FMC-IMAGEON
VITA Pass-Through
Tutorial

Version 1.0

AVNET electronics marketing

Revision History

Version	Description	Date
1.0	VITA Pass-Through Tutorial • Vivado 2013.3 version	Mar 06, 2014

Table of Contents

Table of Figures

Table of Tables

About this Guide

This tutorial describes how to create a Vivado IP Integrator video design from scratch.

This manual contains the following chapters:

- Chapter “**Introduction**” describes provides an overview and features of the FMC-IMAGEON FMC module, as well as the hardware and software required for this tutorial.
- Chapter “**Tutorial Overview**” provides a general overview of this tutorial.
- Chapter “**Implement an VITA Pass-Through**” describes the steps required to implement an Embedded System that implements an VITA pass-through.
- 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 the tools and/or IP used in this tutorial.
- Appendix “**Troubleshooting**” provides a list of troubleshooting suggestions for this tutorial.

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 how the connectivity of the FMC module.

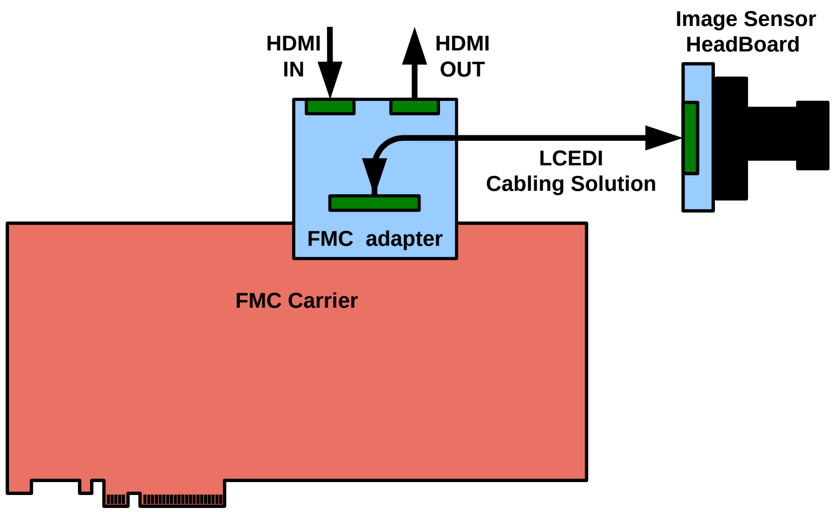


Figure 2 – FMC-IMAGEON Hardware – Connectivity Diagram

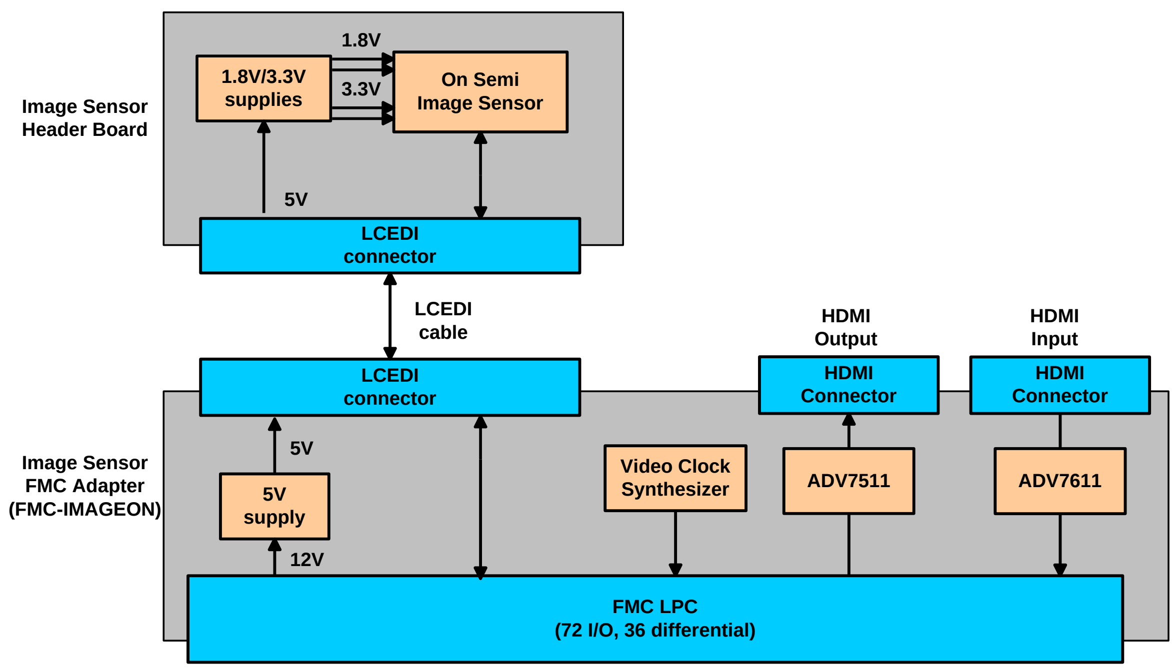


Figure 3 – FMC-IMAGEON Hardware – Block Diagram

Requirements

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

Software

The software required to build, and run the demonstrations is:

- Xilinx Vivado 2013.3
- Terminal Emulator (HyperTerminal or TeraTerm)

Hardware

The bare minimum required to run this reference design is:

- Computer with a minimum of 4 GB to complete a design[1]
- One of the following Zynq carriers
    - ZC702 (including power supply and cables)
    - ZedBoard (including power supply and cables)
- The Avnet HDMI Input/Output FMC Module (FMC-IMAGEON)
- HDMI (or DVI-D) monitor, including HDMI cable
- VITA-2000 camera module, including LCEDI cable

Zynq embedded design tools

This tutorial assumes that you have experience creating Zynq embedded designs. More specifically, it assumes a working knowledge of Vivado tools, including IPI (Vivado IP Integrator) and SDK.

If you do not have this experience, it is HIGHLY recommended that you follow the **Introduction to Zynq** on-line training course, available on zedboard.org.

http://www.zedboard.org/course/introduction-zynq

Xilinx Video IP

This tutorial will make use of several of Xilinx's Video IP cores.

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

Although it is possible to complete this tutorial without consulting the datasheets for these Video IP cores, it is strongly recommended to consult the datasheets for each Video IP core to fully understand and appreciate their potential.

Video Timing Controller

http://www.xilinx.com/products/intellectual-property/EF-DI-VID-TIMING.htm

Video Input to AXI4-Stream

http://www.xilinx.com/products/intellectual-property/video_in_to_axi4_stream.htm

AXI4-Stream to Video Output

http://www.xilinx.com/products/intellectual-property/axi4_stream_to_video_out.htm

AXI Video DMA

http://www.xilinx.com/products/intellectual-property/axi_video_dma.htm
In particular interest is the "Triple Frame Buffer Example" chapter

For the full list of Xilinx Video IP Cores, refer to the following web page:
http://www.xilinx.com/ipcenter/video/video_core_listing.htm

Setup

Before you begin this tutorial, carefully read the following sections which will describe how to extract the tutorial archive and how to test your video equipment.

Extract the tutorial archive on your computer

Extract the tutorial archive in the root of your C:\ drive. It will contain the following directories

C:\FMC_IMAGEON\2013_3\avnet_fmc_imageon_cores
C:\FMC_IMAGEON\2013_3\constraints
C:\FMC_IMAGEON\2013_3\code
C:\FMC_IMAGEON\2013_3\scripts

Tutorial Overview

This tutorial will guide you in creating a video design that implements an VITA pass-through for the Avnet HDMI Input/Output FMC module.

Reusable Components

The tutorial will make use of reusable components for the video input and video output interfaces:

- IP Cores, for each of the video interfaces on the FMC-IMAGEON module
    - FMC-IMAGEON – HDMI Input
    - FMC-IMAGEON – HDMI Output
    - FMC-IMAGEON – VITA Receiver

- TCL scripts : automatically create IP Integrator sub-modules

  For a monochrome image sensor:
    - **fmc_imageon_vita_mono.tcl** : video input path, 8 bits Y format
    - **fmc_imageon_hdmio_y.tcl** : video output path, 8 bits Y format

  For a color image sensor:
    - **fmc_imageon_vita_color.tcl** : video input path, 24 bits RGB format
    - **fmc_imageon_hdmio_rgb.tcl** : video output path, 24 bits RGB format

- XDC constraints : defines pinout and constraints for various carriers
    - **zc702_fmc_imageon_vita_pasthrough.xdc** : constraints for ZC702
    - **zedboad_fmc_imageon_vita_pasthrough.xdc** : constraints for ZedBoard

- C source code : provides example initialization code
    - **fmc_imageon_vita_passthrough.c/.h** : init code for VITA pass-through

For more information on these reusable components, refer to the **FMC-IMAGEON - IP CORES for Vivado** document.

VITA Pass-Through Overview

The user may choose to implement a video pass-through for two video formats.

One pass-through can be used for a color image sensor.

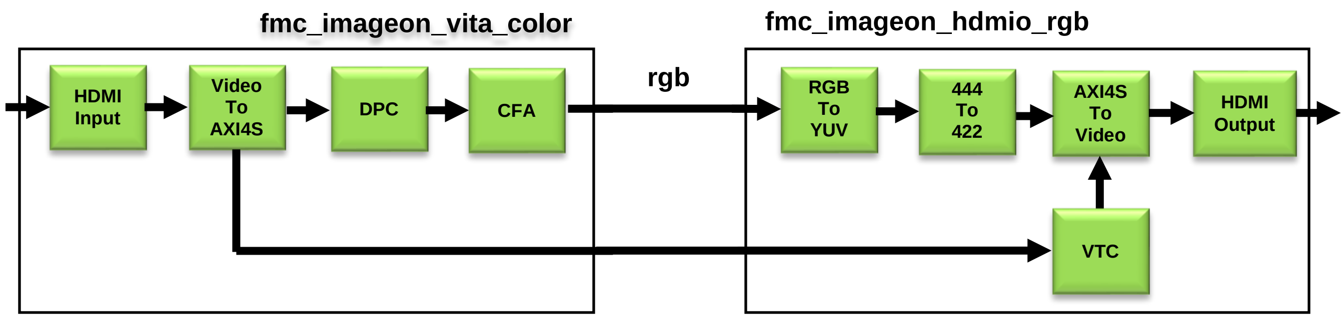


Figure 4 –VITA Pass-Through – Color Sensor

An alternate pass-through can be used for a monochrome image sensor. A monochrome sensor will only have the Y (or luma) component, which is intensity. The AXI4-Stream Protocol Converter core can be used to set the interleaved UV components (or chroma) to 0x80, effectively creating color-less pixels in the 16 bits YUV 4:2:2 video format.

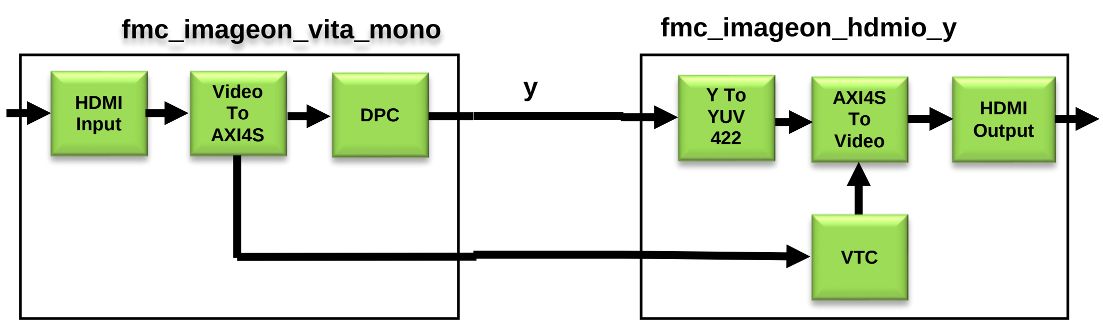


Figure 5 –VITA Pass-Through – Monochrome Sensor

Embedded Design with Processor

This design requires an embedded design that contains a processor, as well as the following peripherals:

- external memory
- serial port (UART or USB-UART)

This tutorial will create a Zynq based embedded design.

Alternatively, for non-Zynq based carriers, a MicroBlaze based embedded design can also be created. However, this is not described in this tutorial, and left as an exercise to the user.

FMC-IMAGEON - I2C Controller

Once the embedded processor design is created, an I2C controller is implemented with the following Xilinx IP core:

- AXI I2C Controller
  - not shown in the previous block diagram, this core will allow the processor to configure the FMC-IMAGEON hardware peripherals, including:
    - ADV7611 : HDMI input device
    - ADV7511 : HDMI output device
    - CDCE913 : video clock synthesizer

FMC-IMAGEON – Video Interfaces

The following cores, provided with the tutorial, will be used to interface to the VITA-2000 image sensor, and the ADV7511 device on the FMC-IMAGEON module. It is important to note that, on the FMC-IMAGEON module, the ADV7511 device is used in 16 bits YCbCr 4:2:2 with “embedded sync” mode.

- FMC-IMAGEON VITA Receiver
  - this core contains logic that will de-serialize the raw pixels from the VITA-2000 image sensor.
- FMC-IMAGEON HDMI Output
  - this core contains logic that will embed the synchronization signals in the 16 bit video data sent to the ADV7511 HDMI output device.

The VITA Receiver and HDMI Output IP cores make a generic parallel video interface available to the design.

AXI4-Stream Bridges

The following cores are used to bridge between the VITA and HDMI interfaces and the AXI4-Stream protocol.

- Video Timing Controller

    - this core is capable of:
        - detecting the video timing on a video input interface
        - (re)generating video timing for a video output interface
    - a single VTC core will be used:
        - the video timing of the video input will be detected by the detector portion of the VTC core
        - the generator portion of the VTC core will be synchronized to the detector, thus re-generating the same video timing on the output
- Video Input to AXI4-Stream
    - this core converts a generic parallel video interface (ie. DVI/HDMI) to the AXI4-Stream for Video protocol
    - the core includes a FIFO allowing the AXI4-Stream for Video interface to run on a different clock
- AXI4-Stream to Video Output
    - this core generates a generic parallel video interface (ie. DVI/HDMI) from a AXI4-Stream for Video interface
    - the core includes a FIFO allowing the AXI4-Stream for Video interface to run on a different clock

To illustrate the back-pressure capability of the AXI4-Stream interface, this video pipeline will be implemented with two separate clock domains. The input and output interfaces will be running on the VITA interface's video clock. The AXI4-Stream interface will be running on a separate clock.

Implement an VITA Pass-Through

In this section, a new Vivado project will be created, implementing the VITA pass-through design for the Avnet FMC-IMAGEON module.

Licensing the Video and Image Processing Pack IP Cores

This design uses several of the Xilinx Video and Image Processing Pack IP cores that must be licensed prior to use. Follow these steps to request an evaluation license:

1. Go to:
   www.xilinx.com/products/intellectual-property/EF-DI-VID-IMG-IP-PACK

2. Click the Evaluate link located on the upper-left of the web page,
   and follow the online instructions

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.

Create a new Vivado project

To create a new project, start the Vivado™ design and analysis software and create a project with an embedded processor system as the top level.

1. Start the Vivado 2013.3 software.
2. Select **Create New Project** to open the New Project wizard
3. Use the information in the table below to make your selections in the wizard screen

Wizard Screen	System Property	Setting or Command to Use
Project Name	Project name	Specify the project name, such as: **tutorial**
	Project location	Specify the directory in which to store the project files:

		C:\FMC_IMAGEON\2013_3
	Create Project Subdirectory	Leave this checked.
Project Type	Specify the type of project to create	Use the default selection, specify **RTL Project**.
Add Sources		Do not make any changes on this screen.
Add Existing IP		Do not make any changes on this screen.
Add Constraints		Do not make any changes on this screen.
Default Part	Specify	Select **Boards**.
	Filter	In the Family list, select **Zynq-7000**
	Board list	Select one of the following board: **Zynq-7 ZC702 Evaluation Board** or **ZedBoard Zynq Evaluation and Development Kit**
New Project Summary	Project summary	Review the project summary before clicking **Finish** to create the project.

Table 1 – New Project Settings

When you click **Finish**, the New Project wizard closes and the project you just created opens in Vivado.

Create the Embedded Hardware Design with IP Integrator

This section will guide you on how to create a basic embedded hardware design. This section may differ slightly for different carriers.

Create a new Vivado IP Integrator block diagram.

1. Click **Create Block Design** in the **IP Integrator** flow navigator.
   The Create Block Design dialog opens.
2. Specify a design name for the block diagram and click **OK**.
3. Click **Next**.

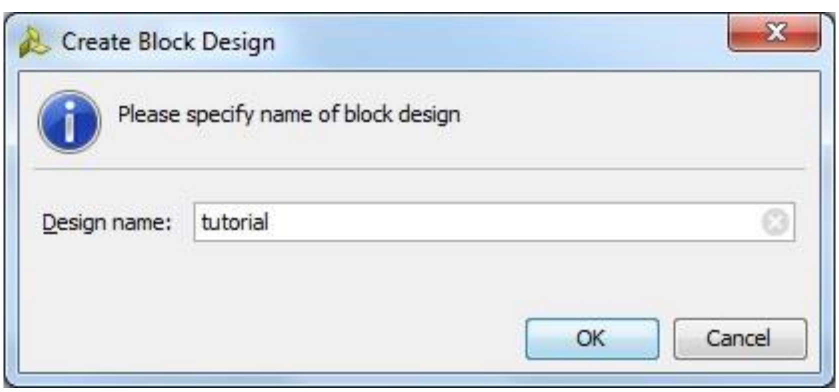


Figure 6 – Create Block Design

Notice that Vivado IP Integrator provides design assistance:

Figure 7 – Designer Assistance – Add IP

Add the ZYNQ7 Processing System core to the block design.

4. Click **Add IP** in the design assistance
   or
   Right-click in the block design, then select **Add IP**.
5. Select the **ZYNQ7 Processing System** core,
   then click **ENTER** (or double-click selection)

Notice that Vivado IP Integrator provides additional design assistance:

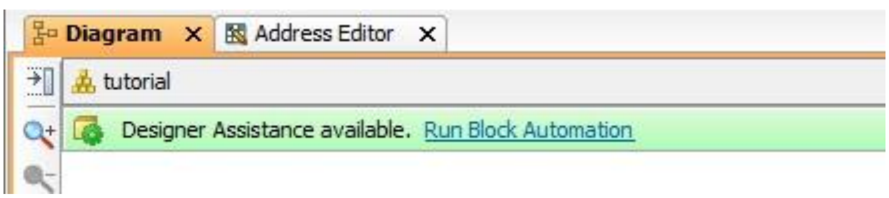


Figure 8 – Designer Assistance – Run Block Automation

Configure the ZYNQ7 Processing System core using the designer assistance:

6. Click **Run Block Automation** in the design assistance
7. Select **/processing_system7_0**
8. Make sure the **Apply Board Preset** option is selected.
9. Click **OK**.

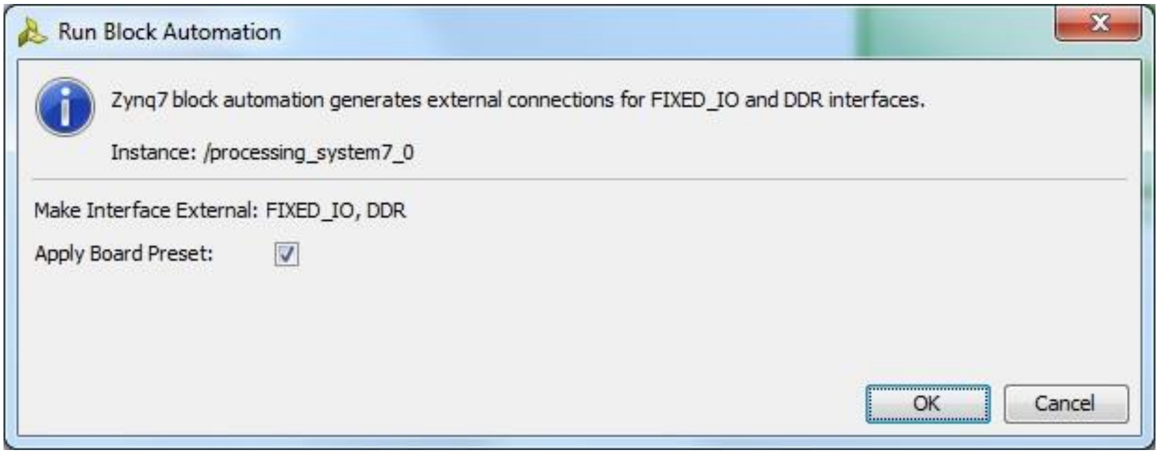


Figure 9 –Run Block Automation

Add the I2C Controller

Add the **AXI IIC** IP core to the design, which will be used to configure the I2C peripherals on the FMC module.

1. **Right-click** in a blank portion of the block design, then select **Add IP**.
2. Select the **AXI IIC** core,
   then click **ENTER** (or double-click selection)

Rename the IIC controller to fmc_imageon_iic_0.

3. Select the **AXI IIC** core in the block diagram.
4. In the **Block Properties** dialog,
   modify the **Name** of the block to **fmc_imageon_iic_0**

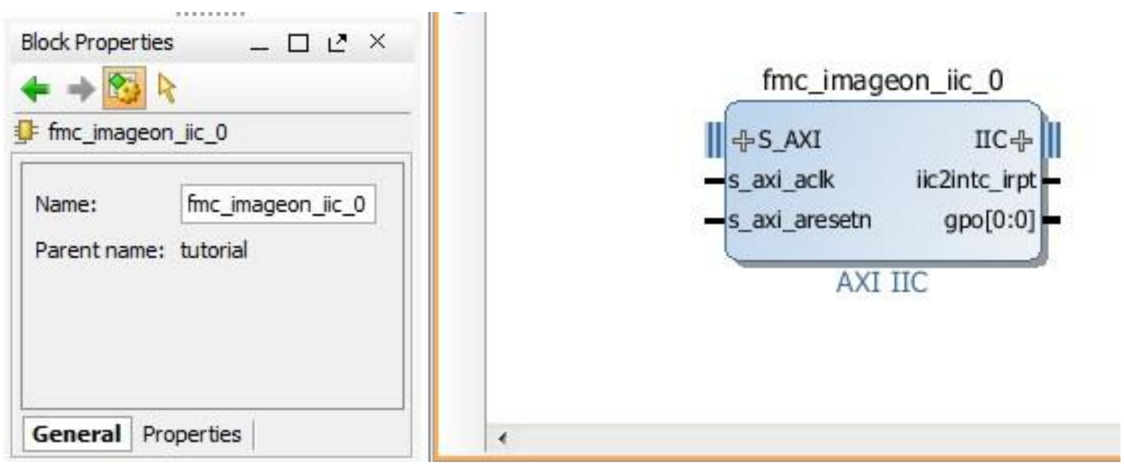


Figure 10 – AXI IIC - Block Properties

Use the designer assistance to connect the AXI IIC core to the ZYNQ7 Processing System's GP0 port.

5. Click **Run Connection Automation** in the design assistance
6. Select **/fmc_imageon_iic_0/S_AXI**
7. Click **OK**.

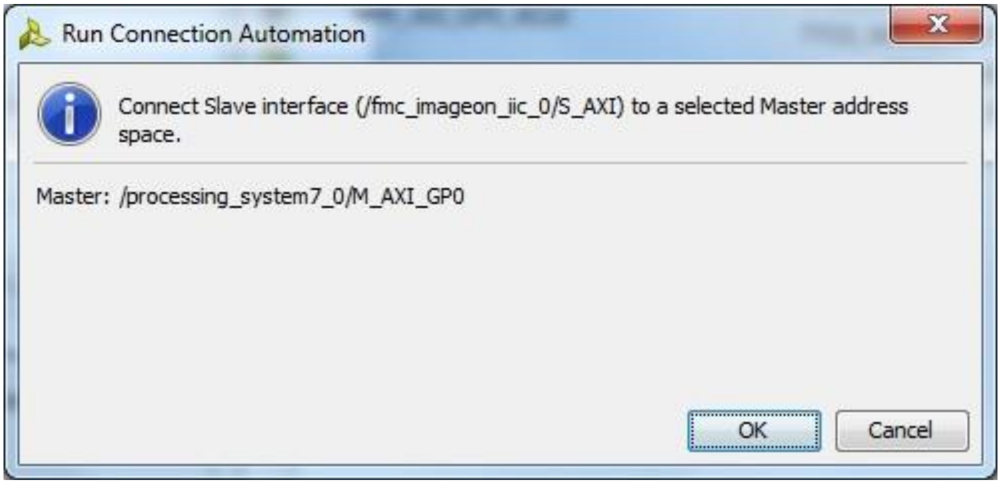


Figure 11 – Run Connection Automation – /fmc_imageon_iic_0/S_AXI

Use the designer assistance to connect the AXI IIC core's external I/O

8. Click **Run Connection Automation** in the design assistance
9. Select **/fmc_imageon_iic_0/IIC**
10. [If the **Select Board Interface** drop down list appears, specify **Custom**]
11. Click **OK**.

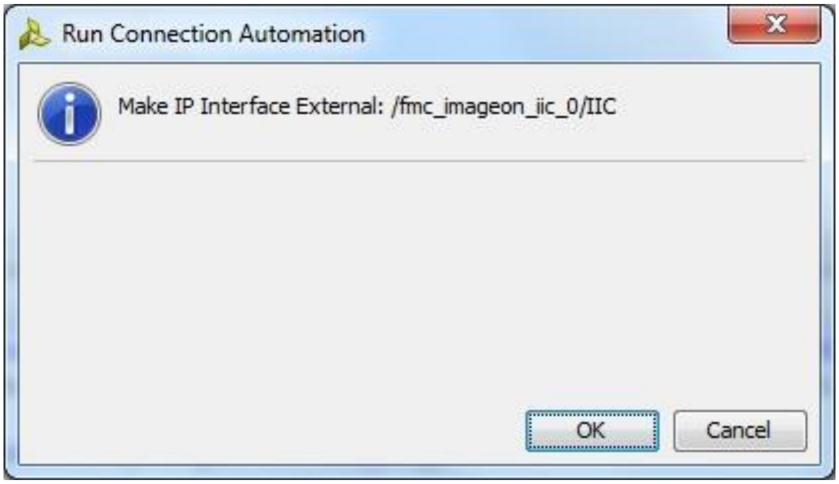


Figure 12 – Run Connection Automation – /fmc_imageon_iic_0/IIC

Rename the external iic_rtl port to fmc_imageon_iic

12. Select the **iic_rtl** external port in the block diagram.
13. In the **External Interface Properties** dialog,
    modify the **Name** of the block to **fmc_imageon_iic**

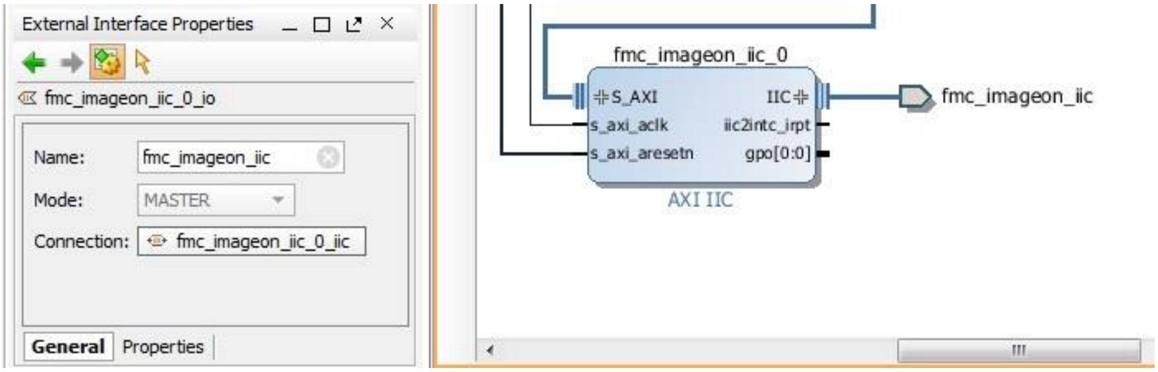


Figure 13 – External Interface Properties – fmc_imageon_iic

Connect the AXI IIC core's gpo[0:0] port to an external port

14. Select the **gpo[0:0]** port on the AXI IIC core
15. **Right-click**, then select **Make External**

Rename the external gpo[0:0] port to fmc_imageon_iic_rst_n

16. Select the **gpo[0:0]** external port in the block diagram.
17. In the **External Interface Properties** dialog,
    modify the **Name** of the block to **fmc_imageon_iic_rst_n**

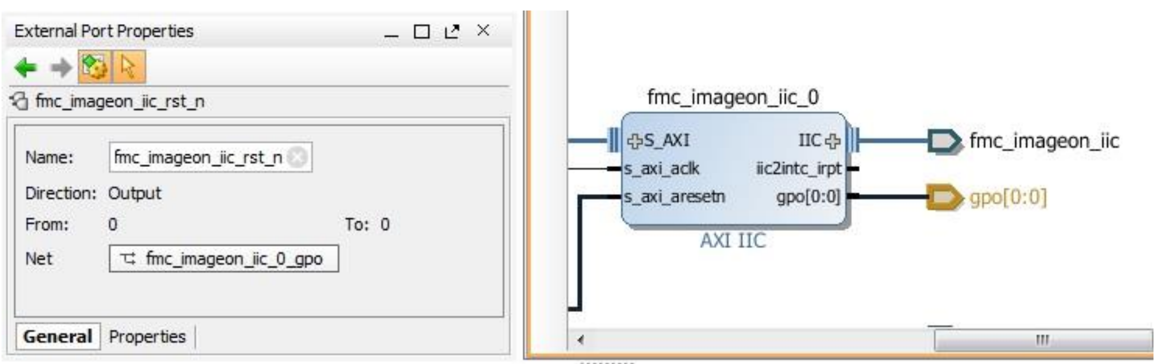


Figure 14 – External Interface Properties – fmc_imageon_iic_rst_n

At this point, validate that you have a correct design.

18. **Right-click** in a blank portion of the block design, then select **Validate Design**

19. If successful, Click **OK** and proceed with the next section.
    Otherwise, fix any errors in the design before proceeding.

Figure 15 – Validate Design

Add the Video Pipeline

The video pipeline will make use of Avnet provided IP cores. In order for Vivado Design Suite to recognize these cores, we need to add the location of the Avnet FMC-IMAGEON cores to the repository path.

1. In the Vivado menu, select **Tools** => **Project Settings**.
   The Project Settings dialog opens.
2. Click on the IP icon on the left.
3. In the **Repository Manager** tab, click the **Add Repository** button.
4. Specify the "**C:\FMC_IMAGEON\2013_3\avnet_fmc_imageon_cores**" directory,
   then click the **Select** button.

The following three IP cores should be detected and appear in the **IP in Selected Repository** window:

- FMC-IMAGEON - HDMI Input
- FMC-IMAGEON - HDMI Output
- FMC-IMAGEON - VITA Receiver

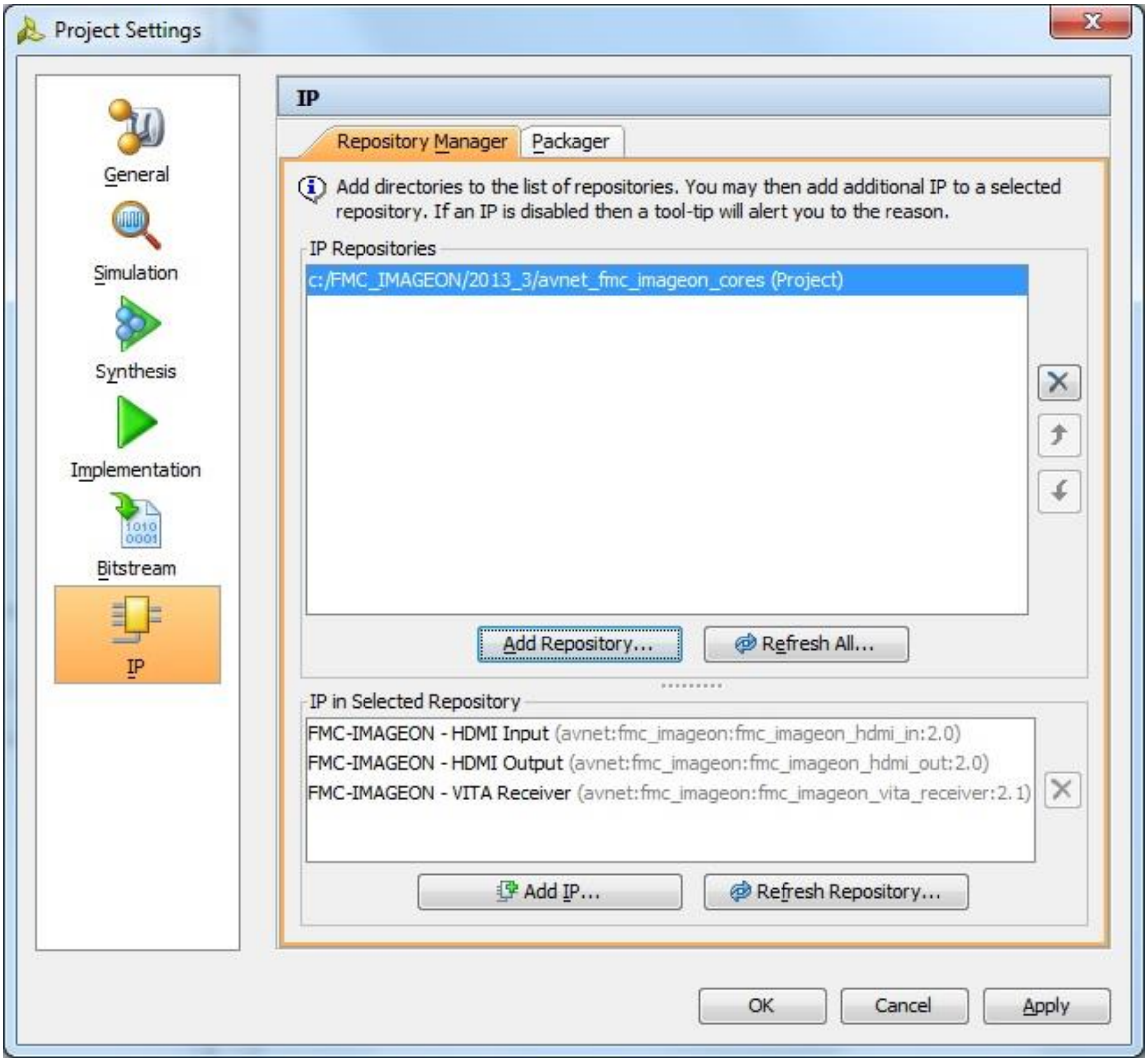


Figure 16 – IP Repository Manager

5. Click **OK**.

The VITA receiver and HDMI output interfaces will be created with TCL scripts.

If you have a color image sensor, run the following two TCL scripts:

- **fmc_imageon_vita_color.tcl** : video input path, 24 bits RGB format
- **fmc_imageon_hdmio_rgb.tcl** : video output path, 24 bits RGB format

If you have a monochrome image sensor, run the following two TCL scripts:

- **fmc_imageon_vita_mono.tcl** : video input path, 8 bits raw Y (intensity) format
- **fmc_imageon_hdmio_y.tcl** : video output path, 8 bits bits Y (intensity) format

Use the TCL console to run the two TCL scripts you have chosen. This tutorial will create the hardware design for the color image sensor, making use the 24 bits RGB video format, but the tutorial instructions can also be applied to create a design for a monochrome image sensor.

6. In the **Tcl Console**, type the following command to generate the video input path
   source C:/FMC_IMAGEON/2013_3/scripts/fmc_imageon_vita_color.tcl
   then press {ENTER}
7. In the **Tcl Console**, press {UP ARROW} to repeat the previous command,
   edit the command to generate the video output path.
   source C:/FMC_IMAGEON/2013_3/scripts/fmc_imageon_hdmio_rgb.tcl
   then press {ENTER}

This will create two new video sub-modules in your block diagram.

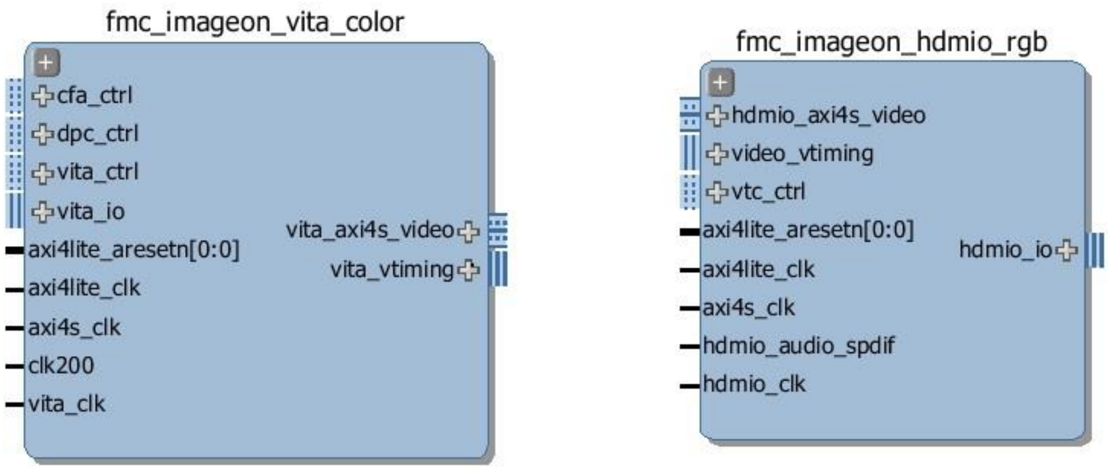


Figure 17 – VITA receiver / HDMI output sub-modules

Connect the AXI4-Lite clocks to the ZYNQ7 Processing System's FCLK_CLK0 port.

1. Select the fmc_imageon_vita_color sub-module's **axi4lite_clk** port
2. Click and hold the left mouse button, then drag to the ZYNQ7's **FCLK_CLK0** port.
3. Release the mouse button to make the connection

4. Select the fmc_imageon_hdmio_rgb sub-module's **axi4lite_ clk** port
5. Click and hold the left mouse button, then drag to the ZYNQ7's **FCLK_CLK0** port.
6. Release the mouse button to make the connection

Connect the AXI4-Lite resets to the proc_sys_reset core's peripheral_aresetn port.

7. Select the fmc_imageon_vita_color sub-module's **axi4lite_aresetn** port
8. Click and hold the left mouse button, then drag to the proc_sys_reset core's **peripheral_aresetn[0:0]** port.
9. Release the mouse button to make the connection

10. Select the fmc_imageon_hdmio_rgb sub-module's **axi4lite_aresetn** port
11. Click and hold the left mouse button, then drag to the proc_sys_reset core's **peripheral_aresetn[0:0]** port.
12. Release the mouse button to make the connection

Use the designer assistance to connect the VITA core's control port (vita_ctrl) and the VTC core's control port (vtc_ctrl) to the ZYNQ7 Processing System's GP0 port.

13. Click **Run Connection Automation** in the design assistance
14. Select **/fmc_imageon_vita_color/vita_ctrl**
15. Click **OK**.

16. Click **Run Connection Automation** in the design assistance
17. Select **/fmc_imageon_hdmio_rgb/vtc_ctrl**
18. Click **OK**.

Use the designer assistance to connect the DPC core's control port (dpc_ctrl) and the CFA core's control port (cfa_ctrl) to the ZYNQ7 Processing System's GP0 port.

19. Click **Run Connection Automation** in the design assistance
20. Select **/fmc_imageon_vita_color/dpc_ctrl**
21. Click **OK**.

22. Click **Run Connection Automation** in the design assistance
23. Select **/fmc_imageon_vita_color/cfa_ctrl**
24. Click **OK**.

Configure the ZYNQ7 Processing System to generate a 150MHz clock on its FCLK_CLK1 port, which will be used for the AXI4-Stream based interconnect and cores. Also, generate a 200MHz clock on its FCLK_CLK2 port, which will be used by the VITA receiver core for its IDELAY primitives.

25. Double-click on the ZYNQ7 Processing System core.
26. Click on the **Clock Generation** block
27. Expand the **PL Fabric Clocks** section
28. Select the **FCLK_CLK1** clock
29. Specify a **Requested Frequency** of **150MHz**.
30. Select the **FCLK_CLK2** clock
31. Specify a **Requested Frequency** of **200MHz**.
32. Click **OK**.

Connect the 200MHz clock to the ZYNQ7 Processing System's FCLK_CLK2 port.

33. Select the fmc_imageon_vita_color sub-module's **clk200** port
34. Click and hold the left mouse button, then drag to the ZYNQ7's **FCLK_CLK2** port.
35. Release the mouse button to make the connection.

Connect the AXI4-Stream clocks to the ZYNQ7 Processing System's FCLK_CLK1 port.

36. Select the fmc_imageon_vita_color sub-module's **axi4s_clk** port
37. Click and hold the left mouse button, then drag to the ZYNQ7's **FCLK_CLK1** port.
38. Release the mouse button to make the connection.

39. Select the fmc_imageon_hdmio_rgb sub-module's **axi4s_clk** port
40. Click and hold the left mouse button, , then drag to the ZYNQ7's **FCLK_CLK1** port.
41. Release the mouse button to make the connection.

Create an external port for the VITA input clock in the block diagram.

42. **Select** the fmc_imageon_vita_color sub-module's **vita_clk** port
43. **Right-click**, then select **Make External**

Connect the HDMI output clock to the VITA input clock.

44. Select the fmc_imageon_hdmio_rgb sub-module's **hdmio_clk** port
45. Click and hold the left mouse button, then drag to the **vita_clk** net.
46. Release the mouse button to make the connection.

Connect the VITA and HDMI I/O ports (vita_io & hdmio_io) to external ports

47. **Select** the fmc_imageon_vita_color sub-module's **vita_io** port
48. **Right-click**, then select **Make External**

49. **Select** the fmc_imageon_hdmio_rgb sub-module's **hdmio_io** port
50. **Right-click**, then select **Make External**

Connect the VITA input and HDMI output sub-modules together, implementing a video pass-through.

51. **Select** the fmc_imageon_vita_color sub-module's **vita_axi4s_video** port
52. Click and hold the left mouse button,
    then drag to the fmc_imageon_hdmio_rgb sub-module's **hdmio_axi4s_video**
53. Release the mouse button to make the connection.

54. **Select** the fmc_imageon_vita_color sub-module's **vita_vtiming** port
55. Click and hold the left mouse button,
    then drag to the fmc_imageon_hdmio_rgb sub-module's **video_vtiming**
56. Release the mouse button to make the connection.

At this point, validate that you have a correct design.

57. **Right-click** in a blank portion of the block design, then select **Validate Design**

58. If successful, Click **OK** and proceed with the next section.
    Otherwise, fix any errors in the design before proceeding.

Figure 18 – Validate Design

Save the block diagram

59. In the menu, select **File** => **Save Block Design**

Build the hardware with Vivado Design Suite

Create the top level HDL file.

1. In the **Sources** tab, select the block diagram
2. **Right-click**, then select **Create HDL Wrapper**

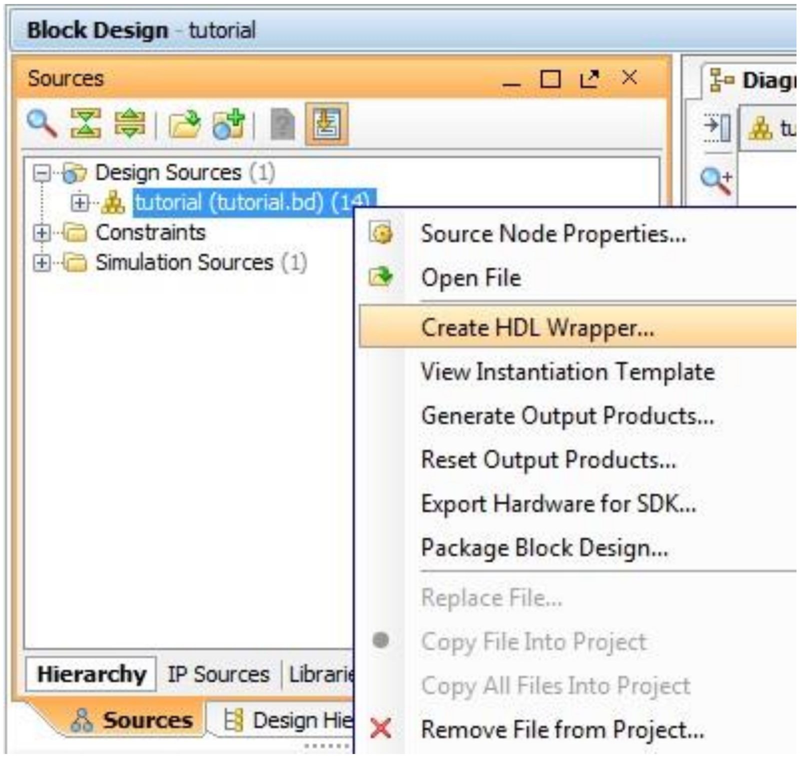


Figure 19 – Create HDL Wrapper

3. When asked whether to add or copy the HDL wrapper, select **Let Vivado manage wrapper and auto-update**.

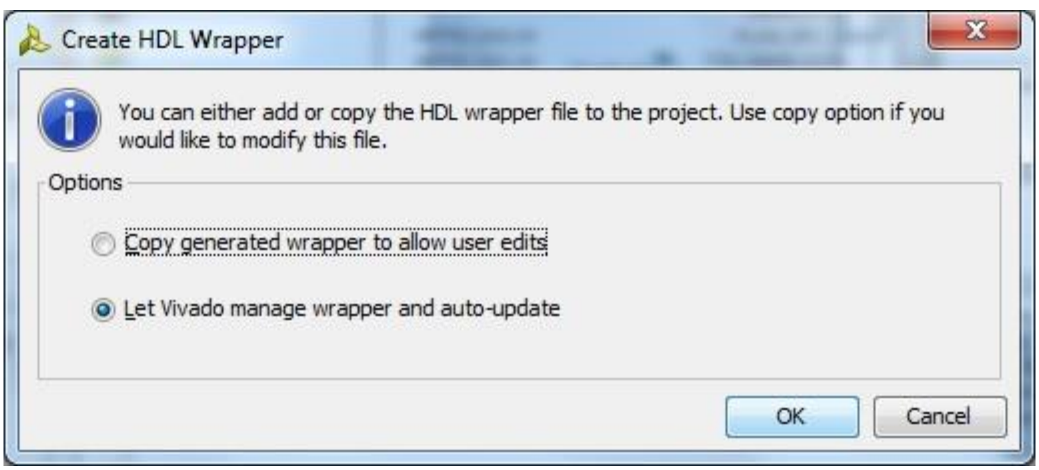


Figure 20 – Let Vivado manage wrapper and auto-update

Add the XDC constraints file.

4. In the Flow Navigator, under the Project Manager section, click **Add Sources**.
5. Select the **Add or Create Constraints** option
6. Click **Next**
7. In the dialog box that opens, click the **Add Files …** button to add an existing XDC file
8. Select one of the following XDC files, depending on your hardware:
    a. For the ZC702:
        ..\constraints\zc702_fmc_imageon_vita_passthrough.xdc
    b. For the ZedBoard:
        ..\constraints\zedboard_fmc_imageon_vita_passthrough.xdc

    Once selected, click **OK**
9. Click **Finish**.

Build the bitstream.

10. In the Flow Navigator, under the Program and Debug section, click **Generate Bitstream**. A dialog box appears asking whether all the processes starting for synthesis should be done.
11. Click **Yes**.

The “Bitstream Generation Completed” dialog box will open, asking what to do Next.

12. Select **Open Implemented Design**
13. Click **OK**.

The resource utilization and power estimation for this design can be seen in the Project Summary. Note that results may be different depending on the video format chosen, and for different carriers.

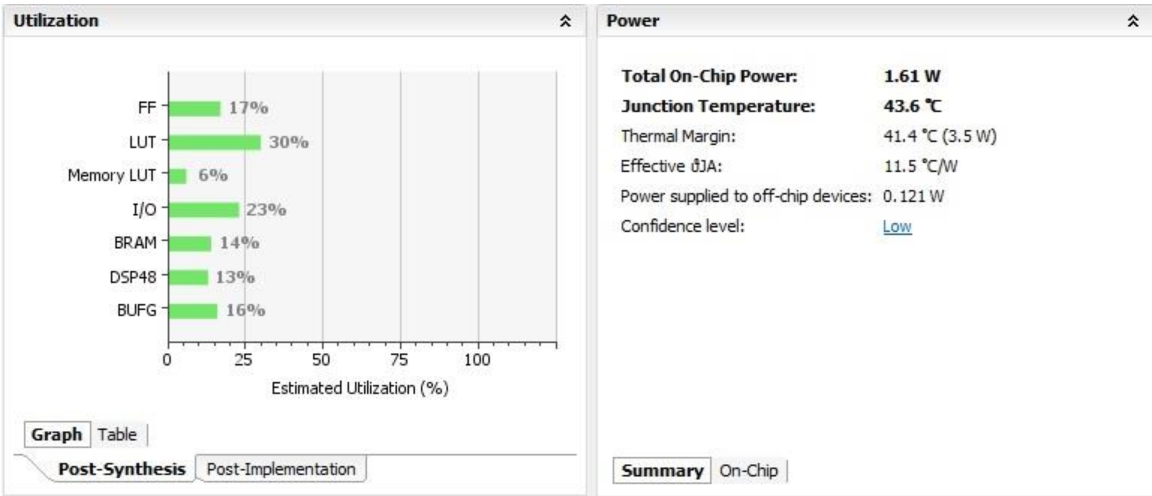


Figure 21 – VITA Pass-Through – Resource Utilization

You have successfully created the hardware design !

Create the Embedded Software Application with SDK

Launch SDK from Vivado Design Suite.

1. In the Vivado menu, Select **File > Export > Export Hardware for SDK**.
   The "Export Hardware for SDK" dialog box opens.
   By default, the "Include Bitstream" and "Export Hardware" check boxes are checked.
2. Check the **Launch SDK** check box.
3. Click **OK**. SDK opens.

Notice that when SDK launched, the hardware description file was automatically read in. The system.xml tab shows the address map for the entire Processing System.

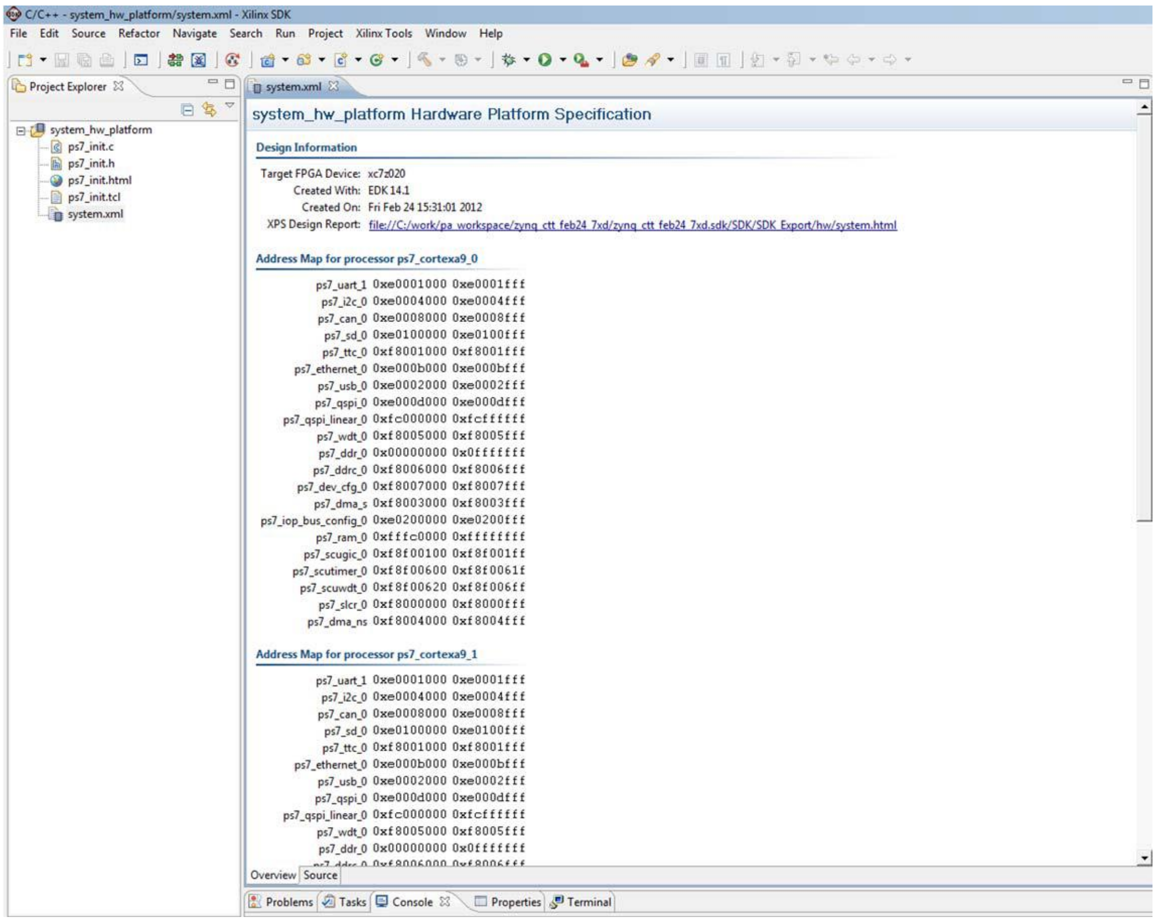


Figure 22 – Address Map in SDK system.xml Tab

The **avnet_fmc_imageon_cores** directory contains some software drivers that we will use in this design. In order for the project to recognize the contents of this directory, the path must be added to the project repositories, as described below.

4. In the SDK menu, select **Xilinx Tools** => **Repositories**
   The Preferences dialog box opens.

5. In the Local Repositories section, click on the **New …** button.
6. Select the **C:\FMC_IMAGEON\2013_3\avnet_fmc_imageon_cores** directory, then click **OK**
7. Click **OK** in the Preferences dialog box.

Create a standalone BSP (board support package).

8. In the SDK menu, select **File** => **New** => **Xilinx Board Support Package**.
   The New Board Support Package Project dialog box opens.
9. In the **Project name** field, type “**vita_passthrough_bsp**”.
10. Keep the default settings, and click **Finish**.
    The Board Support Package Settings dialog box opens.
11. In the Supported Libraries, select the `fmc_iic_sw` and `fmc_imageon_sw` libraries.

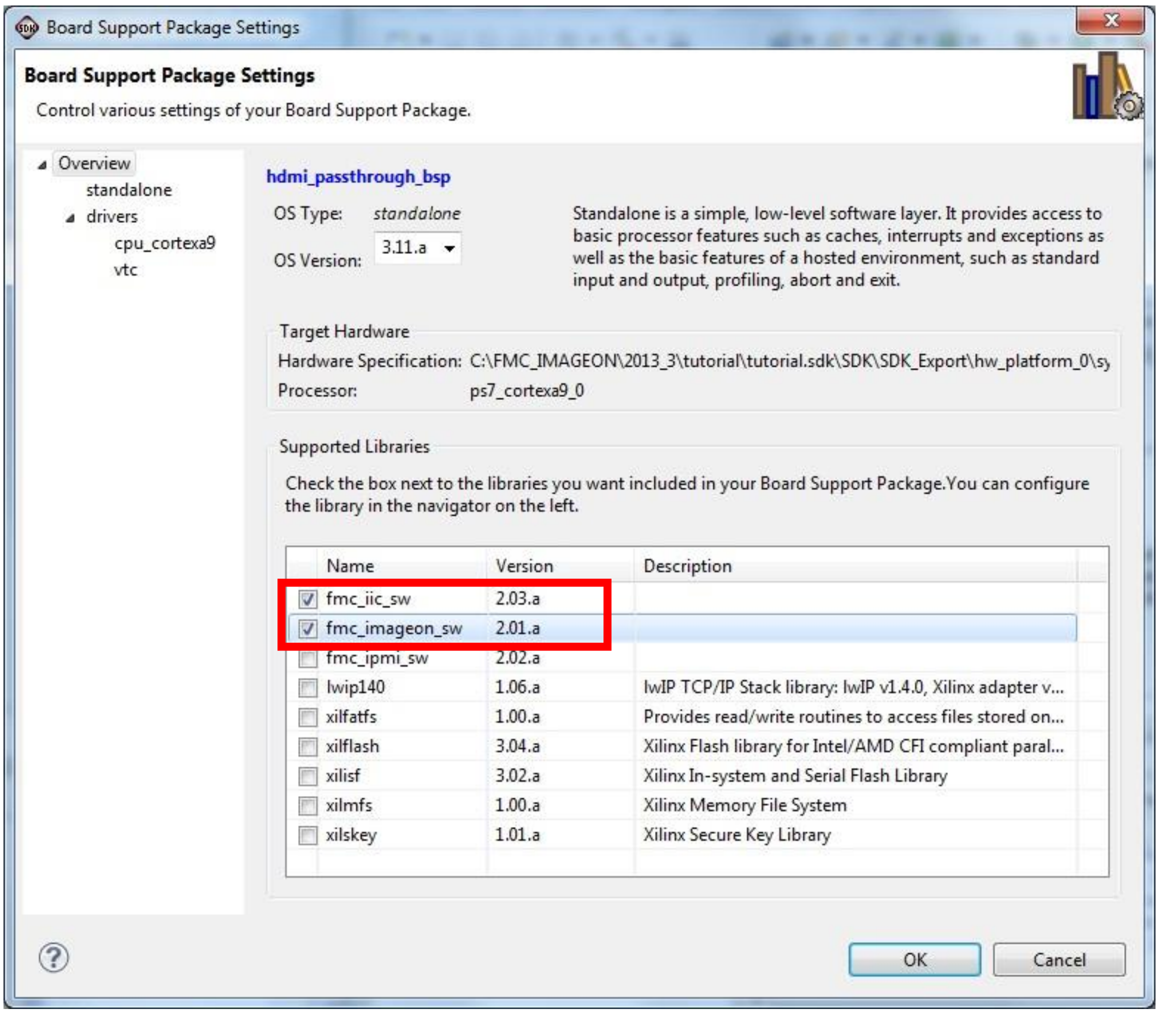


Figure 23 – Board Support Package Settings

12. Click **OK**.
    If the Build Automatically setting is enabled, SDK will automatically build the standalone BSP.

Create a new C project.

13. In the SDK menu, select **File** => **New** => **Application Project**.
    The Application Project dialog box opens.
14. In the **Project Name** field, type "**vita_passthrough_app**".
15. For the **Board Support Package**, select **Use Existing**, then select the BSP that was created previously.
16. Click **Next**.
    The Templates dialog box opens.
17. Select the **Hello World** template.
18. Click **Finish**.

Configure the application's memory map to execute from external memory.

19. Right-click on the **vita_passthrough_app** application
20. Select **Generate Linker Script**
    This opens the Generate a linker script dialog box.

21. Select the **ps7_ddr_0** memory for each of the Code, Data, Heap and Stack sections.

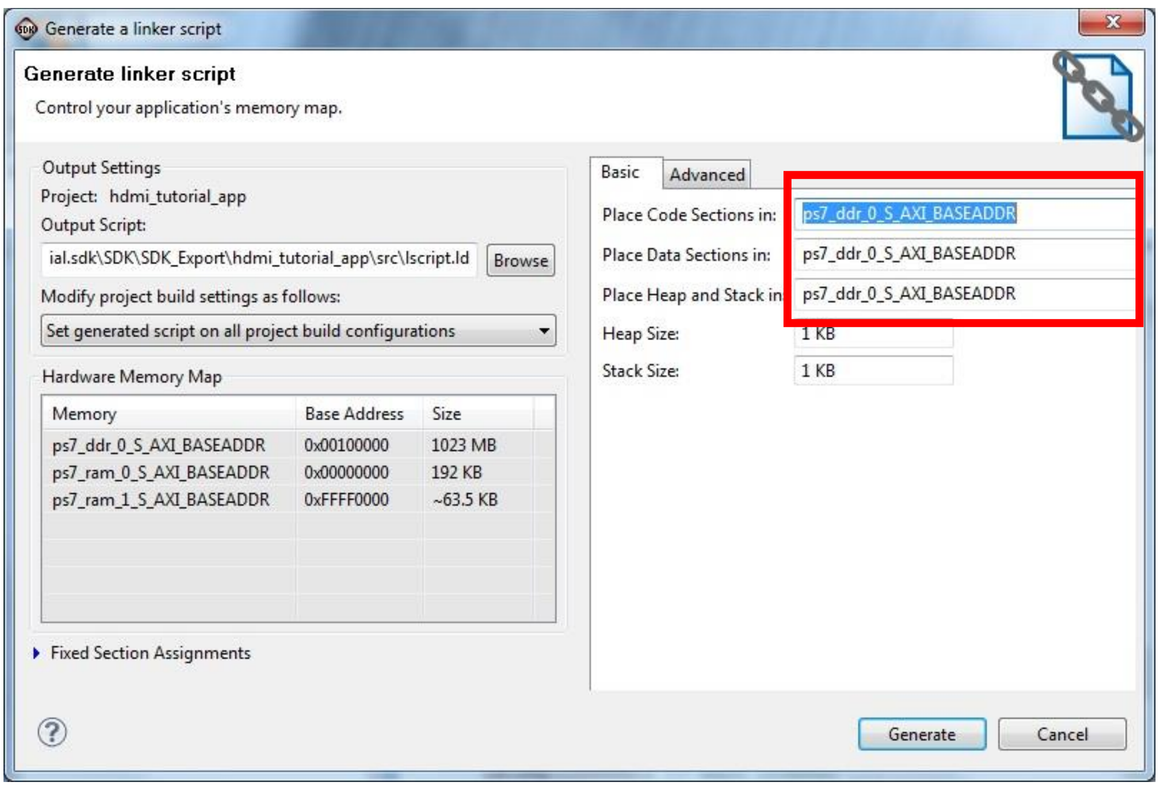


Figure 24 – Generate a Linker Script

22. Click **Generate**.
    A dialog box appears asking whether it is OK to overwrite the existing linker script file.
23. Click **Yes**

Import the provided example source files for the vita_passthrough

1. In the Project Explorer window, select the **vita_passthrough_app** application
2. Right-click, then select **Import** from the pop-up menu.
   The Import wizard appears.
3. Expand the **General** section
4. Select **File System**, then click **Next**.

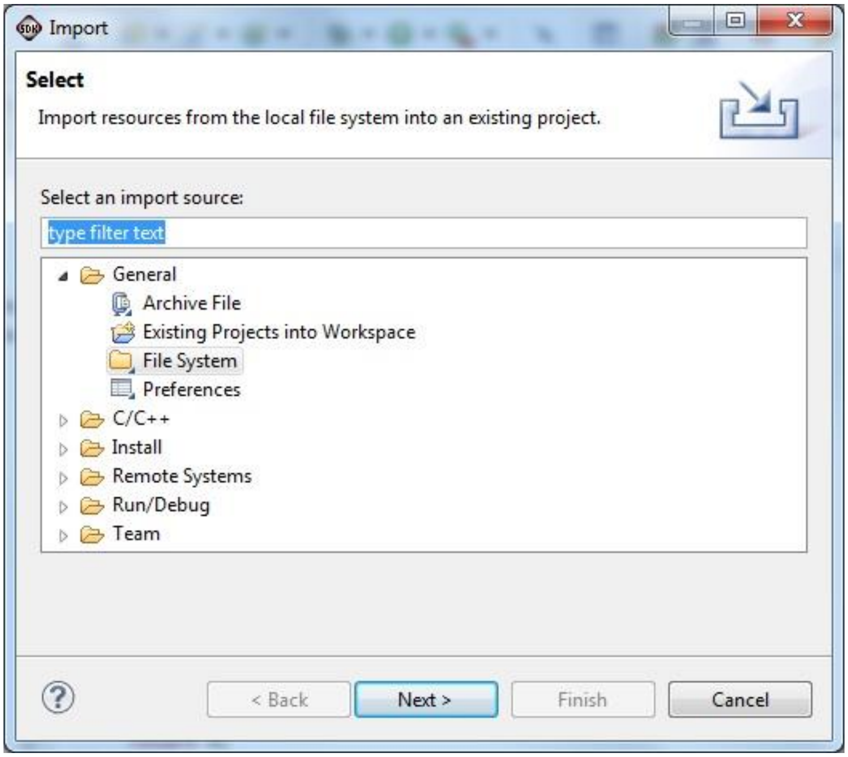


Figure 25 – Import from File System – Dialog 1

The next dialog of the Import wizard appears.

5. Next to the **From directory** field, click the **Browse** button
6. Specify the following directory:
   C:\FMC_IMAGEON\2013_3\code\fmc_imageon_vita_passthrough
7. Click **OK**

8. Select the following source files:
   fmc_imageon_vita_passthrough.c
   fmc_imageon_vita_passthrough.h
   sleep.c
   sleep.h
   video_resolution.c

video_resolution.h
video_generation.c
video_generation.h

9. Next to the **Into directory** field, click the **Browse** button
10. Specify the following directory : **vita_passthrough_app\src**, then click **OK**

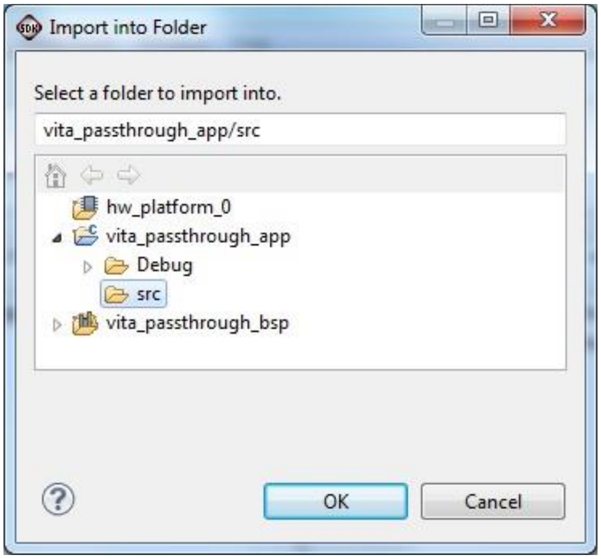


Figure 26 – Import into Folder

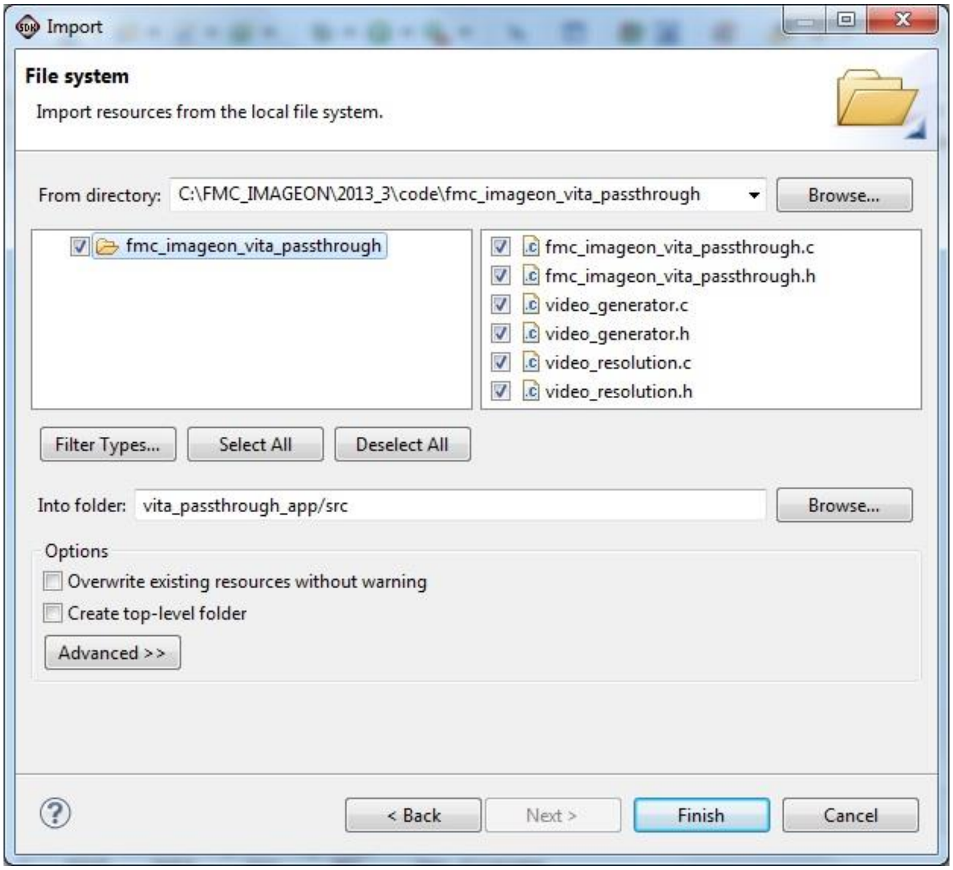


Figure 27 – Import from File System – Dialog 2

11. Click **Finish**.

Modify the hello world application

12. Open the helloworld.c file and edit the source code as follows:

```
/*
 * helloworld.c: simple test application
 */

#include <stdio.h>
#include "platform.h"

#include "fmc_imageon_vita_passthrough.h"
fmc_imageon_vita_passthrough_t demo;

//void print(char *str);
void print( const char *str);

int main()
{
    init_platform();
```

strange bug:
when vtc driver is active, need to modify print declaration to match the one in xil_printf.h

```
    print("Hello World\n\r");

    demo.uBaseAddr_IIC_FmcImageon = XPAR_FMC_IMAGEON_IIC_0_BASEADDR;
    demo.uBaseAddr_VITA_Receiver  =
        XPAR_FMC_IMAGEON_VITA_COLOR_FMC_IMAGEON_VITA_RECEIVER_0_S00_AXI_BASEADDR;
    demo.uDeviceId_VTC_HdmioGenerator =
        XPAR_FMC_IMAGEON_HDMIO_RGB_V_TC_0_DEVICE_ID;
    demo.uBaseAddr_DPC = XPAR_DPC_0_BASEADDR;
    demo.uBaseAddr_CFA = XPAR_CFA_0_BASEADDR;
    fmc_imageon_vita_passthrough_init( &demo );

    cleanup_platform();

    return 0;
}
```

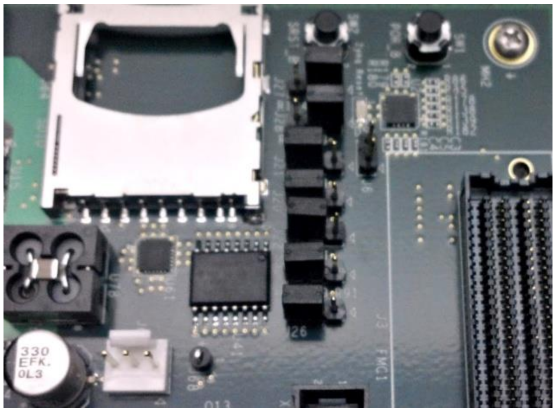
13. If the Build Automatically setting is enabled, SDK will automatically build the application. If not, right-click on the application and select **Build Project** to build the application.

You have successfully created the software application !

Set up your ZC702 Hardware

Setup your ZC7020-based hardware, as described below.

1. Set the ZC702 board's boot mode to cascaded JTAG using jumpers
    a. J21,J20,J22,J25,J26 should all be set to '0'
    b. J27,J28 should be set to '1'



2. Connect a mini USB cable to the ZC702's USB-UART connector (J17)
3. Connect one of the following JTAG connections:
    a. Connect platform USB pod to the ZC702's JTAG header (J2) and set SW10 to '10'
    b. Connect a micro USB cable to the ZC702's on-board Digilent JTAG module and set SW10 to '01'
4. Populate the FMC-IMAGEON board on FMC Slot #2.
5. Connect the VITA-2000 camera to the FMC module's CON200 connector
6. Connect a DVI or HDMI monitor to the FMC module's HDMI OUT connector
7. Power on the ZC702 board

8. Open a serial communication utility for the COM port assigned on your system.
   Note: The standard configuration for Zynq Processing System is baud rate 115200, 8 bit, parity

Set up your ZedBoard Hardware

Setup your ZedBoard hardware, as described below.

1. Set the ZedBoard's boot mode to cascaded JTAG using jumpers
   a. JP7, JP8, JP9, JP10, JP11 should all be set to '0'

2. Connect a micro USB cable to the ZedBoard's USB-UART connector (J14)
3. Connect one of the following JTAG connections:
   a. Connect platform USB pod to the ZedBoard's JTAG header (J15)
   b. Connect a micro USB cable to the ZedBoard's on-board Digilent JTAG connector (J17)
4. Populate the FMC-IMAGEON board on FMC Slot #1.
5. Connect the VITA-2000 camera to the FMC module's CON200 connector
6. Connect a DVI or HDMI monitor to the fMC module's HDMI OUT connector
7. Power on the ZedBoard board
8. Open a serial communication utility for the COM port assigned on your system.
   Note: The standard configuration for Zynq Processing System is baud rate 115200, 8 bit, parity

Execute the VITA Pass-Through Design on Hardware using SDK

From SDK, configure the FPGA bitstream and launch the application.

1. In the SDK menu, select **Xilinx Tools** => **Program FPGA**
   The "Program FPGA" dialog opens.
2. Make sure the path to the bitstream is valid
   (HINT : If you moved the project, you will need to update the path to the bitstream file)
3. Click **Program**.
   It will take approximately 10 seconds to program the bitstream to hardware
4. Right-click **vita_passthrough_app**
   and select **Run as > Run Configurations**
5. Click **Xilinx C/C++ ELF** and click **New launch configurations**.

6. The new run configuration is created named **vita_passthrough_app Debug**.
   The configurations associated with application are pre-populated in the main tab of these launch configurations.
7. Click the **Device Initialization** tab in the launch configurations and check the settings here.
   Notice that there is a configuration path to the initialization TCL file. The path of `ps7_init.tcl` is mentioned here. This is file that was exported when you exported your design to SDK; it contains the initialization information for the processing system.
   (HINT : If you moved the project, you should delete the previous run configuration and create a new one)
8. The STDIO Connection tab is available in the launch configurations settings. You can use this to have your STDIO connected to the console. We will not use this now because we have already launched a serial communication utility. There are more options in launch configurations but we will focus on them later.
9. Click **Apply** and then **Run**.
10. If you get a Reset Status dialog box indicating that the current launch will reset the entire system, click **OK**.
11. You should see something similar to the following on your serial console:

```
Hello World

------------------------------------------------------
--          FMC-IMAGEON VITA Pass-Through           --
------------------------------------------------------

FMC-IMAGEON Initialization ...
Video Clock Synthesizer Configuration ...
ADV7511 Video Output Information
        Video Output     = DVI, 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
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
Video Timing Controller (generator) Initialization ...
        Video Resolution = 1080P
Defect Pixel Correction (DPC) Initialization ...
        DPC done
Color Filter Array Interpolation (CFA) Initialization ...
        CFA done

Done


Press ENTER to re-start ...
```

To re-start the detection of the HDMI input source, press ENTER.

You have successfully executed the HDMI pass-through on hardware !

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 HDMI Input/Output FMC Module
   http://www.em.avnet.com/fmc-imageon ➔ Support Files & Downloads
2. Avnet FMC-IMAGEON – Hardware User Guide
   http://www.em.avnet.com/fmc-imageon ➔ Support Files & Downloads
3. 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

The following reference provides links to documentation for video intellectual property (IP).

4. Video and Image Processing IP
   http://www.xilinx.com/ipcenter/video/video_core_listing.htm
5. Video Timing Controller
   http://www.xilinx.com/products/intellectual-property/EF-DI-VID-TIMING.htm
6. Video Input to AXI4-Stream
   http://www.xilinx.com/products/intellectual-property/video_in_to_axi4_stream.htm
7. AXI4-Stream to Video Output
   http://www.xilinx.com/products/intellectual-property/axi4_stream_to_video_out.htm
8. AXI Video DMA
   http://www.xilinx.com/products/intellectual-property/axi_video_dma.htm

The following reference provides links to documentation for AXI interconnect.

9. UG761 - AXI Reference Guide
10. PG065 – AXI4-Stream Infrastructure

Known Issues and Limitations

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

Hello World Template – error: conflicting types for ‘print’

The “Hello World” C project template has an issue that may manifest itself depending on which drivers are included in the design.

The “print(….)” declaration does not match the declaration in the xil_printf.h file and will result in the following error:

```
helloworld.c:29:6: error: conflicting types for ‘print’
xil_printf.h:39:6: note: previous declaration of ‘print’ was here
```

The solution is to simply fix the “print(…)” declaration as shown below.

```
//void print(char *str);
void print( const char *ptr);
```

Troubleshooting