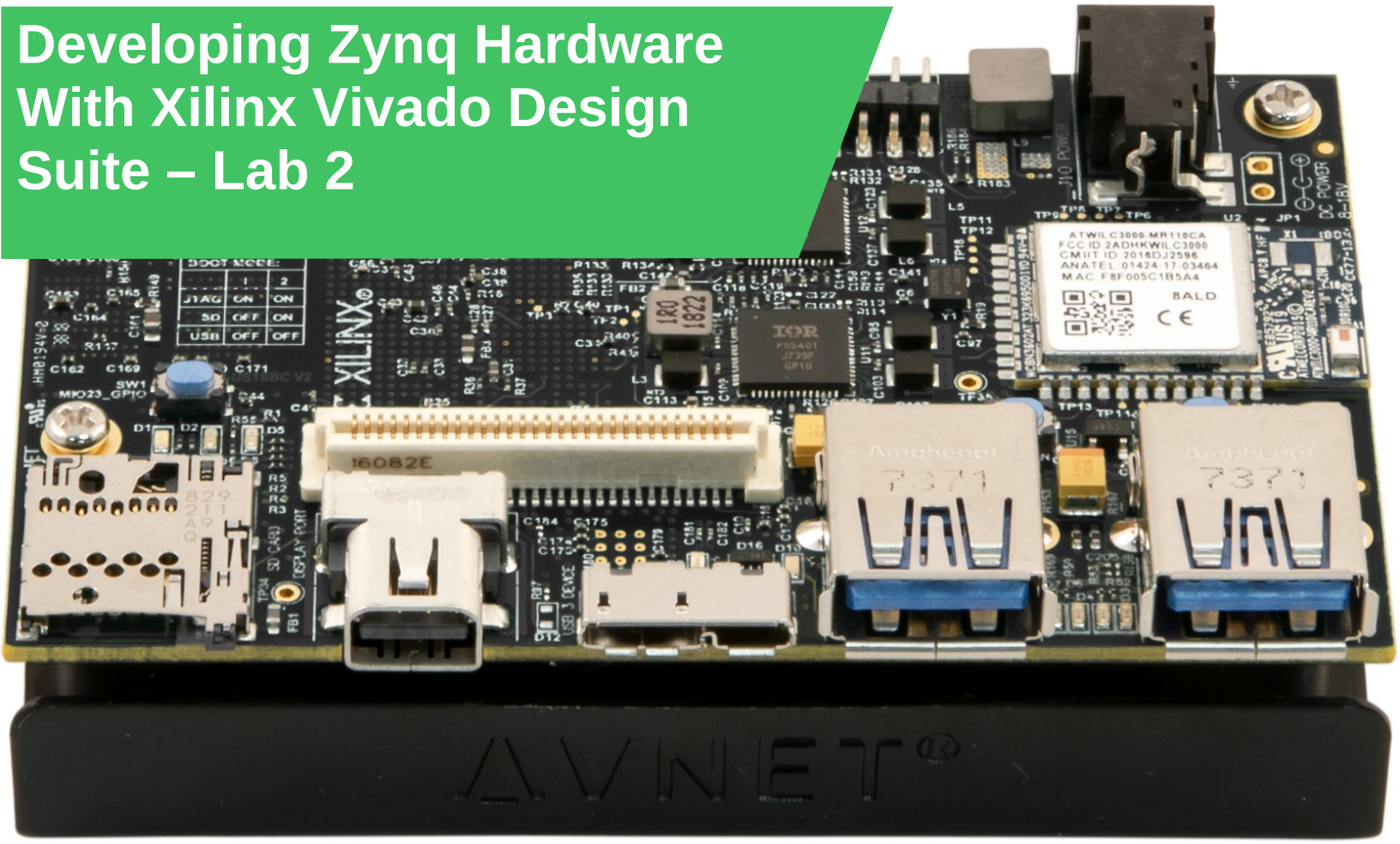
Avnet Technical Training Course

Developing Zynq Hardware With Xilinx Vivado Design Suite – Lab 2



Tools:	2019.1
Training Version:	v13
Date:	July 2019

Lab 2 Overview

At the conclusion of Lab 1, an ARM Processing System was added to the IP Integrator block design. The Zynq Re-customize IP tool appeared when we double-clicked the IP in the block design. In this lab, we will configure our processing subsystem by adding a UART peripheral, configuring internal clocking resources, and setting up our DDR memory controller. When done, we'll export our design to the Software Development Kit (SDK) and create a simple Hello World application.

Lab 2 Objectives

When you have completed Lab 2, you will know how to do the following:

- Enable and map a Zynq PS UART peripheral
- Configure Memory and Clocks for the Zynq PS
- Build the hardware platform
- Export a design to SDK
- Create and run a Hello World application

Experiment 1: Enable and Map a Zynq PS UART peripheral

To start, we'll do something very simple by enabling a single UART peripheral in the design and map it to the Multiplexed I/O (MIO). MIO refers to the pins that the processor can route its peripherals too.

Experiment 1 General Instruction:

For MiniZed

Set the Bank voltages to LVCMOS 3.3V for Bank 0 and LVCMOS 3.3V for Bank 1. Enable the UART1 peripheral and map it to MIO[48:49].

Experiment 1 Step-by-Step Instructions:

1. If not already open, open the ZynqDesign project and open the Block Design, **Z_system.bd**. Double-click the Zynq Processing System to customize the IP.

The Zynq Block Design window shows that no ARM peripherals or Flash Memory interface are currently enabled. You can determine this since all I/O Peripheral and Flash Memory boxes are unchecked, as shown in the figure below. This also means that no MIOs are connected.

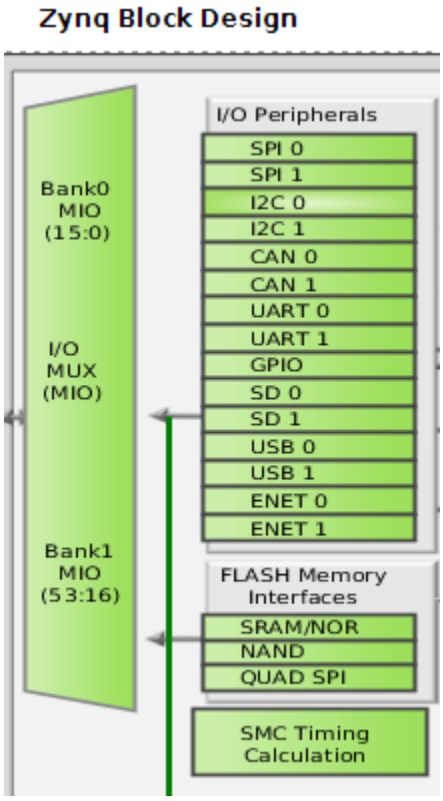


Figure 1 - Zynq I/O Peripherals

2. Click anywhere in the I/O Peripherals Box or select **MIO Configuration** from the Page Navigator. This will open the MIO Configuration page.

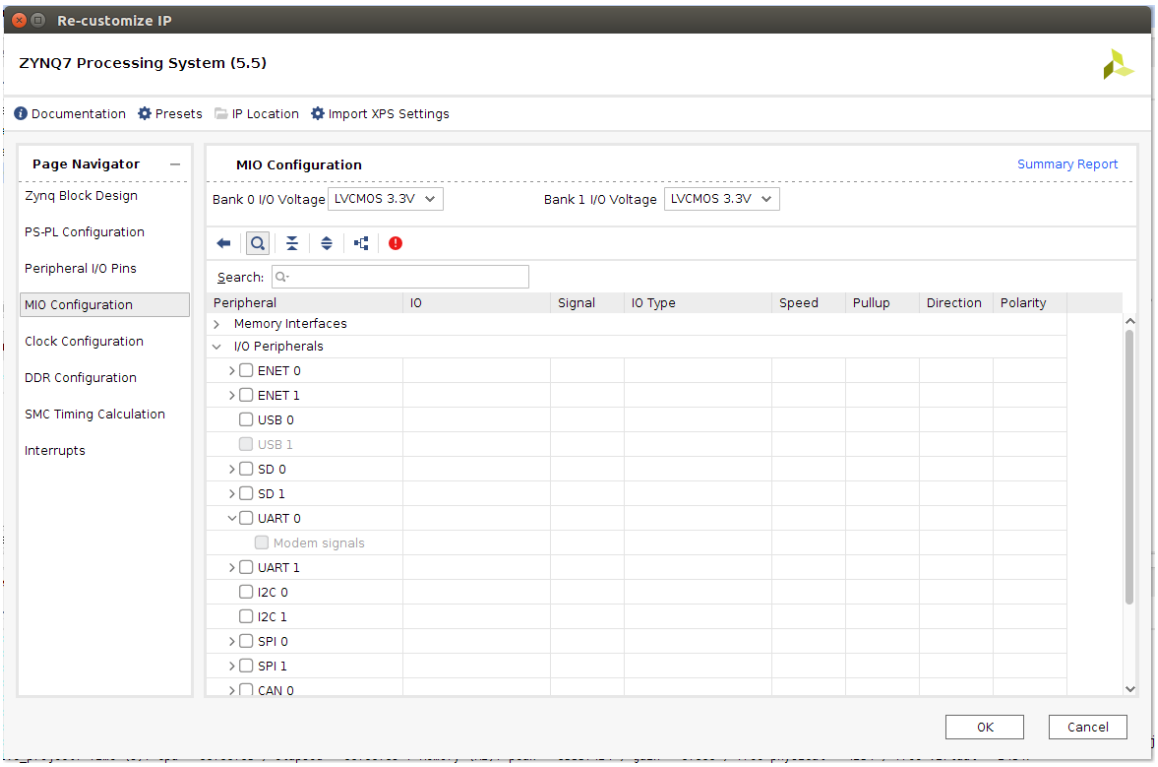


Figure 2 - MIO Configuration

3. At the top of the MIO Configuration window, the Bank Voltage settings are listed. For MiniZed, set Bank 0 I/O Voltage to **LVCMOS 3.3V** and Bank 1 to **LVCMOS 3.3V**. Note: If targeting an Avnet Zynq board other than MiniZed, it is likely required that Bank 1 be set to LVCMOS 1.8V

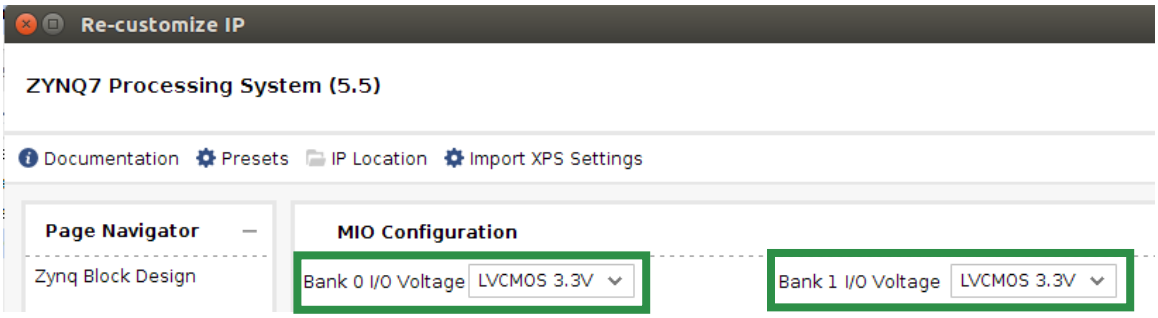


Figure 3 - MIO Bank Voltages

At this time no peripherals are selected. Note, the peripherals are not listed in alphabetical order. The peripherals are listed from top to bottom in order of priority based on their importance in the system (like the Flash) or how limited they are in the possible MIO

mappings. The peripherals are ordered with the least flexible at the top and most flexible at the bottom, with the exception of the USB, as it can only be connected in one location.

When mapping out a board, a designer should start at the top of the list and work their way down. A developer has to carefully balance which peripherals will map to the MIO and which ones are mapped to EMIO (EMIO refers to mapping the processors Peripheral through the PL). However, to simplify this experiment, we will focus just on a single peripheral to show how the process works.

4. There are two UARTs available. Check the box for **UART1**. Click the pull-down for the I/O. Notice that UART1 can be placed on several MIO locations as well as extended MIO (EMIO). Set the I/O to **MIO48 .. 49**, which happens to be the default.

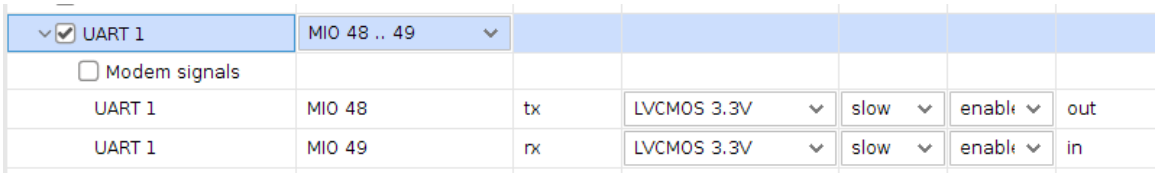


Figure 4 - UART1 Connection

5. Select the **Zynq Block Design** link in the Page Navigator. Notice in the I/O Peripherals that UART1 is checked to indicate it is connected.

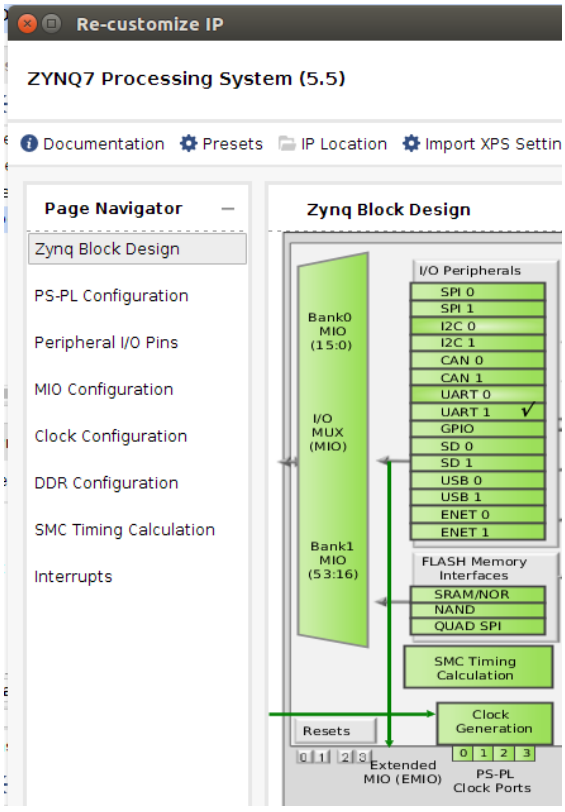


Figure 5 - UART1 MIO Connected

6. In the Zynq Block Design window, click the **General Settings** box or from the Page Navigator window select **PS-PL Configuration**.

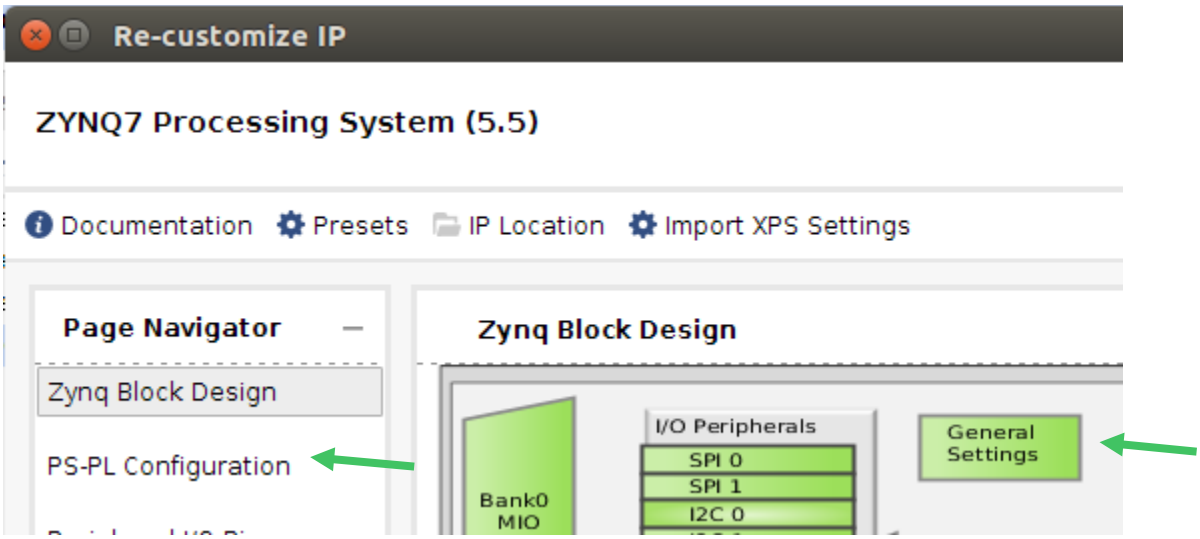


Figure 6 - General Zynq Settings

7. Expand the *General* section, verify the UART1 Baud Rate is set to **115200**.

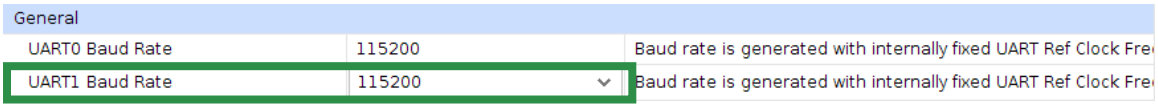


Figure 7 - UART1 Baud Rate Settings

Questions:

Answer the following questions:

- *What do you think is the purpose of EMIO?*

_____

- *Why are the Peripherals not listed alphabetically in the I/O Peripherals Configuration tool?*

_____

- *Extra Credit: If the Modem Signals are used with one of the UART peripherals, where must they be mapped?*

_____

Experiment 2: Configure Memory and Clocks for the Zynq PS

A few critical Zynq PS elements must be configured before even a simple Hello World can be run. This includes the DDR3L memory, as it is the RAM that will execute the Zynq PS applications. Also, the system clocks must be configured correctly.

Experiment 2 General Instruction:

Configure the Memory GUI for a 16-bit interface using Micron DDR3 memory components. Configure the clocks to operate the CPU at 667 MHz and the memory at 533 MHz.

Experiment 2 Step-by-Step Instructions:

1. Click on the box for **Clock Generation** or select **Clock Configuration** from the Page Navigator.

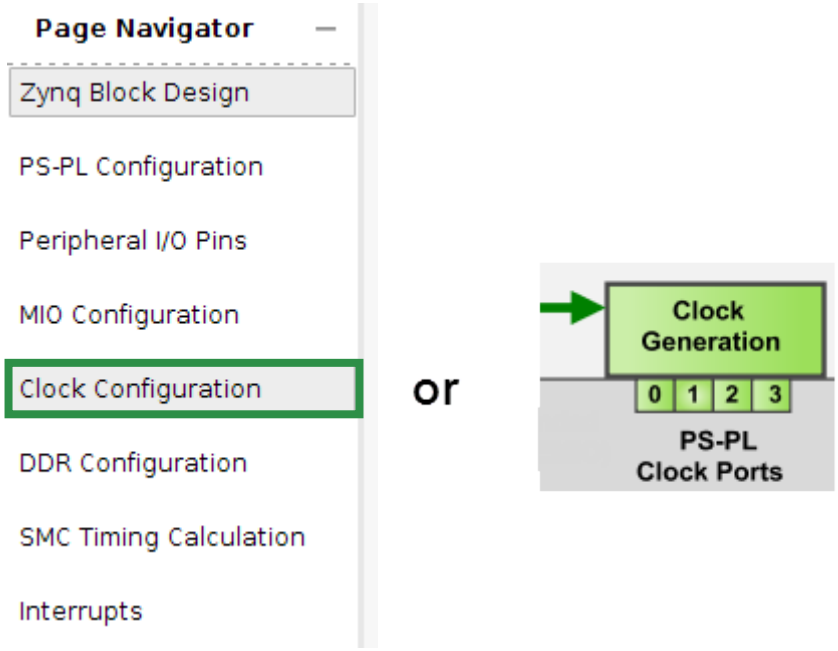


Figure 8 - Clock Configuration

2. Select the **Expand All** button to view all the clocks.

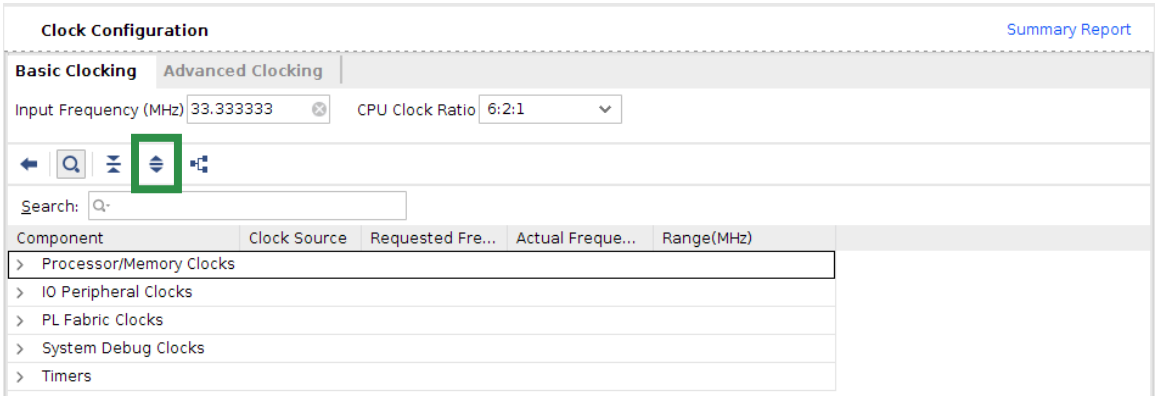


Figure 9 - Expand to see all clocks

3. For the most part, the default clock settings match MiniZed. **Verify** the following:
   - Input frequency is 33.333333 MHz
   - CPU frequency is 666.666666 MHz
   - DDR frequency is 533.333333 MHz

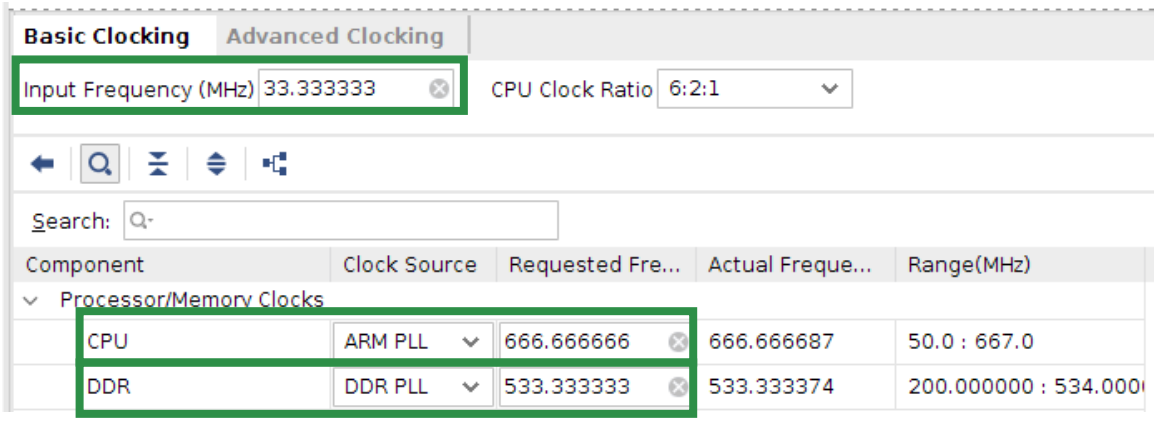


Figure 10 - Clock Settings

4. For now, we will change one of the default settings. One of the PL fabric clocks is enabled but we are not using the PL yet. **Disable** this by unchecking the box for this item.

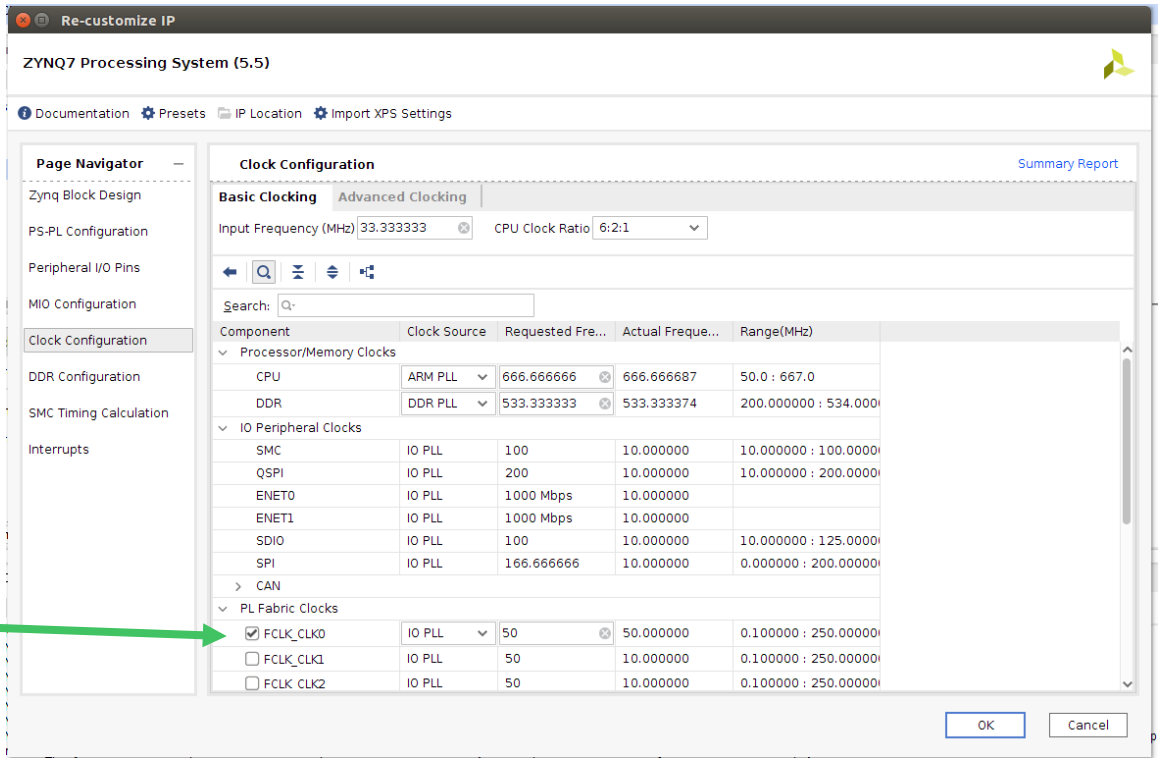


Figure 11 - Disable FCLK_CLK0

5. In addition to disabling the default setting for the PL Fabric Clock, we must also disable the default AXI connection to the PL. This is done in *PS-PL Configuration*. Select *PS-PL Configuration and* expand **AXI Non Secure Enablement → *GP Master AXI* Interface** section, **disable** the ***M AXI GP0 Interface*** by unchecking the box.

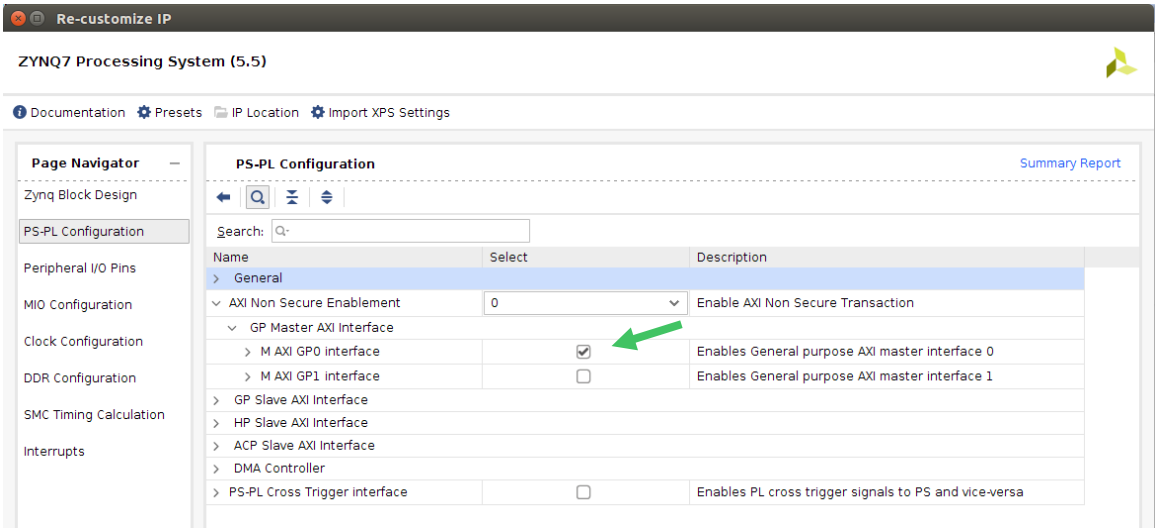


Figure 12 - Disable M AXI GP0 Interface

Next we will configure the memory controller.

6. Select **DDR2/3, LPDDR2 Controller** from the Zynq Block Design or **DDR Configuration** in the Page Navigator.

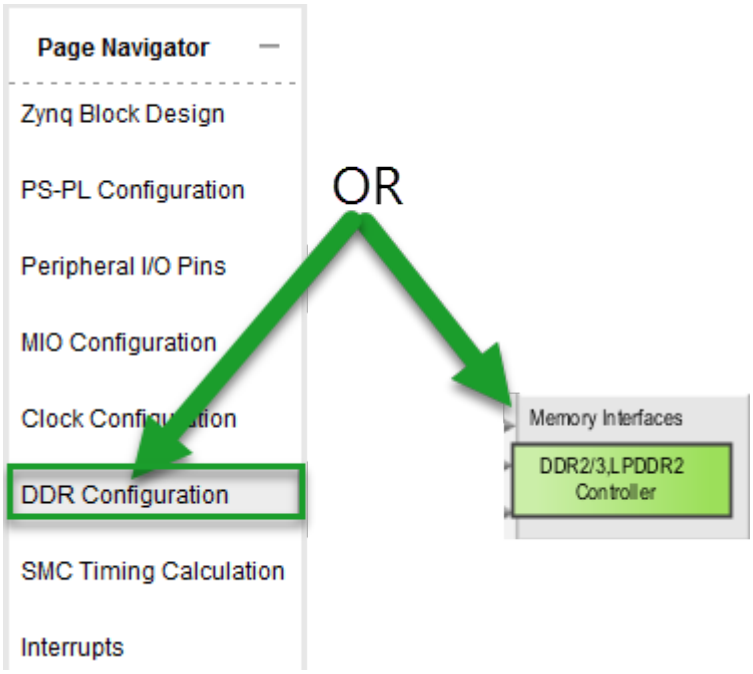


Figure 13 - DDR Memory Configuration

7. Once again, use the **Expand All** button to view all memory parameters. Verify the DDR is **enabled**.

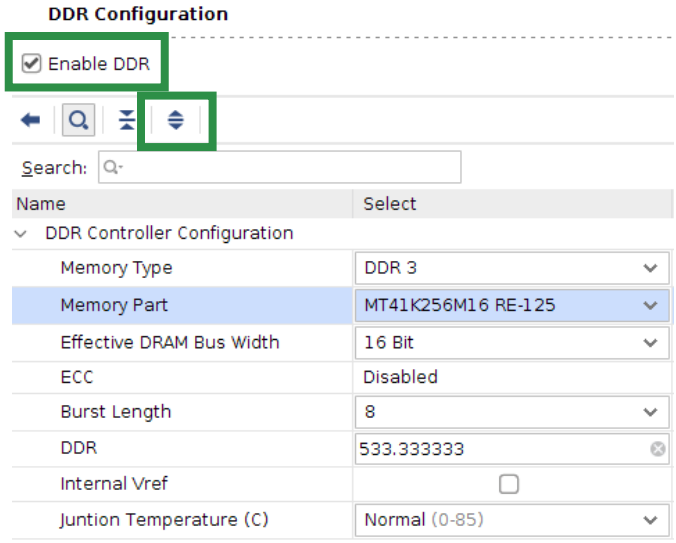


Figure 14 - View Memory Parameters and Enable DDR

8. We must match the DDR settings to the board we are using. Select the following DDR3 memory parameters for the specific board you are targeting:

    - *Memory Type*
        - o MiniZed = **DDR3 (Low Voltage)**
    - *Memory Part*:
        - o MiniZed = **MT41K256M16RE-125**

Notice how the *Memory Part Configuration* section automatically updates when the *Memory Part* is selected. If your memory device of choice was not in the default catalog, you could

select Custom. Then these boxes would be available for manually entering timing parameters for your selected device.

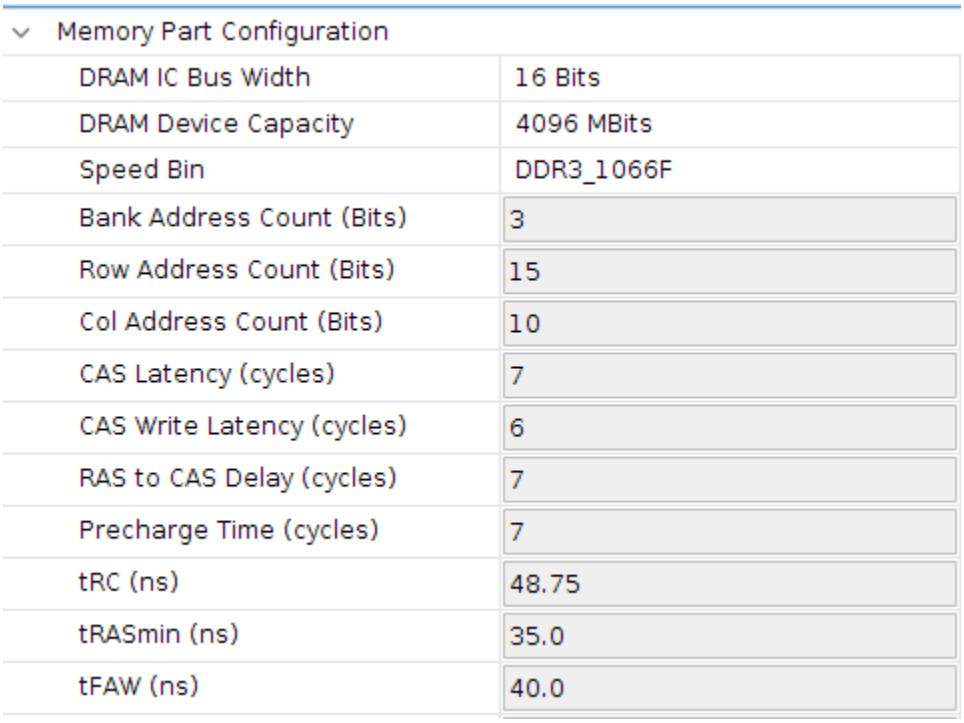


Figure 15 – MiniZed Memory Part Configuration

9. For MiniZed make sure the DRAM Bus width is **16-bits**. Since we are using DDR3 on a 7Z007S device, the ECC and Burst Length settings are predetermined. Notice also that the operating Frequency has automatically been inherited from the Clock Configuration screen to be 533 MHz.

   For MiniZed:
   Make sure the box for Internal Vref is unchecked. The operating temperature can remain at the Normal (0-85) range.

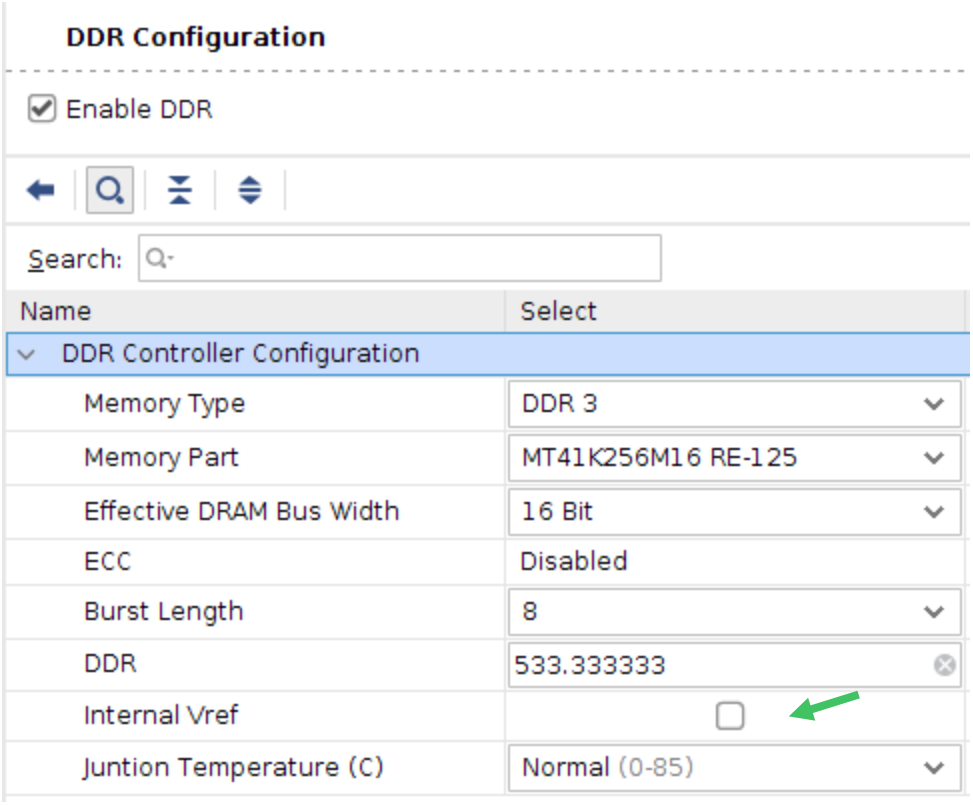


Figure 16 - DDR Controller Configuration (MiniZed)

Setting the *Training/Board Detail* parameters is next.

10. DRAM Training must be enabled for *Write leveling*, *Read gate*, and *Read data* eye options. **Check those 3 boxes** now if not already checked. An explanation of what these are is in the Zynq TRM, Section 10.6.8 UG585 (v1.11) September 27, 2016.

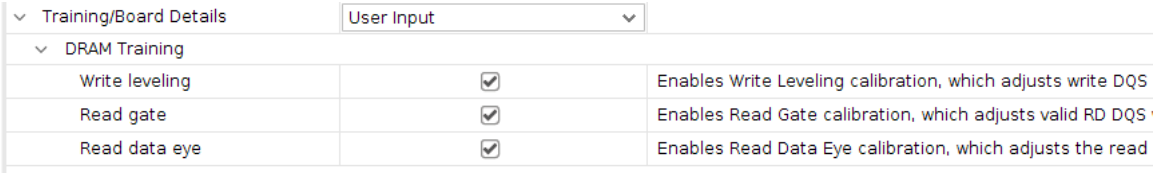


Figure 17 - Training/Board Details

Notice there are four entries to allow for *DQS to Clock Delay (ns)* and *Board Delay (ns)* information to be specified for each of the four byte lanes. These numbers assist the training algorithm with a starting point inside the DDR3 data valid window. All delays by default start at 0.0. Keep in mind the parameters for these fields are specific to each individual PCB design and Zynq package.

The values are based on the PCB trace lengths and the specific Zynq package chosen. Vivado already understands which package has been chosen, but the user must enter several values for the PCB trace lengths.

The procedure to calculate these lengths is included in the attached Appendix. If you have enough time after you finish the lab, complete the exercises there. However, in the interest of time, the Delay numbers will be given to you now.

11. Edit the ***DQS to Clock Delay*** and ***Board Delay*** settings as shown here.

    Note: When entering these values, enter them exactly as shown below. For negative numbers to be successfully passed through entry validation, it is required that the leading **0** placeholder be entered immediately following the minus sign (**-**) and before entering the decimal point (**.**) symbols.

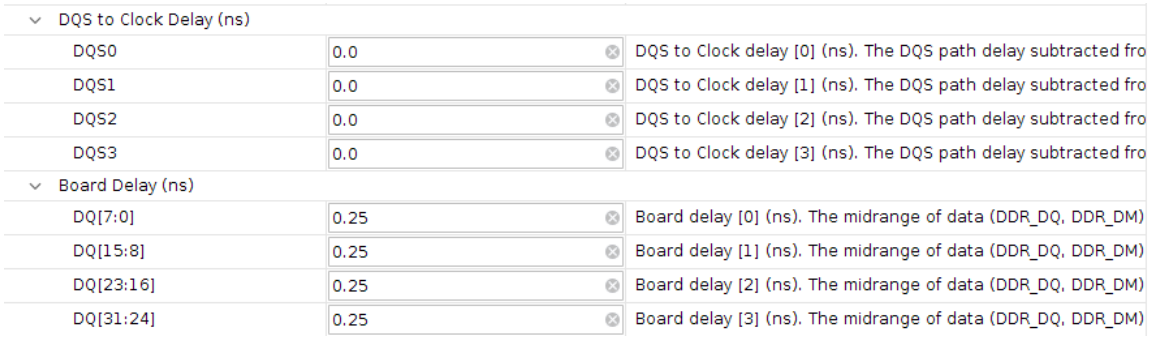


Figure 18 – MiniZed Board Training Details

12. Click **OK** to finish configuration of the Zynq PS.

13. **Save** the Block Design.

Questions:

Answer the following questions:

- *Where can the DDR interface speed be set?*

_____

- *Where did Vivado get the Memory Part Configuration Settings? Where would you get them for a custom part?*

_____

- *What is the maximum speed the DDR3 interface can run at? Extra Credit: What is the slowest?*

_____

Experiment 3: Build the hardware platform and export to SDK

A basic ARM hardware platform is now configured. The configuration includes clock and DDR controller settings. It also enables and maps a UART peripheral. Now we'll build the hardware platform and export to the Software Development Kit (SDK) so that an application can be developed.

Experiment 3 General Instruction:

Add a top-level module for the design. Export the hardware to SDK.

Experiment 3 Step-by-Step Instructions:

1. To validate our Zynq block design, click the **Validate Design** button.

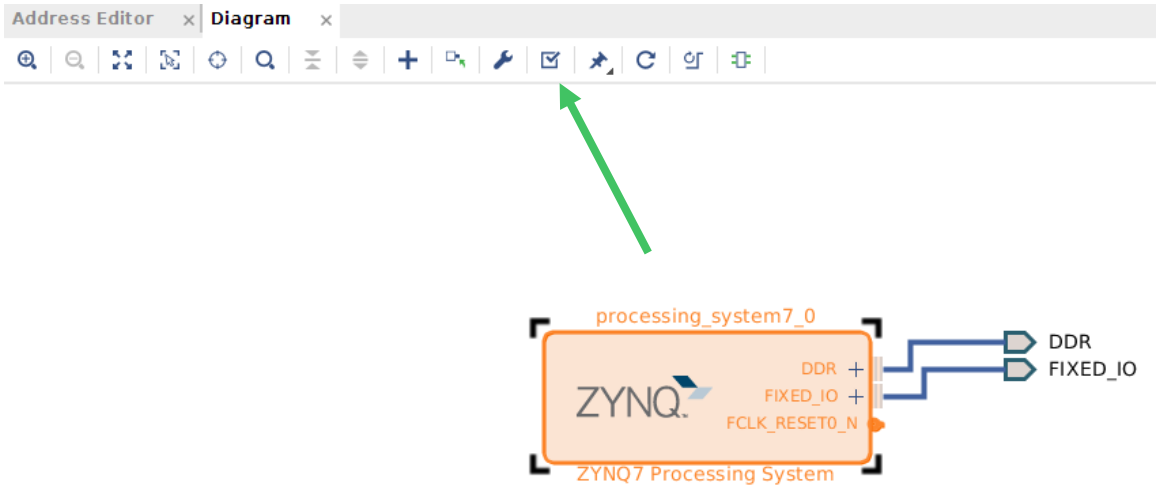


Figure 19 - Validate Block Design

2. If validation is successful, select the Sources Tab in the Vivado *Design* window, right-click on the **Z_system.bd**, then select **Create HDL Wrapper**.

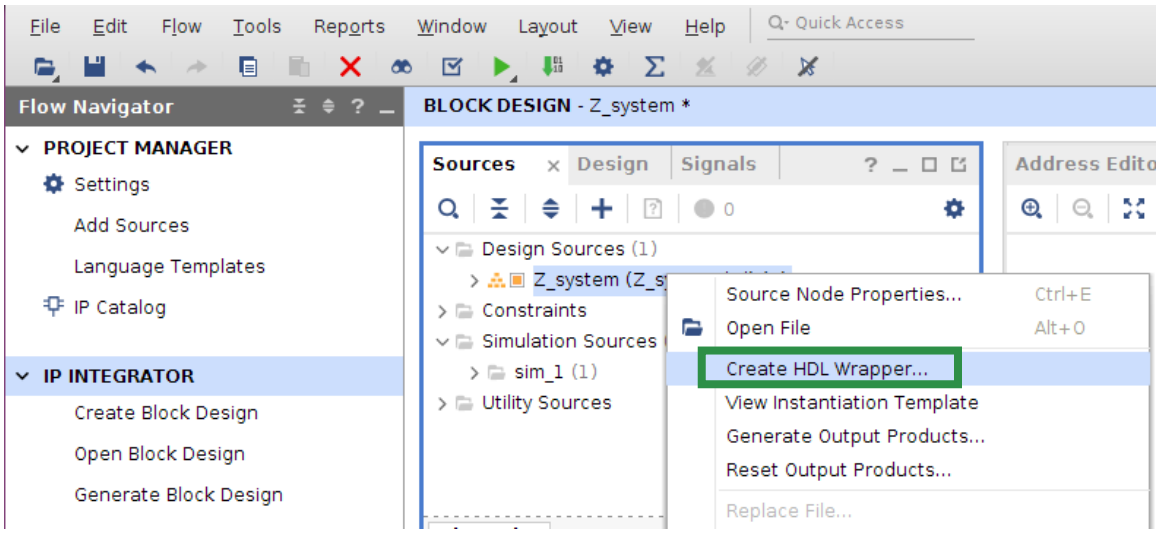


Figure 20 - Create HDL Wrapper

3. Vivado can manage your top-level HDL wrapper for you. Alternatively if this block design is a subset of a larger project, an editable wrapper will be created for instantiation into that project. For this tutorial, we will let Vivado manage our wrapper. Select **OK**.

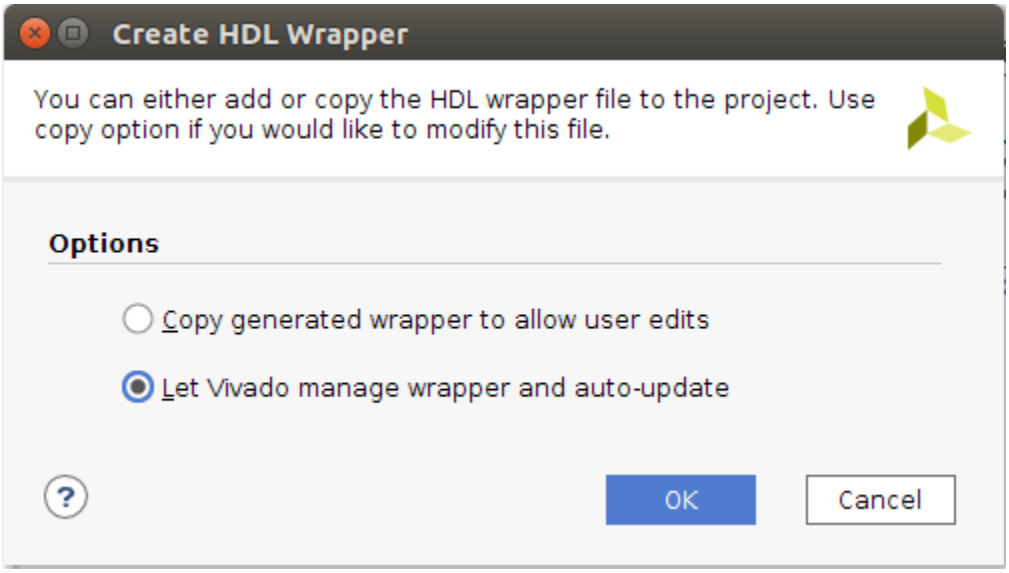


Figure 21 - Create HDL Wrapper Options

4. When completed, expand *Design Sources* and double-click on the newly created top-level wrapper, **Z_system_wrapper.vhd.** You'll see the Z_system instantiation.

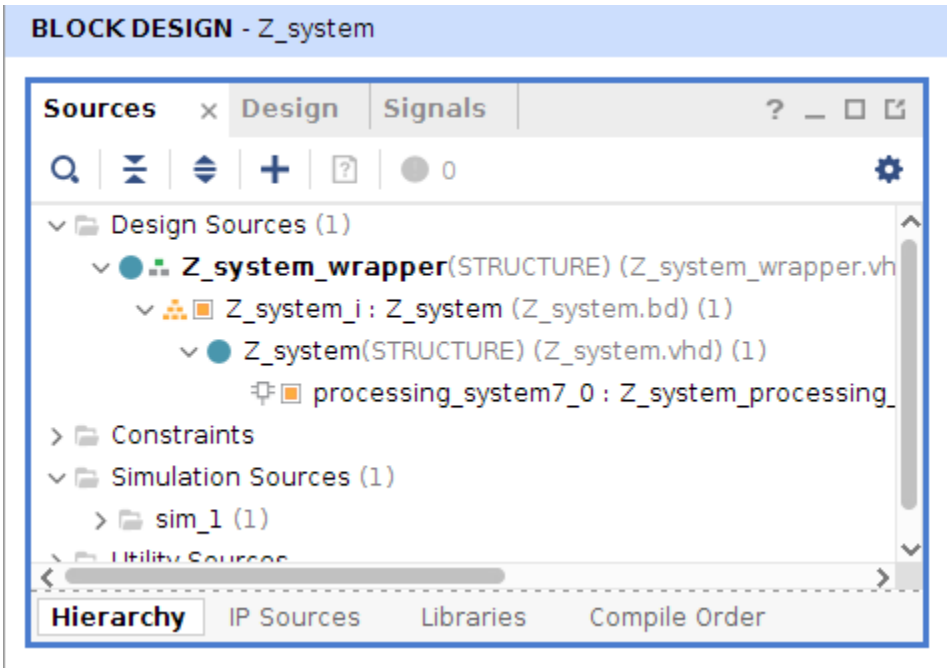


Figure 22 - Design Sources

5. Our design is now ready to be built. Click **Generate Bitstream** from the *Flow Navigator* pane. Note: This is not required as we are not utilizing the PL. However most designs will, thus its good practice to do this.

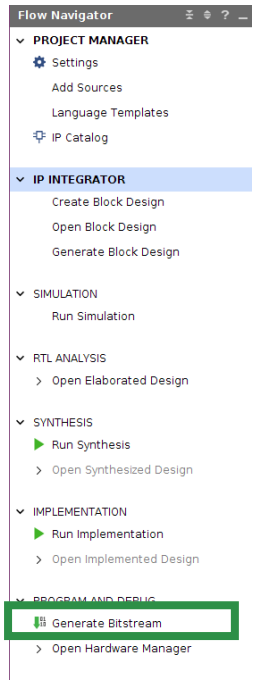


Figure 23 – Generate Bitstream

6. Vivado will warn that no synthesis or implementation results exist and ask to launch these steps. Click **Yes**. You can safely ignore any warnings about an **AXI BFM License** in the Messages Tab at the bottom.

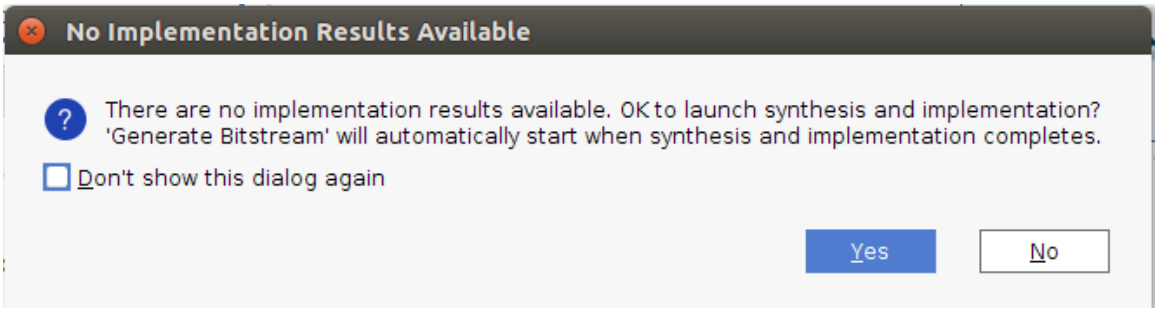


Figure 24 - Launch Synthesis and Implementation

7. You will then be prompted once again to launch the synthesis or implementation runs in a Launch Runs window. **Select** OK.

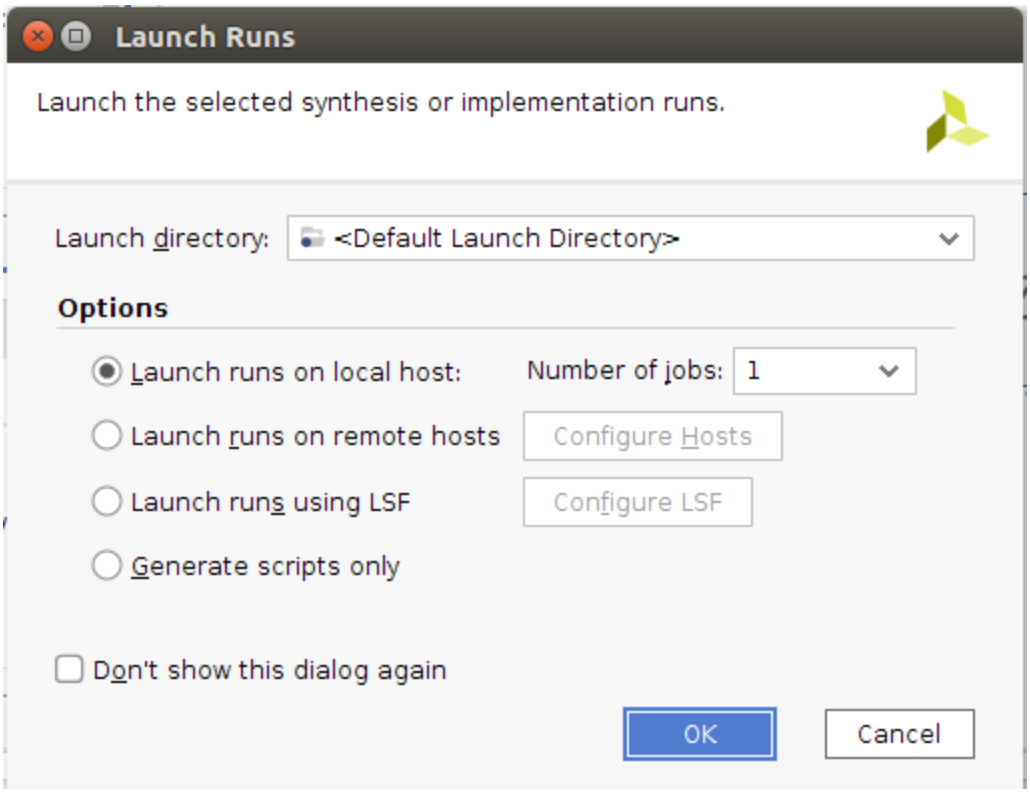


Figure 25 - Launch Runs

8. This will take a few minutes depending on your PC. When Bitstream Generation has completed, select **Open Implemented Design**. Click **OK**. Opening the implemented design enables Vivado to export the bitstream to SDK.

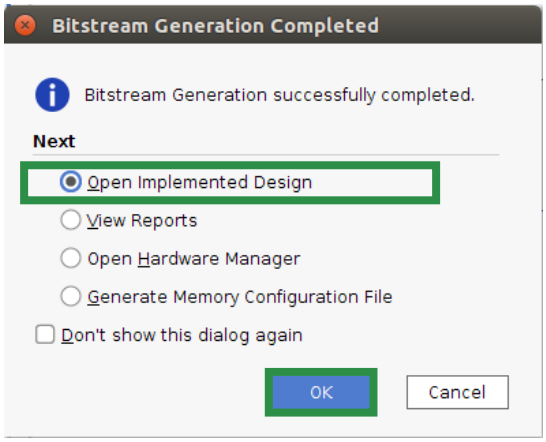


Figure 26 - Open Implemented Design

9. From the pull-down menus at the top of Vivado, select **File → Export → Export Hardware…**

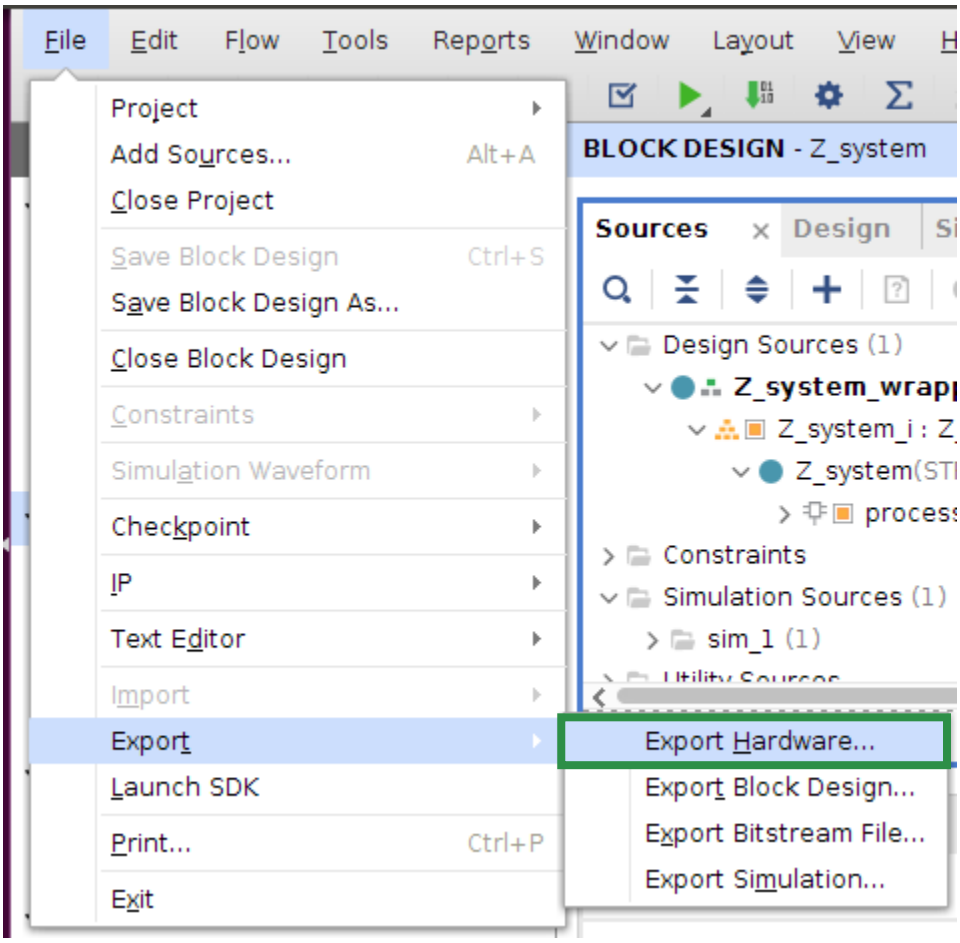


Figure 27 - Export Hardware

10. In the next window, check the **Include bitstream** box and click OK.

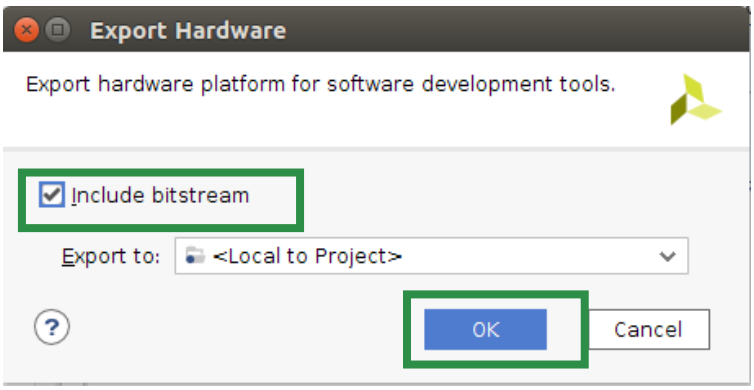


Figure 28 - Export Hardware

11. From the pull-down menus at the top of Vivado, select **File → Launch SDK**.

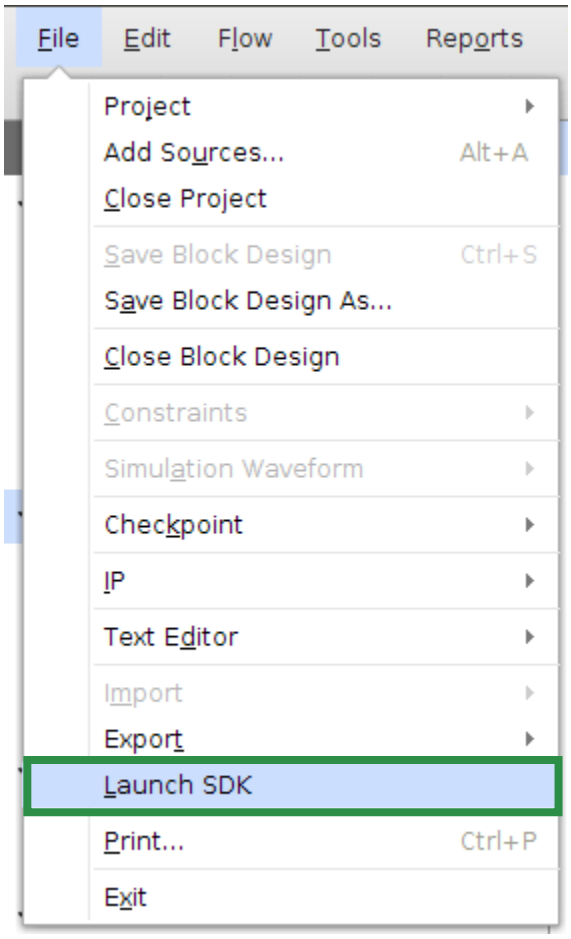


Figure 29 - Export Options

12. Click OK to accept default **Exported location** and **Workspace**.

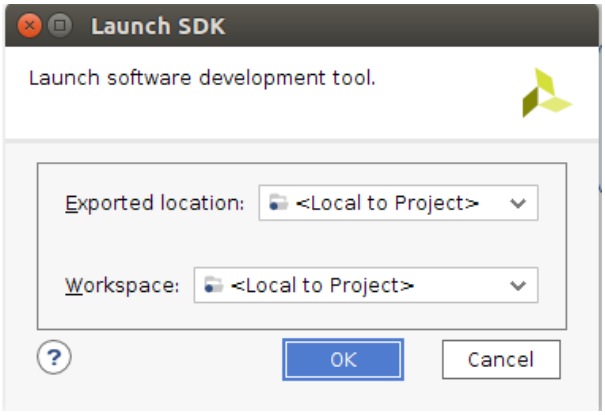


Figure 30 - Launch SDK Dialog Box

When using Windows, a Firewall security alert may appear, just select “allow access”.

13. Using a file explorer, open the SDK_Workspace directory to see what files were exported.

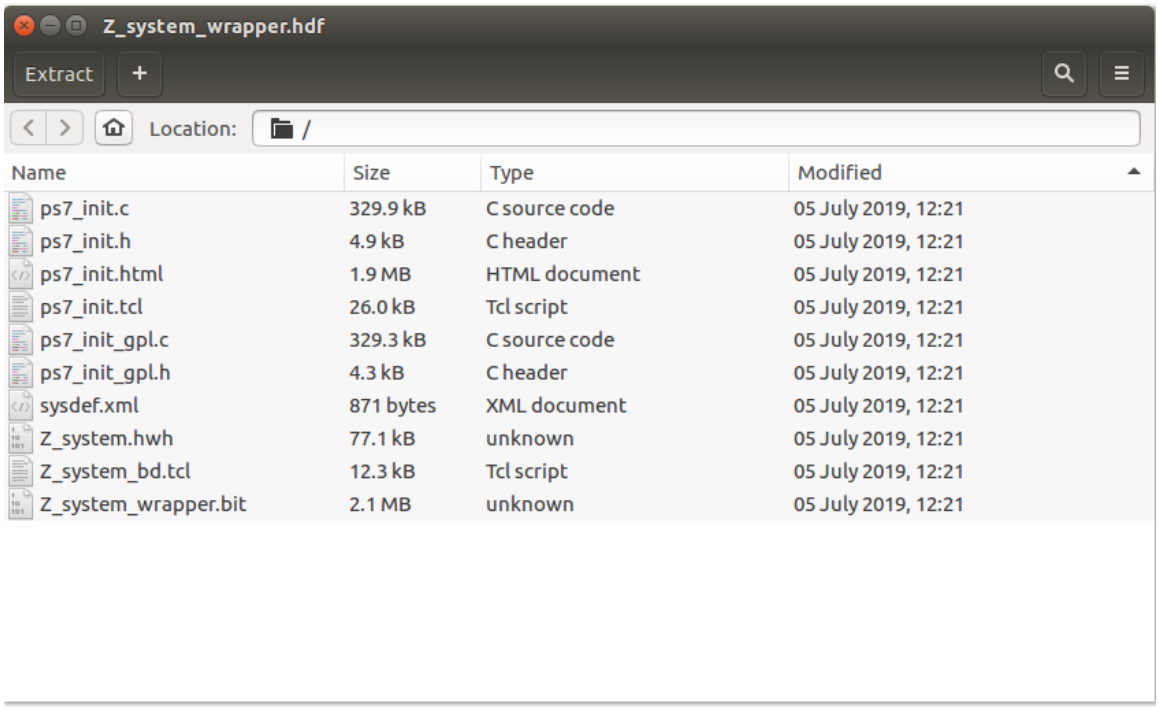


Figure 31 – Exported and Extracted Files for SDK

The Vivado design tool exported the Hardware Platform Specification for your design (system.hdf in this example) to SDK. The other files that you see were extracted from the .hdf by SDK. Note that the .hdf is actually a ZIP file which can be viewed in an extraction tool like 7-zip or archive manager.

The system.hdf file opens by default when SDK launches. The hardware platform's address map is read from this file and shown by default in the SDK window.

The ps7_init.c and ps7_init.h files contain the initialization code for the Zynq Processing System and initialization settings for DDR, clocks, PLLs, and MIOs. SDK uses these settings when initializing the processing system so that applications can be run on top of the processing system. The ps7_init.html displays these settings in an easy to read webpage format.

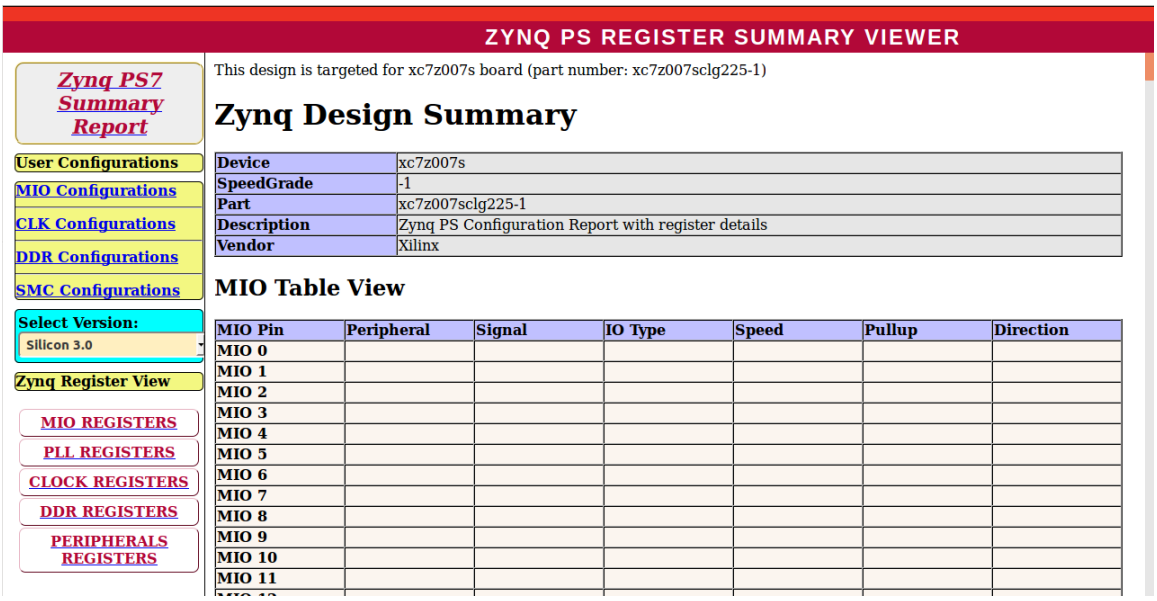


Figure 32 - ps7_init.html (MiniZed)

One more file is created, Z_system_wrapper.bit. This file is the PL bitstream generated when we implemented our design. Currently there is nothing in the PL however a blank bitstream is created that will initialize the PL.

Experiment 4: Create and Run a Hello World application

In this experiment, you will use SDK to create and run a simple Hello World application.

Experiment 4 General Instruction:

Create the Standalone BSP. Generate and run the Hello World application.

Experiment 4 Step-by-Step Instructions:

1. Select **File → New → Board Support Package**.
2. Accept the default settings for the standalone BSP OS. Click **Finish**.

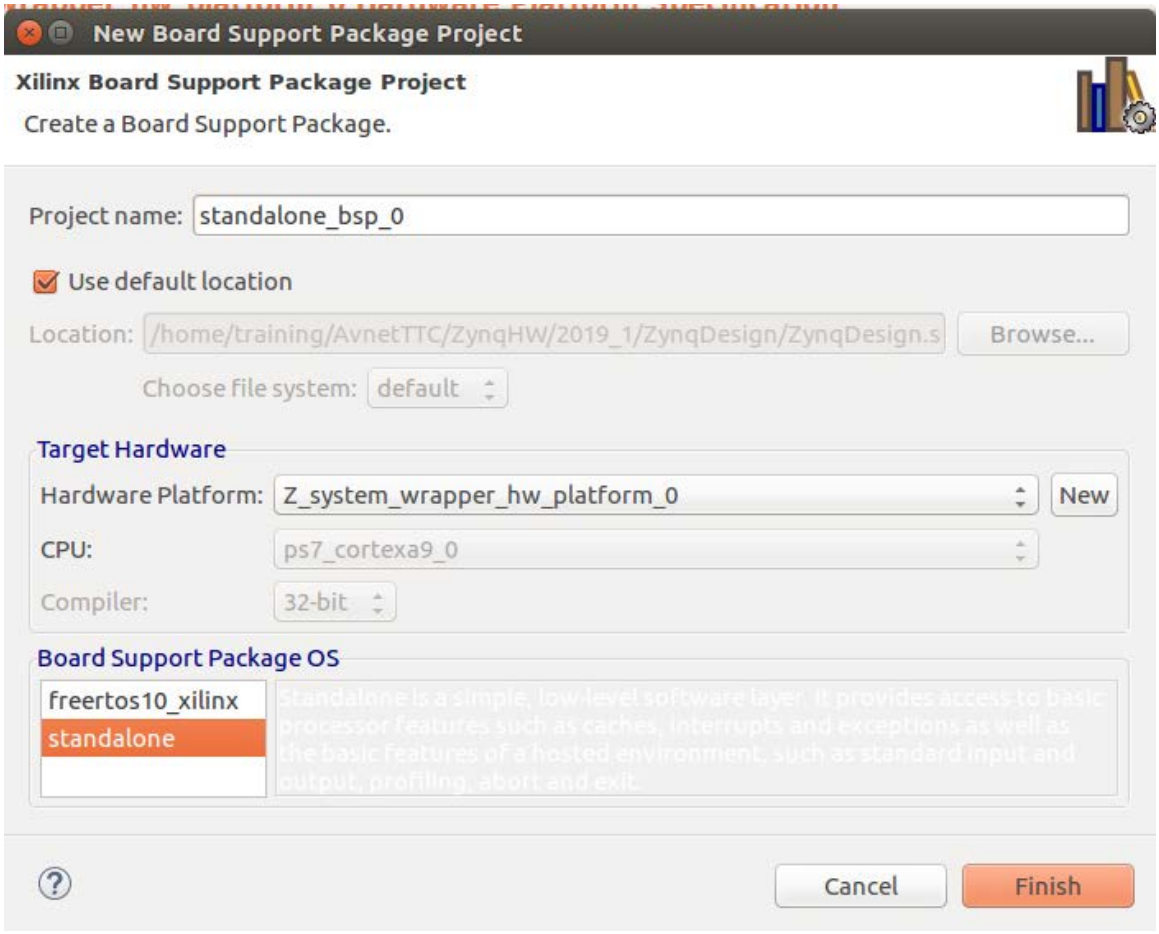


Figure 33 – Standalone BSP

3. Click **standalone** in the **Board Support Package** window. Note that the **stdin** and **stdout** are automatically set to the ps7_uart_1 peripheral, which is correct.

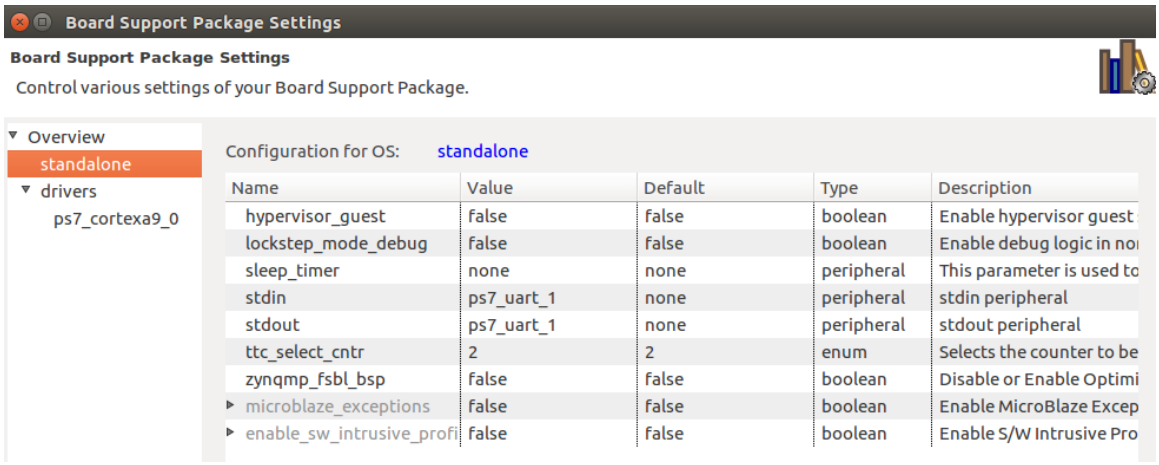


Figure 34 – stdin and stdout settings

4. Click **Overview**. No changes will be made to the BSP settings. None of the Supported Libraries are needed for this experiment. Click **OK** to accept the defaults and close this dialog. Note: It may take a minute to build the BSP.

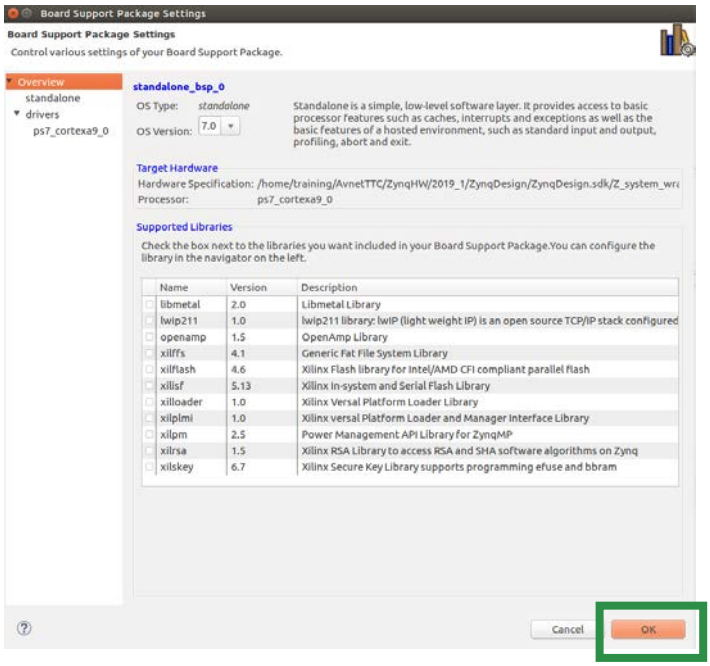


Figure 35 – BSP Settings

Based on the default settings in SDK, the BSP will automatically be built once added to the project. This may take a minute to compile the new BSP. The standalone_bsp_0 is now visible in the *Project Explorer*.

Expand **standalone_bsp_0** under the *Project Explorer*.

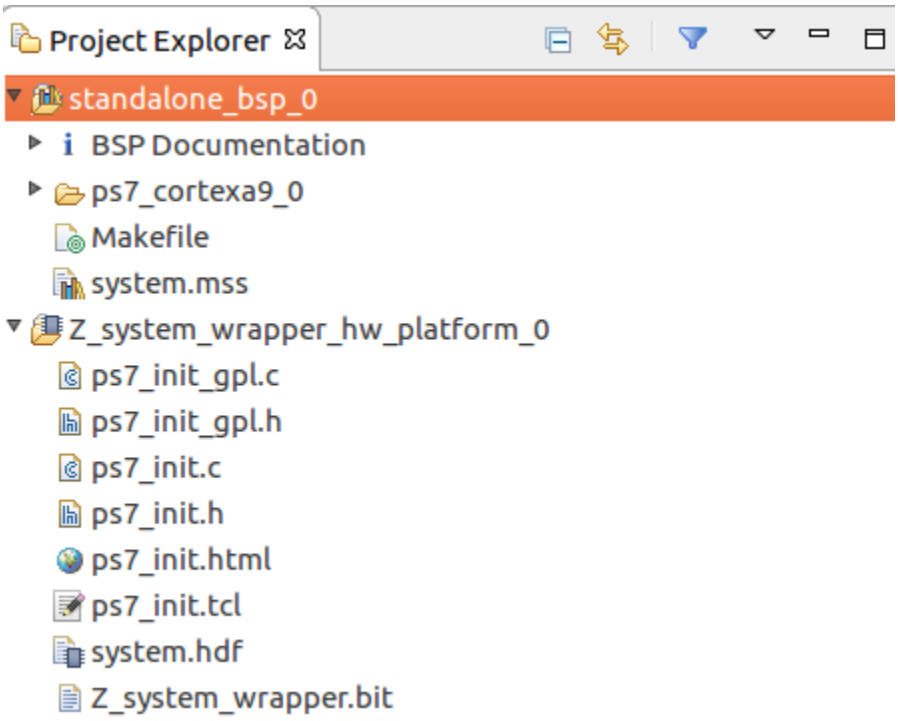


Figure 36 - BSP Added to the Project

5. In SDK, select **File → New → Application Project**.
6. Enter the *Project Name* of **Hello_World** and select *Use existing* BSP, **standalone_bsp_0**. Click **Next >**.

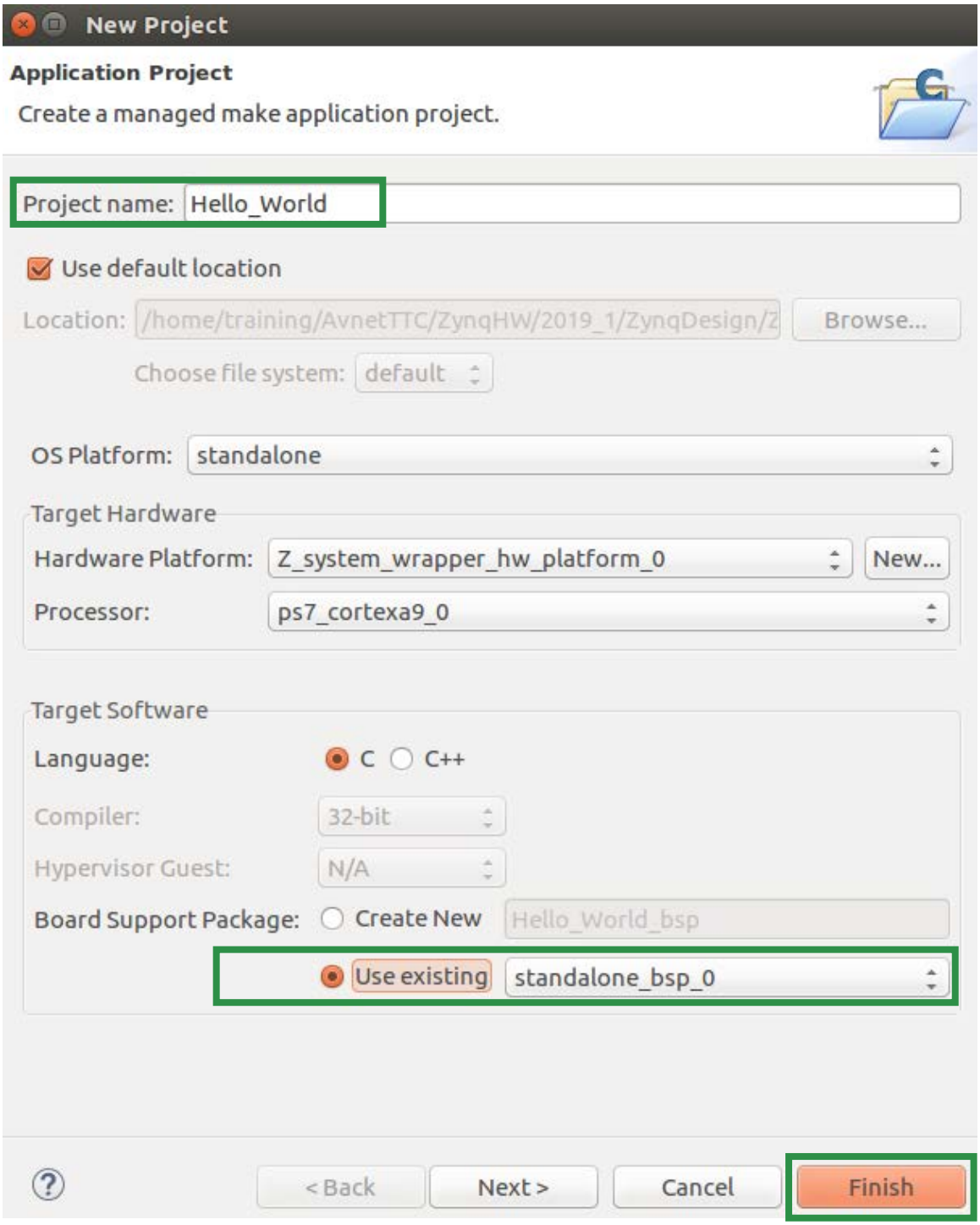


Figure 37 - Application Project Settings

7. Select **Hello World** from the *Available Templates* field. Click **Finish**.

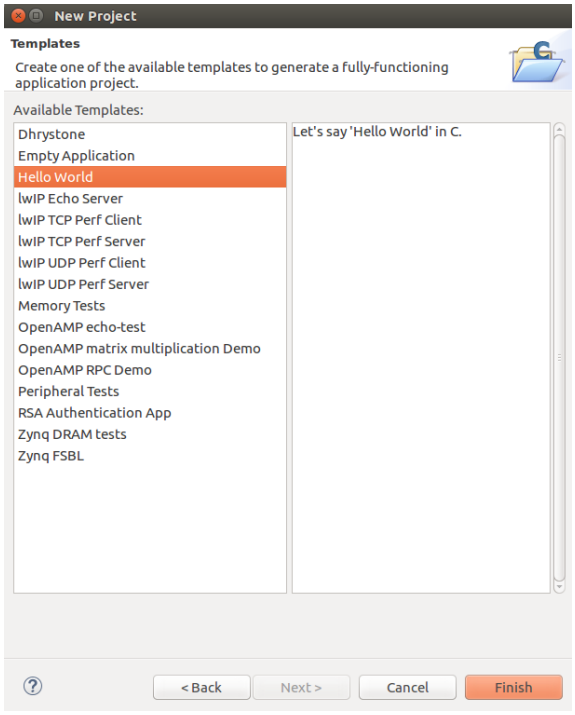


Figure 38 – New Application: Hello World

Hello_World will automatically build (click on 'Console' tab):

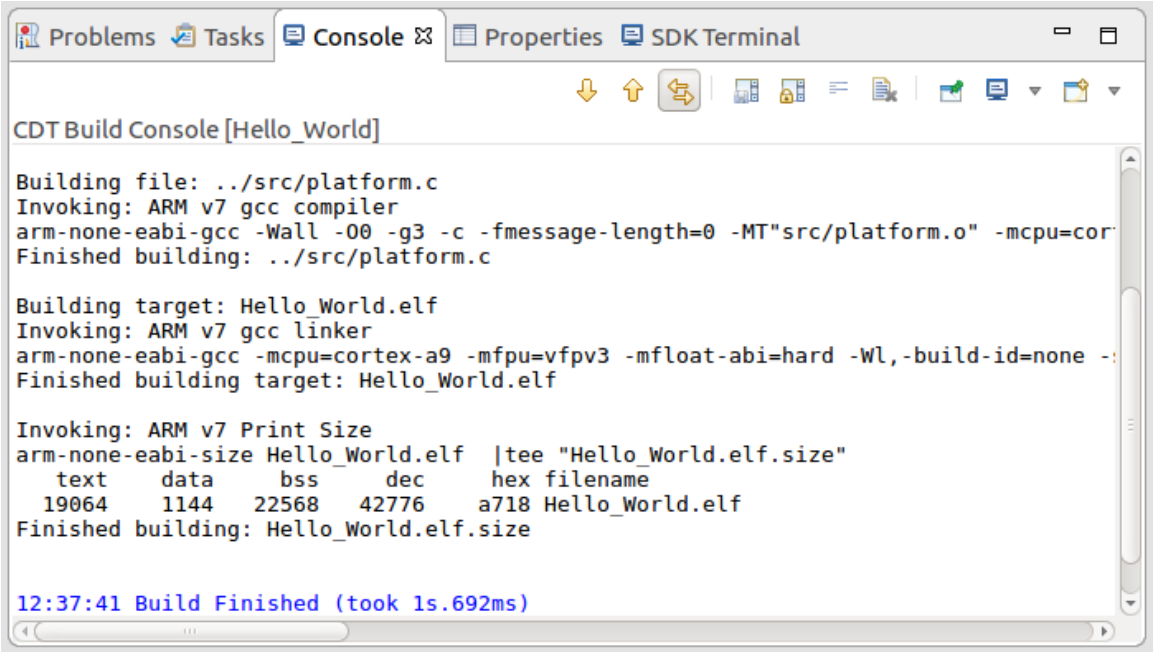


Figure 39 – Hello World Application Automatically Built

Notice that the Hello_World application is now visible in Project Explorer. By default, SDK will build the application automatically after it is added.

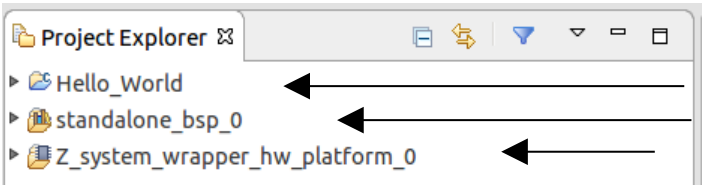


Figure 40 – Project Explorer View with Hello World C Application Added

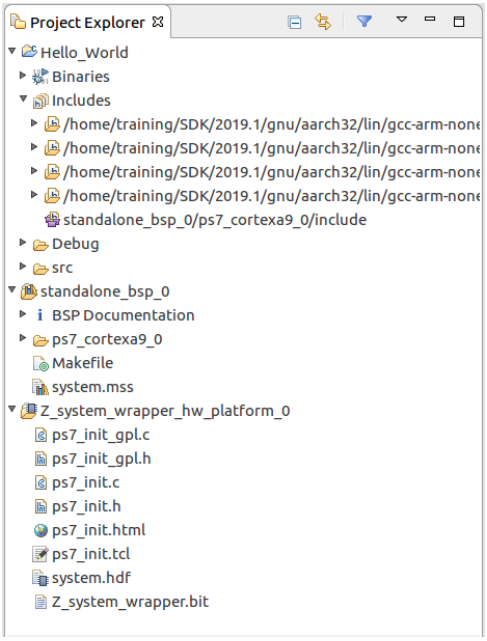
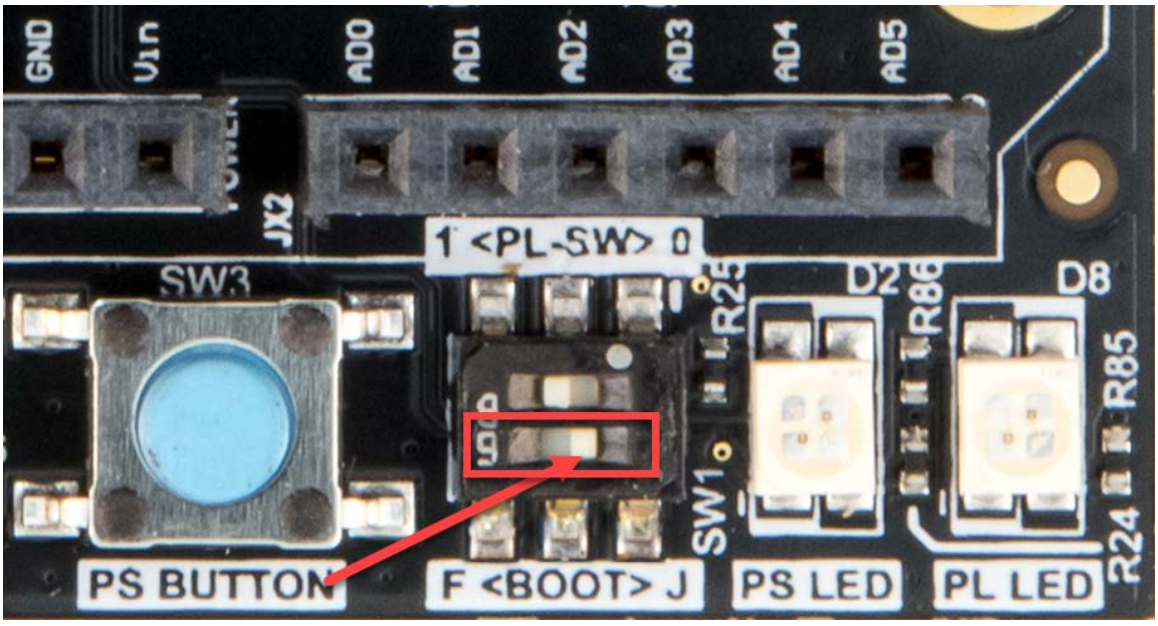


Figure 41 – Hello_World Application Expanded

8. Now we are going to set up our hardware. Set Boot Mode jumpers to **Cascaded JTAG Mode:**

    a. MiniZed – Set outside dip switch towards J (JTag Boot)



MiniZed

Figure 42 –MiniZed Boot Selection

9. Connect the MiniZed USB-JTAG/UART port J2 to your Windows PC. It should automatically install the proper drivers, giving you a confirmation as shown below:

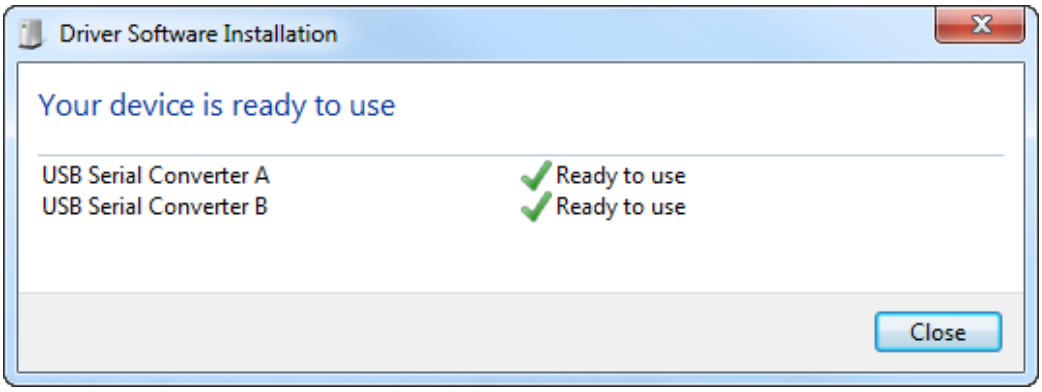


Figure 43 – MiniZed USB-JTAG/UART Installed Correctly

10. In the rare circumstance that the drivers are not auto-installed, then you must manually install the driver for the FTDI FT2232H device. Visit the FTDI website and download the appropriate driver for your operating system. http://www.ftdichip.com/Drivers/VCP.htm

    a. Make sure the MiniZed is unplugged from the PC. Unzip and install the driver.

    b. Reboot your PC then plug in the MiniZed.

11. To determine the COM port for the USB-UART, in a terminal window run the command lsusb /dev/ttyUSB*

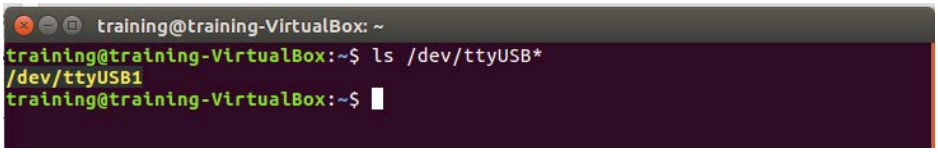


Figure 44 - Find the COM Port Number for the Serial Device

12. Open Terminal, such as **GTKWave**, and set the active **COM port** to the COM setting using setup/Serial Port for your board and set the **Baud Rate at 115,200**.

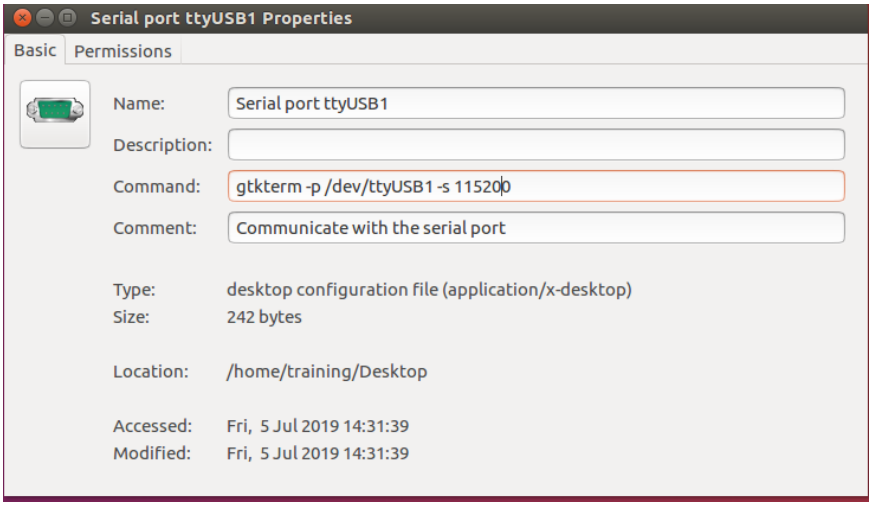


Figure 45 - Find the COM Port Number for the Serial Device

13. Back in SDK, right-click on the **Hello_World** application and select **Run As → Run Configurations…**

14. Select **Xilinx C/C++ Application (System Debugger)** and then click the '**New'** icon  .

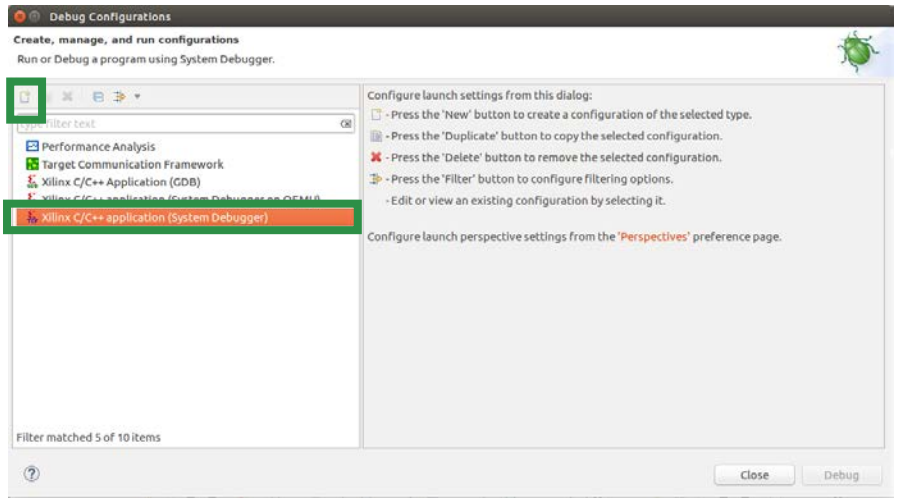


Figure 46 – Create a New Xilinx C/C++ ELF Run Configuration

SDK creates the new Run Configuration and automatically assigns a name to the configuration <*application_name*> *Debug*, which in this case is *Hello_World Debug*.

15. **Browse** through the tabs here to see what's available for this configuration.

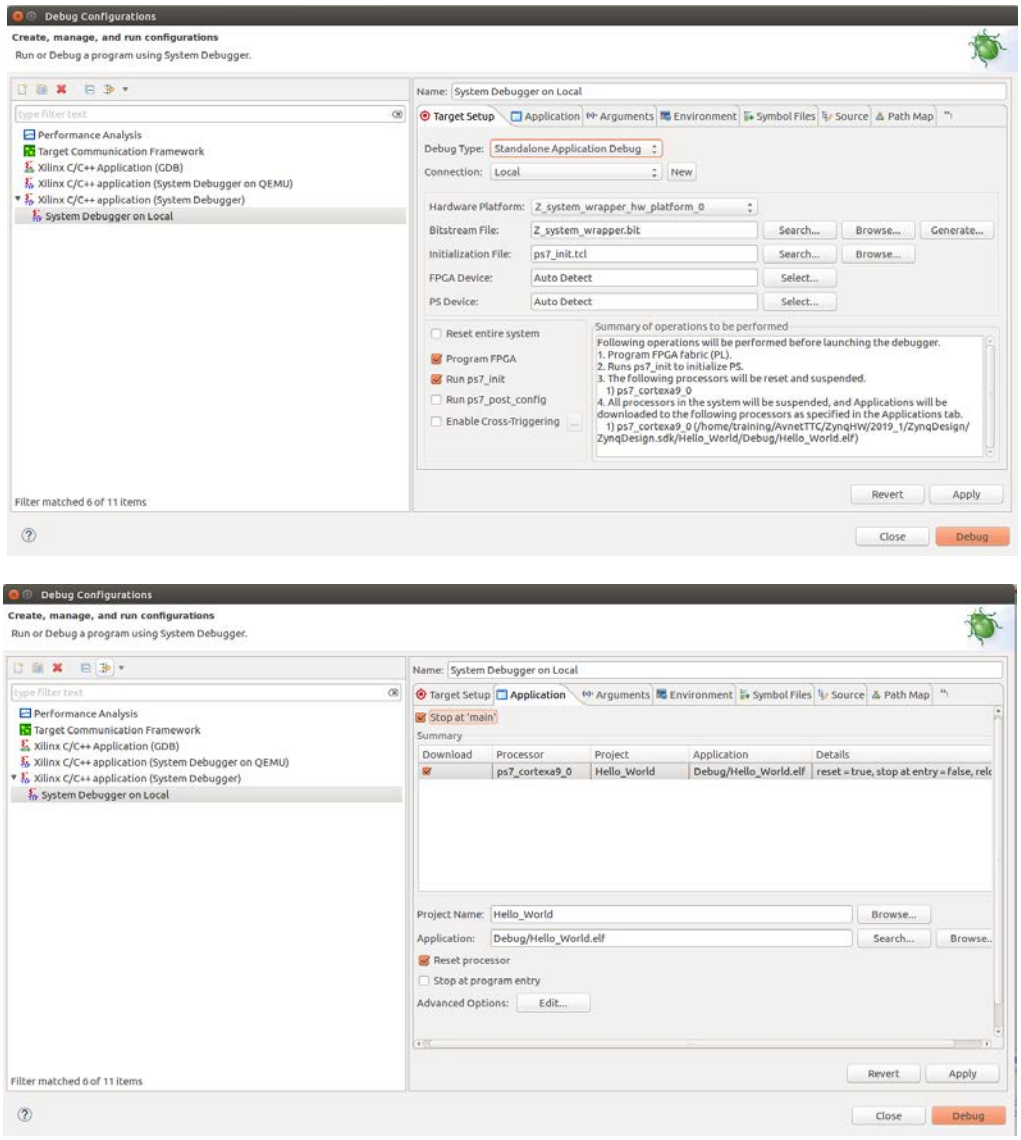


Figure 47 – New Run Configuration

16. Click **Run**.

17. If the FPGA Configuration pops up, click **Yes** to continue.

The tools will now initialize the processor, download the Hello_World.elf to DDR, and then run Hello_World. This takes approximately 5 seconds to complete, depending on the USB traffic in your system. You can follow the progress in the lower right corner of SDK.

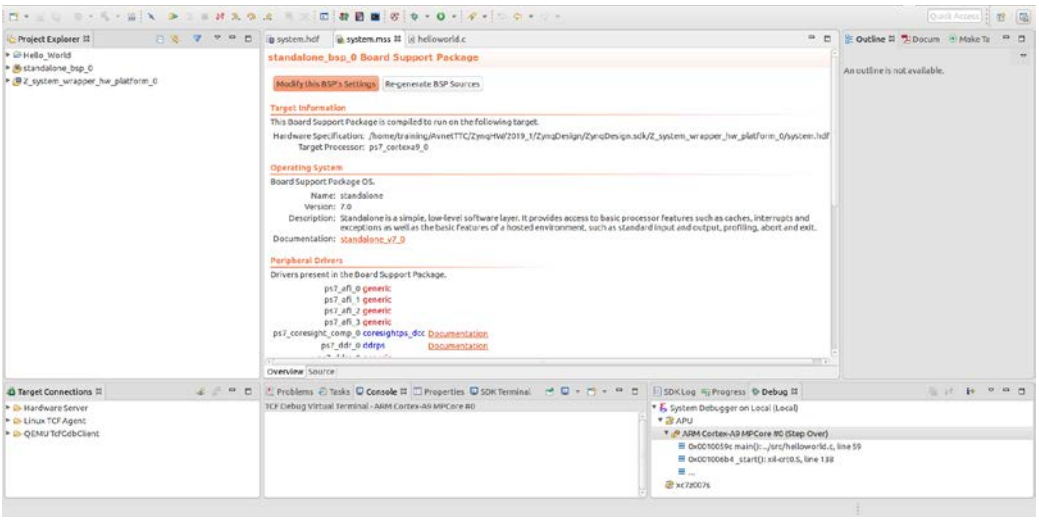


Figure 48 – Launching Hello_World Progress

SDK will download the Hello World ELF to the DDR3 and the ARM begins executing the code.

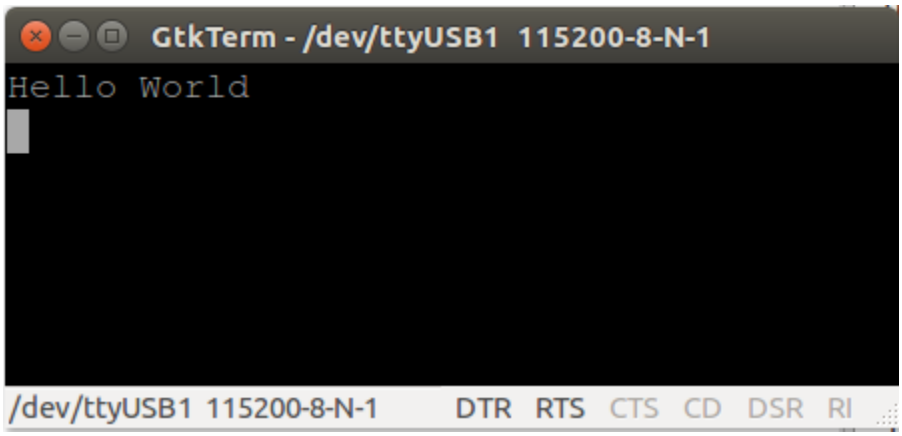


Figure 49 – Hello World Complete

18. You can look at the output in **Tera Term**. **Close** SDK.

Questions:

Answer the following questions:

- *Does the Hello World C source include Zynq initialization?*

_____

- *How did the Zynq get initialized in this Hello World experiment?*

_____

_____

_____

This concludes Lab 2.

Revision History

Date	Version	Revision
6 Nov 13	02	Initial Draft
19 Nov 13	03	Pilot updates
30 Oct 14	04	Updated for Vivado 2014.3
05 Jan 15	05	Updated for Vivado 2014.4
04 Mar 15	06	Finalize for Vivado 2014.4
17 Mar 15	07	Minor edits for release
Oct 2015	08	Updated to Vivado 2015.2
July 2016	09	Updated to Vivado 2016.2
May 2017	10	Updated to Vivado 2017.1 and added MiniZed
June 2017	11	Rebranded and Updated to 2017.1 and MiniZed only
Jan 2018	12	Updated to Vivado 2017.4
July 2019	13	Updated to Vivado 2019.1

Resources

www.minized.org

www.xilinx.com/zynq

www.xilinx.com/sdk

Appendix – Calculating DDR 3 PCB Lengths (MiniZed)

Zynq allows for up to 4 memory devices to be configured for DDR3 4x8 fly-by topology. MiniZed is configured for DDR3L 1x16 topology.

With the DDR3L chip, the DQS's and DQ's are all point-to-point. Those values can easily be obtained from a simple trace length report. MiniZed only has one DDR3L clock pair. The CLK trace length in the report is going to be the total length.

First, let's calculate the length values that we can easily get from a trace length report.

1. A trace length report for MiniZed is provided in the **`Support_documents`** folder. Open MiniZed_Rev1_PCB_Trace_Length_Report.txt now.

All the DDR3L signals are prefixed in this report with "D3L_" so these are easy to search. For differential pairs, calculate the average length of the P and N traces. For the data busses, calculate the average of the Minimum and Maximum values in the set. Each chip services two byte groups, so you'll find 16 DQ's and 2 DQS's per chip, but only one CLK. Therefore, the CLK lengths for byte groups 0 and 1 are the same.

2. Use the trace length report to calculate and then enter the trace lengths for the table below.

Pin Group	ZedBoard Traces Used in Calculation	Length (mm)
CLK0	D3L_CK_N, D3L_CK_P	
CLK1	D3L_CK_N, D3L_CK_P	
DQS0	D3L_LDQS_N, D3L_LDQS_P	
DQS1	D3L_UDQS_N, D3L_UDQS_P	
Q[7:0]	D3L_DQ [7:0]	
DQ[15:8]	D3L_DQ [15:8]	

3. Now we will open the Board Delay Calculation Table in Vivado. Go to the DDR configuration tab in your Zynq Customization page. Under training Board Details change User Input to Calculated. The work sheet should open. Please enter all of your calculated values.

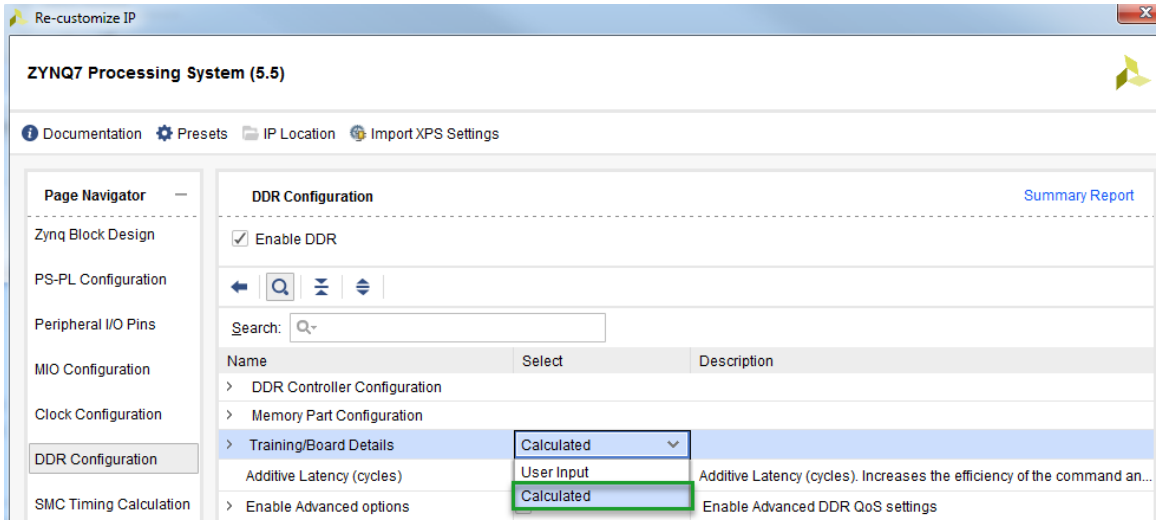


Figure 50 -- Open DDR Calculation Page

4. You may be wondering at this point why we did not calculate for CLK2, CLK3, DQS2, DQS3, DQ[23:16], or DQ[31:24]. The reason being is that the MiniZed as stated above is DDR3L 1x16 topology so those additional values are not applicable for the 16 bit interface. For all of these you can simply enter 0.

The completed worksheet should appear as shown below.

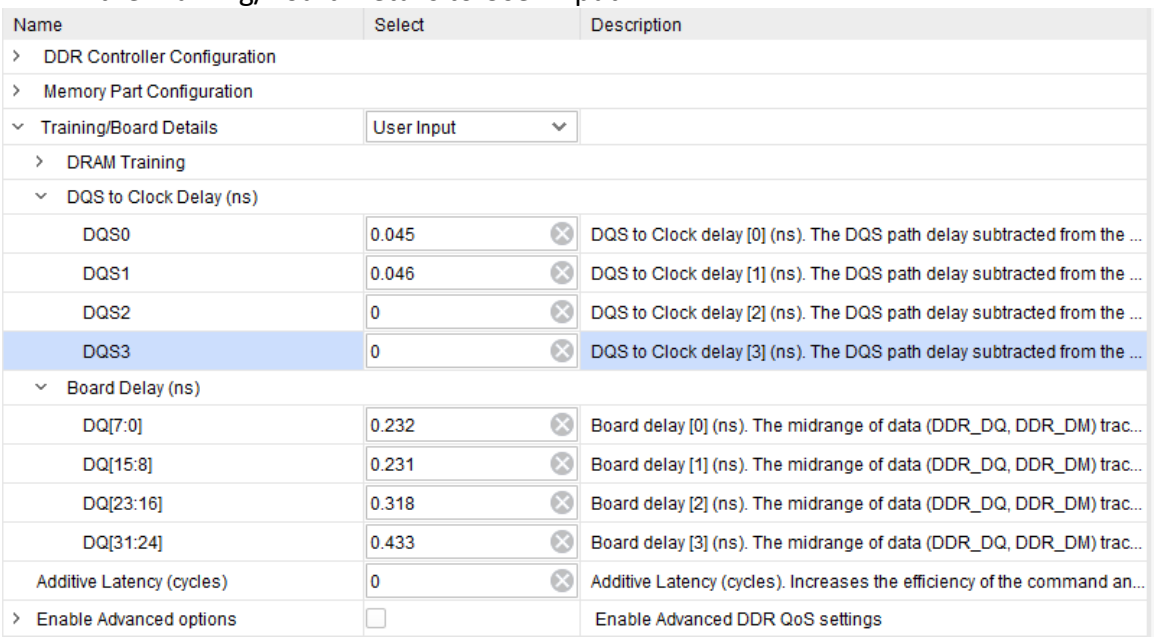
Board Delay Calculation Table.

Pin Group	Length (mm)	Package Delay (ps)	Propagation Delay (ps/inch)	DQS to CLK ...	Board Delay (ns)
CLK0	26.6907	86.1835	160		
CLK1	26.6907	86.1835	160		
CLK2	0	86.1835	160		
CLK3	0	86.1835	160		
DQS0	20.26245	81.244	160	0.045	
DQS1	23.9975	57.044	160	0.046	
DQS2	0	520	160	-0.434	
DQS3	0	700	160	-0.614	
DQ[7:0]	20.91665	77.166	160		0.232
DQ[15:8]	24.47585	53.995	160		0.231
DQ[23:16]	0	550	160		0.318
DQ[31:24]	0	780	160		0.433

Figure 51 – MiniZed PCB Lengths Calculated and Entered

5. Now minimize the calculation table, and expand the Block Delay and Board Delay sections. You will notice your values have populated through. You will also notice

the DQS2 and DQS3 are negative numbers. If we continue Vivado will complain about these negative values. Since MiniZed only utilizes a 16 bit DDR3L device, these number do not matter and can be set to 0. To do so please once again set the Training/Board Details to User Input.



Name	Select	Description
DDR Controller Configuration		
Memory Part Configuration		
Training/Board Details	User Input	
DRAM Training		
DQS to Clock Delay (ns)		
DQS0	0.045	DQS to Clock delay [0] (ns). The DQS path delay subtracted from the ...
DQS1	0.046	DQS to Clock delay [1] (ns). The DQS path delay subtracted from the ...
DQS2	0	DQS to Clock delay [2] (ns). The DQS path delay subtracted from the ...
DQS3	0	DQS to Clock delay [3] (ns). The DQS path delay subtracted from the ...
Board Delay (ns)		
DQ[7:0]	0.232	Board delay [0] (ns). The midrange of data (DDR_DQ, DDR_DM) trac...
DQ[15:8]	0.231	Board delay [1] (ns). The midrange of data (DDR_DQ, DDR_DM) trac...
DQ[23:16]	0.318	Board delay [2] (ns). The midrange of data (DDR_DQ, DDR_DM) trac...
DQ[31:24]	0.433	Board delay [3] (ns). The midrange of data (DDR_DQ, DDR_DM) trac...
Additive Latency (cycles)	0	Additive Latency (cycles). Increases the efficiency of the command an...
Enable Advanced options		Enable Advanced DDR QoS settings

Figure 52 -- DDR Calculated Delay Values

6. These are our calculated DDR3L delay values.

Answers

Experiment 1

- *What do you think is the purpose of EMIO?*

MIO pins are dedicated I/O in the Processing Subsystem. EMIO provide PL fabric access to PS peripherals, which can then be connected to PL I/O.

- *Why are the Peripherals not listed alphabetically in the I/O Peripherals Configuration tool?*

Peripherals on the top of the list should typically be connected first as they have fewer available MIO pin assignment options.

- *Extra Credit: If the Modem Signals are used with one of the UART peripherals, where must they be mapped?*

EMIO

Experiment 2

- *Where can the DDR interface speed be set?*

In the Clock Configuration window or the DDR Configuration window.

- *Where did Vivado get the Memory Part Configuration Settings? Where would you get them for a custom part?*

Vivado has a preset library of memory part details. With each release of Vivado, Xilinx adds new memory devices to this library. If you use a custom part, these values must be extracted from the specific DDR datasheet.

- *What is the maximum speed the DDR3L interface can run at? Extra Credit: What is the slowest?*

534MHz.

Slowest? It depends on the DDR3 memory device. Even though Vivado lists the slowest speed as 200MHz, the DDR3L memory device's internal PLL may not be able to run that slow. When a known memory device is selected, Vivado prevents users from setting a speed to slow. In this case, 303MHz (Tck_max = 3.3ns)

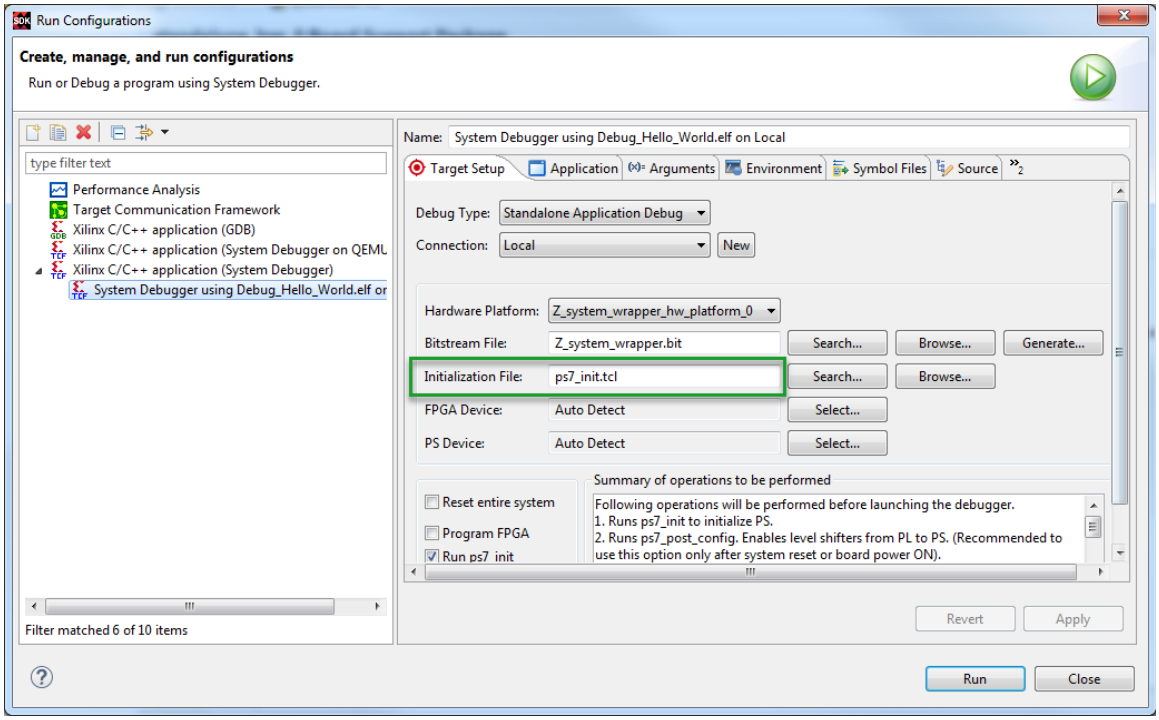
Experiment 4

- *Does the Hello World C source include Zynq initialization?*

No

- *How did the Zynq get initialized in this Hello World experiment?*

When you created the Run Configuration, there is a Target Setup tab. There is a field to point to an initialization TCL file. This is set by default to the ps7_init.tcl file that was created as part of the Export to SDK. Inside this TCL, you will find a number of XMD commands that initialize all the registers exactly how you specified in Vivado.



Appendix

Pin Group	ZedBoard Traces Used in Calculation	Length (mm)
CLK0	D3L_CK_N, D3L_CK_P	26.6907
CLK1	D3L_CK_N, D3L_CK_P	26.6907
DQS0	D3L_LDQS_N, D3L_LDQS_P	20.26245

DQS1	D3L_UDQS_N, D3L_UDQS_P	23.9975
Q[7:0]	D3L_DQ [7:0]	20.91665
DQ[15:8]	D3L_DQ [15:8]	24.47585