



# Configuring Xilinx SDSoc for PetaLinux Based Platforms

Tools:	2018.2
Training Version:	v1.1
Date:	19 January 2019

© 2018 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.

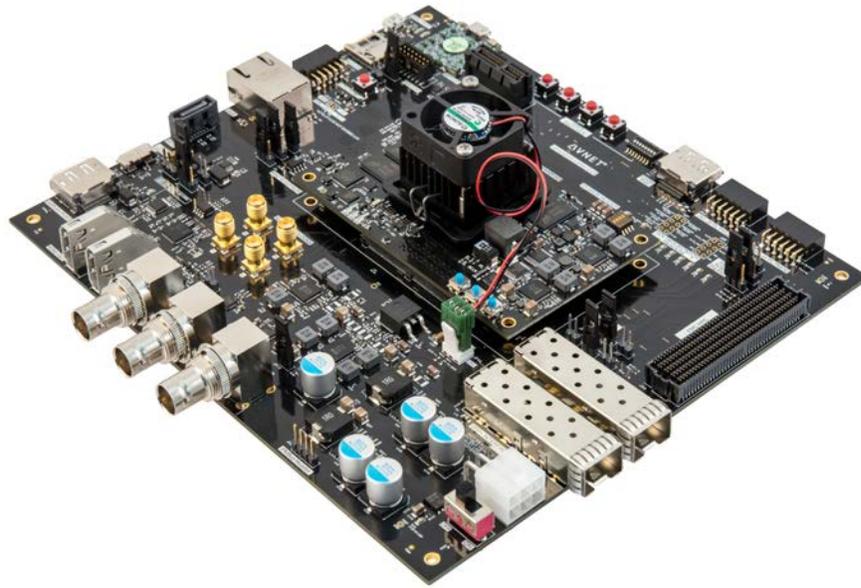
## Designed by Avnet

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.



**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.



**Figure 1 – UltraZed-EV Starter Kit**

Please contact your local Avnet FAE for further details with any of this or any of our Avnet Branded kits.

## Lab 1 Design Objectives

Lab 1 offers system developers an example of how to:

- Work through and become familiar with the SDSoC 2018.2 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 (you will learn later how to create that 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 v2018.2 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 2018.2 (SDSoC License Required)
- Platform archive
- Ultra96v1 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 Ultra96v1 (AES-ULTRA96-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

You must have installed the Xilinx tools and properly licensed them.

If using Windows, you will also need an unzip tool which can handle .tar.gz archives, such as 7-zip, although configuring the SDCARD will require Linux of some sort as an EXT4 partition will be used.

If not using a direct Linux install, it is recommended to follow Avnet's "VirtualBox and Linux VM Installation Guide" located on the UltraZed.org website.

The Board Definition files should be installed into both your SDx based Vivado installation. Instructions for installing board definitions are located on the UltraZed.org website.

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

(u96v1\_avnet, uzev\_avnetpl)

SDSoC comes pre-installed with many platforms that allow you to immediately use many Xilinx boards. The listed development board is not one of them. For this experiment, you will use pre-built hardware platforms called **<developmentBoard>\_avnet** so that you can quickly begin using SDx. You will later learn how to build that hardware platform.

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 `C:\Avnet\platforms` or `~/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 platform. Please note that screen shots of Windows or of Linux versions of software, will be applicable to BOTH Windows and Linux versions unless noted. In this case, a specific operating system or multiple screenshots will be included

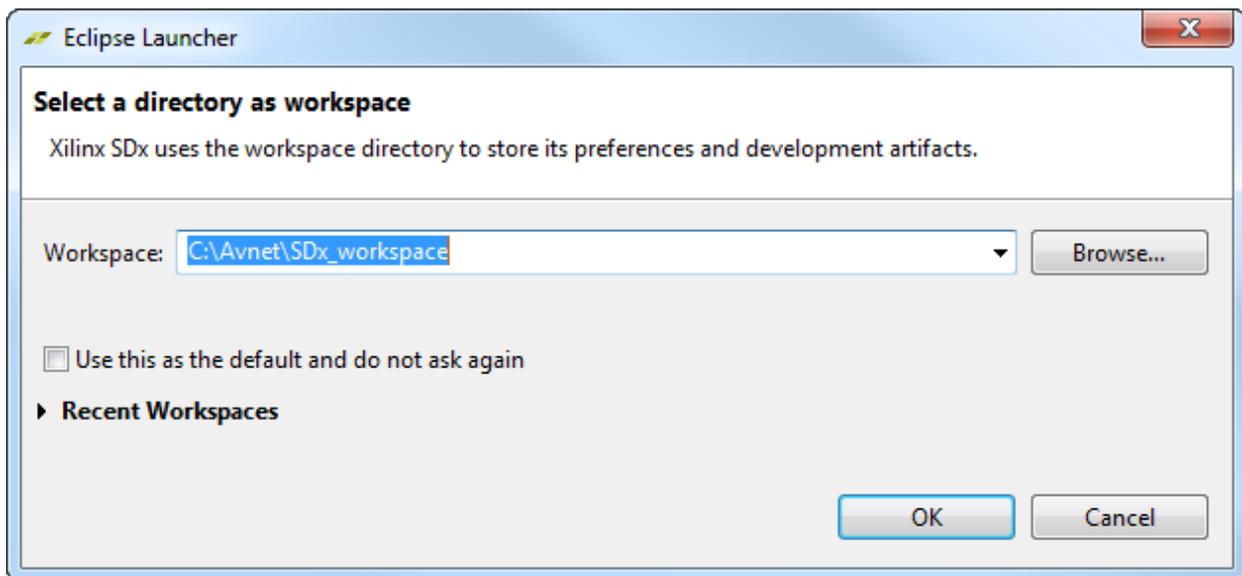
1. Launch the SDx IDE by clicking **Start → Xilinx Design Tools → SDx 2018.2 → SDx IDE 2018.2** or launch the Xilinx SDx IDE by executing the Desktop Icon installed within your Virtual Machine



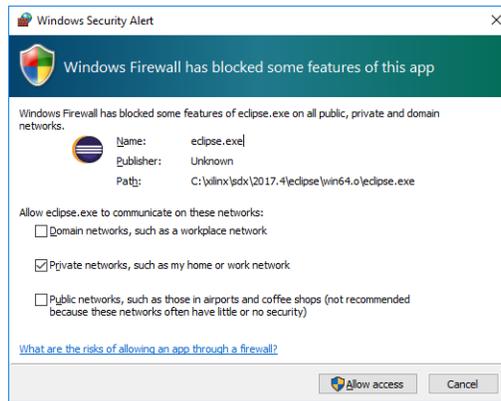
### Launch SDx IDE

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 **OK**.

**C:\Avnet\SDx\_workspace**

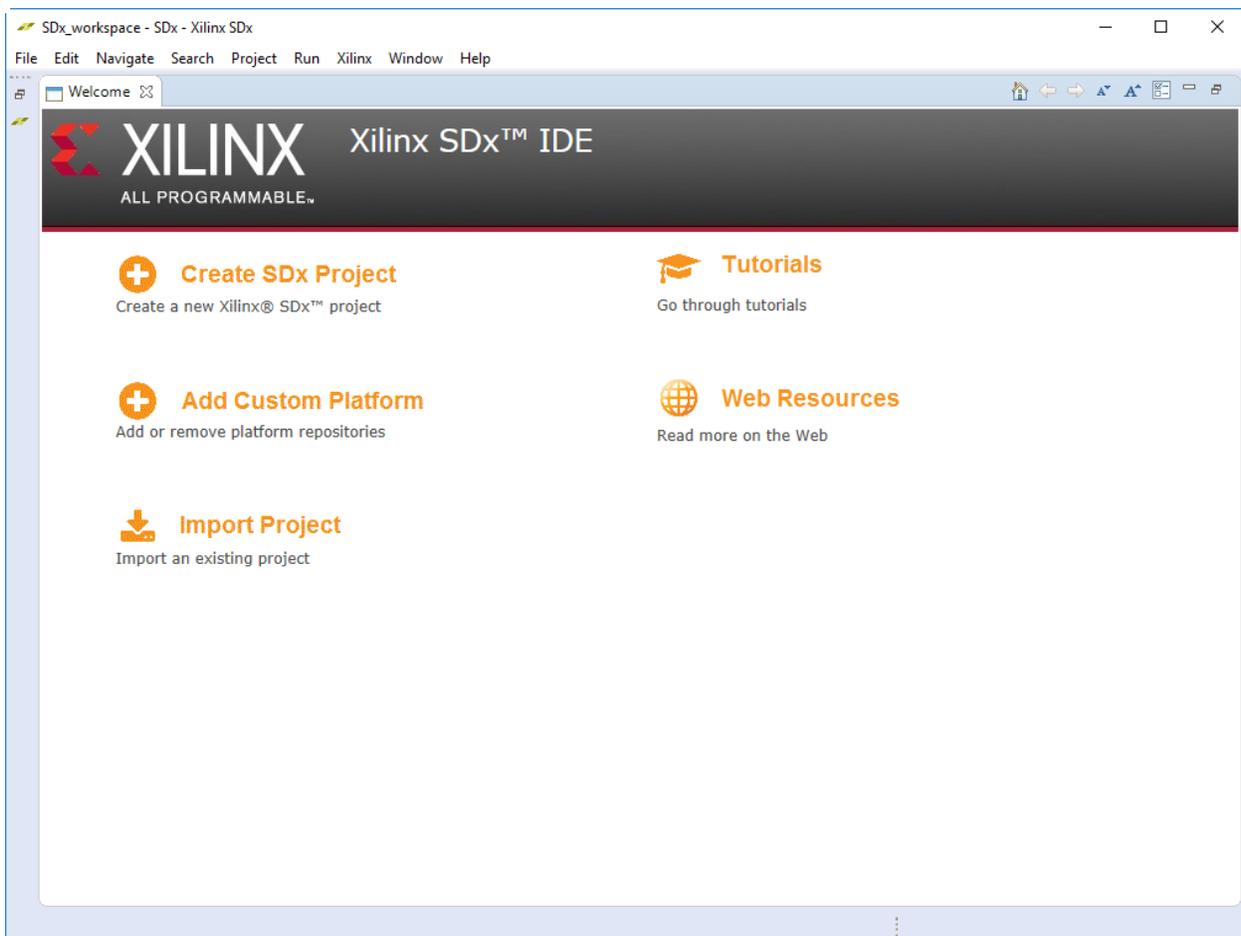


3. If you see a security alert, select **Allow Access**.



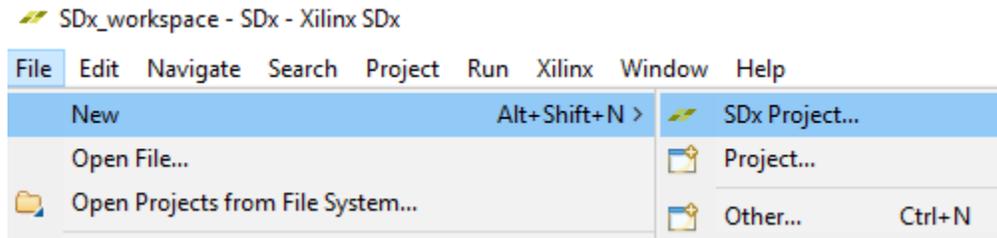
### Security Alert

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



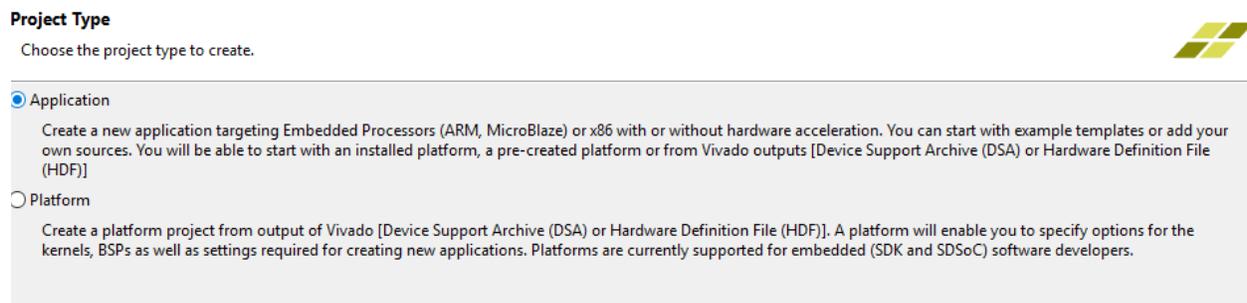
### SDx IDE Dashboard

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



### Xilinx SDx Project Creation

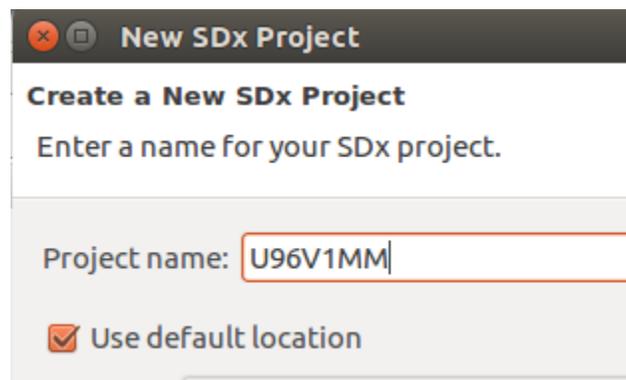
5. For Project Type, leave the default Application Project Selection, click **NEXT>**



### Project Type Selection

6. For Project Name, enter **U96V1MM** or **UZEVMM**

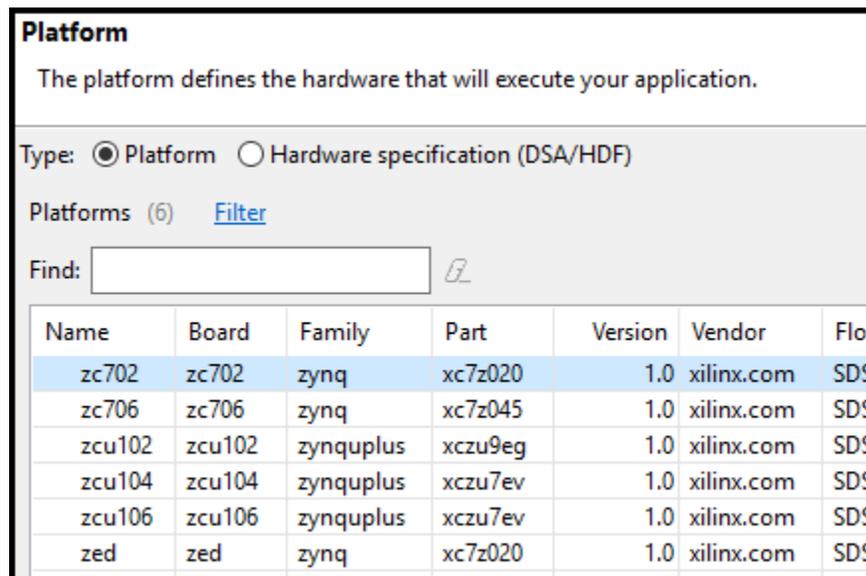
b. NOTE: the location of the project



### Project Location

7. 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.

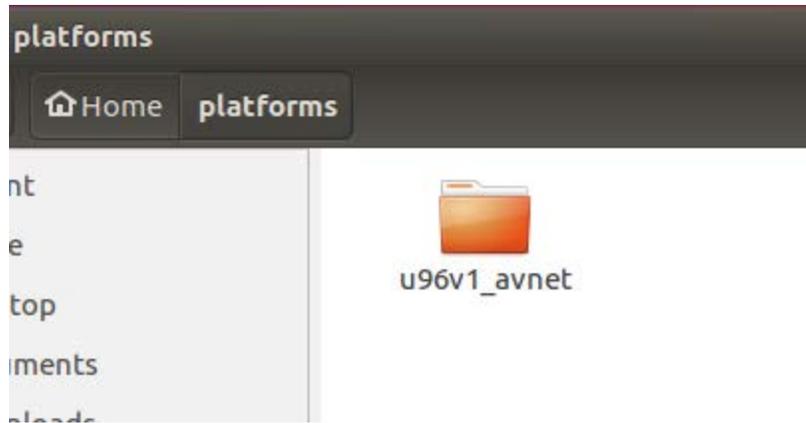


### Pre-installed Hardware Platforms in SDx 2018.2

8. Open Windows Explorer (or similar) and create the directory `C:\Avnet\platforms` or `~/platforms`

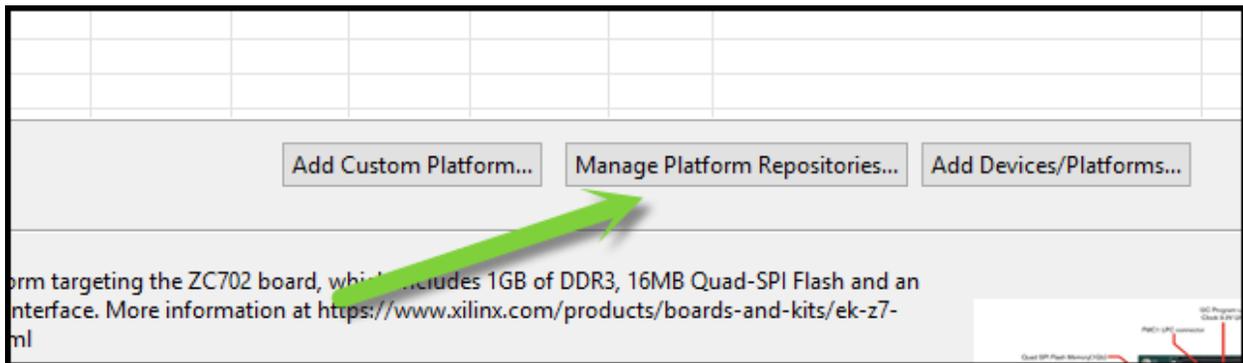
9. Now unzip the `u96v1_avnet_PetaLinux_Platform.tar.gz` archive into the repository at `C:\Avnet\platforms` or `~/platforms`

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



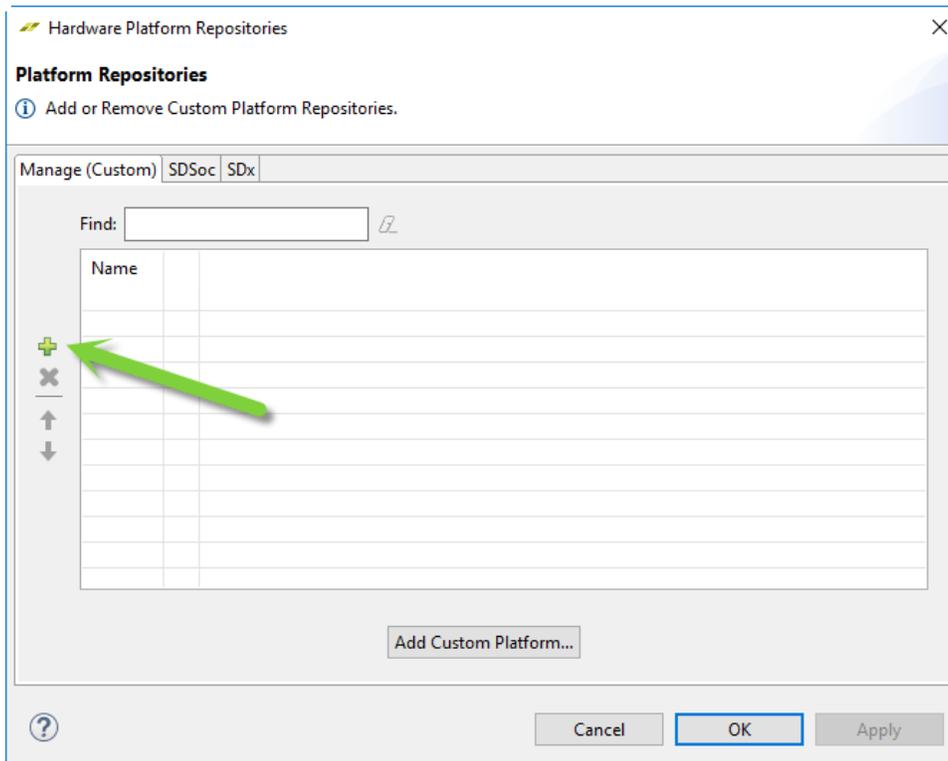
**SDx Platform Repository Folder**

10. In the SDx New Project dialog, click on **Manage Platform Repositories...**



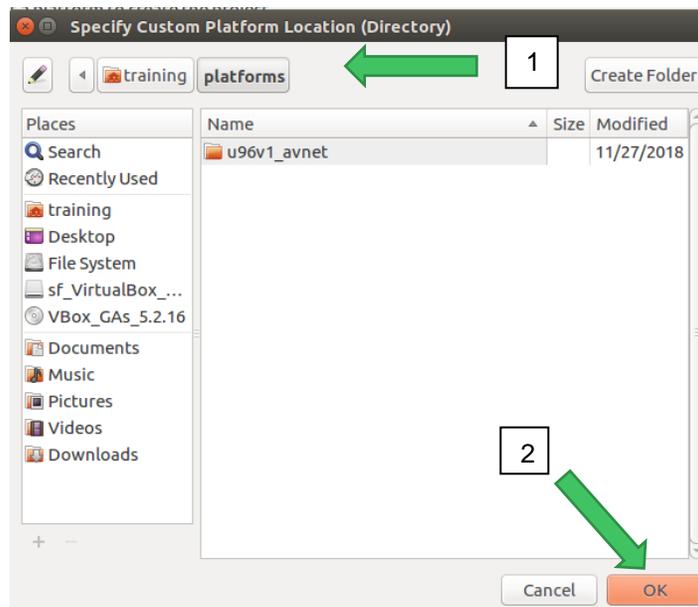
**Manage Repositories**

11. In the next dialog, click on the green plus sign on the left pane.



**Add Custom Platform Location**

12. Navigate to C:\Avnet\platforms or ~/platforms. Select **platforms** and click **OK**.



**Figure 2 – Specify Custom Platform Location**

13. Next Click on Apply. Click OK to dismiss the Hardware Platform Repositories Dialog.

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

Name	Board	Family	Part	Version	Vendor	Flow	Location
u96v1_avnet [custom]	av	zynqplus	xczu3eg	1.0	em.avnet.com	SDSoC	/home/training
zc702	zc702	zynq	xc7z020	1.0	xilinx.com	SDSoC	\$XILINX_SDX/p
zc706	zc706	zynq	xc7z045	1.0	xilinx.com	SDSoC	\$XILINX_SDX/p
zcu102	zcu102	zynqplus	xczu9eg	1.0	xilinx.com	SDSoC	\$XILINX_SDX/p
zcu104	zcu104	zynqplus	xczu7ev	1.0	xilinx.com	SDSoC	\$XILINX_SDX/p
zcu106	zcu106	zynqplus	xczu7ev	1.0	xilinx.com	SDSoC	\$XILINX_SDX/p
zed	zed	zynq	xc7z020	1.0	xilinx.com	SDSoC	\$XILINX_SDX/p

### u96v1\_avnet [custom] Now an Available Platform

You have now installed Avnet's custom platforms.

## Experiment 2: Create the Matrix Multiply Project

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

**New SDx Project**

**Platform**

The platform defines the hardware that will execute your application.

Type:  Platform  Hardware specification (DSA/HDF)

Platforms (7) **Filter**

Find:

Name	Board	Family	Part	Version	Vendor	Flow	Location
<b>u96v1_avnet [custom]</b>	av	zynqplus	xczu3eg	1.0	em.avnet.com	SDSoC	/home/training/platforms/u96v1_av
zc702	zc702	zynq	xc7z020	1.0	xilinx.com	SDSoC	\$XILINX_SDX/platforms/zc702/zc702
zc706	zc706	zynq	xc7z045	1.0	xilinx.com	SDSoC	\$XILINX_SDX/platforms/zc706/zc706
zcu102	zcu102	zynqplus	xczu9eg	1.0	xilinx.com	SDSoC	\$XILINX_SDX/platforms/zcu102/zcu
zcu104	zcu104	zynqplus	xczu7ev	1.0	xilinx.com	SDSoC	\$XILINX_SDX/platforms/zcu104/zcu
zcu106	zcu106	zynqplus	xczu7ev	1.0	xilinx.com	SDSoC	\$XILINX_SDX/platforms/zcu106/zcu
zed	zed	zynq	xc7z020	1.0	xilinx.com	SDSoC	\$XILINX_SDX/platforms/zed/zed.xpf

**Description**

u96v1\_avnet

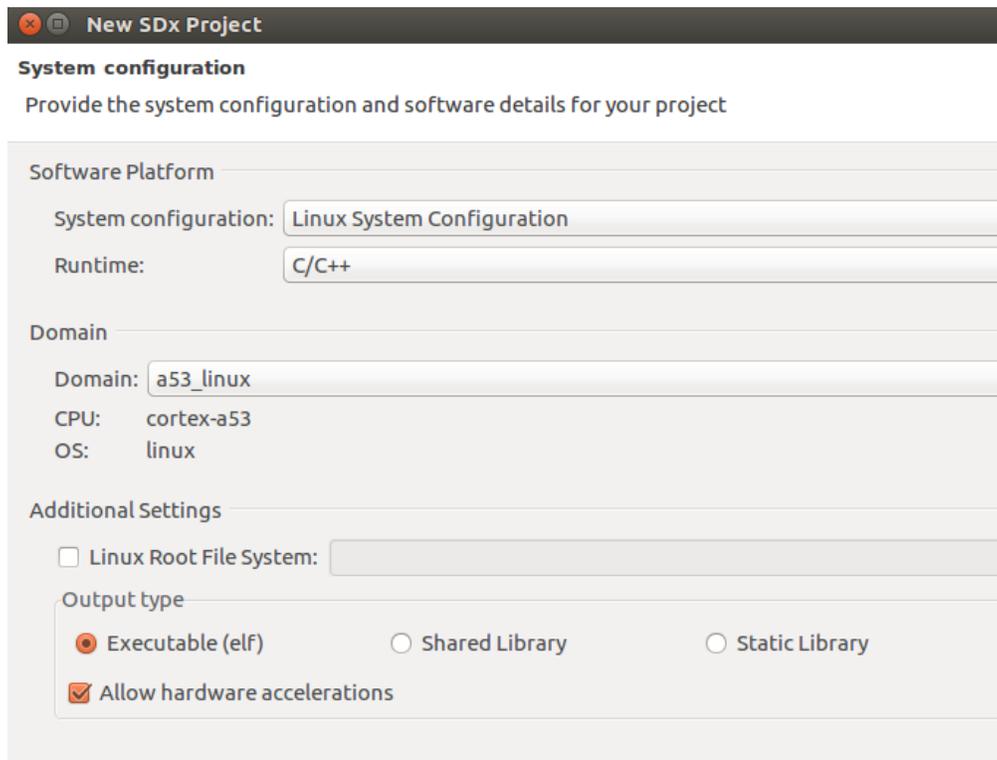
Repository: /home/training/platforms/u96v1\_avnet

**1**

**2**

**Choose u96\_avnet Hardware Platform**

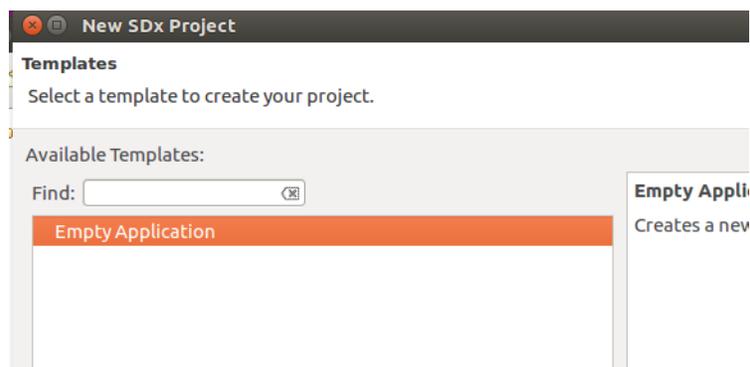
15. You can leave the defaults on the next screen where you need to choose the System Configuration. Note that this is also where you can choose to generate C-Callable Libraries, a new feature in current tools.



### Software Platform

16. Click **Next >** to continue.

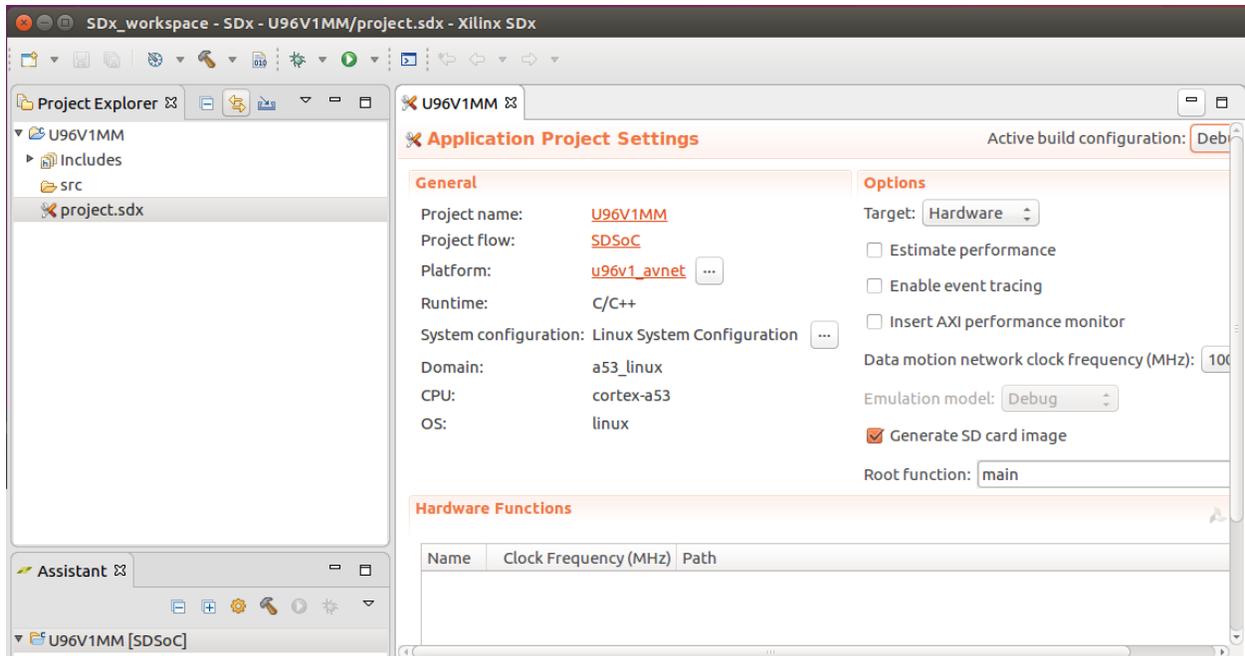
17. Select "Empty Application"



### Example Selection

18. Click **Finish** to create a new Empty application.

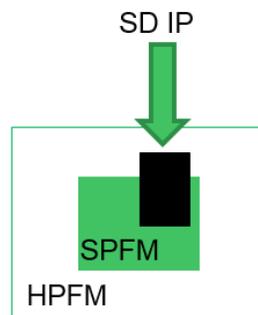
Now the SDx Project Explorer will have the U96V1MM from which we can generate SDSoc projects.



### U96V1MM added to SDx Project Explorer

For the purposes of this experiment, we will use Matrix Multiply Example code. As the template will not show up at this time, we will need to Import these files.

The example code that you choose 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.



**Figure 3 – Target Code is Black Box**

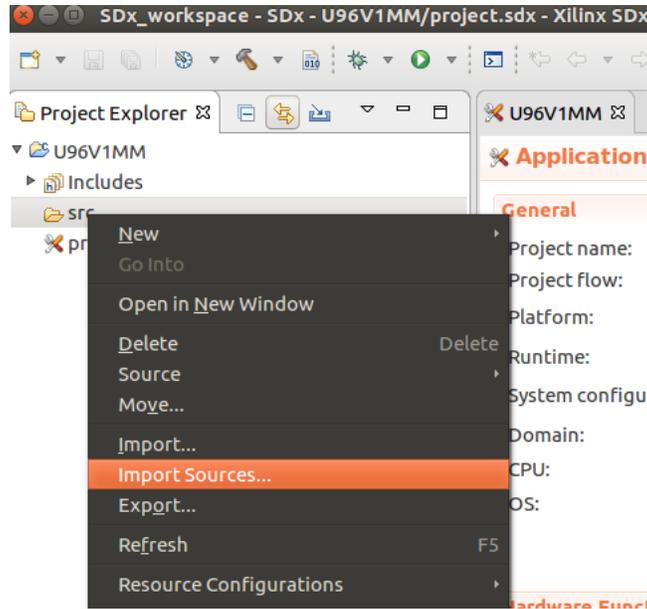
This is one of the most powerful abilities of SDSoC! Since we are using a provided platform, simply imported the example template straight from the platform folder.

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.

In order to import source files into our project, the operational flow is the same as with Vivado SDK. These basic steps are laid out in the following procedure.

19. Right click on the U96V1MM src folder and select **Import Sources...**



### Import Source Files

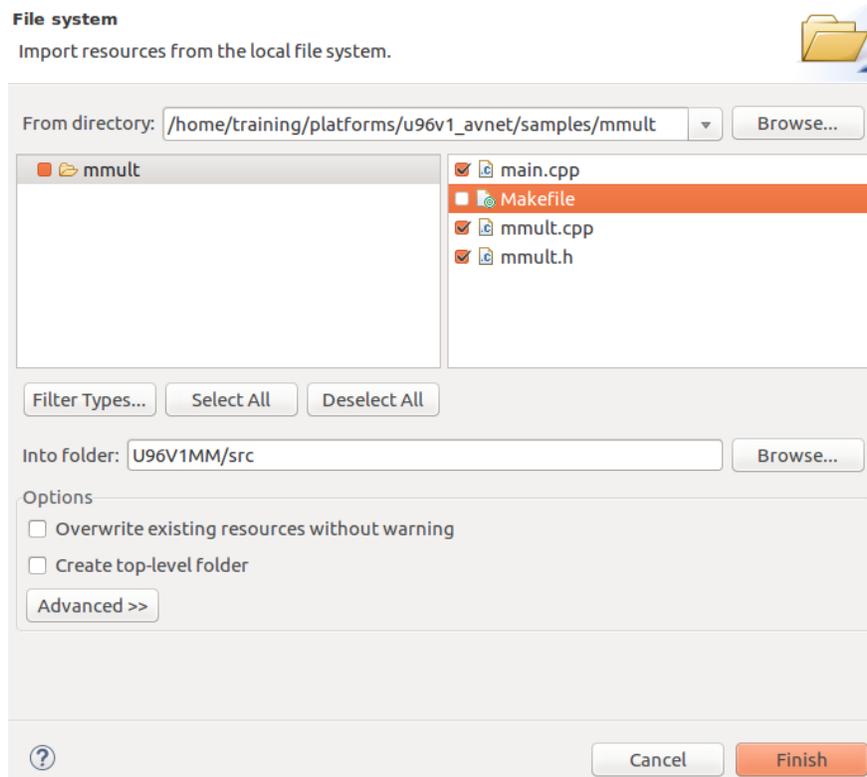
20. For the "From Directory" field, either copy/paste the location below or click on **Browse** then browse to the following example location:

`C:\Avnet\platforms\<platform Name>\samples\mmult`

Or

`~/platforms/<platform Name>/samples/mmult`

21. Next select three checkboxes next to the .cpp and .h files. Ensure that the **Into folder** box shows `<project Name>/src` as well as the checkboxes matches the screen in the below figure.

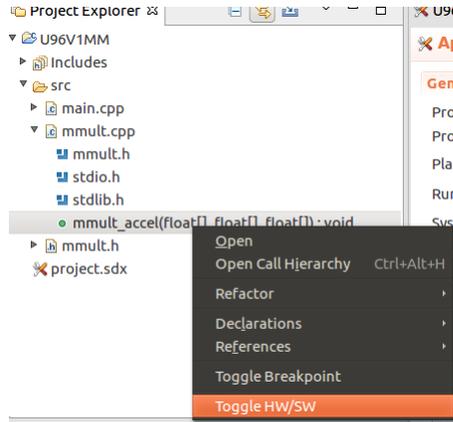


### Import Example Files

Click **Finish**. You should now see the three sources added in Project Explorer.

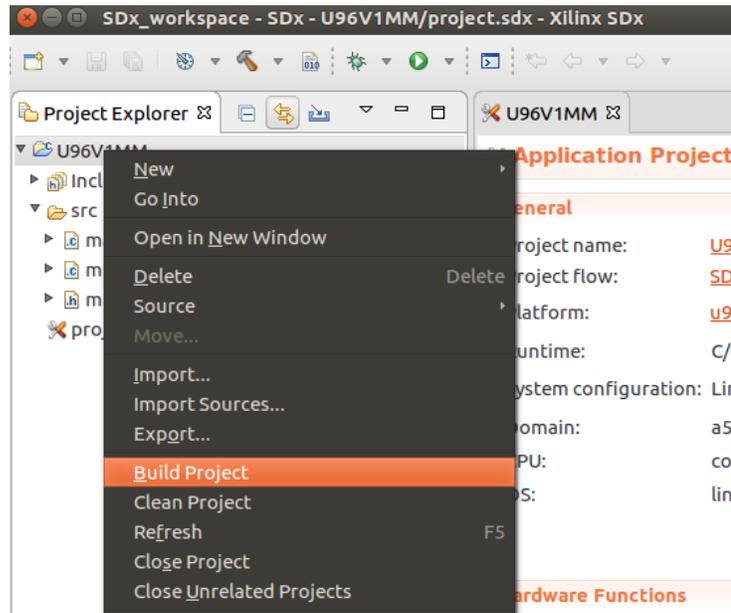
## Experiment 3: Build the Matrix Multiply Project

22. As we imported files, we will need to designate which functions to accelerate. Expand the mmult.cpp and right click on the mmult\_accel function. Select Toggle HW/SW



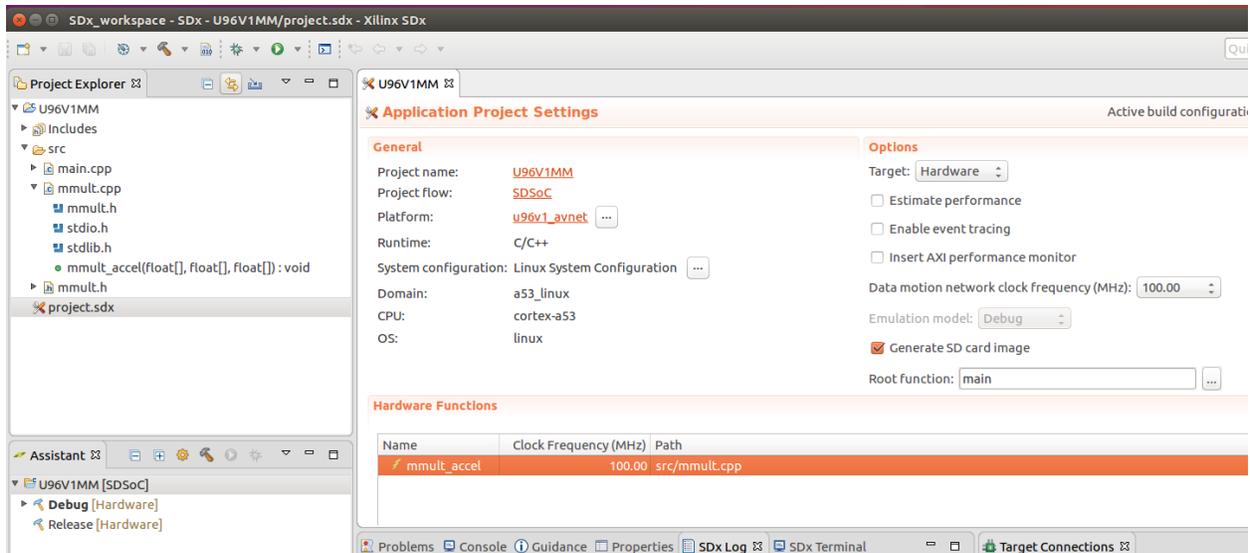
### Toggle Hardware Acceleration

23. Next, back in SDx Project Explorer right click the U96V1MM or UZEVMM project name, click on **Build Project**, remember, you can follow this same flow with any of the provided platforms



### Build Project

24. 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\_accel as a hardware accelerated function.



## Project Settings

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.

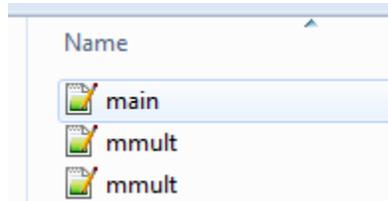
The above steps can be seen in more detail in UG1028 – SDSoC Intro Tutorial v2018.2.

NOTE: Use DocNav or search directly from Xilinx.com as there is a good chance you can end up with an older version of the documentation.

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

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



### **U96V1MM Source files**

More notes regarding the files and the file structure

- Main.cpp runs the functions
- Mmult.cpp has pragmas listed
- Mmult.h has pragmas listed

## Experiment 4: Set Up Your SDCARD

While the project is building, you can also set up your SDCARD. 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>

26. After inserting your SDCARD into your Linux host, open a terminal
27. Run `sudo fdisk /dev/sd?`
  - a. Replace the ? with the letter assigned to your SDCARD
28. From here, we will assign the main FAT32 partition by typing 'n' and then 'p' to make it primary, for size type '+1G'
29. Now make it bootable by typing 'a'
30. From here, assign the remaining space as the EXT4 partition by again typing 'n', then choosing the defaults for the first and last sector
31. 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

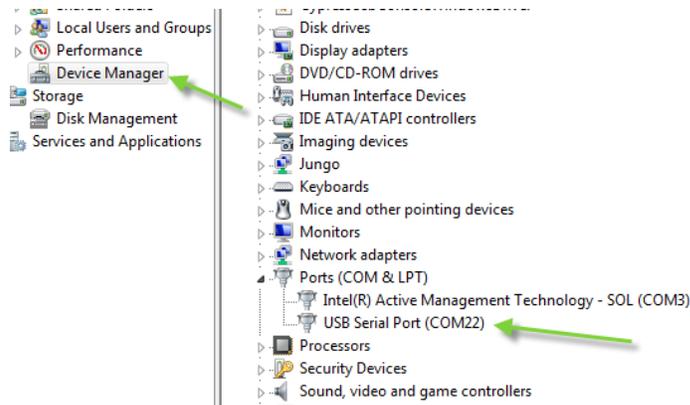
Command (m for help): █
```

32. If you see a similar configuration as the above image, type 'w' to WRITE the partitions and exit
33. 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

34. Insert one USB cable into the Ultra96V1 USB JTAG/UART POD MicroUSB connector
35. Plug the other end into your PC
36. If this is the first time you have connected your JTAG POD, locate the COM Port assigned, if you are using Ubuntu, you have already configured this, skip to step 37
  - a. Right click on “My Computer” and choose Manage
  - b. From here, select Device Manager
  - c. Under the Ports section, locate the USB Serial Port Assignment



**Device Manager Used to Find USB Serial Port COM Port**

37. Open a terminal program such as TeraTerm, or the Serial Terminal configured in the Virtual Machine Setup Guide. Configure it to connect to the COM Port we found in the previous step using 115200/8/n/1/n as settings.
  - a. NOTE: if you open C:\Program Files (x86)\teraterm\TERATERM.INI you can change the defaults so you do not have to continuously modify the settings each time you open a serial port

```
; Serial port parameters
; Port number
ComPort=1
; Baud rate
BaudRate=9600
; Parity (even/odd/none/mark/space)
Parity=none

; Serial port parameters
; Port number
ComPort=4
; Baud rate
BaudRate=115200
; Parity (even/odd/none/mark/space)
Parity=none
```

**Changing TeraTerm Defaults**

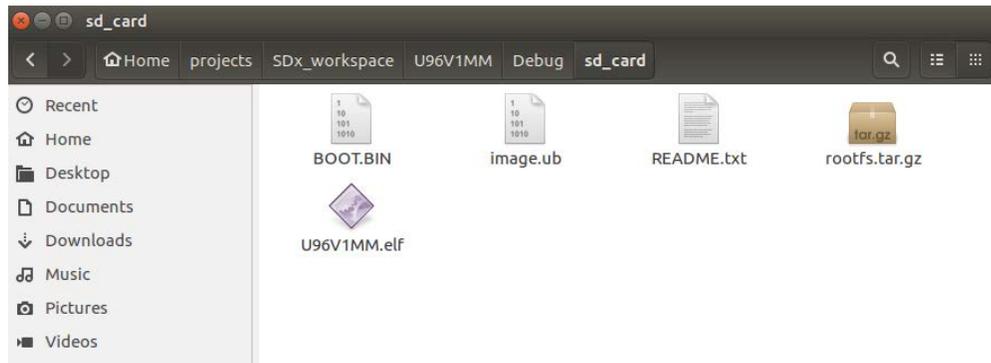
38. Use a file manager to copy the contents of the sd\_card folder to your BOOT partition on the SDCARD. The only file you do not have to copy over is the rootfs.tar.gz file

The files are located:

C:\Avnet\SDx\_workspace\**<project Name>**\Debug\sd\_card

or

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



### Generated Files Located in the project folder



### Project Files on the BOOT partition

39. Next 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$
```

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

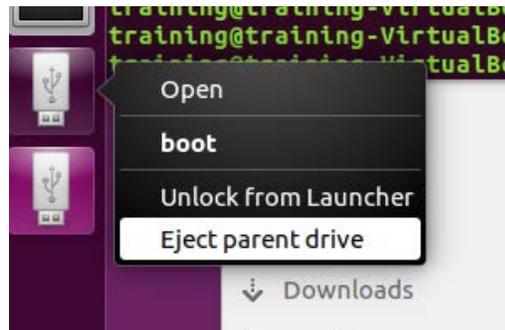
a. `sudo tar -xvf ~/projects/SDx_workspace/U96V1MM/Debug/sd_card/rootfs.tar.gz`

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

41. Once this extraction completes, change out of that folder in preparation for removing the SDCARD from the Linux machine. `cd ~`

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

43. 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



44. Insert the SDCARD into the SDCARD cage.  
45. Turn on the power and you should observe a similar display, as shown below for Ultra96v1

```
[2605:2723:1127/154001:ERROR:browser_gpu_channel_host_factory.cc(123)]
o errors during initialization
[2605:2723:1127/154001:ERROR:browser_gpu_channel_host_factory.cc(123)]
launch GPU process.
[2605:2723:1127/154001:ERROR:browser_gpu_channel_host_factory.cc(123)]
launch GPU process.

PetaLinux 2018.2 xilinx-ultra96-reva-2018_2 /dev/ttyPS0
xilinx-ultra96-reva-2018_2 login: █
```

46. From here, login with `root` as the username and `root` as the password  
47. As we are not using the wifi, in order to help keep our terminal clear of unnecessary messages, run `ifconfig wlan0 down`

```
root@xilinx-ultra96-reva-2018_2:~# ifconfig wlan0 down
[ 134.575320] wlcore: down
root@xilinx-ultra96-reva-2018_2:~# █
```

48. 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 /media/sdcard/`

```
root@xilinx-ultra96-reva-2018_2:~# cd /media/
root@xilinx-ultra96-reva-2018_2:/media# mkdir sdcard
root@xilinx-ultra96-reva-2018_2:/media# mount /dev/mmcblk0p1 /media/sdcard/
root@xilinx-ultra96-reva-2018_2:/media# █
```

49. Now in order to execute the file, we will copy the file from the FAT32 partition to the EXT4 file system. Provide the proper permissions and execute the program
- a. `cp /media/sdcard/<project Name>.elf .`
  - b. `chmod 777 <project Name>.elf`

c. ./ <project Name>.elf

```
root@xilinx-ultra96-reva-2018_2:/media# cp /media/sdcard/U96U1MM.elf .
root@xilinx-ultra96-reva-2018_2:/media# ls
U96U1MM.elf sdcard
root@xilinx-ultra96-reva-2018_2:/media# chmod 777 U96U1MM.elf
root@xilinx-ultra96-reva-2018_2:/media# ./U96U1MM.elf
[ 940.635151] xlnk_eng_probe ...
[ 940.638162] uio name xilinx-xlnk-eng.0
[ 940.642016] xilinx-xlnk-eng xilinx-xlnk-eng.0: physical base : 0xa0000000
[ 940.648732] xilinx-xlnk-eng xilinx-xlnk-eng.0: register range : 0x1000
[ 940.655237] xilinx-xlnk-eng xilinx-xlnk-eng.0: base remapped to: 0xfffff800
```

**Matrix Multiply being copied to executable space,  
and the start of the MM application work with  
the xlnk PetaLinux SDSoc driver**

```
Testing 1024 iterations of 32x32 floating point mmult...
Average number of CPU cycles running mmult in software: 1580742
Avg [ 942.328001] xilinx-xlnk-eng xilinx-xlnk-eng.3: xilinx-xlnk-e
ered
Average number of CPU cycles running mmult in hardware: 50687
Spe [ 942.340711] xilinx-xlnk-eng xilinx-xlnk-eng.0: xilinx-xlnk-
tered
ed up: 31.1863
TEST PASSED
[ 942.352241] xilinx-xlnk-eng xilinx-xlnk-eng.1: xilinx-xlnk-eng
```

**Matrix Multiply Running in Both Software and Hardware, Ultra96v1**

50. Cleaned up, you can see:

```
Testing 1024 iterations of 32x32 floating point mmult...
Average number of CPU cycles running mmult in software: 1580742
Average number of CPU cycles running mmult in hardware: 50687
Speed up: 31.1863

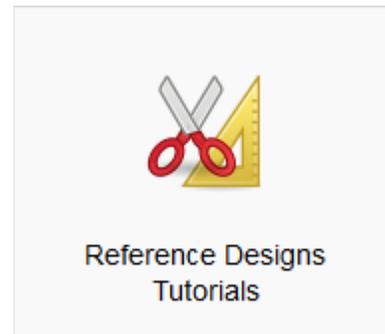
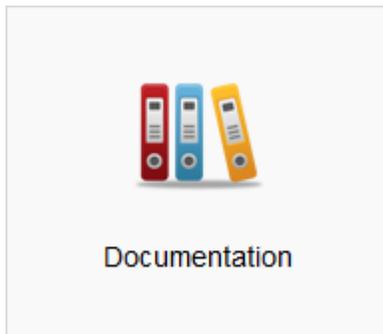
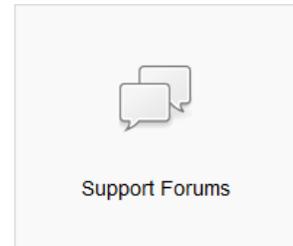
TEST PASSED
```

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

## Appendix A: Getting Support

### Avnet Support

- Technical support is offered online through the [minized.org](http://minized.org) website support forums. MiniZed users are encouraged to participate in the forums and offer help to others when possible.
- To access the most current collateral for the MiniZed, visit the community support page ([www.minized.org/content/support](http://www.minized.org/content/support)) and click one of the icons shown below:



- MiniZed Documentation  
<http://minized.org/support/documentation/18891>
- MiniZed Reference Designs  
<http://minized.org/support/design/18891/146>
- Ultra96 Documentation  
<http://ultra96.org/support/documentation/24166>
- Ultra96 Reference Designs  
<http://ultra96.org/support/design/24166/156>
- UltraZed-EV Documentation  
(Starter Kit) <http://ultrazed.org/support/documentation/22596>  
(SOM) <http://ultrazed.org/support/documentation/25481>  
(Carrier Card) <http://ultrazed.org/support/documentation/22581>
- UltraZed-EV Reference Designs  
<http://ultrazed.org/support/design/22581/166>

## Xilinx Support

The following technical support options are available to Xilinx customers:

- Technical information is available online 24 hours a day from the [Support website](#)
- Technical Support staff are available to respond to your questions in the [Community Forums](#)
- Individual assistance from Xilinx Technical Support **may** be available through [Service Portal](#)
- Phone support is **only** available with an active open case number

## Global Phone Number

Region	Language	Phone**	Support Hours*
North America	EN	1 800-255-7778 or +1 408-879-5199	M-F 7:00 -17:00 PST
Europe, Middle East and Africa	EN, DE, FR	00 800-5152-5152 or +353 1-461-5700	M-F 8:00 -17:00 GMT
China	CH (Mandarin), EN	+86 800 988 0218 +86 400 880 0218 (Mobile Phone)	M-F* 9:00 -18:00 CST
Taiwan	CH (Mandarin), EN	+886 2-8176-1060	M-F 9:00 -18:00 CST
Hong Kong	CH (Mandarin), EN	+852 3187-3855	M-F 9:00 -18:00 CST

\* Support hours listed apply for both standard and daylight savings (summer) time. Please check the [Technical Support Holiday Calendar 2016](#) for support availability during holidays in your region.

\*\* 00 800-5152-5152 is a international free phone (toll free) number available in the following countries: Austria, Belgium, Denmark, Finland, France, Germany, Ireland Israel, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland, and United Kingdom. All other countries must use +353 1-461-5700.

\*\* For the numbers listed, '+' represents the International Direct Dialing (IDD) prefix of the country from which you are calling. Please consult your local telephone service provider for more information on specific IDD instructions.

# Revision History

Date	Version	Revision
30 Nov 18	1p0	First public release