Creating Multi Processor Solutions

Course Workbook

Table of Contents

About this Workbook

The contents of this workbook are created by Adiuvo Engineering & Training, Ltd.

If you have any questions about the contents, or need assistance, please contact Adam Taylor at adam@adiuvoengineering.com.

Pre-Lab

Creating Multi Processor Solutions

Required Hardware

Ultra96V2

JTAG / USB Pod

Ultra96V2 Power Supply

Pre-Lab

Downloads and Installations

Step 1 – Download and install the following at least 1 day prior to the workshop. This may take a significant amount of time and drive space.

Watch the video available here to show how to configure the installation

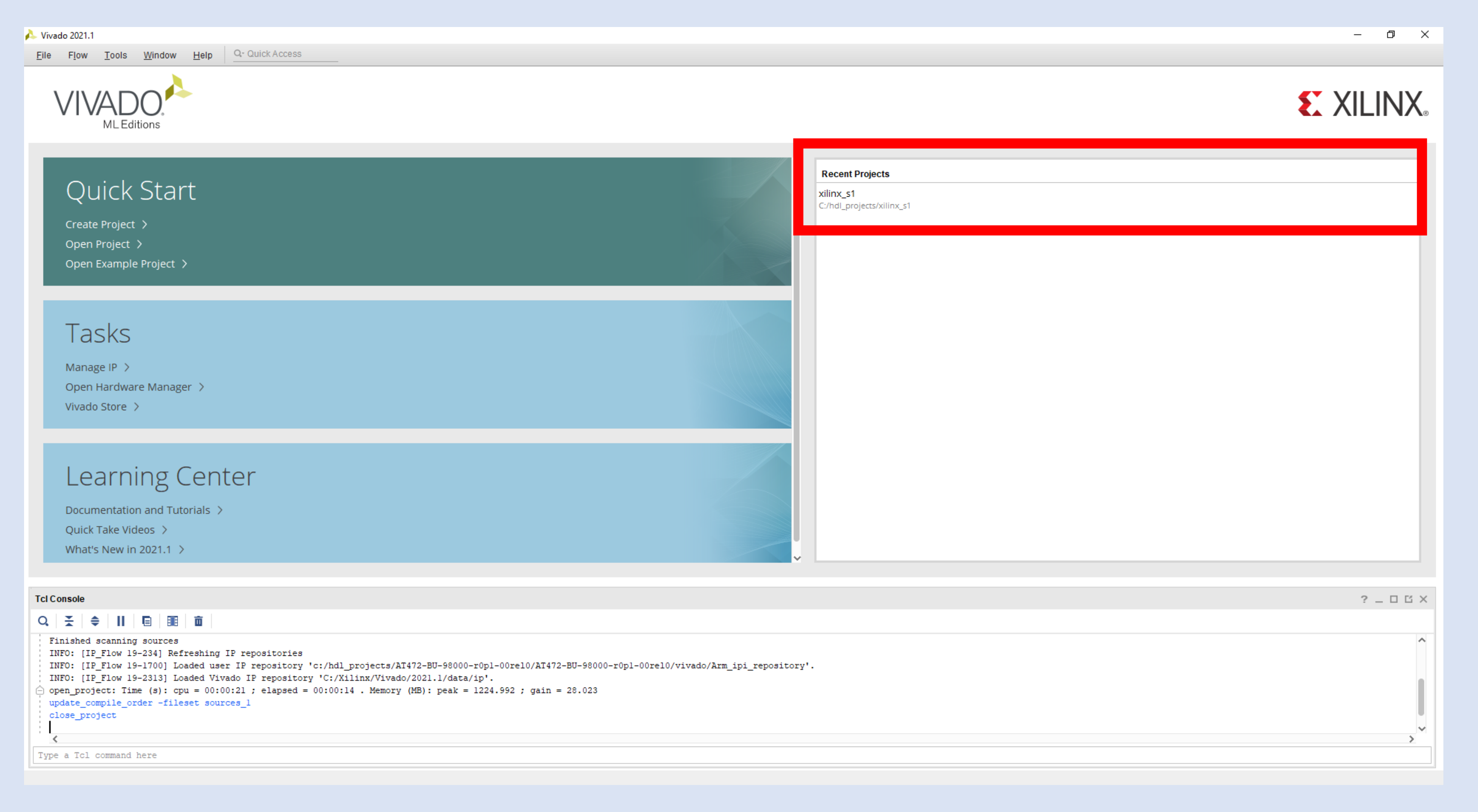
Vitis 2021.1	Download

Lab 2

Project creation & Flow

Lab 2: Understanding Vitis Project creation & Flow

Step 1 – Open Vivado 2021.1 – Open the project from session one



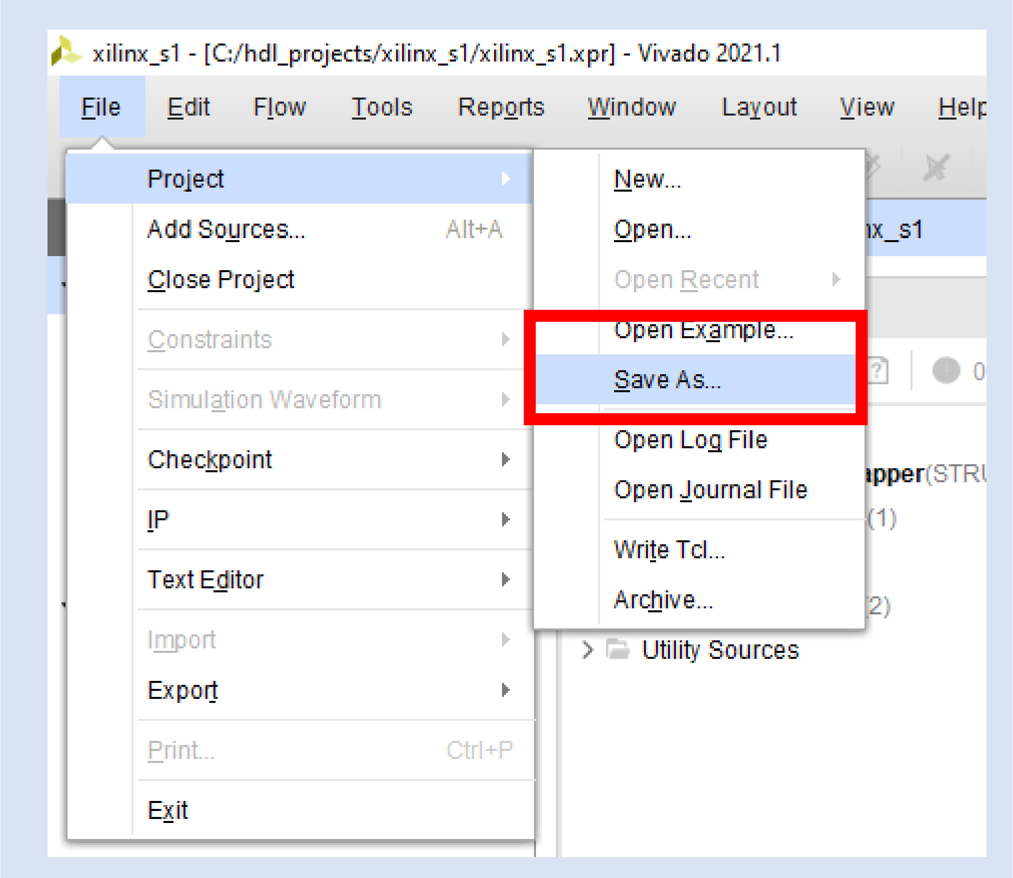
xilinx_s1
C:/hdl_projects/xilinx_s1

```
Finished scanning sources
INFO: [IP_Flow 19-234] Refreshing IP repositories
INFO: [IP_Flow 19-1700] Loaded user IP repository 'c:/hdl_projects/AT472-BU-98000-r0p1-00rel0/AT472-BU-98000-r0p1-00rel0/vivado/Arm_ipi_repository'.
INFO: [IP_Flow 19-2313] Loaded Vivado IP repository 'C:/Xilinx/Vivado/2021.1/data/ip'.
open_project: Time (s): cpu = 00:00:21 ; elapsed = 00:00:14 . Memory (MB): peak = 1224.992 ; gain = 28.023
update_compile_order -fileset sources_1
close_project
```

Type a Tcl command here

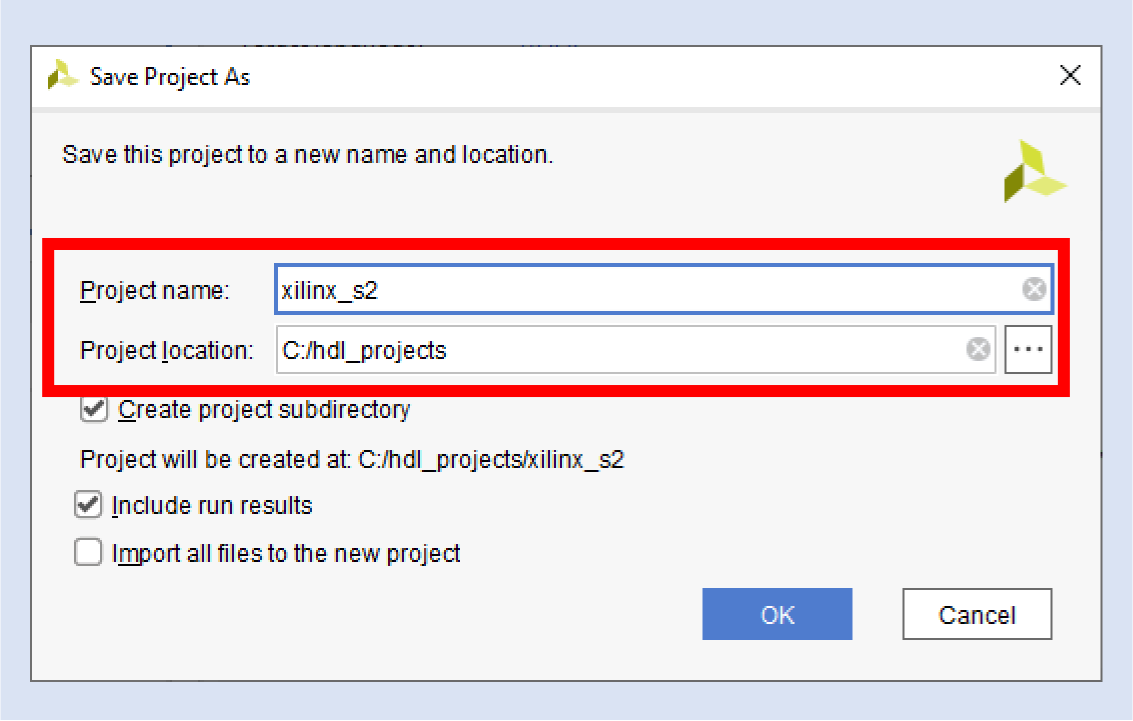
Lab 2: Understanding Vitis Project creation & Flow

Step 2 – From File->Project select save as and name it session two



Lab 2: Understanding Vitis Project creation & Flow

Step 3 – From File->Project select save as and name it session two



Save Project As

Save this project to a new name and location.

Project name: xilinx_s2

Project location: C:/hdl_projects

☑ Create project subdirectory

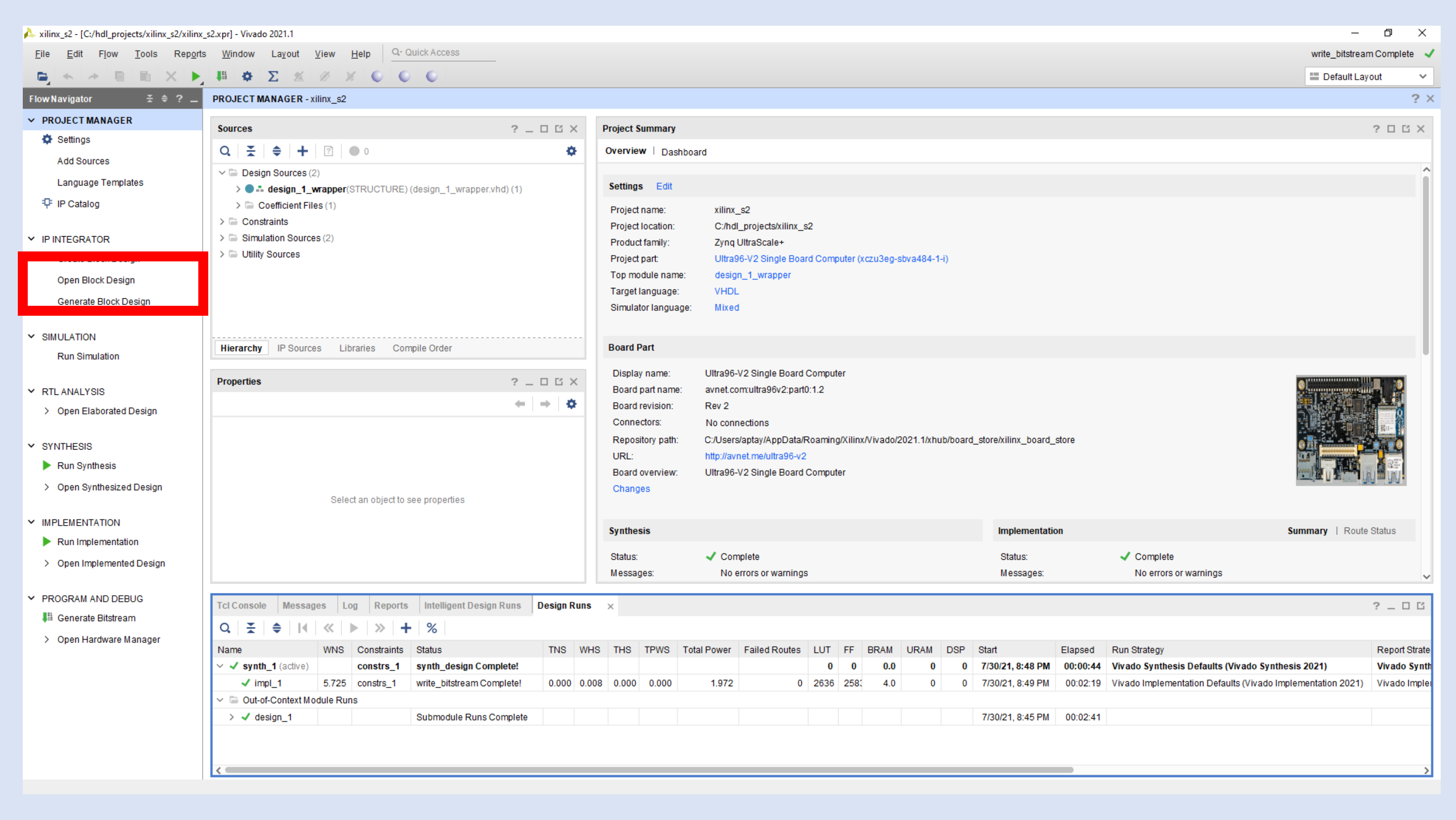
Project will be created at: C:/hdl_projects/xilinx_s2

☑ Include run results

□ Import all files to the new project

Lab 2: Understanding Vitis Project creation & Flow

Step 4 – Open the Block Diagram



Settings Edit

Project name:	xilinx_s2
Project location:	C:/hdl_projects/xilinx_s2
Product family:	Zynq UltraScale+
Project part:	Ultra96-V2 Single Board Computer (xczu3eg-sbva484-1-i)
Top module name:	design_1_wrapper
Target language:	VHDL
Simulator language:	Mixed

Board Part

Display name:	Ultra96-V2 Single Board Computer
Board part name:	avnet.com:ultra96v2:part0:1.2
Board revision:	Rev 2
Connectors:	No connections
Repository path:	C:/Users/aptay/AppData/Roaming/Xilinx/Vivado/2021.1/xhub/board_store/xilinx_board_store
URL:	http://avnet.me/ultra96-v2
Board overview:	Ultra96-V2 Single Board Computer

Changes

Synthesis

Status:	✓ Complete
Messages:	No errors or warnings

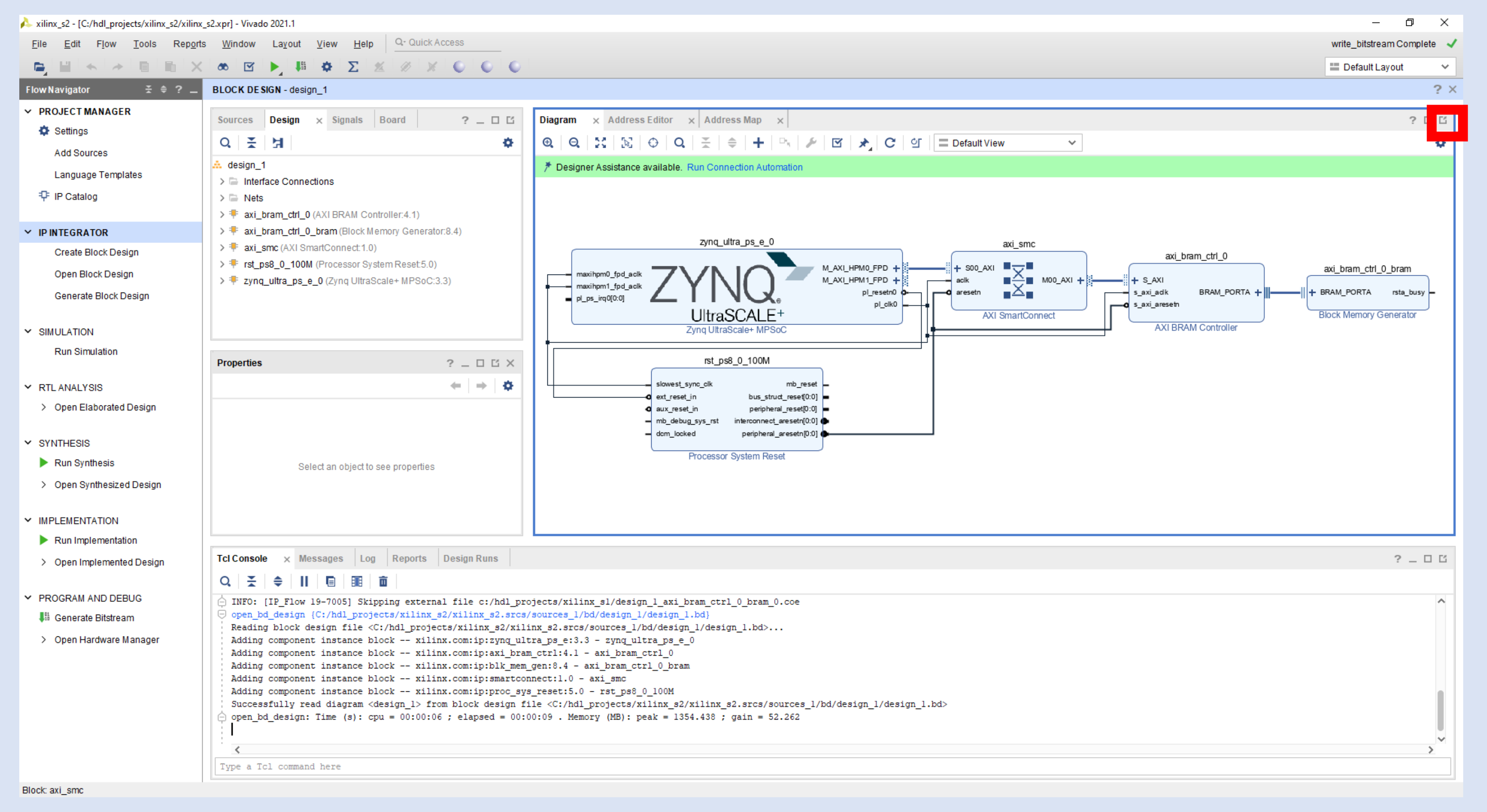
Implementation **Summary** | Route Status

Status:	✓ Complete
Messages:	No errors or warnings

Name	WNS	Constraints	Status	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Run Strategy	Report Strate
✓ synth_1 (active)		constrs_1	synth_design Complete!							0	0	0.0	0	0	7/30/21, 8:48 PM	00:00:44	Vivado Synthesis Defaults (Vivado Synthesis 2021)	Vivado Synth
✓ impl_1	5.725	constrs_1	write_bitstream Complete!	0.000	0.008	0.000	0.000	1.972	0	2636	258:	4.0	0	0	7/30/21, 8:49 PM	00:02:19	Vivado Implementation Defaults (Vivado Implementation 2021)	Vivado Imple
Out-of-Context Module Runs																		
✓ design_1			Submodule Runs Complete												7/30/21, 8:45 PM	00:02:41		

Lab 2: Understanding Vitis Project creation & Flow

Step 5 – Un dock the block diagram, and maximize it



design_1
- Interface Connections
- Nets
- axi_bram_ctrl_0 (AXI BRAM Controller:4.1)
- axi_bram_ctrl_0_bram (Block Memory Generator:8.4)
- axi_smc (AXI SmartConnect:1.0)
- rst_ps8_0_100M (Processor System Reset:5.0)
- zynq_ultra_ps_e_0 (Zynq UltraScale+ MPSoC:3.3)

Select an object to see properties

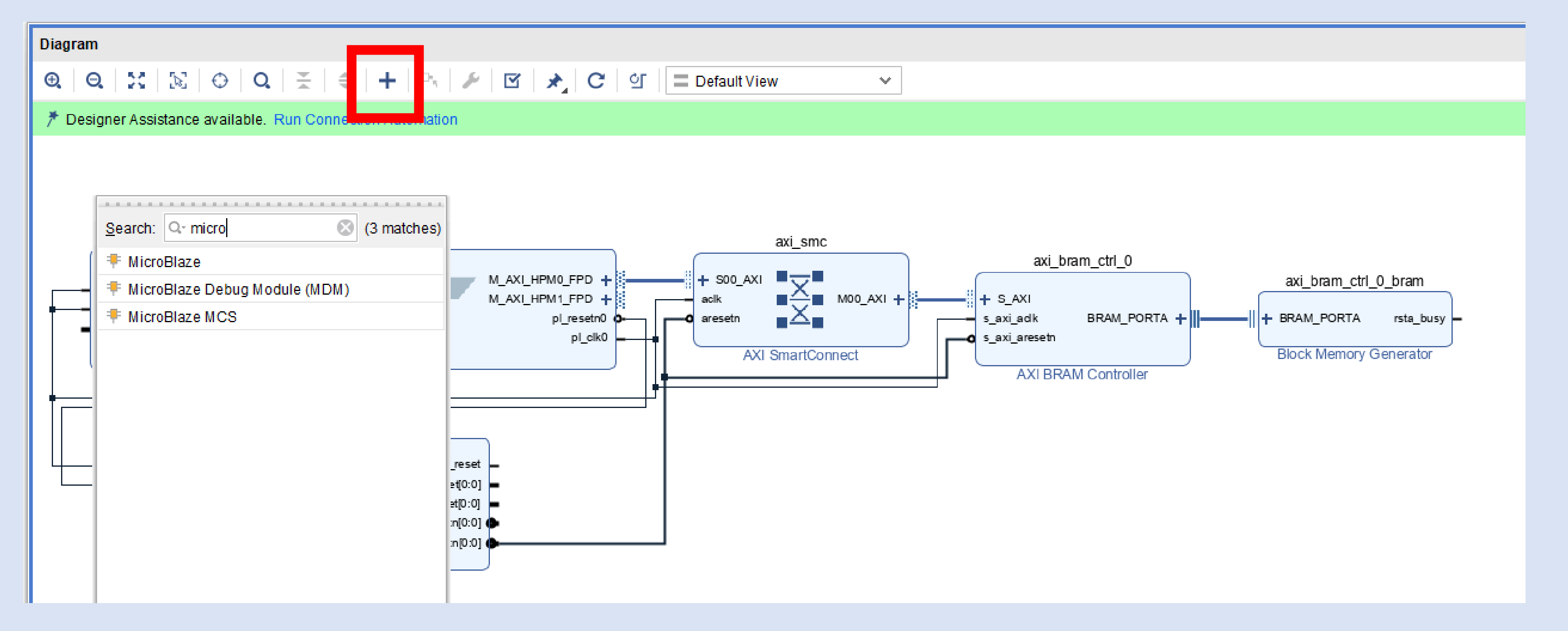
Designer Assistance available. Run Connection Automation

```
INFO: [IP_Flow 19-7005] Skipping external file c:/hdl_projects/xilinx_s1/design_1_axi_bram_ctrl_0_bram_0.coe
open_bd_design {C:/hdl_projects/xilinx_s2/xilinx_s2.srcs/sources_1/bd/design_1/design_1.bd}
Reading block design file <C:/hdl_projects/xilinx_s2/xilinx_s2.srcs/sources_1/bd/design_1/design_1.bd>...
Adding component instance block -- xilinx.com:ip:zynq_ultra_ps_e:3.3 - zynq_ultra_ps_e_0
Adding component instance block -- xilinx.com:ip:axi_bram_ctrl:4.1 - axi_bram_ctrl_0
Adding component instance block -- xilinx.com:ip:blk_mem_gen:8.4 - axi_bram_ctrl_0_bram
Adding component instance block -- xilinx.com:ip:smartconnect:1.0 - axi_smc
Adding component instance block -- xilinx.com:ip:proc_sys_reset:5.0 - rst_ps8_0_100M
Successfully read diagram <design_1> from block design file <C:/hdl_projects/xilinx_s2/xilinx_s2.srcs/sources_1/bd/design_1/design_1.bd>
open_bd_design: Time (s): cpu = 00:00:06 ; elapsed = 00:00:09 . Memory (MB): peak = 1354.438 ; gain = 52.262
```

Type a Tcl command here

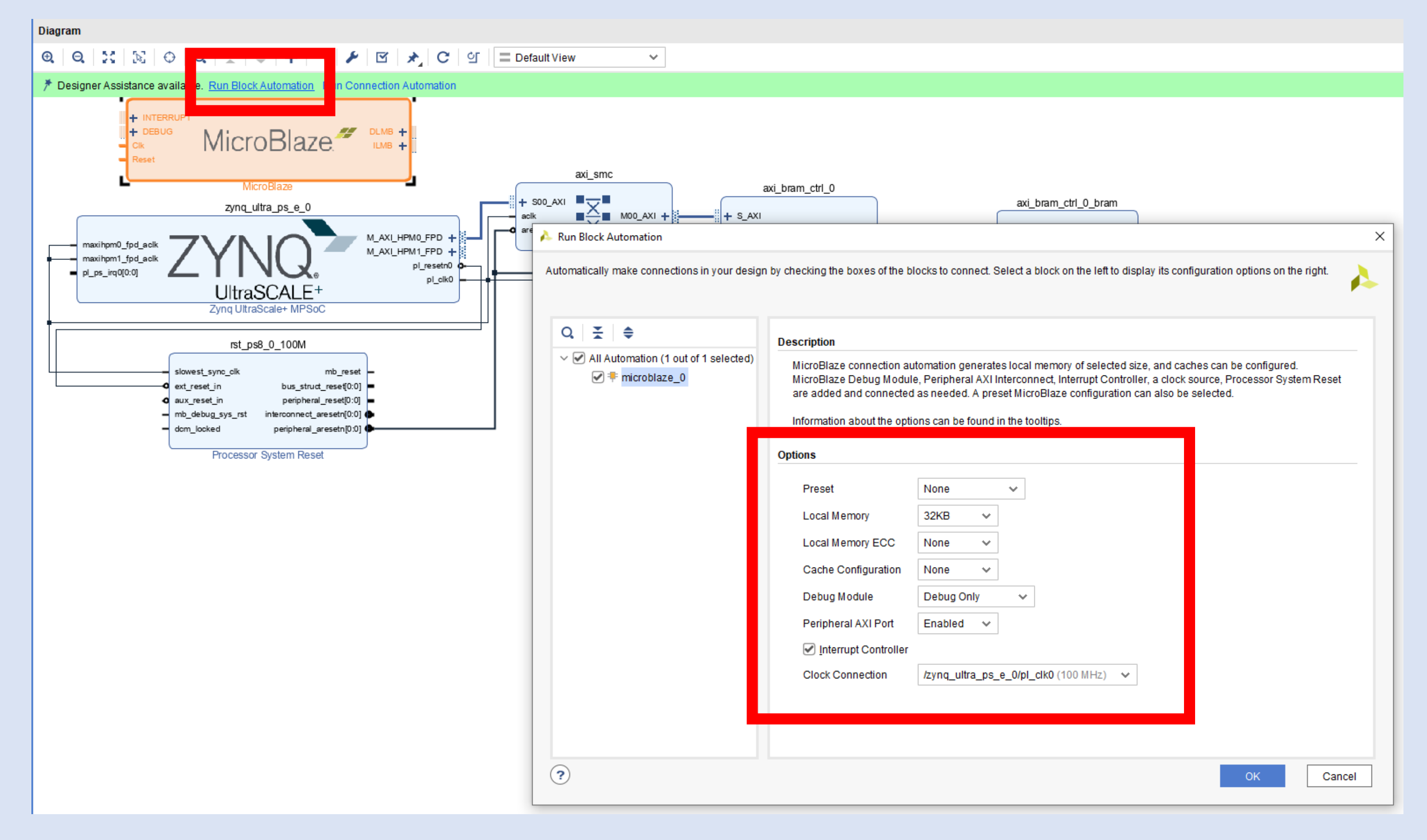
Lab 2: Understanding Vitis Project creation & Flow

Step 6 – Click on + and in the search bar, type microblaze – double click on the MicroBlaze IP to add it to IP Integrator



Lab 2: Understanding Vitis Project creation & Flow

Step 7 – Click on Run Block Automation (this will help you configure the MicroBlaze), Select Local Memory 32KB, No ECC, No Cache, Debug only, AXI enabled, check Interrupt, Clock 100 MHz click OK



Run Block Automation

Automatically make connections in your design by checking the boxes of the blocks to connect. Select a block on the left to display its configuration options on the right.

☑ All Automation (1 out of 1 selected)
☑ microblaze_0

Description

MicroBlaze connection automation generates local memory of selected size, and caches can be configured. MicroBlaze Debug Module, Peripheral AXI Interconnect, Interrupt Controller, a clock source, Processor System Reset are added and connected as needed. A preset MicroBlaze configuration can also be selected.

Information about the options can be found in the tooltips.

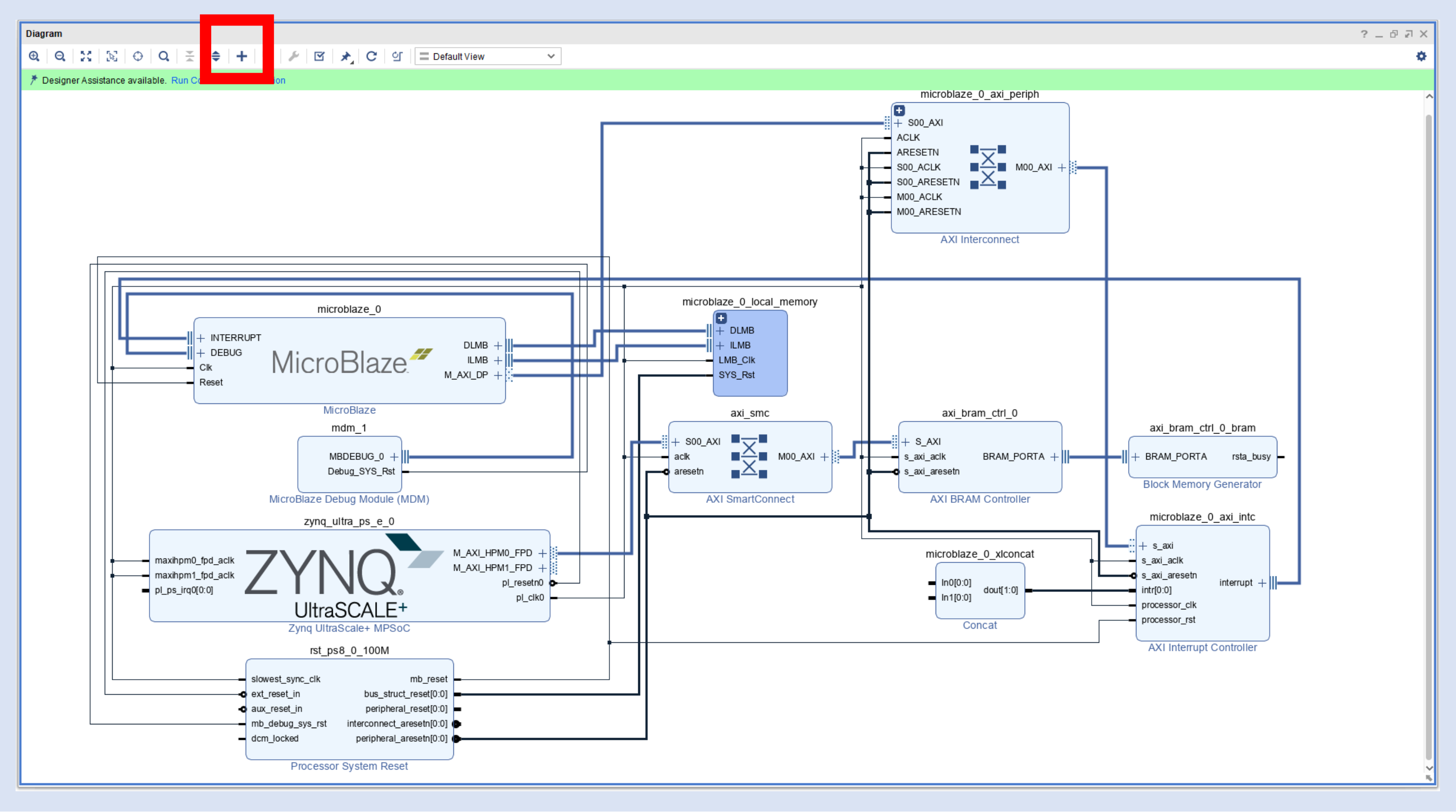
Options

Preset	None
Local Memory	32KB
Local Memory ECC	None
Cache Configuration	None
Debug Module	Debug Only
Peripheral AXI Port	Enabled
☑ Interrupt Controller	
Clock Connection	/zynq_ultra_ps_e_0/pl_clk0 (100 MHz)

OK Cancel

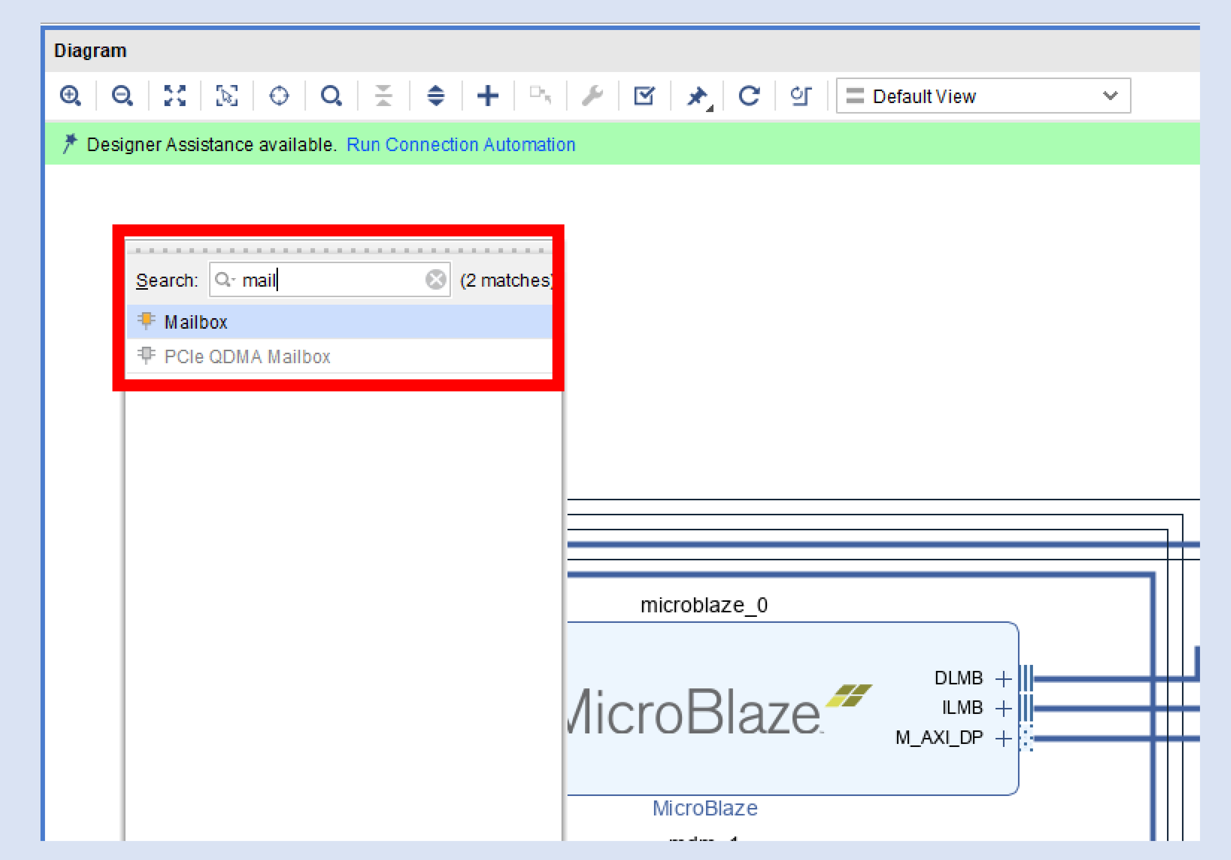
Lab 2: Understanding Vitis Project creation & Flow

Step 8 – Once the automation finishes, click on the + symbol



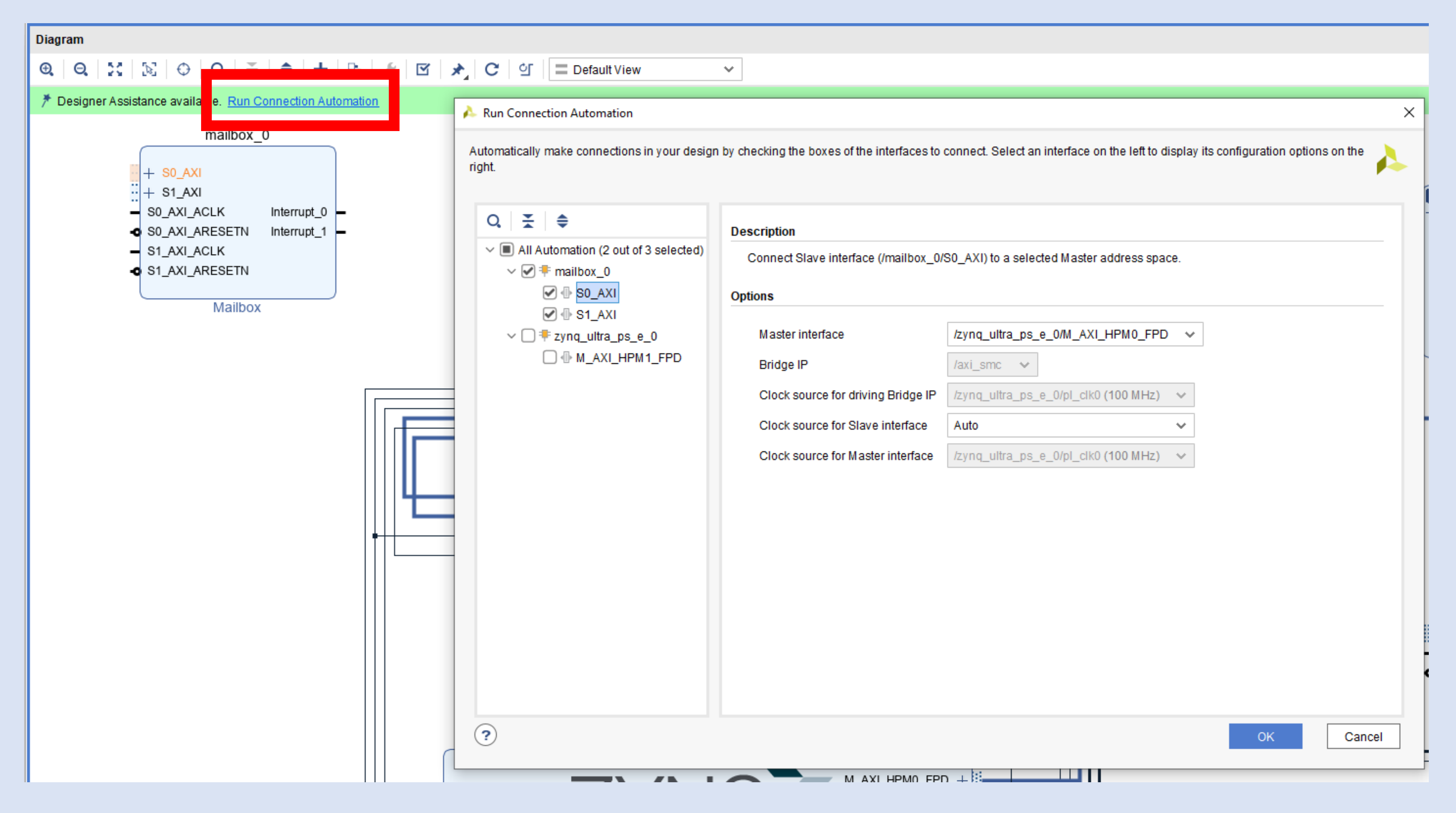
Lab 2: Understanding Vitis Project creation & Flow

Step 9 – Type in Mailbox, double click and add it to the diagram



Lab 2: Understanding Vitis Project creation & Flow

Step 10 – Click on Run Connection Automation, Check Mailbox. For S0_AXI set master interface to AXI HPM0 FPD on the Processing System



Run Connection Automation

Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.

- ■ All Automation (2 out of 3 selected)
  - ☑ mailbox_0
    - ☑ S0_AXI
    - ☑ S1_AXI
  - □ zynq_ultra_ps_e_0
    - □ M_AXI_HPM1_FPD

Description

Connect Slave interface (/mailbox_0/S0_AXI) to a selected Master address space.

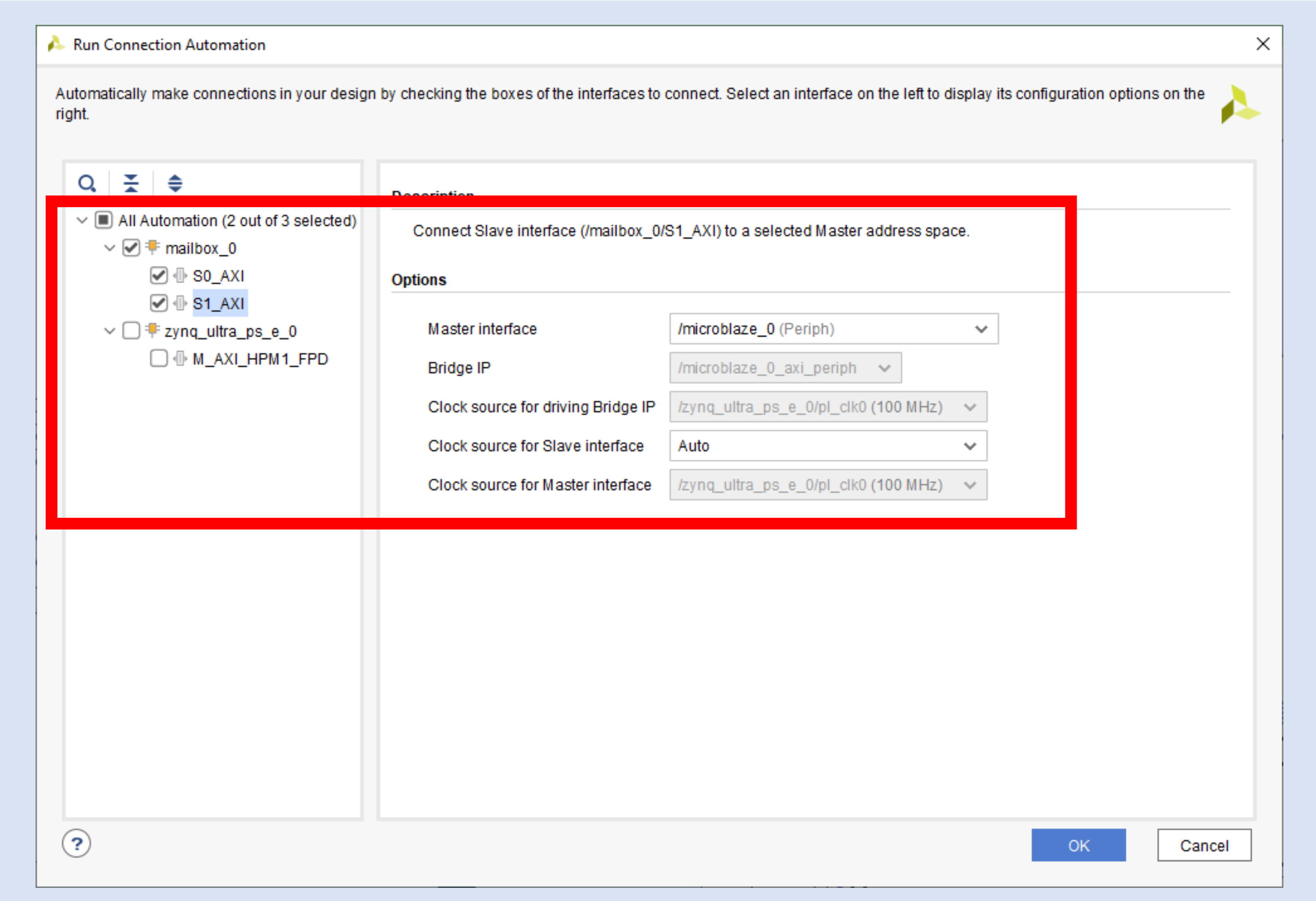
Options

Master interface	/zynq_ultra_ps_e_0/M_AXI_HPM0_FPD
Bridge IP	/axi_smc
Clock source for driving Bridge IP	/zynq_ultra_ps_e_0/pl_clk0 (100 MHz)
Clock source for Slave interface	Auto
Clock source for Master interface	/zynq_ultra_ps_e_0/pl_clk0 (100 MHz)

OK Cancel

Lab 2: Understanding Vitis Project creation & Flow

Step 11 – Click on S1_AXI and select the master interface to be the MicroBlaze system.



Run Connection Automation

Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.

- ▣ All Automation (2 out of 3 selected)
  - ☑ mailbox_0
    - ☑ S0_AXI
    - ☑ S1_AXI
  - □ zynq_ultra_ps_e_0
    - □ M_AXI_HPM1_FPD

Description

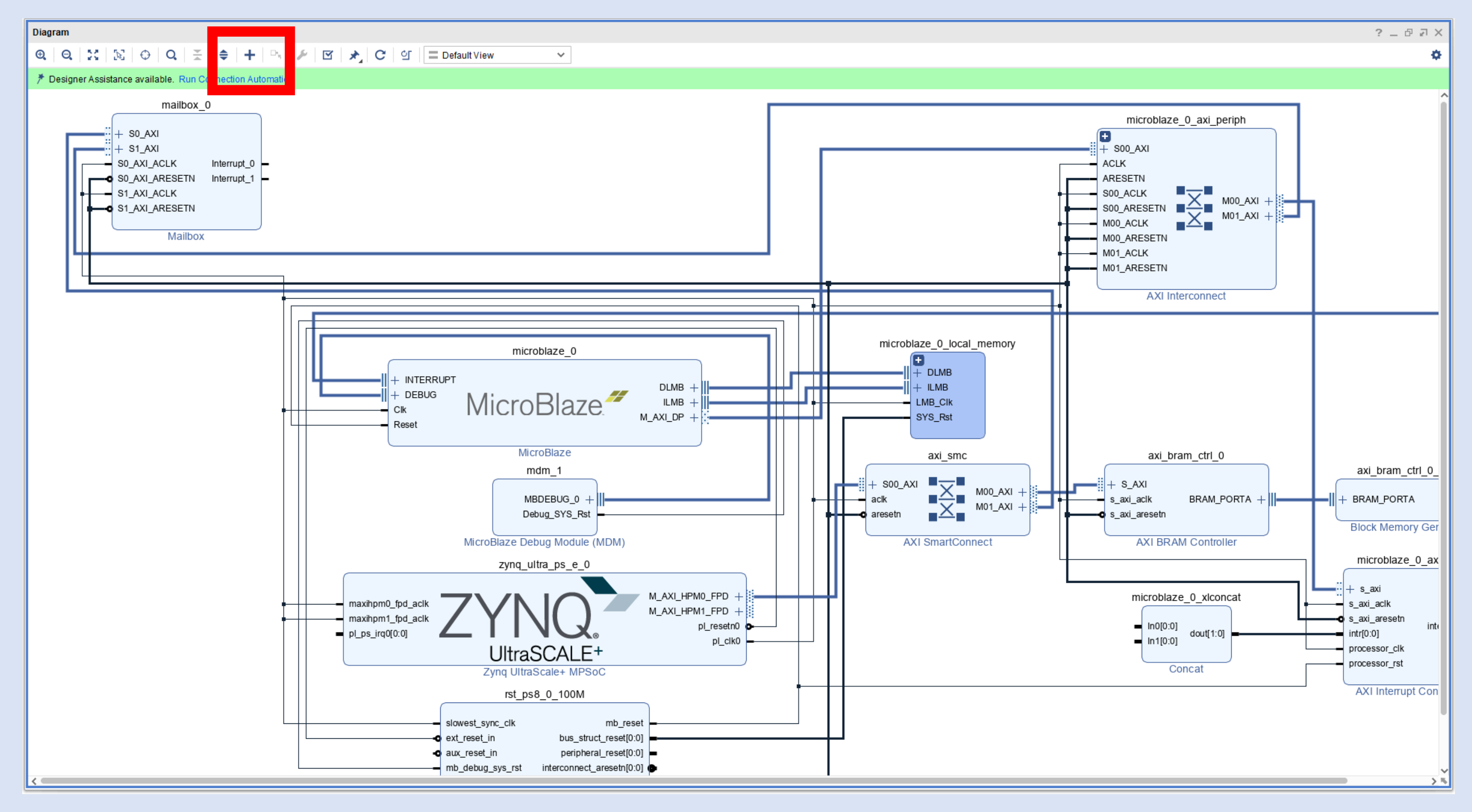
Connect Slave interface (/mailbox_0/S1_AXI) to a selected Master address space.

Options

Master interface	/microblaze_0 (Periph)
Bridge IP	/microblaze_0_axi_periph
Clock source for driving Bridge IP	/zynq_ultra_ps_e_0/pl_clk0 (100 MHz)
Clock source for Slave interface	Auto
Clock source for Master interface	/zynq_ultra_ps_e_0/pl_clk0 (100 MHz)

Lab 2: Understanding Vitis Project creation & Flow

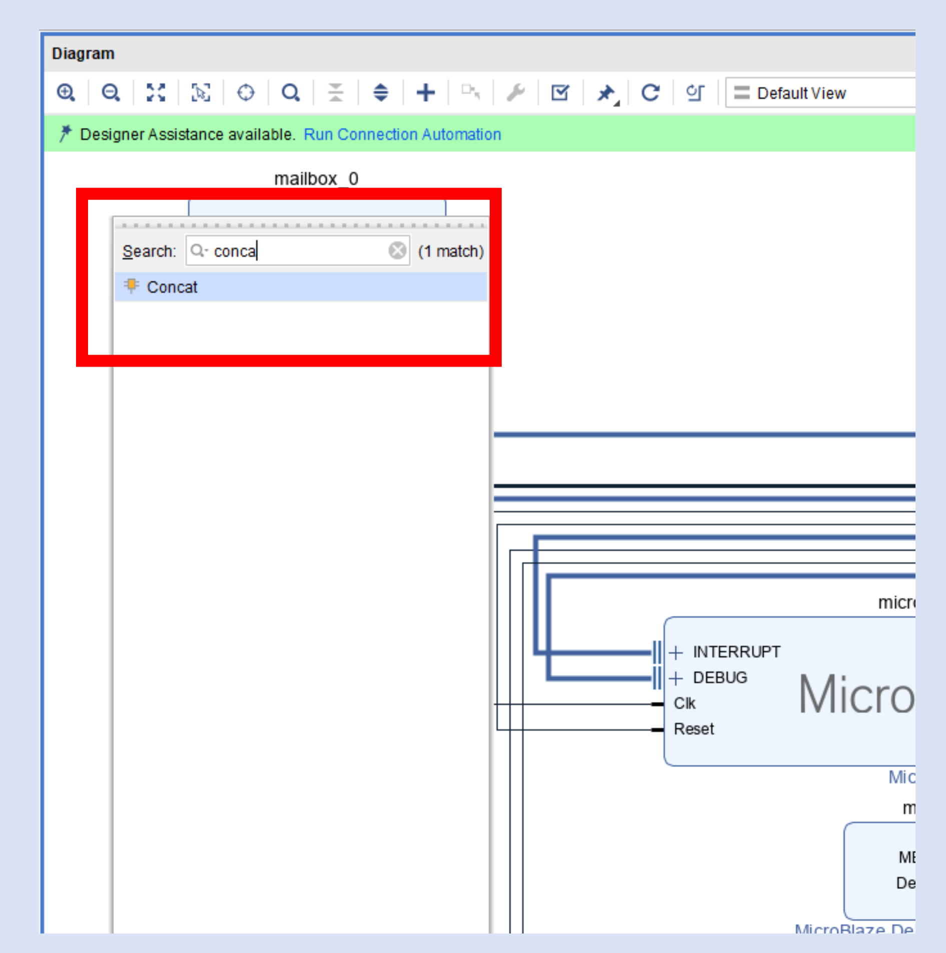
Step 12 – Click +



Designer Assistance available. Run Co[n]nection Automati

Lab 2: Understanding Vitis Project creation & Flow

Step 13 – Type Concat and double click to add in the concatenation block



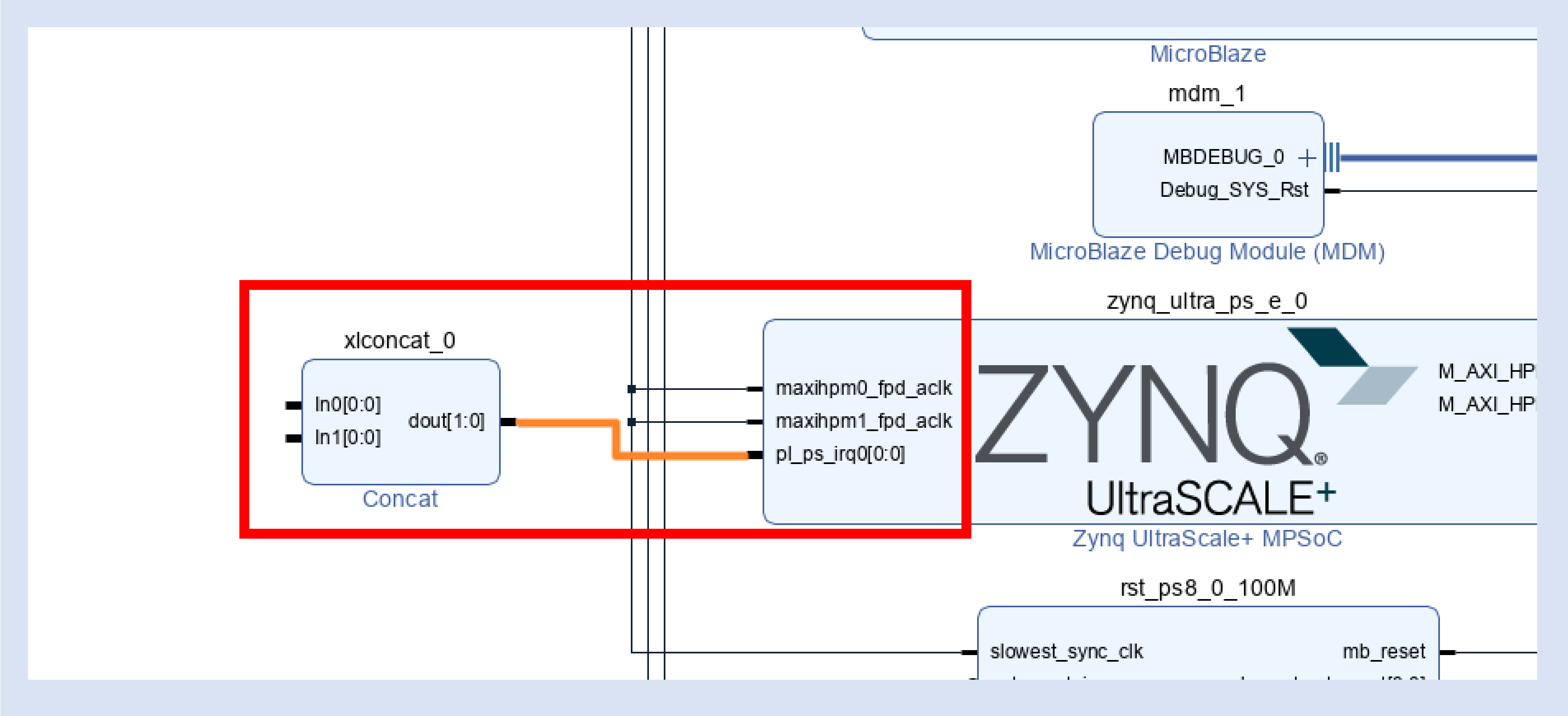
Designer Assistance available. Run Connection Automation

Search: conca (1 match)

Concat

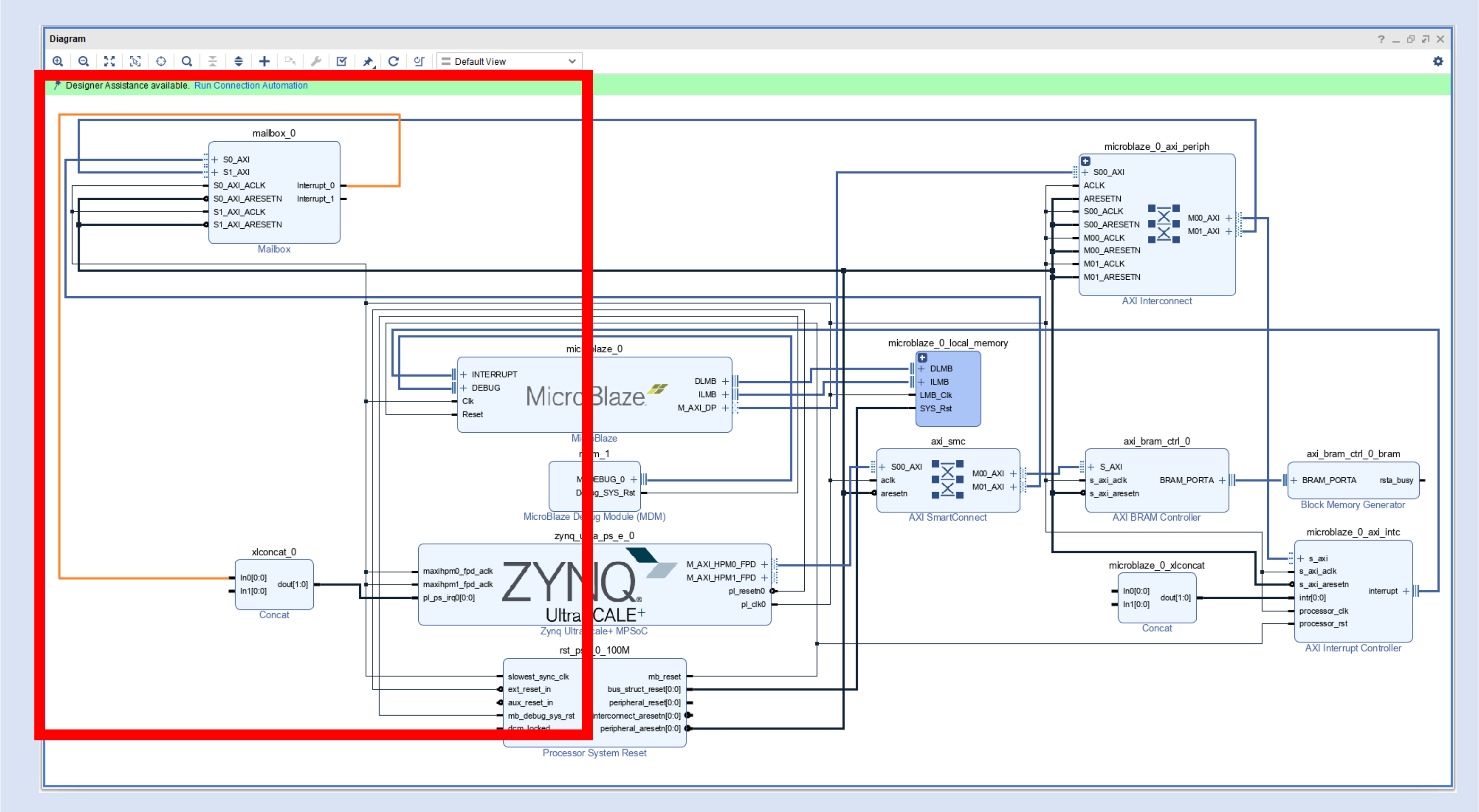
Lab 2: Understanding Vitis Project creation & Flow

Step 14 – Connect the Concat Block to the PL_PS_IRQ input



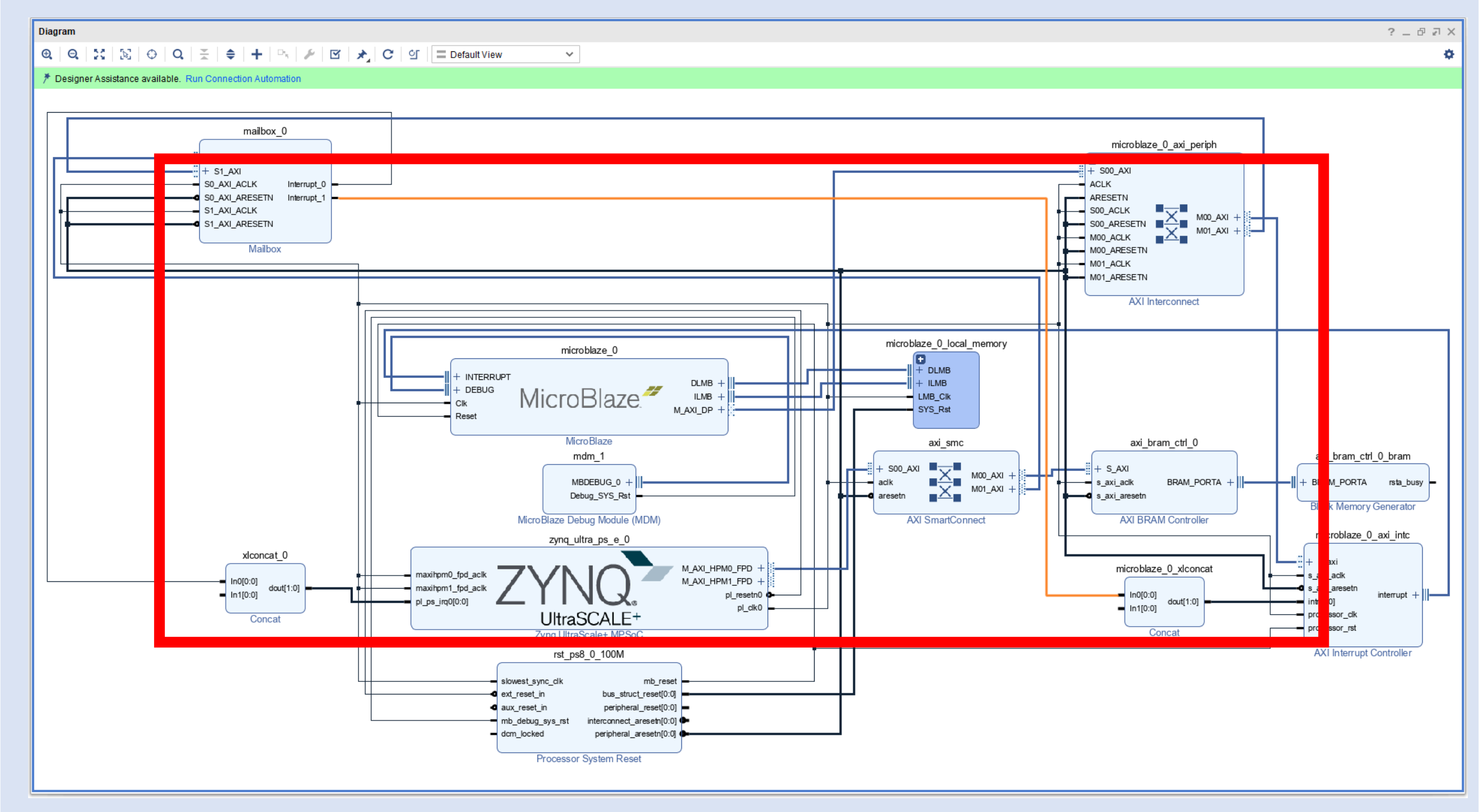
Lab 2: Understanding Vitis Project creation & Flow

Step 15 – Connect the interrupt one output from the mailbox to the concat block just added.



Lab 2: Understanding Vitis Project creation & Flow

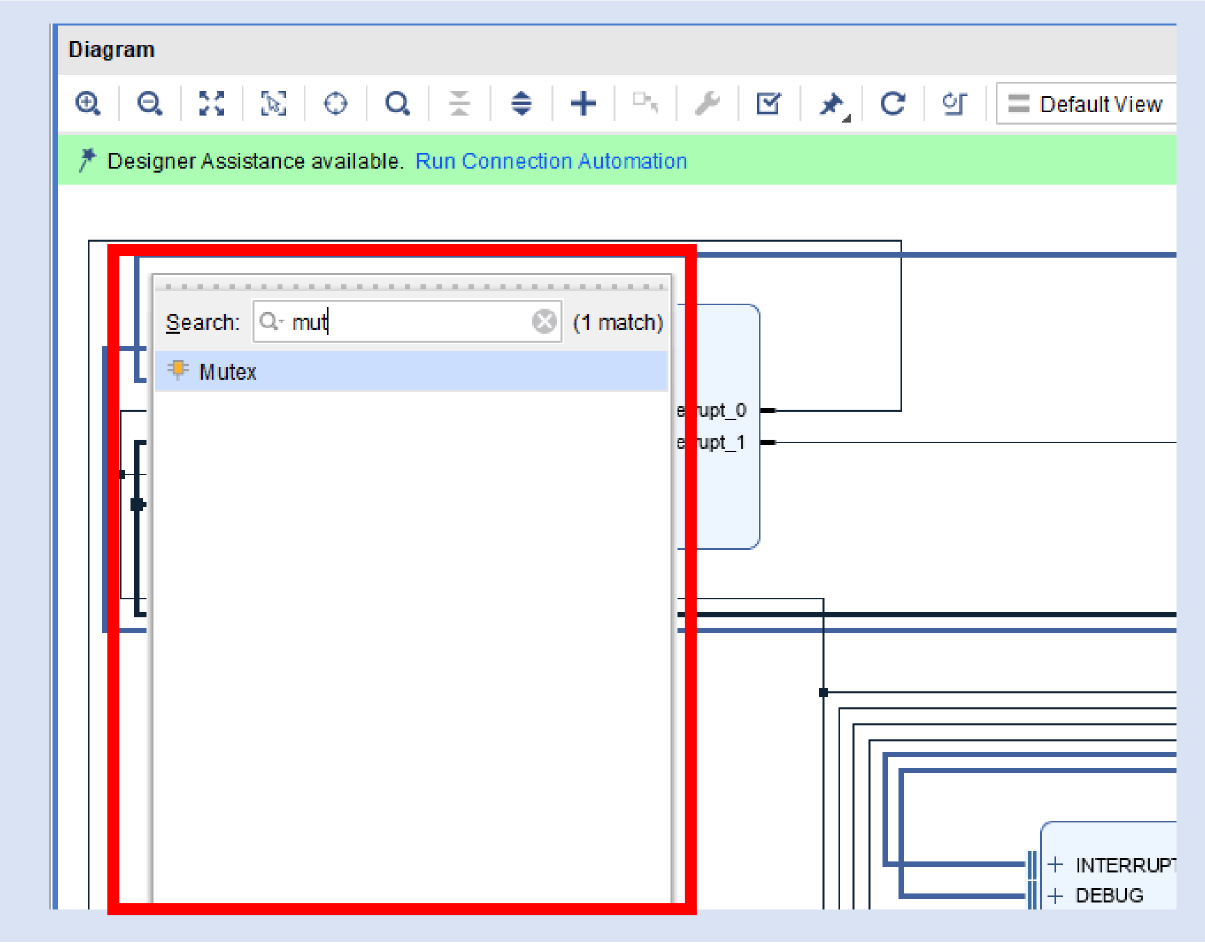
Step 16 – Connect the interrupt two output from the mailbox to the concat block for the MicroBlaze.



Designer Assistance available. Run Connection Automation

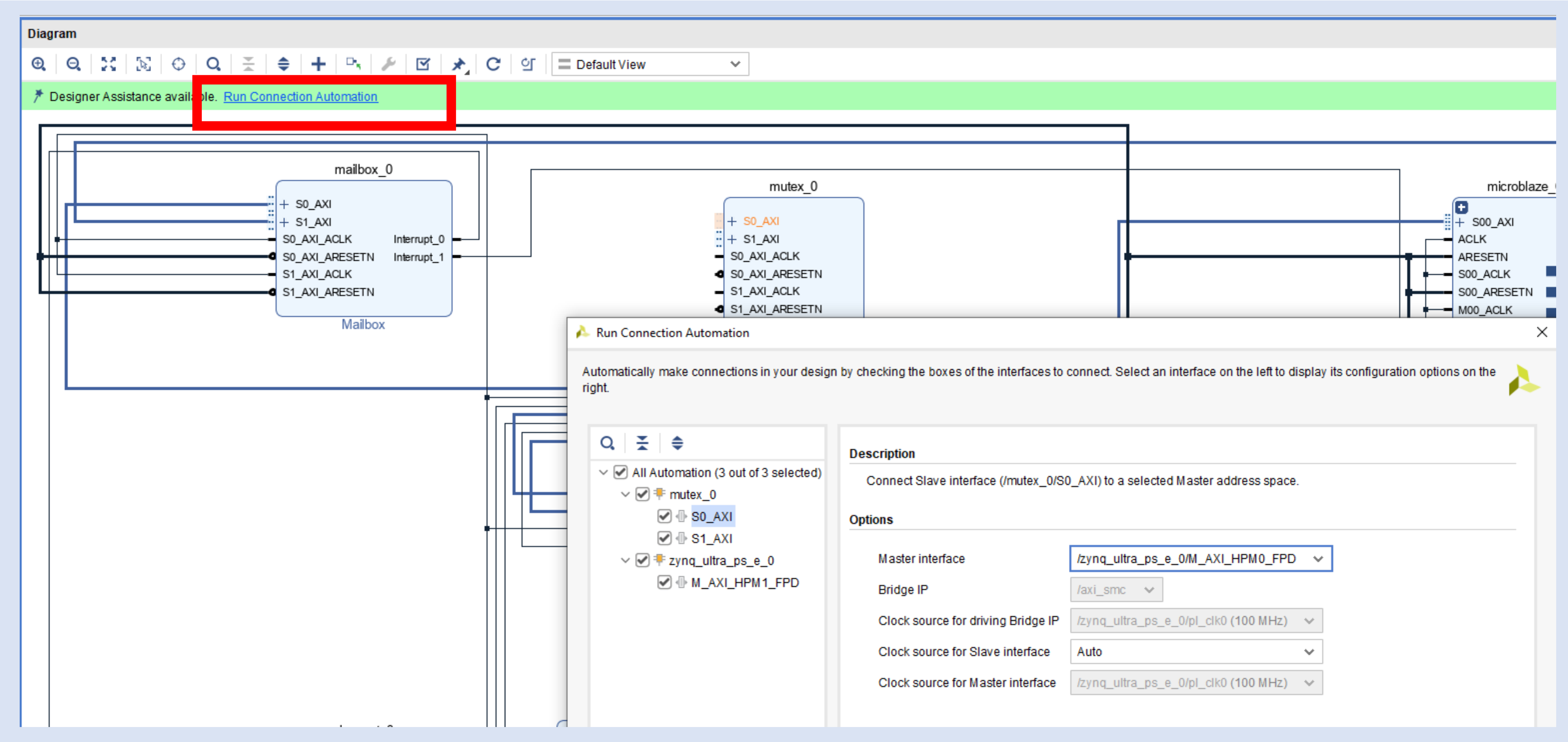
Lab 2: Understanding Vitis Project creation & Flow

Step 17 – Click on + and type Mutex, double click to add it in to IP integrator.



Lab 2: Understanding Vitis Project creation & Flow

Step 18 – Run the block automation, check the Mutex and for S0_AXI select the AXI HMP0 FPD.



Designer Assistance available. Run Connection Automation

Run Connection Automation

Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.

- ☑ All Automation (3 out of 3 selected)
  - ☑ mutex_0
    - ☑ S0_AXI
    - ☑ S1_AXI
  - ☑ zynq_ultra_ps_e_0
    - ☑ M_AXI_HPM1_FPD

Description

Connect Slave interface (/mutex_0/S0_AXI) to a selected Master address space.

Options

Master interface: /zynq_ultra_ps_e_0/M_AXI_HPM0_FPD

Bridge IP: /axi_smc

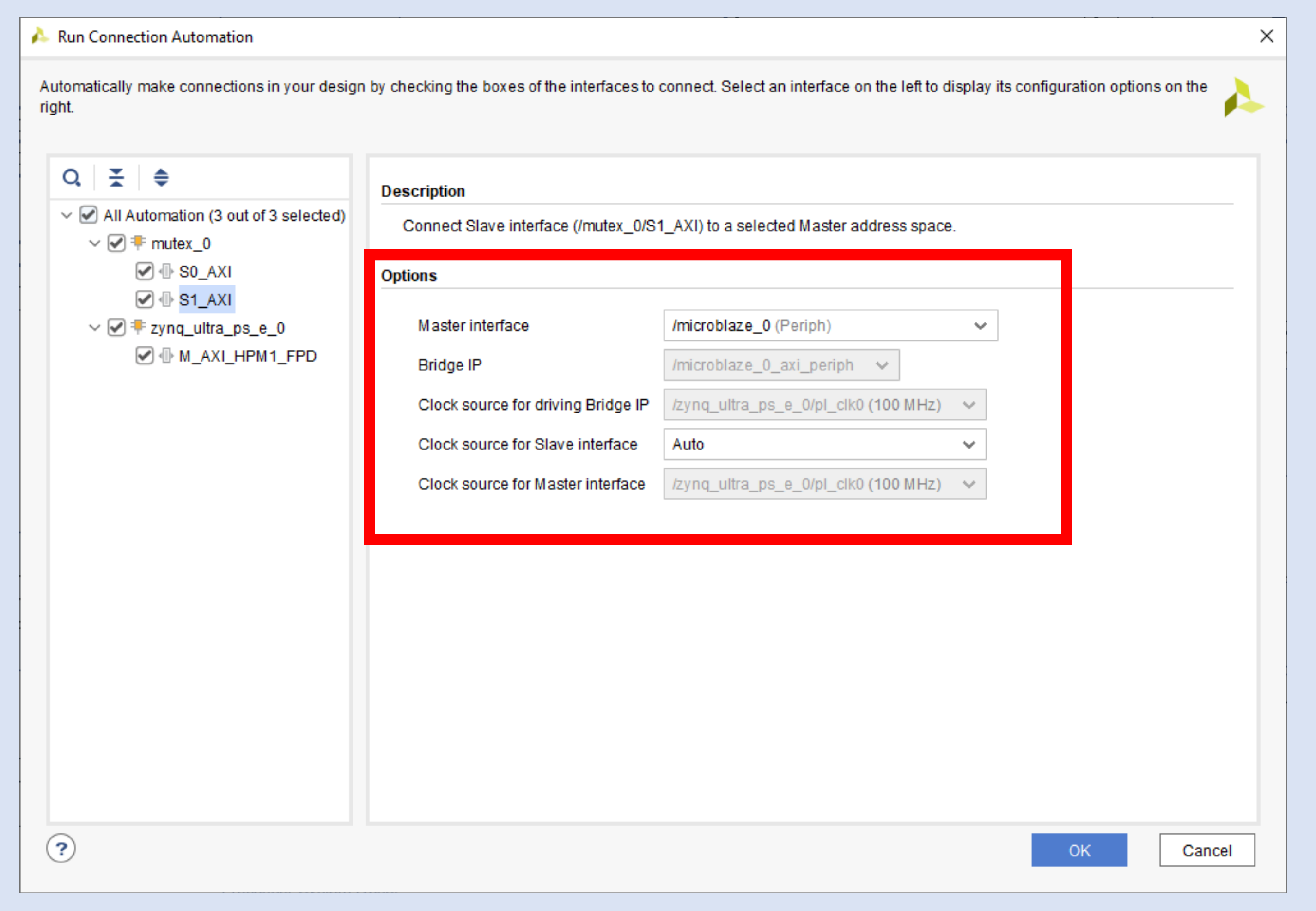
Clock source for driving Bridge IP: /zynq_ultra_ps_e_0/pl_clk0 (100 MHz)

Clock source for Slave interface: Auto

Clock source for Master interface: /zynq_ultra_ps_e_0/pl_clk0 (100 MHz)

Lab 2: Understanding Vitis Project creation & Flow

Step 19 – Select MicroBlaze for the Master Interface on S1 AXI.



Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.

- ☑ All Automation (3 out of 3 selected)
  - ☑ mutex_0
    - ☑ S0_AXI
    - ☑ S1_AXI
  - ☑ zynq_ultra_ps_e_0
    - ☑ M_AXI_HPM1_FPD

Description

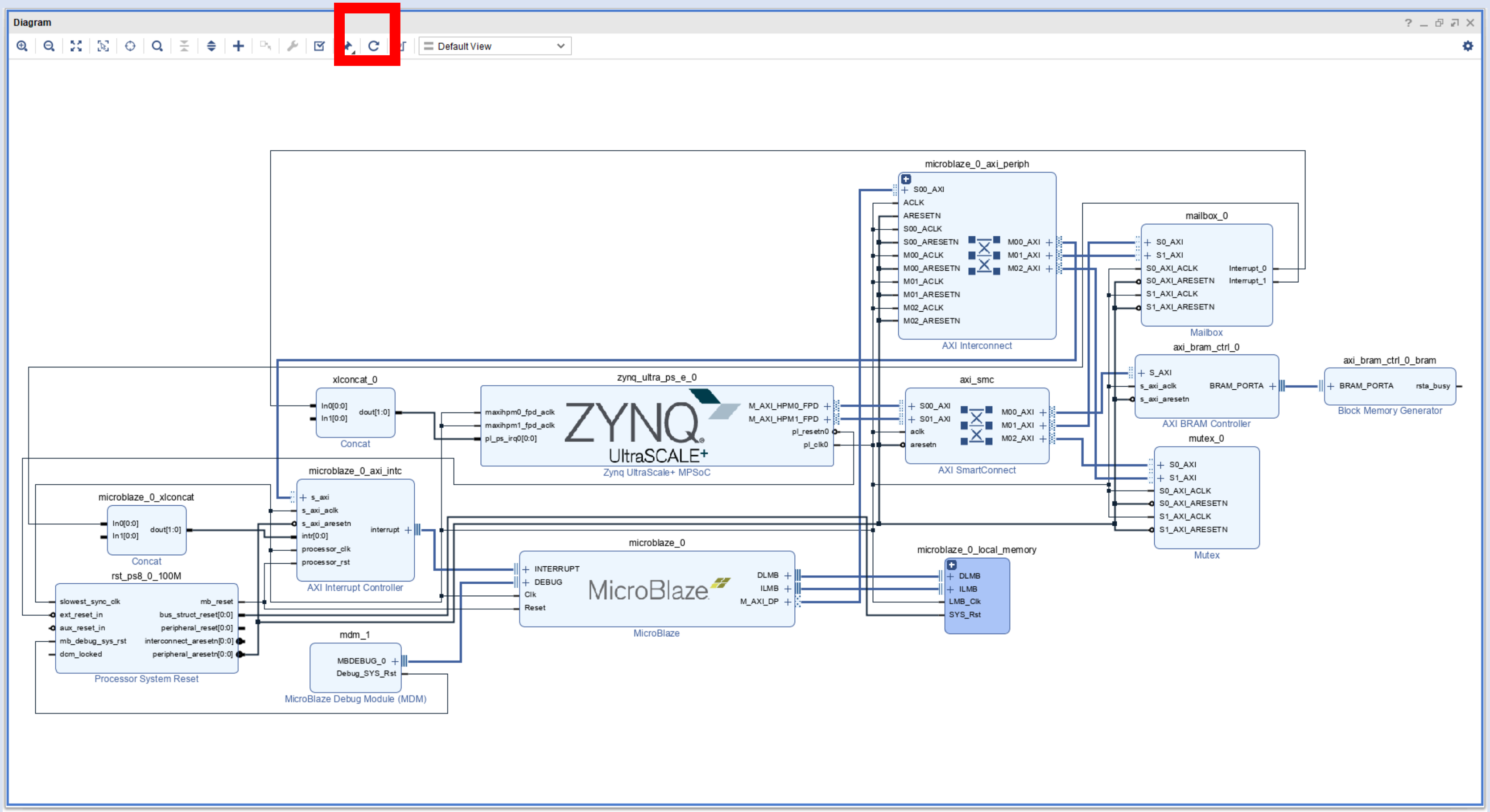
Connect Slave interface (/mutex_0/S1_AXI) to a selected Master address space.

Options

Master interface	/microblaze_0 (Periph)
Bridge IP	/microblaze_0_axi_periph
Clock source for driving Bridge IP	/zynq_ultra_ps_e_0/pl_clk0 (100 MHz)
Clock source for Slave interface	Auto
Clock source for Master interface	/zynq_ultra_ps_e_0/pl_clk0 (100 MHz)

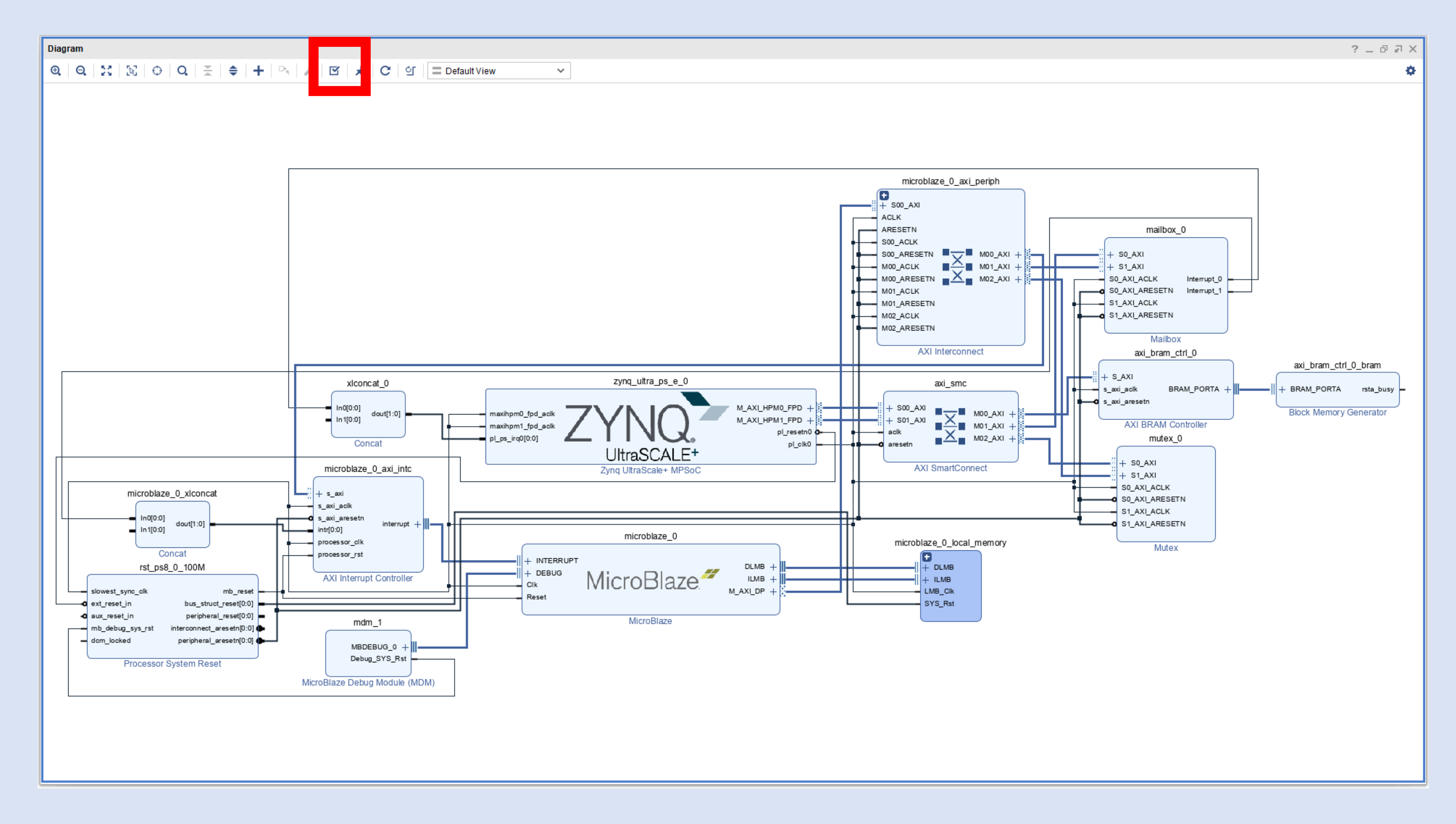
Lab 2: Understanding Vitis Project creation & Flow

Step 20 – Hit the regenerate layout button to create the layout as below



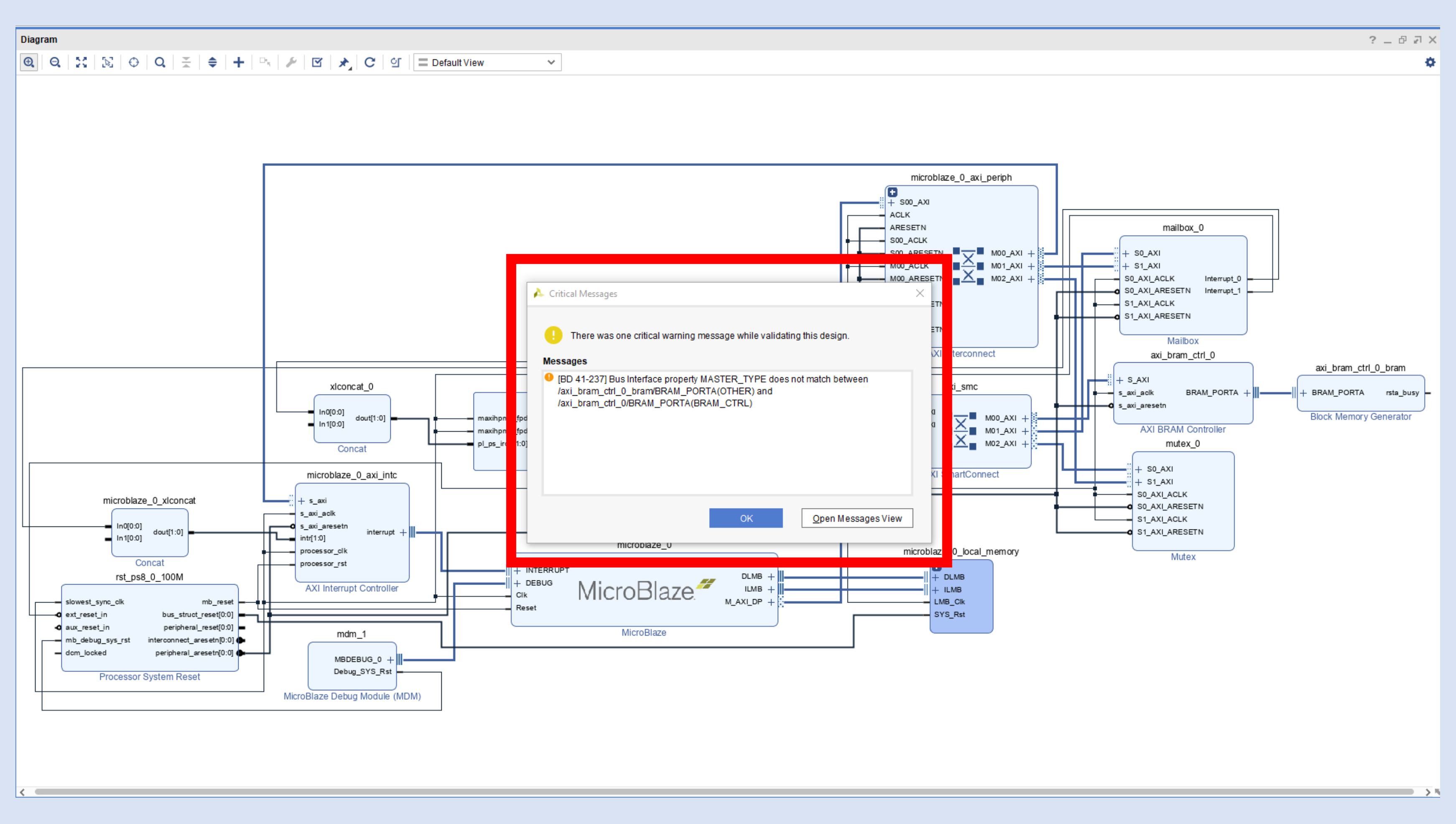
Lab 2: Understanding Vitis Project creation & Flow

Step 21 – Validate the design



Lab 2: Understanding Vitis Project creation & Flow

Step 22 – The only warning messages should be those seen before



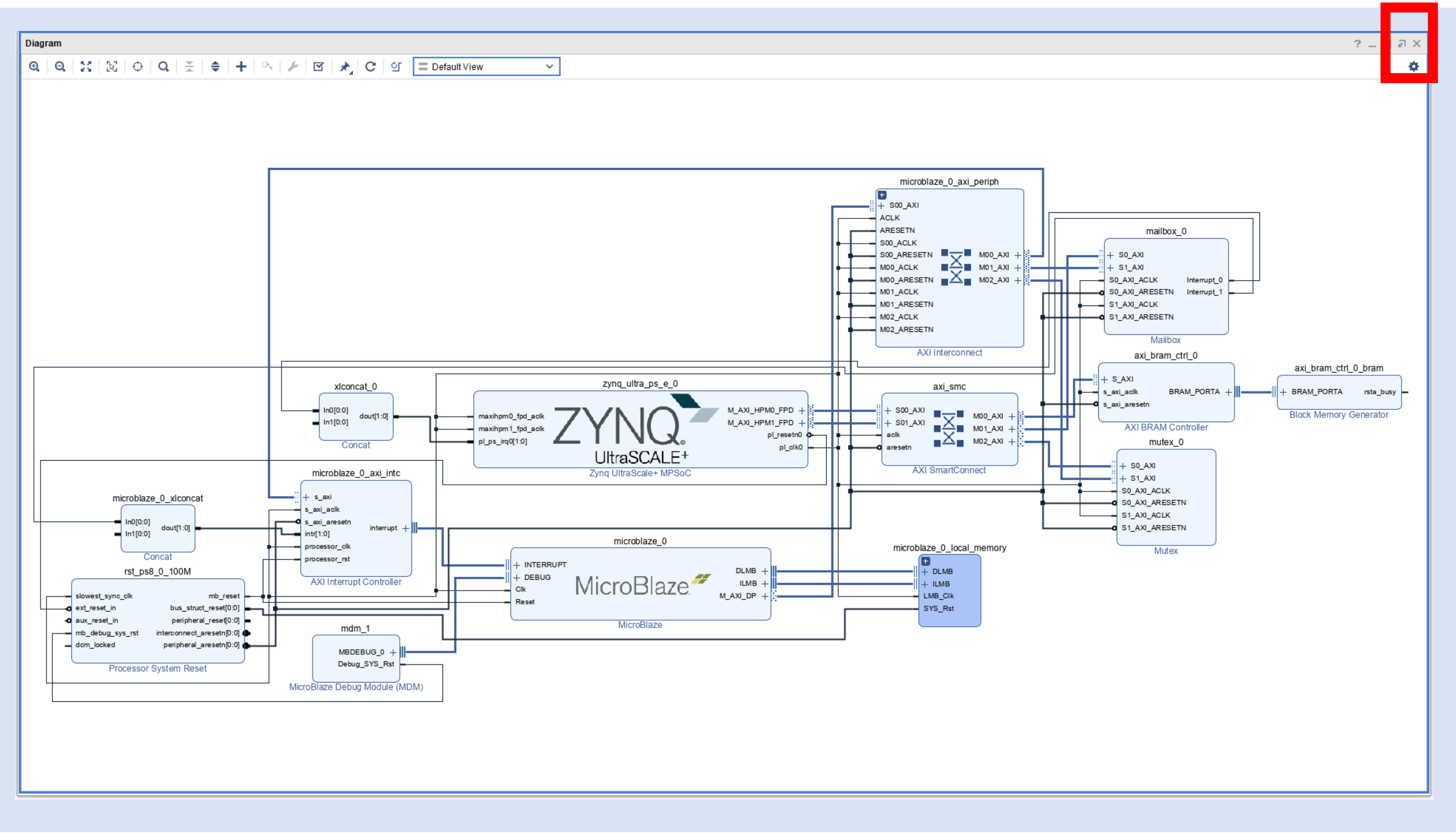
There was one critical warning message while validating this design.

Messages

[BD 41-237] Bus Interface property MASTER_TYPE does not match between /axi_bram_ctrl_0_bram/BRAM_PORTA(OTHER) and /axi_bram_ctrl_0/BRAM_PORTA(BRAM_CTRL)

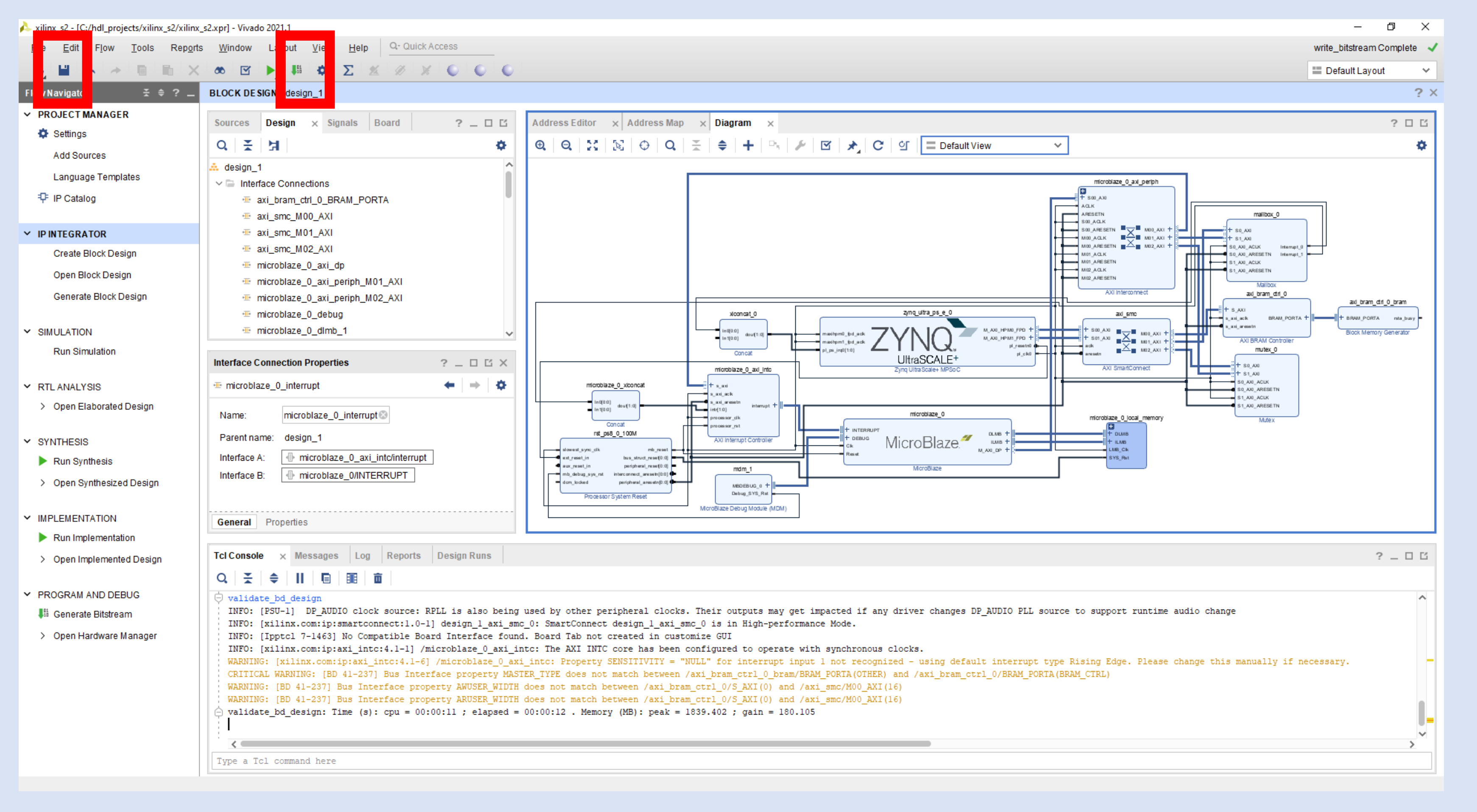
Lab 2: Understanding Vitis Project creation & Flow

Step 23 – Re Dock the IP Integrator window



Lab 2: Understanding Vitis Project creation & Flow

Step 24 – Save the design and then re generate the bit stream



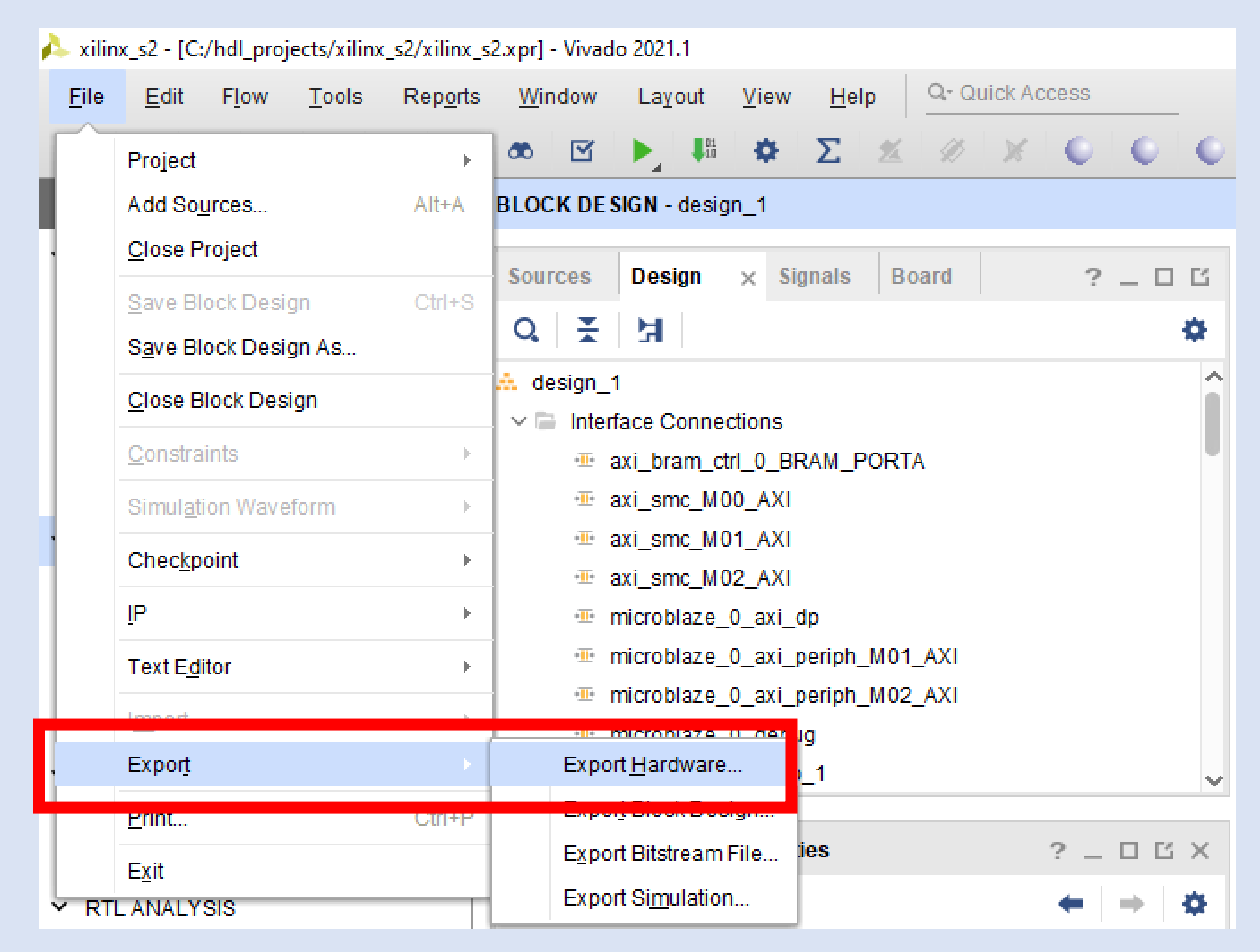
Name: microblaze_0_interrupt
Parent name: design_1
Interface A: microblaze_0_axi_intc/interrupt
Interface B: microblaze_0/INTERRUPT

```
validate_bd_design
INFO: [PSU-1]  DP_AUDIO clock source: RPLL is also being used by other peripheral clocks. Their outputs may get impacted if any driver changes DP_AUDIO PLL source to support runtime audio change
INFO: [xilinx.com:ip:smartconnect:1.0-1] design_1_axi_smc_0: SmartConnect design_1_axi_smc_0 is in High-performance Mode.
INFO: [Ipptcl 7-1463] No Compatible Board Interface found. Board Tab not created in customize GUI
INFO: [xilinx.com:ip:axi_intc:4.1-1] /microblaze_0_axi_intc: The AXI INTC core has been configured to operate with synchronous clocks.
WARNING: [xilinx.com:ip:axi_intc:4.1-6] /microblaze_0_axi_intc: Property SENSITIVITY = "NULL" for interrupt input 1 not recognized - using default interrupt type Rising Edge. Please change this manually if necessary.
CRITICAL WARNING: [BD 41-237] Bus Interface property MASTER_TYPE does not match between /axi_bram_ctrl_0_bram/BRAM_PORTA(OTHER) and /axi_bram_ctrl_0/BRAM_PORTA(BRAM_CTRL)
WARNING: [BD 41-237] Bus Interface property AWUSER_WIDTH does not match between /axi_bram_ctrl_0/S_AXI(0) and /axi_smc/M00_AXI(16)
WARNING: [BD 41-237] Bus Interface property ARUSER_WIDTH does not match between /axi_bram_ctrl_0/S_AXI(0) and /axi_smc/M00_AXI(16)
validate_bd_design: Time (s): cpu = 00:00:11 ; elapsed = 00:00:12 . Memory (MB): peak = 1839.402 ; gain = 180.105
```

Type a Tcl command here

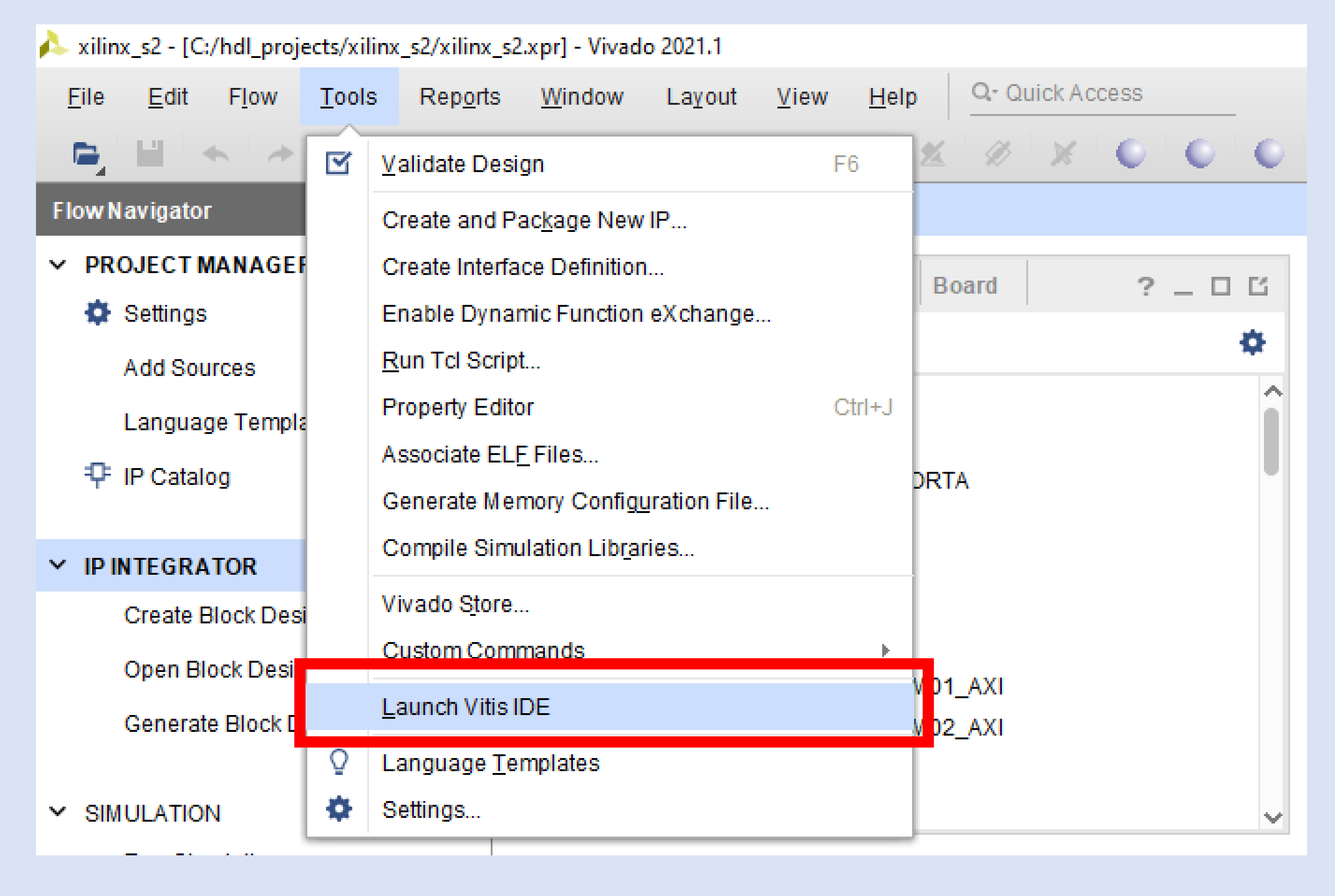
Lab 2: Understanding Vitis Project creation & Flow

Step 24 – Once the bitstream is complete, export the bit stream with the bit file to the project directory



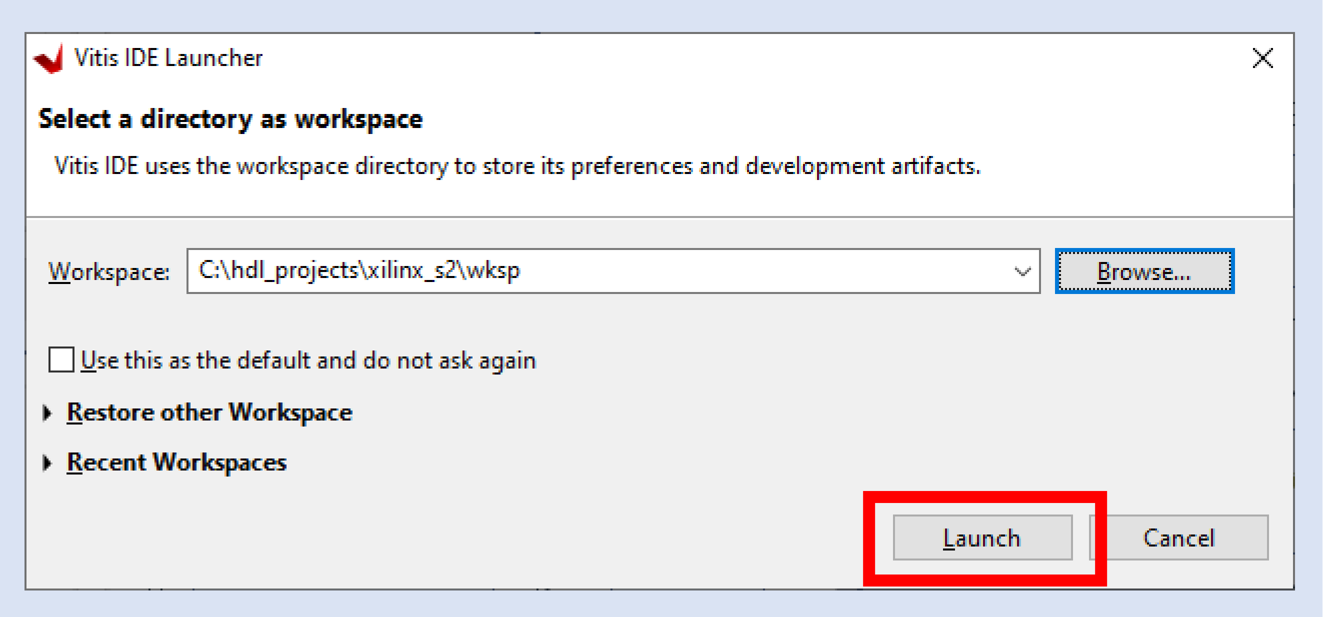
Lab 2: Understanding Vitis Project creation & Flow

Step 25 – From Tools select launch Vitis IDE



Lab 2: Understanding Vitis Project creation & Flow

Step 26 – From Tools select launch Vitis IDE, create a new workspace inside the project directory.



Select a directory as workspace

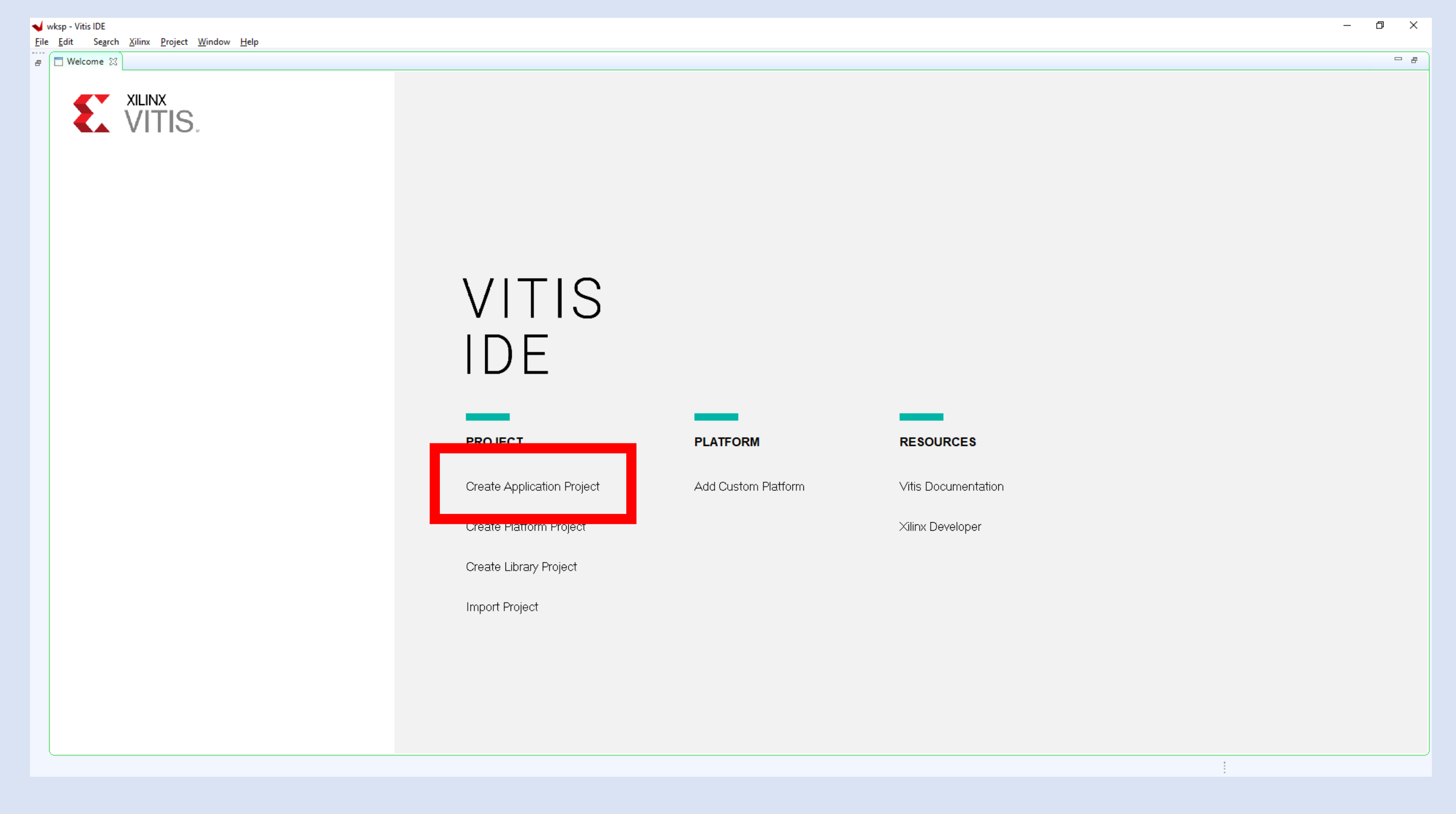
Vitis IDE uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\hdl_projects\xilinx_s2\wksp

□ Use this as the default and do not ask again

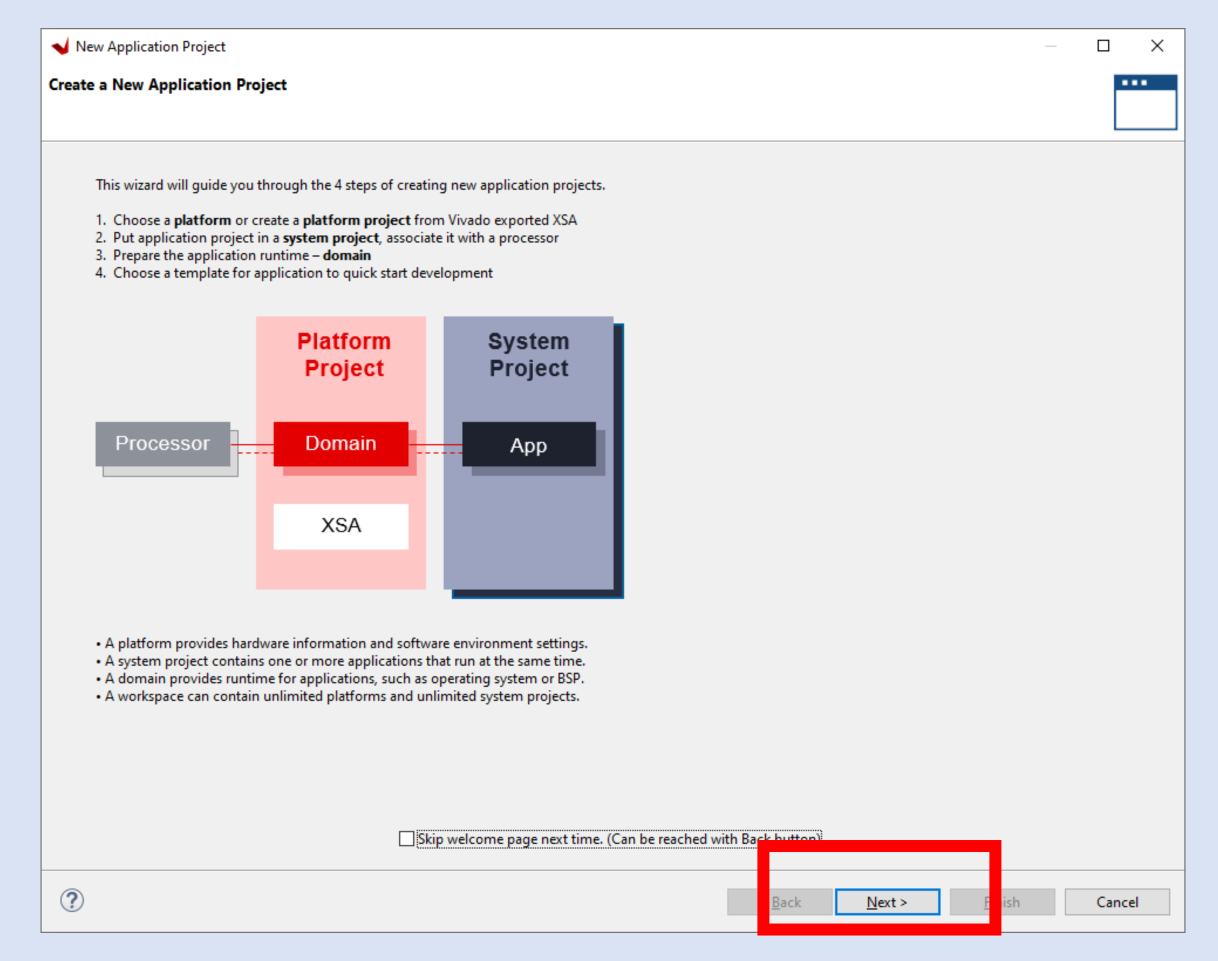
Lab 2: Understanding Vitis Project creation & Flow

Step 27 – Select Create Application Project



Lab 2: Understanding Vitis Project creation & Flow

Step 28 – Click next



New Application Project

Create a New Application Project

This wizard will guide you through the 4 steps of creating new application projects.

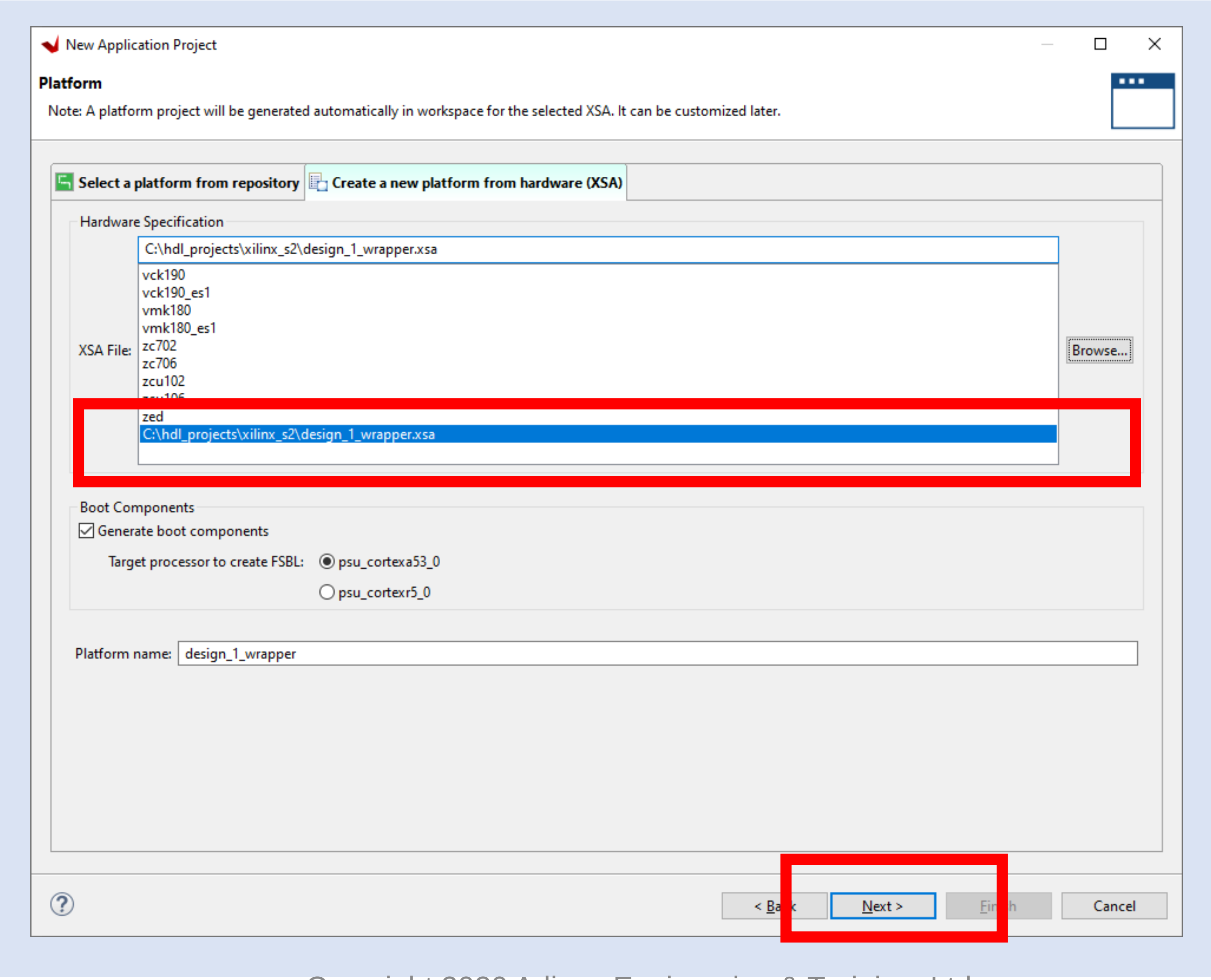
1. Choose a **platform** or create a **platform project** from Vivado exported XSA
2. Put application project in a **system project**, associate it with a processor
3. Prepare the application runtime – **domain**
4. Choose a template for application to quick start development

- A platform provides hardware information and software environment settings.
- A system project contains one or more applications that run at the same time.
- A domain provides runtime for applications, such as operating system or BSP.
- A workspace can contain unlimited platforms and unlimited system projects.

☐ Skip welcome page next time. (Can be reached with Back button)

Lab 2: Understanding Vitis Project creation & Flow

Step 29 – Click Select Create a new platform from hardware – select the XSA previously exported



New Application Project

Platform

Note: A platform project will be generated automatically in workspace for the selected XSA. It can be customized later.

Select a platform from repository
Create a new platform from hardware (XSA)

Hardware Specification

XSA File: C:\hdl_projects\xilinx_s2\design_1_wrapper.xsa

- vck190
- vck190_es1
- vmk180
- vmk180_es1
- zc702
- zc706
- zcu102
- zcu106
- zed
- C:\hdl_projects\xilinx_s2\design_1_wrapper.xsa

Boot Components

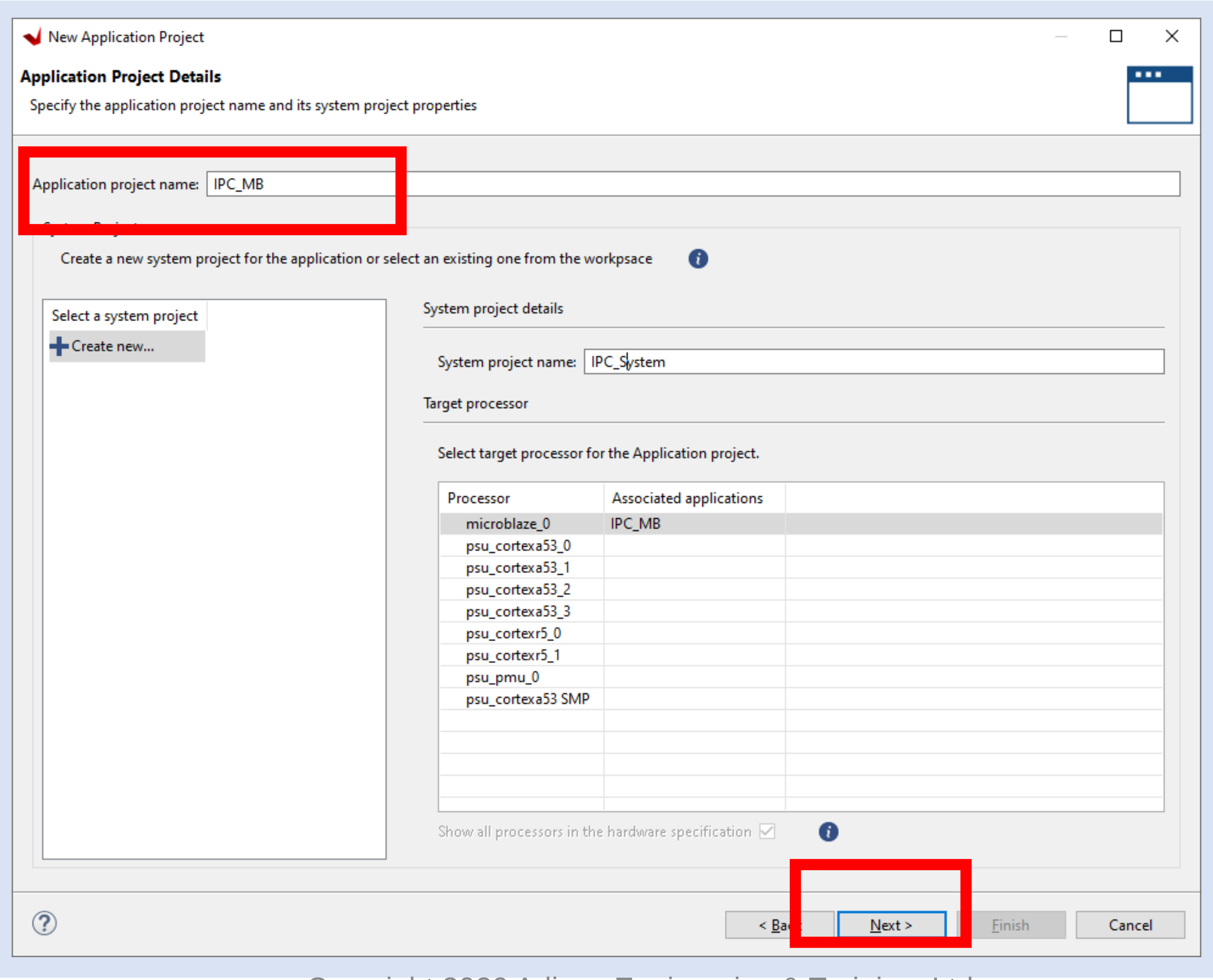
☑ Generate boot components

Target processor to create FSBL: ● psu_cortexa53_0 ○ psu_cortexr5_0

Platform name: design_1_wrapper

Lab 2: Understanding Vitis Project creation & Flow

Step 30 – Enter a project name and click next – This is for the MicroBlaze



Application Project Details

Specify the application project name and its system project properties

Application project name: IPC_MB

Create a new system project for the application or select an existing one from the workpsace

Select a system project

Create new...

System project details

System project name: IPC_System

Target processor

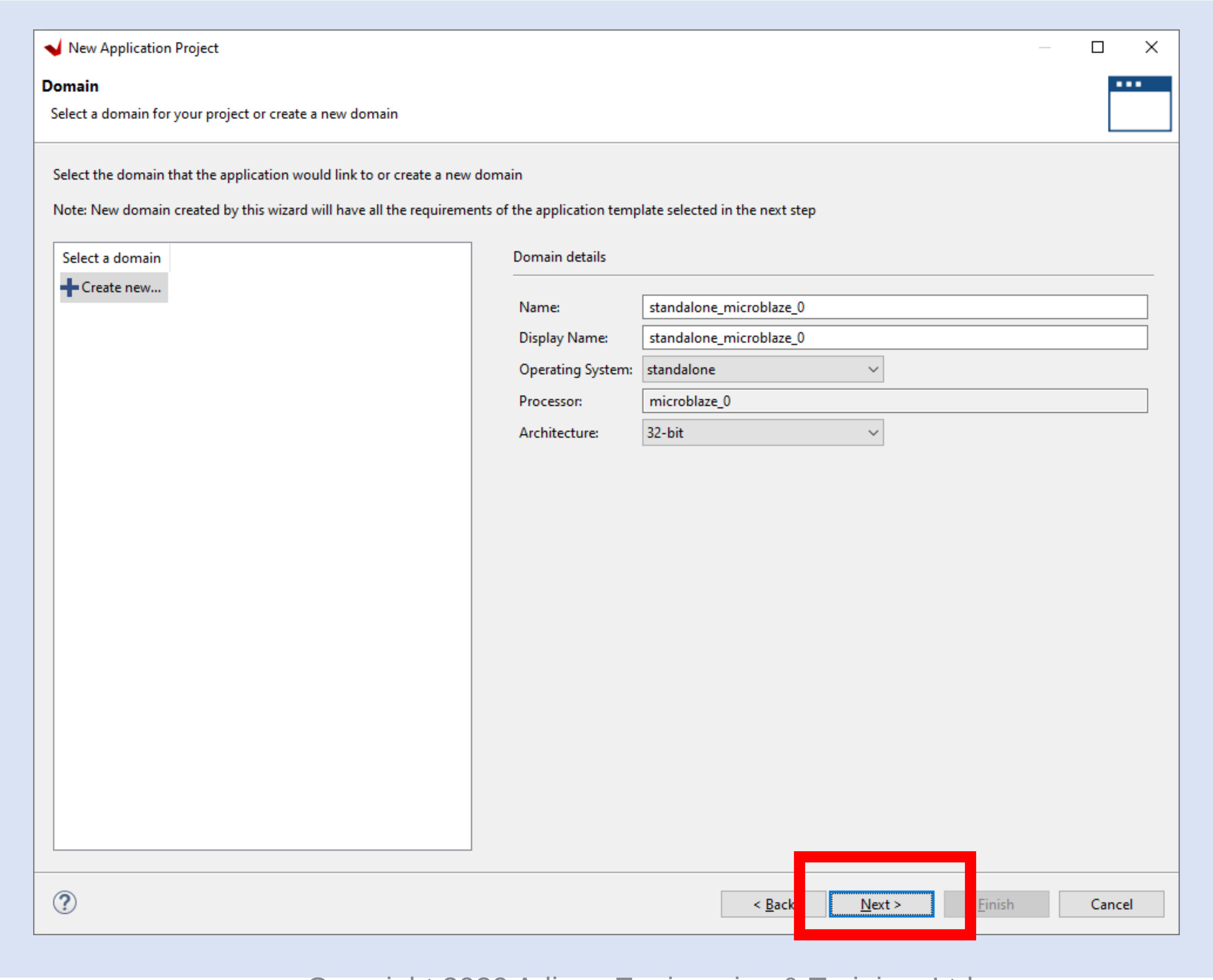
Select target processor for the Application project.

Processor	Associated applications
microblaze_0	IPC_MB
psu_cortexa53_0	
psu_cortexa53_1	
psu_cortexa53_2	
psu_cortexa53_3	
psu_cortexr5_0	
psu_cortexr5_1	
psu_pmu_0	
psu_cortexa53 SMP	

Show all processors in the hardware specification ☑

Lab 2: Understanding Vitis Project creation & Flow

Step 31 – Click on Next



Domain

Select a domain for your project or create a new domain

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

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

Select a domain

Domain details

Name: standalone_microblaze_0

Display Name: standalone_microblaze_0

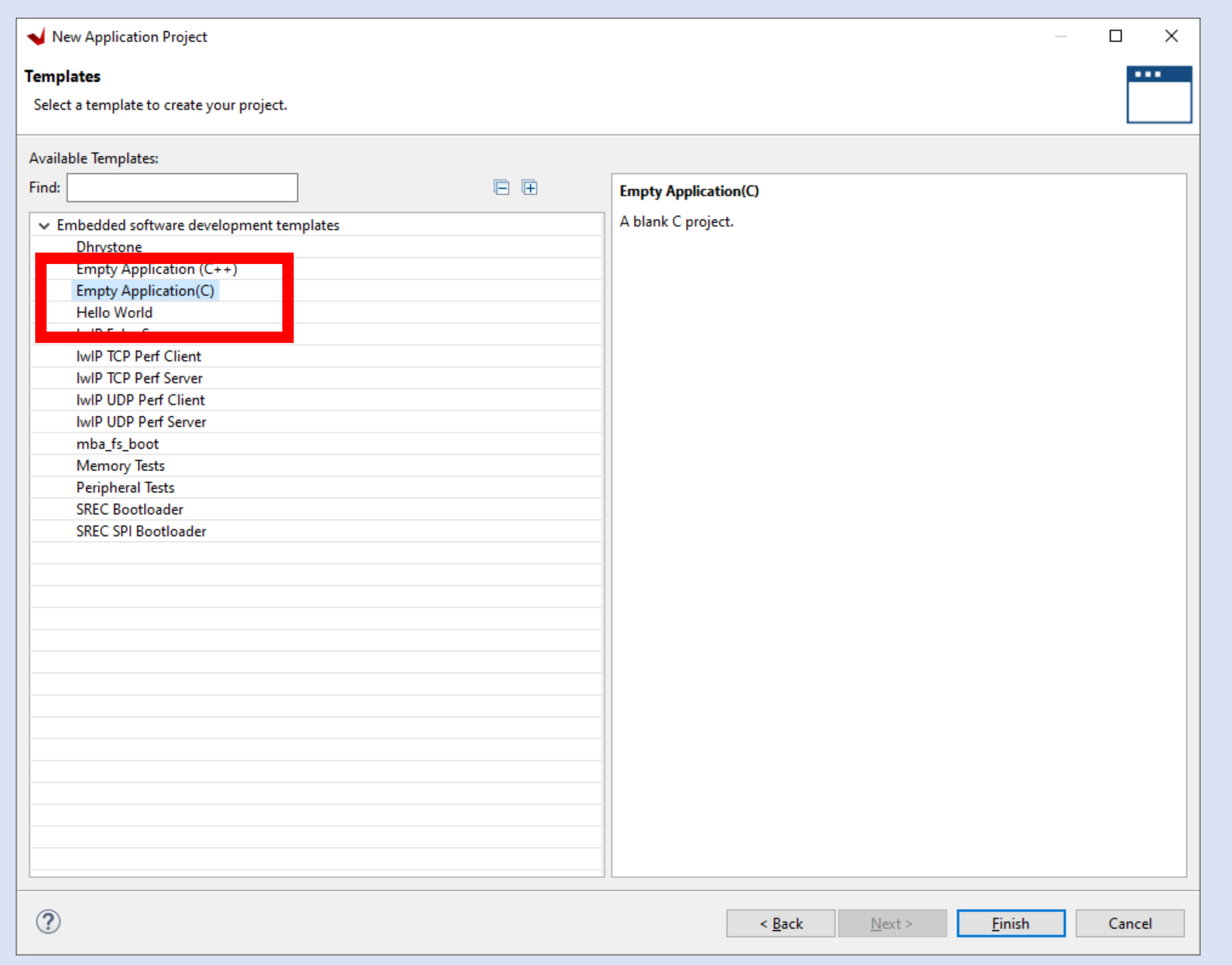
Operating System: standalone

Processor: microblaze_0

Architecture: 32-bit

Lab 2: Understanding Vitis Project creation & Flow

Step 32 – Select Empty Application



New Application Project

Templates

Select a template to create your project.

Available Templates:

Find:

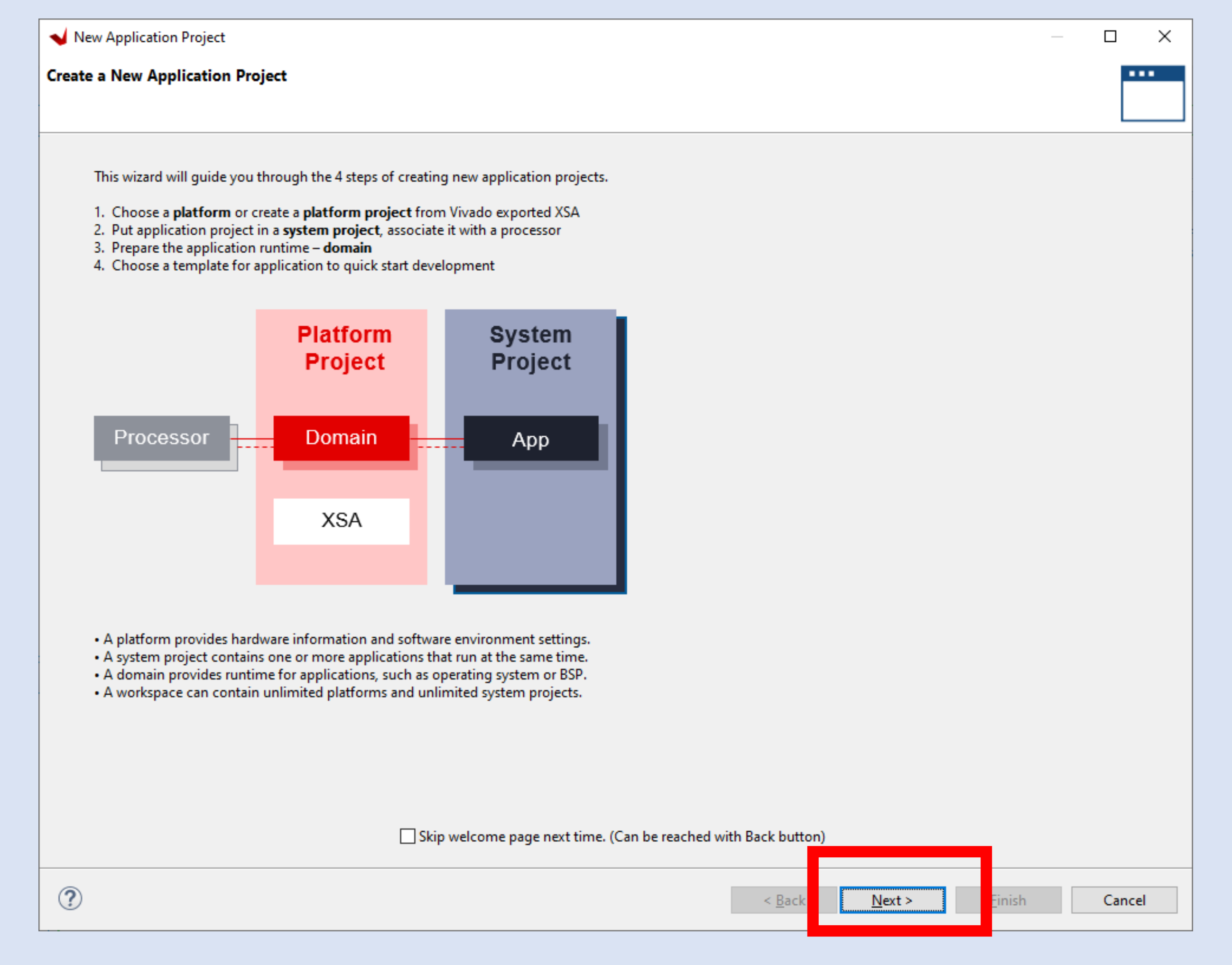
Embedded software development templates
- Dhrystone
- Empty Application (C++)
- Empty Application(C)
- Hello World
- lwIP TCP Perf Client
- lwIP TCP Perf Server
- lwIP UDP Perf Client
- lwIP UDP Perf Server
- mba_fs_boot
- Memory Tests
- Peripheral Tests
- SREC Bootloader
- SREC SPI Bootloader

Empty Application(C)

A blank C project.

Lab 2: Understanding Vitis Project creation & Flow

Step 33 – From File select new application project and click next



Create a New Application Project

This wizard will guide you through the 4 steps of creating new application projects.

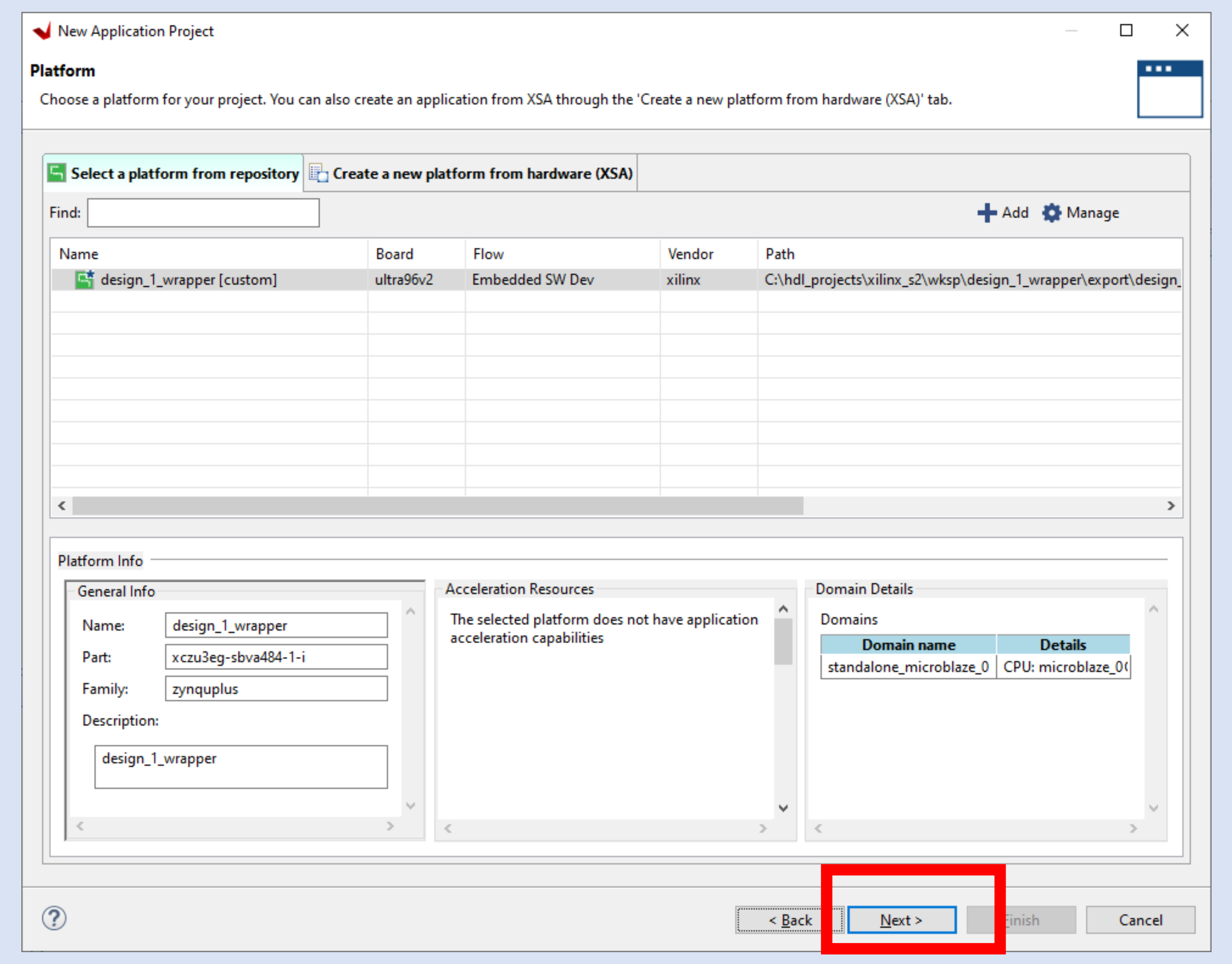
1. Choose a **platform** or create a **platform project** from Vivado exported XSA
2. Put application project in a **system project**, associate it with a processor
3. Prepare the application runtime – **domain**
4. Choose a template for application to quick start development

- A platform provides hardware information and software environment settings.
- A system project contains one or more applications that run at the same time.
- A domain provides runtime for applications, such as operating system or BSP.
- A workspace can contain unlimited platforms and unlimited system projects.

☐ Skip welcome page next time. (Can be reached with Back button)

Lab 2: Understanding Vitis Project creation & Flow

Step 34 – Select the current platform and click next



New Application Project

Platform

Choose a platform for your project. You can also create an application from XSA through the 'Create a new platform from hardware (XSA)' tab.

Select a platform from repository
Create a new platform from hardware (XSA)
Find:
Add
Manage

Name	Board	Flow	Vendor	Path
design_1_wrapper [custom]	ultra96v2	Embedded SW Dev	xilinx	C:\hdl_projects\xilinx_s2\wksp\design_1_wrapper\export\design_

Platform Info

General Info

Name: design_1_wrapper

Part: xczu3eg-sbva484-1-i

Family: zynquplus

Description: design_1_wrapper

Acceleration Resources

The selected platform does not have application acceleration capabilities

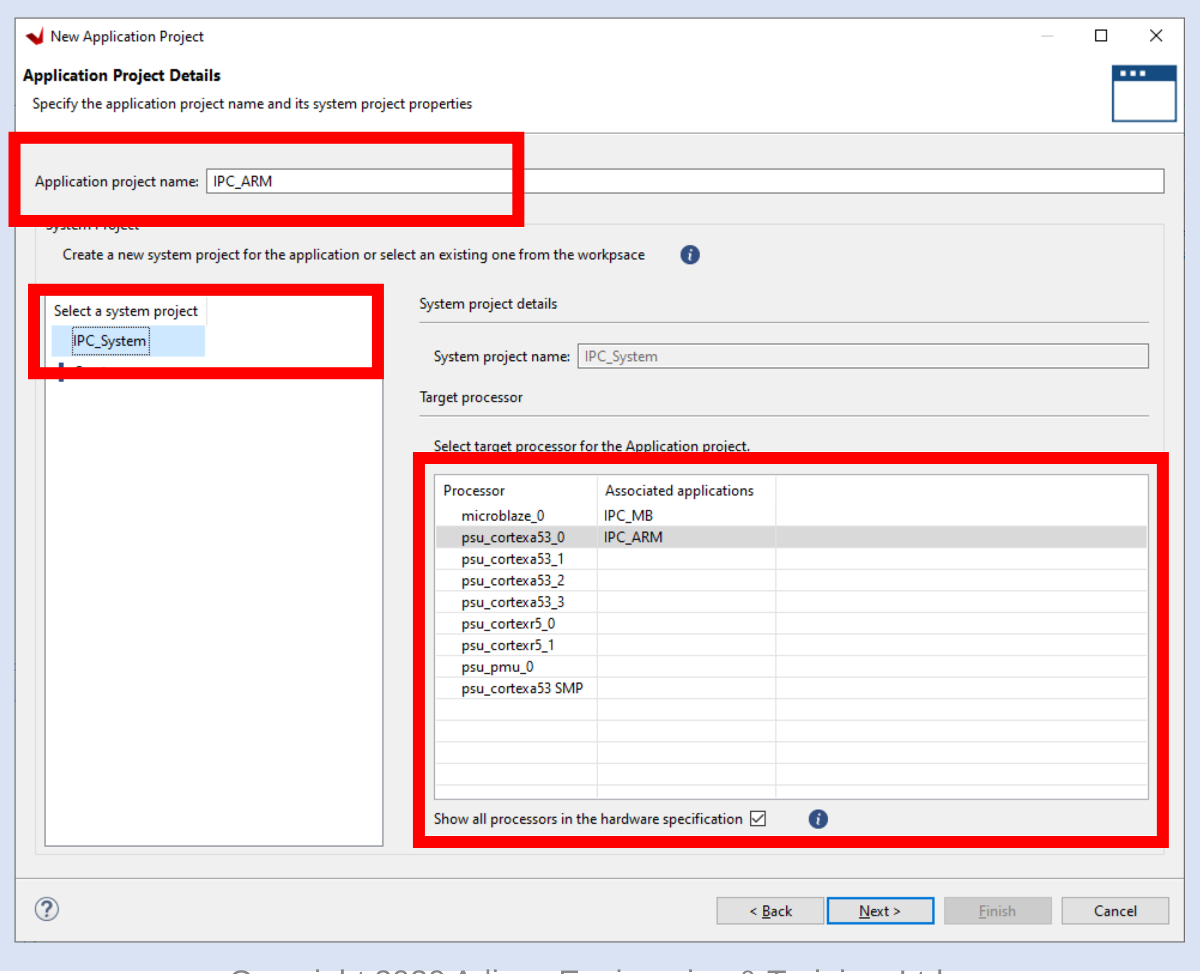
Domain Details

Domains

Domain name	Details
standalone_microblaze_0	CPU: microblaze_0(

Lab 2: Understanding Vitis Project creation & Flow

Step 35 – Select the IPC System, ensure show all processors is selected, select processor A53_0 and enter a name IPC_ARM



New Application Project

Application Project Details

Specify the application project name and its system project properties

Application project name: IPC_ARM

System Project

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

Select a system project

IPC_System

System project details

System project name: IPC_System

Target processor

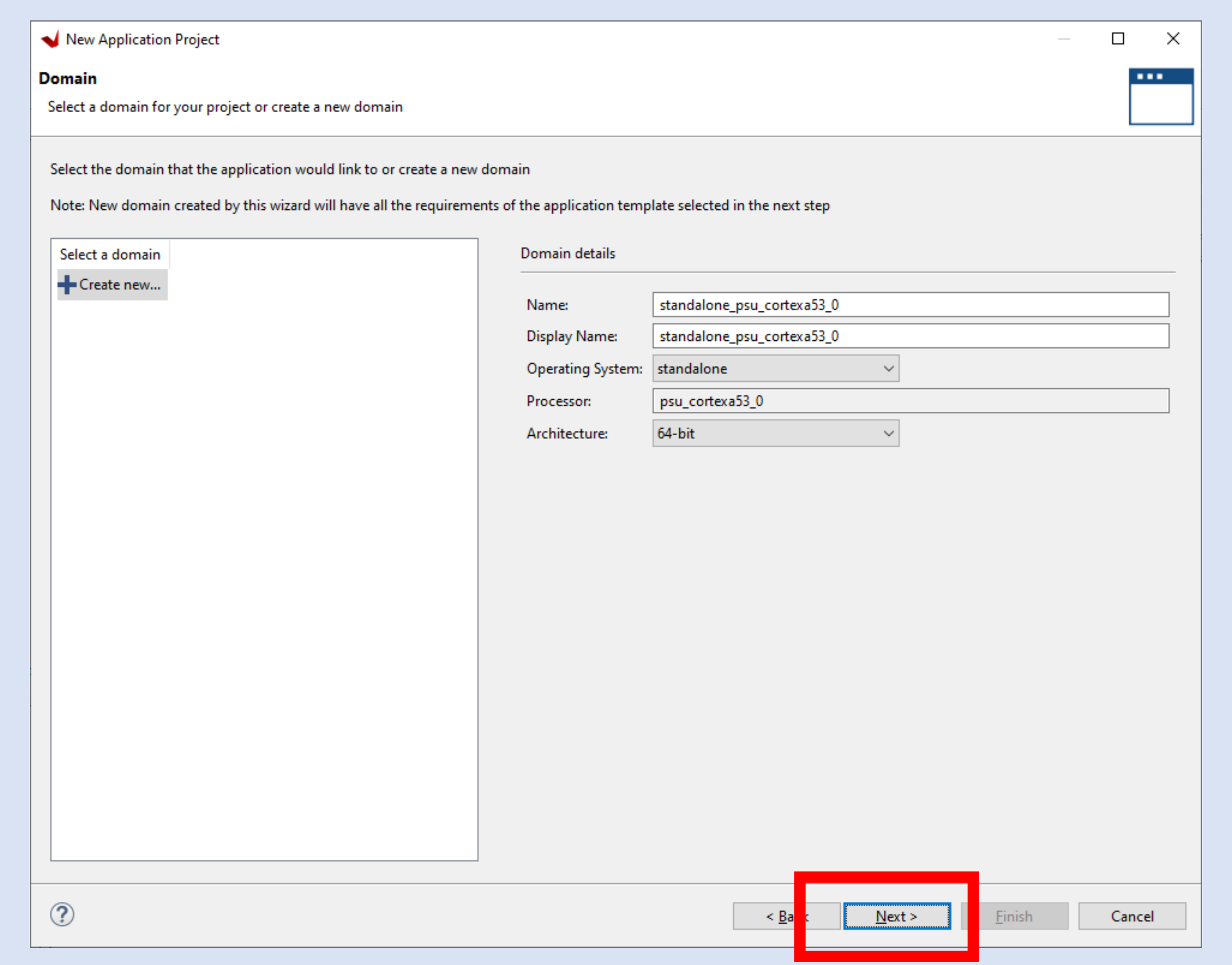
Select target processor for the Application project.

Processor	Associated applications
microblaze_0	IPC_MB
psu_cortexa53_0	IPC_ARM
psu_cortexa53_1	
psu_cortexa53_2	
psu_cortexa53_3	
psu_cortexr5_0	
psu_cortexr5_1	
psu_pmu_0	
psu_cortexa53 SMP	

Show all processors in the hardware specification ☑

Lab 2: Understanding Vitis Project creation & Flow

Step 36 – Click on Next



New Application Project

Domain

Select a domain for your project or create a new domain

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

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

Select a domain

Create new...

Domain details

Name: standalone_psu_cortexa53_0

Display Name: standalone_psu_cortexa53_0

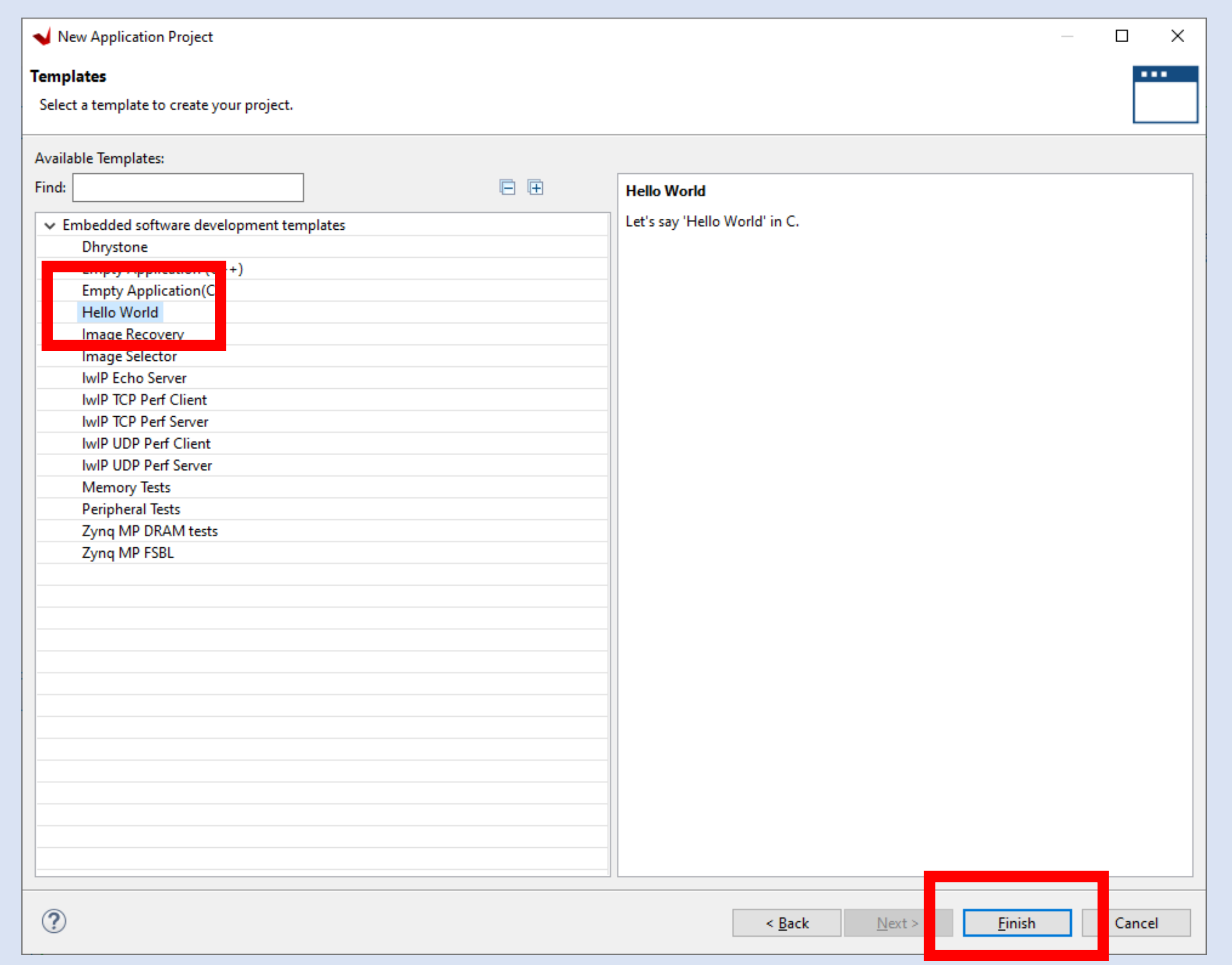
Operating System: standalone

Processor: psu_cortexa53_0

Architecture: 64-bit

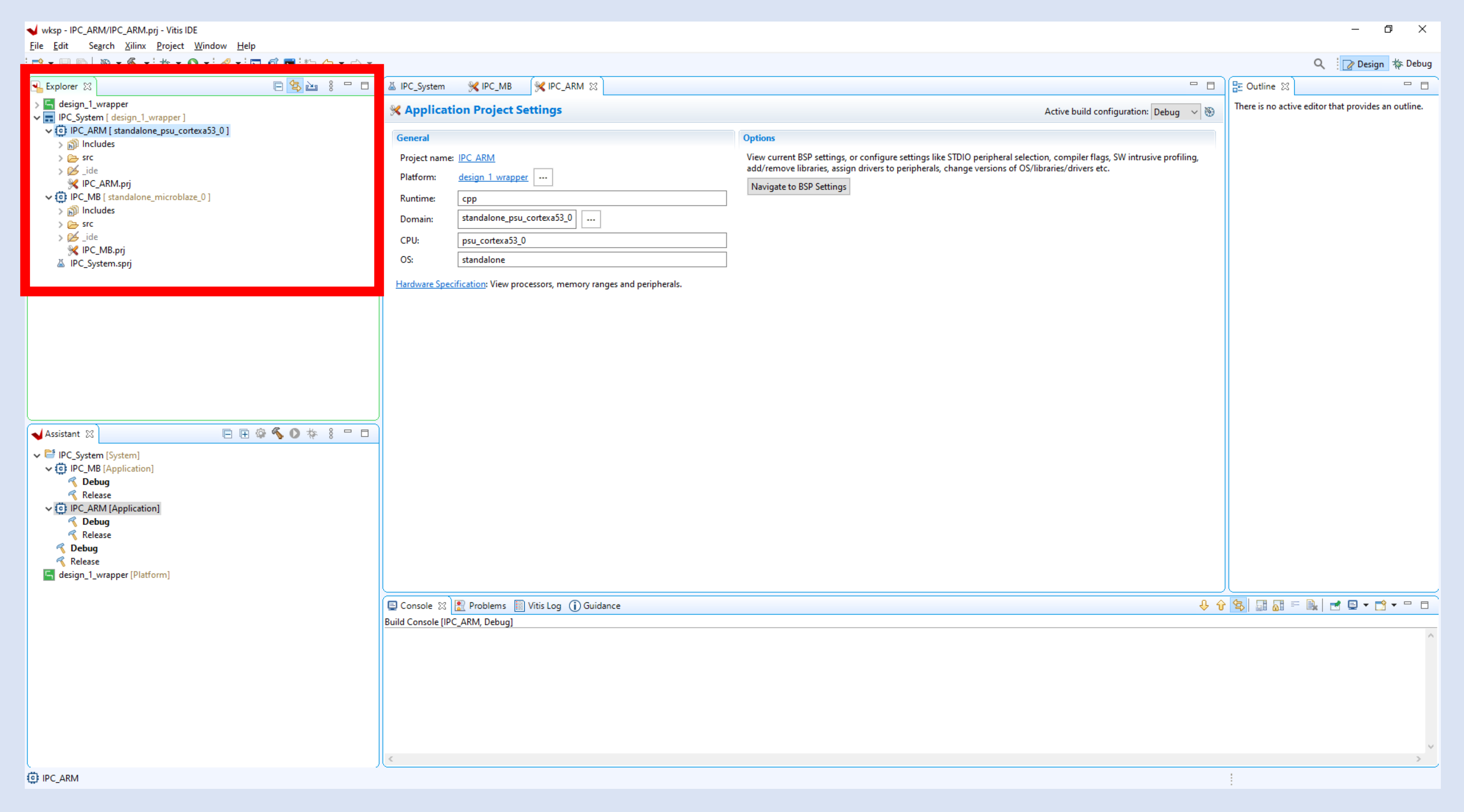
Lab 2: Understanding Vitis Project creation & Flow

Step 37 – Select the Hello World template and click finish



Lab 2: Understanding Vitis Project creation & Flow

Step 38 – You should now see two application projects along with the platform



wksp - IPC_ARM/IPC_ARM.prj - Vitis IDE
File Edit Search Xilinx Project Window Help
Design
Debug
Explorer
Assistant
Outline
Console
Problems
Vitis Log
Guidance

- design_1_wrapper
- IPC_System [design_1_wrapper]
  - IPC_ARM [standalone_psu_cortexa53_0]
    - Includes
    - src
    - _ide
    - IPC_ARM.prj
  - IPC_MB [standalone_microblaze_0]
    - Includes
    - src
    - _ide
    - IPC_MB.prj
  - IPC_System.sprj

IPC_System | IPC_MB | IPC_ARM

Application Project Settings

Active build configuration: Debug

General

Project name: IPC_ARM
Platform: design_1_wrapper
Runtime: cpp
Domain: standalone_psu_cortexa53_0
CPU: psu_cortexa53_0
OS: standalone

Hardware Specification: View processors, memory ranges and peripherals.

Options

View current BSP settings, or configure settings like STDIO peripheral selection, compiler flags, SW intrusive profiling, add/remove libraries, assign drivers to peripherals, change versions of OS/libraries/drivers etc.

Navigate to BSP Settings

There is no active editor that provides an outline.

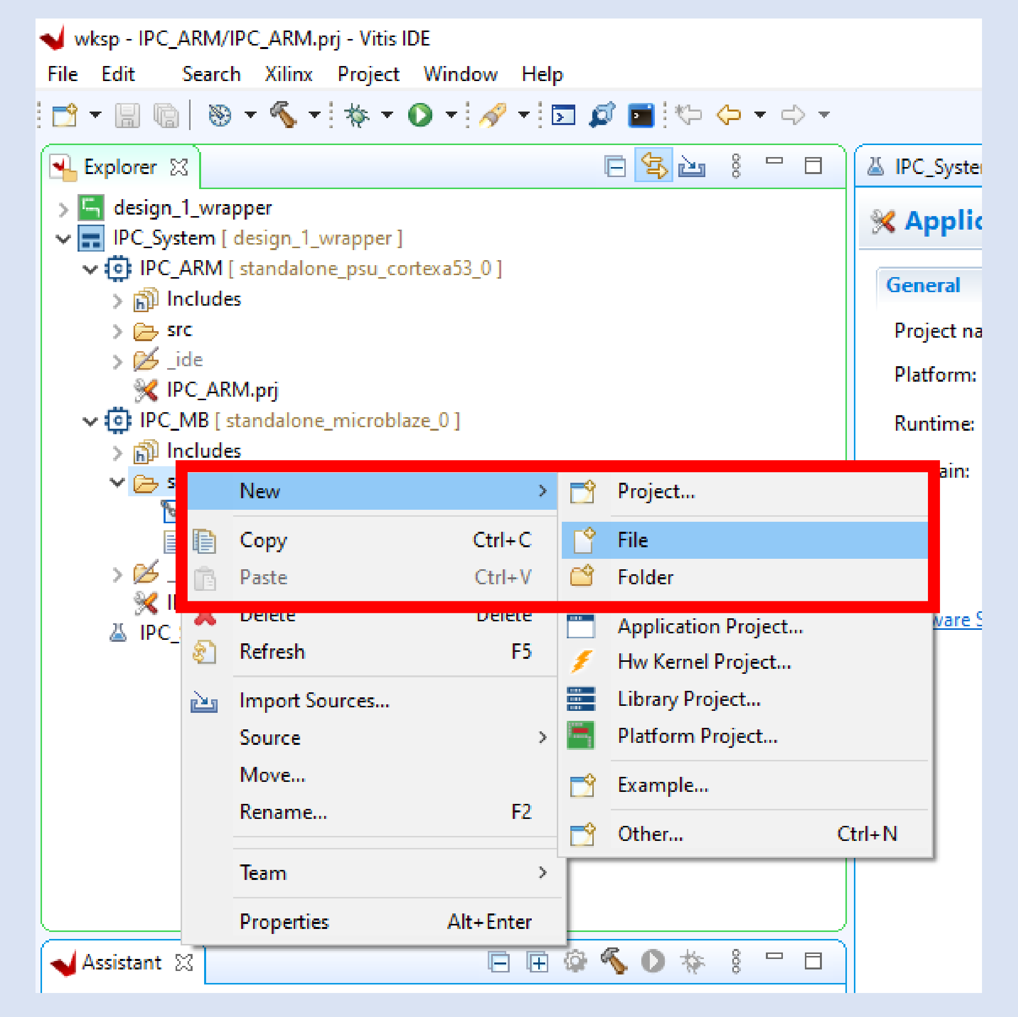
- IPC_System [System]
  - IPC_MB [Application]
    - Debug
    - Release
  - IPC_ARM [Application]
    - Debug
    - Release
  - Debug
  - Release
- design_1_wrapper [Platform]

Build Console [IPC_ARM, Debug]

IPC_ARM

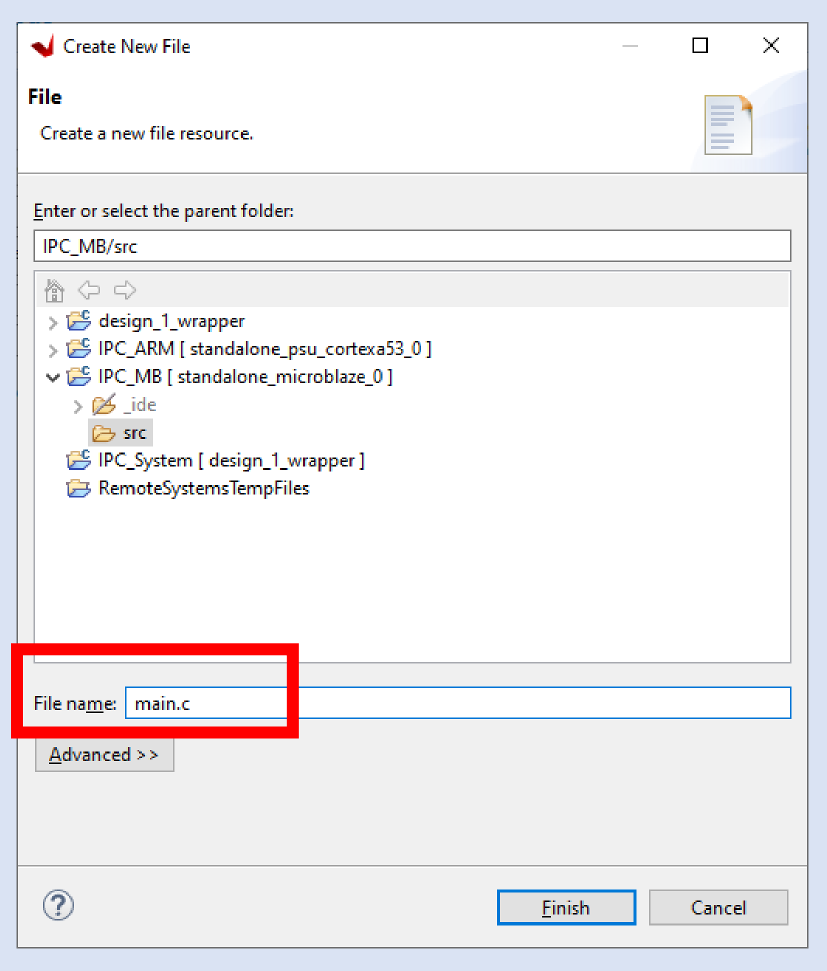
Lab 2: Understanding Vitis Project creation & Flow

Step 39 – Click on the MB project SRC directory, right click and select new File.



Lab 2: Understanding Vitis Project creation & Flow

Step 40 – Enter a name for the file



Create New File

File

Create a new file resource.

Enter or select the parent folder:

IPC_MB/src

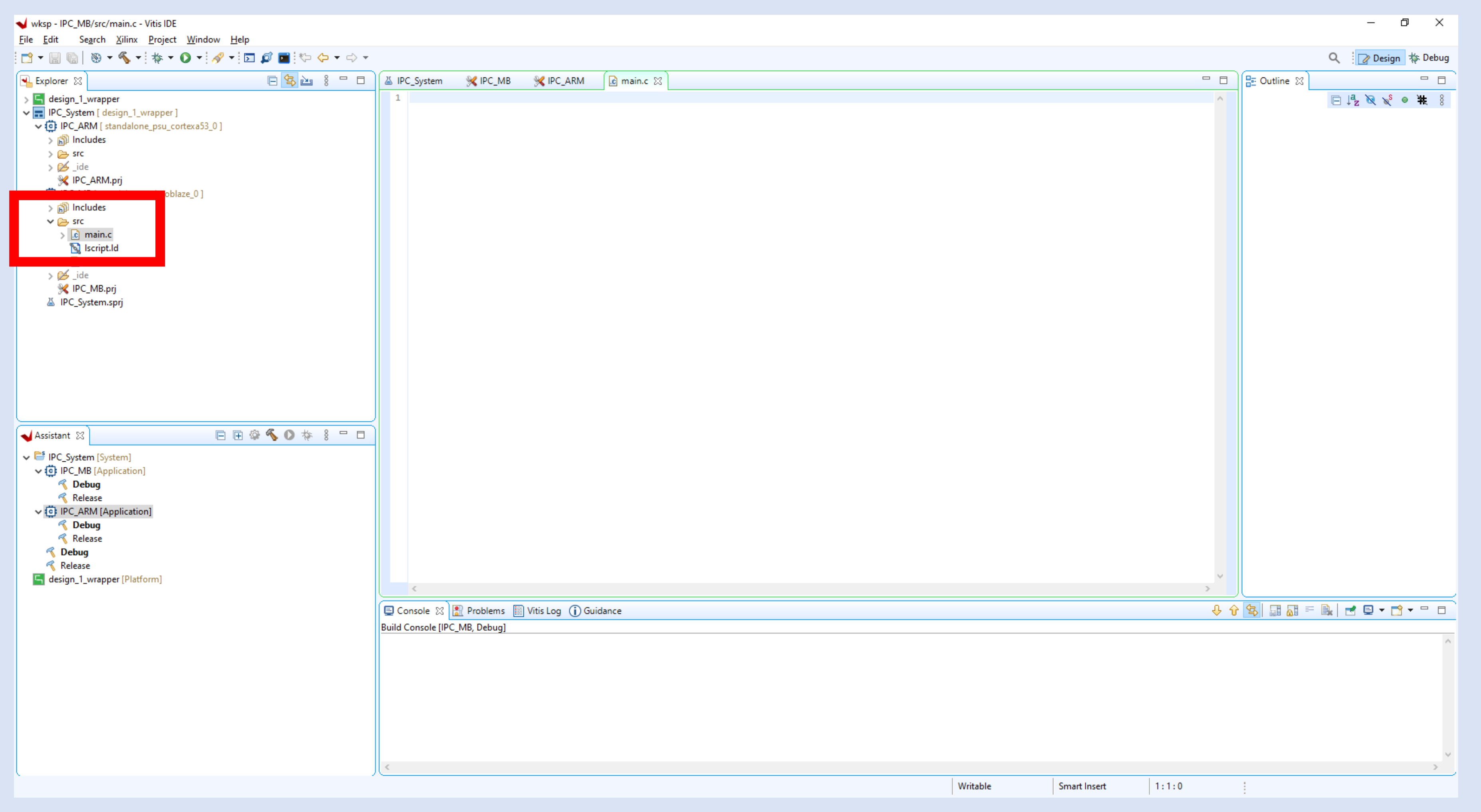
design_1_wrapper
IPC_ARM [standalone_psu_cortexa53_0]
IPC_MB [standalone_microblaze_0]
_ide
src
IPC_System [design_1_wrapper]
RemoteSystemsTempFiles

File name: main.c

Advanced >>
Finish
Cancel

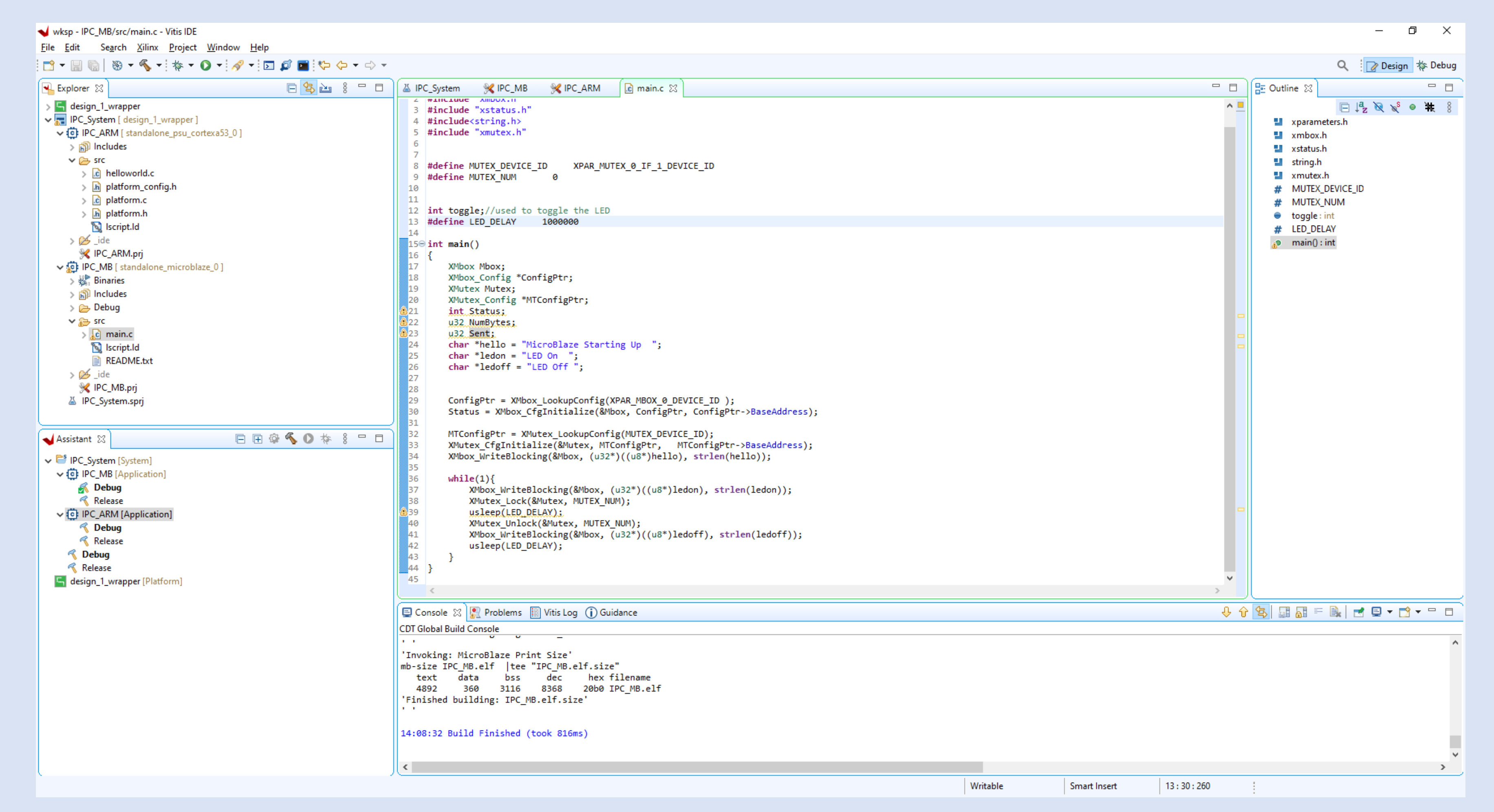
Lab 2: Understanding Vitis Project creation & Flow

Step 41 – Open the newly created file



Lab 2: Understanding Vitis Project creation & Flow

Step 42 – Copy the MB application, into the file and save the file.

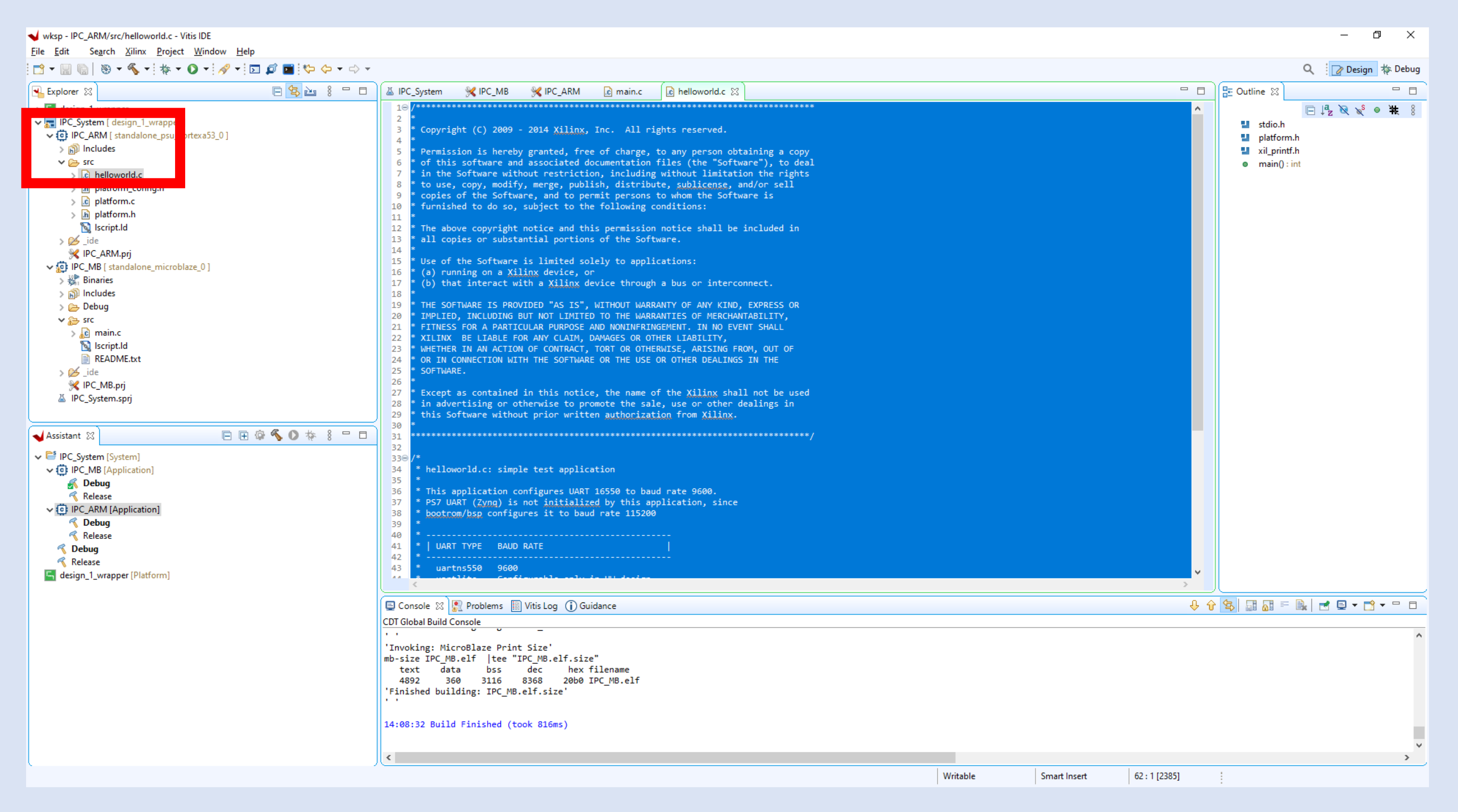


```c
#include "xmbox.h"
#include "xstatus.h"
#include<string.h>
#include "xmutex.h"


#define MUTEX_DEVICE_ID     XPAR_MUTEX_0_IF_1_DEVICE_ID
#define MUTEX_NUM       0


int toggle;//used to toggle the LED
#define LED_DELAY     1000000

int main()
{
    XMbox Mbox;
    XMbox_Config *ConfigPtr;
    XMutex Mutex;
    XMutex_Config *MTConfigPtr;
    int Status;
    u32 NumBytes;
    u32 Sent;
    char *hello = "MicroBlaze Starting Up  ";
    char *ledon = "LED On  ";
    char *ledoff = "LED Off ";


    ConfigPtr = XMbox_LookupConfig(XPAR_MBOX_0_DEVICE_ID );
    Status = XMbox_CfgInitialize(&Mbox, ConfigPtr, ConfigPtr->BaseAddress);

    MTConfigPtr = XMutex_LookupConfig(MUTEX_DEVICE_ID);
    XMutex_CfgInitialize(&Mutex, MTConfigPtr,   MTConfigPtr->BaseAddress);
    XMbox_WriteBlocking(&Mbox, (u32*)((u8*)hello), strlen(hello));

    while(1){
        XMbox_WriteBlocking(&Mbox, (u32*)((u8*)ledon), strlen(ledon));
        XMutex_Lock(&Mutex, MUTEX_NUM);
        usleep(LED_DELAY);
        XMutex_Unlock(&Mutex, MUTEX_NUM);
        XMbox_WriteBlocking(&Mbox, (u32*)((u8*)ledoff), strlen(ledoff));
        usleep(LED_DELAY);
    }
}
```

```
' '
'Invoking: MicroBlaze Print Size'
mb-size IPC_MB.elf  |tee "IPC_MB.elf.size"
   text    data     bss     dec     hex filename
   4892     360    3116    8368    20b0 IPC_MB.elf
'Finished building: IPC_MB.elf.size'
' '


14:08:32 Build Finished (took 816ms)
```

Lab 2: Understanding Vitis Project creation & Flow

Step 43 – Open the IPC_ARM hello world file and select and delete the entire contents

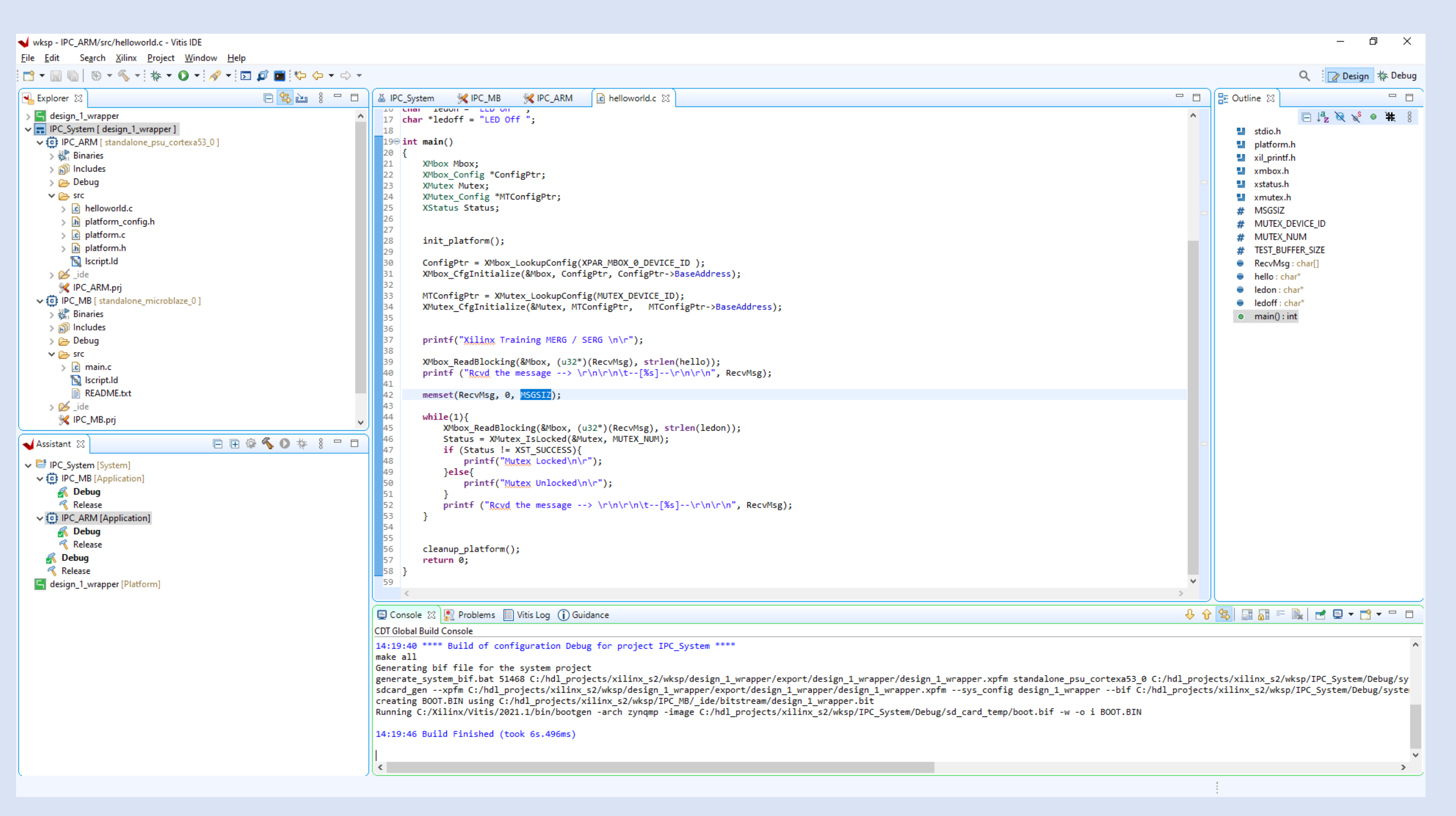


```
/******************************************************************************
*
* Copyright (C) 2009 - 2014 Xilinx, Inc.  All rights reserved.
*
* Permission is hereby granted, free of charge, to any person obtaining a copy
* of this software and associated documentation files (the "Software"), to deal
* in the Software without restriction, including without limitation the rights
* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
* copies of the Software, and to permit persons to whom the Software is
* furnished to do so, subject to the following conditions:
*
* The above copyright notice and this permission notice shall be included in
* all copies or substantial portions of the Software.
*
* Use of the Software is limited solely to applications:
* (a) running on a Xilinx device, or
* (b) that interact with a Xilinx device through a bus or interconnect.
*
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
* XILINX  BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
* WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF
* OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
* SOFTWARE.
*
* Except as contained in this notice, the name of the Xilinx shall not be used
* in advertising or otherwise to promote the sale, use or other dealings in
* this Software without prior written authorization from Xilinx.
*
******************************************************************************/

/*
 * helloworld.c: simple test application
 *
 * This application configures UART 16550 to baud rate 9600.
 * PS7 UART (Zynq) is not initialized by this application, since
 * bootrom/bsp configures it to baud rate 115200
 *
 * ------------------------------------------------
 * | UART TYPE   BAUD RATE                        |
 * ------------------------------------------------
 *   uartns550   9600
```

```
'Invoking: MicroBlaze Print Size'
mb-size IPC_MB.elf  |tee "IPC_MB.elf.size"
   text    data     bss     dec     hex filename
   4892     360    3116    8368    20b0 IPC_MB.elf
'Finished building: IPC_MB.elf.size'
' '


14:08:32 Build Finished (took 816ms)
```

Lab 2: Understanding Vitis Project creation & Flow

Step 44 – Copy across the Zynq Application code.

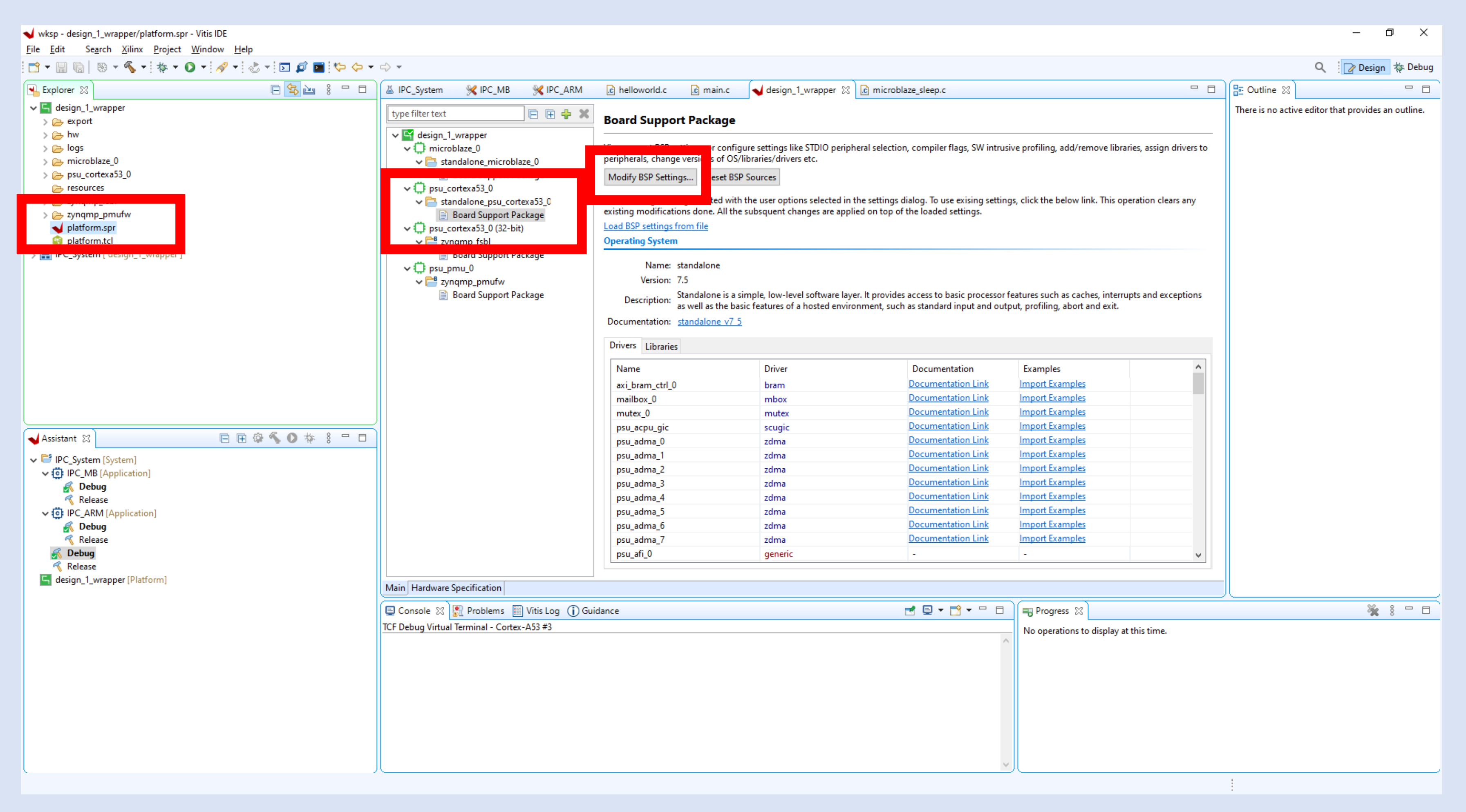


```c
char *ledon = "LED On ";
char *ledoff = "LED Off ";

int main()
{
    XMbox Mbox;
    XMbox_Config *ConfigPtr;
    XMutex Mutex;
    XMutex_Config *MTConfigPtr;
    XStatus Status;


    init_platform();

    ConfigPtr = XMbox_LookupConfig(XPAR_MBOX_0_DEVICE_ID );
    XMbox_CfgInitialize(&Mbox, ConfigPtr, ConfigPtr->BaseAddress);

    MTConfigPtr = XMutex_LookupConfig(MUTEX_DEVICE_ID);
    XMutex_CfgInitialize(&Mutex, MTConfigPtr,   MTConfigPtr->BaseAddress);


    printf("Xilinx Training MERG / SERG \n\r");

    XMbox_ReadBlocking(&Mbox, (u32*)(RecvMsg), strlen(hello));
    printf ("Rcvd the message --> \r\n\r\n\t--[%s]--\r\n\r\n", RecvMsg);

    memset(RecvMsg, 0, MSGSIZ);

    while(1){
        XMbox_ReadBlocking(&Mbox, (u32*)(RecvMsg), strlen(ledon));
        Status = XMutex_IsLocked(&Mutex, MUTEX_NUM);
        if (Status != XST_SUCCESS){
            printf("Mutex Locked\n\r");
        }else{
            printf("Mutex Unlocked\n\r");
        }
        printf ("Rcvd the message --> \r\n\r\n\t--[%s]--\r\n\r\n", RecvMsg);
    }


    cleanup_platform();
    return 0;
}
```

```
14:19:40 **** Build of configuration Debug for project IPC_System ****
make all
Generating bif file for the system project
generate_system_bif.bat 51468 C:/hdl_projects/xilinx_s2/wksp/design_1_wrapper/export/design_1_wrapper/design_1_wrapper.xpfm standalone_psu_cortexa53_0 C:/hdl_projects/xilinx_s2/wksp/IPC_System/Debug/sy
sdcard_gen --xpfm C:/hdl_projects/xilinx_s2/wksp/design_1_wrapper/export/design_1_wrapper/design_1_wrapper.xpfm --sys_config design_1_wrapper --bif C:/hdl_projects/xilinx_s2/wksp/IPC_System/Debug/syste
creating BOOT.BIN using C:/hdl_projects/xilinx_s2/wksp/IPC_MB/_ide/bitstream/design_1_wrapper.bit
Running C:/Xilinx/Vitis/2021.1/bin/bootgen -arch zynqmp -image C:/hdl_projects/xilinx_s2/wksp/IPC_System/Debug/sd_card_temp/boot.bif -w -o i BOOT.BIN

14:19:46 Build Finished (took 6s.496ms)
```

Lab 2: Understanding Vitis Project creation & Flow

Step 45 – Open the Platform.spr file, select BSP for the A53 click Modify BSP Settings



Board Support Package

r configure settings like STDIO peripheral selection, compiler flags, SW intrusive profiling, add/remove libraries, assign drivers to peripherals, change versi s of OS/libraries/drivers etc.

Modify BSP Settings... eset BSP Sources

ted with the user options selected in the settings dialog. To use exising settings, click the below link. This operation clears any existing modifications done. All the subsquent changes are applied on top of the loaded settings.

Load BSP settings from file

Operating System

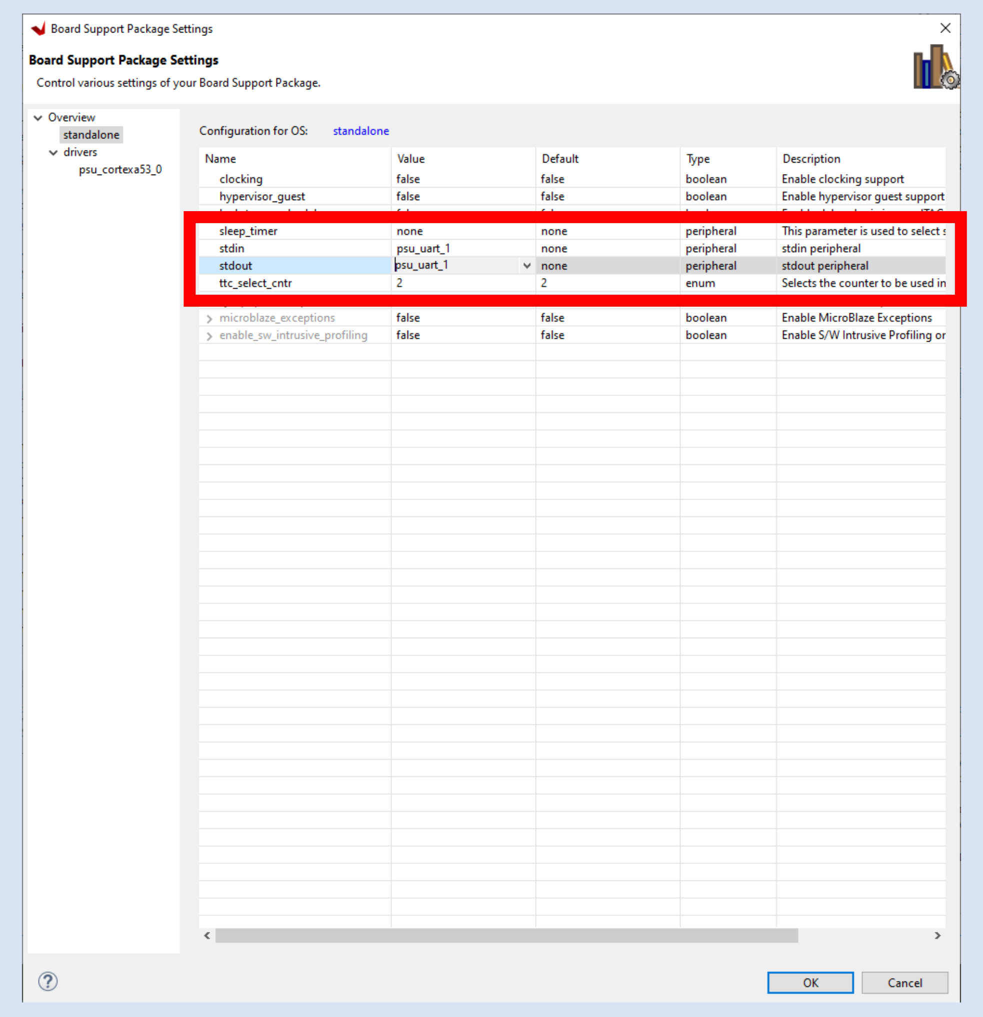
Name: standalone
Version: 7.5
Description: Standalone is a simple, low-level software layer. It provides access to basic processor features such as caches, interrupts and exceptions as well as the basic features of a hosted environment, such as standard input and output, profiling, abort and exit.
Documentation: standalone_v7_5

Drivers | Libraries

Name	Driver	Documentation	Examples
axi_bram_ctrl_0	bram	Documentation Link	Import Examples
mailbox_0	mbox	Documentation Link	Import Examples
mutex_0	mutex	Documentation Link	Import Examples
psu_acpu_gic	scugic	Documentation Link	Import Examples
psu_adma_0	zdma	Documentation Link	Import Examples
psu_adma_1	zdma	Documentation Link	Import Examples
psu_adma_2	zdma	Documentation Link	Import Examples
psu_adma_3	zdma	Documentation Link	Import Examples
psu_adma_4	zdma	Documentation Link	Import Examples
psu_adma_5	zdma	Documentation Link	Import Examples
psu_adma_6	zdma	Documentation Link	Import Examples
psu_adma_7	zdma	Documentation Link	Import Examples
psu_afi_0	generic	-	-

Lab 2: Understanding Vitis Project creation & Flow

Step 46 – On the Standalone tab change the stdin/stdout to PUS_UART 1



Board Support Package Settings

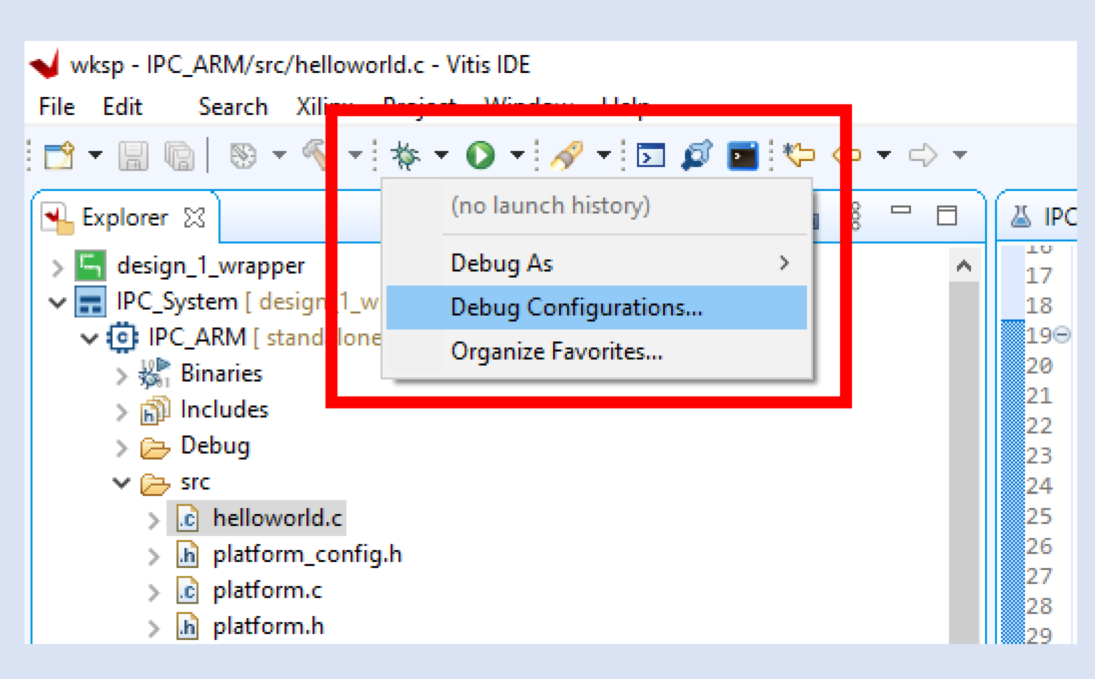
Control various settings of your Board Support Package.

Configuration for OS: standalone

Name	Value	Default	Type	Description
clocking	false	false	boolean	Enable clocking support
hypervisor_guest	false	false	boolean	Enable hypervisor guest support
sleep_timer	none	none	peripheral	This parameter is used to select s
stdin	psu_uart_1	none	peripheral	stdin peripheral
stdout	psu_uart_1	none	peripheral	stdout peripheral
ttc_select_cntr	2	2	enum	Selects the counter to be used in
microblaze_exceptions	false	false	boolean	Enable MicroBlaze Exceptions
enable_sw_intrusive_profiling	false	false	boolean	Enable S/W Intrusive Profiling or

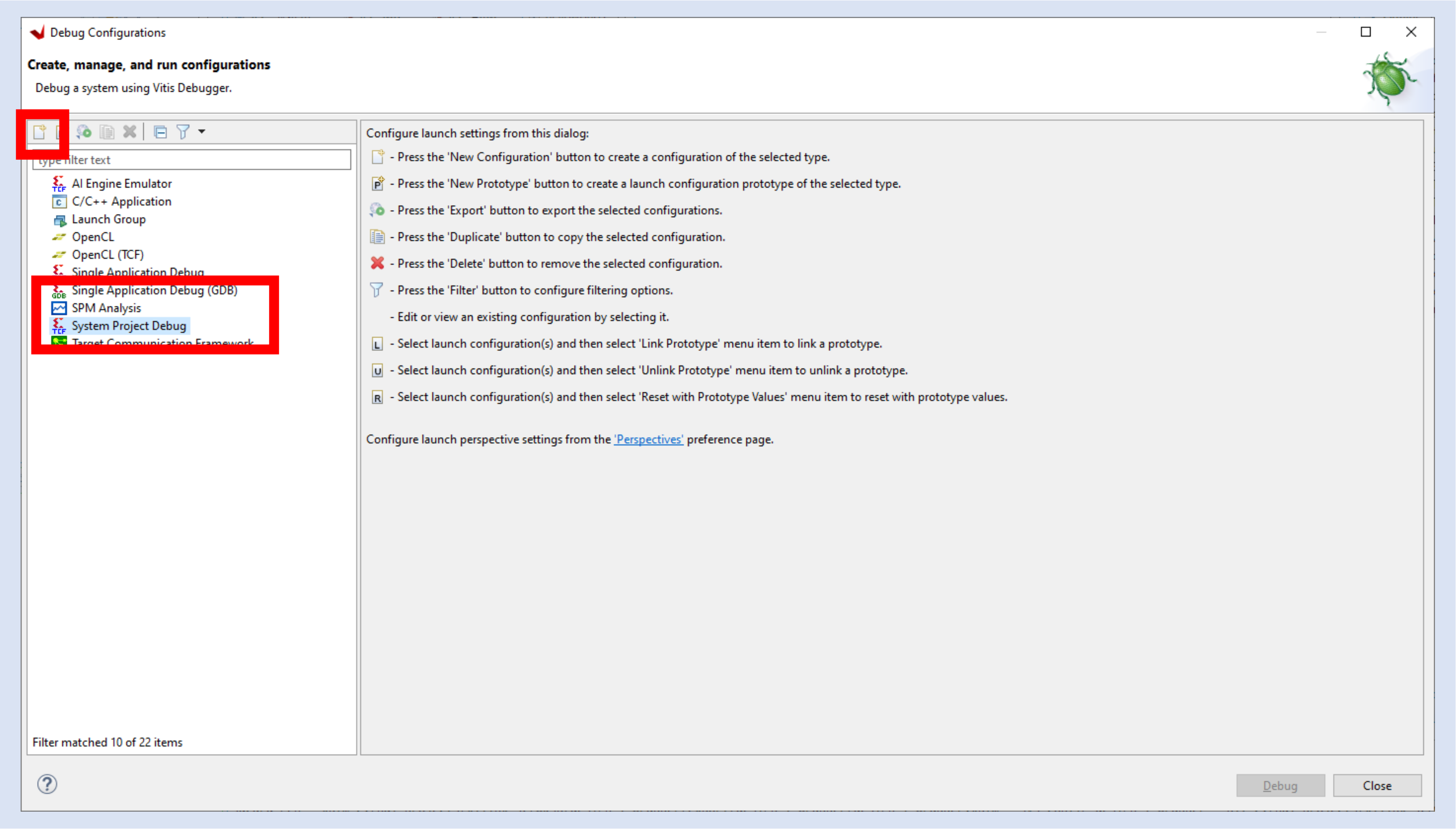
Lab 2: Understanding Vitis Project creation & Flow

Step 47 – From the debug icon, click on the arrow and select debug configuration



Lab 2: Understanding Vitis Project creation & Flow

Step 48 – Select System Project Debug and new



Create, manage, and run configurations

Debug a system using Vitis Debugger.

Filter matched 10 of 22 items

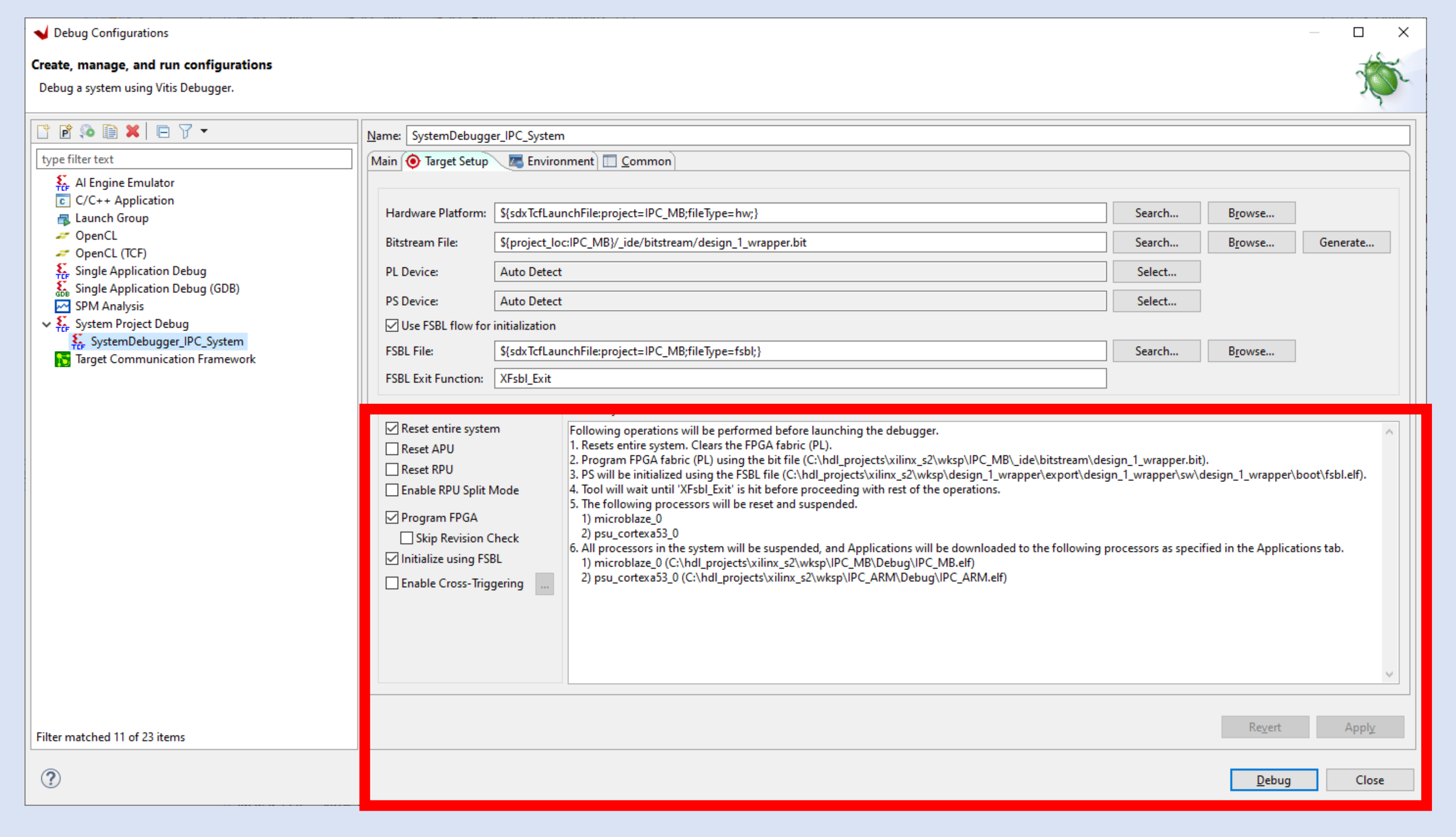
Configure launch settings from this dialog:

- Press the 'New Configuration' button to create a configuration of the selected type.
- Press the 'New Prototype' button to create a launch configuration prototype of the selected type.
- Press the 'Export' button to export the selected configurations.
- 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.
- Select launch configuration(s) and then select 'Link Prototype' menu item to link a prototype.
- Select launch configuration(s) and then select 'Unlink Prototype' menu item to unlink a prototype.
- Select launch configuration(s) and then select 'Reset with Prototype Values' menu item to reset with prototype values.

Configure launch perspective settings from the 'Perspectives' preference page.

Lab 2: Understanding Vitis Project creation & Flow

Step 49 – Click on Target Set up – This should be auto populated. Notice the PS is configured, followed by the PL before both applications are downloaded. Click Debug



Debug Configurations

Create, manage, and run configurations

Debug a system using Vitis Debugger.

type filter text

- AI Engine Emulator
- C/C++ Application
- Launch Group
- OpenCL
- OpenCL (TCF)
- Single Application Debug
- Single Application Debug (GDB)
- SPM Analysis
- System Project Debug
  - SystemDebugger_IPC_System
- Target Communication Framework

Filter matched 11 of 23 items

Name: SystemDebugger_IPC_System

Main | Target Setup | Environment | Common

Hardware Platform: ${sdxTcfLaunchFile:project=IPC_MB;fileType=hw;}

Bitstream File: ${project_loc:IPC_MB}/_ide/bitstream/design_1_wrapper.bit

PL Device: Auto Detect

PS Device: Auto Detect

☑ Use FSBL flow for initialization

FSBL File: ${sdxTcfLaunchFile:project=IPC_MB;fileType=fsbl;}

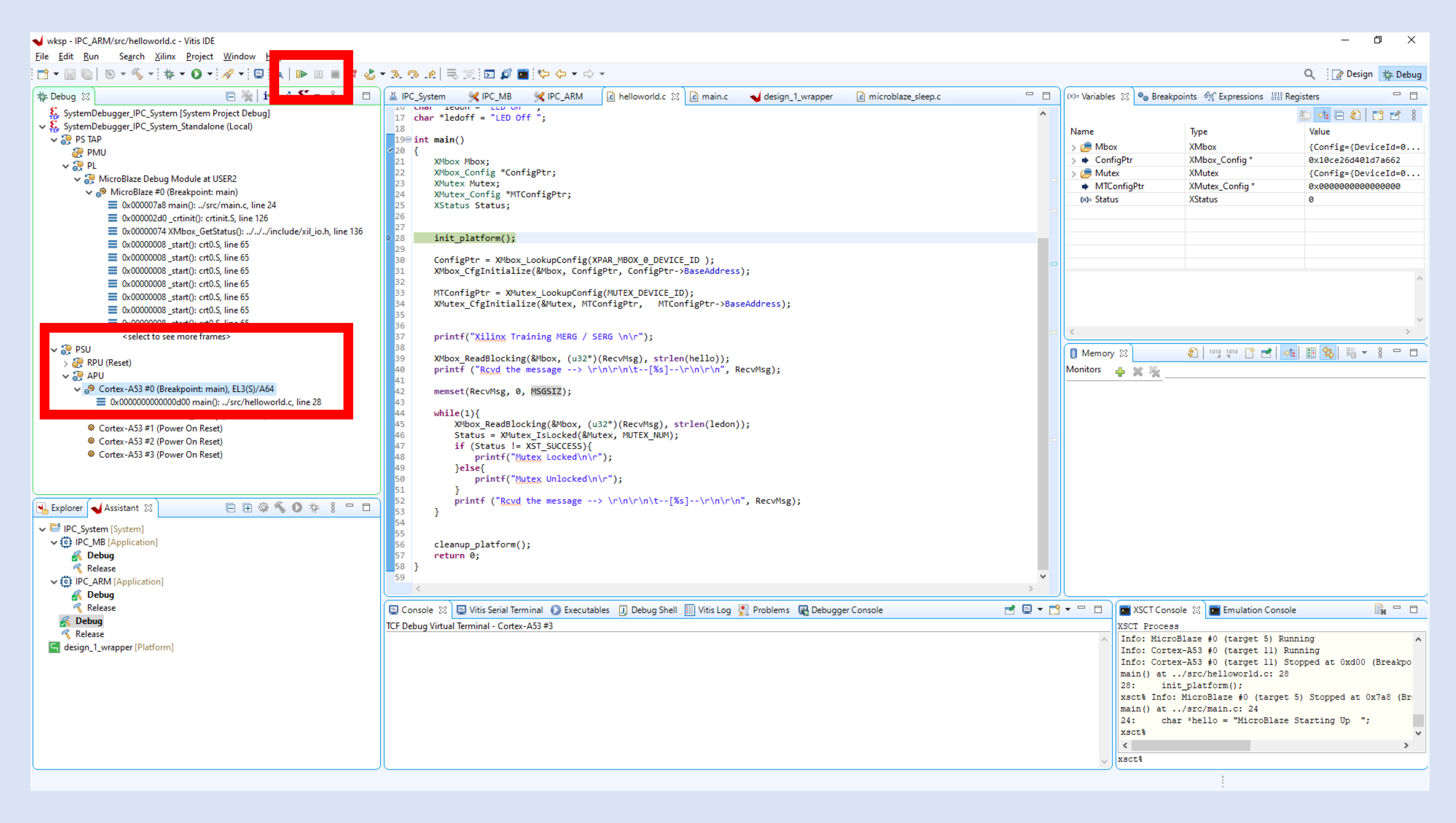
FSBL Exit Function: XFsbl_Exit

☑ Reset entire system
☐ Reset APU
☐ Reset RPU
☐ Enable RPU Split Mode
☑ Program FPGA
☐ Skip Revision Check
☑ Initialize using FSBL
☐ Enable Cross-Triggering

```
Following operations will be performed before launching the debugger.
1. Resets entire system. Clears the FPGA fabric (PL).
2. Program FPGA fabric (PL) using the bit file (C:\hdl_projects\xilinx_s2\wksp\IPC_MB\_ide\bitstream\design_1_wrapper.bit).
3. PS will be initialized using the FSBL file (C:\hdl_projects\xilinx_s2\wksp\design_1_wrapper\export\design_1_wrapper\sw\design_1_wrapper\boot\fsbl.elf).
4. Tool will wait until 'XFsbl_Exit' is hit before proceeding with rest of the operations.
5. The following processors will be reset and suspended.
   1) microblaze_0
   2) psu_cortexa53_0
6. All processors in the system will be suspended, and Applications will be downloaded to the following processors as specified in the Applications tab.
   1) microblaze_0 (C:\hdl_projects\xilinx_s2\wksp\IPC_MB\Debug\IPC_MB.elf)
   2) psu_cortexa53_0 (C:\hdl_projects\xilinx_s2\wksp\IPC_ARM\Debug\IPC_ARM.elf)
```

Lab 2: Understanding Vitis Project creation & Flow

Step 50 – Both applications will download and pause execution, click on the APU and hit run to start it



```
SystemDebugger_IPC_System [System Project Debug]
SystemDebugger_IPC_System_Standalone (Local)
  PS TAP
    PMU
    PL
      MicroBlaze Debug Module at USER2
        MicroBlaze #0 (Breakpoint: main)
          0x000007a8 main(): ../src/main.c, line 24
          0x000002d0 _crtinit(): crtinit.S, line 126
          0x00000074 XMbox_GetStatus(): ../../../include/xil_io.h, line 136
          0x00000008 _start(): crt0.S, line 65
          0x00000008 _start(): crt0.S, line 65
          0x00000008 _start(): crt0.S, line 65
          0x00000008 _start(): crt0.S, line 65
          0x00000008 _start(): crt0.S, line 65
          0x00000008 _start(): crt0.S, line 65
          <select to see more frames>
  PSU
    RPU (Reset)
    APU
      Cortex-A53 #0 (Breakpoint: main), EL3(S)/A64
        0x0000000000000d00 main(): ../src/helloworld.c, line 28
      Cortex-A53 #1 (Power On Reset)
      Cortex-A53 #2 (Power On Reset)
      Cortex-A53 #3 (Power On Reset)
```

```
IPC_System [System]
  IPC_MB [Application]
    Debug
    Release
  IPC_ARM [Application]
    Debug
    Release
  Debug
  Release
design_1_wrapper [Platform]
```

```c
char *ledoff = "LED Off ";

int main()
{
    XMbox Mbox;
    XMbox_Config *ConfigPtr;
    XMutex Mutex;
    XMutex_Config *MTConfigPtr;
    XStatus Status;


    init_platform();

    ConfigPtr = XMbox_LookupConfig(XPAR_MBOX_0_DEVICE_ID );
    XMbox_CfgInitialize(&Mbox, ConfigPtr, ConfigPtr->BaseAddress);

    MTConfigPtr = XMutex_LookupConfig(MUTEX_DEVICE_ID);
    XMutex_CfgInitialize(&Mutex, MTConfigPtr,   MTConfigPtr->BaseAddress);


    printf("Xilinx Training MERG / SERG \n\r");

    XMbox_ReadBlocking(&Mbox, (u32*)(RecvMsg), strlen(hello));
    printf ("Rcvd the message --> \r\n\r\n\t--[%s]--\r\n\r\n", RecvMsg);

    memset(RecvMsg, 0, MSGSIZ);

    while(1){
        XMbox_ReadBlocking(&Mbox, (u32*)(RecvMsg), strlen(ledon));
        Status = XMutex_IsLocked(&Mutex, MUTEX_NUM);
        if (Status != XST_SUCCESS){
            printf("Mutex Locked\n\r");
        }else{
            printf("Mutex Unlocked\n\r");
        }
        printf ("Rcvd the message --> \r\n\r\n\t--[%s]--\r\n\r\n", RecvMsg);
    }


    cleanup_platform();
    return 0;
}
```

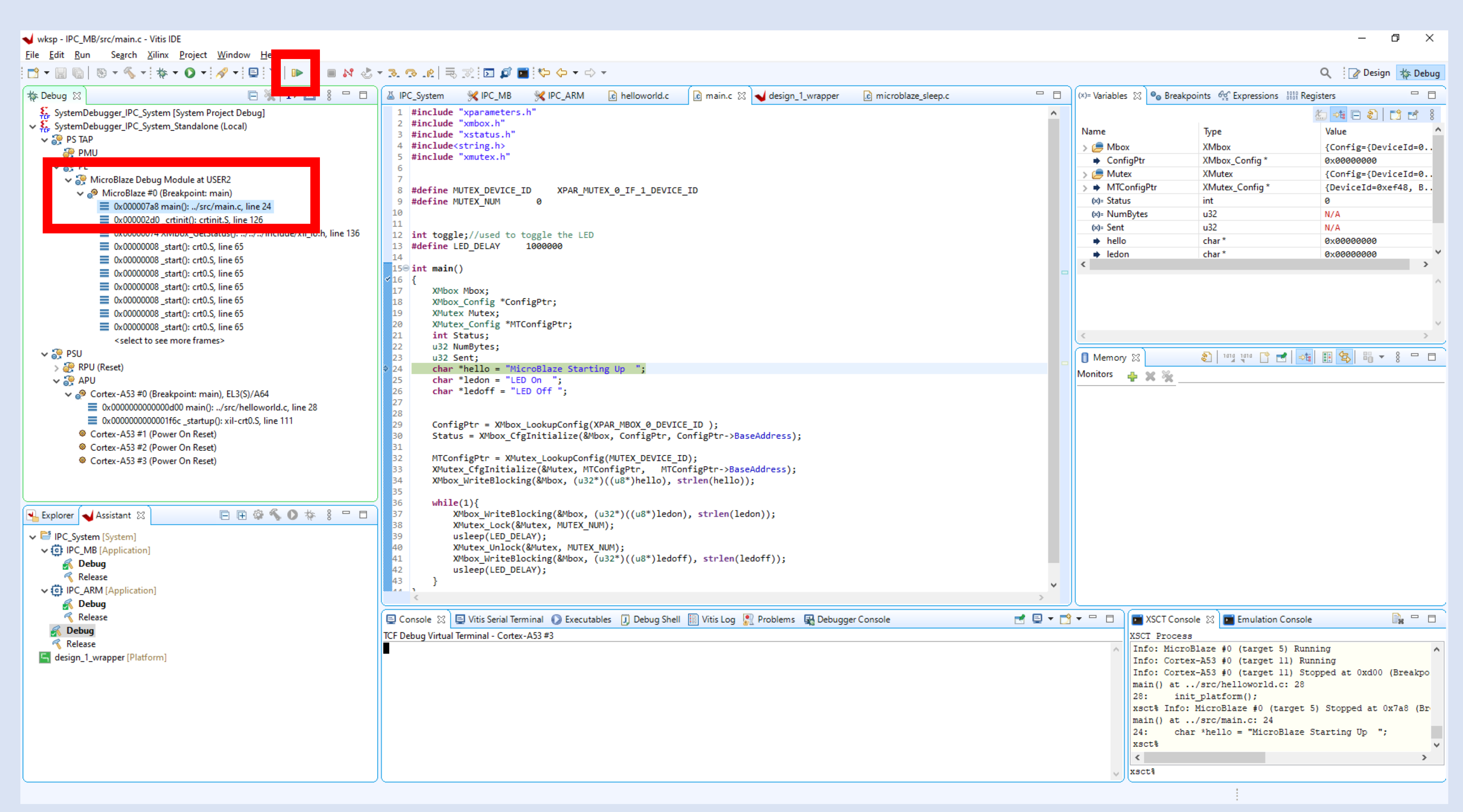
Name	Type	Value
Mbox	XMbox	{Config={DeviceId=0...
ConfigPtr	XMbox_Config *	0x10ce26d401d7a662
Mutex	XMutex	{Config={DeviceId=0...
MTConfigPtr	XMutex_Config *	0x0000000000000000
Status	XStatus	0

TCF Debug Virtual Terminal - Cortex-A53 #3

```
XSCT Process
Info: MicroBlaze #0 (target 5) Running
Info: Cortex-A53 #0 (target 11) Running
Info: Cortex-A53 #0 (target 11) Stopped at 0xd00 (Breakpo
main() at ../src/helloworld.c: 28
28:     init_platform();
xsct% Info: MicroBlaze #0 (target 5) Stopped at 0x7a8 (Br
main() at ../src/main.c: 24
24:     char *hello = "MicroBlaze Starting Up  ";
xsct%
xsct%
```

Lab 2: Understanding Vitis Project creation & Flow

Step 51 – Select the MB application and start that executing



```
SystemDebugger_IPC_System [System Project Debug]
SystemDebugger_IPC_System_Standalone (Local)
  PS TAP
    PMU
    PL
      MicroBlaze Debug Module at USER2
        MicroBlaze #0 (Breakpoint: main)
          0x000007a8 main(): ../src/main.c, line 24
          0x000002d0 _crtinit(): crtinit.S, line 126
          0x00000074 XMbox_GetStatus(): ../../../include/xil_io.h, line 136
          0x00000008 _start(): crt0.S, line 65
          0x00000008 _start(): crt0.S, line 65
          0x00000008 _start(): crt0.S, line 65
          0x00000008 _start(): crt0.S, line 65
          0x00000008 _start(): crt0.S, line 65
          0x00000008 _start(): crt0.S, line 65
          0x00000008 _start(): crt0.S, line 65
          <select to see more frames>
  PSU
    RPU (Reset)
    APU
      Cortex-A53 #0 (Breakpoint: main), EL3(S)/A64
        0x0000000000000d00 main(): ../src/helloworld.c, line 28
        0x0000000000001f6c _startup(): xil-crt0.S, line 111
      Cortex-A53 #1 (Power On Reset)
      Cortex-A53 #2 (Power On Reset)
      Cortex-A53 #3 (Power On Reset)
```

```
IPC_System [System]
  IPC_MB [Application]
    Debug
    Release
  IPC_ARM [Application]
    Debug
    Release
  Debug
  Release
design_1_wrapper [Platform]
```

```c
#include "xparameters.h"
#include "xmbox.h"
#include "xstatus.h"
#include<string.h>
#include "xmutex.h"


#define MUTEX_DEVICE_ID      XPAR_MUTEX_0_IF_1_DEVICE_ID
#define MUTEX_NUM        0


int toggle;//used to toggle the LED
#define LED_DELAY     1000000

int main()
{
    XMbox Mbox;
    XMbox_Config *ConfigPtr;
    XMutex Mutex;
    XMutex_Config *MTConfigPtr;
    int Status;
    u32 NumBytes;
    u32 Sent;
    char *hello = "MicroBlaze Starting Up  ";
    char *ledon = "LED On  ";
    char *ledoff = "LED Off ";


    ConfigPtr = XMbox_LookupConfig(XPAR_MBOX_0_DEVICE_ID );
    Status = XMbox_CfgInitialize(&Mbox, ConfigPtr, ConfigPtr->BaseAddress);

    MTConfigPtr = XMutex_LookupConfig(MUTEX_DEVICE_ID);
    XMutex_CfgInitialize(&Mutex, MTConfigPtr,   MTConfigPtr->BaseAddress);
    XMbox_WriteBlocking(&Mbox, (u32*)((u8*)hello), strlen(hello));

    while(1){
        XMbox_WriteBlocking(&Mbox, (u32*)((u8*)ledon), strlen(ledon));
        XMutex_Lock(&Mutex, MUTEX_NUM);
        usleep(LED_DELAY);
        XMutex_Unlock(&Mutex, MUTEX_NUM);
        XMbox_WriteBlocking(&Mbox, (u32*)((u8*)ledoff), strlen(ledoff));
        usleep(LED_DELAY);
    }
}
```

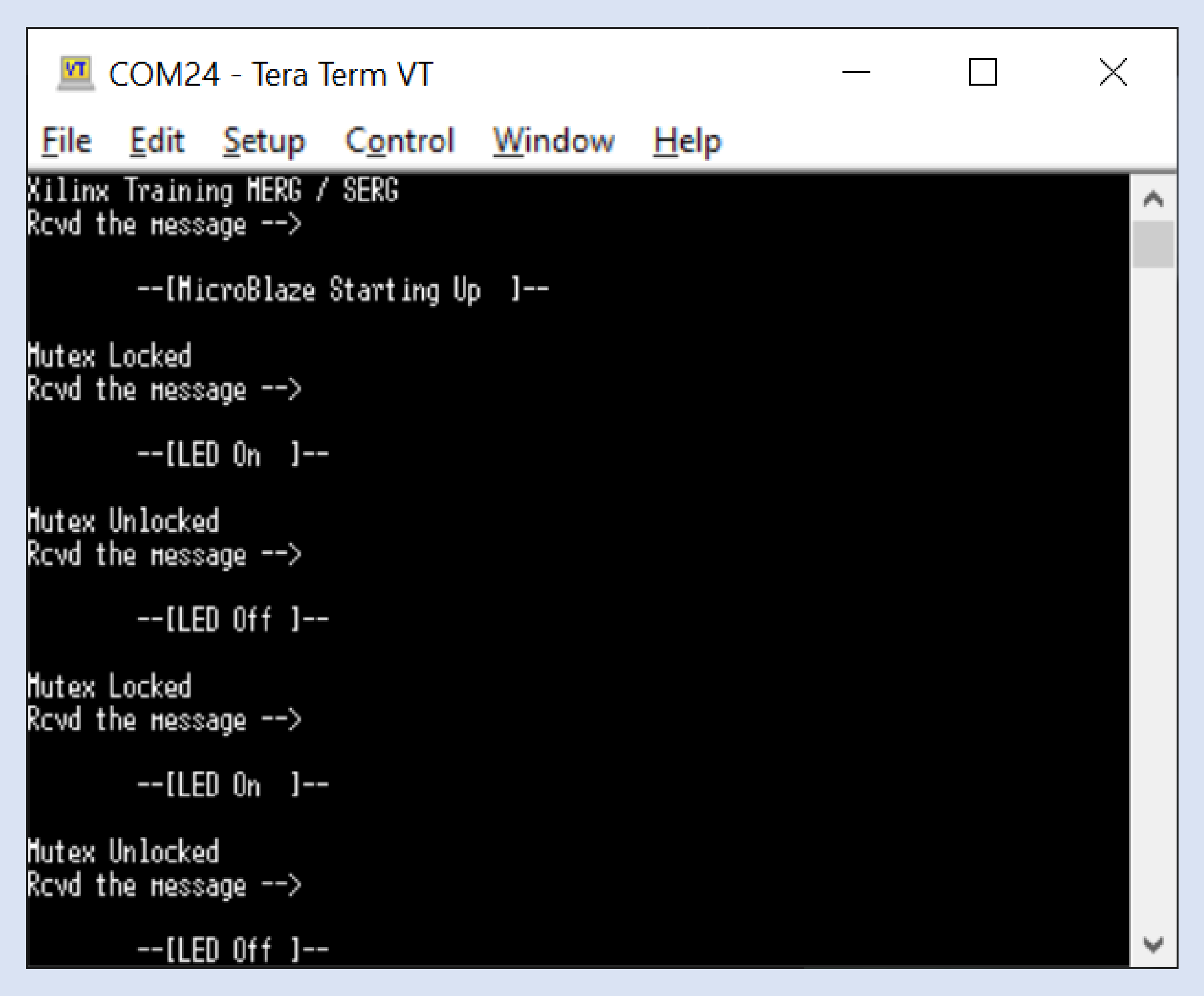
Name	Type	Value
Mbox	XMbox	{Config={DeviceId=0..
ConfigPtr	XMbox_Config *	0x00000000
Mutex	XMutex	{Config={DeviceId=0..
MTConfigPtr	XMutex_Config *	{DeviceId=0xef48, B..
Status	int	0
NumBytes	u32	N/A
Sent	u32	N/A
hello	char *	0x00000000
ledon	char *	0x00000000

```
TCF Debug Virtual Terminal - Cortex-A53 #3
```

```
XSCT Process
Info: MicroBlaze #0 (target 5) Running
Info: Cortex-A53 #0 (target 11) Running
Info: Cortex-A53 #0 (target 11) Stopped at 0xd00 (Breakpo
main() at ../src/helloworld.c: 28
28:     init_platform();
xsct% Info: MicroBlaze #0 (target 5) Stopped at 0x7a8 (Br
main() at ../src/main.c: 24
24:     char *hello = "MicroBlaze Starting Up  ";
xsct%
xsct%
```

Lab 2: Understanding Vitis Project creation & Flow

Step 52 – In a terminal window (114200, 1N8) you should see messages from the MB and ARM



```
Xilinx Training MERG / SERG
Rcvd the message -->

        --[MicroBlaze Starting Up  ]--

Mutex Locked
Rcvd the message -->

        --[LED On  ]--

Mutex Unlocked
Rcvd the message -->

        --[LED Off ]--

Mutex Locked
Rcvd the message -->

        --[LED On  ]--

Mutex Unlocked
Rcvd the message -->

        --[LED Off ]--
```