

Mastering Programmable Logic with SDSoc and Ultra96

Tools:	2018.2
Training Version:	v4
Date:	1 October 2018

© 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 lab work contained herein, you shall learn what an output of SDSoC is, while becoming familiar with the 2018.2 version of SDSoC tool flow. Using this base knowledge, you will be able to better understand the needs of SDSoC while learning to create a platform from scratch. 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.

Lab Design Objectives

Lab 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 Multiple Example output on a Ultra96 using a pre-built SDSoC platform

In order to see where we need to get with SDSoC, we need to take a journey. That journey will start with a pre-built SDSoC platform for Ultra96. This pre- built platform is provided for you, and we will utilize a built in design. Using these outputs, we will begin to understand the advantages to SDSoC. With very little effort, we can generate an entire design with knowing little more than the C/C++ code that we wish to accelerate.

Underneath all of this, SDSoC will be busy determining the optimal processing system, programmable logic, data movers (AXI Interconnect, DMA, etc.), as well as any other available hardware needs to accelerate your design within the constraints that you provided. For this lab, most of this is transparent to the end user.

Leveraging this foreknowledge will help us understand and appreciate all of the things that SDSoC needs and automates for the end user. This will help us understand the value that SDSoC will give to the end user as they concentrate on their development needs.

Example Design Requirements

Software

The software used in this lab includes:

- Xilinx SDx / SDSoC 2018.2
- Ultra96 SDSoC Platform
- Ultra96 v1.2 Board Definition for Vivado

Experiment Setup

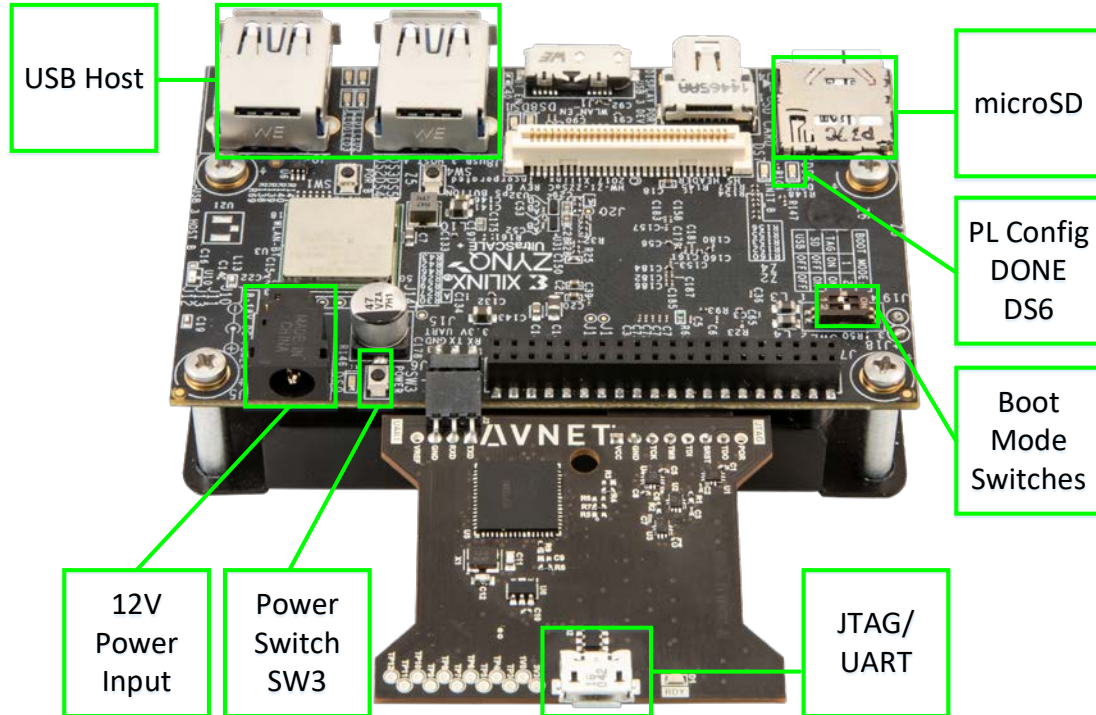


Figure 1 – Ultra96 Details

1. Connect the 12V power supply to the Ultra96 barrel jack (J5).
2. Connect the USB JTAG/UART port of the Ultra96 JTAG/UART adapter (J1) to a PC using a MicroUSB cable.
3. Insert the supplied USB Ethernet dongle into either USB host port on the Ultra96 board and connect the Ethernet port of the dongle directly to the Ethernet port on the host PC.
4. Insert the microSD card into the SD card slot (J4).

Experiment 0: Things to do First


Before we get started with the lab experiments, there are a few things we need to do first

1. Verify the Ultra96 board is assembled and connected to the host laptop as shown in [Experiment Setup](#) and that the boot mode switches are set for SD card boot. The silkscreen next to SW2 on the PCB describes these switch settings.
2. Turn on the Ultra96 board by pressing the small **SW3** power switch near the power input barrel jack. You should see the DS6 (DONE) LED turn green and a short time later the DS8 (WiFi) and DS1 (Bluetooth) LEDs will turn on. Once the board has booted the LED0 LED will blink in a heartbeat pattern.
3. Log into the Ubuntu laptop.

User: user

Password: password



4. Open a command terminal window by clicking the  icon on the Ubuntu taskbar. Alternatively you may right-click on an empty area of the Ubuntu desktop and select **Open Terminal**. Leave this open until told to close it.

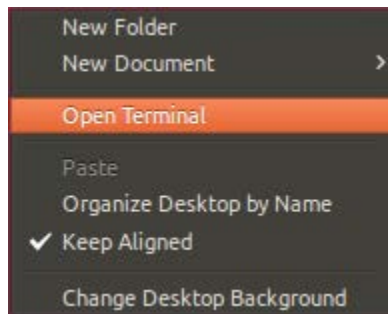


Figure 2 – Open Terminal

5. Determine the COM port being used by the USB UART.

```
$ ls /dev |grep USB
```

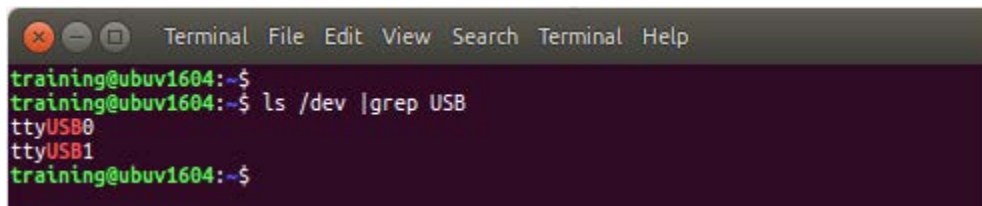
A terminal window titled 'Terminal' with a menu bar containing 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows a user prompt 'training@ubuv1604:~\$' followed by the command 'ls /dev |grep USB'. The output shows 'ttyUSB0' and 'ttyUSB1' on separate lines. The prompt returns to 'training@ubuv1604:~\$'.

Figure 3 – USB Uarts

You should see two USB ports identified. The USB0 device is used for JTAG and the USB1 device is the UART.

6. Open a PuTTY UART console window in a background shell for the COM port used by the Ultra96 board.

```
$ putty /dev/ttyUSB1 -serial -sercfg 115200 &
```

7. Configure gmake through a symbolic link

```
$ sudo ln -s /usr/bin/make /usr/bin/gmake
```

When you are asked for the password, remember the password is...password

8. Install Board Definition Files

- a. Navigate to Avnet's Ultra96.org website

<http://ultra96.org/support/documentation/24166>

- b. Support → Documentation
- c. Select View for Ultra96
- d. Select Download for Board Definition Files

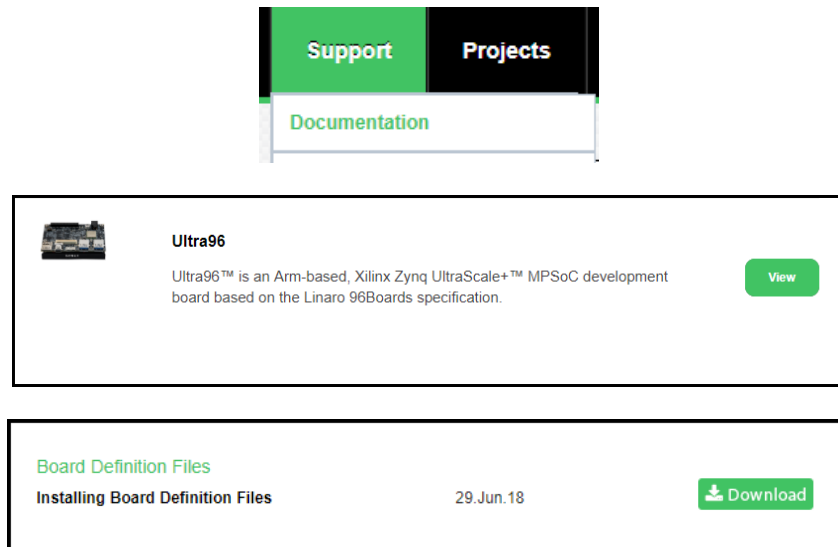


Figure 4 – Navigation Path

Experiment 1: Run the u96_avnet Matrix Multiply Project

We will not be building the project due to time constraints. Instead, first we will run the prebuilt design.

We first need to copy the boot.bin file from the provided work folder to the SDCARD

1. Insert the microSD card into the adapter and connect into the laptop's SDCARD reader



2. Next open the file explorer
3. Navigate to /home/user/xd_f_labs/emb_lab2/wrk

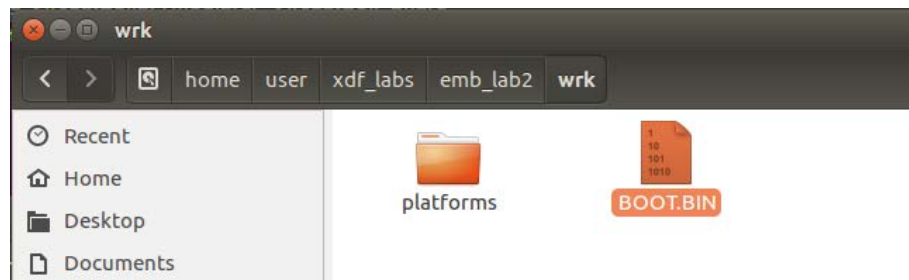


Figure 5 – boot.bin location

4. Copy this file to /media/user/boot

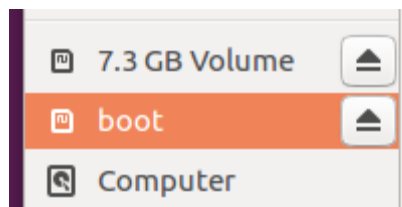
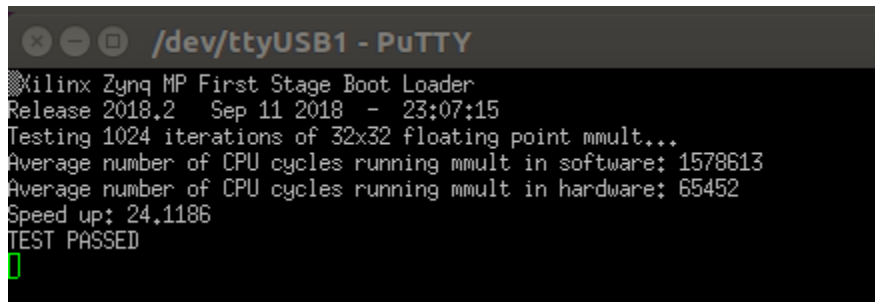


Figure 6 – microSD Card Map and Eject Arrow

5. Once this is copied, press the eject arrow and take the card out of the laptop
6. Insert this into the Ultra96 microSD card cage
7. Apply power by pressing the SW3 as shown in the Experiment Setup

8. Note the 24x increase in performance as indicated by the PuTTY window



```
xilinx Zynq MP First Stage Boot Loader
Release 2018.2   Sep 11 2018   - 23:07:15
Testing 1024 iterations of 32x32 floating point mmult...
Average number of CPU cycles running mmult in software: 1578613
Average number of CPU cycles running mmult in hardware: 65452
Speed up: 24.1186
TEST PASSED
```

Figure 7 – 24x acceleration of matrix multiply algorithm

Experiment 2: Install the Pre-Built Ultra96 (u96_avnet) SDx Hardware Platform

SDSoC comes pre-installed with many platforms that allow you to immediately use many Xilinx boards. Ultra96 with bare metal is not one of them. For this experiment, you will use a pre-built hardware platform called **u96_avnet** 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 `/home/user/xd_f_labs/emb_lab2/wrk/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 `u96_avnet` platform.

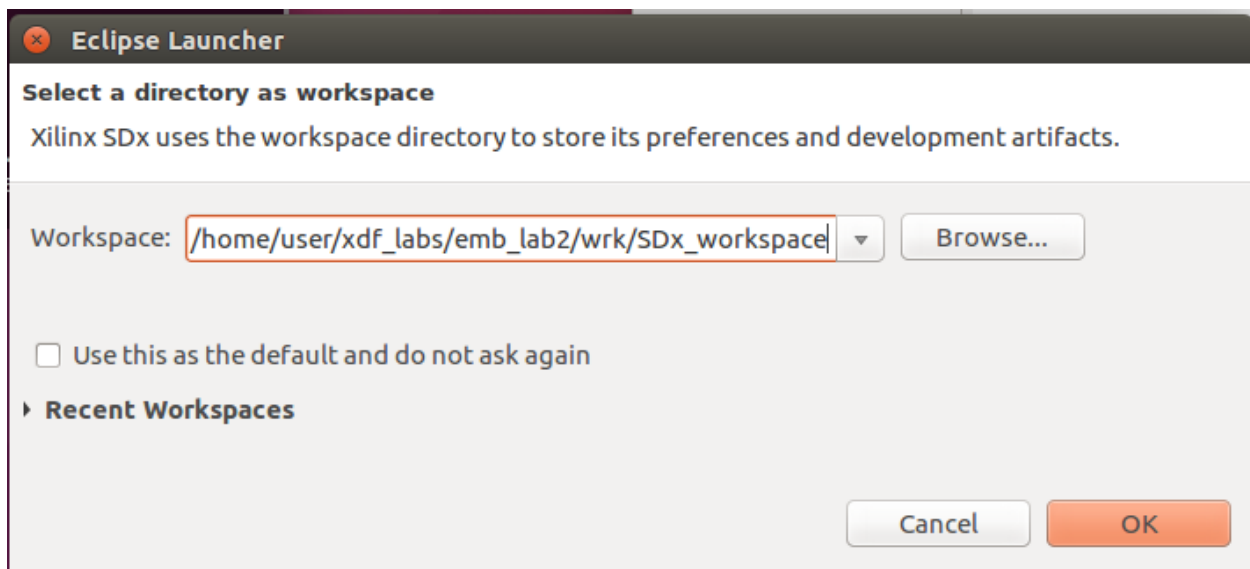
1. Launch the SDx IDE by executing the following command in the already opened terminal window

```
$ source /opt/Xilinx/SDx/2018.2/settings64.sh
```



```
$ sd_x &
```
2. SDSoC requires a workspace. Please enter this specific path for consistency through these labs and then click **OK**.

`/home/user/xd_f_labs/emb_lab2/wrk/SDx_workspace`



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

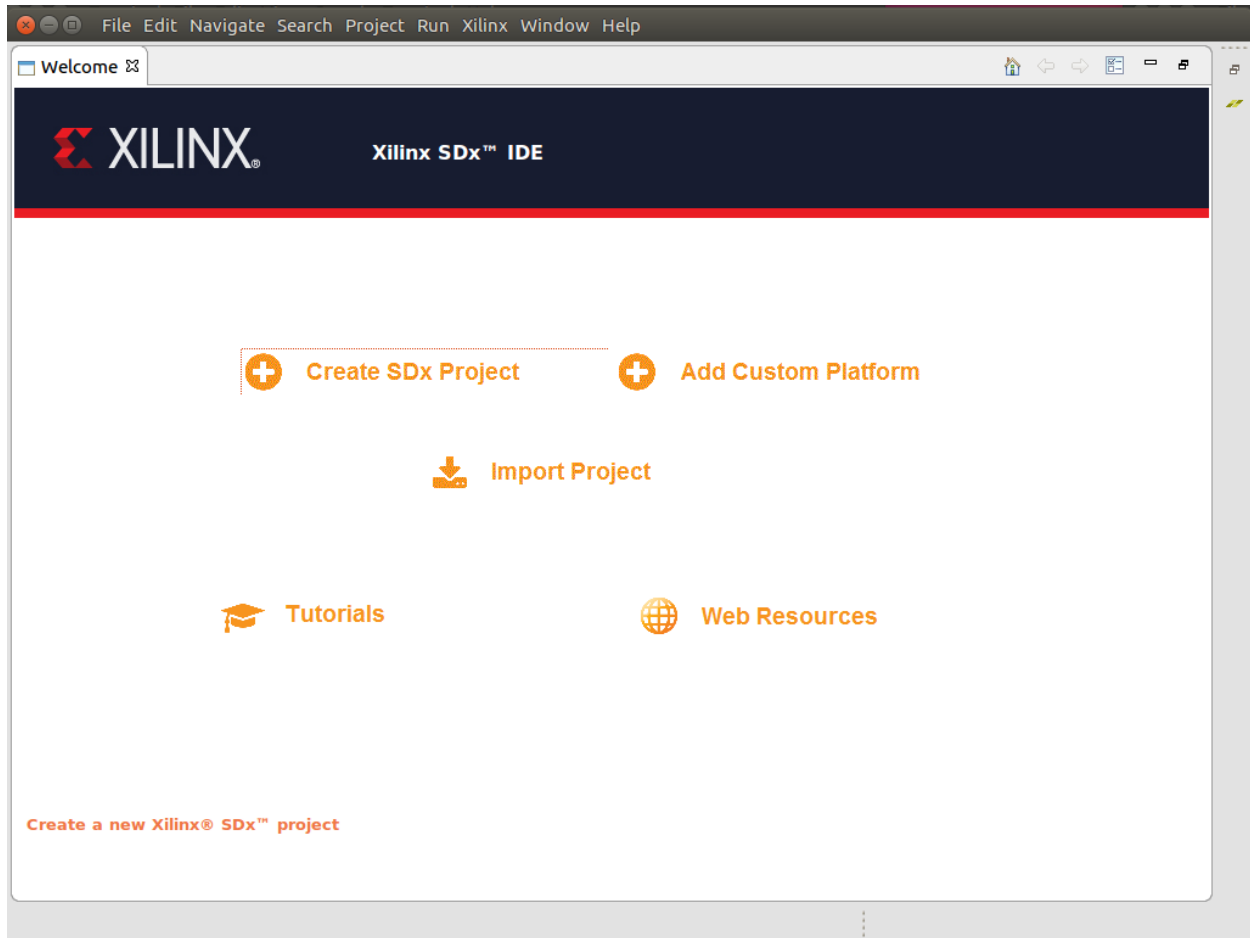


Figure 8 – SDx IDE Dashboard

Begin by creating a new Xilinx SDx project

- a. Select **File → New → SDx Project...** or click **Create SDx Project** on Welcome screen

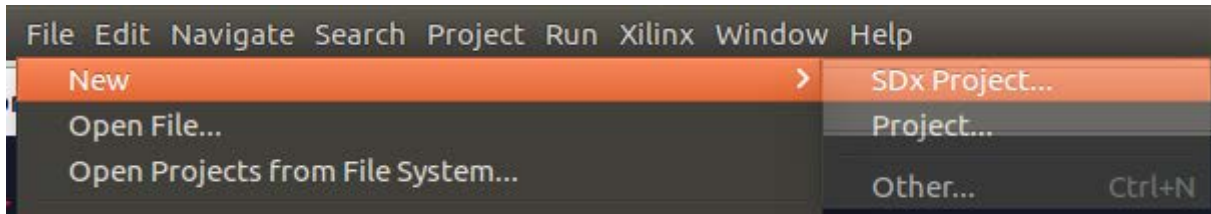


Figure 9 – Xilinx SDx Project Creation

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

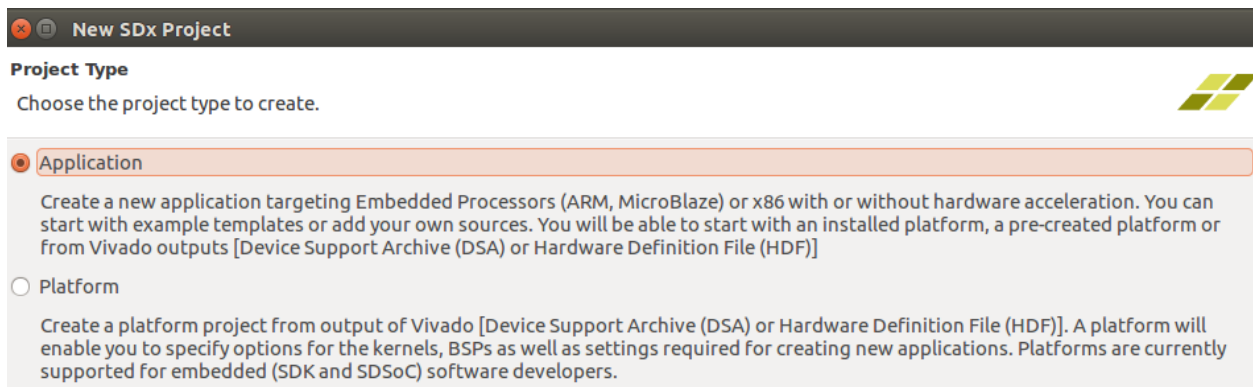
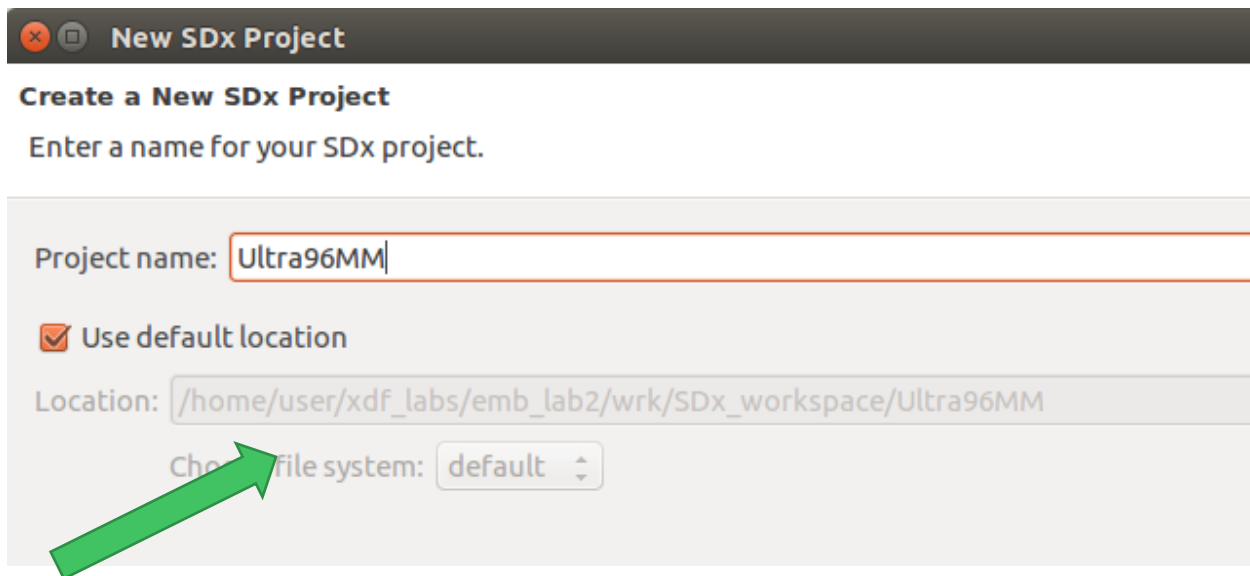


Figure 10 – Project Type Selection

10. For Project Name, enter **Ultra96MM**
 - b. NOTE: the location of the project



New SDx Project

Create a New SDx Project

Enter a name for your SDx project.

Project name:

☒ Use default location

Location:

Choose file system:

Figure 11 – Project Location

11. Click **Next >**

Here you will see the list of pre-installed Hardware Platforms. These are all Zynq-based boards. Notice that Ultra96 is not in the list.

Platform

Choose a platform for your project

Type: ☒ Platform ☐ Hardware specification (DSA/HDF)

Platforms (6) [Filter](#)

Find:

Name	Board	Family	Part	Version	Vendor
zc702	zc702	zynq	xc7z020	1.0	xilinx
zc706	zc706	zynq	xc7z045	1.0	xilinx
zcu102	zcu102	zynqplus	xczu9eg	1.0	xilinx
zcu104	zcu104	zynqplus	xczu7ev	1.0	xilinx
zcu106	zcu106	zynqplus	xczu7ev	1.0	xilinx
zed	zed	zynq	xc7z020	1.0	xilinx

Figure 12 – Pre-installed Hardware Platforms in SDx 2018.2

12. Skip this step if the platforms folder already exists (Experiment 1, step 3):

- a. Create directory `/home/user/xd_f_labs/emb_lab2/wrk/platforms`
- b. Now unzip the `u96_avnet.zip` archive located at `/home/user/xd_f_labs/emb_lab2/src` into the repository at `/home/user/xd_f_labs/emb_lab2/wrk/platforms`

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

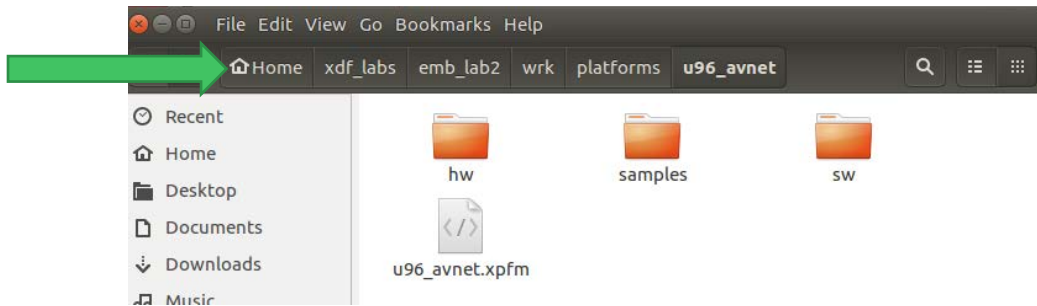


Figure 13 – SDx u96_avnet Installed to Repository

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

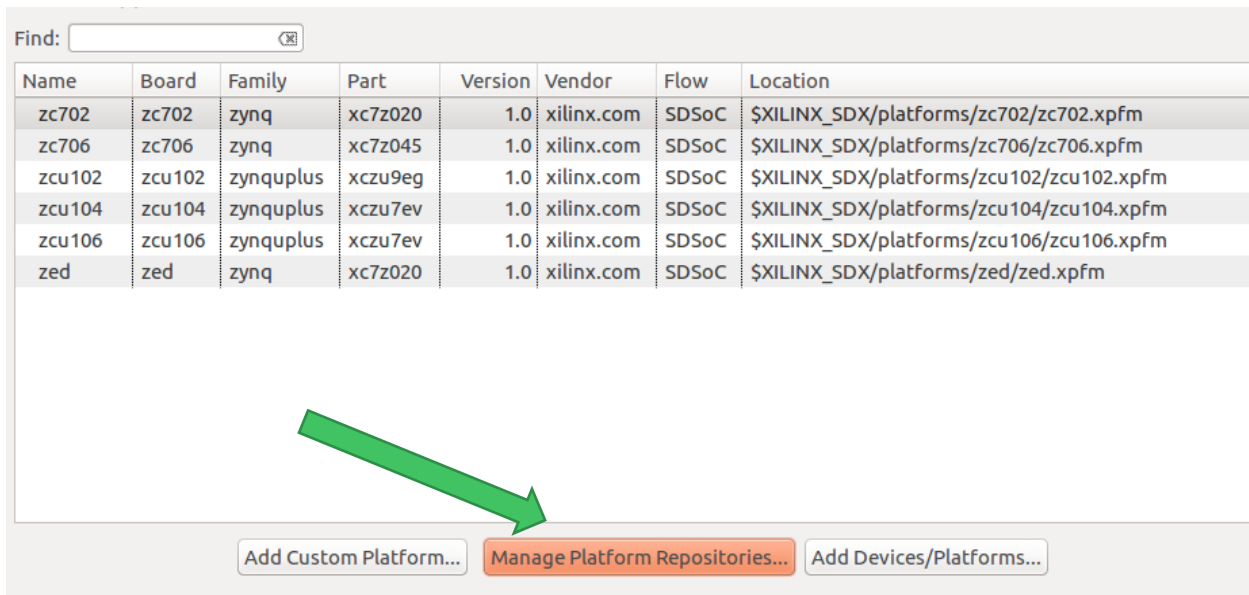


Figure 14 – Manage Repositories

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

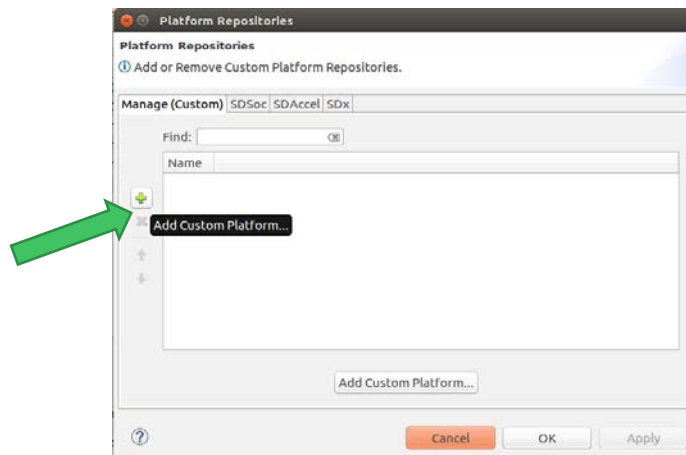


Figure 15 – Add Custom Platform Location

15. Navigate to /home/user/xdp_labs/emb_lab2/wrk/platforms. Select **platforms** and click **OK**.

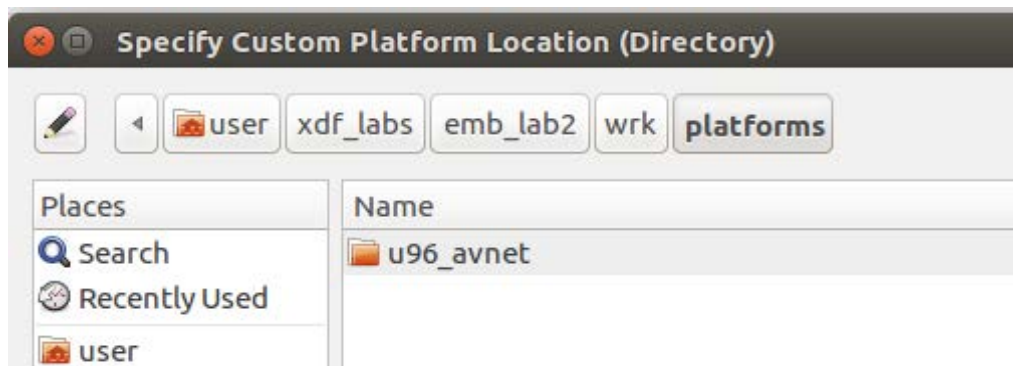


Figure 16 – Specify Custom Platform Location

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

Notice that in the **Choose Hardware Platform** is updated with the **u96_avnet (custom)** Platform as a selection, targeting Part **xczu3eg**.

Platforms (7) [Filter](#)

Find:


Name	Board	Family	Part	Version	Vendor	Flow	Location
 u96_avnet [custom]	av	zynqplus	xczu3eg	1.0	em.avnet.com	SDSoC	/home/user
zc702	zc702	zynq	xc7z020	1.0	xilinx.com	SDSoC	\$XILINX_SE
zc706	zc706	zynq	xc7z045	1.0	xilinx.com	SDSoC	\$XILINX_SE
zcu102	zcu102	zynqplus	xczu9eg	1.0	xilinx.com	SDSoC	\$XILINX_SE
zcu104	zcu104	zynqplus	xczu7ev	1.0	xilinx.com	SDSoC	\$XILINX_SE
zcu106	zcu106	zynqplus	xczu7ev	1.0	xilinx.com	SDSoC	\$XILINX_SE
zed	zed	zynq	xc7z020	1.0	xilinx.com	SDSoC	\$XILINX_SE

Figure 17 – u96_avnet [custom] Is Now an Available Platform


You have now installed a custom platform.

Experiment 3: Create the u96_avnet Matrix Multiply Project

17. Choose the u96_avnet platform by selecting **u96_avnet [custom]** then click **Next >** to continue

Platforms (7) Filter

Find:

Name	Board	Family	Part	Version	Vendor	Flow	Location
 u96_avnet [custom]	av	zynqplus	xczu3eg	1.0	em.avnet.com	SDSoC	/home/user
zc702	zc702	zynq	xc7z020	1.0	xilinx.com	SDSoC	\$XILINX_SE
zc706	zc706	zynq	xc7z045	1.0	xilinx.com	SDSoC	\$XILINX_SE
zcu102	zcu102	zynqplus	xczu9eg	1.0	xilinx.com	SDSoC	\$XILINX_SE
zcu104	zcu104	zynqplus	xczu7ev	1.0	xilinx.com	SDSoC	\$XILINX_SE
zcu106	zcu106	zynqplus	xczu7ev	1.0	xilinx.com	SDSoC	\$XILINX_SE
zed	zed	zynq	xc7z020	1.0	xilinx.com	SDSoC	\$XILINX_SE

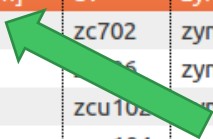


Figure 18 – Choose u96_avnet Hardware Platform

18. You can leave the defaults on the next screen where you need to choose the software platform

Software Platform

System configuration:

Runtime:

Domain

Domain:

CPU: psu_cortexa53_0

OS: standalone

Additional Settings

Output type

☒ Executable (elf) ☐ Static Library ☐ C-Callable Library

☒ Allow hardware accelerations

Figure 19 – Software Platform

This is the selection window where, if your platform has been configured for PetaLinux, or any other operating system selections, you can select them. For example, with PetaLinux you can choose a Static Library or an Executable, allowing your Linux programs to have access to the accelerated functions. While this is also true for a standalone configuration, it is much easier to just have the function `main()` call the acceleration automatically. This was demonstrated in Experiment 1.

19. Click **Next >** to continue.

Templates

Select a template to create your project.

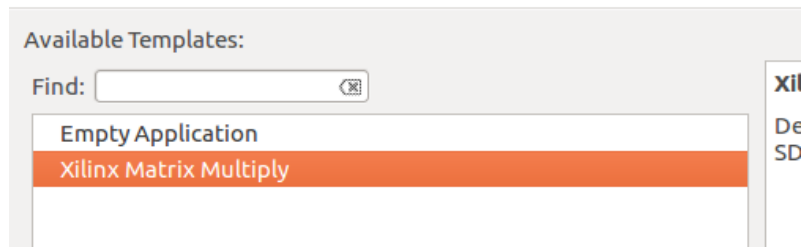


Figure 20 – Example Selection

20. Select the provided Xilinx Matrix Multiply example as shown in Figure 20.

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

Now the SDx Project Explorer will have the Ultra96MM from which we can generate SDSoc projects.

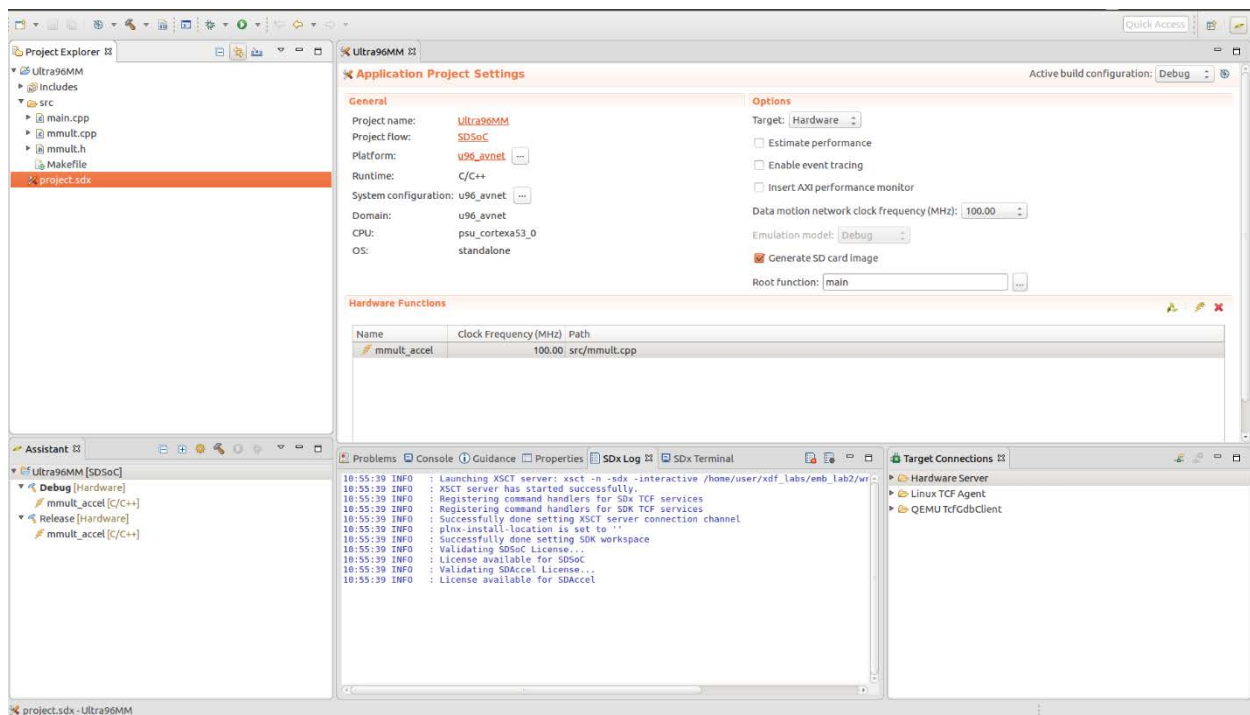


Figure 21 – Ultra96MM added to SDx Project Explorer

For the purposes of this experiment, we will use Matrix Multiply Example code. The sample template automatically accelerates the `mmult_accel` function, which is currently set to run at 100MHz.

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.

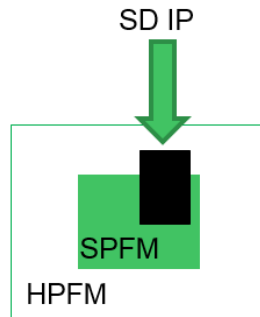


Figure 22 – Target Code is Black Box

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, we are up and running that fast!

Instructions on how to create your own Sample Template is located in User Guide, UG1146.

If you need to import files, the process is the same as with Xilinx SDK.

22. From here, if you have time, you can right click the project and say “Build Project” or click the Hammer in the upper mid-left of the tool bar

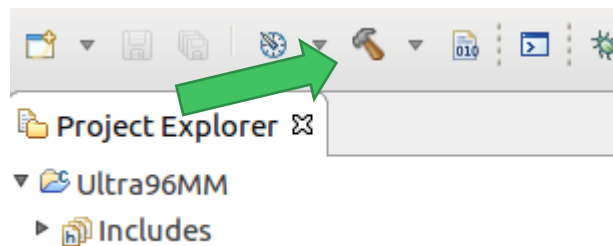


Figure 23 – Build Hammer

23. 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

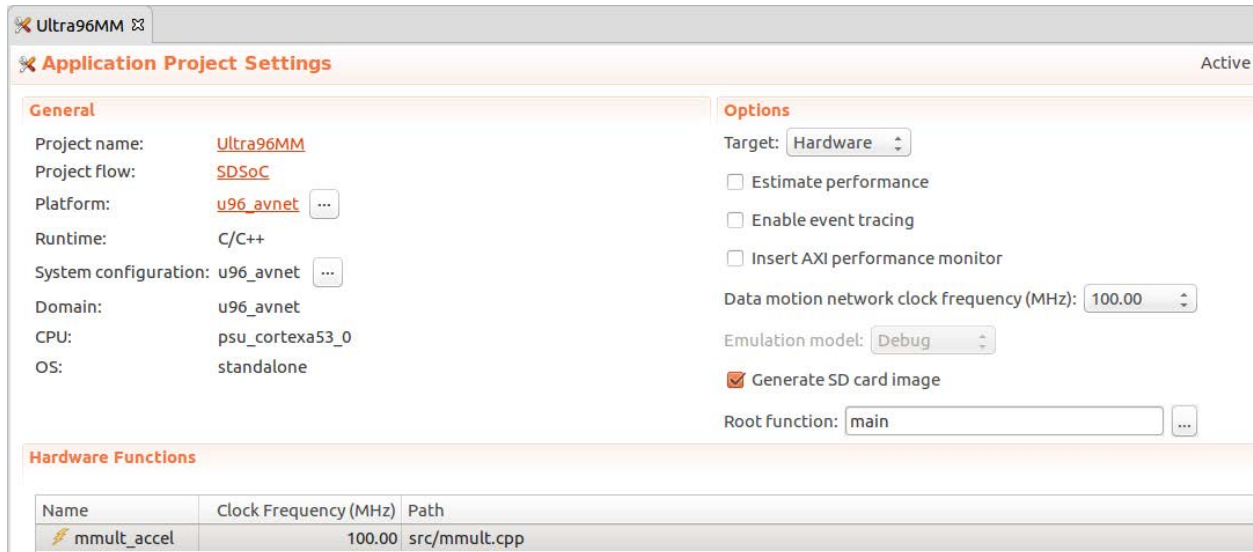


Figure 24 – Project Settings

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.

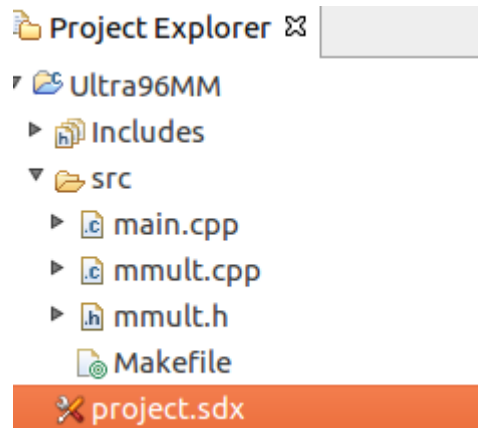


Figure 25 – Ultra96MM Sources

24. Notice that SDSoC generates a layout that looks very similar to the Xilinx SDK. This was a design decision to ensure a familiarity to the tools including the process flow.

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 analyses 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 or was provided as part of the platform

25. More notes regarding the files and the file structure

- a. main.cpp runs the functions
- b. mmult.cpp has pragmas listed
- c. mmult.h has pragmas listed

26. Once the build completes, you can repeat the Experiment 1 steps, replacing the boot.bin from the wrk folder with the file you generated at

```
/home/user/xd_f_labs/emb_lab2/wrk/SDx_workspace/Ultra96MM/Debug/sd_card
```

27. If you still have time, explore the Vivado project by opening the Vivado project at

```
/home/user/xd_f_labs/emb_lab2/wrk/SDx_workspace/Ultra96MM/Debug/_sds/p0/  
Vivado/prj
```