FMC-IMAGEON + VITA-2000-C
Getting Started Design Tutorial

1 May 2015
Version 2014_4

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

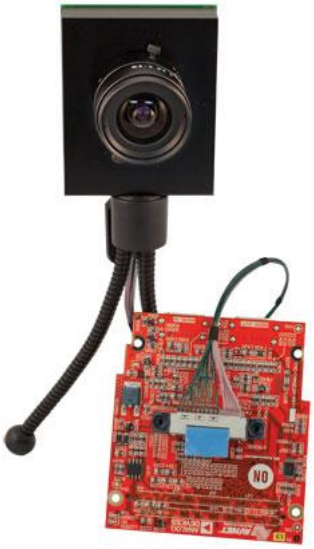


Figure 1 – ON Semiconductor Image Sensor with HDMI Input/Output FMC Bundle

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

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

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.

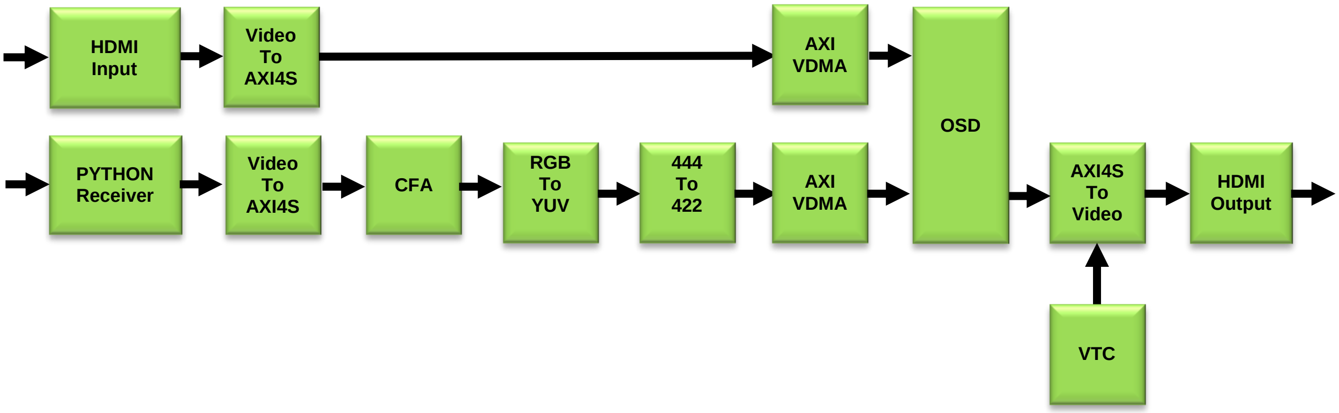


Figure 2 – FMC-IMAGEON Getting Started Reference Design – Hardware Block Diagram

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 2014.4
- MicroZed Board Definition Install for Vivado 2014.4
  - 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) monitor
- 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

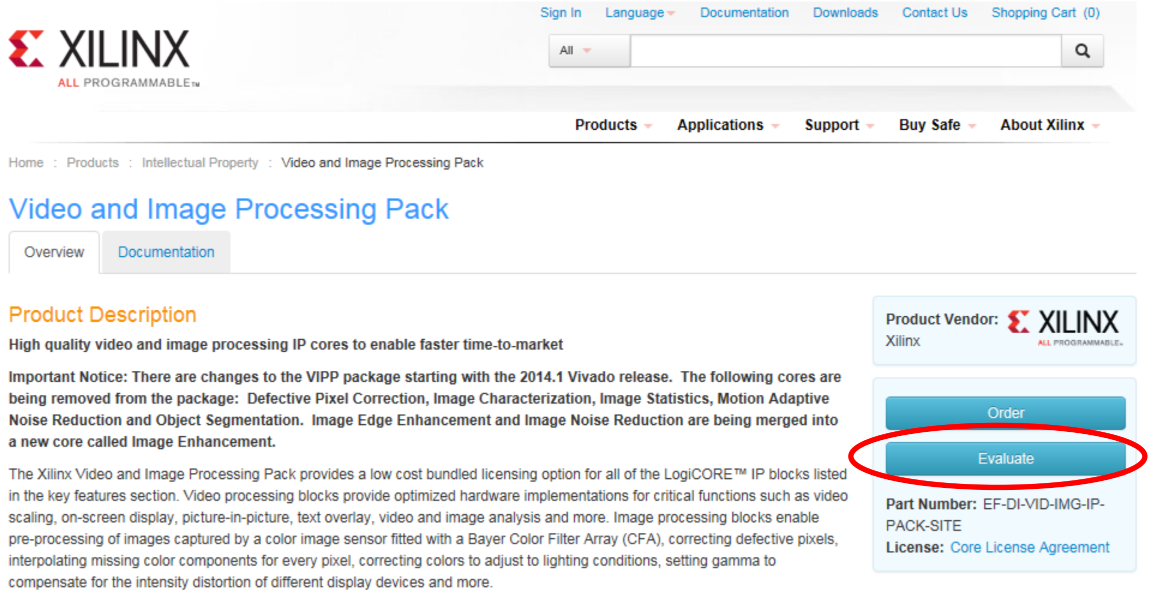
This reference design uses several of the Xilinx Video and Image Processing Pack IP cores. In order to build the hardware design, valid licenses (hardware evaluation, or full license) are required for the following video IP cores:

- 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



Video and Image Processing Pack

Product Description

High quality video and image processing IP cores to enable faster time-to-market

Important Notice: There are changes to the VIPP package starting with the 2014.1 Vivado release. The following cores are being removed from the package: Defective Pixel Correction, Image Characterization, Image Statistics, Motion Adaptive Noise Reduction and Object Segmentation. Image Edge Enhancement and Image Noise Reduction are being merged into a new core called Image Enhancement.

The Xilinx Video and Image Processing Pack provides a low cost bundled licensing option for all of the LogiCORE™ IP blocks listed in the key features section. Video processing blocks provide optimized hardware implementations for critical functions such as video scaling, on-screen display, picture-in-picture, text overlay, video and image analysis and more. Image processing blocks enable pre-processing of images captured by a color image sensor fitted with a Bayer Color Filter Array (CFA), correcting defective pixels, interpolating missing color components for every pixel, correcting colors to adjust to lighting conditions, setting gamma to compensate for the intensity distortion of different display devices and more.

Part Number: EF-DI-VID-IMG-IP-PACK-SITE
License: Core License Agreement

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

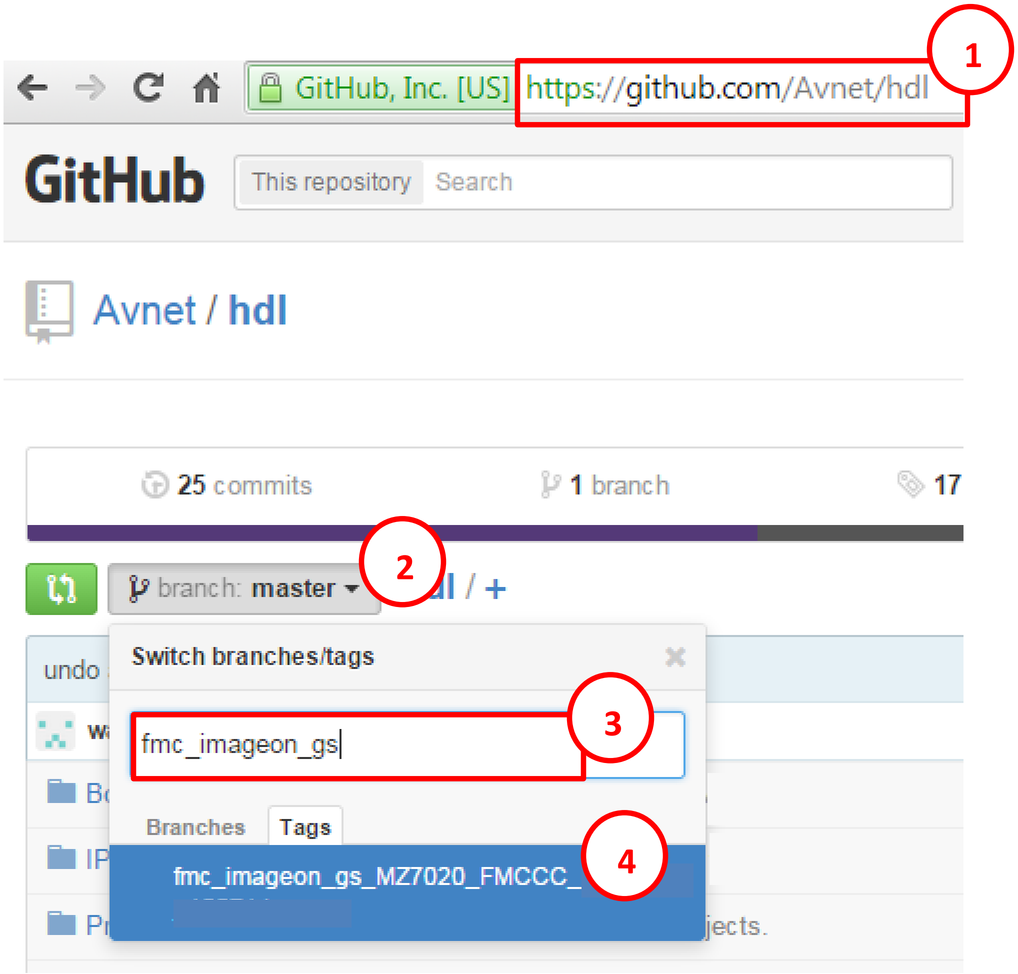


Figure 4 – Avnet GitHub repository – Retrieving specific version with tag

5. Select the **fmc_imageon_gs_MZ7020_FMCCC_20150501_155714** 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

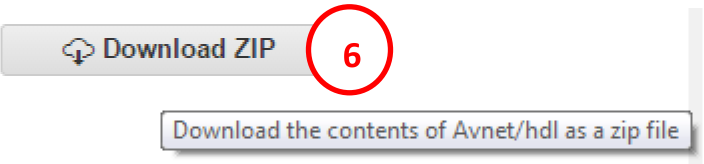


Figure 5 – Avnet GitHub repository – Download ZIP

7. Create an "Avnet" directory in your root C:\ drive
8. Save **hdl-fmc_imageon_gs_MZ7020_FMCCC_20150501_155714.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_20150501_155714" directory to "hdl"
   You should see the following directory structure

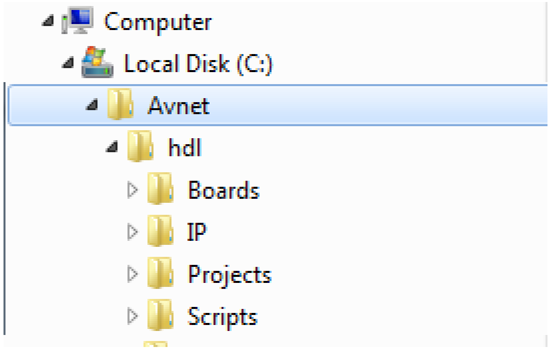


Figure 6 – Extracted C:\Avnet\hdl directory structure

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:

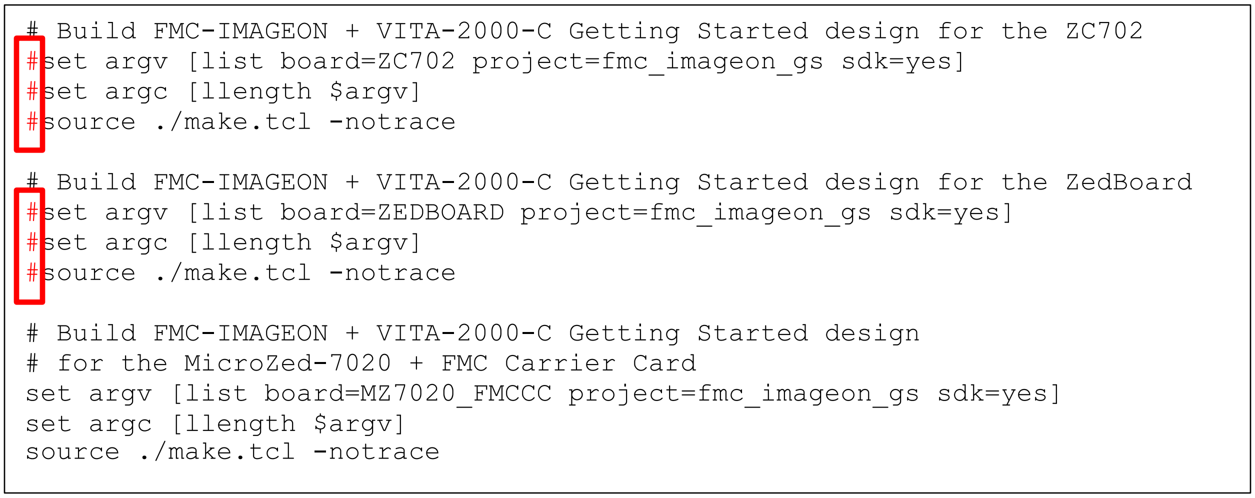
Directory	Content Description
C:\Avnet\hdl\Boards	contains board related files
C:\Avnet\hdl\IP	contains the IP cores used by the ref designs
C:\Avnet\hdl\Projects	contains project related files
C:\Avnet\hdl\Scripts	contains scripts used to automatically build the designs

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

Directory	**Content Description**
C:\Avnet\hdl\IP\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
```

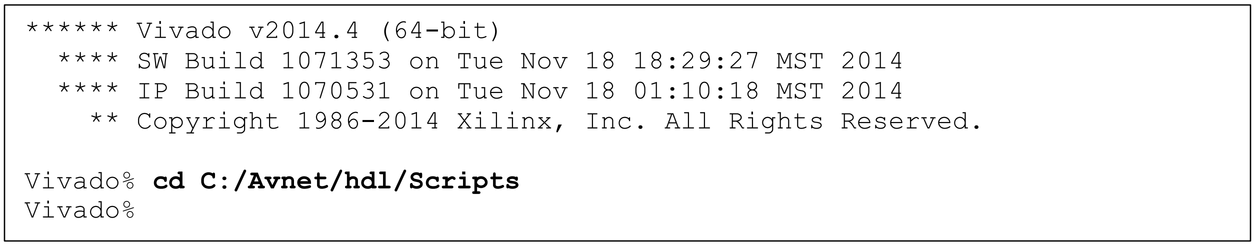
Figure 7 – Editing the make script to build for MicroZed FMC Carrier Card

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 2014.4 TCL Shell” console

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

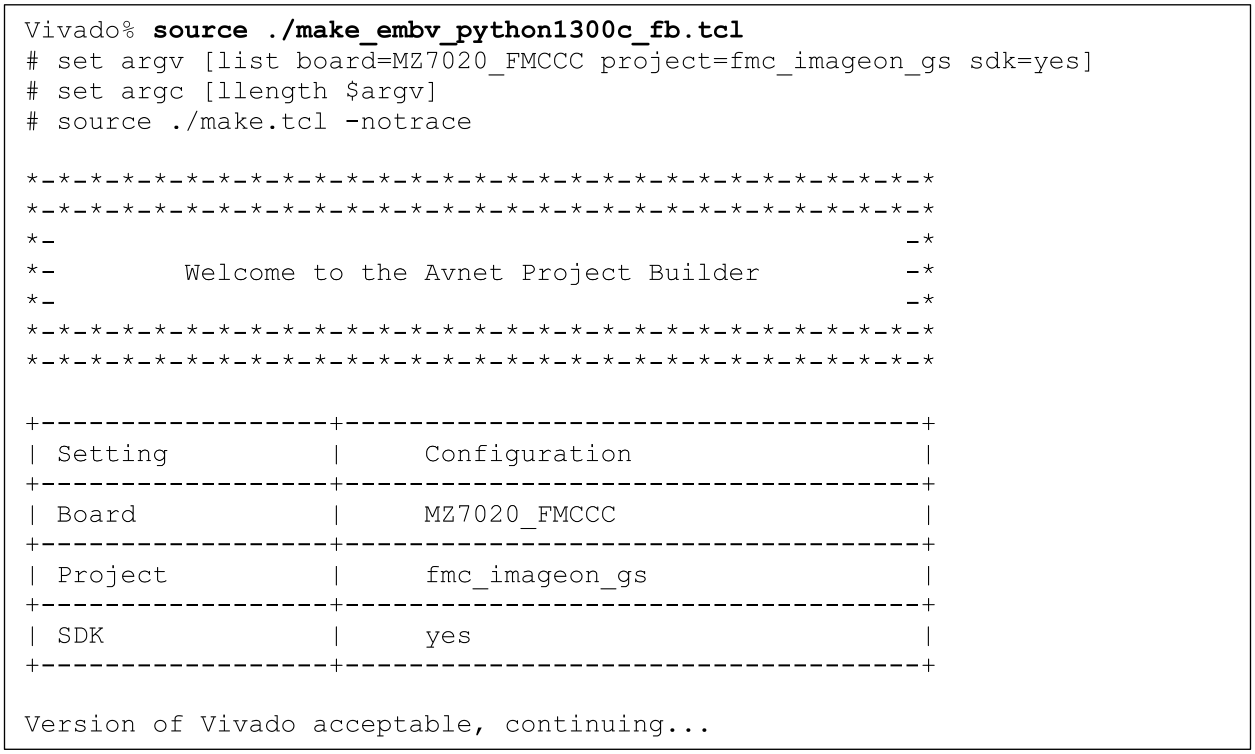


```
****** Vivado v2014.4 (64-bit)
  **** SW Build 1071353 on Tue Nov 18 18:29:27 MST 2014
  **** IP Build 1070531 on Tue Nov 18 01:10:18 MST 2014
    ** Copyright 1986-2014 Xilinx, Inc. All Rights Reserved.

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

Figure 8 – Vivado 2014.4 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]
# 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 of Vivado acceptable, continuing...
```

Figure 9 – Vivado 2014.4 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 2014.4 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 2014.4,
  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.

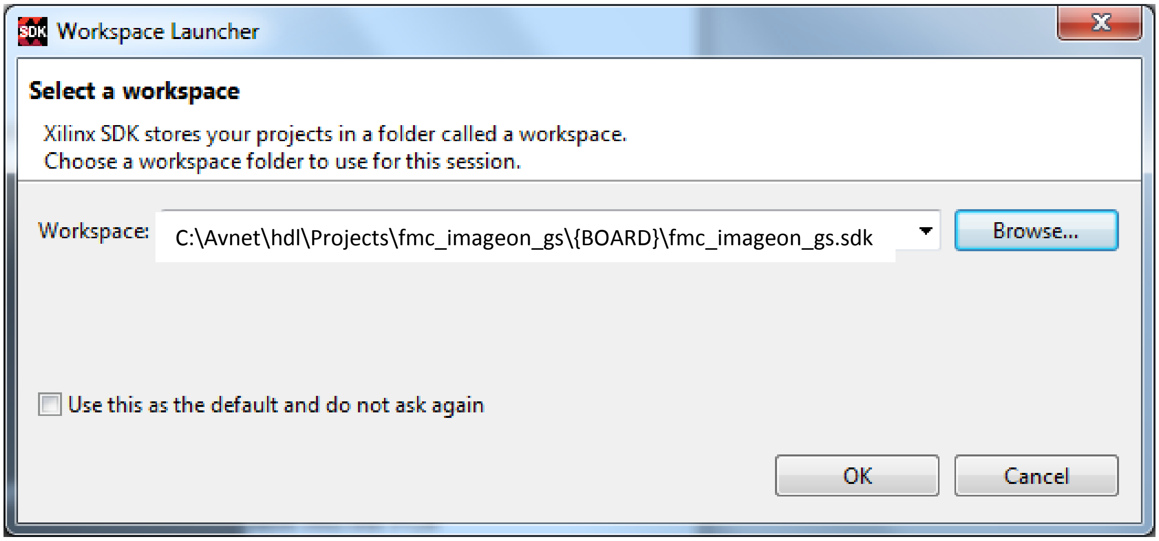
Booting from JTAG with SDK

The hardware and software can also be loaded to hardware using SDK 2014.4 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 2014.4, and specify the following directory for the SDK workspace:

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



Select a workspace

Xilinx SDK stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

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

□ Use this as the default and do not ask again

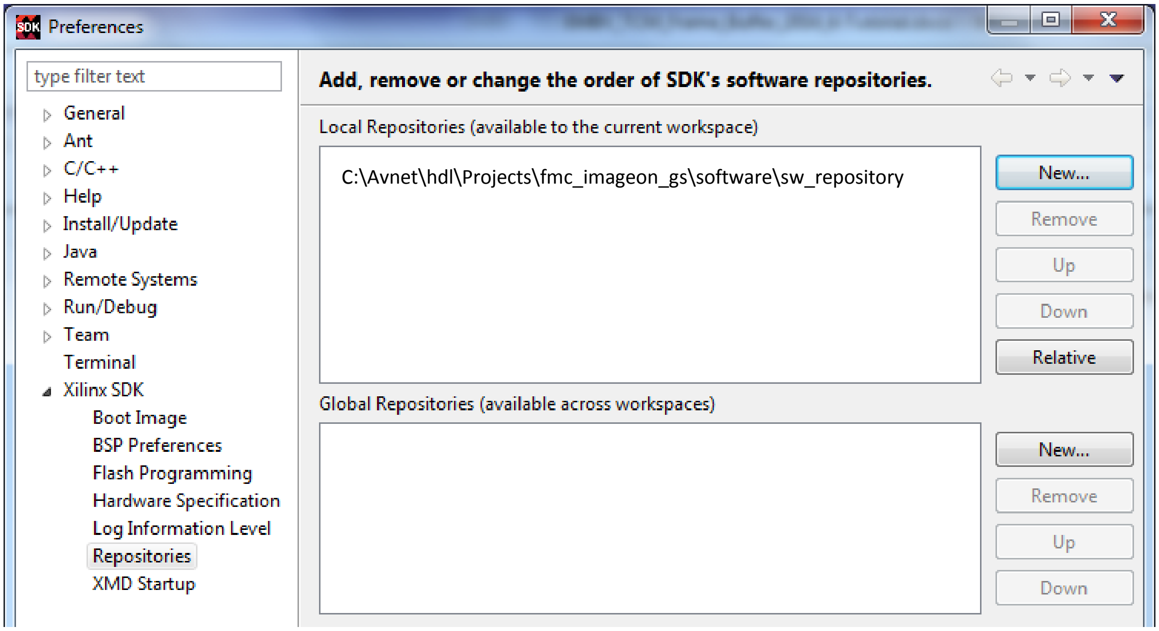
Figure 11 – SDK – Specifying SDK workspace

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



Add, remove or change the order of SDK's software repositories.

Local Repositories (available to the current workspace)

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

Global Repositories (available across workspaces)

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**.

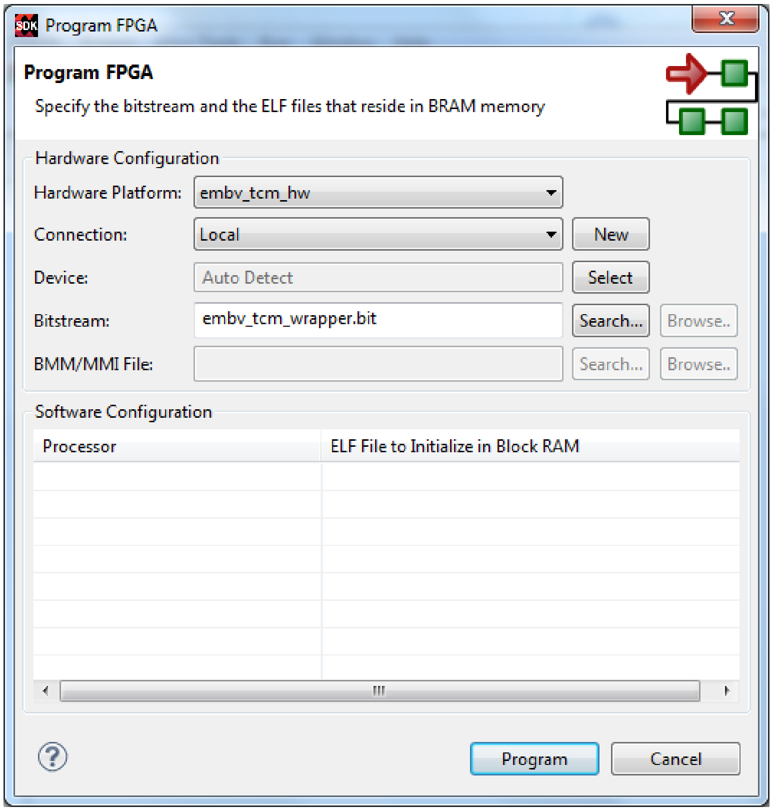
   C:\Avnet\hdl\Projects\fmc_imageon_gs\software\sw_repository

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**



Program FPGA

Program FPGA

Specify the bitstream and the ELF files that reside in BRAM memory

Hardware Configuration

Hardware Platform: embv_tcm_hw

Connection: Local

Device: Auto Detect

Bitstream: embv_tcm_wrapper.bit

BMM/MMI File:

Software Configuration

Processor	ELF File to Initialize in Block RAM

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 something similar to the following on your serial console:

```
------------------------------------------------------
--            Embedded Vision Carrier Card          --
--                PYTHON-1300-C Design              --
------------------------------------------------------

HDMI Initialization
PYTHON Initialization
CFA Initialization
TPG Initialization
VDMA 0 Initialization
VDMA 1 Initialization
OSD Initialization
System Ready!

      Press 0-9 to change alpha blending of camera/tpg layers

      Press ENTER to restart
```

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

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. To re-initialize the capture pipeline for the VITA camera, type the “start vita” command, then {ENTER}.

If you do not have a VITA-2000 camera module, the reference design will attempt to initialize the HDMI input capture pipeline. If you have a valid non-encrypted DVI/HDMI source, you will see the content on the top left portion of the 1080P resolution output. To re-initialize the capture pipeline for the HDMI input, type the “start hdmi” command, then {ENTER}.

Revision History

Date	Version	Revision
01 May 2015	2014_4	Release to microzed.org site