Creating Your First Ultra96V2 Application

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

Workshop Pre-requisites

Pre-Lab

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

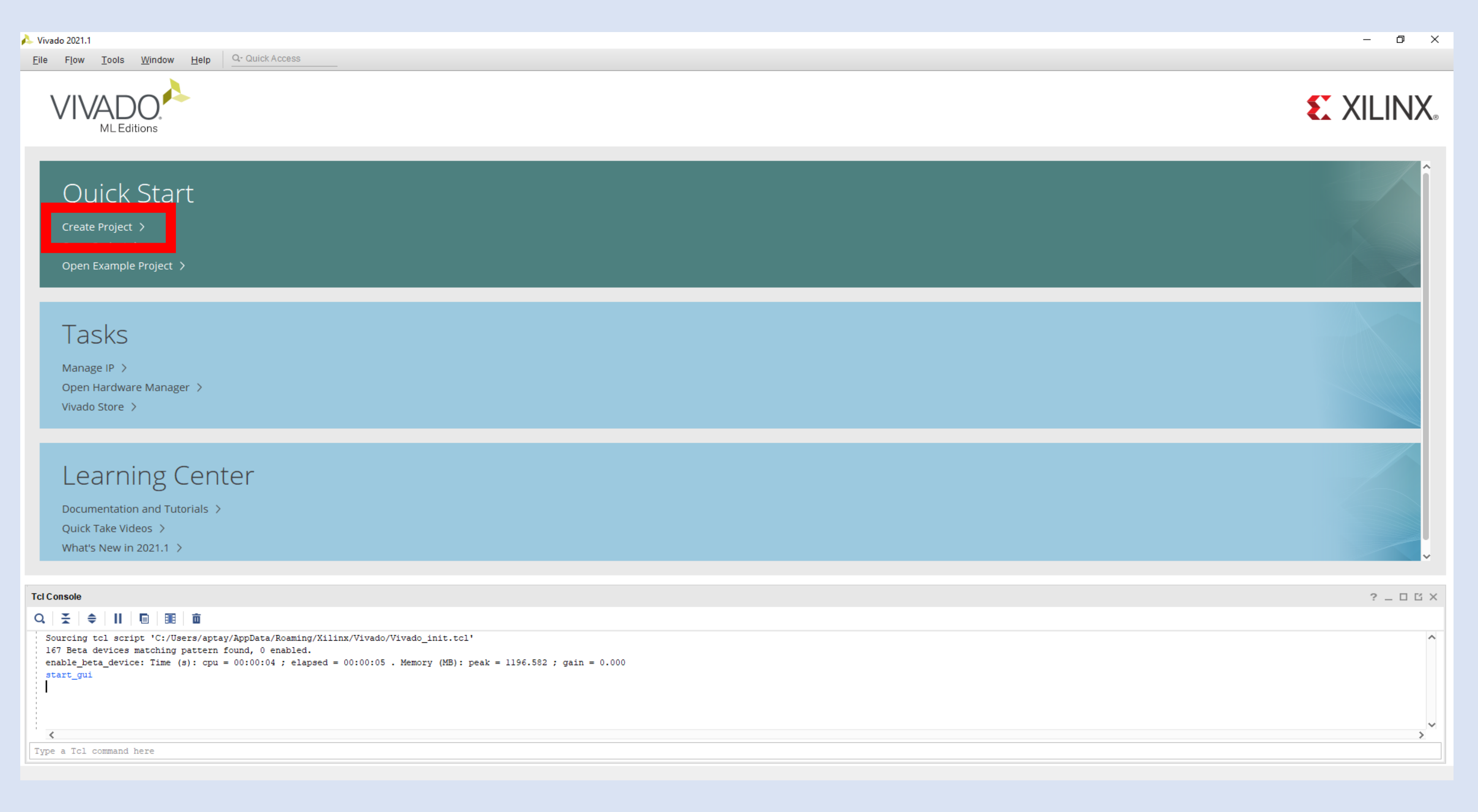
Vitis 2021.1	Download

Lab 1

Project creation & Flow

Lab 1: Understanding Vitis Project creation & Flow

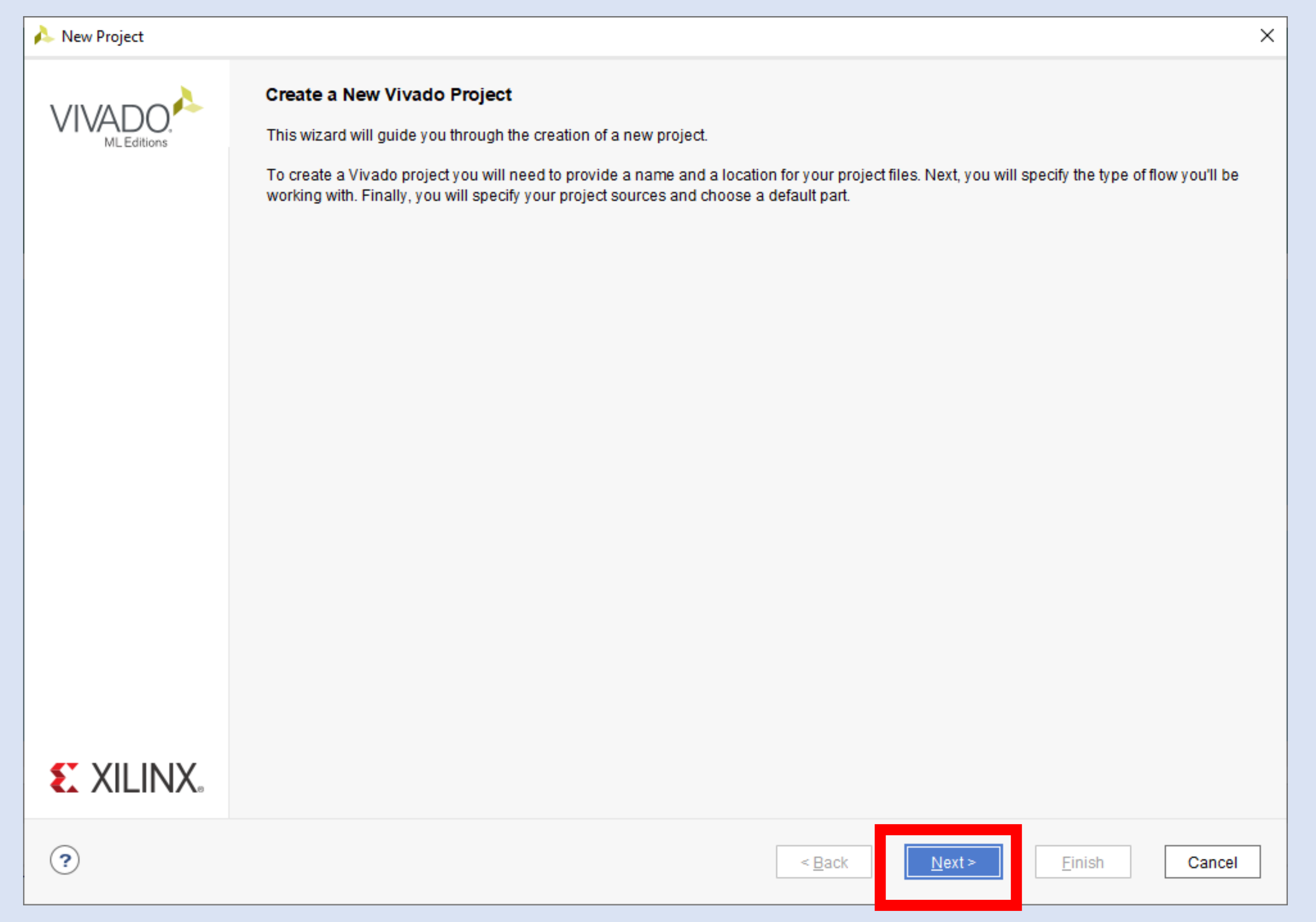
Step 1 – Open Vivado 2021.1 – Click Create New Project



```
Sourcing tcl script 'C:/Users/aptay/AppData/Roaming/Xilinx/Vivado/Vivado_init.tcl'
167 Beta devices matching pattern found, 0 enabled.
enable_beta_device: Time (s): cpu = 00:00:04 ; elapsed = 00:00:05 . Memory (MB): peak = 1196.582 ; gain = 0.000
start_gui
```

Lab 1: Understanding Vitis Project creation & Flow

Step 2 – Click Next



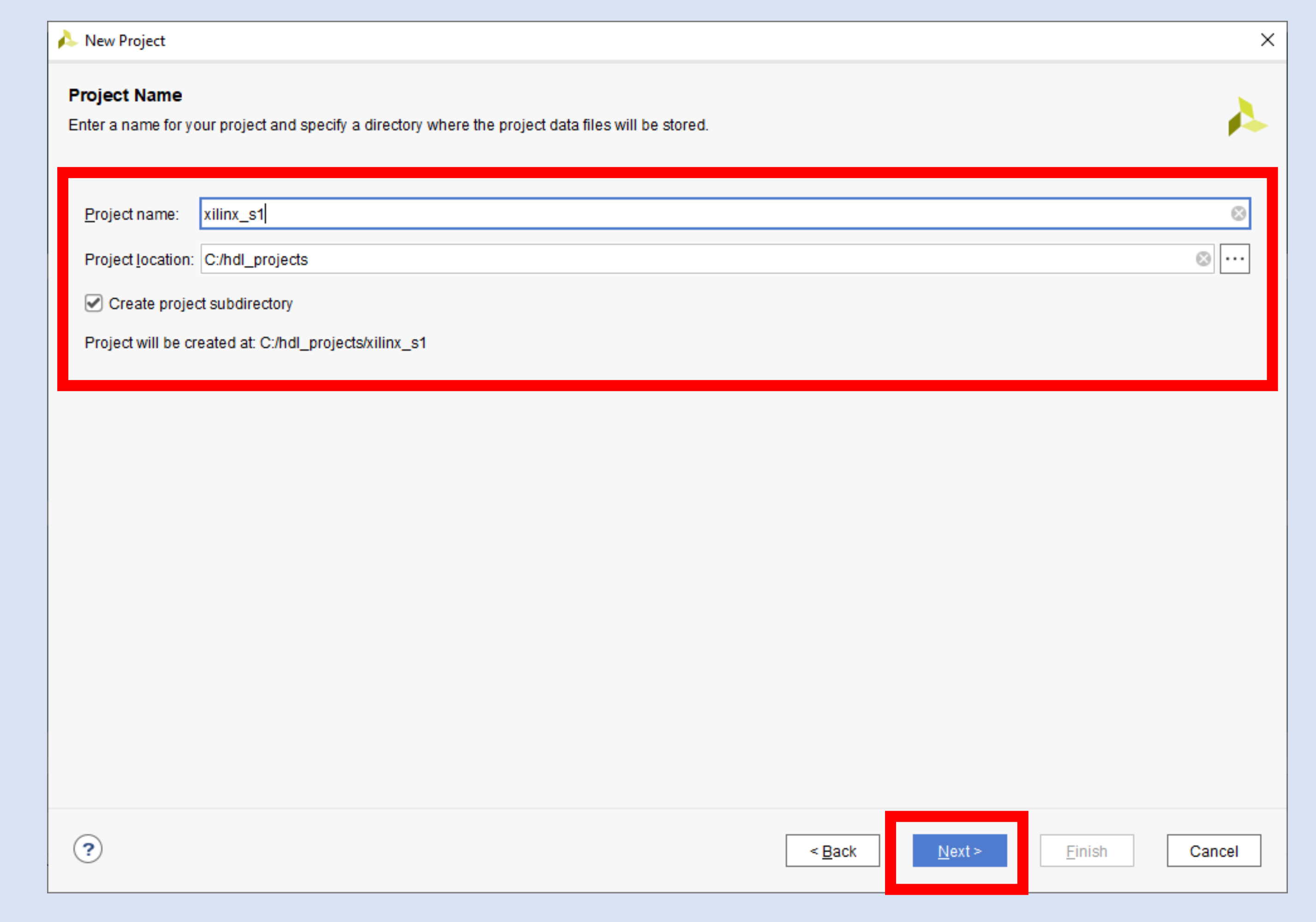
Create a New Vivado Project

This wizard will guide you through the creation of a new project.

To create a Vivado project you will need to provide a name and a location for your project files. Next, you will specify the type of flow you'll be working with. Finally, you will specify your project sources and choose a default part.

Lab 1: Understanding Vitis Project creation & Flow

Step 3 – Enter a project name and location to save the project, click next



Project Name

Enter a name for your project and specify a directory where the project data files will be stored.

Project name: xilinx_s1

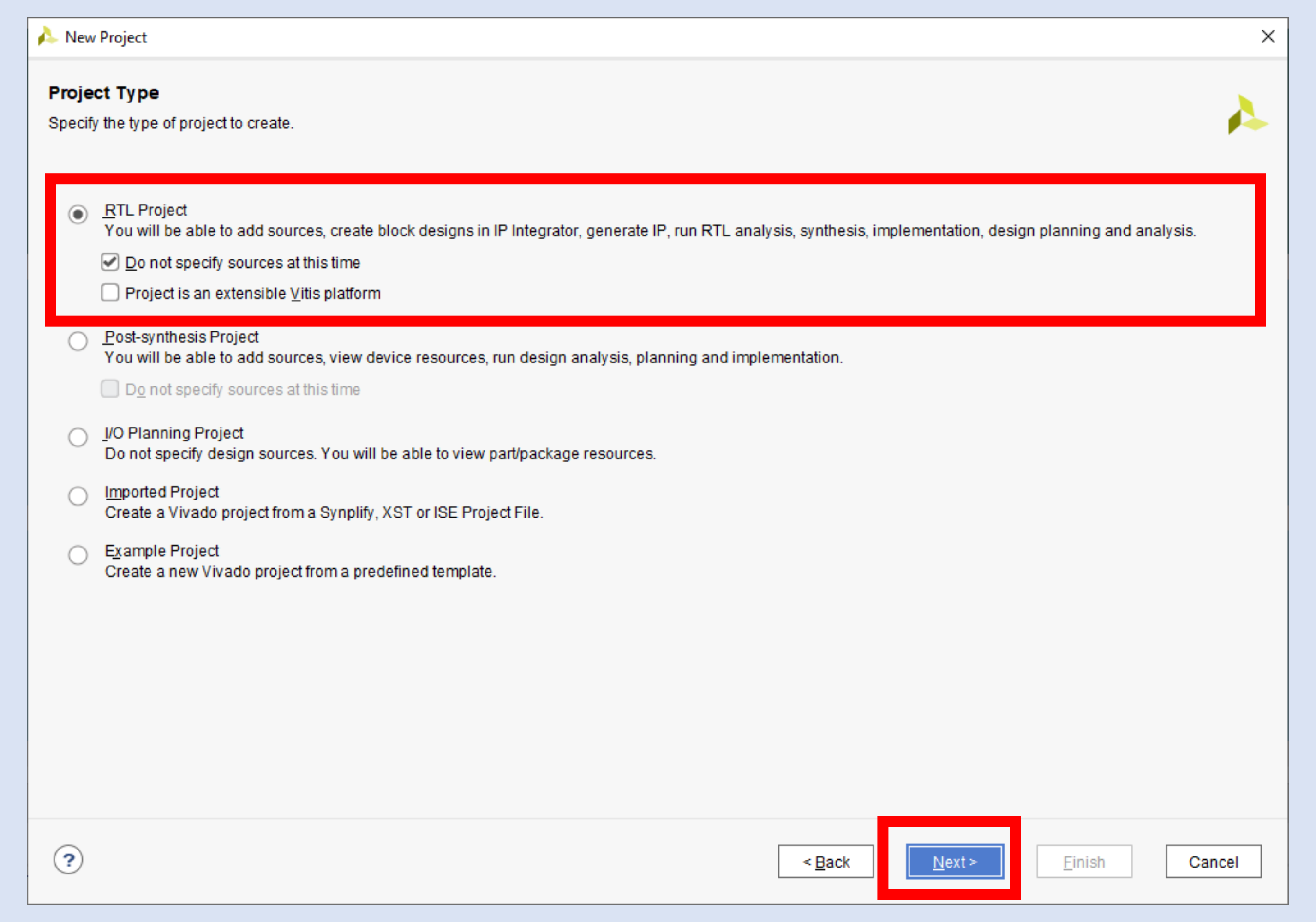
Project location: C:/hdl_projects

☑ Create project subdirectory

Project will be created at: C:/hdl_projects/xilinx_s1

Lab 1: Understanding Vitis Project creation & Flow

Step 3 – Select RTL project and check do not include sources, click next



New Project

Project Type

Specify the type of project to create.

● RTL Project
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
☑ Do not specify sources at this time
□ Project is an extensible Vitis platform

○ Post-synthesis Project
You will be able to add sources, view device resources, run design analysis, planning and implementation.
□ Do not specify sources at this time

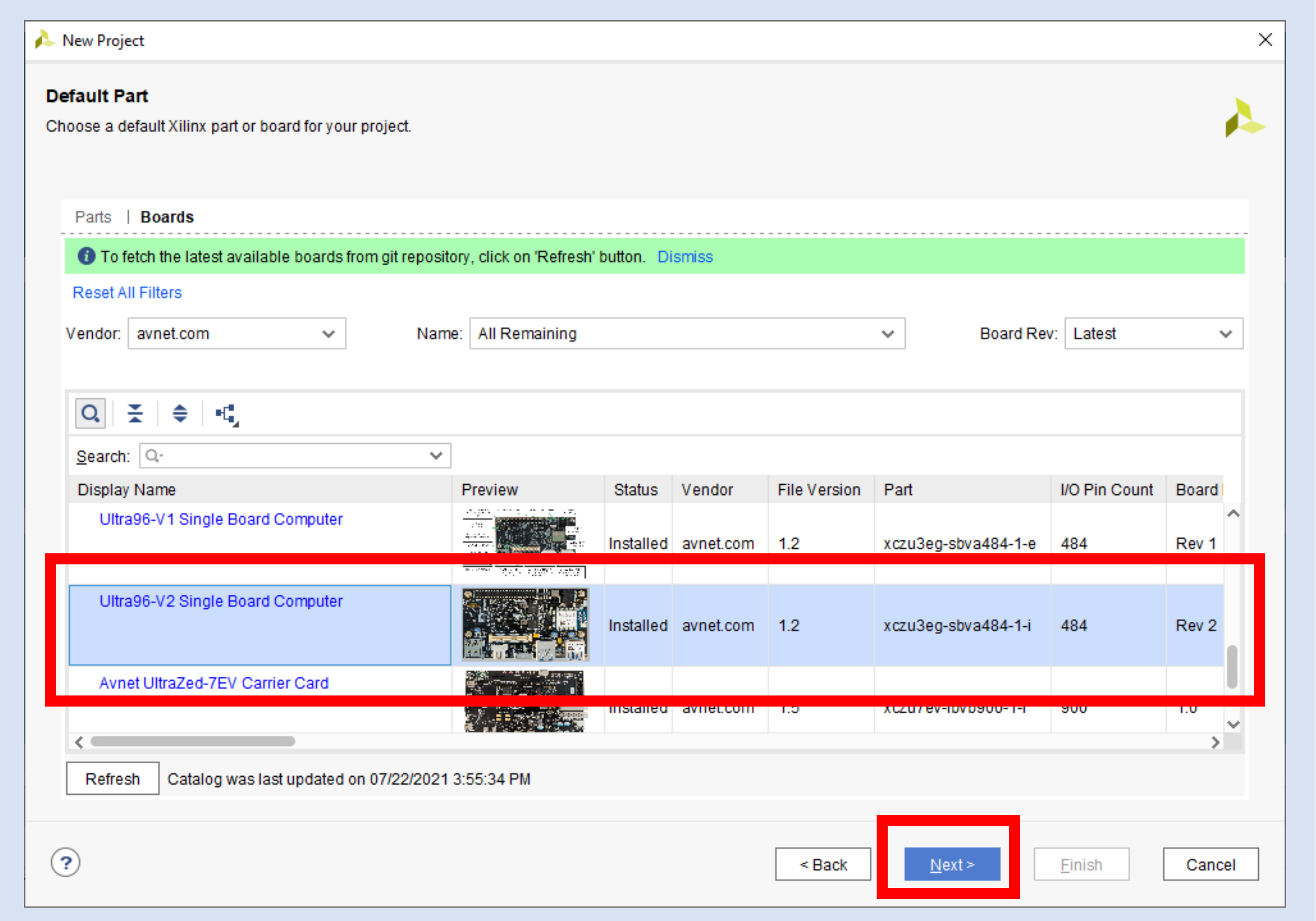
○ I/O Planning Project
Do not specify design sources. You will be able to view part/package resources.

○ Imported Project
Create a Vivado project from a Synplify, XST or ISE Project File.

○ Example Project
Create a new Vivado project from a predefined template.

Lab 1: Understanding Vitis Project creation & Flow

Step 4 – Select Vendor Avnet and Ultra96V2, click next



New Project

Default Part

Choose a default Xilinx part or board for your project.

Parts | **Boards**

To fetch the latest available boards from git repository, click on 'Refresh' button. Dismiss

Reset All Filters

Vendor: avnet.com   Name: All Remaining   Board Rev: Latest

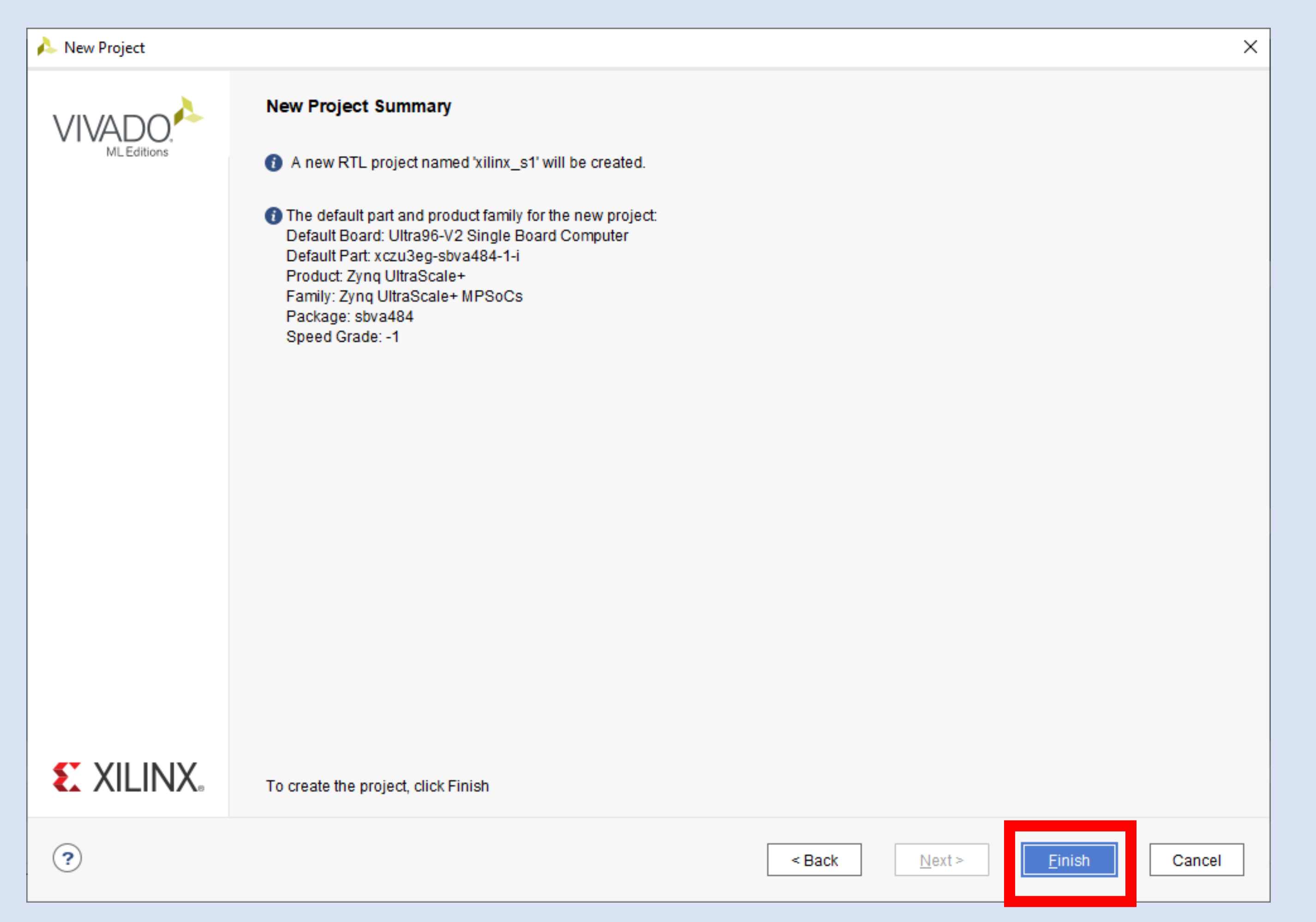
Search:

Display Name	Preview	Status	Vendor	File Version	Part	I/O Pin Count	Board
Ultra96-V1 Single Board Computer		Installed	avnet.com	1.2	xczu3eg-sbva484-1-e	484	Rev 1
Ultra96-V2 Single Board Computer		Installed	avnet.com	1.2	xczu3eg-sbva484-1-i	484	Rev 2
Avnet UltraZed-7EV Carrier Card		Installed	avnet.com	1.5	xczu7ev-fbvb900-1-i	900	1.0

Refresh   Catalog was last updated on 07/22/2021 3:55:34 PM

Lab 1: Understanding Vitis Project creation & Flow

Step 5 – Click Finish



New Project Summary

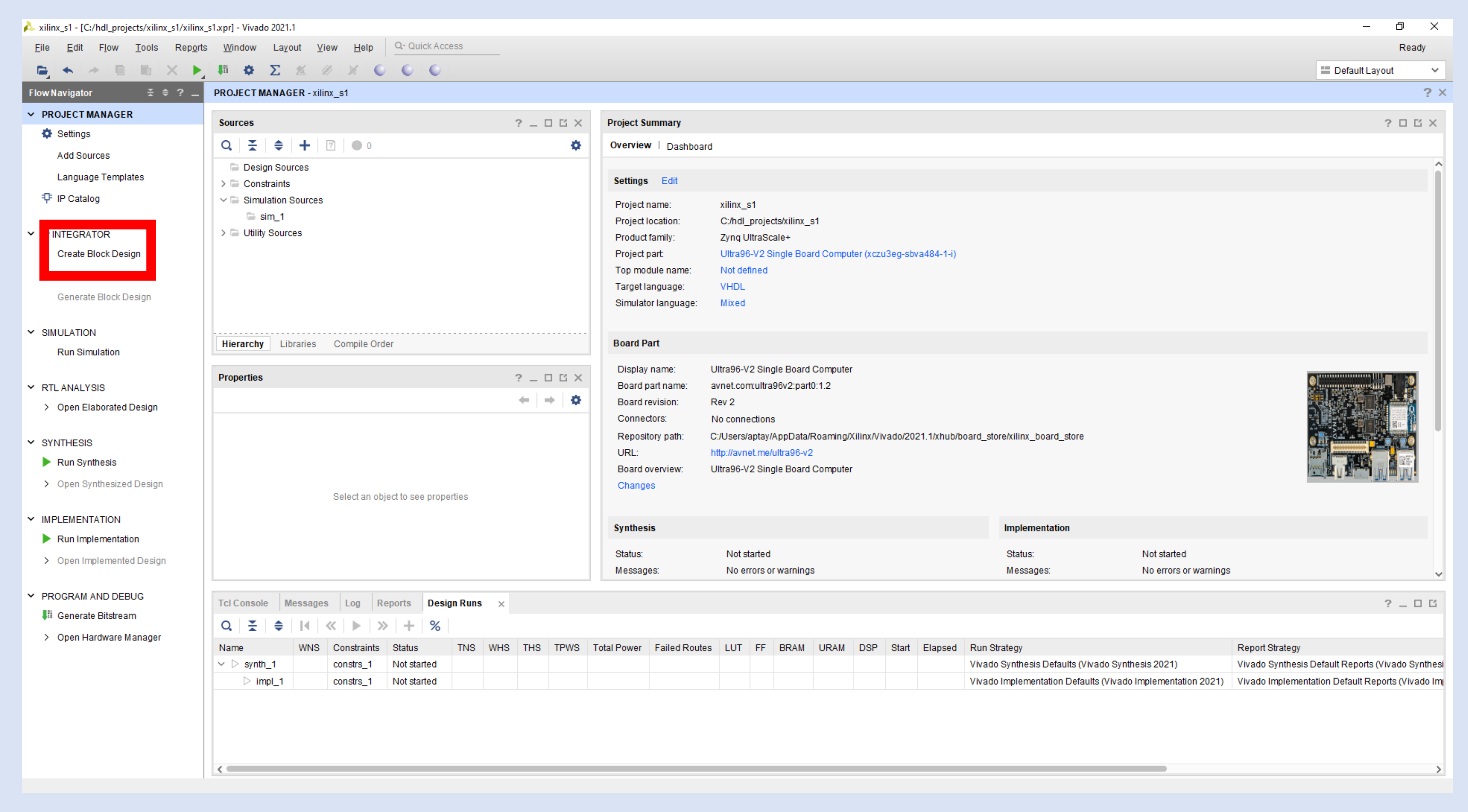
A new RTL project named 'xilinx_s1' will be created.

The default part and product family for the new project:
Default Board: Ultra96-V2 Single Board Computer
Default Part: xczu3eg-sbva484-1-i
Product: Zynq UltraScale+
Family: Zynq UltraScale+ MPSoCs
Package: sbva484
Speed Grade: -1

To create the project, click Finish

Lab 1: Understanding Vitis Project creation & Flow

Step 6 – Click on Create Block Diagram – This will open IP editor



Settings Edit

Project name: xilinx_s1
Project location: C:/hdl_projects/xilinx_s1
Product family: Zynq UltraScale+
Project part: Ultra96-V2 Single Board Computer (xczu3eg-sbva484-1-i)
Top module name: Not defined
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: Not started
Messages: No errors or warnings

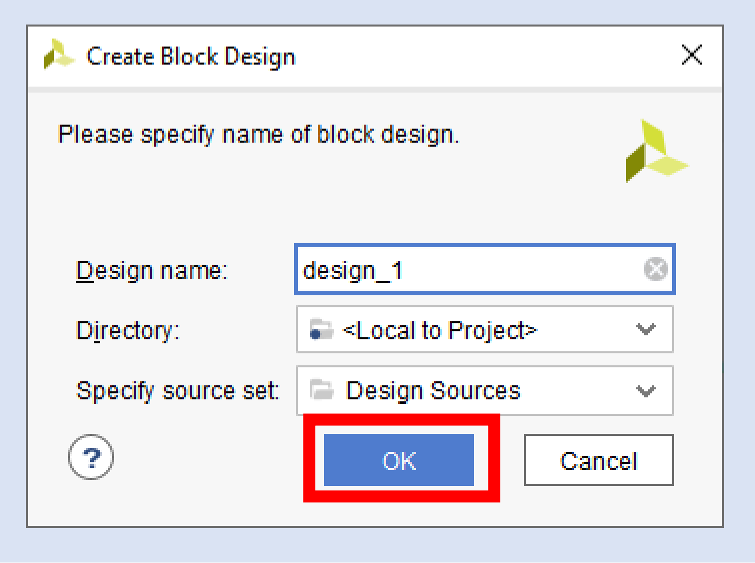
Implementation

Status: Not started
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 Strategy
synth_1		constrs_1	Not started														Vivado Synthesis Defaults (Vivado Synthesis 2021)	Vivado Synthesis Default Reports (Vivado Synthesi
impl_1		constrs_1	Not started														Vivado Implementation Defaults (Vivado Implementation 2021)	Vivado Implementation Default Reports (Vivado Im

Lab 1: Understanding Vitis Project creation & Flow

Step 7 – Leave the settings as default and click OK



Create Block Design

Please specify name of block design.

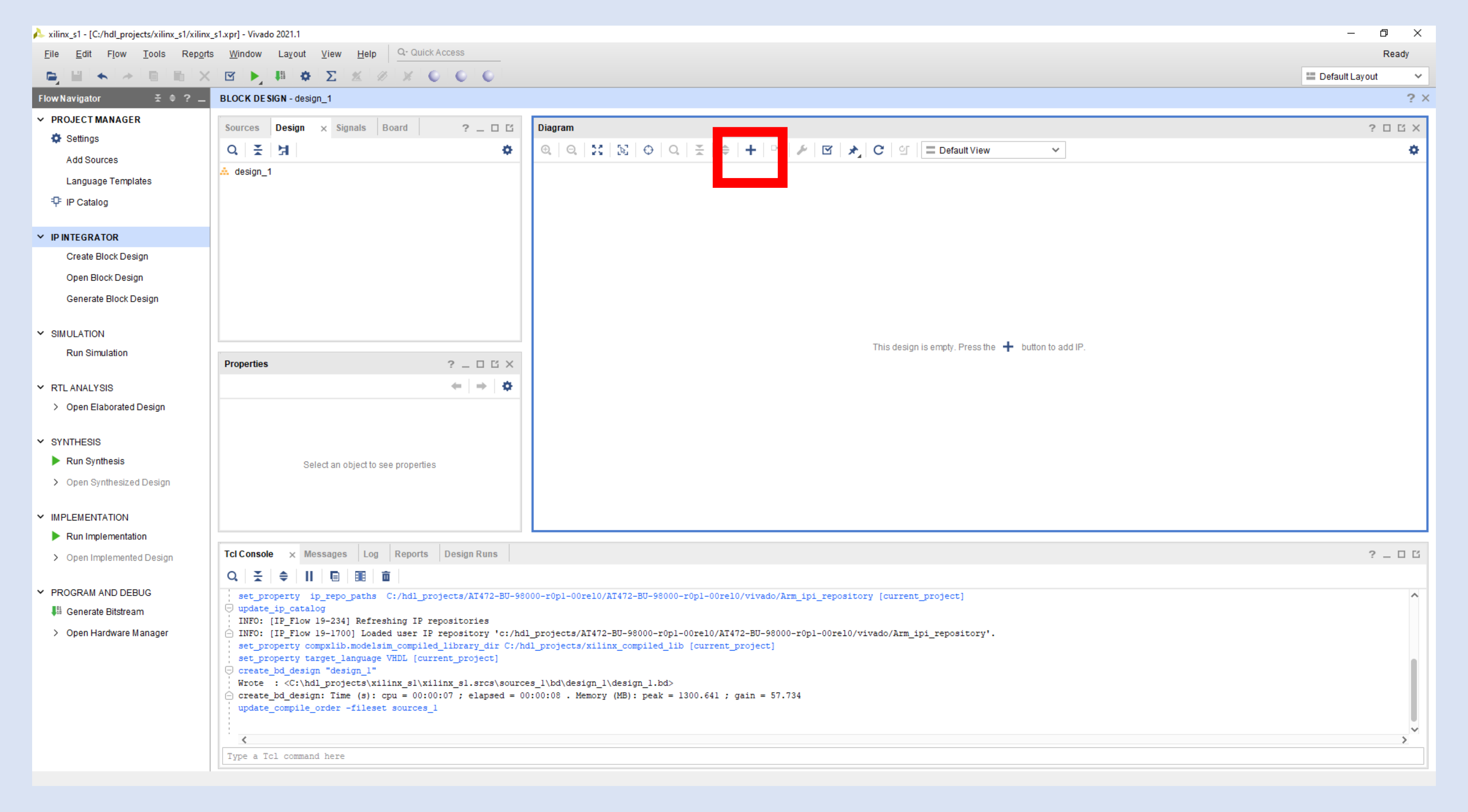
Design name: design_1

Directory: <Local to Project>

Specify source set: Design Sources

Lab 1: Understanding Vitis Project creation & Flow

Step 8 – Click on the + button to open the IP list

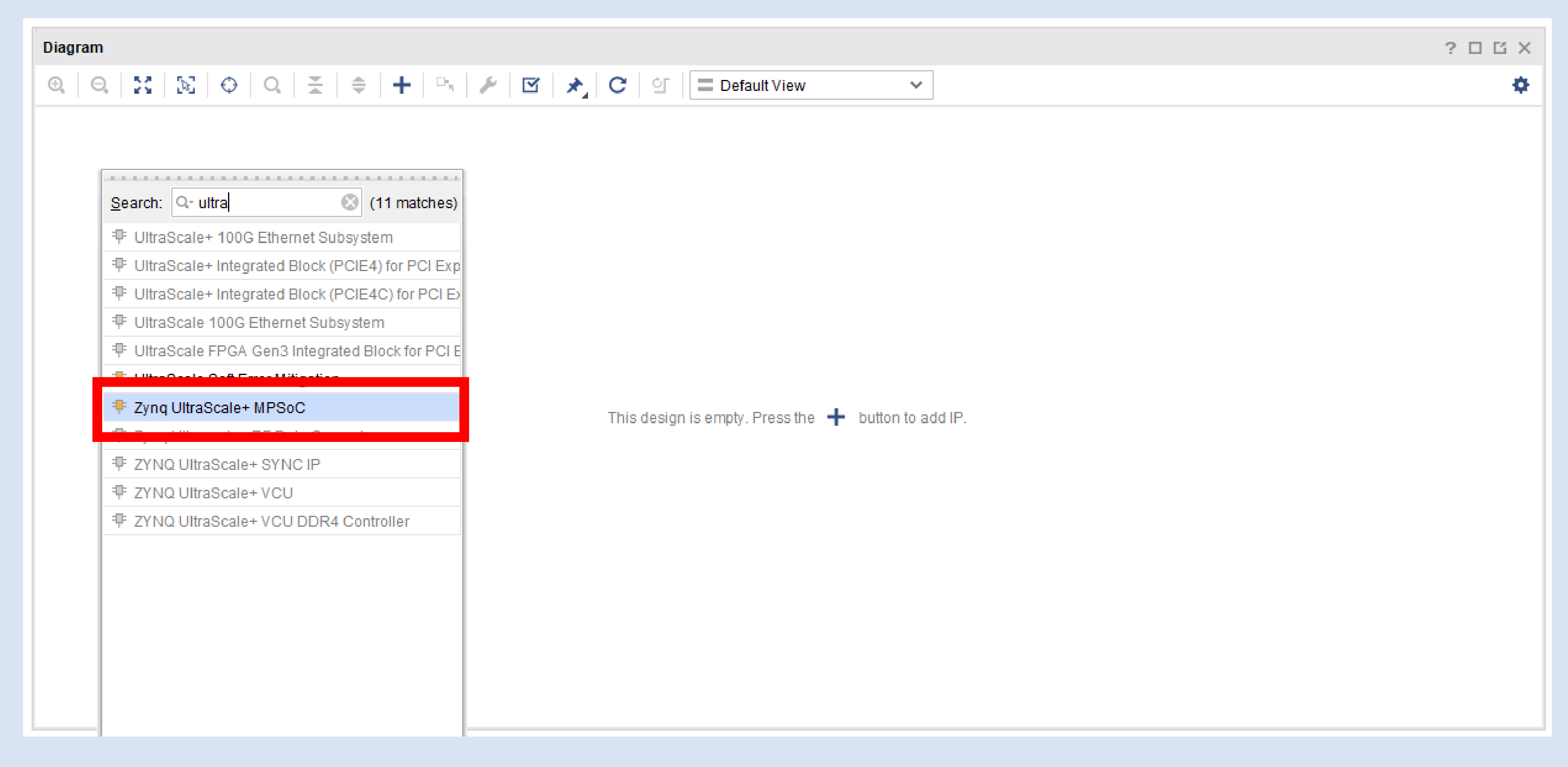


```
set_property  ip_repo_paths  C:/hdl_projects/AT472-BU-98000-r0p1-00rel0/AT472-BU-98000-r0p1-00rel0/vivado/Arm_ipi_repository [current_project]
update_ip_catalog
INFO: [IP_Flow 19-234] Refreshing IP repositories
INFO: [IP_Flow 19-1700] Loaded user IP repository 'c:/hdl_projects/AT472-BU-98000-r0p1-00rel0/AT472-BU-98000-r0p1-00rel0/vivado/Arm_ipi_repository'.
set_property compxlib.modelsim_compiled_library_dir C:/hdl_projects/xilinx_compiled_lib [current_project]
set_property target_language VHDL [current_project]
create_bd_design "design_1"
Wrote  : <C:\hdl_projects\xilinx_s1\xilinx_s1.srcs\sources_1\bd\design_1\design_1.bd>
create_bd_design: Time (s): cpu = 00:00:07 ; elapsed = 00:00:08 . Memory (MB): peak = 1300.641 ; gain = 57.734
update_compile_order -fileset sources_1
```

Type a Tcl command here

Lab 1: Understanding Vitis Project creation & Flow

Step 9 – In the search bar, type Ultra and select the Zynq UltraScale+ MPSoC block. Double click on this to insert the IP block.



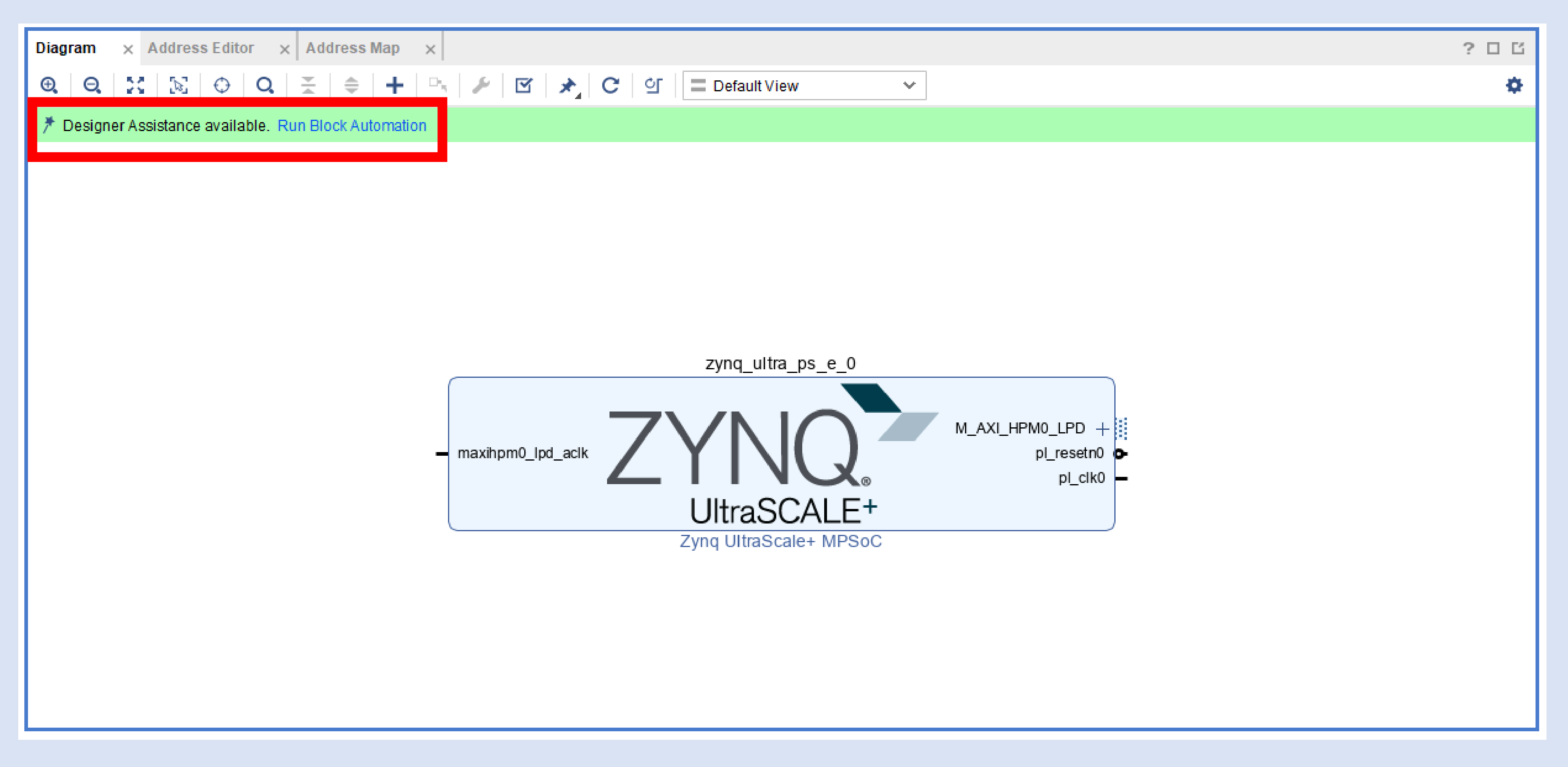
Search: ultra (11 matches)

This design is empty. Press the + button to add IP.

Diagram
Default View
UltraScale+ 100G Ethernet Subsystem
UltraScale+ Integrated Block (PCIE4) for PCI Exp
UltraScale+ Integrated Block (PCIE4C) for PCI E
UltraScale 100G Ethernet Subsystem
UltraScale FPGA Gen3 Integrated Block for PCI E
UltraScale Soft Error Mitigation
Zynq UltraScale+ MPSoC
ZYNQ UltraScale+ SYNC IP
ZYNQ UltraScale+ VCU
ZYNQ UltraScale+ VCU DDR4 Controller

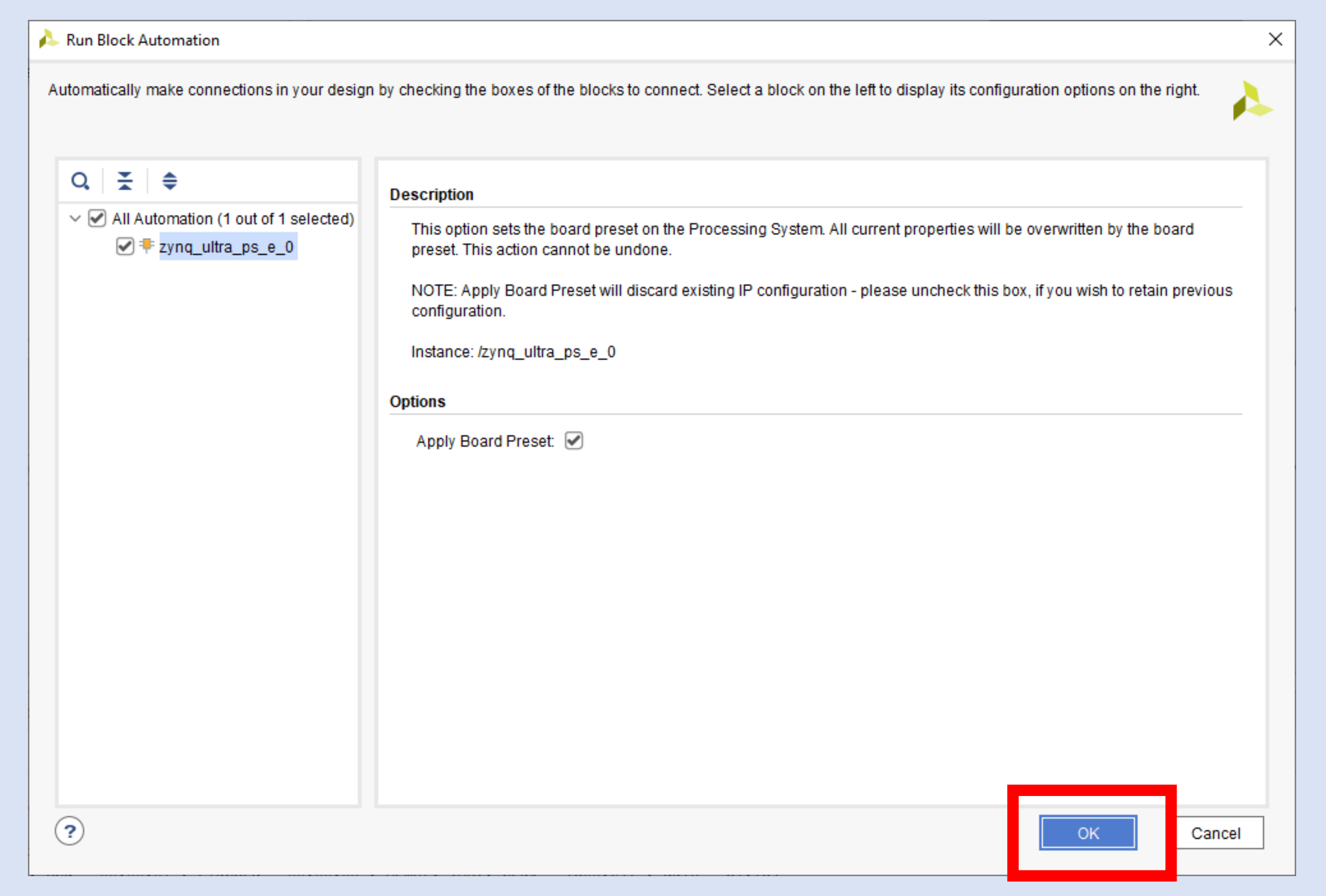
Lab 1: Understanding Vitis Project creation & Flow

Step 10 – Select Run Block Automation – This will configure the Processing System for the ULTRA96V2 setting e.g. DDR timing, Clocking etc.


Designer Assistance available. Run Block Automation

Lab 1: Understanding Vitis Project creation & Flow

Step 11 – Click on the OK button, the automation will run and configure the processing block for the Ultra96V2.



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)
  - ☑ zynq_ultra_ps_e_0

Description

This option sets the board preset on the Processing System. All current properties will be overwritten by the board preset. This action cannot be undone.

NOTE: Apply Board Preset will discard existing IP configuration - please uncheck this box, if you wish to retain previous configuration.

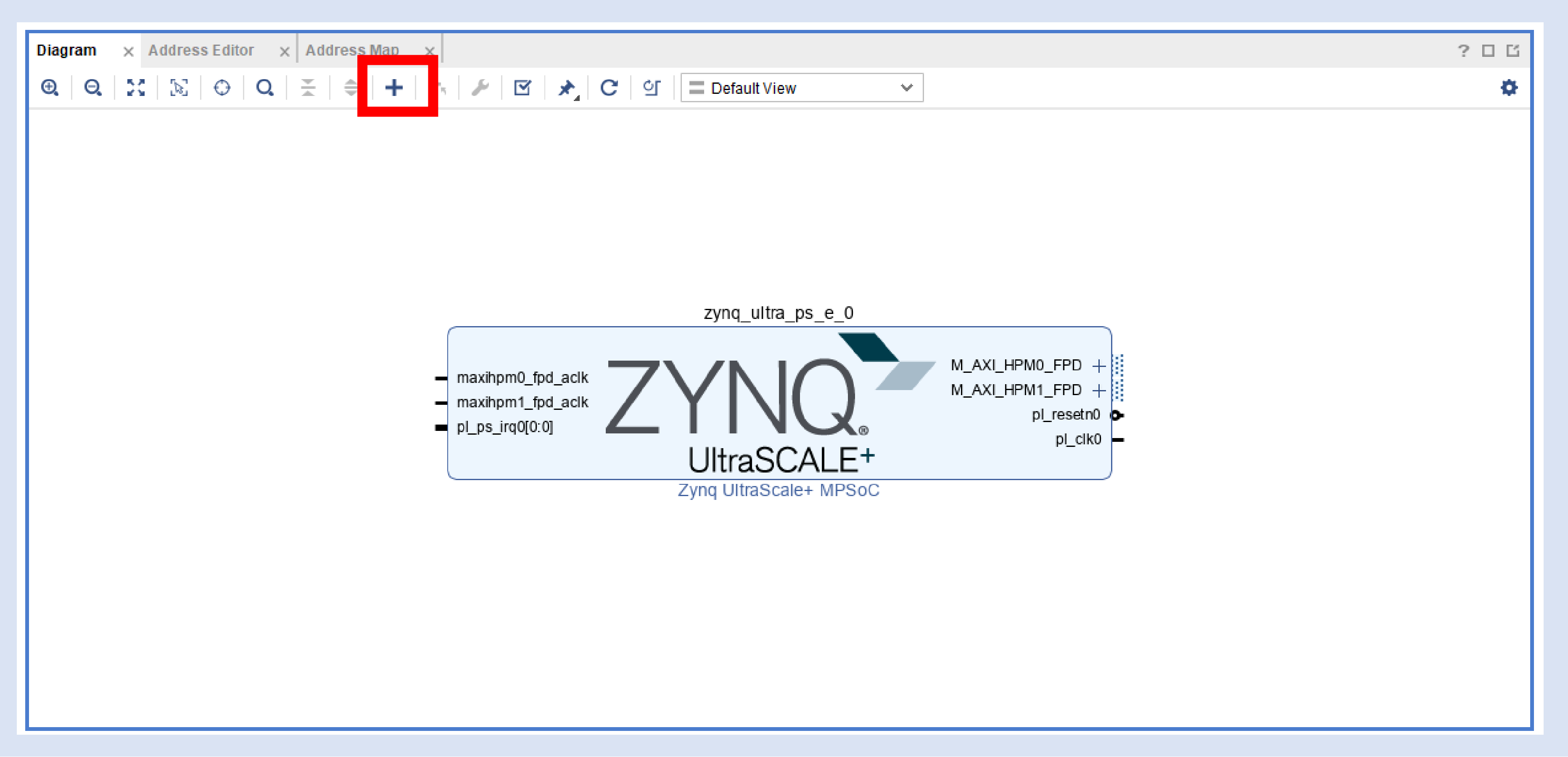
Instance: /zynq_ultra_ps_e_0

Options

Apply Board Preset: ☑

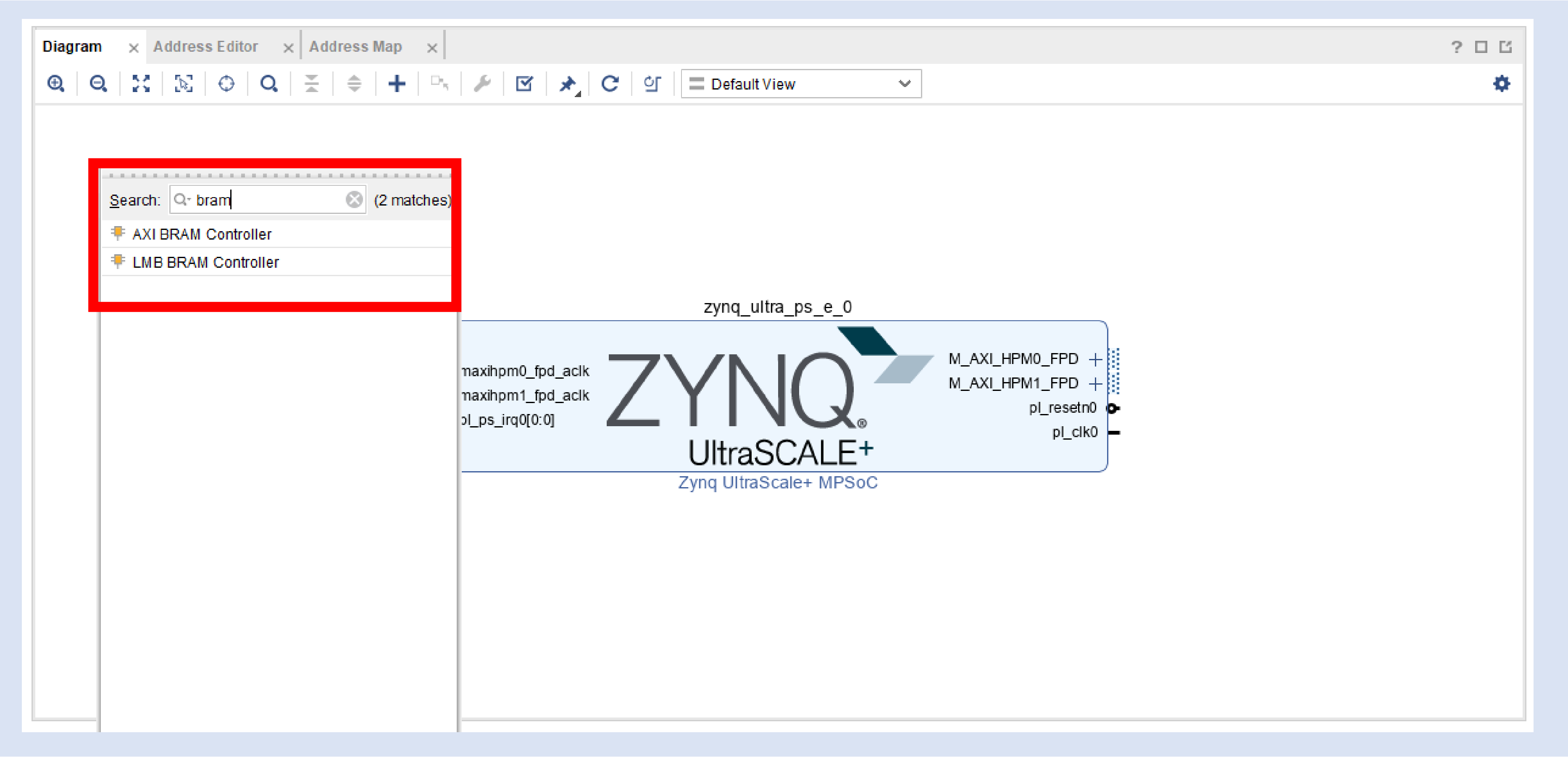
Lab 1: Understanding Vitis Project creation & Flow

Step 12 – Click on the + Symbol



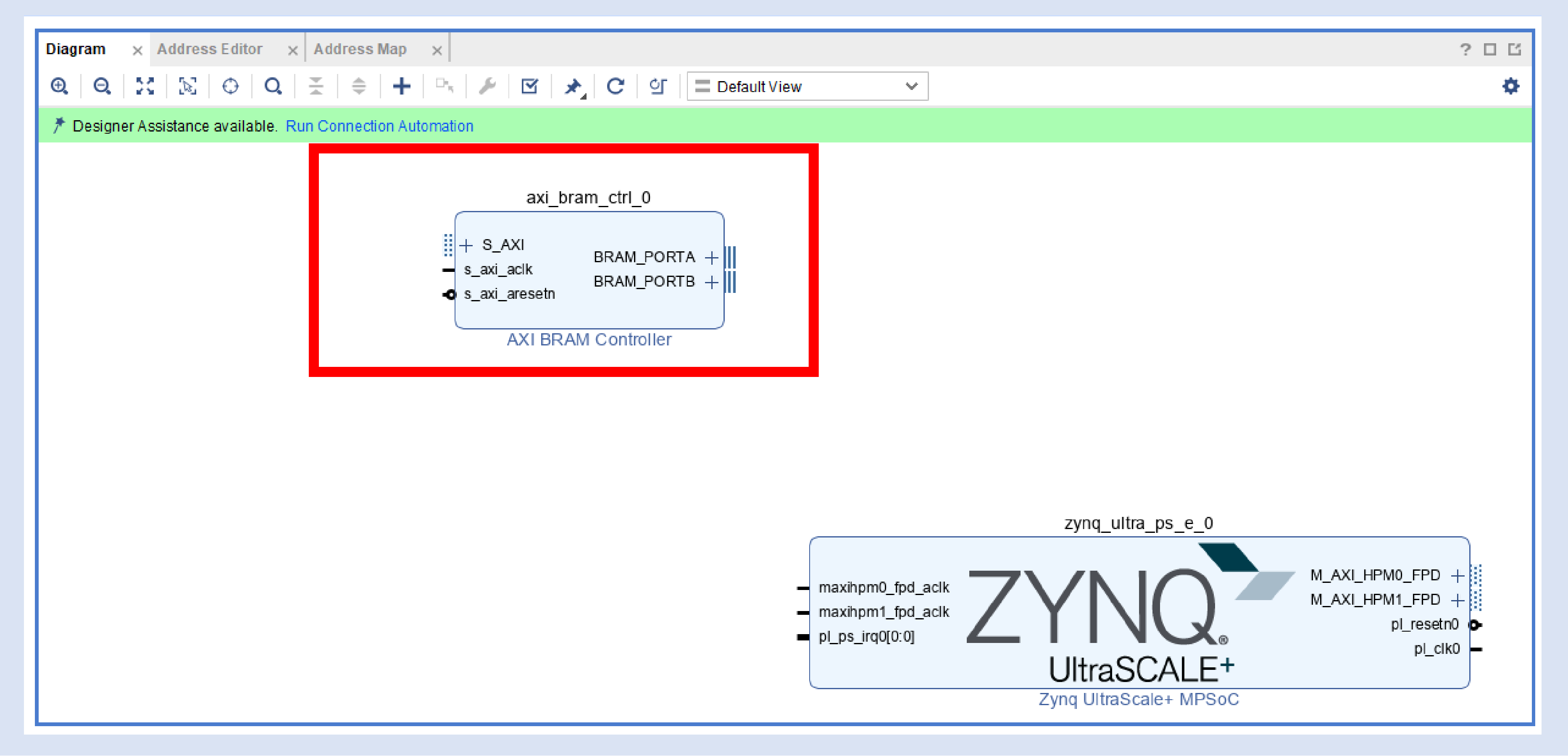
Lab 1: Understanding Vitis Project creation & Flow

Step 13 – Type in BRAM and double click on AXI BRAM Controller to add the IP



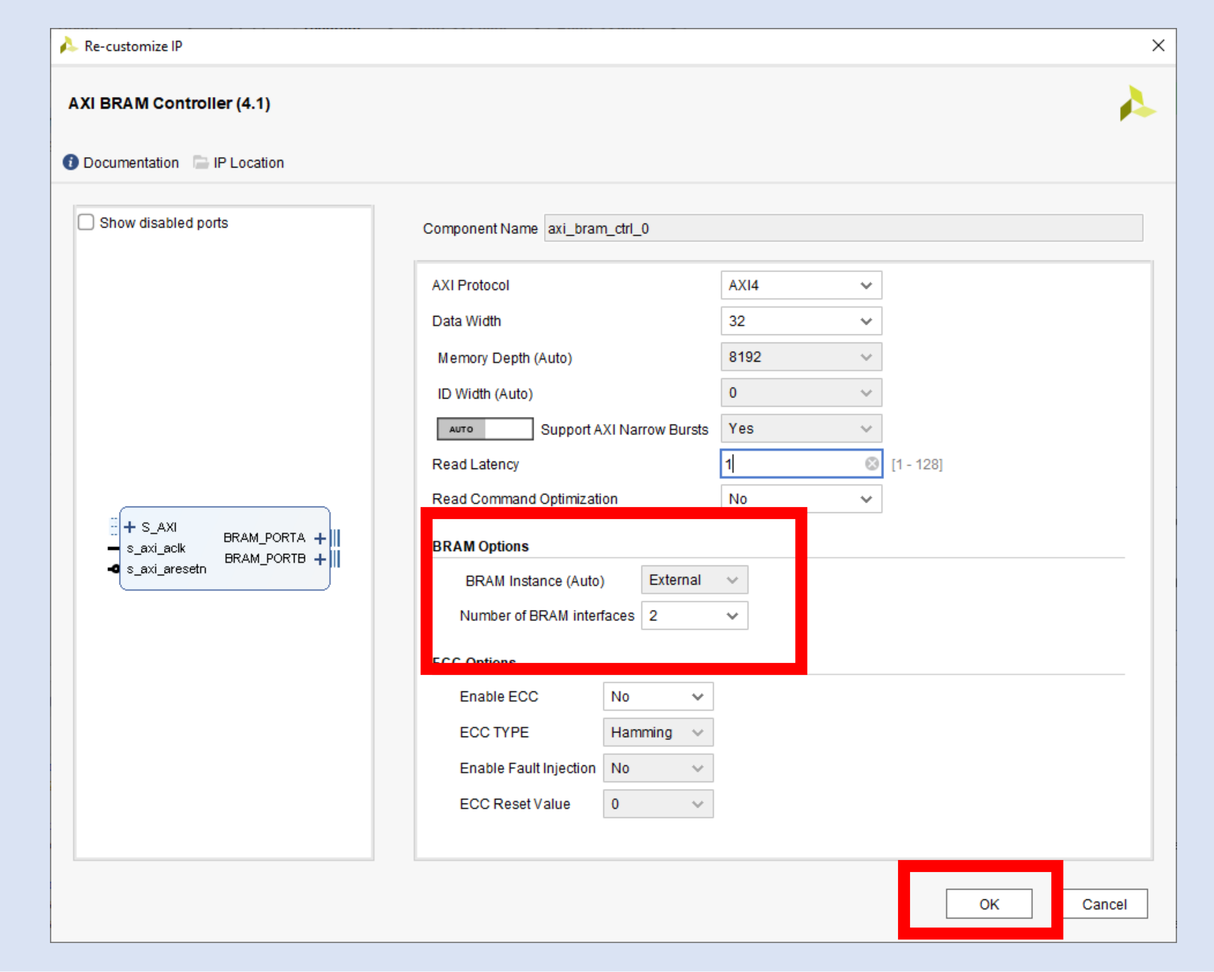
Lab 1: Understanding Vitis Project creation & Flow

Step 14 – Double click on the AXI BRAM Controller



Lab 1: Understanding Vitis Project creation & Flow

Step 15 – Change the Number of BRAM Interfaces to 1, click OK



Re-customize IP

AXI BRAM Controller (4.1)

Documentation IP Location

☐ Show disabled ports

Component Name axi_bram_ctrl_0

AXI Protocol AXI4

Data Width 32

Memory Depth (Auto) 8192

ID Width (Auto) 0

Support AXI Narrow Bursts Yes

Read Latency 1 [1 - 128]

Read Command Optimization No

BRAM Options

BRAM Instance (Auto) External

Number of BRAM interfaces 2

ECC Options

Enable ECC No

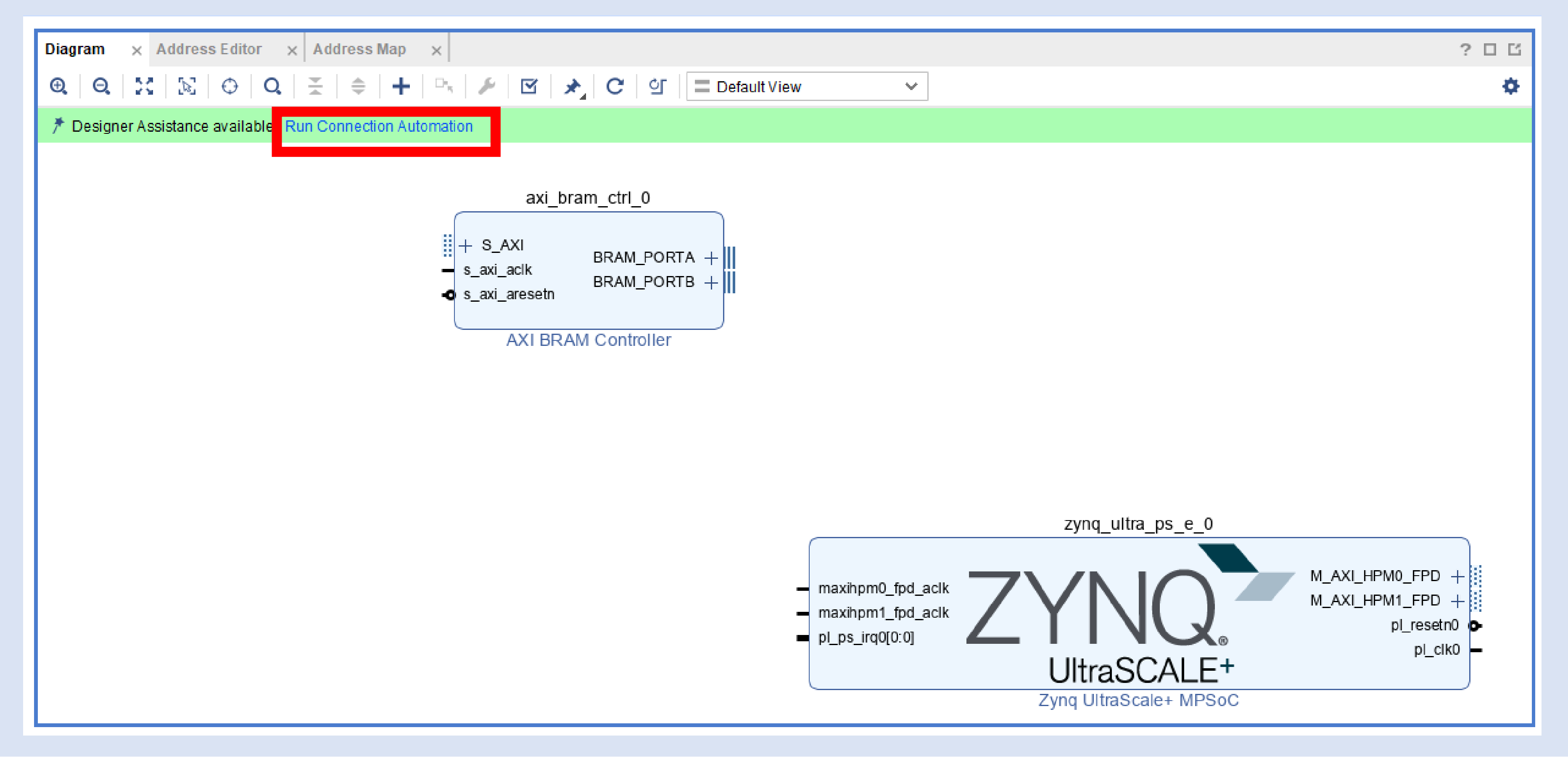
ECC TYPE Hamming

Enable Fault Injection No

ECC Reset Value 0

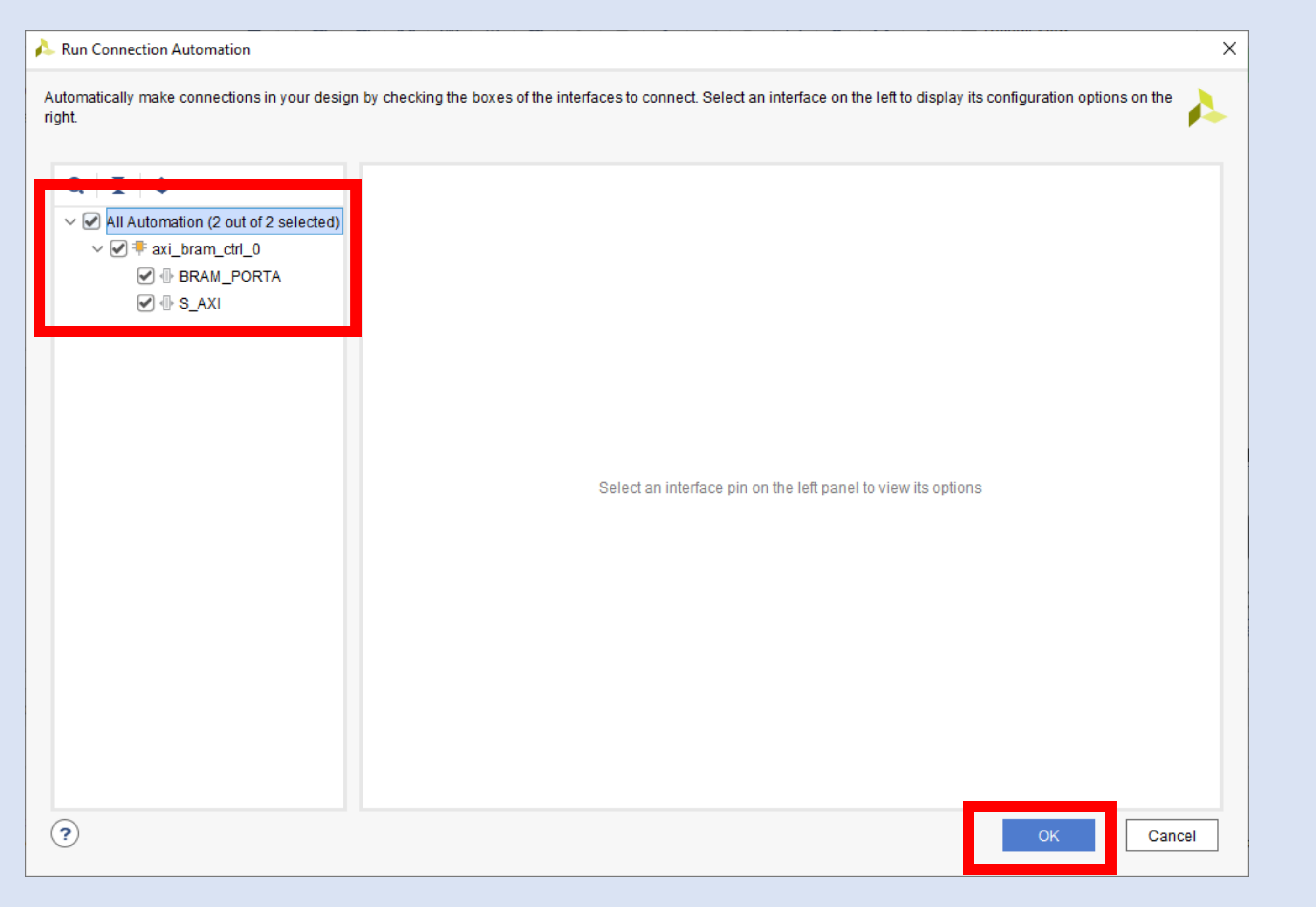
Lab 1: Understanding Vitis Project creation & Flow

Step 16 – Click on Run Connection Automation



Lab 1: Understanding Vitis Project creation & Flow

Step 17 – Check all the boxes, click OK



Run Connection Automation

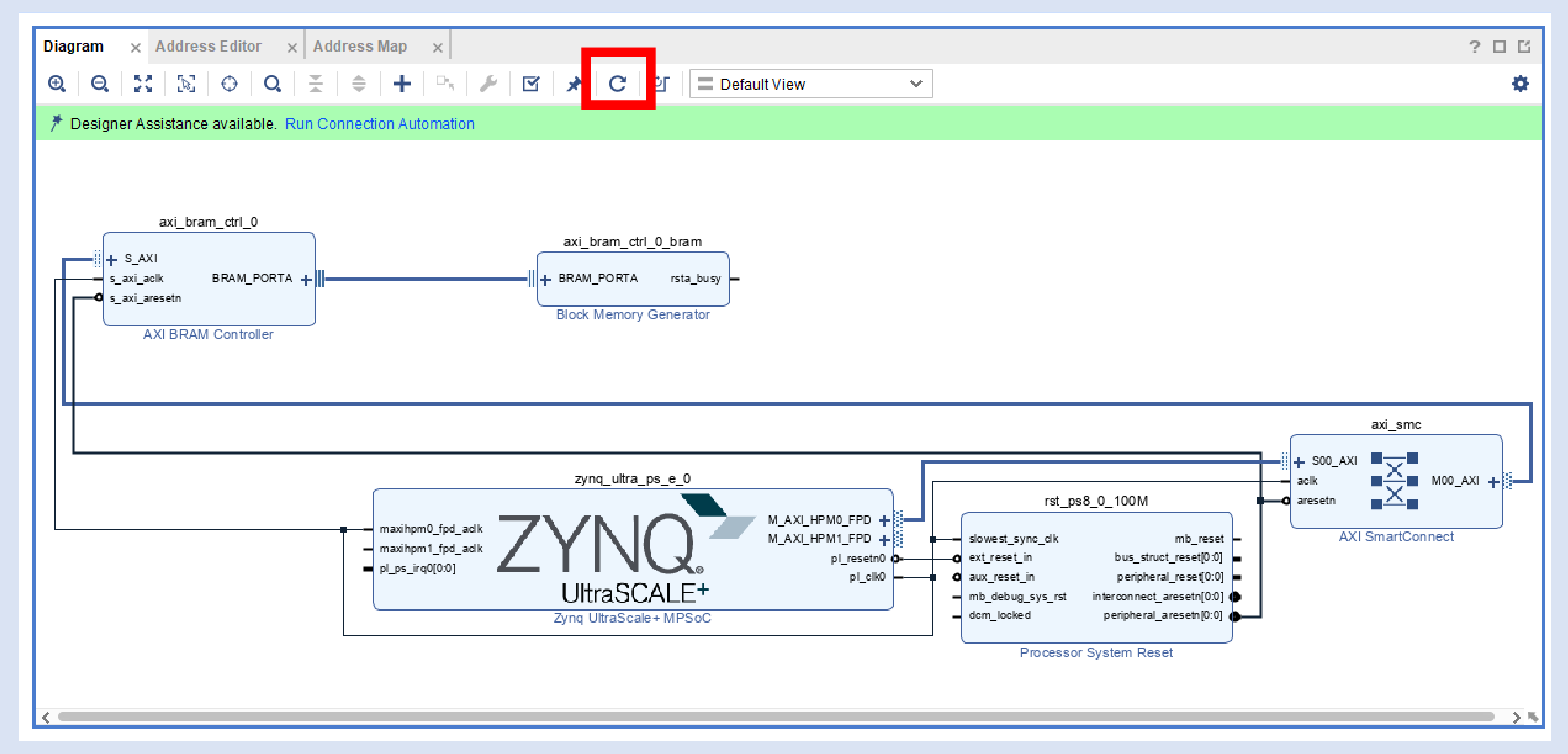
Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.

- ☑ All Automation (2 out of 2 selected)
  - ☑ axi_bram_ctrl_0
    - ☑ BRAM_PORTA
    - ☑ S_AXI

Select an interface pin on the left panel to view its options

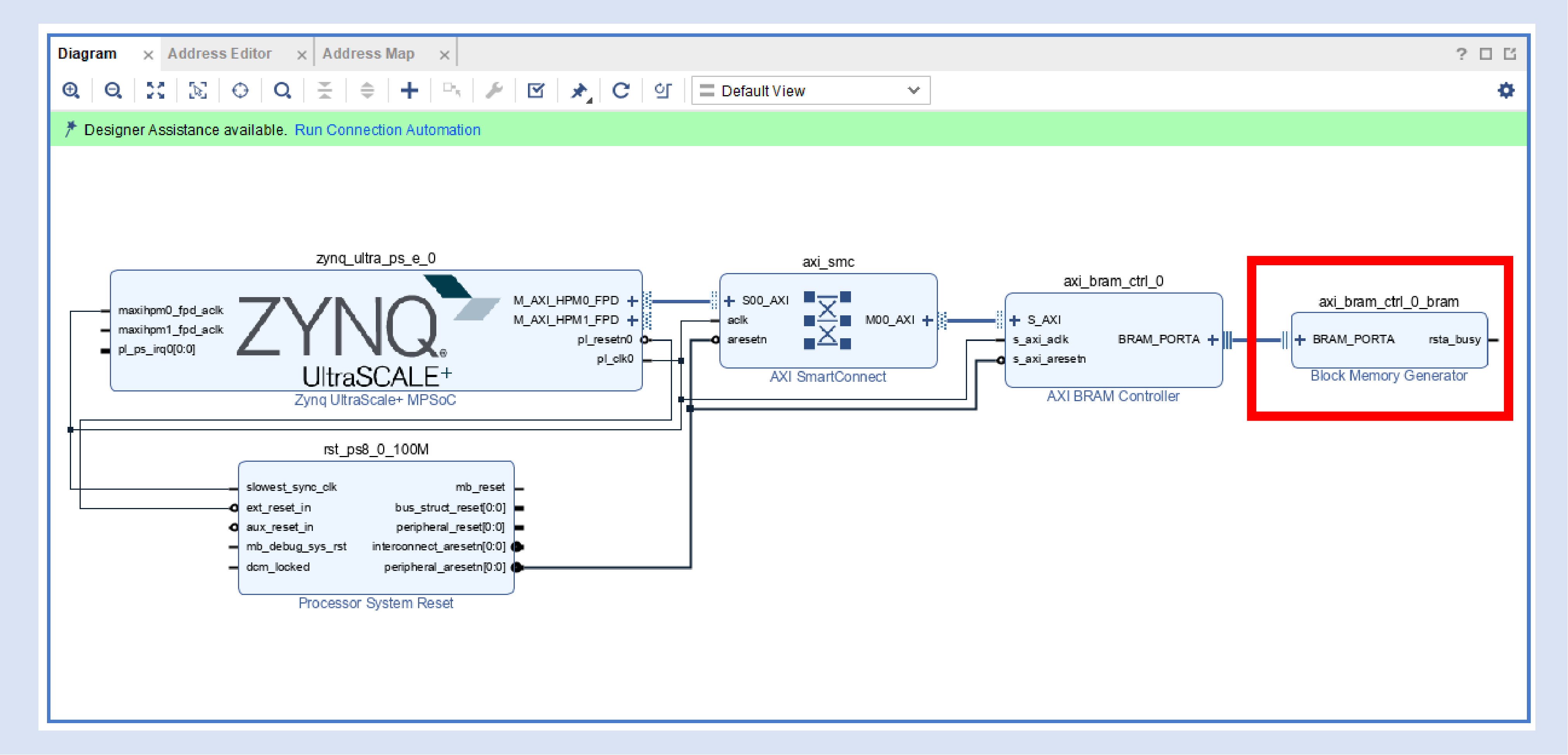
Lab 1: Understanding Vitis Project creation & Flow

Step 18 – Click on the Regenerate Layout – This will mage the diagram more logical



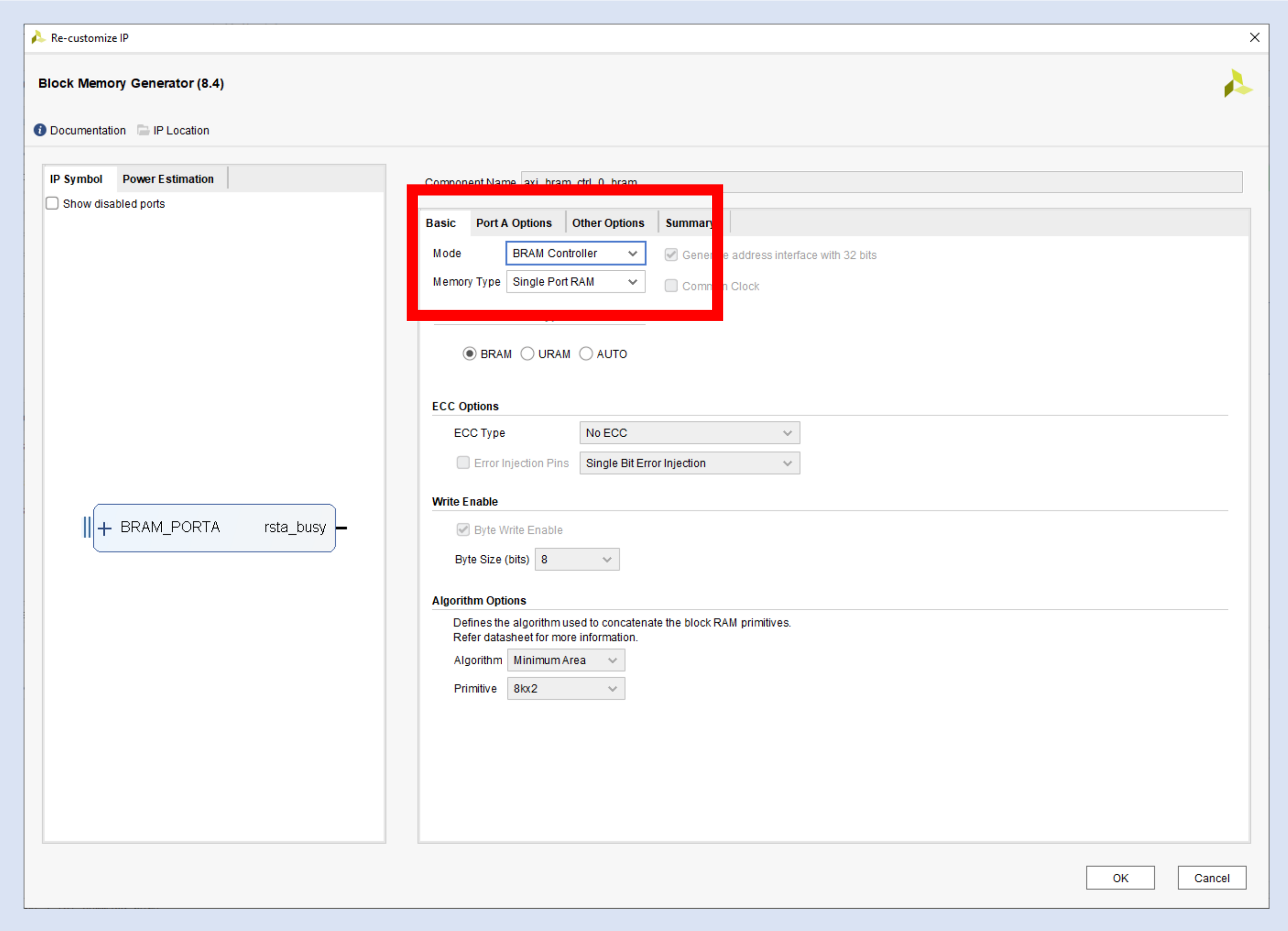
Lab 1: Understanding Vitis Project creation & Flow

Step 19 – Double Click on the BRAM Block



Lab 1: Understanding Vitis Project creation & Flow

Step 20 – Change the mode to Standalone, make sure check use 32 bit addressing is set



□ Show disabled ports

Component Name axi_bram_ctrl_0_bram

Mode: BRAM Controller

Memory Type: Single Port RAM

☑ Gene e address interface with 32 bits

□ Comm n Clock

● BRAM ○ URAM ○ AUTO

ECC Options

ECC Type: No ECC

□ Error Injection Pins: Single Bit Error Injection

Write Enable

☑ Byte Write Enable

Byte Size (bits): 8

Algorithm Options

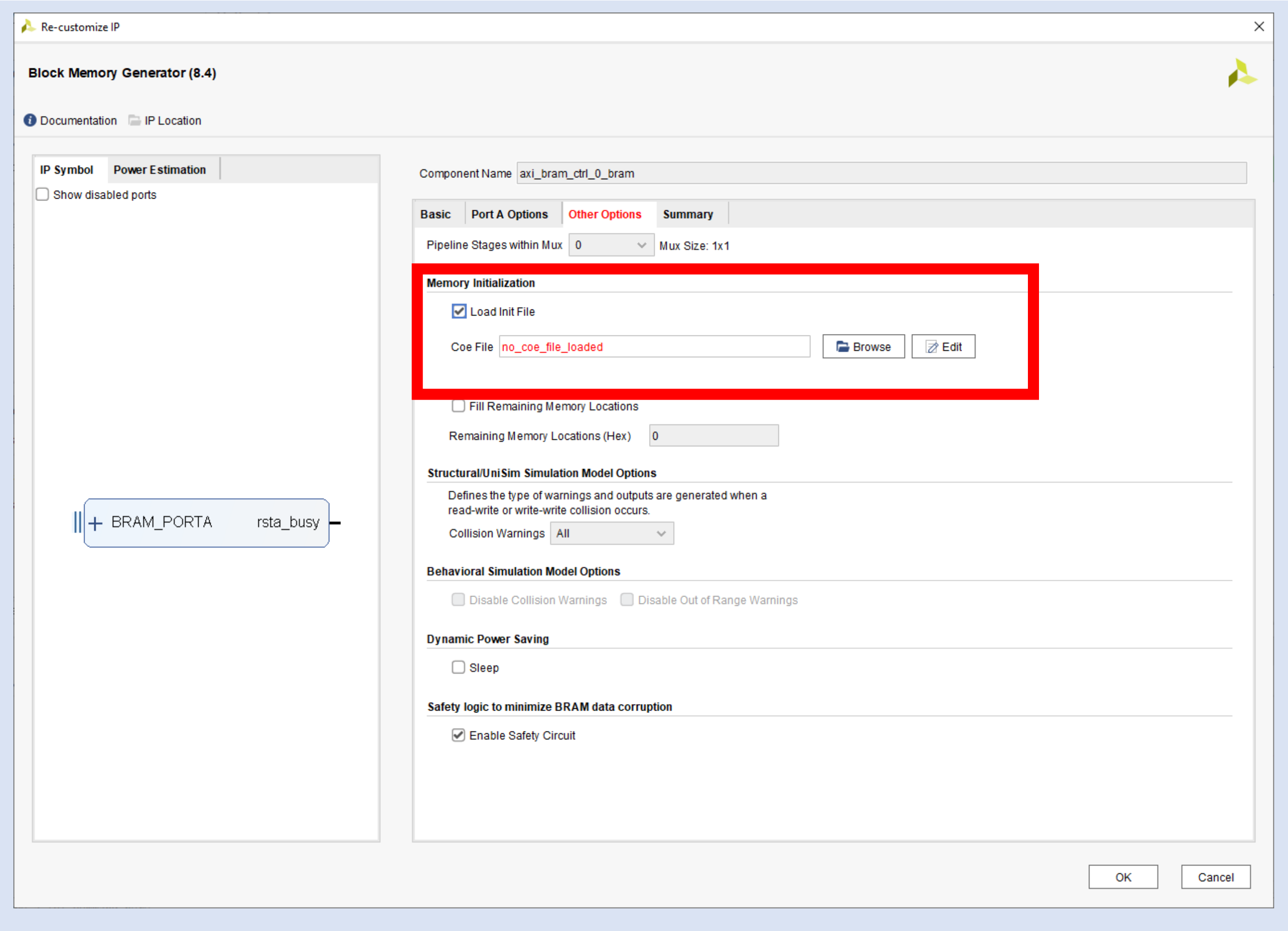
Defines the algorithm used to concatenate the block RAM primitives.
Refer datasheet for more information.

Algorithm: Minimum Area

Primitive: 8kx2

Lab 1: Understanding Vitis Project creation & Flow

Step 21 – Check load Init File, click on edit and when prompted check create



Re-customize IP

Block Memory Generator (8.4)

Documentation IP Location

IP Symbol | Power Estimation

□ Show disabled ports

Component Name axi_bram_ctrl_0_bram

Basic | Port A Options | Other Options | Summary

Pipeline Stages within Mux 0 Mux Size: 1x1

Memory Initialization

☑ Load Init File

Coe File no_coe_file_loaded Browse Edit

□ Fill Remaining Memory Locations

Remaining Memory Locations (Hex) 0

Structural/UniSim Simulation Model Options

Defines the type of warnings and outputs are generated when a read-write or write-write collision occurs.

Collision Warnings All

Behavioral Simulation Model Options

□ Disable Collision Warnings □ Disable Out of Range Warnings

Dynamic Power Saving

□ Sleep

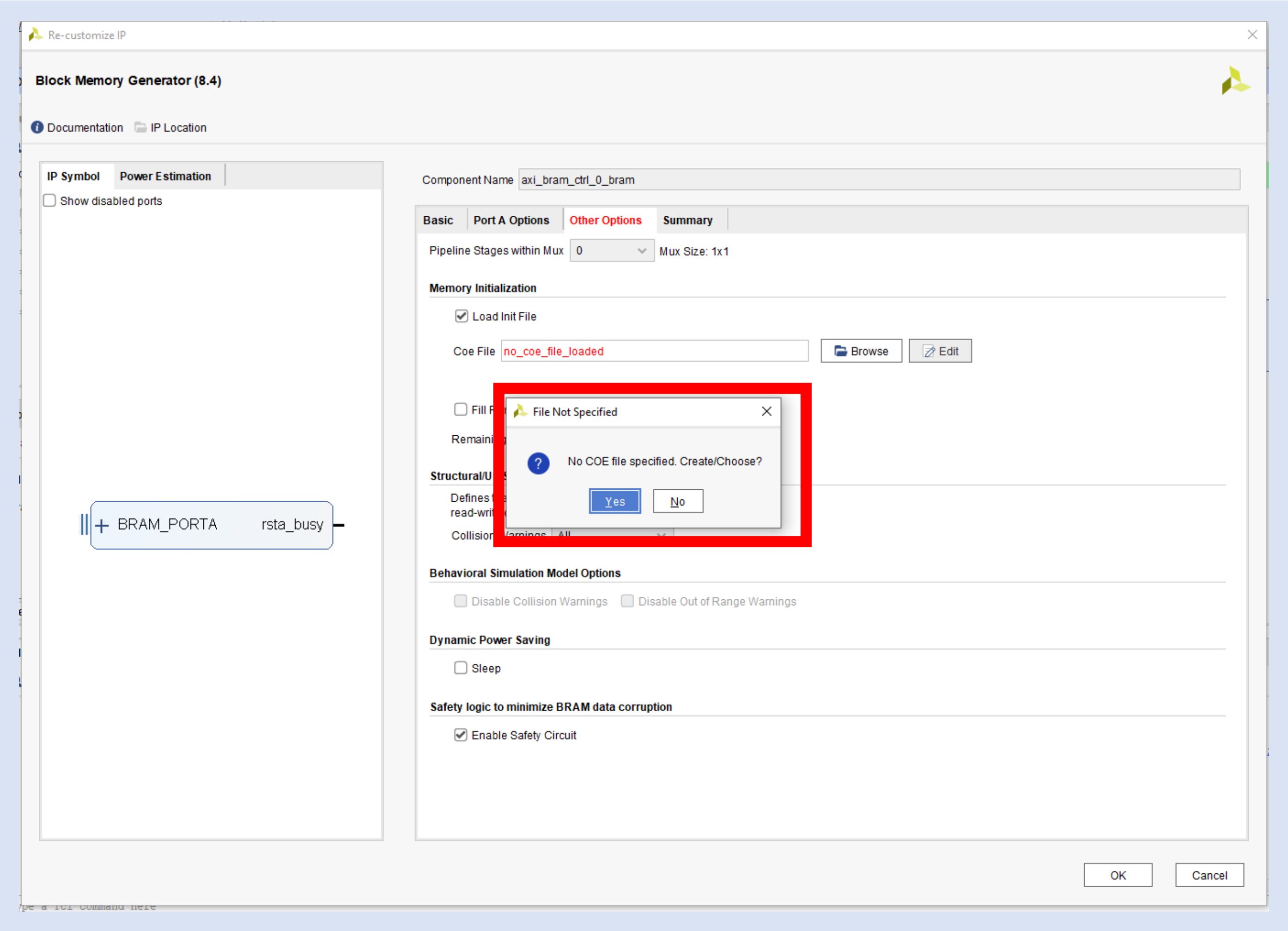
Safety logic to minimize BRAM data corruption

☑ Enable Safety Circuit

OK Cancel

Lab 1: Understanding Vitis Project creation & Flow

Step 22 – Click Yes



Re-customize IP

Block Memory Generator (8.4)

Documentation | IP Location

IP Symbol | Power Estimation

☐ Show disabled ports

Component Name axi_bram_ctrl_0_bram

Basic | Port A Options | Other Options | Summary

Pipeline Stages within Mux 0 Mux Size: 1x1

Memory Initialization

☑ Load Init File

Coe File no_coe_file_loaded Browse Edit

☐ Fill F

Remaini

Structural/U

Defines t
read-writ

Collision Warnings All

File Not Specified

No COE file specified. Create/Choose?

Yes No

Behavioral Simulation Model Options

☐ Disable Collision Warnings ☐ Disable Out of Range Warnings

Dynamic Power Saving

☐ Sleep

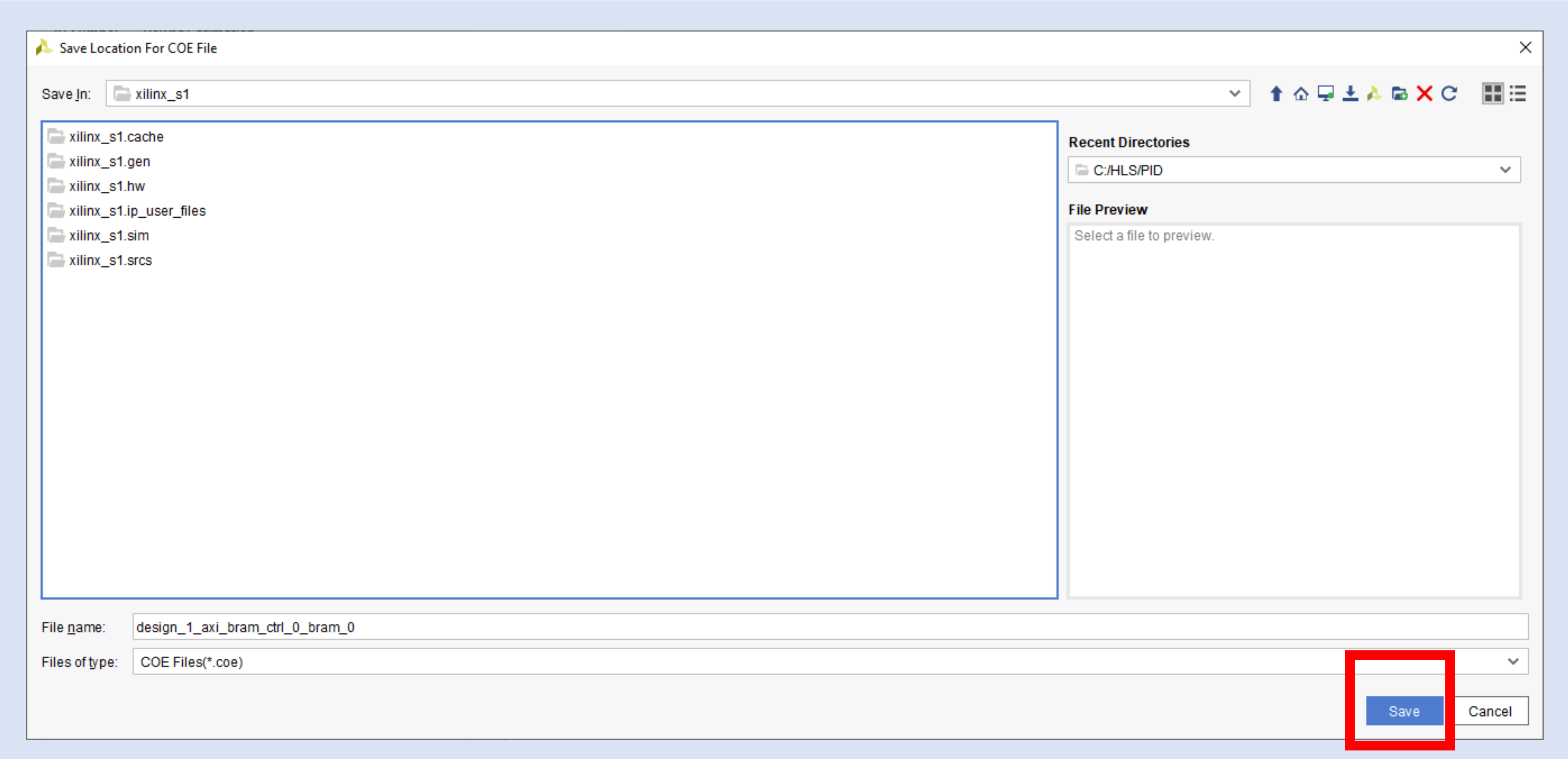
Safety logic to minimize BRAM data corruption

☑ Enable Safety Circuit

OK Cancel

Lab 1: Understanding Vitis Project creation & Flow

Step 23 – Select a location inside the project to save the file and select OK



Save Location For COE File

Save In: xilinx_s1

- xilinx_s1.cache
- xilinx_s1.gen
- xilinx_s1.hw
- xilinx_s1.ip_user_files
- xilinx_s1.sim
- xilinx_s1.srcs

Recent Directories: C:/HLS/PID

File Preview: Select a file to preview.

File name: design_1_axi_bram_ctrl_0_bram_0

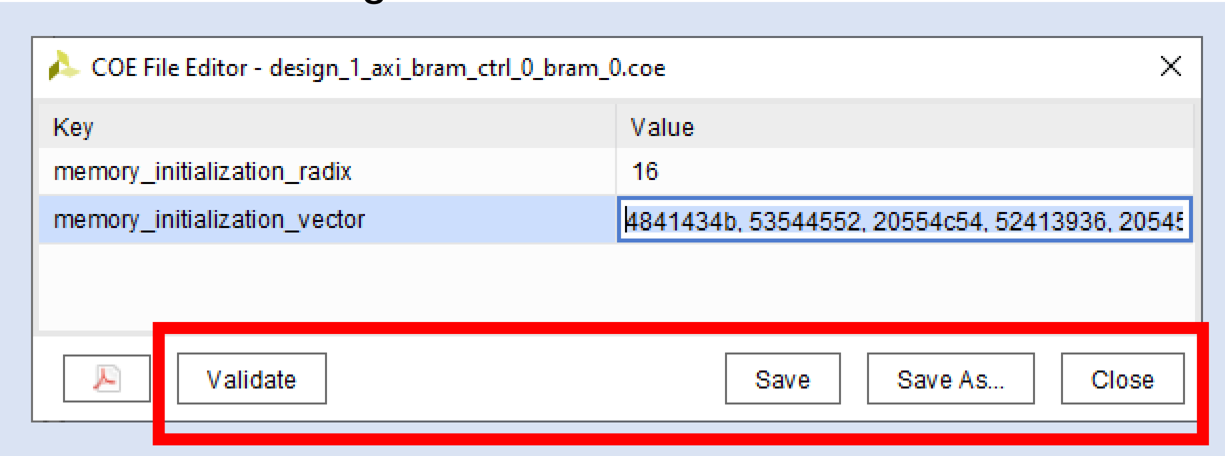
Files of type: COE Files(*.coe)

Lab 1: Understanding Vitis Project creation & Flow

Step 24 – Enter the following

Memory Initialization Radix = 16
Memory Initialization Vector = 4841434b, 53544552, 20554c54, 52413936, 20545554, 4f524941, 0000004c,

Click Save, then Validate before closing

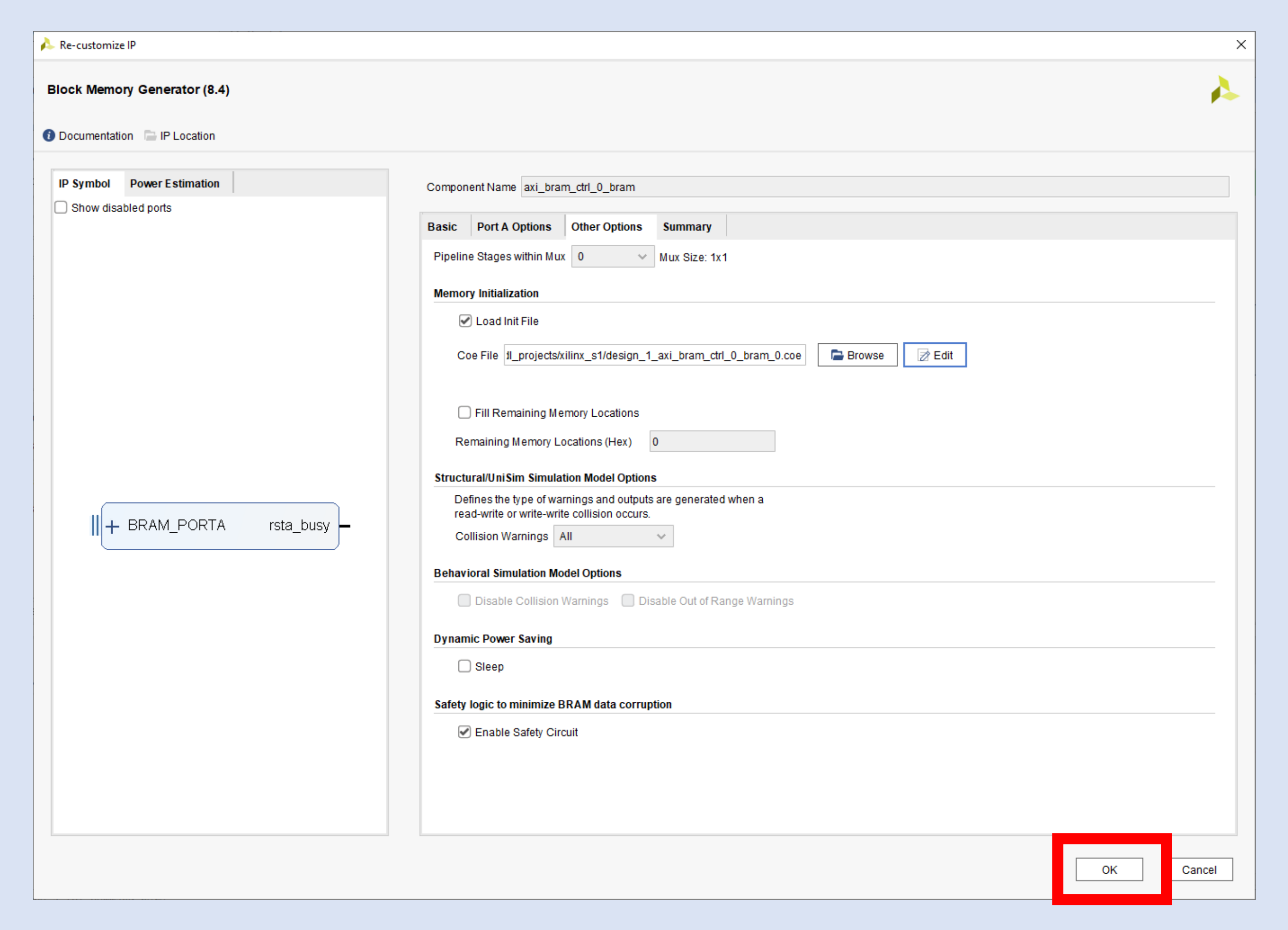


COE File Editor - design_1_axi_bram_ctrl_0_bram_0.coe

Key	Value
memory_initialization_radix	16
memory_initialization_vector	4841434b, 53544552, 20554c54, 52413936, 2054

Lab 1: Understanding Vitis Project creation & Flow

Step 25 – Click OK



Re-customize IP

Block Memory Generator (8.4)

Documentation IP Location

IP Symbol | Power Estimation

□ Show disabled ports

Component Name: axi_bram_ctrl_0_bram

Basic | Port A Options | Other Options | Summary

Pipeline Stages within Mux: 0 Mux Size: 1x1

Memory Initialization

☑ Load Init File

Coe File: ıl_projects/xilinx_s1/design_1_axi_bram_ctrl_0_bram_0.coe Browse Edit

□ Fill Remaining Memory Locations

Remaining Memory Locations (Hex): 0

Structural/UniSim Simulation Model Options

Defines the type of warnings and outputs are generated when a read-write or write-write collision occurs.

Collision Warnings: All

Behavioral Simulation Model Options

□ Disable Collision Warnings □ Disable Out of Range Warnings

Dynamic Power Saving

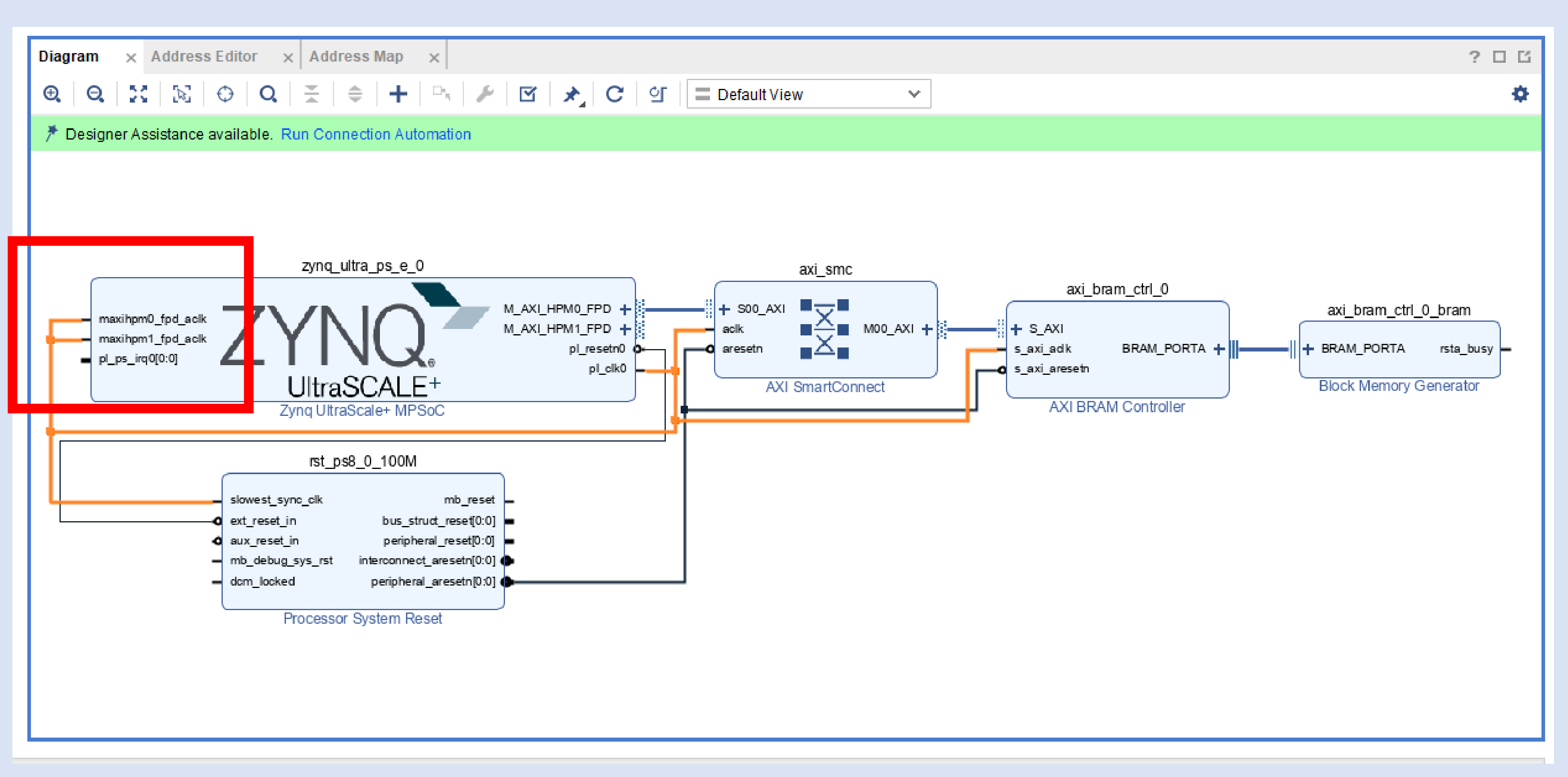
□ Sleep

Safety logic to minimize BRAM data corruption

☑ Enable Safety Circuit

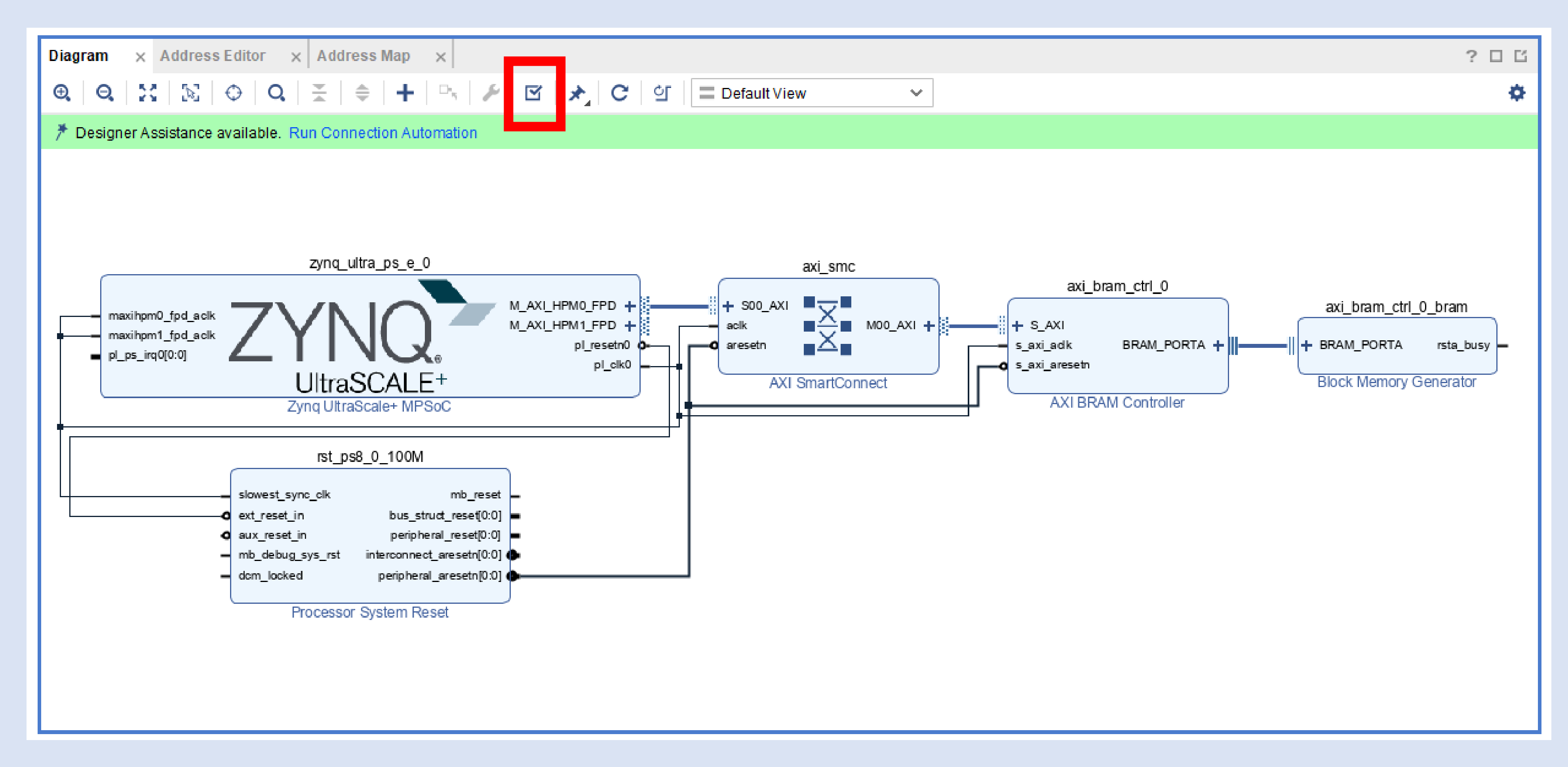
Lab 1: Understanding Vitis Project creation & Flow

Step 26 – Connect MAXIHPM0_FPD_ACLK to MAXIHPM1_FPD_ACLK



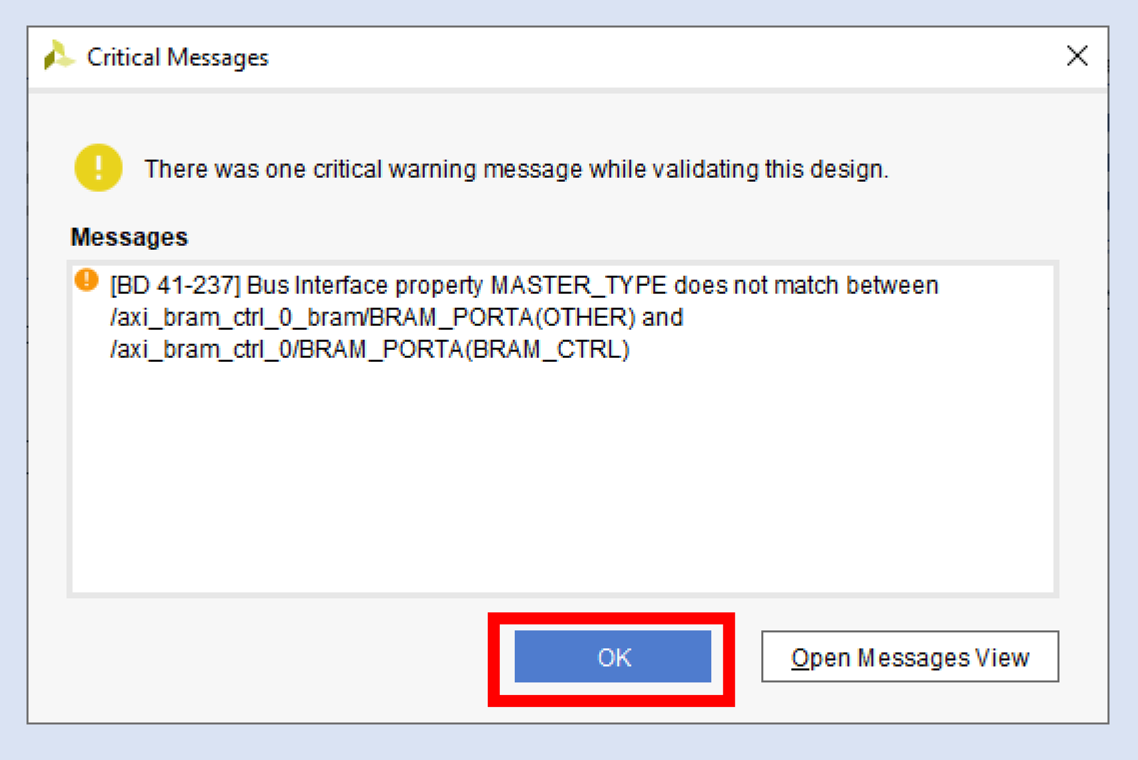
Lab 1: Understanding Vitis Project creation & Flow

Step 27 – Click on Validate the design



Lab 1: Understanding Vitis Project creation & Flow

Step 28 – Click On OK the warning is due to the change from AXI BRAM control



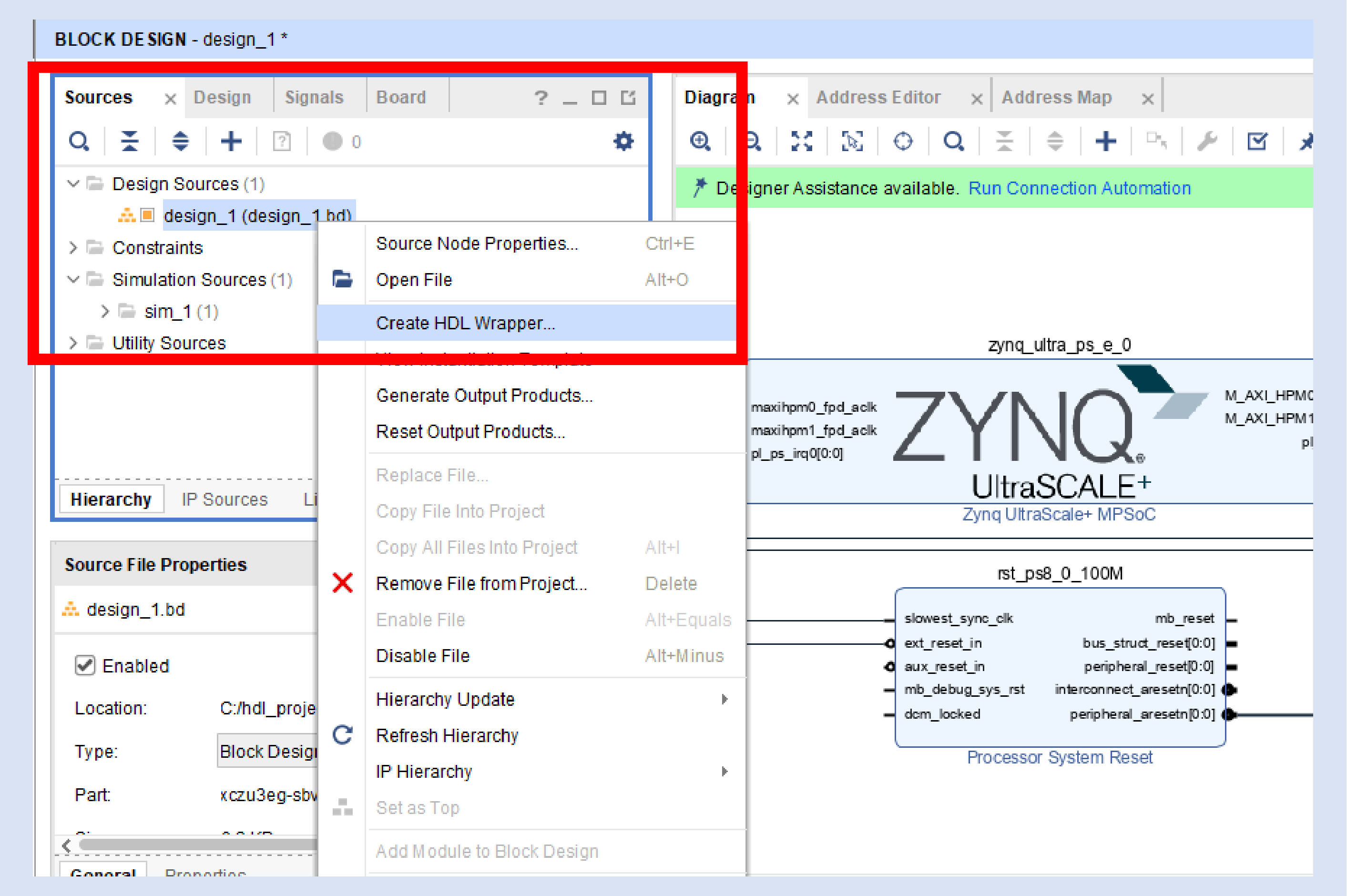
There was one critical warning message while validating this design.

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 1: Understanding Vitis Project creation & Flow

Step 29 – In the sources tab, right click on the block diagram and select create HDL Wrapper



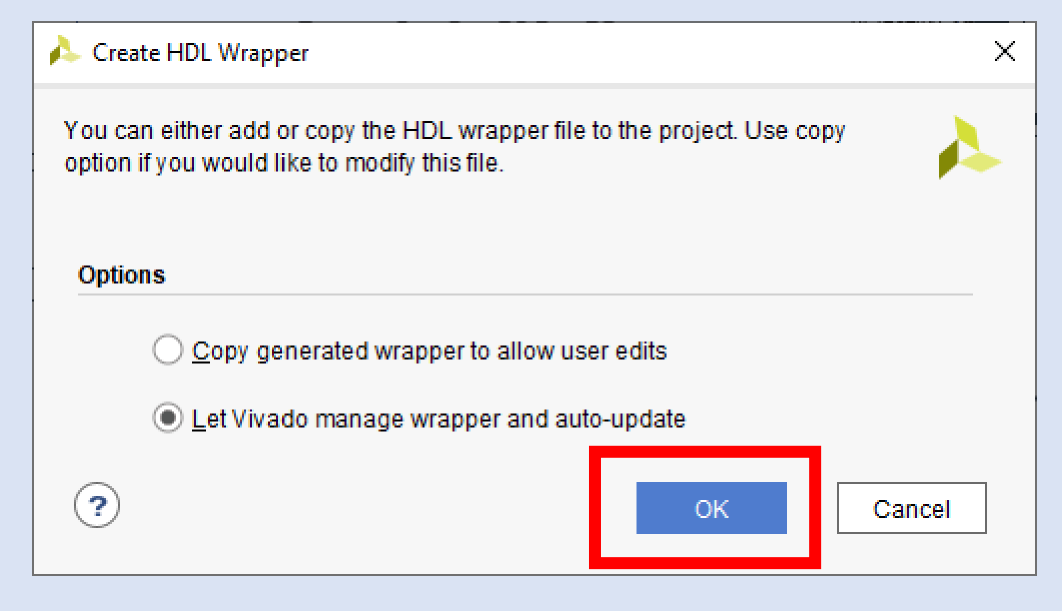
Location: C:/hdl_proje

Type: Block Desig

Part: xczu3eg-sbv

Lab 1: Understanding Vitis Project creation & Flow

Step 30 – leave the options as set and let Vivado manage the wrapper, click OK



Create HDL Wrapper

You can either add or copy the HDL wrapper file to the project. Use copy option if you would like to modify this file.

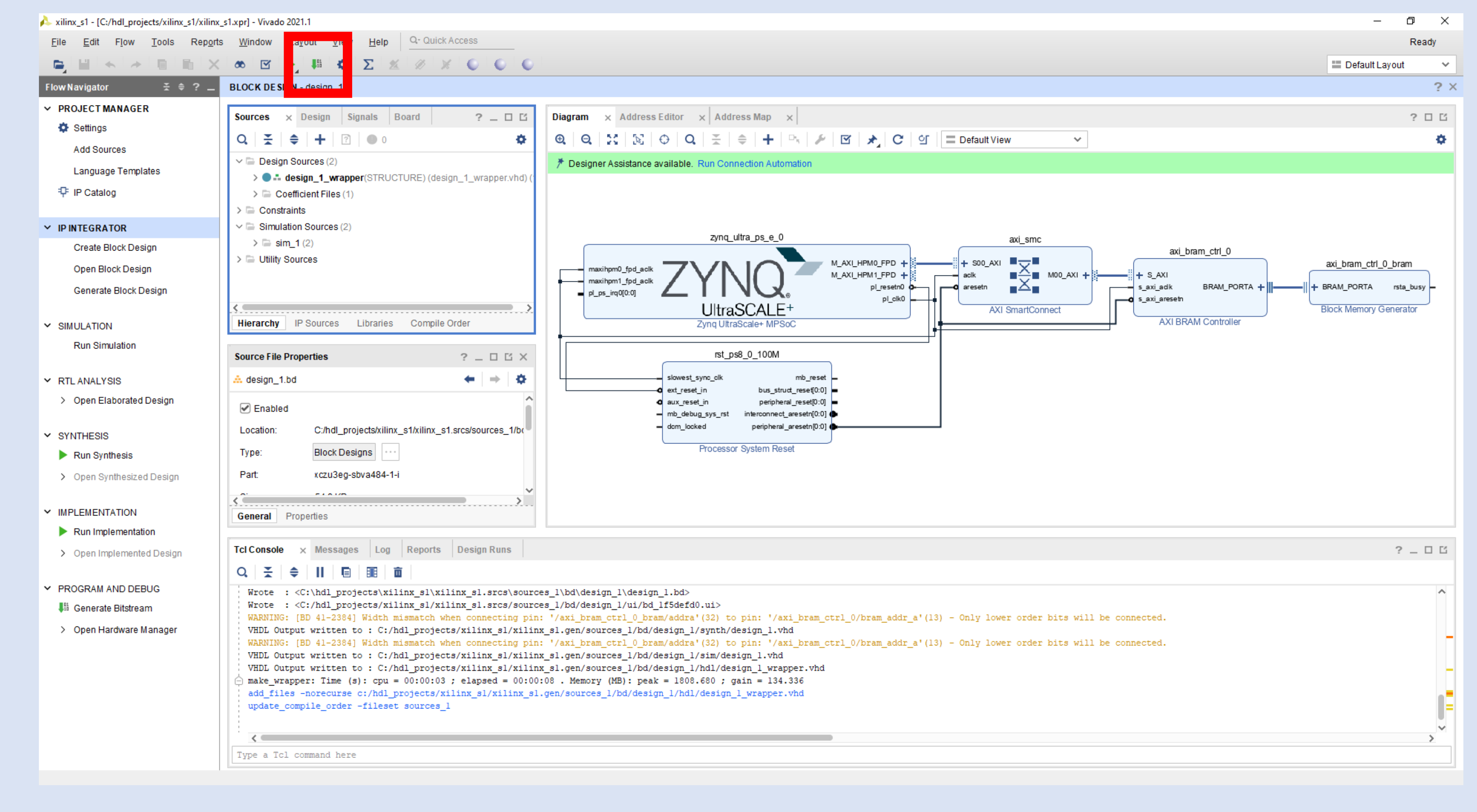
Options

○ Copy generated wrapper to allow user edits

● Let Vivado manage wrapper and auto-update

Lab 1: Understanding Vitis Project creation & Flow

Step 31 – Click on Generate Bit Stream

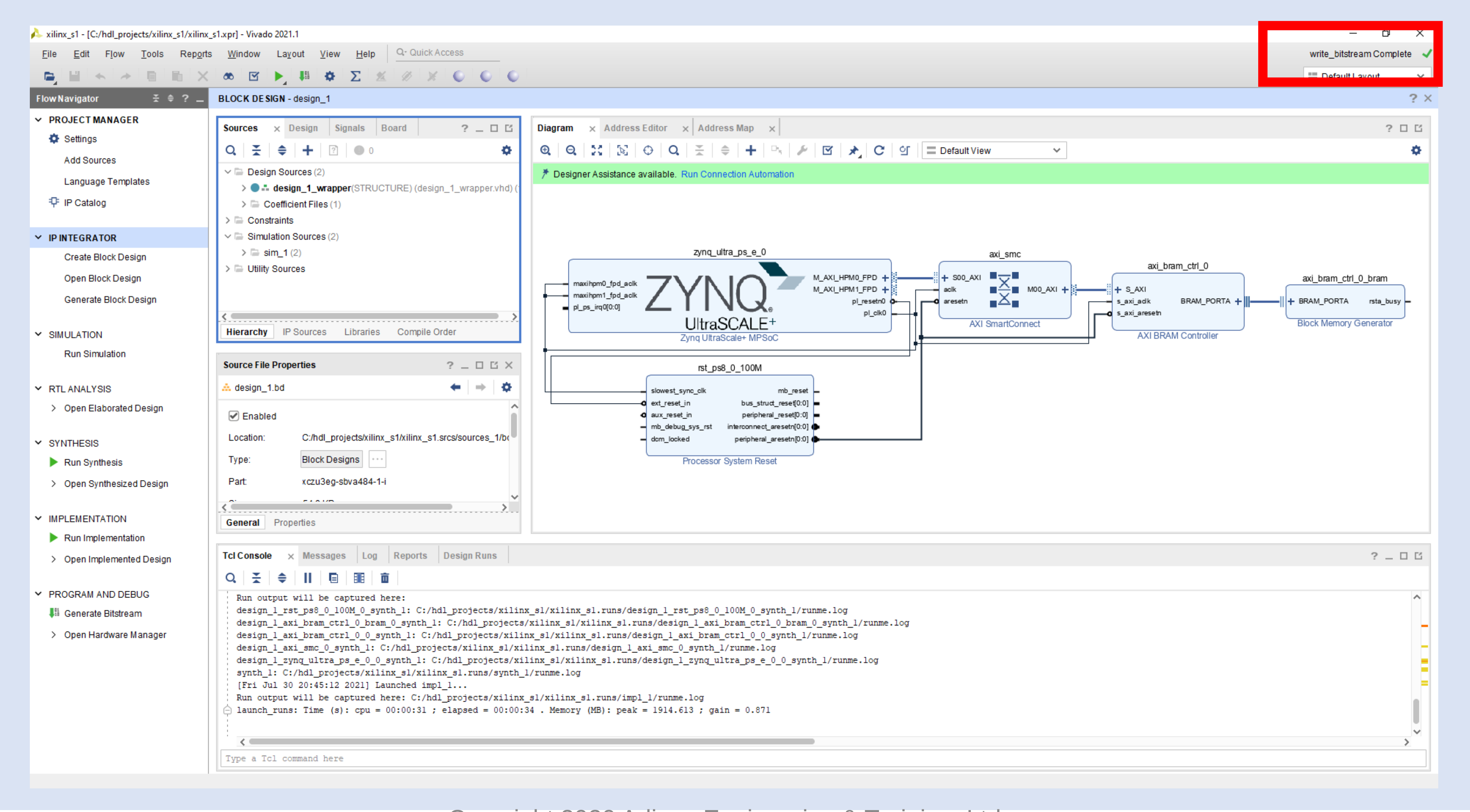


```
Wrote  : <C:\hdl_projects\xilinx_s1\xilinx_s1.srcs\sources_1\bd\design_1\design_1.bd>
Wrote  : <C:/hdl_projects/xilinx_s1/xilinx_s1.srcs/sources_1/bd/design_1/ui/bd_1f5defd0.ui>
WARNING: [BD 41-2384] Width mismatch when connecting pin: '/axi_bram_ctrl_0_bram/addra'(32) to pin: '/axi_bram_ctrl_0/bram_addr_a'(13) - Only lower order bits will be connected.
VHDL Output written to : C:/hdl_projects/xilinx_s1/xilinx_s1.gen/sources_1/bd/design_1/synth/design_1.vhd
WARNING: [BD 41-2384] Width mismatch when connecting pin: '/axi_bram_ctrl_0_bram/addra'(32) to pin: '/axi_bram_ctrl_0/bram_addr_a'(13) - Only lower order bits will be connected.
VHDL Output written to : C:/hdl_projects/xilinx_s1/xilinx_s1.gen/sources_1/bd/design_1/sim/design_1.vhd
VHDL Output written to : C:/hdl_projects/xilinx_s1/xilinx_s1.gen/sources_1/bd/design_1/hdl/design_1_wrapper.vhd
make_wrapper: Time (s): cpu = 00:00:03 ; elapsed = 00:00:08 . Memory (MB): peak = 1808.680 ; gain = 134.336
add_files -norecurse c:/hdl_projects/xilinx_s1/xilinx_s1.gen/sources_1/bd/design_1/hdl/design_1_wrapper.vhd
update_compile_order -fileset sources_1
```

Type a Tcl command here

Lab 1: Understanding Vitis Project creation & Flow

Step 32 – Wait until the bit stream is complete



xilinx_s1 - [C:/hdl_projects/xilinx_s1/xilinx_s1.xpr] - Vivado 2021.1

write_bitstream Complete ✓

Designer Assistance available. Run Connection Automation

Source File Properties

design_1.bd

☑ Enabled

Location: C:/hdl_projects/xilinx_s1/xilinx_s1.srcs/sources_1/bd

Type: Block Designs

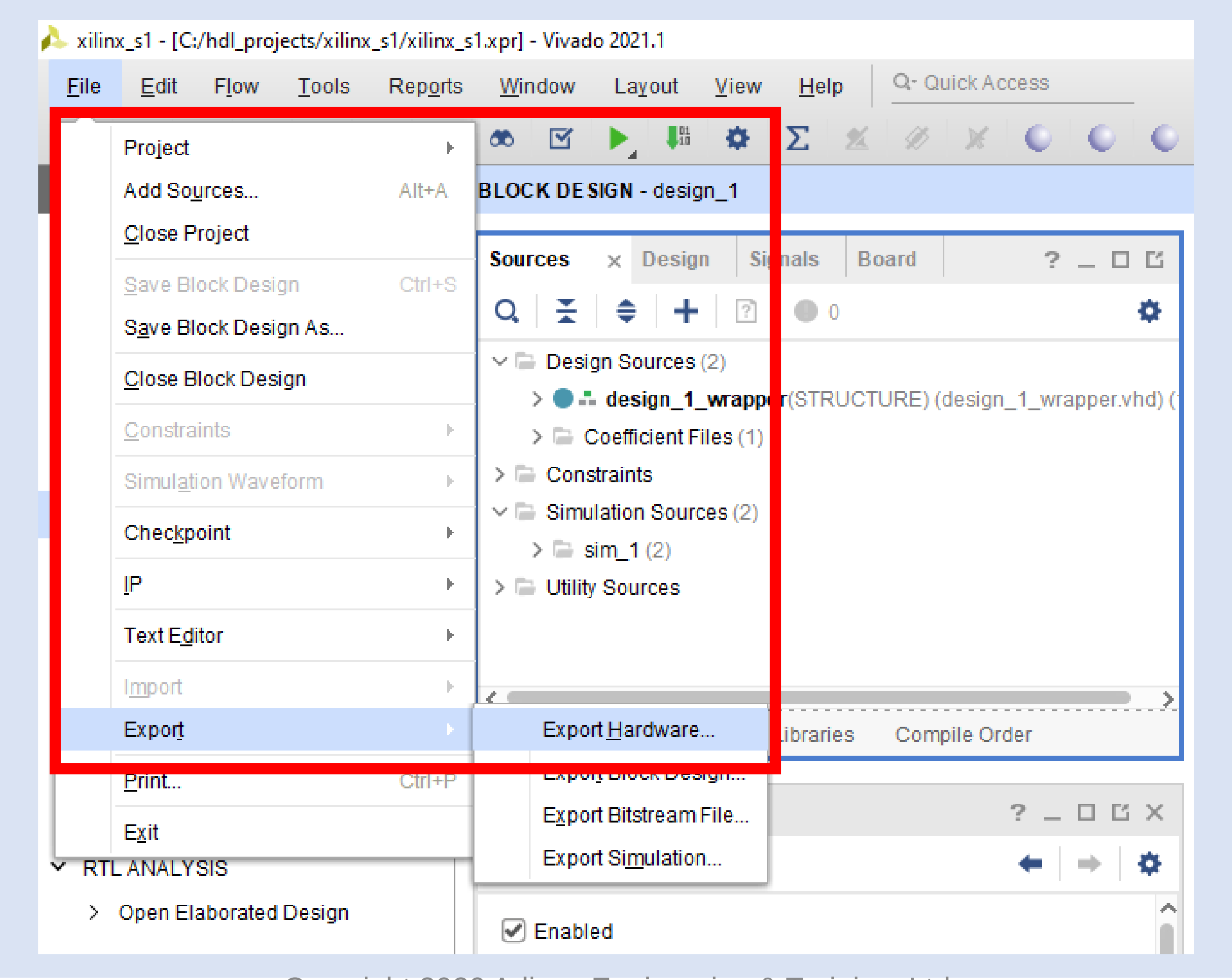
Part: xczu3eg-sbva484-1-i

```
Run output will be captured here:
design_1_rst_ps8_0_100M_0_synth_1: C:/hdl_projects/xilinx_s1/xilinx_s1.runs/design_1_rst_ps8_0_100M_0_synth_1/runme.log
design_1_axi_bram_ctrl_0_bram_0_synth_1: C:/hdl_projects/xilinx_s1/xilinx_s1.runs/design_1_axi_bram_ctrl_0_bram_0_synth_1/runme.log
design_1_axi_bram_ctrl_0_0_synth_1: C:/hdl_projects/xilinx_s1/xilinx_s1.runs/design_1_axi_bram_ctrl_0_0_synth_1/runme.log
design_1_axi_smc_0_synth_1: C:/hdl_projects/xilinx_s1/xilinx_s1.runs/design_1_axi_smc_0_synth_1/runme.log
design_1_zynq_ultra_ps_e_0_0_synth_1: C:/hdl_projects/xilinx_s1/xilinx_s1.runs/design_1_zynq_ultra_ps_e_0_0_synth_1/runme.log
synth_1: C:/hdl_projects/xilinx_s1/xilinx_s1.runs/synth_1/runme.log
[Fri Jul 30 20:45:12 2021] Launched impl_1...
Run output will be captured here: C:/hdl_projects/xilinx_s1/xilinx_s1.runs/impl_1/runme.log
launch_runs: Time (s): cpu = 00:00:31 ; elapsed = 00:00:34 . Memory (MB): peak = 1914.613 ; gain = 0.871
```

Type a Tcl command here

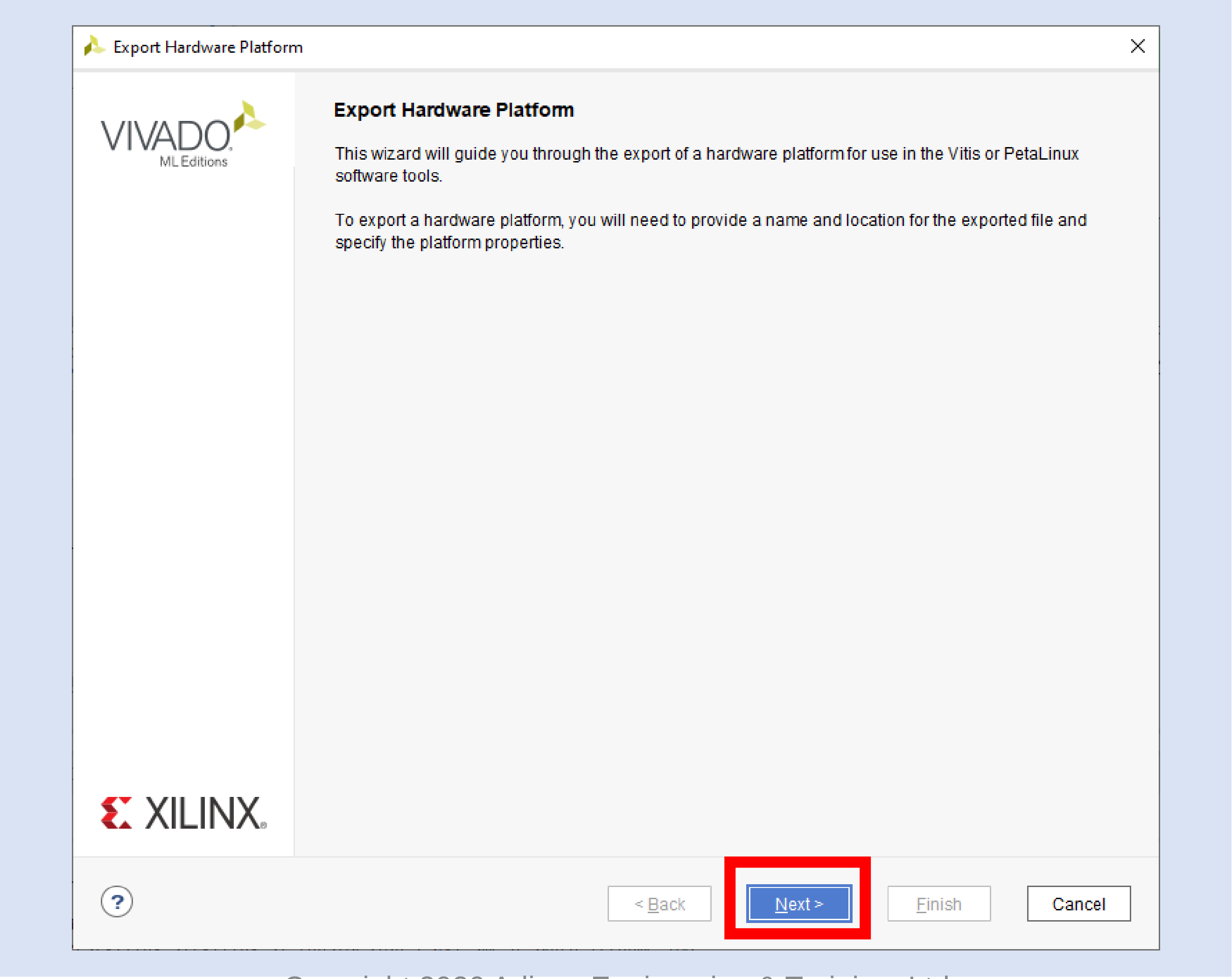
Lab 1: Understanding Vitis Project creation & Flow

Step 33 – From the File Menu select Export->Export Hardware



Lab 1: Understanding Vitis Project creation & Flow

Step 34 – Click on Next



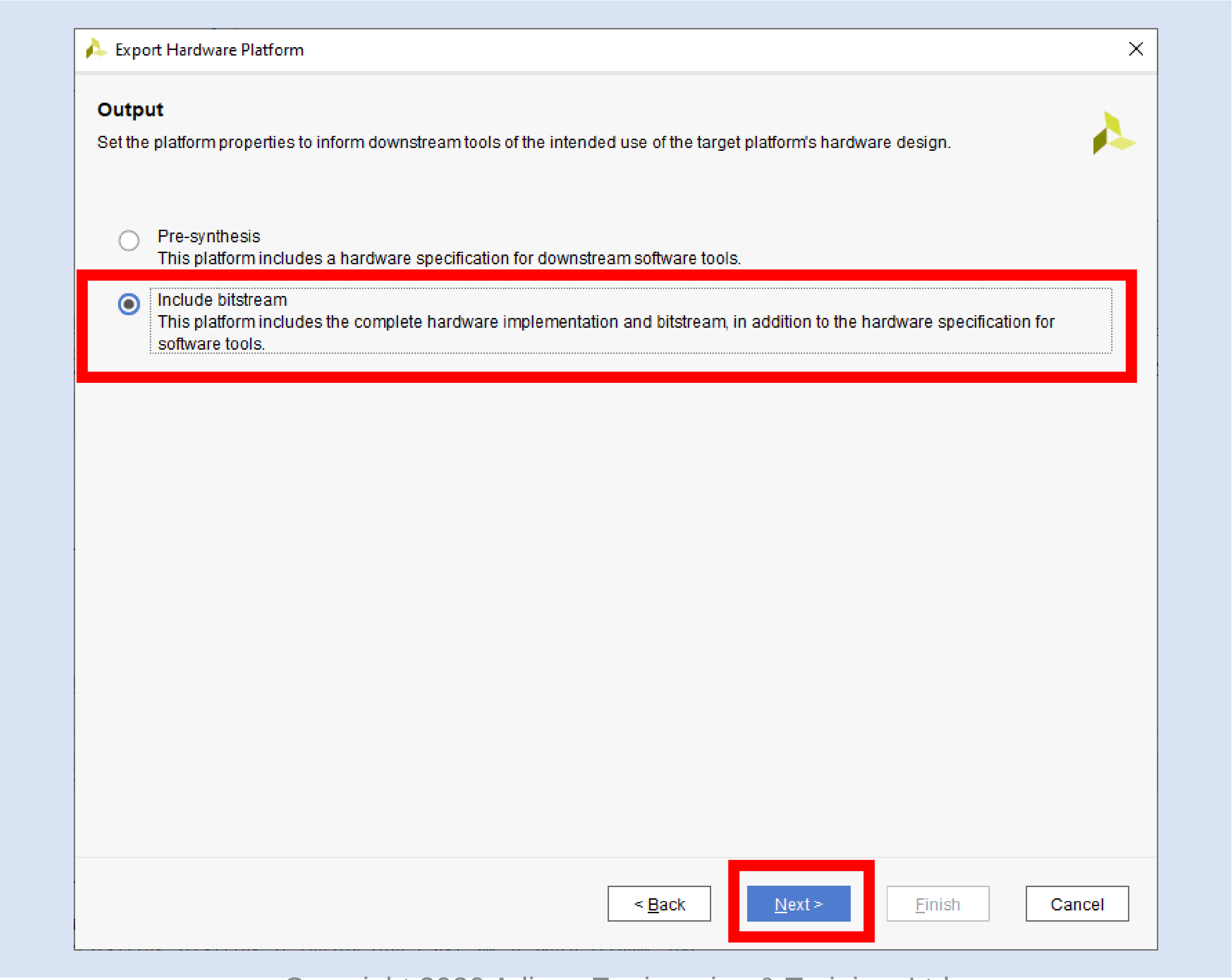
Export Hardware Platform

This wizard will guide you through the export of a hardware platform for use in the Vitis or PetaLinux software tools.

To export a hardware platform, you will need to provide a name and location for the exported file and specify the platform properties.

Lab 1: Understanding Vitis Project creation & Flow

Step 35 – Ensure Include Bitstream is selected and click OK



Export Hardware Platform

Output

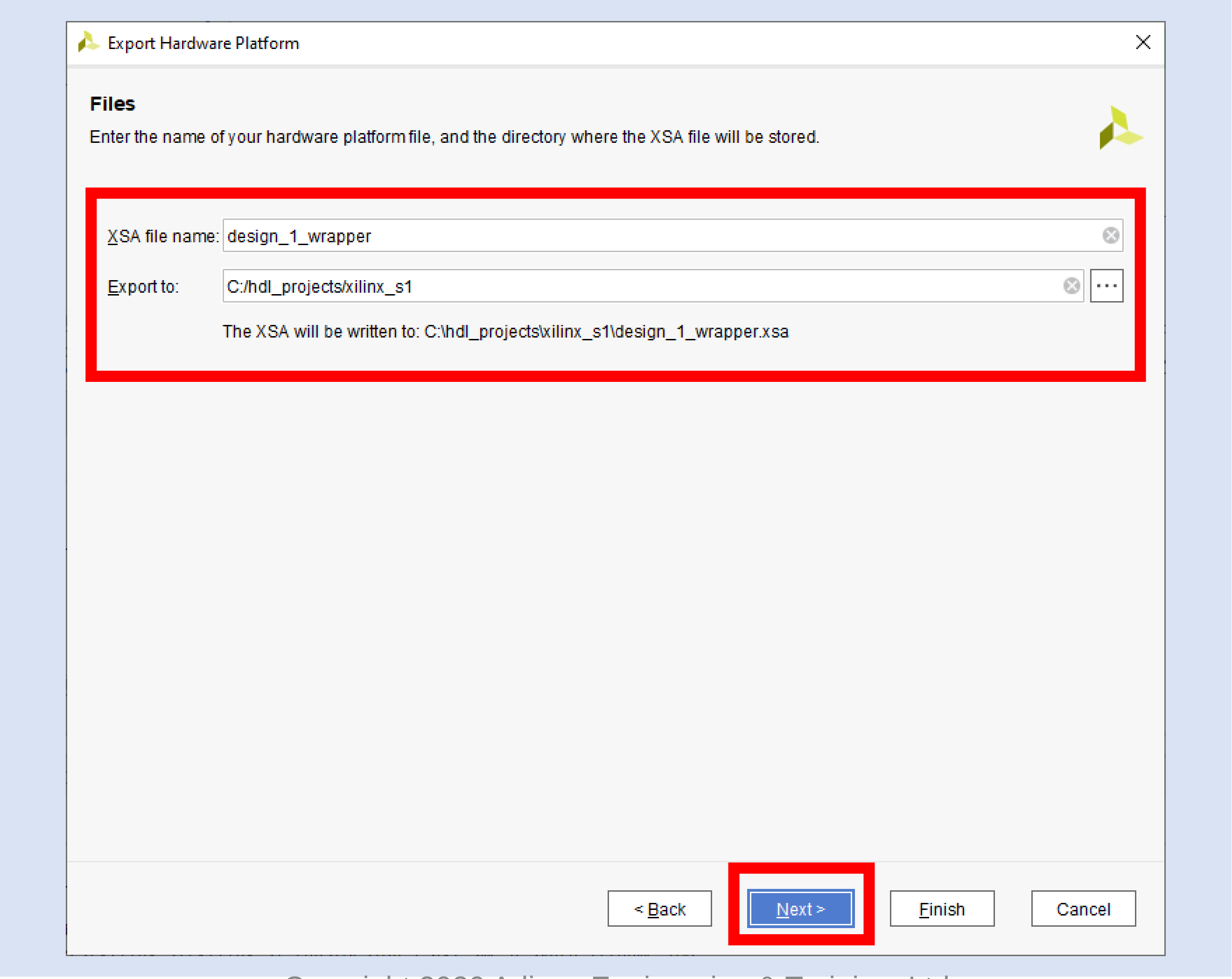
Set the platform properties to inform downstream tools of the intended use of the target platform's hardware design.

○ Pre-synthesis
This platform includes a hardware specification for downstream software tools.

● Include bitstream
This platform includes the complete hardware implementation and bitstream, in addition to the hardware specification for software tools.

Lab 1: Understanding Vitis Project creation & Flow

Step 36 – Leave the defaults as is and click on Next



Files

Enter the name of your hardware platform file, and the directory where the XSA file will be stored.

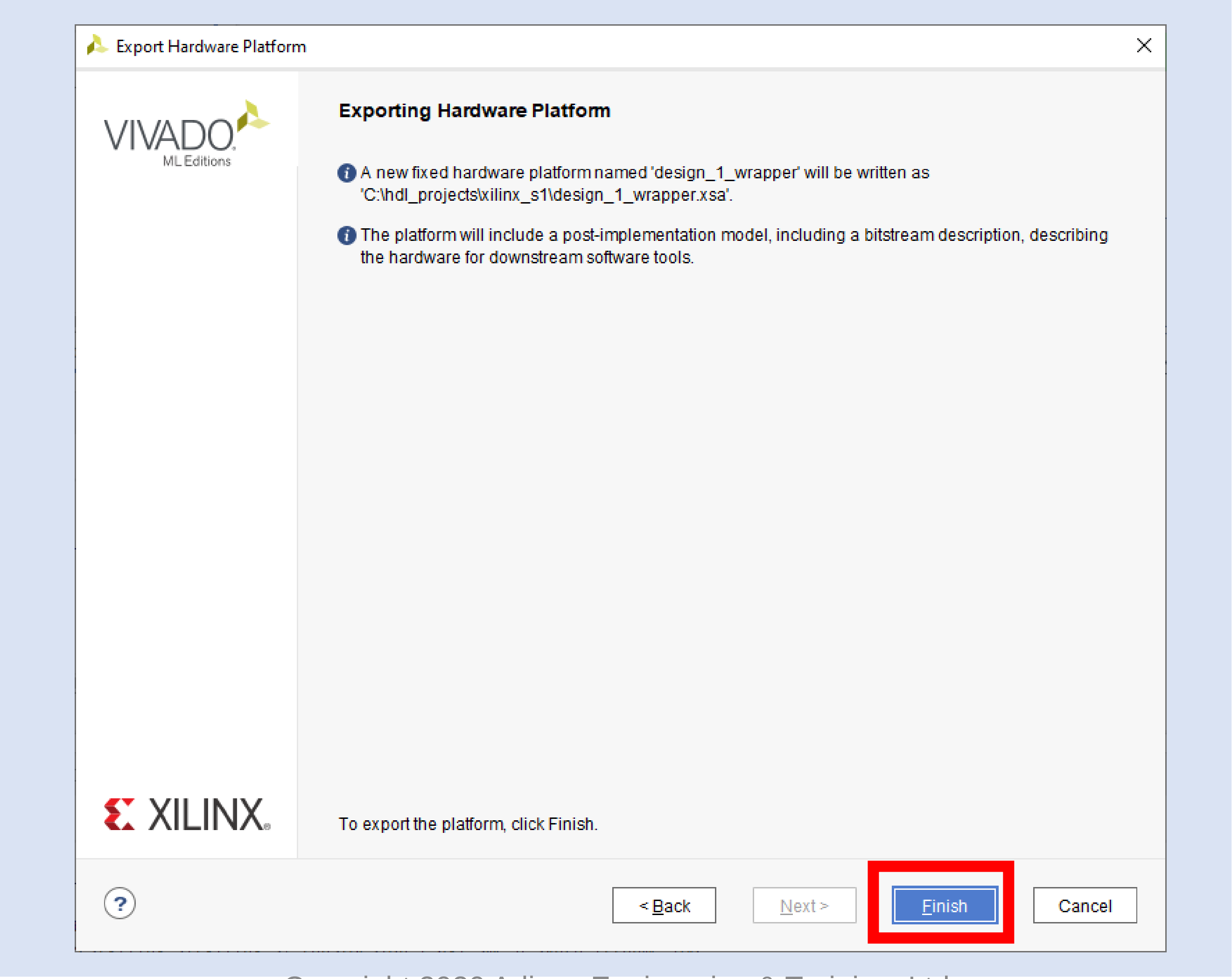
XSA file name: design_1_wrapper

Export to: C:/hdl_projects/xilinx_s1

The XSA will be written to: C:\hdl_projects\xilinx_s1\design_1_wrapper.xsa

Lab 1: Understanding Vitis Project creation & Flow

Step 37 – Click Finish



Exporting Hardware Platform

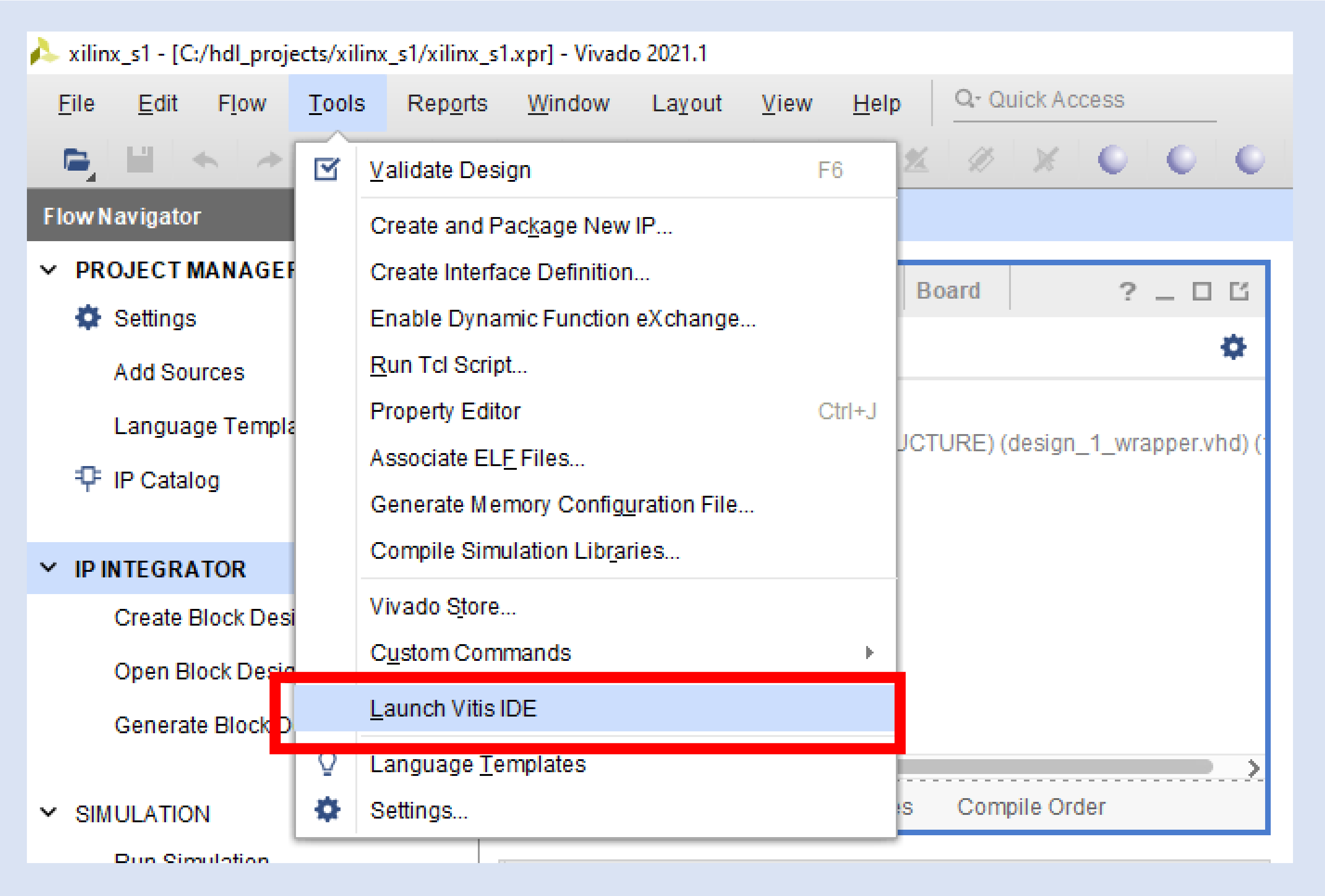
A new fixed hardware platform named 'design_1_wrapper' will be written as 'C:\hdl_projects\xilinx_s1\design_1_wrapper.xsa'.

The platform will include a post-implementation model, including a bitstream description, describing the hardware for downstream software tools.

To export the platform, click Finish.

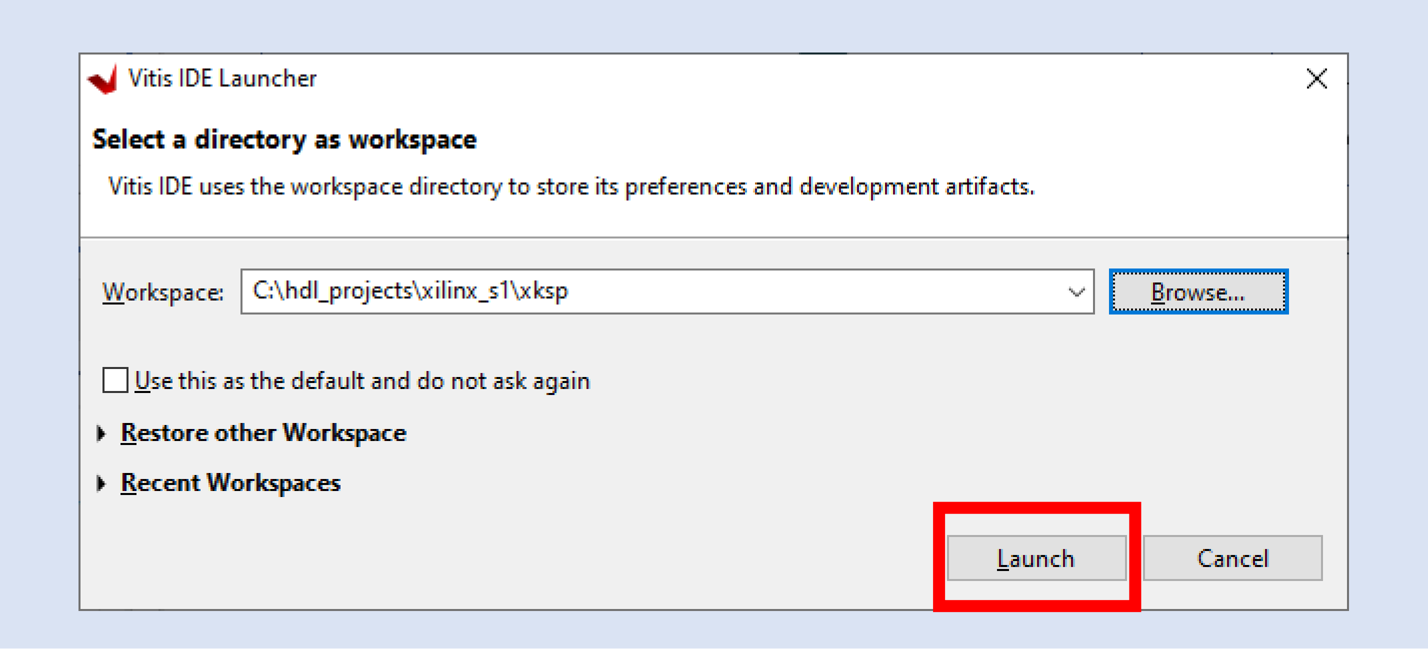
Lab 1: Understanding Vitis Project creation & Flow

Step 38 – From the Tools menu select Launch Vitis IDE



Lab 1: Understanding Vitis Project creation & Flow

Step 39 – At the dialog, create a new folder in your project directory and select launch



Vitis IDE Launcher

Select a directory as workspace

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

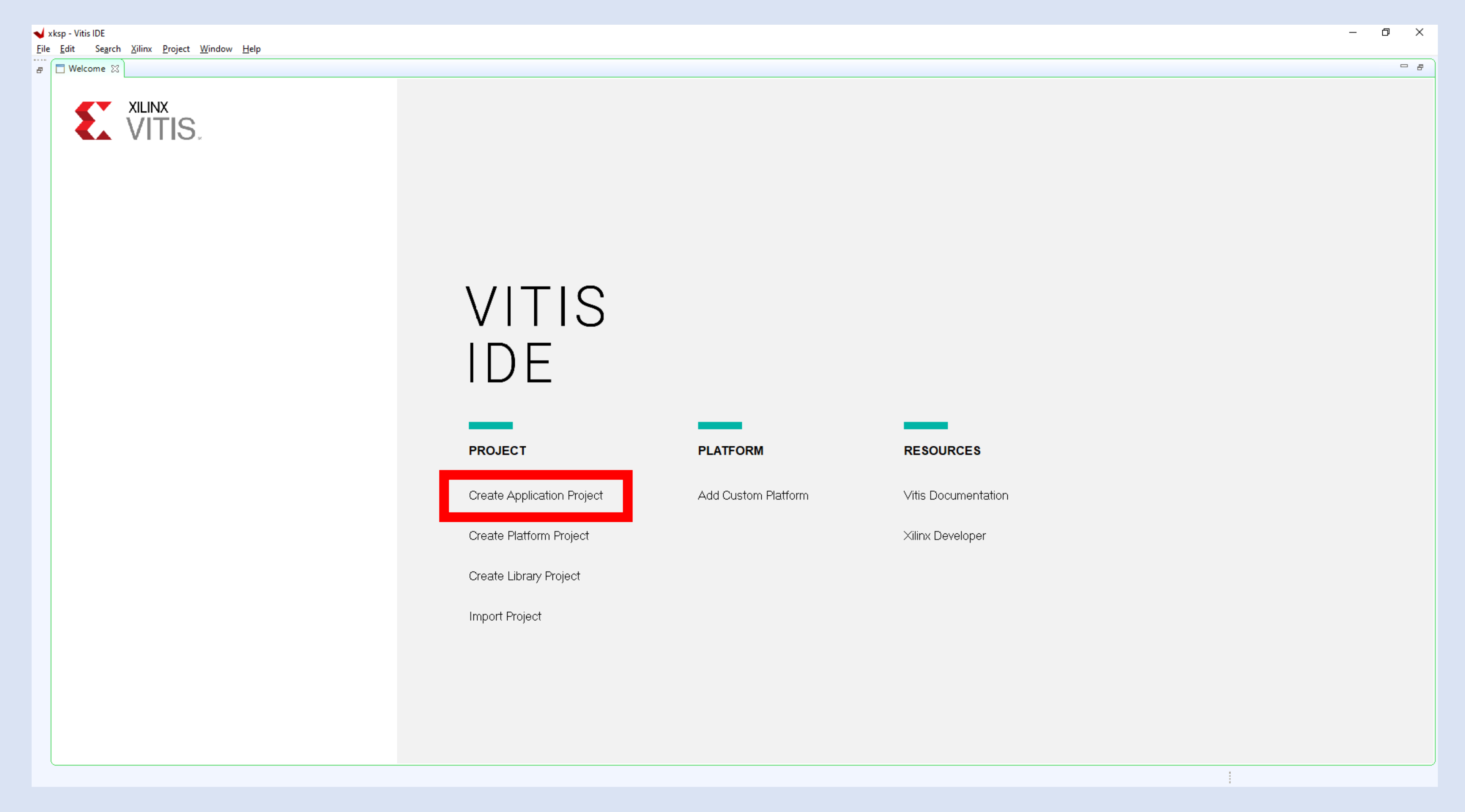
Workspace: C:\hdl_projects\xilinx_s1\xksp

□ Use this as the default and do not ask again

Lab 1

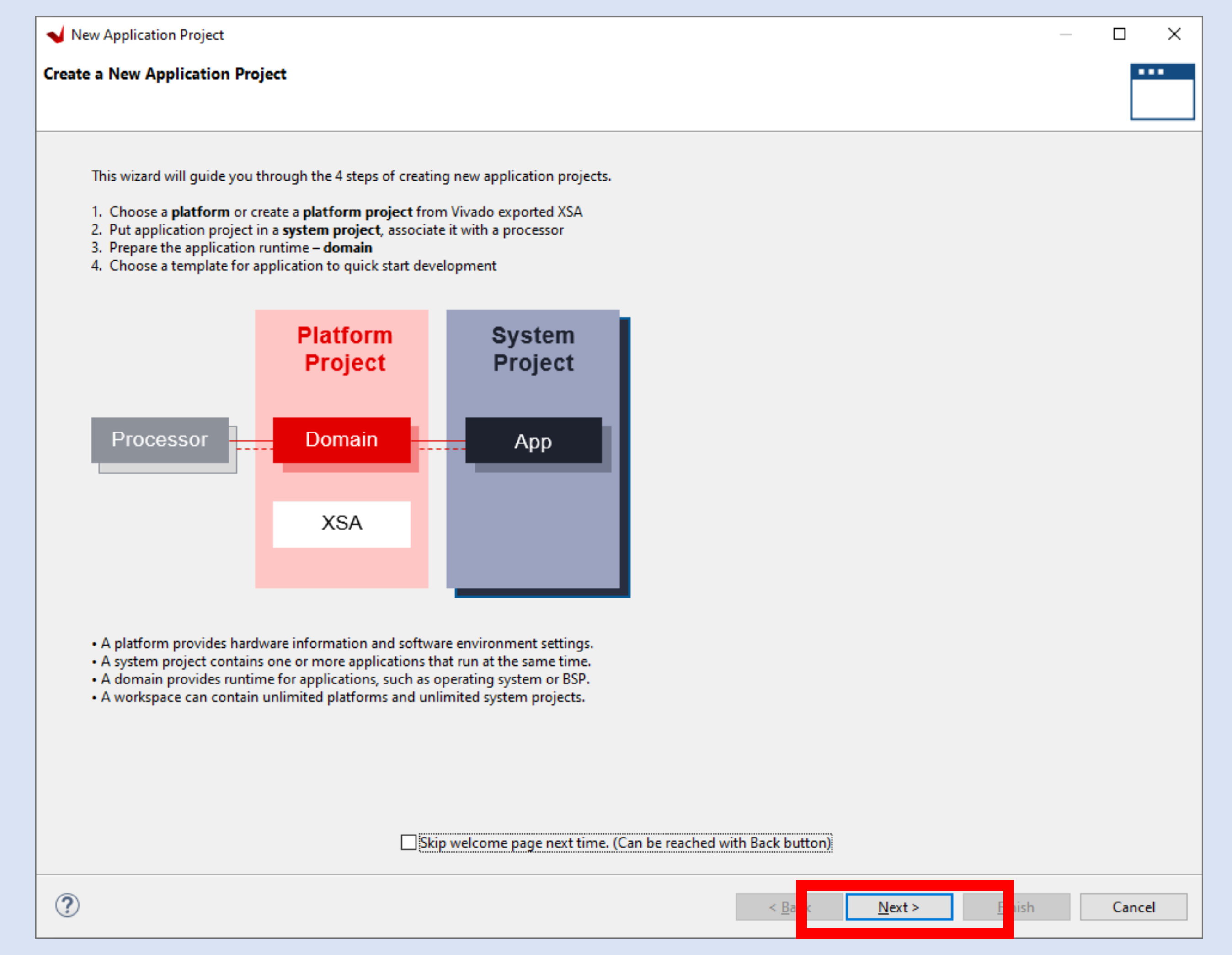
Lab 1: Understanding Vitis Project creation & Flow

Step 40 – Click on Create Application Project



Lab 1: Understanding Vitis Project creation & Flow

Step 41 – Click on next



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

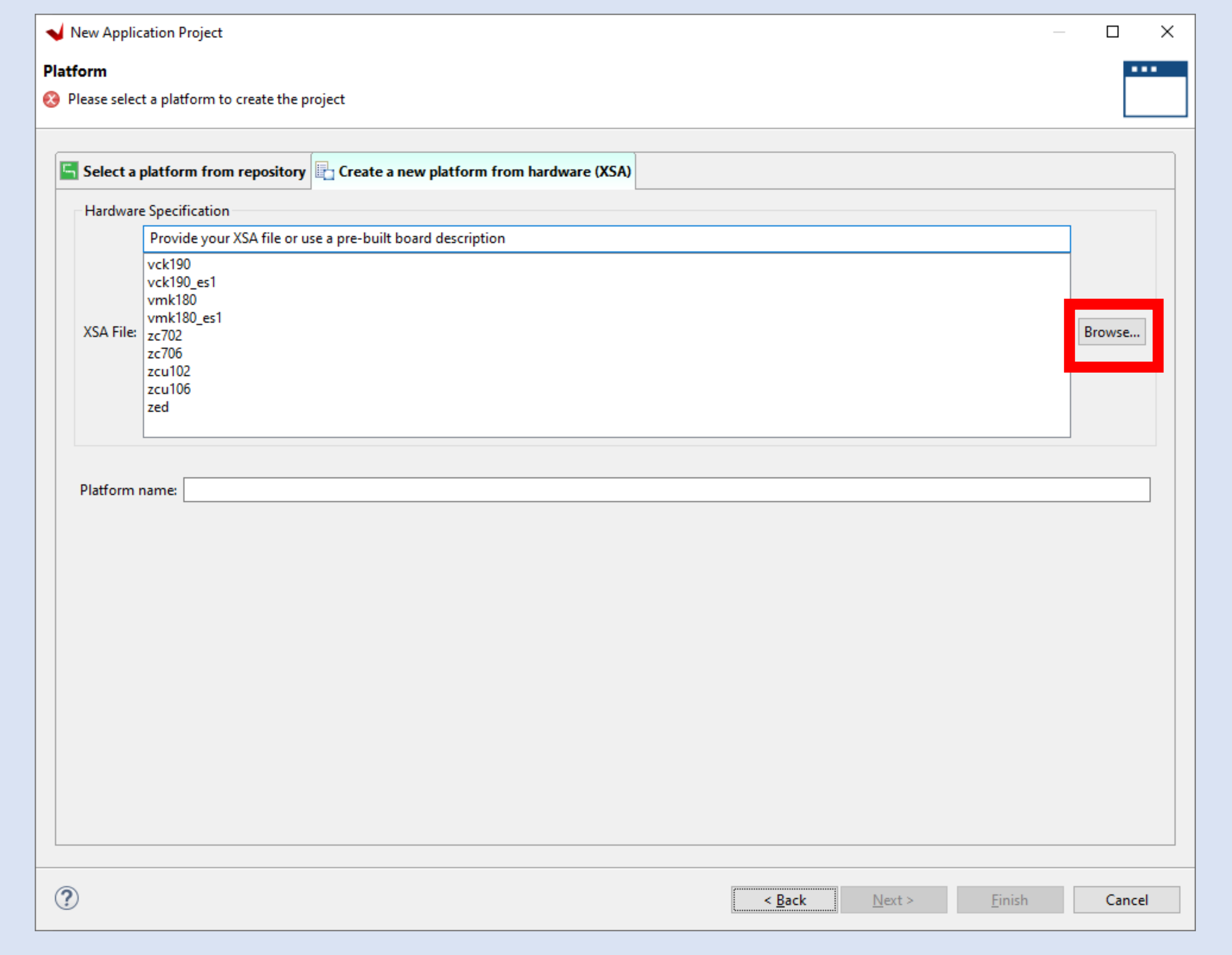
1. Choose a **platform** or create a **platform project** from Vivado exported XSA
2. Put application project in a **system project**, associate it with a processor
3. Prepare the application runtime – **domain**
4. Choose a template for application to quick start development

- 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 1: Understanding Vitis Project creation & Flow

Step 42 – Click on Create a New Platform from Hardware (XSA) and select browse



New Application Project

Platform

Please select a platform to create the project

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

Hardware Specification

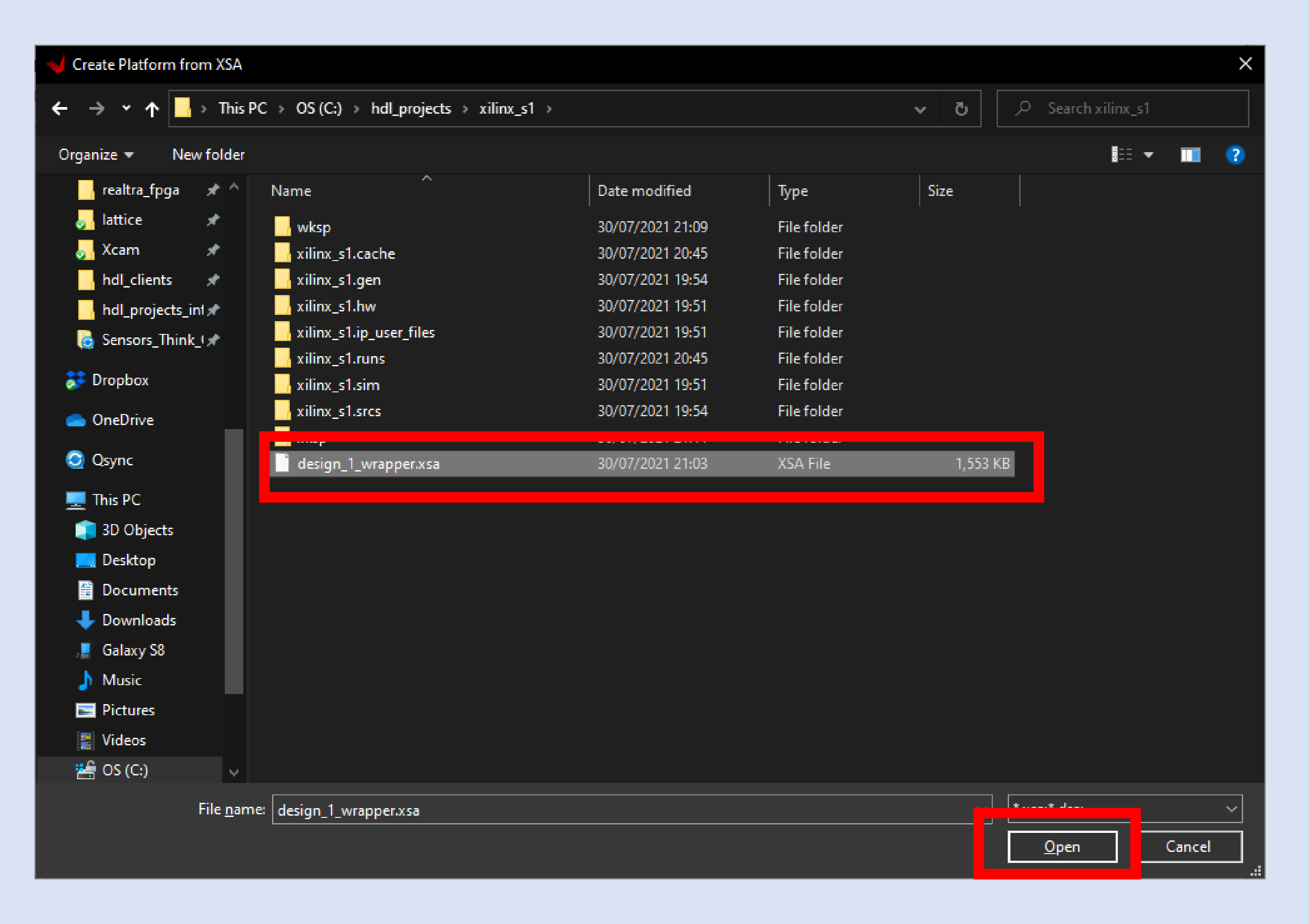
XSA File: Provide your XSA file or use a pre-built board description

vck190
vck190_es1
vmk180
vmk180_es1
zc702
zc706
zcu102
zcu106
zed

Platform name:

Lab 1: Understanding Vitis Project creation & Flow

Step 43 – Select the exported XSA and Click Open

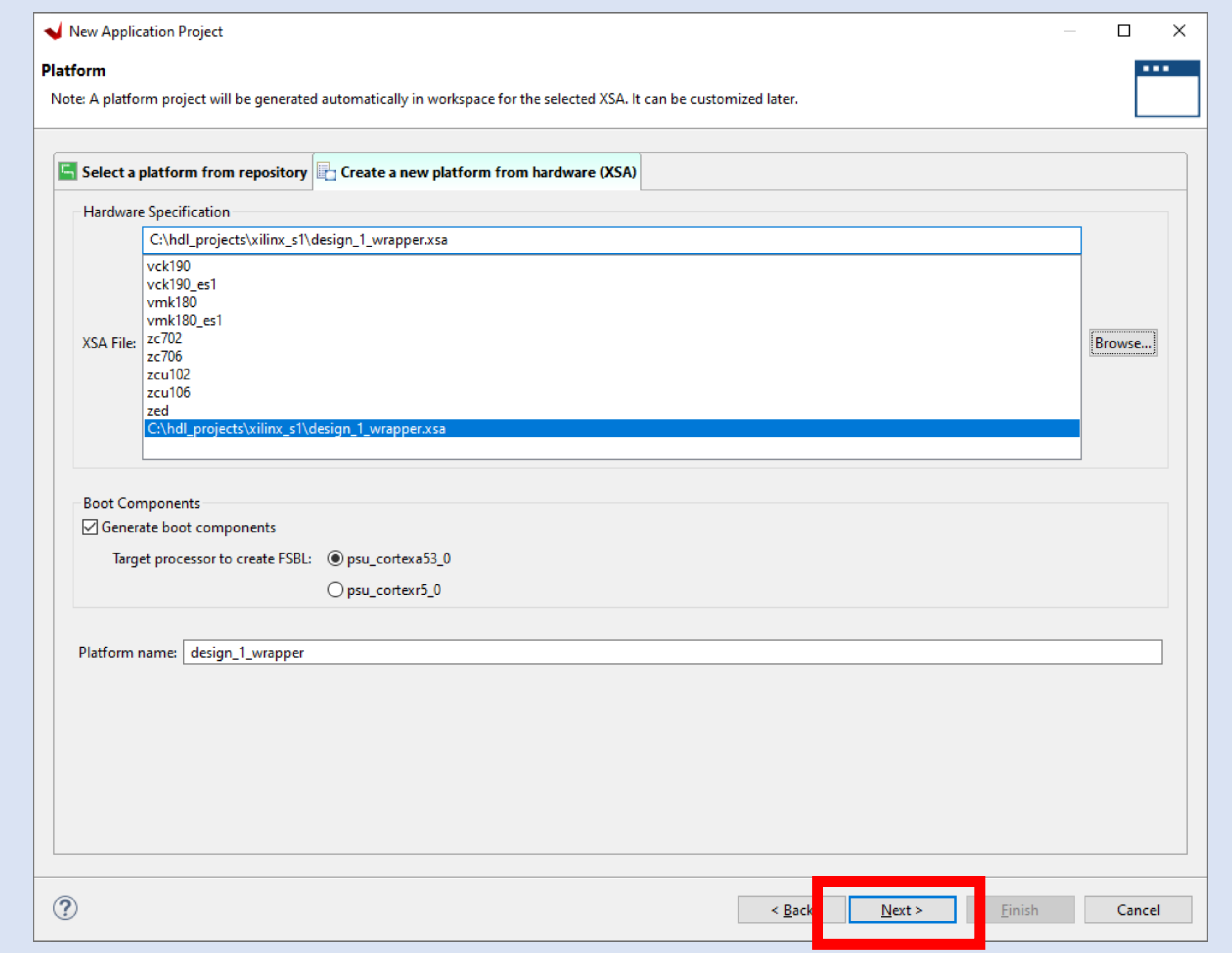


Name	Date modified	Type	Size
wksp	30/07/2021 21:09	File folder	
xilinx_s1.cache	30/07/2021 20:45	File folder	
xilinx_s1.gen	30/07/2021 19:54	File folder	
xilinx_s1.hw	30/07/2021 19:51	File folder	
xilinx_s1.ip_user_files	30/07/2021 19:51	File folder	
xilinx_s1.runs	30/07/2021 20:45	File folder	
xilinx_s1.sim	30/07/2021 19:51	File folder	
xilinx_s1.srcs	30/07/2021 19:54	File folder	
design_1_wrapper.xsa	30/07/2021 21:03	XSA File	1,553 KB

File name: design_1_wrapper.xsa

Lab 1: Understanding Vitis Project creation & Flow

Step 44 – Click on OK



New Application Project
Select a platform from repository
Create a new platform from hardware (XSA)
Browse...
< Back
Next >
Finish
Cancel

Platform

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

Hardware Specification

XSA File: C:\hdl_projects\xilinx_s1\design_1_wrapper.xsa

- vck190
- vck190_es1
- vmk180
- vmk180_es1
- zc702
- zc706
- zcu102
- zcu106
- zed
- C:\hdl_projects\xilinx_s1\design_1_wrapper.xsa

Boot Components

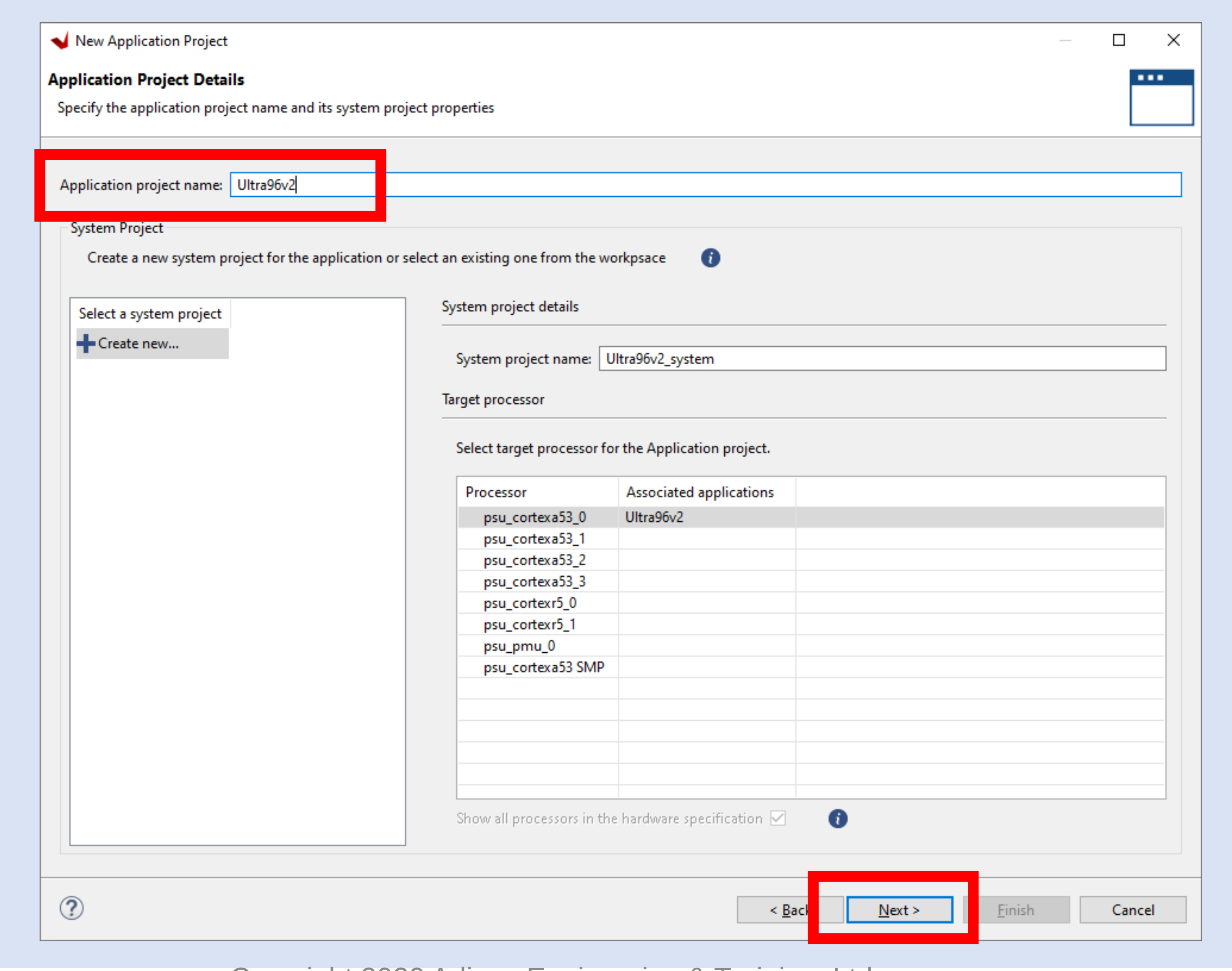
☑ Generate boot components

Target processor to create FSBL: ● psu_cortexa53_0 ○ psu_cortexr5_0

Platform name: design_1_wrapper

Lab 1: Understanding Vitis Project creation & Flow

Step 45 – Enter a project name and select Next



Application Project Details

Specify the application project name and its system project properties

Application project name: Ultra96v2

System Project

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

Select a system project

System project details

System project name: Ultra96v2_system

Target processor

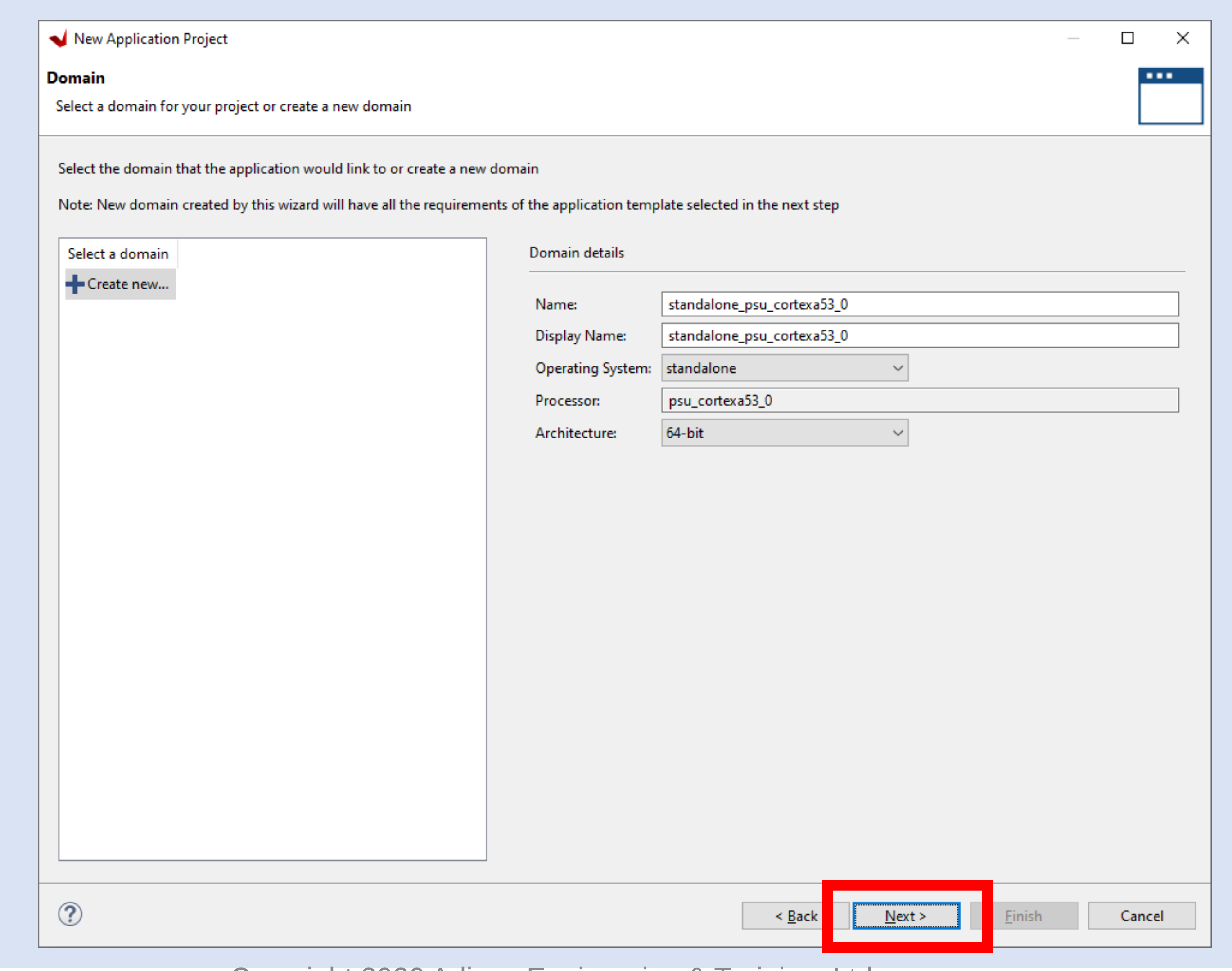
Select target processor for the Application project.

Processor	Associated applications
psu_cortexa53_0	Ultra96v2
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 1: Understanding Vitis Project creation & Flow

Step 46 – Click 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

Domain details

Name: standalone_psu_cortexa53_0

Display Name: standalone_psu_cortexa53_0

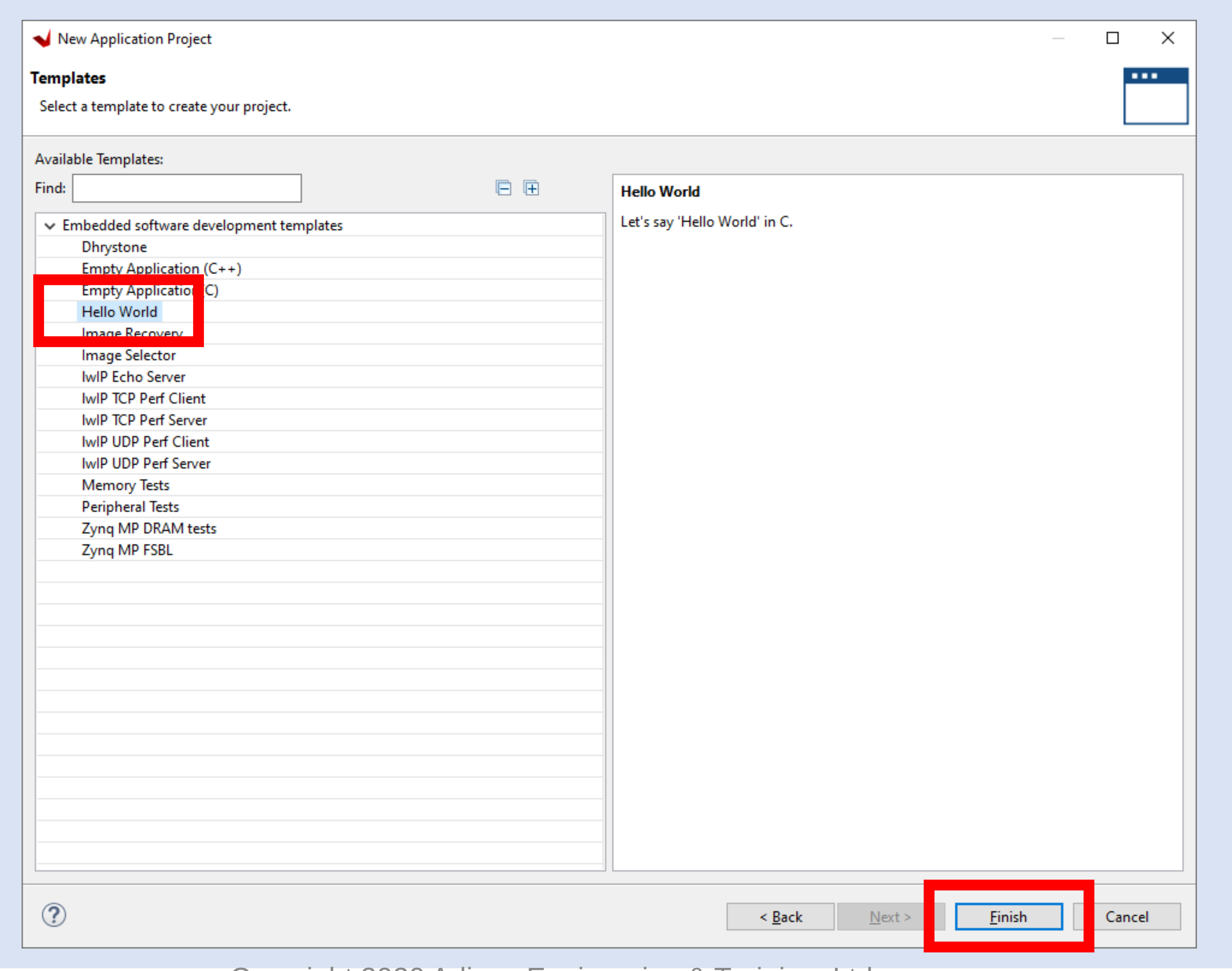
Operating System: standalone

Processor: psu_cortexa53_0

Architecture: 64-bit

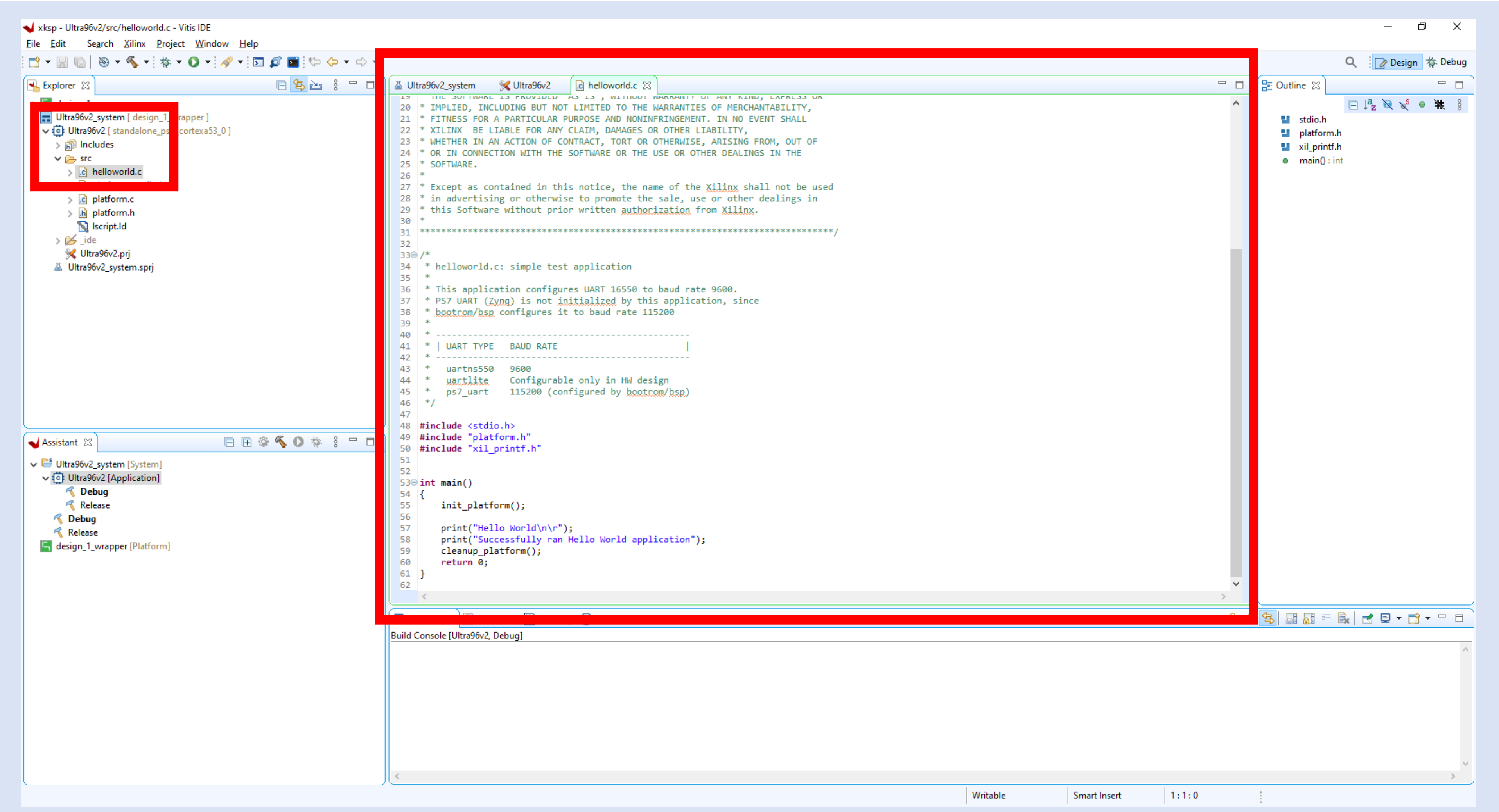
Lab 1: Understanding Vitis Project creation & Flow

Step 47 - Select Hello World and click finish



Lab 1: Understanding Vitis Project creation & Flow

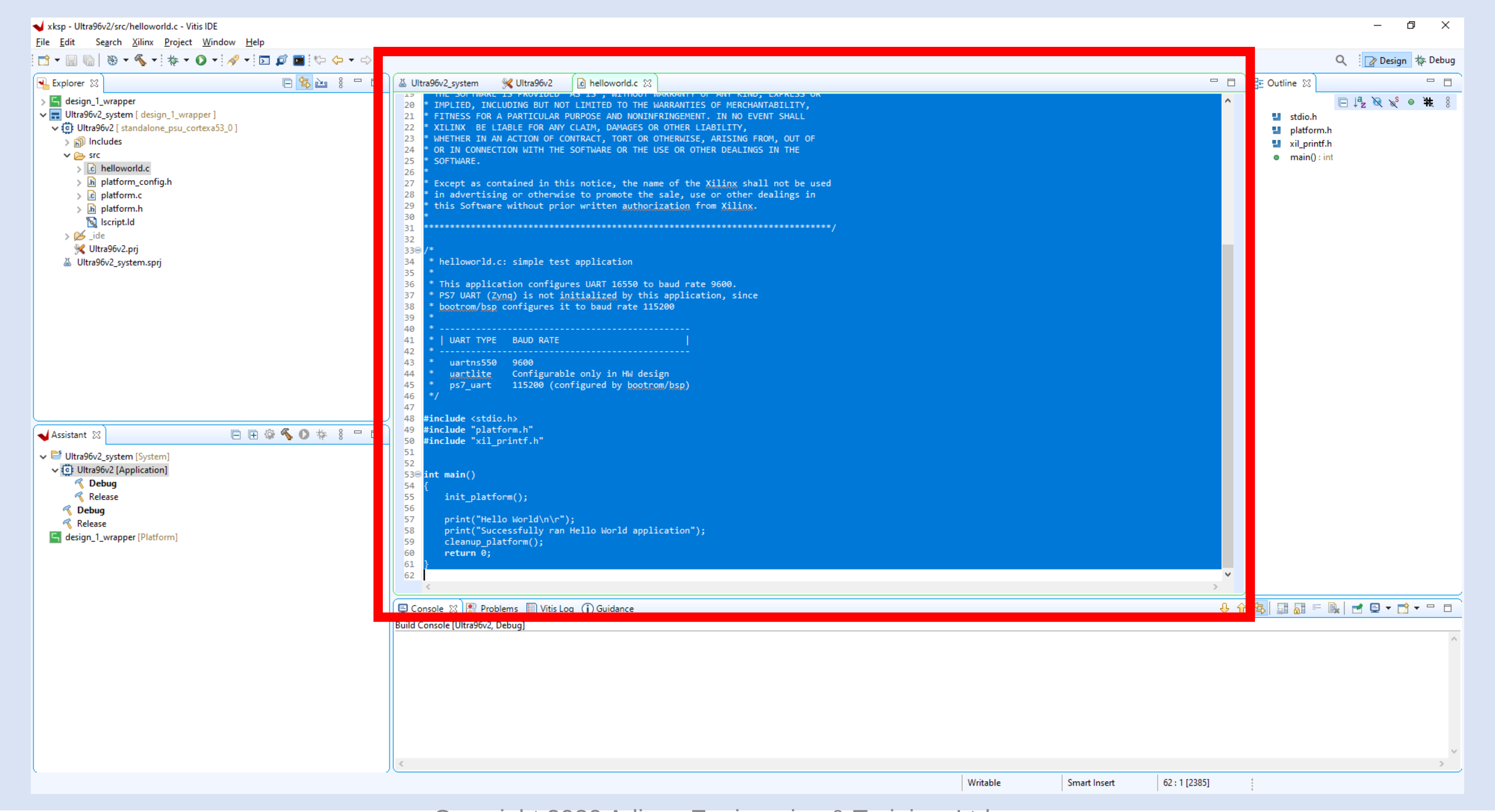
Step 48 - From the application / src folder double click and open the helloworld.c



```
 * 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
 *   uartlite    Configurable only in HW design
 *   ps7_uart    115200 (configured by bootrom/bsp)
 */

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


int main()
{
    init_platform();

    print("Hello World\n\r");
    print("Successfully ran Hello World application");
    cleanup_platform();
    return 0;
}
```

Lab 1: Understanding Vitis Project creation & Flow

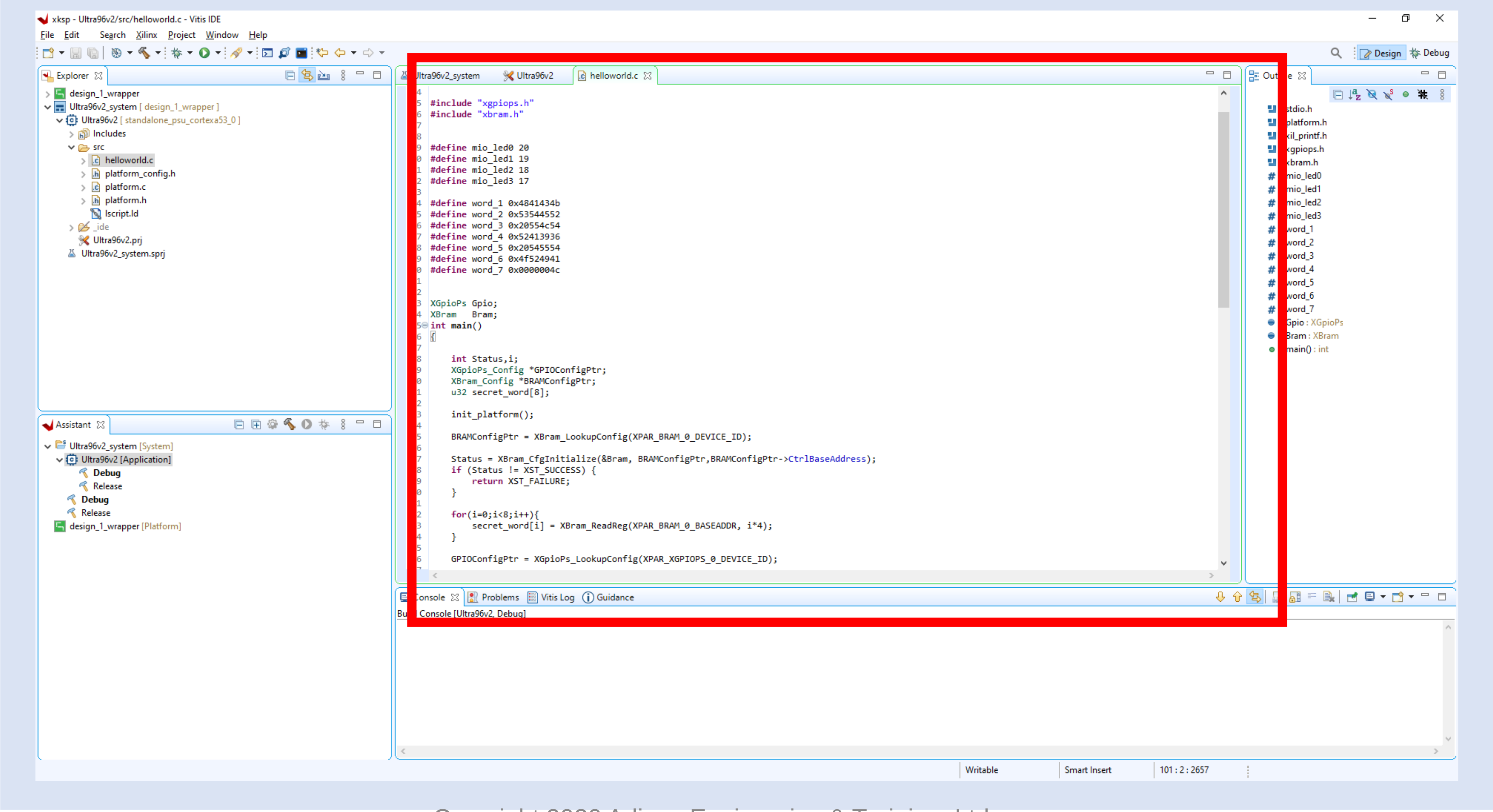
Step 49 - Click CNTRL-A to select all the code in the file and delete it



```
 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
 *   uartlite    Configurable only in HW design
 *   ps7_uart    115200 (configured by bootrom/bsp)
 */

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


int main()
{
    init_platform();

    print("Hello World\n\r");
    print("Successfully ran Hello World application");
    cleanup_platform();
    return 0;
}

```

Lab 1: Understanding Vitis Project creation & Flow

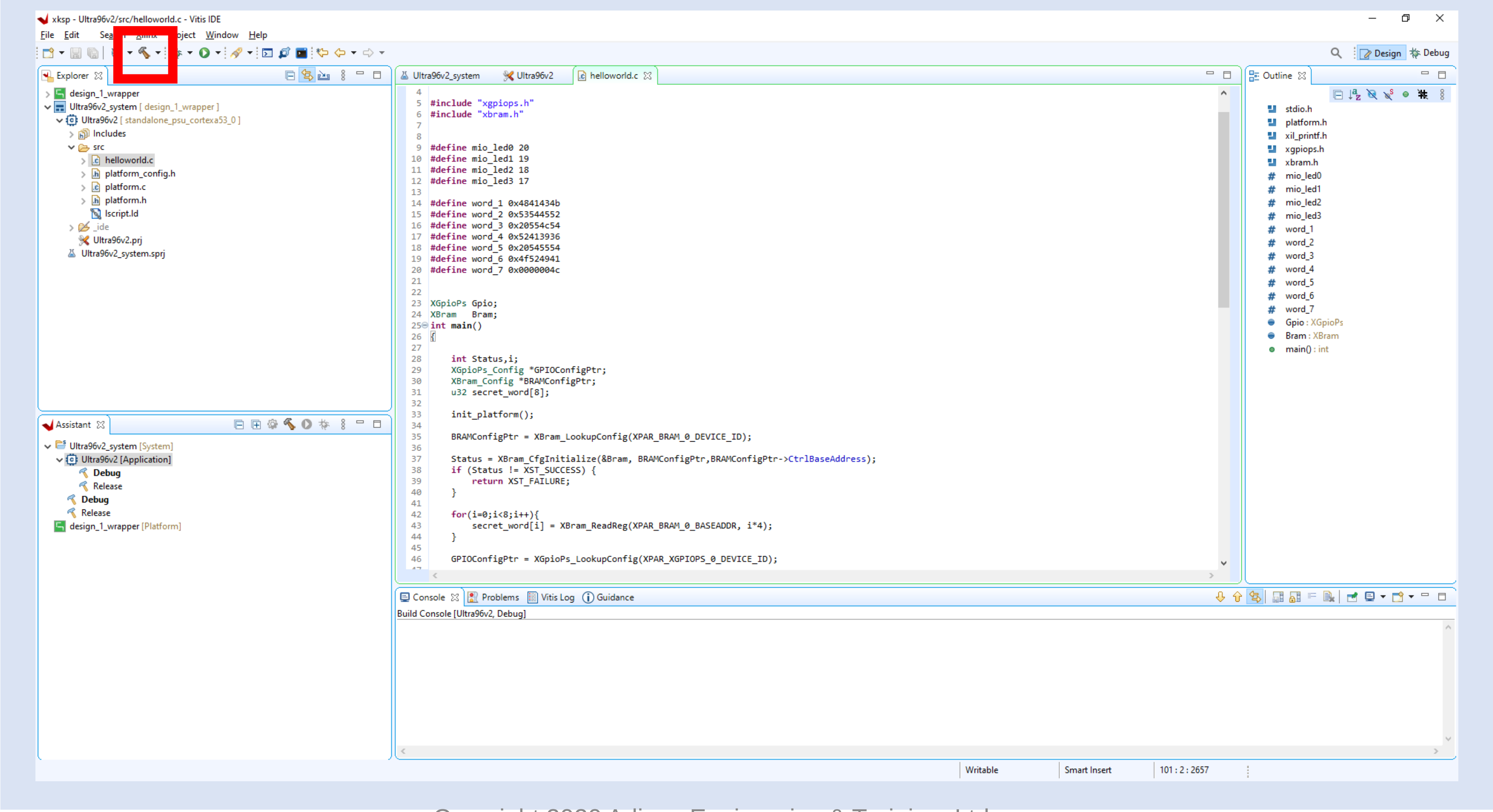
Step 50 - Copy and paste in the code from the Github repo session one lab



```
#include "xgpiops.h"
#include "xbram.h"


#define mio_led0 20
#define mio_led1 19
#define mio_led2 18
#define mio_led3 17

#define word_1 0x4841434b
#define word_2 0x53544552
#define word_3 0x20554c54
#define word_4 0x52413936
#define word_5 0x20545554
#define word_6 0x4f524941
#define word_7 0x0000004c


XGpioPs Gpio;
XBram   Bram;
int main()
{

    int Status,i;
    XGpioPs_Config *GPIOConfigPtr;
    XBram_Config *BRAMConfigPtr;
    u32 secret_word[8];

    init_platform();

    BRAMConfigPtr = XBram_LookupConfig(XPAR_BRAM_0_DEVICE_ID);

    Status = XBram_CfgInitialize(&Bram, BRAMConfigPtr,BRAMConfigPtr->CtrlBaseAddress);
    if (Status != XST_SUCCESS) {
        return XST_FAILURE;
    }

    for(i=0;i<8;i++){
        secret_word[i] = XBram_ReadReg(XPAR_BRAM_0_BASEADDR, i*4);
    }

    GPIOConfigPtr = XGpioPs_LookupConfig(XPAR_XGPIOPS_0_DEVICE_ID);
```

Lab 1: Understanding Vitis Project creation & Flow

Step 51 - Click on the Hammer to Build the project

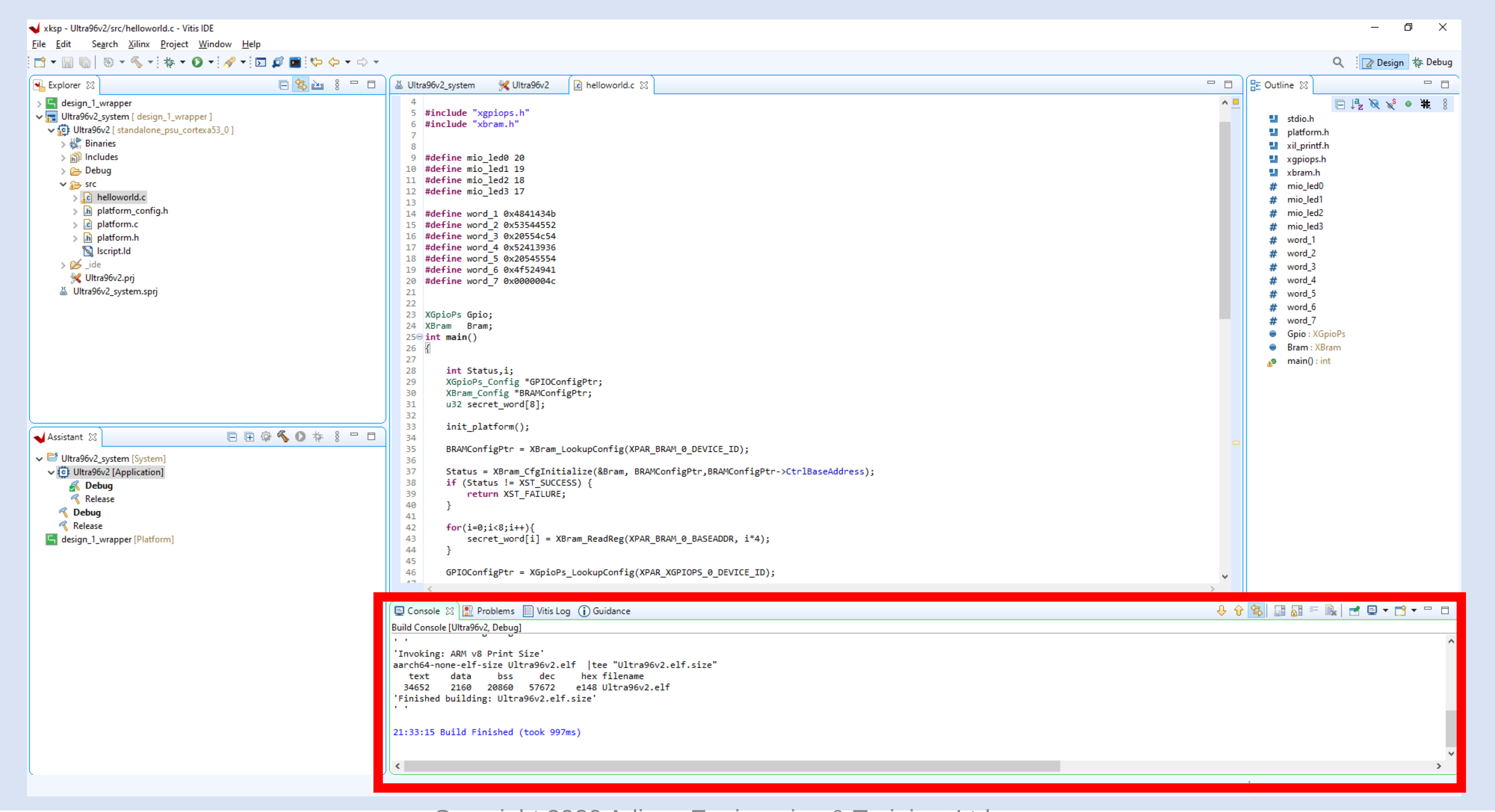


```c

#include "xgpiops.h"
#include "xbram.h"


#define mio_led0 20
#define mio_led1 19
#define mio_led2 18
#define mio_led3 17

#define word_1 0x4841434b
#define word_2 0x53544552
#define word_3 0x20554c54
#define word_4 0x52413936
#define word_5 0x20545554
#define word_6 0x4f524941
#define word_7 0x0000004c


XGpioPs Gpio;
XBram   Bram;
int main()
{

    int Status,i;
    XGpioPs_Config *GPIOConfigPtr;
    XBram_Config *BRAMConfigPtr;
    u32 secret_word[8];

    init_platform();

    BRAMConfigPtr = XBram_LookupConfig(XPAR_BRAM_0_DEVICE_ID);

    Status = XBram_CfgInitialize(&Bram, BRAMConfigPtr,BRAMConfigPtr->CtrlBaseAddress);
    if (Status != XST_SUCCESS) {
        return XST_FAILURE;
    }

    for(i=0;i<8;i++){
        secret_word[i] = XBram_ReadReg(XPAR_BRAM_0_BASEADDR, i*4);
    }

    GPIOConfigPtr = XGpioPs_LookupConfig(XPAR_XGPIOPS_0_DEVICE_ID);
```

Build Console [Ultra96v2, Debug]

Lab 1: Understanding Vitis Project creation & Flow

Step 52 - It will take a few minutes to compile. Successful completion will show as below

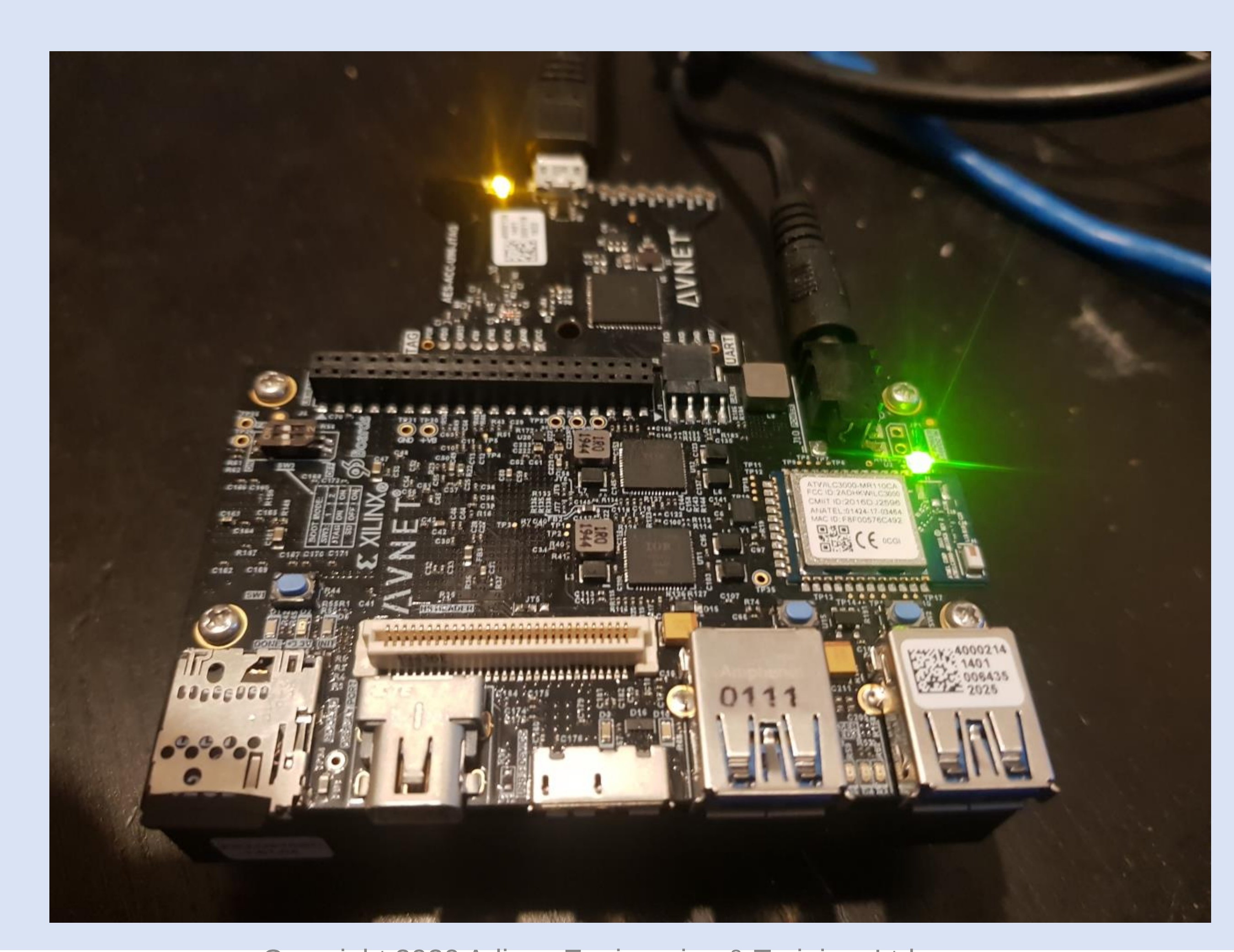


```c

#include "xgpiops.h"
#include "xbram.h"


#define mio_led0 20
#define mio_led1 19
#define mio_led2 18
#define mio_led3 17

#define word_1 0x4841434b
#define word_2 0x53544552
#define word_3 0x20554c54
#define word_4 0x52413936
#define word_5 0x20545554
#define word_6 0x4f524941
#define word_7 0x0000004c


XGpioPs Gpio;
XBram   Bram;
int main()
{

    int Status,i;
    XGpioPs_Config *GPIOConfigPtr;
    XBram_Config *BRAMConfigPtr;
    u32 secret_word[8];

    init_platform();

    BRAMConfigPtr = XBram_LookupConfig(XPAR_BRAM_0_DEVICE_ID);

    Status = XBram_CfgInitialize(&Bram, BRAMConfigPtr,BRAMConfigPtr->CtrlBaseAddress);
    if (Status != XST_SUCCESS) {
        return XST_FAILURE;
    }

    for(i=0;i<8;i++){
        secret_word[i] = XBram_ReadReg(XPAR_BRAM_0_BASEADDR, i*4);
    }

    GPIOConfigPtr = XGpioPs_LookupConfig(XPAR_XGPIOPS_0_DEVICE_ID);
```

```
' '
'Invoking: ARM v8 Print Size'
aarch64-none-elf-size Ultra96v2.elf  |tee "Ultra96v2.elf.size"
   text    data     bss     dec     hex filename
  34652    2160   20860   57672    e148 Ultra96v2.elf
'Finished building: Ultra96v2.elf.size'
' '


21:33:15 Build Finished (took 997ms)
```

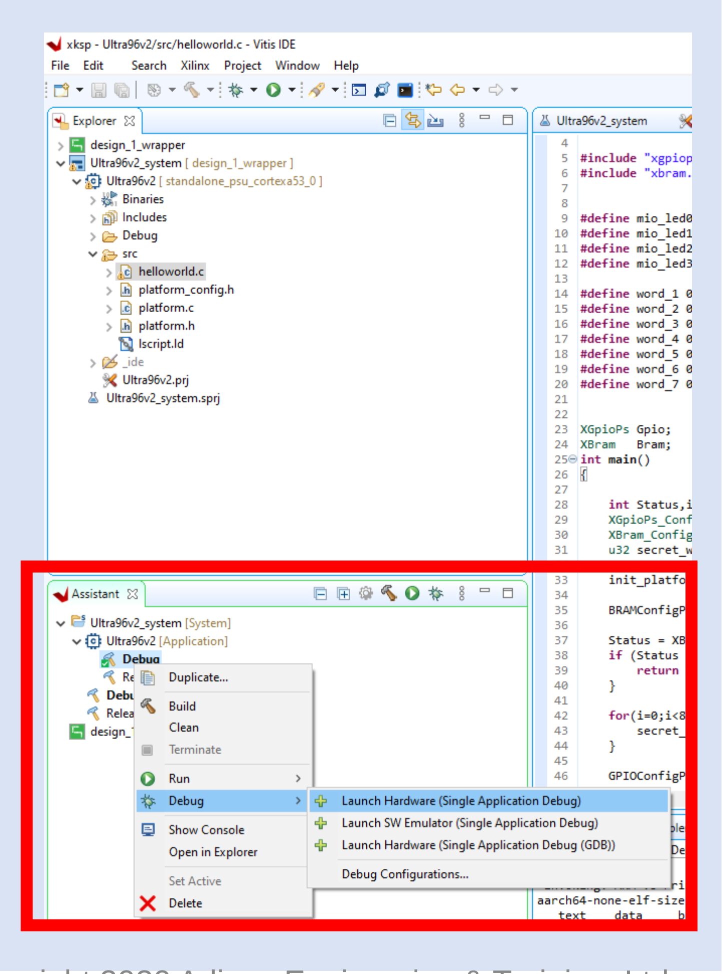
Lab 1: Understanding Vitis Project creation & Flow

Step 53 - Connect the JTAG Pod to the Ultra96V2, connect the USB and Power Cable. Check the book Mode is JTAG. Power on the board.



Lab 1: Understanding Vitis Project creation & Flow

Step 54 - From the assistant view, right click on debug and select debug-> launch on Hardware. This will configure the Ultra96V2, download the PL and the application.

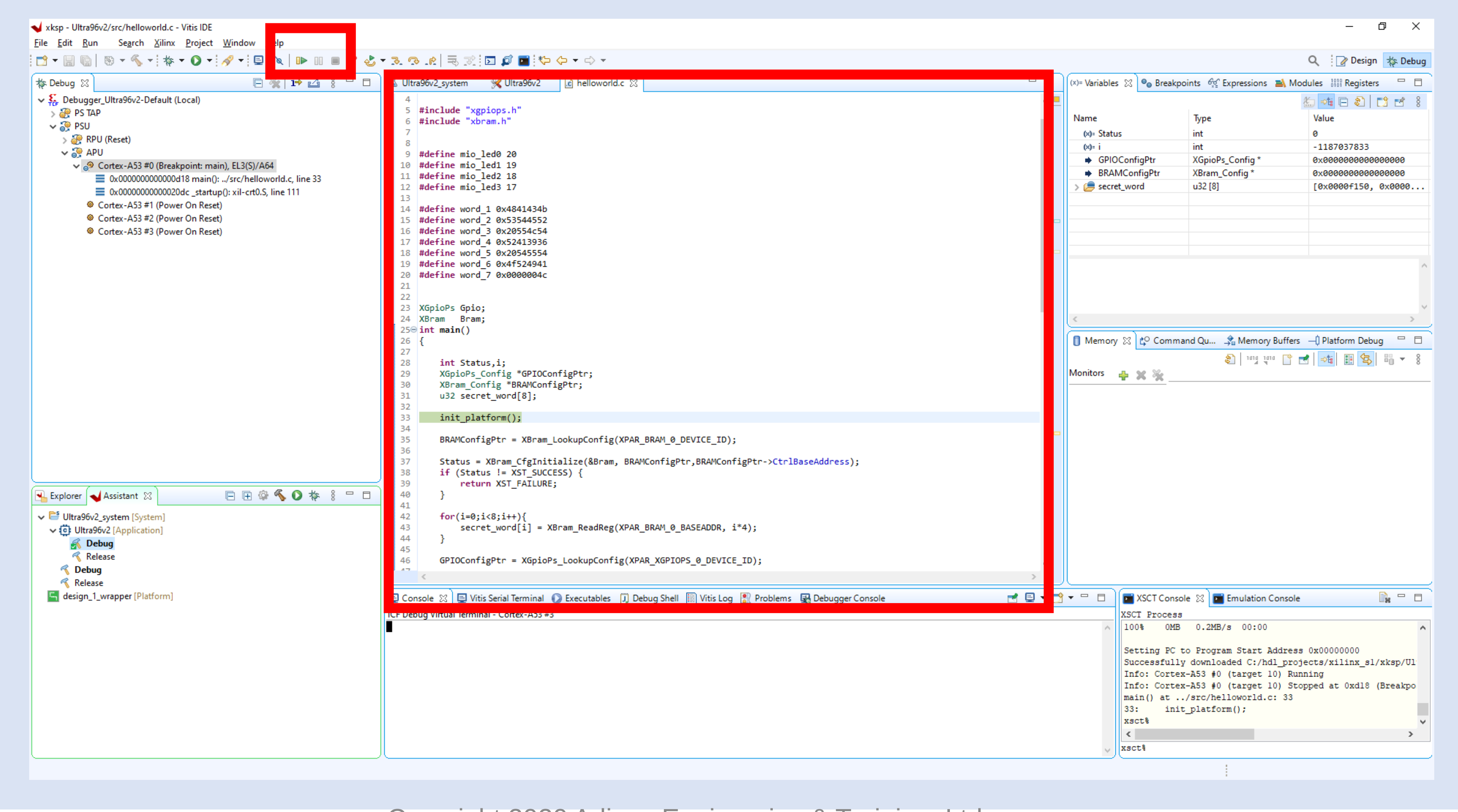


```

#include "xgpiop
#include "xbram.


#define mio_led0
#define mio_led1
#define mio_led2
#define mio_led3

#define word_1 0
#define word_2 0
#define word_3 0
#define word_4 0
#define word_5 0
#define word_6 0
#define word_7 0


XGpioPs Gpio;
XBram   Bram;
int main()
{

    int Status,i
    XGpioPs_Conf
    XBram_Config
    u32 secret_w

    init_platfo

    BRAMConfigP

    Status = XB
    if (Status
        return
    }

    for(i=0;i<8
        secret_
    }

    GPIOConfigP
```

```
aarch64-none-elf-size
   text    data     b
```

Lab 1: Understanding Vitis Project creation & Flow

Step 55 - The application will pause at the first instruction on the Ultra96V2. Click the Run Arrow and check the lights on the Ultra96V2.



Debugger_Ultra96v2-Default (Local)
- PS TAP
- PSU
  - RPU (Reset)
  - APU
    - Cortex-A53 #0 (Breakpoint: main), EL3(S)/A64
      - 0x0000000000000d18 main(): ../src/helloworld.c, line 33
      - 0x00000000000020dc _startup(): xil-crt0.S, line 111
    - Cortex-A53 #1 (Power On Reset)
    - Cortex-A53 #2 (Power On Reset)
    - Cortex-A53 #3 (Power On Reset)

Ultra96v2_system [System]
- Ultra96v2 [Application]
  - **Debug**
  - Release
- **Debug**
- Release
- design_1_wrapper [Platform]

```c

#include "xgpiops.h"
#include "xbram.h"


#define mio_led0 20
#define mio_led1 19
#define mio_led2 18
#define mio_led3 17

#define word_1 0x4841434b
#define word_2 0x53544552
#define word_3 0x20554c54
#define word_4 0x52413936
#define word_5 0x20545554
#define word_6 0x4f524941
#define word_7 0x0000004c


XGpioPs Gpio;
XBram   Bram;
int main()
{

    int Status,i;
    XGpioPs_Config *GPIOConfigPtr;
    XBram_Config *BRAMConfigPtr;
    u32 secret_word[8];

    init_platform();

    BRAMConfigPtr = XBram_LookupConfig(XPAR_BRAM_0_DEVICE_ID);

    Status = XBram_CfgInitialize(&Bram, BRAMConfigPtr,BRAMConfigPtr->CtrlBaseAddress);
    if (Status != XST_SUCCESS) {
        return XST_FAILURE;
    }

    for(i=0;i<8;i++){
        secret_word[i] = XBram_ReadReg(XPAR_BRAM_0_BASEADDR, i*4);
    }

    GPIOConfigPtr = XGpioPs_LookupConfig(XPAR_XGPIOPS_0_DEVICE_ID);
```

Name	Type	Value
Status	int	0
i	int	-1187037833
GPIOConfigPtr	XGpioPs_Config *	0x0000000000000000
BRAMConfigPtr	XBram_Config *	0x0000000000000000
secret_word	u32 [8]	[0x0000f150, 0x0000...

```
XSCT Process
100%    0MB   0.2MB/s  00:00

Setting PC to Program Start Address 0x00000000
Successfully downloaded C:/hdl_projects/xilinx_sl/xksp/Ul
Info: Cortex-A53 #0 (target 10) Running
Info: Cortex-A53 #0 (target 10) Stopped at 0xd18 (Breakpo
main() at ../src/helloworld.c: 33
33:     init_platform();
xsct%
xsct%
```

Lab 1: Understanding Vitis Project creation & Flow

Step 56 - Check the LEDs if the secret code is read from the BRAM the LEDS will flash on and off alternatively. If the word fails all four LEDs will flash on and off together.