4K Format Conversion Reference Design

This application note describes a 4K format conversion reference design. 4K resolution is the next major enhancement in video because of the benefits in picture clarity and realism. Many leading projector, broadcast, and camera manufacturers are shipping 4K enabled systems. Altera enables this next generation format conversion by reducing the system device count, which lowers the overall costs, reduces the cost of development, and simplifies board design. Previous systems required as many as nine off-the-shelf devices to perform 4K format conversion—four 1080p format conversion devices and five devices for serial digital interface (SDI) input and output. The 4K format conversion reference design uses less than 50% of a single Altera® Stratix® IV EP4SGX230 FPGA. With migration paths to any Altera FPGA device families, you can integrate these functions with ample headroom remaining to incorporate other video functionality and interfaces such as DisplayPort and video compression (encoding or decoding) processing.

4K resolution refers to any resolutions that have approximately 4,000 horizontal pixels on a display screen. In digital cinema, a typical resolution is 4,096 by 2,160 pixels, and in computer graphics, quad full high definition (QFHD) is 3, 840 by 2,160 pixels. Typically, the processing of 4K video requires more than four times the processing capability of 1080p60 video.

Features

The reference design offers the following features:

- Support for the following inputs:
  - PAL
  - 720p60
  - 1080i50
  - 1080i60
  - 1080p60
- One QFHD output transmitted over four 3G-SDI 1080p60 outputs.
- Four video processing pipelines running at 148.5 MHz and each including:
  - A prescaler clipper to select a portion of the input video lines to be upscaled
  - A scaler with eight horizontal and eight vertical taps performing a four times upscale
  - Double buffering of the video to external DDR3 SDRAM
- System initialization and run-time configuration in software.

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

- Rapid system capture and design with Qsys, the Quartus® II software, and the Nios® II development environments.
- Highly parameterizable and modular hardware functions:
  - Video and Image Processing Suite MegaCore functions
  - Nios II processor
  - Clear text system-configuration software
  - DDR3 SDRAM controllers
  - Peripherals
  - Autogeneration of switch fabric
- Standard Avalon® Streaming (Avalon-ST) and Avalon Memory-Mapped (Avalon-MM) interfaces for rapid integration and the Avalon-ST Video protocol for video transmission between functions.
- Runs on the Altera Stratix IV GX FPGA Development Kit with two Terasic Transceiver SDI High-Speed Mezzanine Card (HSMC) boards.

For more information on the video and image processing component library and the Avalon-ST Message protocol, refer to the *Video and Image Processing Component Library Functional Description* (available from Altera).

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

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

General Description

The 4K format conversion reference design takes a PAL, 720p, 1080i, or 1080p input over a 3G-SDI interface and upscales it to QFHD resolution output over four 3G-SDI interfaces. The Altera SDI IP core supports the 3G-SDI interfaces in the FPGA. A video server provides the SDI input; the four SDI outputs display on four separate monitors, or a quartered 4K capable monitor. The reference design was demonstrated on a single Stratix IV EP4SGX230 FPGA at International Broadcast Convention (IBC) 2010.

The reference design uses IP cores from the Video and Image Processing Suite and components from the video and image processing component library, which is a collection of components that you use to build video and image processing IP cores or reference designs. The component library allows you to create more complex systems than the Video and Image Processing Suite offers. You cannot use component library components alone—you must also use a scheduler, for example, a CPU or state machine.

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

Performance and Resource Utilization

Table 1 lists the resource utilization on a Stratix IV GX device (S4GX230).

Table 1. Resource Usage

Usage	ALUTs	Logic Registers	Logic Utilization	Total Blocks				DSP Block 18-Bit Elements
				Memory Bits	Memory Implementation Bits	M9K	M144K	
On device	51,810	60,616	77,148	4,824,456	8,985,600	623	22	190
Total available on device	182,400	182,400	182,400	14,625,792	14,625,792	1,235	22	1,288
Percentage used on device	28%	33%	42%	33%	61%	50%	100%	15%

Functional Description

Figure 1 on page 4 shows a block diagram of the reference design.

Figure 1. Block Diagram

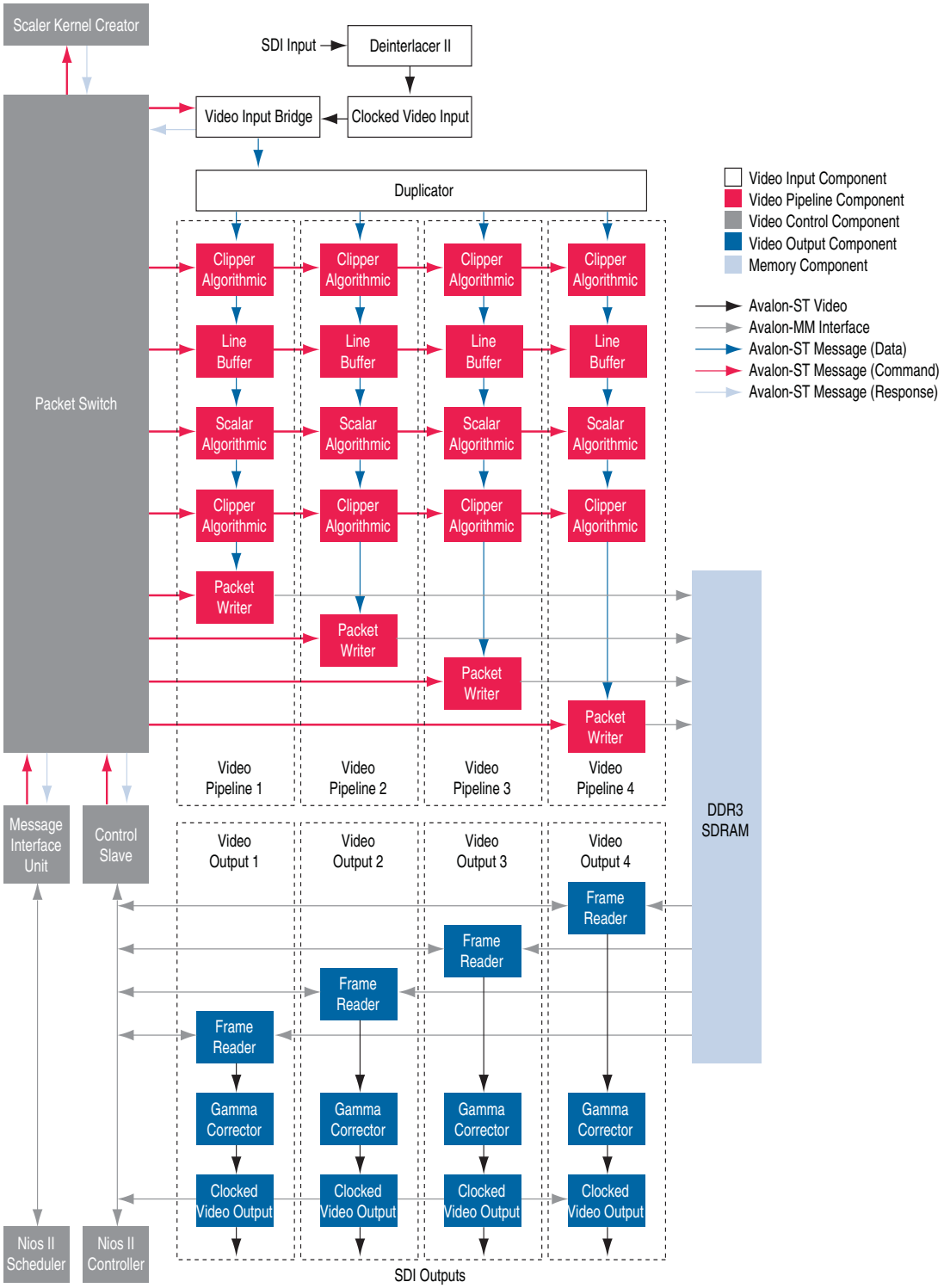


Table 2 describes the video input blocks. The video input takes a single SDI input, checks that it is in a supported format, and then duplicates it to the four video pipelines.

Table 2. Video Input Blocks

Block	Source	Description
Clocked video input IP core	Video and image processing suite	The clocked video input converts the output of the SDI IP core into Avalon-ST Video protocol.
Video input bridge	Video and image processing component library	The video input bridge alerts the scheduler to a new packet arriving on its Avalon-ST Video input and then sends it to the destination that the scheduler commands.
Duplicator	Video and image processing component library	The duplicator duplicates each received input packet to all of its four outputs.

Table 3 describes the video pipeline blocks. The video pipeline takes a portion of the video input and upscales it before writing the result into external memory.

Table 3. Video Pipeline Blocks

Block	Source	Description
Clipper algorithmic IP core	Video and image processing component library	The clipper algorithmic IP core clips away most of the input line allowing only approximately a quarter of it to propagate to its output. The quarter that is propagated is different for each video pipeline.
Line buffer	Video and image processing component library	The line buffer uses on-chip memory to store multiple lines and then outputs them in parallel as one packet. This reference design configures each line buffer to support four vertical taps.
Scaler algorithmic IP core	Video and image processing component library	The scaler algorithmic IP core upscales the input line by a factor of two.
Packet writer	Video and image processing component library	The packet writer writes the lines of output video frame (packets) into external memory.

Table 4 describes the video control blocks. The Nios II scheduler controls the IP cores. It allows you to use the register maps to configure, start, and stop the IP cores and also the component library components. The components require much lower-level control as they only perform tasks, such as processing input packets, when they receive a command from the scheduler.

Table 4. Video Control Blocks (Part 1 of 2)

Block	Source	Description
Scaler kernel creator	Video and image processing component library	The scaler kernel creator is a hardware accelerator block that returns the input lines required to produce an output line. The scheduler uses this fact to determine which input lines need to be stored in the line buffer.
Packet switch	Video and image processing component library	The packet switch routes messages to the end point specified in the destination address. This process allows the Nios II processor to send messages to any component in the reference design by altering the destination address.

Table 4. Video Control Blocks (Part 2 of 2)

Block	Source	Description
Nios II message interface unit	Video and image processing component library	Components use the Avalon-ST Message format to send and receive messages. The message interface unit is a memory-mapped peripheral that the Nios II processor uses to send and receive messages.
Control slave	Video and image processing component library	The control slave provides a register map interface between the Nios II controller and the Nios II scheduler. It updates scaling parameters and synchronizes the switching of the read and write sides of the DDR3 double buffer.
Nios II scheduler	Qsys	The design uses a Nios II processor to schedule the scaling subsystem in the following ways: ■ Sends command messages to the components ■ Receives response messages from the components
Nios II controller	Qsys	The design uses another Nios II processor to control the system in the following ways: ■ Reacts to interrupts from the clocked video input triggered by changes in the input video format ■ Configures the IP cores through their register maps

Table 5 describes the video output blocks. The video output reads the upscaled output video from external memory and feeds it to four SDI outputs.

Table 5. Video Control Blocks

Block	Source	Description
Frame reader	Video and image processing suite	The frame reader reads frames from external memory and converts them into Avalon-ST Video packets.
Gamma corrector	Video and image processing suite	The gamma corrector corrects any out of range values that occur as a result of scaling. This block brings the color values back into the acceptable SDI range of $64 \leq Y \leq 940$ and $64 \leq Cb/Cr \leq 960$.
Clocked video output	Video and image processing suite	In this reference design the clocked video outputs convert Avalon-ST Video to a format the SDI IP core can take in.

The DDR3 SDRAM Controller with ALTMEMPHY and the multiport front end perform the following actions:

- Perform the buffering of video to and from external DDR3 SDRAM.
- Handle the arbitration of multiple packet writers and frame reader masters on the single slave interface of the DDR3 SDRAM Controller with ALTMEMPHY.

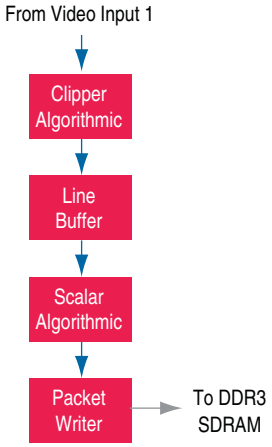
Video Pipelines

The reference design consists of four video pipelines that can each process 1080p60 video. The design splits the 4K upscale processing across the four video pipelines. Extra video processing pipelines can be added to process higher resolutions—the only limit is the size of the FPGA used and the DDR3 SDRAM bandwidth that is available. Conversely, the number of video processing pipelines required is reduced as the achievable f_{MAX} increases. For example, doubling the f_{MAX} results in a halving the number of video processing pipelines required.

You can parameterize each video pipeline, which consists of a chain of processing functions. Avalon-ST interfaces allow you to connect any number of processing functions with Qsys. Figure 2 shows a video pipeline, which has the following features:

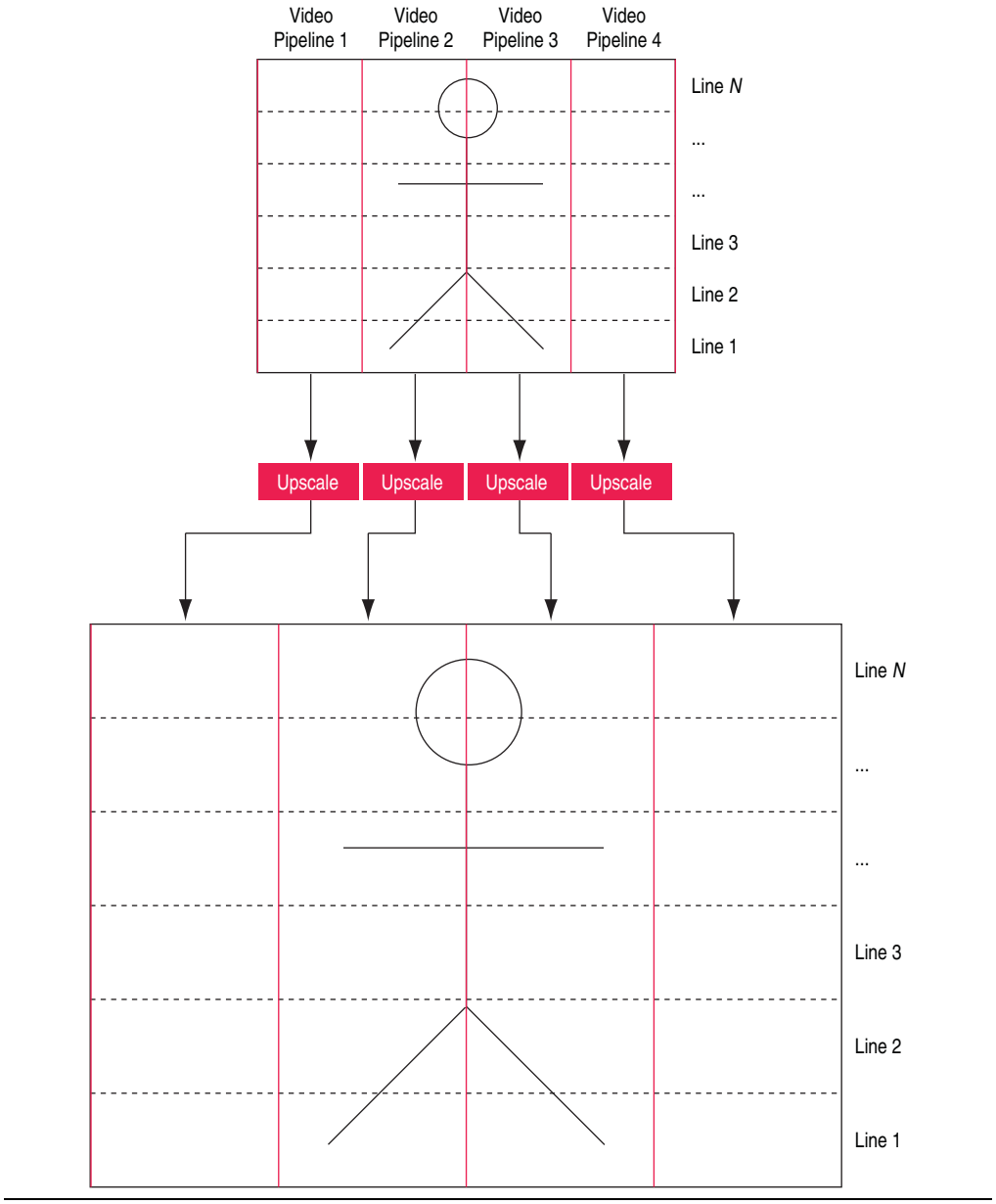
- A clipper algorithmic IP core selects the part of the input line that this video pipeline processes
- A line buffer to buffer multiple input lines and provide the scaler algorithmic IP core with the correct pixel kernels to produce the required output lines
- A scaler algorithmic IP core performs the upscale and produces the output lines
- A packet writer writes the output lines into DDR3 SDRAM

Figure 2. Video Pipeline Block Diagram



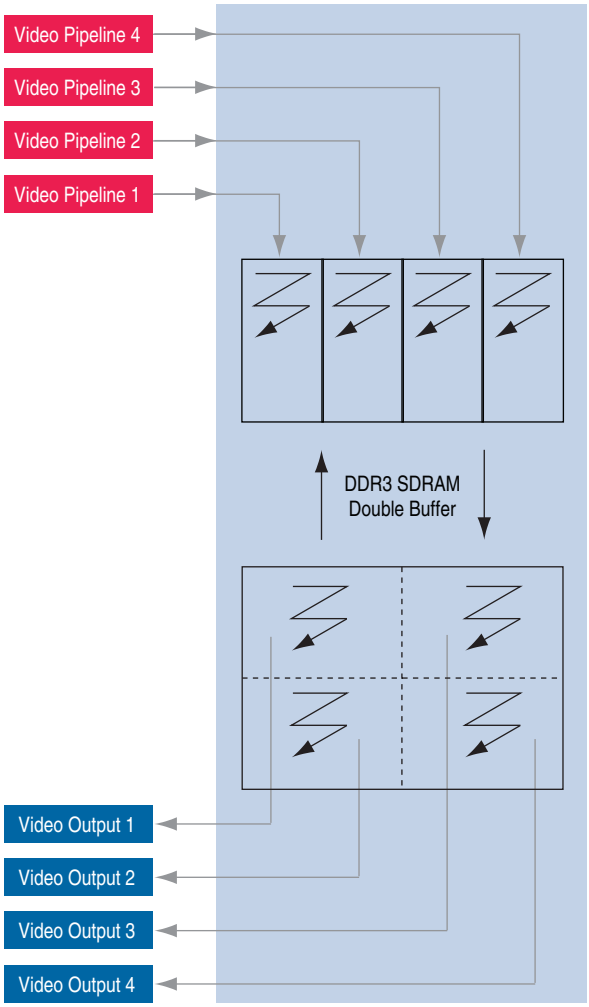
Vertical bands split the incoming video across the four pipelines. Each video pipeline processes a different vertical band of the incoming video frame. Figure 3 shows that each pipeline selects a different quarter of the line (with a small overlap) and upscales only that portion of the video.

Figure 3. A Video Frame Split Over The Video Processing Pipelines



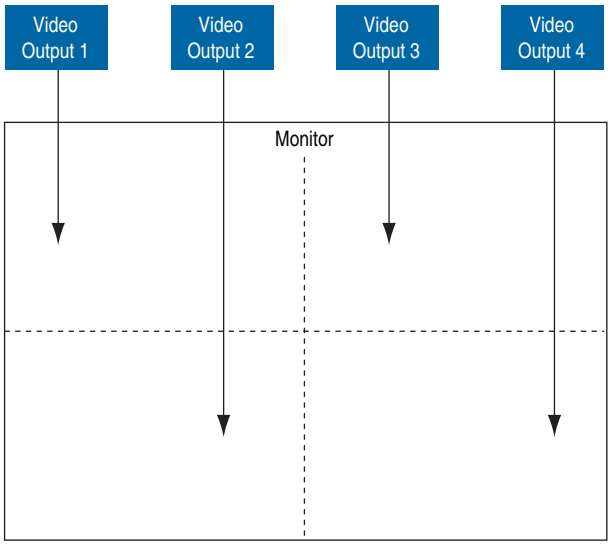
To simultaneously produce the required four SDI outputs, the design must write the 4K frame into a double buffer. Figure 4 shows buffering of the 4K frame. The design writes the 4K frame as vertical bands, then swaps the double buffer, reads out the frame as four quadrants, and sends each to a separate SDI output.

Figure 4. Buffering of 4K Video Frame in DDR3 SDRAM



The design then recombines the upscaled output lines, removing any overlap, to produce the 4K frame (Figure 5).

Figure 5. 4K Frame



Control Interfaces

The reference design introduces the video and image processing component library, which allows greater flexibility and control in video processing designs. The component library is a collection of common video function building blocks that build the video and image processing suite with IP cores. When you use the component library as part of a MegaCore function, the software hides the control and parameterization of the components. In the reference design, a selection of components (the clipper algorithmic IP core, line buffer, scaler algorithmic IP core, and the packet writer) create a flexible video processing pipeline. Each component has a command interface through which it receives messages from a scheduler instructing it to perform a specific function. Some example functions for a line buffer include: receive a new line, shift the lines it contains, or send a kernel of pixels to another component. You can implement the scheduler as an HDL state machine that controls a handful of components, or a CPU that runs software to control large systems of components. The reference design scheduler is a Nios II processor.

For each line the video input bridge receives, it sends a response message to the scheduler. The scheduler then sends out a message to the video input bridge, which instructs it to send the line to a particular destination. In this case, the design sends the line a duplicator, which sends copies of the line to multiple destinations. The scheduler also sends messages to each of the components in the video pipelines. The messages instruct the components which functions to perform on the line.

A software scheduler provides increased flexibility in both system debugging and modifying run-time functionality, to give a more productive design cycle and greatly reduced time-to-market.

Software Schedule

The **main_core.cpp** file contains the software schedule, comments, and a detailed description of the schedule that the Nios II scheduler uses. A number of C macros, in the **alt_vip_beta_nios_ii_miu_api.h** and **alt_vip_beta_nios_ii_miu_regs.h** files file use the message interface unit to allow the Nios II processor to send and receive messages. The macros translate to simple memory-mapped reads or writes and describe the API to allow you to use the Nios II message interface unit.

For more details on the Nios II message interface unit, refer to the *Video and Image Processing Component Library Functional Description* (available from Altera).

Each component in the reference design also has a set of commands that it accepts and a set of responses that it returns. The **alt_vip_common_pkg.h** file lists the commands.

The **main_top.cpp** file contains the top-level control software that monitors and configures the input and output video interfaces. The Nios II controller switches the read side of the video double buffer. The top-level control software contains detailed comments that describe its operation.

Clocks

Table 6 lists the clocks and frequencies.

Table 6. Clocks and Frequencies

Clock Domain	f_{MAX} (MHz)	Description
`sdi_rx_clk[0]`	148.5	The SDI input clock.
`sdi_clk148`	148.5	The SDI output clock for the four outputs.
`vip_clk`	148.5	The video processing pipelines clock.
`altmemddr_0_sysclk`	200.0	The local interface of the memory controller clock.
DDR3 clock	400.0	The DDR3 SDRAM is clocked at 400 MHz

Getting Started

This section describes the following topics:

- Hardware and Software Requirements
- Downloading and Installing the Reference Design
- Generating the Qsys System
- Compiling the Software
- Compiling the Design
- Programming a Device

Hardware and Software Requirements

The reference design requires the following hardware:

- Stratix IV GX FPGA development board

- Two Terasic Transceiver SDI High-Speed Mezzanine Card (HSMC) boards
- A 1080p60 SDI video source
- Four 1080p60 SDI monitors or DVI monitors with SDI-to-DVI converters

The reference design requires the following software:

- Quartus II software v10.1
- Nios II EDS v10.1

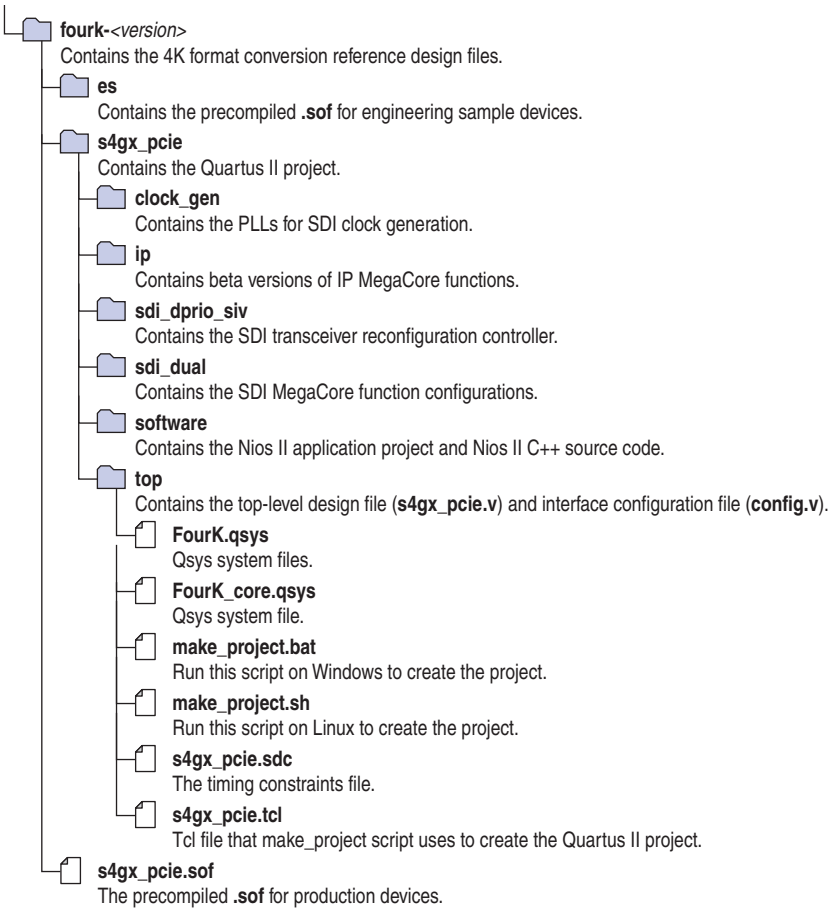
Downloading and Installing the Reference Design

To download and install the reference design, follow these steps:

1. Request the reference design (.zip) files from the 4K Format Conversion Reference Design web page.
2. Extract the contents of the archive file to a directory on your computer. Do not use spaces in the directory path name.

Figure 6 shows the reference design directory structure.

Figure 6. Directory Structure



Generating the Qsys System

To generate the Qsys system, follow these steps:

1. Create the Quartus II project file **s4gx_pcie.qpf**:
   - On Windows operating systems, run the **make_project.bat** script from the Nios II EDS Command Shell by typing `./make_project.bat`.
   - On Linux operating systems, run the **make_project.sh** script.
2. Open the Quartus II software and open the project file **s4gx_pcie.qpf**.
3. On the Tools menu click **Qsys**.
4. Open the **FourK.qsys** system.

   The reference design includes two Qsys systems. The **FourK_core.qsys** system contains the four video scaling pipelines and the components required to schedule them. The **FourK.qsys** system includes one instance of the **FourK_core.qsys** system.

5. In Qsys, click the **Generation** tab.
6. Click **Generate**.
7. When the **System generation was successful** message displays, start the Nios II Software Build Tools for Eclipse: on the Windows Start menu, point to Altera, click Nios II EDS *<version>* and click **Nios II Software Build Tools for Eclipse**.

Compiling the Software

This section describes how to create the **FourK_top_cpu_memory.hex** and **FourK_FourK_core_inst_control_memory.hex** files in the Nios II Software Build Tools for Eclipse. To create the **FourK_top_cpu_memory.hex** file, follow these steps:

1. In the **Workspace Launcher** window click **Browse...** and create a new workspace directory, **workspace**, in the project **s4gx_pcie** directory. Then click **OK** to open the workspace.
2. In the **Nios II – Eclipse** window, right-click in the **Project Explorer** tab, point to **New** and click **Nios II Application and BSP from Template**.
3. In the **Nios II Application and BSP from Template** window fill in the following information:
   - For **SOPC Information File Name**, browse to locate the **FourK.sopcinfo** file.
   - For **CPU name**, select **top_cpu**.
   - For **Project name**, enter `s4gx_pcie_controller`.
   - For **Templates**, select **Blank Project**.
4. Click **Finish**.
5. On the **Project Explorer** tab, right-click on **s4gx_pcie_controller_bsp**, point to **Nios II**, and click **BSP Editor**.
6. On the **Main** tab for **stdout** select **lcd**. Turn on **enable_interrupt_stack** and for **interrupt_stack_memory_region_name** select **top_cpu_memory**.

7. On the **Linker Script** tab for **.bss**, **.heap**, **.rodata**, **.rwdata**, **.stack** and **.text** set the **Linker Region Name** to **top_cpu_memory**. For **reset** set the **Memory Device Name** to **top_cpu_memory**.
8. Click **Generate**.
9. In the **Nios II – Eclipse** window, expand **s4gx_pcie_controller** to open the list of files. Right-click on **main_top.cpp** and **video_standard.cpp** and click **Add to Nios II Build**. Collapse the list of files.
10. In the **Project Explorer** tab, right-click on **s4gx_pcie_controller** and click **Properties**.
11. In the **Properties for s4gx_pcie_controller** window, select **Nios II Application Properties** and change the **Optimization level:** to **Level 3**. Click **OK**.
12. In the **Project Explorer** tab, right-click on the **s4gx_pcie_controller_bsp**, and select **Properties**.
13. In the **Properties for s4gx_pcie_controller_bsp** window, select **Nios II BSP Properties** and change the **Optimization level:** to **Level 3**. Click **OK**.
14. On the **Project Explorer** tab, right-click on the **s4gx_pcie_controller**, and click **Build Project**.
15. On the **Project Explorer** tab, right-click on the **s4gx_pcie_controller**, point to **Make Targets** and click **Build...**.
16. In the **Make Targets** window, select **mem_init_generate** and then click **Build**.
17. The software creates the **FourK_top_cpu_memory.hex** file.

To create the **FourK_FourK_core_inst_control_memory.hex** file, follow these steps:

1. In the **Nios II – Eclipse** window, right-click in the **Project Explorer** tab, point to **New** and click **Nios II Application and BSP from Template**.
2. In the **Nios II Application and BSP from Template** window fill in the following information:
   - For **SOPC Information File Name**, browse to locate the **FourK.sopcinfo** file.
   - For **CPU name**, select **FourK_core_inst_control_cpu**.
   - For **Project name**, enter `core_controller`.
   - For **Templates**, select **Blank Project**.
3. Click **Finish**.
4. In the **Nios II - Eclipse** window, expand **core_controller**, to open the list of files. Right-click on **main_core.cpp** and click **Add to Nios II Build**.
5. On the **Project Explorer** tab, right-click on **core_controller** and click **Properties**.
6. In the **Properties** window, select **Nios II Application Properties** and change **Optimization level:** to **Level 3**. Click **OK**.
7. On the Project Explorer tab, right-click on **core_controller_bsp**, and click **Properties**.
8. In the **Properties** window, select **Nios II BSP Properties** and change **Optimization level:** to **Level 3**. Click **OK**.

9. On the **Project Explorer** tab, right-click on **core_controller**, and click **Build Project**.
10. On the **Project Explorer** tab, right-click on **core_controller**, point to **Make Targets** and click **Build...**.
11. In the **Make Targets** window, select **mem_init_generate** and click **Build**.
12. The software creates the **FourK_FourK_core_inst_control_memory.hex** file.

Compiling the Design

To compile the design in the Quartus II software and create the **s4gx_pcie.sof** file, follow these steps:

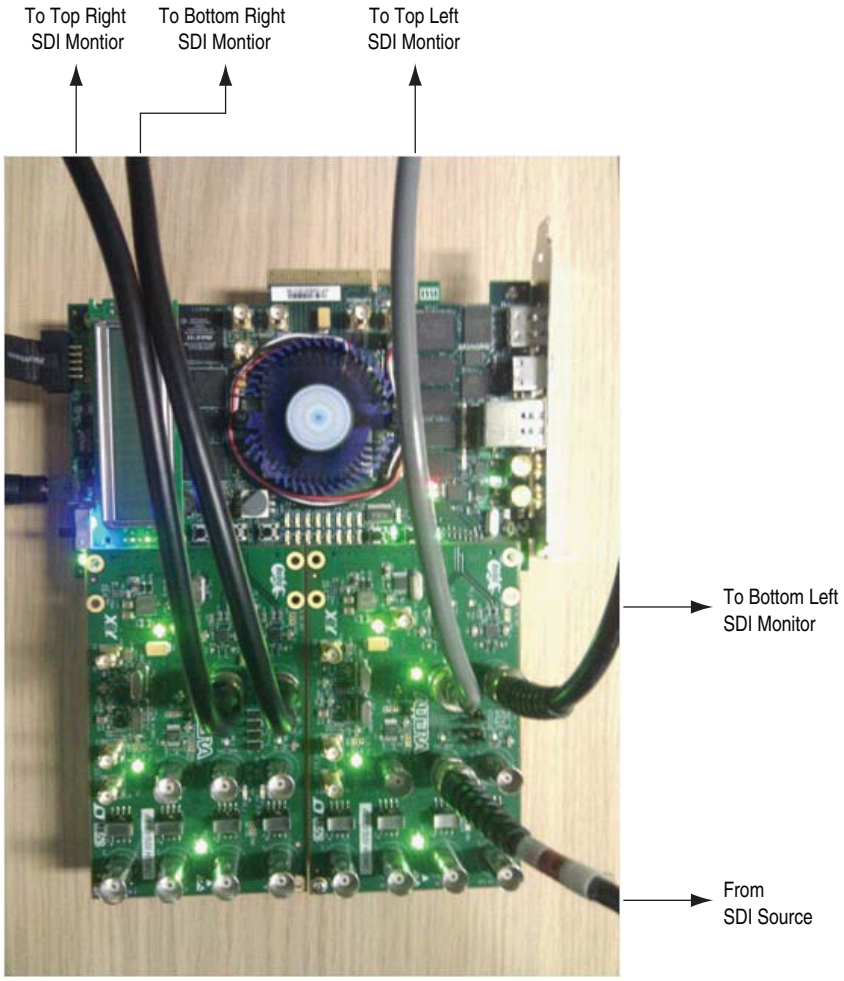
1. On the Tools menu, click **Start Compilation**.
2. When compilation completes, the design creates the **s4gx_pcie.sof** file.

Programming a Device

To program the FPGA and set up the reference design, follow these steps:

1. Connect the two SDI HSMC boards as Figure 7 shows.

Figure 7. Reference Design Setup



2. Connect the four SDI monitor cables to the `SDI_OUT1` and `SDI_OUT2` outputs.
3. Turn on the Stratix IV GX FPGA development board.
4. In the Quartus II software, on the Tools menu, click **Programmer**, to program the FPGA with the **s4gx_pcie.sof** file.
5. Check that LED0 flashes.
6. Connect your SDI source cable to the `SDI_IN1` input. When you connect an input, the LCD displays the detected format.
7. Press push button 0 (PB0) to enable or disable the edge adaptive scaling mode within the scaler algorithmic IP cores.
8. Press push button 1 (PB1) to increase the edge threshold that the edge adaptive scaling algorithm uses.
9. Press push button 2 (PB2) to decrease the edge threshold.

Table 7 describes the Stratix IV GX FPGA Development Board LEDs.

Table 7. LEDs

LED	Description
0	Software heartbeat. Flashes when the software is running on the Nios II processor.
1	Illuminates when the output runs.
2	Illuminates when the SDI_IN1 input detects a supported input format.
3	Illuminates when Illuminates when the input receives overflow.
4	Illuminates when the output transmits underflow.
5	60 frames per second heartbeat. Flashes every 60 frames of the input video.
6	Illuminates when the output is generator locked to the input.
7	Illuminates when you enables edge adaptive scaling.

Document Revision History

Table 8 shows the revision history for this document.

Table 8. Document Revision History

Date	Version	Changes
August 2012	2.0	Updated for Qsys and the Quartus II software v12.0.
May 2011	1.0	Initial release.