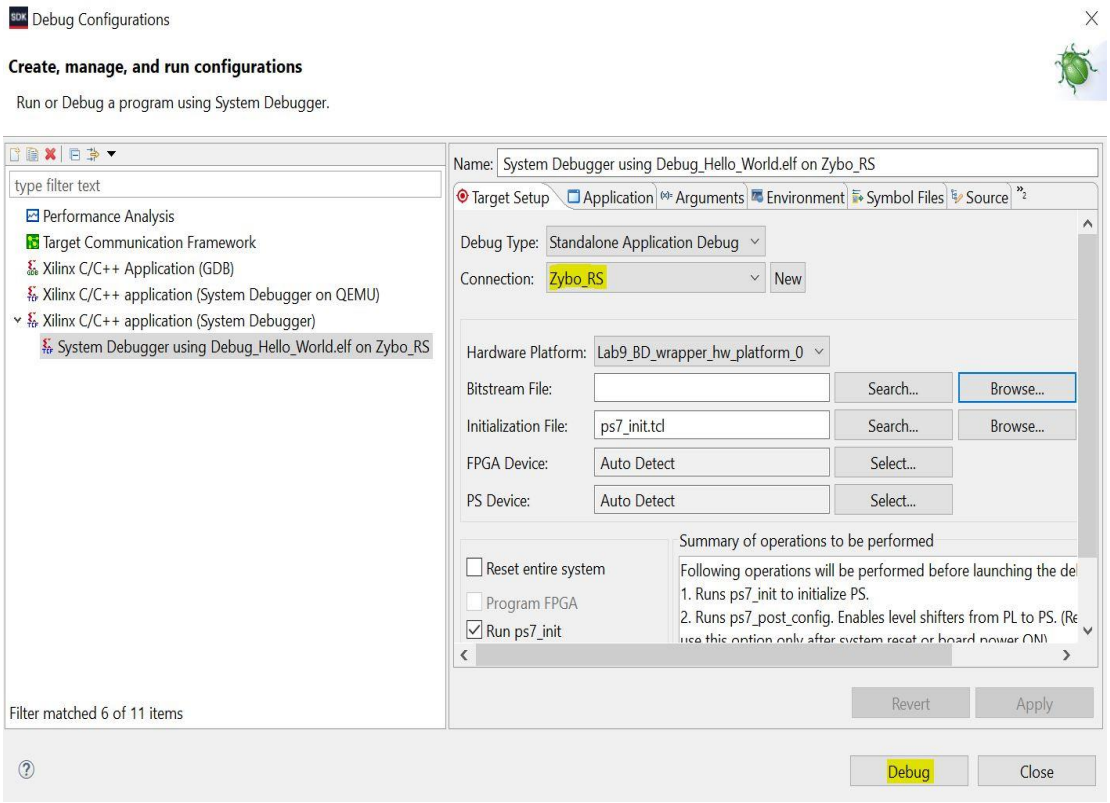


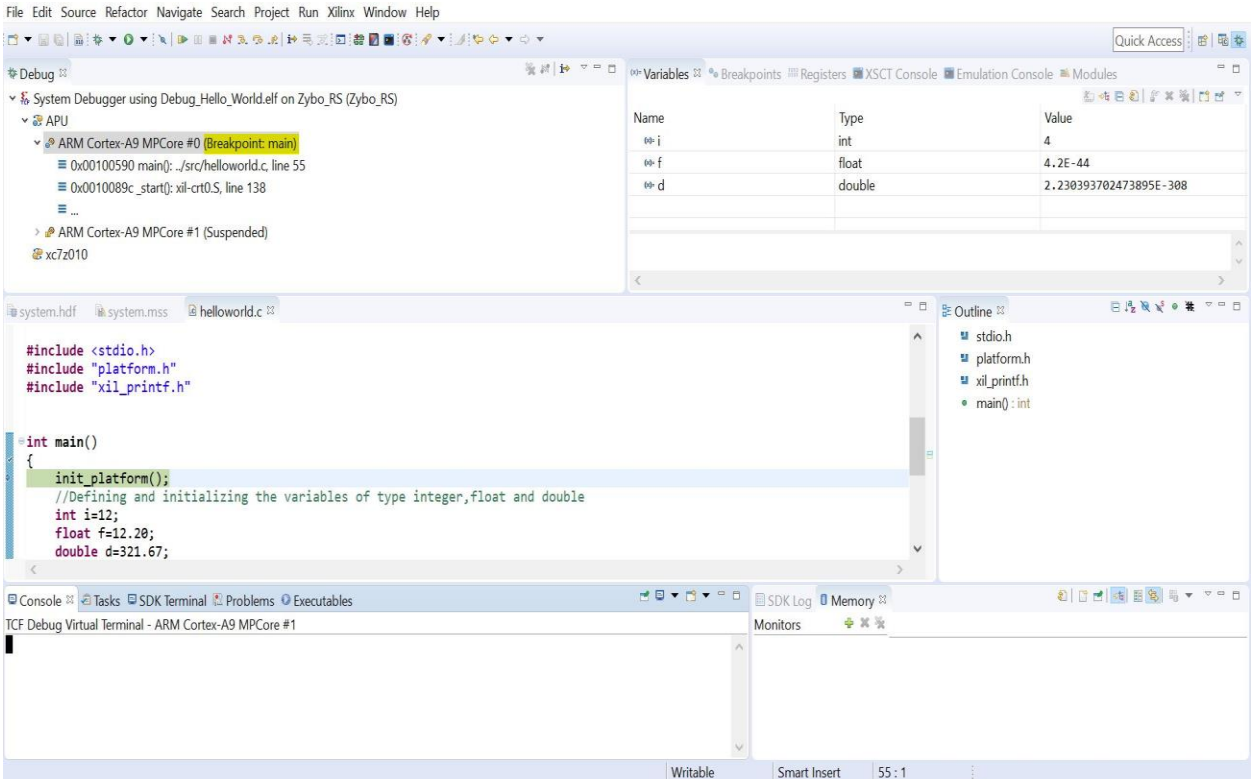
5. Now we will run our application from debug perspective. For that, follow the following steps.



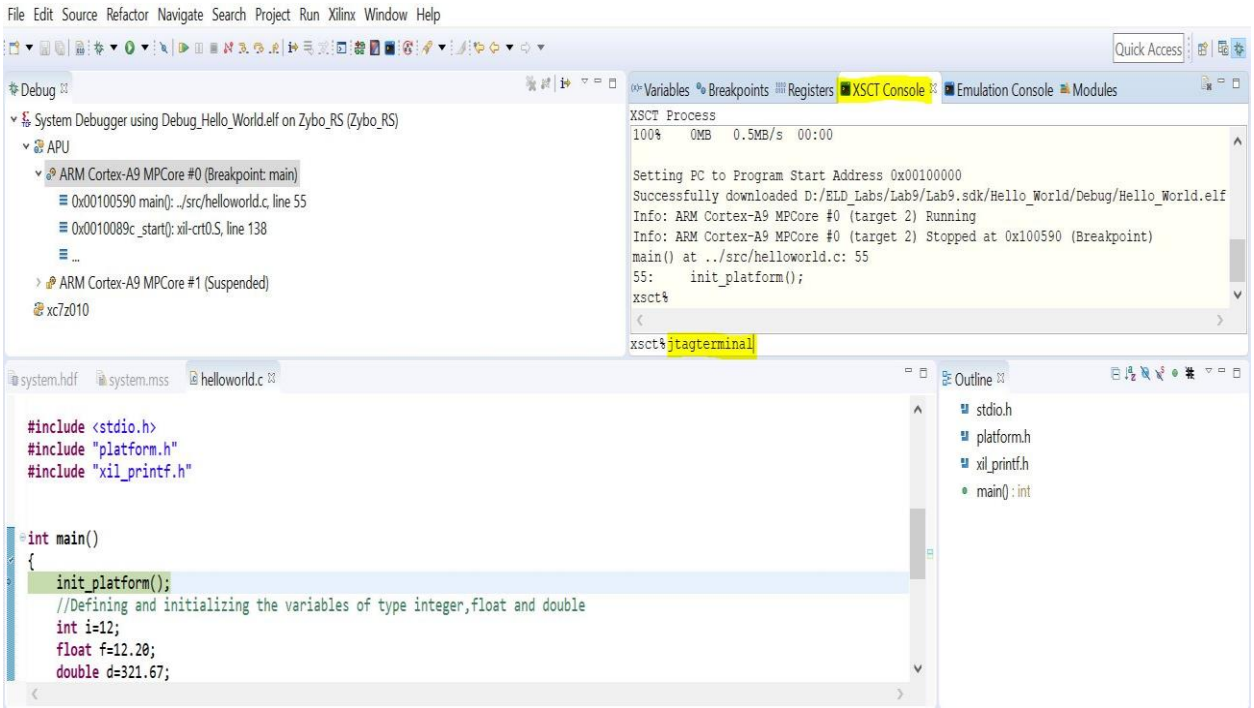

Double Click on the System Debugger, and in the next window, click on Debug.

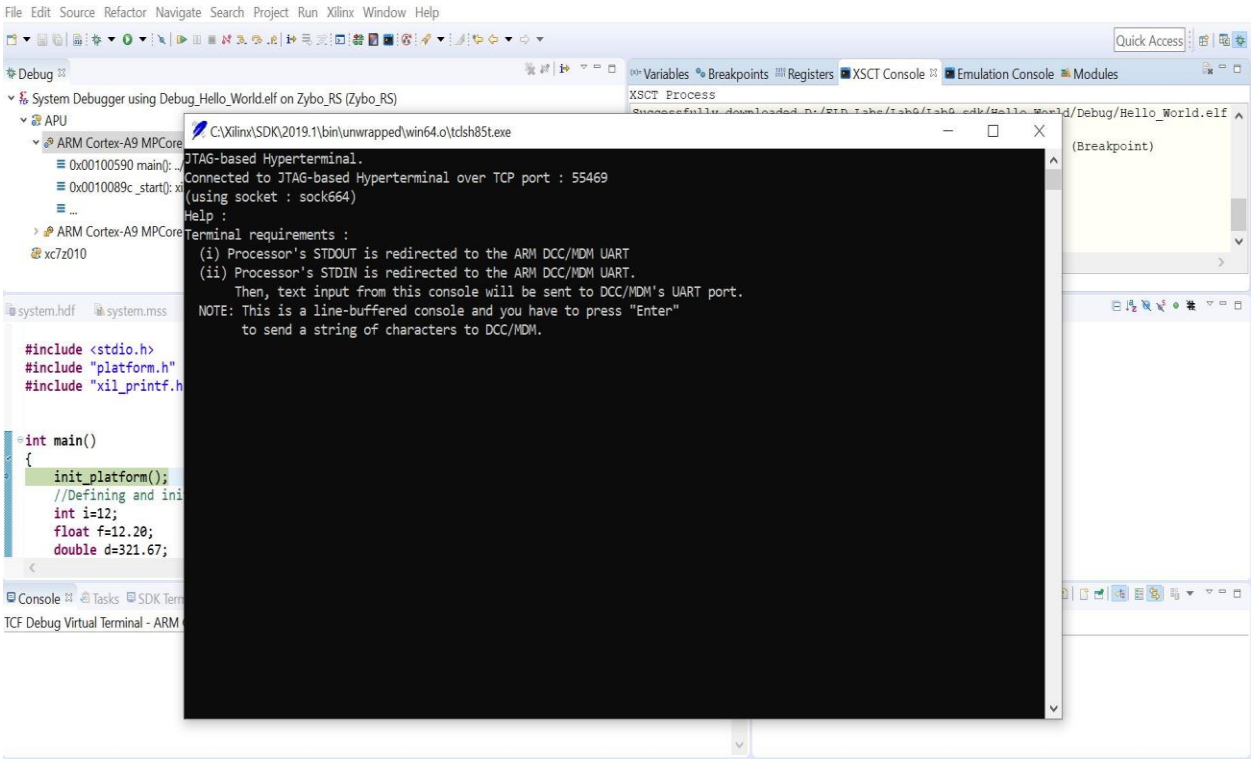


6. Once the debug perspective is launched, the program will stop at the breakpoint main(), as shown below.

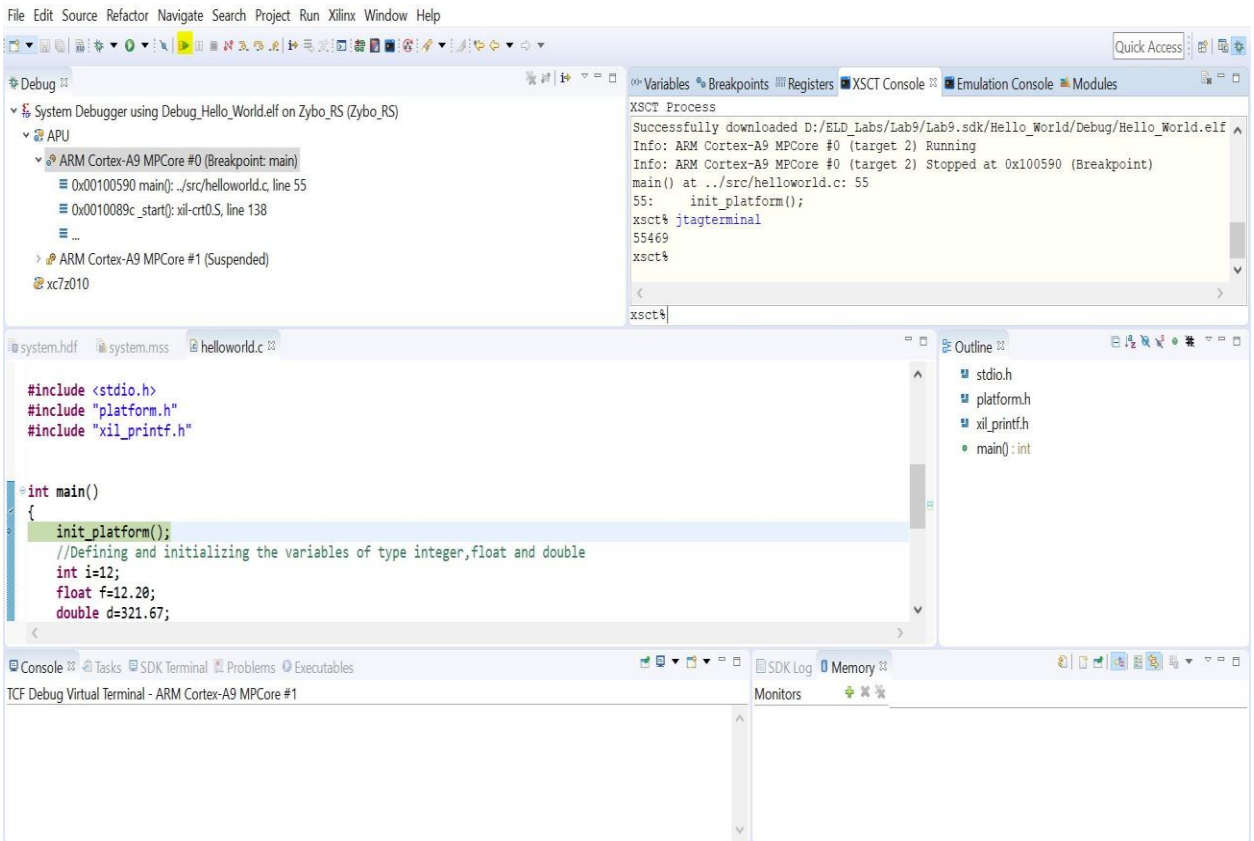


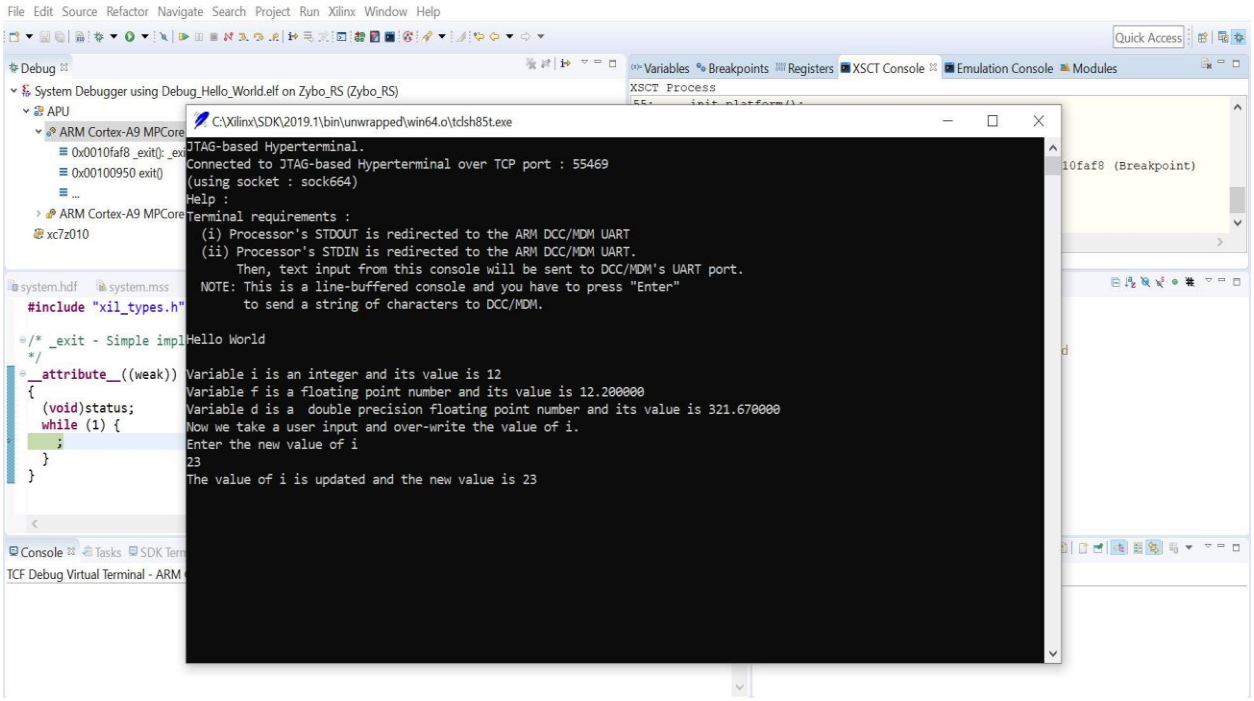

7. Open the JTAG Terminal using the XSCT Console. Type jtagterminal and press enter.





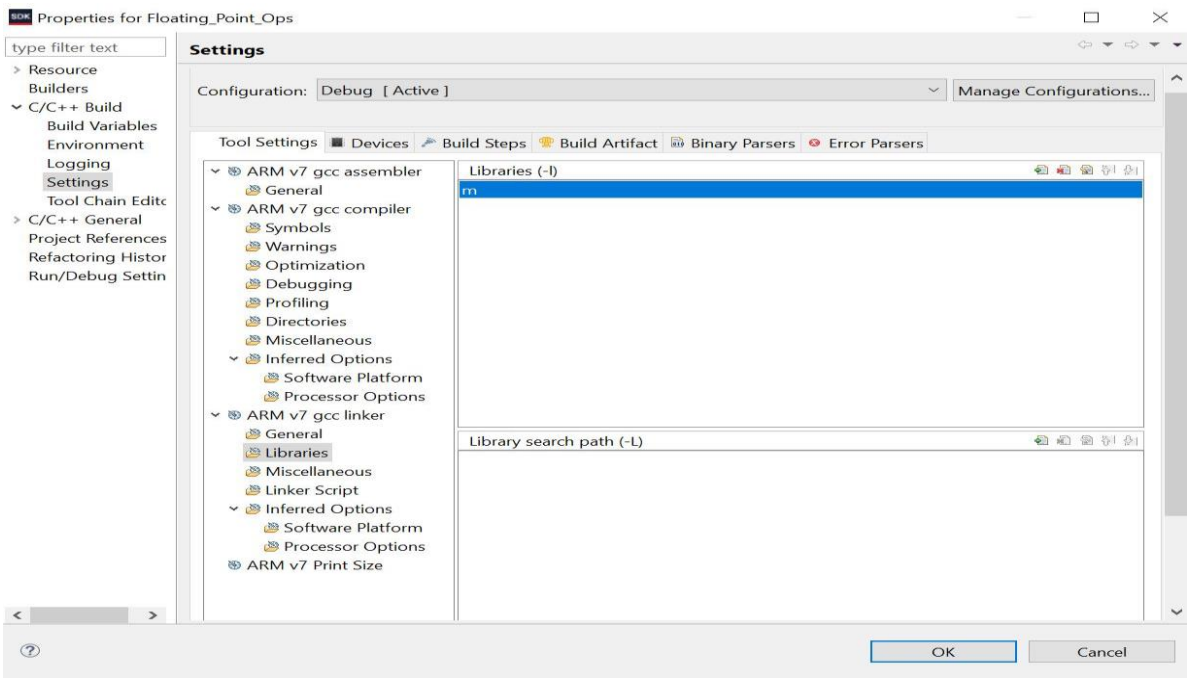

8. Resume the application and look for output statements and user input in the jtagterminal.






## Part-3

1. Create a new application project and modify the BSP settings as we did in Part-2.
2. We will be using math.h header file for mathematical functions. Add this to your source code. At first, the compiler won't recognize it. For that, we need to make changes in C/C++ build settings. Right-click on the project folder and go to C/C++ Build settings and add "m" in the libraries, as shown below.



3. Write the following code and run the application in the debug perspective as we did in Part-2 and verify the result.

system.hdf | system.mss | helloworld.c | system.mss | FPOps.c

```c
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include "math.h"


int main()
{
    init_platform();
    int x=4;
    int n=200;
    int t=2;
    float res1=x/t;
    float res2=(2*log(n))/t;  //log() function calculates the natural logarithm
    float res3=sqrt(res2);
    float result=res1+res3;

    printf("The expression x/t + sqrt(2*ln(n)/t) is being calculated\n");
    printf("x/t resulted in %f\n",res1);
    printf("2*ln(n)/t resulted in %f\n",res2);
    printf("sqrt(2*ln(n)/t) resulted in %f\n",res3);
    printf("The final result of x/t + sqrt(2*ln(n)/t) is %f ",result);

    cleanup_platform();
    return 0;
}
```

C:\Xilinx\SDK\2019.1\bin\unwrapped\win64.o\tclsh85t.exe

```
JTAG-based Hyperterminal.
Connected to JTAG-based Hyperterminal over TCP port : 62579
(using socket : sock660)
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.

The expression x/t + sqrt(2*ln(n)/t) is being calculated
x/t resulted in 2.000000
2*ln(n)/t resulted in 5.298317
sqrt(2*ln(n)/t) resulted in 2.301807
The final result of x/t + sqrt(2*ln(n)/t) is 4.301807
```

**Homework(Graded):**

1. **Paste the screenshots of Block Diagram, C Codes with comments, and JTAG Terminal Output(for both Part 1 and Part 2) in a PDF and submit it to the classroom.**

**Self-Study Questions (Ungraded):**

1. **Explore sleep() command and empty loops to add delay while displaying messages.**
2. **Make the code in Part-3 more generic by taking the input of x,t, and n from the user and display the results.**
3. **Make a calculator which accepts two numbers and the operation as the input and displays the results after performing the selected operation**