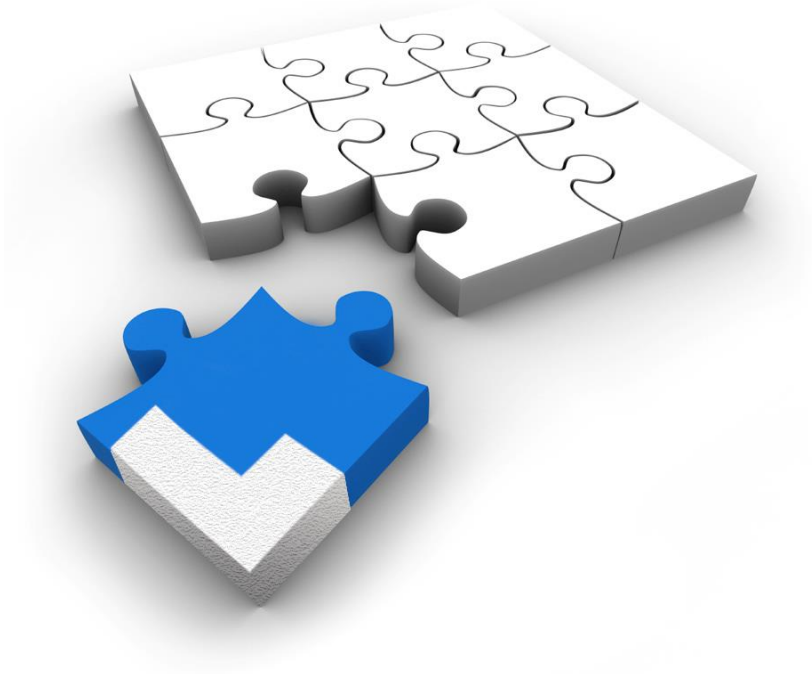
logiCVC-ML

User's Manual

Version: 4.1.2

logiCVC-ML_hum_v4.1.2.docx



1 INTRODUCTION

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

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

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

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

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

1.1 General Description

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

The interface to the frame buffer or the video memory is targeted for the SDRAM (SDR or DDR) or SRAM implementation. For easier system integration, logiCVC-ML uses Xilinx (IBM) CoreConnect PLBv4.6, OPB and AXI4 (AMBA) buses and a special Xylon Memory Bus (XMB). The logiCVC-ML requires relatively low memory bandwidth which depends on the selected display control parameters. When the video memory is implemented in high bandwidth memory, such as SDRAM, the same memory can be used as the CPU working memory.

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

1.2 Features

- Supports Xilinx® Zynq™-7000 AP SoC and all Xilinx FPGA families
- Display controller IP core for LCD and CRT displays (can be tailored for special display types)
- Available SW drivers for Linux, Android™ and Microsoft® Windows® Embedded Compact
- Supports resolutions from 64x1 up to 2048x2048
- Higher resolutions support available on request
- Supports up to 5 layers, the last one configurable as background colour
- Configurable layers' address, size and position
- Alpha blending and colour keyed transparency allows blending of video and graphics content on the screen
- Pixel, layer or colour lookup table (CLUT) alpha blending mode can be set for each layer independently
- Packed pixel layer memory organization:
  - RGB - 8bpp, 8bpp using Colour Look up Table, 16bpp Hicolour 5-6-5 and 24bpp Truecolour 8-8-8
  - YCbCr - 16bpp (4:2:2) and 24bpp (4:4:4)
- Support for multiple display interfaces:
  - Parallel display data bus (RGB or YCbCr): 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
- Configurable PLBv4.6, XMB or ARM® AMBA® AXI4 memory interface data width (32, 64 or 128bits)
- Simple programming of control registers through OPB, PLBv4.6 or AXI4-Lite interface
- Programmable layer memory address and stride
- HW cursors
- Supports synchronization to external parallel input (data used as one layer)
- Double/triple buffering enables flicker free reproduction of live video streams overlaid by graphics HMI
- Display power-on sequencing control signals
- Parametrical VHDL design that allows tuning of slice consumption and features set
- Prepared for Xilinx Vivado® Design Suite and Xilinx Platform Studio (XPS) implementation tools
- Simple Plug'n'Play with other Xylon logicBRICKS™ IP cores, such as:
  - logiMEM Flexible memory controller
  - logiBITBLT Bit Block Transfer 2D Graphic Accelerator
  - logi3D Scalable 3D Graphics Accelerator
  - logiWIN Versatile Video Input Controller

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
- Output Standard Converter
- Registers

2.1 Block Schematic

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

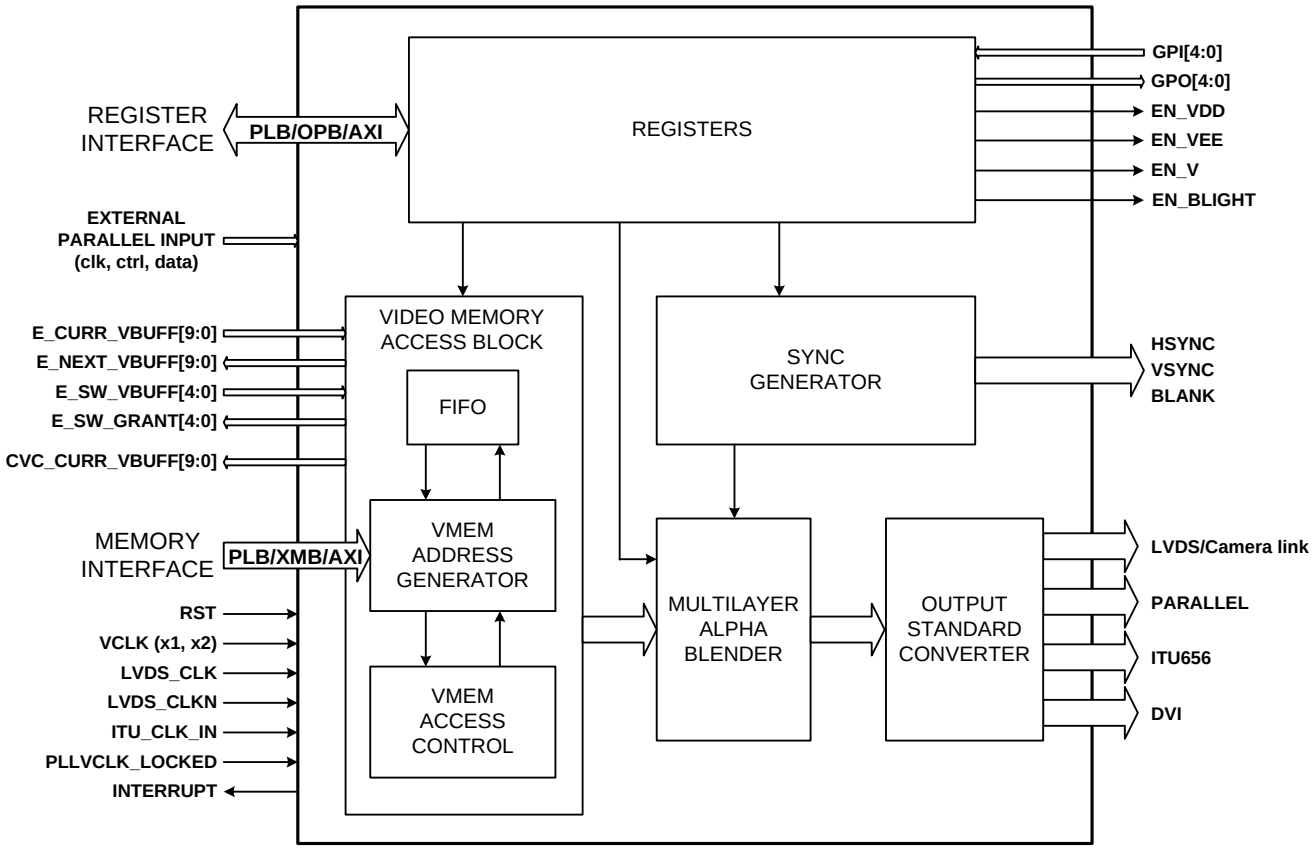


Figure 2.1: logiCVC-ML Architecture

2.1.1 Video Memory Access Block

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

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

2.1.2 Sync Generator

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

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

2.1.3 Multilayer Alpha Blender

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

2.1.4 Output Standard Converter

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

2.1.5 Registers

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

General logiCVC-ML registers include Horizontal and Vertical resolution and sync registers. These registers control horizontal and vertical timing in pixel clock increments, such as HSYNC/VSYNC

active state, delay from HSYNC/VSYNC inactive to data start, delay from end of data up to HSYNC active and delay from VSYNC inactive to first visible pixel line start.

The group of general logiCVC-ML registers also includes Display Type and Control, Background 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 generic parameter C_REGS_INTERFACE.

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

2.2 Pin-out

Table 2.1: Pin-out

Signal name	Type	Description
Global Signals		
RST	I	Global reset input, high active
VCLK	I	Video clock input
VCLK2	I	Video clock input x2 (used for 12bit parallel multiplexed output interface)
PLLVCLK_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
Memory Interface		
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
Register Interface		
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
Display Control Signals		
HSYNC	IO [1)]	Horizontal sync
VSYNC	IO [1)]	Vertical sync
PIX_CLK	IO [1)]	Pixel clock
PIX_CLKN	IO [1)]	Pixel clock inverted
BLANK	IO [1)]	Blank/display enable
D_PIX[n : 0]	IO [1)]	Video pixel data bus

Signal name	Type	Description
LVDS_DATA_OUT_P[3:0]	O	LVDS / Camera link pixel data, positive
LVDS_DATA_OUT_N[3:0]	O	LVDS / Camera link pixel data, negative
LVDS_CLK_OUT_P	O	LVDS / Camera link clock, positive
LVDS_CLK_OUT_N	O	LVDS / Camera link clock, negative
ITU656_CLK	O	ITU656 clock
ITU656_DATA[7:0]	O	ITU656 data
DVI_CLK_P	O	DVI clock, positive
DVI_CLK_N	O	DVI clock, negative
DVI_DATA_P[2:0]	O	DVI pixel data, positive
DVI_DATA_N[2:0]	O	DVI pixel data, negative
Auxiliary Signals		
E_VCLK	I	External VCLK (used when external parallel input is used)
E_VSYNC	I	External VSYNC (used when external parallel input is used)
E_HSYNC	I	External HSYNC (used when external parallel input is used)
E_BLANK	I	External BLANK (used when external parallel input is used)
E_DATA[n : 0]	I	External RGB data (used when external parallel input is used)
E_VIDEO_PRESENT	I	External video present (used when external parallel input is used)
E_CURR_VBUFF[9:0]	I	Current external stream video memory buffer (two bits per layer)
E_NEXT_VBUFF[9:0]	O	Next external stream video memory buffer to write to (two bits per source)
E_SW_VBUFF[4:0]	I	External switch logiCVC-ML video memory buffers (one bit per layer)
E_SW_GRANT[4:0]	O	External switch grant (one bit per source, handshaking signal for E_SW_VBUFF)
CVC_CURR_VBUFF[9:0]	O	Current CVC reading video buffer (two bits per layer)
GPI[4:0]	I	General purpose input
GPO[4:0]	O	General purpose output
INTERRUPT	O	logiCVC Interrupt signal, level sensitive, high active
EN_VDD	O	Enable Vdd power supply
EN_BLIGHT	O	Enable backlight power supply
EN_V	O	Enable display control/data signals
EN_VEE	O	Enable Vee power supply

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

2.3 Generic Parameters

The logiCVC-ML configuration is defined by generic parameters. These parameters are accessible in XPS and Vivado (IP Integrator). 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
Version.General Generics			
C_IP_LICENSE_TYPE [1)]	0, 1, 2, 3	0	Constant: 0 – source, 1 – evaluation, 2 – release, 3 – university evaluation
C_IP_MAJOR_REVISION [1)]	0 – 31	4	Constant: Values from 0 to 31
C_IP_MINOR_REVISION [1)]	0 – 31	1	Constant: Values from 0 to 31
C_IP_PATCH_LEVEL [1)]	0 – 25	1	Constant: Values from 0(a) to 25(z)
C_IP_LICENSE_CHECK [1)]	0, 1	1	Constant: 0 – no, 1 – yes
C_IP_TIME_BEFORE_BREAK [1)]	0, 1, 2, 3	1	Constant: 0 – infinite, 1 – 1h, 2 – 12h, 3 – 24h
Video Memory.General Generics			
C_VMEM_INTERFACE	0, 1, 2	2	Video memory interface: (0 – PLBv46, 1 – XMB, 2 – AXI4)
C_MEM_BURST [2)]	4, 5, 6	4	Number of transfers per burst (4 – 16, 5 – 32, 6 – 64)
C_MEM_LITTLE_ENDIAN	0, 1	0 for PLBv46, 1 for AXI4 or XMB	Endianness (0 – big endian, 1 – little endian)
C_MEM_BYTE_SWAP	0, 1	0	Swap bytes for memory access
C_INCREASE_FIFO	1, 2, 4, 8	1	FIFO size multiplication factor (1=Normal size, 2=x2, 4=x4, 8=x8)
Video Memory.PLB Master v4.6 Generics			
C_MPLB_AWIDTH [3, 8)]	32	32	PLB address bus width
C_MPLB_DWIDTH [3, 8)]	32, 64, 128	64	PLB data bus width
C_MPLB_NUM_MASTERS [3, 8)]	1 – 16	8	Maximum number of PLB masters in the design
C_MPLB_SMALLEST_SLAVE [3, 8)]	32	32	Data bus width of the smallest slave connected to the PLB
C_MPLB_NATIVE_DWIDTH [3, 31)]	32, 64, 128	32	Sets minimum PLB data bus width that logiCVC PLB master interface supports; non-HDL generic parameter

Parameter Name	Allowable values	Default value	Description
C_MPLB_PRIORITY [3)]	0, 1, 2, 3	3	Priority on PLB bus: (0 – the lowest, 3 – the highest)
C_MPLB_P2P [3, 8)]	0, 1	0	PLB point to point, non-HDL generic parameter
C_MPLB_SUPPORT_BURST [3)]	1	1	PLB master supports burst access, non-HDL constant
Video Memory.XMB Generics			
C_XMB_DATA_BUS_WIDTH	32, 64, 128	64	XMB data bus width
Video Memory.AXI4 Master Generics			
C_M_AXI_DATA_WIDTH [22)]	32, 64, 128	64	AXI4 data bus width
C_M_AXI_ADDR_WIDTH [22)]	32	32	AXI4 address bus width
C_M_AXI_SUPPORTS_READ [22)]	1	1	Master supports read transactions, non-HDL constant
C_M_AXI_SUPPORTS_WRITE [22)]	0	0	Master supports write transactions, non-HDL constant
C_M_AXI_SUPPORTS_THREADS [22)]	0	0	Support of threads, non-HDL constant
C_M_AXI_THREAD_ID_WIDTH [22)]	1	1	Width of ID signals for threads
C_M_AXI_SUPPORTS_NARROW_BURST [22)]	0	0	Master issues multi-beat transactions in which the size of the data transfers is narrower than the interface data-width, non-HDL constant
C_M_AXI_PROTOCOL [22)]	AXI4	AXI4	Bus protocol supported, non-HDL constant
C_INTERCONNECT_M_AXI_ARB_PRIORITY [22)]	0 – 15	0	Xilinx AXI interconnect parameter determining priority: higher values indicate higher priority, non-HDL generic parameter. Round-robin arbitration is used among all masters with priority value 0
C_INTERCONNECT_M_AXI_READ_ISSUING [22)]	8	8	Maximum number of data-active read transactions generated
Registers.General Generics			
C_REGS_INTERFACE	0, 1, 2	2	Register interface: (0 – OPB slave, 1 – PLBv46 slave, 2 – AXI4-Lite)
C_READABLE_REGS [4)]	0, 1	1	Readable registers: (0 – disabled, 1 – enabled)
C_REG_BYTE_SWAP	0, 1	0	Swap bytes for register access
Registers.Addresses Generics			

Parameter Name	Allowable values	Default value	Description
C_REGS_BASEADDR	Valid word aligned address	0xFFFFFFFF	Registers base address
C_REGS_HIGHADDR	Valid word aligned address	0x00000000	Registers high address
Registers.OPB Slave Generics			
C_OPB_AWIDTH [5, 6)]	32	32	OPB address bus width
C_OPB_DWIDTH [5, 6)]	32, 64	32	OPB data bus width
Registers.PLB Slave v4.6 Generics			
C_SPLB_AWIDTH [7, 8)]	32	32	PLB address bus width
C_SPLB_DWIDTH [7, 8)]	32, 64	32	PLB data bus width
C_SPLB_MID_WIDTH [7, 8)]	1, 2, 3, 4	1	PLB master ID bus width, the value is: log2(C_SPLB_NUM_MASTERS)
C_SPLB_NUM_MASTERS [7, 8)]	1 – 16	2	Number of masters that can be connected
C_SPLB_NATIVE_DWIDTH [7, 8)]	32	32	Value always set to 32 to allow PLB bus to adjust data width according to the other devices on bus
Registers.AXI4-Lite Slave Generics			
C_S_AXI_ADDR_WIDTH [23)]	32	32	AXI4-Lite address bus width
C_S_AXI_DATA_WIDTH [23)]	32	32	AXI4-Lite data bus width
C_S_AXI_PROTOCOL [23)]	AXI4LITE	AXI4LITE	Bus protocol supported, non-HDL constant
User.General Generics			
C_NUM_OF_LAYERS	1, 2, 3, 4, 5	1	Number of logiCVC layers
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_DISPLAY_COLOR_SPACE	0, 1, 2	0	Display interface colour space (0=RGB, 1=YCbCr 4:2:2, 2=YCbCr 4:4:4)
C_PIXEL_DATA_WIDTH [26)]	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_ROW_STRIDE	512, 1024, 2048	1024	Distance in number of pixels between same colon pixels for adjacent rows

Parameter Name	Allowable values	Default value	Description
C_USE_SIZE_POSITION	0, 1	1	Enable functionality of configurable layer size, position and offset (0 – disabled, 1 – enabled)
C_USE_BACKGROUND [12)]	0, 1	0	Configure last layer as background
C_XCOLOR [25)]	0, 1	0	Enable dithering module
C_USE_VCLK2 [32, 33)]	0, 1	1	vclk2 is used for 12*2 parallel pixel data width
C_VCLK_PERIOD [24)]	8333:52631	25000	vclk clock period defined in picoseconds. XPS assigned
C_DVI_CLK_MODE [24)]	0, 1	0	DVI transmitter clocking mode for 7 Series implementation (0 – MMCM+BUFIO+2xBUFR, 1 – PLL+3xBUFG)
C_USE_XTREME_DSP	0, 1, 2	2	Use DSP resources for blender: (0=no, 1=yes, 2=auto)
C_USE_MULTIPLIER	0, 1, 2	2	Type of multiplier used in blender: (0=LUT, 1=DSP block, 2=auto)
C_USE_E_PARALLEL_INPUT [12)]	0, 1	0	Synchronize logiCVC to external parallel input and use data as one layer (0 – no, 1 – yes)
C_USE_E_VCLK_BUFGMUX [10)]	0, 1	1	Use BUFGMUX for clock switching for external parallel input (0 – no, 1 – yes)
C_E_LAYER [10)]	0, 1, 2, 3, 4	0	Use external parallel input as which layer
C_E_DATA_WIDTH [10)]	8, 16, 24	24	External parallel input data width
C_LVDS_DATA_WIDTH	3, 4	4	LVDS and camera link display interface data width. XPS assigned, do not modify
User.Layer 0 Generics			
C_LAYER_0_DATA_WIDTH [34)]	8, 16, 24	16	Layer data width: 8, 16 or 24 bits per pixel
C_LAYER_0_TYPE	0, 1	0	Layer type (0 – RGB, 1 – YCbCr)
C_LAYER_0_ALPHA_MODE [11)]	0, 1, 2, 3	0	Alpha blending mode: (0=layer, 1=pixel, 2=16bit CLUT, 3=24bit CLUT)
C_LAYER_0_ADDR	Valid pixel aligned address	0x00000000	Layer 0 default memory address [20)]
C_BUFFER_0_OFFSET	integer	1024	Layer 0 double buffer address offset represented in number of lines [21)]

Parameter Name	Allowable values	Default value	Description
User.Layer 1 Generics			
C_LAYER_1_DATA_WIDTH [12, 34)]	8, 16, 24	16	Layer data width: 8, 16 or 24 bits per pixel
C_LAYER_1_TYPE [12)]	0, 1, 2	0	Layer type (0 – RGB, 1 – YCbCr, 2 – Alpha)
C_LAYER_1_ALPHA_MODE [12, 13, 27)]	0, 1, 2, 3	0	Alpha blending mode: (0=layer, 1=pixel, 2=16bit CLUT, 3=24bit CLUT)
C_LAYER_1_ADDR [12, 27)]	Valid pixel aligned address	0x00000000	Layer 1 default memory address [20)]
C_BUFFER_1_OFFSET [12, 27)]	integer	1024	Layer 1 double buffer address offset represented in number of lines [21)]
User.Layer 2 Generics			
C_LAYER_2_DATA_WIDTH [14, 34)]	8, 16, 24	16	Layer data width: 8, 16 or 24 bits per pixel
C_LAYER_2_TYPE [14)]	0, 1	0	Layer type (0 – RGB, 1 – YCbCr)
C_LAYER_2_ALPHA_MODE [14, 15, 28)]	0, 1, 2, 3	0	Alpha blending mode (0=layer, 1=pixel, 2=16bit CLUT, 3=24bit CLUT)
C_LAYER_2_ADDR [14, 28)]	Valid pixel aligned address	0x00000000	Layer 2 default memory address [20)]
C_BUFFER_2_OFFSET [14, 28)]	integer	1024	Layer 2 double buffer address offset represented in number of lines [21)]
User.Layer 3 Generics			
C_LAYER_3_DATA_WIDTH [16, 34)]	8, 16, 24	16	Layer data width: 8, 16 or 24 bits per pixel
C_LAYER_3_TYPE [16)]	0, 1, 2	0	Layer type (0 – RGB, 1 – YCbCr, 2 – Alpha)
C_LAYER_3_ALPHA_MODE [16, 17, 29)]	0, 1, 2, 3	0	Alpha blending mode (0=layer, 1=pixel, 2=16bit CLUT, 3=24bit CLUT)
C_LAYER_3_ADDR [16, 29)]	Valid pixel aligned address	0x00000000	Layer 3 default memory address [20)]
C_BUFFER_3_OFFSET [16, 29)]	integer	1024	Layer 3 double buffer address offset represented in number of lines [21)]
User.Layer 4 Generics			

Parameter Name	Allowable values	Default value	Description
C_LAYER_4_DATA_WIDTH [18, 34)]	8, 16, 24	16	Layer data width: 8, 16 or 24 bits per pixel
C_LAYER_4_TYPE [18)]	0, 1	0	Layer type (0 – RGB, 1 – YCbCr)
C_LAYER_4_ALPHA_MODE [18, 35)]	0, 1, 2, 3	0	Alpha blending mode (0=layer, 1=pixel, 2=16bit CLUT, 3=24bit CLUT)
C_LAYER_4_ADDR [18, 30)]	Valid pixel aligned address	0x00000000	Layer 4 default memory address [20)]
C_BUFFER_4_OFFSET [18, 30)]	integer	1024	Layer 4 double buffer address offset represented in number of lines [21)]

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, double/triple CLUT register, power control and IP version register 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_PARALLEL_INPUT = 1
11. Valid if C_USE_E_ PARALLEL _INPUT = 0 or if (C_USE_E_ PARALLEL _INPUT = 1 and C_E_LAYER != 0)
12. Valid if (C_NUM_OF_LAYERS > 1)
13. Valid if C_USE_E_ PARALLEL _INPUT = 0 or if (C_USE_E_ PARALLEL _INPUT = 1 and C_E_LAYER != 1)
14. Valid if (C_NUM_OF_LAYERS > 2)
15. Valid if C_USE_E_ PARALLEL _INPUT = 0 or if (C_USE_E_ PARALLEL _INPUT = 1 and C_E_LAYER != 2)
16. Valid if (C_NUM_OF_LAYERS > 3)
17. Valid if C_USE_E_ PARALLEL _INPUT = 0 or if (C_USE_E_ PARALLEL _INPUT = 1 and C_E_LAYER != 3)
18. Valid if (C_NUM_OF_LAYERS > 4)
19. Valid if C_USE_E_ PARALLEL _INPUT = 0 or if (C_USE_E_ PARALLEL _INPUT = 1 and C_E_LAYER != 4)
20. Default layer memory address is valid after power up or after reset. User can modify layer memory address at runtime using layer address register described in chapter 10.2.15.

    If logiCVC-ML is configured not to use layer size and positioning (C_USE_SIZE_POSITION = 0), lower bits of layer memory address will be ignored. For more information, please refer to chapter 10.2.15.

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. This parameter is only used with external freme buffer switching.

    For more information on setting up this parameter, please refer to chapter 4.3 Memory address.
22. Valid if C_VMEM_INTERFACE = 2
23. Valid if C_REGS_INTERFACE = 2
24. Required only if DVI display interface is used
25. Valid if C_PIXEL_DATA_WIDTH = 18
26. If C_DISPLAY_COLOR_SPACE = 2, C_PIXEL_DATA_WIDTH must be set to 24, and if C_DISPLAY_COLOR_SPACE = 1, C_PIXEL_DATA_WIDTH must be set to 16.
27. If this layer is the last layer in the configuration and it is set up to be a background layer (C_NUM_OF_LAYERS = 2 and C_USE_BACKGROUND = 1) these parameter is irrelevant and is not used.
28. If this layer is the last layer in the configuration and it is set up to be a background layer (C_NUM_OF_LAYERS = 3 and C_USE_BACKGROUND = 1) these parameter is irrelevant and is not used.
29. If this layer is the last layer in the configuration and it is set up to be a background layer (C_NUM_OF_LAYERS = 4 and C_USE_BACKGROUND = 1) these parameter is irrelevant and is not used.
30. If this layer is the last layer in the configuration and it is set up to be a background layer (C_NUM_OF_LAYERS = 5 and C_USE_BACKGROUND = 1) these parameter is irrelevant and is not used.
31. Setting this generic parameter defines the minimum PLB data bus width that logiCVC-ML PLB master interface supports. If it is the widest interface between all other devices connected to the PLB, then all the other devices will extend their PLB wrapper's data bus width to logiCVC-ML's MPLB_NATIVE_DWIDTH. If there are devices connected with wider data bus width, logiCVC-ML will expand its PLB wrapper's data bus width to match the widest device.
32. Valid if C_PIXEL_DATA_WIDTH = 12.
33. Pix_clk rising edge will be in the middle of the DDR data eye if vclk2 is used, otherwise it will be synchronous to pixel data bus.
34. For RGB layer, 24 represents an RGB 888 format, 16 represents an RGB 565 format while 8 can represent RGB 332 or CLUT format depending on layer alpha mode selection. For YCbCr layer, 24 represents an YCbCr 888 (4:4:4) format and 16 represents an YCb YCr 88 88 (4:2:2) format.
35. Used only to select if data for the last layer is sourced from video memory or from CLUT.

2.4 Port and Parameter Dependencies

Table 2.3: Port and Parameter Dependencies

Parameter Name	Affects Port(s)	Description
C_PIXEL_DATA_WIDTH	vclk2	When C_PIXEL_DATA_WIDTH = 12
C_USE_VCLK2	vclk2	When C_USE_VCLK2 = 1
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
C_DISPLAY_COLOR_SPACE	d_pix	When C_DISPLAY_INTERFACE = 0
C_LVDS_DATA_WIDTH	lvds_data_out_p lvds_data_out_n	Determines LVDS data bus width
C_USE_E_PARALLEL_INPUT	e_vclk e_vsync e_hsync e_blank e_data	When C_USE_E_PARALLEL_INPUT = 1
C_E_DATA_WIDTH	e_data	When C_USE_E_PARALLEL_INPUT = 1 C_E_DATA_WIDTH generic parameter 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 [1)]	PLBv46 Master Interface XMB Interface AXI4 Interface	Only one interface (MPLB, XMB or M_AXI) is available depending on C_VMEM_INTERFACE generic parameter setting

Parameter Name	Affects Port(s)	Description
C_REGS_INTERFACE 2)	OPB Interface PLBV46 Slave Interface AXI4-Lite Slave interface	Only one interface (OPB, SPLB or S_AXI) is available depending on C_REGS_INTERFACE generic parameter setting

1. mplb_clk is used when PLBV46 Master interface is implemented, mclk port is used when XMB interface is implemented, while M_AXI_ACLK is used when AXI4 interface is implemented.
2. opb_clk is used when OPB interface is implemented, splb_clk is used when PLBV46 Slave interface is implemented, while S_AXI_ACLK is used when AXI4-Lite interface is used.

3 CLOCK AND RESET SIGNALS

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

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

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

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

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

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

In case ITU656 output standard is used (C_DISPLAY_INTERFACE = 1), user must supply ITU656 clock signal, ITU_CLK_IN, which has a frequency of 27 MHz as requested by the ITU656 standard. Additionally, VCLK and ITU_CLK_IN must be fully synchronous and VCLK must have a frequency of 13.5 MHz.

In case 12*2 DDR parallel data interface is used (C_PIXEL_DATA_WIDTH = 12) and C_USE_VCLK2 is set to 1, in addition to VCLK also VCLK2 signal must be supplied to logiCVC-ML input. VCLK2 has to be double the VCLK frequency and synchronous to VCLK.

In case LVDS or camera link output standard is used (C_DISPLAY_INTERFACE = 2, 3 or 4), user must supply one or two additional clock inputs, LVDS_CLK and LVDS_CLKN. In case of Spartan-3. Virtex-4, Virtex-5 or Virtex-6 device implementation, LVDS_CLK and LVDS_CLKN clocks must be connected and must have 3.5 times higher frequency and be synchronous to VCLK.

In Spartan-6 and 7 Series FPGA families including the Zynq™-7000 AP SoC, specific HW IO serializers are used for LVDS and camera link output streams. In this case, beside VCLK, user must supply only one additional clock signal, LVDS_CLK, which must have 7 times higher frequency and be synchronous to VCLK.

For Spartan-6 devices, the same PLL module must source these two clock signals. Additionally, LOCKED output of the PLL must be connected to PLLVCLK_LOCKED input of the logiCVC-ML. Figure 3.2 shows the required clock connection for the LVDS or camera link usage in Spartan-6 devices.

For 7 Series devices 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 devices, PLLVCLK_LOCKED input is not used.

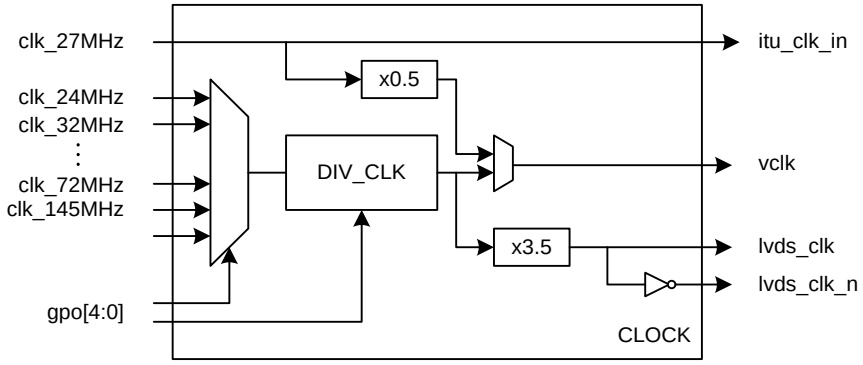


Figure 3.1: Example clock generator block schematic

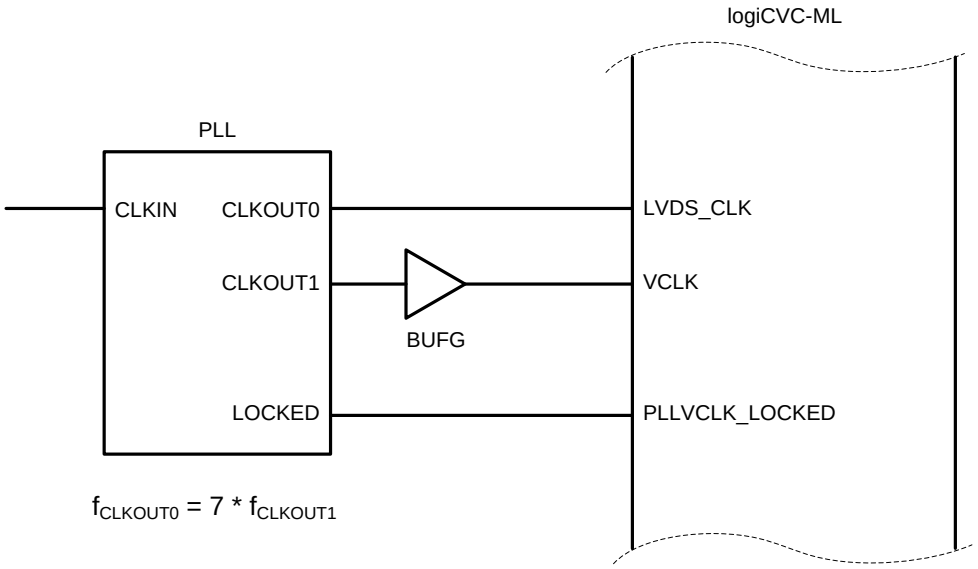


Figure 3.2: Example of clock connection in Spartan-6 family when LVDS or camera link output standard is used

3.1 Pixel Clock Output

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

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

- An FPGA external PLL module that can be programmed in a wide range of video frequencies. Usually, these ICs are controlled via an I^2C or SPI interface.
- Using the logiCLK IP from the Xylon's logicBRICKS IP library, which uses a Xilinx FPGA PLL component programmable through the AXI4-Lite interface.
- A custom FPGA clock module with predefined clock frequencies similar to the one outlined in Figure 3.1. This module can then be controlled with the GPO signals, which are outputs from the logiCVC-ML core.

4 MEMORY INTERFACE

The video image is stored in video memory, which is accessible trough Master PLBv4.6, XMB or AXI4 interface. By default, AXI4 bus is used, but by setting C_VMEM_INTERFACE to '1' logiCVC-ML will access video memory through XMB, and setting C_VMEM_INTERFACE to '0' logiCVC-ML will access video memory through MPLB. Depending on the memory interface, different clock signals are used. Clock input signal MPLB_CLK is used for Master PLBv4.6 interface, MCLK is used for XMB interface, while M_AXI_ACLK is used for AXI4 interface.

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

logiCVC-ML PLB, XMB or AXI4 master interface can be configured as 32-bit, 64-bit or 128-bit wide by setting the appropriate 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 or 128-bit PLB bus, fixed burst lengths of up to 32 or 64 can be used by setting generic parameter C_MEM_BURST. For more information on fixed burst transfers, please refer to PLBv4.6 architecture specifications. When using XMB or AXI4 interface, maximum memory burst can be set with the same generic parameter independently of data width. The out of block accesses are obtained by using single PLB/XMB/AXI4 cycles and bursts lower than defined by C_MEM_BURST. PLB abort cycle contained in PLB bus standard is not used.

For more information on PLBv4.6, XMB or AXI4 bus architecture, please refer to Xilinx / IBM Processor Local Bus, Xylon logiMEM or AXI4 (AMBA) documentation, respectively.

4.1 Memory Layout

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

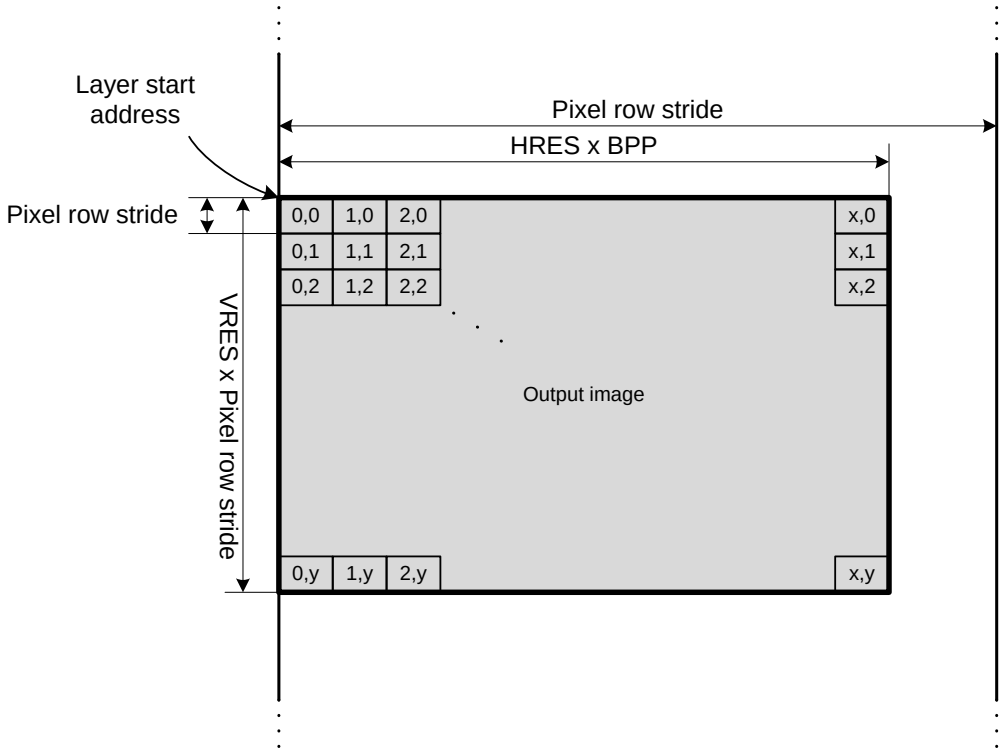


Figure 4.1: Rectangular memory layout

Pixel row stride is defined as the distance in bytes between same colon pixels for adjacent rows. Depending on layer data width (bpp), alpha blending mode and C_ROW_STRIDE generic parameter, 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)	-	-

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

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

Table 4.2, Table 4.3 and Table 4.4 describe memory layout configurations according to layer width, layer type and layer alpha blending mode. All three tables assume that C_MEM_LITTLE_ENDIAN = 1, C_MEM_BYTE SWAP = 0.

Table 4.2: 8bpp layer memory layout

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

Table 4.3: 16bpp layer memory layout

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

Table 4.4: 24bpp layer memory layout

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

Byte swapping configuration affects how logiCVC-ML reads video memory. All bytes in memory data bus are swapped if generic C_MEM_BYTE_SWAP is set. Example on memory layout depending on C_MEM_BYTE_SWAP when memory data bus is 64 and 32-bit is given in Table 4.5:

Table 4.5: Memory layout depending on C_MEM_BYTE_SWAP

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

1. Data bus width is C_MPLB_DWIDTH when C_VMEM_INTERFACE = 0 (video memory interface is PLBv46), C_XMB_DATA_BUS_WIDTH when C_VMEM_INTERFACE = 1 (XMB interface) or C_M_AXI_DATA_WIDTH when C_VMEM_INTERFACE = 2 (AXI4 interface)

Generic C_MEM_LITTLE_ENDIAN determines if little endian or big endian is used for data representation in memory. If set, it reorders amount of data depending on C_LAYER_X_DATA_WIDTH. If C_LAYER_X_DATA_WIDTH is set to 24 then 4 bytes of data are swapped. In case C_LAYER_X_DATA_WIDTH is set to 16 then data of 2 bytes are swapped and if C_LAYER_X_DATA_WIDTH is set to 8 then 1-byte data are swapped. Example on memory layout depending on C_MEM_LITTLE_ENDIAN when memory data bus is 32 and 64-bit is outlined in Table 4.6 and Table 4.7.

Table 4.6: Memory layout depending on endianess (32-bit data bus)

C_LAYER_X_DATA_WIDTH		**24bpp**		**16bpp**		**8bpp**	
C_MEM_LITTLE_ENDIAN		**1**	**0**	**1**	**0**	**1**	**0**
Address	**0**	Pix0	Pix0	Pix 0	Pix 1	Pix 0	Pix 3
	1					Pix 1	Pix 2
	2			Pix 1	Pix 0	Pix 2	Pix 1
	3					Pix 3	Pix 0

Table 4.7: Memory layout depending on endianess (64-bit data bus)

C_LAYER_X_DATA_WIDTH		24bpp		16bpp		8bpp	
C_MEM_LITTLE_ENDIAN		1	0	1	0	1	0
Address	0	Pix0	Pix1	Pix 0	Pix 3	Pix 0	Pix 7
	1					Pix 1	Pix 6
	2			Pix 1	Pix 2	Pix 2	Pix 5
	3					Pix 3	Pix 4
	4	Pix1	Pix0	Pix 2	Pix 1	Pix 4	Pix 3
	5					Pix 5	Pix 2
	6			Pix 3	Pix 0	Pix 6	Pix 1
	7					Pix 7	Pix 0

4.2 Pixel Layout

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

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

4.2.1 8bpp

Table 4.8: 8bpp pixel layout

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

In 8bpp with pixel alpha mode, only 3 bits are used because alpha factor is multiplied with every 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 4.8 outlines default pixel format (RGB). For more information on changing the order of colours inside the pixel, please refer to chapter 10.2.19 Layer Control Register - Lx_CTRL.

logiCVC-ML supports Layers 1 and 3 to be set as Alpha Layers, which means that they consist only of alpha blending factors and not colour values (RGB or YCbCr). In case Layer 1 is configured as Alpha Layer (C_LAYER_1_TYPE = 2), data read from memory is used as pixel alpha factor for calculation between Layers 0 and 2. The same applies for Layer 3 with the exception that it is used for calculation between Layers 2 and 4.

4.2.2 16bpp and 24bpp

Table 4.9: 16bpp and 24bpp RGB pixel layout

Address	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

Table 4.10: 16bpp and 24bpp YCbCr pixel layout

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

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

4.3 Memory address and range

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

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

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

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

The following parameters define each layer memory address and range:

- Layer address register; default value is defined with C_LAYER_x_ADDR generic parameter
- Pixel row stride; defined with C_ROW_STRIDE, C_LAYER_x_DATA_WIDTH and C_LAYER_x_ALPHA_MODE generic parameters. Please refer to Table 4.1.
- Double/triple buffer offset; defined with C_BUFFER_x_OFFSET generic parameter represented in number of lines. This is only relevant if external frame buffer switching is used.
- Vertical resolution; defined with resolution register. This is only relevant if CPU frame buffer switching is used.

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

Pixel row stride is defined in Table 4.1 and explained in more detail in chapter 4.1 Memory Layout.

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

Applying the above gives us the following equation:

$$Layer_x Buffer_0 addr = Layer_x addr$$

$$Layer_x Buffer_1 addr = Layer_x addr + (C\_BUFFER\_x\_OFFSET \times PixelRowStride_x)$$

$$Layer_x Buffer_2 addr = Layer_x addr + (2 \times C\_BUFFER\_x\_OFFSET \times PixelRowStride_x)$$

For more information on frame buffer switching, please refer to chapter 4.4 Frame Buffer Synchronisation.

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

C_NUM_OF_LAYERS = 3,
C_ROW_STRIDE = 1024,
C_LAYER_0_ADDR = 0x10000000,
C_LAYER_0_DATA_WIDTH = 8,
C_LAYER_0_ALPHA_MODE = 3 (CLUT),
C_BUFFER_0_OFFSET = 1024,
C_LAYER_1_ADDR = 0x10300000,

C_LAYER_1_DATA_WIDTH = 16,
C_LAYER_1_ALPHA_MODE = 0 (layer),
C_BUFFER_1_OFFSET = 1024
C_LAYER_2_ADDR = 0x10900000,
C_LAYER_2_DATA_WIDTH = 24,
C_LAYER_2_ALPHA_MODE = 1 (pixel),
C_BUFFER_2_OFFSET = 1024.

Applying the calculations for buffer offsets provides us with Table 4.11, which shows a complete memory layout of logiCVC-ML example configuration mentioned above.

Table 4.11: Example layer and buffer addresses

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

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

In case that external frame buffer switching is not required and we assume the same parameters as explained above, the correct memory layout would be according to Table 4.12.

Table 4.12: Example layer addresses

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

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

4.4 Frame Buffer Synchronisation

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

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

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

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

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

The following chapters describe each synchronisation method in more detail.

4.4.1 CPU synchronisation

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

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

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

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

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

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

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

4.4.2 External (video input) synchronization

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

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

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

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

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

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

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

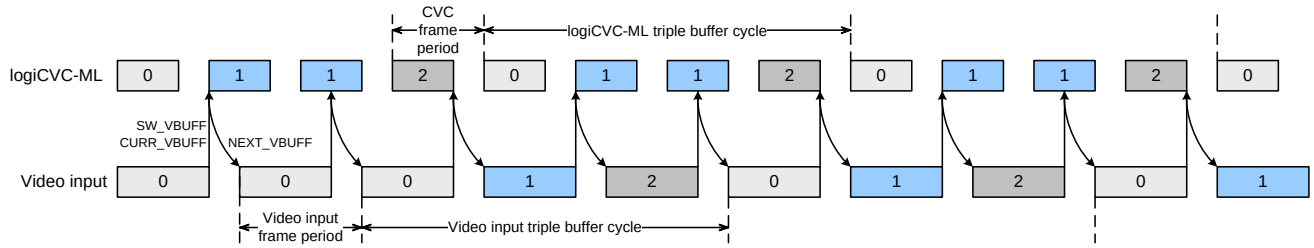


Figure 4.2: Triple buffering example when logiCVC-ML refresh rate is higher than video input

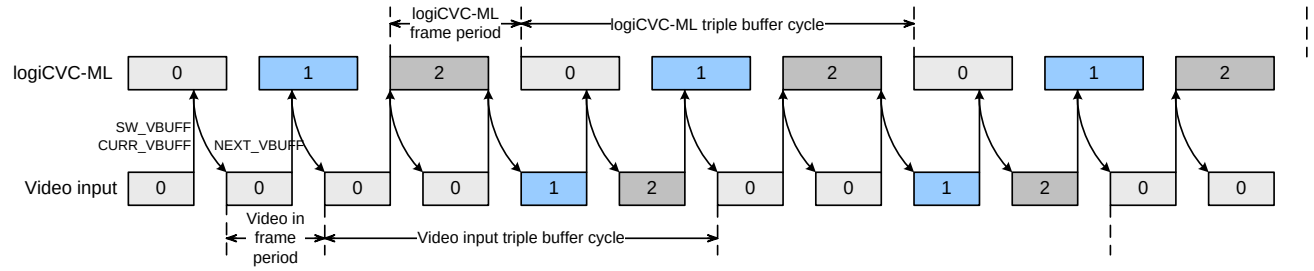


Figure 4.3: Triple buffering example when logiCVC-ML refresh rate is lower than video input

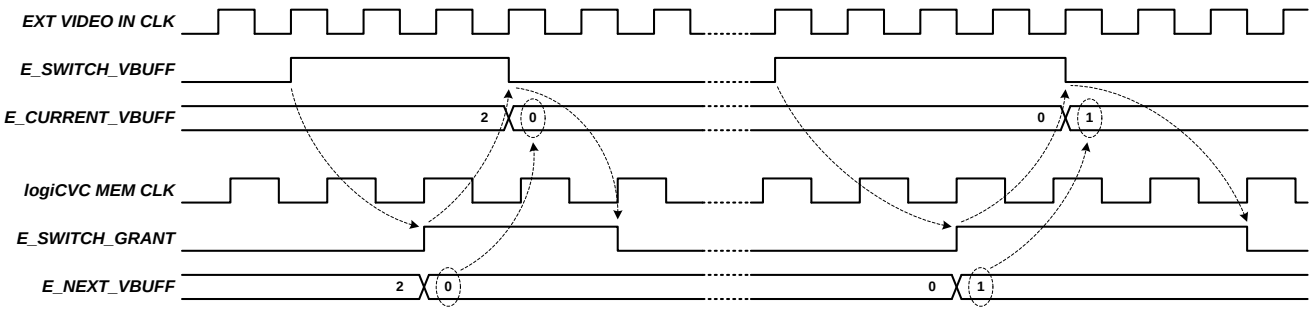


Figure 4.4: External buffer control signals timing diagram

4.4.3 Vertical sync synchronization

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

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

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

4.5 Memory bandwidth requirements

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

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

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

Equation 4.1: Required maximum memory bandwidth

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

where:

RMBW$_{max}$: required memory bandwidth, maximum

layer$_n$bpp: 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$_n$bpp values used for memory bandwidth calculation

		Data width (bit)		
		8	16	24
Layer alpha mode	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

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

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 10.2.17

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

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

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

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

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

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

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

Example 1

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

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

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

Example 2

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

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$_0$ = 400x240,
layer size$_1$ = 512x240,
layer size$_2$= resolution,
layer size$_3$ = resolution.

According to the above equations the required maximum memory bandwidth is

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

while the required average memory bandwidth is

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

As can be seen in this example, layer 4 is not influencing memory bandwidth requirements because logiCVC-ML is configured to use last layer as background layer (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 10.2.7 Background Colour Register - BACKGROUND for more information.

5 CPU INTERFACE

5.1 Register Interface

The logiCVC-ML registers can be accessed through Slave OPB, Slave PLBv4.6 or Slave AXI4-Lite interface depending on generic parameter C_REGS_INTERFACE. Please consult Xilinx OPB, PLBv4.6 or AXI4-Lite 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 generic parameter C_READABLE_REGS to 0. Variable register widths are used; 8, 16, 24 or 32-bit, however, logiCVC-ML only supports 32-bit write accesses into its registers. Non-used bits inside each register are reserved and read '0'.

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

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

5.2 Interrupt Interface

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

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

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

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

One bit in the INT_STAT register is used for 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.

When an external RGB input stream is used to drive one logiCVC-ML layer, one additional bit in the INT_STAT register is used. In that case (C_USE_E_RGB_INPUT = 1), one bit is used to signal to the CPU that external RGB source is present on the input.

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

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

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

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

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 the preceding and the first pixel of a succeeding video picture line. Blank time is also required between the last pixel of a preceding frame and the first pixel of succeeding frame. The blank time is controlled by the BLANK signal, which is active during the valid pixels on 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 is outputted on the D_PIX[C_PIXEL_DATA_WIDTH-1 : 0] data bus.

Parallel interface can be used even when other display interfaces are selected.

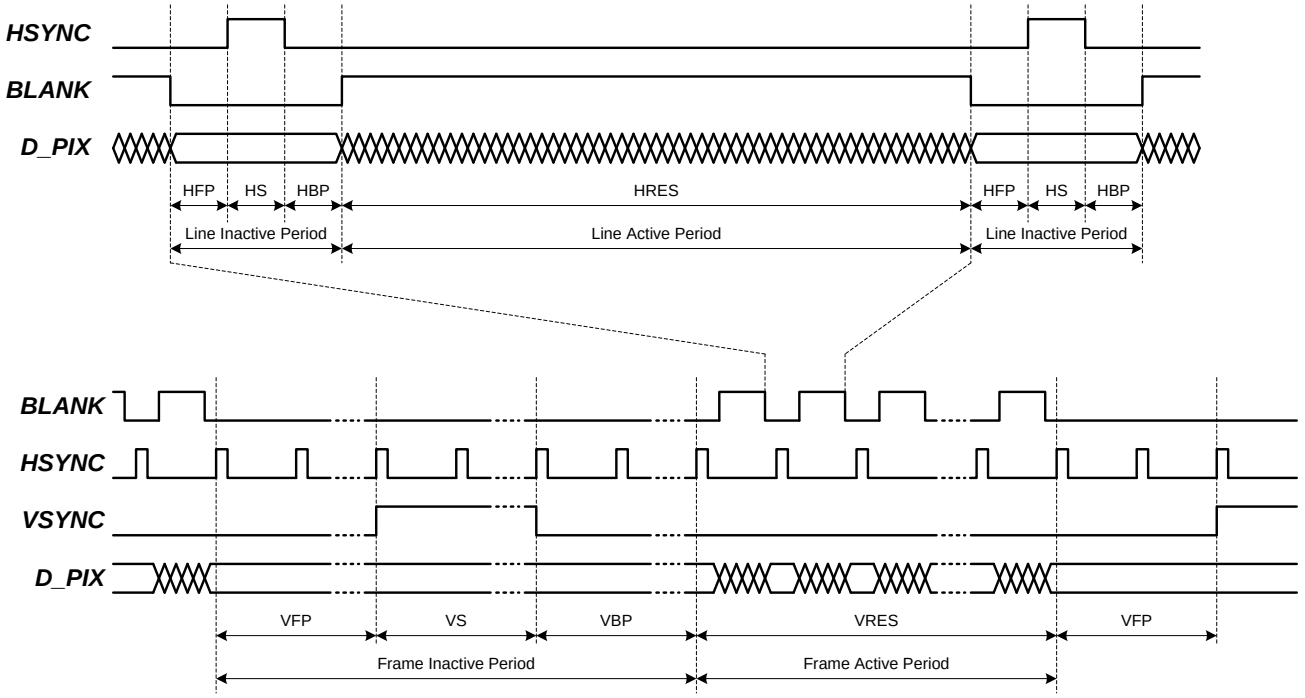


Figure 6.1: Parallel pixel data interface

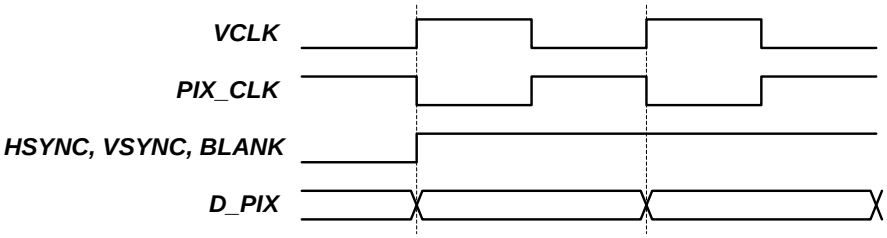


Figure 6.2: Parallel pixel data interface clock alignment

NOTE:

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

6.1.1 RGB interface

In RGB parallel output interface, through 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 = 12 (12x2 DDR data). 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. Have in mind that to use this 12-bit mode if C_USE_VCLK2 is set, user has to supply VCLK2 input clock that has twice the frequency and is synchronous to the VCLK input. In this case,

PIX_CLK rising and falling edges will be in the middle of the D_PIX data bus eye. In case C_USE_VCLK2 = 0, VCLK2 is not used and PIX_CLK is synchronous to D_PIX data bus.

By setting C_DISPLAY_INTERFACE to 0, only parallel interface is active.

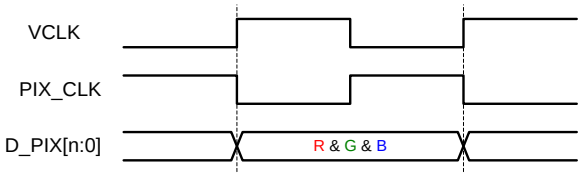


Figure 6.3: RGB data interface

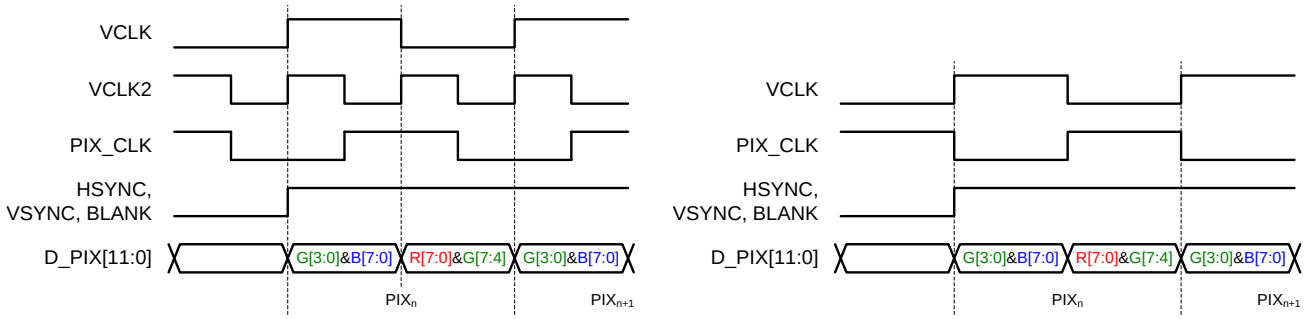


Figure 6.4: RGB 12x2-bit multiplexed data interface with and without VCLK2

6.1.2 YCbCr interface

YCbCr is a colour space that contains three components, Y as a luma component and Cb and Cr as blue-difference and red-difference chroma components. If logiCVC-ML is configured to use YCbCr parallel output interface and some layers are not in YCbCr colour format (C_LAYER_X_TYPE = 1), then their RGB pixel data is converted from RGB to YCbCr colour space using the formulas described in chapter 9.2.5 Colour space converter.

logiCVC-ML supports two sampling formats, 4:4:4 and 4:2:2. When using 4:4:4, each component, Y, Cb and Cr, will appear on D_PIX 24-bit output on every active clock edge. When using 4:2:2 sampling format, only 16-bit D_PIX output data bus is used. Of these 16 bits, 8 bits are used for Y component and the other 8 bits are alternating between Cb and Cr components with every active clock edge. (Figure 6.5).

To use the YCbCr output interface, parameter C_DISPLAY_INTERFACE has to be set to 0 (parallel interface) and C_DISPLAY_COLOR_SPACE has to be set to either 1 or 2 (YCbCr 4:2:2 or YCbCr 4:4:4) depending on the used sampling format. In both cases, the following signals are used: HSYNC (horizontal synchronous signal) VSYNC (vertical synchronous signal), BLANK (data enable signal) and D_PIX (pixel data output signal).

When using YCbCr interface, generic parameter C_PIXEL_DATA_WIDTH cannot be adjusted to all usually supported values. If output interface is chosen to be YCbCr 4:4:4 (C_DISPLAY_COLOR_SPACE=2) C_PIXEL_DATA_WIDTH must be 24 and if YCbCr 4:2:2 (C_DISPLAY_COLOR_SPACE=1) interface is chosen then C_PIXEL_DATA_WIDTH must be 16. Figure 6.5 outlines the order of pixel data on D_PIX signal depending on the chosen sampling mode.

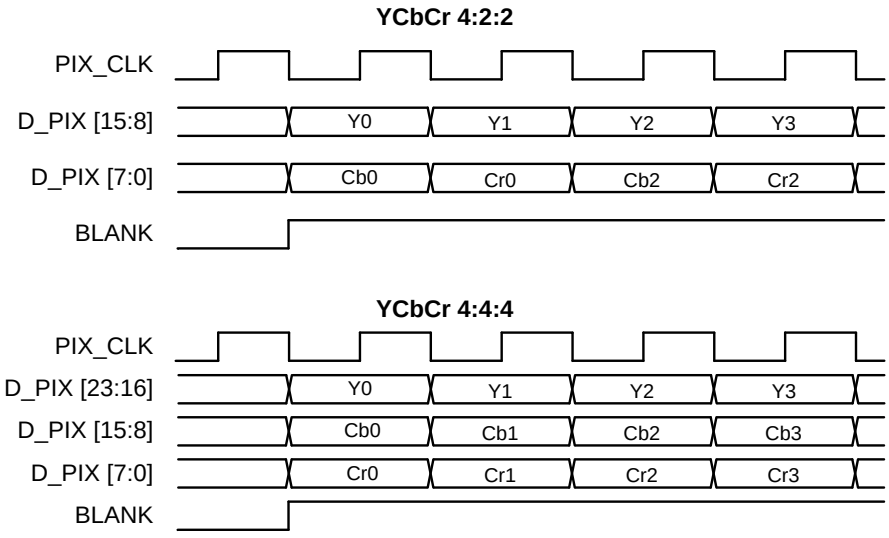


Figure 6.5: Pixel data order on YCbCr 4:4:4 and YCbCr 4:2:2 data output bus

NOTE:

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

6.2 LVDS Interface

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

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

Regardless of the LVDS interface selection, the logiCVC-ML still drives the parallel pixel interface, PIX_CLK, HSYNC, VSYNC, BLANK and D_PIX[C_PIXEL_DATA_WIDTH-1 : 0] if it is enabled in the Power Control register.

6.2.1 Standard LVDS Interface

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

LVDS interface uses four or five LVDS pairs, three or four for the data and one for the clock. To drive 24-bit video data and three control signals, 27 bits in total, over 4 LVDS pairs, LVDS coder uses faster LVDS_CLK clocks. In this way, seven signals are transmitted over one LVDS pair. Figure 6.6 shows the order of data and control bits on four LVDS data pairs.

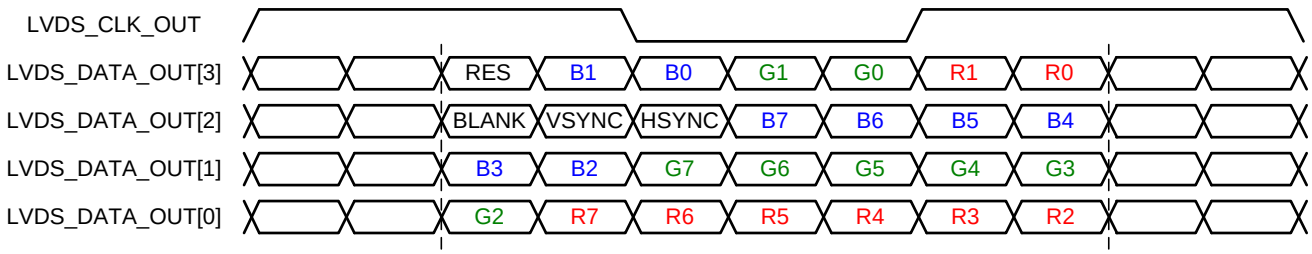


Figure 6.6: Pixel data order on LVDS data bus

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.

6.2.2 Camera Link Interface

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

Figure 6.7 shows the order of data and control bits on four Camera link LVDS data pairs.

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

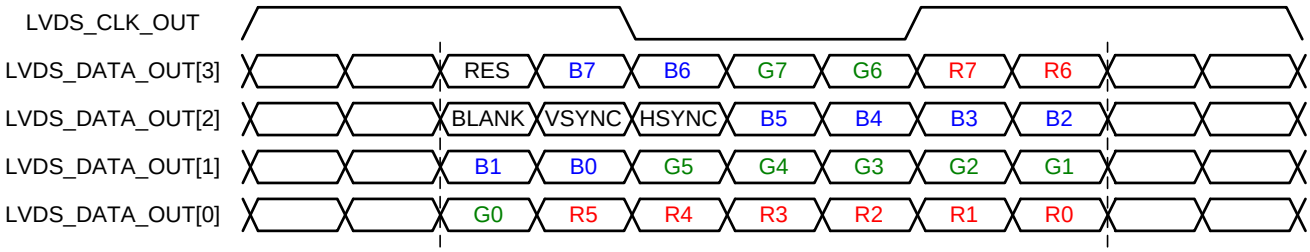


Figure 6.7: Pixel data order on Camera link LVDS data bus

6.3 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, generic parameter C_DISPLAY_INTERFACE has to be set to 1. Please refer to chapter 2.3 Generic Parameters.

Additionally, NTSC or PAL video standard is selected through DTYPE register, see chapter 10.2.6 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 analogue ITU656, i.e. composite video or S-Video output, the digital video encoder shall be connected to logiCVC-ML FPGA output pins. The logiCVC-ML is tested with Philips SAA7121 and Analog Devices ADV7393 digital video encoders.

By setting C_DISPLAY_INTERFACE generic 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 through one bit in the corresponding Layer Control register. For more information, please refer to chapter 10.2.19 Layer Control Register - Lx_CTRL.

Regardless of the ITU656 selection, the logiCVC-ML still drives the parallel pixel interface, PIX_CLK, HSYNC, VSYNC, BLANK and D_PIX[C_PIXEL_DATA_WIDTH-1 : 0], if Power Control register is configured correctly. For more information, refer to chapter 10.2.11 Power Control Register - PWRCTRL.

6.4 Digital Visual Interface (DVI)

The logiCVC-ML supports Digital Visual 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 Spartan-6, 7 Series FPGAs and Zynq™-7000 AP SoC because of two constraints. The first one is that DVI standard requires the use of TMDS IO standard that is only supported in the above mentioned families. The second one is that the implementation is using special IO seralizers that exist only in the above families.

The maximum DVI resolution that logiCVC-ML supports is constrained by the maximum toggle rate of TMDS IOs and clock lines in the required family. We can divide the supported performance into two groups; Spartan-6 and 7 Series. By this, we include the Zynq™-7000 AP SoC into the 7 Series family as the FPGA fabric used in Zynq devices is based on the Kintex-7 or Artix-7 family. So in the below analysis we will treat Zynq as a 7 Series device.

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

Regardless of DVI output selection, the logiCVC-ML still drives the parallel pixel interface, PIX_CLK, HSYNC, VSYNC, BLANK and D_PIX[C_PIXEL_DATA_WIDTH-1 : 0], if Power Control register is configured correctly. For more information, refer to chapter 10.2.11 Power Control Register - PWRCTRL.

6.4.1 Spartan-6 implementation

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 in Spartan-6

Resolution (@60Hz)	Pixel clock (MHz)	Data rate (Mb/s)
VGA (640x480)	25.25	252.5
480p (720x480)	27	270
WVGA (854x480)	32	320
SVGA (800x600)	40	400

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

6.4.2 7 Series implementation

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

Setting C_DVI_CLK_MODE to 0, DVI transmitter uses a MMCM, BUFIO and two BUFR components in order to generate the required clock infrastructure.

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

Setting C_DVI_CLK_MODE to 1, DVI transmitter uses a PLL and three BUFG components in order to generate the required clock infrastructure.

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

Table 6.3 and Table 6.4 outline the maximum performance and supported clock rates for both clock implementation modes. We strongly recommend that users regularly check the latest Xilinx documentation for possible timing changes mentioned in the following tables.

Table 6.3: 7 Series performance

Speed Grade	Mb/s	
	C_DVI_CLK_MODE=0	C_DVI_CLK_MODE=1
-3	1250	1250
-2/-2L/-2G	1250	1250
-1	1250	928

Table 6.4: Common video resolutions supported in 7 Series

Resolution (@60Hz)	Pixel clock (MHz)	Data rate (Mb/s)
VGA (640x480)	25.25	252.5
480p (720x480)	27	270
WVGA (854x480)	32	320

Resolution (@60Hz)	Pixel clock (MHz)	Data rate (Mb/s)
SVGA (800x600)	40	400
XGA (1024x768)	65	650
WXGA (1280x768)	68.25	682.5
HD 720p (1280x720)	74.25	742.5
HD 1080i (1920x1080)	74.25	742.5
SXGA (1280x1024)	108	1080
WSXGA+(1680x1050)	120	1200

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

6.5 CRT Displays

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

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

7 EXTERNAL PARALLEL INPUT INTERFACE

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

- C_USE_E_PARALLEL_INPUT – enables external parallel 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 parallel data (0 – 4)
- C_E_DATA_WIDTH – determines external parallel data width (8, 16, 24)

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

- E_VCLK – external clock
- E_VSYNC – external vertical sync
- E_HSYNC – external horizontal sync
- E_BLANK – external blank
- E_DATA[C_E_DATA_WIDTH – 1 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 parallel input present, i.e. E_VIDEO_PRESENT is low, logiCVC-ML will switch the logic to work on VCLK input clock and will start sending control signals to the display as written in the logiCVC-ML registers.

When 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. Figure 6.1 and Figure 6.2 represent the way logiCVC-ML outputs the control signals by default. In the same way, logiCVC-L expects them at the external parallel input interface. The polarity of input control signals and E_VIDEO_PRESENT signal can be controlled by setting display CTRL register bits 16 to 19 accordingly.

All porch and sync registers in logiCVC-ML are 8 bit so the maximum values that can be measured are 256. Resolution registers are limited with C_ROW_STRIDE 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 10.2.12 External Input Horizontal Resolution - E_HRES and 10.2.13 External Input Vertical Resolution - E_VRES for more information.

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

8 DISPLAY ENHANCEMENT FUNCTIONS

8.1 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 10.2.3 Vertical Sync and Porch Registers - VSY_FP, VSY, VSY_BP.

8.2 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 10.2.5 Control Register - CTRL.

8.3 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_BLIGHT 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 10.2.11 Power Control Register - PWRCTRL.

8.4 Inverse Video

Inverse video can be used with any output interface format. Setting bit 7 in the CTRL register, bit inverts the data output from the logiCVC-ML video controller (e.g. 0xA50011 => 0x5AFFEE), inverting the complete picture data values drawn on the display screen.

8.5 Dithering module

Dithering module is designed for arithmetic calculations on pixels to generate additional levels of colour when lower bit display interfaces are used (18bpp). It is implemented by setting C_XCOLOR generic parameter 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 8.1). 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 8.2).

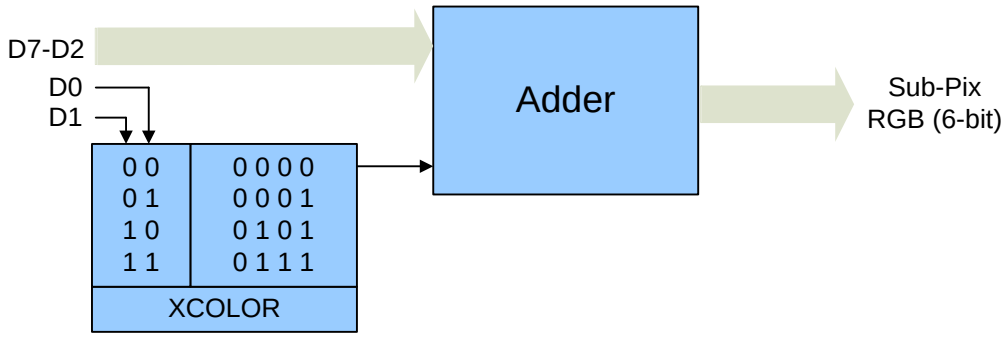


Figure 8.1 XCOLOR dithering module

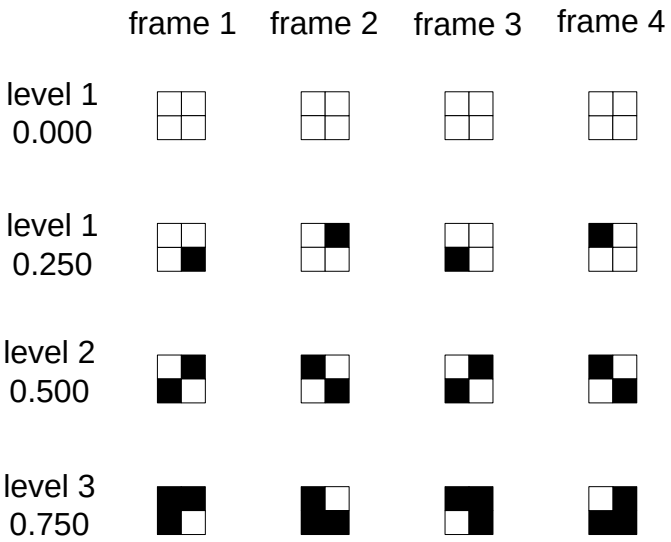


Figure 8.2 XCOLOR temporal and spatial dithering

9 MULTILAYER SUPPORT

logiCVC-ML supports up to 5 layers. The actual number of layers that will be implemented is configurable through the generic parameter C_NUM_OF_LAYERS. Block schematic of multilayer architecture is outlined in Figure 9.1.

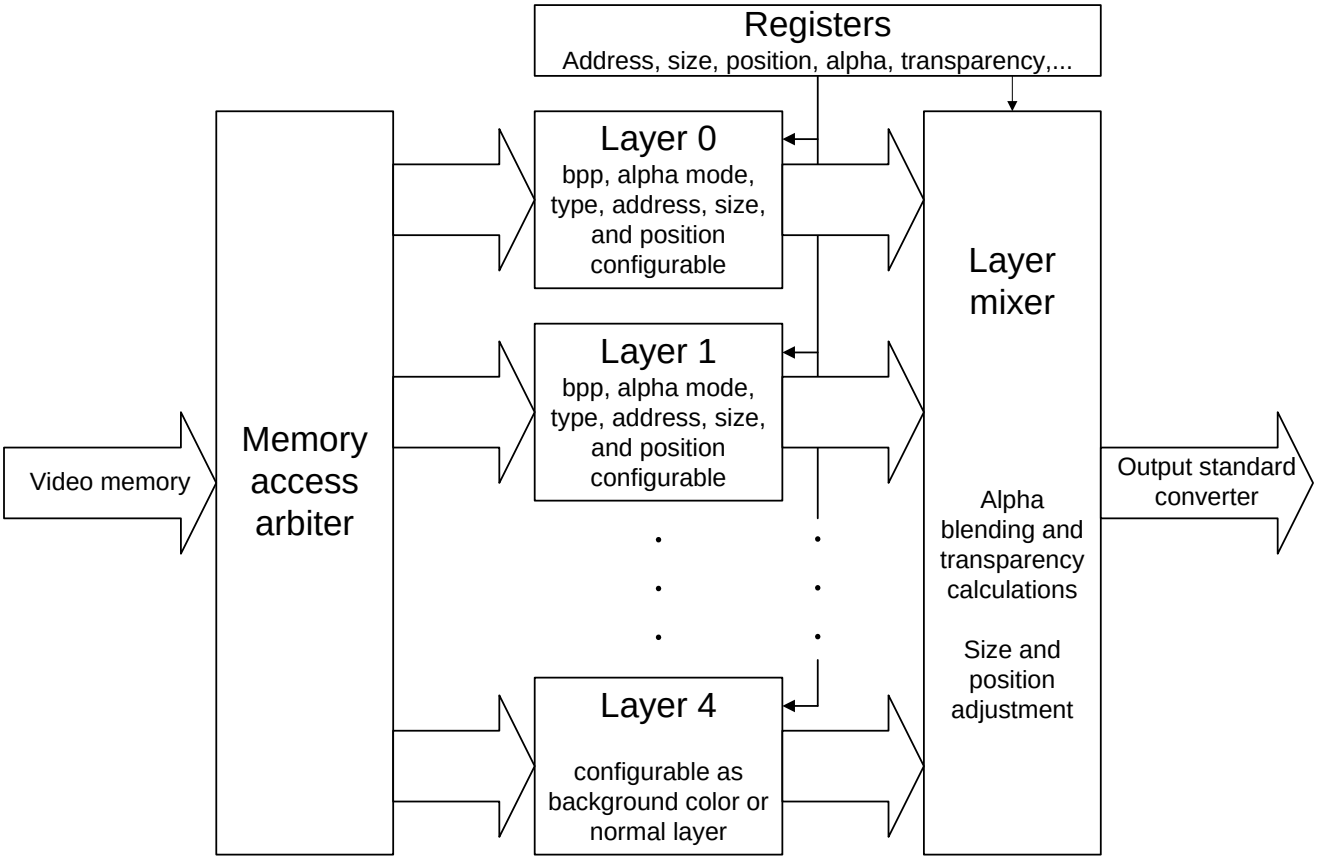


Figure 9.1: Multilayer architecture

9.1 Memory Access Arbiter

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

9.2 Layer Block

Layer consists of the following blocks:

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

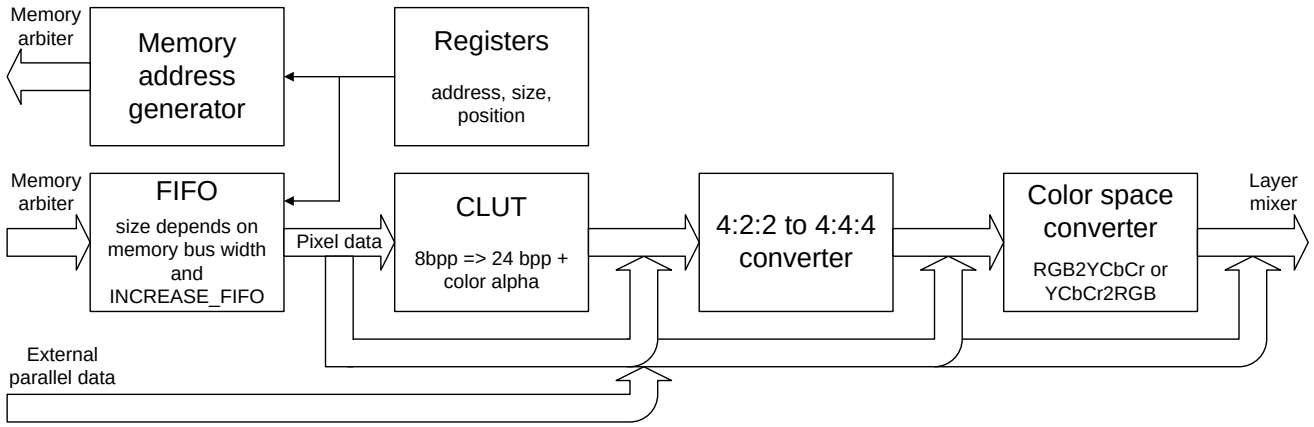


Figure 9.2 Layer block schematic

Depending on the logiCVC-ML configuration, layer can consist of some or all of the above mentioned blocks. For example, if logiCVC-ML is configured to output RGB (C_DISPLAY_COLOR_SPACE=0) and layer is configured as 16bpp RGB, it will only consists of three parts; address generator, FIFO and register set. However, if logiCVC-ML is configured to output RGB (C_DISPLAY_COLOR_SPACE=0) and layer is configured as 16bpp YCbCr (4:2:2) then layer block consists of five parts, address generator, FIFO, register set, 4:2:2 to 4:4:4 converter and YCbCr to RGB converter.

9.2.1 Memory Address Generator

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

In this way, each layer can be placed on the desired location in memory that is user assigned. Please refer to chapter 4.3 Memory address and range, for more information on how to calculate each layer memory range.

Additionally, layer size and position registers are used to resize each layer and move the starting point in memory from which the data will be read, i.e. the first pixel on the display. For more information on memory address, size and position registers and their functionality, please refer to chapters 10.2.15, 10.2.16, 0 and 9.3.

9.2.2 Pixel Data FIFO

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

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

		Memory data bus width		
		32	64	128
C_INCREASE_FIFO	1	1	2	4
	2	2	4	8
	4	4	8	16
	8	8	16	32

9.2.3 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 9.3.2.3 Colour Alpha Blending.

Table 9.2: Data organisation of one CLUT index

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

CLUT data bit	16bpp CLUT	24bpp CLUT	
		RGB	YCbCr
11	Green1	Green3	Cb3
12	Green2	Green4	Cb4
13	Green3	Green5	Cb5
14	Green4	Green6	Cb6
15	Green5	Green7	Cb7
16	Dummy	Red0	Y0
17	Dummy	Red1	Y1
18	Dummy	Red2	Y2
19	Red0	Red3	Y3
20	Red1	Red4	Y4
21	Red2	Red5	Y5
22	Red3	Red6	Y6
23	Red4	Red7	Y7
24	Alpha0	Alpha0	Alpha0
25	Alpha1	Alpha1	Alpha1
26	Alpha2	Alpha2	Alpha2
27	Alpha3	Alpha3	Alpha3
28	Alpha4	Alpha4	Alpha4
29	Alpha5	Alpha5	Alpha5
30	Dummy	Alpha6	Alpha6
31	Dummy	Alpha7	Alpha7

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

In case logiCVC-ML layer is configured as 16bpp YCbCr (C_LAYER_X DATA_WIDTH = 16 and C_LAYER_X_TYPE = 1) it means that data in memory that this layer reads is in YCbCr 4:2:2 format. Therefore, each 16-bit pixel has its own luminance (Y) value but neighbouring pixels share the same chroma values (Cb and Cr). So, in order for the logiCVC-ML alpha blending logic to properly calculate pixel values, 4:2:2 format needs to be converted to 4:4:4. This is achieved, by repeating the chroma values so that each pixel consists of its own Y, Cb and Cr value (Y_0Cb, Y_1Cr => Y_0CbCr, Y_1CbCr).

9.2.5 Colour space converter

In case logiCVC-ML output colour space (C_DISPLAY_COLOR_SPACE) is not the same as layer colour space (C_LAYER_X_TYPE), layer data is converted to the output colour space before alpha blending is performed. In this way, all layers are converted to the same colour space and can be alpha blended together.

If logiCVC-ML is configured to use YCbCr output space (C_DISPLAY_COLOR_SPACE=1 or 2) and some layers are not in YCbCr colour format (C_LAYER_X_TYPE = 0), then their RGB pixel data is converted from RGB to YCbCr colour space. Depending on the output interface selection (parallel or ITU656) different conversion formula coefficients are used. The reason for this is that ITU656 interface requires specific parameters and limits the output range of luminance and chroma values in the range of 16-235. So, in the case of parallel output interface the following formulas are used:

Equation 9.1: Parallel interface RGB to YCbCr conversion formulas

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

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

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

Equation 9.2: ITU656 interface RGB to YCbCr conversion formulas

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

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

If logiCVC-ML is configured to use RGB output space (C_DISPLAY_COLOR_SPACE=0) and some layers are not in RGB colour format (C_LAYER_X_TYPE = 1 or 2), then their YCbCr pixel data is converted from YCbCr to RGB colour space using the following formulas:

Equation 9.3: YCbCr to RGB conversion formulas

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

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

9.3 Layer Mixer

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

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

Figure 9.3 represents logiCVC-ML example configuration. In this example, logiCVC-ML has four layers where the last layer is configured as background layer. The difference between a normal layer and a background one is that background layer does not 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 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 visible because its alpha factor is set to maximum, i.e. 1 (0xFF). Layer 1 has an alpha factor of 0.5 which means that both layers underneath are slightly visible. Layer 2 has an alpha factor of 1 (0xFF) and is completely visible on the places that are not hidden by the upper layers.

Additionally, all layers except the background layer are smaller than horizontal and vertical display resolution and are placed at some positions on the screen. Using 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 10.2.16 Layer Position Registers - Lx_H_POSITION, Lx_V_POSITION and 10.2.17 Layer Size Registers - Lx_WIDTH, Lx_HEIGHT.

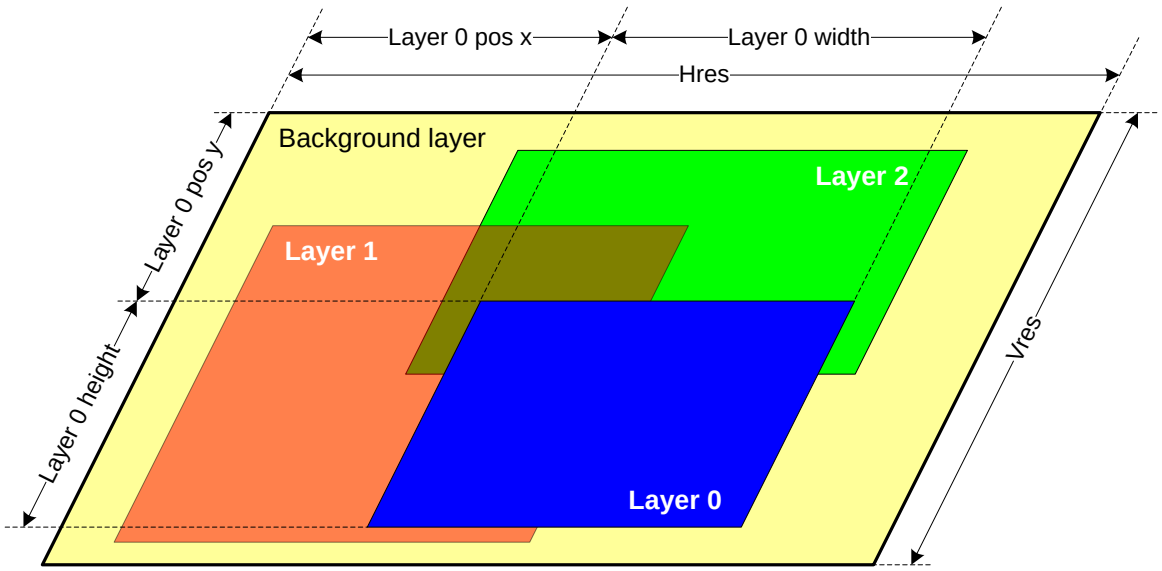


Figure 9.3: logiCVC-ML example configuration

9.3.1 Transparency

Transparency is achieved by using a 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 10.2.20 Layer Transparent Colour Register - Lx_TRANSPARENT for more information.

9.3.2 Alpha Blending

There are three ways to achieve alpha blending, i.e. blending from one picture to the other (also known as morphing). There is layer alpha blending, pixel alpha blending and 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.

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

In the above equation, $layer_0$ is the top layer and $layer_1$ is the layer beneath. The equation is calculated three times for every pixel, once for each RGB colour.

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

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

9.3.2.1 Layer Alpha Blending

When using this mode, alpha factor is stored in layer alpha register (LX_ALPHA) and is constant for all pixels on the specific layer. Figure 9.4 shows a simple usage of layer alpha blending. First picture is representing $layer_0$ and last $layer_1$. The pictures between represent alpha blending with alpha factor steps of 25% (0.25).

Figure 9.4: Layer alpha blending

9.3.2.2 Pixel Alpha Blending

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

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

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

When pixel alpha blending mode is selected by setting C_LAYER_X_ALPHA_MODE to 1, alpha factor is stored together with pixel colour value. In this case, each pixel read from video buffer consists of colour and alpha factor, e.g. ARGB or AYCbCr, and this increases the size of each pixel for alpha bits according to Table 9.3.

Table 9.3: Pixel Alpha blending

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

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. This dummy byte is the downside of this method as it increases the required bandwidth for one byte per pixel (25%).

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

Please note that only layers 1 and 3 can be configured as alpha plane layers (C_LAYER_X_TYPE = 2). In case Layer 1 is configured as alpha plane layer, Layer 1 is considered to hold an alpha value for Layer0 and it is used for alpha blending between Layers 0 and 2. In the same way, if Layer 3 is configured as alpha plane layer it is used for alpha blending between Layers 2 and 4.

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

9.3.2.3 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.

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

10 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/AXI4-Lite Master (logiSPI, Ext CPU OPB/PLB/AXI master,…).

By default, all registers inside logiCVC-ML can be read and write, except IP version and external resolution registers, which can only be read. However, some registers have different write and read values and the reading functionality can be switched off by setting the generic parameter C_READABLE_REGS to 0.

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

10.1 Register Map

All registers are placed at an offset set by generic parameter C_REGS_BASEADDR.

Following table describes the names and addresses of logiCVC-ML registers on the OPB/PLB/AXI4-Lite bus.

Table 10.1: logiCVC-ML register map

Address offset[1)]	Type[2)]	Size (bits)	Name	Description
General Registers				
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	VSY_FP	Vertical Sync front porch
0x0028	R/W	8	VSY	Vertical Sync pulse width
0x0030	R/W	8	VSY_BP	Vertical Sync back porch
0x0038	R/W	16	V_RES	Vertical resolution
0x0040	R/W	24	CTRL	Control
0x0048	R/W	8	DTYPE	Display type
0x0050	R/W	24	BACKGROUND	Background colour
0x0058				Not used
0x0060	R/W	16	CLUT_SEL	CLUT select for all CLUT layers [4)]
0x0068	R/(W)	16	INT_STAT	Interrupt status register [3)]
0x0070	R/W	16	INT_MASK	Interrupt mask register
0x0078	R/W	8	PWRCTRL	Power control

Address offset[1)]	Type[2)]	Size (bits)	Name	Description
0x0080	R	16	E_HRES	External input horizontal resolution
0x0088	R	16	E_VRES	External input vertical resolution
0x00F8	R	32	IP_VERSION	logiCVC-ML IP version information
Layer 0 Registers[5)]				
0x0100	R/W	32	L0_ADDR	Layer0 memory address
0x0108				Not used
0x0110	R/W	16	L0_H_POSITION	Layer0 horizontal position
0x0118	R/W	16	L0_V_POSITION	Layer0 vertical position
0x0120	R/W	16	L0_WIDTH	Layer0 width
0x0128	R/W	16	L0_HEIGHT	Layer0 height
0x0130	R/W	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
Layer 1 Registers[5)]				
0x0180	R/W	32	L1_ADDR	Layer1 memory address
0x0188				Not used
0x0190	R/W	16	L1_H_POSITION	Layer1 horizontal position
0x0198	R/W	16	L1_V_POSITION	Layer1 vertical position
0x01A0	R/W	16	L1_WIDTH	Layer1 width
0x01A8	R/W	16	L1_HEIGHT	Layer1 height
0x01B0	R/W	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
Layer 2 Registers[5)]				
0x0200	R/W	32	L2_ADDR	Layer2 memory address
0x0208				Not used
0x0210	R/W	16	L2_H_POSITION	Layer2 horizontal position
0x0218	R/W	16	L2_V_POSITION	Layer2 vertical position
0x0220	R/W	16	L2_WIDTH	Layer2 width
0x0228	R/W	16	L2_HEIGHT	Layer2 height
0x0230	R/W	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

Address offset[1]	Type[2]	Size (bits)	Name	Description
Layer 3 Registers[5]				
0x0280	R/W	32	L3_ADDR	Layer3 memory address
0x0288				Not used
0x0290	R/W	16	L3_H_POSITION	Layer3 horizontal position
0x0298	R/W	16	L3_V_POSITION	Layer3 vertical position
0x02A0	R/W	16	L3_WIDTH	Layer3 width
0x02A8	R/W	16	L3_HEIGHT	Layer3 height
0x02B0	R/W	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
Layer 4 Registers[5]				
0x0300	R/W	32	L4_ADDR	Layer4 memory address
				Not used
0x0338	R/W	8	L4_CTRL	Layer4 Control
CLUT Registers				
0x1000	R/W	1k Bytes	L0_CLUT_0	Layer0 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

1. All registers are placed at the 64-bit boundary; including CLUT indexes.
2. Readability off all logiCVC-ML registers (except CLUT_SEL, PWRCTRL, INT_STAT and IP_VERSION which can always be read), can be turned OFF with generic parameter C_READABLE_REGS set to 0.
3. Interrupt status register can only be cleared.
4. CLUT select registers have different read/write values.
5. Layer 4 and any layer that is last in the configuration has limited functionality because it can only be the last layer (logiCVC-ML has a maximum of five layers) and there is nothing below this layer that can be shown on the screen. That is the reason some functionality (Lx_H_POSITION, Lx_V_POSITION, Lx_WIDTH, Lx_HEIGHT, Lx_ALPHA and Lx_TRANSPARENT) is not available for layer 4 or any layer that is last in the configuration.

10.2 Register Description

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

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

Table 10.2: HSY_FP Register Description

Bits	Address offset	Name	Type	Reset value	Description
7 - 0	0x000	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 10.3: HSY Register Description

Bits	Address offset	Name	Type	Reset value	Description
7 - 0	0x008	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 10.4: HSY_BP Register Description

Bits	Address offset	Name	Type	Reset value	Description
7 - 0	0x010	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.

Please refer to Figure 6.1 for sync and front and back porch definitions.

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

HSY_FP, HSY and HSY_BP cannot be assigned if ITU656 output standard is used (C_DISPLAY_INTERFACE = 1) because they are predefined by the standard itself.

10.2.2 Horizontal Resolution Register - H_RES

Table 10.5: H_RES Register Description

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

logiCVC-ML supports horizontal resolutions from 64 to 2048 pixels but the maximum horizontal resolution, i.e. the size of horizontal resolution register, is constrained by the C_ROW_STRIDE parameter in a way that the number of bits used in H_RES register is equal to $\log_2$(C_ROW_STRIDE). Otherwise, there are no restrictions on the values written in the registers besides the minimum and the maximum value. The values in the H_RES register should be written in number of pixels.

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

Please refer to Figure 6.1 for horizontal resolution definition.

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.

Please check C_ROW_STRIDE setting for proper functioning. Usually, C_ROW_STRIDE should be set to a maximum required horizontal resolution. For more information on memory row stride, please refer to chapter 4.1 Memory Layout.

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

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

Table 10.6: VSY_FP Register Description

Bits	Address offset	Name	Type	Reset value	Description
7 - 0	0x020	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 10.7: VSY Register Description

Bits	Address offset	Name	Type	Reset value	Description
7 - 0	0x028	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 10.8: VSY_BP Register Description

Bits	Address offset	Name	Type	Reset value	Description
7 - 0	0x030	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).

Please refer to Figure 6.1 for sync and front and back porch definitions.

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

VSY_FP, VSY and VSY_BP cannot be assigned if ITU656 output standard is used (C_DISPLAY_INTERFACE = 1) because they are predefined by the standard itself.

10.2.4 Vertical Resolution Register - V_RES

Table 10.9: V_RES Register Description

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

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.

Please refer to Figure 6.1 for vertical resolution definition.

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.

10.2.5 Control Register - CTRL

Table 10.10: CTRL Register Description

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

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

Table 10.11: Control Register Map

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

GPIO[4:0] bits inside the display control register are directly routed from/to logiCVC-ML input/output signals GPI and GPO. By reading these bits user is presented with the current state of the GPI signal. Performing a write to GPIO bits the state of the GPO signal changes accordingly. This allows the user to use these bits for general purpose. For example, they can be used to control clock signal frequency by adjusting an external clock module or PLL as outlined in Figure 3.1.

10.2.6 Display Type Register - DTYPE

Table 10.12: DTYPE Register Description

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

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

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

Table 10.13: Display Type Register Map

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

NTSC_PAL: This flag is only valid with C_DISPLAY_INTERFACE generic parameter set to 1.

Table 10.14: NTSC_PAL Register Description

NTSC_PAL	Description
0	PAL video standard
1	NTSC video standard

10.2.7 Background Colour Register - BACKGROUND

Table 10.15: BACKGROUND Register Description

Bits	Address offset	Name	Type	Reset value	Description
23 - 0	0x050	BACKGROUND	R/W	0x000000	Background Colour Register

If 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 9.3.

Depending on layer type (RGB or YCbCr), different colour depth modes are supported. For RGB background layer three depths 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. For YCbCr background layer only 24bpp is supported.

10.2.8 CLUT Select Register - CLUT_SEL

Table 10.16: CLUT_SEL Register Description

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

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

Table 10.17: CLUT Select Register Map

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

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

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

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

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

If 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.

10.2.9 Interrupt Status Register - INT_STAT

Table 10.18: INT_STAT Register Description

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

Interrupt status register consists of the following bit groups:

Layer registers update status

Using four interrupt status bits, logiCVC-ML signals that it has performed an update of layer address, size and position registers. This interrupt sources will only become active if there was a write access to the above mentioned registers. Please refer to chapters 10.2.15, 10.2.16 and 0 for more information.

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

V-Sync status

Bit 5 is used for 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 parallel input valid status

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

FIFO underrun status

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

CLUT select status

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

Table 10.19: Interrupt Status Register Map

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

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

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

10.2.10 Interrupt Mask Register - INT_MASK

Table 10.20: INT_MASK Register Description

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

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

Table 10.21: Interrupt Mask Register Map

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

10.2.11 Power Control Register - PWRCTRL

Table 10.22: PWCTRL Register Description

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

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

Additionally, EN_V bit enables the three stated D_PIX, H_SYNC, V_SYNC, BLANK, PIX_CLK, LVDS_DATA_OUT and LVDS_CLK_OUT output signals.

Table 10.23: Power Control Register Map

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

10.2.12 External Input Horizontal Resolution - E_HRES

Table 10.24: E_HRES Register Description

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

If external parallel input interface is enabled (C_USE_E_PARALLEL_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 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).

10.2.13 External Input Vertical Resolution - E_VRES

Table 10.25: E_VRES Register Description

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

If external parallel input interface is enabled (C_USE_E_PARALLEL_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 assertion of E_VIDEO_VALID flag in interrupt status register.

10.2.14 IP Version Register - IP_VERSION

Table 10.26: IP_VERSION Register Description

Bits	Address offset	Name	Type	Reset value1)	Description
31 - 0	0x0F8	IP_VERSION	R	0x00002000	IP Version Register

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

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

Table 10.27: logiCVC IP Version Information

Bits	Description
4 – 0	C_IP_PATCH_LEVEL
10 – 5	C_IP_MINOR_REVISION
16 – 11	C_IP_MAJOR_REVISION
18 – 17	C_IP_LICENSE_TYPE
19	C_IP_LICENSE_CHECK
21 – 20	C_IP_TIME_BEFORE_BREAK
31 – 22	Not used

1. Value stored in this register depends on logiCVC-ML IP version. In the example above, the value is: 0x00002000, which stands for 4_00_a (source version, no license check, infinite time before brake).

10.2.15 Layer Memory Address Register - Lx_ADDR

Table 10.28: LX_ADDR Register Description

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

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

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

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

The size of the lower and higher address parts, outlined in Table 10.29, depends on logiCVC-ML configuration and is determined from pixel row stride value that is explained in more detail in chapter 4.1 Memory Layout, Figure 4.1 and Table 4.1.

Table 10.29: Layer Memory Address Register Map

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

Lower part of the layer address register can only be used if generic parameter C_USE_SIZE_POS is set. If it is not set, these bits are tied to 0.

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

For more information on how much bytes a pixel consists of, please refer to chapter 4.1 Memory Layout and Table 4.2, Table 4.3 and Table 4.4.

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

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

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

If generic parameter C_USE_BACKGROUND is set, last layer is not fetching data from memory so layer memory address register is not used.

10.2.16 Layer Position Registers - Lx_H_POSITION, Lx_V_POSITION

Table 10.30: Lx_H_POSITION Register Description

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

Table 10.31: Lx_V_POSITION Register Description

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

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

has to be moved to the right, the number written to the horizontal position register has to be decreased.

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

Table 10.32: Layer Horizontal Position Register Map

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

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

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

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

Table 10.33: Layer Vertical Position Register Map

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

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

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

To use layer position, generic parameter C_USE_SIZE_POS has to be set.

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

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

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

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

10.2.17 Layer Size Registers - Lx_WIDTH, Lx_HEIGHT

Table 10.34: Lx_WIDTH Register Description

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

Table 10.35: Lx_HEIGHT Register Description

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

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

Table 10.36: Layer Width Register Map

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

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

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

Table 10.37: Layer Height Register Map

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

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

To use layer size feature, generic parameter C_USE_SIZE_POS has to be set.

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

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

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

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

10.2.18 Layer Alpha Register - Lx_ALPHA

Table 10.38: Lx_ALPHA Register Description

Bits	Address offset	Name	Type	Reset value	Description
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 RGB layer mode (RGB565), 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. In 16bpp YCbCr layer mode (4:2:2), all 8 bits are used as each component is 8 bit wide. 8bpp mode uses least significant 3 or 2 bits (bits 2-0 or bits 1-0). Red and green colour use 3 bits, while blue uses 2 bits.

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

Table 10.39: Layer Alpha Register Map

Bits	Name	Colour Depth				Description
		8bpp	16bpp		24bpp	
			RGB	YCbCr		
0	ALPHA_0	ALPHA_0	ALPHA_0	ALPHA_0	ALPHA_0	Layer alpha factor bit 0
1	ALPHA_1	ALPHA_1	ALPHA_1	ALPHA_1	ALPHA_1	Layer alpha factor bit 1
2	ALPHA_2	ALPHA_2	ALPHA_2	ALPHA_2	ALPHA_2	Layer alpha factor bit 2
3	ALPHA_3	Not used	ALPHA_3	ALPHA_3	ALPHA_3	Layer alpha factor bit 3
4	ALPHA_4	Not used	ALPHA_4	ALPHA_4	ALPHA_4	Layer alpha factor bit 4
5	ALPHA_5	Not used	ALPHA_5	ALPHA_5	ALPHA_5	Layer alpha factor bit 5
6	ALPHA_6	Not used	Not used	ALPHA_6	ALPHA_6	Layer alpha factor bit 6
7	ALPHA_7	Not used	Not used	ALPHA_7	ALPHA_7	Layer alpha factor bit 7

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

10.2.19 Layer Control Register - Lx_CTRL

Table 10.40: Lx_CTRL Register Description

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

Layer Enable

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

Disable Transparency

DIS_TRANSP bit is used for disabling 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).

External frame buffer enable

EN_EXT_VBUFF_SW bit is used for enabling external switching of frame buffers. This functionality is used when there is an external video source writing to logiCVC-ML layer video buffer. For more information please refer to chapter 4.4 Frame Buffer Synchronisation.

Interlace enable

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

Pixel format

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

Table 10.41: Layer Control Register Map

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

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

Table 10.42: Supported Layer Pixel Formats

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

10.2.20 Layer Transparent Colour Register - Lx_TRANSPARENT

Table 10.43: 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. 24bpp colour depth mode (RGB or YCbCr 4:4:4) requires use of all three bytes. In 16bpp RGB layer mode (RGB565), two bytes are used for determining transparent colour.

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

8bpp RGB mode uses one byte.

Table 10.44: Layer Transparent Colour Register Map

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

Bits	Name	Colour Depth				Description
		8bpp	16bpp		24bpp	
			RGB	YCbCr		
7	TRANS_7	TRANS_7	TRANS_7	TRANS_7	TRANS_7	Layer transparent colour bit 7
8	TRANS_8	Not used	TRANS_8	TRANS_8	TRANS_8	Layer transparent colour bit 8
9	TRANS_9	Not used	TRANS_9	TRANS_9	TRANS_9	Layer transparent colour bit 9
10	TRANS_0	Not used	TRANS_10	TRANS_10	TRANS_10	Layer transparent colour bit 10
11	TRANS_11	Not used	TRANS_11	TRANS_11	TRANS_11	Layer transparent colour bit 11
12	TRANS_12	Not used	TRANS_12	TRANS_12	TRANS_12	Layer transparent colour bit 12
13	TRANS_13	Not used	TRANS_13	TRANS_13	TRANS_13	Layer transparent colour bit 13
14	TRANS_14	Not used	TRANS_14	TRANS_14	TRANS_14	Layer transparent colour bit 14
15	TRANS_15	Not used	TRANS_15	TRANS_15	TRANS_15	Layer transparent colour bit 15
16	TRANS_16	Not used	Not used	TRANS_16	TRANS_16	Layer transparent colour bit 16
17	TRANS_17	Not used	Not used	TRANS_17	TRANS_17	Layer transparent colour bit 17
18	TRANS_18	Not used	Not used	TRANS_18	TRANS_18	Layer transparent colour bit 18
19	TRANS_19	Not used	Not used	TRANS_19	TRANS_19	Layer transparent colour bit 19
20	TRANS_20	Not used	Not used	TRANS_20	TRANS_20	Layer transparent colour bit 20
21	TRANS_21	Not used	Not used	TRANS_21	TRANS_21	Layer transparent colour bit 21
22	TRANS_22	Not used	Not used	TRANS_22	TRANS_22	Layer transparent colour bit 22
23	TRANS_23	Not used	Not used	TRANS_23	TRANS_23	Layer transparent colour bit 23

1. 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.

11 INTEGRATION

This chapter describes how to synthesize and implement logiCVC-ML HDL code and integrate logiCVC-ML in a host design.
The logiCVC-ML is delivered as encrypted source code module, therefore synthesis is always required.

11.1 Synthesis

The logiCVC-ML source code has configurable number of layers, memory data bus width, pixel format and many more parameters. Prior to synthesis, please select logiCVC-ML configuration parameters through setting generics using Xilinx XPS or Vivado implementation tools. Please refer to chapter 2.3 Generic Parameters for more information.
In order to reduce FPGA resources utilization, user can do the following steps ordered by importance (first on the list has the greatest impact on resource utilization):

- Reduce the number of logiCVC-ML layers in a design
- Turn off layer size, position and offset
- Turn off readable logiCVC-ML registers
- Reduce row stride value
- Reduce memory bus data width to reduce BRAM utilization
- Reduce FIFO size with C_INCREASE_FIFO to reduce BRAM utilization

11.2 Implementation

Recommended logiCVC-ML FPGA implementation options:

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

11.3 IO Interface 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. The following clock signals should be connected to the logiCVC-ML clock inputs: VCLK for video, MPLB_CLK, MCLK or M_AXI_ACLK for memory bus access, OPB_CLK, SPLB_CLK or S_AXI_ACLK for registers access, LVDS_CLK and LVDS_CLKN for LVDS output interface and ITU_CLK_IN for ITU656 output interface. All signals should have stable clock sources. Preferred method of clock generation is using Xilinx Digital Clock Managers (DCMs), Phase Locked Loops (PLLs) or Mixed-Mode Clock Managers (MMCMs).
The VCLK clock signal controls most of the circuits inside the logiCVC-ML core. If External parallel input is used (C_USE_E_PARALLEL_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/AXI4 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/AXI4 frequency and memory bandwidth. For more information, please refer to chapter 4.5 Memory bandwidth requirements.

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

All internal logiCVC-ML registers are controlled either by the OPB_CLK OPB clock signal, by SPLB_CLK PLB clock signal or by S_AXI_ACLK AXI4-Lite clock signal depending on the used register interface, i.e. generic parameter C_REGS_INTERFACE.

11.3.1 Input Signals

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

Writing and reading to logiCVC-ML registers is controlled by the OPB/PLB/AXI4-Lite bus. Please consult Xilinx OPB, PLBv4.6 or AXI4-Lite specification regarding the OPB, PLBv4.6 or AXI4-Lite timing requirements.

The Video memory access is performed via PLBv4.6, XMB or AXI4. Please consult Xilinx PLBv4.6/AXI4 specification or Xylon logiMEM specification regarding timing requirements.

11.3.2 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.

11.4 Timing Constraints

The following timing specifications should be appended to the host design user constraint file (UCF), or Xilinx design constraints (XDC).

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.

Register clock (OPB, SPLB or AXI4-Lite) periods have to be supplemented in lines:

```
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>;
```

Memory clock (MPLB, XMB or AXI4) periods have to be supplemented in lines:

```
TIMESPEC TS_mplb_clk  = PERIOD "mplb_clk" <HERE WRITE PERIOD OF mplb_clk>;
TIMESPEC TS_mclk      = PERIOD "mclk"     <HERE WRITE PERIOD OF mclk>;
TIMESPEC TS_m_axi_aclk = PERIOD "m_axi_aclk"  <HERE WRITE PERIOD OF
m_axi_aclk>;
```

If LVDS output is used in Spartan3, Virtex4, virtex5 or Virtex6, 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 in Spartan-6, 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;
```

12 REVISION HISTORY

Version	Date	Author	Approved by	Note
1.01.a	28.7.2006.	J. Ivanović	J. Ivanović	Initial Xylon release
1.01.b	09.10.2006.	J. Ivanović	J. Ivanović	Same as v1.00.a
1.01.c	17.11.2006.	J. Ivanović	J. Ivanović	Added layer control register
1.02.a	21.04.2007.	J. Ivanović	J. Ivanović	New register widths, interrupt interface, ITU656 support, different buffer switching mechanism…
1.03.a	07.01.2008.	Z. Šafaržik, J. Ivanović	J. Ivanović	Added triple buffer, row stride, resolution up to 2048x2048, XMB port, USE_SIZE_POS, configurable memory burst Corrected LVDS pixel order
1.04.a	07.04.2008.	J. Ivanović	J. Ivanović	PLBv4.6 video memory interface: user selectable OPB or PLBv4.6 Slave register interface Changed some VHDL generic parameter names External control of triple buffering
1.04.c	02.10.2008.	J. Ivanović	J. Ivanović	Added 12-bit multiplexed output mode External RGB input.
1.04.e	27.02.2009.	J. Ivanović	J. Ivanović	Only the name is changed to support new hw version
1.04.f	23.03.2009.	J. Ivanović	J. Ivanović	C_READABLE_REGS, interlaced/progressive ITU656 memory reading, update on ext video input synchronization, maximum addressable memory range
1.04.g	21.09.2009.	Z. Šafaržik	J. Ivanović	Little endianness support added (C_LITTLE_ENDIAN), fixed memory and pixel layout tables
1.04.g	30.09.2009.	R. Končurat	J. Ivanović	On page 41, in table 10.1 logiCVC-ML register map, correction of address offset for Layer0 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	Spartan6 and Virtex6 added to C_FAMILY Parameters C_USE_LVDS and C_USE_ITU656 removed C_DISPLAY_INTERFACE parameter added to replace them both and to add another display interface option: camera link Version parameters added: C_IP_LICENSE_TYPE (IP encryption type); C_IP_MAJOR_REVISION; C_IP_MINOR_REVISION; C_IP_PATCH_LEVEL In addition to these parameters there is a register named IP_VERSION that contains a value representing the IP version ODDR2 components instantiated for Spartan3a, Spartan3e and Spartan6 ODDR components instantiated for Virtex4, Virtex5 and Virtex6
1.05.b	21.12.2009.	K. Mudrovčić	J. Ivanović	User's manual corrections
1.06.a	11.1.2010.	R. Končurat	J. Ivanović	User's manual corrections Bug fix in parallel output interface for Virtex4, Virtex5 and Virtex6 families (ODDR components) Bug fix in external RGB data and video buffer width mismatch
1.06.a	11.1.2010.	A. Jukić	J. Ivanović	New alpha blender New generic parameters added: C_USE_XTREME_DSP and C_USE_MULTIPLIER
1.06.b	13.1.2010.	J. Ivanović	J. Ivanović	Added support for 128-bit XMB/PLB bus Added support for 32 and 64-burst length on 64 and 128 bit PLB bus
1.06.c	2.3.2010.	Z. Šafaržik	J. Ivanović	
1.06.c	19.3.2010.	J. Marjanović	J. Ivanović	Added generic for usage of BUFGMUX for switching vclk to e_vclk Extended CTRL register for inverting polarity of external video control signals
1.06.c	25.3.2010.	A. Cazin	J. Ivanović	Added support for pipelined memory access
1.06.c	10.4.2010.	K. Mudrovčić	J. Ivanović	User's manual corrections
2.00.a	16.9.2010.	J. Marjanović	J. Ivanović	Added support for AXI interfaces

Version	Date	Author	Approved by	Note
2.01.a	14.3.2011.	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	8.02.2012.	J. Marjanović	J. Ivanović	Added XCOLOR dithering module, added support for 7-series devices including LVDS and instructions for clocking, removed Serialized Blender Version parameters added: C_IP_LICENSE_CHECK; C_IP_TIME_BEFORE_BREAK; IP_VERSION register updated
2.05.a	21.03.2012.	M. Mrak	J. Ivanović	Added support for YCbCr 4:4:4 and YCbCr 4:2:2 display output Removed C_LITTLE_ENDIAN and C_REGS_LITTLE_ENDIAN generic and added C_MEM_BYTE_SWAP and C_MEM_LITTLE_ENDIAN and C_REG_BYTE_SWAP DISPLAY_INTERFACE chapter reorganized 12*2 D_PIX interface and vclk2 description, C_USE_VCLK2 generic added User's manual corrections
2.05.b	16.04.2012.	R. Končurat	J. Ivanović	Only the name is changed to support new hw version Generic parameter C_IP_TIME_BEFORE_BREAK modified to support release edition (0 = infinite, 1 = 1h, 2 = 12h) User's manual corrections
2.05.c	03.05.2012.	J. Ivanović	J. Ivanović	Small corrections made in Memory Layout chapter regarding endianess YCbCr interface figure corrected to falling active clock edge Added hyperlinks to register map table

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

Version	Date	Author	Approved by	Note
3.02.a	03.05.2013.	J. Ivanović	J. Ivanović	HSYNC and VSYNC are now edge aligned so parallel interface timing figures are updated accordingly.
3.02.b	23.01.2014.	J. Ivanović	J. Ivanović	Only the name is changed to support new hw version.
4.00.a	01.02.2014.	J. Ivanović	J. Ivanović	Added support for programmable layer address using layer address register. Removed layer offset and vbuff select registers. Removed C_LAYER_OFFSET, C_VMEM_BASEADDR and C_VMEM_HIGHADDR generics. Added C_LAYER_ADDR generics. Changed layer registers update trigger from vertical position register write to layer address register write. Added disable register update bit to control register. Added FIFO underrun bit to interrupt status register. Updated chapters 3 and 4.
4.01.a	21.05.2014.	A. Bogdanić	J. Ivanović	Added current CVC reading buffer output port. Added C_INTERCONNECT_M_AXI_READ_ISSUING parameter to .mpd file. Renamed v_en output port to en_v.
4.01.b	24.07.2014.	J. Ivanović	J. Ivanović	Only the name is changed to support new hw version.
4.1.2	02.10.2014.	R. Končurat	R. Končurat	Documentation version numbering changed according to IP core for Vivado (IP-XACT package). Patch level is replaced by IP core revision tag.