FMC-IMAGEON
EDK Reference Design Tutorial

Version 0.5

Revision History

Version	Description	Date
0.2	Implemented with 13.4 version of tools Support for ML605	Feb. 22, 2012
0.3	Implemented with 14.2 version of tools Supported FMC carriers: - Zynq-7000 EPP : ZC702, ZedBoard - Kintex-7 : KC705 - Virtex-6 : ML605 - Spartan-6 : Spartan-6 LX150T Dev Board Add section describing how to rebuild hardware/software Add section describing how to add Video IP to design	Aug. 09, 2012
0.4	Minor changes	Aug. 29, 2012
0.5	Mention use of SSC on HDMI output interface (HDMIO_CLK)	Sep. 07, 2012

Table of Contents

Table of Figures

Table of Tables

About this Guide

This user guide introduces an EDK reference design which demonstrates the features of the FMC-IMAGEON FMC module.

This manual contains the following chapters:

- Chapter "**Introduction**" provides an overview and features of the FMC-IMAGEON hardware.
- Chapter "**EDK Reference Design**" provides a detailed description of the FMC-IMAGEON EDK reference design, as well as instructions on how to run the demonstration.
- Chapter "**Rebuilding the EDK Reference Design**" describes how to rebuild the hardware design and software application, as well as how to load the design from the SDK environment.
- Chapter "**Modifying the EDK Reference Design**" provides a tutorial on how to add an additional Video IP core to the image processing pipeline.
- The "**EDK Repository**" chapters provide a detailed description of the PCOREs, drivers, and software libraries used in the FMC-IMAGEON EDK reference design.
- Appendix "**References**" provides a list of references to documentation related to the FMC module.
- Appendix "**Known Issues and Limitations**" provides a list of known issues and limitations with this reference design.
- Appendix "**Troubleshooting**" provides a list of troubleshooting suggestions for this reference design.

Additional Documentation

For more information on the VITA-2000 image sensor, please visit the following resources.

- **VITA2000: 2.3 Megapixel, 92 FPS, Global Shutter CMOS Image Sensor**
  - VITA-2000 Product Information
    http://www.onsemi.com/PowerSolutions/product.do?id=VITA2000

For more information on the Analog Devices HDMI devices, please visit the following resources.

- **ADV7511 HDMI Transmitter**
  - ADV7511 Product Information
    http://www.analog.com/adv7511
  - ADV7511 Technical Resources on EngineerZone
    http://ez.analog.com/docs/DOC-1740

- **ADV7611 HDMI Receiver**
  - ADV7611 Product Information
    http://www.analog.com/adv7611
  - ADV7611 Technical Resources on EngineerZone
    http://ez.analog.com/docs/DOC-1745

Additional Support Resources

To access the most current collateral for the FMC-IMAGEON FMC module, please visit the product website at:

http://www.em.avnet.com/fmc-imageon

To access the most current collateral for the On Semi Image Sensor FMC bundle, which includes this FMC module, please visit the product website at:

http://www.em.avnet.com/fmc-imageon-v2000c

Once on the product website:

To access the latest documentation and designs, click on the following link:

To access technical support, click on the following link:

To access the technical forums, visite the following web link:

http://community.em.avnet.com/

To search the database of silicon and software questions and answers or to create a technical support case in WebCase, see the Xilinx website at:

http://www.xilinx.com/support

Introduction

The ON Semiconductor Image Sensor FMC bundle provides several high-definition video interfaces for Xilinx® FMC-enabled baseboards. The FMC module has on-board HDMI input/output interfaces. The ON Semiconductor VITA-2000-color image sensor module provides a high definition camera supporting high frame rates and featuring a global shutter.

This FMC bundle is ideal for developing application for machine vision, motion monitoring, and high-end security and surveillance.

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

As illustrated in Figure 2 and Figure 3, the FMC module connects to an FMC carrier, and provides the following interfaces:

- HDMI Input
- HDMI Output
- LCEDI Interface for VITA Image Sensor modules

The following block diagram illustrates the connectivity of the FMC module.

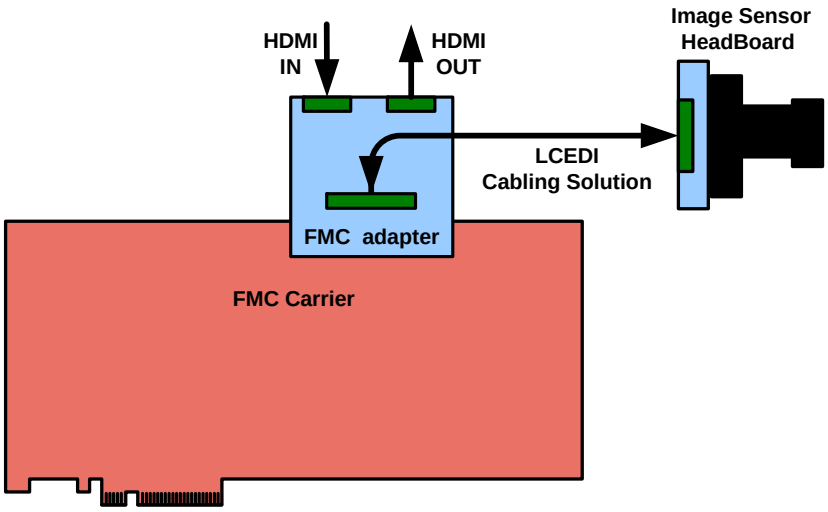


Figure 2 – Connectivity Diagram

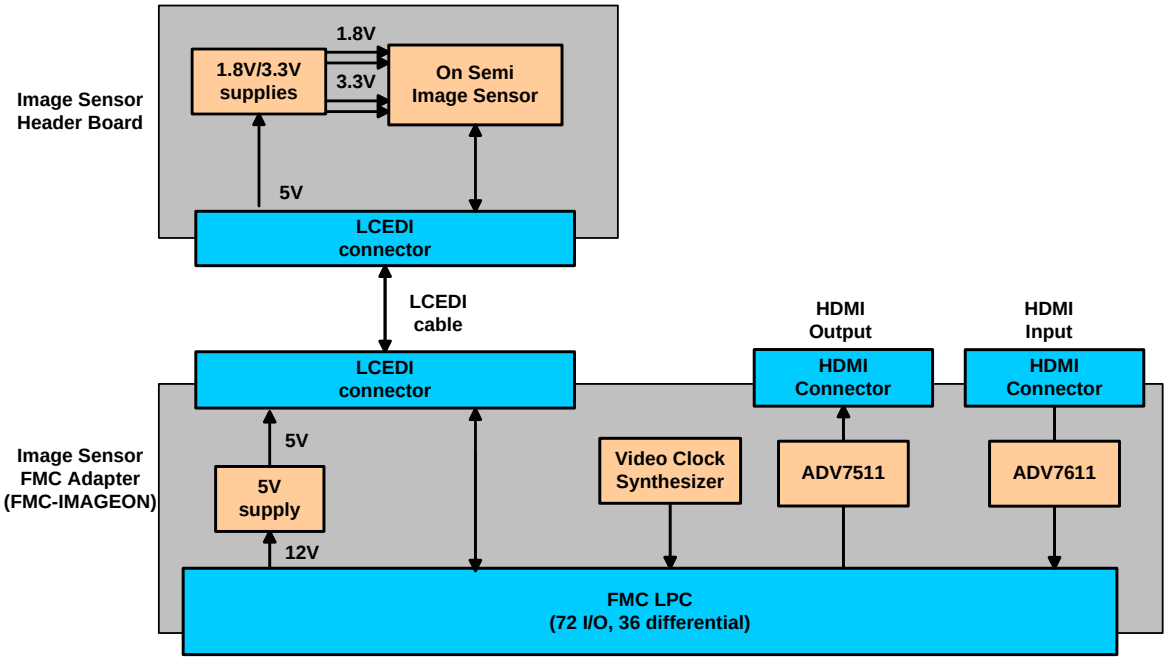


Figure 3 – Block Diagram

Requirements

The software and hardware requirements for this reference design are described in the following sections.

Software

The software required to use and run the demonstrations is:

- Xilinx ISE Design Suite 14.2 : Embedded Edition
- Terminal Emulator (HyperTerminal or TeraTerm)

Hardware

The bare minimum required to run this EDK reference design is:

- Computer with a minimum of 4 GB to complete a design[1]
- HDMI (or DVI-D) monitor

In order to demonstrate the HDMI input feature, the following additional hardware is required:

- HDMI source (non-encrypted content) or DVI-D source

In order to demonstrate the VITA-2000 image sensor, the following additional hardware is required:

- VITA-2000 image sensor (including optics, tripod, and LCEDI cable)

NOTE :
Spread Spectrum Clocking (SSC) is used on the FMC-IMAGEON module's HDMI output interface (HDMIO_CLK) to reduce radiated emissions to industry approved levels. This can be implemented using the SSC feature of the on-board TI CDCE913 video clock synthesizer.
Refer to the "Known Issues and Limitations" section for more information.

[1] Refer to www.xilinx.com/ise/products/memory.htm

EDK Reference Design

Overview

The FMC-IMAGEON EDK 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
- Processing the image sensor content with a simple image processing pipeline

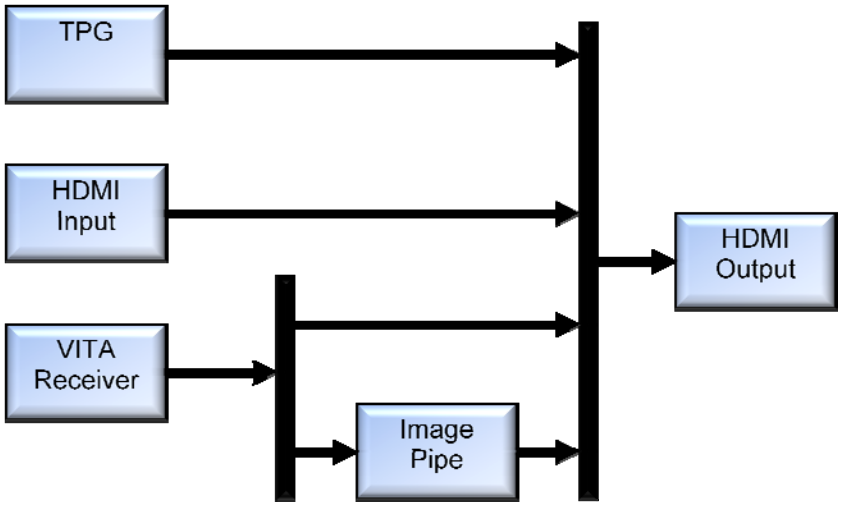


Figure 4 – FMC-IMAGEON EDK Reference Design – Block Diagram

The HDMI Output is configured via software for 1080P60 video resolution.

Any one of the four video content sources can be selected to be driven to the HDMI output:

- Xilinx Test Pattern Generator (TPG)
- HDMI Input
- VITA-2000 Image Sensor – unprocessed raw pixels
- VITA-2000 Image Sensor – processed with the simple image processing pipeline

There is no video frame buffer implemented in this design.

The simple image processing pipeline implemented in the getting started design is expanded in the following block diagram.

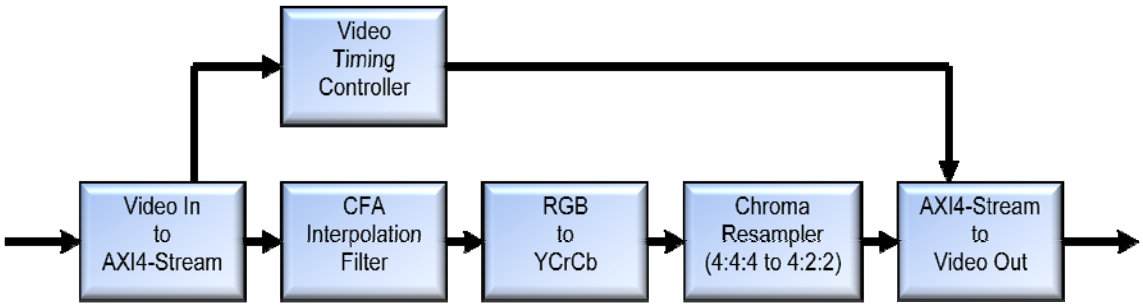


Figure 5 – Simple Image Processing Pipeline – Block Diagram

The "Video In to AXI4-Stream" and "AXI4-Stream to Video Out" cores provide the bridging to/from the AXI4-Stream interface.

The AXI4-Stream interface will preserve frame and line boundaries, but will not preserve video timing. The Video Timing Controller is used to detect the video timing at the input, and regenerate the video timing at the output.

All of the image processing cores (CFA Interpolation, RGB to YCrCb, Chroma Resampler) use AXI4-Stream to receive and transmit video content.

The "CFA Interpolation Filter" core converts raw bayer pixels to RGB color pixels.

The "RGB to YCrCb" core converts from the RGB color space to the YCrCb color space.

The "Chroma Resampler (4:4:4 to 4:2:2)" core sub-samples the chroma components (Cr, Cb) and combines them into a single time division duplex component (CrCb). This reduces the total pixel bit width from 24 bits to 16 bits.

Many more Video IP cores are available from Xilinx to extend the image processing pipeline. For more information on these cores, refer to the following web site:

http://www.xilinx.com/esp/video/refdes_listing.htm#ref_des

A reference design showcasing a more extensive image processing pipeline can be found on the Xilinx Zynq-7000 Video and Image Kit:

http://www.xilinx.com/products/boards-and-kits/DK-Z7-VIDEO-G.htm

The following video resolutions are supported by this demonstration:

Resolution	Pixel Rate (MHz)	Frame Dimensions
1080P60	148.5 MHz	1920 x 1080

Table 1 – Supported Video Resolutions

Download the FMC-IMAGEON EDK Reference Design Tutorial

Go to the Avnet HDMI Input/Output FMC Module product web page:
http://www.em.avnet.com/fmc-imageon
At the bottom of this web page, click on the link “Support Files and Downloads.”
To download the Getting Started Design files,
click on the link “FMC-IMAGEON – EDK Reference Design Tutorial”.
This will download the file, “FMC_IMAGEON_EDK_Reference_Design_Tutorial_{date}.zip” to your computer.
To install the getting started design files, unzip this file using the “Extract to Here” option and then extract the files to the root C:\ drive of your computer. The top-level folder name should be “FMC_IMAGEON”.

C:\FMC_IMAGEON\EDK\...

There are hard paths in the design files that require this exact naming convention.

Directory Structure

The FMC-IMAGEON EDK Reference Design Tutorial has the following directory structure:

- C:\FMC_IMAGEON\EDK\repository => EDK IP repository
- C:\FMC_IMAGEON\EDK\{carrier}_fmc_imageon_getting_started_hw => EDK project
- C:\FMC_IMAGEON\EDK\{carrier}_fmc_imageon_getting_started_sw => SDK project
- C:\FMC_IMAGEON\EDK\{carrier}_fmc_imageon_getting_started_bin => pre-built binaries

Where {carrier} refers to each of the following FMC carriers supported by this demonstration:

- zc702
- zed
- kc705
- ml605
- s6lx150t

Demo Setup Instructions

Each supported FMC carrier will have its own hardware setup instructions in the following text file:

- {carrier}_fmc_imageon_getting_started_bin\{carrier}_fmc_imageon_readme.txt

This text file will describe:

- how to setup your FMC carrier, including
    - which FMC slot to populate the FMC-IMAGEON module
- how to load the demonstration design on your FMC carrier

Using the Application

After loading the design, you should see the following appear on your serial console:

```
------------------------------------------------
--           Text-based Console for           --
-- FMC-IMAGEON Getting Started Demonstration  --
------------------------------------------------
General Commands:
        help       Print the Top-Level menu Help Screen
        quit       Exit console (if applicable)
        verbose    Toggle verbosity on/off
        delay      Wait for specified delay
        mem        Memory accesses
I2C Commands
        iic0       IIC accesses on FMC-IPMI I2C chain
        iic1       IIC accesses on FMC-IMAGEON I2C chain
VITA Commands
        vita       VITA commands (init, status, ...)
        vspi       SPI accesses to VITA sensor
        vreg       Memory accesses to VITA receiver
        again      Analog gain (0-10)
        dgain      Digital gain (0-4095) where 128 corresponds to 1.00
        exposure   Exposure time (1-99) in percentage of frame period
Image Processing Pipeline Commands
        cfa        Color Filter Array Interpolation configuration
Video Source Selection
        video      Select Video source (tpg, hdmi, vita)
        tpg        Test Pattern Generator configuation (menu)

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

AVNET>
```

At this point, the design has initialized the HDMI output interface for 1080P60 video resolution. You should see a “zone plate” test pattern on your HDMI monitor.

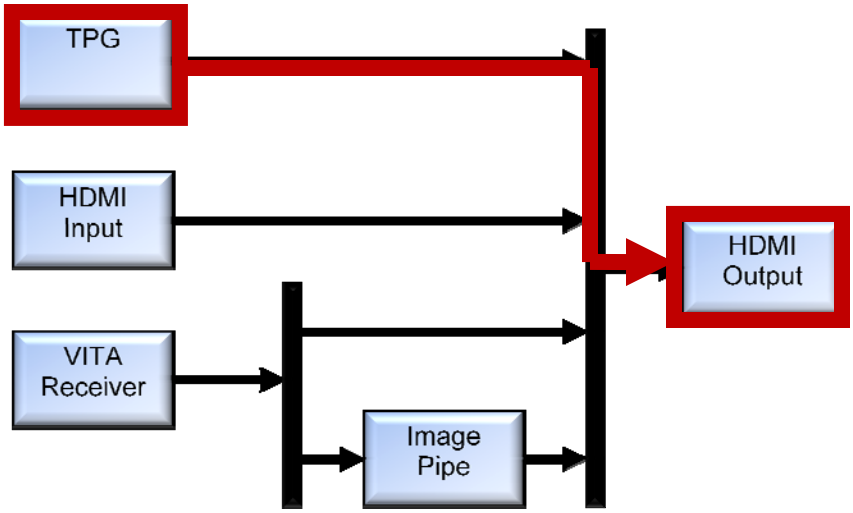


Figure 6 – FMC-IMAGEON EDK Reference Design – "tpg" mode

The console supports several commands. The list of commands can be displayed with the "help" command. More information on each command can be displayed by typing the command, and specifying "help" as a second argument. For example, to display help on the "video" command, type "video help".

```
AVNET>video help
        Syntax :
                video tpg     => Select Test Pattern Generator
                video hdmi    => Select HDMI Input
                video vita    => Select VITA Image Sensor

AVNET>
```

If you have an HDMI source connected, you can enable this input with the "video hdmi" command, as shown below:

```
AVNET>video hdmi
Waiting for ADV7611 to locked on incoming video ...
        ADV7611 Video Input LOCKED
ADV7611 Video Input Information
        Video Input      = HDMI, Progressive
        Color Depth      = 8 bits per channel
        HSYNC Timing     = hav=1920, hfp=88, hsw=44(hsp=1), hbp=148
        VSYNC Timing     = vav=1080, vfp=04, vsw=05(vsp=1), vbp=036
        Video Dimensions = 1920 x 1080
Setting Video Multiplexer to HDMI Input (HDMI)
FMC-IMAGEON HDMI Output Initialization ...
```

```
AVNET>
```

At this point, you should see the contents of your HDMI source on the HDMI monitor.

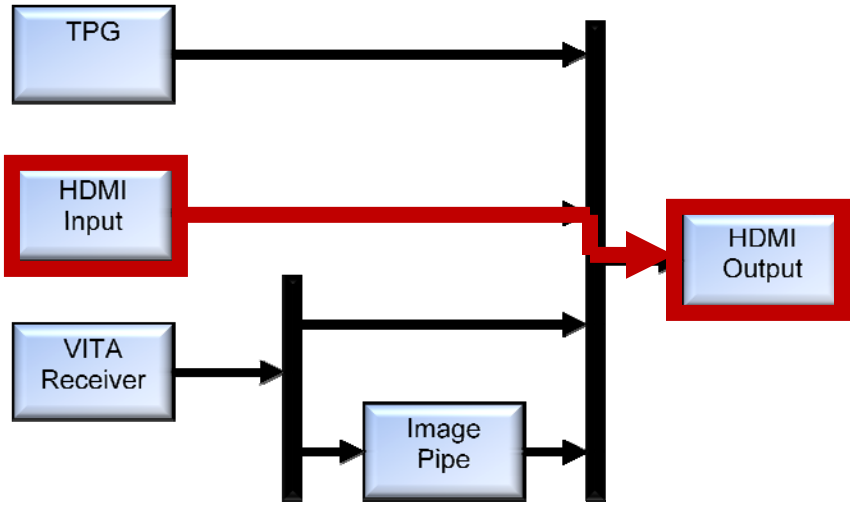


Figure 7 – FMC-IMAGEON EDK Reference Design – “hdmi” mode

If you have the VITA-2000 image sensor connected, you can enable this input with the “video vita” command, as shown below:

```
AVNET>video vita
FMC-IMAGEON VITA Initialization ...
FMC-IMAGEON VITA Configuration for 1080P60 timing ...
VITA Status =
        Image Width  = 1920
        Image Height = 1080
        Frame Rate   = 60 frames/sec
Setting Video Multiplexer to Image Sensor Raw Pixels (VITA)
FMC-IMAGEON HDMI Output Initialization ...

AVNET>
```

At this point, you should see the raw contents of the VITA-2000 image sensor on the HDMI monitor. The image will be gray, and a grid corresponding to the bayer pattern should be visible.

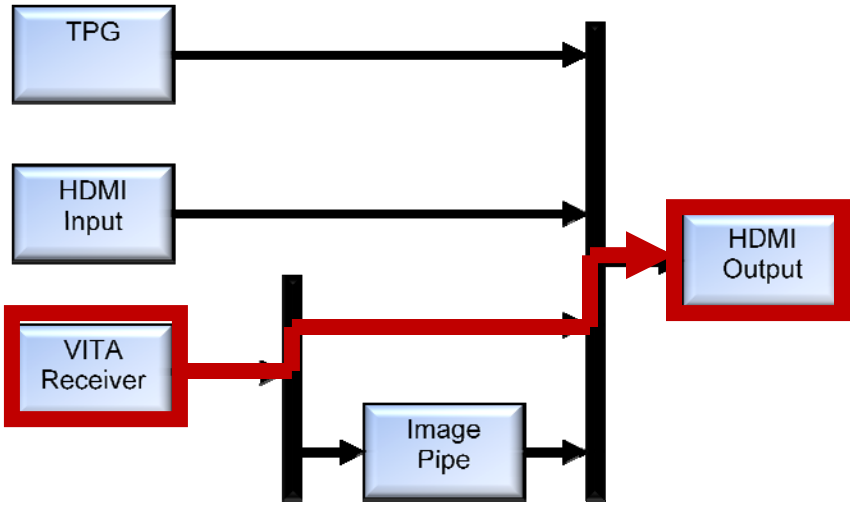


Figure 8 – FMC-IMAGEON EDK Reference Design – “vita” mode

To select the output of the simple image processing pipeline, type the “video ipipe” command, as shown below:

```
AVNET>video ipipe
FMC-IMAGEON VITA Initialization ...
FMC-IMAGEON VITA Configuration for 1080P60 timing ...
VITA Status =
        Image Width  = 1920
        Image Height = 1080
        Frame Rate   = 60 frames/sec
Setting Video Multiplexer to Image Sensor Raw Pixels (VITA)
Video Detector Configuration ...
```

```
        Video Resolution = 1080P
Image Processing Pipeline (iPIPE) Initialization ...
        Chroma Resampler done
        RGB2YCrCb done
        CFA done
Setting Video Multiplexer to Image Processing Pipeline (IPIPE)
FMC-IMAGEON HDMI Output Initialization ...

AVNET>
```

You should now see a color image on the HDMI monitor.

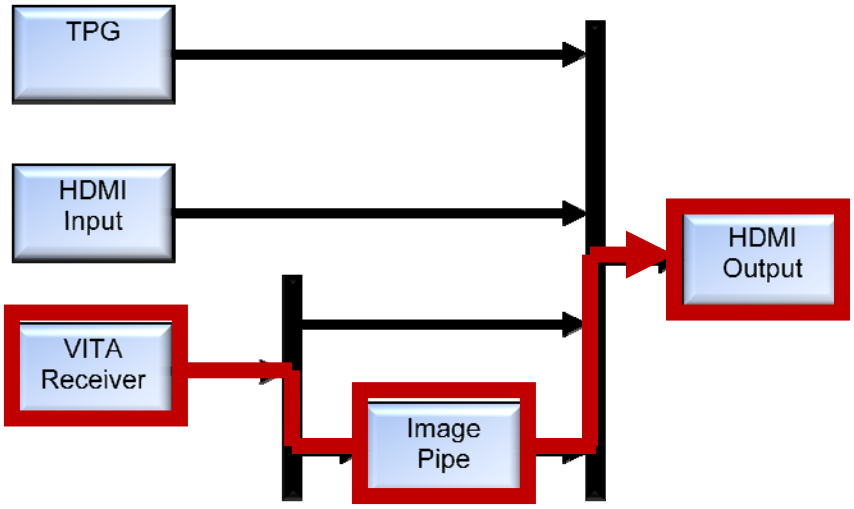


Figure 9 – FMC-IMAGEON EDK Reference Design – “ipipe” mode

The console will allow you to perform various tasks, such as performing memory accesses, performing I2C transactions, and SPI transactions.

You can use the "vspi" command to access the VITA image sensor's register via SPI transactions. The example below shows how to read REG00, which contains the image sensor's identifier.

```
AVNET>vspi help
        Syntax:
                vspi read  {address}
                vspi write {address} {data}
                vspi rmw   {address} {data} {mask}
                vspi poll  {address} {data} {mask}
                vspi dump  {address1} {address2}

AVNET>vspi read 0x0000
        0x0000 => 0x5614

AVNET>
```

You can use the "cfa" command to query the status of configure the Color Filter Array interpolation core in the Xilinx image processing pipeline.

```
AVNET>cfa help
        Syntax :
                cfa status    => Display status of CFA core
                cfa bayer     => Set CFA bayer pattern (0-3)

AVNET>cfa status
Enable Bit: 1
 Register Update Bit: 1
 Reset Bit: 0
 CFA Status: 00000011
 Core Version:   2.1
CFA_CONTROL           :        3
CFA_BAYER_PHASE       :        0

AVNET>cfa bayer 1
Setting bayer to 1

AVNET>cfa bayer 2
Setting bayer to 2

AVNET>cfa bayer 3
Setting bayer to 3

AVNET>cfa bayer 0
Setting bayer to 0
```

```
AVNET>
```

This concludes the demonstration on how to run the demonstration.

Software Overview

The following software components make up the application.

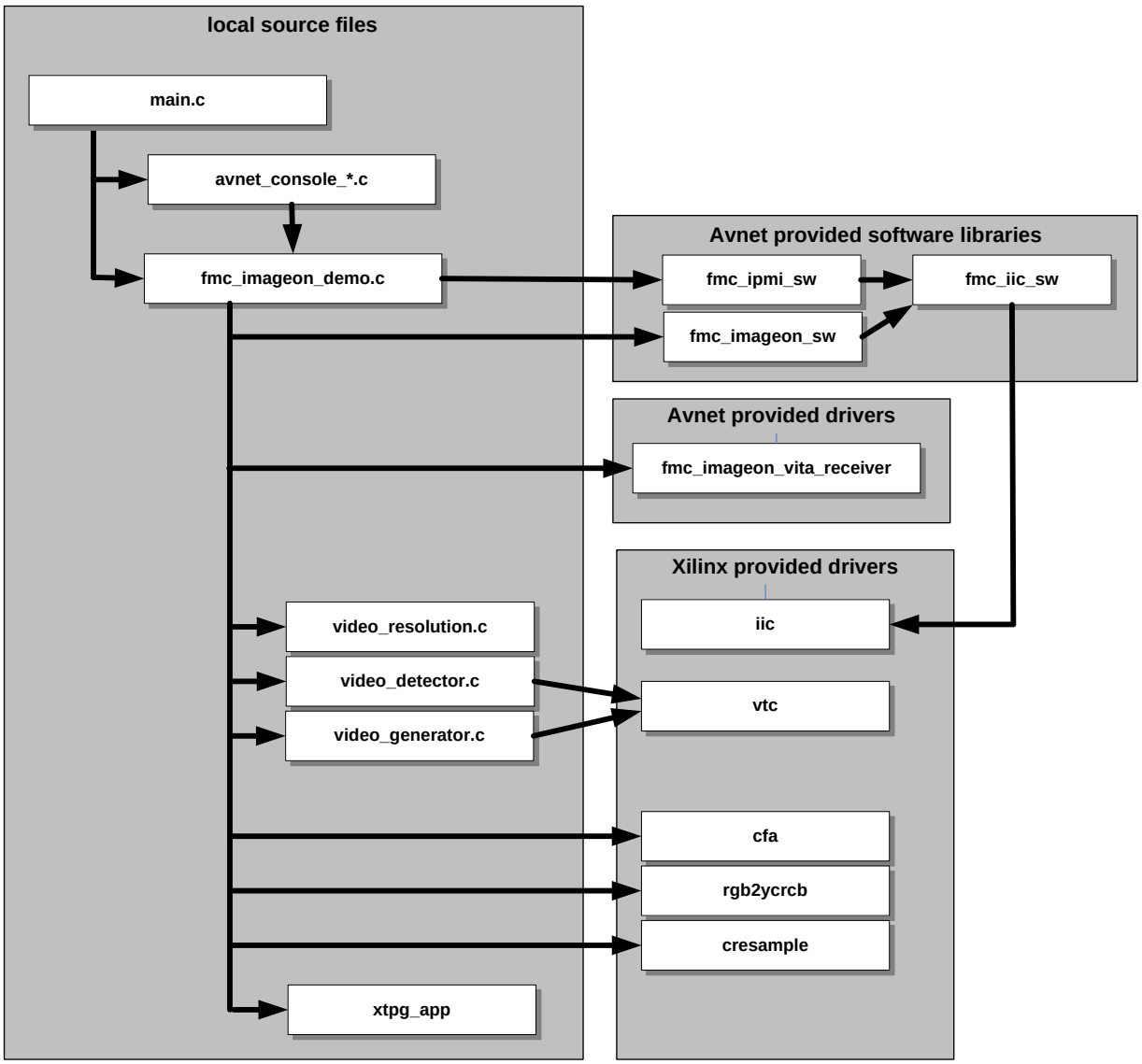


Figure 10 – Getting Started Application – Source Files

The drivers and software libraries provided with this demonstration, described in detail in section **EDK Repository - Software Libraries**, are located in the following central repository:

FMC_IMAGEON\EDK\repository\AvnetProcessorIPLib\drivers
FMC_IMAGEON\EDK\repository\AvnetProcessorIPLib\sw_services

The local source files are located in the following directory:
{carrier}_fmc_imageon_getting_started_sw\{carrier}_fmc_imageon_getting_started_sw\src

The following files in \src are specific to this reference design:

main.c	This file is the main entry point for the FMC-IMAGEON getting started application
avnet_console_*.c	These files implement a text-based console over the serial interface.
fmc_imageon_demo.c fmc_imageon_demo.h	This file contains the code to configure the FMC-IMAGEON getting started demonstration
video_resolution.c video_resolution.h	This file contains definitions for common video resolutions, including the 1080P60 resolution. YThis file can be modified to add new video resolution definitions.
video_detector.c video_detector.h	This file provides code to configure the AXI_VTC PCORE when in use on a video input data path.
video_generator.c video_generator.h	This file provides code to configure the AXI_VTC PCORE when used to generate video timing for a specific video resolution.
xtpg_app.c xtpg_app.h	These files provide configuration code for the Test Pattern Generator PCORE.

Rebuilding the EDK Reference Design

Overview

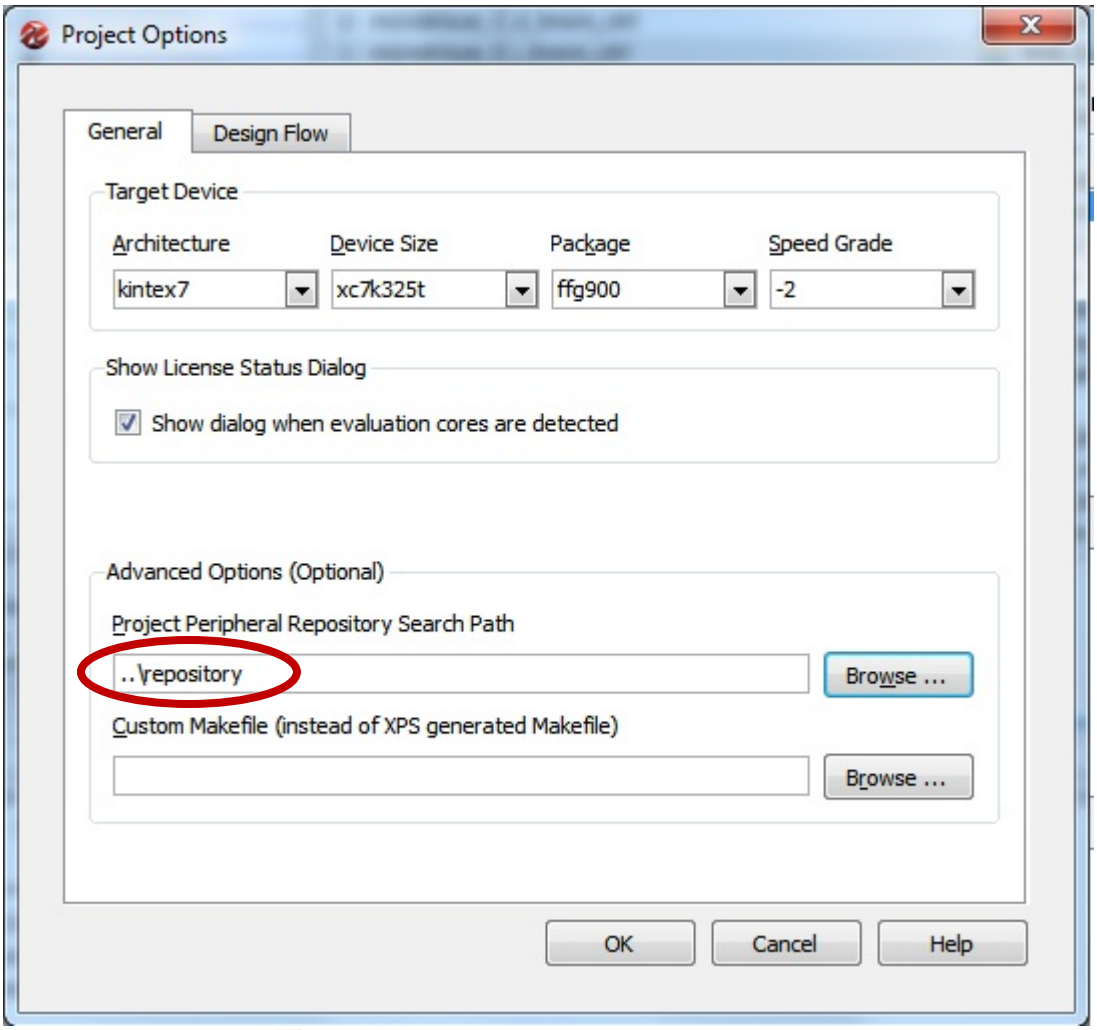
This section describes how to rebuild the EDK reference design.

Rebuilding the Hardware with XPS

Open the XPS project corresponding to your FMC carrier:

C:\FMC_IMAGEON\EDK\{carrier}_fmc_imageon_getting_started_hw\system.xmp

The XPS project makes use of a repository containing several PCOREs specific to the FMC-IMAGEON hardware. To view the reference to this repository, select **Project => Project Options** from the menu.



Since this reference is relative (..\repository), it should be valid regardless of where you extracted the archve for this reference design.

Click **OK** when done.

In the menu, select

Hardware => **Generate Bitstream.**

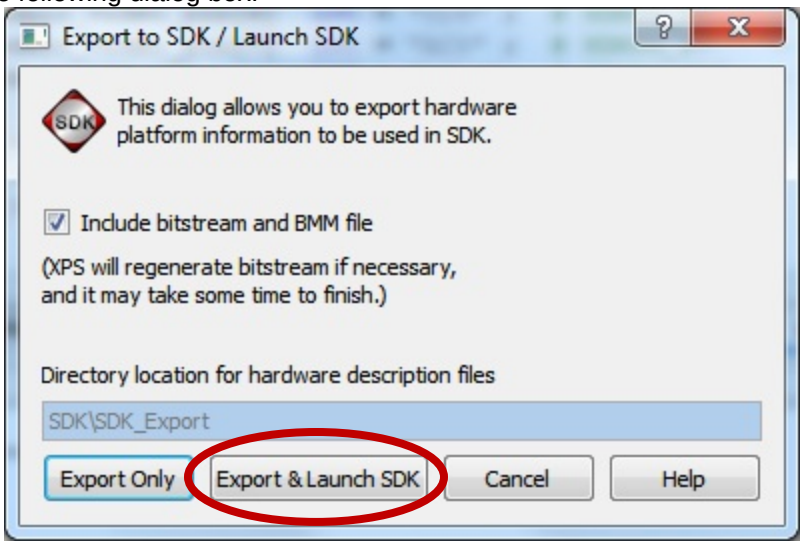
This will rebuild the hardware design.

The next step is to export the new hardware design files to SDK.

In the menu, select

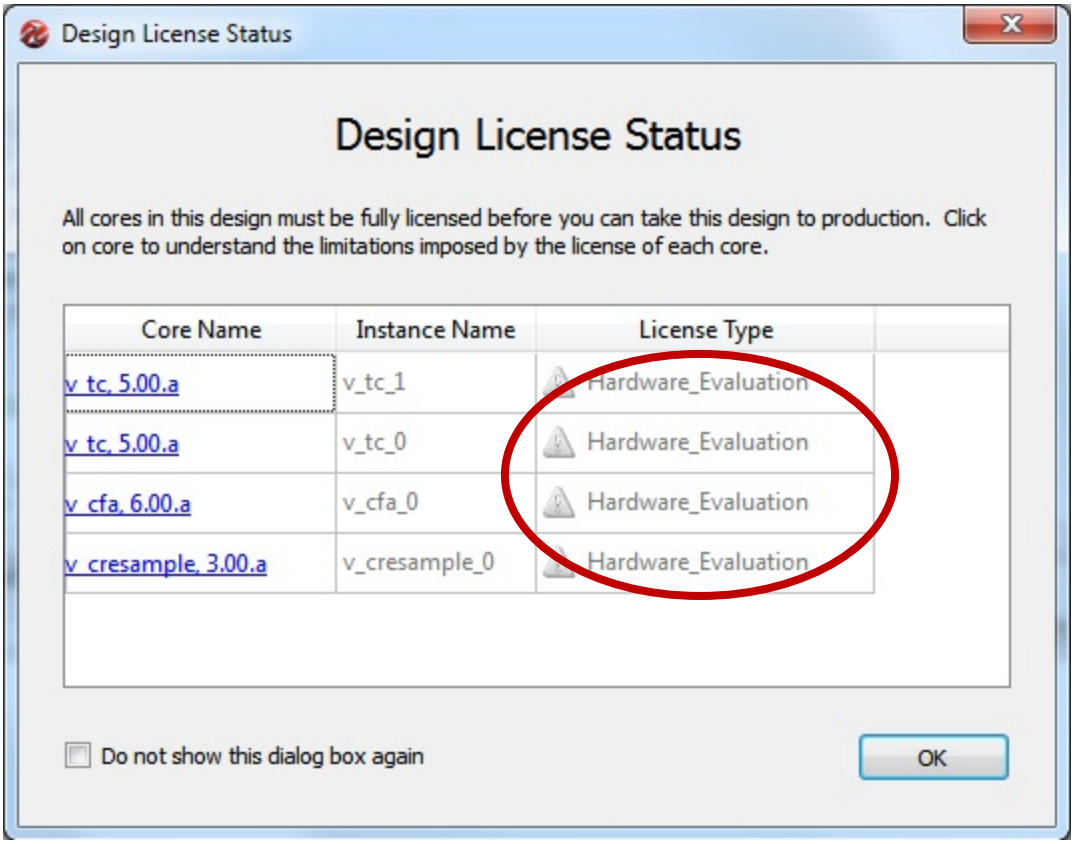
Project => **Export Hardware Design to SDK …**

You will see the following dialog box.



Click on the **Export and Launch SDK** button.

You may see some **Design License Status** dialog boxes, as shown below.



Ensure that you have the “Hardware Evaluation” licenses for each of the cores, then click **OK**.

If you don’t have “Hardware Evalution” licenses of the cores, you will not be able to target the design to hardware.

This will open SDK, which is described in the next section.

Rebuilding the Software with SDK

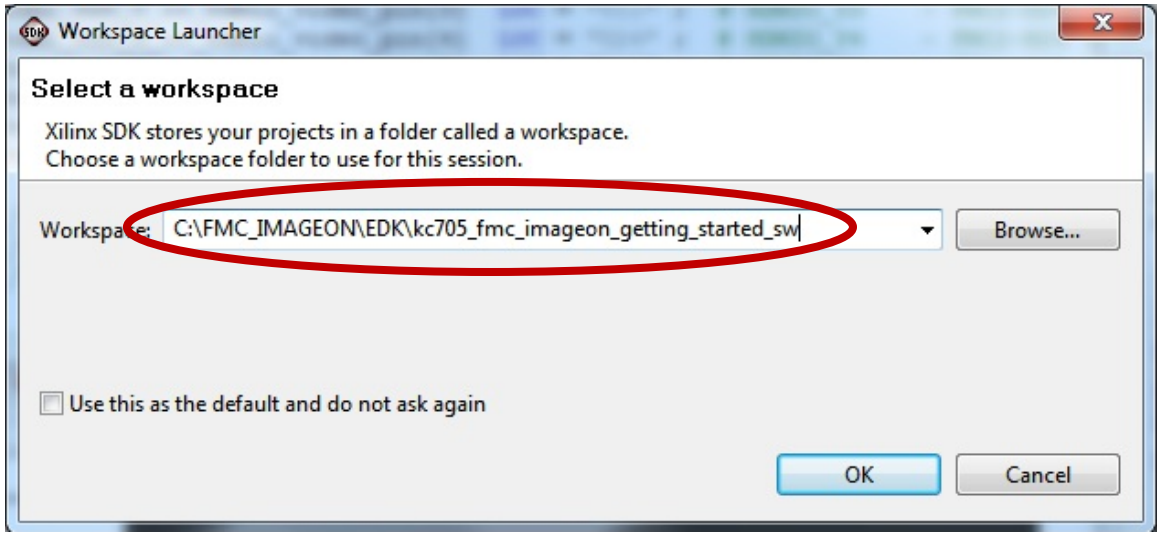
Open the SDK workspace corresponding to your FMC carrier:

C:\FMC_IMAGEON\EDK\{carrier}_fmc_imageon_getting_started_sw

This can be done by manually opening SDK, or via the export feature in XPS (as described in the previous section).

You will see the Workspace Launcher dialog box.

Click the **Browse** button and specify the SDK workspace directory corresponding to your FMC carrier. The following image illustrates the workspace directory for the KC705 carrier.



In the Project Explorer window, you show see three items:

{**carrier**}**_fmc_imageon_getting_started_hw** => this is the hardware design

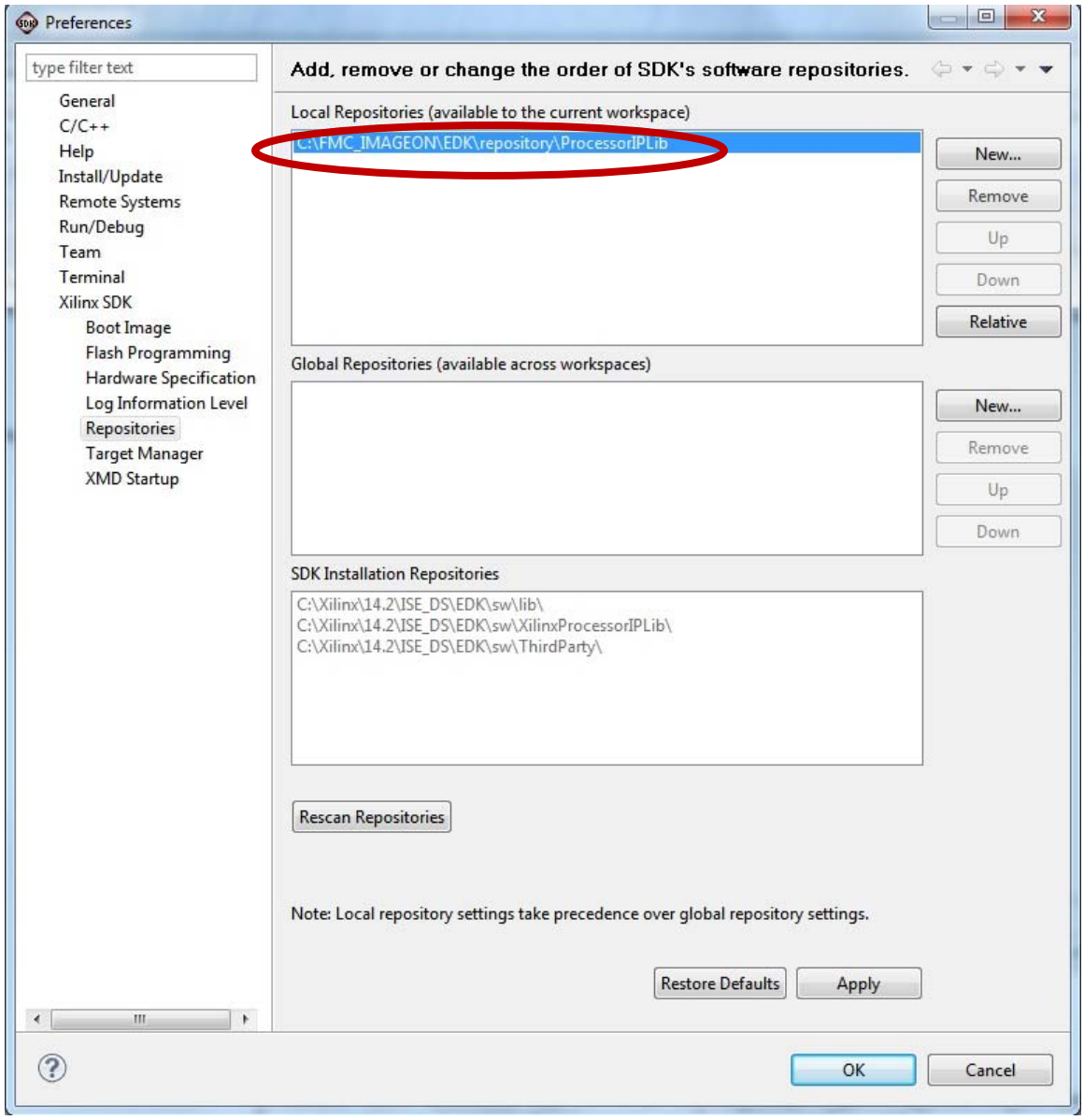
standalone_bsp_0 => this is the board support package, which contains all the drivers

{carrier}_fmc_imageon_getting_started_sw => this is the software application

If you are using a Zynq-7000 based carrier, you will also have an additional item:

zynq_fsbl_0 => this is the first stage boot loader, used to boot from SD card

The SDK workspace makes use of a repository containing several drivers specific to this reference design. To view the reference to this repository, select **Xilinx Tools => Repositories** from the menu.



Assuming that the archive was extracted to the C:\ drive, the location of the local repository will be:

C:\FMC_IMAGEON\EDK\repository\ProcessorIPLib

If this is not the case, you must change the location of the repository in the **Local Repositories** section.

When done, click **OK** to keep modification.

Select the software application, **{carrier}_fmc_imageon_getting_started_sw**, then right-click.

From the pop-up menu, select the **Build Project** to recompile the application, and all related dependencies.

Loading the Reference Design with SDK

The design can be loaded from SDK.

First, the hardware design needs to be loaded.

This can be done from the menu by selecting **Xilinx Tools** => **Program FPGA**

For Zynq-7000 based carriers, do not specify an ELF file.

For other carriers, specify **bootloop** for the ELF file.

Next, the software application needs to be loaded.

Select the software application, **{carrier}_fmc_imageon_getting_started_sw**, then right-click.

From the pop-up menu, select the **Run As => Launch on Hardware** to download and run the application on hardware.

You should see the following appear on your serial console:

```
------------------------------------------------------
--             Text-based Console for               --
--            FMC-IMAGEON Getting Started           --
------------------------------------------------------
General Commands:
        help       Print the Top-Level menu Help Screen
        quit       Exit console (if applicable)
        verbose    Toggle verbosity on/off
        delay      Wait for specified delay
        mem        Memory accesses
I2C Commands
        iic0       IIC accesses on FMC-IPMI I2C chain
        iic1       IIC accesses on FMC-IMAGEON I2C chain
VITA Commands
        vita       VITA commands (init, status, ...)
        vspi       SPI accesses to VITA sensor
        vreg       Memory accesses to VITA receiver
        again      Analog gain (0-10)
        dgain      Digital gain (0-4095) where 128 corresponds to 1.00
        exposure   Exposure time (1-99) in percentage of frame period
Image Processing Pipeline Commands
        cfa        Color Filter Array Interpolation configuration
Video Source Selection
        video      Select Video source (tpg, hdmi, vita)
        tpg        Test Pattern Generator configuation (menu)
```

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

AVNET>
```

Modifying the EDK Reference Design

Overview

Depending on your VITA-2000 image sensor, you will probably have noticed some white dots on the monitor. These will become more apparent if you cover the lens to make the background dark.

These are defective pixels, and are normal. Image sensor manufacturers tolerate certain types of defective pixels, assuming that they can be corrected downstream by an image processing pipeline.

The following image illustrates typical acceptance criteria for defective pixels.

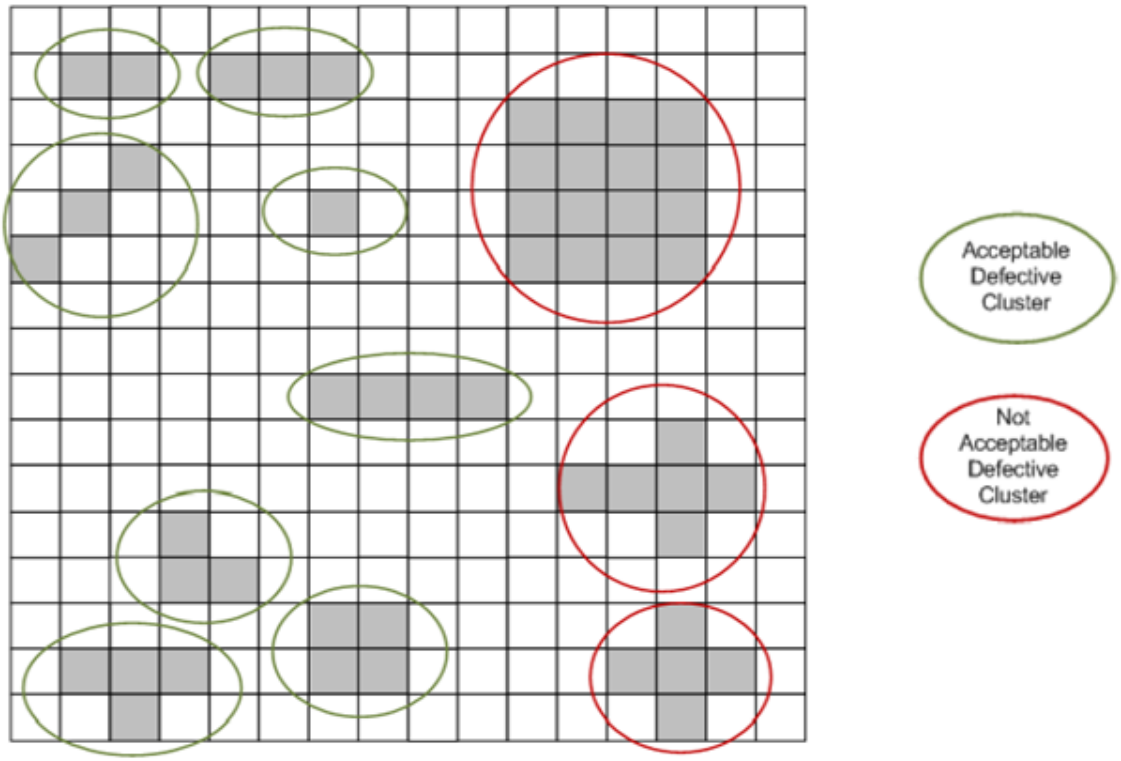


Figure 11 – Typical Defective Pixel Acceptance Criteria

This section describes how to modify the EDK reference design to add the Xilinx Defective Pixel Correction core, which will automatically detect which pixels are defective, and correct them.

Modifying the Hardware with XPS

If not done so already, open the XPS project corresponding to your FMC carrier:

C:\FMC_IMAGEON\EDK\{carrier}_fmc_imageon_getting_started_hw\system.xmp

Add the “Defective Pixel Correction” PCORE to the design.

1. From the IP catalog, expand **EDK Install** => **Video and Image Processing** and double-click on **Defective Pixel Correction** to add it.
   A message appears asking if you want to add the v_spc 6.00.a IP instance to your design.
2. Click **Yes**.
   The configuration window for **v_spc** opens.
3. Leave all parameters as they are.
4. Click **OK**.
5. A message window opens with the message "v_spc IP with version number 6.00.a is instantiated with name v_spc_0". It will ask you to determine to which processor to connect.
   Keep the default choice of processor as “microblaze_0” or "processing_system7_0".
6. Click **OK**.
   There are a few connections that are not done automatically and will be done manually after all cores have been added to the design.

Connect the bus interfaces in the design.

1. Click the **Bus Interfaces** tab, which lists the IPs and their bus connections. Expand `v_vid_in_axi4s_0`, `v_spc_0`, and `v_cfa_0`.
2. Connect the `v_spc_0`’s M_AXIS_VIDEO interface to the `v_vid_in_axi4s_0`’s S_AXIS_VIDEO interface.
3. Connect the `v_cfa_0`’s M_AXIS_VIDEO interface to the `v_spc_0`’s S_AXIS_VIDEO interface.

Connect the ports in the design.
We’ll do this manually in the MHS file, then double-check with graphical user interface.

1. Click on the **Project** tab.
2. In the **Project Files** section, double-click on the `{module name}.mhs` file.
3. Make the following edits to the file:
   a. Connect the ‘**aclk**’ port to the video clock ‘**vid_out_clk**’

```
BEGIN v_spc
 PARAMETER INSTANCE = v_spc_0
 PARAMETER HW_VER = 6.00.a
 PARAMETER C_BASEADDR = 0x7a000000
 PARAMETER C_HIGHADDR = 0x7a00ffff
 BUS_INTERFACE S_AXI = axi_interconnect_1
 BUS_INTERFACE S_AXIS_VIDEO = v_cfa_0_M_AXIS_VIDEO
 BUS_INTERFACE M_AXIS_VIDEO = v_spc_0_M_AXIS_VIDEO
 PORT s_axi_aclk = axi4lite_0_clk
 PORT aclk = vid_out_clk
END
```

4. Click the **Ports** tab, which lists the IPs and their ports. Expand `v_spc_0`.
5. Review the following IP connections. This will confirm that your manual edits in the MHS file were done correctly. If any of these are not connected, double-check the MHS file and update accordingly

IP	Port	Connection
v_spc_0	(BUS_IF) S_AXIS_VIDEO::aclk	vsrc_sel_0::video_clk
	(BUS_IF) M_AXIS_VIDEO::aclk	vsrc_sel_0::video_clk

Table 2 – Defective Pixel Correction – Manual Ports Connections

Run Design Rule Check. This can be invoked from the menu by selecting

Project => **Design Rule Check**

Ensure there are no errors in the console.

NOTE : If there are errors, double-check the steps you followed.

In the menu, select

Hardware => **Generate Bitstream.**

This will rebuild the hardware design.

The next step is to export the new hardware design files to SDK.

In the menu, select

Project => **Export Hardware Design to SDK …**

Click on the **Export and Launch SDK** button.

You may see some **Design License Status** dialog boxes.

Ensure that you have the “Hardware Evaluation” licenses for each of the cores, then click **OK**.

This will open SDK, which is described in the next section.

Modifying the Software with SDK

If not done so already, open the SDK workspace

C:\FMC_IMAGEON\EDK\{carrier}_fmc_imageon_getting_started_sw

In the **Project Explorer** window, expand the software application {carrier}_fmc_imageon_getting_started_sw.

Edit the **fmc_imageon_demo.h** file as follows:

1. Near the top of the file, add the following include statement:
   ```
   #include “dpc.h”
   ```

2. In the **struct_fmc_imageon_demo_t** data structure definition, add the following new element:

```
Xuint32 uBaseAddr_DPC;
```

Edit the **main.c** file as follows:

1. In the **main( )** function, add the following new assignment:
   ```
   fmc_imageon_demo.uBaseAddr_DPC = XPAR_DPC_0_BASEADDR;
   ```

Edit the **fmc_imageon_demo.c** file as follows:

1. In the **fmc_imageon_demo_enable_ipipe( )** function, after the CFA initialization code, add the following DPC initialization code:
   ```
   DPC_Reset( pDemo->uBaseAddr_DPC );
   DPC_ClearReset( pDemo->uBaseAddr_DPC );
   DPC_Enable( pDemo->uBaseAddr_DPC );
   DPC_RegUpdateDisable( pDemo->uBaseAddr_DPC );
   DPC_WriteReg( pDemo->uBaseAddr_DPC,
                 DPC_REG05_THRESH_PIXEL_AGE, 100 );
   DPC_RegUpdateDisable( pDemo->uBaseAddr_DPC );
   OS_PRINTF("\tDPC done\r\n");
   ```

Select the software application, **{carrier}_fmc_imageon_getting_started_sw**, then right-click.

From the pop-up menu, select the **Build Project** to recompile the application, and all related dependencies.

Loading the modified Reference Design with SDK

Load the design to hardware, and run the demonstration again.

You should notice that the defective pixels are now fixed.

EDK Repository - PCOREs

The following table contains the complete list of PCOREs provided in the EDK repository and where to find documentation:

PCORE	Version	Description
FMC-IMAGEON Specific PCOREs		
fmc_imageon_hdmii_in	1.03 a	The fmc_imageon_hdmii_in PCORE provides a connection to the ADV7611 HDMI Receiver on the FMC-IMAGEON module.
fmc_imageon_hdmii_out	1.05 a	The fmc_imageon_hdmii_out PCORE provides a connection to the ADV7511 HDMI Transmitter on the FMC-IMAGEON module.
fmc_imageon_vita_receiver	1.13 a	The fmc_imageon_vita_receiver PCORE implements the Serial LVDS receiver for the VITA image sensor.
Xilinx Video IP PCOREs		

Table 3 – IP Repository – PCORE Overview

The next sections document the PCOREs in greater detail.

fmc_imageon_hdmi_in (1.03 a)

The fmc_imageon_hdmi_in PCORE provides a connection to the ADV7611 HDMI Receiver chip on the FMC-IMAGEON module. The source for the PCORE can be found in the following location:

FMC_IMAGEON\EDK\repository\AvnetProcessorIPLib\pcores\
fmc_imageon_hdmi_in_v1_03_a\

Functionality

The following figure illustrates the functionality of the fmc_imageon_hdmi_in PCORE.

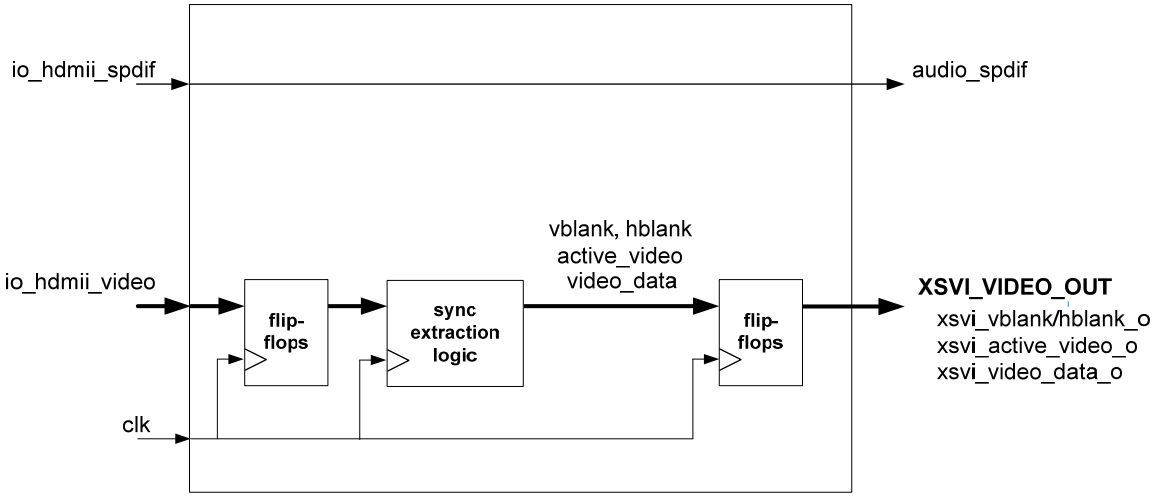


Figure 12 – fmc_imageon_hdmi_in – Block Diagram

The ADV7611 provides a 1 bit audio in SPDIF format which is simply passed through for the internal EDK design.

The ADV7611 provides a 16 bit video in YCbCr 4:2:2 format, with embedded syncs. The PCORE extracts the sync signals and creates a Xilinx Streaming Video interface (XSVI) bus interface for the EDK design.

Parameters

Parameter	Description
C_FAMILY	FPGA Family, determines implementation. Supported values are : spartan6 virtex6 kintex7
C_DATA_WIDTH	Video data width. Supported values are: **16** (default)

Table 4 – fmc_imageon_hdmi_in – Parameter List

Ports

Port	Direction	Bus Interface	Description
clk	I	-	Clock input (pixel rate)
io_hdmii_spdif	I	-	Audio Data - SPDIF
io_hdmii_video[15:0]	I	-	Video input from ADV7611 - 16 bit YCbCr 4:2:2 - embedded syncs
audio_spdif	O	-	Audio input from ADV7611 - SPDIF
xsvi_active_video_o	O	XSVI_VIDEO_OUT	Data Enable strobe
xsvi_vblank_o	O	XSVI_VIDEO_OUT	Vertical Sync strobe
xsvi_hblank_o	O	XSVI_VIDEO_OUT	Horizontal Sync strobe
xsvi_video_data_o[15:0]	O	XSVI_VIDEO_OUT	Video pixel data - 16 bit YCbCr 4:2:2

Table 5 – fmc_ imageon_hdmi_in – Port List

A bus interface called XSVI_VIDEO_OUT has been defined for the fmc_imageon_hdmi_in PCORE outputs. In EDK, these outputs can either be accessed individually by their port names or as a group by their bus interface name.

fmc_imageon_hdmi_out (1.05 a)

The fmc_imageon_hdmi_out PCORE provides a connection to the ADV7511 HDMI Transmitter on the FMC-IMAGEON module. The source for the PCORE can be found in the following location:

FMC_IMAGEON\EDK\repository\AvnetProcessorIPLib\pcores\
fmc_imageon_hdmi_out_v1_05_a\

Functionality

The following figure illustrates the functionality of the fmc_imageon_hdmi_out PCORE.

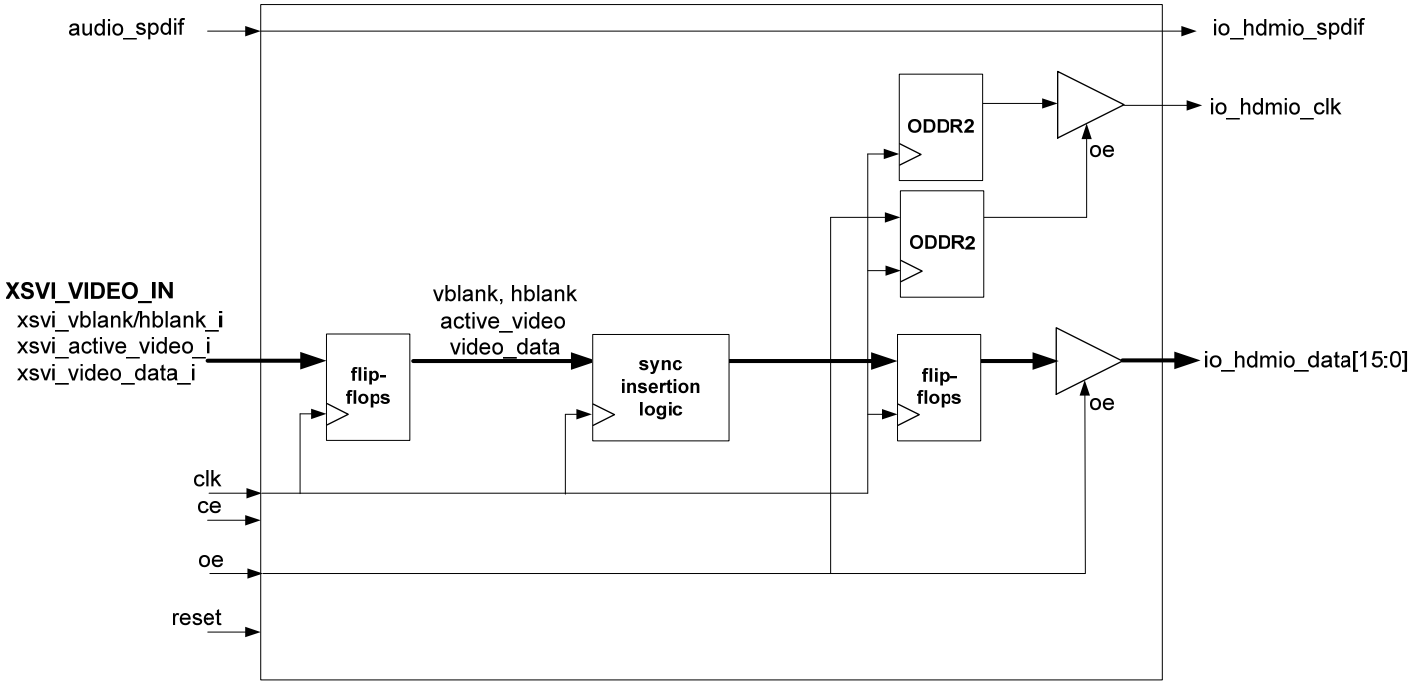


Figure 13 – fmc_imageon_hdmi_out – Block Diagram

A 1 bit audio signal in SPDIF format is passed through to the ADV7511 device.

The PCORE receives a Xilinx Streaming Video Interface in 16 bit YCbCr 4:2:2 format, embeds

the active_video, hblank, and vblank syncs, before driving to the ADV7511 HDMI transmitter.

The clock sent to the ADV7511 output interface is generated with ODDR2 registers in the IOB fabric.

All outputs are enabled with the oe port. This allows the outputs to the FMC connector to be tri-stated until the FMC module has been correctly identified.

Parameters

Parameter	Description
C_FAMILY	FPGA Family, determines implementation. Supported values are : spartan6 virtex6 kintex7
C_DATA_WIDTH	Video data width. Supported values are: **16** (default)

Table 6 – fmc_imageon_hdmi_out – Parameter List

Ports

Port	Direction	Bus Interface	Description
clk	I	-	Clock input (pixel rate)
oe	I	-	Output Enable (can be used to disable all outputs to FMC connector)
audio_spdif	I	-	Audio input - SPDIF
xsvi_active_video_i	I	XSVI_VIDEO_IN	Data Enable strobe
xsvi_vblank_i	I	XSVI_VIDEO_IN	Vertical Sync strobe
xsvi_hblank_i	I	XSVI_VIDEO_IN	Horizontal Sync strobe
xsvi_video_data[15:0]	I	XSVI_VIDEO_IN	Video pixel data - 16 bit YCbCr 4:2:2
io_hdmio_spdif	O	-	Audio output to ADV7511
io_hdmio_clk	O	-	Clock output to ADV7511
io_hdmio_video[15:0]	O	-	Video output to ADV7511 - 16 bit YCbCr 4:2:2 - embedded syncs

Table 7 – fmc_imageon_hdmi_out – Port List

A bus interface called XSVI_VIDEO_IN has been defined for the fmc_hdmi_hdmi_out PCORE inputs. In EDK, these outputs can either be accessed individually by their port names or as a group by their bus interface name.

fmc_imageon_vita_receiver (1.13 a)

The source for the PCORE can be found in the following location:

FMC_IMAGEON\EDK\repository\AvnetProcessorIPLib\pcores\
fmc_imageon_vita_receiver_v1_13_a\

Functionality

The following figure illustrates the functionality of the fmc_imageon_vita_receiver PCORE. The top level isolates the user logic from the AXI bus with an IPIF adapter.

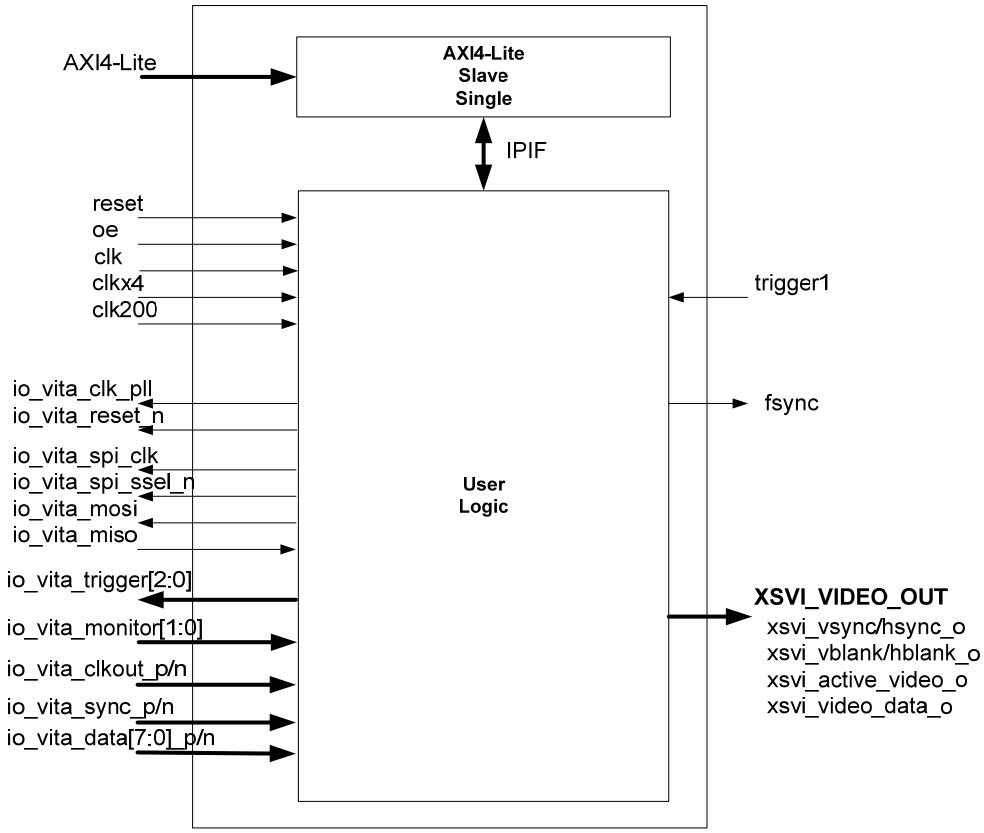


Figure 14 – fmc_imageon_vita_receiver – Block Diagram

The following figure illustrates the user_logic sub-module.

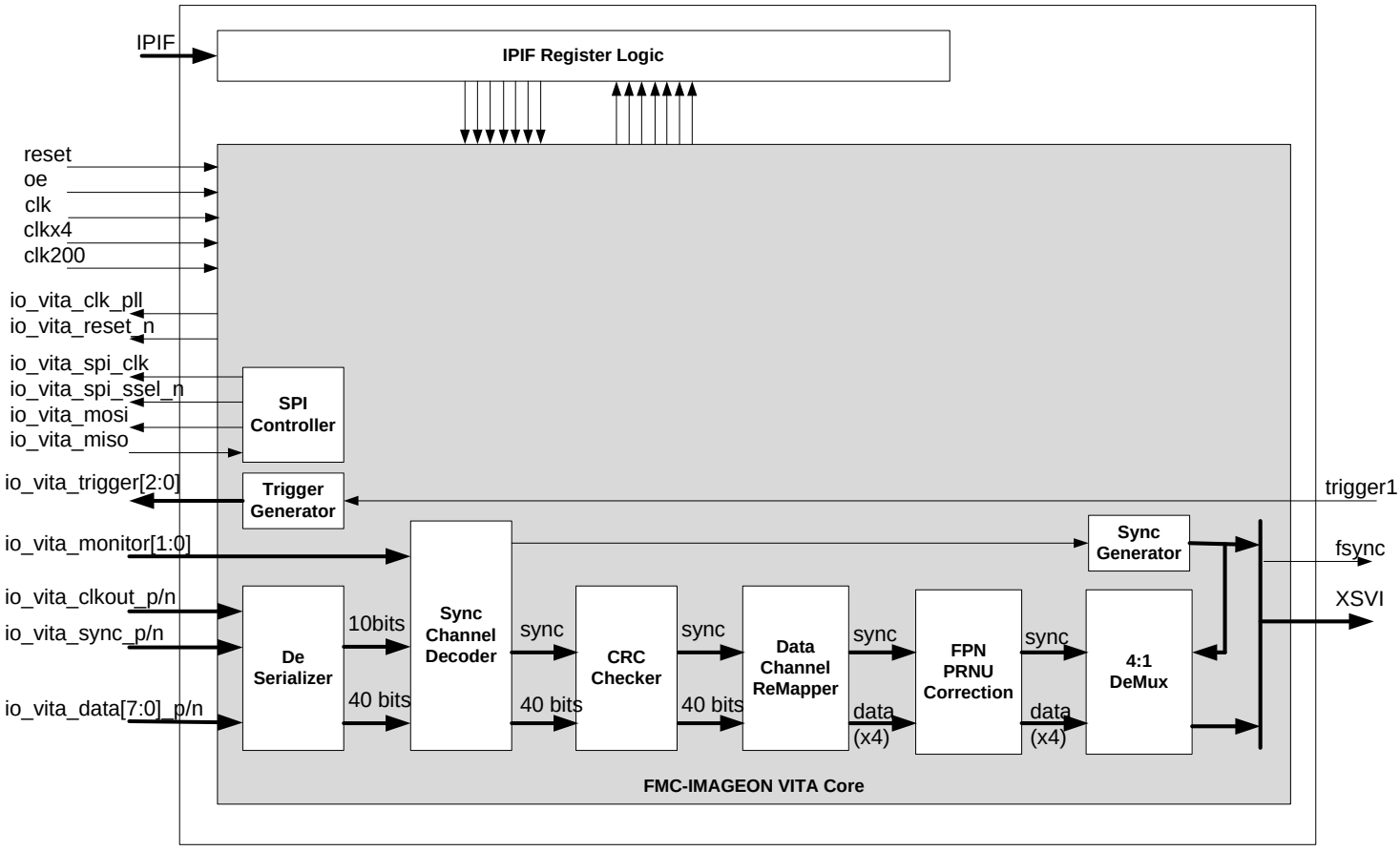


Figure 15 – fmc_imageon_vita_receiver/user_logic – Block Diagram

Memory mapped registers are used to configure the VITA receiver. These registers are implemented directly in the user_logic module.

The fmc_imageon_vita_core module implements the vita receiver. This module can be reused by itself if the AXI wrapper is not needed or desired.

The “De-Serializer” module implements the Serial LVDS receiver.

The “Sync Channel Decoder” module interprets the SYNC channel and generates appropriate synchronization signals.

The “CRC Checker” verifies the integrity of the Serial LVDS link by validating the CRC checksums that are sent with each line of video.

The “Sync Generator” generates the standard VSYNC/HSYNC, VBLANK/HBLANK, ACTIVE_VIDEO synchronization signals needed to create the final Xilinx Streaming Video Interfacae (XSVI).

The “4:1 DeMux” module takes the 4 parallel pixel values synchronous to clock “clk”, and re-generates a single stream of pixels synchronous to clock “clkx4”. This module contains a FIFO that can store up to 6 lines of active video.

The fsync_o port generates a pulse that is active for 1 cycle at the start of each new frame. It can be used to synchronize video DMA transfers.

Parameters

Parameter	Description
C_FAMILY	FPGA Family, determines implementation. Supported values are : virtex6 kintex7
C_XSVI_DATA_WIDTH	Video data width for XSVI input bus interface. Supported values are: 8 **10** (default) 16 24 40 64
C_XSVI_DIRECT_OUTPUT	When active, the video data contains 4 parallel pixel values: **0** (default) 1

Table 8 – fmc_imageon_vita_receiver – Parameter List

In default mode, the C_XSVI_DIRECT_OUTPUT will be 0, in which case, the incoming video data will be de-multiplexed, before being sent to the XSVI bus interface. In this case, the video data is synchronous to the clkx4 port.

When the C_XSVI_DIRECT_OUTPUT is set to 1, the incoming video data will be output directly to the XSVI bus interface. In this case, the video data is synchronous to the clk port.

Ports

Port	Direction	Bus Interface	Description
*	*	S_AXI	AXI4-Lite Slave Port
reset	I	-	Reset
oe	I	-	Output enable
clk	I	-	Clock input (vita ref clock)
clkx4	I	-	Clock input (pixel rate)
clk200	I	-	200 MHz input clock
xsvi_active_video_o	O	XSVI_VIDEO_OUT	Data Enable strobe
xsvi_vsync_o	O	XSVI_VIDEO_OUT	Vertical Sync strobe
xsvi_hsync_o	O	XSVI_VIDEO_OUT	Horizontal Sync strobe
xsvi_vblank_o	O	XSVI_VIDEO_OUT	Vertical Sync strobe
xsvi_hblank_o	O	XSVI_VIDEO_OUT	Horizontal Sync strobe
xsvi_video_data_o[]	O	XSVI_VIDEO_OUT	Video pixel data
io_vita_clk_pll	O	-	reference clock to VITA sensor
io_vita_reset_n	O	-	Active low reset to VITA sensor
io_vita_trigger[2:0]	O	-	Trigger strobes to VITA sensor
Io_vita_monitor[1:0]	I	-	Monitor strobes from VITA sensor
io_vita_spi_sclk	O	-	SPI clock to VITA sensor
io_vita_spi_ssel_n	O	-	SPI select to VITA sensor
io_vita_spi_mosi	O	-	SPI data to VITA sensor
io_vita_spi_miso	I	-	SPI data from VITA sensor
io_vita_clk_out_p/n	I	-	Differential clock from VITA sensor
io_vita_sync_p/n	I	-	Differential sync from VITA sensor
io_vita_data[7:0]_p/n	I	-	Differential data from VITA sensor

Table 9 – fmc_imageon_vita_receiver – Port List

Registers

Offest	Register	R/W	Description
0x00000000	SPI_CONTROL	read/write	[0] VITA_RESET [1] SPI_RESET [8] SPI_STATUS_BUSY [9] SPI_STATUS_EROR [16] SPI_TXFIFO_FULL [23] SPI_RXFIFO_EMPTY
0x00000004	SPI_TIMING	read/write	[[15:0] SPI_TIMING
0x00000008	SPI_TXFIFO_DATA	read/write	[31:0]
0x0000000C	SPI_RXFIFO_DATA	read/write	[31:0]
0x00000010	ISERDES_CONTROL		[0] ISERDES_RESET [1] ISERDES_AUTO_ALIGN [2] ISERDES_ALIGN_START [3] ISERDES_FIFO_ENABLE [8] ISERDES_CLK_READY [9] ISERDES_ALIGN_BUSY

			[10] ISERDES_ALIGNED
0x00000014	ISERDES_TRAINING	read/write	[9:0] ISERDES_TRAINING
0x00000018	ISERDES_MANUAL_TAP	read/write	
0x0000001C	-	-	-
0x00000020	DECODER_CONTROL	read/write	[0] DECODER_RESET [1] DECODER_ENABLE
0x00000024	DECODER_STARTODDEVEN	read/write	
0x00000028	DECODER_CODES_LS_LE	read/write	[15:0] CODE_LS (line start) [31:16] CODE_LE (line end)
0x0000002C	DECODER_CODES_FS_FE	read/write	[15:0] CODE_FS (frame start) [31:16] CODE_FE (frame end)
0x00000030	DECODER_CODES_BL_IMG	read/write	[15:0] CODE_BL (black line) [31:16] CODE_IMG (image line)
0x00000034	DECODER_CODES_TR_CRC	read/write	[15:0] CODE_TR (training) [31:16] CODE_CRC (crc)
0x00000038	DECODER_CNT_BLACK_LINES	read	[31:0] CNT_BLACK_LINES
0x0000003C	DECODER_CNT_IMAGE_LINES	read	[31:0] CNT_IMAGE_LINES
0x00000040	DECODER_CNT_BLACK_PIXELS	read	[31:0] CNT_BLACK_PIXELS
0x00000044	DECODER_CNT_IMAGE_PIXELS	read	[31:0] CNT_IMAGE_PIXELS
0x00000048	DECODER_CNT_FRAMES	read	[31:0] CNT_FRAMES
0x0000004C	DECODER_CNT_WINDOWS	read	[31:0] CNT_WINDOWS
0x00000050	DECODER_CNT_CLOCKS	read	[31:0] CNT_CLOCKS
0x00000054	DECODER_CNT_START_LINES	read	[31:0] CNT_START_LINES
0x00000058	DECODER_CNT_END_LINES	read	[31:0] CNT_END_LINES
0x0000005C	SYNCGEN_DELAY	read/write	[15:0] DELAY (# of clk cycles)
0x00000060	SYNCGEN_HTIMING1	read/write	[15:0] HACTIVE (active video) [31:0] HFPORCH (front porch)
0x00000064	SYNCGEN_HTIMING2	read/write	[14:0] HSYNCW (sync width) [15] HSYNCP (sync polarity) [31:16] HBPORCH (back porch)
0x00000068	SYNCGEN_VTIMING1	read/write	[15:0] VACTIVE (active video) [31:0] VFPORCH (front porch)
0x0000006C	SYNCGEN_VTIMING2	read/write	[14:0] VSYNCW (sync width) [15] VSYNCP (sync polarity) [31:16] VBPORCH (back porch)
0x00000070	CRC_CONTROL	read/write	[0] CRC_RESET [1] CRC_INITVALUE
0x00000074	CRC_STATUS	read/write	[3:0] CRC_STATUS
0x00000078	REMAPPER_CONTROL	read/write	[2:0] REMAPPER_WRITE_CFG [6:4] REMAPPER_MODE
0x0000007C	-	-	-
0x00000080	FPN_PRNU_VALUES[31:0]	read/write	[7:0] PRNU_0 [15:8] FPN_0 [23:16] PRNU_1 [31:24] FPN_1
0x00000084	FPN_PRNU_VALUES[63:32	read/write	[7:0] PRNU_2 [15:8] FPN_2

			[23:16] PRNU_3 [31:24] FPN_3
0x00000088	FPN_PRNU_VALUES[95:64]	read/write	[7:0] PRNU_4 [15:8] FPN_4 [23:16] PRNU_5 [31:24] FPN_5
0x0000008C	FPN_PRNU_VALUES[127:96]	read/write	[7:0] PRNU_6 [15:8] FPN_6 [23:16] PRNU_7 [31:24] FPN_7
0x00000090	FPN_PRNU_VALUES[159:128]	read/write	[7:0] PRNU_8 [15:8] FPN_8 [23:16] PRNU_9 [31:24] FPN_9
0x00000094	FPN_PRNU_VALUES[191:160]	read/write	[7:0] PRNU_10 [15:8] FPN_10 [23:16] PRNU_11 [31:24] FPN_11
0x00000098	FPN_PRNU_VALUES[223:192]	read/write	[7:0] PRNU_12 [15:8] FPN_12 [23:16] PRNU_13 [31:24] FPN_13
0x0000009C	FPN_PRNU_VALUES[255:224]	read/write	[7:0] PRNU_14 [15:8] FPN_14 [23:16] PRNU_15 [31:24] FPN_15
0x000000A0	-	-	-
0x000000A4	-	-	-
0x000000A8	-	-	-
0x000000AC	-	-	-
0x000000B0	-	-	-
0x000000B4	-	-	-
0x000000B8	-	-	-
0x000000BC	-	-	-
0x000000C0	DECODER_CNT_MONITOR0_HIGH	read	[31:0] MONITOR0_HIGH
0x000000C4	DECODER_CNT_MONITOR0_LOW	read	[31:0] MONITOR0_LOW
0x000000C8	DECODER_CNT_MONITOR1_HIGH	read	[31:0] MONITOR1_HIGH
0x000000CC	DECODER_CNT_MONITOR1_LOW	read	[31:0] MONITOR1_LOW
0x000000D0	-	-	-
0x000000D4	-	-	-
0x000000D8	-	-	-
0x000000DC	TRIGGEN_EXT_DEBOUNCE	read/write	[31:0] EXT_DEBOUNCE
0x000000E0	TRIGGEN_CONTROL	read/write	[2:0] TRIGGEN_ENABLE [6:4] SYNC2READOUT [8] READOUTTRIGGER [16] EXT_POLARITY [24] CNT_UPDATE

			[30:28] GEN_POLARITY
0x000000E4	TRIGGEN_DEFAULT_FREQ	read/write	[31:0] TRIG_DEFAULT_FREQ
0x000000E8	TRIGGEN_TRIG0_HIGH	read/write	[31:0] TRIG0_HIGH
0x000000EC	TRIGGEN_TRIG0_LOW	read/write	[31:0] TRIG0_LOW
0x000000F0	TRIGGEN_TRIG1_HIGH	read/write	[31:0] TRIG1_HIGH
0x000000F4	TRIGGEN_TRIG1_LOW	read/write	[31:0] TRIG1_LOW
0x000000F8	TRIGGEN_TRIG2_HIGH	read/write	[31:0] TRIG2_HIGH
0x000000FC	TRIGGEN_TRIG2_LOW	read/write	[31:0] TRIG2_LOW

Table 10 – fmc_imageon_vita_receiver – Registers

The SYNCGEN_* registers define the settings for the video sync generator's video timing. The following figure illustrates how the synchronization signals will be generated from these settings.

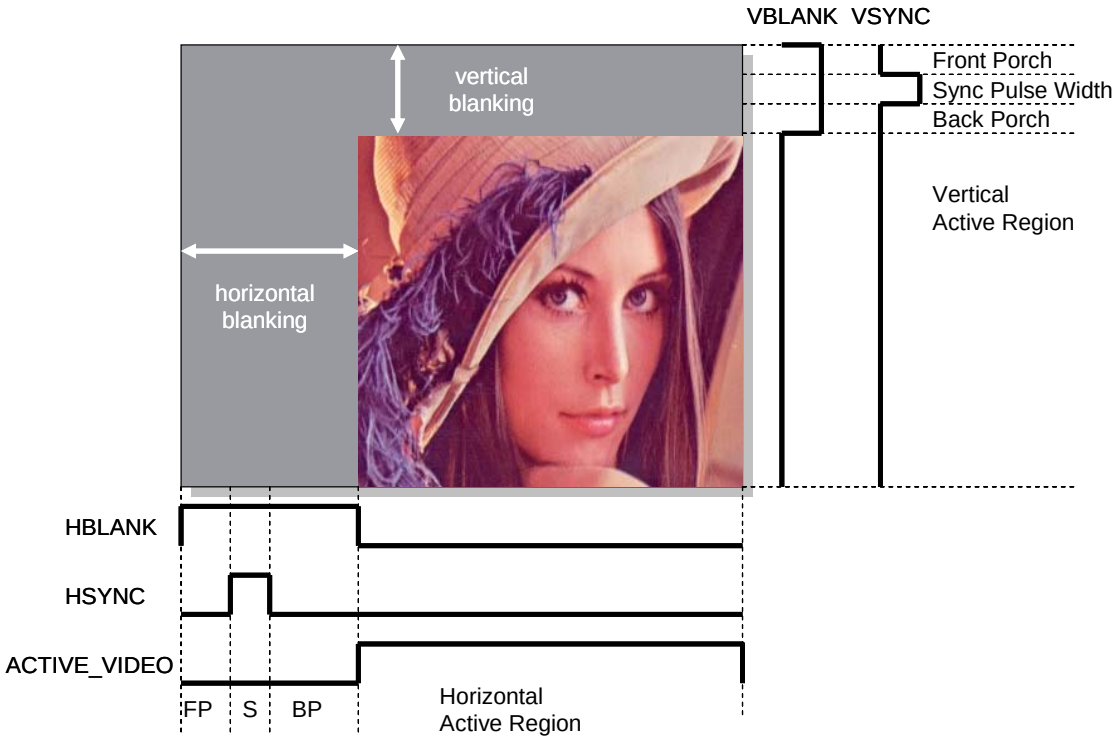


Figure 16 – fmc_imageon_vita_receiver – SYNCGEN Video Timing Diagram

Simulation Testbench

An HDL simulation testbench (ISIM) is provided for the fmc_imageon_vita_core works, and can be found in the following location.

FMC_IMAGEON\EDK\repository\AvnetProcessorIPLib\pcores\
fmc_imageon_vita_receiver_v1_13_a\tb\isim\vita_2000_tb.xise

This simulation testbench DOES NOT simulate the AXI interface, only the fmc_imageon_vita_core module.

EDK Repository - Software Libraries & Drivers

The following table contains the complete list of software libraries and drivers provided in the EDK repository and where to find documentation:

Driver	Version	Description
Generic Software Libraries		
fmc_iic_sw	2.03 a	Library which provides a generic I2C controller API.
fmc_ipmi_sw	2.02 a	Library which provides an API to access the contents of the FMC IPMI EEPROMs.
FMC-IMAGEON specific Software Libraries		
fmc_imageon_sw	1.06 a	Library which provides an API to perform I2C configuration of the FMC-IMAGEON module's HDMI input, HDMI output, and Video clock synthesizer peripherals.
FMC-IMAGEON specific Drivers		
fmc_imageon_vita_receiver	1.13 a	Library which provides an API to perform SPI configuration of the FMC-IMAGEON module's VITA receiver.

Table 11 – IP Repository – Software Libraries Overview

fmc_iic_sw (2.03 a)

The documentation for the FMC-IIC software library can be found in the following location:

FMC_IMAGEON\EDK\repository\AvnetProcessorIPLib\sw_services\
fmc_iic_sw_v2_03_a\doc\FMC-IIC_Software_Guide_1_03.pdf

fmc_ipmi_sw (2.02 a)

The documentation for the FMC-IPMI software library can be found in the following location:

FMC_IMAGEON\EDK\repository\AvnetProcessorIPLib\sw_services\
fmc_ipmi_sw_v2_02_a\doc\FMC-IPMI_Software_Guide_1_01.pdf

fmc_imageon_sw (1.06 a)

The FMC-IMAGEON software library can be found in the following location:

FMC_IMAGEON\EDK\repository\AvnetProcessorIPLib\sw_services\
fmc_imageon_sw_v1_05_a\fmc_imageon.c/.h

The FMC-IMAGEON software library consists of the following functions.

Function	**Description**
General Functions	
fmc_imageon_init	Initialize instance of FMC-IMAGEON software library.
I2C Multiplexer Function	
fmc_imageon_iic_mux_reset	Reset FMC-IMAGEON's I2C multiplexer
fmc_imageon_iic_mux	Select the I2C multiplexer's active ports.
I2C Configuration Functions	
fmc_imageon_iic_config2	Perform generic I2C configuration using specified register/value matrix
fmc_imageon_iic_config3	Perform generic I2C configuration using specified device/register/value matrix
Video Clock Synthesizer (CDCE913) Functions	
fmc_imageon_vclk_init	Initialize video clock synthesizer.
fmc_imageon_vclk_config	Set video clock synthesizer to specified frequency.
HDMI Input (ADV7611) Functions	
fmc_imageon_hdmii_init	Initialize HDMI input interface
fmc_imageon_hdmii_set_hpd	Set HDMI input interface's Hot Plug Detect (HPD) pin
fmc_imageon_hdmii_set_rst	Set HDMI input interface's Reset pin
fmc_imageon_hdmii_get_int	Get HDMI input interface's Interrupt (INT1) pin status
fmc_imageon_hdmii_get_lock	Get HDMI input interface's lock status
fmc_imageon_hdmii_get_timing	Get HDMI input interface's detected video timing
HDMI Output (ADV7511) Functions	
fmc_imageon_hdmio_init	Initialize HDMI output interface.
fmc_imageon_hdmio_set_pd	Set HDMI output interface's Power Down (PD)
fmc_imageon_hdmio_get_hpd	Get HDMI output interface's Hot Plug Detect (HPD) pin status

DDC/EDID EEPROM Functions	
fmc_imageon_hdmii_read_edid	Read HDMI input interface's EDID EEPROM.
fmc_imageon_hdmii_write_edid	Write HDMI input interface's EDID EEPROM.
fmc_imageon_hdmio_read_edid	Read HDMI output interface's EDID EEPROM.
Delay Functions	
fmc_imageon_wait_usec	Wait for specified number of micro-seconds. Requires and calls external usleep( ) function.

Table 12 – fmc_imageon_sw – Function List

For more information on each of these functions, please refer to the software library's source code.

fmc_imageon_vita_receiver (1.13 a)

The FMC-IMAGEON VITA Receiver driver can be found in the following location

FMC_IMAGEON\EDK\repository\AvnetProcessorIPLib\pcores\
fmc_imageon_vita_receiver_v1_13_a\src\fmc_imageon_vita_receiver.c/.h

The FMC-IMAGEON VITA Receiver driver consists of the following functions.

Function	**Description**
General Functions	
fmc_imageon_vita_receiver_init	Initialize instance of FMC-IMAGEON VITA receiver driver.
Low-Level Functions	
fmc_imageon_vita_receiver_reg_read	Read value of specified VITA receiver register.
fmc_imageon_vita_receiver_reg_write	Write to specified VITA receiver register.
fmc_imageon_vita_receiver_reset	Set value of reset to VITA sensor.
fmc_imageon_vita_receiver_spi_config	Configure the VITA receiver's SPI controller
fmc_imageon_vita_receiver_spi_read	Perform SPI transaction to read VITA sensor register
fmc_imageon_vita_receiver_spi_write	Perform SPI transaction to write VITA sensor register
fmc_imageon_vita_receiver_write_sequence	Perform sequence of SPI transactions.
fmc_imageon_vita_receiver_display_sequence	Display sequence of SPI transactions (for debug purposes)
High-Level Functions	
fmc_imageon_vita_receiver_sensor_initialize	Initialize the VITA sensor
fmc_imageon_vita_receiver_get_status	Get status of VITA receiver

Table 13 – fmc_imageon_vita_receiver – Function List

For more information on each of these functions, please refer to the driver's source code.

References

All documentation supporting the ON Semiconductor Image Sensor with HDMI Input/Output FMC Bundle is available on the Avnet Design Resource Center (DRC):
http://www.em.avnet.com/fmc-imageon-v2000c

1. Getting Started with the ON Semiconductor Image Sensor with HDMI Input/output FMC Bundle
   http://www.em.avnet.com/fmc-imageon-v2000c ➔ Support Files & Downloads
2. Avnet FMC-IMAGEON – Hardware User Guide
   http://www.em.avnet.com/fmc-imageon-v2000c ➔ Support Files & Downloads

The following reference provides links to documentation supporting video application notes and video intellectual property (IP).

3. Color Filter Array Interpolation
   http://www.xilinx.com/products/ipcenter/EF-DI-CFA.htm
4. Color Correction Matrix
   http://www.xilinx.com/products/ipcenter/EF-DI-CCM.htm
5. Defective Pixel Correction
   http://www.xilinx.com/products/ipcenter/EF-DI-DEF-PIX-CORR.htm
6. Gamma Correction
   http://www.xilinx.com/products/ipcenter/EF-DI-GAMMA.htm
7. Image Edge Enhancement
   http://www.xilinx.com/products/ipcenter/EF-DI-IMG-ENHANCE.htm
8. Image Noise Reduction
   http://www.xilinx.com/products/ipcenter/EF-DI-IMG-NOISE.htm
9. Image Statistics Engine
   http://www.xilinx.com/products/ipcenter/EF-DI-IMG-STATS.htm
10. Motion Adaptive Noise Reduction
    http://www.xilinx.com/products/ipcenter/EF-DI-IMG-MA-NOISE.htm
11. RGB to YCrCb Color-Space Converter
    http://www.xilinx.com/products/ipcenter/RGB_to_YCrCb.htm
12. YCrCb to RGB Color-Space Converter

http://www.xilinx.com/products/ipcenter/YCrCb_to_RGB.htm

13. Video Direct Memory Access (DMA)
    http://www.xilinx.com/products/ipcenter/EF-DI-VID-DMA.htm
14. Video On Screen Display (OSD)
    http://www.xilinx.com/products/ipcenter/EF-DI-OSD.htm
15. Video Scaler
    http://www.xilinx.com/products/ipcenter/EF-DI-VID-SCALER.htm
16. Video Timing Controller
    http://www.xilinx.com/products/ipcenter/EF-DI-VID-TIMING.htm

Various video related reference designs:

1. XAPP521 - Bridging Xilinx Streaming Video Interface with the AXI4-Stream Protocol
   http://www.xilinx.com/support/documentation/application_notes/xapp521_XSVI_AXI4.pdf
2. XAPP739 - AXI Multi-Ported Memory Controller
   http://www.xilinx.com/support/documentation/application_notes/xapp739_axi_mpmc.pdf
3. XAPP740 - Designing High-Performance Video Systems with the AXI Interconnect
   http://www.xilinx.com/support/documentation/application_notes/xapp740_axi_video.pdf

Known Issues and Limitations

The following issues are known to exist. When applicable, the workaround used is described.

AXI4-Stream Interface

The AXI4-Stream control logic in the Video IP cores has an issue:

1) If an AXI4-Stream interface received a partial frame, which will happen frequently on startup with a live video source, the interface may freeze up.

A modified version of the VTC PCORE has been placed in the repository directory, and contains a modified version of the following AXI4-Stream control logic:

repository\v_tc_v5_00_a\hdl\vhdl**axi4s_control.vhd** (encrypted)

This modified version of the AXI4-Stream control logic no longer freezes up the AXI4-Stream interface.

Video Timing Controller PCORE (v5.00.a)

Version 5.00.a of the Video Timing Controller PCORE has a required port connection that is only visible if the “By Connection” Filters are disabled.

This is not a bug, but may be difficult to find at first.

In the Ports tab, configure the Filters by pressing the “<<” button on the right hand side.

Then enable the **By Connection => Defaults** option to view all the ports.

Video Timing Controller driver (v2.00.a shipping with 14.2)

Version 2.00.a (shipping with 14.2) of the Video Timing Controller driver has two issues:

2) There is a syntax error in the **xvtc.h** file, in the **XVtc_IntrSetLockPolarity( )** macro. The fix consists of removing the comments from the macro definition.
3) There is an error in the **xvtc_hw.h** file, for the **XVTC_RESET_RESET_MASK** definition The fix consists of changing the value of this define from 0x0000000A to 0x80000000.

A modified version of the driver has been placed in the repository directory for convenience.

RGB2YCrCb driver (v5.00.a shipping with 14.2)

Version 5.00.a (shipping with 14.2) of the RGB2YCrCb driver has two issues:

1) The **rgb2ycrcb.h** file is missing the definition of the Xuint32 data type.
   The fix consists of adding the missing **#include "xbasic_types.h"**
2) There is an error in **rgb2ycrcb.c** file, in the **RGB_set_coefficients( )** function, where the base address is hard-coded to XPAR_RGB2YCRCB_0_BASEADDR instead of using the BaseAddress argument.
   The fix consists of replacing XPAR_RGB2YCRCB_0_BASEADDR with BaseAddress.

A modified version of the driver has been placed in the repository directory for convenience.

HDMI Output – HDMIO_CLK – reducing radiated emissions

Spread Spectrum Clocking (SSC) is used on the FMC-IMAGEON module's HDMI output interface (HDMIO_CLK) to reduce radiated emissions to industry approved levels. This can be implemented using the SSC feature of the on-board TI CDCE913 video clock synthesizer.

The recommended setting is to modulate the clock using down-spread clocking by an amount of -0.75%. This technique significantly lowers the radiated emissions, while maintaining functionality on the HDMI output interface.

The following I2C registers settings will program the CDCE913 for down-spread clocking of -0.75%.

Setting	Register	Value	Description
SSC1DC	0x16[7]	0	PLL1 SSC down/center selection = down
SSC1	0x10[2:0]	011	PLL1 SSC Selection (Modulation Amount) = -0.75%

Table 14 – CECE913 I2C Register Settings for SSC

Troubleshooting

ERROR: ADV7611 has not locked on incoming video, aborting !

If you get the following output from the serial console:

```
Hello World

------------------------------------------------------
--          FMC-IMAGEON HDMI Pass-Through           --
------------------------------------------------------

FMC-IMAGEON Initialization ...
HDMI Input Initialization ...
Waiting for ADV7611 to locked on incoming video ...
        ERROR : ADV7611 has NOT locked on incoming video, aborting !
```

Your video input is not connected of not active. Please re-verify your video input source by connecting it directly to your HDMI/DVI monitor.