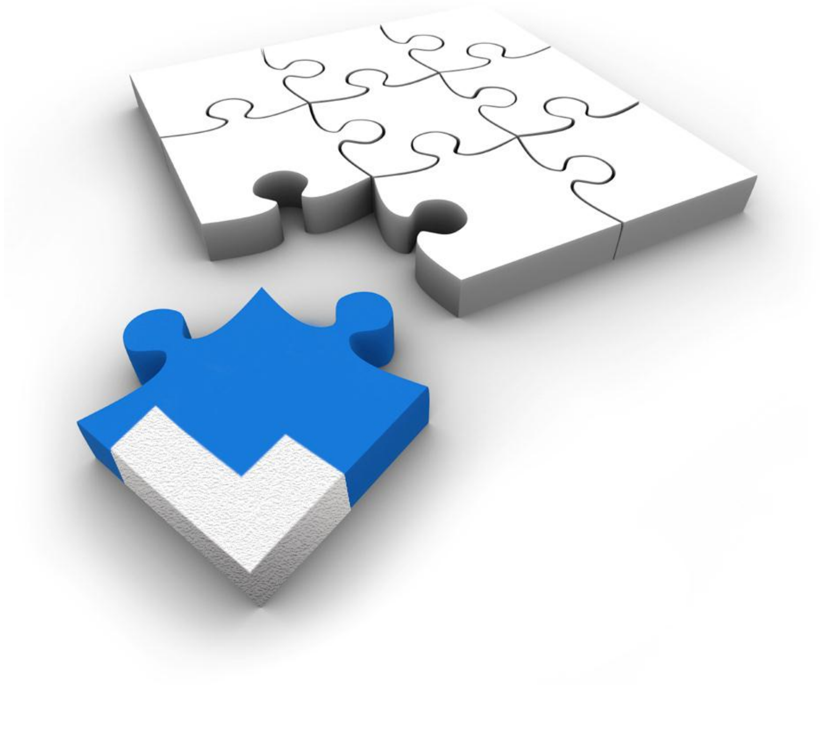
logiCVC-ML

User's Manual

Version: 5.0.1

logiCVC-ML_hum_v5.0.1.docx



1 INTRODUCTION

The logiCVC-ML Compact Multilayer Video Controller is a graphics/video display controller optimized for Xilinx Zynq^{TM}-7000 All Programmable SoC and FPGA devices. logiCVC-ML rich feature set is optimal for embedded electronic devices. It provides all the necessary control signals to interface directly with LCD and other flat panel displays. A wide variety of display types are supported. logiCVC-ML compact size, low slice count utilization can be additionally decreased through feature set configuration by code parameterization.

Its functions include refreshing the display image by reading the video memory and converting the read data into a data stream acceptable for the display interface. It also generates control signals for the display. Multilayer support provides on screen display functions: alpha blending, color keyed transparency among layers, hardware cursors and fast scrolling and pan functions. Powerful blending features enable easy mixing of streaming videos, which can be processed by custom made DSP modules, with graphics content generated by the CPU and/or graphics accelerators. All of these features are supported by hardware and require only low CPU processing power.

Optional processing functions, like bit-block transfer unit, generators or frame grabbing, can easily be added through the integration of other graphic logicBRICKS^{TM} IP cores.

Xylon provides a number of application notes, reference designs, evaluation boards and software drivers. All these simplify integration of the logiCVC-ML with your design, shortening time to the market and increasing your market window. To learn more, please visit:

http://www.logicbricks.com/Products/logiCVC-ML.aspx

1.1 General Description

The logiCVC-ML controls TFT flat panel displays. In order to control other LCD technologies like TNM and STNM, scaled down logiCVC is available upon request. logiCVC-ML supports many commonly used digital video interfaces allowing the user easy integration into its system. By means of an external video digital to analog converter (DAC) it also controls S-Video, Composite Video devices, CRT displays (VGA monitors, for example) and CVBS displays.

The interface to the frame buffer or the video memory is targeted for the SDRAM (SDR, DDR, …) or SRAM implementation. For easier system integration, logiCVC-ML uses ARM AMBA AXI4 buses for accessing the frame buffer and logiCVC-ML register. The logiCVC-ML requires relatively low memory bandwidth which depends on the selected display control parameters. When the video memory is implemented in high bandwidth memory, such as SDRAM, the same memory can be used as the CPU working memory.

This type of hardware architecture is known as UMA (Unified Memory Architecture) and is extremely efficient in the implementation of low-cost systems. Only 37% of the 16-bit SDRAM memory bandwidth is used for refreshing the 800x600 TFT display image, while the remaining 63% can be used by either the CPU, graphic processor, or some other unit.

1.2 Features

- Supports Xilinx® Zynq®-7000 All Programmable SoC and 7 series FPGAs
- Please contact Xylon for Spartan®-6, Virtex®-6, Spartan®-3 and other Xilinx FPGA family support
- Display controller IP core for LCD, CRT and other displays types
- Available SW drivers for Linux, Android™ QNX and Microsoft® Windows® Embedded Compact
- Display resolutions up to 8192x8192 including 4K2K at 60fps
- Supports up to 5 layers, the last one configurable as background color
- Programmable layer address, size and position
- Alpha blending and color keyed transparency allows blending of video and graphics content on the screen
- Pixel, layer or color lookup table (CLUT) alpha blending mode can be set for each layer independently
- Packed pixel layer memory organization supports 3, 6, 8 and 10 bits per color:
    - RGB - 8bpp 3-3-2, 8bpp with Color Look up Table, 16bpp 5-6-5, 24bpp 8-8-8 and 30bpp 10-10-10
    - YCbCr - 16bpp (4:2:2), 20bpp (4:2:2), 24bpp (4:4:4), 30bpp (4:4:4)
- Support for multiple display interfaces:
    - Parallel display data bus (RGB or YCbCr) with 1, 2 or 4 pix per clock: 12x2-bit, 15, 16, 18, 20, 24, 30 bit
    - AXI4-Stream with 1, 2 or 4 pix per clock
    - Digital video ITU-656: PAL and NTSC
    - LVDS output format: 3 or 4 data pairs plus clock
    - Camera link output format: 4 data pairs plus clock
    - DVI output format
- ARM® AMBA® AXI4 memory interface with configurable data width (32, 64, 128 or 256 bits)
- Simple programming of control registers through AXI4-Lite interface
- Supports synchronization to external parallel or AXI4-Stream input (data used as one layer)
- Double/triple buffering enables flicker free reproduction of live video streams overlaid by graphics HMI
- Display power-on sequencing control signals
- Parametrical VHDL design that allows tuning of slice consumption and features set
- Prepared for Xilinx Vivado® Design Suite implementation tool (Xilinx Platform Studio support on request)
- Simple Plug'n'Play with other Xylon logicBRICKS™ IP cores, such as:
    - logiISP Image Signal Processing (ISP) Pipeline
    - logiBITBLT Bit Block Transfer 2D Graphic Accelerator
    - logi3D Scalable 3D Graphics Accelerator
    - logiWIN Versatile Video Input Controller

2 ARCHITECTURE

The logiCVC-ML IP core's main functional modules are:

- Video Memory Access Block
- Video Address Generator
- Sync Generator
- Multilayer Alpha Blender
- Output Standard Converter
- Registers

2.1 Block Schematic

Block schematic on Figure 2.1 shows a basic architecture of logiCVC-ML. Depending on configuration parameters some of the outlined blocks are not used.

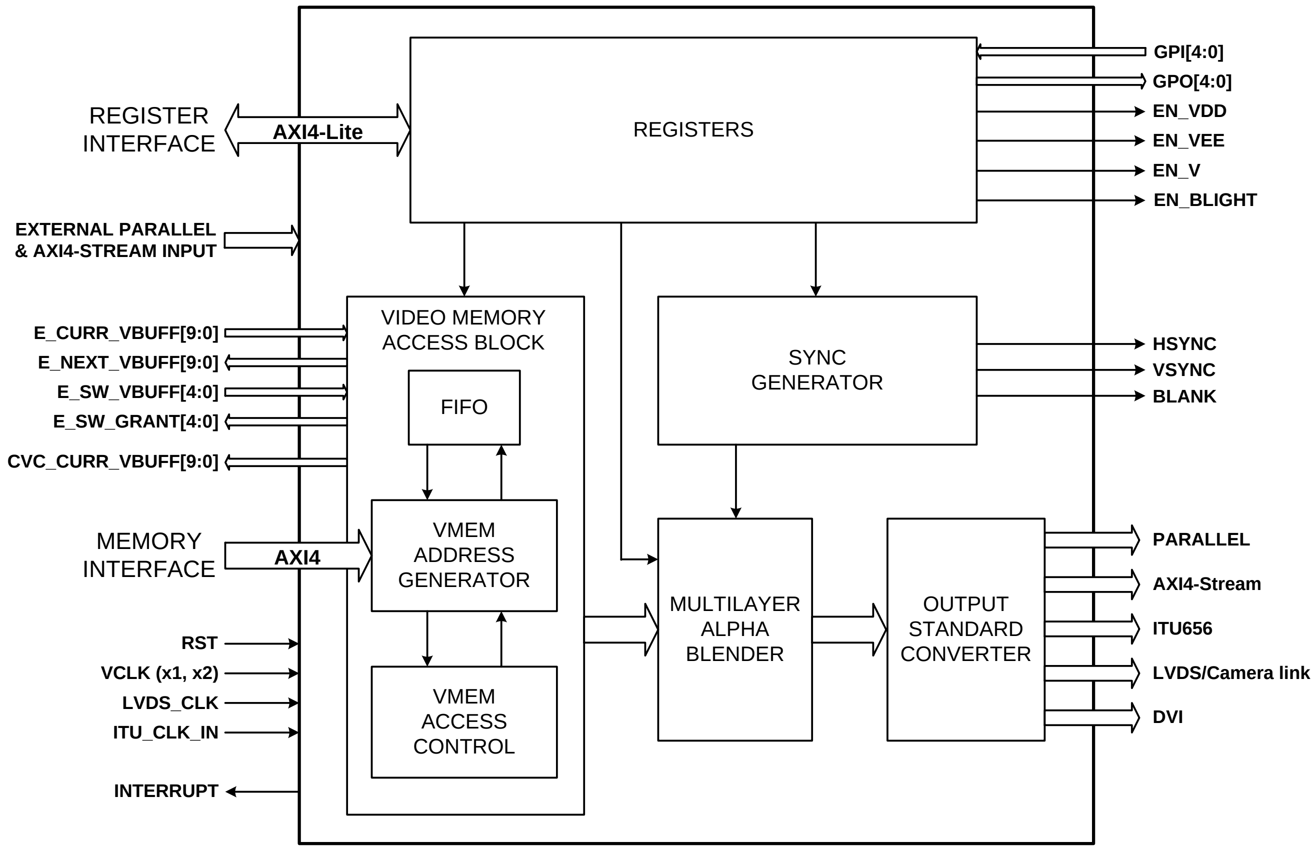


Figure 2.1: logiCVC-ML Architecture

2.1.1 Video Memory Access Block

The Video Memory Access Block consists of three sub modules: Video Memory Access Control, Video Memory Address Generator and FIFOs. The Video Memory Access Block fetches video data from the video memory over AXI4 to the local FIFOs. The Video Address Generator calculates video memory pointers for each layer. The Video Memory Access Block ensures that each FIFO is filled with the required amount of pixels and it performs arbitration between memory requests of each layer.

There is one FIFO per layer used for temporary storage of pixels. The FIFOs optimize the usage of video memory bandwidth and resynchronize incoming data to the display clock.

2.1.2 Sync Generator

The Sync Generator generates video synchronization signals. The duration of sync signals, their relative position to the display data (i.e., visible picture on the screen) and polarity can be adjusted through the set of logiCVC-ML registers. Porch, resolution and sync registers are used for this purpose.

All synchronization signals can also be resynchronized on external input video synchronization signals. This feature is needed for overlay function when external video data is not stored in the memory used by logiCVC-ML.

2.1.3 Multilayer Alpha Blender

The Multilayer Alpha blender block consists of a maximum of five; user configurable layer blocks. The outputs of the layer blocks are converted to the appropriate color space and mixed according to alpha/transparent factors and layer priority. The output of the layer mixer is then routed towards the output standard convertor. The Blender supports layer, pixel and color alpha blending methods. For additional alpha blending information, refer to chapter 9 MULTILAYER SUPPORT.

2.1.4 Output Standard Converter

The Output Standard Converter receives pixel data and control signals and converts them to requested video output format defined in Vivado GUI. It can output ITU656, LVDS (LVDS with one clock and three or four data pairs or Camera link with one clock and four data pairs), DVI, parallel (RGB, YCbCr 4:4:4 or YCbCr 4:2:2) or AXI4-Stream format. Therefore, it consists of five main modules: ITU656 generator, LVDS generator, DVI generator, YCbCr 4:4:4 to 4:2:2 converter and AXI4-Stream generator.

2.1.5 Registers

The Video Control Register Block is made up of two sub-modules: General logiCVC-ML registers and Layer specific registers.

General logiCVC-ML registers include Horizontal and Vertical resolution and sync registers. These registers control horizontal and vertical timing in pixel clock increments, such as HSYNC/VSYNC active state, delay from HSYNC/VSYNC inactive to data start, delay from end of data up to HSYNC active and delay from VSYNC inactive to first visible pixel line start.

The group of general logiCVC-ML registers also includes Display Type and Control, Background color, Power control and IP version registers. The Display Type and Control Register are the two registers used for programming different display types.

Registers can be accessed through the AXI4-Lite interface.

For more information on logiCVC-ML registers, refer to chapter 10 REGISTERS.

2.2 Pin-out

Table 2.1: Pin-out

Signal name	Type	Description
Global Signals		
RST	I	Global reset input, high active
VCLK	I	Video clock input
VCLK2	I	Video clock input x2 (used for 12bit parallel multiplexed output interface)
ITU_CLK_IN	I	ITU656 clock input (27 MHz and synchronous to VCLK)
LVDS_CLK	I	LVDS clock
Memory Interface		
AXI4 Master Interface	Bus	Refer to Xilinx AXI Reference Guide, UG761
Register Interface		
AXI4-Lite Slave Interface	Bus	Refer to Xilinx AXI Reference Guide, UG761
Display Control Signals		
HSYNC	IO 1)	Horizontal sync
VSYNC	IO 1)	Vertical sync
PIX_CLK	IO 1)	Pixel clock
PIX_CLKN	IO 1))	Pixel clock inverted
BLANK	IO 1)	Blank/display enable
D_PIX[n : 0]	IO 1)	Video pixel data bus
LVDS_DATA_OUT_P[3:0]	O	LVDS / Camera link pixel data, positive
LVDS_DATA_OUT_N[3:0]	O	LVDS / Camera link pixel data, negative
LVDS_CLK_OUT_P	O	LVDS / Camera link clock, positive
LVDS_CLK_OUT_N	O	LVDS / Camera link clock, negative
ITU656_CLK	O	ITU656 clock
ITU656_DATA[7:0]	O	ITU656 data
DVI_CLK_P	O	DVI clock, positive
DVI_CLK_N	O	DVI clock, negative

Signal name	Type	Description
DVI_DATA_P[2:0]	O	DVI pixel data, positive
DVI_DATA_N[2:0]	O	DVI pixel data, negative
AXI4-Stream Master Video Interface	Bus	Refer to Xilinx AXI Reference Guide, UG761
External Video Input Signals		
E_VCLK	I	External VCLK (used when external parallel input is used)
E_VSYNC	I	External VSYNC (used when external parallel input is used)
E_HSYNC	I	External HSYNC (used when external parallel input is used)
E_BLANK	I	External BLANK (used when external parallel input is used)
E_DATA[n : 0]	I	External RGB data (used when external parallel input is used)
E_VIDEO_PRESENT	I	External video present (used when external parallel input is used)
AXI4-Stream Slave Video Interface	Bus	Refer to Xilinx AXI Reference Guide, UG761
Auxiliary Signals		
E_CURR_VBUFF[9:0]	I	Current external stream video memory buffer (two bits per layer)
E_NEXT_VBUFF[9:0]	O	Next external stream video memory buffer to write to (two bits per source)
E_SW_VBUFF[4:0]	I	External switch logiCVC-ML video memory buffers (one bit per layer)
E_SW_GRANT[4:0]	O	External switch grant (one bit per source, handshaking signal for E_SW_VBUFF)
CVC_CURR_VBUFF[9:0]	O	Current CVC reading video buffer (two bits per layer)
GPI[4:0]	I	General purpose input
GPO[4:0]	O	General purpose output
INTERRUPT	O	logiCVC Interrupt signal, level sensitive, high active
EN_VDD	O	Enable Vdd power supply
EN_BLIGHT	O	Enable backlight power supply
EN_V	O	Enable display control/data signals
EN_VEE	O	Enable Vee power supply

1. Parallel pixel data interface signals are all defined as input/output so that they are three stated after FPGA boot-up and then enabled by using the power control register. Because the three state buffers only exist in FPGA IO buffers, these signals can only be connected to an FPGA IOs. However, logiCVC-ML has three additional signals; _i, _o and _t that can be used when there is a requirement to use the logiCVC-ML output stream inside the FPGA. In this case, the _o signals (e.g. vsync_o) can be sourced inside the FPGA for further use and the other two, _i and _t, can be left unconnected.

3 CUSTOMIZATION

The logiCVC-ML IP core is prepackaged for Xilinx Vivado Design Suite. IP customization requires no skills beyond general tools knowledge and can be used in same ways as Xilinx IP cores. System designers can easily setup the logiCVC-ML configuration by setting up all IP core's parameters through an easy-to-use Vivado GUI interface.

The logiCVC-ML configuration is defined by VHDL generic parameters. However, when using Vivado GUI the user does not directly change these parameters but instead the configuration is handled through the Vivado GUI which takes care to configure these parameters according to user selection. This allows an easier, higher level configuration which is much simpler for the user. Additionally, the GUI configuration scripts delivered with the logiCVC-ML IP core avoid and restrict possible forbidden configuration combinations which reduces user's design development effort.

The following chapters describe all logiCVC-ML GUI settings that can be modified prior to logiCVC-ML synthesis and implementation.

For additional reference, each GUI setting name described in the following chapters includes a corresponding VHDL generic parameter name outlined in brackets ().

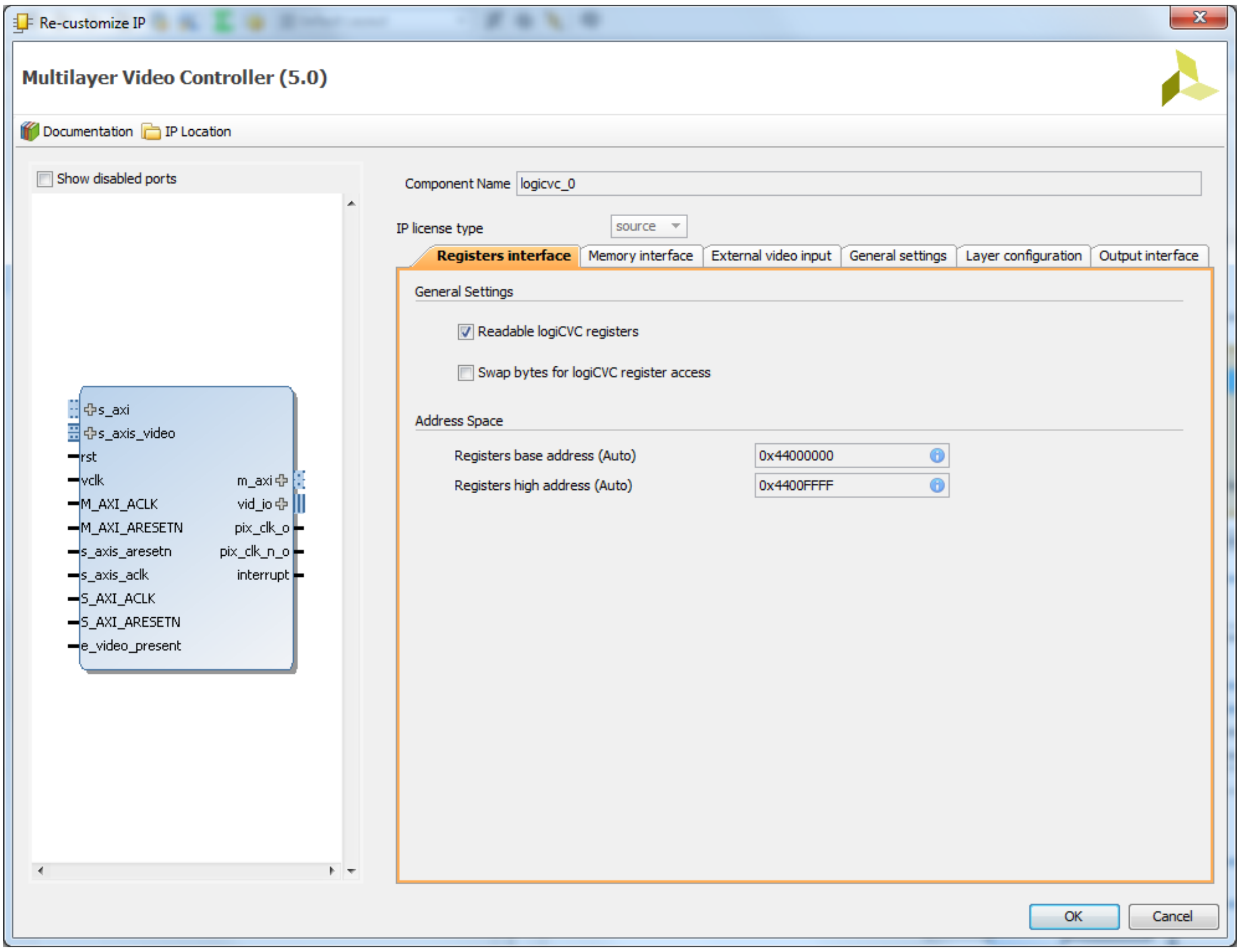
When double clicking on the IP from the Vivado IP catalog or in the Vivado Block Design, the logiCVC-ML configuration GUI opens up as outlined in Figure 3.1.

On the top of the configuration GUI, above all configuration tabs, user can check the license version that the logiCVC-ML will search for at implementation time. There are three License types available as described in Table 3.1.

Table 3.1: logiCVC-ML License Types

License type	License required	Timeout	Encrypted source files
Source	No	No	No
Evaluation	Yes (evaluation)	Yes	Yes
Release	Yes (release)	No	Yes

3.1 Registers Interface Tab



Component Name: logicvc_0

IP license type: source

General Settings

[✓] Readable logiCVC registers

[ ] Swap bytes for logiCVC register access

Address Space

Registers base address (Auto): 0x44000000

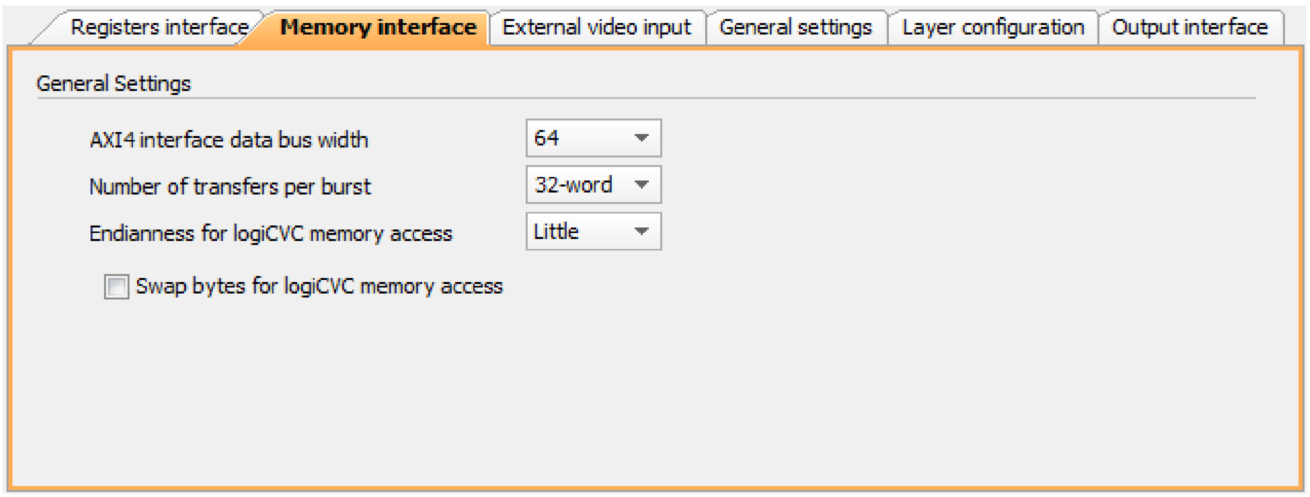
Registers high address (Auto): 0x4400FFFF

Figure 3.1: logiCVC-ML Registers Configuration Tab

- **General Settings**
  - **Readable logiCVC registers** (C_READABLE_REGS)
    When selected, all logiCVC-ML registers are readable through the AXI4-Lite registers interface. Deselecting this option reduces resource utilization by disabling readability of logiCVC-ML registers that do not need to be read for proper control and functioning of the IP core. For a list of these registers, please refer to Table 10.1.
  - **Swap bytes for logiCVC register access** (C_REG_BYTE_SWAP)
    If this parameter is set, byte ordering on the register data bus will change ($B_0B_1B_2B_3$ => $B_3B_2B_1B_0$).
- **Address Space**
  - **Registers base/high address** (C_REGS_BASEADDR, C_REGS_HIGHADDR)
    These parameters are read-only and can be set in Vivado Block Design Address Editor.

3.2 Memory Interface Tab

Stating from this tab onwards, the figures showing the configuration GUI will be cropped for easier reference, i.e. the left part of the tab showing the IP block with pin-out will be omitted.



General Settings

AXI4 interface data bus width: 64

Number of transfers per burst: 32-word

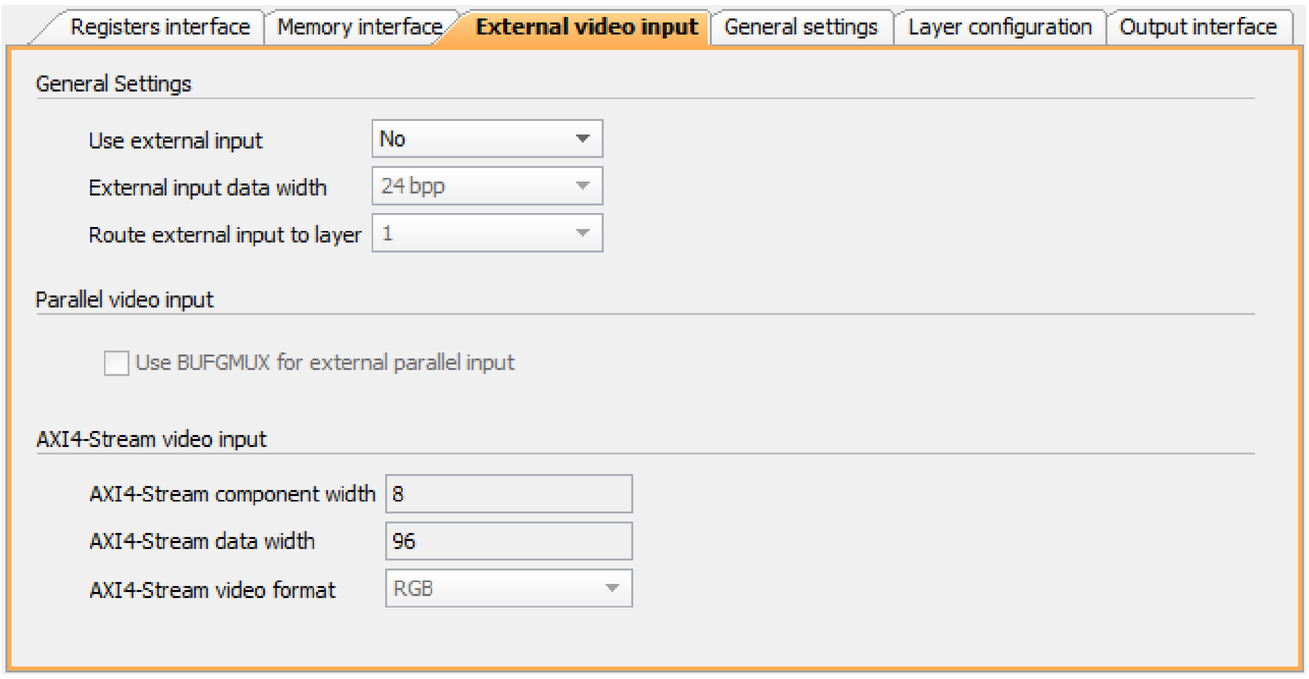
Endianness for logiCVC memory access: Little

☐ Swap bytes for logiCVC memory access

Figure 3.2: logiCVC-ML Memory Interface Configuration Tab

- **General Settings**
  - **AXI4 interface data bus width** (C_M_AXI_DATA_WIDTH)
    logiCVC-ML supports different AXI4 Master data width configurations. Selecting a wider data bus increases available memory bandwidth allowing higher resolution and frame rate support but at the same time increases resource utilization.
  - **Number of transfers per burst** (C_MEM_BURST)
    This parameter defines the maximum burst length that logiCVC will use while accessing memory frame buffer. Increasing burst size increases the efficiency of memory transfers which effectively increases memory bandwidth available to logiCVC-ML.
  - **Endianness for logiCVC memory access** (C_MEM_LITTLE_ENDIAN)
    Choose between little or big endian memory access. For more information on endian support, please refer to chapter 5.1 Memory Layout.
  - **Swap bytes for logiCVC memory access** (C_MEM_BYTE_SWAP)
    If set this option will change byte ordering on memory interface bus. For more information please refer to chapter 5.1 Memory Layout.

3.3 External Video Input Tab



General Settings

Use external input: No

External input data width: 24 bpp

Route external input to layer: 1

Parallel video input

Use BUFGMUX for external parallel input ☐

AXI4-Stream video input

AXI4-Stream component width: 8

AXI4-Stream data width: 96

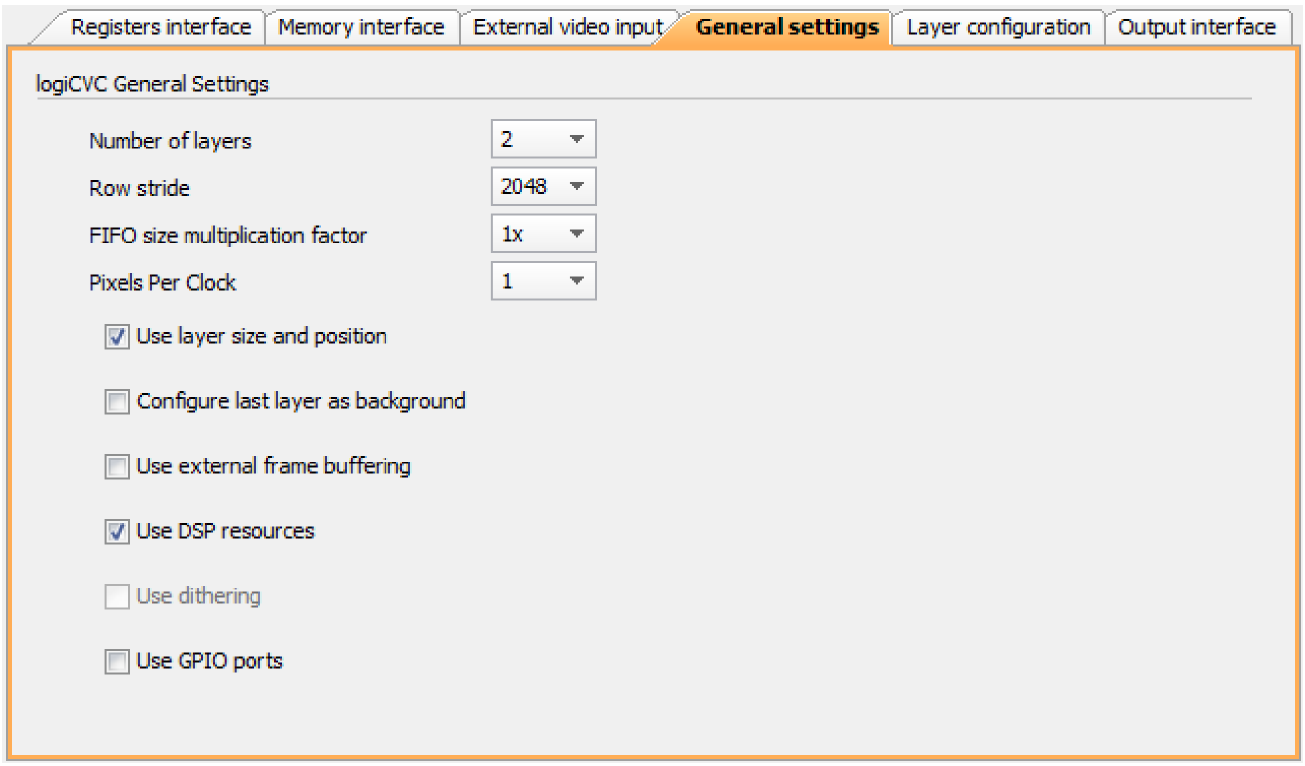
AXI4-Stream video format: RGB

Figure 3.3: logiCVC-ML External Video Input Tab

- **General Settings**
  - **Use external input** (C_USE_E_INPUT)
    Enable external parallel or AXI4-Stream input interface. When set, the data for the selected layer is not read out from memory frame buffer but is received through this interface. Additionally, the complete logiCVC-ML synchronizes to this external input stream so that other layers data can be over or under laid accordingly. For more information on external video input, please refer to chapter 8 EXTERNAL VIDEO INPUT INTERFACE.
  - **External input data width** (C_E_DATA_WIDTH)
    Data width of the external video interface. The selected width must be the same as the data width of the layer the external input is routed to.
  - **Route external input to layer** (C_E_LAYER)
    This option selects the logiCVC-ML layer to which the data from the external video input is routed to.
- **Parallel video input**
  - **Use BUFGMUX for external parallel input** (C_USE_E_VCLK_BUFGMUX)
    Enables using BUFGMUX component for switching clock when external parallel input is used.
- **AXI4-Stream video input**
  - **AXI4-Stream component width** (C_S_AXIS_DATA_WIDTH)
    Color component data width parameter is automatically calculated from External input data width parameter.

- **AXI4-Stream data width** (C_S_AXIS_TDATA_WIDTH)
  Bus data width parameter is automatically calculated from component width and pixels per clock (Output interface tab) parameters.
- **AXI4-Stream video format** (C_S_AXIS_VIDEO_FORMAT)
  Video format is automatically determined from layer type parameter (Layer configuration tab) of the selected External input layer.

3.4 General Settings Tab



Number of layers: 2
Row stride: 2048
FIFO size multiplication factor: 1x
Pixels Per Clock: 1

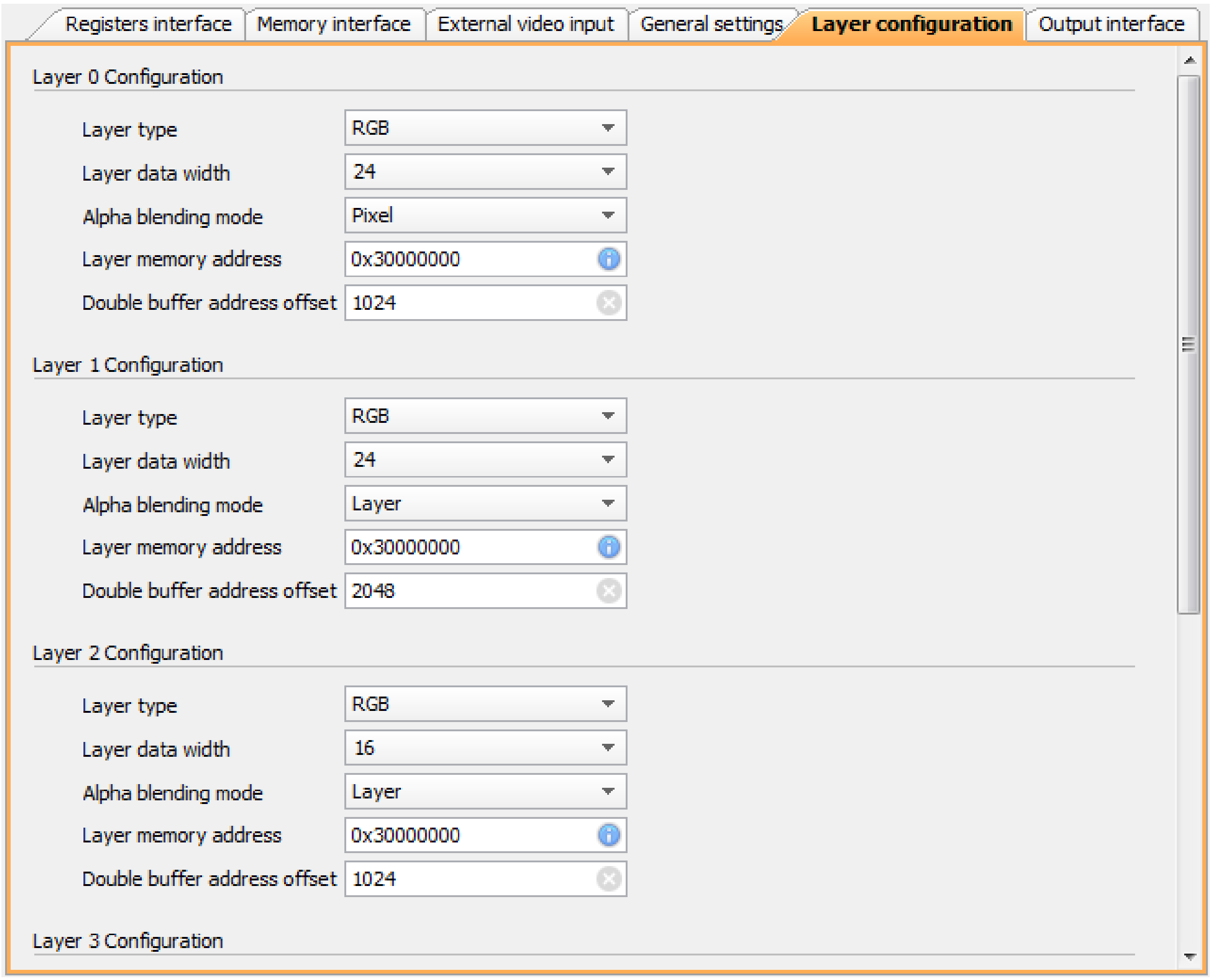
- [✓] Use layer size and position
- [ ] Configure last layer as background
- [ ] Use external frame buffering
- [✓] Use DSP resources
- [ ] Use dithering
- [ ] Use GPIO ports

Figure 3.4: logiCVC-ML General Settings Tab

- **logiCVC General Settings**
  - **Number of layers** (C_NUM_OF_LAYERS)
    Number of logiCVC layers to be instantiated.
  - **Row stride** (C_ROW_STRIDE)
    This parameter selects the distance between same column pixels for adjacent rows and is specified in number of pixels. For more information on row stride, please refer to chapter 5.1 Memory Layout.
  - **FIFO size multiplication factor** (C_INCREASE_FIFO)
    Increasing the size of layer FIFOs increases BRAM utilization but allows logiCVC to sustain good output picture when large memory bandwidth fluctuations are present in the system.

- **Pixels per clock** (C_PIXEL_PER_CLOCK)
  This option enables parallel pixel processing which reduces pixel clock frequency while maintaining the same pixel throughput. This option is only available when using parallel or AXI4-Stream output interfaces.
- **Use layer size and position** (C_USE_SIZE_POSITION)
  This option enables runtime programmable layer size and position functionality. Disabling it reduces resource utilization.
- **Configure last layer as background** (C_USE_BACKGROUND)
  By enabling this option the last layer is configured as background layer which means that pixel value is read from logiCVC background color register and not from video memory buffer. The result is that the last layer will have a solid color fill.
- **Use external frame buffering** (C_USE_EXT_BUFFERING)
  Enables logiCVC-ML ports used for external control of frame buffering. For more information, please refer to chapter 5.4 Frame Buffer Synchronization.
- **Use DSP resources** (C_USE_XTREME_DSP)
  This option allows the user to choose between using DSP or LUT resources for all alpha blending calculations. Enabling this option instructs the synthesis tool to use dedicated DSP resources for alpha blending.
- **Use dithering** (C_XCOLOR)
  Enabling pixel dithering increases the number of colors reproduced when using 18 bits per pixel output interface. For more information, please refer to chapter 7.6 Dithering module.
- **Use GPIO ports** (C_USE_GPIO)
  Enable general purpose input and output ports.

3.5 Layer Configuration Tab



Layer 0 Configuration

Layer type	RGB
Layer data width	24
Alpha blending mode	Pixel
Layer memory address	0x30000000
Double buffer address offset	1024

Layer 1 Configuration

Layer type	RGB
Layer data width	24
Alpha blending mode	Layer
Layer memory address	0x30000000
Double buffer address offset	2048

Layer 2 Configuration

Layer type	RGB
Layer data width	16
Alpha blending mode	Layer
Layer memory address	0x30000000
Double buffer address offset	1024

Layer 3 Configuration

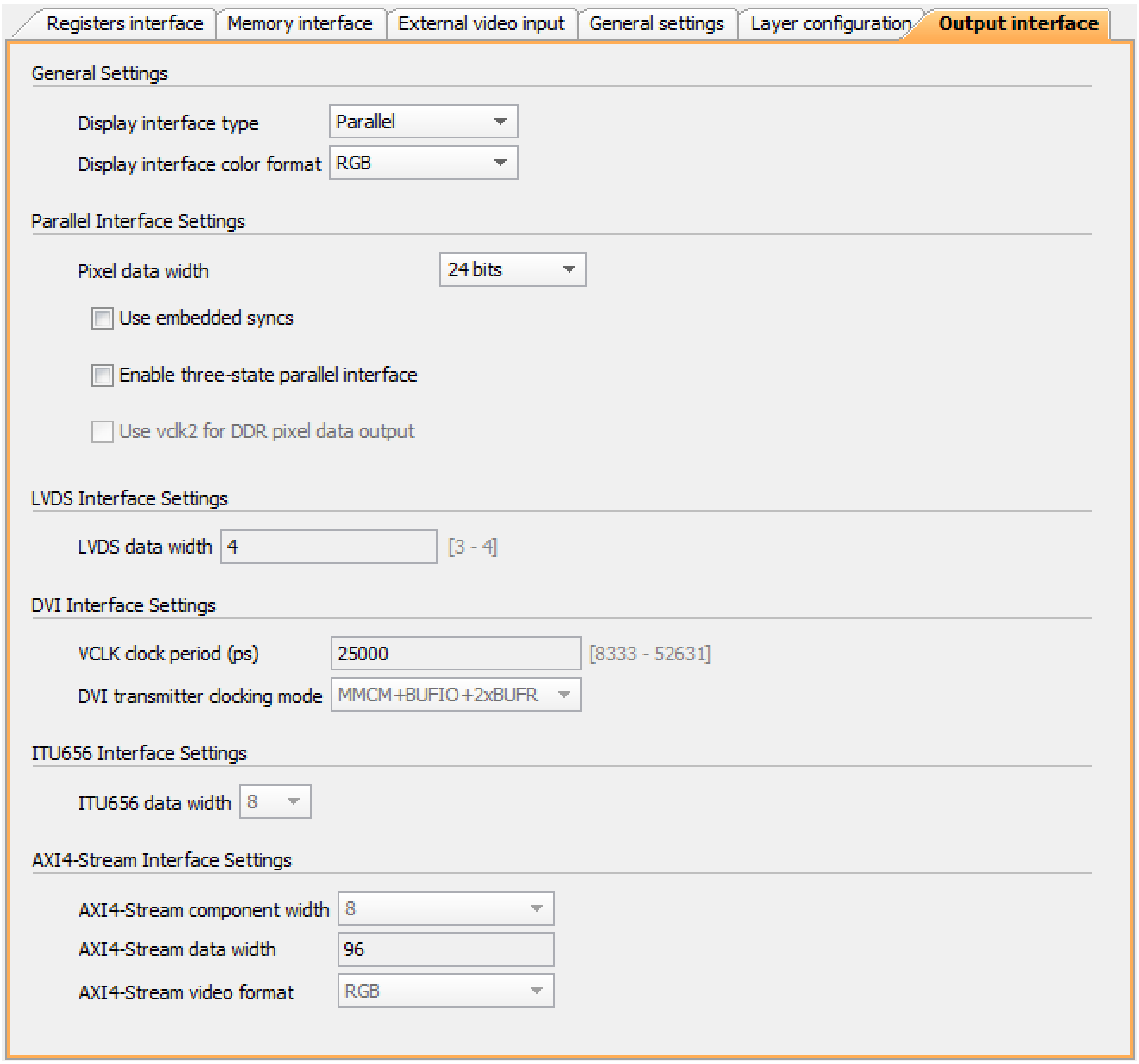
Figure 3.5: logiCVC-ML Layer Configuration Tab

- **Layer Configuration**
  - **Layer type** (C_LAYER_x_TYPE)
    This parameter defines whether the layer is setup as RGB, YCbCr or alpha plane layer. Each layer can be configured to be RGB or YCbCr but alpha plane layers can only be layer 1 and 3. For more information on alpha pane layers, please refer to chapter 9.3.2.2 Pixel Alpha Blending.
  - **Layer data width** (C_LAYER_x_DATA_WIDTH)
    This parameter defines the pixel depth for the selected layer. Supported values are 8, 16, 20, 24 and 30 bits per pixel.
    For an RGB layer type, 30 represents an RGB 101010 format, 24 represents an RGB 888 format, 16 represents RGB 565 format, while 8 can represent either RGB 332 or CLUT format, depending on layer alpha mode selection.

For a YCbCr layer, 30 represents an YCbCr 101010 (4:4:4) format, 24 represents a YCbCr 888 (4:4:4) format, 20 represents a YCb YCr 10 10 (4:2:2) format and 16 represents a YCb YCr 88 88 (4:2:2) format.

- **Alpha blending mode** (C_LAYER_x_ALPHA_MODE)
  Each layer can be configured to use different alpha blending mode; layer, pixel or CLUT. For more information on each of these modes, please refer to chapter 9.3.2 Alpha Blending.
- **Layer memory address** (C_LAYER_x_ADDR)
  This parameter defines the default layer memory address. This value can be overridden runtime using layer memory address register described in chapter 0.
- **Double buffer address offset** (C_LAYER_x_OFFSET)
  This parameter is only used with external double/triple buffer switching. Double buffer address offset relative to layer memory address represented in number of lines. Triple buffer address offset is defined as double the double buffer offset. For more information on double buffer offset, please refer to chapter 5.3 Memory address and range.

3.6 Output Interface Tab



General Settings

Display interface type	Parallel
Display interface color format	RGB

Parallel Interface Settings

Pixel data width	24 bits

- [ ] Use embedded syncs
- [ ] Enable three-state parallel interface
- [ ] Use vclk2 for DDR pixel data output

LVDS Interface Settings

LVDS data width	4	[3 - 4]

DVI Interface Settings

VCLK clock period (ps)	25000	[8333 - 52631]
DVI transmitter clocking mode	MMCM+BUFIO+2xBUFR	

ITU656 Interface Settings

ITU656 data width	8

AXI4-Stream Interface Settings

AXI4-Stream component width	8
AXI4-Stream data width	96
AXI4-Stream video format	RGB

Figure 3.6: logiCVC-ML Output Interface Tab

- **General Settings**
    - **Display interface type** (C_DISPLAY_INTERFACE)
      This parameter defines the logiCVC-ML output interface. Independent of the setting, parallel interface is always active e.g. selecting LVDS output interface will enable it but the parallel stream provided to the internal LVDS serializer will also be present on the output of the logiCVC-ML.

- **Display interface color format** (C_DISPLAY_COLOR_SPACE)
  This parameter defines the pixel color format that is sent out over the selected interface type. Not all color formats are supported with every interface type and the configuration GUI will report if the wrong combination is setup. E.g. ITU 656 interface type can only use YCbCr 4:2:2 color format.

- **Parallel Interface Settings**
  - **Pixel data width** (C_PIXEL_DATA_WIDTH)
    This parameter defines the width of the parallel pixel data bus.
  - **Use embedded syncs** (C_USE_EMB_SYNC)
    Enabling this option instructs the logiCVC-ML to embed the control signals (hsync, vsync, and blank) into the data stream using special sync codes. For more information on embedded syncs, please refer to chapter 7.1.
  - **Enable three-state parallel interface** (C_ENABLE_THREE_STATE_CONTROL)
    Three-state parallel interface signals include power sequence control and three-stated parallel pixel interface.
    Parallel pixel data interface signals, besides the output signals, can also include input (_i) and three state control signals (_t) allowing the bus to use IOB three state buffers so that after FPGA boot-up it is three-stated and then enabled by using the power control register.
  - **Use vclk2 for DDR pixel data output** (C_USE_VCLK2)
    This option is only available when using 12 bit DDR pixel data width and allows changing the alignment of pixel clock relative to the pixel data bus. When enabled, pixel clock rising edge is set in the middle of the DDR data eye. When disabled, pixel clock rising edge is synchronous to the data bus.
- **LVDS Interface Settings**
  - **LVDS data width** (C_LVDS_DATA_WIDTH)
    This parameter is automatically calculated from display interface type setting.
- **DVI Interface Settings**
  - **VCLK clock period (ps)** (C_LVDS_DATA_WIDTH)
    This parameter is used to properly configure PLL inside the DVI transmitter logic.
  - **DVI transmitter clocking mode** (C_DVI_CLK_MODE)
    For more information on setting this parameter, please refer to chapter 7.4 DVI Interface.
- **ITU656 Interface Settings**
  - **ITU656 data width** (C_ITU656_DATA_WIDTH)
    This parameter defines the width of the data bus when ITU656 interface type is used.
- **AXI4-Stream Interface Settings**
  - **AXI4-Stream component width** (C_M_AXIS_DATA_WIDTH)
    This parameter defines the width of the color component.
  - **AXI4-Stream data width** (C_M_AXIS_TDATA_WIDTH)
    Bus data width parameter is automatically calculated from component width and pixels per clock (Output interface tab) parameters.
  - **AXI4-Stream video format** (C_M_AXIS_VIDEO_FORMAT)
    Video format is automatically determined from display interface color format parameter.

4 CLOCK AND RESET

The logiCVC-ML has a global reset input (rst) that must be connected to a valid FPGA system rest. After assertion of the reset signal, logiCVC-ML will reset all internal logic to its default state. For proper reset sequence, reset signal must stay active for at least two cycles of every clock connected to logiCVC-ML.

For proper operation, user must always connect at least three clock signals to logiCVC-ML: video clock, memory interface clock and register interface clock.

The video clock signal, VCLK, controls most circuits inside the logiCVC-ML core, except the memory sub-system (AXI4) related circuits and registers (AXI4-Lite). Memory sub-system uses M_AXI_ACLK, and register sub-system uses S_AXI_ACLK clock signals depending on the interface selection.

In general, the VCLK frequency depends on display resolution and characteristics however, other clock frequencies are applicable.

logiCVC-ML does not require any phase or frequency relationship between the video, memory and register clock.

In addition to the three mandatory clock signals mentioned above, logiCVC-ML can use additional clock signals depending on its configuration.

In case ITU656 output interface is used (“*Display interface type”* parameter), user must supply ITU656 clock signal, ITU_CLK_IN, which has a frequency of 27 MHz as requested by the ITU656 standard. Additionally, VCLK and ITU_CLK_IN must be fully synchronous and VCLK must have a frequency of 13.5 MHz.

In case 12 bit DDR parallel data interface is used (“*Pixel data width”* parameter) and “*Use vclk2 for DDR pixel data output”* parameter is set, in addition to VCLK also VCLK2 signal must be supplied to logiCVC-ML input. VCLK2 has to be double the VCLK frequency and synchronous to VCLK.

In case LVDS or camera link output standard is used (“*Display interface type”* parameter), user must supply an additional LVDS_CLK signal which must have 7 times higher frequency and be synchronous to VCLK. Additionally, these two clock signals must be sourced by the same MMCM module with same buffer types, i.e. if VCLK is driven by a BUFG, LVDS_CLK must be driven by a BUFG as well.

4.1 Pixel Clock Output

The pixel clock output, PIX_CLK, is proportional to VCLK clock input or E_VCLK clock input (if external parallel input is used) and to the control bits in the DTYPE and CTRL register. Please refer to chapter 10.2 Register Description.

To support the functionality of adjustable PIX_CLK clock frequencies and consequently different display resolutions, special clock modules outside of the logiCVC-ML core must be used. The following list outlines three solutions on how to support the adjustable pixel clock frequencies:

- An FPGA external PLL module that can be programmed in a wide range of video frequencies. Usually, these ICs are controlled via an I^2C or SPI interface.
- Using the logiCLK IP from the Xylon's logicBRICKS IP library, which uses a Xilinx FPGA PLL component programmable through the AXI4-Lite interface.
- A custom FPGA clock module with predefined clock frequencies. This module can then be controlled with the GPO signals, which are outputs from the logiCVC-ML core.

5 MEMORY INTERFACE

The video image is stored in video memory, which is accessible through AXI4 interface.

The logiCVC-ML is primarily designed for use with logiMEM – Flexible SDR/DDR memory controller, a member of Xylon's IP library named the logicBRICKS^{TM}. However, any memory controller that supports AXI4 interface can be used.

The logiCVC-ML AXI4 master interface can be configured as 32-bit, 64-bit, 128-bit or 256-bit wide by setting the *AXI4 interface data bus width* parameter from Vivado GUI interface. For better AXI4 and memory bandwidth utilization, logiCVC-ML supports setting the maximum memory burst ("*Number of transfers per burst"* parameter) used when accessing memory. The out of block accesses are obtained by using single AXI4 cycles and bursts lower than defined by the above mentioned parameter.

For more information on AXI4 bus architecture, please refer to Xilinx AXI Reference Guide, UG761.

5.1 Memory Layout

logiCVC-ML uses a rectangular memory configuration. This means that horizontal width of the image stored in the memory is defined by pixel row stride and not the horizontal resolution of the video screen.

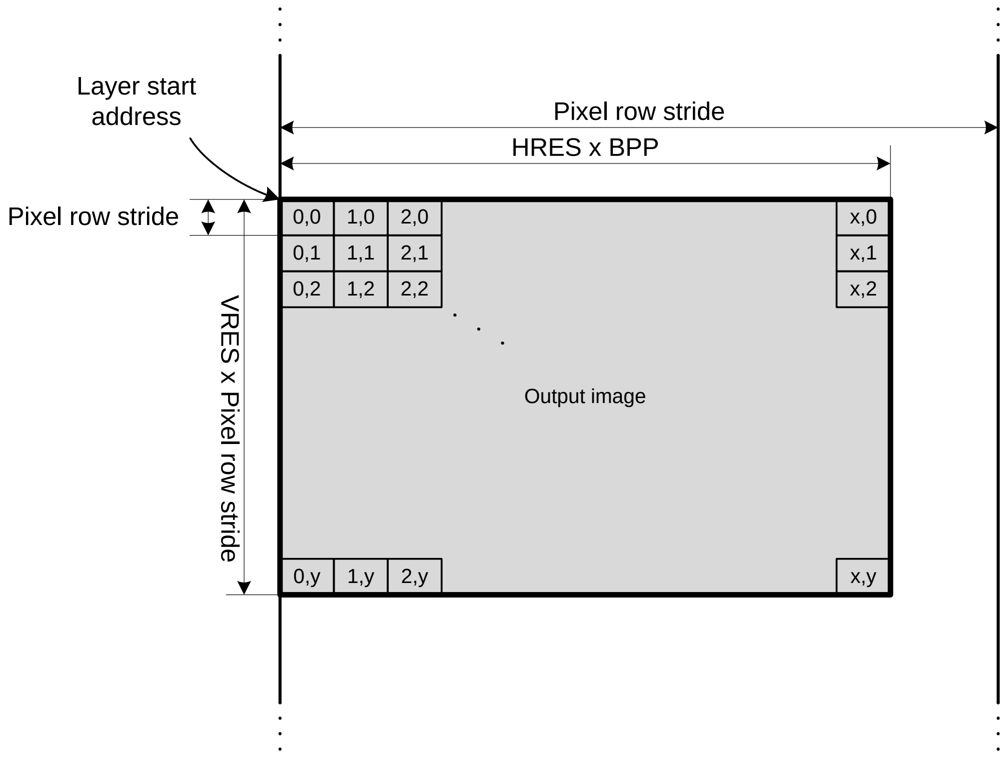


Figure 5.1: Rectangular memory layout

Pixel row stride is defined as the distance in bytes between same colon pixels for adjacent rows. Depending on layer data width (bpp), alpha blending mode and *"Row stride"* parameter, stride is set to 0.5, 1, 2, 4 or 8-kilobyte boundary.

Table 5.1: Pixel row stride

Alpha blending mode	8bpp	16bpp	24bpp
layer	1**Row stride* (B)	2**Row stride* (B)	4**Row stride* (B)
pixel	2**Row stride* (B)	4**Row stride* (B)	4**Row stride* (B)
CLUT	1**Row stride* (B)	-	-

From the above figure and table, we can see that pixel row stride defines the address of the first pixel in each line. For example, line 3 has a starting address of (2 x pixel row stride) relative to layer address.

As logiCVC-ML supports different layer configurations, the actual memory address where each pixel is stored in memory depends on the following six factors: layer address register, layer data width (bpp), layer type (RGB, YCbCr), layer alpha blending mode, byte swapping and endianess.

Table 5.2, Table 5.3 and Table 5.4 describe memory layout configurations according to layer width, layer type and layer alpha blending mode. All three tables assume that *"Endianness for logiCVC memory access"* parameter is set to *Little* and *"Swap bytes for logiCVC memory access"* parameter is not enabled.

Table 5.2: 8bpp layer memory layout

Address	Layer alpha	CLUT indexed	Pixel alpha
0	Pix0	Pix0 index	Pix0
1	Pix1	Pix1 index	
2	Pix2	Pix2 index	Pix1
3	Pix3	Pix3 index	
4	Pix4	Pix4 index	Pix2
5	Pix5	Pix5 index	
6	Pix6	Pix6 index	Pix3
7	Pix7	Pix7 index	

Table 5.3: 16bpp layer memory layout

Address	Layer alpha	Pixel alpha
0	Pix0	Pix0
1		
2	Pix1	
3		
4	Pix2	Pix1
5		
6	Pix3	
7		

Table 5.4: 24bpp and 32bpp layer memory layout

Address	Layer or Pixel alpha
0	Pix0
1	
2	
3	
4	Pix1
5	
6	
7	

Byte swapping configuration affects how logiCVC-ML reads video memory. All bytes in memory data bus are swapped if "*Swap bytes for logiCVC memory access"* parameter is enabled. Example on memory layout depending on this parameter is given in Table 5.5:

Table 5.5: Memory layout depending on "*Swap bytes for logiCVC memory access*"

Data bus width [1)]	**32**			**64**		
C_MEM_BYTE_SWAP		**0**	**1**		**0**	**1**
Data Bus	[7:0]	Byte 0	Byte 3	[7:0]	Byte 0	Byte 7
	[15:8]	Byte 1	Byte 2	[15:8]	Byte 1	Byte 6
	[23:16]	Byte 2	Byte 1	[23:16]	Byte 2	Byte 5
	[31:24]	Byte 3	Byte 0	[31:24]	Byte 3	Byte 4
	[7:0]	Byte 4	Byte 7	[39:32]	Byte 4	Byte 3
	[15:8]	Byte 5	Byte 6	[47:40]	Byte 5	Byte 2
	[23:16]	Byte 6	Byte 5	[55:48]	Byte 6	Byte 1
	[31:24]	Byte 7	Byte 4	[63:56]	Byte 7	Byte 0

1. Data bus width is defined by *"AXI4 interface data bus width"* parameter

Parameter *"Endianness for logiCVC memory access"* determines if little endian or big endian is used for data representation in memory. It reorders amount of data depending on *"Layer Data width"* parameter. If *"Layer data width"* is set to 24 then 4 bytes of data are swapped. In case *"Layer data width"* is set to 16 then data of 2 bytes are swapped and if *"Layer data width"* is set to 8 then 1-byte data are swapped. Example on memory layout depending on *"Endianness for logiCVC memory access"* parameter when memory data bus is 32 and 64-bit is outlined in Table 5.6 and Table 5.7.

Table 5.6: Memory layout depending on endianess (32-bit data bus)

Layer data width		**30 or 24bpp**		**16bpp**		**8bpp**	
Endianness		**Little**	**Big**	**Little**	**Big**	**Little**	**Big**
Address	**0**	Pix0	Pix0	Pix 0	Pix 1	Pix 0	Pix 3
	1					Pix 1	Pix 2
	2			Pix 1	Pix 0	Pix 2	Pix 1
	3					Pix 3	Pix 0

Table 5.7: Memory layout depending on endianess (64-bit data bus)

Layer data width		**30 or 24bpp**		**16bpp**		**8bpp**	
Endianness		**Little**	**Big**	**Little**	**Big**	**Little**	**Big**
Address	**0**	Pix0	Pix1	Pix 0	Pix 3	Pix 0	Pix 7
	1					Pix 1	Pix 6
	2			Pix 1	Pix 2	Pix 2	Pix 5
	3					Pix 3	Pix 4
	4	Pix1	Pix0	Pix 2	Pix 1	Pix 4	Pix 3
	5					Pix 5	Pix 2
	6			Pix 3	Pix 0	Pix 6	Pix 1
	7					Pix 7	Pix 0

5.2 Pixel Layout

The above chapter describes the layout of pixel locations in memory while this chapter focuses on the pixel layout itself.

Pixel layout depends on the following four factors: layer data width (bpp), layer type (RGB, YCbCr), layer alpha blending mode and pixel format (Layer Control Register).

5.2.1 8bpp

Table 5.8: 8bpp pixel layout

Pixel bit	Layer Alpha	CLUT Indexed	Pixel Alpha	Alpha Layer (Alpha Plane)
0	Blue0	Pix index0	Blue0	Alpha0
1	Blue1	Pix index1	Blue1	Alpha1
2	Green0	Pix index2	Green0	Alpha2
3	Green1	Pix index3	Green1	Alpha3
4	Green2	Pix index4	Green2	Alpha4
5	Red0	Pix index5	Red0	Alpha5
6	Red1	Pix index6	Red1	Alpha6
7	Red2	Pix index7	Red2	Alpha7
8			Alpha0	
9			Alpha1	
10			Alpha2	
11			Dummy	
12			Dummy	
13			Dummy	
14			Dummy	
15			Dummy	

In 8bpp with pixel alpha mode, only 3 bits are used because alpha factor is multiplied with every color separately. As 8bpp is defined as RGB 332, 3-bit alpha is used for red and green, and 2-bit for blue. Nevertheless, user has to make sure that in this mode alpha factor spreads from 0 to 7.

Table 5.8 outlines default pixel format (RGB). For more information on changing the order of colors inside the pixel, please refer to chapter 10.2.19 Layer Control Register - Lx_CTRL.

logiCVC-ML supports Layers 1 and 3 to be set as Alpha Layers, which means that they consist only of alpha blending factors and not color values (RGB or YCbCr). In case Layer 1 is configured as Alpha Layer (*"Layer type"* set to *Alpha*), data read from memory is used as pixel alpha factor for calculation between Layers 0 and 2. The same applies for Layer 3 with the exception that it is used for calculation between Layers 2 and 4.

5.2.2 16bpp, 24bpp and 30bpp

Table 5.9: 16bpp, 24bpp and 30bpp RGB pixel layout

Address	Pixel bit	16bpp		24bpp		30bpp
		Layer alpha	Pixel alpha	Layer alpha	Pixel alpha	Layer alpha
0	0	Blue0	Blue0	Blue0	Blue0	Blue0
	1	Blue1	Blue1	Blue1	Blue1	Blue1
	2	Blue2	Blue2	Blue2	Blue2	Blue2
	3	Blue3	Blue3	Blue3	Blue3	Blue3
	4	Blue4	Blue4	Blue4	Blue4	Blue4
	5	Green0	Green0	Blue5	Blue5	Blue5
	6	Green1	Green1	Blue6	Blue6	Blue6
	7	Green2	Green2	Blue7	Blue7	Blue7
1	8	Green3	Green3	Green0	Green0	Blue8
	9	Green4	Green4	Green1	Green1	Blue9
	10	Green5	Green5	Green2	Green2	Green0
	11	Red0	Red0	Green3	Green3	Green1
	12	Red1	Red1	Green4	Green4	Green2
	13	Red2	Red2	Green5	Green5	Green3
	14	Red3	Red3	Green6	Green6	Green4
	15	Red4	Red4	Green7	Green7	Green5
2	16		Dummy	Red0	Red0	Green6
	17		Dummy	Red1	Red1	Green7
	18		Dummy	Red2	Red2	Green8
	19		Dummy	Red3	Red3	Green9
	20		Dummy	Red4	Red4	Red0
	21		Dummy	Red5	Red5	Red1
	22		Dummy	Red6	Red6	Red2
	23		Dummy	Red7	Red7	Red3
3	24		Alpha0	Dummy	Alpha0	Red4
	25		Alpha1	Dummy	Alpha1	Red5
	26		Alpha2	Dummy	Alpha2	Red6
	27		Alpha3	Dummy	Alpha3	Red7
	28		Alpha4	Dummy	Alpha4	Red8
	29		Alpha5	Dummy	Alpha5	Red9
	30		Dummy	Dummy	Alpha6	Dummy
	31		Dummy	Dummy	Alpha7	Dummy

Table 5.10: 16bpp and 24bpp YCbCr pixel layout

Address	Pixel bit	16bpp (4:2:2)	24bpp (4:4:4)		30bpp (4:4:4)
		Layer alpha	Layer alpha	Pixel alpha	Layer alpha
0	0	$Y_0$0	Cr0	Cr0	Cr0
	1	$Y_0$1	Cr1	Cr1	Cr1
	2	$Y_0$2	Cr2	Cr2	Cr2
	3	$Y_0$3	Cr3	Cr3	Cr3
	4	$Y_0$4	Cr4	Cr4	Cr4
	5	$Y_0$5	Cr5	Cr5	Cr5
	6	$Y_0$6	Cr6	Cr6	Cr6
	7	$Y_0$7	Cr7	Cr7	Cr7
1	8	Cb0	Cb0	Cb0	Cr8
	9	Cb1	Cb1	Cb1	Cr9
	10	Cb2	Cb2	Cb2	Cb0
	11	Cb3	Cb3	Cb3	Cb1
	12	Cb4	Cb4	Cb4	Cb2
	13	Cb5	Cb5	Cb5	Cb3
	14	Cb6	Cb6	Cb6	Cb4
	15	Cb7	Cb7	Cb7	Cb5
2	16	$Y_1$0	Y0	Y0	Cb6
	17	$Y_1$1	Y1	Y1	Cb7
	18	$Y_1$2	Y2	Y2	Cb8
	19	$Y_1$3	Y3	Y3	Cb9
	20	$Y_1$4	Y4	Y4	Y0
	21	$Y_1$5	Y5	Y5	Y1
	22	$Y_1$6	Y6	Y6	Y2
	23	$Y_1$7	Y7	Y7	Y3
3	24	Cr0	Dummy	Alpha0	Y4
	25	Cr1	Dummy	Alpha1	Y5
	26	Cr2	Dummy	Alpha2	Y6
	27	Cr3	Dummy	Alpha3	Y7
	28	Cr4	Dummy	Alpha4	Y8
	29	Cr5	Dummy	Alpha5	Y9
	30	Cr6	Dummy	Alpha6	Dummy
	31	Cr7	Dummy	Alpha7	Dummy

The above tables outline default pixel format (RGB or YCbCr). For more information on changing color order inside the pixel, please refer to chapter 10.2.19 Layer Control Register - Lx_CTRL.

5.3 Memory address and range

The logiCVC-ML uses a 32bit address bus to access video data from memory so a frame buffer can be stored at any location inside the 2^{32} address space.

Each logiCVC-ML layer has a configurable memory address that can be assigned using layer address registers. The default values of layer address registers can be assigned using the *"Layer memory address"* parameters. Therefore, if there is no requirement for runtime changing of layer memory address pointers the user does not need to configure the layer address registers beside setting the mentioned parameters prior to synthesis.

As explained above, each layer memory start address is assigned directly with corresponding layer address register.

Layer address range, i.e. size, is defined with several other parameters and must be taken into consideration when assigning layer addresses so that layer address ranges do not overlap.

The following parameters define each layer memory address and range:

- Layer address register; default value is defined with *"Layer memory address"* parameter
- Pixel row stride; defined with *"Row stride"*, *"Layer data width"* and *"Alpha blending mode"* parameters. Please refer to Table 5.1.
- Double/triple buffer offset; defined with *"Double buffer address offset"* parameter represented in number of lines. This is only relevant if external frame buffer switching is used.
- Vertical resolution; defined with resolution register. This is only relevant if CPU frame buffer switching is used.

While layer starting address and vertical resolution are directly defined with registers, buffer offset and pixel row stride must be calculated.

Pixel row stride is defined in Table 5.1 and explained in more detail in chapter 5.1 Memory Layout.

Layer double/triple buffer starting address, which is used with external frame buffer switching, can be calculated by multiplying *"Double buffer address offset"* parameter with pixel row stride and then adding layer starting address.

Applying the above gives us the following equation:

$$Layer_x Buffer_0 addr = Layer_x addr$$

$$Layer_x Buffer_1 addr = Layer_x addr + (DoubleBufferAddressOffset \times PixelRowStride_x)$$

$$Layer_x Buffer_2 addr = Layer_x addr + (2 \times DoubleBufferAddressOffset \times PixelRowStride_x)$$

For more information on frame buffer switching, please refer to chapter 5.4 Frame Buffer Synchronization.

For example, let us assume we have the following logiCVC-ML configuration and that we need to use external frame buffer switching:

Number of layers = 3,

Row stride = 1024,

Layer 0 memory address = 0x10000000,

Layer 0 data width = 8,

Layer 0 alpha blending mode = CLUT,

Layer 0 double buffer address offset = 1024,

Layer 1 memory address = 0x10300000,

Layer 1 data width = 16,
Layer 1 alpha blending mode = Layer,
Layer 1 double buffer address offset = 1024
Layer 2 memory address = 0x10900000,
Layer 2 data width = 24,
Layer 2 alpha blending mode = Pixel,
Layer 2 double buffer address offset = 1024.

Applying the calculations for buffer offsets provides us with Table 5.11, which shows a complete memory layout of logiCVC-ML example configuration mentioned above.

Table 5.11: Example layer and buffer addresses

		Start address
Layer 0	**Buffer 0**	0x10000000
	Buffer 1	0x10100000
	Buffer 2	0x10200000
Layer 1	**Buffer 0**	0x10300000
	Buffer 1	0x10500000
	Buffer 2	0x10700000
Layer 2	**Buffer 0**	0x10900000
	Buffer 1	0x10D00000
	Buffer 2	0x11100000

It is important to note that the above example configuration limits the maximum supported resolution to 1024x1024. If higher horizontal resolution is required then pixel row stride needs to be increased to 2048. If higher vertical resolution is required then buffer offset needs to be increased and consequently the following layer addresses need to be adjusted so that layers do not overlap.

In case that external frame buffer switching is not required and we assume the same parameters as explained above, the correct memory layout would be according to Table 5.12.

Table 5.12: Example layer addresses

	Start address
Layer 0	0x10000000
Layer 1	0x10100000
Layer 2	0x10300000

As in the previous example, the resolution is also limited to 1024x1024. However, in this case the maximal vertical resolution can easily be increased by reprogramming the layer address registers to a different address, i.e. to the one that is further away from the previous layer so that layers do not overlap. The higher horizontal resolution support still requires a change in *"Row stride"* parameter.

5.4 Frame Buffer Synchronization

Smooth and artefact-free video display requires careful synchronization of all System on Chip (SoC) parts included in generation and processing of graphics and video contents, such as the display controller, memory controller, video input units, different graphics accelerators, CPU, etc. A flicker-free display requires synchronization between video and graphics input units' refresh rates and the display's refresh rate. These data rates are rarely equal so logiCVC-ML includes logic that makes the synchronization task much easier.

Multiple frame buffers, i.e. video buffers, are well accepted means for synchronization of video and graphics inputs and output data streams. The buffers prevent simultaneous usage of the same data content by separating video and graphics inputs and video outputs. While video input unit streams or graphics unit draws input data in one frame buffer (off-screen video memory buffer), the video output (display controller) unit reads (displays) graphics and video data from the other one (on-screen video memory buffer), which contains previously stored and fully processed data. Buffers are swapped in sync with display vertical blanking periods. Upon swapping, graphics and video inputs start writing (drawing or streaming) data into a frame buffer previously read by the video output, and vice versa.

The video output always displays fully prepared graphics or video frame buffers. It avoids display of other frames, which are partially updated due to ongoing data input from graphics or video inputs. Separated sets of multiple frame buffers are used for every layer controlled by the logiCVC-ML IP core.

This synchronization mechanism prevents appearance of slow scrolling horizontal line on the display, otherwise caused by an improper synchronization. The downside of this synchronization mechanism is an inserted reproduction latency of one or two frame periods. However, this is not an issue for a vast number of applications.

Besides multiple frame buffer synchronization methods, logiCVC-ML supports graphics synchronization on vertical blanking sync signal and without multiple frame buffers. This synchronization method works for graphics input units and cannot be used for synchronization of input video with the video output. This type of synchronization is referred to as the vertical sync synchronization.

The following chapters describe each synchronization method in more detail.

5.4.1 CPU synchronization

This method of frame buffer synchronization usually includes two buffers but it can be used also with multiple frame buffers.

The actual switching of frame buffers is controlled by programming logiCVC-ML layer address registers to point to the required frame buffer in video memory. For more information on layer address register, please refer to chapter 10.2.15 Layer Memory Address Register - Lx_ADDR.

CPU controlled synchronization method works well for synchronization of graphics input units, such as the CPU like the Xilinx MicroBlaze™ or ARM® Cortex™-A9, Xylon's logiBITBLT Bit Block Transfer 2D Graphics Accelerator and logiBMP Bitmap 2.5D Graphics Accelerator, with the logiCVC-ML video output.

The buffering requires use of two separated frame buffers (buffer_0 and buffer_1 implemented in the video memory). In cases where logiCVC-ML uses more than one layer, two frame buffers must be setup for every layer.

The logiCVC-ML can read graphics data from the buffer_1 while graphic input unit draws new graphics content in the buffer_0. In this case, the buffer_1 presents the on-screen memory, while the buffer_0 presents the off-screen memory. Buffers swapping must take place during the display's vertical sync (blanking period). This sync signal corresponds to a finished data reading from the buffer_1. The logiCVC-ML signals the swapping action to the graphics input units which must stop drawing actions in buffer_0, which becomes the on-screen memory read during the next display frame. The same swapping action happens at every vertical sync signal, and the buffer_1 and the buffer_0 continuously change roles.

This method, also known as page flipping, works well only with graphics input units that can stall their operation, i.e. the CPU can stop drawing new graphics in the frame buffer and wait for some time before it starts drawing again.

The double buffering cannot be used with video input units unless their input frame rate is equal to the frame rate of the video output. Any difference between the input and the output frame rates would result with unwanted artefacts visible on the display.

5.4.2 External (video input) synchronization

SoC systems implementing video input units, such as Xylon's logiWIN Versatile Video Input and logiVIEW Perspective Transformation and Lens Correction Image Processor require implementation of more complex triple buffering method. Video inputs' frame rates and video outputs' frame rates are rarely equal, and the SoC must cope with frame rate conversions from lower to higher frame rate, or vice versa.

External video synchronization requires three separate frame buffers (buffer_0, buffer_1 and buffer_2 implemented in the video memory). In SoC designs with logiCVC-ML using more than one layer, three frame buffers must be setup for every logiCVC-ML layer. An additional frame buffer in triple buffering method provides an advantage over the double buffering synchronization method, since the video input units do not have to wait on buffers swapping as they always have a spare frame buffer for writing new frame data.

To support this feature, logiCVC-ML uses video input synchronization control port which consists of two input signals E_CURRENT_VBUFF[C_NUM_OF_LAYERS*2-1:0] and E_SWITCH_VBUFF[C_NUM_OF_LAYERS-1:0] and two output signals E_NEXT_VBUFF [C_NUM_OF_LAYERS*2-1:0] and E_SWITCH_GRANT[C_NUM_OF_LAYERS-1:0].

With the input signals (E_CURRENT_VBUFF[n*2+1:n*2] and E_SWITCH_VBUFF[n]) external video source signals to logiCVC-ML layer n on which buffer it is currently writing data and when to switch buffers (typically on the end of its active frame of external video source). With output signal E_SWITCH_GRANT[C_NUM_OF_LAYERS-1:0] logiCVC-ML grants a switch to the video source to start writing its next frame to E_NEXT_VBUFF[n*2+1:n*2].

logiCVC-ML is constantly sampling E_CURRENT_VBUFF and E_SWITCH_VBUFF inputs with memory clock. When E_SWITCH_VBUFF high state is detected, logiCVC samples E_CURRENT_VBUFF and asserts E_SWITCH_GRANT along with the associated E_NEXT_VBUFF. External logic should constantly sample E_SWITCH_GRANT signal, and when it detects that E_SWITCH_GRANT is high, it should sample E_NEXT_VBUFF and de-assert E_SWITCH_VBUFF. When logiCVC detects E_SWITCH_VBUFF low, it de-asserts E_SWITCH_GRANT signal on the next memory clock cycle. E_SWITCH_VBUFF and E_SWITCH_GRANT signals are used as handshake signals between logiCVC and external logic. This kind of implementation supports switching of buffers between logiCVC and external logic running on synchronous and on asynchronous clocks.

To enable external frame buffer synchronization for particular layer, user has to enable it by setting the EN_EXT_VBUFF_SW bit to 1 in the corresponding layer control register. Please refer to chapter 10.2.19 Layer Control Register - Lx_CTRL.

If external video input signals are connected to the logiCVC-ML's video input synchronization control port and synchronization are turned off (EN_EXT_VBUFF_SW=0), logiCVC-ML will always signal the external video input to write data to buffer 0, i.e. E_NEXT_VBUFF[n*2+1:n*2]=0. At the same time, logiCVC-ML will work in the CPU synchronization mode so it will read memory buffer, which is defined with layer address register.

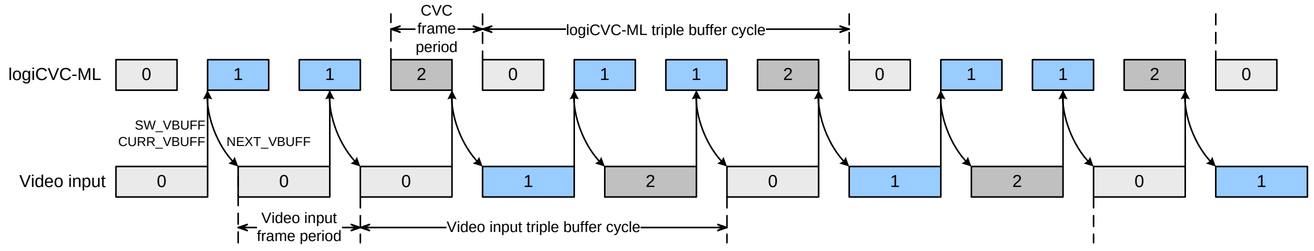


Figure 5.2: Triple buffering example when logiCVC-ML refresh rate is higher than video input

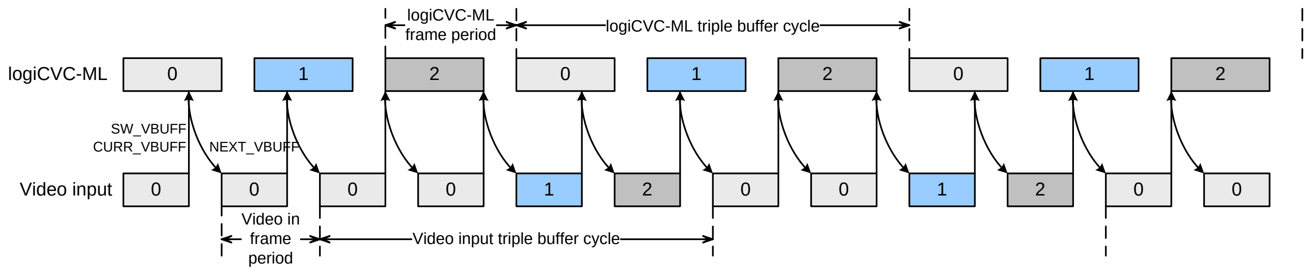


Figure 5.3: Triple buffering example when logiCVC-ML refresh rate is lower than video input

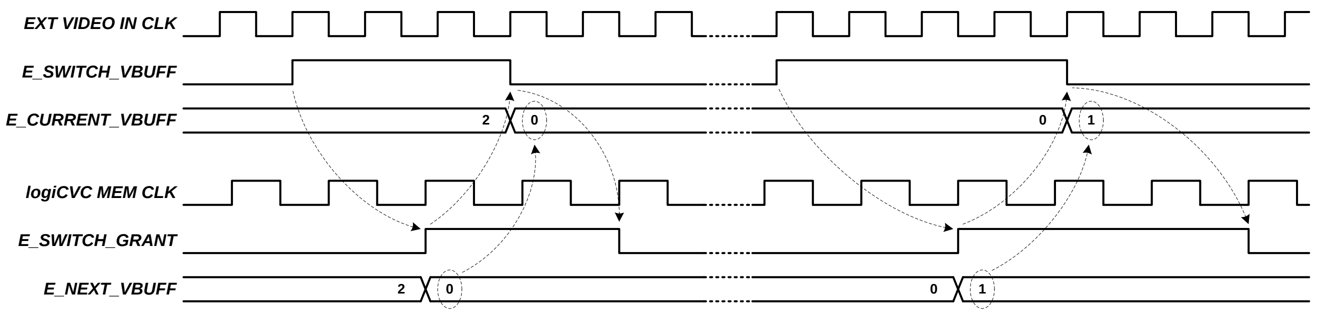


Figure 5.4: External buffer control signals timing diagram

5.4.3 Vertical sync synchronization

Generation of graphics objects (drawing) by a system CPU or graphics accelerators can be synchronized with a display's refresh rate without multiple frame buffers which was described in previous chapters. SoCs utilizing this type of synchronization require a single frame buffer per logiCVC-ML layer. This frame buffer continually operates as the on-screen memory, since there is no swapping between the on-screen and the off-screen video memory.

Graphics input units must monitor display's vertical blanking sync signal (interrupt signal or interrupt status register) and draw graphics data in the frame buffer only during this blanking period. Duration of the vertical blanking period is relatively short in comparison to display's refresh time, but allows for a smooth graphics and some level of animation.

This synchronization can also be combined with any of the two multiple frame buffer synchronization methods in more complex SoC designs.

5.5 Memory bandwidth requirements

Required memory bandwidth greatly depends on the logiCVC-ML configuration parameters described in chapter 3 CUSTOMIZATION, display resolution and video clock frequency.

The actual required bandwidth can be calculated with two formulas, one describing the maximum required memory bandwidth and the other describing the average required memory bandwidth.

Maximum memory bandwidth requirement can be calculated by the following equation:

Equation 5.1: Required maximum memory bandwidth

$$RMBW_{\max} = (layer_0 bpp + layer_1 bpp + \cdots + layer_n bpp) \times VCLK\ \ freq$$

where:

$RMBW_{max}$: required memory bandwidth, maximum

$layer_n bpp$: layer n bits per pixel configuration which depends on *"Layer data width"* and *"Alpha blending mode"* parameters. Please refer to Table 5.13 for exact values.

Table 5.13: $layer_n$bpp values used for memory bandwidth calculation

		Data width (bit)		
		8	16	24
Alpha blending mode	Layer alpha	8	16	32
	Pixel alpha	16	32	32
	CLUT	8	-	-

Average memory bandwidth requirement is calculated by the following equation:

Equation 5.2: Required average layer memory bandwidth

$$RMBW_{avg} = ((bpp \times width \times height)_0 + \cdots + (bpp \times width \times height)_n) \times refresh\, rate$$

where:

$RMBW_{avg}$: required memory bandwidth, average

bpp: layer bits per pixel configuration outlined in Table 5.13

width: layer horizontal size in pixels defined by layer size registers described in chapter 10.2.17

height: layer vertical size in lines defined by layer size registers described in chapter 10.2.17

n: number of logiCVC-ML layers defined by parameter C_NUM_OF_LAYERS.

refresh rate: display refresh rate which is defined by VCLK frequency, display resolution and porches.

If logiCVC-ML has a maximum bandwidth value at its disposal, it will function properly in any design configuration independent of other peripherals in the system. If logiCVC-ML has an average bandwidth value at its disposal, a stable functioning of logiCVC-ML depends on other peripherals in the system requesting memory, and it is not guaranteed.

It is very important to note that the values calculated with these equations represent the required efficient memory bandwidth in order for logiCVC-ML to show a stable picture. In other words, the equations do not take into account the efficiency of the memory interface bus.

The efficiency depends on the memory interface (AXI4), memory controller in the design, off chip memory devices and memory burst length (C_MEM_BURST). For example, the bigger the burst size, i.e. number of transfers per burst, the higher the efficiency.

The logiCVC-ML interrupt status register includes one status bit (FIFO_UNDERRUN) that signals insufficient memory bandwidth. Please refer to chapter 10.2.9 Interrupt Status Register - INT_STAT for more information.

Example 1

Let us assume we have the following logiCVC-ML configuration:

Number of layers = 2,

Layer 0 data width = 16, *Layer 0 alpha blending mode* = Pixel,

Layer 1 data width = 24, *Layer 1 alpha blending mode* = Layer,

resolution = 1024x768 @60Hz (XGA),

pixel clock = 65MHz,

layer sizes = resolution.

According to the above equations the required maximum memory bandwidth is

$$RMBW_{\max} = (32 + 32) \times 65MHz = 4160000000b/s \cong 496MB/s$$

while the required average memory bandwidth is

$$RMBW_{avg} = ((32 \times 1024 \times 768) + (32 \times 1024 \times 768)) \times 60 \cong 360MB/s$$

Example 2

Let us assume we have the following logiCVC-ML configuration:

Number of layers = 5,

Layer 0 data width = 8, *Layer 0 alpha blending mode* = CLUT,

Layer 1 data width = 16, *Layer 1 alpha blending mode* = Pixel,

Layer 2 data width = 16, *Layer 2 alpha blending mode* = Layer,

Layer 3 data width = 24, *Layer 3 alpha blending mode* = Layer,

Layer 4 data width = 24, *Layer 4 alpha blending mode* = Layer,

Configure last layer as background = true

resolution = 854x480 @60Hz (WVGA),

pixel clock = 28MHz,

layer size$_0$ = 400x240,

layer size$_1$ = 512x240,

layer size$_2$= resolution,

layer size$_3$ = resolution.

According to the above equations the required maximum memory bandwidth is

$$RMBW_{\max} = (8 + 32 + 16 + 32 + 0) \times 28MHz = 2464000000b/s \cong 294MB/s$$

while the required average memory bandwidth is

$$RMBW_{avg} = ((8 \times 400 \times 240) + (32 \times 512 \times 240) + (16 \times 854 \times 480) + (32 \times 854 \times 480)) \times 60 \cong 175MB/s$$

As can be seen in this example, layer 4 is not influencing memory bandwidth requirements because logiCVC-ML is configured to use last layer as background layer, which means data for this layer is not fetched from memory but is read from internal logiCVC-ML register. Please refer to chapter 10.2.7 Background Color Register - BACKGROUND for more information.

6 CPU INTERFACE

6.1 Register Interface

The logiCVC-ML registers can be accessed AXI4-Lite slave interface. Please consult Xilinx AXI4-Lite specification regarding the functionality of this interface.

By default, all registers inside logiCVC-ML can be read and write, except IP version register, which can only be read. However some registers have different write and read values and the reading functionality can be switched off by disabling *"Readable logiCVC registers"* parameter. Variable register widths are used; 8, 16, 24 or 32-bit, however, logiCVC-ML only supports 32-bit write accesses into its registers. Non-used bits inside each register are reserved and read '0'.

Register memory map and description is given in chapter 10 REGISTERS.

S_AXI clock is independent of other clocks connected to logiCVC-ML, i.e. they do not need to be in any relationship with other logiCVC-ML clock signals.

6.2 Interrupt Interface

logiCVC-ML interrupt output is configured as active high, level sensitive. It is used for special handling of several logiCVC-ML features:

- Layer registers update (address, size and position)
- V-Sync
- External video input
- FIFO underrun status
- CLUT select (double CLUT)

Layer registers update interrupt signals that logiCVC-ML has loaded a new set of values in the layer memory address, size and position registers. As these three set of registers usually need to be updated together, this mechanism allows the user to modify all required registers and by a final write to layer address registers, requests the loading of all updated values at the same time. When logiCVC-ML does this in vertical inactive period, it will raise register update interrupt status bit to inform the CPU of the update.

Each layer that has CLUT alpha blending enabled has double CLUT. Switching of CLUTs is controlled by CLUT_SEL register. At the beginning of a new frame, CLUT is switched and this is signaled to CPU by setting the corresponding bit in the INT_STAT register. This is how logiCVC-ML signals to the CPU that it has switched the active CLUT and that new contents can be written in the inactive CLUT. In this way, flicker-free reproduction is guaranteed. For more on double CLUT refer to chapter 10.2.8 CLUT Select Register - CLUT_SEL.

One bit in the INT_STAT register is used for signaling to the CPU that a new frame is starting. This can be used as feedback to the CPU for applications where user wants to change some logiCVC-ML parameters once per frame. Example for this would be adjusting layer position, size or memory offset once per frame.

When an external video input stream is used to drive one logiCVC-ML layer, one additional bit in the INT_STAT register is used. In that case one bit is used to signal to the CPU that external video source is present on the input.

FIFO underrun interrupt status bit is set if logiCVC-ML FIFOs are not sufficiently filled with new memory data. It usually occurs when there is not enough memory bandwidth available for logiCVC-ML to properly refresh the display, i.e. the input read data rate is lower than the required output data rate.

All of the above mentioned interrupt sources can be masked using INT_MASK register.

For more information on interrupt handling and interrupt status and mask registers, please refer to chapters 10.2.10 Interrupt Mask Register - INT_MASK and 10.2.9 Interrupt Status Register - INT.

7 DISPLAY INTERFACE

The logiCVC-ML video controller supports five different types of display interfaces:

- Parallel pixel data interface including three control signals and clock
  - RGB output
  - YCbCr output (4:4:4 or 4:2:2)
- LVDS interface
  - LVDS interface, four data and one clock pair
  - LVDS interface, three data and one clock pair
  - Camera link interface, four data and one clock pair
- Video interface according to the ITU656 standard; 8-bit data bus and clock signal
- DVI interface, three data and one clock pair
- AXI4-Stream video interface

7.1 Parallel Pixel Data Interface

The parallel pixel data interface is controlled by the following logiCVC-ML signals:

- HSYNC is a horizontal synchronous signal and is active synchronously with each pixel row start.
- VSYNC is a vertical synchronous signal and is active synchronously with every first pixel line occurrence.
- BLANK is a data valid signal and is active when valid pixel data is present on the pixel data bus.

The TFT and CRT video displays require some blank time between the last pixel of the preceding and the first pixel of a succeeding video picture line. Blank time is also required between the last pixel of a preceding frame and the first pixel of succeeding frame. The blank time is controlled by the BLANK signal, which is active during the valid pixels on the data bus.

PIX_CLK is the main display clock and PIX_CLKN is the inverted version for use on some displays/LCDs that use differential clock input.

The display manufacturers use different naming conventions for display control signals like:

- HSYNC i.e. LP, CL1
- VSYNC i.e. FLM, S
- BLANK i.e. DE
- PIX_CLK, SHCLK, CL2

The pixel data is outputted on the D_PIX data bus.

Parallel interface can be used even when other display interfaces are selected however when *"Display interface type"* parameter is set to *Parallel*, only the parallel interface is active.

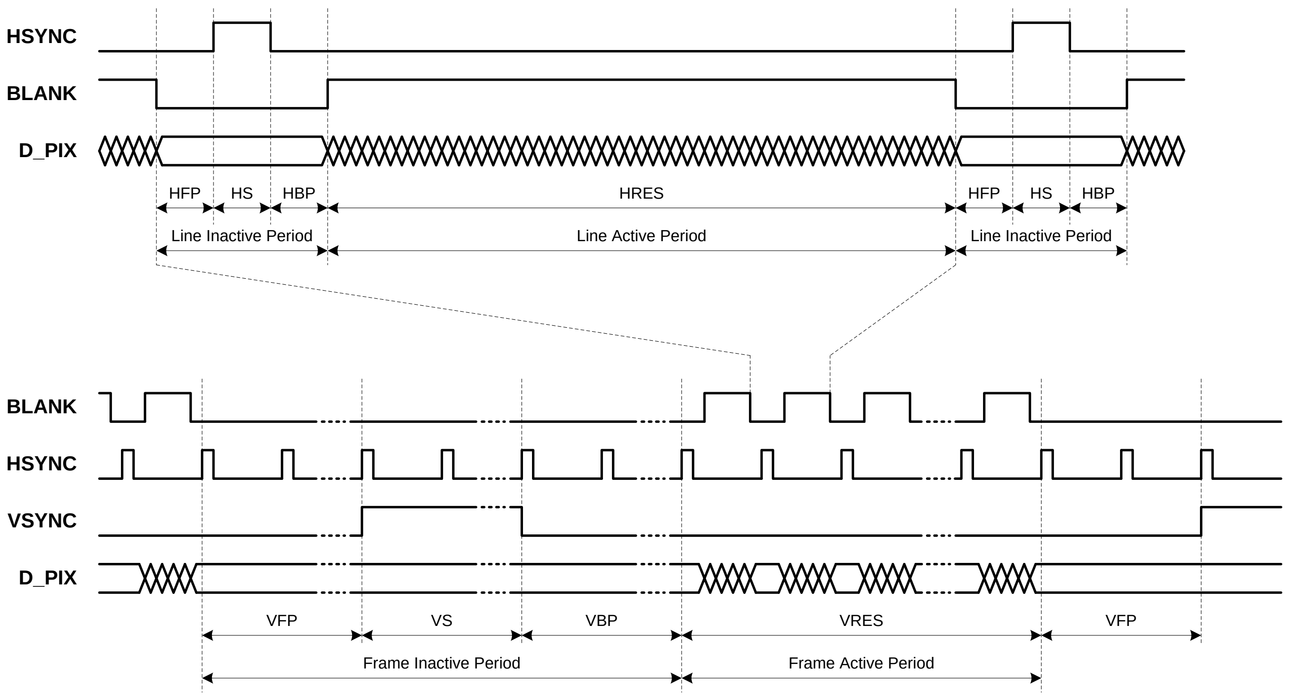


Figure 7.1: Parallel pixel data interface

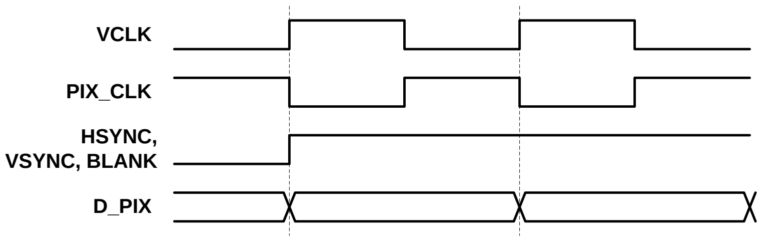


Figure 7.2: Parallel pixel data interface clock alignment

1. To activate the pixel data interface user must enable the interface through the Power control register. For more information, refer to chapter 10.2.11 Power Control Register - PWRCTRL.
2. The PIX_CLK active edge and polarity of the control signals depend on the settings in the CTRL register. The above figures represent default settings (falling active clock edge, high active control signals).

7.1.1 RGB interface

When logiCVC-ML is configured to use RGB color format on its output (*Display interface color format* parameter) the pixel data width can be adjusted to different widths (15, 16, 18, 24, 12 DDR and 30) by using the *"Pixel data width"* parameter. This way the data width on the output of the logiCVC-ML can be, independently of the selected layer widths, tailored to the requirements of the connected display.

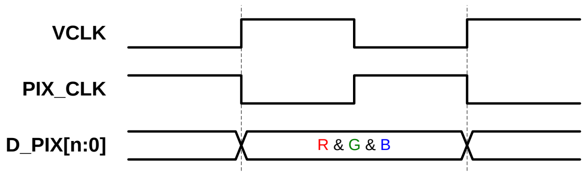


Figure 7.3: RGB data interface

7.1.2 YCbCr interface

YCbCr is a color space that contains three components, Y as a luma component and Cb and Cr as blue-difference and red-difference chroma components. If logiCVC-ML is configured to use YCbCr parallel output interface and some layers are not in YCbCr color format, then their RGB pixel data is converted from RGB to YCbCr color space using the formulas described in chapter 9.2.5 Color space converter.

When logiCVC-ML is configured to use YCbCr color format on its output (*Display interface color format* parameter) the pixel data width can be adjusted to different widths (16, 20, 24, 12 DDR and 30) by using the *"Pixel data width"* parameter. This way the data width on the output of the logiCVC-ML can be, independently of the selected layer widths, tailored to the requirements of the connected display.

logiCVC-ML supports two sampling formats, 4:4:4 and 4:2:2, outlined in Figure 7.4.

When using 4:4:4, the data width can be set to 24, 12 DDR or 30 bits wide which means each component can be 8 or 10 bits wide. In 4:4:4 format each component, Y, Cb and Cr, will appear on D_PIX output on every active clock edge.

When using 4:2:2, the data width can be set to 16 or 20 bits wide which means each component can be 8 or 10 bits wide. In 4:2:2 format the higher part of the bus is used for Y component and the lower part is used to alternately send out Cb and Cr components. On each active clock edge one component is sent out.

Figure 7.4 outlines the order of pixel data on D_PIX bus depending on the chosen sampling mode.

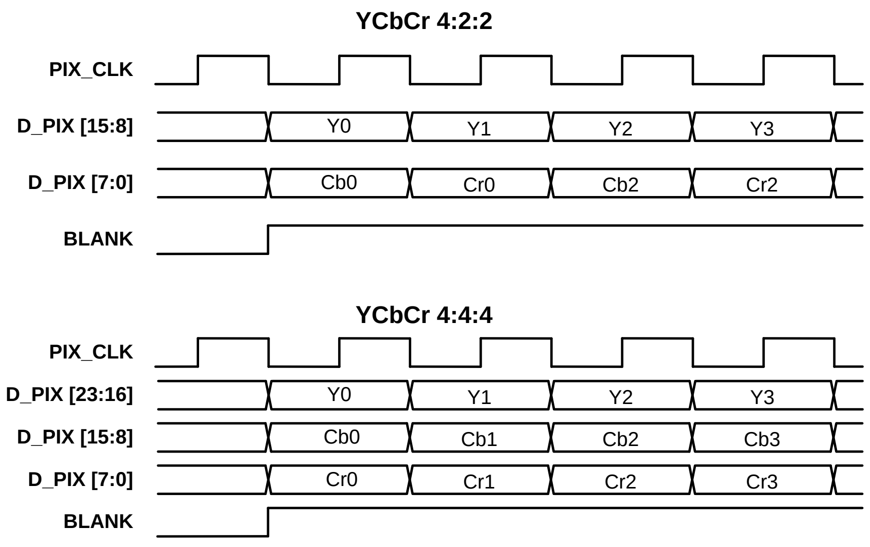


Figure 7.4: Pixel data order on YCbCr 4:4:4 and YCbCr 4:2:2 data output bus

1. To activate the pixel data interface user must enable the interface through the Power control register. For more information, refer to chapter 10.2.11 Power Control Register - PWRCTRL.
2. The PIX_CLK active edge and polarity of the control signals for both modes depend on the settings in the CTRL register. The above figures represent default settings (falling active edge).
3. In case of 20 bit YCbCr 4:2:2, D_PIX [19:10] is used for Y and D_PIX [9:0] is used for Cb and Cr.
4. In case of 30 bit YCbCr 4:4:4, D_PIX [29:20] is used for Y, D_PIX [19:10] is used for Cb and D_PIX [9:0] is used for Cr component.

7.1.3 Parallel pixel processing

The width of the D_PIX data bus is defined by *"Pixel data width"* and *"Pixels per clock"* parameters. The *"Pixel data width"* parameter defines the size of one pixel while *"Pixels per clock"* parameter defines how many of these pixels are present on the D_PIX bus on each active clock edge. The *"Pixels per clock"* parameter allows sending out one, two or four pixels per clock which allows increasing the pixel throughput while maintaining the same pixel clock frequency. This is especially useful when using relatively high resolutions and frame rates which require pixel clock frequencies that are too high for FPGA implementation, e.g. 4k2k@60fps.

To support this feature, logiCVC-ML uses parallel pixel processing which means all pixel processing logic (alpha blending, color space conversion) will be doubled or quadrupled depending on the *"Pixels per clock"* parameter.

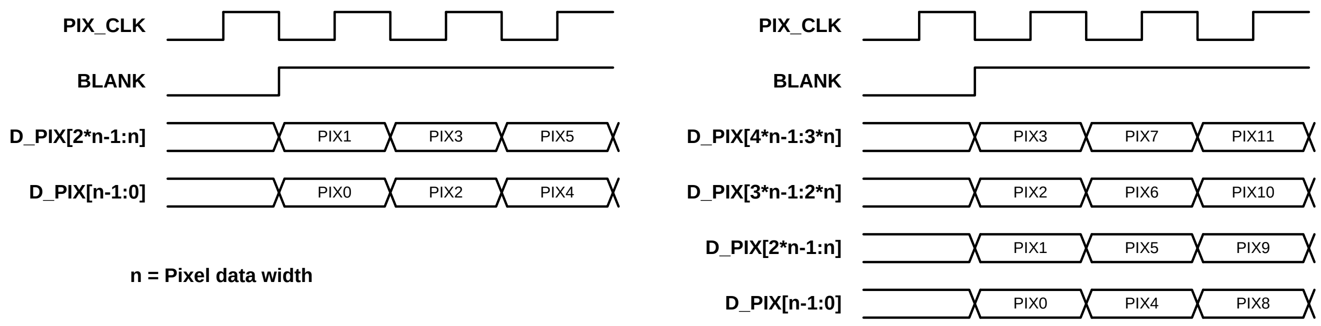


Figure 7.5: Parallel pixel data bus with two and four pixels per clock

7.1.4 12 bit DDR interface

When using 12bit DDR interface (*"Pixel data width"* parameter), data is outputted on both edges of PIX_CLK clock effectively producing a 24-bit interface but reducing the number of signals to 12.

By default, the data is outputted on both edges of the clock as outlined on the right side of Figure 7.6. However, logiCVC-ML supports outputting data in such a way that clock edge is aligned with the middle of the data bus eye as outlined on the left side of Figure 7.6. In this case, user has to supply VCLK2 input clock that has twice the frequency and is synchronous to the VCLK input. This mode is enabled by using the *"Use vclk2 for DDR pixel data output"* parameter.

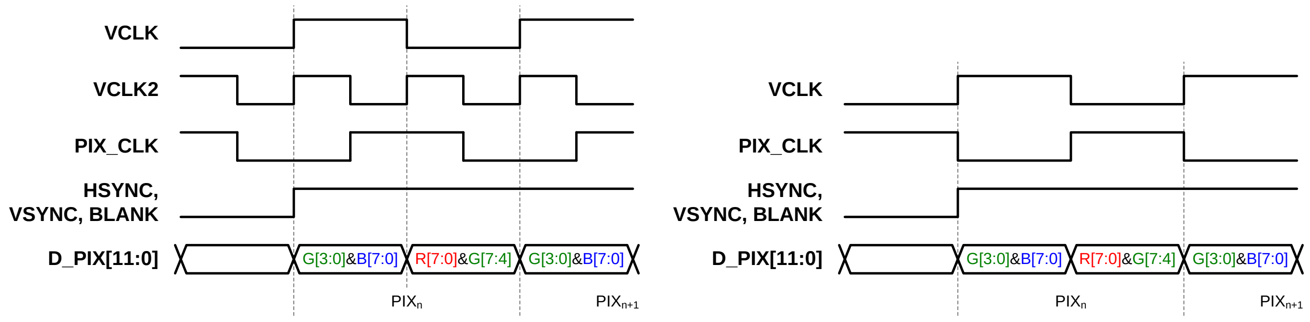


Figure 7.6: 12-bit DDR data interface with and without using VCLK2

7.1.5 Embedded synchronization signals

The logiCVC-ML supports transmission of synchronization signals embedded within the video stream. This option can be enabled using the *"Use embedded syncs"* parameter.

Figure 7.7 illustrates parallel interface data format when the embedded syncs are enabled. Instead of transmitting timing signals (HSYNC, VSYNC and BLANK), the special synchronization packets (also known as timing reference signals or TRS) are sent embedded in the stream. TRS defines active video and blanking intervals. Each TRS consists of four 8-bit words, the first three words are always 0xFF, 0x00 and 0x00, and the fourth word contains flag and protection bits. The flag bits that can be found in fourth synchronization word are known as F, H and V flags. The H bit indicates the start of horizontal blanking interval, and the fourth synchronization word in packet has H flag set to one. Such TRS define End of Active Video (EAV). The stream of active data words starts immediately after synchronization word with H set to 0. Such TRS determinates the Start of Active Video (SAV). The V bit defines start of vertical blanking period, the EAV packet with V flag set to 1 defines vertical interval. EAV packet with V bit set to 0 determinates that the line is part of the active picture. The F bit that indicates even/odd line field is always set to zero.

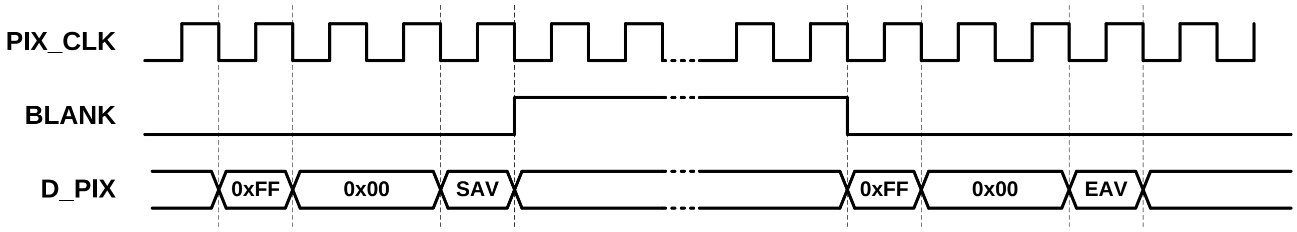


Figure 7.7: Parallel pixel data interface with embedded timing signals

Table 7.1 lists four possible combinations of status words that are sent in SAV and EAV packets. P_{0-3} are protection bits that enable detection and correction of single bit errors at the receiver side.

Table 7.1: Status Word in EAV and SAV packets

	D7 MSB	D6 F	D5 V	D4 H	D3 P3	D2 P2	D1 P1	D0 P0
Status word	1	0	0	0	0	0	0	0
	1	0	1	0	1	0	1	1
	1	0	0	1	1	1	0	1
	1	0	1	1	0	1	1	0

Video data should use any 8-bit value (R, G, B or Y, Cb, Cr components in pixel) in the range 1 to 254 (0x01 to 0xFE), the values 0x00 and 0xFF are reserved.

7.1.6 CRT Displays

Because CRT displays require analogue input signals between 0 – 0.7 Vpp and logiCVC-ML outputs are digital data according to FPGA pin specifications; an FPGA external Video DAC must be used. The video DAC converts the digital data to analogue signals.

logiCVC-ML was tested with multiple Video DACs including CH7301C and ADV7123.

7.2 LVDS Interface

The logiCVC-ML supports two different types of LVDS interfaces which are named standard LVDS and Camera Link interface. Both use the same output signals, lvds_data_out and lvds_clk_out, however they differ in the number of data lines and the way the pixel data is spread out onto the serialized data pairs. The following chapters outline the differences between the two.

To activate the LVDS interface signals, user must enable it through the Power control register. For more information, refer to chapter 10.2.11 Power Control Register - PWRCTRL.

Regardless of the LVDS interface selection, the logiCVC-ML still drives the parallel pixel interface, PIX_CLK, HSYNC, VSYNC, BLANK and D_PIX if it is enabled in the Power Control register.

7.2.1 Standard LVDS Interface

To use the LVDS interface, parameter C_DISPLAY_INTERFACE has to be set to either 2 or 4 (LVDS 4bit or LVDS 3bit). Additionally, user must supply one or two additional LVDS input clocks to the LVDS_CLK and LVDS_CLKN input ports. Please refer to chapter 4 on clocking the logiCVC-ML in case of LVDS output interface.

LVDS interface uses four or five LVDS pairs, three or four for the data and one for the clock. To drive 24-bit video data and three control signals, 27 bits in total, over 4 LVDS pairs, LVDS coder uses faster LVDS_CLK clocks. In this way, seven signals are transmitted over one LVDS pair. Figure 7.8 shows the order of data and control bits on four LVDS data pairs.

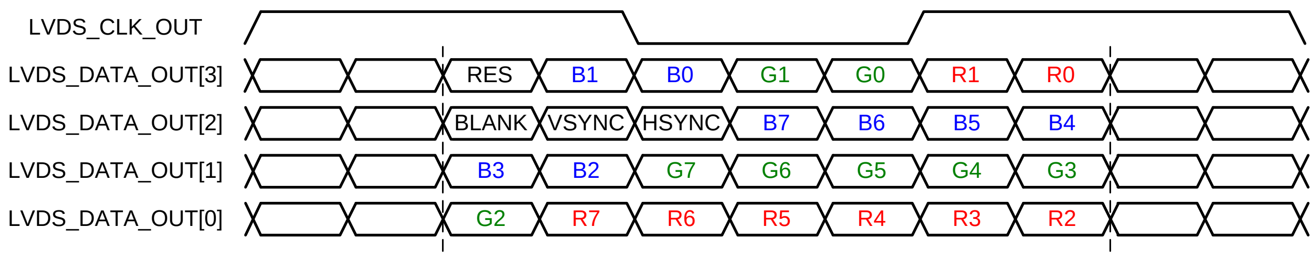


Figure 7.8: Pixel data order on LVDS data bus

In the case that only three data pairs are required, i.e. 18 bpp, user can set the C_DISPLAY_INTERFACE parameter to 4 (LVDS 3bit) in which case the highest data pair LVDS_DATA_OUT[3] carrying least significant bits will not be used.

7.2.2 Camera Link Interface

The only difference between LVDS 4bit interface and camera link interface is in pixel data order on the LVDS bus. Both interfaces use the same signals, output ports and clocks.

Figure 7.9 shows the order of data and control bits on four Camera link LVDS data pairs.

Camera link in the form of 3bit data interface is the same as LVDS 3bit interface. To use camera link interface, parameter C_DISPLAY_INTERFACE has to be set to 3.

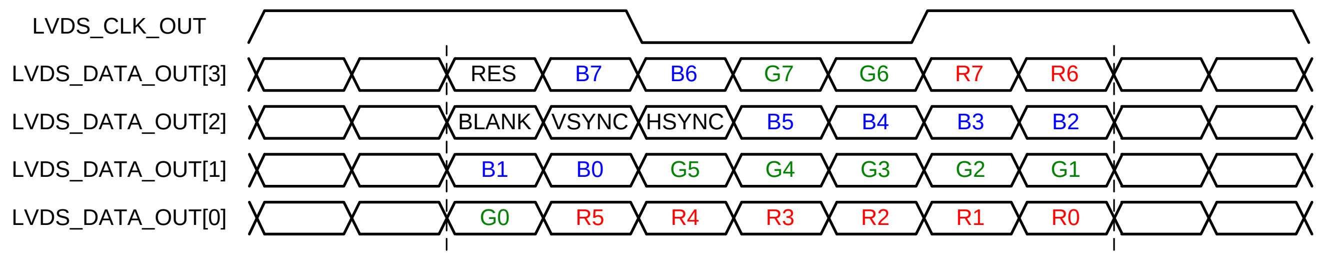


Figure 7.9: Pixel data order on Camera link LVDS data bus

7.3 ITU656 Interface

The logiCVC-ML is capable of driving both ITU-656 video standards, NTSC and PAL on the ITU656_DATA output bus.

The NTSC or PAL standard is selected runtime using the DTYPE register, please refer to chapter 10.2.6 Display Type Register - DTYPE. The logiCVC-ML generates control and data signals on the ITU656_DATA bus according to ITU656 standard and the width of the bus can be configured to 8 or 10 bit using the *"ITU656 data width"* parameter.

In order to use this interface, user must provide two clocks to logiCVC-ML, ITU_CLK_IN and VCLK. According to ITU656 standard, the ITU_CLK_IN clock has to be 27 MHz and accordingly the VCLK has to be synchronous to ITU_CLK_IN and have half the frequency, i.e. 13.5 MHz.

For most applications that require analogue ITU656, i.e. composite video or S-Video output, the digital video encoder shall be connected to logiCVC-ML FPGA output pins. The logiCVC-ML is tested with Philips SAA7121 and Analog Devices ADV7393 digital video encoders.

When using ITU656 interface, user has some logiCVC-ML registers including resolution and sync registers cannot be changed because they are predefined with the ITU656 standard.

logiCVC-ML supports two modes of reading data from memory when ITU656 interface is used, interlaced and progressive. In interlaced mode logiCVC-ML reads only odd lines (1, 3, 5, …) in one frame and only even lines (0, 2, 4, …) in the next. In the progressive mode, the same lines are read from memory in every frame (0, 1, 2, …). The output resolution for both modes is always according to the ITU656 standard (PAL or NTSC) and is independent of the mode selected, but in interlaced mode active video buffer is twice the size than in progressive mode. This feature can be controlled for each logiCVC-ML layer separately through one bit in the corresponding Layer Control register. For more information, please refer to chapter 10.2.19 Layer Control Register - Lx_CTRL.

Regardless of the ITU656 selection, the logiCVC-ML still drives the parallel pixel interface, PIX_CLK, HSYNC, VSYNC, BLANK and D_PIX, if Power Control register is configured correctly. For more information, refer to chapter 10.2.11 Power Control Register - PWRCTRL.

7.4 DVI Interface

The logiCVC-ML supports Digital Visual Interface (DVI) output by directly driving the DVI monitor from the FPGA. When selecting DVI output by setting the *"Display interface type"* parameter to *DVI*, user is enabling three data and one clock LVDS pair outputs from logiCVC-ML.

The maximum DVI resolution that the logiCVC-ML supports is constrained by the maximum toggle rate of TMDS IOs and clock lines in the selected FPGA family/device. The following chapter describes the maximum performance of the DVI interface depending on the configuration.

Please note that additional external protection diodes (ESD) need to be used in order to satisfy DVI compliance tests for contact and air discharge protection. Please consult the Xilinx device documentation and DVI specification for further information.

Regardless of DVI output selection, the logiCVC-ML still drives the parallel pixel interface, PIX_CLK, HSYNC, VSYNC, BLANK and D_PIX, if Power Control register is configured correctly. For more information, refer to chapter 10.2.11 Power Control Register - PWRCTRL.

7.4.1 DVI transmitter clocking

The logiCVC-ML supports two different clocking solutions when implementing DVI display interface in 7 Series devices. Maximum resolution supported is determined by clocking mode defined with *"DVI transmitter clocking mode"* parameter.

Setting *"DVI transmitter clocking mode"* to "*MMCM+BUFIO+2xBUFR"*, DVI transmitter uses a MMCM, BUFIO and two BUFR components in order to generate the required clock infrastructure.

The advantage of this mode is that it supports maximum data rates and uses only regional clock buffers (BUFIO, BUFR). Disadvantage is that because it uses regional buffers, DVI logic and pins must be located in the same clock region.

Setting *"DVI transmitter clocking mode"* to *"PLL+3xBUFG"*, DVI transmitter uses a PLL and three BUFG components in order to generate the required clock infrastructure.

The advantage of this mode is that DVI logic and pins can be located across different clock regions. Disadvantage is that it supports a lower maximum data rate and it consumes three global clock buffers (BUFG).

Table 7.2 and Table 7.3 outline the maximum performance and supported clock rates for both clock implementation modes. We strongly recommend that users regularly check the latest Xilinx documentation for possible timing changes mentioned in the following tables.

Table 7.2: 7 Series performance

Speed Grade	**Mb/s**	
	MMCM+BUFIO+2xBUFR	PLL+3xBUFG
-3	1250	1250
-2/-2L/-2G	1250	1250
-1	1250	928

Table 7.3: Common video resolutions supported

Resolution (@60Hz)	Pixel clock (MHz)	Data rate (Mb/s)
VGA (640x480)	25.25	252.5
480p (720x480)	27	270
WVGA (854x480)	32	320
SVGA (800x600)	40	400
XGA (1024x768)	65	650
WXGA (1280x768)	68.25	682.5
HD 720p (1280x720)	74.25	742.5
HD 1080i (1920x1080)	74.25	742.5
SXGA (1280x1024)	108	1080
WSXGA+(1680x1050)	120	1200

As mentioned in the previous section, DVI standard requires TMDS IO standard. 7 Series devices support TMDS IOs only in High Range (HR) IO bank types. Therefore, please consult the required 7 Series documentation for proper pin assignment.

7.5 AXI4-Stream Interface

Even though AXI4-Stream is not intended to be sent towards the display, logiCVC-ML supports outputting pixel data using the AXI4-Stream video interface in cases that logiCVC-ML output needs to be used inside the FPGA as an input to some other image processing blocks.

AXI4-Stream interface is compliant with AXI4-Stream Video Protocol described in Xilinx AXI Reference Guide User Guide, UG761 (v14.3). Please consult the mentioned documentation for more information on signal and protocol description.

The *"AXI4-Stream component width"* parameter defines the size of color component width and can be configured to 8 or 10 bit.

The width of the pixel present on the AXI4-Stream bus is defined by component data width and "*AXI4-Stream video format"* parameter which defines if the pixel is represented in RGB, YCbCr 4:4:4 or YCbCr 4:2:2 color format.

The width of the actual AXI4-Stream data bus, M_AXIS_VIDEO_TDATA, is defined by two previously mentioned parameters; *"AXI4-Stream component width"*, *"AXI4-Stream video format",* and a third parameter *"Pixels per clock"*.

"Pixels per clock" parameter defines how many pixels are present on the output data bus on each active clock edge. The *"Pixels per clock"* parameter allows sending out one, two or four pixels per clock which allows increasing the pixel throughput while maintaining the same pixel clock frequency. This is especially useful when using relatively high resolutions and frame rates which require pixel clock frequencies that are too high for FPGA implementation, e.g. 4k2k@60fps.

To support this feature, logiCVC-ML uses parallel pixel processing which means all pixel processing logic (alpha blending, color space conversion) will be doubled or quadrupled depending on the *"Pixels per clock"* parameter.

7.6 Dithering module

Dithering module is designed for arithmetic calculations on pixels to generate additional levels of color when lower bit display interfaces are used (18bpp). It can be enabled when using18-bit parallel or 3-bit LVDS interface by enabling *"Use dithering"* parameter.

Dithering module's pixel processing includes spatial dithering and temporal dithering (see Figure 7.10). Two least significant bits of each 8-bit color of 24 bit wide pixel-bus are inputs to module. Dependent on their value, each 6-bit color of 18-bit wide output pixel-bus is averaged over four frames in terms of rounding in arithmetic unit. To avoid flickering, not all pixels should be dithered in the same rhythm, therefore spatial dithering is also implemented (see Figure 7.11).

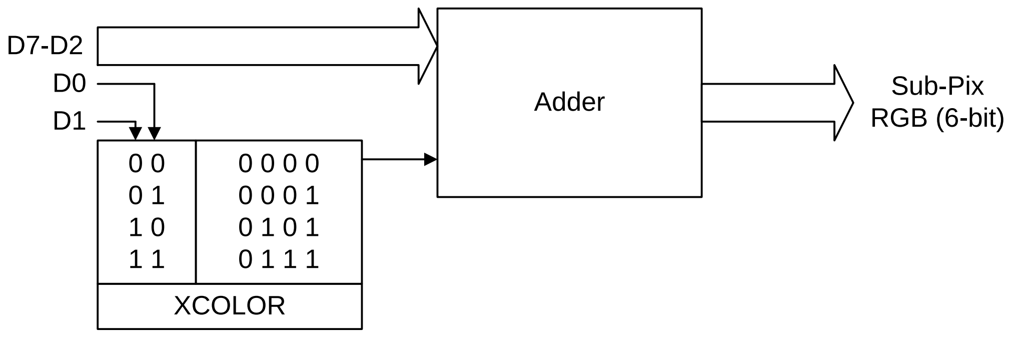


Figure 7.10: Dithering module

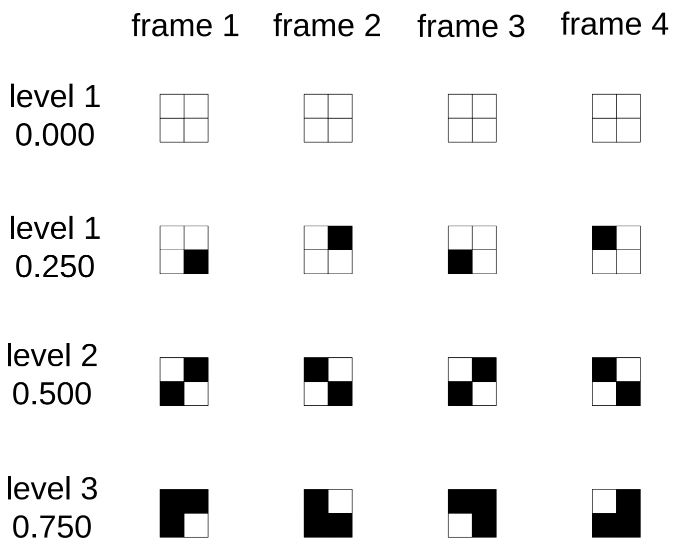


Figure 7.11: Temporal and spatial dithering

8 EXTERNAL VIDEO INPUT INTERFACE

The logiCVC-ML can be configured to use external video input stream data as one of its layers. In this case the selected layers' data is not read from memory frame buffer but is received through this external input.

External video input supports two interface types; parallel and AXI4-Stream video. The following chapters describe both interfaces.

8.1 Parallel Interface

The logiCVC-ML input signals used for parallel input interface are:

- E_VCLK – external clock
- E_VSYNC – external vertical sync
- E_HSYNC – external horizontal sync
- E_BLANK – external blank
- E_DATA[C_E_DATA_WIDTH – 1 : 0] – external data
- E_VIDEO_PRESENT – external parallel input present flag

After E_VIDEO_PRESENT signal goes high, logiCVC-ML starts to measure the parameters of the input stream so that it can configure its state machines and registers accordingly. The parameters that are measured are: horizontal sync, horizontal front and back porch, horizontal resolution, vertical sync, vertical front and back porch and vertical resolution. After the measurements are complete, logiCVC-ML resets its internal logic and starts to send control and data to the display according to the measured parameters. It also signals to the CPU that it has detected and finished measuring the input stream by asserting E_VIDEO_VALID flag in the interrupt status register. If there is no parallel input present, i.e. E_VIDEO_PRESENT is low, logiCVC-ML will switch the logic to work on VCLK input clock and will start sending control signals to the display as written in the logiCVC-ML registers.

When *"Use BUFGMUX for external parallel input"* parameter is not enabled, user has to instance BUFGMUX manually and switch VCLK to E_VCLK when E_VIDEO_PRESENT is high. In this case, the output of the BUFGMUX has to be connected to VCLK port of logiCVC-ML. This is useful when LVDS or camera link interface is used since LVDS_CLK has to be synchronous to E_VCLK when E_VIDEO_PRESENT is high and synchronous to VCLK when E_VIDEO_PRESENT is low.

All input control signals are sampled on rising edge of E_VCLK so user has to make sure to provide them accordingly. Figure 7.1 and Figure 7.2 represent the way logiCVC-ML outputs the control signals by default. In the same way, logiCVC-ML expects them at the external parallel input interface. The polarity of input control signals and E_VIDEO_PRESENT signal can be controlled by setting display CTRL register bits 16 to 19 accordingly.

All porch and sync registers in logiCVC-ML are 10 bit so the maximum values that can be measured are 1024. Resolution registers are limited with *"Row stride"* parameter. As for the minimum value, it is the same for all measurements and is fixed to 1. Therefore, for example, horizontal front porch has to last at least one E_VCLK clock period.

User can use two registers that represent the measured horizontal and vertical resolution to check the input source resolution. Please refer to 10.2.12 External Input Horizontal Resolution - E_HRES and 10.2.13 External Input Vertical Resolution - E_VRES for more information.

Please note that the polarity of control signals depends on the settings in the CTRL register. For more information, refer to chapter 10.2.5 Control Register - CTRL.

8.2 AXI4-Stream Slave Interface

AXI4-Stream interface is compliant with AXI4-Stream Video Protocol described in Xilinx AXI Reference Guide User Guide, UG761 (v14.3). Please consult the mentioned documentation for more information on signal and protocol description.

In order for logiCVC-ML to function properly when using AXI4-Stream input interface, the video resolution of the input stream has to be the same as the configured logiCVC-ML resolution. Additionally, as the logiCVC-ML has to keep its output at a fixed refresh rate that is defined by the resolution and porch registers and pixel clock frequency, the AXI4-Stream input has to be able to provide the data on the logiCVC-ML input at the same refresh rate.

The internal FIFOs on the logiCVC-ML AXI4-Stream slave interface can level out small bottlenecks on the interface but only to a limited amount. If there is need for more buffering on the AXI4-Stream interface, external FIFOs can be implemented between the AXI4-Stream Master and logiCVC-ML AXI4-Stream Slave interface. Xilinx provides such AXI4-Stream FIFO IP through the Vivado IP catalogue.

The AXI4-Stream slave interface uses a dedicated clock signal, S_AXIS_ACLK. The logiCVC-ML is internally using a FIFO for clock crossing to the logiCVC-ML video clock (VCLK) domain. The AXI4-Stream clock can be asynchronous to the VCLK but the data throughput of the interface needs to be similar to the logiCVC-ML output interface.

The *"AXI4-Stream component width"* parameter defines the size of color component width and can be configured to 8 or 10 bit.

The width of the pixel present on the AXI4-Stream bus is defined by component data width and "*AXI4-Stream video format"* parameter which defines if the pixel is represented in RGB, YCbCr 4:4:4 or YCbCr 4:2:2 color format.

The width of the actual AXI4-Stream data bus, S_AXIS_VIDEO_TDATA, is defined by two previously mentioned parameters; *"AXI4-Stream component width"*, *"AXI4-Stream video format",* and a third parameter *"Pixels per clock"*.

"Pixels per clock" parameter defines how many pixels are present on the input and output data bus on each active clock edge. The *"Pixels per clock"* parameter allows receiving and sending out one, two or four pixels per clock which allows increasing the pixel throughput while maintaining the same pixel clock frequency. This is especially useful when using relatively high resolutions and frame rates which require pixel clock frequencies that are too high for FPGA implementation, e.g. 4k2k@60fps.

To support this feature, logiCVC-ML uses parallel pixel processing which means all pixel processing logic (alpha blending, color space conversion) will be doubled or quadrupled depending on the *"Pixels per clock"* parameter.

9 MULTILAYER SUPPORT

The logiCVC-ML supports up to five layers. The actual number of layers that will be implemented is configurable through the *"Number of layers"* parameter. Block schematic of multilayer architecture is outlined in Figure 9.1.

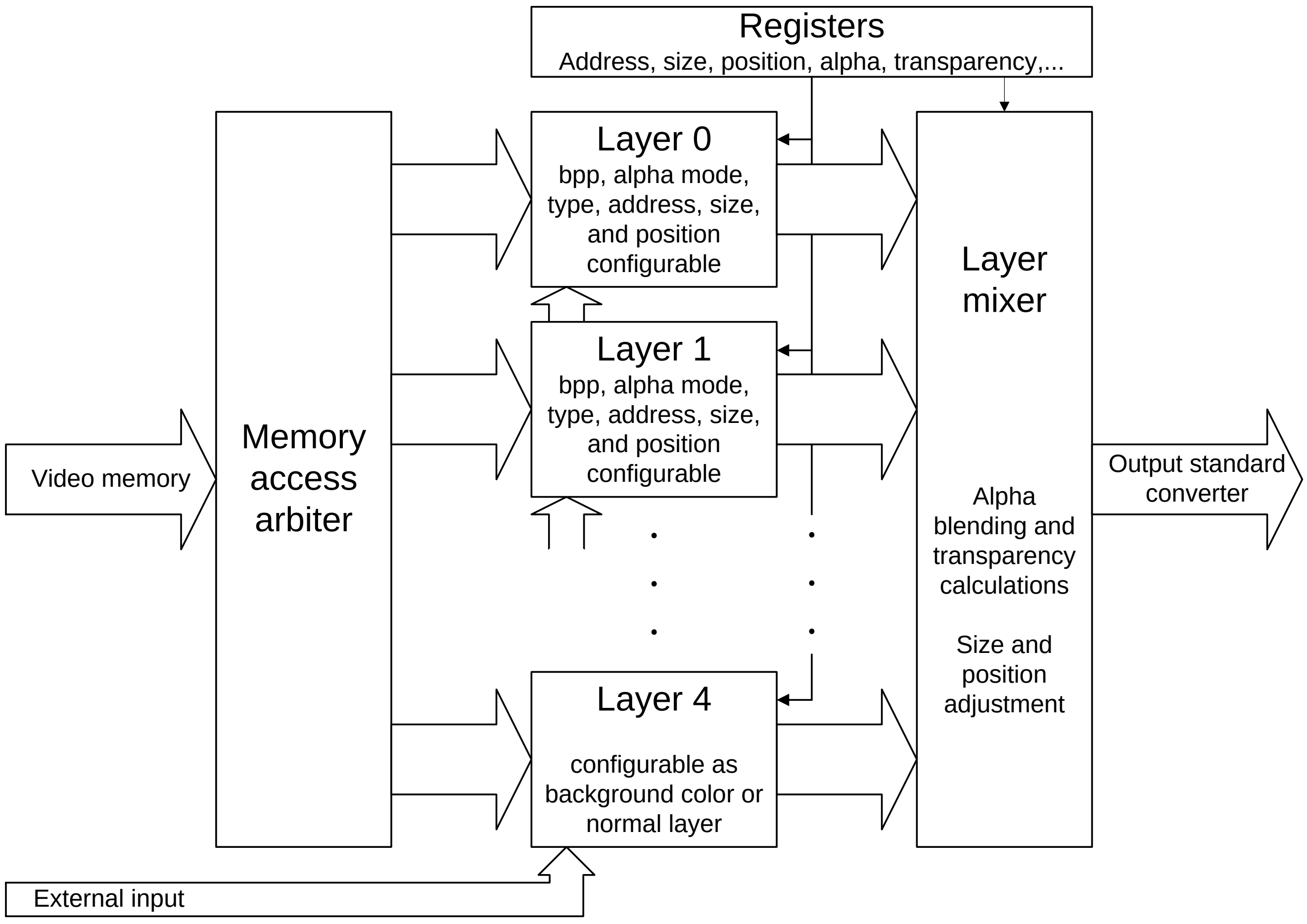

Registers
Address, size, position, alpha, transparency,...
Layer 0
bpp, alpha mode, type, address, size, and position configurable
Layer 1
bpp, alpha mode, type, address, size, and position configurable
Layer 4
configurable as background color or normal layer
Memory access arbiter
Video memory
Layer mixer
Alpha blending and transparency calculations
Size and position adjustment
Output standard converter
External input

Figure 9.1: Multilayer architecture

9.1 Memory Access Arbiter

Memory access arbiter arbitrates between n-number of requests from each layer and assigns equal amount of time to each layer. The arbitration algorithm used is round-robin. This method circulates priority between multiple layers, i.e. FIFOs. When one of them asserts request to memory, it grants a predefined amount of time to the FIFO. The arbitration continues when the layer which was granted request receives acknowledge from the AXI4 bus.

9.2 Layer Block

Layer consists of the following blocks:

- Memory address generator
- Pixel data FIFO
- Color Look-up Table, if used
- Layer related register set (address, size, position, …)
- 4:2:2 to 4:4:4 converter
- Color space converter

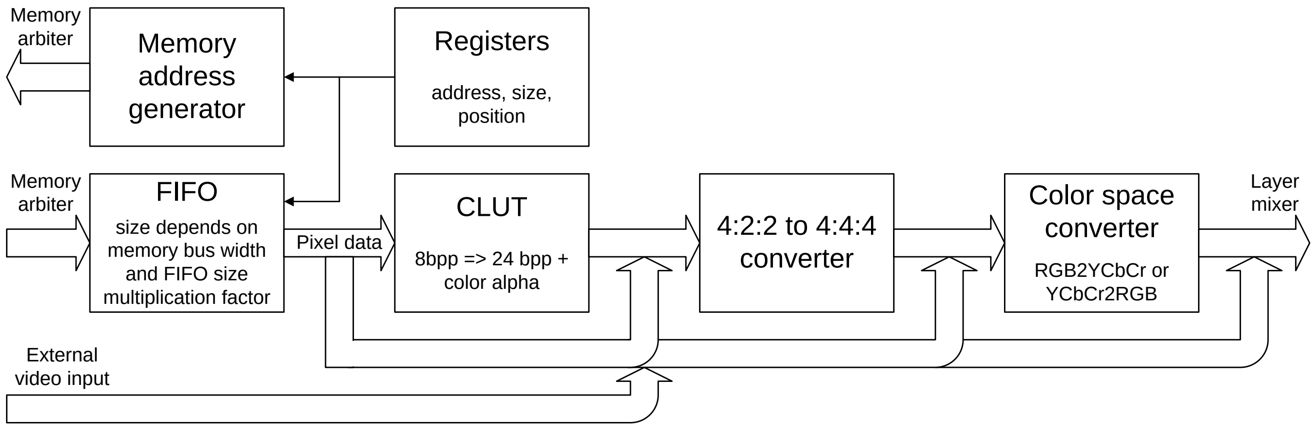


Figure 9.2: Layer block schematic

Depending on the logiCVC-ML configuration, layer can consist of some or all of the above mentioned blocks. For example, if logiCVC-ML is configured to output RGB (*Display interface color format*) and layer is configured as 16bpp RGB, it will only consist of three parts; address generator, FIFO and register set. However, if logiCVC-ML is configured to output RGB and layer is configured as 16bpp YCbCr (4:2:2) then layer block consists of five parts, address generator, FIFO, register set, 4:2:2 to 4:4:4 converter and YCbCr to RGB converter.

9.2.1 Memory Address Generator

Each logiCVC-ML layer that reads data from video buffer (not from external video input) has its starting address from which it reads data. This address is set by configuring layer address register.

In this way, each layer can be placed on the desired location in memory that is user assigned. Please refer to chapter 5.3 Memory address and range, for more information on how to calculate each layer memory range.

Additionally, layer size and position registers are used to resize each layer and move the starting point in memory from which the data will be read, i.e. the first pixel on the display. For more information on memory address, size and position registers and their functionality, please refer to chapters 10.2.15, 10.2.16, 10.2.17 and 9.3.

9.2.2 Pixel Data FIFO

Each layer that reads data from video buffer has one FIFO. That FIFO is used to temporarily buffer pixel data that is read from video memory on memory clock and output on pixel clock (video clock). Usually, memory clock is higher than pixel clock but this might not be the case for high video resolutions. Depending on configuration parameters, FIFO can be configured in many combinations. The parameters that determine the size of FIFO are *"FIFO size multiplication factor"* and *"AXI4 interface data bus width"*.

Table 9.1: Layer FIFO size in number of BRAMs (2kB)

		AXI4 interface data bus width		
		32	64	128
FIFO size multiplication factor	1	1	2	4
	2	2	4	8
	4	4	8	16
	8	8	16	32

9.2.3 Color Look-Up Table

If layer is configured as 8bpp and uses CLUT alpha blending mode, CLUT instance is generated. In this configuration, data read from video memory buffer represents indexes in the CLUT. At every of 256 indexes one 32-bit value is stored. Depending if 16bpp or 24bpp CLUT is used 16 or 24 bits of this value are pixel data and 6 or 8 bits are alpha value. In this way, alpha factor is assigned per color and every indexed color in CLUT can have its own alpha value. For more on CLUT alpha blending mode, please refer to chapter 9.3.2.3 Color Alpha Blending.

Table 9.2: Data organization of one CLUT index

CLUT data bit	**16bpp CLUT**	**24bpp CLUT**	
		RGB	**YCbCr**
0	Dummy	Blue0	Cr0
1	Dummy	Blue1	Cr1
2	Dummy	Blue2	Cr2
3	Blue0	Blue3	Cr3
4	Blue1	Blue4	Cr4
5	Blue2	Blue5	Cr5
6	Blue3	Blue6	Cr6
7	Blue4	Blue7	Cr7
8	Dummy	Green0	Cb0
9	Dummy	Green1	Cb1
10	Green0	Green2	Cb2
11	Green1	Green3	Cb3

CLUT data bit	16bpp CLUT	24bpp CLUT	
		RGB	YCbCr
12	Green2	Green4	Cb4
13	Green3	Green5	Cb5
14	Green4	Green6	Cb6
15	Green5	Green7	Cb7
16	Dummy	Red0	Y0
17	Dummy	Red1	Y1
18	Dummy	Red2	Y2
19	Red0	Red3	Y3
20	Red1	Red4	Y4
21	Red2	Red5	Y5
22	Red3	Red6	Y6
23	Red4	Red7	Y7
24	Alpha0	Alpha0	Alpha0
25	Alpha1	Alpha1	Alpha1
26	Alpha2	Alpha2	Alpha2
27	Alpha3	Alpha3	Alpha3
28	Alpha4	Alpha4	Alpha4
29	Alpha5	Alpha5	Alpha5
30	Dummy	Alpha6	Alpha6
31	Dummy	Alpha7	Alpha7

9.2.4 4:2:2 to 4:4:4 converter

In case logiCVC-ML layer is configured as 16bpp YCbCr (*"Layer data width"* and *"Layer type"* parameters) the pixel data in memory that this layer reads is in YCbCr 4:2:2 format. Therefore, each 16-bit pixel has its own luminance (Y) value but neighboring pixels share the same chroma values (Cb and Cr). So, in order for the logiCVC-ML alpha blending logic to properly calculate pixel values, 4:2:2 format needs to be converted to 4:4:4. This is achieved, by repeating the chroma values so that each pixel consists of its own Y, Cb and Cr value (Y_0Cb, Y_1Cr => Y_0CbCr, Y_1CbCr).

9.2.5 Color space converter

In case logiCVC-ML output color space (*"Display interface color format"*) is not the same as layer color space (*"Layer type"*), layer data is converted to the output color space before alpha blending is performed. In this way, all layers are converted to the same color space and can be alpha blended together.

If logiCVC-ML is configured to use YCbCr output space and some layers are not in YCbCr color format, then their RGB pixel data is converted from RGB to YCbCr color space. Depending on the output interface selection (parallel or ITU656) different conversion formula coefficients are used. The reason for this is that ITU656 interface requires specific parameters and limits the output range of luminance and chroma values in the range of 16-235. So, in the case of parallel output interface the following formulas are used:

Equation 9.1: Parallel interface RGB to YCbCr conversion formulas

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$
$$Cr = 0.4998 \cdot R - 0.4185 \cdot G - 0.08128 \cdot B + 128$$
$$Cb = -0.16868 \cdot R - 0.33107 \cdot G + 0.4997 \cdot B + 128$$

Input range of red, green and blue components is [0, 255], and output range of Y, Cb and Cr is [0, 255].

In the case of ITU656 output interface, the following formulas are used for RGB layers:

Equation 9.2: ITU656 interface RGB to YCbCr conversion formulas

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B + 16$$
$$Cr = 0.51138 \cdot R - 0.4282 \cdot G - 0.08316 \cdot B + 128$$
$$Cb = -0.17258 \cdot R - 0.33881 \cdot G + 0.5114 \cdot B + 128$$

Input range of red, green and blue components is [0, 255], and output range of Y, Cb and Cr is [16, 235].

If logiCVC-ML is configured to use RGB output space and some layers are not in RGB color format, then their YCbCr pixel data is converted from YCbCr to RGB color space using the following formulas:

Equation 9.3: YCbCr to RGB conversion formulas

$$R = Y + 1.402524 \cdot Cr - 179$$
$$G = Y - 0.714403 \cdot Cr - 0.34434 \cdot Cb + 135$$
$$B = Y + 1.773049 \cdot Cb - 226$$

Input range of luminance and chroma components is [0, 255], and output range of R, G and B is [0, 255].

9.3 Layer Mixer

Layer mixer handles transparency, alpha blending, layer size and layer position on the display.

Layer 0 is always the top layer, i.e. has the highest priority. Depending on number of layers, alpha factors, transparent color, layer sizes and positions, layers below layer 0, will or will not be visible.

Figure 9.3 represents logiCVC-ML example configuration. In this example, logiCVC-ML has four layers where the last layer is configured as background layer. The difference between a normal layer and a background one is that background layer does not read data from the video memory but outputs data from the logiCVC-ML background color register. To configure last layer as background user has to enable *"Configure last layer as background"* parameter.

The other three layers are randomly placed on the screen and are all using layer alpha blending mode. It can be seen that layer 0 has the highest priority and is completely visible because its alpha factor is set to maximum, i.e. 1 (0xFF). Layer 1 has an alpha factor of 0.5 which means that both layers underneath are slightly visible. Layer 2 has an alpha factor of 1 (0xFF) and is completely visible on the places that are not hidden by the upper layers.

Additionally, all layers except the background layer are smaller than horizontal and vertical display resolution and are placed at some positions on the screen. Using layer position registers user can easily move the starting point of the layer around the screen.

For more information on using layer size and position features, please refer to chapters 10.2.16 Layer Position Registers - Lx_H_POSITION, Lx_V_POSITION and 10.2.17 Layer Size Registers - Lx_WIDTH, Lx_HEIGHT.

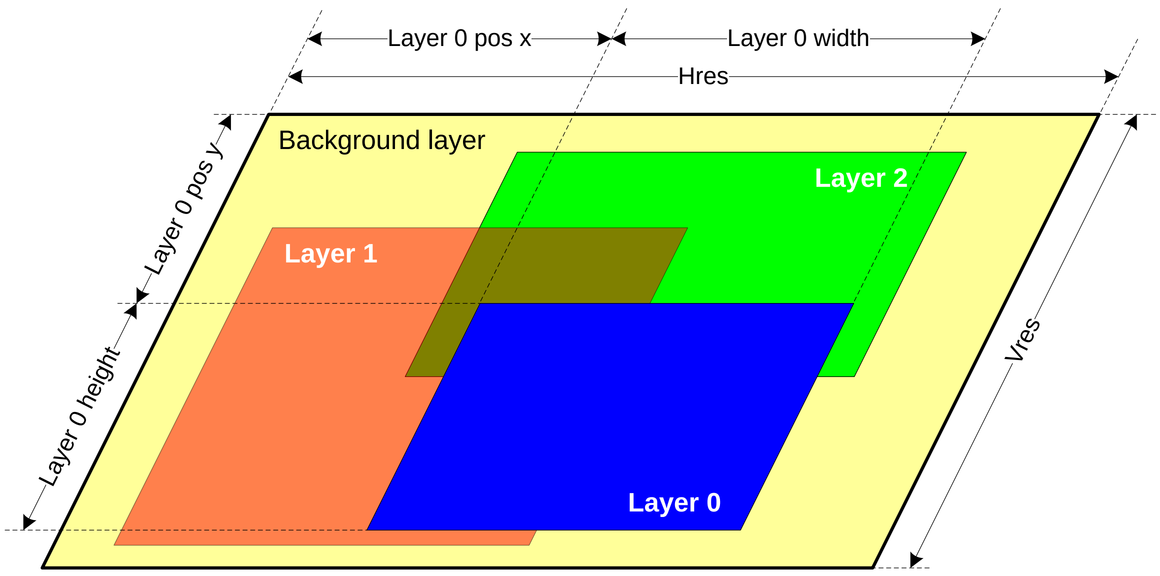


Figure 9.3: logiCVC-ML example configuration

9.3.1 Transparency

Transparency is achieved by using a color key, i.e. one dedicated color value per layer that is not displayed but used for transparency purposes. The pixel value equal to the assigned color key will not be visualized and instead the bellow layer pixel (layer with the lower priority) will be visualized. The color key is applied for all 8bpp, 16bpp, 24bpp and 30bpp color depths and depends on the layer color depth. It can be seen that for 8bpp one of the 256 colors will not be visible on the screen. Color key overrides alpha factor setting for that layer i.e. it has higher priority.

Each layer has its own transparency color register. Refer to chapter 10.2.20 Layer Transparent Color Register - Lx_TRANSPARENT for more information.

9.3.2 Alpha Blending

There are three ways to achieve alpha blending, i.e. blending from one picture to the other (also known as morphing). There is layer alpha blending, pixel alpha blending and color alpha blending using Color Look-Up Table (CLUT).

All of the above-mentioned modes use the same mathematic formula for calculating the result of alpha blending between two layers. The following formula is applied to every color of every pixel on the screen for n-1 times, where n is the number of layers used.

$$result\_layer = \alpha_0 \times layer_0 + (1-\alpha_0) \times layer_1$$

In the above equation, layer$_0$ is the top layer and layer$_1$ is the layer beneath. The equation is calculated three times for every pixel, once for each RGB color.

For the example configuration represented on Figure 9.3, the following equations would be used:

$$\begin{aligned} temp_{23} &= \alpha_2 \times layer_2 + (1-\alpha_2) \times layer_3 \\ temp_{123} &= \alpha_1 \times layer_1 + (1-\alpha_1) \times temp_{23} \\ result &= \alpha_0 \times layer_0 + (1-\alpha_0) \times temp_{123} \end{aligned}$$

9.3.2.1 Layer Alpha Blending

When using this mode, alpha factor is stored in layer alpha register (LX_ALPHA) and is constant for all pixels on the specific layer. Figure 9.4 shows a simple usage of layer alpha blending. First picture is representing layer$_0$ and last layer$_1$. The pictures between represent alpha blending with alpha factor steps of 25% (0.25).

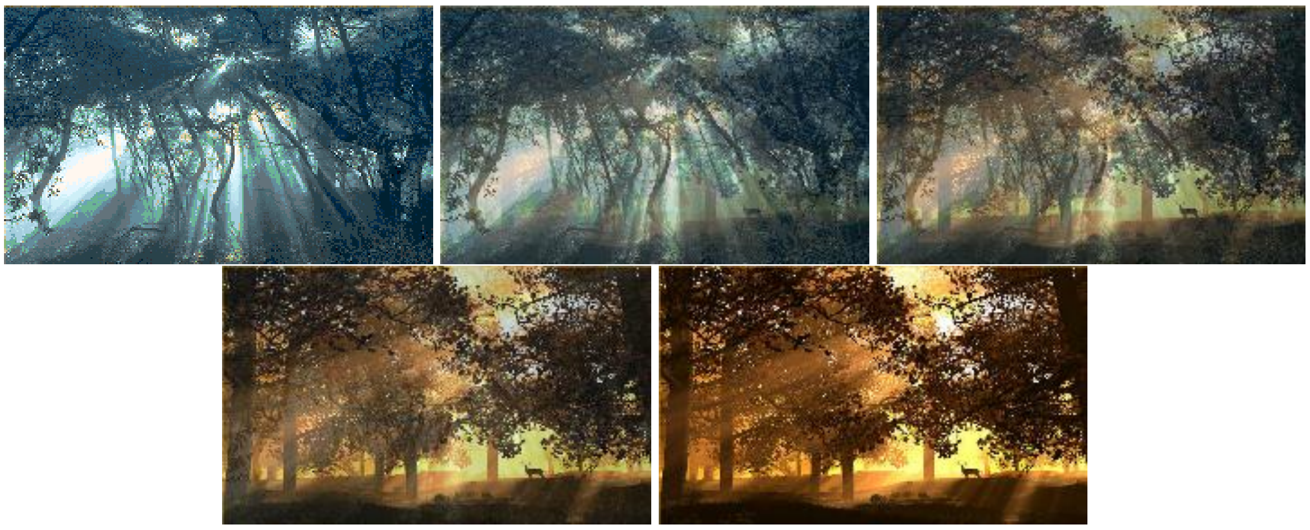


Figure 9.4: Layer alpha blending

9.3.2.2 Pixel Alpha Blending

In contrast to layer alpha blending mode where one alpha factor is used for all pixels on that layer, in pixel alpha mode every pixel has its own alpha factor.

Pixel alpha blending is particularly useful for blending of computer-generated graphics and live video from TV tuners or cameras. In that case, alpha blending may be used to remove jaggy edges form the graphic objects over live video.

The logiCVC-ML supports two modes of pixel alpha blending, which differentiate themselves by the store location of alpha blending factor for each pixel.

When pixel alpha blending mode is selected by setting *"Alpha blending mode"* to *Pixel*, alpha factor is stored together with pixel color value. In this case, each pixel read from video buffer consists of color and alpha factor, e.g. ARGB or AYCbCr, and this increases the size of each pixel for alpha bits according to Table 9.3.

Table 9.3: Pixel Alpha blending

Pixel color depth	Color value	Alpha value	Pixel width	Notes
8bpp	8-bit	3-bit	16-bit	
16bpp	16-bit	6-bit	32-bit	3rd byte is dummy
24bpp	24-bit	8-bit	32-bit	
30bpp	30-bit	NA	32-bit	Pixel alpha blending not supported

For example, when 16bpp color depth is used with pixel alpha blending, two bytes represent pixel color value, one byte represents alpha factor and one byte is dummy. This dummy byte is the downside of this method as it increases the required bandwidth for one byte per pixel (25%).

For this reason, logiCVC-ML supports a second mode of pixel alpha blending which is called alpha plane. In this case, a separate layer is used to read alpha factors that will be used for blending two layers. Therefore, two data layers are used for reading pixel color values and one is used to read pixel alpha values.

Please note that only layers 1 and 3 can be configured as alpha plane layers (*"Layer type"* set to *Alpha*). In case Layer 1 is configured as alpha plane layer, Layer 1 is considered to hold an alpha value for Layer 0 and it is used for alpha blending between Layers 0 and 2. In the same way, if Layer 3 is configured as alpha plane layer it is used for alpha blending between Layers 2 and 4.

Alpha plane layers can only be 8-bit as the maximum alpha value can be 255.

9.3.2.3 Color Alpha Blending

Color alpha blending is achieved by using Color Look-Up Table (CLUT), which is structured as 256 locations of 32 bits. Additionally, to transform 8bpp to 24bpp, CLUT blending mode adds 8-bit alpha factor to each location in CLUT. Here the alpha factor is associated to one color and therefore object with same color will have equal alpha factor. Then the alpha blending factor for 8bpp is not assigned to each pixel of one layer but to all pixels on one layer having the same color. It is possible to have up to 256 different alpha factors. This alpha blending mode saves memory bandwidth due to keeping color depth in video memory at 8-bit per pixel.

CLUT supports RGB and YCbCr 4:4:4 color spaces. YCbCr 4:2:2 CLUT values are not supported.

10 REGISTERS

The logiCVC-ML registers are accessed through the AXI4-Lite Slave data interface by ARM, MicroBlaze or any other AXI4-Lite Master.

By default, all registers inside logiCVC-ML can be read and write, except IP version and external resolution registers, which can only be read. However, some registers have different write and read values and the reading functionality can be switched off disabling the *"Readable logiCVC registers"* parameter.

Using the *"Swap bytes for logiCVC register access"* parameter, user can choose to swap bytes for logiCVC-ML register access. If this parameter is set, byte ordering on the bus will change ($B_0B_1B_2B_3$ => $B_3B_2B_1B_0$).

10.1 Register Map

All logiCVC-ML registers are placed at an address offset defined by *"Registers base/high address"* parameters that can be configured from Vivado Block Design Address Editor.

The following table describes the names and addresses of logiCVC-ML registers on the AXI4-Lite bus.

Table 10.1: logiCVC-ML register map

Address offset1)	**Type**2)	**Size (bits)**	**Name**	**Description**
General Registers				
0x0000	(R)/W	16	HSY_FP	Horizontal Sync front porch
0x0008	(R)/W	16	HSY	Horizontal Sync pulse width
0x0010	(R)/W	16	HSY_BP	Horizontal Sync back porch
0x0018	(R)/W	16	H_RES	Horizontal resolution
0x0020	(R)/W	16	VSY_FP	Vertical Sync front porch
0x0028	(R)/W	16	VSY	Vertical Sync pulse width
0x0030	(R)/W	16	VSY_BP	Vertical Sync back porch
0x0038	(R)/W	16	V_RES	Vertical resolution
0x0040	(R)/W	24	CTRL	Control
0x0048	(R)/W	8	DTYPE	Display type
0x0050	(R)/W	24	BACKGROUND	Background color
0x0058				Not used
0x0060	R/W	16	CLUT_SEL	CLUT select for all CLUT layers 3)
0x0068	R/(W)	16	INT_STAT	Interrupt status register
0x0070	(R)/W	16	INT_MASK	Interrupt mask register

Address offset1)	Type2)	Size (bits)	Name	Description
0x0078	R/W	8	PWRCTRL	Power control
0x0080	(R)	16	E_HRES	External input horizontal resolution
0x0088	(R)	16	E_VRES	External input vertical resolution
0x00F8	R	32	IP_VERSION	logiCVC-ML IP version information
Layer 0 Registers4)				
0x0100	(R)/W	32	L0_ADDR	Layer0 memory address
0x0108				Not used
0x0110	(R)/W	16	L0_H_POSITION	Layer0 horizontal position
0x0118	(R)/W	16	L0_V_POSITION	Layer0 vertical position
0x0120	(R)/W	16	L0_WIDTH	Layer0 width
0x0128	(R)/W	16	L0_HEIGHT	Layer0 height
0x0130	(R)/W	16	L0_ALPHA	Layer0 layer alpha factor
0x0138	(R)/W	8	L0_CTRL	Layer0 Control
0x0140	(R)/W	24	L0_TRANSPARENT	Layer0 color key transparency value
Layer 1 Registers4)				
0x0180	(R)/W	32	L1_ADDR	Layer1 memory address
0x0188				Not used
0x0190	(R)/W	16	L1_H_POSITION	Layer1 horizontal position
0x0198	(R)/W	16	L1_V_POSITION	Layer1 vertical position
0x01A0	(R)/W	16	L1_WIDTH	Layer1 width
0x01A8	(R)/W	16	L1_HEIGHT	Layer1 height
0x01B0	(R)/W	16	L1_ALPHA	Layer1 layer alpha factor
0x01B8	(R)/W	8	L1_CTRL	Layer1 Control
0x01C0	(R)/W	24	L1_TRANSPARENT	Layer1 color key transparency value
Layer 2 Registers4)				
0x0200	(R)/W	32	L2_ADDR	Layer2 memory address
0x0208				Not used
0x0210	(R)/W	16	L2_H_POSITION	Layer2 horizontal position
0x0218	(R)/W	16	L2_V_POSITION	Layer2 vertical position
0x0220	(R)/W	16	L2_WIDTH	Layer2 width
0x0228	(R)/W	16	L2_HEIGHT	Layer2 height
0x0230	(R)/W	16	L2_ALPHA	Layer2 layer alpha factor
0x0238	(R)/W	8	L2_CTRL	Layer2 Control
0x0240	(R)/W	24	L2_TRANSPARENT	Layer2 color key transparency value

Address offset[1]	Type[2]	Size (bits)	Name	Description
Layer 3 Registers[4]				
0x0280	(R)/W	32	L3_ADDR	Layer3 memory address
0x0288				Not used
0x0290	(R)/W	16	L3_H_POSITION	Layer3 horizontal position
0x0298	(R)/W	16	L3_V_POSITION	Layer3 vertical position
0x02A0	(R)/W	16	L3_WIDTH	Layer3 width
0x02A8	(R)/W	16	L3_HEIGHT	Layer3 height
0x02B0	(R)/W	16	L3_ALPHA	Layer3 layer alpha factor
0x02B8	(R)/W	8	L3_CTRL	Layer3 Control
0x02C0	(R)/W	24	L3_TRANSPARENT	Layer3 color key transparency value
Layer 4 Registers[4]				
0x0300	(R)/W	32	L4_ADDR	Layer4 memory address
				Not used
0x0338	(R)/W	8	L4_CTRL	Layer4 Control
CLUT Registers				
0x1000	(R)/W	1k Bytes	L0_CLUT_0	Layer0 Color Look-Up Table 0
0x1800	(R)/W	1k Bytes	L0_CLUT_1	Layer0 Color Look-Up Table 1
0x2000	(R)/W	1k Bytes	L1_CLUT_0	Layer1 Color Look-Up Table 0
0x2800	(R)/W	1k Bytes	L1_CLUT_1	Layer1 Color Look-Up Table 1
0x3000	(R)/W	1k Bytes	L2_CLUT_0	Layer2 Color Look-Up Table 0
0x3800	(R)/W	1k Bytes	L2_CLUT_1	Layer2 Color Look-Up Table 1
0x4000	(R)/W	1k Bytes	L3_CLUT_0	Layer3 Color Look-Up Table 0
0x4800	(R)/W	1k Bytes	L3_CLUT_1	Layer3 Color Look-Up Table 1
0x5000	(R)/W	1k Bytes	L4_CLUT_0	Layer4 Color Look-Up Table 0
0x5800	(R)/W	1k Bytes	L4_CLUT_1	Layer4 Color Look-Up Table 1

1. All registers are placed at the 64-bit boundary; including CLUT indexes.
2. R – readable; W – writeable; (R) – readable when *"Readable logiCVC registers"* parameter is enabled; (W) – clear only.
3. CLUT select registers have different read/write values.
4. Layer 4 and any layer that is last in the configuration has limited functionality because it can only be the last layer (logiCVC-ML has a maximum of five layers) and there is nothing below this layer that can be shown on the screen. That is the reason some functionality (Lx_H_POSITION, Lx_V_POSITION, Lx_WIDTH, Lx_HEIGHT, Lx_ALPHA and Lx_TRANSPARENT) is not available for layer 4 or any layer that is last in the configuration.
5. External input horizontal and vertical resolution registers exist only when *"Use external input"* parameter is set to *Parallel*.

10.2 Register Description

10.2.1 Horizontal Sync and Porch Registers - HSY_FP, HSY, HSY_BP

The values in registers HSY_FP, HSY and HSY_BP should be written in the number of VCLK periods. Note that sum of HSY_FP, HSY and HSY_BP has to be equal to the increments of the pixel clock period.

Table 10.2: HSY_FP Register Description

Bits	Address offset	Name	Type	Reset value	Description
9 - 0	0x000	HSY_FP	(R)/W	0x07F	Horizontal front porch

The value written into HSY_FP register determines the duration of the horizontal front porch (HFP). HFP equals the time between the deactivation of the BLANK signal and the next active HSYNC signal.

The duration of HFP is equal to the value written into the HSY_FP register incremented by one in VCLK periods.

Table 10.3: HSY Register Description

Bits	Address offset	Name	Type	Reset value	Description
9 - 0	0x008	HSY	(R)/W	0x07F	Horizontal sync

The duration of the HSYNC signal equals the value written into the HSY register incremented by 1 in VCLK periods.

Table 10.4: HSY_BP Register Description

Bits	Address offset	Name	Type	Reset value	Description
9 - 0	0x010	HSY_BP	(R)/W	0x07F	Horizontal back porch

The value written into HSY_BP register determines duration of the horizontal back porch (HBP). HBP equals the time from the moment of HSYNC signal deactivation to the moment of valid display data (BLANK signal activation). The duration of HBP equals the value written into the HSY_BP register incremented by one in VCLK periods.

Please refer to Figure 7.1 for sync and front and back porch definitions.

CTRL register controls the horizontal sync signal polarity and usage. Refer to chapter 10.2.5 Control Register - CTRL for more information.

HSY_FP, HSY and HSY_BP cannot be assigned if ITU656 output standard is used (*"Display interface type"* parameter) because they are predefined by the standard itself.

10.2.2 Horizontal Resolution Register - H_RES

Table 10.5: H_RES Register Description

Bits	Address offset	Name	Type	Reset value	Description
15 - 0	0x018	H_RES	(R)/W	0x027F	Horizontal resolution

logiCVC-ML supports horizontal resolutions from 64 to 8192 pixels but the maximum horizontal resolution, i.e. the size of horizontal resolution register, is constrained by the *"Row stride"* parameter in a way that the number of bits used in H_RES register is equal to $\log_2$(*Row stride*). Otherwise, there are no restrictions on the values written in the registers besides the minimum and the maximum value. The values in the H_RES register should be written in number of pixels.

The picture's horizontal resolution equals to the value written in this register incremented by 1 in pixels.

Please refer to Figure 7.1 for horizontal resolution definition.

H_RES cannot be assigned if ITU656 output standard is used (*"Display interface type"* parameter) because it is predefined by the standard itself and NTSC_PAL flag in DTYPE register.

Please check the *"Row stride"* parameter for proper functioning. Usually, *"Row stride"* should be set to a maximum required horizontal resolution. For more information on memory row stride, please refer to chapter 5.1 Memory Layout.

10.2.3 Vertical Sync and Porch Registers - VSY_FP, VSY, VSY_BP

The values in VSY_FP, VSY and VSY_BP registers are specified in the number of pixel rows (picture lines).

Table 10.6: VSY_FP Register Description

Bits	Address offset	Name	Type	Reset value	Description
9 - 0	0x020	VSY_FP	(R)/W	0x000	Vertical Front Porch Register

The value written into VSY_FP register determines duration of the vertical front porch (VFP). The VFP duration is the time between the moment of the valid pixel data end (BLANK signal deactivation) and the moment of VSYNC signal activation.

The duration of VFP equals the value written in the VSY_FP register incremented by one in pixel rows (lines).

Table 10.7: VSY Register Description

Bits	Address offset	Name	Type	Reset value	Description
9 - 0	0x028	VSY	(R)/W	0x000	Vertical Sync Register

The value written into VSY register determines the duration of the VSYNC signal (vertical sync).

The duration of the VSYNC signal equals the value written in the VSY register incremented by 1 in pixels rows (lines).

Table 10.8: VSY_BP Register Description

Bits	Address offset	Name	Type	Reset value	Description
9 - 0	0x030	VSY_BP	(R)/W	0x000	Vertical Back Porch Register

The value written to the VSY_BP register determines the duration of the vertical back porch (VBP). The VBP duration time is the time between the moment of VSYNC signal deactivation and the occurrence of the first pixel in a frame (BLANK signal activation).

The duration of VBP equals the value written in the VSY_BP register increased by 1 in pixel rows (lines).

Please refer to Figure 7.1 for sync and front and back porch definitions.

CTRL register controls the vertical sync polarity and usage. Refer to chapter 10.2.5 Control Register - CTRL for more information.

VSY_FP, VSY and VSY_BP cannot be assigned if ITU656 output standard is used (*"Display interface type"* parameter) because they are predefined by the standard itself.

10.2.4 Vertical Resolution Register - V_RES

Table 10.9: V_RES Register Description

Bits	Address offset	Name	Type	Reset value	Description
15 - 0	0x038	V_RES	(R)/W	0x01DF	Vertical Resolution Register

The logiCVC-ML supports vertical resolutions from 1 to 8192 pixel lines and there are no restrictions on the values written in the registers besides the minimum and the maximum value. The value in V_RES register is specified in the number of pixel rows (picture lines).

Vertical resolution equals the value written in this register incremented by 1 in picture lines.

Please refer to Figure 7.1 for vertical resolution definition.

V_RES cannot be assigned if ITU656 output standard is used (*"Display interface type"* parameter) because it is predefined by the standard itself and NTSC_PAL flag in DTYPE register.

10.2.5 Control Register - CTRL

Table 10.10: CTRL Register Description

Bits	Address offset	Name	Type	Reset value	Description
23 - 0	0x040	CTRL	(R)/W	0x0000	Control Register

The value written into CTRL register enables, disables and inverts the polarity of the control video signals. Additionally, CTRL register enables an inversion of the PIX_CLK signal (by default logiCVC-ML outputs data on falling edge of PIX_CLK), inverts the polarity of external parallel interface control signals and enables/disables updating of layer address, size and position registers.

Table 10.11: Control Register Map

Bits	Name	Type	Reset value	Description
0	HSEN	(R)/W	0	HSYNC enable (0 - disabled, 1 - enabled)
1	HSINV	(R)/W	0	HSYNC invert (0 - not inverted, 1 - inverted)
2	VSEN	(R)/W	0	VSYNC enable (0 - disabled, 1 - enabled)
3	VSINV	(R)/W	0	VSYNC invert (0 - not inverted, 1 - inverted)
4	ENEN	(R)/W	0	BLANK/DE enable (0 - disabled, 1 - enabled)
5	ENINV	(R)/W	0	BLANK/DE invert (0 - not inverted, 1 - inverted)
6				Not used
7	PIXINV	(R)/W	0	Pixel data invert (0 - not inverted, 1 - inverted)
8	CLKINV	(R)/W	0	PIX_CLK invert (0 - not inverted, 1 - inverted)
9	DIS_UPDATE	(R)/W	0	Disable updating of layer address, size and position registers (0 - enabled, 1 - disabled)
10			0	Not used
15 - 11	GPIO	(R)/W	0	General purpose input/output
16	E_V_PR_INV	(R)/W	0	External video present invert (0 - not inverted, 1 - inverted)
17	EVSINV	(R)/W	0	External VSYNC invert (0 - not inverted, 1 - inverted)
18	EHSINV	(R)/W	0	External HSYNC invert (0 - not inverted, 1 - inverted)
19	EBLINV	(R)/W	0	External BLANK/DE invert (0 - not inverted, 1 - inverted)
23 - 20				Not used

GPIO[4:0] bits inside the display control register are directly routed from/to logiCVC-ML input/output signals GPI and GPO. By reading these bits user is presented with the current state of the GPI signal. Performing a write to GPIO bits the state of the GPO signal changes accordingly. This allows the user to use these bits for general purpose. For example, they can be used to control clock signal frequency by adjusting an external clock module or PLL.

10.2.6 Display Type Register - DTYPE

Table 10.12: DTYPE Register Description

Bits	Address offset	Name	Type	Reset value	Description
7 - 0	0x048	DTYPE	(R)/W	0x00	Display Type Register

The value written into DTYPE register defines the general characteristics of the display used: TFT or other video display type.

Performing a write to this register generates internal logiCVC-ML reset which reloads the state machine and restarts the logiCVC-ML (registers are not affected). Because of this, it is advised that DTYPE register is written last.

Table 10.13: Display Type Register Map

Bits	Name	Type	Reset value	Description
5 - 0				Not used
6	NTSC_PAL	(R)/W	0	NTSC/PAL ITU656 standard selection
7				Not used

NTSC_PAL: This flag is only valid when *"Display interface type"* parameter is set to *ITU656*.

Table 10.14: NTSC_PAL Register Description

NTSC_PAL	Description
0	PAL video standard
1	NTSC video standard

10.2.7 Background Color Register - BACKGROUND

Table 10.15: BACKGROUND Register Description

Bits	Address offset	Name	Type	Reset value	Description
29 - 0	0x050	BACKGROUND	(R)/W	0x000000	Background Color Register

If *"Configure last layer as background"* parameter is enabled, last layer is configured as background and data is not read from video memory but from background color register. If all alpha factors from the layers above the last one are set to 0, background color will be visible. Also, if layer sizes of the layers above are smaller than display size background will be visible, see Figure 9.3.

Depending on layer type (RGB or YCbCr), different color depth modes are supported. For RGB background layer four depths are supported: 8bpp, 16bpp, 24bpp and 30bpp. If last layer is configured as 24bpp (C_LAYER_X_DATA_WIDTH = 24) or 8bpp with CLUT (24bpp), 24 bits from the register are used. In the same way, if last layer is configured as 16bpp (C_LAYER_X_DATA_WIDTH = 16) or 8bpp with CLUT (16bpp) only lowest 16 bits (RGB565) from the register are used. For 8bpp, only lowest 8 bits are used. For YCbCr background layer only 32bpp or 24bpp is supported.

10.2.8 CLUT Select Register - CLUT_SEL

Table 10.16: CLUT_SEL Register Description

Bits	Address offset	Name	Type	Reset value	Description
15 - 0	0x060	CLUT_SEL	R/W	0x0000	CLUT Select Register

This register controls which of the two CLUTs the logiCVC-ML uses on the specific layer. Every layer configured for CLUT alpha blending has its own CLUT and all of them can be controlled through this register.

Table 10.17: CLUT Select Register Map

Bits	Name	Type	Reset value	Description
0	L0_CLUT_SEL_0	R/W	0	Layer0 CLUT select bit 0
1	L0_CLUT_SEL_1	R/W	0	Layer0 CLUT select bit 1
2	L1_CLUT_SEL_0	R/W	0	Layer1 CLUT select bit 0
3	L1_CLUT_SEL_1	R/W	0	Layer1 CLUT select bit 1
4	L2_CLUT_SEL_0	R/W	0	Layer2 CLUT select bit 0
5	L2_CLUT_SEL_1	R/W	0	Layer2 CLUT select bit 1
6	L3_CLUT_SEL_0	R/W	0	Layer3 CLUT select bit 0
7	L3_CLUT_SEL_1	R/W	0	Layer3 CLUT select bit 1
8	L4_CLUT_SEL_0	R/W	0	Layer4 CLUT select bit 0
9	L4_CLUT_SEL_1	R/W	0	Layer4 CLUT select bit 1
10	L0_CLUT_SEL_WR_EN	R/W	0	Layer0 CLUT select write enable
11	L1_CLUT_SEL_WR_EN	R/W	0	Layer1 CLUT select write enable
12	L2_CLUT_SEL_WR_EN	R/W	0	Layer2 CLUT select write enable
13	L3_CLUT_SEL_WR_EN	R/W	0	Layer3 CLUT select write enable
14	L4_CLUT_SEL_WR_EN	R/W	0	Layer4 CLUT select write enable
15				Not used

After writing data in non-active CLUT, i.e. not used by logiCVC-ML, i.e. not selected in this register, CPU writes to one of the CLUT select register bits an incremented value signaling to the logiCVC-ML to switch the corresponding CLUT. After finishing reading current frame from video memory logiCVC-ML will switch the pointer of the selected CLUT to the next CLUT, as stated in the register, and set the corresponding bit in the INT_STAT register signaling completion. The CLUT is swapped on vertical retrace (VSYNC signal active), so there is no image flickering.

Additional write enable bits exist in the CLUT select register, which enables the user to change CLUT of only some layers without the need to read the register in order not to overwrite other bits. So for example, if user wants layer 1 to use CLUT 1 it has to write 0x0804.

All bits can be written at the same time so that double CLUT selecting occurs simultaneously for all CLUT layers.

When reading this register user will get the information which CLUT the logiCVC-ML uses currently, not the value it has written. Only after the CLUT is switched the same value will be read.

If *"Configure last layer as background"* parameter is enabled, last layer is not fetching data from memory or CLUT so CLUT select cannot be used for that layer.

10.2.9 Interrupt Status Register - INT_STAT

Table 10.18: INT_STAT Register Description

Bits	Address offset	Name	Type	Reset value	Description
15 - 0	0x068	INT_STAT	R/(W)	0x0000	Interrupt Status Register

Interrupt status register consists of the following bit groups:

Layer registers update status

Using four interrupt status bits, logiCVC-ML signals that it has performed an update of layer address, size and position registers. This interrupt sources will only become active if there was a write access to the above mentioned registers. Please refer to chapters 10.2.15, 10.2.16 and 10.2.17 for more information.

The update, and the interrupt assertion, will be executed on the next frame start.

V-Sync status

Bit 5 is used for signaling that a new frame is starting. This can be used as a feedback to the CPU for applications where user wants to change some logiCVC-ML parameters once per frame.

External parallel input valid status

This flag is used for signaling to the CPU that external parallel input had been detected and its parameters, porches, syncs and resolutions were all measured. After this bit is set, internal logiCVC-ML state machines and video logic are reset and reloaded with the measured parameters. In addition, after this bit is set, values in E_HRES and E_VRES are valid.

This flag is only valid when *"Use external input"* parameter is set to *Parallel*.

FIFO underrun status

One interrupt status bit is used to signal a FIFO underrun error. When this bit is asserted, one or more layer FIFOs have underrun which means they are not sufficiently filled with new memory data. It usually occurs when there is not enough memory bandwidth available for logiCVC-ML to properly refresh the display and it represents itself as a corrupt display picture. This bit can become active only once per frame.

CLUT select status

Through these interrupt status bits, logiCVC-ML signals that it has performed an action of switching CLUT pointer as stated in the CLUT_SEL register.

Table 10.19: Interrupt Status Register Map

Bits	Name	Type	Reset value	Description
0	L0_REGS_UPDATED	R/(W)	0	Layer0 address, size and position registers updated
1	L1_ REGS_UPDATED	R/(W)	0	Layer1 address, size and position registers updated
2	L2_ REGS_UPDATED	R/(W)	0	Layer2 address, size and position registers updated
3	L3_ REGS_UPDATED	R/(W)	0	Layer3 address, size and position registers updated
4	L4_ REGS_UPDATED	R/(W)	0	Layer4 address, size and position registers updated
5	V_SYNC	R/(W)	0	Vertical sync active
6	E_VIDEO_VALID	R/(W)	0	External parallel input valid
7	FIFO_UNDERRUN	R/(W)	0	FIFO underrun error
8	L0_CLUT_SW	R/(W)	0	Layer0 CLUT switched
9	L1_CLUT_SW	R/(W)	0	Layer1 CLUT switched
10	L2_CLUT_SW	R/(W)	0	Layer2 CLUT switched
11	L3_CLUT_SW	R/(W)	0	Layer3 CLUT switched
12	L4_CLUT_SW	R/(W)	0	Layer4 CLUT switched
15 - 13				Not used

1. If *“Configure last layer as background”* parameter is enabled, last layer is not fetching data from memory or CLUT so CLUT switched interrupt cannot be used.
2. If a certain layer is disabled using Enable bit in Layer control register, that layer cannot generate CLUT buffer switched interrupt in the Interrupt status register.
3. E_VIDEO_VALID flag is only valid when *“Use external input”* parameter is set to *Parallel*.

INT_STAT register can only be cleared by writing ‘1’ to the active flag. All flags inside the INT_STAT register generate an interrupt on the logiCVC-ML interrupt signal pin if the corresponding bit is not masked in the INT_MASK register.

10.2.10 Interrupt Mask Register - INT_MASK

Table 10.20: INT_MASK Register Description

Bits	Address offset	Name	Type	Reset value	Description
15 - 0	0x070	INT_MASK	(R)/W	0xFFFF	Interrupt Mask Register

Each bit in the interrupt status register, INT_STAT, has its corresponding bit in the interrupt mask register, INT_MASK, allowing it to be disabled from generating an interrupt on the INTERRUPT pin of the logiCVC-ML. Writing '1' to one of the interrupt mask bits, disables that interrupt source from generating an interrupt.

Table 10.21: Interrupt Mask Register Map

Bits	Name	Type	Reset value	Description
0	L0_REGS_UPDATED_MASK	(R)/W	1	Layer0 registers updated interrupt mask
1	L1_REGS_UPDATED_MASK	(R)/W	1	Layer1 registers updated interrupt mask
2	L2_REGS_UPDATED_MASK	(R)/W	1	Layer2 registers updated interrupt mask
3	L3_REGS_UPDATED_MASK	(R)/W	1	Layer3 registers updated interrupt mask
4	L4_REGS_UPDATED_MASK	(R)/W	1	Layer4 registers updated interrupt mask
5	V_SYNC_MASK	(R)/W	1	Vertical sync active interrupt mask
6	E_VIDEO_VALID_MASK	(R)/W	1	External parallel input valid interrupt mask
7	FIFO_UNDERRUN_MASK	(R)/W	1	FIFO underrun interrupt mask
8	L0_CLUT_SW_MASK	(R)/W	1	Layer0 CLUT switch interrupt mask
9	L1_CLUT_SW_MASK	(R)/W	1	Layer1 CLUT switch interrupt mask
10	L2_CLUT_SW_MASK	(R)/W	1	Layer2 CLUT switch interrupt mask
11	L3_CLUT_SW_MASK	(R)/W	1	Layer3 CLUT switch interrupt mask
12	L4_CLUT_SW_MASK	(R)/W	1	Layer4 CLUT switch interrupt mask
13				Not used
14				Not used
15				Not used

10.2.11 Power Control Register - PWRCTRL

Table 10.22: PWCTRL Register Description

Bits	Address offset	Name	Type	Reset value	Description
7 - 0	0x078	PWCTRL	R/W	0x00	Power Control Register

The liquid crystal inside LCD displays require precise timing sequencing to protected them from voltages that can permanently damage them. Most common power sequencing procedure consists of:

- Enabling the V_{DD} power supply that supplies the on-board circuitry and activates the display's internal clock signal. The internal clock signal sets the AC signal of the display's pins. It prevents the flow of DC current through the liquid crystal.
- After V_{DD} power supply is stabilized, the display clock, control and data signals can be supplied.
- Upon their stabilization, power supply V_{EE} can be supplied.
- After the stabilization of V_{EE} power supply, signal DISP_ON can be activated.

Programming the PWRCTRL register fulfils this power-sequencing procedure by providing separate bits for controlling each stage of the power sequence.

PWRCTRL register bits are directly routed to the logiCVC-ML output signals with the same signal names.

The EN_V bit besides being routed to the logiCVC-ML output also enables the three stated D_PIX, H_SYNC, V_SYNC, BLANK, PIX_CLK, LVDS_DATA_OUT and LVDS_CLK_OUT output signals.

Table 10.23: Power Control Register Map

Bits	Name	Type	Reset value	Description
0	EN_BLIGHT	R/W	0	Enable backlight (0 – disabled, 1 – enabled)
1	EN_VDD	R/W	0	Enable VDD (0 – disabled, 1 – enabled)
2	EN_VEE	R/W	0	Enable VEE (0 – disabled, 1 – enabled)
3	EN_V	R/W	0	Enable display control and data signals in parallel and LVDS interface (0 – disabled, 1 – enabled)
7 - 4				Not used

10.2.12 External Input Horizontal Resolution - E_HRES

Table 10.24: E_HRES Register Description

Bits	Address offset	Name	Type	Reset value	Description
15 - 0	0x080	E_HRES	(R)	0x0000	External Input Horizontal Resolution Register

If external parallel input interface is enabled *("Use external input"* parameter is set to *Parallel*), this register holds the measured value of external parallel input horizontal resolution. The value is automatically reset at every rising or falling edge of E_VIDEO_PRESENT input signal and is valid after activation of E_VIDEO_VALID flag in interrupt status register.

The maximum value that can be measured by logiCVC-ML, i.e. the size of E_HRES register, is constrained by *"Row stride"* parameter in a way that the number of bits used in E_HRES register is equal to $\log_2$(*Row stride*).

10.2.13 External Input Vertical Resolution - E_VRES

Table 10.25: E_VRES Register Description

Bits	Address offset	Name	Type	Reset value	Description
15 - 0	0x088	E_VRES	(R)	0x0000	External Input Vertical Resolution Register

If external parallel input interface is enabled *("Use external input"* parameter is set to *Parallel*), this register holds the measured value of external parallel input vertical resolution. The value is automatically reset at every rising or falling edge of E_VIDEO_PRESENT input signal and is valid after assertion of E_VIDEO_VALID flag in interrupt status register.

10.2.14 IP Version Register - IP_VERSION

Table 10.26: IP_VERSION Register Description

Bits	Address offset	Name	Type	Reset value[1)]	Description
31 - 0	0x0F8	IP_VERSION	R	0x00002801	IP Version Register

Value stored in the IP version register holds information about the version of the current logiCVC-ML IP. It is a 22-bit value, while bits 22 to 31 are read as '0'. User cannot write to this register, i.e. its content can only be read.

This value stored in this register is constructed from a set of parameters as outlined in Table 10.27.

Table 10.27: logiCVC-ML IP Version Information

Bits	Name	Type	Reset value	Description
4 - 0	REVISION	R	0b00001	IP Revision (1 - 31)
10 - 5	MINOR_VERSION	R	0b000000	IP Minor Version (0 - 63)
16 - 11	MAJOR_VERSION	R	0b000101	IP Major Version (0 - 63)
18 - 17	LICENSE_TYPE	R	0b00	IP License type (0 - source, 1 - evaluation, 2 - release, 3 - university evaluation)
19	LICENSE_CHECK	R	0	IP License check (0 - no, 1 - yes)
21 - 20	TIME_BEFORE_BRAKE	R	0b00	IP Time before break (0 - infinite, 1 - 1h, 2 - 12h, 3 - 24h)
31 - 22				Not used

1. Value stored in this register depends on logiCVC-ML IP version. In the example above, the value is: 0x00002801, which stands for 5.0.1, source version, no license check and infinite time before brake.

10.2.15 Layer Memory Address Register - Lx_ADDR

Table 10.28: LX_ADDR Register Description

Bits	Address offset	Name	Type	Reset value	Description
31 - 0	0x0100	L0_ADDR	R/W	C_LAYER_0_ADDR	Layer 0 Memory Address
31 - 0	0x0180	L1_ADDR	R/W	C_LAYER_1_ADDR	Layer 1 Memory Address
31 - 0	0x0200	L2_ADDR	R/W	C_LAYER_2_ADDR	Layer 2 Memory Address
31 - 0	0x0280	L3_ADDR	R/W	C_LAYER_3_ADDR	Layer 3 Memory Address
31 - 0	0x0300	L4_ADDR	R/W	C_LAYER_4_ADDR	Layer 4 Memory Address

Layer memory address register defines the starting point in the memory from which logiCVC-ML layer reads the first pixel. The reset value of this register is defined by *"Layer memory address"* parameter.

The register holds an absolute memory address of 32-bits. However, the register is logically divided into two parts, lower and higher address. Increasing the lower address will consequently move the picture shown on the screen to the left. If the lower address is increased so much that the sum of address and width (in bytes) of the layer falls out of the lower address range, i.e. pixel row stride, the logiCVC-ML will start showing data from the start of the lower address, i.e. the beginning of the same line in memory.

Increasing the higher address will move the picture shown on the display up. If the higher address is increased so much that the sum of high address and layer height falls out of the high address range, the logiCVC-ML will start showing data from the start of the higher address, i.e. high address = 0.

The size of the lower and higher address parts, outlined in Table 10.29, depends on logiCVC-ML configuration and is determined from pixel row stride value that is explained in more detail in chapter 5.1 Memory Layout, Figure 5.1 and Table 5.1.

Table 10.29: Layer Memory Address Register Map

Bits	Name	Type	Description
($\log_2$(pixel row stride) - 1) – 0	LOWER_ADDR	R/(W)	Lower Part of Layer Memory address. Can be used only when C_USE_SIZE_POS = 1
31 – $\log_2$(pixel row stride)	HIGHER_ADDR	R/W	Higher Part of Layer Memory address

Lower part of the layer address register can only be used if parameter *"Use layer size and position"* is enabled. If it is disabled, these bits are tied to 0.

The value written in the layer address register has to be a multiple of the number of bytes that one pixel of the corresponding layer occupies. For example, if a layer is configured to be 16bpp and uses layer alpha blending each pixel consists of 2 bytes, which means layer address has to be divisible by two. If a layer is configured to be 24bpp or 30bpp each pixel consists of 4 bytes, which means layer address has to be divisible by four. In other words, logiCVC-ML will ignore one or two lowest bits written in the address register depending on layer configuration.

For more information on how much bytes a pixel consists of, please refer to chapter 5.1 Memory Layout and Table 5.2, Table 5.3 and Table 5.4.

In order to avoid image glitches when changing layer address, size and position registers, it is essential that all new values become active at the same time and inside the video inactive period. Therefore, logiCVC-ML will update these registers only once per frame on vertical sync, i.e. in vertical inactive period. Additionally, in order to update them at the same time the address, size and position registers will be ready for update only after a write is performed into the layer address register.

For example, if user wants to change layer address and size registers it must modify the size registers and then last in the sequence it must perform a write to the layer address register. This will trigger the logiCVC-ML layer registers update procedure that will be executed in the next vertical sync period. When the update is executed, logiCVC-ML will signal the update by asserting the corresponding layer update bit, Lx_REGS_UPDATED, in the interrupt status register described in chapter 10.2.9.

In case the user wants to update layer registers of more than one layer at the same time, user can disable the update of layers registers until all of them have been updated and then enable the update. This is achieved using the DIS_UPDATE bit in the control register, CTRL, described in chapter 10.2.5. This allows that address, size and position registers of all layers are updated at the same time, i.e. at the following vertical sync.

If parameter *"Configure last layer as background"* is enabled, last layer is not fetching data from memory so layer memory address register is not used.

10.2.16 Layer Position Registers - Lx_H_POSITION, Lx_V_POSITION

Table 10.30: Lx_H_POSITION Register Description

Bits	Address offset	Name	Type	Reset value	Description
15 - 0	0x0110	L0_H_POSITION	R/W	0x027F	Layer 0 Horizontal Position
15 - 0	0x0190	L1_H_POSITION	R/W	0x027F	Layer 1 Horizontal Position
15 - 0	0x0210	L2_H_POSITION	R/W	0x027F	Layer 2 Horizontal Position
15 - 0	0x0290	L3_H_POSITION	R/W	0x027F	Layer 3 Horizontal Position

Table 10.31: Lx_V_POSITION Register Description

Bits	Address offset	Name	Type	Reset value	Description
15 - 0	0x0118	L0_V_POSITION	R/W	0x01DF	Layer 0 Vertical Position
15 - 0	0x0198	L1_V_POSITION	R/W	0x01DF	Layer 1 Vertical Position
15 - 0	0x0218	L2_V_POSITION	R/W	0x01DF	Layer 2 Vertical Position
15 - 0	0x0298	L3_V_POSITION	R/W	0x01DF	Layer 3 Vertical Position

When changing horizontal position, the displayed picture will be moved to the right for a number of pixels. That number is specified by writing into the horizontal position register. Pixels that lay between the left side of the display and the adjusted position will be transparent. If not adjusted separately, the horizontal position will be, by default, the same as the horizontal resolution. That means that, if image has to be moved to the right, the number written to the horizontal position register has to be decreased.

When adjusting horizontal position there is one restriction, according to which the number representing the horizontal position has to be equal or greater than the number representing the horizontal size (width).

Table 10.32: Layer Horizontal Position Register Map

Bits	Name	Type	Reset value	Description
0	HOR_POS_0	R/W	1	Layer horizontal position bit 0 [1)]
1	HOR_POS_1	R/W	1	Layer horizontal position bit 1 [1)]
2	HOR_POS_2	R/W	1	Layer horizontal position bit 2 [1)]
3	HOR_POS_3	R/W	1	Layer horizontal position bit 3 [1)]
4	HOR_POS_4	R/W	1	Layer horizontal position bit 4 [1)]
5	HOR_POS_5	R/W	1	Layer horizontal position bit 5 [1)]
6	HOR_POS_6	R/W	1	Layer horizontal position bit 6 [1)]
7	HOR_POS_7	R/W	0	Layer horizontal position bit 7 [1)]
8	HOR_POS_8	R/W	0	Layer horizontal position bit 8 [1)]
9	HOR_POS_9	R/W	1	Layer horizontal position bit 9 [1)]
10	HOR_POS_10	R/W	0	Layer horizontal position bit 10 [1)]
15 - 11			0	Not used

1. Maximum value of horizontal layer position is constrained by configured logiCVC-ML resolution. As the maximum horizontal resolution is constrained by *"Row stride"* parameter, the number of bits used from layer position registers is equal to $\log_2$(*Row stride*).

When changing vertical position, the displayed picture will be moved down for a number of lines. That number is specified by writing into the vertical position register. Lines that lay between the top of the display and the adjusted position will be transparent. If not adjusted separately, the vertical position will be, by default, the same as the vertical resolution. That means that, if image has to be moved down, the number written to the vertical position register has to be decreased.

When adjusting vertical position user has to take care that the number representing the vertical position is equal or greater than the number representing the vertical size (height).

Table 10.33: Layer Vertical Position Register Map

Bits	Name	Type	Reset value	Description
0	VER_POS_0	R/W	1	Layer vertical position bit 0 [1)]
1	VER_POS_1	R/W	1	Layer vertical position bit 1 [1)]
2	VER_POS_2	R/W	1	Layer vertical position bit 2 [1)]
3	VER_POS_3	R/W	1	Layer vertical position bit 3 [1)]
4	VER_POS_4	R/W	1	Layer vertical position bit 4 [1)]
5	VER_POS_5	R/W	0	Layer vertical position bit 5 [1)]
6	VER_POS_6	R/W	1	Layer vertical position bit 6 [1)]

Bits	Name	Type	Reset value	Description
7	VER_POS_7	R/W	1	Layer vertical position bit 7 [1)]
8	VER_POS_8	R/W	1	Layer vertical position bit 8 [1)]
9	VER_POS_9	R/W	0	Layer vertical position bit 9 [1)]
10	VER_POS_10	R/W	0	Layer vertical position bit 10 [1)]
15 - 11			0	Not used

1. Maximum value of vertical layer position is constrained by configured logiCVC-ML resolution.

To use layer position, *"Use layer size and position"* parameter has to be enabled.

In order to avoid image glitches when changing layer address, size and position registers, it is essential that all new values become active at the same time and inside the video inactive period. Therefore, logiCVC-ML will update these registers only once per frame on vertical sync, i.e. in vertical inactive period. Additionally, in order to update them at the same time the address, size and position registers will be ready for update only after a write is performed into the layer address register.

For example, if user wants to change the layer position registers it must modify the position registers and then last in the sequence it must perform a write to the layer address register. This will trigger the logiCVC-ML layer registers update procedure that will be executed in the next vertical sync period. When the update is executed, logiCVC-ML will signal the update by asserting the corresponding layer update bit, Lx_REGS_UPDATED, in the interrupt status register described in chapter 10.2.9.

In case the user wants to update layer registers of more than one layer at the same time, user can disable the update of layers registers until all of them have been updated and then enable the update. This is achieved using the DIS_UPDATE bit in the control register, CTRL, described in chapter 10.2.5. This allows that address, size and position registers of all layers are updated at the same time, i.e. at the following vertical sync.

Please note that the last layer's position must not be changed because there is nothing to show beneath it in the places that become transparent. That is the reason why there is no layer position register for layer 4 (it is the last possible layer).

10.2.17 Layer Size Registers - Lx_WIDTH, Lx_HEIGHT

Table 10.34: Lx_WIDTH Register Description

Bits	Address offset	Name	Type	Reset value	Description
15 - 0	0x0120	L0_WIDTH	R/W	0x027F	Layer 0 Width
15 - 0	0x01A0	L1_WIDTH	R/W	0x027F	Layer 1 Width
15 - 0	0x0220	L2_WIDTH	R/W	0x027F	Layer 2 Width
15 - 0	0x02A0	L3_WIDTH	R/W	0x027F	Layer 3 Width

Table 10.35: Lx_HEIGHT Register Description

Bits	Address offset	Name	Type	Reset value	Description
15 - 0	0x0128	L0_HEIGHT	R/W	0x01DF	Layer 0 Height
15 - 0	0x01A8	L1_HEIGHT	R/W	0x01DF	Layer 1 Height
15 - 0	0x0228	L2_HEIGHT	R/W	0x01DF	Layer 2 Height
15 - 0	0x02A8	L3_HEIGHT	R/W	0x01DF	Layer 3 Height

Horizontal size (width) defines how much pixels within the line will be visible. The rest of the pixels will be transparent. If not adjusted separately, the horizontal size will be, by default, the same as the horizontal resolution after first write to the horizontal resolution register. The layer's horizontal size equals to the value written in this register incremented by one in pixel.

Table 10.36: Layer Width Register Map

Bits	Name	Type	Reset value	Description
0	WIDTH_0	R/W	1	Layer width bit 0 1)
1	WIDTH_1	R/W	1	Layer width bit 1 1)
2	WIDTH_2	R/W	1	Layer width bit 2 1)
3	WIDTH_3	R/W	1	Layer width bit 3 1)
4	WIDTH_4	R/W	1	Layer width bit 4 1)
5	WIDTH_5	R/W	1	Layer width bit 5 1)
6	WIDTH_6	R/W	1	Layer width bit 6 1)
7	WIDTH_7	R/W	0	Layer width bit 7 1)
8	WIDTH_8	R/W	0	Layer width bit 8 1)
9	WIDTH_9	R/W	1	Layer width bit 9 1)
10	WIDTH_10	R/W	0	Layer width bit 10 1)
15 - 11				Not used

1. Maximum value of width register is constrained by configured logiCVC-ML resolution. As the maximum resolution is constrained by *"Row stride"* parameter, the number of bits used from width register is equal to $\log_2$(*Row stride*).

Vertical size defines how much lines within the frame will be visible. The rest of the lines will be transparent. If not adjusted separately, the vertical size will be, by default, the same as the vertical resolution after the first write to the vertical resolution register. The layer's vertical size equals to the value written in this register incremented by one in lines.

Table 10.37: Layer Height Register Map

Bits	Name	Type	Reset value	Description
0	HEIGHT_0	R/W	1	Layer height bit 0 1)
1	HEIGHT_1	R/W	1	Layer height bit 1 1)
2	HEIGHT_2	R/W	1	Layer height bit 2 1)
3	HEIGHT_3	R/W	1	Layer height bit 3 1)
4	HEIGHT_4	R/W	1	Layer height bit 4 1)
5	HEIGHT_5	R/W	0	Layer height bit 5 1)
6	HEIGHT_6	R/W	1	Layer height bit 6 1)
7	HEIGHT_7	R/W	1	Layer height bit 7 1)
8	HEIGHT_8	R/W	1	Layer height bit 8 1)
9	HEIGHT_9	R/W	0	Layer height bit 9 1)
10	HEIGHT_10	R/W	0	Layer height bit 10 1)
15 - 11				Not used

1. Maximum value of height register is constrained by configured logiCVC-ML resolution.

To use layer position, *"Use layer size and position"* parameter has to be enabled.

In order to avoid image glitches when changing layer address, size and position registers, it is essential that all new values become active at the same time and inside the video inactive period. Therefore, logiCVC-ML will update these registers only once per frame on vertical sync, i.e. in vertical inactive period. Additionally, in order to update them at the same time the address, size and position registers will be ready for update only after a write is performed into the layer address register.

For example, if user wants to change the layer size registers it must modify the size registers and then last in the sequence it must perform a write to the layer address register. This will trigger the logiCVC-ML layer registers update procedure that will be executed in the next vertical sync period. When the update is executed, logiCVC-ML will signal the update by asserting the corresponding layer update bit, Lx_REGS_UPDATED, in the interrupt status register described in chapter 10.2.9.

In case the user wants to update layer registers of more than one layer at the same time, user can disable the update of layers registers until all of them have been updated and then enable the update. This is achieved using the DIS_UPDATE bit in the control register, CTRL, described in chapter 10.2.5. This allows that address, size and position registers of all layers are updated at the same time, i.e. at the following vertical sync.

Please note that the last layer's size must not be changed because there is nothing to show beneath it in the places that become transparent. That is the reason why there is no layer size register for layer 4 (it is the last possible layer).

10.2.18 Layer Alpha Register - Lx_ALPHA

Table 10.38: Lx_ALPHA Register Description

Bits	Address offset	Name	Type	Reset value	Description
9 - 0	0x0130	L0_ALPHA	R/W	0x3FF	Layer 0 Alpha Value
9 - 0	0x01B0	L1_ALPHA	R/W	0x3FF	Layer 1 Alpha Value
9 - 0	0x0230	L2_ALPHA	R/W	0x3FF	Layer 2 Alpha Value
9 - 0	0x02B0	L3_ALPHA	R/W	0x3FF	Layer 3 Alpha Value

Layer alpha register contains alpha factor for the specific layer if that layer is configured to use layer alpha blending mode (*"Alpha blending mode"* parameter set to *layer*).

Four color depth modes are supported: 8bpp, 16bpp, 24bpp and 30bpp. 30bpp color depth mode requires use of all 10 bits. 24bpp color depth mode requires use of 8 bits. In 16bpp RGB layer mode (RGB565), least significant 5 or 6 bits are used (bits 4-0 or bits 5-0). Red and blue color use 5 bits, while green uses 6 bits. In 16bpp YCbCr layer mode (4:2:2), 8 bits are used as each component is 8 bit wide. 8bpp mode uses least significant 3 or 2 bits (bits 2-0 or bits 1-0). Red and green color use 3 bits, while blue uses 2 bits.

For more on alpha blending please refer to chapter 9.3.2 Alpha Blending.

Table 10.39: Layer Alpha Register Map

Bits	Color Depth					Description
	8bpp	16bpp		24bpp	30bpp	
		RGB	YCbCr			
0	ALPHA_0	ALPHA_0	ALPHA_0	ALPHA_0	ALPHA_0	Layer alpha factor bit 0
1	ALPHA_1	ALPHA_1	ALPHA_1	ALPHA_1	ALPHA_1	Layer alpha factor bit 1
2	ALPHA_2	ALPHA_2	ALPHA_2	ALPHA_2	ALPHA_2	Layer alpha factor bit 2
3		ALPHA_3	ALPHA_3	ALPHA_3	ALPHA_3	Layer alpha factor bit 3
4		ALPHA_4	ALPHA_4	ALPHA_4	ALPHA_4	Layer alpha factor bit 4
5		ALPHA_5	ALPHA_5	ALPHA_5	ALPHA_5	Layer alpha factor bit 5
6			ALPHA_6	ALPHA_6	ALPHA_6	Layer alpha factor bit 6
7			ALPHA_7	ALPHA_7	ALPHA_7	Layer alpha factor bit 7
8					ALPHA_8	Layer alpha factor bit 8
9					ALPHA_9	Layer alpha factor bit 9

1. Last layer's alpha factor cannot be changed (fixed to '1') because there is nothing to show beneath the last layer. That is the reason why there is no layer alpha factor register for layer 4 (it is the last possible layer).

10.2.19 Layer Control Register - Lx_CTRL

Table 10.40: Lx_CTRL Register Description

Bits	Address offset	Name	Type	Reset value	Description
7 - 0	0x0138	L0_CTRL	R/W	0x0001	Layer 0 Control Register
7 - 0	0x01B8	L1_CTRL	R/W	0x0000	Layer 1 Control Register
7 - 0	0x0238	L2_CTRL	R/W	0x0000	Layer 2 Control Register
7 - 0	0x02B8	L3_CTRL	R/W	0x0000	Layer 3 Control Register
7 - 0	0x0338	L4_CTRL	R/W	0x0000	Layer 4 Control Register

Layer Enable

Layer Control register's LSB is used for enabling/disabling a layer. By disabling it, layer is not fetching data from memory and acts as a transparent layer.

Disable Transparency

DIS_TRANSP bit is used for disabling color key transparency per layer. By setting this bit, logiCVC-ML will not perform a comparison between pixel color and the value stored in the Transparent Color register and pixel will be seen (if alpha value is set appropriately).

External frame buffer enable

EN_EXT_VBUFF_SW bit is used for enabling external switching of frame buffers. This functionality is used when there is an external video source writing to logiCVC-ML layer video buffer. For more information please refer to chapter 5.4 Frame Buffer .

Interlace enable

INTERLACED bit is used to control the memory-reading mode when logiCVC-ML is configured to output ITU656 standard. If set to '1' logiCVC-ML toggles between reading even and odd lines from the memory depending on the current frame it is outputting. When set to '0', i.e. progressive mode, every frame consists of continuous lines from memory. If ITU656 output standard is not used (C_DISPLAY_INTERFACE is not set to 1) this bit is not used.

Pixel format

PIXEL_FORMAT bits are used to define the order of color components inside the pixel. Currently supported formats are outlined in Table 10.42.

Table 10.41: Layer Control Register Map

Bits	Name	Type	Reset value	Description
0	ENABLE	R/W	0 [1)]	Layer ON/OFF bit
1	DIS_TRANSP	R/W	0	Disable Color Transparency bit
2	EN_EXT_VBUFF_SW	R/W	0	Enable external frame buffer switching
3	INTERLACED	R/W	0	Interlaced/progressive memory reading
6 - 4	PIXEL_FORMAT	R/W	0	Pixel format bits
7				Not used

1. After reset, only layer 0 ENABLE bit is set.

Table 10.42: Supported Layer Pixel Formats

PIXEL_FORMAT (2:0)	Layer type		
	RGB	YCbCr 4:4:4	YCbCr 4:2:2
000	ARGB	AYCbCr	CbY_0CrY_1
001	ABGR	ACrCbY	Y_0CbY_1Cr
Not used			

10.2.20 Layer Transparent Color Register - Lx_TRANSPARENT

Table 10.43: Lx_TRANSPARENT Register Description

Bits	Address offset	Name	Type	Reset value	Description
29 - 0	0x0140	L0_TRANSPARENT	R/W	0x000000	Layer 0 Transparent Color
29 - 0	0x01C0	L1_TRANSPARENT	R/W	0x000000	Layer 1 Transparent Color
29 - 0	0x0240	L2_TRANSPARENT	R/W	0x000000	Layer 2 Transparent Color
29 - 0	0x02C0	L3_TRANSPARENT	R/W	0x000000	Layer 3 Transparent Color

Transparent color register holds the color value that will be transparent if it is located in video memory. Therefore, if transparent color is read from the video memory, alpha factor set for this layer (or pixel or color) will be ignored and transformed into 0 (not visible).

Four color depths are supported: 8bpp, 16bpp, 24bpp and 30bpp. 30bpp color depth mode uses 30 bits, 24bpp color depth mode (RGB or YCbCr 4:4:4) requires use of 24 bits. In 16bpp RGB layer mode (RGB565), 16 are used for determining transparent color.

In 16bpp YCbCr layer mode (4:2:2), 24 bits are used because logiCVC-ML performs 4:2:2 to 4:4:4 conversion before searching for transparent pixels.

8bpp RGB mode uses 8 bits.

Table 10.44: Layer Transparent Color Register Map

Bits	Color Depth					Description
	8bpp	16bpp		24bpp	30bpp	
		RGB	YCbCr			
0	TRANS_0	TRANS_0	TRANS_0	TRANS_0	TRANS_0	Layer transparent color bit 0
1	TRANS_1	TRANS_1	TRANS_1	TRANS_1	TRANS_1	Layer transparent color bit 1
2	TRANS_2	TRANS_2	TRANS_2	TRANS_2	TRANS_2	Layer transparent color bit 2
3	TRANS_3	TRANS_3	TRANS_3	TRANS_3	TRANS_3	Layer transparent color bit 3
4	TRANS_4	TRANS_4	TRANS_4	TRANS_4	TRANS_4	Layer transparent color bit 4
5	TRANS_5	TRANS_5	TRANS_5	TRANS_5	TRANS_5	Layer transparent color bit 5
6	TRANS_6	TRANS_6	TRANS_6	TRANS_6	TRANS_6	Layer transparent color bit 6

Bits	Color Depth					Description
	8bpp	16bpp		24bpp	30bpp	
		RGB	YCbCr			
7	TRANS_7	TRANS_7	TRANS_7	TRANS_7	TRANS_7	Layer transparent color bit 7
8		TRANS_8	TRANS_8	TRANS_8	TRANS_8	Layer transparent color bit 8
9		TRANS_9	TRANS_9	TRANS_9	TRANS_9	Layer transparent color bit 9
10		TRANS_10	TRANS_10	TRANS_10	TRANS_10	Layer transparent color bit 10
11		TRANS_11	TRANS_11	TRANS_11	TRANS_11	Layer transparent color bit 11
12		TRANS_12	TRANS_12	TRANS_12	TRANS_12	Layer transparent color bit 12
13		TRANS_13	TRANS_13	TRANS_13	TRANS_13	Layer transparent color bit 13
14		TRANS_14	TRANS_14	TRANS_14	TRANS_14	Layer transparent color bit 14
15		TRANS_15	TRANS_15	TRANS_15	TRANS_15	Layer transparent color bit 15
16			TRANS_16	TRANS_16	TRANS_16	Layer transparent color bit 16
17			TRANS_17	TRANS_17	TRANS_17	Layer transparent color bit 17
18			TRANS_18	TRANS_18	TRANS_18	Layer transparent color bit 18
19			TRANS_19	TRANS_19	TRANS_19	Layer transparent color bit 19
20			TRANS_20	TRANS_20	TRANS_20	Layer transparent color bit 20
21			TRANS_21	TRANS_21	TRANS_21	Layer transparent color bit 21
22			TRANS_22	TRANS_22	TRANS_22	Layer transparent color bit 22
23			TRANS_23	TRANS_23	TRANS_23	Layer transparent color bit 23
24					TRANS_24	Layer transparent color bit 24
25					TRANS_25	Layer transparent color bit 25
26					TRANS_26	Layer transparent color bit 26
27					TRANS_27	Layer transparent color bit 27
28					TRANS_28	Layer transparent color bit 28
29					TRANS_29	Layer transparent color bit 29

1. Last layer's transparent color cannot be set because there is nothing to show beneath the last layer. That is the reason why there is no layer transparent color register for layer 4 (it is the last possible layer).
2. Each layers' color transparency functionality can be disabled by using DIS_TRANSP bit in corresponding Layer Control register.

11 INTEGRATION

This chapter describes how to synthesize and implement logiCVC-ML HDL code and integrate logiCVC-ML in a host design.

The logiCVC-ML is delivered as encrypted source code module, therefore synthesis is always required.

11.1 Synthesis

The logiCVC-ML source code has configurable number of layers, memory data bus width, pixel format and many more parameters. Prior to synthesis, please select logiCVC-ML configuration parameters by using Vivado implementation tools. Please refer to chapter 3 CUSTOMIZATION for more information.

In order to reduce FPGA resources utilization, user can do the following steps ordered by importance (first on the list has the greatest impact on resource utilization):

- Reduce the number of logiCVC-ML layers (*"Number of layers"* parameter)
- Turn off layer size and position (*"Use layer size and position"* parameter)
- Turn off readable logiCVC-ML registers (*"Readable logiCVC registers"* parameter)
- Remove or reduce parallel pixel processing (*"Pixels per clock"* parameter)
- Reduce row stride value (*"Row stride"* parameter)
- Reduce layer data width *("Layer data width"* parameter)
- Use the same color space for all layers to remove multiple color space conversions (*"Layer type"* parameter)
- Reduce AXI4 data bus width to reduce BRAM utilization (*"AXI4 interface data bus width"* parameter)
- Reduce FIFO size to reduce BRAM utilization (*"FIFO size multiplication factor"* parameter)

11.2 Implementation

Recommended logiCVC-ML FPGA implementation options:

- Effort: high
- Pack I/O Registers/Latches into IOBs for: Inputs and Outputs

11.3 IO Interface

The logiCVC-ML - Compact Multilayer Video Controller is a fully synchronous digital design. Depending on the configuration used, logiCVC-ML uses minimum three separated clock signals and maximum six. The following clock signals should be connected to the logiCVC-ML clock inputs: VCLK for video, M_AXI_ACLK for AXI4 memory bus access, S_AXI_ACLK for AXI4-Lite registers access, LVDS_CLK and LVDS_CLKN for LVDS output interface and ITU_CLK_IN for ITU656 output interface. All signals should have stable clock sources. Preferred method of clock generation is using Xilinx Phase Locked Loops (PLLs) or Mixed-Mode Clock Managers (MMCMs).

The VCLK clock signal controls most of the circuits inside the logiCVC-ML core. If external parallel input is used (*"Use external input"* set to *Parallel*) and E_VIDEO_PRESENT is active, E_VCLK is used instead of VCLK when *"Use BUFGMUX for external parallel input"* parameter is enabled. When *"Use BUFGMUX for external parallel input"* parameter is disabled, user has to instance BUFGMUX manually and provide VCLK according to E_VIDEO_PRESENT. This is useful when LVDS or camera link interface is used.

Usually, the AXI4 clock frequency is higher than VCLK frequency. The sustained data rate between memory and logiCVC-ML should be assured by selecting an appropriate AXI4 clock frequency and memory bandwidth. For more information, please refer to chapter 5.5 Memory bandwidth requirements.

When ITU656 output standard is used, user must provide ITU_CLK_IN frequency of 27 MHz, but additionally assure that the VCLK is synchronous to ITU_CLK_IN and has the frequency of 13.5 MHz. This is required because of the ITU656 standard.

All internal logiCVC-ML registers are controlled by S_AXI_ACLK AXI4-Lite clock signal.

11.3.1 Input Signals

The setup and hold times for FPGA internal FFs are inherently fulfilled by the use of the Xilinx FPGA implementation tools. The clock skew should be presumed to 0.

Writing and reading to logiCVC-ML registers is controlled by the AXI4-Lite bus. Please consult Xilinx AXI Reference Guide, UG761, on AXI4-Lite timing requirements.

The Video memory access is performed via AXI4 interface. Please consult Xilinx AXI Reference Guide, UG761, on AXI4-Lite timing requirements.

11.3.2 Parallel Data and Control Signals

When using parallel pixel data interface, all display signals should be configured to use IOB output FFs. In this case, the HSYNC, VSYNC, BLANK, and D_PIX[N:0] signals are synchronous with PIX_CLK have a minimal negligible skew between them and PIX_CLK.

In order for Vivado to place the output FFs into the IOB, user has to manually instruct the tools to do so. Please make sure that you modify your XDC to include the following constraint:

```
set_property IOB TRUE [get_ports {D_PIX_*}] ## rename the signal name
accordingly
```

11.4 Timing Constraints

The logiCVC-ML IP core requires specifying the clock constraint on all used clocks. The following timing specifications should be appended to the host design constraints (XDC).

The lowest clock period, i.e. the highest VCLK frequency, should be supplemented in line:

```
create_clock -period <PERIOD OF vclk> -name vclk [get_nets <CLOCK NET NAME>]
```

Register clock (AXI4-Lite) period have to be supplemented in lines:

```
create_clock -period <PERIOD OF s_axi_aclk> -name s_axi_aclk [get_nets <CLOCK NET NAME>]
```

Memory interface clock (AXI4) period have to be supplemented in lines:

```
create_clock -period <PERIOD OF m_axi_aclk> -name m_axi_aclk [get_nets <CLOCK NET NAME>]
```

If additional clocks are used by logiCVC-ML (itu_656_lvds_clk, …) those clocks also have to be defined in a similar way as described above.

Additionally, as logiCVC-ML internally handles clock domain crossings, timing analysis of nets between these clocks can be ignored by specifying false path constraints:

```
set_false_path -from [get_clocks m_axi_aclk] -to [get_clocks s_axi_aclk]
set_false_path -from [get_clocks s_axi_aclk] -to [get_clocks m_axi_aclk]

set_false_path -from [get_clocks s_axi_aclk] -to [get_clocks vclk]
set_false_path -from [get_clocks vclk] -to [get_clocks s_axi_aclk]

set_false_path -from [get_clocks vclk] -to [get_clocks m_axi_aclk]
set_false_path -from [get_clocks m_axi_aclk] -to [get_clocks vclk]
```

Please note that these false path constraints will ignore clock crossing for the complete design which might erroneously include paths of additional design logic outside of logiCVC-ML IP that should not be ignored in timing analysis. For these cases additional options like `–from`, `-to`, `-through` can be used to limit the above false path constraint to include only logiCVC-ML internal paths.
For more information on using constraints please refer to Vivado Design Suite User Guide, Using Constraints, UG903.

12 REVISION HISTORY

Version	Date	Author	Approved by	Note
1.01.a	28.07.2006.	J. Ivanović	J. Ivanović	Initial Xylon release
1.01.b	09.10.2006.	J. Ivanović	J. Ivanović	Same as v1.00.a
1.01.c	17.11.2006.	J. Ivanović	J. Ivanović	Added layer control register
1.02.a	21.04.2007.	J. Ivanović	J. Ivanović	New register widths, interrupt interface, ITU656 support, different buffer switching mechanism…
1.03.a	07.01.2008.	Z. Šafaržik, J. Ivanović	J. Ivanović	Added triple buffer, row stride, resolution up to 2048x2048, XMB port, USE_SIZE_POS, configurable memory burst Corrected LVDS pixel order
1.04.a	07.04.2008.	J. Ivanović	J. Ivanović	PLBv4.6 video memory interface: user selectable OPB or PLBv4.6 Slave register interface Changed some VHDL generic parameter names External control of triple buffering
1.04.c	02.10.2008.	J. Ivanović	J. Ivanović	Added 12-bit multiplexed output mode External RGB input.
1.04.e	27.02.2009.	J. Ivanović	J. Ivanović	Only the name is changed to support new hw version
1.04.f	23.03.2009.	J. Ivanović	J. Ivanović	C_READABLE_REGS, interlaced/progressive ITU656 memory reading, update on ext video input synchronization, maximum addressable memory range
1.04.g	21.09.2009.	Z. Šafaržik	J. Ivanović	Little endianness support added (C_LITTLE_ENDIAN), fixed memory and pixel layout tables
1.04.g	30.09.2009.	R. Končurat	J. Ivanović	On page 41, in table 10.1 logiCVC-ML register map, correction of address offset for Layer0 Color Look-Up table: 0x1000 – 0x17FF
1.04.h	22.10.2009.	Z. Šafaržik	J. Ivanović	No changes, only renamed according to core version
1.05.a	27.10.2009.	T. Anić	J. Ivanović	Added support for Spartan6 FPGA specific components (C_USE_IO_HW_SERIALIZER) Updated figures 3.1 and 12.1

Version	Date	Author	Approved by	Note
1.05.b	21.12.2009.	R. Končurat	J. Ivanović	Spartan6 and Virtex6 added to C_FAMILY Parameters C_USE_LVDS and C_USE_ITU656 removed C_DISPLAY_INTERFACE parameter added to replace them both and to add another display interface option: camera link Version parameters added: C_IP_LICENSE_TYPE (IP encryption type); C_IP_MAJOR_REVISION; C_IP_MINOR_REVISION; C_IP_PATCH_LEVEL In addition to these parameters there is a register named IP_VERSION that contains a value representing the IP version ODDR2 components instantiated for Spartan3a, Spartan3e and Spartan6 ODDR components instantiated for Virtex4, Virtex5 and Virtex6
1.05.b	21.12.2009.	K. Mudrovčić	J. Ivanović	User's manual corrections
1.06.a	11.01.2010.	R. Končurat	J. Ivanović	User's manual corrections Bug fix in parallel output interface for Virtex4, Virtex5 and Virtex6 families (ODDR components) Bug fix in external RGB data and video buffer width mismatch
1.06.a	11.01.2010.	A. Jukić	J. Ivanović	New alpha blender New generic parameters added: C_USE_XTREME_DSP and C_USE_MULTIPLIER
1.06.b	13.01.2010.	J. Ivanović	J. Ivanović	Added support for 128-bit XMB/PLB bus Added support for 32 and 64-burst length on 64 and 128 bit PLB bus
1.06.c	02.03.2010.	Z. Šafaržik	J. Ivanović	
1.06.c	19.3.2010.	J. Marjanović	J. Ivanović	Added generic for usage of BUFGMUX for switching vclk to e_vclk Extended CTRL register for inverting polarity of external video control signals
1.06.c	25.3.2010.	A. Cazin	J. Ivanović	Added support for pipelined memory access
1.06.c	10.4.2010.	K. Mudrovčić	J. Ivanović	User's manual corrections
2.00.a	16.9.2010.	J. Marjanović	J. Ivanović	Added support for AXI interfaces

Version	Date	Author	Approved by	Note
2.01.a	14.03.2011.	J. Ivanović	J. Ivanović	Added DVI interface support, removed camera link 3bit interface, added bit order picture for camera link, corrected bit ordering in register descriptions (x - 0)
2.04.a	08.02.2012.	J. Marjanović	J. Ivanović	Added XCOLOR dithering module, added support for 7-series devices including LVDS and instructions for clocking, removed Serialized Blender Version parameters added: C_IP_LICENSE_CHECK; C_IP_TIME_BEFORE_BREAK; IP_VERSION register updated
2.05.a	21.03.2012.	M. Mrak	J. Ivanović	Added support for YCbCr 4:4:4 and YCbCr 4:2:2 display output Removed C_LITTLE_ENDIAN and C_REGS_LITTLE_ENDIAN generic and added C_MEM_BYTE_SWAP and C_MEM_LITTLE_ENDIAN and C_REG_BYTE_SWAP DISPLAY_INTERFACE chapter reorganized 12*2 D_PIX interface and vclk2 description, C_USE_VCLK2 generic added User's manual corrections
2.05.b	16.04.2012.	R. Končurat	J. Ivanović	Only the name is changed to support new hw version Generic parameter C_IP_TIME_BEFORE_BREAK modified to support release edition (0 = infinite, 1 = 1h, 2 = 12h) User's manual corrections
2.05.c	03.05.2012.	J. Ivanović	J. Ivanović	Small corrections made in Memory Layout chapter regarding endianess YCbCr interface figure corrected to falling active clock edge Added hyperlinks to register map table

Version	Date	Author	Approved by	Note
3.00.a	17.09.2012.	J. Ivanović	J. Ivanović	Added C_DISPLAY_COLOR_SPACE generic that determines the output interface color space Added support for YCbCr 444 and 422 layers (C_LAYER_X_TYPE) Added support for two alpha layers (alpha plane) so now layers 1 and 3 can be configured as alpha layers (C_LAYER_X_TYPE. Added support for different byte ordering inside pixels (RGB, BGR, YCbCr, CrCbY, ...) Added global reset input. Added C_INCREASE_FIFO generic that allows FIFO to be increased in size up to 8 times Increased maximal layer vertical offset from 2048 to 4096 Removed functionality of enabling/disabling pixclk during hsync (ctrl reg(9)) External "RGB input" name changed to "parallel input" as it can now also be in YCbCr format Added support for university evaluation license
3.01.a	07.03.2013.	R. Končurat, J. Ivanović	J. Ivanović	DTYPE register width corrected to 8 bits Reset (default) value of layer horizontal position registers is 0x027F Reset (default) value of layer vertical position registers is 0x01D Reset (default) value of layer horizontal size (width) registers is 0x027F Reset (default) value of layer vertical size (height) registers is 0x01D Added chapter 4.1.1 Layer memory address and range. Added support for DVI in 7 Series Removed vclksel and vcdivsel output signals and bits in control register and added gpi and gpo signals and bits. mclk is now used only for XMB bus. Corrected clock requirements for LVDS output standard in chapter 4 CLOCK AND RESET
3.02.a	03.05.2013.	J. Ivanović	J. Ivanović	HSYNC and VSYNC are now edge aligned so parallel interface timing figures are updated accordingly.

Version	Date	Author	Approved by	Note
3.02.b	23.01.2014.	J. Ivanović	J. Ivanović	Only the name is changed to support new hw version.
4.00.a	01.02.2014.	J. Ivanović	J. Ivanović	Added support for programmable layer address using layer address register. Removed layer offset and vbuff select registers. Removed C_LAYER_OFFSET, C_VMEM_BASEADDR and C_VMEM_HIGHADDR generics. Added C_LAYER_ADDR generics. Changed layer registers update trigger from vertical position register write to layer address register write. Added disable register update bit to control register. Added FIFO underrun bit to interrupt status register. Updated chapters 3 and 4.
4.01.a	21.05.2014.	A. Bogdanić	J. Ivanović	Added current CVC reading buffer output port. Added C_INTERCONNECT_M_AXI_READ_ISSUING parameter to .mpd file. Renamed v_en output port to en_v.
4.01.b	24.07.2014.	J. Ivanović	J. Ivanović	Only the name is changed to support new hw version.
4.1.2	02.10.2014.	R. Končurat	R. Končurat	Documentation version numbering changed according to IP core for Vivado (IP-XACT package). Patch level is replaced by IP core revision tag.
4.2.1	14.11.2014.	A. Samaržija	J. Ivanović	Added generic parameter C_USE_EMB_SYNC Added description of embedded timing signals within the video stream of parallel video interface in chapter 7.1.

Version	Date	Author	Approved by	Note
5.0.1	03.04.2015.	J. Ivanović	J. Ivanović	Added support for 10 bit per color. Increased resolution support to 8192x8192. Added parallel pixel processing with "Pixels per clock" parameter. AXI4-Stream input and output interface support. Added 256-bit AXI4 data interface support. Increased porch registers to 10 bits. Removed support for XPS (ISE) Design Suite. Removed PLB, XMB and OPB interfaces. Added chapter 3 CUSTOMIZATION. Removed "Display enhancement functions" chapter and rearranged its content. Changed chapter 11.4 Timing Constraints to support XDC format.