FMC-MULTI-CAM4
reVISION 2018.2 Tutorial

Introduction

The Avnet Multi-Camera FMC module supports up to four (4) high definition camera modules using MAXIM Integrated' s GMSL (Gigabit Multimedia Serial Link) technology.

GMSL is widely used in the automotive industry for in-vehicle high speed communication of video streams. Making use of low-cost coax cable up to 15 meters in length, GMSL meets the most stringent electromagnetic compatibility (EMC) requirements of the automotive industry.

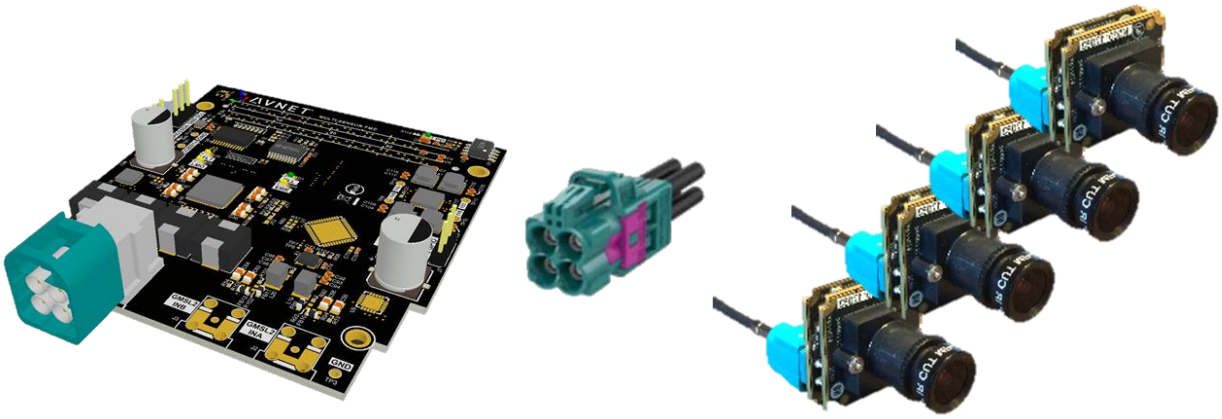
The provided linux-based reference designs support the ON Semiconductor MARS (Modular Automotive Reference System) based camera modules, including the MAXIM Integrated MAX96705 GMSL Serializer board and the ON Semiconductor AR0231AT image sensor board.

The AR0231AT image sensor is an automotive-grade image sensor which uses the latest 3.0 micron Back Side Illuminated (BSI) pixel with ON Semiconductor's DR-Pix™ technology, which offers dual conversion gain for improved performance under all lighting conditions. It includes LED Flicker Mitigation (LFM), and achieves 120dB of High-Dynamic Range with 4-exposure HDR.

The Multi-Camera FMC Module is available in two options:

- FMC module only (without any cable or cameras)
- Quad AR0231AT Camera FMC bundle (including cable and four cameras)

The Quad AR0231AT Camera FMC Bundle includes the Multi-Camera FMC module, a cable assembly, as well as four AR0231AT camera modules with GMSL serialization.



The Quad AR0231AT Camera FMC Bundle is meant to be used with Xilinx Zynq-UltraScale+ FMC carriers, including the ZCU102, the ZCU104, as well as the Avnet UltraZed EV SOM + Carrier. The Quad Camera FMC Bundle is fully integrated to the Xilinx reVISION stack, including SDSoC platforms and design examples.

Overview

This tutorial integrates the Multi-Camera FMC into the reVISION stack by providing an SDSoC platform for various Zynq-UltraScale+ MPSoC based FMC carriers. The reVISION / SDSoC platform provides a feature rich framework for the development of video applications on the Xilinx Zynq UltraScale+ MPSOC. It is based on the original reVISION platform from Xilinx, which provides the following infrastructure:

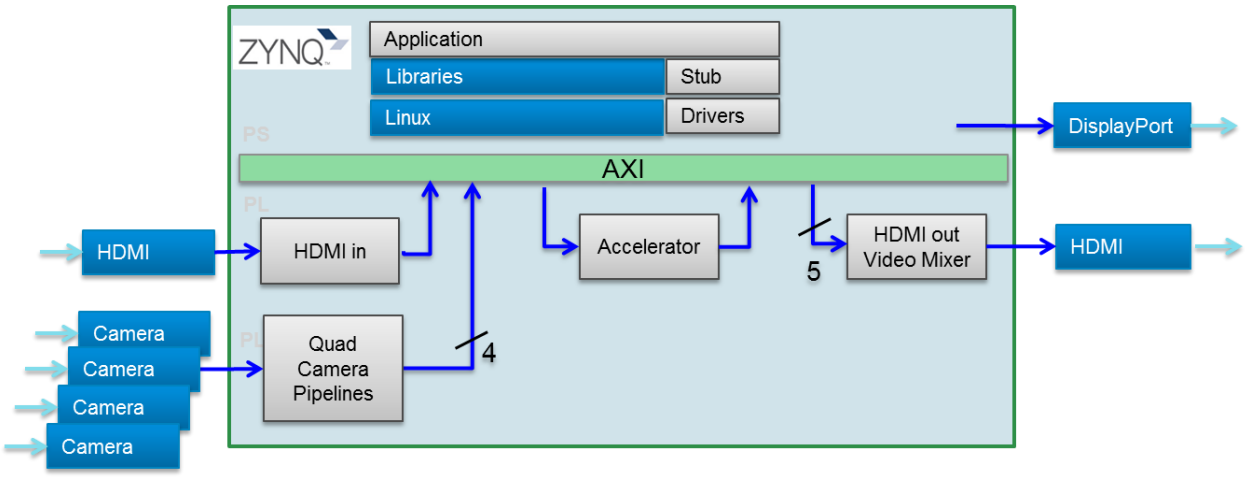
- video sources (or capture pipelines)
- computer vision accelerators implemented as memory-to-memory (m2m) pipelines
- video sinks (or output/display pipelines)

The following documentation should be consulted for more detailed information on the original Xilinx reVISION platform:

https://github.com/Xilinx/Revision-Getting-Started-Guide

The original reVISION platform was modified to remove single sensor capture pipeline, and replaced with the quad camera capture pipelines.

The following figure illustrates the simplified block diagram for the SDSoC platform.



NOTE : The HDMI capture pipeline is only included in the ZCU102 platform, and excluded from the ZCU104 platform to save resources.

Objectives

This tutorial will guide the user how to:

- Execute a SDSoC sample on hardware
- Rebuild a SDSoC sample design

SDSoC Samples

File I/O

These are the simplest design examples. Typically they will read a frame of video from a standard image file using a standard OpenCV call (such as cv::imread()), process that frame with a call to an xfopencv function, and output the result to a file, (e.g., using cv::imwrite()). They illustrate use of five different xfopencv HW accelerated versions of popular OpenCV functions.

- Bilateral Filter
- Harris Filter
- Dense Optical Flow
- Stereo Vision (Depth Detection)
- Warp Transformation

Live I/O

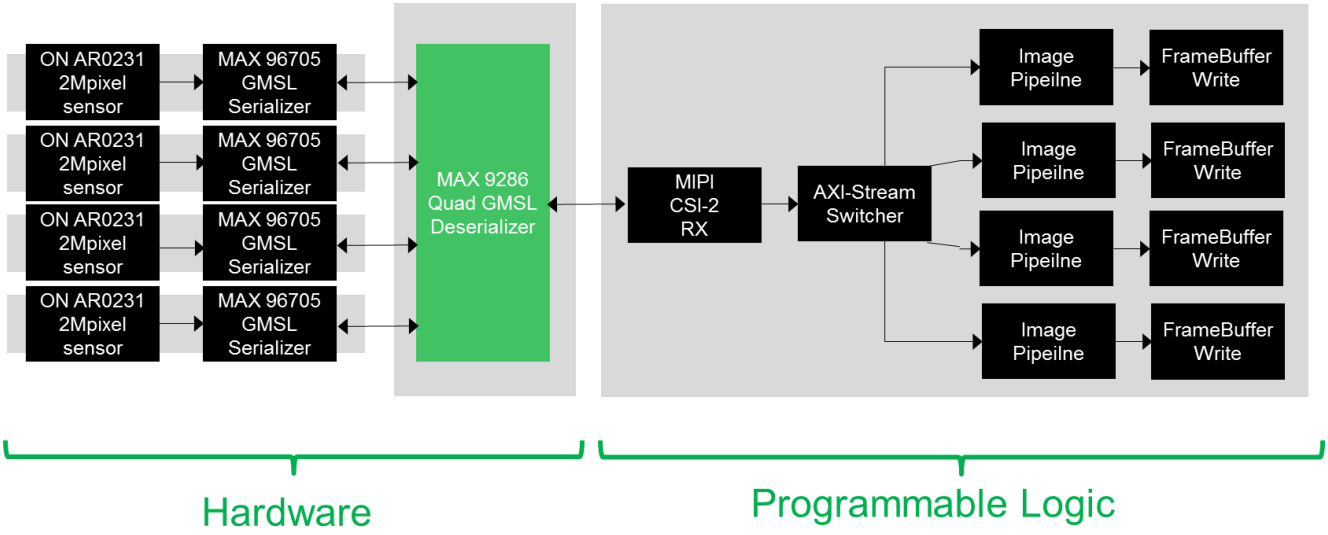
These examples input and output live video.

- Dense Optical Flow - requires Multi-Camera FMC Module + AR0231AT camera modules
    - This algorithm uses two successive images in time, and calculates the direction and magnitude of motion at every pixel position in the image. The calculation is a simple implementation of the Lucas–Kanade method for optical flow estimation. The optical flow algorithm returns two signed numbers at each pixel position, representing up or down motion in the vertical direction, and left or right motion in the horizontal direction. The brightness of the false-color output, from black up to bright color, indicates the magnitude of the motion, and the color indicates the direction.

- Filter2D - - requires Multi-Camera FMC Module + AR0231AT camera modules
    - Convolution is a common image processing technique that changes the intensity of a pixel to reflect the intensities of the surrounding pixels. This is widely used in image filters to achieve popular image effects like blur, sharpen, and edge detection. The implemented algorithm uses a 3x3 kernel with programmable filter coefficients.

Quad Camera Capture Pipelines

The following block diagram illustrates the full quad camera capture pipelines included in this reVISION / SDSoC platform.



The image pipeline for each of the four camera capture pipelines is illustrated in the following block diagram.

NOTE : The Scaler has been excluded in this version of the ZCU102 and ZCU104 platforms to save resources.

Valid licenses (hardware evaluation, or full license) are required for the following video IP cores:

- Demosaic (Color Filter Array Interpolation)
- Video Processing Sub-System, including:
  - RGB to YcrCb Color-Space Converter
  - Chroma Resampler
  - Scaler
- Frame Buffer Write

Experiment Setup

The following sections describe the software and hardware requirements for this tutorial.

Software

The software required to build, and execute the reference design is:

- Linux or Windows host machine with a minimum memory of 32GB
- Terminal Emulator (HyperTerminal or TeraTerm)
- Xilinx SDSoC 2018.2

Hardware

The hardware required to build, and execute the reference design is:

- One of the following supported FMC carriers:
    - ZCU102
    - ZCU104
- Quad AR0231AT Camera FMC Bundle, including:
    - AES-FMC-MULTICAM4-G FMC module
    - Quad-HFM to 4x FAKRA Cable Assembly
    - 4 camera modules. Each composed of:
        - MAX96705 Serializer Kit (MARS1-MAX96705-GEVK)
        - AR0231AT Image Sensor Board (MARS1-AR0231AT7-GEVB)
- Monitor with HDMI input supporting one of the following resolutions:
    - 3840x2160 **or**
    - 1920x1080

Experiment 1: Extract the design files

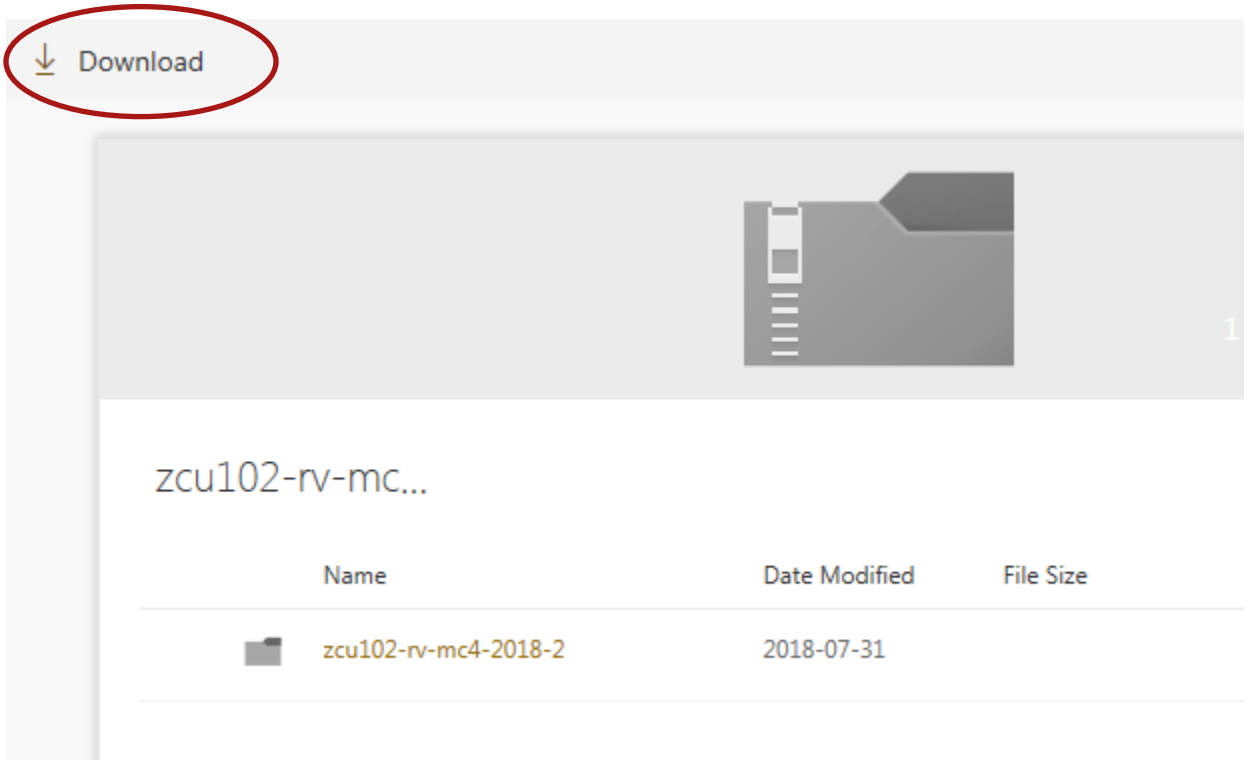
In this section, the design files for the reVISION platform, including pre-built examples, will be extracted to your computer.

1. Download one of the following reVISION platform archives, depending on which Zynq-UltraScale+ carrier you are using.

ZCU102:
zcu102-rv-mc4-2018-2-20180828.zip
https://avtinc.sharepoint.com/:u:/t/ET-Downloads/EZr0t89EN9lDuJd8jXgYJIkBRkTYfh2R4x2OtcfjucTrjA
MD5 SUM Value : 8e3c9ae70d5207f6373bdc170b851959

ZCU104:
zcu104-rv-mc4-2018-2-20180828.zip
https://avtinc.sharepoint.com/:u:/t/ET-Downloads/ERm03RSrZXNOviqUxgdzh3QBRwDtNTzInuY88JMmSr4jGg
MD5 SUM Value : 18e04af5477ef036c73e5fa77d0c841f

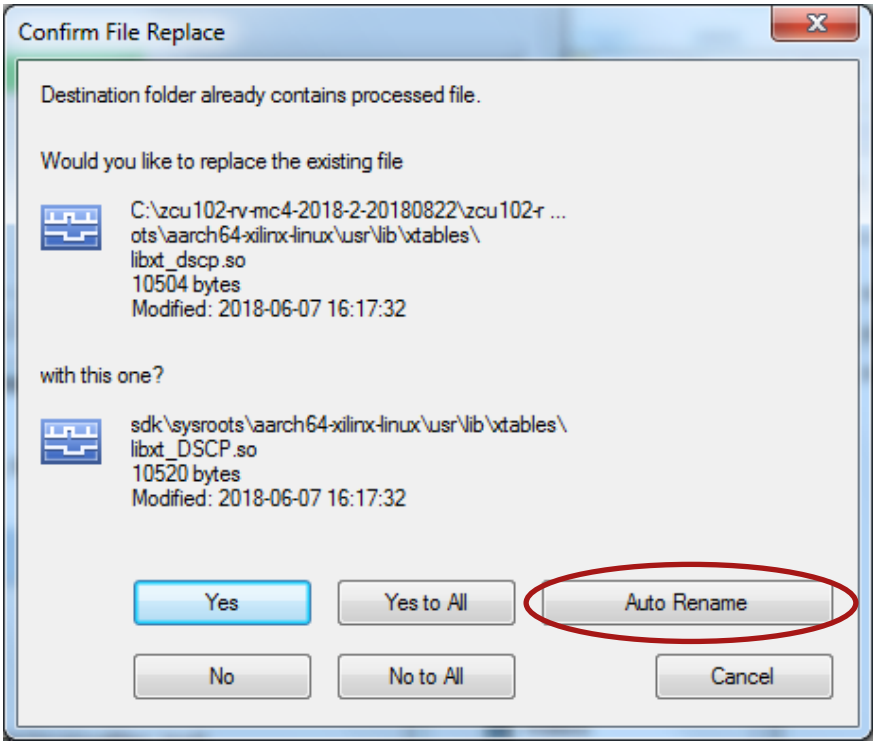
2. Click on the download link, then press the “Download” button on the top of the page.



3. Extract the reVISION platform archive to your computer
    a. For Linux, use the unzip utlity.
    b. For Windows, make sure that the reference design zip file is unzipped in a directory path which contains no spaces. Use the 7zip utility and follow the steps below. If you need 7zip, get it here

        7zip

4. Once the reference design zip file is unzipped, navigate to the petalinux directory and unzip the file sdk.zip which contains the sysroot.
    c. When prompted to confirm file replace, select 'Auto Rename' (Windows only)



Once extracted, you will have a **{target}-rv-mc4-2018-2** directory, where **{target}** will be one of either **zcu102** or **zcu104**. This directory will contain the following sub-directories:

Sub-directory	Content Description
sd_card	contains pre-built SD card images that enable the user to run the live I/O example applications on the ZCU10x board.
{target}_rv_mc4	SDSoC platform for ZCU102 or ZCU104
hw	contains the .dsa file describing the hardware platform.
sw	contains software - bootloaders and other code and support files for the processors on the ZCU10x target board.
samples	contains sample app code. Each sample directory has a .json file describing the build process. These are the SDx sample apps that appear in the "Template" dialog when creating a new project using the reVISION platform.
file_IO	projects are self-contained
live_IO	projects are more complex, and are built in several steps
petalinux	contains a petalinux bsp with device tree info, hardware description files, and other system setup files. An advanced user has the option of creating their own platform. A sysroot is included as zip file.
workspaces	contains a workspace directory structure you may use to build the live_IO samples

Experiment 2: Running the pre-built designs

The following live I/O designs have been pre-built and are ready to run on hardware:

- Filter 2D
- Optical Flow
- Filter 2D + Optical Flow

Setup the ZCU102 or ZCU104 hardware as follows:

1. Connect the FMC-MULTICAM4 FMC module to the FMC connector (J5 for HPC0 FMC on ZCU102, J5 for LPC FMC on ZCU104)
2. Attach the four AR0231AT camera modules to their respective MAX96705 Serializer modules, and connect to the FMC-MULTICAM4 FMC module with the cable assembly
3. Connect the power supply to the 12V connector (J52 on ZCU102, J52 on ZCU104)
4. Connect the HDMI display with an HDMI cable to the top connector of dual HDMI connector (P7 on ZCU102, P7 on ZCU104)
5. Connect a micro-USB cable to the USB-UART connector (J83 on ZCU102, J164 on ZCU104)
   Use the following settings for your terminal emulator:
   - Baud Rate: 115200
   - Data: 8 bit
   - Parity: None
   - Stop: 1 bit
   - Flow Control: None
6. Insert SD card (FAT formatted) with pre-built image copied from one of the following directories:
   - Filter 2D : sd_card/filter2d
   - Optical Flow : sd_card/opticalflow
   - Filter 2D + Optical Flow : **sd_card/filter2d_optflow**
7. Set boot mode to SD card
   - ZCU102 : SW6[4:1]: **off,off,off, on**
   - ZCU104 : SW6[4:1]: **off,off,off, on**
8. Turn on the power switch (SW1 on ZCU102, SW1 on ZCU104)

Once linux has finished booting, you should see the linux prompt on the serial console:

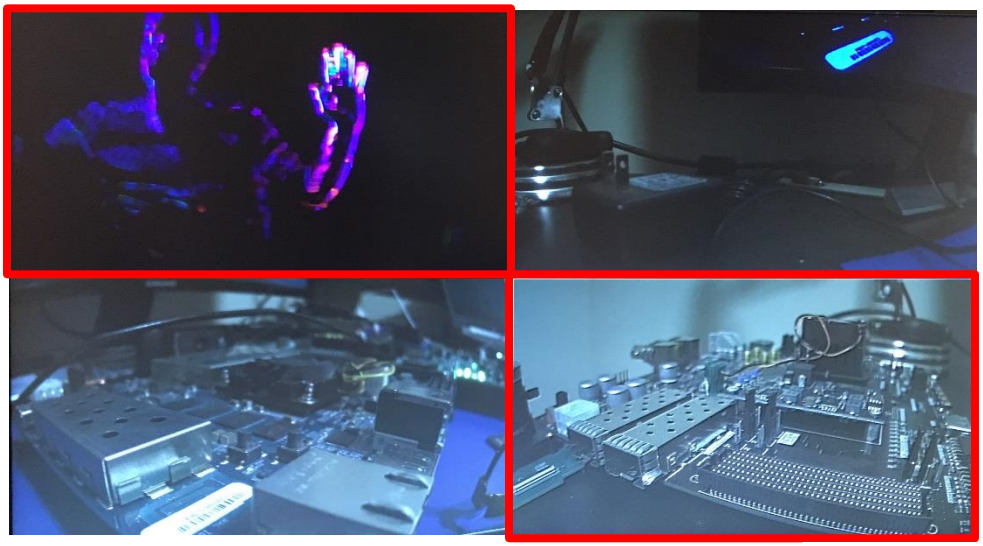
```
…
Setting console loglevel to 0 ...
root@xilinx:~# cd /media/card
root@xilinx:/media/card# ./gstfilter2d_optflow.sh
```

9. Change the active directory to the sd card with the following command : **cd /media/card**
10. Execute the design with the following command : **./gstfilter2d_optflow.sh**

You should observe the live captures from the four cameras on the 4K monitor.

- the top-left quadrant is running the “Optical Flow” accelerator, indicating movement in the scene
- the bottom-right quadrant is running the “Filter 2D” accelerator, enhancing the edges in the scene
- the other two quadrants are running a passthrough

Optical Flow



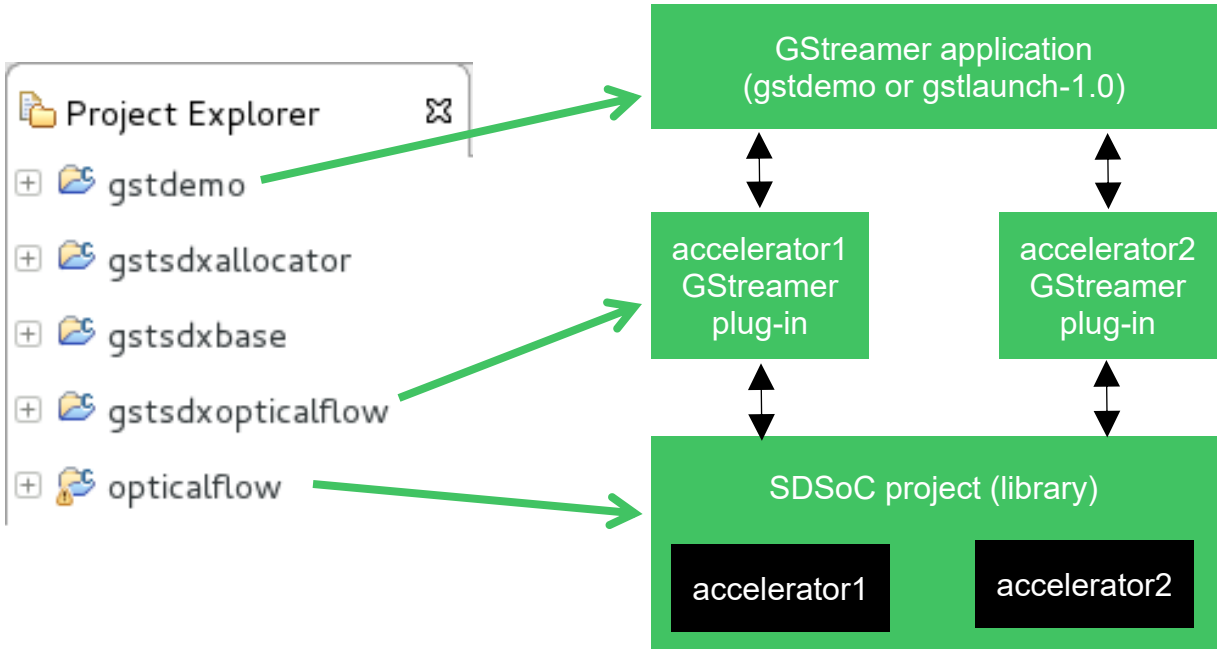
Filter 2D

Experiment 3: Modifying the GStreamer Pipelines

This version of the reVISION platforms make use of the GStreamer framework. The sample design examples are built as GStreamer plugins.

An application (gstdemo) is also included with each sample design to exercise the sample plugins.

As you will see in the next sections, the sample design examples consist of a multi-project workspace, as shown in the following figure.



An alternate method of launching the GStreamer pipelines is to use the gstlaunch-1.0 utility, which is what was used in the previous experiment. The advantage of this method, is that it allows the user to quickly modify and experiment with the GStreamer pipelines.

Make a copy of the gstfilter2d_optflow.sh script, and edit the file with your favorite editor.

```
…
gst-launch-1.0 \
        v4l2src device=/dev/video2 io-mode="dmabuf" ! \
        "video/x-raw, width=${w}, height=${h}, format=${fmt}, framerate=${framerate}" ! \
        ${accel2} filter-mode=${accel2_mode} ! \
        queue ! \
        kmssink plane-id=29 bus-id=${bus_id} render-rectangle="<0,0,${qw},${qh}>" \
        \
        v4l2src device=/dev/video3 io-mode="dmabuf" ! \
        "video/x-raw, width=${w}, height=${h}, format=${fmt}, framerate=${framerate}" ! \
        kmssink plane-id=30 bus-id=${bus_id} render-rectangle="<${qw},0,${qw},${qh}>" \
        \
        v4l2src device=/dev/video4 io-mode="dmabuf" ! \
        "video/x-raw, width=${w}, height=${h}, format=${fmt}, framerate=${framerate}" ! \
        kmssink plane-id=31 bus-id=${bus_id} render-rectangle="<0,${qh},${qw},${qh}>" \
        \
        v4l2src device=/dev/video5 io-mode="dmabuf" ! \
        "video/x-raw, width=${w}, height=${h}, format=${fmt}, framerate=${framerate}" ! \
        ${accel1} filter-mode=${accel1_mode} filter-preset=${accel1_filter} ! \
        queue ! \
        kmssink plane-id=32 bus-id=${bus_id} render-rectangle="<${qw},${qh},${qw},${qh}>" \
        -v
```

If we take a closer look at the first pipeline, we can see that the video source is defined as a V4L2 video source, where the “/dev/video2” device identifies the first camera of the multi-camera FMC module.

```
v4l2src device=/dev/video2 io-mode="dmabuf" ! \
"video/x-raw, width=${w}, height=${h}, format=${fmt}, framerate=${framerate}" ! \
```

The pipeline includes a video processing function, implemented by one of our GStreamer plug-ins, and defined as follows:

```
${accel2} filter-mode=${accel2_mode} ! \
queue ! \
```

The output of the pipeline is defined as one of the planes of the HDMI output display controller, defined as:

```
kmssink plane-id=29 bus-id=${bus_id} render-rectangle="<0,0,${qw},${qh}>" \
```

Note that the render-rectangle property defines where on the monitor, the content will be displayed. In this case, the top-left quadrant is defined.

Modify the GStreamer pipelines, and re-run on hardware to experiment with the GStreamer infrastructure.

As an example, In the following example, the optical flow accelerator (accel2) was moved from camera 1 to camera 2 pipeline.

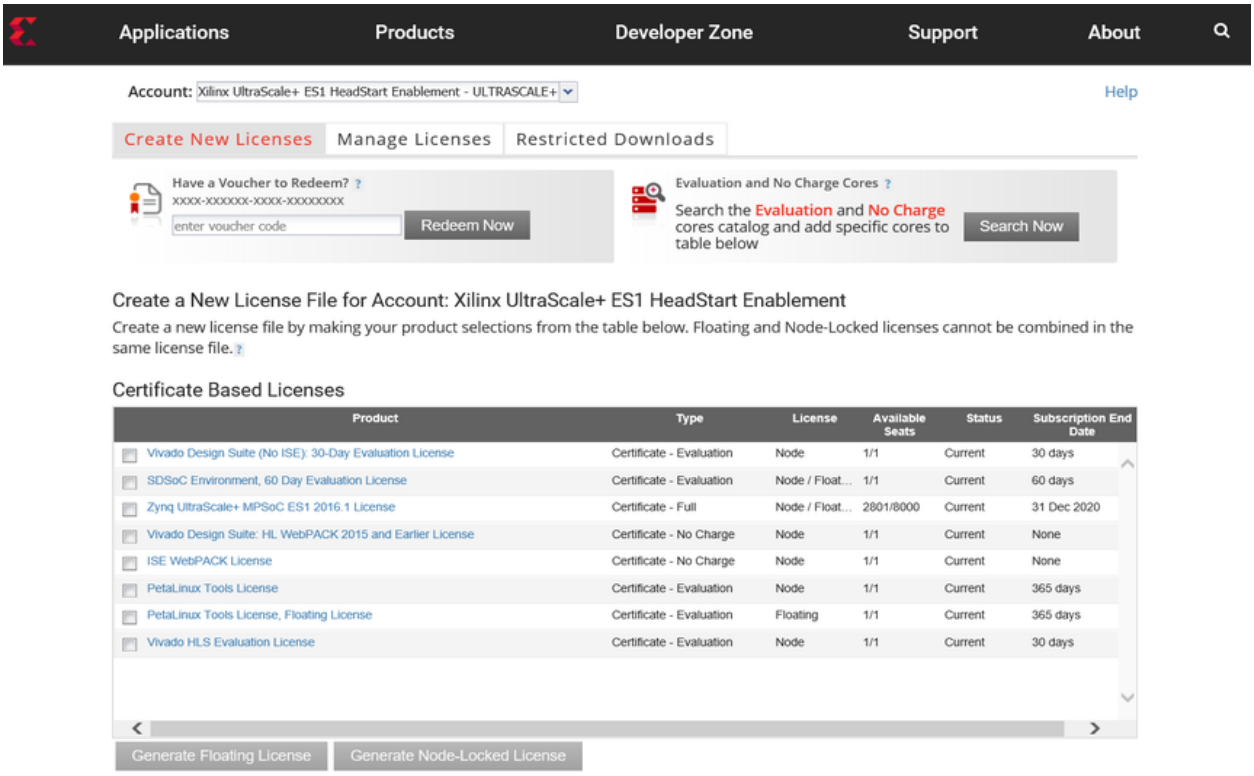
```
…
gst-launch-1.0 \
        v4l2src device=/dev/video2 io-mode="dmabuf" ! \
        "video/x-raw, width=${w}, height=${h}, format=${fmt}, framerate=${framerate}" ! \
        kmssink plane-id=29 bus-id=${bus_id} render-rectangle="<0,0,${qw},${qh}>" \
        \
        v4l2src device=/dev/video3 io-mode="dmabuf" ! \
        "video/x-raw, width=${w}, height=${h}, format=${fmt}, framerate=${framerate}" ! \
        ${accel2} filter-mode=${accel2_mode} ! \
        queue ! \
        kmssink plane-id=30 bus-id=${bus_id} render-rectangle="<${qw},0,${qw},${qh}>" \
        \
        v4l2src device=/dev/video4 io-mode="dmabuf" ! \
        "video/x-raw, width=${w}, height=${h}, format=${fmt}, framerate=${framerate}" ! \
        kmssink plane-id=31 bus-id=${bus_id} render-rectangle="<0,${qh},${qw},${qh}>" \
        \
        v4l2src device=/dev/video5 io-mode="dmabuf" ! \
        "video/x-raw, width=${w}, height=${h}, format=${fmt}, framerate=${framerate}" ! \
        ${accel1} filter-mode=${accel1_mode} filter-preset=${accel1_filter} ! \
        queue ! \
        kmssink plane-id=32 bus-id=${bus_id} render-rectangle="<${qw},${qh},${qw},${qh}>" \
        -v
```

Experiment 4: Licensing

The remaining of the experiments will require a valid Xilinx SDSoC license, which also includes the licenses for the video IP cores that are used in this reference design.

The following steps to generate a proper SDSoC license:

1. Log in here with your work E-mail address (If you do not yet have an account, follow the steps under Create Account)
2. Generate a license from “Create New Licenses” by checking "SDSoC Environment, 60 Day Evaluation License"



3. Under system information, give the host details.
4. Proceed until you get the license agreement and accept it.
5. The License (.lic file) will be sent to the email-id mentioned in the login details.
6. Copy the license file locally and give the same path in the SDSoC license manager.

Experiment 5: Re-Build the Sample Design

Before starting SDSoC, the SYSROOT environment variable must be configured to point to a directory inside the platform. The platform root directory is abbreviated and needs to be replaced with your exact local path : <platform>/petalinux/sdk/sysroots/aarch64-xilinx-linux

1. Under Linux:
    - export SYSROOT=<platform>/petalinux/sdk/sysroots/aarch64-xilinx-linux
2. Under Windows:
    - Start->Control Panel->System->Advanced System Settings->Advanced->Environment Variables
    - Create environment variable SYSROOT with value <platform>/petalinux/sdk/sysroots/aarch64-xilinx-linux

The platform ships with five file IO and two live IO design examples demonstrating popular OpenCV functions accelerated on the programmable logic. A third live I/O example shows how to combine the other two live I/O designs into one design, allowing the two accelerated functions to reside and run in parallel in the FPGA.

The live IO sample design examples are based on GStreamer. See GStreamer The open source GStreamer framework code is included with the reVISION platform, and design examples are built as GStreamer plugins. Pipelines may be run using the `gst-launch-1.0` utility.

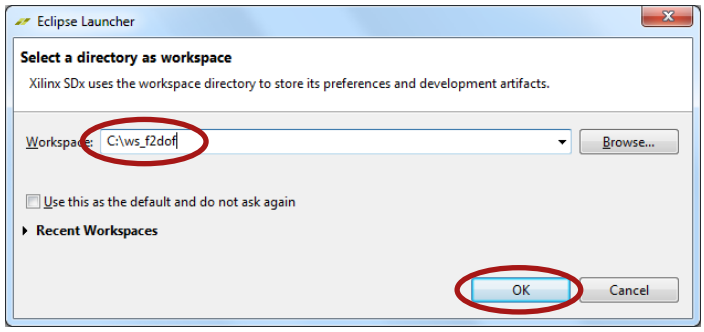
The three workspace names <ws_name> are `ws_f2d`, `ws_of`, and `ws_f2dof`.
The three sample names <name> are `filter2d`, `opticalflow`, and `filter2d_optflow`.

A GStreamer plugin is a shared library. In the case of the reVISION sample designs, the GStreamer plugin consists of two linked parts. These "top" and "bottom" parts are separate shared libraries produced by separate project builds. The top part is the GStreamer plugin itself, containing the code for interfacing with the GStreamer framework. See the `./workspaces/<ws_name>/gst/plugins/<name>` directory. The top part links with the bottom part which contains the code for the HW accelerated function(s). This bottom project generates the BOOT.BIN file containing the programmable logic used for the HW function(s). These are SDx projects: See the `./samples/live_IO/<name>` directory.

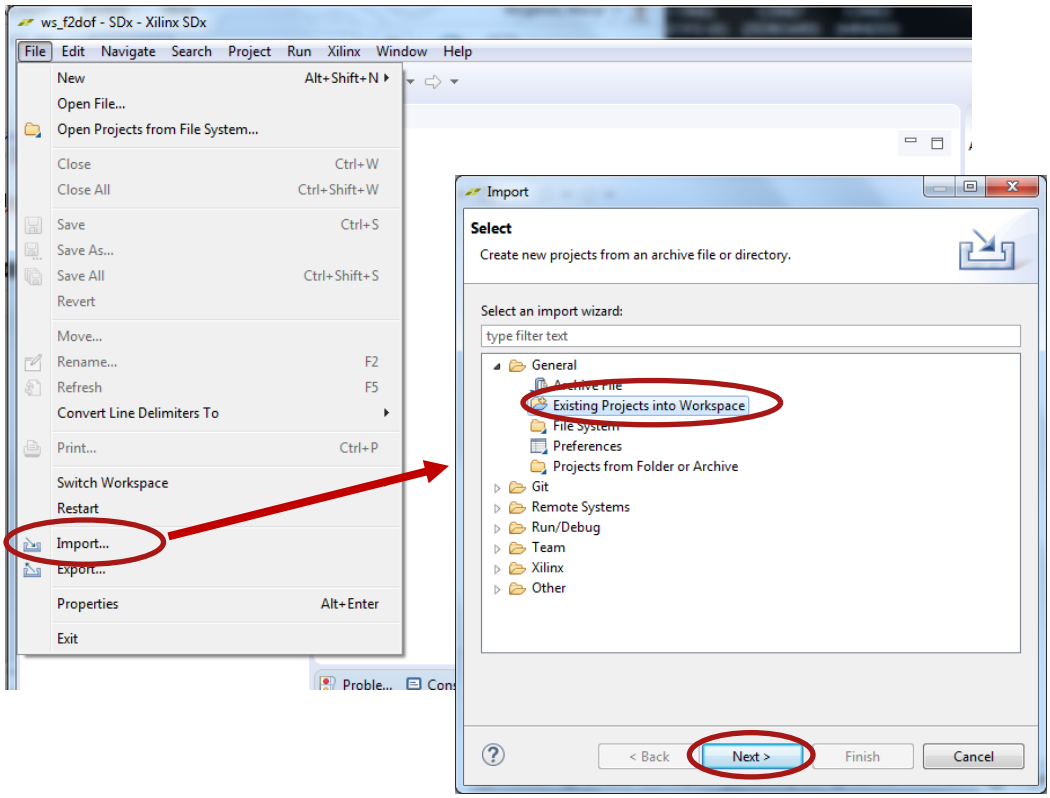
3. Make a copy of the `./workspaces/ws_f2dof/` workspace to another location on your hardware drive

NOTE : **If you are working on Windows there is a restriction**, i.e. file path lengths are restricted to 256 characters. The Xilinx build process creates some very deep directory structures with long names as it goes through the build process. You are advised, therefore, to keep the path to the workspace as short as possible. E.g. `C:\ws_f2dof\...`

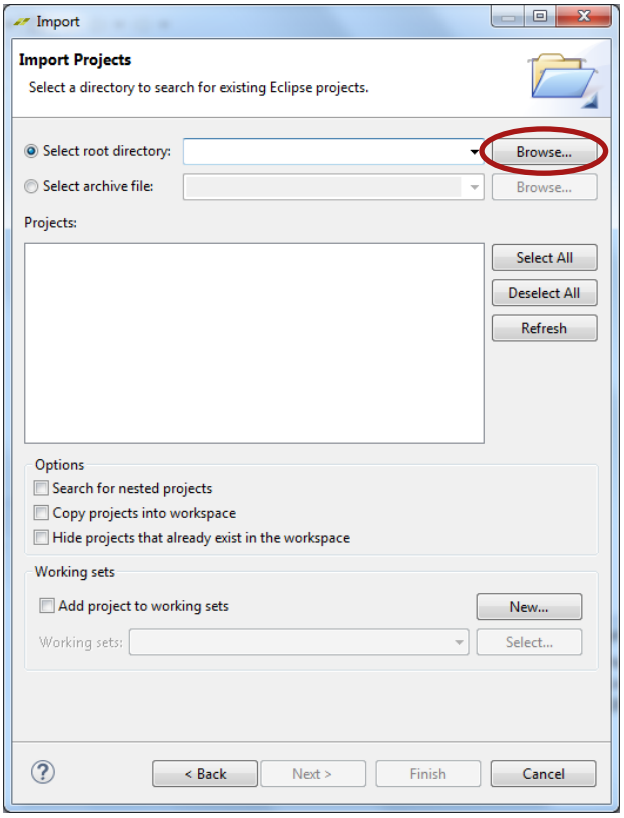
4. Start SDx and select your copy of the workspace `./ws_f2dof`.
   Make sure you use the same shell to run SDx as the one where you have set $SYSROOT
   Click 'OK'



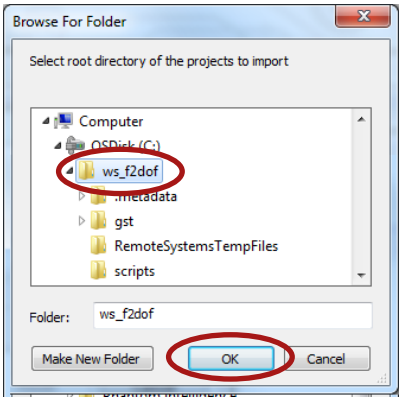
5. Close the Welcome screen and
   select 'File'→'Import'→'General'→'Existing Projects into Workspace'→'Next'.



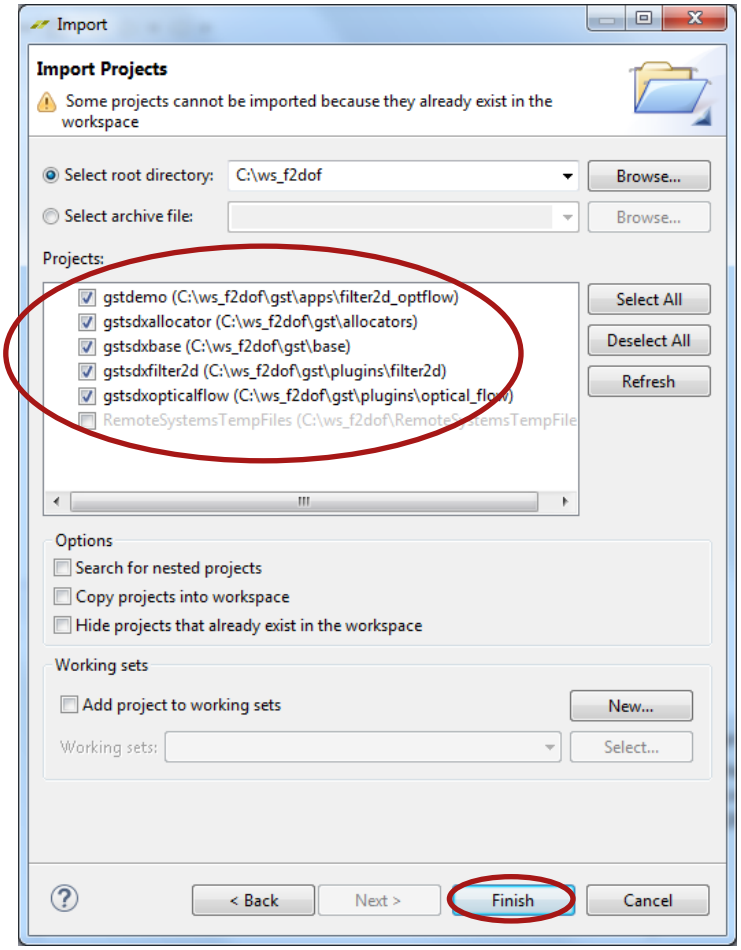
6. In the Import dialog, to the right of 'Select root directory', click 'Browse'.



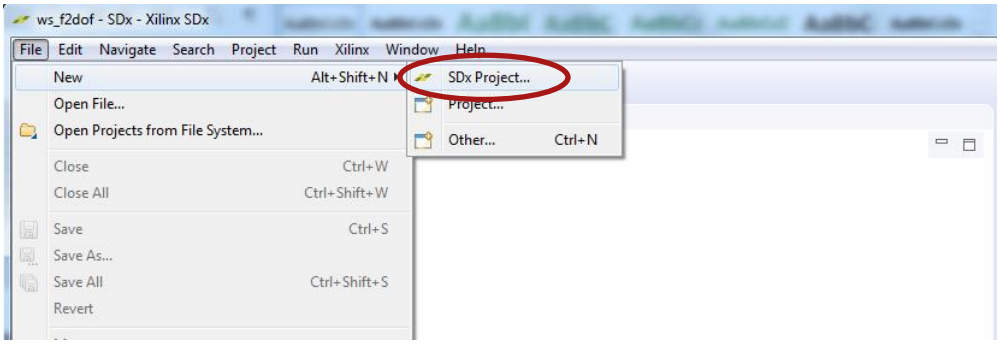
7. In the ‘Browse for Folder’ dialog, select your copy of the **ws_f2dof** workspace, then click 'OK'.



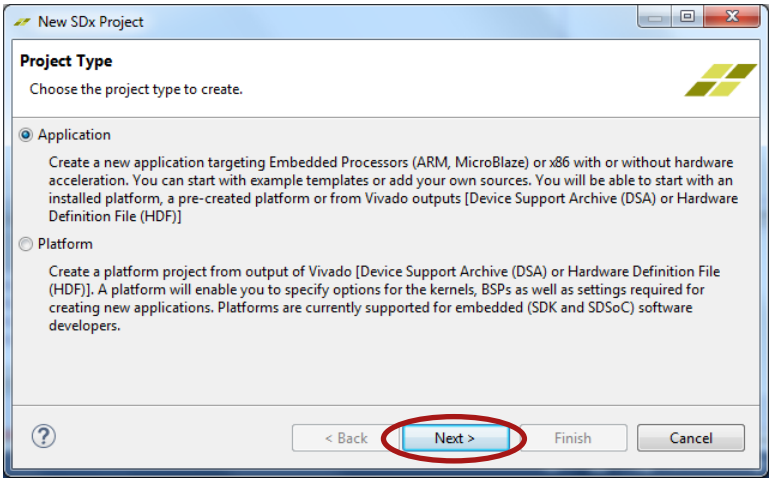
8. You should see a list of projects, with gstdemo, gstsdxallocator, gstsdxbase, gstsdxfilter2d and gstsdxopticalflow selected, click 'Finish'.



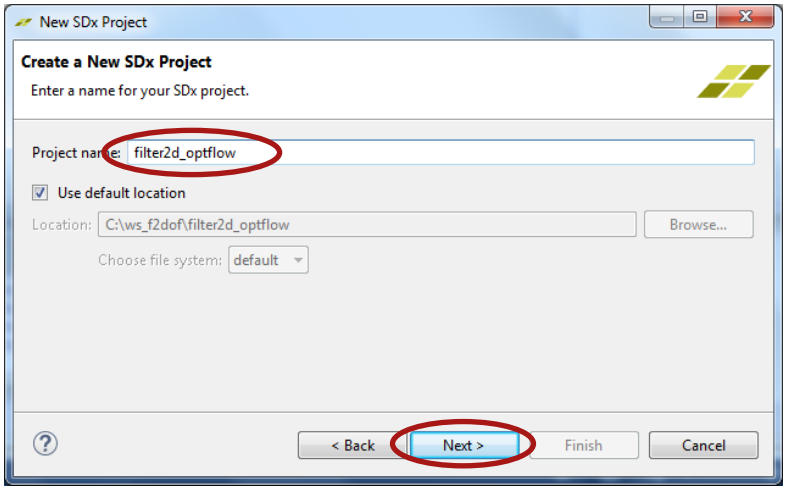
9. Back at the main window, the four imported projects appear in the Project Explorer pane. Now select 'File'→'New'→'SDx Project'... from the menu bar.



10. This brings up the Project Type dialog box, with Application Project selected, click 'Next'.



11. In the 'Create a New SDx Project' dialog, enter Project name 'filter2d_optflow', click 'Next'.



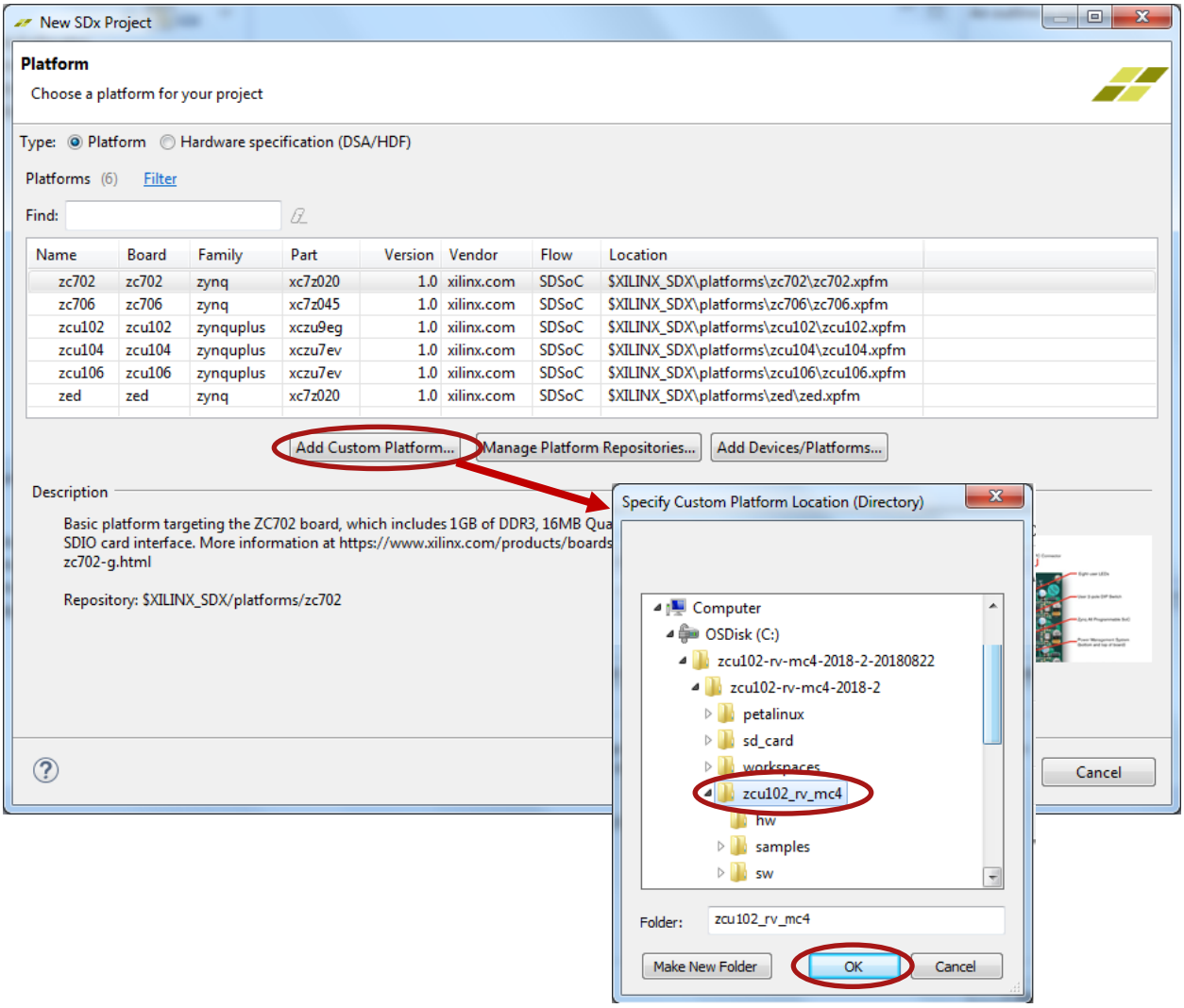
NOTE : The 'project name' is important, since the other gstreamer projects refer to this project by name for include directories and library directories. If you choose to specify a different name, you will need to change these directory references in the gstreamer projects.

The required 'project name' for the Filter 2D sample is `filter2d`.

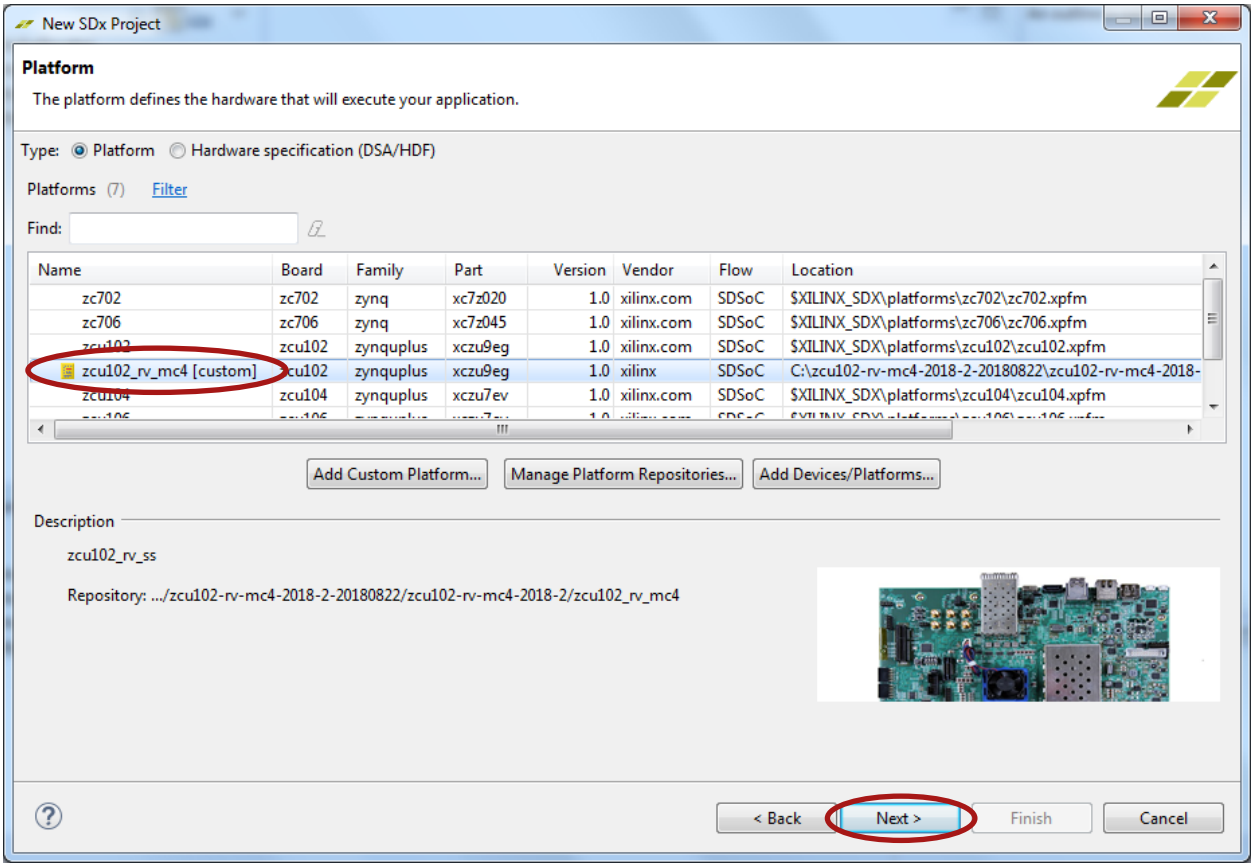
The required 'project name' for the Optical Flow sample is `opticalflow`.

The required 'project name' for the Filter 2D + Optical Flow sample is `filter2d_optflow`.

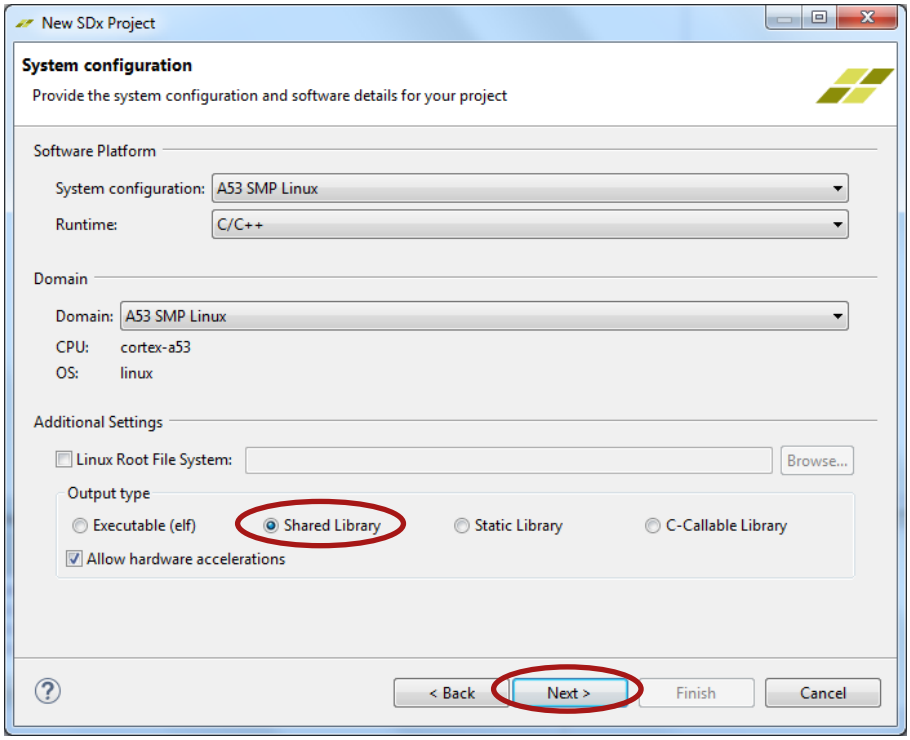
12. In the Platform dialog, click 'Add Custom Platform', find your way to the top directory where you unzipped the reVISION platform, called `zcu102_rv_mc4` or `zcu104_rv_mc4`.
    Click 'OK'.



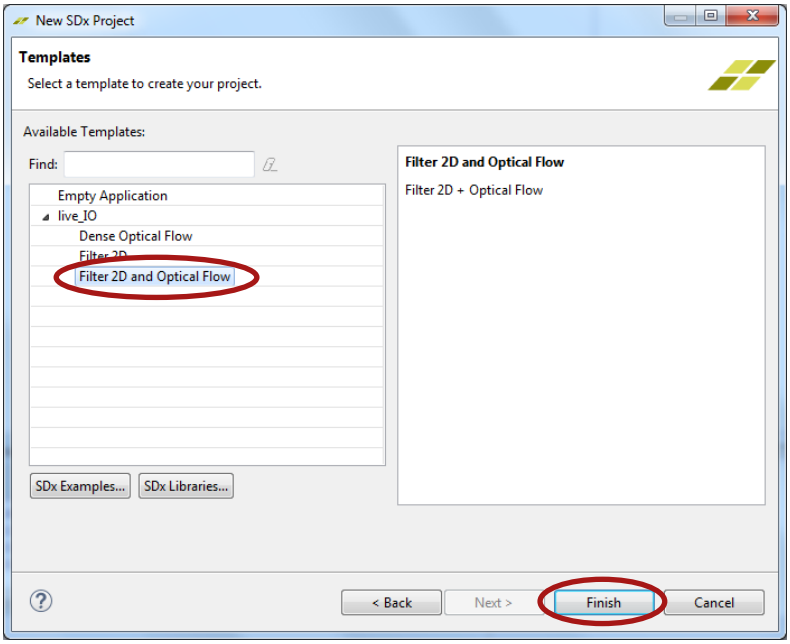
13. Back in the Platform dialog, the new platform appears in the list, but is not selected. Select it, then click 'Next'.



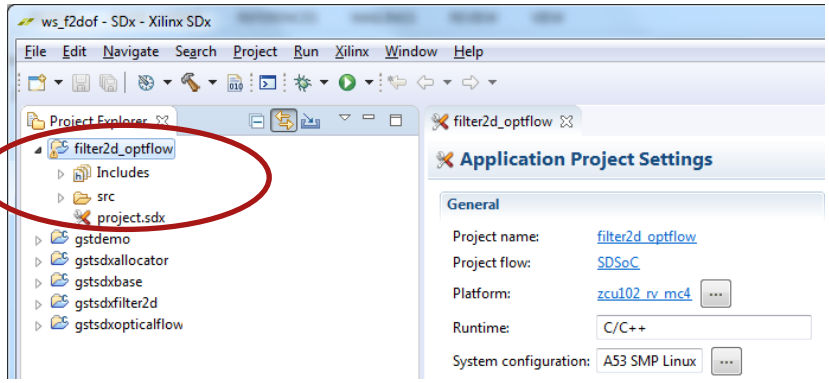
14. In the System configuration dialog, under Output type, select 'Shared Library', click 'Next'.



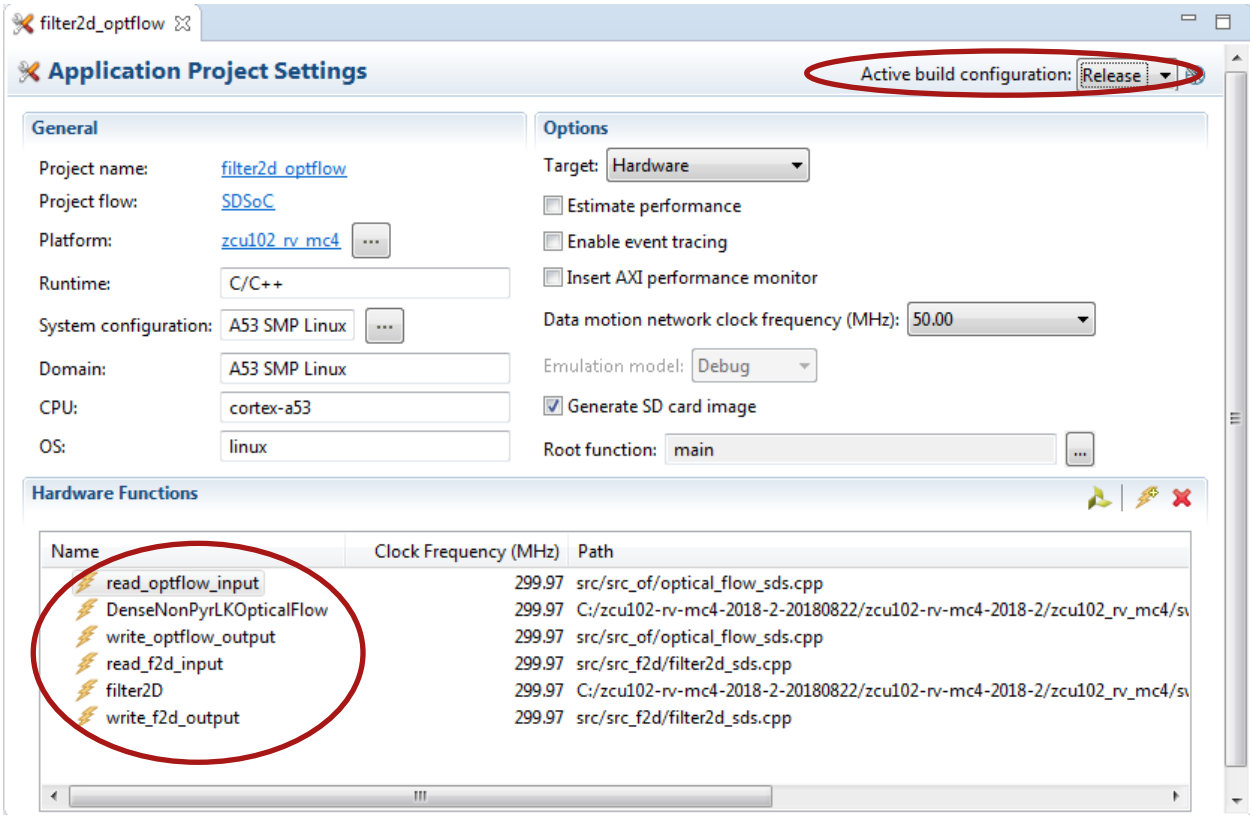
15. In the Templates dialog, under live_IO select ‘Filter 2D and Optical Flow’ and click 'Finish'.



16. Back again at the main window, the new project 'filter2d_optflow' appears with the five imported projects in the Project Explorer pane.

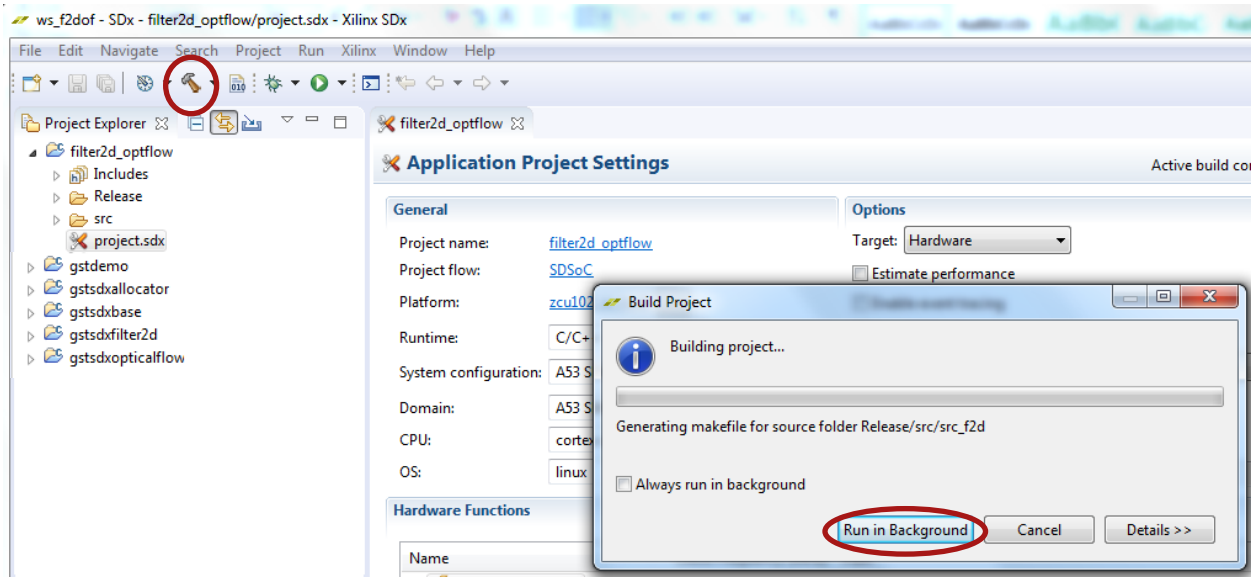


17. Switch the 'Active Build Configuration' for the opticalflow project to Release. Note that three routines are marked as Hardware Functions.



18. Build the filter2d_optflow project - do this by clicking right and choosing Build Project, or by clicking the 'hammer' icon.

In the small Build Project dialog that opens, you may hit the "Run in Background" button. That causes the small dialog box to disappear, though you can still see a progress icon in the lower right part of the GUI, showing that work is in progress. Select the Console tab in the lower central pane of the GUI to observe the steps of the build process as it progresses. The build process may take tens of minutes, up to several hours, depending on the power of your host machine, whether you are running on Linux or Windows, and of course the complexity of your design. By far the most time is spent processing the routines that have been tagged for realization in hardware - note the "HW functions" window in the lower part of the SDx Project Settings pane.
In our example above, the routines `read_optflow_input`, `DenseNonPyrLKOpticalFlow`, `write_optflow_output`, `read_f2d_input`, `filter2D`, and `write_f2d_output` are tagged to be built in hardware. The synthesis of the C code found in these routines into RTL, and the Placement and Routing of that RTL into the programmable logic in the Zynq MPSoC, are the steps that take most of the time.



Once the Build completes, you will find an sd_card directory has been created containing these files you'll need to transfer to your SD card.

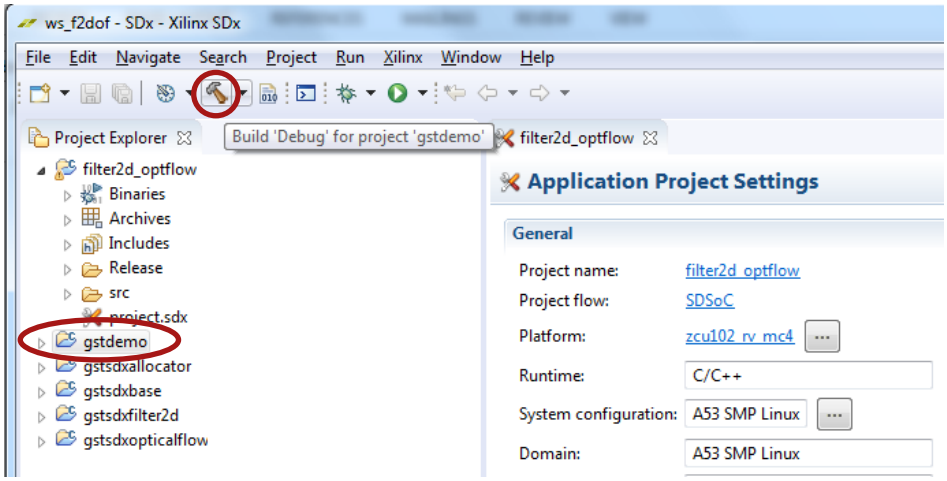
19. In order to prepare the SD card for newly built content, start by creating the following two directories on the sdcard

    - <sdcard>/**lib**
    - <sdcard>/**gstreamer-1.0**

20. Copy the following files to the SD card

Files to copy to SD card	**Destination on SD card**
./ws_f2dof/filter2d_optflow/Release/sd_card/**BOOT.BIN**	<sdcard>/
./ws_ f2dof/filter2d_optflow/Release/sd_card/**libfilter2d_optflow.so**	<sdcard>/lib/
./ws_ f2dof/filter2d_optflow/Release/sd_card/**image.ub**	<sdcard>/
./ws_ f2dof/filter2d_optflow/Release/sd_card/**video_cmd**	<sdcard>/

21. Now that the "bottom" shared library is built, you may build the "top" part, that will be linked with the bottom library. Now select the gstdemo project, and build it. Doing this will build all four of the gst--- projects.



This entire process should take only a few minutes.

22. Copy the following files to the SD card

Files to copy to SD card	Destination on SD card
./ws_f2dof/gst/allocators/Debug/**libgstsdxallocator.so**	<sdcard>/lib/
./ws_f2dof/gst/base/Debug/**libgstsdxbase.so**	<sdcard>/lib/
./ws_f2dof/gst/plugins/filter2d/Debug/**libgstsdxfilter2d.so**	<sdcard>/gstreamer-1.0/
./ws_f2dof/gst/plugins/optical_flow/Debug/**libgstsdxopticalflow.so**	<sdcard>/gstreamer-1.0/

Also, there are additional scripts that are needed to launch the gst-launch-1.0 utility.

23. Copy the following files to the SD card

Files to copy to SD card	Destination on SD card
./ws_f2dof/scripts/**gstpass.sh**	<sdcard>/
./ws_f2dof/scripts/**gstfilter2d_optflow.sh**	<sdcard>/

Finally, the gstdemo application, which is an example of how to directly call the GStreamer API can also be copied to the SD card.

24. Copy the following files to the SD card

Files to copy to SD card	Destination on SD card
./ws_f2dof/gst/apps/filter2d_optflow/**gstf2dofdemo.sh**	<sdcard>/
./ws_f2dof/gst/apps/filter2d_optflow/Debug/**gstdemo**	<sdcard>/

Now that you have re-built the SD card image for the filter2d_optflow sample, refer to **Experiment 2: Running the pre-built designs** for instructions on running the design on hardware.

Known Issues & Limitations

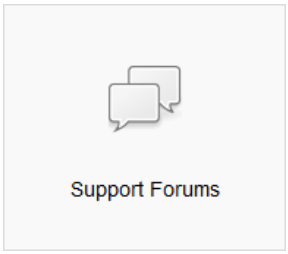
The following list enumerates the known issues and limitations with this reVISION platform

- only four camera configuration is supported
    - single camera configuration is not supported
    - dual camera configuration is not supported
- HDMI input is not supported with this reVISION platform

Appendix I: Getting Support

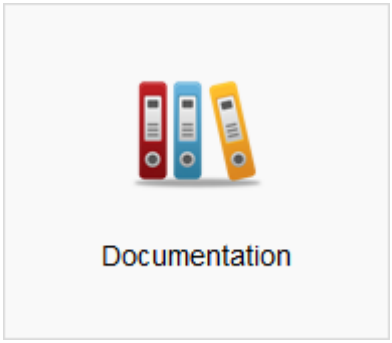
Avnet Support

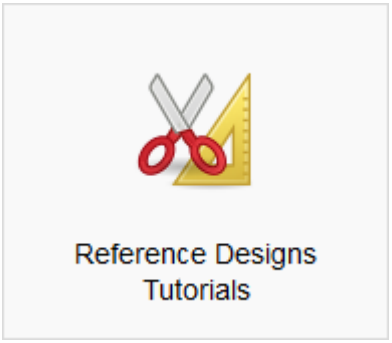
- Technical support is offered online through the ultrazed.org website support forums. Avnet dev kit users are encouraged to participate in the forums and offer help to others when possible.
  http://ultrazed.org/forums/zed-english-forum
  http://ultrazed.org/forums/software-application-development



- For questions regarding the community website, please direct questions to the ultrazed.org Web Master (webmaster@ultrazed.org).

- To access the most current collateral for the Avnet dev kits, visit the community support page (www.ultrazed.org/content/support) and click one of the icons shown below:





Xilinx Support

For questions regarding products within the Product Entitlement Account, send an email message to the Customer Service Representative in your region:

- Canada, USA and South America - isscs_cases@xilinx.com
- Europe, Middle East, and Africa - eucases@xilinx.com
- Asia Pacific including Japan - apaccase@xilinx.com

For technical support, including the installation and use of the product license file, contact Xilinx Online Technical Support at www.xilinx.com/support. The following assistance resources are also available on the website:

- Software, IP and documentation updates
- Access to technical support Web tools
- Searchable answer database with over 4,000 solutions
- User forums

Revision History

Date	Version	Revision
27 Aug 2018	2018.2.01	Initial Release
28 Aug 2018	2018.2.02	Change instructions from opticalflow to filter2d_optflow Add section on modifying the GStreamer pipelines
08 Nov 2018	2018.2.03	Change download links