**Intro to System-on-Chip Design Course**

**Getting Started Guide**

**Issue 1.0**

Contents

[1 Introduction 1](#_Toc83818021)

[1.1 Scope of this Guide 1](#_Toc83818022)

[2 Requirements 2](#_Toc83818023)

[2.1 Software requirements 2](#_Toc83818024)

[3 Keil setup 3](#_Toc83818025)

[3.1 Download and Installation of Keil 3](#_Toc83818026)

[3.2 Creating a new KEIL µVision5 Project 4](#_Toc83818027)

[3.3 Configuring the Project Options 5](#_Toc83818028)

[3.4 Add Files to the Project 8](#_Toc83818029)

[3.5 Build project to generate code.hex file 9](#_Toc83818030)

[3.6 Remove Files from the Project 9](#_Toc83818031)

[3.7 Add Group to the Project 9](#_Toc83818032)

[3.8 Rename newly added group 9](#_Toc83818033)

[3.9 Remove Group from the Project 9](#_Toc83818034)

[4 Xilinx Vivado 10](#_Toc83818035)

[4.1 Download and Installation of Vivado 10](#_Toc83818036)

[4.2 Creating new Vivado project 10](#_Toc83818037)

[4.3 Edit Constraint file 11](#_Toc83818038)

# Introduction

## Scope of this Guide

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

**NOTE:** Keil MDK only supports the Windows OS at the time of writing this.

# Requirements

## Software requirements

The following table is a list of recommended software tools that are used in this guide.

|  |  |  |  |
| --- | --- | --- | --- |
| **Software** | **Website** | **Version**1 | **OS** |
| Keil MDK | <http://www.keil.com/> | V5.25.2.0 | Windows |
| Xilinx Vivado ML | <https://www.xilinx.com/support/download.html> | 2021.1 | Windows, Linux |
| TeraTerm | https://ttssh2.osdn.jp/index.html.en | 4.106 | Windows |

**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.

# Keil setup

## Download and Installation of Keil

Ensure that you have installed the Arm Keil.

To download the KEIL µVision 5, follow these instructions:

1. Go to [Keil Product Downloads](https://www.keil.com/download/product/) page.
2. Click on the MDK-Arm download option.
3. Download the installer by clicking on ‘MDK535.EXE’ and wait for the download to finish.

Graphical user interface, text, website

Description automatically generated

Figure 1: Keil Download page

1. Navigate to your “Downloads” folder and click on the installer.
2. Run the installer using all the default settings and wait for the installation to finish.
3. Click ‘Finish’. This should prompt up the KEIL pack installer.
4. Install the relevant packs related to your board and/or project.
   1. On the left, Expand ARM -> ARM Cortex M0
   2. Click on CMSDK\_CM0
   3. On the right install the Device Specific packs and also CMSIS.
5. You are done!

## Creating a new KEIL µVision5 Project

This section explains the process of creating a new KEIL µVision 5 project.

1. Start KEIL and click on Project 🡪 Create new µVision project.
2. Select the target device and click OK. In this course ARM Cortex-M0 is used.

Graphical user interface, text, application

Description automatically generated

Figure 2: Select Target Device Window

1. In the Manage Run-Time environment window that opens,
   1. Expand CMSIS and select Core.

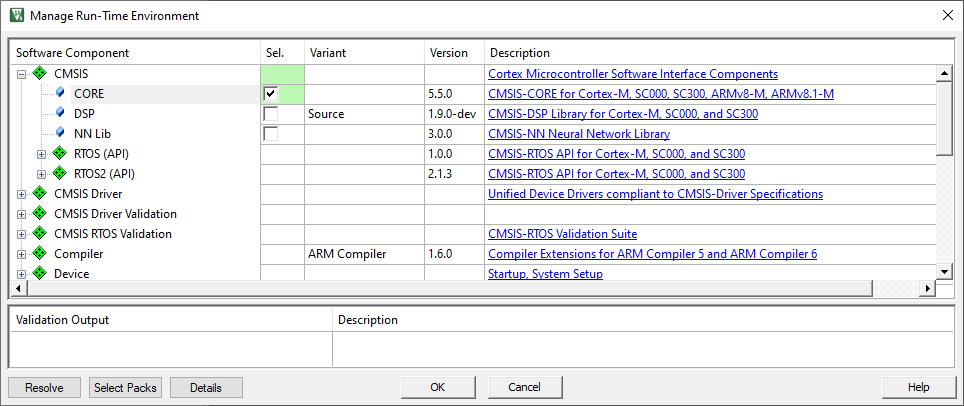


Figure 3: Manage Run Time Environment Window

1. Click ‘OK’ and your project should be setup.
2. Expand ‘Target 1’ on the project file explorer on the left of the screen.
3. To add files
   1. Right Click on the ‘Source Group 1’ and ‘Add new item to group ‘Source Group 1’.
   2. Select the type of file and enter its name.

## Configuring the Project Options

This section concerns on configuring the project to the suitable settings.

1. Click on the ‘Options for Target’ option on the tool bar.

Graphical user interface, application, Word

Description automatically generated

Figure 4: Select Options for Target

Each tab in this window relates to different project settings. Here is a brief explanation for each. **The settings presented on screenshots are correct settings for programs in this course.** Pay careful attention to set your settings in the same way, as not doing so may result in your software not working.

1. Device: Select the device from the µVision Device Database. In this course ARM Cortex-M0 is used.

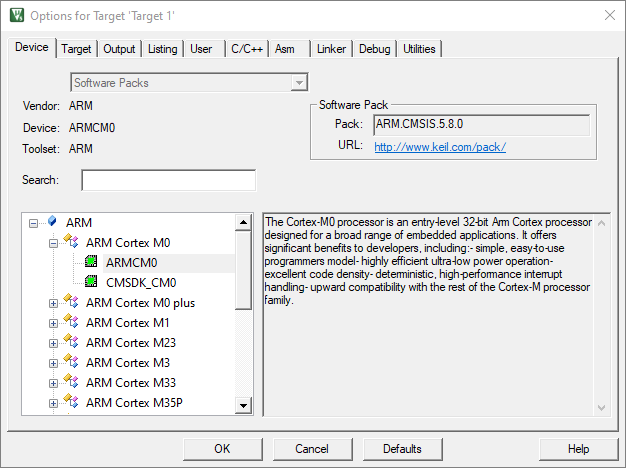


Figure 5: Options for Target: Device Selection

1. Target: Specify the hardware of your application

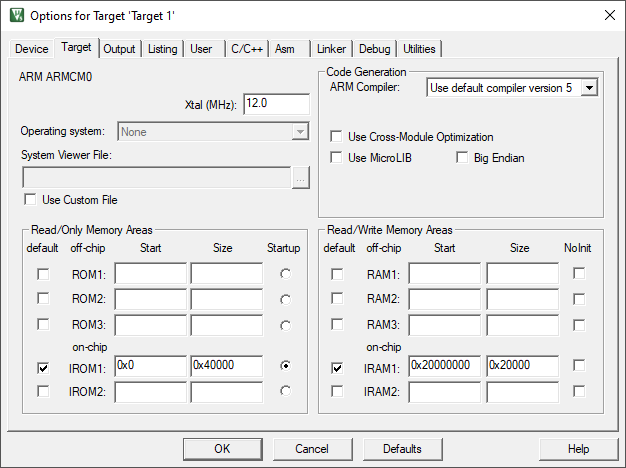


Figure 6: Options for Target: Target Tab

* + NOTE: The settings for Read/Only Memory Areas and Read/Write Memory Areas are used to create a linker scatter file.
  + This requires that the setting Use Memory Layout from the Target Dialog is enabled in the Linker dialog.

1. Output: Define the output files of the tool chain and allows you to start user programs after the build process. Be sure to change **Name of Executable** to **“code”**.

Graphical user interface, text, application, email

Description automatically generated

Figure 7: Options for Target: Output Tab

1. Listing: Specify all listing files generated by the tool chain.

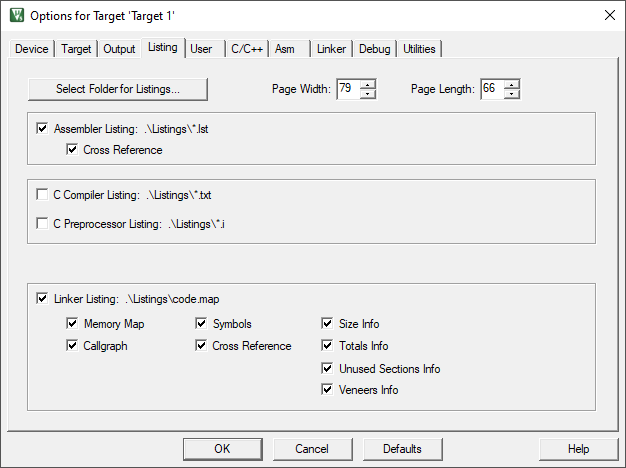


Figure 8: Options for Target: Listing Tab

1. User: Specify user programs executed before and after compilation/build or after build.

* In After Build/Rebuild: Run #1 enter

fromelf -cvf .\Objects\code.axf --vhx --32x1 -o code.hex

* In After Build/Rebuild: Run #2 enter

fromelf -cvf .\Objects\code.axf -o disasm.txt

After building, a **code.hex** will be created.

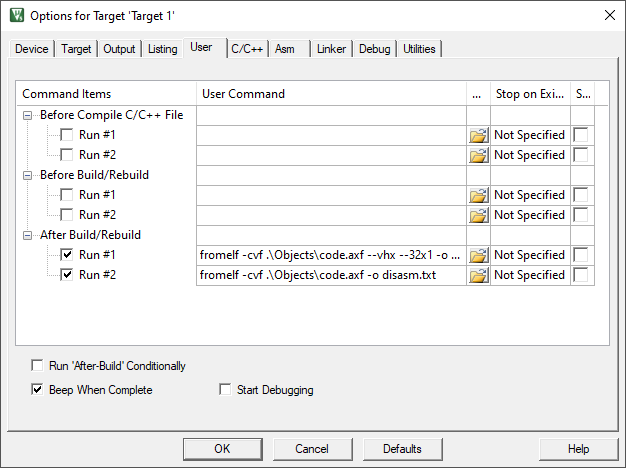


Figure 9: Options for Target: User Tab

1. C/C++: Set C/C++ compiler-specific tool options like optimization or variable allocation.

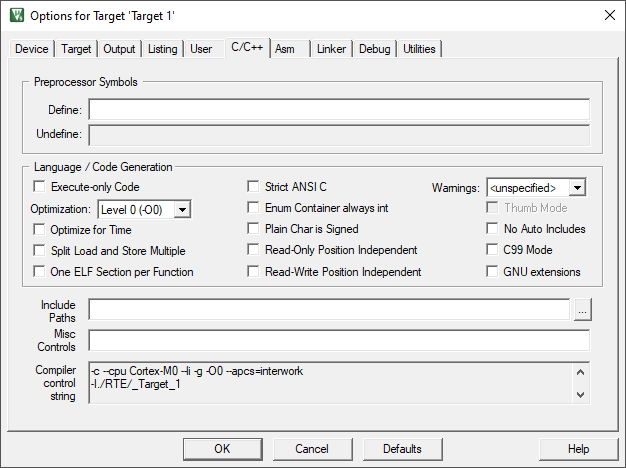


Figure 10: Options for Target: C/C++ Tab

1. Asm: Set assembler-specific tool options like macro processing.

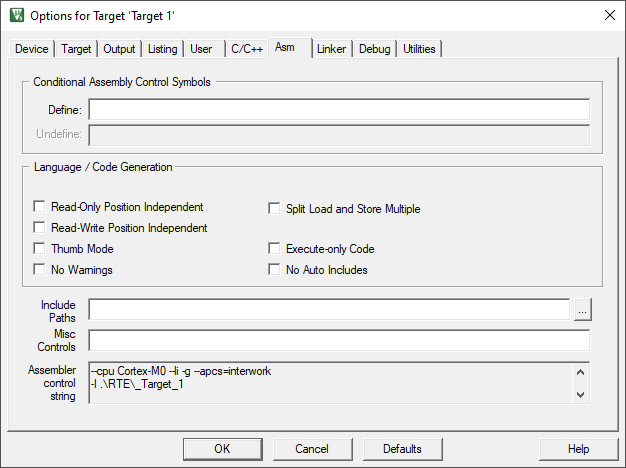


Figure 11: Options for Target: Asm Tab

1. Linker: Set linker-related options. Typically, linker settings are required to configure the physical memory layout of the target system. Define the location of memory classes and sections in here.

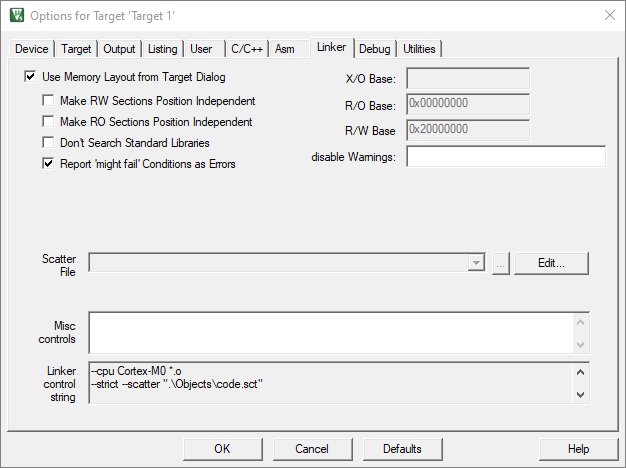


Figure 12: Options for Target: Linker Tab

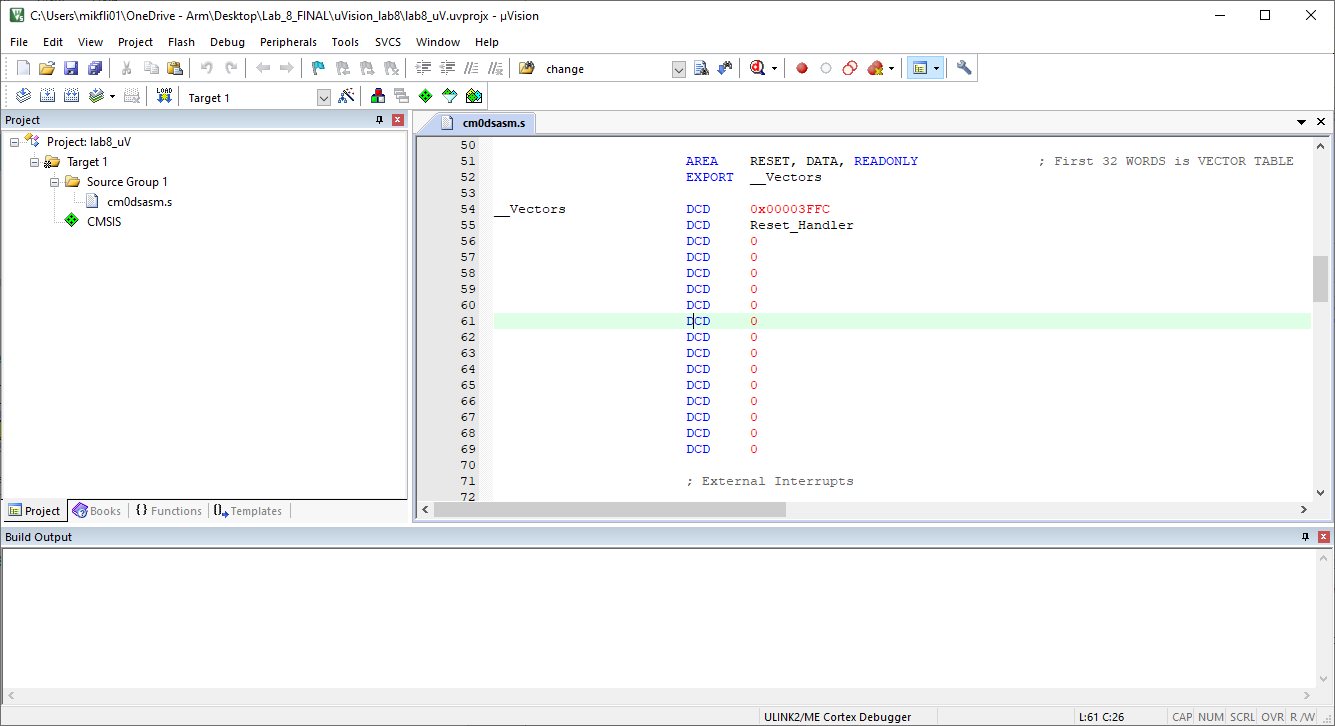
1. Debug: Settings for the µVision Debugger. (Leave as default)
2. Utilities: Configure utilities for Flash programming. (Leave as default)

## Add Files to the Project

On the project pane, right click on Source Group 1 *(note this is a default group created when a new project is created and can be renamed)* and select either of the following:

* Add new item to group ‘Source Group 1’
  + If you select this option, select the file type and then enter file name.
  + This option is best when you are creating a new file
* Add existing files to group ‘Source Group 1’
  + Navigate to the location of the file, select it and click add

Here is main screen of Keil uVision:



**3**

Figure 13: Layout of the IDE

**1**

**2**

* In the area on the left (labeled 1) you can see and manage your proram files,
* To the right (labeled 2) is code editor,
* At the bottom (labeled 3) is console, when you will see messages, warnings and error reports.

## Build project to generate code.hex file

If you are sure the settings mentioned above have been entered correctly, build the project by pressing “F7” on the keyboard or going to Project -> Build ‘<Project-…>’.

Please note <Project-…> is the name of your project.

After build is complete without any errors, a file ***code.hex*** file will be generated in the Keil project directory. Copy this file to the same location as the AHB2BRAM.v included in your Vivado project.

## Remove Files from the Project

To remove a file from the project, on the project pane, expand the group e.g “Source Group 1” and right click on the target file click Remove file

## Add Group to the Project

On the project pane, right click on Target 1 and select “Add Group”. A new group “**New Group**” will be created.

## Rename newly added group

* On the project pane, right click on Target 1 and select “Manage Project Items”.
* On the window that appears, double click on the group you want to rename under Groups to rename the group.
* Click OK when done.

## Remove Group from the Project

To remove a group from the project,

* on the project pane, right click on the group and select “**Remove Group ‘….’and its Files**”
  + Note: This also removes the files under the affected group

# Xilinx Vivado

## Download and Installation of Vivado

Download and install Vivado from <https://www.xilinx.com/developer/products/vivado.html>

## Steps to Program the Xilinx Device

The following are the steps needed to program the board with your design in Vivado:

1. Create a new Vivado project
   1. you will need to select the appropriate device,
   2. add/create and edit source files and a constraint file.
2. Run Synthesis successfully
3. Run Implementation successfully
4. Program and Debug the Device.

Please see subsections 4.3 – 4.6 for more details of the steps listed above.

## Creating new Vivado project

1. To create new project, open Vivado. Following screen should appear. Choose “Create Project”.

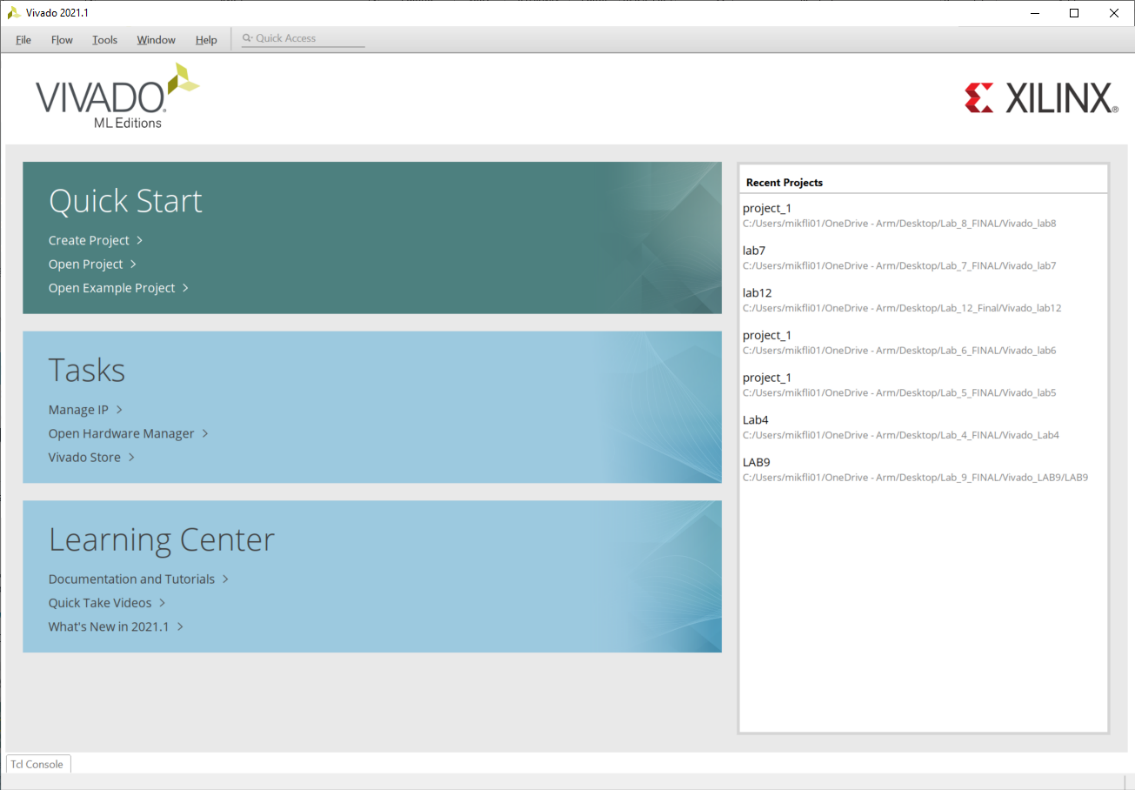


Figure 14: Newly launched Vivado page

1. Choose project name and directory. Click “Next”.
2. Select “RTL Project”, click “Next”.
3. Add source files (Verilog files provided for this course, including the Cortex-M0 files) and Click “Next”.
4. Add constraints file. This file is specific to the FPGA board and informs Vivado of pin layout and other board-specific characteristics.
5. Click Create File and enter a name for your file.
6. Click “Next”.
7. Choose FPGA as XC7A35TCPG236-1. Then click “Next”.
8. Another screen with project details summary should appear. Check if details are in line with requirements, click “Finish” and the project is created!

You would see that all the added files are nested under **AHBLITE\_SYS (1)** as shown in Figure 15.

Graphical user interface, application, email

Description automatically generated

**2**

**1**

Figure 15: Vivado IDE

## Edit Constraint file

Please see Figure 15 for label (2).

To edit the constraint file, expand Constraint on the Sources Pane (2) and double click on the **.*xdc*** file to open it. Enter the following and save the file.

# Clock signal

set\_property -dict {PACKAGE\_PIN W5 IOSTANDARD LVCMOS33} [get\_ports CLK]

create\_clock -add -name sys\_clk\_pin -period 10.00 -waveform {0 5} [get\_ports CLK]

#Active high RESET - BTNR on Basys 3 board

set\_property -dict {PACKAGE\_PIN T17 IOSTANDARD LVCMOS33} [get\_ports RESET]

#LEDs

set\_property -dict {PACKAGE\_PIN U16 IOSTANDARD LVCMOS33} [get\_ports LED[0]]

set\_property -dict {PACKAGE\_PIN E19 IOSTANDARD LVCMOS33} [get\_ports LED[1]]

set\_property -dict {PACKAGE\_PIN U19 IOSTANDARD LVCMOS33} [get\_ports LED[2]]

set\_property -dict {PACKAGE\_PIN V19 IOSTANDARD LVCMOS33} [get\_ports LED[3]]

set\_property -dict {PACKAGE\_PIN W18 IOSTANDARD LVCMOS33} [get\_ports LED[4]]

set\_property -dict {PACKAGE\_PIN U15 IOSTANDARD LVCMOS33} [get\_ports LED[5]]

set\_property -dict {PACKAGE\_PIN U14 IOSTANDARD LVCMOS33} [get\_ports LED[6]]

set\_property -dict {PACKAGE\_PIN V14 IOSTANDARD LVCMOS33} [get\_ports LED[7]]

##VGA Connector

set\_property -dict {PACKAGE\_PIN G19 IOSTANDARD LVCMOS33} [get\_ports vgaRed[0]]

set\_property -dict {PACKAGE\_PIN H19 IOSTANDARD LVCMOS33} [get\_ports vgaRed[1]]

set\_property -dict {PACKAGE\_PIN J19 IOSTANDARD LVCMOS33} [get\_ports vgaRed[2]]

set\_property -dict {PACKAGE\_PIN N18 IOSTANDARD LVCMOS33} [get\_ports vgaBlue[0]]

set\_property -dict {PACKAGE\_PIN L18 IOSTANDARD LVCMOS33} [get\_ports vgaBlue[1]]

set\_property -dict {PACKAGE\_PIN J17 IOSTANDARD LVCMOS33} [get\_ports vgaGreen[0]]

set\_property -dict {PACKAGE\_PIN H17 IOSTANDARD LVCMOS33} [get\_ports vgaGreen[1]]

set\_property -dict {PACKAGE\_PIN G17 IOSTANDARD LVCMOS33} [get\_ports vgaGreen[2]]

set\_property -dict {PACKAGE\_PIN P19 IOSTANDARD LVCMOS33} [get\_ports Hsync]

set\_property -dict {PACKAGE\_PIN R19 IOSTANDARD LVCMOS33} [get\_ports Vsync]

#USB-RS232 Interface

set\_property -dict {PACKAGE\_PIN B18 IOSTANDARD LVCMOS33} [get\_ports RsRx]

set\_property -dict {PACKAGE\_PIN A18 IOSTANDARD LVCMOS33} [get\_ports RsTx]

#Switches

set\_property -dict {PACKAGE\_PIN V17 IOSTANDARD LVCMOS33} [get\_ports sw[0]]

set\_property -dict {PACKAGE\_PIN V16 IOSTANDARD LVCMOS33} [get\_ports sw[1]]

set\_property -dict {PACKAGE\_PIN W16 IOSTANDARD LVCMOS33} [get\_ports sw[2]]

set\_property -dict {PACKAGE\_PIN W17 IOSTANDARD LVCMOS33} [get\_ports sw[3]]

set\_property -dict {PACKAGE\_PIN W15 IOSTANDARD LVCMOS33} [get\_ports sw[4]]

set\_property -dict {PACKAGE\_PIN V15 IOSTANDARD LVCMOS33} [get\_ports sw[5]]

set\_property -dict {PACKAGE\_PIN W14 IOSTANDARD LVCMOS33} [get\_ports sw[6]]

set\_property -dict {PACKAGE\_PIN W13 IOSTANDARD LVCMOS33} [get\_ports sw[7]]

#7 segment display

set\_property -dict {PACKAGE\_PIN W7 IOSTANDARD LVCMOS33} [get\_ports seg[0]]

set\_property -dict {PACKAGE\_PIN W6 IOSTANDARD LVCMOS33} [get\_ports seg[1]]

set\_property -dict {PACKAGE\_PIN U8 IOSTANDARD LVCMOS33} [get\_ports seg[2]]

set\_property -dict {PACKAGE\_PIN V8 IOSTANDARD LVCMOS33} [get\_ports seg[3]]

set\_property -dict {PACKAGE\_PIN U5 IOSTANDARD LVCMOS33} [get\_ports seg[4]]

set\_property -dict {PACKAGE\_PIN V5 IOSTANDARD LVCMOS33} [get\_ports seg[5]]

set\_property -dict {PACKAGE\_PIN U7 IOSTANDARD LVCMOS33} [get\_ports seg[6]]

set\_property -dict {PACKAGE\_PIN V7 IOSTANDARD LVCMOS33} [get\_ports dp]

set\_property -dict {PACKAGE\_PIN U2 IOSTANDARD LVCMOS33} [get\_ports an[0]]

set\_property -dict {PACKAGE\_PIN U4 IOSTANDARD LVCMOS33} [get\_ports an[1]]

set\_property -dict {PACKAGE\_PIN V4 IOSTANDARD LVCMOS33} [get\_ports an[2]]

set\_property -dict {PACKAGE\_PIN W4 IOSTANDARD LVCMOS33} [get\_ports an[3]]

## Run Synthesis

In the Flow Navigator pane on the left, the Synthesis option allows the user to create and launch synthesis design runs. To run synthesis, do the following:

* Click **Run Synthesis**. If all goes well, you should not get any errors.

## Run Implementation

In the Flow Navigator pane, the Implementation option allows the user to create and launch synthesis design runs. To run implementation, do the following:

* Click **Run Implementation**. If all goes well, you should not get any errors.

## Program and Debug

In the Flow Navigator pane, the Program and Debug option allows the user to generate the bitstream file which will be used to program the hardware. To generate bitstream and program the board, do the following:

* Expand PROGRAM AND DEBUG

### Step 1

* Click **Generate Bitstream**.
* If this step completes without error, move to Step 2.

### Step 2

* Expand Open Hardware Manager and click **Open Target**.
  + This opens a popup window
* In the pop window, Click **Auto Connect**.
  + The connected Xilinx board will/should be detected.
* Right Click on the board and select **Program Device.**
  + A program Device popup window will appear
* Navigate to the location of the Bitstream File and select it.

Please note:

* + If the Keil generated ***code.hex*** was present during synthesis, Implementation and Generate Bitstream runs, then select the bitstream file named ***AHBLITE\_SYS.bit*** which is located at Implementation Run folder.
  + If you ran the Update Bitstream script provided, the bitstream file generated is named ***reflash.bit.*** Select this file, which should be in the same directory as the Update Bitstream script.
* Click **Program** to program the FPGA Device.