High-Definition Video Reference Design (UDX4)

AN-627-1.1 Application Note

The Altera® high-definition video reference designs deliver high-quality up, down, and cross conversion (UDX) designs for standard-definition (SD), high-definition (HD), 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.

For more information about the video series of reference designs, refer to the Broadcast page on the Altera website or refer to the *High Definition Video Reference Design (V1)*, *High Definition Video Reference Design (V2)*, and *High-Definition Video Reference Design (UDX3)* application notes.

The UDX4 reference design improves on the UDX3 reference design by including the latest Deinterlacer II and Scaler II video IP cores, which are part of the Video and Image Processing Suite.

For information about these IP cores, refer to the *Video and Image Processing Suite User Guide*.

The new Deinterlacer II IP core significantly improves image quality with better low-angle edge interpolation and 3:2 or 2:2 cadence detection for perfect deinterlacing of film content. Video latency through the Deinterlacer IP core was approximately one field in the UDX3 reference design; in the UDX4 reference design it is now a couple lines—except when deinterlacing cadenced content, as for algorithmic reasons, latency cannot fall below one field.

The Scaler II IP core now supports subsampled data. The UDX4 reference design uses this new feature to perform a significant proportion of the video processing in the YCbCr color space with subsampled 4:2:2 data. This feature gives a slightly detrimental visual effect on the digital video interface (DVI) to DVI video datapath, but reduces the memory bandwidth requirements and the resource usage of the design.

The UDX4 reference design uses a prototype debugging extension of the video and image processing framework. The design contains multiple instances of the video trace module to monitor the Avalon® Streaming (Avalon-ST) Video connections between video and image processing IP cores from a PC hosting the debugging session. The reference design also contains instrumentation to monitor the multiport front end and gather general statistics related to the traffic between the multiport front end and the DDR SDRAM controller on the Avalon Memory-Mapped (Avalon-MM) interface.

101 Innovation Drive
San Jose, CA 95134
www.altera.com

Features

- Files for targeting the Stratix IV GX development kit
- A unified memory architecture with a multiport front end to allow CPU, on-screen display (OSD), and video processing to use one 64-bit DDR3 SDRAM
- A switch block to allow run-time reconfiguration of the two video processing paths
- Two active inputs, run-time selectable from:
  - Two triple-rate serial digital interface (SDI) inputs: standard definition (SD), high definition (HD) or 3Gbps (3G), interlaced or progressive, any resolution and frame rate up to 1080i60
  - One DVI input: progressive, any resolution and frame rate up to 1080p60
- Two active outputs, run-time selectable from:
  - Two triple rate SDI outputs: SD, HD or 3G, interlaced or progressive, any resolution and frame rate up to 1080i60
  - One high-definition multimedia interface (HDMI) output: interlaced or progressive, any resolution and frame rate up to 1080p60
  - One DVI output: progressive, any resolution and frame rate up to 1080p60
- Two high-quality UDX video-processing paths:
  - Low-latency motion adaptive deinterlacer with low-angle edge interpolation and cadence detection
  - Scaler (12 × 12 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 video trace module IP monitors Avalon-ST Video protocol connections and routes protocol information to a debugging host in real time
  - The System Monitor and Trace Monitor applications running on the host control hardware debugging agents on the board, providing system information while running. For example memory bandwidth usage and performance through the multiport front end debugging port and real-time status of the system

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*. For more information about the Avalon-MM and Avalon-ST interfaces, refer to the *Avalon Interface Specifications*.

System Requirements

This section describes the hardware and software requirements to run the UDX4 reference design.

Hardware Requirements

The UDX4 reference design requires the following hardware components:

- Stratix IV GX FPGA Development Board

  For more information, refer to the *Stratix IV FPGA Development Board Reference Manual*.

- SDI HSMC board

  For more information refer to the *SDI HSMC Reference Manual*.

- Bitec HSMC DVI card

  For more information, refer to the Bitec website.

- Any combination of the following video sources:
  - Up to two SDI video sources that provide progressive or interlaced output up to 1080p60 resolution and frame rate with BNC connectors to connect to the SDI HSMC board
  - Up to one DVI video source providing progressive output up to 1080p60 resolution and frame rate

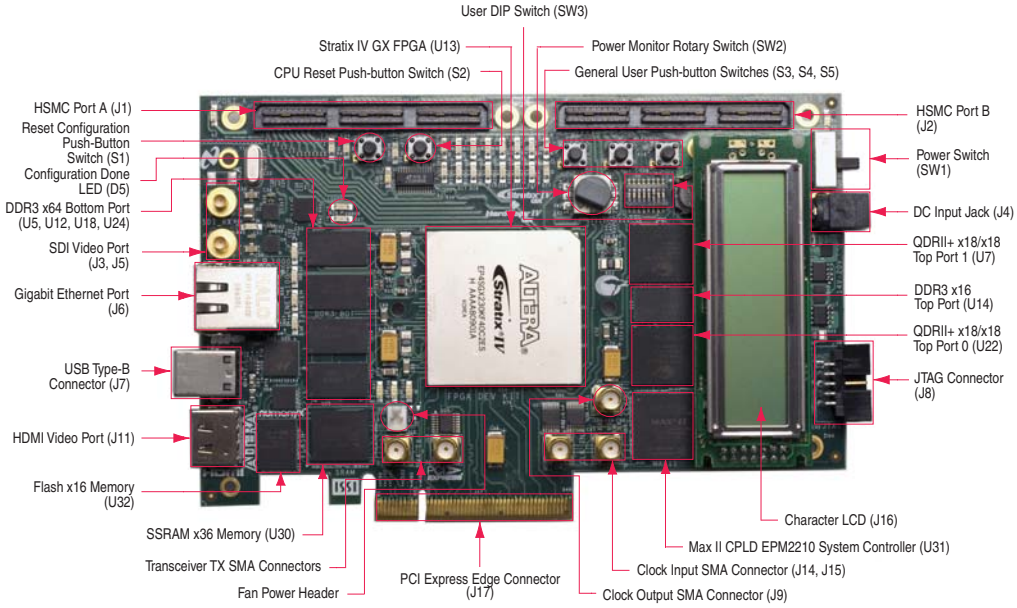
- Any combination of the following video displays:
  - Up to two monitors with DVI or HDMI inputs supporting 1920 × 1080 pixel resolution
  - Up to two SDI monitors that can support 1920 × 1080 pixel resolution

    or
  - Two 3G SDI-to-DVI converter boxes and two monitors with DVI inputs supporting 1920 × 1080 pixel resolution

The Altera Audio Video Development Kit, Stratix IV GX Edition provides both the Altera Stratix IV GX FPGA Development Board and the Altera SDI HSMC board. Alternatively, you can acquire the two boards separately, as the Altera Stratix IV GX FPGA Development Kit and the Terasic Transceiver SDI HSMC board. The Terasic Transceiver SDI HSMC board is available on the Terasic website. To acquire the Bitec HSMC DVI card, refer to the Bitec website.

Figure 1 shows the Altera Stratix IV GX FPGA development board.

Figure 1. Stratix IV GX FPGA Development Board



Software Requirements

The UDX4 reference design requires the following software:

- The Altera Complete Design Suite v11.0, which includes the Quartus II software, Qsys, Nios® II EDS, and MegaCore IP Library (including the Video and Image Processing Suite)
- Windows XP or Linux OS

Only the Windows XP OS supports the debugging applications to monitor the system on the host. These applications also require a Java Virtual Machine version 1.6. A 32-bit Java Virtual Machine must be used on a 64-bit operating system.

Resource Utilization

Table 1 shows the resource utilization for the UDX4 reference design.

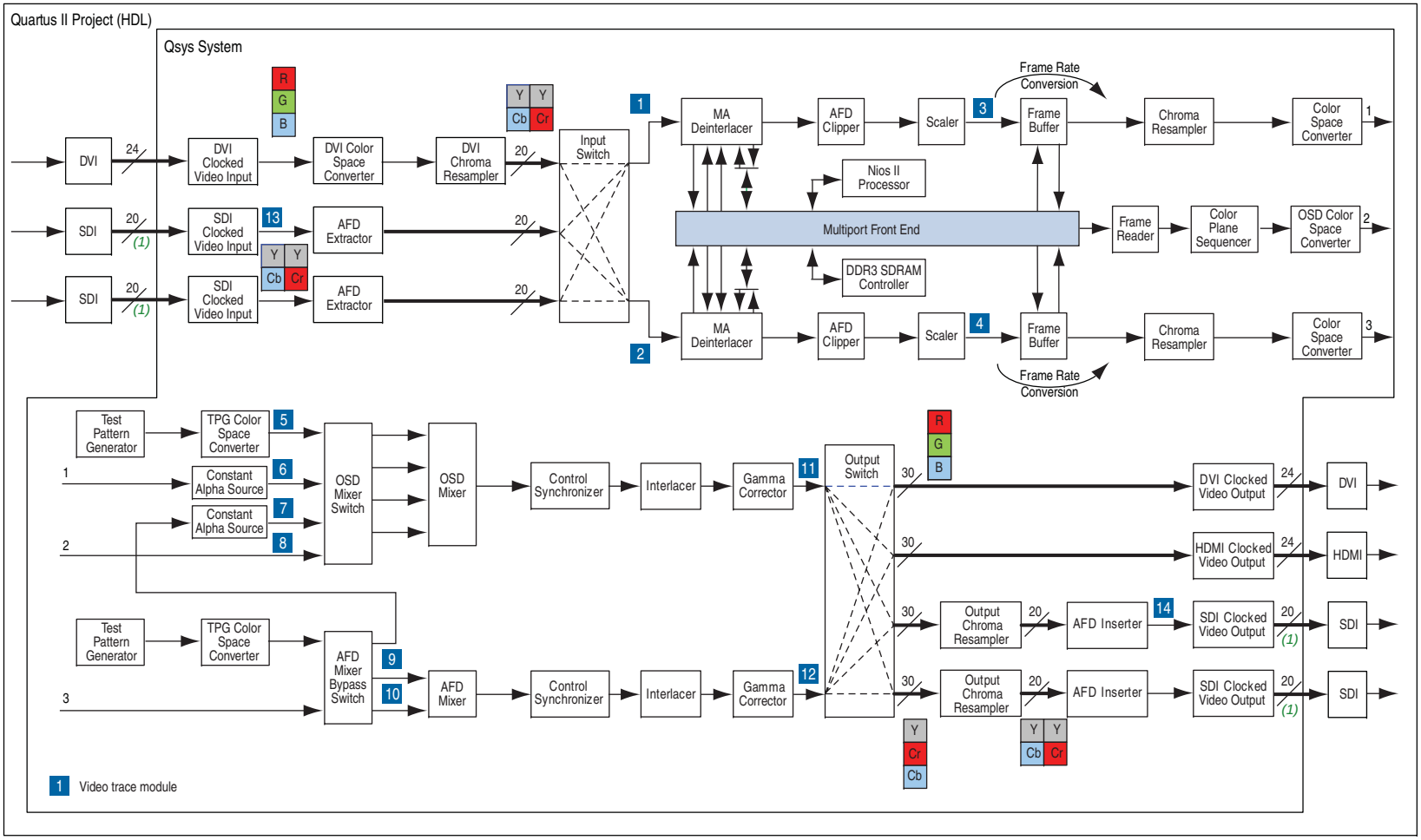
Table 1. Resource Utilization

Design	Combinational ALUTs	ALMs	Dedicated Logic Registers	M9Ks	M144Ks	18-bit DSPs
UDX4	78,500	70,000	96,500	528	10	208
UDX4 without debugging	70,000	62,500	87,500	462	10	208

Functional Description

Figure 2 shows a detailed block diagram of the UDX4 reference design.

Figure 2. UDX4 Reference Design Block Diagram



Note to Figure 2:

(1) In SD mode, only the 10 least significant bits of the 20-bit data width contain data. The other bits are unused.

The UDX4 reference design comprises two video processing paths. This section provides a functional description of the blocks in both paths.

UDX4 Reference Design Blocks

Most of the blocks in the UDX4 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 describes the blocks.

Table 2. Reference Design Blocks (Part 1 of 3)

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 (×2)	AFD Inserter	Inserts the AFD code (0 to 15) it receives on its Avalon-MM slave interface into 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 mixer bypass 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.
Video trace module	Video trace module	Fourteen video trace modules allow debugging of the design. The modules use a global timestamp. Arbitration and hardware use the JTAG cable to link between the video trace modules and the debugging host.
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 YCbCr input format to 10-bit RGB or YCbCr, 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.
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 ALTMEMPHY	Provides DDR3 SDRAM access with 64-bit wide, 400-MHz operations to 512 MB of memory split into eight banks. Runs in half-rate mode with a 256-bit 200-MHz Avalon-MM interface.
DVI clocked video output	Clocked Video Output	Inputs 10-bit video formatted with three colors in parallel and separate synchronization information, and outputs 8-bit DVI data that is the 8 most significant bits (MSBs) of the incoming 10 bits. The gamma converter upstream converts the 10-bit color values to 8-bit color values, thus removing the two LSBs.
DVI color space converter	Color Space Converter	Converts 10-bit red, green, blue (RGB) to 10-bit YCbCr (using the BT.709 specifications for HD). This conversion is not run-time configurable.

Table 2. Reference Design Blocks (Part 2 of 3)

Reference Design Block	MegaCore Function	Description
DVI input	—	DVI receiver chip on the Bitec DVI HSMC card.
DVI output	—	External DVI transmitter chip on the Bitec DVI HSMC card.
DVI chroma resampler	Chroma Resampler	Performs 4:4:4 to 4:2:2 conversion This conversion is not run-time configurable.
DVI clocked video input	Clocked Video Input	Shifts 8-bit incoming DVI data to eight MSBs of the ten bits, with 2 least significant bits (LSBs) tied to 0. Three colors in parallel and separate synchronization information.
Frame buffer (×2)	Frame Buffer	Provides run-time configurable double buffering, or triple buffering to support simple 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 DVI and HDMI output. Conversion is run-time configurable.
HDMI output	—	External HDMI transmitter chip on the Stratix IV GX FPGA development board.
HDMI clocked video output	Clocked Video Output	Inputs 10-bit video formatted with three colors in parallel and separate synchronization information, and outputs 8-bit HDMI data that is the 8 MSBs of the incoming 10 bits. The gamma converter upstream converts the 10-bit color values to 8-bit color values, thus removing the two LSBs.
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.
MA deinterlacer (×2)	Deinterlacer II	Deinterlaces video input using motion-adaptive (MA) algorithm with the following parameters: ■ Low-angle edge interpolator ■ Motion bleed enabled ■ 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
Multiport front end	Multiport Front End	Provides efficient access from multiple masters to the single external memory. Refer to "Memory Subsystem Architecture" on page 18 and to the *Multiport Front End User Guide* in the reference design installation.
Nios II processor	Nios II Processor	Embedded Nios II/f processor with a Level 1 JTAG debug module and an internal interrupt controller.
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.

Table 2. Reference Design Blocks (Part 3 of 3)

Reference Design Block	MegaCore Function	Description
OSD mixer	Alpha Blending Mixer	In multi-view mode, merges the two video input channel 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 mixer 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 (×2)	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.
Scaler (×2)	Scaler II	Scales images for 4:2:2 data in parallel in polyphase mode with 12×12 taps, run-time configurable output resolution and run-time configurable filter coefficients.
SDI input and output (×2)	SDI	SDI MegaCore function (not in the Video and Image Processing IP Suite) configured for triple standard SDI bidirectional communication. Although Figure 2 indicates the presence of four SDI blocks, actually there are two MegaCore functions—each MegaCore function handles both input and output SDI communication for one of the two video processing channels of the reference design.
SDI clocked video input (×2)	Clocked Video Input (Beta)	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 (×2)	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 any 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 it to 10-bit YCbCr format, depending on the format of the output video.

All of the MegaCore functions, except for the SDI MegaCore function, are in the Video and Image Processing Suite. For more information, refer to the *Video and Image Processing Suite User Guide*.

SDI MegaCore Function

The SDI MegaCore function is configured 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.

For more information about the SDI MegaCore function, refer to the *SDI MegaCore Function User Guide*.

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 run-time user input:

- Avalon-MM performance monitor for debugging
- A JTAG UART to display software `printf` output.
- An LCD display of reference design start-up information that includes the standard of the input video streams (refer to Table 11 on page 36).
- LEDs to display system status (refer to Table 12 and Table 13 on page 37).
- Push-button switches to allow switching between video output resolutions (refer to "User Controls" on page 37).

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 Stratix IV GX FPGA development board 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 several 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 Stratix IV GX FPGA development board to change the video processing channel connectivity and the resolution on the output displays. For details, refer to Table 14 on page 37.

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. For connectivity of the input and output channels, refer to Figure 2 on page 6.

Video Input Channel

A video input channel starts at a clocked video input block and extends to the color space converter. The reference design has two identical video input channels, `input_channel_1` and `input_channel_2`.

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 that precede the color space converter 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 any back pressure 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 in the case of 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 takes input from two video input channels and one OSD input channel
- The `afd_output_channel`, which takes input from only the second video input channel (`input_channel_2`).

Modes

The design has the following two run-time configurable modes:

- An independent mode, in which the two video processing paths operate independently. This mode is the default mode. In this 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 any input format to any output format.

- 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.

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 to multi-view mode, software sets the AFD mixer control synchronizer to perform the following steps on the next start-of-image packet:

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

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

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

1. Use the AFD mixer switch to route `input_channel_2` to the AFD mixer instead of the OSD mixer switch.
2. Disable the OSD mixer layer 2.

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

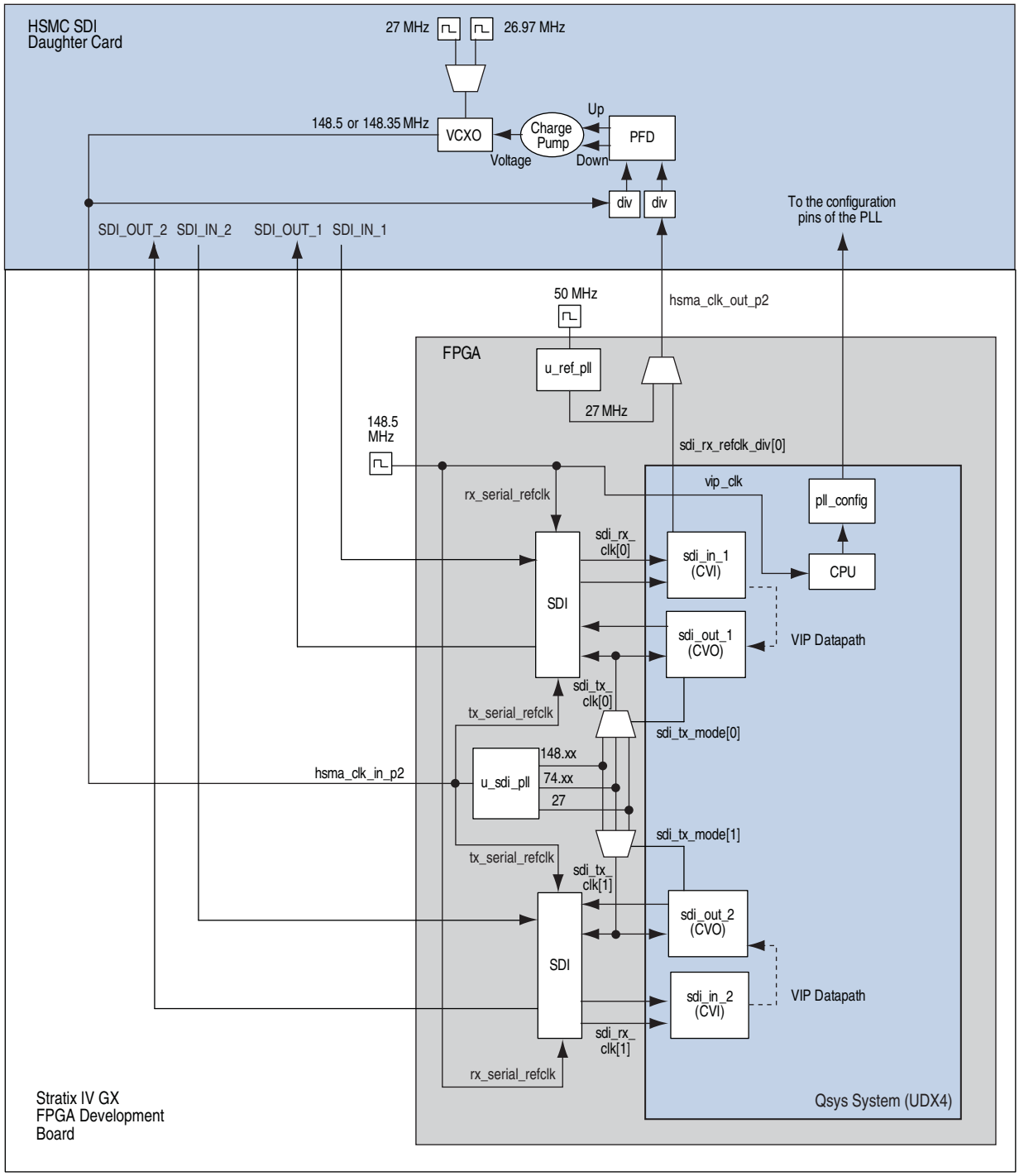
"User Controls" on page 37 provides information about how to change the mode at run time.

Clocking

The video processing datapath and the Nios II processor are clocked by the system clock `vip_clk` 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 memory subsystem, including the multiport front end, runs at 200 MHz; the DDR3 SDRAM runs at 400 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 frequency conforms to the relevant video protocol.

Figure 3 shows the top-level clock connections for SDI data and control.

Figure 3. Top-Level Clock Connections for SDI Data and Control



The SDI multi-frequency voltage controlled crystal oscillator (VCXO) femto clock video PLL (*ICS81001–21*) on the SDI HSMC board provides a 148.5 MHz or a 148.35 MHz clock that clocks the video outputs. The pll_config block on the Stratix IV GX FPGA sets the SDI_XTAL_SEL input to the PLL, which determines the choice of clock frequency. The PLL clock output passes through the HSMC connector to the reference design internal signal hsma_clk_in_p2. The reference design u_sdi_pll block then divides the hsma_clk_in_p2 signal and outputs clock frequencies (Table 3). The 27-MHz clock is available only if the hsma_clk_in_p2 frequency is 148.5 MHz. This functionality restricts the possible output frequency combinations in the case of two SDI video output channels. In this case, the same hsma_clk_in_p2 output clocks the video output streams from both of these channels.

Table 3 shows a list of the available output frequencies for sdi_tx_clk[0] and sdi_tx_clk[1].

Table 3. Available Output Frequencies

Input Frequency (hsma_clk_in_p2) (MHz)	Output Frequencies from u_sdi_pll (MHz)
148.35	148.35 or 74.175
148.5	148.5, 74.25, or 27

If the clock frequencies and formats are compatible, you can lock all of the video outputs to the frequency of the video input stream that the reference design receives on the SDI Input Channel 1 (J9) connector of the SDI HSMC card. The reference design locks the hsma_clk_in_p2 clock to the feedback clock that passes from the FPGA through the HSMC connector to the SDI HSMC card as the hsma_clk_out_p2 signal. The pll_config block sets the SDI HSMC card input pins SDI_CLK_V0 through SDI_CLK_V3 to configure the PLL to lock its output signal to the hsma_clk_out_p2 signal. The Clock Video Input MegaCore function on SDI input channel 1 divides down its vid_clk input signal and outputs the divided down clock as its refclk_div signal, sdi_rx_refclk_div[0], which is output from the FPGA as the hsma_clk_out_p2 signal. However, if the reference design cannot lock hsma_clk_in_p2 to hsma_clk_out_p2, the FPGA outputs a 27-MHz reference clock instead of the refclk_div signal.

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 or 148.35 MHz clock must drive the tx_serial_refclk input clock that is locked to the video output clock, sdi_tx_clk[0]. In the reference design, the hsma_clk_in_p2 signal is the source clock for the tx_serial_refclk input signals of the SDI MegaCore functions.

The u_sdi_pll block also drives the DVI and HDMI output video clocks. These clocks have their own clock multiplexers that the hdmi_mode_match and div_mode_match signals control, which select the correct output frequency. The DVI clocked video output block and the HDMI clocked video output block, respectively, drive the clock, sync, and data signals of the DVI and HDMI transmitter chips.

The DVI input clock, sync, and data signals from the DVI receiver chip drive the DVI clocked video input block directly.

Generator Lock

The *generator lock* can lock the timing of video outputs to a reference source. Sources locked to a single reference clock, allows 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 section, 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:

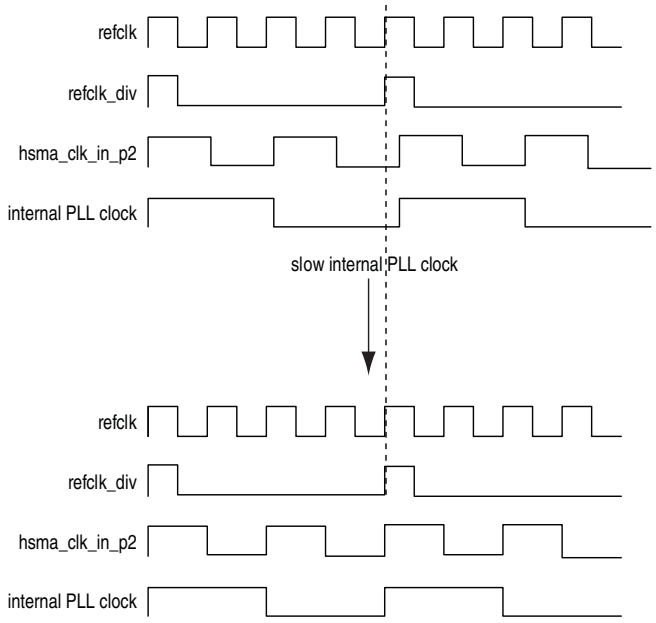
1. Clock locking—aligns the output video clock to the input video clock and tracking any changes.
2. Frame locking—aligns the start of frame (SOF) of the output video to the SOF of the input video.

Clock Locking

The reference design uses the VCXOs on the Stratix IV GX FPGA development board 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 any changes in `refclk_div` (Figure 4).

Figure 4. Clock Locking



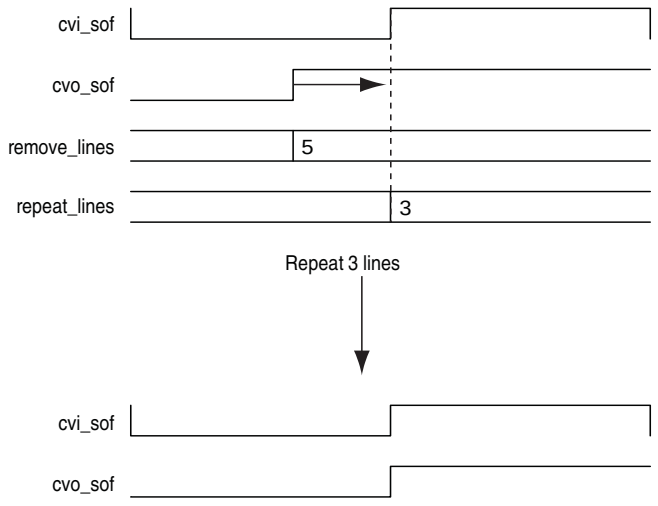
Frame Locking

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 5 shows an example of the SOF transitions.

Figure 5. Frame Locking



The Clocked Video Output MegaCore function compares the two SOF signals to determine how far apart they are. It 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 (64 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 (128 words in the GUI), the reference design can transfer a burst to and from the DDR3 SDRAM while it 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 datapath runs.

Bank interleaving further 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
- Five deinterlacer masters per video processing channel
- Two frame buffer masters per video processing channel
- One frame reader master for OSD logos

With two channels of video the system has 17 masters.

To achieve the optimal minimum memory access efficiency of 90%, a multiport front end controls access to the DDR3 SDRAM controller. The multiport front end arbitrates between the master accesses to the single slave port of the memory controller. The reference design uses the Qsys default round-robin arbitration scheme to reduce the number of ports on the multiport front end to 14. Figure 6 shows the block diagram of the memory subsystem, and indicates the arbitration priorities. Ports with lower numbers have higher priority.

Figure 6. Memory Subsystem Block Diagram

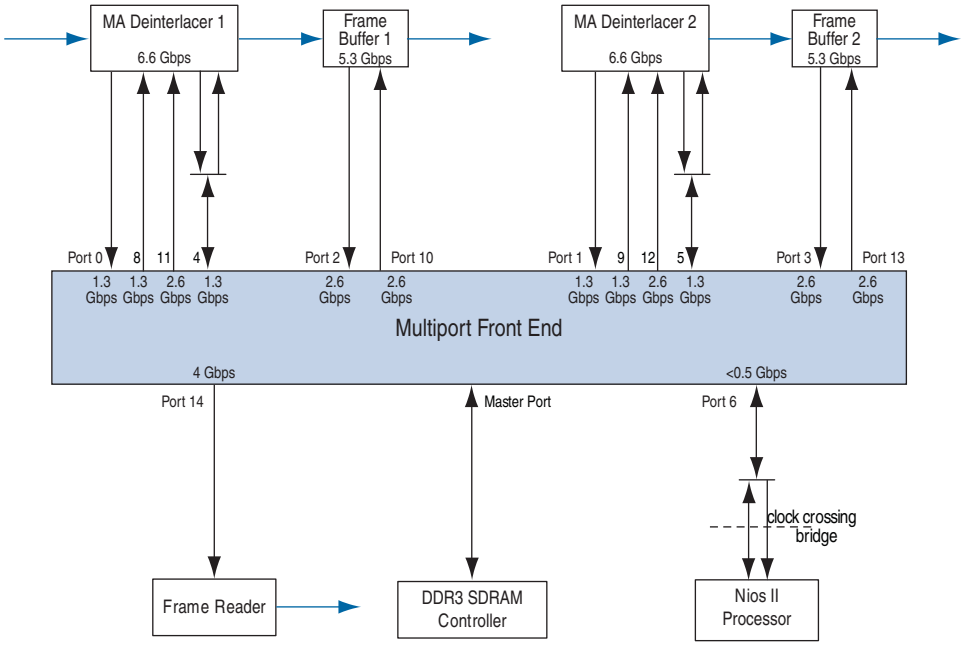


Table 4 describes the access patterns on each multiport front end Avalon-MM port, and shows the allocation of ports to different memory banks. the reference design allocates ports to different banks according to the base addresses in Qsys.

Table 4. Memory Subsystem Avalon-MM Ports (Part 1 of 2)

Port	Width (Bits)	Max Burst Size (Words)	Description	Memory Bank	Address Space
0	256	64	MA deinterlacer 1 write port	1	A
1	256	64	MA deinterlacer 2 write port	3	D
2	256	64	Frame buffer 1 write port	2	C
3	256	64	Frame bBuffer 2 write port	4	F
4	256	32	MA deinterlacer 1 motion write port motion read port	1	B

Table 4. Memory Subsystem Avalon-MM Ports (Part 2 of 2)

Port	Width (Bits)	Max Burst Size (Words)	Description	Memory Bank	Address Space
5	256	32	MA deinterlacer 2 motion write port motion read port	3	E
6	32	1	Nios II processor read and write ports	0	H
7	32	—	Unused	—	—
8	256	64	MA deinterlacer 1 read port 0	1	A
9	256	64	MA deinterlacer 2 read port 0	3	D
10	256	64	Frame buffer 1 read port	2	C
11	256	64	MA deinterlacer 1 read port 1	1	A
12	256	64	MA deinterlacer 2 read port 1	3	D
13	256	64	Frame buffer 2 read port	4	F
14	256	64	Frame reader read port	5	G

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 × 1080 × 21.33 bits × 60 fps = 2.654 Gbps

For output format 1080p60, the output rate is:

1920 × 1080 × 21.33 bits × 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

OSD to Frame Reader Bandwidth Calculation

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

1920 × 1080 × 32 bits × 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

Using the results from the preceding sections, Table 5 shows the calculations of the total bandwidth for the video paths only and for the video paths plus OSD and Nios II processor.

Table 5. Total System Bandwidth Calculation

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
Video path total	23.224
Video paths with OSD and the Nios II processor:	
Video path total	23.224

Table 5. Total System Bandwidth Calculation

Block	Bandwidth (Gbps)
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's 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 64-bit DDR3 SDRAM on the Stratix IV GX development board is:

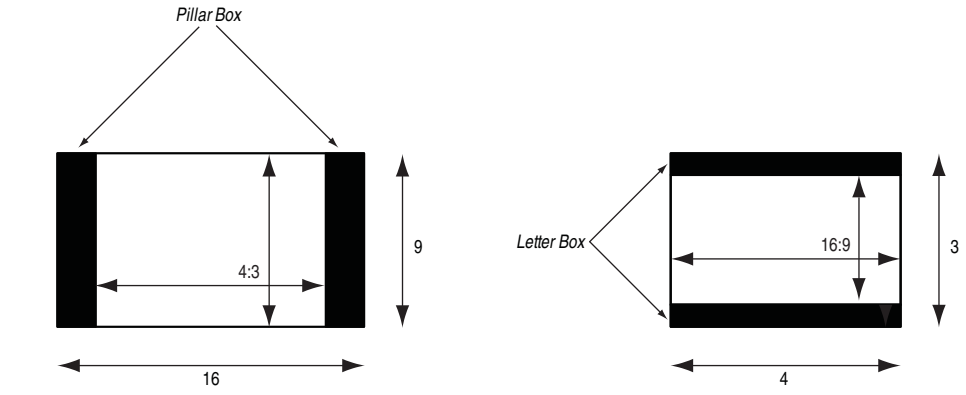
400 MHz × 64 bits × 2 (double rate: both clock edges used) = 51.2 Gbps

For the video-path only design, without the OSD input channel and with the Nios II processor code stored in a different memory, the DDR3 SDRAM access efficiency utilization is approximately 45%. Adding the OSD input channel and storing the Nios II software in the same DDR3 SDRAM gives a DDR3 SDRAM access efficiency utilization of approximately 54%.

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 that contain 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 7 shows two common aspect ratios and the unused areas when data originally in one format displays in the other format.

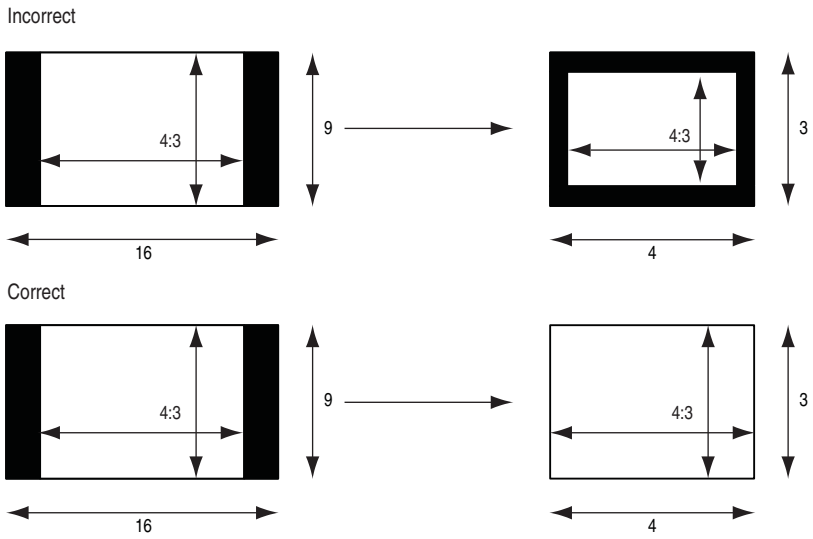
Figure 7. Example Aspect Ratios and Their Display



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

Figure 8. Incorrect and Correct AFD Handling



For more information on the ancillary packets, on 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** file. The comments for this function include a full list of AFD codes. Table 6 lists the supported codes, but does not show the unextended software ignored AFD codes.

Table 6. Supported 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, available at www.smpte.org.

System Memory Map

Before you run the reference design, you 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. The boot loader is located at address `0x22820000` (offset `0x2820000` 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** file. You can update the **.zip** file with your preferred OSD logos before you store it at offset `0x3020000` in the flash memory device.

The hardware image file that you store at offset `0xC20000` in the flash memory device configures the FPGA when the board powers up. To load the user-defined hardware image file, you must set the power monitor rotary switch (SW2) on the Stratix IV GX FPGA development board to 1.

Table 7 and Table 8 show the memory maps for the Nios II processor and the flash memory device.

Table 7. Nios II Processor Instruction Master Memory Map

Slave Device	Description	Nios II Processor Address Base or Range	Memory Bank
Flash memory	—	`0x20000000` to `0x23FFFFFF` (Refer to Table 8 for internal offsets)	—
DDR3 SDRAM	Unused	`0x1C000000`	7
	Frame buffer 1 base	`0x18000000`	6
	Unused	`0x10400000`	5
	Deinterlacer 1 base	`0x10000000`	4
	Deinterlacer 2 base	`0xC000000`	3
	Frame buffer 2 base	`0x8000000`	2
	Frame reader (Frame 2) base	`0x5000000`	1
	Frame reader (Frame 1) base	`0x4000000`	
	Nios II software code and data	`0x0` to `0x2000000`	0

Table 8 shows the flash memory map. The table lists the offset ranges of the different blocks in the flash memory device.

Table 8. Flash Memory Map (Part 1 of 2)

Memory Block Use	Size (KB)	Flash Memory Device Offset Range
Unused	128	0x03FE0000–0x03FFFFFF
User software image	24,320	0x02820000–0x03FDFFFF

Table 8. Flash Memory Map (Part 2 of 2)

Memory Block Use	Size (KB)	Flash Memory Device Offset Range
Factory software image	8,192	0x02020000–0x0281FFFF
.zip file system	8,192	0x01820000–0x0201FFFF
User hardware image	12,288	0x00C20000–0x0181FFFF
Factory hardware image	12,288	0x00020000–0x00C1FFFF
Parallel flash loader option bits	32	0x00018000–0x0001FFFF
Board information	32	0x00010000–0x0001FFFF
Ethernet option bits	32	0x00008000–0x0000FFFF
User design reset vector	32	0x00000000–0x00007FFF

Video Trace Module

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

For more information about the Avalon-ST Video protocol, refer to the *Video and Image Processing Suite User Guide*.

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.

UDX4 Control Panel

The UDX4 control panel demonstrates several features of the UDX4 reference design that are not available when using the push buttons on the board. You can enable these features by modifying the software code that controls the design from the Nios II processor. The control panel connects to the slave interfaces of multiple video and image processing IP cores in the system. You can use the control panel to monitor their status or to modify their behavior. For example, AFD code that the AFD extractor blocks recover located after the SDI inputs, or generate alpha-blending values and output resolution of the scalers.

Avalon-MM Performance Monitor and Multiport Front End Debugging Port

This topic discusses the System Monitor and the Avalon Trace Monitor.

System Monitor

The System Monitor application connects to the JTAG masters in the multiport front end and the Avalon-MM performance monitor components. When the reference design establishes a connection to the board, the PC hosting the debugging session polls the debugging registers of these two components at regular intervals and presents the statistics collected in a readable format or as graphical elements. The information from the multiport front end contains the percentage of utilization of each slave, the number of grants, average and worst latencies, and so on.

For an extensive description of the debugging information exposed through its debugging slave interface, refer to the *Multiport Front End User Guide*.

The information that the Avalon performance monitor returns contains general statistics on bus usage, efficiency, and latency.

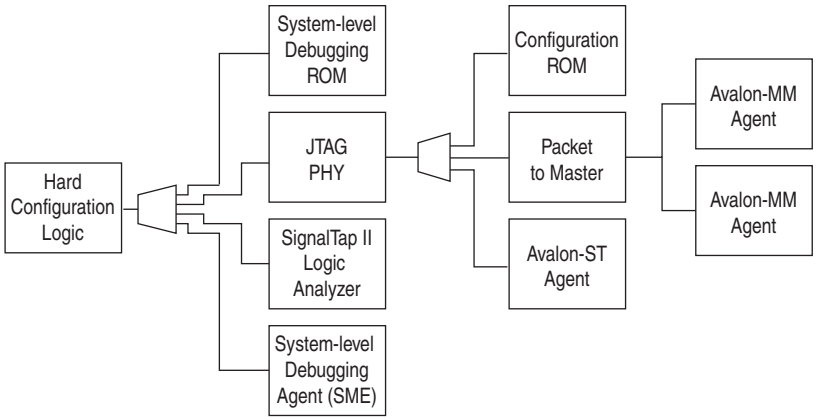
Avalon-MM Trace Monitor

The reference design parameterizes the Avalon-MM performance monitor to contain two JTAG masters. The first JTAG master collects general statistics on the Avalon-MM bus; you can use the second (optional) master for low-level debugging of the Avalon-MM interface. The Trace Monitor provides a time-stamped trace of the successive Avalon-MM transfer events on the bus between the multiport front end and the DDR SDRAM controller. This functionality is similar to the debugging features that the SignalTap® II logic analyzer offers. A trigger typically marks the start of a capture. The depth of the trace buffer set at compile time limits number of events captured. You can improve the capture span by taking advantage of the run-time filtering capabilities of the hardware agent.

Hardware Debugging Tools

JTAG connects the host PC to the hardware agents running on the board. A debugging hub (Figure 9) arbitrates accesses of the various agents to the JTAG PHY.

Figure 9. Debugging System on the Board



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

The design may instantiate and hide the components of a debugging system within Qsys components. For example, the JTAG master debug port of the multiport front end and the JTAG masters of the Avalon-MM performance monitor. Or the design may explicitly instantiate the components of a debugging system within Qsys. For example, the video trace module and the control panel.

The debugging fabric is in the **ip\directory**.

Running the Reference Design

To run the reference design, follow these steps:

1. Install the reference design files on your computer.
2. Connect the hardware.
3. Use the precompiled flash files.
4. Create and download a **.sof** file to configure the FPGA with a flash programmer target design. For this purpose, we use the **.sof** file that contains the hardware image of the reference design.
5. Program the flash memory on your board with the following three files:
   - The **.sof** file that contains the hardware image of the reference design
   - The Executable and Linking Format File (**.elf**) that contains the software image of the reference design
   - The OSD logos to mix, potentially, with incoming video data

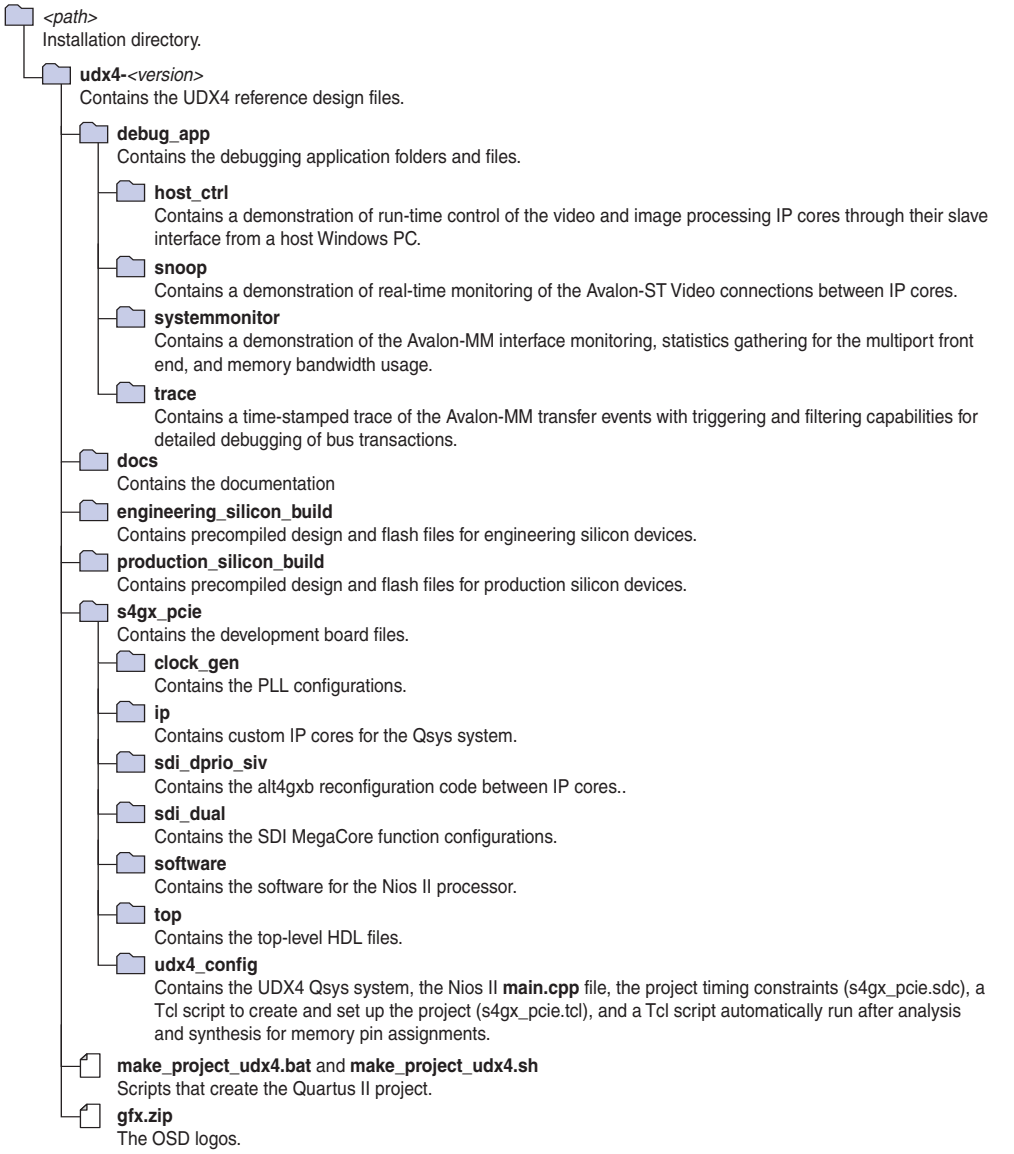
This section tells you how to perform these actions.

Installing the Reference Design

The **UDX4_**<*version*>**.zip** file 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, unzip the **UDX4**_<*version*>**.zip** file into a directory you designate for this project. Figure 10 shows the directory structure and top-level files for the UDX4 reference design files after you extract them from the .**zip** file.

Figure 10. Reference Design Directory Structure



Connecting the Hardware

To connect the hardware to run the UDX4 reference design, follow these steps:

1. Ensure that you turn off the Power Switch (SW1) on your Stratix IV GX FPGA development board.
2. Connect the SDI HSMC board to port A (J1) of the Stratix IV GX FPGA development board.

3. Connect the Bitec HSMC DVI card to port B (J1) of the Stratix IV GX FPGA development board.
4. Connect the one of the following first video sources to the hardware system:
   - Connect an SDI video source to the SDI Input Channel 1 connector (J9) on the SDI HSMC board.
   - Connect a DVI video source to the DVI-RX connector on the Bitec HSMC DVI card.

   If you connect both an SDI video source and a DVI video source, the DVI input takes precedence.
5. Connect the second video source to the SDI Input Channel 2 connector (J2) on the SDI HSMC board. This video source can be an SD, HD, or 3G video source.
6. Connect an HDMI monitor to the HDMI Video Port (J11) on the Stratix IV GX FPGA development board.

   To demonstrate the 1080p60 resolution capability of the design you require a 24-inch monitor that supports 1920x1080 pixels.
7. Connect an SDI monitor to the SDI Output Channel 1 connector (J8) on the SDI HSMC board.

   To demonstrate the 1080p60 resolution capability of the design you require a 24-inch monitor that supports 1920x1080 pixels.
8. Connect a second SDI monitor to the SDI Output Channel 2 connector (J1) on the SDI HSMC board.

   To demonstrate the 1080p60 resolution capability of the design you require a 24-inch monitor that supports 1920x1080 pixels.
9. Connect a DVI monitor to the DVI-TX connector on the Bitec HSMC DVI card.

   To demonstrate the 1080p60 resolution capability of the design you require a 24-inch monitor that supports 1920x1080 pixels.
10. Connect a USB-Blaster™ download cable to your computer and to the JTAG Connector (J8) on the Stratix IV GX FPGA development board.

    For information about how to install 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*.
11. Connect your Stratix IV GX FPGA development board to as power supply.

Using the Precompiled Flash Files for a Quick Start

The following sections describe how to compile the design and build a **.sof** programming file with the Quartus II software, how to build a Nios II software image and how to use the flash programmer to download the files to the board. However, you can directly download to the board the precompiled files included with the design, by following these steps:

1. Start a Nios II command shell:
   - 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**.
   - On Linux OS, type the following command:

   ```
   $SOPC_KIT_NIOS2/sdk_shell
   ```

   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 flash files for your board, by typing one of the following commands:

   ```
   <path to UDX4>\<UDX4 version>\engineering_silicon_build
   ```

   or

   ```
   <path to UDX4>\<UDX4 version>\production_silicon_build
   ```

3. Turn on the Power Switch (SW1) the Stratix IV GX FPGA development board and set the rotary switch (SW2) to position 1.
4. Program the board with the precompiled **.sof** file, by typing the following command:

   ```
   quartus_pgm -m jtag -o p\;s4gx_pcie.sof
   ```

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

   ```
   ./flash_program.sh
   ```

6. Turn off and then turn on again the Power Switch (SW1) to load and run the UDX4 reference design.

The design starts up in independent mode. The SDI input channel 1 (or DVI-RX) is output from the HDMI output. The SDI input channel 2 is output from the DVI-TX.

Compiling the Quartus II Project

To compile the Quartus II project, follow these steps:

1. To create the UDX4 reference design Quartus II Project File (**.qpf**), **s4gx_pcie.qpf**, and set up the design, perform one of the following actions:
   - In a Windows OS, double-click the script **make_project_UDX4.bat**.
   - On the Linux command line, type the following command:

     ```
     sh make_project_UDX4.sh ↵
     ```

2. Open the Quartus II project file **s4gx_pcie.qpf**.

3. On the Assignments menu, click **Device**. The **Device Settings** dialog box appears.
4. Under **Available devices**, select **EP4SGX230KF40C3**.
5. Click **OK**.
6. In the Quartus II software, on the Tools menu, click **Qsys**.
7. Review the UDX4 reference design Qsys system. Figure 2 on page 6 shows the block diagram of the Qsys system.
8. In Qsys, click the **Generation** tab.
9. Turn off **Simulation.** System generation executes much faster when simulation is off.
10. Click **Generate**. System generation may take a few minutes.
11. When the **System generation was successful** message displays, click **Exit**.
12. If you are prompted to save your changes, click **Save**.
13. In the Quartus II software, on the Processing menu, click **Start Compilation**. Compilation creates the **s4gx_pcie.sof** file.
14. After compilation completes, on the Tools menu, click **Programmer**.
15. In the Quartus II Programmer, under **Mode**, verify that **JTAG** is selected.
16. Click **Hardware Setup** to configure the programming hardware. The **Hardware Setup** dialog box appears.
17. In the **Hardware** column, double click **USB-Blaster**.
18. Click **Close** to exit the **Hardware Setup** window.
19. In the Quartus Programmer, click **Add File**, navigate to the **s4gx_pcie.sof** file you created in step 13, and click **Open**.
20. Ensure the **Program/Configure** box for the new file contains a checkmark.
21. Click **Start**. The software downloads the **.sof** file to your Stratix IV GX FPGA development board and configures the FPGA with the hardware image.

For information about Qsys, refer to the *System Design with Qsys* section in volume 1 of the *Quartus II Handbook*.

For information about the Quartus II Programmer, refer to the *Quartus II Programmer* chapter in volume 3 of the *Quartus II Handbook*.

Compiling the UDX4 Reference Design Software

In this section, you compile the UDX4 reference design software using the Nios II Software Build Tools (SBT) for Eclipse™.

To compile and download the UDX4 reference design software in the Nios II SBT for Eclipse, follow these steps:

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

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 **s4gx_pcie** directory and create a new workspace folder.
3. Right-click in 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 **s4gx_pcie/UDX4.sopcinfo**.
6. For **Project name**, type `s4gx_pcie_controller`.
7. In the **Templates** list, select **Blank Project**.
8. Click **Finish**. The Nios II SBT for Eclipse creates your project and adds it to the **Project Explorer** tab.
9. After the project is created, in the **Project Explorer** tab, right-click **s4gx_pcie_controller_bsp**, and on the Nios II menu, click **BSP Editor**.
10. In the **Linker Script** tab, for the linker region **uniphy_ddr3**, 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.
11. In the **Software Packages** tab, enable the **altera_ro_zipfs** software package and set its parameter values (Table 9).

Table 9. altera_ro_zipfs Software Package Settings

Parameter	Value
`ro_zipfs_base`	0x20000000
`ro_zipfs_name`	`/mnt/gfx`
`ro_zipfs_offset`	0x3020000

12. Click **Generate**.

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

13. On the File menu, click **Exit**. The BSP Editor closes.
14. In the Nios II SBT for Eclipse, in the **Project Explorer** tab, right-click **s4gx_pcie_controller**, and on the Run As menu, click **2 Nios II Hardware**. The project compiles and the software downloads the resulting **.elf** file to your Stratix IV GX FPGA development board.

If a panel appears to inform you that there is 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** file 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**.

For information about the Nios II SBT for Eclipse, refer to the *Getting Started with the Graphical User Interface* and *Nios II Software Build Tools* chapters of the *Nios II Software Developer's Handbook*.

Changing the OSD Logos

The **gfx.zip** file holds the OSD logos in 8-bit raw RGBA format. To replace OSD logos in the UDX4 reference design, you must format and add the new logos before you program the **gfx.zip** file to flash memory. This section teaches you how to format and add OSD logos in this file.

If you do not want to add new OSD logos to the UDX4 reference design, skip this section and continue with "Programming Flash Memory" on page 34.

The instructions in this section tell you how to convert a new OSD logo to the correct format using the free software Gimp from www.gimp.org. Before you perform the instructions, you must install the Gimp software on your computer.

The Gimp software instructions in this section 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** file, 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 file **s4gx_pcie/software/s4gx_pcie/controller/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 **s4gx_pcie/software/s4gx_pcie/controller/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
```

Programming Flash Memory

In this section, you use the Nios II Flash Programmer to program the UDX4 reference design in flash memory on the Stratix IV FPGA development board, 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 33 before following the instructions in this section.

To program the UDX4 reference design **.sof** file, **.elf** file, and OSD logos to flash memory, follow these steps:

1. In the Nios II SBT for Eclipse, in the **Project Explorer** tab, right-click **s4gx_pcie_controller_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 **s4gx_pcie/software/s4gx_pcie_controller_bsp/settings.bsp** file.
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 **s4gx_pcie/software/s4gx_pcie_controller/s4gx_pcie_controller.elf** file.
11. Click **Select**.
12. Under **File generation command**, click **Properties**.
13. Table 10 shows the values to set in the **Properties** dialog box.

Table 10. File Generation Command Property Settings for s4gx_pcie_controller.elf

Property	Value
CPU reset address	0x22820000
Flash base address	0x20000000
Flash end address	0x24000000

14. Click **Close**.
15. Under **Files for flash conversion**, click **Add**.
16. Browse to locate the **s4gx_pcie/s4gx_pcie.sof** file.
17. Click **Select**.
18. Double-click the **s2gx_pcie.sof** entry row in the **Flash Offset** column and type the value 0xC20000.
19. Click **Add**.
20. Click **Properties**.
21. Add the following additional arguments and close the **Properties** window:

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

22. Browse to locate the **s4gx_pcie/gfx.zip** file.
23. Click **Select**.
24. Double-click the **gfx.zip** entry row in the **Flash Offset** column and type the value 0x3020000.
25. Click **Start** to program all of the files in the flash memory device.

CAUTION

Boards with dual-die common flash interface (CFI) device (sold until June 2010) may need a flash override. Copy the **nios2-flash-override-s4gx_pcie.cfi_override** file from the **production_silicon_build** directory into the <*Nios II EDS install path*>**/bin** directory.

For information about the Nios II Flash Programmer, refer to the *Nios II Flash Programmer User Guide*.

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

1. Set the power monitor rotary switch (SW2) on the Stratix IV GX FPGA development board to 1.

2. Power cycle the board to load and run the UDX4 reference design.

Status Displays

The LCD screen displays messages that describe the state of the software initialization and start up. Table 11 describes these messages.

Table 11. LCD Status Messages

Message	Description
Init UDX4.<*version*>...	The software sends this message to the LCD as soon as it starts to run.
GFX load failed	The software has failed to load the logos from the **gfx.zip** file in flash memory. In this case, the software exits.
GFX load passed	The software has successfully loaded the logos from the **gfx.zip** file 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 Stratix IV GX FPGA development board LEDs display the current running status of the system. Table 12 and Table 13 list the meanings of the different LED settings. Refer to Figure 1 on page 4 for the locations of the LEDs on the Stratix IV GX FPGA development board.

Table 12. LED Status Information

LED	Status
0, 1, 2	These three LEDs indicate the current mode. Refer to Table 13 for details.
7	Heartbeat. This LED flashes when the software is running.
8	Indicates channel 1 is active.
9	Indicates channel 2 is active.
10	Indicates the HDMI output data is genlocked to SDI Input Channel 1.
11	Indicates the SDI output channel 1 data is genlocked to SDI input channel 1.
12	Indicates that the AFD clipper of input channel 1 is set for "letter box" or "pillar box" cropping.
13	Indicates that the AFD clipper of input channel 2is set for "letter box" or "pillar box" cropping.
14	Indicates one of the clocked video output channels has underflowed.
15	Indicates one of the clocked video input channels has overflowed.

Table 13 shows the correlation between the settings of LEDs 0, 1, and 2 and the current mode.

Table 13. LED Display of UDX4 Reference Design Mode

LED			Meaning
5	6	7	
Off	Off	Off	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 HDMI video port (J11) on the Stratix IV GX FPGA development board. Channel 2 output data is sent to the DVI-TX connector on the Bitec HSMC DVI card.
On	Off	Off	Multi-view mode: channel 1 multi-view, channel 2 test pattern. Channel 1 output data is sent to the HDMI video port (J11) on the Stratix IV GX FPGA development board. Channel 2 output data is sent to the SDI Output Channel 2 connector (J1) on the SDI HSMC board.
Off	On	Off	Multi-view mode: channel 1 multi-view, channel 2 test pattern. Channel 1 output data is sent to the SDI Output Channel 1 connector (J8) on the SDI HSMC board. Channel 2 output data is sent to the SDI Output Channel 2 connector (J1) on the SDI HSMC board.
On	On	Off	Independent mode. Channel 1 output data is sent to the SDI Output Channel 1 connector (J8) on the SDI HSMC board. Channel 2 output data is sent to the SDI Output Channel 2 connector (J1) on the SDI HSMC board.

User Controls

While the software is running, you can control the run-time configurable parameters using the Stratix IV GX FPGA development board push-button switches. Table 14 shows the functions of the individual push-button switches. Refer to Figure 1 on page 4 for the locations of the push-button switches on the Stratix IV GX FPGA development board.

Table 14. Push-Button Switch Controls (Part 1 of 2)

Push-Button Switch	Function
S5 (PB0)	Cycles through the four modes (Table 13). You must synchronize operations that involve the mixer blocks carefully before switching between independent mode and multi-view mode.
S4 (PB1)	Changes the output resolution of channel 1: ■ HDMI output cycles through 720p60, 480p60, and 1080p60. ■ SDI output cycles through 720p60, NTSC, and 1080i60.

Table 14. Push-Button Switch Controls (Part 2 of 2)

Push-Button Switch	Function
S3 (PB2)	Changes the output resolution of channel 2: ■ DVI output cycles through 720p60, 480p60, and 1080p60. ■ SDI output cycles through 720p60, NTSC, and 1080i60.
S2 (CPU Reset Push-Button Switch)	Resets the UDX4 reference design by restarting the software.

Using the Debugging Applications (Windows Only)

To use the debugging applications, follow these steps:

1. Browse to the directory of the control panel application, <*installation directory*>**\UDX4\debug_app\host_ctrl**:
   a. Open the **host_ctrl.bat** file in a text editor.
   b. Follow the instructions in the **.bat** file to edit it to match your system environment and parameterize it for your design.
   c. Save the **.bat** file and close the text editor.
2. Ensure that the design runs correctly and switch into multi-view mode.
3. Double-click the **.bat** file to start the control panel appplication.
4. Try the various controls. Be aware that the debugging application does not protect user inputs and that many actions may cause the design to crash.

   ☞ You may reset the design at any point but you must also restart the control panel application.

5. Close the control panel application and browse to video trace directory, <*installation directory*>**\UDX4\debug_app\snoop**.
6. Ensure that the system still runs correctly and double-click the **snoop.bat** file to launch the video trace module.

   ☞ By default, the video trace module GUI monitors two connections in the first video input channel (before the deinterlacer and after the scaler).

7. Close the video trace module and browse to **systemmonitor** directory <*installation directory*>**\UDX4\debug_app\systemmonitor**.
8. Open the **system_monitor.bat** file in a text editor.
9. Follow the instructions in the .bat file to edit it to match your system environment and parameterize it for your design.
10. Save the **system_monitor.bat** file and close the text editor.
11. Double click the **system_monitor.bat** file to start the System Monitor application.
12. Browse the tabs to visualize the debug information received from the multiport front end and the Avalon-MM performance monitor.

13. Close System Monitor.

Document Revision History

Table 15 shows the revision history for this document.

Table 15. Document Revision History

Date	Version	Changes
June 2011	1.1	Added the *Using the Precompiled Flash Files for a Quick Start* section.
February 2011	1.0	Initial release.