ZedBoard
HDMI Display Controller Tutorial

Version 1.3

Revision History

Version	Description	Date
1.0	HDMI Display Controller Tutorial • Vivado 2013.3 version	Mar 25, 2014
1.1	Fix typo in S_AXI_HP0_ACLK connection instructions Add instructions to assign address to axi_vdma_0 core TCL script updated to configure AXI4-Stream to Video Out core for Master timing mode. Fix typo in XDC constraints file.	Jun 23, 2014
1.2	Port to Vivado 2013.4	Jun 23, 2014
1.3	Port to Vivado 2014.1	Jun 23, 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 the hardware and software required for this tutorial.
- Chapter "**Tutorial Overview**" provides a general overview of this tutorial.
- Chapter "**Implement an HDMI Display Controller**" describes the steps required to implement an Embedded System that implements an HDMI Display Controller Tutorial.
- Appendix "**References**" provides a list of references to documentation related to the Video IP cores.
- 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

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 2013.4
- 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]
- ZedBoard (including power supply and cables)
- HDMI (or DVI-D) monitor, 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 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

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:\AVNET_ZED_HDMI\2014_1\avnet_zed_hdmi_core
C:\AVNET_ZED_HDMI\2014_1\constraints
C:\AVNET_ZED_HDMI\2014_1\code
C:\AVNET_ZED_HDMI\2014_1\scripts

Tutorial Overview

This tutorial will guide you in creating a video design that implements an HDMI display controller for the ZedBoard.

Reusable Components

The tutorial will make use of reusable components:

- IP Core
    - ZedBoard – HDMI Output

- TCL script : automatically create IP Integrator sub-modules
    - **zed_hdmi_display.tcl** : video output path, 24 bits RGB format

- XDC constraints : defines pinout and constraints for various carriers
    - **zedboad_hdmi_display.xdc** : constraints for ZedBoard

- C source code : provides example initialization code
    - **zed_hdmi_display.c/.h** : init code for HDMI display controller

HDMI Display Controller Overview

The HDMI display controller will make use of the AXI Video DMA IP core, as shown below.

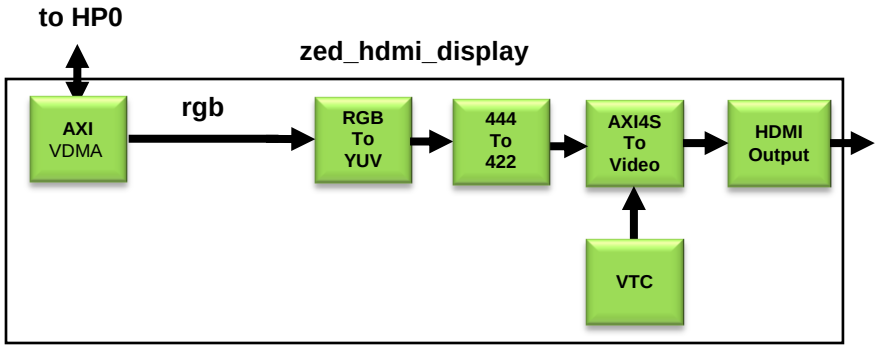

Figure 1 – HDMI Display Controller

The AXI Video DMA core will fetch video content from the Zynq's external memory via one of the High-Performance Ports (HP0), and produce video content in the AXI4-Stream protocol.

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.

HDMI 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 following hardware peripheral:
    - ADV7511 : HDMI output device

HDMI Video Interface

The following core, provided with the tutorial, will be used to interface to the ADV7511 device on the ZedBoard. It is important to note that, on the ZedBoard, the ADV7511 device is used in 16 bits YCbCr 4:2:2 mode, with external syncs.

- ZED HDMI Output
    - this core contains minimal logic for the 16 bit video data sent to the ADV7511 HDMI output device.

The HDMI Output IP core makes a generic parallel video interface available to the design.

AXI4-Stream Bridges

The following cores are used to bridge between the HDMI interface 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:
        - to generate the video timing required for the HDMI output interface.
- 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

Implement an HDMI Display Controller

In this section, a new Vivado project will be created, implementing the HDMI Display Controller 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 2013.4 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:\AVNET_ZED_HDMI\2014_1
	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 the following board: **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.

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**.

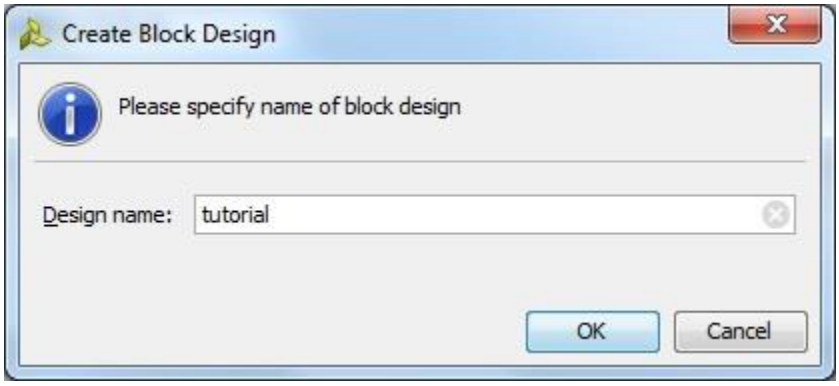

Figure 2 – Create Block Design

Notice that Vivado IP Integrator provides design assistance:

Figure 3 – 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:

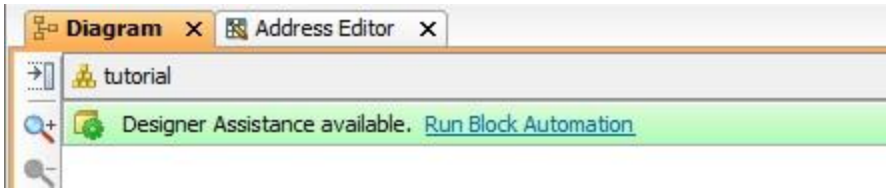

Figure 4 – Designer Assistance – Run Block Automation

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**.

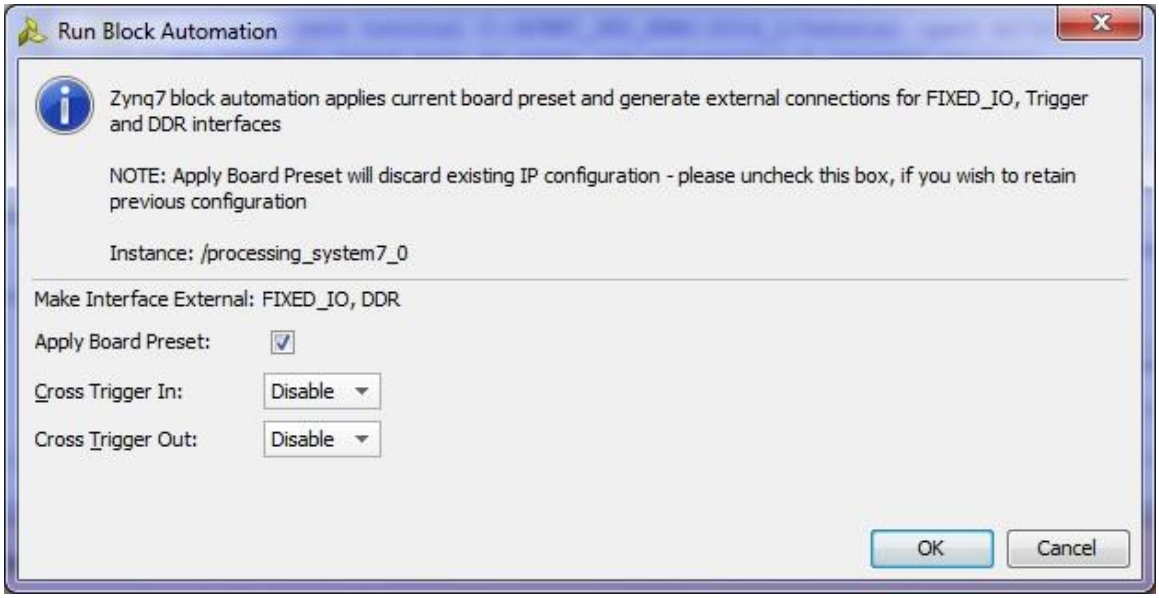

Figure 5 –Run Block Automation

Add the Video and AXI4-Stream clocks

This video design will use two video related clocks:

- 150MHz : clock for AXI4-Stream based interconnect and IP cores
- 148.5 MHz : clock for the HDMI output interface

Creating a separate clock for the HDMI output interface allows it to be eventually changed for different video resolution, without having to change the AXI4-Stream clock.

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.

In addition, generate a 100MHz clock on its FCLK_CLK2 port, which will be used to derive the 148.5MHz clock used for the HDMI output interface.

1. Double-click on the ZYNQ7 Processing System core.
2. Click on the **Clock Generation** block
3. Expand the **PL Fabric Clocks** section
4. For the **FCLK_CLK0** clock
5. Specify a **Requested Frequency** of **75MHz**.
6. Select the **FCLK_CLK1** clock
7. Specify a **Requested Frequency** of **150MHz**.
8. Select the **FCLK_CLK2** clock
9. Specify a **Requested Frequency** of **100MHz**.

The FCLK_RESET1_N port will also be enabled, providing a reset for the FCLK_CLK1 clock domain.

10. In the **Page Navigator** on the left, select the **PS-PL Configuration**
11. Expand the **General** section
12. Expand the **Enable Clock Resets** sub-section
13. Select the **FCLK_RESET1_N** reset
14. Click **OK**.

Add a the **Clocking Wizard** IP core to the block diagram, and configure it to use the 100MHz FCLK_CLK2 clock to generate a 148.5 MHz clock.

15. **Right-click** in a blank portion of the block design, then select **Add IP**.
16. Select the **Clocking Wizard** core,
    then click **ENTER** (or double-click selection)

17. **Double-click** on the Clocking Wizard core.
18. Under the **Clocking Options** tab:
    a. Specify **100.0** MHz for the **Primary Input Clock** frequency
19. Under the **Output Clocks** tab:
    a. Specify **148.5** MHz for the **clk_out1** output clock
    b. De-select the reset port
    c. De-select the locked port

Connect the Clocking Wizard core's **clk_in1** port to the ZYNQ7's **FCLK_CLK2** port.

20. Select the Clocking Wizard core's **clk_in1** port
21. Click and hold the left mouse button, then drag to the ZYNQ7's **FCLK_CLK2** port.
22. Release the mouse button to make the connection

Add the HDMI I2C Controller

Add the **AXI IIC** IP core to the design, which will be used to configure the HDMI output interface (ADV7511) via I2C.

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 zed_hdmi_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 **zed_hdmi_iic_0**

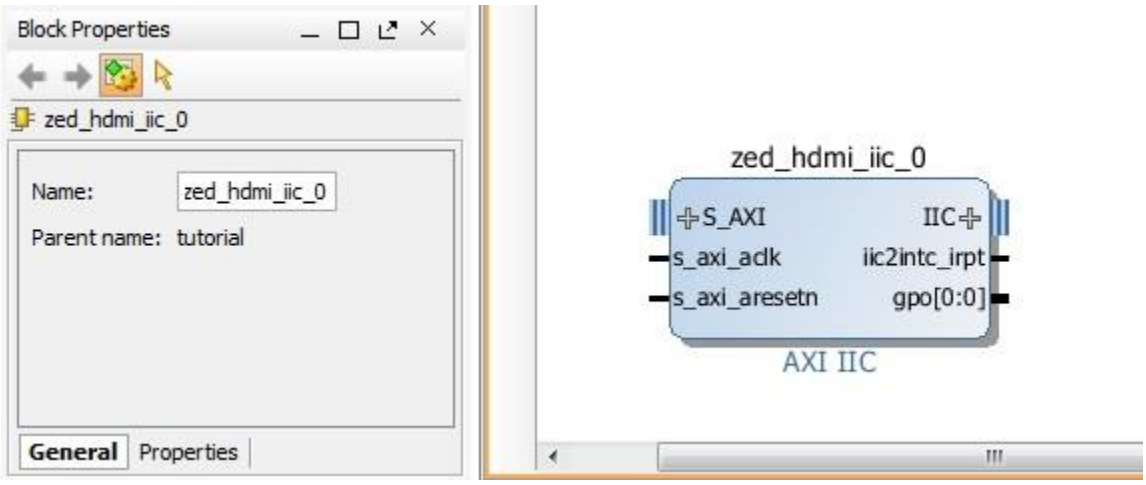

Figure 6 – AXI IIC - Block Properties

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 **/zed_hdmi_iic_0/S_AXI**
7. The Run Connection Automation dialog will appear
8. For the **Clock Connection**, select **/processing_system7_0/FCLK_CLK0**
9. Click **OK**.

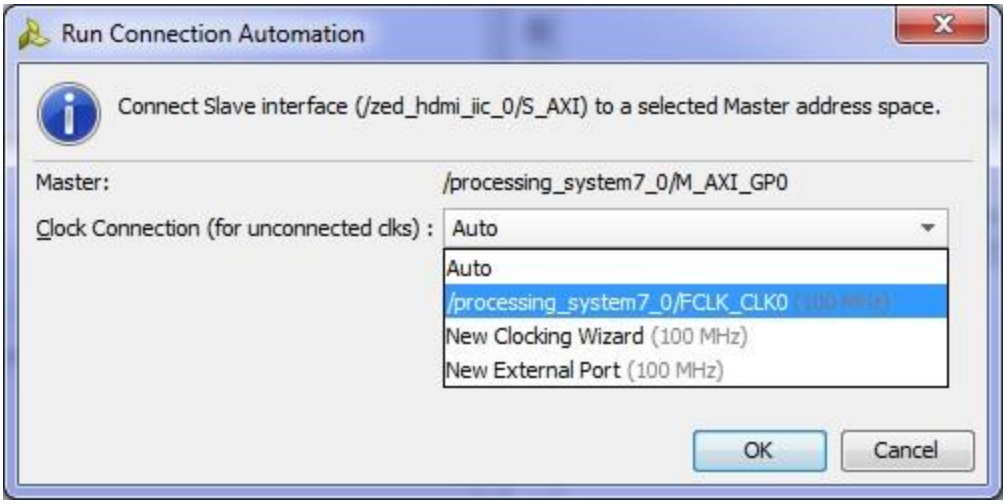

Figure 7 – Run Connection Automation – /zed_hdmi_iic_0/S_AXI

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

10. Click **Run Connection Automation** in the design assistance
11. Select **/zed_hdmi_iic_0/IIC**
12. [If the **Select Board Interface** drop down list appears, specify **Custom**]
13. Click **OK**.

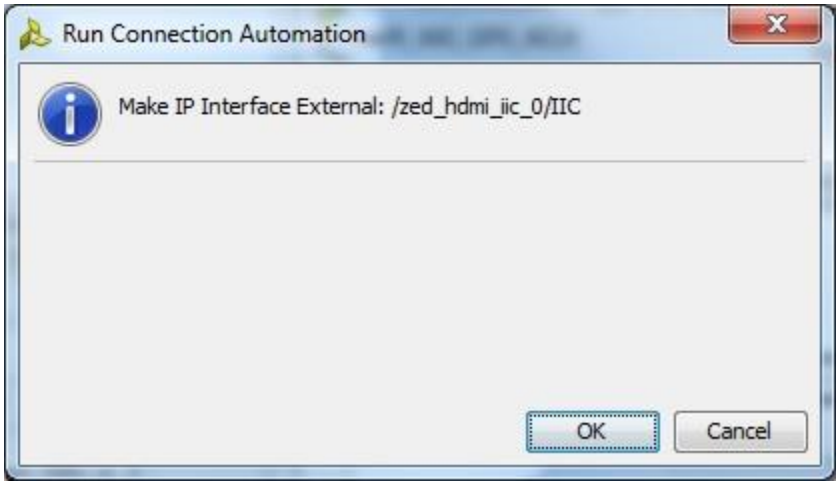

Figure 8 – Run Connection Automation – /zed_hdmi_iic_0/IIC

Rename the external iic_rtl port to zed_hdmi_iic

14. Select the **iic_rtl** external port in the block diagram.
15. In the **External Interface Properties** dialog, modify the **Name** of the block to **zed_hdmi_iic**

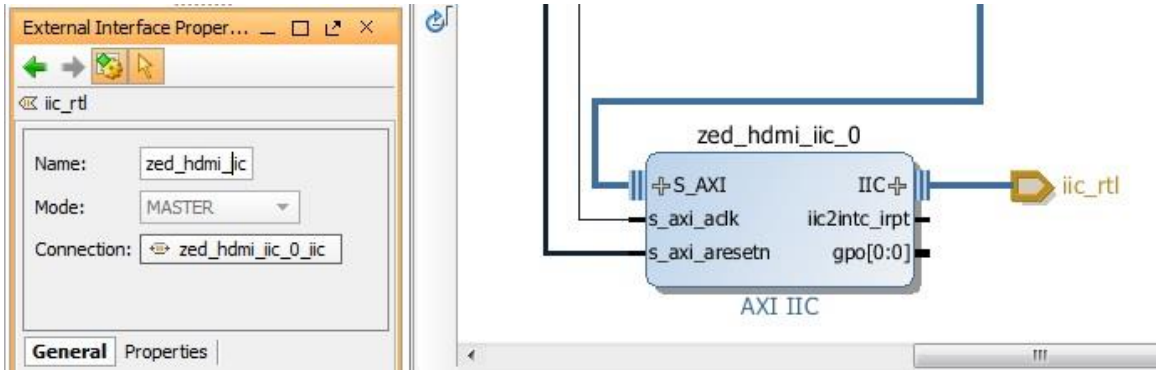

Figure 9 – External Interface Properties – zed_hdmi_iic

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

16. **Right-click** in a blank portion of the block design, then select **Validate Design**
17. If successful, Click **OK** and proceed with the next section.
    Otherwise, fix any errors in the design before proceeding.

Figure 10 – Validate Design

Add the Video Output Path

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 ZED HDMI core 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:\AVNET_ZED_HDMI\2014_1\avnet_zed_hdmi_core**" directory,
   then click the **Select** button.

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

- ZedBoard - HDMI Output

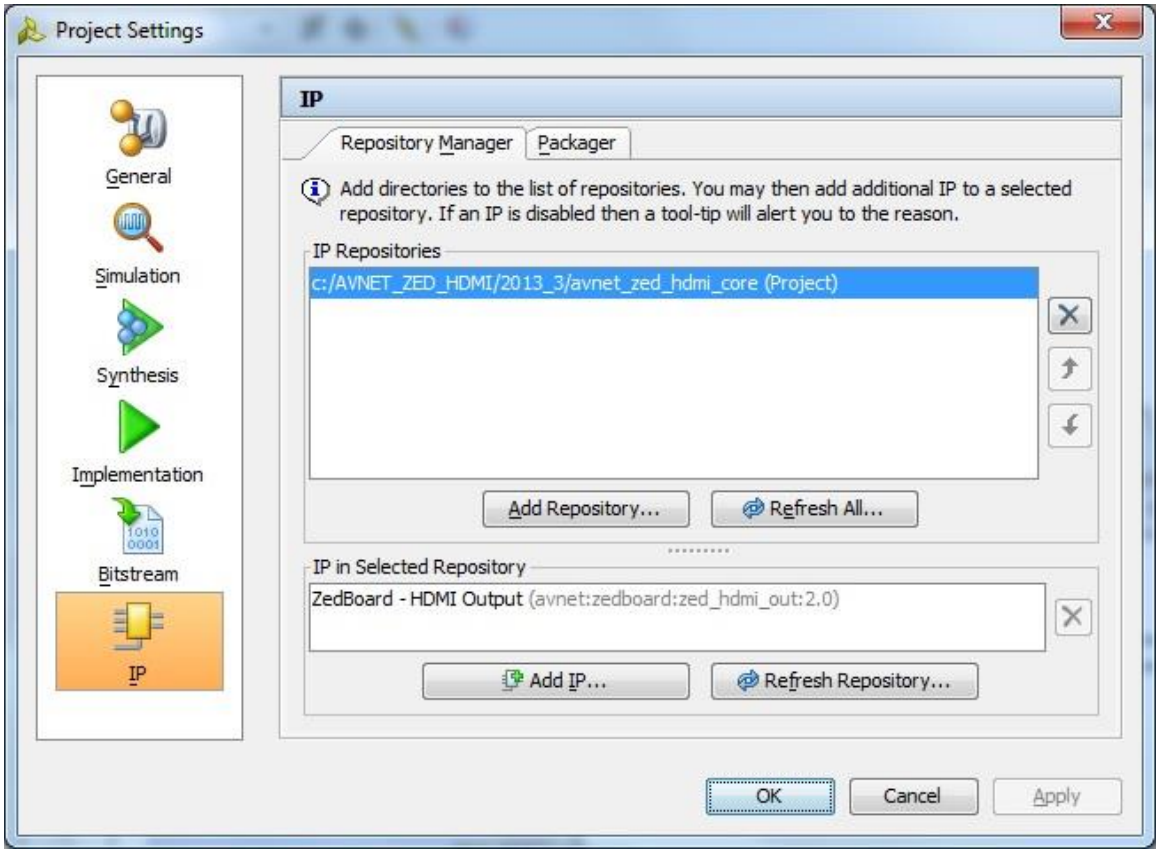

Figure 11 – IP Repository Manager

5. Click **OK**.

The display controller will work in the 24bits RGB color space. The following TCL script will be used to create the video output path, including the HDMI output and Video DMA cores:

- **zed_hdmi_display.tcl** : video output path, 24 bits RGB format

Use the TCL console to run the TCL script.

6. In the **Tcl Console**, enter the following command to generate the video output path.
   source C:/AVNET_ZED_HDMI/2014_1/scripts/zed_hdmi_display.tcl
   then press {ENTER}

This will create one new video sub-module in your block diagram.

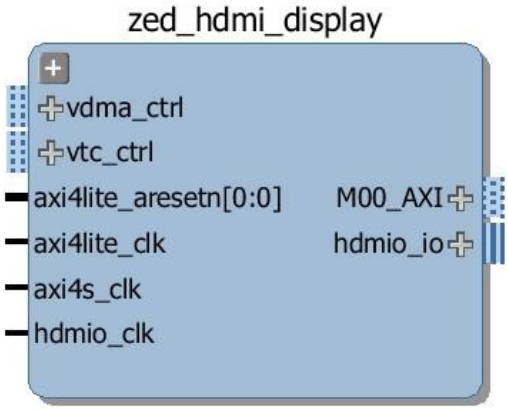

Figure 12 – ZED HDMI display sub-modules

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

7. Select the zed_hdmi_display sub-module's **axi4lite_ clk** port
8. Click and hold the left mouse button, then drag to the ZYNQ7's **FCLK_CLK0** port.
9. Release the mouse button to make the connection

Connect the AXI4-Lite resets to the rst_processing_system7_0_###M core's peripheral_aresetn port.

10. Select the zed_hdmi_display sub-module's **axi4lite_aresetn** port
11. Click and hold the left mouse button, then drag to the rst_processing_system7_0_###M core's **peripheral_aresetn[0:0]** port.
12. Release the mouse button to make the connection

Use the designer assistance to connect the VTC core's and the VDMA cores's control port (vtc_ctrl & vdma_ctrl) to the ZYNQ7 Processing System's GP0 port.

13. Click **Run Connection Automation** in the design assistance
14. Select **/zed_hdmi_display/vtc_ctrl**
15. The Run Connection Automation dialog will appear
16. For the **Clock Connection**, select **/processing_system7_0/FCLK_CLK0**
17. Click **OK**.

18. Click **Run Connection Automation** in the design assistance
19. Select **/zed_hdmi_display/vdma_ctrl**
20. The Run Connection Automation dialog will appear
21. For the **Clock Connection**, select **/processing_system7_0/FCLK_CLK0**
22. Click **OK**.

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

23. Select the zed_hdmi_display sub-module's **axi4s_clk** port
24. Click and hold the left mouse button, then drag to the ZYNQ7's **FCLK_CLK1** port.

25. Release the mouse button to make the connection.

Connect the AXI4-Stream reset to the ZYNQ7 Processing System's FCLK_RESET1_N port.

1. Select the zed_hdmi_display sub-module's **axi4s_resetn** port
2. Click and hold the left mouse button, then drag to the ZYNQ7's **FCLK_RESET1_N** port.
3. Release the mouse button to make the connection.

Connect the HDMI output clock to the Clocking Wizard's 148.5MHz clock.

4. **Select** the zed_hdmi_display sub-module's **hdmio_clk** port
5. Click and hold the left mouse button, then drag to the clk_wiz_0 core's **clk_out1** port.
6. Release the mouse button to make the connection.

Configure the ZYNQ7 Processing System to enable the HP0 high-performance port.

7. Double-click on the ZYNQ7 Processing System core.
8. Click on the **High Performance AXI 32b/64b Slave Ports** block
9. Expand the **HP Slave AXI Interface** section
10. Select the **S AXI HP0 interface** port
11. Click **OK**.

Connect the HP0 clock to the ZYNQ7 Processing System's FCLK_CLK1 port.

12. Select the ZYNQ7's **S_AXI_HP0_ACLK** port
13. Click and hold the left mouse button, then drag to the ZYNQ7's **FCLK_CLK1** port.
14. Release the mouse button to make the connection.

Connect the zed_hdmi_display sub-module's **M00_AXI** port to the ZYNQ7's **S_AXI_HP0** port.

15. Select the zed_hdmi_display sub-module's **M00_AXI** port
16. Click and hold the left mouse button, then drag to the ZYNQ7's **S_AXI_HP0** port.
17. Release the mouse button to make the connection.

Connect the HDMI I/O ports (hdmio_io) to external ports

18. **Select** the zed_hdmi_display sub-module's **hdmio_io** port
19. **Right-click**, then select **Make External**

In the Address Editor tab, specify an address (Auto Assign Address) for the zed_hdmi_display/axi_vdma_0 core's HP0_DDR_LOWOCM address space.

1. Select the Address Editor tab
2. In the Address Editor, Expand the **zed_hdmi_display_0/axi_vdma_0** item
3. Expand the **Data_MM2S** item
4. Expand the **Unmapped Slaves** item
5. **Right-click** on the **processing_system7_0** item, then select **Assign Address**

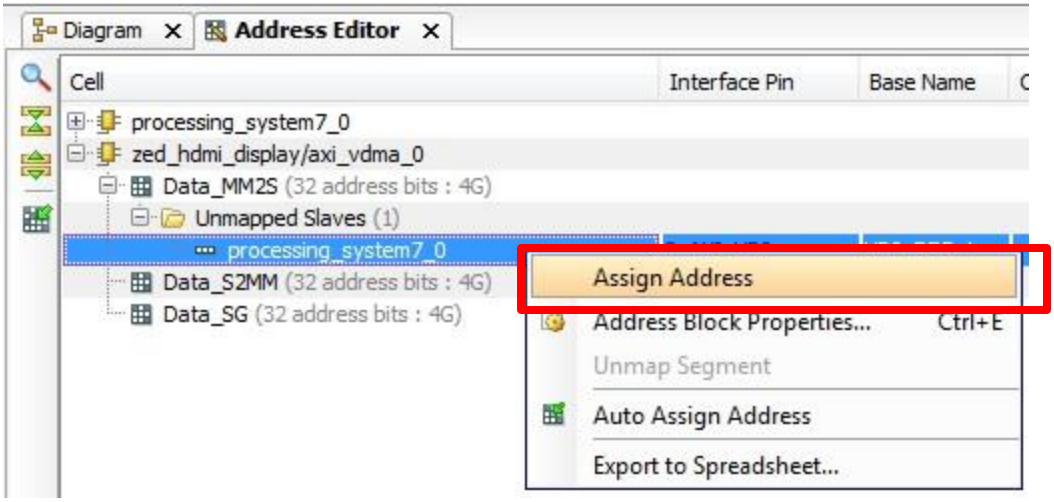

Figure 13 – Assign Address in Address Editor

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

20. **Right-click** in a blank portion of the block design, then select **Validate Design**
21. If successful, Click **OK** and proceed with the next section.
    Otherwise, fix any errors in the design before proceeding.

Figure 14 – Validate Design

Save the block diagram

22. 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**

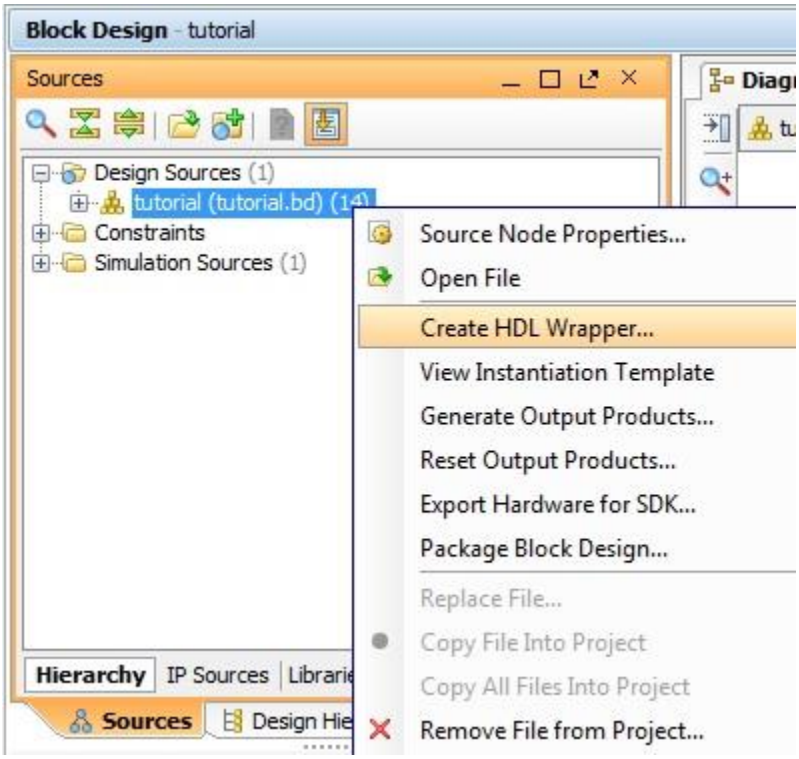

Figure 15 – Create HDL Wrapper

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

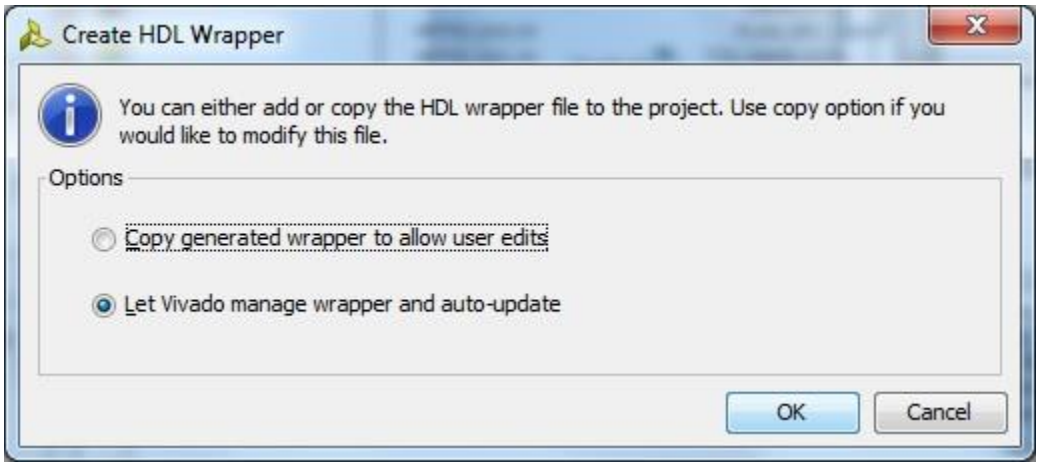

Figure 16 – Let Vivado manage wrapper and auto-update

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 the following XDC file:
    a. ..\constraints\zedboard_hdmi_display.xdc
9. Once selected, click **OK**
10. Click **Finish**.

Build the bitstream.

11. 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.
12. Click **Yes**.

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

13. Select **Open Implemented Design**
14. 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.

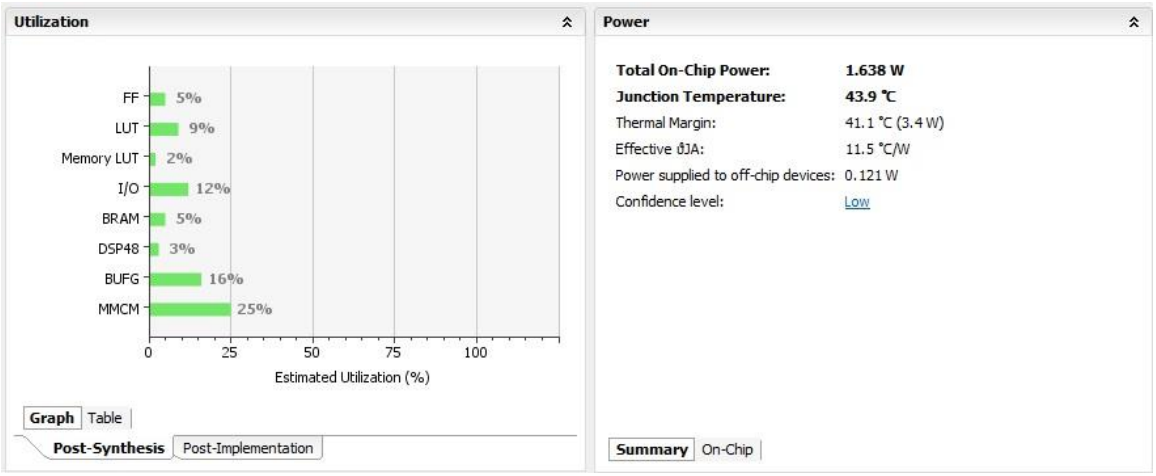

Figure 17 – HDMI Display Controller – Resource Utilization

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.

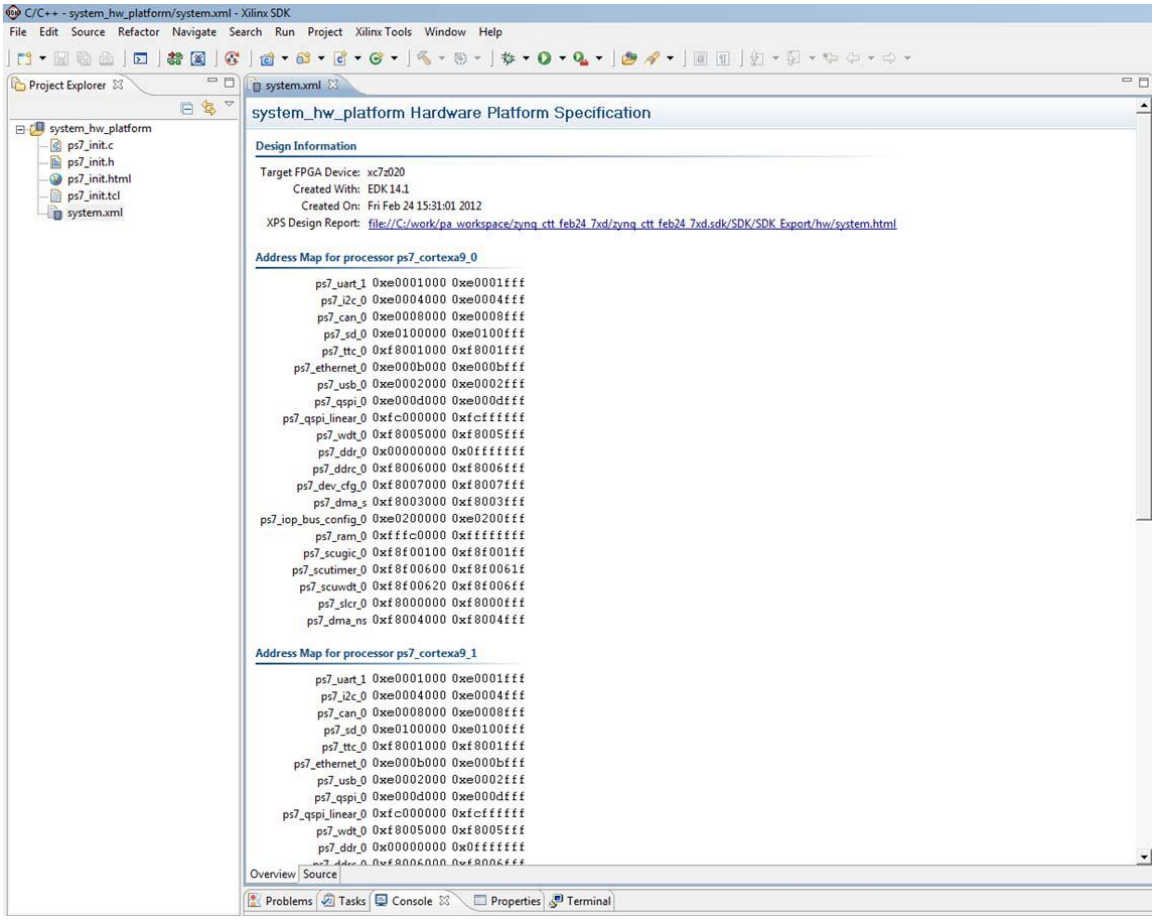

Figure 18 – Address Map in SDK system.xml Tab

Create a standalone BSP (board support package).

4. In the SDK menu, select **File** => **New** => **Board Support Package**.
   The New Board Support Package Project dialog box opens.
5. In the **Project name** field, type “**hdmi_display_bsp**”.
6. Keep the default settings, and click **Finish**.
   The Board Support Package Settings dialog box opens.

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

Create a new C project.

8. In the SDK menu, select **File** => **New** => **Application Project**.
   The Application Project dialog box opens.
9. In the **Project Name** field, type "**hdmi_display_app**".
10. For the **Board Support Package**, select **Use Existing**, then select the BSP that was created previously.
11. Click **Next**.
    The Templates dialog box opens.
12. Select the **Hello World** template.
13. Click **Finish**.

Import the provided example source files for the hdmi_display.

1. In the Project Explorer window, select the **hdmi_display_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**.

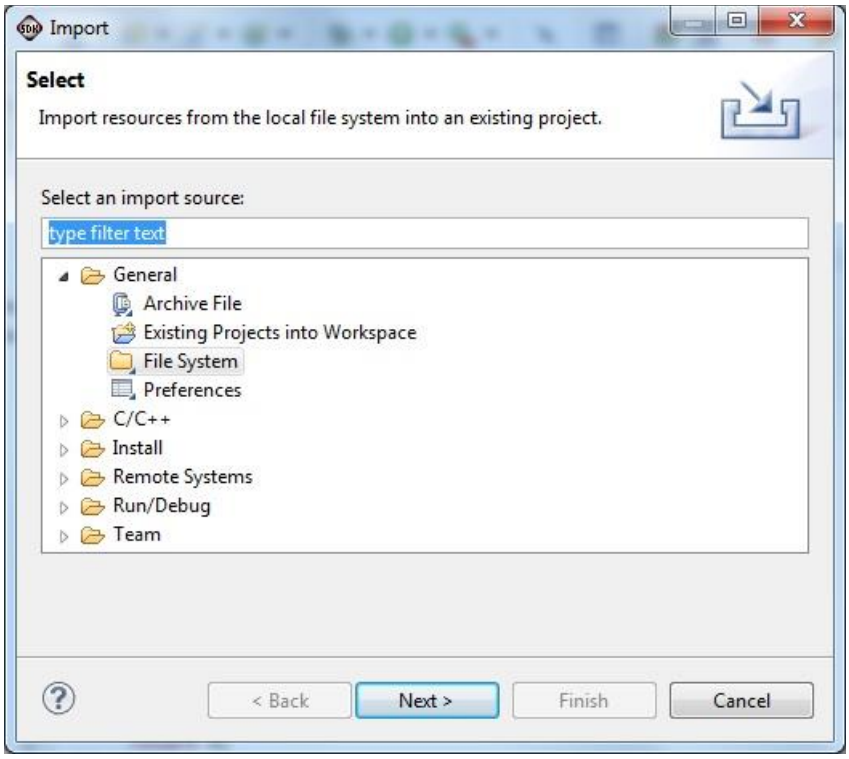

Figure 19 – Import from File System – Dialog 1

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:\AVNET_ZED_HDMI\2014_1\code\zed_hdmi_display
7. Click **OK**

8. Select the following source files:
   video_frame_buffer.c
   video_frame_buffer.h
   video_generation.c
   video_generation.h
   video_resolution.c
   video_resolution.h
   zed_hdmi_display.c
   zed_hdmi_display.h
   zed_iic_axi.c
   zed_iic.h

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

Figure 20 – Import into Folder

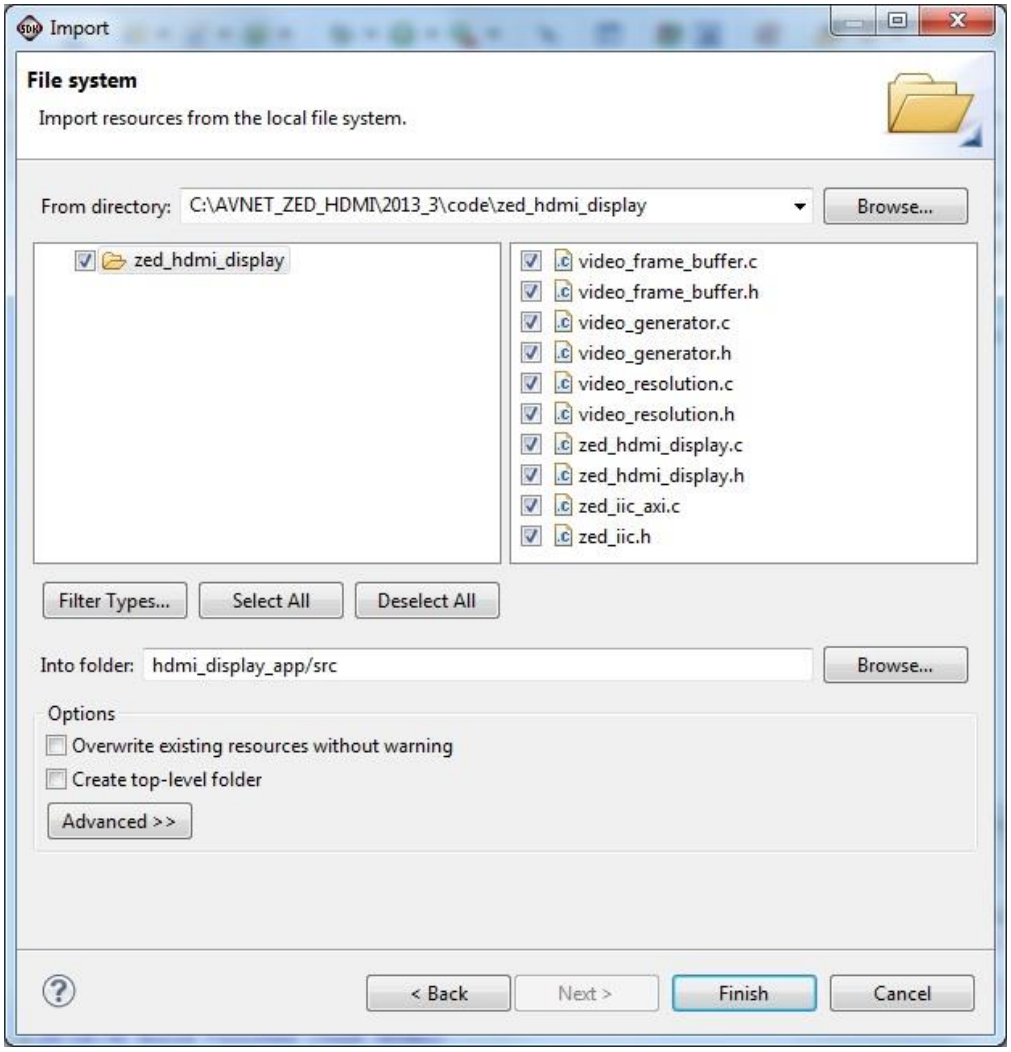

Figure 21 – Import from File System – Dialog 2

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 "zed_hdmi_display.h"
zed_hdmi_display_t demo;

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

int main()
{
    init_platform();

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

    demo.uBaseAddr_IIC_HdmiOut = XPAR_ZED_HDMI_IIC_0_BASEADDR;
    demo.uDeviceId_VTC_HdmioGenerator =
        XPAR_ZED_HDMI_DISPLAY_V_TC_0_DEVICE_ID;
    demo.uDeviceId_VDMA_HdmiDisplay = XPAR_ZED_HDMI_DISPLAY_AXI_VDMA_0_DEVICE_ID;
    demo.uBaseAddr_MEM_HdmiDisplay = XPAR_DDR_MEM_BASEADDR + 0x10000000;
    demo.uNumFrames_HdmiDisplay = XPAR_AXIVDMA_0_NUM_FSTORES;
    zed_hdmi_display_init( &demo );

    return 0;
}
```

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

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 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. Connect a DVI or HDMI monitor to the ZedBoard's HDMI OUT connector (J9)
5. Power on the ZedBoard board
6. 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 Display Controller 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 **hdmi_display_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_display_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

------------------------------------------------------
--        ZedBoard HDMI Display Controller          --
------------------------------------------------------

HDMI IIC Initialization ...
HDMI Output Initialization ...
Clear Frame Buffer
Video Frame Buffer Initialization ...
Video DMA (Output Side) Initialization ...
Video Timing Controller (generator) Initialization ...
        Video Resolution = 1080P
Generate Color Bars
HDMI Output Re-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 display controller on hardware !

References

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

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

The following reference provides links to documentation for AXI interconnect.

6. UG761 - AXI Reference Guide
7. 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