FMC-IMAGEON
Building a Video Design from Scratch Tutorial

Version 1.3

Revision History

Version	Description	Date
1.0	HDMI Pass-Through + AXI4-Stream tutorial - 14.3 version	Nov. 12, 2012
1.1	Updated fmc_imageon_sw software library - fix issue with 720P60 resolution on Rev.B h/w	Nov. 30, 2012
1.2	Add a Video Frame Buffer	Dec. 06, 2012
1.3	14.4 version	Jan. 03, 2013

Table of Contents

Table of Figures

Table of Tables

About this Guide

This tutorial describes how to create an EDK-based Video Design from scratch..

This manual contains the following chapters:

- Chapter "**Introduction**" 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 "**Step 1) Implement an HDMI Pass-Through**" describes the steps required to implement an Embedded System that implements an HDMI pass-through.
- Chapter "**Step 2) Bridge to/from the AXI4-Stream Interface**" describes the steps required to bridge to/from the AXI4-Stream interface.
- Chapter "**Step 2d) Debug AXI4-Stream Interface (Optional)**" describes how to use ChipScope to debug the AXI4-Stream interface.
- Chapter "**Step 3) Add a Video Frame Buffer**" describes the steps required to implement a video frame buffer in external memory.
- 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.

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-v2000-c

Once on the product website:

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

To access the commiunity forums, visit 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 how the connectivity of the FMC module.

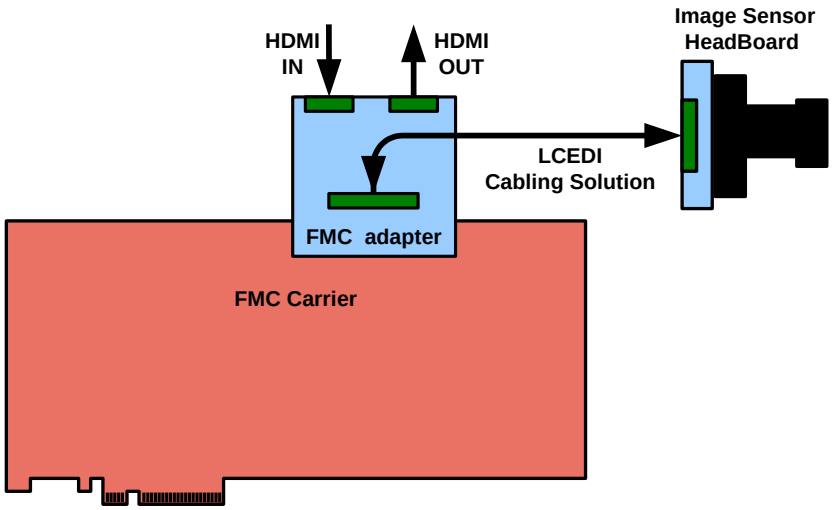


Figure 2 – FMC-IMAGEON Hardware – Connectivity Diagram

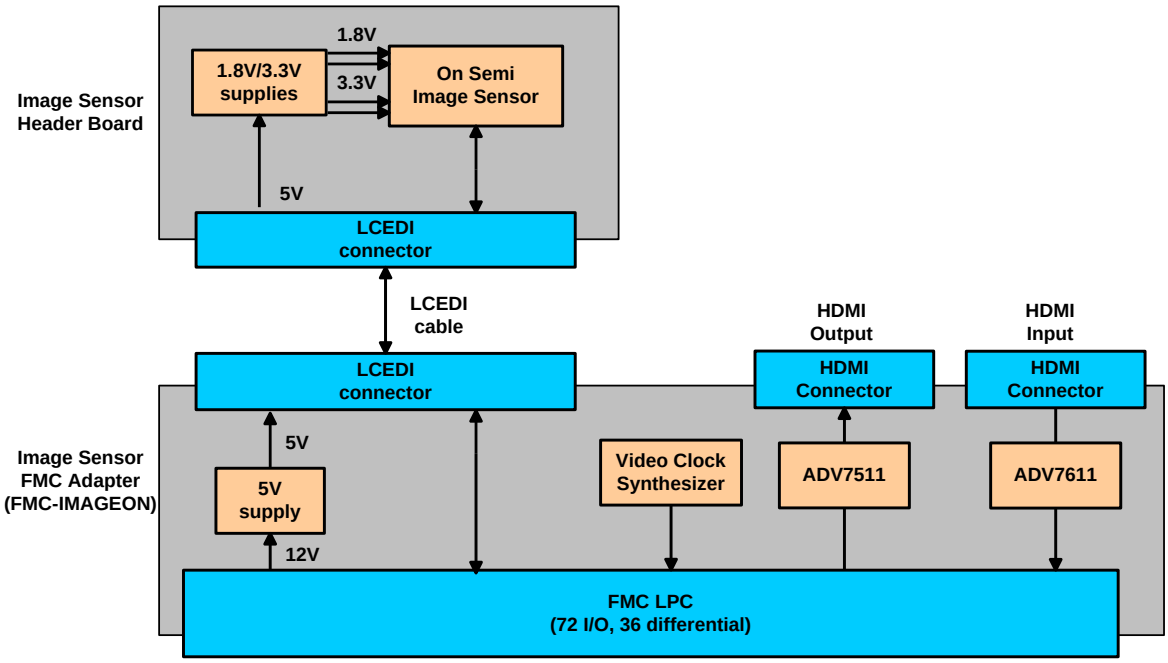


Figure 3 – FMC-IMAGEON Hardware – Block Diagram

Requirements

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

Software

The software required to use and run the demonstrations is:

- Xilinx ISE Design Suite 14.4 : 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]
- One of the following FMC baseboards
    - 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
- HDMI source (non-encrypted content) or DVI-D source, including HDMI cable

Zynq embedded design tools

This tutorial assumes that you have experience creating Zynq embedded designs. More specifically, it assumes a working knowledge of PlanAhead and the embedded tools (XPS, SDK).

If you do not have this experience, it is HIGHLY recommended to get familiar with the Zynq Concept, Tools, and Techniques (CTT) literature.

For ZC702:

Zynq Concepts, Tools, and Techniques
http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_3/ug873-zynq-ctt.pdf

For ZedBoard:

Zynq Concepts, Tools, and Techniques on ZedBoard
http://www.zedboard.org/design

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

Xilinx Video IP

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

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_Tutorial\board
C:\FMC_IMAGEON_Tutorial\code
C:\FMC_IMAGEON_Tutorial\repository

This tutorial requires the repository that was included in the archive. This repository contains the following IP:

PCOREs
..\repository\ProcessorIPLib\pcores**fmc_imageon_hdmi_in_v2_01_a**
..\repository\ProcessorIPLib\pcores**fmc_imageon_hdmi_out_v2_02_a**

Software Libraries:
..\repository\ProcessorIPLib\sw_services**fmc_iic_sw_v2_03_a**
..\repository\ProcessorIPLib\sw_services**fmc_imageon_sw_v1_07_a**

Test your DVI-D or HDMI equipment

Before going through this tutorial, test your DVI-D and/or HDMI equipment:

1. Connect your DVI-D or HDMI source to your DVI-D or HDMI monitor
2. Verify that you can see the video source on your monitor
3. Using the menu settings on your DVI-D or HDMI monitor to validate the video resolution of your DVI-D or HDMI source. Make sure that it is generating one of the following supported video resolutions

Resolution	Pixel Rate (MHz)	Frame Dimensions	Frame Rate
1080P60	148.5 MHz	1920 x 1080	60 Hz
SXGA	110 MHz	1280 x 1024	60 Hz
720P60	74.25 MHz	1280 x 720	60 Hz
XGA	65 MHz	1024 x 768	60 Hz
SVGA	40 MHz	800 x 600	60 Hz
576P50	27 MHz	720 x 576	50 Hz
480P60	27 MHz	720 x 480	60 Hz
VGA	25.175 MHz	640 x 480	60 Hz

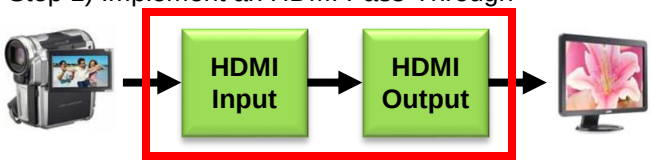
Table 1 – Supported Video Resolutions

Tutorial Overview

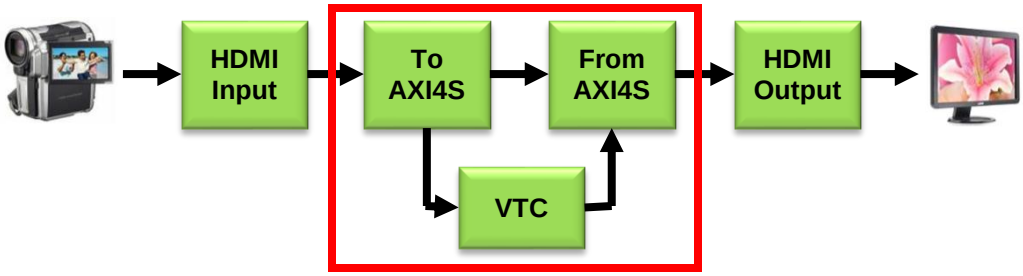
This tutorial will guide you in creating a video design based on the Avnet HDMI Input/Output FMC module from scratch.

The tutorial will build the design in several incremental steps:

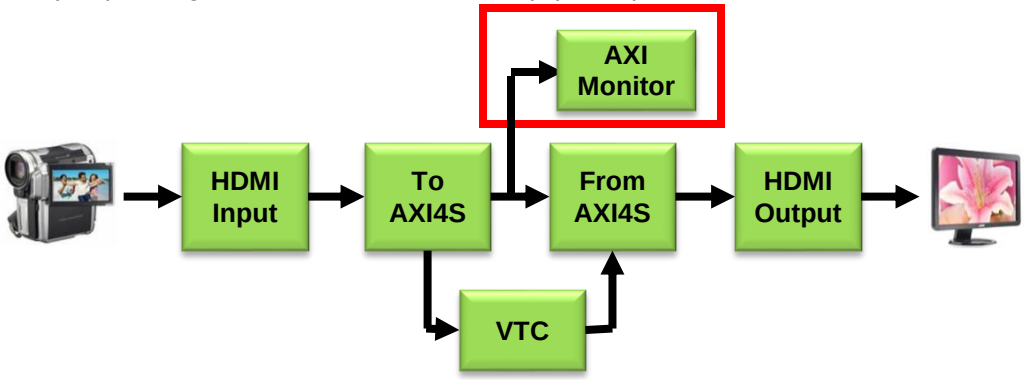
- Step 1) Implement an HDMI Pass-Through



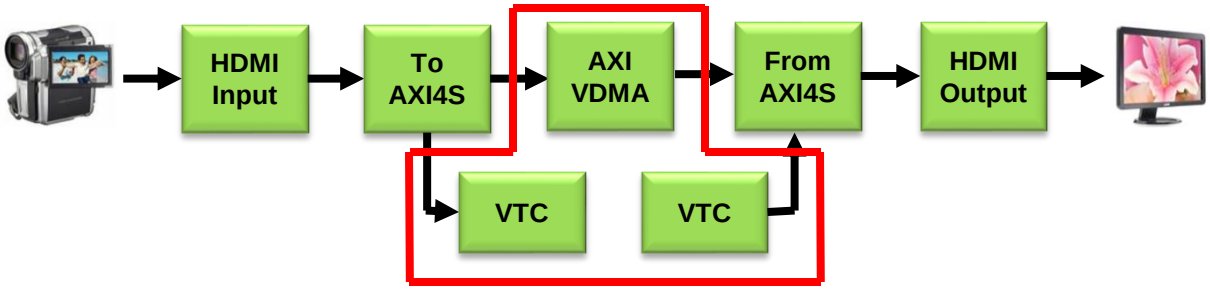
- Step 2) Bridge to/from the AXI4-Stream Interface



- Step 2d) Debug the AXI4-Stream Interface (Optional)



- Step 3) Add a Video Frame Buffer



Step 1) Implement an HDMI Pass-Through

In this section, a new PlanAhead project will be created, implementing a very simple HDMI pass-through design for the FMC-IMAGEON hardware.

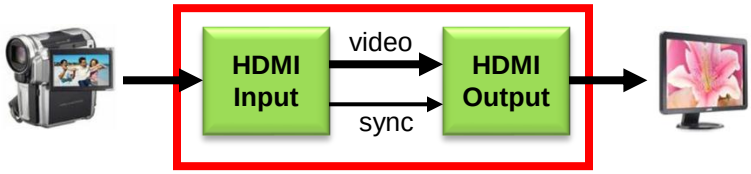


Figure 4 – HDMI Pass-Through – Block Diagram

The processor sub-system is configured, and 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 (used in step 3)

The following cores, provided with the tutorial, will be used to interface to the ADV7611 and ADV7511 devices on the FMC-IMAGEON module. It is important to note that, on the FMC-IMAGEON module, these devices are used in 16 bits YCbCr 4:2:2 with “embedded sync” mode.

- FMC-IMAGEON HDMI Input
  - this core contains logic that will extract the embedded synchronization signals from the 16 bit video data received from the ADV7611 HDMI input device.
- 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.

This entire video pipeline will be running on the HDMI input interface’s video clock.

Create a new PlanAhead project

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

1. Start the PlanAhead 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_Tutorial**
	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 2 – New Project Settings

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

IMPORTANT: *The Design Runs module at the bottom of the PlanAhead interface has a Strategy column. Review this column to verify that the values are the XST Default (XST 14) and ISE Default (ISE 14). If these do not show the correct values, correct them in the Synthesis Settings and Implementation Settings*.

Create the Embedded Hardware Design with EDK

Use the Add Sources wizard to create an embedded processor project.

1. Click **Add Sources** in the Project Manager.
   The Add Sources wizard opens.
2. Select the **Add or Create Embedded Sources** option.
3. Click **Next**.

4. In the “Add or Create Embedded Source” window, click **Create Sub-Design**.
5. Type a name for the module and click **OK**.
   The module you created displays in the sources list.
6. Click **Finish**.
   Platform Studio will open a dialog box indicating that the project is blank, and asking if you want to create a Base System using the BSB Wizard.
7. Click **Yes**.
8. The “Create New XPS Project Using BSB Wizard” dialog opens.
9. For the **Peripheral Repository Search path**, click the **Browse** button, then specify the **C:\FMC_IMAGEON_Tutorial\repository** directory
10. Click **OK**
11. In the “Board and System Selection” window,
    Keep the default settings
12. Click **Next**.
13. In the “Peripherals” window,
    Keep the default settings
14. Click **Finish**.

The XPS System Assembly View opens with the Zynq tab displayed.

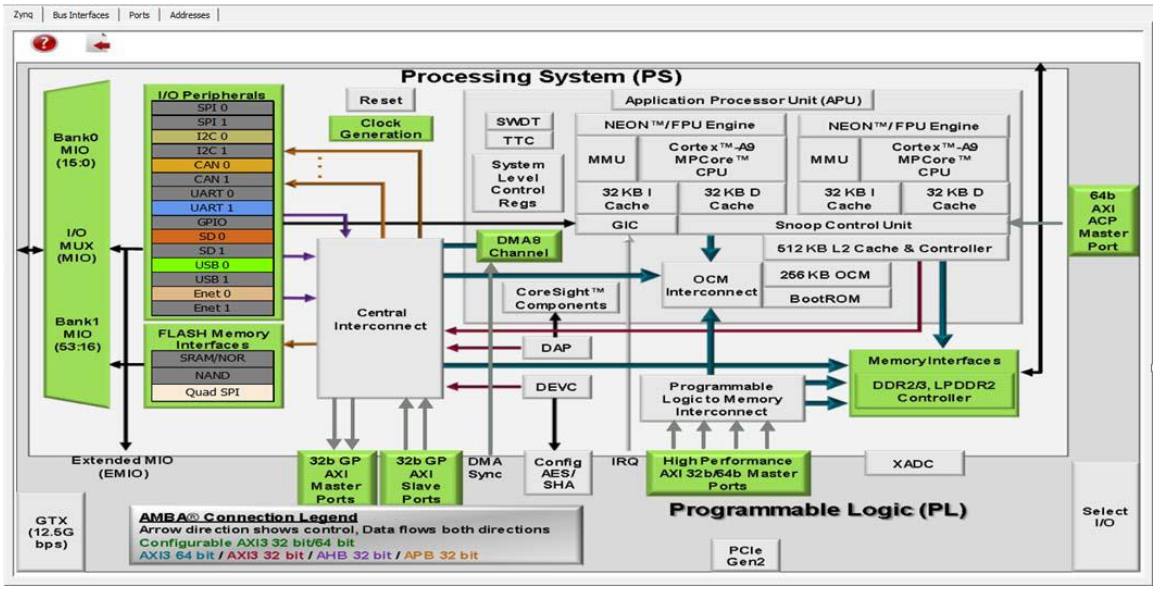


Figure 5 – Zynq Processing System

Notice that several I/O peripherals were selected (color coded) for the ZC702 or ZedBoard.

Add the axi_iic PCORE to the design, which will be used to configure the I2C peripherals on the FMC module.

15. In the System Assembly View, click the **Bus Interfaces** tab.
16. From the IP catalog, expand **EDK Install** => **Communication Low-Speed** and double-click on **AXI IIC Interface** (1.02.a) to add it to the design.

A message appears asking if you want to add the axi_iic 1.02.a IP instance to your design.

17. Click **Yes**.
    The configuration window for **AXI IIC Interface** opens.
18. Change the **Component Instance Name** to **fmc_imageon_iic_0**
    Notice Output Frequency of SCL Signal is set to 100,000 (ie. 100KHz). This is correct.
    Leave all other parameters as they are.
19. Click **OK**.
    A message window opens with the message "axi_iic IP with version number 1.02.a is instantiated with name fmc_imageon_iic_0". It will ask you to determine to which processor to connect.
    Remember you are designing with a dual core ARM processor. The message also says XPS will make the Bus Interface Connection, assign the address, and make IO ports external.
    The default choice of processor is "processing_system7_0". Do not change this.
20. 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.

Add the fmc_imageon_hdmi_in PCORE to the design.

21. From the IP catalog, expand **Project Peripheral Repository2** => **FMC-IMAGEON** and double-click on **fmc_imageon_hdmi_in** to add it.
    A message appears asking if you want to add the fmc_imageon_hdmi_in 2.01.a IP instance to your design.
22. Click **Yes**.
    The configuration window for **fmc_imageon_hdmi_in** opens.
    Leave all parameters as they are.
23. Click **OK**.

Add the fmc_imageon_hdmi_out PCORE to the design.

24. Now double-click on **fmc_imageon_hdmi_out** to add it to the design
    A message appears asking if you want to add the fmc_imageon_hdmi_out 2.02.a IP instance to your design.
25. Click **Yes**.
    The configuration window for **fmc_imageon_hdmi_out** opens.
    Leave all parameters as they are.
26. Click **OK**.

View the bus interfaces in the design.

27. Click the **Bus Interfaces** tab, which lists the IPs and their bus connections. Expand `fmc_imageon_iic_0`,
28. Notice that the `axi_iic_0` was automatically connected to the `axi4lite_0`.

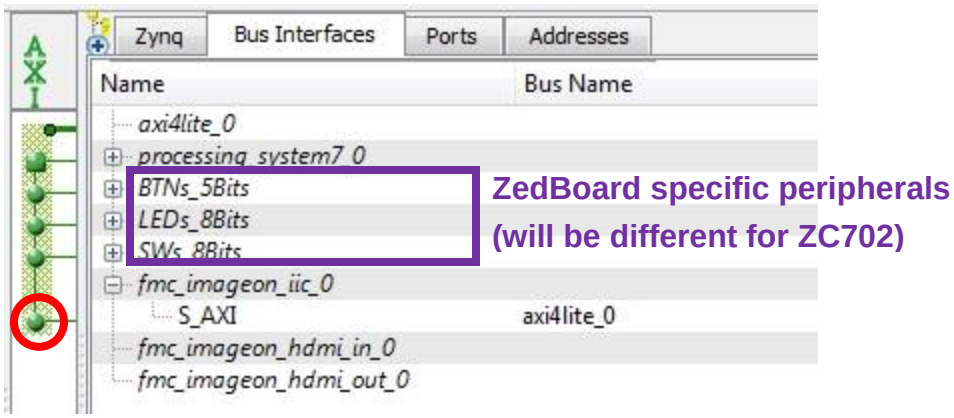


Figure 6 – Bus Interfaces tab with fmc_imageon_iic_0 expanded

Connect the ports in the design.

29. Click the **Ports** tab, which lists the IPs and their ports. Expand `axi4lite_0`, `fmc_imageon_iic_0`, `fmc_imageon_hdmi_in_0`, and `fmc_imageon_hdmi_out_0`.
30. Review the following IP connections. If any of these aren't already connected, connect them now

IP	Port	Connection
axi4lite_0	INTERCONNECT_ACLK	Processing_ps7_0::FCLK_CLK0
	INTERCONNECT_ARESET	Processing_ps7_0::FCLK_RESET0_N
fmc_imageon_iic_0	(BUS_IF) S_AXI::X_AXI_ACLK	Processing_ps7_0::FCLK_CLK0
	(IO_IF) iic_0::Gpo	External Ports:: fmc_imageon_iic_0_Gpo_pin
	(IO_IF) iic_0::Sda	External Ports:: fmc_imageon_iic_0_Sda_pin
	(IO_IF) iic_0::Scl	External Ports:: fmc_imageon_iic_0_Scl_pin

Table 3 – Automatic Ports Connections

The IO_IF ports (Scl, Sda, Gpo) for the fmc_imageon_iic_0 pcore were automatically connected as external pins in the design. Since we will be using the general purpose (Gpo) port as a reset for the FMC's on-board I2C multiplexer, we will rename it accordingly.

31. In the **Ports** tab, expand the `External Ports` section, select the `fmc_imageon_iic_0_Gpo_pin` port, and rename it to `fmc_imageon_iic_0_`**`Reset`**`_pin`.

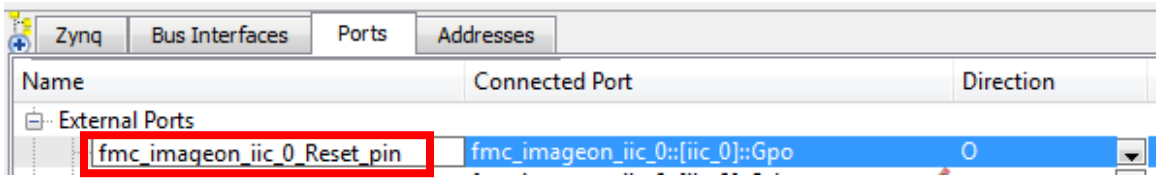


Figure 7 – Renaming the Gpo port to fmc_imageon_iic_0_Reset_pin

The IO_IF ports for the fmc_imageon_hdmi_in_0 and fmc_imageon_hdmi_out_0 pcores were not automatically connected as external pins, so we need to do this manually.

32. In the **Ports** tab, expand the `fmc_imageon_hdmi_in_0` section, select `(IO_IF) HDMII_IO`, then select **Make Ports External** in the popup menu.

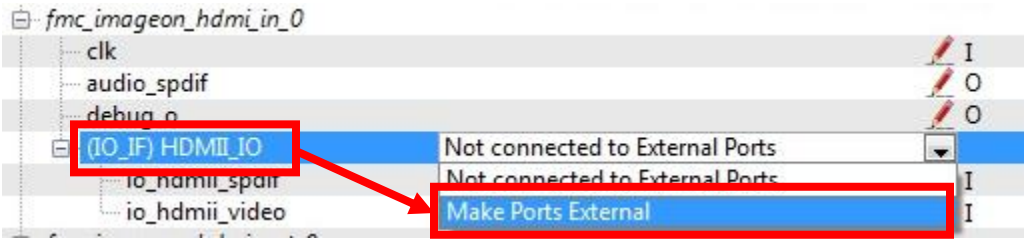


Figure 8 – Make Ports External for fmc_imageon_hdmi_in_0

33. Similarly, expand the `fmc_imageon_hdmi_out_0` section, select `(IO_IF) HDMIO_IO`, then select **Make Ports External** in the popup menu.

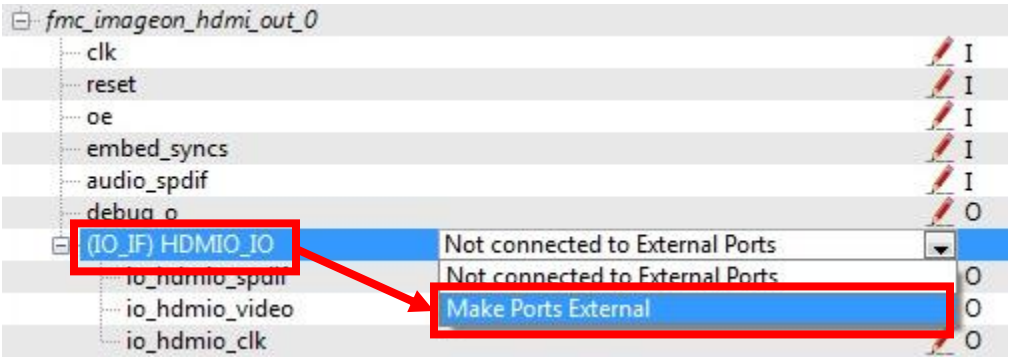


Figure 9 – Make Ports External for fmc_imageon_hdmi_out_0

There are a few more ports that need to be connected manually.

34. Make the following new connections in the Ports tab.

IP	Port	Connection
fmc_imageon_hdmi_in_0	clk	External Ports:: fmc_imageon_hdmi_in_0_clk_pin
	audio_spdif	fmc_imageon_hdmi_out_0::audio_spdif
	video_vblank	fmc_imageon_hdmi_out_0::video_vblank
	video_hblank	fmc_imageon_hdmi_out_0::video_hblank
	video_de	fmc_imageon_hdmi_out_0::video_de
	video_data	fmc_imageon_hdmi_out_0::video_data
fmc_imageon_hdmi_out_0	clk	External Ports:: fmc_imageon_hdmi_in_0_clk_pin

	reset	net_gnd
	oe	net_vcc
	embed_syncs	net_vcc
	audio_spdif	fmc_imageon_hdmi_in_0::audio_spdif
	video_vblank	fmc_imageon_hdmi_in_0::video_vblank
	video_hblank	fmc_imageon_hdmi_in_0::video_hblank
	video_de	fmc_imageon_hdmi_in_0::video_de
	video_data	fmc_imageon_hdmi_in_0::video_data

Table 4 – Ports Connections

Once done, the port connections should match the ones shown in the following figure.

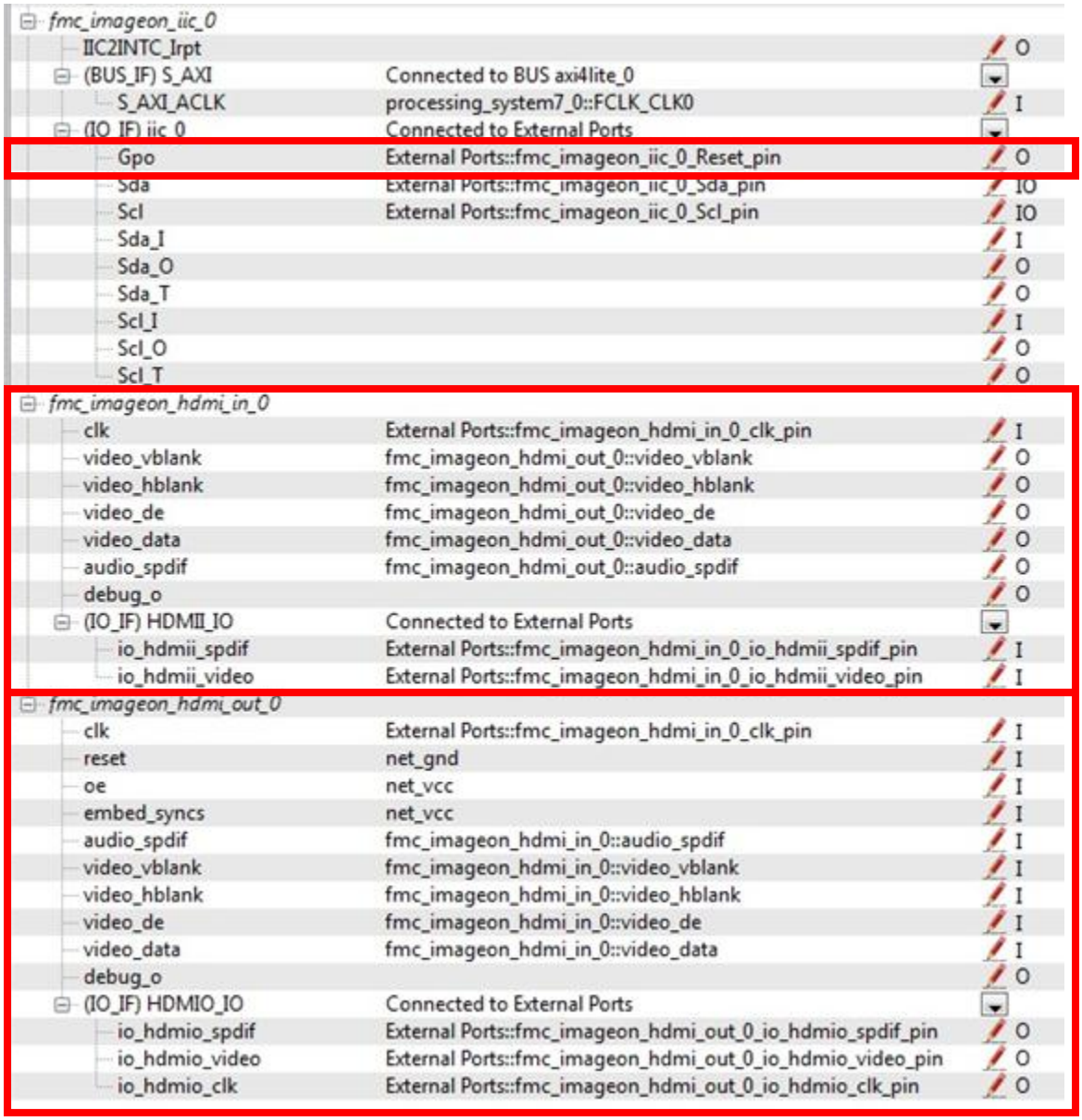


Figure 10 – Completed Ports Connections

35. Collapse all IPs and expand `External Ports`.
36. For the fmc_imageon_hdmii_in_0_clk_pin port, make sure the Class is set to CLK, and set the Frequency (Hz) to 148500000, as shown below

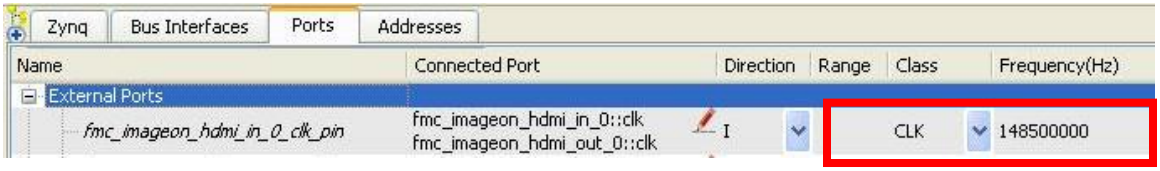


Figure 11 – Configuring fmc_imageon_hdmi_in_0_clk_pin as an external 148.5MHz clock

Perform the Design Rule Check

37. Run Design Rule Check. This can be invoked from the menu by selecting **Project** => **Design Rule Check**
38. Ensure there are no errors in the console.

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

39. Close the XPS. The PlanAhead™ window becomes active again.

Build the hardware with PlanAhead

The active PlanAhead session updates with the project settings.

Create a top level HDL file.

1. In Design Sources, right-click `{module name}.xmp` and select **Create Top HDL**. PlanAhead generates the `{module name}_stub.v` file.

Add the UCF constraints file.

2. Click **Add Sources**.
3. Select the **Add or Create Constraints** option
4. Click **Next**
5. In the dialog box that opens, click the **Add Files …** button to add an existing UCF file
6. Select on of the following UCF files, depending on your hardware:
   a. For the ZC702:
      ..\boards\zc702\zc702_fmc_imageon_hdmi_tutorial.ucf
   b. For the ZedBoard:
      ..\boards\zedboard\zedboard_fmc_imageon_hdmi_tutorial.ucf

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

Build the bitstream.

8. In the Program and Debug list in the Flow Navigator, click **Generate Bitstream**.
   A dialog box appears asking whether all the processes starting for synthesis should be done.
9. Click **Yes**.
   Ignore the following critical warnings that may appear:
   Cannot loc instance 'processing_system7_0_PS_PORB_IBUF' at site B5, …
   Cannot loc instance 'processing_system7_0_PS_SRSTB_IBUF' at site C6, …
   Cannot loc instance 'processing_system7_0_PS_CLK_IBUF' at site F7, …

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

10. Select **Open Implemented Design**
11. Click **OK**.

The resource utilization for this design can be seen in the Project Summary.

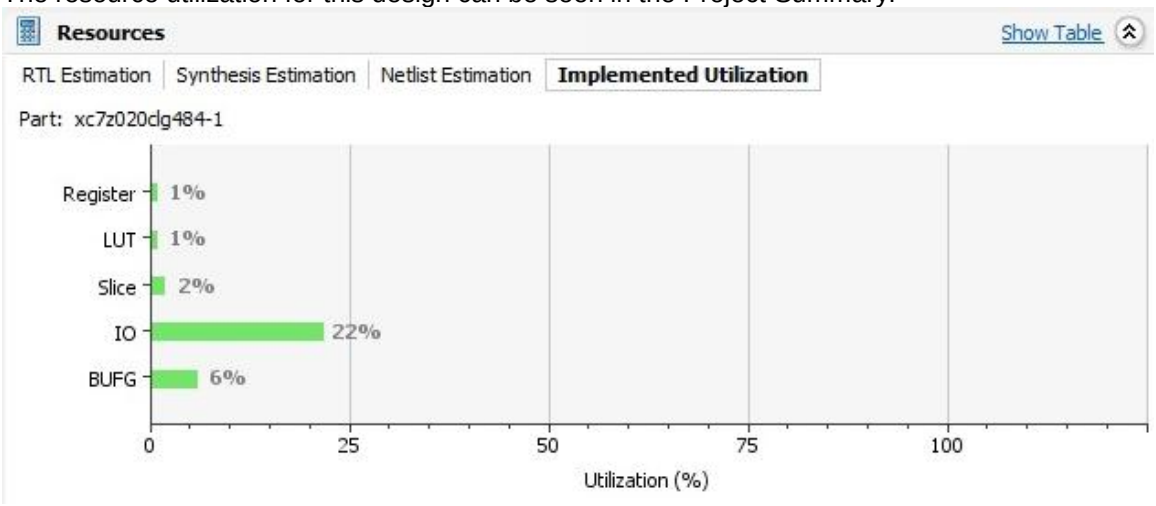


Figure 12 – HDMI Pass-Through – Resource Utilization

You have successfully created the hardware design !

Create the Embedded Software Application with SDK

Launch SDK from the PlanAhead software.

1. In the PlanAhead software, 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.

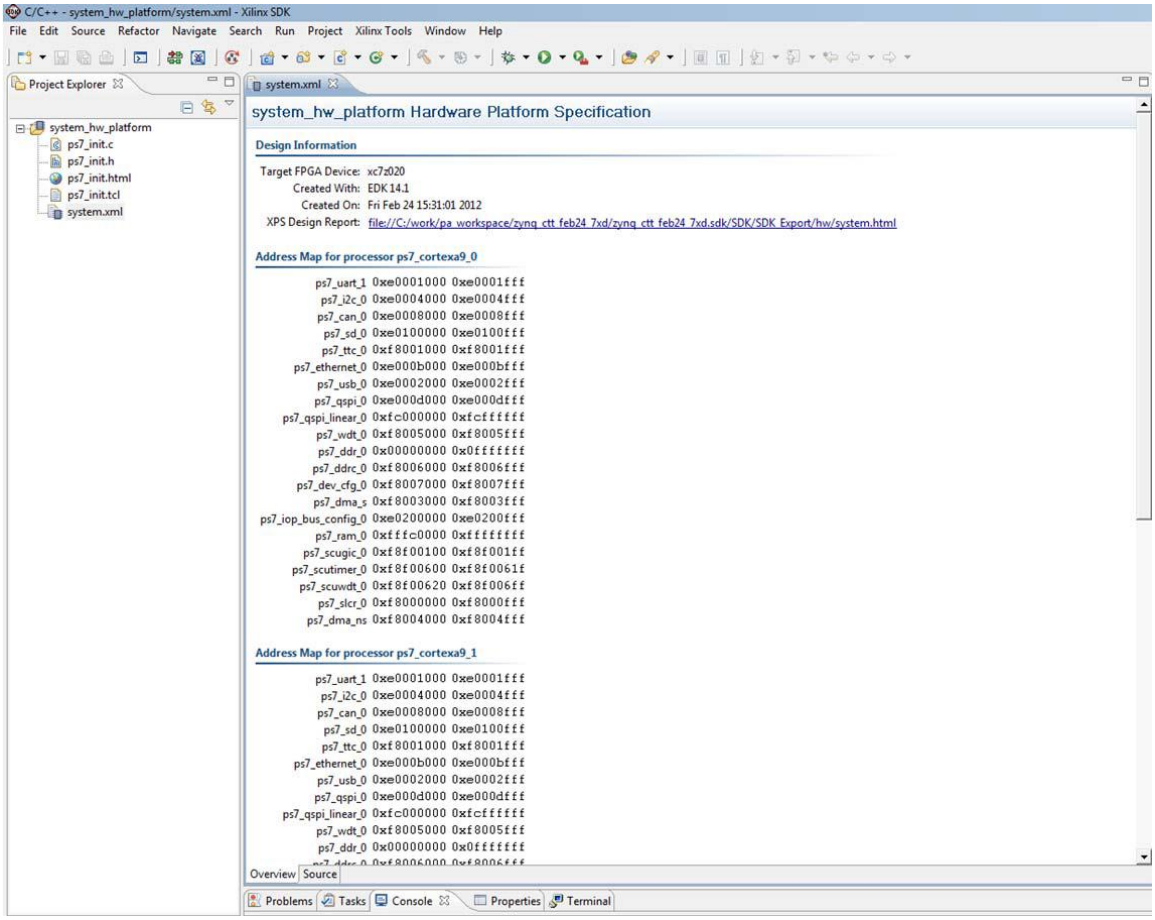


Figure 13 – Address Map in SDK system.xml Tab

The **repository** 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_Tutorial\repository** 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 "**hdmi_tutorial_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.

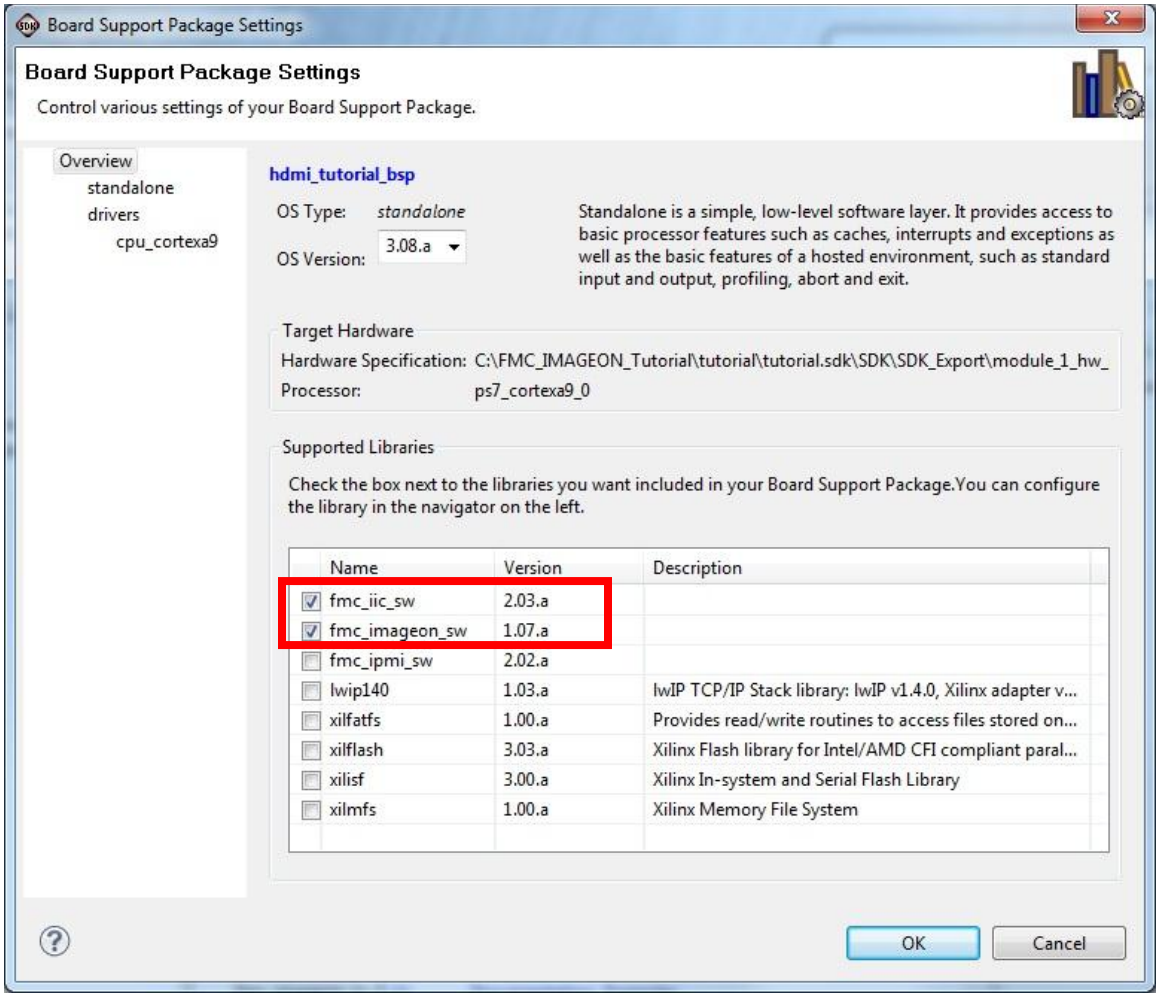


Figure 14 – Board Support Package Settings

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 "**hdmi_tutorial_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 `hdmi_tutorial_app` application, and select **Generate Linker Script**
    This opens the Generate a linker script dialog box.

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

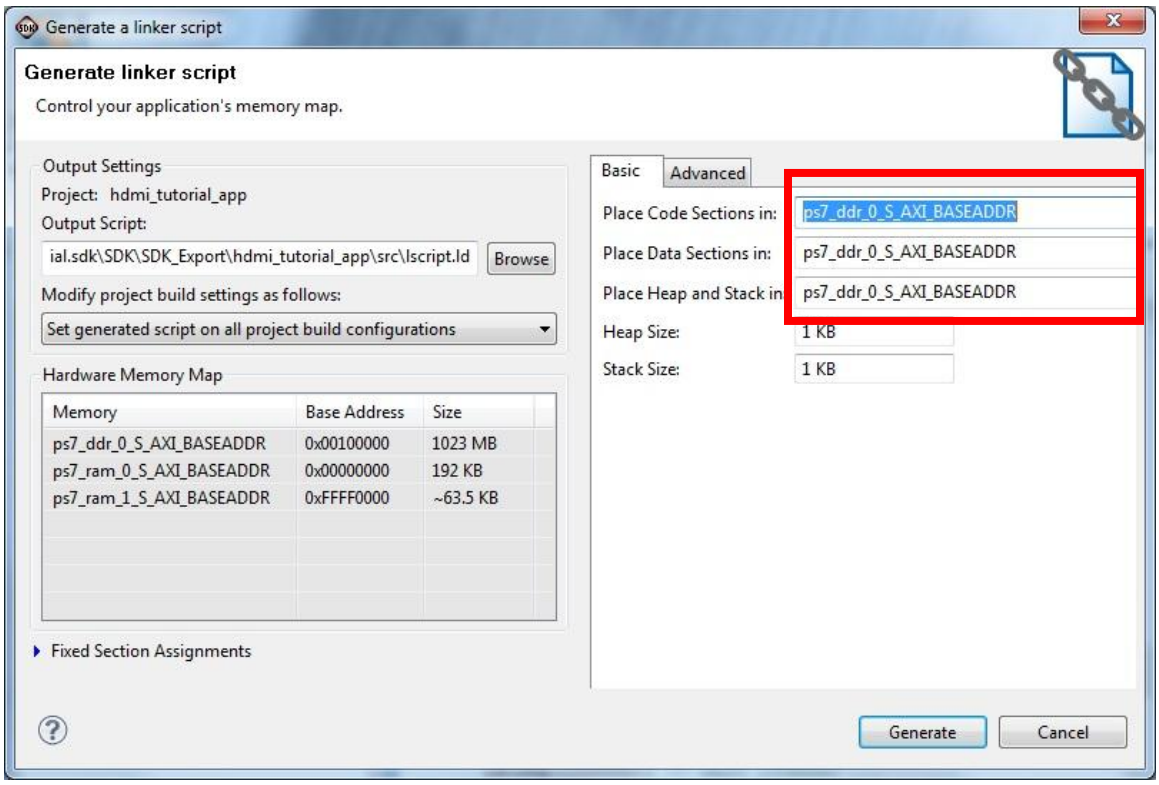


Figure 15 – Generate a Linker Script

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

Copy the provided example C files to the new application.
Note that the source for the new application is placed in the following directory:
C:\FMC_IMAGEON_Tutorial\tutorial\tutorial.sdk\SDK\SDK_Export\hdmi_tutorial_app\src

23. From the following directory:
    C:\FMC_IMAGEON_Tutorial**code\fmc_imageon_hdmi_passthrough**
    Copy the following files to the src directory of the new application.
    fmc_imageon_hdmi_passthrough.c
    fmc_imageon_hdmi_passthrough.h
    video_resolution.c
    video_resolution.h
    In the Project Explorer window, select the `hdmi_tutorial_app` application, right-click,

then select **Refresh** from the pop-up menu.
SDK will recognize the new source files.
If the Build Automatically setting is enabled, SDK will automatically build the application.

Modify the hello world application

24. 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_hdmi_passthrough.h"
fmc_imageon_hdmi_passthrough_t demo;

void print(char *str);

int main()
{
    init_platform();

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

    demo.uBaseAddr_IIC_FmcImageon = XPAR_FMC_IMAGEON_IIC_0_BASEADDR;
    fmc_imageon_hdmi_passthrough_init( &demo );

    cleanup_platform();

    return 0;
}
```

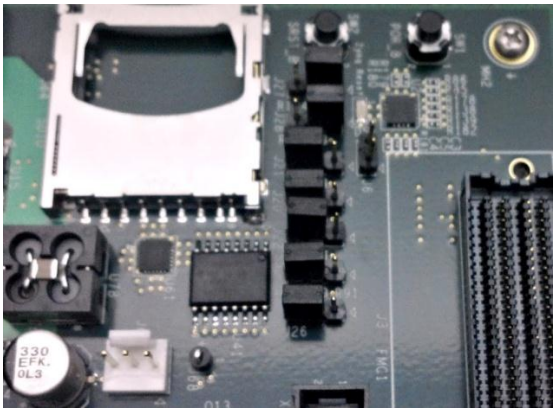
25. If the Build Automatically setting is enabled, SDK will automatically build the application.
    If not, right-click on the application and select **Build Project** to build the application.

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 a DVI or HDMI source to the FMC module's HDMI IN 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 a DVI or HDMI source to the FMC module's HDMI IN 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 HDMI 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 **OK**.
   It will take approximately 10 seconds to program the bitstream to hardware
4. Right-click **hdmi_tutorial_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 `hdmi_tutorial_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 HDMI Pass-Through           --
------------------------------------------------------

FMC-IMAGEON Initialization ...
HDMI Input Initialization ...
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
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 ...

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 !

Step 2) Bridge to/from the AXI4-Stream Interface

In this section, the design will be augmented to include bridges to/from the AXI4-Stream for Video interface.

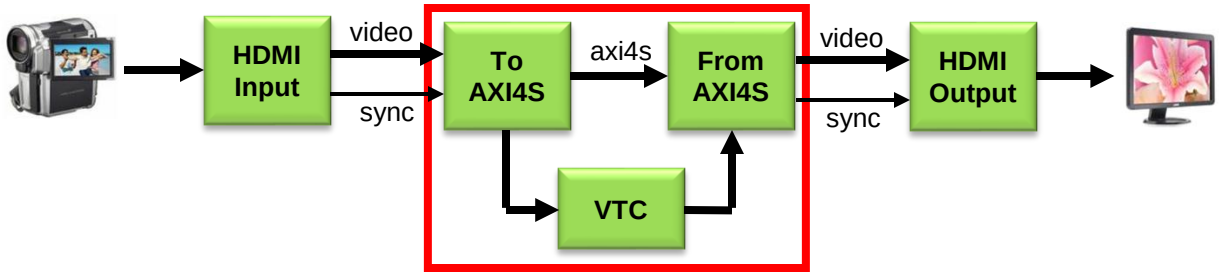


Figure 16 – AXI4-Stream based HDMI Pass-Through - Block Diagram

This is accomplished with the following video IP cores from Xilinx:

- 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 HDMI input interface's video clock. The AXI4-Stream interface will be running on a separate clock.

If not done so already, open the Plan Ahead project created in Step 1.

Modify the Embedded Hardware Design with EDK

Open the Embedded sub-system

1. In Design Sources, double-click `{module name}.xmp`.

Create the new internal clock that will be used for the AXI4-Stream for Video interface.

2. In the **Zynq** tab, click on the **Clock Generation** box.
   The Clock Wizard opens.
3. Expand the **PL Fabric Clocks** section.
   Notice that FCLK_CLK0, used by the axi4lite_0 interconnect, is set to 50 or 100 MHz.
4. Set the **FCLK_CLK1** Requested Frequency to **150.0 MHz**.
5. Click the **Validate Clocks** button.

   Notice that the Actual Frequency is 142.x MHz. Since the video input and video output interfaces are running at 148.5 MHz, is it acceptable to set FCLK_CLK1 to 142.x MHz ? Actually, the answer is yet.
   During active video, the AXI4-Stream will support the back-pressure on the slower clock using the FIFOs built into the AXI4-Stream bridges. During blanking periods, the FIFOs will empty, and the AXI4-Stream interface will catch up.

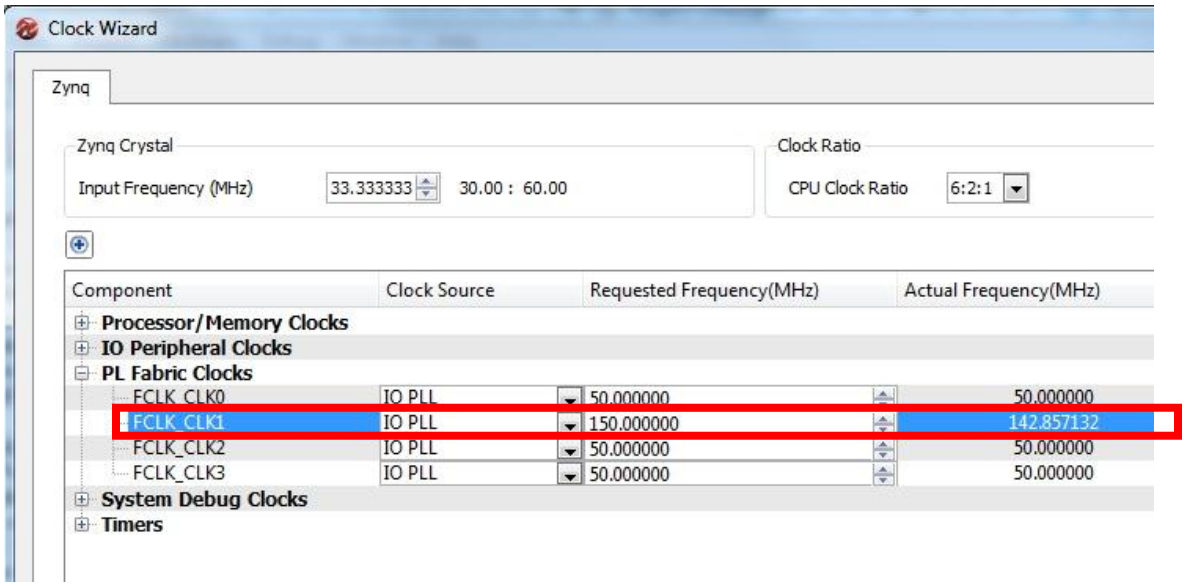


Figure 17 – Configuring FCLK_CLK1 as an internal 150MHz clock

6. Click **OK**.

Add the “Video Timing Controller” PCORE to the design.

7. From the IP catalog, expand **EDK Install** => **Video and Image Processing** and double-click on **Video Timing Controller** to add it.
   A message appears asking if you want to add the v_tc 5.01.a IP instance to your design.
8. Click **Yes**.
   The configuration window for **v_tc_v5_01_a** opens.

9. In the **Optional Features** section,
   Leave all the parameters as they are.
10. In the **Options** section, make the following changes:

Enable Generation	Verify that it is **enabled**
Enable Detection	Set to **enabled**

    Leave all other parameters as they are.
11. In the **Generation Options** section, make the following changes:

Synchronize Generator to Detector or to fsync_in	Set to **enabled**

    Leave all other parameters as they are.
12. In the **Detection Options** section, make the following changes:

Vertical Sync	Set to **disabled**
Horizontal Sync	Set to **disabled**

    (NOTE : the fmc_imageon_hdmi_in pcore does not generate the vsync/hsync signals)
    Leave all other parameters as they are.
13. Click **OK**.
14. A message window opens with the message "v_tc IP with version number 5.01.a is instantiated with name v_tc_0". It will ask you to determine to which processor to connect.
    Keep the default choice of processor as "processing_system7_0".
15. 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.

Add the “Video In to AXI4-Stream” PCORE to the design.

16. From the IP catalog, expand **EDK Install** => **Video and Image Processing** and double-click on **Video In to AXI4-Stream** to add it.
    A message appears asking if you want to add the v_vid_in_axi4s 2.01.a IP instance to your design.
17. Click **Yes**.
    The configuration window for **v_vid_in_axi4s_v2_01_a** opens.
18. In the **General** section, make the following changes:

Video Format	Set to **YUV_4:2:2**

    (NOTE : the ADV7611 device, or HDMI input, is configured for the YUV 4:2:2 format)
    Leave all other parameters as they are.
19. Click **OK**.
20. There are a few connections that are not done automatically and will be done manually after all cores have been added to the design.

Add the “AXI4-Stream to Video Out” PCORE to the design.

21. From the IP catalog, expand **EDK Install** => **Video and Image Processing** and double-click on **AXI4-Stream to Video Out** to add it.
    A message appears asking if you want to add the v_axi4s_vid_out 2.01.a IP instance to your design.
22. Click **Yes**.
    The configuration window for **v_axi4s_vid_out_v2_01_a** opens.
23. In the General section, make the following changes:

Master or Slave mode	Verify that it is set to **Slave**

Video Format	Set to **YUV_4:2:2**

(NOTE : the ADV7511 device, or HDMI output, is configured for the YUV 4:2:2 format)
Leave all other parameters as they are.

24. Click **OK**.
25. 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.

26. Click the **Bus Interfaces** tab, which lists the IPs and their bus connections. Expand `v_vid_in_axi4s_0`, `v_vtc_0`, and `v_axi4s_vid_out_0`.
27. Notice that the `v_tc_0` was automatically connected to the `axi4lite_0`.
28. Connect the `v_vid_in_axi4s_0`'s M_AXIS_VIDEO interface to the `v_axi4s_vid_out_0`'s S_AXIS_VIDEO interface.
29. Connect the `v_vid_in_axi4s_0`'s VTIMING_OUT interface to the `v_tc_0`'s VTIMING_IN interface.
30. Connect the `v_tc_0`'s VTIMING_OUT interface to the `v_axi4s_vid_out_0`'s VTIMING_IN interface.

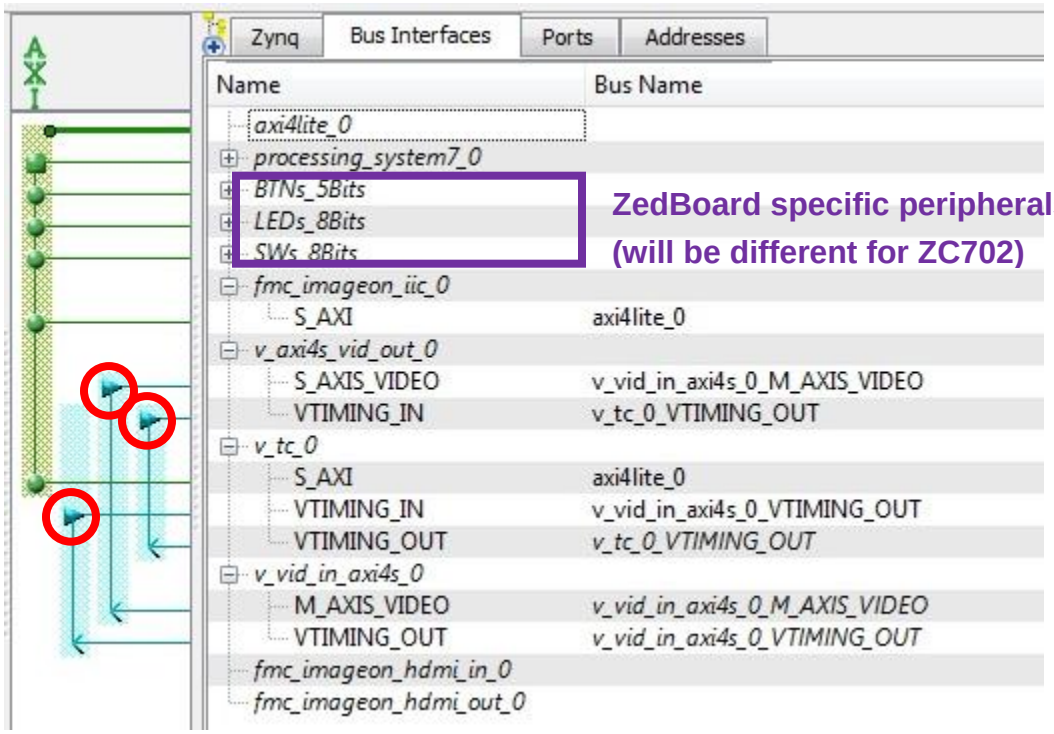


Figure 18 – Bus Interfaces tab with video cores expanded

Connect the ports in the design.

31. Make the following additional port connections in the Ports tab.

IP	Port	Connection

v_tc_0	s_axi_aclken	net_vcc
	resetn	net_vcc
	clken	net_vcc
	det_clken	net_vcc
	gen_clken	v_axi4s_vid_out_0::vtg_ce
v_vid_in_axi4s_0	vid_in_clk	External Ports:: fmc_imageon_hdmi_in_0_clk_pin
	vid_de	fmc_imageon_hdmi_in_0::video_de
	vid_vblank	fmc_imageon_hdmi_in_0:: video_vblank
	vid_hblank	fmc_imageon_hdmi_in_0:: video_hblank
	vid_data	fmc_imageon_hdmi_in_0:: video_data
	rst	net_gnd
	aresetn	net_vcc
	aclken	net_vcc
	axis_enable	net_vcc
	(BUS_IF) S_AXIS_VIDEO::aclk	processing_system7_0::FCLK_CLK1
v_axi4s_vid_out_0	vid_out_clk	External Ports:: fmc_imageon_hdmi_in_0_clk_pin
	video_de	fmc_imageon_hdmi_out_0:: video_de
	video_vblank	fmc_imageon_hdmi_out_0:: video_vblank
	video_hblank	fmc_imageon_hdmi_out_0:: video_hblank
	video_data	fmc_imageon_hdmi_out_0:: video_video_data
	rst	net_gnd
	aresetn	net_vcc
	aclken	net_vcc
	(BUS_IF) S_AXIS_VIDEO::aclk	processing_system7_0::FCLK_CLK1

Table 5 – Graphical Ports Connections (Part 1)

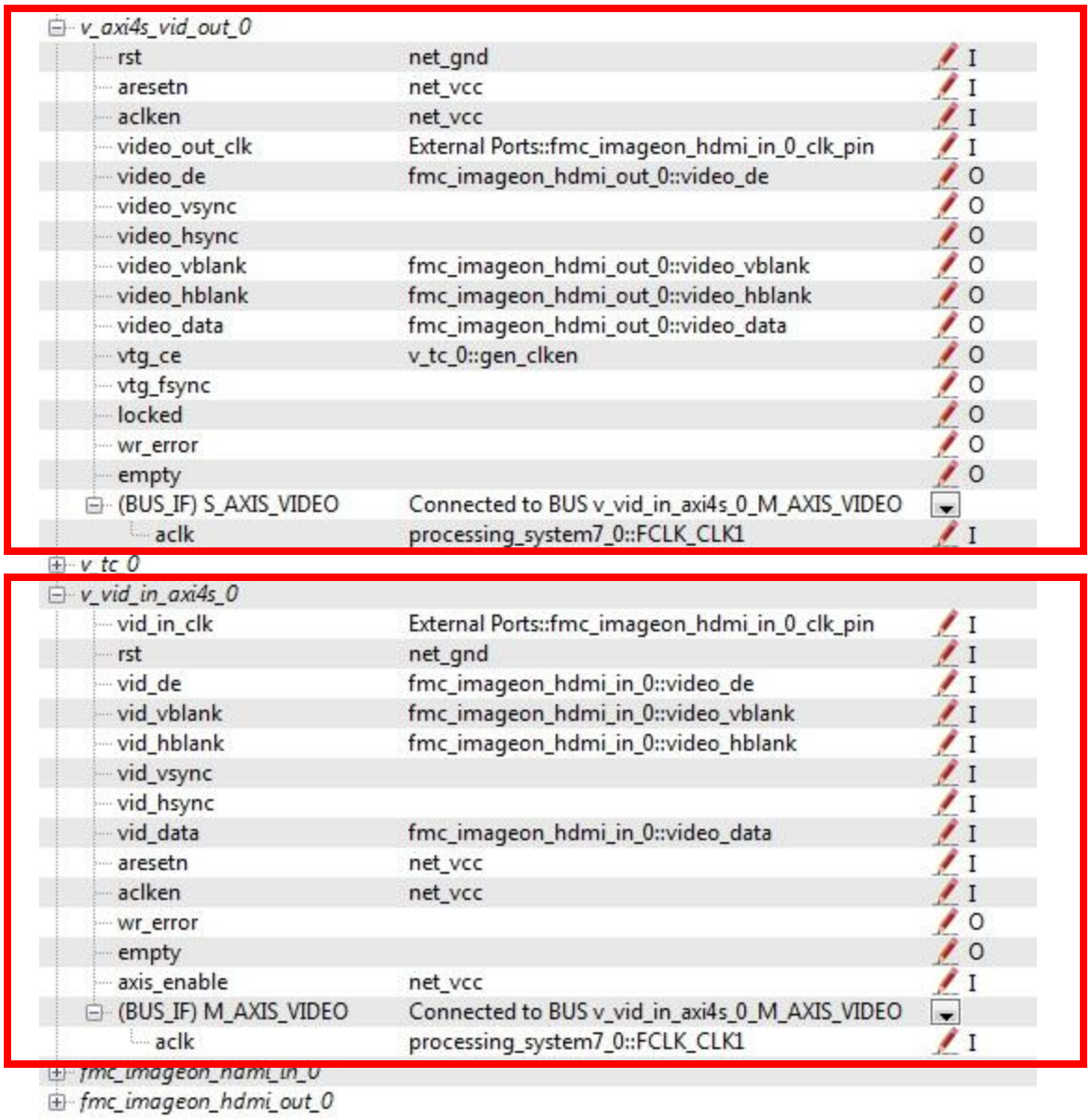


Figure 19 – Completed Ports Connections

Some ports are not visible in the graphical Ports user interface. The Filters in the Ports tab must be configured to view these missing ports, and make connections.

32. Open the "**<< Filters**" button on the right hand side
33. Enable the **By Connection => Defaults** option.
34. Connect the following ports
    (both refer to the same port, so connecting one will connect the other as well)

IP	Port	Connection

v_tc_0	(BUS_IF) VTIMING_IN::clk	External Ports:: fmc_imageon_hdmi_in_0_clk_pin
	(BUS_IF) VTIMING_OUT::clk	External Ports:: fmc_imageon_hdmi_in_0_clk_pin

Table 6 – Graphical Ports Connections (Part 2) – hidden v_tc_0 ports

35. When done, disable the **By Connection => Defaults** option.

Perform the Design Rule Check

36. Run Design Rule Check. This can be invoked from the menu by selecting
    Project => **Design Rule Check**
37. Ensure there are no errors in the console.

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

38. Close the XPS. The PlanAhead™ window becomes active again.

Build the hardware with PlanAhead

The active PlanAhead session updates with the project settings

Re-Build the bitstream.

1. In the Program and Debug list in the Flow Navigator, click **Generate Bitstream**.
   A dialog box appears asking whether all the processes starting for synthesis should be done.
2. Click **Yes**.
3. Ignore the following critical warnings that may appear:
   Cannot loc instance 'processing_system7_0_PS_PORB_IBUF' at site B5, …
   Cannot loc instance 'processing_system7_0_PS_SRSTB_IBUF' at site C6, …
   Cannot loc instance 'processing_system7_0_PS_CLK_IBUF' at site F7, …
   … 4 other warnings …

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

4. Select **Open Implemented Design**
5. Click **OK**.

The resource utilization for this design can be seen in the Project Summary.

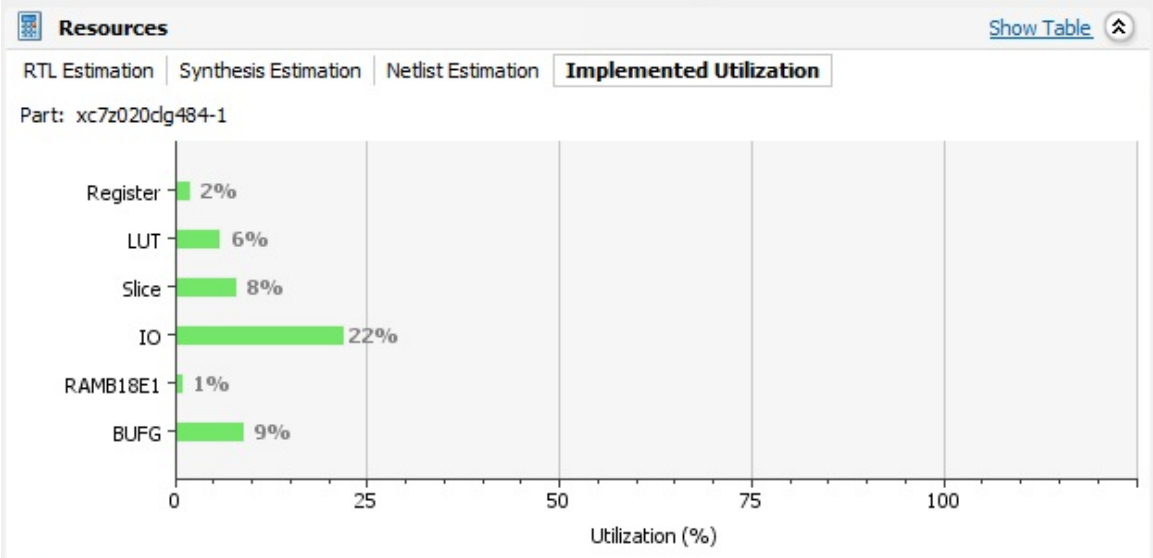


Figure 20 – AXI4-Stream based HDMI Pass-Through – Resource Utilization

You have successfully built the AXI4-Stream based HDMI Pass-Through design !

Modify the Embedded Software Application with SDK

Launch SDK from the PlanAhead software.

1. In the PlanAhead software, 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**.
4. If you get a “Module Already Exported” dialog box,
   click **Yes** to overwrite with the new design.
   SDK opens.

The new design contains one additional PCORE that can be configured by the processor, the Video Timing Controller. Notice that the VTC driver was automatically added to the Board Support Package.

5. In the Project Explorer, select **hdmi_tutorial_bsp**, then right-click and select **Board Support Package Settings**.
   The BSP dialog will open.
6. Select the drivers category, and notice that version 3.00.a of the vtc driver was added.

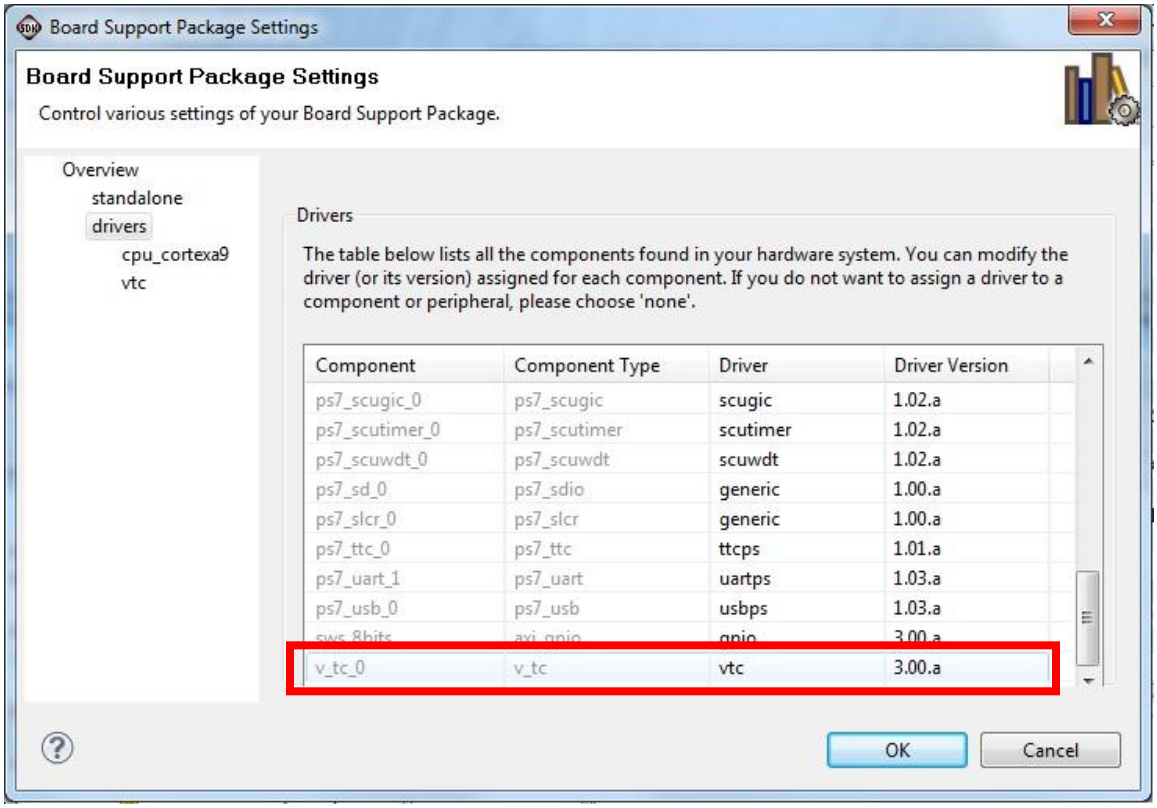


Figure 21 – VTC driver automatically added to BSP

7. Click **OK**.

8. If the Build Automatically setting is enabled, the standalone BSP will automatically rebuild. If not, right-click on the standalone BSP and select **Build Project**.

The fmc_imageon_hdmi_passthrough.c/.h source files already contains initialization code for the Video Timing Controller that will automatically activate (using the #if / #endif directives). The main program must be modified to specify the base address of this PCORE.

9. 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_hdmi_passthrough.h"
fmc_imageon_hdmi_passthrough_t demo;

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

int main()
{
```

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

```
    init_platform();

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

    demo.uBaseAddr_IIC_FmcImageon = XPAR_FMC_IMAGEON_IIC_0_BASEADDR;

    demo.uBaseAddr_VTC_Axi4sTiming = XPAR_V_TC_0_BASEADDR;

    fmc_imageon_hdmi_passthrough_init( &demo );

    cleanup_platform();

    return 0;
}
```

10. 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.

Execute the HDMI AXI4-Stream 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 **OK**.
   It will take approximately 10 seconds to program the bitstream to hardware
4. Right-click **hdmi_tutorial_app**
   and select **Run as > Run Configurations**
5. Select the run configuration that was created in the previous step :
   `hdmi_tutorial_app Debug`.
   Click the **Device Initialization** tab in the launch configurations and make sure the path to the initialization TCL file is valid.
   (HINT : If you moved the project, you should delete the previous run configuration and create a new one)
6. Click **Apply** and then **Run**.
7. If you get a Reset Status dialog box indicating that the current launch will reset the entire system, click **OK**.
8. You should see something similar to the following on your serial console:

```
Hello World

------------------------------------------------------
--          FMC-IMAGEON HDMI Pass-Through           --
--           with Video Timing Detector             --
------------------------------------------------------

FMC-IMAGEON Initialization ...
HDMI Input Initialization ...
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
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 ...
Video Timing Controller Initialization ...

Done


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

You have successfully executed the AXI4-Stream based HDMI pass-through on hardware !

Step 2d) Debug AXI4-Stream Interface (Optional)

ChipScope is a very powerful debug tool. It allows the user to insert a "logic analyzer" in the fabric for debug purposes. This section explains how to add an AXI monitor on the AXI4-Stream for Video Interface.

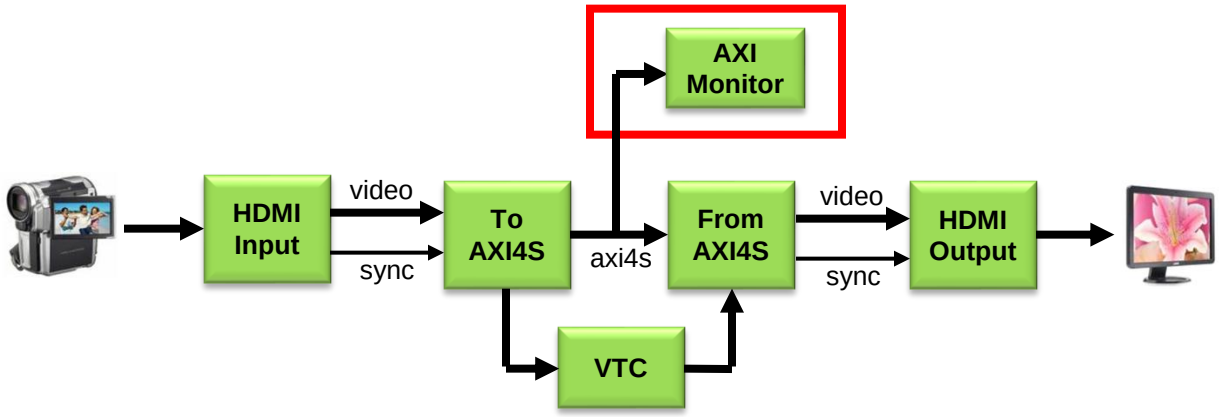


Figure 22 – AXI4-Stream based HDMI Pass-Through - Block Diagram

If not done so already, open the Plan Ahead project created in Step 2.

Modify the Embedded Hardware Design with EDK

Open the Embedded sub-system

1. In Design Sources, double-click `{module name}.xmp`.

Insert an AXI monitor into the design

2. From the menu, select **Debug** => **Debug Configuration**
   The "Debug Configuration" dialog opens.
3. Click on the "**Add ChipScope Peripheral…**" button.
   The "Add New ChipScope Peripheral" dialog opens.

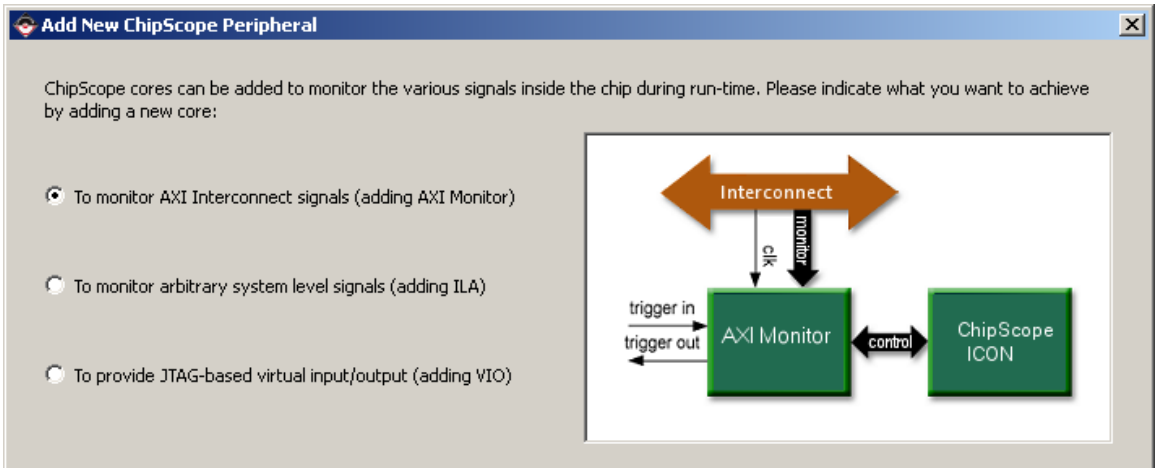


Figure 23 – Monitor AXI Interconnect

4. Select the "**To monitor AXI Interconnect signals (adding AXI Monitor)**" option, then click **OK**.
5. Under the **Basic** tab:
    a. For the **Monitor Bus Signals**, select the following AXI4-Stream bus interface : **v_vid_in_axi4s_0.M_AXIS_VIDEO**
    b. For the **number of signal samples to collect**, select **4096**

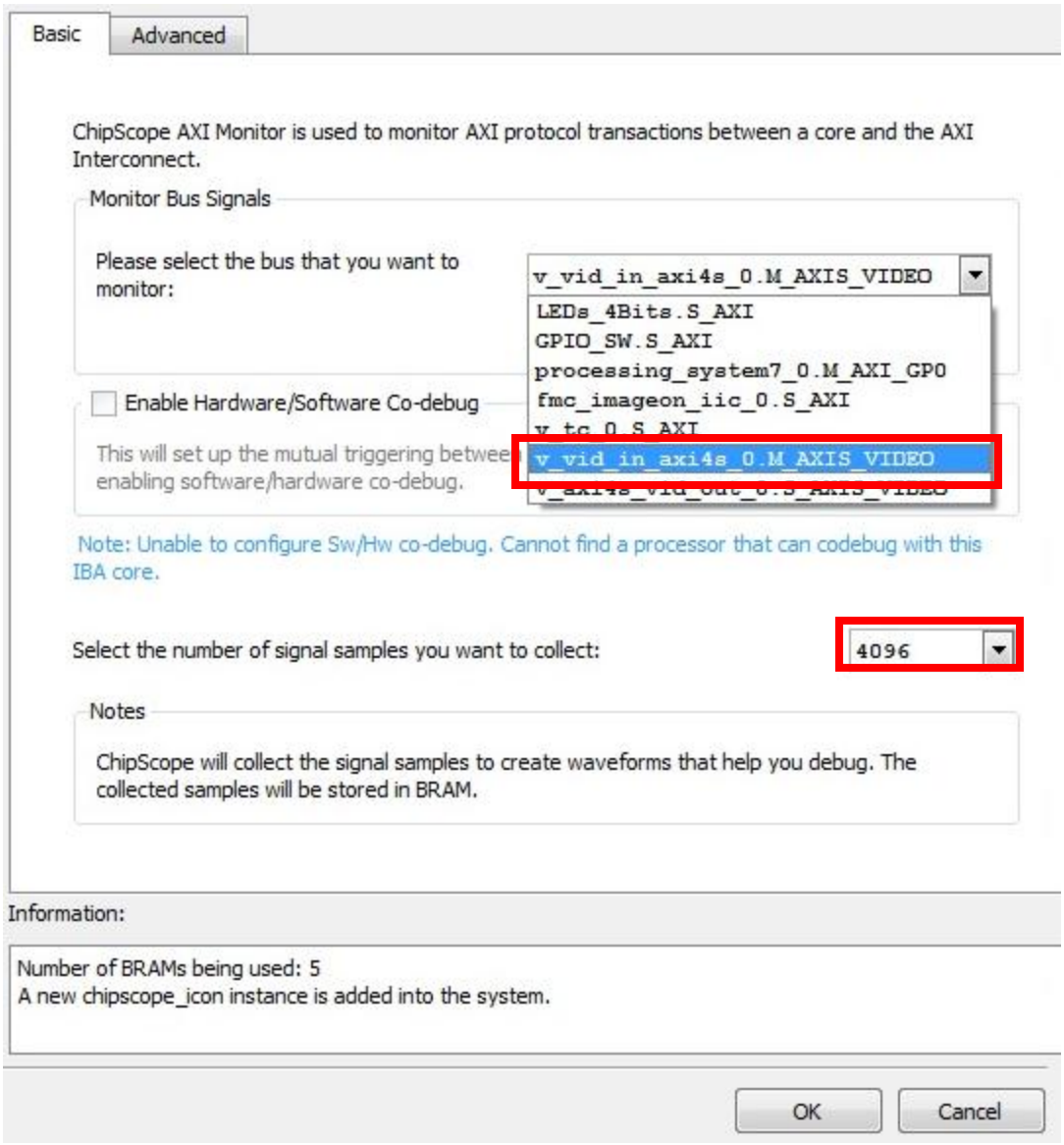


Figure 24 – Configuring AXI Monitor

6. Notice that this configuration consumes **5 BRAM**s.
7. Click **OK**.
8. You will get the following error message:
   ERROR: EDK – Could not connect MONITOR chipscope_axi_monitor_0's RESET MON_AXI_ARESETN to INSTANCE v_vid_in_axi4s_0's RESET PORT as the latter is itself un-connected.

This error occurs because the Chipscope AXI Monitor clock and reset ports are associated with the AXI4-Stream clock and reset ports. If these ports do not exist for the AXI4-Stream interface, the AXI Monitor is not able to populate with a clock and reset connection.

There is a simple workaround, that involves connecting the AXI_MON_ACLK and AXI_MON_ARESETN ports manually in the MHS file.

9. Click on the **Project** tab, then double-click on the {module name}**.mhs** file.
10. Scroll down to the end of the MHS file, and manually add the following two port connections to the chipscope_axi_monitor_0 instance.

```
BEGIN chipscope_axi_monitor
  PARAMETER INSTANCE = chipscope_axi_monitor_0
  PARAMETER HW_VER = 3.05.a
  PARAMETER C_USE_INTERFACE = 1
  PARAMETER C_NUM_DATA_SAMPLES = 4096
  BUS_INTERFACE MON_AXI_S = v_vid_in_axi4s_0_M_AXIS_VIDEO
  PORT CHIPSCOPE_ICON_CONTROL = chipscope_axi_monitor_0_icon_ctrl
  PORT mon_axi_aclk = processing_system7_0_FCLK_CLK1
  PORT mon_axi_aresetn = net_vcc
END
```

11. Save the file.

Notice that the AXI Monitor connection can be seen in the Bus Interfaces tab.

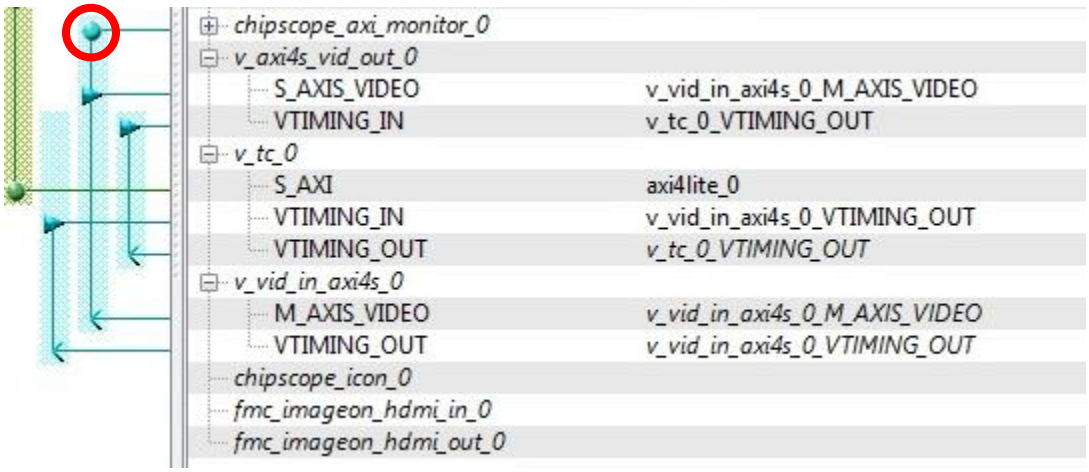


Figure 25 – Bus Interfaces tab with AXI Monitor connection

Perform the Design Rule Check.

12. Run Design Rule Check. This can be invoked from the menu by selecting **Project** => **Design Rule Check**
13. Ensure there are no errors in the console.
    NOTE : If there are errors, double-check the steps you followed.

14. Close the XPS. The PlanAhead™ window becomes active again.

Build the hardware with PlanAhead

The active PlanAhead session updates with the project settings

Re-Build the bitstream.

1. In the Program and Debug list in the Flow Navigator, click **Generate Bitstream**.
   A dialog box appears asking whether all the processes starting for synthesis should be done.
2. Click **Yes**.
3. Ignore the following critical warnings that may appear:
   Cannot loc instance 'processing_system7_0_PS_PORB_IBUF' at site B5, …
   Cannot loc instance 'processing_system7_0_PS_SRSTB_IBUF' at site C6, …
   Cannot loc instance 'processing_system7_0_PS_CLK_IBUF' at site F7, …
   … 12 other warnings …

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

4. Select **Open Implemented Design**
5. Click **OK**.

The resource utilization for this design can be seen in the Project Summary.

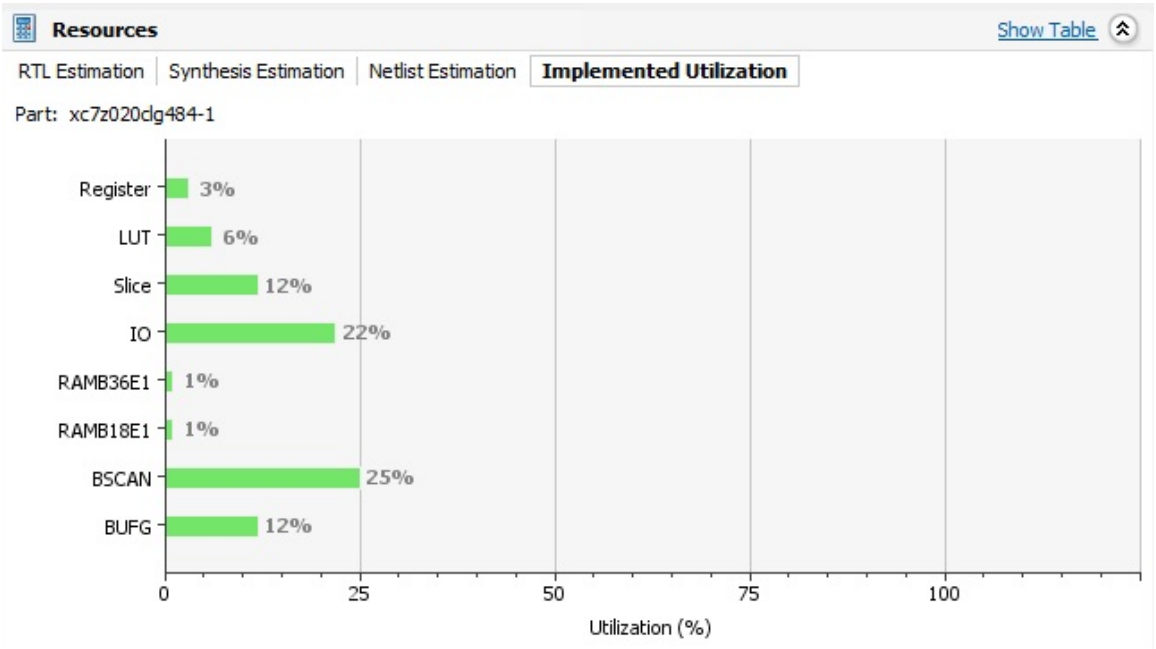


Figure 26 – AXI4-Stream Debug Design – Resource Utilization

Compare this resource utilization with that of the previous section. How much resources are being used by the AXI Monitor ?

Execute the AXI4-Stream Debug Design on Hardware using SDK

Launch SDK from the PlanAhead software.

1. In the PlanAhead software, 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.
   If the "Include Bitstream" checkbox is not checked, the build process may not have finished yet ...
2. Check the **Launch SDK** check box.
3. Click **OK**.
4. If you get a "Module Already Exported" dialog box,
   click **Yes** to overwrite with the new design.
   SDK opens.

From SDK, rebuild the standalone BSP and application

5. In the Project Explorer section, right-click on the application and select **Build Project** to re-build the stand-alone BSP and application.

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

6. In the SDK menu, select **Xilinx Tools** => **Program FPGA**
   The "Program FPGA" dialog opens.
7. 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)
8. Click **OK**.
   It will take approximately 10-15 seconds to program the bitstream to hardware
9. Right-click **hdmi_tutorial_app**
   and select **Run as > Run Configurations**
10. Select the run configuration that was created in the previous step :
    `hdmi_tutorial_app Debug`.
    Click the **Device Initialization** tab in the launch configurations and make sure the path to the initialization TCL file is valid.
    (HINT : If you moved the project, you should delete the previous run configuration and create a new one)
11. Click **Apply** and then **Run**.
12. If you get a Reset Status dialog box indicating that the current launch will reset the entire system, click **OK**.
13. You should see the same output as in the previous section on your serial console:

```
Hello World

------------------------------------------------------
--          FMC-IMAGEON HDMI Pass-Through           --
--           with Video Timing Detector             --
------------------------------------------------------

FMC-IMAGEON Initialization ...
HDMI Input Initialization ...
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
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 ...
Video Timing Controller Initialization ...

Done


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

Now that the design is running on hardware, launch ChipScope Analyzer to debug the AXI4-Stream for Video interface.

Debug the AXI4-Stream Interface with ChipScope Analyzer

In the PlanAhead cockpit, launch ChipScope Analyzer

1. In the Flow Navigator, under the **Program and Debug** section, click on **Launch ChipScope Analyzer**
   ChipScope Analyzer opens.
2. Click on the **Open Cable/Search JTAG Chain** button.
   The "JTAG Chain Device Order" dialog opens.
3. Click **OK**.

The PlanAhead project automatically created a configuration file called debug_nets.cdc for you. Import this file to configure your ChipScope Analyzer project.

4. From the menu, select **File** => **Import**.
   The "Signal Import" dialog opens.
5. Click the **Select New File** button.
   The "Open Signal File" dialog opens.
6. Specify the following file in the .runs directory:
   C:\FMC_IMAGEON_Tutorial\tutorial\tutorial.runs\impl_1**debug_nets.cdc**
7. Click **Open**.
8. In the "Signal Import" dialog,
   ensure that the **Auto-create Buses** option is enabled.
9. Click **OK**.

At this point, you can analyze the AXI4-Stream interface signals.

If you are not able to operate ChipScope, get familiar with the Zynq Concept, Tools, and Techniques (CTT) literature.

For ZC702:

Zynq Concepts, Tools, and Techniques
http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_3/ug873-zynq-ctt.pdf

For ZedBoard:

Zynq Concepts, Tools, and Techniques on ZedBoard
http://www.zedboard.org/design

Congratulations ! You have added debug capabilities to your AXI4-Stream based video design ! You are now ready to add more Xilinx Video IP cores to the design.

Step 3) Add a Video Frame Buffer

In this section, the design will be augmented with a Video Frame Buffer.

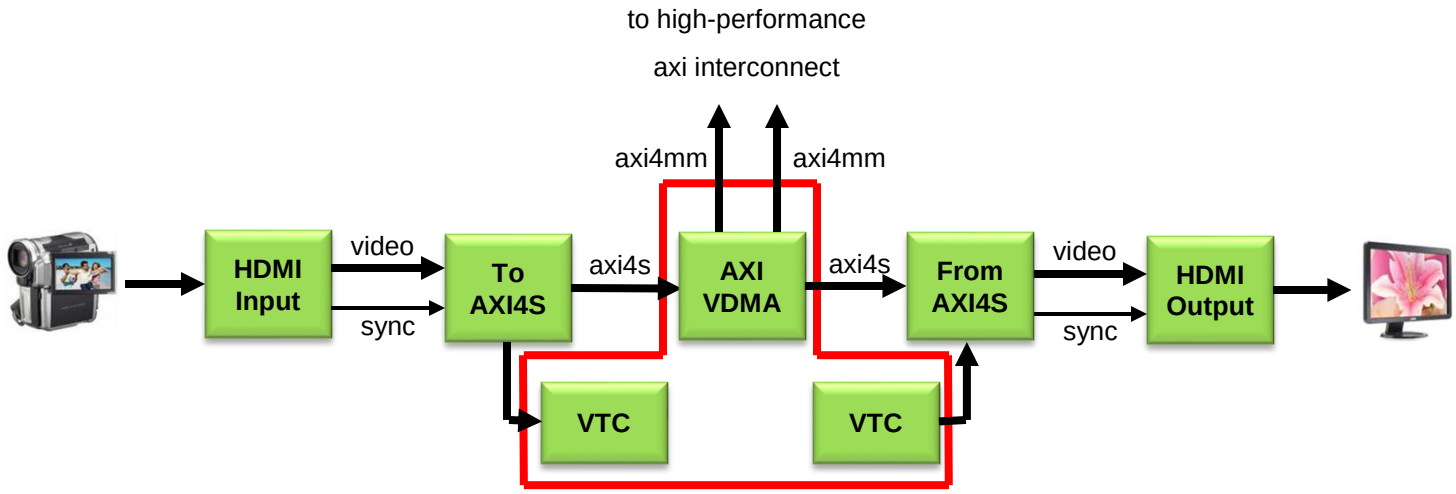


Figure 27 – Block Diagram – HDMI Video Frame Buffer

This is accomplished with the following video IP cores from Xilinx:

- Video Timing Controller
  - this core is capable of:
    - detecting the video timing on the 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
- AXI Video DMA
  - this core is a DMA engine, specialized for video applications
  - it will route AXI4-Stream traffic to/from external memory, effectively implementing a video frame buffer

The video frame buffer allows the input and output video paths to be de-coupled from each other, allowing them to operate on different clocks.

If you are new to the AXI_VDMA frame buffer, the best place to start is to read the “Triple Frame Buffer Example” chapter in the AXI_VDMA datasheet. At a very minimum, you absolutely MUST read this chapter. This chapter describes the most typical use case for the core, which happens to be the case that will be implemented in this tutorial.

Before diving into the implementation, a few topics are worth mentioning:

For the AXI Video DMA core, the bus interfaces, ports, and parameters are identified with the following acronyms:

- S2MM
  - signifies AXI4-Stream to AXI4-Memory-Mapped
  - used to identify the portion that is “writing” video content to external memory
- MM2S
  - signifies AXI4-Memory-Mapped to AXI4-Stream
  - used to identify the portion that is “reading” video content from external memory

There are a lot of parameters and registers related to GENLOCK and FSYNC, so understanding the distinction between the two is important in order to fully understand these numerous parameters and registers.

- GENLOCK
  - identifies which frame store is being accessed
  - relates to video storage (memory) location
- FSYNC
  - identifies when a video frame starts
  - relates to video timing

The design that will be implemented in this part of the tutorial will actually run on three separate clock domains. The input interface runs on the HDMI input interface’s video clock. The output interface runs on a separate video clock, generated by the CDCE913 video clock synthesizer on the FMC-IMAGEON module. The AXI4-Stream interface, including the AXI Video DMA, run on the internally generated clock.

If not done so already, open the Plan Ahead project created in Step 2.

Modify the Embedded Hardware Design with EDK

Open the Embedded sub-system

1. In Design Sources, double-click `{module name}.xmp`.

If you went through (Step 2d) of this tutorial, remove the AXI monitor from the embedded design.

2. From the menu, select **Debug** => **Debug Configuration**
   The “Debug Configuration” dialog opens.
3. Select **chipscope_axi_monitor_0**.
4. Click on the “**Delete ChipScope Peripheral…**” button.
5. Click **Yes** to confirm.
6. Click **OK**.

Add the “fmc_imageon_video_clk1” external port to the design. This port is driven by the FMC module’s CDCE913 video clock synthesizer. We will use this clock source for the video output path.

7. In the Ports tab, click on the “Add External Port” button
   The Connection dialog appears, make the following changes:

External Port Name	Set to **fmc_imageon_video_clk1_pin**
Direction	Set to **I**
Class	Set to **CLK**

Net Name	Set to **fmc_imageon_video_clk1**

Click **OK**.

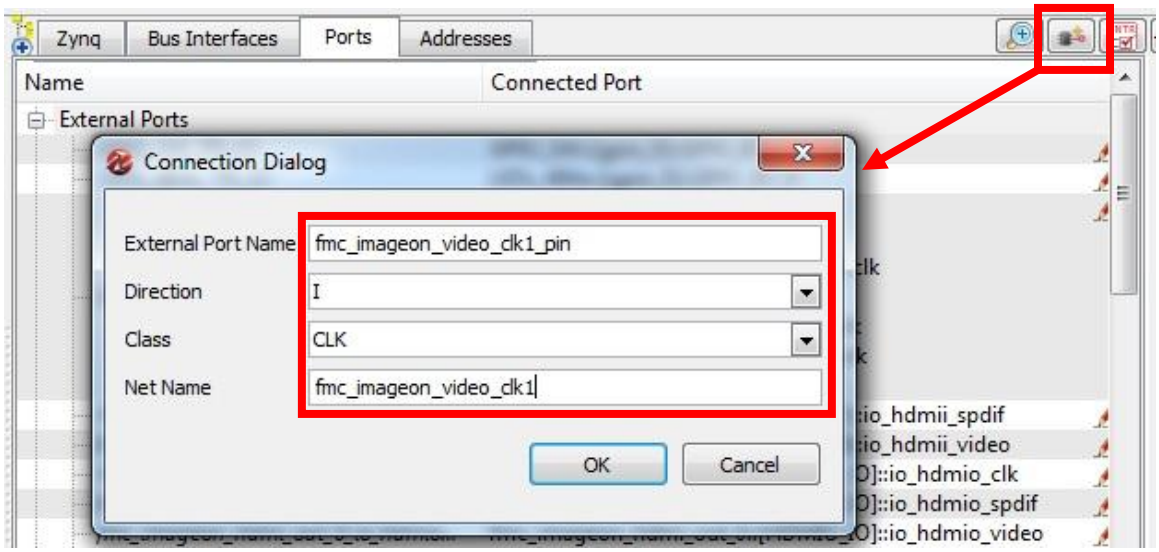


Figure 28 – Creating new fmc_imageon_clk1_pin port

8. For the new fmc_imageon_video_clk1_pin port, set the **Frequency (Hz)** to **148500000**

Connect the video output path to this new video clock source.

9. Change the following clock port connections to **fmc_imageon_video_clk1**

IP	Port	Connection
v_axi4s_vid_out_0	video_out_clk	External Ports:: fmc_imageon_video_clk1_pin
fmc_imageon_hdmi_out_0	clk	External Ports:: fmc_imageon_video_clk1_pin

Table 7 –Graphical Ports Connections (Part 1) - fmc_imageon_video_clk1

Modify the first “Video Timing Controller” PCORE to disable the “generation” option. A separate timing controller will be added for this purpose.

10. In the Bus Interfaces tab, double-click on the **v_tc_0** PCORE.
    In the Options section, make the following changes:

Enable Generation	Set to **disabled**

    Leave all other parameters as they are.
11. Click **OK**.

Add a second “Video Timing Controller” PCORE to the design.

12. From the IP catalog, expand **EDK Install** => **Video and Image Processing** and double-click on **Video DMA** to add it.
    A message appears asking if you want to add the v_tc 5.01.a IP instance to your design.
13. Click **Yes**.
    The configuration window for **v_tc_v5_01_a** opens.
14. In the **Optional Features** section,
    Leave all the parameters as they are.
15. In the **Options** section, make the following changes:

Enable Generation	Verify that it is **enabled**
Enable Detection	Verify that it is **disabled**

    Leave all other parameters as they are.
16. In the **Generation Options** section,
    Leave all parameters as they are.
17. Click **OK**.
18. A message window opens with the message "v_tc IP with version number 5.01.a is instantiated with name v_tc_1". It will ask you to determine to which processor to connect.
    Keep the default choice of processor as "processing_system7_0".
19. 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.

With the second Video Timing Controller, the video output path will be independent of the video input path. For this reason, the “AXI4-Stream to Video Output” PCORE must be re-configured to operate in master mode.

20. In the Bus Interfaces tab, double-click on the **v_axi4s_vid_out_0** PCORE.
    In the Options section, make the following changes:

Master or Slave mode	Set to **Master**

    Leave all other parameters as they are.
21. Click **OK**.

Add the “AXI Video DMA” PCORE to the design.

22. From the IP catalog, expand **EDK Install** => **DMA and Timer** and double-click on **AXI Video DMA** to add it.
    A message appears asking if you want to add the axi_vdma 5.04.a IP instance to your design.
23. Click **Yes**.
    The configuration window for **axi_vdma_v5_04_a** opens.
24. In the **User - S2MM Channel Options** section, make the following changes:

Enable Channel	Verify that it is **enabled**
Memory Map Data Width	Set to **64**
Stream Data Width	Set to **16**
Allow Unaligned Transfers (DRE)	Set to **enabled**
Enable Store and Forward	Verify that it is **enabled**
Maximum Burst Size	Verify that it is **16**
GenLock Mode	Verify that it is **0** (master)
Line Buffer Depth	Set to **4096**
Enable Frame Repeat on Error	Verify that it is **enabled**
Enable Start-Of-Frame on tuser(0)	Verify that it is **enabled**

Leave all other parameters as they are.

25. In the **User - MM2S Channel Options** section, make the following changes:

Enable Channel	Verify that it is **enabled**
Memory Map Data Width	Set to **64**
Stream Data Width	Set to **16**
Allow Unaligned Transfers (DRE)	Set to **enabled**
Enable Store and Forward	Verify that it is **enabled**
Maximum Burst Size	Verify that it is **16**
GenLock Mode	Verify that it is **1** (slave)
Line Buffer Depth	Set to **4096**
Output Start-Of-Frame on tuser(0)	Verify that it is **enabled**

Leave all other parameters as they are.

26. In the **User - VDMA Options** section, make the following changes:

Frame Stores	Set to **3**
Use Fsync	Set to **3** (S2MM uses FSYNC)
Enable Flush on Fsync	Set to **3** (S2MM uses flush)
Include Internal GenLock Bus	Verify that it is **enabled**.

Leave all other parameters as they are.

27. Click **OK**.
28. A message window opens with the message "axi_vdma IP with version number 5.03.a is instantiated with name axi_vdma_0". It will ask you to determine to which processor to connect.
    Keep the default choice of processor as "processing_system7_0".
29. Click **OK**.
30. 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.

31. Click the **Bus Interfaces** tab, which lists the IPs and their bus connections. Expand `axi_vdma_0`, `v_axi4s_vid_out_0`, `v_tc_0`, `v_tc_1`, and `v_vid_in_axi4s_0`.
32. Notice that the `axi_vdma_0` was automatically connected to the `axi4lite_0`.
33. Connect the `v_vid_in_axi4s_0`'s M_AXIS_VIDEO interface to the `axi_vdma_0`'s S_AXIS_S2MM interface.
34. Connect the `axi_vdma_0`'s M_AXIS_MM2S interface to the `v_axi4s_vid_out_0`'s S_AXIS_VIDEO interface.
35. Connect the `v_tc_1`'s VTIMING_OUT interface to the `v_axi4s_vid_out_0`'s VTIMING_IN interface.

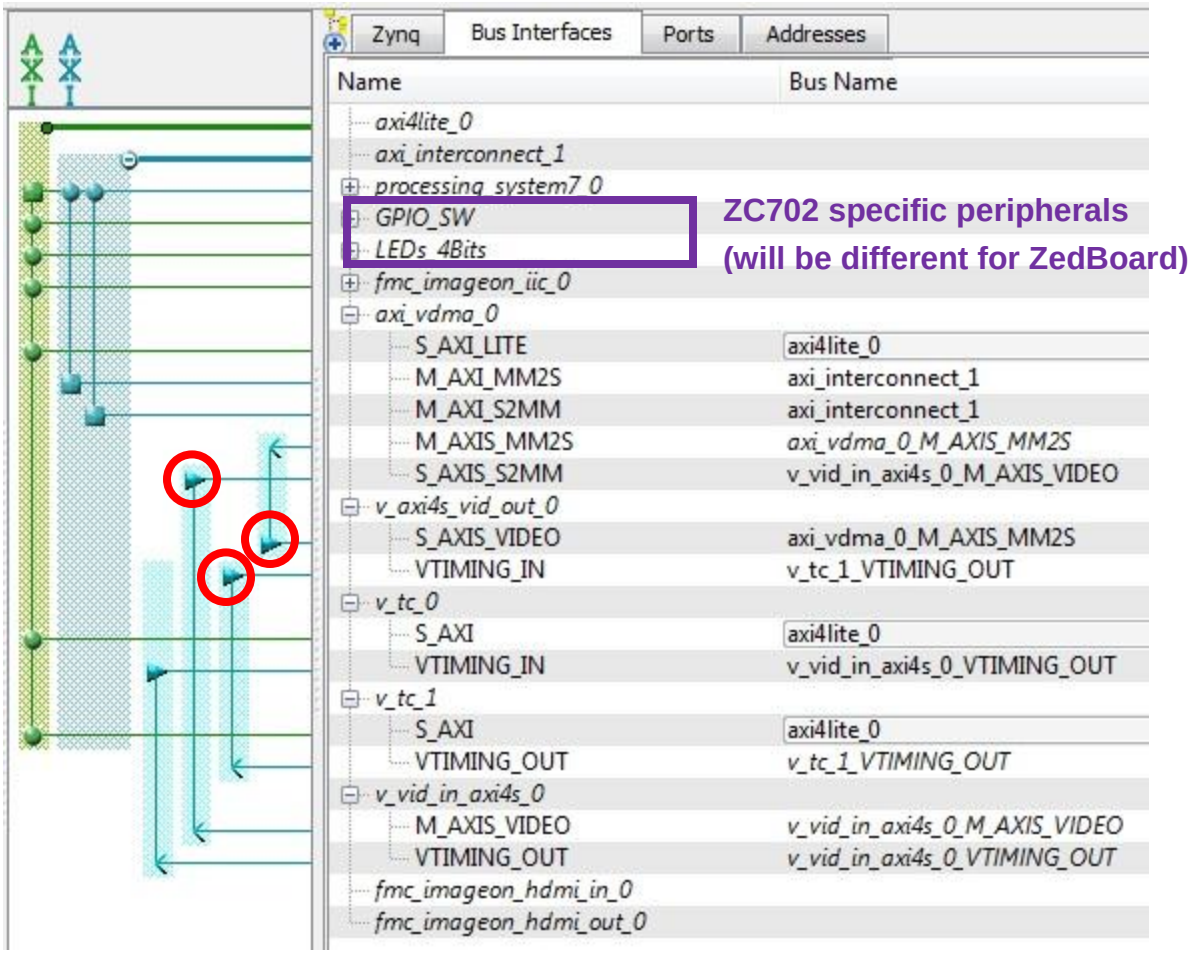


Figure 29 – Bus Interfaces tab with video cores expanded

Connect the ports in the design.

36. Make the following additional port connections in the Ports tab.

IP	Port	Connection
v_tc_1	s_axi_aclken	net_vcc
	resetn	net_vcc
	clken	net_vcc
	gen_clken	v_axi4s_vid_out_0::vtg_ce
axi_vdma_0	m_axis_mm2s_aclk	processing_system7_0::FCLK_CLK1
	s_axis_s2mm_aclk	processing_system7_0::FCLK_CLK1
	(BUS_IF) M_AXI_MM2S:: m_axi_mm2s_aclk	processing_system7_0::FCLK_CLK1
	(BUS_IF) M_AXI_S2MM:: s_axi_s2mm_aclk	processing_system7_0::FCLK_CLK1
processing_system7_0	(BUS_IF) S_AXI_HP0:: S_AXI_HP0_CLK	processing_system7_0::FCLK_CLK1
axi_interconnect_1	INTERCONNECT_ACLK	processing_system7_0::FCLK_CLK1

	INTERCONNECT_ARESETN	processing_system7_0::FCLK_RESET1_N

Table 8 – Graphical Ports Connections (Part 2) – v_tc_1 and axi_vdma_0 ports

Figure 30 – Completed Ports Connections – axi_vdma_0

Some ports are not visible in the graphical Ports user interface. The Filters in the Ports tab must be configured to view these missing ports, and make connections.

37. Open the "**<< Filters**" button on the right hand side
38. Enable the **By Connection => Defaults** option.
39. Connect the following ports
    (both refer to the same port, so connecting one will connect the other as well)

IP	Port	Connection
v_tc_1	(BUS_IF) VTIMING_OUT::clk	External Ports:: fmc_imageon_video_clk1_pin

Table 9 – Graphical Ports Connections (Part 3) – hidden v_tc_1 port

40. When done, disable the **By Connection => Defaults** option.

Perform the Design Rule Check

41. Run Design Rule Check. This can be invoked from the menu by selecting
    Project => **Design Rule Check**
42. Ensure there are no errors in the console.

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

43. Close the XPS. The PlanAhead™ window becomes active again.

Build the hardware with PlanAhead

The active PlanAhead session updates with the project settings

Re-generate the top level HDL file.

1. In Design Sources, right-click `{module name}.xmp` and select **Create Top HDL**. PlanAhead re-generates the `{module name}_stub.v` file.

Edit the UCF constraints file to add the new external fmc_imageon_video_clk1 pin.

2. In the Project Manager, expand the Constraints section, and double click on the UCF file
3. In the UCF file, un-comment the lines that define the fmc_imageon_video_clk1 pin

   For the ZC702, in the **zc702_fmc_imageon_hdmi_tutorial.ucf** file, this will be:

```
# Video Clock Synthesizer
NET "fmc_imageon_video_clk1_pin" LOC = "Y18"; # VCLK_1 - FMC-H4
NET fmc_imageon_video_clk1_pin TNM_NET = vout_clk
TIMESPEC TS_vout_clk = PERIOD vout_clk 148500 kHz;
```

   For the ZedBoard, in the **zed_fmc_imageon_hdmi_tutorial.ucf** file, this will be:

```
# Video Clock Synthesizer
NET "fmc_imageon_video_clk1_pin" LOC = "L18"; # VCLK_1 - FMC-H4
NET fmc_imageon_video_clk1_pin TNM_NET = vout_clk
TIMESPEC TS_vout_clk = PERIOD vout_clk 148500 kHz;
```

4. Save the file.

Re-Build the bitstream.

5. In the Program and Debug list in the Flow Navigator, click **Generate Bitstream**.
   A dialog box appears asking whether all the processes starting for synthesis should be done.
6. Click **Yes**.
7. Ignore the following critical warnings that may appear:
   Cannot loc instance 'processing_system7_0_PS_PORB_IBUF' at site B5, …
   Cannot loc instance 'processing_system7_0_PS_SRSTB_IBUF' at site C6, …
   Cannot loc instance 'processing_system7_0_PS_CLK_IBUF' at site F7, …
   … several other warnings …

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

8. Select **Open Implemented Design**
9. Click **OK**.

The resource utilization for this design can be seen in the Project Summary.

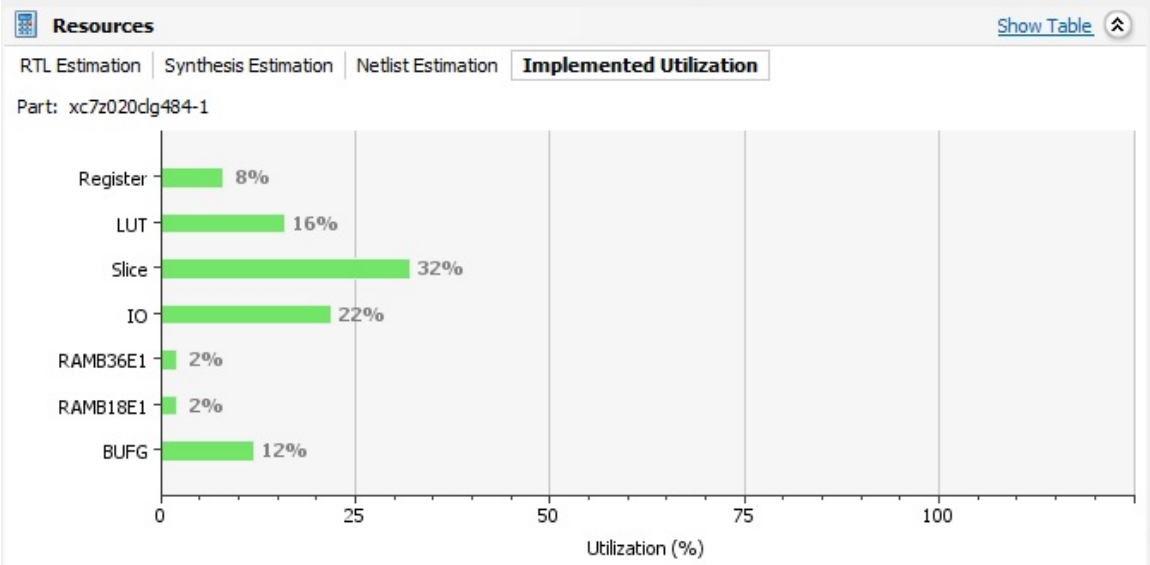


Figure 31 – Video Frame Buffer – Resource Utilization

You have successfully re-built the hardware design !

Modify the Embedded Software Application with SDK

Launch SDK from the PlanAhead software.

1. In the PlanAhead software, 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.
   If the "Include Bitstream" checkbox is not checked, the build process may not have finished yet ...
2. Check the **Launch SDK** check box.
3. Click **OK**.

4. If you get a “Module Already Exported” dialog box,
   click **Yes** to overwrite with the new design.
   SDK opens.

The new design contains one additional PCORE that can be configured by the processor, the AXI Video DMA. Notice that the AXIVDMA driver was automatically added to the Board Support Package.

5. In the Project Explorer, select **hdmi_tutorial_bsp**, then right-click and select **Board Support Package Settings**.
   The BSP dialog will open.
6. Select the drivers category, and notice that version 4.02.a of the axivdma driver was added.

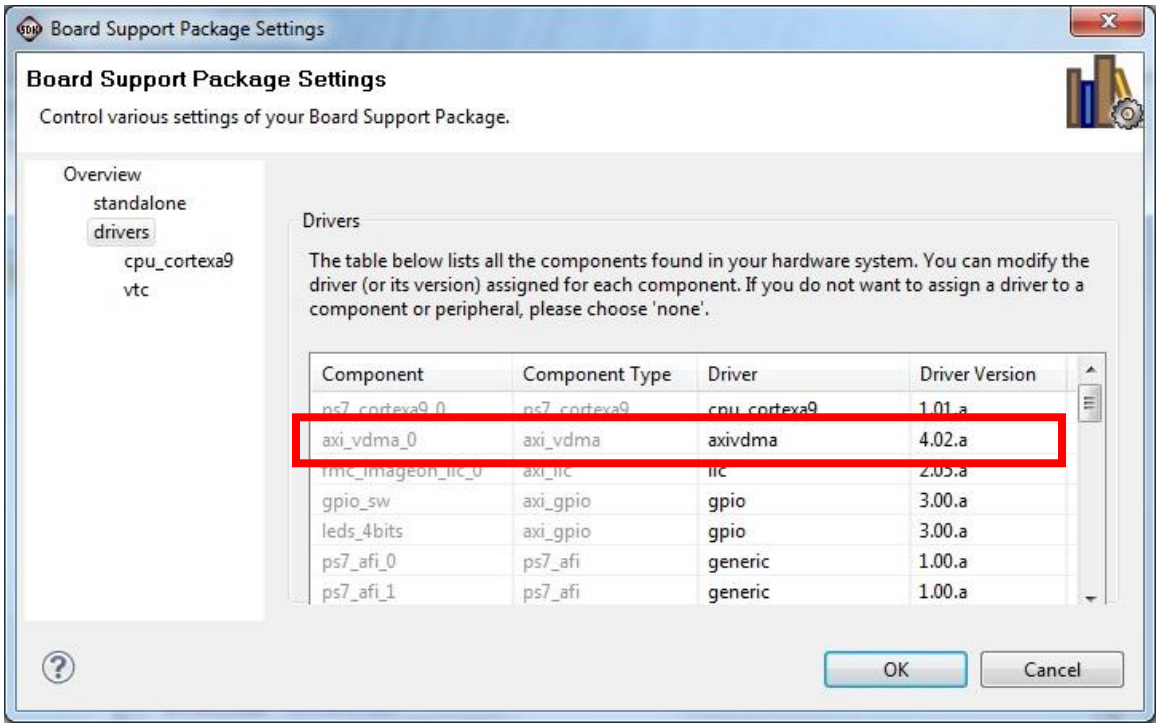


Figure 32 – VDMA driver automatically added to BSP

7. Click **OK**.

8. If the Build Automatically setting is enabled, the standalone BSP will automatically rebuild. If not, right-click on the standalone BSP and select **Build Project**.

Copy the provided example C files to the new application.
Note that the source for the new application is placed in the following directory:
C:\FMC_IMAGEON_Tutorial\tutorial\tutorial.sdk\SDK\SDK_Export\hdmi_tutorial_app\src

9. From the following directory:
   C:\FMC_IMAGEON_Tutorial**code\fmc_imageon_hdmi_framebuffer**

Copy the following files to the src directory of the new application.
- fmc_imageon_hdmi_frambuffer.c / .h
- video_frame_buffer.c / .h
- video_detector.c / .h
- video_generator.c / .h
- video_resolution.c / .h

If asked to replace existing files, answer yes.
Delete the following files, which are no longer needed.
- fmc_imageon_hdmi_passthrough.c / .h

In the Project Explorer window, select the `hdmi_tutorial_app` application, right-click, then select **Refresh** from the pop-up menu.
SDK will recognize the new source files.
If the Build Automatically setting is enabled, SDK will automatically build the application.

Modify the hello world application

10. 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_hdmi_framebuffer.h"
fmc_imageon_hdmi_framebuffer_t demo;

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

int main()
{
    init_platform();

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

    demo.uBaseAddr_IIC_FmcImageon       = XPAR_FMC_IMAGEON_IIC_0_BASEADDR;
    demo.uDeviceId_VTC_HdmiiDetector    = XPAR_V_TC_0_DEVICE_ID;
    demo.uDeviceId_VTC_HdmioGenerator   = XPAR_V_TC_1_DEVICE_ID;
    demo.uDeviceId_VDMA_HdmiFrameBuffer = XPAR_AXI_VDMA_0_DEVICE_ID;
    demo.uBaseAddr_MEM_HdmiFrameBuffer  = XPAR_DDR_MEM_BASEADDR + 0x10000000;
    demo.uNumFrams_HdmiFrameBuffer      = XPAR_AXIVDMA_0_NUM_FSTORES;
    fmc_imageon_hdmi_framebuffer_init( &demo );

    cleanup_platform();

    return 0;
}
```

11. 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.

Execute the HDMI AXI4-Stream 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 **OK**.
   It will take approximately 10-15 seconds to program the bitstream to hardware
4. Right-click **hdmi_tutorial_app**
   and select **Run as > Run Configurations**
5. Select the run configuration that was created in the previous step :
   `hdmi_tutorial_app Debug`.
   Click the **Device Initialization** tab in the launch configurations and make sure the path to the initialization TCL file is valid.
   (HINT : If you moved the project, you should delete the previous run configuration and create a new one)
6. Click **Apply** and then **Run**.
7. If you get a Reset Status dialog box indicating that the current launch will reset the entire system, click **OK**.
8. You should see something similar to the following on your serial console:

```
Hello World

------------------------------------------------------
--       FMC-IMAGEON HDMI Video Frame Buffer        --
------------------------------------------------------

FMC-IMAGEON Initialization ...
HDMI Input Initialization ...
Video Clock Synthesizer Configuration ...
HDMI Output Initialization ...
Video DMA (Output Side) Initialization ...
Video Timing Controller (generator) Initialization ...
        Video Resolution = 1080P
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
Video DMA (Input Side) Initialization ...
HDMI Output Re-Initialization ...

Done


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

You have successfully executed the Video Frame Buffer design 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
   http://www.xilinx.com/support/documentation/axi_ip_documentation.htm => UG761

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);
```

Video Timing Controller PCORE (v5.01.a)

Version 5.01.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.

AXIVDMA Driver (v4.02.a) – Cannot set MM2S GenLockSource to '1'

The AXIVDMA driver has a function to set the MM2S's GenLockSource bit.

```
XAxiVdma_GenLockSourceSelect(
      InstancePtr, XAXIVDMA_INTERNAL_GENLOCK, XAXIVDMA_READ );
```

However, this function does not work.

As a workaround, the example code provided with this tutorial manually performs a read-modify-write operation to set the bit to '1', corresponding to the Internal GenLock source.

AXI_MONITOR - Could not connect AXI MONITOR's RESET port !

When adding an AXI MONITOR on an AXI4-Stream for video interface, you may get an error similar to the following:

```
ERROR: EDK – Could not connect MONITOR chipscope_axi_monitor_0's
RESET MON_AXI_ARESETN to INSTANCE v_vid_in_axi4s_0's RESET PORT
as the latter is itself un-connected.
```

This error occurs because the Chipscope AXI Monitor clock and reset ports are associated with the AXI4-Stream clock and reset ports. If these ports do not exist for the AXI4-Stream interface, the AXI Monitor is not able to populate with a clock and reset connection.

There is a simple workaround, that involves connecting the AXI_MON_ACLK and AXI_MON_ARESETN ports manually in the MHS file.

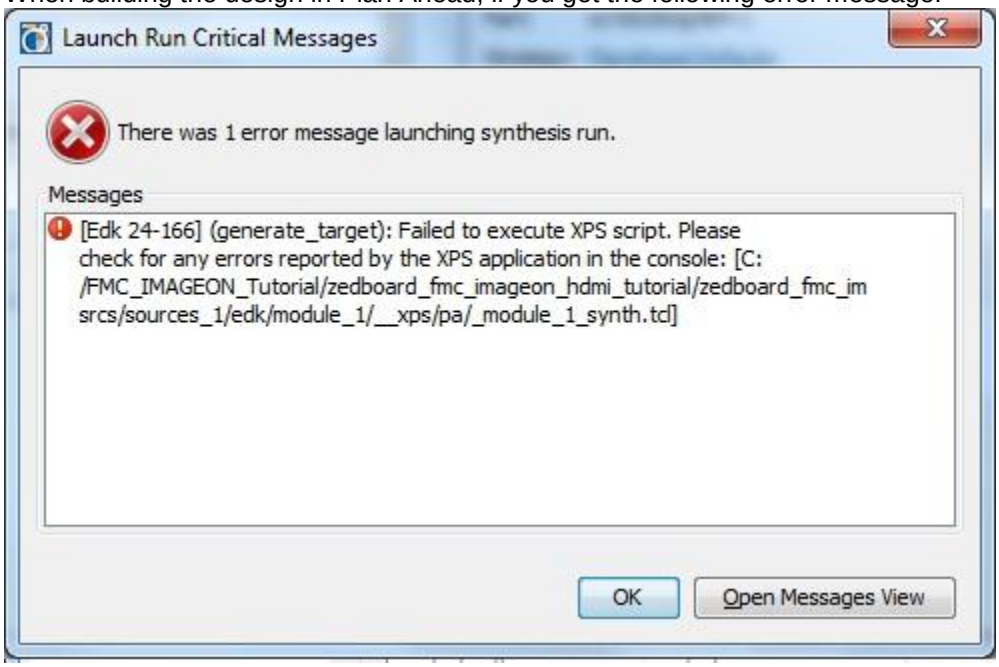
For example:

```
BEGIN chipscope_axi_monitor
  PARAMETER INSTANCE = chipscope_axi_monitor_0
  PARAMETER HW_VER = 3.05.a
  PARAMETER C_USE_INTERFACE = 1
  PARAMETER C_NUM_DATA_SAMPLES = 4096
  BUS_INTERFACE MON_AXI_S = v_vid_in_axi4s_0_M_AXIS_VIDEO
  PORT CHIPSCOPE_ICON_CONTROL = chipscope_icon_0_control0
  PORT mon_axi_aclk = processing_system7_0_FCLK_CLK1
  PORT mon_axi_aresetn = net_vcc
END
```

Troubleshooting

[Edk 24-166] (generate_target): Failed to execute XPS Script

When building the design in Plan Ahead, if you get the following error message:



It indicates that an error was encountered while building the embedded sub-system.

To get a more descriptive message, open the embedded sub-system,

and select **Hardware** => **Generate Netlist**.

Once the error is fixed, close XPS and re-attempt to build in Plan Ahead.

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

If you get the following output from the serial console:

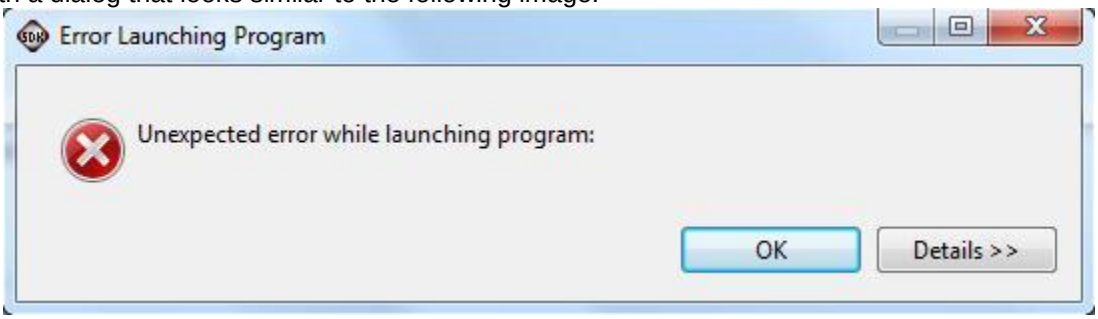
```
Hello World

------------------------------------------------------
--          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.

SDK - Error Launching Program

When launching a program for execution on the remote target, an error message is encountered with a dialog that looks similar to the following image:



The SDK console may also output an error message similar to the following.

```
10:52:09 ERROR : Unexpected error while launching program. java.lang.RuntimeException:
Error while running xzynqresetstatus:
        at com.xilinx.sdk.targetmanager.internal.TM.resetProcessor(Unknown Source)
        at com.xilinx.sdk.debug.core.internal.AppRunner.run(Unknown Source)
```

Cause:

Application launched to the target while the Boot Mode jumpers are set to SD card boot mode rather than JTAG boot mode as indicated in tutorial instructions.

Corrective Action:

1. Adjust the Boot Mode jumpers for JTAG boot mode.
2. Close the SDK application.
3. Open Windows Task Manager.
4. Locate all instances of **eclipse.exe** in the **Processes** tab and use the **End Process** button to terminate process execution.
5. Locate all instances of **javaw.exe** in the **Processes** tab and use the **End Process** button to terminate process execution.

6. Restart SDK and resume the lab activity from the point where the process is launched.

AXI4S to Video Out – locked port never asserts

When connected to the AXI_VDMA core, the AXI4S to Video Out core's locked port will not assert is the AXI_VDMA's MM2S channel is configured to use FSYNC.

When connected to the AXI4S to Video Out core, the AXI_VDMA's MM2S channel must be configure for free run, which means streaming mode.

AXI_VDMA – Video has vertical jitter

When using the AXI_VDMA core, vertical jitter may be caused by End of Line (EOL) errors. This can be.

This symptom has been observed with this design when the AXI_VDMA's "Maximum Burst Size" parameters are set to 256.

To resolve the issue, set the AXI_VDMA's "Maximum Burst Size" to 16.

AXI_VDMA – Video has horizontal jitter

When using the AXI_VDMA core, horizontal jitter may occur if the core is not configured correctly.

This symptom has been observed with this design when the AXI_VDMA's "Line Buffer Depth" was set to 2048.

To resolve the issue, set the AXI_VDMA's "Line Buffer Depth" to 4096, which corresponds to the next power of 2 above the active line size in bytes.