Introduction to Zynq Hardware
Lab 9
The Power of Scripting using Tcl

June 2017
Version 1

Lab 9 Overview

In lab 9 we explore the benefits of scripting as we prepare to finalize our hardware project to be used in the Developing Zynq Software Speedway. We then explore and discuss the various changes that were made and interfaces added.

Lab 9 Objectives

When you have completed Lab 9, you will know how to do the following:

- Source a Tcl script
- Understand the power of scripting
- Understanding of the hardware platform that will be used in the Developing Zynq Software Speedway

Experiment 1: Finish Hardware Build using Tcl

This experiment shows how to run a script using the Tcl console.

Experiment 1 General Instruction:

Source lab9.tcl to finish hardware platform build

Experiment 1 Step-by-Step Instructions:

1. <Optional> If you did not complete Lab 8 or wish to start with a clean copy, delete the `ZynqDesign` and `ip_repo` folders in the `ZynqHW/2017_1` folder. Then unzip **`Solutions\ ZynqHW_Lab8_Solution.zip`** to the `2017_1` folder. If you have 7-zip installed, you can do this by right-clicking and dragging **`ZynqHW_Lab7_Solution.zip`** to the `2017_1` folder. Select **7-Zip → Extract Here**.

2. Open Vivado and make sure that the block design is open

3. Open the Tcl console below

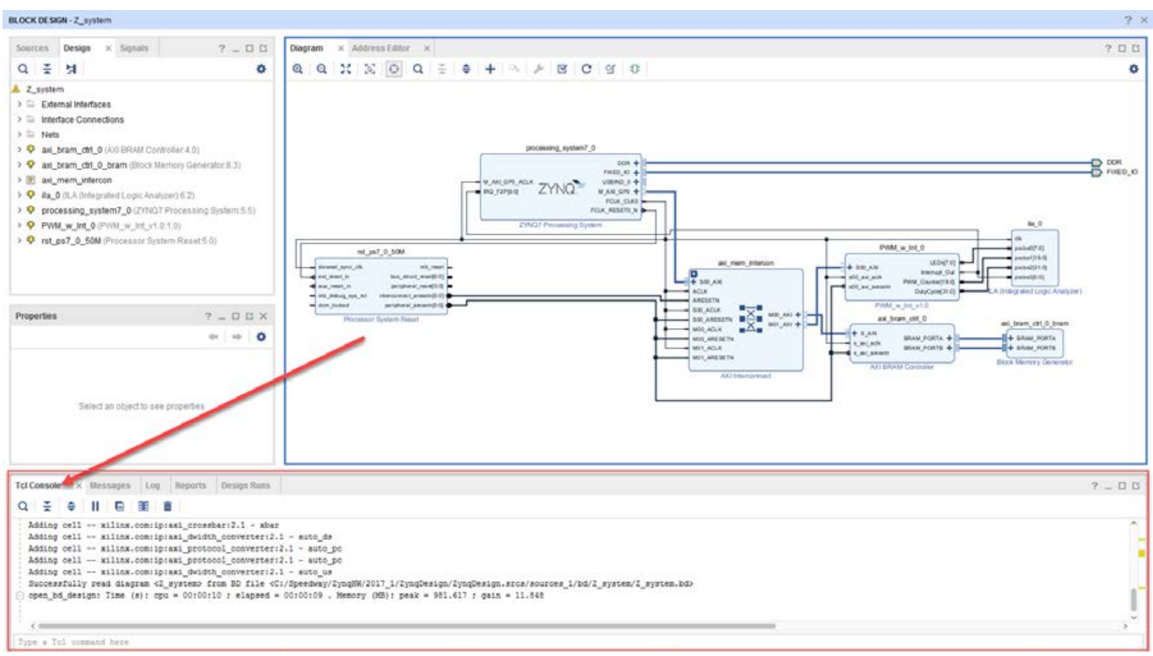

Figure 1 -- Tcl Console

4. Type the following Tcl commands to get to the correct directory in which the Tcl script we will run to finish the Hardware build is located. Then run the pwd command to verify you are in the C:/Speedway/ZynqHW/2017_1/Support_documents/Lab9/ directory.

```
cd C:/Speedway/ZynqHW/2017_1/Support_documents/Lab9/
```

```
pwd
```

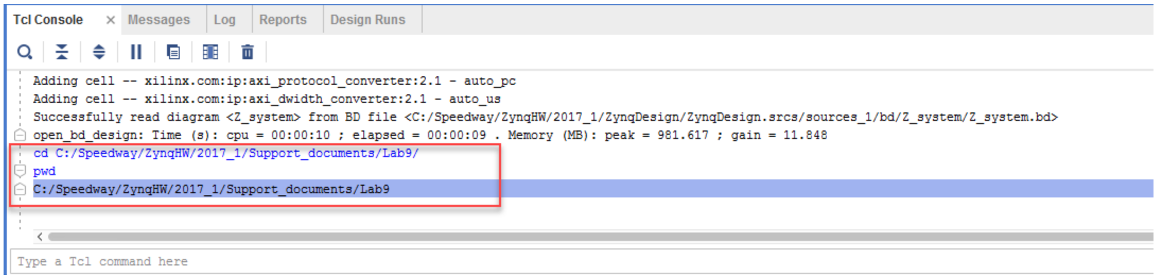

Figure 2 – Lab 9 Directory

5. Now source the lab9.tcl script in that LAB9 directory we just went into by running the following Tcl command.

You will notice the block design adding IP and connecting the various interfaces. Scripting is an excellent way to share projects will colleagues. For instance if not for this script, it would most likely take you an hour and a half to make all the various connections while the script makes all the connections in about 1 minute.

```
source ./lab9.tcl
```

*****NOTE*** If you receive an error running the Tcl command above please delete your entire ZynqDesign project and ip repo. Then start from Experiment 1 step 1 in which you unzip a pre-built lab 8 project and ip repo to the correct location. The reason you received an error was most likely due to an incorrect naming convention that was done in a previous lab.**

Experiment 2: Finish Hardware Build using Tcl

You will notice looking at your block design that various IP is being added to your design and connected to your existing hardware block design. To really understand what is going on lets take a look at the lab9.tcl script we just ran

Experiment 2 General Instruction:

Examine lab9.tcl script

Experiment 2 Step-by-Step Instructions:

1. In the **C:\Speedway\ZynqHW\2017_1\Support_documents\Lab9** directory open the lab9.tcl script using a text editor such as WordPad or Notepad++.

2. The first section of interest is where we declare were we are generating our Avnet IP located around line 55. The Avnet IP we are adding is a wireless manager block that deals with all the wireless radio interfaces that are available via the Murata module located on the MiniZed.

```
53  # ---------------------------------------------------------------------------
54
55  # Generate Avnet IP
56  # puts "***** Generating MiniZed Wireless Manager IP..."
57
58  # Add Avnet IP Repository
59  # The IP_REPO_PATHS looks for a <component>.xml file, where <component> is the name of the IP to add to the catalog. The XML file identifies the various files that define the IP.
60  # The IP_REPO_PATHS property does not have to point directly at the XML file for each IP in the repository.
61  # The IP catalog searches through the sub-folders of the specified IP repositories, looking for IP to add to the catalog.
62  puts "***** Updating Vivado to include IP Folder"
63
64  set_property ip_repo_paths  {./IP/ ../../ip_repo/PWM_w_Int_1.0} [current_project]
65  update_ip_catalog
66
67  # General Config
68  puts "***** General Configuration for Design..."
69  set_property target_language VHDL [current_project]
70
71  # Add Minized-specific RTL design sources
72  # Set 'sources_1' fileset object
73  set obj [get_filesets sources_1]
74  set files [list \
75   "[file normalize "./IP/wireless_mgr.vhd"]"\
76  ]
77  add_files -norecurse -fileset $obj $files
78
79  # Add RTL-based IP modules without having to go through the process of packaging the RTL as an IP to be added through the Vivado IP catalog.
80  # See chap 12 'Referencing RTL Modules' https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_1/ug994-vivado-ip-subsystems.pdf
81
82  source ./make_RTL_ip.tcl
83  avnet_generate_RTL_ip wireless_mgr wireless_mgr_0
84
85  # ---------------------------------------------------------------------------------------------------
```

Figure 3 -- Adding Wireless Manager

3. Scrolling down to around line 92, we begin creating various ports, to be added to the block design.

```
92   # Create interface ports
93   set iic_rtl_0 [ create_bd_intf_port -mode Master -vlnv xilinx.com:interface:iic_rtl:1.0 iic_rtl_0 ]
94   set iic_rtl_1 [ create_bd_intf_port -mode Master -vlnv xilinx.com:interface:iic_rtl:1.0 iic_rtl_1 ]
95   set iic_rtl_2 [ create_bd_intf_port -mode Master -vlnv xilinx.com:interface:iic_rtl:1.0 iic_rtl_2 ]
96   set pl_led_g [ create_bd_intf_port -mode Master -vlnv xilinx.com:interface:gpio_rtl:1.0 pl_led_g ]
97   set pl_led_r [ create_bd_intf_port -mode Master -vlnv xilinx.com:interface:gpio_rtl:1.0 pl_led_r ]
98   set pl_sw_1bit [ create_bd_intf_port -mode Master -vlnv xilinx.com:interface:gpio_rtl:1.0 pl_sw_1bit ]
99
100  # Create ports
101  set BT_CTS_IN_N [ create_bd_port -dir O BT_CTS_IN_N ]
102  set BT_HOST_WAKE [ create_bd_port -dir I BT_HOST_WAKE ]
103  set BT_REG_ON [ create_bd_port -dir O BT_REG_ON ]
104  set BT_RTS_OUT_N [ create_bd_port -dir I BT_RTS_OUT_N ]
105  set BT_RXD_IN [ create_bd_port -dir O BT_RXD_IN ]
106  set BT_TXD_OUT [ create_bd_port -dir I BT_TXD_OUT ]
107  set WL_HOST_WAKE [ create_bd_port -dir I WL_HOST_WAKE ]
108  set WL_REG_ON [ create_bd_port -dir O WL_REG_ON ]
109  set WL_SDIO_CLK [ create_bd_port -dir O -type clk WL_SDIO_CLK ]
110  set WL_SDIO_CMD [ create_bd_port -dir IO WL_SDIO_CMD ]
111  set WL_SDIO_DAT [ create_bd_port -dir IO -from 3 -to 0 WL_SDIO_DAT ]
```

Figure 4 – Ports

4. Now looking at around line 126 to line 603, this is where we add all the other IP blocks to the system along with setting the properties of these blocks and existing blocks. Looking at the script you will notice we are adding/changing the following blocks along with an explanation as to what they are being used for.

    a. axi_gpio_0 – This GPIO block controls the user LEDs.

    b. axi_gpio_1 – This GPIO block controls the PL 1 bit switch.

    c. axi_iic_0 – This IIC controller controls the IIC communication between the Zynq device and the onboard accelerometer/temperature reader.

    d. axi_iic_1 – This IIC controller controls the I2C signals being sent to Pmod 1. This IIC interface is used to communicate to your TE Pmod in the Developing Zynq Software Speedway.

    e. axI_iic_2 - This IIC controller controls the I2C signals being sent to Pmod 2. This IIC interface is used to communicate to your TE Pmod in the Developing Zynq Software Speedway.

    f. bluetooth_uart – This block controls the bluetooth communication.

    g. processing_system7_0 – Zynq processor modifications are made to accommodate new IP.

    h. axi_mem_intercon – AXI interconnect is modified to accommodate new GPIO/IIC IP blocks.

    i. xlconcat_0 – This xlconcat block is used to concatenate the various interrupt bus signals that are sent to the Zynq device.

    j. xlconstant_1 – This constant block provides a constant value of 0 to the SDIO1_CDN and SDIO1_WP on the Zynq device.

5. Scrolling down further, around line 603 to line 655, this is where all the block design connections are being made. You will notice one section is labeled interface connections and the other port connections. Interface connections refers to block to block connections while port connections refers to connecting the IP blocks to the various ports we generated earlier in the script.

6. Looking at around line 657 to line 681 we accomplish many tasks.

    a. First we regenerate the layout so all of our Block Designs look the same.

    b. Next we set the addresses of various IP that are existing and new to the Block Design.

    c. After that we add all the constraints to route out GPIO and I2C signals from the various IP blocks we just added.

    d. Finally we save our Vivado project, then run Synthesis and Implementation followed by generating the Bitstream for our Hardware Project. Generating the bitstream will take anywhere from 15-40 minutes based on the machine you are running Vivado on.

7. Now returning to Vivado, after Bitstream generation is complete you will be prompted Open Implemeted Design, select **OK**

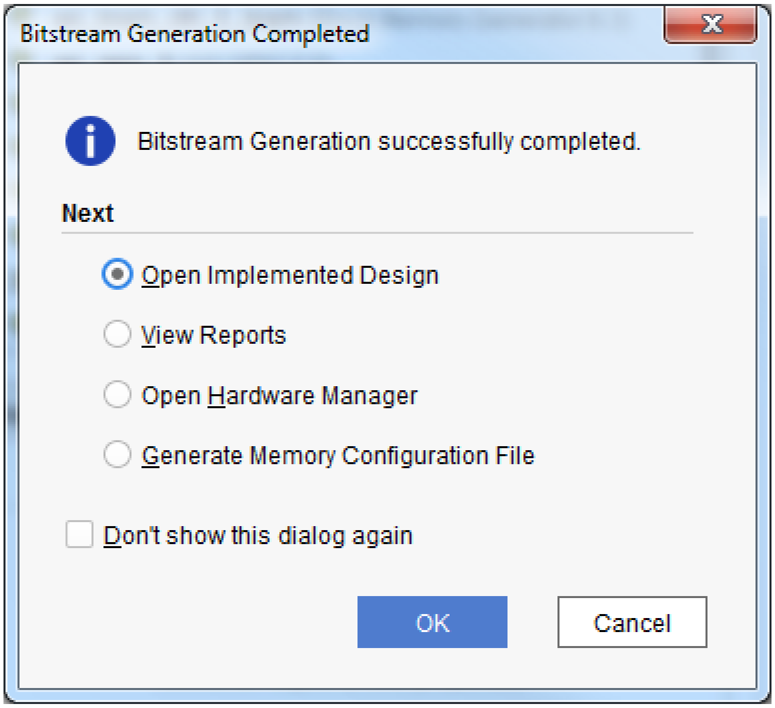

Figure 5 -- Open Implemented Design

8. After you finish opening the implemented design, return to the block design to view in the block diagram all the new IP that was added. See figure below

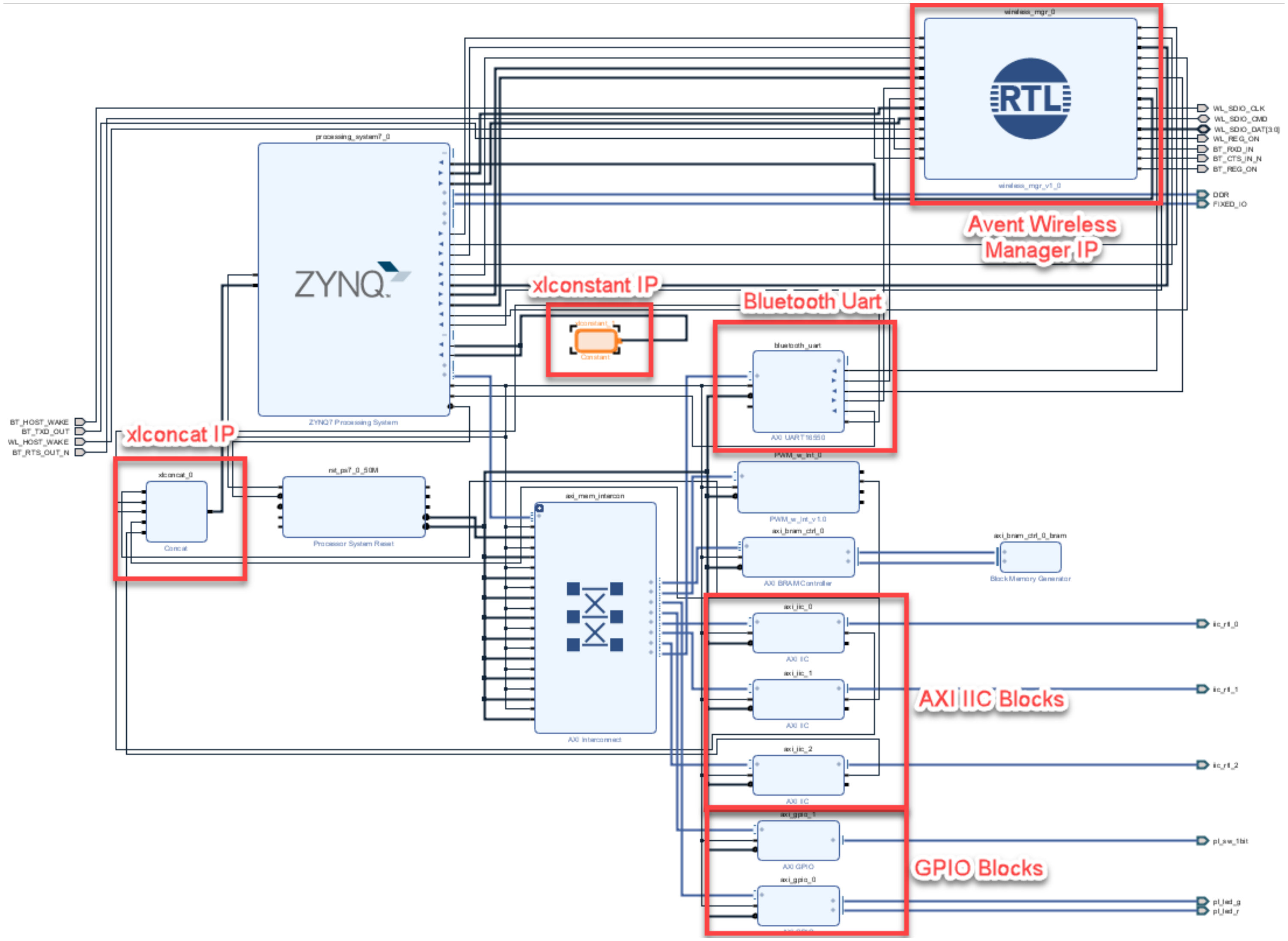

Figure 6 -- Final Hardware Platform

9. Now remembering back to everything else we added in the lab9.tcl script let us take a look at the constraints we added. In the Sources Window under constraints **double click** on MiniZed_Speedway.xdc.

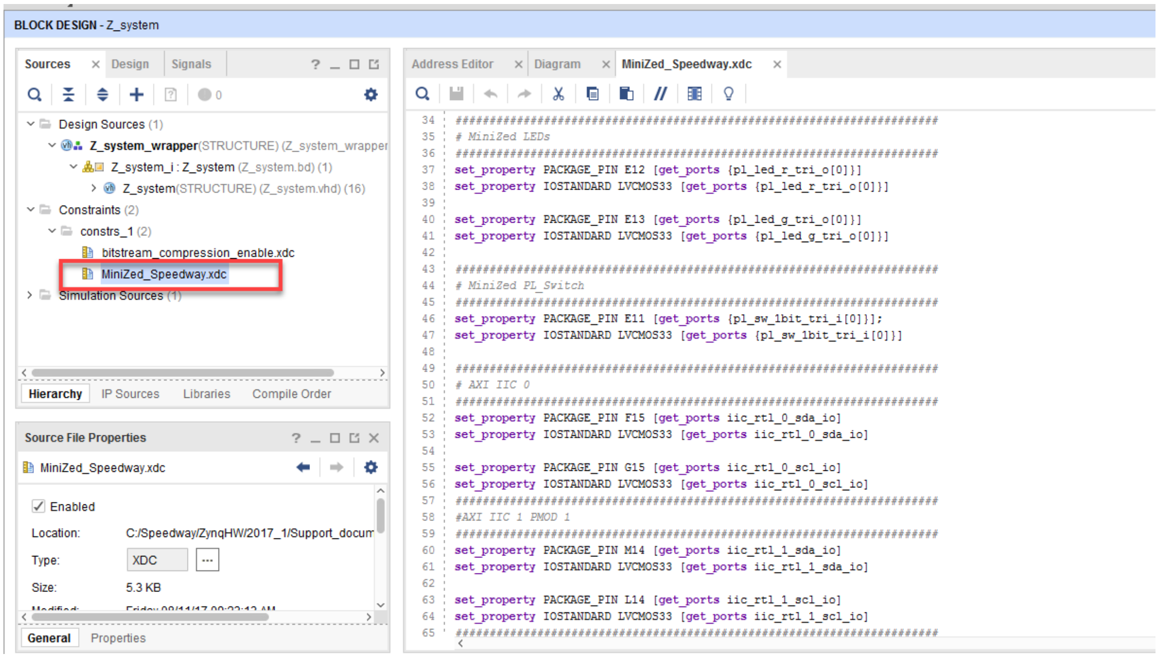

Figure 7 -- Open Constraints File

10. Looking through the MiniZed_Speedway.xdc file you will notice the various constraints that will route your interface signals through the Programmable Logic out to the various interfaces on your MiniZed. A few interfaces of note being routed are the AXI IIC controllers, GPIOs, and a Wireless Manager interface. One thing you will notice is that each interfaces IO standard is set to LVCMOS33, this is the electrical interface associated with the various interfaces. Please refer to the MiniZed Hardware user guide if you have any questions in regards to the MiniZed's electrical interfaces.

11. Now please look at the bitstream_compression_enabled.xdc. In this constraint file we set the Bitstream.General.Compress flag as true. What this does is compress the Bitstream generated from the design. Setting this flag as true is useful for instances that you have a smaller Boot storage such as the MiniZeds QSPI. For more information please refer Xilinx AR# 16996

Experiment 3: Delivering Hardware to the Software Team

In this experiment we go through the steps to export our hardware platform to a location in which we will use it for the Developing Zynq Software Speedway.

Experiment 3 General Instruction:

Export Hardware for Developing Zynq Software Speedway

Experiment 3 Step-by-Step Instructions:

1. Now that we have thoroughly explored the changes made to the Vivado project made by the lab9.tcl script, reopen the Implemented Design.

2. Export the hardware design (File → Export → Export Hardware) to the following path for the SW SpeedWay.

   C:/SpeedWay/ZynqSW/2017_1/ZynqDesign/MiniZ/ for MiniZed users

   Since the software team will need to make use of the custom IP that is in the PL, make sure you export the bitstream as well. Click **OK**.

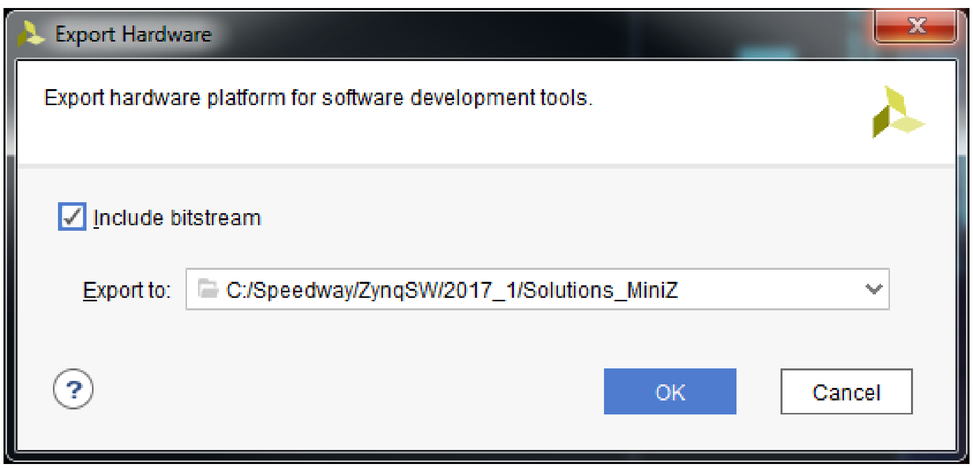

Figure 8 – Export hardware for SDK

3. Close Vivado.

Exploring Further

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

- Please take the Developing Zynq Software Speedway

This concludes Lab 9.

Revision History

Date	Version	Revision
16 Aug 17	01	Initial Draft

Resources

www.minized.org

www.microzed.org

www.picozed.org

www.zedboard.org

www.xilinx.com/zynq

www.xilinx.com/sdk

www.xilinx.com/vivado