AN 889: 8K DisplayPort Video Format Conversion Design Example

Contents

Send Feedback

About the 8K DisplayPort Video Format Conversion Design Example

The 8K DisplayPort Video Format Conversion Design Example integrates the Intel DisplayPort 1.4 video connectivity IP with a video processing pipeline. The design delivers high-quality scaling, color space conversion, and frame rate conversion for video streams up to 8K at 30 frames per second, or 4K at 60 frames per second.

The design is highly software and hardware configurable, enabling rapid system configuration and redesign. The design targets Intel® Arria® 10 devices and uses the latest 8K ready Intel FPGA IP from the Video and Image Processing Suite in Intel Quartus® Prime v19.2.

About DisplayPort Intel FPGA IP

To create Intel Arria 10 FPGA designs with DisplayPort interfaces, instantiate the DisplayPort Intel FPGA IP. However, this DisplayPort IP only implements the protocol encode or decode for DisplayPort. It does not include the transceivers, PLLs, or transceiver reconfiguration functionality required to implement the high-speed serial component of the interface. Intel provides separate transceiver, PLL, and reconfiguration IP components. Selecting, parameterizing, and connecting these components to create a fully compliant DisplayPort receiver or transmitter interface requires specialist knowledge.

Intel provides this design for those who are not transceiver experts. The parameter editor GUI for the DisplayPort IP allows you to build the design.

You create an instance of the DisplayPort IP (which may be receiver only, transmitter only or combined receiver and transmitter) in either Platform Designer or the IP Catalog. When you parameterize the DisplayPort IP instance, you can select to generate an example design for that particular configuration. The combined receiver and transmitter design is a simple passthrough, where the output from the receiver feeds directly in to the transmitter. A fixed-passthrough design creates a fully functional receiver PHY, transmitter PHY, and reconfiguration blocks that implement all the transceiver and PLL logic. You can either directly copy the relevant sections of the design, or use the design as a reference. The design generates a DisplayPort Intel Arria 10 FPGA IP Design Example and then adds many of the files generated directly into the compile list used by the Intel Quartus Prime project. These include:

- Files to create parameterized IP instances for transceivers, PLLs and reconfig blocks.
- Verilog HDL files to connect these IPs into the higher level receiver PHY, transmitter PHY, and Transceiver Reconfiguration Arbiter blocks
- Synopsys design constraint (SDC) files to set the relevant timing constraints.

ISO 9001:2015 Registered

(intel®)

Features of the 8K DisplayPort Video Format Conversion Design Example

- Input:
  - DisplayPort 1.4 connectivity supports resolutions from 720x480 up to 3840x2160 at any frame rate up to 60 fps, and resolutions up to 7680x4320 at 30 fps.
  - Hot-plug support.
  - Support for both RGB and YCbCr (4:4:4, 4:2:2 and 4:2:0) color formats at the input.
  - Software automatically detects the input format and sets up the processing pipeline appropriately.
- Output:
  - DisplayPort 1.4 connectivity selectable (via DIP switches) for either 1080p, 1080i or 2160p resolution at 60 fps, or 2160p at 30 fps.
  - Hot-plug support.
  - DIP switches to set the required output color format to RGB, YCbCr 4:4:4, YCbCr 4:2:2, or YCbCr 4:2:0.
- Single 10-bit 8K RGB processing pipeline with software configurable scaling and frame rate conversion:
  - 12-tap Lanczos down-scaler.
  - 16-phase, 4-tap Lanczos up-scaler.
  - Triple buffering video frame buffer provides frame rate conversion.
  - Mixer with alpha-blending allows OSD icon overlay.

**ISO
9001:2015
Registered**

intel

Getting Started with the 8K DisplayPort Video Format Conversion Design Example

Hardware and Software Requirements

The 8K DisplayPort Video Format Conversion Design Example requires specific hardware and software.

Hardware:

- Intel Arria 10 GX FPGA Development Kit, including the DDR4 Hilo Daughter Card
- Bitec DisplayPort 1.4 FMC daughter card (revision 11)
- DisplayPort 1.4 source that produces up to 3840x2160p60 or 7680x4320p30 video
- DisplayPort 1.4 sink that displays up to 3840x2160p60 video
- VESA certified DisplayPort 1.4 cables.

Software:

- Windows or Linux OS
- The Intel Quartus Prime Design Suite v19.2, which includes:
  — Intel Quartus Prime Pro Edition
  — Platform Designer
  — Nios® II EDS
  — Intel FPGA IP Library (including the Video and Image Processing Suite)

The design only works with this version of Intel Quartus Prime.

Downloading and Installing the Intel 8K DisplayPort Video Format Conversion Design Example

The design is available on the Intel Design Store.

1. Download the archived project file `udx10_dp.par`.
2. Extract the Intel Quartus Prime project from the archive:
   a. Open Intel Quartus Prime Pro Edition.
   b. Click **File ➤ Open Project**.
      The **Open Project** window opens.
   c. Navigate to and select the `udx10_dp.par` file.
   d. Click **Open**.
   e. In the **Open Design Template** window, set the **Destination** folder to the desired location for the extracted project.

The entries for the design template file and project name should be correct and you need not change them.

f. Click **OK**.

Design Files for the Intel 8K DisplayPort Video Format Conversion Design Example

Table 1.   Design Files

File or Folder Name	Description
ip	Contains the IP instance files for all the Intel FPGA IP instances in the design: • A DisplayPort IP (transmitter and receiver) • A PLL that generates clocks at the top level of the design • All the IP that make up the Platform Designer system for the processing pipeline. .
master_image	Contains `pre_compiled.sof`, which is a precompiled board programming file for the design.
non_acds_ip	Contains source code for additional IP in this design that Intel Quartus Prime does not include.
sdc	Contains an SDC file that describes the additional timing constraints that this design requires. The SDC files included automatically with the IP instances do not handle these constraints.
software	Contains source code, libraries, and build scripts for the software that runs on the embedded Nios II processor to control the high-level functionality of the design.
udx10_dp	A folder into which Intel Quartus Prime generates output files for the Platform Designer system. The `udx10_dp.sopcinfo` output file allows you to generate the memory initialization file for the Nios II processor software memory. You need not first generate the full Platform Designer system.
non_acds_ip.ipx	This IPX file declares all of the IP in the `non_acds_ip` folder to Platform Designer so it appears in the IP Library.
README.txt	Brief instructions to build and run the design.
top.qpf	The Intel Quartus Prime project file for the design.
top.qsf	The Intel Quartus Prime project settings file for the design. This file lists all the files required to build the design, along with the pin assignments and a number of other project settings.
top.v	The top-level Verilog HDL file for the design.
udx10_dp.qsys	The Platform Designer system that contains the video processing pipeline, the Nios II processor, and its peripherals.

Compiling the 8K DisplayPort Video Format Conversion Design Example

Intel provides a precompiled board programming file for the design in the `master_image` directory (`pre_compiled.sof`) to allow you to run the design without running a full compilation.

STEPS:

1. In the Intel Quartus Prime software, open the `top.qpf` project file. The downloaded archive creates this file when you unzip the project.

2. Click **File ➤ Open** and select `ip/dp_rx_tx/dp_rx_tx.ip`.
   The parameter editor GUI for the DisplayPort IP opens, showing the parameters for the DisplayPort instance in the design.

3. Click **Generate Example Design** (not **Generate**).

4. When the generation completes, close the parameter editor.

5. In File Explorer, navigate to the software directory and unzip the `vip_control_src.zip` archive to generate the `vip_control_src` directory.

6. In a BASH terminal, navigate to `software/script` and run the shell script `build_sw.sh`.
   The script builds the Nios II software for the design. It creates both an `.elf` file that you can download to the board at run time, and a `.hex` file to compile into the board programming `.sof` file.

7. In the Intel Quartus Prime software, click **Processing ➤ Start Compilation**.

   • Intel Quartus Prime generates the `udx10_dp.qsys` Platform Designer system.

   • Intel Quartus Prime sets the project to `top.qpf`.

   The compilation creates `top.sof` in the `output_files` directory when it completes.

Viewing and Regenerating the Platform Designer System

1. Click **Tools ➤ Platform Designer**.

2. Select *system name*`.qsys` for the Platform Designer system option.

3. Click **Open**.
   Platform Designer opens the system.

4. Review the system.

5. Regenerate the system:

   a. Click **Generate HDL...**.

   b. In the Generation Window, turn on **Clear output directories for selected generation targets**.

   c. Click **Generate**

Compiling the 8K DisplayPort Video Format Conversion Design Example with the Nios II Software Build Tools for Eclipse

You set up an interactive Nios II Eclipse workspace for the design to produce a workspace that uses the same folders that the build script uses. If you previously run the build script, you should delete the `software/vip_control` and `software/vip_control_bsp` folders before creating the Eclipse workspace. If you re-run the build script at any point it overwrites the Eclipse workspace.

STEPS:

1. Navigate to the software directory and unzip the `vip_control_src.zip` archive to generate the `vip_control_src` directory.

2. In the installed project directory, create a new folder and name it `workspace`.

3. In the Intel Quartus Prime software, click **Tools ➤ Nios II Software Build Tools for Eclipse**.

   a. In the **Workspace Launcher** window, select the `workspace` folder you created.

   b. Click **OK**.

4. In the **Nios II - Eclipse** window, click **File ➤ New ➤ Nios II Application and BSP from Template**.

   The **Nios II Application and BSP from Template** dialog box appears.

   a. In the **SOPC Information File** box, select the `udx10_dp/udx10_dp.sopcinfo` file.
      The Nios II SBT for Eclipse fills in the CPU name with the processor name from the .sopcinfo file.

   b. In the Project name box, type `vip_control`.

   c. Select **Blank Project** from the Templates list.

   d. Click **Next**.

   e. Select **Create a new BSP project** based on the application project template with the project name **vip_control_bsp**.

   f. Turn on **Use default location**.

   g. Click **Finish** to create the application and the BSP based on the .sopcinfo file.

   After the BSP generates, the vip_control and vip_control_bsp projects appear in the Project Explorer tab.

5. In Windows Explorer, copy the contents of the `software/vip_control_src` directory to the newly created `software/vip_control` directory.

6. In the **Project Explorer** tab of the **Nios II - Eclipse** window, right click on the **vip_control_bsp** folder and select **Nios II > BSP Editior**.

   a. Select **None** from the drop-down menu for **sys_clk_timer**.

   b. Select **cpu_timer** from the drop-down menu for **timestamp_timer**.

   c. Turn on **enable_small_c_library**.

   d. Click **Generate**.

   e. When generation completes, click **Exit**.

7. In the **Project Explorer** tab, right-click the `vip_control` directory and click **Properties**.

   a. In the **Properties for vip_control** window, expand **Nios II Application properties** and click **Nios II Application Paths**.

   b. Click **Add...** next to **Library Projects**.

   c. In the **Library Projects** window, navigate to the `udx10.dp\spftware\vip_control_src` directory and select the `bkc_dprx.syslib` directory.

   d. Click **OK**.
      A message appears **Convert to a relative path**. Click **Yes**.

   e. Repeat steps 7.b on page 8 and 7.c on page 8 for the `bkc_dptx.syslib` and `bkc_dptxll_syslib` directories

   f. Click **OK**.

8. Select **Project ➤ Build All** to generate the file `vip_control.elf` in the `software/vip_control` directory.

9. Build the `mem_init` file for the Intel Quartus Prime compilation:

   a. Right click `vip_control` in the **Project Explorer** window.

   b. Select **Make Targets ➤ Build...**.

c. Select **mem_init_generate**.

d. Click **Build**.

The Intel Quartus Prime software generates the `udx10_dp_onchip_memory2_0_onchip_memory2_0.hex` file in the `software/vip_control/mem_init` directory.

10. With the design running on a connected board, run the `vip_control.elf` programming file created by the Eclipse build.

a. Right click `vip_control` folder in the **Project Explorer** tab of the **Nios II - Eclipse** window.

b. Selecting **Run As ➤ Nios II Hardware**. If you have a Nios II terminal window open, close it before downloading the new software.

Setting up the Intel Arria 10 GX FPGA Development Kit

Describes how to set up the kit to run the 8K DisplayPort Video Format Conversion Design Example.

Figure 1.   Intel Arria 10 GX Development Kit with HiLo Daughter Card

The figure shows the board with the blue heat sink removed to show the positioning of the DDR4 Hilo card. Intel recommends that you do not run the design without the heat sink in position.
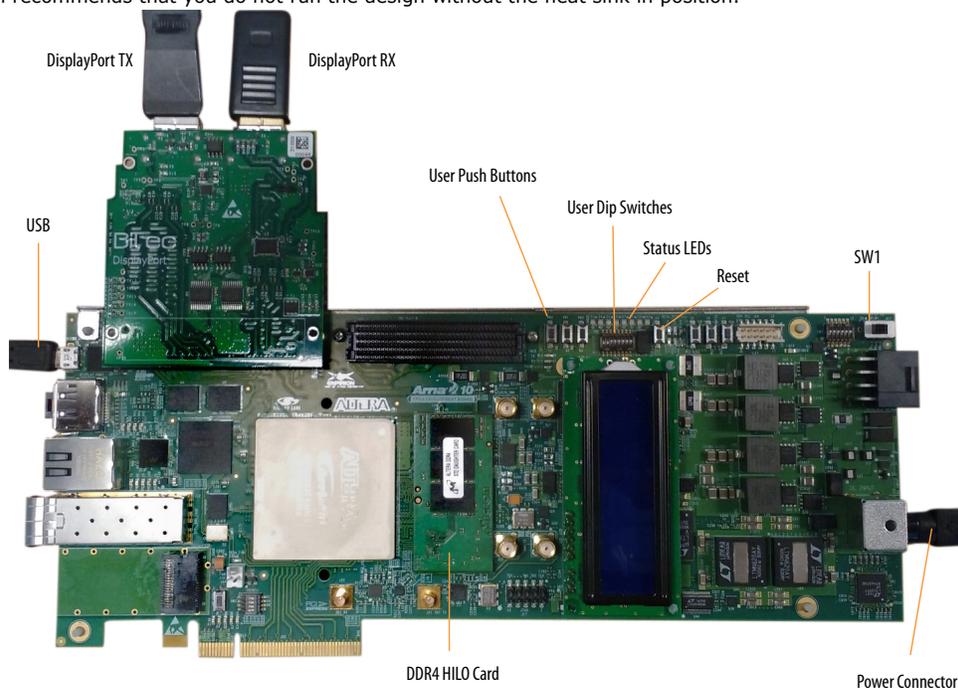
STEPS:

1. Fit the Bitec DisplayPort 1.4 FMC card to the development board using FMC Port A.

2. Ensure the power switch (SW1) is turned off, then connect the power connector.

3. Connect a USB cable to your computer and to the MicroUSB Connector (J3) on the development board.

4. Attach a DisplayPort 1.4 cable between the DisplayPort source and the Receiver port of the Bitec DisplayPort 1.4 FMC card and ensure the source is active.

5. Attach a DisplayPort 1.4 cable between the DisplayPort display and the Transmitter port of the Bitec DisplayPort 1.4 FMC card and ensure the display is active.

6. Turn on the board using SW1.

Board Status LEDs, Push Buttons and DIP Switches

The Intel Arria 10 GX FPGA Development Kit has eight status LEDs (with both green and red emitters), three user push buttons and eight user DIP switches. The 8K DisplayPort Video Format Conversion Design Example illuminates the LEDs to indicate the state of the DisplayPort receiver link. The push buttons and DIP switches allow you to alter design settings.

Status LEDs

Table 2.   Status LEDs

The table lists the status that each LED indicates. Each LED position has both red and green indicators that can illuminate independently. Any LED glowing orange means that both the red and green indicators are on.

LED	Description
Red LEDs	
0	DDR4 EMIF calibration in progress.
1	DDR4 EMIF calibration failed.
7:2	Unused.
Green LEDs	
0	Illuminates when DisplayPort receiver link training completes successfully, and the design receives stable video.
5:1	DisplayPort receiver lane count: 00001 = 1 lane 00010 = 2 lanes 00100 = 4 lanes
7:6	DisplayPort receiver lane speed: 00 = 1.62 Gbps 01 = 2.7 Gbps 10 = 5.4 Gbps 11 = 8.1 Gbps

User Push Buttons

User push button 0 controls the display of the Intel logo in the top right-hand corner of the output display. At startup, the design enables the display of the logo. Pressing push button 0 toggles the enable for the logo display.

User push button 1 controls the scaling mode of the design. When a source or sink is hot-plugged the design defaults to either:

- Passthrough mode, if the input resolution is less than or equal to the output resolution

- Downscale mode, if the input resolution is greater than the output resolution

Each time you press user push button 1 the design swaps to the next scaling mode (passthrough > upscale, upscale > downscale, downscale > passthrough).

User push button 2 is unused.

User DIP Switches

The DIP switches control the optional Nios II terminal printing and the settings for the output video format driven through the DisplayPort transmitter.

Table 3. **DIP Switches**

The table lists the function of each DIP switch. The DIP switches, numbered 1 to 8 (not 0 to 7), match the numbers printed on the switch component. To set each switch to ON, move the white switch towards the LCD and away from the LEDs on the board.

Switch	Function
1	Enables Nios II terminal printing when set to ON.
2	Set output bits per color: OFF = 8 bit ON = 10 bit
4:3	Set output color space and sampling: SW4 OFF, SW3 OFF = RGB 4:4:4 SW4 OFF, SW3 ON = YCbCr 4:4:4 SW4 ON, SW3 OFF = YCbCr 4:2:2 SW4 ON, SW3 ON = YCbCr 4:2:0
6:5	Set output resolution and frame rate: SW4 OFF, SW3 OFF = 4K60 SW4 OFF, SW3 ON = 4K30 SW4 ON, SW3 OFF = 1080p60 SW4 ON, SW3 ON = 1080i60
8:7	Unused

Running the 8K DisplayPort Video Format Conversion Design Example

You must download the compiled `.sof` file for the design to the Intel Arria 10 GX FPGA Development Kit to run the design.

STEPS:

1. In the Intel Quartus Prime software, click **Tools ➤ Programmer**.

2. In the **Programmer** window, click **Auto Detect** to scan the JTAG chain and discover the connected devices.

   If a pop-up window appears asking you to update the Programmer's device list, click **Yes**.

3. In the device list, select the row labeled 10AX115S2F45.

4. Click Change File…

   - To use the precompiled version of the programming file that Intel includes as part of the design download, select `master_image/pre_compiled.sof`.

   - To use your programming file created by the local compile, select `output_files/top.sof`.

5. Turn on **Program/Configure** in the 10AX115S2F45 row of the device list.

6. Click **Start**.
   When the programmer completes, the design runs automatically.

7. Open a Nios II terminal to receive the output text messages from the design, otherwise the design locks up after a number of switch changes (only if you set user DIP switch 1 to ON).

   a. Open a terminal window and type `nios2-terminal`

   b. Press **Enter**.

When the design is running, the output appears on the display, even if no source is connected at the input. With no source, the output is a black screen with the Intel logo in the top right-hand corner of the screen.

(intel)

Functional Description of the 8K DisplayPort Video Format Conversion Design Example

The Platform Designer system, udx10_dp.qsys, contains the DisplayPort receiver and transmitter protocol IP, the video pipeline IP, and the Nios II processor components. The design connects the Platform Designer system to the DisplayPort receiver and transmitter PHY logic (which contains the interface transceivers) and the transceiver reconfiguration logic at the top level in a Verilog HDL RTL design file (top.v). The design comprises a single video processing path between the DisplayPort input and the DisplayPort output.

Figure 2.   Block Diagram

The diagram shows the blocks in the 8K DisplayPort Video Format Conversion Design Example. The diagram does not show some of the generic peripherals connected to the Nios II, the Avalon-MM between the Nios II processor, and the other components of the system. The design accepts video from a DisplayPort source on the left, processes the video through the video pipeline from left to right before passing the video out to the DisplayPort sink on the right.
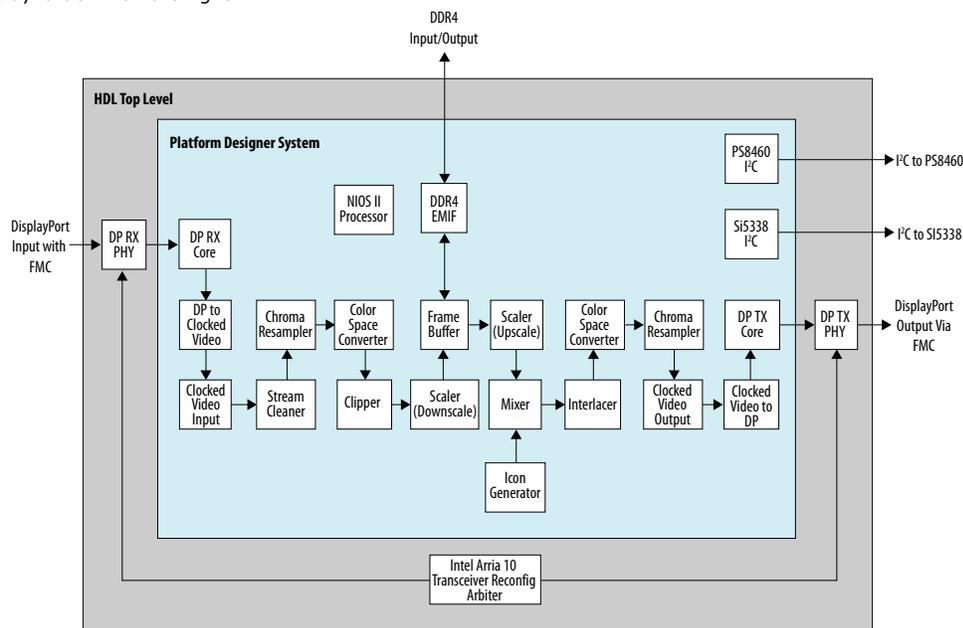
DisplayPort Receiver PHY and DisplayPort Receiver IP

The Bitec DisplayPort FMC card provides a buffer for the DisplayPort 1.4 signal from the DisplayPort source. The combination of DisplayPort Receiver PHY and DisplayPort Receiver IP decodes the incoming signal to create a video stream. The DisplayPort receiver PHY contains the transceivers to deserialize the incoming data and the DisplayPort receiver IP decodes the DisplayPort protocol. The combined DisplayPort

Receiver IP processes the incoming DisplayPort signal without any software. The resulting video signal from the DisplayPort receiver IP is a native packetized streaming format. The design configures the DisplayPort receiver for 10-bit output.

DisplayPort to Clocked Video IP

The packetized streaming data format output by the DisplayPort receiver is not directly compatible with the clocked video data format that the Clocked Video Input IP expects. The DisplayPort to Clocked Video IP is a custom IP for this design. It converts the DisplayPort output into a compatible clocked video format that you can connect directly to the Clocked Video Input. The DisplayPort to Clocked Video IP can modify the wire signaling standard and can alter the ordering of the color planes within each pixel. The DisplayPort standard specifies color ordering that is different than the Intel video pipeline IP ordering. The Nios II processor controls the color swap. It reads the current color space for transmission from the DisplayPort receiver IP with its Avalon-MM slave interface. It directs the DisplayPort to Clocked Video IP to apply the appropriate correction with its Avalon-MM slave interface.

Clocked Video Input

The clocked video input processes the clocked video interface signal from the DisplayPort to Clocked Video IP and converts it to Avalon-ST Video signal format. This signal format strips all horizontal and vertical blanking information from the video leaving only active picture data. The IP packetizes it as one packet per video frame. It also adds additional metadata packets (referred to as control packets) that describe the resolution of each video frame. The Avalon-ST Video stream through the processing pipe is four pixels in parallel, with three symbols per pixel. The clocked video input provides clock crossing for the conversion from the variable rate clocked video signal from the DisplayPort receiver IP to the fixed clock rate (300 MHz) for the video IP pipeline.

Stream Cleaner

The stream cleaner ensures that the Avalon-ST Video signal passing to the processing pipeline is error free. Hot plugging of the DisplayPort source can cause the design to present incomplete frames of data to the clocked video input IP and to generate errors in the resulting Avalon-ST Video stream. The size of the packets containing the video data for each frame then do not match the size reported by the associated control packets. The stream cleaner detects these conditions and adds additional data (grey pixels) to the end of the offending video packets to complete the frame and match the specification in the control packet.

Chroma Resampler (Input)

The video data that the design receives at the input from DisplayPort may be 4:4:4, 4:2:2, or 4:2:0 chroma sampled. The input chroma resampler takes the incoming video in any format and converts it to 4:4:4 in all cases. To provide higher visual quality, the chroma resampler uses the most computationally expensive filtered algorithm. The Nios II processor reads the current chroma sampling format from the DisplayPort receiver IP via its Avalon-MM slave interface. It communicates the format to the chroma resampler via its Avalon-MM slave interface.

Color Space Converter (Input)

The input video data from DisplayPort may use either the RGB or YCbCr color space. The input color space converter takes the incoming video in whatever format it arrives and converts it to RGB in all cases. The Nios II processor reads the current color space

from the DisplayPort receiver IP with its Avalon-MM slave interface; it loads the correct conversion coefficients to the chroma resampler through its Avalon-MM slave interface.

Clipper

The clipper selects an active area from the incoming video stream and discards the remainder. The software control running on the Nios II processor defines the region to select. The region depends on the resolution of the data received at the DisplayPort source and the output resolution and scaling mode. The processor communicates the region to the Clipper through its Avalon-MM slave interface.

Scaler

The design applies scaling to the incoming video data according to the input resolution received, and the output resolution you require. You may also select between three scaling modes (upscale, downscale and passthrough). Two Scalar IPs provide the scaling functionality: one implements any required downscaling; the other implements upscaling. The design requires two scalers.

- When the scaler implements a downscale, it does not produce valid data on every clock cycle at its output. For example, if implementing a 2x downscale ratio, the valid signal at the output is high every other clock cycle while the design receives each even numbered input line, and then low for the entirety of the odd numbered input lines. This bursting behavior is fundamental to the process of reducing the data rate at the output, but is incompatible with the downstream Mixer IP, which generally expects a more consistent data rate to avoid underflow at the output. The design requires the Frame Buffer between any downscale and mixer. The Frame Buffer allows the Mixer to read the data at the rate it requires.

- When the scaler implements an upscale, it produces valid data on every clock cycle, so the following mixer has no issues. However, it may not accept new input data on every clock cycle. Taking a 2x upscale as an example, on the even numbered output lines it accepts a new beat of data every other clock cycle, then accepts no new input data on the odd numbered output lines. However, the upstream Clipper may produce data at an entirely different rate if it is applying a significant clip (e.g. during a zoom-in). Therefore, a Clipper and upscale must generally be separated by a Frame Buffer, requiring the Scaler to sit after the Frame Buffer in the pipeline. The Scaler must sit before the Frame Buffer for downscales, so the design implements two separate scalers either side of the Frame Buffer: one for upscale; the other for downscale.

Two Scalers also reduce the maximum DDR4 bandwidth required by the Frame Buffer. You must always apply downscales before the Frame Buffer, minimizing the data rate on the write side. Always apply upscales after the Frame Buffer, which minimizes the data rate on the read side.

Each Scaler gets the required input resolution from the control packets in the incoming video stream, while the Nios II processor with the Avalon-MM slave interface sets the output resolution for each Scaler.

Frame Buffer

The frame buffer uses the DDR4 memory to perform triple buffering that allows the video and image processing pipeline to perform frame rate conversion between the incoming and outgoing frame rates. The design can accept any input frame rate, but the total pixel rate must not exceed 1 giga pixels per second. The Nios II software sets the output frame rate to either 30 or 60 fps, according to the output mode you select.

The output frame rate is a function of the Clocked Video Output settings and the output video pixel clock. The backpressure that the Clocked Video Output applies to the pipeline determines the rate at which the read side of the Frame Buffer pulls video frames from the DDR4.

Mixer

The mixer generates a fixed size black background image that the Nios II processor programs to match the size of the current output image. The mixer has two inputs. The first input connects to the upscaler to allow the design to show the output from the current video pipeline. The second input connects to the icon generator block. The design only enables the mixer's first input when it detects active, stable video at the clocked video input. Therefore, the design maintains a stable output image at the output while hot-plugging at the input. The design alpha blends the second input to the mixer, connected to the icon generator, over both the background and video pipeline images with 50% transparency.

Color Space Converter (Output)

The output color space converter transforms the input RGB video data to either RGB or YCbCr color space based on the runtime setting from software.

Chroma Resampler (Output)

The output chroma resampler converts the format from 4:4:4 to one of 4:4:4, 4:2:2, or 4:2:0 formats. The software sets the format. The output chroma resampler also uses filtered algorithm to achieve high-quality video.

Clocked Video Output

The clocked video output converts the Avalon-ST Video stream to the clocked video format. The clocked video output adds horizontal and vertical blanking and synchronization timing information to the video. The Nios II processor programs the relevant settings in the clocked video output depending on the output resolution and frame rate that you request. The clocked video output converts the clock, crossing from the fixed 300 MHz pipeline clock to the variable rate of the clocked video.

Clocked Video to DisplayPort

The DisplayPort transmitter component accepts data formatted as clocked video. Differences in the wire signaling and declaration of the conduit interfaces in Platform Designer prevent you connecting the Clocked Video Output directly to the DisplayPort transmitter IP. The Clocked Video to DisplayPort component is design-specific custom IP to provide the simple conversion required between the Clocked Video Output and the DisplayPort transmitter IP. It also swaps the ordering of the color planes in each pixel to account for the different color formatting standards used by Avalon-ST Video and DisplayPort.

DisplayPort Transmitter IP and DisplayPort Transmitter PHY

The DisplayPort transmitter IP and DisplayPort transmitter PHY together work to convert the video stream from clocked video to a compliant DisplayPort stream. The DisplayPort transmitter IP handles the DisplayPort protocol and encodes the valid DisplayPort data, while the DisplayPort transmitter PHY contains the transceivers and creates the high-speed serial output.

Send Feedback

Nios II Processor and Peripherals

The Platform Designer system contains a Nios II processor, which manages the DisplayPort receiver and transmitter IPs and the runtime settings for the processing pipeline. The Nios II processor connects to these basic peripherals:

- An on-chip memory to store the program and its data.

- A JTAG UART to display software printf output (via a Nios II terminal).

- A system timer to generate millisecond level delays at various points in the software, as required by the DisplayPort specification of minimum event durations.

- LEDs to display system status.

- Push-button switches to allow switching between scaling modes and to enable and disable display of the Intel logo.

- DIP switches to allow switching of the output format and to enable and disable the printing of messages to a Nios II terminal.

Hot-plug events on both the DisplayPort source and sink fire interrupts that trigger the Nios II Processor to configure the DisplayPort transmitter and pipeline correctly. The main loop in the software code also monitors that values on the push-buttons and DIP switches and alters the pipeline setup accordingly.

I²C Controllers

The design contains two I²C controllers (Si5338 and PS8460) to edit the settings of three of the other components on the Intel Arria 10 10 GX FPGA Development Kit.

Two Si5338 clock generators on the Intel Arria 10 GX FPGA Development Kit connect to the same I²C bus. The first generates the reference clock for the DDR4 EMIF. By default, this clock is set to 100 MHz for use with 1066 MHz DDR4, but this design runs the DDR4 at 1200 MHz, which requires a reference clock of 150 MHz. At startup the Nios II processor, via the I²C controller peripheral, changes the settings in the register map of the first Si5338 to increase the speed of the DDR4 reference clock to 150MHz. The second Si5338 clock generator generates the vid_clk for the clocked video interface between the pipeline and the DisplayPort transmitter IP. You must adjust the speed of this clock for each different output resolution and frame rate supported by the design. You can adjust the speed at run time when the Nios II processor requires.

The Bitec DisplayPort 1.4 FMC daughter card makes use of the Parade PS8460 jitter cleaning repeater and retimer. At startup the Nios II processor edits the default settings of this component to meet the requirements of the design.

AN-889 | 2019.05.30

Software Description

The 8K DisplayPort Video Format Conversion Design Example includes IP from the Intel Video and Image Processing Suite and the DisplayPort interface IP

All these IPs can process frames of data without any further intervention when setup correctly. You must implement external high-level control to setup the IPs to begin with and when the system changes, e.g. DisplayPort receiver or transmitter hot-plug events or user push button activity. In this design, a Nios II processor, running bespoke control software, provides the high-level control.

At startup the software:

- Sets the DDR4 ref clock to 150 MHz to allow for 1200 MHz DDR speed, then resets external memory interface IP to recalibrate on the new reference clock.

- Sets up the PS8460 DisplayPort repeater and retimer.

- Initializes the DisplayPort receiver and transmitter interfaces.

- Initializes the processing pipeline IPs.

When initialization is complete the software enters a continuous while loop, checking for, and reacting to, a number of events.

Changes to the Scaling Mode

The design supports three basic scaling modes; passthrough, upscale, and downscale. In passthrough mode the design does no scaling of the input video, in upscale mode the design upscales input video, and in downscale mode the design downscales input video.

The four blocks in the processing pipeline; the Clipper, the downscaler, the upscaler and the Mixer determine the presentation of the final output in each mode. The software controls the settings of each block depending on the current input resolution, output resolution, and the scaling mode that you select. In most cases, the Clipper passes the input through unaltered, and the Mixer background size is the same size as the final, scaled version of the input video. However, if the input video resolution is greater than the output size, it is not possible to apply an upscale to the input video without first clipping it. If the input resolution is less than the output the software cannot apply a downscale without applying a Mixer background layer that is larger than the input video layer, which adds black bars around the output video.

Table 4. Processing Block Pipelines

This table lists the action of the four processing pipeline blocks in each of the nine combinations of scaling mode, input resolution and output resolution.

Mode	in > out	in = out	in < out
Passthrough	Clip to output size No downscale	No clip No downscale	No clip No downscale
			continued...

Mode	in > out	in = out	in < out
	No upscale No black border	No upscale No black border	No upscale Black border pads to output size
Upscale	Clip to 2/3 output size No downscale Upscale to output size No black border	Clip to 2/3 output size No downscale Upscale to output size No black border	No clip No downscale Upscale to output size No black border
Downscale	No clip Downscale to output size No upscale No black border	No clip Downscale to output size No upscale No black border	No clip Downscale to 2/3 input size No upscale Black border pads to output size

Change between modes by pressing user push button 1. The software monitors the values on the push buttons on each run through the loop (it does a software debounce) and configures the IPs in the processing pipeline appropriately.

Changes at the DisplayPort Input

On each run through the loop the software polls the status of the Clocked Video Input, looking for changes in the stability of the input video stream. The software considers the video is stable if:

- The Clocked Video Input reports that the clocked video is successfully locked.
- The input resolution and color space has no changes since the previous run through the loop.

If the input was stable but it has lost lock or the properties of the video stream have changed, the software stops the Clocked Video Input sending video through the pipeline. It also sets the Mixer to stop displaying the input video layer. The output remains active (showing a black screen and the Intel logo) during any receiver hot-plug events or resolution changes.

If the input was not stable but is now stable, the software configures the pipeline to display the new input resolution and color space, it restarts the output from the CVI, and it sets the Mixer to display the input video layer again. The re-enabling of the mixer layer is not immediate as the Frame Buffer may still be repeating old frames from a previous input and the design must clear these frames. Then you can re-enable the display to avoid glitching. The frame buffer keeps a count of the number of frames read from the DDR4, which the Nios II processor can read. The software samples this count when the input becomes stable and re-enables the Mixer layer when the count has increased by four frames, which ensures the design flushes out any old frames from the buffer.

DisplayPort transmitter Hot-plug Events

Hot-plug events at the DisplayPort transmitter fire an interrupt within the software that sets a flag to alert the main software loop of a change in the output. When the design detects a transmitter hot plug, the software reads the EDID for the new display to determine which resolutions and color spaces its supports. If you set the DIP switches to a mode that the new display cannot support, the software falls back to a less demanding display mode. It then configures the pipeline, DisplayPort transmitter IP, and the Si5338 part that is generating the transmitter vid_clk for the new output mode. When the input sees changes, the Mixer layer for the input video does not

display as the software edits settings for the pipeline. The software does not re-enable the display until after four frames when the new settings pass through the frame buffer.

Changes to User DIP Switch Settings

The positions of user DIP switches 2 to 6 control the output format (resolution, frame rate, color space and bits per color) driven through the DisplayPort transmitter. When the software detects changes on these DIP switches, it runs through a sequence that is virtually identical to a transmitter hot plug. You need not query the transmitter EDID as it does not change.

Revision History for AN 889: 8K DisplayPort Video Format Conversion Design Example

Table 5. **Revision History for AN 889: 8K DisplayPort Video Format Conversion Design Example**

Document Version	Changes
2019.05.30	Initial release.