Getting Started with Vivado and MicroBlaze

Prerequisites

Prior to starting this guide make sure to install Vivado. For more information, see Lab 0. Be sure to go through all the instructions in Lab 0, including installing Vivado board files for the Basys3.

Introduction

The goal of this guide is to familiarize the reader with the Vivado tools through the hello world of hardware, blinking an LED.

This guide was created using Vivado 2017.4.

1. Starting Vivado

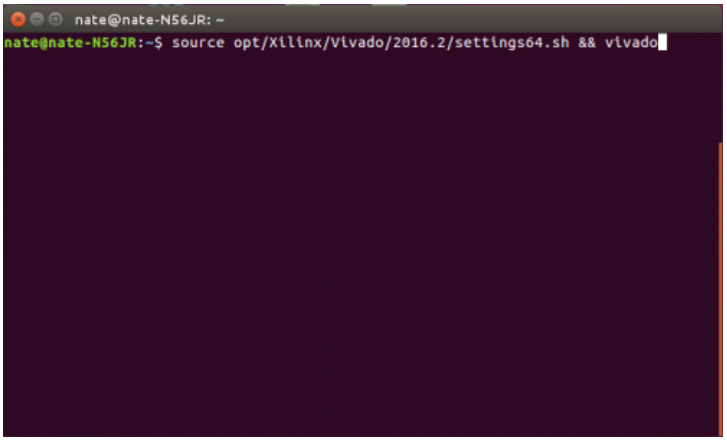
Windows

Open the Vivado software. Make sure you open the standard Vivado software and NOT the HLS. Icon should like below:

Linux:

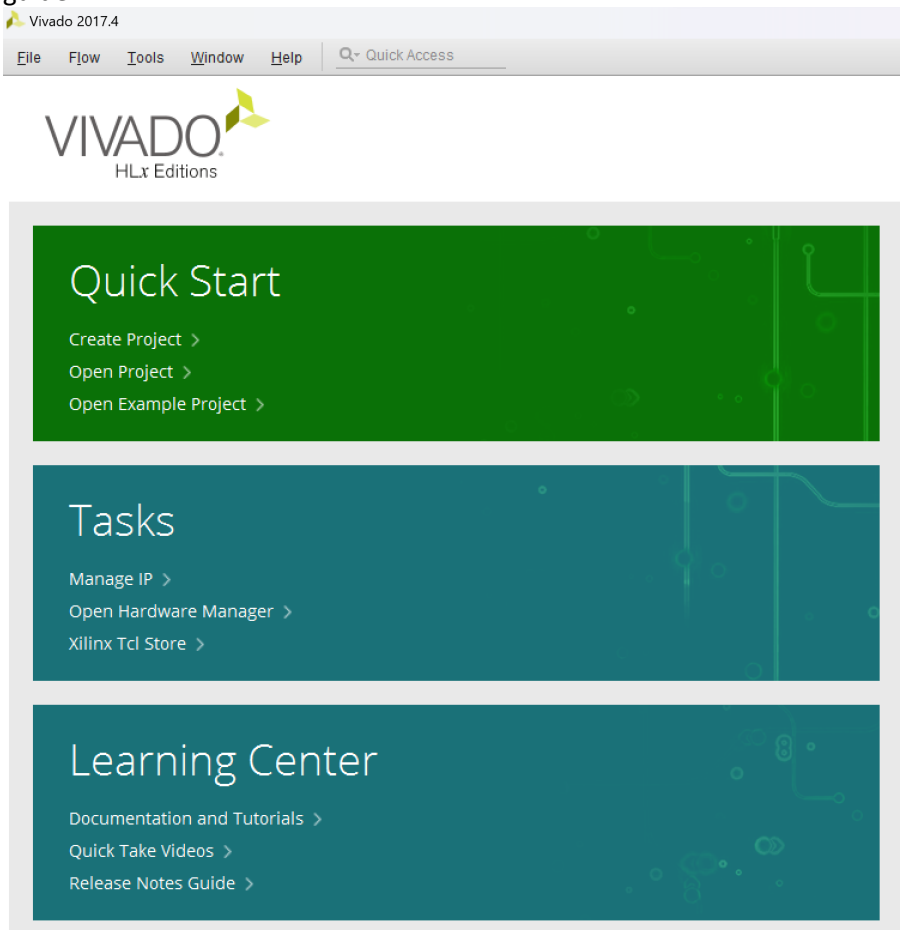
Open a Terminal and run

```
>> source <install_path>/Vivado/<version>/settings64.sh && vivado
```



2. The Start Page

This is the screen that displays after Vivado starts up. The buttons are described below using the image as a guide.



2.1. Create New Project: This button will open the New Project wizard. This wizard steps the user through creating a new project. The wizard is stepped through in section 3.

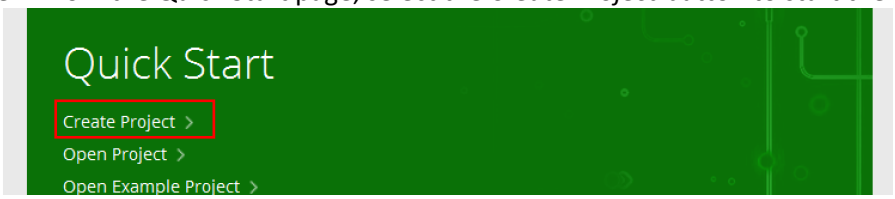
2.2. Open Project: This button will open a file browser. Navigate to the desired XPR file and click *Open* to open the project in Vivado.

2.3. Open Example Project: This will guide the user through creating a new project based on an example project. These projects will not work on all devices.

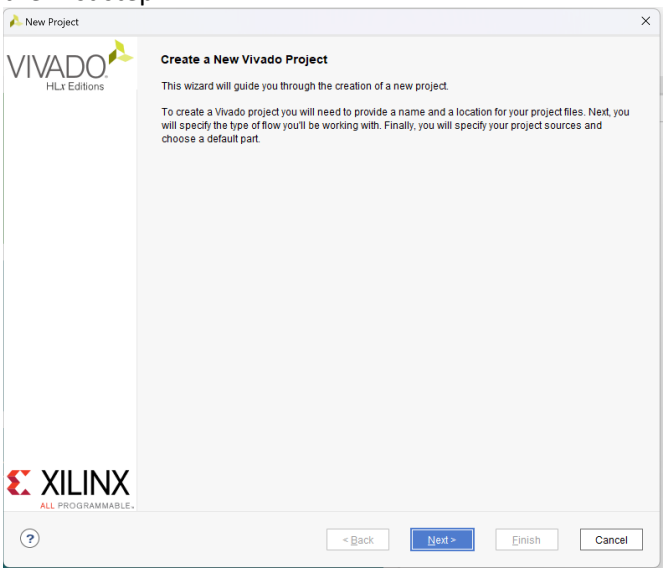
2.4. Open Hardware Manager: This will open the Hardware Manager without an associated project. If connecting to and programming a device is all that is required by the user this is the button to use.

3. Creating a New Project

3.1 From the Quick Start page, select the *Create Project* button to start the New Project Wizard.

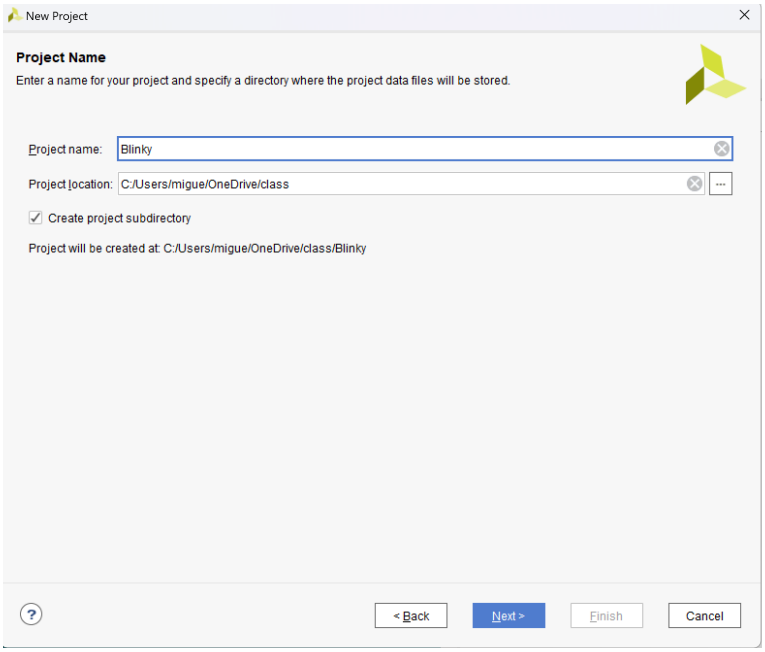


3.2 The text in the dialog describes the steps that will be taken to create a project. Click *Next* to continue to the first step.



3.3 The first step is to set the name of the project. Vivado will use this name when generating its folder structure.

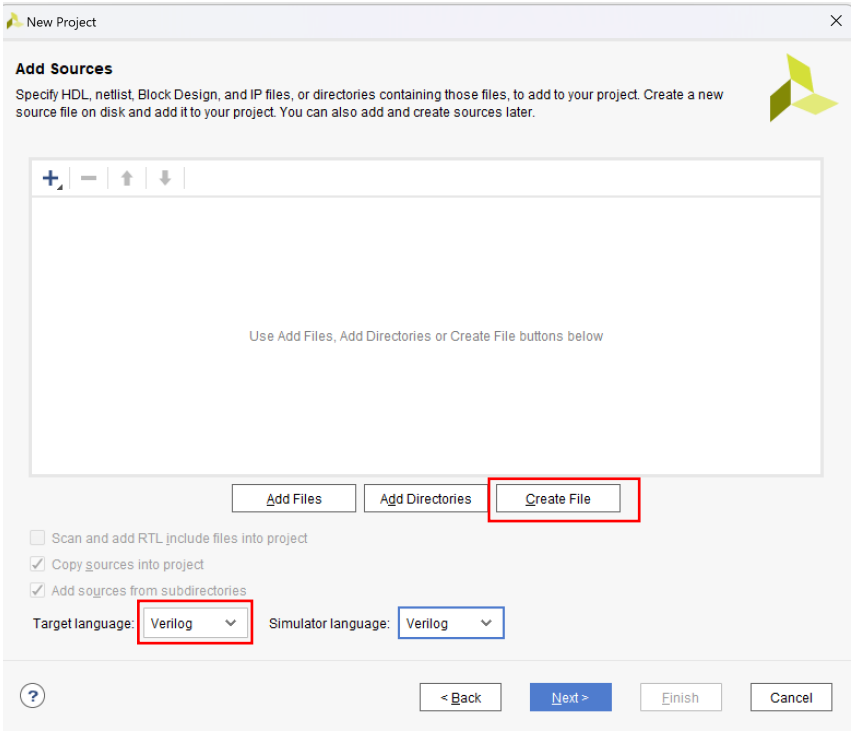
Important: Do NOT use spaces in your project name or location path. This will cause problems with Vivado. Instead use an underscore, a dash, or CamelCase.



3.4

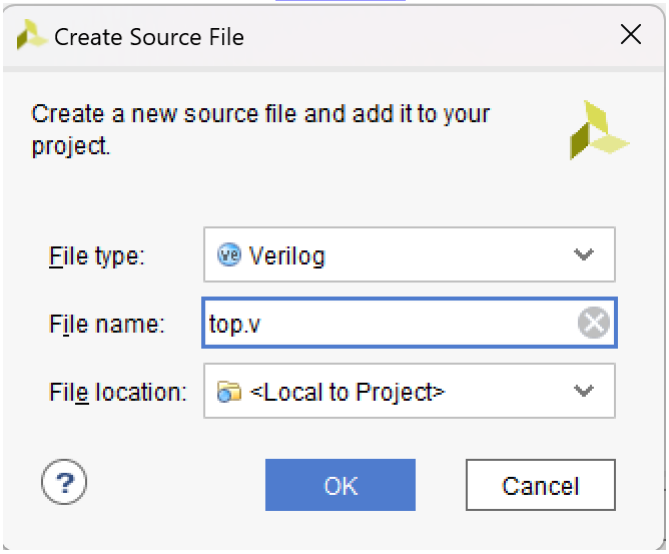
Now that the project has a name and a place to save important files, click ahead on **next** to create project type of **RTL Project.**

We need to create the necessary source files. To add a Verilog or VHDL file click the create file button.

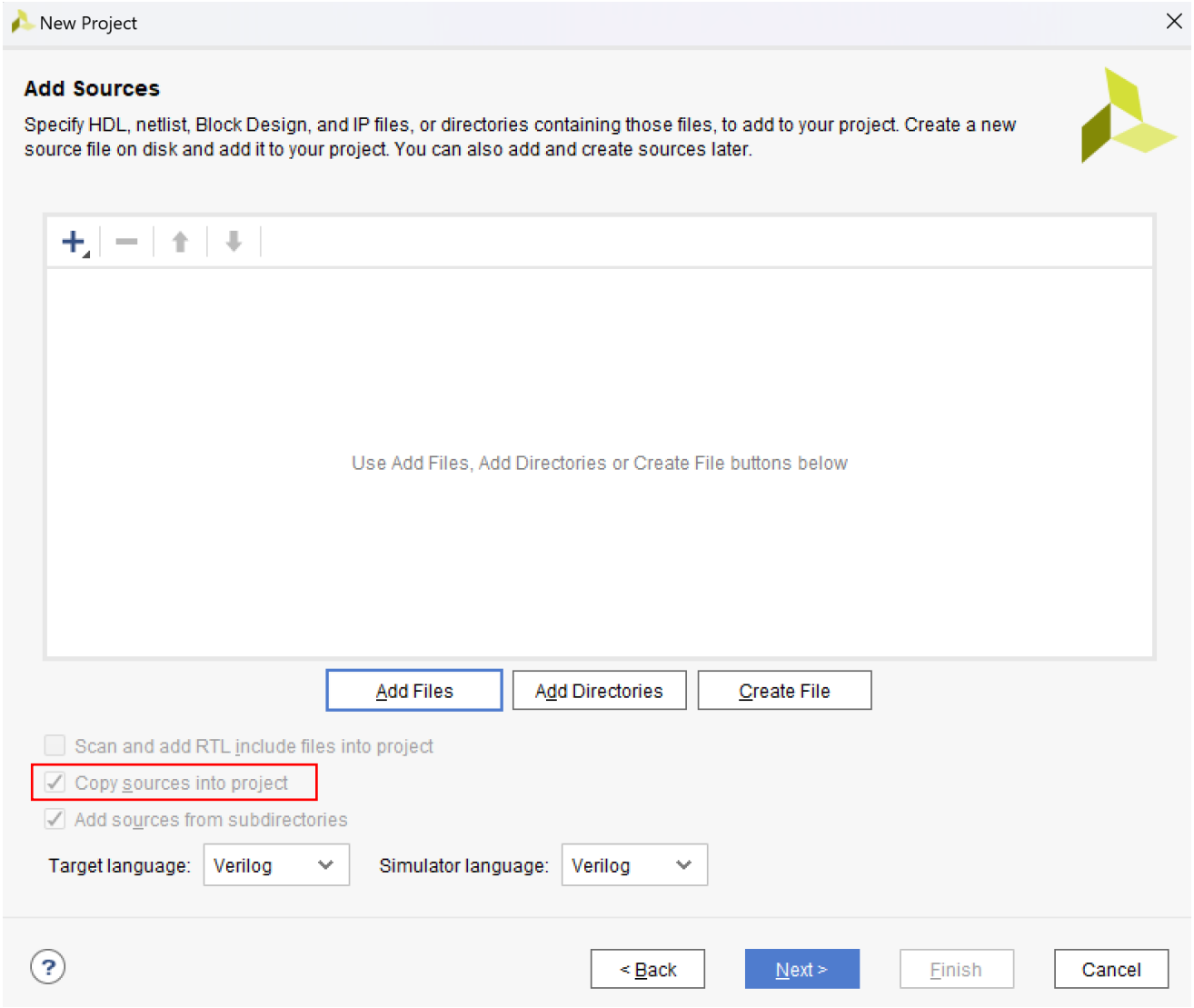


3.5 This will open the *Create Source File* dialog, shown below. This dialog has three options for the new file. The first is the type of file. This allows for selecting between different HDL file types, Verilog, VHDL, Verilog Header, and SystemVerilog. The final type available is the Memory File option. This is used for memory device initialization, most of the time this can be ignored.

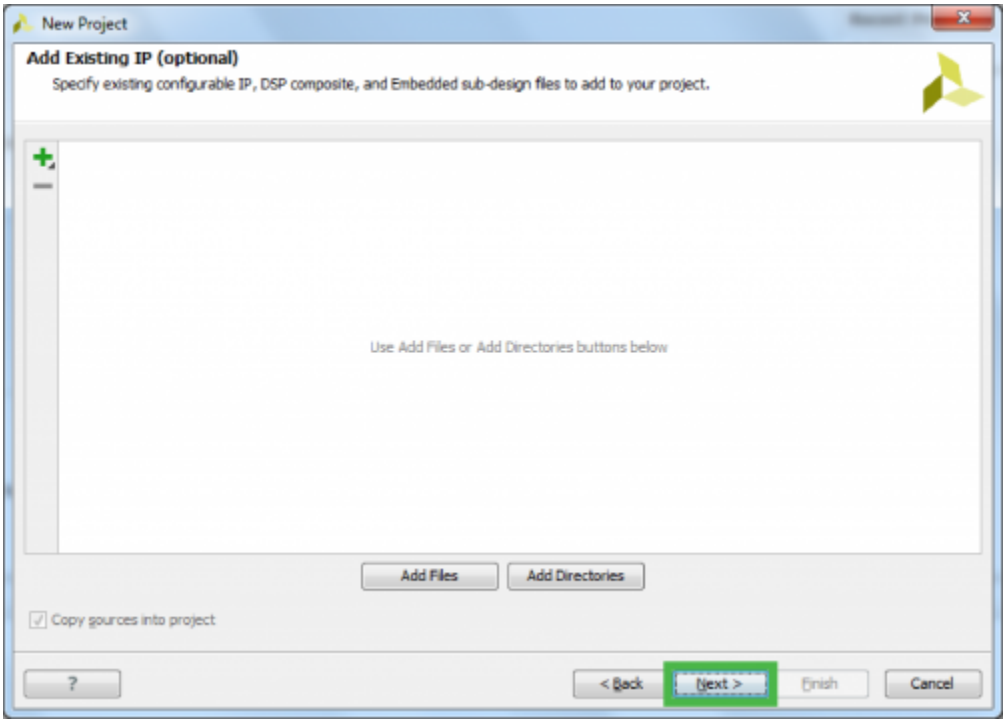
The second box is for the file name. Enter the name for the file to create. Including the file extension is optional, Vivado will add one based on the file type selected. The final option is where to store the file. Most of the time this can be left as *<Local to Project>*.
Important: Do NOT use spaces in your file name. This will cause problems with Vivado. Instead use an underscore, a dash, or CamelCase.



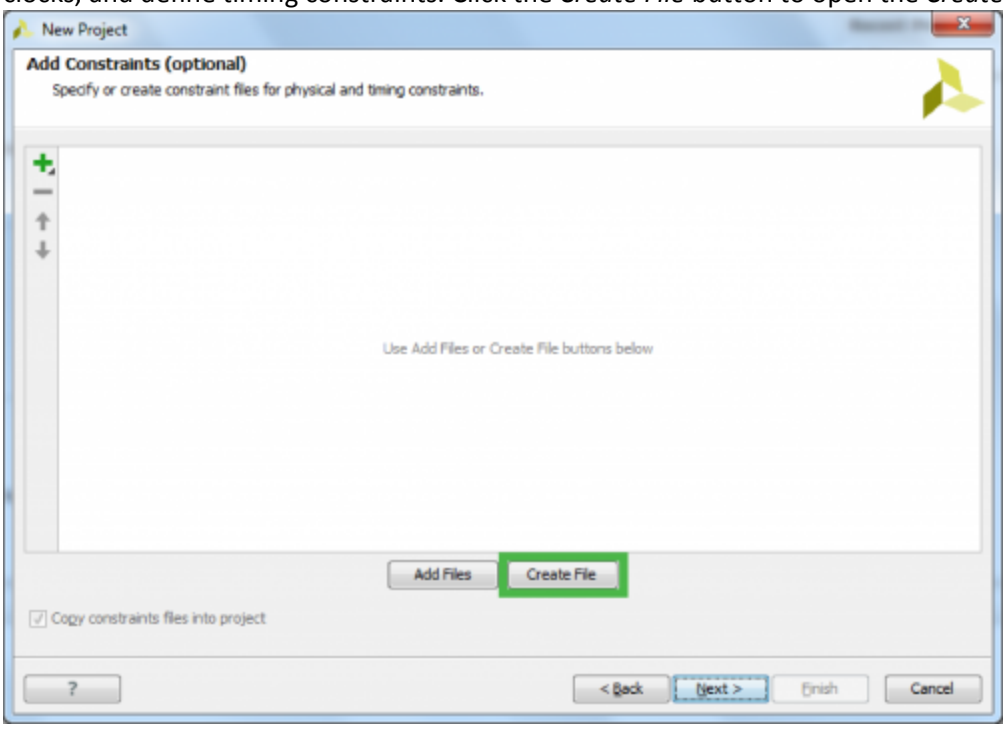
3.6 It is also possible to import source files into the project. In the *Add Sources* dialog click the *Add Files* button and navigate to the source file to import into the project. Unless there is reason not to, make sure that the *Copy sources into project* check box is checked to make a copy of the source file local to the project. Since the file is copied, any changes made to the file inside one project do not affect the original file.



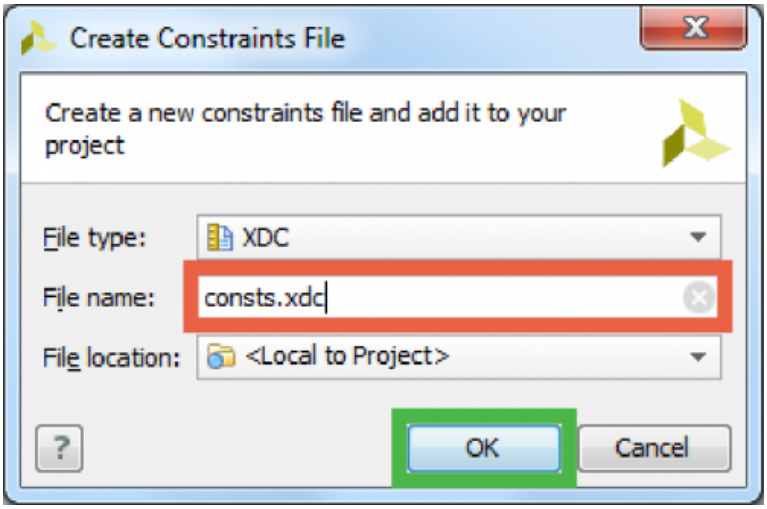
3.7 After adding the HDL files and clicking *Next* in the main dialog, the next dialog is displayed. This dialog is for adding IP to the project. This is used in more advanced designs and is outside the scope of this guide. Click *Next*.



3.8 The next section is for adding a constraints file. Constraints files control how signals are routed, define clocks, and define timing constraints. Click the *Create File* button to open the *Create Constraints File* dialog.

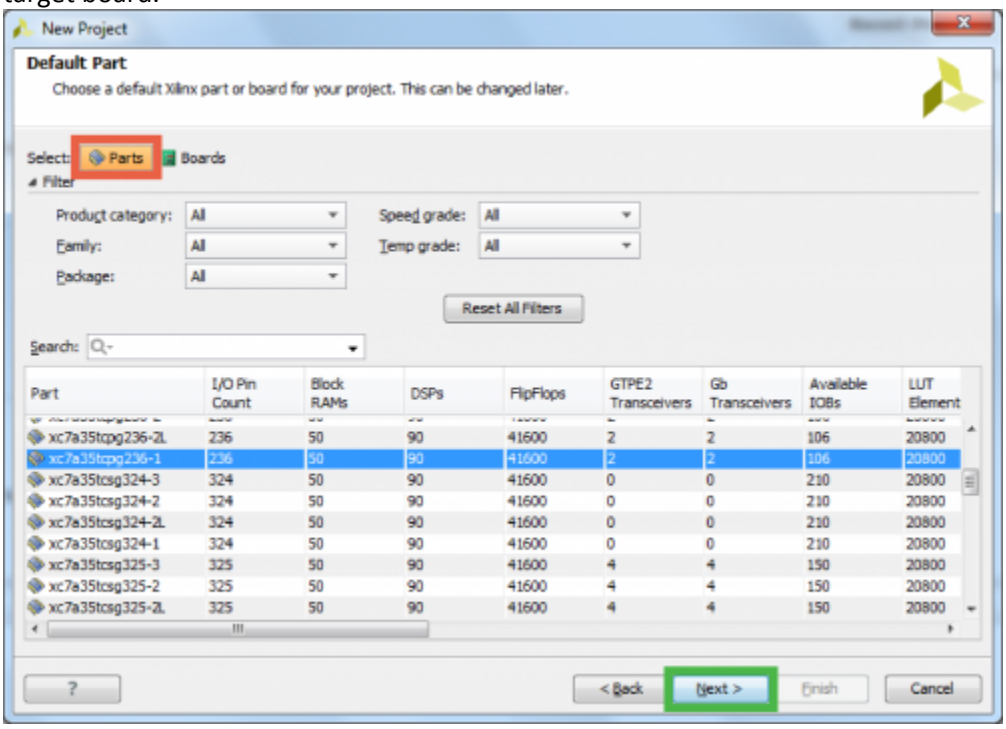


3.9 Again the dialog presents three options. the file name and file location have the same properties as the ones in the *Create Source File* dialog. What is different is the file types that can be created. The only option available is XDC which stands for Xilinx Design Constraints. This file will define clocking, pin mapping, and timing restrictions.
Important: Do NOT use spaces in your file name. This will cause problems with Vivado. Instead use an underscore, a dash, or CamelCase.

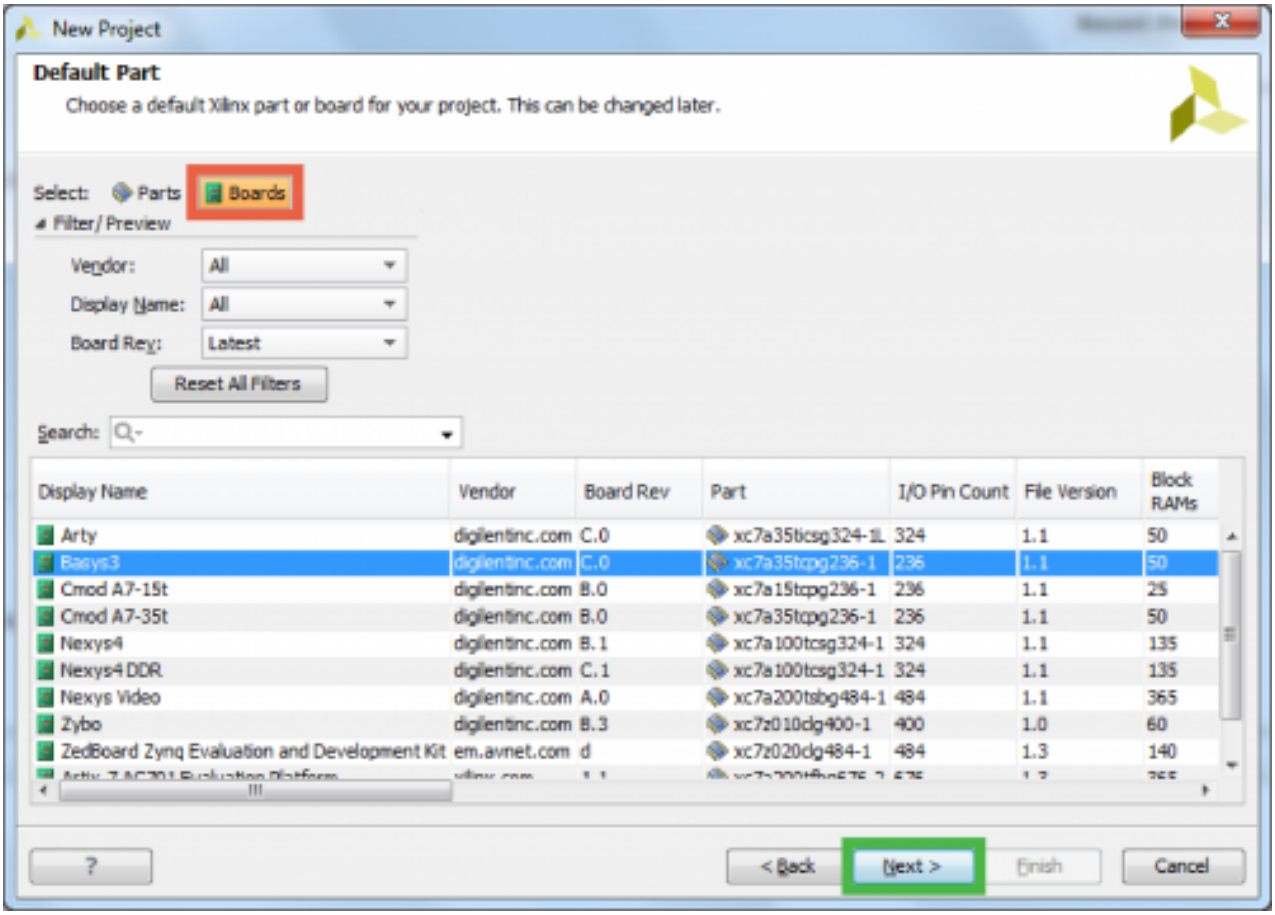


3.10 Now that the necessary files have been created it is time to choose the target device. There are two ways to choose a target. The first is from the actual on board component and the second is from installing the appropriate board files and picking a board.

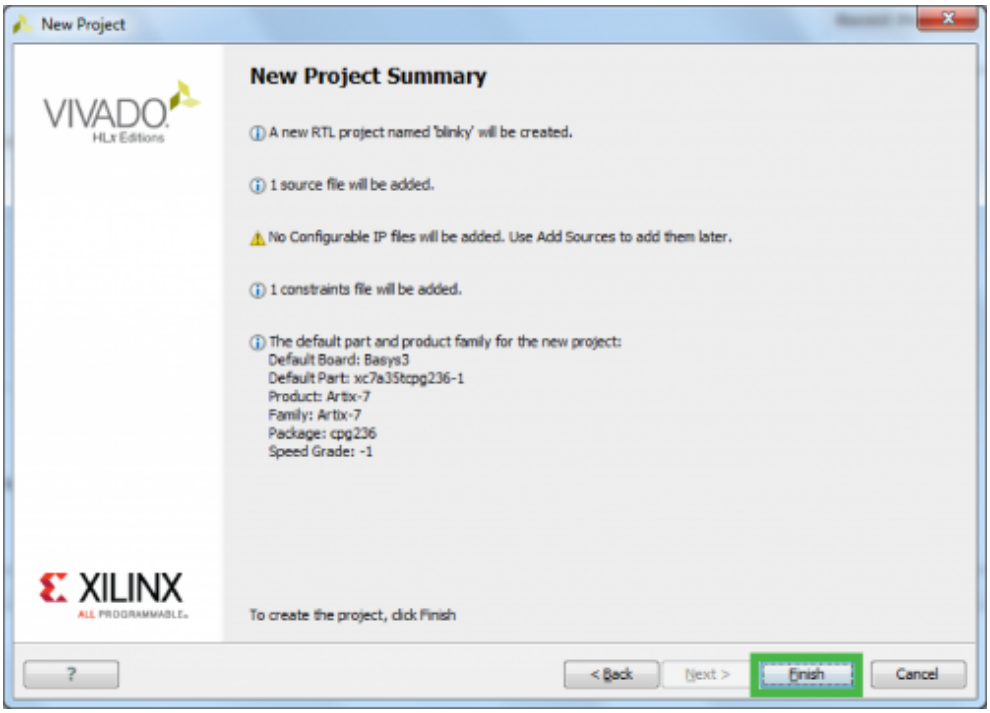
3.10.1 To pick a target from the on board component find the part number for the IC and find it in the selection list. There are two places to find a part number for our boards. The first is to look at the actual component on the board and try to discern the writing. The other is to look on the reference page for the target board.



3.10.2: To pick a target from the boards list, install the board files prior to creating a new project. Then select the board to use as a target from the list.



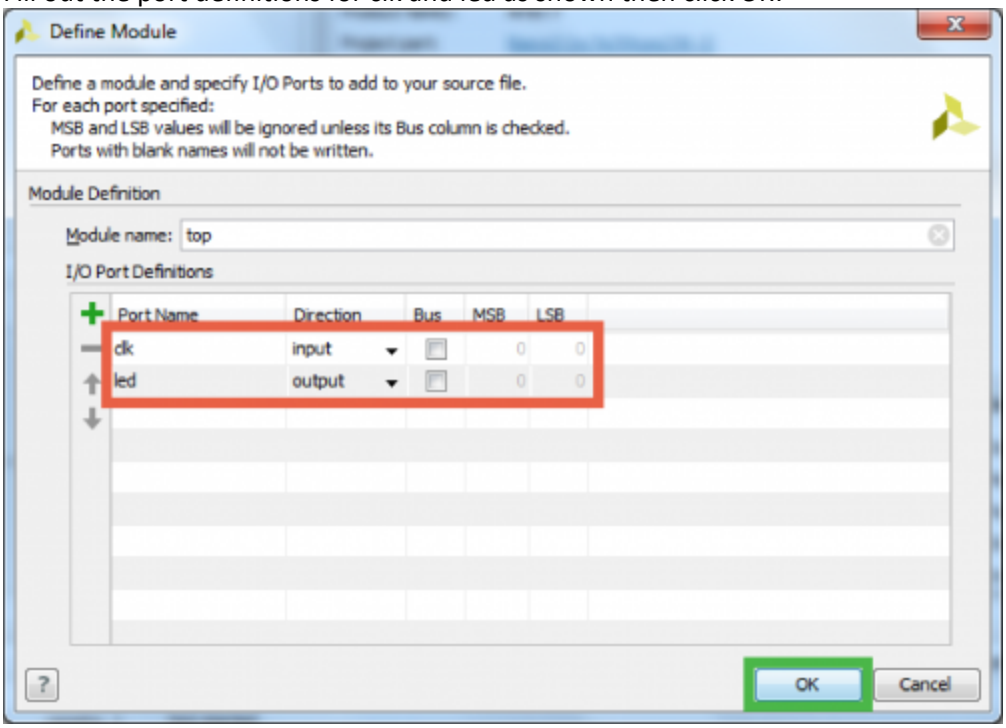
3.11: The next section gives a summary of the options selected throughout the wizard. Verify that the information looks correct and click *Finish*.

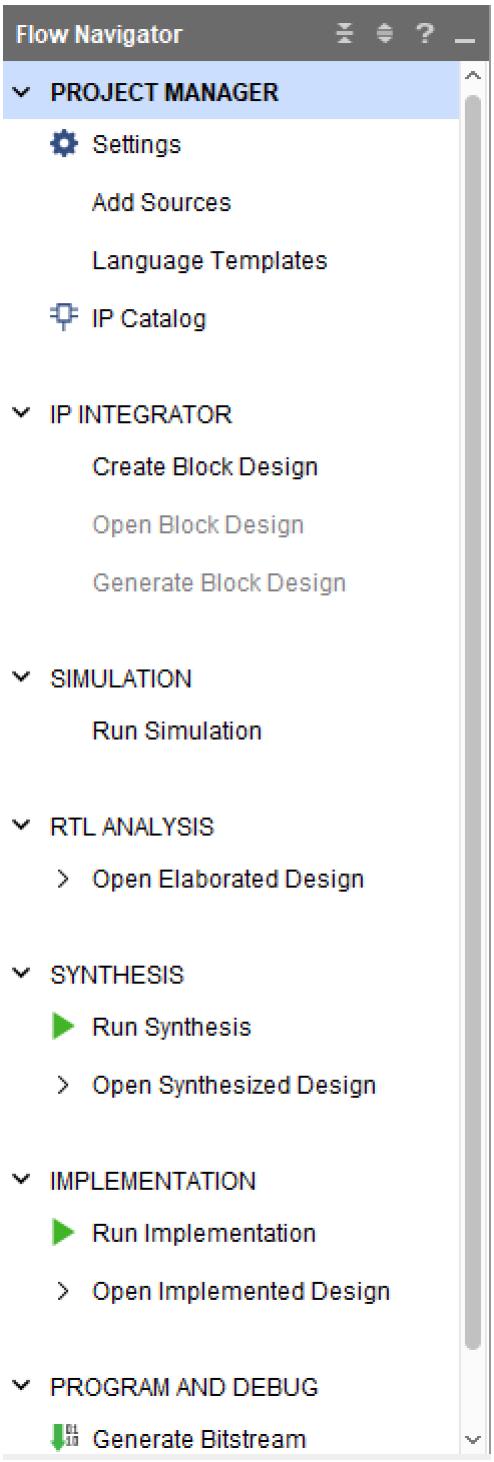


3.12: Since new source files were created, Vivado will now show a new dialog box for defining a module for each new source file that it was told to create. Here the inputs and outputs of a module can be defined. Each section can be interpreted as follows:

Field	Description	Example
Port Name	The name to give the new port	*leds*
Direction	Is the port an input, output, or bi-directional	*output*
Bus	Is the port a bus / multiple bits?	*x*
MSB	What is the Most Significant Bit?	*7*
LSB	What is the Least Significant Bit?	*0*

Fill out the port definitions for clk and led as shown then click *OK*.





4. The Flow Navigator

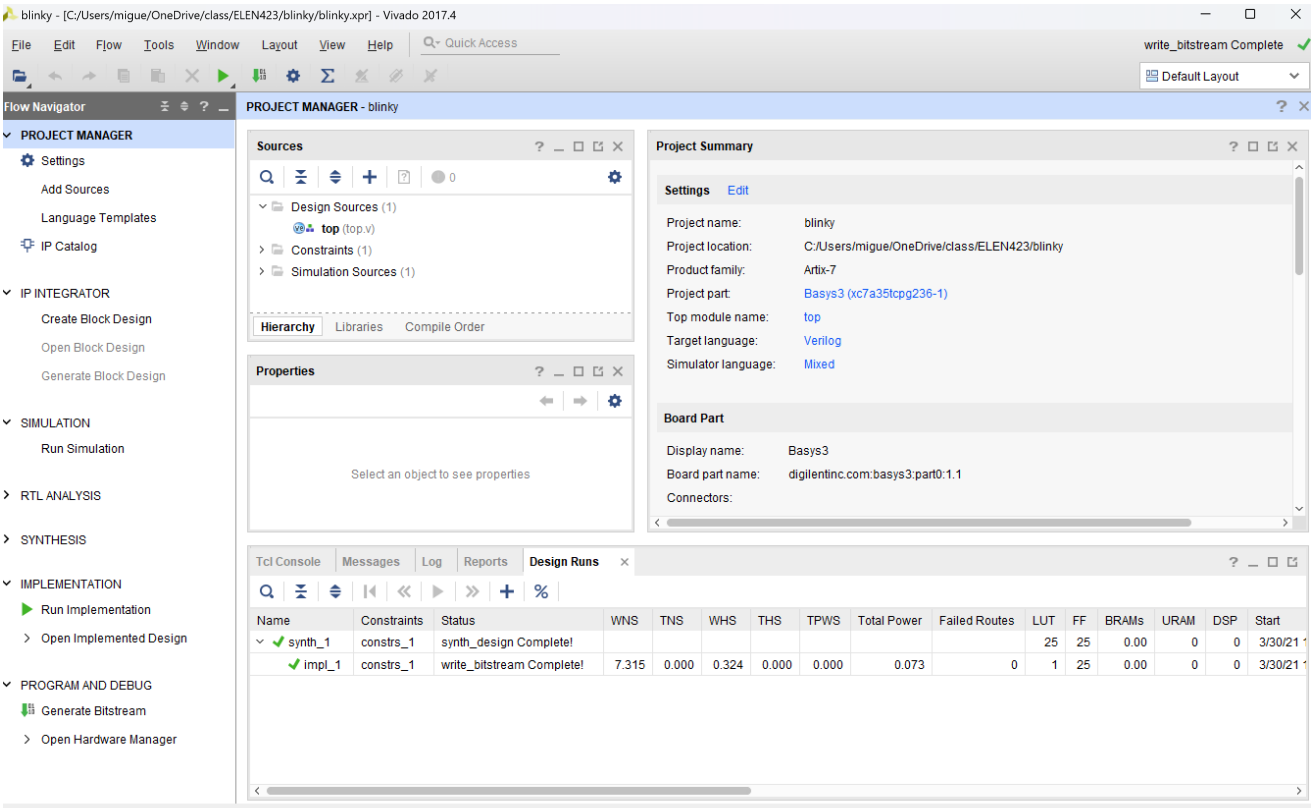
The Flow Navigator is the most important pane to know. It is how a user navigates between different Vivado tools.

The Navigator is broken into seven sections:

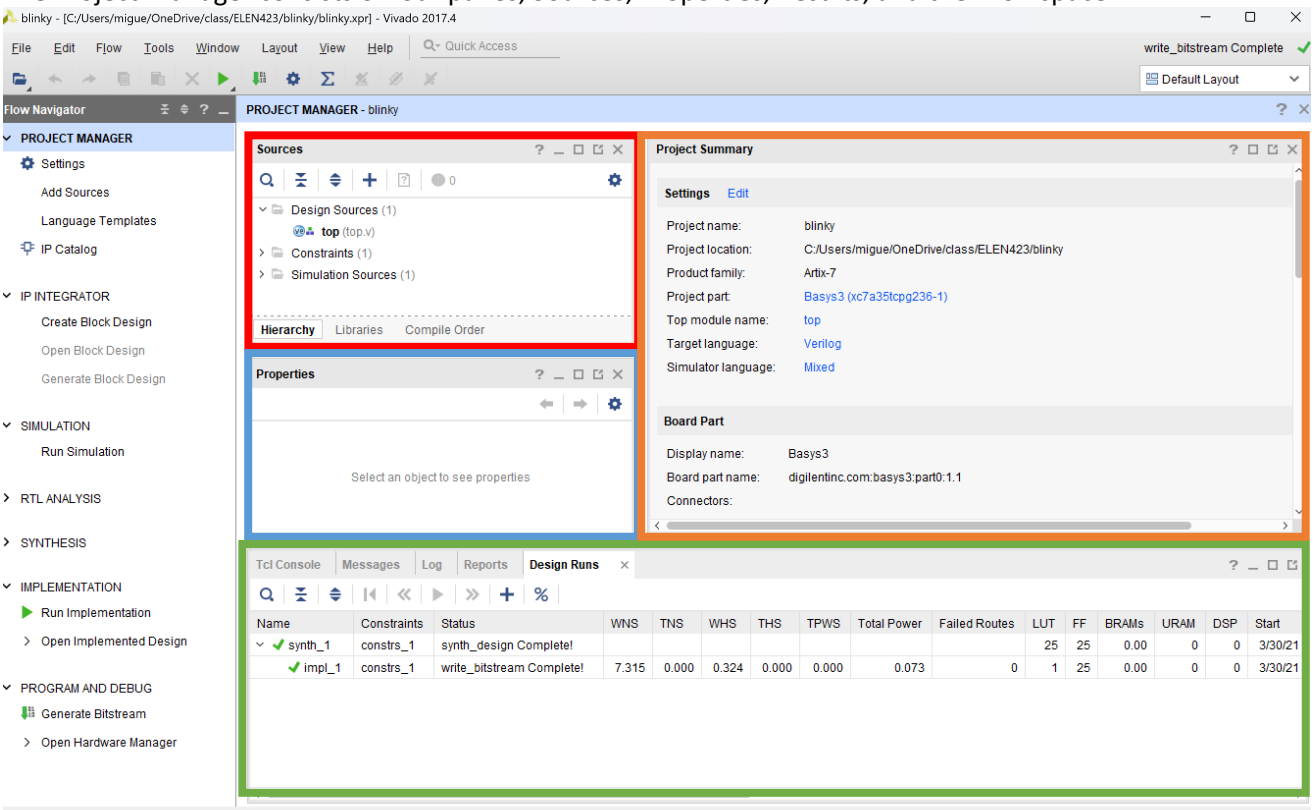
- Project Manager
  - Allows for quick access to project settings, adding sources, language templates, and the IP catalog
- IP Integrator
  - Tools for creating Block Designs
- Simulator
  - Allows a developer to verify the output prior to programming their device
- RTL Analysis
  - lets the developer see how the tools are interpreting their code
- Synthesis
  - Gives access to Synthesis settings and post-synthesis reports
- Implementation
  - Gives access to Implementation settings and post-implementation reports
- Program and Debug
  - Access to settings for bitstream generation and the Hardware Manager

5. The Project Manager

This tool is where most development will occur and is the initial tool open after creating a new project.

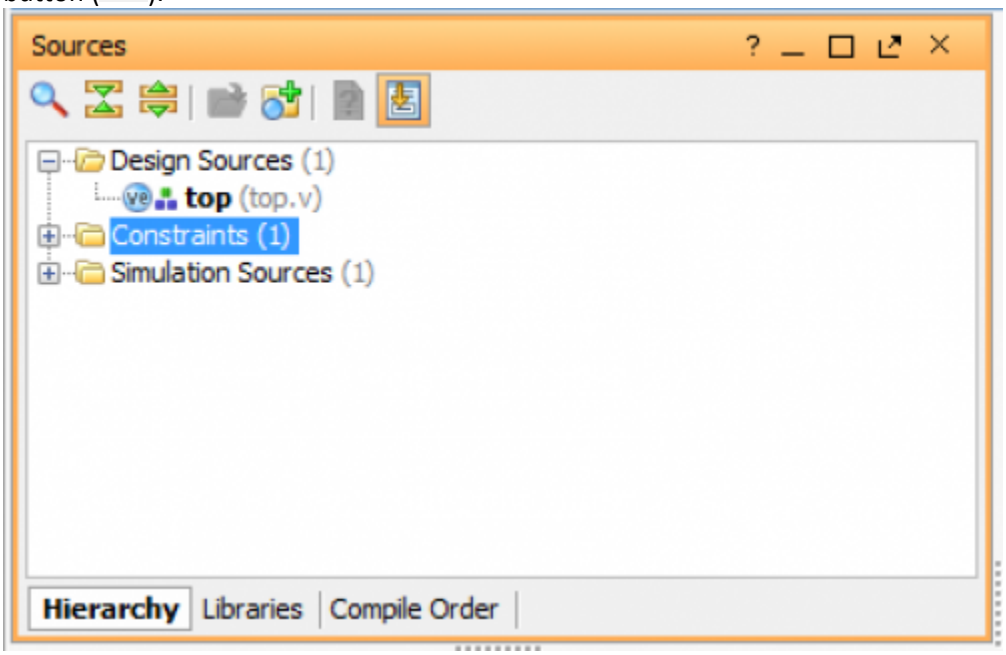


The Project Manager consists of four panes, Sources, Properties, Results, and the Workspace.

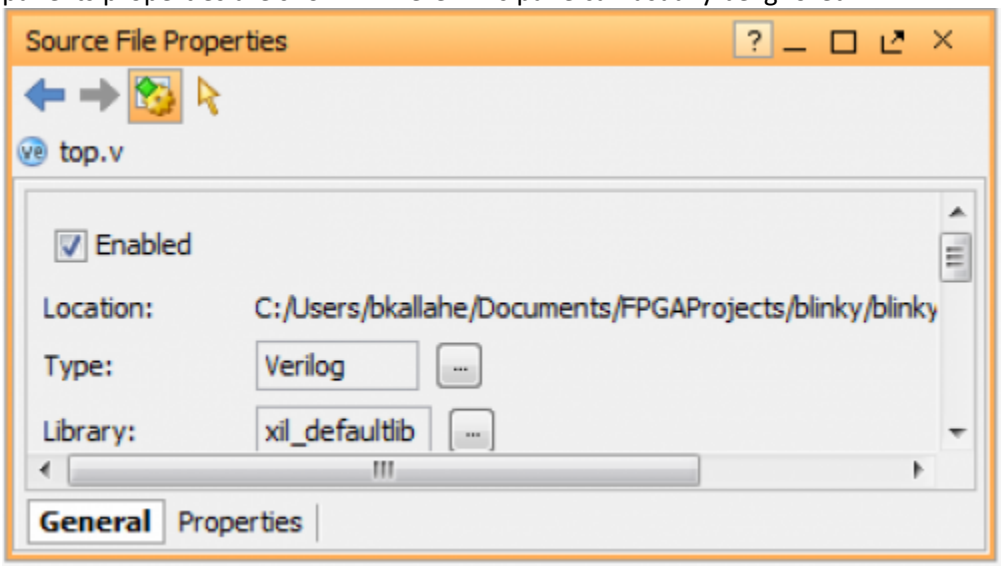


The Sources pane contains the project hierarchy and is used for opening up files. The folder structure is organized such that the HDL files are kept under the *Design Sources* folder, constraints are kept under the

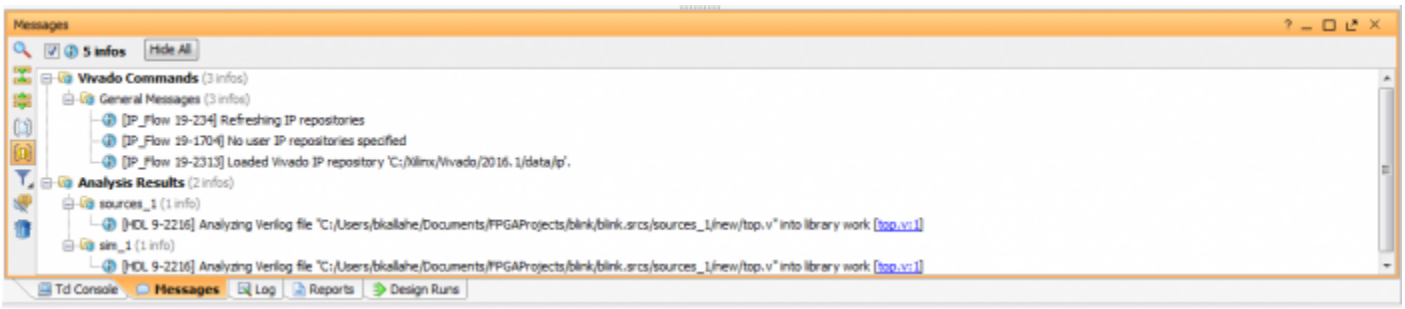
Constraints folder, and simulation files are kept under the *Simulation Sources* folder. Files can be opened in the Workspace by double-clicking on the corresponding entry in the Sources pane. Sources can also be added by either right clicking the folder to add the file to and selecting *Add Sources* or by clicking the *Add Sources* button ( ).



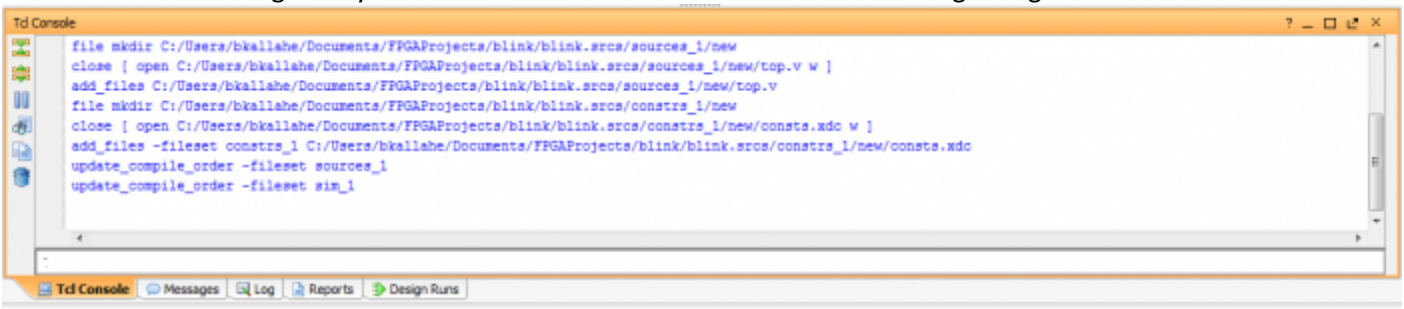
The Properties pane allows for viewing and editing of file properties. When a file is selected in the Sources pane its properties are shown in here. This pane can usually be ignored.



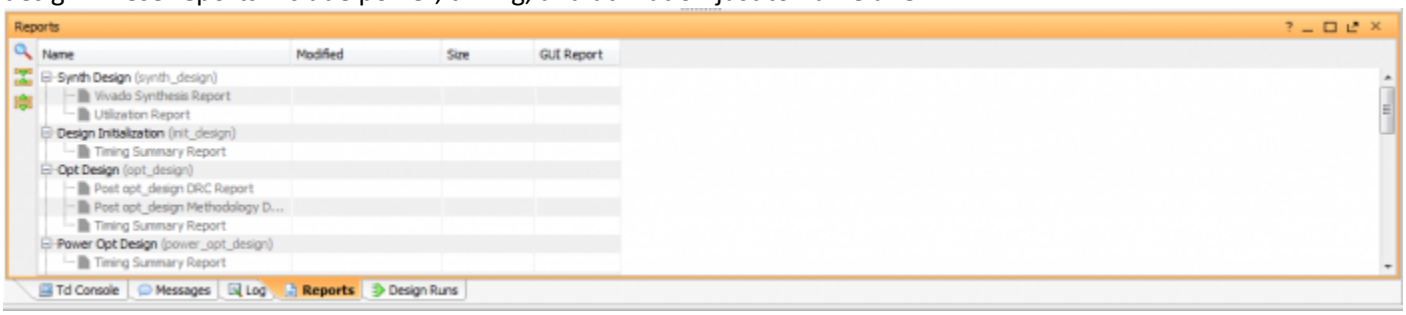
The Results pane consists of several different useful tools for debugging a project. The most important one to know is the Messages tool. This tool parses the Tcl console for errors, warnings, and other important information and displays it in an informative way.



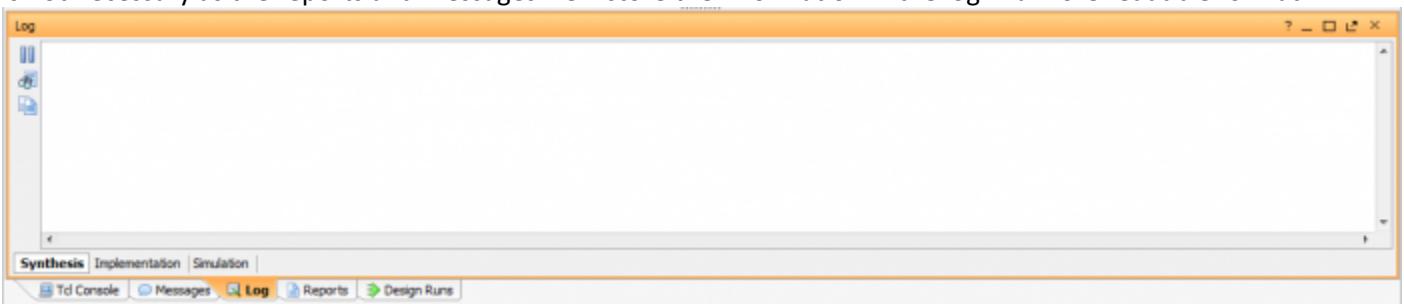
The Tcl console is a tool that allows for running commands directly without the use of the main user interface. Some messages may direct to the Tcl console for more information regarding an error.



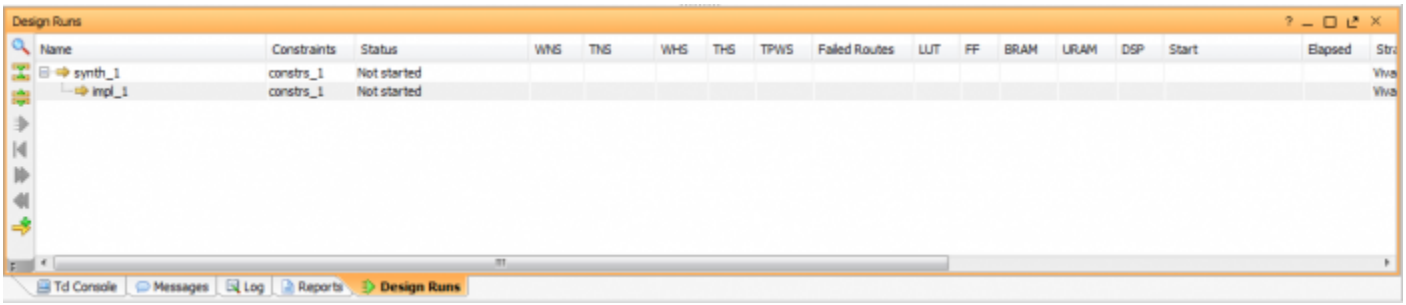
The Reports tool is useful for quickly jumping to any one of the many reports that Vivado generates on a design. These reports include power, timing, and utilization just to name a few.



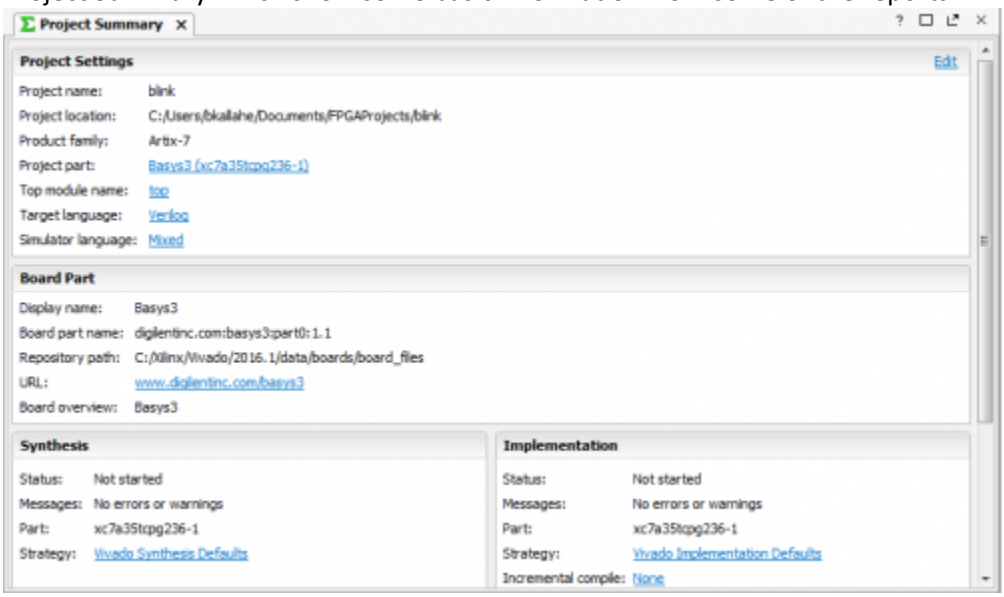
The Log displays the output from the latest Synthesis, Implementation, and Simulation runs. Digging into this is not necessary as the reports and messages view store the information in the log in a more readable format.



The last tool is the Design Runs. Using this tool run settings can be edited and new runs can be created. This tool is useful when targeting multiple devices with the same design.

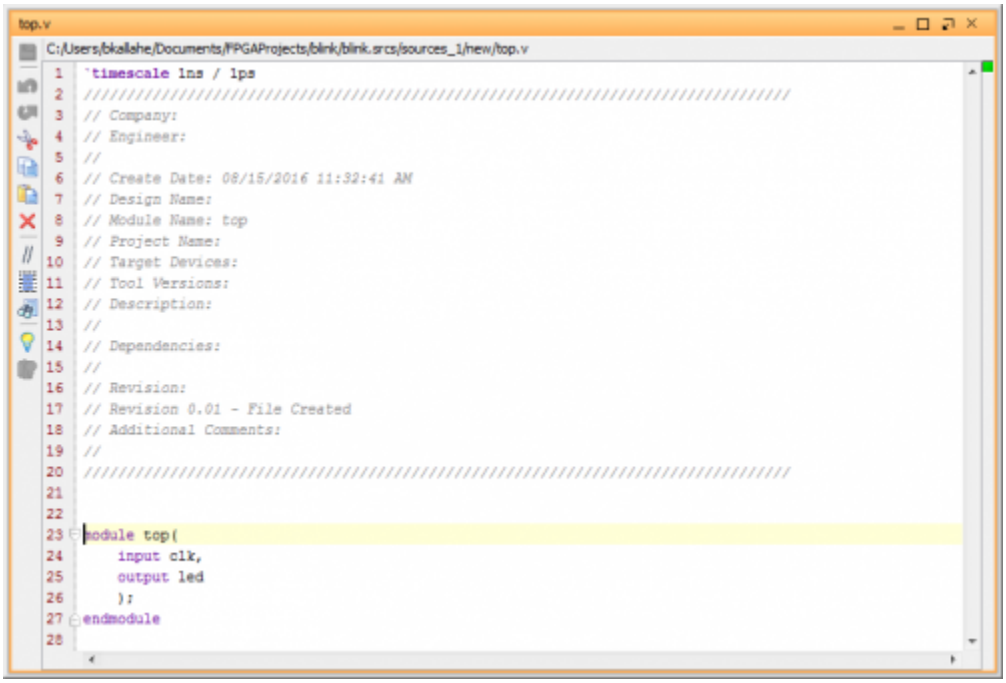


The most important pane in the Project Manager is the Workspace. The Workspace is where reports are opened for viewing and HDL/constraints files are opened for editing. Initially the Workspace displays the Project Summary which show some basic information from some of the reports.



6. Writing HDL and Constraints

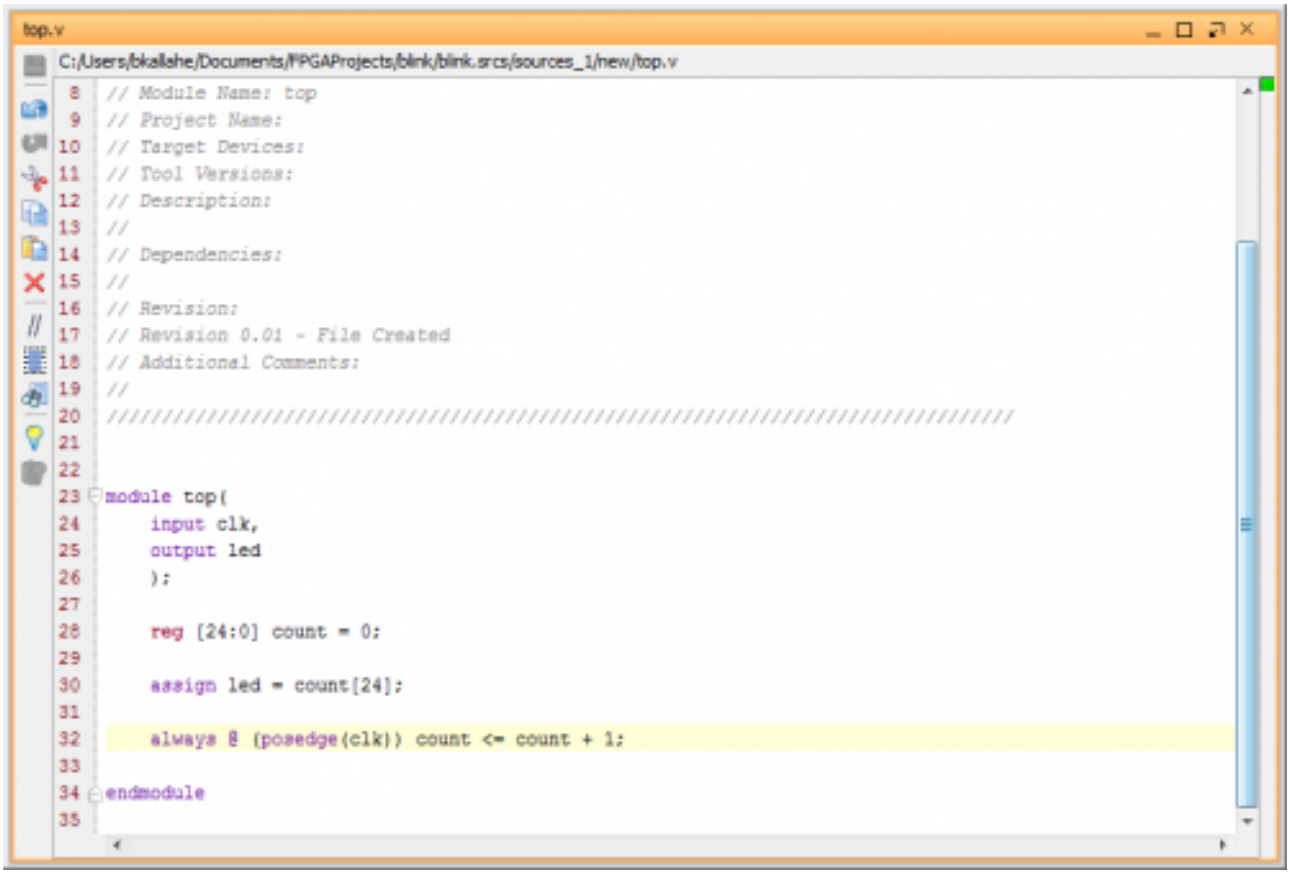
To actually get the board to blink an LED, code needs to be written in both the Verilog and the constraints files. Opening up the Verilog file (**top.v** in this guide) this window should open in the workspace area.



Between lines 26 and 27 add the following code:

```
reg [24:0] count = 0;

assign led = count[24];

always @ (posedge(clk)) count <= count + 1;
```

Once added the source file will look like this:

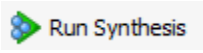


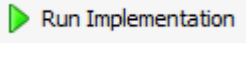
Now open the XDC file. At the moment it is empty so paste in the code below:

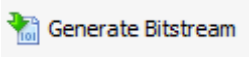
```
set_property -dict { PACKAGE_PIN W5   IOSTANDARD LVCMOS33 } [get_ports clk]
  create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
set_property -dict { PACKAGE_PIN U16  IOSTANDARD LVCMOS33 } [get_ports led]
```

Once that is done the code is ready to be “compiled.”

8. Synthesis, Implementation, and Bitstream Generation

To create a file that can be used to program the target, the “compilation-pipeline” needs to be done.
This starts with Synthesis. Synthesis turns HDL files into a transistor level description based on timing and I/O constraints. To run Synthesis click either  in the toolbar or 

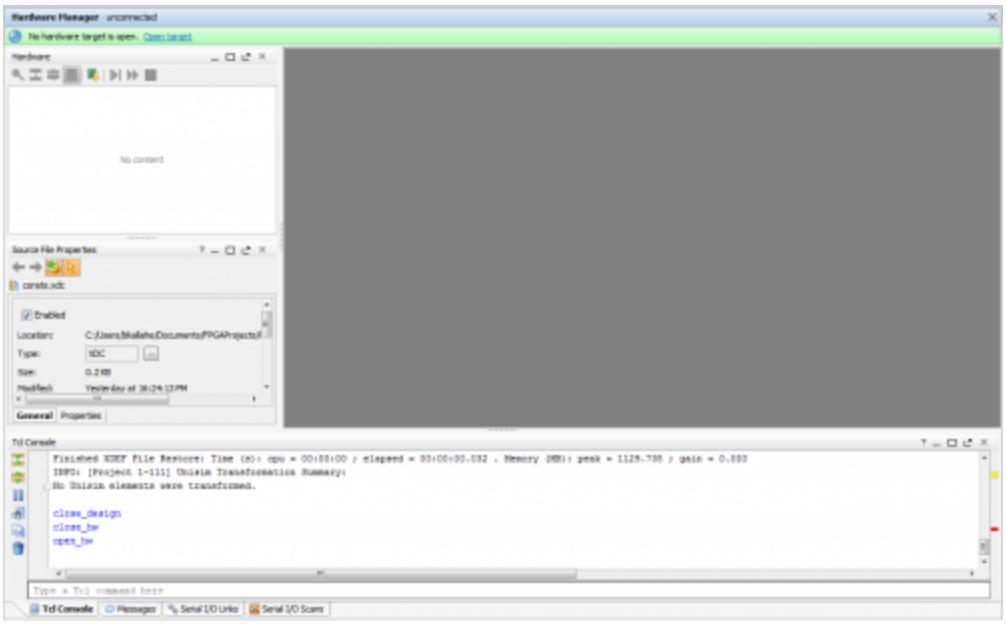
in the Flow Navigator. The output of Synthesis is then passed to Implementation.
Implementation has several steps. The steps that are always run are Opt Design (Optimize the design to fit on the target FPGA), Place Design (Layout the design in the target FPGA fabric), and Route Design (Route signals through the fabric). To run Implementation click either  in the toolbar or 

in the Flow Navigator. This output is then passed on to the Bitstream Generator.
The Bitstream Generator generates the final outputs needed for programming the FPGA. To run Bitstream Generation click either  in the toolbar or 

in the Flow Navigator. With no settings changed, the generator will create a BIT file.

9. The Hardware Manager

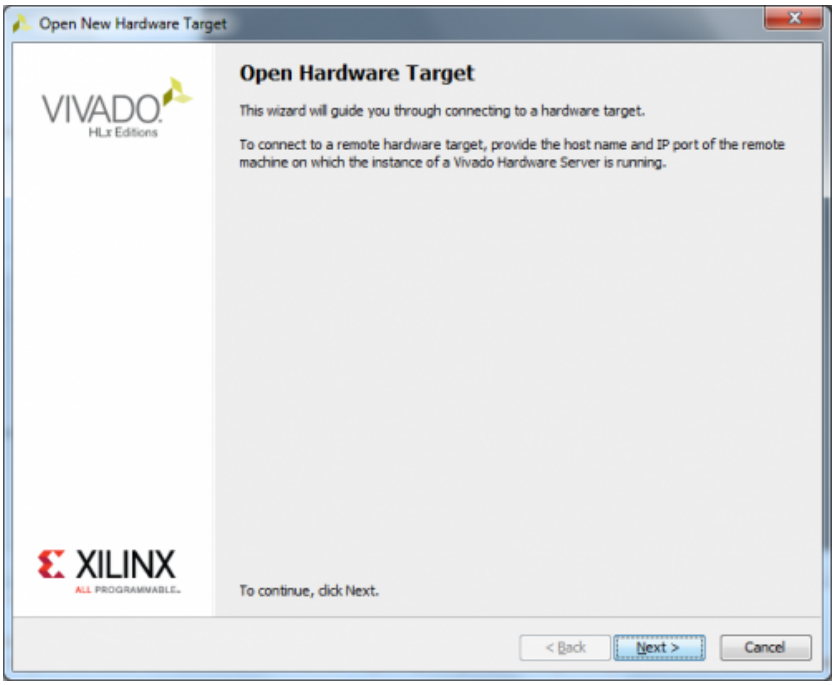
The Hardware Manager is used for programming the target device.



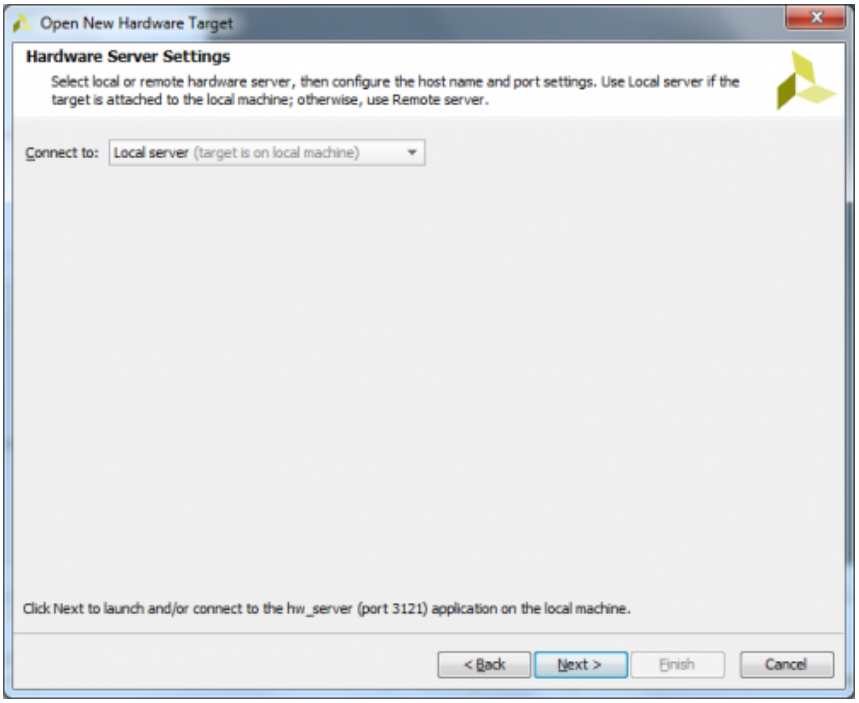
The first step to programming a device is to connect to it. There are two ways to do this.

9.1. Open New Hardware Target

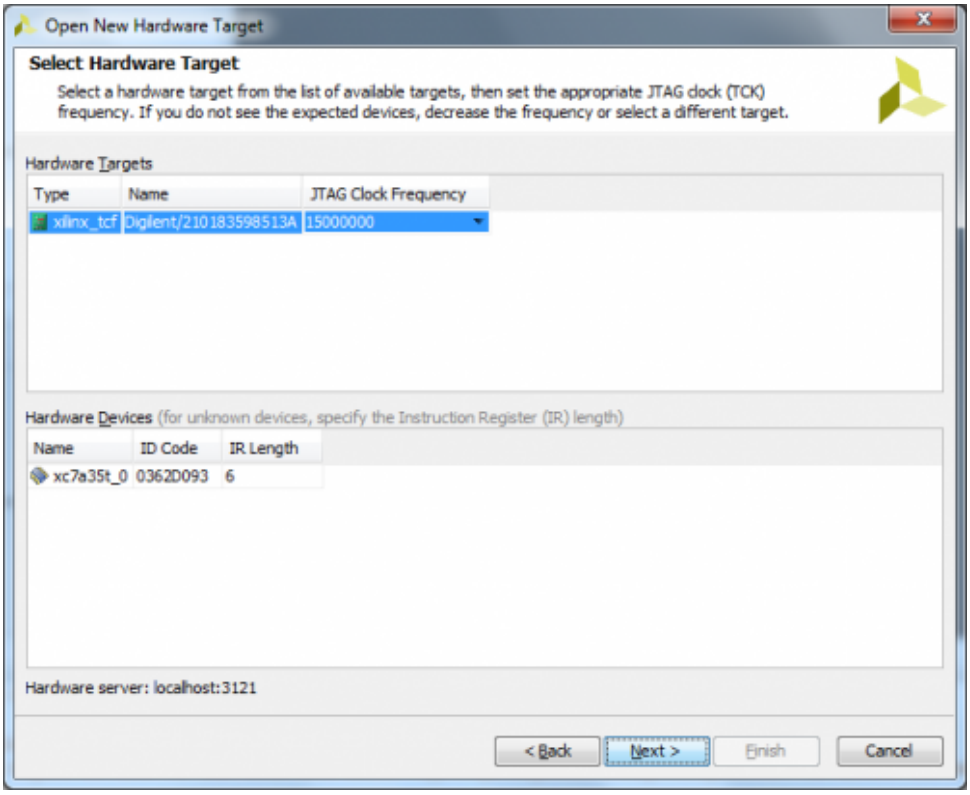
The first method is to manually open the target. This is required if the hardware is connected to another computer. To get to the Open Hardware Target wizard either open the Hardware Manager and click the Open target link in the green banner or click the Open Target button in the Flow Navigator under Hardware Manager. From the dropdown that opens select Open New Target... . The wizard will open, click *Next*.



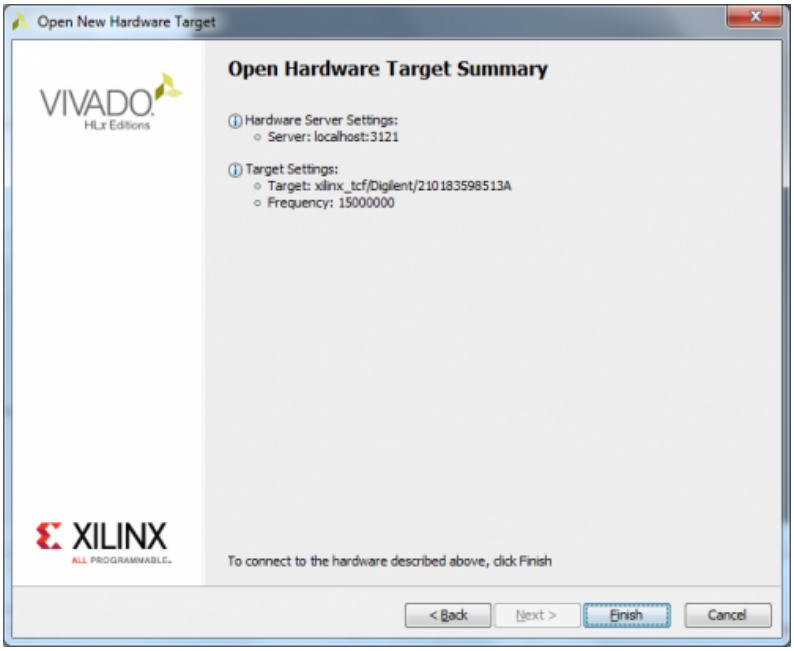
The next screen asks if the hardware server is local or remote. If the board is connected to the host computer choose local, if it is connected to another machine choose remote and fill in the *Host Name* and *Port* fields appropriately. Click *Next*.



This screen gives a list of devices connected to the hardware server. If there is only one connected it should be the only device shown. If there are multiple connected determine the serial number of the device to connect to and find it in the list. Click *Next*.



The final screen shows a summary of the options selected in the wizard. Verify the information and click *Finish*. The board is now connected to the hardware manager.

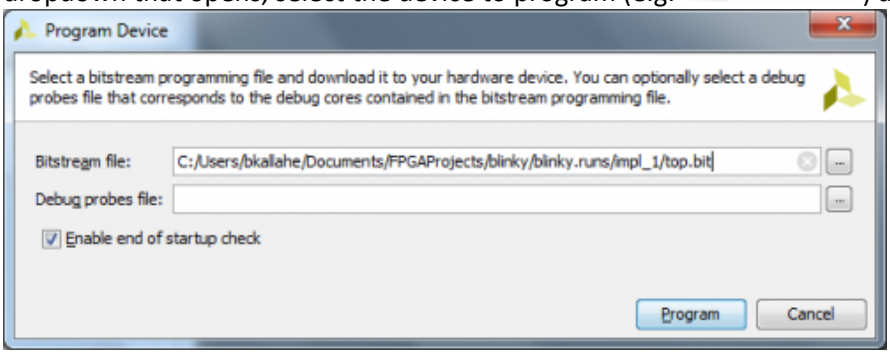


9.2. Auto-Connect

The second method is to automatically open the target. To get to the Auto Connect button either open the Hardware Manager and click the Open target link in the green banner or click the Open Target button in the Flow Navigator under Hardware Manager. From the dropdown that opens select Auto Connect. Vivado will attempt to find a hardware server running on the local machine and will connect to the device on the server.

9.3 Programming

To program the device with the BIT file generated earlier either click the Program device link in the green banner or click the Program Device button in the Flow Navigator under Hardware Manager. From the dropdown that opens, select the device to program (e.g. xc7a35t_0) and the following window opens:



The *Bitstream File* field should be filled in with the bit file generated earlier. If not, click the ... button at the right end of the field and navigate to *<Project Directory>/<Project Name>.runs/impl_1/* and select the bit file (Example: top.bit). Now click *Program*. This will connect to the board, clear the current configuration, and program using the new bit file.

10: To get checked off for the 1st part, raise your hand to call over the instructor and show your Basys3.

Basys 3 Getting started in Microblaze

Overview

This guide will provide a step by step walk-through of creating a Microblaze based hardware design using the Vivado IP Integrator for the Basys 3 FPGA board.

At the end of this tutorial you will have:

- Created a Microblaze based hardware (HW) design in Xilinx Vivado
- Created a .C Project in Xilinx Vivado SDK (Software Development Kit) to display Hello World using the hardware design shown in the previous step
- Displayed the final output on both the SDK console and Tera Term

---

Prerequisites

Hardware

- **Digilent Basys 3 FPGA Board and Micro USB Cable for UART communication and JTAG programming**

Software

- **Xilinx Vivado 2016.X with the SDK package.**

Board Support Files: *These files will describe GPIO interfaces on your board and make it easier to select your FPGA board and add GPIO IP blocks*

Microblaze is a soft IP core from Xilinx that will implement a microprocessor entirely within the Xilinx FPGA general purpose memory and logic fabric. For this tutorial, we are going to add a Microblaze IP block using the Vivado IP Integrator tool.

In addition to the Microblaze IP block, a UART (universal asynchronous receiver/transmitter) IP block will be added to communicate between the host PC and the soft processor core running on the Basys 3.

General Design Flow

I. Vivado

- Open Vivado and select Basys 3 board
- Create a new Vivado Project
- Create empty block design workspace inside the new project
- Add required IP blocks using the IP integrator tool and build Hardware Design
- Validate and save block design
- Create HDL system wrapper
- Run design Synthesis and Implementation
- Generate Bit File
- Export Hardware Design including the generated bit stream file to SDK tool
- Launch SDK

Now the Hardware design is exported to the SDK tool. The Vivado to SDK hand-off is done internally through Vivado. We will use SDK to create a Software application that will use the customized board interface data and FPGA hardware configuration by importing the hardware design information from Vivado.

II. SDK

- Create new application project and select default Hello World template
- Program FPGA
- Run configuration by selecting the correct UART COM Port and Baud Rate

---

11. Creating a New Project

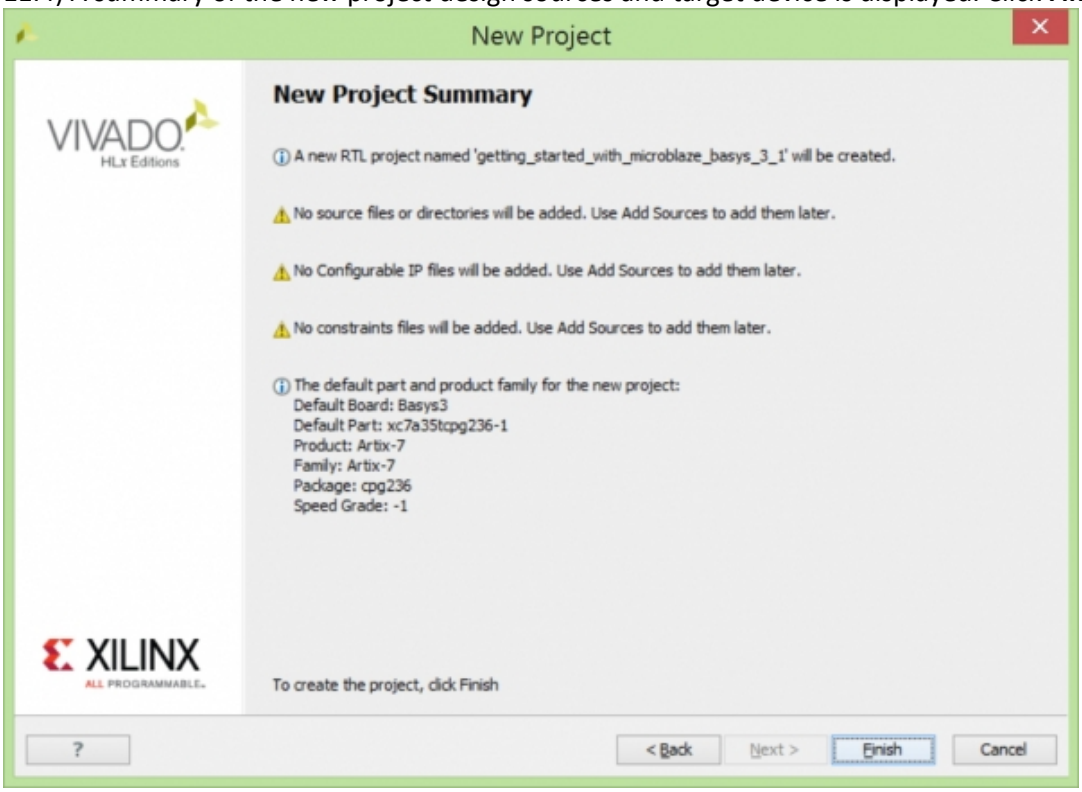
Refer back to steps 3.1 – 3.11 on how to open a new project!

When you first run Vivado this will be the main start window where you can create a new project or open a recent one.
11.1) Click on **Create New Project**. Choose the Project Name and Location such that there are **no blank spaces**. This is an important naming convention to follow for project names, file names and location paths. Underscore is a good substitute for empty spaces. It is good practice to have a dedicated folder for Vivado Projects, preferably with the smallest possible path length. Example: C:/Vivado_Projects. Name your Project and select the Project location and click **Next**.

11.2) Choose Project Type as **RTL Project**. Leave the **Do not specify sources** box unchecked and click **Next**.

11.3) If you have followed the Board Support File Wiki guide then click next until you can select **Boards**. From the filter options make required selections for Vendor, Display Name and Board Revision. **Basys 3** should be displayed in the selection list. A mismatch in selecting the correct board name will cause errors. Choose Basys 3 and click next.

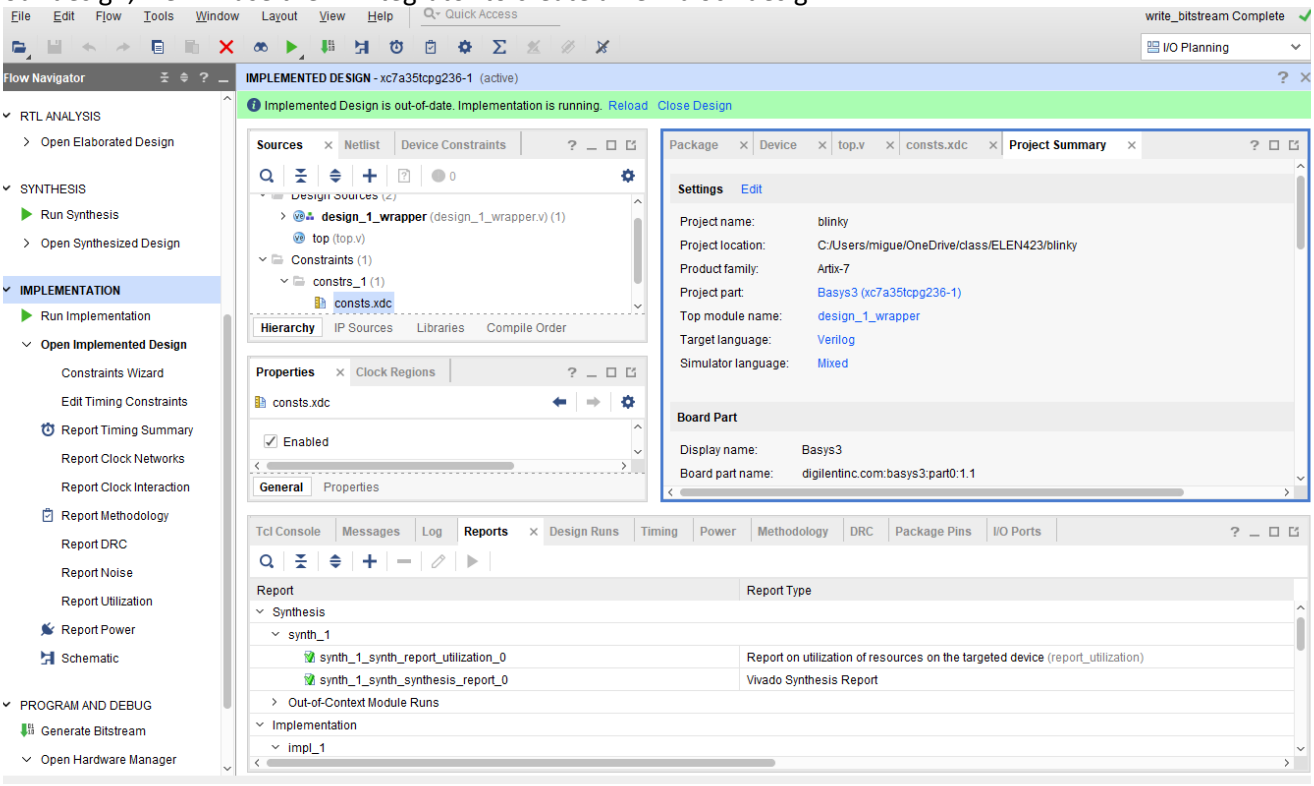
11.4) A summary of the new project design sources and target device is displayed. Click **Finish**.



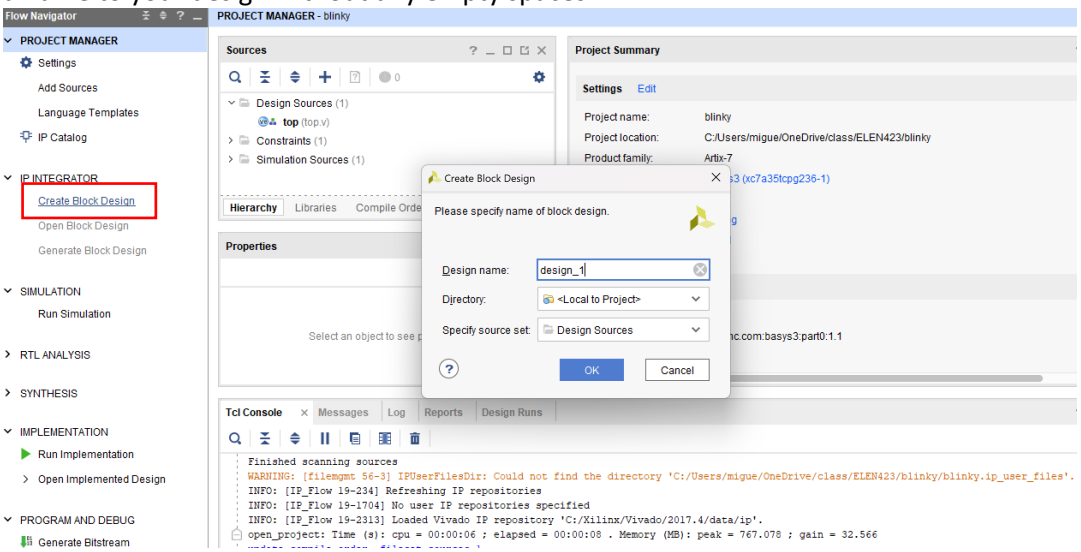
---

12. Creating New Block Design

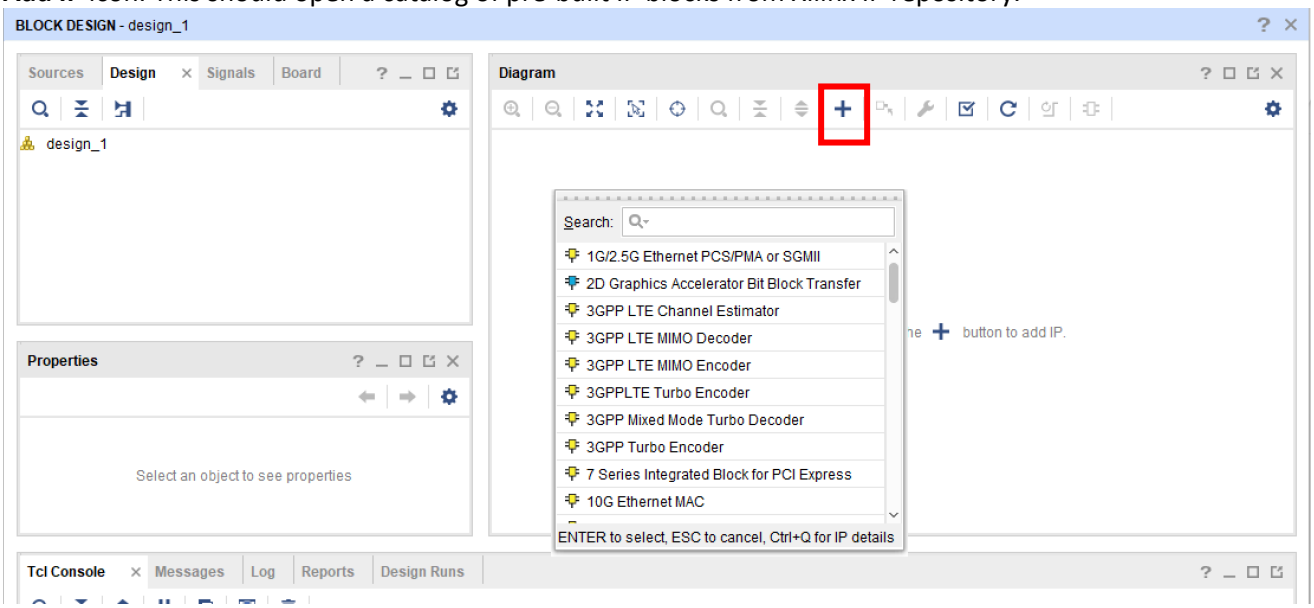
12.1) This is the main project window where you can create a IP based block design or add RTL based design sources. The flow navigator panel on the left provides multiple options on how to create a hardware design, perform simulation, run synthesis and implementation and generate a bit file. You can also program the board directly from Vivado with the generated bit file for an RTL project using the Hardware Manager. For our design, we will use the IP Integrator to create a new block design.



12.2) On the left you should see the Flow Navigator. Select **Create Block Design** under the IP Integrator. Give a name to your design without any empty spaces.

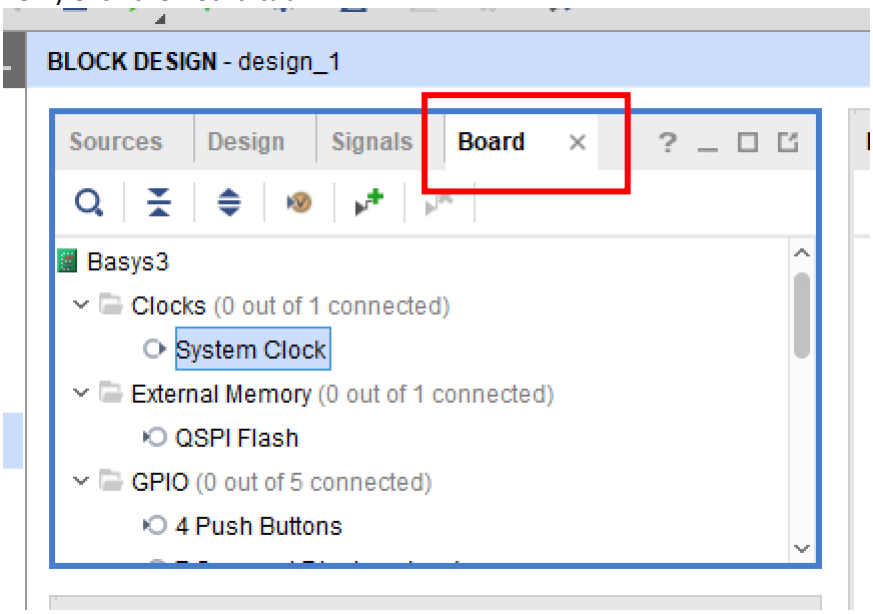


12.3) An empty design workspace is created where you can add IP blocks. Add an IP core by clicking on the **Add IP** icon. This should open a catalog of pre-built IP blocks from Xilinx IP repository.



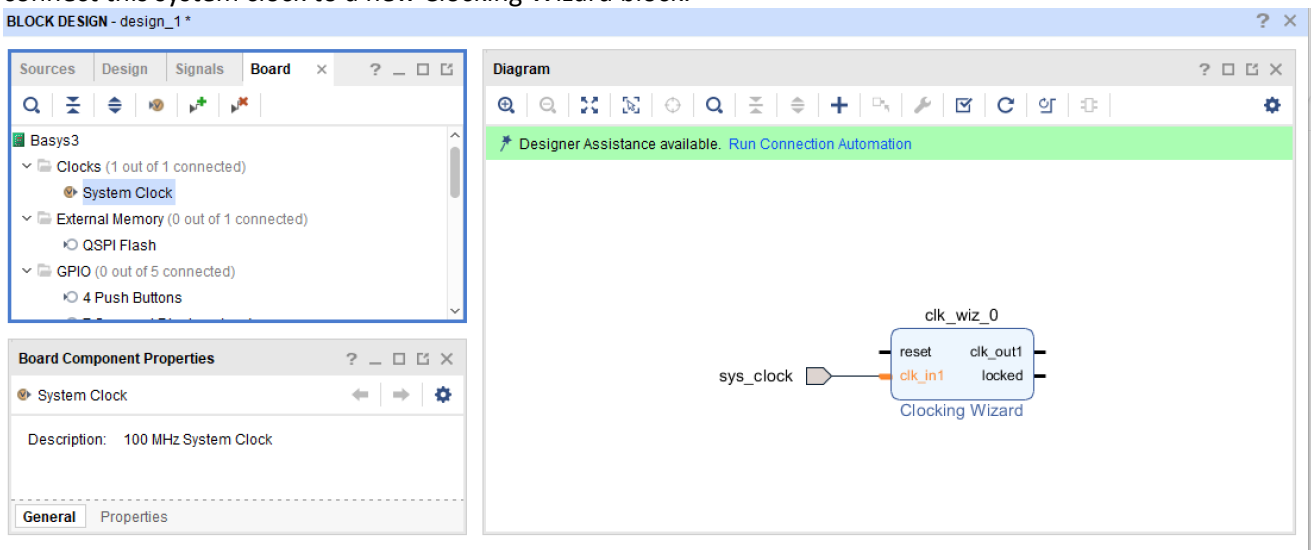
13. Adding the clock

13.1) Click the **Board** tab

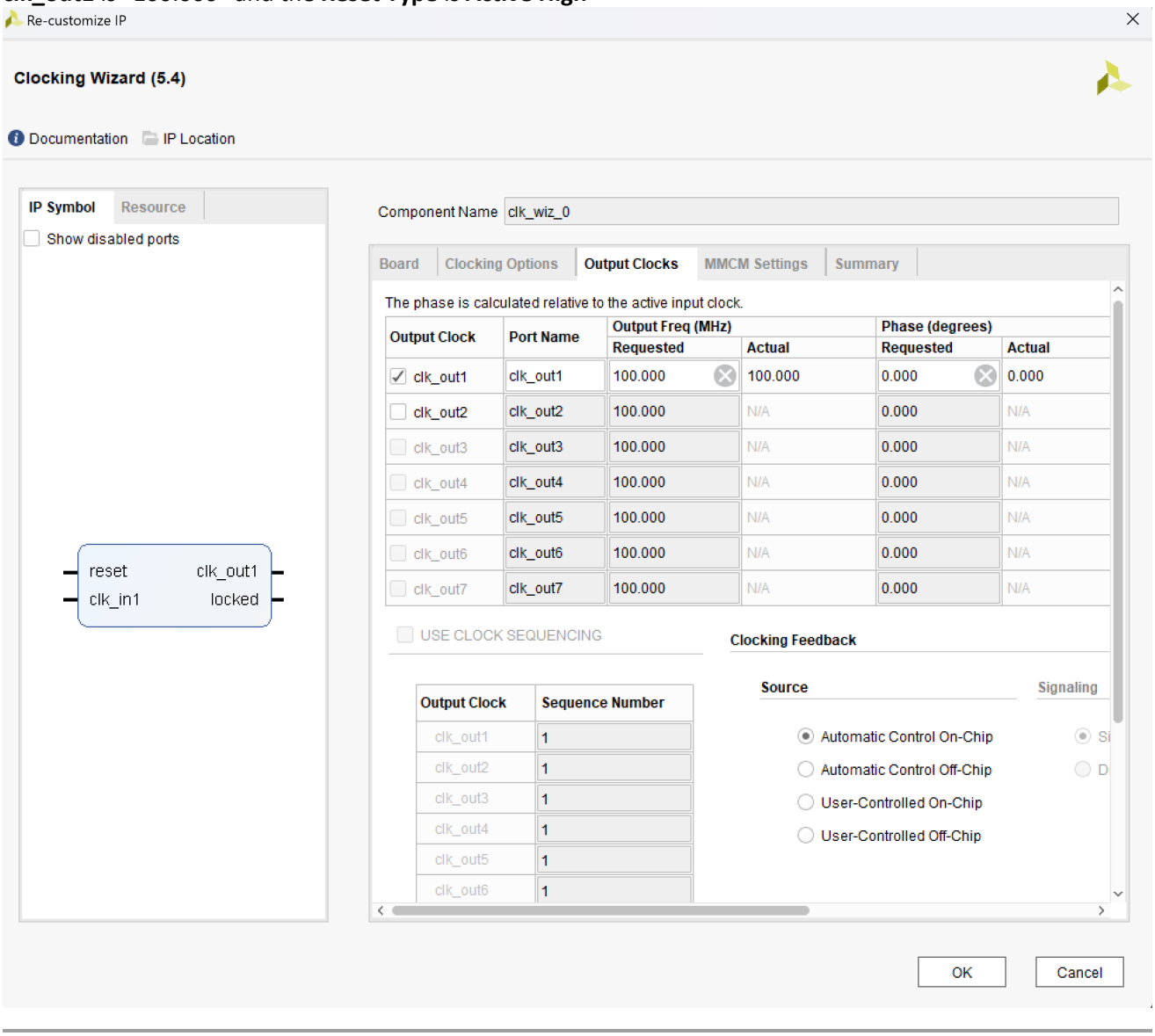


This list contains all of the components defined in the board file you installed before. These are already configured to work with several Vivado IPs.

13.2) Click and drag the **System Clock** component onto the empty block design. Vivado will automatically connect this system clock to a new Clocking Wizard block.

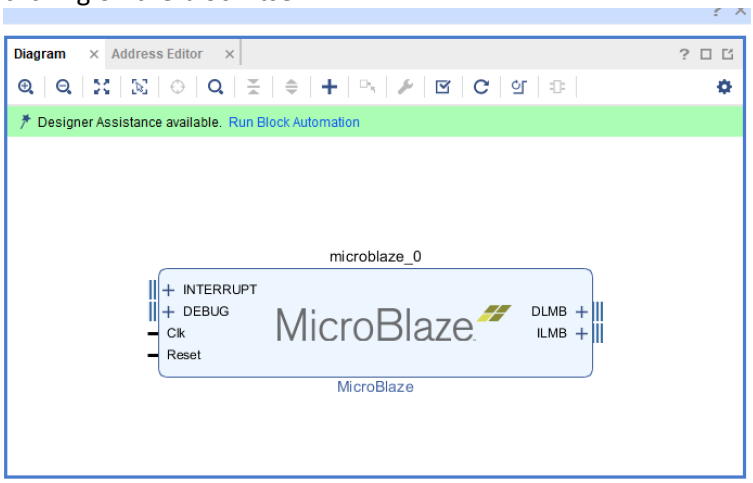


13.3) Double click the **Clocking Wizard** block to customize it. Click on the **Output Clocks** tab. Make sure **clk_out1** is “100.000” and the **Reset Type** is **Active High**

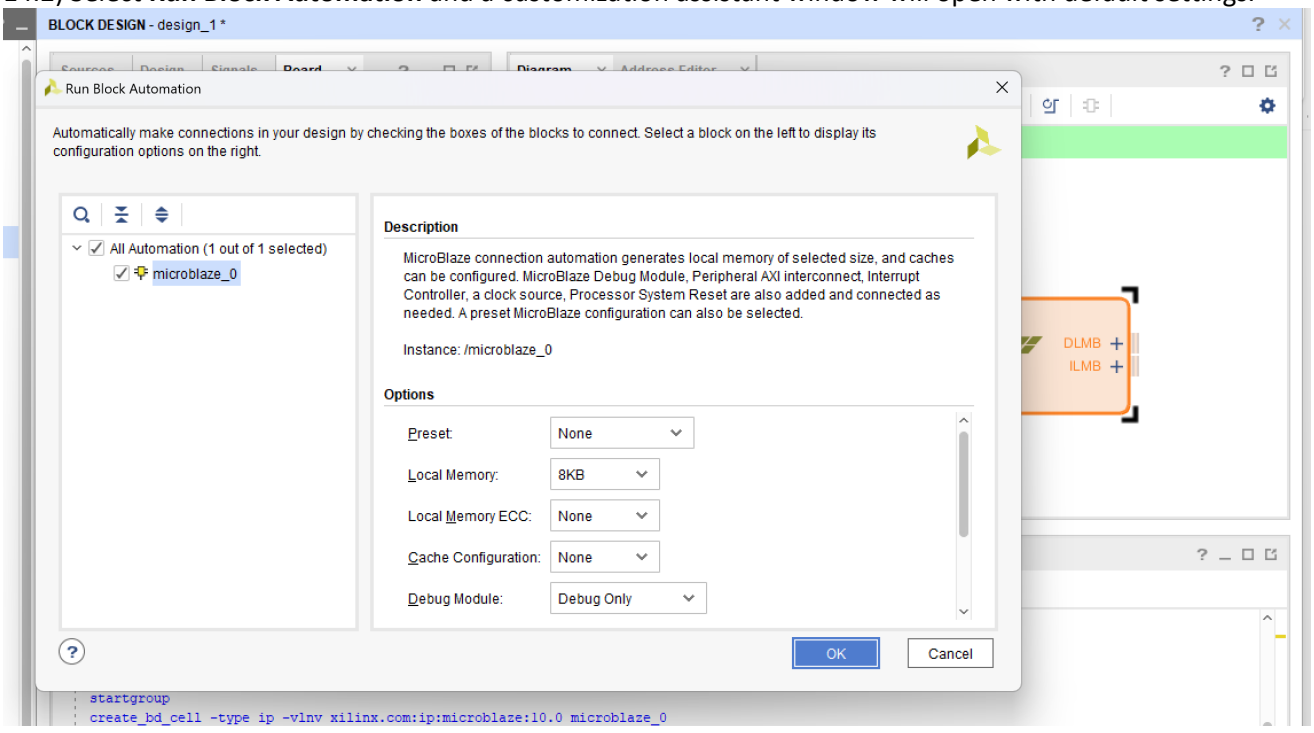


14. Adding Microblaze IP and Customization

14.1) Add an IP core by clicking on the **Add IP** icon. Search for "Microblaze" and double click on it to add the IP block to your design. This is the Xilinx Microblaze IP block. When a new IP block is added the user can customize the block properties by either clicking on the **Run Block Automation** message prompt or by double clicking on the block itself.

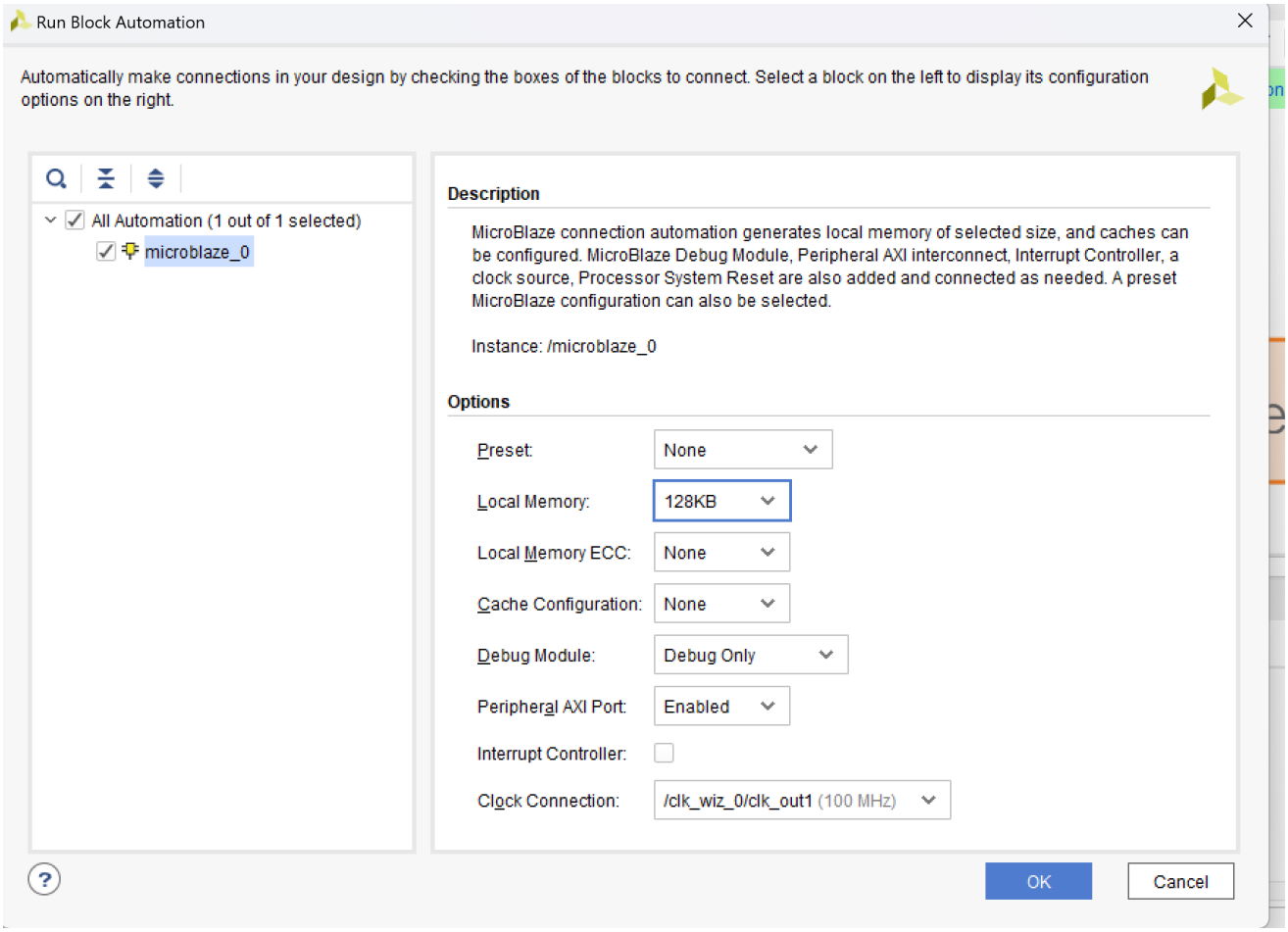


14.2) Select **Run Block Automation** and a customization assistant window will open with default settings.

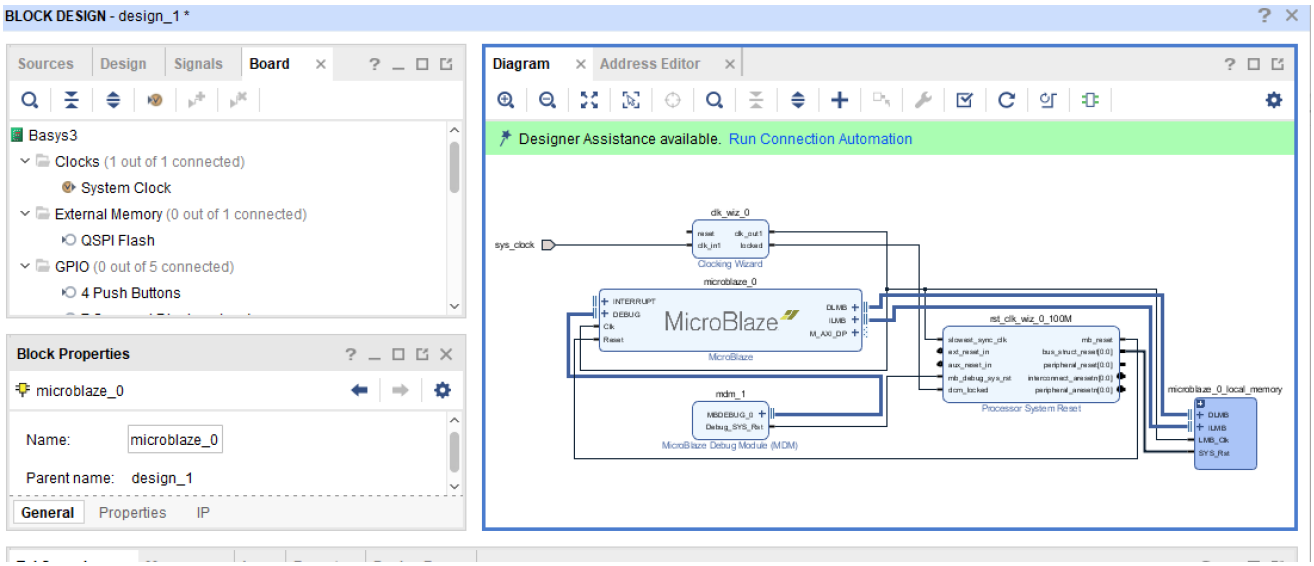


14.3) Change default settings in the block options as shown below and click **OK**. This will customize the block with our new user settings.

Local Memory: 128KB	**Local Memory ECC: None**	**Cache Configuration: None**
Debug Module: Debug only	**Peripheral AXI Port: Enabled**	**Interrupt Controller: unchecked**
Clock Connection: /clk_wiz0/clk_out1(100 MHZ)		



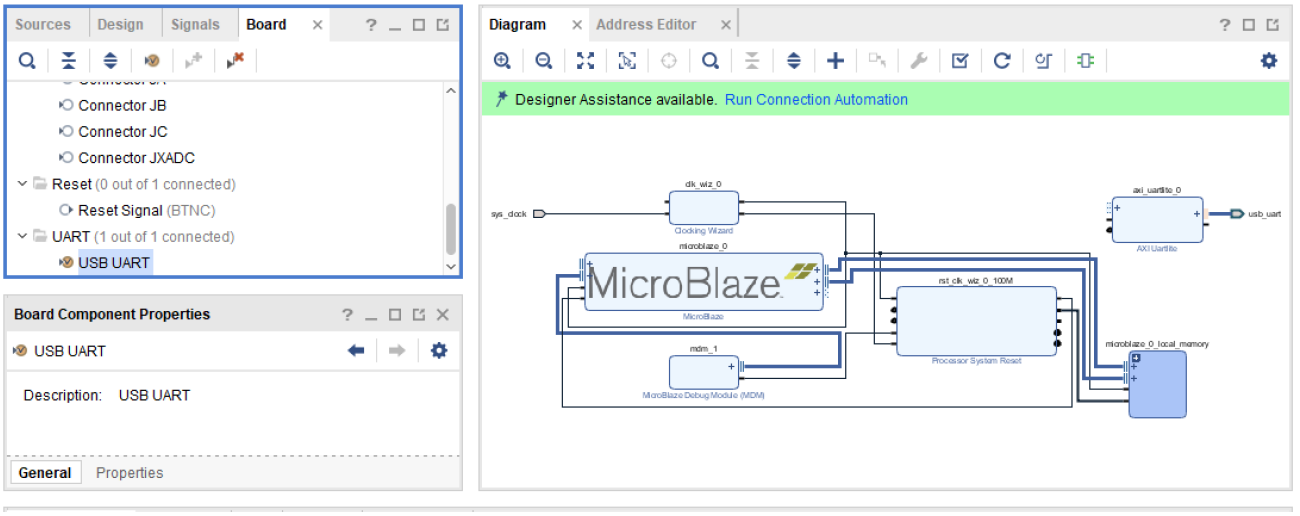
14.4) Running the block automation will auto-generate a set of additional IP blocks which will be added to our hardware design automatically based on the options selected in the previous step. **Do not click on Run Connection Automation yet.**



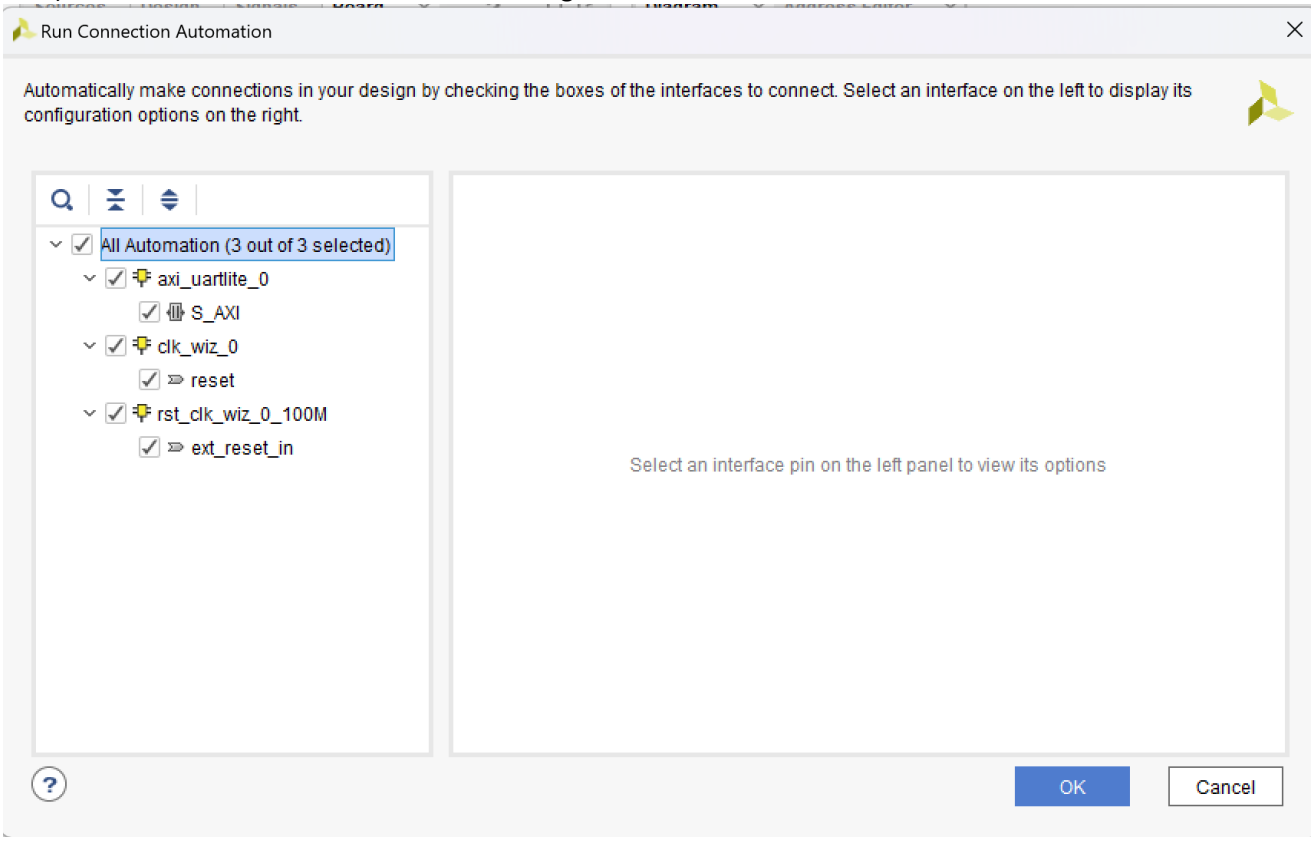
15. Adding Peripheral Components

15.1) Go into the **Boards** tab again and find the **USB UART** component. **Click and drag** this onto the block design to add the Uartlite block to your design.

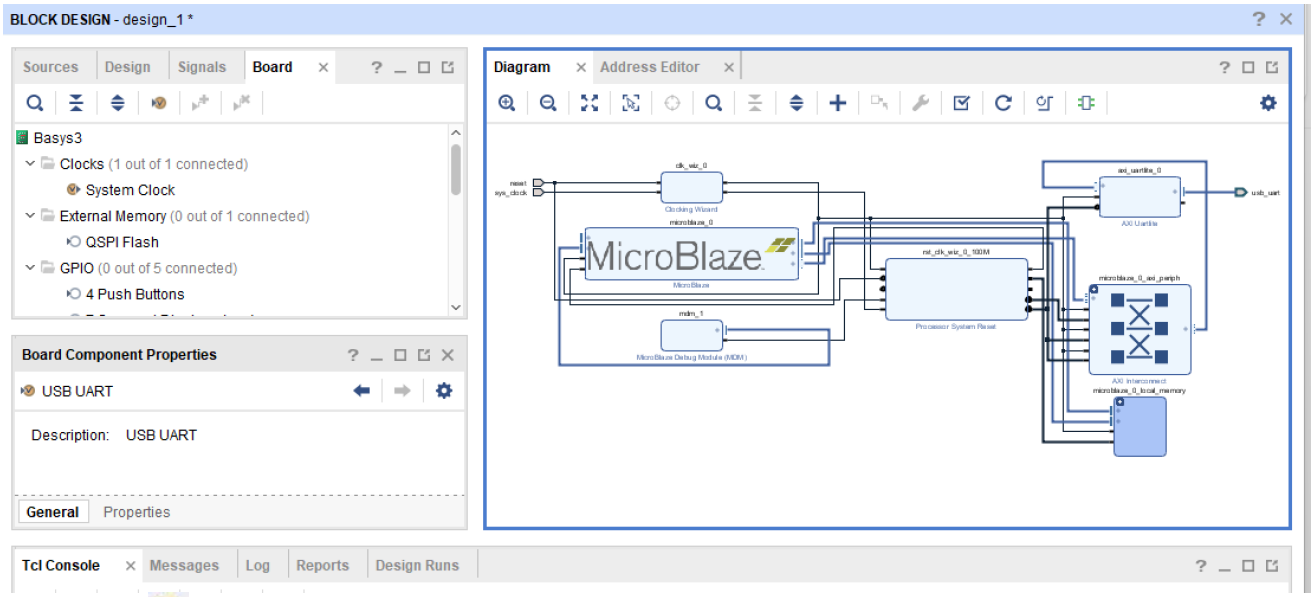
:



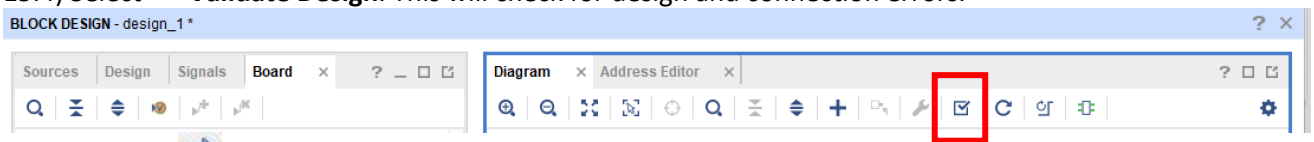
15.2) Click **Run Connection Automation** in the green banner. Check the **All Automation** box and click **OK**.



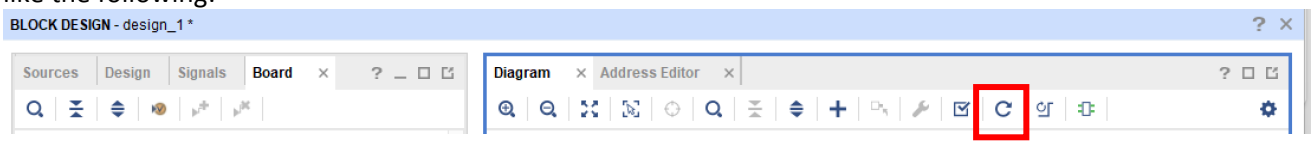
15.3) Click **Run Connection Automation** again in the green banner. Check the **All Automation** box and then click **OK**.

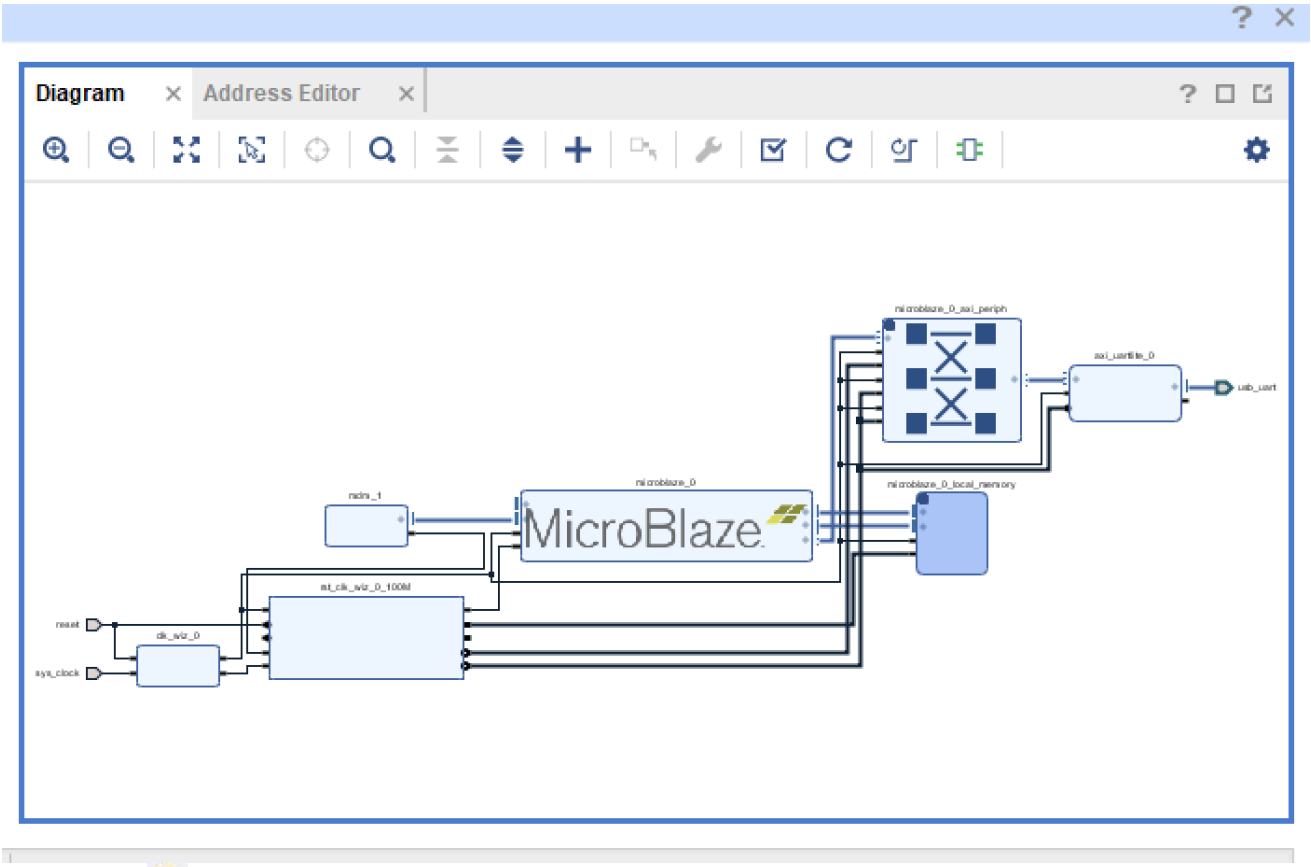


15.4) Select  **Validate Design**. This will check for design and connection errors.

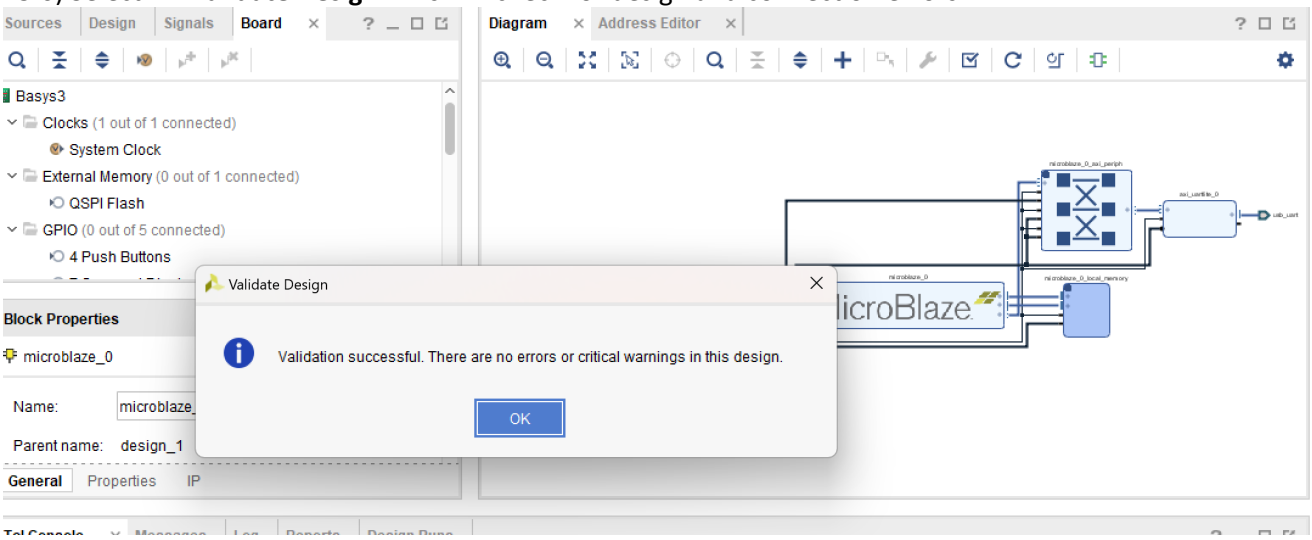


15.5) Click the  **Regenerate Layout** button to rearrange your block design. The block design should look like the following:



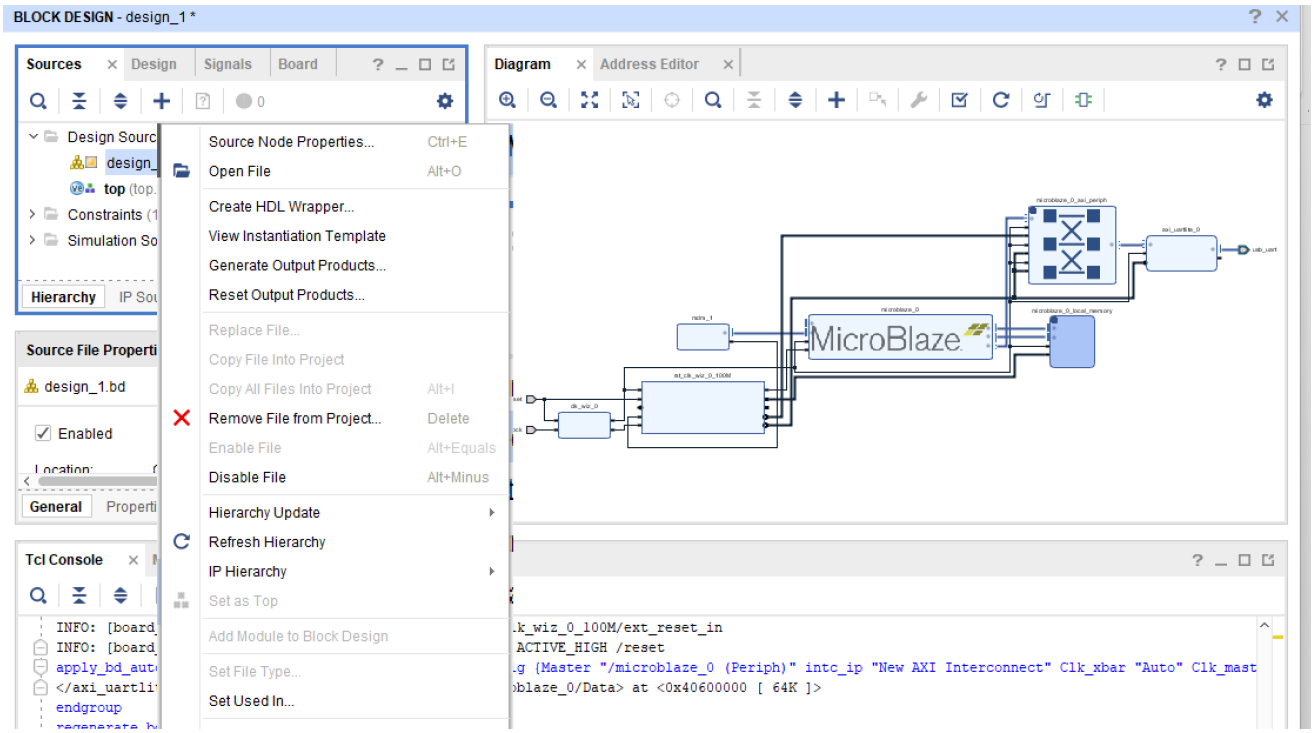


15.6) Select **Validate Design**. This will check for design and connection errors.



16. Making an HDL Wrapper

16.1) After the design validation step we will proceed with creating a HDL System Wrapper. Click on the **Sources** tab and find your block design.



16.2) Right click on your block design and click **Create HDL Wrapper**. Make sure **Let Vivado manage wrapper and auto-update** is selected and click **OK**.

This will create a top module in Verilog and will allow you to generate a bitstream.

---

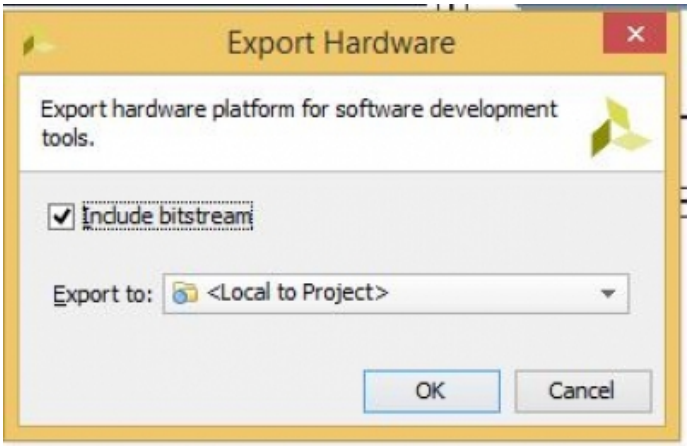
17. Generate a bitstream

17.1) Click Generate bitstream

---

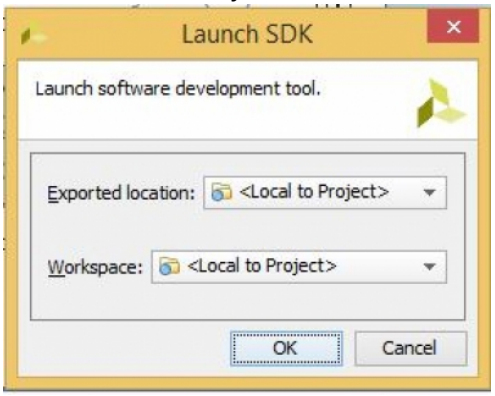
18. Exporting Hardware Design to SDK

18.1) On the main toolbar, click **File** and select **Export→Export Hardware**. Check the box to **Include Bitstream** and click **OK**. This will export the hardware design with system wrapper for the Software Development Tool - Vivado SDK.



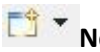
A new file directory will be created under **Hello_World.SDK** similar to the Vivado hardware design project name. Two other files, *.sysdef* and *.hdf* are also created. This step essentially creates a new SDK Workspace.

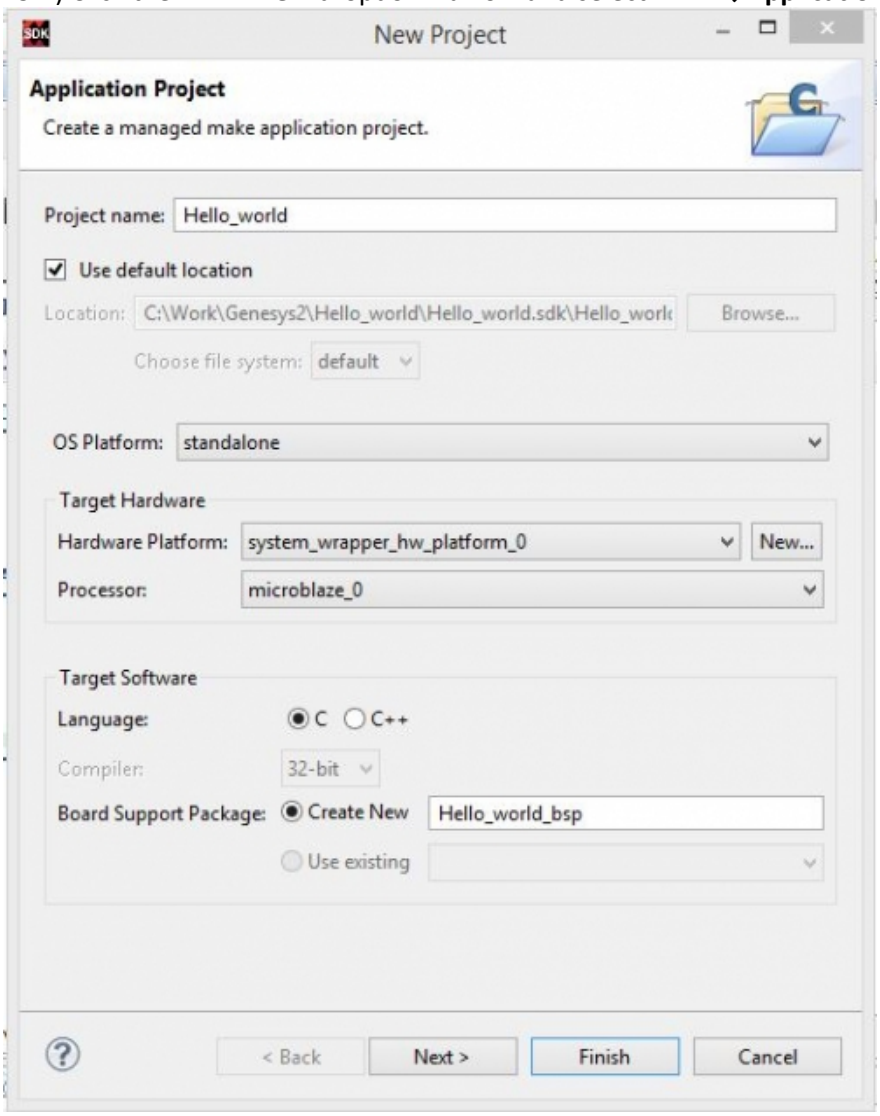
18.2) On the main toolbar, click **File** and then **Launch SDK**. Leave both of the dropdown menus as their default *Local to Project* and click **OK**. This will open Xilinx SDK and import your hardware.



---

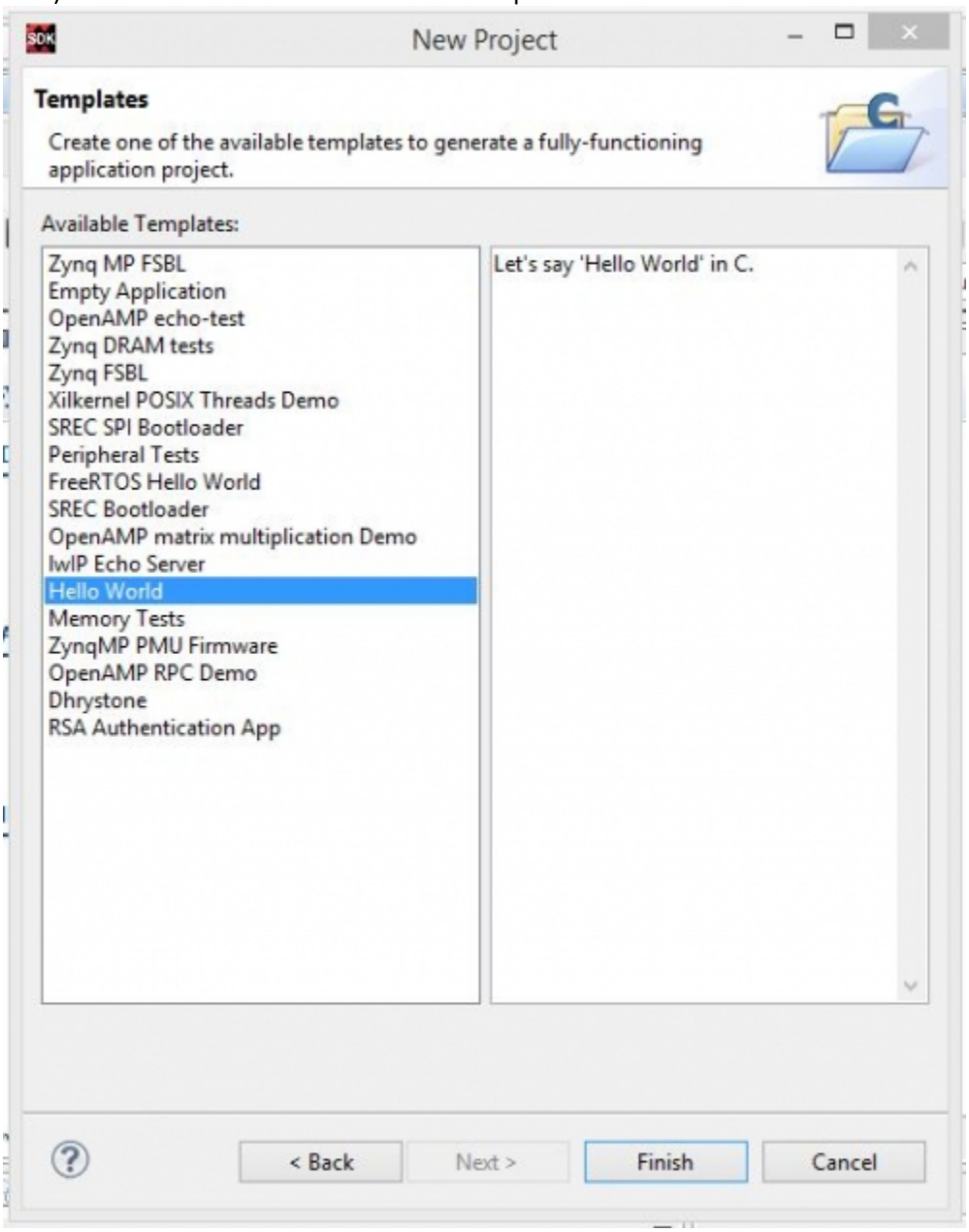
19. Creating New Application Project in SDK

19.1) Click the  **New** dropdown arrow and select **Xilinx→Application Project**.



Give your project a name that has no empty spaces and click **Next**.

19.2) Select **Hello World** from the list of templates and click **Finish**.



You will see two new folders in the **Project Explorer** panel.

- **Hello_world** which contains all the binaries, .C and .H (Header) files
- **Hello_world_bsp** which is the board support folder

Hello_world is our main working source folder. This also contains an important file shown here which is the "lscript.ld". This is a Xilinx auto generated linker script file. Double click on this file to open.

19.3) Back in the **Project Explorer**, double click and open **helloworld.c** under the **src** folder.
This is the main .C file which will print “Hello World” in the console when executed.

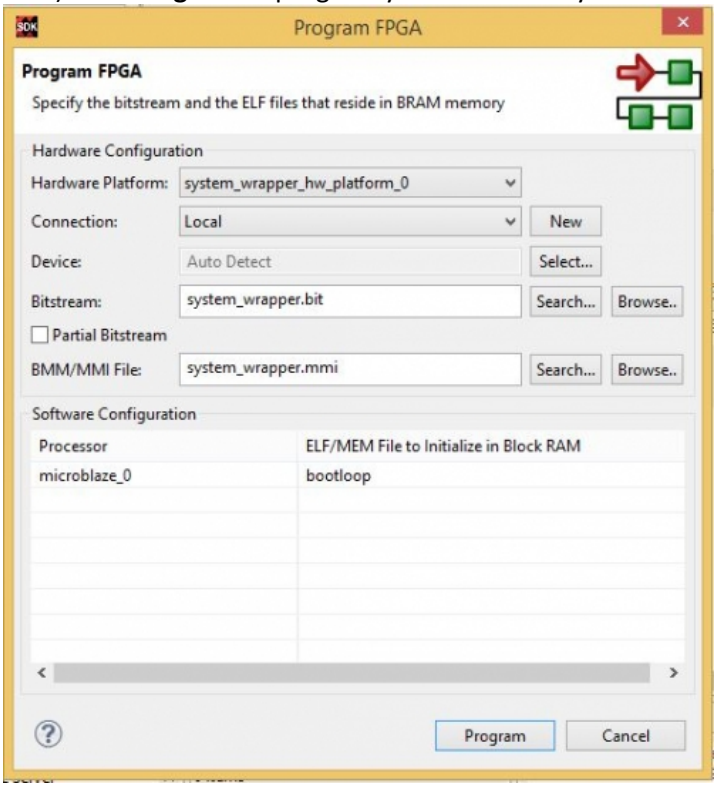
```
/* Copyright (C) 2009 - 2014 Xilinx, Inc.  All rights reserved.

/*
 * 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();

    xil_printf(" Hello World\n\r");

    cleanup_platform();
    return 0;
}
```

20. Programming FPGA with Bit File

20.1) Make sure that the Basys 3 is turned on and connected to the host PC via the USB-JTAG port - this port will serve dual purpose as the USB-UART connection to the Microblaze.

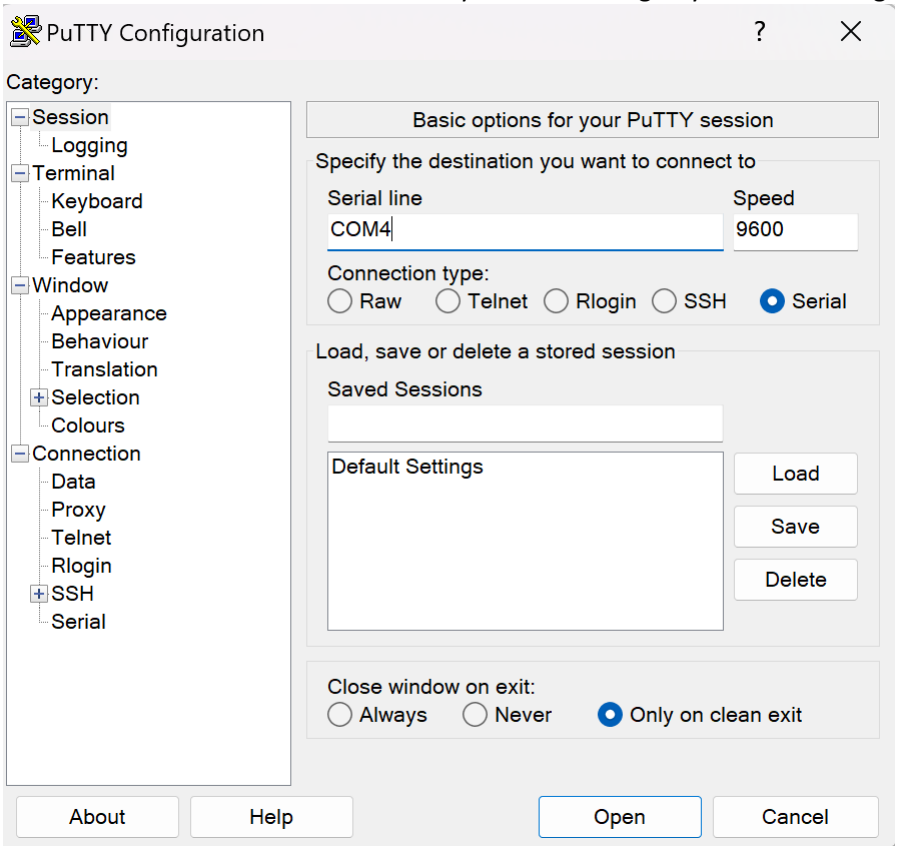
On the top toolbar, click the **Program FPGA** button.

20.2) Click **Program** to program your FPGA with your hardware design.

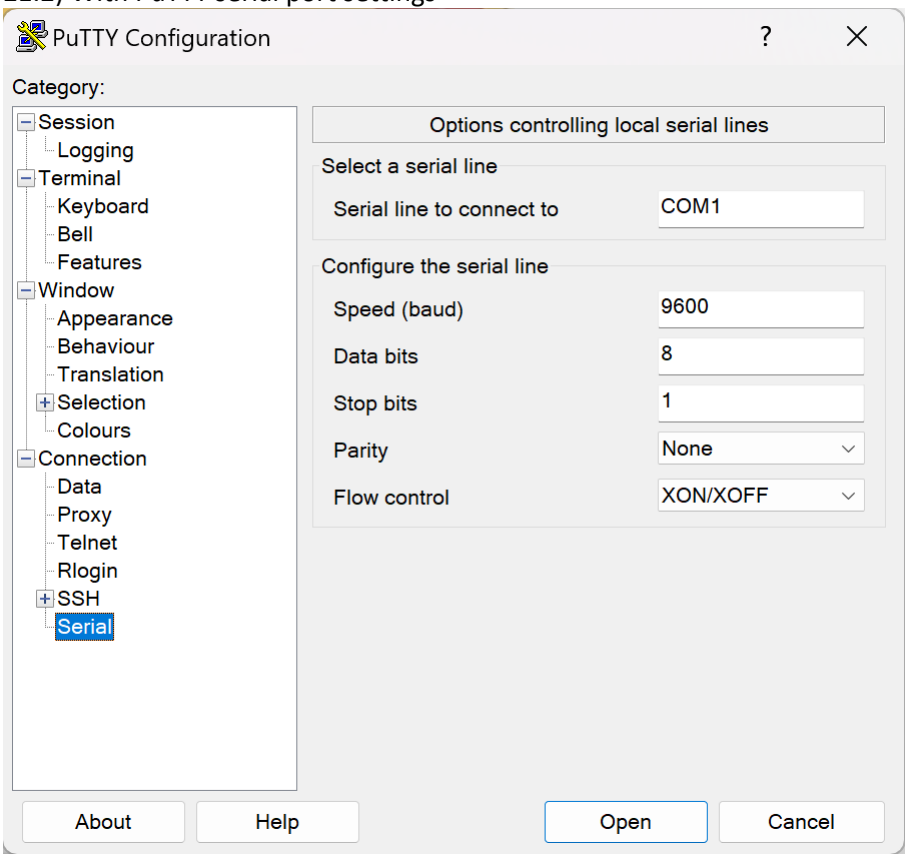


21. Setting up UART Terminal

21.1) Open up a Serial Terminal application like PuTTY. Connect to the Basys 3 UART port with a baud rate of 9600. This baud rate can be altered in your block design by double clicking the Uartlite block.

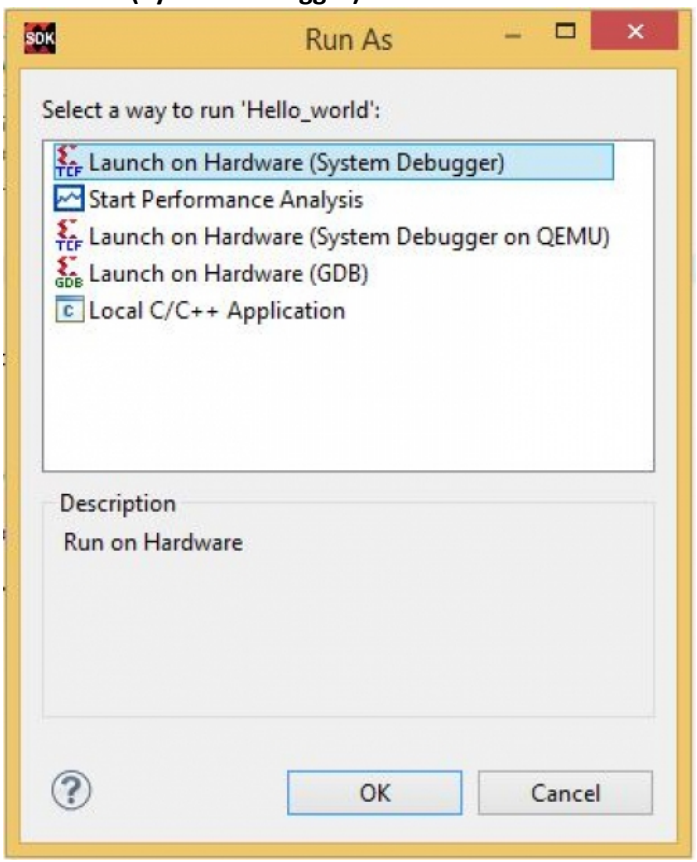


21.2) With PuTTY serial port settings

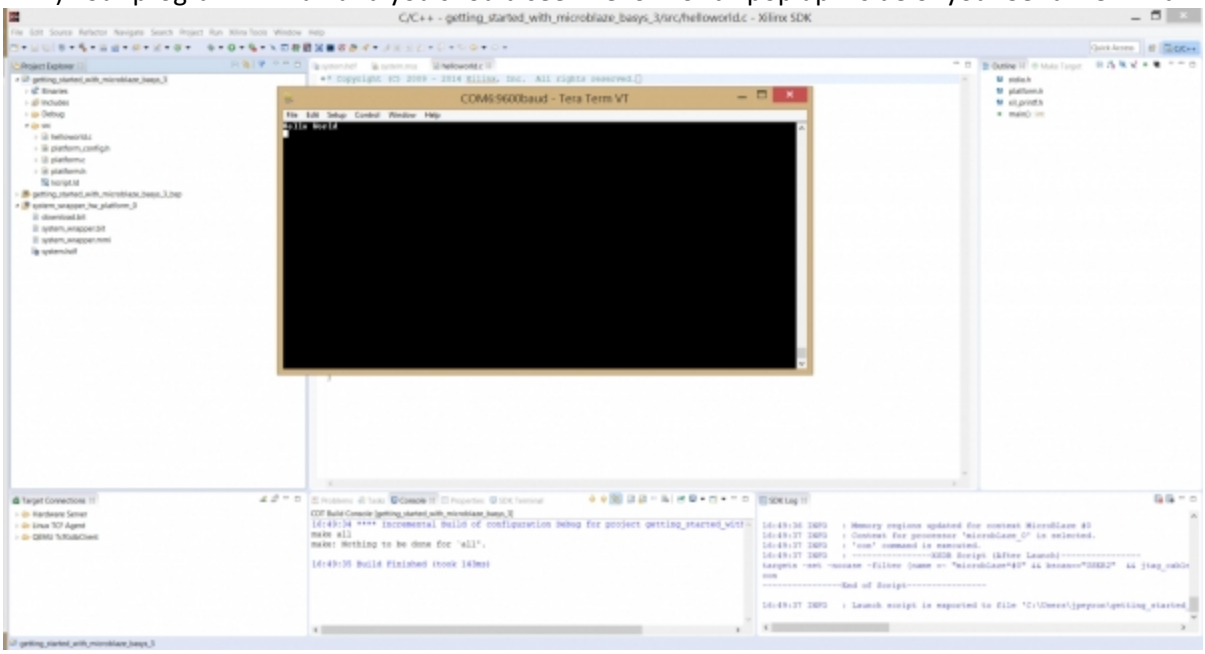


22. Program the Microblaze Processor

22.1) Back in SDK, select your **Hello_world** project and click the  **Run As…** button. Select **Launch on Hardware (System Debugger)** and click **OK**.



22.2) Your program will run and you should see “Hello World” pop up inside of your Serial Terminal. Hooray!



23) Take a snapshot of the terminal for your records. Call over the instructor to get checked off.


COM7 - PuTTY
Hello World
Hello Mr Foot