LogiCORE IP Video Scaler v3.0

DS724 September 21, 2010 **Product Specification**

Introduction

The Xilinx Video Scaler LogiCORE™ IP is an optimized hardware block that converts an input color image of one size to an output image of a different size. This highly configurable core supports in-system programmability on a frame basis. System design is made easier through support of both streaming-video and frame buffer-based interfaces.

The Video Scaler core allows the filter coefficients to be updated dynamically. It supports RGB/4:4:4, YUV4:2:2, and YUV4:2:0 color formats for 8, 10, or 12-bit video. The architecture takes advantage of the high-performance XtremeDSP™ slices.

The Video Scaler core may be fed with live video but also supports the option of a memory interface.

CORE Generator™ technology generates the core as either an EDK pCore, a standalone netlist for a General Purpose Processor (GPP) or as a Fixed Mode netlist.

LogiCORE IP Facts Table					
Supported Device Family(1)	Spartan®-3A DSP, Spartan-6, Virtex®-5, Virtex-6				
Supported User Interfaces	General Purpose Processor (GPP), EDK pCore PLB, Constant				
Supported Operating Systems	Windows XP Professional 32-Bit/64-bit, Windows Vista Business 32-Bit/64-bit, Red Hat Enterprise Linux WS v4.0 32-bit/64-bit, Red Hat Enterprise Desktop v5.0 32-bit/64-bit (with Workstation Option), SUSE Linux Enterprise (SLE) desktop and server v10.1 32-bit/64-bit				
	Resources(2)(3)				**Frequency**
Configuration	**LUTs**	**FFs**	**DSP48s**	**Block RAMs**	**Max. Freq.(4)**
Spartan6 (-2) YC4:2:2 Single Engine 8-bit video 4Hx4VTaps,e.g.,640 x480 to 800x600 Constant Mode	1158	700	12	10	150 MHz
Virtex5 (-1) YC4:2:2 Dual Engine 10-bit video 8Hx8VTaps,e.g,720P /60 to 1080P/60 for broadcast	1668	3149	40	23	225 MHz
Virtex6 (-1) RGB/4:4:4 Triple Engine 12-bit video 9Hx7VTaps,e.g.,High -end specialist application	1605	4446	52	31	250 MHz
Provided with Core					
Documentation	Product Specification				
Design Files, Example Design, Test Bench, Constraints File, Simulation Model	Not Provided				
Tested Design Tools					
Design Entry Tools	CORE Generator™, Platform Studio (XPS)				
Simulation	ModelSim v6.5c, Xilinx ISIM 12.3				
Synthesis Tools	ISE® 12.3				
Support					
Provided by Xilinx, Inc.					

1. For a complete listing of supported devices, see the release notes for this core.
2. For more complete resource utilization numbers, see Table 5, Table 6, Table 7, and Table 8.
3. All figures are extracted from XST synthesis reports.
4. For more complete performance data, see "Performance," page 24.

Features

- Support for all speed grades of Virtex®-6, Virtex-5, Spartan®-6, and Spartan-3A DSP devices
- Independently programmable scaling factor for horizontal and vertical size
- Support for 2-12 taps in both horizontal and vertical directions
- Supports 8, 10, and 12-bit video data for RGB/4:4:4, YC4:2:2 or YC4:2:0 chroma formats
- Configurable number of phases for each coefficient set (2-16 inclusive, or 32 or 64)
- Internal storage for up to 16 coefficient sets
- Negative start phase support
- Programmable subject and target area size
- User-loadable 16-bit coefficients
- Highly scalable spatial resolution support from 176x144 to 4096x4096
  - Supported standard SD formats are 480i, 480P, 576i, 576P
  - Supported standard HD formats are 720P, 1080i, 1080P
  - Digital Cinema at 4Kx2K
  - Supports all PC resolutions
- Maximum achievable frame-rate is dependent upon spatial resolution and clock-frequency used:
  - 1080P @ 60Hz may be achieved in all device families
  - 4kx2k @ 24 Hz may be achieved in Virtex-5 and Virtex-6 devices
- Support for streaming video and memory interfaces
- Supports shareable coefficients in the Y/C domains and in the H/V domains
- Supports user-definable preloaded coefficients for all interface modes (pCore, GPP. Constant)
- CORE Generator software support for Generic, Constant and pCore configurations
- MicroBlaze™ processor driver source code provided for the pCore option, including coefficients

Applications

- Broadcast Displays, Cameras, Switchers, Video Servers, etc.
- LED Wall
- Multi-Panel Displays
- Digital Cinema
- 4Kx2K Projectors
- Post-processing block for image scaling
- Medical Endoscope
- Video Surveillance
- Consumer Displays
- Video Conferencing
- Machine Vision

General Introduction

Video scaling is the process of converting an input color image of dimensions X_{in} pixels by Y_{in} lines to an output color image of dimensions X_{out} pixels by Y_{out} lines.

Video scaling is a form of 2D filter operation which can be approximated with the equation shown in Figure 1.

$$Pix_{out}[x,y] = \sum_{HTaps-1}^{i=0} \sum_{VTaps-1}^{j=0} Pix_{in}[x-(HTaps/2)+i, y-(VTaps/2)+j)] \times Coef[i,j]$$

Figure 1: **Generic Image Filtering Equation**

In this equation, x and y are discrete locations on a common sampling grid; Pix_{out} (x, y) is an output pixel that is being generated at location (x, y); Pix_{in} (x, y) is an input pixel being used as part of the input scaler aperture; Coef (i, j) is an array of coefficients that depend upon the user application; and HTaps, VTaps are the number of horizontal and vertical taps in the filter.

The coefficients in this equation represent weightings applied to the set of input samples chosen to contribute to one output pixel, according to the scaling ratio.

The set of coefficients constitute filter banks in a polyphase filter whose frequency response is determined by the amount of scaling applied to the input samples. The phases of the filter represent subfilters for the set of samples in the final scaled result.

The number of coefficients and their values are dependent upon the required low-pass, anti-alias response of the scaling filter; for example, smaller scaling ratios require lower passbands and more coefficients. Filter design programs based on the Lanczos algorithm are suitable for coefficient generation. Moreover, MATLAB® product fdatool/fvtool may be used to provide a wider filter design toolset. More information about coefficients is located in the *Xilinx LogiCORE IP Video Scaler v3.0 User Guide* UG678.

A direct implementation of this equation suggests that a filter with VTaps x HTaps multiply operations per output are required. However, the Xilinx Video Scaler uses a separable filter, which completes an approximation of the 2-D operation using two 1-D stages in sequence – a vertical filter (V-filter) stage and a horizontal filter (H-filter) stage. The summed intermediate result of the first stage is fed sequentially to the second stage.

The vertical filter stage filters only in the vertical domain, for each incrementing horizontal raster scan position x, creating an intermediate result described as Vpix (Figure 2).

$$VPix_{\text{int}}[x,y] = \sum_{VTaps-1}^{j=0} Pix_{in}[x, y-(VTaps/2)+j)] \times Coef[j]$$

Figure 2: **Vertical Portion of Scaling Equation**

The output result of the vertical component of the scaler filter is input into the horizontal filter with the appropriate rounding applied (Figure 3).

$$Pix_{out}[x,y] = \sum_{HTaps-1}^{i=0} VPix_{\text{int}}[x-(HTaps/2)+i, y] \times Coef[i]$$

Figure 3: **Horizontal Portion of Scaling Equation**

The separation means this can be reduced to the shown VTaps and HTaps multiply operations, saving FPGA resources.

Polyphase Concept

For scaling, the input and output sampling grids are assumed to be not common, in contrast to the preceding. To express a discrete output pixel in terms of input pixels, it is necessary to know or estimate the location of the output pixel relative to the closest input pixels when superimposing the output sampling grid upon the input sampling grid for the equivalent 2-D space. With this knowledge, the algorithm approximates the output pixel value by using a filter with coefficients weighted accordingly. Filter taps are consecutive data-points drawn from the input image.

As an example, Figure 4 shows a desired 5x5 output grid ("O") superimposed upon an original 6x6 input grid ("X"), occupying common space. In this case, estimating for output position (x, y) = (1, 1), shows the input and output pixels to be co-located. The user may weight the coefficients to reflect no bias in either direction, and may even select a unity coefficient set. Output location (2, 2) is offset from the input grid in both vertical and horizontal dimensions. Coefficients may be chosen to reflect this, most likely showing some bias towards input pixel (2, 2), etc. Filter characteristics may be built into the filter coefficients by appropriately applying anti-aliasing low-pass filters.

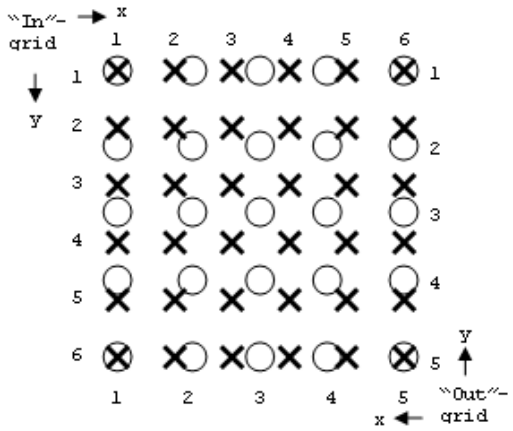

Figure 4: **5x5 Output Grid ("O") Super-imposed over 6x6 Input Grid ("X")**

The space between two consecutive input pixels in each dimension is conceptually partitioned into a number of bins or phases. The location of any arbitrary output pixel will always fall into one of these bins, thus defining the phase of coefficients used. The filter architecture should be able to accept any of the different phases of coefficients, changing phase on a sample-by-sample basis.

A single dimension is shown in Figure 5. As illustrated in this figure, the five output pixels shown from left to right could have the phases 0, 1, 2, 3, 0.

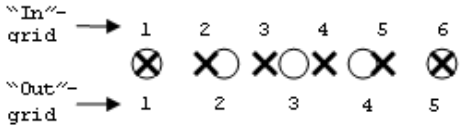

Figure 5: **Super-imposed Grids for 1 Dimension**

The examples in Figure 4 and Figure 5 show a conversion where the ratio $X_{in}/X_{out} = Y_{in}/Y_{out} = 5/4$. This ratio is known as the Scaling Factor, or SF. Knowledge of this factor is required before using the scaler, and it is a direct input to the system. Usually it is defined by the system requirements at a higher level, and it may be different in H and V dimensions. A typical example is drawn from the broadcast industry, where some footage may be shot using 720p (1280x720), but the cable operator needs to deliver it as per the broadcast standard 1080p (1920x1080). The SF becomes 2/3 in both H and V dimensions.

Typically, when $X_{in} > X_{out}$, this conversion is known as horizontal down-scaling (SF > 1). When $X_{in} < X_{out}$, it is known as horizontal up-scaling (SF < 1).

Pin Tables

Table 1 and Table 2 are the pin-tables for Video Interface and Control Interface respectively. There are two possible options for Video Interface and three possible options for Control Interface, selectable in CORE Generator software.

Table 1: **Video Interface Pins**

Signal	Notes	Signal Direction	Width	Description
clk		Input	1	System clock
sclr		Input	1	System sync reset (active high)
video_in_clk		Input	1	Input pixel-rate clock
active_video_in	Live data-source only	Input	1	Write-enable to input data FIFO
video_data_in	Live data-source only	Input	16	YC data input: (7:0): Luma (15:8): Chroma
line_request	Live data-source only	Output	1	Input data FIFO may accept another input line
active_chroma_in	1. Live data-source only 2. 4:2:0 only	Input	1	Chroma input-line validation 4:4:4 and 4:2:2 operation: set to '1' permanently 4:2:0 operation: set to '1' for active chroma lines only
vblank_in	Live data-source only	Input	1	Vertical synchronization pulse – must be high during blanking period
hblank_in	Live data-source only	Input	1	Horizontal synchronization pulse – must be high during blanking period
video_data_out	Live data-source only	Output	16	YC data output: (7:0): Luma (15:8): Chroma
frame_rst	Live data-source only	Output	1	General purpose reset signal asserted for one line period during vertical blanking
rd_data	Memory data-source only	Input	16	YC data input: (7:0): Luma (15:8): Chroma
rd_empty	Memory data-source only	Input	1	Empty flag from memory source – to be asserted when one full line of data is not available
rd_re	Memory data-source only	Output	1	Memory read enable
vsync_in	Memory data-source only	Input	1	Vertical sync signal indicating that the next line at the input to the scaler will be the top line in the input frame
video_out_clk		Input	1	Output pixel-rate clock
video_out_full		Input	1	Full flag input from next block
video_out_we		Output	1	Write-enable to output

Table 2: **Control Interface Pins**

Signal	Notes	Signal Direction	Width	Description
Interrupt Pins *Pins not available in pCore – use PLB interface*				
intr_output_frame_done	GPP and Constant Interfaces	Output	1	Issued once per complete output frame
intr_input_error	GPP and Constant Interfaces	Output	1	Issued if active_video_in is asserted before the scaler is ready to receive a new line
intr_output_error	GPP and Constant Interfaces	Output	1	Issued if frame period completes before full output frame has been delivered
intr_reg_update_done	GPP only	Output	1	Issued during Vertical blanking when the register values have been transferred to the active registers
intr_coef_wr_error	GPP only	Output	1	Issued if coefficient is written into coefficient FIFO when the FIFO is not ready
intr_coef_fifo_rdy	GPP only	Output	1	Issued when the coefficient FIFO is ready to receive a coefficient for the current set; stays low once a full set has been written into FIFO; sent high during Vertical blanking
Coefficient Interface *Pins not available in pCore – use PLB Control Interface* *Pins not available in Constant Interface*				
coef_wr_en	GPP only	Input	1	Write-enable for coefficient – active high
coef_data_in	GPP only	Input	32	Coefficient input bus
coef_set_wr_addr	GPP only	Input	4	Coefficient memory write address
Dynamic Control Register Interface *Pins not available in pCore – use PLB Control Interface* *Pins not available in Constant Interface*				
hsf	GPP only	Input	24	Horizontal **Shrink** Factor Format 24.20 Range 12.0 (0xC00000) to 1/12 (0x015555) Note: Conceptually, this input value is the reciprocal of the horizontal scale factor: hsf > 1.0 for horizontal downscaling cases hsf < 1.0 for horizontal upscaling cases For example, upscaling 640 to 1024 pixels, hsf = 0.625 (0x0A0000)
vsf	GPP only	Input	24	Vertical **Shrink** Factor Format 24.20 Range 12.0 (0xC00000) to 1/12 (0x015555) Note: Conceptually, this input value is the reciprocal of the vertical scale factor: vsf > 1.0 for vertical downscaling cases vsf < 1.0 for vertical upscaling cases For example, downscaling 1080 to 720 lines, vsf = 1.5 (0x180000)

Table 2: **Control Interface Pins *(Cont'd)***

Signal	Notes	Signal Direction	Width	Description
aperture_start_pixel	GPP only	Input	11	Location of first subject pixel in input line, relative to first active pixel in that line *Important Note: Currently, an even number must be specified for this input value.*
aperture_end_pixel	GPP only	Input	11	Location of final subject pixel in input line, relative to first active pixel in that line. *Important Note: Currently, an even number must be specified for this input value.*
aperture_start_line	GPP only	Input	11	Location of first subject line in input image, relative to first active line in that image
aperture_end_line	GPP only	Input	11	Location of final subject line in input image, relative to first active line in that image
output_h_size	GPP only	Input	11	Desired width of output rectangle (pixels) **Important Note: Currently, an even number must be specified for this input value.**
output_v_vize	GPP only	Input	11	Desired height of output image (lines)
num_h_phases	GPP only	Input	7	Number of phases of coefficients in current horizontal filter set
num_v_phases	GPP only	Input	7	Number of phases of coefficients in current vertical filter set
h_coeff_set	GPP only	Input	4	Active coefficient set to use in horizontal filter operation
v_coeff_set	GPP only	Input	4	Active coefficient set to use in vertical filter operation
start_hpa_y	GPP only	Input	21	Fractional value used to initialize horizontal accumulator at rectangle left edge for luma
start_vpa_y	GPP only	Input	21	Fractional value used to initialize vertical accumulator at rectangle top edge for luma
start_hpa_c	GPP only	Input	21	Fractional value used to initialize horizontal accumulator at rectangle left edge for chroma
start_vpa_c	GPP only	Input	21	Fractional value used to initialize vertical accumulator at rectangle top edge for chroma
control	GPP only	Input	32	General control register
PLB Control Interface *pCore only*				
SPLB_Clk	pCore only	Input	1	Slave PLB Clock
SPLB_Rst	pCore only	Input	1	Slave PLB Reset
PLB_ABus	pCore only	Input	C_SPLB_AWIDTH	PLB Address Bus
PLB_PAValid	pCore only	Input	1	PLB Primary Address Valid
PLB_masterID	pCore only	Input	C_SPLB_MID_WIDTH	PLB Current Master identifier
PLB_abort	pCore only	Input	1	PLB Abort Bus Request
PLB_RNW	pCore only	Input	1	PLB Read Not Write

Table 2: **Control Interface Pins *(Cont'd)***

Signal	Notes	Signal Direction	Width	Description
PLB_BE	pCore only	Input	C_SPLB_DWIDTH/8	PLB Byte Enables
PLB_MSize	pCore only	Input	2	PLB Master Data Bus Size
PLB_size	pCore only	Input	4	PLB Transfer Size
PLB_type	pCore only	Input	3	PLB Transfer Type
PLB_wrDBus	pCore only	Input	C_SPLB_DWIDTH	PLB Write Data Bus
PLB_wrBurst	pCore only	Input	1	PLB Burst Write Transfer indicator
PLB_rdBurst	pCore only	Input	1	PLB Burst Read Transfer indicator
PLB_SAValid	pCore only	Input	1	PLB Secondary Address Valid
PLB_UABus	pCore only	Input	32	PLB Upper Address Bus
PLB_BusLock	pCore only	Input	1	PLB Bus Lock
PLB_LockErr	pCore only	Input	1	PLB Lock Error
PLB_TAttribute	pCore only	Input	16	PLB Attribute
PLB_RdPrim	pCore only	Input	1	PLB Read Primary
PLB_WrPrim	pCore only	Input	1	PLB Write Primary
PLB_RDPendPri	pCore only	Input	2	PLB Read Pending on Primary
PLB_WrPendPri	pCore only	Input	2	PLB Write Pending on Primary
PLB_RdPendReq	pCore only	Input	1	PLB Read Pending Request
PLB_WrPendReq	pCore only	Input	1	PLB Write Pending Request
Sl_addAck	pCore only	Output	1	Slave Address Acknowledge
Sl_SSize	pCore only	Output	2	Slave Data Bus Size
Sl_wait	pCore only	Output	1	Slave Wait indicator
Sl_rearbitrate	pCore only	Output	1	Slave Rearbitrate Bus indicator
Sl_wrDAck	pCore only	Output	1	Slave Write Data Acknowledge
Sl_wrComp	pCore only	Output	1	Slave Write Transfer Complete indicator
Sl_wrBTerm	pCore only	Output	1	Slave Terminate Write Burst Transfer
Sl_rdDBus	pCore only	Output	C_SPLB_DWIDTH	Slave Read Data Bus
Sl_rdWdAddr	pCore only	Output	4	Slave Read Word Address
Sl_rdDAck	pCore only	Output	1	Slave Read Data Acknowledge
Sl_rdComp	pCore only	Output	1	Slave Read Transfer Complete indicator
Sl_rdBTerm	pCore only	Output	1	Slave Terminate Read Burst Transfer
Sl_MBusy	pCore only	Output	C_SPLB_NUM_MASTERS	Slave Busy indicator
Sl_MrdErr	pCore only	Output	C_SPLB_NUM_MASTERS	Slave Read Error indicator
Sl_MwrErr	pCore only	Output	C_SPLB_NUM_MASTERS	Slave Write Error indicator
Sl_MIRQ	pCore only	Output	C_SPLB_NUM_MASTERS	Slave Interrupt
IP2INTC_Irpt	pCore only	Output	1	Interrupt Signal

Table 2: **Control Interface Pins *(Cont'd)***

Signal	Notes	Signal Direction	Width	Description
dcm_locked	pCore only	Input	1	Not used; Drive to '1'
LEDsOut	pCore only	Input	8	Debug output: **Bit0:** Input "video_in_clk" heartbeat: 1 Hz flash for 100 MHz video_in_clk **Bit1:** Input "clk" heartbeat: 1 Hz flash for 100 MHz clk **Bit2**: Not used **Bit3:** Not used **Bit4:** Not used **Bit5:** Input VBlank heartbeat: 1 Hz flash for 60Hz vblank_in (Live mode) **Bit6:** video_out_full **Bit7:** rd_empty (Memory mode)

GUI Parameters

Component Name: The component name is used as the base name of output files generated for the module. Names must begin with a letter and must be composed from characters: a to z, 0 to 9 and "_".

Interface Selection: The user may select the interface type between EDK pCore, GPP or Constant interface types.

Num H Taps: This represents the number of multipliers that may be used in the system for the horizontal filter, and may vary between 2 and 12 inclusive. The user should be aware that increasing this number increases XtremeDSP slice usage.

Num V Taps: This represents the number of multipliers that may be used in the system for the vertical filter, and may vary between 2 and 12 inclusive. The user should be aware that increasing this number increases XtremeDSP slice usage.

Input/output rectangle Maximum Frame Dimensions (for pCore and GPP only): These fields represent the maximum anticipated rectangle size on the input and output of the Video Scaler. The rectangle may vary between 32x32 through 4096x4096. These dimensions affect BRAM usage in the input and output line-buffers, and in the Vertical filter line-stores. They also have an effect on the calculation of the maximum frame-rate achievable when using the scaler core.

Max Number of Phases (for pCore and GPP only)**:** This represents the maximum number of phases that the designer intends for a particular system. It may vary between 2 and 16 inclusive, but also may be set to 32 or 64. Setting this value high has two consequences: increased coefficient storage (block RAM), and increased time required to download each coefficient set.

Video Data Bitwidth: 8, 10, or 12 bits. This specifies both the output and input bitwidths.

Max Coef Sets (for pCore and GPP only)**:** This represents the maximum number of sets of coefficients that may be stored internally to the scaler. It may vary between 2 and 16. The coefficient set to be used during the scaling of the current frame is selected using the `h_coeff_set` and `v_coeff_set` controls. Increasing this value simply increases block RAM usage.

Chroma Format: Set this according to the chroma format required, either 4:2:2 (default), 4:2:0, or 4:4:4. Selecting 4:2:0 causes greater block RAM usage to align luma and chroma vertical apertures prior to the filters, and to realign the output lines after the filters.

Data Source Selection: The user may select how he intends to deliver video data to the core – Live or Memory data source.

Frame Reset Line Number (for Live-Video data-source only)**:** The user may set this value to move the position of the `frame_rst` output signal within the vertical blanking. It must be set such that `frame_rst` occurs while `vblank_in` is high. For more information, see the "Video IO Interface and Timing" section in the *Video Scaler v3.0 User Guide* (UG678).

YC Filter Configuration: (YC 4:2:2 only). When running 4:2:2 data, the scaler may be configured to perform Y and C operations in parallel (two engines) or sequentially (one engine). Selecting "Auto" allows the tool to select whether to use single- or dual engines. The "Information" tab indicates the estimated maximum frame-rate achievable given the user's parameter settings. It makes this decision according to the specified desired frame rate. The user may also manually select between the two options. When in 4:4:4/RGB mode, the scaler is implemented with three engines in parallel. In 4:2:0 mode, only one engine is used.

Coefficient File Input: The user may specify a .coe file to preload the coefficient store with coefficients. When using Constant mode, this is a necessary step. The .coe file format is described in more detail in the "Control Interface" section of *Video Scaler v3.0 User Guide* (UG678).

The user may specify whether the same coefficients are used for Y and C filter operations.The user may also specify whether the H and V operations use the same coefficients. This is only an option if the specified number of horizontal taps is equal to the specified number of vertical taps. Specifying the same coefficients in this way may make for a smaller implementation.

General Purpose Processor (GPP) Interface

The General Purpose Processor interface exposes the control, status, address and data registers as ports. See Table 2.

Dynamic Register Interface

The dynamic register inputs used for run-time scaler control are listed Table 2. They may be driven in by the user as desired, but all these values will only become active in the core during `vblank_in`, using an active value capture register.

Coefficient Table

One single size-configurable, block RAM-based, dual port RAM block stores all H and V coefficients combined, and holds different coefficients for luma and chroma as desired. The coefficient memory output width equals `num_taps` x `coef_width` bits. Hence, because the coefficient bit width is 16, all implementations require one block RAM per two taps.

The user may want more than one coefficient set from which to choose. For example, it may be necessary to select different filter responses for different scale factors. This is often true when down-scaling by different factors to eliminate aliasing artifacts.

The number of phases for each set may also vary, dependent upon the nature of the conversion, and how the user has elected to generate and partition the coefficients. The maximum number of phases per set defines the size of the memory required to store them, and this may have an impact on resource usage.

Coefficient Interface

The scaler uses only one set of coefficients per frame period. To change to a different set of stored coefficients for the next frame, the user must use the `h_coeff_set` and `v_coeff_set` dynamic register inputs. Vertical coefficients are stored in the upper half of the coefficient memory.

The user may load new coefficients into a different location in the coefficient store during some frame period before they are required. The user may load one coefficient set per frame period. Subsequently, this coefficient set may be selected for use when using `h_coeff_set` and `v_coeff_set`.

Filter coefficients may be loaded into the scaler using the coefficient memory interface. This comprises:

- **coef_data_in(31:0)**: 32 bit coefficient input bus
- **coef_wr_en**: Coefficient write-enable
- **coef_set_wr_addr(3:0)**: Common write address bus for coefficient set
- **intr_coef_fifo_rdy**: Output flag indicating the readiness of the that scaler to accept another coefficient

Coefficients are written from the coefficient interface into a loading FIFO before being transferred into the main coefficient memory for use by the filters. Loading the FIFO must take place during the frame period before it is required. The transferal process from FIFO to coefficient memory takes place during the next vertical blanking period. Following vertical blanking, `intr_coef_fifo_rdy` will be driven high by the scaler. Following the delivery of the final coefficient of a set into the scaler, `intr_coef_fifo_rdy` will be driven low.

CORE Generator – General Purpose Processor (GPP) Mode GUI

Figure 6 shows the GUI in GPP Mode. For more information on the individual parameters, see "GUI Parameters."

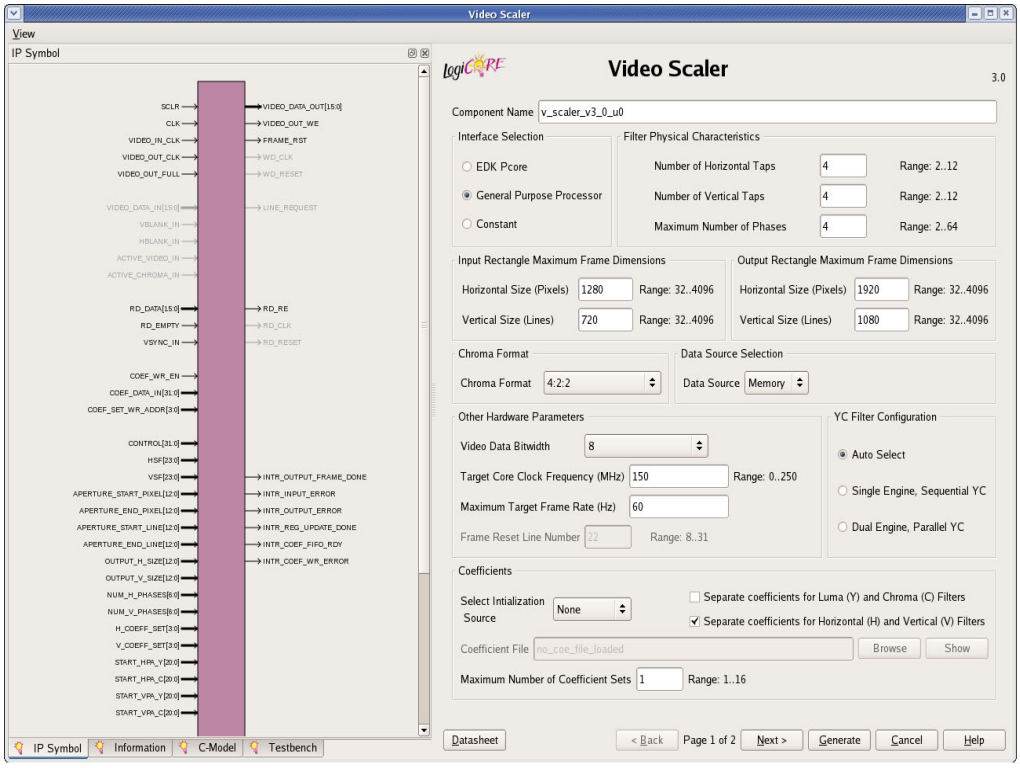

Figure 6: **Video Scaler Graphical User Interface for GPP Mode**

pCore Interface

CORE Generator software may be configured to generate the scaler as a pCore to be built into an EDK project. In this case, CORE Generator creates un-synthesized encrypted VHDL source code. All options in the GUI are greyed-out in this case - the user must parameterize the scaler pCore in the EDK environment. See Figure 7.

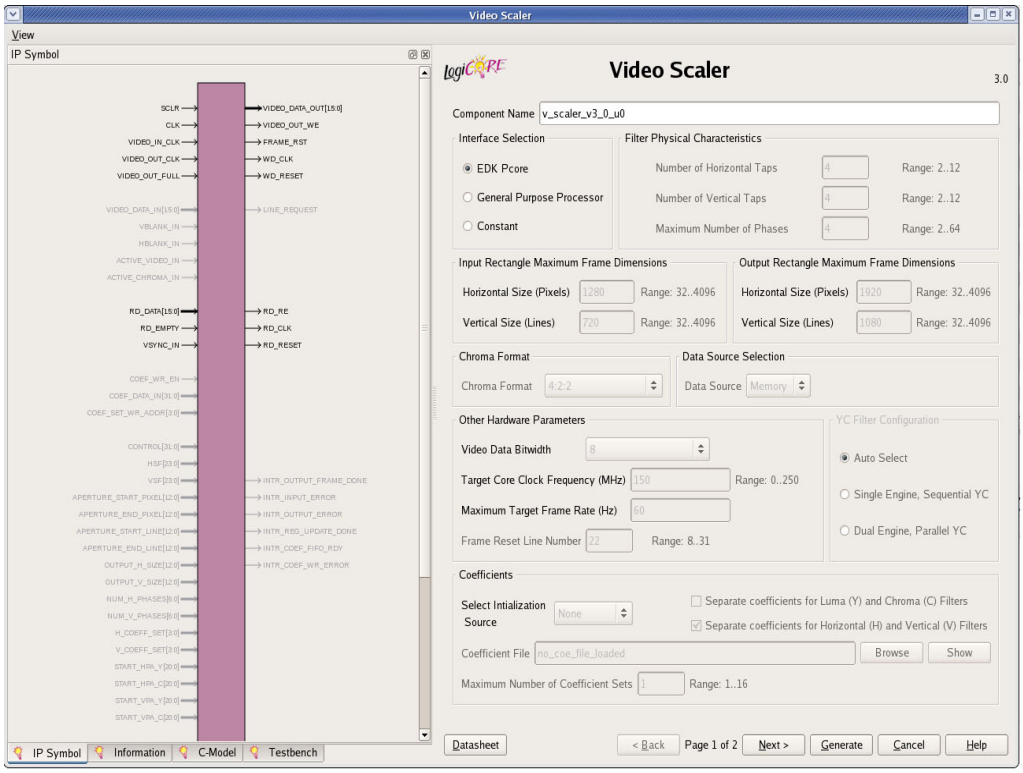

Figure 7: **Video Scaler CORE Generator Graphical User Interface for pCore Mode**

For more information on the individual parameters, see "GUI Parameters."

When in the EDK environment, the Video scaler GUI looks lightly different (see Figure 8) but offers the same options as the GPP CORE Generator GUI.

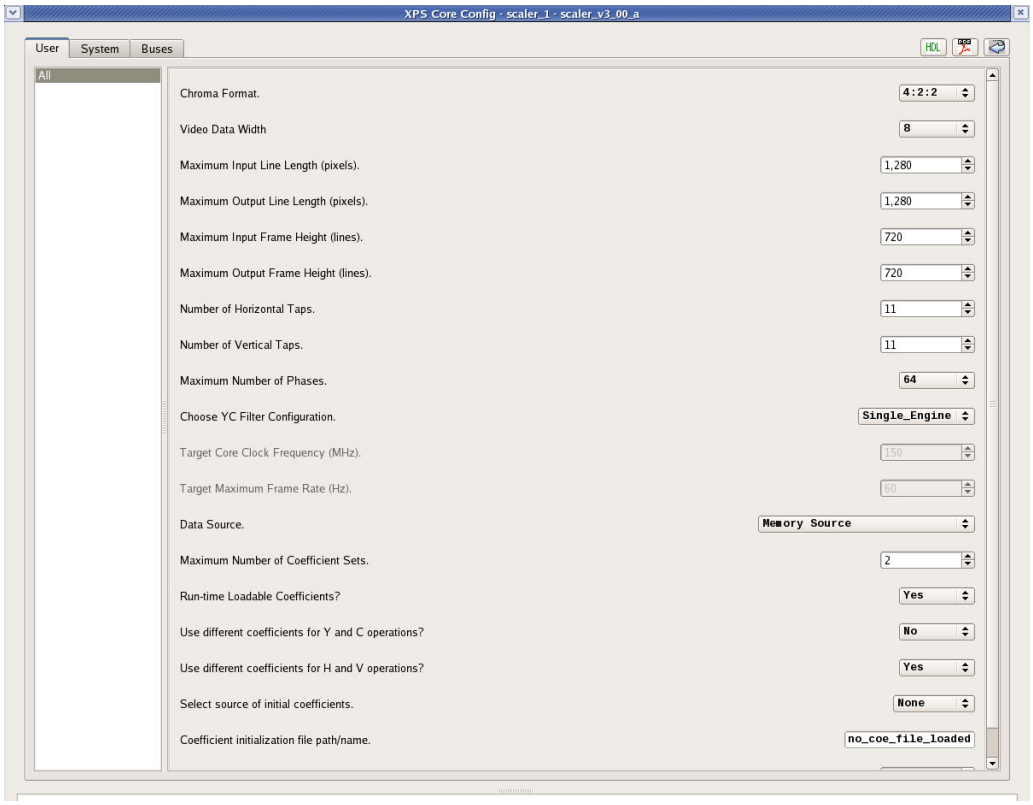

Figure 8: **Video Scaler EDK GUI**

The GPP interface ports that are described in the "General Purpose Processor (GPP) Interface" section exist in the wrapper but are driven by registers on the PLB. These unused ports are greyed-out in the CORE Generator symbol, and are replaced by the PLB interface.

Fully verified MicroBlaze software driver source code is also provided by CORE Generator software for driving all of the control inputs. These are briefly described in Table 3.

Table 3: **pCore Driver Files Delivered from CORE Generator**

File name	Description
\drivers\scaler_v2_00_a\example**example.c**	Examples that demonstrate how to control the scaler core; Up-scaling and downscaling examples included
\drivers\scaler_v2_00_a\src**xscaler.h**	Declaration of all driver functions and driver instance data structure definition
\drivers\scaler_v2_00_a\src**xscaler_hw.h**	Register and bit definition of the scaler device
\drivers\scaler_v2_00_a\src**xscaler.c**	Implementation of general driver functions
\drivers\scaler_v2_00_a\src**xscaler_intr.c**	Implementation of the interrupt-related functions
\drivers\scaler_v2_00_a\src**xscaler_sinit.c**	Implementation of the static initialization function
\drivers\scaler_v2_00_a\src**xscaler_g.c**	Definition of scaler device list, with each element defining parameters for a scaler device, such as base address, vertical tap number, etc.
\drivers\scaler_v2_00_a\src**xscaler_coefs.c**	Definition of all coefficients

More details on how to use the scaler driver software are given in the *Video Scaler v3.0 User Guide* (UG678).

Register Definitions

This pCore provides a memory-mapped interface for the programmable registers within the core, which are described in Table 4. Use of these registers are described in more detail in the *Video Scaler v3.0 User Guide* (UG678).

Note: All registers default to 0x00000000 on power-up or software reset.

Table 4: **Video Scaler Registers Overview**

Address	Name	Read/Write	Description
0x0000	control	R/W	General control register
0x0004	status	R	General readable status register
0x0008	status_error	R	General readable status register for errors
0x000c	status_done	R/W	General read register for status done
0x0010	horz_shrink_factor	R/W	Horizontal Shrink Factor
0x0014	vert_shrink_factor	R/W	Vertical Shrink Factor
0x0018	aperture_horz	R/W	aperture_start_pixel Location of first subject pixel in input line, relative to first active pixel in that line aperture_end_pixel Location of final subject pixel in input line, relative to first active pixel in that line
0x001c	aperture_vert	R/W	aperture_start_line Location of first subject line in input image, relative to first active line in that image aperture_end_line Location of final subject line in input image, relative to first active line in that image

Table 4: **Video Scaler Registers Overview *(Cont'd)***

Address	Name	Read/Write	Description
0x0020	output_size	R/W	output_h_size Width of output image (pixels) output_v_size Height of the outuput image (lines)
0x0024	num_phases	R/W	num_h_phases Number of phases of coefficients in current horizontal filter set num_v_phases Number of phases of coefficients in current vertical filter set
0x0028	coeff_sets	R/W	hcoeffset Active coefficient set to use in horizontal filter operation vcoeffset Active coefficient set to use in vertical filter operation
0x002c	start_hpa_y	R/W	Fractional value used to initialize horizontal accumulator at rectangle left edge for luma
0x0030	start_hpa_c	R/W	Fractional value used to initialize vertical accumulator at rectangle top edge for luma
0x0034	start_vpa_y	R/W	Fractional value used to initialize horizontal accumulator at rectangle left edge for chroma
0x0038	start_vpa_c	R/W	Fractional value used to initialize vertical accumulator at rectangle top edge for chroma
0x003c	coef_write_set_addr	R/W	Coefficient set write address to indicate which coefficient bank to write
0x0040	coef_values	W	Coefficient values to write
0x0100	Software_Reset	W	Writing a SOFT_RESET value to this register resets the software registers and the Video Scaler IP core. The SOFT_RESET value is determined by EDK.
0x021C	GIER	R/W	Global Interrupt Enable Register
0x0220	ISR	R/W	Interrupt Status Register; read to determine the source of the interrupt, write to clear the interrupt
0x0228	IER	R/W	Interrupt Enable Register; 0 to mask out an interrupt, 1 to enable an interrupt

Constant (Fixed Mode) Interface

This option generates a netlist whose scaling parameters are predetermined on page 2 of the CORE Generator GUI (Figure 10). This option removes the need for the user to control the inputs dynamically if a fixed-mode scaler is desired, and reduces resource usage. See Figure 9 and Figure 10.

In this mode, the coefficients are hard-coded into the netlist. The user must provide the desired coefficients as an external .coe file, specifying this file in the CORE Generator GUI.

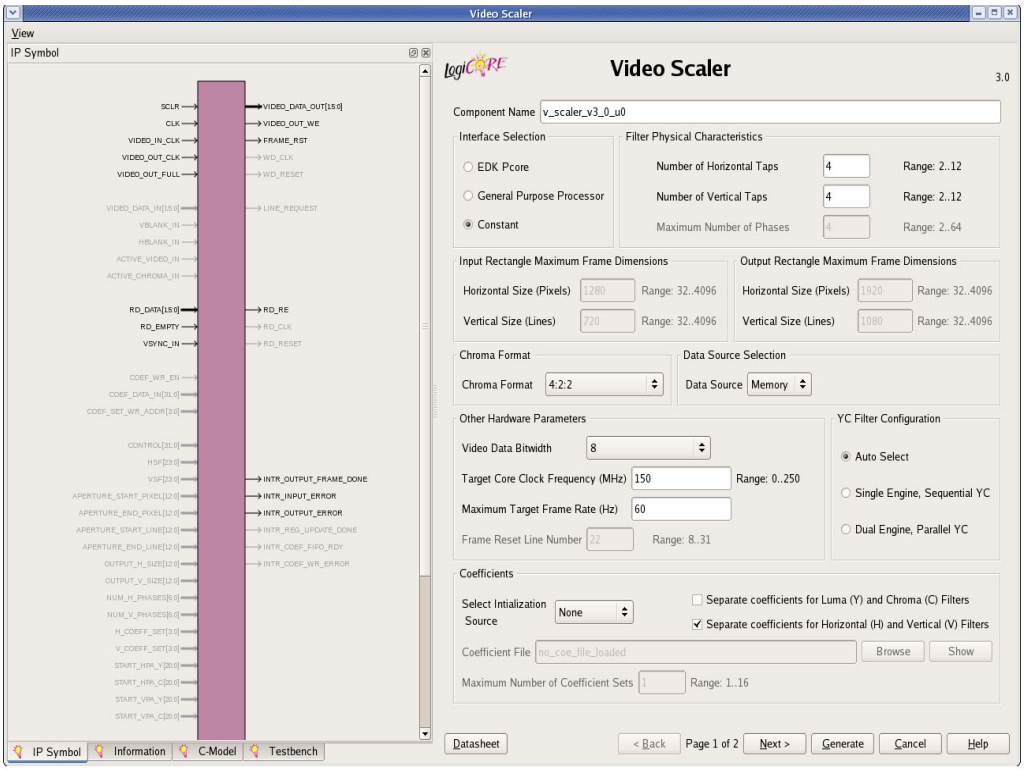

Figure 9: **Video Scaler Graphical User Interface for Constant Mode (page 1)**

For more information on the individual parameters, see "GUI Parameters."

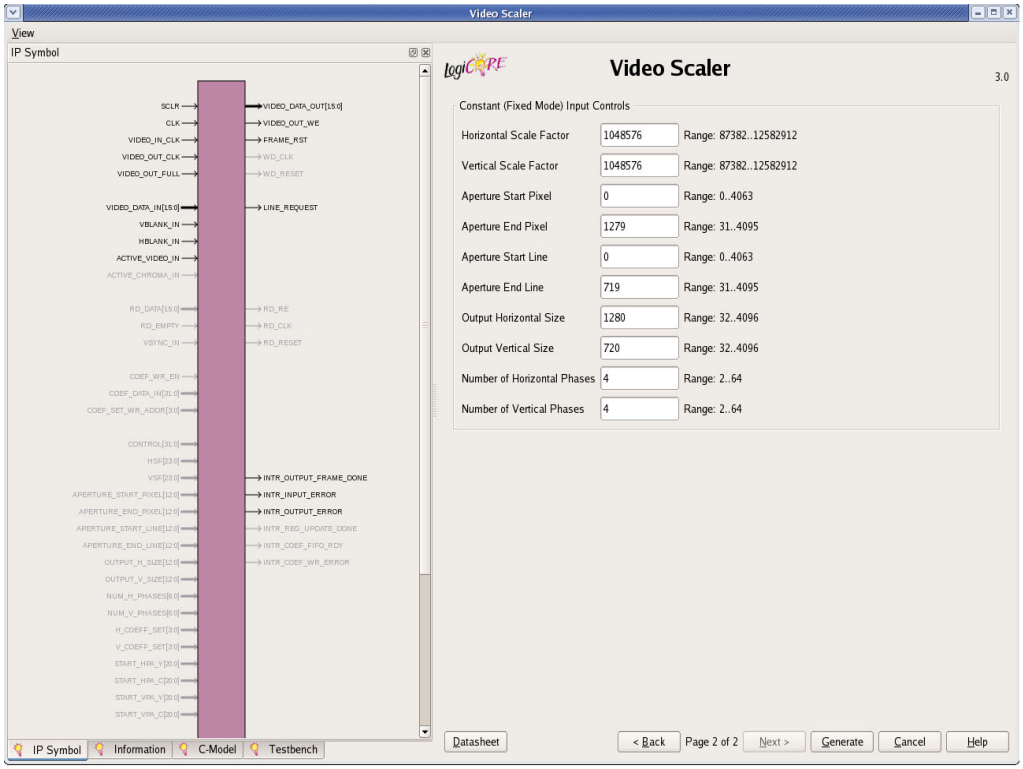

Figure 10: **Video Scaler Graphical User Interface for Constant Mode (page 2)**

Constant-Mode GUI Parameters

Horizontal Scale Factor, Vertical Scale Factor (for Constant Mode only)**:** Specify, as unsigned integers, the 24-bit numbers that represent the desired fixed scale factors. Calculation of these values is described as HSF, VSF in the "Control Values" section in the *Video Scaler v3.0 User Guide* (UG678).

Aperture Start Pixel, Aperture End Pixel, Aperture Start Line, Aperture End Line (for Constant Mode only)**:** See the "Control Values" section in the *Video Scaler v3.0 User Guide* (UG678). These parameters define the size and location of the input rectangle. They are explained in detail in the "Scaler Aperture" section in UG678. The cropping feature is only available when using Live Video data-source. In Memory mode, Aperture Start Pixel and Aperture Start Line are fixed at 0.

Output Horizontal Size, Output Vertical Size (for Constant Mode only)**:** These two parameters define the size of the output rectangle. They do not determine anything about the target video format. The user must determine what do with the scaled rectangle that emerges from the scaler core.

Number of Horizontal/Vertical Phases (for Constant Mode only)**:** Non power-of-two numbers of phases are supported.

Coefficient File Input (for Constant Mode only)**:** The user must specify a .coe file so that the coefficients are hard-coded into the netlist. This is described in more detail in the "Control Interface" section of the *Video Scaler v3.0 User Guide* (UG678).

Data Source: Memory

When this mode is selected, the scaler asserts `RD_RE` for each line of data in a frame. One `video_in_clk` cycle later, valid input data is expected on the `RD_DATA` input bus. All the preceding GUI images show the Memory data source settings. Memory mode is a simple, generic interface. It may be used in many different user systems, or adapted easily to do so. It should be noted that it has been developed to cleanly interface with an external memory block via the MPMC and a VFBC port, using the Xilinx VDMA, as shown in Figure 11. It has been tested in this environment.

Documentation on the Xilinx VDMA is available from the VDMA product page.

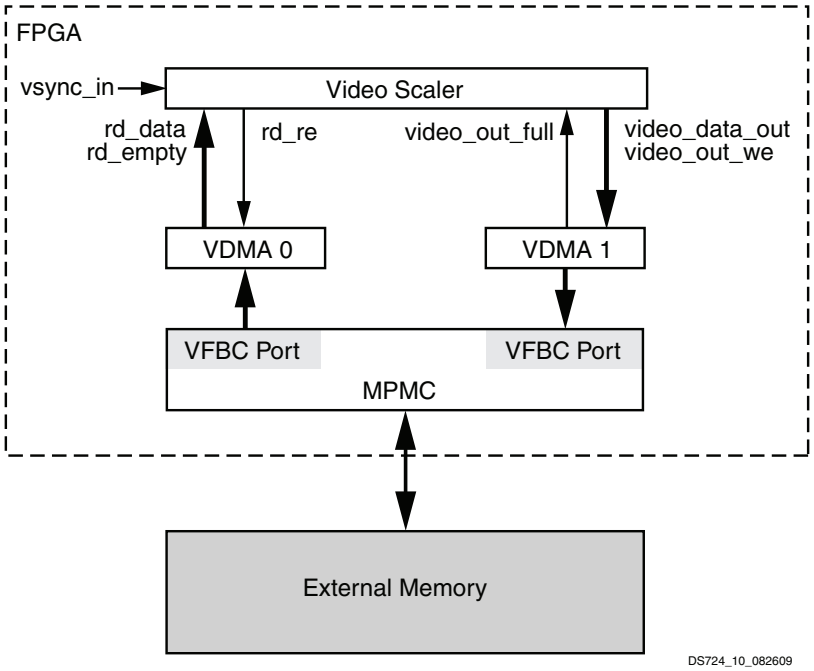

Figure 11: **Memory Source Use Model**

The user may alternatively elect to build an internal buffering solution – the size and nature of the internal buffer depend heavily upon the user's worst-case scaling requirements.

The scaler can throttle the input feed, hence making it easier to upscale parts of an input image. This mode is intended to interface with a memory controller such as the Xilinx MPMC. The memory controller may assert `RD_EMPTY` when a full line of data is not immediately available.

See the MPMC data sheet for additional information.

Data Source: Live

When this mode is selected, the scaler expects valid video data aligned with the `ACTIVE_VIDEO_IN` signal. Horizontal and Vertical synchronization signals must also be provided on the `hblank_in` and `vblank_in` pins. This usage is shown in Figure 12. A snapshot of the CORE Generator GUI page for this mode is shown in Figure 13.

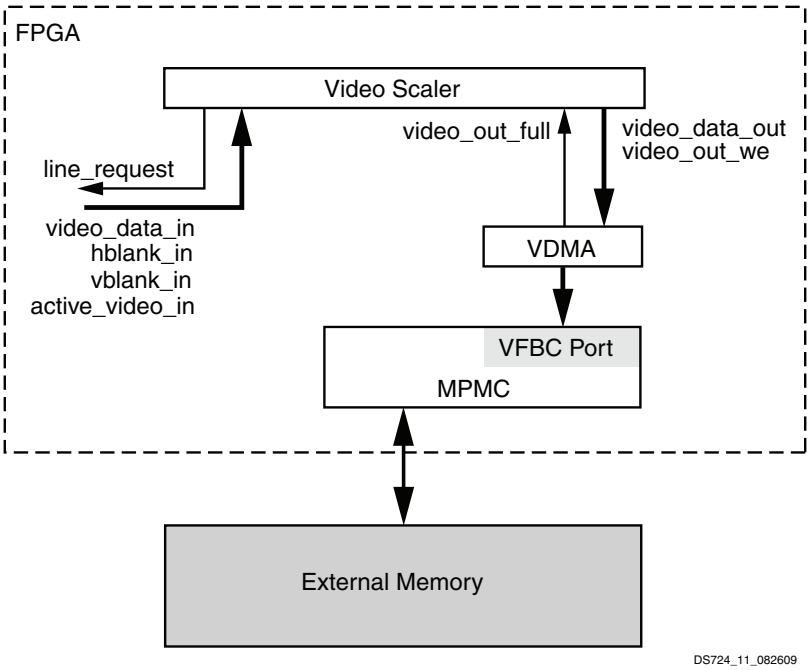

Figure 12: **Live Source Use Model**

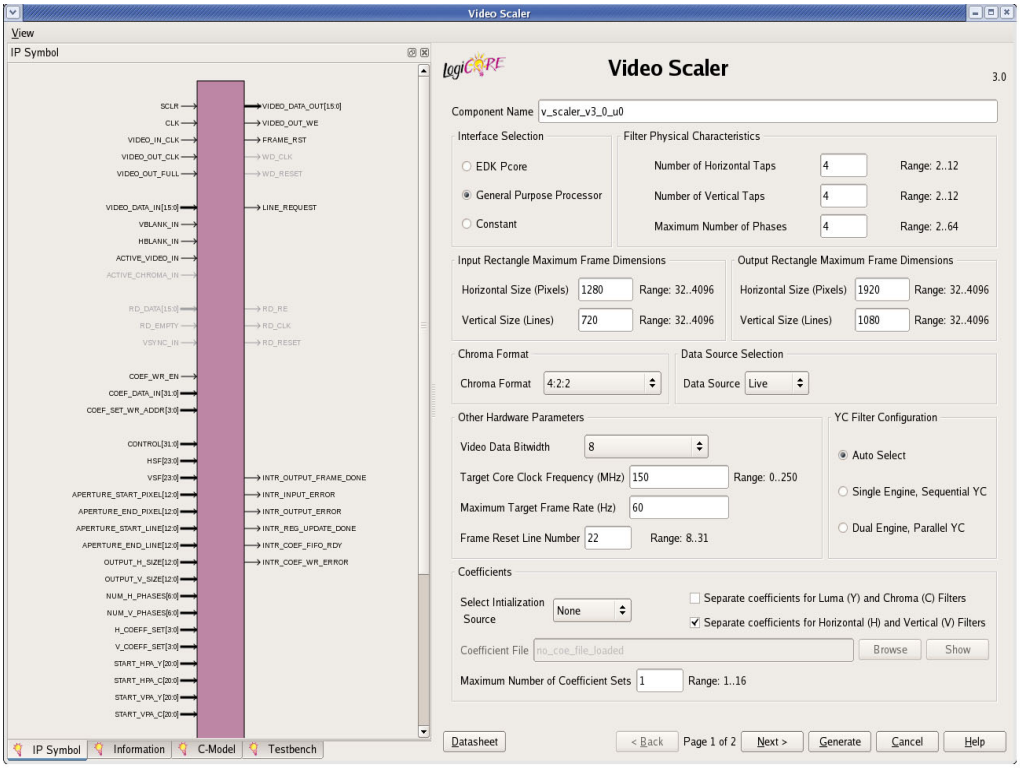

Figure 13: **Video Scaler Graphical User Interface for Live Data Source**

For more information on the individual parameters, see "GUI Parameters."

Live Data Source – Input Control Signals and Timing

Valid video data is written into the input line-buffer, using `active_video_in`, shown in Figure 14. `active_video_in` must remain in a high state for the duration of the active input line.

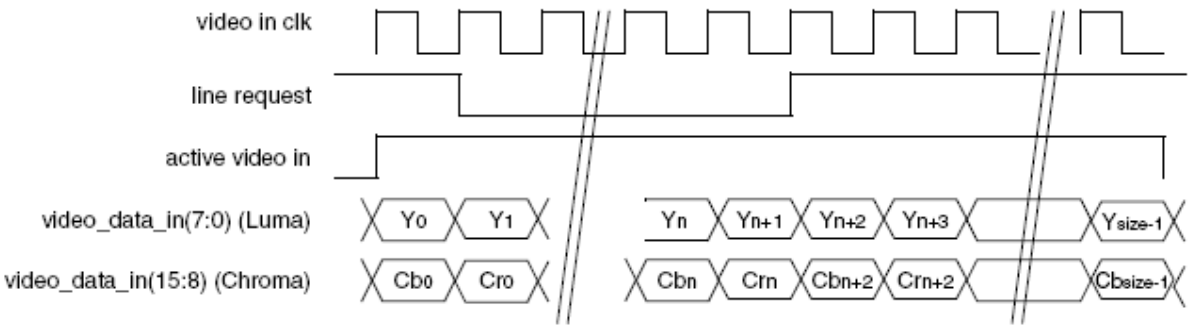

Figure 14: **Scaler 4:2:2 Input Timing**

An additional input, `active_chroma_in`, is required in the 4:2:0 case. This must be asserted high on all lines for 4:2:2, but only for alternate lines for 4:2:0, as shown in Figure 15. There must be valid data at line 1 (counting from line 1; *not* line 0) and at every odd numbered line after line 1.

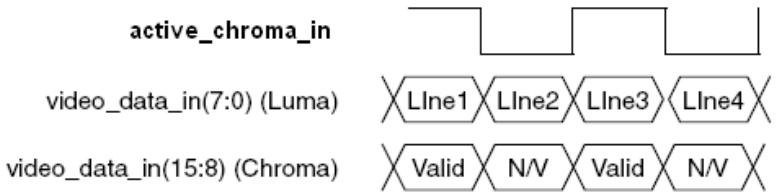

Figure 15: **Scaler 4:2:0 Input Chroma Validation**

Output Signals and Timing

When a line of data becomes available in the output buffer, and the `video_out_full` flag is low, the `video_out_we` flag is asserted as shown in Figure 16, and data is driven out. The target must deassert `video_out_full` when it is ready to accept the entire line.

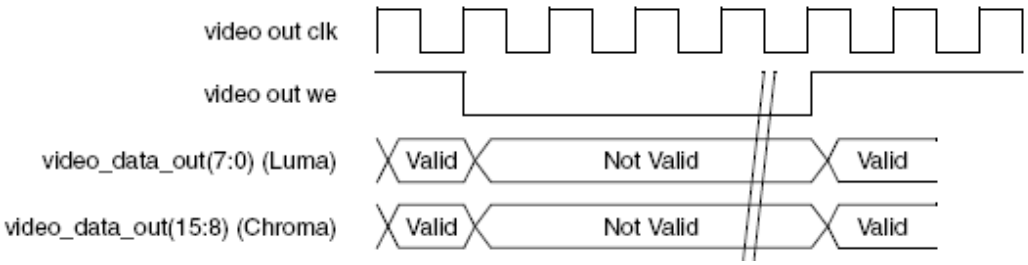

Figure 16: **Scaler Output Timing**

Core Resource Utilization

Resource tables (Table 5 through Table 8) are given here for each of the supported families. They address a number of datapoints in each case. They are derived from post-synthesis reports, and may change during MAP and PAR.

Notes:

1. Figures are given for “Live” Video Source. When using “Memory” source for Virtex-5 device, add approximately 100 FFs and 150 LUTs (all families).
2. When using pCore interface, add approximately 600 FFs and 850 LUTs (all families).

Table 5: **Virtex-5 Resource Estimates**

Control Interface	Chroma Format	Data Width	HW Configuration	HTaps	VTaps	Max Input Line Length	Max Output Line Length	Max Phases	Max Coef Sets	Block RAMs	DSP48s	LUTs	FFs
GPP	4:2:2	8	YC Single-Engine	4	4	1920	1920	4	8	10	12	1140	1741
GPP	4:2:2	8	YC Single-Engine	5	5	1920	1920	4	8	11	14	1154	1806
GPP	4:2:2	8	YC Single-Engine	12	12	1920	1920	4	8	18	28	1271	2133
GPP	4:2:2	8	YC Single-Engine	4	4	512	512	4	8	6	12	1044	1685
GPP	4:2:2	10	YC Single-Engine	4	4	1920	1920	4	8	13	12	1148	1779
GPP	4:2:2	12	YC Single-Engine	4	4	1920	1920	4	8	14	12	1156	1817
GPP	4:2:2	8	YC Dual-Engine	4	4	1920	1920	4	8	10	24	1539	3292
GPP	RGB/4:4:4	8	Triple Engine	4	4	1920	1920	4	8	14	28	1169	2567
GPP	4:2:0	8	YC Single-Engine	4	4	1920	1920	4	8	16	12	1219	1761
Constant	4:2:2	8	YC Single-Engine	4	4	128	256	4	n/a	6	10	846	1279

Table 6: **Spartan-6 Resource Estimates**

Control Interface	Chroma Format	Data Width	HW Configuration	HTaps	VTaps	Max Input Line Length	Max Output Line Length	Max Phases	Max Coef Sets	Block RAMs	DSP48s	LUTs	FFs
GPP	4:2:2	8	YC Single-Engine	4	4	1920	1920	4	8	17	12	1261	1747
GPP	4:2:2	8	YC Single-Engine	5	5	1920	1920	4	8	20	14	1293	1883
GPP	4:2:2	8	YC Single-Engine	12	12	1920	1920	4	8	37	28	1521	2850
GPP	4:2:2	8	YC Single-Engine	4	4	512	512	4	8	7	12	1169	1704
GPP	4:2:2	10	YC Single-Engine	4	4	1920	1920	4	8	21	12	1273	1790
GPP	4:2:2	12	YC Single-Engine	4	4	1920	1920	4	8	24	12	1283	1826
GPP	4:2:2	8	YC Dual-Engine	4	4	1920	1920	4	8	17	24	1766	2774
GPP	RGB/4:4:4	8	Triple Engine	4	4	1920	1920	4	8	24	28	1407	2595
GPP	4:2:0	8	YC Single-Engine	4	4	1920	1920	4	8	33	12	1325	1769
Constant	4:2:2	8	YC Single-Engine	4	4	128	256	4	n/a	9	12	862	1235

Table 7: **Spartan-3A DSP Resource Estimates**

Control Interface	Chroma Format	Data Width	HW Configuration	HTaps	VTaps	Max Input Line Length	Max Output Line Length	Max Phases	Max Coef Sets	Block RAMs	DSP48s	LUTs	FFs
GPP	4:2:2	8	YC Single-Engine	4	4	1920	1920	4	8	17	12	1330	1735
GPP	4:2:2	8	YC Single-Engine	5	5	1920	1920	4	8	19	14	1333	1800
GPP	4:2:2	8	YC Single-Engine	12	12	1920	1920	4	8	33	28	1463	2127
GPP	4:2:2	8	YC Single-Engine	4	4	512	512	4	8	9	12	1238	1683
GPP	4:2:2	10	YC Single-Engine	4	4	1920	1920	4	8	23	12	1338	1773
GPP	4:2:2	12	YC Single-Engine	4	4	1920	1920	4	8	24	12	1346	1811
GPP	4:2:2	8	YC Dual-Engine	4	4	1920	1920	4	8	17	24	1751	2725
GPP	RGB/4:4:4	8	Triple Engine	4	4	1920	1920	4	8	24	28	1293	2562
GPP	4:2:0	8	YC Single-Engine	4	4	1920	1920	4	8	33	12	1377	1750
Constant	4:2:2	8	YC Single-Engine	4	4	128	256	4	n/a	6	12	970	1335

Table 8: **Virtex-6 Resource Estimates**

Control Interface	Chroma Format	Data Width	HW Configuration	HTaps	VTaps	Max Input Line Length	Max Output Line Length	Max Phases	Max Coef Sets	Block RAMs	DSP48s	LUTs	FFs
GPP	4:2:2	8	YC Single-Engine	4	4	1920	1920	4	8	10	12	1250	1773
GPP	4:2:2	8	YC Single-Engine	5	5	1920	1920	4	8	12	14	1282	1941
GPP	4:2:2	8	YC Single-Engine	12	12	1920	1920	4	8	22	28	1510	2940
GPP	4:2:2	8	YC Single-Engine	4	4	512	512	4	8	6	12	1154	1721
GPP	4:2:2	10	YC Single-Engine	4	4	1920	1920	4	8	13	12	1260	1809
GPP	4:2:2	12	YC Single-Engine	4	4	1920	1920	4	8	14	12	1270	1845
GPP	4:2:2	8	YC Dual-Engine	4	4	1920	1920	4	8	10	24	1677	2818
GPP	RGB/4:4:4	8	Triple Engine	4	4	1920	1920	4	8	14	28	1277	2666
GPP	4:2:0	8	YC Single-Engine	4	4	1920	1920	4	8	14	12	1269	1783
Constant	4:2:2	8	YC Single-Engine	4	4	128	256	4	n/a	6	12	871	1316

Performance

- Spartan-3A DSP (-4), Spartan-6 (-2): 150 MHz (ISE® 12.3 tools)
- Virtex-5 (-1), Virtex-6 (-1): 225 MHz (ISE® 12.3 tools)
- Virtex-6 (-1):250 MHz (12.3 tools)

Support

Xilinx provides technical support for this LogiCORE product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Licensing Information

License Options

The Video Scaler provides three licensing options. After installing the required Xilinx ISE software and IP Service Packs, choose a license option.

Simulation Only

The Simulation Only Evaluation license key is provided with the Xilinx CORE Generator™ tool. This key lets you assess core functionality with either the example design provided with the Video Scaler core, or alongside your own design and demonstrates the various interfaces to the core in simulation. (Functional simulation is supported by a dynamically-generated HDL structural model.)

Full System Hardware Evaluation

The Full System Hardware Evaluation license is available at no cost and lets you fully integrate the core into an FPGA design, place-and-route the design, evaluate timing, and perform functional simulation of the Video Scaler core.

In addition, the license key lets you generate a bitstream from the placed-and-routed design, which can then be downloaded to a supported device and tested in hardware. The core can be tested in the target device for a limited time before timing out (ceasing to function), at which time it can be reactivated by reconfiguring the device.

Full

The Full license key is available when you purchase the core and provides full access to all core functionality both in simulation and in hardware, including:

- Functional simulation support
- Full implementation support including place-and route-and bitstream generation
- Full functionality in the programmed device with no time outs

Obtaining Your License Key

This section contains information about obtaining a simulation, full system hardware, and full license keys.

Simulation License

No action is required to obtain the Simulation Only Evaluation license key; it is provided by default with the Xilinx CORE Generator™ software.

Full System Hardware Evaluation License

1. Navigate to the product page for this core:
   www.xilinx.com/products/ipcenter/EF-DI-VID-SCALER.htm
2. Click Evaluate.
3. Follow the instructions to install the required Xilinx ISE software and IP Service Packs.

Obtaining a Full License

To obtain a Full license key, you must purchase a license for the core. After doing so, click the "Access Core" link on the Xilinx.com IP core product page for further instructions.

Installing Your License File

The Simulation Only Evaluation license key is provided with the ISE CORE Generator system and does not require installation of an additional license file. For the Full System Hardware Evaluation license and the Full license, an email will be sent to you containing instructions for installing your license file. Additional details about IP license key installation can be found in the ISE Design Suite Installation, Licensing and Release Notes document.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
04/24/09	1.0	Initial Xilinx release.
09/16/09	2.0	Updated for core version 2.0.
04/19/10	2.1	Updated for core version 2.1.
09/21/10	3.0	Updated for core version 3.0.

Notice of Disclaimer