VGA/LCD Core v2.0 Specifications

Author: Richard Herveille
rherveille@opencores.org

Document rev. 1.2
March 20, 2003

This page left intentionally blank

Revision History

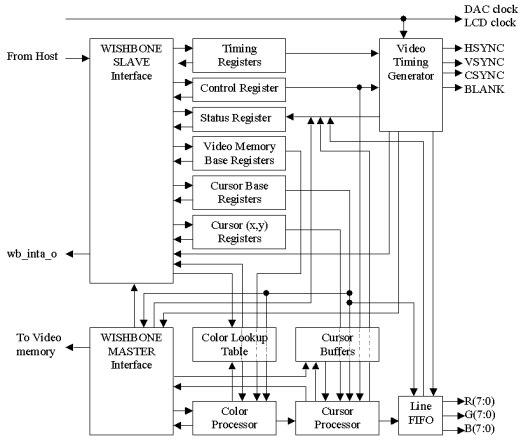
Rev.	Date	Author	Description
0.1	10/04/01	Richard Herveille	First Draft
0.1a	20/04/01	Richard Herveille	Changed *proposal* to *specifications* Added Appendix A Extended Register Specifications
0.2	21/05/01	Richard Herveille	First official release Added OpenCores logo Changed Chapter 1, Introduction Finished Chapter 2, IO ports Finished Chapter 3, Registers Extended Chapter 4, Operation Changed Chapter 5, Architecture Added Appendix B
0.3	28/05/01	Richard Herveille	Fixed some inconsistencies.
0.4	03/06/01	Richard Herveille	Changed all references to address related subjects (core fix & documentation fix). Added Appendix C
0.4a	04/06/01	Richard Herveille	Fixed some minor typing errors in the document (credits: Rudolph Usselmann)
0.5	15/07/01	Richard Herveille	Added Color Lookup Table bank switching. Added embedded CLUT section. Revised horizontal & vertical timing section.
0.6	31/07/01	Richard Herveille	Added Power-on-Reset description. Changed CBSE & VBSE bits functionality. Added Bank Switch Section. Added VGA & CLUT section to Appendix B. Changed introduction page.
0.7	10/19/01	Richard Herveille	Major VGA/LCD Core changes; core v2.0. Changed Manual to reflect core changes. Removed all references to external CLUT v2.0 core has CLUT internally.
0.8	28/01/02	Richard Herveille	Fixed some typos. Added 32bpp mode. Added Bandwidth Issues section.
1.0	28/03/02	Richard Herveille	Expanded Bandwidth Issues section. Added Hardware Cursor sections. Added Table of Contents. Added Appendix-D. Changed Architecture section. Changed Operation section. Changed introduction page. Changed table headers. Added OpenCores logo to page header. Revised entire document.
1.1	20/04/02	Richard Herveille	Changed VGA timing section.
1.2	18/03/03	Richard Herveille	Added support for WISHBONE revB.3 Synchronous Registered Feedback Cycles.

Table of contents

1 Introduction

Features

- **CRT and LCD display support**
- **Separate VSYNC/HSYNC and combined CSYNC synchronization signals**
- **Composite BLANK signal**
- **User programmable video timing**
- **User programmable video resolutions**
- **User programmable video control signals polarization levels**
- **32bpp, 24bpp and 16bpp color modes**
- **8bpp grayscale and 8bpp pseudo-color modes**
- **Supports video- and/or color-lookup-table bank switching during vertical retrace**
- **Support for up to two hardware cursors**
- **Per cursor user selectable resolutions, 23x23 pixels and 64x64 pixels**
- **Alpha blending support for 3D cursors**
- **Triple display support**
- **32bit WISHBONE RevB.3 compliant Slave and Master interfaces**
- **Operation from a wide range of input clock frequencies**
- **Static synchronous design**
- **Full synthesizability**



Core overview

General Description

The OpenCores Enhanced VGA/LCD Controller Core provides VGA capabilities for embedded systems. It supports both CRT and LCD displays with user programmable resolutions and video timings, thus providing compatibility with almost all available LCD and CRT displays.

The core supports a number of color modes, including 32bpp, 24bpp, 16bpp, 8bpp grayscale, and 8bpp-pseudo color. The video memory is located outside the primary core, thus providing the most flexible memory solution possible. It can be located on-chip or off-chip, shared with the system's main memory (VGA on demand) or be dedicated to the VGA system. The color lookup table is located inside the core, to reduce memory bandwidth requirements and to provide higher throughput. Image data is fetched automatically via the WISHBONE Master interface, making this an ideal "program-and-forget" video solution. More demanding video applications, like streaming video or video games, can benefit from the video-bank-switching function. Flicker and cluttered images are reduced by automatically switching between video-memory pages and/or color lookup tables on each vertical retrace.

The optional hardware cursors provide additional flexibility through two 32x32 16bpp or 64x64 4bpp hardware generated cursors. The two cursors can be displayed at the same time.

Typically, one is for the GUI and one for user applications. Cursor patterns are stored in an off-screen portion of the video memory or, if accessible by the core, in the main memory and are automatically loaded into internal buffers to reduce memory bandwidth requirements. Moving the cursors on the screen is as simple as changing a single register.

The core can interrupt the host on each horizontal and/or vertical sync pulse. The horizontal, vertical, and composite synchronization polarization levels, as well as the blanking polarization level are programmable by software.

2

IO ports

2.1 Core Parameters

Parameter	Type	Default	Description
ARST_LVL	Bit	1'b0	Asynchronous reset level
LINE_FIFO_AWIDTH	Integer	7	Line Fifo Size

2.1.1 ARST_LVL

The asynchronous reset level can be set to either active high (1'b1) or active low (1'b0).

2.1.2 LINE_FIFO_AWIDTH

The line FIFO size can be altered by changing the amount of address bits the FIFO logic should use. The line FIFO depth (amount of entries) can be calculated as follows:

$$entries = 2^{LINE\_FIFO\_AWIDTH}$$

2.2 WISHBONE Syscon Interface Connections

Port	Width	Direction	Description
wb_clk_i	1	Input	Master clock input
wb_rst_i	1	Input	Synchronous active high reset
rst_i	1	Input	Asynchronous reset
wb_inta_o	1	Output	Interrupt request signal

2.2.1 wb_clk_i

All internal WISHBONE logic is registered to the rising edge of the [wb_clk_i] clock input. The frequency range over which the core can operate depends on the technology used and the pixel clock needed; [wb_clk_i] may not be slower than the pixel clock [clk_p_i].

2.2.2 wb_rst_i

The active high synchronous reset input [wb_rst_i] forces the core to restart. All internal registers are preset and all state-machines are set to an initial state.

2.2.3 rst_i

The asynchronous reset input [rst_i] forces the core to restart. All internal registers are preset and all state-machines are set to an initial state. The reset level, either active high or active low, is set by the ARST_LVL parameter.

rst_i is not a WISHBONE-compatible signal. It is primarily provided for FPGA implementations. Using [rst_i] instead of [wb_rst_i] can result in lower cell usage and higher performance, because most FPGAs provide a dedicated asynchronous reset path. Use either [rst_i] or [wb_rst_i]. Hardcode the unused reset input to a negated state.

The core requires a power-on reset, allowing all internal registers to propagate to a known state. The power-on reset must be held asserted until all clocks are stable. When all clocks are stable the reset signal must remain asserted for at least 3 clock cycles of the slowest available clock [clk_p_i].

2.2.4 wb_inta_o
The interrupt request output is asserted when the core needs service from the host system.

2.3 WISHBONE Slave Interface Connections

Port	Width	Direction	Description
wbs_adr_i	12	Input	Lower address bits
wbs_dat_i	32	Input	Slave Data bus input
wbs_dat_o	32	Output	Slave Data bus output
wbs_sel_i	4	Input	Byte select signals
wbs_we_i	1	Input	Write enable input
wbs_stb_i	1	Input	Strobe signal/Core select input
wbs_cyc_i	1	Input	Valid bus cycle input
wbs_ack_o	1	Output	Bus cycle acknowledge output
wbs_err_o	1	Output	Bus cycle error output

2.3.1 wbs_adr_i
The address array input [wbs_adr_i] is used to pass a binary coded address to the core. The most significant bit is at the higher number of the array.

2.3.2 wbs_dat_i
The data array input [wbs_dat_i] is used to pass binary data from the current WISHBONE Master to the core. All data transfers are 32bit wide.

2.3.3 wbs_dat_o
The data array output [wbs_dat_o] is used to pass binary data from the core to the current WISHBONE Master. All data transfers are 32bit wide.

2.3.4 wbs_sel_i
The byte select array input [wbs_sel_i] indicates where valid data is placed on the [wbs_dat_i] input array during writes to the core, and where it is expected on the [wbs_dat_o] output array during reads from the core. The core requires all accesses to be 32bit wide [wbs_sel_i(3:0) = ‘1111’b].

2.3.5 wbs_we_i
When asserted, the write enable input [wbs_we_i] indicates whether the current bus cycle is a read or a write cycle. The signal is asserted during write cycles and negated during read cycles.

2.3.6 wbs_stb_i
The strobe input [wbs_stb_i] is asserted when the core is being addressed. The core only responds to WISHBONE cycles when [wbs_stb_i] is asserted, except for the [wb_rst_i] and [rst_i] reset signals, which always receive a response.

2.3.7 wbs_cyc_i
When asserted, the cycle input [wbs_cyc_i] indicates that a valid bus cycle is in progress. The logical AND function of [wbs_cyc_i] and [wbs_stb_i] indicates a valid transfer cycle to/from the core.

2.3.8 wbs_ack_o
When asserted, the acknowledge output [wbs_ack_o] indicates the normal termination of a valid bus cycle.

2.3.9 wbs_err_o
When asserted, the error output [wbs_err_o] indicates an abnormal termination of a bus cycle. The [wbs_err_o] output signal is asserted when the host tries to access the controller's internal registers not using 32-bit aligned data; i.e. when [wbs_sel_i(3:0)] is unequal to '1111'b.

2.4 WISHBONE Master Interface Connections

Port	Width	Direction	Description
wbn_adr_o	32	Output	Address bus output
wbm_dat_i	32	Input	Data bus input
wbm_sel_o	4	Output	Byte select signals
wbm_we_o	1	Output	Write enable output
wbm_stb_o	1	Output	Strobe signal
wbm_cyc_o	1	Output	Valid bus cycle output
wbm_cti_o	3	Output	Cycle type identifier output
Wbm_bte_o	2	Output	Burst type extensions output
wbm_ack_i	1	Input	Bus cycle acknowledge input
wbm_err_i	1	Input	Bus cycle error Input

2.4.1 wbm_adr_o
The address array output [wbm_adr_o] is used to pass a binary coded address from the core to the external video memory. The most significant bit is at the higher number of the array.

2.4.2 wbm_dat_i
The data array input [wbm_dat_i] is used to pass binary data from the external video memory to the core. All data transfers are 32bit wide.

2.4.3 wbm_sel_o
The byte select array output [wbm_sel_o] indicates where valid data is expected on the [wbm_dat_i] input array. The core supports 32-bit wide accesses only [wbm_sel_o(3:0) = '1111'b].

2.4.4 wbm_we_o
When asserted, the write enable output [wbm_we_o] indicates whether the current bus cycle is a read or a write cycle. The core only reads from the external memory; therefore, [wbm_we_o] is always negated ('0').

2.4.5 wbm_stb_o
The strobe output [wbm_stb_o] is asserted when the core wants to read from the external video memory.

2.4.6 wbm_cyc_o
The cycle output [wbm_cyc_o] is asserted when the core wants to read from the external video memory.

2.4.7 wbm_cti_o
The Wishbone revB.3 cycle type identifier output [wbm_cti_o] gives compliant slaves additional information about the current cycle. The vga core supports the Registered Feedback Cycles introduced in the Wishbone revB.3 specs. The core supports 'Classic' and 'Incrementing Burst' transfers. The table below shows the values [wbm_cti_o] can take, any other values should be considered a core error.

wbm_cti_o	Meaning
000b	Wishbone Classic (i.e. revB.2) transfer
010b	Incrementing burst transfer
111b	End-of-Burst

2.4.8 wbm_bte_o
The Wishbone revB.3 burst type extension output [wbm_bte_o] gives compliant slaves additional information about the requested burst. The vga core only supports linear incrementing bursts. Therefore [wbm_bte_o] is always 2'b00.

2.4.9 wbm_ack_i
When asserted, the acknowledge input [wbm_ack_i] indicates the normal termination of a valid bus cycle.

2.4.10 wbm_err_i
When asserted, the error input [wbm_err_i] indicates an abnormal termination of a bus cycle. When the [wbm_err_i] signal is asserted, the core stops the current transfer. After [wbm_err_i] has been asserted, the state of the core is undefined.

2.5 VGA Port Connections

Port	Width	Direction	Description
clk_p_I	1	Input	Pixel Clock
hsync_pad_o	1	Output	Horizontal Synchronization Pulse

vsync_pad_o	1	Output	Vertical Synchronization Pulse
csync_pad_o	1	Output	Composite Synchronization Pulse
blank_pad_o	1	Output	Blank signal
r_pad_o	8	Output	Red Color Data
g_pad_o	8	Output	Green Color Data
b_pad_o	8	Output	Blue Color Data

2.5.1 clk_p_i
All internal video logic is registered to the rising edge of the [clk_p_i] clock input. The frequency range over which the core can operate depends on the technology used and the pixel clock needed; [clk_p_i] may not be faster than the WISHBONE clock [wb_clk_i].

2.5.2 hsync_pad_o
The horizontal synchronization pulse is asserted when the raster scan ray needs to return to the start position (the left side of the screen).

2.5.3 vsync_pad_o
The vertical synchronization pulse is asserted when the raster scan ray needs to return to the vertical start position (the top of the screen).

2.5.5 csync_pad_o
The composite synchronization pulse is a combined horizontal and vertical synchronization signal.

2.5.6 blank_pad_o
The blank output is asserted when no image is projected onto the screen, i.e during the back porch, the synchronization pulses, and the front porch.

2.5.7 r_pad_o, g_pad_o, b_pad_o
Red, green, and blue pixel data: the RGB lines contain invalid data while the BLANK signal [blank_pad_o] is asserted.

3

Registers

3.1 Registers List

Name	wbs_adr_i[11:0]	Width	Access	Description
CTRL	0x000	32	R/W	Control Register
STAT	0x004	32	R/W	Status Register
HTIM	0x008	32	R/W	Horizontal Timing Register
VTIM	0x00C	32	R/W	Vertical Timing Register
HVLEN	0x010	32	R/W	Horizontal and Vertical Length Register
VBARa	0x014	32	R/W	Video Memory Base Address Register A
VBARb	0x018	32	R/W	Video Memory Base Address Register B
	0x01C-0x02C	32	R/W	*reserved*
C0XY	0x030	32	R/W	Cursor0 X,Y Register
C0BAR	0x034	32	R/W	Cursor0 Base Address Register
	0x038-0x03C	32	R/W	*reserved*
C0CR	0x040-0x05C	32	R/W	Cursor0 Color Registers
	0x060-0x06C	32	R/W	*reserved*
C1XY	0x070	32	R/W	Cursor1 X,Y Register
C1BAR	0x074	32	R/W	Cursor1 Base Address Register
	0x078-0x07C	32	R/W	*reserved*
C1CR	0x080-0x09C	32	R/W	Cursor1 Color Registers
	0x0A0-0x7FC	32	R/W	*reserved*
PCLT	0x800-0xFFC	32	R/W	8bpp Pseudo Color Lookup Table

3.2 Accessing Reserved Address Locations

It is not allowed to access reserved memory locations.
No error is generated when these addresses are accessed; all transfers are terminated normally. Write accesses are ignored, read accesses return all zeros.

3.3 Control Register [CTRL]

Bit #	Access	Description
31:26	R/W	*reserved*
25	R/W	HC1R, Hardware Cursor1 Resolution 0: 32x32 pixel mode 1: 64x64 pixel mode
24	R/W	HC1E, Hardware Cursor1 Enable 0: Hardware Cursor1 disabled 1: Hardware Cursor1 enabled
23:22	R/W	*reserved*
21	R/W	HC0R, Hardware Cursor1 Resolution 0: 32x32 pixel mode 1: 64x64 pixel mode
20	R/W	HC0E, Hardware Cursor0 Enable 0: Hardware Cursor0 disabled 1: Hardware Cursor0 enabled
19:16	R/W	*reserved*
15	R/W	BL, Blanking Polarization Level 0: Positive 1: Negative
14	R/W	CSL, Composite Synchronization Pulse Polarization Level 0: Positive 1: Negative
13	R/W	VSL, Vertical Synchronization Pulse Polarization Level 0: Positive 1: Negative
12	R/W	HSL, Horizontal Synchronization Pulse Polarization Level 0: Positive 1: Negative
11	R/W	PC, 8-bit Pseudo Color 0: 8-bit grayscale 1: 8-bit pseudo color
10,9	R/W	CD, Color Depth 11: 32 bits per pixel 10: 24 bits per pixel 01: 16 bits per pixel 00: 8 bits per pixel
8,7	R/W	VBL, Video memory Burst Length 11b: 8 cycles 10b: 4 cycles 01b: 2 cycles 00b: 1 cycle
6	R/W	CBSWE, CLUT Bank Switching Enable 0: Color lookup table bank switching disabled 1: Color lookup table bank switching enabled
5	R/W	VBSWE, Video Bank Switching Enable 0: Video memory bank switching disabled 1: Video memory bank switching enabled
4	R/W	CBSIE, CLUT Bank Switch Interrupt Enable

		0: Color lookup table bank switching interrupt disabled 1: Color lookup table bank switching interrupt enabled
3	R/W	VBSIE, Video Bank Switch Interrupt Enable 0: Video memory bank switching interrupt disabled 1: Video memory bank switching interrupt enabled
2	R/W	HIE, HSync Interrupt Enable 0: Horizontal synchronization pulse interrupt disabled 1: Horizontal synchronization pulse interrupt enabled
1	R/W	VIE, VSync Interrupt Enable 0: Vertical synchronization pulse interrupt disabled 1: Vertical synchronization pulse interrupt enabled
0	R/W	VEN, Video Enable 0: Video system disabled 1: Video system enabled

Reset Value: 0x00000000

3.3.1 BL

The Blanking Polarization Level defines the voltage level of the blank output [blank_pad_o] when the blank signal is asserted. When BL is cleared (‘0’), [blank_pad_o] is at a high voltage level when the blank signal is asserted and at a low voltage level when the blank signal is negated (i.e. blank is active high). When BL is set (‘1’), [blank_pad_o] is at a low voltage level when the blank signal is asserted and at a high voltage level when the blank signal is negated (i.e. blank is active low).

3.3.2 CBSIE

When the CLUT Bank Switch Interrupt Enable bit is set (‘1’) and a bank switch is requested, the host is interrupted. The Bank Switch interrupt is independent of the CLUT Bank Switch Enable bit setting. Setting this bit while the CLUT Bank Switch Interrupt Pending (CBSINT) flag is set generates an interrupt. Clearing this bit while CBSINT is set disables the interrupt request, but does not clear the interrupt pending flag.

3.3.3 CBSWE

When the CLUT Bank Switch Enable bit is set (‘1’) and a complete video frame has been read into the line buffer, the core switches between the two available color lookup tables located at the memory addresses that are set in the CLUT Memory Base Address register. The Active CLUT Memory Page (ACMP) flag reflects the current active color lookup table. The core automatically clears this bit after the bank switch. Software should set this bit each time a bank switch is desired.

3.3.4 CD

The Color Depth bits define the number of bits per pixel (bpp): 8, 16, 24, or 32 bits per pixel.

CD	Color Depth
00b	8bpp
01b	16bpp
10b	24bpp
11b	32bpp

3.3.5 CSL
The Composite Sync Polarization Level defines the voltage level of the composite synchronization output [csync_pad_o] when the composite sync signal is asserted. When CSL is cleared (‘0’), [csync_pad_o] is at a high voltage level when the composite sync signal is asserted and at a low voltage level when the composite sync signal is negated (i.e. csync is active high). When CSL is set (‘1’), [csync_pad_o] is at a low voltage level when the composite sync signal is asserted and at a high voltage level when the composite sync signal is negated (i.e. csync is active low).

3.3.6 HC0E
When the Hardware Cursor0 Enable bit is set (‘1’), the first hardware cursor will be displayed. When it is cleared (‘0’), the hardware cursor will be removed.
To avoid corrupted images, displaying and removing the hardware cursor is synchronous to the vertical retrace; i.e. the cursor will be displayed/removed in the next video frame. All related registers should be set to their corresponding values before enabling the cursor.

3.3.7 HC1E
When the Hardware Cursor1 Enable bit is set (‘1’), the second hardware cursor will be displayed. When it is cleared (‘0’), the hardware cursor will be removed.
To avoid corrupted images, displaying and removing the hardware cursor is synchronous to the vertical retrace; i.e. the cursor will be displayed/removed in the next video frame. All related registers should be set to their corresponding values before enabling the cursor.

3.3.8 HC0R
The Hardware Cursor0 Resolution bit sets the pattern size and the color depth for the first hardware cursor. When HC0R is set (‘1’), hardware cursor0 is set for a resolution of 64x64x4bpp. When HC0R is cleared (‘0’), hardware cursor0 is set for a resolution of 32x32x16bpp. It may not be changed while the cursor is being displayed. To change the cursor’s Resolution bit, first turn off the cursor by clearing the Hardware Cursor0 Enable bit, then change the cursor’s resolution bit value, (re)write the cursor’s Base Address register to load the new cursor pattern, and finally re-enable the cursors by setting the Hardware Cursor0 Enable bit. To avoid displaying corrupted cursors, wait for a vertical sync interrupt after clearing the Hardware Cursor0 Enable bit.

3.3.9 HC1R
The Hardware Cursor1 Resolution bit sets the pattern size and the color depth for the second hardware cursor. When HC1R is set (‘1’), hardware cursor1 is set for a resolution of 64x64x4bpp. When HC1R is cleared (‘0’), hardware cursor1 is set for a resolution of 32x32x16bpp. It may not be changed while the cursor is being displayed. To change the cursor’s Resolution bit, first turn off the cursor by clearing the Hardware Cursor1 Enable bit, then change the cursor’s Resolution bit value, (re)write the cursor’s Base Address register to load the new cursor pattern, and finally re-enable the cursors by setting the Hardware Cursor1 Enable bit. To avoid displaying corrupted cursors, wait for a vertical sync interrupt after clearing the Hardware Cursor1 Enable bit.

3.3.10 HIE

When the Horizontal Interrupt Enable bit is set ('1') and a horizontal interrupt is pending, the host system is interrupted. Setting this bit while the Horizontal Interrupt Pending (HINT) flag is set generates an interrupt. Clearing this bit while HINT is set disables the interrupt request but does not clear the interrupt pending flag.

3.3.11 HSL

The Horizontal Sync Polarization Level defines the voltage level of the horizontal synchronization output [hsync_pad_o] when the horizontal sync signal is asserted. When HSL is cleared ('0'), [hsync_pad_o] is at a high voltage level when the horizontal sync signal is asserted and at a low voltage level when the horizontal sync signal is negated (i.e. hsync is active high). When HSL is set ('1'), [hsync_pad_o] is at a low voltage level when the horizontal sync signal is asserted and at a high voltage level when the horizontal sync signal is negated (i.e. hsync is active low).

3.3.12 PC

When in 8bpp mode, the pixel data can be used as black and white information (256 grayscales) or as an index to a color lookup table (pseudo color mode). When the PC bit is set ('1'), the core operates in pseudo color mode and the pixel data is used to read the color data from the CLUT. When the PC bit is cleared ('0'), the pixel-data is placed on the red, green, and blue outputs, effectively producing a black and white image with 256 different grayscales.

3.3.13 VBSIE

When the Video Bank Switch Interrupt Enable bit is set ('1') and a bank switch is requested, the host is interrupted. The Bank Switch interrupt is independent of the Video Bank Switch Enable bit setting. Setting this bit while the Video Bank Switch Interrupt Pending (VBSINT) flag is set generates an interrupt. Clearing this bit while VBSINT is set disables the interrupt request but does not clear the interrupt pending flag.

3.3.14 VBSWE

When the Video Bank Switch Enable bit is set ('1') and a complete video frame has been read into the line buffer, the core switches between the two available video pages located at the memory addresses set in the Video Memory Base Address (VBAR) registers. The Active Video Memory Page (AVMP) flag reflects the current active video page. The core automatically clears this bit after the bank switch. Software should set this bit each time a bank switch is desired.

3.3.15 VBL

The Video Burst Length bits define the number of transfers during a single block read access to the video memory: 1 (single access), 2, 4, or 8 accesses per block read. The core will perform multiple consecutive block reads; the total number of accesses during a read is therefore always a multiple (i.e. one or more) of the Video Burst Length.

VBL	Burst length
00b	1 transfer
01b	2 transfers
10b	4 transfers
11b	8 transfers

3.3.16 VEN
The video circuit is disabled when the Video Enable bit is cleared (‘0’). The video circuit is enabled when the Video Enable bit is set (‘1’). This bit must be cleared before changing any register contents. After (re)programming all registers, this bit may be set.

3.3.17 VIE
When the Vertical Interrupt Enable bit is set (‘1’) and a vertical interrupt is pending, the host system is interrupted. Setting this bit while the Vertical Interrupt Pending (VINT) flag is set generates an interrupt. Clearing this bit while VINT is set disables the interrupt request but does not clear the interrupt pending flag.

3.3.18 VSL
The Vertical Sync Polarization Level defines the voltage level of the vertical synchronization output [vsync_pad_o] when the vertical sync signal is asserted. When VSL is cleared (‘0’), [vsync_pad_o] is at a high voltage level when the vertical sync signal is asserted and at a low voltage level when the vertical sync signal is negated (i.e. vsync is active high). When VSL is set (‘1’), [vsync_pad_o] is at a low voltage level when the vertical sync signal is asserted and at a high voltage level when the vertical sync signal is negated (i.e. vsync is active low).

3.4 Status Register [STAT]

Bit #	Access	Description
31:25	R	*reserved*
24	R	HC1A, Hardware cursor1 available
23:21	R	*reserved*
20	R	*HC0A, Hardware cursor0 available*
19:18	R	*reserved*
17	R	ACMP, Active CLUT Memory Page
16	R	AVMP, Active Video Memory Page
15:8	R	*reserved*
7	R/W	CBSINT, CLUT Bank Switch Interrupt Pending
6	R/W	VBSINT, Bank Switch Interrupt Pending
5	R/W	HINT, Horizontal Interrupt Pending
4	R/W	VINT, Vertical Interrupt Pending
3:2	R/W	*reserved*
1	R/W	LUINT, Line FIFO Under-Run Interrupt Pending
0	R/W	SINT, System Error Interrupt Pending

Reset Value: 0x00000000 ~ 0x00110000

3.4.1 ACMP

The Active CLUT Memory Page flag is cleared (‘0’) when the active color lookup table is CLUT0; it is set (‘1’) when the active color lookup table is CLUT1. This flag is cleared when the Video Enable bit is cleared. Refer to the CLUT Base Address register for more information on CLUT0 and CLUT1.

3.4.2 AVMP

The Active Video Memory Page flag is cleared (‘0’) when the active memory page is located at Video Base Address A (VBARa); it is set (‘1’) when the active memory page is located at Video Base Address B (VBARb). This flag is cleared when the Video Enable bit is cleared.

3.4.3 CBSINT

The CLUT Bank Switch Interrupt Pending flag is set (‘1’) when all video data from the current active memory page has been translated into pixel colors by the currently active color lookup table. When the CBSIE bit is set (‘1’) and CBSINT is asserted, the host system is interrupted. Software must clear the interrupt by writing a (‘0’) to this bit.

3.4.4 HC0A

The Hardware Cursor0 Available bit is a hard coded flag that is set (‘1’) when Hardware Cursor0 is available and cleared (‘0’) when Hardware Cursor0 is not available.

3.4.5 HC1A

The Hardware Cursor1 Available bit is a hard coded flag that is set (‘1’) when Hardware Cursor1 is available and cleared (‘0’) when Hardware Cursor1 is not available.

3.4.6 HINT
The Horizontal Interrupt Pending flag is set (‘1’) when the horizontal synchronization pulse [hsync_pad_o] is asserted. When the HIE bit is set (‘1’) and HINT is asserted, the host system is interrupted. Software must clear the interrupt by writing a (‘0’) to this bit.

3.4.7 LUINT
The Line FIFO Under-Run Interrupt Pending flag is set (‘1’) when pixels are read from the Line FIFO while it is empty. This can be caused by a locked bus, reading from an illegal video memory address, or to few entries in the FIFO. When LUINT is asserted, the host system is interrupted. Software must clear the interrupt by writing a (‘0’) to this bit.
The Line FIFO Under-Run Interrupt is a non-maskable interrupt.

3.4.8 SINT
The System Error Interrupt Pending flag is set (‘1’) when [wbm_err_i] is asserted during a read from the video memory. When SINT is asserted, the host system is interrupted. Software must clear the interrupt by writing a (‘0’) to this bit.
The System Error Interrupt is a non-maskable interrupt.

3.4.9 VBSINT
The Video Bank Switch Interrupt Pending flag is set (‘1’) when all video data from the current active memory page has been read. When the VBSIE bit is set (‘1’) and VBSINT is asserted, the host system is interrupted. Software must clear the interrupt by writing a (‘0’) to this bit.

3.4.10 VINT
The Vertical Interrupt Pending flag is set (‘1’) when the vertical synchronization pulse [vsync_pad_o] is asserted. When the VIE bit is set (‘1’) and VINT is asserted, the host system is interrupted. Software must clear the interrupt by writing a (‘0’) to this bit.

3.5 Horizontal Timing Register [HTIM]

Bit #	Access	Description
31:24	R/W	Thsync, Horizontal synchronization pulse width
23:16	R/W	Thgdel, Horizontal gate delay time
15:0	R/W	Thgate, Horizontal gate time

Reset Value: 0x00000000

3.5.1 Thsync
The horizontal synchronization pulse width, measured in pixels -1.
Example: Thsync = 5 → hsync length = 6 pixels

3.5.2 Thgdel
The horizontal gate delay width, measured in pixels -1.
Example: Thgdel = 12 → gate delay = 13 pixels

3.5.3 Thgate
The horizontal gate width, measured in pixels -1.
Example: Thgate = 799 → gate length = 800 pixels

The horizontal gate width is dependent on the programmed Video memory Burst Length [VBL] and the Color Depth [CD]. It must be divisible by the burst length and the number of pixels per memory access; see the table below for more information.

CD	(Thgate +1) dividable by:
00b	$4 * VBL$
01b	$2 * VBL$
10b	$\frac{4}{3} * VBL$
11b	$1 * VBL$

3.6 Vertical Timing Register [VTIM]

Bit #	Access	Description
31:24	R/W	Tvsync, vertical synchronization pulse width
23:16	R/W	Tvgdel, vertical gate delay time
15:0	R/W	Tvgate, vertical gate time

Reset Value: 0x00000000

3.6.1 Tvsync
The vertical synchronization pulse width, measured in horizontal lines -1.
Example: Tvsync = 5 → vsync length = 6 lines

3.6.2 Tvgdel
The vertical gate delay time, measured in horizontal lines -1.
Example: Tvgdel = 2 → gate delay = 3 lines

3.6.3 Tvgate
The vertical gate width, measured in horizontal lines -1.
Example: Tgate = 479 → gate length = 480 lines

3.7 Horizontal and Vertical Length Register [HVLEN]

Bit #	Access	Description
31:16	R/W	Thlen, horizontal length
15:0	R/W	Tvlen, vertical length

Reset Value: 0x00000000

3.7.1 Thlen
The total horizontal line time, measured in pixels –1.
Example: Thlen = 1023 → line length = 1024 pixels

3.7.2 Tvlen

The total vertical frame time, measured in horizontal lines -1.
Example: Tvlen = 599 → frame length = 600 lines

3.8 Video Base Address [VBARa] [VBARb]

Bit #	Access	Description
31:2	R/W	VBA, Video Base Address
1:0	R	*Always zero*

Reset Value: 0x00000000

3.8.1 Video Base Address

The Video Base Address register defines the starting point of the video memory. The image is stored in consecutive memory locations, starting at this address. The byte memory location of a pixel can be calculated as follows:

Adr = ((Y * Thgate) + X) * bytes_per_pixel;

The core supports memories with burst capabilities. Burst transfers of 1, 2, 4, and 8 accesses are supported. The lower address bits must reflect the value entered in the Video Memory Burst Length bits as shown in the table below, where an 'x' represents a don't care value.

VBL	VBAR[4:0]*
00b	xxx00b
01b	xx000b
10b	x0000b
11b	00000b

3.9 Hardware Cursor Base Address [C0BAR] [C1BAR]

Bit #	Access	Description
31:10	R/W	CBA, Cursor Base Address
9:0	R	*Always zero*

Reset Value: 0x00000000

3.9.1 Cursor Base Address

The Cursor Base Address register defines the starting point of the cursor pattern to use. The cursor pattern is stored in consecutive memory locations, starting at this address.

3.10 Hardware Cursor (X,Y) Register [C0XY] [C1XY]

Bit #	Access	Description
31:16	R/W	CY, Cursor Y location
15:0	R/W	CX, Cursor X location

Reset Value: 0x00000000

3.10.1 CY

The cursor's upper left pixel's vertical position related to the upper left corner of the image. CY is always positive, i.e. a larger value means moving the cursor down the screen. A smaller value means moving the cursor up the screen.

3.10.2 CX

The cursor's upper left pixel's horizontal position related to the upper left corner of the image. CX is always positive, i.e. a larger value means moving the cursor to the right of the screen. A smaller value means moving the cursor to the left of the screen.

3.11 Hardware Cursor Color Registers [C0CR] [C1CR]

Bit #	Access	Description
31:16	R/W	Color data (odd numbered color register)
15:0	R/W	Color data (even numbered color register)

Reset Value: 0x00000000

3.11.1 Cursor Color Register

The Cursor Color registers define the cursor colors for 64x64x4bpp cursor mode, which is enabled when the Hardware Cursor Resolution bit is set ('1'). In this mode each cursor pixel uses 4bits. The 4bits are used in a lookup table fashion to select a single color register from a total of 16. The 16 color registers are mapped to 8 addresses, where the 16LSBs store an even-numbered color register (i.e. 0, 2, 4, etc) and the 16MSBs store an odd-numbered color register (i.e. 1, 3, 5, etc).

Address Cursor0	Address Cursor1	Bit 31:16	Bit 15:0
0x028	0x058	Color Register 1	Color Register 0
0x02c	0x05C	Color Register 3	Color Register 2

0x030	0x060	Color Register 5	Color Register 4
0x034	0x064	Color Register 7	Color Register 6
0x038	0x068	Color Register 9	Color Register 8
0x03C	0x06C	Color Register 11	Color Register 10
0x040	0x070	Color Register 13	Color Register 12
0x044	0x074	Color Register 15	Color Register 14

Reset Value: *undefined*

These registers are available only when the dedicated hardware cursor is implemented, i.e. C0CR is available when hardware cursor0 is available, and C1CR is available when hardware cursor1 is available. Whether or not a hardware cursor is implemented can be checked via the Status register. When a hardware cursor is not implemented the memory locations are reserved and the rules for accessing reserved memory locations apply.

Note: The contents of these registers is undefined after a reset.

3.12 8bpp Pseudo Color Lookup Table [PCLT]

3.12.1 Color Lookup Table

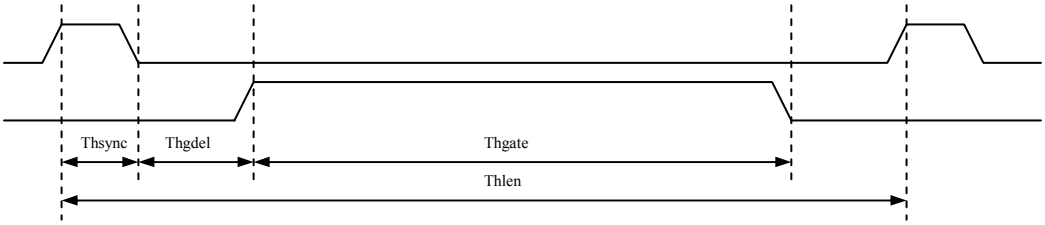
The color lookup table is mapped into the core's address range. It can be accessed (read and write) via the WISHBONE Slave interface, starting at address 0x800. See section *4.2.5 Color Lookup Table* for more information.

4

Operation

4.1 Video Timing

4.1.1 Horizontal Video Timing



4.1.1.1 Thsync

The Horizontal Synchronization Time is the duration of the horizontal synchronization pulse, measured in pixel clock ticks.

4.1.1.2 Thgdel

The Horizontal Gate Delay Time is the duration of the time between the end of the horizontal synchronization pulse and the start of the horizontal gate, measured in pixel clock ticks. The image can be shifted left/right over the screen by modifying Thgdel. In video timing diagrams, this is mostly referred to as the back porch.

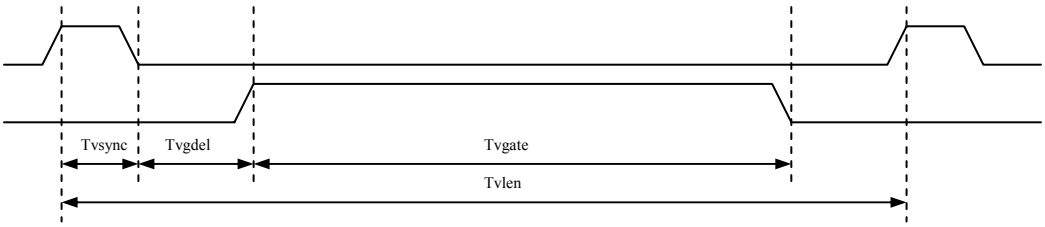
4.1.1.3 Thgate

The Horizontal Gate Time is the duration of the visible area of a video line, measured in pixel clock ticks. In video timing diagrams, this is mostly referred to as the active time.

4.1.1.4 Thlen

The Horizontal Length Time is the duration of a complete video line, from the start of the horizontal synchronization pulse till the start of the next horizontal synchronization pulse, measured in pixel clock ticks.

4.1.2 Vertical Video Timing



4.1.2.1 Tvsync

The Vertical Synchronization Time is the duration of the vertical synchronization pulse, measured in horizontal lines.

4.1.2.2 Tvgdel

The Vertical Gate Delay Time is the duration of the time between the end of the vertical synchronization pulse and the start of the vertical gate, measured in horizontal lines. The image can be shifted up/down the screen by modifying Tvgdel. In video timing diagrams, this is mostly referred to as the back porch.

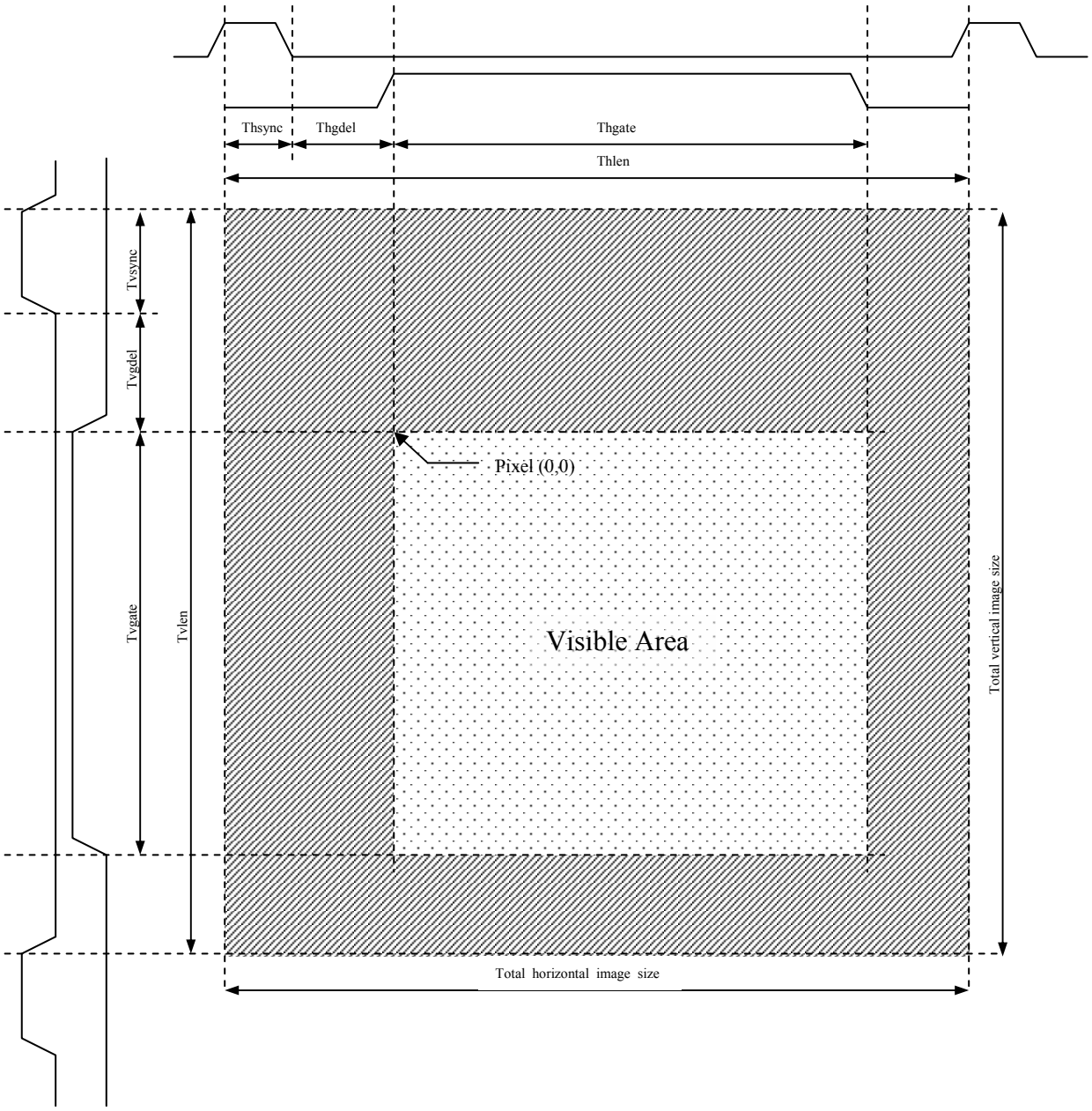
4.1.2.3 Tvgate

The Vertical Gate Time is the duration of the visible area of a video frame, measured in horizontal lines. In video timing diagrams, this is mostly referred to as the active time.

4.1.2.4 Tvlen

The Vertical Length Time is the duration of a complete video frame, from the start of the vertical synchronization pulse till the start of the next vertical synchronization pulse, measured in horizontal lines.

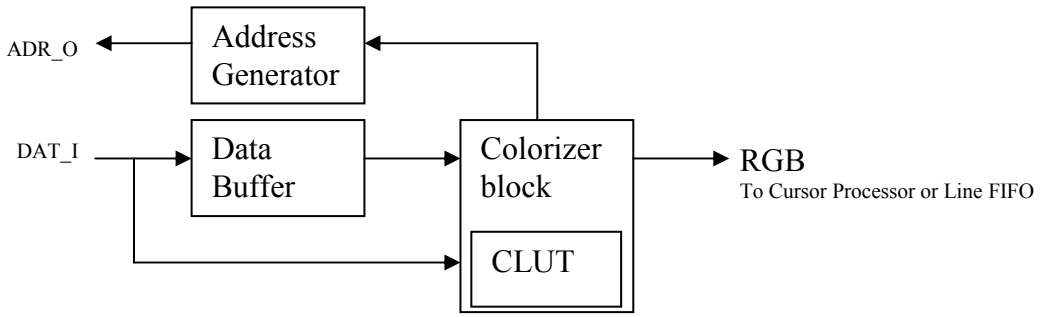
4.1.3 Combined Video Frame Timing



The video frame is composed of Tvlen video lines, each Thlen pixels long. The logical AND function of the horizontal gate and the vertical gate defines the visible area, the rest of the image is blanked.

4.2 Pixel Color Generation

4.2.1 Color Processor Internals



The Color Processor, together with the WISHBONE Master interface and the Line FIFO, handles the pixel color generation. The internal structure of the Color Processor, including parts of the WISHBONE Master interface, is shown in the figure above.

4.2.2 Address Generator

The address generator is part of the WISHBONE Master interface. It generates the video memory addresses, performs video memory bank switching, and keeps track of the number of pixels to read. When all pixels are read, the video memory bank is switched, the video memory offset (i.e. the pixel counter) is reset and - when enabled - the bank switch interrupt is generated. The bank switch interrupt is only dependent on the amount of pixels read, i.e. it has no fixed timing relation to the horizontal or vertical synchronization pulses.

4.2.3 Data Buffer

The data buffer temporarily stores the data read from the video memory. It can contain 16 32-bit entries. The system tries to keep the data buffer at least half full. The data is read from the video memory by a consecutive address burst; i.e. [wbm_cab_o] is asserted. The burst length is determined by the Video memory Burst Length [VBL] bits in the control registers. It is possible that multiple burst accesses are executed within a single access cycle.
All data is stored consecutively, and all available bits are used independent of color depth. In 8bpp mode, a 32-bit word stores 4 pixels; in 16bpp mode it stores 2 pixels, in 24bpp mode 1 1/3 pixels, and in 32bpp 1 pixel.

4.2.4 Colorizer

The colorizer translates the data stored in the data buffer into colors (see the examples below).

The table below shows the Data Buffer contents used in the examples. Only 8 out of the 16 possible entries are shown. The buffer is read from the top to the bottom, i.e. 0x01234567 is the first data read, and 0x89abcdef is the second etc.

Data Buffer contents
0x01234567
0x89abcdef
0x01234567
0x89abcdef
0x01234567
0x89abcdef
0x01234567
0x89abcdef

4.2.4.1 32bpp example.

In 32-bits-per-pixel mode, the lower 24 bits carry the pixel data. The upper 8 bits are ignored, they can be used for Z-buffer, alpha channel, stencil buffer, or similar purposes.

The table below shows the RGB values generated from the sample data in the Data Buffer. Only the first 4 pixels are shown.

Color Data	R	G	B
0x01234567	0x23	0x45	0x67
0x89abcdef	0xab	0xcd	0xef
0x01234567	0x23	0x45	0x67
0x89abcdef	0xab	0xcd	0xef

4.2.4.2 24bpp example.

In 24-bits-per-pixel mode, the RGB values are generated as shown in the following sequence: Da(31:8), Da(7:0)Db(31:16), Db(15:0)Dc(31:24), Dc(23:0).

The table below shows the RGB values generated from the sample data in the Data Buffer.

Color Data	R	G	B
0x12345	0x01	0x23	0x45
0x6789ab	0x67	0x89	0xab
0xcdef01	0xcd	0xef	0x01
0x234567	0x23	0x45	0x67

4.2.4.3 TrippleDisplay mode

The system is capable of driving up to three different displays at the same time. The system operates in TrippleDisplay mode when it is setup for 24bpp mode, but each of the three colors contains grayscale information for a single display.

4.2.4.4 16bpp example.

In 16-bits-per-pixel mode, the upper 16bits carry the data for the first pixel and the lower 16 bits carry the data for the second pixel. The 24-bit RGB data is extracted from the 16-bit color data as follows:
R(7:0) = color_data(15:11), 000b
G(7:0) = color_data(10:5), 00b
B(7:0) = color_data(4:0), 000b

The table below shows the RGB values generated from the sample data in the Data Buffer. Only the first 4 pixels are shown.

Color Data	R	G	B
0x0123	0x00	0x24	0x18
0x4567	0x40	0xac	0x38
0x89ab	0x88	0x34	0x58
0xcdef	0xc8	0xbc	0x78

4.2.4.5 8bpp grayscale example.

In 8-bits-per-pixel grayscale mode, the color data for each of the three colors are equal. The information stored in one byte is sent to all three colors, effectively producing a black-and-white image with 256 grayscales.

The table below shows the RGB values generated from the sample data in the Data Buffer. Only the first 4 pixels are shown.

Color Data	R	G	B
0x01	0x01	0x01	0x01
0x23	0x23	0x23	0x23
0x45	0x45	0x45	0x45
0x67	0x67	0x67	0x67

4.2.4.6 8bpp pseudo-color example.

In 8-bits-per-pixel pseudo-color mode, the color data represents an offset in the internal color lookup table (CLUT). The CLUT contains the RGB color information. This way it is possible to generate an image with 256 different colors with minimal memory requirements.
R = clut_data_out(23:16)
G = clut_data_out(15:8)
B = clut_data_out(7:0)

The table below shows the CLUT addresses for the first 4 pixels.

Color Data	CLUT offset
0x01	0x01
0x23	0x23
0x45	0x45
0x67	0x67

4.2.5 Color Lookup Table

The color lookup table (or CLUT) is a 512x24 bit single-clock synchronous static random access memory divided into two separate CLUTs, of 256x24 bit each. Either one of them is accessed by the colorizer, depending on the Active CLUT Memory Page [ACMP] flag in the Status register. When the ACMP flag is cleared (‘0’), CLUT0 is accessed. When the ACMP flag is set (‘1’), CLUT1 is accessed.

The CLUT memory is mapped into the core’s address range. It can be externally accessed (read and write) via the WISHBONE Slave interface, starting at address 0x800. CLUT0 is located at memory range 0x800 – 0xBFC, CLUT1 at 0xC00 – 0xFFC. All external accesses to the CLUT are 32-bit, but the CLUT itself is only 24 bit wide. The top-most bits[31:24] are ignored for write accesses and are always zero for read accesses.

4.3 Hardware Cursors

4.3.1 Introduction

The Enhanced VGA/LCD Core provides up to two hardware cursors. If and which of the two cursors are implemented is dependent on the system designer. The core takes two definition-parameters (VGA_HWC0 and VGA_HWC1) as input. The define statements are located in the "vga_defines.v" file. If both definition parameters are undefined, no logic is generated for the hardware cursors. If a definition parameter is defined, logic for the appropriate cursor is generated.

Cursor0 is normally used to provide the arrow pointer in GUI applications and operating systems. Cursor1 has no pre-assigned purpose; it can be used to provide some form of user cursor in a pop-up window.

Off-screen memory in the frame buffer or, if accessible by the core, system memory is used to provide the locations where the patterns for both cursors are stored. This allows each cursor to be displayed and used without altering the main display image stored in the frame buffer. The hardware takes care of selecting between the cursor and the image. The Cursor Base Address register determines the cursor's pattern location. Each cursor may have multiple patterns stored in memory, making it possible to change each cursor's appearance by switching from one pattern to another by simply changing the appropriate Base Address register.

4.3.2 Cursor Patterns

The amount of memory allocated for each cursor pattern is 16Kbit. The cursor resolutions are user-selectable, either 32x32 pixels and 16bpp color depth, or 64x64 pixels and 4bpp color depth. The cursor pattern is stored in consecutive memory locations, starting at the address set by the cursor's Base Address register. Each address location contains data for multiple cursor pixels: 2 pixels in 32x32 pixel mode and 8 pixels in 64x64 pixel mode.

4.3.2.1 32x32 Pixel Mode

In 32x32 pixel mode, each pixel has a 16-bit color depth, divided into a selection bit and 15-bit cursor colors (32Kcolors) or an 8-bit alpha channel. The MSB selects between cursor color mode and alpha channel mode. The alpha channel is used to generate transparent pixels or 3D effects (see Pattern Color Data).

4.3.2.2 64x64 Pixel Mode

In 64x64 pixel mode, each pixel has a 4-bit color depth. The 4 bits are used in a lookup table fashion to select a Color register from the available 16 Cursor Color registers. Each Color register contains a 16-bit value, that has the same features as the cursor pattern data in the 32x32 pixel mode, i.e. 1 selection bit and 15 color bits or an 8-bit alpha channel.

4.3.2.3 Cursor Pixel Data

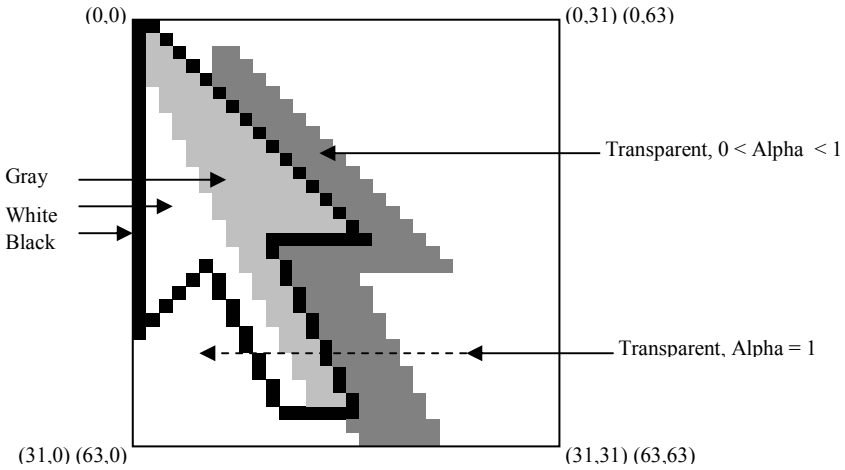
Each cursor pixel is represented by a 16-bit color value. The MSB selects between color mode and Alpha/Transparency mode.

bit 15	bit 14:8	bit 7:0	
0	Color Data		Color mode
1	always zero	Alpha Data	Alpha / Transparency mode

In color mode the LSBs represent a 15-bit RGB value, resulting in 32K colors. The 32K cursor colors are generated by equally distributing the color information over the RGB components, i.e. 5 bits for red, 5 bits for green, and 5 bits for blue. Internally the 5 bit R, G, and B values are extended to 8 bits, the lower 3 bits for each color are set to zero.

In Alpha/Transparency mode, the LSBs are divided into two sections. The first section (bits 14:8) is reserved and should always be read and written as zero. The second section (bits 7:0) represents an 8-bit alpha value. The alpha value is a cross-fader setting between the image pixel value and the black level (RGB = 0). Alpha is normally defined as a value between 0 and 1, where 0 = '00'hex and 1 = 'FF'hex. Setting the Alpha value to 0 results in the black level being displayed. Setting the Alpha value to 1 results in the image pixel being displayed. Any value between 0 and 1 results in a linear mix between the image pixel value and black. This can be used to add the effect of shadow to a cursor, thus creating 3D cursors.

The image below shows how to create the 3D cursor from the Redwood scheme.

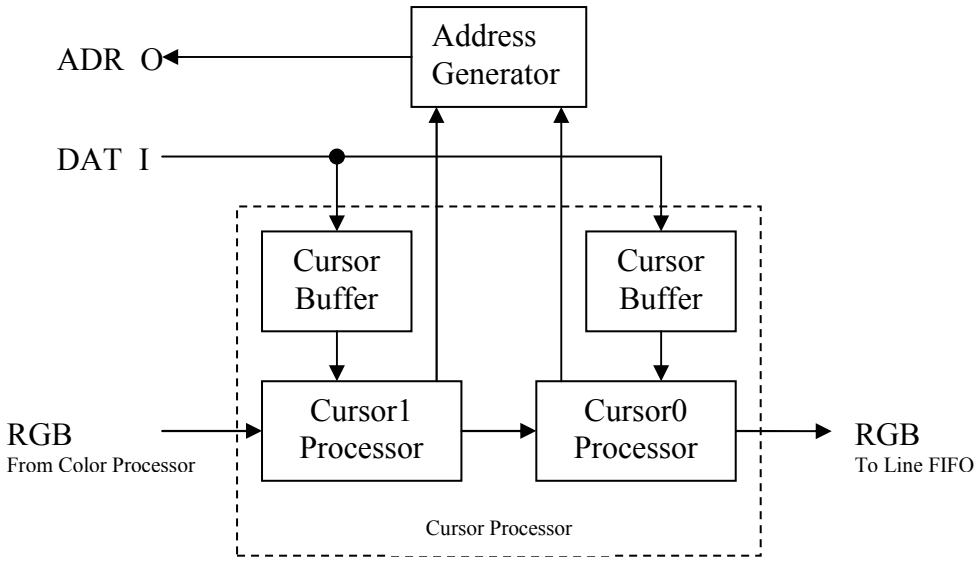


4.3.3 Turning off 3D support.

The alpha-blending logic requires quite an amount of resources. Therefore, the ability to turn off the 3D support has been provided. When 3D support is turned off, the cursor processor ignores the alpha data and generates a transparent pixel. Instead of a shadow effect, the image pixel is displayed. This behavior guarantees that the 3D and non-3D cursors are displayed correctly.

3D cursor support is enabled when VGA_HWC_3D is defined.
3D cursor support is disabled when VGA_HWC_3D is undefined.

4.3.4 Cursor Processor Internals



The Cursor Processor handles the hardware cursors together with the WISHBONE Master interface. The internal structure of the Cursor Processor, including parts of the WISHBONE Master interface is shown in the figure above. If a cursor is not implemented, it is a pass-through function. The above schematic still applies, but no logic is generated for that cursor.

4.3.5 Address Generator

The address generator is part of the WISHBONE Master interface. When copying a cursor into one of the cursor buffers, it generates the memory addresses and writes the data read into the buffers. The cursor processors issue a *cursor read request* to the address generator when their corresponding Cursor Base address [C0BAR][C1BAR] is written to. When the WISHBONE Master finishes reading the current video frame it honors one cursor read request. The cursor data is read in one continuous stream before the start of the next frame.

If both cursors need to be reloaded, one is reloaded before the next frame. It's cursor read request is negated. The second cursor read request is not honored; it remains asserted. When the WISHBONE Master finishes reading the new frame, it honors the second cursor read request.

Cursor0 has a higher priority than Cursor1. When both cursors need to be reloaded, Cursor0 is reloaded first. This implies that continuously reloading Cursor0 results in Cursor1 never being reloaded. However, this situation should never occur during normal operation.

4.3.6 Cursor Buffer

The cursor buffers are 512x32 bit synchronous static random access memories. The address generator writes a copy of the cursor pattern into the cursor buffer whenever the cursor base address [C0BAR][C1BAR] is written to.

4.3.7 Cursor0/Cursor1 Processor

The two cursor processors are the intelligent part of the cursor system. Each cursor processor handles 1 cursor. It keeps track of the raster-scan position, determines whether or not the cursor pattern should be updated, whether or not the cursor should be displayed, and generates the cursor colors including the alpha mixing.

4.4 Bank switching

4.4.1 Introduction

The bank switching system is implemented as a double buffering scheme, also known as a Ping-Pong system. The core reads pixel information from one memory bank while the second bank is being filled. When the second bank has been filled, the host sets the Video Bank Switch Enable bit [VBSE] and/or the Color Lookup Table Bank Switch Enable bit [CBSE]. The core finishes reading the current bank until the entire frame has been read. It then switches to the second bank and starts reading the new frame. The core automatically resets the VBSE and CBSE bits to avoid accidentally switching to the previous bank. A Video Bank Switch Interrupt is generated when the core switches between the two video memory banks, and a CLUT Bank Switch Interrupt is generated when the core switches between the two Color Lookup Tables.

4.4.2 Host notes

The host should not set the VBSE or CBSE bits until all frame information has been written to the video memory. The host system should wait for the Bank Switch Interrupt before filling the previous memory bank.

4.4.3 Sequence

1) Fill video bank0.
2) Fill video bank1.
3) Set VBSE, CBSE, BSIE.
4) Wait for interrupt.
5) Fill video bank0.
6) Set VBSE, CBSE.
7) Wait for interrupt.
8) Fill video bank1.
9) Set VBSE, CBSE.
10) Go to step 4.

4.5 Bandwidth Issues

4.5.1 Introduction

Video displays are real-time devices. The video data stream needs to be generated uninterrupted, or images will be corrupted. The VGA_LCD core provides some flexibility through the use of internal FIFOS, including the large dual-clocked Line-FIFO. But still the average bandwidth required by the video must be met.

4.5.2 Calculations

The required video bandwidth can be calculated using the following formula:

$$BW_{video} = Hpix * Vlin * F_{refr} \ (pps)$$
$$Hpix = number\_of\_visible\_horizontal\_pixels \ (T_{hgate})$$
$$Vlin = number\_of\_visible\_vertical\_lines \ (T_{vgate})$$
$$F_{refr} = refresh\_rate \ (Hz)$$

For example, a standard VGA display with 640*480 visible pixels and a refresh rate of 60Hz requires a bandwidth of BW = 640 * 480 * 60 = 18.4 Mpixels_per_sec (Mpps). A SVGA display with 1024*768 pixels and a 75Hz refresh rate requires 59Mpps. Note that this number also represents the pixel-clock frequency, because only 1 pixels is displayed at a time.

The required host bus bandwidth is dependent on the required number of bits per pixel, as shown in the next formula:

$$BW_{required} = BW_{video} * N_{bits\_per\_pixel} \ (bps)$$

Using the previous examples we can calculate the following table:

Color depth	640*480 @60Hz	1024*768 @75Hz
32bpp	590Mbps	1.9Gbps
24bpp	443Mbps	1.4Gbps
16bpp	295Mbps	944Mbps
8bpp	147Mbps	472Mbps

The host bus occupation is dependent on the total host bus bandwidth, the initial memory latency, the memory access/acknowledge latency, and the programmed video burst length. It can be calculated as follows:

$$O_{bus} = \frac{BW_{required}}{BW_{bus}} * 100\%$$
$$BW_{bus} = host\_bus\_bandwidth \ (Mbps)$$

Or more detailed:

$$O_{bus} = \frac{BW_{required}}{F_{bus} * N_{bus}} * \left(\frac{Mlat_{initial} + VBL * Mlat_{acc}}{VBL} \right) * 100\%$$

$$F_{bus} = host\_bus\_frequency\ (Hz)$$

$$N_{bus} = host\_bus\_width\ (bits)$$

$$Mlat_{initial} = initial\_video\_memory\_latency\ (clk\_cycles)$$

$$Mlat_{acc} = video\_memory\_access\_latency\ (clk\_cycles)$$

$$VBL = Video\_Burst\_Length$$

4.5.3 Examples

4.5.3.1 Example 1

Assume the following system: 200MHz, 32-bit host system using SDRAMS as video memory, running at half the bus frequency, displaying a 1024*768 image @75Hz 24bpp.
Fbus = 200MHz
Nbus = 32-bit
BWrequired = 1.4Gbps
Mlat(initial) = 6 (2* CAS-latency of 3)
Mlat(acc) = 2 (single cycle bursts at half the bus frequency)
Video_burst_length = 4

Total host bus occupation = 77.4%

4.5.3.2 Example 2

Assume a system with an average memory bandwidth of 250MBps displaying an 800*600 image @60Hz 16bpp.
BWrequired = 461Mbps
BWbus = 2Gbps

Total host bus occupation = 23%

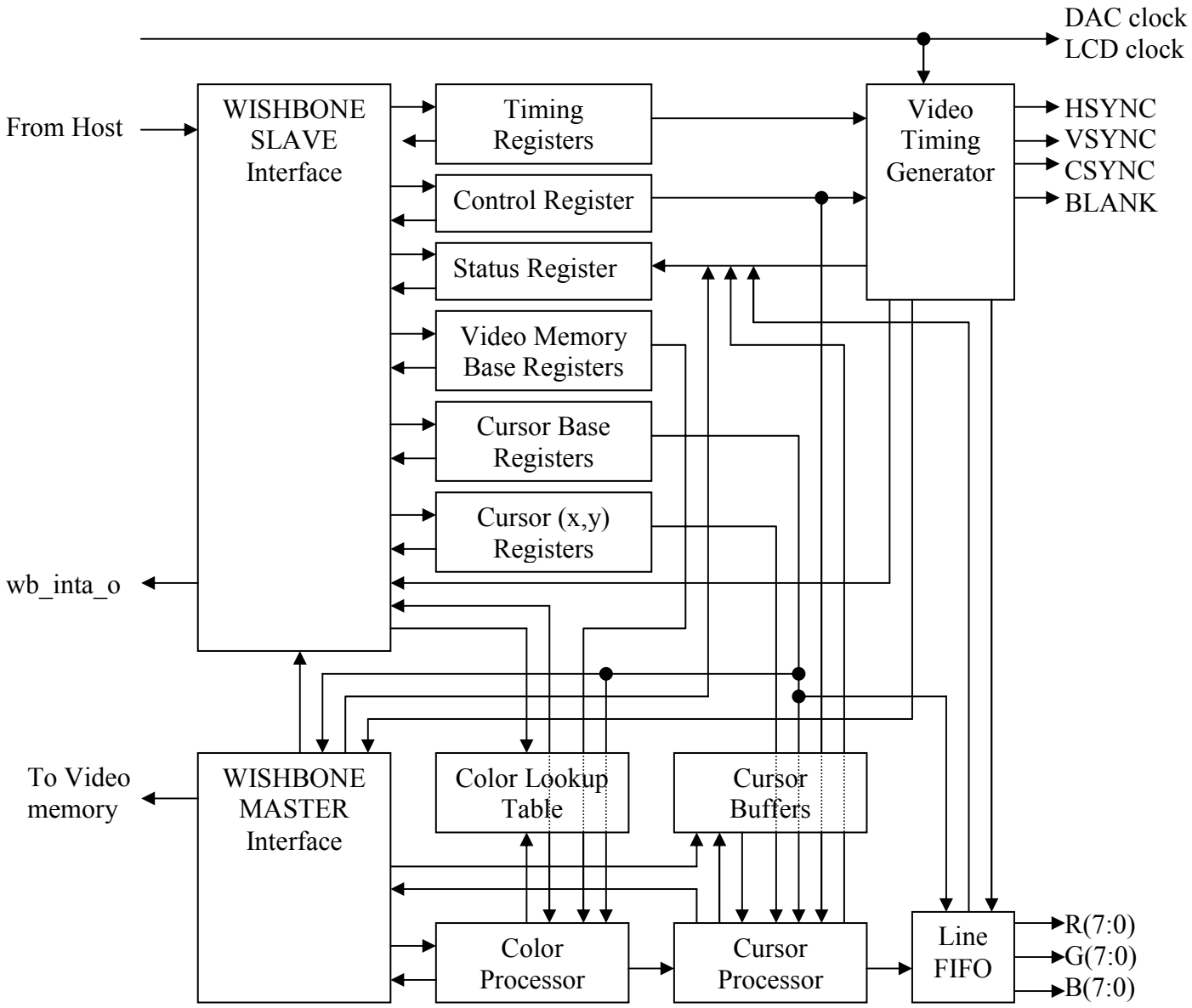
4.5.3.3 Example 3

Assume the following system: 30MHz, 32-bit host system using SRAMS as video memory, displaying a 320*240 image @60Hz 8bpp.
Fbus = 30MHz
Nbus = 32-bit
BWrequired = 37Mbps
Mlat(initial) = 1 (access selector)
Mlat(acc) = 2 (address setup)
Video_burst_length = 8

Total host bus occupation = 8.2%

Note that these numbers are for reading only. The video memory needs to be filled in order to be able to display something. Depending on the application, filling the video memory can require a considerable amount of bandwidth too.

5 Architecture



5.1 Color Lookup Table

The Color Lookup Table (or CLUT) is a 512x24 bit single clock synchronous static random access memory divided into two separate CLUTs of 256x24 bit each. Each color lookup table contains a 24-bit RGB value for each entry. The color processor

uses 8bpp pseudo color data as an address input to the color lookup table. The output from the color lookup table is the RGB data for the current pixel.

5.2 Cursor Base Registers

The Cursor Base registers contain the starting address of the current cursors. Each cursor is 32x32 pixels large. Each pixel is always in 16bpp color mode. Therefore, 512 address locations are required to store a single cursor. A cursor is stored consecutively, starting at pixel (0,0) representing the upper left corner of the cursor, then continuing to pixels (0,1), (0,31), (1,0), and (1,31) etc. A cursor can be located anywhere in memory as long as the memory is accessible by the VGA_LCD core and it starts at a cursor boundary, i.e. the lower 10 address bits must be zero.

5.2 Cursor Buffers

The cursor buffers are 512x32 bit single clock synchronous static memories. Each buffer contains a copy of the current cursor pattern. The core reads the cursor patterns from the external memory and stores them in the cursor buffers, thus avoiding having to read it every frame. The core copies a cursor pattern whenever the Cursor Base Address register is written to. This also opens the possibility to display another cursor than is actually stored in the external memory. Simply rewriting the same address to the Cursor Base Address register is enough to read the new cursor data and display the new cursor.

5.3 Cursor Processor

The cursor processor translates the stored cursor pattern into a visible cursor. It manages the cursor location and determines the pixel information for the current pixel - being image or cursor - including cursor transparency and alpha blending.

5.4 Color Processor

The Color Processor translates the received pixel data to RGB color information. When in 32-bit and 24-bit color mode, this is a pass-through function. In 16-bit color mode this is a linear translation: 5-bit Red, 6-bit Green, and 5-bit Blue. When in 8-bit grayscale mode the same data is placed on the red, green, and blue color outputs, effectively generating a black-and-white image. When in 8-bit pseudo color mode the received pixel data is sent through the internal color lookup table.

5.5 Line FIFO

The dual-clocked Line FIFO ensures a continuous data stream towards the VGA or LCD display and ensures a correct transformation from the WISHBONE clock domain to the VGA clock domain.

5.6 Video Memory Base Registers

The Video Memory Base registers contain the starting addresses of the external video memory banks.

5.7 Video Timing Generator

The Video Timing Generator generates the horizontal synchronization pulse [hsync_pad_o], the vertical synchronization pulse [vsync_pad_o], the corresponding interrupt signals [HINT] and [VINT], the composite synchronization pulse

[csync_pad_o], the blanking signal [blank_pad_o] and the read request to the Line FIFO.

5.8 Wishbone Master Interface

The WISHBONE Master interface manages all accesses to the external memory. It consists of a number of interacting state machines. The color processor and the cursor processor issue requests to the WISHBONE Master. The WISHBONE Master interface then generates the memory addresses for the image and the cursors.

5.9 Wishbone Slave Interface

The WISHBONE Slave interface manages all accesses to user readable/writeable registers.

Appendix A

VGA Modes

This appendix describes some common VGA modes.

A.1 Vertical Timing Information Common VGA Modes

Mode	Resolution	Refresh rate	Line Width	Sync Pulse		Back porch		Active time		Front porch		Frame Total	
			usec	usec	lin	usec	lin	usec	lin	usec	lin	usec	lin
QVGA	320x240	60 Hz											
VGA	640x480	60 Hz	31.78	63	2	953	30	15382	484	285	9	16683	525
VGA	640x480	72 Hz	26.41	79	3	686	26	12782	484	184	7	13735	520
SVGA	800x600	56 Hz	28.44	56	1	568	20	17177	604		-1*	17775	625
SVGA	800x600	60 Hz	26.40	106	4	554	21	15945	604		-1*	16579	628
SVGA	800x600	72 Hz	20.80	125	6	436	21	12563	604	728	35	13853	666

- The Active Time includes 4 overscan borderlines. Some timing tables include these into the back and front porch.
- When the Active Time is increased, it passes the rising edge of the vsync signal, hence the –1 Front Porch.

A.2 Horizontal Timing Information Common VGA Modes

Mode	Resolution	Refresh rate	Pixel Clock	Sync Pulse		Back porch	Active time	Front porch	Line Total
			MHz	usec	pix	pix	pix	pix	pix
QVGA	320x240	60 Hz							
VGA	640x480	60 Hz	25.175	3.81	96	45	646	13	800
VGA	640x480	72 Hz	31.5	1.27	40	125	646	21	832
SVGA	800x600	56 Hz	36	2	72	125	806	21	1024
SVGA	800x600	60 Hz	40	3.2	128	85	806	37	1056
SVGA	800x600	72 Hz	50	2.4	120	61	806	53	1040

- The Active Time includes 6 overscan borderlines. Some timing tables include these into the back and front porch.

Partially taken from Jere Makela, *Software Design for a Video Conversion Equipment. Master's Thesis*, Helsinki Univerity of Technology.

Appendix B

Target Dependent Implementations

The parts of the system that could be target dependent for FPGA implementations and are absolutely target dependent for ASIC implementations are the dual clocked RAM block for the Line FIFO as well as the single clock RAM blocks for the color lookup table and the cursor buffers.

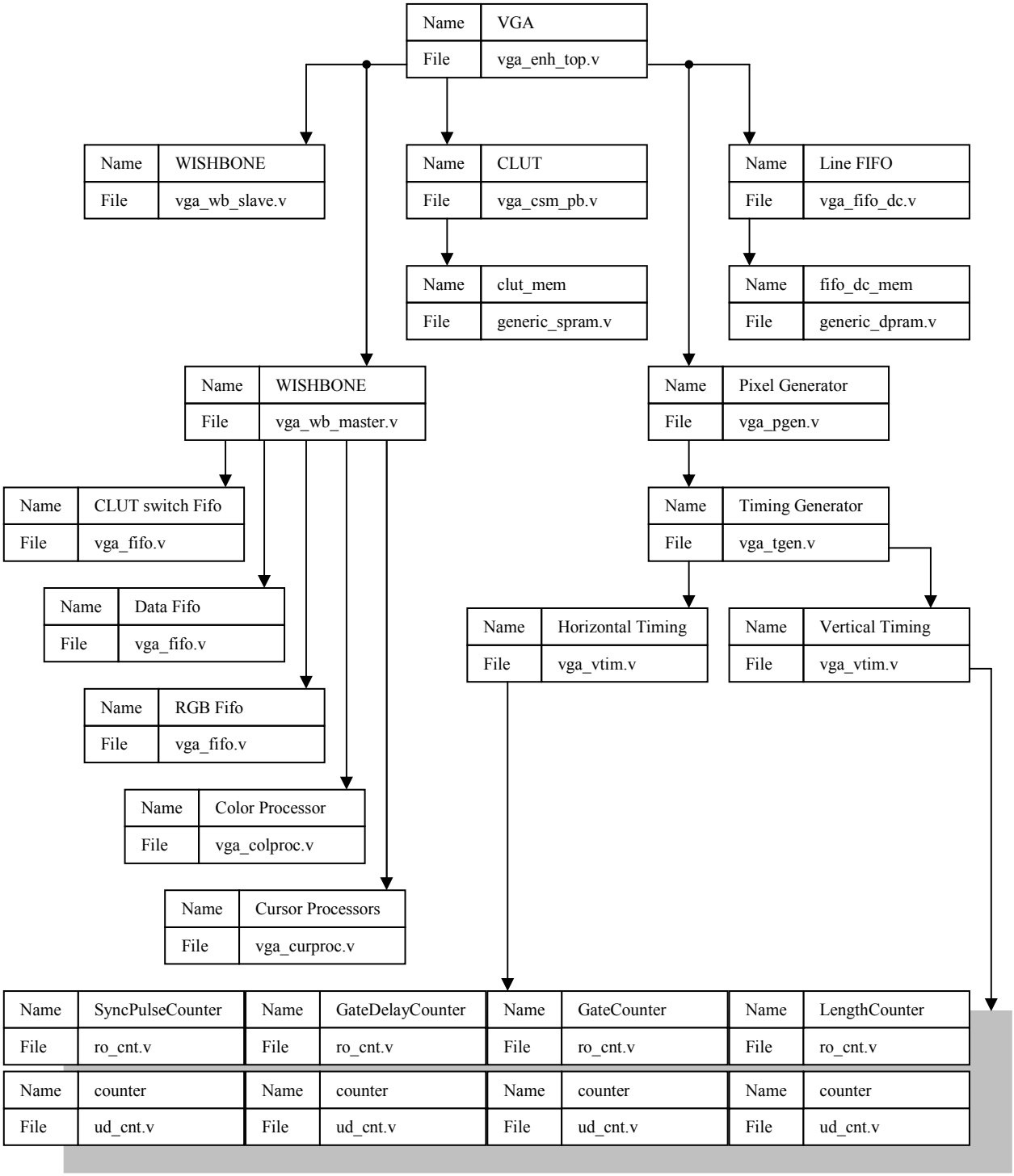
The RAM blocks are instantiated by the generic_spram.v and generic_dpram.v files. These files contain an FPGA-synthesizable model, that has been tested with Exemplar's LeonardoSpectrum and Symplicity's Synplify for Altera (FLEX, ACEX, APEX) and Xilinx devices (Virtex, Virtex-E, Spartan-II). They also contain modules for some ASIC technologies.

The technology is set by a define statement in the vga_defines.v file.
`define VENDOR_FPGA → use FPGA (Xilinx and Altera) synthesizable model
`define VENDOR_ARTISAN → use Artisan memories
`define VENDOR_VIRTUALSILICON → use VirtualSilicon memories
.
.
.

Check the generic_spram.v and generic_dpram.v files for more information.

Appendix C

Core Structure



Appendix D

Design Notes

D.1 Introduction

This section contains flow and timing diagrams of the core's internal blocks. The diagrams are provided for reference only. They are intended to provide a better understanding of the internal signal flow. They are not intended to serve as a detailed step-through discussion of the core's internals.

D.2 vga_curproc

This section shows the signal flow inside the cursor processor blocks. The letters in the data busses are intended to ease the data flow overview. They represent signals that are somehow related to each other and have a common timing spec, for example cbuf_a-A represents address-A into the cursor-buffer, cbuf_q-A is the cursor buffer's output at address-A.