AN 776: UHD HDMI 2.0 Video Format Conversion Design Example

Subscribe
Send Feedback

AN-776 | 2021.04.15
Latest document on the web: PDF | HTML

Contents

1. About the UHD HDMI 2.0 Video Format Conversion Design Example

The ultra-high-definition (UHD) HDMI 2.0 video format conversion design example integrates the Intel HDMI 2.0 video connectivity IP with a video processing pipeline based on Intel® FPGA IP from the Intel Video and Image Processing Suite.

The design delivers high-quality scaling, color space conversion, and frame rate conversion for video streams up to 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 4K ready IP from the Video and Image Processing Suite in the Intel Quartus® Prime Design Suite.

Related Information

Intel HDMI IP Core User Guide

1.1. UHD HDMI 2.0 Video Format Conversion Design Example Features

- Input:
  - HDMI 2.0 connectivity supports from 720x480 up to 3840x2160 resolution at any frame rate up to and including 60 fps.
  - Input hot-plug support.
  - Supports both RGB and YCbCr (4:4:4, 4:2:2 and 4:2:0) colour formats at the input.
  - Supports input at 8 and 10 bits per color
  - Software automatically detects the input format and sets up the processing pipeline appropriately.
- Output:
  - HDMI 2.0 connectivity selectable for either 1080p, 1080i or 2160p resolution at 60 fps, or 2160p at 30 fps
  - Output hot-plug support
  - DIP switches set the required output color format to RGB, YCbCr-4:4:4 or YCbCr-4:2:2, or YCbCr 4:2:0)
  - DIP switches set the output to 8 or 10 bits per color
- Single 10-bit RGB processing pipeline with software configurable scaling and frame rate conversion:
  - 12 tap Lanczos down-scaler
  - 16 phase, 4 tap Lanczos up-scaler
  - Triple buffer video frame buffer provides frame rate conversion
  - Mixer with alpha-blending allowing OSD icon overlay

Related Information

- Avalon Interface Specifications
  Information about Avalon memory-mapped and Avalon streaming interfaces
- Video and Image Processing Suite User Guide
  Information about Avalon streaming video interface
- AN 556: Using the Design Security Features in Intel FPGAs

2. UHD HDMI 2.0 Video Format Conversion Design Example Getting Started

2.1. Hardware and Software Requirements for the UHD HDMI 2.0 Video Format Conversion Design Example

The design requires the following hardware:

- Intel Arria 10 GX FPGA Development Kit
- Bitec HDMI 2.0 FMC daughter card, revision 11
- HDMI 2.0 source that produces up to 3840x2160p60 RGB and YCbCr unencrypted video
- HDMI 2.0 sink that displays up to 3840x2160p60 RGB and YCbCr video
- Intel recommends VESA certified HDMI 2.0 cables

The design requires the following software:

- Windows or Linux OS
- The Intel Quartus Prime Design Suite v20.4 that includes:
  - Intel Quartus Prime Pro Edition
  - Platform Designer
  - Nios® II EDS
  - Intel FPGA IP Library (including the Video and Image Processing Suite)

Related Information

- Arria 10 GX FPGA Development Kit
- Bitec HDMI FMC Daughter Card

2.2. Downloading and Installing the UHD HDMI 2.0 Video Format Conversion Design Example

1. Download the project file `udx10_hdmi_204.zip` from the Intel Resource and Design Center.
2. Extract the contents of the `.zip` archive.
   The directory contains the Intel Quartus Prime `top.qsf` and `top.qpf` files and all other files for the design.

Related Information

Intel Resource and Design Center

2.2.1. Installation Files for the UHD HDMI 2.0 Video Format Conversion Design Example

Table 1. Files and Directories

File or Directory Name	Description
`ip`	Contains the IP instance files for all the Intel FPGA IP in the design. Including IP instances for : • An HDMI core (transmit and receive) • A PLL that generates clocks at the top level of the design • All the IPs in the Platform Designer system for the processing pipeline.
`master_image`	Contains `pre_compiled.sof` – a precompiled board programming file for the design.
`non_acds_ip`	Contains source code for additional IP in this design that the Intel Quartus Prime Design Suite does not include: • Source for an icon generator • A reset synchronizer • Interface components to allow direct connection between HDMI and Clocked Video IPs.
`sdc`	Contains an SDC file that describes the additional timing constraints required by this design that are not handled by SDC files included automatically with the IP instances.
`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.
`non_acds_ip.ipx`	This `.ipx` file declares all the IP in the `non_acds_ip` directory to Platform Designer so it appears in the IP Library
`pre_compile_flow.tcl`	A Tcl script that the Intel Quartus project uses before compilation to automate the required build steps
`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, the pin assignments, and other project settings.
`top.v`	The top level Verilog HDL file for the design.
`udx10_hdmi.qsys`	The Platform Designer system containing the video processing pipeline and the Nios II processor and its peripherals.

2.3. Setting up the Intel Arria 10 FPGA development board

To run the UHD Video Format Conversion Design Example:

1. Fit the Bitec HDMI 2.0 FMC card to the Intel Arria 10 GX FPGA development board using FMC Port A.
2. Ensure the power switch (SW1) is turned off, then connect the power connector.
3. Connect a USB Blaster II download cable to your computer and to the MicroUSB Connector (J3) on the Intel Arria 10 GX FPGA development board.
4. Attach a HDMI 2.0 cable between the HDMI video source and the Rx port of the Bitec HDMI 2.0 FMC card and ensure the source is active.
5. Attach a HDMI 2.0 cable between the HDMI display and the Tx port of the Bitec HDMI 2.0 FMC card and ensure the display is active.
6. Turn on board using SW1.

2.3.1. Board Status Lights, DIP Switches, and Pushbuttons

The Intel Arria 10 GX FPGA Development board has eight status lights, each of which contains both red and green LEDs, and three push-buttons that the Intel Arria 10 UHD design uses.

Figure 1. Location of Board Status Lights, DIP Switches, and Pushbuttons

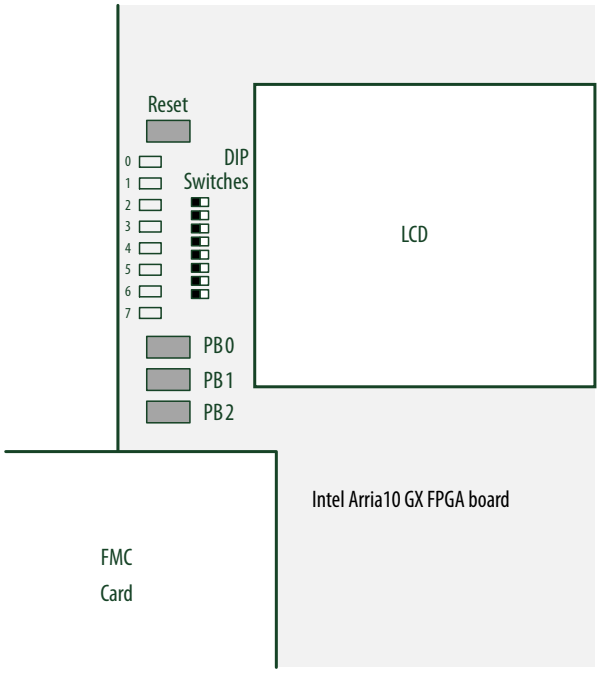


Table 2. Status Lights

While the design is running on the Intel Arria 10 GX FPGA development board, the board's status lights display the current status of the system. Each status light position contains a combined red and green LED.

LED	Description
Green LEDs	
0	HDMI Rx IO PLL locked
1	HDMI Rx ready

continued...

LED	Description
2	HDMI Rx locked
3	HDMI Rx oversample
4	HDMI Tx IO PLL locked
5	HDMI Tx ready
6	HDMI Tx PLL locked
7	HDMI Tx oversample
Red LEDs	
0	DDR4 EMIF calibration in progress
1	DDR4 EMIF calibration fail
7:2	Unused.

Table 3. Push Buttons

Pushbutton	Description
PB0	Controls the display of the Intel icon in the top right-hand corner of the output display. At startup display of the icon is enabled. Pressing PB0 toggles the enable for the icon display.
PB1	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 PB1 the design swaps to the next scaling mode (passthrough > upscale, upscale > downscale, downscale > passthrough). .
PB2	Unused

Table 4. DIP Switches

The user DIP switches control the optional Nios II terminal printing and the settings for the output video format driven through the HDMI TX. The DIP switches are numbered 1 to 8 (not 0 to 7) to 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 user LEDs on the board.

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

2.4. Compiling the UHD HDMI 2.0 Video Format Conversion Design Example

Intel also provides a precompiled board programming file `precompiled.sof` as part of the project file in the `master_image` directory, so you can run the design without running a full compilation.

The steps show you how to compile the design, but the Intel Quartus project includes a Tcl script that automates steps 2 to 6, so you can choose to skip those step. Intel includes all the steps for compiling the design so you understand how the design is assembled.

1. In the Intel Quartus Prime software, open the project file `top.qpf`.
2. 2. Click **File ➤ Open** and select `ip/hdmi_subsys/hdmi_subsys.ip`.
   The paramerer parameter editor GUI for the HDMI IP opens, showing the parameters for the HDMI instance in the design.
3. Click **Generate Example Design** (not **Generate**).
4. When the generation completes, close the parameter editor.
5. Click **Tools ➤ Platform Designer** to open Platform Designer.
   a. Select `udx10_hdmi.qsys` for the Platform Designer system option and click **Open**
   b. Review the video processing pipeline.
   c. To generate the system, click **Generate HDL...**
   d. In the **Generation** window, turn on **Clear output directories for selected generation targets**.
   e. Click **Generate**.
6. In a terminal, navigate to `software/script` and run the shell `script build_sw.sh`.
   The software builds the Nios II software for the design, creating both the `vip_control.elf` file that you can download to the board at runtime, and a `.hex` file that compiles into the board programming `.sof` file.
7. Click **Processing ➤ Start Compilation**.
   The compilation creates the `top.sof` file in the `output_files` directory.

Related Information

Downloading and Installing the UHD HDMI 2.0 Video Format Conversion Design

2.5. Compiling the UHD HDMI 2.0 Video Format Conversion Design Example with the Nios II Software Tools for Eclipse

The design includes a shell script file (`/software/script/script_build_sw.sh`) to help you quickly build the Nios II software code for the design. The script allows you to quickly generate the programming files for the Nios II processor. However, it does not set up a workspace that allows interactive debugging of the software code.

You can follow the steps to compile the design software, which allows you to debug the design. Or you can run the Intel-provided script. To run the script:

1. In Windows Explorer, navigate to the `<install_dir>/software/script` directory with all of the necessary software files.
2. In a terminal from the `script` directory run the shell script `build_sw.sh`, which generates an executable vip_control.elf in the `vip_control` directory.

   Note: this scripts overwrites files in the `vip_control` directory. Edit any source files in the `vip_control_src` directory only.

STEPS:

1. In the installed project directory, create a new folder and name it `workspace`.
2. In the Intel Quartus Prime software, click **Tools ➤ Nios II Software Build Tools for Eclipse ➤ .**
   a. In the **Workspace Launcher** window, select the `workspace`.
   b. Click **OK**.
3. 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_hdmi/udx10_hdmi.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 and then click **Next**.
   d. Select **Create a new BSP project** based on the application project template with the project name `vip_control_bsp` and turn on **Use default location**.
   e. 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.
4. In Windows Explorer, copy the contents of the `software/vip_control_src` directory to the `software/vip_control` directory.
5. In 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**
6. Select **Project ➤ Build All** to generate the file `vip_control.elf` in the `software/vip_control` directory.
7. Build the `mem_init` file for the Intel Quartus Prime compilation:
   a. Right click on **vip_control** in the **Project Explorer** window.

b. Select **Make Targets ➤ Build...**

c. Select **mem_init_generate** and click **Build**.
The Intel Quartus Prime software generates the `udx10_hdmi_onchip_memory2_0_onchip_memory2_0.hex` file in the `software/vip_control/mem_init` directory

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

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

b. Select **Run As ➤ Nios II Hardware**.

If you have a Nios II terminal window already open, close it before trying to download new software.

Related Information

Downloading and Installing the UHD HDMI 2.0 Video Format Conversion Design Example

2.6. Running the UHD HDMI 2.0 Video Format Conversion Design Example on the Hardware

Download the compiled `.sof` for the design to the Intel Arria 10 GX FPGA Development Kit and run the design.

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 with the message do you want to update the programmer's device list, click **Yes**.
3. In the device list, select the row labelled **10AX115S2F45** and click **Change File...** then:

   a. To use the precompiled `.sof` included with the design, select the `.sof` in the `master_image` directory.

   b. To use your compiled `.sof`, select the `.sof` in the `output_files` directory.
4. Turn on **Program/Configure** in the **10AX115S2F45** row.
5. Click **Start**.
   When the programmer completes, the design runs automatically.
6. If you set user DIP switch 1 is to the ON position, open a Nios II terminal to receive the output text messages from the design, otherwise the design locks. If user DIP Switch 1 is set to off, do not open the Nios II terminal.

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

When the design is running, output appears on the display, even if no source is connected at the input. The output is a black screen with the Intel icon in the top right-hand corner of the screen. If you build the software using the Nios II Software Build Tools for Eclipse, you can edit, compile, and download the software programming file at any point after you program the board.

7. In the Nios II – Eclipse window, run the `vip_control.elf` programming file created by the Eclipse build .

   If a Nios II terminal window is already open, close it before trying to download new software.

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

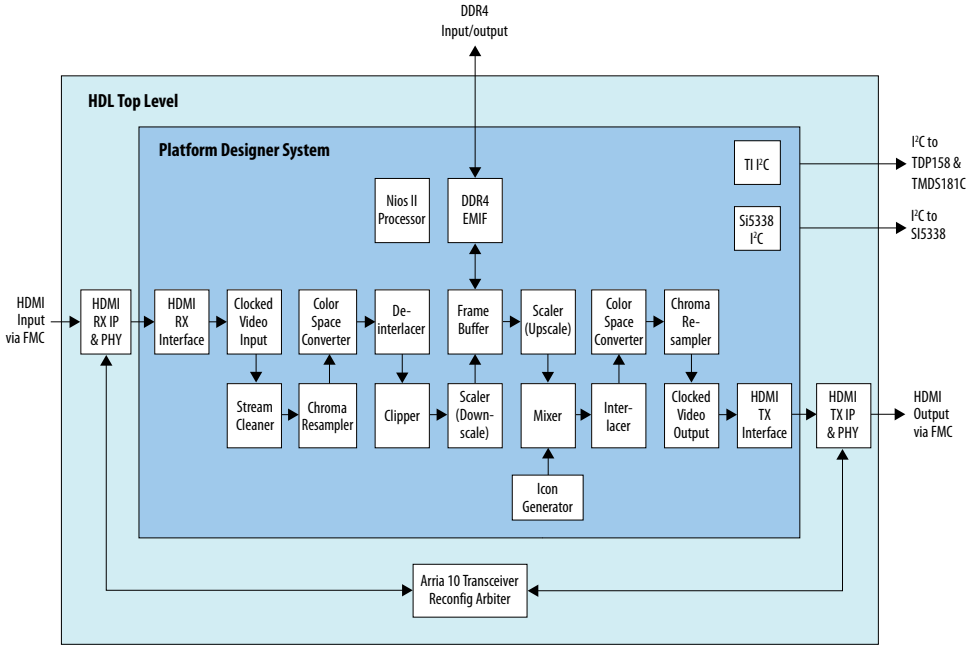
   b. Select **Run As ➤ Nios II Hardware**.

3. UHD HDMI 2.0 Video Format Conversion Design Example Functional Description

A Platform Designer system, `udx10_hdmi.qsys`, contains the video pipeline IP and the Nios II processor components. The top-level Verilog HDL file (top.v) connects the Platform Designer system to the HDMI RX and TX. The design comprises a single video processing path between the HDMI input and the HDMI output.

Figure 2. Block Diagram

The diagram shows the incoming video from the HDMI source on the left. The design processes the video through the video pipeline from left to right before passing the video out to the HDMI sink on the right.. The diagram does not show some of the generic peripherals connected to the Nios II processor or the Avalon memory-mapped interface between the Nios II processor and the other components of the system.



HDMI RX and PHY

The Bitec HDMI FMC card provides a buffer for the HDMI 2.0 signal from the HDMI source. The combination of HDMI RX PHY and HDMI RX IP decode the incoming signal to create a video stream. The HDMI RX PHY contains the transceivers to deserialize the incoming data and the HDMI RX IP decodes the HDMI protocol. The combined HDMI RX IP processes the incoming HDMI signal without any software intervention. The resulting video signal from the HDMI RX IP is a clocked video streaming format. The design configures the HDMI RX for 10-bit output.

HDMI RX Interface

The clocked video streaming data format output by the HDMI RX IP is compatible with the clocked video data format expected by the Clocked Video Input IP, which is next in the processing chain. However, the wire level interfaces have subtle differences that prevent a direct connection between the two blocks. The design-specific custom HDMI RX interface aligns the signals output by the HDMI and received by the Clocked Video Input IP.

The HDMI RX interface modifes the wire signaling standard and alters the order of the color planes within each pixel. This is required to translate between the HDMI standard color ordering and that used by the Intel video pipeline IP. The color swap is controlled by the HDMI RX AVI Infoframe data, which is an additional input to this block.

This component serves as a convenient register map based interface to access the HDMR RX AVI Infoframe data, program the RX EDID, and provide some of the transceiver reconfiguration settings. For more information on the register map, refer to *HDMI RX Interface Register Map*.

Clocked Video Input

The clocked video input processes the clocked video interface signal from the HDMI RX IP and converts it to Intel proprietary Avalon streaming Video format. This format strips all horizontal and vertical blanking information from the video, leaving only active picture data. The design packetizes the data as one packet per video frame and adds additional metadata packets (referred to as control packets) that describe the resolution of each video frame. For a full description of Avalon streaming video interface refer to the *Avalon Interface Specification*. The Avalon streaming video stream through the processing pipe is two 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 HDMI RX IP to the fixed clock rate (300 MHz) for the video IP pipeline.

Stream Cleaner

The stream cleaner ensures that the Avalon streaming video signal passing to the processing pipeline is error free. Hot-plugging of the HDMI source can cause the design to present incomplete frames of data to the clocked video input IP, which generates errors in the resulting Avalon-ST Video stream where the size of the packets containing the video data for each frame 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 received at the input via HDMI may be 4:4:4, 4:2:2 or 4:2:0 chroma sampled. The input chroma resampler takes the incoming video in whatever format it arrives and converts it to 4:4:4. To provide higher visual quality, the chroma resampler use the most computationally expensive filtered algorithm. The Nios II processor reads the current chroma sampling format from the HDMI RX via its Avalon memory-mapped agent interface, and communicates this data to the chroma resampler via its Avalon memory-mapped agent interface.

Color Space Converter (Input)

The video data received at the input via HDMI 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, for the Mixer IP later in the pipeline. The Nios II processor reads the current color space from the HDMI RX via its Avalon memory-mapped agent interface and loads the correct conversion coefficients to the color space converter via its Avalon memory-mapped agent interface.

Deinterlacer

The deinterlacer creates progressive video content from interlaced streams received at the input. It propagates progressive data unaltered. The deinterlacer can only run up to 150 MHz, so the design includes clock crossing and data width conversion (2->4 pixels per clock at the input, 4->2 pixels per clock at the output) components on either side of the deinterlacer. The deinterlacer is limited to the standard highest resolution for interlaced data of 1080i60.

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 HDMI source and the output resolution and scaling mode you select via the DIP switches and push buttons on the board. This design communicates this region to the Clipper via its Avalon memory-mapped agent interface.

Scaler

The design applies scaling to the incoming video data according the input resolution and the output resolution that you request. You may also select one of three scaling modes (upscale, downscale and passthrough) that affect how the video scales and displays. Two separate Scaler IPs provide the scaling functionality: one which implements any required downscaling, and another which upscales. The design requires two scalers for the following reasons.

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 each even numbered input line is received, 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 Frame Buffer must sit between any downscale and the Mixer, as going through 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 for the following Mixer. 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. The upstream Clipper produces data at an entirely different rate if it is applying a significant clip (e.g. during a zoom-in). For this reason, you must generally separate a Clipper and upscale 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 we must use two separate scalers either side of the Frame Buffer and use one for upscale and the other for downscale.

Using two Scalers reduces the maximum DDR4 bandwidth required by the Frame Buffer. Downscales are always applied before the Frame Buffer, minimizing the data rate on the write side. Upscales are applied after the Frame Buffer, which minimizes the data rate on the read side.

Each Scaler obtains the required input resolution from the control packets in the incoming video stream, while the output resolution for each Scaler is set by the Nios II processor via the Avalon memory-mapped agent interface. At least one of the scalers is configured for passthrough in each scaling mode. So if the design is upscaling video content then the downscaler pass video through unaltered, and if the design is downscaling the upscaler passes video through unaltered.

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 assuming the total pixel rate does not exceed 1 giga pixels per second. The output frame rate is set to either 30 or 60 fps by the Nios II software, according to the output mode you select. The output frame rate is actually a function of the Clocked Video Output settings and the output video pixel clock and is not set in the Frame Buffer. The backpressure applied by the Clocked Video Output to rest of the pipeline determines the rate at which the read side of the Frame Buffer pulls video frames from the DDR4 memory.

Mixer

The mixer generates a fixed size black background image that the Nios II processor 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. 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 and 4:2:0 and is set by the software. The output chroma resampler also uses filtered algorithm to achieve high-quality video.

Clocked Video Output

The clocked video output converts the Avalon streaming 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 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.

HDMI TX Interface

The HDMI TX interface accepts data formatted as clocked video. Subtle differences in the wire signaling and declaration of the conduit interfaces in Platform Designer prevent the design connecting the Clocked Video Output directly to the HDMI TX IP. The design-specific custom HDMI TX interface provides the simple conversion required between the Clocked Video Output and the HDMI TX IP. It also swaps the ordering of the color planes in each pixel to account for the different color formatting standards used by Avalon streaming video and HDMI, and provides a register map to access some of the transceiver reconfiguration and HDMI TX AVI Infoframe settings. For more information on the register map, refer to *HDMI TX Interface Register Map*.

HDMI TX IP and PHY

THE HDMI TX IP and PHY convert the video stream from clocked video to a compliant HDMI stream. The HDMI TX IP handles the HDMI protocol and encodes the valid HDMI data. The HDMI TX PHY contains the transceivers and creates the high-speed serial output.

Nios II Processor and Peripherals

The Platform Designer system contains a Nios II Processor that manages the HDMI RX and TX IPs and the runtime settings for the processing pipeline. The Nios II processor connects to serval other 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 HDMI 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 icon
- 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 HDMI source and sink fire interrupts that trigger the Nios II Processor to configure the HDMI TX and pipeline correctly. The main loop in the software code also monitors the values on the push-buttons and DIP switches and alters the pipeline setup accordingly.

I²C controllers

- The design contains two I²C controllers to edit the settings of four other components on the Arria 10 GX FPGA Development Kit and Bitec HDMI 2.0 daughter card:
- — Si5338 I²C. The Arria 10 GX FPGA Development Kit includes two Si5338 clock generators on which both 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 for 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 150 MHz. The second Si5338 clock generator generates the `vid_clk` for the clocked video interface between the pipeline and the HDMI TX IP. The Nios II processor adjusts the speed of this clock at runtime for each different output resolution and frame rate supported by the design.
  - — TI I²C .The Bitec HDMI 2.0 FMC daughter card uses the TI TDP158 HDMI 2.0 redriver and TI TMDS181C retimer. At startup the Nios II processor edits the default settings of these component to meet the requirements of the design.

Related Information

- Altera High-Definition Multimedia Interface (HDMI) IP Core User Guide
- Video and Image Processing Suite User Guide
  Information about Avalon-ST video interface

3.1. Software Description

All the IPs in the UHD HDMI 2.0 Video Format Converison Design Example can process frames of data without any further intervention once they are setup correctly. However, the design requires external high-level control to setup the IPs to begin with and when any changes occur in the system, e.g. HDMI RX or TX hot-plug events, or user push button activity. In the 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 EMIF to recalibrate on the new reference clock.
- Sets up the TI TDP158 HDMI 2.0 redriver and TI TMDS181C retimer
- Initializes the HDMI RX and TX interfaces
- Initializes the processing pipeline IPs

After initialization, the software enters a continuous `while` loop, checking for and reacting to the following events.

Changes to the Scaling Mode

The design supports three basic scaling modes; passthrough, upscale, and downscale. In passthrough mode the input video is not scaled; in upscale mode the input video is upscaled, and in downscale mode the input video is downscaled. Four blocks determine the presentation of the final output in each mode in the processing pipeline: the clipper, the downscaler, the upscaler, and the mixer. 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, the design cannot apply an upscale to the input video without first clipping it. And if the input resolution is less than the output, the design cannot apply a downscale without using a mixer background layer that is larger than the input video layer, which adds black bars around the output video.

Table 5. Pipeline actions in each scaling mode

The 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	Input size > output size	Input size = output size	Input size < output size
Passthrough	• Clip to output size • No downscale • No upscale • No black border	• No clip • No downscale • No upscale • No black border	• No clip • No downscale • 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

You switch 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 HDMI Input

On each run through the loop the software polls the status of the clocked video input (CVI), looking for changes in the stability of the input video stream. The design considers the video stable if the CVI reports that the clocked video is successfully locked, and if the input resolution and color space has not changed since the previous run through the loop.

If the design previously considers the input stable, but it loses the lock or the properties of the video stream change, the software stops the CVI sending video through the pipeline, and sets the mixer to stop displaying the input video layer. Then the output remains active (showing a black screen and the Intel icon) during any RX hot-plug events or resolution changes.

If the input was not previously stable but is now stable, the design configures the pipeline to best display the new input resolution and color space, restarts the output from the CVI, and sets the mixer to display the input video layer once 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, which the design must clear before you can re-enable the display to avoid glitching. The frame buffer keeps a count of the number of frames the design reads from the DDR4 memory, and the Nios II processor reads this count. The software samples this count when the input becomes stable, and re-enables the Mixer layer when the count increases by four frames. More than sufficient to ensure the design flushes any old frames from the buffer.

HDMI TX Hot-plug Events

The software polls the HDMI TX IP on each run through the loop to check for hot-plug events. When the design detects a TX hot plug, the design reads the EDID for the new display to determine which resolutions and color spaces it 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, HDMI TX IP and the Si5338 part that generates the TX `vid_clk` for the new output mode. The design does not display the mixer layer for the input video while it edits the settings for the pipeline. The design does not re-enable the display until four frames with the new settings pass through the frame buffer.

Changes to User DIP Switch Settings

The positions of DIP switches 2 to 6 control the output format (resolution, frame rate, color space and bits per color) that drives through the HDMI TX. When the design detects any changes on these DIP switches, the software runs through a sequence that is similar to a TX hot-plug. The only difference in this case is that the TX EDID does not need to be queried as this has not changed.

4. Considering Design Security

Intel provide this design as a showcase for the Intel FPGA IP and do not intend it for use in production or deployed systems. Several features of the design may not meet customer security requirements. You should conduct a security review of your final design to ensure it meets your security goals.

Not all precautions apply to all designs or IP.

1. Remove the JTAG interface from your designs.
2. To guarantee video data integrity, restrict access to memory allocated to the frame buffer.
3. Control access to areas of memory to prevent unauthorised transactions or corruption by other IP in the design.
4. Ensure that you correctly configure the IP via the I²C interface and that the input video is valid.
5. Protect the bitstreams for your design using the security features built-in to Intel Quartus Prime.
6. Enable a password for the design's ARM processor.
7. Protect access to your design through development kit ports.
8. Restrict debugging access by tools such as Signal Tap.
9. Encrypt information on SD cards, FPGA bitstreams, and within DDR memory devices.
10. Apply security features to the video data is storage.
11. Consider using an HDCP encryption scheme.
12. Consider the boot sequence and boot security aspects of your own design.
13. Implement Intel's FPGA bitstream encryption technology to further protect the FPGA design content of your products. For information on FPGA bitstream encryption technology, refer to *Using the Design Security Features in Intel FPGA*s.

5. UHD HDMI 2.0 Video Format Conversion Design Example Document Revision History

Date	Version	Changes
April 2021	2021.04.15	• Renamed to UHD Video Format Conversion Design Example • Updated *Downloading and Installing*
January 2018	2018.01.11	• Updated for Intel Quartus Prime v17.1 • Added support for YCbCr video • Deleted `.sdc` file • Removed `refclk_sdi_p clock`; added `refclk_fmcb_p` • Removed duplicate stream cleaner parameters table. • Added default design settings via DIP switch. • Added support for deinterlacer.
August 2016	2016.08.01	Initial release.

A. HDMI RX Interface Register Map

The HDMI RX interface component presents two Avalon memory-mapped agent interfaces for connection to the Nios II processor.

The `edid_slave` interface provides a mechanism to connect to the EDID Avalon memory-mapped agent interface on the HDMI protocol IP, which sits outside the Platform Designer system. The register map for this interface is in the *HDMI IP User Guide*.
The `info_slave` interface is primarily allows the Nios II to access the HDMI RX AVI Infoframe data from the HDMI RX IP, but which also provides access to some signals associated to configuring the transceivers that otherwise needs to be accessed via PIOs.

Table 6. HDMI RX Register Map

Address (byte)	Address (Word)	Permission	Name	Description
0	0	Read only	HDMI RX GCP	HDMI General Control Packet currently output by the HDMI RX IP
1 – 13	4 – 52	Read only	HDMI RX AVI Infoframe	HDMI AVI Infoframe currently output by the HDMI RX IP. The AVI Infoframe is output by the HDMI RX as a 112 bit signal. Bits[7:0] are the checksum and are not exposed through the register map. Registers 1 through 13 each provide access to one byte of the remaining 104 bits of this interface, with bits[15:8] in register 1 and bits [103:96] in register 13
14	56	Read only	TMDS Bit clock ratio	Bit[0] of this register provides the current value of TMDS Bit clock ratio output by the HDMI RX IP. This value indicates if TMDS Bit Rate is greater than 3.4 Gbps.
15	60	Read only	Unused	Unused
16	64	Read only	PMA Busy	Bit[0] is 1 if the transceiver reconfig is busy
17	68	Writeable	RX reset transceiver	The value in bit[0] is driven onto the transceiver reset for the HDMI RX
18	72	Writeable	RX transceiver reconfig enable	Writing 1 to bit[0] of this register enables reconfiguration of the RX transceiver settings
19	76	Writeable	RX transceiver reconfig channel	Sets which RX transceiver channel new settings should be applied to

Related Information

Intel HDMI Intel FPGA IP User Guide

B. HDMI TX Interface Register Map

The HDMI TX interface component presents two Avalon memory-mapped agent interfaces for connection to the Nios II processor.

The `i2c_slave` interface provides a mechanism to connect to the i2c Avalon memory-mapped agent interface on the HDMI protocol IP, which sits outside the Platform Designer system. The register map for this interface is in the *HDMI IP User Guide*
The `info_slave` interface primarily allows the Nios II to write the HDMI TX AVI Infoframe data from the HDMI TX IP. It also provides access to some signals associated to configuring the transceivers and PLLs that otherwise you need to access via PIOs.

Table 7. HDMI TX Register Map

Address (byte)	Address (Word)	Permission	Name	Description
0	0	Writeable	HDMI TX GCP	HDMI General Control Packet for the HDMI TX IP
1 – 13	4 – 52	Writeable	HDMI TX AVI Infoframe	HDMI AVI Infoframe for the HDMI TX IP. The AVI Infoframe is input to the HDMI TX as a 112 bit signal. Bits[7:0] are the checksum and are automatically generated inside this component so are not exposed through the register map. Registers 1 through 13 each provide access to one byte of the remaining 104 bits of this interface, with bits[15:8] in register 1 and bits [103:96] in register 13
14	56	Writeable	HDMI 2 Mode	Bit[0] of this register indicates to the HDMI TX IP to transmit using HDMI 2.0 data rates
15	60	Writeable	Unused	Unused
16	64	Read only	Status	• Bit[0] indicates if a TX hot-plug has occurred • Bit[1] indicates if the transceiver calibration is busy. • Bit[2] indicates if the transceiver reconfig is busy • Bit[3] indicates if the PLL reconfig is busy • Bit[4] indicates if the IOPLL reconfig is busy
17	68	Writeable	TX Hot-plug acknowledge	Bit[0] of this register drives the TX hot-plug acknowledge signal
18	72	Writeable	TX transceiver reset	The value in bit[0] is driven onto the transceiver reset for the HDMI TX
19	76	Writeable	TX PLL reset	The value in bit[0] is driven onto the PLL reset for the HDMI TX
20	80	Writeable	TX transceiver reconfig enable	Writing 1 to bit[0] of this register enables reconfiguration of the TX transceiver settings
21	84	Writeable	TX transceiver reconfig channel	Sets which TX transceiver channel new settings should be applied to

Related Information

HDMI Intel FPGA IP User Guide