VC2
Specification

Leslie Neft
Silicon Graphics Inc.
Rev. 2.0

nexus:/d/newport/asics/vc2/spec/vc2.spec

1.0 Introduction

VC2 is the Video Controller chip for Newport Graphics. It generates video timing, cursor and display-mode identification codes (DID's).

1.1 Part Name and Number

Part Name: VC2
SGI Part Number: 099-8918-001
Vendor: Toshiba
Vendor Part Number: s0895
Process: TC160G
Base Wafer: TC160G54
Package: 144 PQFP (20 mil pitch)
Gate Count: 27537

1.2 General Description

The main operation of the VC2 chip is the reading and writing of tables in an external RAM. The VC2 interprets these tables to generate cursor data, state information for programmable video timing generation and DID data which is run-length encoded. The format of these tables in the RAM is similar to those used in the VC1. The data is written into the external RAM by the host via the Display Control Bus (DCB).

DID and cursor data are output in a 2-wide pixel stream using a clock which runs at 1/2 the monitor pixel clock rate. The video timing channels are output at the same rate and thus have 2-pixel resolution.

1.3 Features

Video Timing Generator

•21 programmable video output channels

•16x28 state/duration FIFO

•Minimum clock period 14.0 ns (70 MHz)

•genlock

Display ID Generator

•5 bit display ID's

•64x20 DID/duration FIFO

Cursor Generation

•32x32x2 user-definable cursor

•Full screen crosshair cursor

•64x64x1 user definable cursor

Host Interface

•Asynchronous interface to Display Control Bus

External RAM

•64 K bytes table space in external static RAM, organized as 32Kx16, expandable to 64Kx16

1.4 Related Documents

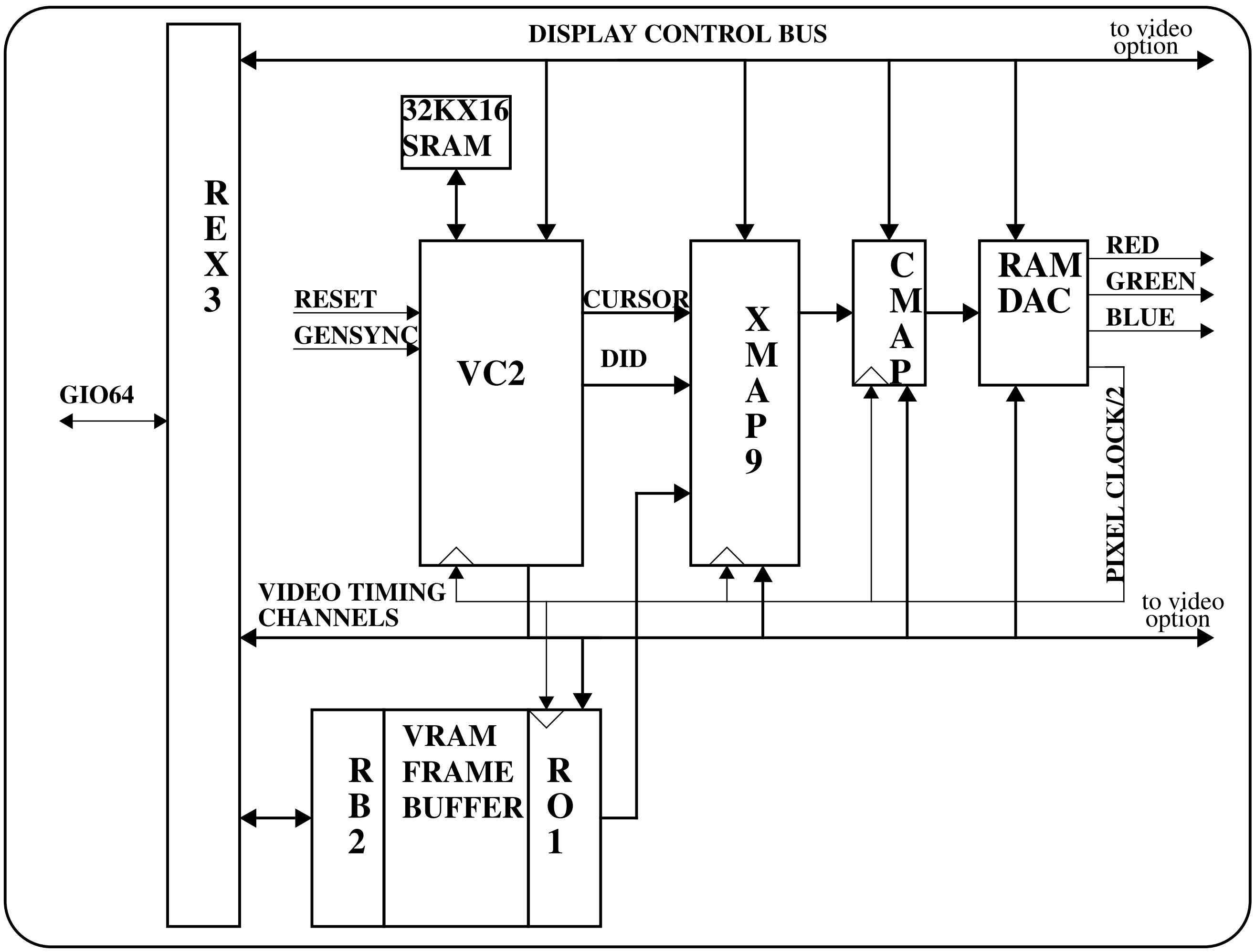
For additional information and background, please refer to the following related documents:

Display Control Bus Specification, Eric Linstadt

VC2 State Tables

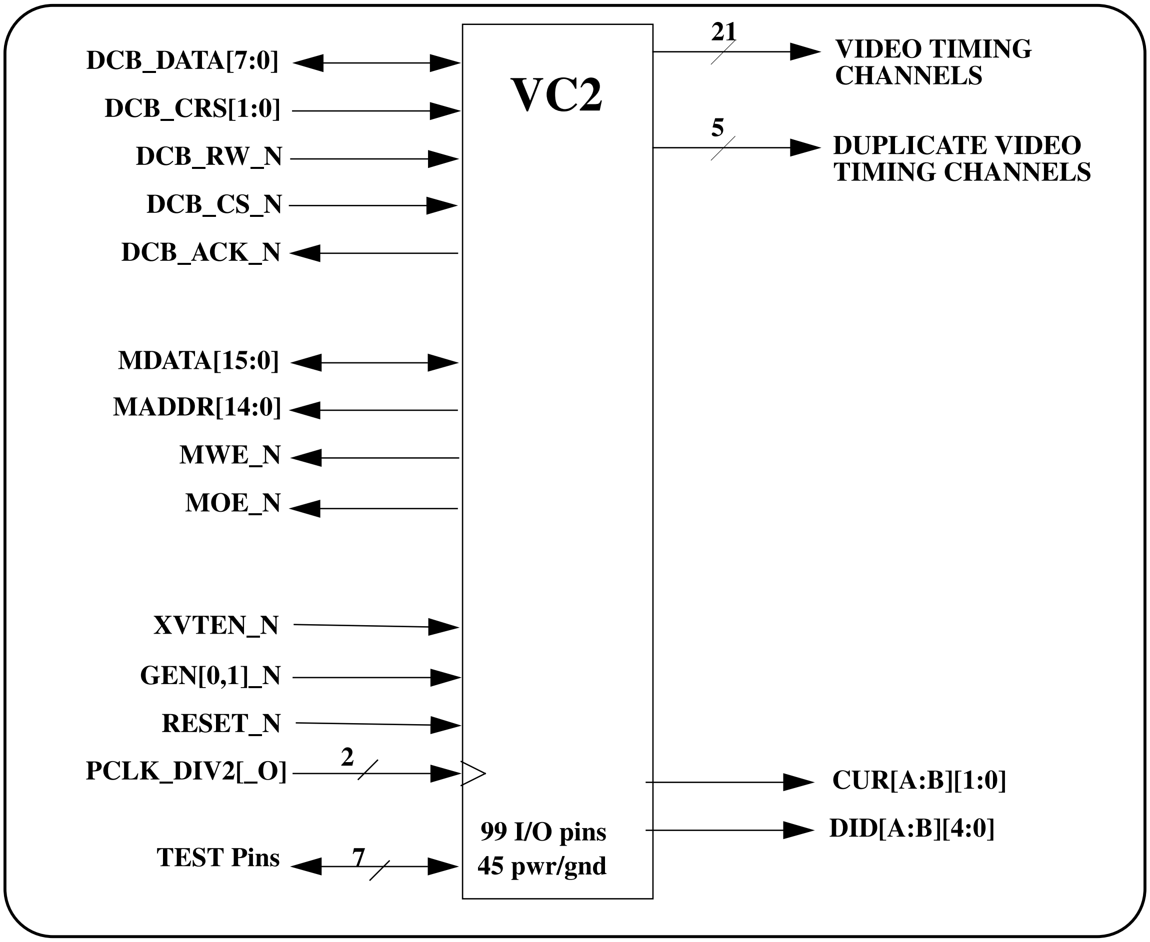
2.0 Device Interface

2.1 System Block Diagram



2.2 Pin Diagram

FIGURE 1. VC2 Pin Diagram



2.3 Pin Descriptions

The VC2 has 99 signal pins plus 45 power/ground pins. The following tables list each VC2 pin, its polarity, direction (I, O, I/O), switching levels (TTL or CMOS) and provide a brief functional description

2.3.1 Display Control Bus Interface.

Pin Name	bits	pol	dir	Type	level	Function
DCB_DATA[7:0]	8	H	I/O	BD16T	TTL	Display Control Bus Data to/from REX3 chip
DCB_CRS[1:0]	2	H	I	TLCHT	TTL	Display Control Bus register select field
DCB_RW_N	1	L	I	TLCHT	TTL	Read/Write direction signal
DCB_CS_N	1	L	I	TLCHTH	TTL	Display Control Bus command strobe indicating that DCB_CRS, DCB_RW_N and, for write transfers, DCB_DATA are valid.
DCB_ACK_N	1	L	O	BT8	TTL	Acknowledge signal for Display Control Bus to handshake transfers with the REX3. When asserted during write cycles, indicates that the slave device has accepted the DCB_DATA. During read cycles, indicates that valid data has been placed on DCB_DATA

2.3.2 External Memory Interface

Pin Name	bits	pol	dir	Type	level	Function
MEM_DAT[15:0]	16	H	I/O	BD8TH	TTL	Data bus to/from external SRAM
MEM_ADR[15:0]	16	H	O	B8H	TTL	Address bus to external SRAM
MEM_OE_N	1	L	O	B8H	TTL	Output enable to external SRAM
MEM_WE_N	1	L	O	B8H	TTL	Write enable to external SRAM

2.3.3 XMAP Interface

Pin Name	bits	pol	dir	Type	level	Function
CURA[1:0]	2	H	O	B8H	CMOS	Cursor for least significant (left-most/even) pixel
CURB[1:0]	2	H	O	B8H	CMOS	Cursor for most significant (right-most/odd) pixel
DIDA[4:0]	5	H	O	B8H	CMOS	Window display ID for least significant (left-most/even) pixel
DIDB[4:0]	5	H	O	B8H	CMOS	Window display ID for most significant (right-most/odd) pixel

2.3.4 Video Timing Generation

Pin Name	bits	pol	dir	Type	level	Function
TX_REQ_REX_N	1	L	O	B8H		To REX3: transfer request for VRAM transfer cycle.
SET_TSC_REX_N	1	L	O	B8H		To REX3: set scan counter.
VERT_INT_REX_N	1	L	O	B8H		To REX3: vertical interrupt
VERT_STAT_GIO_N	1	L	O	B8H		To Host: vertical status
CBLANK_DAC_N	1	L	O	B8H		To RAMDAC: composite blanking
CSYNC_DAC_N	1	L	O	B8H		To RAMDAC: composite sync
CSYNC_ARC_N	1	L	O	B8H		To monitor: composite sync
HSYNC_ARC_N	1	L	O	B8H		To monitor: horizontal sync

Pin Name	bits	pol	dir	Type	level	Function
VSYNC_ARC_N	1	L	O	B8H		To monitor: vertical sync
SER_EN_RO_N[2:0]	3	L	O	B8H		To RO1: serial enable to enable clocking of VRAM data
DSPLY_EN_RO_N[2:0]	3	L	O	B8H		To RO1: display enable drive pixel data out
CBLANK_CMAP_N	1	L	O	B8H		To CMAP: composite blanking, loads look-up tables
EOF_AB_N	1	L	O	B8H		To AB1: end of frame
HBLANK_AB_N	1	L	O	B8H		To AB1, XMAP: horizontal blanking
CBLANK_XMAP_N	1	L	O	B8H		To XMAP: composite blanking
SPARE	1	L	O	B8H		spare timing channel
VPOS_VC_N	1	L	O	B8H		internal: vertical position for cursor
EOF_VC_N	1	L	O	B8H		internal: end of field/frame for DID
HPOS_VC_N	1	L	O	B8H		internal: horizontal position for cursor
VIS_LN_VC_N	1	L	O	B8H		internal:visible portion of line for DID
ODDFIELD_VC_N	1	L	O	B8H		monitor, internal: oddfield in interlaced mode, or left/right eye in stereo mode

For details on the Video Timing Channels see Programmer's Interface/ Table Formats/ Video Timing Table.

2.3.5 Miscellaneous

Pin Name	bits	pol	dir	Type	level	Function
GEN0_N	1	L	I	TLCHNH	TTL	Genlock input used to synchronize frame
GEN1_N	1	L	I	TLCHNH	TTL	Genlock input used to synchronize frame
RESET_N	1	L	I	TLCHT	TTL	System reset
XVTEN_N	1	L	I	TLCHT	TTL	External video timing enable (for Nextgen)

2.3.6 Clocks

Pin Name	bits	pol	dir	Type	level	Function
PCLK_DIV2	1	H	I	TLCHNH 2xIDRV16	TTL	Pixel clock divided by 2. Used to drive the internal array.
PCLK_DIV2_O	1	H	I	TLCHNH IDRV4	TTL	Pixel clock divided by 2. Used to drive the output registers.

2.3.7 ASIC Mandatory Test Pins

Pin Name	bits	pol	dir	Type	level	Function
JTMS	1		I	TLCHTU		Required test pin
JTCK	1		I	TLCHTU		Required test pin
JTDI	1		I	TLCHTU		Required test pin
JTDO	1		O	B4		Required test pin
TP0	1		I	TLCHTD		Required test pin
TP1	1		I	TLCHTD		Required test pin
ENTEI	1		I	TLCHTHD		Required test pin. Connect to power on reset on board.

2.3.8 Power and Ground

Pin Name	bits	Level	dir	Type	Drive	Function
VSS/VDD	4 5					

2.4 Package Pin Assignment

Table 1: VC2 Pin Assignments

pin #	signal name	pin #	signal name	pin #	signal name	pin #	signal name
1	VSS2	37	P_MEM_ADR_15	73	VSS1	109	P_VT_18 TX_REQ_REX_N
2	P_VT_0 VERT_INT-REX_N	38	P_MEM_ADR_14	74	P_MEM_DAT_8	110	P_VT_17 SER_EN_EO_N
3	P_DCB_ACK_N	39	P_MEM_ADR_13	75	P_MEM_DAT_7	111	VSS1
4	P_DCB_DATA_7	40	VSS1	76	P_MEM_DAT_6	112	P_SEREN_RO_A
5	P_DCB_DATA_6	41	P_MEM_ADR_12	77	P_MEM_DAT_5	113	P_SEREN_RO_B
6	VSS1	42	P_MEM_ADR_11	78	VSS1	114	P_DSPEN_RO_A
7	VDD	43	P_MEM_ADR_10	79	VDD1	115	VSS1
8	P_DCB_DATA_5	44	VSS1	80	P_MEM_DAT_4	116	VSS2
9	P_DCB_DATA_4	45	VDD1	81	P_MEM_DAT_3	117	P_DSPEN_RO_B
10	P_DCB_DATA_3	46	P_MEM_ADR_9	82	P_MEM_DAT_2	118	P_VT_16 DSPLY_EN_RO_N
11	VSS1	47	P_MEM_ADR_8	83	P_MEM_DAT_1	119	P_VT_15 HPOS_VC_N
12	P_DCB_DATA_2	48	P_MEM_ADR_7	84	VSS1	120	VSS1
13	P_DCB_DATA_1	49	P_MEM_ADR_6	85	P_MEM_DAT_0	121	VDD1
14	P_DCB_DATA_0	50	VSS1	86	P_DIDB_4	122	P_PCLK_DIV2O
15	VSS1	51	P_MEM_ADR_5	87	P_DIDB_3	123	VDD3
16	P_DCB_CRS_1	52	P_MEM_ADR_4	88	P_DIDB_2	124	P_PCLK_DIV2
17	P_DCB_CRS_0	53	P_MEM_ADR_3	89	VSS1	125	VDD1
18	VDD3	54	VDD1	90	VDD1	126	VSS1
19	P_DCB_RW_N	55	VSS1	91	P_DIDB_1	127	P_VT_14 VIS_LN_VC_N
20	P_DCB_CS_N	56	P_MEM_ADR_2	92	P_DIDB_0	128	P_VT_13 VPOS_VC_N
21	P_XVTEN_N	57	P_MEM_ADR_1	93	P_CURB_1	129	P_VT_12 EOF_VC_N
22	P_RESET_N	58	P_MEM_ADR_0	94	P_CURB_0	130	P_VT_11 ODDFIELD_N
23	P_GEN1_N	59	VSS1	95	VSS1	131	VDD1
24	P_GEN0_N	60	P_MEM_WE_N	96	VDD1	132	P_VT_10 SET_TSC_REX_N
25	VDD1	61	VDD1	97	P_DIDA_4	133	P_VT_9 CBLANK_CMAP_N
26	VSS3	62	P_MEM_OE_N	98	P_DIDA_3	134	P_VT_8 EOF_AB_N
27	P_TP1	63	P_MEM_DAT_15	99	P_DIDA_2	135	P_VT_7 HBLANK_AB_N
28	P_TP0	64	VSS1	100	P_DIDA_1	136	VSS3
29	P_JTMS	65	P_MEM_DAT_14	101	VSS1	137	VSS1
30	P_ENTEI	66	P_MEM_DAT_13	102	VDD1	138	P_VT_6 CBLANK_XMAP_N
31	P_JTCK	67	P_MEM_DAT_12	103	P_DIDA_0	139	P_VT_5 spare
32	P_JTDI	68	VSS1	104	P_CURA_1	140	P_VT_4 VERT_STAT_GIO_N
33	VDD1	69	VDD1	105	P_CURA_0	141	VSS1
34	VSS1	70	P_MEM_DAT_11	106	P_VT_20 CBLANK_DAC_N	142	P_VT_3 CSYNC_ARC_N
35	P_JTDO	71	P_MEM_DAT_10	107	VSS1	143	P_VT_2 HSYNC_ARC_N
36	VSS2	72	P_MEM_DAT_9	108	P_VT_19 CSYNC_DAC_N	144	P_VT_1 VSYNC_ARC_N

2.5 Timing Specifications

Table 2: AC Timing Specifications

from	to	edge	load	min	max
PCLK_DIV2_O	VT(20:0) CUR[A,B](1:0) DID[A,B](4:0) SEREN_RO_[A,B] DSPEN_RO_[A,B]		35 pf	2.0 ns	11.12 ns
PCLK_DIV2	MEM_ADDR(15:0)		20 pf	2.7 ns	13.4 ns
PCLK_DIV2	MEM_WE_N	lh hl	20 pf	4.3 ns 5.0 ns	17.2 ns 19.8 ns
PCLK_DIV2	MEM_OE_N	lh hl	20 pf	2.9 ns 3.3 ns	11.5 ns 13.2 ns
PCLK_DIV2	MEM_DATA(15:0)	zh/zl hz/lz	20 pf	7.8 ns 7.9 ns	33.5 ns 33.5 ns

pin	relative to	parameter	best case	worst case
MEM_DATA(15:0)	PCLK_DIV2	setup	-0.9 ns	-3.6 ns
MEM_DATA(15:0)	PCLK_DIV2	hold	1.4 ns	5.5 ns

2.6 Synchronizing VC2 (for Nextgen)

VC2 has some hooks in order to allow 2 VC2 chips to generate identical video timing. This feature will most likely be used in the Nextgen board. First, the 2 chips must be synchronized, that is, the dual clock portion of the chips must be in phase. This is done by synchronizing the system reset signal which drives the ENTEI pin. Secondly, the video timing tables must be set up and video timing enabled before XVTEN_N is asserted. Whenever the video timing is disabled, XVTEN_N must first be deasserted then re-asserted after video timing is re-enabled.

For Newport, or other implementations which use a single VC2, ENTEI should be connected to the system reset signal, which need not be synchronized, and XVTEN_N should be tied low.

3.0 Programmer Interface

3.1 Registers

The table below shows all registers in the VC2 which are accessible to the host. Only a subset of these registers (indicated by shading) will be programmed by the host under normal operations. The others are internally maintained registers which have been made available for diagnostic purposes, but must never be written under normal operation. All writable registers will be readable, also for diagnostics purposes. Note that some of these registers will have side effects in the chip when written. The programmer take into account the function of each register when writing diagnostic code.

REGISTER SUMMARY

CRS	Index	Prog access	Diag access	Name	bits	Description
0	--	R/W	R/W	Register File Index	5	Register file address for host reads and writes
1,2	0x00	R/W	R/W	Video Entry Pointer	16	Pointer to start of video timing table in external RAM
1,2	0x01	R/W	R/W	Cursor Entry Pointer	16	Pointer to start of cursor data in external RAM
1,2	0x02	R/W	R/W	Cursor X Location	16	Next valid cursor X location (max value = 4095)
1,2	0x03	R/W	R/W	Cursor Y location	16	Next valid cursor Y location (max value = 2047)
1,2	0x04	R/--	R/W	Current Cursor X	16	Current cursor X location (after update)
1,2	0x05	R/W	R/W	DID Entry Pointer	16	Pointer to start of Display ID table in external RAM
1,2	0x06	R/W	R/W	Scanline Length	16	Total number of visible pixels in scanline (for DID processing, max value=2047)
1,2	0x07	R/W	R/W	RAM Address	16	Address used for host access to external RAM
1,2	0x08	--/--	R/W	VT Frame Table Pointer	16	Running pointer to current location in video timing frame table
1,2	0x09	--/--	R/W	VT Line Sequence Ptr	16	Running pointer to current location in line sequence table
1,2	0x0A	--/--	R/W	VT Lines in Run	16	Count of lines remaining in current state run
1,2	0x0B	R/--	R/W	Vertical Line Counter	16	Current line count including blanked lines
1,2	0x0C	--/--	R/W	Cursor Table Pointer	16	Running pointer to next byte of cursor pixel data
1,2	0x0D	--/--	R/W	Working Cursor Y	16	Value of cursor Y location for current frame
1,2	0x0E	--/--	R/W	DID Frame Table Ptr	16	Running pointer to current line in DID frame table
1,2	0x0F	--/--	R/W	DID Line Table Pointer	16	Running pointer to current location in DID line table
1,2	0x10	R/W	R/W	Display Control	16	Video display and cursor control (see detail following)
1,2	0X1F	R/W	R/W	Configuration	16	Clock configuration, chip reset, revision code
3	--	R/W	R/W	RAM Data	16	External RAM data (virtual register)

Note: CRS 1 addresses the high byte; CRS2 addresses the low byte. See section entitled Programming via the Display Control Bus.

Although all registers (excepth the Index Register) are 16 bits, the allowable range of values for a register may be less (as noted). External Ram Address pointers may have a maximum value of 32767 (15 bits). In general, the programmable registers (that is, those which are writable by the programmer) may be written at any time. The VC2 will ensure a smooth transition.

3.1.1 Index Register

This Index Register holds the pointer to the VC2 register space in order to indicate which register is read or written during DCB accesses to the VC2. The index register is required since the register address space of the VC2 exceeds the register address space of the DCB. See section “Programming via the Display Control Bus” below**.**

bit	Name	Description
4:0	Index	Index pointer to VC2 registers
7:5	reserved	

3.1.2 Configuration Register

bit	Name	Description
0	Soft Reset	0= reset VC2 1= normal operation This bit is reset to “0” during system reset. In order to enable the operation of the VC2, the programmer must write a “1” into this bit. The soft reset will reset all portions of the chip except for the DCB interface. The chip must be held in the reset state whenever the pixel clock is changed.
1	Slow Clock	0= fast mode, monitor pixel rate >= 70 Mhz 1= slow mode, monitor pixel rate < 70 Mhz This bit allows VC2 operations to execute more efficiently for slower speed monitors. It should be set, when required, while the VC2 is in reset mode. This bit is reset to “0” by either the system or soft reset.
2	Cursor Error Flag	0 = no error 1 = Cursor error has occured This bit is set by the hardware if HPOS (indicating the start of the cursor for the current line) is asserted before the required cursor processing for that line has completed. The programmer can clear the flag by writing a “0”.
3	DID Error Flag	0 = no error 1 = DID fifo underflow has occured This bit is set by the hardware if the DID generation logic has attempted to read from an empty fifo. This indicates that the quantity and spacing of the transitions in the DID table are such that the hardware could not process the DID table fast enough. The programmer can clear the flag by writing a “0”.
4	VTG Error Flag	0 = no error 1 = VTG fifo underflow has occured This bit is set by the hardware if the video timing generation logic has attempted to read from an empty fifo. This indicates that the hardware was not able to process the video timing table fast enough to keep up with required video timing channel requirements. The programmer can clear the flag by writing a “0”.
7:5	Revision code	Revision code of VC2 chip. 000 = first revision of VC2
15:8	Temp Hibyte	Read only. This byte contains the data from the most recent DCB read or write from/to CRS 1 (for registers 0x00 to 0x10 only). It can be used the retrieve the high byte of a write operation if the operation is not completed.

NOTE: The configuration and index registers can be written or read after power up. However, **before accessing any other register or the external SRAM, the soft reset bit in the configuration register must be set to a ‘1’ for normal operation.** Otherwise the VC2 will not respond, and both the VC2 and the Display Control Bus can hang.

3.1.3 Display Control Register

Display Control Register Detail

bit	Name	Description
0	VINTR enable	0= disable vertical interrupt 1= enable vertical interrupt A vertical interrupt will occur when the vertical interrupt timing channel is active and the interrupt is enabled.
1	Blackout	0= blackout display 1= enable display When blackout is indicated, the video timing channel, CBLANK (to the Ramdac only)is forced to the active state.
2	Video Timing Enable	0= disable video timing function 1= enable video timing function When enabled, the VC2 will get video timing from external RAM and drive the video timing channels. When disabled, the video timing fifo will be reset and the video timing channels will be driven to their default state.
3	DID Function Enable	0= disable DID function 1= enable DID function When enabled, the VC2 will read DID’s from the external RAM. When disabled, the DID fifo will be reset and DID value of 0 will be driven.
4	Cursor Function Enable	0= disable cursor function 1= enable cursor function When enabled, the VC2 will read cursor data from the external RAM. When disabled, undefined cursor values will be driven.
5	Gensync Enable	0= disable Gensync input 1= enable Gensync input When enabled and the GENSYNC input to the chip is asserted, the video timing control will be reset to the start of frame. The video timing fifo will not be flushed, thus there will be latency between the assertion of Gensync and the start of frame.
6	Interlace	0= non-interlaced monitor 1= interlaced monitor Must be programmed by the host to indicate the monitor type.
7	Cursor Enable	0= disable cursor 1= enable cursor When enabled, valid cursor data is driven. When disabled all 0’s are driven on the cursor lines. When the cursor is enabled or disabled, the affect is seen immediately, not delayed until after the next vertical retrace.
8	Cursor Mode	0= glyph cursor (32x32x2) 1= crosshair cursor
9	Cursor Size	0=32x32x2 1=64x64x1 This bit is used only when glyph cursor mode is selected.
10	Genlock Select	0=Gen0 1=Gen1 This bit is used only when genlock is enabled. It selects which genlock input to the chip is used.
15:11	reserved	

Note: at power-on all bits in the Display Control Register will default to ‘0’.

This register must be used cautiously during diagnostics. Before any of the chip functions are enabled, the appropriate registers and memory tables must be set up. Otherwise, the VC2 could get into an unknown state, not only hanging itself, but possibly the Display Control Bus as well.

3.1.4 Register File Registers

While the Index, Configuration and Display Control Register (DCR) are implemented as discrete registers, the following registers reside in the register file, implemented with a 16x16 internal ram.

3.1.4.1 Video Entry Pointer

This register contains the address of the start of the Video Timing Frame Table in external Ram. This register must contain a valid address pointing to a valid video timing table before the video timing function is enabled by setting the appropriate bit in the DCR. Otherwise, the VC2 will attempt to interpret invalid data in the ram as video timing, resulting in erroneous operation, which may include hanging the DCB.

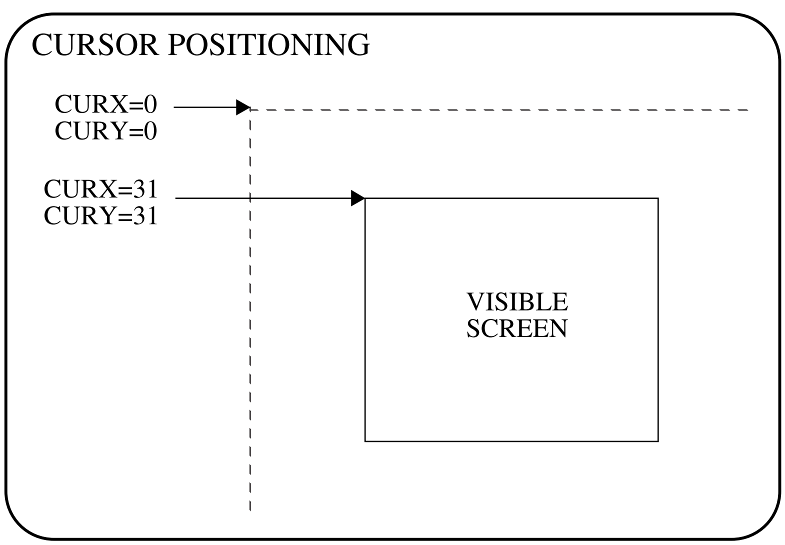
3.1.5 Cursor Entry Pointer

This register contains the address of the start of the cursor glyph data in external Ram. This register may be written at any time; the VC2 will ensure that in each frame, only one cursor pattern is displayed. The VC2 maintains a seperate copy of the cursor table pointer which will be updated with the Cursor Entry Pointer during the vertical blanking period, never while the cursor is being displayed. However, if the programmer is changing the Cursor Entry Pointer or the Cursor pattern table in memory coincident with the X/Y location or the cursor mode, then the cursor should be disabled while the changes are made to ensure a smooth transition between the previous X/Y/pattern and the new X/Y/pattern.

The Cursor Entry Pointer must be on a 128-word boundary for the 32x32x2 glyph and a 256-word boundary for the 64x64x1 glyph.

3.1.6 Cursor X Location, Cursor Y Location and Current Cursor X

For crosshair cursor mode, the X and Y cursor locations specify the center of the crosshair. For cursor glyph, X and Y locations specify the upper lefthand corner of the glyph. In order to support off-screen cursor, the X and Y locations are relative to a point which is 31 pixels to the top and left of the visible portion of the screen. Thus, if the cursor glyph were positioned at X=0 and Y=0, it would be entirely off-screen except for one pixel (the lower right corner of the glyph), which would appear in the upper left corner of the screen. A cursor glyph positioned at X=31 and Y=31 would be entirely visible in the upper left corner of the screen. The cursor can also be moved off-screen to the right and bottom of the screen. The diagram below shows this conceptually.



The actual cursor positioning is determined via programmable video timing channels, and thus could be modified by the programmer. The X origin, however, can only be positioned to be an odd number of pixels from the upper left-hand corner of the screen.

In order for the cursor location to change in a smooth fashion, the X and Y locations used by the cursor control logic will always change coincidentally. This is accomplished by having a shadow register for the X location. To change the cursor position, the programmer must write the Cursor X Location register first, followed by the Cursor Y Location register. When the Cursor Y location is written, the VC2 will copy the value from the Cursor X Location to the Current Cursor X register. Cursor Y Location and Current Cursor X will then define the cursor position. The Cursor location registers are managed in a similar fashion to the Cursor Entry Pointer, such that there will be only one cursor displayed on the screen per frame. The VC2 maintains a seperate copy of the curosr position registers which will be updated with the Current Cursor X Location and the Cursor Y Location during the vertical blanking period, never while the cursor is being displayed. Thus, when making a change to just the cursor location, the programmer need not disable the cursor. (See previous subsection).

Note that the maximum allowable Cursor X Location is 2047.

3.1.7 DID Entry Pointer

This register contains the address of the start of the Display ID Table in external Ram. This register may be written at any time; the VC2 will not use the new pointer until after the next (or current) vertical retrace period.

3.1.8 Scanline Length

bit	Name	Description
4:0	reserved	
15:5	Scanline Length	Must be set by the programmer to the total number of pixels in the visible portion of the scanline, for use in DID generation. The maximum allowable scanline is 2046 pixels.

3.1.9 RAM Address

This register contains the address pointer for all host reads and writes to the external Ram. Whenever the host reads or writes the external Ram, this register will be automatically incremented.

3.1.10 VT Frame Table Pointer, VT Line Sequence Pointer, VT Lines in Run

These are running pointers and counters used for reading the video timing table from external Ram. They convey no useful information to the programmer other than for diagnostics. The VT Frame Table Pointer and VT Line Sequence Pointers point to the next location to be used in the video timing frame table and line sequence table, respectively. VT Lines in run is a count of the number of lines remaining in the current line sequence run.

3.1.10.1 Vertical Line Counter

This is a running counter of lines in the current frame. The counter is consistent with the Cursor Y position and, thus, starts 31 lines before the visible portion of the frame. The programmer may read this register to obtain vertical position information.

3.1.11 Cursor Table Pointer and Working Cursor Y

These registers are for cursor generation. They convey no useful information to the programmer other than for diagnostics. The Cursor Table Pointer is a running pointer to the next location to be used in the cursor pattern table and also keeps track of the vertical position within the cursor glyph. Working Cursor Y is updated during vertical blanking with the Cursor Y Location and incremented during lines which contain the glyph cursor. It is used to determine if the cursor (either glyph or crosshair) appears on the current line.

3.1.12 DID Frame Table Pointer and DID Line Table Pointer

These are running pointers used for reading the display ID table from external Ram. They convey no useful information to the programmer other than for diagnostics. The DID Frame Table Pointer and DID Line Table Pointer point to the next location to be used in the DID frame table and line table, respectively.

3.2 Resets, Start-up and Changing Video Timing

The VC2 has 2 resets - System Reset and Soft Reset. System Reset is basically a power-on reset and will reset the entire VC2. The Soft Reset is a bit in the VC2 Configuration register and thus under control of the programmer. The Soft Reset does not affect the DCB interface or any of the programmable registers.

After power-up, the System reset will go away, leaving the soft reset in affect. (The system reset resets the Soft reset). The programmer would then perform the following procedure:

- •1• Program the Ramdac for the desired pixel clock rate.
- •2• Set the Slow Clock bit (if required) in the Configuration register and release the Soft Reset (may be done coincidentally). The chip is now ready for operation.
- •3• Load the Video Timing tables into external Ram and write the Video Entry Pointer.
- •4• Enable video timing with the following operations by setting the Video Timing Enable bit in the Display and Cursor control Register. The interlace bit should be set to the appropriate value at this time. Approximately 32 pixel clocks later, the VC2 will begin generating video. At this point the programmer may set the VINTR enable, the Blackout (display enable) and the Gensync enable, or these operations can be done at a later time.
- •5• Write the DID tables to external Ram and write the following registers: DID Entry Pointer, Scanline Length. (This may have all ready been done).
- •6• Enable DID function by setting the bit in the Display and Cursor Control Register.
- •7• Set the Cursor Function Enable bit in the Display and Cursor Control Register. [The cursor will initially come up in glyph mode and disabled.] After enabling this function, the programmer can display the cursor by writing the cursor pattern into memory and setting the appropriate bits and registers. This step may be executed before step #5.

Note that during normal operation, the Video Timing Enable, DID Function Enable and Cursor Function Enable will always be set to enabled.

To change the display monitor and pixel clock, the programmer should use the following procedure:

- •1• Disable Cursor Function enable, DID function enable, Video timing enable, VINTR enable, Gensync enable and Blackout display. (May be done coincidentally).
- •2• Perform a Soft Reset.
- •3• Perform steps 1, 2,3 and 4 under start-up procedure.
- •4• The DID Function and Cursor Function can now be enabled. The DID tables, Cursor pattern table and associated registers and control bits will retain their previous values.

Note: It may be necessary for the programmer to ensure a minimum delay time between the setting and clearing of Soft Reset and the changing of the monitor clock.

3.2.1 Power-up States

The VC2 will always power up with the video disabled and the monitor blacked out. Otherwise the programmer should not assume the state of any registers or control bits (other than the soft reset) and should program them to the desired values before enabling the corresponding functions.

3.3 Programming via the Display Control Bus

For DCB accesses the VC2 uses 4 Control Register Selects (CRS's): one for the Index register, one for external Ram data and one for each byte of the remaining group of registers. The mechanisms for accessing registers and external Ram data are different in order to optimize each. For registers, the host will generally write to 1 or 2 registers at a time, each requiring a different index value. By making use of the auto-increment feature in the REX3 DCB interface, the writing of the index and the register can be combined into one GIO operation. That is, the first byte of a 32- or 64-bit word will go to the index register (CRS=0) to indicate the register to which the next 2 bytes (CRS=1,2) will be written. For Ram data, the host will generally write a large number of bytes to contiguous addresses. In this case, the host would first write the starting address into the Ram Address register. Then, (without the DCB auto-increment) data is written to the Rams (CRS=3). After each 2 bytes is written, the Ram Address will be automatically incremented, so that any (even) number of bytes can be written to the Ram without re-writing the DCB mode register.

To read or write external Ram data (CRS location 3) 2 bytes must always be accessed atomically in big-endian fashion, that is the most significant byte first followed by the least significant byte. For writes, the VC2 will save the first byte transfered in a temporary register. When the second byte is written, the 16-bit word will be written into the external Ram at the address indicated by the Ram address register. For reads, the VC2 will read the external Ram at the address indicated by the Ram address register. The first byte is transfered onto the bus, while the second byte is held in a temporary register. When the next DCB read occurs, the second byte is transferred onto the bus from the temporary register. Following a 2-byte read or write, the Ram address register will be incremented. Since the programmer must always access memory with 2-byte atomic operations, the hardware and software will never get "out of sync" as to whether the high or low byte of memory is being accessed. However, for diagnostic purposes, a write to the Ram Address Register will reset the High/Low byte pointer so that the next byte accesed is always the high byte.

The programmer may read or write the Index register using a single byte access, although it is expected that this operation would be combined with the read or write of the desired register by taking advantage of the DCB auto increment mode provided by the REX3. For writes, the programmer writes 3 bytes of a 32-bit word to start at CRS location 0 with the auto increment feature turned on. The REX3 DCB interface will then write 3 bytes to the VC2 at CRS locations 0, 1 and 2. The first byte is written into the Index register, the second byte is saved in a temporary register. When the third byte is written, the target register (as indicated by the Index register) is updated with the 16-bit value. For reads, the programmer will turn on the auto-increment feature and start at CRS location 0. The programmer writes one byte for the index register and then reads 2 bytes. The REX3 DCB interface will write the first byte into CRS location 0, and then read 2 bytes from location 1 and 2. The write to CRS 0 goes to the Index register. When CRS 1 is read, the target register (as indicated by the Index register) is read and the high byte is driven to the DCB data bus, while the low byte is saved in a temporary register. When CRS 2 is read the data from the temporary register is driven onto the data bus. Note that the VC2 assumes that reads and writes for CRS locations 1 (most significant byte) and 2 (least significant byte) are always done atomically in that order (big-endian). Any deviation from this will cause erroneous data to be transfered.

For the programmer to access the VC2 on the DCB, first the DCBMODE register in the REX3 must be set up. Thus a write to any VC2 register requires 2 GIO operations: 1 write to DCBMODE, 1 write to DCBDATA. A read from a VC2 register (except Index) requires 3 GIO operations: 1 write to DCBMODE, 1 write to DCBDATA (for index register), 1 read from DCBDATA. To read or write external Ram data requires 1 write to DCBMODE followed by 1 or more reads or writes to DCBDATA. The programmer must ensure that these operations meet the requirements for atomicity described above.

3.3.1 Display Control Bus Mode Parameters

The following list shows the required DCB Mode parameters for accessing the VC2 over the DCB.

DCBADDR = as required by system

DCBCRS = 0, 1 or 3 as required by type of access (see previous section “Programming via the Display Control Bus”)

CS_SETUP = 0 (board implementation may require a different parameter; Newport board will probably require 1)

CS_WIDTH = 0

CS_HOLD = 0

ENABACK= 1 (enable acknowledge handshake)

SYNCACK = 0 (asynchronous acknowledge)

3.4 Table Formats

3.4.1 Video Timing Table

The Video Timing table has two parts - a frame table and a line sequence table as shown below.

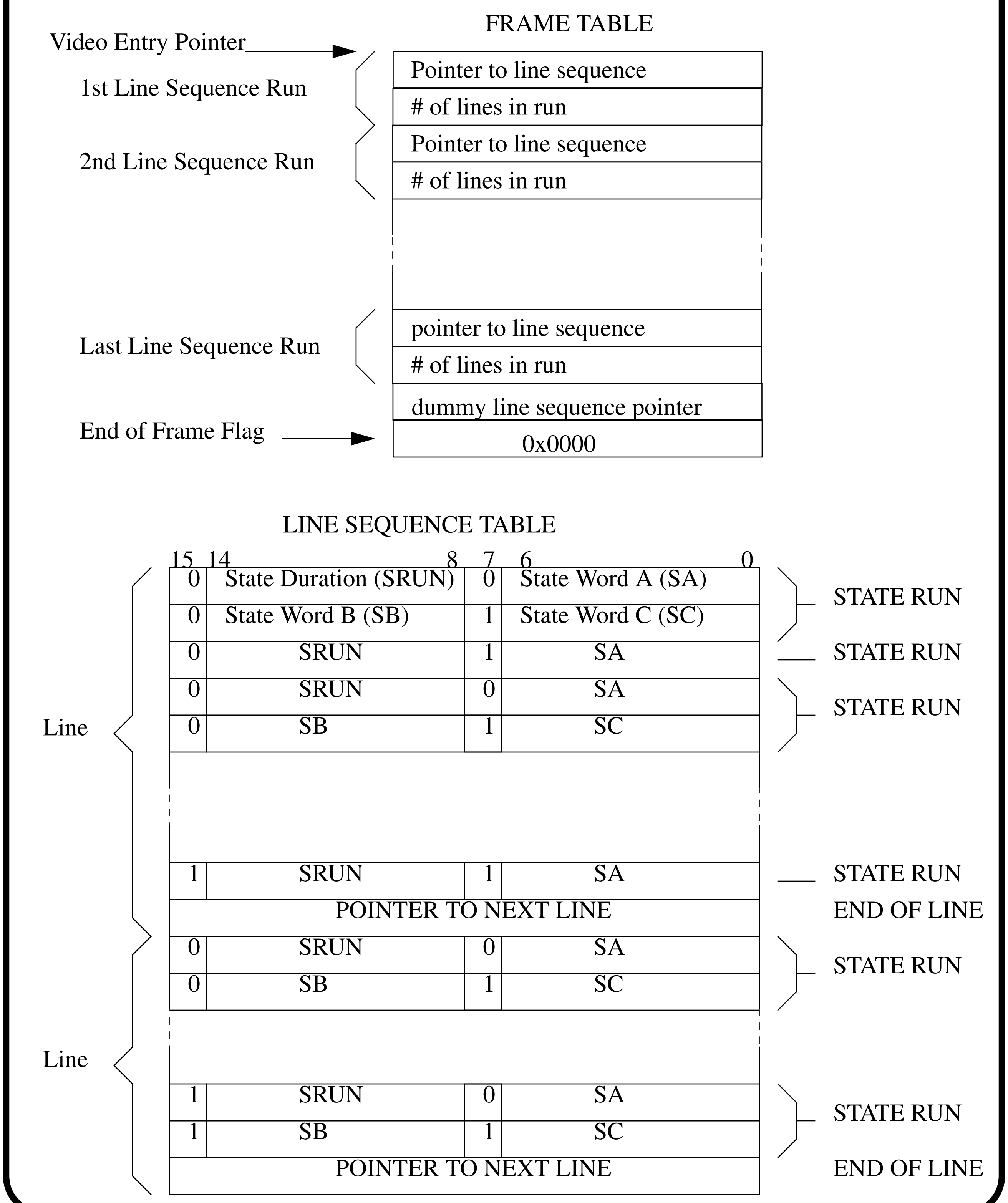
Before explaining the tables, it is useful to define the terms used. A state defines the value of each timing channel. A state consists of 21 timing channels divided into 3 groups of 7 channels each: state A, state B and state C. State A contains the most frequently changed channels, while state B and C contain the least frequently changed channels, regardless of whether the channels are used internally or externally. A state run defines a state and its duration, that is, the number of clock cycles that it lasts. The state duration, SRUN, is specified in units of 2-pixel clocks. A line is a list of state runs, and is intended to correspond to one horizontal line of the monitor. A line sequence is a circular linked list of lines. A line sequence run defines a line sequence and the number of lines the seuence repeats. A frame is defined by a list of line sequence runs.

The frame table consist of a linear list of line sequence run entries. Each line sequence run entry contains two words - a pointer to a line sequence and the number of lines for which the sequence is to be repeated. Note that this is the number of individual lines, not sequence cycles. Thus if a sequence cycles between four lines (L1, L2, L3 L4) , a count of 6 would generate six lines (L1, L2, L3, L4, L1, L2) then start the next sequence in the frame table. A line sequence entry with a line count of 0 indicates the end of the frame, which causes the timing generator to restart at the top of the list of line sequence runs (ie, the top of the frame table).

The Line sequence table contains a collection of lines linked by pointers. Each line is defined by a linear list of state runs. A state contains either 1 or 2 words that define 21 bits of state and 7 bits of state duration. Bits 15 and 7 of the first state word are used to flag the interpretation of the following words. Each state must contain state A and the state duration (the first word). The second word contains states B and C. If states B and C do not change from the previous state, then the second word may be omitted. Bit 7 of the SA/SRUN word indicates whether SB and SC are present: 0 = SB/SC present; 1= SB/SC not present. The maximum state duration is 127 clock cycles. The end of line is denoted by bit15 of SA/SRUN and SB/SC (if present): 1= end of line. (Note: bit 15 of the 2 state words must be the same). Bit 7 of SB/SC is always set to “1”. (Bits 7 and 15 in the SB/SC word are used to simplify the state machine).

Each line is followed by a pointer to the next line in sequence. A line that points to itself is a perfectly acceptable sequence. [In fact, in current VC1 timing tables, a line is always followed by a pointer to itself. Thus, a line sequence run contains only one line that repeats the specified number of times.] Another sequence example is a sequence of two lines that define even visible and odd visible lines of the display. Using the concept of a sequence allows the entire visible portion of the frame to be described with one entry in the frame table. The video timing table for a typical high resolution non-interlaced monitor will probably be about 150 words. For NTSC or PAL the table will probably occupy about 300 words.

VIDEO TIMING TABLE



The following table shows the assignment of video timing channels:

VIDEO TIMING CHANNELS

Name	#	level	power-on state	state A/B/C	To:	Description
TX_REQ_REX_N	18	L	I	A	REX3	Transfer request for VRAM transfer cycle. To skip a line in the frame (for interlaced modes) 2 transfer requests are executed.
SET_TSC_REX_N	10	L	I	B	REX3	Vertical sync used to clear line counter.
VERT_INT_REX_N	0	L	I	C	REX3	Vertical interrupt.
VERT_STAT_GIO_N	4	L	I	C	GIO	Vertical status.
CBLANK_DAC_N	20	L	I	A	RAMDAC	Composite blanking.
CSYNC_DAC_N	19	L	I	A	RAMDAC	Composite sync.
CSYNC_ARC_N	3	L	I	C	monitor	Composite sync for ARC-compatible monitor.
HSYNC_ARC_N	2	L	I	C	monitor	Horizontal Sync for ARC-compatible monitor.
VSYNC_ARC_N	1	L	I	C	monitor	Vertical Sync for ARC-compatible monitor.
SER_EN_RO_N	17	L	I	A	RO1	Serial enable to enable clocking of VRAM data. There will be 3 copies of this signal driven from the VC2.
DSPLY_EN_RO_N	16	L	I	A	RO1	Display enable to enable RO1 to drive pixel data out. There will be 3 copies of this signal driven from the VC2.
CBLANK_CMAP_N	9	L	I	B	CMAP	Composite blanking - loads fifo into cmap look up tables.
EOF_AB_N	8	L	I	B	AB1	End of frame. This signal is relevant only for high resolution monitors. It is a don't care for interlaced or stereo modes as they are not supported by the AB1 video.
HBLANK_AB_N	7	L	I	C	AB1	Horizontal blanking.
CBLANK_XMAP_N	6	L	I	C	XMAP	Composite blanking.
spare	5	L	I	C		Spare
VPOS_VC_N	13	L	I	B	internal	End of field/frame. Used to reset vertical line counter and to position cursor vertically.
EOF_VC_N	12	L	I	B	internal	End of field/frame. Used to flush DID fifo.
HPOS_VC_N	15	L	I	A	internal	Horizontal position. Leading edge positions cursor horizontally . Falling edge increments the vertical line counter and triggers cursor eol processing.]
VIS_LN_VC_N	14	L	I	A	internal	Visible line - asserted during the active portion of the line/frame. Used for DID generation.
ODDFIELD_VC_N	11	L	I	B	internal monitor	Indicates odd field in interlaced mode. Used by stereo monitor to indicate right/left frame.

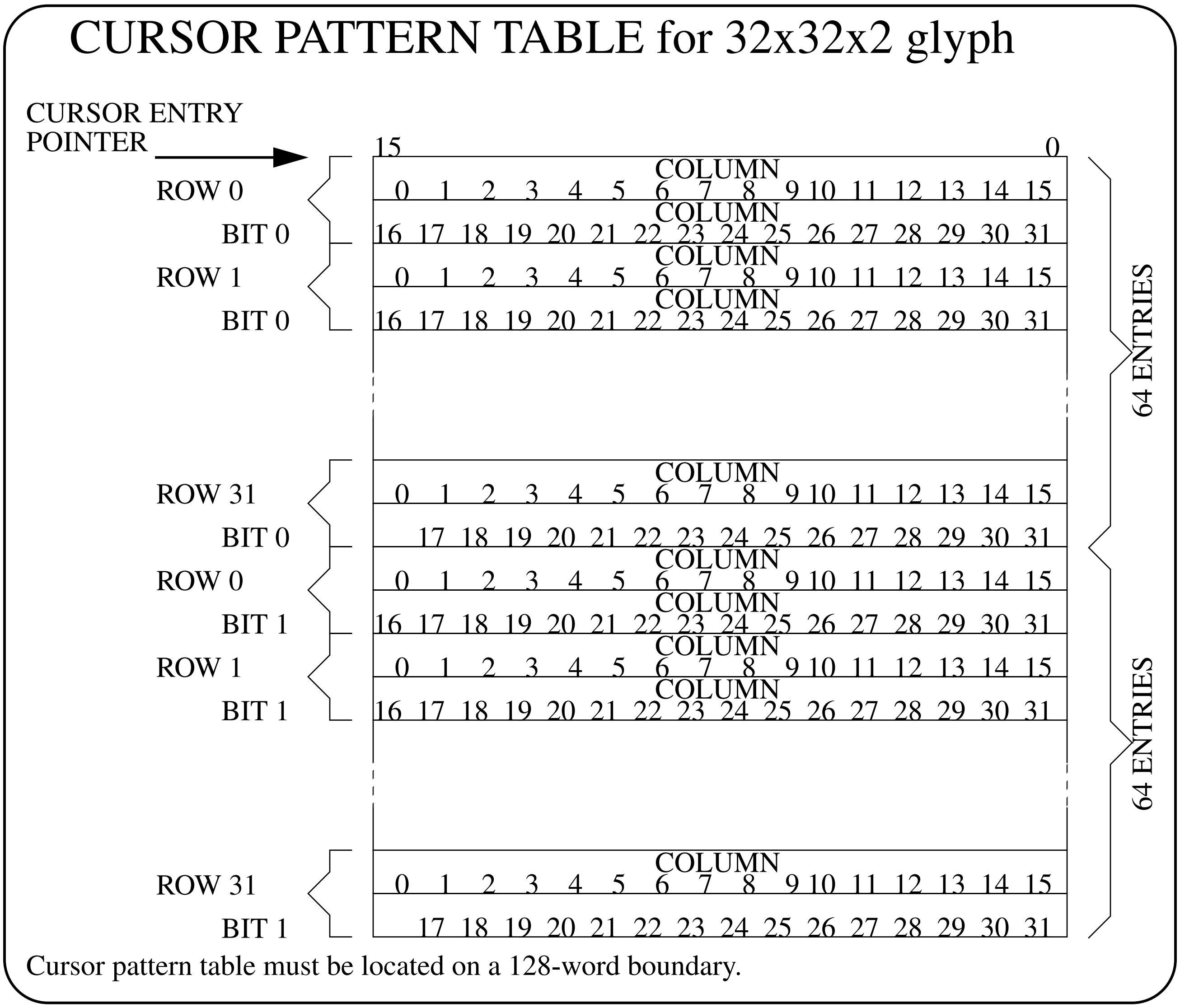
I=inactive, A=active.
SA = VT<20:14>
SB = VT<13:7>
SC = VT<6:0>

Video timing channel functions, as well as polarity are programmable. The above table indicates the presumed channel assignments and polarities for the Newport/Guiness implementation. In other systems, this may change. All channels power up to the high state, regardless of programming. Since the channels are assumed to be active low, this means that they will power up in the inactive state. If a channel is re-defined as active high, then it will power-up in the active state. Note that while the video timing channel for CBLANK_DAC_N powers up to the inactive state, the blackout bit in the Display Control Register defaults to "Blackout" at power-up which forces CBLANK_DAC_N to active (screen blanked) at power-up.

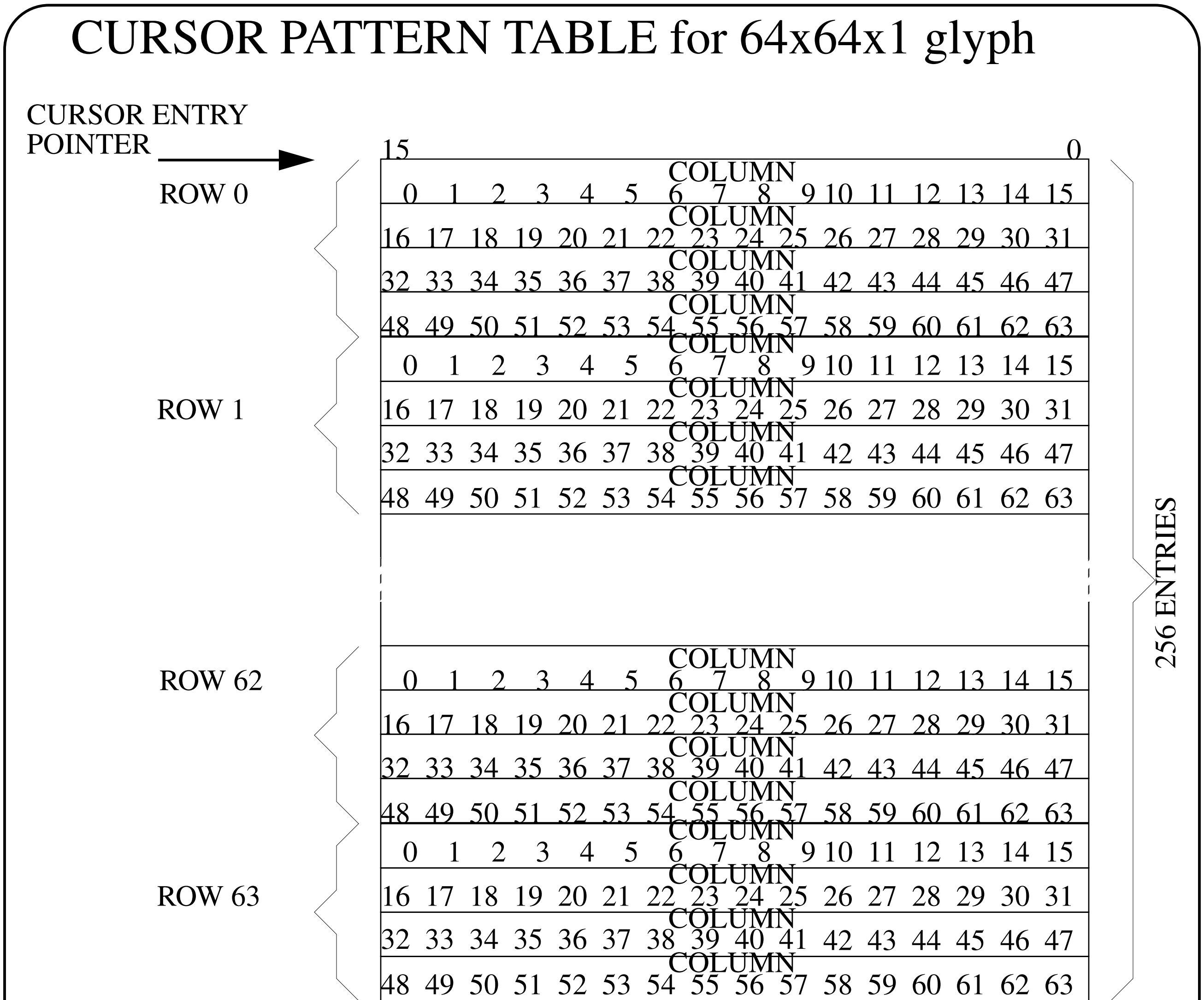
If additional channels are needed, it may be possible to combine some of the above signals, such as SET_TSC_REX, EOF_AB and/or EOF_VC.

3.4.2 Cursor Table

When the cursor glyph mode is used the data for the cursor glyph pattern is stored in a table in external memory as shown below:



Cursor pattern table must be located on a 128-word boundary.



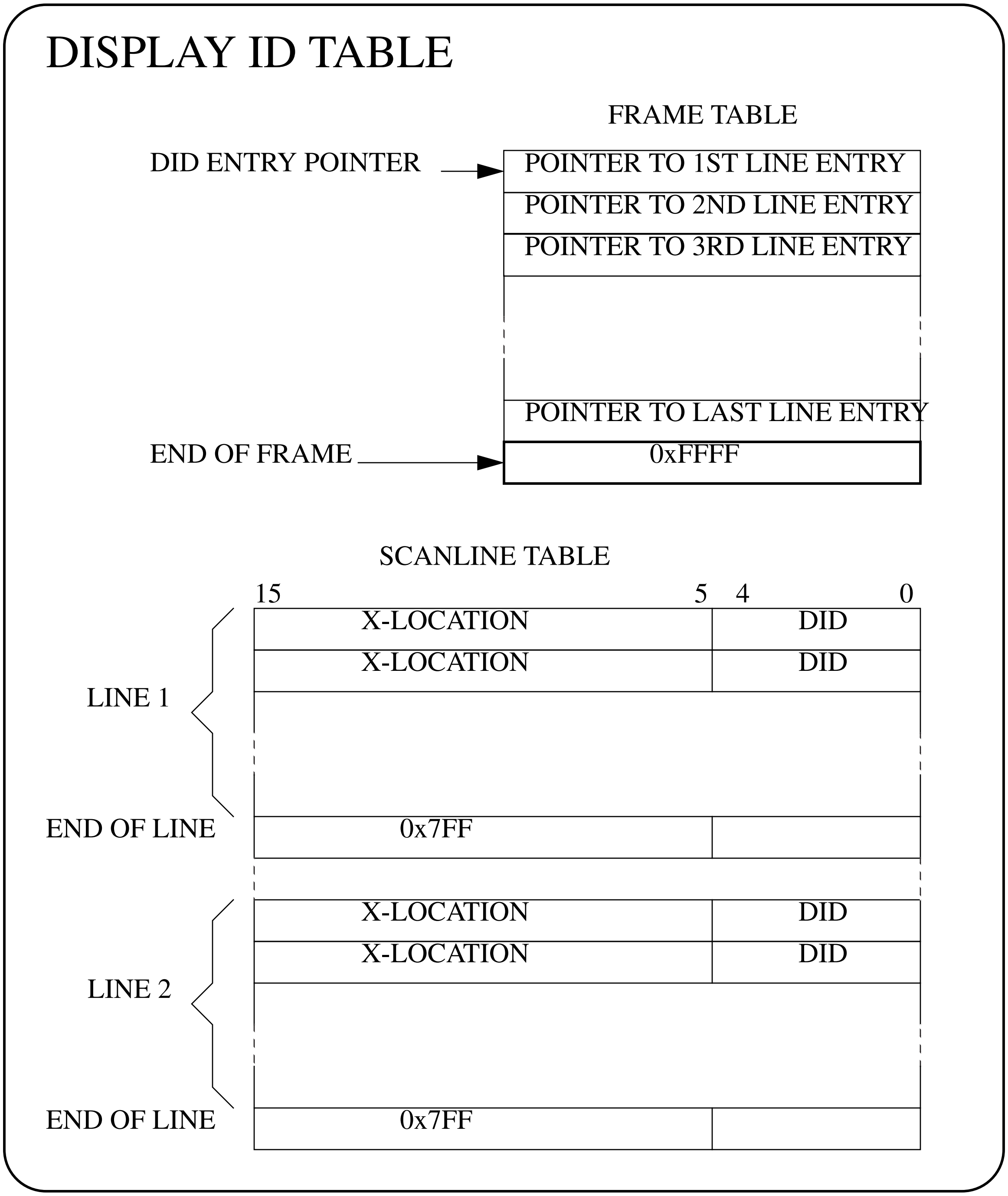
Cursor pattern table must be located on a 256 word boundary.

3.4.3 Display ID Table

The Display ID table consist of two parts, a frame table and a scanline sequence table as shown below.

The DID frame table consists of one entry for each scanline in the frame, followed by an end of frame flag. Each entry is a pointer to the start of the corresponding line in the scanline table. The end of frame flag is an entry of all 1's (0xFFFF). A scanline sequence consists of one or more transition records, followed by an end of line flag. Each 16 bit transition record has a 5 bit DID and an 11 bit horizontal coordinate denoting the X location at which the DID is active. The first X location in a scanline sequence must be 0. The end of line flag is a transition record with the X-location set to 2047 (all 1's). There is no relationship between the scanline sequences in memory. They need not be contiguous.

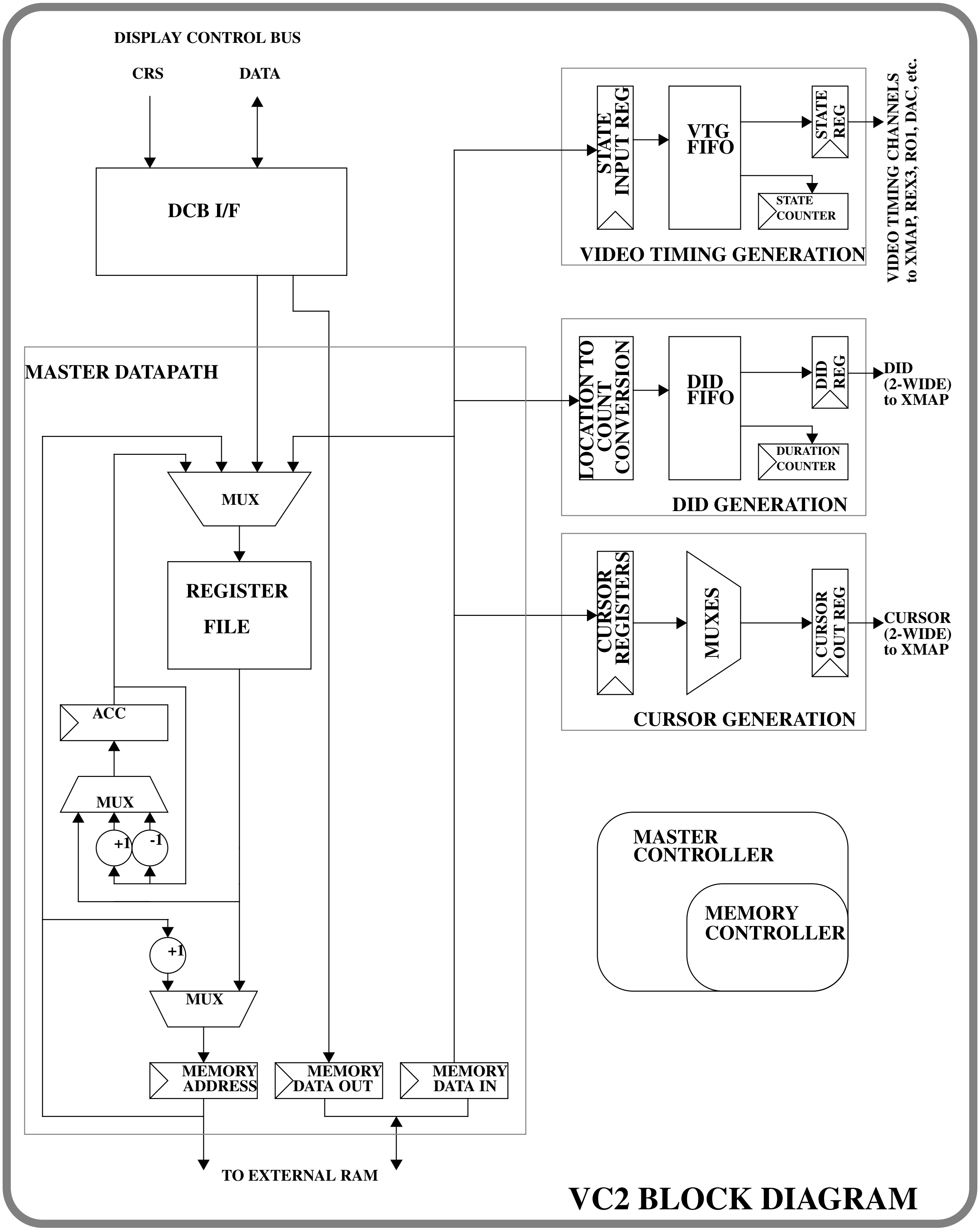
A scanline in the Scanline Table may exceed the actual width of the monitor. The VC2 will stop generating DIDs (and reading the DID table) when the scanline length (as indicated by the Scanline Length register) is exceeded or the end of line is



detected. In this way, the programmer may use the same DID table for different resolution monitors, merely by changing 2 registers: the Scanline Length and the DID Entry Pointer.

If DID table entries are updated while the VC2 is reading them from memory, it is possible for the hardware to read invalid data from memory and generate erroneous DID information. If this occurs, it will be corrected during the next vertical retrace period by flushing the DID fifo.

4.0 Architectural Description



The VC2 contains 6 major blocks as shown in the block diagram:

- •Display Control Bus Interface
- •Master Controller
- •Master Datapath
- •Video Timing Generation
- •Display ID Generation
- •Cursor Generation

Note: For readability the block diagram and following descriptions include the internal Rams as part of the functional block in which they are used. The actual VHDL for the chip will have a top level comprised of I/O buffers, 3 Rams and one block containing all other logic.

4.1 Clocks

The VC2 receives a clock which is one half the pixel rate (PCLK/2). The maximum frequency for PCLK/2 is 70 MHz. The entire chip will run with a multiple of PCLK/2. The backend chip functions (video timing, DID and cursor generation) and the DCB interface will always use PCLK/2. The rest of the chip (master controller and master datapath) will use one quarter the pixel rate (PCLK/4) for higher resolution monitors (where pixel clock rate >= 70 MHz) and PCLK/2 for lower resolution monitors (where pixel clock rate < 70 MHz).

The use of multiple clocks is done to accomodate the wide variation in pixel clock rates. The highest pixel clock rate for the VC2 is 140 MHz (PCLK/2=70MHz/14ns). Although the backend functions must run at this rate, it is too fast for the master controller and datapath, as well as the interface to ram. These front end functions can not run faster than 35Mhz (28 ns). The lowest resolution monitor (NTSC) has a pixel clock rate of 12.27 Mhz (PCLK/2=6.135 MHz/162 ns). To run the front end functions at PCLK/4 for this pixel rate would be far too slow, so PCLK/2 will be used.

In order to avoid clock skew associated with the muxing of clocks, all clocked elements use PCLK/2. A clock enable signal is generated which is equivalent to PCLK/4 in hi-res modes. This is used as a load enable for every flipflop or register which needs to run at PCLK/4. In low-res modes the clock enable signal is always asserted.

4.2 Display Control Bus Interface

The Display Control Bus (DCB) interface handshakes with the DCB in order for the host to read or write the external Ram tables and the internal status and control registers. The VC2 synchronzies the DCB command strobe signal (DCB_CS) to PCLK/2 and then responds with the acknowledge signal. When a DCB write occurs, the DCB interface will hold the first byte in a temporary register. When the second byte is written, the interface will assert a flag to the master controller, which then writes the 2 bytes of data to the appropriate register. Subsequent DCB accesses to the VC2 will be held off, by not asserting the acknowledge signal, until the write has completed. When a DCB read occurs, the interface asserts a flag to the master controller which then reads the addressed register. The first byte is driven onto the DCB, while the second byte is held in a temporary register for the next DCB read.

4.3 Master Datapath and Controller

The main job of the master controller is to process requests from the other blocks to read or write the external Ram. At the heart of this is a 16x16 ram register file which contains most of the VC2's programmable registers as well as all address pointers used to access the external Ram tables. When a block requests a memory read or write, the master controller will get the appropriate address pointer from the register file, perform the memory read or write, and direct the data to and from the required registers. The address pointer is returned to the register file, and any other required functions (such as decrementing line counts) will be executed. In addition, during the vertical retrace period the controller performs some necessary setup and cleanup functions.

4.4 Video Timing Generation

Video timing generation comprises a 16x28 fifo along with input and output registers and control logic. The 28 bits of data in the fifo consists of 21 video timing channels and 7 bits of duration, indicating how many PCLK/2 cycles the state will be applied for. Fifo depth is monitored and a FIFO_NOT_FULL signal is driven to the master controller. Whenever the fifo is not full, the master controller treats this as a request for the next video timing state. Due to pipelining and latency in the master controller, FIFO_NOT_FULL will be de-asserted when the fifo depth reaches 12. When the master controller has read the next video timing state from memory, it is written into the fifo.

At the other end, a video timing state is read from the fifo and driven to points outside and within the chip for the number of cycles specified by the state duration field. An attempt to read from an empty fifo is considered an error condition and should never happen during normal operation. Should this condition occur, the VC2 should be able to recover and generate timing once the fifo has been re-filled.

4.5 Cursor Generation

During the horizontal blanking period, the memory controller will load the cursor registers with all 32 pixels of cursor. The cursor generation logic maintains a count of pixels in the X direction relative to the start of horizontal blanking. When the X counter matches the X cursor location, the appropriate 2 pixels (or 1 pixel and 0’s) will be selected from the cursor registers and driven to the Xmap9 chip each cycle, until all 32 cursor pixels have been sent. If the cursor is not enabled or there is no cursor on the current scanline, then 0’s will be driven on the cursor lines. If the crosshair cursor mode is selected, then the crosshair cursor data will be driven only when the X counter matches the X cursor location, or when the Vertical Line Count matches the Y cursor location. Otherwise, all 0’s will be driven on the cursor lines.

4.6 DID Generation

DID generation consists of a 64x21 fifo along with output registers and control logic. Additionally, there is a sub-block which converts the X location of the DID from the Ram table into a count of the number of cycles each DID is to be applied. The 21 bits of fifo contains 2 DID’s (5 bits each) and a 11-bit duration field which specifies how many cycles to apply the DIDs.

The operation of the fifos and control logic is similar to the video timing generation. Fifo depth is monitored and a FIFO_NOT_FULL signal is driven to the master controller. Whenever the fifo is not full, the memory controller treats this as a request for the next DID/X location. Due to pipelining and latency in the master controller, FIFO_NOT_FULL will be de-asserted when the fifo depth reaches 58.

During the visible portion of the screen, DID’s are read from the fifo (2 pixels wide) and driven to the Xmap9 chip for the number of cycles indicated by the duration field. An attempt to read from an empty fifo is considered an error condition. This can happen if there are too many DID transitions in a line. Should this condition occur, the VC2 should be able to recover durring the next vertical retrace period.

5.0 Theory of Operation

This section gives a description of the internal operation of the VC2. To supplememnt this description, the reader should refer to the actual vhdl code which can be found in the following directory:

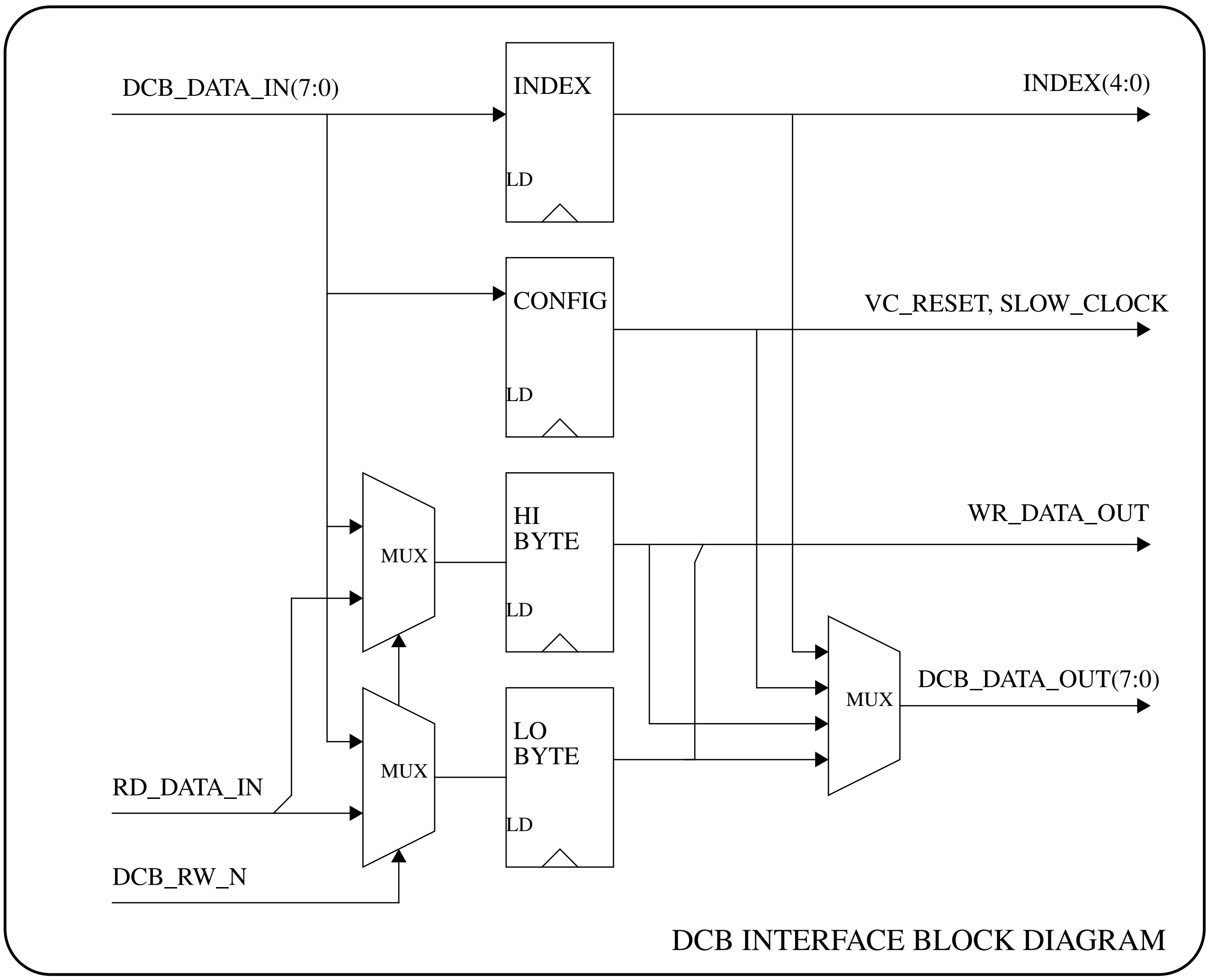
nexus:/d/newport/asics/vc2/vhdl/tsrc

The name of the relevant block for each subsection is noted in brackets [ ]. The vhdl is divided into 6 major blocks (cur, dcb, did, mac, mdp, vtg). These blocks are connected together into a higher level block called “vc”. The top chip level is “vc2”, which contains the top logic level (vc) the I/O drivers (io) and the pseudo jtag module (bstop). Modules below the 6 major blocks generally have the following naming convention:

majorblock_subblock.vhdl

5.1 Display Control Bus Interface [dcb]

A detailed block diagram for the Display Control Bus section is shown below:

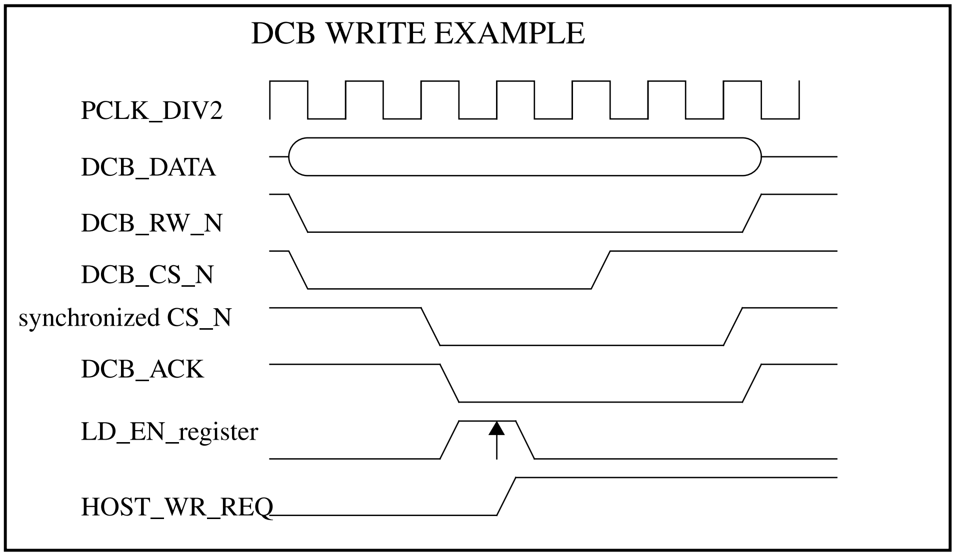


5.1.1 Synchronization

Since the VC2 is running asynchronously to the DCB, the command strobe (DCB_CS) will be synchronized before being used by any logic.

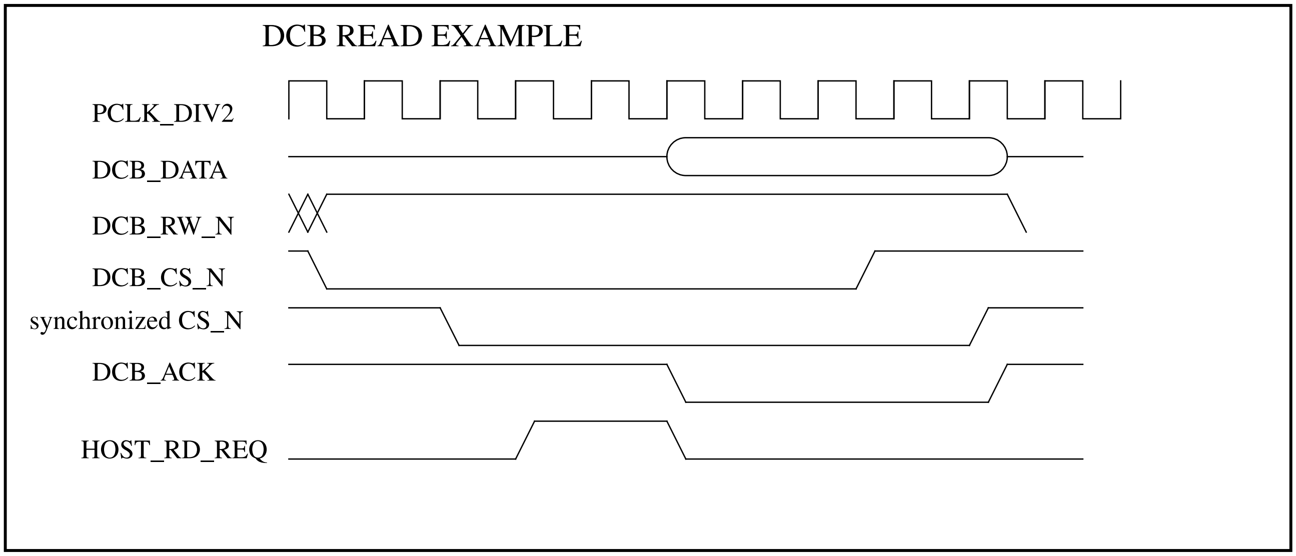
5.1.2 DCB Writes

When a DCB write occurs to the Index register (CRS=0) or the configuration register (CRS=2, Index=31), the specified register is loaded with DCB data and the operation is complete. When a DCB write occurs to the high byte of any other register (CRS=1) or of memory data (CRS=3), the DCB data is saved in the temporary register HIBYTE. When a DCB write occurs to the low byte of any other register (CRS=2) or of memory data (CRS=3), the DCB data is saved in the temporary register, LOBYTE. At the same time, a flag is set to indicate to the master controller that a host write request is waiting to be processed. The assertion of this flag will stall any future DCB writes to the VC2 (by means of the DCB_ACK signal) until cleared by the master controller. When a stall is not indicated, there will be an immediate acknowledge of the DCB write.



5.1.3 DCB Reads

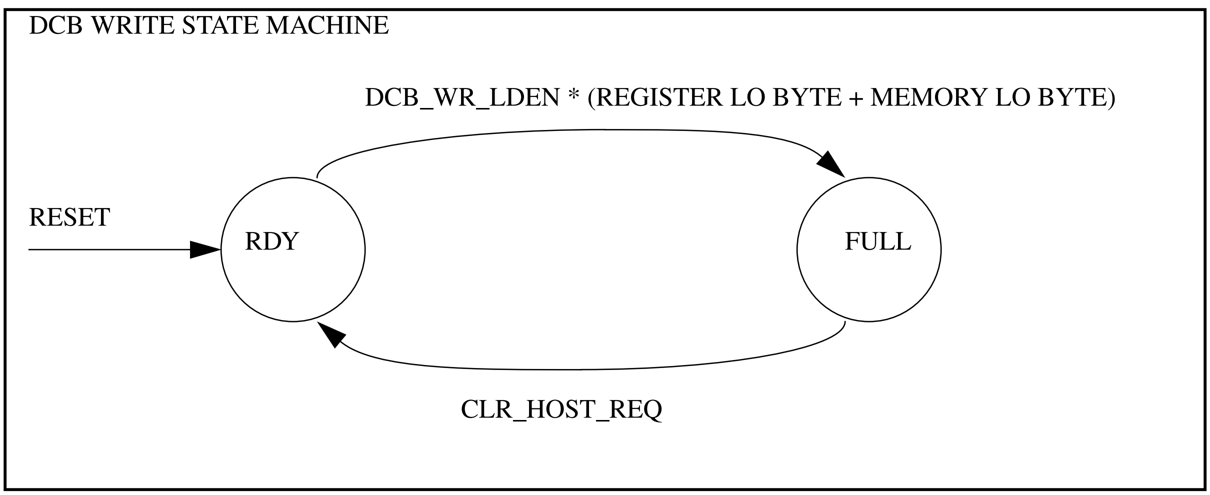
When the DCB reads from either the Index register (CRS=0) or the configuration register (CRS=1,2, Index=31), the data will be driven onto the DCB data bus from the specified register and the request acknowledged. When the DCB reads from the high byte of any other register (CRS=1) or of memory data (CRS=3), a flag is set which indicates to the master controller that a host read request is waiting to be processed. The master controller will process the request and load the read data into the HIBYTE and LOBYTE registers. The load enable for these registers indicates that the request has been processed. The HIBYTE register will be driven onto the DCB data bus and the request will be acknowledged. When the DCB reads the low byte of a register (CRS=2) or of memory data (CRS=3), the controller assumes that the required data remains in the LOBYTE register from the previous HIBYTE read and will immediately acknowledge the request and drive LOBYTE to the DCB data bus.



5.1.4 DCB Control [dcb_ctrl]

5.1.4.1 Write State Machine

The main control element for DCB writes is the state machine shown below:



Each byte written over the DCB to the VC2 will fit one of two situations. 1) It can be handled entirely by the DCB interface or 2) it requires processing by the master controller. The operations which require processing by the master controller are writes to the low byte of either a register (other than Index or Configuration) or memory data. Writes to the high byte of a register or memory data, as well as writes to the Index or Configuration registers require no processing by the master controller.

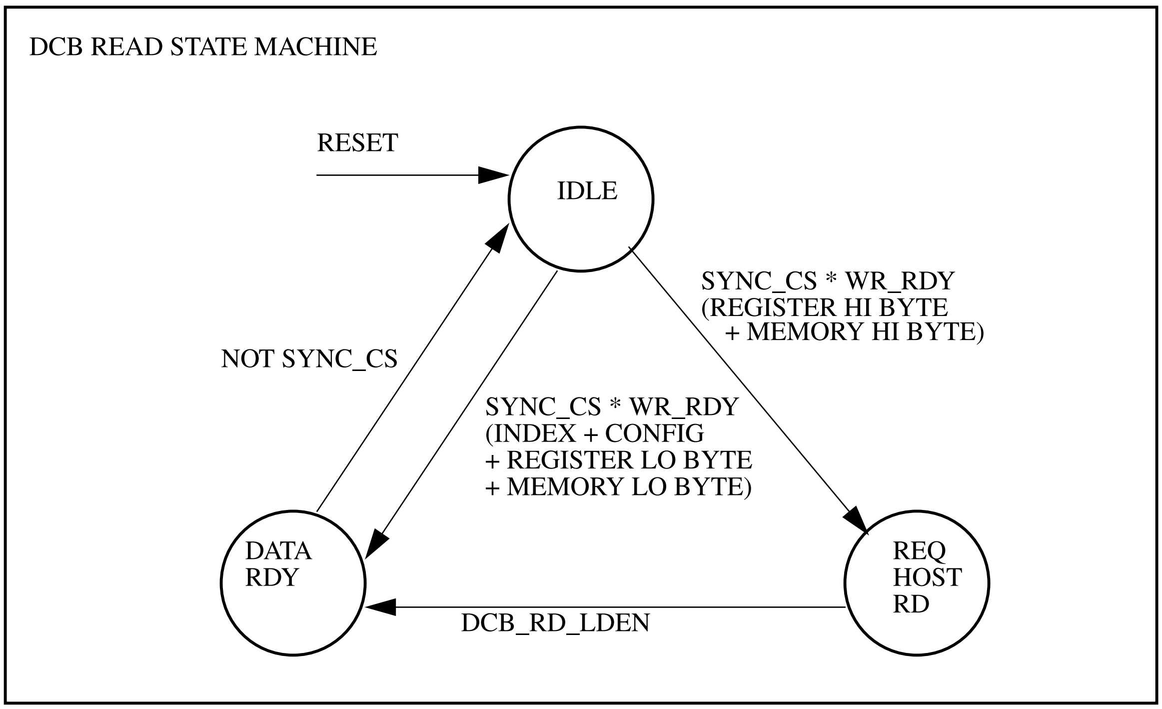
The procedure for a write begins with the synchronization of DCB_CS. When the synchronized CS is detected, DCB_ACK will be asserted if the STALL condition is not in effect, that is, the temporary registers do not contain data from a previous write which has not yet been processed. The write strobe (DCB_WR_LDEN) is asserted during the first cycle of acknowledge, and the appropriate register is loaded. If this write requires no processing then the operation is complete. If processing is required, the state machine cycles to the FULL state which asserts the HOST_WR_REQ flag, indicating to the master controller that the temporary register contains 2 bytes of write data which needs to be written to either a register or memory.

The current DCB write will be acknowledged, but any subsequent writes while in the FULL state will be stalled. When the master controller finishes processing the write, the request is cleared (CLR_HOST_REQ) and the state machine returns to the RDY state. If the DCB write operation is for memory data, a flag (WR_REQ_IS_MEM) will be set to indicate this to the master controller. Also, the MD state machine (described below) will be toggled.

The STALL condition, which inhibits the assertion of DCB_ACK, will be asserted at the end of a DCB operation when the Write State is FULL, and will be desserted immediately when the Write State goes to RDY. Thus, the FULL signal will be delayed by a flip-flop which is loaded during the first cycle where SYNC_CS is deasserted. STALL uses this delayed version of FULL so that it does not affect the acknowledge for the DCB operation which caused the FULL state, but may affect the acknowledge for subsequent DCB operations.

5.1.4.2 Read State Machine

The main control element for DCB reads is the state machine shown below:



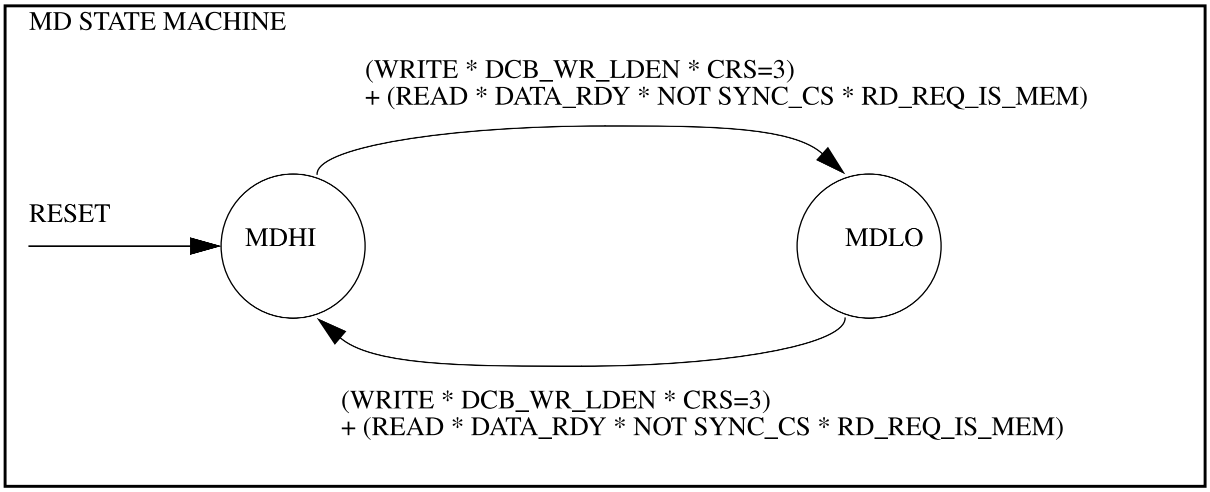
Each byte read over the DCB from the VC2 will fit one of two situations. 1) It can be handled entirely by the DCB interface or 2) it requires processing by the master controller. A read requires no processing by the master controller if the data all ready exisits in a register in the DCB interface. This is is the case for the low byte of a register or memory data, as well as the Index or Configuration registers. The operations which require processing by the master controller are reads from the high byte of either a register (other than Index or Configuration) or memory data. Since the high byte is read first, the word has not yet been transfered into the holding register.

The procedure for a read begins with the synchronization of DCB_CS. When the synchronized CS is detected and processing is required, the state machine cycles to REQ HOST RD which sets the HOST_RD_REQ flag to the master controller. When the data has been retreived, the master controller will assert the read strobe, DCB_RD_LDEN, which simultaneously loads the holding register with the register or memory word, and sends the state machine to DATA RDY. In the DATA RDY state, the appropriate byte of data is driven to the DCB and acknowledge is asserted. When CS is deasserted, the state machine returns to IDLE. If the read operation requires no processing, then the state machine goes directly to DATA RDY, driving the data and asserting acknowledge. Since it is possible for a read operation to occur before a previous write has completed, the read state machine remains in IDLE if the write state machine indicates FULL. If the DCB write operation is for memory data, a flag

(RD_REQ_IS_MEM) will be set during the first cycle of SYNC_CS, to indicate this to the master controller. Also, the MD state machine (described below) will be toggled.

5.1.4.3 MD State Machine

The MD state machine keeps track of whether the current read or write to CRS 3 references the high or low byte of memory data.



Since the high byte and low byte of memory data are accessed through the same CRS, it is necessary to keep track of which byte the current (or next) DBC operation will reference. This is done with a single state machine for both reads and writes. The state machine toggles states every time there is a DCB read or write to CRS 3.

5.1.4.4 Resets

From the hardware point of view, there are 3 resets in the VC2.

- •1• The asynchronous reset comes directly from the board. It is used for only 2 things: to tristate bidirectional I/O drivers during power up and to generate the synchronous reset.
- •2• The system reset is a synchronized version of the asynchronous reset. It will reset the entire chip (all state machines and any registers which need to come up in a known state) either directly or indirectly (via the soft reset). The DCB interface and the Display and Cursor Control Register are reset by the system reset.
- •3• The soft reset is the programmatic reset from the Configuration Register. Since it is reset by the system reset, it is effectively the OR of the programmatic and system resets. It resets everything which needs to be in a quiescent state when the pixel clock frequency is changed. Most of the chip, with the exception of the DCB Interface and the Display and Cursor Control Register is reset by Soft Reset.

5.1.4.5 Clock Enable [dcb_clk]

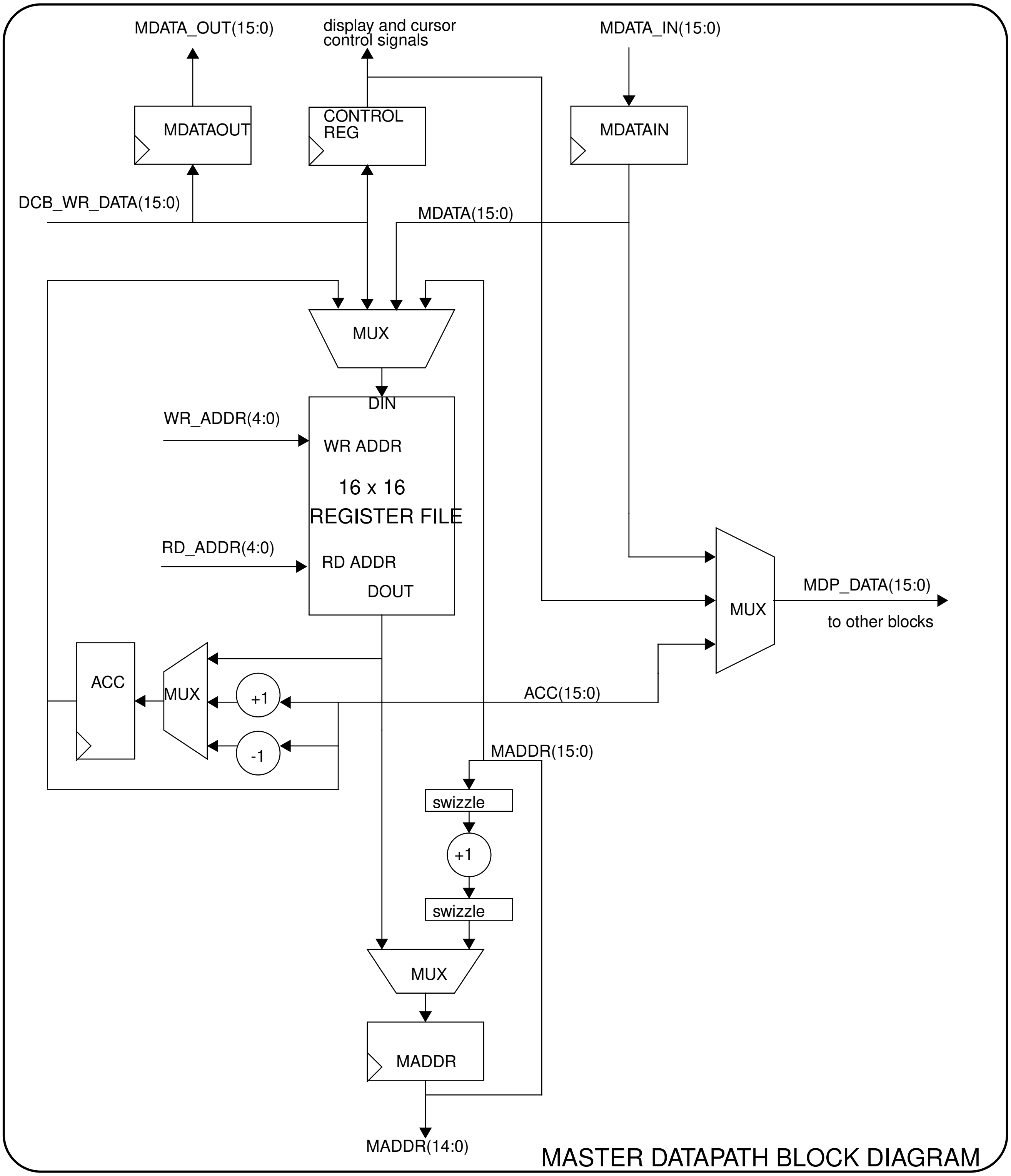
Part of the VC2 will always run at the PCLK/2 rate. Part will run at either PCLK/2 or PCLK/4 depending on the clock frequency. This is controlled by the clock enable (CLK_EN) signal. When the Slow Clock bit in the configuration register is set, clock enable will always be asserted and those parts of the chip that use the dual clocking scheme will run at PCLK/2. When the Slow Clock bit is not set, the clock enable will be driven by PCLK/4, that is, clock enable will be asserted every other cycle, and the parts of the chip which use the dual clocking scheme will effectively run at PCLK/4.

The DCB interface always runs at PCLK/2. Since the master controller uses the dual clock, any signals it sends to the DCB interface must be qualified by the clock enable, so that they are asserted to the DCB interface only when valid.

The clock enable signal is reset by the ENTEI pin for the sole purpose of having PCLK/4 clock enable come up in a known state. This allows the chip to run in a known phase on tester. This will also be usefulfor Nextgen which will use 2 VC2's which must run in sync.

5.2 Master Datapath [mdp]

A detailed block diagram of the Master Datapath is shown below.



All programmer-accessible registers (except Index , Configuration and Display and Cursor Control) reside in a Register File which is implemented using an internal dual-ported Ram. The master controller handles all reads and writes to external Ram using these registers and one common datapath. When the controller receives a request (from VTG, Cursor, DID or DCB) to access the Ram, it retrieves the appropriate pointers and counters from the Register File, and using the other registers provided (Accumulator, memory address registers, etc) it will process the request, update the pointers and counters and return them to the Register File.

5.2.1 MADDR - Memory Address Register [mdp_maddr]

MADDR is a 16-bit register which holds the memory address for any access to the Ram. The register can be loaded from the register file or incremented by either 1 or 2. Generally when the controller is accessing a table in Ram, the pointer will be loaded into MADDR and will be incremented following every read or write. Thus, after processing, MADDR will point to the next entry in the table and can then be written back into the Register File.

The SWIZZLE function re-arranges the bits at the input and output of the incrementer. When reading the cursor glyph pattern table, it is necessary to read 4 locations in memory which are not contiguous. For a given row of the cursor, the 4 words of pattern data are located at +0, +1, +40 and +41 relative to the running cursor table pointer. The simplest way for the hardware to address these 4 locations is to use the incrementer, with the bits swizzled so that the required addresses are generated:

INCREMENTER<15:0> <=> MADDR<15:7, 5, 4, 3, 2, 1, 0, 6>

Thus, incrementing with the SWIZZLE function enabled produces the following addresses (starting from 0):

00
40
01
41
02
42
03
43
04

A flag, MADDR_6to0_IS_0, is asserted when MADDR<6:0> = 0.

5.2.2 ACC - Accumulator [mdp_acc]

ACC is a 16-bit general purpose accumulator register. It can be loaded from the Register file, cleared to all 0's, incremented by either 1 or 2, or decremented by 1. This register is used for any counting that needs to be done during processing, ie, decrementing or incrementing line counts. It is also used for compare funtions. A flag, ACC_IS_0, is asserted when ACC<15:0> =0.

5.2.3 MDATA - Memory Data Register [mdp_mdata]

MDATA consists of 2-16 bit registers, MDATAOUT and MDATAIN. MDATAOUT holds data to be written to the Sram. It is loaded with DCB_WR_DATA, memory write data from the DCB interface. MDATAIN holds data which has been read from the Sram.

5.2.4 DCR - Display and Cursor Control Register [mdp_dcr]

The DCR is maintained outside the register file since it contains control bits which are frequently used by the master controller and othe blocks. It is loaded with DCB_WR_DATA, register write data from the DCB inetrface. The DCR is cleared to all 0's (everything disabled) by the system reset. It is not affected by the soft reset.

5.2.5 Register File (RF) and Datapath Muxes (RFMUX and DMUX) [mdp_rfmux, mdp_dmux]

The Register File contains 16 16-bit registers. The address for the Register File is generated by the Master Controller. Since it is a dualport Ram, the Register File can be written and read simultaneously, although not at the same location. The data input to the register file can come from a datapath register (MADDR, MDATAIN, ACC) or from the DCB Interface (DCB_WR_DATA). The selection of the data source is sone via the RFMUX.

The DMUX drives the MDP_DATA bus, which is the means for transfering data from the master datapath to the other blocks. The source for MDP_DATA can be either ACC, DCR or MDATAIN. This bus is used to write DID and Video Timing State into the fifos, to write cursor pattern data into the cursor registers, and write memory or register data to the DCB interface.

5.3 Master Controller [mac]

Due to the complexity of the Master Controller state machines , they are not documented in detail either here nor in the vhdl code. Instead the reader should refer to the document "VC2 Master Controller State Tables" in the file:

nexus:/d/newport/asics/vc2/spec/statetable

The master controller consists of 5 state machines and a request prioritizer. Together, the state machines provide control signals to the master datapath (ie, register loads, increments, mux selects and register file addresses) as well as the control signals to the external SRam (output enable and write strobe).

The Request prioritizer takes all the signals which indicate a request for the master controller and outputs a request vector (REQ_VECT) indicating the highest priority request. The request vector is 10 bits wide. At any given time, only one bit will be set - the bit which corresponds to the current highest priority request.

Each of the five state machines operates on a different set of requests. Only one state machine will operate at a time, on the current highest priority request. In addition to the request vector, there are 2 request signals from the MSM, DID_EOL_REQ and VTEOL_REQ, which go directly to the DSM and the VSM, respectively, for end of line processing. These requests will take priority over all other requests.

Most of the control signals from the state machines are generated in the cycle in which they are used. The corresponding signals from all the state machines are OR'ed together and then driven to the datapaths or backend blocks. The register file addresses and write signals are generated 1 cycle early and then OR'ed and latched. This is done in order to meet the internal Ram timing.

5.3.1 Request Prioritizer [mac_req]

The Request block takes all the signals which indicate a request for the master controller and outputs a request vector (REQ_VECT) indicating the highest priority request. The request vector is 10 bits wide. At any given time, only one bit will be set - the bit which corresponds to the current highest priority request. The requests, in priority order, are as follows:

VT1_REQ - First video timing request. This occurs when the video timing function is first enabled in the DCR. This request must be explicitly cleared by the state machine after the request is processed.

VT_REQ - Video timing request. This request occurs whenever the video timing function is enabled and the VTG fifo is not full.

CUR_EOF_REQ - Cursor end of field request. This request occurs when the timing channel VPOS_VC is asserted. The request occurs only when the video timing function is enabled, but occurs whether or not the cursor function is enabled. The request must be explicitly cleared by the state machine after the request is processed.

CUR_EOL_REQ - Cursor end of line request. This request occurs when the timing channel HPOS_VC is deasserted. The request occurs only when the video timing function is enabled, but occurs whether or not the cursor function is enabled. The request must be explicitly cleared by the state machine after the request is processed.

HOST_MEM_WR_REQ - Host memory write request. This request occurs when the dcb interface indicates a write to external memory. The request must be explicitly cleared by the state machine after the request is processed.

HOST_REG_WR_REQ - Host register write request. This request occurs when the dcb interface indicates a write to a datapath register. The request must be explicitly cleared by the state machine after the request is processed.

HOST_MEM_RD_REQ - Host memory read request. This request occurs when the dcb interface indicates a read from external memory. The request must be explicitly cleared by the state machine after the request is processed.

HOST_REG_RD_REQ - Host register read request. This request occurs when the dcb interface indicates a read from a datapath register. The request must be explicitly cleared by the state machine after the request is processed.

DID1_REQ - First did request. This occurs when the did function is first enabled in the DCR. This request must be explicitly cleared by the state machine after the request is processed.

DID_EOF_REQ - Did end of field request. This request occurs when the timing channel EOF_VC is asserted. The request must be explicitly cleared by the state machine after the request is processed.

DID_REQ - Did request. This request occurs whenever the DID function is enabled and the DID fifo is not full.

5.3.2 Main Processing State Machine [mac_msm]

The Main Processing State machine (or MSM) handles the most time-critical operations. That is, those which must be executed with a minimum of wasted states in order to provide adequate performance. The requests which fall into this category are those which involve filling the VTG or DID fifos and host memory write requests (VT_REQ, DID_REQ, HOST_MEM_WR_REQ). These are the requests that may occur many times on during a line on the screen and thus have the most impact on performance.

5.3.3 Host Processing State Machine [mac_hsm]

The Host Processing State Machine (HSM) handles all host requests (except for host memory writes which are handled by the MSM. These request are HOST_MEM_RD_REQ, HOST_REG_RD_REQ and HOST_REG_WR_REQ.

5.3.4 Cursor Processing State Machine [mac_csm]

The cursor processing state machine (CSM) handles CUR_EOF_REQ and CUR_EOL_REQ. That is, cursor end of line and end of field processing, which includes everything having to do with the Y component of the cursor. Specifically, the CSM will clear and increment the vertical line counter, compare the current Y position with the programmed Cursor Y position to indicate to the cursor generation logic when the cursor is appearing on the current line, and transfer cursor patterns from memory to the pattern registers in the Cursor Generation Block.

5.3.5 VT End of Line/ End of Run/ End of Field Processing State Machine [mac_vsm]

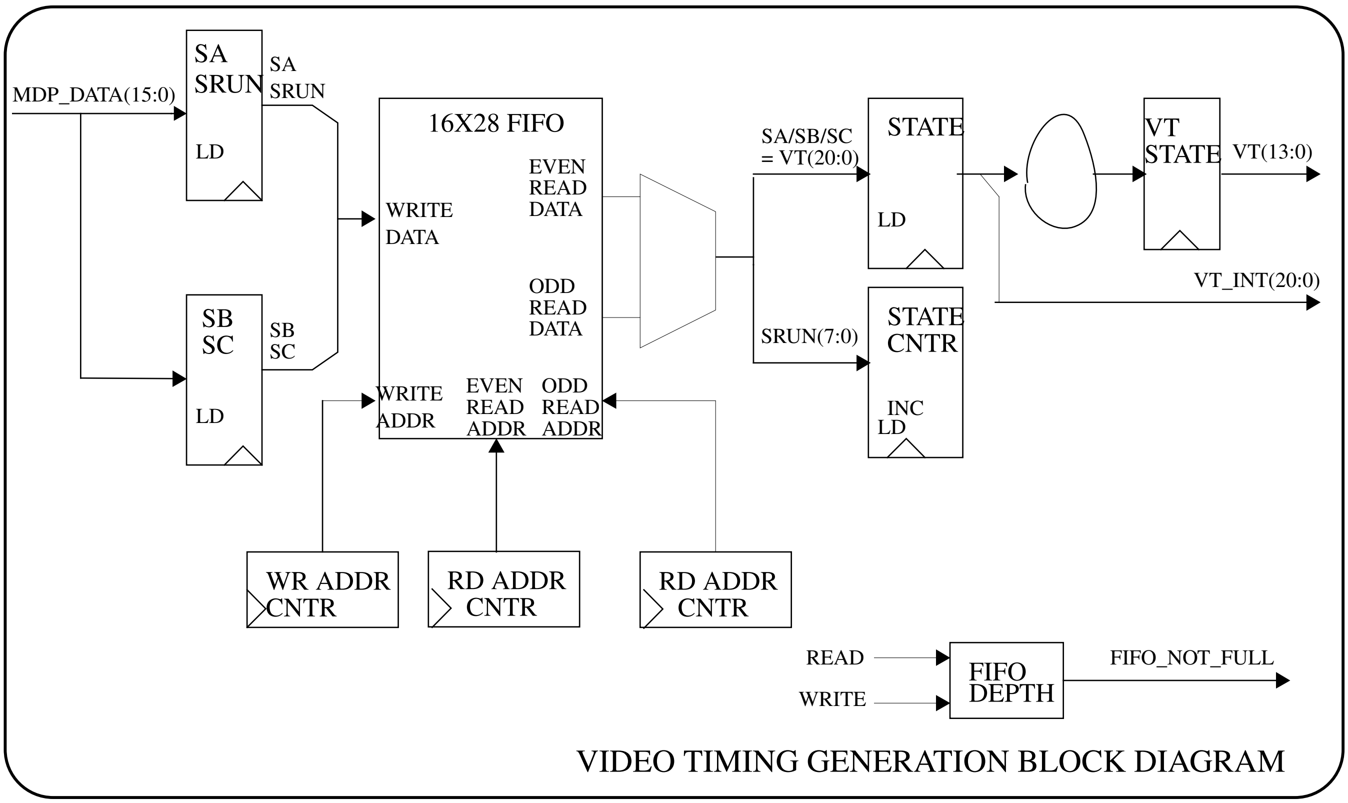
The VT EOL/EOR/EOF processing state machine (VSM) handles certain special cases involved in managing the video timing tables. It executes the first video timing request (VT1_REQ) as well as genlock. It also handles the VT end of line request which comes directly from the MSM. Basically, the VSM manages the pointers and counters so that whenever the MSM executes, the appropriate registers in the register file will always point to the next video timing state.

5.3.6 DID End of Line/ End of Field Processing State Machine [mac_dsm]

The DID EOL/EOF processing state machine (DSM) handles certain special cases involved in managing the Did tables. It executes the first DID request (DID1_REQ) as well as the DID end of field request (DID_EOF_REQ). It also handles the DID end of line request which comes directly from the MSM. Basically, the DSM manages the pointers and counters so that whenever the MSM executes, the appropriate registers in the register file will always point to the next Did.

5.4 Video Timing Generation [vtg]

A detailed block diagram for the Video Timing Generation section is shown below:

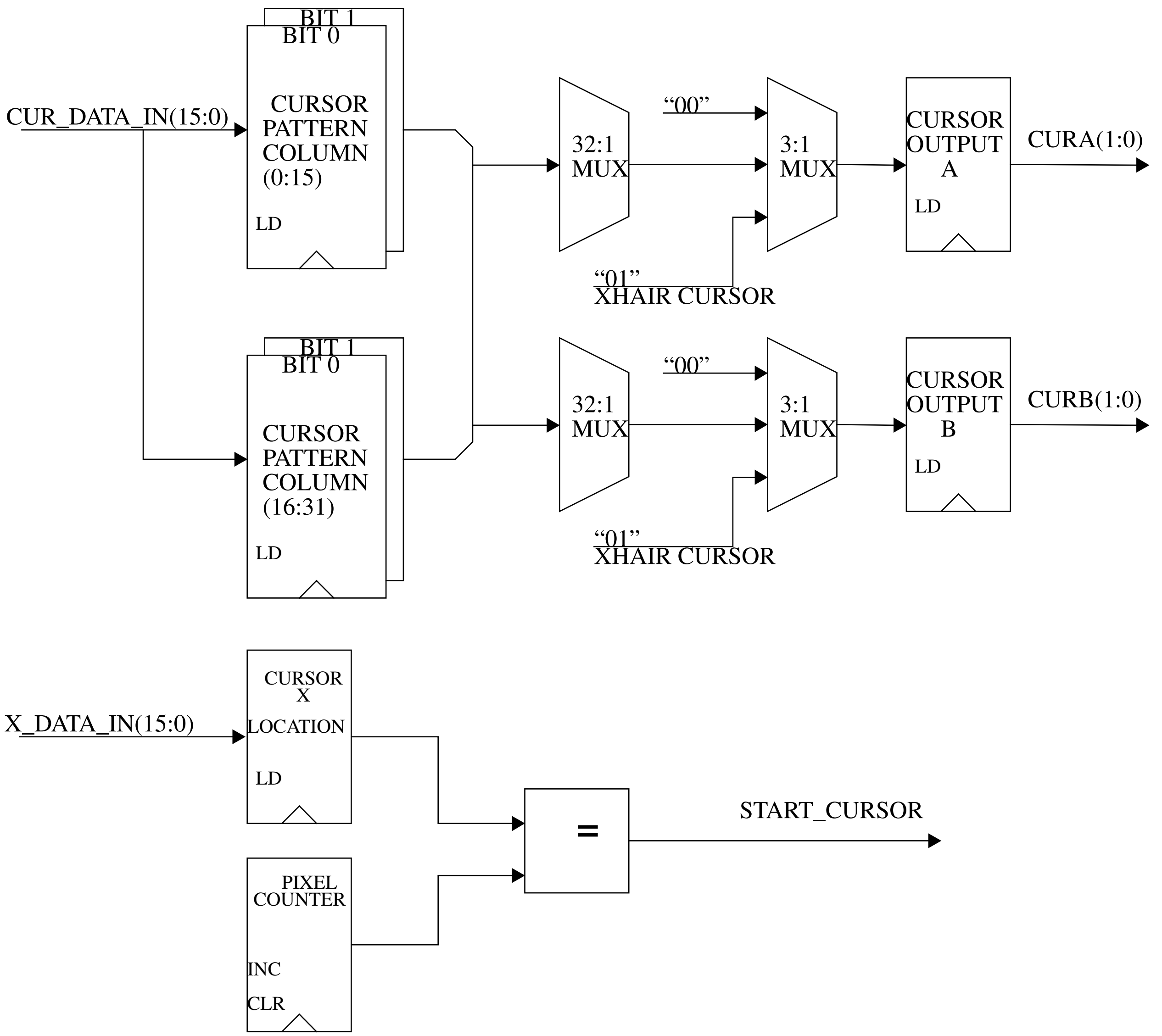


The main element of the Video Timing Generation is a 16x28 triple-port ram configured as a fifo, which holds the video timing states as they are retreived from external ram. When the fifo is not full, as determined by the depth counter, the master controller reads the next video timing state from external ram and writes into the state input registers (SA/SRUN/SB/SC). The following cycle, the state is written into the fifo and the write address counter is incremented. When a state is read from the fifo, the 21 bits of timing channels are loaded into the State register and the 7 bits of SRUN are loaded into the state counter. Each subsequent cycle, the state counter is decremented. When it reaches 0, the next state is read from the fifo and the read address counter is incremented. For timing purposes, additional pipeline registers are used to drive the video timing channels out of the chip.

5.5 Cursor Generation [cur]

A detailed block diagram for the Cursor Generation section is shown below:

At the start of the Hblank period, if the cursor is present on the upcoming scanline, the master controller will read the cursor pattern for the row of 32 pixels and load it into the cursor pattern registers. The pixel counter is also cleared at the start of



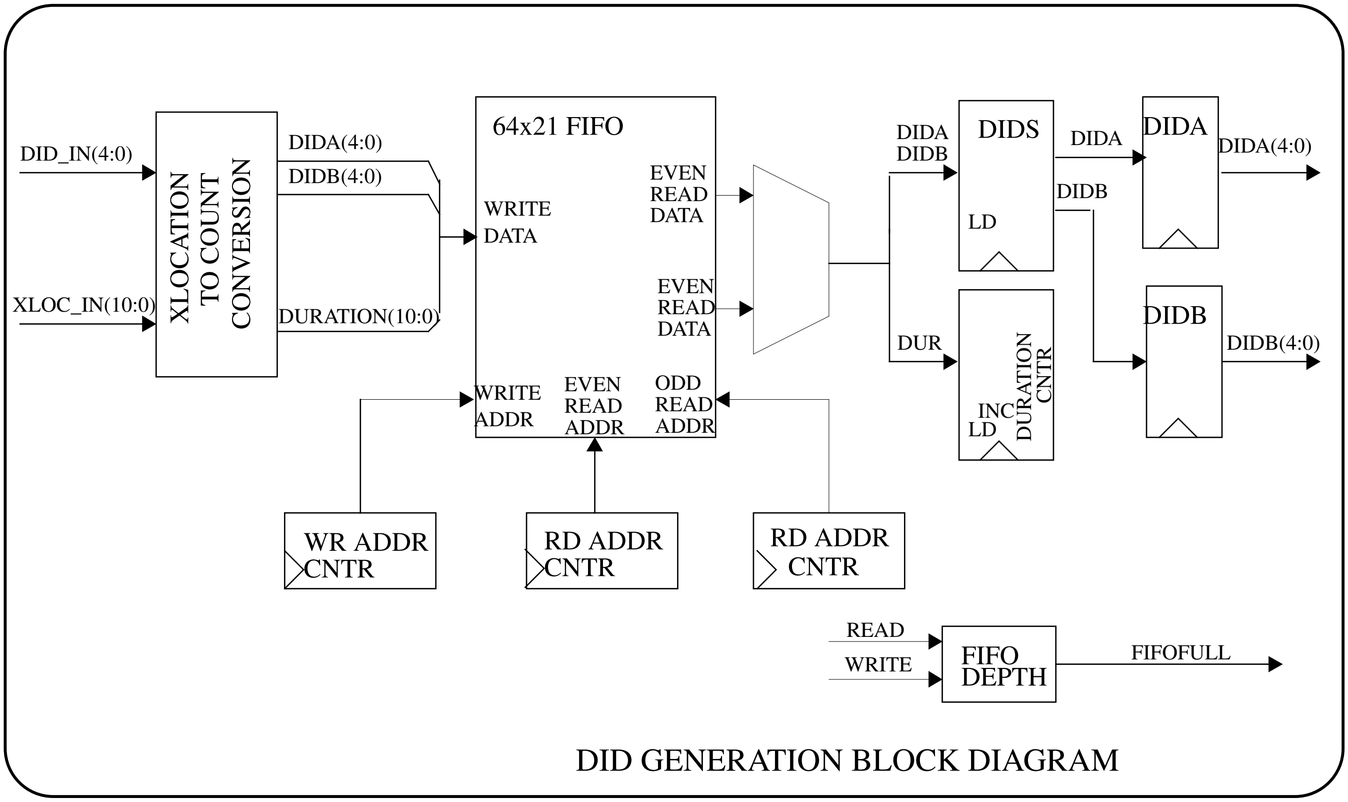
CURSOR GENERATION BLOCK DIAGRAM

horizontal blanking and then incremented every clock. When the pixel counter is equal to the cursor X location, cursor generation begins. A 32 to 1 mux selects the correct pixel in each clock cycle. A subsequent mux will select 0's if the cursor is not present on the current line or not enabled. This mux handles 2 other special cases: the first or last pixel when the cursor starts on an odd-pixel boundary, and the crosshair cursor.

5.6 DID Generation [did]

A detailed block diagram for the DID Generation section is shown below:

The main element of the DID Generation is a 64x21 triple-port ram configured as a fifo, which holds the DIDs as they are retreived from external ram. When the fifo is not full, as determined by depth counter, the master controller reads the next DID transition record from external ram and writes into the Conversion Block registers. In the following cycles there may be 1 or 2 writes to the fifo of the converted DID transition record (DIDA/DIDB/Duration). The write address counter is incremented accordingly. When a DID record is read from the fifo, the 10 bits of the DID pair are loaded into a register and the 11 bits of



Duration are loaded into the duration counter. Each subsequent cycle, the counter is decremented. When it reaches 0, the next DID record is read from the fifo and the read address counter is incremented. For timing purposes, additional pipeline registers are used to drive the DID pairs from the chip.

5.6.1 X-Location to Count Conversion [did_conv]

Each DID transition record in external ram contains a DID value and an X pixel location at which that DID will be applied. That DID will be applied until the X pixel location which is specified by the next transition record. Before the DID transition record is entered into the DID fifo, the record must be converted to a pair of DIDs (to apply to the 2 consecutive pixels) and a duration indicating the number of cycles to apply that pair. This pre-conversion is required so that the DID pairs can be sent out at the pixel clock/2 rate.

XLOCN and DIDN refer to a transition record somewhere in the scanline. XLOCN+1 and DIDN+1 refer to the next transition record. When each subsequent transition record in the scanline is processed, XLOCN+1 and DIDN+1 become XLOCN and DIDN, respectively and XLOCN+1 and DIDN+1 represent the new transition record. DIDA is the least significant DID in the pixel pair and DIDB is the most significant. The basic algorithm is to subtract XLOCN from XLOCN+1 to obtain the duration value for DIDN, however a deviation is required when either Xlocation value is an odd number. Consider the following cases:

•4• XLOCN and XLOCN+1 are both even. This is the simplest case. The duration is the difference (XLOCN+1 - XLOCN) divided by 2, and DIDN is used for both DIDA and DIDB.

•5• XLOCN is even and XLOCN+1 is odd. In this case two fifo entries will be needed. For the first, the duration is the difference (XLOCN+1 - XLOCN) divided by 2, truncating the remainder, and DIDN is used for both DIDA and DIDB. The second entry will have a duration of one where DIDA gets DIDN and DIDB gets DIDN+1. Since this entry has actually included the first pixel specified by transition record N+1, XLOCN+1 will be incremented before being used to process the next transition record. Thus, the next time around, the subtrahend (now XLOCN) will be even. Thus it is not necessary to consider the case of XLOCN being odd.

•6• XLOCN is equal to XLOCN+1 (and both are even). This case can happen only if the previous transition record was case #2 and XLOCN and XLOCN+1 were 1 pixel apart (before the case 2 incrementing operation). In this situation, the previous case #2 has all ready accounted for the 1-pixel duration DID and no fifo entry is required.

The following example will illustrate this further. The DID scanline table contains the following entries:

X-Location	DID	
0	did1	From the start of the scanline, did1 is used for the first 4 pixels
4	did2	did2 is used for the next 3 pixels
7	did3	did3 is used for 1 pixel
8	did4	did4 is used for the remainder of the scanline
1280	xxx	

This translates to the following fifo entries. (Remember that duration refers to the number of pixel clock/2 cycles).

DIDA	DIDB	Duration	
did1	did1	2	Apply did1 for 4 pixels
did2	did2	1	Apply did2 for 2 pixels
did2	did3	1	Apply did2 for 1 pixel and did3 for 1 pixel
did4	did4	636	Apply did4 for 636 pixels

The conversion procedure is as follows:

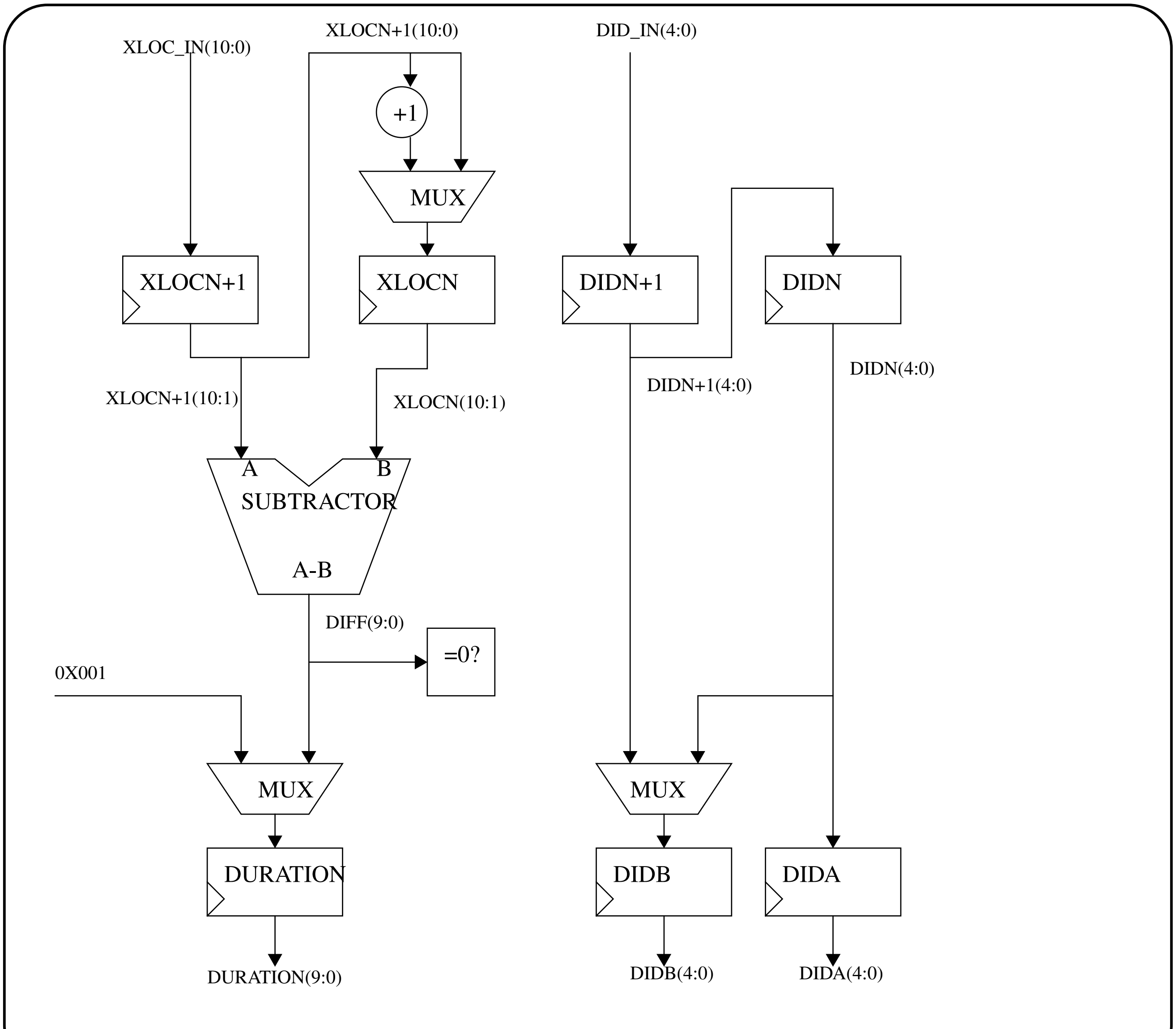
•1• XLOCN=0 and XLOCN+1=4. Both are even so this is case #1. The duration = (XLOCN+1 - XLOCN)/2 = 2. Both DIDA and DIDB get DIDN (did1).

•2• XLOCN=4 and XLOCN+1 = 7. XLOCN+1 is odd so this is case #2. The duration for the first fifo entry = (XLOCN+1 - XLOCN)/2 = 1. Both DIDA and DIDB get DIDN(did2). The next entry will have a duration of 1 with DIDA = DIDN (did2) and DIDB = DIDN+1 (did3). XLOCN+1 is incremented to 8, for processing the next record.

•3• XLOCN=8 and XLOCN+1=8. Since these are equal, case #3 applies and no operation is performed.

•4• XLOCN=8 and XLOCN+1=1280. This is the last record for this line and is also case#1. The duration = (XLOCN+1 - XLOCN)/2 = 636. Both DIDA and DIDB get DIDN (did4)

.



DID X-LOCATION TO COUNT CONVERSION BLOCK DIAGRAM

The input to the Conversion block consists of 2 sets of pipeline registers. XLOCN and DIDN contain the X-Location and DID value for DID transition record “N”; XLOCN+1 and DIDN+1 contain the X-Location and DID for the next transition record, “N+1”. The Master Controller retrieves a DID transition record from external Ram and then writes it into the XLOCN+1 and DIDN+1 registers. When these registers are loaded, their previous contents are written into the XLOCN and DIDN registers. The loadidng of the N+1 registers triggers the conversion process.

Each transition record takes 2 cycles to process, and may result in 0, 1 or 2 entries in the fifo, as described above. During the first cycle after XLOCN+1 and DIDN+1 are loaded with a new record, a subtraction occurs (XLOCN+1(10:1) - XLOCN(10:1)). The difference is loaded into the Duration register and DIDN is loaded into both DIDA and DIDB registers. A fifo write is done if the difference is not zero. During the second cycle, the Duration register is loaded with 0x001. DIDA is loaded with DIDN and DIDB is loaded with DIDN+1. A fifo write is done if XLOCN+1(0) =1.

5.7 Summary of Video Timing Channels

Following is a summary of internal Video Timing Channels and a description of how they are used. For the purpose of this description, references to the visible portion of the line or frame refer to the valid cursor and did output of the VC2. Timing relatuonships refer to the external video timing channels. The internal version of the timing channel is actually 1 pclk/2 cycle earlier.

EOF_VC - End of field/frame. The assertion of this signal triggers DID end of field/frame processing, and also flushes the DID fifo. EOF_VC should optimally be asserted at the start of the first line of vertical blanking and remain asserted for the entire line.

VPOS_VC - Cursor Vertical Position. The assertion of VPOS is used to indicate the Y=0 position of the cursor, which inititates cursor end of field/frame processing and resets the vertical line counter. For non-interlaced modes, VPOS is asserted 32 lines before the first visible line of the frame. For interlaced formats, VPOS is asserted 17 or 16 lines, respectively, before the first visible line of the even or odd field. Optimally, VPOS should be asserted for the duration of the line.

HPOS_VC - Cursor Horizontal position. The assertion of HPOS is used to indicate the X=0 position of the cursor; the deassertion indicates the end of the visible portion of the line for cursor output, which triggers end of line cursor processing. HPOS will be asserted once per line starting in the line after VPOS, through the last visible line of the frame. In each line, the assertion of HPOS must precede the first visible pixel cursor by 31 pixels (16 pclk/2 cycles) plus 4 pclk/2 cycles of pipe delay (a total of 20 pclk/2 cycles). Optimally, HPOS should be deasserted immediately after the last visible cursor pixel.

For example: for a 1280x1024 non-interlaced monitor, there will be 1 blanked line in which VPOS is asserted, followed by 31 blanked lines in which HPOS is asserted, followed by 1024 active lines in which HPOS is asserted.

ODDFIELD_VC - Odd field of an interlaced format. ODDFIELD is used by both cursor and DID processing in order to handle interlaced formats. Must be valid for the upcoming field either before or coincident with the assertions of EOF and VPOS and remain valid for the entire field. Optimally, ODDFIELD should be asserted at the start of the first line of vertical blanking which precedes the oddfield, and remain asserted for the duration of the line. Note: ODDFIELD may have different timing requirements when used with a stereo monitor to indicate right/left eye.

VIS_LN_VC - Visible Line. Visible line indicates the active portion of the frame. It is essentially the inverse of composite blanking. VIS_LN is used by the DID generation section, so that DIDs are driven only during the visible portion of the frame. VIS_LN will be asserted once per line in every line of the visible portion of the frame. There is 1 pclk/2 cycle delay between VIS_LN and the DID outputs of the chip. Thus, VIS_LN must be asserted 1 pclk/2 cycle before the first visible DID pixel and deasserted 1 pclk/2 cycle before the last visible DID pixel.

6.0 Summary of Functional Changes from VC1

Following are the changes from VC1 to VC2 which affect the programmer's interface:

- •1• External Ram size (for video timing table, cursor pattern and DID's) has been increased to 64K bytes, organized as 32Kx16. Due to the 16 bit datapath, all entries in the tables will be on 16-bit rather than byte boundaries. Thus, there will be a small modification to the video timing table to accommodate this. The change does not affect the overall method for programming the video timing. No changes are required for the cursor pattern table.
- •2• Host accessible registers have different addresses and format. All host writes to both control registers and external Ram must be written atomically by the host as a 16-bit entity in big-endian mode. The VC2 assumes that the most significant byte is written first and saves it in a temporary register. When the second (least significant) byte is written, both bytes will be written into the target register (or external Ram).
- •3• Special cursor mode has been eliminated. A 64x64x1 cursor glyph mode has been added.
- •4• The line repeat count from in the DID frame table has been eliminated. All entries in the DID frame table are now line entry pointers. The DID Scanline table uses an end of line flag rather than a word count. DID generation will automatically stop at the end of the visible line, regardless of the number of entries in the DID scanline table. The DID frame table will use an end of frame flag to indicate the end of the frame table, rather than a pointer to the end of the table.
- •5• Video Timing, DID and cursor generation run at one-half the pixel clock rate, thus the video timing channels have 2-pixel resolution. The maximum clock rate is 70MHz which supports a 1280x1024 monitor at 76Hz.
- •6• Internal counters and registers will accommodate a monitor width of 2046 pixels.
- •7• The Video Timing fifo depth has been increased to 16. All 21 timing channels will be available external to the VC2, although 4 channels (5 in interlaced mode) will be dedicated to internal use.
- •8• The DID fifo depth has been increased to 64 (2-pixels wide). This will provide for a greater number of DID transitions per line.
- •9• The Video Timing frame table will use an end of frame flag to indicate the end of the frame table, rather than an entry indicating the number of lines in the frame.
- •10• The X and Y locations of the cursor will be relative to a point which is 31 pixels to the top and left of the visible portion of the screen, for all monitor formats.

7.0 Performance

The two measures of VC2 performance are the number of DID transitions per line and the update rate of the DID table. These numbers are difficult to specify accurately since they are dependent on many factors. Currently, the VC2 is expected to handle at least 100 DID transitions (evenly spaced) per line, while the host is updating the table. However, a worst case spacing of DID transitions could cause the actual number to be lower. Assuming a typical DID table size of 32K bytes, the VC2 can update the DID table once per frame. To achieve this requires about 60% of the Display Control Bus bandwidth in high resolution modes and 100% of the bus bandwidth in low resolution modes.