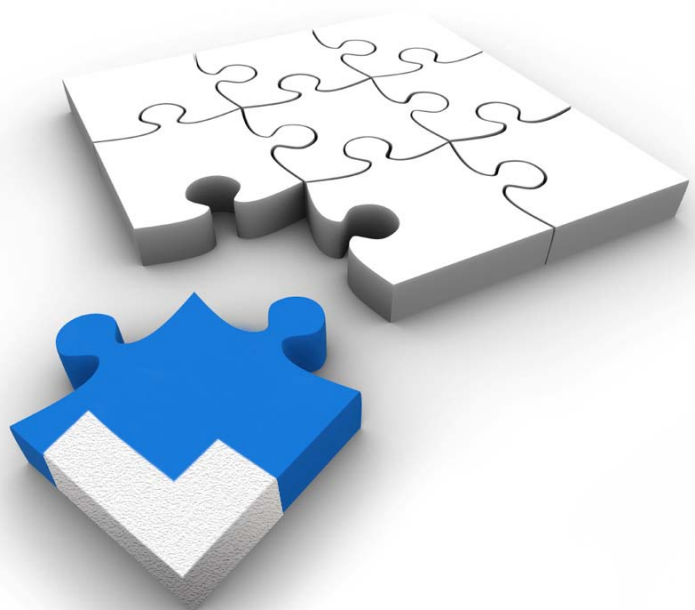


# logiCVC-ML

## *User's Manual*

**Version: 2.04.a**

logiCVC-ML\_hum\_v2.04.a.doc





All rights reserved. This manual may not be reproduced or utilized without the prior written permission issued by Xylon.

Copyright © Xylon d.o.o. logicBRICKS™ is a registered Xylon trademark.

All other trademarks and registered trademarks are the property of their respective owners.

This publication has been carefully checked for accuracy. However, Xylon does not assume any responsibility for the contents or use of any product described herein. Xylon reserves the right to make any changes to product without further notice. Our customers should ensure to take appropriate action so that their use of our products does not infringe upon any patents.

<b>1</b>	<b>INTRODUCTION.....</b>	<b>6</b>
1.1	GENERAL DESCRIPTION.....	6
1.2	FEATURES.....	7
<b>2</b>	<b>IP CORE ARCHITECTURE.....</b>	<b>8</b>
2.1	BLOCK SCHEMATIC.....	8
2.1.1	Video Memory Access Block .....	9
2.1.2	Sync Generator .....	9
2.1.3	Multilayer Alpha Blender.....	9
2.1.4	Registers .....	9
2.2	PINOUT .....	10
2.3	LOGICVC-ML PARAMETERS.....	12
2.4	LOGICVC-ML PORT AND PARAMETER DEPENDENCIES .....	17
<b>3</b>	<b>CLOCK SIGNALS.....</b>	<b>19</b>
<b>4</b>	<b>MEMORY INTERFACE.....</b>	<b>21</b>
4.1	PIXEL ROW STRIDE .....	21
4.2	MEMORY LAYOUT .....	24
4.3	MEMORY BANDWIDTH REQUIREMENTS .....	29
<b>5</b>	<b>CPU INTERFACE .....</b>	<b>32</b>
5.1	REGISTER INTERFACE.....	32
5.2	INTERRUPT INTERFACE .....	32
<b>6</b>	<b>DISPLAY INTERFACE.....</b>	<b>34</b>
6.1	PARALLEL PIXEL DATA INTERFACE .....	34
6.2	LVDS INTERFACE.....	35
6.3	CAMERA LINK INTERFACE.....	36
6.4	VIDEO INTERFACE (ITU656).....	36
6.5	DIGITAL VIDEO INTERFACE (DVI).....	37
6.6	FLAT PANEL DISPLAYS.....	38
6.7	CRT DISPLAYS.....	38
6.8	PIXEL CLOCK .....	39
<b>7</b>	<b>DITHERING MODULE .....</b>	<b>40</b>
<b>8</b>	<b>EXTERNAL RGB INPUT INTERFACE .....</b>	<b>41</b>
<b>9</b>	<b>DISPLAY ENHANCEMENT FUNCTIONS .....</b>	<b>43</b>
9.1	VERTICAL AND HORIZONTAL CENTRING .....	43
9.2	PROGRAMMABLE OUTPUTS.....	43
9.3	POWER SEQUENCING.....	44
9.4	INVERSE VIDEO .....	44

<b>10</b>	<b>MULTILAYER SUPPORT .....</b>	<b>45</b>
10.1	MEMORY ACCESS ARBITER .....	45
10.2	LAYER BLOCK.....	46
10.2.1	Memory Address Generator.....	46
10.2.2	Pixel Data FIFO .....	47
10.2.3	Colour Look-Up Table.....	47
10.3	LAYER MIXER .....	48
10.3.1	Transparency .....	49
10.3.2	Alpha Blending.....	49
10.3.3	Layer Streams Synchronizations .....	51
<b>11</b>	<b>REGISTERS .....</b>	<b>53</b>
11.1	REGISTER MAP .....	53
11.2	REGISTER DESCRIPTION .....	56
11.2.1	Horizontal Sync and Porches - HSY_FP, HSY, HSY_BP.....	56
11.2.2	Horizontal Resolution - H_RES .....	57
11.2.3	Vertical Sync and Porches - VSY_FP, VSY, VSY_BP .....	57
11.2.4	Vertical Resolution - V_RES .....	59
11.2.5	Display Control Register – CTRL .....	59
11.2.6	Display Type Register - DTYPE .....	61
11.2.7	Background Colour Register – BACKGROUND.....	61
11.2.8	Video Buffer Select Register - VBUFF_SEL .....	62
11.2.9	CLUT Select Register - CLUT_SEL .....	63
11.2.10	Interrupt Status Register – INT .....	64
11.2.11	Interrupt Mask Register - INT_MASK .....	65
11.2.12	Power Control Register – PWRCTRL .....	66
11.2.13	External Input Horizontal Resolution - E_HRES .....	67
11.2.14	External Input Vertical Resolution - E_VRES.....	67
11.2.15	IP Version Register - IP_VERSION.....	68
11.2.16	Layer Memory Offset Registers - LX_H_OFFSET, LX_V_OFFSET.....	68
11.2.17	Layer Position Registers - LX_H_POSITION, LX_V_POSITION.....	69
11.2.18	Layer Size Registers - LX_WIDTH, LX_HEIGHT .....	71
11.2.19	Layer Alpha Register - LX_ALPHA .....	73
11.2.20	Layer Control Register - LX_CTRL .....	74
11.2.21	Transparent Colour Register - LX_TRANSPARENT .....	75
<b>12</b>	<b>INTEGRATION.....</b>	<b>77</b>
12.1	SYNTHESIS .....	77
12.2	IMPLEMENTATION.....	77
12.3	INTEGRATION.....	77
12.4	IO INTERFACE DESCRIPTION.....	78
12.4.1	Input Signals .....	79

12.4.2	Control and Data Display Signals .....	79
12.5	TIMING CONSTRAINTS .....	79

## 1 INTRODUCTION

The logiCVC-ML - Compact Video Controller with Multilayer support is a graphic video controller optimized for Xilinx FPGA. logiCVC-ML rich feature set is optimal for embedded electronic devices. It provides all the necessary control signals to interface directly with flat LCD and other flat panel displays. A wide variety of LCD display types are supported. logiCVC-ML compact size, low slice count utilization can be additionally decreased through feature set configuration by VHDL 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 alpha blending, transparency, hardware cursors and fast scrolling by using low CPU processing power. Optional processing functions, like bit-block unit, generators or frame grabbing can easily be added through integration of other graphic logicBRICKS™ IP cores.

To achieve portability of the core source code across various Xilinx FPGA families, all IO blocks, clock drivers, and family-specific circuits are marked for easy replacement. The logiCVC-ML core is supplied with a test-bench. The test-bench simulates all other required hardware modules: clock generation, whole memory subsystem and register interface. The logiCVC-ML can easily be re-initialized for different video displays and its response can simply be checked against the video display's specifications.

Apart from the test bench, there are a number of application notes, reference designs, evaluation boards, and software drivers available upon request. All these simplify integration of the logiCVC-ML with your design, shortening time to the market and increasing your market window.

### 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. 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. Additionally, Digital Video Interface (DVI) compliant displays can be controlled by using the appropriate drivers.

The interface to the frame buffer, or the video memory, is targeted for the SDRAM (SDR or DDR) or SRAM implementation. For easier system integration, logiCVC-ML uses Xilinx (IBM) Coreconnect PLBv4.6, OPB and AXI (AMBA) buses and a special Xylon Memory Bus (XMB). The logiCVC-ML requires relatively low memory bandwidth; it 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 Spartan™-3, Spartan™-3A, Spartan™-3E, Spartan™-6, Virtex™-4, Virtex™-5, Virtex™-6, Virtex™-7 and Kintex™-7 Xilinx FPGAs
- Supports LCD and CRT displays (easily tailored for special display types)
- Supports resolutions from 64x1 up to 2048x2048
- Supports up to 5 layers, last configurable as background colour
- Configurable layer size and position
- Alpha blending and Colour keyed transparency
- Pixel, Layer or Colour lookup table alpha blending mode
- Packed pixel layer memory organization – pixel colour depth 8bpp (Colour look up table), 16bpp Hicolour RGB 5-6-5 and Truecolour 24bpp
- Configurable PLBv4.6, XMB or AXI4 memory interface data width (32, 64 or 128)
- Programmable layer memory base address and stride
- Simple programming due to small number of control registers
- Support for multiple output formats:
  - Parallel display data bus: 12x2-bit, 15-bit, 16-bit, 18-bit or 24-bit
  - 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
- HW cursors
- Supports synchronization to external parallel input
- Versatile and programmable sync signals timing
- Double/triple buffering enables flicker-free reproduction
- Tested with many popular displays
- Display power-on sequencing control signals
- Free Xilinx Platform Studio reference design including demo software application
- Parametrical VHDL design that allows tuning of slice consumption and features set
- Prepared for Xilinx Platform Studio (XPS) and the EDK
- Additional options:
  - logiMEM – Flexible memory controller IP
  - logiBITBLT – BITMAP graphic accelerator
  - logiWIN – Versatile Video input

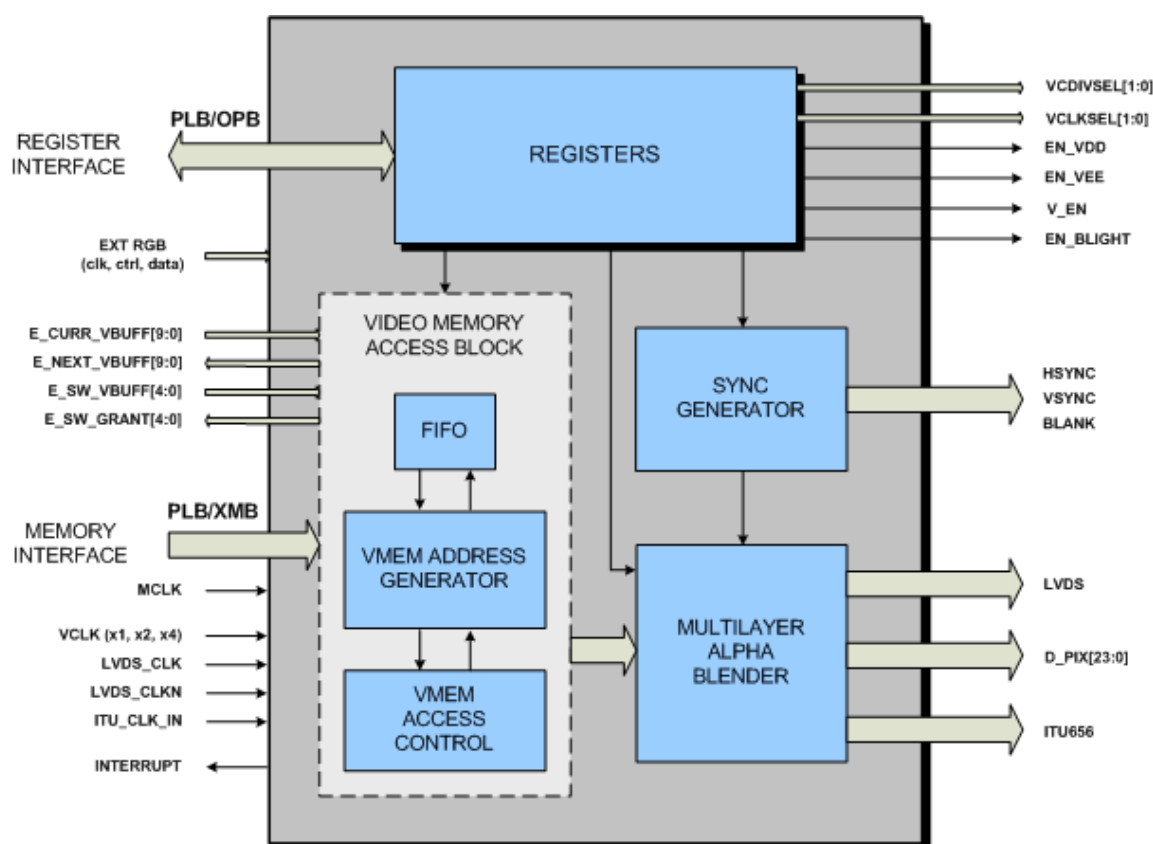
## 2 IP CORE ARCHITECTURE

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

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

### 2.1 Block Schematic

Block schematic on Figure 2.1 shows one possible input channels configuration. Different configurations are available through generic parameters.



**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 PLB, XMB or AXI 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 video synchronization signals. This feature is needed for overlay function when external video data is not stored in the memory used by CVC, or is in analog format.

## 2.1.3 Multilayer Alpha Blender

The Multilayer Alpha blender block consists of a maximum five, defined by C\_NUM\_OF\_LAYERS, equal configurable layer blocks. The outputs of the layer blocks are mixed according to alpha/transparent factors and layer priority. The output of the layer mixer is then routed toward display. The Blender supports layer, pixel and colour alpha blending methods. For additional alpha blending information, refer to chapter 10 MULTILAYER SUPPORT.

## 2.1.4 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/VSNC active state, delay from HSYNC/VSNC 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 colour, 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 Slave PLB, Slave OPB or AXI4-Lite interface, which is configurable by setting the VHDL generic parameter C\_REGS\_INTERFACE.

For more information on logiCVC-ML registers, refer to chapter 11 REGISTERS.

## 2.2 Pinout

Table 2.1: Pinout

Signal name	Type	Description
<b>Global Signals</b>		
VCLK	I	Video clock input
VCLK2	I	Video clock input x2 (used for serialized blender or 12bit parallel multiplexed output interface)
VCLK4	I	Video clock input x4 (used for serialized blender, if selected)
PLLCLK_LOCKED	I	Indication of stable VCLK generated by PLL.
ITU_CLK_IN	I	ITU656 clock input (27 MHz and synchronous to VCLK)
LVDS_CLK	I	LVDS clock
LVDS_CLKN	I	LVDS clock inverted
<b>Memory Interface</b>		
MCLK	I	Memory clock input
PLBV46 Master Interface	Bus	Refer to Xilinx-IBM Core connect specification
XMB Interface	Bus	Xylon Memory Bus. Refer to logiMEM User's Manual
AXI4 Interface	Bus	Refer to AMBA AXI version 4 specification from ARM
<b>Register Interface</b>		
PLBV46 Slave Interface	Bus	Refer to Xilinx-IBM Core connect specification
OPB Slave Interface	Bus	Refer to Xilinx-IBM Core connect specification
AXI4-Lite Interface	Bus	Refer to AMBA AXI version 4 specification from ARM
<b>Display Control Signals</b>		
HSYNC	O	Horizontal sync
VSYNC	O	Vertical sync
PIX_CLK	O	Pixel clock
PIX_CLKN	O	Pixel clock inverted
BLANK	O	Blank/display enable
D_PIX[n : 0]	O	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

Signal name	Type	Description
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
DVI_DATA_P[2:0]	O	DVI pixel data, positive
DVI_DATA_N[2:0]	O	DVI pixel data, negative
<b>Auxiliary Signals</b>		
E_VCLK	I	External VCLK (used when external RGB input is used)
E_VSYNC	I	External VSYNC (used when external RGB input is used)
E_HSYNC	I	External HSYNC (used when external RGB input is used)
E_BLANK	I	External BLANK (used when external RGB input is used)
E_DATA[n : 0]	I	External RGB data (used when external RGB input is used)
E_VIDEO_PRESENT	I	External video present (used when external RGB input is used)
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)
VCDIVSEL[1:0]	O	Video clock divider select
VCLKSEL[2:0]	O	Video clock select
INTERRUPT	O	CVC Interrupt signal, level sensitive, high active
EN_VDD	O	Enable Vdd power supply
EN_BLIGHT	O	Enable backlight power supply
V_EN	O	Enable display control/data signals
EN_VEE	O	Enable Vee power supply

## 2.3 logiCVC-ML Parameters

The logiCVC-ML configuration is defined by generic parameters. These parameters are accessible in XPS. User can set these prior to the IP core's code synthesis to configure logiCVC-ML and eventually reduce slice count.

**Table 2.2: logiCVC-ML Parameters**

Parameter Name	Allowable values	Default value	Description
<b>Version.General Generics</b>			
C_IP_LICENSE_TYPE <sup>1)</sup>	0, 1, 2	0	Constant: 0 - source, 1 - evaluation, 2 - release
C_IP_MAJOR_REVISION <sup>1)</sup>	0 - 31	2	Constant: Values from 0 to 31
C_IP_MINOR_REVISION <sup>1)</sup>	0 - 31	01	Constant: Values from 0 to 31
C_IP_PATCH_LEVEL <sup>1)</sup>	0 - 25	0	Constant: Values from 0(a) to 25(z)
<b>Video Memory.General Generics</b>			
C_VMEM_INTERFACE	0, 1, 2	0	Video memory interface: (0 - PLBv46, 1 - XMB, 2 - AXI)
C_MEM_BURST <sup>2)</sup>	4, 5, 6	4	Number of transfers per burst (4 - 16, 5 - 32, 6 - 64)
C_LITTLE_ENDIAN	0, 1	0	Endianness (0 - big endian, 1 - little endian)
<b>Video Memory.Addresses Generics</b>			
C_VMEM_BASEADDR <sup>22)</sup>	Valid word aligned address	0xffffffff	Video memory base address
C_VMEM_HIGHADDR <sup>22)</sup>	Valid word aligned address	0x00000000	Video memory high address
<b>Video Memory.PLB Master v4.6 Generics</b>			
C_MPLB_AWIDTH <sup>3,8)</sup>	32	32	PLB address bus width
C_MPLB_DWIDTH <sup>3,8)</sup>	32, 64, 128	64	PLB data bus width
C_MPLB_NUM_MASTERS	1-16	2	Number of masters that can be connected
C_MPLB_NATIVE_DWIDTH <sup>3)</sup>	32, 64, 128	64	Set this value to force PLB bus to adjust data width to the native data width.
C_MPLB_PRIORITY <sup>3)</sup>	0, 1, 2, 3	2	Priority on PLB bus: (0 - the lowest, 3 - the highest)
C_MPLB_P2P <sup>3)</sup>	0, 1	0	PLB point to point, non-VHDL constant

Parameter Name	Allowable values	Default value	Description
C_MPLB_SUPPORT_BURST <sup>3)</sup>	1	1	PLB master supports burst access, non-VHDL constant
<b>Video Memory.XMB Generics</b>			
C_XMB_DATA_BUS_WIDTH	32, 64, 128	64	XMB data bus width
<b>Video Memory.AXI4 Master Generics</b>			
C_M_AXI_SUPPORTS_THREADS <sup>23)</sup>	0	0	Support of threads, non-VHDL constant
C_M_AXI_THREAD_ID_WIDTH <sup>23)</sup>	1	1	Width of ID signals for threads
C_M_AXI_SUPPORTS_READ <sup>23)</sup>	1	1	Master supports read transactions, non-VHDL constant
C_M_AXI_SUPPORTS_WRITE <sup>23)</sup>	0	0	Master supports write transactions, non-VHDL constant
C_M_AXI_SUPPORTS_NARROW_BURST <sup>23)</sup>	0	0	Master issues multi-beat transactions in which the size of the data transfers is narrower than the interface data-width, non-VHDL constant
C_M_AXI_DATA_WIDTH <sup>23)</sup>	32, 64, 128	64	AXI data bus width
C_M_AXI_ADDR_WIDTH <sup>23)</sup>	32	32	AXI address bus width
C_M_AXI_PROTOCOL <sup>23)</sup>	AXI4	AXI4	Bus protocol supported, non-VHDL constant
<b>Registers.General Generics</b>			
C_REGS_INTERFACE	0, 1, 2	0	Interface to registers: (0 - OPB slave, 1 - PLBv4.6 slave, 2 - AXI4-Lite)
C_READABLE_REGS <sup>4)</sup>	0, 1	1	Readable registers: (0 - disabled, 1 - enabled)
<b>Registers.Addresses Generics</b>			
C_REGS_BASEADDR	Valid word aligned address	0xffffffff	Registers base address
C_REGS_HIGHADDR	Valid word aligned address	0x00000000	Registers high address
<b>Registers.OPB Slave Generics</b>			
C_OPB_AWIDTH <sup>5, 6)</sup>	32	32	OPB address bus width
C_OPB_DWIDTH <sup>5, 6)</sup>	32, 64	32	OPB data bus width
<b>Registers.PLB Slave v4.6 Generics</b>			
C_SPLB_AWIDTH <sup>7, 8)</sup>	32	32	PLB address bus width

Parameter Name	Allowable values	Default value	Description
C_SPLB_DWIDTH <sup>7, 8)</sup>	32, 64	32	PLB data bus width
C_SPLB_MID_WIDTH <sup>7, 8)</sup>	1, 2, 3, 4	1	PLB Master ID Bus Width. The value is log2(C_SPLB_NUM_MASTERS).
C_SPLB_NUM_MASTERS <sup>7, 8)</sup>	1-16	2	Number of masters that can be connected
C_SPLB_NATIVE_DWIDTH <sup>7, 8)</sup>	32	32	Value always set to 32 to allow PLB bus to adjust data width according to the other devices on bus
<b>Registers.AXI4-Lite Slave Generics</b>			
C_S_AXI_ADDR_WIDTH <sup>24)</sup>	32	32	AXI4-Lite address bus width
C_S_AXI_DATA_WIDTH <sup>24)</sup>	32	32	AXI4-Lite data bus width
C_S_AXI_PROTOCOL <sup>24)</sup>	AXI4LITE	AXI4LITE	Bus protocol supported, non-VHDL constant
<b>User.General Generics</b>			
C_NUM_OF_LAYERS	1, 2, 3, 4, 5	1	Number of logiCVC layers
C_ROW_STRIDE	512, 1024, 2048	1024	Distance in number of pixels between same color pixels for adjacent rows
C_XCOLOR <sup>26)</sup>	0, 1	0	Enable xcolor functionality
C_USE_SIZE_POSITION	0, 1	1	Enable functionality of configurable layer size, position and offset (0 – disabled, 1 – enabled)
C_PIXEL_DATA_WIDTH	12, 15, 16, 18, 24	16	Parallel pixel data width towards the display: (12=12*2, 15=15, 16=16, 18=18, 24=24)
C_DISPLAY_INTERFACE	0, 1, 2, 3, 4, 5	0	Enable different output types. Parallel output (RGB) is always active (0=parallel only, 1=ITU656, 2=LVDS 4bit, 3=camera link 4bit, 4=LVDS 3bit, 5=DVI)
C_LVDS_DATA_WIDTH	3, 4	4	LVDS and camera link display interface data width. XPS assigned, do not modify
C_VCLK_PERIOD <sup>25)</sup>	1000:52630	25000	vclk clock period defined in pico seconds. XPS assigned.
C_USE_BACKGROUND <sup>12)</sup>	0, 1	0	Configure last layer as background
C_USE_SERIALIZED_BLENDER <sup>14)</sup>	0, 1	0	Use only one set of multipliers on higher frequency for alpha blending
C_USE_XTREME_DSP	0, 1, 2	2	Use DSP resources for blender: (0=no, 1=yes, 2=auto)

Parameter Name	Allowable values	Default value	Description
C_USE_MULTIPLIER	0, 1, 2	2	Type of multiplier used in blender: (0=lut, 1=block, 2=auto)
C_USE_E_RGB_INPUT <sup>12)</sup>	0, 1	0	Synchronize logiCVC to external RGB input and use data as one layer (0 - no, 1 - yes)
C_USE_E_VCLK_BUFGMUX <sup>10)</sup>	0, 1	1	Use BUFGMUX for clock switching for external RGB (0 - no, 1 - yes)
C_E_LAYER <sup>10)</sup>	0, 1, 2, 3, 4	0	Use extern RGB input as which layer
C_E_DATA_WIDTH <sup>10)</sup>	8, 16, 24	24	Extern RGB input data width
<b>User.Layer 0 Generics</b>			
C_LAYER_0_DATA_WIDTH	8, 16, 24	16	Layer data width: 8, 16 or 24 bits per pixel
C_LAYER_0_ALPHA_MODE <sup>11)</sup>	0, 1, 2, 3	0	Alpha blending mode: (0=layer, 1=pixel, 2=16bit clut, 3=24bit clut)
C_LAYER_0_OFFSET	integer	0	Layer 0 address offset represented in number of lines <sup>20)</sup>
C_BUFFER_0_OFFSET	integer	1024	Layer 0 double buffer address offset represented in number of lines <sup>21)</sup>
<b>User.Layer 1 Generics</b>			
C_LAYER_1_DATA_WIDTH <sup>12)</sup>	8, 16, 24	16	Layer data width: 8, 16 or 24 bits per pixel
C_LAYER_1_ALPHA_MODE <sup>12, 13)</sup>	0, 1, 2, 3	0	Alpha blending mode: (0=layer, 1=pixel, 2=16bit clut, 3=24bit clut)
C_LAYER_1_OFFSET <sup>12)</sup>	integer	2048	Layer 1 address offset represented in number of lines <sup>20)</sup>
C_BUFFER_1_OFFSET <sup>12)</sup>	integer	1024	Layer 1 double buffer address offset represented in number of lines <sup>21)</sup>
<b>User.Layer 2 Generics</b>			
C_LAYER_2_DATA_WIDTH <sup>14)</sup>	8, 16, 24	16	Layer data width: 8, 16 or 24 bits per pixel
C_LAYER_2_ALPHA_MODE <sup>14, 15)</sup>	0, 1, 2, 3	0	Alpha blending mode (0=layer, 1=pixel, 2=16bit clut, 3=24bit clut)
C_LAYER_2_OFFSET <sup>14)</sup>	integer	4096	Layer 2 address offset represented in number of lines <sup>20)</sup>
C_BUFFER_2_OFFSET <sup>14)</sup>	integer	1024	Layer 2 double buffer address offset represented in number of lines <sup>21)</sup>
<b>User.Layer 3 Generics</b>			
C_LAYER_3_DATA_WIDTH <sup>16)</sup>	8, 16, 24	16	Layer data width: 8, 16 or 24 bits per pixel

Parameter Name	Allowable values	Default value	Description
C_LAYER_3_ALPHA_MODE <sup>16, 17)</sup>	0, 1, 2, 3	0	Alpha blending mode (0=layer, 1=pixel, 2=16bit clut, 3=24bit clut)
C_LAYER_3_OFFSET <sup>16)</sup>	integer	6144	Layer 3 address offset represented in number of lines <sup>20)</sup>
C_BUFFER_3_OFFSET <sup>16)</sup>	integer	1024	Layer 3 double buffer address offset represented in number of lines <sup>21)</sup>
<b>User.Layer 4 Generics</b>			
C_LAYER_4_DATA_WIDTH <sup>18)</sup>	8, 16, 24	16	Layer data width: 8, 16 or 24 bits per pixel
C_LAYER_4_ALPHA_MODE <sup>18, 19)</sup>	0, 1, 2, 3	0	Alpha blending mode (0=layer, 1=pixel, 2=16bit clut, 3=24bit clut)
C_LAYER_4_OFFSET <sup>18)</sup>	integer	8192	Layer 4 address offset represented in number of lines <sup>20)</sup>
C_BUFFER_4_OFFSET <sup>18)</sup>	integer	1024	Layer 4 double buffer address offset represented in number of lines <sup>21)</sup>

1. These parameters are constant. Default values depend on current IP version and purchased license.
2. Valid if C\_VMEM\_INTERFACE = 1 or 2, or C\_VMEM\_INTERFACE = 0 and C\_MPLB\_DWIDTH >= 64
3. Valid if C\_VMEM\_INTERFACE = 0
4. To save some resources user can disable the read register interface. In this mode only interrupt status register and double/triple video/CLUT buffer registers are readable
5. Valid if C\_REGS\_INTERFACE = 0
6. These parameters are normally calculated by the XPS based on what devices are connected to the OPB bus
7. Valid if C\_REGS\_INTERFACE = 1
8. These parameters are normally calculated by the XPS based on what devices are connected to the PLB bus
9. Valid if LVDS or camera link display interface is used (C\_DISPLAY\_INTERFACE = 2, 3, 4 (LVDS or camera link))
10. Valid if C\_USE\_E\_RGB\_INPUT = 1
11. Valid if C\_USE\_E\_RGB\_INPUT = 0 or if (C\_USE\_E\_RGB\_INPUT = 1 and C\_E\_LAYER != 0)
12. Valid if (C\_NUM\_OF\_LAYERS > 1)
13. Valid if C\_USE\_E\_RGB\_INPUT = 0 or if (C\_USE\_E\_RGB\_INPUT = 1 and C\_E\_LAYER != 1)
14. Valid if (C\_NUM\_OF\_LAYERS > 2)
15. Valid if C\_USE\_E\_RGB\_INPUT = 0 or if (C\_USE\_E\_RGB\_INPUT = 1 and C\_E\_LAYER != 2)
16. Valid if (C\_NUM\_OF\_LAYERS > 3)
17. Valid if C\_USE\_E\_RGB\_INPUT = 0 or if (C\_USE\_E\_RGB\_INPUT = 1 and C\_E\_LAYER != 3)
18. Valid if (C\_NUM\_OF\_LAYERS > 4)
19. Valid if C\_USE\_E\_RGB\_INPUT = 0 or if (C\_USE\_E\_RGB\_INPUT = 1 and C\_E\_LAYER != 4)
20. Layer address offset from video memory base address represented in number of lines where each line can have different size. For example, 1KB for 8bpp and C\_ROW\_STRIDE=1024, 2KB for 16bpp and



- C\_ROW\_STRIDE=1024, 4KB for 24bpp layer and C\_ROW\_STRIDE=1024 and 8KB for 24bpp layer and C\_ROW\_STRIDE=2048
21. Double buffer address offset relative to layer address offset represented in number of lines where each line can have different size. For example 1KB for 8bpp and C\_ROW\_STRIDE=1024, 2KB for 16bpp and C\_ROW\_STRIDE=1024 and 4KB for 24bpp layer and C\_ROW\_STRIDE=1024 and 8KB for 24bpp layer and C\_ROW\_STRIDE=2048. Triple buffer address offset is defined as double the double buffer offset
  22. logiCVC-ML can address a maximum of 256MB. However, there is one restriction on setting the base address C\_VMEM\_BASEADDR. In order to reduce the slice consumption, user has to take care to set the base address in boundaries of the address range the logiCVC-ML will use. For example, if logiCVC-ML uses 32MB of memory than the base address can be set only in boundaries of 32MB (C\_VMEM\_BASEADDR = 0x2000000, 0x4000000, ...)
  23. Valid if C\_VMEM\_INTERFACE = 2
  24. Valid if C\_REGS\_INTERFACE = 2
  25. Required only if DVI display interface is used
  26. Valid if C\_PIXEL\_DATA\_WIDTH = 18

## 2.4 logiCVC-ML Port and Parameter Dependencies

**Table 2.3: Port and Parameter Dependencies**

Parameter Name	Affects Port(s)	Description
C_USE_SERIALIZED_BLENDER C_NUM_OF_LAYERS	vclk2 vclk4	When C_USE_SERIALIZED_BLENDER = 1 for C_NUM_OF_LAYERS < 4 vclk2 is used for C_NUM_OF_LAYERS > 3 vclk4 is used
C_PIXEL_DATA_WIDTH	vclk2	When C_PIXEL_DATA_WIDTH = 12
C_PIXEL_DATA_WIDTH	d_pix	Determines pixel data bus width
C_DISPLAY_INTERFACE	itu_clk_in itu656_clk_o itu656_data_o	When C_DISPLAY_INTERFACE = 1
C_DISPLAY_INTERFACE	lvds_clk lvds_clkn lvds_data_out_p lvds_data_out_n lvds_clk_out_p lvds_clk_out_n	When C_DISPLAY_INTERFACE = 2, 3, 4
C_DISPLAY_INTERFACE	dvi_data_p dvi_data_n dvi_clk_p dvi_clk_n	When C_DISPLAY_INTERFACE = 5

Parameter Name	Affects Port(s)	Description
C_LVDS_DATA_WIDTH	lvds_data_out_p lvds_data_out_n	Determines LVDS data bus width
C_USE_E_RGB_INPUT	e_vclk e_vsync e_hsync e_blank e_data	When C_USE_E_RGB_INPUT = 1
C_E_DATA_WIDTH	e_data	When C_USE_E_RGB_INPUT = 1 C_E_DATA_WIDTH determines port width
C_NUM_OF_LAYERS	e_curr_vbuff e_next_vbuff e_sw_vbuff e_sw_grant	Port width is C_NUM_OF_LAYERS dependent, buffer switching is independent for each layer
C_VMEM_INTERFACE <sup>1)</sup>	PLBV46 Master Interface XMB Interface AXI Interface	Only one interface (MPLB, XMB or M_AXI) is available depending on C_VMEM_INTERFACE setting
C_REGS_INTERFACE	OPB Interface PLBV46 Slave Interface AXI4-Lite Slave interface	Only one interface (OPB, SPLB or S_AXI) is available depending on C_REGS_INTERFACE setting

1. mclk port is used for all memory interfaces as memory clock input

## 3 CLOCK SIGNALS

The VCLK clock signal controls all circuits inside the logiCVC-ML core, except the video memory sub-system (PLB, XMB or AXI) related circuits and registers (OPB, PLB or AXI). Memory sub-system uses MCLK and register sub-system uses OPB\_CLK, SPLB\_CLK or S\_AXI\_ACLK clock signals. The VCLK clock signal frequency should be set according to the video display resolution.

In general, the VCLK frequency depends on display characteristics. Other clock frequencies are applicable, too.

In case ITU656 output standard is used (C\_DISPLAY\_INTERFACE = 1), user must supply input ITU656 clock signal ITU\_CLK\_IN which has a frequency of 27 MHz as requested by the ITU656 standard. Additionally, VCLK must be supplied to logiCVC-ML input and it has to be 13.5 MHz and fully synchronous to ITU\_CLK\_IN.

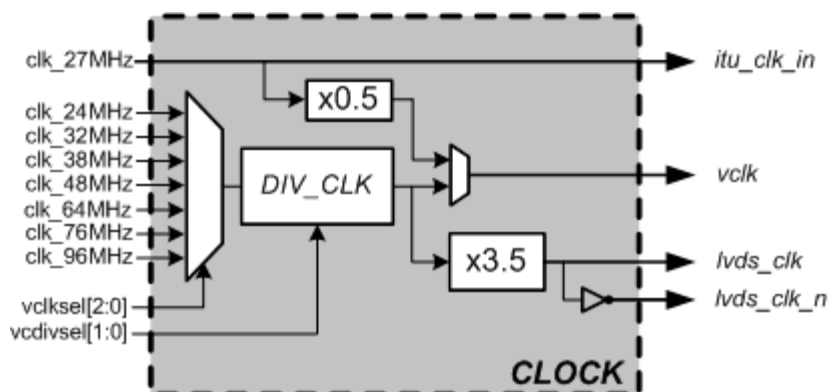
In case LVDS output standard is used (C\_DISPLAY\_INTERFACE = 2 or 4), or camera link output is used (C\_DISPLAY\_INTERFACE = 3) user must supply two additional clock inputs, LVDS\_CLK and LVDS\_CLKN. These clock signals must have 3.5 times higher frequency and be synchronous to VCLK.

In Spartan-6 and 7 series FPGAs family specific HW IO serializers are used for LVDS or camera link output streams (OSERDES2 and OSERDESE2 respectively). In this case, beside VCLK, user must supply only one additional clock input LVDS\_CLK. This clock signal must have 7 times higher frequency and be synchronous to VCLK. For Spartan-6 FPGAs these two clock signals must be sourced by the same PLL module. Additionally, LOCKED output of the PLL must be connected to PLLVCLK\_LOCKED input of the logiCVC-ML. Figure 3.2 shows principle of clock connection to logiCVC-ML for the here described case. For 7 series FPGAs 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. For 7 series FPGAs PLLVCLK\_LOCKED input is not used.

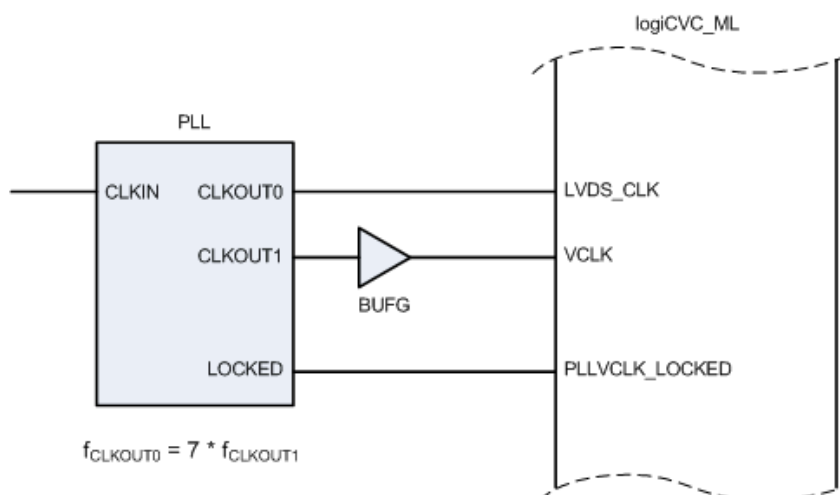
In case serialized layer blender is used (C\_USE\_SERIALIZED\_BLENDER = 1) user has to provide in addition to VCLK also VCLK2 (double the VCLK frequency) and possibly VCLK4 (four times the VCLK frequency). VCLK4 has to be provided if logiCVC-ML is configured to use four or five layers.

For example, the logiCVC-ML reference design uses the multiplexer in front of the VCLK clock source. The 24, 32, 38, 48, 64, 76, 96 MHz clock signals are inputs to the multiplexer. The VCLKSEL[2:0] control bus from CTRL register selects one of these frequencies. Additionally clock signals can be divided by 2, 4 or 8. The VCLK divider is controlled by VCDIVSEL[1:0] signals (bits) from the CTRL register.

These frequencies can be obtained by using DCMs (Xilinx FPGA primitive) with a 96 MHz input frequency by dividing factors of 2, 2.5, 3 and 4.



**Figure 3.1: Clock generator block schematic**



**Figure 3.2 : Example of clock connection in Spartan-6 FPGA when family specific HW IO serializer (OSERDES2) is used for LVDS and camera link streams**

## 4 MEMORY INTERFACE

The video image is stored in video memory, which is accessible through Master PLBv4.6, XMB or AXI interface. By default MPLB bus is used but by setting C\_VMEM\_INTERFACE to '1' logiCVC-ML will access video memory through XMB, setting C\_VMEM\_INTERFACE to '2' logiCVC-ML will access video memory through AXI. However, have in mind that memory clock input signal MCLK is used for all interfaces so which ever interface is used this signal has to be connected to memory clock source.

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™.

The logiMEM supports the UMA (Unified Memory Architecture). The UMA architecture has a number of sub-systems that share a common memory (usually SDRAM memory chips). Memory sharing is obtained by using Xilinx (IBM) PLB bus arbitration and multiple ports.

logiCVC-ML PLB, XMB or AXI master interface can be configured as 32-bit, 64-bit or 128-bit wide by setting the appropriate VHDL generic parameter (C\_XMB\_DATA\_BUS\_WIDTH, C\_MPLB\_NATIVE\_DWIDTH or C\_M\_AXI\_DATA\_WIDTH). For better PLB and memory bandwidth utilization, fixed burst accesses of 16 are used on 32-bit PLB bus. In case of 64-bit PLB bus, fixed burst lengths of up to 32 or 64 can be used by setting VHDL generic parameter C\_MEM\_BURST. For more information on fixed burst transfers, please refer to PLBv4.6 architecture specifications. When using XMB or AXI interface, maximum memory burst can be set with the same VHDL generic parameter independently of data width. Out of block accesses are obtained by using single PLB/XMB/AXI cycles and bursts lower than defined by C\_MEM\_BURST. PLB abort cycle contained in PLB bus standard is not used.

With the VHDL generic parameter C\_VMEM\_BASEADDR user can set the base address offset from which logiCVC-ML will read the memory. Altogether, logiCVC-ML can address a maximum of 256MB. However, there is one restriction on setting the base address C\_VMEM\_BASEADDR. In order to reduce the slice consumption, user has to take care to set the base address in boundaries of the address range the logiCVC-ML will use. For example, if logiCVC-ML uses 32MB of memory than the base address can be set only in boundaries of 32MB (C\_VMEM\_BASEADDR = 0x2000000, 0x4000000, ...).

For more information on PLBv4.6, XMB or AXI bus architecture, please refer to Xilinx / IBM Processor Local Bus, Xylon logiMEM or AXI (AMBA) documentation, respectively.

### 4.1 Pixel Row Stride

Stride is the distance in bytes between same color pixels for adjacent rows. Depending on layer data width (bpp), alpha blending mode and C\_ROW\_STRIDE, stride is set to 0.5, 1, 2, 4 or 8-kilobyte boundary.

Table 4.1: Pixel row stride

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

Table.4.2: Layer configured for 8bpp, layer alpha blending mode, C\_ROW\_STRIDE=1024

Memory address			Pixel display row
0000	...	003FF	row 0
0400	...	007FF	row 1
0800	...	00BFF	row 2
...	...	...	
...	...	...	
...	...	...	
3BC00	...	3BFFF	row 239
3C000	...	3C3FF	row 240
...	...	...	
...	...	...	
...	...	...	
77800	...	77BFF	row 478
77C00	...	77FFF	row 479
...	...	...	

**Note:** Start address is calculated from C\_VMEM\_BASEADDR and C\_LAYER\_X\_OFFSET. End of layer is limited to maximum 2048 lines or by beginning of the next layer.

**Table 4.3: Layer configured for 16bpp, layer alpha blending, C\_ROW\_STRIDE=1024**

Memory address			Pixel display row
0000	...	007FE	row 0
0800	...	00FFE	row 1
1000	...	017FE	row 2
...	...	...	
...	...	...	
...	...	...	
77800	...	77FFE	row 239
78000	...	787FE	row 240
...	...	...	
...	...	...	
...	...	...	
EF000	...	EF7FE	row 478
EF800	...	EFFFE	row 479
...	...	...	

**Note:** Start address is calculated from C\_VMEM\_BASEADDR, and C\_LAYER\_X\_OFFSET. End of layer is limited to maximum 2048 lines or by beginning of the next layer.

**Table 4.4: Layer configured for 1280x480, 16bpp, layer alpha blending, C\_ROW\_STRIDE=2048**

Memory address			Pixel display row
0000	...	009FE	row 0
1000	...	019FE	row 1
2000	...	029FE	row 2
...	...	...	
...	...	...	
...	...	...	
EF000	...	EF9FE	row 239
F0000	...	F09FE	row 240
...	...	...	
...	...	...	
...	...	...	
1DE000	...	1DE9FE	row 478
1DF000	...	1DF9FE	row 479
...	...	...	

**Note:** Start address is calculated from C\_VMEM\_BASEADDR and C\_LAYER\_X\_OFFSET. End of layer is limited to maximum 2048 lines or by beginning of the next layer.

**Table 4.5: Layer configured for 640x480, 24bpp, layer alpha blending, C\_ROW\_STRIDE=1024**

Memory address			Pixel display row
0000	...	00FFC	row 0
1000	...	01FFC	row 1
2000	...	02FFC	row 2
...	...	...	
...	...	...	
...	...	...	
EF000	...	EFFFC	row 239
F0000	...	F0FFC	row 240
...	...	...	
...	...	...	
...	...	...	
1DE000	...	1DEFFC	row 478
1DF000	...	1DFFFC	row 479
...	...	...	

**Note:** Start address is calculated from C\_VMEM\_BASEADDR and C\_LAYER\_X\_OFFSET. End of layer is limited to maximum 2048 lines or by beginning of the next layer.

## 4.2 Memory Layout

Memory layout for each layer depends on three factors: layer data width (bpp), layer alpha blending mode and endianness. Following tables show different memory configurations.

Endianness configuration affects how logiCVC-ML reads video memory. Bytes in pixel are reordered according to little or big endian setting (C\_LITTLE\_ENDIAN). The following tables describe memory and pixel layout configurations according to layer width and endianness.

**Table 4.6: 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 4.7: 8bpp pixel layout – big endian**

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

**Table 4.8: 8bpp pixel layout – little endian**

Pixel bit	Layer alpha	CLUT indexed	Pixel alpha
0	Blue0	Pix index0	Blue0
1	Blue1	Pix index1	Blue1
2	Green0	Pix index2	Green0
3	Green1	Pix index3	Green1
4	Green2	Pix index4	Green2
5	Red0	Pix index5	Red0
6	Red1	Pix index6	Red1
7	Red2	Pix index7	Red2
8			Alpha0
9			Alpha1
10			Alpha2
11			Dummy
12			Dummy
13			Dummy

Pixel bit	Layer alpha	CLUT indexed	Pixel alpha
14			Dummy
15			Dummy

**Note:** In 8bpp with pixel alpha mode, only 3 bits are used because alpha factor is multiplied with every colour 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 4.9: 16bpp layer memory layout**

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

**Table 4.10: 24bpp layer memory layout**

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

**Table 4.11: 16bpp and 24bpp pixel layout – big endian**

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

Table 4.12: 16bpp and 24bpp pixel layout – little endian

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

**Note:** In 16bpp with pixel alpha mode, only 6 bits are used because alpha factor is multiplied with every colour separately. As 16bpp is defined as RGB 565, 5-bit alpha is used for red and blue and 6-bit for green. Nevertheless, user has to make sure that in this mode alpha factor spreads from 0 to 63.

## 4.3 Memory bandwidth requirements

Required memory bandwidth greatly depends on the logiCVC-ML configuration parameters described in chapter 2.3 logiCVC-ML Parameters, display resolution and video clock frequency.

The actual required bandwidth is calculated by two formulas, one describing the maximum required memory bandwidth and the other describing the average required memory bandwidth.

Maximum memory bandwidth requirement is calculated by the following equation:

### Equation 4.1: Required maximum memory bandwidth

$$RMBW_{max} = (layer_0bpp + layer_1bpp + \dots + layer_nbpp) \times VCLK\ freq$$

where:

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

$layer_nbpp$ : layer n bits per pixel configuration which depends on C\_LAYER\_n\_DATA\_WIDTH and C\_LAYER\_n\_ALPHA\_MODE generic parameters. Please refer to Table 4.13 for exact values.

**Table 4.13:  $layer_nbpp$  values used for memory bandwidth calculation**

		Data width (bit)		
		8	16	24
<b>Layer alpha mode</b>	Layer alpha (0)	8	16	32
	Pixel alpha (1)	16	32	32
	CLUT (2, 3)	8	-	-

Average memory bandwidth requirement is calculated by the following equation:

### Equation 4.2: Required average layer memory bandwidth

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

where:

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

$bpp$ : layer bits per pixel configuration outlined in Table 4.13

$width$ : layer horizontal size in pixels defined by layer size registers described in chapter 11.2.18 Layer Size Registers - LX\_WIDTH, LX\_HEIGHT

$height$ : layer vertical size in lines defined by layer size registers described in chapter 11.2.18 Layer Size Registers - LX\_WIDTH, LX\_HEIGHT

$n$ : 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 at his disposal maximum bandwidth value, it will function properly in any design configuration independent of other peripherals in the system. If logiCVC-ML has at his disposal an

average bandwidth value, a stable functioning of logiCVC-ML depends on other peripherals in the system requesting memory and 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 (PLB, XMB, AXI), memory controller in the design, off chip memory devices and memory burst length (C\_MEM\_BURST). For example, the bigger the burst size, number of transfers per burst, the higher the efficiency.

### Example 1

Let's assume we have the following logiCVC-ML configuration:

C\_NUM\_OF\_LAYERS = 2,  
 C\_LAYER\_0\_DATA\_WIDTH = 16, C\_LAYER\_0\_ALPHA\_MODE = 1 (pixel),  
 C\_LAYER\_1\_DATA\_WIDTH = 24, C\_LAYER\_1\_ALPHA\_MODE = 0 (layer),  
 resolution = 1024x768 @60Hz (XGA),  
 pixel clock = 65MHz,  
 layer sizes = resolution.

According to the above equations the required maximum memory bandwidth is

$$RMBW_{\max} = (32 + 32) \times 65MHz = 416000000b/s \cong 496MB/s$$

while the required average memory bandwidth is

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

### Example 2

Let's assume we have the following logiCVC-ML configuration:

C\_NUM\_OF\_LAYERS = 5,  
 C\_LAYER\_0\_DATA\_WIDTH = 8, C\_LAYER\_0\_ALPHA\_MODE = 2 (CLUT),  
 C\_LAYER\_1\_DATA\_WIDTH = 16, C\_LAYER\_1\_ALPHA\_MODE = 1 (pixel),  
 C\_LAYER\_2\_DATA\_WIDTH = 16, C\_LAYER\_2\_ALPHA\_MODE = 0 (layer),  
 C\_LAYER\_3\_DATA\_WIDTH = 24, C\_LAYER\_3\_ALPHA\_MODE = 0 (layer),  
 C\_LAYER\_4\_DATA\_WIDTH = 24, C\_LAYER\_4\_ALPHA\_MODE = 0 (layer),  
 C\_USE\_BACKGROUND = 1,  
 resolution = 854x480 @60Hz (WVGA),  
 pixel clock = 28MHz,  
 layer size<sub>0</sub> = 400x240,  
 layer size<sub>1</sub> = 512x240,

layer size<sub>2</sub>= resolution,  
 layer size<sub>3</sub> = resolution.

According to the above equations the required maximum memory bandwidth is

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

while the required average memory bandwidth is

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

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 (C\_USE\_BACKGROUND = 1), which means data for this layer is not fetched from memory but is read from internal logiCVC-ML register. Please refer to chapter 11.2.7 Background Colour Register – BACKGROUND for more information.

## 5 CPU INTERFACE

### 5.1 Register Interface

The logiCVC-ML registers are accessed through Slave OPB, Slave PLBv4.6 or Slave AXI4-Lite interface depending on VHDL generic parameter C\_REGS\_INTERFACE. Please consult Xilinx OPB, PLBv4.6 or AXI specification regarding the functionality of these interfaces.

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 setting the VHDL generic parameter C\_READABLE\_REGS to 0. Variable register widths are used; 8, 16, 24 or 32-bit. Non-used bits inside each register are reserved or read '0'. Register memory map and description is given in chapter 11 REGISTERS.

OPB, SPLB and S\_AXI clocks are independent of other clocks connected to logiCVC-ML.

### 5.2 Interrupt Interface

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

- Video buffer select (double/triple buffering)
- CLUT select (double CLUT)
- V-Sync
- External video input

Each layer has three buffers in video memory. Switching of the buffers is controlled by VBUFF\_SEL register and is executed by memory address generator at the beginning of a new frame. At this time, logiCVC-ML sets the corresponding layer switch buffer interrupt in the INT register. This is how logiCVC-ML signals to the CPU that it has switched the buffers and that new image can be written in the other buffer. In this way, flicker-free reproduction is guaranteed. For more on double/triple buffering, refer to chapter 11.2.8 Video Buffer Select Register - VBUFF\_SEL.

Each layer that has CLUT alpha blending enabled has double CLUT. Similarly as above but with a restriction to two CLUTs, switching of CLUTs is controlled by CLUT\_SEL register. At the beginning of a new frame, CLUT is switched and this is signalled to CPU by setting the corresponding bit in the INT register. For more on double CLUT refer to chapter 11.2.9 CLUT Select Register - CLUT\_SEL.

One bit in the INT register is used for signalling 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.



Additionally, one bit in the INT register is used when an external RGB input stream is used to drive one logiCVC-ML layer. In that case ( $C\_USE\_E\_RGB\_INPUT = 1$ ), one bit is used to signal to the CPU that external RGB source is present.

All of the above mentioned interrupt sources can be masked using INT\_MASK register. Refer to chapter 11.2.11 Interrupt Mask Register - INT\_MASK for more information.

## 6 DISPLAY INTERFACE

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

- Parallel pixel data interface including three control signals and clock
- LVDS interface, four 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

### 6.1 Parallel Pixel Data Interface

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

HSYNC is a horizontal synchronous signal. It is active synchronously with each pixel row start.

VSYNC is a vertical synchronous signal. It is active synchronously with every first pixel line occurrence.

The TFT and CRT video displays require some blank time between the last pixel of a 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 D\_PIX data bus.

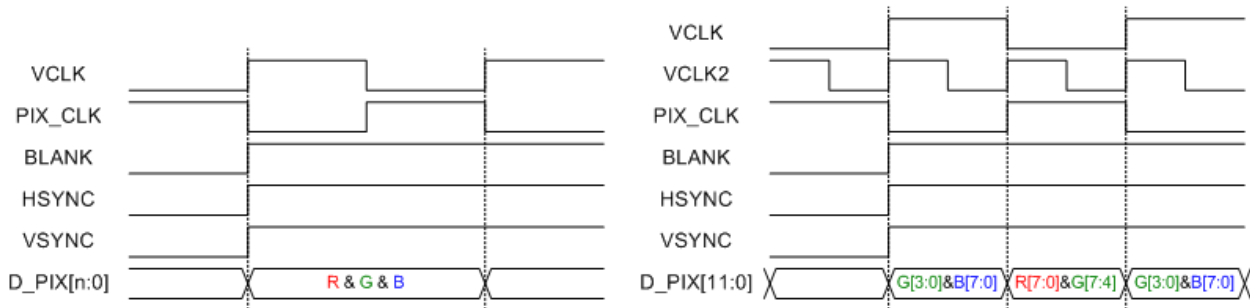
PIX\_CLK is the main display clock and PIX\_CLKN is the inverted version for use on some displays/LCDs that use differential clock input.

The display manufacturers use different naming conventions for display control signals, here are some:

- HSYNC i.e. LP, CL1
- VSYNC i.e. FLM, S
- BLANK i.e. DE
- PIX\_CLK, SHCLK, CL2

The pixel data are output on the D\_PIX[C\_PIXEL\_DATA\_WIDTH-1 : 0] data bus. Through VHDL generic parameter C\_PIXEL\_DATA\_WIDTH user can adjust the data bus width (15, 16, 18, and 24) according to displays requirement. Additionally, user can select C\_PIXEL\_DATA\_WIDTH = 12x2. This activates a mode in which data is outputted at both edges of PIX\_CLK effectively getting a 24 bit interface, but reducing the number of signals to 12. Please, have in mind that to use this 12-bit mode, user has to supply VCLK2 input clock that has twice the frequency and is synchronous to the VCLK input.

Parallel interface can be active even when other display interfaces are selected. Only parallel interface can be selected by setting parameter C\_DISPLAY\_INTERFACE = 0.



**Figure 6.1: Parallel data interface and 12x2-bit multiplexed data interface**

**NOTE:**

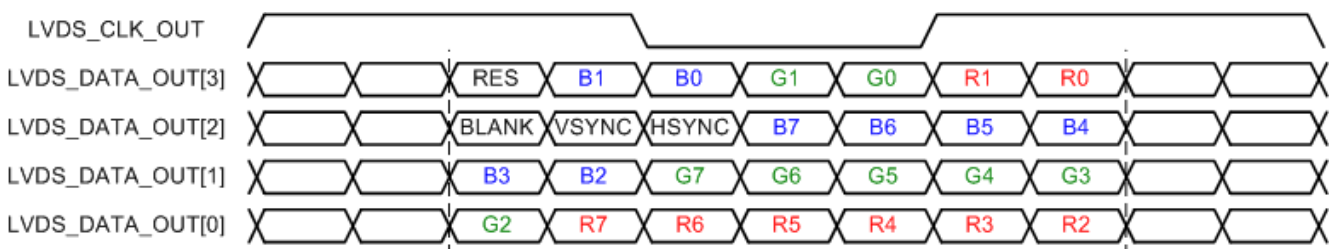
1. To activate the pixel data interface user must enable the interface through the Power control register. For more information, refer to chapter 11.2.12 Power Control Register – PWRCTRL.
2. The PIX\_CLK active edge and polarity of the control signals for both modes depend on the settings in the CTRL register. The above figures represent default settings (falling active edge).

## 6.2 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 two LVDS input clocks to the `LVDS_CLK` and `LVDS_CLKN` input ports. The frequency of these clocks has to be 3.5 times faster than video clock `VCLK`. In addition, `LVDS_CLK` has to be synchronous to `VCLK` and in counter-phase to `LVDS_CLKN`, i.e. shifted for 180 degrees.

In case of Spartan-6 and 7-series FPGAs, i.e. use of family specific HW IO serializers for LVDS outputs, beside `VCLK`, user must supply only `LVDS_CLK`. Frequency of this clock has to be 7 times faster than video clock `VCLK`. In addition, `LVDS_CLK` has to be synchronous to `VCLK`.

LVDS interface uses four or five LVDS pairs, four or three for data and one for clock. To drive 24-bit video data and three control signals, 27 bits in total, over 4 LVDS pairs, LVDS coder uses 3.5 times faster clock (falling and rising edge). In this way, seven signals are transmitted over one LVDS pair. Figure 6.2 shows the order of data and control bits on four LVDS data pairs.



**Figure 6.2: 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.

The pixel data order on the LVDS bus is made according to National semiconductors LVDS Transmitter/Receiver DS90CF384/DS90C383.

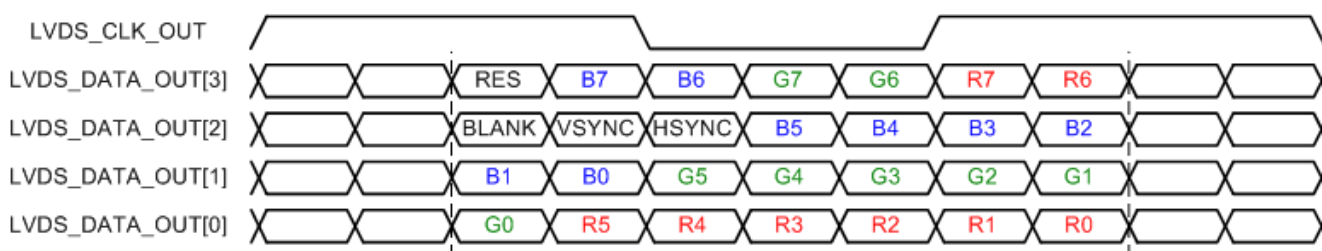
Regardless of the LVDS interface selection, the logiCVC-ML still drives the parallel pixel interface, PIX\_CLK, HSYNC, VSYNC, BLANK and D\_PIX[C\_PIXEL\_DATA\_WIDTH-1 : 0], if Power Control register is configured correctly. For more information, refer to chapter 11.2.12 Power Control Register – PWRCTRL.

## 6.3 Camera Link Interface

To use camera link interface, parameter C\_DISPLAY\_INTERFACE has to be set to 3.

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



**Figure 6.3: Pixel data order on Camera link LVDS data bus**

Camera link in the form of 3bit data interface is the same as LVDS 3bit interface.

## 6.4 Video Interface (ITU656)

The logiCVC-ML is capable of driving both ITU-656 video standards, NTSC and PAL on the ITU656\_DATA[7:0] output bus.

To enable the ITU656 interface, VHDL generic parameter C\_DISPLAY\_INTERFACE has to be set to 1. Please refer to chapter 2.3 logiCVC-ML Parameters.

Additionally, NTSC or PAL video standard is selected through DTYPE register, see chapter 11.2.6 Display Type Register - DTYPE. The logiCVC-ML generates control and data signals on the ITU656\_DATA[7:0] bus according to ITU656 NTSC or PAL standard.

When this interface is used 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. Note that logiCVC-ML referent design does not have these clocks generated according to ITU656 standard.

For most applications that require analog 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.

By setting C\_DISPLAY\_INTERFACE parameter to 1, user has no more influence on some logiCVC-ML registers including resolution and sync registers 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 thru one bit in the corresponding Layer Control register. For more information, please refer to chapter 11.2.20 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[C\_PIXEL\_DATA\_WIDTH-1 : 0], if Power Control register is configured correctly. For more information, refer to chapter 11.2.12 Power Control Register – PWRCTRL.

## 6.5 Digital Video Interface (DVI)

The logiCVC-ML supports Digital Video Interface (DVI) output by directly driving the DVI monitor from FPGA. When selecting DVI output by setting C\_DISPLAY\_INTERFACE parameter to 5 user is enabling three data and one clock LVDS pair output from logiCVC-ML

DVI output is only supported in Xilinx Spartan-6 family because of two constraints. The first one is that DVI standard requires the use of TMDS IO standard that is only supported in Spartan-6 family. The second one is that the implementation is using special Spartan-6 serializers (OSERDES2).

The maximum DVI resolutions that logiCVC-ML supports are constrained by the maximum toggle rate of Spartan6 TMDS IOs. Table 6.1 outlines the maximum Spartan-6 TMDS IO performance while Table 6.2 outlines common video resolutions and required pixel clocks. We strongly recommend that users regularly check the latest Xilinx Spartan-6 documentation for possible timing changes mentioned in Table 6.1.

**Table 6.1: Spartan-6 TMDS IO performance**

Speed Grade	Mb/s
-3	1080
-3N	1050
-2	950
-1L	500

**Table 6.2: Common video resolutions supported**

Resolution (@60Hz)	Pixel clock (MHz)	Data rate (Mb/s)
VGA (640x480)	25.25	252.5

Resolution (@60Hz)	Pixel clock (MHz)	Data rate (Mb/s)
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

Regardless of DVI output selection, the logiCVC-ML still drives the parallel pixel interface, PIX\_CLK, HSYNC, VSYNC, BLANK and D\_PIX[C\_PIXEL\_DATA\_WIDTH-1 : 0], if Power Control register is configured correctly. For more information, refer to chapter 11.2.12 Power Control Register – PWRCTRL.

## 6.6 Flat Panel Displays

The logiCVC-ML supports flat panel displays built with different technologies: plasma, electroluminescent and liquid crystal displays LCD (TFT, analog RGB TFT).

The single panel displays require drawing sequence to be similar to the one required by CRT displays, i.e. the video data is captured from one video memory area.

The logiCVC-ML supports display resolutions from 64x1 up to 2048x2048. The most popular display resolutions of 128x64, 320x240, 400x234, 640x480, 800x480, 800x600, 1024x768, 1280x1024, 1920x1200 are supported, as well as non-standard display resolutions.

The only limitations on horizontal resolution are its minimum and maximum size, which are 64 and 2048 pixels accordingly. The horizontal resolution must be greater than 64 because of the HSYNC control signal generation.

Interface signals and timing requirements vary from display to display. The logiCVC-ML can be adjusted to these requirements by programming only a few registers.

## 6.7 CRT Displays

The logiCVC-ML supports CRT display resolutions from 64x1 to 2048x2048. The CRT requires analog input signals between 0 – 0.7 Vpp. The logiCVC-ML outputs are digital data; therefore, an external Video DAC must be added. The video DAC converts RGB digital to analog signals.

## 6.8 Pixel Clock

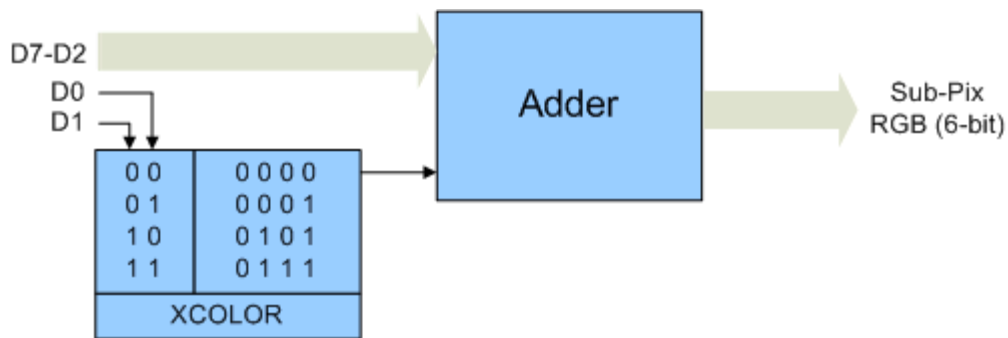
The pixel clock output, PIX\_CLK, is proportional to VCLK clock input or E\_VCLK clock input (if external parallel input is used) and to the control bits in the DTYPE and CTRL register. Please refer to chapter 11.2 Register Description. To support the functionality of adjustable PIX\_CLK clock frequencies (when external parallel input is not used), special clock module is requested outside of the logiCVC-ML core. This module is then controlled with the VCDIVSEL and VCLKSEL signals, which are outputs from the logiCVC-ML core.

Depending on the selected TFT display used, many PIX\_CLK frequencies can be obtained.

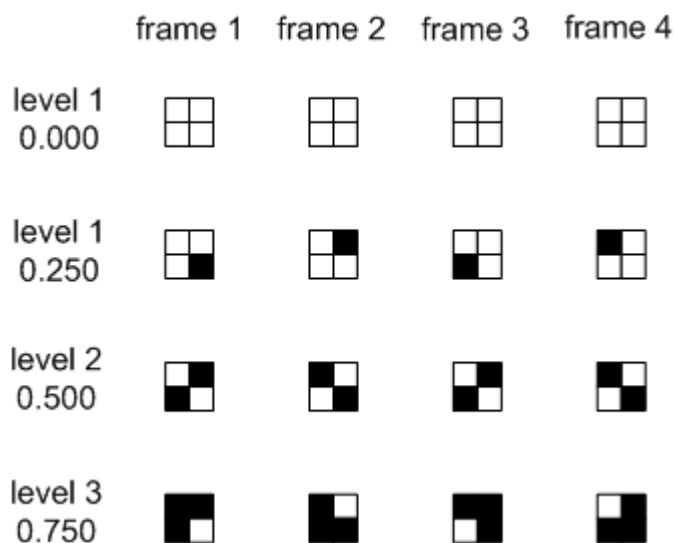
In addition, if CRT display is used, some predefined VGA or SVGA frequencies have to be selected.

## 7 DITHERING MODULE

Dithering module is designed for arithmetic calculations on pixels to generate additional levels of colour. It is implemented by setting C\_XCOLOR generic to 1 when C\_PIXEL\_DATA\_WIDTH is set to 18 or 3bit LVDS interface is used (C\_DISPLAY\_INTERFACE = 4). XCOLOR pixel processing includes spatial dithering and temporal dithering (see Figure 7.1). Two least significant bits of each 8-bit colour of 24 bit wide pixel-bus are inputs to XCOLOR unit. Dependent on their value, each 6-bit colour 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 7.2).



**Figure 7.1 XCOLOR dithering module**



**Figure 7.2 XCOLOR temporal and spatial dithering**



## 8 EXTERNAL RGB INPUT INTERFACE

logiCVC-ML can be configured to use external RGB input stream data as one of its layers. Four VHDL generic parameters are used to configure it:

- C\_USE\_E\_RGB\_INPUT – enables external RGB input functionality (0, 1)
- C\_USE\_E\_VCLK\_BUFGMUX – enables usage of BUFGMUX for switching video clock from vclk to e\_vclk when e\_video\_present is set
- C\_E\_LAYER – determines which layer will be used for RGB data (0 – 4)
- C\_E\_DATA\_WIDTH – determines external RGB data width (8, 16, 24)

logiCVC-ML Input signals used for RGB 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 downto 0] – external data
- E\_VIDEO\_PRESENT – external RGB 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 RGB 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 C\_USE\_E\_VCLK\_BUFGMUX is not set, 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. Please check Figure 11.1 and Figure 11.2 that represent the way logiCVC-ML outputs the control signals by default, which is the same way it expects them at the external RGB 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 8 bit so the maximum values that can be measured are 256. Resolution registers are limited with C\_ROW\_STRIDE VHDL generic 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 11.2.13 External Input Horizontal Resolution - E\_HRES and 11.2.14 External Input Vertical Resolution - E\_VRES for more information.

**NOTE:**

1. The polarity of the control signals depend on the settings in the CTRL register. For more information, refer to chapter 11.2.5 Display Control Register – CTRL.

## 9 DISPLAY ENHANCEMENT FUNCTIONS

### 9.1 Vertical and Horizontal Centring

The vertical and horizontal centring is used to position the picture on the display screen. That is particularly useful for applications using lower resolutions on displays supporting a higher picture resolution.

CRT displays have a non-fixed number of horizontal and vertical lines and therefore they can easily (inherently) display the pictures with different picture resolutions.

The structure of LCD displays has a fixed number of display lines and columns used for picture presentation.

Displaying the lower resolution picture, a part of display screen is unused. Programming horizontal and vertical back porch and front porch registers can set the picture position on screen.

This feature is available for TFT LCDs only.

The horizontal back porch register defines the width of unused left screen side. The horizontal front porch register defines the width of unused right screen side.

The number of unused display lines below the picture can be defined by setting the value of a vertical back porch (VSY\_BP) register.

The VSY\_FP register is used to define the vertical front porch – a number of unused lines above the picture. For more information on setting vertical porches, please refer to chapter 11.2.3 Vertical Sync and Porches - VSY\_FP, VSY, VSY\_BP.

### 9.2 Programmable Outputs

The polarity of display control signals differs for various displays.

By simple programming of the CTRL register, the polarity can be adjusted for HSYNC, VSYNC, BLANK (enable) and PIX\_CLK signals. These display control signals can be set to an active high or low level.

Some displays forbid the simultaneous activation of a HSYNC and VSYNC signals. Therefore, it is possible to disable one of them while the other one is active.

For more information on setting the CTRL register, refer to chapter 11.2.5 Display Control Register – CTRL.

## 9.3 Power Sequencing

The liquid crystal inside LCD displays must be protected from voltages that can permanently damage it. Therefore, the LCD power supplies require precise timing sequencing. Power sequencing procedure: the first step is 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 PWRCTRL register fulfils power-sequencing procedure.

The EN\_xx signals are used for power sequencing. The EN\_VDD turns on/off the display logic (main) power supply, the EN\_VEE turns on/off the LCD liquid crystal power supply, while EN\_BRIGHT turns on/off the backlight power supply.

The V\_EN signal can enable or three-state display control and data signals. It is used in the power-sequencing procedure.

More data about PWRCTRL programming can be found in chapter 11.2.12 Power Control Register – PWRCTRL.

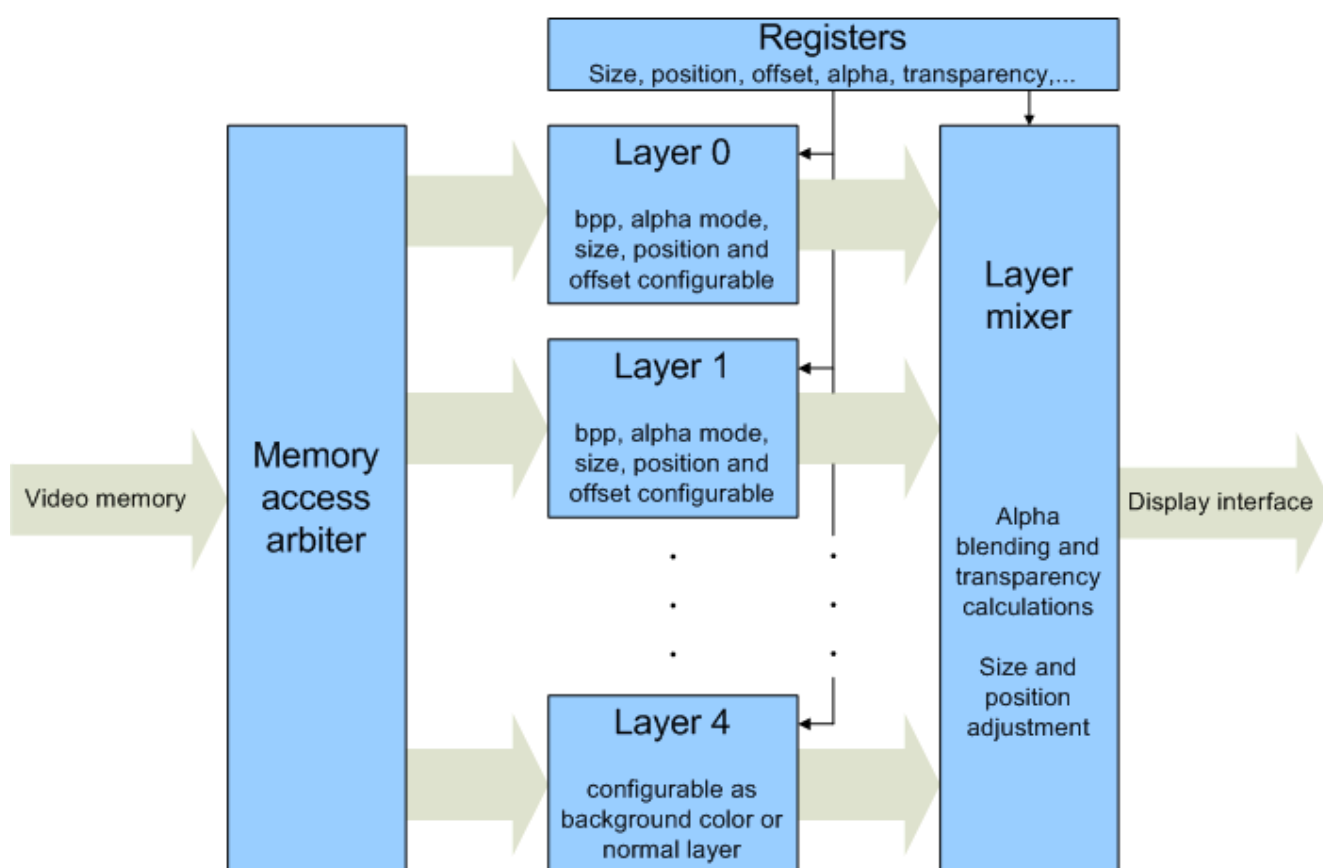
## 9.4 Inverse Video

Inverse video can be used only with LCD displays. Setting bit 7 in the CTRL register inverts the data output from the logiCVC-ML video controller, inverting the complete picture drawn on the display screen.

Note that inverse video cannot be used with ITU656 output standard.

## 10 MULTILAYER SUPPORT

logiCVC-ML supports up to 5 layers. The number of layers used is configurable through the VHDL generic parameter C\_NUM\_OF\_LAYERS. Block schematic of multilayer architecture can be seen on Figure 10.1.



**Figure 10.1 Multilayer architecture**

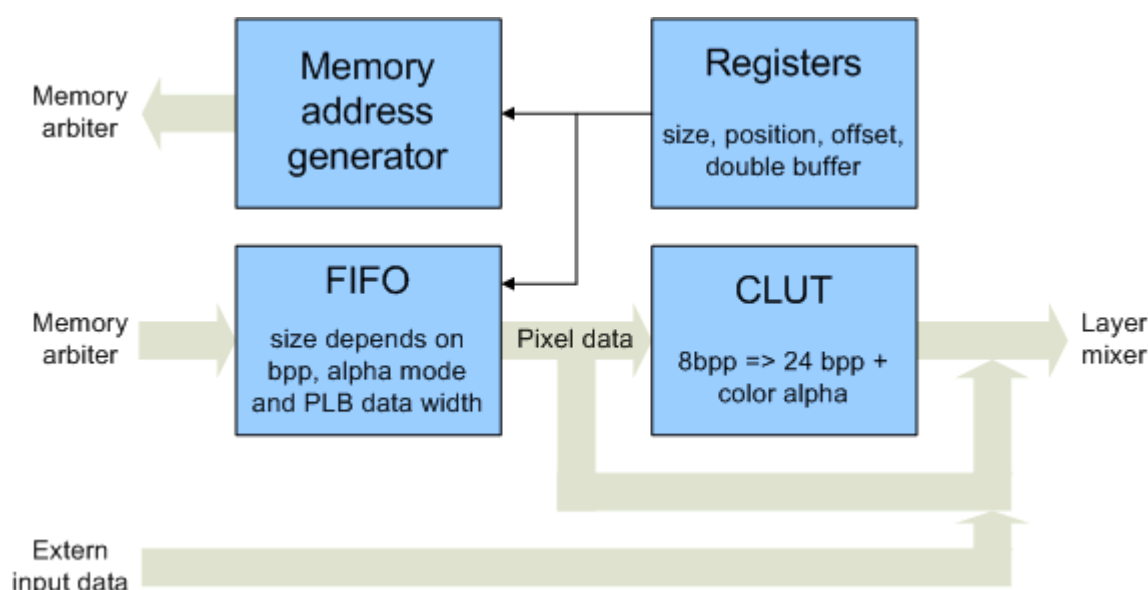
### 10.1 Memory Access Arbiter

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 PLB/XMB/AXI bus.

## 10.2 Layer Block

Layer consists of the following blocks:

- Memory address generator
- Pixel data FIFO
- Colour Look-up Table, if used
- Layer related register set (size, position, ...)



**Figure 10.2 Layer block schematic**

Depending on the logiCVC-ML configuration, layer can consist of some or all of the above mentioned blocks. For example, if layer is configured not to use external RGB input (`C_USE_E_RGB_INPUT = 0`) and does not use CLUT alpha blending, it consists of address generator, FIFO and layer related register set. However, if layer is configured to use external RGB input, it consists of only the size and position register set as data is directly inputted to the layer mixer.

### 10.2.1 Memory Address Generator

Each logiCVC-ML layer that reads data from video buffer (not from external RGB input) has its starting address from which it reads data. This address is set by configuring VHDL generic parameters `C_VMEM_BASEADDR` and `C_LAYER_X_OFFSET`.

`C_VMEM_BASEADDR` is the base address of the whole logiCVC-ML core. Using `C_LAYER_X_OFFSET` parameters every layer can be configured to read from its own base address. In this way, layers can be placed on the desired locations in memory. All this parameters are configured before FPGA synthesis.

Additionally, offset register is used to move the starting point in memory from which the data will be read, i.e. the first pixel on display. For more information on memory offset registers, please refer to chapter 11.2.16 Layer Memory Offset Registers - LX\_H\_OFFSET, LX\_V\_OFFSET.

Each layer that reads data from video buffer has its double and triple buffer in video memory. Switching of buffers is controlled by VBUFF\_SEL register and is executed by memory address generator at the beginning of a new frame. In this way, flicker-free reproduction is guaranteed. For more on double/triple buffering, refer to chapter 11.2.8 Video Buffer Select Register - VBUFF\_SEL.

## 10.2.2 Pixel Data FIFO

Each layer that reads data from video buffer has one FIFO. That FIFO is used to temporarily buffer pixel data which is read from video memory on memory clock and output data on pixel clock (video clock). Usually memory clock is higher than pixel clock. Depending on VHDL generic parameters, FIFO can be configured in many combinations. The parameters that determine the width and size of FIFO are C\_LAYER\_X\_DATA\_WIDTH, C\_LAYER\_X\_ALPHA\_MODE and memory interface data width (C\_XMB\_DATA\_BUS\_WIDTH or C\_MPLB\_DWIDTH or C\_M\_AXI\_DATA\_WIDTH depending which memory interface is used XMB, PLB or AXI).

## 10.2.3 Colour 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 colour and every indexed colour in CLUT can have its own alpha value. For more on CLUT alpha blending mode, please refer to chapter 10.3.2.3 Colour Alpha Blending.

**Table 10.1: Data organisation of one CLUT index**

CLUT data bit	16bpp CLUT	24bpp CLUT
0	Dummy	Blue0
1	Dummy	Blue1
2	Dummy	Blue2
3	Blue0	Blue3
4	Blue1	Blue4
5	Blue2	Blue5
6	Blue3	Blue6
7	Blue4	Blue7
8	Dummy	Green0
9	Dummy	Green1
10	Green0	Green2

CLUT data bit	16bpp CLUT	24bpp CLUT
11	Green1	Green3
12	Green2	Green4
13	Green3	Green5
14	Green4	Green6
15	Green5	Green7
16	Dummy	Red0
17	Dummy	Red1
18	Dummy	Red2
19	Red0	Red3
20	Red1	Red4
21	Red2	Red5
22	Red3	Red6
23	Red4	Red7
24	Alpha0	Alpha0
25	Alpha1	Alpha1
26	Alpha2	Alpha2
27	Alpha3	Alpha3
28	Alpha4	Alpha4
29	Alpha5	Alpha5
30	Dummy	Alpha6
31	Dummy	Alpha7

## 10.3 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 colour, layer sizes and positions, layers below layer 0, will or will not be visible.

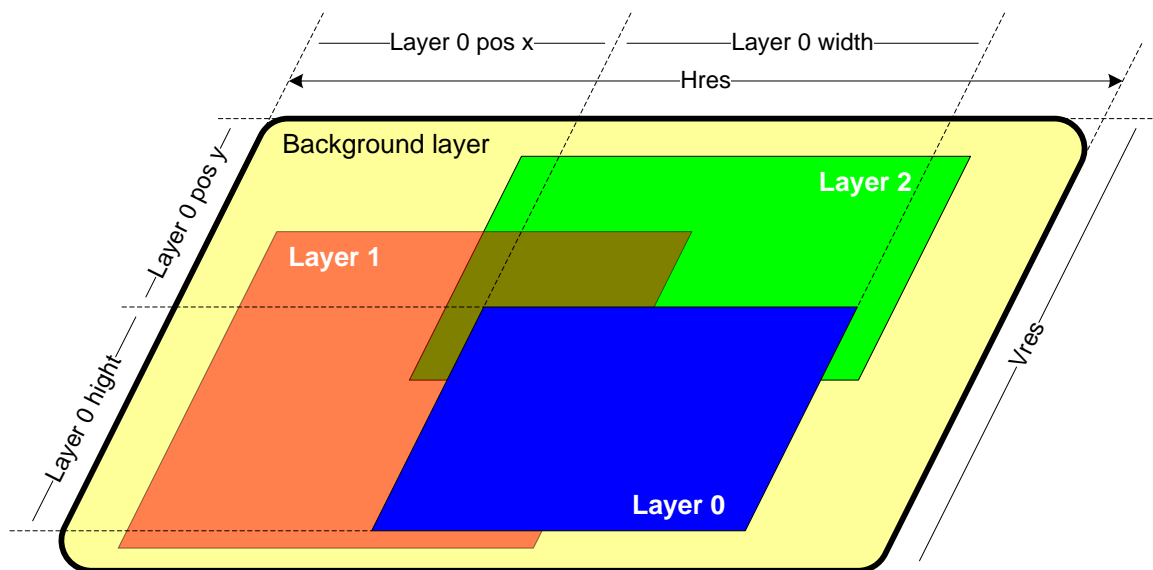
Figure 10.3 represents logiCVC-ML example configuration. In this example, logiCVC-ML has four layers where the last layer is configured as background layer. The difference between a normal layer and a background one is that background layer does not fetch data from the video memory but outputs data from the logiCVC-ML background colour register. To configure last layer as background user has to enable VHDL generic parameter C\_USE\_BACKGROUND.

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 seen 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 and is completely visible on the places that are not hidden by the upper layers.



Additionally, all layers are smaller than horizontal and vertical display resolution and are placed at some positions on the screen. Using positions 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 11.2.17 Layer Position Registers - LX\_H\_POSITION, LX\_V\_POSITION and 11.2.18 Layer Size Registers - LX\_WIDTH, LX\_HEIGHT.



**Figure 10.3 logiCVC-ML example configuration**

### 10.3.1 Transparency

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

Each layer has its own transparency colour register. Refer to chapter 11.2.21 Transparent Colour Register - LX\_TRANSPARENT for more information.

### 10.3.2 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 colour alpha blending using Colour 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 colour of every pixel on the screen for n-1 times, where n is the number of layers used.

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

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 colour.

For the example configuration represented on Figure 10.3, the following formulas would be used:

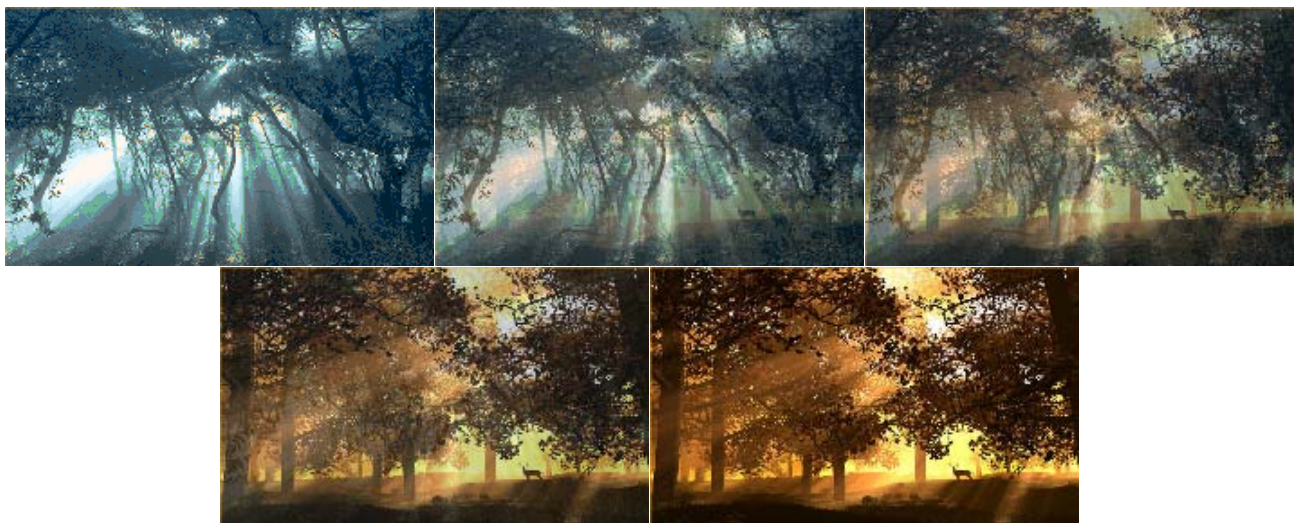
$$temp_{23} = \alpha_2 \times (layer_2 - layer_3) + layer_3$$

$$temp_{123} = \alpha_1 \times (layer_1 - temp_{23}) + temp_{23}$$

$$result = \alpha_0 \times (layer_0 - temp_{123}) + temp_{123}$$

### 10.3.2.1 Layer Alpha Blending

When using this mode alpha factor is stored in  $layer_x$  alpha register and is constant for all pixels on the specific layer. Figure 10.4 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 10.4 Layer alpha blending**

### 10.3.2.2 Pixel Alpha Blending

In contrast to layer alpha blending mode where one alpha factor is used for all pixels on that layer, pixel alpha mode uses one alpha factor for every pixel. To achieve this, every pixel in video memory is additionally extended for one byte where alpha factor for that pixel is stored.

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 from the graphic objects over live video.

The colour depth for each pixel is as follows:

**Table 10.2: Pixel Alpha blending**

Pixel colour depth	Colour value	Alpha value	Pixel width	Notes
8bpp	8-bit	3-bit	16-bit	
16bpp	16-bit	6-bit	32-bit	3 <sup>rd</sup> byte is dummy
24bpp	24-bit	8-bit	32-bit	

For example, when 16bpp colour depth is used with pixel alpha blending, two bytes represent pixel colour value, one byte represents alpha factor and one byte is dummy.

### 10.3.2.3 Colour Alpha Blending

Colour alpha blending is achieved by using Colour 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 colour and therefore object with same colour 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 colour. It is possible to have up to 256 different alpha factors. This alpha blending mode saves memory bandwidth due to keeping colour depth in video memory at 8-bit per pixel.

## 10.3.3 Layer Streams Synchronizations

Each layer can be synchronized with three possible sources:

1. Generation of graphics objects by the system CPU
2. Video input stream which is writing data to video memory buffer
3. External RGB input stream that provides data to one layer

### 10.3.3.1 CPU Synchronization

The generation of graphics objects by the system CPU needs to be synchronized with display refresh. This synchronization is achieved with notification (interrupt signal or interrupt status register) of display vertical retrace and/or double/triple buffering of layers with CPU generated graphic. The period of vertical retrace in which the CPU or 2D graphic accelerator draws (puts) graphic object to on-screen video buffer is relatively short in comparison to the whole frame refresh. Using the double/triple buffer feature almost non-stop flicker-free CPU or 2D graphic engine access to the on-screen video buffer is assured. In addition to the layer double/triple buffering, the CLUT is double buffered.

For more on using double/triple buffering using CPU, please refer to chapters 11.2.8 Video Buffer Select Register - VBUFF\_SEL, 11.2.9 CLUT Select Register - CLUT\_SEL and 11.2.10 Interrupt Status Register – INT.

### 10.3.3.2 Video Input, Writing to Memory Buffer, Synchronization

In order to provide a flicker-free, no artefacts display image, the display refresh rate and video input streams have to be synchronized. This is achieved by using triple buffers for video input and video

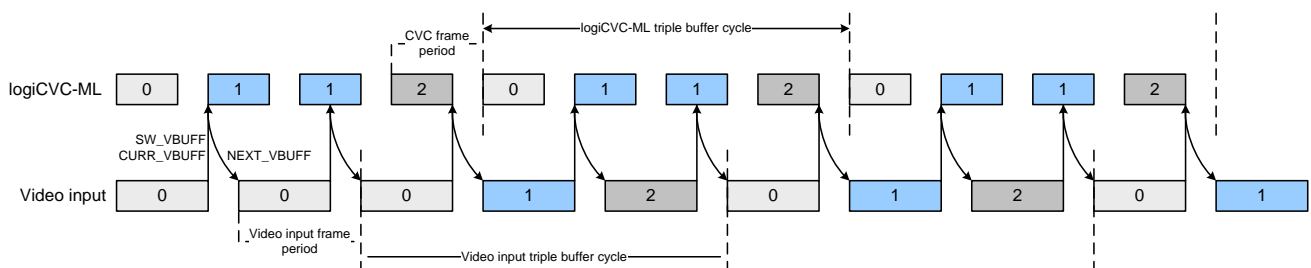
output. Triple buffer assures that the current display refresh buffer in memory would not be overwritten by the video input, i.e. that video input writing pointer to the video buffer would not overrun display refresh reading pointer. The effect of non-synchronized layer streams error would be shown as slow scrolling horizontal line of the screen.

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  $E\_SWITCH\_GRANT[C\_NUM\_OF\_LAYERS-1:0]$ .

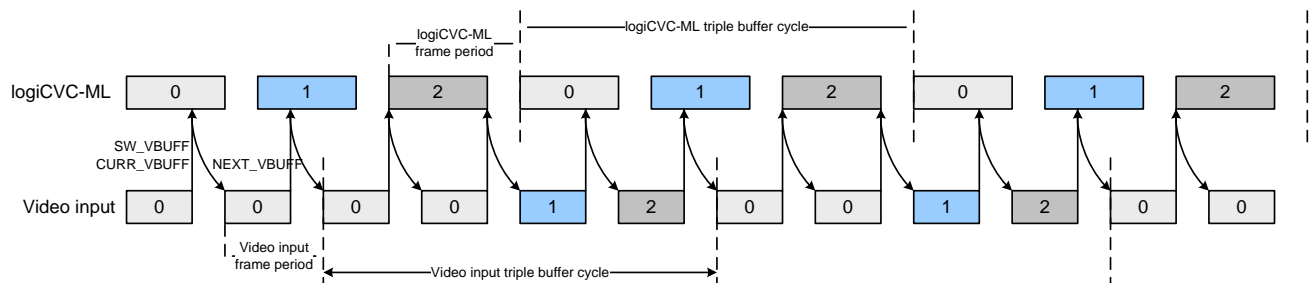
With the input signals ( $E\_CURRENT\_VBUFF[n*2+1:n*2]$  and  $E\_SWITCH\_VBUFF[n]$ ) external video source signals to logiCVC-ML layer n on which buffer it is currently writing data and when to switch buffers (typically on the end of its active frame of external video source). With output signal  $E\_SWITCH\_GRANT[C\_NUM\_OF\_LAYERS-1:0]$  logiCVC-ML grants a switch to the video source to start writing its next frame to  $E\_NEXT\_VBUFF[n*2+1:n*2]$ .

To enable video input 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 11.2.20 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 selected in the  $VBUFF\_SEL$  register. For more information on CPU synchronization mechanism, please refer to chapter 10.3.3.1 CPU Synchronization.



**Figure 10.5 Triple buffering example when logiCVC-ML refresh rate is higher than Video Input**



**Figure 10.6 Triple buffering example when logiCVC-ML refresh rate is lower than Video Input**

## 11 REGISTERS

logiCVC-ML registers are accessed through the OPB Slave, PLBv4.6 Slave or AXI4-Lite Slave data interface by Microblaze, PowerPC or any other OPB/PLB/AXI Master (logiSPI, Ext CPU OPB/PLB/AXI master,...).

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 setting the VHDL generic parameter C\_READABLE\_REGS to 0.

### 11.1 Register Map

All registers are placed at an offset set by VHDL generic parameter C\_REGS\_BASEADDR.

Following table describes the names and addresses of logiCVC-ML registers on the OPB/PLB/AXI bus.

**Table 11.1: logiCVC-ML register map**

Address offset <sup>1)</sup>	Type <sup>2)</sup>	Size (bits)	Name	Description
<b>General Registers</b>				
0x0000	R/W	8	HSY_FP	Horizontal Sync front porch
0x0008	R/W	8	HSY	Horizontal Sync pulse width
0x0010	R/W	8	HSY_BP	Horizontal Sync back porch
0x0018	R/W	16	H_RES	Horizontal resolution
0x0020	R/W	8	VSF_FP	Vertical Sync front porch
0x0028	R/W	8	VSF	Vertical Sync pulse width
0x0030	R/W	8	VSF_BP	Vertical Sync back porch
0x0038	R/W	16	V_RES	Vertical resolution
0x0040	R/W	24	CTRL	Control
0x0048	R/W	16	DTYPE	Display type
0x0050	R/W	24	BACKGROUND	Background colour
0x0058	R/W	16	VBUFF_SEL	Video buffer select for all layers <sup>4)</sup>
0x0060	R/W	16	CLUT_SEL	CLUT select for all CLUT layers <sup>4)</sup>
0x0068	R/(W)	16	INT	Interrupt status register <sup>3)</sup>
0x0070	R/W	16	INT_MASK	Interrupt mask register
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

Address offset <sup>1)</sup>	Type <sup>2)</sup>	Size (bits)	Name	Description
0x00F8	R	32	IP_VERSION	logiCVC-ML IP version information
<b>Layer 0 Registers</b>				
0x0100	R/W	16	L0_H_OFFSET	Layer0 horizontal memory offset
0x0108	R/W	16	L0_V_OFFSET	Layer0 vertical memory offset
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	8	L0_ALPHA	Layer0 layer alpha factor
0x0138	R/W	8	L0_CTRL	Layer0 Control
0x0140	R/W	24	L0_TRANSPARENT	Layer0 colour key transparency value
<b>Layer 1 Registers</b>				
0x0180	R/W	16	L1_H_OFFSET	Layer1 horizontal memory offset
0x0188	R/W	16	L1_V_OFFSET	Layer1 vertical memory offset
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	8	L1_ALPHA	Layer1 layer alpha factor
0x01B8	R/W	8	L1_CTRL	Layer1 Control
0x01C0	R/W	24	L1_TRANSPARENT	Layer1 colour key transparency value
<b>Layer 2 Registers</b>				
0x0200	R/W	16	L2_H_OFFSET	Layer2 horizontal memory offset
0x0208	R/W	16	L2_V_OFFSET	Layer2 vertical memory offset
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	8	L2_ALPHA	Layer2 layer alpha factor
0x0238	R/W	8	L2_CTRL	Layer2 Control
0x0240	R/W	24	L2_TRANSPARENT	Layer2 colour key transparency value
<b>Layer 3 Registers</b>				
0x0280	R/W	16	L3_H_OFFSET	Layer3 horizontal memory offset
0x0288	R/W	16	L3_V_OFFSET	Layer3 vertical memory offset
0x0290	R/W	16	L3_H_POSITION	Layer3 horizontal position



Address offset <sup>1)</sup>	Type <sup>2)</sup>	Size (bits)	Name	Description
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	8	L3_ALPHA	Layer3 layer alpha factor
0x02B8	R/W	8	L3_CTRL	Layer3 Control
0x02C0	R/W	24	L3_TRANSPARENT	Layer3 colour key transparency value
<b>Layer 4 Registers</b>				
0x0300	R/W	16	L4_H_OFFSET	Layer4 horizontal memory offset
0x0308	R/W	16	L4_V_OFFSET	Layer4 vertical memory offset
0x0338	R/W	8	L4_CTRL	Layer4 Control
<b>CLUT Registers</b>				
0x1000	R/W	1k Bytes	L0_CLUT_0	Layer0 Colour Look-Up Table 0
0x1800	R/W	1k Bytes	L0_CLUT_1	Layer0 Colour Look-Up Table 1
0x2000	R/W	1k Bytes	L1_CLUT_0	Layer1 Colour Look-Up Table 0
0x2800	R/W	1k Bytes	L1_CLUT_1	Layer1 Colour Look-Up Table 1
0x3000	R/W	1k Bytes	L2_CLUT_0	Layer2 Colour Look-Up Table 0
0x3800	R/W	1k Bytes	L2_CLUT_1	Layer2 Colour Look-Up Table 1
0x4000	R/W	1k Bytes	L3_CLUT_0	Layer3 Colour Look-Up Table 0
0x4800	R/W	1k Bytes	L3_CLUT_1	Layer3 Colour Look-Up Table 1
0x5000	R/W	1k Bytes	L4_CLUT_0	Layer4 Colour Look-Up Table 0
0x5800	R/W	1k Bytes	L4_CLUT_1	Layer4 Colour Look-Up Table 1

**NOTE:**

1. All registers are placed at the 64-bit boundary; including CLUT indexes.
2. Readability off all logiCVC-ML registers can be turned OFF with VHDL generic parameter C\_READABLE\_REGS set to 0.
3. Interrupt status register can only be cleared. Refer to chapter 11.2.10 Interrupt Status Register – INT for more information.
4. Video buffer select and CLUT select registers have different read/write values. Please refer to chapters 11.2.8 Video Buffer Select Register - VBUFF\_SEL and 11.2.9 CLUT Select Register - CLUT\_SEL for more information.

## 11.2 Register Description

### 11.2.1 Horizontal Sync and Porches - HSY\_FP, HSY, HSY\_BP

The values in registers HSY\_FP, HSY and HSY\_BP should be written in the number of VCLK periods. Please note that sum of HSY\_FP, HSY and HSY\_BP must be equal to pixel clock period increments.

**Table 11.2: HSY\_FP Register Description**

Bits	Name	Type	Reset value	Description
7 - 0	HSY_FP	R/W	0x7F	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 11.3: HSY Register Description**

Bits	Name	Type	Reset value	Description
7 - 0	HSY	R/W	0x7F	Horizontal sync

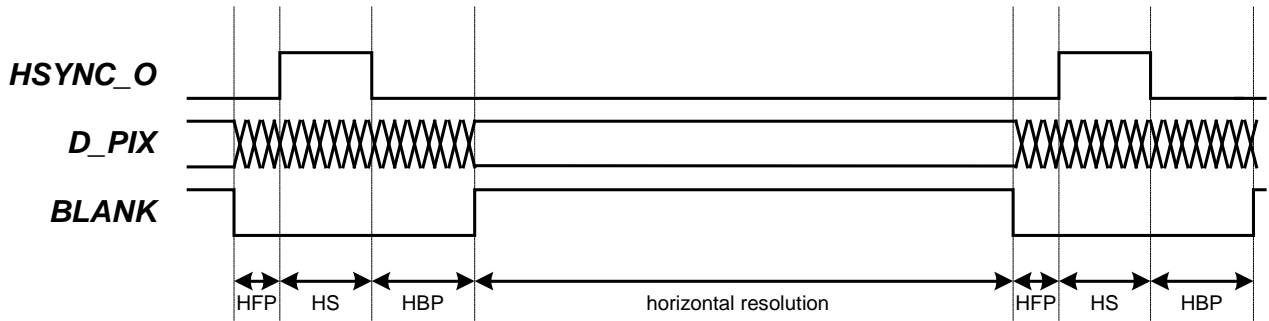
The duration of the HSYNC signal equals the value written into the HSY register incremented by 1 in VCLK periods.

**Table 11.4: HSY\_BP Register Description**

Bits	Name	Type	Reset value	Description
7 - 0	HSY_BP	R/W	0x7F	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.





**Figure 11.1: Horizontal sync and horizontal sync porches**

**NOTE:**

1. CTRL register controls the horizontal sync and porches usage. Refer to chapter 11.2.5 Display Control Register – CTRL for more information.
2. HSY\_FP, HSY and HSY\_BP cannot be assigned if ITU656 output standard is used (C\_DISPLAY\_INTERFACE = 1) because they are predefined by the standard itself.

## 11.2.2 Horizontal Resolution - H\_RES

**Table 11.5: H\_RES Register Description**

Bits	Name	Type	Reset value	Description
15 - 0	H_RES	R/W	0x027F	Horizontal resolution

logiCVC-ML supports horizontal resolutions from 64 to 2048 pixels but the maximum horizontal resolution, i.e. the size of horizontal resolution register, is constrained by the C\_ROW\_STRIDE parameter. 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.

**NOTE:**

1. H\_RES cannot be assigned if ITU656 output standard is used (C\_DISPLAY\_INTERFACE = 1) because it is predefined by the standard itself and NTSC\_PAL flag in DTYPE register.
2. Please check C\_ROW\_STRIDE setting for proper functioning. Usually, C\_ROW\_STRIDE should be set to maximum horizontal resolution used. For more information on memory stride please refer to chapter 4.1 Pixel Row Stride.

## 11.2.3 Vertical Sync and Porches - 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 11.6: VSY\_FP Register Description**

Bits	Name	Type	Reset value	Description
7 - 0	VSY_FP	R/W	0x00	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 11.7: VSY Register Description**

Bits	Name	Type	Reset value	Description
7 - 0	VSY	R/W	0x00	Vertical Sync Register

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

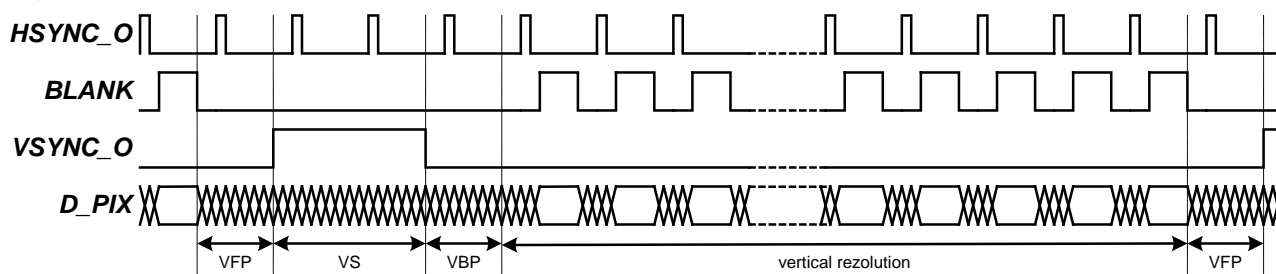
The duration of the VSYNC signal equals the value written in the VSY register incremented by 1 in pixels rows (lines).

**Table 11.8: VSY\_BP Register Description**

Bits	Name	Type	Reset value	Description
7 - 0	VSY_BP	R/W	0x00	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).



**Figure 11.2: Vertical sync and vertical sync porches**

**NOTE:**

1. CTRL register controls the vertical sync and porches usage. Refer to chapter 11.2.5 Display Control Register – CTRL for more information.
2. VSY\_FP, VSY and VSY\_BP cannot be assigned if ITU656 output standard is used (C\_DISPLAY\_INTERFACE = 1) because they are predefined by the standard itself.

## 11.2.4 Vertical Resolution - V\_RES

**Table 11.9: V\_RES Register Description**

Bits	Name	Type	Reset value	Description
15 - 0	V_RES	R/W	0x01DF	Vertical Resolution Register

logiCVC-ML supports vertical resolutions from 1 to 2048 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.

**NOTE:**

1. V\_RES cannot be assigned if ITU656 output standard is used (C\_DISPLAY\_INTERFACE = 1) because it is predefined by the standard itself and NTSC\_PAL flag in DTYPE register.

## 11.2.5 Display Control Register – CTRL

**Table 11.10: CTRL Register Description**

Bits	Name	Type	Reset value	Description
15 - 0	CTRL	R/W	0x0000	Display Control Register

The value written into CTRL register enables, disables and inverts the polarity of the control video signals. Additionally, CTRL register selects a VCLK source, enables an inversion of the PIX\_CLK signal (by default logiCVC-ML outputs data on falling edge of PIX\_CLK) and defines whether the PIX\_CLK is enabled during the horizontal back and front porch and HSYNC signal time.

**Table 11.11: Display 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	HSY_CLK	R/W	0	PIX_CLK disable during HSYNC active (0 - enabled, 1 - disabled)
10				Not used

11	VCDIVSEL0	R/W	0	Divide constant select bit 0 (See Table 11.12)
12	VCDIVSEL1	R/W	0	Divide constant select bit 1 (See Table 11.12)
13	VCLKSEL0	R/W	0	Clock frequency select bit 0 (See Table 11.13)
14	VCLKSEL1	R/W	0	Clock frequency select bit 1 (See Table 11.13)
15	VCLKSEL2	R/W	0	Clock frequency select bit 2 (See Table 11.13)
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

VCDIVSEL0&VCDIVSEL1: define the clock signal frequency division constant.

**Table 11.12 Clock Division Factor**

VCDIVSEL1	VCDIVSEL0	Division factor
0	0	1
0	1	2
1	0	4
1	1	8

**Note:** Only supported if external clock component is used

VCLKSEL0 & VCLKSEL1 & VCLKSEL2: define the clock signal frequency.

**Table 11.13 Clock Signal Frequency**

VCLKSEL2	VCLKSEL1	VCLKSEL0	Frequency
0	0	0	24 MHz
0	0	1	32 MHz
0	1	0	38 MHz
0	1	1	48 MHz
1	0	0	64 MHz
1	0	1	76 MHz
1	1	0	96 MHz

**Note:** Only supported if external clock component is used

Bits VCDIVSEL and VCLKSEL are directly routed towards the logiCVC-ML output pins. In this way, user has to make sure that there is a special clock configuration logic present that will make it possible for these configurations to work.

The VCLKSEL and VCDIVSEL tables' show the values used in the reference design. A user is free to change the division factors and input clock frequencies as it pleases.

## 11.2.6 Display Type Register - DTYPE

**Table 11.14: DTYPE Register Description**

Bits	Name	Type	Reset value	Description
7 - 0	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 11.15: 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 with C\_DISPLAY\_INTERFACE VHDL generic parameter set to 1.

**Table 11.16: NTSC\_PAL Register Description**

NTSC_PAL	Description
0	PAL video standard
1	NTSC video standard

## 11.2.7 Background Colour Register – BACKGROUND

**Table 11.17: BACKGROUND Register Description**

Bits	Name	Type	Reset value	Description
23 - 0	BACKGROUND	R/W	0x000000	Background Colour Register

If VHDL generic parameter C\_USE\_BACKGROUND is set, last layer is configured as background and data is not read from video memory but from background colour register. If all alpha factors from the layers above the last one are set to 0, background colour will be visible. Also, if layer sizes of the layers above are smaller than display size background will be visible, see Figure 10.3.

Three colour depth modes are supported: 8bpp, 16bpp and 24bpp. If last layer is configured as 24bpp (C\_LAYER\_X\_DATA\_WIDTH = 24) or 8bpp with CLUT (24bpp) all 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.

## 11.2.8 Video Buffer Select Register - VBUFF\_SEL

Table 11.18: VBUFF\_SEL Register Description

Bits	Name	Type	Reset value	Description
15 - 0	VBUFF_SEL	R/W	0x0000	Video Buffer Select Register

This register controls which video buffer logiCVC-ML fetches from memory. As every layer has its own buffer, all of them can be controlled through this register.

Table 11.19: Video Buffer Select Register Map

Bits	Name	Type	Reset value	Description
0	L0_BUFF_SEL_0	R/W	0	Layer0 buffer select bit 0
1	L0_BUFF_SEL_1	R/W	0	Layer0 buffer select bit 1
2	L1_BUFF_SEL_0	R/W	0	Layer1 buffer select bit 0
3	L1_BUFF_SEL_1	R/W	0	Layer1 buffer select bit 1
4	L2_BUFF_SEL_0	R/W	0	Layer2 buffer select bit 0
5	L2_BUFF_SEL_1	R/W	0	Layer2 buffer select bit 1
6	L3_BUFF_SEL_0	R/W	0	Layer3 buffer select bit 0
7	L3_BUFF_SEL_1	R/W	0	Layer3 buffer select bit 1
8	L4_BUFF_SEL_0	R/W	0	Layer4 buffer select bit 0
9	L4_BUFF_SEL_1	R/W	0	Layer4 buffer select bit 1
10	L0_BUFF_SEL_WR_EN	R/W	0	Layer0 buffer select write enable
11	L1_BUFF_SEL_WR_EN	R/W	0	Layer1 buffer select write enable
12	L2_BUFF_SEL_WR_EN	R/W	0	Layer2 buffer select write enable
13	L3_BUFF_SEL_WR_EN	R/W	0	Layer3 buffer select write enable
14	L4_BUFF_SEL_WR_EN	R/W	0	Layer4 buffer select write enable
15				Not used

After creating graphic objects in non-active video buffer, i.e. not visualized on screen, i.e. not selected in this register, CPU writes to one of the video buffer select register bits an incremented value signalling to the logiCVC-ML to switch the corresponding layer buffer. After finishing reading current frame from video memory logiCVC-ML will switch the address pointer of the selected layer to the next buffer, as stated in the register, and set the corresponding bit in the INT register signalling

completion. The buffer is swapped on vertical retrace (VSYNC signal active), so there is no image flickering.

User can choose between using double or triple buffering depending on the application.

Additional write enable bits exist in the video buffer select register, which enable the user to change video buffer 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 3 to use buffer 1 it has to write 0x2040.

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

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

**NOTE:**

1. If VHDL generic parameter C\_USE\_BACKGROUND is set, last layer is not fetching data from memory so video buffer select cannot be used for that layer.

## 11.2.9 CLUT Select Register - CLUT\_SEL

**Table 11.20: CLUT\_SEL Register Description**

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

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

**Table 11.21: 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

Bits	Name	Type	Reset value	Description
14	L4_CLUT_SEL_WR_EN	R/W	0	Layer4 CLUT select write enable
15				Not used

Same as with video buffer select register, 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 signalling 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 layer to the next CLUT, as stated in the register, and set the corresponding bit in the INT register signalling completion. The CLUT is swapped on vertical retrace (VSYNC signal active), so there is no image flickering.

Unlike video buffer select, only double CLUT can be used.

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 video buffer logiCVC-ML uses currently not the value it has written. Only after the buffer is switched the same value will be read.

**NOTE:**

1. If VHDL generic parameter C\_USE\_BACKGROUND is set, last layer is not fetching data from memory or CLUT so CLUT select cannot be used for that layer.

## 11.2.10 Interrupt Status Register – INT

**Table 11.22: INT Register Description**

Bits	Name	Type	Reset value	Description
15 - 0	INT	R/(W)	0x0000	Interrupt Status Register

Interrupt status register is divided into three parts:

### Video buffer select status

Through these interrupt status bits, logiCVC-ML signals that it has performed an action of switching video buffer pointer as stated in the VBUFF\_SEL register.

### 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.

### V-Sync status

Bit 5 is used for signalling 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 RGB input valid status

This flag is used for signalling to the user interface that external RGB input was detected and its parameters, porches, syncs and resolutions are 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.

**Table 11.23: Interrupt Status Register Map**

Bits	Name	Type	Reset value	Description
0	L0_VBUFF_SW	R/W	0	Layer0 video buffer switched
1	L1_VBUFF_SW	R/W	0	Layer1 video buffer switched
2	L2_VBUFF_SW	R/W	0	Layer2 video buffer switched
3	L3_VBUFF_SW	R/W	0	Layer3 video buffer switched
4	L4_VBUFF_SW	R/W	0	Layer4 video buffer switched
5	V_SYNC	R/W	0	Vertical sync active
6	E_VIDEO_VALID	R/W	0	External RGB input valid
7		R/W	0	Not used
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

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

#### NOTE:

1. If VHDL generic parameter C\_USE\_BACKGROUND is set, last layer is not fetching data from memory or CLUT so video buffer switched or CLUT switched interrupt can not be used.
2. If a certain layer is disabled using Enable bit in Layer control register, that layer cannot generate Video buffer switched or CLUT buffer switched interrupt in the Interrupt status register.

### 11.2.11 Interrupt Mask Register - INT\_MASK

**Table 11.24: INT\_MASK Register Description**

Bits	Name	Type	Reset value	Description
15 - 0	INT_MASK	R/W	0xFFFF	Interrupt Mask Register

Same as interrupt status register, interrupt mask register is divided into three parts; Video buffer select interrupt mask, CLUT select interrupt mask and V-Sync interrupt mask.

**Table 11.25: Interrupt Mask Register Map**

Bits	Name	Type	Reset value	Description
0	L0_VBUFF_SW_INT_MASK	R/W	1	Layer0 video buffer switch interrupt mask
1	L1_VBUFF_SW_INT_MASK	R/W	1	Layer1 video buffer switch interrupt mask
2	L2_VBUFF_SW_INT_MASK	R/W	1	Layer2 video buffer switch interrupt mask
3	L3_VBUFF_SW_INT_MASK	R/W	1	Layer3 video buffer switch interrupt mask
4	L4_VBUFF_SW_INT_MASK	R/W	1	Layer4 video buffer switch interrupt mask
5	V_SYNC_INT_MASK	R/W	1	Vertical sync active interrupt mask
6	E_VIDEO_VALID_MASK	R/W	1	External RGB input valid interrupt mask
7				Not used
8	L0_CLUT_SW_INT_MASK	R/W	1	Layer0 CLUT switch interrupt mask
9	L1_CLUT_SW_INT_MASK	R/W	1	Layer1 CLUT switch interrupt mask
10	L2_CLUT_SW_INT_MASK	R/W	1	Layer2 CLUT switch interrupt mask
11	L3_CLUT_SW_INT_MASK	R/W	1	Layer3 CLUT switch interrupt mask
12	L4_CLUT_SW_INT_MASK	R/W	1	Layer4 CLUT switch interrupt mask
13				Not used
14				Not used
15				Not used

Writing '1' to one of the interrupt mask bits disables the corresponding bit in the interrupt status register of generating interrupt on the INTERRUPT pin of the logiCVC-ML.

## 11.2.12 Power Control Register – PWRCTRL

**Table 11.26: PWCTRL Register Description**

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

PWRCTRL register defines control signals for VDAC (for CRT display control) and video display power sequencing. Bits in this register are directly routed to the logiCVC-ML output signals.

Additionally, V\_EN bit enables the three stated D\_PIX, H\_SYNC, V\_SYNC, BLANK, PIX\_CLK output signals.

**Table 11.27: 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	V_EN	R/W	0	Enable display control and data signals (0 – disabled, 1 – enabled)
7 - 4				Not used

### 11.2.13 External Input Horizontal Resolution - E\_HRES

**Table 11.28: E\_HRES Register Description**

Bits	Name	Type	Reset value	Description
15 - 0	E_HRES	R	0x0000	External Input Horizontal Resolution Register

If external RGB input interface is enabled ( $C\_USE\_E\_RGB\_INPUT = 1$ ), this register holds the measured value of external RGB 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 VHDL generic parameter  $C\_ROW\_STRIDE$  in a way that the number of bits used in E\_HRES register is equal to  $\log_2(C\_ROW\_STRIDE)$ .

### 11.2.14 External Input Vertical Resolution - E\_VRES

**Table 11.29: E\_VRES Register Description**

Bits	Name	Type	Reset value	Description
15 - 0	E_VRES	R	0x0000	External Input Vertical Resolution Register

If external RGB input interface is enabled ( $C\_USE\_E\_RGB\_INPUT = 1$ ), this register holds the measured value of external RGB input vertical 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.

## 11.2.15 IP Version Register - IP\_VERSION

**Table 11.30: IP\_VERSION Register Description**

Bits	Name	Type	Reset value	Description
31 - 0	IP_VERSION	R	0x00001020	IP Version Register

Value stored in IP version register holds information about the version of the current logiCVC-ML IP. It is a 19-bit value, while the bits 20 to 32 are read as '0'. User cannot write to this register, and its content can only be read if VHDL parameter C\_READABLE\_REGS is set.

This value is calculated using VHDL parameters: C\_IP\_MAJOR\_REVISION, C\_IP\_MINOR\_REVISION, C\_IP\_PATCH\_LEVEL and C\_IP\_LICENSE\_TYPE.

**Table 11.31 logiCVC IP Version Information**

Bits	Designation
4 - 0	PATCH LEVEL
10 - 5	MINOR REVISION
16 - 11	MAJOR REVISION
18 - 17	LICENSE TYPE
31 - 19	Not used

**NOTE:**

- Value stored in this register depends on logiCVC-ML IP version. In the example above, the value is: 0x00001020, which stands for 2\_01\_a (source version).

## 11.2.16 Layer Memory Offset Registers - LX\_H\_OFFSET, LX\_V\_OFFSET

**Table 11.32: LX\_H\_OFFSET Register Description**

Bits	Address offset	Name	Type	Reset value	Description
15 - 0	0x0100	L0_H_OFFSET	R/W	0x0000	Layer 0 Horizontal Memory Offset
15 - 0	0x0180	L1_H_OFFSET	R/W	0x0000	Layer 1 Horizontal Memory Offset
15 - 0	0x0200	L2_H_OFFSET	R/W	0x0000	Layer 2 Horizontal Memory Offset
15 - 0	0x0280	L3_H_OFFSET	R/W	0x0000	Layer 3 Horizontal Memory Offset
15 - 0	0x0300	L4_H_OFFSET	R/W	0x0000	Layer 4 Horizontal Memory Offset

**Table 11.33: LX\_V\_OFFSET Register Description**

Bits	Address offset	Name	Type	Reset value	Description
15 - 0	0x0108	L0_V_OFFSET	R/W	0x0000	Layer 0 Vertical Memory Offset
15 - 0	0x0188	L1_V_OFFSET	R/W	0x0000	Layer 1 Vertical Memory Offset
15 - 0	0x0208	L2_V_OFFSET	R/W	0x0000	Layer 2 Vertical Memory Offset

Bits	Address offset	Name	Type	Reset value	Description
15 - 0	0x0288	L3_V_OFFSET	R/W	0x0000	Layer 3 Vertical Memory Offset
15 - 0	0x0308	L4_V_OFFSET	R/W	0x0000	Layer 4 Vertical Memory Offset

Layer memory offset is used for determining the starting point in the memory from which logiCVC-ML layer reads the first pixel. Horizontal offset is given as a number of pixels while vertical offset is given in the number of lines. Both values are set to zero by default.

If horizontal offset is increased, the picture will be moved to the left. The number of pixels read from the memory is determined by adjusting the horizontal size. If display controller has reached the end of the line in memory, and if there are still pixels to be sent towards the display, the display controller will read those additional pixels starting from the beginning of the same line. This "beginning", however, is not the "beginning" set by the horizontal offset, but the default line starting address (offset = 0).

If vertical offset is increased, the picture will be moved up. The number of lines read from the memory is determined by adjusting the vertical size.

**NOTE:**

1. In order to avoid image glitches when changing size, position or offset parameters, it is essential that all new values become active at the same time. Therefore, in order to activate any new size, position or offset value, the software has to perform write into the lower part of the vertical position register. If several size and position values have to be changed, it is required to perform a write to the vertical position register only once at the end of the update procedure.
2. If VHDL generic parameter C\_USE\_BACKGROUND is set, last layer is not fetching data from memory so layer memory offset cannot be altered for that layer.
3. To use layer memory offset, VHDL generic parameter C\_USE\_SIZE\_POS has to be enabled.

## 11.2.17 Layer Position Registers - LX\_H\_POSITION, LX\_V\_POSITION

**Table 11.34: LX\_H\_POSITION Register Description**

Bits	Address offset	Name	Type	Reset value	Description
15 - 0	0x0110	L0_H_POSITION	R/W	0x0000	Layer 0 Horizontal Position
15 - 0	0x0190	L1_H_POSITION	R/W	0x0000	Layer 1 Horizontal Position
15 - 0	0x0210	L2_H_POSITION	R/W	0x0000	Layer 2 Horizontal Position
15 - 0	0x0290	L3_H_POSITION	R/W	0x0000	Layer 3 Horizontal Position

**Table 11.35: LX\_V\_POSITION Register Description**

Bits	Address offset	Name	Type	Reset value	Description
15 - 0	0x0118	L0_V_POSITION	R/W	0x0000	Layer 0 Vertical Position
15 - 0	0x0198	L1_V_POSITION	R/W	0x0000	Layer 1 Vertical Position
15 - 0	0x0218	L2_V_POSITION	R/W	0x0000	Layer 2 Vertical Position

Bits	Address offset	Name	Type	Reset value	Description
15 - 0	0x0298	L3_V_POSITION	R/W	0x0000	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.

**Table 11.36: Layer Horizontal Position Register Map**

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

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.

**Table 11.37: Layer Vertical Position Register Map**

Bits	Name	Type	Reset value	Description
0	VER_POS_0	R/W	0	Layer vertical position bit 0 (Note 4)
1	VER_POS_1	R/W	0	Layer vertical position bit 1 (Note 4)
2	VER_POS_2	R/W	0	Layer vertical position bit 2 (Note 4)
3	VER_POS_3	R/W	0	Layer vertical position bit 3 (Note 4)

Bits	Name	Type	Reset value	Description
4	VER_POS_4	R/W	0	Layer vertical position bit 4 (Note 4)
5	VER_POS_5	R/W	0	Layer vertical position bit 5 (Note 4)
6	VER_POS_6	R/W	0	Layer vertical position bit 6 (Note 4)
7	VER_POS_7	R/W	0	Layer vertical position bit 7 (Note 4)
8	VER_POS_8	R/W	0	Layer vertical position bit 8 (Note 4)
9	VER_POS_9	R/W	0	Layer vertical position bit 9 (Note 4)
10	VER_POS_10	R/W	0	Layer vertical position bit 10 (Note 4)
15 - 11				Not used

When adjusting vertical position it has to watch that that the number representing the vertical position has to be equal or greater than the number representing the vertical size.

**NOTE:**

1. In order to avoid image glitches when changing size, position or offset parameters, it is essential that all new values become active at the same time. Therefore, in order to activate any new size, position or offset value, the software has to perform write into the lower part of the vertical position register. If several size and position values have to be changed, it is required to perform a write to the vertical position register only once at the end of the update procedure.
2. Please note that the last layer's position cannot be altered 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).
3. To use layer position, VHDL generic parameter C\_USE\_SIZE\_POS has to be enabled.
4. Maximum value of horizontal and vertical layer position is constrained by configured logiCVC-ML resolution. As the maximum resolution is constrained by VHDL generic parameter C\_ROW\_STRIDE the number of bits used from layer position registers is equal to  $\log_2(C\_ROW\_STRIDE)$ .

## 11.2.18 Layer Size Registers - LX\_WIDTH, LX\_HEIGHT

**Table 11.38: LX\_WIDTH Register Description**

Bits	Address offset	Name	Type	Reset value	Description
15 - 0	0x0120	L0_WIDTH	R/W	0x0000	Layer 0 Width
15 - 0	0x01A0	L1_WIDTH	R/W	0x0000	Layer 1 Width
15 - 0	0x0220	L2_WIDTH	R/W	0x0000	Layer 2 Width
15 - 0	0x02A0	L3_WIDTH	R/W	0x0000	Layer 3 Width

**Table 11.39: LX\_HEIGHT Register Description**

Bits	Address offset	Name	Type	Reset value	Description
15 - 0	0x0128	L0_HEIGHT	R/W	0x0000	Layer 0 Height
15 - 0	0x01A8	L1_HEIGHT	R/W	0x0000	Layer 1 Height



Bits	Address offset	Name	Type	Reset value	Description
15 - 0	0x0228	L2_HEIGHT	R/W	0x0000	Layer 2 Height
15 - 0	0x02A8	L3_HEIGHT	R/W	0x0000	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 11.40: Layer Width Register Map**

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

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 11.41: Layer Height Register Map**

Bits	Name	Type	Reset value	Description
0	HEIGHT_0	R/W	0	Layer height bit 0 (Note 4)
1	HEIGHT_1	R/W	0	Layer height bit 1 (Note 4)
2	HEIGHT_2	R/W	0	Layer height bit 2 (Note 4)
3	HEIGHT_3	R/W	0	Layer height bit 3 (Note 4)
4	HEIGHT_4	R/W	0	Layer height bit 4 (Note 4)
5	HEIGHT_5	R/W	0	Layer height bit 5 (Note 4)
6	HEIGHT_6	R/W	0	Layer height bit 6 (Note 4)



Bits	Name	Type	Reset value	Description
7	HEIGHT_7	R/W	0	Layer height bit 7 (Note 4)
8	HEIGHT_8	R/W	0	Layer height bit 8 (Note 4)
9	HEIGHT_9	R/W	0	Layer height bit 9 (Note 4)
10	HEIGHT_10	R/W	0	Layer height bit 10 (Note 4)
15 - 11				Not used

**NOTE:**

1. In order to avoid image glitches when changing size, position or offset parameters, it is essential that all new values become active at the same time. Therefore, in order to activate any new size, position or offset value, the software has to perform write into the lower part of the vertical position register. If several size and position values have to be changed, it is required to perform a write to the vertical position register only once at the end of the update procedure.
2. Please note that the last layer's size cannot be altered 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).
3. To use layer size, VHDL generic parameter C\_USE\_SIZE\_POS has to be enabled.
4. Maximum value of width and height is constrained by configured logiCVC-ML resolution. As the maximum resolution is constrained by VHDL generic parameter C\_ROW\_STRIDE, the number of bits used from width and height registers is equal to  $\log_2(C\_ROW\_STRIDE)$ .

## 11.2.19 Layer Alpha Register - LX\_ALPHA

**Table 11.42: LX\_ALPHA Register Description**

Bits	Address offset	Name	Type	Reset value	Description
7 - 0	0x0130	L0_ALPHA	R/W	0xFF	Layer 0 Alpha Value
7 - 0	0x01B0	L1_ALPHA	R/W	0xFF	Layer 1 Alpha Value
7 - 0	0x0230	L2_ALPHA	R/W	0xFF	Layer 2 Alpha Value
7 - 0	0x02B0	L3_ALPHA	R/W	0xFF	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 (C\_LAYER\_X\_ALPHA\_MODE = 0).

Three colour depth modes are supported: 8bpp, 16bpp and 24bpp. 24bpp colour depth mode requires use of all 8 bits. In 16bpp mode, least significant 5 or 6 bits are used (bits 4-0 or bits 5-0). Red and blue colour use 5 bits, while green uses 6 bits. 8bpp mode uses least significant 3 or 2 bits (bits 2-0 or bits 1-0). Red and green colour use 3 bits, while blue uses 2 bits.

For more on alpha blending please refer to chapter 10.3.2 Alpha Blending.

**Table 11.43: Layer Alpha Register Map**

Bits	Name	Colour Depth	Description
------	------	--------------	-------------

		8bpp	16bpp	24bpp	
0	ALPHA_0	ALPHA_0	ALPHA_0	ALPHA_0	Layer alpha factor bit 0
1	ALPHA_1	ALPHA_1	ALPHA_1	ALPHA_1	Layer alpha factor bit 1
2	ALPHA_2	ALPHA_2	ALPHA_2	ALPHA_2	Layer alpha factor bit 2
3	ALPHA_3	Not used	ALPHA_3	ALPHA_3	Layer alpha factor bit 3
4	ALPHA_4	Not used	ALPHA_4	ALPHA_4	Layer alpha factor bit 4
5	ALPHA_5	Not used	ALPHA_5	ALPHA_5	Layer alpha factor bit 5
6	ALPHA_6	Not used	Not used	ALPHA_6	Layer alpha factor bit 6
7	ALPHA_7	Not used	Not used	ALPHA_7	Layer alpha factor bit 7

**NOTE:**

1. Please note that the last layer's alpha factor cannot be altered (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).

## 11.2.20 Layer Control Register - LX\_CTRL

**Table 11.44: 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 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.

Layer Control register's DIS\_TRANSP bit is used for disabling colour key transparency per layer. By setting this bit, logiCVC-ML will not perform a comparison between pixel colour and the value stored in the Transparent Colour register and pixel will be seen (if alpha value is set appropriately).

Layer Control register's EN\_EXT\_VBUFF\_SW bit is used for enabling external switching of video 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 10.3.3.2 Video Input, Writing to Memory Buffer, Synchronization.

Layer Control register's 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.

**Table 11.45: Layer Control Register Map**

Bits	Name	Type	Reset value	Description
0	ENABLE	R/W	0 <sup>1)</sup>	Layer ON/OFF bit
1	DIS_TRANSP	R/W	0	Disable Colour Transparency bit
2	EN_EXT_VBUFF_SW	R/W	0	Enable external video buffer switching
3	INTERLACED	R/W	0	Interlaced/progressive memory reading
7 - 4				Not used

**NOTE:**

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

## 11.2.21 Transparent Colour Register - LX\_TRANSPARENT

**Table 11.46: LX\_TRANSPARENT Register Description**

Bits	Address offset	Name	Type	Reset value	Description
23 - 0	0x0140	L0_TRANSPARENT	R/W	0x000000	Layer 0 Transparent Colour
23 - 0	0x01C0	L1_TRANSPARENT	R/W	0x000000	Layer 1 Transparent Colour
23 - 0	0x0240	L2_TRANSPARENT	R/W	0x000000	Layer 2 Transparent Colour
23 - 0	0x02C0	L3_TRANSPARENT	R/W	0x000000	Layer 3 Transparent Colour

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

Three colour depths are supported: 8bpp, 16bpp and 24bpp. 8bpp with CLUT(24bpp) and 24bpp colour depth mode requires use of all three bytes, while 8bpp with CLUT(16bpp) or 16bpp mode requires two bytes for determining transparent colour. For normal 8bpp layer colour mode, only the lowest byte is used.

**Table 11.47: Layer Transparent Colour Register Map**

Bits	Name	Colour Depth			Description
		8bpp	16bpp	24bpp	
0	TRANSP_0	TRANSP_0	TRANSP_0	TRANSP_0	Layer transparent colour bit 0
1	TRANSP_1	TRANSP_1	TRANSP_1	TRANSP_1	Layer transparent colour bit 1
2	TRANSP_2	TRANSP_2	TRANSP_2	TRANSP_2	Layer transparent colour bit 2
3	TRANSP_3	TRANSP_3	TRANSP_3	TRANSP_3	Layer transparent colour bit 3
4	TRANSP_4	TRANSP_4	TRANSP_4	TRANSP_4	Layer transparent colour bit 4
5	TRANSP_5	TRANSP_5	TRANSP_5	TRANSP_5	Layer transparent colour bit 5

Bits	Name	Colour Depth			Description
		8bpp	16bpp	24bpp	
6	TRANSP_6	TRANSP_6	TRANSP_6	TRANSP_6	Layer transparent colour bit 6
7	TRANSP_7	TRANSP_7	TRANSP_7	TRANSP_7	Layer transparent colour bit 7
8	TRANSP_8	Not used	TRANSP_8	TRANSP_8	Layer transparent colour bit 8
9	TRANSP_9	Not used	TRANSP_9	TRANSP_9	Layer transparent colour bit 9
10	TRANSP_0	Not used	TRANSP_10	TRANSP_10	Layer transparent colour bit 10
11	TRANSP_11	Not used	TRANSP_11	TRANSP_11	Layer transparent colour bit 11
12	TRANSP_12	Not used	TRANSP_12	TRANSP_12	Layer transparent colour bit 12
13	TRANSP_13	Not used	TRANSP_13	TRANSP_13	Layer transparent colour bit 13
14	TRANSP_14	Not used	TRANSP_14	TRANSP_14	Layer transparent colour bit 14
15	TRANSP_15	Not used	TRANSP_15	TRANSP_15	Layer transparent colour bit 15
16	TRANSP_16	Not used	Not used	TRANSP_16	Layer transparent colour bit 16
17	TRANSP_17	Not used	Not used	TRANSP_17	Layer transparent colour bit 17
18	TRANSP_18	Not used	Not used	TRANSP_18	Layer transparent colour bit 18
19	TRANSP_19	Not used	Not used	TRANSP_19	Layer transparent colour bit 19
20	TRANSP_20	Not used	Not used	TRANSP_20	Layer transparent colour bit 20
21	TRANSP_21	Not used	Not used	TRANSP_21	Layer transparent colour bit 21
22	TRANSP_22	Not used	Not used	TRANSP_22	Layer transparent colour bit 22
23	TRANSP_23	Not used	Not used	TRANSP_23	Layer transparent colour bit 23

**NOTE:**

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

## 12 INTEGRATION

This chapter describes how to synthesize 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.

### 12.1 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 through setting generics using Xilinx EDK tools. See chapter 2.3 logiCVC-ML Parameters for more information.

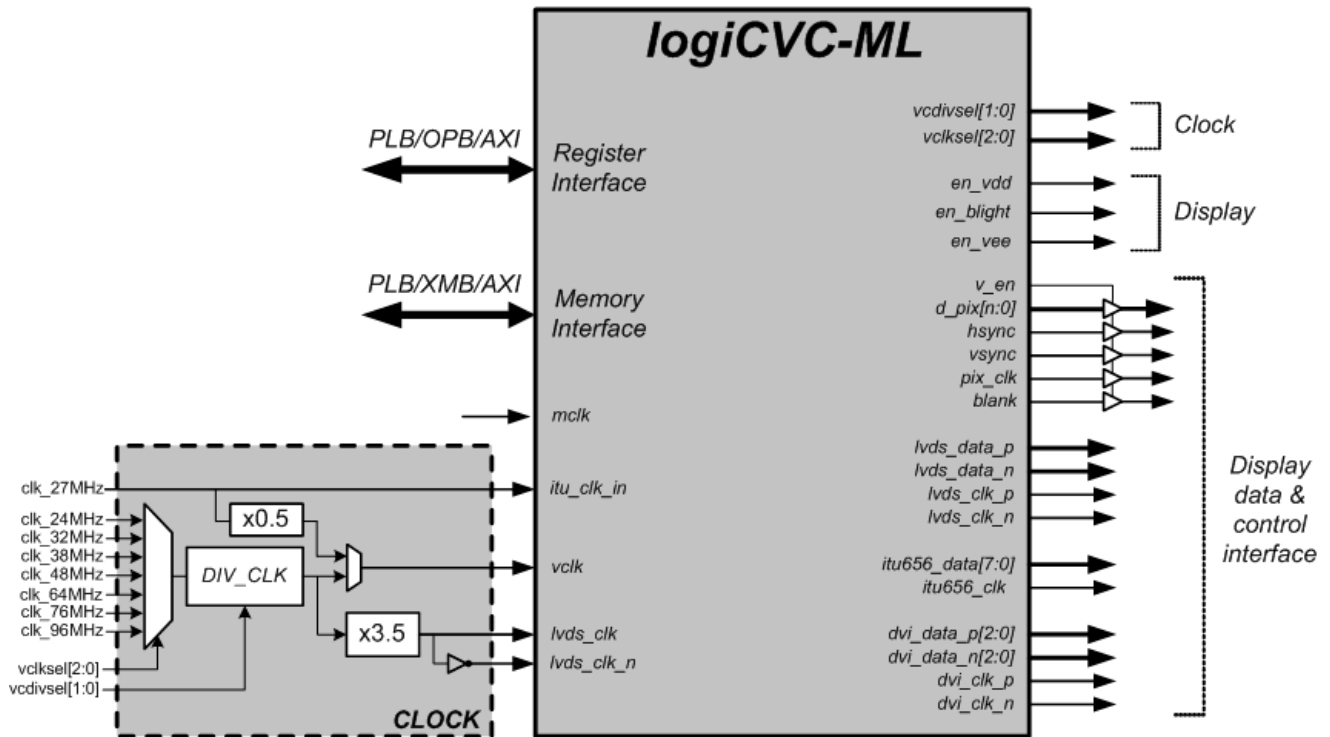
### 12.2 Implementation

Recommended logiCVC-ML FPGA implementation options:

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

### 12.3 Integration

Figure 12.1 shows recommended logiCVC-ML system integration. By enabling ITU656, LVDS, camera link interface or DVI user can change the configuration and connect one of the four output formats to the display or DAC required for the display. Additionally, it has to provide correct clock inputs.



**Figure 12.1: logiCVC-ML System Integration**

The reference design is a part of logiCVC-ML deliverables. It contains the Xilinx Platform Studio project with Microblaze, logiMEM - DDR PLB/XMB controller, OPB/PLB/AXI interface and DLL clock generators.

## 12.4 IO Interface Description

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. These clock signals should be connected to the logiCVC-ML clock inputs VCLK Video clock, MCLK MPLB/XMB/AXI clock for memory bus access, OPB\_CLK OPB clock, SPLB\_CLK PLB clock or S\_AXI\_ACLK for AXI clock for registers accesses, LVDS\_CLK and LVDS\_CLKN for LVDS output interface and ITU\_CLK\_IN for ITU656 output standard. All signals should have stable clock sources. Preferred method of clock generation is using Xilinx Virtex/Spartan Digital Clock Managers (DCMs) / Phase Locked Loops (PLLs).

The VCLK clock signal controls most of the circuits inside the logiCVC-ML core. If External RGB input is used (C\_USE\_E\_RGB\_INPUT = 1) and E\_VIDEO\_PRESENT is active, E\_VCLK is used instead of VCLK when C\_USE\_E\_VCLK\_BUFGMUX = 1. When C\_USE\_E\_VCLK\_BUFGMUX = 0 user has to instance BUFGMUX manually and provide VCLK accordingly to E\_VIDEO\_PRESENT. This is useful when LVDS or camera link interface is used.

Usually, the PLB/XMB/AXI clock frequency is higher than VCLK frequency. The sustained data rate between memory and logiCVC-ML should be assured by selecting an appropriate PLB/XMB/AXI frequency and memory bandwidth.

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 either by the OPB\_CLK OPB clock signal, by SPLB\_CLK PLB clock signal or by S\_AXI\_ACLK AXI clock signal depending on the used register interface, i.e. VHDL generic parameter C\_REGS\_INTERFACE.

## 12.4.1 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 OPB/PLB/AXI bus. Please consult Xilinx OPB, PLBv4.6 or AXI specification regarding the OPB, PLBv4.6 or AXI timing requirements.

The Video memory access is performed via PLBv4.6, XMB or AXI. Please consult Xilinx PLBv4.6/AXI specification or Xylon logiMEM specification regarding timing requirements

## 12.4.2 Control and Data Display Signals

All display signals are using IOB output FFs, therefore the HSYNC, VSYNC, BLANK, and D\_PIX[N:0] signals are synchronous with PIX\_CLK. There is a minimal negligible skew among PIX\_CLK and other signals.

## 12.5 Timing Constraints

The following timing specifications should be appended to the host design user constraint file (UCF).

**WARNING:** The logiCVC-ML related UCF timing constraints can be preceded by the higher priority timing constraints. For example: The PCF timing constraints have the higher priority over UCF constraints. If the logiCVC-ML related timing constraint is preceded, the implemented design may not work properly due to an erroneous timing constraint.

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

```
TIMESPEC TS_vclk = PERIOD "vclk" <HERE WRITE PERIOD OF vclk>;
```

If the host design uses more than one VCLK clock signal source, the value of the supplemented clock period should be the lowest one i.e. the highest clock frequency.

Memory clock and register clock (OPB, PLB or AXI) periods have to be supplemented in lines:

```
TIMESPEC TS_mclk = PERIOD "mclk" <HERE WRITE PERIOD OF mclk>;
```

```
TIMESPEC TS_opb_clk = PERIOD "opb_clk" <HERE WRITE PERIOD OF opb_clk>;
```

```
TIMESPEC TS_splb_clk = PERIOD "splb_clk" <HERE WRITE PERIOD OF
splb_clk>;
TIMESPEC TS_s_axi_aclk = PERIOD "s_axi_aclk " <HERE WRITE PERIOD OF
s_axi_aclk>;
```

If LVDS output is used, the following constraints must be added:

```
TIMESPEC "TS_lvds_clk" = PERIOD "lvds_clk" <HERE WRITE PERIOD OF
lvds_clk>;
TIMESPEC "TS_lvds_clkn" = PERIOD "lvds_clkn" "TS_lvds_clk" PHASE + <HERE
WRITE HALFPERIOD OF lvds_clk>;
TIMESPEC "TS_lvds_clk_vclk" = FROM "lvds_clk" TO "vclk" = TS_lvds_clk;
TIMESPEC "TS_lvds01" = FROM "vclk" to "lvds_clk" = TS_lvds_clk*2;
TIMESPEC "TS_lvds02" = FROM "lvds_clk" to ffs(logicvc*/**clk_div4) =
TS_lvds_clk*2 ;
TIMESPEC "TS_lvds03" = FROM "vclk" to ffs(logicvc*/**datain_d*) =
TS_lvds_clk/2;
```

ITU656 standard defines the ITU clock frequency to 27MHz so the following lines must be added:

```
NET "itu_clk_in" TNM_NET = "ITU_CLK_IN";
TIMESPEC "TS_ITU_CLK_IN" = PERIOD "ITU_CLK_IN" 37 ns;
```

If DVI output is used the following constraints must be added:

```
TIMEGRP "dram_out" = RAMS(*dvi_tx_inst/mux_2_to_1_inst/dram_out<*>);
TIMEGRP "dram_out_d" = FFS(*dvi_tx_inst/mux_2_to_1_inst/dram_out_d<*>);
TIMEGRP "read_addr" = FFS(*dvi_tx_inst/mux_2_to_1_inst/read_addr<*>);
TIMESPEC "TS_ramdo_0" = FROM "dram_out" TO "dram_out_d" TS_vclk;
TIMESPEC "TS_ramra_0" = FROM "read_addr" TO "dram_out_d" TS_vclk;
```



## Revision History

Version	Date	Author	Approved by	Note
1.01.a	28.7.2006.	J. Ivanović	J. Ivanović	Initial Xylon release
1.01.b	9.10.2006.	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 12bit multiplexed output mode, external RGB input.
1.04.e	27.02.2009 .	J. Ivanović	J. Ivanović	Only name change 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 endian 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 Colour 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	<p>Spartan6 and Virtex6 added to C_FAMILY.</p> <p>Parameters C_USE_LVDS and C_USE_ITU656 removed.</p> <p>C_DISPLAY_INTERFACE parameter added to replace them both and to add another display interface option: camera link.</p> <p>Version parameters added:</p> <p>C_IP_LICENSE_TYPE (IP encryption type);</p> <p>C_IP_MAJOR_REVISION;</p> <p>C_IP_MINOR_REVISION;</p> <p>C_IP_PATCH_LEVEL.</p> <p>In addition to these parameters there is a register named IP_VERSION that contains a value representing IP version.</p> <p>ODDR2 components instantiated for Spartan3a, Spartan3e and Spartan6.</p> <p>ODDR components instantiated for Virtex4, Virtex5 and Virtex6.</p>
1.05.b	21.12.2009	K. Mudrovčić	J. Ivanović	User's manual corrections.
1.06.a	11.1.2010.	R. Končurat	J. Ivanović	<p>User's manual corrections.</p> <p>Bug fix in parallel output interface for Virtex4, Virtex5 and Virtex6 families (ODDR components).</p> <p>Bug fix in external RGB data and video buffer width mismatch.</p>
1.06.a	11.1.2010.	A. Jukić	J. Ivanović	<p>New alpha blender.</p> <p>New generic parameters added:</p> <p>C_USE_XTREME_DSP and</p> <p>C_USE_MULTIPLIER.</p>
1.06.b	13.1.2010.	J. Ivanović	J. Ivanović	<p>Added support for 128 bit XMB/PLB bus.</p> <p>Added support for 32 and 64 burst length on 64 and 128 bit PLB bus.</p>
1.06.c	2.3.2010.	Z. Šafaržik	J. Ivanović	
1.06.c	19.3.2010.	J.Marjanović	J. Ivanović	<p>Added generic for usage of BUFGMUX for switching vclk to e_vclk</p> <p>Extended CTRL register for inverting polarity of external video control signals</p>
1.06.c	25.3.2010.	A.Cazin	J. Ivanović	Added support for pipelined memory access

Version	Date	Author	Approved by	Note
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
2.01.a	14.3.2011.	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	7.11.2011.	J. Marjanović	J. Ivanović	Added XCOLOR dithering module, added support for 7-series devices including LVDS and instructions for clocking.