

Integrating Custom Logic in Camera Frame Buffer Demo

Application Note

Author: Jeff Olsen

Date: 11/19/2010

EXECUTIVE SUMMARY

The Xilinx Spartan 3A-3400 Xtreme DSP Video Starter Kit (VSK) provides users the ability to develop a variety of video and image processing applications on a field programmable gate array (FPGA). The VSK includes demos that display the capabilities of the FPGA and give the user a reference point for developing custom applications using Xilinx's Embedded Development Kit (EDK) and System Generator. This application note will focus on implementing custom logic in the Camera Frame Buffer Demo.

KEYWORDS: VSK, FPGA, EDK, System Generator

Table of Contents

Introduction	3
Objective	3
Resources	3
Background.....	4
System Generator	4
Xilinx Platform Studio.....	4
Directory Structure.....	4
Implementation.....	5
Modifying Camera Frame Buffer Demo	5
Generating Pcore	6
Opening the Project in XPS.....	8
Conclusion	11
References.....	12

Introduction

FPGAs are used in a variety of applications that include digital signal processing, computer vision, speech recognition, medical imaging and many other areas. They are comprised of an array of configurable logic blocks and reconfigurable interconnects that allow the blocks to be connected to each other. FPGAs are programmed using a hardware description language (HDL). Xilinx has its own set of software tools that assist the user in the development process of their FPGA application without having to learn a new language.

Objective

The objective of this application note is to demonstrate how to implement custom logic into the Camera Frame Buffer Demo reference design included with the Xilinx Spartan 3A-3400 Xtreme DSP Video Starter Kit. It will emphasize the process of using System Generator and the Xilinx EDK to implement custom logic rather than focus on the logic that will be implemented. The concepts demonstrated in this application note can then be used to implement custom logic specific to the users requirements.

Resources

This application note uses the following resources:

- Xilinx Spartan 3A-3400 Xtreme DSP Video Starter Kit
- Xilinx ISE Design Suite 11.4 software
- System Generator 11.4
- MATLAB 2008a with Simulink
- VSK Reference Designs v3.0

Note: This application note assumes that the reader understands how to correctly set up the development board and run the preloaded demos. It is recommended that the reader review Spartan-3A DSP FPGA Video Starter Kit Assembly by Pascha Grant if unfamiliar with this process.

Background

System Generator

System Generator is a software tool developed by Xilinx that provides model based FPGA design in the Simulink environment. The models are built with the Xilinx provided blockset. With the click of a button, System Generator will automatically generate the files necessary to program the FPGA.

Xilinx Platform Studio

Xilinx Platform Studio (XPS) is a software tool used for the creation of a complete embedded processor system for a Xilinx FPGA. Through the graphical user interface the user connects the processor to the Xilinx provided peripherals or custom peripherals created by the user. XPS also provides a simulation environment to test the embedded processor design.

Directory Structure

The EDK_Demonstrations directory is the main directory for all reference designs. It is important to be familiar with these directories and files before implementing custom logic in any of the reference designs. The following directories and files will be referenced throughout this application note:

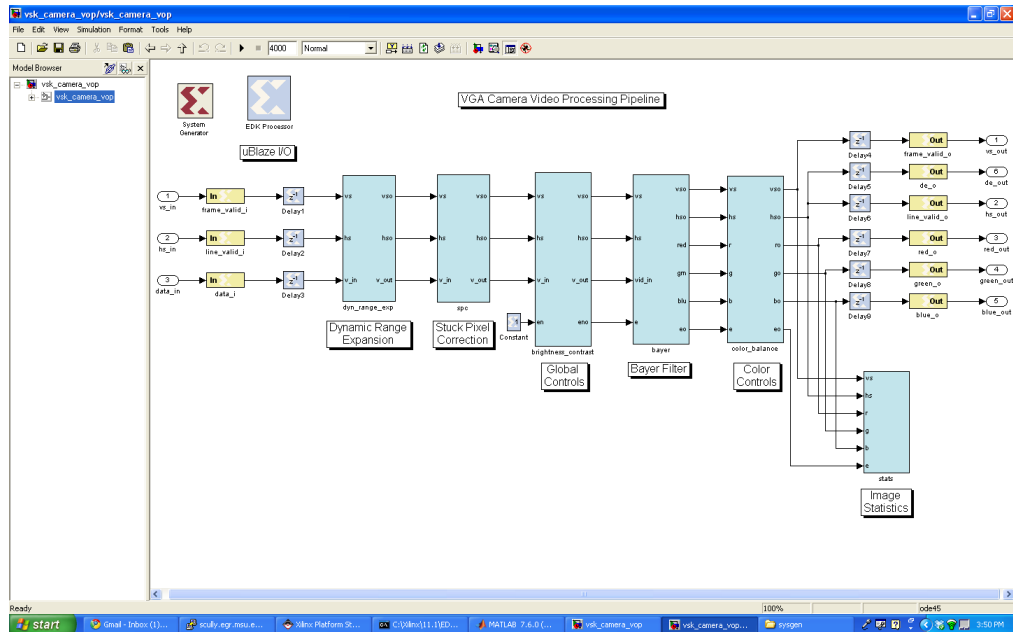
\Camera_Frame_Buffer_Demo	Main folder for demo
\pcores	Location of pcores generated in system generator
system.xmp	XPS project design file
download.bit	Hardware configuration bit file
executable.elf	Software that runs on MicroBlaze

Note: The reference designs can be downloaded on Xilinx's website at <http://www.xilinx.com/support/documentation/do-s3adsp-video-sk-uni-g.htm>

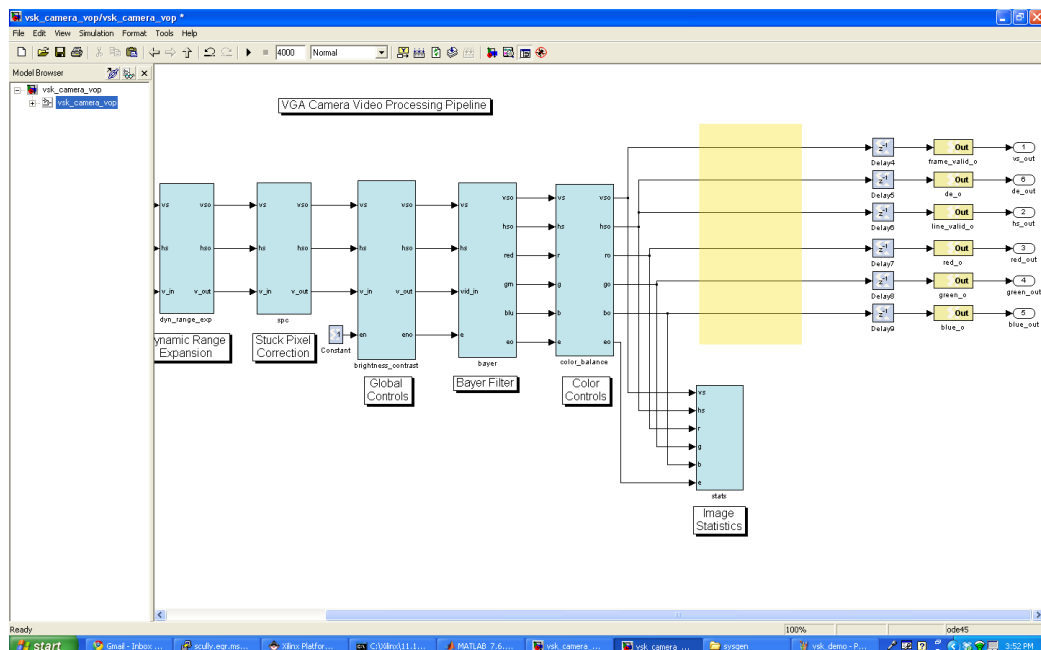
Implementation

Modifying Camera Frame Buffer Demo

1. Open the following system generator model located in the pc core directory
\\vsk_camera_vop_plbw_v3_00_b\\sysgen\\vsk_camera_vop.mdl



2. Custom logic will be placed after the Color Correction block in the highlighted area shown in the diagram below.

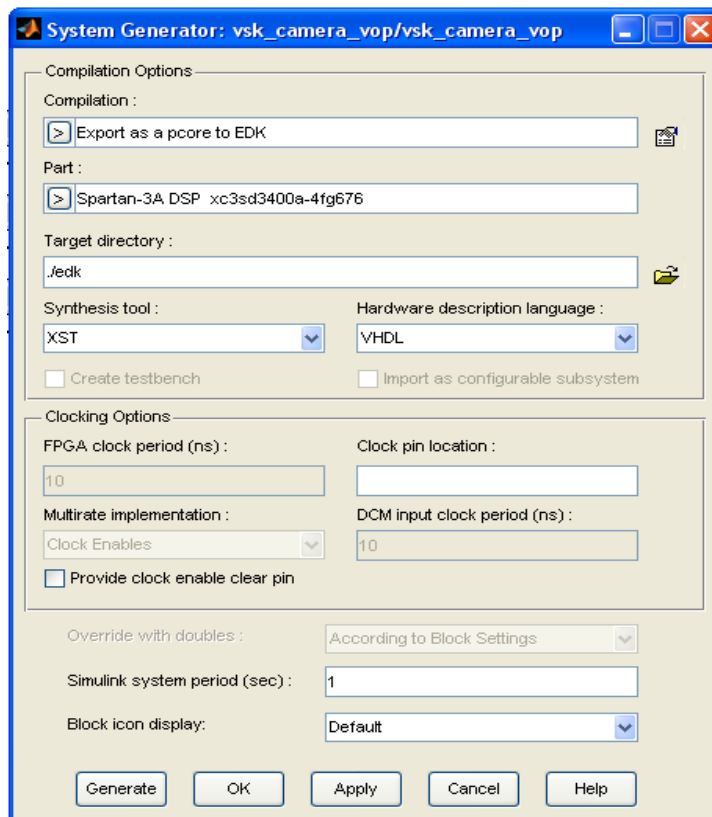


The Camera Frame Buffer Demo divides the camera signal into a Red, Green, and Blue (RGB) component respectively. Adding custom logic to the RGB signals will modify the camera stream according. It is also important to be sure to synchronize the vsync and hsync with the proper amount of delay that the custom logic inserts into the system. For an example of custom logic that can be implemented into the camera stream reference “Grayscale Conversion of a Colored Image Using Simulink and Xilinx Blocks” by Emmett Kuhn.

Generating Pcore

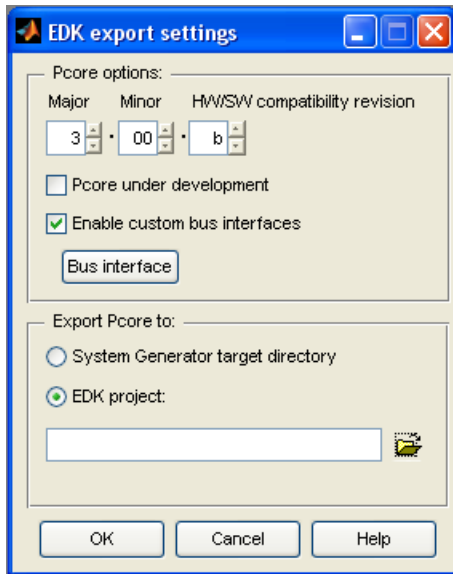
Once the custom logic has been inserted into the demo, the next step is generating the PCORE that will be integrated in the embedded system design in XPS. System Generator will take care of creating all the necessary files in a few simple steps.

1. Double-click the System Generator token in the model to bring up the following menu:



2. Under compilation make sure the Export as a pcore to EDK option is selected.

3. At the end of the line, click the icon with the finger pointing. This will bring up a new menu to select the project where the Pcore will be exported.

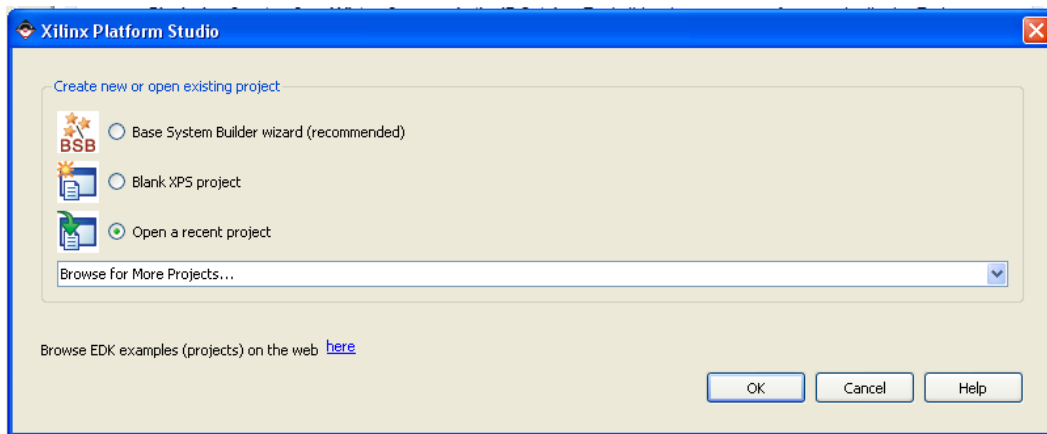


4. If not already selected, select the EDK project radio button under Export Pcore to. Click the folder to browse for the project. Select the system.xmp file referenced in the Directory Structure of the Background section. When the file is found select OK.
5. The next step is to specify the target directory for the Pcore. In this case we will use the pcore directory referenced in the Directory Structure of the Background section. Selecting the same directory as the original Pcore used in the demo will save the need to connect the Pcore to the proper buses once the project is opened in XPS, as it has already been done in the original version of the demo.
6. Once the above steps have been complete the Pcore can now be generated. Clicking on the Generate button will start this process. A pop up window will appear to let you know the Pcore generation is complete. The Pcore will not generate if your custom logic produces any errors. If this happens, follow the errors to the source and fix according.

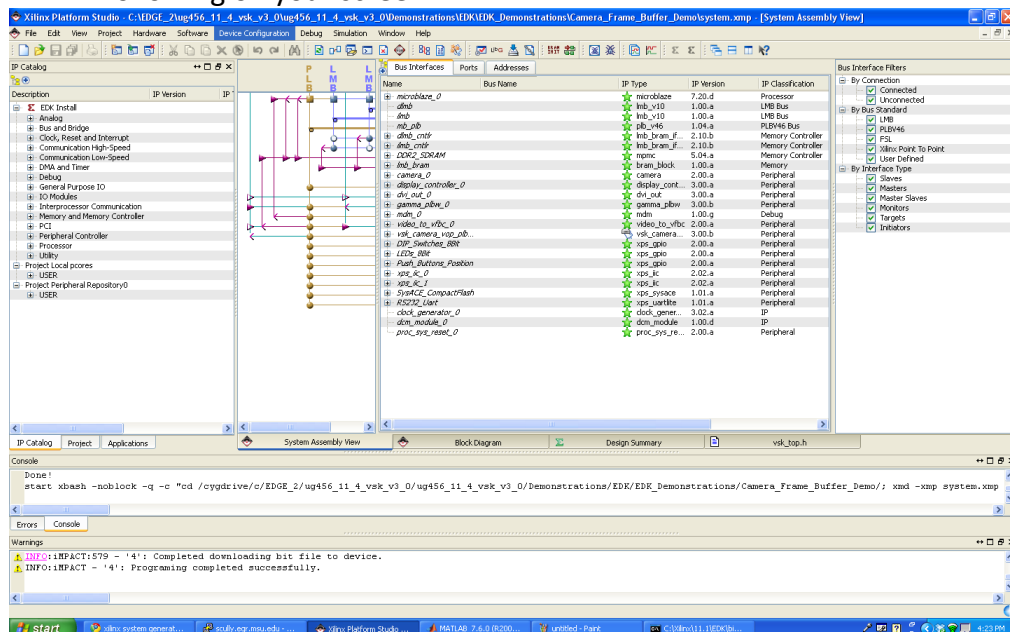
Opening the Project in XPS

Once the Pcore has been generated the project can now be opened in XPS to begin the process of downloading the program onto the board.

1. Locate the XPS program and open it
2. The following prompt will appear once the program is open.



3. Select open a recent project and browse to the same system.xmp file as Step 4 in the Generating Pcore section. The project will open and you should see the following on your screen.

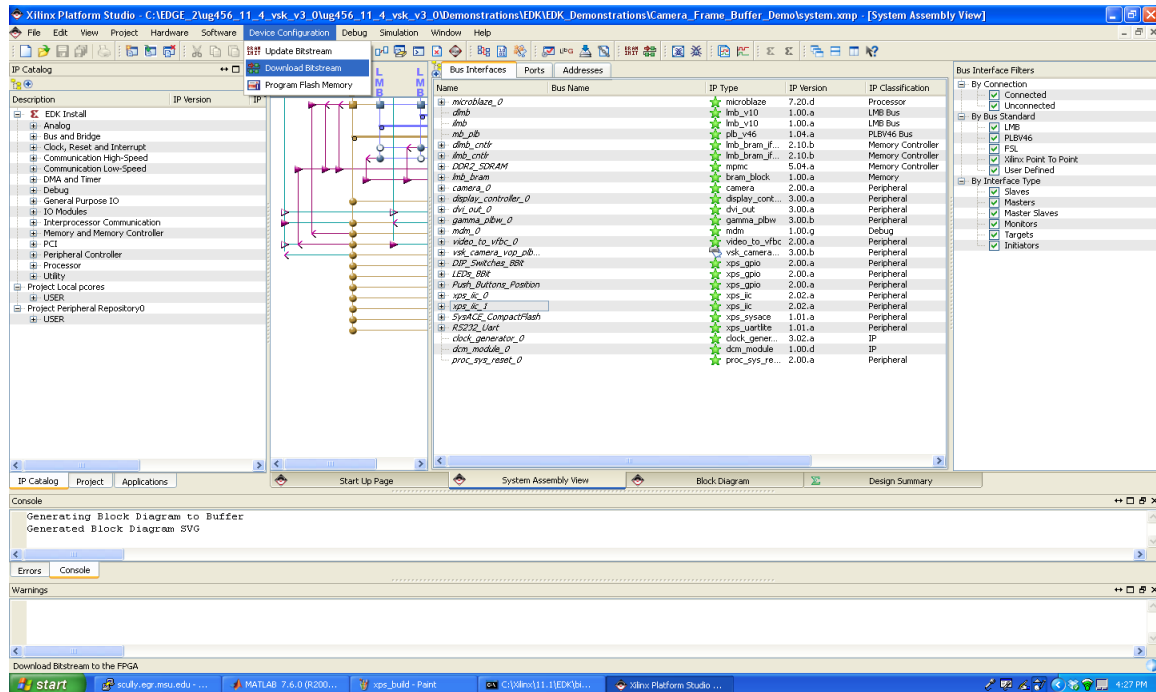


4. The next step is to generate the bitstream that will be downloaded to the board. Select Hardware > Generate Bitstream from the main menu to do this.

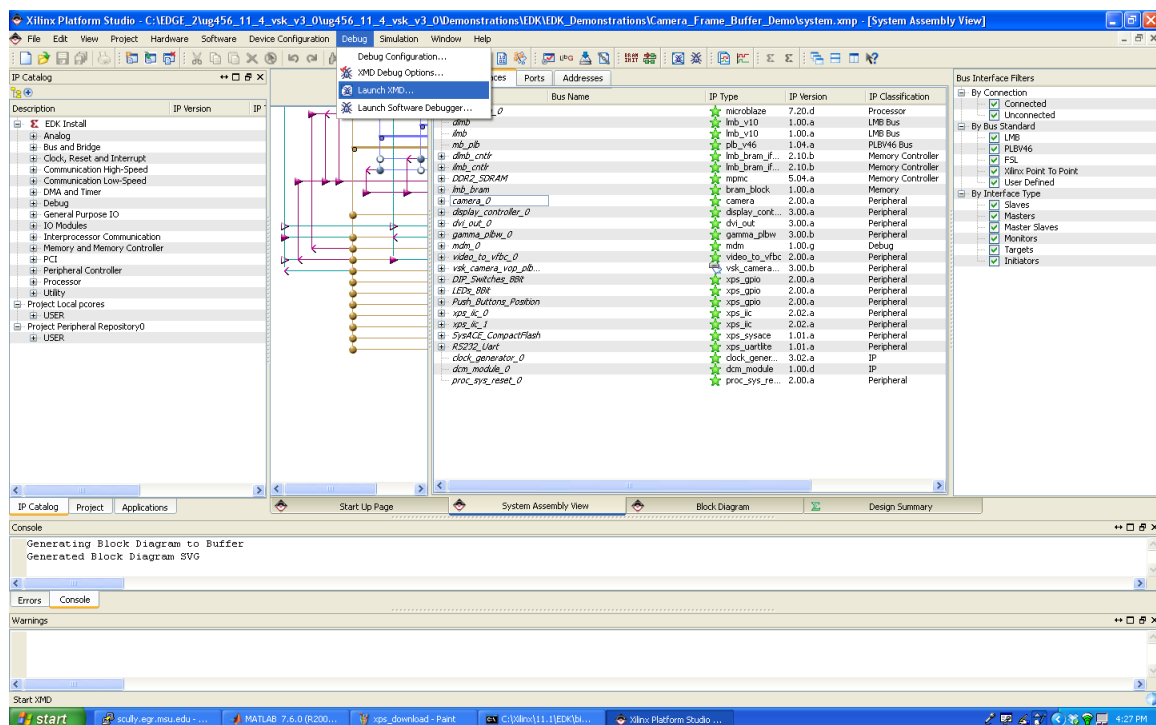


The screenshot displays the Xilinx Platform Studio (CSD) environment. The top menu bar includes options like File, Edit, View, Project, Hardware, Software, Debug, Simulation, Windows, and Help. The 'Software' menu is open, showing options such as 'Build All User Applications', 'Assign Default Drivers', 'Generate Libraries and BSPs', 'Get Program Set', 'Generate Linker Script...', 'Clean Libraries', 'Clean Programs', and 'Clean Software'. The central workspace shows a block diagram of a system with various IP blocks and interconnects. The right sidebar shows the 'Bus Interface Filters' and 'Bus Interface Filter' settings. The bottom status bar indicates the current project is 'Generated Block Diagram to Buffer'.

6. Now that the bitstream and software applications have been generated, it is time to download the bitstream to the board. This is accomplished by selecting Device Configuration > Download Bitstream from the main menu.



7. We are now ready to simulate our design and see the results of the custom logic that has been implemented into the demo. Under Debug select Launch XMD.



8. A command window will pop up and will go through a series of steps before it is ready to accept input. Once the window has stopped executing type stop and hit enter to ensure that the FPGA is ready to download the executable file.
9. On the next line type:

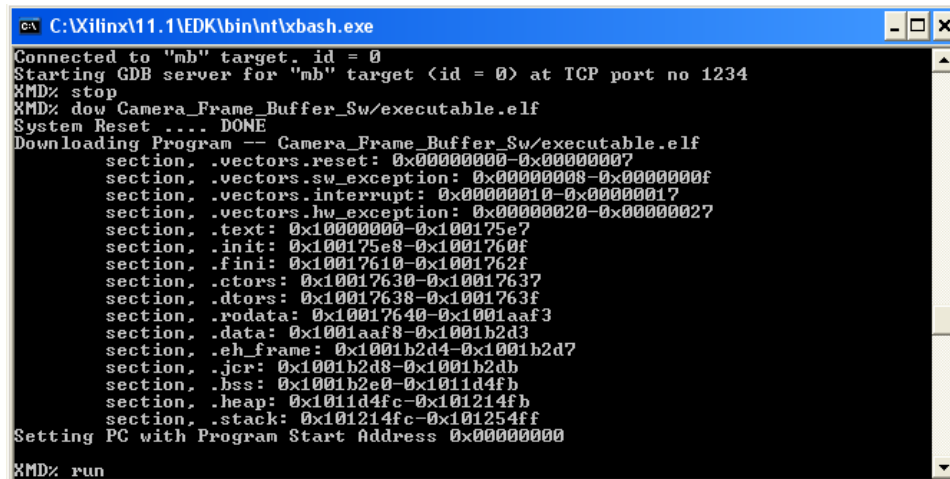
dow Camera_Frame_Buffer_Sw/executable.elf

This will download the executable file to the board. A series of addresses will appear on the screen to let you know the location of program.

10. Now that the program has been downloaded onto the board type:

run

Your program will now execute and you will see the output of the camera with your custom logic on the display.



```
C:\Xilinx\11.1\EDK\bin\nt\xbash.exe
Connected to "mb" target. id = 0
Starting GDB server for "mb" target (id = 0) at TCP port no 1234
XMD% stop
XMD% dow Camera_Frame_Buffer_Sw/executable.elf
System Reset .... DONE
Downloading Program -- Camera_Frame_Buffer_Sw/executable.elf
section, .vectors.reset: 0x00000000-0x00000007
section, .vectors.sw_exception: 0x00000008-0x0000000f
section, .vectors.interrupt: 0x00000010-0x00000017
section, .vectors.hw_exception: 0x00000020-0x00000027
section, .text: 0x10000000-0x100175e7
section, .init: 0x100175e8-0x1001760f
section, .fini: 0x10017610-0x1001762f
section, .ctors: 0x10017630-0x10017637
section, .dtors: 0x10017638-0x1001763f
section, .rodata: 0x10017640-0x1001aaf3
section, .data: 0x1001aaf8-0x1001b2d3
section, .eh_frame: 0x1001b2d4-0x1001b2d7
section, .jcr: 0x1001b2d8-0x1001b2db
section, .bss: 0x1001b2e0-0x1011d4fb
section, .heap: 0x1011d4fc-0x101214fb
section, .stack: 0x101214fc-0x101254ff
Setting PC with Program Start Address 0x00000000
XMD% run
```

Conclusion

This application note demonstrates one of the easiest ways to implement custom logic into a camera stream without having to learn a new programming language. Once the concepts covered in this application note are understood, this document can be used as a building block into more advanced programs on the FPGA.

References

“Spartan-3A DSP FPGA Video Starter Kit Xilinx” document UG456 (v2.1)

“Spartan-3A DSP FPGA Video Starter Kit Assembly: Getting Started With Demos”
Pascha Grant

“Grayscale Conversion of a Colored Image Using Simulink and Xilinx Blocks”
Emmett Kuhn.

<http://www.xilinx.com/support/documentation/do-s3adsp-video-sk-uni-g.htm>
VSK 3.0 Reference Designs