

## Overview

The Vivado HLS Reference Design provides a feature rich framework for the development of video applications on the Xilinx Zynq-7000 SoC. It is based on the Xilinx ZC702 Base TRD, which provides the following infrastructure:

- HDMI capture pipeline
- Co-Processing 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 Vivado HLS Reference Design.

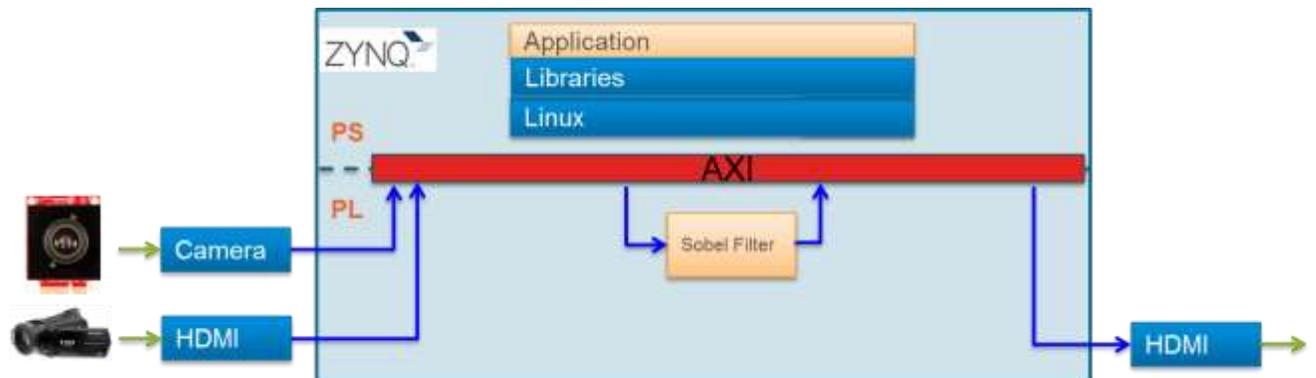


Figure 1 – FMC-HDMI-CAM + PYTHON-1300 – Hardware Block Diagram

## 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.

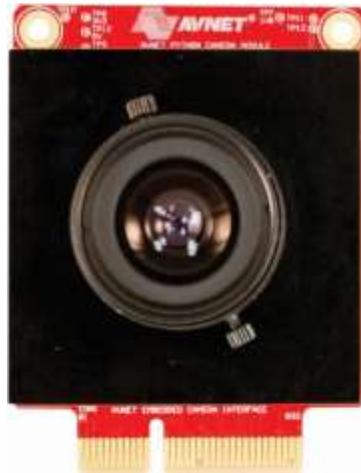


Figure 2 – PYTHON-1300-C Camera Module

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.

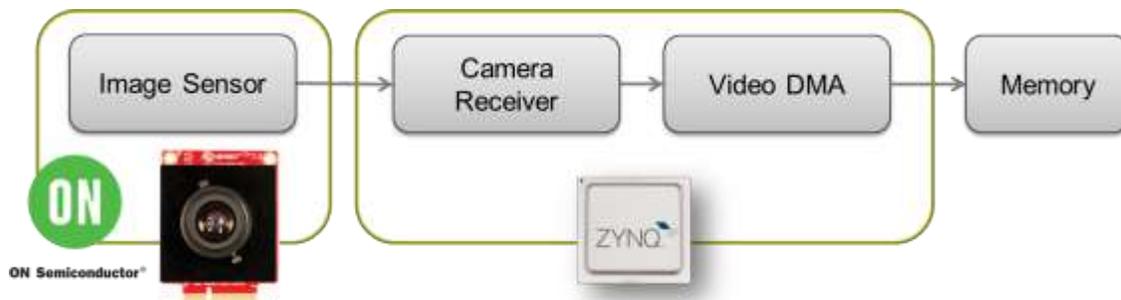


Figure 3 – PYTHON-1300 – Camera Receiver IP Core

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

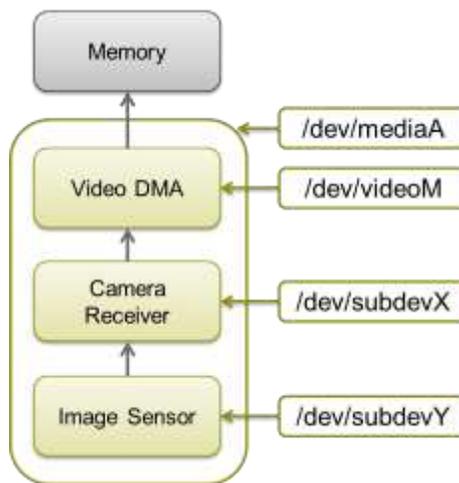


Figure 4 – PYTHON-1300 – Sub-device linux drivers

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

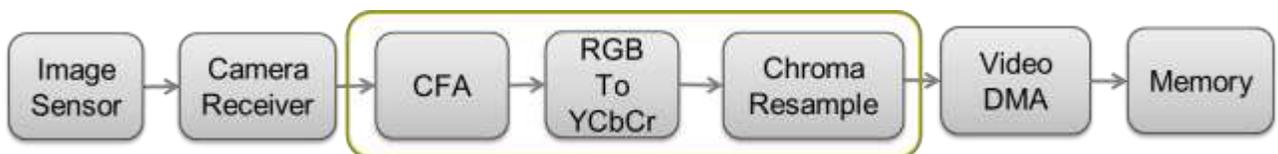


Figure 5 – PYTHON-1300 – Complete capture pipeline

---

## References to the ZC702 Base TRD

The Vivado HLS Reference Design is based on the Xilinx ZC702 Base TRD.

The following documentation should be consulted for more detailed information:

### 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)

### RDF 0286 – ZC702 ZVIK Base TRD - Design Files

[rdf0286-zc702-zvik-base-trd-2015-4.zip](http://www.xilinx.com/support/documentation/boards_and_kits/zc702_zvik/2015_4/rdf0286-zc702-zvik-base-trd-2015-4.zip)

### 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\_hls\_{target}\_2015\_4\_01.zip archive to the C:\ root directory, where {target} is one of pzfmc2, zed, or zc702.
2. You should see the following directory structure, with one of the {target} subdirectories:

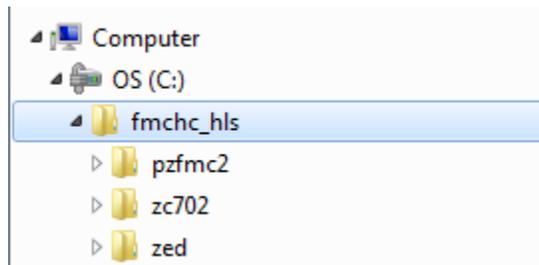


Figure 6 – Extracted C:\fmchc\_hls directory structure

**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\_hls directory will contain one of the following sub-directories:

Directory	Content Description
C:\fmchc_hls\pzfmc2	Contains the reference design for the PicoZed 7030 SOM + PicoZed FMC Carrier V2 hardware.
C:\fmchc_hls\zc702	Contains the reference design for the ZC702.
C:\fmchc_hls\zed	Contains the reference design for the ZedBoard.

Each {target} sub-directory (being one of pzfmc2, zc702 or zed) contains the following directory structure:

Directory	Content Description
C:\fmchc_hls\{target}\ready_to_test	Pre-built SD card image, ready to test on {target} hardware.
C:\fmchc_hls\{target}\hardware	Source files for hardware design
C:\fmchc_hls\{target}\software	Source files for software applications, as well as petalinux images

---

## Experiment 2: Executing pre-built design on hardware

This section will describe how to execute the pre-built binaries on one of the following FMC Carriers:

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

1. Copy the contents of the “ready\_for\_test” 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/bin/video_cmd

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 (OpenCV)  (*)

----- Toggle Filter Mode -----
5 : Filter OFF/SW/HW  (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' to 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 (OpenCV)  (*)

----- Toggle Filter Mode -----
5 : Filter OFF/SW/HW  (OFF)

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

PYTHON1300: PYTHON-1300 Sensor detected

Enter your choice :

```

## Selecting the Filter Mode

The VIDEO\_CMD application supports three filter modes:

Filter Mode	Description
OFF	No image processing
SW	Image processing performed in software
HW	Image processing performed in hardware (accelerated)

- To select the different filter modes (OFF, SW, HW), 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 (OpenCV)  (*)

----- Toggle Filter Mode -----
5 : Filter OFF/SW/HW  (SW)

----- 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/SW/HW (HW)

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

PYTHON1300: PYTHON-1300 Sensor detected

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/SW/HW (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:~#
```

## Experiment 3: 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

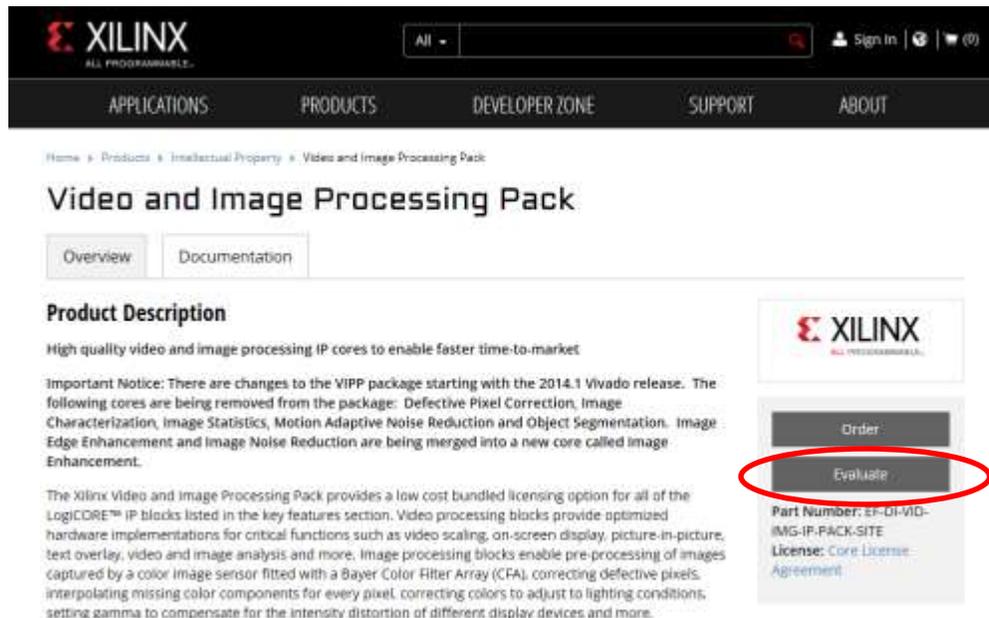


Figure 7 – Video and Image Processing Pack – product page

3. The generated license file is sent by email. Follow the enclosed instructions to add the evaluation license features for the logiCLK core.

## Experiment 4: Build the reference design

In this section, the Vivado project will be created and built with TCL scripts, implementing the FMC-HDMI-CAM + PYTHON-1300-C reference design for the selected FMC carrier.

### Create the Vivado project

1. From the Start menu, open the “Vivado 2015.4 TCL Shell” console
2. Change to the **C:\fmchc\_hls\{target}\hardware\vivado** directory, where {target} is one of the following targeted hardware platforms
  - a. **zc702** : ZC702 + FMC-HDMI-CAM + PYTHON-1300-C camera module
  - b. **zed** : ZedBoard + FMC-HDMI-CAM + PYTHON-1300-C camera module
  - c. **pzfm2** : PicoZed 7030 SOM + PicoZed FMC Carrier V2 + FMC-HDMI-CAM + PYTHON-1300-C camera module

```
Vivado% cd C:/fmchc_hls/pzfm2/hardware/vivado
Vivado%
```

Figure 8 – Vivado 2015.4 TCL Shell – Changing directory

3. Create the Vivado project with the following command :

```
Vivado% source ./scripts/create_project.tcl
...
```

Figure 9 – Vivado 2015.4 TCL Shell – Creating the Vivado project

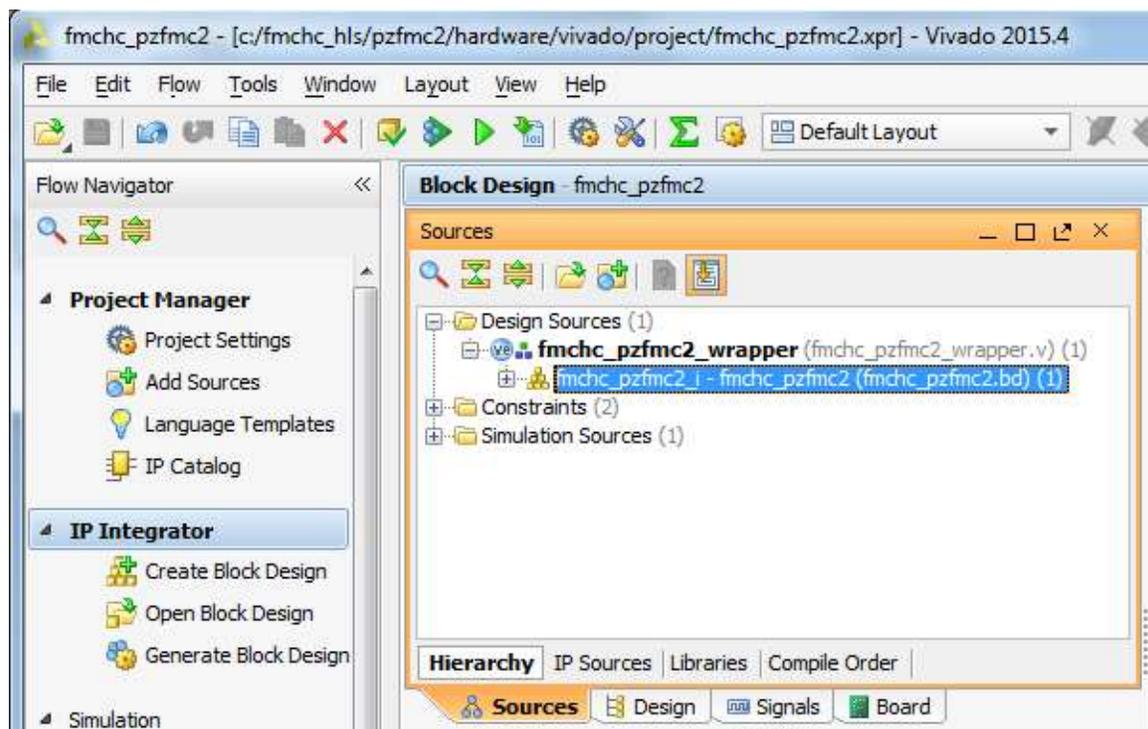
### Open the Vivado project

4. Open the Vivado GUI with the “start\_gui” command, as shown below:

```
...
# update_compile_order -fileset sources_1
# update_compile_order -fileset sim_1
# set_property strategy Performance_Explore [get_runs impl_1]
Vivado% start_gui
```

**Figure 10 – Vivado 2015.4 TCL Shell – Starting the Vivado GUI**

7. Once the Vivado 2015.4 GUI is open, use the Sources navigator to double-click on the `fmchc_{target}.bd` file to open the block design



**Figure 11 – Vivado 2015.4 GUI – Block Design**

The block design contains the following capture pipelines:

- **tpg\_input** : capture pipeline for the test pattern generator (TPG)
- **fmc\_hdmi\_input** : capture pipeline for the FMC's HDMI input interface
- **python1300c\_capture** : capture pipeline for the PYTHON-1300-C camera module

The block design also contains the following sub-modules for the display pipeline:

- **fmc\_hdmi\_clock** : clock generation for the video display pipeline
- **fmc\_hdmi\_output** : display pipeline for the FMC's HDMI output interface

Finally, the block design contains the memory to memory (co-processing) pipeline:

- **processing** : memory to memory pipeline with sobel co-processor

In addition to the previous video pipelines, the design also contains the following sub-modules which are required for the design:

- **processing\_system\_7** : sub-module representing the Zynq’s processing system
- **axi\_interconnect\_gp0** : AXI4-Lite interconnect to configure the various cores
- **axi\_iic** : AXI IIC controller for the FMC-HDMI-CAM’s I2C peripherals
- **gpio** : general purpose I/O used by design
- **interrupts** : concatenation of all interrupts used by design
- **axi\_interconnect\_hp0** : high-performance AXI-4 interconnect for video capture & display pipelines
- **axi\_interconnect\_hp2** : high-performance AXI-4 interconnect for co-processor.
- **axi\_perf\_mon** : AXI performance monitor

### FMC-HDMI-CAM + PYTHON, v2015.4

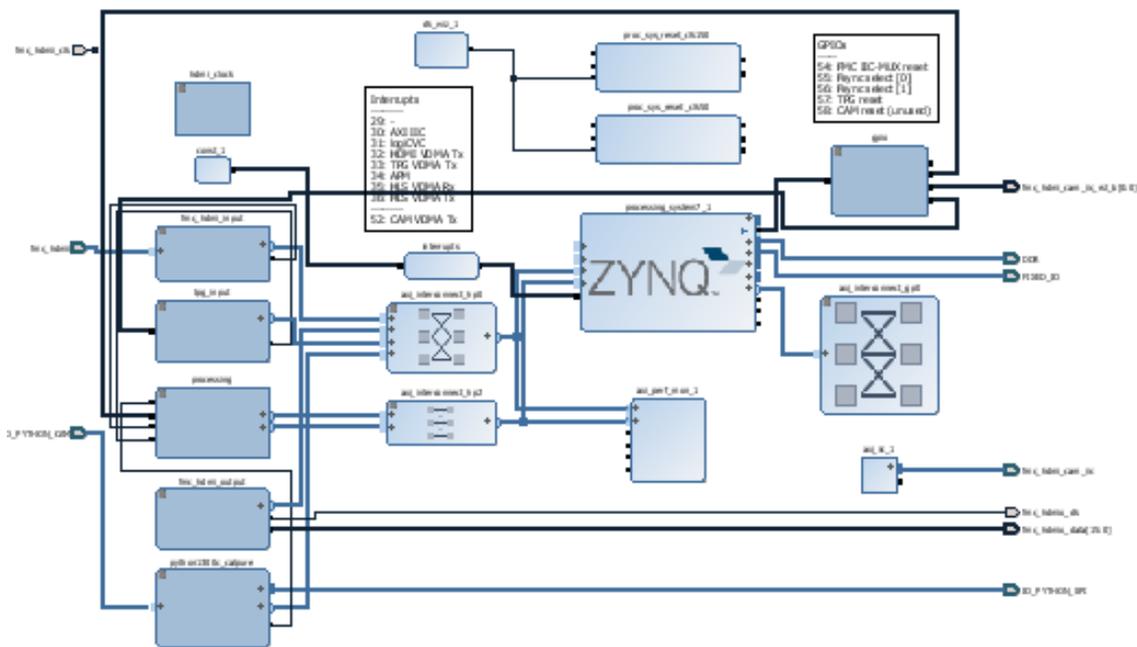


Figure 12 – FMC-HDMI-CAM + PYTHON - Block Design

You will notice that not all connections are visible in the block design viewer. This makes the block design easier to visualize and navigate.

If you prefer to view all the connections, click on the “Block Design Options” icon in the Diagram viewer. Other options, such as which type of connections (clock, reset, etc...) are visible or not, can also be configured within the “Block Design Options”.

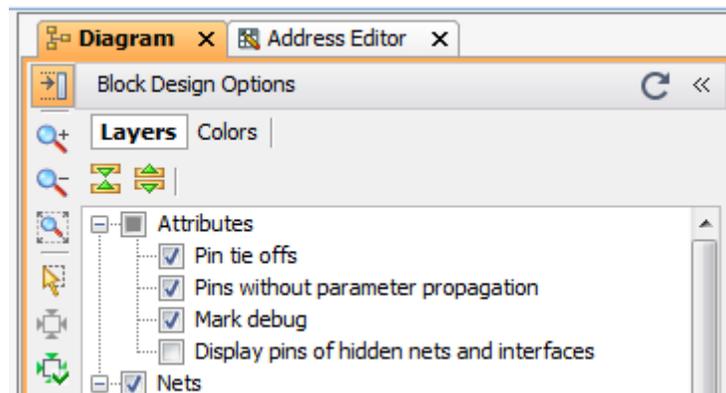


Figure 13 – Vivado 2015.4 GUI – Block Design Options

## Build the Vivado project

8. In the Flow Navigator, click on “Generate Bitstream” to build the Vivado project.

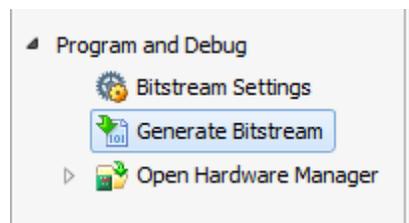
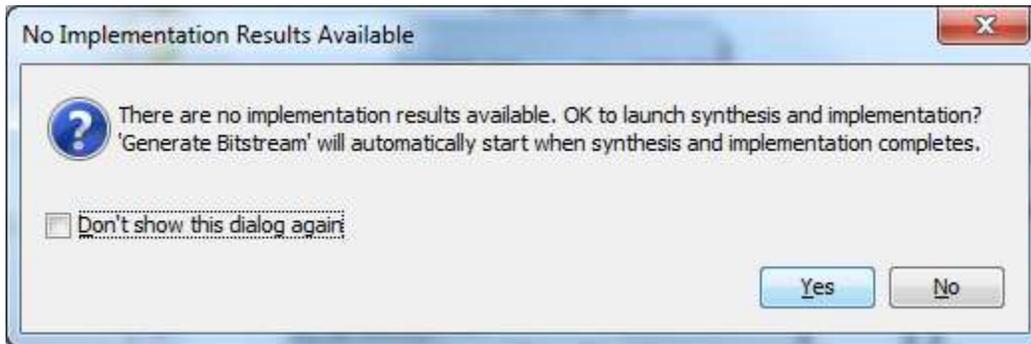


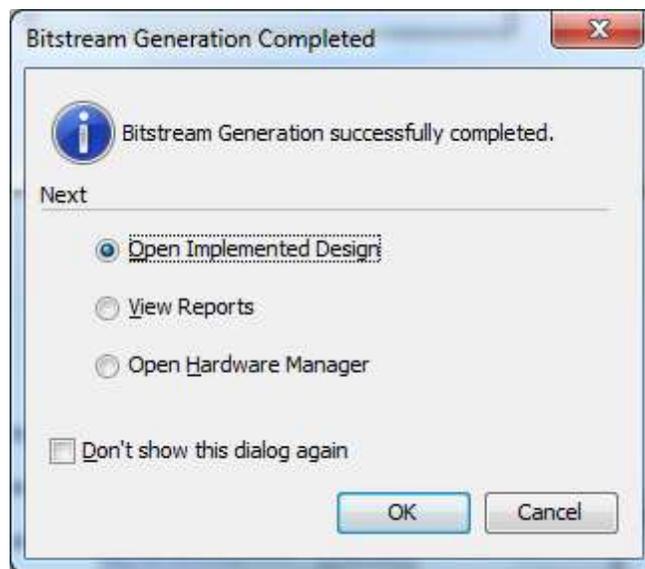
Figure 14 – Vivado 2015.4 GUI – Generate Bitstream

9. If asked if OK to launch synthesis and implementation, click OK.



**Figure 15 – Vivado 2015.4 GUI – OK to launch synthesis and implementation ?**

10. If asked what to do Next, after the Bitstream Generation successfully completed, click OK.



**Figure 16 – Vivado 2015.4 GUI – Bitstream Generation Completed**

At this point, you have generated the hardware portion of the design (ie. bitstream). The next sections will indicate where to find instructions on how to build the software portion of the design.

## Exporting the hardware platform for use with Xilinx SDK

Please refer to Xilinx's ZC702 Base TRD" wiki page for instructions on how to export the hardware design for use with Xilinx SDK.

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

## Re-building the SOBEL co-processor with Vivado HLS

Please refer to Xilinx's ZC702 Base TRD" wiki page for instructions on how to re-build the SOBEL co-processor with Vivado HLS.

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

## Exporting the hardware platform for use with Xilinx SDK

Please refer to Xilinx's ZC702 Base TRD" wiki page for instructions on how to export the hardware design for use with Xilinx SDK.

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

## Rebuilding the Petalinux images

Please refer to Xilinx's ZC702 Base TRD" wiki page for instructions on how to re-build the Petalinux images.

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

Instructions specific to this version of the design can be found in the hardware target's specific directory.

<b>Petalinux build instructions</b>
C:\fmchc_hls\pzfmc2\software\how-to-build-petalinux-image.txt
C:\fmchc_hls\zc702\software\how-to-build-petalinux-image.txt

C:\fmchc\_hls\zed\software\how-to-build-petalinux-image.txt

## Rebuilding the linux applications

Please refer to Xilinx's ZC702 Base TRD" wiki page for instructions on how to re-build the linux applications.

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

The command line application (VIDEO\_CMD) can be compiled on a Windows machine.

The graphical application (VIDEO\_QT) must be compile on a Linux machine. The following Qt/Qwt build instructions must be performed prior to compiling the VIDEO\_QT application.

<http://www.wiki.xilinx.com/Qt+%26+Qwt+Build+Instructions+%28Qt+5.4.2%2C+Qwt+6.1.2%29>

## 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
```

19 May 2016

```
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/media3
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)
```

19 May 2016

```
type V4L2 subdev subtype Unknown flags 0
device node name /dev/v4l-subdev3
pad0: Sink
    [fmt:SRGGB8/1280x1024]
    <- "PYTHON1300":0 [ENABLED]
pad1: Source
    [fmt:SRGGB8/1280x1024]
    -> "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:SRGGB8/1280x1024]
      <- "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:~#
```

In order to validate the memory to memory (co-processing) 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: vm2m_hls output 0 (1 pad, 1 link)
      type Node subtype V4L flags 0
      device node name /dev/video5
      pad0: Sink
            <- "400d0000.hls":1 [ENABLED]

- entity 2: vm2m_hls input 1 (1 pad, 1 link)
      type Node subtype V4L flags 0
      device node name /dev/video6
      pad0: Source
            -> "400d0000.hls":0 [ENABLED]

- entity 3: 400d0000.hls (2 pads, 2 links)
      type V4L2 subdev subtype Unknown flags 0
      device node name /dev/v4l-subdev1
      pad0: Sink
            [fmt:UYVY/0x0]
            <- "vm2m_hls input 1":0 [ENABLED]
      pad1: Source
            [fmt:UYVY/0x0]
            -> "vm2m_hls output 0":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 2015	2015.4.02	Replace logiCLK core with axi_clkgen core