XAPP1285 (v1.0) June 10, 2016

Scaling Live Video with the Video Processing Subsystem

Author: Brian Wiec

Summary

Many broadcast and consumer video applications require changing video from one frame size to another. The technique used to solve this problem, known as scaling, involves interpolating or decimating the image (depending on whether up-scaling or down-scaling is being performed) to create or discard intermediate pixel values. Particularly as video frame sizes continue to grow, FPGAs are becoming an increasingly more attractive platform for performing video scaling due to the high processing bandwidth requirements and the inherent parallel nature of video scaling algorithms. FPGAs also provide the flexibility to integrate industry standard interfaces and do any arbitrary resolution scaling typically not supported by ASSP's.

This application note demonstrates how to use the LogiCORE™ Video Processing Subsystem (VPSS) core in a typical video scaling application. The VPSS is a hierarchical IP that bundles a collection of IP sub-cores to implement a plurality of video processing functions. While the core supports a variety of advanced configurations, this design shows how to use the core specifically to implement video scaling and thus disable other unused functional blocks as part of the IP subsystem. This application note should be used by designers interested in learning how to use the VPSS and migrating to it from other video scaling IP. The primary intent is to show how to bring up a simple video scaling pipeline.

The design associated with this application note was built using the Vivado® Design Suite 2015.4 and targets the Zynq®-7000 All Programmable SoC ZC702 evaluation kit. It can also optionally support the FMC-IMAGEON FMC module from Avnet for accepting live video input.

Introduction

This design showcases the use of the VPSS in Scaler-Only configuration to resize live video data in a typical video processing datapath. Video can be sent to the datapath using the ADV7611 HDMI™ input from the FMC-IMAGEON FMC card from Avnet. If this card is not available, the LogiCORE IP video test pattern generator is used as an internal video source. After the data is converted to AXI4-Stream, it is sent to the VPSS where scaling is performed. Because scaling generally requires different input and output pixel rates, the input and output video interfaces use different pixel clock sources. So to avoid artifacts such as tearing when asynchronous clocks are used, a triple frame buffer is used on the output of the VPSS that provides proper decoupling of input and output video clocks. After the frame buffer, the video data is converted back to a native video interface and sent in parallel to the ADV7511 transmitters on both the ZC702 board and the FMC-IMAGEON card, so that either HDMI connector can be used for observing the output video.

There is a second parallel datapath where the Video Scaler v8.1 performs the video re-sizing. Both the datapaths accept video data from the same source and, aside from having different IP cores performing the scaling, are otherwise identical. The final output data is sent to a software-controlled multiplexer before being sent to the ADV7511 transmitters. This allows you to dynamically select from the various datapaths at run time to compare the image quality.

The design also includes an embedded software component that runs on the ARM® processor in the processing subsystem (PS) of the Zynq-7000 AP SoC. The software performs initialization and configuration of the various IP cores in the system and allows you to dynamically change the scaler datapath to be used, the input and output frame sizes, and whether or not to bypass the Video Test Pattern Generator.

Implementation Details

Hardware

Most of the hardware for this example design was built using the Vivado IP integrator because it provides maximum productivity by allowing the designer to rapidly instantiate and connect IP. The flow minimizes design iterations by validating settings and avoiding user error associated with manual connection of IP interfaces.

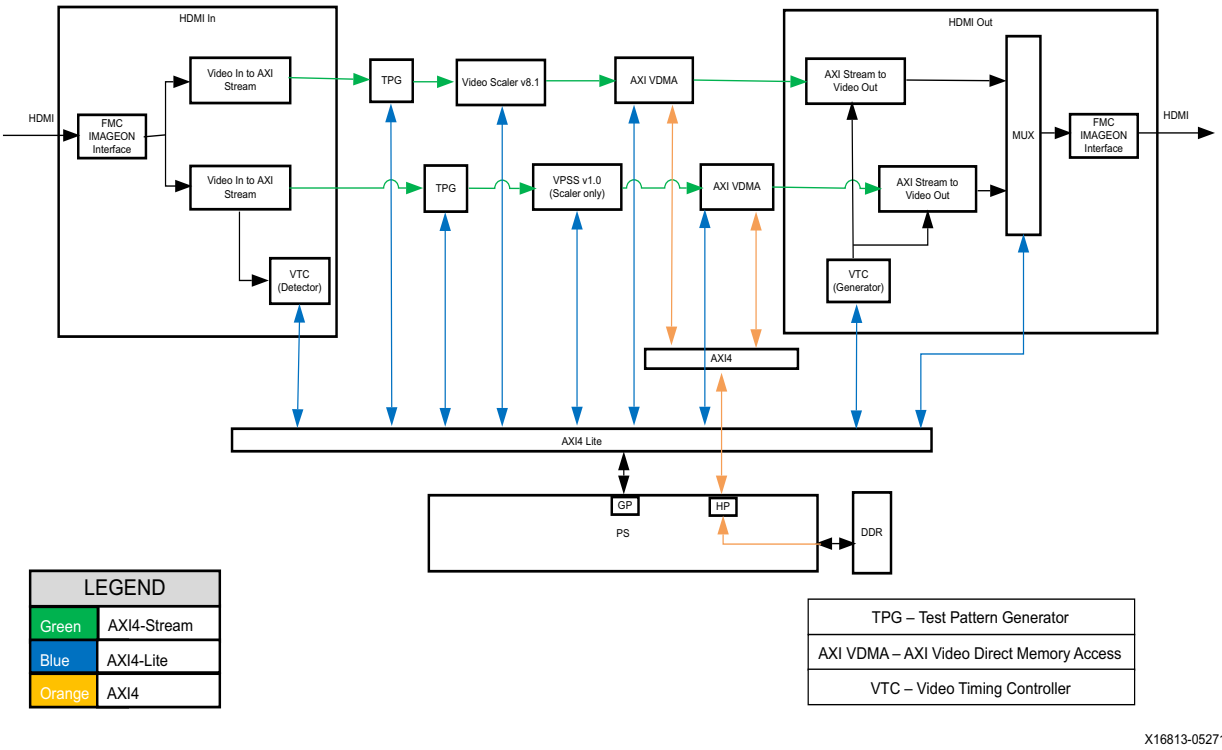

Figure 1: **Functional Block Diagram of the Hardware Design**

IP Cores

Video In to AXI4-Stream

The Video In to AXI4-Stream core implements a bridge from the native video interface (i.e., hblank/vblank or hsync/vsync) used by the ADV7611 to the AXI4-Stream video protocol used by Xilinx video processing IP cores.

For more information on this core, refer to:

Video In to AXI4-Stream v4.0 (PG043)

Video Timing Controller Detector

The Video Timing Controller (VTC), configured as a timing detector, locks onto and detects incoming video frame sizes. When used in conjunction with the Video In to AXI4-Stream, these IP cores prevent the propagation of partial frames (due to start-up core or hot-plug events) to downstream logic. The frame size data captured by this core can also be used to configure downstream cores with appropriate incoming frame sizes via the software access of AXI4-Lite registers. This VTC Generator is also used to re-generate native video timing signals based on the desired output video format.

For more information on this core, refer to:

Video Timing Controller v6.1 (PG016)

Video Test Pattern Generator

The Video Test Pattern Generator (TPG) design instantiates two Video Test Pattern Generator cores; one in each datapath. They are used as an alternative video source aside from live video. This is useful for debugging and also when the IMAGEON FMC card is not used.

The reason that two separate TPGs are used is because the two datapaths might have slightly different throttling behavior due to the different scaling cores that are used. By using two separate TPGs, the same video data could be guaranteed because they are configured identically, but subtle interface timing differences could be avoided more easily.

The Video Test Pattern Generator cores are configured to generate a color bar test pattern. There is also a small box that bounces around the screen. The color of the box indicates which datapath signal is sent to the monitor for display. When the box is blue, the Scaler v8.1 datapath is active. When the box is red, the VPSS datapath is being used.

For more information on this core, refer to:

Video Test Pattern Generator v7.0 (PG103)

Video Processing Subsystem

The VPSS performs the video scaling functionality. While the core supports a variety of image processing capabilities, it is configured in this design to only scale the video data. For a demonstration of the complete functionality of the core see the Video Processing Subsystem Reference Design.

While the VPSS supports ultra-high definition (UHD) resolutions and beyond by using the multiple pixel per clock capability, in this design the core is configured to support HD formats by using single pixel per clock mode to provide backwards compatibility with the Video Scaler v8.1 which only supports one pixel per clock.

The VPSS has two clocks that need to be connected. The aclk_ctrl is the clock for the AXI4-Lite control interface and is a 50 MHz clock sourced from the PS. The aclk_axis pin is the clock for the data processing and AXI4-Stream interfaces. This clock has been selected to run at 150MHz to provide sufficient processing bandwidth for the target HD resolutions (aclk_axis must run at least as fast as the maximum active pixel rate). Specifically, this design supports up to ultra

extended graphics array (UXGA) resolution which has a 162MHz pixel clock (though the active pixel rate is 1600 pixels per line × 1200 lines per frame × 60 frames per second = 115.2 megapixels per second, so a 150 MHz clock is more than sufficient).

The core also has an output reset signal called aresetn_io_axis. The purpose of this signal is to hold off upstream cores from sending data until the VPSS is configured and ready to accept the aresetn_io_axis signal. This signal is asserted any time the XVprocSs_SetSubsystemConfig() is called by the software. XVprocSs_SetSubsystemConfig() must be called any time the VPSS configuration is changed. Thus, upstream cores such as the Video Test Pattern Generator are reset any time the VPSS configuration needs to be changed (i.e., to change resolution or color format). Therefore, the software must take care to handle reconfiguration of any upstream cores when this event occurs.

When configured for Full Fledged mode, the VPSS supports crop, zoom, and picture-in-picture features which are accomplished using both the scaling and the Video DMA capabilities of the core. Because the Scaler-Only mode does not include a DMA engine inside the core itself, the cropping feature (similar to what was available in Scaler v8.1) could also be accomplished by placing the AXI VDMA instance from this design upstream (rather than downstream) of the VPSS. Then, through manipulation of the AXI VDMA MM2S channel's start addresses, hsize, stride, and vsize settings, a subset of the incoming image could be cropped before being sent to the VPSS where it is then scaled to full resolution. Cropping is not implemented in this design.

For more information on the VPSS core, refer to:

Video Processing Subsystem v1.0 (PG231)

Video Scaler

The Video Scaler IP can also implement video re-sizing functionality and is included in this design as a point of reference from which a system designer might migrate to the VPSS. It is configured to support the same feature set used by the VPSS.

Note: The Video Scaler IP should not be used for new designs. It is only used here to provide an example point of reference from which you might migrate to the VPSS in Scaler-Only mode.

For more information on this core, refer to:

Video Scaler v8.1 (PG009)

AXI4-Stream Subset Converters

The ADV7611 receivers and ADV7511 transmitters used in this design are configured for YCbCr 4:2:2 color format with 8 bits per component. This results in a video data interface width of 16 bits per clock cycle (8 bits for the luma component and 8 bits for the interleaved chroma red and blue difference components). The Video Test Pattern Generators and the VPSS support run-time configuration of the color format (which the Scaler v8.1 does not support) and, therefore, always have hardware data widths to support three color components (for example, 8 bits per component RGB or YCbCr 4:4:4 would require 24 bits per pixel on the hardware, even if only two of those color components are used when the core is configured via software for 4:2:2 subsampling). Because this design only uses two color components, the AXI4-Stream

subset converters are used to zero-pad or bit-select only the components used for YCbCr 4:2:2 (the lower 16 bits) from the interface. These blocks are ultimately optimized out of the design as wires and have no area cost.

For more information on this core, refer to:

AXI4-Stream Infrastructure IP Suite (PG085)

AXI Video Direct Memory Access

The AXI Video Direct Memory Access (VDMA) cores are used to decouple the input and output pixel clocks. They are configured as triple frame buffers and store video data in external DDR through their connection to the PS memory controller via the HP ports.

The maximum theoretical video memory bandwidth (per stream) is calculated as follows:

Bandwidth (per stream) = max_frame_rate × max_active_lines_per_frame × max_active_pixels_per_line × bits_per_pixel (b/s).

Because this design supports up to UXGA resolution (1600 x 1200) at 60 frames per second, max_frame_rate is 60, max_active_lines_per_frame is 1600, and max_active_pixels_per_line is 1200. The design also supports only the YCbCr 4:2:2 color format with 8 bits per component, giving 16 bits per pixel. Using the above equation, the final bandwidth requirement calculation for this design is:

Bandwidth (per stream) = 60 × 1600 × 1200 × 16 = 1,843,200,000 = 1.8432 (Gb/s)

The design uses a total of 4 streams (two AXI VDMA instances, each configured for both read and write) so the total aggregate memory bandwidth requirement is:

Bandwidth (total) = 1.8432 × 4 = 7.3728 (Gb/s)

Each of the AXI VDMA's memory mapped interfaces must be configured to support at least the required bandwidth per stream. They are configured to be 64 bits wide and run at a rate of 200MHz. This gives a maximum theoretical supported bandwidth of 64 × 200 = 12,800 (Mb/s) = 12.8 (Gb/s).

For more information on this core, refer to:

AXI Video Direct Memory Access v6.2 (PG020)

AXI Interconnect

This design uses three instances of the AXI Interconect IP. The axi_interconnect_0 in the control_path hierarchy of the design is a low-area, low-bandwidth AXI4-Lite interconnect that provides access to control interfaces of the various IP cores. This interconnect does not handle any video data.

The axi_interconnect_1 instances of the core used in the framebuffer_old and framebuffer_new hierarchies are configured for higher performance to support transfer of video data to and from DDR in real time. Specifically, these instances each handle two streams of video (one read and one write), so they need to be configured to support up to 1.8432 × 2 = 3.6864 (Gb/s) of

aggregate bandwidth. They are configured to be 64 bits wide and run at a rate of 200 MHz. This gives a maximum theoretical supported bandwidth of 64 × 200 = 12800 (Mb/s) = 12.8 (Gb/s).

For more information on this core, refer to:

AXI Interconnect v2.1 (PG059)

Video Timing Controller Generator

The Video Timing Controller (configured as a timing generator) is used to re-generate native video timing signals based on the desired output video format.

For more information on this core, refer to:

Video Timing Controller v6.1 (PG016)

AXI4-Stream to Video Out

This IP provides conversion from the AXI4-Stream video format used by Xilinx video IP cores to native video format used to send video data to the ADV7511 devices.

For more information on this core, refer to:

Video In to AXI4-Stream v4.0 (PG043)

AXI IIC

The AXI IIC core is used for configuring the ADV7511 transmitters and ADV7611 receivers on the IMAGEON FMC card.

For more information on this core, refer to:

AXI IIC Bus Interface v2.0 (PG090)

AXI GPIO

The AXI GPIO core is used to control the output video multiplexer which allows for run-time switching between the two datapaths via software.

For more information on this core, refer to:

AXI GPIO v2.0 (PG144)

Software

The application-level software architecture for the datapath of this design includes three layers of software as shown in Figure 2.

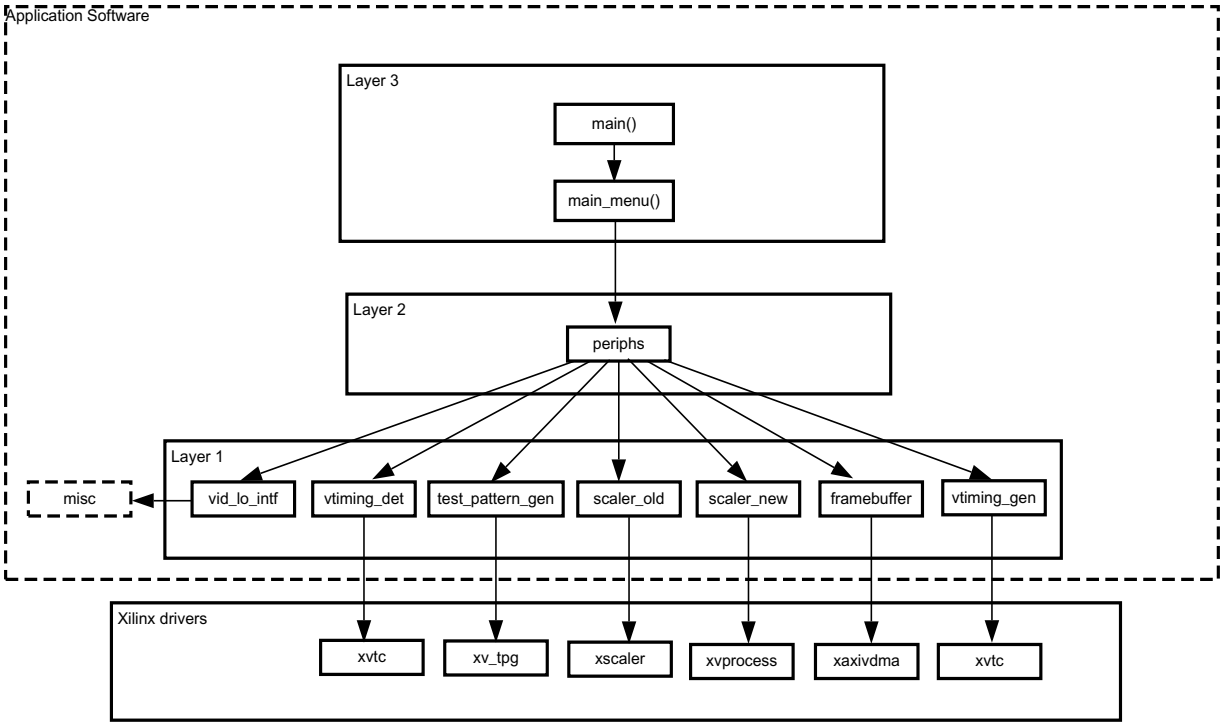

Figure 2: **Application-Level Software Architecture**

The highest layer of code (layer 3) is in `main.c` and it is the entry point into the application. It contains an instance of the periphs_t object and calls appropriate layer 2 software to initialize the various devices. After the initialization is complete, the software then passes control to a background process that waits on the user to press a key via the UART. Once a key is pressed to change some configuration option, the software again calls appropriate layer 2 functions to update the hardware.

The next level down is the peripheral (layer 2). This software encapsulates instances of all peripherals (including IP in the PL) used in the design as well as the main logic for the IP configuration sequencing. This software layer exposes the functionality to detect the FMC card status, detect the input video frame size, set the input video frame size (used when there is no FMC card connected), update the output frame size, select which datapath to send to the display, and toggle whether or not the TPG is active or bypassed.

The lowest layer of software (layer 1) written for this design provides extra abstraction over individual IP drivers to simplify the interface to the given IP driver and expose only the subset of functionality needed for this design. Each IP in the datapath has its own wrapper which expects that the caller has properly allocated and initialized the IP driver instance that is worked on.

The one exception to this programming model is the software used to configure the video input and output interfaces (i.e., the ADV7511, ADV7611, and video clock synthesizer). This functionality is noted as '*misc*' in Figure 2. This code has a different structure under the layer 1 software because it leverages existing libraries provided by other vendors which have different interfaces and paradigms. Because this portion of the design is not important for the purpose of this application note, it will not be discussed further. Refer to the code comments and documentation from those vendors for more information.

The software is organized with the goal of showing how to easily handle migration to the VPSS (and secondarily to ease the removal of the old video scaling datapath). As such, IP cores used exclusively in the old scaling datapath use the naming convention *_old_* in the code itself. Similarly, IP cores used exclusively in the new scaling datapath use the naming convention *_new_*. Further, all function calls to corresponding IP cores in each datapath are directly adjacent to one another and use the same interface. For example, `periphs.c` has a function called periphs_set_output_fsize() which is called when you decide to change the output video frame size. To do this, one action (among others) that needs to be performed is to update both the Video Scaler v8.1 (termed scaler_old in the code) and the VPSS (termed scaler_new in the code), as shown in this snippet of code:

```
// Set old Scaler parameters
xil_printf("Setting up old scaler.\n\r");
status = scaler_old_set_output_size
(
    p_periphs_inst->p_scaler_old_inst,
    p_periphs_inst->p_vid_io_intf_inst->p_output_timing_inst->HActiveVideo,
    p_periphs_inst->p_vid_io_intf_inst->p_output_timing_inst->VActiveVideo,
    0 // Print configuration
);
if (status != SCALER_OLD_SUCCESS)
{
    xil_printf("ERROR! Failed to set output size on the old scaler.\n\r");
    return PERIPHS_ERROR_UNKNOWN;
}
// Set new (VPSS-based) Scaler parameters
xil_printf("Setting up new VPSS-based scaler.\n\r");
status = scaler_new_set_output_size
(
    p_periphs_inst->p_scaler_new_inst,
    p_periphs_inst->p_vid_io_intf_inst->p_output_timing_inst->HActiveVideo,
    p_periphs_inst->p_vid_io_intf_inst->p_output_timing_inst->VActiveVideo,
    0 // Print configuration
);
if (status != SCALER_NEW_SUCCESS)
{
    xil_printf("ERROR! Failed to set input size on the new VPSS-based scaler.\n\r");
    return PERIPHS_ERROR_UNKNOWN;
}
```

Because the function calls to corresponding parallel datapath IP cores are side-by-side in the code and the naming convention is consistent, it is clear which functions pertain to which datapath and the *_old_* function calls can easily be removed as desired.

Reference Design

The reference design files for this application note can be downloaded at:

https://secure.xilinx.com/webreg/clickthrough.do?cid=424057

Hardware Requirements

1. ZC702 board
2. Video monitor capable of displaying the desired output resolutions
3. HDMI cable (ZC702 to display)
4. Micro-USB cable
5. Mini-USB cable
6. (Optional) IMAGEON FMC card from Avnet
7. (Optional) Video source (must support non-encrypted video)
8. (Optional) HDMI cable (video source to IMAGEON FMC)

Software Requirements

1. Vivado Design Suite 2015.4 with SDK
2. Serial terminal client (e.g., PuTTY, Terra Term)

Installing Design Files

Download the `xapp1285.zip` and extract the contents of the ZIP file to the C: drive of the host PC:

```
C:\xapp1285
```

Note: The Windows operating system has a 260 character limitation on the maximum length for a path. Make sure that the installation path is short to prevent path length related errors. If another location is chosen, there should be no spaces in the folder names.

Setting Up the Hardware

1. Connect the IMAGEON FMC card to the FMC2 slot (J4) of the ZC702. (Optional)
2. Connect an unencrypted HDMI video source to the 'HDMI IN' connector of the IMAGOEN FMC card using an HDMI cable. (Optional)
3. Connect an HDMI cable from the ZC702 on-board HDMI connector (P1) to the video monitor.
4. Connect the micro USB cable from the host PC to the USB JTAG connector (U23) on the ZC702.

5. Connect the mini USB cable from the host PC to the USB UART connector (J17) on the ZC702.
6. Set the boot mode jumpers to JTAG (SW16) by setting all switches to the OFF position.
7. Connect the power supply to the board and switch on the power to the board.
8. Using the serial terminal client software on the host PC, connect to the appropriate COM port with these settings:
   a. 115200 baud
   b. 8 data bits
   c. No parity
   d. 1 stop bit
   e. No flow control

Running the Design Using the Provided Binaries

1. Launch the Xilinx Software Command Line Tool (xsct) from the SDK installation.
2. From the xsct terminal, navigate to `<xapp_extract_directory>/xapp1285/ready_for_download/`
3. From the xsct terminal, run the command `xmd-tcl runme.tcl`.

   This TCL script performs the following steps:

   a. Connects to the target (ZC702) via JTAG.
   b. Downloads the bitstream to the target.
   c. Sources the `ps7_init.tcl` script and runs appropriate utilities to initialize the PS.
   d. Downloads the ELF file to the target.
   e. Runs the ELF file.
4. There is activity on the UART menu as the application is initialized. After a few seconds when initialization is complete, this line appears:

   ```
   What would you like to do? Press 'p' to print available commands.
   ```
5. The application can now be manipulated using the UART interface. Press **p** to see the available options as shown in Figure 3.

```
Hello World!
Initializing Video Input/Output interface.
Initializing IIC for FMC IMAGEON.
Checking for FMC IMAGEON card.
FMC IMAGEON not detected.
Initializing remaining peripheral drivers.
Initializing GPIO for output video mux.
Initializing VTG.
Initializing VDMA.
Initializing Scaler.
Initializing VPSS-based Scaler.
Initializing VTD.
Initializing TPG.
Initialize input to 1080p.
Initialize output to 720p.
Setting up framebuffers.
Setting up old scaler.
Setting up new VPSS-based scaler.
Setting up TPG.
Enabling new scaler by default.
Initialization complete. Switching to background process.
What would you like to do? Press 'p' to print available commands.

------------ Scaler Demo ------------
'n' = Set to new scaler datapath
'o' = Set to old scaler datapath
't' = Enable/bypass TPG in the new scaler datapath
'd' = Detect or set input frame size
's' = Set output frame size
'p' = Print this menu
-------------------------------------
```

Figure 3: **Available Commands**

- Selecting **n** or **o** switches the output multiplexer to show the new (VPSS) and old (Video Scaler v8.1) datapaths, respectively.
- Selecting **t** toggles the TPG between bypass and pass through modes when the FMC card is connected. When the TPG is in use and the box that moves around the screen is blue, the old scaler datapath is in use. When the box is red, the new scaler datapath is in use. If the FMC card is not detected, this menu option does nothing.
- Selecting **d** causes the application to attempt to re-detect the incoming frame size when the FMC card is connected. This option should be selected any time the video source format changes in real timeto avoid artifacts in the video data. When the FMC card is not connected, only the internal test pattern generator can be used as the video source and a new menu is used to manually select supported input formats. From this context, press **p** to list supported resolutions as shown in Figure 4.

```
What would you like to do? Press 'p' to print available commands.
FMC IMAGEON card is not connected. Using internal TPG as video source. Please select a reso
lution to set it to. Press 'p' to print available resolutions.

------------- Resolutions -------------
0 = VGA
1 = 480P
2 = 576P
3 = SVGA
4 = XGA
5 = 720P
6 = SXGA
7 = 1080P
8 = UXGA
'p' = Print this menu
'q' = Quit
---------------------------------------
```

Figure 4: **List of Supported Resolutions**

- Selecting **s** allows you to set the output frame size. From this context, press **p** to list the supported resolutions as shown in Figure 5.

```
Setting output frame size. What resolution would you like? Press 'p' to print available res
olutions.

------------- Resolutions -------------
0 = VGA
1 = 480P
2 = 576P
3 = SVGA
4 = XGA
5 = 720P
6 = SXGA
7 = 1080P
8 = UXGA
'p' = Print this menu
'q' = Quit
---------------------------------------
```

Figure 5: **Output Frame Size**

Rebuilding the Hardware Application

1. Launch the Vivado tools 2015.4
   a. On a Windows host, select **Start** > **All Programs** > **Xilinx Design Tools** > **Vivado 2015.4** > **Vivado 2015.4**.
   b. On a Linux host, enter `vivado` at a command prompt.
2. From the Vivado tools welcome screen in the TCL console, run the following commands:
   a. `cd <xapp_extract_directory>/xapp1285/HW/tcl`
   b. `source all.tcl`
3. After the script completes, the bitstream is created and the hardware is exported to SDK.
4. Now, launch SDK by selecting **File** > **Launch SDK** and click **OK**.

Rebuilding the Software Application

1. After the SDK opens, the new hardware platform exists in the project. Now create a new application and board support package (BSP) by selecting **File** > **New** > **Application Project**.
2. In the Project Name field, enter `vpss_scaler` and click **Next**. Select **Empty Application** and then click **Finish**.
3. Next, add various software libraries used in the design by selecting **Xilinx Tools** > **Repositories**. Then click **New** and point to the **`<xapp_extract_directory>/xapp1285/SW/sw_lib`** directory. Click **OK** and then click **OK** again.
4. In the Project Explorer window, right-click **vpss_scaler_bsp** and select **Board Support Package Settings**.
5. In the Overview section of the Board Support Package Settings dialog, check the boxes for **fmc_iic_sw** and **fmc_imageon** in the Supported Libraries subheading. Click **OK**.
6. In the Project Explorer window, drop down **vpss_scaler**, right-click the **`src`** directory, and select **Import**.
7. In the Import dialog, drop down the **General** folder, choose **File System**, and select **Next**.
8. Click the **Browse** button and point to the `<xapp_extract_directory>/xapp1285/SW/src` directory.
9. Check the boxes for all files in the `src` directory and click **Finish**.
10. Select **Project** > **Clean** and then click **OK**.
11. After the software builds, select **Xilinx Tools** > **Program FPGA** then click **Program**.
12. In the Project Explorer window, right-click **vpss_scaler** and choose **Run As** > **Launch On Hardware (System Debugger)**.
13. The design can now be manipulated in the same way as described in step 5 of the Running the Design Using the Provided Binaries.

Debugging – Running the Design in Hardware

Issue: The monitor does not detect any input signal.

Resolution: Try the following:

- Make sure the monitor detects a signal when the source is connected directly to it.
- Make sure all cables are plugged in before the board is powered on and the application is run. The design does not support hot plug detection of the HDMI cable.
- Use a high-quality HDMI cable to connect from the ZC702 to the monitor. Try another cable.
- From the UART menu, try setting the output resolution to a smaller frame size that uses a slower pixel clock so the monitor has an easier time synchronizing.

- Ensure the monitor supports the selected output resolution. The application's default output resolution is 720p at 60 frames per second.

Issue: The monitor detects the video, but the screen is blank or a solid color when using the live video input.

Resolution: First, try using the internal TPG to make sure the output path is working properly. Next, this issue can happen if the video source is encrypting the data with high-bandwidth digital content protection (HDCP). This design only supports non-encrypted video sources. Try another source. Also, if the hardware is being re-built then a license is required for both the Video Test Pattern Generator and the VPSS. If a hardware evaluation license is used, the cores will timeout after a period of time and the monitor might go blank.

Issue: While the application is running, the image on the screen becomes distorted when the input video resolution is changed.

Resolution: The design does not automatically detect changes in input format. When you change the input video resolution, press the **d** key in the UART menu to force the application to re-detect the input format and re-configure the various IP cores appropriately.

Debugging – Rebuilding the Design

Issue: The design fails during generation or synthesis of the VPSS.

Resolution: Make sure licenses have been installed for the Test Pattern Generator and the VPSS. Also, see Xilinx Answer 66403 and Xilinx Answer 66692.

Reference Design Matrix

Download the reference design files for this application note from the Xilinx website.

Table 1 indicates the tool flow and verification procedures used for the provided reference design.

Table 1: **Reference Design Matrix**

Parameter	Description
General	
Developer name	Brian Wiec
Target devices	Zynq-7000 AP SoC
Source code provided	Yes
Source code format	Verilog, c
Design uses code and IP from existing Xilinx application note, reference designs, third party or Vivado software? If yes, list.	Video IP from Vivado IP Catalog, Avnet FMC IMAGEON supporting IP
Simulation	
Functional simulation performed	No
Timing simulation performed	No

Table 1: **Reference Design Matrix *(Cont'd)***

Parameter	Description
Test bench used for functional and timing simulations	No
Test bench format	N/A
Simulator software/version used	N/A
SPICE/IBIS simulations	N/A
Implementation	
Implementation software tools/versions used	Vivado tools 2015.4
Static timing analysis performed	Yes
Hardware Verification	
Hardware verified	Yes
Hardware platform used for verification	ZC702 board with IMAGEON FMC card

References

1. *LogiCORE IP Video In to AXI4-Stream Product Guide* (PG043)
2. *LogiCORE IP Video Timing Controller Product Guide* (PG016)
3. *LogiCORE IP Video Test Pattern Generator Product Guide* (PG103)
4. *LogiCORE IP Video Processing Subsystem Product Guide* (PG231)
5. *LogiCORE IP Video Scaler Product Guide* (PG009)
6. *AXI4-Stream Infrastructure IP Suite Product Guide* (PG085)
7. *LogiCORE IP AXI Video Direct Memory Access Product Guide* (PG020)
8. *LogiCORE IP AXI Interconnect Product Guide* (PG059)
9. *LogiCORE IP AXI4- Stream to Video Out Product Guide* (PG044)
10. *LogiCORE IP AXI IIC Product Guide* (PG090)
11. *LogiCORE IP AXI GPIO Product Guide* (PG144)
12. *ZC702 Evaluation Board for the Zynq-7000 XC7Z020 All Programmable SoC User Guide* (UG850)
13. *Zynq-7000 All Programmable SoC Technical Reference Manual* (UG585)
14. *AXI4-Stream Video IP and System Design Guide* (UG934)
15. *HDMI Input/Output FMC Module* (http://www.em.avnet.com/en-us/design/drc/Pages/HDMI-Input-Output-FMC-module.aspx)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/10/2016	1.0	Initial Xilinx release.

Please Read: Important Legal Notices