Multioutput Scaler Reference Design

AN-648-1.0 Application Note

This application note describes the Altera® Multioutput Scaler Reference Design. Scaling an input video stream to multiple output resolutions is common in many video conferencing and studio multiviewer products. Dedicating a full scaling engine, such as the Altera Scaler II MegaCore® function, to each output resolution can lead to inefficient solutions, because the design can share the video line buffers that each IP core require across all the scaling engines. Depending on the output resolutions, the design may time-division multiplex the algorithmic IP core of a single scaling engine to produce multiple output resolutions.

The Multioutput Scaler reference design demonstrates how to perform the following actions:

- Combine IP cores from the video and image processing component library to create flexible scaling solutions.
- Share line buffers across multiple algorithmic functions.
- Use time-division multiplexed (TDM) algorithmic functions across multiple outputs.

Features

The reference design offers the following features:

- One 3G-serial digital interface (3G-SDI) 1080p60 input.
- One 3G-SDI 1080p60 output containing up to five output resolutions mixed over a test pattern base layer.
- Three scaler algorithmic IP cores:
  - One with four horizontal and vertical taps for upscale only
  - One with 12 horizontal and vertical taps for upscale and up to three-times downscale
  - One with 16 horizontal and vertical taps for upscale and up to four-times downscale
- System initialization and run-time configuration in software.
- Rapid system capture and design with Qsys, the Quartus® II software, and the Nios® II development environments.

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

- Highly parameterizable and modular hardware functions:
  - Video and Image Processing Suite MegaCore functions
  - Nios II processor
  - Clear text system-configuration software
  - DDR 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.
- Files to run the design on the Altera Stratix IV GX FPGA Development Kit with one Terasic Transceiver SDI High-Speed Mezzanine Card (HSMC) board.

For more information about the video and image processing component library and the Avalon-ST Message protocol, refer to *AN654: Video and Image Processing Component Library*.

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 multioutput scaler reference design performs the following actions:

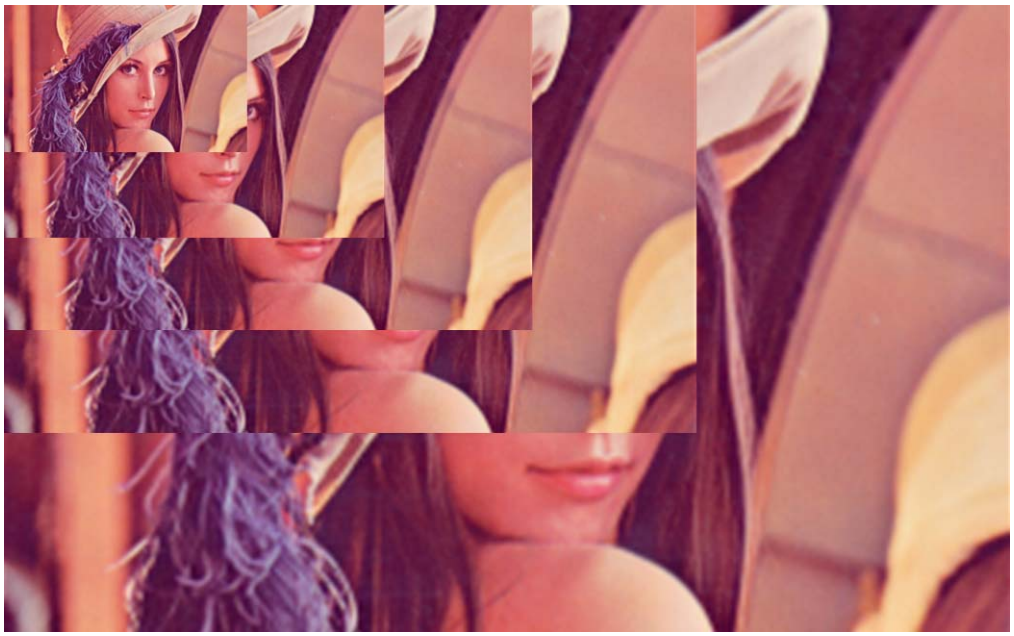
- Takes a 1080p60 or 720p60 input over a 3G-SDI
- Scales the input to up to five output resolutions
- Mixes the five output resolutions over a 1080p test pattern base layer
- Outputs the result over a 3G-SDI

The Altera SDI IP core supports the 3G-SDIs in the FPGA. The reference design has the following five output resolutions:

- 1920 × 1080
- Pass-through of input
- Multimode—cycling between 1920 × 1080, 1280 × 720, 854 × 480 and 1/4 input resolution
- 1/2 input resolution
- 1/3 input resolution

You use push button 1 (PB1) on the Stratix IV GX FPGA development board to select the current resolution for the multimode output. PB0 allows you to enable and disable layers of the mixer to turn each output resolution on and off. Figure 1 shows an example output from the reference design with a 720p version of Lena as the input.

Figure 1. Reference Design Output



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 is a collection of common video function building blocks that 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 (Part 1 of 2)

Usage	ALUTs	Logic Registers	Logic Utilization	Total Blocks				DSP Block 18-Bit Elements
				Memory Bits	Memory Implementation Bits	M9K	M144K	
On device	29,230	38,700	47,949	2,115,714	4,386,816	268	13	132

Table 1. Resource Usage (Part 2 of 2)

Usage	ALUTs	Logic Registers	Logic Utilization	Total Blocks				DSP Block 18-Bit Elements
				Memory Bits	Memory Implementation Bits	M9K	M144K	
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	16%	21%	26%	14%	30%	22%	59%	10%

Functional Description

Figure 2 on page 5 shows a block diagram of the reference design.

Figure 2. Block Diagram

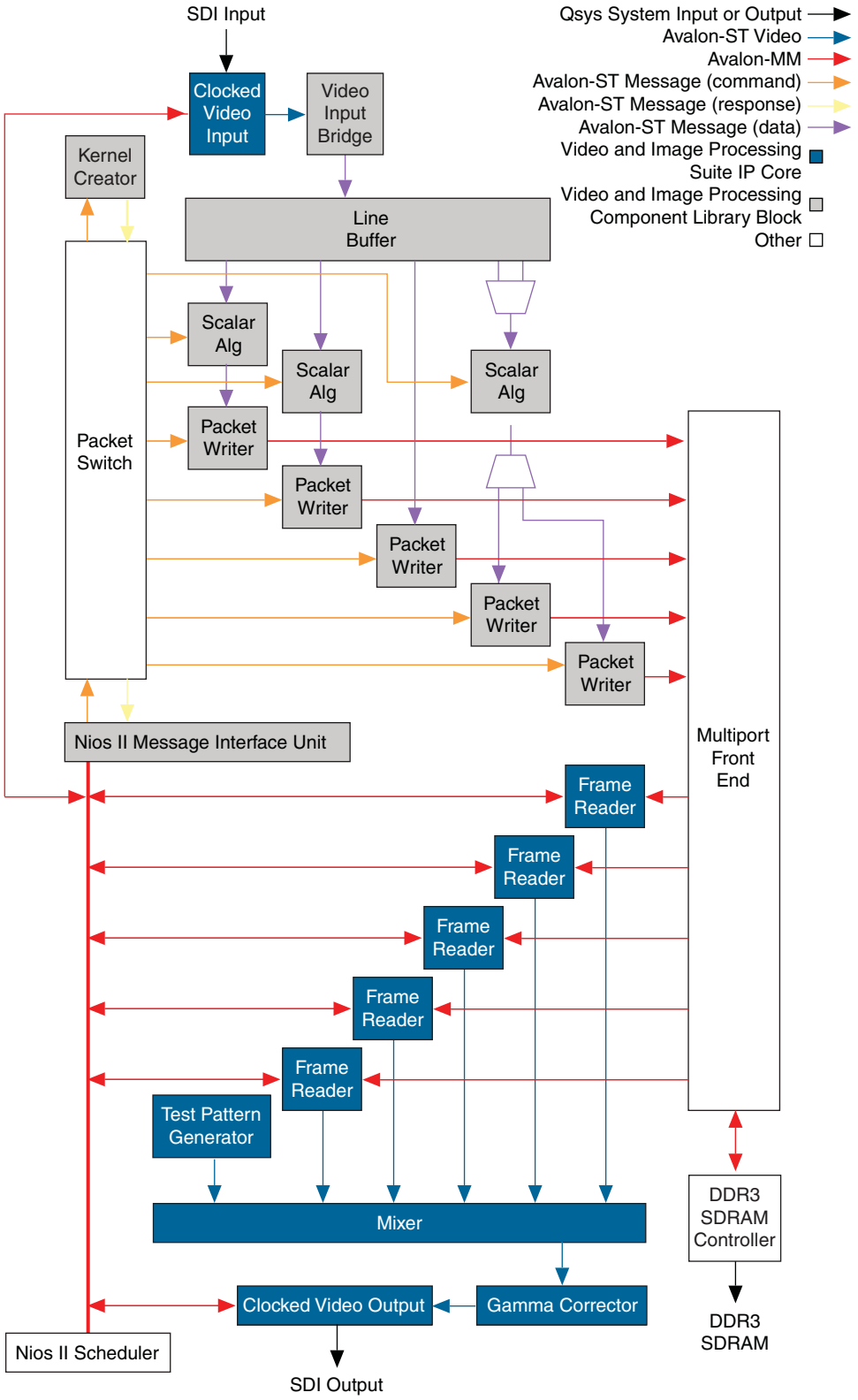


Table 2 describes the video input blocks. The video input takes a single SDI input, checks that it is in a supported format, and then sends it to the line buffer.

Table 2. Video Input Blocks

Block	Source	Description
Clocked video input	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.

Table 3 describes the video pipeline blocks. The video pipeline takes the input video, produces the five output resolutions, and writes them to memory.

Table 3. Video Pipeline Blocks

Block	Source	Description
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. The design may send each output packet simultaneously to multiple destinations through multiple outputs. This reference design configures the line buffer to store 16 lines of input video and has five outputs.
Scaler algorithmic	Video and image processing component library	The scaler algorithmic IP core upscales or downscales the input line by a specified factor.
Packet writer	Video and image processing component library	The packet writer writes the lines of output video frame into external memory.

Table 4 describes the video control blocks. The video control blocks direct the actions of the video pipeline blocks—they configure, start and stop the video and image processing suite IP cores, and schedule the actions of the component library cores. 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
Kernel creator	Video and image processing component library	The kernel creator is a hardware accelerator block that returns the input lines required to produce each output line. The scheduler uses this component 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.
Nios II scheduler	Qsys	A Nios II processor controls 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 ■ Sends command messages to the components ■ Receives response messages from the components

Table 5 describes the video output blocks. The video output reads the five output resolutions from external memory, mixes the outputs over a test pattern base layer, and sends the resulting video to the SDI output.

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.
Mixer	Video and image processing suite	The mixer provides picture-in-picture mixing, combining the five output resolutions to form a single 1080p video output. Runtime control allows each layer (output resolution) to be enabled (displayed) or disabled. The design uses a1080p test pattern as a base layer for cases when the 1080p output resolution layer is disabled.
Test pattern generator	Video and image processing suite	The test pattern generator continually outputs video frames of fixed colour bars.
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 UniPHY and the multiport front end perform the following actions:

- Buffer the video to and from external DDR3 SDRAM.
- Arbitrate multiple packet writers and frame reader masters on the single slave interface of the DDR3 SDRAM Controller with UniPHY.

Video Pipeline

The Multioutput Scaler Reference Design has a single video line buffer component that feeds three scaler algorithmic IP cores. This reference design allows you to scale a single input video stream to (up to) five output resolutions simultaneously—though one output resolution is a nonscaled direct feed through of the input. The design mixes the scaled video data to form a single 1080p60 output video stream, with each of the five output resolutions buffered on a per frame basis in offchip DDR3 SDRAM before mixing. The design mixes the output resolutions to form a single video stream

for demonstrations although the design can output each resolution independently as separate streams. The frame buffering (double buffering) prevents the underflow or overflow issues that occur when mixing smaller and larger resolutions in a single output. Without mixing, you may omit the double buffering, with the clocked video input and clocked video output IP cores typically only requiring a single line of buffering.

Each scaler algorithmic IP core processes a set of input video lines, producing a single output line for each input set. For each scaler algorithmic IP core, you can upscale or downscale the output line, or set the output line to the same length as the lines in the input set. Each scaler algorithmic IP core is clocked at 150 MHz and can scale input data up to 1080p (the input and output frame rates are fixed at 60 fps). The scaler algorithmic IP cores run at up to 300 MHz on Stratix IV devices, supporting larger output resolutions or higher frame rates.

The number of horizontal and vertical taps that the polyphase algorithm uses is parameterizable and determines the maximum downscale factor for each scaler algorithmic IP core. Altera recommends that, for a Lanczos two-coefficient set, the number of taps should be four times the downscale factor. Two of the scaler algorithmic IP cores have 16 and 12 taps respectively (both horizontal and vertical), allowing downscale factors of four and three. The other scaler algorithmic IP core has 4 taps and is only recommended for upscale or passthrough. This design includes three scaler algorithmic IP cores with 4, 12 and 16 taps respectively, but you may modify the design to include more scaler algorithmic IP cores with varying numbers of taps. You can configure the video line buffer with up to 16 outputs, allowing up to 16 scaler algorithmic IP cores to share a single line buffer. You can configure each scaler algorithmic IP core to have up to 64 horizontal or vertical taps, allowing downscale factors of 16.

Output Resolutions

The Multioutput Scaler Reference Design time-division multiplexes the 12-tap scaler algorithmic IP core to produce the following output resolutions:

- 1/2 the input resolution
- 1/3 the input resolution

The design time-division multiplexes per line, with the scaler algorithmic IP core switching between output lines for the two resolutions as directed by top-level scheduling. This design shares the scaler algorithmic IP core across two output video streams. However, you can share one algorithmic IP core across three or more output resolutions, if the sum of the output lines is less than or equal to the number of input lines.

Altera configures the Multioutput Scaler Reference Design to detect 1080p60 or 720p60 inputs. When the design detects a 1080p60 or 720p60 input the design produces a 1080p60 output with up to five output resolutions mixed over a 1080p test pattern base layer. For any other input format, the design disables the output. To support further input formats, you can modify the control software, which runs on the Nios II processor.

The control software configures the 4-tap scaler algorithmic IP core to always produce a 1920 × 1080 output. The 12-tap scaler algorithmic IP core produces one output with 1/2 vertical and horizontal input resolution; and a second output with 1/3 vertical and horizontal input resolution. The control software allows you to switch the output resolution of the 16-tap scaler algorithmic IP core between the following four modes:

- 1920 × 1080
- 1280 × 720
- 854 × 480
- Horizontal and vertical input resolution

You can switch between output modes and the software also allows you to enable and disable mixer layers. Each mixer layer is one output resolution. At reset, the design enables all five output resolutions. You can disable a mixer layer until only the base test pattern remains or re-enable the layers in reverse order.

Control Interfaces

The reference design uses IP cores from the video and image processing suite and components from the video and image processing component library. The design uses the video and image processing IP cores on the output side of the design—reading data from external DDR3 SDRAM and mixing the five output resolutions to form a single output stream. Each video and image processing IP core is a stand-alone block that processes video data on a frame-by-frame basis.

The design embeds the main control logic within each IP core. After the Nios II scheduler configures and starts the IP cores through its Avalon-MM slave interfaces, they then continually process data with no intervention.

The components from the video and image processing component library process data line-by-line. The logic for the fine grain intra-line control is embedded within each component. However, an external scheduler must send commands across an Avalon-MM message interface to drive the line-by-line actions. The Nios II scheduler provides the line-by-line control for the component library IP cores. The Nios II scheduler executes a software schedule (plain text C code) that drives commands and receives responses through its Avalon-MM master interface. The message interface unit converts your commands from the Avalon-MM standard to Avalon-ST message interface and routes them through the packet switch to the correct component. The packet switch routes responses from the component blocks back to the message interface unit, which converts them to Avalon-MM signals for the Nios II scheduler to read.

Nios II Scheduler

The Nios II scheduler is a Nios II processor that runs the **main.cpp** file, which contains the software for the scheduler with comments to provide a full description. A number of C macros, in the **nios_miu.h** 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 application programming interface (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).

A loop encloses the main body of the Nios II scheduler code. The loop executes once for each input line or each output line of the largest output resolution, whichever is larger. As the video input bridge starts each new video line, it sends a response to the Nios II scheduler to indicate that a new line is waiting. The Nios II scheduler uses speculative execution to create the commands one line in advance rather than waiting for each new input line to decide which commands it should generate and send. The scheduler pushes the commands that it must send when the next input line arrives into a FIFO buffer in the message interface unit. The message interface unit holds the commands until it receives the new line response. When the scheduler receives the expected new line response, it sends a further command to the message interface unit to send the FIFO contents. If the next response is unexpected the scheduler can instruct the message interface unit to discard the FIFO contents and not send commands. The speculative generation of commands minimizes the delay in processing the input video data, which fills the input FIFO buffer in the clocked video input IP core until the scheduler send the commands.

For each new line response, the scheduler sends a command to the video input bridge instructing it to send the data to the line buffer. The scheduler commands the line buffer to shift its current stored lines through the kernel by one line and receive the new input line. If the new line completes the kernel required for the next output line of any scaler algorithmic IP core, the scheduler also commands the line buffer to send the new kernel through the appropriate outputs. The scheduler commands the appropriate scaler algorithmic IP cores to process the kernel with the correct scaling factor. The scheduler commands the appropriate packet writers to write the resulting line to memory. The design uses the kernel creator as a hardware accelerator to determine which input line should form the center of the kernel for each output line of each output resolution.

At the end of the loop iteration for the new input line response, the loop may run again if any scaler algorithmic IP core is upscaling and needs to reuse the same kernel for another output line. In this case, the scheduler sends no command to the video input bridge and the scheduler instructs the line buffer to send the existing kernel without receiving any new data.

Line Buffer Details

You can parameterize the number of outputs from the video line buffer, the kernel size (number of lines) for each output, and the offset for each output relative to the oldest line stored in the buffer. For a line buffer with a total of N lines of storage, the design defines line 0 as the oldest line in the buffer and line $(N – 1)$ as the newest. The design shares the memory that stores the video data across all the outputs, with each output selecting the required lines from the memory read output. The design constrains output 0 to use line 0. You can set all other enabled outputs to any start line greater than or equal to 0. The design defines the total size of the line buffer to the maximum of the line plus kernel size across all outputs. In this reference design, output 0 feeds the 16-tap scaler algorithmic IP core and has a kernel size of 16 lines. Output 1 feeds the 4-tap scaler algorithmic IP core and has a kernel size of 4 lines. Output 2 feeds a packet writer to create the pass-through output and only requires a single line.

Outputs 3 and 4 feed the 12-tap scaler algorithmic IP core with a packet multiplexer component (configured to select an input on a first-come first-served basis) and requires 12 lines. Table 6 shows how the kernels for the outputs are offset.

Table 6. Output Line Buffers

Output	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									x							
1									x							
2									x							
3									x							
4										x						

Table 6 shows the design needs 16 lines of buffering to serve all of the outputs. The 'x' marked for each output denotes the *center line* for that output. For a scaler algorithmic IP core with N vertical taps, an input kernel of N lines is required. The input kernel comprises a *center line*, *floor*$((N-1)/2)$ lines above the *center line* in the input frame and *floor*$(N/2)$ lines below the *center line* in the input frame. The number of the line required as the *center line* by the scaler algorithmic IP core to produce output line x is:

floor$(x \times in\_height / out\_height)$

where:

in_height and *out_height* are the number of lines in the input and output video frames respectively.

To keep the software scheduler for the system simple, the design aligns the kernels for all the outputs (except output 4) to have a *center line* of 7. The scheduler tracks a single *center line* value for the whole line buffer and keeps all the scaler algorithmic IP cores synchronized to the same input line.

Line buffer output 4 uses line 6 as its *center line*, offsetting it towards older lines by one line relative to all the other kernels. This offsetting allows the design to time-division multiplex the 12-tap scaler algorithmic IP core across the 1/2 and 1/3 input resolution outputs. However, the design does not necessarily use output 3 for the 1/2 resolution output and output 4 for the 1/3 resolution output. You can always use output 3 to generate the 1/2 resolution output and, you may use this output to generate the 1/3 resolution output to save any multiplexing. However, in some cases, the same *center line* from the input frame may be required for both output resolutions. You can send the kernel to the scaler algorithmic IP core through output 3 with the correct *center line* to generate the 1/2 resolution output; however, to meet the needs of the input video rate, the kernel may need to be shifted by one line every time you send a kernel. The design now increases the *center line* of the kernel on output 3 by one line and is no longer correct if you try to send the kernel through output 3 to generate the 1/3 resolution output. By including output 4 in the design, with an offset one line closer to line 0, the *center line* for output 4 is now correct and you can use this output to send the kernel. Where the 1/3 resolution output uses a *center line* that the design does not use for the 1/2 resolution output, you can still use output 3 to send the data to the scaler algorithmic IP core. Table 7 and Table 8 show how the design uses outputs 3

and 4 to switch between different input resolutions.

Table 7. Kernel Center Lines for 720p Input

Loop Iteration	Center Line		
	1080p	1/2 Resolution	1/3 Resolution
0	0	0	—
1	0	—	**0**
2	1	—	—
3	2	2	—
4	2	—	—
5	3	—	3
6	4	4	—
7	4	—	—
8	5	—	—
9	6	6	—
10	6	—	**6**

Table 7 shows the following points:

- The *center line* that the 1080p upscale output requires on each pass through the loop in the scheduler code for a 720p input.
- The lines that the 1/2 and 1/3 resolutions require.
- The 1/2 and 1/3 resolutions require the same *center line* for every third input line.

However, the design uses each even number input line twice (that is, the design sends the kernel) by the 4-tap scaler algorithmic IP core to generate the upscaled 1080p output. Hence, you can send the kernel to the 12-tap scaler algorithmic IP core twice through output 3 when required, by delaying the generation of the 1/3 resolution output by one line and avoiding the need to use output 4. Table 7 shows in bold when the 1/3 resolution output uses the second send of the kernel.

Table 8. Kernel Center Lines for 1080p Input (Part 1 of 2)

Loop Iteration	Center Line		
	1080p	1/2 Resolution	1/3 Resolution
0	0	0	—
1	1	—	**0**
2	2	2	—
3	3	—	3
4	4	4	—
5	5	—	—
6	6	6	—
7	7	—	**6**
8	8	8	—

Table 8. Kernel Center Lines for 1080p Input (Continued) (Part 2 of 2)

Loop Iteration	Center Line		
	1080p	1/2 Resolution	1/3 Resolution
9	9	—	9
10	10	10	—

Table 8 shows an example in which the design requires output 4 and uses the same scaler algorithmic IP core to generate the two outputs. The input is 1080p and so the design must shift the kernel by one line every time it is sent, to keep up with the input video rate. The 1/2 and 1/3 output resolutions still clash on their required *center line* on every sixth input line, but the *center line* for the kernel of output 3 (which is the same as the 1080p *center line* in Table 8) shifts by one before the design sends the output 3 kernel for a second time. In these cases (bold in Table 8), you must use output 4. The scheduler code controls the switching between outputs 3 and 4.

Clocks

Table 9 lists the clocks and frequencies.

Table 9. 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.
`vip_clk`	148.5	The video processing pipelines clock.
`uniphy_ddr3_afi_clk`	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
- Terasic Transceiver SDI High-Speed Mezzanine Card (HSMC) board
- A 1080p60 or 720p60 SDI video source

- A 1080p60 SDI monitor or DVI monitor with SDI-to-DVI converter

The reference design requires the following software:

- Quartus II software v11.1
- Nios II EDS v11.1

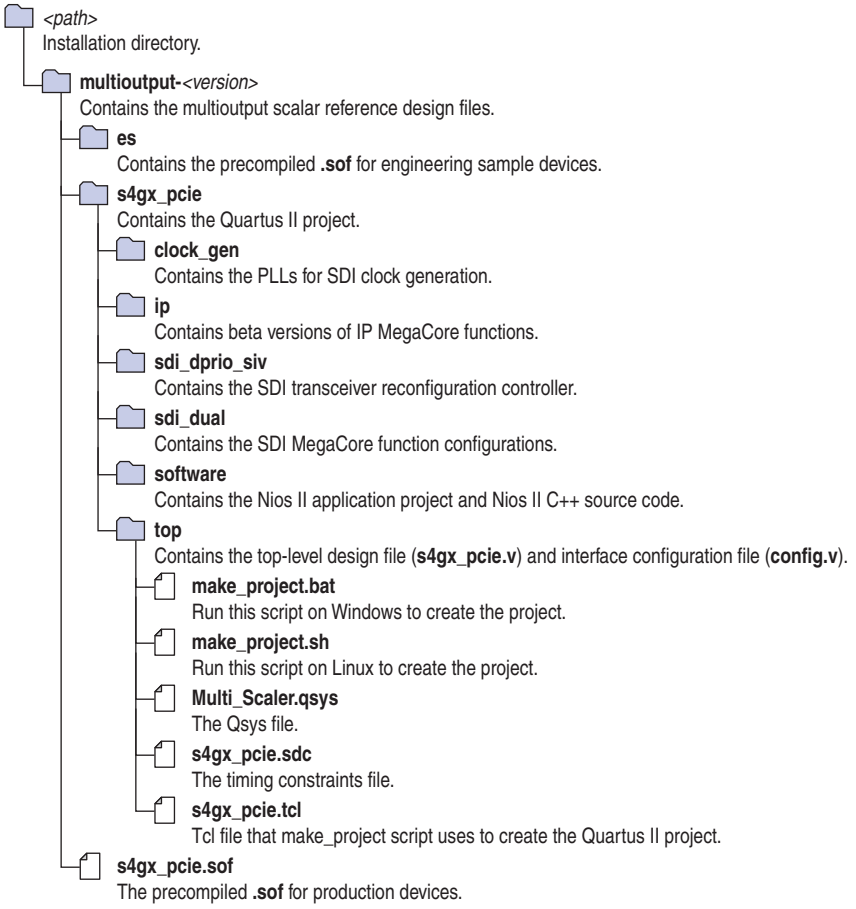
Downloading and Installing the Reference Design

To download and install the reference design, perform the following steps:

1. Request the reference design (.zip) files from the Multioutput Scalar 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 3 shows the reference design directory structure.

Figure 3. Directory Structure



Generating the Qsys System

To generate the Qsys system, perform the following steps:

1. Create the Quartus II project file **s4gx_pcie.qpf**:
   - On Windows operating systems, run the **make_project.bat** script.
   - On Linux operating systems, run the **make_project.sh** script.
2. Open the Quartus II project file **s4gx_pcie.qpf**.
3. Open the Command Prompt and change to the <*Quartus II installation*>**\quartus\sopc_builder\bin** directory.
4. Type the following command to open Qsys in debug mode:

   ```
   qsys-edit --debug
   ```

5. In Qsys, on the File menu, click **Open** and select **Multi_Scaler.qsys**.
6. On the **Component Library** tab, right click on **Library** and click **Show Hidden Components**. Expand **Video and Image Processing**, and expand **Component Library** to see all video and image processing library components.
7. In Qsys, click the **Generation** tab.
8. Click **Generate**.
9. When the system generation is successful, in Qsys, on the Tools menu, click **Nios II Software Build Tools for Eclipse**.

Compiling the Software

To compile the software and create the **onchip_memory2_0.hex** file, in the Nios II Software Build Tools for Eclipse, perform the following 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 select **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 **Multi_Scalar.sopcinfo** file.
   - For **Project name**, enter `s4gx_pcie_controller`.
   - For **Templates**, select **Blank Project**.
4. Click the **Finish** button.
5. In the **Project Explorer** tab, right-click on **s4gx_pcie_controller_bsp**, point to **Nios II**, and select **Generate BSP**.
6. In the **Nios II – Eclipse** window, right-click on the **+** symbol to the left of **s4gx_pcie_controller** to open the list of files. Right-click on the file **main.cpp** and select **Add to Nios II Build**.
7. In the **Project Explorer** tab, right-click on **s4gx_pcie_controller** in the **Project Explorer** tab and select **Properties**.

8. In the **Properties for s4gx_pcie_controller** window, select **Nios II Application Properties** and change the **Optimization level** to **Level 3**. Then click **OK**.
9. In the **Project Explorer** tab, right-click on **s4gx_pcie_controller_bsp** and select **Properties**.
10. In the **Properties for s4gx_pcie_controller_bsp** window, select **Nios II BSP Properties** and change the **Optimization level:** to **Level 3**. Then click **OK**.
11. In the **Project Explorer** tab, right-click on **s4gx_pcie_controller**, and select **Build Project**.
12. In the **Project Explorer** tab, right-click on **s4gx_pcie_controller**, point to **Make Targets** and select **Build...**.
13. In the **Make Targets** window, select **mem_init_generate** and then click **Build**.
14. The design creates the **cpu_memory.hex** file.

Compiling the Design

To compile the design in the Quartus II software and create the **s4gx_pcie.sof** file, on the Tools menu, click **Start Compilation**. When compilation completes, the Quartus II software creates the **s4gx_pcie.sof** file.

Programming a Device

To program the FPGA and set up the reference design, perform the following steps:

1. Connect the SDI HSMC board to `HSMC PORT A` input on the Stratix IV GX FPGA development board
2. Connect the SDI monitor cables to the `SDI_OUT1` output.
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 1080p60 or 720p60 SDI source cable to the `SDI_IN1` input. Various LEDs illuminate or flash (Table 10).

Table 10 describes the Stratix IV GX FPGA development board LEDs.

Table 10. LEDs

LED	Description
0	Software heartbeat. Flashes when the software is running on the Nios II processor.
1	Illuminates when the output is running.
2	Illuminates when the input is locked.
3	Illuminates when the board detects an overflow.
4	Illuminates when the board detects an underflow.

Document Revision History

Table 11 shows the revision history for this document.

Table 11. Document Revision History

Date	Version	Changes
August 2012	1.1	Corrected minor errors.
January 2012	1.0	Initial release.