



A Practical Guide to Getting Started with Xilinx SDSoC

Tools:	2019.1
Training Version:	v5
Date:	25 October 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.

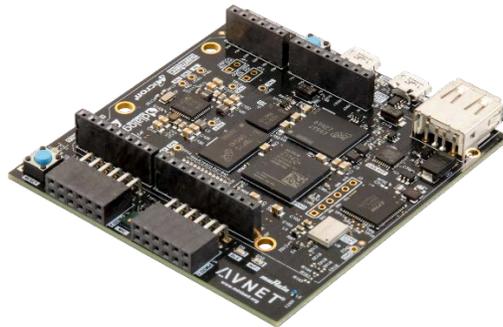
NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You 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

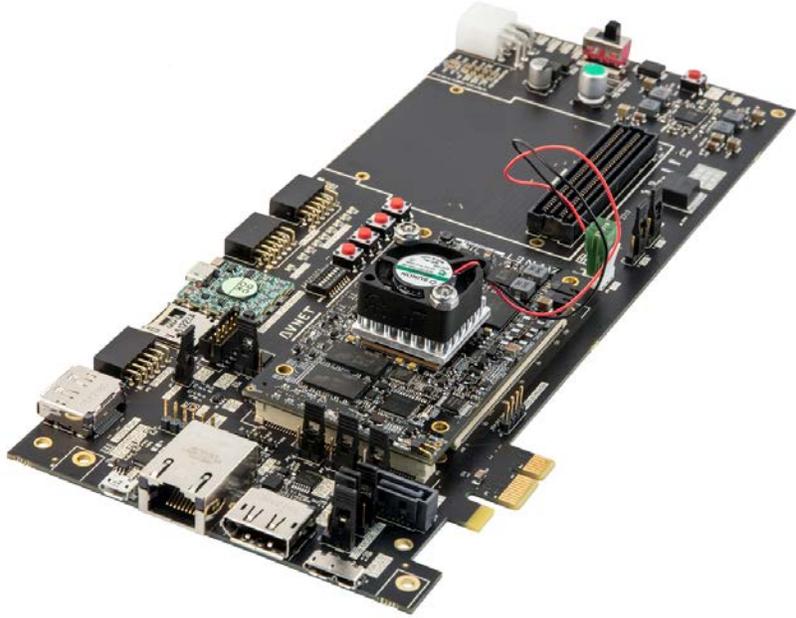
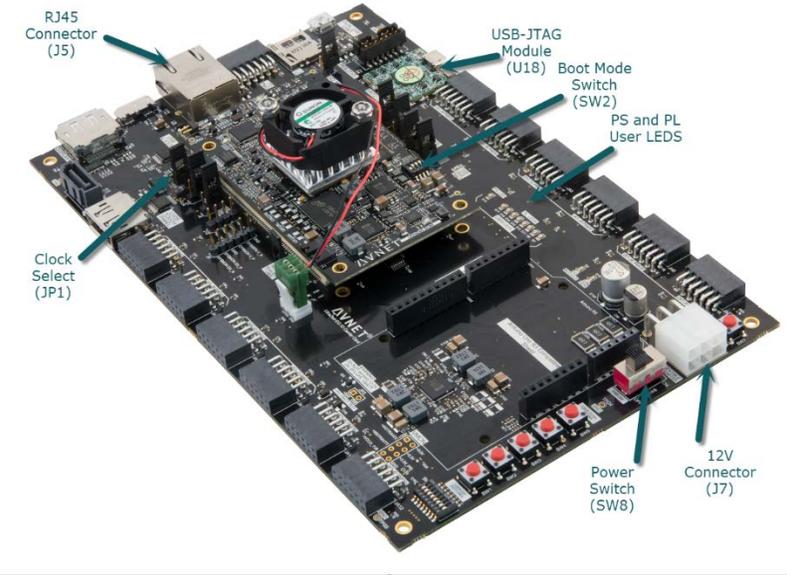
MiniZed™ is a Zynq® 7Z007S single-core development board. With the advent of the latest cost-optimized portfolio from Xilinx, this board targets entry-level Zynq developers with a low-cost prototyping platform.



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 (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 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 Ultra96 development platform. This Technical Training Course is called "A Practical Guide to Getting Started with Xilinx SDSoC".

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, such as 7-zip.

Experiment 1: Install the Pre-Built SDx Hardware Platform

(mz_avnet, u96v1_avnet, u96v2_avnet, ultra96v2_oob, uzegiocc_avnet, uzegpcie_avnet, uzev_avnet)

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 **<developmentBoard>_avnet** or **<developmentBoard>_oob** 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

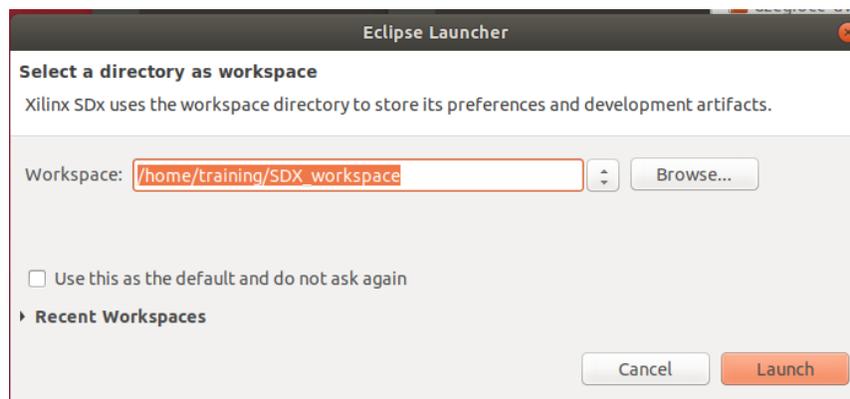
We'll start this experiment by creating a new project so that you can see the pre-installed platforms. Then you will add the Avnet platforms.

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

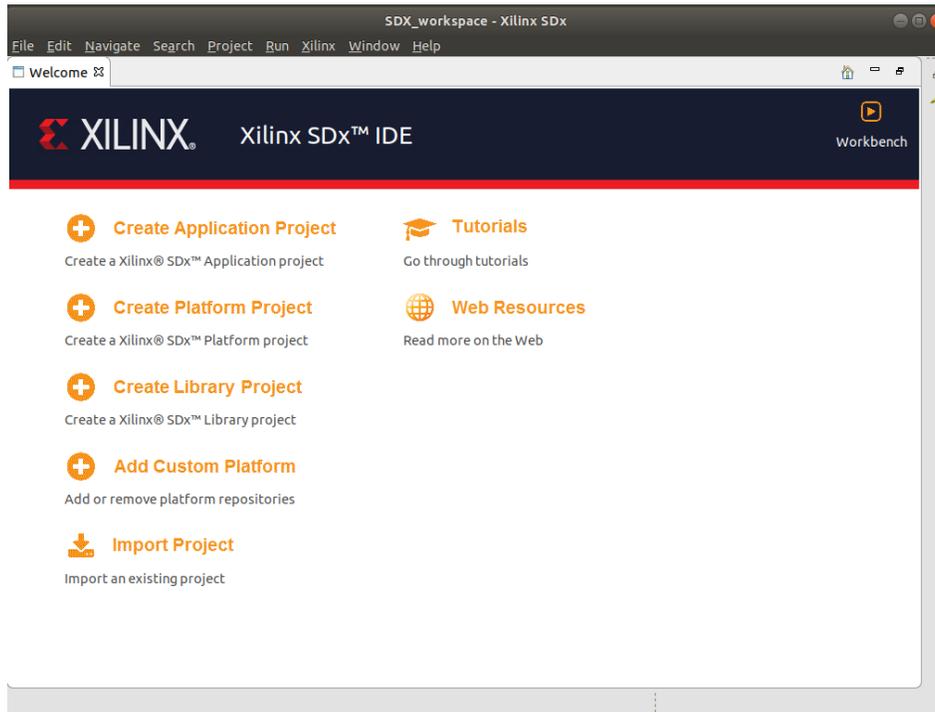


2. SDSoc requires a workspace. Due to Windows path length constraints. It is important that this path be short. Please enter this specific path for consistency through these labs and then click **Launch**.

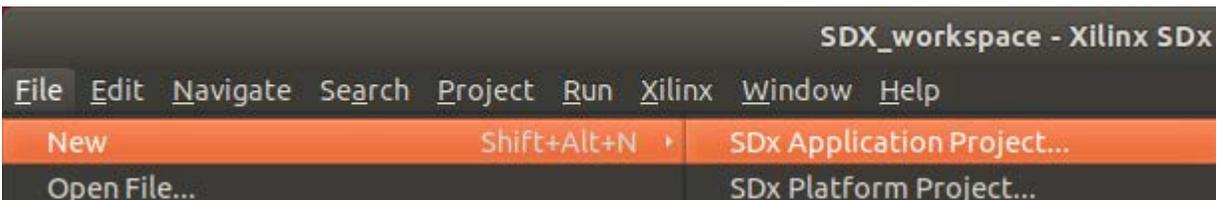
`/home/training/SDx_workspace`



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 **MINIZEDMM, U96V1MM, U96V2MM, U96V2OOBMM, 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:

Use default location

Location:

Choose system:



6. Click **Next >**

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

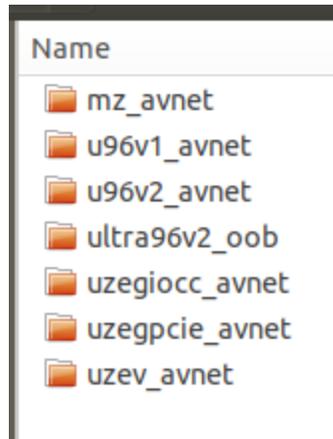
Platform

Choose a platform for your project

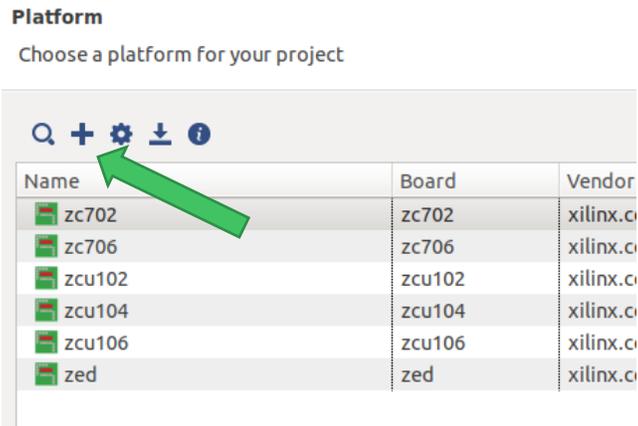
Name	Board	Vendor
 zc702	zc702	xilinx.c
 zc706	zc706	xilinx.c
 zcu102	zcu102	xilinx.c
 zcu104	zcu104	xilinx.c
 zcu106	zcu106	xilinx.c
 zed	zed	xilinx.c

7. Open Explorer and create the directory `~/platforms`
8. Now copy the **platform** folder from the `SDSoC_v2019p1.zip` archive into the repository at `~/platforms`

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

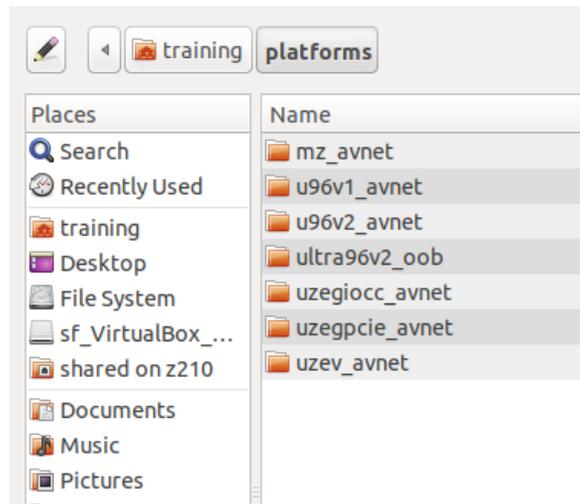


9. In the SDx New Project dialog, click on the **+** button



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

Select **platforms** and click **OK**



11. Notice that now the Platform List is updated.

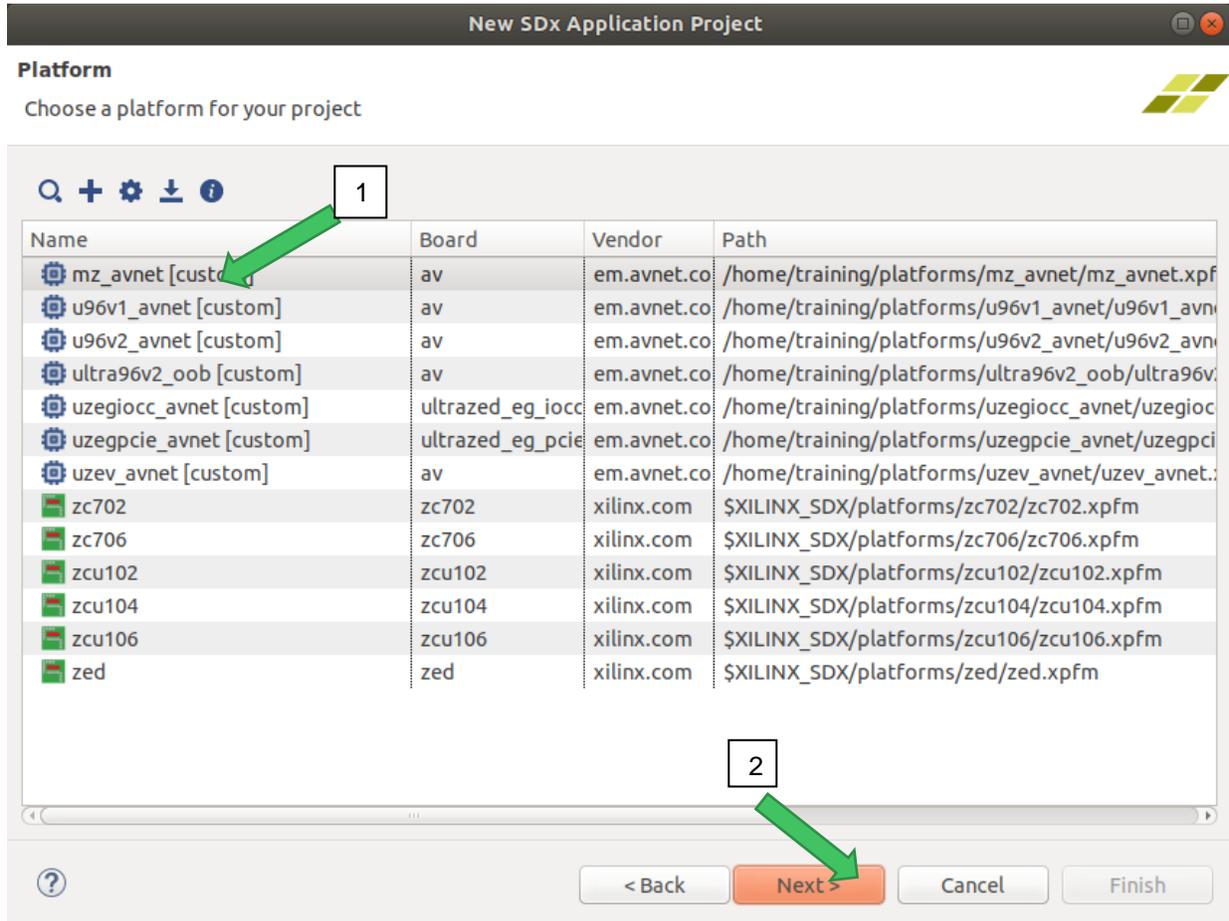
Notice that in the **Choose Hardware Platform** is updated with the custom platforms and noted with a [custom] tag

Name	Board	Vendor	Path
mz_avnet [custom]	av	em.avnet.co	/home/t
u96v1_avnet [custom]	av	em.avnet.co	/home/t
u96v2_avnet [custom]	av	em.avnet.co	/home/t
ultra96v2_oob [custom]	av	em.avnet.co	/home/t
uzegeiocc_avnet [custom]	ultrazed_eg_iocc	em.avnet.co	/home/t
uzegepcie_avnet [custom]	ultrazed_eg_pcie	em.avnet.co	/home/t
uzev_avnet [custom]	av	em.avnet.co	/home/t
zc702	zc702	xilinx.com	\$XILINX
zc706	zc706	xilinx.com	\$XILINX
zc702	zc702	xilinx.com	\$XILINX

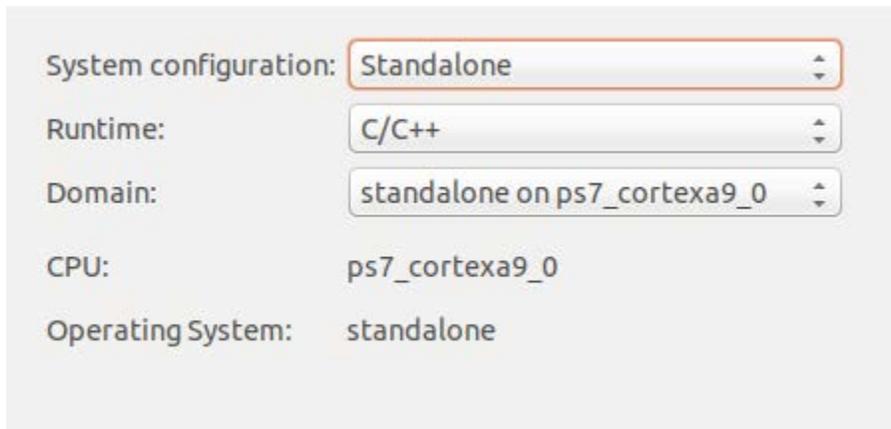
You have now installed Avnet's custom platforms.

Experiment 2: Create the mz_avnet Matrix Multiply Project

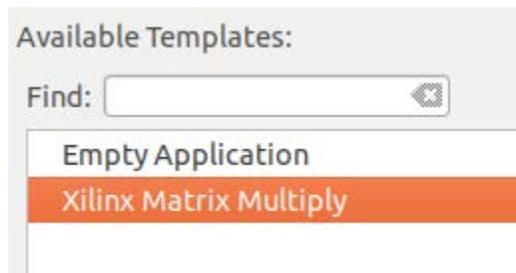
1. Now, we can choose the platform you are targeting, here we choose the mz_avnet platform by selecting **mz_avnet [custom]** then click **Next >** to continue. You can also choose one of the other Avnet platforms.



2. Please validate the default values per the below screen capture.

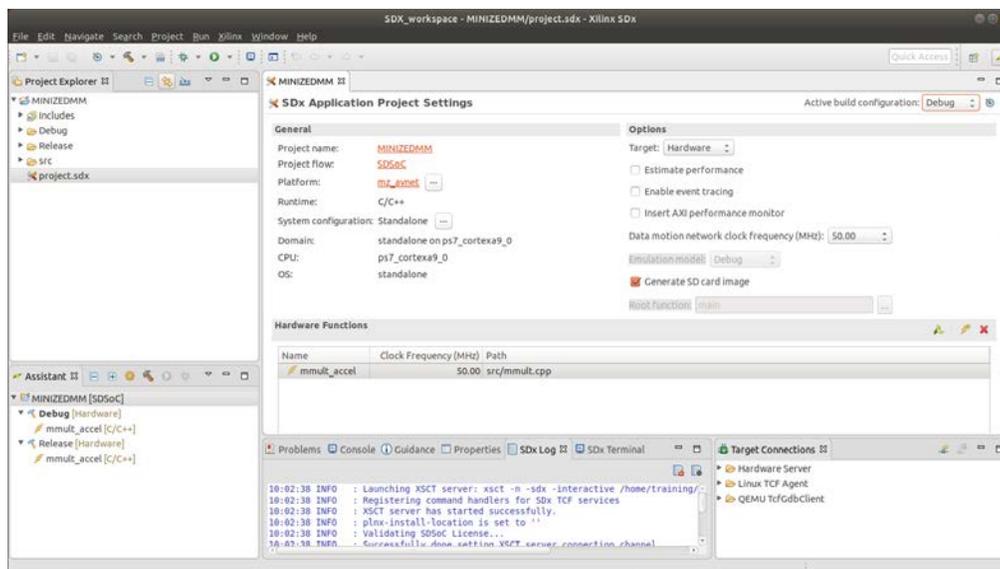


3. Click **Next >** to continue.
4. Select the provided Xilinx Matrix Multiply example as shown below



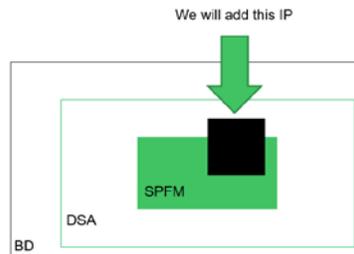
5. Click **Finish** to create a new Empty application.

Now the SDx Project Explorer will have the MiniZedMM 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 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.



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. To learn how to import the matrix multiply files from within the SDSoC installation see the Appendix.

Experiment 3: Build the mz_avnet Matrix Multiply Project

1. Next, back in SDx Project Explorer right click the MiniZedMM project name, click on **Build Project**, remember, you can follow this same flow with any of the provided platforms
2. 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

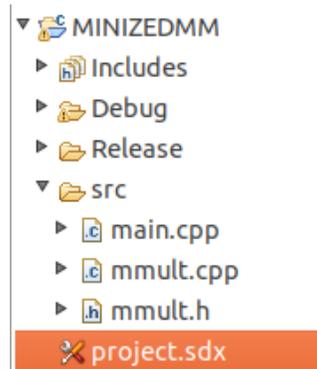
Name	Clock Frequency (MHz)	Path
mmult_accel	50.00	src/mmult.cpp

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.

NOTE: Later versions of UG1028 are maintained as a project on the Xilinx WIKI

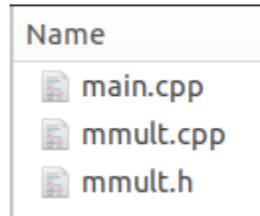
3. Notice that SDSoC generates a layout that looks very similar to the Xilinx SDK. This was a design decision to ensure a familiarity to the tools including the process flow



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 as part of the platform

4. Navigate to `~/SDx_workspace/MINIZEDMM/src`. Notice the 3 source files there.
Note: Swap the bolded 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.cpp has pragmas listed
- Mmult.h has pragmas listed

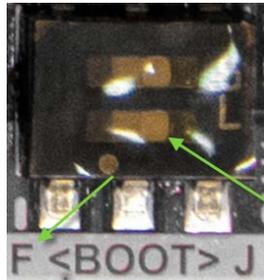
Experiment 4: Set Up Your MiniZed

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

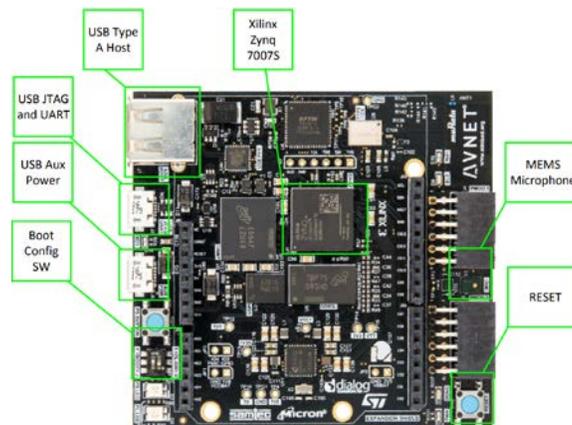
1. First configure the boot jumper. For non-MiniZed kits, refer to the Quick Start Card or Manual for how to configure the Kit for SDCARD booting. If not using a MiniZed, skip to the next step.

For MiniZed, you can select between FLASH and JTAG booting. We want to ensure switch 1 is set towards the F or PS_Button.

- a. Note: From the Factory the switch's protective film should be removed and already set to F. If it is not, the switch will look similar to the below



- b. If your MiniZed boot configuration switch is similar to the above, remove the protective film and slide switch 1 (indicated by the silkscreen DASH above the F) to be toggled to FLASH Booting (F).
2. Next plug the Development Kit into your PC in order to register the board with a COM port
 - a. Note: Windows 10 has been known to create two COM ports when plugging the MiniZed into the PC.
 - b. Note: Ultra96 will need the Debug Adapter Board, refer to the Manual for more information
 - c. For a MiniZed, with a factory fresh board, open two instances of Tera Term, one for each COM port; 8,N,1,115200
 - d. Reboot your MiniZed using the Reset button



Experiment 5: Run the Design

1. Insert one USB cable into the MiniZed USB JTAG/UART MicroUSB connector
 - a. For other kits, plug one MicroUSB cable into the UART USB connector
2. Plug the other end into your PC
3. Open GTKTerm terminal program. If you have not already, configure it to connect to the COM Port we found during setup of the VirtualBox Installation Guide, using 115200/8/n/1/n as settings.
 - a. NOTE: if you execute `ls /dev/ttyUSB*` you will get a listing of the USB terminal devices you can attempt to access

```
training@training-VirtualBox:~/projects$ ls /dev/ttyUSB*  
/dev/ttyUSB0 /dev/ttyUSB1  
training@training-VirtualBox:~/projects$
```

If this is the first time you have powered up your MiniZed, you will see a LOT of text as the default factory boot image is PetaLinux.

SDSoC builds BOOT.BIN files. While the intention is to be able to use these with SDCARD boot, such as with devices like Ultra96 and UltraZed-EV, this is just a standard BOOT.BIN file and can be treated as such for devices like MiniZed, which do not have an SDCARD cage.

4. For products with an SDCARD cage, skip to step 11
5. Using a explorer, navigate to the below and inspect for files. They will appear once the build is complete
`~/SDX_workspace/MINIZEDMM/Debug/sd_card/`
6. Once the build is complete, in the VirtualBox Installation Guide configured **TERMINAL**, change the working directory to the location of BOOT.BIN

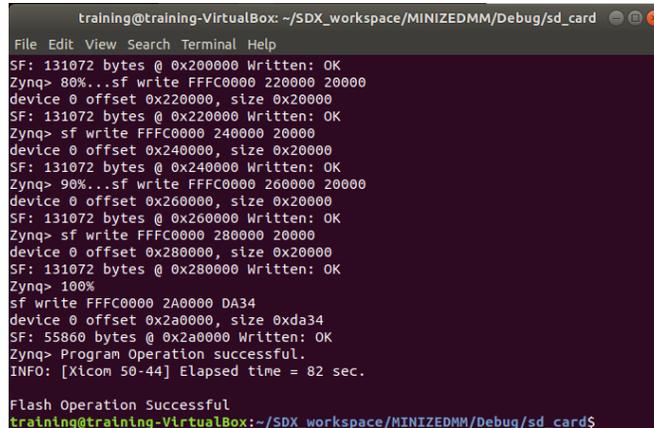
```
training@training-VirtualBox: ~/SDX_workspace/MINIZEDMM/Debug/sd_card  
File Edit View Search Terminal Help  
training@training-VirtualBox:~/projects$ cd ~/SDX_workspace/MINIZEDMM/Debug/sd_card/  
training@training-VirtualBox:~/SDX_workspace/MINIZEDMM/Debug/sd_card$
```

```
cd ~/SDX_workspace/MINIZEDMM/Debug/sd_card/
```

- Once at the \$ prompt, execute the command to program the QSPI over JTAG in the MiniZed.

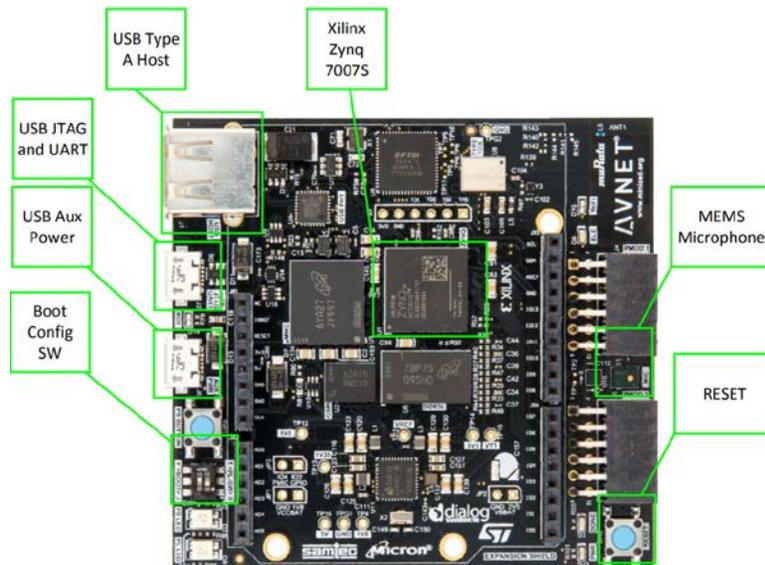
```
program_flash -f BOOT.BIN -fsbl  
~/platforms/mz_avnet/sw/sysconfig1/boot/fsbl.elf -flash_type qspi-  
x4-single
```

- Notice the blue DONE LED turns off. This indicates the image is being programmed, do not interrupt this process!
- When complete, you should see the message, "Flash Operation Successful"



```
training@training-VirtualBox: ~/SDX_workspace/MINIZEDMM/Debug/sd_card  
File Edit View Search Terminal Help  
SF: 131072 bytes @ 0x200000 Written: OK  
Zynq> 80%..sf write FFFC0000 220000 20000  
device 0 offset 0x220000, size 0x20000  
SF: 131072 bytes @ 0x220000 Written: OK  
Zynq> sf write FFFC0000 240000 20000  
device 0 offset 0x240000, size 0x20000  
SF: 131072 bytes @ 0x240000 Written: OK  
Zynq> 90%..sf write FFFC0000 260000 20000  
device 0 offset 0x260000, size 0x20000  
SF: 131072 bytes @ 0x260000 Written: OK  
Zynq> sf write FFFC0000 280000 20000  
device 0 offset 0x280000, size 0x20000  
SF: 131072 bytes @ 0x280000 Written: OK  
Zynq> 100%  
sf write FFFC0000 2A0000 DA34  
device 0 offset 0x2a0000, size 0xda34  
SF: 5860 bytes @ 0x2a0000 Written: OK  
Zynq> Program Operation successful.  
INFO: [Xicom 50-44] Elapsed time = 82 sec.  
Flash Operation Successful  
training@training-VirtualBox:~/SDX_workspace/MINIZEDMM/Debug/sd_card$
```

- It is advised to NOT leave your terminal IN the same folder as the BOOT.BIN image (/debug/sd_card), in some cases, if you were to rebuild the SDx tools will COMPLETE all building, BIT FILE, HLS IPI generation, etc., and JUST as it is ready to combine into a boot.bin, it can fail the build as in some cases, the terminal will hold the directory while the SDx tool tried to delete the folder – thus failing after the 20 minute+ build. Therefore, closing the terminal or changing back to a safe folder (~/projects) is critical.
- Once you get the successful message, press the RESET button on the MiniZed



Reset Button

10. The BLUE Done should light up again and you will see after each press of the reset, the Matrix Multiply will run.
11. For MiniZed, skip this step. For other products with an SDCARD cage, copy the `BOOT.BIN` file to your included SDCARD. The file is located:
`~/SDx_workspace/[project_name]/Debug/sd_card`
12. For MiniZed, skip this step. For other products with an SDCARD cage, insert the SDCARD into the SDCARD cage. Turn on the power and you should observe a similar display, as shown in the corresponding image below.
 - a. Note that with MiniZed, due to the reduced resources in a Zynq ZC7007s, the Matrix Multiply has been reduced to a 24x24 array. For other products, you will see a 32x32 array
 - b. Note that the acceleration that MiniZed provides is based on a 50MHz clock. The other products utilize a 100MHz clock. Acceleration results will be higher on these platforms.
13. You can now CLOSE the SDx IDE.

```

COM5 - Tera Term VT
File Edit Setup Control Window Help
Testing 1024 iterations of 24x24 floating point mmult...
Average number of CPU cycles running mmult in software: 527166
Average number of CPU cycles running mmult in hardware: 34742
Speed up: 15.1737
TEST PASSED
  
```

Matrix Multiply Running in Both Software and Hardware, MiniZed

```

COM6:115200bps - Tera Term VT
File Edit Setup Control Window Help
Testing 1024 iterations of 32x32 floating point mmult...
Average number of CPU cycles running mmult in software: 1579746
Average number of CPU cycles running mmult in hardware: 65870
Speed up: 23.9828
TEST PASSED
  
```

Matrix Multiply Running in Both Software and Hardware, Ultra96v1

```

COM9:115200baud - Tera Term VT
File Edit Setup Control Window Help
Testing 1024 iterations of 32x32 floating point mmult...
Average number of CPU cycles running mmult in software: 1579744
Average number of CPU cycles running mmult in hardware: 65901
Speed up: 23.9715
TEST PASSED
  
```

Matrix Multiply Running in Both Software and Hardware, Ultra96v2

```
COM9:115200baud - Tera Term VT
File Edit Setup Control Window Help
Testing 1024 iterations of 32x32 floating point mmult...
Average number of CPU cycles running mmult in software: 1579740
Average number of CPU cycles running mmult in hardware: 65374
Speed up: 24.1647
TEST PASSED
```

Matrix Multiply Running in Both Software and Hardware, Ultra96v2 OOB Based

```
COM18:115200baud - Tera Term VT
File Edit Setup Control Window Help
Xilinx Zynq MP First Stage Boot Loader
Release 2019.1 Oct 25 2019 - 09:05:04
Testing 1024 iterations of 32x32 floating point mmult...
Average number of CPU cycles running mmult in software: 1579820
Average number of CPU cycles running mmult in hardware: 58497
Speed up: 27.0069
TEST PASSED
```

Matrix Multiply Running in Both Software and Hardware, UltraZed-EG IOCC

```
COM14:115200baud - Tera Term VT
File Edit Setup Control Window Help
Xilinx Zynq MP First Stage Boot Loader
Release 2019.1 Oct 25 2019 - 09:10:01
Testing 1024 iterations of 32x32 floating point mmult...
Average number of CPU cycles running mmult in software: 1579821
Average number of CPU cycles running mmult in hardware: 58531
Speed up: 26.9912
TEST PASSED
```

Matrix Multiply Running in Both Software and Hardware, UltraZed-EG PCIECC

```
COM16:115200baud - Tera Term VT
File Edit Setup Control Window Help
Xilinx Zynq MP First Stage Boot Loader
Release 2019.1 Oct 25 2019 - 09:00:41
Testing 1024 iterations of 32x32 floating point mmult...
Average number of CPU cycles running mmult in software: 1579817
Average number of CPU cycles running mmult in hardware: 54284
Speed up: 29.1028
TEST PASSED
```

Matrix Multiply Running in Both Software and Hardware, on UltraZed-EV

Appendix A: Importing Files

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. This is only necessary if you are importing files instead of using the example built into the provided platform.

1. Right click on the MiniZedMM src folder and select **Import...**

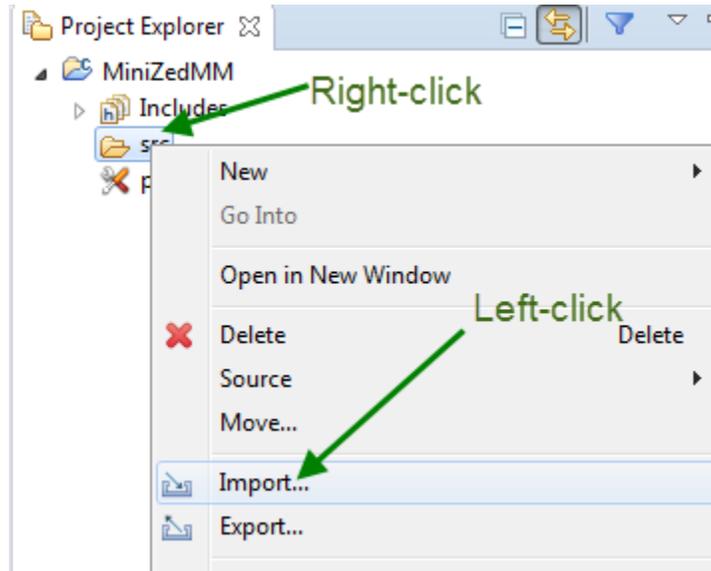


Figure 1 – Import Source Files

2. Select **General**, then **File System** and **Next >**.

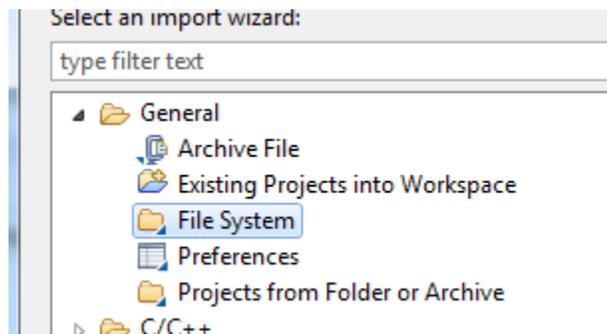


Figure 2 – Import from the File System

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

```
/tools/Xilinx/SDx/2019.1/samples/mmult
```

- Next select three checkboxes next to the .cpp and .h files. Ensure that the **Into folder** box shows MiniZedMM/src as well as the checkboxes matches the screen in the below figure.

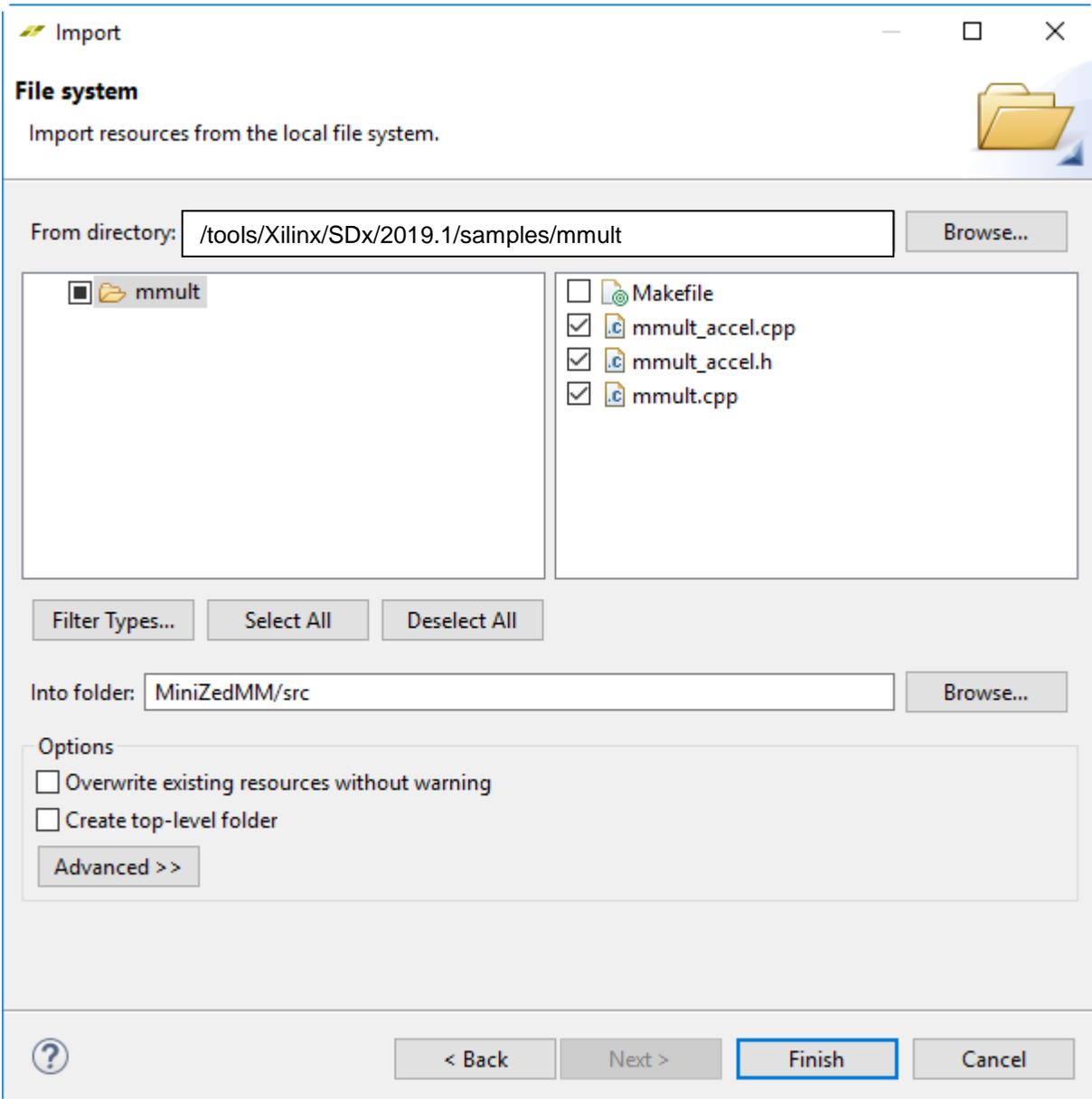


Figure 3 – Import Example Files

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

Revision History

Date	Version	Revision
25 Aug 17	00	Preliminary release
28 Aug 17	00	Edits per Author
17 Sep 17	1	First public release
19 Apr 18	2	Updated to SDSoC 2017.4
18 Aug 18	3	Updated to SDSoC 2018.2, added UltraZed-EV and Ultra96
19 Oct 18	3.1	Updated for new Ultra96 BDF name
25 Oct 18	3.2	Added in UltraZed-EG IOCC and PCIECC
27 Feb 19	4	Updated to SDSoC v2018.3
25 Oct 19	5	Overhaul to use Ubuntu as host machine and 2019.1 tools