

# Configuring Xilinx SDSoC for PetaLinux Based Platforms

Tools:	2019.1
Training Version:	v3.1
Date:	26 November 2019

© 2019 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

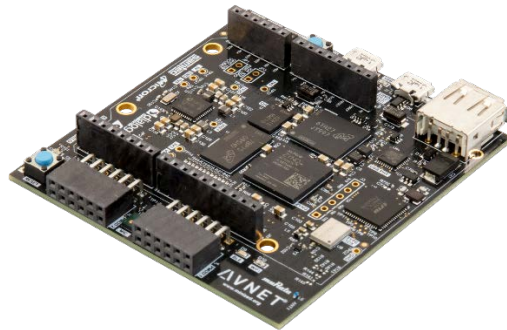
NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

## Introduction

Using the instructions contained herein, you shall learn how to install custom platforms as well as what an output of SDSoC is using the provided example project. Using this base knowledge, you will be able to better understand the needs of SDSoC while leveraging this platform for your own projects. There are some places where we will accept some acceleration (use of Board Presets), however there is no reason that you would be required to do such. For instance, if you were generating the SDSoC platform for a custom board.

## Designed by Avnet

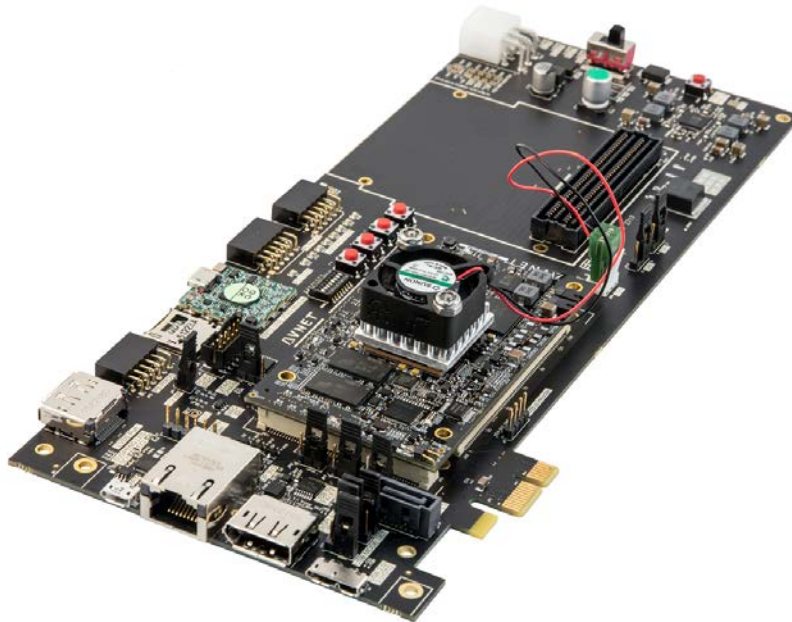
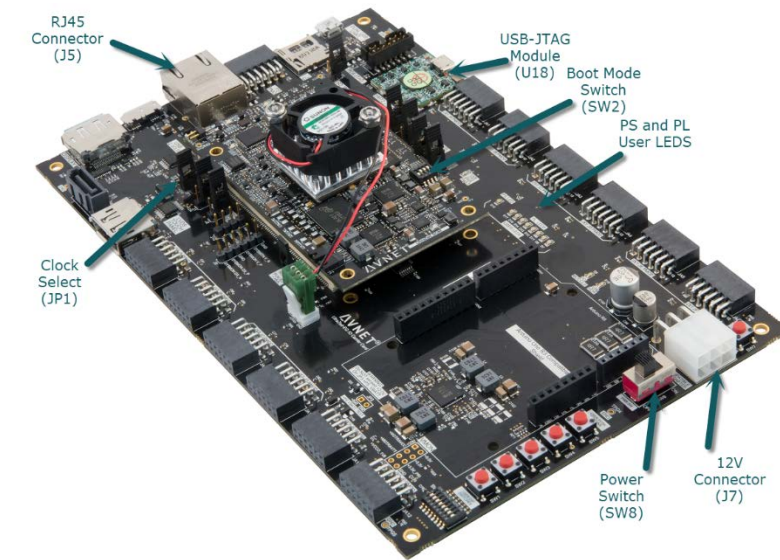
MiniZed™ is a Zynq® 7Z007S single-core development board. With the advent of the latest cost-optimized portfolio from Xilinx, this board targets entry-level Zynq developers with a low-cost prototyping platform.



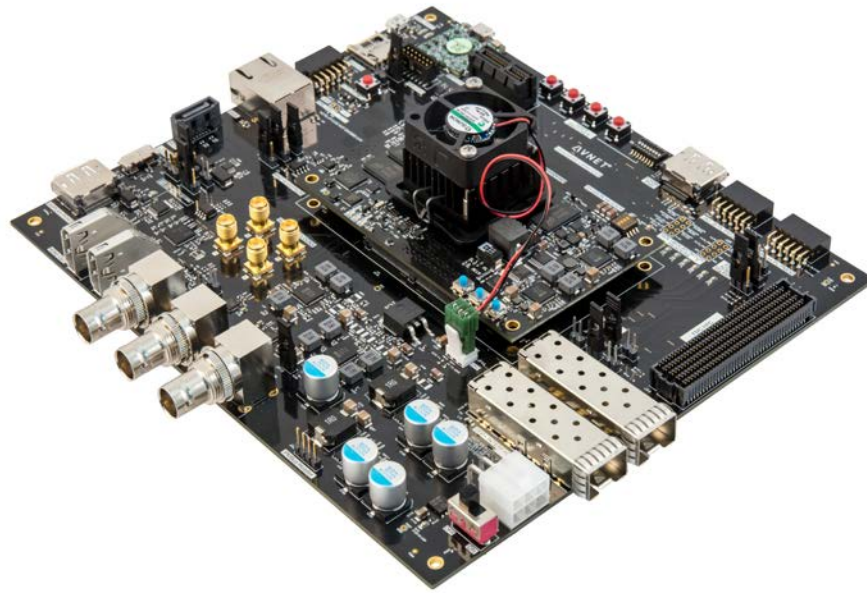
Ultra96 is the first 96boards development board with 64bit ARM and programmable logic. Using the Zynq UltraScale+™ MPSoC XCZU3EG multi-core SoC with accelerators, this makes a perfect platform for starting out with highly complex SDSoC applications. This board targets entry-level Zynq UltraScale+ developers with a low-cost 96boards compatible prototyping platform.



UltraZed-EG SOM is a highly integrated System-on-Module (SOM) based on the powerful Xilinx Zynq UltraScale+ MPSoC family of devices. Designed in a small form factor, the UltraZed-EG SOM packages all the necessary functions such as system memory, Ethernet, USB, and configuration memory needed for an embedded processing system. While this SOM shares the same family as the Ultra96v1 and Ultra96v2, the Xilinx XCZU3EG-1SFVA625 device provides many optimizations and breakout capabilities that are not available on the much smaller package contained in the Ultra96.



UltraZed-EV is a Zynq UltraScale+ MPSoC multi-core SOM which is centered around the Xilinx XCZU7EV. This high performance, full featured SOM mates with the Avnet UltraZed-EV carrier card, which breaks out the FBVB900 package to connect to many transceivers (PS and PL), many video standards, GigE, SATA 3.0, USB 2.0/3.0, PCIe Gen 2 Root Complex, as well as a FMC-HPC allowing access to the PCIe Gen 3 core through the PL interfaces. Being the MPSoC is a 7EV, this means this SOM also include the new H.264/H.265 video codec. This Video Codec Unit (VCU), can do simultaneous 4K2K encode and decode up to 60FPS! This board targets high performance-level Zynq MPSoC developers with a full featured media prototyping platform.



Please contact your local Avnet FAE for further details with any of these kits.

## Lab 1 Design Objectives

Lab 1 offers system developers an example of how to:

- Work through and become familiar with the SDSoC 2019.1 tool flow, from a designer's perspective
- Demonstrate a Matrix Multiply Example output on one of the development kits listed using a pre-built SDSoC platform
- Learn how to install custom SDSoC Platforms

In order to install a custom platform, you will first need to generate the platform. In this case, Xilinx and Avnet has multiple resources available. The main reference you should use will be the Xilinx User Guide 1146. The SDSoC Environment Platform Development Guide v2019.1 document has all the proper references to the details one would need to create their own platform. Avnet and Xilinx also both offer trainings. Avnet has a training guide which steps the user through creation of a custom platform for the MiniZed development board. This SpeedWay is called "A Practical Guide to Getting Started with Xilinx SDSoC". In the case of this specific release, we will not dive into creating the PetaLinux based SDSoC design as the complexities of building with PetaLinux warrant a much more detailed guide and the goal of this guide is familiarity with the tools and working with existing designs.

## Example Design Requirements

### Software

The software used to test this reference design is:

- Xilinx SDx / SDSoC 2019.1 (SDSoC License Required)
- Platform archive
- MiniZed, Ultra96v1, Ultra96v2, UltraZed-EG with Carrier, or UltraZed-EV Board Definition for Vivado



## Hardware

The hardware setup used to test this reference design includes:

- Lenovo ThinkPad T420 Laptop
  - Intel® Core i5-2540M CPU - 2.60 GHz
  - 4GB DDR3 Memory
  - SD card slot on PC or external USB-based SD card reader
- Avnet MiniZed (AES-MINIZED-7Z007-G)
  - Or
- Avnet Ultra96v1 (AES-ULTRA96-G)
  - JTAG Pod (AES-ACC-U96-JTAG)
  - Or
- Avnet Ultra96v2 (PN AES-ULTRA96-V2-G)
  - JTAG Pod (AES-ACC-U96-JTAG)
  - Or
- Avnet UltraZed-EG Starter Kit (AES-ZU3EG-1-SK-G)
  - Or
- Avnet UltraZed-EG SOM with PCIECC (AES-ZU3EG-1-SOM-G and AES-ZU-PCIECC-G)
  - Or
- Avnet UltraZed-EV Starter Kit (AES-ZU7EV-1-SK-G)
- 1 - USB cable (Type A to Micro-USB Type B)

## Experiment Set Up

As there is greater support for Linux based build environments, this guide is being transitioned to Linux. It is recommended that the user use a native Linux build environment. If this is not possible, it is strongly suggested to use the Avnet VirtualBox Installation Guide to create an Ubuntu build environment. You must have installed the Xilinx tools and properly licensed them. The Board Definition files should be installed into both your SDx based Vivado installation.

- The installation guide is located on the Element 14 Zedboard Community site (<http://avnet.me/vbox-install-guide>).
- Instructions for installing board definitions are located on the Element 14 Zedboard Community Site (<http://avnet.me/InstallBDF>).

**NOTE** the BDF install procedure is the same for all Avnet boards and tool versions from at least 2017.4+ including the most recent tool versions of Vivado

You will also need an unzip tool.

## Experiment 1: Install the Pre-Built SDx Hardware Platform

(UZ3EG\_IOCC, UZ3EG\_PCIEC, UZ7EV\_EVCC)

SDSoC comes pre-installed with many platforms that allow you to immediately use many Xilinx boards. The listed development boards are not included. For this experiment, you will use pre-built hardware platforms called **<developmentBoardShortName>** so that you can quickly begin using SDx.

Installation of an SDx hardware platform is a two-step process:

- Copy the hardware platform files to a repository. For our purposes today, we will use ~/platforms as our repository location.
- Setup SDx to use a repository, pointed at the repository location

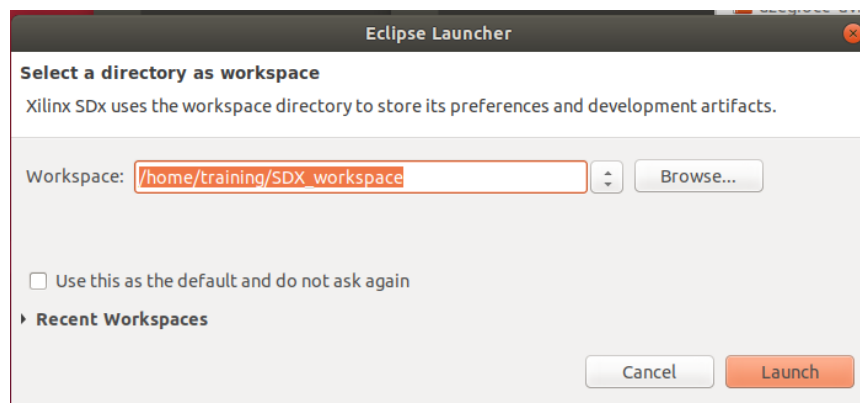
We'll start this experiment by creating a new project so that you can see the pre-installed platforms. Then you will add the Avnet platforms.

1. Having completed the VirtualBox Install Guide steps, Launch the SDx IDE by double clicking the SDx Desktop Icon



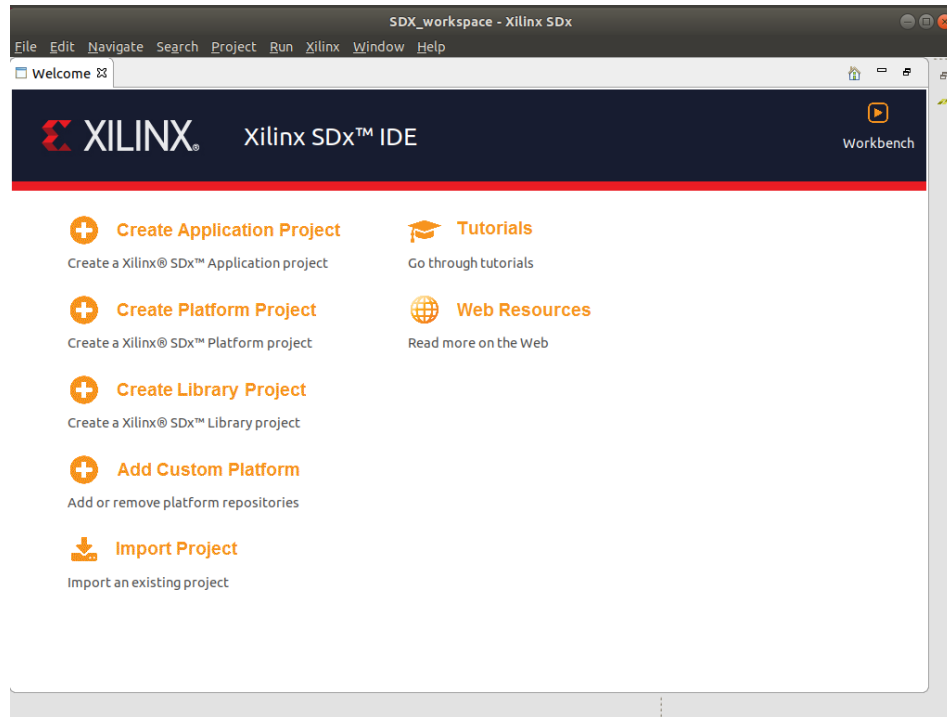
2. SDSoc requires a workspace. Due to Windows path length constraints. It is important that this path be short. Please enter this specific path for consistency through these labs and then click **Launch**.

**`/home/training/SDx_workspace`**

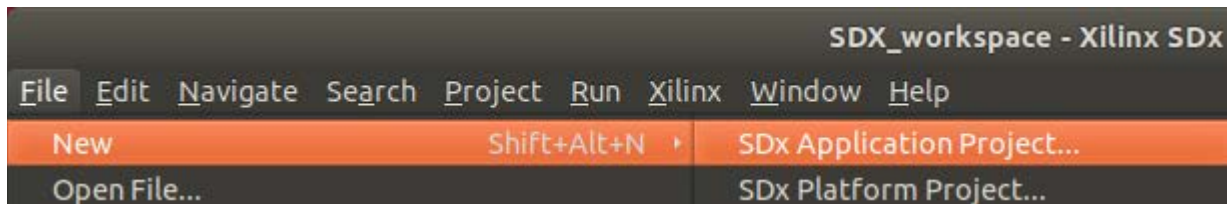




3. The SDx IDE will launch and validate your license. You should see the Welcome dashboard as shown below.



4. Begin by creating a new Xilinx SDx project
  - a. Select **File** → **New** → **SDx Application Project...** or click **Create Application Project** on Welcome screen

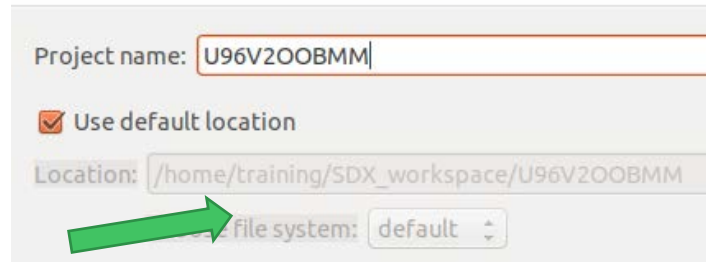


5. For Project Name, enter **UZEGIOCCMM**, **UZEGPCIECCMM** or **UZEVMM**

b. NOTE: the location of the project

#### Create a New SDx Application Project

Enter a name for your SDx Application project.



Project name: U96V2OOBMM












☒ Use default location

Location: /home/training/SDX\_workspace/U96V2OOBMM

file system: default

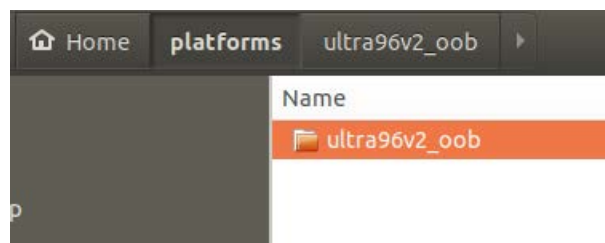
6. Click **Next >**

Here you will see the list of pre-installed Hardware Platforms. These are all Zynq-based boards. Notice that our platform is not in the list.

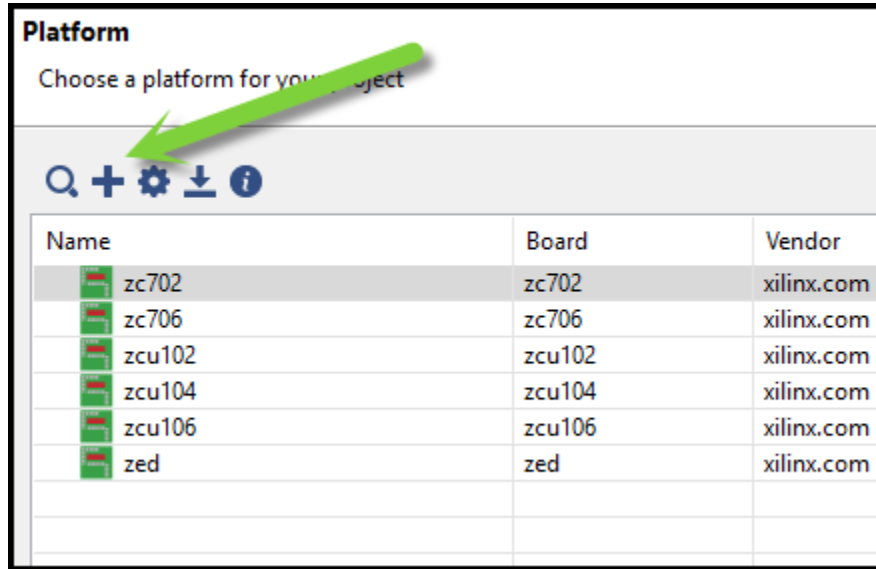
Platform		
Choose a platform for your project		
    		
Name	Board	Vendor
 zc702	zc702	xilinx.com
 zc706	zc706	xilinx.com
 zcu102	zcu102	xilinx.com
 zcu104	zcu104	xilinx.com
 zcu106	zcu106	xilinx.com
 zed	zed	xilinx.com

7. Open Nautilus file manager (or similar) and create the directory `~/platforms`
8. Now unzip the `ultra96v2_oob_2019p1_PetaLinux_SDSoc_Platform.tar.gz` archive into the folder at `~/platforms`

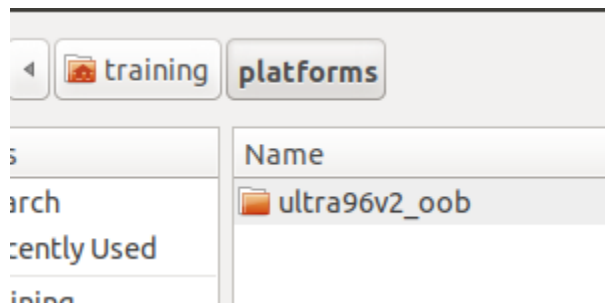
When complete you should have a directory structure as shown below:



9. In the New SDx Application Project dialog, click on **+**



10. In the next dialog, navigate to the ~/platforms folder.



11. Click **OK**.

Notice that in the **Choose Hardware Platform** is updated with the custom platforms and noted with a [custom] tag

Name	Board
ultra96v2_oob [custom]	av
zc702	zc702
zc706	zc706
zcu102	zcu102
zcu104	zcu104
zcu106	zcu106
zed	zed

You have now installed one of Avnet's custom platforms. You can select other platforms and extract them the same way!

## Experiment 2: Create the Matrix Multiply Project

1. Now, we can choose the platform you are targeting, here we choose the u96\_avnet platform by selecting **ultra96v2\_oob [custom]** then click **Next >** to continue. You can also choose one of the other Avnet platforms.

### Platform

Choose a platform for your project



Name	Board	Vendor	Path
ultra96v2_oob [custom]	av	em.avnet.co	/home/training/platforms/ultra96v2_oob/ultra96v
zc702	zc702	xilinx.com	\$XILINX_SDX/platforms/zc702/zc702.xpfm
zc706	zc706	xilinx.com	\$XILINX_SDX/platforms/zc706/zc706.xpfm
zcu102	zcu102	xilinx.com	\$XILINX_SDX/platforms/zcu102/zcu102.xpfm
zcu104	zcu104	xilinx.com	\$XILINX_SDX/platforms/zcu104/zcu104.xpfm
zcu106	zcu106	xilinx.com	\$XILINX_SDX/platforms/zcu106/zcu106.xpfm
zed	zed	xilinx.com	\$XILINX_SDX/platforms/zed/zed.xpfm

< Back Next > Cancel Finish

2. You can leave the defaults on the next screen where you need to choose the System Configuration. Notice the System configuration and Domain selection verbiage. This is set when we create a platform (see Appendix A: Quick Steps to Create PetaLinux Platform)

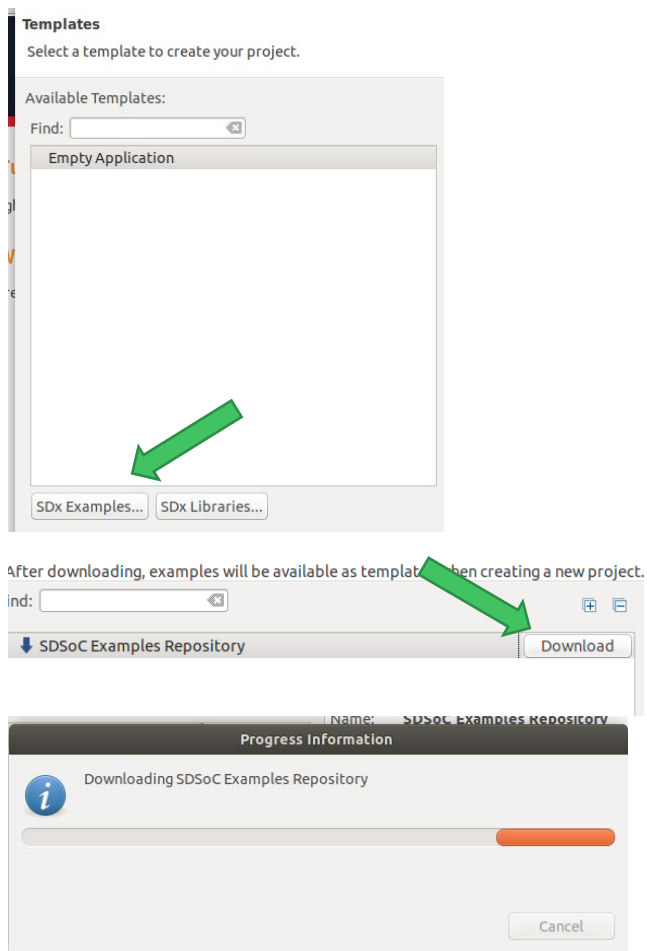
### System Configuration

Provide the system configuration and software details for your project

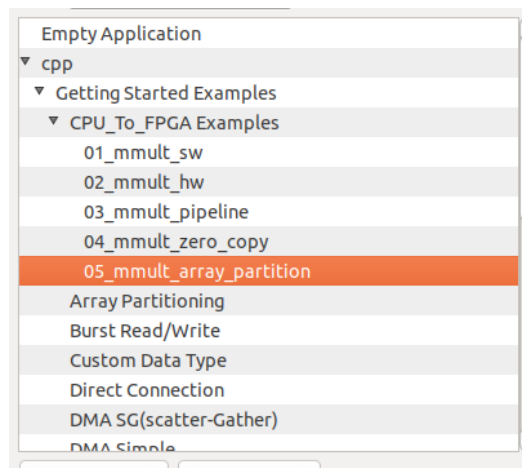
System configuration:	PetaLinux
Runtime:	C/C++
Domain:	PetaLinux 2019p1
CPU:	cortex-a53
Operating System:	linux
<input type="checkbox"/> Sysroot path:	

3. Click **Next >** to continue.

4. Select the “SDx Examples...” button, and install the SDSoC examples, select **OK** when complete

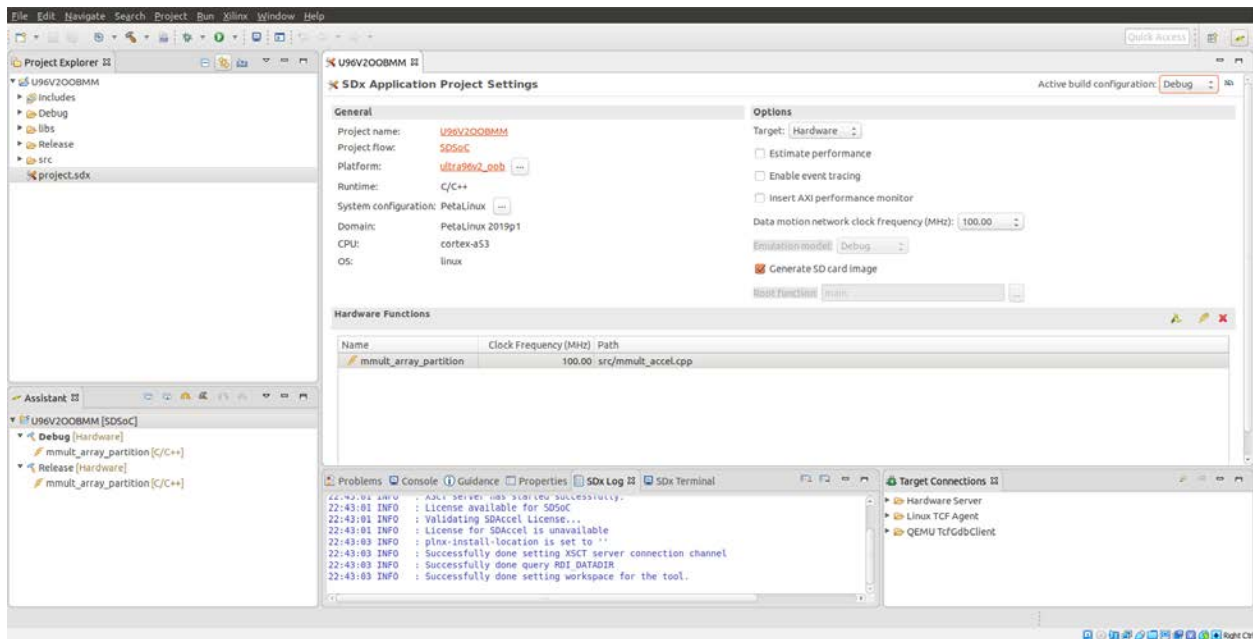


5. Select the 05\_mmult\_array\_partition example, as a matrix multiply is typically seen as the SDSoC Hello World



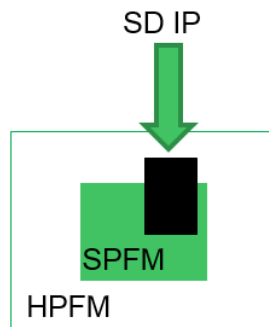
6. Click **Finish** to create a new application.

Now the SDx Project Explorer will have the **ULTRA96V2OOBMM** from which we can generate SDSoC projects.



For the purposes of this experiment, we will use Matrix Multiply Example code.

The example code that you use here is not really relevant as we are interested in the outputs of SDSoC as well as the platform itself. This should be seen as a black box where any accelerator (ex. video processing, LTE engine, AES engine, etc) could be inserted.

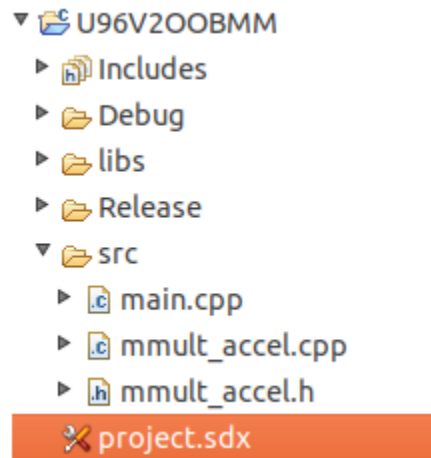


**Target Code is Black Box**

This is one of the most powerful abilities of SDSoC! Since we are using a provided platform, simply imported the example template straight from the platform folder.



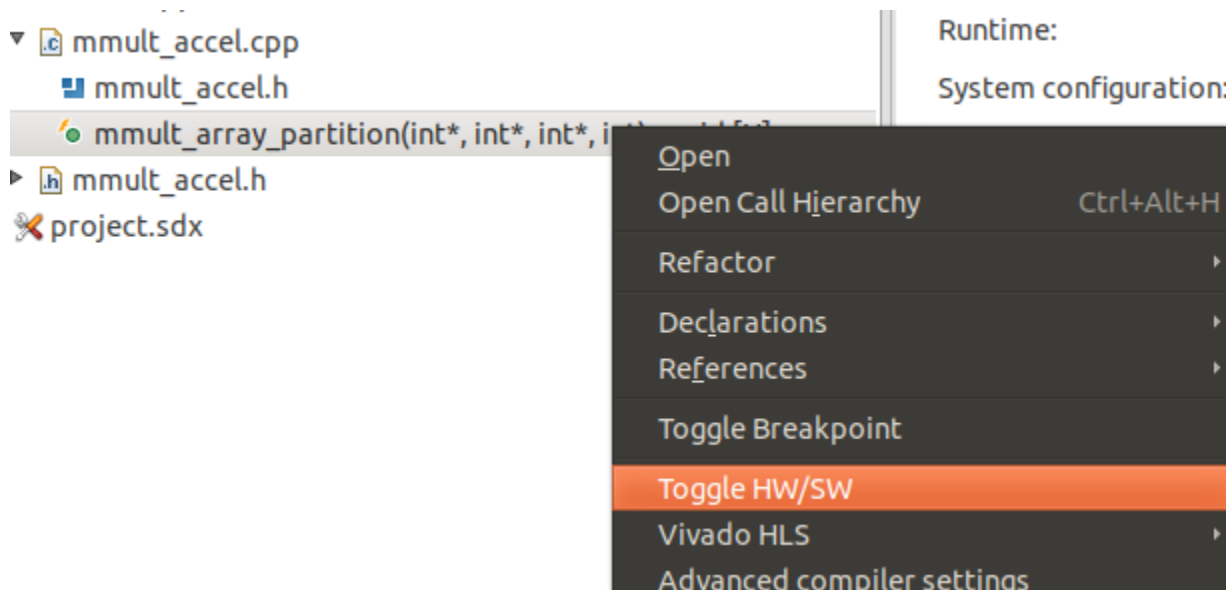
Instructions on how to create your own Sample Template is located in UG1146. If you need to import files, the process is the same as with Xilinx SDK.




Three sources added in Project Explorer

## Experiment 3: Build the Matrix Multiply Project

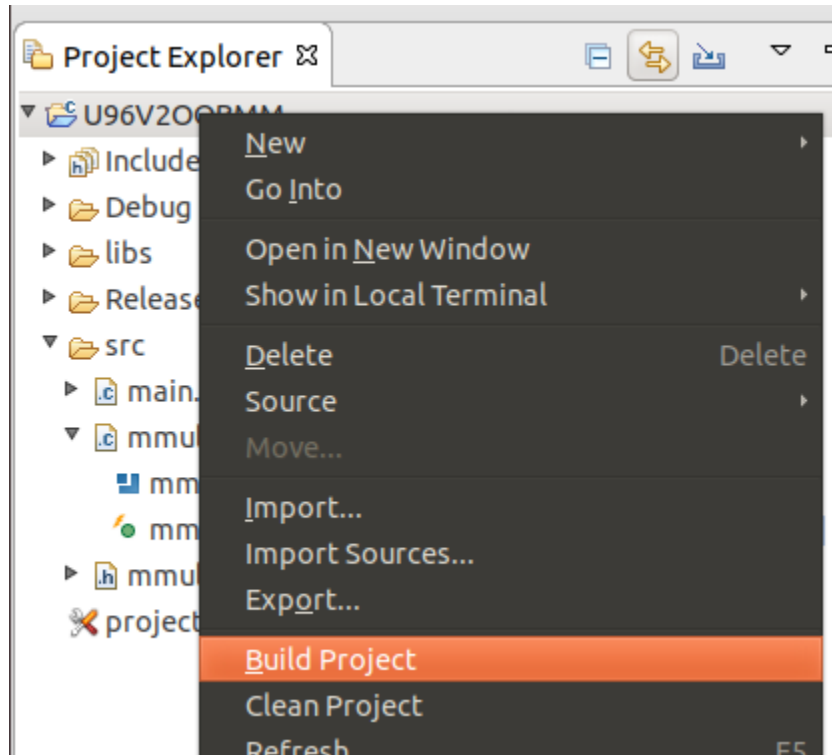
1. As the files were imported for us through the example, we will NOT need to designate which functions to accelerate. If you had the need to, you would expand the mmult\_accel.cpp and right click on the mmult\_array\_partition function. Select Toggle HW/SW



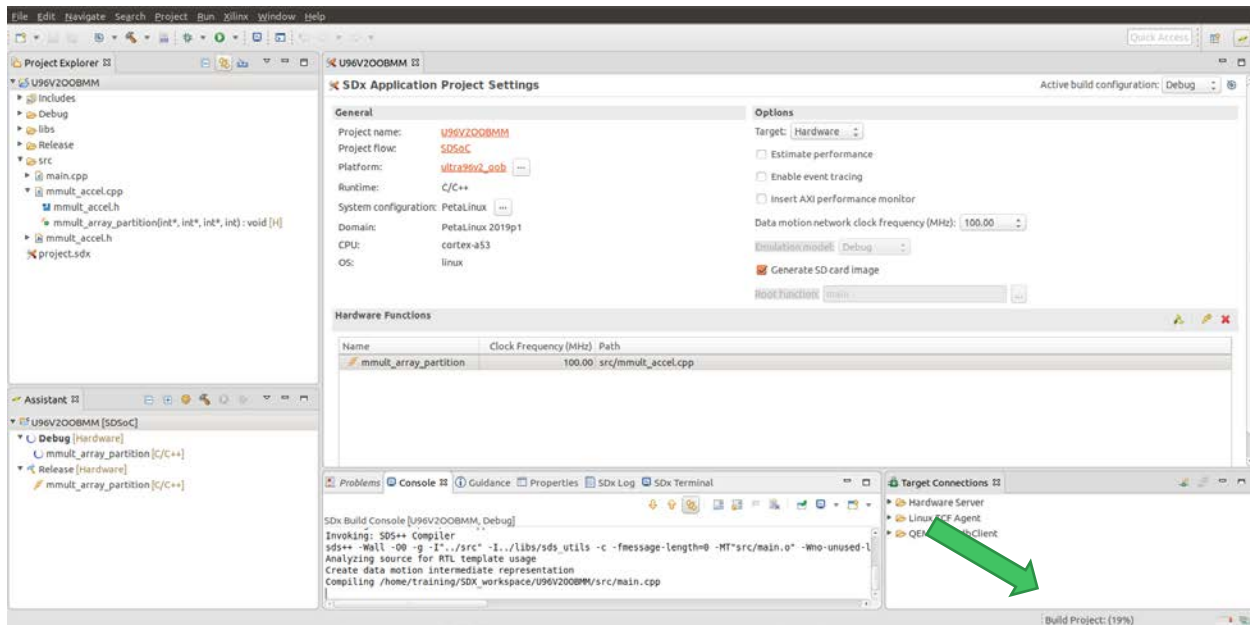
2. Note that the function is already listed under the **Hardware Functions** list

Name	Clock Frequency (MHz)	Path
 mmult_array_partition	100.00	src/mmult_accel.cpp

- Next, back in SDx Project Explorer right click the U96V1MMPL project name, click on **Build Project**, remember, you can follow this same flow with any of the provided platforms



4. While this is building, notice the SDx project Settings. This is a great place to see the overview of the settings that are going into building this project. From here you can see that we HAVE selected the mmult\_array\_partition as a hardware accelerated function.



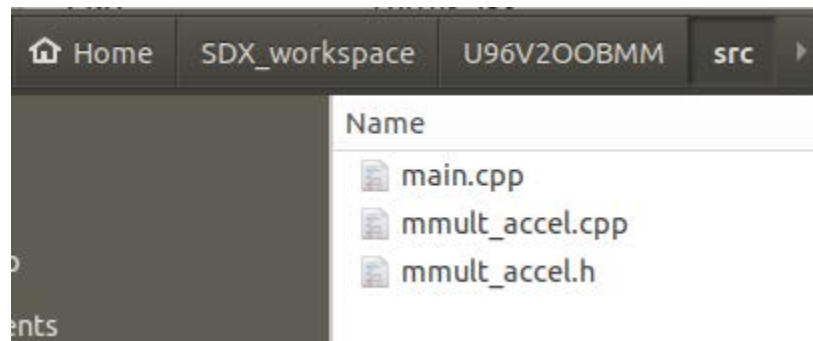
We left the project as Debug and did not select Release. If we had selected Release, the tool would have performed additional runtime optimizations, which would have increased our build time.

The above steps can be seen in more detail in UG1028 – SDSoc Intro Tutorial v2019.1 (now listed on Xilinx GITHUB <https://github.com/Xilinx/SDSoC-Tutorials> ).

While we wait for the build, let's explore. What is happening while this is building?

- The tool copies the Vivado project from the platform into your local build area
- The tool analyzes the provided C code, including pragmas, and builds an internal data motion graph – what connects to what, how, etc. It will make decisions at this stage based on your memory configuration, buffer sizes (if known), etc. to determine interfaces, data movers, etc.
- The C code moving to hardware is synthesized by HLS
- SDSoc updates the BD to incorporate the data motion infrastructure and the generated HLS IPs
- The tool updates your C code to seamlessly call the accelerator instead of the C function (you can see the results of this in the `_sds` directory in the project build area)
- Generate a bitstream
- Combine the bitstream into BOOT.BIN using the FSBL, BIF, etc. that you provide as part of the platform

5. Navigate to `~/SDx_workspace/<project Name>/src`.  
Notice the 3 source files there.  
Note: Swap the project folder name for the one you chose to build.



More notes regarding the files and the file structure

- main.cpp runs the functions
- mmult\_accel.cpp has pragmas listed
- mmult\_accel.h has pragmas listed and defines the data array size

## Experiment 4: Set Up Your SDCARD

While the project is building, you can also set up your SDCARD, if necessary. This is necessary for Ultra96V2. Other platforms Avnet creates utilize a RAM disk for the performance gains. This is a selection made during PetaLinux creation and can be changed with a rebuild of the PetaLinux BSP. Please note that this section **MUST** be completed on a Linux machine as you will be working with the EXT4 file system.

We will be following a summarized set of instructions from the Xilinx Wiki, where you will create a 1GB FAT32 (W95 version) BOOT partition, as well as a EXT4 Root File System Partition

<https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842385/How+to+format+SD+card+for+SD+boot>

1. After inserting your SDCARD into your Linux host, open a terminal
2. Run `sudo fdisk /dev/sd?`
  - a. Replace the ? with the letter assigned to your SDCARD
3. From here, we will assign the main FAT32 partition by typing 'n' and then 'p' to make it primary, for size type '+1G'
4. Now make it bootable by typing 'a'
5. From here, assign the remaining space as the EXT4 partition by again typing 'n', then choosing the defaults for the first and last sector
6. You should now see two partitions, similar to the below image from the WIKI article posted above

```
Command (m for help): p
Disk /dev/sdc: 14.9 GiB, 15931539456 bytes, 31116288 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x72c90224

Device      Boot    Start        End    Sectors    Size Id Type
/dev/sdc1   *           2048    2099199    2097152     1G 83 Linux
/dev/sdc2                2099200  31116287  29017088    13.9G 83 Linux

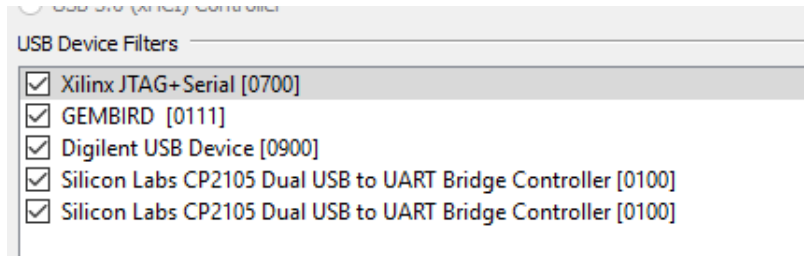
Command (m for help): █
```

7. If you see a similar configuration as the above image, type 'w' to WRITE the partitions and exit
8. Now back at the terminal, you will need to format the two partitions. Run the below commands, but again, replace the ? with the letter assigned to your SDCARD
  - b. `mkfs.vfat -F 32 -n boot /dev/sd?1`
  - c. `mkfs.ext4 -L root /dev/sd?2`

Note if the above has issues running, check that you have changed the ? to the letter assigned to your SDCARD. You might also need to `sudo` the command

## Experiment 5: Run the Design

1. Insert one USB cable into the Ultra96V2 USB JTAG/UART POD MicroUSB connector
2. Plug the other end into your PC
3. If this is the first time you have connected your JTAG, ensure you have it selected to auto-capture into the VirtualBox

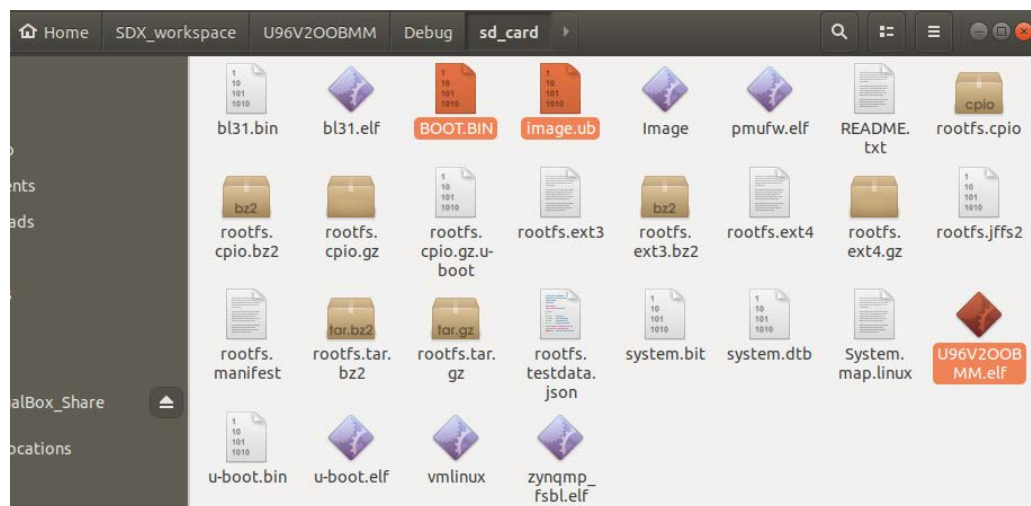


4. If your build is complete, use a file manager to copy the three below selected files from the sd\_card folder to your BOOT partition on the SDCARD. Note the location includes a rootfs.tar.gz file, which is only necessary for operating systems that use a static filesystem.

If you are still building, skip this step and return later. This can be completed at any time prior to booting the Ultra96V2 with this card

The files are located:

~/projects/SDx\_workspace/<project Name>/Debug/sd\_card





5. If you are using a static filesystem (Ultra96V2 for example) follow the following steps, otherwise skip to the next step.

- a. Within your LINUX environment, Open a terminal and cd to the root partition

```
training@training-VirtualBox:~$ cd /media/training/  
boot/ root/  
training@training-VirtualBox:~$ cd /media/training/root/  
training@training-VirtualBox:/media/training/root$
```

- b. From here extract the root file system from the generated rootfs.tar.gz

Note: replace the folder location to point to where you have located the `rootfs.tar.gz` file

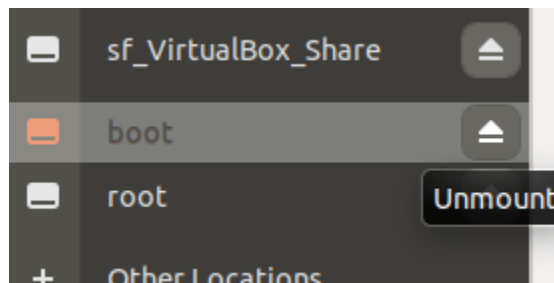
- i. `sudo tar -xvf ~/SDX_workspace/<projectName>/Debug/sd_card/rootfs.tar.gz`

```
x:~$ cd /media/training/root/  
x:/media/training/root$ sudo tar -xvf ~/projects/SDx_workspace/U96V1MM/Debug/sd_card/rootfs.tar.gz
```

- c. To ensure all caches are flushed, type `sync` this command might take some time

```
./etc/iproute2/rt_tables  
./etc/iproute2/group  
./etc/iproute2/nl_protos  
./etc/iproute2/bpf_pinning  
./etc/gshadow  
./dev/  
./proc/  
./mnt/  
training@training-vb:/media/training/root$ sync
```

- d. Once this extraction and sync completes, change out of that folder in preparation for removing the SDCARD from the Linux machine. `cd ~`
6. Once this is complete, eject the drive per your Linux operating system. In the case of the Ubuntu install from the Avnet Virtual Machine Setup Guide, right click the thumbdrive icon and eject parent drive



7. If you have not completed step 4 and come back to step 8, otherwise insert the SDCARD into the SDCARD cage.
8. Turn on the power and you should observe a similar display, as shown below for Ultra96v2

```
ultra96v2-oob-2019-1 login:   
  
/dev/ttyUSB1 115200-8-N-1
```

9. From here, login with `root` as the username and `root` as the password
10. Now in order to access the executable (elf) file, we will need to mount the boot partition of the SDCARD

- a. `cd /media`
- b. `mkdir sdcard`
- c. `mount /dev/mmcblk0p1 ./sdcard/`

NOTE different platforms use different SDCARD locations, for example, UltraZed-EV would use the command  
`mount /dev/mmcblk1p1 ./sdcard`

```
root@ultra96v2-oob-2019-1:~# cd /media
root@ultra96v2-oob-2019-1:/media# mkdir sdcard
root@ultra96v2-oob-2019-1:/media# mount /dev/mmcblk0p1 /media/sdcard/
```

11. Now execute the file, we will change to the SDCARD folder and execute the program

- a. `cd sdcard/`
- b. `./<project Name>.elf`

```
root@ultra96v2-oob-2019-1:/media# cd sdcard/
root@ultra96v2-oob-2019-1:/media/sdcard# ls -al
total 24420
drwxrwx--- 2 root disk      4096 Jan  1  1970 .
drwxr-xr-x 3 root root      4096 Nov 20 07:25 ..
-rwxrwx--- 1 root disk 6463016 Nov 20  2019 BOOT.BIN
-rwxrwx--- 1 root disk  51536 Nov 20  2019 U96V200BMM.elf
-rwxrwx--- 1 root disk 18480840 Nov 20  2019 image.ub
root@ultra96v2-oob-2019-1:/media/sdcard# ./U96V200BMM.elf
```

```
root@xilinx-ultra96-reva-2018_3:/media/sdcard# ./U96V1MMPL.elf
Testing 1024 iterations of 32x32 floating point mmult...
Average number of CPU cycles running mmult in software: 1580895
Average number of CPU cycles running mmult in hardware: 50936
Speed up: 31.0369
TEST PASSED
root@xilinx-ultra96-reva-2018_3:/media/sdcard#
```

12. You have successfully generated an SDSoC program utilizing the provided platform. You can now CLOSE the SDx IDE.

## Appendix A: Quick Steps to Create PetaLinux Platform

(ultra96v2\_oob, UZ3EG\_IOCC, UZ3EG\_PCIEC, UZ7EV\_EVCC)

To create the platform you are using, you will need a set of files. The details of what files you will need are listed in UG1146. A quick list is shown below.

- \*\_fsbl.elf
- u-boot.elf
- bl31.elf (mandatory for Zynq MPSoC, NOT Zynq-7000)
- pmufw.elf
- image.ub
- rootfs.tar.gz
- <developmentBoardShortName>.dsa

To generate these files, Avnet has scripted build flows and has started to include all proper SDSoc hooks into BSP images. This means, from your standpoint – using a Designed by Avnet development kit or product can save you time and effort as we have already created the proper infrastructure for you to work with the hardware. This also means you can concentrate on any system specific changes you might need instead of the entire architecture!

### General Instruction:

As you have already setup the VirtualBox installation (see Lab 1 “Experiment Set Up”), run the below commands from a terminal window and you will pull down all the necessary source files for a build.

```
$ cd ~  
  
$ mkdir projects  
  
$ cd ~/projects  
  
$ git clone https://github.com/Avnet/bdf.git  
  
$ git clone https://github.com/Avnet/hdl.git  
  
$ git clone https://github.com/Avnet/petalinux.git  
  
$ cd hdl ; git checkout uz_petalinux_20191126_171905 ; cd ..  
  
$ cd petalinux ; git checkout uz_petalinux_20191126_171905 ; cd scripts
```

The Avnet scripted build flow is smart enough to be able to detect if the hardware image has been built and as long as the hdl repository is at the same hierarchical level as the petalinux repository, running ONLY the script to generate the PetaLinux BSP is enough to create the entire BSP, including hardware image.

From here, execute one of the SDSoC enabled BSP generation scripts.

Board	Script to execute
Ultra96V2	make_ultra96v2_oob_bsp.sh
UltraZed-EG PCIECC	make_uz_petalinux_bsp.sh*
UltraZed-EG IOCC	make_uz_petalinux_bsp.sh*
UltraZed-EV CC	make_uz_petalinux_bsp.sh*

\*NOTE the UltraZed family uses a combined build script. To ensure you build ONLY the target(s) you want, you will need to modify both make\_uz\_petalinux.tcl as well as make\_uz\_petalinux\_bsp.sh scripts.

Script	Location	Modification
make_uz_petalinux.tcl	hdl/Scripts	Comment OUT the platforms you are NOT building
make_uz_petalinux_bsp.sh	Petalinux/scripts	Change the "Platform build images" variables you do NOT want to build from yes to no

```
$ ./make_<boardProjectName>_bsp.sh
```

1. Once the execution is complete, we will need to generate a BIF file. An example is included in UG1146
2. From the same terminal where the make script was executed, run the below commands
3. Zynq 7000:

```
$ cd ~/projects/petalinux

$ echo -e 'the_ROM_image:\n{\n [bootloader]<mz_avnet/boot/fsbl.elf>\n <bitstream> \n
<elf>\n}' > ./linux.bif
```

Zynq MPSoC:

```
$ cd ~/projects/petalinux

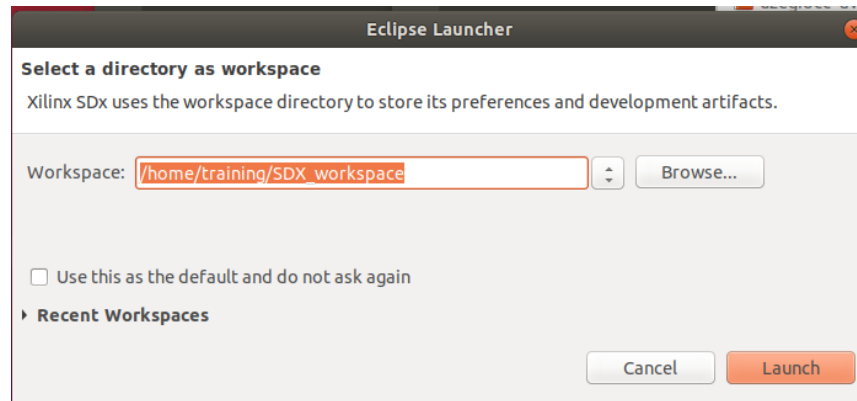
$ echo -e ' the_ROM_image:\n{\n [fsbl_config] a53_x64\n [bootloader]
<a53_linux/boot/zynqmp_fsbl.elf>\n [pmufw_image] <a53_linux/boot/pmufw.elf>\n
[destination_device=pl] <bitstream>\n [destination_cpu=a53-0, exception_level=el-3, trustzone]
<a53_linux/boot/bl31.elf>\n [destination_cpu=a53-0, exception_level=el-2] <a53_linux/boot/u-
boot.elf>\n}' > ./linux.bif
```

4. If SDSoC is NOT open, and having completed the VirtualBox Install Guide steps, Launch the SDx IDE by double clicking the SDx Desktop Icon

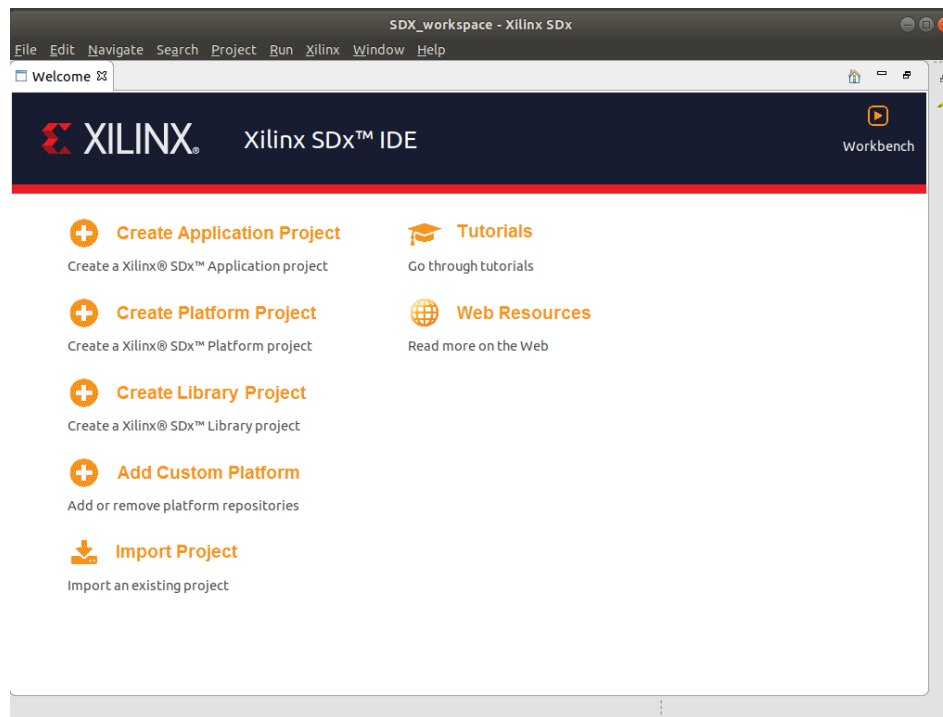


- SDSoC requires a workspace. Please enter this specific path for consistency through these labs and then click **Launch**.

**/home/training/SDx\_workspace**

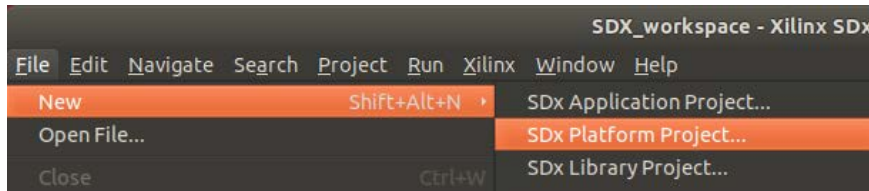


- The SDx IDE will launch and validate your license. You should see the Welcome dashboard as shown below.



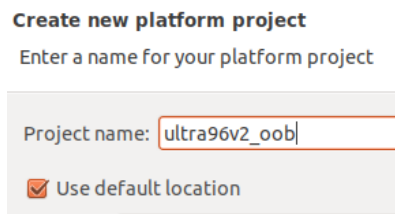
- Begin by creating a new Xilinx SDx project

8. Select **File** → **New** → **SDx Platform Project...** or click **Create Platform Project** on Welcome screen



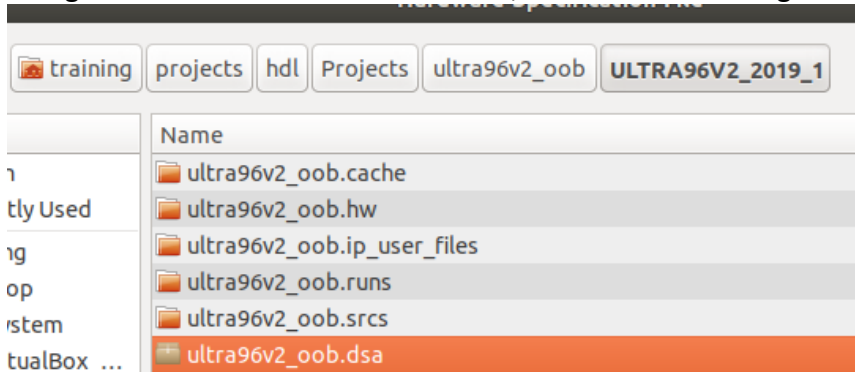
9. For Project name, insert the Platform name from the list in the beginning of this section, then click **Next>**

Here we are using the Ultra96V2 (ultra96v2\_oob)



10. Select “Create from hardware specification (DSA/HDF)” and click **Next>**

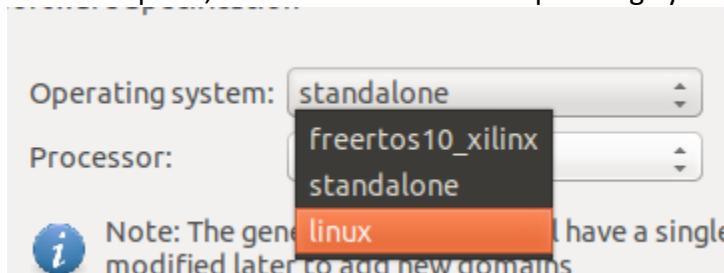
11. Navigate to the location of the DSA file, as noted in the image



12. Click **OK** to continue

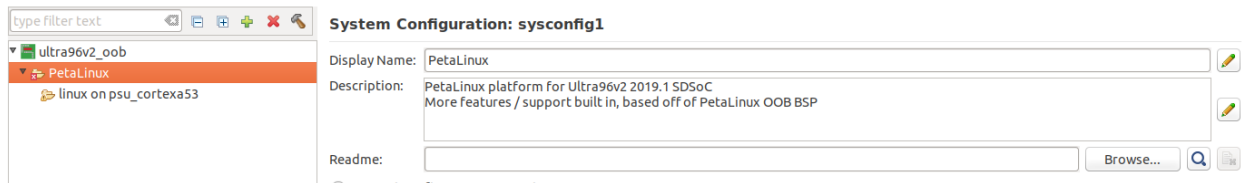
13. SDx will process the DSA

14. Once complete, select Linux under the Operating System selector

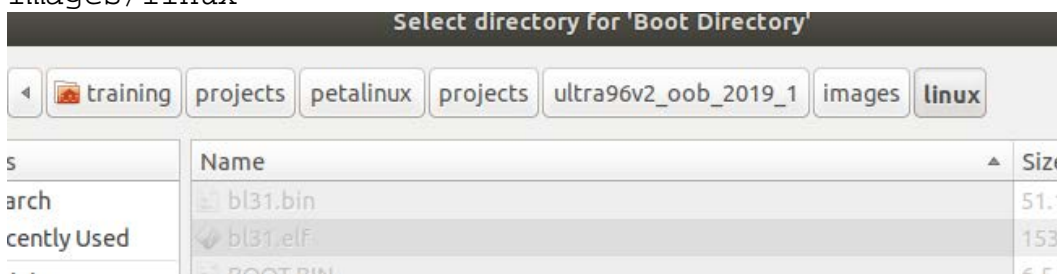


15. Click **Finish**

16. Once SDx has returned to the GUI, select the sysconfig1, and by clicking on the pencil button (on the RIGHT), change the Display Name and Description. Use a similar scheme as indicated in the image



17. Next, for the Boot Directory, click on **Browse**
18. Navigate to the location of the PetaLinux project, if using the recommended folder scheme, your files will be located under  
`~/projects/petalinux/projects/ultra96v2_oob_2019_1/images/linux`



19. For the Bif File, select the location where we stored the linux.bif from step 3  
`/home/training/projects/linux.bif`
20. Your sysconfig1 tab should look similar to the below

**System Configuration: sysconfig1**

Display Name: PetaLinux

Description: PetaLinux platform for Ultra96v2 2019.1 SDSoC  
More features / support built in, based off of PetaLinux OOB BSP

Readme:

☐ Generate software components  
Choosing this option would generate boot artifacts, fsbl, bif file etc.

☒ Use pre-built software components

Boot Directory: /home/training/projects/petalinux/projects/ultra96v2\_oob\_2019\_1/images/linux

Bif File: /home/training/projects/linux.bif

21. Now open the “linux on psu\_cortexa53” tab
22. Again, select the Pencil buttons and edit the Display Name and Description as indicated

Display Name: PetaLinux 2019p1

Description: PetaLinux System Configuration for Ultra96v2 2019.1 SDSoC



23. Now, click on the Orange “Click here” hyperlink

PetaLinux System Configuration f  
2019.1 SDSoC

is not configured. [Click here](#) to update.

24. The next dialog should be filled in as we have already selected the Boot Directory and BIF files. If these are NOT filled in, fill them in by pointing to the directory that contains your boot files as well as the linux.bif that we generated in step 3

25. Select **OK** to accept and continue

26. New selections will appear

Description:	PetaLinux System Configuration for Ultra96v2 2019.1 SDSoC
Image:	<input type="text"/>
Sysroot:	<input type="text"/>

27. We will need to point to the Image only – meaning the rootfs.tar.gz file

28. As before, click the browse button, then navigate to the images/linux folder under the PetaLinux project, selecting **OK** to dismiss the folder selection dialog

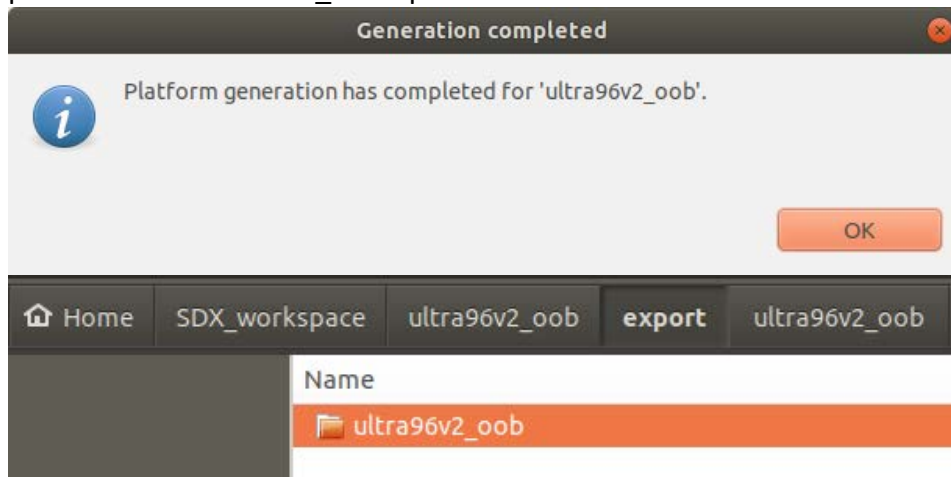
29. Now we can select Generate Platform from the 1,2,3 steps at the bottom of the page  
NOTE: we did not have to select steps 1,2 as they were already present in this version of the tools – had we wanted to add new system configurations or processor selections, we WOULD have to use steps 1,2

Image:	<input type="text" value="/home/training/projects/petalinux/projects/ultra96v2_oob_2019_1/images/linux"/>	<input data-bbox="1328 1192 1412 1218" type="button" value="Browse..."/>
Sysroot:	<input type="text"/>	<input data-bbox="1328 1234 1412 1260" type="button" value="Browse..."/>
QEMU Data:	<input type="text"/>	<input data-bbox="1328 1276 1412 1302" type="button" value="Browse..."/>
QEMU Arguments:	<input type="text"/>	<input data-bbox="1328 1318 1412 1344" type="button" value="Browse..."/>
PMU QEMU Arguments:	<input type="text"/>	<input data-bbox="1328 1360 1412 1386" type="button" value="Browse..."/>

#### Quick Links

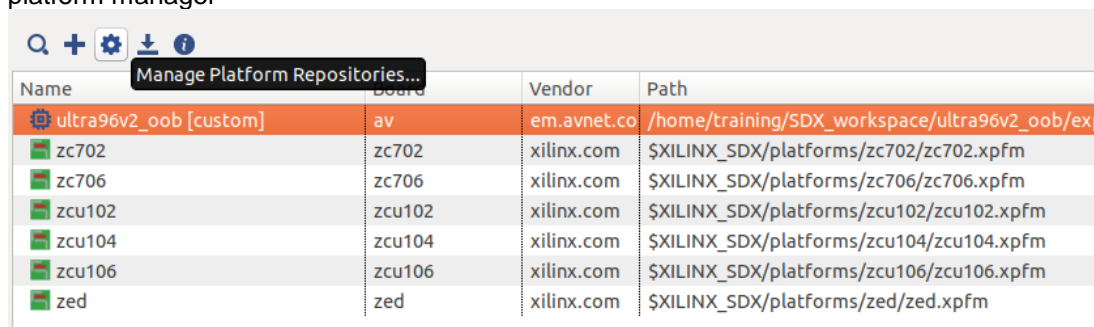
1 [Define System Configuration](#) 2 [Add Processor Group/Domain](#) 3 [Generate Platform](#)

30. Once complete, you can dismiss the dialog and you can locate an exportable copy of the platform under the SDX\_workspace folder

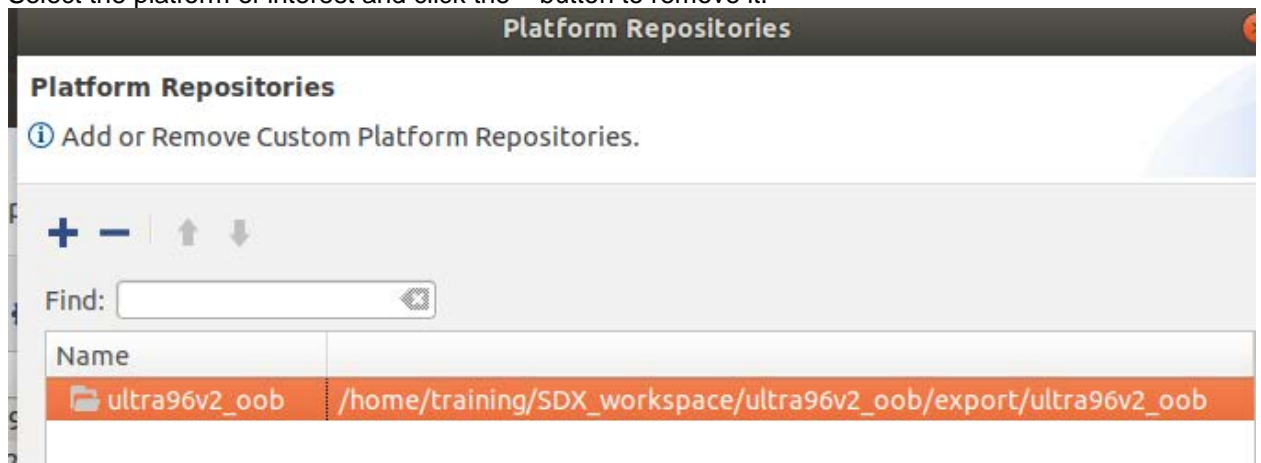


NOTE by default, SDSoc will include created platforms into the tool that it was generated on, however once you lose the SDX\_workspace folder, you will lose this platform and thus it is suggested to move this platform to the ~/platforms folder or to a safe backup location. To keep SDx from seeing TWO platforms, you can wipe the SDX\_workspace folder or manually REMOVE the extra platform by looking at the FOLDER that it has come from to determine the correct platform to remove.

To remove a platform, under the Platform selection window, click the GEAR to bring up the platform manager



Select the platform of interest and click the – button to remove it.



## Revision History

Date	Version	Revision
30 Nov 18	1p0	First public release
22 MAR 19	2p0	Updated to 2018.3 for U96V1 only
26 NOV 19	3p1	Updated to 2019.1 for UltraZed Platforms with instructions for Ultra96v2 when source files are released
02 DEC 19	3p2	Corrected Appendix GIT commands