***Advanced System on Chip Design Course***

**Getting Started Guide**

**Issue 1.0**

Contents

[1 Introduction 1](#_Toc86677301)

[1.1 Learning Outcomes 1](#_Toc86677302)

[1.2 Scope of this guide 1](#_Toc86677303)

[2 Requirements 2](#_Toc86677304)

[2.1 Software requirements 2](#_Toc86677305)

[2.2 Hardware requirements 2](#_Toc86677306)

[3 Xilinx software setup 3](#_Toc86677307)

[3.1 Vivado 3](#_Toc86677308)

[3.1.1 Starting a new project 3](#_Toc86677309)

[3.2 Vitis IDE 4](#_Toc86677310)

[3.2.1 Creating an application project 4](#_Toc86677311)

[3.3 Arm Development Studio 6](#_Toc86677312)

[3.3.1 Importing existing DS-5 projects 6](#_Toc86677313)

[4 Troubleshooting 7](#_Toc86677314)

[4.1 Issues when using Vivado 7](#_Toc86677315)

[4.1.1 Facing NSTD-1 and UCIO-1 errors when generating bitstream 7](#_Toc86677316)

[4.1.2 Critical warnings: PCW\_UIPARAM\_DDR\_DQS\_TO\_CLK\_DELAY\_x has negative value 7](#_Toc86677317)

[4.2 Issues when using Vitis IDE 7](#_Toc86677318)

[4.2.1 Makefile errors 7](#_Toc86677319)

# Introduction

## Learning Outcomes

At the end of this lab, you will be able to:

* Identify the hardware and software requirements for the lab exercises
* Demonstrate how to create a project in Xilinx Vivado and Vitis tools.
* Use Arm Development Studio to import lab projects.
* Identify known issues and apply troubleshooting solutions.

## Scope of this guide

This Getting Started Guide is intended to give you the necessary information to install and configure the tools for the labs of this course.

**NOTE:** This guide assumes that:

* You are working with a Windows OS

# Requirements

This section describes the software and hardware requirements in order to attempt all the labs within this course.

|  |  |  |  |
| --- | --- | --- | --- |
| **Software** | **Website** | **Version**1 | **OS**  |
| Xilinx Vivado ML | <https://www.xilinx.com/support/download.html> | 2020.1 | Windows |
| Xilinx Vitis IDE | <https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vitis.html> | 2021.1 | Windows |
| Arm Development Studio  | <https://developer.arm.com/tools-and-software/embedded/arm-development-studio/evaluate>  | Platinum 2021.b | Windows |

##  Software requirements

Table 1: Software tools and versions

**1** The software versions listed here are versions that we have verified to be working with our labs. You can use the latest available (and most stable) versions of the software, if backward and forward compatibility is supported.

## Hardware requirements

* Diligent Zybo Z7-10 FPGA board
* Monitor with HDMI support
* 2 HDMI cables
* USB A to micro USB Cable

# Xilinx software setup

## Vivado

To create a new project, first choose a workspace such that the path does not contain any spaces. For Windows OS, this could be a folder in the main working directory like “C:/Workspace”.

### Starting a new project

Open Vivado, click “Create Project” and choose the location as the workspace:



Figure 1: Vivado start page

Give it a meaningful name and click next. Select RTL Project and click “Next”:



Figure 2: Vivado project type

Skip through the next two pages as the constraints file can be added at a later stage. In the next page first search for “Zybo” in the search bar and click the download button next to “Zybo Z7-10” if the option is available. Then select it and click “Next” (Do not click the hyperlink but click the empty area next to “Zybo Z7-10”). Finally click “Finish”.



Figure 3: Choosing a board

A new project has been created and block designs can now be added. Once done with the block diagram, add the constraints file and generate the bitstream. Then, click “File” and “export hardware”. Select the option to include the bitstream and save the XSA file in the project folder.

## Vitis IDE

To open the Vitis IDE from Vivado, click “Tools” in the menu bar and select “Launch Vitis IDE”. Choose the same project folder as your workspace.

### Creating an application project

Click “File” -> “New” -> “Application Project”.



Figure 4: Creating an application project

Select the following tab and click browse to select the XSA file saved earlier:



Figure 5: Choosing the XSA file

Click next and give a name to the application project. Click “Next” until you reach the following page and choose “Empty Application(C)” and click “Finish”:



Figure 6: Creating an empty application

Once this is done, C code can be written within a file and added to the “src” folder within the application project.

## Arm Development Studio

The Arm DS-5 IDE has been superseded by Arm Development Studio. Some of the labs were developed using the Arm DS-5, these should also work in the latest Arm Development Studio.

### Importing existing DS-5 projects

Existing DS-5 projects can be imported into Arm Development Studio.

When importing existing DS-5 projects into Arm Development Studio, you might get prompted to migrate the existing projects to the new Arm Development Studio for future compatibility. Select **OK**. For example, see the snapshot below:



# Troubleshooting

This section provides some guidance on resolving known issues when using Vivado and Vitis

## Issues when using Vivado

### Facing NSTD-1 and UCIO-1 errors when generating bitstream

Follow the solution given at: <https://www.xilinx.com/support/answers/56354.html>.

### Critical warnings: PCW\_UIPARAM\_DDR\_DQS\_TO\_CLK\_DELAY\_x has negative value

Read the hardware errata document: https://digilent.com/reference/programmable-logic/zybo-z7/reference-manual?redirect=1#hardware\_errata

###

## Issues when using Vitis IDE

### Makefile errors

Replace all the code in 3 makefiles located in the following places in the explorer tab:

1. hw/drivers/<IP NAME>/src/Makefile
2. ps7\_cortexa9\_0/standalone\_ ps7\_cortexa9\_0/bsp/ ps7\_cortexa9\_0/libsrc/<IP NAME>/src/Makefile
3. zynq\_fsbl/zynq\_fsbl\_bsp/ps7\_cortexa9\_0/libsrc/<IP NAME>/src/Makefile

|  |  |
| --- | --- |
| Lights On | <IP NAME> can vary depending on what procedure is followed in the lab. For example, in Lab 7, if the downloaded AXI4 peripheral files are used then <IP NAME> will be “axi\_gpio\_asoc\_v1\_0”. |

Replace all the code with the following:

COMPILER**=**

ARCHIVER**=**

CP**=**cp

COMPILER\_FLAGS**=**

EXTRA\_COMPILER\_FLAGS**=**

LIB**=**libxil.a

RELEASEDIR**=**../../../lib

INCLUDEDIR**=**../../../include

INCLUDES**=**-I./. -I${INCLUDEDIR}

INCLUDEFILES**=**\*.h

LIBSOURCES**=** $(wildcard \*.c)

OUTS **=** \*.o

OBJECTS **=** $(addsuffix .o, $(basename $(wildcard \*.c)))

ASSEMBLY\_OBJECTS **=** $(addsuffix .o, $(basename $(wildcard \*.S)))

libs**:**

 echo "Compiling axi\_gpio\_asoc..."

 $(COMPILER) $(COMPILER\_FLAGS) $(EXTRA\_COMPILER\_FLAGS) $(INCLUDES) $(LIBSOURCES)

 $(ARCHIVER) -r ${RELEASEDIR}/${LIB} ${OBJECTS} ${ASSEMBLY\_OBJECTS}

 make clean

include**:**

 ${CP} $(INCLUDEFILES) $(INCLUDEDIR)

clean**:**

 rm -rf ${OBJECTS} ${ASSEMBLY\_OBJECTS}