LatticeCORE

2D Scaler IP Core User's Guide

August 2013
IPUG88_01.2

Table of Contents

Chapter 1. Introduction 4
Quick Facts 4
Features 4
Release Information 5
Chapter 2. Functional Description 6
Key Concepts 6
Block Diagram 6
Algorithm and Supported Filter Kernels 7
Dynamic Parameter Updating 7
Rounding 8
Primary I/O 9
Interface Descriptions 9
Video Input/Output 9
Parameter Register Read/Write Interface 10
Control Signals and Timing 10
Chapter 3. Parameter Settings 14
Architecture 15
Frame Dimensions 15
Filter Physical Characteristics 15
I/O Specification 16
Implementation 17
Chapter 4. IP Core Generation 18
Licensing the IP Core 18
Getting Started 18
Configuring the 2D Scaler IP Core in IPexpress 18
IPexpress-Created Files and Top-Level Directory Structure 20
Instantiating the Core 21
Running Functional Simulation 21
Synthesizing and Implementing the Core in a Top-Level Design 21
Hardware Evaluation 22
Enabling Hardware Evaluation in Diamond 22
Enabling Hardware Evaluation in ispLEVER 22
Updating/Regenerating the IP Core 22
Regenerating an IP Core in Diamond 22
Regenerating an IP Core in ispLEVER 23
Chapter 5. Support Resources 24
Lattice Technical Support 24
Online Forums 24
Telephone Support Hotline 24
E-mail Support 24
Local Support 24
Internet 24
References 24
Revision History 24
Appendix A. Resource Utilization 25
LatticeECP3 Devices 25
Ordering Part Number 25
LatticeECP2M and LatticeECP2MS Devices 25
Ordering Part Number 25

LatticeECP2 and LatticeECP2S Devices 25
Ordering Part Number 26
LatticeXP2 Devices 26
Ordering Part Number 26

Introduction

The 2D Scaler IP core converts input video frames of one size to output video frames of a different size. Its flexible architecture supports a wide variety of scaling algorithms. The highly configurable design takes advantage of the embedded DSP blocks available in Lattice FPGAs. A simple I/O handshake makes the core suitable for either streaming video or bursty input video data. In-system input and output frame sizes updating is possible on a frame basis.

Quick Facts

Table 1-1 gives quick facts about the 2D Scaler IP core.

Table 1-1. 2D Scaler IP Core Quick Facts

		2D Scaler IP Core Configuration					
		480P to 720P (YCbCr4:2:2, serial)		**720P to 480P** (YCbCr4:2:2, parallel)		**720P to 1080P** (RGB, parallel, dynamic)	
Core Requirements	FPGA Families Supported	LatticeECP3™, LatticeECP2M™, LatticeECP2MS, LatticeECP2™, LatticeECP2S, LatticeXP2™					
	Minimum Device Required	LFE2-6E	LFE3-17EA	LFE2-6E	LFE3-17EA	LFE2-12E	LFE3-17EA
Resource Utilization	Targeted Device	LFE2-20E-7F484C	LFE3-17EA-8FN484C	LFE2-20E-7F484C	LFE3-17EA-8FN484C	LFE2-20E-7F484C	LFE3-17EA-8FN484C
	Taps	4x4		4x4		4x4	
	Pixel width	8		8		8	
	Coefficient width	9		9		9	
	Registers	953	955	1116	1113	1515	1522
	LUTs	1278	1343	1267	1338	1715	1829
	EBRs	4	4	7	7	9	9
	MULT9x9	8	8	16	16	24	24
Design Tool Support	Lattice Implementation	Lattice Diamond® 1.3 or ispLEVER® 8.1					
	Synthesis	Synopsys® Synplify™ Pro for Lattice E-2011.03L					
	Simulation	Aldec® Active-HDL™ 8.2 Lattice Edition Mentor Graphics® ModelSim™ SE 6.3F					

Features

- Supports single-color, YCbCr 4:2:2, YCbCr 4:4:4 and RGB video formats
- Supports serial and parallel processing
- Dynamic parameter updating
- Supports multi-scaling algorithms
- Configurable number of filter taps for Lanczos coefficient set
- Configurable number of phases for Bicubic, Mitchell and Lanczos coefficient sets
- Configurable pixel data width
- Configurable coefficient width
- Configurable parameter bus width and separate parameter bus clock

- Selectable memory type for line buffer and coefficient memories
- Option for sharing vertical and horizontal filter coefficient memories

Release Information

- 2D Scaler IP Core version 2.0; last updated September 2, 2011

Key Concepts

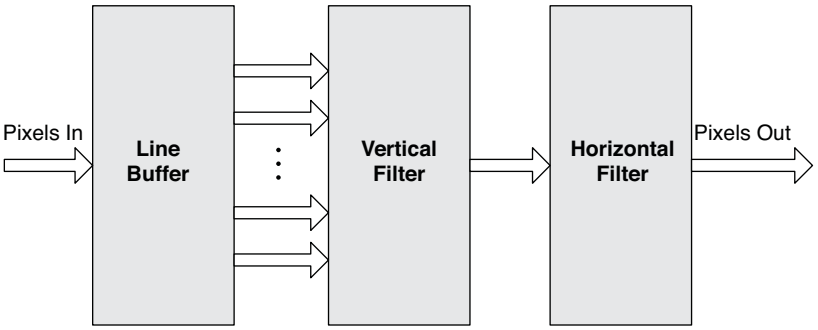
Video scaling is the process of calculating values for the pixels in an output frame of dimensions Xout-by-Yout from the values of pixels in an input frame of dimensions Xin-by-Yin. Scaling up by an integer multiple involves inserting new pixels between the original pixels in the input frame and calculating each new pixel value as a weighted sum of nearby original pixel values. The number of original pixel values and their weights depends on the scaling algorithm employed. In general, including more original pixels in the calculation results in a higher quality result (but requires more FPGA resources).

Conversely, down-scaling by an integer multiple involves dropping unneeded input pixels. Typically, the drop operation is preceded by a two-dimensional low-pass filter to avoid a jagged appearance in the output frame. The low-pass filtering operation is itself a weighted sum of nearby input pixels. The set of weights is referred to as the “filter kernel” or “coefficient set”.

Block Diagram

The high-level architecture of the 2D Scaler IP core is shown in Figure 2-1.

Figure 2-1. 2D Scaler IP Core Block Diagram



The 2D Scaler IP core allows different scaling factors for the horizontal and vertical dimensions. It uses a separable filter architecture which, as depicted by the block diagram, performs the vertical and horizontal scaling in two steps. The input pixel data is first stored in the line buffer. The size of the line buffer is dictated by the number of the vertical filter taps and the maximum input frame width. Pixel data are read out of the line buffer and passed to the vertical filter column by column. Likewise, the vertical filter coefficients are read out of the coefficient memories and passed to the vertical filter for processing along with the pixel data. The row outputs from the vertical filter are then passed to the horizontal filter to generate the output pixels. Output precision control is then performed on the final output pixel value.

The 2D Scaler IP core supports in-system re-programming of the input and output frame sizes via a parameter bus. If the IP core is configured for dynamic parameter updating then the maximum input and output frame resolutions need to be specified so that line buffer and various counters can be configured appropriately. Also the parameter bus can be configured to run on a separate clock. By default, the parameter bus runs on the input pixel sample clock.

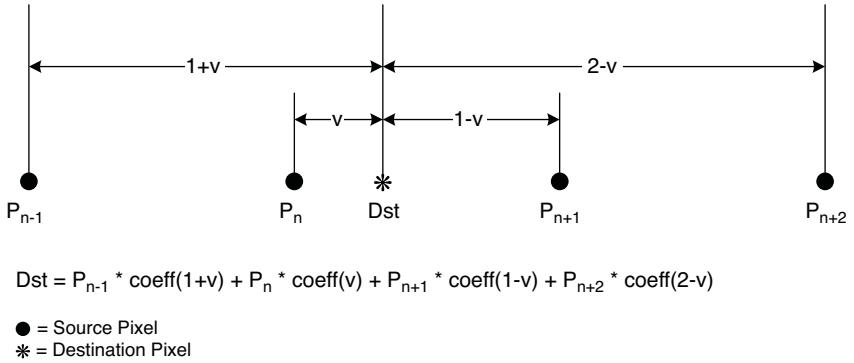
When processing YCbCr 4:2:2 video format, the core averages neighboring pixels’ Cb and Cr vectors to construct YCbCr 4:4:4 format for scaling.

Algorithm and Supported Filter Kernels

Video scaling is a process of generating pixels that do not exist in the original image. In order to compute an output pixel from a set of fixed input pixels, it is necessary to map the output pixel to the input pixel grid and estimate the location of the output pixel relative to the input pixels. The algorithm approximates the output pixel value by using a filter with coefficients weighted accordingly.

A 4-tap Bicubic filter uses four adjacent input pixels to interpolate an output pixel, for example. The spaces between the adjacent pixels are divided into a configurable number of positions called phases so that the output pixel is mapped into one of these positions. The weights of the coefficients are determined by the positions or phases and how close they are to the output pixel. The higher the number of phases, the more accurate is the interpolated output pixel.

Figure 2-2. Scaling Process with 4-Tap Bicubic Filter



The 2D Scaler IP core supports five adaptive scaling kernels: 1-tap nearest neighbor, 2-tap bilinear, 4-tap bicubic, 4-tap Mitchell and configurable multi-tap Lanczos filters. The range for the size of the Lanczos filters is from 4 taps to 12 taps. Filter coefficients are generated at compile time when the kernel is configured.

Dynamic Parameter Updating

The 2D Scaler IP core supports in-system input and output frame sizes updating via a register read/write interface called parameter ports. The parameter registers are listed below.

Address	Registers	Size	R/W	Description
0x0000	FRMWIDTH	32	R/W	**Input frame width register** – The FRMWIDTH value must be the input frame width minus 1. The minimum value is 31, and the maximum value is the maximum input frame width specified on the IP GUI minus 1. The default value is the maximum value. Input frame width must be an even number for YCbCr4:2:2 format (i.e. FRMWIDTH must be odd).
0x0004	FRMHEIGHT	32	R/W	**Input frame height register** – The FRMHEIGHT must be the input frame height minus 1. The minimum value is 31, and the maximum value is the maximum input frame height specified on the IP GUI minus 1. The default value is the maximum value.
0x0008	OUTWIDTH	32	R/W	**Output frame width register** – The OUTWIDTH must be the output frame width minus 1. The minimum value is 31, and the maximum value is the maximum output frame width specified on the IP GUI minus 1. The default value is the maximum value. Output frame width must be an even number for YCbCr4:2:2 format (i.e. OUTWIDTH must be odd).
0x000C	OUTHEIGHT	32	R/W	**Output frame height register** – The OUTHEIGHT must be the output frame height minus 1. The minimum value is 31, and the maximum value is the maximum output frame height specified on the IP GUI minus 1. The default value is the maximum value.

Address	Registers	Size	R/W	Description
0x0010	VSFACTOR	32	R/W	**Vertical scaling factor register** – VSFACTOR = ((FRMHEIGHT+1)*(1<<VFCBPWIDTH))/ (OUTHEIGHT+1). Where VFCBPWIDTH is the maximum value of log2 (maximum output frame height) and log2 (number of vertical filter phases).
0x0014	HSFACTOR	32	R/W	**Horizontal scaling factor register** – HSFACTOR = ((FRMWIDTH+1)*(1<< HFCBPWIDTH))/ (OUTWIDTH+1). Where HFCBPWIDTH is the maximum value of log2 (maximum output frame width) and log2 (number of the horizontal filter phases).
0x0018	UPDATE	32	R/W	**Update parameter enable register** – When the writing of the parameter registers is complete, users should set this register to 1 to enable the parameter's status. The default value is 0. When the core updates the new parameters, it re-sets this register to 0.

The parameter registers can be written to only when the UPDATE register bit is 0. When the UPDATE bit is set to 1, the six parameters inside the core are updated with the new values when the frmsync_in signal is active, indicating a new input frame is arriving. After updating its internal parameters, the core resets the UPDATE bit to 0 to indicate that the parameter registers are now empty and can take on new values.

When dynamic parameter updating is enabled, the user needs to configure the largest input and output frame sizes the system expects to handle. This is so that the line buffer and various counters within the core can be configured properly. The core uses the default values for the input and output frame sizes to start until the driving block updates the parameters in the subsequent frames. The default values are the maximum input frame size and the maximum output frame size.

For most cases, VFCBPWIDTH equals to log2(output frame height) for fixed scaler and log2(maximum output frame height) for dynamic scaler. HFCBPWIDTH equals to log2(output frame width) for fixed scaler and log2 (maximum output frame width) for dynamic scaler. When the number of vertical phases is greater than the maximum output frame height, the VFCBPWIDTH equals to log2(number of the vertical filter phases). When the number of horizontal phases is greater than the maximum output frame width, the HFCBPWIDTH equals to log2(number of horizontal filter phases).

For the nearest neighbor and bilinear kernels, when the VFCBPWIDTH is smaller than the vertical coefficient bit width, its value should be replaced by the vertical coefficient bit width. Similarly, when the HFBPWIDTH is smaller than the horizontal coefficient bit width, its value should be replaced by the horizontal coefficient bit width.

The values of VFCBPWIDTH and HFCBPWIDTH can be found in the generated parameter value file: \<project_dir>\scaler_eval\<username>\rtl\params\params.v

Rounding

The 2D Scaler IP core provides three types of rounding on the output pixel value. This is set at compile time by the parameter Rounding Mode. The values are:

- **Truncation**: discards all bits to the right of the output's least significant bit
- **Normal**: rounds away from zero if the fractional part is exactly one-half
- **Convergent**: rounds to the nearest integer if the fractional part is exactly one-half

Primary I/O

Figure 2-3. 2D Scaler IP Core I/O

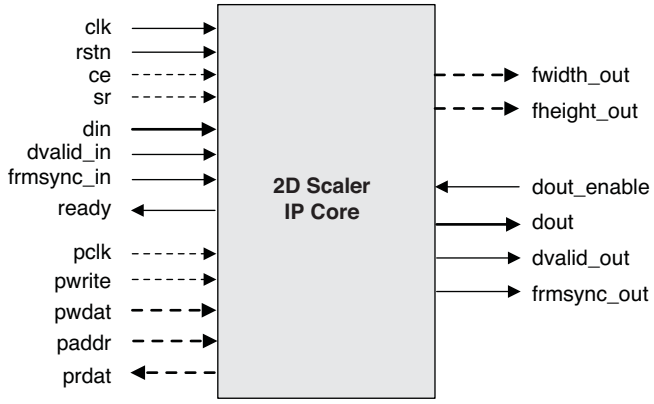


Table 2-1. Primary I/O

Port	Size	I/O	Description
Global Signals			
clk	1	I	System clock
rstn	1	I	System asynchronous active-low reset signal
ce	1	I	Active high clock enable, optional
sr	1	I	Active high synchronous reset, optional
Video Input			
dvalid_in	1	I	Input valid
frmsync_in	1	I	Input frame sync signal, indicating current input pixel is at row 0, column 0
din	8 - 48	I	Pixel data in
ready	1	O	Core is ready for input
Video Output			
dout_enable	1	I	Input from down-stream module to enable output data, active high
dvalid_out	1	O	Output valid
frmsync_out	1	O	Output frame sync signal, indicating current output pixel is at row 0, column 0
dout	8 - 48	O	Pixel data out
fwidth_out	16	O	Current output frame width, optional, only for dynamic mode
fheight_out	16	O	Current output frame height, optional, only for dynamic mode
Parameter Bus (optional)			
pclk	1	I	Parameter bus clock
pwrite	1	I	Parameter bus write enable, active-high
paddr	5	I	Parameter bus address
pwdat	8/16/32	I	Parameter bus write data
prdat	8/16/32	O	Parameter bus read data

Interface Descriptions

Video Input/Output

The 2D Scaler IP core uses a simple handshake to pass pixel data into the core. The core asserts its ready output when it is ready to receive data. When the driving module has data to give the core, it drives the core's *dvalid_in* port to a '1' synchronously with the rising edge of the *clk* signal, providing the input pixel data on port din. The

frmsync_in input should be driven to a '1' during the clock cycle when the first pixel of the first row in the incoming video frame is active.

Correspondingly, *dvalid_out* is active when valid output pixel data is available on *dout*, and *frmsync_out* marks the first pixel, first row of the output video frame.

When the input signal *dout_enable* is asserted, the core will output video pixels. When *dout_enable* is de-asserted, the core stops generating output pixels after some pipeline delay. The maximum pipeline delay is not greater than 26 clock cycles.

Parameter Register Read/Write Interface

The 2D Scaler IP core uses a simple register read/write interface to update the input and output frame size control registers for dynamic scaling. The parameter bus interface can be configured to run on a separate clock. It operates at the input pixel clock by default.

When *pwrite* is high, *pwdat* and *paddr* should contain valid data. The contents of all parameter registers will be transferred to the core's internal storage when UPDATE is asserted. If a parameter has not been written to before the assertion of UPDATE, its old value will be transferred into the internal storage. The *prdat* contains the register read data corresponding to the address value on the *paddr* in the previous clock cycle.

When parameter bus data width equals 32, *paddr*[1:0] should be fixed to 0. When parameter bus data width equals to 16, *paddr*[0] should be fixed to 0.

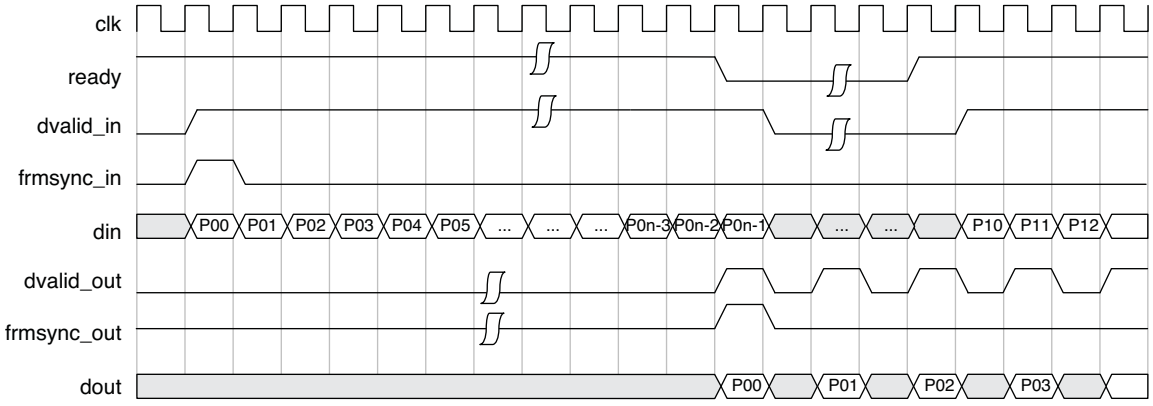
The parameter bus data width is determined by the system CPU data width.

Control Signals and Timing

Line and Frame Flushing

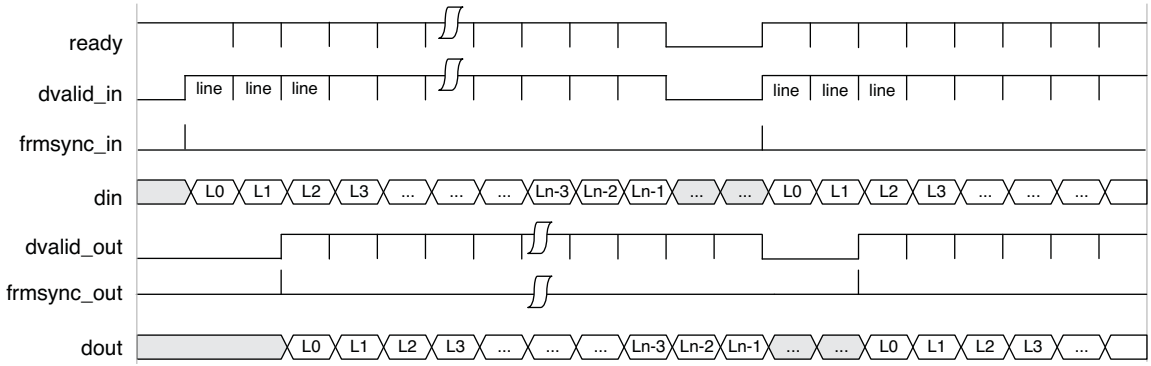
The 2D Scaler IP core can accept pixels line by line and output the pixels line by line. There are several clock cycles required at the end of every line for the core to flush the current output line. Figure 2-4 shows the line flushing cycles. The core cannot accept new input pixels during this time and will pull its *ready* output signal low.

Figure 2-4. Timing Diagram for Line Flushing Cycles



Similarly, there is a frame flushing period at the end of each input frame for the core to complete the current output frame. The number of clock cycles required equals ((number of vertical filter taps) / 2) * (input frame width + line flushing cycle). Figure 2-5 shows the frame flushing cycle.

Figure 2-5. Timing Diagram of Frame Flushing Cycles



Video Input/Output Timing

The 2D Scaler IP core supports single color, YCbCr 4:2:2, YCbCr 4:4:4 and RGB video formats.

For YCbCr 4:4:4 or RGB video formats, the three planes are interleaved for serial scaling and combined on the *din* and *dout* ports for parallel scaling. Figures 2-6 and 2-7 show the timing of RGB serial scaling and parallel scaling.

Figure 2-6. RGB Serial Scaling

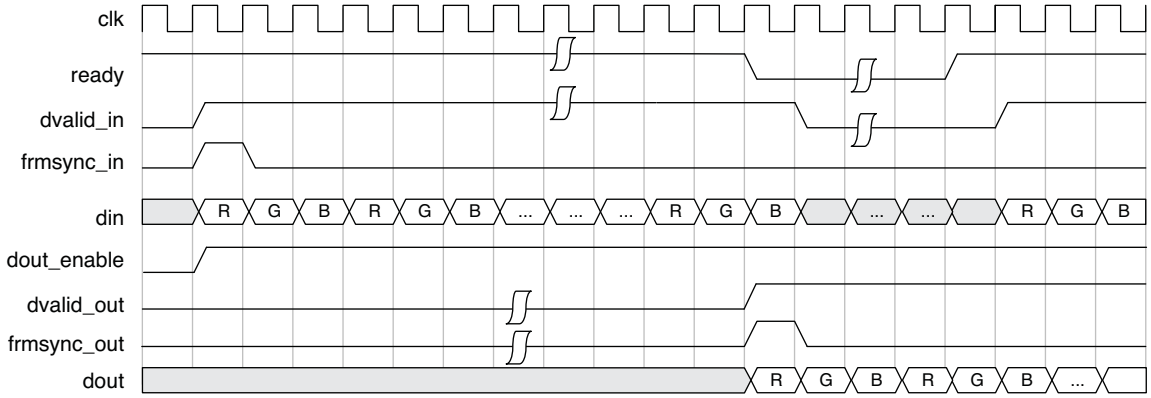
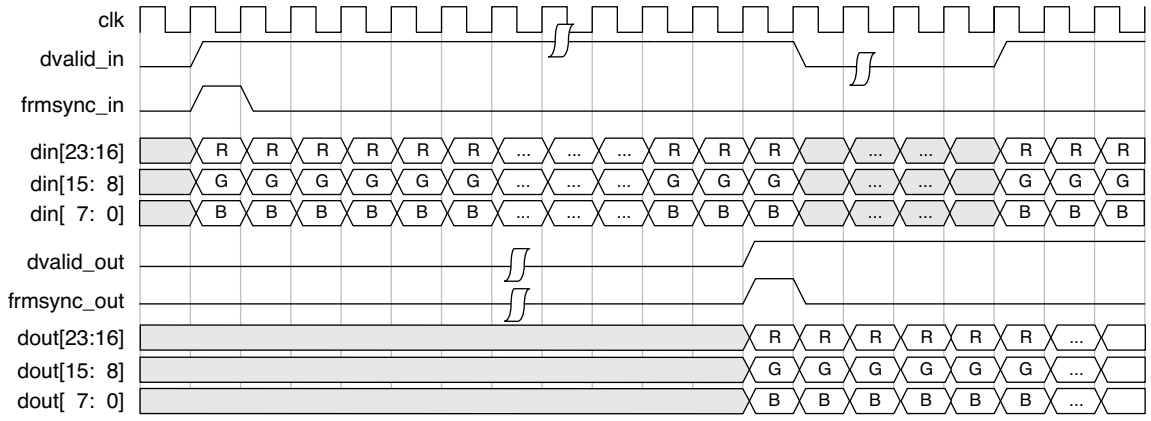


Figure 2-7. RGB Parallel Scaling (8-bit pixel)



For YCbCr 4:2:2 video serial scaling, the input and output sequence should be Cb, Y, Cr, Y, …. For parallel scaling, the Y plane occupies the upper bits of the *din* and *dout* ports, and the Cb and Cr planes occupy the lower bits. Cb and Cr planes are interleaved in the lower half, and Cb comes before Cr. Figures 2-8 and 2-9 show the timing of YCbCr 4:2:2 serial scaling and parallel scaling.

Figure 2-8. YCbCr 4:2:2 Serial Scaling

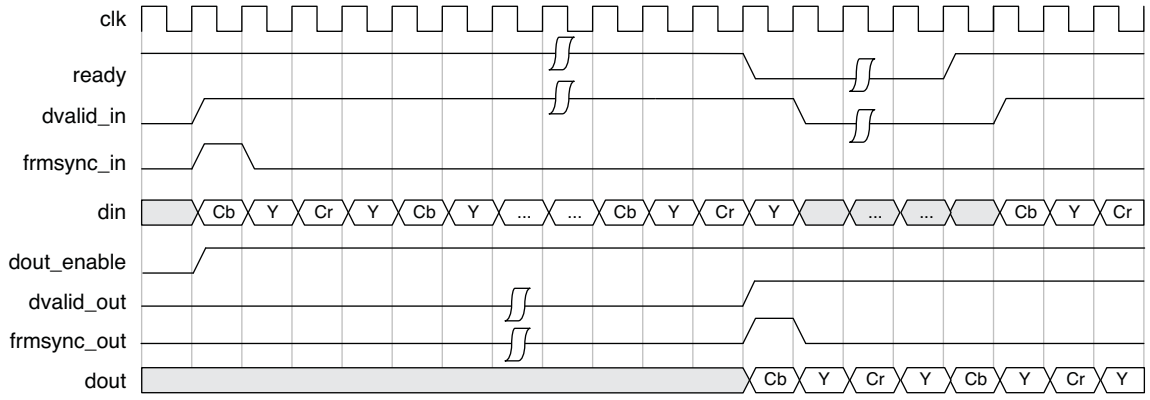


Figure 2-9. YCbCr 4:2:2 Parallel Scaling (8-bit pixel)

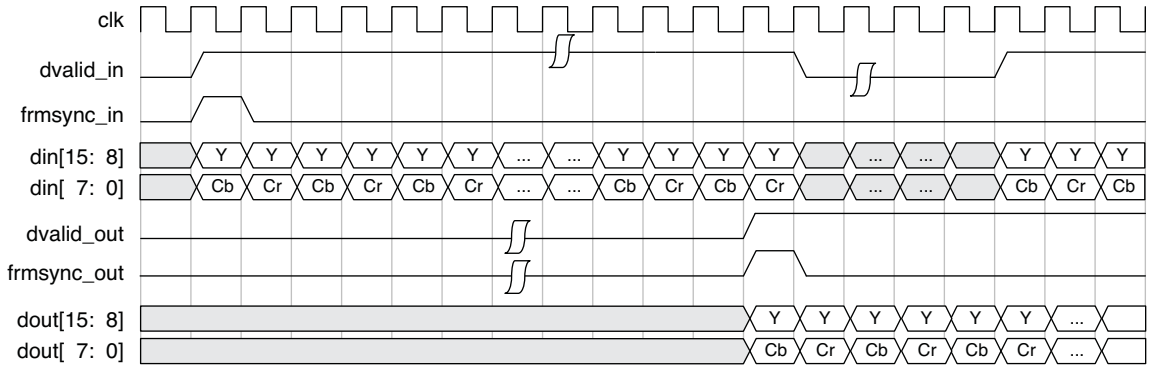
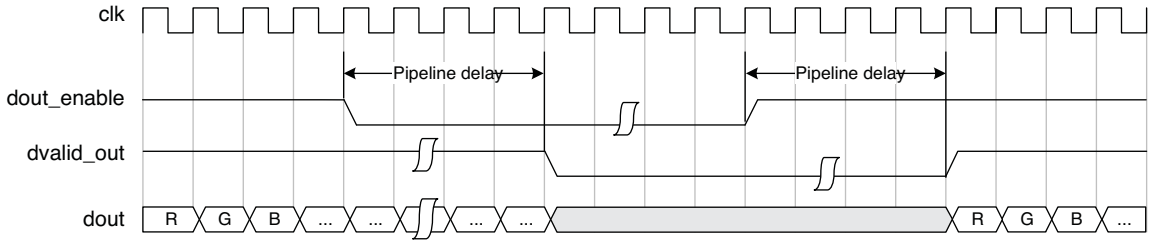


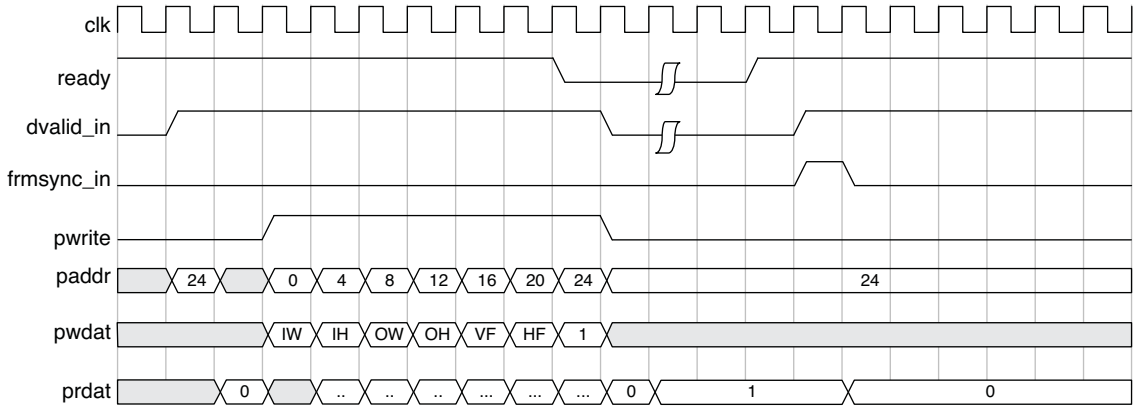
Figure 2-10 shows the dout_enable control timing. When the dout_enable is de-asserted, the core will stop outputting data after the pipeline delay. Similarly when the dout_enable is asserted, the core will begin outputting data after a certain pipeline delay. The asserting and de-asserting of dout_enable can be used to generate horizontal blank and vertical blank depending on the output video format.

Figure 2-10. dout_enable Control Timing



Dynamic Parameter Updating

Figure 2-11. Timing Diagram for Dynamic Parameter Updating (parameter bus width = 32)



In Figure 2-11, IW is the input frame width, and IH is input frame height. OW and OH are the output frame width and height, respectively. VF is the vertical scaling factor, and HF the horizontal scaling factor. The parameter registers are writable only when UPDATE register is 0. When the UPDATE register bit is set to 1, the driving block should start the parameter updating at the next active *frmsync_in* and reset the UPDATE register to 0.

Parameter Settings

This section describes how to generate the 2D Scaler IP core using the IPexpress™ tool. Refer to "IP Core Generation" on page 18 for a description of how to generate the IP.

The 2D Scaler configuration GUI is accessed via the IPexpress tool and provides an interface for setting the desired parameters and invoking the IP core generator. Since the values of some parameters affect the size of the resultant core, the maximum value for these parameters may be limited by the size of the target device. Table provides a list of the user-configurable parameters for the 2D Scaler IP core.

Table 3-1. 2D Scaler IP Core Parameters

Parameter	Range/Options	Default
Video Frame In and Out		
Video format	Single color, YCbCr4:2:2, YcbCr4:4:4 or RGB	YCbCr4:2:2
(Max.) Input frame width	32-4096	720
(Max.) Input frame height	32-4096	576
(Max.) Output frame width	32-4096	1280
(Max.) Output frame height	32-4096	720
Parallel processing	Yes, No	Yes
Dynamic parameter updating	Yes or No	No
Filter Specification		
Kernel	Nearest, Bilinear, Bicibuc, Mitchell, Lanczos	Lanczos
Number of vertical filter taps	4-12 (for Lanczos only)	4
Number of horizontal filter taps	4-12 (for Lanczos only)	4
Number of vertical filter phases	16, 32, 64, 128, 256, 512 (for Mitchell, Bicubic and Lanczos)	64
Number of horizontal filter phases	16, 32, 64, 128, 256, 512 (for Mitchell, Bicubic and Lanczos)	64
I/O Specifications		
Input pixel bit width	8-16	8
Coefficient bit width	6-16	9
Output pixel bit width	8-16	8
Parameter bus bit width	8, 16, 32	32
Separate parameter bus clock	Yes, No	No
Optional Ports		
Synchronous reset	Yes or No	No
Clock enable	Yes or No	No
Output frame size ports	Yes, No	No
Precision Control		
Rounding mode	Truncation, Normal, Convergent	Normal
Memory Type		
Line buffer type	EBR, Distributed	EBR
Vertical coefficient memory type	EBR, Distributed	Distributed
Horizontal coefficient memory type	EBR, Distributed	Distributed
Share vertical and horizontal coefficient memories	Yes or No	No
Synthesis Options		
Frequency constraint (MHz)	1-400	250

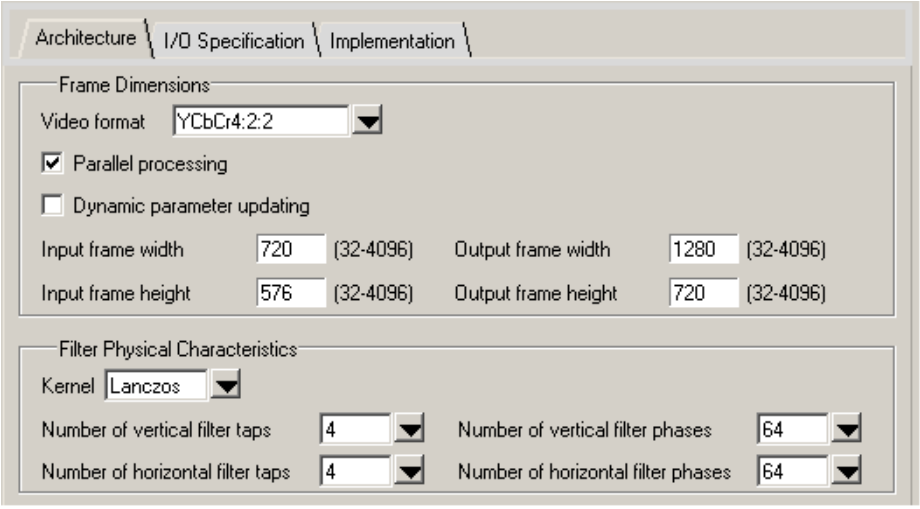
Table 3-1. 2D Scaler IP Core Parameters (Continued)

Parameter	Range/Options	Default
Fanout limit	1-200	100
Resource sharing	Yes, No	Yes
Pipelining and retiming	Yes, No	No

Architecture

The Architecture tab provides settings for video frames and algorithm options.

Figure 3-1. Architecture Tab



Frame Dimensions

This section provides settings that define the number of color planes of the video stream, input/output frame dimensions and in-system (dynamic) parameter updating.

Video format defines the format of a video stream. It can be single-color, YCbCr4:2:2, YCbCr4:4:4 or RGB.

The **Parallel processing checkbox** determines whether the core processes video streams in parallel.

The **Dynamic parameter updating checkbox** determines whether the core supports dynamic scaling. Refer to the "Dynamic Parameter Updating" on page 7 for more information.

Input frame width and **Input frame height** define the input video frame size for fixed scaling.

Output frame width and **Output frame height** define the output video frame size for fixed scaling

When dynamic scaling is enabled, the **Max input frame width**, **Max input frame height**, **Max output frame width** and **Max output frame height** specify the largest input and output frame sizes the core needs to support.

The range for the frame dimensions is 32 to 4096 for both fixed scaling and dynamic scaling.

Filter Physical Characteristics

Kernel selects the scaling algorithm of the core.

Number of vertical filter taps represents the number of multipliers that may be used by the core for the vertical filter; this number varies between 4 and 12 only for the Lanczos filter.

Number of horizontal filter taps represents the number of multipliers that may be used by the core for the horizontal filter; this number varies between 4 and 12 only for the Lanczos filter.

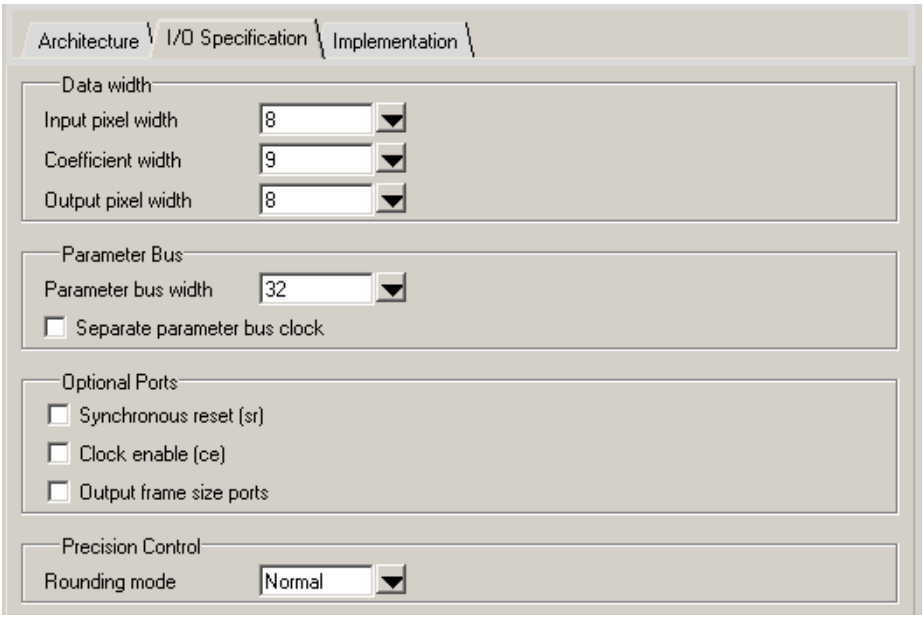
Number of vertical filter phases is a power of 2 number which provides the setting for the number of phases of the vertical filter and may vary between 16 and 512. Setting this number high increases the vertical coefficient storage.

Number of horizontal filter phases is a power of 2 number which provides the setting for the number of phases of the horizontal filter and may vary between 16 and 512. Setting this number high increases the horizontal coefficient storage.

I/O Specification

The I/O Specification tab provides settings for pixel data and coefficient widths, coefficient data type, coefficient binary point, and rounding mode.

Figure 3-2. I/O Specification Tab



Input pixel width sets the bit width of the incoming pixel values and may vary between 8 and 16, inclusive.

Coefficients width sets the bit width of the coefficients and may vary between 6 and 18, inclusive.

Output pixel width sets the output pixel bit width and may vary between 8 and 16, inclusive.

Parameter bus width sets the bus width of the parameter register's read/write interface. This parameter is only available when dynamic parameter updating is selected.

The **Separate parameter bus clock checkbox** determines whether the core uses a separate clock to run the parameter bus interface. This parameter is only available when dynamic parameter updating is selected.

The **Synchronous reset checkbox** determines whether the core has a synchronous reset port.

The **Clock enable checkbox** determines whether the core has a clock enable port.

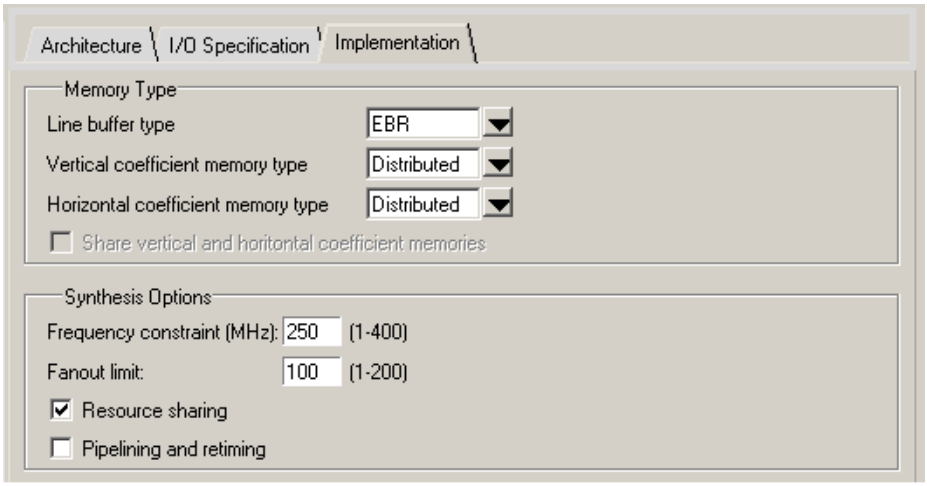
The **Output frame size ports checkbox** determines whether the core provides output frame size ports. This parameter is only available when dynamic parameter updating is selected.

Rounding mode selects between Truncation, Normal (away from zero), and Convergent rounding.

Implementation

The Implementation tab provides settings for the optional ports and synthesis constraints.

Figure 3-3. Implementation Tab



The **Line buffer type** parameter selects EBR or Distributed RAM for the line buffer implementation.

Vertical coefficient memory type selects EBR or Distributed RAM for the vertical coefficient memory.

Horizontal coefficient memory type selects EBR or Distributed RAM for the horizontal coefficient memory.

The **Share vertical and horizontal coefficient memories checkbox** determines whether the core uses one memory for both the vertical and horizontal coefficients. This option is only available when both the vertical and horizontal coefficient memories are set as the EBR type.

Frequency constraint sets the required clock frequency in MHz. This option is active for all the clock domains in the core. **Fanout limit** sets the fanout limit value for the synthesis tool. Resource sharing and pipelining and retiming, if enabled, are synthesis directives that are used in the core generation. Users can adjust these options to get a better timing result.

IP Core Generation

This section provides information on how to generate the 2D Scaler IP core using the Diamond or ispLEVER IPexpress tool, and how to include the core in a top-level design.

Licensing the IP Core

An IP core- and device-specific license is required to enable full, unrestricted use of the 2D Scaler IP core in a complete, top-level design. Instructions on how to obtain licenses for Lattice IP cores are given at www.latticesemi.com/products/intellectualproperty/aboutip/isplevercoreonlinepurchas.cfm.

Users may download and generate the 2D Scaler IP core and fully evaluate the core through functional simulation and implementation (synthesis, map, place and route) without an IP license. The 2D Scaler IP core supports the Lattice IP hardware evaluation capability, which makes it possible to create versions of the IP core which operate in hardware for a limited time (approximately four hours) without requiring an IP license. See "Hardware Evaluation" on page 22 for further details. However, a license is required to enable timing simulation, to open the design in the Diamond or ispLEVER EPIC tool, and to generate bitstreams that do not include the hardware evaluation timeout limitation.

Getting Started

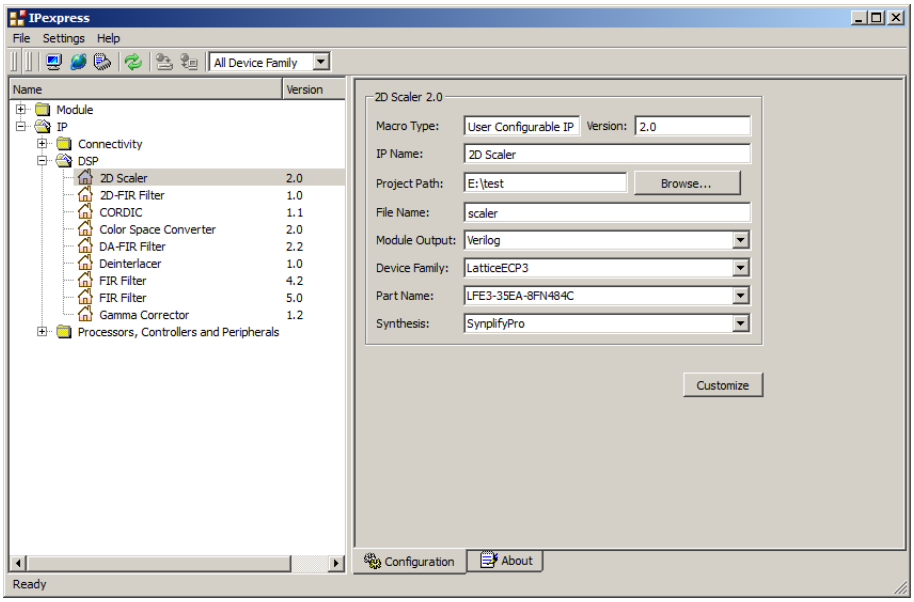
The 2D Scaler IP core is available for download from the Lattice IP Server using the IPexpress tool. The IP files are automatically installed using ispUPDATE technology in any user-specified directory. After the IP core has been installed, the IP core will be available in the IPexpress GUI dialog box shown in Figure 4-1. To generate a specific IP core configuration, the user specifies:

- **Project Path** – Path to the directory where the generated IP files will be located.
- **File Name** – "username" designation given to the generated IP core and corresponding folders and files.
- **(Diamond) Module Output** – Verilog or VHDL.
- **(ispLEVER) Design Entry Type** – Verilog HDL or VHDL.
- **Device Family** – Device family to which IP is to be targeted (e.g. LatticeECP2M, LatticeECP3, etc.). Only families that support the particular IP core are listed.
- **Part Name** – Specific targeted part within the selected device family.

Configuring the 2D Scaler IP Core in IPexpress

The 2D Scaler configuration GUI is accessed via the IPexpress tool, and provides an interface for setting the desired parameters and invoking the IP core generator. The start-up IPexpress page allows the user to select the IP to be generated, project directory, user-designated module name, design entry type, and target device. The "File Name" will be used as the username in the core generation. The 2D Scaler IP core is found under **IP >DSP**, as shown below.

Figure 4-1. IPexpress Dialog Box (Diamond Version)

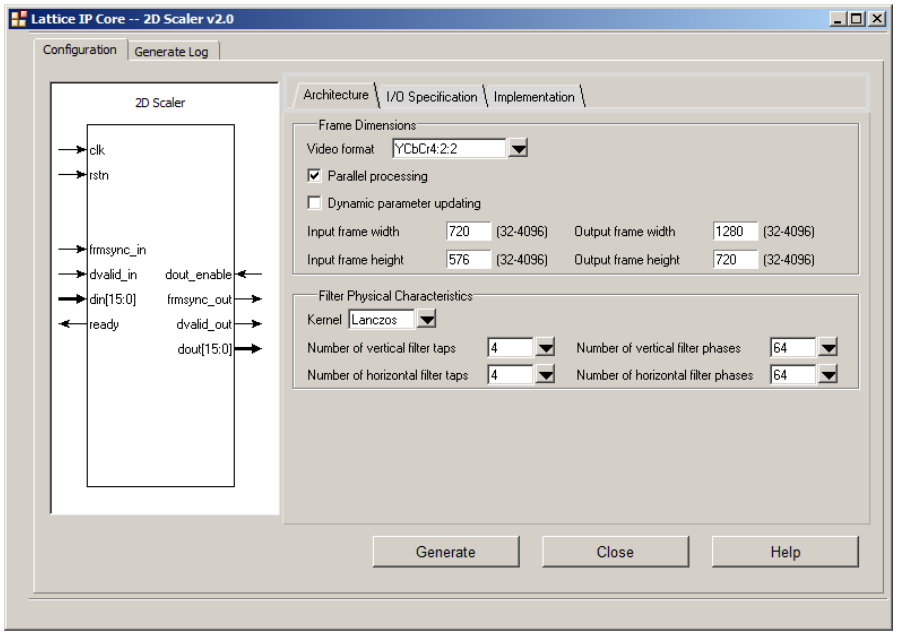


Important Note: File Name cannot be "scaler_core," as this name has been used in the internal design of the core.

Note that if the IPexpress tool is called from within an existing project, Project Path, Module Output (Design Entry in ispLEVER), Device Family and Part Name default to the specified project parameters. Refer to the IPexpress tool online help for further information.

To create a custom configuration, the user clicks the **Customize** button in the IPexpress tool dialog box to display the 2D Scaler IP core configuration GUI, as shown in Figure 4-2. From this dialog box, the user can select the IP parameter options specific to their application. Refer to "Parameter Settings" on page 14 for more information on the 2D Scaler IP core parameter settings.

Figure 4-2. Configuration GUI (Diamond Version)



IPexpress-Created Files and Top-Level Directory Structure

When the user clicks the **Generate** button in the IP Configuration dialog box, the IP core and supporting files are generated in the specified "Project Path" directory. The directory structure of the generated files is shown in Figure 4-3. This example shows the directory structure generated with the 2D Scaler for LatticeECP3 device.

Figure 4-3. 2D Scaler IP Core Directory Structure

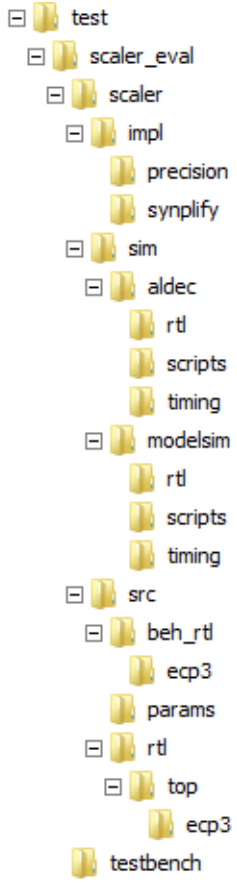


Table 4-1 provides a list of key files and directories created by the IPexpress tool and how they are used. The IPexpress tool creates several files that are used throughout the design cycle. The names of most of the created files are customized to the user's module name specified in the IPexpress tool.

Table 4-1. File List

File	Description
<username>.lpc	This file contains the IPexpress tool options used to recreate or modify the core in the IPexpress tool.
<username>.ipx	The IPX file holds references to all of the elements of an IP or module after it is generated from the IPexpress tool (Diamond version only). The file is used to bring in the appropriate files during the design implementation and analysis. It is also used to re-load parameter settings into the IP/module generation GUI when an IP/module is being re-generated.
<username>.ngo	This file provides the synthesized IP core.
<username>_bb.v	This file provides the synthesis black box for the user's synthesis.
<username>_inst.v	This file provides an instance template for the 2D Scaler IP core.
<username>_beh.v	This file provides the front-end simulation library for the 2D Scaler IP core.

Table 4-2 provides a list of additional key files that provide IP core generation status information and command line generation capability that are generated in the user's project directory.

Table 4-2. Additional Files

File	Description
<username>_generate.tcl	This file is created when the GUI **Generate** button is pushed. This file may be run from the command line.
<username>_generate.log	This is the synthesis and map log file.
<username>_gen.log	This is the IPexpress IP generation log file.

Instantiating the Core

The generated 2D Scaler IP core package includes black-box (<username>_bb.v) and instance (<username>_inst.v) templates that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file that can be used as an instantiation template for the IP core is provided in \<project_dir>\scaler_eval\<username>\src\rtl\top. Users may also use this top-level reference as the starting template for the top level of their complete design.

Running Functional Simulation

Simulation support for the 2D Scaler IP core is provided for the Aldec Active-HDL (Verilog and VHDL) simulator and the Mentor Graphics ModelSim simulator. The functional simulation includes a configuration-specific behavioral model of the 2D Scaler IP core. The test bench sources stimulus to the core, and monitors output from the core. The generated IP core package includes the configuration-specific behavior model (<username>_beh.v) for functional simulation in the Project Path root directory. The simulation scripts supporting ModelSim evaluation simulation is provided in \<project_dir>\scaler_eval\<username>\sim\modelsim\scripts. The simulation script supporting Aldec evaluation simulation is provided in \<project_dir>\scaler_eval\<username>\sim\aldec\scripts. Both ModelSim and Active-HDL simulation is supported via test bench files provided in \<project_dir>\scaler_eval\testbench. Models required for simulation are provided in the corresponding \models folder.

Users may run the Active-HDL evaluation simulation by doing the following:

1. Open Active-HDL.
2. Under the **Tools** tab, select **Execute Macro**.
3. Browse to folder \<project_dir>\scaler_eval\<username>\sim\aldec\scripts and execute one of the “do” scripts shown.

Users may run the ModelSim evaluation simulation by doing the following:

1. Open ModelSim.
2. Under the **File** tab, select **Change Directory** and choose the folder <project_dir>\scaler_eval\<username>\sim\modelsim\scripts.
3. Under the **Tools** tab, select **Execute Macro** and execute the ModelSim “do” script shown.

   Note: When the simulation is complete, a pop-up window will appear asking “Are you sure you want to finish?” Choose **No** to analyze the results. Choosing **Yes** closes ModelSim.

Synthesizing and Implementing the Core in a Top-Level Design

Synthesis support for the 2D Scaler IP core is provided for Mentor Graphics Precision RTL or Synopsys Synplify. The 2D Scaler IP core itself is synthesized and is provided in NGO format when the core is generated in IPexpress. Users may synthesize the core in their own top-level design by instantiating the core in their top level as described previously and then synthesizing the entire design with either Synplify or Precision RTL synthesis.

The top-level file <username>_eval_top.v provided in \<project_dir>\scaler_eval\<username>\src\top supports the ability to implement the 2D Scaler IP core in isolation. Push-button implementation of this top-level design with

either Synplify or Precision RTL synthesis is supported via the project files <username>_eval.ldf (Diamond) or .syn (ispLEVER) located in the \<project_dir>\scaler_eval\<username>\impl\synplify and the \<project_dir>\scaler_eval\<username>\impl\precision directories, respectively.

To use this project file in Diamond:

1. Choose **File > Open Project**.
2. Browse to \<project_dir>\scaler_eval\<username>\impl\(synplify or precision) in the Open Project dialog box.
3. Select and open **<username>.ldf**. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.
4. Select the **Process** tab in the left-hand GUI window.
5. Implement the complete design via the standard Diamond GUI flow.

To use this project file in ispLEVER:

1. Choose **File > Open Project**.
2. Browse to \<project_dir>\scaler_eval\<username>\impl\(synplify or precision) in the Open Project dialog box.
3. Select and open <username>.syn. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.
4. Select the device top-level entry in the left-hand GUI window.
5. Implement the complete design via the standard ispLEVER GUI flow.

Hardware Evaluation

The 2D Scaler IP core supports the Lattice IP hardware evaluation capability, which makes it possible to create versions of the IP core that operate in hardware for a limited period of time (approximately four hours) without requiring the purchase of an IP license. It may also be used to evaluate the core in hardware in user-defined designs.

Enabling Hardware Evaluation in Diamond

Choose **Project > Active Strategy > Translate Design Settings**. The hardware evaluation capability may be enabled/disabled in the Strategy dialog box. It is enabled by default.

Enabling Hardware Evaluation in ispLEVER

In the **Processes for Current Source** pane, right-click the **Build Database** process and choose **Properties** from the drop-down menu. The hardware evaluation capability may be enabled/disabled in the Properties dialog box. It is enabled by default.

Updating/Regenerating the IP Core

By regenerating an IP core with IPexpress, you can modify any of its settings including device type, design entry method, and any of the options specific to the IP core. Regenerating can be done to modify an existing IP core or to create a new but similar one.

Regenerating an IP Core in Diamond

To regenerate an IP core in Diamond:

1. In IPexpress, click the **Regenerate** button.
2. In the Regenerate view of IPexpress, choose the IPX source file of the module or IP you wish to regenerate.

3. IPexpress shows the current settings for the module or IP in the Source box. Make your new settings in the **Target** box.
4. If you want to generate a new set of files in a new location, set the new location in the **IPX Target File** box. The base of the file name will be the base of all the new file names. The IPX Target File must end with an .ipx extension.
5. Click **Regenerate**. The module's dialog box opens showing the current option settings.
6. In the dialog box, choose the desired options. To get information about the options, click **Help**. Also, check the **About** tab in IPexpress for links to technical notes and user's guides. IP may come with additional information. As the options change, the schematic diagram of the module changes to show the I/O and the device resources the module will need.
7. To import the module into your project if it is not already there, select **Import IPX to Diamond Project** (not available in stand-alone mode).
8. Click **Generate**.
9. Check the **Generate Log** tab to check for warnings and error messages.
10. Click **Close**.

The IPexpress package file (.ipx) supported by Diamond holds references to all of the elements of the generated IP core required to support simulation, synthesis and implementation. The IP core may be included in a user's design by importing the .ipx file to the associated Diamond project. To change the option settings of a module or IP that is already in a design project, double-click the module's .ipx file in the File List view. This opens IPexpress and the module's dialog box showing the current option settings. Then go to step 6 above.

Regenerating an IP Core in ispLEVER

To regenerate an IP core in ispLEVER:

1. In the IPexpress tool, choose **Tools > Regenerate IP/Module**.
2. In the **Select a Parameter File** dialog box, choose the Lattice Parameter Configuration (.lpc) file of the IP core you wish to regenerate, and click **Open**.
3. The **Select Target Core Version, Design Entry, and Device** dialog box shows the current settings for the IP core in the Source Value box. Make your new settings in the **Target Value** box.
4. If you want to generate a new set of files in a new location, set the location in the **LPC Target File** box. The base of the .lpc file name will be the base of all the new file names. The LPC Target File must end with an .lpc extension.
5. Click **Next**. The IP core's dialog box opens showing the current option settings.
6. In the dialog box, choose the desired options. To get information about the options, click **Help**. Also, check the **About** tab in the IPexpress tool for links to technical notes and user's guides. The IP core may come with additional information. As the options change, the schematic diagram of the IP core changes to show the I/O and the device resources the IP core will need.
7. Click **Generate**.
8. Click the **Generate Log** tab to check for warnings and error messages.

Support Resources

Lattice Technical Support

There are a number of ways to receive technical support as listed below.

E-mail Support

techsupport@latticesemi.com

Local Support

Contact your nearest Lattice sales office.

Internet

www.latticesemi.com

References

- HB1009, LatticeECP3 Family Handbook
- HB1003, LatticeECP2/M Family Handbook
- HB1002, LatticeXP2 Family Handbook

Revision History

Date	Document Version	IP Core Version	Change Summary
February 2011	01.0	1.0	Initial release.
October 2011	01.1	2.0	Updated to support 2D Scaler IP version 2.0.
August 2013	01.2	02.1	Modified the title for the Resource Utilization tables
			Updated corporate logo.
			Updated Lattice Technical Support information.
			Updated References format.

Appendix A:

Resource Utilization

This appendix gives resource utilization information for Lattice FPGAs using the 2D Scaler IP core. IPexpress is the Lattice IP configuration utility, and is included as a standard feature of the Diamond and ispLEVER design tools. Details regarding the use of IPexpress can be found in the IPexpress, Diamond and ispLEVER online help systems. For more information on the Diamond or ispLEVER design tools, visit the Lattice web site at: www.latticesemi.com/software.

LatticeECP3 Devices

Table A-1. Performance and Resource Utilization Examples[1]

Video Format	Max. Input Frame Size	Max. Output Frame Size	Parallel Processing	Dynamic	Taps	Pixels Width	Coefficient Width	Registers	LUT4s	Slices	EBRs	MULT9X9s	f_{MAX}
YCbCr422	720x480	1280x720	No	No	4x4	8	9	955	1343	948	4	8	271
YCbCr422	1280x720	720x480	Yes	No	4x4	8	9	1113	1338	996	7	16	246
RGB	1280x720	1920x1080	Yes	Yes	4x4	8	9	1522	1829	1356	9	24	275

1. Performance and utilization data are generated targeting a LFE3-35EA-8FN484C device using Lattice Diamond 1.3 and Synplify Pro E-2011.03L software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP3 family.

Ordering Part Number

The Ordering Part Number (OPN) for the 2D Scaler IP core targeting LatticeECP3 devices is SCALER-E3-U1.

LatticeECP2M and LatticeECP2MS Devices

Table A-2. Performance and Resource Utilization Examples[1]

Video Format	Max. Input Frame Size	Max. Output Frame Size	Parallel Processing	Dynamic	Taps	Pixels Width	Coefficient Width	Registers	LUT4s	Slices	EBRs	MULT9X9s	f_{MAX}
YCbCr422	720x480	1280x720	No	No	4x4	8	9	953	1278	936	4	8	249
YCbCr422	1280x720	720x480	Yes	No	4x4	8	9	1116	1267	991	7	16	262
RGB	1280x720	1920x1080	Yes	Yes	4x4	8	9	1515	1715	1362	9	24	268

1. Performance and utilization data are generated targeting a LFE2M20E-7F484C device using Lattice Diamond 1.3 and Synplify Pro E-2011.03L software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP2M and LatticeECP2MS family.

Ordering Part Number

The Ordering Part Number (OPN) for the 2D Scaler IP core targeting LatticeECP2M/S devices is SCALER-PM-U1.

LatticeECP2 and LatticeECP2S Devices

Table A-3. PPerformance and Resource Utilization Examples[1]

Video Format	Max. Input Frame Size	Max. Output Frame Size	Parallel Processing	Dynamic	Taps	Pixels Width	Coefficient Width	Registers	LUT4s	Slices	EBRs	MULT9X9s	f_{MAX}
YCbCr422	720x480	1280x720	No	No	4x4	8	9	955	1278	936	4	8	248
YCbCr422	1280x720	720x480	Yes	No	4x4	8	9	1116	1267	991	7	16	263
RGB	1280x720	1920x1080	Yes	Yes	4x4	8	9	1515	1715	1362	9	24	269

1. Performance and utilization data are generated targeting a LFE2-20E-7F484C device using Lattice Diamond 1.3 and Synplify Pro E-2011.03L software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP2 and LatticeECP2S family.

Ordering Part Number

The Ordering Part Number (OPN) for the 2D Scaler IP core targeting LatticeECP2 and LatticeECP2S devices is SCALER-P2-U1.

LatticeXP2 Devices

Table A-4. Performance and Resource Utilization Examples[1]

Video Format	Max. Input Frame Size	Max. Output Frame Size	Parallel Processing	Dynamic	Taps	Pixels Width	Coefficient Width	Registers	LUT4s	Slices	EBRs	MULT9X9s	f_{MAX}
YCbCr422	720x480	1280x720	No	No	4x4	8	9	953	1278	936	4	8	220
YCbCr422	1280x720	720x480	Yes	No	4x4	8	9	1116	1267	991	7	16	233
RGB	1280x720	1920x1080	Yes	Yes	4x4	8	9	1515	1715	1362	9	24	238

1. Performance and utilization data are generated targeting a LFXP2-30E-7F484C device using Lattice Diamond 1.3 and Synplify Pro E-2011.03L software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeXP2 family.

Ordering Part Number

The Ordering Part Number (OPN) for the 2D Scaler IP core targeting LatticeXP2 devices is SCALER-X2-U1.