AN 604: High Definition Video Reference Design (UDX3)

© March 2010 AN-604-1.0

Introduction

The Altera® video series of reference designs deliver high-quality up, down, and cross conversion of standard definition (SD), high definition (HD), and 3 gigabits per second (Gbps) video streams in interlaced or progressive format. The reference designs are highly software and hardware configurable, enabling rapid system configuration and design. The designs have been developed targeting typical broadcast applications such as switcher, multi-viewer, 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 contact the Altera Broadcast Business Unit.

The Up/Down/Cross Conversion (UDX3) reference design combines the functionality of the Altera high-definition video reference designs described in the application notes *AN559: High Definition Video Reference Design (V1)* and *AN581: High Definition Video Reference Design (V2)*. It features two input channels and two output channels, all with flexible, run-time selectable formats, and two high-quality up/down/cross (UDX) conversion video processing paths, as well as Active Format Description (AFD) extraction and insertion. The UDX3 reference design is software-configurable to support multi-view (with optional on-screen display (OSD)) or UDX conversion, and the two fully independent, high-quality video channels both handle resolutions up to 1080p60.

The UDX3 reference design uses the Altera Video and Image Processing Suite MegaCore® functions library, the SDI MegaCore function, the DDR3 High Performance Memory Controller MegaCore function, the Nios® II processor, and supporting development tools.

For information about these MegaCore functions, refer to the *Video and Image Processing Suite User Guide*, the *SDI MegaCore function User Guide*, the *DDR3 SDRAM High-Performance Controller and ALTMEMPHY IP User Guide* section in volume 3 of the External Memory Interfaces Handbook, and the Nios II Processor handbooks on the Altera website.

Features

Table 1 lists the key features of the UDX3 reference design.

Table 1. Key Features of the UDX3 Reference Design (Part 1 of 2)

Feature	**Description**
Input	Two fully independent input streams in any of the following run-time selectable formats: ■ Two SD-SDI, HD-SDI, or 3G-SDI progressive or interlaced input streams up to 1080p60 (such as: NTSC, PAL, 720p, 1080i, and 1080p). All resolution and frame rate combinations up to 1080p60 are supported. ■ One DVI progressive input stream. All resolution and frame rate combinations up to 1080p60 are supported.
Output	Two fully independent output streams in any of the following run-time selectable formats: ■ Two SD-SDI, HD-SDI, or 3G-SDI progressive or interlaced output streams up to 1080p60 (such as: NTSC, PAL, 720p, 1080i, and 1080p). All resolution and frame rate combinations up to 1080p60 are supported. ■ One HDMI interlaced or progressive output stream. All resolution and frame rate combinations up to 1080p60 are supported. ■ One DVI progressive output stream. All resolution and frame rate combinations up to 1080p60 are supported.
Processing	Two fully independent high-quality UDX conversion video processing paths. Format conversion functions include clipping, deinterlacing, scaling and frame rate conversion. ■ Mixer on the first video processing path supports multi-view with OSD. Design includes a dedicated Frame Reader MegaCore function to support streaming of OSD from a frame buffer in memory. Both paths support the following features: ■ High quality processing (motion adaptive deinterlacer, 6×6 tap scaler) ■ Two-field latency for interlaced input stream and two-frame latency for progressive input stream (one frame of latency in the deinterlacer and one frame of latency in the frame buffer) ■ Run-time controllable frame rate conversion in the frame buffer function between input and output to support any input or output frame rate ■ Run-time controllable interlacing in the Interlacer MegaCore function to support interlaced output ■ Generator lock (genlock) ■ Active Format Description (AFD) extraction and insertion. Dynamic clipping, scaling, and padding to support bidirectional format conversion between 4:3 and 16:9 aspect ratios based on an AFD code.
Improvements on Altera V1 and V2 reference designs	Unified memory architecture: a single multi-port front end allows processor, OSD Frame Reader MegaCore function, and video processing functions to use a single 64-bit DDR3 memory. Switch MegaCore function enables run-time reconfiguration of the video processing paths.
Software	System initialization and run-time configuration in software. Run-time configuration of input and output formats, resolution, or frame rate. Class application programming interface (API) provided to facilitate read/write access to the Video and Image Processing Suite register maps at run time. Software loads from flash memory together with SRAM Object File (**.sof**) and OSD logos.
System Tools	Rapid system capture and design with SOPC Builder, Quartus® II, and the Nios II development environments.

Table 1. Key Features of the UDX3 Reference Design (Part 2 of 2)

Feature	Description
Design Framework	Highly parameterizable and modular hardware functions (7 Video and Image Processing Suite MegaCore functions, Nios II processor, open source system-configuration software, video frame buffers, high performance memory controllers, peripherals, auto-generation of switch fabric). Standard Avalon-ST and Avalon-MM interfaces for rapid integration, and Avalon-ST Video protocol for video transmission between functions.
Hardware	The UDX3 reference design runs on the Altera Audio Video Development Kit, Stratix IV GX Edition (or alternatively, on the Altera Stratix IV GX FPGA Development Kit with a Terasic Transceiver SDI High-Speed Mezzanine Card (HSMC) board) and a Bitec HSMC DVD interface card.

System Requirements

This section describes the hardware and software requirements to run the UDX3 reference design.

Hardware Requirements

The UDX3 reference design requires the following hardware components:

- Stratix IV GX FPGA Development Board
- SDI HSMC board

  ☞ 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.

- Bitec HSMC DVI interface card

  ☞ To acquire the Bitec HSMC DVI interface card, refer to the the Bitec website.

- Any combination of the following video sources:
  - As many as two SDI video sources in SD, HD, or 3G format, that provide progressive or interlaced output up to 1080p60 resolution and frame rate, with BNC connectors to connect to the SDI HSMC board
  - As many as one DVI video source providing progressive output up to 1080p60 resolution and frame rate

- Any combination of the following video equipment for display:
  - As many as two displays with a DVI or HDMI interface supporting 1920 × 1080 pixel resolution
  - As many as two SDI monitors capable of supporting 1920 × 1080 pixel resolution

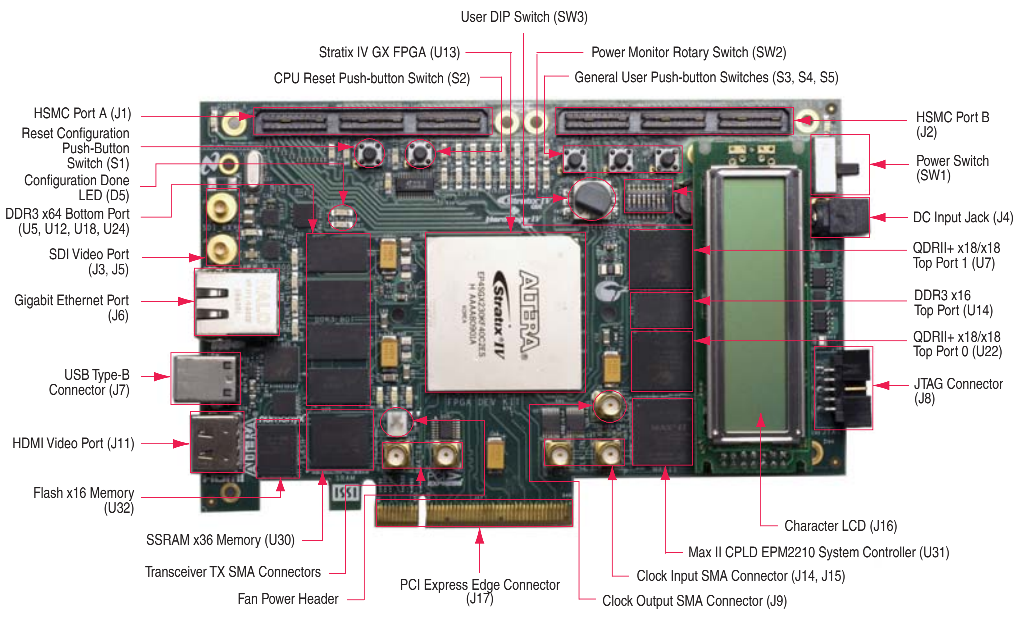
    or

    as many as two 3G SDI-to-DVI converter boxes and as many as two monitors with DVI interfaces supporting 1920 × 1080 pixel resolution

  Each SDI monitor must be able to demonstrate the 1080p60 capability of the UDX3 reference design. Most 24-inch monitors meet this requirement, but smaller monitors may also meet the requirement.

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

Figure 1. Stratix IV GX FPGA Development Board


User DIP Switch (SW3)
Stratix IV GX FPGA (U13)
CPU Reset Push-button Switch (S2)
Power Monitor Rotary Switch (SW2)
General User Push-button Switches (S3, S4, S5)
HSMC Port A (J1)
Reset Configuration Push-Button Switch (S1)
Configuration Done LED (D5)
DDR3 x64 Bottom Port (U5, U12, U18, U24)
SDI Video Port (J3, J5)
Gigabit Ethernet Port (J6)
USB Type-B Connector (J7)
HDMI Video Port (J11)
Flash x16 Memory (U32)
HSMC Port B (J2)
Power Switch (SW1)
DC Input Jack (J4)
QDRII+ x18/x18 Top Port 1 (U7)
DDR3 x16 Top Port (U14)
QDRII+ x18/x18 Top Port 0 (U22)
JTAG Connector (J8)
Character LCD (J16)
Max II CPLD EPM2210 System Controller (U31)
Clock Input SMA Connector (J14, J15)
Clock Output SMA Connector (J9)
PCI Express Edge Connector (J17)
Fan Power Header
Transceiver TX SMA Connectors
SSRAM x36 Memory (U30)

Software Requirements

Table 2 shows the operating systems and software tool versions that are supported by the UDX3 reference design.

Table 2. Operating Systems and Software Tool Versions supported by the UDX3 Reference Design

Operating System	Altera Software Tools	Altera Development Kit
Windows XP or Linux	v9.1 SP1 *(1)*	Stratix IV GX FPGA Development Kit or Audio Video Development Kit, Stratix IV GX Edition

Note to Table 2:

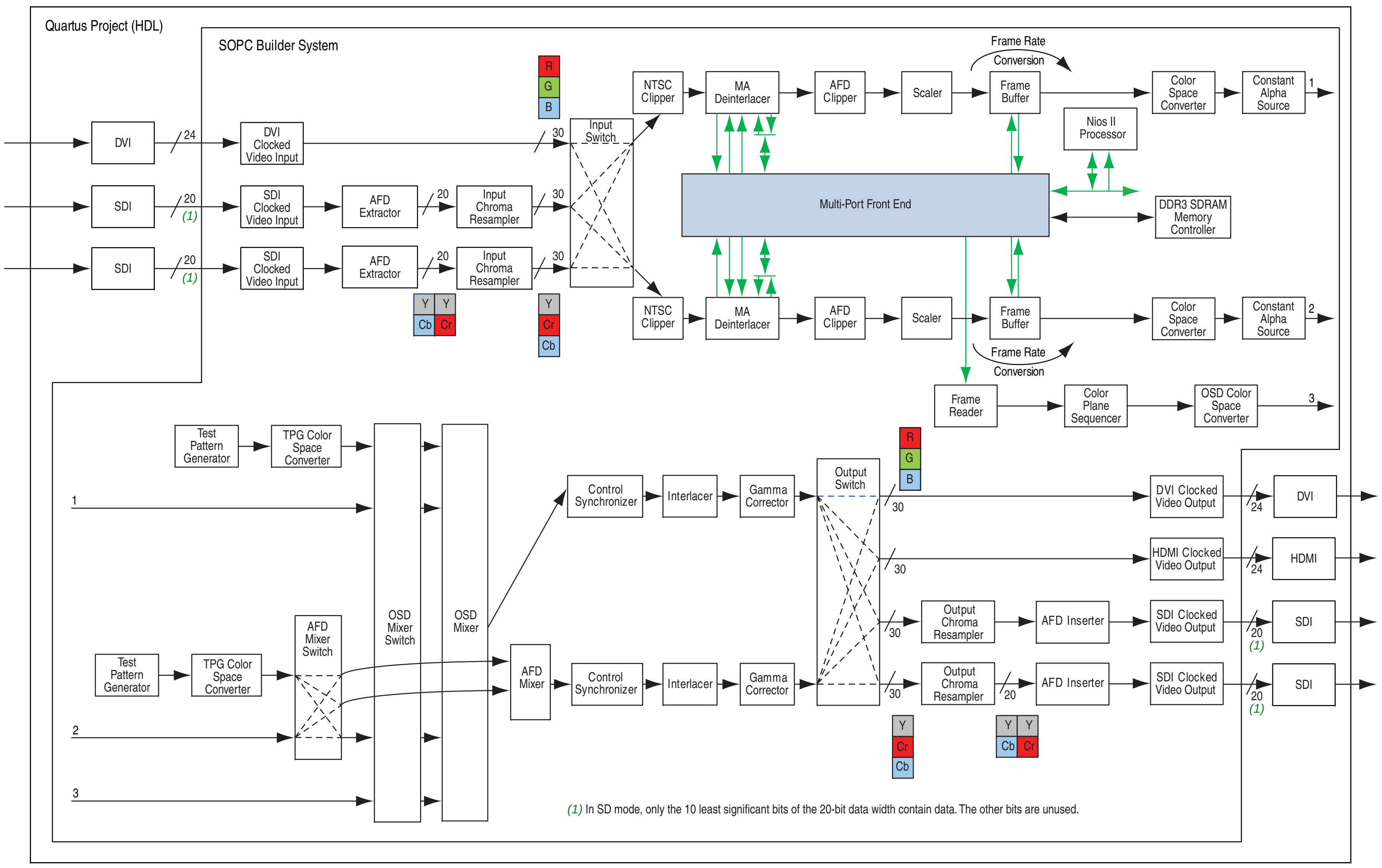
(1) The Altera software tools include the Quartus II software, SOPC Builder, Nios II EDS, and MegaCore IP Library (including the Video and Image Processing Suite).

For more information about the Altera development kit software installation, refer to the documentation provided with the development kit.

Functional Description

Figure 2 shows a detailed block diagram of the UDX3 reference design.

Figure 2. UDX3 Reference Design Block Diagram



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

The UDX3 reference design comprises two video processing paths. This section provides a functional description of the blocks in both paths.

UDX3 Reference Design Blocks

Most of the blocks in the UDX3 reference design are instances of Video and Image Processing (VIP) Suite MegaCore functions. The VIP MegaCore functions are runtime-configurable using the register map through the Avalon-MM interface. Software executing on the Nios II processor performs the configuration changes.

All of the MegaCore functions are configured with a maximum resolution of 1920 × 1080 pixels. Table 3 describes the configuration choices for the blocks.

Table 3. UDX3 Reference Design Blocks (Part 1 of 3)

Reference Design Block	**MegaCore Function**	**Configuration Details and Functionality**
DVI	Not a MegaCore function	DVI receiver chip on the Bitec DVI HSMC card
DVI Clocked Video Input	Clocked Video Input (Beta)	Shifts 8-bit incoming DVI data to eight MSBs of the ten bits, with 2 LSBs tied to 0; three colors in parallel and separate sync information
SDI (×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 UDX3 reference design
SDI Clocked Video Input (×2)	Clocked Video Input (Beta)	Inputs 10-bit SD SDI and 20-bit HD SDI with embedded sync information; extracts embedded ancillary data packets from vertical blanking region and inserts them in Avalon-ST Video ancillary packets (type 0xD)
AFD Extractor (×2)	AFD Extractor (Beta)	Extracts AFD code (0–15) from Avalon-ST Video ancillary packets; supports access to the AFD code through an Avalon-MM slave control interface
Input Chroma Resampler (×2)	Chroma Resampler	Performs 4:2:2 to 4:4:4 conversion; this conversion is not runtime-configurable
Input Switch	Switch	Connects three Avalon-ST Video inputs to two Avalon-ST Video outputs; allows two of the three input streams to be selected for input to the two video processing channels
NTSC Clipper (×2)	Clipper	Clips input images to a runtime-configurable clipping window based on the resolution detected by the Clocked Video Input (CVI) MegaCore function; clips interlaced fields with different numbers of lines *(1)*; for example, NTSC with F0=244 lines and F1=243 lines
MA Deinterlacer (×2)	Deinterlacer	Deinterlaces video input using motion-adaptive method with motion bleed enabled, 4:4:4 mode, double-buffering in external RAM, one output frame for each input field, runtime-configurable locked frame-rate conversion
AFD Clipper (×2)	Clipper	Clips input images based on the detected AFD code
Scaler (×2)	Scaler	Scales images in polyphase mode with 6×6 taps, runtime-configurable output resolution and runtime-configurable filter coefficients
Frame Buffer (×2)	Frame Buffer	Provides runtime-configurable double-buffering, or triple-buffering to support simple frame rate conversion (dropping and repeating of frames)

Table 3. UDX3 Reference Design Blocks (Part 2 of 3)

Reference Design Block	**MegaCore Function**	**Configuration Details and Functionality**
Color Space Converter (×2)	Color Space Converter	Converts 10-bit red, green, blue (RGB) to 10-bit YCbCr or 10-bit YCbCr to 10-bit RGB, depending on the format of the output video; applied to convert the two video streams to a common color space prior to mixing
Constant Alpha Source (×2)	Chroma Key (Beta)	Outputs a constant alpha value synchronized to the video stream; runtime-configurable to alter the alpha value on a frame-by-frame basis; Chroma Key Beta MetaCore function provides additional modes not used in the UDX3 reference design
Nios II Processor	Nios II Processor	Embedded Nios II/f processor with a Level 1 JTAG debug module and an internal interrupt controller
Multi-Port Front End	Multi-Port Front End (Beta)	Provides efficient access from multiple masters to the single external memory; refer to "Memory Subsystem Architecture" on page 18 and to **MPFE_user_guide.doc** in the UDX3 reference design installation.
DDR3 HP2 Memory Controller	DDR3 SDRAM High-Performance Controller (using ALTMEMPHY)	Provides DDR3 SDRAM memory access with 64-bit wide, 400-MHz operations to 512 Mbytes of memory split into eight banks; runs at half-rate with a 256-bit 200-MHz bus
Frame Reader	Frame Reader	Provides Avalon-MM to Avalon-ST Video bridge; runtime-configurable memory read address; performs frame repeating to convert between slow frames/second rate of the OSD updates and faster frames/second rate of the video output
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
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
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 runtime-configurable
TPG Color Space Converter	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
AFD Mixer Switch	Switch	Connects two Avalon-ST Video inputs to three Avalon-ST Video outputs; in multi-view mode, sends the video input to the OSD Mixer block; in independent mode, sends the video input and the test pattern to the AFD Mixer, or bypasses the AFD Mixer if AFD correction is not required (such as for 1080i to 1080i); to bypass the AFD Mixer, sends the video input to layer 0 and disables the test pattern
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
OSD Mixer	Alpha Blending Mixer	Alpha input enabled; four layers
AFD Mixer	Alpha Blending Mixer	Alpha input disabled; two layers
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

Table 3. UDX3 Reference Design Blocks (Part 3 of 3)

Reference Design Block	MegaCore Function	Configuration Details and Functionality
Interlacer (×2)	Interlacer (Beta)	Optionally changes progressive video stream to interlaced output; passes interlaced input stream through unchanged; configured for interlaced output (drops half the lines of each input frame), output field rate equals input frame rate, and control port to allow runtime configuration of pass-through mode for progressive video
Gamma Corrector	Gamma Corrector	Converts 10-bit color values to 8-bit color values for DVI and HDMI output; conversion is runtime-configurable
Output Switch	Switch	Connects 2 Avalon-ST Video inputs to 4 Avalon-ST Video outputs; allows two output streams to be selected for output from the video processing channels
Output Chroma Resampler (×2)	Chroma Resampler	Performs 4:4:4 to 4:2:2 conversion; this conversion is not runtime-configurable
AFD Inserter (×2)	AFD Inserter (Beta)	Inserts the AFD code (0–15) it receives on its Avalon-MM slave interface into an Avalon-ST Video ancillary packet (type 0xD)
DVI Clocked Video Output	Clocked Video Output (Beta)	Inputs 10-bit video formatted with three colors in parallel and separate sync information, and outputs 8-bit DVI data which is the 8 MSBs of the incoming 10 bits; the Gamma Converter upstream converts the 10-bit color values to 8-bit color values, so removing the two bits does not lose information
HDMI Clocked Video Output	Clocked Video Output (Beta)	Inputs 10-bit video formatted with three colors in parallel and separate sync information, and outputs 8-bit HDMI data which is the 8 MSBs of the incoming 10 bits; the Gamma Converter upstream converts the 10-bit color values to 8-bit color values, so removing the two bits does not lose information
SDI Clocked Video Output (×2)	Clocked Video Output (Beta)	Outputs 10-bit SD SDI and 20-bit HD SDI with embedded sync information
Outgoing DVI	Not a MegaCore function	External DVI transmitter chip on the Bitec DVI HSMC card
HDMI	Not a MegaCore function	External HDMI transmitter chip on the Stratix IV GX FPGA development board

Note to Table 3:

(1) The Clipper MegaCore function can handle fields of different heights, but the Deinterlacer MegaCore function cannot do so.

Most of the MegaCore functions listed in Table 3 are in the Video and Image Processing Suite. For more information about the non-Beta MegaCore functions, refer to the *Video and Image Processing Suite User Guide*.

For a full description of the Avalon-ST Video protocol and Avalon interfaces, refer to the "Interfaces" section of 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*.

SDI MegaCore Function

The SDI MegaCore function is configured as a triple-rate receiver serial digital interface. Two instances are required, one for each video processing channel. The SDI input clock frequency is 148.5 or 148.35 MHz for 3Gbps (3G-SDI) or 74.25 or 74.175 MHz for 1.5Gbps (HD-SDI) video inputs, or 27.0 MHz for SD-SDI video inputs. In this design, a clock frequency of 148.5 MHz 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 runtime user input:

- 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 15 on page 36).
- LEDs to display system status (refer to Table 16 and Table 17 on page 37).
- Push buttons configured to allow switching between video output resolutions (refer to "User Controls" on page 37).

Runtime Configuration

Input resolution on each video processing channel is determined from the input video stream. You can use the push-button switches on the Stratix IV GX FPGA development board to modify the UDX3 reference design video processing paths 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 stable video is detected, the Clocked Video Input MegaCore function generates control packets containing the new resolution and progressive/interlaced format. The control packets propagate through each function in the video processing path, configuring each MegaCore function 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

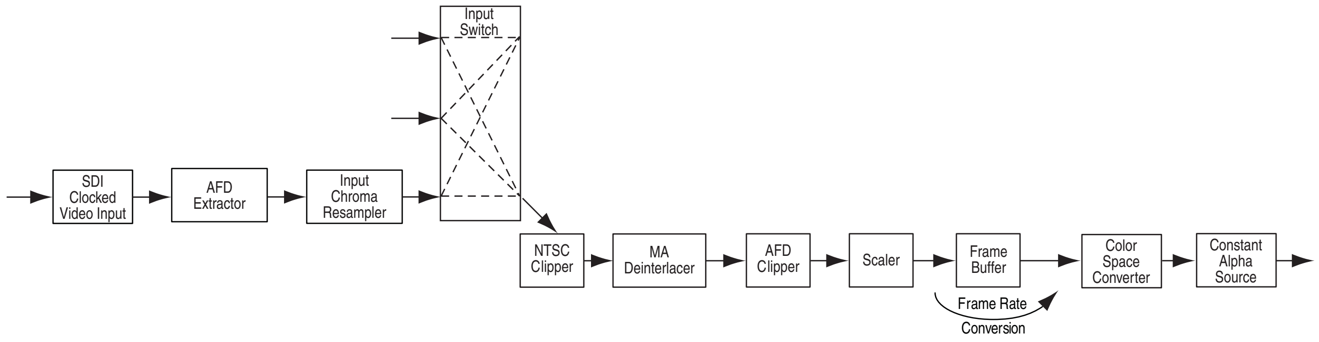
You can use the general purpose push-button switches on the Stratix IV GX FPGA development board to change the video processing channel connectivity and the resolution on the output displays. For details, refer to Table 18 on page 37.

Control Software

For control purposes, each video processing path in the UDX3 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.

A video input channel starts at a clocked video input block and extends to the Constant Alpha Source block. Figure 3 shows a video input channel.

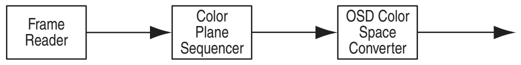
Figure 3. Video Input Channel



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 UDX3 reference design has two identical video input channels, input_channel_1 and input_channel_2.

The OSD input channel starts at the Frame Reader block and extends to the OSD Color Space Converter block. Figure 4 shows the OSD input channel.

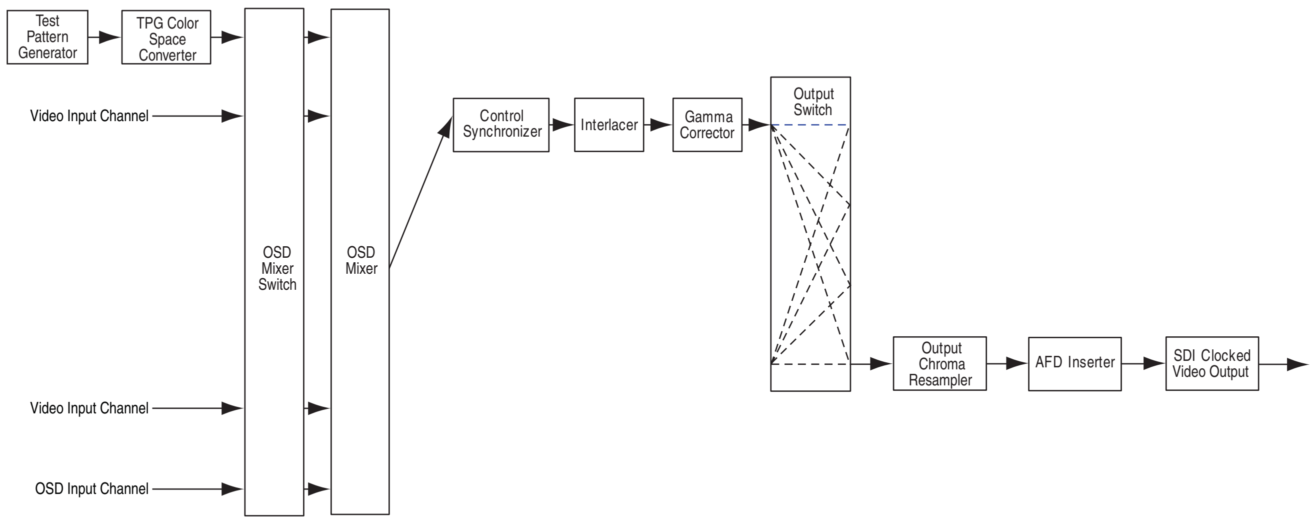
Figure 4. OSD Input Channel



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, 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 UDX3 reference design has a single OSD input 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. Figure 5 shows a video output channel.

Figure 5. Video Output Channel



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 UDX3 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).

The design has the following two runtime-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.

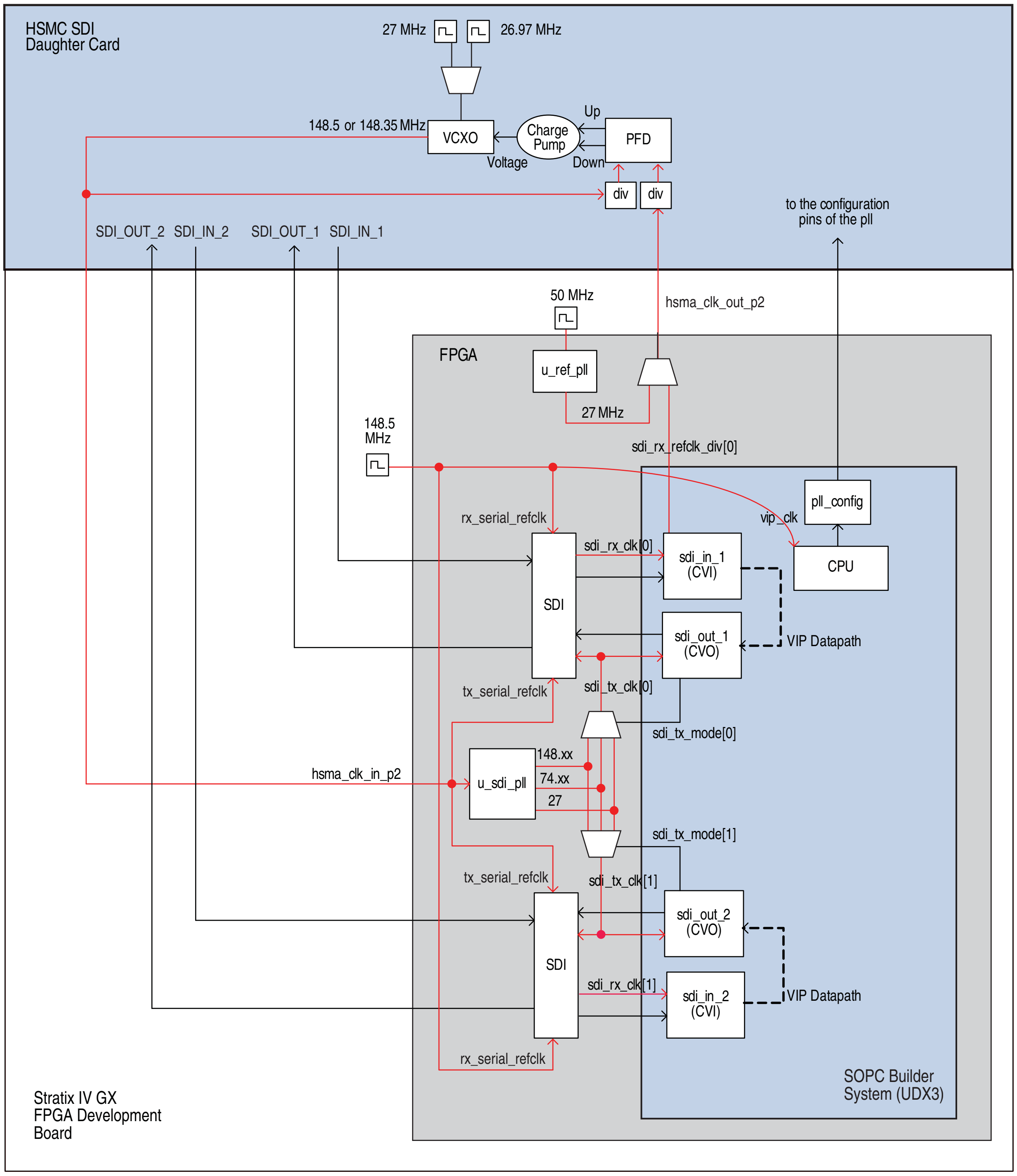
"User Controls" on page 37 provides information about how to change the mode at run time.

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 multi-port front end, runs at 200 MHz; the DDR3 memory runs at 400 MHz. The deinterlacer and the frame buffer include clock-crossing FIFOs 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 FIFOs to support communication between the video processing datapath and the incoming and outgoing video data, whose frequency conforms to the relevant video protocol.

Figure 6 shows the top-level clock connections for SDI data and control.

Figure 6. UDX3 Reference Design 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 as shown in Table 4. 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 video output streams from both of these channels are clocked by the same `hsma_clk_in_p2` output clock.

Table 4 shows a list of the available output frequencies for `sdi_tx_clk[0]` and `sdi_tx_clk[1]`.

Table 4. 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, all of the video outputs can be locked to the frequency of the video input stream that the UDX3 reference design receives on the SDI Input Channel 1 (J9) connector of the SDI HSMC card. The `hsma_clk_in_p2` clock is locked 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 CVI 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 `hsma_clk_in_p2` signal cannot be locked to `hsma_clk_out_p2`, then the FPGA outputs a 27 MHz reference clock instead of the `refclk_div` signal.

The `rx_serial_refclk` input clock to each SDI MegaCore function must be driven by a 148.5 MHz clock, to allow triple standard detection. The `tx_serial_refclk` input clock must be driven by a 148.5 MHz or 148.35 MHz clock that is locked to the video output clock, `sdi_tx_clk[0]`. In the UDX3 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 multiplexors, controlled by the `hdmi_mode_match` and `div_mode_match` signals, that select the correct output frequency. The clock, sync, and data signals of the DVI and HDMI transmitter chips are driven directly from the DVI clocked video output block and the HDMI clocked video output block, respectively.

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

Generator Lock

Generator lock is the technique of locking the timing of video outputs to a reference source. You can switch cleanly between sources that are locked to the same reference clock on a frame boundary. There are two derivatives of generator lock:

- Genlock—aligning 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)—aligning 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 is used to refer to both genlock and framelock.

Video output channel 1 is locked using the `sdi_in_1` CVI block as a reference source. Locking is achieved in two stages:

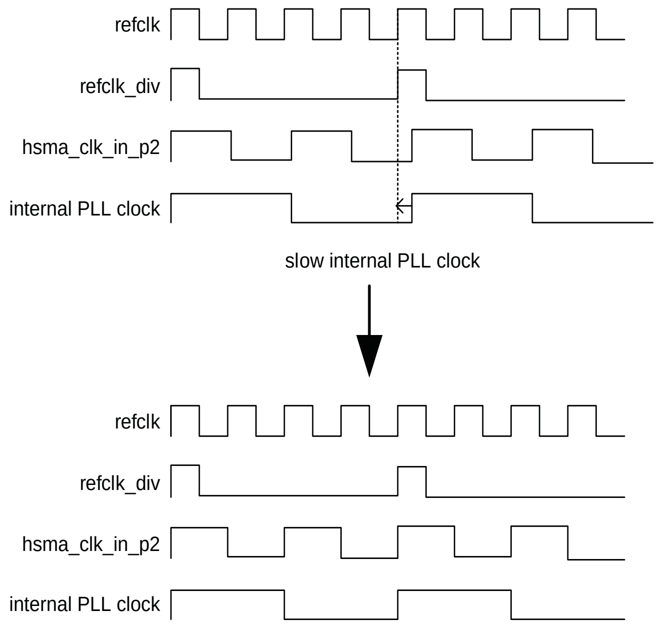
1. Clock locking—aligning the output video clock to the input video clock and tracking any changes.
2. Frame locking—aligning the start of frame (SOF) of the output video to the SOF of the input video.

Clock Locking

The UDX3 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 (CVI) 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 CVI 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`, as shown in Figure 7.

Figure 7. Clock Locking



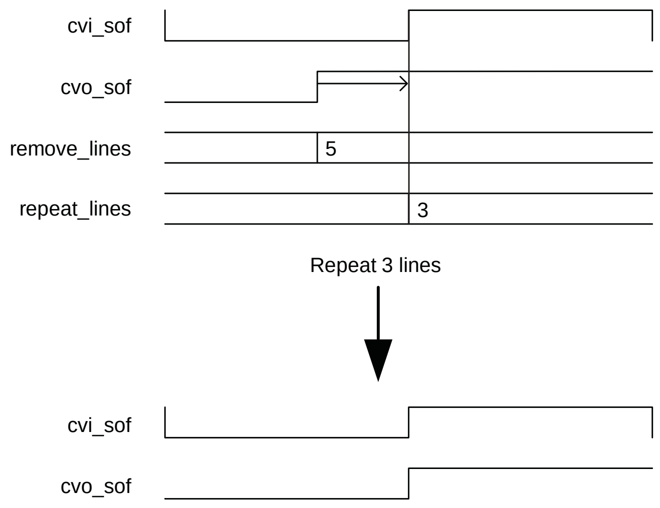
Frame Locking

The Clocked Video Output MegaCore function performs frame locking by aligning the SOF of the output video to the SOF produced by the Clocked Video Input MegaCore function. Each clocked video output block receives the SOF signal from the `sdi_in_1` CVI 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 is connected to the Clocked Video Output MegaCore function. A SOF is indicated by a rising edge transition (0 to 1). The SOF position can be set 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 8 shows an example of the SOF transitions.

Figure 8. 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 two fields for interlaced input—one field delay through the deinterlacer, and one field delay through the frame buffer—and two frames for progressive input—one frame delay in the deinterlacer and one frame delay 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 UDX3 reference design shares a single external DDR3 SDRAM memory for video buffering, OSD drawing, and the Nios II control software. The SOPC Builder system includes a single high-performance controller to coordinate accesses to this memory.

The memory subsystem is designed to minimize the impact of bank management commands (non-data transfer) commands sent 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), a burst can be be transferred to and from the DDR3 SDRAM while the next or previous burst is processed. This action allows the video processing data path to cope with the longer latencies created by the arbitration of a multi-master system all performing long bursts. The on-chip buffers are also used for crossing the clock domains between the Video and Image Processing Suite MegaCore functions and the memory controller. This action allows the memory to be run 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 17 masters require access to the external memory through the DDR3 SDRAM high-performance memory 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

To achieve the optimal minimum memory access efficiency of 90%, a multi-port front end controls access to the DDR3 high-performance memory controller. The multi-port front end arbitrates between the master accesses to the single slave port of the memory controller. The UDX3 reference design uses the SOPC Builder default round-robin arbitration scheme to reduce the number of ports on the multi-port front end to 14. Figure 9 shows the block diagram of the memory subsystem, and indicates the arbitration priorities. Ports with lower numbers have higher priority.

Figure 9. UDX3 Reference Design Memory Subsystem Block Diagram

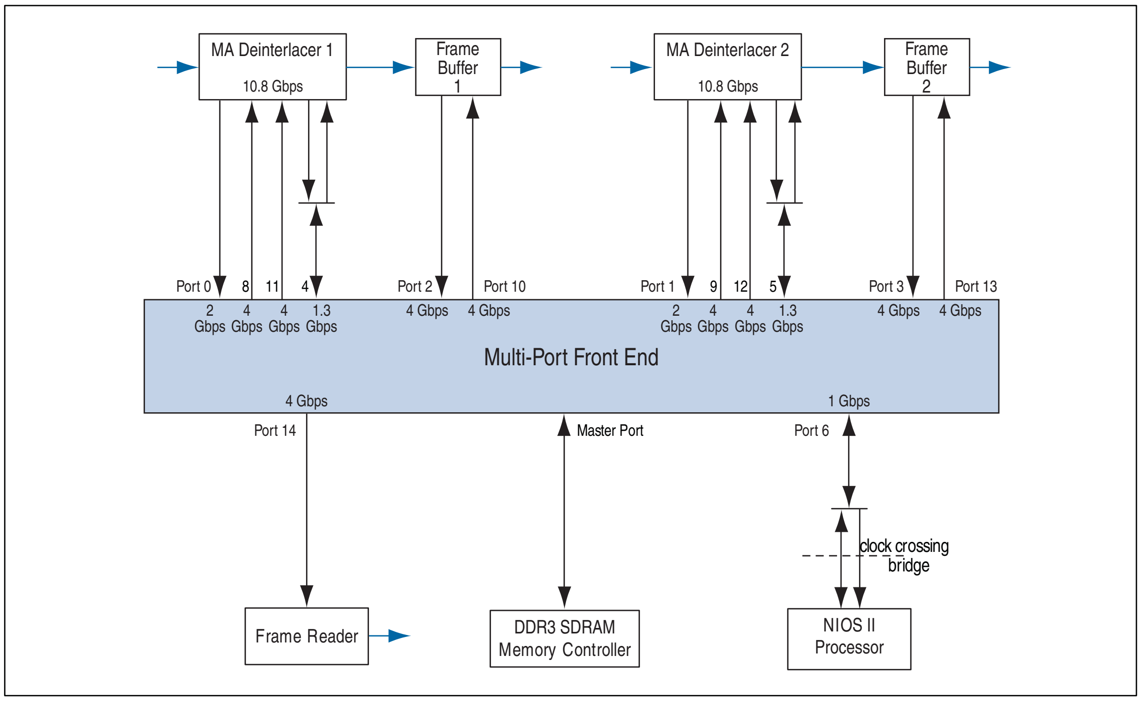


Table 5 describes the access patterns on each multi-port front end Avalon-MM port, and shows the allocation of ports to different memory banks. Ports are allocated to different banks according to the base addresses in SOPC Builder.

Table 5. Memory Subsystem Avalon-MM Port Access Patterns (Part 1 of 2)

Port	Width (Bits)	Max Burst Size (Words)	Description	Access Pattern	Latency (1), (2)	Bandwidth (Gbps)	Memory Bank
0	256	64	MA Deinterlacer 1 write port	Burst write of length 64 every 6.9 µs Delay of 3.7 µs after every fourth write (horizontal blanking) Extra 1 burst write after 4050 burst writes (start of frame) of length 64	6.9 µs	1.991	4
1	256	64	MA Deinterlacer 2 write port	Same as Port 0	6.9 µs	1.991	3
2	256	64	Frame Buffer 1 write port	Burst write of length 64 every 3.45 µs Delay of 1.85 µs after every fourth write (horizontal blanking) Extra 1 burst write after 8100 burst writes (start of frame) of length 64	3.45 µs	3.981	6
3	256	64	Frame Buffer 2 write port	Same as Port 2	3.45 µs	3.981	2

Table 5. Memory Subsystem Avalon-MM Port Access Patterns (Part 2 of 2)

Port	Width (Bits)	Max Burst Size (Words)	Description	Access Pattern	Latency (1), (2)	Bandwidth (Gbps)	Memory Bank
4	256	64	MA Deinterlacer 1 motion write port motion read port	 Burst write of length 64 every 21.54 µs Burst write of length 64 every 21.54 µs	10.77 µs	 0.637 0.637	4
5	256	64	MA Deinterlacer 2 motion write port motion read port	Same as Port 4	10.77 µs	 0.637 0.637	3
6	32	1	Nios II processor read and write ports	Cache-line fills and write-backs; Width adapter converts these accesses to 256-bit transactions	—	1	0
7	Unused						
8	256	64	MA Deinterlacer 1 read port 0	Burst read of length 64 every 3.45 µs Delay of 1.85 µs after every fourth read (horizontal blanking) Extra 2 × 1 burst write after 4050 burst reads (start of frame) of length 64	3.45 µs	3.981	4
9	256	64	MA Deinterlacer 2 read port 0	Same as Port 8	3.45 µs	3.981	3
10	256	64	Frame Buffer 1 read port	Burst write of length 64 every 3.45 µs Delay of 1.85 µs after every fourth read (horizontal blanking) Extra 1 burst write after 8100 burst reads (start of frame) of length 64	3.45 µs	3.981	6
11	256	64	MA Deinterlacer 1 read port 1	Same as Port 8	3.45 µs	3.981	4
12	256	64	MA Deinterlacer 2 read port 1	Same as Port 8	3.45 µs	3.981	3
13	256	64	Frame Buffer 2 read port	Same as Port 10	3.45 µs	3.981	2
14	256	64	Frame Reader read port	Burst write of length 64 every 3.447 µs Delay of 1.85 µs after every fourth read (horizontal blanking) Extra 1 burst read after 7776 burst reads (start of frame) of length 64	3.447 µs	3.981	1

Notes to Table 5:

(1) Write latency is the time from the write request arrival on the Avalon-MM port to acceptance of the first beat of data by the requestor.

(2) Read latency is the time from the read request arrival on the Avalon-MM port to receipt of the first beat of data by the requestor.

Bandwidth Calculations

Because the memory subsystem packs 30-bit colors in 256-bit data words, some of the bits in a data word are not used. These bits must be added to the bandwidth requirement calculation. The following basic calculations are incorporated in the bandwidth calculation for each block:

256 bits / 30 bits = 8.53 30-bit samples

8 samples × 30 bits = 240 bits

256 bits – 240 bits = 16 bits

16 bits / 8 samples = 2 extra bits per 30-bit sample

and

256 bits / 10 bits = 25.6 10-bit motion data rate values

25 motion values × 10 bits = 250 bits

256 bits – 250 bits = 6 bits

6 bits / 25 motion values = 0.24 extra bits per 10-bit motion value

MA Deinterlacer Bandwidth Calculation

For input format 1080i60, the input rate is

1920 × 1080 × 32 bits × 60/2 frames per second (fps) = 1.991 Gbps

For output format 1080p60, the output rate is

1920 × 1080 × 32 bits × 60 fps = 3.981 Gbps

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

1920 × 1080 × 10.24 bits × 60/2 fps = 0.637 Gbps

A memory access consists of the following operations:

- 1 write at input rate 1.991 Gbps
- 1 write at motion rate 0.637 Gbps
- 1 read at motion rate 0.637 Gbps
- 2 reads at total output rate 7.963 Gbps

which comes to a total bandwidth of

1.991 + 0.637 + 0.637 + 7.963 = 11.228 Gbps

Frame Buffer Bandwidth Calculation

For input format 1080p60, the input rate is

1920 × 1080 × 32 bits × 60 fps = 3.981 Gbps

For output format 1080p60, the output rate is

1920 × 1080 × 32 bits × 60 fps = 3.981 Gbps

A memory access consists of the following operations:

- 1 write at input rate 3.981 Gbps
- 1 read at output rate 3.981 Gbps

which comes to a total bandwidth of

3.981 + 3.981 = 7.963 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

Profiling the Nios II memory usage in hardware shows approximately 1 Gbps for control and OSD update functions.

Total System Bandwidth Calculation

Using the results from the preceding sections, the calculations of the total bandwidth for the video paths only and for the video paths plus OSD and Nios II processor are shown in Table 6.

Table 6. Total System Bandwidth Calculation

Block	**Bandwidth (Gbps)**
Video paths only:	
MA Deinterlacer 1	11.228
MA Deinterlacer 2	11.228
Frame Buffer 1	7.963
Frame Buffer 2	7.963
Total	38.382
Video paths with OSD and the Nios II processor:	
Video path total	38.382
OSD to Frame Reader	3.981
Nios II Processor	1
Total	**43.363 Gbps**

Memory Bandwith Requirements

Memory bandwidth efficiency is determined by a number of factors, 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

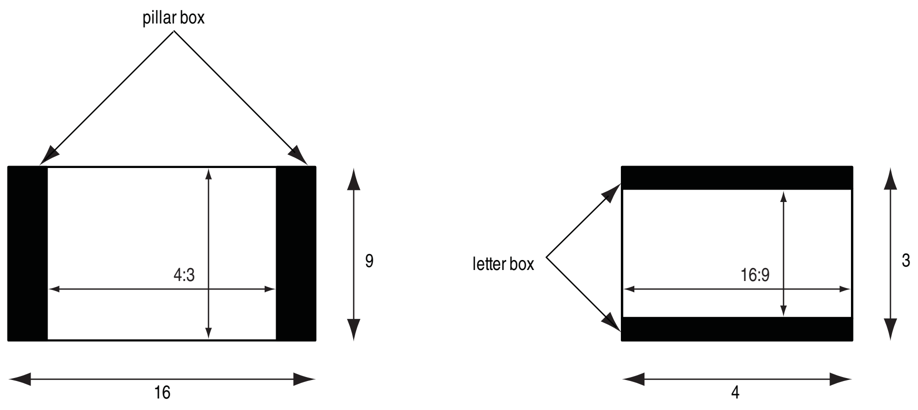
400 MHz $\times$ 64 bits $\times$ 2 (double rate: both clock edges used) = 51.2 Gbps

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 memory access efficiency utilization is approximately 75%. Adding the OSD input channel and storing the Nios II software in the same DDR3 memory leads to a DDR3 memory access efficiency utilization of approximately 85%.

Active Format Description

The active format description (AFD) is a standard code that enables the active picture region of the video stream to be interpreted correctly. The code defines the areas of the active picture that contain valid video data and the areas that are unused. This encoding allows content with one aspect ratio to be transmitted and displayed correctly using a different format. For example, video data formatted for a 4:3 aspect ratio can be displayed correctly in 16:9 ratio if the AFD code of the incoming data indicates its 4:3 format. Figure 10 shows two common aspect ratios and the unused areas when data originally in one format is displayed in the other format.

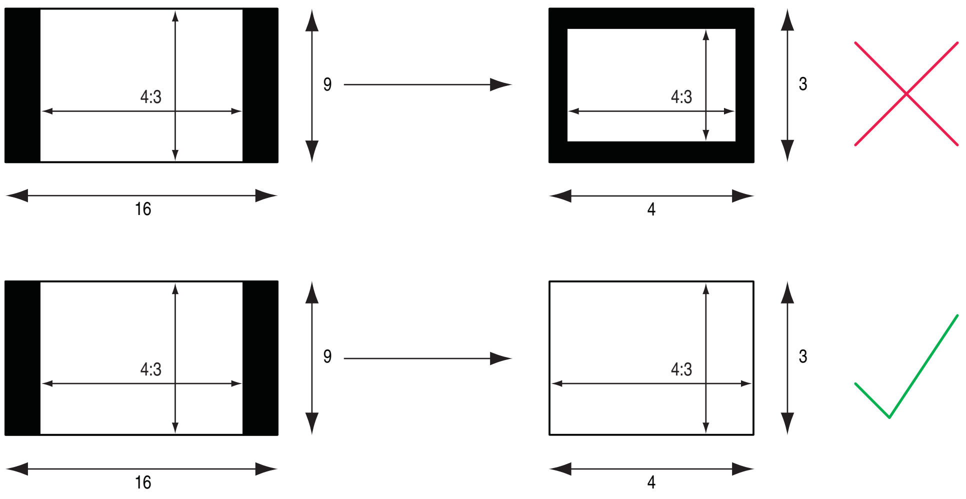
Figure 10. Example Aspect Ratios and Their Display



In the UDX3 reference design, the scaler and mixer are configured to maintain aspect ratio. When the output resolution is set, the Nios II processor compares the output aspect ratio with that of the incoming video content. If the aspect ratio is different, the scaler is configured 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.

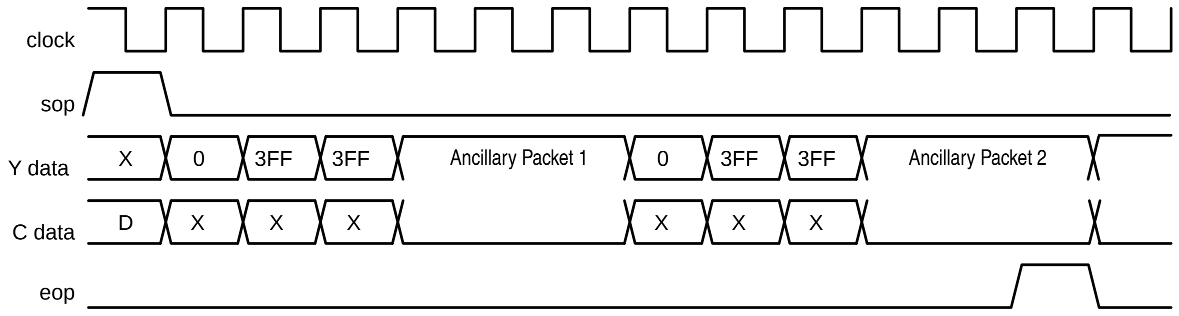
SDI input streams include their own AFD codes. The UDX3 reference design uses the AFD code to maintain the aspect ratio of the incoming video by having AFD Clipper blocks clip the pillar box or letter box bars when necessary. Figure 11 shows what can happen if the AFD code is not read and the bars are not clipped.

Figure 11. Incorrect and Correct AFD Handling



In embedded sync mode, the clocked video input block extracts the ancillary packets from the Y channel during the vertical blanking period. The block outputs these packets with a packet type of 0xD on its Avalon-ST interface. Figure 12 shows an example of an Avalon-ST Video ancillary packet that contains two SDI ancillary packets.

Figure 12. Ancillary Packet Format



The AFD extractor block removes every ancillary packet from the stream and checks its Data ID (DID) and secondary DID to determine whether this packet is an AFD packet. An AFD packet has DID value 0x41 and secondary DID value 0x5. If the packet is an AFD packet, the AFD extractor block extracts the contents of the ancillary packet and places the values in its register map.

Table 7 shows the AFD extractor block register map.

Table 7. AFD Extractor Block Register Map

Address	Register	Description
0	Control	Bit 0 determines whether the AFD extractor block discards all packets (at default value 0) or passes through all packets except ancillary packets (at value 1).
1	Reserved	

Table 7. AFD Extractor Block Register Map

Address	Register	Description
2	Interrupt	Bit 1 indicates that a change has been detected in the AFD data and the interrupt has fired. Writing the value 1 to bit 1 clears the interrupt.
3	AFD	Bits [3:0] contain the decoded AFD code.
4	AR	Bit 0 contains the decoded aspect ratio code.
5	Bar Data Flags	If the AFD code is 4'b0000 or 4'b0100, bits [3:0] describe the content of the Bar Data Value 1 and 2 registers. However, the current software implementation does not support these AFD codes, so the contents of this register and the Bar Data Value registers are ignored.
6	Bar Data Value 1	Bits [15:0] contain the information described in the Bar Data Flags register.
7	Bar Data Value 2	Bits [15:0] contain the information described in the Bar Data Flags register.
8	AFD Valid	Bit 0 indicates the current image packet includes an AFD code.

The AFD inserter block adds an AFD ancillary packet to the stream following each control packet. Table 8 shows the AFD inserter block register map.

Table 8. AFD Inserter Block Register Map

Address	Register	Description
0	Control	Bit 0 determines whether the AFD inserter block passes through all packets (at default value 0) or passes through all packets and inserts an AFD ancillary packet after each control packet (at value 1).
1	Reserved	
2	Reserved	
3	AFD	Bits [3:0] contain the decoded AFD code that the AFD inserter block inserts.
4	AR	Bit 0 contains the decoded aspect ratio code that the AFD inserter block inserts.
5	Bar Data Flags	Bits [3:0] contain the Bar data flags that the AFD inserter block inserts.
6	Bar Data Value 1	Bits [15:0] contain the Bar Data Value 1 value to insert.
7	Bar Data Value 2	Bits [15:0] contain the Bar Data Value 2 value to insert.

In embedded sync format, the clocked video output block inserts ancillary packets in the output video data during the vertical blanking period. The block inserts the packets starting on the F0 and F1 lines specified in the mode registers for the block, and stops inserting the packets at the end of the vertical blanking period. Refer to Table 9 for a description of the mode registers.

The Clocked Video Output MegaCore function register map is an extended version of the register map documented in the Video and Image Processing Suite User Guide. Table 9 shows the modified register map for the clocked video output blocks in the UDX3 reference design.

Table 9. Clocked Video Output Block Register Map (Part 1 of 2)

Address	Register	Description
0–25	MegaCore-defined registers	As shown for the Clocked Video Output MegaCore function in the *Video and Image Processing Suite User Guide*.
26	Mode1 Ancillary Line	The line number to begin inserting ancillary data for F0 fields or progressive frames.

Table 9. Clocked Video Output Block Register Map (Part 2 of 2)

Address	Register	Description
27	Mode1 F1 Ancillary Line	The line number to begin inserting ancillary data for F1 fields.
28	Mode1 Valid	Video mode 1 valid. Set to indicate that this mode is valid and can be used for video output.
29	Mode2 Control	For this register and following registers, refer to the *Video and Image Processing Suite User Guide.*

The UDX3 reference design software recognizes and inserts only a limited set of AFD codes. You can easily extend the software to support all the AFD codes, by changing the `update_coefficients()` function in the **Video_Input_Channel.hpp** file. A full list of AFD codes is included in the comments for the function. Table 10 lists the supported codes. The unextended software ignores AFD codes not listed in Table 10.

Table 10. 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 UDX3 reference design, you load the control software to 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 memory, 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 stored 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 11 and Table 12 show the memory maps for the Nios II processor and the flash memory device.

Table 11. Nios II Processor Instruction Master Memory Map

Slave Device	Nios II Processor Address Base or Range		
Flash memory	0x20000000–0x23FFFFFF (Refer to Table 12 for internal offsets)		
DDR3 SDRAM	**Description**	**Address Base or Range**	**Memory Bank**
	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	0x0–0x2000000	0

Table 12 shows the flash memory map. The table lists the offset ranges of the different blocks in the flash memory device.

Table 12. Flash Memory Map

Memory Block Use	Size (KBytes)	Flash Memory Device Offset Range
Unused	128	0x03FE0000–0x03FFFFFF
User software image	24320	0x02820000–0x03FDFFFF
Factory software image	8192	0x02020000–0x0281FFFF
ZIPFS file system	8192	0x01820000–0x0201FFFF
User hardware image	12288	0x00C20000–0x0181FFFF
Factory hardware image	12288	0x00020000–0x00C1FFFF
PFL option bits	32	0x00018000–0x0001FFFF
Board information	32	0x00010000–0x0001FFFF
Ethernet option bits	32	0x00008000–0x0000FFFF
User design reset vector	32	0x00000000–0x00007FFF

Running the Reference Design

To run the reference design, perform the following actions:

1. Install the reference design files on your computer.
2. Connect the hardware.
3. Create and download a SRAM Object File (**.sof**) to configure the FPGA with a flash programmer target design. For this purpose, you can use the **.sof** file that contains the hardware image of the reference design.

4. 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 be mixed, potentially, with incoming video data

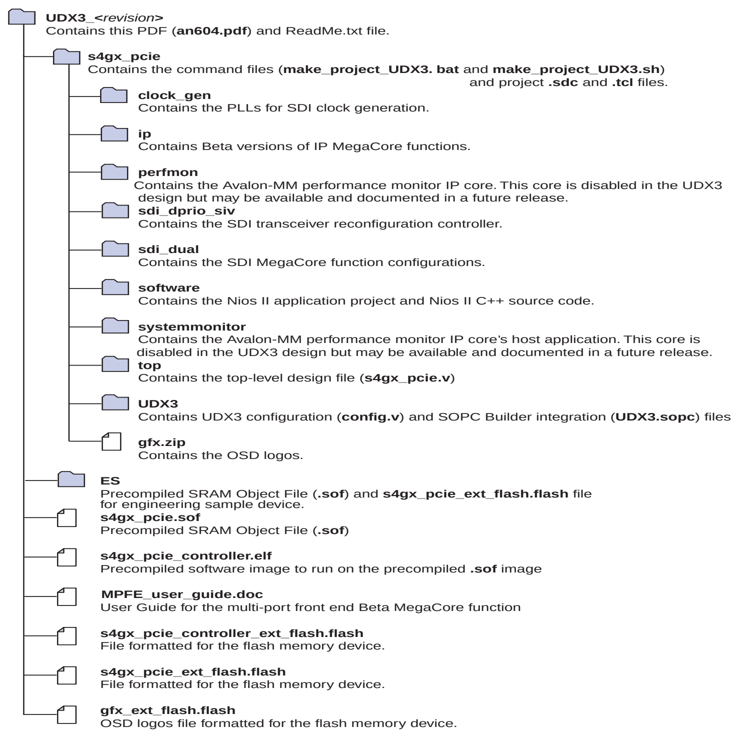
This section tells you how to perform these actions.

Installing the Reference Design

All of the files necessary for this reference design are included in the **UDX3_**<*version*>**.zip** file. This file is available for you to download from the Format Conversion Reference Design (UDX3) web page. Alternatively, from the Broadcast web page on the Altera website, follow the link to the Reference Designs page.

Unzip the **UDX3_**<*version*>**.zip** file in the working directory you designated for this project. Figure 13 shows the directory structure for the UDX3 reference design files after you extract them from the **.zip** file.

Figure 13. Reference Design Directory Structure


UDX3_<revision>
Contains this PDF (an604.pdf) and ReadMe.txt file.
s4gx_pcie
Contains the command files (make_project_UDX3. bat and make_project_UDX3.sh) and project .sdc and .tcl files.
clock_gen
Contains the PLLs for SDI clock generation.
ip
Contains Beta versions of IP MegaCore functions.
perfmon
Contains the Avalon-MM performance monitor IP core. This core is disabled in the UDX3 design but may be available and documented in a future release.
sdi_dprio_siv
Contains the SDI transceiver reconfiguration controller.
sdi_dual
Contains the SDI MegaCore function configurations.
software
Contains the Nios II application project and Nios II C++ source code.
systemmonitor
Contains the Avalon-MM performance monitor IP core's host application. This core is disabled in the UDX3 design but may be available and documented in a future release.
top
Contains the top-level design file (s4gx_pcie.v)
UDX3
Contains UDX3 configuration (config.v) and SOPC Builder integration (UDX3.sopc) files
gfx.zip
Contains the OSD logos.
ES
Precompiled SRAM Object File (.sof) and s4gx_pcie_ext_flash.flash file for engineering sample device.
s4gx_pcie.sof
Precompiled SRAM Object File (.sof)
s4gx_pcie_controller.elf
Precompiled software image to run on the precompiled .sof image
MPFE_user_guide.doc
User Guide for the multi-port front end Beta MegaCore function
s4gx_pcie_controller_ext_flash.flash
File formatted for the flash memory device.
s4gx_pcie_ext_flash.flash
File formatted for the flash memory device.
gfx_ext_flash.flash
OSD logos file formatted for the flash memory device.

The **s4gx_pcie** subdirectory contains command file **make_project_UDX3.bat** for Windows and **make_project_UDX3.sh** for Linux. In "Compiling the UDX3 Reference Design Software" on page 32 you run this command file to generate the Quartus II project file for the UDX3 reference design.

Connecting the Hardware

In this section, you connect the Stratix IV GX FPGA development board and the required daughter cards to support the UDX3 reference design.

For information about the Stratix IV GX FPGA Development Kit, including a user guide and a reference manual, refer to the Stratix IV GX FPGA Development Kit webpage or the Audio Video Development Kit, Stratix IV GX Edition webpage.

For information about the SDI HSMC daughter card, refer to the *SDI HSMC Reference Manual* or the Terasic website. For information about the DVI HSMC daughter card, refer to the Bitec website.

To connect the boards to run the UDX3 reference design, perform the following steps:

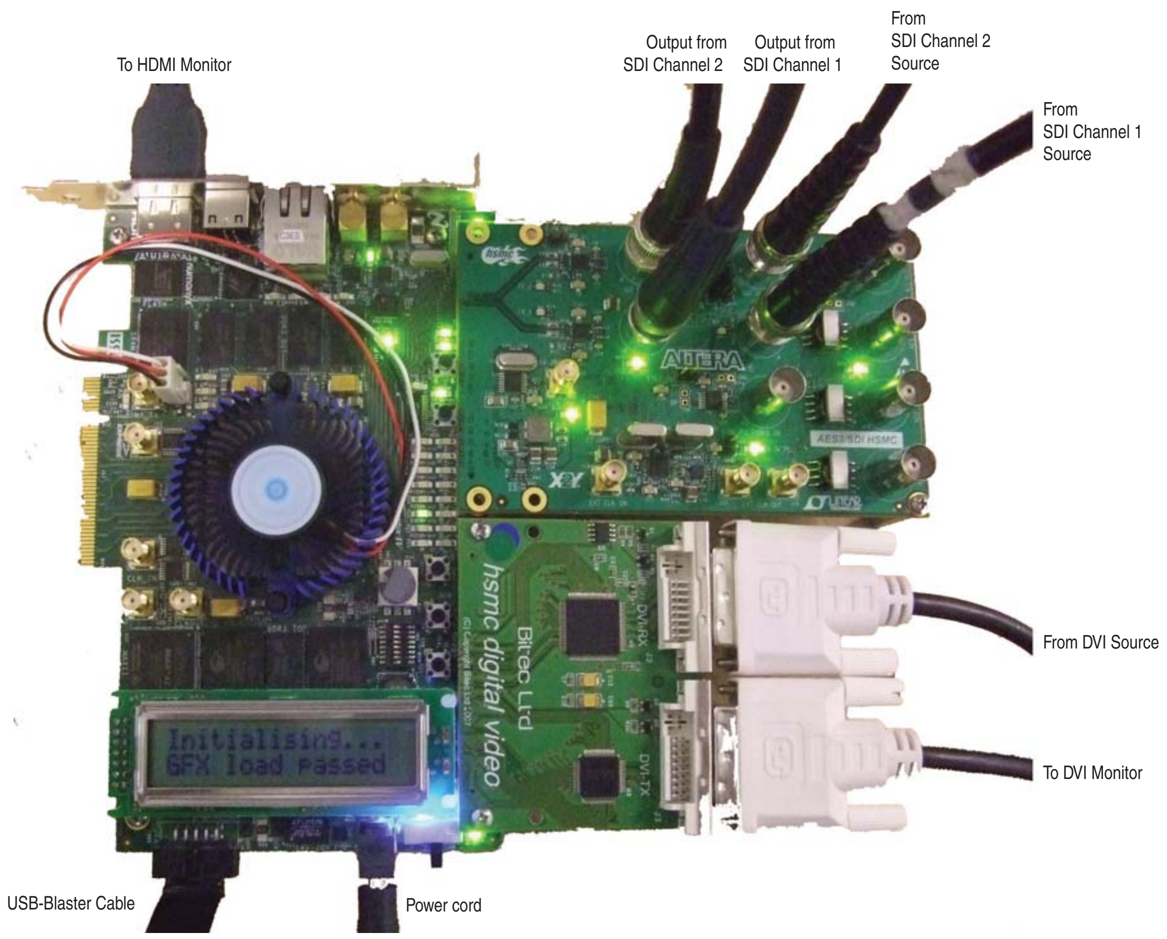
1. Ensure that the Power Switch (SW1) on your Stratix IV GX FPGA development board is turned off.
2. Connect the SDI HSMC board to HSMC Port A (J1) of the Stratix IV GX FPGA development board.
3. Connect the Bitec HSMC DVI interface card to HSMC Port B (J2) of the Stratix IV GX FPGA development board.
4. Connect the first video source to the hardware system, by performing one of the following actions:
   - 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 interface card.

   In the UDX3 design, 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, independent of the expected format of the first video source.
6. Connect an HDMI monitor to the HDMI Video Port (J11) on the Stratix IV GX FPGA development board.
7. Connect a DVI monitor to the DVI-TX connector on the Bitec HSMC DVI interface card.
8. Connect an SDI monitor to the SDI Output Channel 1 connector (J8) on the SDI HSMC board.
9. Connect a second SDI monitor to the SDI Output Channel 2 connector (J1) on the SDI HSMC board.

10. Depending on your Stratix IV GX FPGA development board JTAG control DIP switch (J6) settings, perform one of the following actions:
    - Connect a USB-Blaster™ download cable to your computer and to the JTAG Connector (J8) on the Stratix IV GX FPGA development board.
    - Connect a USB download cable to your computer and to the USB-Type B Connector (J7) on the Stratix IV GX FPGA development board.
11. If your Stratix IV GX FPGA development board is disconnected from power, connect it to power.
12. Turn on the Power Switch (SW1) on your Stratix IV GX FPGA development board.

Figure 14 shows an example hardware setup for the UDX3 reference design.

Figure 14. Stratix IV GX FPGA Development Board With Daughter Cards


To HDMI Monitor
Output from SDI Channel 2
Output from SDI Channel 1
From SDI Channel 2 Source
From SDI Channel 1 Source
From DVI Source
To DVI Monitor
USB-Blaster Cable
Power cord

The two channels operate independently by default, unless you reconfigure them at run time. By default, the SDI HSMC board SDI Input Channel 1 or Bitec HSMC DVI interface card DVI-RX connector provides input to the reference design video processing path whose output appears on the HDMI monitor, and the SDI HSMC board SDI Input Channel 2 provides input to the reference design video processing path whose output appears on the Bitec HSMC DVI interface card DVI-TX connector.

Compiling the Quartus II Project and Configuring the FPGA for Flash Programming

In this section, you use the Quartus II software, SOPC Builder, and the Quartus II Programmer to compile the Quartus II project and configure the FPGA.

To compile the Quartus II project and configure the FPGA, perform the following steps:

1. Change directory to **s4gx_pcie**.
2. To create the UDX3 reference design Quartus II Project File (**.qpf**), **s4gx_pcie.qpf**, perform one of the following actions:
   - In a Windows operating system, double-click the script **make_project_UDX3.bat**.
   - On the Linux command line, type the following command:

     ```
     sh make_project_UDX3.sh ↵
     ```
3. Open the Quartus II project file **s4gx_pcie.qpf**.
4. On the Assignments menu, click **Device**. The **Device Settings** dialog box appears.
5. Under **Available devices**, select **EP4SGX230KF40C3**.
6. Click **OK**.
7. In the Quartus II software, on the Tools menu, click **SOPC Builder**.
8. Review the UDX3 reference design SOPC Builder system. Figure 2 on page 6 shows the block diagram of the SOPC Builder system.
9. In SOPC Builder, click the **System Generation** tab.
10. Turn off **Simulation. Create project simulator files.** System generation executes much faster when simulation is off.
11. Click **Generate**. System generation may take a few minutes.
12. When the **System generation was successful** message displays, click **Exit**.
13. If you are prompted to save your changes, click **Save**.
14. In the Quartus II software, on the Processing menu, click **Start Compilation**. Compilation creates the **s4gx_pcie.sof** file.
15. After compilation completes, on the Tools menu, click **Programmer**.
16. In the Quartus II Programmer, under **Mode**, verify that **JTAG** is selected.
17. Click **Hardware Setup** to configure the programming hardware. The **Hardware Setup** dialog box appears.
18. In the **Hardware** column, double-click **USB Blaster**.
19. Click **Close** to exit the **Hardware Setup** window.
20. In the Quartus Programmer, click **Add File**, navigate to the **s4gx_pcie.sof** file you created in step 14, and click **Open**.
21. Ensure the **Program/Configure** box for the new file contains a checkmark.
22. Click **Start**. The **.sof** file is downloaded to your Stratix IV GX FPGA development board and the FPGA is configured with the hardware image.

For information about SOPC Builder, refer to *Volume 4: SOPC Builder* 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*. 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*.

Compiling the UDX3 Reference Design Software

In this section, you compile the UDX3 reference design software using the Nios II Software Build Tools (SBT) for Eclipse™.

To compile and download the UDX3 reference design software in the Nios II SBT for Eclipse, perform the following steps:

1. On the **Start** or **Programs** menu, under **Altera** > **Nios II EDS** <*version*>, click **Nios II** <*version*> **Software Build Tools for Eclipse**; or in SOPC Builder, in the **System Generation** tab, click **Nios II 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/UDX3.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 project creation completes, 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 **altmemddr_0**, set the **Size (bytes)** value to 33553825. This value limits the Nios II address space to 32 MBytes and prevents the stack from overlapping an area of memory used by a frame buffer or deinterlacer.
11. In the **Software Packages** tab, enable the **altera_ro_zipfs** software package and set its parameter values as shown in Table 13.

Table 13. 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 you are prompted 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 resulting **.elf** file is downloaded to the DDR3 memory on your Stratix IV GX FPGA development board.

    Although the following UDX3 reference design instructions program the **.elf** file to flash memory, the SBT for Eclipse copies the file directly to DDR3 memory to enable you to test software changes quickly.

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 On-Screen Display Logos

The **gfx.zip** file holds the OSD logos in 8-bit raw RGBA format. To replace OSD logos in the UDX3 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 UDX3 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, perform the following 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 by performing the following steps:
    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, perform the following 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. Note the height (<*height*>) and width (<*width*>) of the modified logo, which appear in the title bar of the main Gimp window.
6. To add the new logo to the **gfx.zip** file, perform the following 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, 262144 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 UDX3 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 On-Screen Display Logos" on page 33 before following the instructions in this section.

To program the UDX3 reference design **.sof** file, **.elf** file, and OSD logos to flash memory, perform the following 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_0** 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. In the **Properties** dialog box, ensure the settings are as shown in Table 14.

Table 14. 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 **s4gx_pcie.sof** entry row in the **Flash Offset** column and type the value `0xC20000`.
19. Click **Add**.
20. Browse to locate the **s4gx_pcie/gfx.zip** file.
21. Click **Select**.
22. Double-click the **gfx.zip** entry row in the **Flash Offset** column and type the value `0x3020000`.
23. Click **Start** to program all of the files in the flash memory device.

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, perform the following steps:

1. Set the power monitor rotary switch (SW2) on the Stratix IV GX FPGA development board to 1.
2. Turn SW1 off and back on to load and run the UDX3 reference design.

If the red D27 light on your Stratix IV GX FPGA development board is lit after you perform step 2, perform step 15 to step 22 in "Compiling the Quartus II Project and Configuring the FPGA for Flash Programming" on page 31 to download the **s4gx_pcie/s4gx_pcie.sof** file.

Status Displays

The LCD screen displays messages that describe the state of the software initialization and start up. Table 15 describes these messages.

Table 15. LCD Status Messages

Message	Description
Initializing...	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 16 and Table 17 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 16. LED Status Information

LED	Status
0	When this LED is lit, one of the clocked video output channels has underflowed. The LED resets every second.
1	When this LED is lit, one of the clocked video input channels has overflowed. The LED resets every second
2	Heartbeat. This LED flashes when the software is running.
3	When this LED is lit, Channel 1 is active.
4	When this LED is lit, Channel 2 is active.
5, 6, 7	These three LEDs indicate the current mode. Refer to Table 17 for details.
8	The HDMI output data is genlocked to SDI Input Channel 1.
9	The SDI Output Channel 1 data is genlocked to SDI Input Channel 1.
10	The SDI Output Channel 2 data is genlocked to SDI Input Channel 1.
11	The DVI output data is genlocked to SDI Input Channel 1.

Table 17 shows the correlation between the settings of LEDs 5, 6, and 7 and the current mode. LED 7 is included to support future expansion of the reference design.

Table 17. LED Display of UDX3 Reference Design Mode

Value of LED			
5	**6**	**7**	**Meaning**
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 the DVI-TX connector on the Bitec HSMC DVI interface card.
Lit	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	Lit	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.
Lit	Lit	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 runtime-configurable parameters using the Stratix IV GX FPGA development board push-button switches. Table 18 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 18. Push-Button Switch Controls (Part 1 of 2)

Push-Button Switch	Function
S5 (PB0)	Cycles through the four modes shown in Table 17. The control software synchronizes 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, as follows: ■ HDMI output cycles through 720p60, 480p60, and 1080p60. ■ SDI output cycles through 720p60, NTSC, and 1080i60.

Table 18. Push-Button Switch Controls (Part 2 of 2)

Push-Button Switch	Function
S3 (PB2)	Changes the output resolution of channel 2, as follows: ■ DVI output cycles through 720p60, 480p60, and 1080p60. ■ SDI output cycles through 720p60, NTSC, and 1080i60.
S2 (CPU Reset Push-Button Switch)	Resets the UDX3 reference design by restarting the software.

Revision History

Table 19 shows the revision history for this document.

Table 19. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
March 2010 1.0	Initial release.	—

101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support
www.altera.com/support