

Introduction to Zynq Hardware

Lab 2

PS Configuration Part 1 & Hello World



November 2013
Version 03

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).

Experiment 1 General Instruction:

Set the Bank voltages to LVCMOS 3.3V for Bank 0 and LVCMOS 1.8V 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.

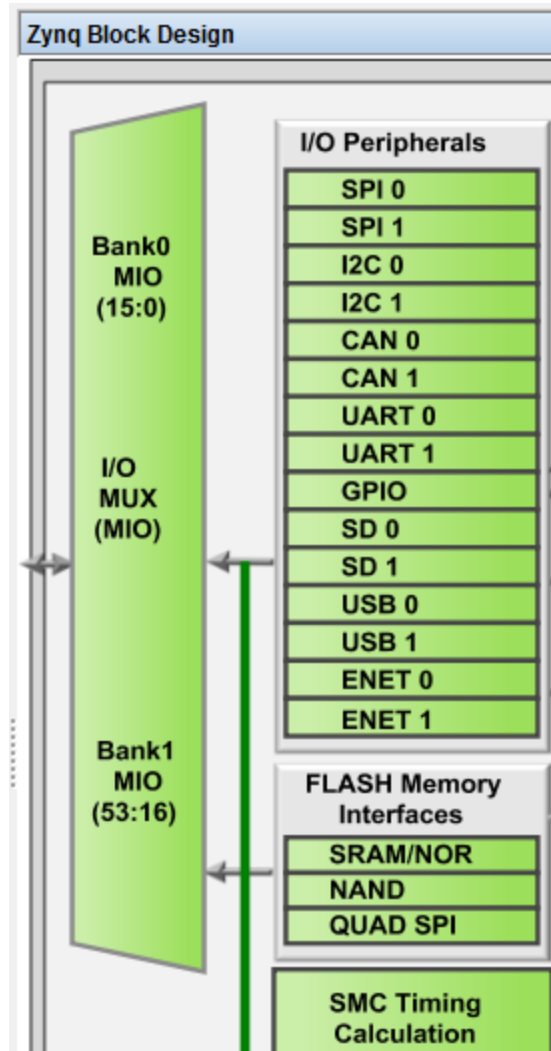


Figure 1 - Zynq I/O Peripherals

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

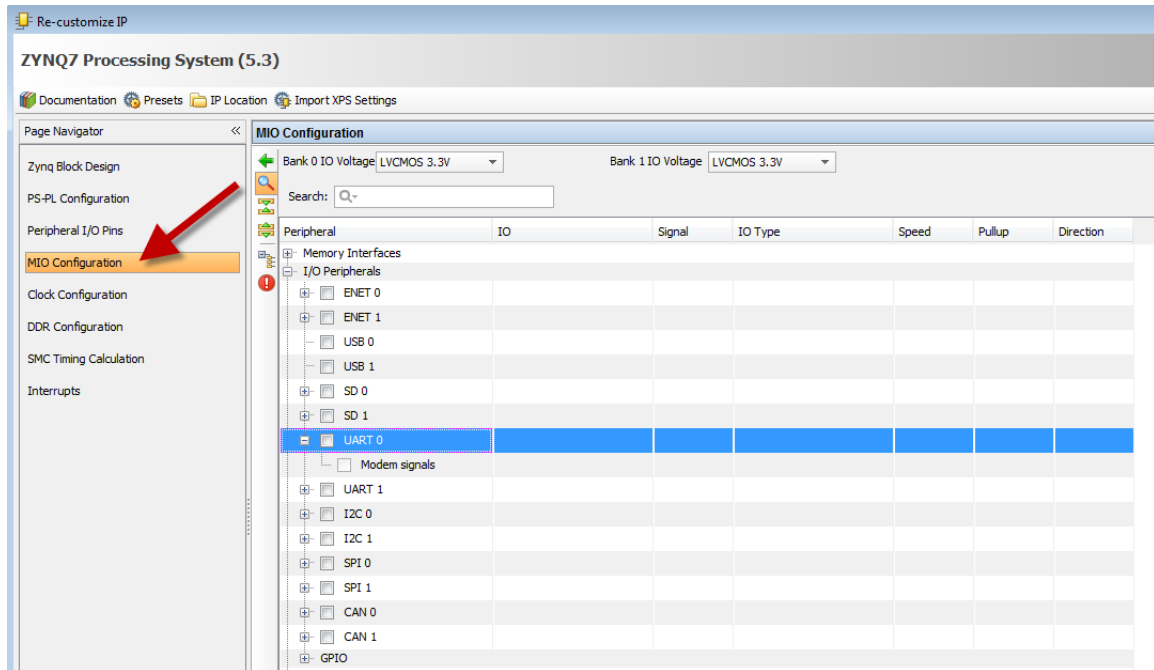


Figure 2 - MIO Configuration

3. At the top of the MIO Configuration window, the Bank Voltage settings are listed. Set Bank 0 I/O Voltage to **LVCMOS 3.3V** and Bank 1 I/O Voltage to **LVCMOS 1.8V**.

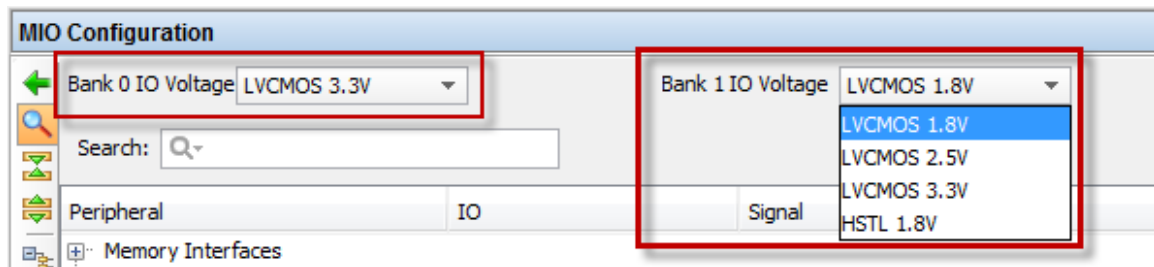


Figure 3 - MIO Bank Voltages

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. 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.

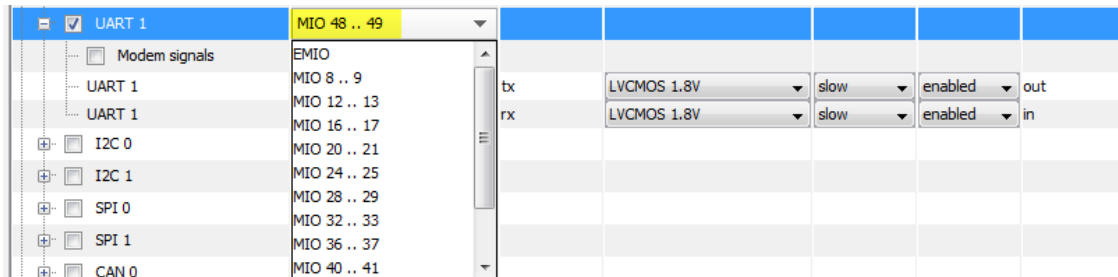


Figure 4 - UART1 Connection

Schematics are included in the support documents folder. If interested, open the schematics to verify the MIO pin connections to the on-board hardware.

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.

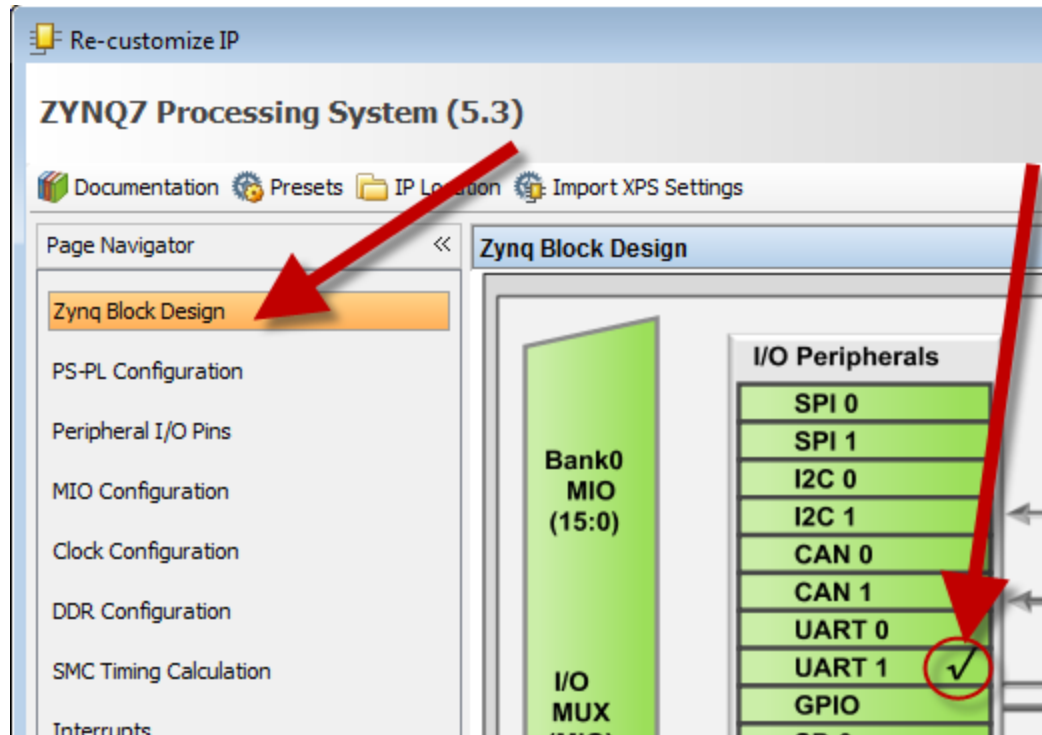


Figure 5 - UART1 MIO Connected

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

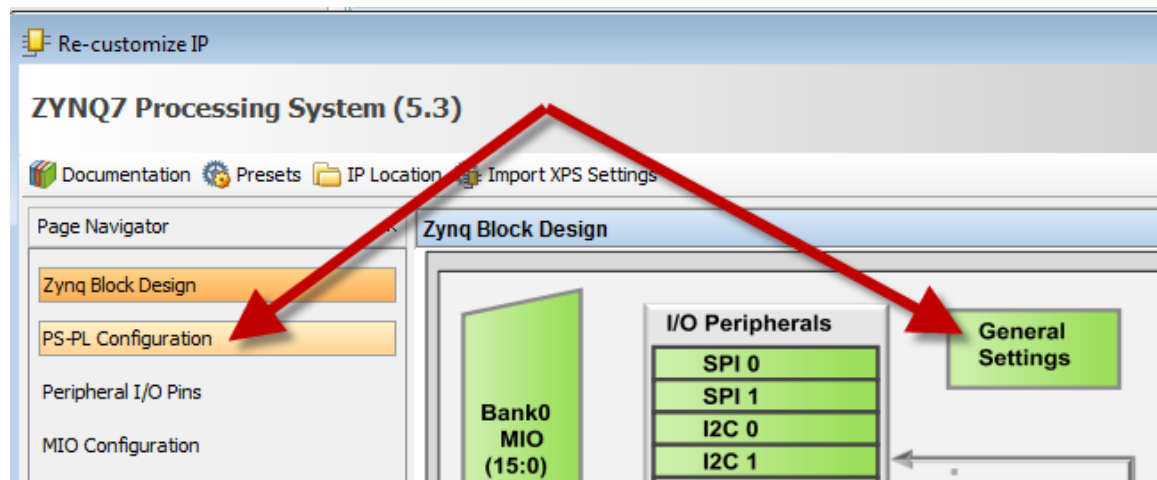


Figure 6 - General Zynq Settings

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

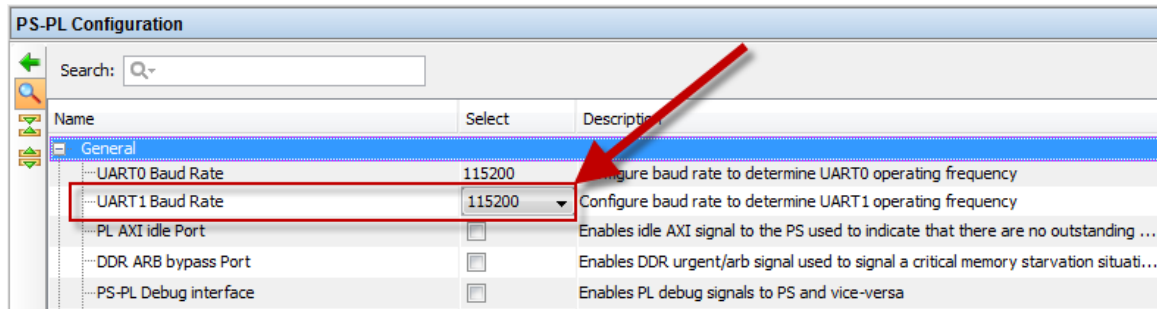


Figure 7 - UART1 Baud Rate Settings

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 DDR3 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 32-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.

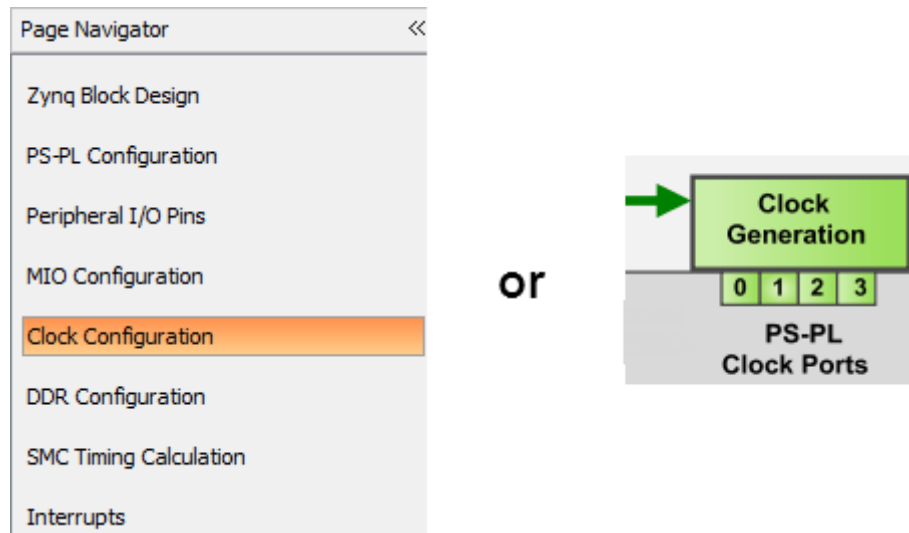


Figure 8 - Clock Configuration

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

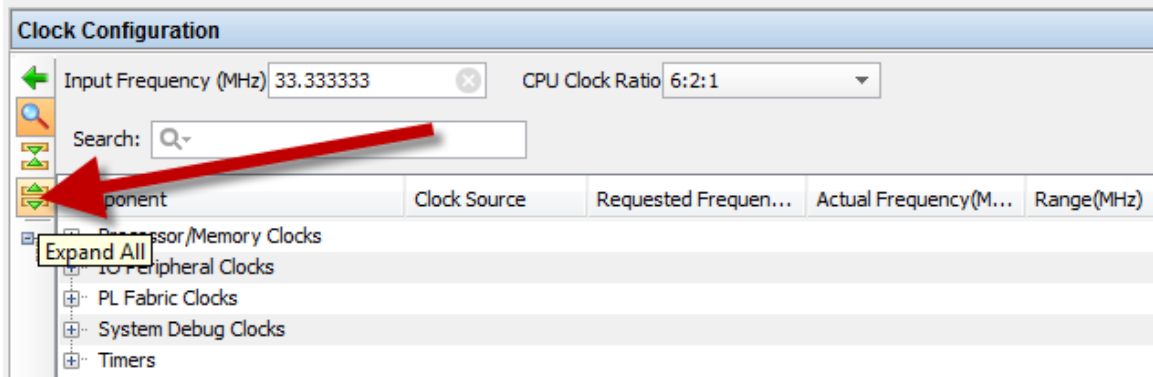


Figure 9 - Expand to see all clocks

3. For the most part, the default clock settings match ZedBoard and MicroZed.

Verify the following:

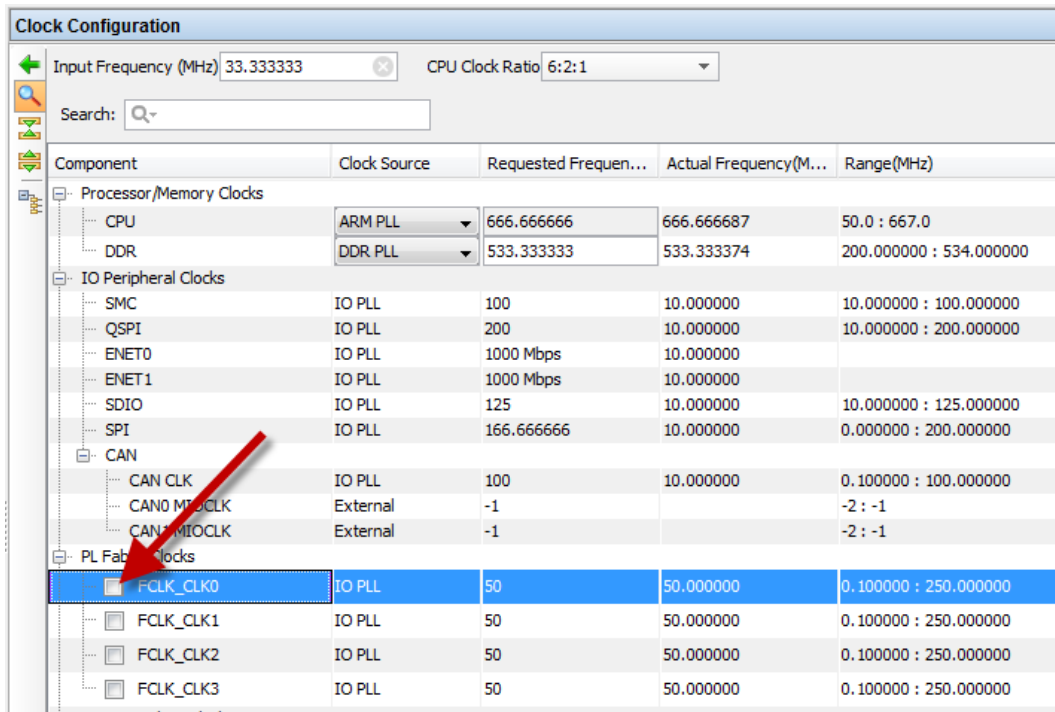
- Input frequency is 33.33333 MHz
- CPU frequency is 666.666666 MHz
- DDR frequency is 533.333333 MHz

The screenshot shows the 'Clock Configuration' window with the 'Processor/Memory Clocks' category expanded. The table below shows the settings for CPU and DDR.

Component	Clock Source	Requested Frequency (MHz)	Actual Frequency (MHz)	Range (MHz)
CPU	ARM PLL	666.666666	666.666687	50.0 : 667.0
DDR	DDR PLL	533.333333	533.333374	200.000000 : 534.000000

Figure 10 - Clock Settings

- 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.



Clock Configuration

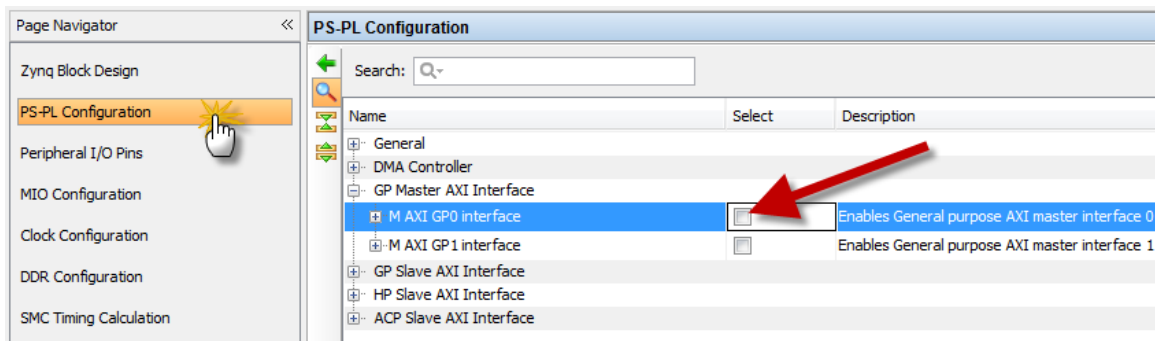
Input Frequency (MHz): 33.333333 CPU Clock Ratio: 6:2:1

Search:

Component	Clock Source	Requested Frequen...	Actual Frequency(M...	Range(MHz)
Processor/Memory Clocks				
CPU	ARM PLL	666.666666	666.666687	50.0 : 667.0
DDR	DDR PLL	533.333333	533.333374	200.000000 : 534.000000
IO Peripheral Clocks				
SMC	IO PLL	100	10.000000	10.000000 : 100.000000
QSPI	IO PLL	200	10.000000	10.000000 : 200.000000
ENET0	IO PLL	1000 Mbps	10.000000	
ENET1	IO PLL	1000 Mbps	10.000000	
SDIO	IO PLL	125	10.000000	10.000000 : 125.000000
SPI	IO PLL	166.666666	10.000000	0.000000 : 200.000000
CAN				
CAN CLK	IO PLL	100	10.000000	0.100000 : 100.000000
CAN0 MIOCLK	External	-1		-2 : -1
CAN1 MIOCLK	External	-1		-2 : -1
PL Fabric Clocks				
<input checked="" type="checkbox"/> FCLK_CLK0	IO PLL	50	50.000000	0.100000 : 250.000000
<input type="checkbox"/> FCLK_CLK1	IO PLL	50	50.000000	0.100000 : 250.000000
<input type="checkbox"/> FCLK_CLK2	IO PLL	50	50.000000	0.100000 : 250.000000
<input type="checkbox"/> FCLK_CLK3	IO PLL	50	50.000000	0.100000 : 250.000000

Figure 11 - Disable FCLK_CLK0

- 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 the *GP Master AXI Interface* section, **disable** the *M AXI GP0 Interface* by unchecking the box.



PS-PL Configuration

Search:

Name	Select	Description
General		
DMA Controller		
GP Master AXI Interface		
<input checked="" type="checkbox"/> M AXI GP0 interface	<input type="checkbox"/>	Enables General purpose AXI master interface 0
<input checked="" type="checkbox"/> M AXI GP1 interface	<input type="checkbox"/>	Enables General purpose AXI master interface 1
<input checked="" type="checkbox"/> GP Slave AXI Interface		
<input checked="" type="checkbox"/> HP Slave AXI Interface		
<input checked="" type="checkbox"/> ACP Slave AXI Interface		

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.

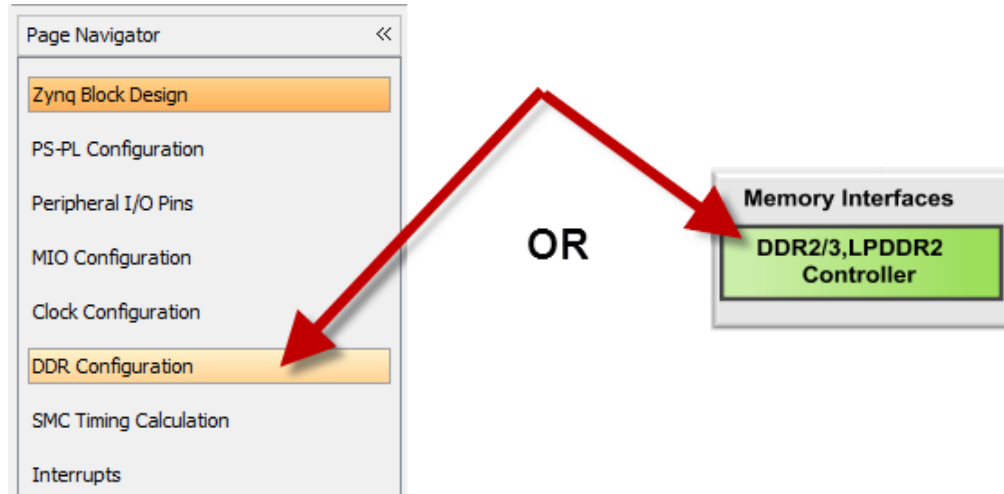


Figure 13 - DDR Memory Configuration

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

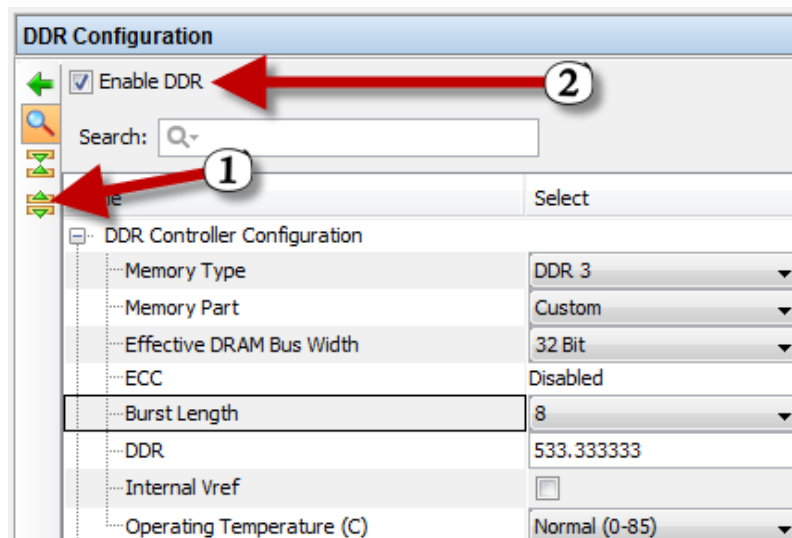


Figure 14 - View Memory Parameters and enable DDR

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* = **DDR3**
- *Memory Part*:
 - o MicroZed = **MT41K256M16RE-125**
 - o ZedBoard = **MT41J128M16HA-15E**

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.

Memory Part Configuration		
DRAM IC Bus Width	16 Bits	Width of individual DRAM components.
DRAM Device Capacity	4096 MBits	Storage capacity of individual DRAM components.
Speed Bin	DDR3_1066F	Speed bin of the individual DRAM components.
Bank Address Count (Bits)	3	Number of bank address pins.
Row Address Count (Bits)	15	Number of row address pins.
Col Address Count (Bits)	10	Number of column address bits.
CAS Latency (cycles)	7	Column Access Strobe (CAS) latency in memory clock cycles. It refers to the am...
CAS Write Latency (cycles)	6	CAS write latency setting in memory clock cycles.
RAS to CAS Delay (cycles)	7	tRCD. Row address to column address delay time. It is the time required betwe...
Precharge Time	7	tRP. Precharge Time is the number of clock cycles needed to terminate access t...
tRC (ns)	48.75	Row cycle time (ns)
tRASmin (ns)	35.0	Minimum number of memory clock cycles required between an Active and Precha...
tFAW (ns)	40.0	Determines the number of activates that can be performed within a certain wind...

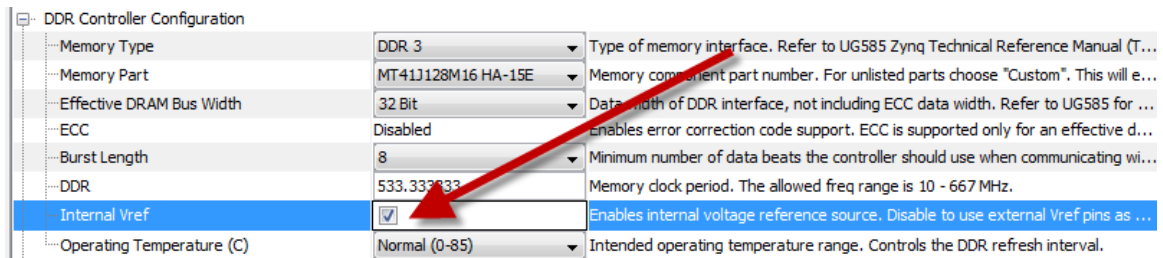
Figure 15 – MicroZed Memory Part Configuration

Memory Part Configuration		
DRAM IC Bus Width	16 Bits	Width of individual DRAM components.
DRAM Device Capacity	2048 MBits	Storage capacity of individual DRAM components.
Speed Bin	DDR3_1066F	Speed bin of the individual DRAM components.
Bank Address Count (Bits)	3	Number of bank address pins.
Row Address Count (Bits)	14	Number of row address pins.
Col Address Count (Bits)	10	Number of column address bits.
CAS Latency (cycles)	7	Column Access Strobe (CAS) latency in memory clock cycles. It refers to the am...
CAS Write Latency (cycles)	6	CAS write latency setting in memory clock cycles.
RAS to CAS Delay (cycles)	7	tRCD. Row address to column address delay time. It is the time required betwe...
Precharge Time	7	tRP. Precharge Time is the number of clock cycles needed to terminate access t...
tRC (ns)	49.5	Row cycle time (ns)
tRASmin (ns)	36.0	Minimum number of memory clock cycles required between an Active and Precha...
tFAW (ns)	45.0	Determines the number of activates that can be performed within a certain wind...

Figure 16 - ZedBoard Memory Part Configuration

9. In the DDR Controller Configuration section, the *Effective DRAM Bus Width* must be assigned to **32-bits** as we are using a 2x16 DDR3 configuration. Since we are using DDR3 on a 7Z010 (or 7Z020) 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.

10. Check the box for **Internal Vref**. The operating temperature can remain at the **Normal (0-85)** range.

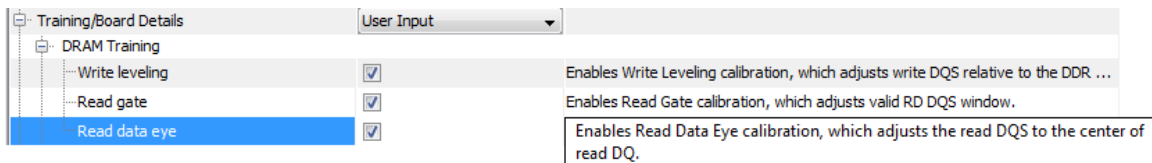


DDR Controller Configuration		
Memory Type	DDR 3	Type of memory interface. Refer to UG585 Zynq Technical Reference Manual (T...
Memory Part	MT41J128M16 HA-15E	Memory component part number. For unlisted parts choose "Custom". This will e...
Effective DRAM Bus Width	32 Bit	Data width of DDR interface, not including ECC data width. Refer to UG585 for ...
ECC	Disabled	Enables error correction code support. ECC is supported only for an effective d...
Burst Length	8	Minimum number of data beats the controller should use when communicating wi...
DDR	533.333333	Memory clock period. The allowed freq range is 10 - 667 MHz.
Internal Vref	<input checked="" type="checkbox"/>	Enables internal voltage reference source. Disable to use external Vref pins as ...
Operating Temperature (C)	Normal (0-85)	Intended operating temperature range. Controls the DDR refresh interval.

Figure 17 - DDR Controller Configuration (ZedBoard)

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

11. DRAM Training must be enabled for *Write leveling*, *Read gate*, and *Read data eye* options. **Check those 3 boxes** now. An explanation of what these are is in the [Zynq TRM](#), Section 10.6.8.



Training/Board Details		
		User Input
DRAM Training		
Write leveling	<input checked="" type="checkbox"/>	Enables Write Leveling calibration, which adjusts write DQS relative to the DDR ...
Read gate	<input checked="" type="checkbox"/>	Enables Read Gate calibration, which adjusts valid RD DQS window.
Read data eye	<input checked="" type="checkbox"/>	Enables Read Data Eye calibration, which adjusts the read DQS to the center of read DQ.

Figure 18 - 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.

12. Edit the **DQS to Clock Delay** and **Board Delay** settings as show here.

DQS to Clock Delay (ns)		
DQS0	-0.073	DQS to Clock delay [0] (ns). The DQS path delay subtracted from the clock path..
DQS1	-0.072	DQS to Clock delay [1] (ns). The DQS path delay subtracted from the clock path..
DQS2	0.024	DQS to Clock delay [2] (ns). The DQS path delay subtracted from the clock path..
DQS3	0.023	DQS to Clock delay [3] (ns). The DQS path delay subtracted from the clock path..
Board Delay (ns)		
DQ[7:0]	0.294	Board delay [0] (ns). The midpoint of data (DDR_DQ, DDR_DM) trace delays av...
DQ[15:8]	0.298	Board delay [1] (ns). The midpoint of data (DDR_DQ, DDR_DM) trace delays av...
DQ[23:16]	0.338	Board delay [2] (ns). The midpoint of data (DDR_DQ, DDR_DM) trace delays av...
DQ[31:24]	0.334	Board delay [3] (ns). The midpoint of data (DDR_DQ, DDR_DM) trace delays av...
Additive Latency (ns)	0	Additive Latency (ns). Increases the efficiency of the command and data bus fo...

Figure 19 - MicroZed Board Training details

DQS to Clock Delay (ns)		
DQS0	0.029	DQS to Clock delay [0] (ns). The DQS path delay subtracted from the clock path...
DQS1	0.030	DQS to Clock delay [1] (ns). The DQS path delay subtracted from the clock path...
DQS2	-0.002	DQS to Clock delay [2] (ns). The DQS path delay subtracted from the clock path...
DQS3	-0.011	DQS to Clock delay [3] (ns). The DQS path delay subtracted from the clock path...
Board Delay (ns)		
DQ[7:0]	.345	Board delay [0] (ns). The midpoint of data (DDR_DQ, DDR_DM) trace delays av...
DQ[15:8]	.345	Board delay [1] (ns). The midpoint of data (DDR_DQ, DDR_DM) trace delays av...
DQ[23:16]	.270	Board delay [2] (ns). The midpoint of data (DDR_DQ, DDR_DM) trace delays av...
DQ[31:24]	.272	Board delay [3] (ns). The midpoint of data (DDR_DQ, DDR_DM) trace delays av...
Additive Latency (ns)	0	Additive Latency (ns). Increases the efficiency of the command and data bus fo...

Figure 20 - ZedBoard Board Training details

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

14. **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.

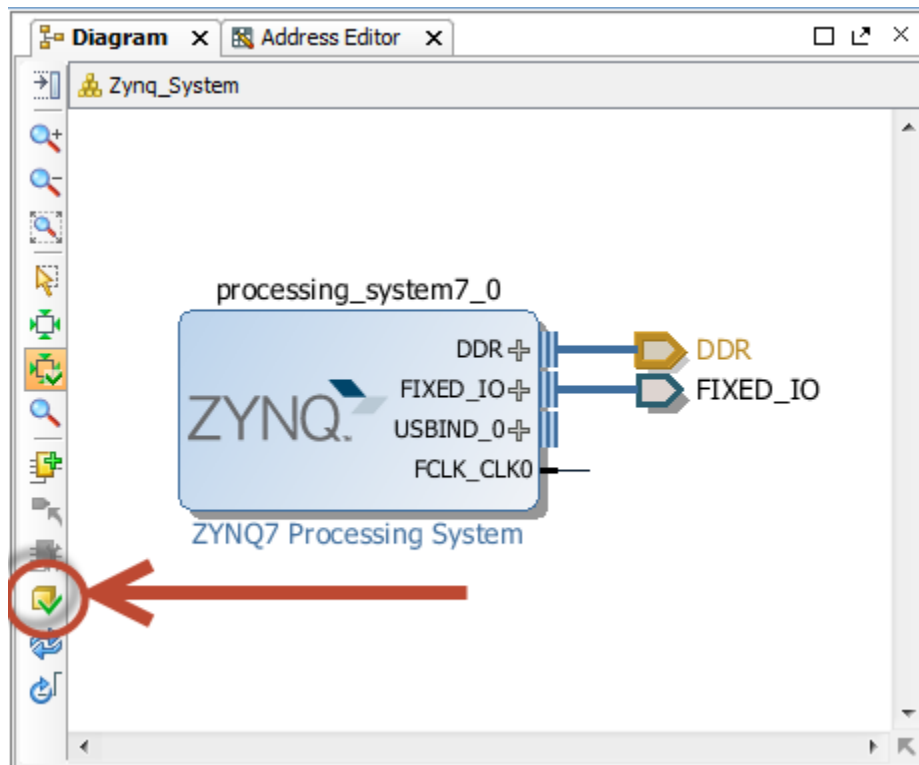


Figure 21 - Validate Block Design

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

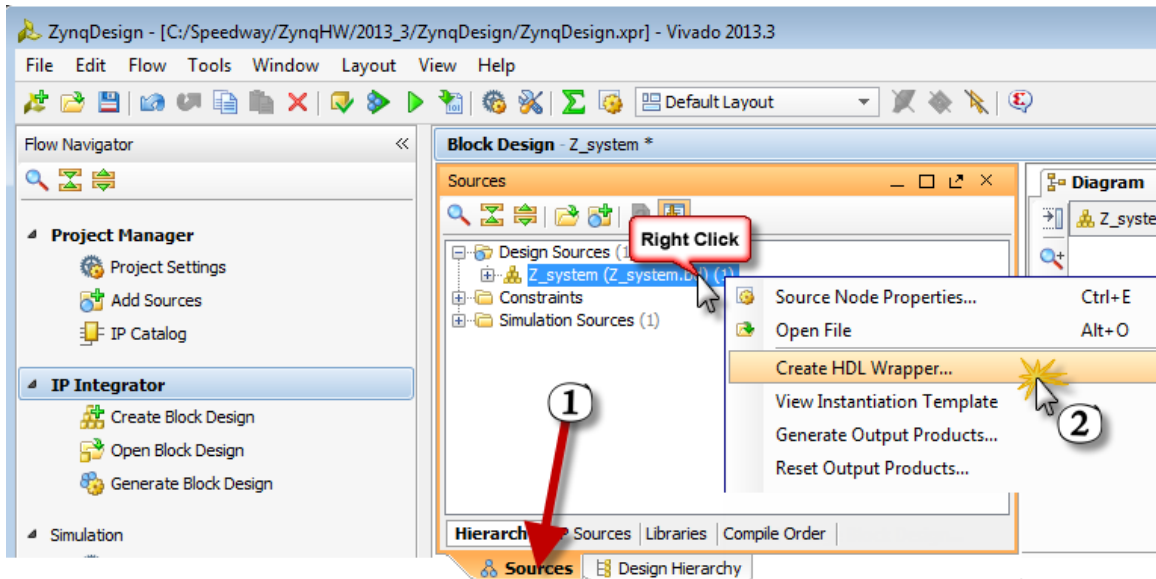


Figure 22 - Create HDL Wrapper

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**.

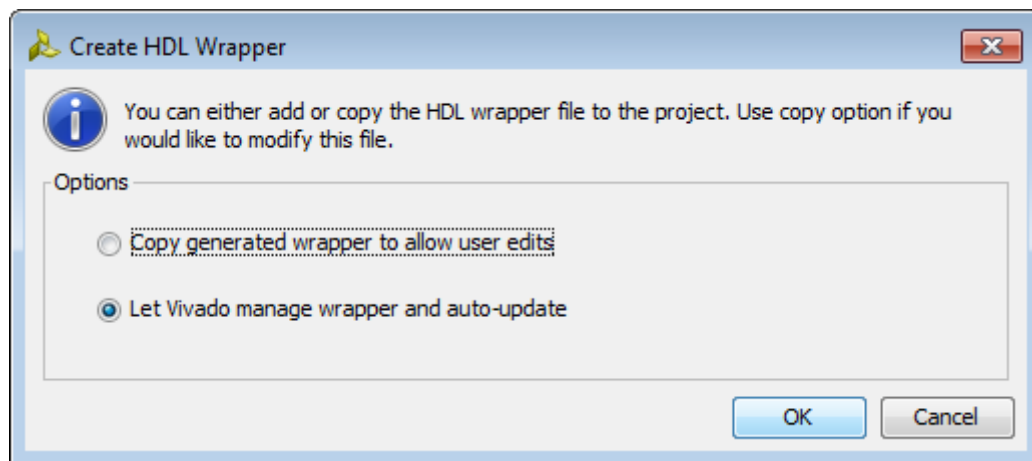


Figure 23 - Create HDL Wrapper options

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

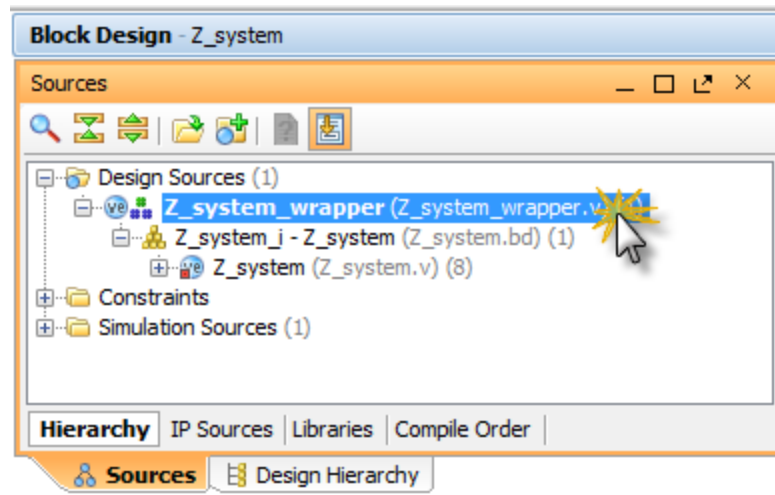


Figure 24 - Design Sources

- 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.

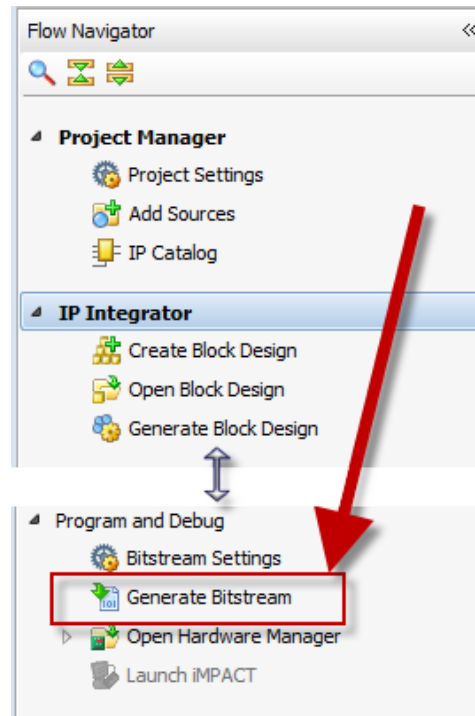


Figure 25 – Generate Bitstream

6. Vivado will warn that no synthesis or implementation results exist and ask to launch these steps. Click **Yes**.

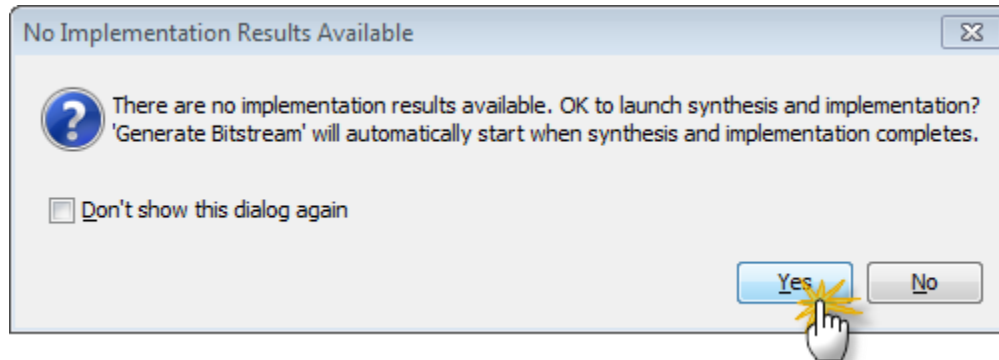


Figure 26 - Launch Synthesis and Implementation

7. 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.

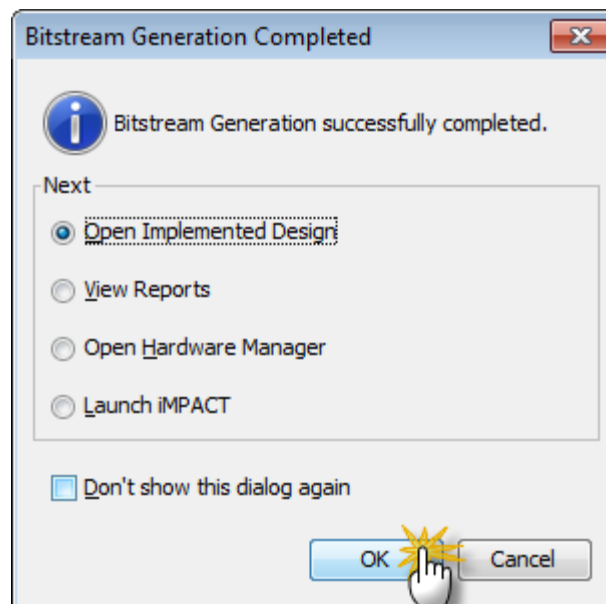


Figure 27 - Open Implemented Design

8. From the pull-down menus at the top of Vivado, select **File → Export → Export Hardware for SDK...**

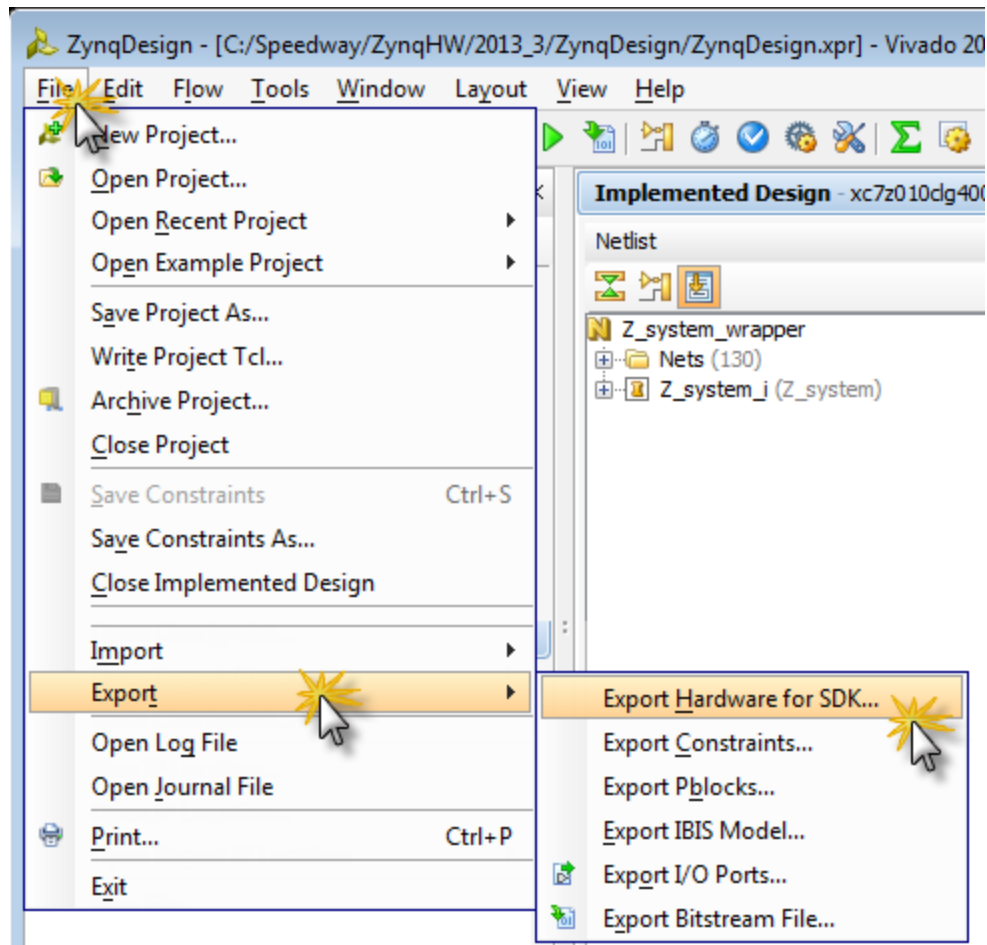


Figure 28 - Export Hardware for SDK...

9. In the next window, notice the Include bitstream box is checked. This was done because we opened an implemented design in our last step. Set the *Export to* and *Workspace* fields to: **C:\Speedway\ZynqHW\2013_3\SDK_Workspace**

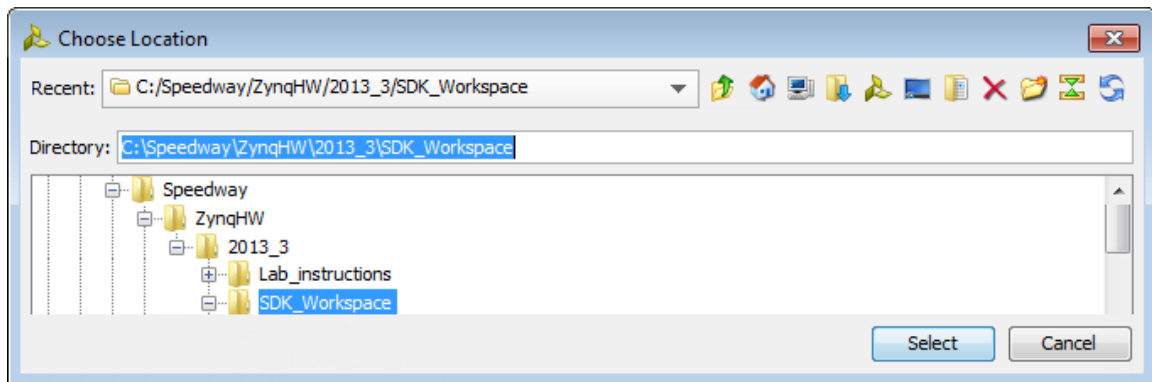


Figure 29 - Export Directory

10. Check the box to **Launch SDK**. Click **OK**.

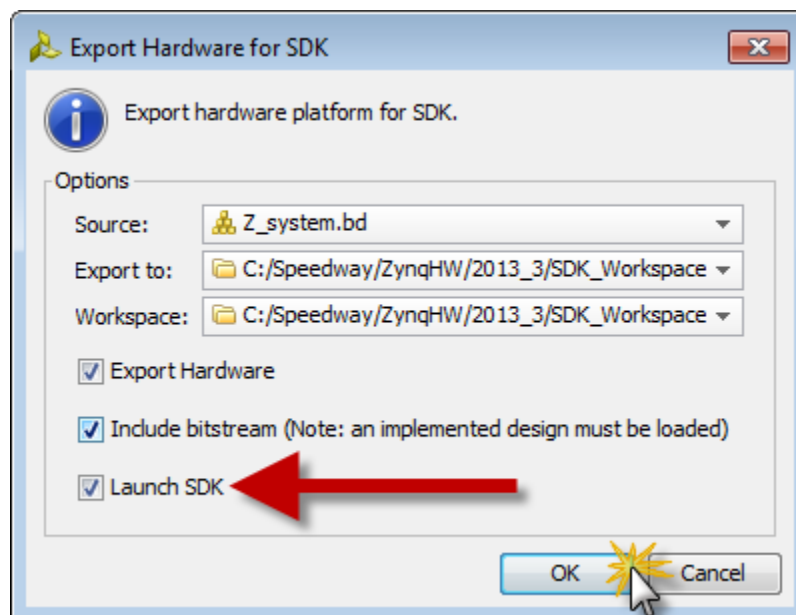


Figure 30 - Export Options

- Using a file explorer, open the SDK_Workspace directory to see what files were exported. Expand the *hw_platform_0* folder:

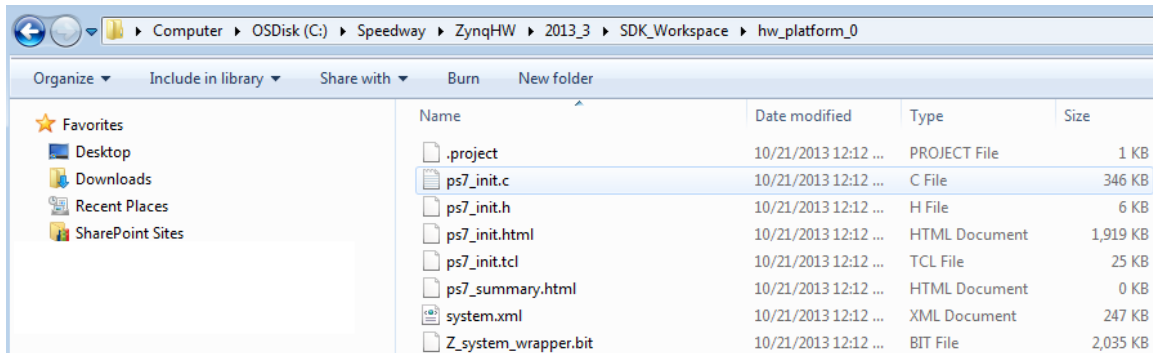


Figure 31 - Exported Files for SDK

The Vivado design tool exported the Hardware Platform Specification for your design (system.xml in this example) to SDK. In addition to system.xml, there are six more files exported to SDK. They are ps7_init.c, ps7_init.h, ps7_init.tcl, ps7_init.html, ps7_summary.html and Z_system_wrapper.bit.

The system.xml file opens by default when SDK launches. The address map of your system read from this file is 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 (MicroZed)

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**.

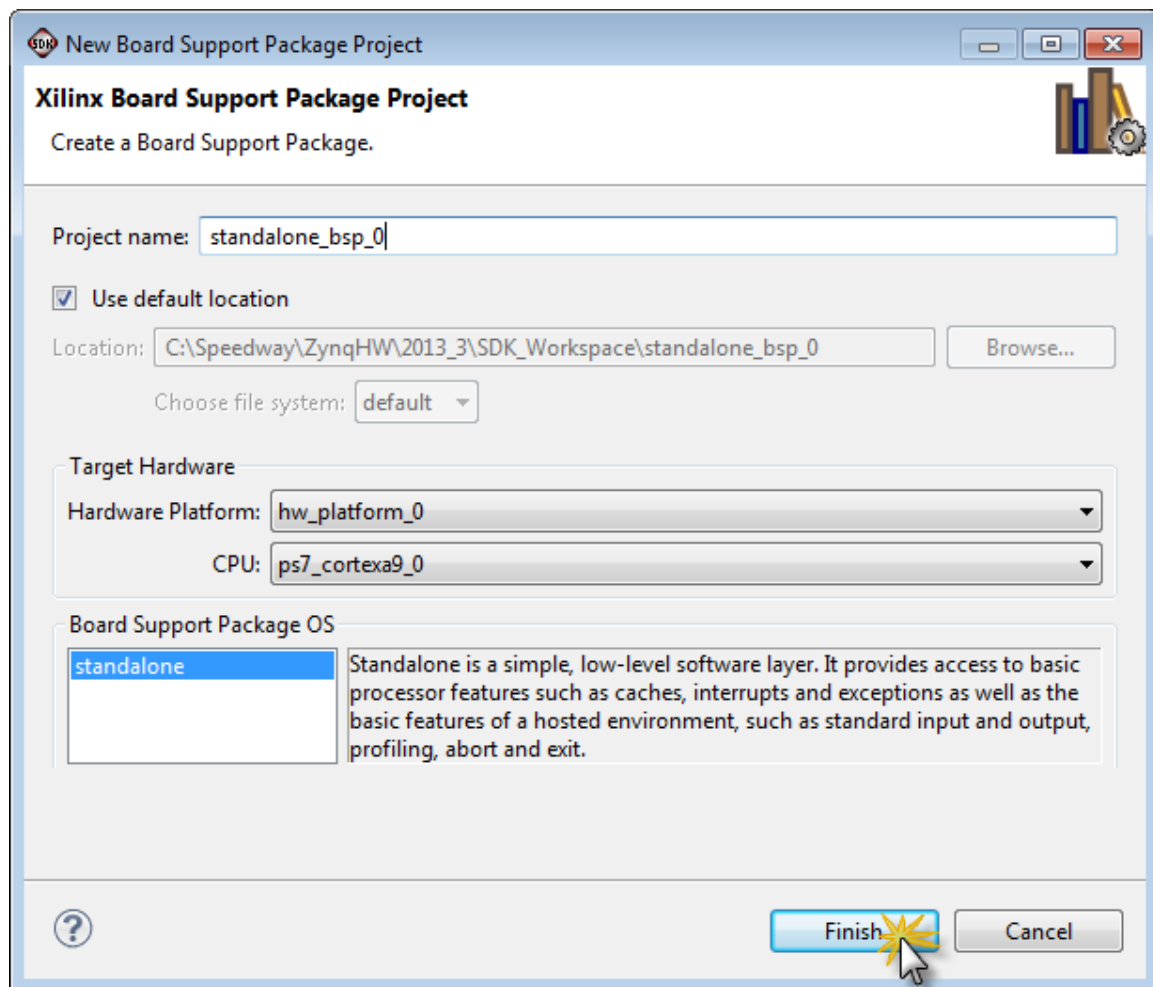


Figure 33 – Standalone BSP

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

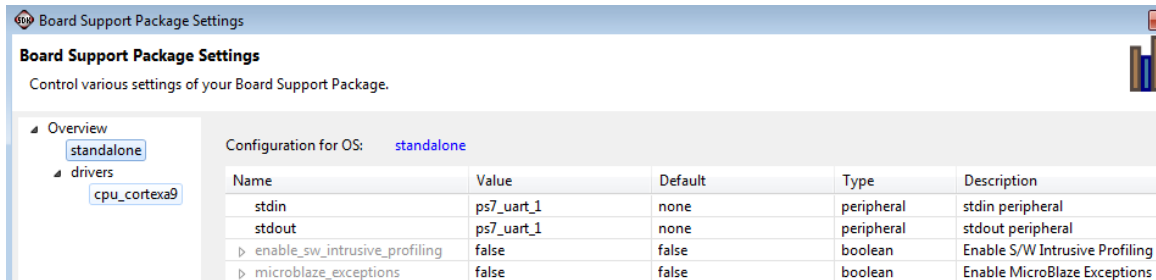


Figure 34 – stdin and stdout settings

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.

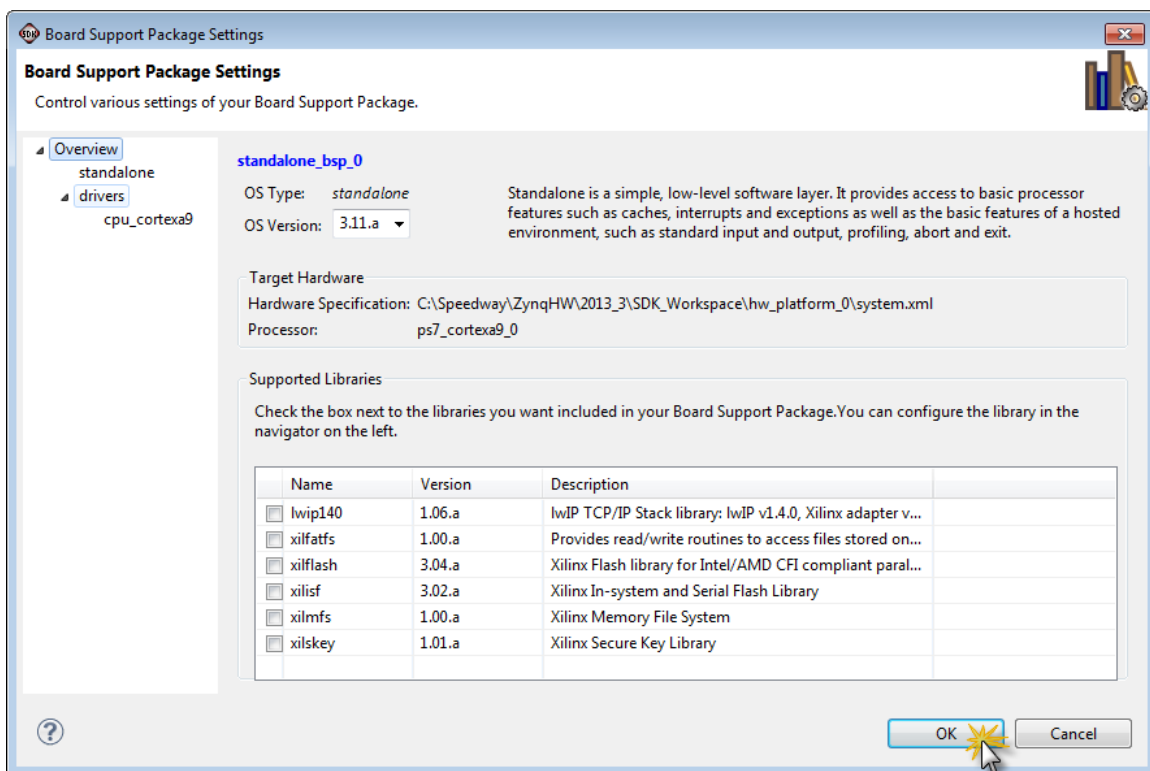


Figure 35 – BSP Settings

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.

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

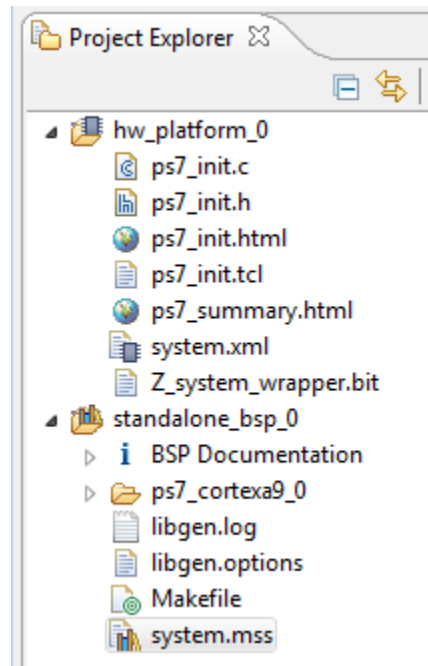


Figure 36 – BSP Added to the Project

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

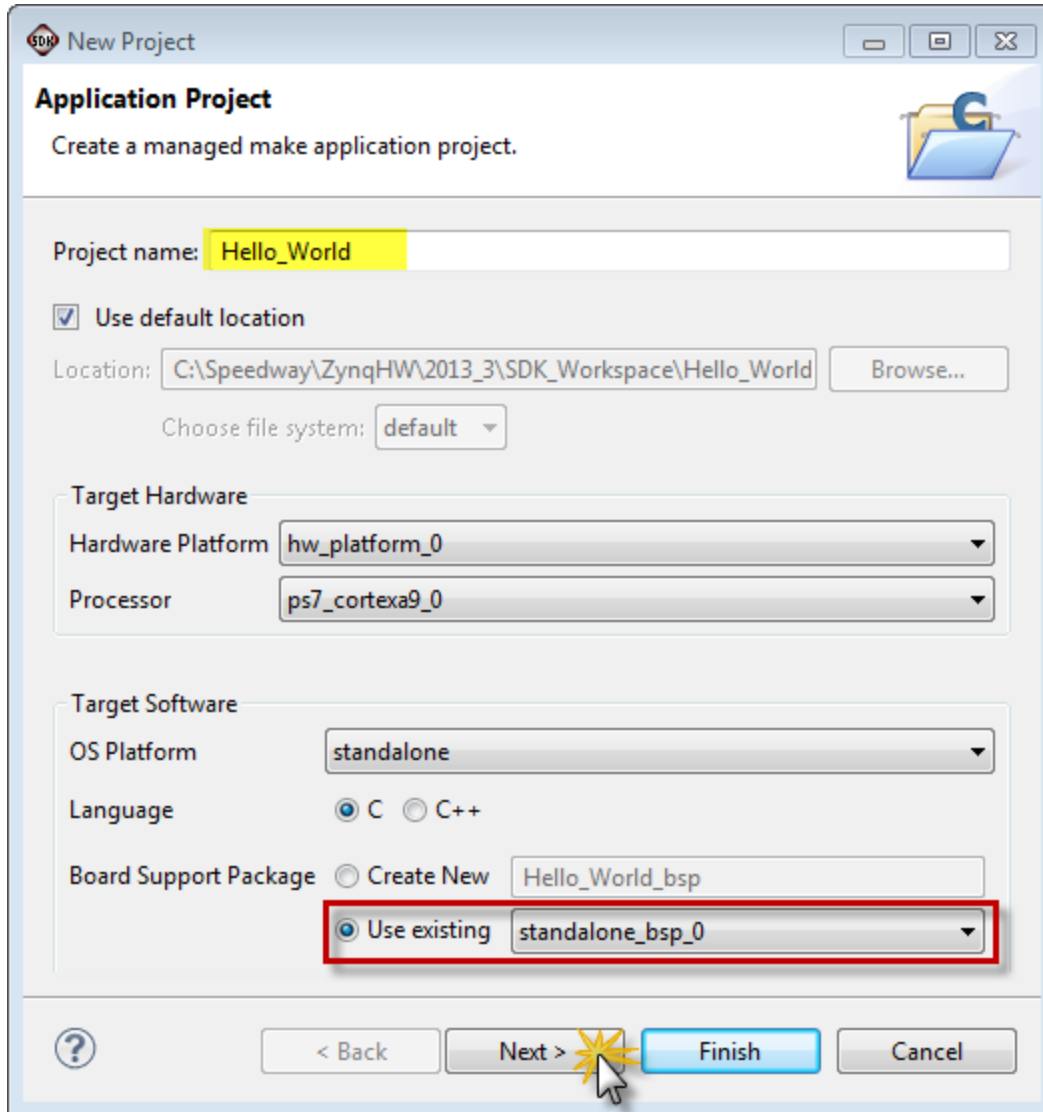


Figure 37 - Application Project Settings

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

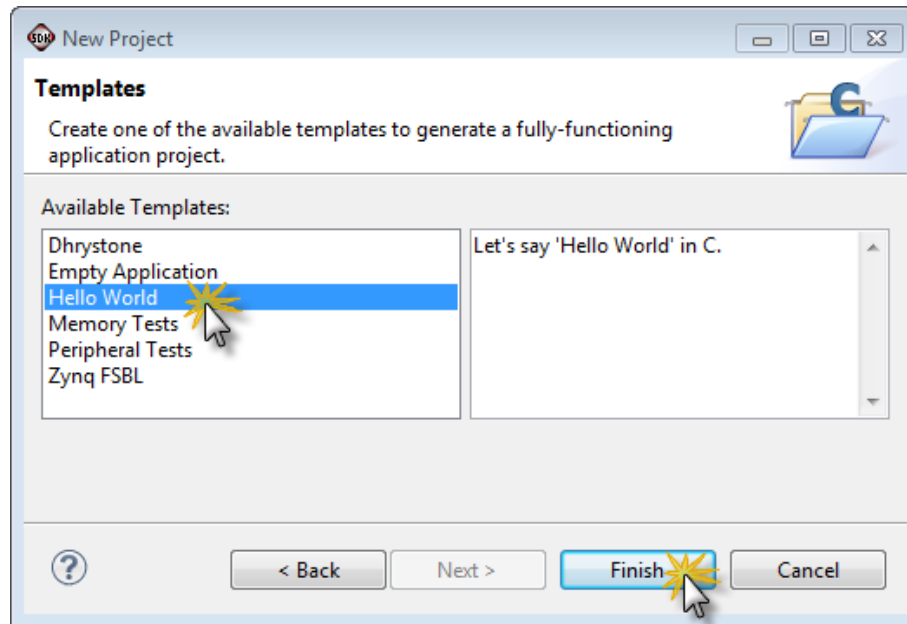


Figure 38 – New Application: Hello World

Hello_World will automatically build:

```
00:29:15 **** Build of configuration Debug for project Hello_World ****
make all
'Building file: ../src/helloworld.c'
'Invoking: ARM gcc compiler'
arm-xilinx-eabi-gcc -Wall -O0 -g3 -c -fmessage-length=0 -I../standalone_bsp_0/ps7_cortexa9_0/include -MMD -MP
-MF"src/helloworld.d" -MT"src/helloworld.d" -o "src/helloworld.o" "../src/helloworld.c"
'Finished building: ../src/helloworld.c'
'
'
'Building file: ../src/platform.c'
'Invoking: ARM gcc compiler'
arm-xilinx-eabi-gcc -Wall -O0 -g3 -c -fmessage-length=0 -I../standalone_bsp_0/ps7_cortexa9_0/include -MMD -MP
-MF"src/platform.d" -MT"src/platform.d" -o "src/platform.o" "../src/platform.c"
'Finished building: ../src/platform.c'
'
'
'Building target: Hello_World.elf'
'Invoking: ARM gcc linker'
arm-xilinx-eabi-gcc -Wl,-T ../src/ldscript.ld -L../standalone_bsp_0/ps7_cortexa9_0/lib -o
"Hello_World.elf" ./src/helloworld.o ./src/platform.o -Wl,--start-group,-lxil,-lgcc,-lc,--end-group
'Finished building target: Hello_World.elf'
'
'
'Invoking: ARM Print Size'
arm-xilinx-eabi-size Hello_World.elf |tee "Hello_World.elf.size"
  text  data   bss   dec   hex filename
22872  1096  29780  53748  d1f4 Hello_World.elf
'Finished building: Hello_World.elf.size'
'
'

00:29:17 Build Finished (took 2s.936ms)
```

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.

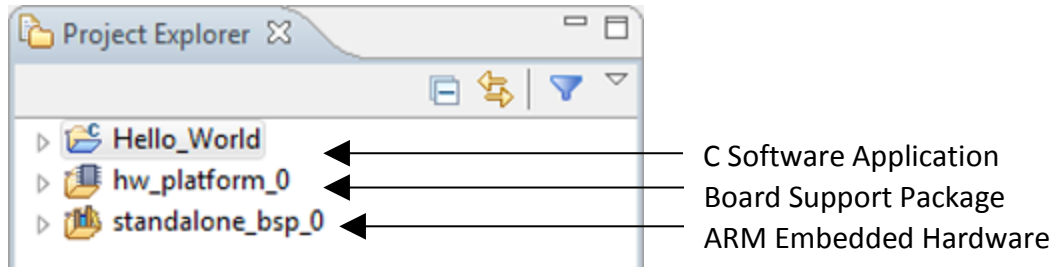


Figure 40 – Project Explorer View with Hello World C Application Added

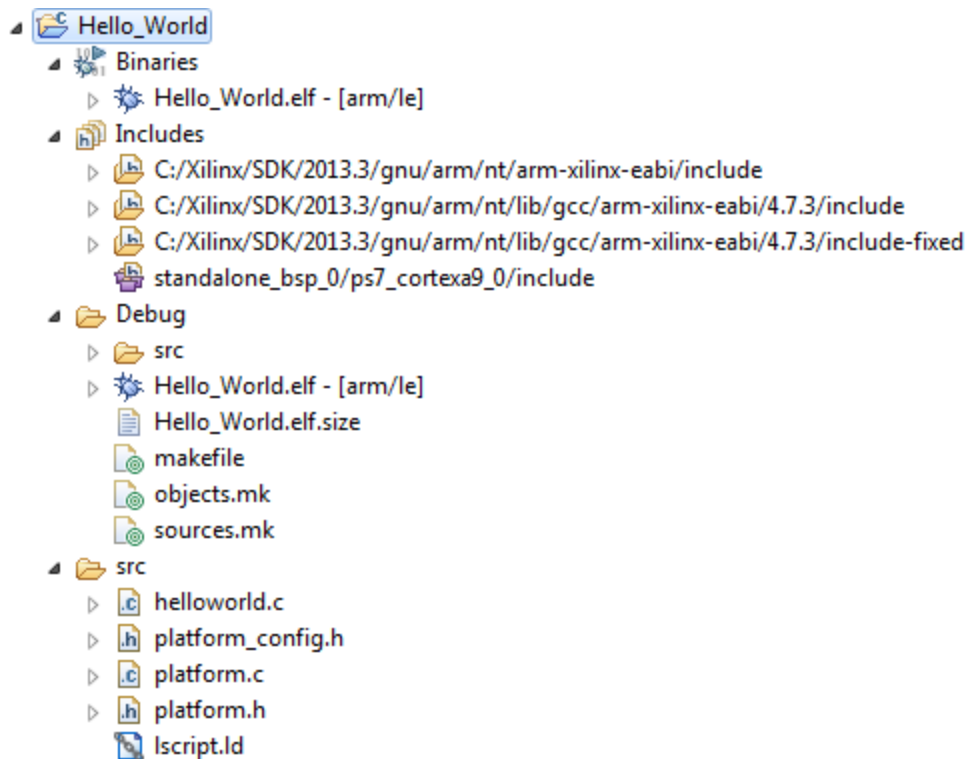


Figure 41 – Hello_World Application Expanded

9. MicroZed users, **Connect a USB cable** between a PC and the microUSB port on MicroZed. **Connect a Xilinx JTAG Programming cable** to J3.

ZedBoard users, **Connect the power cable** to the ZedBoard. **Connect two micro-USB cables** between the Windows Host machine and the ZedBoard connectors J17 (JTAG) and J14 (UART).

10. Set jumpers to **Cascaded JTAG Mode**:



ZedBoard



MicroZed

Figure 42 – Mode Pin Jumpers

11. **ZedBoard Only: Power** on the board. If this is the first time you've connected either of these boards to this computer, you may see Windows install device drivers for the USB-UART and Digilent JTAG.
12. Use Device Manager to determine the COM port for the USB-UART. In Windows 7, click **Start → Control Panel**, and then click **Device Manager**. Click **Yes** to confirm.
13. Expand **Ports**. Note the COM port number for the Cypress Serial device. This example shows COM6.

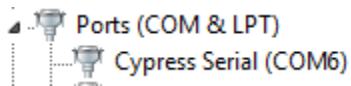


Figure 43 – Find the COM port number for the Serial device

14. Right-click on the **Hello_World** application and select **Run As → Run Configurations...**

15. Select **Xilinx C/C++ Application (GDB)** and then click the 'New' icon .

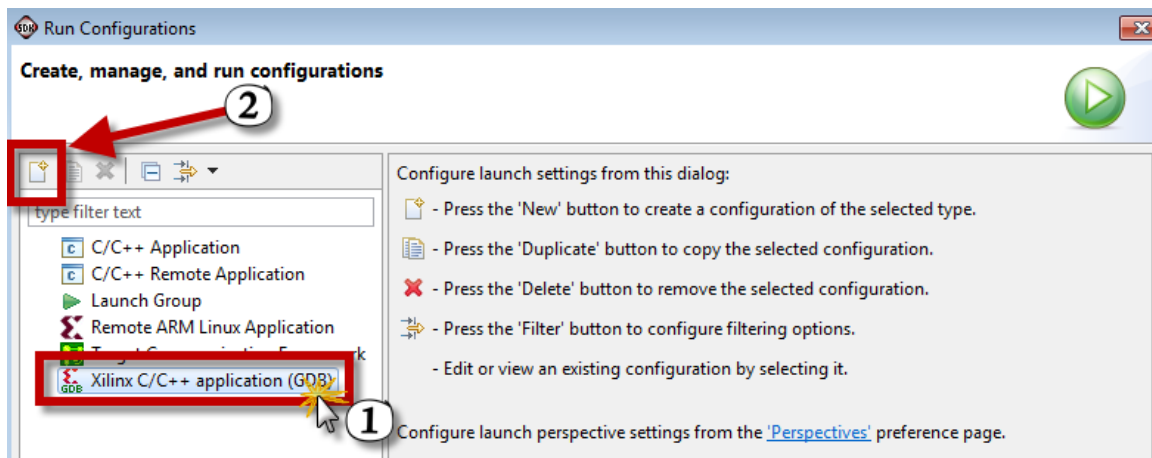


Figure 44 – Create a New Xilinx C/C++ ELF Run Configuration

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`.

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

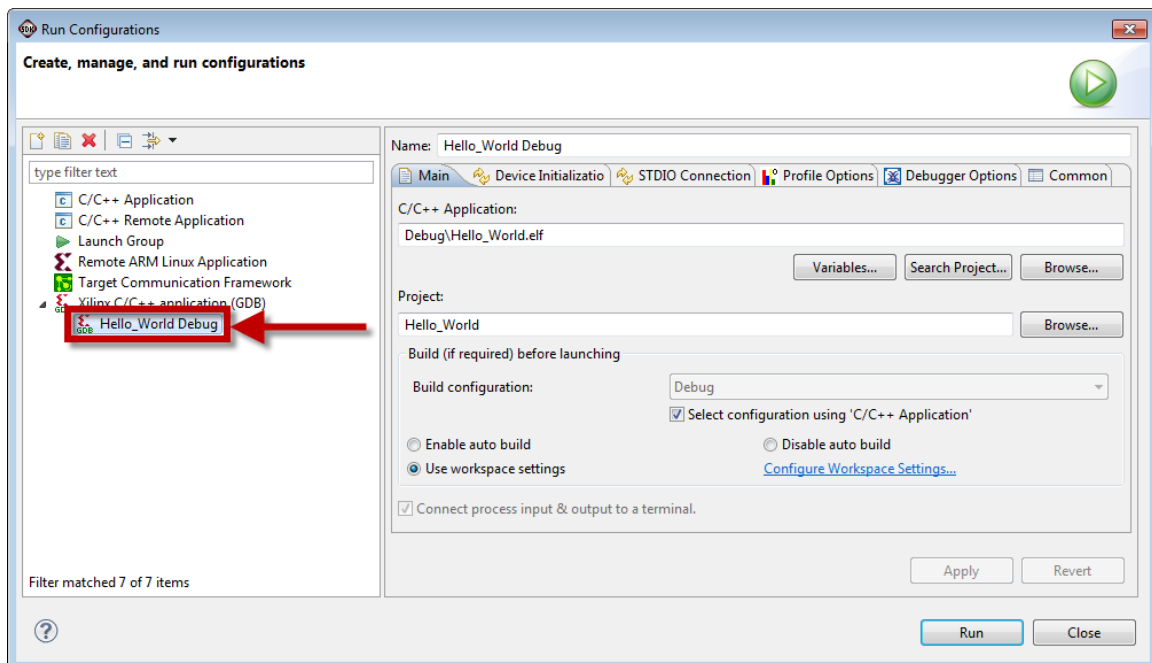


Figure 45 – New Run Configuration

17. Switch to the **STDIO Connection** tab.

18. Check the box for **Connect STDIO to Console**.

19. Select the **PORT** for the USB-UART.

20. Set the BAUD Rate to **115200**.

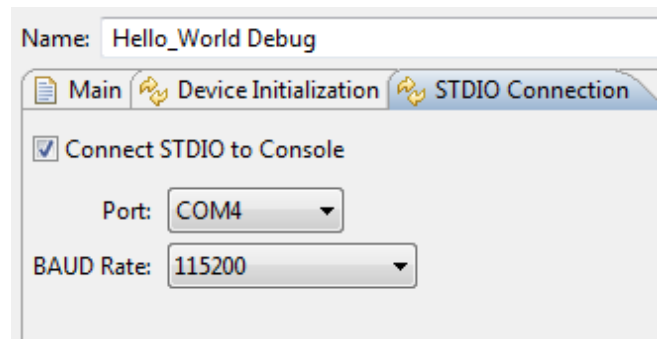


Figure 46 – STDIO Connection

21. Click **Apply** and then **Run**.

The tools will now initialize the processor, download the Hello_World.elf to DDR, and then run hello_world. This takes approximately 15 seconds to complete, depending on the USB traffic in your system. You can follow the progress in the lower right corner of SDK.

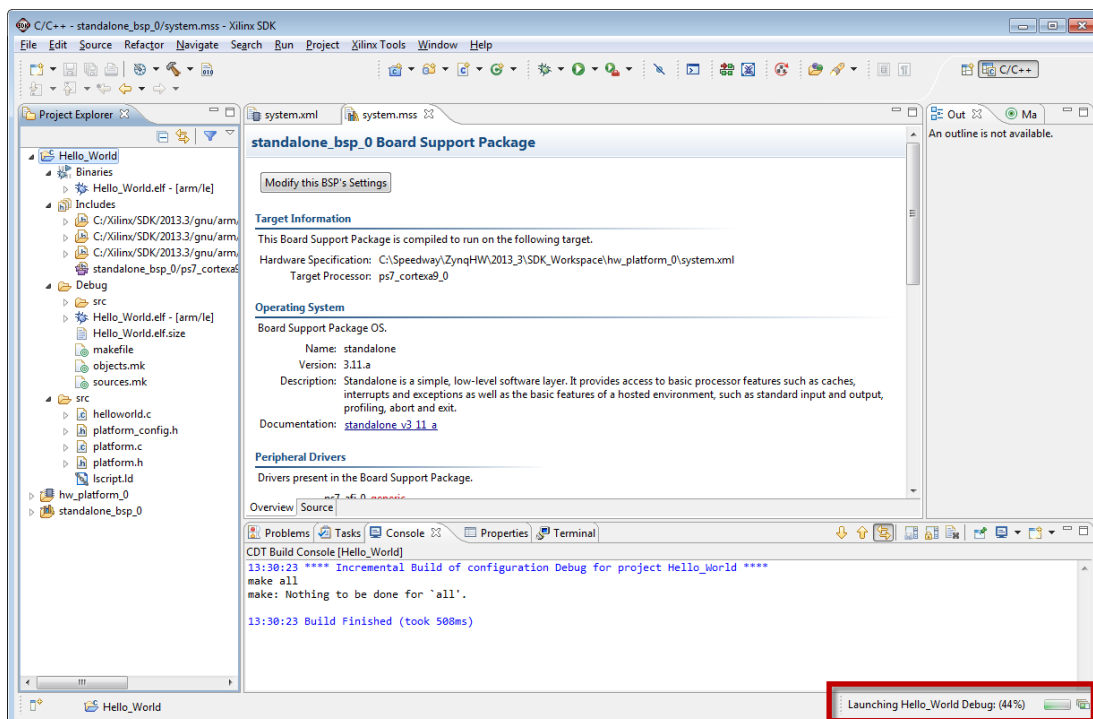


Figure 47 – Launching hello_world Progress

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

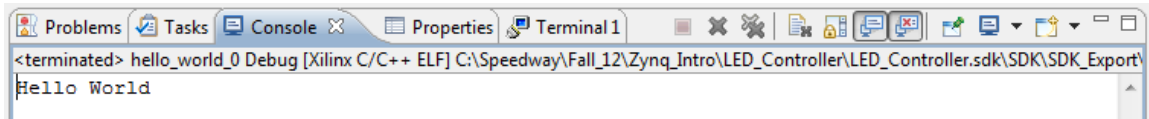


Figure 48 – Hello World Complete

22. 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?*

Exploring Further

If you have more time and would like to investigate more...

- Read the Appendix on calculating DDR 3 PCB Delays

This concludes Lab 2.

Revision History

Date	Version	Revision
6 Nov 13	02	Initial Draft
19 Nov 13	03	Pilot updates

Resources

www.microzed.org

www.zedboard.org

www.xilinx.com/zyng

www.xilinx.com/sdk

www.xilinx.com/vivado

Appendix – Calculating DDR 3 PCB Lengths (ZedBoard)

Zynq allows for up to 4 memory devices to be configured for DDR3 4x8 fly-by topology. ZedBoard is configured for DDR3 2x16 fly-by topology. For a description of fly-by topology, see [Micron TN-41-13](#). Figures 1 and Figure 7 show the basic daisy-chain topology used for Command/Address/Control/Clocks.

With the two DDR3 chips, the DQS's and DQ's are all point-to-point. Those values can easily be obtained from a simple trace length report. However, because of the fly-by routing, the CLK routing is daisy-chained. ZedBoard only has one DDR3 clock pair, which goes to IC26 first and then IC25. The CLK trace length in the report is going to be the total length. Getting the CLK2 and CLK3 lengths will be more challenging, and we'll have to open the layout to get this measurement.

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

1. A trace length report for ZedBoard is provided in the **Support_documents** folder. Open [ZedBoard RevB PCB Trace Length Report.txt](#) now.

All the DDR3 signals are prefixed in this report with "DDR3-" 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; CLK2 and CLK3 are also the same.

2. Use the trace length report to calculate and then enter the trace lengths for CLK0, DQS0, and DQ[7:0].

Pin Group	ZedBoard Traces Used in Calculation	Length (mm)
CLK0	DDR3-CLK0_N, DDR3-CLK0_P	
DQS0	DDR3-DQS0_N, DDR3-DQS0_P	
DQ[7:0]	DDR3-D[7:0]	

Since CLK2 and CLK3 must be calculated differently, we'll look at them next. CLK2 is associated with Byte Group 2. CLK3 is associated with Byte Group 3. Byte Groups 2 and 3 are driven from IC26. Since traces DDR3-CLK0_P/N route daisy-chain first to IC26 and then IC25, the length to the trace segments between Zynq and IC26 must be measured manually. This must be done in Altium or Altium Viewer, which is the layout source for

ZedBoard. To simplify this exercise, these trace segments were previously measured and screen captured.

3. Average these two trace segments and enter for CLK2 and CLK3.

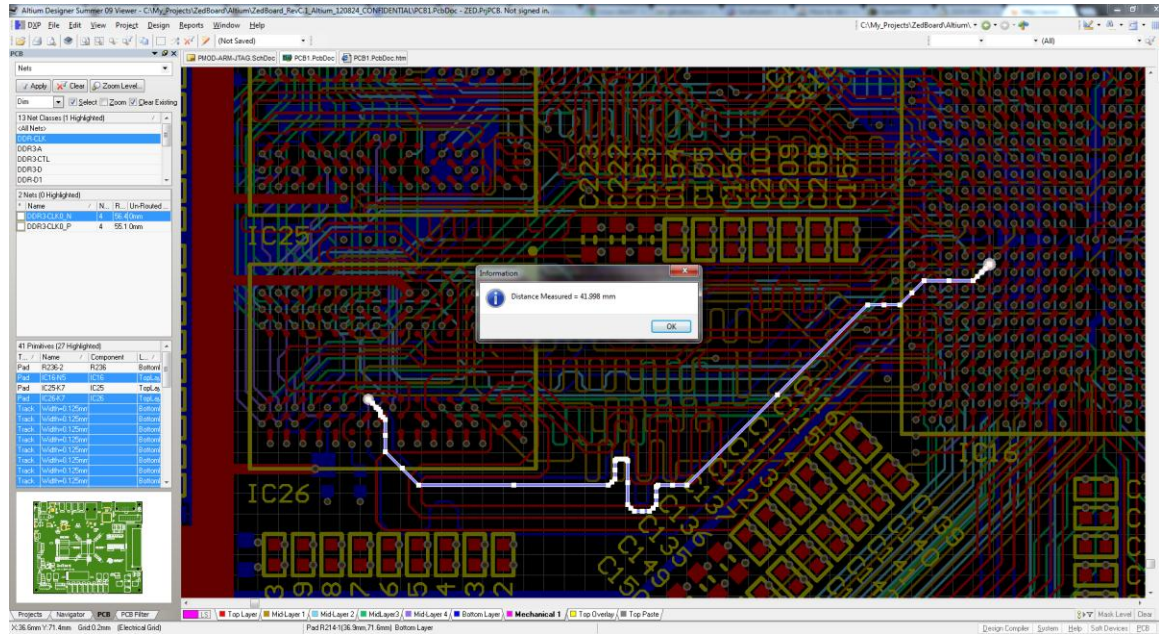


Figure 49 – Trace Segment DDR3-CLK0_N = 41.998 mm

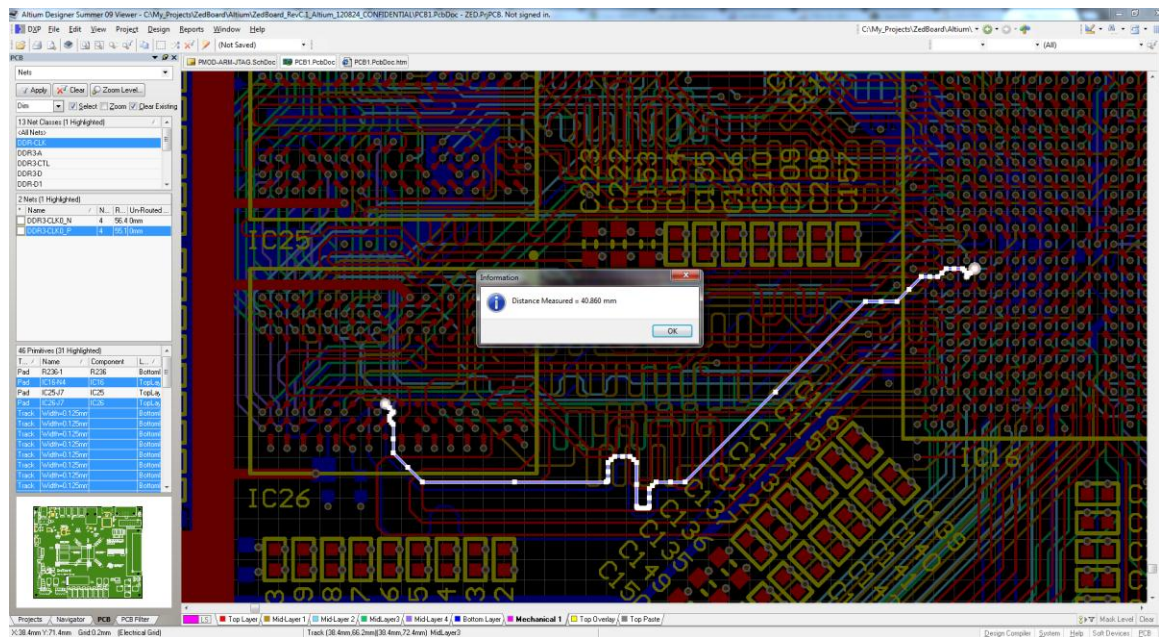


Figure 50 – Trace Segment DDR3-CLK0_P = 40.860 mm

4. Calculate the remaining lengths in similar fashion. Enter these now.

Pin Group	Length (mm)
CLK1	
DQS1	
DQS2	
DQS3	
DQ[15:8]	
DQ[23:16]	
DQ[31:24]	

The completed worksheet should appear as shown below.

Pin Group	Length (mm)	Package Length (mils)	Propagation Delay (ps/inch)	DQS to CLK Delay (ns)	Board Delay (ns)
CLK0	55.77	470.0	160		
CLK1	55.77	470.0	160		
CLK2	41.43	470.0	160		
CLK3	41.43	470.0	160		
DQS0	51.00	504	160	0.025	
DQS1	50.77	495	160	0.027	
DQS2	41.59	520	160	-0.009	
DQS3	41.9	835	160	-0.061	
DQ[7:0]	50.63	465	160		0.410
DQ[15:8]	50.71	480	160		0.411
DQ[23:16]	40.89	550	160		0.341
DQ[31:24]	40.58	780	160		0.358

Figure 51 – ZedBoard PCB Lengths Calculated and Entered

Questions:

Answer the following questions:

- *In a multi-component memory design, how do you calculate the CLK trace length for the first component in the daisy chain?*

- *How do you calculate the trace length for a DQ byte group?*

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?*

The peripherals are listed in order of priority. Peripherals on the top of the list generally need to be connected first and less MIO 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 DDR3 interface can run at? Extra Credit: What is the slowest?*

533MHz.

Slowest? It depends on the DDR3 memory device. Even though Vivado lists the slowest speed as 10MHz, the DDR3 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)

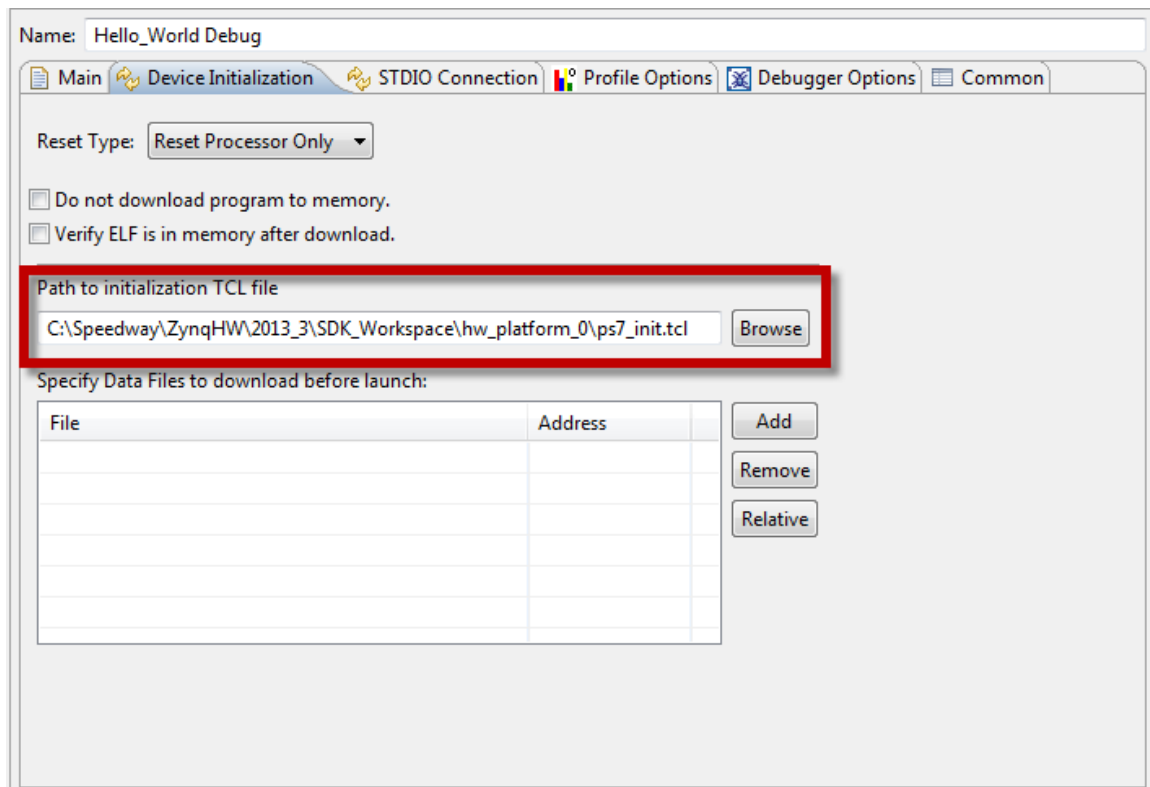
Experiment 4

- 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 was a Device Initialization tab. Here 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 to Average	Length (mm)
CLK0	DDR3-CLK0_N, DDR3-CLK0_P	Average(56.4267, 55.1225) = 55.7746 mm
DQS0	DDR3-DQS0_N, DDR3-DQS0_P	Average (51.3839,50.6222) = 51.00305 mm
DQ[7:0]	DDR3-D[7:0]	Min = 50.1401 Max = 51.1172 Avg = 50.6287 mm

Pin Group	Length (mm)
CLK1	55.77
DQS1	50.77
DQS2	41.59
DQS3	41.90
DQ[15:8]	50.71
DQ[23:16]	40.89
DQ[31:24]	40.58

- In a multi-component memory design, how do you calculate the CLK trace length for the first component in the daisy chain?*

You must open the layout tool to find the trace lengths for clk_p and clk_n from the Zynq to that component. Then average the _p and _n lengths.

- How do you calculate the trace length for a DQ byte group?*

Find the minimum length and the maximum length out of the 8 traces then average them.