Overview

The FMC-IMAGEON Getting Started Reference Design illustrates the following capabilities of the FMC-IMAGEON FMC module:

•   Driving video content on the HDMI output interface
•   Receiving video content from the HDMI input interface
•   Receiving video content from the VITA-2000 image sensor

**Figure 1 – ON Semiconductor Image Sensor with HDMI Input/Output FMC Bundle**

Objectives

This tutorial will guide the user how to:

• Retrieve the design files from the public Avnet git repository
• Build the reference design
• Execute the reference design on hardware

Reference Design Overview

The example design uses the Zynq processing system (PS) to initialize the VITA-2000-C camera, the HDMI input interface, as well as the HDMI output interface. The design also implements a simple image sensor pipeline (ISP) and video frame buffer inside the programmable logic (PL).

The following figure illustrates the block diagram for the programmable logic (PL) hardware implementation.
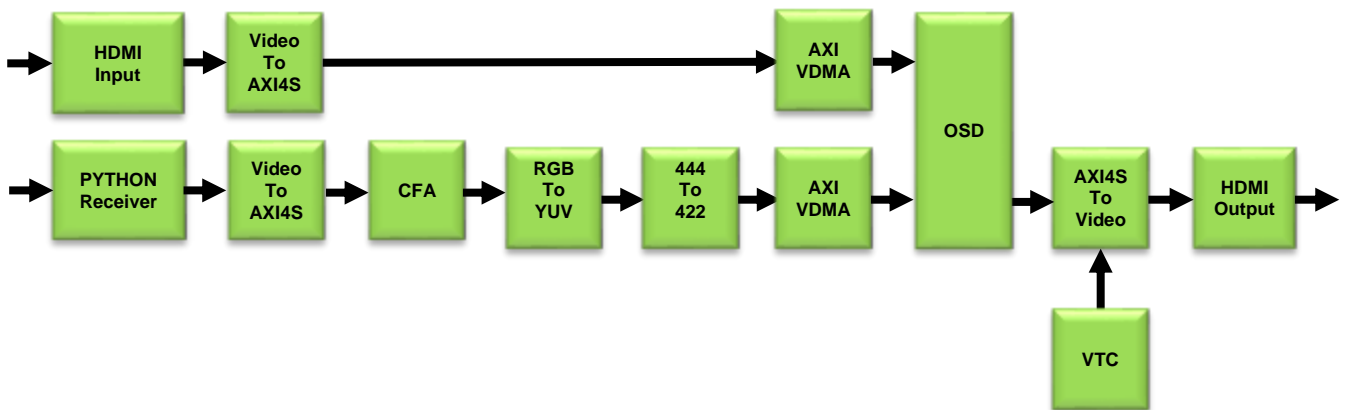
Figure 2 – FMC-IMAGEON Getting Started Reference Design – Hardware Block Diagram

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

- Color Filter Array Interpolation (CFA) v7.0
- Chroma Resampler v4.0
- Video On Screen Display (OSD) v6.0
- RGB to YcrCb Color-Space Converter v7.1
- Video Timing Controller (VTC) v6.1

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:

- Windws-7 64-bit
- Terminal Emulator (HyperTerminal or TeraTerm)
- Xilinx Vivado Design Suite 2015.4
- MicroZed Board Definition Install for Vivado 2015.4
  - http://www.microzed.org/support/documentation/1519

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[1]
- One of the following supported FMC carriers:
  - ZC702
  - ZedBoard
  - MicroZed 7020 SOM + FMC Carrier Card
- ON Semiconductor Image Sensor with HDMI Input/Output FMC Bundle, including:
  - FMC-IMAGEON FMC module
  - VITA-2000-C Camera module (optional)
- HDMI (or DVI-D) video source
- HDMI monitor (1080P60 capable)
- USB cable (Type A to Micro-USB Type B)
- 4GB MicroSD card

---

[1] Refer to http://www.xilinx.com/design-tools/vivado/memory.htm

Experiment 1: 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
- Chroma Resampler v4.0
- Video On Screen Display (OSD) v6.0
- RGB to YcrCb Color-Space Converter v7.1
- Video Timing Controller (VTC) v6.1

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,
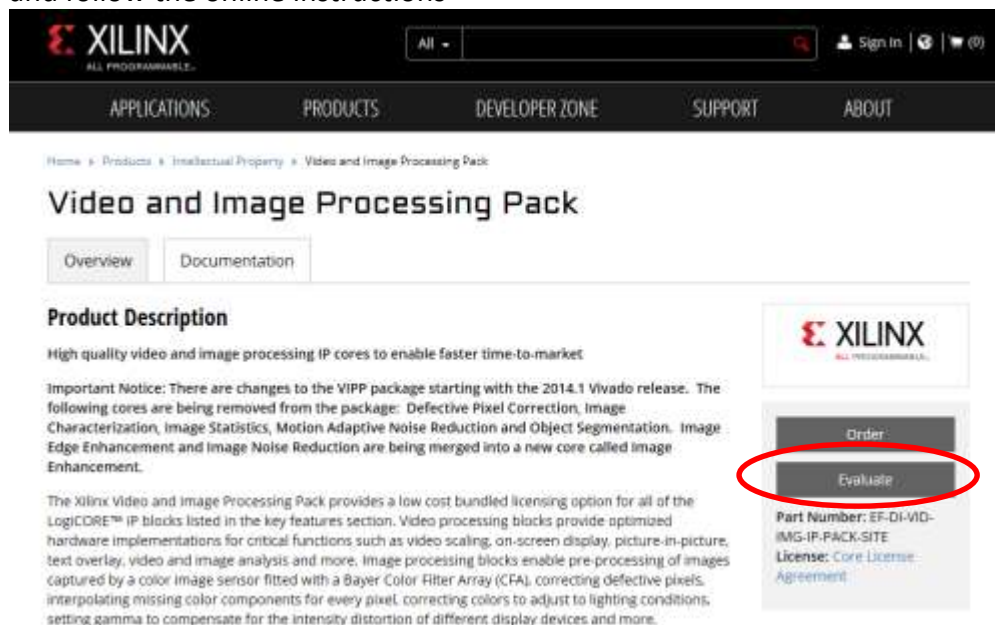   and follow the online instructions

Figure 3 – 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 Video and Image Processing Pack.

Experiment 2: Retrieve the design files

In this section, the design files for the reference design will be retrieved from the Avnet git repository.

1. Navigate to the following web site : https://github.com/Avnet/hdl
2. Click the **branch:master** button
3. Specify the following search criteria : fmc_imageon_gs
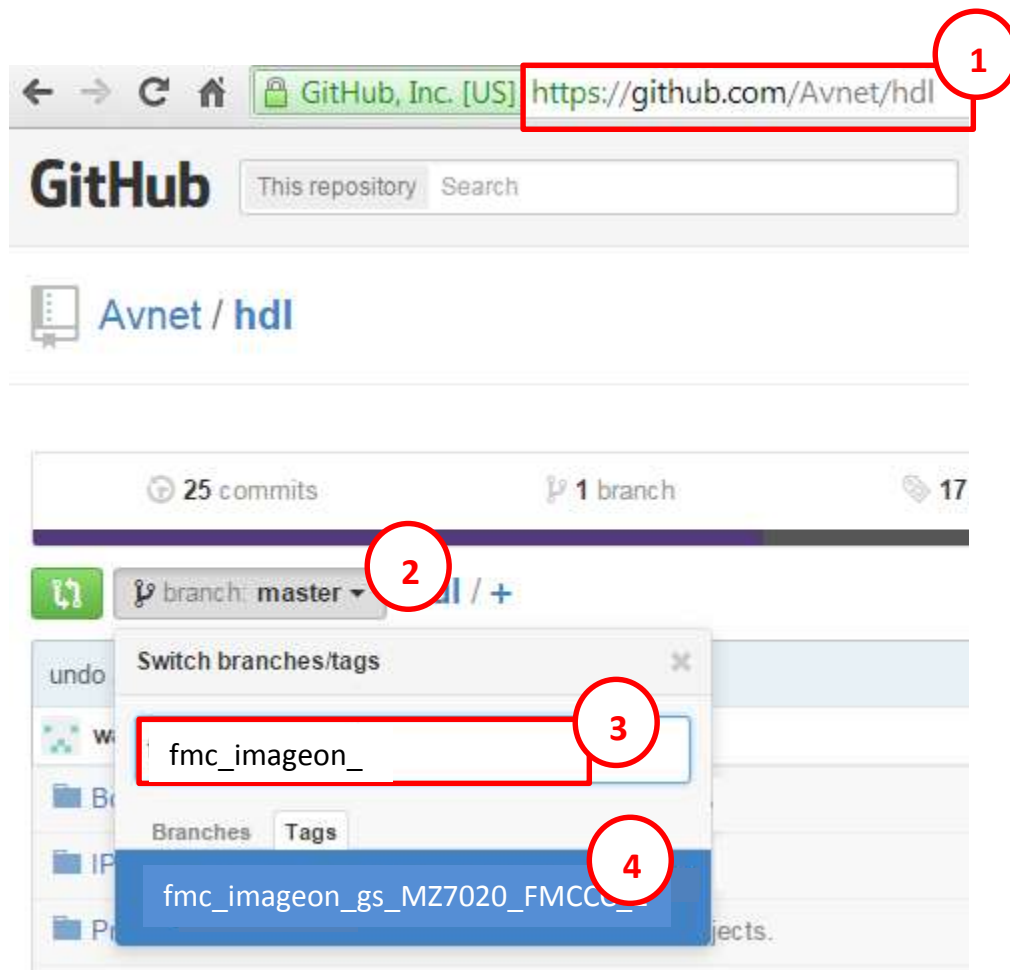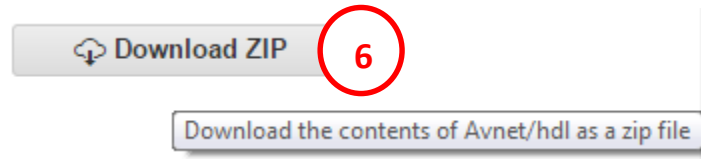4. Click the **Tags** tab

Figure 4 – Avnet GitHub repository – Retrieving specific version with tag

5. Select the **fmc_imageon_gs_MZ7020_FMCCC_20160225_161355** tag
   This will retrieve a known working version of the design files for the FMC-IMAGEON Getting Started reference design.

6.   Click the Download ZIP file button

Figure 5 – Avnet GitHub repository – Download ZIP

7.   Create an "Avnet" directory in your root C:\ drive
8.   Save **hdl-fmc_imageon_gs_MZ7020_FMCCC_20160225_161355.zip** file to the
     C:\Avnet directory, and extract the contents of the zip file in this directory
9.   Rename the "hdl-fmc_imageon_gs_MZ7020_FMCCC_20160225_161355"
     directory to "hdl"
     You should see the following directory structure
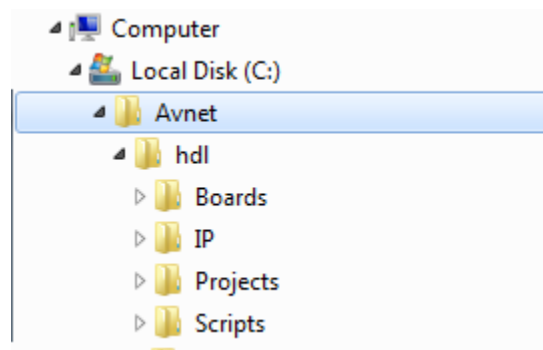
Figure 6 – Extracted C:\Avnet\hdl directory structure

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:\Avnet\hdl** repository contains the following sub-directories:

Directory	Content Description
C:\Avnet\hdl**Boards**	contains board related files
C:\Avnet\hdl**IP**	contains the IP cores used by the ref designs
C:\Avnet\hdl**Projects**	contains project related files
C:\Avnet\hdl**Scripts**	contains scripts used to automatically build the designs

For the FMC-IMAGEON Getting Started reference design, the following content is of interest:

Directory	Content Description
C:\Avnet\hdl\IP**avnet_hdmi_in** C:\Avnet\hdl\IP**avnet_hdmi_out**	IP cores (including HDL source) for the HDMI input/output interfaces including embedded sync code insertion/detection
C:\Avnet\hdl\IP**onsemi_vita_spi**	IP core (including HDL source) for the SPI controller for use with the VITA/PYTHON image sensors
C:\Avnet\hdl\IP**onsemi_vita_cam**	IP core (including HDL source) for the VITA/PYTHON camera receiver
C:\Avnet\hdl\Projects**fmc_imageon_gs**	files for the FMC-IMAGEON Getting Started reference design
C:\Avnet\hdl\Scripts\ **make_fmc_imageon_gs.tcl**	script to build the FMC-IMAGEON Getting Started reference design

By default, the script will build the design for the ZC702, ZEDBOARD, and MicroZed7020 + FMC Carrier Card.

1. Edit the **make_fmc_imageon_gs.tcl** script to only build for your FMC carrier.
   As an example, if you have a MicroZed 7020 SOM + FMC carrier card,
   comment out the build for the ZC702 and ZEDBOARD, as shown below:

```
# Build FMC-IMAGEON + VITA-2000-C Getting Started design for the ZC702
#set argv [list board=ZC702 project=fmc_imageon_gs sdk=yes]
#set argc [llength $argv]
#source ./make.tcl -notrace

# Build FMC-IMAGEON + VITA-2000-C Getting Started design for the ZedBoard
#set argv [list board=ZEDBOARD project=fmc_imageon_gs sdk=yes]
#set argc [llength $argv]
#source ./make.tcl -notrace

# Build FMC-IMAGEON + VITA-2000-C Getting Started design
# for the MicroZed-7020 + FMC Carrier Card
set argv [list board=MZ7020_FMCCC project=fmc_imageon_gs sdk=yes]
set argc [llength $argv]
source ./make.tcl -notrace
```

Figure 7 – Editing the make script to build for MicroZed FMC Carrier Card

Experiment 3: Build the reference design

In this section, the Vivado project will be created and built with TCL scripts, implementing the FMC-IMAGEON Getting Started reference.

1. From the Start menu, open the "Vivado 2015.4 TCL Shell" console

2. Change to the **C:\Avnet\hdl\Scripts** directory

```
****** Vivado v2015.4 (64-bit)
  **** SW Build 1412921 on Wed Nov 18 09:43:45 MST 2015
  **** IP Build 1412160 on Tue Nov 17 13:47:24 MST 2015
    ** Copyright 1986-2015 Xilinx, Inc. All Rights Reserved.

Vivado% cd C:/Avnet/hdl/Scripts
Vivado%
```

Figure 8 – Vivado 2015.4 TCL Shell – Changing to C:/Avnet/hdl/Scripts directory

3. Launch the build with the "**source ./make_fmc_imageon_gs.tcl**" command

```
Vivado% source ./make_embv_python1300c_fb.tcl
# set argv [list board=MZ7020_FMCCC project=fmc_imageon_gs sdk=yes
version_override=yes]
# set argc [llength $argv]
# source ./make.tcl -notrace


*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
*-                                                  -*
*-         Welcome to the Avnet Project Builder      -*
*-                                                  -*
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*


+-----------------+---------------------------------+
| Setting         |      Configuration              |
+-----------------+---------------------------------+
| Board           |      MZ7020_FMCCC               |
+-----------------+---------------------------------+
| Project         |      fmc_imageon_gs             |
+-----------------+---------------------------------+
| SDK             |      yes                         |
+-----------------+---------------------------------+
| Version override |      yes                         |
+-----------------+---------------------------------+
```

Figure 9 – Vivado 2015.4 TCL Shell – Launching the build

As a convenience, before building the hardware design, the scripts will verify if valid licenses are installed for the video IP cores used in the design.

```
***** Check for Video IP core licenses...

+----------------+----------------------------------+
| Video IP Core  | License Status                   |
+----------------+----------------------------------+
| v_cfa          | VALID (Hardware Evaluation)      |
+----------------+----------------------------------+
| v_cresample    | VALID (Hardware Evaluation)      |
+----------------+----------------------------------+
| v_osd          | VALID (Hardware Evaluation)      |
+----------------+----------------------------------+
| v_rgb2ycrcb    | VALID (Full License)             |
+----------------+----------------------------------+
| v_tc           | VALID (Full License)             |
+----------------+----------------------------------+
```

Figure 10 – Vivado 2015.4 TCL Shell – Video IP Core license verification

Each of the video IP cores requires a full license or hardware evaluation license in order to successfully build a bitstream.

The build will perform the following steps, where {BOARD} will be one of ZC702, ZEDBOARD, or MZ7020_FMCCC:

- Create and build the hardware design with Vivado 2015.4,
  including the IP Integrator block design

  C:\Avnet\hdl\Projects\fmc_imageon_gs\{BOARD}**fmc_imageon_gs.xpr**

- Create and build the SDK workspace, including board support package (BSP),
  software application, and first stage boot loader (FSBL)

  C:\Avnet\hdl\Projects\fmc_imageon_gs\{BOARD}**fmc_imageon_gs.sdk**

- Create the SD card image (BOOT.bin)

  C:\Avnet\hdl\Projects\fmc_imageon_gs\{BOARD}**BOOT.bin**

Experiment 4: Execute the reference design on hardware

This section describes how to execute the reference design on the hardware.

For instructions on how to setup the hardware, please refer to the Getting Started Guide for your FMC carrier, and the FMC-IMAGEON + VITA-2000.

Booting from SD card image

The BOOT.bin SD card image created in the previous experiment can be used to execute the reference design on hardware.

For more detailed instructions on how to boot from the SD card, please refer to the Getting Started Guide for your FMC carrier card.

Booting from JTAG with SDK

The hardware and software can also be loaded to hardware using SDK 2015.4 and a JTAG emulator.

For more detailed instructions on how to boot from the JTAG, please refer to the Getting Started Guide for your FMC carrier card, then open the SDK workspace.

1. Launch SDK 2015.4, and specify the following directory for the SDK workspace:

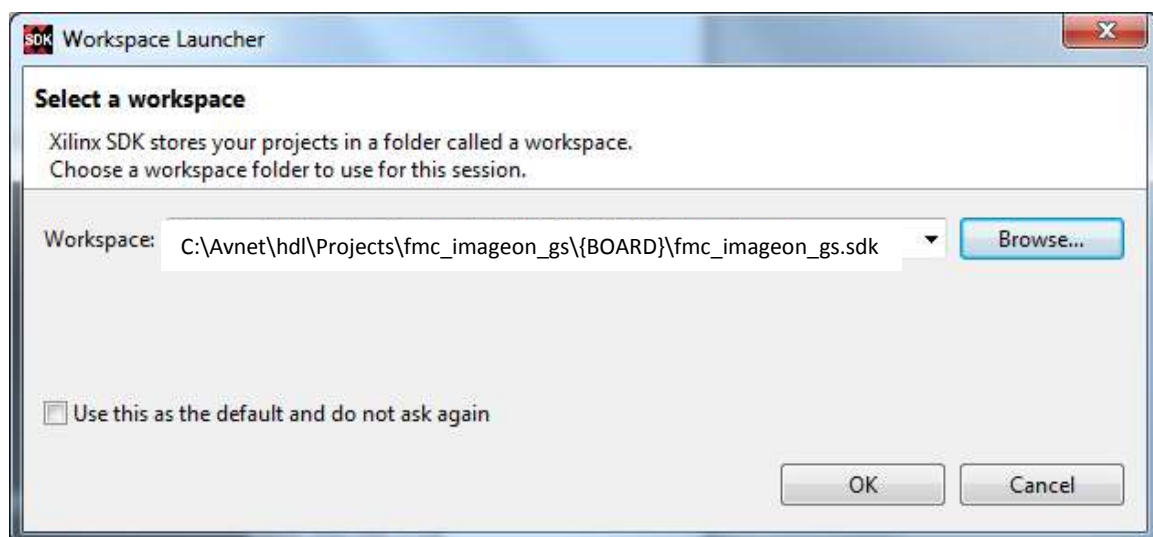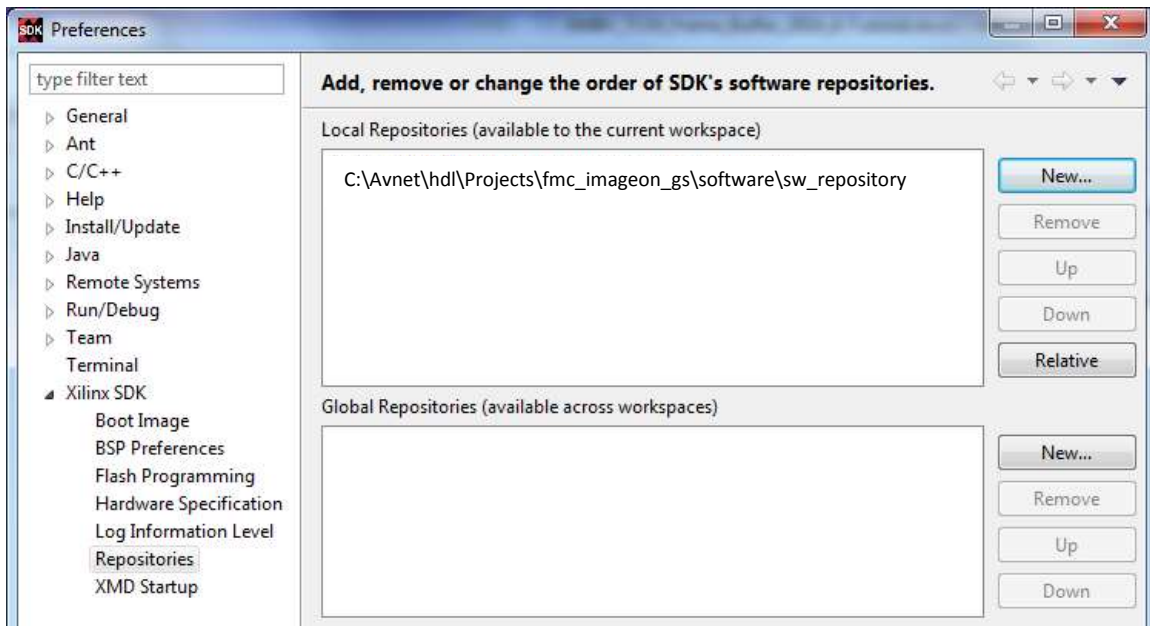   C:\Avnet\hdl\Projects\fmc_imageon_gs\{BOARD}**fmc_imageon_gs.sdk**

Figure 11 – SDK – Specifying SDK workspace

2. Close the Welcome Window

3. In the SDK menu, select **Xilinx Tools** => **Repositories**

4. Verify that the following Local Repository is specified:

    C:\Avnet\hdl\Projects\fmc_imageon_gs\software\sw_repository

Figure 12 – SDK – Specifying local repository

5. If the local repository is not specified, click on the **New** button,
    navigate to the following directory, then click **OK**.

    C:\Avnet\hdl\Projects\fmc_imageon_gs\software\sw_repository

6. When done, click **OK**.

NOTE : The local repository was not saved in the SDK workspace due to a limitation of
the SDK's scripting mode.

Now that the SDK workspace is correctly configured, the hardware and software can be
loaded and executed on the hardware.

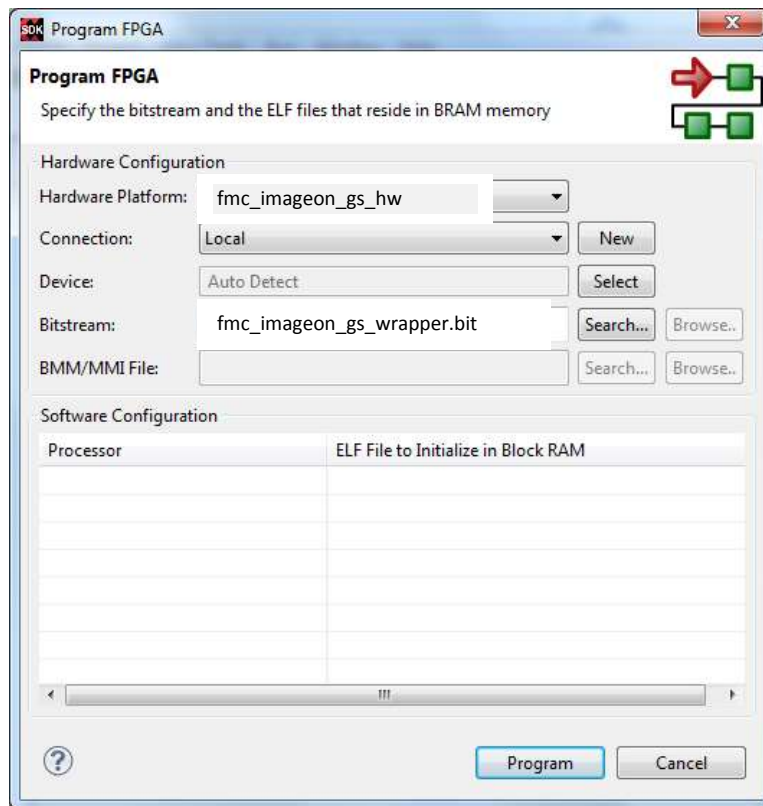7. In the SDK menu, select **Xilinx Tools** => **Load FPGA**

Figure 13 – SDK – Program FPGA

8. Click the **Program** button.
    It will take approximately 10 seconds to program the bitstream to hardware
9. Right-click **fmc_imageon_gs_app**
    and select **Run as** => Run **Configurations**.

10. Click **Xilinx C/C++ Application (GDB)** and click **New launch configurations**.

11. The new run configuration is created named **fmc_imageon_gs_app Debug**
    The configurations associated with application are pre-populated in the main tab
    of these launch configurations

12. Click on the **Application** tab

13. Next to the Application: edit box, click the **Search** button.

14. Select the **fmc_imageon_gs_app.elf** application, then click **OK**.

15. Click **Apply** and then **Run**.

16. If you get a Reset Status dialog box indicating that the current launch will reset the entire system, click **OK**.

17. You should see text on the serial console, as shown in the next section.

If you have a VITA-2000 camera module, you will observe the content captured by the VITA-2000-C image sensor on the DVI/HDMI monitor.

You have successfully executed the FMC-IMAGEON + VITA-2000 getting started design on hardware !

Using the text based serial console

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

```
-------------------------------------------------------
--                      FMC-IMAGEON                 --
--              Getting Started Design              --
-------------------------------------------------------

FMC-IMAGEON Initialization ...
Video Clock Synthesizer Configuration ...
HDMI Output Initialization ...
FMC-IMAGEON VITA Receiver Initialization ...
FMC-IMAGEON VITA SPI Config for 10MHz ...
FMC-IMAGEON VITA Initialization ...
FMC-IMAGEON VITA Configuration for 1080P60 timing ...
VITA Status =
        Image Width  = 1920
        Image Height = 1080
        Frame Rate   = 60 frames/sec
        CRC Status   = 0
CFA Initialization
VDMA 0 Initialization
VDMA 1 Initialization
OSD Initialization (hdmi=0x00, cam=0xFF)


-------------------------------------------------------
--                      FMC-IMAGEON                 --
--              Getting Started Design              --
-------------------------------------------------------
General Commands:
        help         Print the Top-Level menu Help Screen
        verbose on   Enable verbose
        verbose off  Disable verbose
Getting Started Commands
        start        start and select video source (hdmi|cam)

---------------------------------------------

FMC_IMAGEON_GS>
```

Figure 14 – FMC-IMAGEON Getting Started Design – Serial Console Output

If you have a VITA-2000-C camera module, you will observe the content captured by the VITA-2000-C image sensor on the DVI/HDMI monitor.   If you do not have a camera module, you will see green content, the default value of pixels in the camera frame buffer.

Getting Help for commands

At any time, you can type the "help" command to display the list of commands supported by the demonstration.

```
FMC_IMAGEON_GS>help

-------------------------------------------------------
--                      FMC-IMAGEON                 --
--               Getting Started Design             --
-------------------------------------------------------
General Commands:
        help         Print the Top-Level menu Help Screen
        verbose on   Enable verbose
        verbose off  Disable verbose
Getting Started Commands
        start        start and select video source (hdmi|cam)


---------------------------------------------

FMC_IMAGEON_GS>
```

For more detailed information on a particular command, type the command followed by the "help" argument as follows:

```
FMC_IMAGEON_GS>start help
        Syntax :
                start cam  => Start CAM video source
                start hdmi => Start HDMI video source

FMC_IMAGEON_GS>
```

Starting the HDMI video pipeline

To enable the HDMI pipeline (HDMI input => HDMI output), press the "**start hdmi**" command. This will initialize the HDMI input interface, the corresponding Video DMA core, as well as configure the OSD core to display the HDMI video path.

If the resolution of the video source is 1080P, the content will fill the entire 1080P resolution output.

```
FMC_IMAGEON_GS>start hdmi
HDMI Input Initialization
Waiting for ADV7611 to locked on incoming video ...
        ADV7611 Video Input LOCKED
        Input resolution = 1920 X 1080
VDMA 0 Initialization
VDMA 1 Initialization
OSD Initialization (hdmi=0xFF, cam=0x00)

FMC_IMAGEON_GS>
```

If the resolution of the video source is less than 1080P, the video source will be displayed in the top-left portion of a 1080P resolution output.

```
FMC_IMAGEON_GS>start hdmi
HDMI Input Initialization
Waiting for ADV7611 to locked on incoming video ...
        ADV7611 Video Input LOCKED
        Input resolution = 1280 X 720
VDMA 0 Initialization
VDMA 1 Initialization
OSD Initialization (hdmi=0xFF, cam=0x00)

FMC_IMAGEON_GS>
```

Starting the Camera video pipeline

To enable the camera pipeline (VITA input => HDMI output), press the "**start vita**" command. This will initialize the VITA-2000 image sensor, the corresponding Video DMA core, as well as configure the OSD core to display the VITA-2000 video path.

```
FMC_IMAGEON_GS>start cam
FMC-IMAGEON VITA Receiver Initialization ...
FMC-IMAGEON VITA SPI Config for 10MHz ...
FMC-IMAGEON VITA Initialization ...
FMC-IMAGEON VITA Configuration for 1080P60 timing ...
VITA Status =
        Image Width  = 1920
        Image Height = 1080
        Frame Rate   = 60 frames/sec
        CRC Status   = 0
CFA Initialization
VDMA 0 Initialization
VDMA 1 Initialization
OSD Initialization (hdmi=0x00, cam=0xFF)

FMC_IMAGEON_GS>
```

Enabling Verbose

Additional verbose can be enabled with the "verbose on" command. This is useful for diagnostic purposes, when needed.

```
FMC_IMAGEON_GS>verbose on
        verbose = on

FMC_IMAGEON_GS>
```

The expected output, in verbose mode, for the camera input is shown below:

```
FMC_IMAGEON_GS>start cam
        start cam
Video Frame Buffer Initialization ...
FMC-IMAGEON VITA Receiver Initialization ...
FMC-IMAGEON VITA SPI Config for 10MHz ...
FMC-IMAGEON VITA Initialization ...
VITA SYNCGEN - Setting Video Timing
```

```
        HSYNC  Timing        = hav=1920, hfp,=88, hsw=44 (hsp=1),
hbp=132 (x4)
        VSYNC  Timing        = hav=1081, hfp,=04, hsw=05 (hsp=1),
hbp=300 (x4)
VITA ISERDES - Setting Training Sequence to 0x000003A6
VITA ISERDES - Setting Manual Tap to 0x00000019
VITA DECODER - Configuring Sync Codes
VITA REMAPPER - Configuring for image lines in normal mode
VITA REMAPPER - Control = 0x00000001
VITA ISERDES - Asserting Reset
VITA DECODER - Asserting Reset
VITA CRC - Asserting Reset
VITA SPI Sequence 0 - Assert RESET_N pin
VITA ISERDES - Releasing Reset
VITA DECODER - Releasing Reset
VITA CRC - Releasing Reset
VITA SPI Sequence 0 - Releasing RESET_N pin
        VITA_SPI[0x0000] => 0x5614
        VITA-2000 Sensor detected
VITA SPI Sequence 1 - Enable Clock Management - Part 1
        VITA_SPI[0x0002] <= 0x0000
        VITA_SPI[0x0020] <= 0x2004
        VITA_SPI[0x0014] <= 0x0000
        VITA_SPI[0x0011] <= 0x2113
        VITA_SPI[0x001A] <= 0x2280
        VITA_SPI[0x001B] <= 0x3D2D
        VITA_SPI[0x0008] <= 0x0000
        VITA_SPI[0x0010] <= 0x0003
VITA SPI Sequence 2 - Verify PLL Lock Indicator
        VITA_SPI[0x0018] => 0x0001
VITA SPI Sequence 3 - Enable Clock Management - Part 2
        VITA_SPI[0x0009] <= 0x0000
        VITA_SPI[0x0020] <= 0x2006
        VITA_SPI[0x0022] <= 0x0001
VITA SPI Sequence 4 - Required Register Upload
        VITA_SPI[0x0029] <= 0x0000
        VITA_SPI[0x0081] => 0xC001
                           0x2000
        VITA_SPI[0x0081] <= 0xC001
        VITA_SPI[0x0041] <= 0x288B
        VITA_SPI[0x0042] <= 0x53C6
        VITA_SPI[0x0043] <= 0x0344
        VITA_SPI[0x0044] <= 0x0085
        VITA_SPI[0x0046] <= 0x4888
        VITA_SPI[0x0051] <= 0x86A1
        VITA_SPI[0x0080] <= 0x460F
        VITA_SPI[0x00B0] <= 0x00F5
        VITA_SPI[0x00B4] <= 0x00FD
        VITA_SPI[0x00B5] <= 0x0144
        VITA_SPI[0x00C2] <= 0x0404
        VITA_SPI[0x00DA] <= 0x160B
        VITA_SPI[0x00E0] <= 0x3E13
        VITA_SPI[0x0187] <= 0x1010
        VITA_SPI[0x01C8] <= 0x0386
VITA SPI Sequence 5 - Soft Power-Up
```

```
            VITA_SPI[0x0020] <= 0x2007
            VITA_SPI[0x000A] <= 0x0000
            VITA_SPI[0x0040] <= 0x0001
            VITA_SPI[0x0048] <= 0x0203
            VITA_SPI[0x0028] <= 0x0003
            VITA_SPI[0x0030] <= 0x0001
            VITA_SPI[0x0070] <= 0x0007
VITA ISERDES - Asserting Reset
VITA DECODER - Asserting Reset
VITA CRC - Asserting Reset
VITA ISERDES - Releasing Reset
VITA DECODER - Releasing Reset
VITA CRC - Releasing Reset
VITA ISERDES - Status = 0x61610100
VITA ISERDES - Status = 0x61610100
VITA ISERDES - Waiting for CLK_RDY to assert
VITA ISERDES - Status = 0x61610100
VITA ISERDES - Align Start
VITA ISERDES - Waiting for ALIGN_BUSY to assert
VITA ISERDES - Status = 0x61610304
VITA ISERDES - Waiting for ALIGN_BUSY to de-assert
VITA ISERDES - Status = 0x61610100
VITA ISERDES - Enabling FIFO enable
VITA DECODER - Enabling Sync Channel Decoder
VITA DECODER - Control = 0x00000002
VITA CRC - Status = 0x0000000F
VITA CRC - Status = 0x0000000F
VITA CRC - Status = 0x0000000F
VITA CRC - Status = 0x0000000F
VITA CRC - Status = 0x0000000F
VITA CRC - Status = 0x0000000F
VITA CRC - Status = 0x0000000F
VITA CRC - Status = 0x0000000F
VITA CRC - Status = 0x0000000F
VITA CRC - Status = 0x0000000F
VITA CRC - Status = 0x0000000F
VITA SPI Sequence 6 - Enable Sequencer
            VITA_SPI[0x00C0] => 0x0000
                                0x0001
            VITA_SPI[0x00C0] <= 0x0001
            VITA_SPI[0x00C1] => 0x0000
                                0xFF00
            VITA_SPI[0x00C1] <= 0x0000
FMC-IMAGEON VITA Configuration for 1080P60 timing ...
VITA 1080P60 - Disable Sequencer
            VITA_SPI[0x00C0] <= 0x0000
VITA 1080P60 - Adjust line spacing in VITA
            VITA_SPI[0x00C1] <= 0x0400
            VITA_SPI[0x00C0] <= 0x0040
VITA  1080P60  -  Tolerate  6  lines  of  jitter  (required  for
programmable exposure)
            VITA_REG[0x005C] <= 0x00000CE4
VITA 1080P60 - Adjust line spacing in sync generator
            VITA_REG[0x0060] <= 0x00580780
            VITA_REG[0x0064] <= 0x0094802C
```

```
VITA 1080P60 - Adjust frame spacing in VITA
        VITA_SPI[0x00C7] <= 0x0001
        VITA_SPI[0x00C8] <= 0x0000
        VITA_SPI[0x00C2] <= 0x0000
VITA 1080P60 - Adjust frame spacing in sync generator
        VITA_REG[0x0068] <= 0x00040438
        VITA_REG[0x006C] <= 0x00248005
VITA 1080P60 - Crop ROI0 from 1920x1200 to 1920x1080
        VITA_SPI[0x0101] <= 0x003C
        VITA_SPI[0x0102] <= 0x0474
VITA 1080P60 - Disable auto-exposure
        VITA_SPI[0x00A0] <= 0x0010
VITA 1080P60 - Enable trig generator
        VITA_REG[0x00E4] <= 0x000970FC
        VITA_REG[0x00E8] <= 0x0000F1B2
        VITA_REG[0x00EC] <= 0x00000001
        VITA_REG[0x00E0] <= 0x31000011
        VITA_REG[0x00E0] <= 0x30000011
VITA 1080P60 - Exposure related settings
        VITA_SPI[0x00C2] <= 0x0400
        VITA_SPI[0x0029] <= 0x0700
VITA 1080P60 - Enable sequencer
        VITA_SPI[0x00C0] => 0x0040
        VITA_SPI[0x00C0] <= 0x0071
VITA Status =
        Image Width  = 1920
        Image Height = 1080
        Frame Rate   = 60 frames/sec
        CRC Status   = 0
VITA DECODER - Reading Statistics
        Black Lines  = 1
        Image Lines  = 1080
        Black Pixels = 1920
        Image Pixels = 1920
        Frames       = 174
        Windows      = 1
        Start Lines  = 193707
        End Lines    = 193708
        (End-Start)  = 1
        Clocks       = 618749
VITA CRC - Status = 0x00000000
CFA Initialization
VDMA 0 Initialization
VDMA 1 Initialization
OSD Initialization (hdmi=0x00, cam=0xFF)

FMC_IMAGEON_GS>
```

Revision History

Date	Version	Revision
09 Sep 2015	2014.4.01	Release to microzed.org site
21 Nov 2015	2015.2.01	Update to Vivado 2015.2
24 Feb 2016	2015.4.01	Update to Vivado 2015.4