LogiCORE IP Video On-Screen Display v2.0

DS837 March 1, 2011 **Product Specification**

Introduction

The Xilinx Video On-Screen Display LogiCORE™ IP provides a flexible video processing block for alpha blending and compositing as well as simple text and graphics generation. Support for up to eight layers using a combination of external video inputs (from frame buffer) and internal graphics controllers (including text generators) is provided. Supports image sizes up to 4kx4k with YUVA 4:4:4 or 4:2:2 and RGBA image formats up to 60fps. The core is programmable through a comprehensive register interface for setting and controlling screen size, background color, layer position, and more using logic or a microprocessor. A comprehensive set of interrupt status bits is provided for processor monitoring. The LogiCORE IP is provided with two different interfaces: General Purpose Processor and EDK pCore AXI4-Lite (including device driver).

LogiCORE IP Facts Table					
Core Specifics					
Supported Device Family[1]	Spartan®-3A DSP, Spartan-6, Virtex®-5, Virtex-6				
Supported User Interfaces	General Purpose Processor (GPP), EDK pCore AXI4-Lite				
	Resources				**Frequency**
Configuration	**LUTs**	**FFs**	**DSP Slices**	**Block RAMs**	**Max. Freq.**
Spartan-3A DSP	See Table 4				200 MHz
Spartan-6	See Table 5				150 MHz
Virtex-5	See Table 6				225 MHz
Virtex-6	See Table 7				225 MHz
Provided with Core					
Documentation	Product Specification, User Guide				
Design Files	Netlist, EDK pCore files, C Driver				
Example Design	Not Provided				
Test Bench	VHDL				
Constraints File	Not Provided				
Simulation Model	Bit Accurate C Model				
Tested Design Tools					
Design Entry Tools	CORE Generator™, Platform Studio (XPS)				
Simulation	ModelSim v6.6d, Xilinx ISIM 13.1				
Synthesis Tools	ISE® 13.1				
Support					
Provided by Xilinx, Inc.					

1. For a complete listing of supported devices, see the release notes for this core.

Features

- Selectable processor interface
  - EDK pCore (AXI4-Lite)
  - General Purpose Processor
- Supports video frame sizes up to 4096x4096 pixels
- Supports video frame rates up to 60fps (includes all HD rates such as 720p60 and 1080P60)
- Supports the following color-spaces with or without alpha channel (color-spaces can be mixed with addition of chroma-resampler and color-space converter cores)
  - RGB
  - RGBa
  - YUV-4:4:4
  - YUVa-4:4:4
  - YUV-4:2:2
  - YUVa-4:2:2
  - YUV-4:2:0
  - YUVa-4:2:0
- Supports 2 or 3 color component channels
- Supports 8, 10 or 12 bits per color component channel (equivalent supported bits per pixel: 16, 20, 24, 30 or 36 bits)
- Supports alpha-blending 8 video/graphics layers
- Provides programmable background color
- Provides programmable layer position, size and z-plane order
- Generates filled and outlined transparent boxes
- Generates text with 1-bit or 2-bit per pixel color depth
- Provides configurable internal text string memory
- Provides configurable internal font memory for 8x8 or 16x16 pixel fixed distance fonts
- Provides scaling text by 1x, 2x, 4x or 8x
- Supports graphics color palette of 16 or 256 colors
- Supports easy integration with other Xilinx Video IP Cores including the VFBC, VDMA and Video Timing Controller

Applications

Applications range from broadcast and consumer to automotive, medical and industrial imaging and can include:

- Video Surveillance
- Machine Vision
- Video Conferencing
- Set-top box displays

Overview

The Xilinx Video On-Screen Display LogiCORE IP produces output video from multiple external video sources as well as from multiple internal graphics controllers. Each graphics controller generates simple text and graphics overlays. Each video and graphics source is assigned an image layer. Up to eight image layers can be dynamically positioned, resized, brought forward or backward and combined using alpha-blending.

Alpha-blending is the convex combination of two image layers allowing for transparency. Each layer in the OSD has a definite Z-plane order, or conceptually, each layer resides closer or farther from the observer having a different depth. Thus, the image and the image directly "over" it are blended. The order and amount of blending is programmable in real-time.

An example Xilinx Video On-Screen Display Output is shown in Figure 1.

Figure 1: **Example of OSD Output**

Here we show an example OSD output with multiple video and graphics layers. The three video layers (Video 1, 2 and 3) can be still images or live video and are combined with transparency to the programmable background color. Simple boxes and text are generated with one or multiple internal graphics controllers (as shown with the yellow text and menu buttons) and also blended with the other layers. One other video layer (the Xilinx Logo), can be generated from on-chip or external memory, showing that the OSD output can be easily extended with external logic, a microprocessor or memory storage.

Basic Architecture

The Xilinx On-Screen Display LogiCORE IP reads 2D video image data in raster order from up to eight sources. Each data source can be configured to be a FIFO interface or an internal graphics controller. If a FIFO interface is selected, ports on the OSD are available for connecting to and reading data from the Xilinx Video Frame Buffer Controller or from the Video Direct Memory Access Controller. These ports are also generic enough for easy integration with any FIFO. If an internal graphics controller is selected to be a source, then the OSD automatically handles interfacing to each graphics controller.

Pixel data from each source is combined using alpha-blending. The resultant output is a 2D video image stream that can be presented to one of two output interfaces – to a FIFO interface or to a Xilinx Streaming Video Interface (XSVI). If a FIFO interface is used, the output FIFO almost full flag and the input FIFO empty flags (from each FIFO input source) will halt operation of the OSD until cleared. If an XSVI interface is used, the data is presented on the output along with video synchronization signals (Vertical Blank, Horizontal Blank, etc.). The data is presented at the time required by the synchronization signals, and thus, the input FIFO empty flags are ignored. Care must be taken to make sure each input FIFO does not underflow.

Only one output interface can be selected. The OSD cannot drive both interfaces at the same time. See the Migrating to the EDK pCore AXI4-Lite Interface section for more information on both the output FIFO (VFBC Write Data Interface) and the output XSVI interfaces.

The Xilinx On-Screen Display also requires an input Xilinx Streaming Video Interface. The OSD does not receive video data from this interface. This interface is only used to receive the horizontal/vertical blank and sync signals as well as an active video signal. The horizontal and vertical sync signals are not used internally to the OSD and are only delayed and presented on the output XSVI interface for use with driving external display hardware. The same is true for the active video signal and is used only to delineate the presence of valid output. Only the horizontal and vertical blank signals are used internally to the OSD for control and synchronization of frame data.

See the Xilinx Video Timing Controller data sheet for more information on video timing signals.

The use of specific video timing signals requires that the input data is always present when requested (the OSD will not look at the input FIFO empty flags) if an XSVI interface is used. If an output FIFO interface is used, it is required that the input and output FIFO flags do not delay the operation of OSD too often, as this may cause frames to not be completely processed. Specific interrupt status bits will alert such errors.

An example OSD configuration with three data sources (layers) is shown in Figure 2. Data for layer 0 and layer 1 are read from input FIFOs. Data for layer 2 are read from a graphics controller instance.

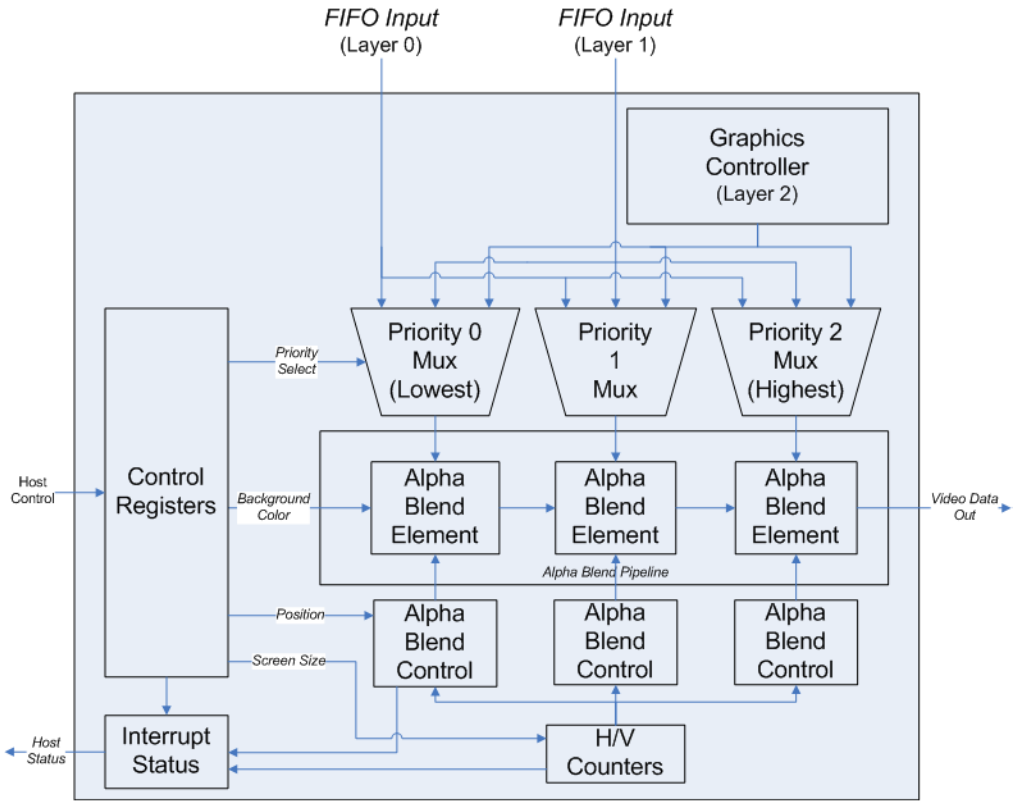

Figure 2: **Example OSD Block Diagram**

In addition to the video data interfaces, the Xilinx On-Screen Display has a control interface for setting registers that control the background color and screen size. The size, (x,y) position and priority (Z-plane order) of each layer can also be configured. Registers for overriding pixel based alpha values with a global alpha and for enabling/disabling layers are also provided.

All control registers can be set dynamically in real time. The OSD internally double-buffers all control registers every frame. Thus, control registers can be updated without introducing artifacts on screen. In addition, the OSD provides a "Register Update Enable" bit in the control register that allows controlling the timing of the double-buffered register updates for further flexibility.

A 32-bit interrupt status register output is also provided that flags internal errors or general events that may require host processor intervention. Interrupt status bits flag events for vertical blanking start and end, frame error, frame complete, input FIFO underflow, output FIFO overflow and graphics controller errors (discussed later).

Alpha-Blending Pipeline

The Xilinx On-Screen Display alpha-blending pipeline includes from one to eight alpha-blending elements connected in succession. Each element blends the pixel data from one layer to the pixel data from the layer underneath, and controls whether a layer is enabled and if pixel-level alpha should be read from the input alpha channel or a global alpha value should be used.

Layer data is blended in the order dictated by the priority setting for each layer in the control registers. The priority values are used to multiplex layer data to the correct alpha-blending element.

A basic flow chart diagram showing the alpha-blending process is shown in Figure 3.

The alpha-blending pipeline architecture takes advantage of the high-performance XtremeDSP™ DSP48 slices available in the target device families. These slices are utilized for multiplication and some addition operations and time-shared efficiently between color component channels.

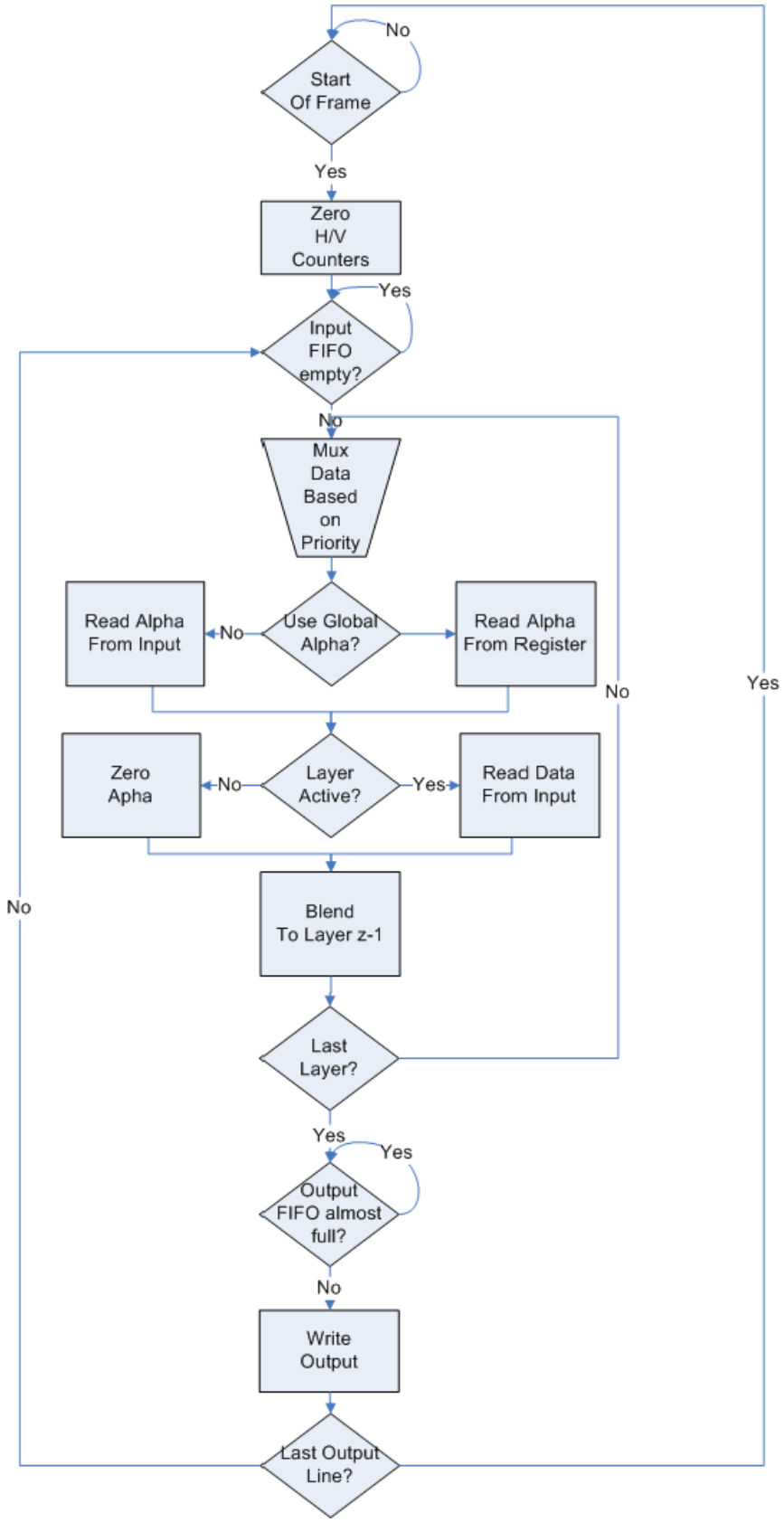

Figure 3: **Alpha-Blending Pipeline Flow Chart**

Graphics Controller

The Xilinx On-Screen Display internal graphics controller can generate two graphics elements –boxes and text strings. Boxes can be drawn filled or outlined. The color, position, size and outline weight of each box are configurable via host control registers (graphics controller host interface). Text strings can be drawn with a scale factor of 1x, 2x, 4x or 8x the original size. The color and position are also configurable.

Figure 4 shows the internal structure of the graphics controller.

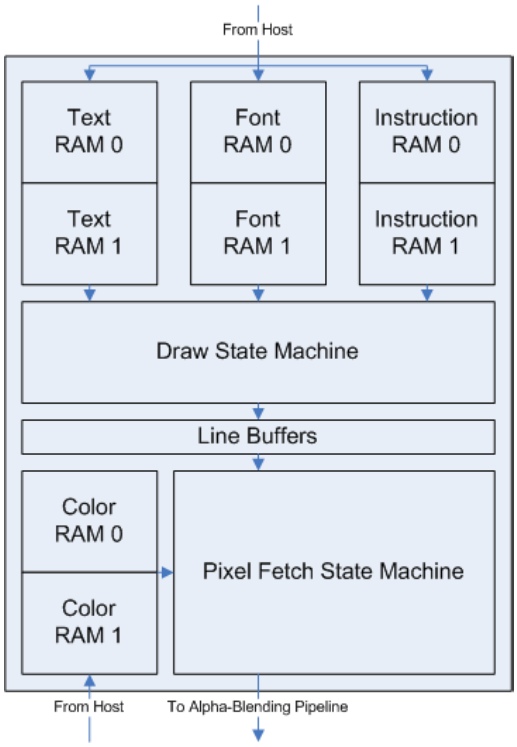

Figure 4: **OSD Graphics Controller Block Diagram**

The graphics controller is configured to draw boxes and text by a host processor. The host processor must write graphics instructions into an Instruction RAM. Each instruction can configure the graphics controller to draw a box, a text string, a combined box/text graphics element or to perform an internal function. The maximum number of instructions is configured with the "Instructions" field of the CORE Generator™ GUI.

During every video line, the draw state-machine fetches instructions from an Instruction RAM and draws multiple graphics elements to a line buffer. A box draw instruction will cause the draw state-machine to draw a box of the selected color to a line buffer. A text draw instruction will cause the draw state-machine to fetch a text string from a Text RAM. This text string is used to fetch character data from a Font RAM. The character data along with the color selected by the instruction is used to write pixels in a line buffer.

The pixel fetch state-machine generates output pixel data. It reads the data in the line buffers and uses this data to select a color from the Color RAM for any given pixel. Output pixel data is generated in real-time in raster order. The color and alpha for each output pixel is decided upon when requested. This eliminates the need for external memory storage. The pixel fetch state-machine never reads from the same line buffer as is being written to by the draw state-machine.

Note that for each memory type (Instruction, Color, Text and Font), there are two memories – RAM 0 and RAM 1. This duplication allows the host processor to write to one memory while the graphics controller is reading from another. This eliminates screen artifacts while the processor is configuring the graphics controller.

Memory boundaries are conceptual only. Some graphics controller memories may be efficiently combined in order to save Block RAM or Distributed RAM storage.

Each graphics controller has a set of parameters that controls its configuration. These parameters affect the size of each memory and the resources used by the Xilinx On-Screen Display. See Table 1, CORE Generator GUI Field Descriptions for more information on the graphics controller parameters.

CORE Generator Graphical User Interface (GUI)

The CORE Generator GUI is shown in Figure 5 and Figure 6. Field descriptions are provided in Table 1. Each field sets a parameter used at build time to configure different hardware options.

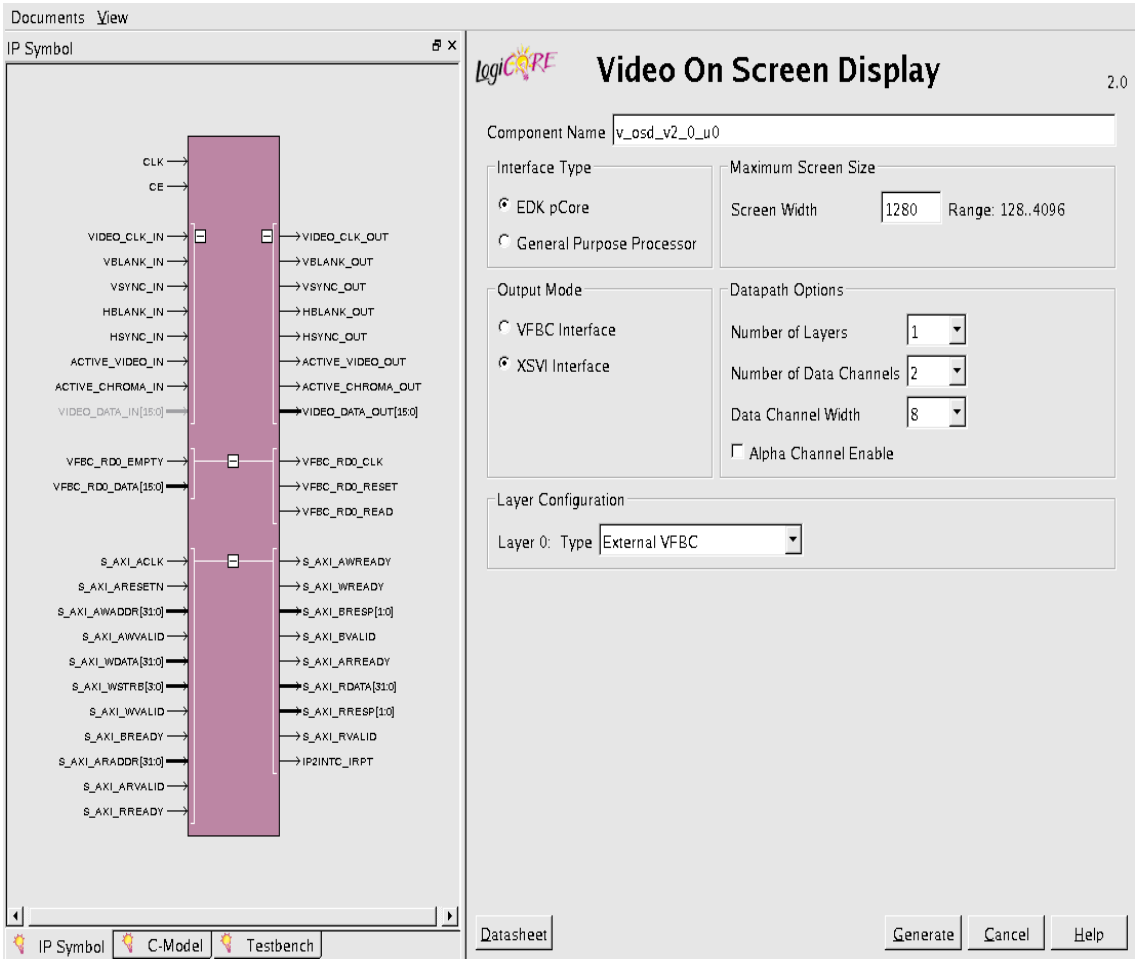

Figure 5: **CORE Generator GUI - Main Window**

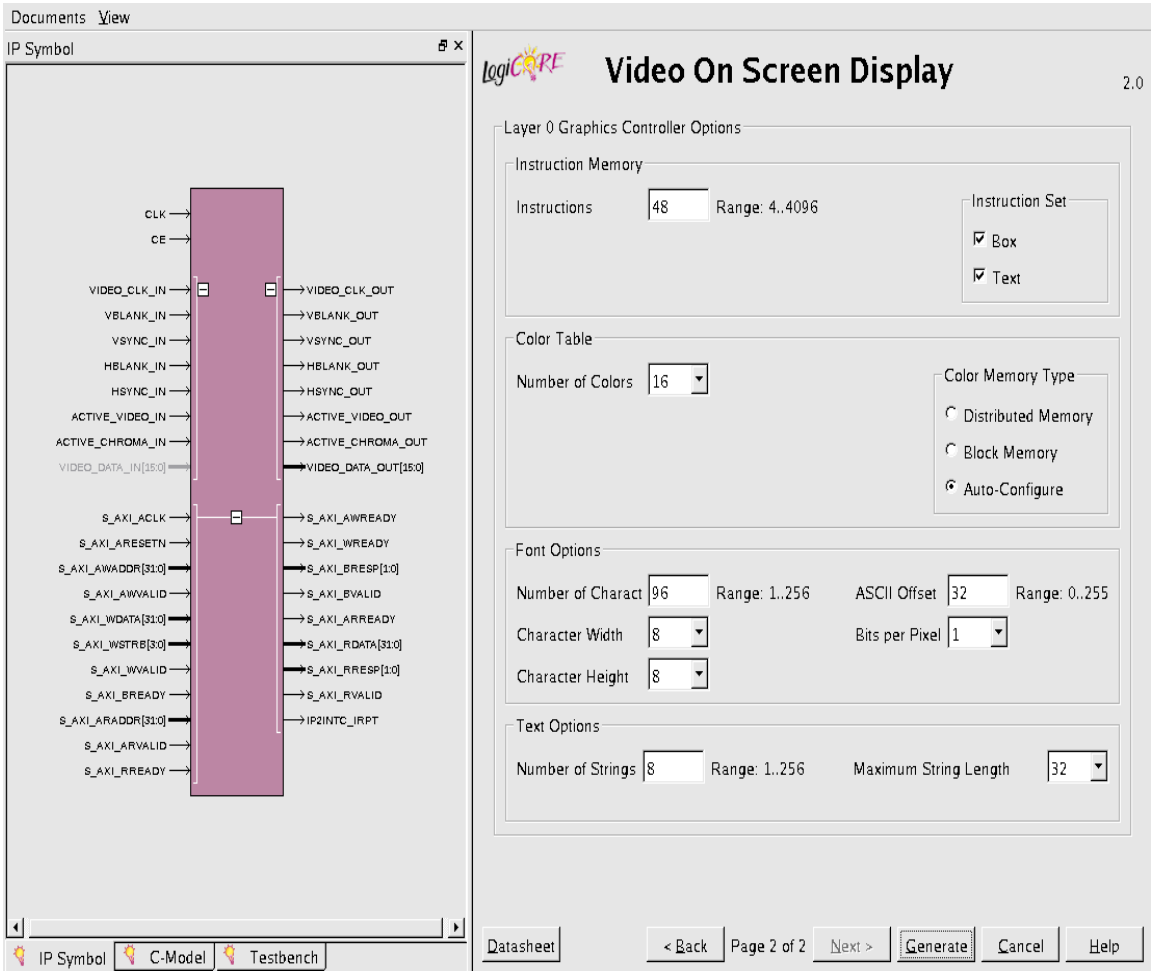

Figure 6: **Core Generator GUI - Graphics Controller Options Window**

Note: The Graphics Controller Options Window is available only if the Layer Type is set to “Internal Graphics Controller.”

Table 1: **CORE Generator GUI Field Descriptions**

Field	Description
Global 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 9and "_".
Interface Type	The On-Screen Display is generated with one of two control interfaces. • EDK pCore Interface: CORE Generator generates the core as a pCore which can be easily imported into an EDK project as a hardware peripheral. The core registers can be programmed in real-time via MicroBlaze™ and the AXI4-Lite Interface. See the EDK pCore AXI4-Lite Interface section. • General Purpose Processor Interface: CORE Generator generates a set of ports that can be used to program the core. See the General Purpose Processor Interface section.
Maximum Screen Size	This field configures the maximum allowed screen size. The Maximum screen width is configurable. Changing this field affects several counters, comparators and memory (Block RAM) usage. Increased screen size increases resource usage. Valid range for Screen Width is {128 .. 4096}.
Output Mode	This field configures the On-Screen Display for one of two output interfaces. • VFBC Interface: The core is generated with the Video Frame Buffer Controller ports enabled. These ports are a generic read FIFO with empty flag. See the Write FIFO Interface section. • XSVI Interface: The core is generated with the Xilinx Streaming Video Interface port enabled. See the Xilinx Streaming Video Interface (XSVI) section. • Corresponds to the `C_OUTPUT_MODE` Parameter of the EDK pCore.
Number of Layers	This field configures the number of layers to alpha blend together. Each layer can be configured to read data from the FIFO inputs or from one of the internal Graphics Controllers. Valid range is (1 .. 8). Corresponds to the `C_NUM_LAYERS` Parameter of the EDK pCore.
Number of Data Channels	This field configures the number of data channels. Valid values are 2 and 3. 2 = Two data channels. Typically used for YUV 4:2:2 data. Either channel can be luminance (Y) or chrominance (Cb,Cr) channels if no internal Graphics Controller is used. If using a Graphics Controller layer, the Graphics Controller outputs luminance (Y) on channel 0 and chrominance (Cb,Cr) on channel 1. 3 = Three data channels. Used for RGB and YUV 4:4:4 data. This mode is color component agnostic. Each channel can be configured for any color component. Corresponds to the `C_NUM_DATA_CHANNELS` Parameter of the EDK pCore.
Data Channel Width	This field configures the data width of each color component channel. Valid values are 8, 10 and 12. Configuring the Data Channel Width and the Number of Data Channels yields an effective bits per pixel of 16, 20, 24, 30 or 36 bits.
Alpha Channel Enable	This field enables the alpha channel on the XSVI and VFBC Read input data buses. Corresponds to the `C_ALPHA_CHANNEL_EN` Parameter of the EDK pCore.

Table 1: **CORE Generator GUI Field Descriptions *(Cont'd)***

Field	Description
Layer Configuration – Layer # Type	These fields configure the type, or data source, of each layer, one field for each layer. Each layer is numbered from 0 to 7. The maximum number of layers is set by the Number of Layers field. Three data sources are valid: • External VFBC: This is a Read FIFO interface with data, read enable and empty ports. Typically connected to the Video Frame Buffer Controller or the Video DMA read FIFO interfaces. See the Read FIFO Interface section. • Internal Graphics Controller: If the layer is configured for this type, then the Read FIFO interface inputs are ignored and all data is generated and read from an internal Graphics Controller. • External XSVI: Selecting this data source, will internally connect this layer data bus to the `video_data_in` bus from the XSVI input interface. The XSVI `video_data_in` bus input can be routed to one or more layers.
Graphics Controller Parameters	
Instructions	This field configures the maximum number of Graphics Controller instructions that can be executed per frame. Increasing this number increases the number of Block RAMs utilized.
Instruction Set	This field configures which instructions are valid for the Graphics Controller implementation. Two instructions are currently configurable: box and text. Other instructions, including NoOp, are always available.
Number of Colors	This field configures the size of the color palette used by the Graphics Controller. Valid values are 16 and 256.
Color Memory Type	This field configures how the color palette is implemented in hardware, as Distributed RAM, as Block RAM or Auto-Configured. In auto-configuration mode, distributed RAM will be used if the color palette is small enough. The RAM type can be overridden if it is known which type is preferred for the application.
Number of Characters	This field configures the number of characters to be stored within the internal Font RAM. Valid values are 1 to 256. This field, along with the Character Width, Character Height, ASCII Offset and Bit per Pixel fields, affects the overall size of the Font RAM.
Character Width	This field configures the width of each character. The width is in pixels. Valid values are 8 and 16.
Character Height	This field configures the height of each character. The height is in video lines. Valid values are 8 and 16.
ASCII Offset	This field configures the ASCII value of the first location in the Font RAM. This is useful if it is known that certain ASCII values will not be used.
Bits per Pixel	This field configures the bits per pixel of each character. Valid values are 1 and 2. 1 = One bit per pixel. This yields a foreground and a background color for each character. 2 = Two bits per pixel. This allows each character pixel to be programmed to one of four different colors.

Table 1: **CORE Generator GUI Field Descriptions *(Cont'd)***

Field	Description
Number of Strings	This field configures the maximum number of strings to be stored within the Text RAM. This field, along with the Maximum String Length field, affects the overall size of the Text RAM. The maximum number of strings cannot exceed 256.
Maximum String Length	This field configures the maximum string length allowed for each string within the Text RAM. Valid values are 32, 64, 128 and 256.

Migrating to the EDK pCore AXI4-Lite Interface

The Video On-Screen Display v2.0 changed from the PLB processor interface to the EDK pCore AXI4-Lite interface. As a result, all of the PLB-related connections have been replaced with an AXI4-Lite interface. For more information, see:

http://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf

Port Descriptions

Table 2 shows the I/O signals on the Xilinx Video On-Screen Display.

Table 2: **Port Descriptions**

Port Name	Dir	Width	Description
clk	I	1	CORE CLOCK Core clock (active high edge).
sclr	I	1	SYNCHRONOUS CLEAR/RESET System synchronous reset (active high). Asserting `sclr` synchronously with `clk` resets the Xilinx On- Screen Display internal state machines. `sclr` has priority over `ce`.
ce	I	1	CLOCK ENABLE Used to halt processing and hold current values.
Input Xilinx Streaming Video Interface			
video_clk_in	I	1	INPUT XSVI CLOCK This is the standard clock on the XSVI interface. This input is only present on the pCore Interface.
vblank_in	I	1	INPUT VERTICAL BLANKING PERIOD SIGNAL Denotes the video lines during which no active video pixel is present. Can be active high or active low polarity. Used internally for synchronization. The known polarity of this input must be set in the OSD Control register before enabling the OSD. The minimum requirements for vblank are based on the use of the graphics controller. If a user is using the graphics controller of the OSD, then the vblank must be held for 1 line. If the user is not using the graphics controller, then the vblank can be as short as 1 clock cycle. This one cycle can be pulsed before an hblank pulse.
vsync_in	I	1	INPUT VERTICAL SYNCHRONIZATION SIGNAL This is the vertical synchronization pulse. It is not used internally to the OSD. It is delayed and presented on the `vsync_out` output aligned to the output data.

Table 2: **Port Descriptions *(Cont'd)***

Port Name	Dir	Width	Description
hblank_in	I	1	INPUT HORIZONTAL BLANKING PERIOD SIGNAL Denotes the cycles during which no active video pixel is present. Can be active high or active low polarity. Used internally for synchronization. The known polarity of this input must be set in the OSD Control register before enabling the OSD. The minimum requirement for hblank is 1 clock cycle.
hsync_in	I	1	INPUT HORIZONTAL SYNCHRONIZATION SIGNAL This is the horizontal synchronization pulse. It is not used internally to the OSD. It is delayed and presented on the `hsync_out` output.
active_video_in	I	1	INPUT ACTIVE VIDEO Denotes cycles during which active video is present. Can be active high or active low. It is delayed and presented on the `active_video_out` output.
active_chroma_in	I	1	INPUT ACTIVE CHROMA Denotes cycles during which valid chroma is present on the `video_data_in` bus. It is not used internally to the OSD. It is delayed and presented on the `active_chroma_out` output.
video_data_in	I	[(C_NUM_DATA_CHANNELS+ C_ALPHA_CHANNEL_EN)*C_ DATA_WIDTH-1:0]	INPUT VIDEO DATA Streaming input data. This data can be configured to be routed to one or more layers, which allows enabling graphics layers to be blended to streaming data without the use of external memory.
VFBC Read Data Interface			
vfbc_rd_data	I	[C_NUM_LAYERS*n-1 : 0](3)	VFBC READ FIFO DATA Input layer data for all non Graphics Controller layers. Data is read the following clock cycle after the `vfbc_rd_read` signal is high. *m* is `C_DATA_WIDTH` for the following bit definitions. *Data format for Layer 0 (2 Channels):* Bits (n-1)–3*m: RESERVED(5) Bits (3*m-1)–2*m: Alpha Channel Bits (2*m-1)–m: Data Channel 1 Bits (m-1)–0: Data Channel 0 *Data format for Layer 0 (3 Channels):* Bits (n-1)–4*m: RESERVED(5) Bits (4*m-1)–3*m: Alpha Channel Bits (3*m-1)–2*m: Data Channel 2 Bits (2*m-1)–m: Data Channel 1 Bits (m-1)–0: Data Channel 0 Data format for Layers 1–7 is the same for Layer 0 and repeated for bits (`C_NUM_LAYERS`*n-1)–n.
vfbc_rd_empty	I	[C_NUM_LAYERS-1 : 0]	VFBC READ FIFO EMPTY FLAGS Active high when the input Read FIFO is empty. Bits 7–0 correspond to flags for layers 7–0 respectively.
vfbc_rd_read	O	[C_NUM_LAYERS-1 : 0]	VFBC READ FIFO READ ENABLE The `vfbc_rd_data` is captured one cycle after this port is active high and the `vfbc_rd_empty` is low. Bits 7–0 correspond to read enables for layers 7–0 respectively.

Table 2: **Port Descriptions *(Cont'd)***

Port Name	Dir	Width	Description
vfbc_rd_clk	O	[C_NUM_LAYERS-1 : 0]	VFBC READ FIFO CLOCK Bits 7–0 correspond to the clock for layers 7–0, respectively. Added for ease of use with EDK to allow for quick connect of clock ports. This port is only present on the EDK pCore Interface.
vfbc_rd_reset	O	[C_NUM_LAYERS-1 : 0]	VFBC READ FIFO RESET Bits 7–0 correspond to the reset for layers 7–0, respectively. Used to reset the VFBC FIFO when the OSD core is reset. This port is only present on the EDK pCore Interface.
VFBC Write Data Interface			
vfbc_wd_data	O	[n-1 : 0](4)	VFBC WRITE FIFO DATA Output screen data. Data is asserted the same clock cycle as the `vfbc_wd_write` signal is high. Data format is the same as the `vfbc_rd_data` format for layer 0 for all data channels. The `vfbc_wd_data` has no alpha channel output.
vfbc_wd_afull	I	1	VFBC WRITE FIFO ALMOST FULL FLAG Active high when write FIFO is almost full. Bits 7–0 correspond to flags for layers 7–0 respectively.
vfbc_wd_write	O	1	VFBC WRITE FIFO WRITE ENABLE Active high the same cycle `vfbc_wd_data` is asserted.
vfbc_wd_clk	O	1	VFBC WRITE FIFO CLOCK The clock used by the VFBC Write FIFO. Added for ease of use with EDK to allow for quick connect of clock ports. This port is only present on the EDK pCore Interface.
vfbc_wd_reset	O	1	VFBC WRITE FIFO RESET Used to reset the VFBC FIFO when the OSD core is reset. This port is only present on the EDK pCore Interface.
Output Xilinx Streaming Video Interface			
video_clk_out	O	1	OUTPUT VIDEO CLOCK This port is only present on the EDK pCore Interface.
vblank_out	O	1	OUTPUT VERTICAL BLANKING PERIOD SIGNAL Delayed `vblank_in`.
vsync_out	O	1	OUTPUT VERTICAL SYNCHRONIZATION SIGNAL Delayed `vsync_in`.
hblank_out	O	1	OUTPUT HORIZONTAL SYNCHRONIZATION SIGNAL Delayed `hblank_in`.
hsync_out	O	1	OUTPUT HORIZONTAL BLANKING PERIOD SIGNAL Delayed `hsync_in`.
active_video_out	O	1	OUTPUT ACTIVE VIDEO Delayed `active_video_in`.
active_chroma_out	O	1	OUTPUT ACTIVE CHROMA Delayed `active_chroma_in`.
video_data_out	O	[C_NUM_DATA_CHANNELS*C_DATA_WIDTH-1:0]	OUTPUT VIDEO DATA The resultant output data of the OSD after alpha blending all layers. Data format is the same as the `vfbc_rd_data` format for layer 0 for all data channels. The `video_data_out` port has no alpha channel.

Table 2: **Port Descriptions *(Cont'd)***

Port Name	Dir	Width	Description
General Purpose Processor Interface			
control	I	[31:0]	OSD CONTROL REGISTER Bits 31–6: RESERVED(5) Bit 5: Input vertical blank polarity Bit 4: Input horizontal blank polarity Bit 3: RESERVED(5) Bit 2: OSD Register Update Enable Bit 1: RESERVED(5) Bit 0: OSD Enable
bgcolor0	I	[C_DATA_WIDTH-1:0]	Background Color Channel 0 Background color component for channel 0. If the number of channels is set to two, then this must be the luminance (Y) value.
bgcolor1	I	[C_DATA_WIDTH-1:0]	Background Color Channel 1 Background color component for channel 1. If the number of channels is set to two, then this must be the chrominance (Cb) value.
bgcolor2		[C_DATA_WIDTH-1:0]	Background Color Channel 2 Background color component for channel 2. If the number of channels is set to two, then this must be the chrominance (Cr) value.
layer_enable	I	[C_NUM_LAYERS-1 : 0]	LAYER ENABLES Each bit controls one layer. Bits 7–0 correspond to layers 7–0 respectively. 0: Layer does not show on screen 1: Layer is enabled and shows on screen
layer_priority	I	[C_NUM_LAYERS*32-1 : 0]	LAYER PRIORITIES Controls the Z-plane order of each layer. Bits 31–3: RESERVED(5) Bits 2–0: Layer 0 priority Bits 255–32 are repeated for layers 1–7. A layer with a lower priority than another layer will be displayed beneath. 0 is the lowest priority; 7 is the highest priority.
layer_alpha	I	[C_NUM_LAYERS*32-1 : 0]	LAYER ALPHA CONTROL Global Alpha values and Global Alpha Enables for each layer. Bit 31: Layer 0 Global Alpha enable When high, the pixel alpha from the `vfbc_rd_data` is ignored and the global alpha value from bits 7–0 is used for every pixel in this layer. Bits 30–C_DATA_WIDTH: RESERVED(5) Bits (C_DATA_WIDTH-1)–0: Global Alpha value Bits 255–32 are repeated for layers 1–7. ***Note:*** Useful for YUV or RGB formats that do not have a corresponding pixel alpha.

Table 2: **Port Descriptions *(Cont'd)***

Port Name	Dir	Width	Description
layer_x_pos	I	[C_NUM_LAYERS*32-1 : 0]	LAYER STARTING X POSITION Horizontal start pixel of origin of each layer. Origin of screen is located at (0,0). Bits 31–12: RESERVED(5) Bits 11–0: Layer 0 horizontal start pixel This is the pixel of the layer origin (upper-left corner). Bits 255–32 are repeated for layers 1–7 If the OSD is configured for 2-channels, then the layer starting x position should be set to an even pixel boundary to avoid artifacts.
layer_y_pos	I	[C_NUM_LAYERS*32-1 : 0]	LAYER STARTING Y POSITION Vertical start line of origin of each layer. Origin of screen is located at (0,0). Bits 31–12: RESERVED(5) Bits 11–0: Layer 0 vertical start line This is the line of the layer origin (upper-left corner). Bits 255–32 are repeated for layers 1–7.
layer_x_size	I	[C_NUM_LAYERS*32-1 : 0]	LAYER X SIZE Horizontal Size of Each Layer. Bits 31–12: RESERVED(5) Bits 11–0: Layer 0 horizontal width in number of pixels Bits 255–32 are repeated for layers 1–7. If the OSD is configured for 2-channels, then the layer x size should be set to an even pixel value to avoid artifacts.
layer_y_size	I	[C_NUM_LAYERS*32-1 : 0]	LAYER Y SIZE Vertical Size of Each Layer. Bits 31–12: RESERVED(5) Bits 11–0: Layer 0 vertical height in number of lines Bits 255–32 are repeated for layers 1–7.
screen_x	I	[11:0]	Output Screen X Size Horizontal Width of OSD Output.
screen_y	I	[11:0]	Output Screen Y Size Vertical Height of OSD Output.
intr_status	O	[31:0]	INTERRUPT STATUS Active high edge interrupt status register.
version	O	[31:0]	CORE VERSION Bits 31-28: Major version as a single 4-bit hexadecimal value. Bits 27-20: Minor version as two separate 4-bit hexadecimal values (00 – FF). Bits 19:16: Revision letter as a hexadecimal character from 'a' – 'f'; mapping is as follows: 0XA-'>a', 0xB->'b', 0xC->'c', 0xD->'d', etc. Bits 15-12: Core Generator Patch Revision. Bits 11-0: Reserved.

Table 2: **Port Descriptions *(Cont'd)***

Port Name	Dir	Width	Description
Graphics Controller Interface			
gc_active_bank_addr	I	[31:0]	Active Bank Address Sets the Active Memory Bank for each RAM in each Graphics Controller (GC). For all bits: '0' selects RAM 0, '1' selects RAM 1. Bits 31-24: Active Font RAM for GC 7-0 Bits 23-16: Active Text RAM for GC 7-0 Bits 15-8: Active Color Bank for GC 7-0 Bits 7-0: Active Instruction Bank for GC 7-0
gc_write_bank_addr	I	[5:0]	Write Bank Address Controls which memory bank to write data. Bits 5–3: The Graphics Controller Layer Number Selects which Graphics Controller to write to. Bits 2–0: 000: Write data into Instruction RAM 0 001: Write data into Instruction RAM 1 010: Write data into Color RAM 0 011: Write data into Color RAM 1 100: Write data into Text RAM 0 101: Write data into Text RAM 1 110: Write data into Font RAM 0 111: Write data into Font RAM 1
gc_write_bank_we	I	1	Write Bank Address Write Enable
gc_data	I	[31:0]	GRAPHICS CONTROLLER DATA The data from the host to be written to internal Graphics Controller memory.
gc_data_we	I	1	GRAPHICS CONTROLLER write enable Data is written to the currently selected memory bank when the `gc_data_we` is active. The memory address is automatically incremented after each write. Active high.
AXI Global System Signals(1)			
S_AXI_ACLK	I	1	AXI Clock
S_AXI_ARESETN	I	1	AXI Reset, active low
IP2INTC_Irpt	O	1	Interrupt request output
AXI Write Address Channel Signals(1)			
S_AXI_AWADDR	I	[(C_S_AXI_ADDR_WIDTH-1):0]	AXI4-Lite Write Address Bus. The write address bus gives the address of the write transaction.
S_AXI_AWVALID	I	1	AXI4-Lite Write Address Channel Write Address Valid. This signal indicates that valid write address is available. 1 = Write address is valid. 0 = Write address is not valid.
S_AXI_AWREADY	O	1	AXI4-Lite Write Address Channel Write Address Ready. Indicates core is ready to accept the write address. 1 = Ready to accept address. 0 = Not ready to accept address.

Table 2: **Port Descriptions *(Cont'd)***

Port Name	Dir	Width	Description
AXI Write Data Channel Signals(1)			
S_AXI_WDATA	I	[(C_S_AXI_DATA_WIDTH-1):0]	AXI4-Lite Write Data Bus
S_AXI_WSTRB	I	[C_S_AXI_DATA_WIDTH/8-1:0]	AXI4-Lite Write Strobes. This signal indicates which byte lanes to update in memory.
S_AXI_WVALID	I	1	AXI4-Lite Write Data Channel Write Data Valid. This signal indicates that valid write data and strobes are available. 1 = Write data/strobes are valid. 0 = Write data/strobes are not valid.
S_AXI_WREADY	O	1	AXI4-Lite Write Data Channel Write Data Ready. Indicates core is ready to accept the write data. 1 = Ready to accept data. 0 = Not ready to accept data.
AXI Write Response Channel Signals(1)			
S_AXI_BRESP(2)	O	[1:0]	AXI4-Lite Write Response Channel. Indicates results of the write transfer. 00b = OKAY - Normal access has been successful. 01b = EXOKAY - Not supported. 10b = SLVERR - Error. 11b = DECERR - Not supported.
S_AXI_BVALID	O	1	AXI4-Lite Write Response Channel Response Valid. Indicates response is valid. 1 = Response is valid. 0 = Response is not valid.
S_AXI_BREADY	I	1	AXI4-Lite Write Response Channel Ready. Indicates Master is ready to receive response. 1 = Ready to receive response. 0 = Not ready to receive response.
AXI Read Address Channel Signals(1)			
S_AXI_ARADDR	I	[(C_S_AXI_ADDR_WIDTH-1):0]	AXI4-Lite Read Address Bus. The read address bus gives the address of a read transaction.
S_AXI_ARVALID	I	1	AXI4-Lite Read Address Channel Read Address Valid. 1 = Read address is valid. 0 = Read address is not valid.
S_AXI_ARREADY	O	1	AXI4-Lite Read Address Channel Read Address Ready. Indicates core is ready to accept the read address. 1 = Ready to accept address. 0 = Not ready to accept address.
AXI Read Data Channel Signals(1)			
S_AXI_RDATA	O	[(C_S_AXI_DATA_WIDTH-1):0]	AXI4-Lite Read Data Bus
S_AXI_RRESP(2)	O	[1:0]	AXI4-Lite Read Response Channel Response. Indicates results of the read transfer. 00b = OKAY - Normal access has been successful. 01b = EXOKAY - Not supported. 10b = SLVERR - Error. 11b = DECERR - Not supported.

Table 2: **Port Descriptions *(Cont'd)***

Port Name	Dir	Width	Description
S_AXI_RVALID	O	1	AXI4-Lite Read Data Channel Read Data Valid. This signal indicates that the required read data is available and the read transfer can complete. 1 = Read data is valid. 0 = Read data is not valid.
S_AXI_RREADY	I	1	AXI4-Lite Read Data Channel Read Data Ready. Indicates master is ready to accept the read data. 1 = Ready to accept data. 0 = Not ready to accept data.

1. The function and timing of these signals are defined in the AMBA AXI Protocol Version: 2.0 Specification.
2. For signals S_AXI_RRESP[1:0] and S_AXI_BRESP[1:0], the core does not generate the Decode Error ('11') response. Other responses like '00' (OKAY) and '10' (SLVERR) are generated by the core based upon certain conditions.
3. The data width, *C_NUM_LAYERS*n* of the vfbc_rd_data bus is calculated as the next power of 2 greater than the data channel width multiplied by the number of data channels including the alpha channel, or (C_NUM_DATA_CHANNELS+C_ALPHA _CHANNEL_EN)*C_DATA_WIDTH.
4. The data width, *n*, of the vfbc_wd_data bus is calculated as the next power of 2 greater than the data channel width multiplied by the number of data channels *excluding* the alpha channel, or C_NUM_DATA_CHANNELS*C_DATA_WIDTH.
5. All reserved input pins must be driven by '0'.

See the AMBA AXI4 Interface Protocol web site (http://www.xilinx.com/ipcenter/axi4.htm) for more information on the AXI4 and AXI4-Lite ports and operation.

I/O Interface and Timing

This section describes the signals and timing of the different interfaces of the Xilinx Video On-Screen Display.

Read FIFO Interface

The Xilinx Video On-Screen Display can be configured to have up to eight Read FIFO Interfaces. These are simple FIFO interfaces with a read enable output and a read latency of one clock cycle, typically used to connect to a VFBC interface within a video system.

Figure 7 shows an example read FIFO transaction for when the output interface is set to VFBC.

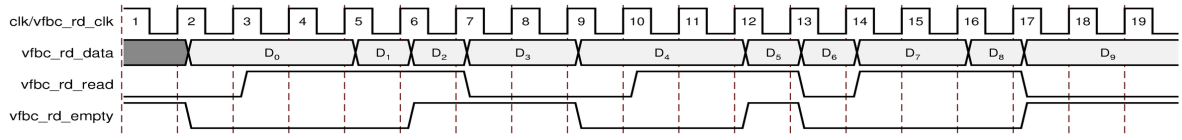

Figure 7: **Read FIFO Interface Timing**

The `vfbc_rd_read` signal is the read enable and is driven out by the OSD. The following clock cycle after the read enable is asserted high, the data on the `vfbc_rd_data` bus is read and used by the OSD (when the data becomes valid by the VFBC).

The `vfbc_rd_empty` (empty flag) input is used only if the output interface of the OSD is configured for a VFBC (Write FIFO) interface. The `vfbc_rd_read` output will not be gated if the OSD output interface is set to XSVI interface.

Write FIFO Interface

The output interface of the Xilinx Video On-Screen Display can be configured to be a VFBC (Write FIFO) interface. In this mode, the OSD has a simple FIFO interface with a write enable output. Valid data is presented on the `vfbc_wd_data` output bus the same clock cycle the `vfbc_wd_write` output is asserted high. See the Migrating to the EDK pCore AXI4-Lite Interface section for more information on the data format of the `vfbc_wd_data` bus.

Figure 8 shows an example write FIFO transaction.

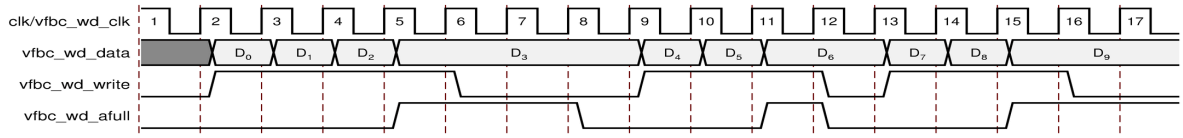

Figure 8: **Write FIFO Interface Timing**

The `vfbc_wd_afull` (FIFO almost full flag) input will be monitored and output data will not advance when it is asserted high.

Xilinx Streaming Video Interface (XSVI)

The output interface of the Xilinx Video On-Screen Display can be configured to be a Xilinx Streaming Video Interface (XSVI). In this mode, valid data is presented on the `video_data_out` output bus in raster order. The `video_data_out` bus is aligned to six video timing signal outputs. These video timing signals outputs are `vblank_out` (for denoting the vertical blanking period), `vsync_out` (for denoting the vertical sync), `hblank_out` (for denoting the horizontal blanking period), `hsync_out` (for denoting the horizontal sync), `active_video_out` (for denoting those clock cycles that contain valid data on the `video_data_out` bus) and active_chroma_out (for denoting those video lines that contain valid chroma).

Figure 9 shows an example XSVI output waveform. All video timing signals are shown as active high polarity. The output video frame in this example is configured for four active video lines, three lines of vertical blanking and one line of vertical sync.

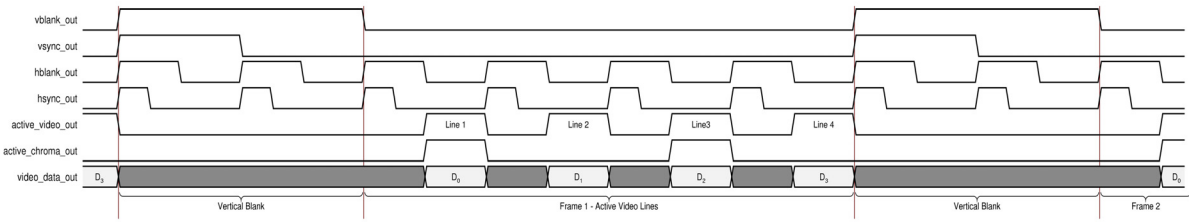

Figure 9: **XSVI Interface Timing**

The video timing signal outputs are not generated internally by the OSD. Instead there are six similar video timing signal inputs. These inputs are delayed and present on the XSVI output interface. Valid data on the `video_data_out` output bus is aligned to the video timing signal outputs.

Each video timing signal input may be active high or active low. Only the vertical and horizontal blank signals are used by the OSD. The vertical and horizontal sync signals and the active video signal are optional and required only if the hardware connected to the XSVI output interface of the OSD expects these signals.

The polarity of the horizontal and vertical blank input signals must be known and programmed in the OSD Control Register.

Graphics Controller Host Interface

Data is written into the internal memory of each graphics controller via a single host interface. The Xilinx Video On-Screen Display provides five input ports for controlling and loading this data. The `gc_write_bank_addr` port selects which internal memory bank (RAM) of which graphics controller to be written. The `gc_write_bank_addr` is captured the first clock cycle that the `gc_write_bank_we` (bank write enable) port is asserted high. The `gc_data` port should be driven with the data to be written to internal memory during the first clock cycle the `gc_data_we` (data write enable) port is asserted high. The `gc_active_bank_addr` selects the active memory bank for instruction, font, text and color memories for each graphics controller.

Figure 10 shows an example graphics controller memory load operation for two memories.

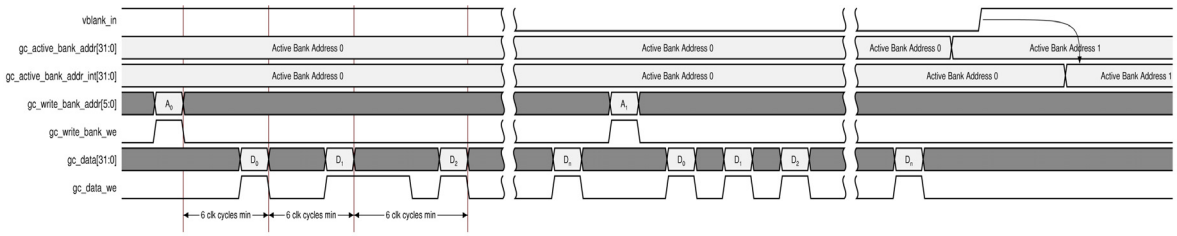

Figure 10: **Graphics Controller Host Interface Timing**

In order to write into the internal memory of a graphics controller, the host processor must:

1. Write the bank address to the `gc_write_bank_addr` register. This selects the target memory of the target graphics controller.
2. Write all data in single cycle writes until all data is written. The graphics controller will automatically increment its internal address pointer after each write. The graphics controller will also set its corresponding address overflow interrupt status bit if the host tries to write beyond the depth of the currently selected memory.

For the new data to be used by the graphics controller, the host processor must set the memory active in the `gc_active_bank_addr` register after all data is written. The graphics controller will use the new setting after the start of the next vertical blanking interval period defined by the `vblank_in` port. The host processor should avoid writing to memory that is currently set active in the `gc_active_bank_addr` register.

There must be at least six clock cycles between each write to the `gc_write_bank_addr` or `gc_data` registers. The `gc_data_we` input can be asserted high for more than one clock cycle. The `gc_data` bus is read only during the first clock cycle that the `gc_data_we` is high. The `gc_write_bank_we` and `gc_write_bank_addr` behave similarly. The `gc_write_bank_addr` is only read during the first cycle that the `gc_write_bank_we` is asserted high.

The OSD contains and keeps track of multiple address pointers internally and increments after each gc_data_we. The internal address pointer(s) are reset when the gc_write_bank_we is asserted high. There are three internal address pointers, one for the instruction RAM, one for the color RAM and one for font and text RAMs. Depending on the gc_write_bank_addr value(00=instruction, 01=color, 10=text, 11=font), one of the three pointers is reset.

Interrupts

The Xilinx Video On-Screen Display provides a 32-bit output bus, intr_status[31:0], for host processor interrupt status when configured for the General Purpose Processor (GPP) interface. All interrupt status bits can trigger an interrupt on the active high edge. Status bits are set high when the internal event occurs and are cleared ether at the start or at the end of the vertical blanking interval period defined by the vblank_in port.

Interrupt status bits 31-3 are cleared at the start of the vertical blanking interval period. These bits include the graphics controller address overflow, the graphics controller instruction error, the output FIFO overflow error, the input FIFOs underflow error and the vertical blanking interval end interrupt status bits.

Interrupt status bits 2-0 are cleared at the end of the vertical blanking interval period. These bits include the vertical blanking interval period start, frame error and frame done interrupt status bits.

The interrupt status output bus can easily be integrated with an external interrupt controller that has independent interrupt enable/mask, interrupt clear and interrupt status registers and that allows for interrupt aggregation to the system processor. An example system showing the OSD and other processor peripherals connected to an interrupt controller is depicted in Figure 11.

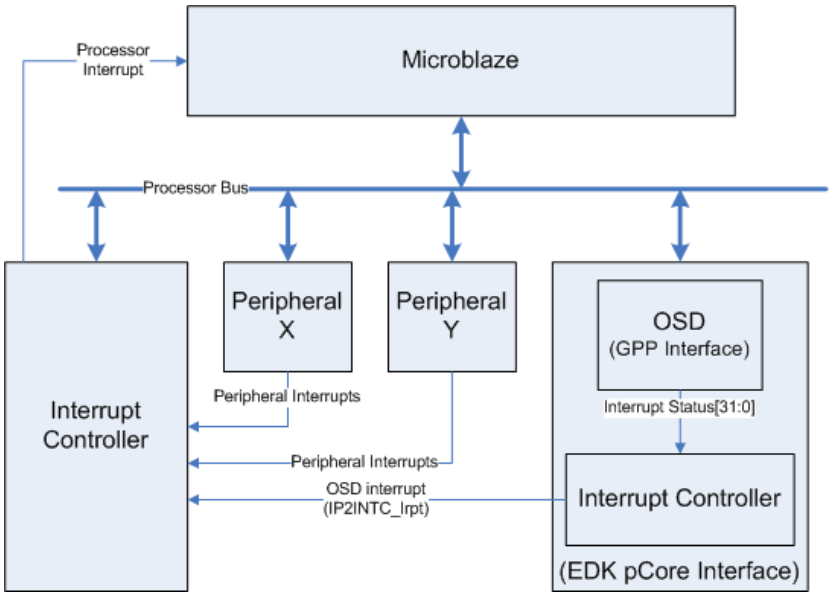

Figure 11: **Interrupt Controller Processor Peripherals**

The Xilinx Video On-Screen Display, when configured for the EDK pCore Interface, automatically contains an internal interrupt controller for enabling/masking and clearing each interrupt. The 1-bit output port, `IP2INTC_Irpt`, is the interrupt output in this mode. The OSD internal interrupt controller is the Xilinx Interrupt Control LogiCORE system.

General Purpose Processor Interface

The General Purpose Processor Interface is a dynamic register interface and exposes all control and status registers as ports. These ports can easily be connected to any processor with a user-defined bus interface via a Register File and minimal logic. The interrupt status register is included in this interface.

All input ports are double-buffered and captured internally at the start of the vertical blanking interval period defined by the `vblank_in` port. In addition, the register update enable (bit 2 of the OSD Control Register) can disable updates to the internally buffered registers. This allows the host processor to update OSD control registers during multiple video frames. The register update enable bit is not double-buffered.

The General Purpose Processor ports for this core are defined in Table 2 in the Migrating to the EDK pCore AXI4-Lite Interface section.

EDK pCore AXI4-Lite Interface

The Xilinx Video On-Screen Display, when configured as an EDK pCore, uses the AXI4-Lite Interface to interface to a microprocessor. Refer to the AMBA AXI4 Interface Protocol web site (http://www.xilinx.com/ipcenter/axi4.htm) for more information on the AXI4 and AXI4-Lite interface signals.

When the developer selects the EDK pCore interface, Xilinx CORE Generator creates a pCore and all support files that can be added to an EDK project as a hardware peripheral. This pCore provides a memory mapped interface for the programmable registers within the core and a complete device driver to enable rapid application development.

Xilinx CORE Generator will place all EDK pCore source files in the "pcores" subdirectory located in the core output directory. The core output directory is given the same name as the component. For example, if the component name is set to "v_osd_v2_0_u0," then the EDK pCore source files will be located in the following directory:

```
<coregen project directory>/v_osd_v2_0_u0/pcores/axi_osd_v2_00_a
```

The pCore should be copied to the user's <EDK_Project>/pcores directory or to a user pCores repository.

Parameter Modification in CORE Generator

EDK pCore parameters found in the <coregen project directory>/v_osd_v2_0_u0/pcores/axi_osd_v2_00_a/data/axi_osd_v2_1_0.mpd file *cannot* be modified in the Xilinx CORE Generator tool. Parameters shown on the CORE Generator Graphical User Interface will be disabled if the EDK pCore (AXI4-Lite) Interface is selected. Xilinx recommends that all parameter changes be made with the Video On-Screen Display pCore GUI in the EDK environment.

pCore Device Driver

The Xilinx On-Screen Display pCore includes a software driver written in the C Language that the user can use to control the Xilinx OSD devices. A high-level API is provided and can be used without detailed knowledge of the Xilinx OSD devices. Application developers are encouraged to use this API to access the device features. A low-level API is also provided in case applications prefer to access the devices directly through the system registers described in the previous section.

Table 3 lists the files that are included with the Xilinx OSD pCore driver and their description.

Table 3: **Device Driver Source Files**

File Name	Description
xosd.h	Contains all prototypes of high-level API to access all of the features of the Xilinx OSD devices.
xosd.c	Contains the implementation of high-level API to access all of the features of the Xilinx OSD devices except interrupts.
xosd_intr.c	Contains the implementation of high-level API to access interrupt feature of the Xilinx OSD devices.
xosd_sinit.c	Contains static initialization methods for the Xilinx OSD device driver.
xosd_g.c	Contains a template for configuration table of Xilinx OSD devices. This file is used by the high-level API and will be automatically generated to match the OSD device configurations by Xilinx EDK/SDK tools when the software project is built.
xosd_hw.h	Contains Low-level API (that is, register offset/bit definition and register-level driver API) that can be used to access the Xilinx OSD devices.
example.c	An example that demonstrates how to control the Xilinx OSD devices using the high-level API.

Xilinx CORE Generator software will place all EDK pCore driver files in the "drivers" subdirectory located in the core output directory. The core output directory is given the same name as the component. For example, if the component name is set to "v_osd_v2_0_u0," then the device driver source files will be located in the following directory:

```
<coregen project directory>/v_osd_v2_0_u0/drivers/osd_v1_02_a/
```

The driver software should be copied to the user's <EDK_Project>/drivers directory or to a user pCores repository.

Resource Estimates

Resources required for Spartan®-3A DSP devices are estimated in Table 4, Table 5, Table 6 and Table 7 and show the same estimates for Spartan®-6, Virtex®-5, and Virtex®-6 devices. Resource usage values were generated using the Xilinx CORE Generator in ISE® 13.1 tools. They are derived from post-MAP reports, but may change due to optimization settings or post-PAR optimization. All numbers are generated using the General Purpose Processor Interface of the Xilinx Video On-Screen Display.

All resource estimate configurations containing Graphics Controller layers have the Graphics Controller parameters set to the following:

- Instructions = 48
- Number of Colors = 16
- Number of Characters = 96
- Character Width = 8
- Character Height = 8
- ASCII Offset = 32
- Character Bits per Pixel = 1
- Number of Strings = 8
- Maximum String Length = 32

Different Graphics Controller parameter settings affect block RAM utilization. The following equation yields the upper bound of the block RAM utilization for Virtex-5 and Virtex-6 devices. The actual utilization may be lower due to block RAM data packing.

Number of Block RAMs <=

*(Maximum Screen Width) * LOG2(Number of Colors) /8192*

+ Instructions / 128

*+ (Number of Characters) * (Character Width) * (Character Height) * (Character Bits per Pixel) / 8192*

*+ (Number of Strings) * (Maximum String Length) / 1024*

The following equation yields the upper bound of the block RAM utilization for Spartan-3A DSP and Spartan-6 devices. The actual utilization may be lower due to block RAM data packing.

Number of Block RAMs <=

*(Maximum Screen Width) * LOG-2(Number of Colors) /4096*

+ Instructions / 128

*+ (Number of Characters) * (Character Width) * (Character Height) * (Character Bits per Pixel) / 8192*

*+ (Number of Strings) * (Maximum String Length) / 1024*

The EDK pCore interface adds an estimated additional 1130 FFs and 550 LUTs for configurations registers and the AXI4-Lite interface.

Table 4: **Spartan-3A DSP Resource Estimates**

Layer Type	Data Width	Channels	Layers	Maximum Screen Width(2)	XtremeDSP Slices	BRAM(1),(3)	LUTs	FFs
Graphics Controller	8	2	1	320	2	2	1,739	1,326
Graphics Controller	8	2	1	4096	2	5	1,783	1,338
Graphics Controller	8	2	2	320	4	4	3,394	2,427
Graphics Controller	8	2	2	4096	4	10	3,488	2,471
Graphics Controller	8	2	3	320	6	6	5,234	3,714
Graphics Controller	8	2	3	4096	6	15	5,375	3,780
Graphics Controller	8	2	4	320	8	8	6,992	4,960
Graphics Controller	8	2	4	4096	8	20	7,156	5,008
Graphics Controller	8	2	5	320	10	10	9,218	6,605
Graphics Controller	8	2	5	4096	10	25	9,423	6,655
Graphics Controller	8	2	6	320	12	12	11,090	7,953
Graphics Controller	8	2	6	4096	12	30	11,336	8,013
Graphics Controller	8	2	7	320	14	14	13,177	9,442
Graphics Controller	8	2	7	4096	14	35	13,480	9,526
Graphics Controller	8	2	8	320	16	16	15,133	10,912
Graphics Controller	8	2	8	4096	16	40	15,461	11,008
Graphics Controller	8	3	1	320	3	2	1,784	1,433
Graphics Controller	8	3	1	4096	3	5	1,823	1,445
Graphics Controller	8	3	2	320	6	4	3,514	2,634
Graphics Controller	8	3	2	4096	6	10	3,592	2,658
Graphics Controller	8	3	3	320	9	6	5,464	4,069
Graphics Controller	8	3	3	4096	9	15	5,581	4,105

Table 4: **Spartan-3A DSP Resource Estimates**

Graphics Controller	8	3	4	320	12	8	7,282	5,387
Graphics Controller	8	3	4	4096	12	20	7,438	5,435
Graphics Controller	8	3	5	320	15	10	9,733	7,246
Graphics Controller	8	3	5	4096	15	25	9,928	7,301
Graphics Controller	8	3	6	320	18	12	11,694	8,675
Graphics Controller	8	3	6	4096	18	30	11,940	8,747
Graphics Controller	8	3	7	320	21	14	13,979	10,436
Graphics Controller	8	3	7	4096	21	35	14,259	10,520
Graphics Controller	8	3	8	320	24	16	16,044	12,006
Graphics Controller	8	3	8	4096	24	40	16,367	12,086
Graphics Controller	10	2	1	320	2	2	1,803	1,434
Graphics Controller	10	2	1	4096	2	5	1,847	1,446
Graphics Controller	10	2	2	320	4	4	3,537	2,630
Graphics Controller	10	2	2	4096	4	10	3,631	2,674
Graphics Controller	10	2	3	320	6	6	5,488	4,053
Graphics Controller	10	2	3	4096	6	15	5,629	4,119
Graphics Controller	10	2	4	320	8	8	7,339	5,414
Graphics Controller	10	2	4	4096	8	20	7,502	5,462
Graphics Controller	10	2	5	320	10	10	9,768	7,269
Graphics Controller	10	2	5	4096	10	25	9,977	7,319
Graphics Controller	10	2	6	320	12	12	11,761	8,758
Graphics Controller	10	2	6	4096	12	30	12,012	8,818
Graphics Controller	10	2	7	320	14	14	14,015	10,435
Graphics Controller	10	2	7	4096	14	35	14,317	10,519

Table 4: **Spartan-3A DSP Resource Estimates**

Graphics Controller	10	2	8	320	16	16	16,099	12,060
Graphics Controller	10	2	8	4096	16	40	16,434	12,156
Graphics Controller	10	3	1	320	3	2	1,849	1,559
Graphics Controller	10	3	1	4096	3	5	1,888	1,571
Graphics Controller	10	3	2	320	6	4	3,662	2,872
Graphics Controller	10	3	2	4096	6	10	3,741	2,896
Graphics Controller	10	3	3	320	9	6	5,748	4,470
Graphics Controller	10	3	3	4096	9	15	5,867	4,506
Graphics Controller	10	3	4	320	12	8	7,667	5,924
Graphics Controller	10	3	4	4096	12	20	7,826	5,972
Graphics Controller	10	3	5	320	15	10	10,355	8,042
Graphics Controller	10	3	5	4096	15	25	10,554	8,097
Graphics Controller	10	3	6	320	18	12	12,457	9,640
Graphics Controller	10	3	6	4096	18	30	12,703	9,712
Graphics Controller	10	3	7	320	21	14	14,949	11,629
Graphics Controller	10	3	7	4096	21	35	15,235	11,713
Graphics Controller	10	3	8	320	24	16	17,168	13,382
Graphics Controller	10	3	8	4096	24	40	17,495	13,462
Graphics Controller	12	2	1	320	2	2	1,864	1,509
Graphics Controller	12	2	1	4096	2	5	1,908	1,521
Graphics Controller	12	2	2	320	4	4	3,673	2,768
Graphics Controller	12	2	2	4096	4	10	3,767	2,812
Graphics Controller	12	2	3	320	6	6	5,736	4,293
Graphics Controller	12	2	3	4096	6	15	5,877	4,359

Table 4: **Spartan-3A DSP Resource Estimates**

Graphics Controller	12	2	4	320	8	8	7,681	5,737
Graphics Controller	12	2	4	4096	8	20	7,844	5,785
Graphics Controller	12	2	5	320	10	10	10,304	7,768
Graphics Controller	12	2	5	4096	10	25	10,509	7,818
Graphics Controller	12	2	6	320	12	12	12,416	9,366
Graphics Controller	12	2	6	4096	12	30	12,662	9,426
Graphics Controller	12	2	7	320	14	14	14,838	11,197
Graphics Controller	12	2	7	4096	14	35	15,139	11,281
Graphics Controller	12	2	8	320	16	16	17,061	12,945
Graphics Controller	12	2	8	4096	16	40	17,389	13,041
Graphics Controller	12	3	1	320	3	2	1,911	1,652
Graphics Controller	12	3	1	4096	3	5	1,950	1,664
Graphics Controller	12	3	2	320	6	4	3,809	3,044
Graphics Controller	12	3	2	4096	6	10	3,887	3,068
Graphics Controller	12	3	3	320	9	6	6,026	4,773
Graphics Controller	12	3	3	4096	9	15	6,143	4,809
Graphics Controller	12	3	4	320	12	8	8,047	6,329
Graphics Controller	12	3	4	4096	12	20	8,203	6,377
Graphics Controller	12	3	5	320	15	10	10,975	8,674
Graphics Controller	12	3	5	4096	15	25	11,170	8,729
Graphics Controller	12	3	6	320	18	12	13,202	10,406
Graphics Controller	12	3	6	4096	18	30	13,448	10,478
Graphics Controller	12	3	7	320	21	14	15,902	12,590
Graphics Controller	12	3	7	4096	21	35	16,182	12,674

Table 4: **Spartan-3A DSP Resource Estimates**

Graphics Controller	12	3	8	320	24	16	18,273	14,493
Graphics Controller	12	3	8	4096	24	40	18,593	14,573
VFBC	8	2	1	-	2	0	318	518
VFBC	8	2	2	-	4	0	524	840
VFBC	8	2	3	-	6	0	904	1,367
VFBC	8	2	4	-	8	0	1,185	1,805
VFBC	8	2	5	-	10	0	1,945	2,702
VFBC	8	2	6	-	12	0	2,351	3,279
VFBC	8	2	7	-	14	0	2,996	4,074
VFBC	8	2	8	-	16	0	3,481	4,734
VFBC	8	3	1	-	3	0	330	608
VFBC	8	3	2	-	6	0	576	1,016
VFBC	8	3	3	-	9	0	1,035	1,677
VFBC	8	3	4	-	12	0	1,363	2,215
VFBC	8	3	5	-	15	0	2,310	3,340
VFBC	8	3	6	-	18	0	2,791	4,043
VFBC	8	3	7	-	21	0	3,561	5,021
VFBC	8	3	8	-	24	0	4,128	5,814
VFBC	10	2	1	-	2	0	336	589
VFBC	10	2	2	-	4	0	574	970
VFBC	10	2	3	-	6	0	1,020	1,601
VFBC	10	2	4	-	8	0	1,347	2,119
VFBC	10	2	5	-	10	0	2,268	3,200
VFBC	10	2	6	-	12	0	2,749	3,887
VFBC	10	2	7	-	14	0	3,509	4,840
VFBC	10	2	8	-	16	0	4,084	5,625
VFBC	10	3	1	-	3	0	351	683
VFBC	10	3	2	-	6	0	636	1,149
VFBC	10	3	3	-	9	0	1,188	1,926
VFBC	10	3	4	-	12	0	1,577	2,547
VFBC	10	3	5	-	15	0	2,715	3,892
VFBC	10	3	6	-	18	0	3,290	4,713
VFBC	10	3	7	-	21	0	4,226	5,877
VFBC	10	3	8	-	24	0	4,905	6,806
VFBC	12	2	1	-	2	0	354	647
VFBC	12	2	2	-	4	0	624	1,075
VFBC	12	2	3	-	6	0	1,139	1,793
VFBC	12	2	4	-	8	0	1,518	2,379

Table 4: **Spartan-3A DSP Resource Estimates**

VFBC	12	2	5	-	10	0	2,585	3,629
VFBC	12	2	6	-	12	0	3,142	4,411
VFBC	12	2	7	-	14	0	4,031	5,510
VFBC	12	2	8	-	16	0	4,695	6,406
VFBC	12	3	1	-	3	0	372	773
VFBC	12	3	2	-	6	0	701	1,316
VFBC	12	3	3	-	9	0	1,341	2,227
VFBC	12	3	4	-	12	0	1,789	2,953
VFBC	12	3	5	-	15	0	3,126	4,536
VFBC	12	3	6	-	18	0	3,789	5,494
VFBC	12	3	7	-	21	0	4,887	6,863
VFBC	12	3	8	-	24	0	5,676	7,946
XSVI	8	2	1	-	2	0	366	614
XSVI	8	2	2	-	4	0	625	974
XSVI	8	2	3	-	6	0	1,106	1,564
XSVI	8	2	4	-	8	0	1,462	2,033
XSVI	8	2	5	-	10	0	2,450	2,986
XSVI	8	2	6	-	12	0	3,066	3,600
XSVI	8	2	7	-	14	0	3,844	4,430
XSVI	8	2	8	-	16	0	4,466	5,099
XSVI	8	3	1	-	3	0	402	741
XSVI	8	3	2	-	6	0	725	1,199
XSVI	8	3	3	-	9	0	1,333	1,953
XSVI	8	3	4	-	12	0	1,769	2,534
XSVI	8	3	5	-	15	0	3,056	3,745
XSVI	8	3	6	-	18	0	3,842	4,504
XSVI	8	3	7	-	21	0	4,803	5,539
XSVI	8	3	8	-	24	0	5,563	6,354
XSVI	10	2	1	-	2	0	396	697
XSVI	10	2	2	-	4	0	699	1,116
XSVI	10	2	3	-	6	0	1,270	1,817
XSVI	10	2	4	-	8	0	1,688	2,366
XSVI	10	2	5	-	10	0	2,894	3,511
XSVI	10	2	6	-	12	0	3,632	4,236
XSVI	10	2	7	-	14	0	4,554	5,229
XSVI	10	2	8	-	16	0	5,295	6,023
XSVI	10	3	1	-	3	0	442	858
XSVI	10	3	2	-	6	0	821	1,399
XSVI	10	3	3	-	9	0	1,559	2,305

Table 4: **Spartan-3A DSP Resource Estimates**

XSVI	10	3	4	-	12	0	2,078	2,994
XSVI	10	3	5	-	15	0	3,642	4,461
XSVI	10	3	6	-	18	0	4,595	5,370
XSVI	10	3	7	-	21	0	5,763	6,620
XSVI	10	3	8	-	24	0	6,676	7,596
XSVI	12	2	1	-	2	0	426	780
XSVI	12	2	2	-	4	0	773	1,258
XSVI	12	2	3	-	6	0	1,437	2,069
XSVI	12	2	4	-	8	0	1,923	2,699
XSVI	12	2	5	-	10	0	3,331	4,034
XSVI	12	2	6	-	12	0	4,194	4,874
XSVI	12	2	7	-	14	0	5,273	6,028
XSVI	12	2	8	-	16	0	6,129	6,947
XSVI	12	3	1	-	3	0	481	972
XSVI	12	3	2	-	6	0	922	1,595
XSVI	12	3	3	-	9	0	1,784	2,651
XSVI	12	3	4	-	12	0	2,386	3,450
XSVI	12	3	5	-	15	0	4,234	5,171
XSVI	12	3	6	-	18	0	5,347	6,228
XSVI	12	3	7	-	21	0	6,717	7,693
XSVI	12	3	8	-	24	0	7,783	8,829

1. Block RAM type for this device family is RAMB16BWER.
2. The "Maximum Screen Width" parameter does not affect the VFBC or XSVI layer resources.
3. All Graphics Controller configurations have default parameter settings.

Table 5: **Spartan-6 Resource Estimates**

Layer Type	Data Width	Channels	Layers	Maximum Screen Width(2)	XtremeDSP Slices	BRAM(1),(3)	LUTs	FFs
Graphics Controller	8	2	1	320	2	2	1,523	1,397
Graphics Controller	8	2	1	4096	2	5	1,486	1,410
Graphics Controller	8	2	2	320	4	4	2,995	2,673
Graphics Controller	8	2	2	4096	4	10	2,907	2,707
Graphics Controller	8	2	3	320	6	6	4,604	4,096
Graphics Controller	8	2	3	4096	6	15	4,481	4,151
Graphics Controller	8	2	4	320	8	8	5,998	5,443
Graphics Controller	8	2	4	4096	8	20	5,958	5,509
Graphics Controller	8	2	5	320	10	10	7,907	7,230
Graphics Controller	8	2	5	4096	10	25	8,060	7,296
Graphics Controller	8	2	6	320	12	12	9,567	8,729
Graphics Controller	8	2	6	4096	12	30	9,711	8,808

Table 5: **Spartan-6 Resource Estimates**

Graphics Controller	8	2	7	320	14	14	11,191	10,426
Graphics Controller	8	2	7	4096	14	35	11,439	10,518
Graphics Controller	8	2	8	320	16	16	12,990	12,009
Graphics Controller	8	2	8	4096	16	40	13,158	12,115
Graphics Controller	8	3	1	320	3	2	1,579	1,500
Graphics Controller	8	3	1	4096	3	5	1,529	1,512
Graphics Controller	8	3	2	320	6	4	2,835	2,872
Graphics Controller	8	3	2	4096	6	10	3,115	2,918
Graphics Controller	8	3	3	320	9	6	4,810	4,461
Graphics Controller	8	3	3	4096	9	15	4,601	4,519
Graphics Controller	8	3	4	320	12	8	6,651	5,908
Graphics Controller	8	3	4	4096	12	20	6,503	5,961
Graphics Controller	8	3	5	320	15	10	8,526	7,970
Graphics Controller	8	3	5	4096	15	25	8,651	8,036
Graphics Controller	8	3	6	320	18	12	10,399	9,626
Graphics Controller	8	3	6	4096	18	30	10,260	9,705
Graphics Controller	8	3	7	320	21	14	12,049	11,522
Graphics Controller	8	3	7	4096	21	35	12,224	11,615
Graphics Controller	8	3	8	320	24	16	14,048	13,659
Graphics Controller	8	3	8	4096	24	40	14,128	13,757
Graphics Controller	10	2	1	320	2	2	1,554	1,500
Graphics Controller	10	2	1	4096	2	5	1,538	1,513
Graphics Controller	10	2	2	320	4	4	3,152	2,869
Graphics Controller	10	2	2	4096	4	10	2,983	2,905
Graphics Controller	10	2	3	320	6	6	4,662	4,432
Graphics Controller	10	2	3	4096	6	15	4,616	4,486
Graphics Controller	10	2	4	320	8	8	6,233	5,899
Graphics Controller	10	2	4	4096	8	20	5,545	5,926
Graphics Controller	10	2	5	320	10	10	8,193	7,906
Graphics Controller	10	2	5	4096	10	25	8,083	7,971
Graphics Controller	10	2	6	320	12	12	9,689	9,557
Graphics Controller	10	2	6	4096	12	30	10,048	9,636
Graphics Controller	10	2	7	320	14	14	11,684	11,448
Graphics Controller	10	2	7	4096	14	35	11,796	11,540
Graphics Controller	10	2	8	320	16	16	13,348	13,196
Graphics Controller	10	2	8	4096	16	40	13,657	13,302
Graphics Controller	10	3	1	320	3	2	1,660	1,622
Graphics Controller	10	3	1	4096	3	5	1,569	1,635
Graphics Controller	10	3	2	320	6	4	3,296	3,137

Table 5: **Spartan-6 Resource Estimates**

Graphics Controller	10	3	2	4096	6	10	3,153	3,159
Graphics Controller	10	3	3	320	9	6	4,509	4,854
Graphics Controller	10	3	3	4096	9	15	4,417	4,893
Graphics Controller	10	3	4	320	12	8	6,791	6,461
Graphics Controller	10	3	4	4096	12	20	6,781	6,513
Graphics Controller	10	3	5	320	15	10	8,951	8,800
Graphics Controller	10	3	5	4096	15	25	8,943	8,866
Graphics Controller	10	3	6	320	18	12	10,651	10,640
Graphics Controller	10	3	6	4096	18	30	10,842	10,719
Graphics Controller	10	3	7	320	21	14	12,668	13,153
Graphics Controller	10	3	7	4096	21	35	12,911	13,245
Graphics Controller	10	3	8	320	24	16	15,223	15,189
Graphics Controller	10	3	8	4096	24	40	15,192	15,294
Graphics Controller	12	2	1	320	2	2	1,601	1,573
Graphics Controller	12	2	1	4096	2	5	1,578	1,585
Graphics Controller	12	2	2	320	4	4	2,836	3,004
Graphics Controller	12	2	2	4096	4	10	3,136	3,050
Graphics Controller	12	2	3	320	6	6	4,292	4,666
Graphics Controller	12	2	3	4096	6	15	4,306	4,705
Graphics Controller	12	2	4	320	8	8	6,486	6,239
Graphics Controller	12	2	4	4096	8	20	6,147	6,273
Graphics Controller	12	2	5	320	10	10	8,342	8,442
Graphics Controller	12	2	5	4096	10	25	8,635	8,508
Graphics Controller	12	2	6	320	12	12	10,111	10,217
Graphics Controller	12	2	6	4096	12	30	10,425	10,296
Graphics Controller	12	2	7	320	14	14	12,139	12,274
Graphics Controller	12	2	7	4096	14	35	12,313	12,366
Graphics Controller	12	2	8	320	16	16	13,831	14,162
Graphics Controller	12	2	8	4096	16	40	14,184	14,268
Graphics Controller	12	3	1	320	3	2	1,660	1,714
Graphics Controller	12	3	1	4096	3	5	1,575	1,727
Graphics Controller	12	3	2	320	6	4	3,452	3,321
Graphics Controller	12	3	2	4096	6	10	2,932	3,323
Graphics Controller	12	3	3	320	9	6	4,674	5,179
Graphics Controller	12	3	3	4096	9	15	4,560	5,217
Graphics Controller	12	3	4	320	12	8	7,033	6,904
Graphics Controller	12	3	4	4096	12	20	7,111	6,954
Graphics Controller	12	3	5	320	15	10	9,212	9,493
Graphics Controller	12	3	5	4096	15	25	9,384	9,558

Table 5: **Spartan-6 Resource Estimates**

Graphics Controller	12	3	6	320	18	12	11,274	11,868
Graphics Controller	12	3	6	4096	18	30	11,427	11,947
Graphics Controller	12	3	7	320	21	14	13,679	14,281
Graphics Controller	12	3	7	4096	21	35	13,531	14,373
Graphics Controller	12	3	8	320	24	16	16,007	16,506
Graphics Controller	12	3	8	4096	24	40	16,056	16,611
VFBC	8	2	1	-	2	0	345	543
VFBC	8	2	2	-	4	0	551	885
VFBC	8	2	3	-	6	0	803	1,434
VFBC	8	2	4	-	8	0	1,127	1,913
VFBC	8	2	5	-	10	0	1,570	2,702
VFBC	8	2	6	-	12	0	2,538	3,530
VFBC	8	2	7	-	14	0	3,717	4,418
VFBC	8	2	8	-	16	0	4,202	5,104
VFBC	8	3	1	-	3	0	379	642
VFBC	8	3	2	-	6	0	724	1,074
VFBC	8	3	3	-	9	0	962	1,770
VFBC	8	3	4	-	12	0	1,225	2,359
VFBC	8	3	5	-	15	0	1,924	3,338
VFBC	8	3	6	-	18	0	2,433	4,319
VFBC	8	3	7	-	21	0	4,176	5,400
VFBC	8	3	8	-	24	0	4,954	6,276
VFBC	10	2	1	-	2	0	400	621
VFBC	10	2	2	-	4	0	620	1,028
VFBC	10	2	3	-	6	0	916	1,686
VFBC	10	2	4	-	8	0	1,315	2,254
VFBC	10	2	5	-	10	0	1,847	3,199
VFBC	10	2	6	-	12	0	2,486	4,389
VFBC	10	2	7	-	14	0	2,864	5,140
VFBC	10	2	8	-	16	0	4,846	6,070
VFBC	10	3	1	-	3	0	408	721
VFBC	10	3	2	-	6	0	690	1,219
VFBC	10	3	3	-	9	0	1,205	2,032
VFBC	10	3	4	-	12	0	1,504	2,718
VFBC	10	3	5	-	15	0	2,065	4,129
VFBC	10	3	6	-	18	0	4,590	5,084
VFBC	10	3	7	-	21	0	4,871	6,280
VFBC	10	3	8	-	24	0	6,365	7,345
VFBC	12	2	1	-	2	0	433	685

Table 5: **Spartan-6 Resource Estimates**

VFBC	12	2	2	-	4	0	633	1,140
VFBC	12	2	3	-	6	0	1,055	1,891
VFBC	12	2	4	-	8	0	1,386	2,535
VFBC	12	2	5	-	10	0	2,142	3,846
VFBC	12	2	6	-	12	0	2,696	4,715
VFBC	12	2	7	-	14	0	3,279	5,857
VFBC	12	2	8	-	16	0	6,371	6,958
VFBC	12	3	1	-	3	0	439	820
VFBC	12	3	2	-	6	0	773	1,400
VFBC	12	3	3	-	9	0	1,305	2,356
VFBC	12	3	4	-	12	0	1,781	3,341
VFBC	12	3	5	-	15	0	2,628	4,816
VFBC	12	3	6	-	18	0	5,183	5,964
VFBC	12	3	7	-	21	0	6,231	7,389
VFBC	12	3	8	-	24	0	7,562	8,609
XSVI	8	2	1	-	2	0	405	656
XSVI	8	2	2	-	4	0	628	1,039
XSVI	8	2	3	-	6	0	1,040	1,661
XSVI	8	2	4	-	8	0	1,394	2,172
XSVI	8	2	5	-	10	0	2,501	3,181
XSVI	8	2	6	-	12	0	4,328	4,480
XSVI	8	2	7	-	14	0	4,998	5,161
XSVI	8	2	8	-	16	0	3,328	5,465
XSVI	8	3	1	-	3	0	475	797
XSVI	8	3	2	-	6	0	785	1,290
XSVI	8	3	3	-	9	0	1,315	2,173
XSVI	8	3	4	-	12	0	1,599	2,725
XSVI	8	3	5	-	15	0	2,462	3,747
XSVI	8	3	6	-	18	0	5,064	5,351
XSVI	8	3	7	-	21	0	5,613	6,572
XSVI	8	3	8	-	24	0	6,455	7,447
XSVI	10	2	1	-	2	0	451	747
XSVI	10	2	2	-	4	0	725	1,195
XSVI	10	2	3	-	6	0	1,170	1,935
XSVI	10	2	4	-	8	0	2,502	2,854
XSVI	10	2	5	-	10	0	3,946	4,119
XSVI	10	2	6	-	12	0	4,753	4,997
XSVI	10	2	7	-	14	0	5,266	6,163
XSVI	10	2	8	-	16	0	6,061	6,993

Table 5: **Spartan-6 Resource Estimates**

XSVI	10	3	1	-	3	0	508	922
XSVI	10	3	2	-	6	0	923	1,510
XSVI	10	3	3	-	9	0	2,499	2,823
XSVI	10	3	4	-	12	0	3,275	3,902
XSVI	10	3	5	-	15	0	3,017	4,775
XSVI	10	3	6	-	18	0	5,305	6,443
XSVI	10	3	7	-	21	0	7,286	7,844
XSVI	10	3	8	-	24	0	8,651	9,088
XSVI	12	2	1	-	2	0	513	840
XSVI	12	2	2	-	4	0	799	1,356
XSVI	12	2	3	-	6	0	2,171	2,501
XSVI	12	2	4	-	8	0	2,932	3,464
XSVI	12	2	5	-	10	0	3,956	4,716
XSVI	12	2	6	-	12	0	5,476	5,757
XSVI	12	2	7	-	14	0	5,931	7,106
XSVI	12	2	8	-	16	0	7,853	8,257
XSVI	12	3	1	-	3	0	570	1,051
XSVI	12	3	2	-	6	0	1,002	1,727
XSVI	12	3	3	-	9	0	2,892	3,267
XSVI	12	3	4	-	12	0	3,796	4,485
XSVI	12	3	5	-	15	0	3,143	5,542
XSVI	12	3	6	-	18	0	7,038	7,485
XSVI	12	3	7	-	21	0	8,343	9,121
XSVI	12	3	8	-	24	0	8,003	10,309

1. Block RAM type for this device family is RAMB16BWER.
2. The "Maximum Screen Width" parameter does not affect the VFBC or XSVI layer resources.
3. All Graphics Controller configurations have default parameter settings.

Table 6: **Virtex-5 Resource Estimates**

Layer Type	Data Width	Channels	Layers	Maximum Screen Width(2)	XtremeDSP Slices	BRAM(1),(3)	LUTs	FFs
Graphics Controller	8	2	1	320	2	2	1,342	1,294
Graphics Controller	8	2	1	4096	2	3	1,381	1,306
Graphics Controller	8	2	2	320	4	4	2,611	2,363
Graphics Controller	8	2	2	4096	4	6	2,696	2,387
Graphics Controller	8	2	3	320	6	6	4,046	3,618

Table 6: **Virtex-5 Resource Estimates**

Graphics Controller	8	2	3	4096	6	9	4,173	3,654
Graphics Controller	8	2	4	320	8	8	5,385	4,792
Graphics Controller	8	2	4	4096	8	12	5,551	4,840
Graphics Controller	8	2	5	320	10	10	7,197	6,385
Graphics Controller	8	2	5	4096	10	15	7,402	6,445
Graphics Controller	8	2	6	320	12	12	8,669	7,689
Graphics Controller	8	2	6	4096	12	18	8,915	7,761
Graphics Controller	8	2	7	320	14	14	10,366	9,190
Graphics Controller	8	2	7	4096	14	21	10,656	9,274
Graphics Controller	8	2	8	320	16	16	11,913	10,568
Graphics Controller	8	2	8	4096	16	24	12,246	10,664
Graphics Controller	8	3	1	320	3	2	1,384	1,390
Graphics Controller	8	3	1	4096	3	3	1,421	1,402
Graphics Controller	8	3	2	320	6	4	2,726	2,548
Graphics Controller	8	3	2	4096	6	6	2,797	2,572
Graphics Controller	8	3	3	320	9	6	4,273	3,940
Graphics Controller	8	3	3	4096	9	9	4,385	3,976
Graphics Controller	8	3	4	320	12	8	5,689	5,215
Graphics Controller	8	3	4	4096	12	12	5,843	5,263
Graphics Controller	8	3	5	320	15	10	7,722	7,031
Graphics Controller	8	3	5	4096	15	15	7,907	7,091
Graphics Controller	8	3	6	320	18	12	9,300	8,459
Graphics Controller	8	3	6	4096	18	18	9,526	8,531
Graphics Controller	8	3	7	320	21	14	11,159	10,142

Table 6: **Virtex-5 Resource Estimates**

Graphics Controller	8	3	7	4096	21	21	11,422	10,226
Graphics Controller	8	3	8	320	24	16	12,821	11,654
Graphics Controller	8	3	8	4096	24	24	13,116	11,750
Graphics Controller	10	2	1	320	2	2	1,380	1,394
Graphics Controller	10	2	1	4096	2	3	1,422	1,406
Graphics Controller	10	2	2	320	4	4	2,699	2,550
Graphics Controller	10	2	2	4096	4	6	2,786	2,574
Graphics Controller	10	2	3	320	6	6	4,229	3,933
Graphics Controller	10	2	3	4096	6	9	4,351	3,969
Graphics Controller	10	2	4	320	8	8	5,631	5,215
Graphics Controller	10	2	4	1920	8	12	5,736	5,239
Graphics Controller	10	2	5	320	10	10	7,629	7,009
Graphics Controller	10	2	5	4096	10	15	7,837	7,069
Graphics Controller	10	2	6	320	12	12	9,193	8,446
Graphics Controller	10	2	6	4096	12	18	9,441	8,518
Graphics Controller	10	2	7	320	14	14	11,022	10,127
Graphics Controller	10	2	7	4096	14	21	11,315	10,211
Graphics Controller	10	2	8	320	16	16	12,679	11,652
Graphics Controller	10	2	8	4096	16	24	13,025	11,748
Graphics Controller	10	3	1	320	3	2	1,426	1,508
Graphics Controller	10	3	1	4096	3	3	1,461	1,520
Graphics Controller	10	3	2	320	6	4	2,824	2,770
Graphics Controller	10	3	2	4096	6	6	2,899	2,794
Graphics Controller	10	3	3	320	9	6	4,485	4,317

Table 6: **Virtex-5 Resource Estimates**

Graphics Controller	10	3	3	4096	9	9	4,594	4,353
Graphics Controller	10	3	4	320	12	8	5,979	5,720
Graphics Controller	10	3	4	4096	12	12	6,127	5,768
Graphics Controller	10	3	5	320	15	10	8,212	7,787
Graphics Controller	10	3	5	4096	15	15	8,400	7,847
Graphics Controller	10	3	6	320	18	12	9,912	9,376
Graphics Controller	10	3	6	4096	18	18	10,134	9,448
Graphics Controller	10	3	7	320	21	14	11,954	11,279
Graphics Controller	10	3	7	4096	21	21	12,215	11,363
Graphics Controller	10	3	8	320	24	16	13,748	12,966
Graphics Controller	10	3	8	4096	24	24	14,046	13,062
Graphics Controller	12	2	1	320	2	2	1,417	1,461
Graphics Controller	12	2	1	4096	2	3	1,459	1,473
Graphics Controller	12	2	2	320	4	4	2,794	2,672
Graphics Controller	12	2	2	4096	4	6	2,874	2,696
Graphics Controller	12	2	3	320	6	6	4,403	4,149
Graphics Controller	12	2	3	4096	6	9	4,526	4,185
Graphics Controller	12	2	4	320	8	8	5,880	5,505
Graphics Controller	12	2	4	4096	8	12	6,044	5,553
Graphics Controller	12	2	5	320	10	10	8,027	7,468
Graphics Controller	12	2	5	4096	10	15	8,233	7,528
Graphics Controller	12	2	6	320	12	12	9,702	9,006
Graphics Controller	12	2	6	4096	12	18	9,946	9,078
Graphics Controller	12	2	7	320	14	14	11,675	10,833

Table 6: **Virtex-5 Resource Estimates**

Graphics Controller	12	2	7	4096	14	21	11,973	10,917
Graphics Controller	12	2	8	320	16	16	13,442	12,473
Graphics Controller	12	2	8	4096	16	24	13,781	12,569
Graphics Controller	12	3	1	320	3	2	1,464	1,593
Graphics Controller	12	3	1	4096	3	3	1,502	1,605
Graphics Controller	12	3	2	320	6	4	2,922	2,926
Graphics Controller	12	3	2	4096	6	6	2,995	2,950
Graphics Controller	12	3	3	320	9	6	4,690	4,596
Graphics Controller	12	3	3	4096	9	9	4,800	4,632
Graphics Controller	12	3	4	320	12	8	6,260	6,093
Graphics Controller	12	3	4	4096	12	12	6,409	6,141
Graphics Controller	12	3	5	320	15	10	8,707	8,379
Graphics Controller	12	3	5	4096	15	15	8,892	8,439
Graphics Controller	12	3	6	320	18	12	10,513	10,094
Graphics Controller	12	3	6	4096	18	18	10,736	10,166
Graphics Controller	12	3	7	320	21	14	12,735	12,184
Graphics Controller	12	3	7	4096	21	21	12,991	12,268
Graphics Controller	12	3	8	320	24	16	14,651	14,013
Graphics Controller	12	3	8	4096	24	24	14,948	14,109
VFBC	8	2	1	-	2	0	310	497
VFBC	8	2	2	-	4	0	527	800
VFBC	8	2	3	-	6	0	906	1,294
VFBC	8	2	4	-	8	0	1,191	1,705
VFBC	8	2	5	-	10	0	1,940	2,541
VFBC	8	2	6	-	12	0	2,359	3,081
VFBC	8	2	7	-	14	0	2,991	3,821
VFBC	8	2	8	-	16	0	3,484	4,437

Table 6: **Virtex-5 Resource Estimates**

VFBC	8	3	1	-	3	0	323	581
VFBC	8	3	2	-	6	0	577	963
VFBC	8	3	3	-	9	0	1,037	1,582
VFBC	8	3	4	-	12	0	1,368	2,083
VFBC	8	3	5	-	15	0	2,306	3,130
VFBC	8	3	6	-	18	0	2,801	3,784
VFBC	8	3	7	-	21	0	3,555	4,694
VFBC	8	3	8	-	24	0	4,130	5,433
VFBC	10	2	1	-	2	0	328	564
VFBC	10	2	2	-	4	0	576	923
VFBC	10	2	3	-	6	0	1,023	1,512
VFBC	10	2	4	-	8	0	1,353	1,998
VFBC	10	2	5	-	10	0	2,261	3,003
VFBC	10	2	6	-	12	0	2,757	3,645
VFBC	10	2	7	-	14	0	3,502	4,532
VFBC	10	2	8	-	16	0	4,086	5,263
VFBC	10	3	1	-	3	0	343	649
VFBC	10	3	2	-	6	0	637	1,085
VFBC	10	3	3	-	9	0	1,191	1,810
VFBC	10	3	4	-	12	0	1,583	2,390
VFBC	10	3	5	-	15	0	2,713	3,638
VFBC	10	3	6	-	18	0	3,298	4,403
VFBC	10	3	7	-	21	0	4,221	5,483
VFBC	10	3	8	-	24	0	4,905	6,348
VFBC	12	2	1	-	2	0	347	619
VFBC	12	2	2	-	4	0	627	1,020
VFBC	12	2	3	-	6	0	1,144	1,691
VFBC	12	2	4	-	8	0	1,523	2,239
VFBC	12	2	5	-	10	0	2,579	3,401
VFBC	12	2	6	-	12	0	3,150	4,130
VFBC	12	2	7	-	14	0	4,026	5,153
VFBC	12	2	8	-	16	0	4,695	5,985
VFBC	12	3	1	-	3	0	365	735
VFBC	12	3	2	-	6	0	701	1,243
VFBC	12	3	3	-	9	0	1,343	2,091
VFBC	12	3	4	-	12	0	1,792	2,765
VFBC	12	3	5	-	15	0	3,121	4,233
VFBC	12	3	6	-	18	0	3,800	5,125
VFBC	12	3	7	-	21	0	4,880	6,394

Table 6: **Virtex-5 Resource Estimates**

VFBC	12	3	8	-	24	0	5,680	7,400
XSVI	8	2	1	-	2	0	359	579
XSVI	8	2	2	-	4	0	628	915
XSVI	8	2	3	-	6	0	1,063	1,466
XSVI	8	2	4	-	8	0	1,408	1,900
XSVI	8	2	5	-	10	0	2,294	2,786
XSVI	8	2	6	-	12	0	2,798	3,357
XSVI	8	2	7	-	14	0	3,515	4,124
XSVI	8	2	8	-	16	0	4,098	4,746
XSVI	8	3	1	-	3	0	396	694
XSVI	8	3	2	-	6	0	728	1,118
XSVI	8	3	3	-	9	0	1,266	1,821
XSVI	8	3	4	-	12	0	1,680	2,357
XSVI	8	3	5	-	15	0	2,821	3,480
XSVI	8	3	6	-	18	0	3,430	4,184
XSVI	8	3	7	-	21	0	4,304	5,139
XSVI	8	3	8	-	24	0	5,002	5,895
XSVI	10	2	1	-	2	0	390	656
XSVI	10	2	2	-	4	0	701	1,045
XSVI	10	2	3	-	6	0	1,216	1,698
XSVI	10	2	4	-	8	0	1,618	2,207
XSVI	10	2	5	-	10	0	2,697	3,268
XSVI	10	2	6	-	12	0	3,292	3,943
XSVI	10	2	7	-	14	0	4,141	4,860
XSVI	10	2	8	-	16	0	4,831	5,597
XSVI	10	3	1	-	3	0	434	799
XSVI	10	3	2	-	6	0	822	1,298
XSVI	10	3	3	-	9	0	1,473	2,140
XSVI	10	3	4	-	12	0	1,965	2,775
XSVI	10	3	5	-	15	0	3,346	4,132
XSVI	10	3	6	-	18	0	4,074	4,973
XSVI	10	3	7	-	21	0	5,140	6,126
XSVI	10	3	8	-	24	0	5,972	7,029
XSVI	12	2	1	-	2	0	420	732
XSVI	12	2	2	-	4	0	776	1,176
XSVI	12	2	3	-	6	0	1,371	1,931
XSVI	12	2	4	-	8	0	1,837	2,513
XSVI	12	2	5	-	10	0	3,095	3,751
XSVI	12	2	6	-	12	0	3,781	4,530

Table 6: **Virtex-5 Resource Estimates**

XSVI	12	2	7	-	14	0	4,776	5,597
XSVI	12	2	8	-	16	0	5,568	6,447
XSVI	12	3	1	-	3	0	473	903
XSVI	12	3	2	-	6	0	922	1,477
XSVI	12	3	3	-	9	0	1,681	2,458
XSVI	12	3	4	-	12	0	2,250	3,192
XSVI	12	3	5	-	15	0	3,877	4,785
XSVI	12	3	6	-	18	0	4,717	5,763
XSVI	12	3	7	-	21	0	5,968	7,112
XSVI	12	3	8	-	24	0	6,933	8,162

1. Block RAM type for this device family is RAMB36.
2. The "Maximum Screen Width" parameter does not affect the VFBC or XSVI layer resources.
3. All Graphics Controller configurations have default parameter settings.

Table 7: **Virtex-6 Resource Estimates**

Layer Type	Data Width	Channels	Layers	Maximum Screen Width(2)	XtremeDSP Slices	BRAM(1),(3)	LUTs	FFs
Graphics Controller	8	2	1	320	2	2	1,484	1,322
Graphics Controller	8	2	1	4096	2	3	1,375	1,335
Graphics Controller	8	2	2	320	4	4	2,692	2,325
Graphics Controller	8	2	2	4096	4	6	2,663	2,347
Graphics Controller	8	2	3	320	6	6	3,970	3,557
Graphics Controller	8	2	3	4096	6	9	4,040	3,593
Graphics Controller	8	2	4	320	8	8	5,430	4,710
Graphics Controller	8	2	4	4096	8	12	5,465	4,758
Graphics Controller	8	2	5	320	10	10	6,800	6,556
Graphics Controller	8	2	5	4096	10	15	6,910	6,343
Graphics Controller	8	2	6	320	12	12	8,585	7,566
Graphics Controller	8	2	6	4096	12	18	8,321	7,638
Graphics Controller	8	2	7	320	14	14	9,701	9,046

Table 7: **Virtex-6 Resource Estimates**

Graphics Controller	8	2	7	4096	14	21	9,600	9,124
Graphics Controller	8	2	8	320	16	16	11,175	10,886
Graphics Controller	8	2	8	4096	16	24	11,325	10,977
Graphics Controller	8	3	1	320	3	2	1,548	1,424
Graphics Controller	8	3	1	4096	3	3	1,448	1,437
Graphics Controller	8	3	2	320	6	4	2,877	2,511
Graphics Controller	8	3	2	4096	6	6	2,817	2,534
Graphics Controller	8	3	3	320	9	6	4,201	3,881
Graphics Controller	8	3	3	4096	9	9	4,258	3,917
Graphics Controller	8	3	4	320	12	8	5,821	5,136
Graphics Controller	8	3	4	4096	12	12	5,903	5,184
Graphics Controller	8	3	5	320	15	10	7,542	7,257
Graphics Controller	8	3	5	4096	15	15	7,664	6,998
Graphics Controller	8	3	6	320	18	12	9,133	8,344
Graphics Controller	8	3	6	4096	18	18	8,856	8,416
Graphics Controller	8	3	7	320	21	14	10,298	10,490
Graphics Controller	8	3	7	4096	21	21	10,309	10,566
Graphics Controller	8	3	8	320	24	16	11,946	12,063
Graphics Controller	8	3	8	4096	24	24	12,167	12,154
Graphics Controller	10	2	1	320	2	2	1,415	1,425
Graphics Controller	10	2	1	4096	2	3	1,410	1,438
Graphics Controller	10	2	2	320	4	4	2,789	2,511
Graphics Controller	10	2	2	4096	4	6	2,730	2,534
Graphics Controller	10	2	3	320	6	6	4,132	3,872

Table 7: **Virtex-6 Resource Estimates**

Graphics Controller	10	2	3	4096	6	9	4,132	3,908
Graphics Controller	10	2	4	320	8	8	5,677	5,132
Graphics Controller	10	2	4	4096	8	12	5,726	5,180
Graphics Controller	10	2	5	320	10	10	7,106	7,206
Graphics Controller	10	2	5	4096	10	15	6,988	6,967
Graphics Controller	10	2	6	320	12	12	9,017	8,707
Graphics Controller	10	2	6	4096	12	18	8,597	8,389
Graphics Controller	10	2	7	320	14	14	10,020	9,983
Graphics Controller	10	2	7	4096	14	21	10,086	10,061
Graphics Controller	10	2	8	320	16	16	11,738	12,023
Graphics Controller	10	2	8	4096	16	24	11,928	12,115
Graphics Controller	10	3	1	320	3	2	1,599	1,488
Graphics Controller	10	3	1	4096	3	3	1,484	1,501
Graphics Controller	10	3	2	320	6	4	2,944	2,732
Graphics Controller	10	3	2	4096	6	6	2,912	2,755
Graphics Controller	10	3	3	320	9	6	4,453	4,257
Graphics Controller	10	3	3	4096	9	9	4,423	4,295
Graphics Controller	10	3	4	320	12	8	6,061	5,890
Graphics Controller	10	3	4	4096	12	12	6,089	5,689
Graphics Controller	10	3	5	320	15	10	8,044	8,048
Graphics Controller	10	3	5	4096	15	15	8,040	7,753
Graphics Controller	10	3	6	320	18	12	9,673	9,711
Graphics Controller	10	3	6	4096	18	18	9,423	9,779
Graphics Controller	10	3	7	320	21	14	10,992	11,684

Table 7: **Virtex-6 Resource Estimates**

Graphics Controller	10	3	7	4096	21	21	9,591	11,760
Graphics Controller	10	3	8	320	24	16	12,823	13,448
Graphics Controller	10	3	8	4096	24	24	12,822	13,539
Graphics Controller	12	2	1	320	2	2	1,451	1,442
Graphics Controller	12	2	1	4096	2	3	1,464	1,454
Graphics Controller	12	2	2	320	4	4	2,872	2,633
Graphics Controller	12	2	2	4096	4	6	2,812	2,656
Graphics Controller	12	2	3	320	6	6	4,303	4,088
Graphics Controller	12	2	3	4096	6	9	4,362	4,124
Graphics Controller	12	2	4	320	8	8	5,888	5,660
Graphics Controller	12	2	4	4096	8	12	5,894	5,471
Graphics Controller	12	2	5	320	10	10	7,350	7,705
Graphics Controller	12	2	5	4096	10	15	7,253	7,426
Graphics Controller	12	2	6	320	12	12	9,306	9,317
Graphics Controller	12	2	6	4096	12	18	8,885	8,949
Graphics Controller	12	2	7	320	14	14	10,363	11,202
Graphics Controller	12	2	7	4096	14	21	10,463	11,283
Graphics Controller	12	2	8	320	16	16	12,234	12,918
Graphics Controller	12	2	8	4096	16	24	12,358	13,009
Graphics Controller	12	3	1	320	3	2	1,571	1,576
Graphics Controller	12	3	1	4096	3	3	1,514	1,586
Graphics Controller	12	3	2	320	6	4	3,034	2,888
Graphics Controller	12	3	2	4096	6	6	2,963	2,911
Graphics Controller	12	3	3	320	9	6	4,208	4,534

Table 7: **Virtex-6 Resource Estimates**

Graphics Controller	12	3	3	4096	9	9	4,681	4,573
Graphics Controller	12	3	4	320	12	8	6,284	6,297
Graphics Controller	12	3	4	4096	12	12	6,342	6,062
Graphics Controller	12	3	5	320	15	10	8,305	8,691
Graphics Controller	12	3	5	4096	15	15	8,275	8,346
Graphics Controller	12	3	6	320	18	12	10,197	10,494
Graphics Controller	12	3	6	4096	18	18	9,803	10,562
Graphics Controller	12	3	7	320	21	14	11,439	12,665
Graphics Controller	12	3	7	4096	21	21	10,297	12,741
Graphics Controller	12	3	8	320	24	16	13,457	14,590
Graphics Controller	12	3	8	4096	24	24	13,318	14,681
VFBC	8	2	1	320	2	0	304	497
VFBC	8	2	2	320	4	0	482	800
VFBC	8	2	3	320	6	0	757	1,294
VFBC	8	2	4	320	8	0	1,014	1,705
VFBC	8	2	5	320	10	0	1,493	2,542
VFBC	8	2	6	320	12	0	1,807	3,085
VFBC	8	2	7	320	14	0	2,218	3,826
VFBC	8	2	8	320	16	0	2,543	4,437
VFBC	8	3	1	320	3	0	343	581
VFBC	8	3	2	320	6	0	572	963
VFBC	8	3	3	320	9	0	882	1,582
VFBC	8	3	4	320	12	0	1,183	2,083
VFBC	8	3	5	320	15	0	1,750	3,131
VFBC	8	3	6	320	18	0	2,168	3,788
VFBC	8	3	7	320	21	0	2,577	4,696
VFBC	8	3	8	320	24	0	2,988	5,433
VFBC	10	2	1	320	2	0	343	564
VFBC	10	2	2	320	4	0	528	923
VFBC	10	2	3	320	6	0	878	1,512
VFBC	10	2	4	320	8	0	1,121	1,998
VFBC	10	2	5	320	10	0	1,734	3,004

Table 7: **Virtex-6 Resource Estimates**

VFBC	10	2	6	320	12	0	2,133	3,649
VFBC	10	2	7	320	14	0	2,546	4,537
VFBC	10	2	8	320	16	0	3,018	5,263
VFBC	10	3	1	320	3	0	380	649
VFBC	10	3	2	320	6	0	621	1,085
VFBC	10	3	3	320	9	0	1,004	1,811
VFBC	10	3	4	320	12	0	1,306	2,390
VFBC	10	3	5	320	15	0	2,017	3,638
VFBC	10	3	6	320	18	0	2,552	4,407
VFBC	10	3	7	320	21	0	3,141	5,485
VFBC	10	3	8	320	24	0	3,647	6,348
VFBC	12	2	1	320	2	0	370	619
VFBC	12	2	2	320	4	0	565	1,020
VFBC	12	2	3	320	6	0	913	1,691
VFBC	12	2	4	320	8	0	1,253	2,239
VFBC	12	2	5	320	10	0	1,928	3,402
VFBC	12	2	6	320	12	0	2,374	4,134
VFBC	12	2	7	320	14	0	2,902	5,158
VFBC	12	2	8	320	16	0	3,317	5,985
VFBC	12	3	1	320	3	0	379	735
VFBC	12	3	2	320	6	0	623	1,243
VFBC	12	3	3	320	9	0	1,079	2,091
VFBC	12	3	4	320	12	0	1,454	2,765
VFBC	12	3	5	320	15	0	2,329	4,234
VFBC	12	3	6	320	18	0	2,912	5,129
VFBC	12	3	7	320	21	0	3,432	6,399
VFBC	12	3	8	320	24	0	4,091	7,400
XSVI	8	2	1	320	2	0	363	579
XSVI	8	2	2	320	4	0	572	915
XSVI	8	2	3	320	6	0	932	1,466
XSVI	8	2	4	320	8	0	1,246	1,900
XSVI	8	2	5	320	10	0	1,874	2,787
XSVI	8	2	6	320	12	0	2,295	3,361
XSVI	8	2	7	320	14	0	2,737	4,128
XSVI	8	2	8	320	16	0	3,201	4,746
XSVI	8	3	1	320	3	0	403	694
XSVI	8	3	2	320	6	0	688	1,118
XSVI	8	3	3	320	9	0	1,135	1,822
XSVI	8	3	4	320	12	0	1,501	2,357

Table 7: **Virtex-6 Resource Estimates**

XSVI	8	3	5	320	15	0	2,345	3,481
XSVI	8	3	6	320	18	0	2,869	4,188
XSVI	8	3	7	320	21	0	3,430	5,144
XSVI	8	3	8	320	24	0	3,994	5,895
XSVI	10	2	1	320	2	0	413	656
XSVI	10	2	2	320	4	0	660	1,045
XSVI	10	2	3	320	6	0	1,077	1,698
XSVI	10	2	4	320	8	0	1,431	2,207
XSVI	10	2	5	320	10	0	2,166	3,268
XSVI	10	2	6	320	12	0	2,732	3,947
XSVI	10	2	7	320	14	0	3,266	4,865
XSVI	10	2	8	320	16	0	3,777	5,597
XSVI	10	3	1	320	3	0	456	799
XSVI	10	3	2	320	6	0	817	1,298
XSVI	10	3	3	320	9	0	1,325	2,140
XSVI	10	3	4	320	12	0	1,806	2,775
XSVI	10	3	5	320	15	0	2,768	4,133
XSVI	10	3	6	320	18	0	3,455	4,977
XSVI	10	3	7	320	21	0	4,049	6,131
XSVI	10	3	8	320	24	0	4,705	7,029
XSVI	12	2	1	320	2	0	429	732
XSVI	12	2	2	320	4	0	738	1,176
XSVI	12	2	3	320	6	0	1,173	1,931
XSVI	12	2	4	320	8	0	1,595	2,513
XSVI	12	2	5	320	10	0	2,464	3,752
XSVI	12	2	6	320	12	0	3,057	4,534
XSVI	12	2	7	320	14	0	3,651	5,601
XSVI	12	2	8	320	16	0	4,321	6,447
XSVI	12	3	1	320	3	0	504	903
XSVI	12	3	2	320	6	0	878	1,477
XSVI	12	3	3	320	9	0	1,476	2,459
XSVI	12	3	4	320	12	0	1,976	3,192
XSVI	12	3	5	320	15	0	3,204	4,786
XSVI	12	3	6	320	18	0	3,936	5,767
XSVI	12	3	7	320	21	0	4,564	7,117
XSVI	12	3	8	320	24	0	5,426	8,162

1. Block RAM type for this device family is RAMB36E1.
2. The "Maximum Screen Width" parameter does not affect the VFBC or XSVI layer resources.
3. All Graphics Controller configurations have default parameter settings.

Performance

Table 8 lists typical clock frequencies for the target families.The maximum achievable clock frequency could vary and in most cases will be higher. The maximum achievable clock frequency and all resource counts may be affected by other tool options, additional logic in the FPGA device, using a different version of Xilinx tools, and other factors.

Table 8: **Target Family Clock Frequencies**

Device Family	Maximum Clock Frequency (FMax)
For Virtex-6 devices	225 MHz
For Spartan6 devices	150 MHz
For Virtex-5 devices	225MHz
For Spartan-3A DSP devices	150 MHz

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*.

Refer to the IP Release Notes Guide (XTP025) for further information on this core. There will be a link to all the DSP IP and then to the relevant core being designed with. For each core, there is a master Answer Record that contains the Release Notes and Known Issues list for the core being used. The following information is listed for each version of the core:

- New Features
- Bug Fixes
- Known Issues

Licensing Options

The Xilinx Video On-Screen Display LogiCORE system 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 the core functionality with your own design and demonstrates the various interfaces on 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 back-annotated gatelevel simulation of the core using the demonstration test bench provided with the 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. This core is configured to timeout after 8 hours of operation.

Full

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

- Functional simulation support
- Back annotated gate-level 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

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.
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, 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.

Ordering Information

The Xilinx Video On-Screen Display v2.0 core is provided under the SignOnce IP Site License and can be generated using the Xilinx CORE Generator system v13.1 or higher. The CORE Generator system is shipped with Xilinx ISE Design Suite development software.

A simulation evaluation license for the core is shipped with the CORE Generator system. To access the full functionality of the core, including FPGA bitstream generation, a full license must be obtained from Xilinx. For more information, please visit the product page for this core.

Please contact your local Xilinx sales representative for pricing and availability of additional Xilinx LogiCORE modules and software. Information about additional Xilinx LogiCORE modules is available on the Xilinx IP Center.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
09/16/09	1.0	Initial Xilinx release.
03/01/11	2.0	Updated core to version 2.0.

Notice of Disclaimer