

## Overview

The SDSoC platform for FMC-HDMI-CAM + PYTHON-1300-C camera module provides a feature rich framework for the development of video applications on the Xilinx Zynq-7000 SoC. It is based on the SDSoC platform for Xilinx ZC702 Base TRD, which provides the following infrastructure:

- HDMI capture pipeline
- HDMI display pipeline

Avnet augments this design with the following additional infrastructure:

- PYTHON capture pipeline

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

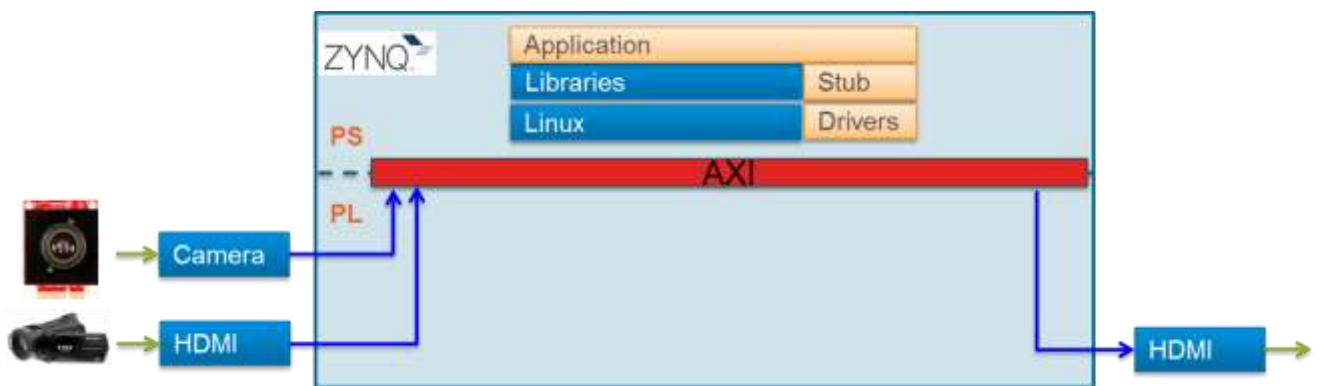


Figure 1 – FMC-HDMI-CAM + PYTHON-1300 – Hardware Block Diagram

## Objectives

This tutorial will guide the user how to:

- Execute the reference design on hardware
- Rebuild the reference design

## PYTHON-1300-C Camera Module

This reference design supports the PYTHON-1300-C camera module. The camera module features ON Semiconductor's PYTHON-1300 color image sensor. The PYTHON-1300 is a 1/2 inch Super-eXtended Graphics Array (SXGA) CMOS image sensor with a pixel array of 1280 by 1024 pixels. Designed to address the needs of general purpose industrial image sensing applications, the new global shutter image sensor combines flexibility in configuration and resolution with high speed and high sensitivity for the industrial imaging market.

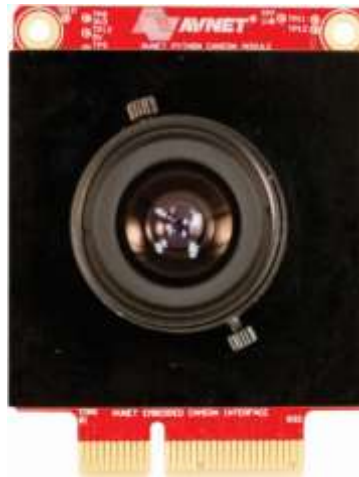


Figure 2 – PYTHON-1300-C Camera Module

The PYTHON-1300-C camera module is supported by a camera receiver IP core which consists of HDL source, and is provided free of charge, and royalty free. It is meant to be used with other cores to form a complete capture pipeline.

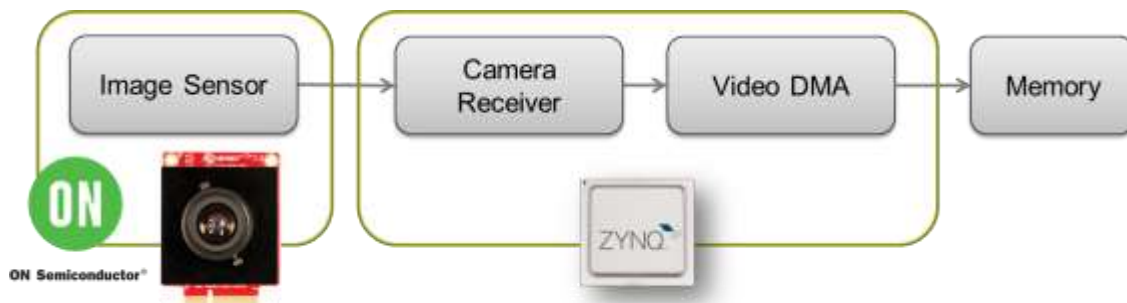


Figure 3 – PYTHON-1300 – Camera Receiver IP Core

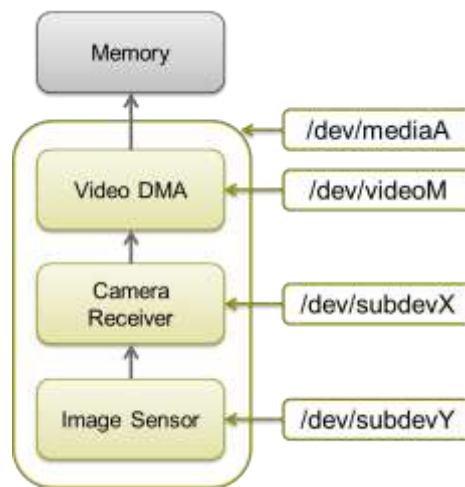
The PYTHON-1300-C camera module is also supported by linux drivers. The V4L2 framework was chosen, where complete pipelines (capture pipeline, memory pipeline, display pipeline) can be implemented.

The capture pipeline will capture video from the camera receiver, optionally process the video, then send the content to an external frame buffer using a video DMA engine.

The V4L2 framework specifies the capture pipeline as a video node, consisting of several sub-device drivers.

V4L2 sub-device drivers are provided for:

- Image Sensor : configures the image sensor via SPI
- Camera Receiver : de-serializes the video content from the image sensor

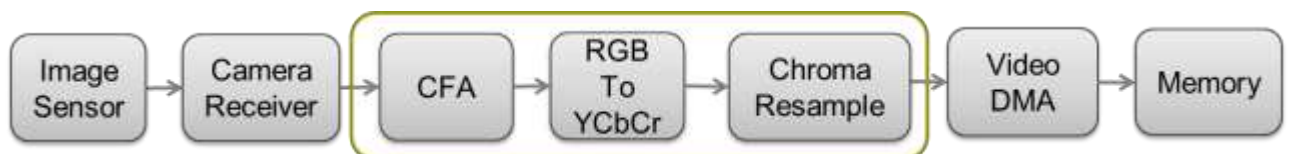


**Figure 4 – PYTHON-1300 – Sub-device linux drivers**

The V4L2 capture pipeline is fully user configurable via the Device Tree.

Using this mechanism, the user can add additional processing cores, such as:

- Color Filter Array (CFA) interpolation, to interpolate the missing colors of the incoming bayer pattern
- Color Space Conversion (CSC), to switch from RGB to YCbCr color spaces
- Chroma Resample (CRESAMPLE), to down-sample the chroma (Cb/Cr) components and reduce bandwidth



**Figure 5 – PYTHON-1300 – Complete capture pipeline**

## References to the SDSoC platform for ZC702 Base TRD

The SDSoC platform for FMC-HDMI-CAM + PYTHON-1300-C camera module is based on the SDSoC platform for Xilinx ZC702 Base TRD.

The following documentation should be consulted for more detailed information:

### **RDF 0369 – ZC702 ZVIK Base TRD - Design Files**

[rdf0369-zc702-zvik-base-trd-sdsoc-2015-4.zip](#)

### **UG 925 - ZC702 Base Targeted Reference Design User Guide**

[http://www.xilinx.com/support/documentation/boards\\_and\\_kits/zc702\\_zvik/2015\\_4/ug925-zynq-zc702-base-trd.pdf](http://www.xilinx.com/support/documentation/boards_and_kits/zc702_zvik/2015_4/ug925-zynq-zc702-base-trd.pdf)

### **Xilinx Wiki - Zc702 Base TRD**

<http://www.wiki.xilinx.com/Zc702+Base+TRD>

## Required Licenses

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

- Color Filter Array Interpolation (CFA) v7.0
- RGB to YcrCb Color-Space Converter v7.1
- Chroma Resampler v4.0
- Video Timing Controller (VTC) v6.1
- Test Pattern Generator (TPG) v7.0

The design also makes use of the XYLON display controller (logiCVC) core. The logiCVC core does not require an explicit evaluation license, and defaults to evaluation mode.

- logiCVC v5.0.1 - <http://www.logicbricks.com/Products/logiCVC-ML.aspx>

---

## Experiment Setup

This tutorial makes use of Xilinx Vivado Design Suite in scripting mode in order to create a project. The resulting project can be opened with the graphical (GUI) version of the tools for further analysis and modification.

## Software

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

- Windows-7 64-bit
- Terminal Emulator (HyperTerminal or TeraTerm)
- Xilinx Vivado Design Suite 2015.4
- PicoZed FMC Carrier V2 - Board Definition Install for Vivado 2015.4
  - <http://picozed.org/support/documentation/13076>

## Hardware

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

- Win-7 PC with a recommended 2 GB RAM available for the Xilinx tools to complete a XC7Z020 design<sup>1</sup>
- One of the following supported FMC carriers:
  - PicoZed 7030 SOM + FMC Carrier Card V2
  - ZedBoard
  - ZC702
- FMC-HDMI-CAM FMC module
- ON Semiconductor PYTHON-1300-C Camera Module (optional)
- DVI or HDMI video source
- HDMI (or DVI-D) monitor (1080P60 capable)
- USB cable (Type A to Micro-USB Type B)
- 4GB MicroSD card

---

<sup>1</sup> Refer to <http://www.xilinx.com/design-tools/vivado/memory.htm>

## Experiment 1: Extract the design files

In this section, the design files for the reference design will be extracted to your computer.

1. Extract the fmchc\_sdsoc\_{target}\_2015\_4\_02.zip archive to the C:\ root directory, where {target} is one of pzfm2, zed, or zc702.
2. You should see the following directory structure, with one of the {target} subdirectories:

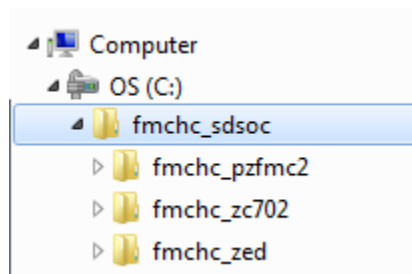


Figure 6 – Extracted C:\fmchc\_hls directory structure

**NOTE** : the exact directory name is not critical, but it must remain short on Windows machines, due to the directory length limitation of Windows

The C:\fmchs\_sdsoc directory will contain one of the following sub-directories:

Directory	Content Description
C:\fmchc_sdsoc\fmchc_pzfmc2	Contains the SDSoc platform for the PicoZed 7030 SOM + PicoZed FMC Carrier V2 hardware.
C:\fmchc_sdsoc\fmchc_zc702	Contains the SDSoc platform for the ZC702.
C:\fmchc_sdsoc\fmchc_zed	Contains the SDSoc platform for the ZedBoard.

Each fmchc\_{target} sub-directory (being one of pzfm2, zc702 or zed) is a SDSoc platform and contains the following directory structure:

Directory	Content Description
\fmchc_{target}\fmchc_{target}_hw.pfm	Platform hardware description file
\fmchc_{target}\fmchc_{target}_sw.pfm	Platform software description file
\fmchc_{target}\arm-xilinx-linux-gnueabi	Linux header files and libraries
\fmchc_{target}\boot	Boot files
\fmchc_{target}\boot\petalinux	Source files for re-building petalinux images
\fmchc_{target}\hardware	Pre-built hardware
\fmchc_{target}\samples	Platform sample applications
\fmchc_{target}\vivado	Vivado project

\fmchc_{target}\sd_card	Pre-built SD card image
-------------------------	-------------------------

## Experiment 2: Licensing the Video and Image Processing Pack IP Cores

This reference design uses several of the Xilinx Video and Image Processing Pack IP cores. In order to build the hardware design, valid licenses (hardware evaluation, or full license) are required for the following video IP cores:

- Color Filter Array Interpolation (CFA) v7.0
- RGB to YcrCb Color-Space Converter v7.1
- Chroma Resampler v4.0
- Video Timing Controller (VTC) v6.1
- Test Pattern Generator (TPG) v7.0

Follow these steps to request an evaluation license:

1. Navigate to the “Video and Image Processing Pack” product page on the Xilinx web site :  
<http://www.xilinx.com/products/intellectual-property/ef-di-vid-img-ip-pack.html>
2. Click the Evaluate link located on the right of the web page, and follow the online instructions

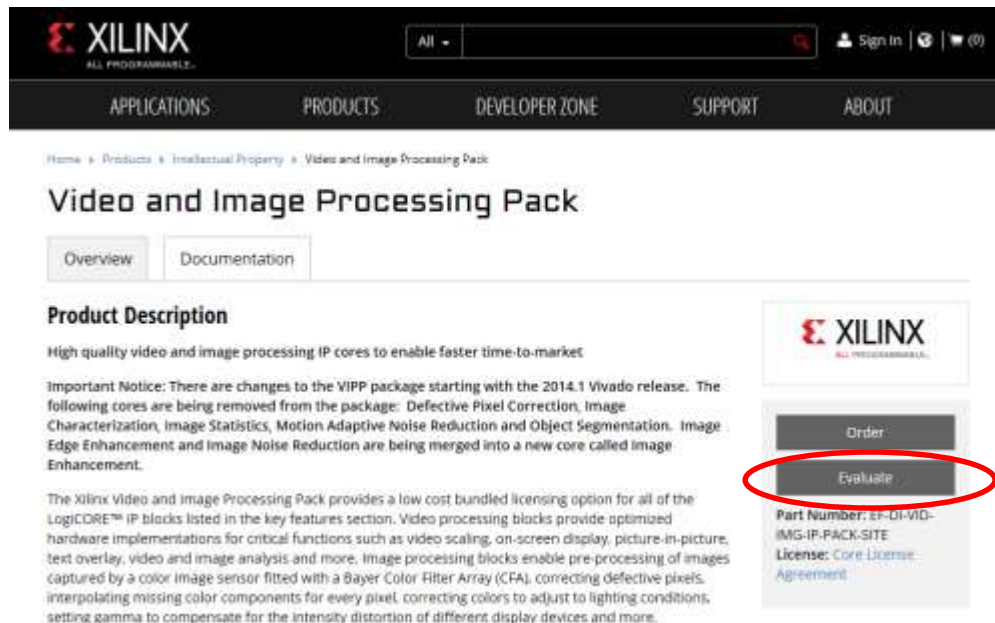


Figure 7 – Video and Image Processing Pack – product page

3. The generated license file is sent by email. Follow the enclosed instructions to add the evaluation licenses.

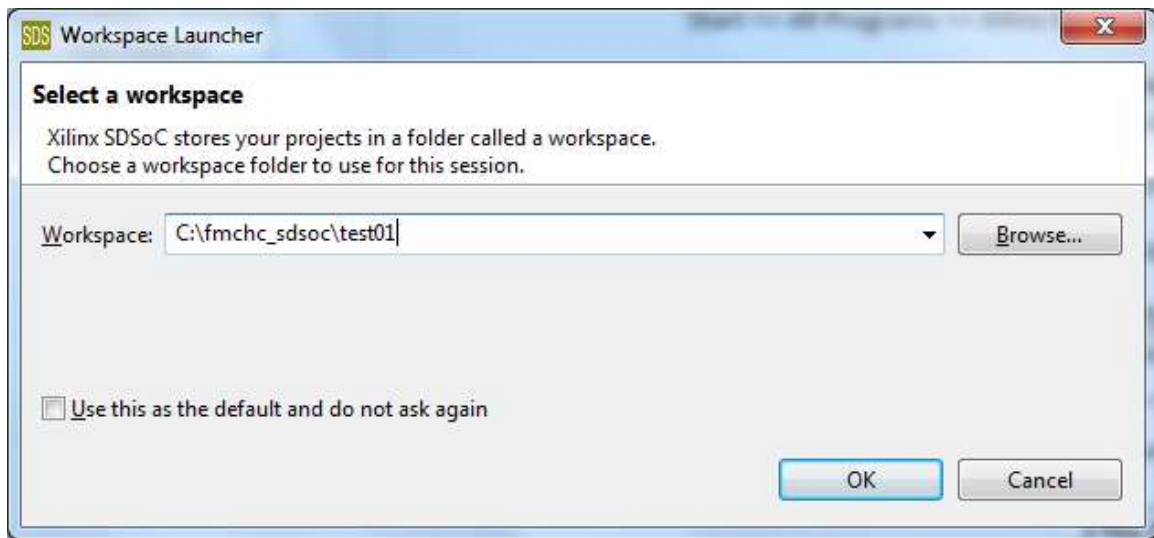


## Experiment 3: Build the Platform Sample Application

In this section, the SDSoC sample application will be built with SDSoC.

### Building the Platform Sample Application for sw implementation

1. From the Start menu, start SDSoC 2015.4 :  
Start => All Programs => Xilinx Design Tools => SDSoC 2015.4 => SDSoC 2015.4
2. Specify a workspace, such as “C:\fmchc\_sdsoc\test01”



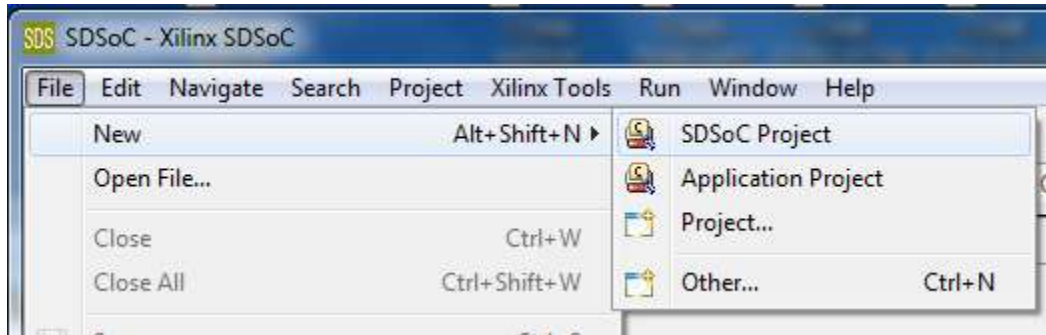
**Figure 8 – SDSoC 2015.4 – Select a Workspace**

**NOTE :** the exact directory name is not critical, but it must remain short on Windows machines, due to the directory length limitation of Windows

3. Close the Welcome window.

20 May 2016

4. Create a new SDSoC Project, with the following menu selection:  
File => New => SDSoC Project



**Figure 9 – SDSoC 2015.4 – Create New SDSoC Project**

5. In the New Project dialog
  - a. Specify a SDSoC Project name, such as “**video\_cmd**”
  - b. Specify one of the following SDSoC platforms by clicking on the “Others” button next to the Target Platform selection:
    - i. **C:\fmchc\_sdsoc\fmchc\_pzfm2** : PicoZed 7030 SOM + PicoZed FMC Carrier V2 + FMC-HDMI-CAM + PYTHON-1300-C camera module
    - ii. **C:\fmchc\_sdsoc\fmchc\_zc702** : ZC702 + FMC-HDMI-CAM + PYTHON-1300-C camera module
    - iii. **C:\fmchc\_sdsoc\fmchc\_zed** : ZedBoard + FMC-HDMI-CAM + PYTHON-1300-C camera module
  - c. Click **Next**

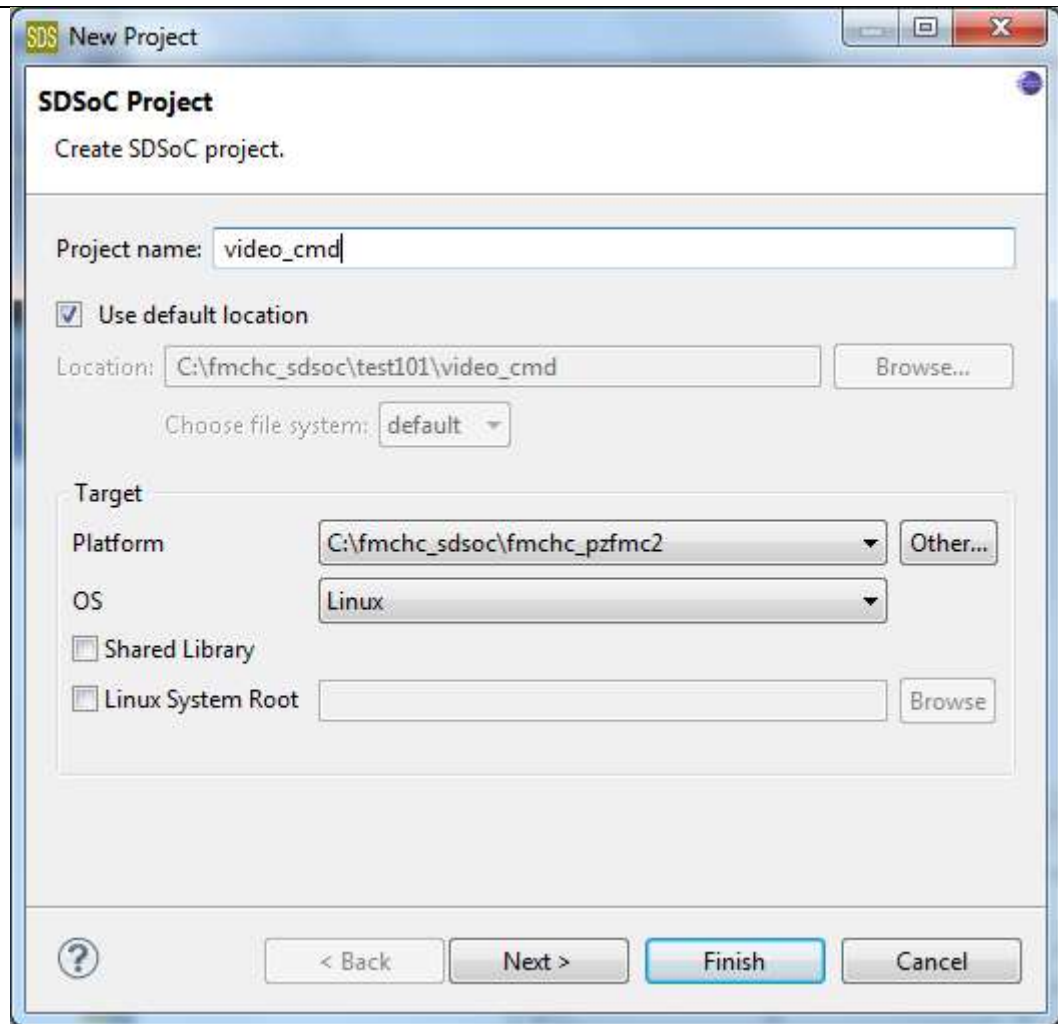


Figure 10 – SDSoC 2015.4 – New Project – SDSoC Project

6. In the New Project – Templates dialog
  - a. Select the Video Command Line sample
  - b. Click **Finish**

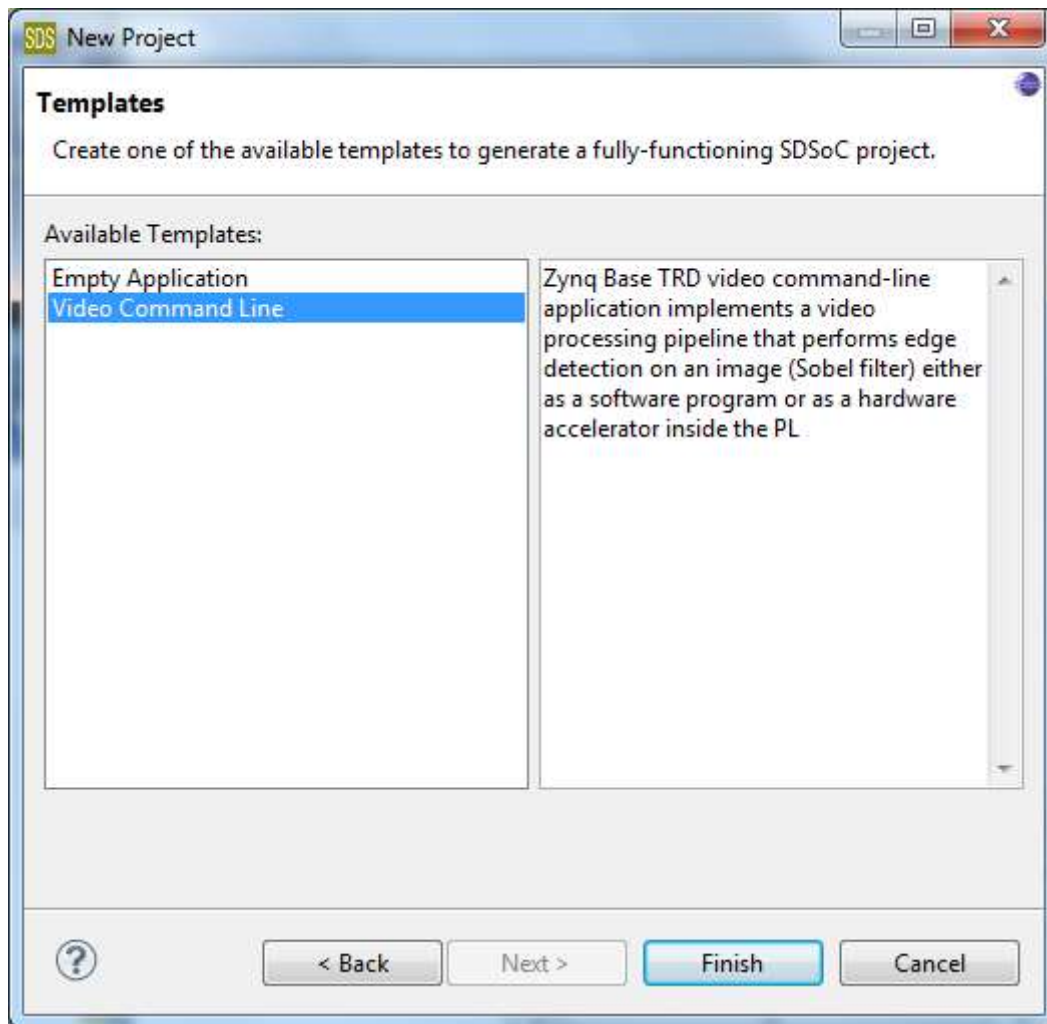
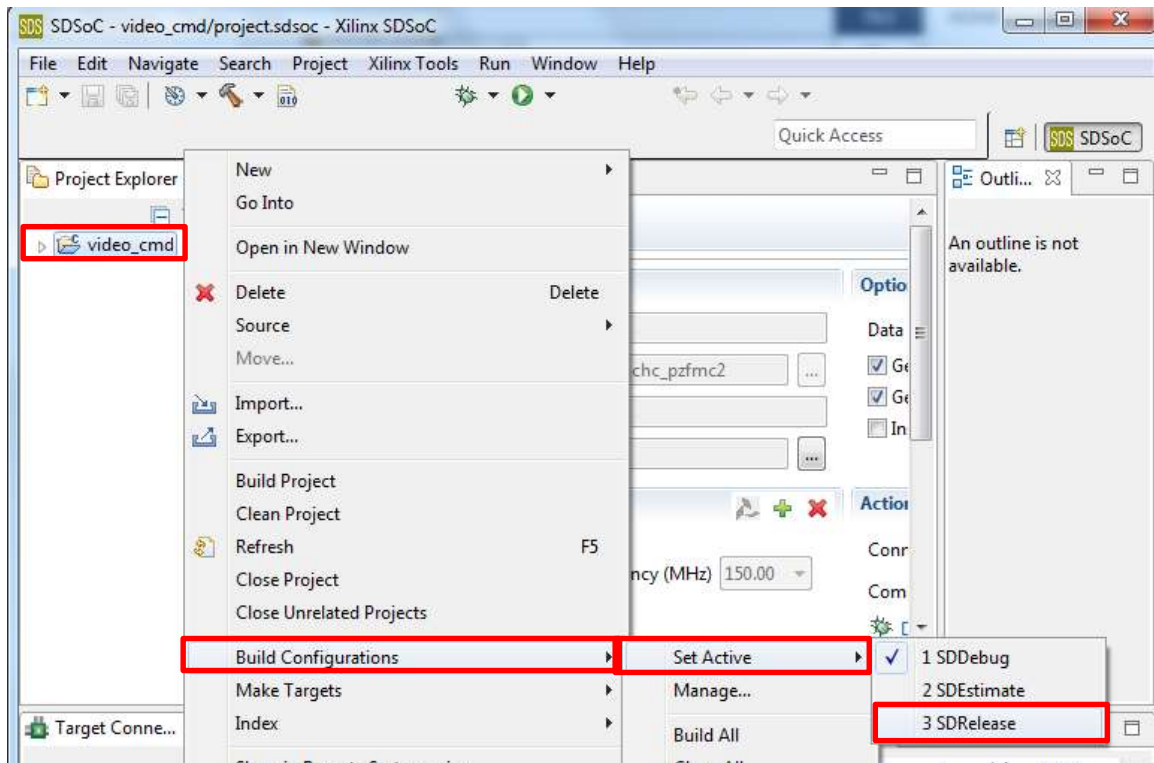


Figure 11 – SDSoC 2015.4 – New Project – Templates

20 May 2016

7. In order to build an optimized implementation, set the Build Configuration to SDRelease by right-clicking on the “video\_cmd” project, then making the following selection:

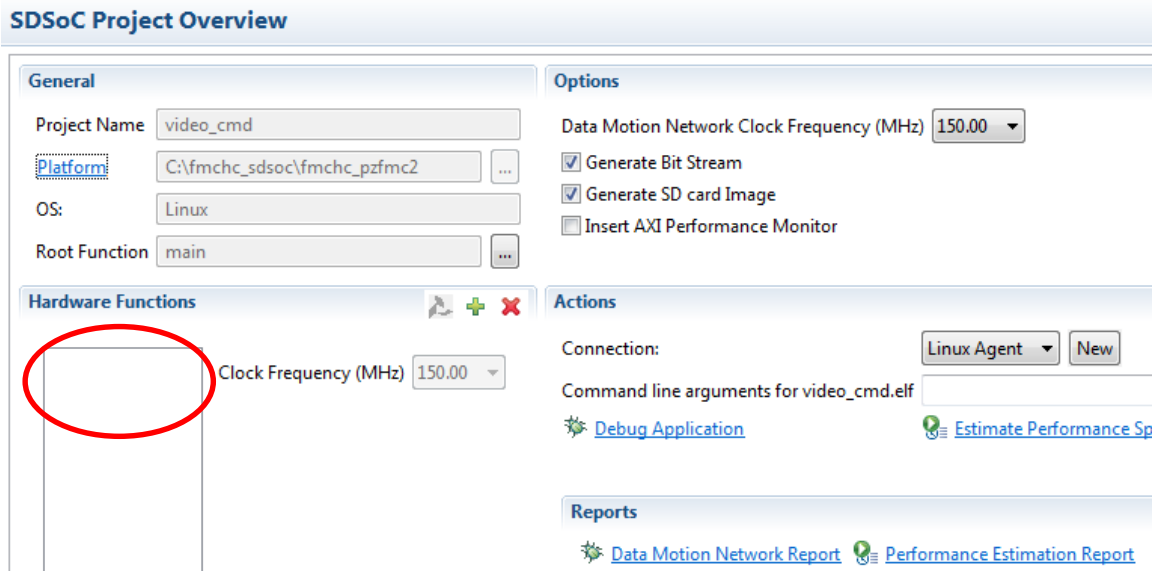
*Build Configurations => Set Active => SDRelease*



**Figure 12 – SDSoC 2015.4 – SDRelease Build Configuration**

20 May 2016

In the SDSoC Project Overview, notice that the “Hardware Functions” list is empty. This is normal, since we have not selected any functions to accelerate to hardware.



**Figure 13 – SDSoC 2015.4 – SDSoC Project Overview**

8. Build the SDSoC project by right-clicking on the “video\_cmd” project, then making the following selection :  
*Build Project*

Since no functions have been selected for acceleration to hardware, the build will complete relatively quickly, making use of the pre-built hardware files in the SDSoC platform.

Proceed to “**Experiment 4: Executing SD card image on hardware**” for instructions on how to execute the SD card image on hardware.

## Building the Platform Sample Application for hw acceleration

In this section, we will once again build the SDSoC project, but this time identifying a function for hardware implementation.

9. Navigate within the “video\_cmd” project in the Project Explorer and select the following function:

`video_cmd => src => src => sds_sobel.cpp => sds_sobel( ... )`

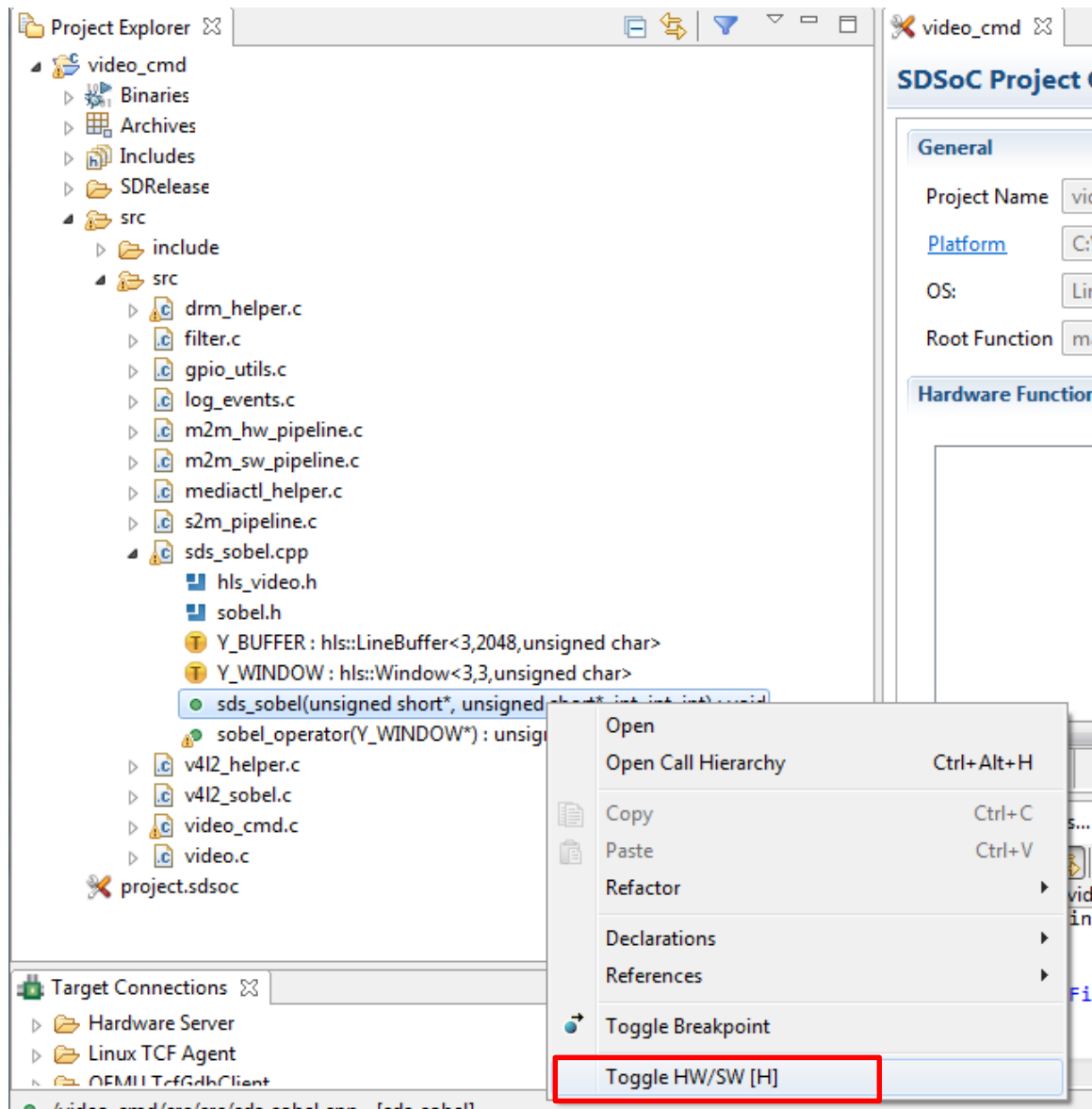
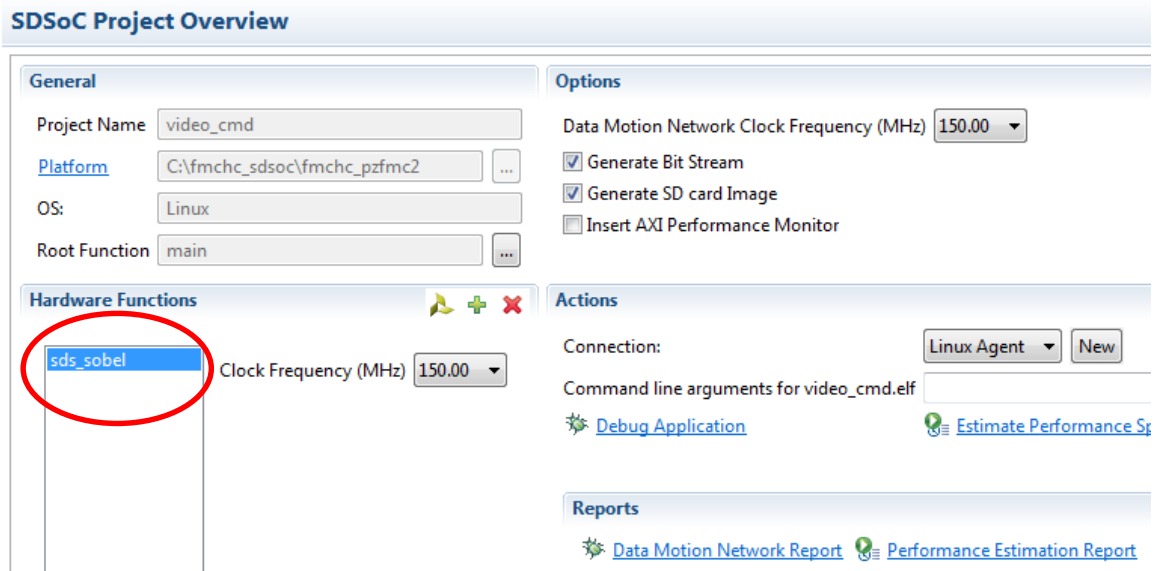


Figure 14 – SDSoC 2015.4 – sds\_sobel : Toggle HW/SW

10. Right-click on the `sds_sobel( ... )` function, then select Toggle HW/SW to select the function for hardware implementation.

In the SDSoC Project Overview, notice that the “Hardware Functions” list now contains one function, the “`sds_sobel( ... )`” function.



**Figure 15 – SDSoC 2015.4 – SDSoC Project Overview**

11. Clean the previous build for the SDSoC project by right-clicking on the “video\_cmd” project, then making the following selection :  
*Clean Project*
12. Build the SDSoC project by right-clicking on the “video\_cmd” project, then making the following selection :  
*Build Project*

Since we have selected a function acceleration to hardware, the build will take longer since a the hardware platform will be modified to add the `sds_sobel` function, as well as all required data mover infrastructure, then re-built.

Proceed to “**Experiment 4: Executing SD card image on hardware**” for instructions on how to execute the SD card image on hardware.

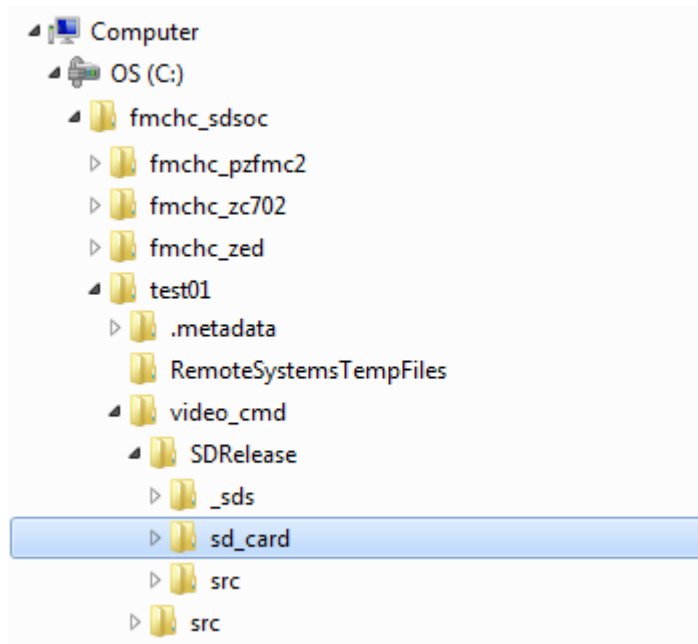


## Experiment 4: Executing SD card image on hardware

This section will describe how to execute the SD card image on one of the following FMC Carriers:

- ZedBoard
- ZC702
- PicoZed 7030 + PicoZed FMC Carrier V2

In the previous experiment, an SDSoC project was built, which resulted in the creation of a SD card image, as shown below:



**Figure 16 – SDSoC 2015.4 – SDRlease build artifacts**

1. Copy the contents of the “sd\_card” directory to the targeted hardware’s SD card.
2. Boot the target hardware from the SD card

## Linux Prompt (Serial Console)

3. Once the design is running on hardware, you should see something similar to the following on your serial console:

```
...  
Built with PetaLinux v2015.4 (Yocto 1.8) zynq /dev/ttyPS0  
zynq login:
```

4. Specify "root" as the user and password to login to the linux console:

```
zynq login: root  
Password: root  
login[1094]: root login on 'ttyPS0'  
  
root@zynq:~#
```

## Launching the VIDEO\_CMD application

The video\_cmd application defaults to 1080P video output resolution.

5. To launch the VIDEO\_CMD application,  
execute the following command in the linux console:

```
root@zynq:~# /media/card/video_cmd.elf  
  
Video Control application:  
-----  
DRM module name: xylon-drm  
HDMI output resolution: 1920x1080  
  
----- Select Video Source -----  
1 : Test Pattern Generator  (*)  
2 : HDMI Input  
3 : PYTHON-1300 Camera  
  
----- Select Filter Type -----  
4 : Sobel  (*)  
  
----- Toggle Filter Mode -----  
5 : Filter OFF/ON  (OFF)
```

```

----- Exit Application -----
0 : Exit

Enter your choice :

```

## Selecting the Video Source

This version of the VIDEO\_CMD application supports three video sources:

Video Source	Description
Test Pattern Generator	Internal test pattern generator
HDMI input	External video source from HDMI input interface
PYTHON-1300-C Camera	External video source from PYTHON-1300-C camera module

6. To select the PYTHON-1300-C camera video source, type '3' then 'ENTER'

```

Enter your choice : 3

----- Select Video Source -----
1 : Test Pattern Generator
2 : HDMI Input
3 : PYTHON-1300 Camera  (*)

----- Select Filter Type -----
4 : Sobel  (*)

----- Toggle Filter Mode -----
5 : Filter OFF/ON  (OFF)

----- Exit Application -----
0 : Exit

PYTHON1300: PYTHON-1300 Sensor detected

Enter your choice :

```

## Selecting the Filter Mode

The VIDEO\_CMD application supports two filter modes:

Filter Mode	Description
OFF	Image processing disabled
ON	Image processing enabled

7. To select the different filter modes (OFF, ON), type '5' then 'ENTER'

```
Enter your choice : 5

----- Select Video Source -----
1 : Test Pattern Generator
2 : HDMI Input
3 : PYTHON-1300 Camera  (*)
PYTHON1300: PYTHON-1300 Sensor detected

----- Select Filter Type -----
4 : Sobel  (*)

----- Toggle Filter Mode -----
5 : Filter OFF/ON  (ON)

----- Exit Application -----
0 : Exit

Enter your choice : 5

----- Select Video Source -----
1 : Test Pattern Generator
2 : HDMI Input
3 : PYTHON-1300 Camera  (*)

----- Select Filter Type -----
4 : Sobel (OpenCV)  (*)

----- Toggle Filter Mode -----
5 : Filter OFF/ON  (OFF)

----- Exit Application -----
0 : Exit

PYTHON1300: PYTHON-1300 Sensor detected

Enter your choice :
```

## Exiting the application

8. To exit the application, '0' then 'ENTER'

```
Enter your choice : 0  
root@zynq:~#
```

## Appendix 1 : Verifying the Video Pipelines

The Petalinux image includes the following utilities, which can be used to query and/or test the various video pipelines in the design:

- kmstest
- media-ctl
- modetest
- modeprint
- proptest
- v4l2-compliance
- v4l2-ctl
- v4l2-sysfs-path
- vbltest

In order to validate the display pipeline, the modeprint utility can be used:

```
root@zynq:~# modeprint xylon-drm
Starting test
Resources

count_connectors : 1
count_encoders   : 1
count_crtcs      : 1
count_fbs        : 0

Connector: HDMI-A-1
    id          : 33
    encoder id   : 32
    conn         : disconnected
    size         : 0x0 (mm)
    count_modes  : 0
    count_props  : 2
    props        : 1 2
    count_encoders : 1
    encoders     : 32

Encoder
    id          : 32
    crtc_id     : 0
    type        : 2
    possible_crtcs : 0x1
    possible_clones : 0x0

Crtc
    id          : 27
    x           : 0
    y           : 0
    width       : 0
    height      : 0
```

```

mode           : 0x13234
gamma size     : 0

```

```

Ok
root@zynq:~#

```

In order to validate the HDMI capture pipeline, the media-ctl utility can be used:

```

root@zynq:~# media-ctl -p -d /dev/media0
Media controller API version 0.1.0

Media device information
-----
driver           xilinx-video
model            Xilinx Video Composite Device
serial
bus info
hw revision      0x0
driver version   0.0.0

Device topology
- entity 1: vcap_hdmi output 0 (1 pad, 1 link)
    type Node subtype V4L flags 0
    device node name /dev/video3
    pad0: Sink
        <- "adv7611 3-004c":1 [ENABLED]

- entity 2: adv7611 3-004c (2 pads, 1 link)
    type V4L2 subdev subtype Unknown flags 0
    device node name /dev/v4l-subdev0
    pad0: Sink
        [dv.caps:BT.656/1120 ...
    pad1: Source
        [fmt:YUYV2X8/640x480]
        [dv.caps:BT.656/1120 ...
        [dv.detect:BT.656/1120 ...
        [dv.current:BT.656/1120 ...
        -> "vcap_hdmi output 0":0 [ENABLED]

root@zynq:~#

```

In order to validate the test pattern (TPG) capture pipeline, the media-ctl utility can be used:

```

root@zynq:~# media-ctl -p -d /dev/media1
Media controller API version 0.1.0

Media device information
-----
driver           xilinx-video
model            Xilinx Video Composite Device

```

```

serial
bus info
hw revision      0x0
driver version   0.0.0

Device topology
- entity 1: vcap_tpg output 0 (1 pad, 1 link)
    type Node subtype V4L flags 0
    device node name /dev/video4
    pad0: Sink
        <- "40080000.tpg":0 [ENABLED]

- entity 2: 40080000.tpg (1 pad, 1 link)
    type V4L2 subdev subtype Unknown flags 0
    device node name /dev/v4l-subdev7
    pad0: Source
        [fmt:UYVY/0x0]
        -> "vcap_tpg output 0":0 [ENABLED]

root@zynq:~#

```

In order to validate the PYTHON-1300-C capture pipeline, the media-ctl utility can be used:

```

root@zynq:~# media-ctl -p -d /dev/media2
Media controller API version 0.1.0

Media device information
-----
driver      xilinx-video
model       Xilinx Video Composite Device
serial
bus info
hw revision  0x0
driver version 0.0.0

Device topology
- entity 1: vcap_python output 0 (1 pad, 1 link)
    type Node subtype V4L flags 0
    device node name /dev/video7
    pad0: Sink
        <- "43c50000.cresample":1 [ENABLED]

- entity 2: PYTHON1300 (1 pad, 1 link)
    type V4L2 subdev subtype Unknown flags 0
    device node name /dev/v4l-subdev2
    pad0: Source
        [fmt:SRGB8/1280x1024]
        -> "PYTHON1300_RXIF":0 [ENABLED]

- entity 3: PYTHON1300_RXIF (2 pads, 2 links)

```



```

        type V4L2 subdev subtype Unknown flags 0
        device node name /dev/v4l-subdev3
    pad0: Sink
        [fmt:SRGB8/1280x1024]
        <- "PYTHON1300":0 [ENABLED]
    pad1: Source
        [fmt:SRGB8/1280x1024]
        -> "43c30000.cfa":0 [ENABLED]

- entity 4: 43c30000.cfa (2 pads, 2 links)
    type V4L2 subdev subtype Unknown flags 0
    device node name /dev/v4l-subdev4
    pad0: Sink
        [fmt:SRGB8/1280x1024]
        <- "PYTHON1300_RXIF":1 [ENABLED]
    pad1: Source
        [fmt:unknown/1280x1024]
        -> "43c40000.rgb2yuv":0 [ENABLED]

- entity 5: 43c50000.cresample (2 pads, 2 links)
    type V4L2 subdev subtype Unknown flags 0
    device node name /dev/v4l-subdev5
    pad0: Sink
        [fmt:unknown/1280x1024]
        <- "43c40000.rgb2yuv":1 [ENABLED]
    pad1: Source
        [fmt:UYVY/1280x1024]
        -> "vcap_python output 0":0 [ENABLED]

- entity 6: 43c40000.rgb2yuv (2 pads, 2 links)
    type V4L2 subdev subtype Unknown flags 0
    device node name /dev/v4l-subdev6
    pad0: Sink
        [fmt:unknown/1280x1024]
        <- "43c30000.cfa":1 [ENABLED]
    pad1: Source
        [fmt:unknown/1280x1024]
        -> "43c50000.cresample":0 [ENABLED]

root@zynq:~#

```

## Revision History

Date	Version	Revision
18 Apr 2015	2015.4.01	First Version, supporting PicoZed FMC Carrier V2, ZedBoard, and ZC702
19 May 2016	2015.4.02	Replace logiCLK core with axi_clkgen core