High-Definition Video Reference Design (UDX6)

2013.03.01
an679 Subscribe Feedback

Introduction

The Altera® high-definition video reference designs deliver high-quality up-, down-, cross-conversion (UDX) designs for standard-definition, high-definition, and 3 gigabits per second (Gbps) video streams in interlaced or progressive format. These reference designs are highly software and hardware configurable, enabling rapid system configuration and design. The designs target typical broadcast applications such as switcher, multiviewer, converter, and video conferencing products.

The UDX6 reference design improves on the UDX5 reference design by including the Noise Reduction intellectual property (IP) core.

The UDX6 reference design uses the debugging extension of the video and image processing framework. The design contains multiple instances of the Avalon-ST Video monitor to debug the Avalon® Streaming (Avalon-ST) Video connections between video and image processing IP cores from a PC hosting the debugging session.

Related Links

BroadcastDesign (V1)

High Definition Video Reference Design (V1)

High Definition Video Reference Design (V2)Reference Design (UDX4)

High-Definition Video Reference Design (UDX4)

High-Definition Video Reference Design (UDX5)

Features

- Files for targeting the Arria V GX Starter Kit
- A unified memory architecture to allow CPU, on-screen display (OSD), and video processing to use one 32-bit DDR3 SDRAM
- A switch block to allow run-time reconfiguration of the two video processing paths
- Two input channels, both connected to one triple-rate serial digital interface (SDI) input: standard definition, high definition or 3 Gbps (3G), interlaced or progressive, resolution and frame rate up to 1080i60
- Active output, connected to one triple rate SDI output: SD, HD or 3G, interlaced or progressive, resolution and frame rate up to 1080i60
- Two high-quality UDX video-processing paths:

- Low-latency motion adaptive deinterlacer with improved Sobel-based edge detection and interpolation and cadence detection
- Scaler (8 × 8 taps)
- Interlacer
- One frame of latency (frame buffer) when inputs and outputs are locked, less than two frames of latency in other cases
- Run-time controllable frame rate conversion
- Genlock
- Alpha-blending mixer to allow multiview and OSD on one video processing datapath—the design retrieves the OSD from a buffer in memory
- System initialization and run-time configuration of input and output formats and resolutions in software. Software loads from flash memory together with SRAM Object File (**. sof**) and OSD logos.
- Active format description (AFD) extraction and insertion with dynamic clipping, scaling, and padding to support 4:3 to 16:9 and 16:9 to 4:3 format conversion based on an AFD code.
- Run-time debugging:
- The Avalon-ST Video monitor components allow you to debug Avalon-ST Video protocol connections and routes protocol information to a debugging host in real time
- System Console running on the host controls hardware debugging agents on the board, providing system information while running.

For a full description of the Avalon-ST Video protocol and Avalon interfaces, refer to the "Interfaces" chapter in the *Video and Image Processing Suite User Guide*Interface Specifications. For more information about the Avalon-MM and Avalon-ST interfaces, refer to the *Avalon* Interface Specifications.

Related Links

Video and Image Processing Suite User GuideInterface Specifications

Avalon Interface Specifications

System Requirements

The hardware and software requirements to run the UDX6 reference design.

Hardware Requirements

The UDX6 reference design requires the following hardware components:

- Arria V GX Starter Kit
- For more information, refer to the *Arria V GX Starter Kit User Guide*.
- Combination of the following video sources:
- A single SDI video source providing progressive or interlaced output up to 1080p60 resolution and frame rate. This source connects to the onboard SDI RX SMB connector.
- Combination of the following video displays:
- A single SDI monitors supporting 1920 × 1080 pixel resolution. This monitor connects to the onboard SDI TX SMB connector.

or

- A 3G SDI-to-Display Port converter box and a monitor with Display Port input supporting 1920 × 1080 pixel resolution.

Related Links

ArriaV GX Starter Kit User Guide.

Software Requirements

The UDX6 reference design requires the Altera Complete Design Suite v12.0, which includes the Quartus II software, Qsys, Nios® II EDS, and MegaCore IP Library (including the Video and Image Processing Suite).

Resource Utilization

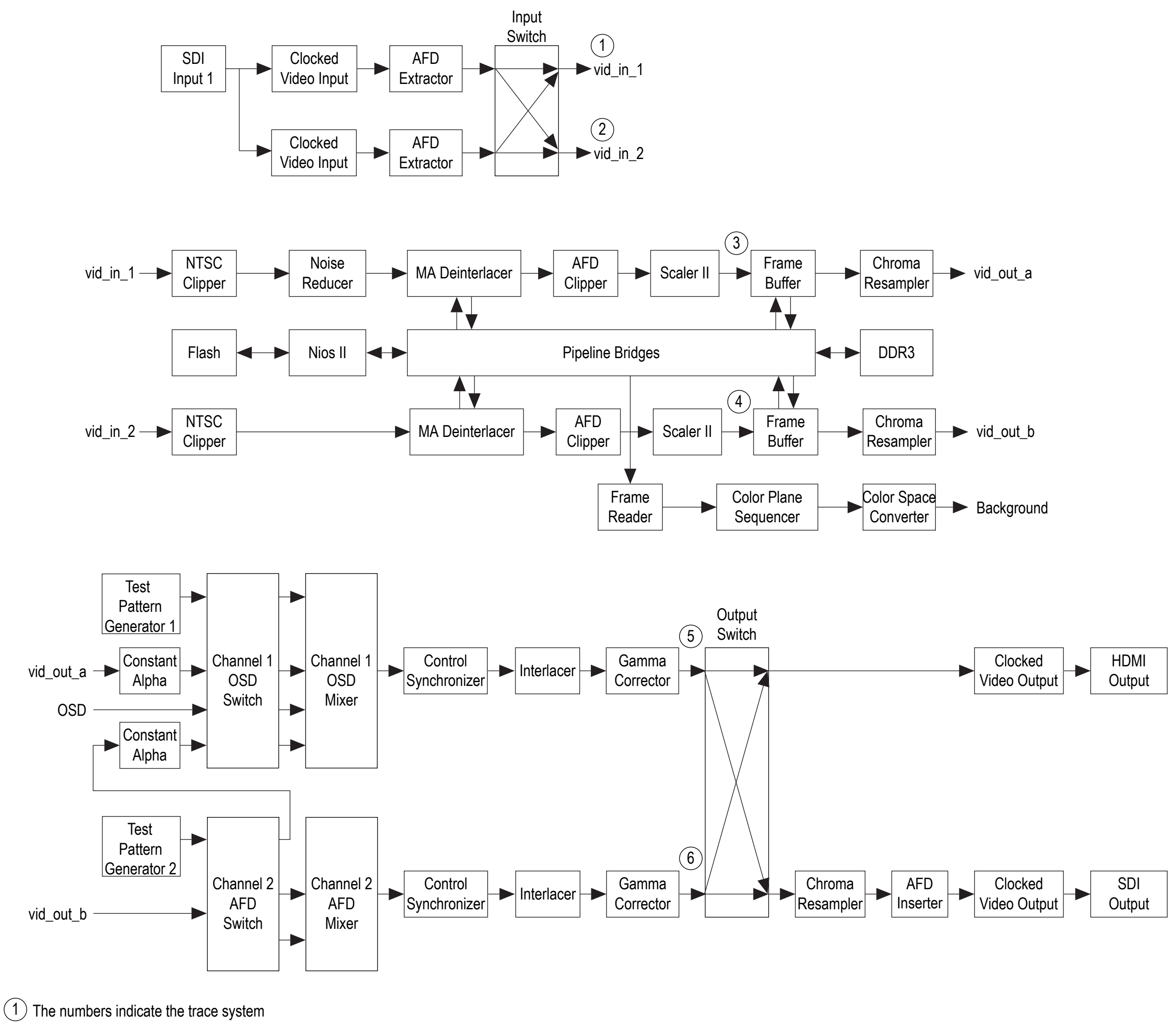
Table 1: Design Combinational ALUTs ALMs Dedicated Logic Registers M10Ks 18-bit DSPs

Design	Combinational ALUTs	ALMs	Dedicated Logic Registers	M10Ks	18-bit DSPs
UDX6	84,380	63,000	103,850	657	196
UDX6 without debugging	80,340	58,420	96,220	566	196

Functional Description

Figure 1: Block Diagram

UDX6 Reference Design Block Diagram



The UDX6 reference design comprises two video processing paths

Design Blocks

Most of the blocks in the UDX6 reference design are instances of Video and Image Processing Suite IP cores. Altera parameterizes all of the IP cores with a maximum resolution of 1920 × 1080 pixels.

Table 2: Reference Design Blocks

Reference Design Block	MegaCore Function	Description
AFD clipper (×2)	Clipper	Clips input images based on the detected AFD code and removes the letterbox or pillarbox.
AFD extractor (×2)	AFD Extractor	Extracts AFD code (0 to 15) from Avalon-ST Video ancillary packets. Supports access to the AFD code through an Avalon-MM slave control interface.
AFD inserter	AFD Inserter	Inserts the AFD code (0 to 15) the AFD Inserter receives on its Avalon-MM slave interface to an Avalon-ST Video ancillary packet (type `0xD`).
AFD mixer	Alpha Blending Mixer	Second stage of the AFD correction. Adds letterbox bars or pillarbox bars to the second video input channel.
AFD switch	Switch	In multi-view mode, sends the second video input channel to the OSD mixer; in independent mode, sends the second video input channel and the test pattern to the AFD mixer.
Chroma resampler (×2)	Chroma Resampler	Provides YCbCr 4:2:2 to YCbCr 4:4:4 upsampling.
Color plane sequencer	Color Plane Sequencer	Splits the red, green, blue, alpha (RGBA) input to two output streams, one RGB and one A, on separate Avalon-ST ports.
Color space converter	Color Space Converter	Run-time configurable to convert the 10-bit RGB or YUV input format, depending on the color spaces of the input and output videos.
Constant alpha source (×2)	Chroma Key (beta)	Outputs a constant alpha value synchronized to the video stream. Run-time configurable to alter the alpha value on a frame-by-frame basis.

Reference Design Block	MegaCore Function	Description
Control synchronizer (×2)	Control Synchronizer	Ensures that switching of the relevant mixer switch is synchronized to the start of an image packet in the video stream, so that the switch and corresponding mixer are updated on the same frame, preventing deadlock when changing mode.
DDR3 SDRAM controller	DDR3 SDRAM Controller with UniPHY	Provides DDR3 SDRAM access with 32-bit wide, 333-MHz operations to 512 MB of memory split into eight banks. Runs in half-rate mode with a 128-bit 166-MHz Avalon-MM interface.
DisplayPort Output	-	-
Frame buffer (×2)	Frame Buffer	Provides run-time configurable double buffering, or triple buffering to support frame-rate conversion (dropping and repeating of frames).
Frame reader	Frame Reader	Provides Avalon-MM to Avalon-ST Video bridge. Run-time configurable memory read address. Performs frame repeating to convert between slow frames per second rate of the OSD updates and faster frames per second rate of the video output.
Gamma corrector (×2)	Gamma Corrector	Converts 10-bit color values to 8-bit color values for Display Port output. Conversion is run-time configurable.
Input switch	Switch	Allows you to select two of the three input streams for input to the two video processing channels.
Interlacer (×2)	Interlacer	Optionally changes progressive video stream to interlaced output. Configured for interlaced output (drops half the lines of each input frame), and control port to allow run-time configuration of pass-through mode for progressive video.

Reference Design Block	MegaCore Function	Description
MA deinterlacer (×2)	Deinterlacer II	Deinterlaces video input using motion-adaptive (MA) algorithm with the following parameters: • Sobel-derived low-angle edge interpolator • Motion bleed turned on • 4:2:2, two channels in parallel processing • Low-latency mode • Rate doubling—output frame rate equals input field rate • 3:2 and 2:2 cadence detection
Nios II processor	Nios II Processor	Embedded Nios II/f processor with a Level 1 JTAG debug module and an internal interrupt controller.
Noise reducer	Noise Reducer	-
NTSC clipper	-	-
OSD color space converter	Color Space Converter	Converts the 8-bit RGB format of the OSD to 10-bit RGB or YCbCr, depending on the format of the output video.
OSD mixer	Alpha Blending Mixer	In multi-view mode, merges the two video input channels with a background test pattern and sends them to the first video output. In independent mode, applies AFD correction to the first input channel.
OSD switch	Switch	Connects four Avalon-ST Video inputs to four Avalon-ST Video outputs. In multi-view mode, sends each of the two video input streams, test pattern, and frame reader output to its own arbitrary layer of the OSD mixer; in independent mode, sends the video input and the test pattern to the OSD mixer.
Output chroma resampler	Chroma Resampler	Performs 4:4:4 to 4:2:2 conversion. This conversion is not run-time configurable.
Output switch	Switch	Connects two Avalon-ST Video inputs to four Avalon-ST Video outputs. Allows two output streams to be selected for output from the video processing channels.

Reference Design Block	MegaCore Function	Description
Pipeline bridge (x5)	Pipeline Bridge	Provides pipelining for multiple masters to the single external memory.
Scaler (×2)	Scaler II	Scales images for 4:2:2 data in parallel in edge-dependent polyphase mode with 12 × 12 taps, run-time configurable output resolution and run-time configurable filter coefficients.
SDI input and output	SDI	SDI MegaCore function (not in the Video and Image Processing IP Suite) configured for triple standard SDI bidirectional communication. A single SDI MegaCore function treats both input and output SDI communication for one video processing channel of the reference design. The Arria V GX Starter Kit contains a single SDI RX connector, which is connected to both input channels. The Arria V GX Starter Kit contains a single TX connector, so the second video output channel is discarded inside the FPGA.
SDI clocked video input	Clocked Video Input	Inputs 10-bit SD SDI and 20-bit HD SDI with embedded synchronization information. Extracts embedded ancillary data packets from vertical blanking region and inserts them in Avalon-ST Video ancillary packets (type `0xD`).
SDI clocked video output	Clocked Video Output	Outputs 10-bit SD SDI and 20-bit HD SDI with embedded synchronization information.
Test pattern generator (×2)	Test Pattern Generator	Outputs progressive RGB 4:4:4 test pattern of width and height up to 1920 × 1080. Width and height are run-time configurable.
TPG color space converter (×2)	Color Space Converter	Passes through the 10-bit RGB test pattern or converts the test pattern to 10-bit YCbCr format, depending on the format of the output video.
Trace system	Trace System	Provides an interface between individual video monitors and the JTAG cable. This module provides a global timestamp and shared buffering of debug information.

Reference Design Block	MegaCore Function	Description
Video trace modules (x6)	Avalon-ST Video Monitor	Six video trace monitors allow debugging of the design.

All of the MegaCore functions, except for the SDI MegaCore function, are in the Video and Image Processing Suite.

Related Links

Video and Image Processing Suite User Guide

Noise Reducer

The Noise Reducer IP core uses a motion adaptive temporal filtering (MATF) algorithm to attenuate unwanted grain noise in the input video. It uses both spatial (intra-frame) and temporal (inter-frame) low -pass filtering to average pixel values and reduce noise (typically zero-mean white noise).

The Noise Reducer IP core uses a motion adaptive temporal filtering (MATF) algorithm to attenuate unwanted grain noise in the input video. It uses both spatial (intra-frame) and temporal (inter-frame) low -pass filtering to average pixel values and reduce noise (typically zero-mean white noise).

The noise reducer IP core uses both spatial dimensions and the third temporal dimension and is a 3D noise reducer. For each pixel the algorithm compares the surrounding 3x3 window of pixels with the corresponding window of pixels from a reference (the reference frame is essentially the previous output frame). It calculates the sum of absolute differences (SAD) between the two pixel windows and, if the SAD is less than a given motion threshold value, the image is considered to be static at that pixel. In regions where the input video is static and there is no motion, the IP core averages the pixel values across multiple input frames to create the output pixel, which cancels out the zero-mean noise. The IP core uses an Impulse Response (IIR) filter to calculate a weighted sum of the input pixel with the corresponding pixel from the reference frame. Because the reference pixel is the output pixel from the previous frame, which is the sum of an input pixel and reference pixel, you can create a moving average across all previous input pixels. The IP core stores a reference frame and it also stores a value for each pixel that indicates for how many successive frames there is no motion at that pixel. The IP core uses this value to adjust the weightings in the IIR filter - the longer the pixel has been static the higher the weighing towards the reference pixel.

In areas where motion is detected (the SAD value exceeds the current motion threshold) the temporal IIR filter does not work, so the algorithm uses spatial filtering. It averages pixels in the 5x5 pixel window around the current input pixel to produce the output pixel using a Finite Impulse Response (FIR) filter. To avoid blurring edges, the IP core also includes an edge detector. In the areas where edges are detected, the FIR filter is not applied. So each pixel in the output frame is either the output from the inter-frame IIR filter, the output from the intra-frame FIR filter, or simply the original input pixel. The reference frame for the next input frame is then a copy of the output frame that the IP core writes to a frame buffer (typically off-chip DDR3).

SDI MegaCore Function

The design uses a SDI MegaCore function as a triple-rate receiver SDI. The reference design requires two instances: one for each video processing channel. The SDI input clock frequency is 148.5/148.35 MHz for 3 Gbps (3G-SDI) or 74.25/74.175 MHz for 1.5 Gbps (HD-SDI) video inputs, or 27.0 MHz for SD-SDI video inputs. This design uses a clock frequency of 148.5 MHz, which allows use of SD-SDI, HD-SDI and 3G-SDI input clocks.

Related Links

SDI MegaCoreFunction User Guide

Noise Reducer

The Noise Reducer IP core uses a motion adaptive temporal filtering (MATF) algorithm to attenuate unwanted grain noise in the input video. It uses both spatial (intra-frame) and temporal (inter-frame) low -pass filtering to average pixel values and reduce noise (typically zero-mean white noise).

The Noise Reducer IP core uses a motion adaptive temporal filtering (MATF) algorithm to attenuate unwanted grain noise in the input video. It uses both spatial (intra-frame) and temporal (inter-frame) low -pass filtering to average pixel values and reduce noise (typically zero-mean white noise).

The noise reducer IP core uses both spatial dimensions and the third temporal dimension and is a 3D noise reducer. For each pixel the algorithm compares the surrounding 3x3 window of pixels with the corresponding window of pixels from a reference (the reference frame is essentially the previous output frame). It calculates the sum of absolute differences (SAD) between the two pixel windows and, if the SAD is less than a given motion threshold value, the image is considered to be static at that pixel. In regions where the input video is static and there is no motion, the IP core averages the pixel values across multiple input frames to create the output pixel, which cancels out the zero-mean noise. The IP core uses an Impulse Response (IIR) filter to calculate a weighted sum of the input pixel with the corresponding pixel from the reference frame. Because the reference pixel is the output pixel from the previous frame, which is the sum of an input pixel and reference pixel, you can create a moving average across all previous input pixels. The IP core stores a reference frame and it also stores a value for each pixel that indicates for how many successive frames there is no motion at that pixel. The IP core uses this value to adjust the weightings in the IIR filter - the longer the pixel has been static the higher the weighing towards the reference pixel.

In areas where motion is detected (the SAD value exceeds the current motion threshold) the temporal IIR filter does not work, so the algorithm uses spatial filtering. It averages pixels in the 5x5 pixel window around the current input pixel to produce the output pixel using a Finite Impulse Response (FIR) filter. To avoid blurring edges, the IP core also includes an edge detector. In the areas where edges are detected, the FIR filter is not applied. So each pixel in the output frame is either the output from the inter-frame IIR filter, the output from the intra-frame FIR filter, or simply the original input pixel. The reference frame for the next input frame is then a copy of the output frame that the IP core writes to a frame buffer (typically off-chip DDR3).

System Peripherals

The system contains the following memory-mapped peripheral and parallel I/O components to provide information about the status of the design and to support your run-time input:

- A JTAG UART to display software `printf` output.
- An LCD display of reference design start-up information including the standard of the input video streams.
- LEDs to display system status.
- Push-button switches to allow switching between video output resolutions.

Run-time Configuration

The design determines input resolution on each video processing channel from the input video stream. You can use the push-button switches on the Arria V GX Starter Kit to control the reference design video processing function and the output video resolution.

The Clocked Video Input MegaCore function retains a count of the height, width, and format (progressive or interlaced) of the input video, and detects changes to the resolution or format. After the design detects stable video, the Clocked Video Input MegaCore function generates control packets containing the new resolution and progressive or interlaced format. The control packets propagate through each function in the video processing path, parameterizing each IP core to receive the new video data.

The Clocked Video Input MegaCore function also generates an interrupt when a resolution change occurs, with the interrupt line connected to the Nios II processor. When an interrupt is generated, the Nios II interrupt service routine performs various functions, including the following functions:

- Calculates new coefficients based on the new scaling ratio
- Writes new clipping area parameters
- Writes new scaling coefficients to the scaler control interface

You can use the general purpose push-button switches on the Arria V GX Starter Kit to change the video processing channel connectivity and the resolution on the output displays.

Control Software

For control purposes, each video processing path in the reference design is split into two parts, a video input channel or OSD input channel, and a video output channel.

Video Input Channel

A video input channel starts at a clocked video input blocks. The reference design has two almost identical video input channels, `input_channel_1` and `input_channel_2`. The `input_channel_1` includes a Noise Reduction IP core.

The configuration of the input switch block determines the clocked video input for this video input channel. The video input channel converts the incoming video stream to a progressive format in the correct color space, resolution, and frame rate for the video output channel to which it is connected. The design performs most of the video processing on YCbCr 4:2:2 subsampled data. The design uses the Chroma Resampler MegaCore functions, to upsample data to 4:4:4 just before the end of the input channel.

OSD Input Channel

The OSD input channel starts at the frame reader block and extends to the OSD color space converter block.

The frame reader reads an 8-bit RGBA frame from memory. In this format, each pixel is represented by 32 bits. The frame reader streams the data out on its Avalon-ST Video output. The color plane sequencer splits the 32-bit pixels into an 8-bit RGB stream and an 8-bit alpha stream. The color space converter converts the 8-bit RGB data to 10-bit RGB data or YCbCr data. The 8-bit alpha stream feeds through a FIFO buffer, which absorbs backpressure differences between the split streams, to a gamma corrector that inverts the alpha value, making the alpha-value 0 fully opaque and the alpha-value 255 fully transparent. The reference design has a single OSD input channel.

Video Output Channel

A video output channel starts at a test-pattern generator and extends to a clocked video output block, passing through a mixer and its switch.

The configuration of the output switch block determines the clocked video output for this video output channel. The video output channel mixes the video streams (or AFD padding for a single video stream) and predetermined OSD logos, when appropriate, before performing gamma correction or dithering and optionally creating interlaced output. The inputs to a video output channel can be one or more video input channels or OSD input channel. The reference design has the following two video output channels:

- The `osd_output_channel`, which requires input from two video input channels and one OSD input channel
- The `afd_output_channel`, which requires input from only the second video input channel (`input_channel_2`).

Modes

The design has the following run-time configurable modes:

- Independent mode 1, in which the two video processing paths operate independently. In this default mode, `input_channel_1` connects to the `osd_output_channel` and `input_channel_2` connects to the `afd_output_channel`. The two independent video processing paths convert video data from input format to output format.
- Independent mode 2, which connects `input_channel_2` to the `osd_output_channel` . In this mode you can view the channel full-screen without noise reduction when switching between modes.
- A multi-view mode, in which two input channels are merged to a single output channel. In the multi-view mode, the two input data channels are alpha-blended with an OSD logo.

The software synchronizes operations carefully before switching between independent mode and multi-view mode. Without careful synchronization, deadlock can occur if a mixer expects video data that is not forthcoming in the new mode.

Before switching from independent mode 1 to independent mode 2, the software sets the AFD Mixer Control synchronizer to perform the following steps on the next start-of-image packet

1. Uses the AFD mixer switch to route `input_channel_2` to the OSD mixer switch instead of the AFD Mixer.
2. Uses the AFD mixer switch to disconnect `input_channel_1` from the OSD mixer switch.
3. Disables the AFD mixer layer 1.

Before switching from independent mode 2 to multi-view mode the software set the AFD mixer control synchronizer to perform the following steps on the next start of image packet:

1. Uses the AFD mixer switch to connect `input_channel_1` to the OSD mixer switch.
2. Disables the AFD mixer layer 1.

Before switching from independent mode 1 to multi-view mode, the software sets the AFD mixer control synchronizer to perform these steps on the next start-of-image packet:

1. Uses the AFD mixer switch to route `input_channel_2` to the OSD mixer switch instead of the AFD Mixer.
2. Disables the AFD mixer layer 1.

After these changes occur, OSD mixer layer 2 is enabled.

Before switching from multi-view mode to independent mode, the software sets the OSD mixer control synchronizer to perform these steps on the next start-of-image packet:

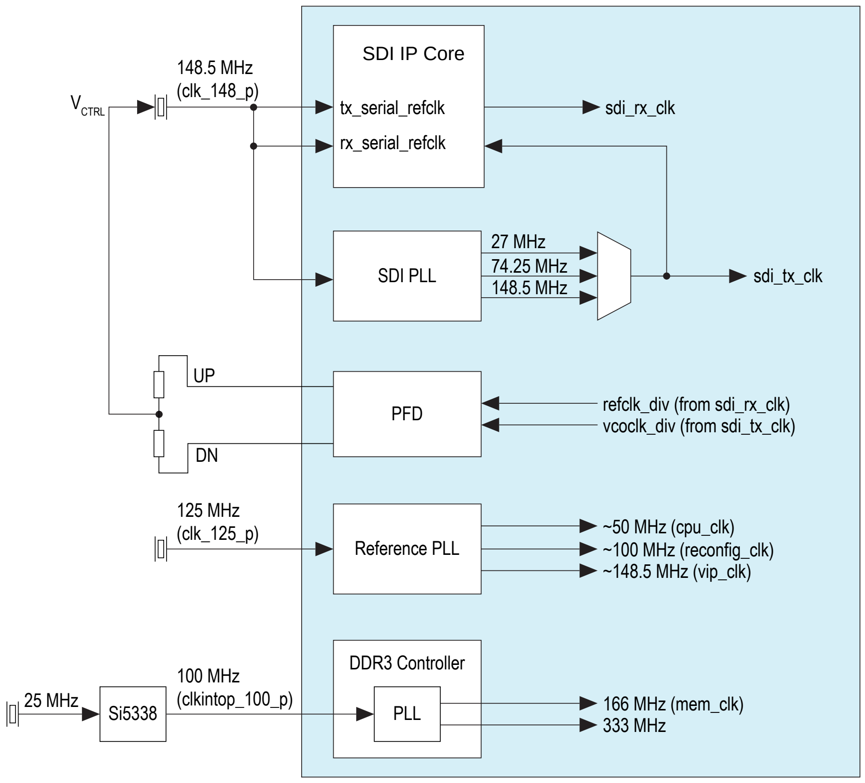
1. Uses the AFD mixer switch to route `input_channel_2` to the AFD mixer instead of the OSD mixer switch.
2. Disables the OSD mixer layer 2.

After these changes occur, AFD mixer layer 1 is enabled.

Clocking

The system clock `vip_clk` clocks the video processing datapath at 148.5 MHz, to allow the processing functions to process progressive frames of video with resolutions up to 1920 × 1080 pixels at a rate of 60 frames per second. The `cpu_clk` clocks the Nios II processor at approximately 50 MHz. The memory subsystem, including the pipeline bridges, runs at 166 MHz; the DDR3 SDRAM runs at 333 MHz. The deinterlacer and the frame buffer include clock-crossing FIFO buffers to support connection to a memory subsystem running at a different clock frequency. The clocked video input and clocked video output blocks include clock-crossing FIFO buffers to support communication between the video processing datapath and the incoming and outgoing video data. The external video frequency conforms to the relevant video protocol.

Figure 2: Top-Level Clock Connections for SDI Data and Control



The SDI voltage controlled crystal oscillator (VCXO) to clock video PLL (Si571) on the SDI HSMC board provides a 148.5 MHz clock that clocks the video outputs. The `pll_config` block on the FPGA sets the `SDI_XTAL_SEL` input to the PLL, which determines the choice of clock frequency.

The UDX6 reference design video interfaces use clocks rates that are derivatives of 148.5 MHz (as opposed to 148.35 MHz). The output formats are 720p60 or 1080i60 rather than 720p59 or 1080i59. The reference design `u_sdi_pll` block then divides the `clk_148_p` signal and outputs clock frequencies.

Table 3: Available Output Frequencies

This table shows a list of the available output frequencies for `sdi_tx_clk`.

Input Frequency (hsma_clk_in_p2) (MHz)	Output Frequencies from u_sdi_pll (MHz)
148.5	148.5, 74.25, or 27

If the clock frequencies and formats are compatible, you can lock the video output to the frequency of the video input stream that the reference design receives on the SDI RX (J12) connector of the Arria V GX FPGA Starter Kit. The reference design uses the `sdi_clk148_up` and `sdi_clk148_dn` pins to control the exact frequency of the `clk_148_p` signal from the Si571 VCXO. The Clock Video Input MegaCore function on SDI input channel 1 divides the `vid_clk` input signal and provides the refclk_div output signal. Similarly, the Clock Video Output MegaCore function on SDI output channel 1 divides the `vid_clk` input signal and provides the `vcoclk_div` output. The phase frequency detector block of the reference design compares the relative phases of the two divided clock signals and adjusts the `sdi_clk148_up` and `sdi_clk148_dn` pins accordingly

A 148.5 MHz clock must drive the `rx_serial_refclk` input clock to each SDI MegaCore function, to allow triple standard detection. A 148.5 MHz clock must drive the `tx_serial_refclk` input clock that is locked to the video output clock, `sdi_tx_clk`. In the reference design, the `clk_148_p` signal is the source clock for the `rx_serial_refclk` and `tx_serial_refclk` input signals of the SDI MegaCore functions.

Generator Lock

The *generator lock* can lock the timing of video outputs to a reference source. Sources locked to a single reference clock allow switching cleanly between video inputs. The reference design offers the following two types of generator lock:

- Genlock—aligns the horizontal (`hsync`) and vertical (`vsync`) timing of the output video to a reference source with the same format and frame rate. For example, 1080p60 to 1080p60.
- Framelock (crosslock)—aligns the start of frame of the output video to a reference source of a different format with a lockable frame rate. For example, 720p60 to 1080i60, or 720p30 to 1080p60.

For the rest of this topic, the term *genlock* refers to both genlock and framelock.

The reference design uses the `sdi_in_1` clock video input block as a reference source to lock video output channel 1. The reference design achieves locking in the following two stages:

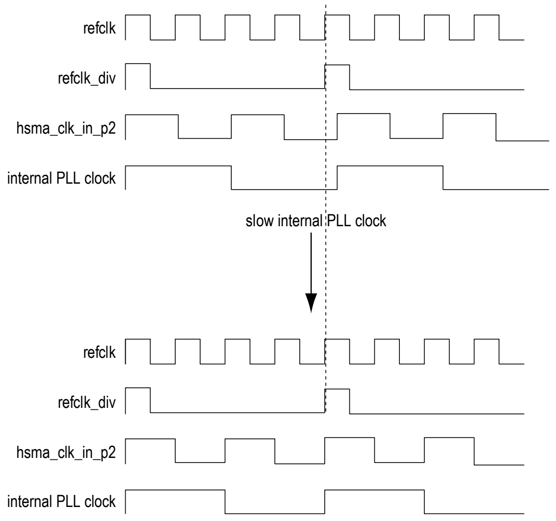
1. Clock locking—aligns the output video clock to the input video clock and tracks changes.
2. Frame locking—aligns the start of frame (SOF) of the output video to the SOF of the input video.

Clock Locking

The reference design uses the VCXOs on the Arria V GX FPGA Starter Kit to perform clock locking. The control software reads back the clock frequency of the input video from the registers of the Clock Video Input MegaCore function. The software compares this clock frequency to the clock frequency of the output video and determines the divider values for both the input and output video clocks that produce the same period.

The Clock Video Input MegaCore function outputs `refclk_div`, a divided down version of the input video clock. The phase frequency detector (PFD) block of the PLL on the SDI HSMC board compares the phase of the `refclk_div` to its internally generated clock and—using the VCXO to speed up or slow down the internal clock—matches changes in `refclk_div`.

Figure 3: Clock Locking



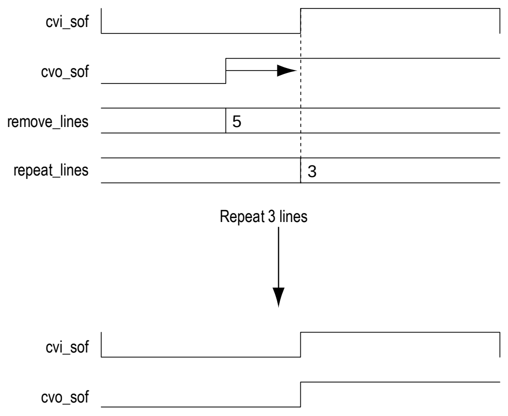
Frame Locking

The Clocked Video Output MegaCore function performs frame locking by aligning the SOF of the output video to the SOF that the Clocked Video Input MegaCore function produces. Each clocked video output block receives the SOF signal from the `sdi_in_1` clock video input block. If the input video format and the output video format in a video processing channel have lockable frame rates, the output video is frame locked to `SDI_IN_1`.

The Clocked Video Input MegaCore function outputs a SOF signal that the reference design connects to the Clocked Video Output MegaCore function. A rising edge transition indicates a SOF. You can set the SOF position anywhere within the input video frame by loading the registers of the Clock Video Input MegaCore function (and Clocked Video Output MegaCore function) with the sample and line (measured from the rising edge of the V sync) on which the SOF is to occur.

Figure 4: Frame Locking

An example of SOF transitions.



The Clocked Video Output MegaCore function compares the two SOF signals to determine how far apart the signals are. The Clocked Video Output MegaCore function then repeats or removes that number of samples and lines in the output video to align the two SOF signals. If the SOFs are less than a threshold value of samples apart (the value of the `vcoclk` divider register), the Clocked Video Output MegaCore function does not alter the output video.

Latency

When a channel is frame locked, the latency of the channel is one field for interlaced inputs and one frame for progressive input. The delay occurs in the frame buffer. When a channel is not frame locked, the latency increases by y lines and x samples, where x and y vary up to the total width and height of a field or frame.

External Memory

The reference design shares a single external DDR3 SDRAM for video buffering, OSD drawing, and the Nios II control software. The Qsys system includes a single SDRAM controller to coordinate accesses to this memory.

The memory subsystem minimizes the impact of bank management commands (nondata transfer) commands that the reference design sends to the DDR3 SDRAM, by supporting long bursts, large on-chip buffers, and memory bank interleaving.

Long bursts (within a row) minimize the impact of the initial setup commands. Making the deinterlacer and frame buffer burst size a factor of the row size (32 words) ensures that a burst does not cross a row boundary.

Large on-chip buffers (in the deinterlacer and frame buffer) are necessary to buffer enough data to create and receive the long bursts. By making the buffer size twice the burst size (64 words in the GUI), the reference design can transfer a burst to and from the DDR3 SDRAM while the reference design processes the next or previous burst. This action allows the video processing datapath to cope with the longer latencies that the arbitration of a multi-master system creates, all performing long bursts. The reference design uses on-chip buffers for crossing the clock domains between the Video and Image Processing Suite IP cores and the memory controller. This action allows you to run the memory at a higher frequency without affecting the speed at which the video datapath runs.

Bank interleaving reduces the penalty for switching rows by overlapping the bank management commands of one bank with the data transfer to and from another bank.

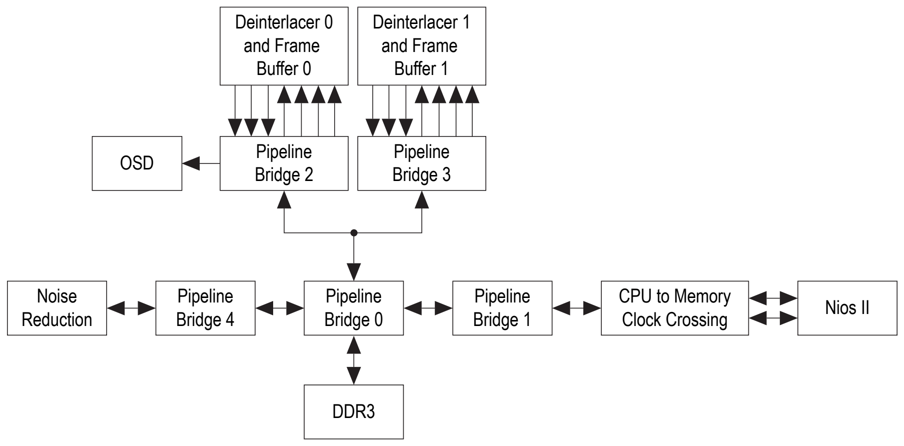
Memory Subsystem Architecture

The following masters require access to the external memory through the DDR3 SDRAM controller:

- Two Nios II processor masters (instruction and data masters)
- Five deinterlacer masters per video processing channel
- Two frame buffer masters per video processing channel
- Two noise reduction masters on the first video processing channel.
- One frame reader master for OSD logos
- With two channels of video the system has 19 masters.

The design cascades standard Qsys pipeline bridges to achieve an efficient memory subsystem, with four pipeline bridges connected via a fifth pipeline bridge to the DDR3 SDRAM controller.

Figure 5: Memory Subsystem Block Diagram



The following table describes the access patterns on each Avalon-MM master port, and shows the allocation of ports to different memory banks. The reference design allocates ports to different banks according to the base addresses in Qsys.

Table 4:

Master	Width (Bits)	Max Burst Size (Words)	Description	Memory Bank
0 and 1	32	1	Nios II processor bidirectional instruction and data ports	0
2	128	64	Frame buffer 1 - write port	1
3	128	64	Frame buffer 1 - read port	1
4	128	64	MA deinterlacer 1 - write port	2
5	128	64	MA deinterlacer 1 - read port 0	2
6	128	64	MA deinterlacer 1 - read port 1	2
7	128	32	MA deinterlacer 1 - motion write port	2
8	128	32	MA deinterlacer 1 - motion read port	2
9	128	64	OSD frame reader - read port	3

Master	Width (Bits)	Max Burst Size (Words)	Description	Memory Bank
10	128	Noise reduction – write port	4	
11	128	Noise reduction – read port	4	
12	128	64	Frame buffer 2 - write port	5
13	128	64	Frame buffer 2 - read port	5
14	128	64	MA deinterlacer 2 - write port	6
15	128	64	MA deinterlacer 2 - read port 0	6
16	128	64	MA deinterlacer 2 - read port 1	6
17	128	32	MA deinterlacer 2 - motion write port	6
18	128	32	MA deinterlacer 2 - motion read port	6

Bandwidth Calculations

Because the memory subsystem packs 20-bit colors in 256-bit data words, the design does not use some of the bits in a data word. You must add these bits to the bandwidth requirement calculation. The design incorporates the following basic calculations in the bandwidth calculation for each block:

256 bits / 20 bits = 12.8 samples

12 samples × 20 bits = 240 sample bits

256 bits – 240 bits = 16 bits

16 bits / 12 samples = 1.33 extra bits per 20-bit sample

MA Deinterlacer Bandwidth Calculation

For input format 1080i60, the input rate is:

1920 × 540 × 21.33 bits × 60 frames per second (fps) = 1.327 Gbps

For output format 1080p60, the output rate is:

1920 × 1080 × 21.33 bits × 60 fps = 2.654 Gbps

For motion format of one value per sample, the motion data rate is:

1920 × 540 × 8 bits × 60 fps = 0.498 Gbps

A memory access consists of the following operations:

- 1 write at input rate 1.327 Gbps
- 1 write at motion rate 0.498 Gbps
- 1 read at motion rate 0.637 Gbps
- 1.5 reads at total output rate 3.981 Gbps

which comes to a total bandwidth of:

1.327 + 0.637 + 0.637 + 3.981= 6.304 Gbps

Frame Buffer Bandwidth Calculation

For input format 1080p60, the input rate is:

$1920 \times 1080 \times 21.33$ bits $\times 60$ fps = 2.654 Gbps

For output format 1080p60, the output rate is:

$1920 \times 1080 \times 21.33$ bits $\times 60$ fps = 2.654 Gbps

A memory access consists of the following operations:

- 1 write at input rate 2.654 Gbps
- 1 read at output rate 2.654 Gbps

which comes to a total bandwidth of:

2.654 + 2.654 = 5.308 Gbps

Noise Reduction Calculation

Each frame or field of processing has one read from memory (of historically stored motion and IIR information) and one write to memory (of new motion and IIR information). Each read or write is of the following size:

1920 x 540 x 26 x 60 = 1.617 Gbps (per access)

Note: **1.** Interlaced fields assumed for system of "noise reducer followed by deinterlacer"
2. The number 26 comes from: 20 bits for 4:2:2 pixel data (IIR information) and 6 bits for motion field information per pixel.

Total bandwidth = 2 x 1.617 => 3.234 Gbps.

OSD to Frame Reader Bandwidth Calculation

For OSD output format 1080p60, the output rate is:

$1920 \times 1080 \times 32$ bits $\times 60$ fps = 3.981 Gbps

A memory access consists of 1 read at OSD output rate 3.981 Gbps, which comes to a total bandwidth of 3.981 Gbps.

Nios II Processor Bandwidth Calculation

The Nios II memory usage in hardware is approximately 0.5 Gbps for control and OSD update functions.

Total System Bandwidth Calculation

Table 5: Total System Bandwidth Calculation

This table shows the calculations of the total bandwidth for the video paths only and for the video paths plus OSD and Nios II processor.

Block	Bandwidth (Gbps)
Video paths only:	
MA Deinterlacer 1	6.304
MA Deinterlacer 2	6.304
Frame buffer 1	5.308
Frame buffer 2	5.308
Noise Reduction 1	
Video path total	23.224
Video paths with OSD and the Nios II processor:	
Video path total	23.224
OSD to Frame reader	3.981
Nios II processor	0.5
Total	27.705

Memory Bandwidth Requirements

A number of factors determine memory bandwidth efficiency, such as randomness of addresses, refresh rate, turnaround times between reads and writes, and burst lengths. Altera memory controllers can reach an efficiency of up to about 90% if the access conditions are right (long bursts of writes to the same column followed by long bursts of reads).

The maximum theoretical bandwidth for the reference design is:

333 MHz × 32 bits × 2 (double rate: both clock edges used) = 21.3 Gbps

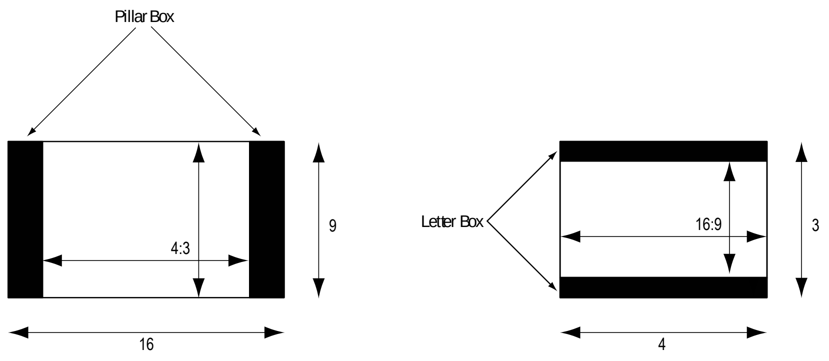
The available memory bandwidth is not sufficient to process two 1080i60 inputs simultaneously. Because of this limitation only SD-SDI is supported when both input channels are active. It is intended to address this in future versions of the UDX6 design by increasing the memory clock rate.

AFD

The AFD is a standard code that enables the design to correctly interpret the active picture region of the video stream. The code defines the areas of the active picture containing valid video data and the areas that are unused. This encoding allows the reference design to correctly transmit and display content with one aspect ratio with a different format. For example, video data formatted for a 4:3 aspect ratio displays correctly in 16:9 ratio if the AFD code of the incoming data indicates its 4:3 format.

Figure 6: Example Aspect Ratios And Their Displays

The figure shows two common aspect ratios and the unused areas when data in one format displays in the other format.

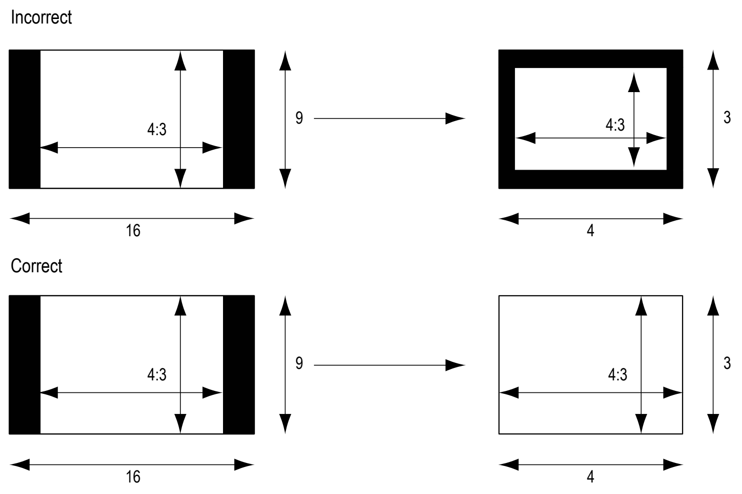


In the reference design, the scaler and mixer maintain the aspect ratio. When you set the output resolution, the Nios II processor compares the output aspect ratio with that of the incoming video content. If the aspect ratio is different, the reference design parameterizes the scaler to maintain the input aspect ratio for upscaling or downscaling, and the mixer adds pillar box or letter box bars to allow the video content to display with the correct aspect ratio.

SDI input streams include their own AFD codes. The reference design uses the AFD code to maintain the aspect ratio of the incoming video by having AFD clipper blocks clip the pillar box or letter box bars when necessary.

Figure 7: Incorrect and correct AFD Handling

The figure shows what can happen if the design does not read the AFD code and does not clip the bars.



For more information about the ancillary packets, about the handling of AFD data by the clocked video input and the clocked video output and on the AFD extractor and inserter, refer to the Video and Image Processing Suite User Guide.

The reference design software recognizes and inserts only a limited set of AFD codes. To easily extend the software to support all the AFD codes, change the `update_coefficients()` function in the **Video_Input_Channel.hpp**. The comments for this function include a full list of AFD codes.

Table 6: Supported AFD Codes

This table lists the supported codes, but does not show the unextended software ignored AFD codes.

Aspect Ratio(Corresponding Aspect Ratio Code)	AFD Code	Description
4:3 (0)	8	4:3 full frame
4:3 (0)	10	16:9 centered
16:9 (1)	8	16:9 full frame
16:9 (1)	9	4:3 centered

For a more detailed description of the AFD codes, refer to the Society of Motion Picture and Television Engineers (SMPTE) Standard 2016-1-2007, Format for Active Format Description and Bar Data and updates

Related Links

Video and Image Processing Suite User Guide

www.smpte.org

System Memory Map

Before you run the reference design, you must load the control software to the flash memory. When the reference design runs, the boot loader copies the software from the flash memory to the code and data sections of DDR3 SDRAM, from which it is run by the Nios II processor. You can locate the boot loader at address `0x2 5 280000` (offset `0x 5 280000` in the flash memory device), which is the CPU reset address.

The OSD logos blended with the video image by the OSD mixer are in an uncompressed **.zip**. You can update the **.zip** with your preferred OSD logos before you store the **.zip** at offset `0x 600 0000` in the flash memory device.

The hardware image file that you store at offset `0x 164 0000` in the flash memory device configures the FPGA when the board powers up. To load the hardware image file that you defined, you must press PGM_SEL (S2) until the middle LED (D25) illuminates, then press PGM_CONFIG (S1) to reconfigure the FPGA.

Table 7: Nios II Processor Instruction Master Memory Map

This table shows the memory maps for the Nios II processor and the flash memory device.

Slave Device	Description	Nios II Processor Address Base or Range	Memory Bank
Flash memory	—	`0x20000000` to `0x2 7 FFFFFFF`	—

Slave Device	Description	Nios II Processor Address Base or Range	Memory Bank
DDR3 SDRAM	Unused	`0x1E000000`	7
	Deinterlacer 2 base	`0x1C000000`	6
	Frame buffer 2 base	`0x1A000000`	5
	Noise reducer (Frame 2) base	`0x19000000`	4
	Noise reducer (Frame 1) base	`0x18000000`	
	Frame reader (Frame 2) base	`0x17000000`	3
	Frame reader (Frame 1) base	`0x16000000`	
	Deinterlacer 1 base	`0x14000000`	2
	Frame buffer 1 base	`0x12000000`	1
	Nios II software code and data	`0x10000000` to `0x11FFFFFF`	0

Table 8: Flash Memory Map

This table lists the offset ranges of the different blocks in the flash memory device.

Memory Block Use	Size (KB)	Flash Memory Device Offset Range
Unused	128	0x07FE0000–0x07FFFFFF
User software image	46,464	0x05280000–0x07FDFFFF
Factory software image	8,192	0x04A80000–0x0527FFFF
.zip file system	8,192	0x04280000–0x04A7FFFF
User hardware image 2	22,656	0x02C60000–0x0427FFFF
User hardware image 1	22,656	0x01640000–0x02C5FFFF
Factory hardware image	22,656	0x00020000–0x0163FFFF
Parallel flash loader option bits	32	0x00018000–0x0001FFFF
Board information	32	0x00010000–0x0001FFFF

Memory Block Use	Size (KB)	Flash Memory Device Offset Range
Ethernet option bits	32	0x00008000–0x0000FFFF
User design reset vector	32	0x00000000–0x00007FFF

System Debug Features

Avalon-ST Video Monitor

The Avalon-ST Video Monitor is a debugging tool on an Avalon-ST Video protocol connection between two video and image processing IP cores. The module streams back information on the video stream flowing through to the host PC in real time. The module extracts, encodes, and returns important Avalon-ST Video metadata to help you track where, why, or when a system is breaking.

The reference design integrally monitors control packets and user packets. The reference design cannot fully stream back image data packets to the debug host PC for obvious bandwidth limitations on the JTAG cable, but the reference design streams back their size nevertheless to allow for the detection of inconsistencies with the resolution in preceding control packets.

Trace System

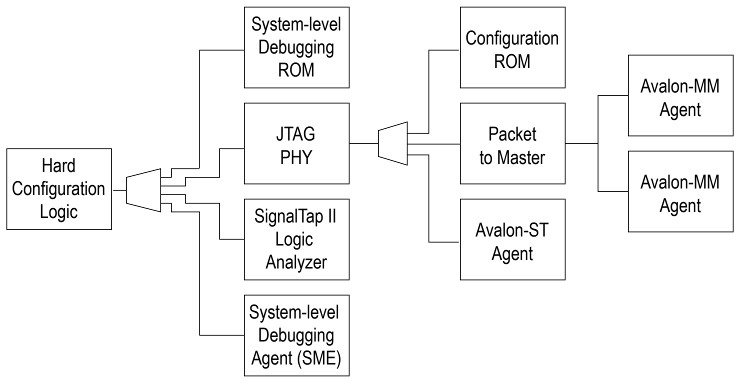
The trace system collects the debug information from multiple Avalon-ST Video Monitors. It provides a single consistent timestamp across multiple monitors. It also provides shared buffering for debug information, to allow the individual monitors to have a lower resource usage.

Hardware Debugging Tools

JTAG connects the host PC to the hardware agents running on the board.

Figure 8: Debugging System on the Board

The figure shows a debugging hub arbitrates accesses of the various agents to the JTAG PHY.



The Quartus II software may automatically insert the components of a debugging system, for example, the SignalTap II logic analyzer.

Running the Reference Design

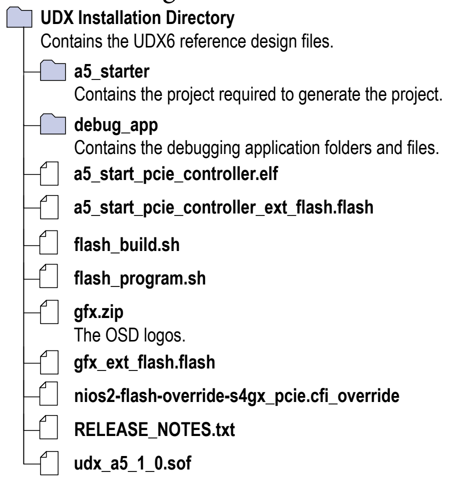
Installing the Reference Design

The **UDX6 _<*version*>.zip** includes all of the files necessary for this reference design. You can download this file from the Broadcast web page on the Altera website. To install the reference design, run the installed reference design and point the reference design to a designated directory for this project.

Note: Do not use spaces in any directory name.

Figure 9: Directory Structure

The following figure shows the directory structure and top-level files for the UDX6 reference design files after installing the files


UDX Installation Directory
Contains the UDX6 reference design files.
a5_starter
Contains the project required to generate the project.
debug_app
Contains the debugging application folders and files.
a5_start_pcie_controller.elf
a5_start_pcie_controller_ext_flash.flash
flash_build.sh
flash_program.sh
gfx.zip
The OSD logos.
gfx_ext_flash.flash
nios2-flash-override-s4gx_pcie.cfi_override
RELEASE_NOTES.txt
udx_a5_1_0.sof

Related Links

Broadcast

Connecting the Hardware

To connect the hardware to run the UDX6 reference design, follow these steps:

1. Ensure that you turn off the power switch (SW5) on your Arria V GX Starter Kit.
2. Connect an SDI video source to the SDI RX connector (J12) on the Arria V GX Starter Kit.
3. Connect an SDI monitor to the SDI TX connector (J11) on the Arria V GX Starter Kit.

   Note: To demonstrate the 1080p60 resolution capability of the design you require a 24-inch monitor that supports 1920x1080 pixels.

4. Connect either:
   a. A USB-Blaster download cable to your computer and to the JTAG Connector (J9) on the Arria V GX Starter Kit.
   b. An ordinary USB download cable to your computer and to the embedded USB-Blaster connector (J14) on the Arria V GX Starter Kit. For information about installing the USB-Blaster software driver on the host PC (located at <*quartus_install_dir*>**\drivers\usb-blaster**), refer to the USB-Blaster Download Cable User Guide.
5. Connect your Arria V GX Starter Kit to a power supply.

Related Links

USB-Blaster Download Cable User Guide

Using the Precompiled .sof and Flash Files for a Quick Start

You can compile the design and build a **. sof** programming file with the Quartus II software, build a Nios II software image and use the flash programmer to download the files to the board. However, to directly download to the board the precompiled files included with the design, follow these steps:

1. Start a Nios II command shell:

   a. On Windows OS, on the Start menu, point to **All Programs**, point to **Altera**,point to **Nios II EDS** <*version*>, and click **Nios II** <*version*> **Command Shell**.

   b. On Linux OS, type the following command:

   ```
   $SOPC_KIT_NIOS2/sdk_shell
   ```

   Note: For more information about the Nios II command shell, refer to Nios II Command-Line Tools.

2. Change to the directory that contains the precompiled files for your board, by typing <*path to UDX6* >.
3. Turn on the power switch (SW5) on the Arria V GX Starter Kit.
4. Program the board with the precompiled **. sof**, by typing the following command:

   ```
   quartus_pgm -m jtag -o p\;udx_a5_1_0.sof
   ```

5. Use the **flash_program.sh script** provided with the design to download the files to the board:

   ```
   ./flash_program.sh
   ```

6. Load the UDX6 reference design by pressing the PGM_SEL push button (S2) until LED D25 (the middle of the three options) illuminates.
7. Then press the PGM_CONFIG push button (S1) to configure the FPGA from the selected flash page.

   The design starts in independent mode. The SDI output displays the input channel .

Related Links

Nios II Command line Tools

Compiling the Quartus II Project

To compile the Quartus II project, follow these steps:

1. To create the UDX6 reference design Quartus II Project File (**. qpf**), **udx _ a5_1_0 .qpf**, and set up the design, perform one of the following actions:

   a. In a Windows OS, double-click the **make_project_ udx_a5 .bat**.

   b. On the Linux command line, type the following command:

   ```
   sh make_project_udx_a5.sh r
   ```

2. Open the **udx_a5_1_0 .qpf**.
3. On the Assignments menu, click **Device**. The **Device Settings** dialog box appears.
4. In the Quartus II software, on the Tools menu, click **Qsys**.
5. Select **UDX _A5_1_0 .qsys** and click **Open**.
6. Review the UDX6 reference design Qsys system..
7. In Qsys, click the **Generation** tab.
8. For **Create simulation model**, select **None**. System generation runs faster when simulation is off.
9. Click **Generate**. System generation may take a few minutes.
10. When the **Generate completed** message displays, click **Close**.
11. If you are prompted to save your changes, click **Save**.

12. In the Quartus II software, on the Processing menu, click **Start Compilation**. Compilation creates the **udx_a5_1_0 .sof**.
13. After compilation completes, on the Tools menu, click **Programmer**.
14. In the Quartus II Programmer, under **Mode**, verify that **JTAG** is selected.
15. Click **Hardware Setup** to configure the programming hardware. The **Hardware Setup** dialog box appears.
16. In the **Hardware** column, double click **USB- Blaster II**.
17. Click **Close** to exit the **Hardware Setup** window.
18. In the Quartus Programmer, click **Add File**, navigate to the **udx_a5_1_0 .sof**.
19. In the Quartus II software, on the Processing menu, click , and click **Open**.
20. Ensure the **Program/Configure** box for the new file contains a checkmark.
21. Click **Start**. The software downloads the **. sof** to your Arria V GX Starter Kit and configures the FPGA with the hardware image.

Related Links

System Design with Qsys

Quartus II Programmer

Compiling the UDX6 Reference Design Software

To compile and download the UDX6 reference design software in the Nios II SBT for Eclipse, follow these steps:

1. On the **Start** menu, point to **Programs**, point to **Altera**, point to **Nios II EDS** <*version*>, click **Nios II** <*version*> **Software Build Tools for Eclipse**.

   Note: If the Nios II SBT for Eclipse welcome screen appears, click **Workbench** to continue.

2. If the **Workspace Launcher** dialog box appears, click **Browse** to navigate to your unzipped **a5_starter** directory and create a new workspace folder.
3. Right-click the **Project Explorer** tab. A menu appears.
4. On the menu, click **New** and select **Nios II Application and BSP from Template**.
5. For **SOPC Information File name**, browse to **a5_starter / UDX _A5_1_0 .sopcinfo**.
6. For **Project name**, type `a5_start_pcie_controller`.
7. In the **Templates** list, select **Blank Project**.
8. Click **Finish**. The Nios II SBT for Eclipse creates and adds your project to the **Project Explorer** tab.
9. After your project is created, in the **Project Explorer** tab, right-click **a5_start_pcie_controller _bsp**, and on the Nios II menu, click **BSP Editor**.
10. In the Main tab under Settings > Common, select **None** for **stderr** and for **stdout**.
11. In the **Linker Script** tab, for the linker region **mem_if _ddr3 _emif_ 0**, set the **Size (bytes)** value to 33553824. This value limits the Nios II address space to 32 MB and prevents the stack from overlapping an area of memory that a frame buffer or deinterlacer uses.
12. In the **Software Packages** tab, enable the **altera_ro_zipfs** software package and set its parameter values.

Table 9: altera_ro_zipfs Software Package Settings

Parameter	Value
`ro_zipfs_base`	0x20000000
`ro_zipfs_name`	/mnt/gfx
`ro_zipfs_offset`	0x6000000

13. When the software prompts you to save changes, click **Yes, Save**.

14. In the Nios II SBT for Eclipse, in the **Project Explorer** tab, left-click **a5_start_pcie_controller** to expand the list of files. For each **. cpp**, right-click and select **Add to Nios II Build**.
15. Right-click **a5_start_pcie_controller _bsp**. In the Nios 2 menu, click **Generate BSP**.
16. If a target connection error occurs, enable the **Ignore mismatched system ID** and **Ignore mismatched system timestamp** options.
17. In the Nios II SBT for Eclipse, in the **Project Explorer** tab, right-click **a5_start_pcie_controller**, and on the Run As menu, click **3 Nios II Hardware**. Your project compiles and the software downloads the resulting Executable and Linking Format File (**.elf)** to your Arria V GX Starter Kit.

Note: If a panel appears to inform you of an issue while programming the Nios II code on the board, ensure that the board is on, connected to the PC with the JTAG blaster, and that the **. sof** has successfully downloaded. To establish a successful connection, highlight **cpu** in the **Processors** section. Also ensure that you disable the Nios II console view, or highlight **jtag_uart** for your **Byte Stream Device**.

Related Links

Getting Started with the Graphical User Interface

Nios II Software Build Tools

Changing the OSD Logos

The **gfx.zip** contains the OSD logos in 8-bit raw RGBA format. To replace OSD logos in the UDX6 reference design, you must format and add the new logos before you program the **gfx.zip** to flash memory. This topic describes how to format and add OSD logos in this file.

If you do not want to add new OSD logos to the UDX6 reference design, skip this topic and continue with Programming Flash Memory.

This topic describes how to convert a new OSD logo to the correct format with the Gimp software from www.gimp.org. Before you perform the instructions, you must install the Gimp software on your computer.

Note: The Gimp software instructions in this topic are correct for the Gimp software v2.6.8. For different versions of the Gimp software, you must determine the correct method to implement the format conversion steps.

After you install the Gimp software, to convert an image to an OSD logo, follow these steps:

1. Open the image in the Gimp software.
2. To convert the image to the RGB color space, on the **Image** menu, click **Mode** and select **RGB**.
3. If the image does not have an alpha (transparency) specification, you can add an alpha channel:
   a. On the **Layer** menu, click **Transparency** and select **Add Alpha Channel**.
   b. In the Toolbox, click the Eraser Tool.
   c. In the Toolbox, in the Eraser settings, adjust the **Opacity** setting to set the level of transparency.
   d. Move the Eraser Tool in the image to select areas of the logo to be made transparent.
4. To save the modified logo in the correct format, follow these steps:
   a. On the File menu, click **Save As**.
   b. For Name, type <*logo name*>`. rgba`.
   c. Expand **Select File Type**.
   d. Click **Raw image data**.
   e. Click **Save**. The **Raw Image Save** dialog box appears.
   f. For **RGB Save Type**, select **Standard (R ,G,B)**.
   g. For **Index Palette Type**, select **B ,G,R,X (BMP style)**.
   h. Click **OK**.
5. Determine the height (<*height*>) and width (<*width*>) of the modified logo.
6. To add the new logo to the **gfx.zip**, follow these steps:

a. Open **gfx.zip** in WinZip.
b. Drag the new logo to the WinZip file window. The **Add** dialog box appears.
c. If the **Compression** option is not set to **None**, click **Change Compression** and select **None**, and click **OK**.
d. Click **Add**.

7. Open the **a5_starter /software/ udx_A5_processor /main.cpp** in a text editor.
8. Replace the following line in the file:

   gfx_load_passed = gfx_load_passed && \\osd_channel.load_logo("/mnt/gfx/ibc-logo.rgba",132,43)

   with

   gfx_load_passed = gfx_load_passed &&
   \\osd_channel.load_logo("/mnt/gfx/<*logo name*>.rgba",<*width*>,<*height*>)

9. If your logo <*width*> × <*height*> exceeds the currently specified maximum number of pixels (by default, 262,144 pixels), open the file **a5_starter /software/ udx_a5_processor /OSD_Input_Channel.hpp** in a text editor and modify the following line to specify the minimum, sufficiently large number of pixels for your image:

   #define MAX_NO_OF_PIXELS 262144

Related Links

Programming Flash Memory on page 29

Programming Flash Memory on page 29

www.gimp.org

Programming Flash Memory

Use the Nios II Flash Programmer to program the UDX6 reference design in flash memory on the Arria V Starter Kit, and then reboot from the flash memory.

If you want to add your own OSD logos or modify the default set, you must follow the instructions in *Changing the OSD Logos* on page 28 before following the instructions in this topic.

To program the UDX6 reference design **. sof**, **.elf**, and OSD logos to flash memory, follow these steps:

1. In the Nios II SBT for Eclipse, in the **Project Explorer** tab, right-click **udx_a5_processor _bsp**, and on the Nios II menu, click **Flash Programmer**. The Nios II Flash Programmer GUI appears.
2. In the Nios II Flash Programmer, on the File menu, click **New**. The **New Flash Programmer Settings File** dialog box appears.
3. Select **Get flash programmer system details from BSP Settings File**.
4. Browse to locate the **a5_starter /software / udx_a5_processor _bsp/ settings.bsp**.
5. Click **OK**.
6. Click **Hardware Connections**. The **Hardware Connections** dialog box appears.
7. Click **Refresh Connections**. The **cpu** processor appears in the **Processors** list.
8. Click **Close**.
9. Under **Files for flash conversion**, click **Add**.
10. Browse to locate the **a5_starter /software/ udx_a5_processor / udx_a5_processor .elf**.
11. Click **Select**.
12. Under **File generation command**, click **Properties**.
13. The following table lists the values to set in the **Properties** dialog box.

Table 10: File Generation Command Property Settings for udx_a5_processor.elf

Property	Value
CPU reset address	`0x25280000`
Flash base address	`0x20000000`
Flash end address	`0x28000000`

--pfl --optionbit=0x18000 --programmingmode=PS

14. Click **Add**.
15. Browse to locate the **a5_starter /gfx.zip**.
16. Click **Select**.
17. Double-click the **gfx.zip** entry row in the **Flash Offset** column and type the value `0x0600 0000`.
18. Click **Start** to program all of the files in the flash memory device.

To reboot the reference design from the newly programmed flash memory device, follow these steps:

1. Select the correct page in flash by pressing the PGM_SEL push button (S2) until LED D25 (the middle of the three options) is lit.
2. Press the PGM_CONFIG push button (S1) to configure the FPGA from the selected flash page.

Related Links

Changing the OSD Logos on page 28

Nios II Flash Programmer User Guide

Status Displays

The LCD screen displays messages that describe the state of the software initialization and start up.

Table 11: LCD Status Messages

Message	Description
Init UDX AV ...	The software sends this message to the LCD when running.
GFX load failed	The software fails to load the logos from the **gfx.zip** in flash memory. In this case, the software exits.
GFX load passed	The software successfully loads the logos from the **gfx.zip** in flash memory.
ch 1 <*format*>	Video input data in format <*format*> is detected on Channel 1.
ch 2 <*format*>	Video input data in format <*format*> is detected on Channel 2.

After the software begins running on the Nios II processor, the Arria V GX Starter Kit LEDs display the current running status of the system.

Table 12: LED Status Information

User LED	Status
0 (D20)	Indicates one of the clocked video output channels has underflowed.
1 (D21)	Indicates one of the clocked video input channels has overflowed.
2 (D22)	Currently unused
3 (D23)	Heartbeat. This LED flashes when the software is running.
PCIe x1 (D16)	Indicates SDI IP core status for onboard SDI RX.
PCIe x4 (D17)	Indicates DDR3 memory initialization succeeded.
PCIe x8 (D18)	Currently unused.
PCIe g2 (D19)	Currently unused.

Table 13: UDX6 Reference Design Modes

Mode	Meaning
0	• Independent mode. This independent mode setting is the initial mode in which the reference design starts up. • Channel 1 output data is sent to the SDI TX video port (J11) on the Arria V GX Starter Kit. • Channel 2 output data is discarded inside the FPGA.
1	• Multi-view mode: channel 1 multi-view, channel 2 test pattern. • Channel 1 output data is sent to the SDI TX video port (J11) on the Arria V GX Starter Kit. • Channel 2 output data is discarded inside the FPGA.

User Controls

While the software is running, you can control the run-time configurable parameters with the Arria V GX Starter Kit push-button switches.

Table 14: PushButton Switch Controls

Push-Button Switch	Function
S5 (PB0)	Cycles between the two modes. You must synchronize operations involving the mixer blocks carefully before switching between independent mode and multi-view mode.

Push-Button Switch	Function
S6 (PB1)	• Changes the output resolution of channel 1: • SDI output cycles through 720p60, NTSC, and 1080i60.
S7 (PB2)	• Changes the output resolution of channel 2: • Currently not connected to any physical output.
S4 (CPU Reset Push-Button Switch)	Resets the UDX6 reference design by restarting the software.

Using the Debugging System

To use the debugging system, follow these steps:

1. Open the UDX6 design within Qsys, **UDX_A5_1_0.qsys**
2. On the Qsys menu, click **Tools** and select **System Console**.
3. In the **System Explorer** panel, under the **devices** folder, right click on the Arria V device.
4. Under **Link device to**, select **udx_a5_1_0.q p f**
5. In the **Tools** menu of **System Console**, select **Trace Table View .**
6. In the **Capture Settings** window choose the UDX6 design from the **Select Hardware** drop down box.
7. For the **Avalon-ST Video Monitor** that you wish to probe, tick the **Value** checkbox and click **OK**.
8. Start the data capture by pressing the **Start** button (red arrow and magnifying glass icon).
9. Examine the captured Avalon-ST Video control packets.
10. Close the **System Console** application.