PYTHON-1300-C
Frame Buffer Design Tutorial

20 November 2015
Version 2015_2

Overview

This reference design is the getting started design for the PYTHON-1300-C camera module. The camera module features ON Semiconductor's PYTHON-1300 color image sensor. The PYTHON-1300 is a 1/2 inch Super-eXtended Graphics Array (SXGA) CMOS image sensor with a pixel array of 1280 by 1024 pixels. Designed to address the needs of general purpose industrial image sensing applications, the new global shutter image sensor combines flexibility in configuration and resolution with high speed and high sensitivity for the industrial imaging market.

Figure 1 – PYTHON-1300-C Camera Module

Objectives

This tutorial will guide the user how to:

- Retrieve the design files from the public Avnet git repository
- Build the reference design
- Execute the reference design on hardware

Reference Design Overview

The example design uses the Zynq processing system (PS) to initialize the PYTHON-1300-C camera, as well as the HDMI output interface. The design also implements a simple image sensor pipeline (ISP) and video frame buffer inside the programmable logic (PL).

The following figure illustrates the block diagram for the programmable logic (PL) hardware implementation.

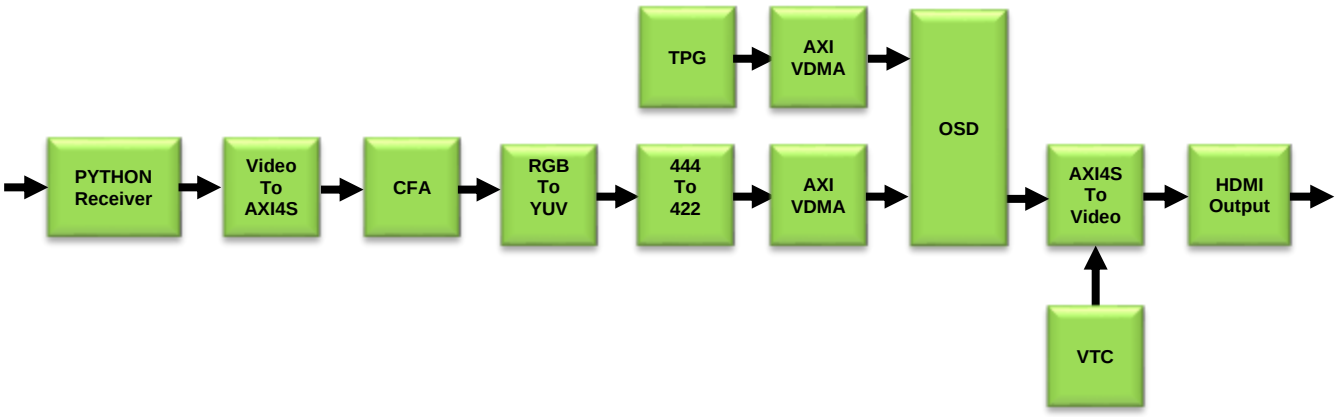

Figure 2 – PYTHON-1300-C Camera Reference Design – Hardware Block Diagram

Valid licenses (hardware evaluation, or full license) are required for the following video IP cores:

- Color Filter Array Interpolation (CFA) v7.0
- Chroma Resampler v4.0
- Video On Screen Display (OSD) v6.0
- RGB to YcrCb Color-Space Converter v7.1
- Video Timing Controller (VTC) v6.1

Experiment Setup

This tutorial makes use of Xilinx Vivado Design Suite in scripting mode in order to create a project. The resulting project can be opened with the graphical (GUI) version of the tools for further analysis and modification.

Software

The software required to build, and execute the reference design is:

- Windws-7 64-bit
- Terminal Emulator (HyperTerminal or TeraTerm)
- Xilinx Vivado Design Suite 2015.2
- MicroZed Board Definition Install for Vivado 2015.2
    - http://www.microzed.org/support/documentation/1519

Hardware

The hardware required to build, and execute the reference design is:

- Win-7 PC with a recommended 2 GB RAM available for the Xilinx tools to complete a XC7Z020 design[1]
- MicroZed Embedded Vision carrier card
- This reference design supports the following MicroZed SOMs
    - MicroZed 7020 SOM
- ON Semiconductor PYTHON-1300-C Canera Module
- HDMI (or DVI-D) monitor (1080P60 capable)
- USB cable (Type A to Micro-USB Type B)
- 4GB MicroSD card

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

Experiment 1: Licensing the Video and Image Processing Pack IP Cores

This reference design uses several of the Xilinx Video and Image Processing Pack IP cores. In order to build the hardware design, valid licenses (hardware evaluation, or full license) are required for the following video IP cores:

- Color Filter Array Interpolation (CFA) v7.0
- Chroma Resampler v4.0
- Video On Screen Display (OSD) v6.0
- RGB to YcrCb Color-Space Converter v7.1
- Video Timing Controller (VTC) v6.1

Follow these steps to request an evaluation license:

1. Navigate to the "Video and Image Processing Pack" product page on the Xilinx web site : http://www.xilinx.com/products/intellectual-property/ef-di-vid-img-ip-pack.html

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

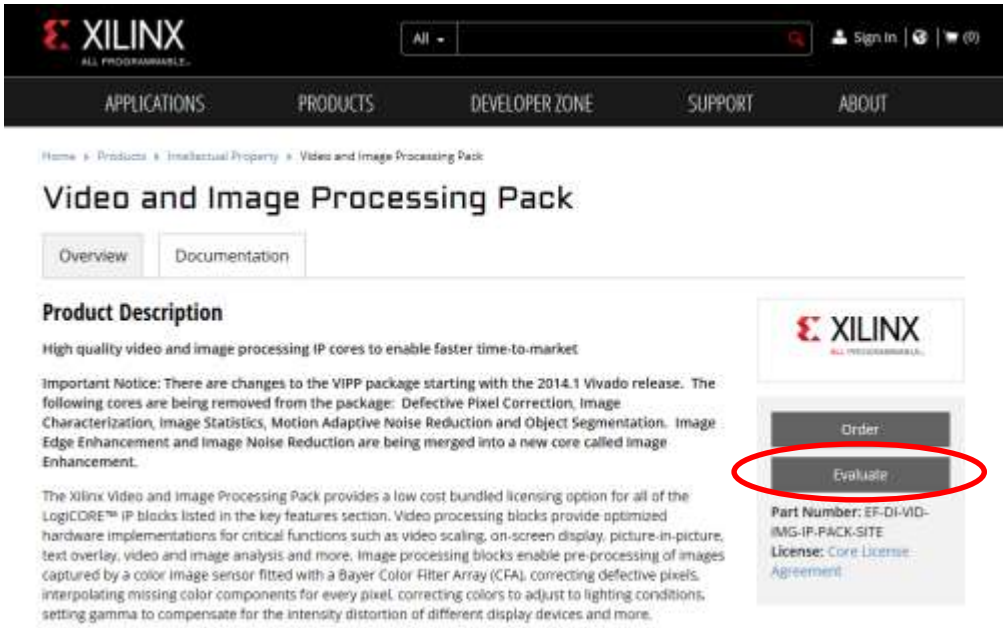

Figure 3 – Video and Image Processing Pack – product page

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.

Experiment 2: Retrieve the design files

In this section, the design files for the reference design will be retrieved from the Avnet git repository.

1. Navigate to the following web site : https://github.com/Avnet/hdl
2. Click the **branch:master** button
3. Specify the following search criteria : embv_python
4. Click the **Tags** tab

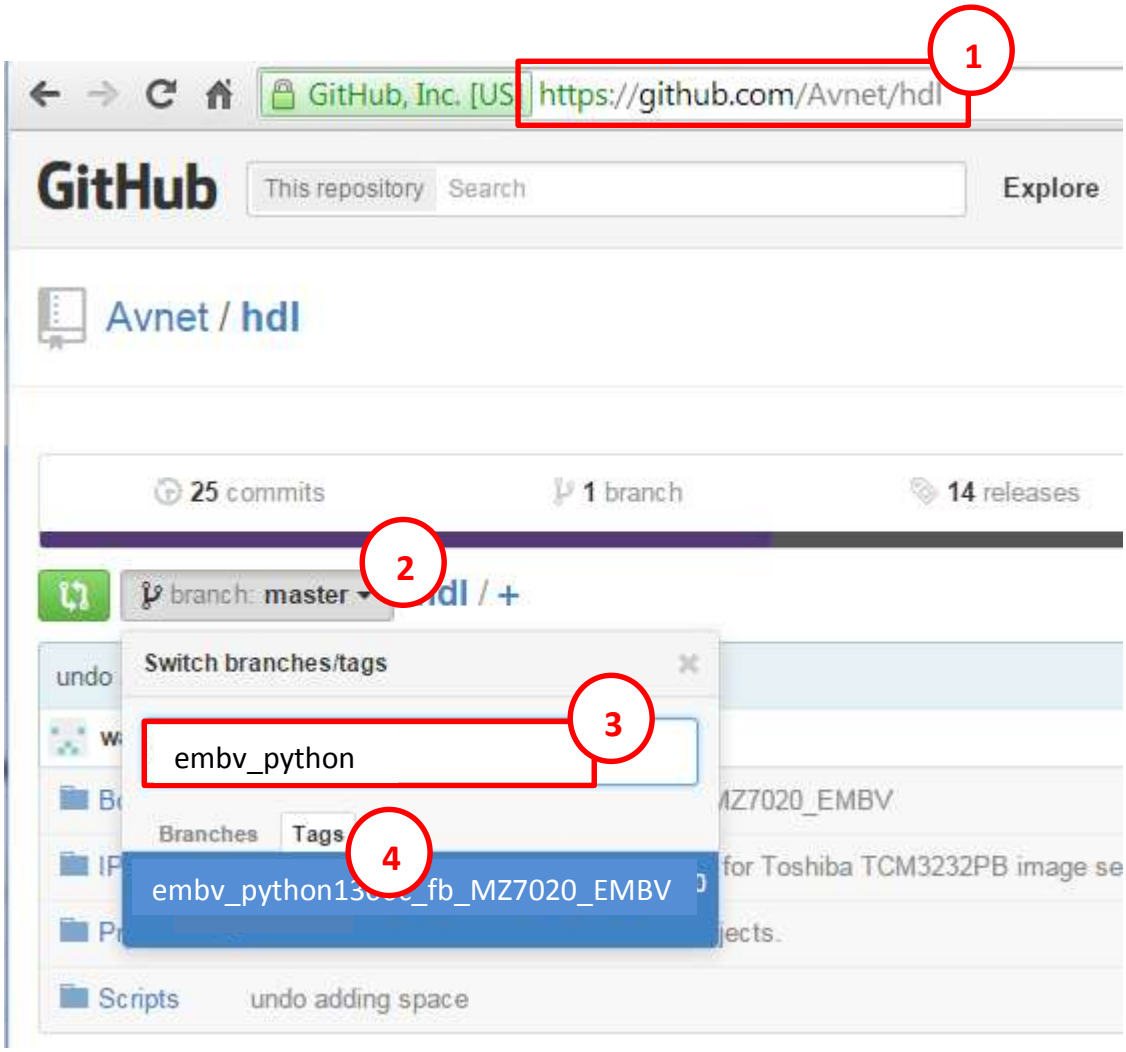

Figure 4 – Avnet GitHub repository – Retrieving specific version with tag

5. Select the **embv_python1300c_fb_MZ7020_EMBV_20151120_203702** tag
   This will retrieve a known working version of the design files for the PYTHON-1300-C reference design.

6. Click the Download ZIP file button

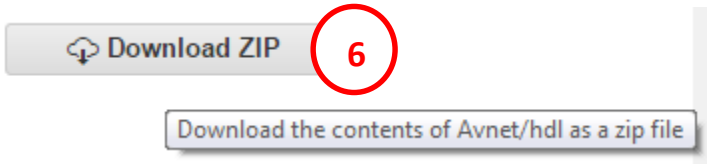

Figure 5 – Avnet GitHub repository – Download ZIP

7. Create an "Avnet" directory in your root C:\ drive
8. Save **hdl-embv_python1300c_fb_MZ7020_EMBV_20151120_203702.zip** file to the **C:\Avnet** directory, and extract the contents of the zip file in this directory
9. Rename the "hdl-embv_python1300c_fb_MZ7020_EMBV_20151120_203702" directory to "hdl"
   You should see the following directory structure

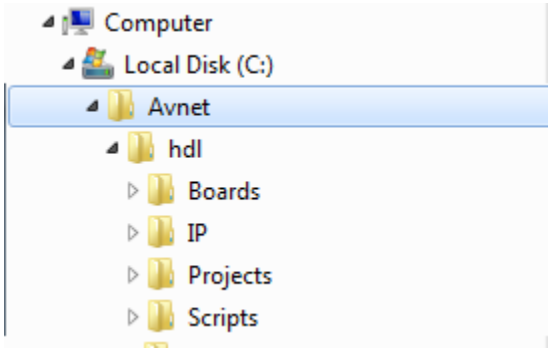

Figure 6 – Extracted C:\Avnet\hdl directory structure

NOTE : the exact directory name is not critical, but it must remain short on Windows machines, due to the directory length limitation of Windows

The **C:\Avnet\hdl** repository contains the following sub-directories:

Directory	Content Description
C:\Avnet\hdl**Boards**	contains board related files
C:\Avnet\hdl**IP**	contains the IP cores used by the ref designs
C:\Avnet\hdl**Projects**	contains project related files
C:\Avnet\hdl**Scripts**	contains scripts used to automatically build the designs

For the PYTHON-1300-C reference design, the following content is of interest:

Directory	Content Description
C:\Avnet\hdl\IP**avnet_hdmi_out**	IP core (including HDL source) for the HDMI output interface including embedded sync code insertion
C:\Avnet\hdl\IP**onsemi_vita_spi**	IP core (including HDL source) for the SPI controller for use with the VITA/PYTHON image sensors
C:\Avnet\hdl\IP**onsemi_vita_cam**	IP core (including HDL source) for the VITA/PYTHON camera receiver
C:\Avnet\hdl\Projects\ **embv_python1300c_fb**	files for PYTHON-1300-C reference design
C:\Avnet\hdl\Scripts\ **make_embv_python1300c_fb.tcl**	script to build PYTHON-1300-C reference design for MicroZed 7020

Experiment 3: Build the reference design

In this section, the Vivado project will be created and built with TCL scripts, implementing the PYTHON-1300-C reference design for the MicroZed Embedded Vision carrier card.

1. From the Start menu, open the "Vivado 2015.2 TCL Shell" console

2. Change to the **C:\Avnet\hdl\Scripts** directory

```
****** Vivado v2015.2 (64-bit)
  **** SW Build 1266856 on Fri Jun 26 16:35:25 MDT 2015
  **** IP Build 1264090 on Wed Jun 24 14:22:01 MDT 2015
    ** Copyright 1986-2015 Xilinx, Inc. All Rights Reserved.

Vivado% cd C:/Avnet/hdl/Scripts
Vivado%
```

Figure 7 – Vivado 2015.2 TCL Shell – Changing to C:/Avnet/hdl/Scripts directory

3. Launch the build with the "**source ./make_embv_python1300c_fb.tcl**" command

```
Vivado% source ./make_embv_python1300c_fb.tcl
# set argv [list board=MZ7020_EMBV project=embv_python1300c_fb sdk=yes
version_override=yes]
# set argc [llength $argv]
# source ./make.tcl -notrace

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
*-                                                                        -*
*-        Welcome to the Avnet Project Builder                            -*
*-                                                                        -*
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

+------------------+-----------------------------------+
| Setting          |     Configuration                 |
+------------------+-----------------------------------+
| Board            |     MZ7020_EMBV                   |
+------------------+-----------------------------------+
| Project          |     embv_python1300c_fb           |
+------------------+-----------------------------------+
| SDK              |     yes                           |
+------------------+-----------------------------------+
| Version override |     yes                           |
+------------------+-----------------------------------+
```

Figure 8 – Vivado 2015.2 TCL Shell – Launching the build

As a convenience, before building the hardware design, the scripts will verify if valid licenses are installed for the video IP cores used in the design.

```
***** Check for Video IP core licenses...

+------------------+-------------------------------------+
| Video IP Core    | License Status                      |
+------------------+-------------------------------------+
| v_cfa            | VALID (Hardware Evaluation)         |
+------------------+-------------------------------------+
| v_cresample      | VALID (Hardware Evaluation)         |
+------------------+-------------------------------------+
| v_osd            | VALID (Hardware Evaluation)         |
+------------------+-------------------------------------+
| v_rgb2ycrcb      | VALID (Full License)                |
+------------------+-------------------------------------+
| v_tc             | VALID (Full License)                |
+------------------+-------------------------------------+
| v_tpg            | VALID (Full License)                |
+------------------+-------------------------------------+
```

Figure 9 – Vivado 2015.2 TCL Shell – Video IP Core license verification

Each of the video IP cores requires a full license or hardware evaluation license in order to successfully build a bitstream.

The build will perform the following steps:

- Create and build the hardware design with Vivado 2015.2, including the IP Integrator block design

  C:\Avnet\hdl\Projects\embv_python1300c_fb\MZ7020_EMBV\
  embv_ python1300c_fb.xpr

- Create and build the SDK workspace, including board support package (BSP), software application, and first stage boot loader (FSBL)

  C:\Avnet\hdl\Projects\embv_python1300c_fb\MZ7020_EMBV\
  embv_ python1300c_fb.sdk

- Create the SD card image (BOOT.bin)

  C:\Avnet\hdl\Projects\embv_python1300c_fb\MZ7020_EMBV**BOOT.bin**

Experiment 4: Execute the reference design on hardware

This section describes how to execute the reference design on the hardware.

For instructions on how to setup the hardware, please refer to the PYTHON-1300-C Camera Module's Getting Started Guide.

Booting from SD card image

The BOOT.bin SD card image created in the previous experiment can be used to execute the reference design on hardware with the MicroZed Embedded Vision Kit.

For more detailed instructions on how to boot from the SD card, please refer to the PYTHON-1300-C Camera Module's Getting Started Guide.

Booting from JTAG with SDK

The hardware and software can also be loaded to hardware using SDK 2015.2 and a JTAG emulator.

Set up the hardware as described in the PYTHON-1300-C Camera Module's Getting Started Guide, with the following exceptions:

1. Set the MicroZed boot mode to cascaded JTAG using the following jumper settings:

JP1	JP2	JP3
1-2	1-2	1-2

2. Connect the JTAG Programming Cable (Platform Cable, Digilent HS1 or HS2 cable) to the PC using a USB cable and then plug the 14-Pin PC4 header or cable into the PC4 connector on the Embedded Vision Carrier Card.

3. Launch SDK 2015.2, and specify the following directory for the SDK workspace:

   C:\Avnet\hdl\Projects\embv_python1300c_fb\MZ7020_EMBV\
   embv_ python1300c_fb.sdk

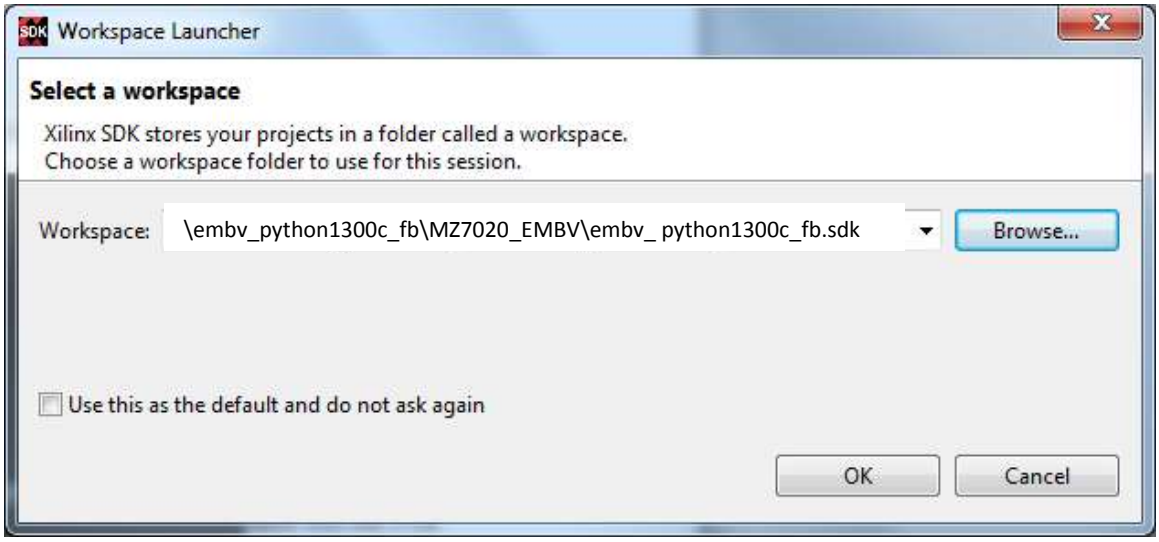

Figure 10 – SDK – Specifying SDK workspace

4. Close the Welcome Window

5. In the SDK menu, select **Xilinx Tools** => **Repositories**

6. Verify that the following Local Repository is specified:

   C:\Avnet\hdl\Projects\embv_python1300c_fb\software\sw_repository

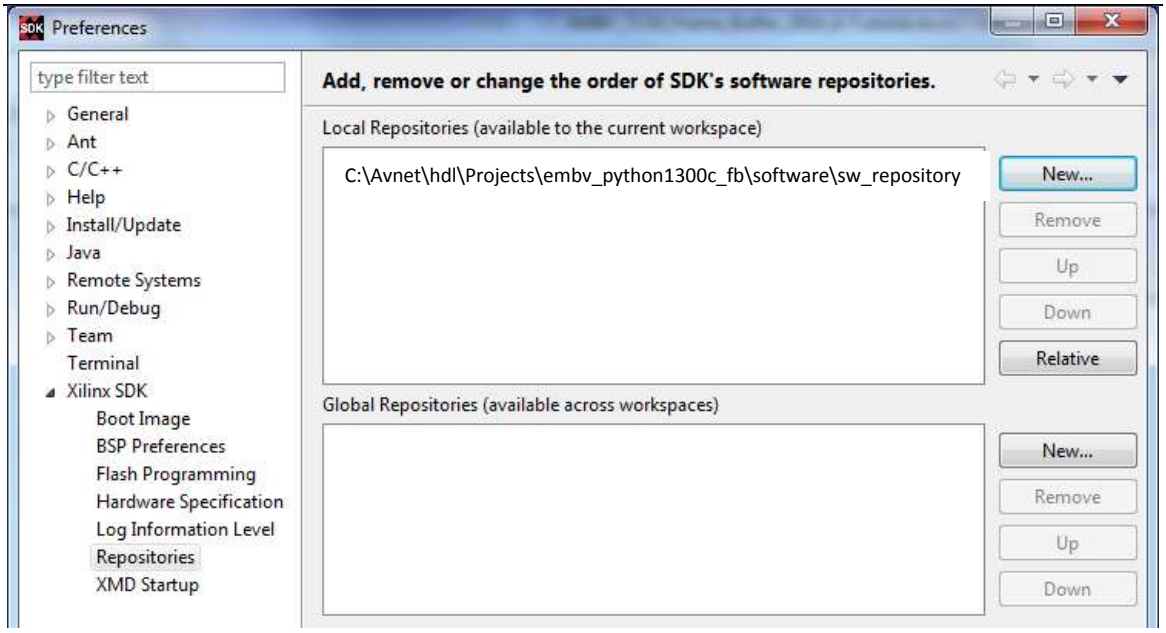

Figure 11 – SDK – Specifying local repository

7. If the local repository is not specified, click on the **New** button, navigate to the following directory, then click **OK**.

   C:\Avnet\hdl\Projects\embv_python1300c_fb\software\sw_repository

8. When done, click **OK**.

Now that the SDK workspace is correctly configured, the hardware and software can be loaded and executed on the hardware.

9. In the SDK menu, select **Xilinx Tools** => **Load FPGA**

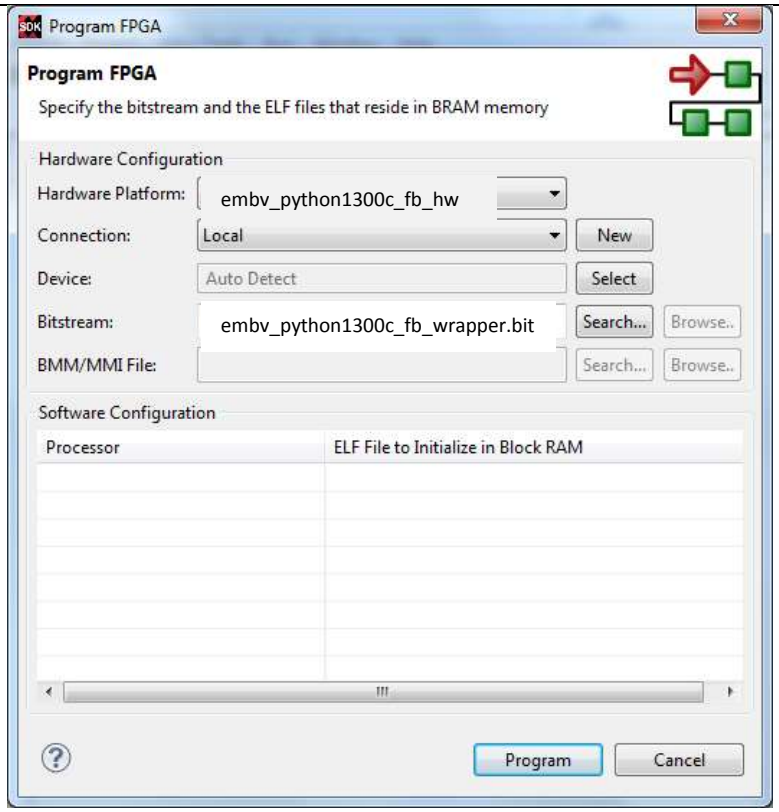

Figure 12 – SDK – Program FPGA

10. Click the **Program** button.
    It will take approximately 10 seconds to program the bitstream to hardware

11. Right-click **embv_python1300c_fb_app**
    and select **Run as** => Run **Configurations**.

12. Click **Xilinx C/C++ Application (GDB)** and click **New launch configurations**.

13. The new run configuration is created named **embv_python1300c_fb_app Debug**
    The configurations associated with application are pre-populated in the main tab of these launch configurations

14. Click on the **Application** tab

15. Next to the Application: edit box, click the **Search** button.

16. Select the **embv_python1300c_fb_app.elf** application, then click **OK**.

17. Click **Apply** and then **Run**.

18. If you get a Reset Status dialog box indicating that the current launch will reset the entire system, click **OK**.

19. You should see something similar to the following on your serial console:

```
------------------------------------------------------
--            Embedded Vision Carrier Card          --
--               PYTHON-1300-C Design               --
------------------------------------------------------

HDMI Initialization
PYTHON Initialization
CFA Initialization
TPG Initialization
VDMA 0 Initialization
VDMA 1 Initialization
OSD Initialization (camera=0xFF, tpg=0x00)
System Ready!

        Press 0-9 to change alpha blending of camera/tpg layers

        Press ENTER to restart
```

Figure 13 – PYTHON-1300-C Camera Reference Design – Serial Console Output

You will observe the content captured by the PYTHON-1300-C image sensor on the DVI/HDMI monitor.

To re-initialize the PYTHON image sensor, press the <ENTER> key.
You have successfully executed the PYTHON-1300-C reference design on hardware !

For more details on the available commands available in the serial console, please refer to the PYTHON-1300-C Camera Module's Getting Started Guide.

Revision History

Date	Version	Revision
4 Sep 2015	2014.4.01	Release to microzed.org site
20 Nov 2015	2015.5.01	Update to Vivado 2015.2