Creating and Using Platform for an Application

Introduction

This lab guides you through the steps of creating a custom platform for an audio application.

Objectives

After completing this lab, you will be able to:

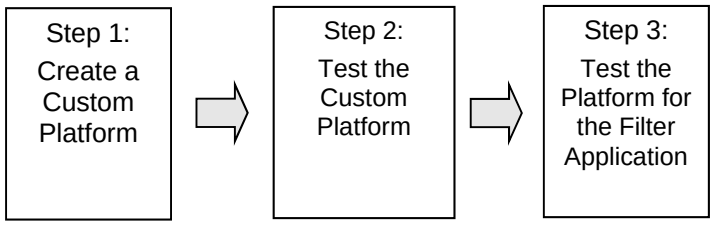
- Create an SDx platform for an custom application
- Use the SDx environment to test the platform for an audio filtering Standalone application

Procedure

This lab is separated into steps that consist of general overview statements that provide information on the detailed instructions that follow. Follow these detailed instructions to progress through the lab.

This lab comprises three primary steps: You will create an SDx project, test the custom platform, and test the platform for the filter application.

General Flow for this Lab



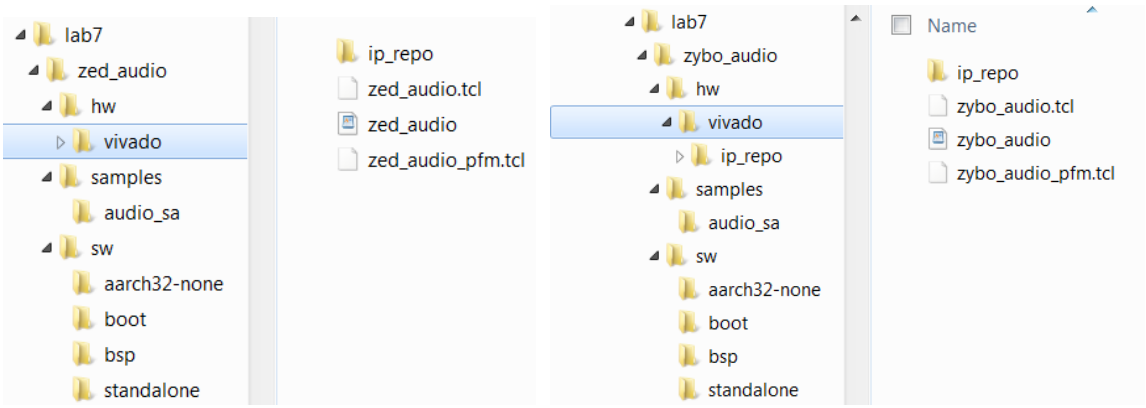
Create a Custom Platform Step 1

1-1. Launch Vivado, create the platform design, and generate an archive of the project.

1-1-1. Using the Windows Explorer, copy the **zed_audio** directory (for Zed) or **zybo_audio** directory (for Zybo) from the *source\lab7* directory and place it in the *c:\xup\SDSoC\labs\lab7* directory.

This will copy all the necessary directories and files, creating the required directory structure.

Note another file (not shown below), called *zed_audio_sw.pfm* (for Zed) or *zybo_audio_sw.pfm*, (for zybo) is provided. Typically this will have to be hand created. The file describes the SDx software component. It defines file names and locations of the library and boot components.



(a) Zed **(b) Zybo**

Figure 1. The directory structure for creating the SDx platform

```
<?xml version="1.0" encoding="UTF-8"?>
<sdx:platform sdx:vendor="xilinx.com"
              sdx:library="sdx"
              sdx:name="zed_audio"
              sdx:version="1.0"
              sdx:schemaVersion="1.0"
              xmlns:sdx="http://www.xilinx.com/sdx" >

    <sdx:description>Platform targeting the ZedBoard for an audio application. More
    <sdx:systemConfigurations sdx:defaultConfiguration="standalone">
        <sdx:configuration sdx:name="standalone"
                           sdx:displayName="Standalone OS (Zynq 7000)"
                           sdx:defaultProcessorGroup="a9_0">
            <sdx:description>Standalone OS running on Zynq 7000</sdx:description>
            <sdx:bootImages sdx:default="standard">
                <sdx:image sdx:name="standard"
                           sdx:bif="boot/standalone.bif"
                           sdx:readme="boot/generic.readme"
                />
            </sdx:bootImages>
            <sdx:processorGroup sdx:name="a9_0"
                                sdx:displayName="A9_0"
                                sdx:cpuInstance="ps7_cortexa9_0"
                                sdx:cpuType="cortex-a9">
                <sdx:os sdx:name="standalone"
                        sdx:displayName="Standalone OS"
                        sdx:includePaths="aarch32-none/include"
                        sdx:ldscript="standalone/lscript.ld"
                        sdx:bspConfig="bsp/system.mss"
                />
            </sdx:processorGroup>
        </sdx:configuration>
    </sdx:systemConfigurations>
</sdx:platform>
```

Figure 2. The <board>_audio_sw.spfm file content

1-1-2. Open Vivado by selecting **Start > All Programs > Xilinx Design Tools > SDx 2016.3 > Vivado Design Suite > Vivado 2016.3**

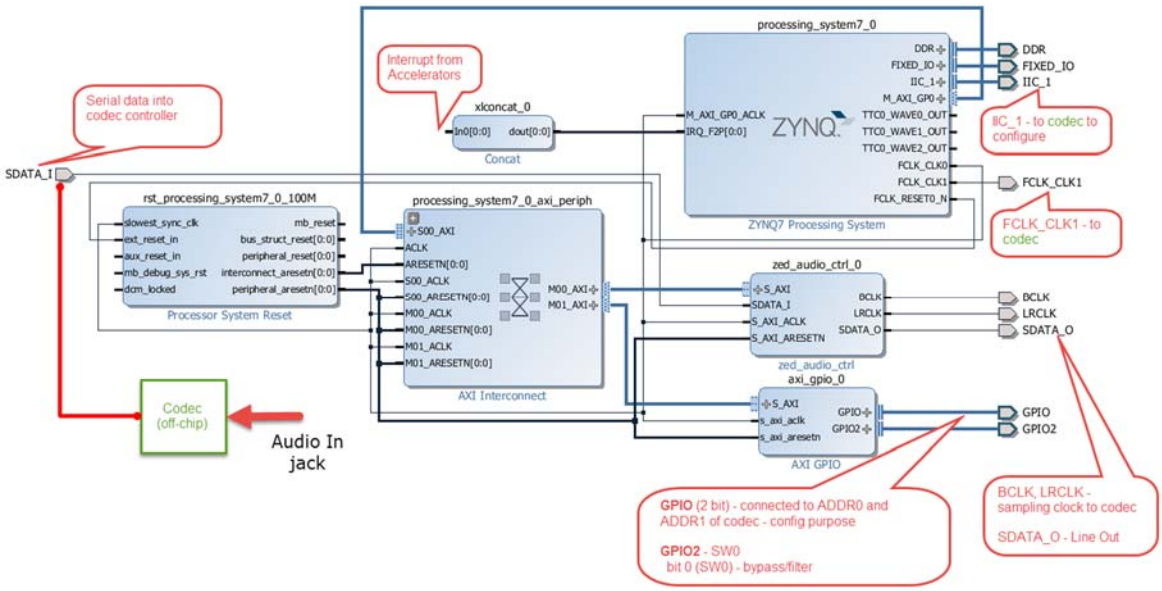
1-1-3. In the Vivado's Tcl Console window change the directory to the *c:/xup/SDSoC/labs/lab7/zed_audio/hw/vivado/* or *c:/xup/SDSoC/labs/lab7/zybo_audio/hw/vivado/* using the **cd** command.

```
cd c:/xup/SDSoC/labs/lab7/<zed | zybo>_audio/hw/vivado
```

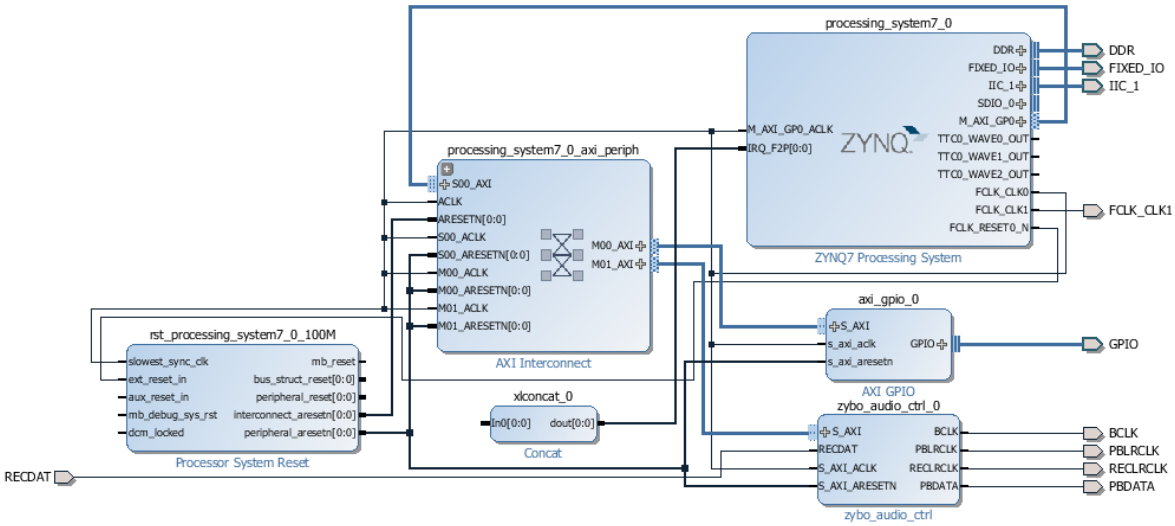
1-1-4. Execute the following command to generate the platform hardware.

```
source ./<zed | zybo>_audio.tcl
```

This will create an IPI design and an HDL wrapper, and add an xdc constraints file.



(a) Zed



(b) Zybo

Figure 3. The IPI design

1-1-5. Since the design contains an audio controller IP, which is not part of the standard Vivado installation IP, we need to archive the project so the custom IP is part of the platform.

1-1-6. Select **File > Archive Project…**

1-1-7. Click on the browse button of the *Temporary location* path and set it to **c:\temp** or a shorter path. Change the *Archive name* to **zybo_audio_hw_design**. Change the *Archive location* to **c:/xup/SDSoC/labs/lab7** or some other place than where the project was created. **Uncheck** the *Include configuration settings* and *Include run results* check boxes.

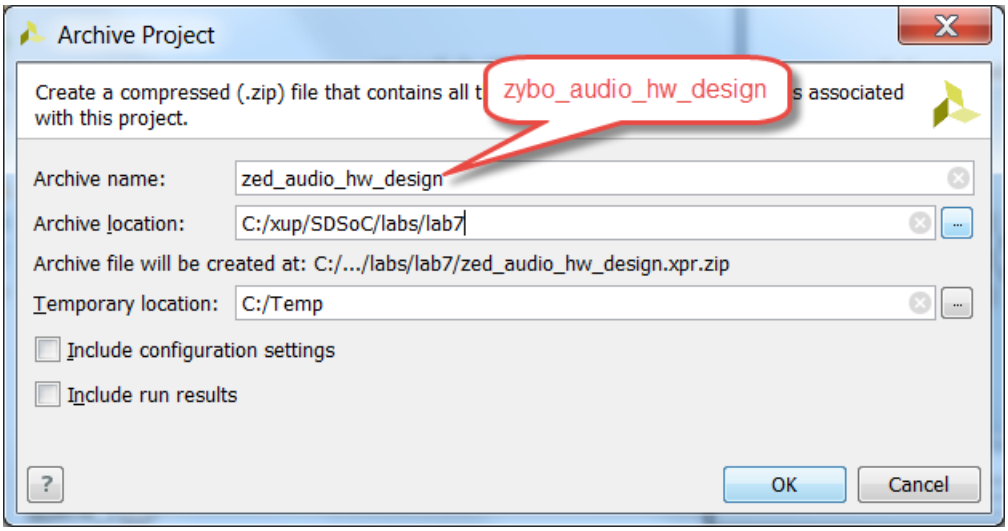


Figure 4. Archiving the project

1-1-8. Click **OK**.

This will generate the **zed_audio_hw_design.zip** or **zybo_audio_hw_design.zip** file in the specified directory. Note that although the zip filename is different, the folder inside is still *zed_audio* or *zybo_audio*.

1-1-9. Close the Vivado project by selecting **File > Close Project**.

1-2. Unzip the archived project and copy the relevant files/directories.

1-2-1. Unzip the *zed_audio_hw_design.zip* or *zybo_audio_hw_design.zip* into **c:\xup\SDSoc\labs.**

This is required since the zip file contains a root folder called *zed_audio* or *zybo_audio*.

1-2-2. Using the Windows Explorer, delete all the original files and directories, **except** the **_audio_pfm.tcl**, under the *c:\xup\SDSoC\labs\lab7\<board>_audio\hw\vivado* directory.

1-2-3. Copy everything from the extracted directory and place them in the *c:\xup\SDSoC\labs\lab7\<board>_audio\hw\vivado* directory.

1-2-4. Delete the *.cache, .hw, .ip_user_files* folders and the *.jou*, and *.log* files. Keep only the *.ipdefs, .srcs directories,* and the *_pfm.tcl, archive_project_summary.txt* and *.xpr* files.

1-3. Open the Vivado project.

1-3-1. Select **File > Open Project** in Vivado.

1-3-2. Browse to *C:/xup/SDSoC/labs/lab7/<zed | zybo>_audio/hw/vivado* and select <zed | zybo>_**audio.xpr**.

1-3-3. Open the block design.

The block design must be open in order to the next step.

1-3-4. Change directory by typing the following command in the Tcl console.

```
cd c:/xup/SDSoC/labs/lab7/<zed | zybo>_audio/hw/vivado
```

1-4. Generate the hardware description for the board.

1-4-1. Source the SDSoC platform creation tcl script by executing the following command:

```
source -notrace c:/Xilinx/SDx/2016.3/scripts/vivado/sdsoc_pfm.tcl
```

Replace `c:/Xilinx/SDx/2016.3` with the location of your installation if necessary.

1-4-2. Source the provided hardware platform creation tcl file by executing the following command:

```
source ./<board>_audio_pfm.tcl
```

Where <board> is either zed or zybo . You can open this file before or after executing it to examine its contents.

This command will execute the following commands to generate the <board>_audio_hw.hpfm where <board> is either zed or zybo.

```
set pfm [sdsoc::create_pfm <board>_audio_hw.hpfm]
```

This script will execute the following commands to generate the <board>_audio_hw.hpfm.

Define VLNV:

```
sdsoc::pfm_name $pfm "xilinx.com" "xd" "<board>_audio" "1.0"
```

Declare a brief platform description:

```
sdsoc::pfm_description $pfm "Zynq <board> with audio codec"
```

Define the main clock which will be used as the default clock along with any other clock domains that SDx may use while generating accelerators.

```
sdsoc::pfm_clock $pfm FCLK_CLK0 processing_system7_0 0 true
rst_ps7_0_100M
```

Note that FCLK_CLK0 is the clock domain which will be used by default. The *processing_system7_0* is the instance name of the processor and *rst_processing_system7_0_100M* is the instance name of the processor reset block associated to the clock domain. The "*0*" before the *true* indicates the clock number, and *true* indicates that this is the default clock which will be used by SDx, unless the user selects a different clock, for example, in the case of a multi-clock design.

Declare the platform AXI bus interfaces by executing the following commands:

```
sdsoc::pfm_axi_port $pfm M_AXI_GP1 processing_system7_0 M_AXI_GP
```

In this section you define the AXI ports which you want SDx to use when connecting accelerators. In our design, M_AXI_GP0 is being used to communicate with the CODEC controller and an AXI GPIO through the `processing_system7_0_axi_periph` instance (of AXI_Interconnect). Since our FIR filter will connect to the processor using an AXI-Lite interface, the first command will allow it to connect to M_AXI_GP1.

```
sdsoc::pfm_axi_port $pfm S_AXI_ACP processing_system7_0 S_AXI_ACP
```

SDx expects at least one slave interface so the hardware accelerator can use DDR memory and/or DMA. The above command instructs SDx that S_AXI_ACP may be used for that purpose.

If you would like to allow other interfaces to be used by SDx then use the appropriate commands from the listed below.

```
sdsoc::pfm_axi_port $pfm S_AXI_HP0 processing_system7_0 S_AXI_HP
sdsoc::pfm_axi_port $pfm S_AXI_HP1 processing_system7_0 S_AXI_HP
sdsoc::pfm_axi_port $pfm S_AXI_HP2 processing_system7_0 S_AXI_HP
sdsoc::pfm_axi_port $pfm S_AXI_HP3 processing_system7_0 S_AXI_HP
```

You may see a warning that can be ignored.

Define the available interrupts by executing the following command:

```
for {set i 0} {$i < 16} {incr i} {

sdsoc::pfm_irq $pfm In$i xlconcat_0

}
```

The above command makes all sixteen interrupt pins available to SDx. If the user defined platform requires some interrupts, they will occupy positions starting at interrupt 0 and in such a case `set i 0` should be set to the next available interrupts. e.g. if two 2 interrupts are required by the platform (interrupts 0,1) the SDx interrupts will start at 2. *i* should be set to 2 in the above command:.

Generate the platform hardware description metadata file by executing the following command:

```
sdsoc::generate_hw_pfm $pfm
```

This will generate the **zed_audio_hw.hpfm** or **zybo_audio_hw.hpfm** file in the *vivado* directory.

1-4-3. Using the Windows Explorer, copy the ***.hpfm** file to the *c:\xup\SDSoC\labs\lab7\<zed | zybo>_audio\hw* directory.

1-5. Export the vivado project and generate the software description.

1-5-1. In Vivado, export the hardware by selecting **File > Export > Export Hardware**

The *Generate Output Products* dialog box will appear.

1-5-2. Click on the **Generate Output Products** button.

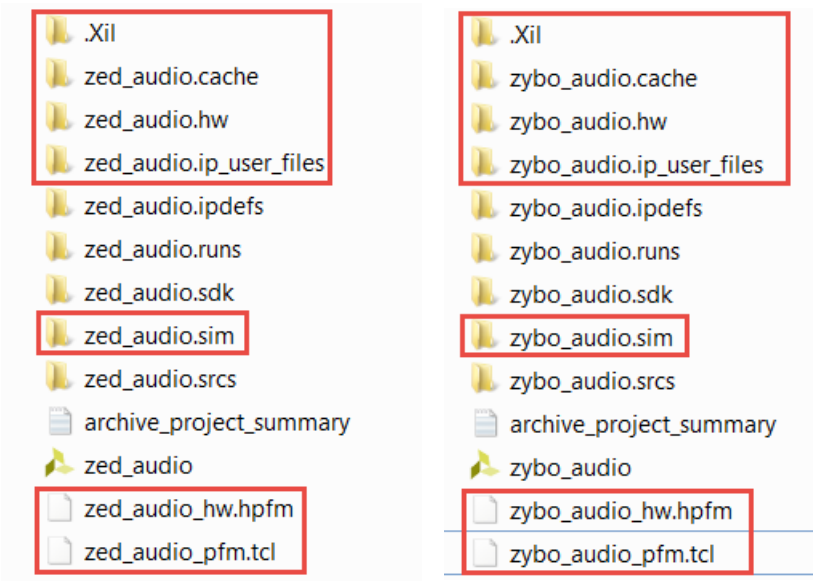
This will generate the output products and present another dialog box. Click **OK**.

The **zed_audio.sdk** or **zybo_audio.sdk** directory will be created under the *vivado* directory.

1-5-3. Click **OK** to close the critical warning window if displayed.

1-5-4. Close the project by selecting **File > Close Project**

1-5-5. Using the Windows Explorer, delete the files and directories, under *vivado* directory, shown in the red boxes below.



(a) Zed **(b) Zybo**

Figure 5. Directory content

1-6. Build the software templates

1-6-1. Open SDx by selecting **Start > All Programs > Xilinx Design Tools > SDx 2016.3 > SDx IDE 2016.3**

The workspace dialog box will open.

1-6-2. Click on the **Browse** button, select the *c:\xup\SDSoC\labs\lab7\zed_audio\hw\vivado\zed_audio.sdk* (for zed) OR *c:\xup\SDSoC\labs\lab7\zybo_audio\hw\vivado\zybo_audio.sdk* (for zybo) directory and click **OK**. Click **OK** again.

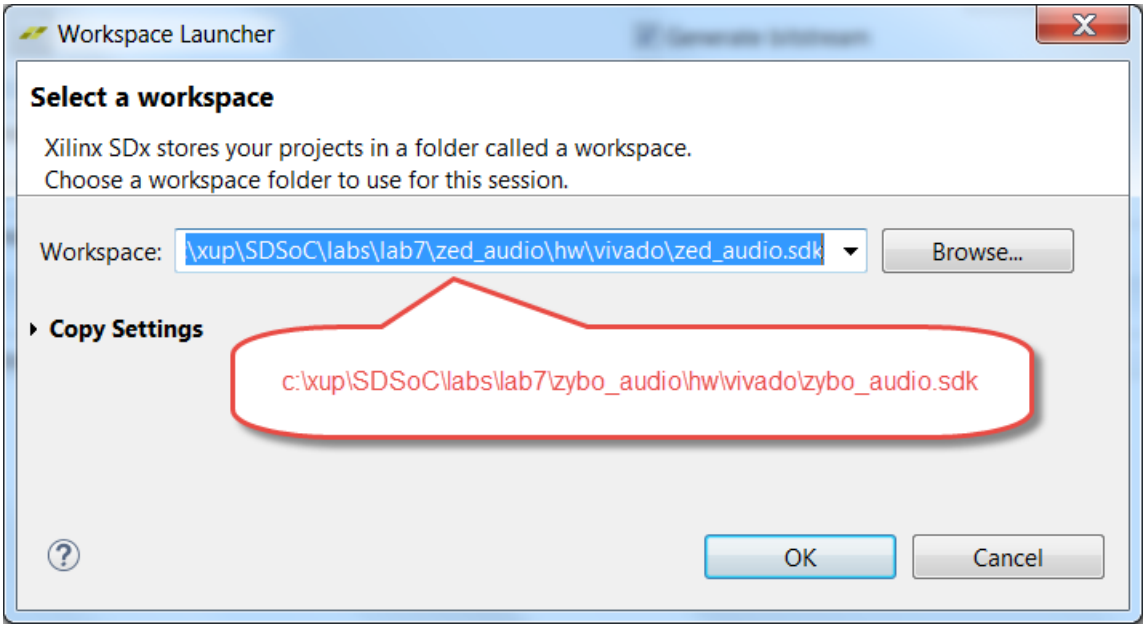


Figure 6. Selecting the workspace

1-6-3. Close the **Welcome** Page.

1-6-4. Select **File > New > Project,** then expand Xilinx, select *Hardware Platform Specification* and click **Next**.

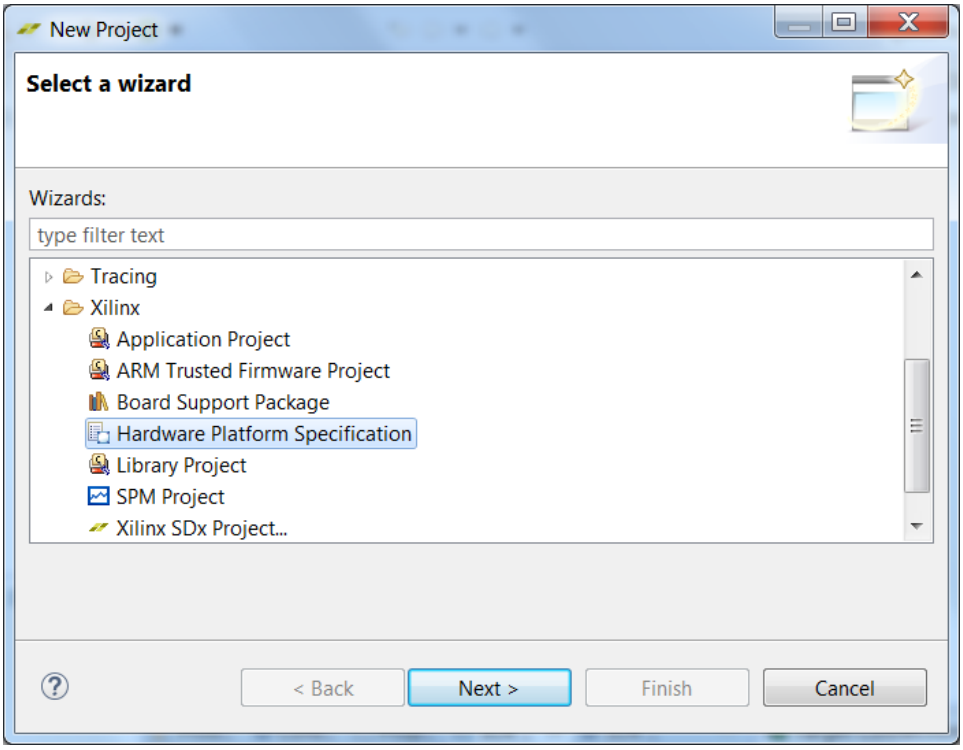


Figure 7. Creating the hardware platform specification project

The hardware platform specification describes the hardware design. This includes a full system memory map, the type(s) of processors present, active peripherals in the PS and PL for Zynq systems or a list of all peripherals for a non-Zynq systems.

Based on this description, software such as the board support package (BSP) and application can be tailored to the hardware

1-6-5. Enter **zed_audio** or **zybo_audio** as the *project name*, click on the browse button of *Target Hardware Specification* and browse to *c:\xup\SDSoC\labs\lab7\<zed | zybo>_audio\hw\vivado\<zed | zybo>_audio.sdk*, select *<zed | zybo>_audio_wrapper.hdf*, click **Open**, and then click **Finish**.

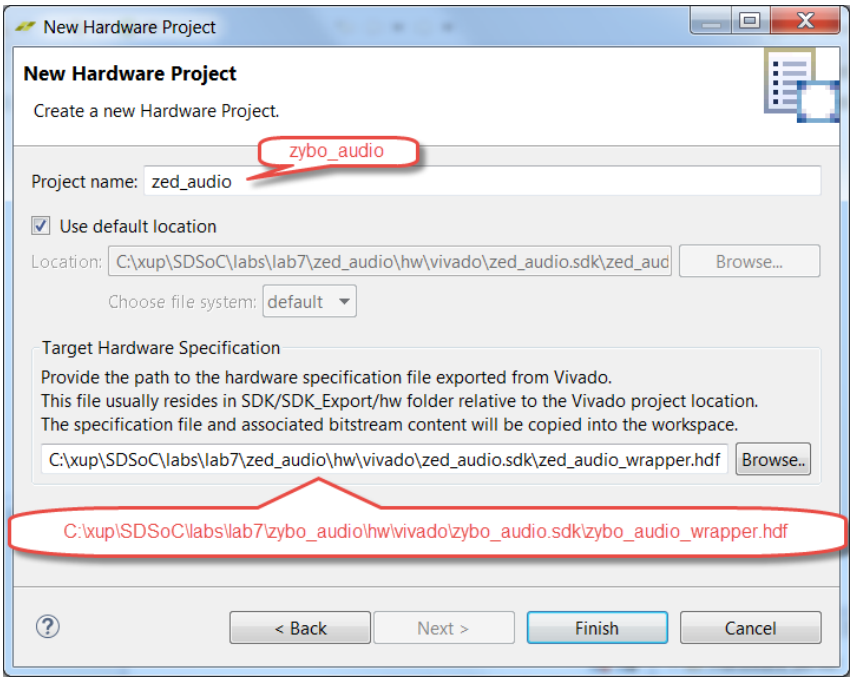


Figure 8. Creating hardware project in SDx

1-6-6. Select **File > New > Project**, then expand **Xilinx** and select **Board Support Package** and click **Next**.

1-6-7. Click **Finish** with *standalone_bsp_0* as the *Project name*, making sure that *zed_audio* or *zybo_audio* is selected as the *Hardware Platform*.

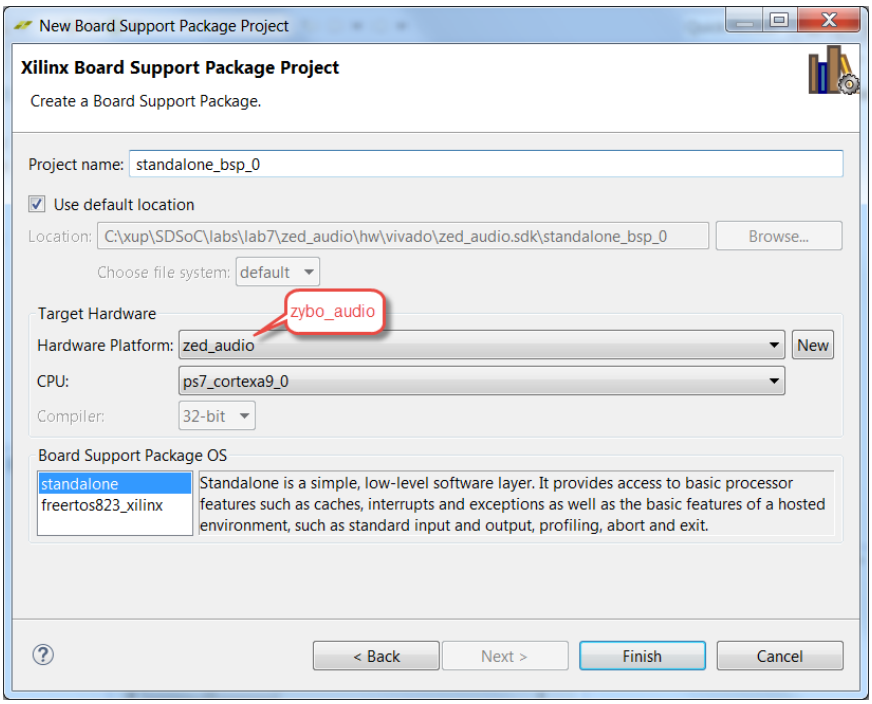


Figure 9. Creating the board support package for the platform

The *Board Support Package Settings* will open.

1-6-8. Select *xilffs* library and click **OK**.

1-6-9. Right-click on the *standalone_bsp_0* and select **build project**.

1-7. Generate the FSBL application so the board can be boot from the SD card.

1-7-1. Select **File> New > Application Project**

1-7-2. Enter **fsbl** in the *Project name* field.

1-7-3. For the Board Support Package, select **Create New** and click **Next.**

1-7-4. Select **Zynq FSBL** from the *Available Templates* pane, and click **Finish.**

1-7-5. Click **Yes** to open the C/C++ perspective.

1-7-6. Expand the **src** folder under the *fsbl* project in the Project Explorer pane and double-click the **lscript.ld** entry.

Notice that the Heap size is assigned as 0x2000 by default. If your application is going to transfer a large amount of data using sds_alloc, you may need more Heap space.

1-7-7. Change the Heap size to 0x4000 and then press **Ctrl+S** to save the changes

1-7-8. Right-click on the *fsbl_bsp* entry and select **build project.**

1-7-9. Right-click on the *fsbl* entry and select **build project.**

The *fsbl.elf* file is required to create a bootable SD image.

1-7-10. Using the Windows Explorer, copy the following files and directories

The **system.mss** file from *c:\xup\SDSoC\labs\lab7\<zed | zybo>_audio\hw\vivado\<zed | zybo>_audio.sdk\standalone_bsp_0* into *c:\xup\SDSoC\labs\lab7\<zed | zybo>_audio\sw\bsp*

The **include** directory from *c:\xup\SDSoC\labs\lab7\<zed | zybo>_audio\hw\vivado\<zed | zybo>_audio.sdk\standalone_bsp_0\ps7_cortexa9_0* into *c:\xup\SDSoC\labs\lab7\<zed | zybo>_audio\sw\aarch32-none*

The **lscript.ld** file from *c:\xup\SDSoC\labs\lab7\<zed | zybo>_audio\hw\vivado\<zed | zybo>_audio.sdk\fsbl\src* into *c:\xup\SDSoC\labs\lab7\<zed | zybo>_audio\sw\standalone*

Copy the **fsbl.elf** file from *c:\xup\SDSoC\labs\lab7\<zed | zybo>_audio\hw\vivado\<zed | zybo>_audio.sdk\fsbl\Debug* into *c:\xup\SDSoC\labs\lab7\<zed | zybo>_audio\sw\boot*

1-7-11. The above copying of the files was done to match what is pointed in the provided <board>_audio_sw.spfm file.

```
<?xml version="1.0" encoding="UTF-8"?>
<sdx:platform sdx:vendor="xilinx.com"
              sdx:library="sdx"
              sdx:name="zed_audio"
              sdx:version="1.0"
              sdx:schemaVersion="1.0"
              xmlns:sdx="http://www.xilinx.com/sdx" >

    <sdx:description>Platform targeting the ZedBoard for an audio application. More
    <sdx:systemConfigurations sdx:defaultConfiguration="standalone">
        <sdx:configuration sdx:name="standalone"
                           sdx:displayName="Standalone OS (Zynq 7000)"
                           sdx:defaultProcessorGroup="a9_0">
            <sdx:description>Standalone OS running on Zynq 7000</sdx:description>
            <sdx:bootImages sdx:default="standard">
                <sdx:image sdx:name="standard"
                           sdx:bif="boot/standalone.bif"
                           sdx:readme="boot/generic.readme"
                />
            </sdx:bootImages>
            <sdx:processorGroup sdx:name="a9_0"
                                sdx:displayName="A9_0"
                                sdx:cpuInstance="ps7_cortexa9_0"
                                sdx:cpuType="cortex-a9">
                <sdx:os sdx:name="standalone"
                        sdx:displayName="Standalone OS"
                        sdx:includePaths="aarch32-none/include"
                        sdx:ldscript="standalone/lscript.ld"
                        sdx:bspConfig="bsp/system.mss"
                />
            </sdx:processorGroup>
        </sdx:configuration>
    </sdx:systemConfigurations>
</sdx:platform>
```

The **bif** file looks for the **fsbl.elf** in the boot directory

Figure 10. The <board>_audio_sw.spfm file

Test the Built Platform — Step 2

2-1. In SDx change the workspace to c:\xup\SDSoC\labs\lab7. Create a new SDx project called *audio_test* using zed_audio or zybo_audio as the platform and Standalone as the OS, and selecting Audio Playback template provided in the *samples* directory.

2-1-1. In SDx change the workspace to *c:\xup\SDSoC\labs\lab7* by selecting **File > Switch Workspace > other**.

2-1-2. Click **OK**.

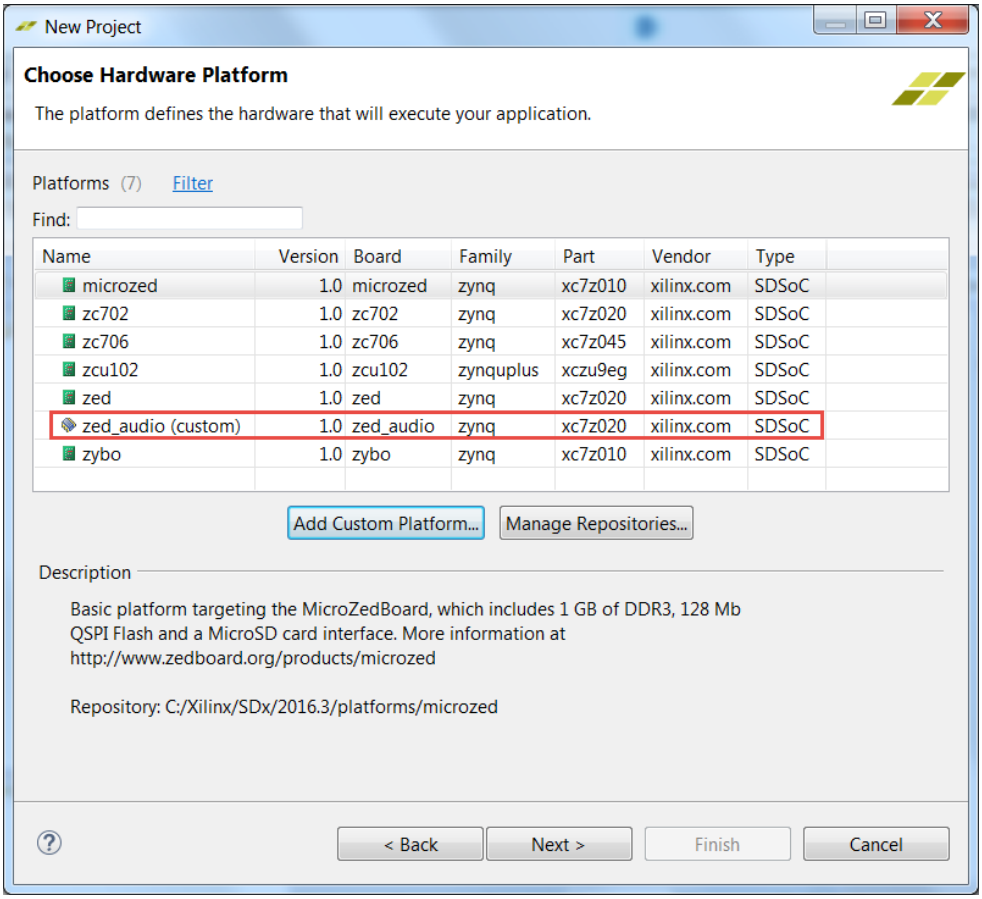
2-1-3. Close the **Welcome** page.

2-1-4. Click on the *Create SDx Project* in the Welcome tab or select **File > New > Xilinx SDx Project**

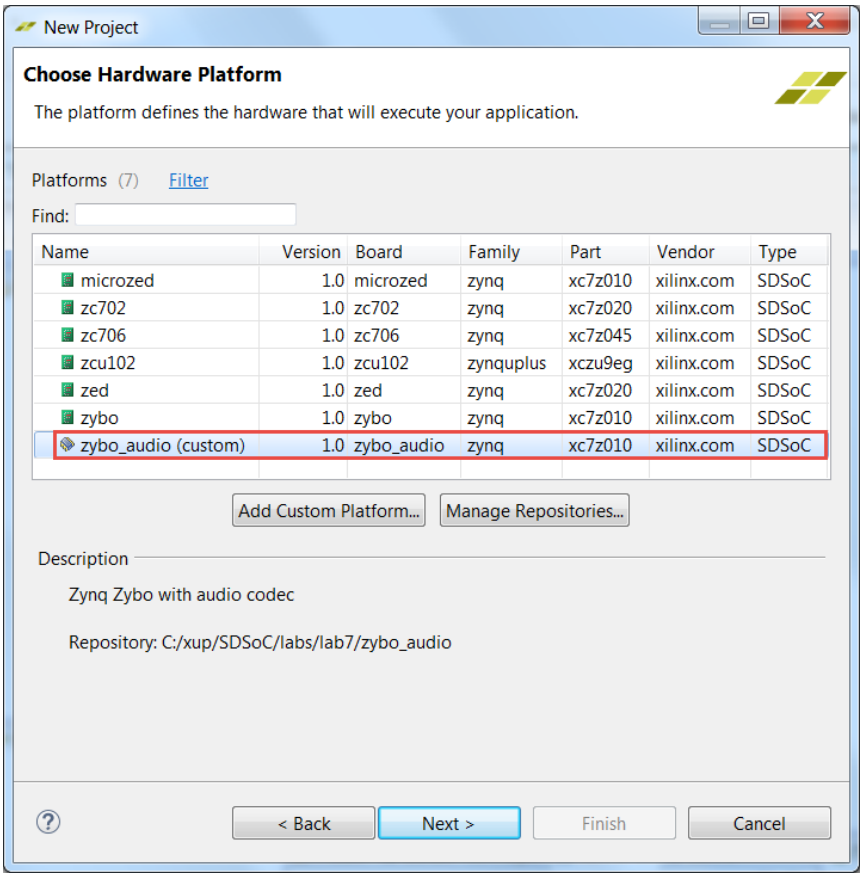
2-1-5. Enter **audio_test** in the *Project name* field and click **Next**.

2-1-6. Click **Add Custom Platform…,** browse to *c:\xup\SDSoC\labs\lab7* and select either **zed_audio** or **zybo_audio** and click **OK**.

The custom platform entry will appear in the available.



(a) Zed



(b) Zybo

Figure 11. Selecting and adding the custom platform

2-1-7. Select *zed_audio* or *zybo_audio* and click **Next**.

2-1-8. For the *OS*, select **Standalone**.

2-1-9. Click **Next**.

The Templates window will be displayed with Audio Playback as one of the two possible templates. This entry is picked up from the samples directory of the created platform.

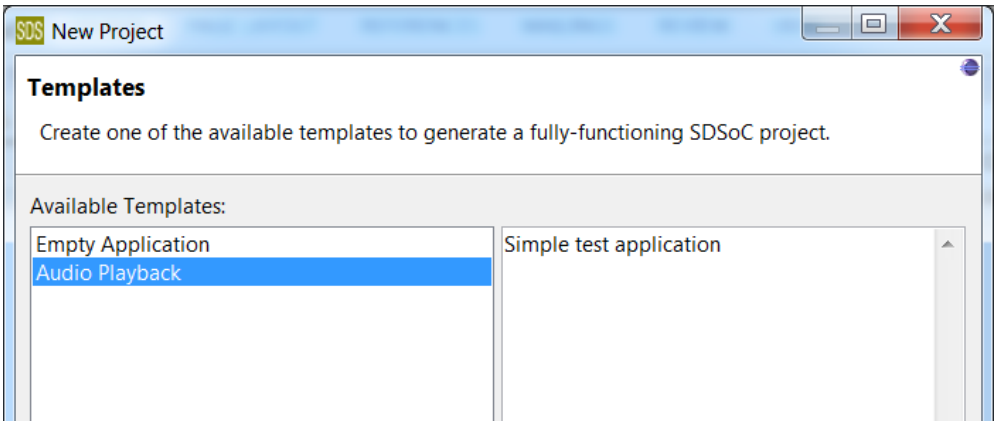


Figure 12. Selecting a test template

2-1-10. Select the *Audio Playback* application and click **Finish**.

2-1-11. Expand the *audio_test* entry in the Project Explorer pane and note the two source files (audio.h and audio.c) make up the application.

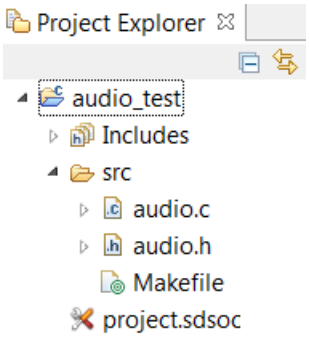


Figure 13. Test application directory

The *audio.c* application configures the CODEC, samples the CODEC and writes back into the CODEC illustrating the platform does function.

2-1-12. Uncheck Generate SD Card Image box as we will test it using JTAG mode.

2-1-13. Right-click on the *audio_test* entry and select **Build Project**.

2-2. Connect the board and test the application.

2-2-1. Connect an audio patch cable between the PC's headphone output and Line-In connector of the board.

2-2-2. Connect a headphone to the Line Out connector (on Zed) or HPH OUT connector (on Zybo) of the board.

2-2-3. Connect the board and power it ON.

2-2-4. Right-click on the *audio_test* folder and select **Run As > Launch on Hardware (SDSoC Debugger)** to run the application.

This will download the bit file to configure the FPGA, download the *audio_test.elf* application, and run the application.

2-2-5. Play some music on the PC and you should be able to hear the same on your headphone.

2-2-6. When satisfied, power OFF the board.

Test the Platform for the Filter Application Step 3

3-1. Create a fir_test application targeting the custom platform and Standalone OS. Import the provided audio.h, audio.c, and fir_test.c files from the c:\xup\SDSoC\source\lab7 folder.

3-1-1. Select **File > New > Xilinx SDx Project**

3-1-2. Enter **fir_test** in the *Project name* field and click **Next**.

3-1-3. Select either **zed_audio** or **zybo_audio** and click **Next**.

3-1-4. Select **Standalone** as the *Target OS* and click **Next**.

3-1-5. Select *Empty Application* template and click **Finish**.

3-1-6. Expand **fir_test > src**, right-click on the *src* folder and select **Import…**

3-1-7. Expand General, select File System and click **Next**

3-1-8. Browse to *c:\xup\SDSoC\source\lab7*, and import **audio.c, audio.h, fir_coef.dat** and **fir_test.c** files.

3-1-9. Add the **fir** function in the *HW Function* panel.

A java error message may appear. Click OK to ignore it.

3-1-10. Right-click on the **fir_test** entry in the *Project Explorer* panel, and select **Build Project**.

This will take about 15 minutes.

3-1-11. When the build is complete, using the Windows Explorer, copy the **BOOT.BIN** file from *c:\xup\SDSoC\labs\lab7\fir_test\Debug\sd_card* into the SD card.

3-2. Connect the board and test the application.

3-2-1. Connect an audio patch cable between the PC's headphone output and Line-In connector of the board.

3-2-2. Connect a headphone to the Line Out connector (on Zed) or HPH OUT connector (on Zybo) of the board.

3-2-3. Connect the board and power it ON.

3-2-4. Right-click on the *fir_test* folder and select **Run As > Launch on Hardware (SDSoC Debugger)** to run the application.

This will download the bit file to configure the FPGA, download the *fir_test.elf* application, and run the application.

3-2-5. Play some music on the PC and you should be able to hear the same on your headphone.

3-2-6. When satisfied, turn OFF the board and exit the SDx program.

3-3. Open Vivado and view the built design.

3-3-1. Start Vivado by selecting **Start > All Programs > Xilinx Design Tools > SDx 2016.3 > Vivado Design Suite > Vivado 2016.3**

3-3-2. Click the **Open Project** link, open the design by browsing to *c:\xup\SDSoC\labs\lab7\fir_test\Debug_sds\p0\ipi* and selecting either the **zybo_audio.xpr** or **zed_audio.xpr**.

3-3-3. Click on **Open Block Design** in the *Flow Navigator* pane. The block design will open.

3-3-4. Click on the **show interface connections only** ( ) button followed by click on the **regenerate layout** ( ) button.

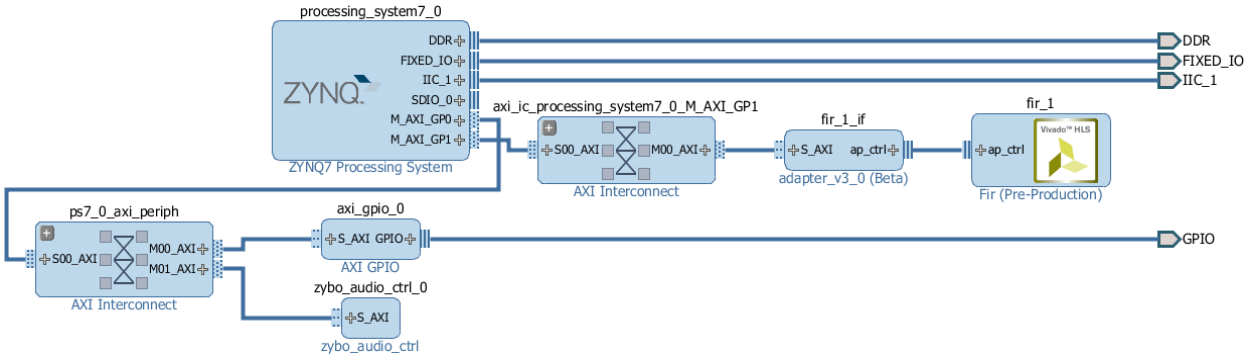


Figure 14. The generated block design

You can see *Fir* filter instance and the datamover adapter.

3-3-5. Close Vivado by selecting **File > Exit**

Conclusion

In this lab, you created a custom platform utilizing an audio CODEC IP in the base design. You then created a test application using the provided test template to test the custom platform. You then created a user application, importing the provided source files, targeted a *fir* function in hardware and then tested the application.