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:

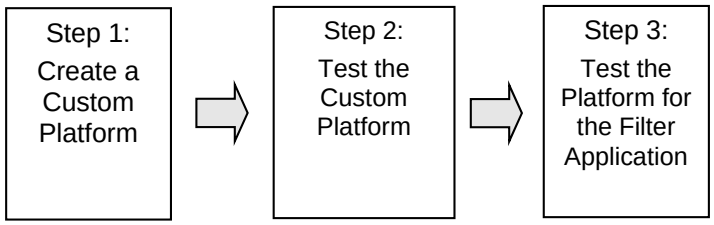
- Create an SDSoC platform for an custom application
- Use the SDSoC environment to test the platform for an audio filtering Standalone application

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 SDSoC project, test the custom platform, and test the platform for the filter application.

General Flow for this Lab



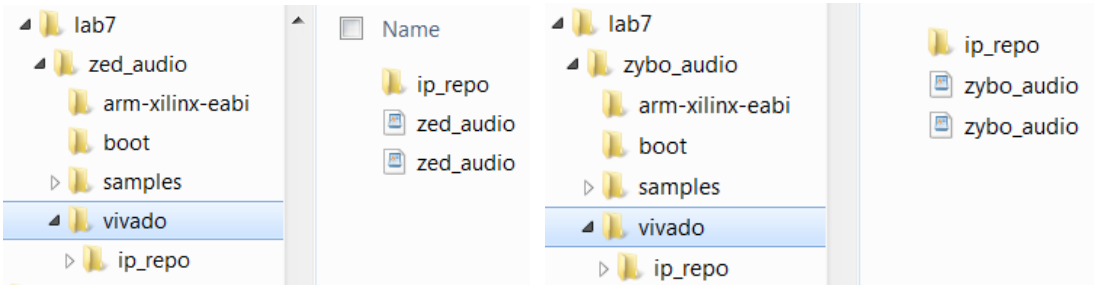
Create a Custom Platform Step 1

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 SDSoC 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 SDSoC platform

```
<?xml version="1.0" encoding="UTF-8"?>
<xd:repository xd:vendor="xilinx.com" xd:library="xd" xd:name="zybo_audio"
xd:version="1.0" xmlns:xd="http://www.xilinx.com/xd" >

    <xd:description>Platform targeting the ZYBO board for an audio applicat
ion. More information at http://www.digilentinc.com/</xd:description>

    <xd:libraryFiles
      xd:os="standalone"
      xd:libName="xil"
      xd:bspconfig="arm-xilinx-eabi/system.mss"
      xd:includeDir="arm-xilinx-eabi/include"
      xd:ldscript="arm-xilinx-eabi/lscript.ld"
    />

    <xd:bootFiles
      xd:os="standalone"
      xd:bif="boot/standalone.bif"
      xd:readme="boot/generic.readme"
    />

</xd:repository>
```

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

1-1-2. Open Vivado by selecting **Start > All Programs > Xilinx Design Tools > SDSoC 2015.4 > Vivado Design Suite > Vivado 2015.4**

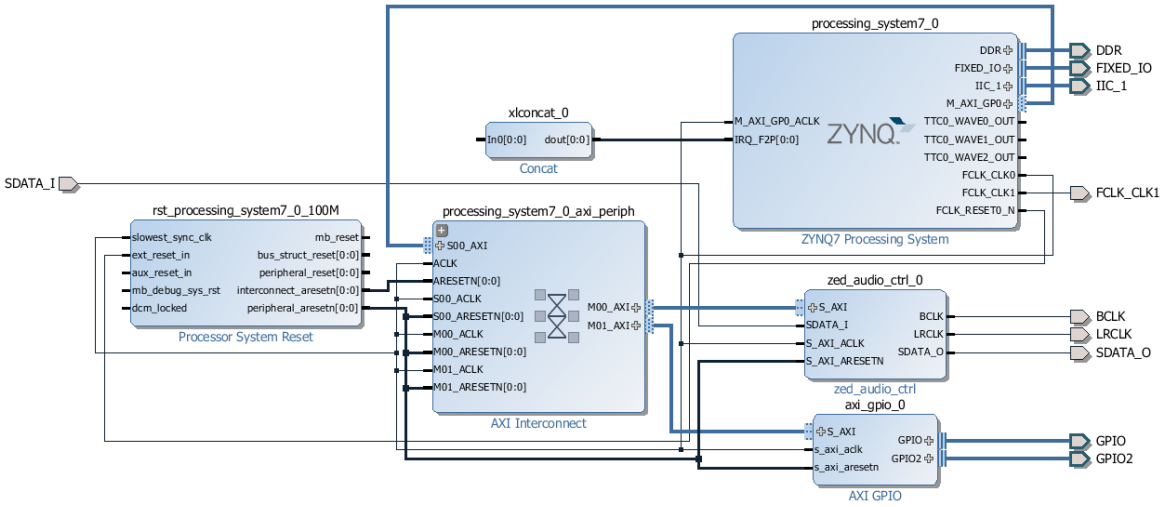
1-1-3. In the Vivado's Tcl Console window change the directory to the *c:/xup/SDSoC/labs/lab7/zed_audio/vivado/* or *c:/xup/SDSoC/labs/lab7/zybo_audio/vivado/* using the **cd** command.

```
cd c:/xup/SDSoC/labs/lab7/zed_audio/vivado (for Zed) or
cd c:/xup/SDSoC/labs/lab7/zybo_audio/vivado (for Zybo)
```

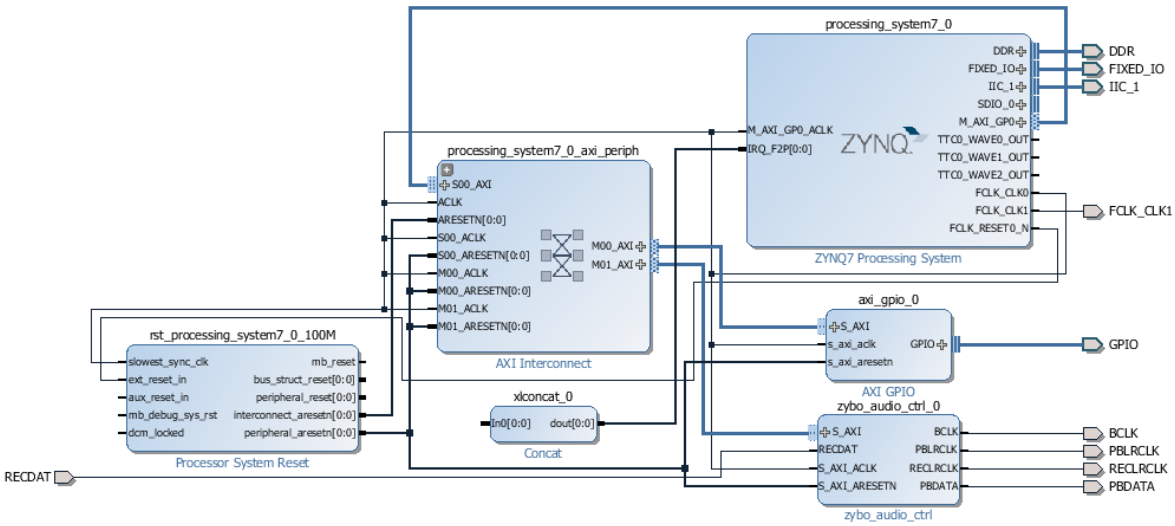
1-1-4. Execute the following command to generate the platform hardware.

```
source ./zed_audio.tcl    (for Zed) or
source ./zybo_audio.tcl   (for Zybo)
```

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.

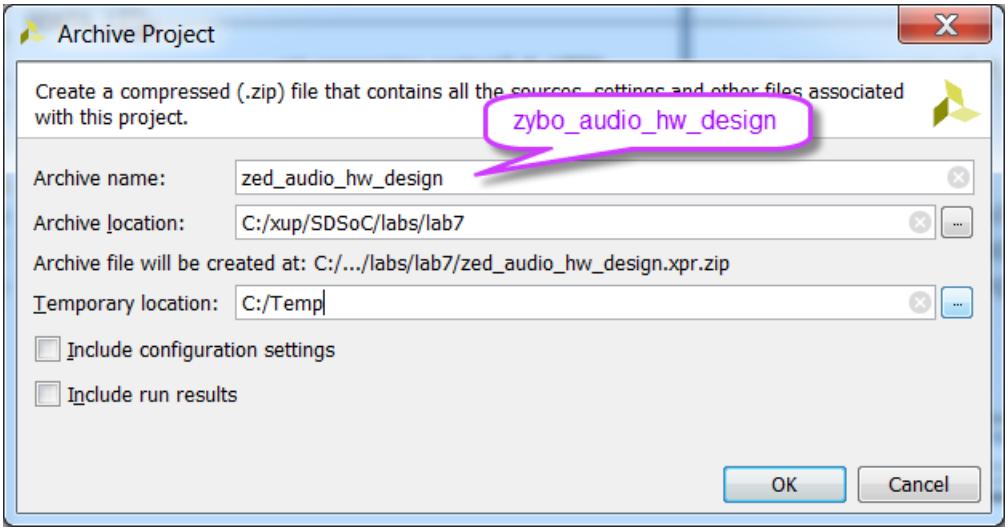


Figure 4. Archiving the project

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 files and directories under the *c:\xup\SDSoC\labs\lab7\zed_audio\vivado* or *c:\xup\SDSoC\labs\lab7\zybo_audio\vivado* directory.

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

1-2-4. Keep only the *zybo_audio.ipdefs, zybo_audio.srcs directories,* and the *archive_project_summary.txt* and *zybo_audio.xpr* files, and delete all other files and directories.

1-3. Generate the hardware description for the board.

1-3-1. In Vivado, open the *zed_audio* or *zybo_audio* project and open the block design.

Note that the block design must be open to execute the following steps to generate the hardware description.

1-3-2. Source the SDSoC platform creation tcl script by executing the following command:

```
source -notrace c:/Xilinx/SDSoC/2015.4/scripts/vivado/sdsoc_pfm.tcl

set pfm [sdsoc::create_pfm zed_audio_hw.pfm]
```

or

```
set pfm [sdsoc::create_pfm zybo_audio_hw.pfm]
```

Replace `c:/Xilinx/SDSoC/2015.4` with the location of your installation if necessary.

The second command will start the creation of a platform called *zed_audio_hw* or *zybo_audio_hw*.

1-3-3. Declare a brief platform description:

```
sdsoc::pfm_name $pfm "xilinx.com" "xd" "zed/zybo_audio" "1.0"

sdsoc::pfm_description $pfm "Zynq ZedBoard/Zybo with audio codec"
```

1-3-4. Define the main clock which will be used as the default clock along with any other clock domains that SDSoC may use while generating accelerators.

```
sdsoc::pfm_clock $pfm FCLK_CLK0 processing_system7_0 0 true
rst_processing_system7_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 SDSoC, unless the user selects a different clock, for example, in the case of a multi-clock design.

1-3-5. Declare the platform AXI bus interfaces by executing the following commands:

```
sdsoc::pfm_axi_port $pfm M02_AXI processing_system7_0_axi_periph
M_AXI_GP

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

In this section you define the AXI ports which you want SDSoC 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 share M_AXI_GP using the next available port on the AXI Interconnect.

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

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

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

1-3-6. 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 SDSoC. 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 SDSoC interrupts will start at 2. *i* should be set to 2 in the above command:.

1-3-7. Generate the platform hardware description metadata file by executing the following command:

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

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

1-3-8. Using the Windows Explorer, copy the ***.pfm** file to the *c:\xup\SDSoC\labs\lab7\zed_audio* or *\zybo_audio* directory.

1-4. Export the vivado project and generate the software description.

1-4-1. In Vivado, export the hardware by selecting **File > Export > Export Hardware**

The *Generate Output Products* dialog box will appear.

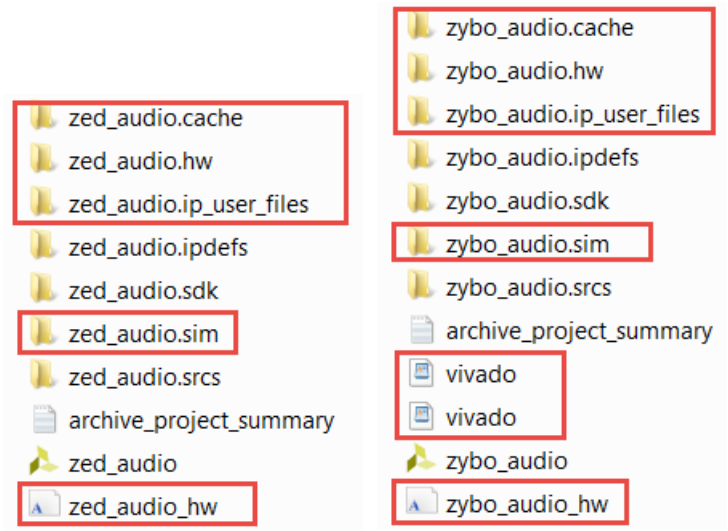
1-4-2. Click on the **Generate Output Products** button.

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.

1-4-3. Click **OK** to close the critical warning window if displayed.

1-4-4. Using the Windows Explorer, delete the files and directories shown in the red boxes below.



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

Figure 5. Directory content

1-4-5. Open SDSoC by selecting **Start > All Programs > Xilinx Design Tools > SDSoC 2015.4 > SDSoC 2015.4**

The workspace dialog box will open.

1-4-6. Click on the **Browse** button, select the *c:\xup\SDSoC\labs\lab7\zybo_audio\vivado\zybo_audio.sdk* directory and click **OK**. Click **OK** again.

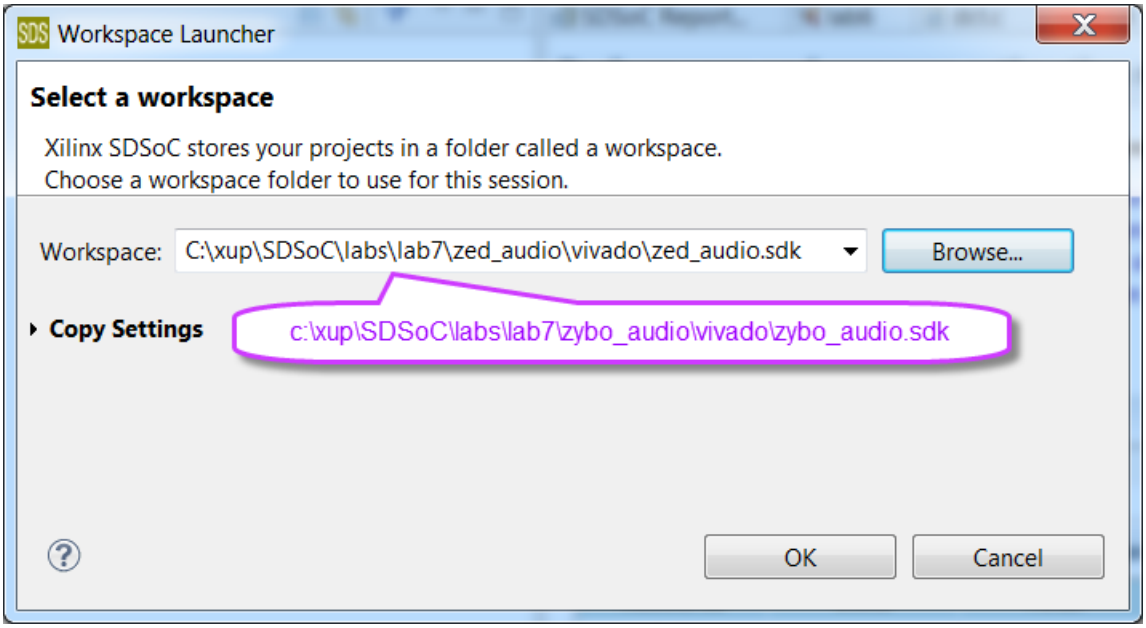


Figure 6. Selecting the workspace

1-4-7. Select **File > New > Projects** and then expand Xilinx and select *Hardware Platform Specification* and click Next.

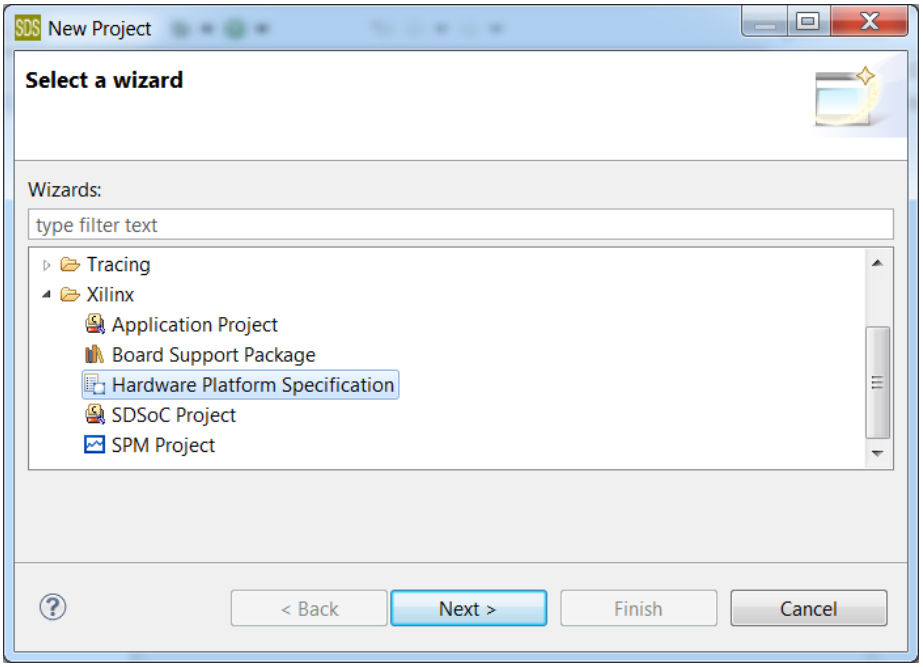


Figure 7. Creating the hardware platform specification project

1-4-8. 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_audio\vivado\zed_audio.sdk or c:\xup\SDSoC\labs\lab7\zybo_audio\vivado\zybo_audio.sdk*, select *zed_audio_wrapper.hdf or zybo_audio_wrapper.hdf*, and click **Finish**.

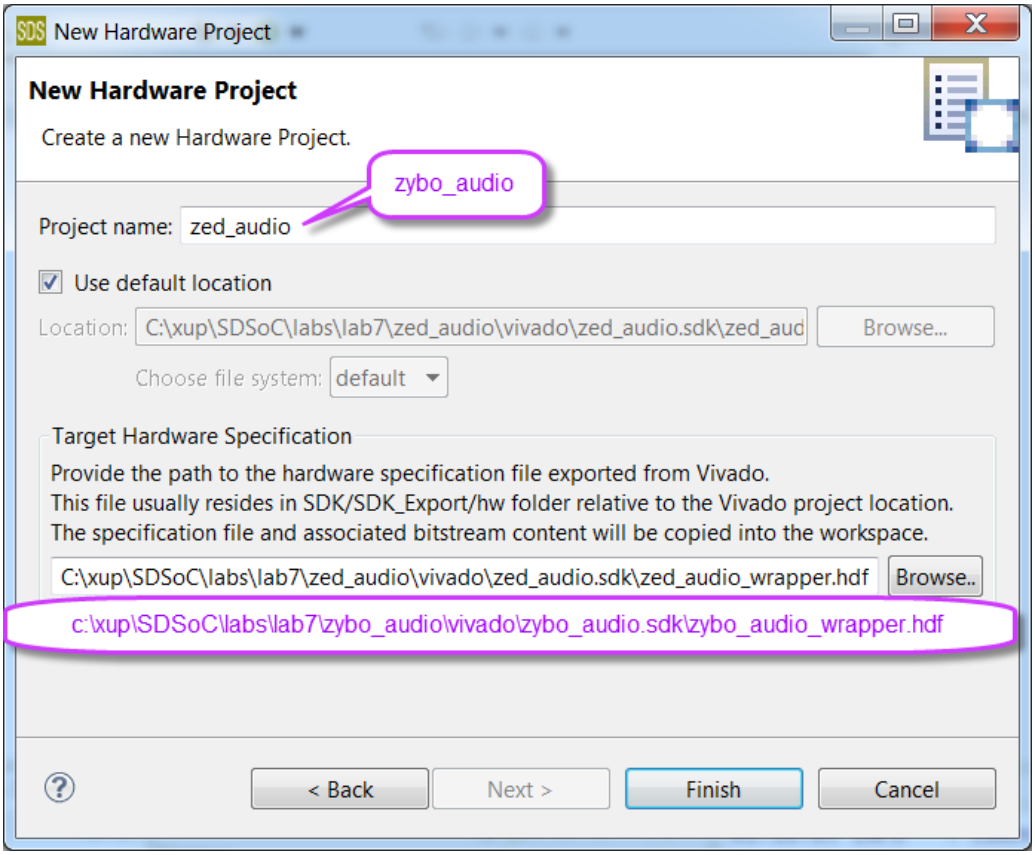


Figure 8. Creating hardware project in SDSoC

1-4-9. Select **File > New > Projects**, then expand **Xilinx** and select **Board Support Package** and click **Next**.

1-4-10. Click **Finish** with *standalone_bsp_0* as the *Project name*, making sure that *zed_audio* or *zybo_audio* is selected as the *Hardware Platform*.

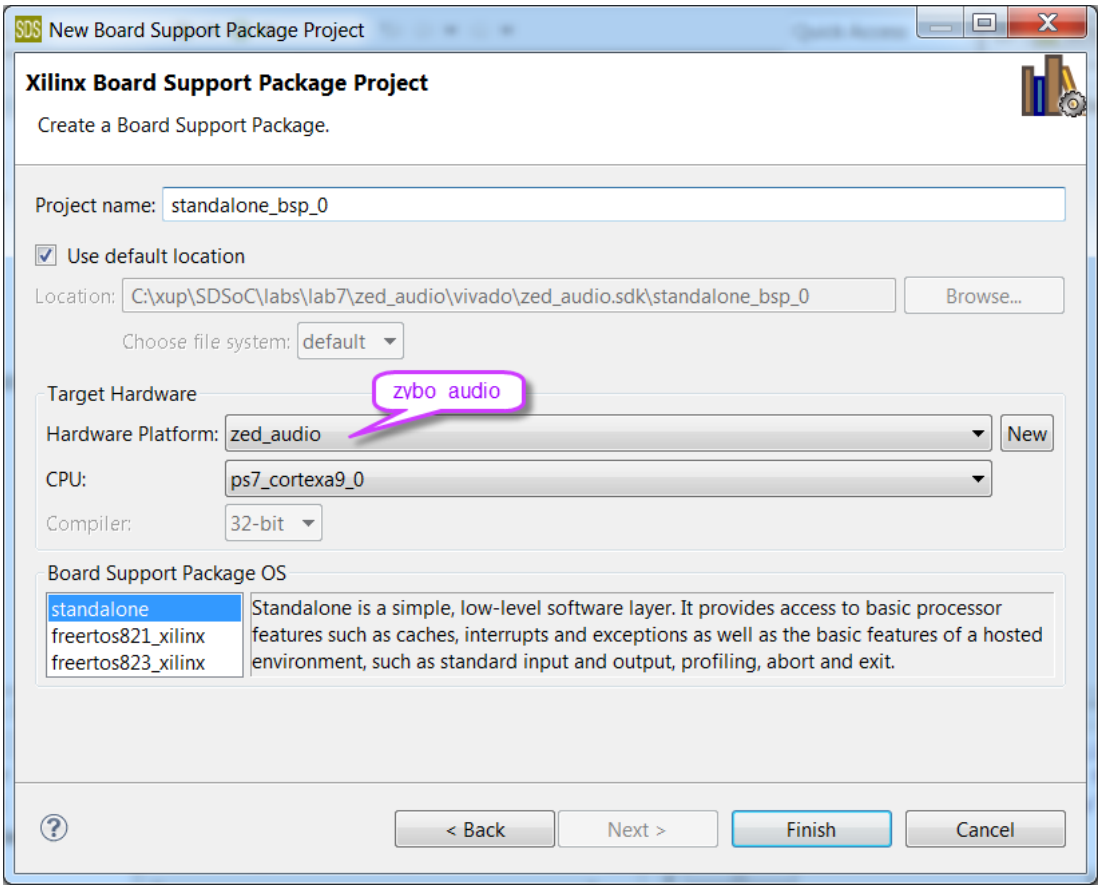


Figure 9. Creating the board support package for the platform

The *Board Support Package Settings* will open.

1-4-11. Select *xilffs* library and click **OK**.

1-4-12. Right-click on the *standalone_bsp_0* and select **build project**.

1-5. Create an Hello World application to generate a linker script.

1-5-1. Select **File > New > Application Project** and then expand Xilinx and select *Hardware Platform Specification* and click **Next**.

1-5-2. Enter **hello_world** in the *Project name* field, select **Use existing BSP** (with standalone_bsp_0 selected) and click **Next**.

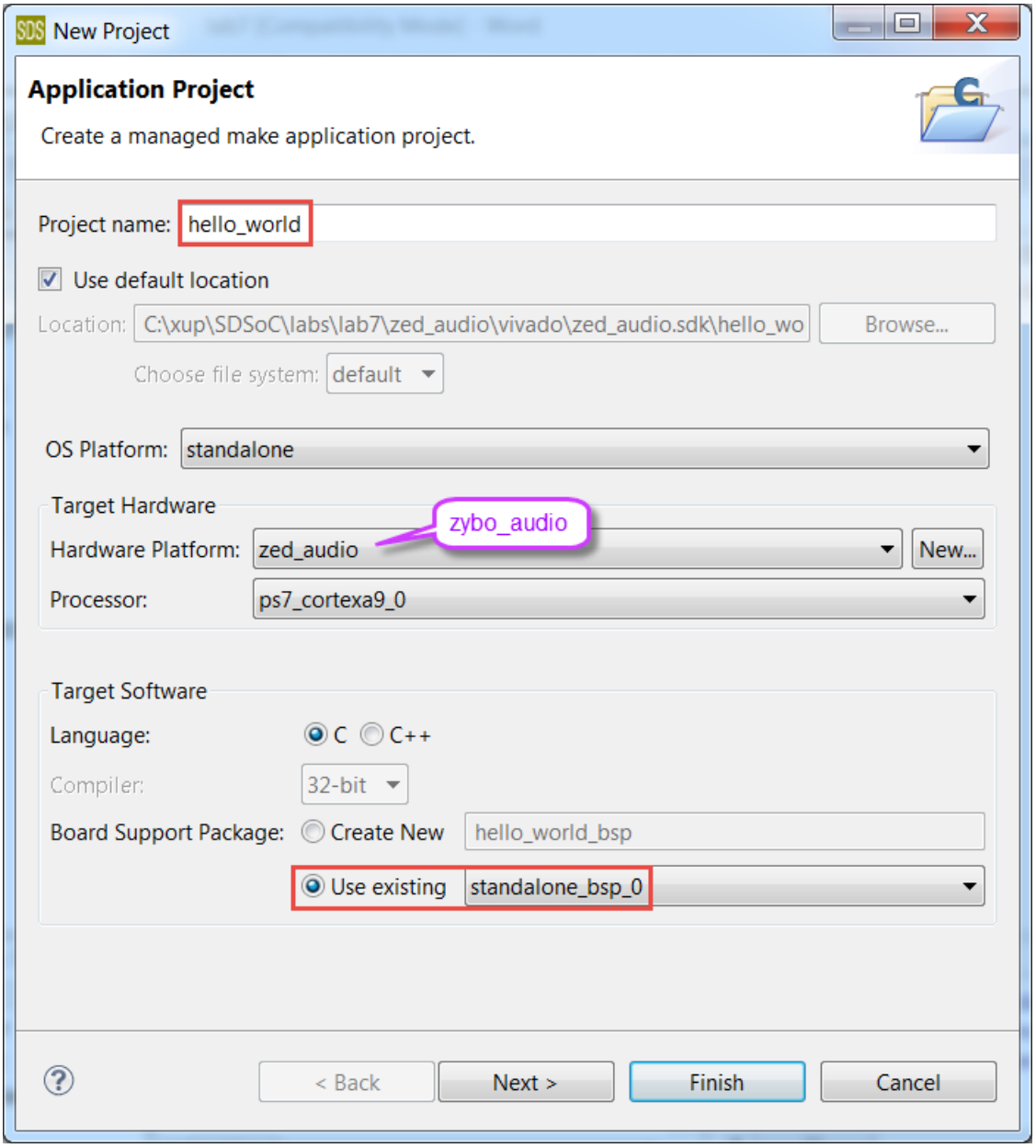


Figure 10. Creating hello world application to generate the linker script

1-5-3. Select **Hello World** from the *Available Templates* pane, and click **Finish**.

1-5-4. Click **Yes** to open the *C/C++ perspective*.

1-5-5. Expand the *src* folder under the **hello_world** project in the Project Explorer pan, and double-click on the **lscript.ld** entry.

Notice that the Stack and Heap sizes are assigned 0x2000 by default. Depending on your application, you may need more *Heap* space if the application is going to transfer a large amount of data using sds_alloc.

1-5-6. Change the *Heap size* to **0x4000** and then press *Ctl+S* to save the changes.

1-6. Generate the FSBL application so the board can be boot from the SD card.

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

1-6-2. Enter **fsbl** in the *Project name* field.

1-6-3. For the Board Support Package, select **Create New** and click **Next.**

1-6-4. Select **Zynq FSBL** from the *Available Templates* pane, and click **Finish.**

1-6-5. Right-click on the *fsbl_bsp* entry and select **build project.**

1-6-6. Right-click on the *fsbl* entry and select **build project.**

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

1-6-7. Using the Windows Explorer, copy the following files and directories to *c:\xup\SDSoC\labs\lab7\zed_audio\arm-xilinx-eabi* or *c:\xup\SDSoC\labs\lab7\zybo_audio\arm-xilinx-eabi*

The **system.mss** file and the **include** directory from *c:\xup\SDSoC\labs\lab7\zed_audio\vivado\zed_audio.sdk\standalone_bsp_0*

OR

c:\xup\SDSoC\labs\lab7\zybo_audio\vivado\zybo_audio.sdk\standalone_bsp_0

The **linker.ld** file from

c:\xup\SDSoC\labs\lab7\zed_audio\vivado\zed_audio.sdk\hello_world\src

OR

c:\xup\SDSoC\labs\lab7\zybo_audio\vivado\zybo_audio.sdk\hello_world\src

Copy the **fsbl.elf** file from *c:\xup\SDSoC\labs\lab7\zed_audio\vivado\zed_audio.sdk\fsbl\Debug* into *c:\xup\SDSoC\labs\lab7\zed_audio\boot*.

OR

The **fsbl.elf** file from *c:\xup\SDSoC\labs\lab7\zybo_audio\vivado\zybo_audio.sdk\fsbl\Debug* into *c:\xup\SDSoC\labs\lab7\zybo_audio\boot*.

Test the Built Platform — Step 2

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

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

2-1-3. Click on the *Create SDSoC Project* in the Welcome tab or select **File > New > SDSoC Project**

2-1-4. Enter **audio_test** in the *Project name* field

2-1-5. For the *Platform,* click on the **Other…** button, browse to *c:\xup\SDSoC\labs\lab7* and select either **zed_audio** or **zybo_audio**.

2-1-6. For the *OS*, select **Standalone**.

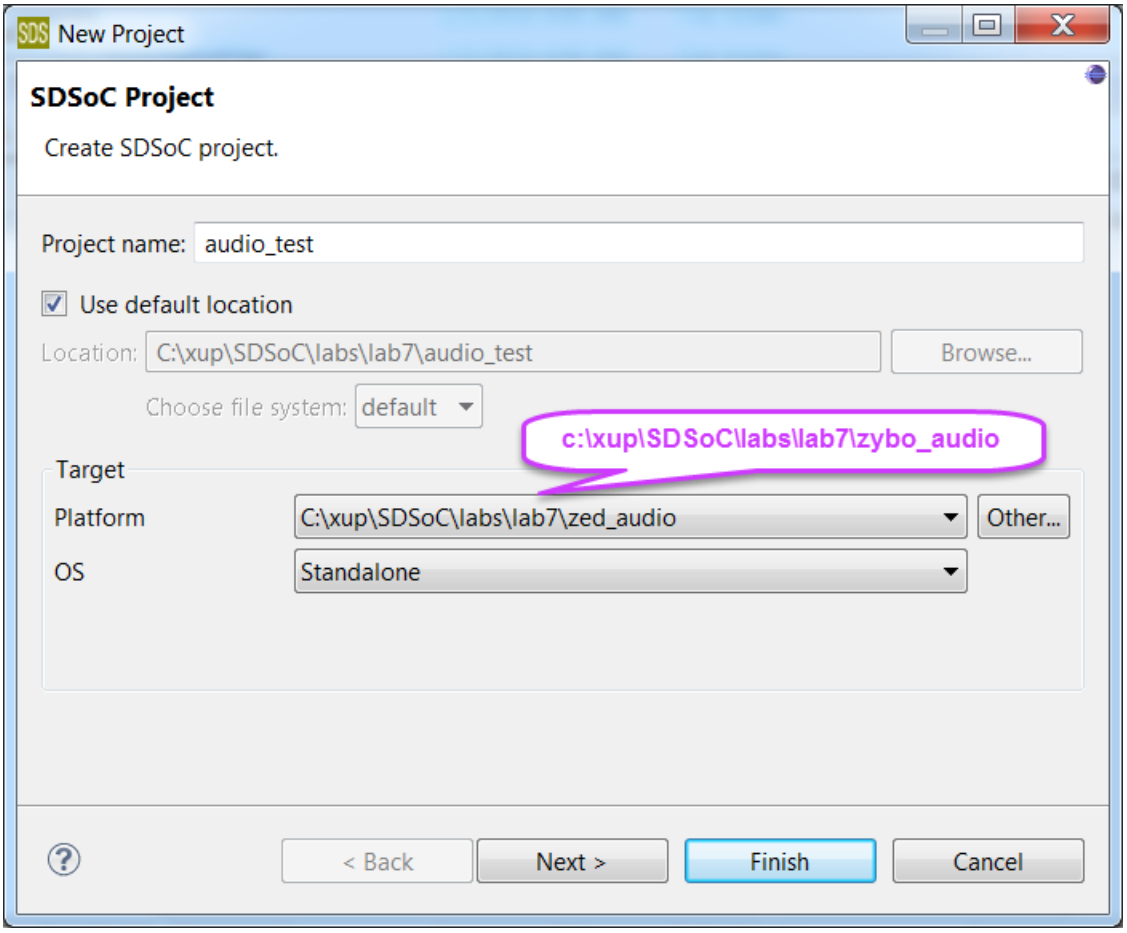


Figure 11. Creating an SDSoC project to test the platform

2-1-7. 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.

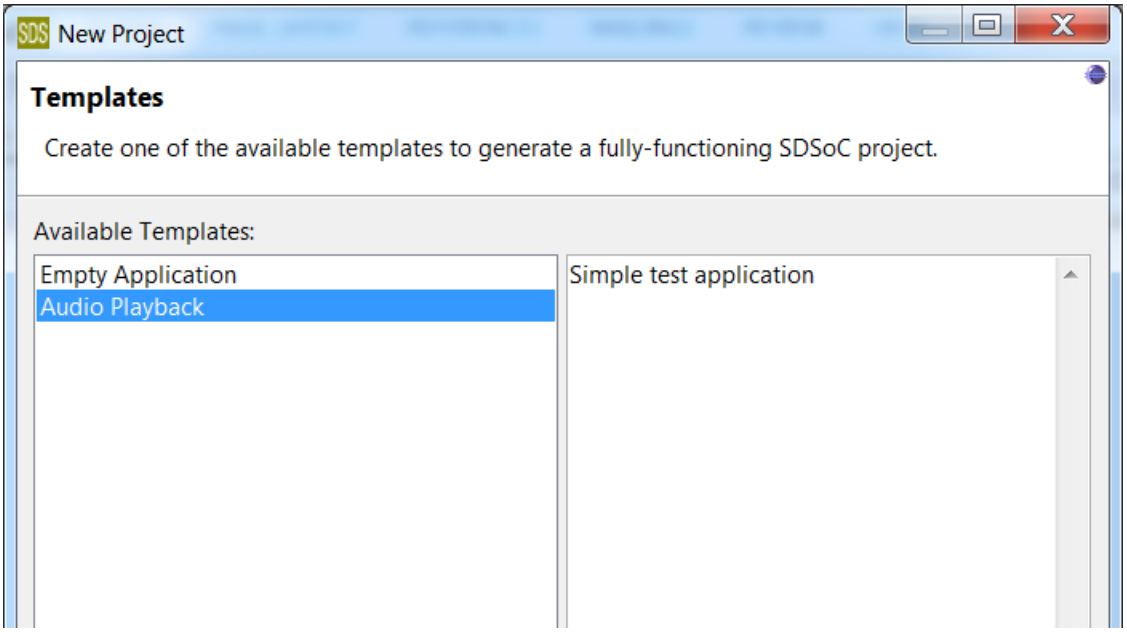


Figure 12. Selecting a test template

2-1-8. Select the *Audio Playback* application and click **Finish**.

2-1-9. 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.

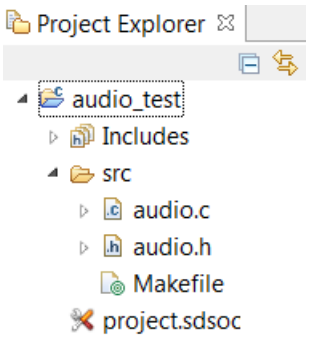


Figure 13. Test application directory

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

2-1-10. 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 (SDDebug)** 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 an 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 > SDSoC Project**

3-1-2. Enter **fir_test** as the project name and click on the *Other…* button of the *Target Platform*.

3-1-3. Browse to either *c:\xup\SDSoC\labs\lab7\zed_audio* or *c:\xup\SDSoC\labs\lab7\zybo_audio* and click **OK**.

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 *Hardware Function* panel.

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\SDDebug\sd_card* into the SD card.

3-2. Configure the board for the SD card boot and connect the board. Test the application.

3-2-1. Place the SD card into the board and configure the board for the SD card boot.

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

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

3-2-4. Connect the board and power it ON.

3-2-5. Open a terminal window and press *PS_SRST* button on the board.

You should see the impulse response in the terminal window.

3-2-6. Play some music on the PC and you should be able to hear the same on your headphones.

3-2-7. When satisfied, turn OFF the board and exit the SDSoC program.

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.