VIPeR

Video Image Processor

Data Book

GRAPHICS BY

TSENG

LABS

A

Contents

1.0 Introduction

The Tseng Labs Video Image ProcEssoR delivers the necessary features to transform a graphics adapter into a fully functional video adapter. The VIPeR was designed to enhance both live video streams from a source such as a commercial camcorder or VCR, or digital motion video retrieved from a hardware or software compression/decompression engine (CODEC) such as Intel Indeo™ or Microsoft Video for Windows™ .AVI files. The VIPeR intelligently sizes and scales data to make it appear larger or smaller on screen without decreasing CPU bandwidth. The VIPeR architecture provides numerous benefits to the designer, including optimal use of the Image Memory Access (IMA)™ port featured on the Tseng Labs W32 family of graphics accelerator chips.

The VIPeR allows graphics adapters based on the W32 family of accelerators from Tseng Labs to truly become video cards. By definition these video cards provide high-quality frame grabbing from any video source and apply the VIPeR's sizing and scaling to software-decompressed video, such as Microsoft's Video for Windows.

Today's CPUs are capable of 15/30 frame per second decompression of some algorithms. Typically the default resolution of CPU decompressed video is no greater than 160x120 pixels. Since many GUI users use high resolution displays, the video will need to be enlarged to occupy a significant screen area. The CPU does not have adequate bandwidth to increase resolutions. When executing both decompression and expansion, system performance deteriorates dramatically. Quality is also an issue, as CPU-based enlargement creates jagged un-lifelike images, while the VIPeR generates enlarged images smoothly without absorbing CPU bandwidth or dropping the frame rate of the motion stream. As a result the VIPeR is ideal for audio/video communication across a personal computer network, TV-in-a-window, digital authoring, video telecommunications, and other multimedia applications.

The VIPeR video processor from Tseng Labs allows users to place a high quality digital video window any where on their display. By leveraging the capabilities of the Tseng Labs W32 family of graphics accelerators, including the IMA port and CRTCB feature, the VIPeR can be programmed to display 16-bit or 24-bit full motion video at an arbitrary size on the personal computers display monitor. The VIPeR controls the digitization of video and performs digital processing on motion video data. Under an operating environment like Microsoft Windows, an application will allow the user to drag open a video window of any size from 16 by 16 to 720x480, when used with the Tseng Labs W32.

When using a decompression CODEC, like an MPEG decoder, for example, video is of a fixed output resolution and format. It can't easily be resized or moved to arbitrary locations on the display. With the VIPeR, this output decoded video stream can be resized to any arbitrary screen size desired, and placed in any screen location. Further, with the video/graphics mixing capabilities of the VIPeR/W32 graphics controller combination, the video and graphics windows can occlude and mix with each other on a pixel by pixel basis. Finally, the scaling and windowing is independent of compression/decompression algorithm used.

The VIPeR performs a number of control functions for the digitization of video. Using it's bullet-proof sync technology, it provides the clock for the external Analog to Digital converters, separates the composite sync into horizontal sync, vertical sync, and framing information. This process delivers enhanced synchronization separation, which allows for worst-case video sync input, such as fast-forward, reverse, and freeze-frame in consumer VCR's to display properly on screen.

The VIPeR uses all of the information available in each field to create a video window. In creating the digital display, the VIPeR removes various unfavorable affects of converting the interlaced video to non-interlaced VGA through digital processing. The digital processor is capable of both reduction and enlargement of the incoming video data, without recalculation from the CPU, by converting the video stream to the requested X and Y dimensions. A sample clock of approximately 15 megahertz will allow up to 20 pixels in the X dimension. With a one-line internal FIFO the VIPeR is capable of converting the incoming video lines up to 480 lines per field for NTSC; 576 for PAL.

Furthermore, the VIPeR expands CPU bit maps or motion video as if they had entered the VIPeR through the Analog to Digital port, delivering real-time enlargement of lower resolution stored video sequences. Because the enlargement uses the VIPeR's processing power and is not the result of simple pixel replication, the quality of the image is optimal.

VIPeR performs smooth scaling, based on an area average algorithm, in both the horizontal and vertical directions. All the input data is used to create the output video stream. No pixel dropping or pixel replication is done. No line dropping or line replication is done.

The CPU communicates with the VIPeR through memory mapped bus. Using the Tseng Labs Image Memory Access (IMA) connector, register reads and writes, as well as deliverance of unscaled pixel data can be done. In ISA or MicroChannel configurations, VIPeR has a 16 bit host bus connection via a buffered data bus it shares with the W32/W32i/W32p. In PCI or VESA Local bus configurations, VIPeR's host bus connection is an 8 bit bi-directional, address/data multiplexed bus passed directly from the W32x. It is called the DD bus. All host communications to the VIPeR are memory mapped by the W32x, and repeated over the DD bus. In this way, the VIPeR does not contribute an electrical load to the host CPU bus, and PCI 2.0 compliance is maintained.

There are two main mechanisms for delivering (unscaled) video data to the VIPeR. The first is via the CPU interface bus, and is generally used when playing back data from the host computer. This is the same bus which is used to program the VIPeR registers, and is a part of the Image Memory Access (IMA) connector. The second is via the Analog to Digital Port, and is generally used when playing back live digitized data. Decompression chip outputs can feed their data to either input port, however, the Analog to Digital Port will generally be easier for the system board design.

There are two main mechanisms for outputting scaled video from the VIPeR. The first is over the Image Data Port, the other part of the Image Memory Access (IMA) connector. This is an 8 bit serial data port which feeds video data to the W32/W32i/ W32p graphics controller for storage in the frame buffer. The second is over the processed digital data output bus. This is a 24 bit (red/green/blue) output which is the direct scaled video output. This video stream can be input to a compression engine for encoding scaled video streams.

The VIPeR output video data is written to the W32/W32i/W32p byte serially over the Image Data Port. When the graphics controller receives the VIPeR's video data, it immediately writes it into pre-programmed regions of its own frame buffer. Therefore, adding VIPeR video functionality does NOT require the system designer to add any additional frame buffer memory. The VIPeR/W32x system uses a single frame buffer architecture. Video memory space is allocated out of the existing graphics frame buffer. This leads to large system cost reductions.

VIPeR contains an 8 bit bi-directional Video Control (VC) Port. Data into and out of these pins is accessible by the host CPU via internal registers in VIPeR which source or read the values on the VCPort pins. This capability allows for CPU access to other devices on the video card, through the VIPeR, through W32x memory mapped access to this internal VC port register. These VC port bits can be used to control and read an IIC bus, for example.

In conjunction with the W32/W32i/W32p, VIPeR provides an alpha channel bit along with each video pixel to allow for pixel by pixel mixing of video and graphics in the display frame buffer. The video/graphics "flag" is stored in an external 256Kx4 DRAM, which the VIPeR controls. Writes to this alpha bit (mask) DRAM are done via memory mapped register access, similar to all other VIPeR register accesses. VIPeR controls reading this mask information and synchronizing it with the scaled video output stream. The W32x reads the alpha (mask) bit with each video pixel it receives from the VIPeR. The W32x is responsible for writing the appropriate video or graphics pixel in the frame buffer, and out to the display device, based on the corresponding mask bit.

A final VIPeR feature is the ability to accept digital audio data in, and synchronize it with the video. VIPeR contains an internal FIFO for storing audio (or any other relatively lower frequency, like SMPTE) data. There is a pass through

mechanism whereby this data is written, over the Image Data Port, and stored in unused portions of the frame buffer. Although no processing on this data is done, the FIFO and IMA transfer capability potentially simplify lip sync issues of audio/video streams.

VIPeR is highly integrated for use with the Tseng Labs W32 family of graphics accelerators. The video system architect can choose any of the options desired to enhance the VIPeR/W32 video/graphics system. Following diagrams show the major elements of a VIPeR display subsystem.

Figure 1.1 shows a bare bones configuration: W32/W32i/W32p with host bus access, DRAM frame buffer, connection to RAMDAC, and IMA connector for adding VIPeR video capabilities. VIPeR host bus connection is 8 bit DD mode, via the IMA connector to the DD bus of W32. Without a front end A/D converter to accept live video, this system will still be able to read stored data streams from the CPU bus (like Video for Windows .AVI files), scale the video, and display it over the top of the graphics windows.

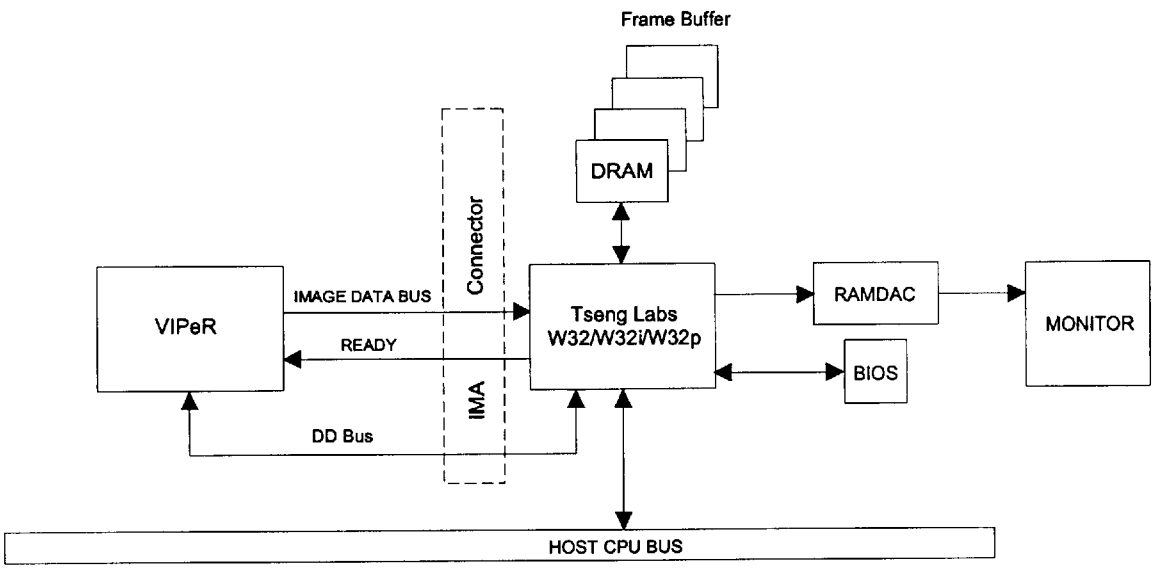


Figure 1.1 DD Bus, Stored Video Playback Only

Figure 1.2 shows a system with the an added A/D front end converting incoming live video to digital format. With this additional component, and connector to live video source, this system will now be able to do what the previous did, and also scale and display live video sources as well. In addition, this system gives the ability to capture (scaled) live video and store it onto hard drive or system memory. In this system, the video window must be always on top of the graphics.

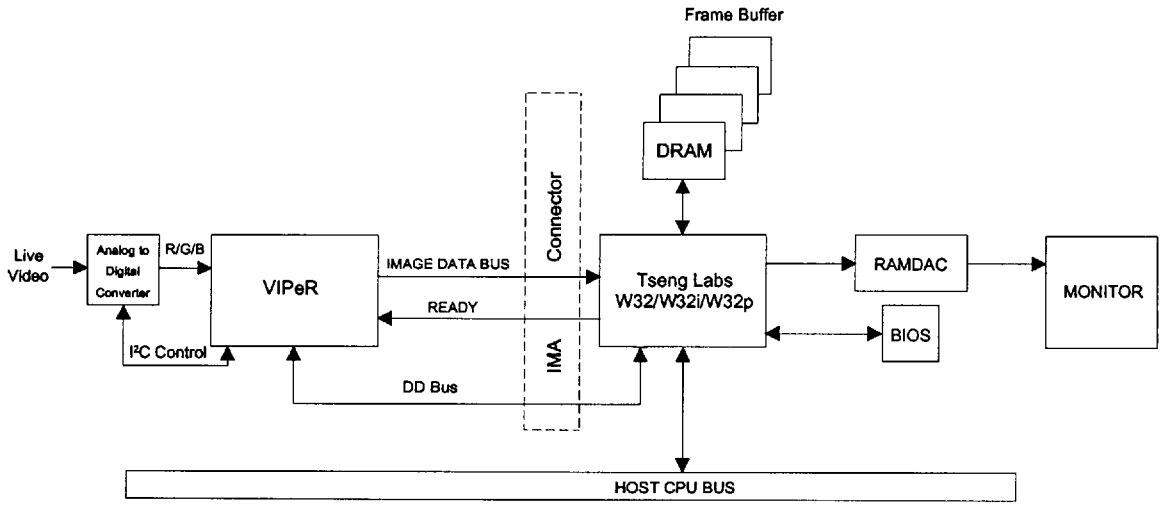


Figure 1.2 DD Bus, Live and Stored Video Playback and Capture

Figure 1.3 shows a system with the addition of a 256K x 4 bit DRAM. With this new component, video can now be mixed with the graphics on a pixel by pixel basis. The video can be occluded by the graphics, not only in rectangular regions (necessary for operating in a full Windows environment) but also on pixel boundaries (useful for authoring systems with text overlays on top of the video).

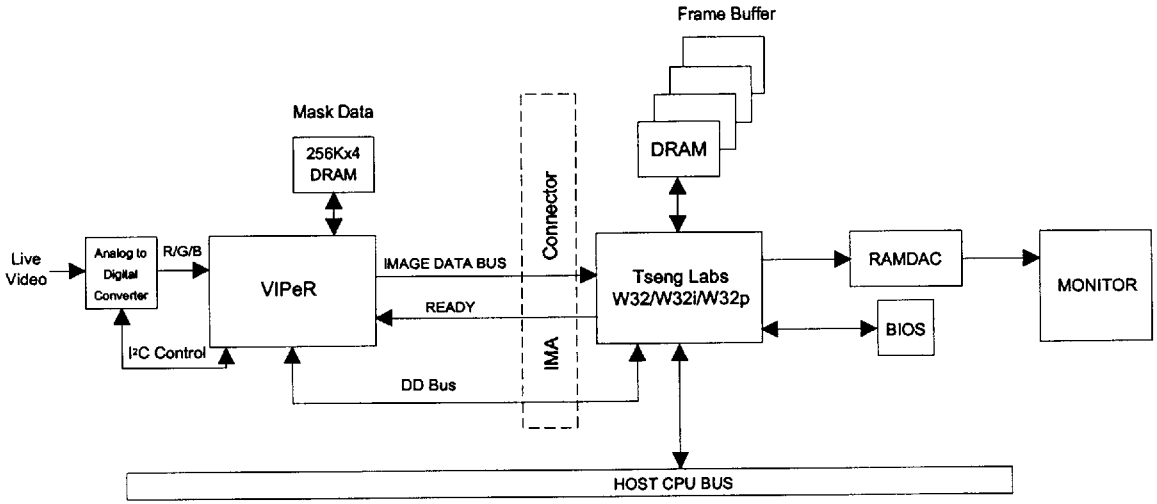


Figure 1.3 DD Bus: Video Playback, Capture, Store; Fully Windows Compatible

Figure 1.4 shows a system shows the above system but with VIPeR connected to the host bus in 16-bit BDB mode. This requires buffering the host bus.

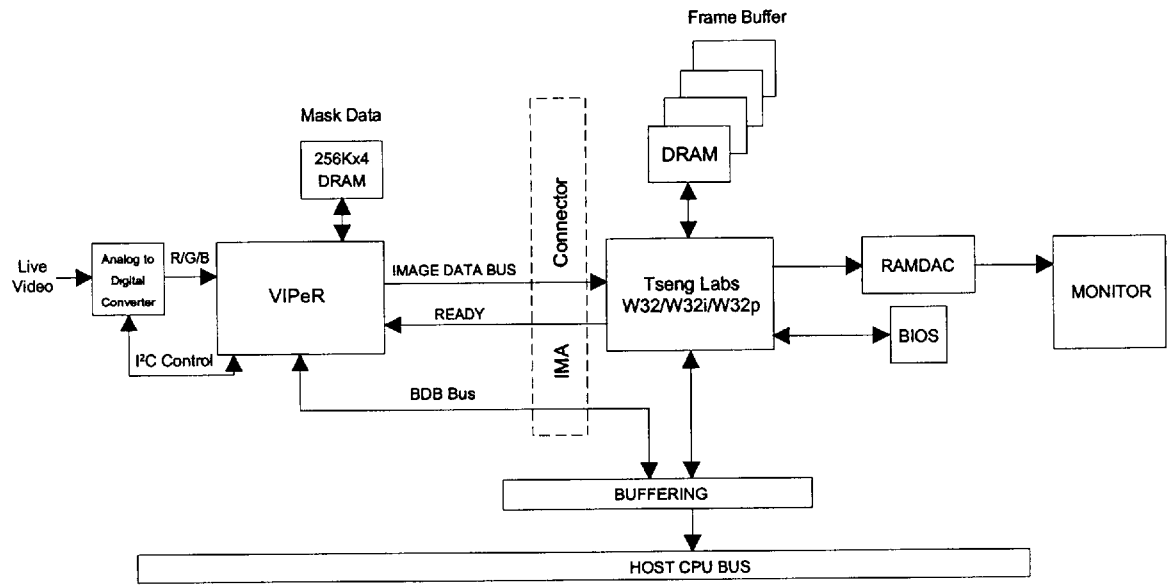
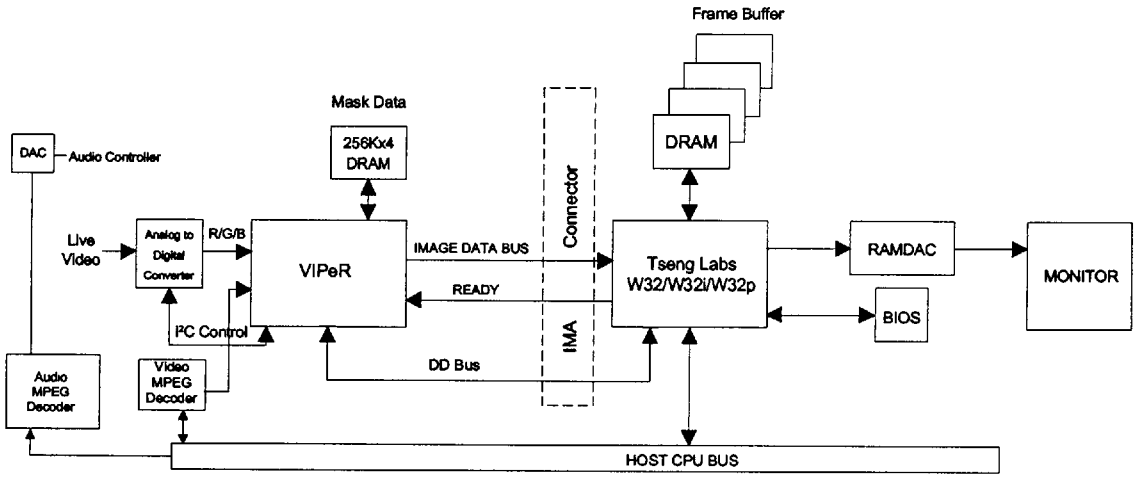


Figure 1.4 BDB Bus, Video Playback, Capture, Store; Fully Windows Compatible

Figure 1.5 shows the DD bus mode again, but with the addition of an MPEG decoder. With this additional component, and its associated memory, the video system now supports not only live and stored video playback and capture, fully compatible with Windows, but now can accept MPEG encoded video data streams, decode, scale, and mix video with the graphics window



NOTE: MPEG decoder programming input data stream can be via DD Bus or Host CPU Bus connection

Figure 1.5 DD Bus, Video Playback, Capture, Store; MPEG Decode and Scale; Fully Windows Compatible

Figure 1.6 shows the addition of a video encoder. With this additional component, one can now encode live video streams and store them to the hard disk or system memory.

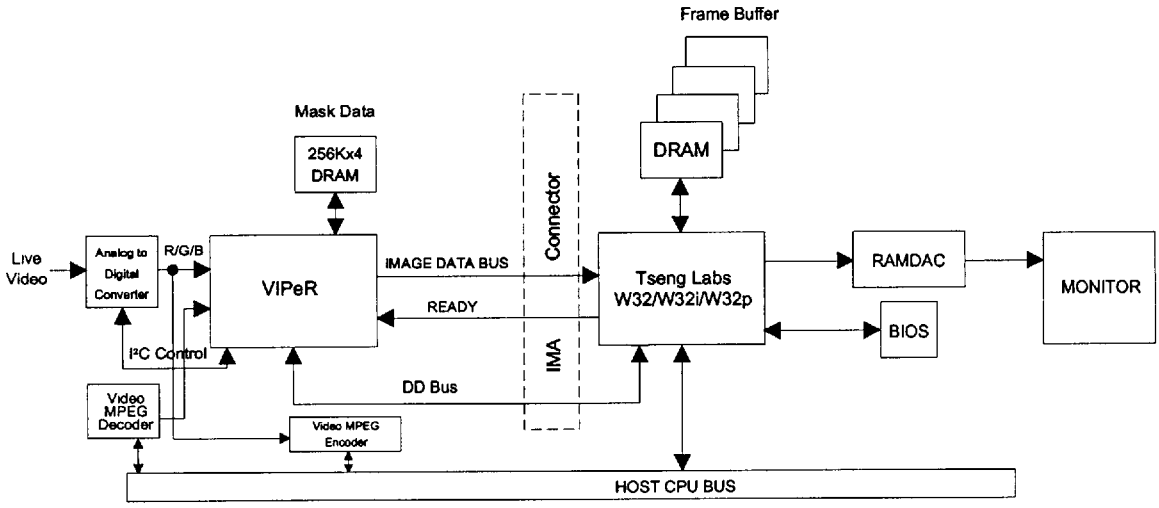


Figure 1.6 DD Bus: Video Playback, Capture, Store: MPEG Decode/Encode and Scale: Fully Windows Compatible

Figure 1.7 shows the addition of an audio receiver/digitizer. Adding this component allows for audio capture in addition to all the video features.

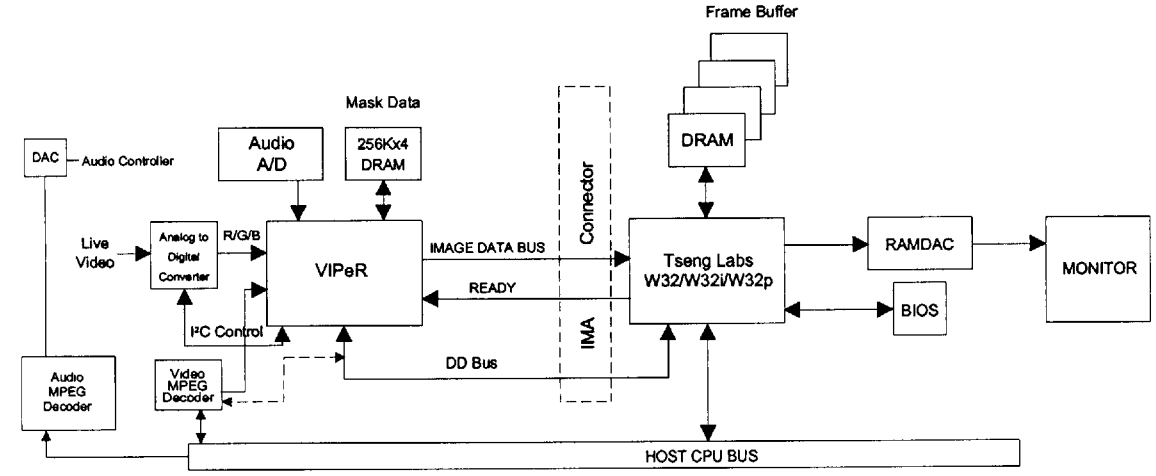


Figure 1.7 DD Bus: Video Playback, Capture, Store: MPEG Decode/Encode and Scale: Fully Windows Compatible, Audio Capture

1.1 ET4000/V33 (VIPeR) Specifications

The following is an outline of the ET4000/V33 Video Image ProcEssoR specifications.

1. Input Interface

Following are the primary VIPeR input busses:

A. Host access port, BDB or DD, bus
   1. 8 or 16-bit memory mapped I/O bus
   2. Multiplexed address/data
   3. 16 bits (BDB<15:0>) in ISA, MicroChannel bus modes, 8 bits (DD<7:0>) in Local bus or PCI modes
   4. Driven by W32x in DD bus mode; driven by buffered data bus in ISA mode
   5. Register I/O and digital video input
   6. Maximum 33MHz, 16.5 Mbyte/second via DD bus; maximum 33MHz, 33Mbyte/second via BDB bus

B. ADC interface
   1. 24-bit digital video input only
   2. 25MHz maximum data input rate
   3. Accepts composite or separated sync inputs

C. Audio interface
   1. 8 bit digital data input
   2. Data FIFO'd internally, sent out at end of video scan line

D. Video Control Port
   1. 8 bit I/O bus
   2. Each bit independently set to input or output
   3. Information on pins reflected in software accessible internal VIPeR register

2. Output Interface

Following are the primary VIPeR output busses:

A. Image Port Output
   1. 8 bit data output bus: scaled video sent to W32x
   2. Address implied: internally generated via W32x programming
   3. Tseng IMA protocol
   4. Maximum 50MHz, 50Mbyte/sec output

B. Processed Data Output
   1. 24-bit digital scaled video (tristateable) output
   2. Video output only: result of X/Y scaling, can be sent to devices besides W32x
   3. Maximum 50MHz, 150Mbyte/sec output

3. Display Formats
   A. Plane, linear byte, linear word
   B. Resolution from 16x16 to 720x480 (580 for PAL)
   C. 15/16/24 bit/pixel
   D. RGB or YUV pixel formats

4. Inputting Source Video to VIPeR
   - A. ADC/Host access
     1. Data accepted via either ADC or Host bus
     2. Interlaced or non-interlaced video accepted
   - B. Pixel Formats: RGB, YUV444, YUV422A, YUV422B
     1. 15/16/24 bit RGB
     2. YUV444
     3. YUV422 modes: UV nibbles, or toggling UV bytes
   - C. Two Sync Sources
     1. Composite sync input
     2. Independent horizontal/vertical/ sync, field polarity

5. Outputting Scaled Video from VIPeR

Video stored in shared graphics frame buffer—No need for dedicated video frame buffer or video/graphics switching logic.

15/16/24 bit RGB output formats available

- A. IMA output
  1. Delivered over 8 bit IMA bus, via Tseng IMA protocol, to W32x, for storage in graphics frame buffer
- B. Processed data output
  1. 24-bit scaled RGB video output

6. Video Scaling
   - A. Horizontal Scaling
     1. Smooth scaling: uses ALL incoming data, no pixel replication or pixel dropping
     2. Scale up/Scale down: maximum input/output ratios, 1/64 or 64/1 limited by minimum video 16x16 and maximum video resolution 720x1024
   - B. Vertical Scaling
     1. Smooth scaling: uses ALL incoming data, no line replication or line dropping
     2. Scale up/Scale down: maximum input/output ratios, 1/64 or 64/1 limited by minimum video 16x16 and maximum video resolution 720x1024

7. YUV/RGB Conversion
   - A. CCIR601 compatible YUV to RGB conversion done on fly
   - B. Selectable: perform conversion or bypass conversion

8. Sync Separation
   - A. Lock to multiple video sources
     1. Program windows for anticipated sync pulse timing genlock to high or consumer quality sources
     2. Composite sync or independent line/frame/field inputs
   - B. Scan conversion
     1. Converts from broadcast interlaced to high refresh rate non-interlaced

9. Video/Graphics Overlay Control
   - A. Motion video selectable on pixel-by-pixel basis
     1. Control logic and I/O pins for control of external 256Kx4-bit DRAM. DRAM is programmed (via VIPeR registers) with video/graphics select data

2. Two mask formats available: one 1024x1024, or two 1024x512 resolution video overlay windows
3. Pixel-by-pixel overlay control allows for graphics/video window including, non-rectangular shaped video windows, and pixel resolution graphics text overlay

10. Video Control Port
    A. Read/Write external devices via program I/O port
        1. Programmed register sets input/output functionality of each pin.
        2. Register writes to output configured pins pass corresponding register data to output VC port pin.
        3. Logic states on wires hooked to input configured pins are readable via corresponding register bit reads.
    B. Useful for attaching to and controlling IIC (or similar) busses
    C. Gives software accessibility to external video board hardware

11. Audio Input
    A. Accepts audio (or other) relatively slower digital data
        1. 8 bits data, 1 clock input loads internal 128 byte FIFO
    B. Writes data out to unused portions of frame buffer
        1. Programmed registers set maximum number of bytes to be appended to end of video scan line.

1.2 VIPeR Overview

- Single 160-pin Quad Plastic Flat Pack Integrated Circuit.
- Real-time dynamic enlargement and reduction of motion data streams using a proprietary imaging algorithm. This intelligent processing eliminates jagged edges caused by pixel multiplication, or significant data loss of pixel decimation.
- Smooth scaling is performed in both X and Y directions.
- Interface to the ISA/EISA/MCA, as well as VESA VL-Bus and Intel PCI architecture.
- Directly interfaces to the Tseng Labs W32/W32i/W32p Image Memory Access port without external logic.
- Accepts input data in 15/16-bit and 24-bit RGB, as well as 4:4:4 and sub-sampled 4:2:2 YUV formats.
- Outputs 16-bit and 24-bit RGB, ideal for Microsoft Windows and other personal computer graphics environments.
- IMA/VIPeR architecture creates a single frame buffer that includes both the graphics display and the motion video.
- Accepts either live input or stored/decompressed digital data from the host bus.
- Refresh rate: selectable, 30 frames per second, 60 fields per second, or 30 fields per second.
- Digitization Resolution: up to 720 x 480 (580 for PAL) by 24-bit (dynamic).
- Screen resolution from 16x16 (icon) to 720 x 480 (580 for PAL) by 16 bits or 24 bits of color.
- High quality still frame capture.
- Bullet-proof sync separation will lock to all consumer grade and better NTSC/PAL/SVHS video sources.
- Handles scan conversion from interlaced broadcast standards to high-refresh non-interlaced
- Provides pins to control the storage of digitized audio data in unused areas of the frame buffer.
- Hardware cursor overlay for the video window when using the W32 family's CRTCB feature.
- Requires no external VRAM or DRAM for digitization. Control signals are provided for one optional external 256Kbx4 DRAM chip for overlays. The VIPeR can use this buffer to support a mask bit to display motion data on a pixel-by-pixel basis.
- Supports hardware cropping.
- Memory mapped registers are fully programmable in parallel.
- Compatible with Microsoft Windowsä and Video for Windowsä, IBM OS/2ä, and Intel Indeoä software.

2.0 VIPeR Functional Description

All major elements of the VIPeR are contained within a 160 pin Quad Flat Package. Figure 2.0.1 below shows the internal architecture. The following sections provide a breakdown of the major elements of the chip.

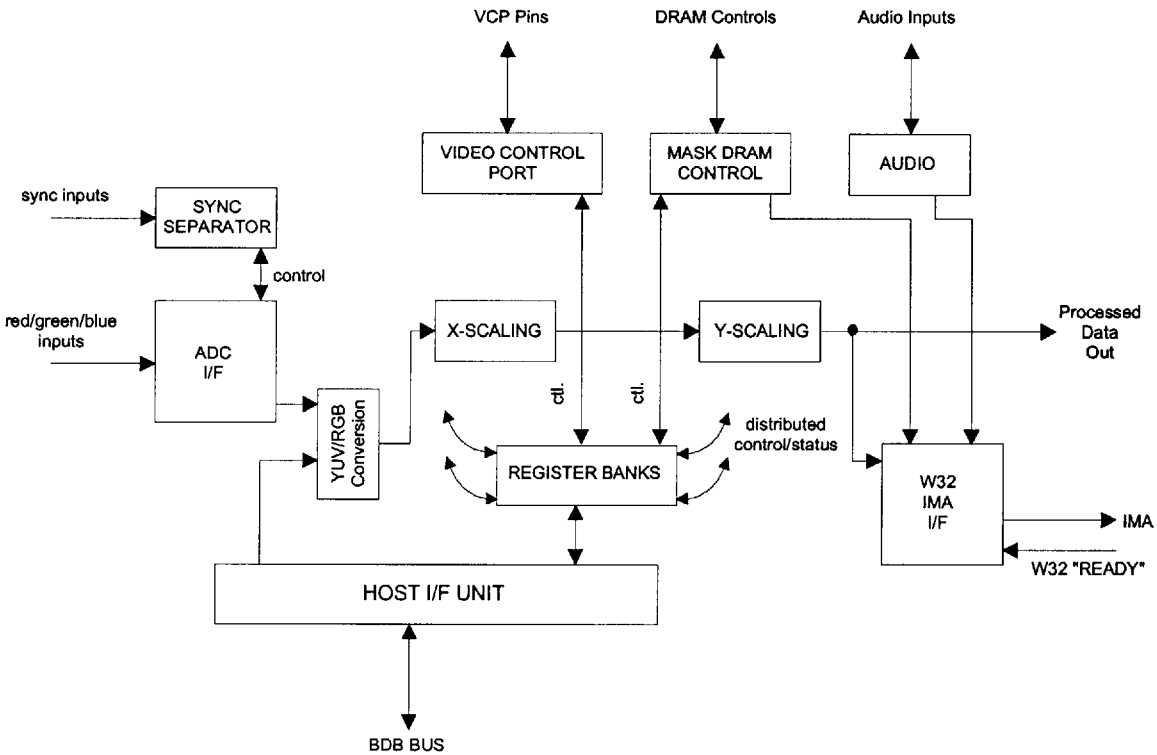


Figure 2.01 VIPeR Internal Architecture

2.0.1 Image Memory Access (IMA) Connector

Tseng Labs has devised a 50-pin connector which joins the video image processing functions with the graphics display functions. By using this connector, OEMs can add motion video enhancing daughter cards to either their system planer, or video adapter. The IMA connector includes 8 bits of Image Data (output from the VIPeR), control and synchronization signals, the buffered data bus, and power/ground wires. An IMA connector is not required if direct connection within a single PC board is designed, linking the VIPeR and graphics accelerator directly.

The Tseng Labs family of graphics accelerators control the resolution and color depth of the display monitor, as well as provide the memory for the motion video, within its graphics frame buffer. VIPeR video output travels over the IMA bus into the graphics accelerator, which then stores it at a CPU-definable address in the frame buffer.

The W32/W32i/W32p provides two separate and independent CRT control windows, CRTC and CRTCB, known also as the A and B windows respectively. The primary CRTC is used to display the personal computer's primary graphics session, including such things as Microsoft Windows, or DOS. The second, CRTCB, (B window), can be a hardware cursor, or it can be an independent overlay window, ideal for live video. Using the W32/W32i/W32p, the motion video window can be at different color depth than the primary window. For example, the graphics can be in 8bpp mode, while the video window is 16bpp. Other combinations are also possible. For more information, contact your Tseng Labs representative for W32 family graphics accelerator information.

2.0.2 Buffered Data Bus

VIPeR and W32/W32i/W32p share access to the buffered data bus. As a result, the VIPeR is accessible to the host CPU and can be configured, reprogrammed, and status monitored by the programmer. Additionally, data decompressed by the CPU or external CODEC on the bus, intended for display, can be routed to the VIPeR across this bus for real time processing. This processing relieves the CPU and CPU bus from having to scale, and YUV/RGB convert, live, full resolution motion video.

There are two main operating modes of the host buffered data bus communication with the VIPeR and W32/W32i/W32p. These are called, 16-bit buffered data bus mode, and 8-bit DD bus mode. Choice of these depends on the W32/W32i/W32p configuration. In ISA and MCA modes, 16 bits of BDB are available from the graphics accelerator, so the full 16 BDB mode can be chosen for host to VIPeR communication. In PCI and VESA local host bus modes, however, only 8 bits of host bus are available for VIPeR communication. These 8 bits are directed from the W32/W32i/W32p directly. They are the remapped address/data as seen on the host bus, when an access to the graphics accelerator external device address space is detected. In 8-bit DD bus mode, host to VIPeR accesses are still 16 bits, however, the access is done in two passes: low byte/high byte, occupying four DD bus cycles as follows.

2.0.2.1 DD Bus Mode Writes

DD Bus Cycle #	DD<7:0>
1	low address byte
2	low data byte
3	high address byte
4	high data byte

2.0.2.2 DD Bus Mode Reads

DD Bus Cycle #	DD<7:0>
1	low address byte
2	NULL cycle
3	high address byte
4	VIPeR readback data byte

NOTE: A different read mechanism is available with Rev.B VIPeR only. See the PREREAD mode description in Section 5.0, Register Descriptions, Control Register, bit 3.

In 16 bit BDB bus mode, host to VIPeR accesses are 16 bit, direct, as follows.

2.0.2.3 BDB bus mode writes

BDB Bus Cycle #	BDB<15:0>
1	address word
2	data word

2.0.2.4 BDB bus mode reads

BDB Bus Cycle #	BDB<15:0>
1	address word
2	VIPeR readback data word

NOTE: unless specified as VIPeR driven during corresponding bus cycles in read modes above, all host bus activity is driven by W32/W32i/W32p.

2.0.3 Operation

When the external RESET signal is activated, writable VIPeR control and configurations register bits return to zero(*). VIPeR is now ready to be initialized.

NOTE: there are 2 exceptions to this reset to zero state. Register bits 7 and 0, CPUSYNC, and READHI, described in detail in the Register Programming Section 5.0, reset to value 1.

Initialization takes place, after RESET has returned to its inactive state, by a sequence of register writes to the VIPeR control and configuration registers. A full description of these is given in Section 5.0. The order in which these registers are written is unimportant.

VIPeR registers are accessed by the host CPU via memory (re)mapped accesses to the Tseng Labs graphics accelerator, W32/W32i/W32p. The graphics accelerator has a 4K external device address space starting at host address 0xBE000. Host accesses to this address space are remapped by the graphics accelerator to the VIPeR. The VIPeR configuration and status registers reside in the 0x00 through 0x38 address offset of the external device address space.

After initial programming, VIPeR remains in a state where it is waiting for video data. Based on programmed state, this video can come from either the ADC inputs or from continued host writes to offset 0x400 through 0x7FF addresses within the external device address space. Playback of CPU stored video will typically arrive at the VIPeR as series of host writes. Video synchronization information also arrives via register writes, in this case, to bits in the Control Register, address 0x00. For live video inputs, or other input (like from a video decompression CODEC), video can come into the VIPeR, digitally, via the ADC input. Along with the data, a clock, LLCLKIN, is provided to sample the incoming video. VIPeR accepts either 15, 16, or 24 bit/pixel video inputs in either RGB, YUV444, or YUV422 modes.

Since the video into VIPeR is digital, an analog to digital converter must be used when the source of video is live, analog video, like NTSC, PAL, or SVHS. Systems focusing on enhancing .AVI or Indeo playback may not require this analog to digital converter front end. In general, if the source of live video is from the CPU only, in a particular system, then the front end A/D component is not needed.

As input video comes into the VIPeR, horizontal and vertical direction scaling, YUV/RGB conversion (if required), formatting, and synchronization with input sync pulses and corresponding mask bit data, take place based on programmed settings. Scaled/converted/synchronized data is FIFO'd and, based on W32/W32i/W32p ability to take the output video data (as communicated on the W32/W32i/W32p output, VIPeR input IXRDY*), is sent, over the Image Data Port portion of the IMA connector, to the W32/W32i/W32p. 15, 16, or 24 bit output data is time multiplexed onto the 8 bit IXDATA<7:0> outputs. W32/W32i/W32p accepts this image video data, along with a mask bit, on IXMSK VIPeR output pin, and writes this video stream into preprogrammed rectangular regions of the main graphics frame buffer. In this way, the W32/W32i/W32p mix the video and graphics together. The mask bit associated with each video pixel tells the W32/W32i/W32p whether video or graphics should be displayed for that particular pixel. In this way, graphics windows can occlude arbitrary portions of the video window. Further, since the mask capability is on a pixel by pixel basis, text overlays are possible over the video window.

2.0.4 VIPeR Device ID (Signature)

As described in the register programming section, 5.0, VIPeR returns a device ID, or signature, when address 0x00 is read. This signature resides in the upper byte, bits <15:8> of control register, address 0x00. For VIPeR, Rev.A, the signature value is 0xBB. For VIPeR Rev.B, the signature is 0xCB. 256 values of signature is sufficient because only devices physically residing in the W32/W32i/W32p extended device address space are locatable here.

2.1 Host Interface

VIPeR and W32/W32i/W32p share access to the buffered data bus. As a result, the VIPeR is accessible to the host CPU and can be configured, reprogrammed, and status monitored by the programmer. Additionally, data decompressed by the CPU or external CODEC on the CPU bus, intended for display, can be routed to the VIPeR across this bus for real time processing.

The host interface bus is a bi-directional, multiplexed address/data, 16 bit connection to the W32/W32i/W32p buffered host bus. It operates in two main modes, based on W32/W32i/W32p configuration. These are, sixteen bit buffered data bus mode (BDB mode) and 8 bit direct data (DD mode). The pin names are BDB<15:0> (the actual address and data pins), MEMW*, the read/write indicator, and IXMAD*, the address/data phase indicator. MEMW* and IXMAD* are driven by W32/W32i/W32p. BDB bus is driven by the buffered host data bus in BDB mode, during non readback data cycles. BDB bus is driven by W32/W32i/W32p in DD mode, during non readback data cycles. BDB bus is driven by the VIPeR during readback data cycles.

When MEMW* is low, host accesses to the VIPeR are writes. When MEMW* is high, VIPeR is being read, and drives the BDB lines after receipt of a valid address.

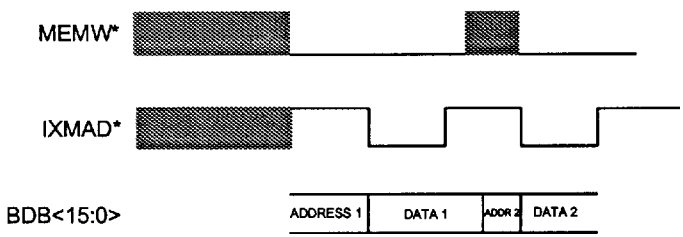
In general, IXMAD* is driven high by W32/W32i/W32p during the address phase on BDB bus. After IXMAD* is driven low, data is driven onto BDB bus for writes to the VIPeR, and VIPeR drives data back when IXMAD* is low during a read access to the VIPeR (as indicated by MEMW*). Timing of these signals can be tuned (VIPeR Rev.B silicon only) for different board design configurations, via zero ohm jumpers to ground, inserted or removed, on PRED<0> and PBLUE<0>. Details of the bus timing is provided in Section 4.1.

When the W32/W32i/W32p is in either PCI or VESA local bus mode, 8 bit DD bus mode must be chosen as the VIPeR connection to the host. This is because the VIPeR/host connection is made via the DD bus of the W32/W32i/W32p. DD bus is 8 bits when in PCI or VESA local bus mode. Since the VIPeR address decode and data access are all 16 bits, a double cycle sequence is done to access the VIPeR from the host. Low byte access is done first, followed immediately by the high byte access. BDB<7:0> is connected to the W32/W32i/W32p DD bus. BDB <15:8> are not used in this bus mode, and can be left floating.

When the W32/W32i/W32p is in either ISA or MicroChannel bus mode, 16 bit BDB bus mode must be chosen as the VIPeR/host connection. The VIPeR 16 bit host bus, BDB<15:0>, is connected directly to the host buffered data bus, not to the W32/W32i/W32p DD bus. Read/write and address/data signals, MEMW*, and IXMAD*, are still driven by the W32/W32i/W32p. All BDB<15:0> bits are used in this bus mode.

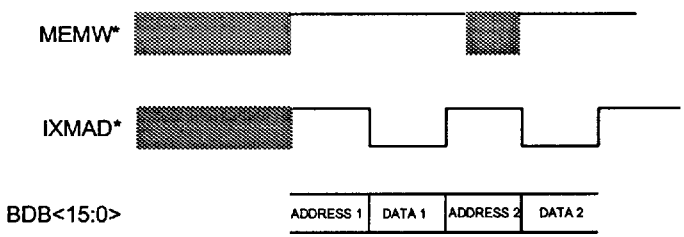
A functional timing diagram showing the register accesses is shown in figure 2.1.1 below. Included are BDB read and write operations. DD mode writes, and DD mode reads in its two read modes: preread, and not preread. These preread modes are discussed in detail in register programming section, 5.0.

BDB Bus Mode WRITE Access



* BDB always driven by host buffered data bus

BDB Bus Mode READ Access



* BDB always driven by VIPeR when DATA 1, Data2;
BDB driven by host buffered data bus otherwise.

Figure 2.1.1 Register Access Functional Timing Diagram

DD Bus Mode WRITE ACCESS

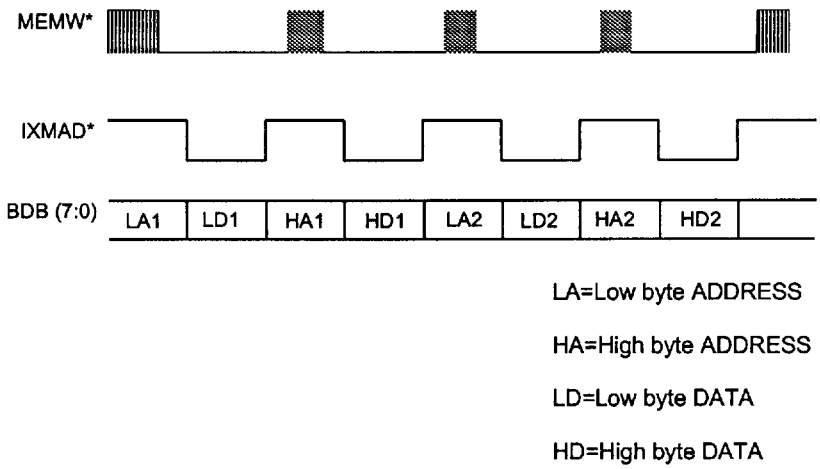
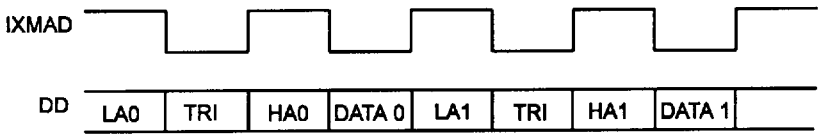


Figure 2.1.1 (continued) Register Access Functional Timing Diagram

DD Bus Mode Read Access (not pre-read mode)



LA= Low byte address (driven by W32p) TRI= Tri-state VIPeR outputs
HA=High byte address (driven by W32p * VIPeR outputs on DD Bus also tri-stated when
DATA=Low Byte of DATA if READHI=0 W32p driving high or low byte addresses
High byte of DATA if READHI-1

DD Bus Mode Pre-Read Mode

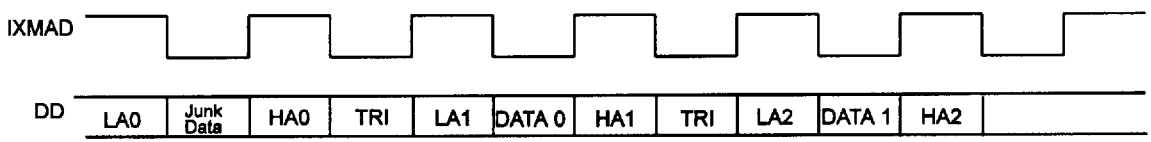
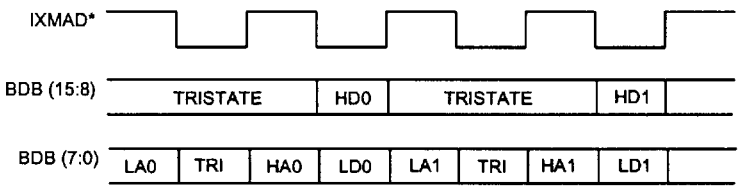


Figure 2.1.1 (continued)



LA0=Low Address 0= Low byte of external device address (w32x driven)

HA0=High Address 0= High byte of external device address (w32x driven)

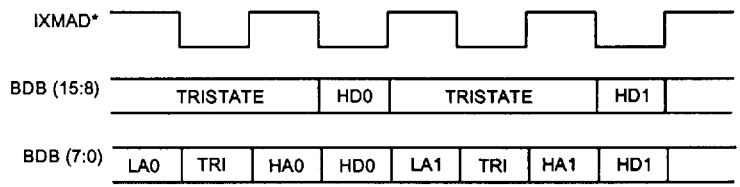
LD0=Low byte register data (VIPeR driven)

HD0=High byte register data (VIPeR driven)

TRI=Tristate

PREREAD = 0
READHI = 0

Figure 2.1.1 (cont'd) VIPeR (DD Bus Mode) Reads



LA0=Low Address 0= Low byte of external device address (w32x driven)

HA0=High Address 0= High byte of external device address (w32x driven)

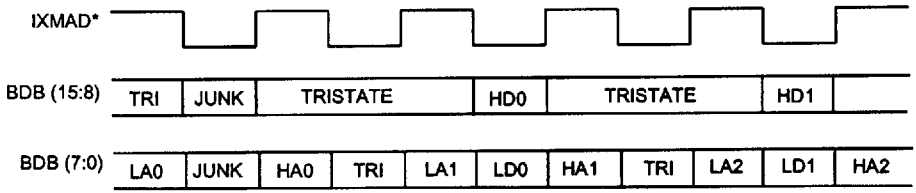
LD0=Low byte register data (VIPeR driven)

HD0=High byte register data (VIPeR driven)

TRI=Tristate

PREREAD = 0
READHI = 1

Figure 2.1.1 (cont'd) VIPeR (DD Bus Mode) Reads



Junk=VIPeR drives unknown data

LA0=Low Address 0= Low byte of external device address (w32x driven)

HA0=High Address 0= High byte of external device address (w32x driven)

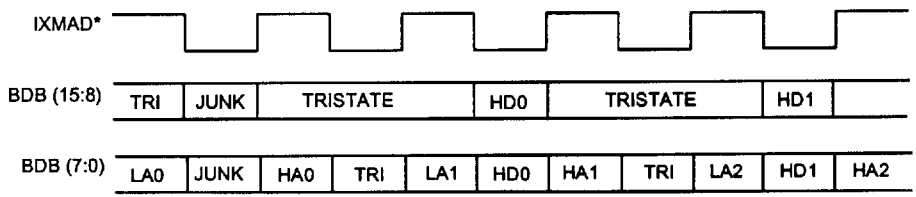
LD0=Low byte register data (VIPeR driven)

HD0=High byte register data (VIPeR driven)

TRI=Tristate

PREREAD = 1
READHI = 0

Figure 2.1.1 (cont'd) VIPeR (DD Bus Mode) Reads



Junk=VIPeR drives unknown data

LA0=Low Address 0= Low byte of external device address (w32x driven)

HA0=High Address 0= High byte of external device address (w32x driven)

LD0=Low byte register data (VIPeR driven)

HD0=High byte register data (VIPeR driven)

TRI=Tristate

PREREAD = 1
READHI = 1

Figure 2.1.1 (cont'd) VIPeR (DD Bus Mode) Reads

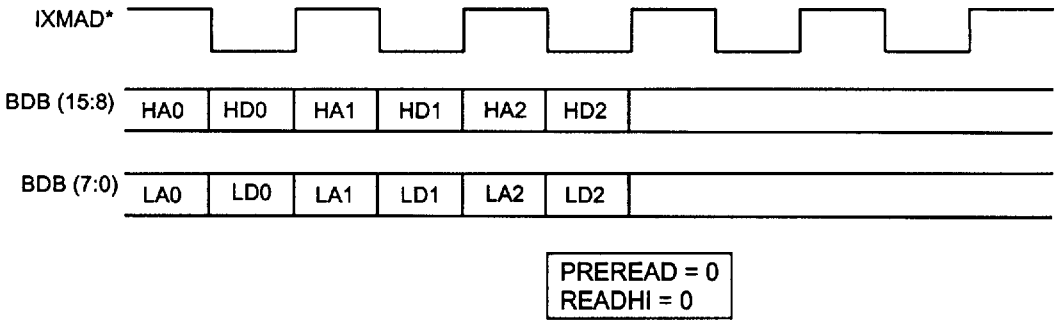


Figure 2.1.1 (cont'd) VIPeR (16-bit) BDB Bus Mode Reads

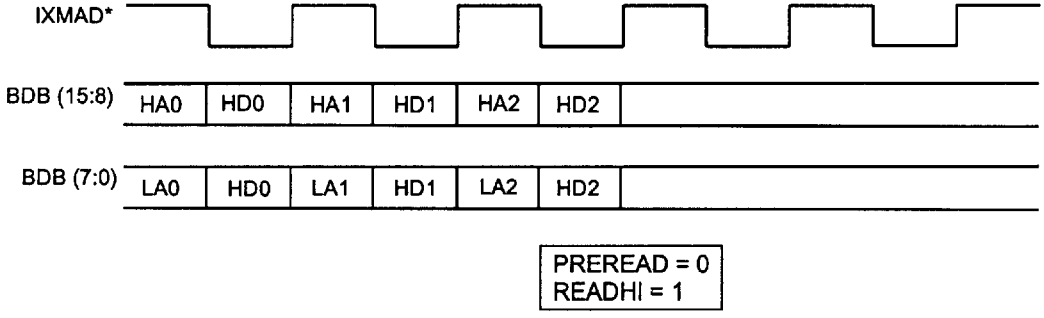


Figure 2.1.1 (cont'd) VIPeR (16-bit) BDB Bus Mode Reads

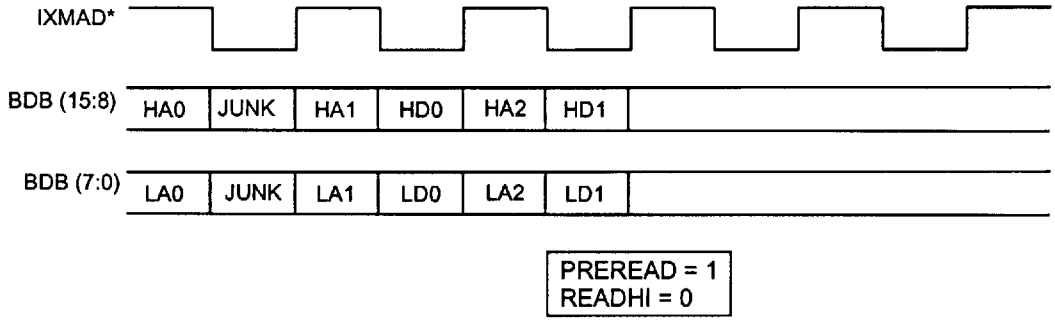


Figure 2.1.1 (cont'd) VIPeR (16-bit) BDB Bus Mode Reads

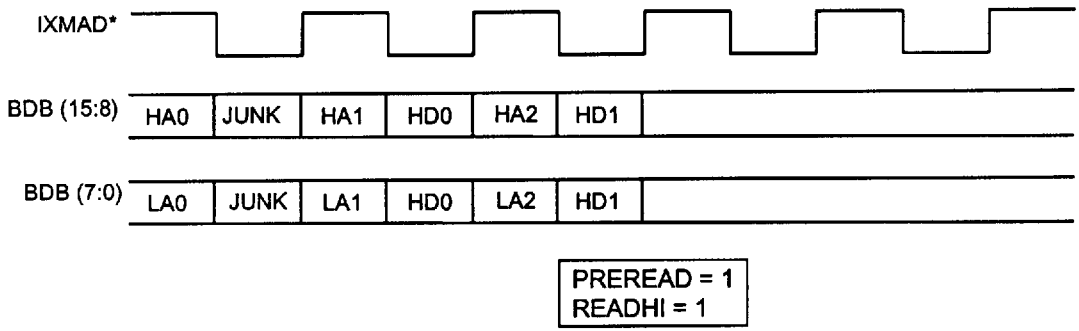


Figure 2.1.1 (cont'd) VIPeR (16-bit) BDB Bus Mode Reads

In addition to register accesses, the optional DRAM mask is written via host accesses over the BDB interface. When mask write mode is selected, via program settings, host accesses to VIPeR addresses 0x400 through 0x7FF are translated to DRAM mask writes, via a protocol detailed in Section 2.6.

Finally, pixel data streams can be input to the VIPeR over the BDB interface. When mask write mode is disabled, and CPUSYNC mode is programmed, host accesses to VIPeR addresses 0x400 through 0x7FF are accepted by VIPeR as input pixel data. Pixel addressing is implied, based on arrival of CPULSL, and CPUFSL (register access driven line sync and frame sync), and is serial, left to right, top to bottom. That is, first input pixel data sample written to VIPeR after CPULSL is assumed to be the leftmost pixel on that scan line; next input pixel data sample written by the host is assumed to be the second pixel (moving left to right) on that scan line. Similar for lines: first line of input pixel data after CPUFSL is assumed to be the top video input line, etc.

Addressing for input pixel streams is a function of the input pixel resolution mode selected. VIPeR accepts 15, 16, or 24 bit/pixel data streams in either RGB, YUV444, or YUV422 modes. For the 15 or 16 bit/pixel modes, (and since VIPeR is a 16 bit device) each host write contains the entire input pixel. Any address in the 0x400 through 0x7FF will be accepted by VIPeR as an input pixel. For 24 bit modes, VIPeR requires two host writes to get the whole 24 bit input pixel. These accesses are done as follows. Even addresses within the 0x400 through 0x7FF window are used to write the Green/Blue (or Y/U data in YUV444 mode) components of the pixel. Odd addresses within the 0x400 through 0x7FF window are used to write the Red (or Y in YUV444 mode) component of the pixel. Green (or V) occupies the upper data byte in even address transfers, Blue (or U) the lower data byte in even address transfers. Red (or V) occupies the lower data byte in odd address transfers. The upper data byte in odd transfers is don't care.

For 24-bit input modes, the host must deliver the data a full pixel at a time, via the two transfers; first the even address transfer, then the odd address transfer. A functional timing diagram showing the pixel writes is shown in Figure 2.1.2 below. Any other host write sequencing of delivering the pixel data will lead to unreliable operation.

2.2 ADC Interface

VIPeR has a primary data input bus which accepts digital video in (RED<7:0>, GREEN<7:0>, BLUE<7:0>), along with a data enable clock (LLCLKIN). Formats for this data input path can be either, RGB24, RGB16, RGB15, YUV444, or YUV422. Fuller, lower resolution ranges are also possible. For example, if 18 bits of Red/Green/Blue is available from the external A/D converter component, then VIPeR should be programmed as RGB24 input format, and the lower order Red/Green/Blue input pins should be grounded. For YUV422 mode, the inputs are wired as follows:

Y<7:0>	Red<7:3>,Green<7:5>
UV<7:0>	Green<4:2>,Blue<7:3>

Lower order bits should be grounded.

Based on programmed state, and LLCLKIN input, a VIPeR output, ADCCLK is driven back out of the chip. A final status output pin, ADCEN, identifies the state of the ADCEN register bit written by the programmer (in Configuration Register B). Actual choice of ADC (Red<7:0>,Green<7:0>,Blue<7:0>) pixel input, or host bus (BDB<15:0>) pixel input is based on the programmed setting of CPUSYNC register bit in the Control register, address 0x00. The selection is NOT based on the ADCEN register bit setting. ADC input is selected as the input pixel stream if CPUSYNC is reset to 0. The host bus is selected as the input pixel stream is CPUSYNC is set to 1.

ADCCLK output can be used by the external A/D converter to sample the incoming analog video. LLCLKIN is an input to the VIPeR from the A/D component, which is a digital data clock. VIPeR latches the input digital Red/Green/Blue, based on LLCLKIN, and programmed state. Program choice states allow VIPeR to latch digital input data on either rising or falling edge of LLCLKIN, or on either edge of an internally generated LLCLKIN divided by two signal. Detailed description of programming of choices is given in the register programming section, 5.0.

The ADCCLK output signal reflects the LLCLKIN input. ADCCLK is one of four choices; based on programming of ADCCLKDIV2, and ADCCLKPOL bits in Configuration Register B, address 0x04.

	ADCCLKDIV2	ADCCLKPOL	ADCCLK
mode0	0	0	inverted LLCLKIN
mode1	0	1	LLCLKIN
mode2	1	0	inverted LLCLKIN, divided by two
mode3	1	1	LLCLKIN, divided by two

In all 4 modes, ADCCLK is delayed about 5 nanoseconds after the referenced LLCLKIN version.

The internal signal VIPeR used to latch the Red/Green/Blue data is effectively the ADCCLK output. It is gated internally by active display time. Active display time is determined by vertical and horizontal delay programmable parameters. These are referenced to input frame and line sync information, respectively.

Functional timing of these modes is shown in Figure 2.2.1.

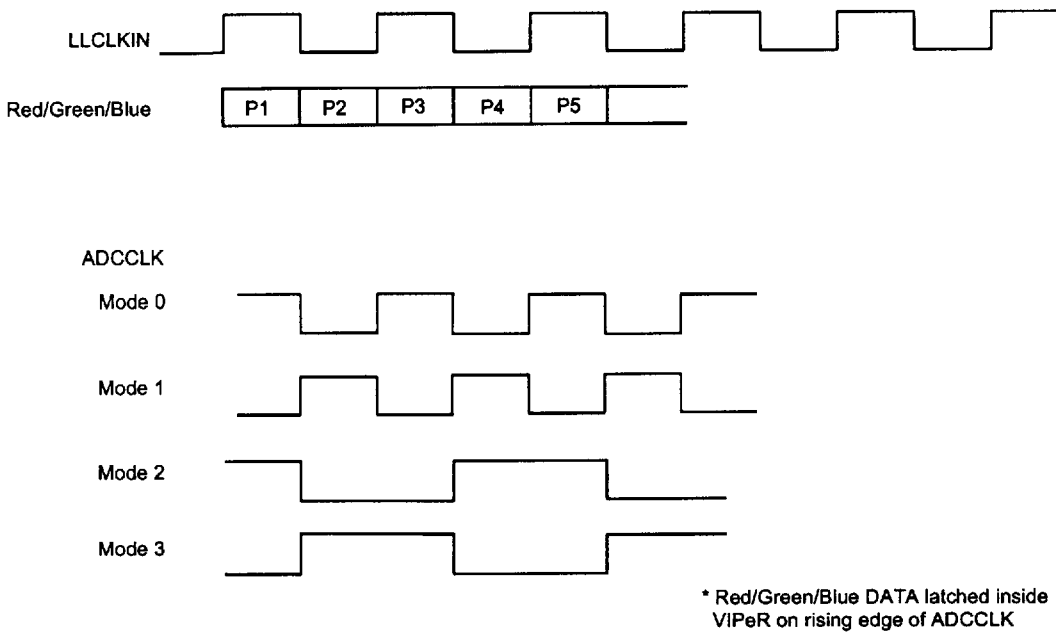


Figure 2.2.1 ADC Interface Timing

Internal to VIPeR, pixel data from this ADC interface block, and pixel data from the host interface block are multiplexed together, based on programmable choice of ADC data, or CPU data (as determined by CPUSYNC program bit in Control Register, address 0x00). Result of this multiplexing goes through bit/pixel formatting, to the YUV/RGB conversion circuitry.

2.3 Sync Separator

When operating in ADC input data mode (by resetting CPUSYNC register bit in Control Register, address 0x00 to zero), VIPeR is supplied video synchronization information in one of two ways. The first is from the direct FSIN*, LSIN*, OFIN inputs. These inputs are low active video frame sync, video line sync, and odd field indicators. The second ADC input data mode synchronization source comes into VIPeR via the composite sync input, CSYNC. The following chart describes the sync source selection, based on the program register bits, CPUSYNC, and SSBYPASSL. SSBYPASSL is a register bit programmed in Configuration Register B, address 0x04.

CPUSYNC	SSBYPASSL	Sync Source
0	0	CSYNC input: fed into sync separator circuit
0	1	FSIN*, LSIN*, OFIN, inputs: used directly
1	X	CPUFSL, CPULSL register bits: used directly

VIPeR uses the sync source information it chooses, to generate internal vertical and horizontal sync signals, as well as a field indicator signal. These internal video synchronization signals are used to determine active video display times, and to time the processing, digital storage, and display of video information.

When the sync separator is enabled (by resetting CPUSYNC and SSBYPASSL to 0), VIPeR video sync information is gotten from the input composite sync, CSYNC, input. The Sync Separator unit takes TTL level CSYNC and separates it into horizontal and vertical sync components, detects field information, and determines whether the input signal is interlaced or non-interlaced. Using this detection circuitry, it generates internal VIPeR vertical and horizontal sync signals, as well as the field indicator signal. If the sync separator is not activated, and sync input information is coming from one of the other sources, then the internal VIPeR vertical and horizontal sync signals, and the field indicator signal, come directly from the other chosen source. These other chosen sources are FSIN*/LSIN*/OFIN inputs, or CPUFSL/CPULSL register bits. In addition to the video control functions discussed above, these internal sync signals are output from the VIPeR on the output pins, VSYNCL, HSYNCL, and FIELD.

When programmed to CPUSYNC=1 mode, the CSYNC, FSINL, LSINL, and OFIN inputs are don't cares for VIPeR. When ADC mode is selected, and the sync separator is enabled, CSYNC input is delivering the video sync information. FSINL, LSINL, and OFIN are don't cares. When ADC mode is selected, and the sync separator is in bypass mode, VIPeR gets its primary video sync information from the FSINL, LSINL, and OFIN inputs. However, CSYNC input is NOT a don't care in this mode. Horizontal sync information must be input to CSYNC (the LSINL input is a good choice to feed into CSYNC as well). Further, the sync separator should still be programmed for the expected video sync frequencies. Programming the sync separator is discussed below, and also in the register programming section, 5.0.

The VIPeR sync separator allows inputting of varying quality composite sync information, from studio quality through consumer grade VCRs in slow motion and fast forward modes. VIPeR can lock to not only a wide quality range of input composite sync signals, but also to a wide range of video input formats, including NTSC, PAL, and SECAM. Because it is user programmable, the sync separator allows VIPeR to lock to non standard video sources as well.
The sync separator contains four basic functional sub-units: noise cancellation, horizontal sync separation, vertical sync separation, and field detection. Each of these will be described in the paragraphs below.

The noise cancellation sub-unit filters out any pulses on CSYNC input which are one SCLK time duration or less. Based on 50MHz SCLK, this translates to a filter which removes any CSYNC pulses of less than 20 nanoseconds.

The horizontal sync separator and vertical sync separator sub-units operate in almost identical fashion: one on the line sync information, the other on the frame sync information. In order to program these sync separator sub-units, the programmer must know the general parameters of the composite sync input. These are, the expected horizontal and vertical frequencies.

The sub-units contain timers which the programmer sets. These timers instruct the sync separator to look for input sync pulses in defined time windows, to block out input sync pulses during unexpected times, and reset based on expected sync pulse frequencies.

The horizontal sync separator uses three window timer settings to lock onto line sync components of the input CSYNC signal. These are user-programmed values called, Horizontal Sync Width, Horizontal Window Closed, and Horizontal Window Open register/timers. They are accessed as VIPeR registers, at addresses, 0x10, 0x12, and 0x14, respectively. These are discussed in the register programming section, 5.0. Values are programmed in units of number of SCLKs, minus 1.

The width, window closed, and window open register/timers define periods when the VIPeR sync separator is expecting to see horizontal sync pulses.

If the sync separator detects a horizontal sync signal during unexpected (Horizontal Sync Width or Horizontal Window Closed) time, it is ignored. If one is detected in the expected (Horizontal Window Open) time, it is used as the line sync information, passed on as the internal VIPeR horizontal sync signal, and output on HSYNCL. Starting when the internal horizontal sync is detected, the internal horizontal sync signal, as well as the HSYNCL output signal, are activated for a length of time equal to the value programmed in the Horizontal Sync Width register. When the horizontal sync signal is detected on CSYNC during the window open time, the timers reset themselves. If a horizontal sync signal is not detected on CSYNC during the entire horizontal window open period, then the horizontal sync separator sub-unit will create one: based on the expiring of the horizontal window open timer. If this happens, the timers all reset themselves just as if this generated horizontal sync pulse was actually derived from the input CSYNC signal. The internal VIPeR horizontal sync pulse and HSYNCL output activate for the horizontal width time: all just as if a horizontal sync signal was detected on the CSYNC input. Figure 2.3.1 illustrates the horizontal windowing times.

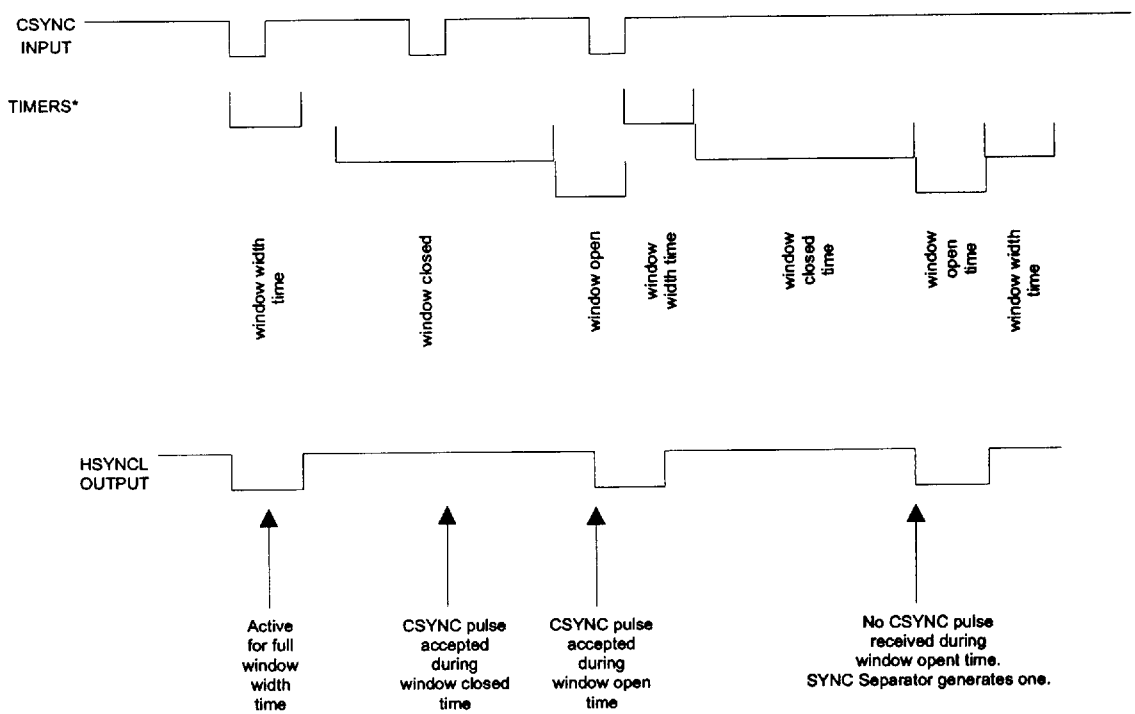


Figure 2.3.1 Sync Separator Windowing

Programming the horizontal sync separator requires knowledge of the expected horizontal sync frequency. For NTSC composite video, with a horizontal frequency of 63.5 microseconds, reasonable numbers to set the window timer values (assuming 50MHz SCLK) are as follows.

Horizontal Sync Width	4.7μS	0x00EA
Horizontal Sync Window Closed	57.6μS	0x0B3F
Horizontal Sync Window Opened	6.5μS	0x0145

These values control the horizontal sync separator as follows. When a horizontal sync is detected (or generated) all the counter/timers reset. The horizontal sync width timer activates first, and at the same time, the internal horizontal sync signal, and the output HSYNCL activate. They remain active for 4.7μS (the programmed horizontal sync width value). During this 4.7μS, the window for other horizontal sync inputs on CSYNC is closed. 4.7μS after the first horizontal sync the horizontal sync width timer expires, and the horizontal sync window closed timer begins. For the next 57.6μS (the programmed value), the horizontal sync separator remains closed. Any horizontal sync pulses input on CSYNC will continue to be ignored. This brings us up to 62.5μS after the last horizontal sync (4.7 plus 57.6). Now, the horizontal window closed timer expires, and the horizontal window open timer begins. For the next 6.5μS, (a window from 62.5μS through 69.0μS) the horizontal sync separator is looking for a horizontal sync pulse on the CSYNC input. IF at any time within this 6.5μS window, a horizontal sync is detected, it is used, and the horizontal window open timer immediately closes. This resets everything back to the beginning. If a horizontal sync pulse is not detected on CSYNC for the entire 6.5μS window open time, then a horizontal sync is generated, the horizontal window open timer closes, and everything resets back to the beginning again.

The result of this is a digital band pass filter. The larger the open window time, the faster VIPeR will lock to the incoming sync. The trade-off here is that VIPeR is more likely to allow bad pulses on CSYNC to be used as valid horizontal sync pulses.

The vertical sync separator works much the same as the horizontal sync separator, with window width, window closed and window open timers. For the vertical sync separator, however, there is an additional timing parameter, minimum vertical sync pulse width. This timer works similarly to the noise cancellation sub-unit. It watches input vertical sync pulses on CSYNC input, and allows only pulses whose duration are longer than the programmed minimum pulse width to be passed.

The user programmable vertical sync separator window register/timers are Vertical Sync Width, Vertical Window Closed, Vertical Window Open, and Vertical Sync Detector Width (the minimum vertical sync pulse width). They are accessed as VIPeR registers at addresses, 0x18, 0x1A, 0x1C, and 0x16 respectively. Values for the first three registers programmed in units of horizontal syncs pulses, minus 1. Values of the Vertical Sync Detector Width are programmed in system clocks, SCLK, minus 1. For NTSC video inputs, a reasonable value of Vertical Sync Detector Width is 10μS, or about 0x200 for 50MHz SCLK.

The final sub-unit is the field detector. It is controlled by the Field Detection Window Width register, programmable in horizontal sync pulses, minus 1, at VIPeR register address 0x1E. After a vertical sync pulse is received, the field unit counts the programmable number of horizontal syncs, and determines, via the monitored level of the CSYNC input, whether the field is even or odd. Result of this measurement is broadcast to the VIPeR internally, to effect video processing functions, and is output on the FIELD output signal pin.

2.4 YUV to RGB Conversion

The VIPER YUV/RGB conversion is done to CCIR601 specification. RGB values range from 0 to 255. YUV values between 0 and 255 are converted, although the specification is from 16<Y<235, and 16<U,V<240. Resultant RGB values less than 0 are truncated to 0. Resultant RGB values greater than 255 are saturated to 255. Nine bits of precision are maintained throughout the conversion. It is done in real time, as a pipeline delay structure between the input video stream formatting, and the video scaling units. Mathematics of the YUV to RGB conversion are as follows:

Green = Y - (0.698) V - (0.338) U + 132.56

Red = Y + (1.371) V - 175.45

Blue = Y + (1.732) U - 221.75

VIPeR converts YUV data input either from the ADC input port or from the host bus, BDB<15:0> bus. This gives VIPeR the capability of working with either front end A/D converters which deliver digitized live data in YUV format, or playback, scale and convert YUV based stored video.

Three types of YUV formats are accepted, and converted to 24-bit RGB. These are YUV444, and two types of YUV422, YUV422a, and YUV422b where the formats of each are as follows. The YUV422 are inherently 16 bit/pixel modes. YUV444 is 24 bits, 8 each of Y, U, and V.

YUV422a:

input pixel 1<15:0> = Y0<7:0>,U0,1<7:0>
input pixel 2<15:0> = Y1<7:0>,V0,1<7:0>
input pixel 3<15:0> = Y2<7:0>,U2,3<7:0>
input pixel 4<15:0> = Y3<7:0>,V2,3<7:0>
etc...

YUV422b:

input pixel 1<15:0> = Y0<7:0>,U0,1<7:4>,V0,1<7:4>
input pixel 2<15:0> = Y1<7:0>,U0,1<3:0>,V0,1<3:0>
input pixel 3<15:0> = Y2<7:0>,U2,3<7:4>,V2,3<7:4>
input pixel 4<15:0> = Y3<7:0>,U2,3<3:0>,V2,3<3:0>
etc...

The YUV422 are inherently 16 bit/pixel modes. YUV444 is 24 bits, 8 each of Y, U, and V.

Further, the byte lanes of the Y versus UV data can be switched via setting program bit YLSBIN, bit 10 in Configuration Register B, address 0x04. This feature, however, is available in Rev.B silicon only.

2.4.1 Playback YUV/RGB Conversion

For playback of stored video in YUV mode, YUV422a is by far the most popular format, so we will detail that mode here. The host CPU delivers YUV422a mode pixels to VIPeR over the BDB<15:0> lines. In general, Y<7:0> is input on the most significant byte of the input pixel, and UV<7:0> is input over the least significant byte. U,V is time multiplexed, so that pixels alternate YU, or YV samples. VIPeR then does a linear interpolation of the 2:1 subsampled U and V values to bring the YUV up to 24 bits per pixel. This 24-bit YUV is then input to the actual conversion block to create the 24 bit RGB output. This 24-bit RGB output then goes on to the video scaling units (described in Section 2.5). Programmer has option of switching the VIPeR interpretation of data to be Y in the LSB and UV in the MSB, by setting YINLSB. In this case, CPU must deliver the Y data in the LSB, etc.

2.4.2 YUV/RGB Conversion of Live Video, via ADC Input Port

VIPeR accepts YUV formatted video input to its ADC (Red<7:0>, Green<7:0>, and Blue<7:0>) input pins as well. Again, more popular front end A/D conversion components output YUV422, so we detail that mode here. Just as with the host input, VIPeR accepts the YUV422 input data on ADC, and multiplexing it with the host bus, feeds it to the YUV422 interpolator, and then on to the YUV to RGB conversion. 2:1 subsampled UV data is brought up to full resolution by the linear interpolator, to take input YUV422 to YUV444, 24 bit/pixel. Then, this 24-bit YUV is changed to 24-bit RGB and fed into the video scaling units.

Inputting YUV422 video to the ADC pins must follow this convention:

Input Data	ADC Pin Input
Y<7>	RED<7>
Y<6>	RED<6>
Y<5>	RED<5>
Y<4>	RED<4>
Y<3>	RED<3>
Y<2>	GREEN<7>
Y<1>	GREEN<6>
Y<0>	GREEN<5>
UV<7>	GREEN<4>
UV<6>	GREEN<3>
UV<5>	GREEN<2>
UV<4>	BLUE<7>
UV<3>	BLUE<6>
UV<2>	BLUE<5>
UV<1>	BLUE<4>
UV<0>	BLUE<3>

RED<2:0>, GREEN<1:0>, and BLUE<2:0> input pins should be grounded.

2.5 Horizontal/Vertical Scaling

The VIPeR scaler can be used to convert either live video or stored video to arbitrary sizes in both the horizontal and vertical dimensions. It calculates visually correct images at up to one pixel per clock cycle. By smoothly scaling via an area average algorithm, VIPeR uses all of the incoming video to create the resulting scaled output video. In scaling down, no pixels are dropped, and no lines are dropped. Rather, all the incoming pixels and lines are used: every bit of the incoming data has a contribution to generating the output video stream. In scaling video up, no pixel replication or line replication is done. Similarly, all the incoming video is weighted and used to generate the output video stream. By using this sophisticated area averaging algorithm, very high quality video scaling is done: visible artifacts in the scaling are minimized.

In order to program the VIPeR horizontal and vertical scaling units, the CPU passes the input and desired output horizontal and vertical video resolutions. The VIPeR software driver then calculates the program register loads required to scale from the input video resolutions to the output video resolutions. Programmed registers that control the scaling are:

X Scaling Coefficient	register address 0x30
XK Scaling Coefficient	register address 0x32
Y Scaling Coefficient	register address 0x34
YK Scaling Coefficient	register address 0x36

Since the video scaling algorithm is Tseng Labs confidential, the programming of these registers is undocumented. VIPeR driver software calculates the register values based on the horizontal and vertical input and output scaling resolutions.

VIPeR can scale video from 1024x1024 input resolution to 16x16 output resolution in one extreme, and from 16x16 input resolution to *720x1024 output resolution in the other extreme. All other input to output scaling factors within this range are possible.

*720x1024 for VIPeR Rev.B. 640x1024 for VIPeR Rev.A.

When scaling live video, as input from the ADC input port (Red<7:0>, Green<7:0>, Blue<7:0), the maximum frequency which the input pixel stream can come into the VIPeR is 1/2 SCLK frequency. Based on SCLK frequency, this could potentially limit the maximum video input resolution. A constraint on the scaling is imposed for video coming into the VIPeR over the ADC port. Since VIPeR must generate lines of video when scaling up in the vertical dimension, data overruns can occur based on VIPeR processing, W32/W32i/W32p ability to accept the VIPeR output video stream, and the horizontal video resolutions. In general, no data overruns leading to visible artifacts will occur if the video vertical output resolution is no greater than the video vertical input resolution. Care must be taken in scaling up ADC input video by ratios of between 1 and 2. Scaling up video to factors greater than 2:1 in the vertical dimension is not recommended for ADC input video streams, unless provision can be made to "stall" the input video stream based on instantaneous VIPeR video processing status. This video processing status information is discussed below.

When scaling stored video, as input from the host bus (BDB<15:0>), the full range of video scaling ratios are possible. Maximum video input rate over the host bus is determined by host bus clock frequency, as determined by the W32/W32i/W32p, and by the host bus configuration: either BDB mode or DD mode. See Section 2.1 for a description of these host bus modes. As an example, while operating in DD host bus mode, at 33MHz, W32/W32i/W32p writes a new pixel to VIPeR at about a 5-6 Mpixel/sec rate. At 16bpp, this is about 10-12 Mbytes per second. These rates are more than sufficient to write typical format, typical resolution stored video streams, such as Microsoft Video for Windows, .AVI files, Indeo format, Cinepak format, and MPEG1 video streams. Care must be taken when vertically scaling up input video from the host bus. As was mentioned for the ADC port live video input, when VIPeR is generating lines for vertically scaled up video, care must be taken to throttle the input video stream so as to not create data overruns in the IMA (W32 interface) FIFO. This throttling is more easily done when input video comes over the host bus, since the CPU controls the data flow rate directly. Throttling can be accomplished by software monitoring, via periodic register reads, of the IMA FIFO empty flag. This flag is the EMPTY register status bit in Status Register B, address 0x0A. When the FIFO is empty, the CPU can safely send the next line of input video, which triggers VIPeR to deliver not only that output scaled video line, but potentially other generated video lines of output data when scaling up in the vertical direction.

The VIPeR scaling algorithm, and data input stream sources supported, make VIPeR ideal for live video scale and show, for video capture (when used in conjunction with W32/W32i/W32p), and for scaled video playback of stored video.

2.6 External Mask DRAM Control

Overview

VIPeR has a 256Kx4 DRAM controller which it uses to access alpha bit (mask overlay) information from an external 256Kx4 DRAM. VIPeR passes one bit per pixel, over the IXMSK output to the W32x, as an alpha bit paired with each video pixel. The W32x then uses this alpha (or mask) bit to determine whether video or graphics data should be written into the frame buffer at that pixel location. In this way, video/graphics overlaying, and pixel by pixel mixing of video and graphics is done.

The external (optional) DRAM overlay mask is controlled by the VIPeR's preset operating modes, and it's programming. The mask DRAM can be written at any time via sequences of register accesses to the mask control register, address 0x38, and to the VIPeR data address space, 0x400-0x7FF. However, it is strongly recommended that writes happen only during non-active video display times. A CAS before RAS refresh cycle, in addition to the accessed row refresh, occurs whenever a mask DRAM write is done.

The mask DRAM is *not* directly readable through the VIPeR. Contents of the DRAM can be indirectly inferred by loading various video, graphics, and mask patterns, running video into the graphics frame buffer, and downloading the contents of the frame buffer through the W32x. Contents of the mask DRAM are read out, and paired up with corresponding video pixels, during active video display times. Addressing is done automatically based on the VIPeR's current internal line and pixel numbers. Two refresh cycles are done every video scan line: one read access refresh to the row corresponding to that particular video line, and one CAS before RAS refresh cycle, triggered by the horizontal sync pulse.

Data Organization

The mask DRAM is organized logically into 512 rows and 512 columns, each of 4 bits. The mask DRAM is programmed, and VIPeR interprets the data, as one bit of mask for each video pixel. Each mask bit is paired up (by the VIPeR) with the corresponding video pixel it is generating, and sent over the IMA port to the W32x. The W32x uses this mask bit to mix the video and graphics in the single frame buffer. The optional masking capability is enabled by setting IXMSKEN, bit 14 in VIPeR register 0x02. The polarity of the read out mask bit can be inverted by setting IXMSKPOL, bit 15 in VIPeR register 0x02.

When reading data out of the mask DRAM during active video display time, the VIPeR associates it's current line number with a row address in the mask DRAM. Rows addresses are accessed in descending order: video display line number 0 corresponds to mask DRAM row address 511, video display line number 1 corresponds to mask DRAM row address 510, and so forth. The final video display line number N corresponds to mask DRAM address (511-N). Horizontally, the mask DRAM data within each row address (within each video display line) corresponds directly to video display pixel number. The first video display pixel, pixel 0, corresponds to column address 0. Since the data in the mask DRAM is 4 bits per location, however, there are 4 mask bits read by the VIPeR every access. The VIPeR stores 4 bits of mask for each access and pairs them up as the following. For video display pixels 0-3, column address 0 is used, and the data is taken from bits 0-3 for pixels 0-3. Therefore upon writing the mask DRAM pattern, the programmer can think of the mask data pattern as looking physically like the video screen, with inverted row addresses (inverted line numbers).

The organization of the mask data, in this case, is 512 video lines vertically, by 2048 video pixels horizontally. This is the default organization mode: when MSKCOL8, bit 10 in register 0x38 is reset to zero. An alternate organization, one useful for video display formats with greater than 512 scan lines (like PAL) is discussed below. See figure 2.6.1, Default Mask DRAM Organization

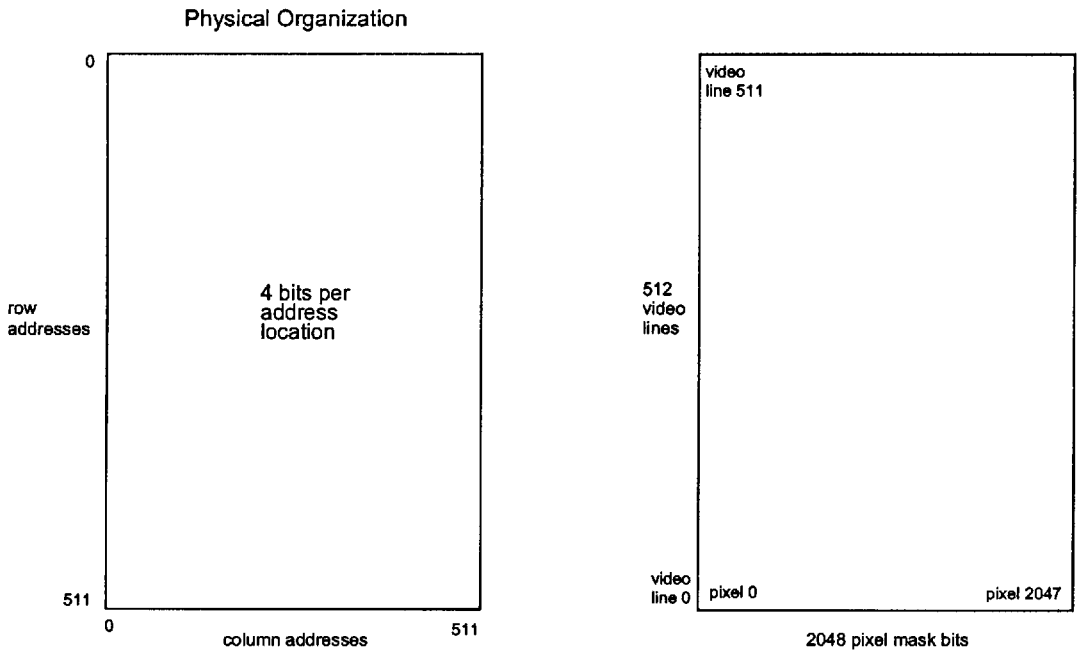


Figure 2.6.1, Default Mask DRAM Organization

2.6.1 Split mask DRAM modes

Setting MSKCOL8, bit 10 in register 0x38, forces the column address to range from 256-511 when reading mask data during active video display. In this mode, the active display column address equals the current pixel number (divided by 4, for 4 mask bits per column address) plus the offset column address in the most significant bit. In this mode, the mask data can be thought of as two different mask patterns, each of 512 rows by 256 columns, or 512 lines by 1024 pixels. An additional mode, enabled by setting G512LINES, bit 6 in address 0x02, enables wrap over of the two mask patterns. When video line number is less than 511, the lower half column addresses are accessed: when the video line number, N, becomes greater than 511 (like for the bottom of a PAL screen) the row addresses go to N-511, and the column addresses go to (pixelnumber/4 + 256) See split mask dram organization, Figure 2.6.2.

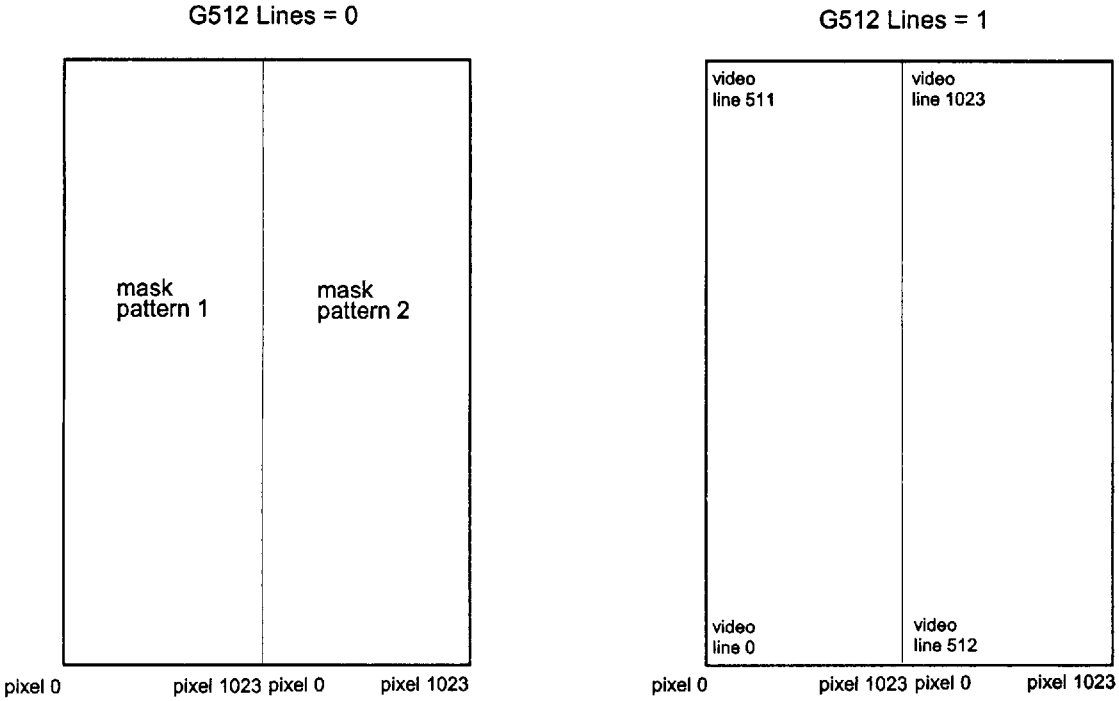


Figure 2.6.2 Split DRAM Mask Organization

2.6.2 Write Sequence

The mask DRAM is written by first programming the desired row address in mask control register, address 0x38. In addition, CPUMSKWRL, bit 11 must also be reset. The MSKCOL8 bit is also set to the desired value based on whether split mask DRAM mode is being used or not. This register access loads the desired row address to be written, tells the host bus interface logic to interpret subsequent accesses to addresses 0x400-0x7FF as mask DRAM write data, as well as enabling the mask DRAM write state machine in the VIPeR.

After this is done, a write is done to an address in the 0x400-0x7FF range. The lower 7 bits of this register address define the upper 7 (of 9) column address to bits to be written to. The lower 2 bits of the column address are provided by sequencing of an internal VIPeR state machine during the actual mask DRAM write sequence. Sixteen bits of data associated with this register access are then loaded by the VIPeR and written in 4 CAS write cycles, to 4 consecutive column addresses to the mask DRAM.

After the 4 CAS write cycles (writing 16 bits of data) are completed, a CAS before RAS refresh cycle is completed. Timing of the write cycling is NOT programmable, however, it is done slowly enough so that 80 ns DRAMs can be used.

To ensure proper execution of the write cycles the programmer must wait a minimum of 640 nanoseconds between subsequent mask DRAM writes. The 640 nanoseconds is measured from one access to 0x400-0x7FF to the next 0x400-0x7FF access. See write mask timing diagram, Figure 2.6.3

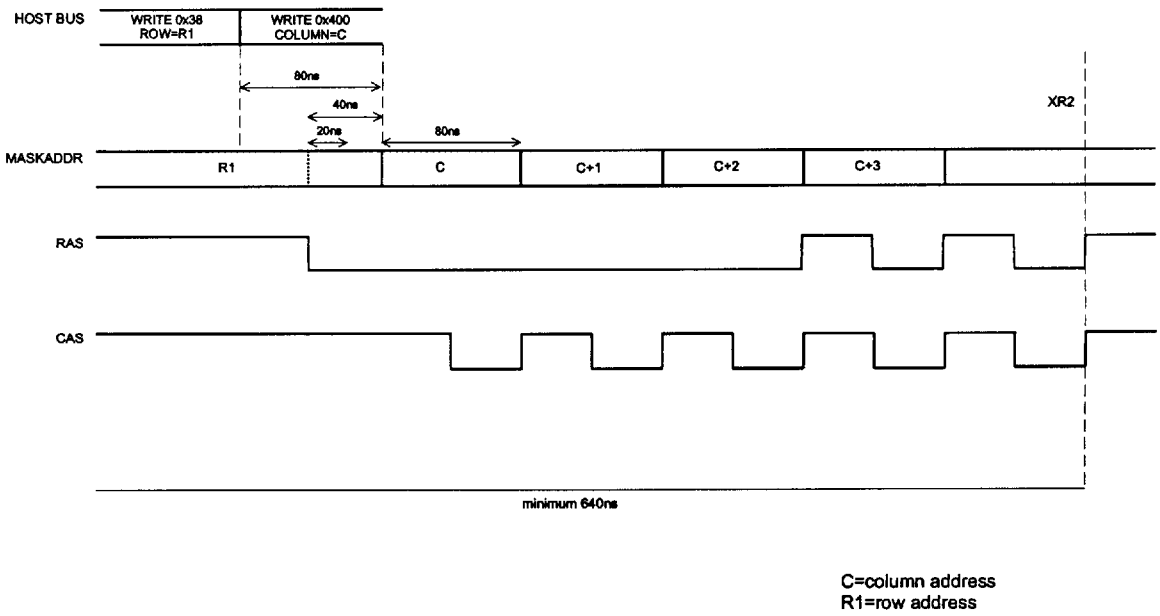


Figure 2.6.3 Write Mask Timing Diagram

2.6.3 Read Sequence

The mask DRAM is read by the VIPeR during active video display times. Addressing is done automatically by internal vertical line and horizontal pixel number counters. The row address, and a RAS cycle is done at the beginning of every video scan line (as initiated by horizontal sync). The row address is equal to the inverted video scan line, or the video scan line minus 511, if G512LINES is set to 1, and the current pixel line number is greater than 511.

CAS cycles are initiated by the VIPeR for every fourth video pixel. Timing of the access is controlled by internal VIPeR state machines. Column address is equal to the video pixel number, divided by four.

At the end of every scan line, a CAS before RAS refresh is executed.

2.6.4 Special refresh considerations

While displaying video, refreshes to the mask DRAM occur at effectively twice the video line rate. While writting data to the mask DRAM, refreshes occur once for the specific row being written, and one CBR refresh. When CPUMSKWRL is left active (reset to 0), CBR refreshes happen automatically, one every 1.28 usec. Care must be taken when leaving CPUMSKWRL set to 1, and with no mask writes, or with no active video data moving through the VIPeR. In this case, refreshes to the mask DRAM do NOT occur, and data loss may result. If VIPeR is left in CPUMSKWRL=1 mode, with no video input, and no mask DRAM writes, for greater than the mask DRAM refresh time requirements, the mask DRAM data will need to be reloaded before video can be enabled.

2.7 Processed Data Output

VIPeR has a 25 pin processed data output bus. It consists of a strobe signal, PSTBL, and 24 data outputs, PRED<7:0>, PGREEN<7:0>, PBLUE<7:0>. The data signals are the direct output of the horizontal and vertical scaling units.

PSTBL is the data valid strobe associated with the output data. It is driven low for 1 SCLK duration whenever a new pixel is output on PRED, PGREEN,PBLUE. PSTB is driven 5 (±3) nanoseconds after the falling edge of SCLK. Pixel data output on PRED, PGREEN, PBLUE is driven 5 (±3) nanoseconds after the rising edge of SCLK. PSTB lags the pixel data outputs by ½ SCLK.

Pixel data on PRED<7:0>, PGREEN<7:0>, and PBLUE<7:>, hereafter referred to as PRGB, reflects the input programmed pixel resolution. 24 bits of PRGB are always output, however, lower bits of PRGB are padded with zeros if the input pixel resolution is less than 24 bits/pixel. For example, if the input pixel source is 16 bit, RGB565, then PRED<2:0>, PGREEN<1:0>, PBLUE<2:0>, will appear always as logic value 0. In YUV422 mode, although the input pixel resolution source is really 16 bit/pixel, VIPeR is internally interpolating (thereby expanding) the data up to 24 bit/pixel. PRGB outputs are the full 24 bits for YUV422 modes.

This PRGB output pixel data is the result of input processing, YUV/RGB conversion (if selected) and the horizontal and vertical scaling. It is the pixel data stream as it is input to the W32 interface Image Data Port unit. There it is FIFO'd, converted to the programmed output pixel resolution, and delayed (stored in the FIFO) until the W32/W32i/W32p is able to take the data. In general, the PRGB pixel output is earlier than the pixel output stream on the Image Data Port. Also, since it is delivered 24 bit/pixel, and the Image Data Port is time multiplexed down to 8 bit per clock cycle output, the PRGB data can come out at higher frequency.

One use for the PRGB output pixel stream is as a direct feed for an external compression engine. Since it is output a full pixel at a time (not time multiplexed onto a narrower output bus, and also has its own data valid strobe, intergating the processed data outputs to a compression component is fairly straightforward. Care must be taken, however, with the line and frame sync outputs. The VSYNCL, HSYNCL, and FIELD sync output signals should be used with the PRGB data over the processed data outputs. The IXFS, IXLS, and IXOF are the output sync signals associated with the Image Data Port.

For VIPeR Rev.A silicon, all 24 bits of PRGB are outputs at all times. For VIPeR Rev.B silicon, two of the bits, PRED<0>, and PBLUE<0> are I/O pins and are configured as inputs when RESET is active. They are used as POR pins to select host bus BDB<15:0> address and data timings. When RESET is inactive, PRED<0> and PBLUE<0> are outputs always, and the functionality returns to the same as the other 22 PRGB outputs. The way PRED<0> and PBLUE<0> get their input is through internal weak pullups or via jumpers to weak pulldowns on the video board. These pullups and pulldowns have negligible effect on the pins' output operation when RESET is inactive.

2.8 Audio Interface

The VIPeR audio input is provided to accept digital audio data in, and synchronize it with the video. VIPeR contains an internal FIFO for storing audio (or any other relatively lower frequency, like SMPTE) data. There is a pass through mechanism whereby this data is output, over the Image Data Port, IXDATA<7:0>, and stored in unused portions of the frame buffer. Although no processing on this data is done, the transfer capability of appending audio data to the end of every video scan line can potentially simplify lip sync issues of audio/video streams.

VIPeR has 9 input pins associated with the audio interface. They are AUD<7:0>, and AUDCLK. Digital audio data is presented 8 bits at a time, and written into the VIPeR when AUDCLK tranitions from low to high. Data accumulates in the internal FIFO, and is output on the Image Data Port pins, IXDATA<7:0>, after the end of each video scan line, based on program settings, video horizontal blanking time, and W32/W32i/W32p ability to accept Image Data.

VIPeR outputs whatever audio data it has in the FIFO only when the following conditions are met.

1. There is data in the audio FIFO which has not been output yet

2. All video data for current output video line has passed over the Image data port to the W32/W32i/W32p.

3. Horizontal sync has not yet occured (signifying start of next video line)

While these conditions are met, VIPeR will output a (maximum) number of audio data bytes, on IXDATA<7:0>, equal to a programmed value. The programmed value is set indirectly as the difference between the programmed maximum number of bytes per line, and the programmed number of video bytes per line. These are VIPeR registers, addresses 0x28, and 0x26 respectively. If any of the three conditions above become invalid, VIPeR immediately stops outputting audio data.

Accumulation of audio data is done independently, and asynchronsously, from the outputting. Although it is specified for operation up to SCLK rates, audio data must typically come into VIPeR at much slower rates, so as not to overflow the internal audio FIFO.

The audio FIFO is 128 bytes deep. There is no protection for FIFO overrun should the stack of audio data exceed 128 bytes. Data is overwritten in the case of audio FIFO overruns. Effects on the VIPeR audio output will be to effectively skip input samples.

As an example of "typical" operating environments, we consider now NTSC output video rates, with CD quality audio sound streaming into VIPeR, being appended to the end of the video scan lines, and stored in the graphics frame buffer. NTSC video occupies 63.5 uS of time per scan line, 240 lines per field, and 60 fields per second. CD quality audio is 44 Ksamples per second, at 16 bits/sample: 88 Kbytes/second. For this system, [(1/60sec/field)*(88000bytes/sec)] = 1467bytes/field must be output by VIPeR in order to aviod audio FIFO overrun. This corresponds to [(1467bytes/field)/(240lines/field)] = 6.1 audio bytes per line. To program an NTSC video, CD audio system using VIPeR, the programmer should set maximum bytes per line to be at least 7 bytes more than the programmed video bytes per line set. In addition, system resources should be allocated, including memory in the graphics frame buffer, to allow for at least 7 bytes of audio data to be appended to each (about 1K byte) video scan line data.

2.9 Video Control Port

VIPeR has an 8 bit, general purpose, Video Control Port. These 8 signals can be independently configured as either input or output pins, by program settings.

Data into and out of the pins is accessible by the host CPU via internal registers in VIPeR which write or read the values on the Video Control Port (VCP) pins. This capability allows for CPU access to other devices on the video card, through the VIPeR W32/W32i/W32p memory mapped access to this internal Video Control Port register.

This port can be used to monitor status of signals on the video board, allowing host software access to signals it otherwise would not have access to. As outputs, VCP pins can be used to control other devices such as external DACs or IIC busses, or A/D video controls.

On reset, all VCP pins are configured as inputs. Programmer controls the VCP via access to the VCP register, address 0x2E. Bits 15:8 of this register determine the direction (input or output) of the VCP pins, VCPORT<7:0>. For example, if the programmer sets VCP register bits <15:8> to be value 01010011, then VCPORT pins will be configured as follows:

VCPORT<7>	input
VCPORT<6>	output
VCPORT<5>	input
VCPORT<4>	output
VCPORT<3>	input
VCPORT<2>	input
VCPORT<1>	output
VCPORT<0>	output

Values on the output configured pins will be determined by the data the programmer writes into the lower byte, VCP<7:0> of the VCP register. In this case, if the programmer writes X0X1XX10, then the output configured pins will drive the following data:

VCPORT<6>	0
VCPORT<4>	1
VCPORT<1>	1
VCPORT<0>	0

The don't care values reflect the fact that, for input configured pins, the corresponding value written in VCP register bits <7:0> is NOT driven out of the VIPeR, (because the pin is configured as input).

Upon reads of the VCP register, bits 15:8 will return the previously programmed values. Bits 7:0 will return the logic state of the wire attached to the corresponding VCPORT pin, for those pins configured as inputs. For VCPORT pins configured as outputs, reads of these register bits will return the previously programmed values written into VCP register bits 7:0.

Effects of programming the VCP register are seen on the VCPORT pins within about 10 nanoseconds after the registers are written. Timing for writing of the VCP register is determined by the host bus cycling. This is based on the IXMADL input signal of the host bus, described in Section 4.1.

Since the VCP register, and VCPORT pins can toggle at host bus access rates, the VCPORT is suited to control of slower external circuitry, like static board configurations, and IIC busses. IIC bus clock can be run via an output configured VCPORT pin, while IIC data supplied by another of the output configured VCPORT pins.

2.10 Image Data (W32/W32i/W32p) Interface Unit

The Image Data (W32x) interface unit takes scaled video data from the vertical scaling unit, the audio interface unit, and the mask DRAM control unit, formats, FIFOs, and delivers it to the W32x via the Image Data Port pins. This unit also accepts internal sync pulses from the vertical scaling unit, retimes, and sends them to the W32, also over Image Data Port pins.

Video data is output to the W32x based on the Tseng Labs IMA protocol. Output formats of either 15, 16, or 24 bits/pixel are time multiplexed onto the IXDATA<7:0> bus, regulated by the W32x ability to recieve it. The IMA is an asynchronous interface between the VIPeR and W32x. VIPeR recieves a "ready" signal (IXRDY) from the W32, and sends data and a data valid strobe on its IXDATA<7:0> and IXCMD* respectively. IXLS, IXFS, and IXOF are sync signals from the VIPeR which the W32x uses for pitch offset address calculations.

The Image Data Interface unit regulates not only the format, but the amount of video and audio data which it delivers to the W32, based on register programming. Pixels per line, maximum bytes per line, and video bytes per line register data is used by counters to deliver the expected amount of video and audio data to the W32. If enabled by setting the AUDIOEN bit in Configuration Register B, VIPeR will output any audio data which may have accumulated in the audio unit's FIFO during the previous scan video lines. Maximum number of audio bytes delivered is equal to the Maximum Bytes per line programmed, minus the Video Bytes per line programmed. W32 contains a Video Bytes per Line register/counter as well. Using this, and counting IXCMD* strobes, enables it to differentiate video from audio data presented to it on IXDATA<7:0>.

This unit also inserts horizontal, vertical and field information correctly within the video/audio output data.

The Image Data Interface Unit also outputs video/graphics masking information with each video pixel, on the IXMSK output. This unit's responsibility is to synchronize the fetching of mask information (done by the mask DRAM control unit) with the scaled video pixel data it delivers to the W32x.

Although timing of the IXDATA<7:0> and IXMSK signals are fixed with respect to the input SCLK (see timing Section 4.2) the programmer can vary timing of the enable strobe (IXCMD*) via bits in the Configuration Register A. This is useful tuning the IMA signalling for different system board implementations having different trace (or cable) lengths.

Chip level output signal pins from the Image Data Interface unit include:

IXDATA<7:0>
IXCMD*
IXMSK
IXAEN
IXLS
IXFS
IXOF

and the input IXRDY signal.

See Section 3.2 for a fuller description of these pins.

3.0 VIPeR Pinout

VIPeR is a 160-pin plastic quad flat pack. There are 128 signal I/O and 32 power/ground pins. Rev.B VIPeR pin list is as follows. Rev.A VIPeR differences are listed to right.

PIN #	SIGNAL NAME	I/O	Rev. A Signal I/O
1	VDD	-	
2	IXMADL I		
3	MEMWL I		
4	BDB15	I/O	
5	BDB14	I/O	
6	BDB13	I/O	
7	BDB12	I/O	
8	BDB11	I/O	
9	BDB10	I/O	
10	BDB9	I/O	
11	BDB8	I/O	
12	BDB7	I/O	
13	BDB6	I/O	
14	BDB5	I/O	
15	BDB4	I/O	
16	BDB3	I/O	
17	BDB2	I/O	
18	BDB1	I/O	
19	BDB0	I/O	
20	VDD	-	
21	GND	-	
22	SCLK	CLK	
23	GND	-	
24	ISABUS	I	* functionality changed *
25	PCIBUSL	I	
26	LBUSL	I	* functionality changed *
27	RESET	I	
28	AUD7	I	
29	AUD6	I	
30	AUD5	I	
31	AUD4	I	
32	AUD3	I	
33	AUD2	I	
34	AUD1	I	
35	AUD0	I	
36	AUDCLK	I	
37	VCPORT7	I/O	

PIN #	SIGNAL NAME	I/O	Rev. A Signal I/O	PIN #	SIGNAL NAME	I/O	Rev. A Signal I/O
38	VCPORT6	I/O		85	LLCLKIN	I	
39	VCPORT5	I/O		86	HSYNCL	O	
40	VDD	-		87	VSYNCL	O	
41	GND	-		88	FIELD	O	
42	GND	-		89	PSTBL	O	
43	VCPORT4	I/O		90	PRED7	O	
44	VCPORT3	I/O		91	GND	-	
45	VCPORT2	I/O		92	PRED6	O	
46	VCPORT1	I/O		93	PRED5	O	
47	VCPORT0	I/O		94	PRED4	O	
48	RED7	I		95	PRED3	O	
49	RED6	I		96	PRED2	O	
50	GND	-		97	PRED1	O	
51	RED5	I		98	PRED0	I/O	Output only
52	RED4	I		99	PGREEN7	O	
53	RED3	I		100	VDD	-	
54	RED2	I		101	GND	-	
55	RED1	I		102	PGREEN6	O	
56	RED0	I		103	PGREEN5	O	
57	GREEN7	I		104	PGREEN4	O	
58	GREEN6	I		105	PGREEN3	O	
59	GREEN5	I		106	PGREEN2	O	
60	VDD	-		107	PGREEN1	O	
61	GREEN4	I		108	PGREEN0	O	
62	GREEN3	I		109	PBLUE7	O	
63	GREEN2	I		110	PBLUE6	O	
64	GREEN1	I		111	GND	-	
65	GREEN0	I		112	PBLUE5	O	
66	BLUE7	I		113	PBLUE4	O	
67	BLUE6	I		114	PBLUE3	O	
68	BLUE5	I		115	PBLUE2	O	
69	BLUE4	I		116	PBLUE1	O	
70	GND	-		117	PBLUE0	I/O	Output only
71	BLUE3	I		118	MSKADD8	O	
72	BLUE2	I		119	MSKADD7	O	
73	BLUE1	I		120	VDD	-	
74	BLUE0	I		121	GND	-	
75	OFIN	I		122	GND	-	
76	ADCEN	O		123	MSKADD6	O	
77	CSYNC	I		124	MSKADD5	O	
78	ADCCLK	I		125	MSKADD4	O	
79	GND	-		126	MSKADD3	O	
80	GND	-		127	MSKADD2	O	
81	VDD	-		128	MSKADD1	O	
82	FSINL	I		129	VDD	-	
83	LSINL	I		130	GND	-	
84	CLKSYNC	O		131	MSKADD0	O	

PIN #	SIGNAL NAME	I/O	Rev. A Signal I/O
132	MSKRASL	O	
133	MSKCASL	O	
134	MSKWRL	O	
135	MSKOEL	O	
136	MSKD3	I/O	
137	MSKD2	I/O	
138	MSKD1	I/O	
139	MSKD0	I/O	
140	GND	-	
141	VDD	-	
142	IXCMDL O		
143	IXMSK	O	
144	IXOF	O	
145	IXLS	O	
146	IXFS	O	
147	IXRDY	I	
148	IXAEN	O	
149	IXDATA7	O	
150	GND	-	
151	VDD	-	
152	IXDATA6	O	
153	IXDATA5	O	
154	IXDATA4	O	
155	IXDATA3	O	
156	IXDATA2	O	
157	IXDATA1	O	
158	IXDATA0	O	
159	GND	-	
160	GND	-	

* Pins ISABUS, and LBUSL have changed functionality between Rev.A and Rev.B VIPeR. See Section 6.0 for pin difference descriptions

* PRED<0> and PBLUE<0> changed from output only to I/O going from Rev.A to Rev.B VIPeR.

3.1 Power On Reset Initialize (PORI)

During the active to inactive transition of the RESET signal, PORI signals, (PRED<0>, PBLUE<0>) are latched internally. These latched data bits are used to determine the host bus interface write timing. PORI bits are pulled high internally, and can be pulled down via a resistor to ground, or driven low with a tri-state buffer during reset. These two signals, PRED<0>, and PBLUE<0> are PORI bits in Rev.B VIPeR only. There are no PORI bits in Rev.A VIPeR.

Value latched on PRED<0> during reset is used to control the host bus data write timing.

Value latched on PBLUE<0> during reset is used to control the host bus address decode timing during register writes. Sections 3.1 and 4.1 detail the internal functionality of these PORI controls.

3.2 Pin Descriptions

VIPeR is a 160-pin plastic quad flat package chip. There are 132 signal I/O and 28 power/ground pins. They are broken down into ten sections, each described below. All signal pins are TTL compatible. Power is supplied at +5 volts.

3.2.1 Host Bus Interface

SYMBOL	PIN#	I/O	DESCRIPTION
BDB<15:0>	4-19	I/O	Buffered Data Bus <15:0>. Host bus I/O interface. These are the pins VIPeR uses to communicate with the host processor: multiplexed address and data, configurable in 16 bit BDB, or 8 bit DD modes. In DD mode, only lower 8 bits are used. In BDB mode, communication host/VIPeR is done: Address<15:0>/Data<15:0> in two cycles. In DD mode host/VIPeR communication is done on BDB<7:0> in 4 cycles as follows: Address<7:0>/Data<7:0>/Address<15:8>/Data<15:8>. VIPeR contains internal latching to hold the low byte address and low byte data until the high byte information is received, and the host access can execute inside VIPeR.
IXMADL	2	I	Image data port address/data strobe. Signal provided by W32/W32i/W32p. Driven low when host access is to the graphics controller extended memory space. Used by VIPeR as a chip enable for host access on BDB<15:0> bus. Addresses on BDB<15:0> are decoded when IXMADL is high. Data is accessed on BDB<15:0> when IXMADL is low.
MEMWL	3	I	Memory write, active low. CPU memory write signal, driven by W32/W32i/W32p. Used by VIPeR in conjunction with IXMADL, and BDB<15:0> to determine whether a host access to VIPeR is a read or a write access.
ISABUS	24	I	Data latch timing/CPU ISA bus mode** ** Pin has different meanings for Rev.B and Rev.A silicon. In Rev.B VIPeR, pin indicates timing for VIPeR latching of host writes over BDB<15:0> bus. When ISABUS=1, VIPeR Rev.B latches host write data on rising edge of IXMADL. When ISABUS=0, VIPeR Rev.B latches host write data 6ns after falling edge of IXMADL. In Rev.A VIPeR, pin indicates host bus is ISA. When ISABUS=1, VIPeR Rev.A lets MEMWL, memory write indicator, through unaffected. When ISABUS=0, VIPeR Rev.A inverts MEMWL internally.
LBUSL	26	I	Invert reset/CPU local bus mode** ** Pin has different meanings for Rev.B and Rev.A silicon. In Rev.B VIPeR, pin indicates polarity of input RESET signal. LBUSL=0 means input RESET is used directly by VIPeR, (assumed to be low active RESET). When LBUSL=1, VIPeR Rev.B inverts RESET as it comes into chip, (RESET assumed to be high active). In Rev.A VIPeR, pin indicates host bus is VESA Local Bus. When LBUSL=1, VIPeR Rev.A uses addressing information from the host directly, as the register addresses in VIPeR Rev.A. When LBUSL=0, the address decode within VIPeR Rev.A is decoded as CPUAD<0>=BDB<0>, CPUAD<1>=BDB<0> ANDed with BDB<1>.
PCIBUSL	25	I	CPU PCI or (VESA Local) bus mode. Pin indicates mode of BDB<15:0> host interface bus. PCIBUSL=1 indicates operation in 16 bit BDB mode. PCIBUSL=0 indicates operation in 8 bit DD bus mode. In the latter, an 8 bit connection to BDB<7:0> is assumed, and BDB<15:8> are don't care.

3.2.2 ADC Interface

SYMBOL	PIN#	I/O	DESCRIPTION
RED<7:0>	48,49, 51-56	I	Red video input data. When VIPeR is programmed to accept video input from the ADC port, by resetting register bit CPUSYNC to 0, these 8 bits accept the digital RED* component of that video.
GREEN<7:0>	57-59,	I	Green video input data. When VIPeR is programmed to accept video input from the ADC port, these 8 bits accept the digital GREEN* component of that video.
BLUE<7:0>	66-69, 71-74	I	Blue video input data. When VIPeR is programmed to accept video input from the ADC port, these 8 bits accept the digital BLUE* component of that video. * RED/GREEN/BLUE interpretations of data are valid for RGB input modes over the ADC input port. When YUV color mode video is input to the VIPeR ADC input pins, the Y/U/V input video data is wired into the RED/GREEN/BLUE input pins. Specific board level wiring is detailed in Section 2.4.2.
LLCLKIN	85	I	Line Locked Clock input. Clock from the external video A/D converter. Used by VIPeR to latch incoming video data on ADC data input pins. LLCLKIN rising or falling edge can be selected via programming. Also, full or 1/2 LLCLKIN frequency can be selected for ADC video input data rate sampling frequency, via programming of the ADCCLKDIV2 register bit. Maximum frequency of LLCLKIN is 1/2 SCLK. Minimum high time is 1/2 SCLK durations. Minimum low time is 1/2 SCLK duration.
ADCCLK	78	O	Analog to Digital Converter Clock. Output clock which reflects the input LLCLKIN, Line Locked Clock input. The polarity and frequency can be programmed to one of 4 different modes, as detailed in Section 2.2. This ADCCLK output can be used by the external video A/D converter to sample incoming analog video.
ADCEN	76	O	ADC port enable. Reflects register program bit ADCEN logic state. Can be used to tristate outputs of the external video A/D converter.

3.2.3 Video Sync Signals

SYMBOL	PIN#	I/O	DESCRIPTION
CSYNC	77	I	Composite Sync Input. Input from the external video A/D converter. Signal contains horizontal and vertical synchronization information. In interlaced input video mode, it also contains field information. This input is sampled by VIPeR when ADC input and Sync Separator modes are chosen by the programmer.* Horizontal sync information is required at this input when ADC input and Sync Separator bypass modes are chosen by the programmer. This can be accomplished by wiring the line sync output from the external A/D component into CSYNC. * ADC input means register bit CPUSYNC=0 * Sync Separator chosen means register bit SSBYPASS=0

SYMBOL	PIN#	I/O	DESCRIPTION
FSINL	82	I	Field Sync Input, active low. Input from the external video A/D converter. Signal contains vertical sync information from an external sync separator, contained within the video A/D components. When Sync Separator is bypassed, SSBYPASS=1 programmed, this FSINL is used by VIPeR as the source of vertical sync information, rather than the vertical sync generated by VIPeR's internal sync separator.
LSINL	83	I	Line Sync Input, active low. Input from the external video A/D converter. Signal contains horizontal sync information from an external sync separator, contained within the video A/D components. When Sync Separator is bypassed, SSBYPASS=1 programmed, this LSINL is used by VIPeR as the source of horizontal sync information, rather than the horizontal sync generated by VIPeR's internal sync separator.
OFIN	75	I	Odd field input indicator. Input from the external video A/D converter. Signal contains odd/even field information from an external sync separator, contained within the video A/D components. When Sync Separator is bypassed, SSBYPASS=1 programmed, this OFIN is used by VIPeR as the source of odd/even field information, rather than the field indicator generated by VIPeR's internal sync separator. For non-interlaced input video, OFIN is low.
VSYNCL	87	O	Vertical Sync output, active low. VIPeR vertical sync output pulse. Source is the internal vertical sync signal. This internal sync signal is derived from one of 3 sources: the CPUFS register bit in CPU input video mode; the sync separated vertical sync in ADC video input, and SSBYPASS=0 mode; and the FSINL pin in ADC video input, SSBYPASS=1 mode.
HSYNCL	86	O	Vertical Sync output, active low. VIPeR horizontal sync output pulse. Source is the internal horizontal sync signal. This internal sync signal is derived from one of 3 sources: the CPULS register bit in CPU input video mode; the sync separated vertical sync in ADC video input, and SSBYPASS=0 mode; and the LSINL pin in ADC video input, SSBYPASS=1 mode. In any of the three modes, the HSYNCL output can be disabled during vertical blanking time, or not, by programming of GSELSYNC register bit (Register 0, bit 6).

3.2.3 Video Sync Signals (cont'd)

SYMBOL	PIN#	I/O	DESCRIPTION
FIELD	88	O	Odd Field output, active high. VIPeR odd/even field indicator. Source is the internal odd/even field indicator signal. This internal signal is derived from one of 3 sources: the CPUOF register bit in CPU input video mode; the sync separated vertical sync in ADC video input, and SSBYPASS=0 mode; and the OFIN pin in ADC video input, SSBYPASS=1 mode.
CLKSYNC	84	O	Line locked clock sync pulse. Pulse generated by falling edge of the currently selected HSYNCL. It can be used to control an external gated oscillator, so that it goes inactive during blanking times.

3.2.4 Processed Data Output

SYMBOL	PIN#	I/O	DESCRIPTION
PRED<7:0>	90, 92-98	O	Processed RED video output. Eight bit RED* channel output of the VIPeR horizontal and vertical scaling units. This data is fed (internally) into the Image Data Port (W32) interface unit, formatted, FIFO'd, time multiplexed with the green/blue pixel components and output over IXDATA<7:0> to the Image port. * If YUV formatted video is input to VIPeR, and the YUVEN register bit is reset to 0, then full resolution, Y<7:0> data will be output on PRED<7:0>.
PGREEN<7:0>	99, 102-108	O	Processed GREEN video output. Eight bit GREEN* channel output of the VIPeR horizontal and vertical scaling units. This data is fed (internally) into the Image Data Port (W32) interface unit, formatted, FIFOed, time multiplexed with the red/blue pixel components and output over IXDATA<7:0> to the Image port. * If YUV formatted video is input to VIPeR, and the YUVEN register bit is reset to 0, then full resolution, V<7:0> data will be output on PGREEN<7:0>.
PBLUE<7:0>	109,110 112-117	O	Processed BLUE video output. Eight bit BLUE* channel output of the VIPeR horizontal and vertical scaling units. This data is fed (internally) into the Image Data Port (W32) interface unit, formatted, FIFOed, time multiplexed with the red/green pixel components and output over IXDATA<7:0> to the Image port. * If YUV formatted video is input to VIPeR, and the YUVEN register bit is reset to 0, then full resolution, U<7:0> data will be output on PBLUE<7:0>.
PSTBL	89	O	Processed data output enable strobe, low active. Signal identifies valid output pixel data on PRED<7:0>,PGREEN<7:0>,PBLUE<7:0>. PSTBL transitions LOW, for one SCLK for each new output pixel. PSTBL goes low for 15ns after rising edge of SCLK.

3.2.5 Audio* Input

***NOTE:** Although called audio input, this bus interface can be used for any (relatively slower) non video pixel data stream. SMPTE code is a non audio data example.

SYMBOL	PIN#	I/O	DESCRIPTION
AUD<7:0>	28-35	I	Audio data inputs. When VIPeR is programmed to output audio data, via setting register bit AUDIOEN, digital data streams into VIPeR via these 8 pins.
AUDCLK	36	I	Audio data input clock. Audio data clock. Maximum frequency, 10MHz.

3.2.6 Video Control Port

SYMBOL	PIN#	I/O	DESCRIPTION
VCPORT<7:0>	37-39 43-47	I/O	Video Control Port I/O Configurable via register bits as inputs or outputs (on a bit by bit basis). These general purpose I/O pins reflect logic state of internally set register bit (when configured as output). In the output case, data written by host CPU to corresponding register bit is driven out of that VCPORT pin. In input case, board logic level driven into that VCPORT pin, can be read on corresponding register bit, by CPU host reads. A variety of board level controls can be handled using VCPORT. Examples are, control of an I(squared)C bus, programming of external A/D converter, setting and reading various video board level logic states, etc.

3.2.7 Image Data Port

SYMBOL	PIN#	I/O	DESCRIPTION
IXDATA<7:0>	147, 152-158	O	Image Data Port pixel data.. This is the output pixel data stream which travels over the Image Connector, via Tseng Labs' Image Memory Access (IMA) protocol, to the W32/W32i/W32p, and finally, into the graphics frame buffer. An eight bit output, data is time multiplexed based on programmed output format: two 8-bit transfers for 15 and 16 bit/pixel output formats, three 8-bit transfers for 24 bit/pixel output format.
IXMSK	143	O	Image Data Port mask control signal. One bit associated with each pixel output over IXDATA<7:0>, which functions effectively as a one bit alpha channel for overlaying video and graphics. Tells the W32/W32i/W32p whether the corresponding video pixel should be displayed, (if IXMSK=1), or if the underlying graphics pixel should be displayed in place of this video pixel (if IXMSK=0).
IXCMDL	142	O	Image Data Port data enable strobe, low active. Output strobe identifies when a valid byte of data is present on IXDATA<7:0>. Used by W32/W32i/W32p to latch IXDATA<7:0> data, and to increment IMA video counter. IXCMDL remains active, LOW, for 1/2 SCLK duration. Polarity and timing of IXCMDL are widely tunable via register program settings. Register bits used to control these are: CMDPOL, CMDCK<2:0>, CMDCKDIV2<2:0>.

3.2.7 Image Data Port (cont'd)

SYMBOL	PIN#	I/O	DESCRIPTION
IXRDY	147	I	Image Data Port ready signal. Output from W32/W32i/W32p, tells VIPeR that it can send the next byte of data on IXDATA<7:0>. When IXRDY=0, VIPeR must wait the sending of next pixel byte on IXDATA<7:0> until it sees IXRDY=1 again.
IXFS	146	O	Image Data Port frame sync. Active during beginning of video vertical blanking time. Tells W32/W32i/W32p to reset its IMA video address pointers back to top left corner of video window.
IXLS	145	O	Image Data Port line sync. Active during beginning of video horizontal blanking time. Tells W32/W32i/W32p to reset its IMA video address pointers back to beginning of next video line.
IXOF	144	O	Image Data Port odd field. Active during beginning of video vertical blanking time. Tells W32/W32i/W32p to reset its IMA video address pointers back to top left corner of even or odd field video window.,
IXAEN	148	O	Image Data Port tri-state enable. High active output, indicates that VIPeR Image Data Port signals are enabled. When IXAEN=0, indicates that VIPeR Image Data Port signals are tri-stated.

3.2.8 Overlay Mask Control

SYMBOL	PIN#	I/O	DESCRIPTION
MSKADD<8:0>	118,119 123-128 131	O	Overlay Mask Control, DRAM address. DRAM address for accessing external, 256K x 4, optional overlay DRAM.
MSKRASL	132	O	Overlay Mask Control, DRAM RAS, low active. DRAM RAS control for accessing external, 256K x 4, optional overlay DRAM.
MSKCASL	133	O	Overlay Mask Control, DRAM CAS, low active. DRAM CAS control for accessing external, 256K x 4, optional overlay DRAM
MSKWRL	134	O	Overlay Mask Control, DRAM WRITE, low active. DRAM read/write control for accessing external, 256K x 4, optional overlay DRAM.
MSKOEL	135	O	Overlay Mask Control, DRAM output enable, low active. DRAM output enable control for accessing external, 256K x 4, optional overlay DRAM.
MSKD<3:0>	136-139	I/O	Overlay Mask Control, DRAM I/O data lines. DRAM input/output data lines for accessing external, 256K x 4, optional overlay DRAM

3.2.9 System and Configuration

SYMBOL	PIN#	I/O	DESCRIPTION
SCLK	22	I	System clock. Main VIPeR input system clock. Maximum frequency is lesser of 50MHz, or W32/W32i/W32p system clock. Duty cycle must be between 40/60 and 60/40 percent.
RESET	27	I	Hardware reset Main VIPeR hardware reset. Polarity is programmable via LBUSL (VIPeR Rev.B) only. LBUSL=0 corresponds to low active reset: LBUSL=1 corresponds to high active reset.

3.2.10 Power and Ground

SYMBOL	PIN#	I/O	DESCRIPTION
VDD	1,20 40,60 81,100 120,129 141,151	I	+5 volt VIPeR power supply.
GND	21,23 41,42 50,70 79,80 91,101 111,121 122,130 140,150 159,160	I	GROUND.

3.3 VIPeR Rev. A versus Rev. B Pinout Differences

The VIPeR Rev.A versus Rev.B are in identical packages, with identical footprints. Names of all the pins are identical. There are, however, four pins which are different between the Rev.A and Rev.B silicon. Two of the pins (both system configuration inputs) have different functionality. The other two have changed from output only to PORI I/O pins. The details are described below.

1. Function of pins ISABUS, and LBUS* have changed.

Rev. A definition:

ISABUS = ISA host CPU bus connection

ISABUS=1: 16 bit buffered data bus host bus connection
MEMW* input is low active
ISABUS=0: 8 bit DD host bus connection
MEMW* input is high active

LBUS* = VESA Local bus mode

LBUS*=1: during host access address, DD<1> used directly as the VIPeR address bit<1>
LBUS*=0: during host access address, DD<1> and DD<0> are ANDed together internally, and used as the VIPeR address bit<1>

Rev. B definition:

LBUS* = invert reset

LBUS*=1: input RESET* is inverted internally
LBUS*=0: input RESET* is not inverted, but used directly

ISABUS = delay latching of data on BDB during VIPER register access

ISABUS=0: data on BDB<15:0> is latched 6 (+/-2) ns after falling edge IXMAD
ISABUS=1: data on BDB<15:0> is latched 0 (+/-2) ns after rising edge IXMAD

2. PRED<0>, PBLUE<0> outputs have additional functionality.

Upon RESET going inactive, Rev.B latches values on PRED<0> and PBLUE<0> and uses these two values to control the choice of BDB address and data register access timing. Programmable with jumperable weak pulldowns on PRED<0> and PBLUE<0>; latching of values on these pins leads to generation of two internal signals (in combination with PCIBUS* input), EARLYADDRESS and EARLYDATA. Values of these are determined by:

EARLYADDRESS = PCIBUS* (exclusive-or'ed) with latched PBLUE<0>
EARLYDATA = PCIBUS* (exclusive-or'ed) with latched PRED<0>

if EARLYADDRESS = 0: BDB address timing same as VIPER-A wrt IXMAD
if EARLYADDRESS = 1: internal BDB address path sped up ~6ns (expected address later on BDB wrt IXMAD)

if EARLYDATA = 0: BDB data latched with same timing as VIPER-A
if EARLYDATA = 1: internal BDB data path up ~6ns (expected data later on BDB wrt IXMAD)

NOTE: See timing diagram in Section 4.1 for details of host bus timings, and how they are quantitatively effected by PRED<0> and PBLUE<0>

3.4 Electrical Specifications

Maximum Ratings

Storage temperature	-40 to + 125 deg. C
Operating free-air temperature range	0 to +70 deg. C
Supply voltage applied to ground potential	-0.5 to +7.0 V
DC voltage applied to outputs for high output state	-0.5 to Vdd max.
DC input voltage	+4.75V to +5.25 V
Supply current	200mA typ / 300 max.

3.4.1 Electrical Characteristics

The following condition applies unless otherwise specified:

T(A) = 0 + 70 deg. Vdd = 5.0 V ± 5%

DC Characteristics Over Operating Temperature

Symbol	Parameter	Condition	Min.	Typ.	Max.	Unit
V_{IH}	High level input voltage					V
	TTL		2.0			
	CMOS SCHMITT trigger		4.0			
V_{IL}	Low level input voltage					V
	TTL				0.8	
	CMOS level SCHMITT trigger				1.0	
I_{IH}	High level input current	$V_{IN} = V_{OD}$	-10		10	µA
	Input buffer with pull-down		10		200	
I_{IL}	Low level input current	$V_{IN} = V_{SS}$	-10		10	µA
	Input buffer with pull-up		-200		-10	
V_{OH}	High level output voltage					V
	BUFFER B2	I_{OH} = -2 mA	2.4			
	BUFFER B4	I_{OH} = -4 mA				
	BUFFER B8	I_{OH} = -8 mA				
		I_{OL} = 1 µA	V_{DD} -0.05			
V_{OL}	Low level output voltage					
	BUFFER B2	I_{OL} = 2 mA	2.4			
	BUFFER B4	I_{OL} = 4 mA				
	BUFFER B8	I_{OL} = 8 mA				
		I_{OL} = 1 µA			0.05	
I_{OZ}	High impedance leakage current	$V_{OUT} = V_{DD}$ or V_{SS}	-10		10	µA
	Output buffer with pull-up		-200		-10	
	Output buffer with pull-down		10		200	
VH	SCHMITT trigger hysteresis voltage					V
	CMOS			0.6		

4.0 VIPeR I/O Timing Specification

Timing of the input/output signal pins is described in the following sections. These include signals for each of these major external interfaces:

Host Bus
Image Data Port
ADC input
Sync Signals
Processed data output
Video Control Port
Audio
Overlay Mask DRAM control

4.1 Host Bus Signal Timing

BDB(15:0) is the multiplexed address/data I/O host bus of the VIPER. Host CPU register access to VIPeR is controlled by two input pins, IXMADL, and MEMWL. Address and data of the register access is delivered over the BDB<15:0> lines. In 16-bit BDB mode, as chosen by setting input pin PCIBUSL high, all 16 bits of BDB<15:0> are used for host CPU accesses to VIPeR. In 8-bit DD bus mode, as chosen by setting PCIBUSL low, only BDB<7:0> are used. In either case, the timing specifications are the same.

IXMADL determines the address versus data phase of the register access. In general, VIPER decodes addresses on BDB(15:0) when IXMADL is high, and latches corresponding data on BDB(15:0) some time after IXMADL transitions to a low. BDB<15:8> is ignored by the VIPeR when in DD bus mode.

MEMWL determines if the access is a register read, or a register write. MEMWL low is a write, MEMWL high is a read. MEMWL timing is such that is must be valid for the entire time IXMADL is low, and be setup at least 5 nanoseconds before the falling edge of IXMADL, and held at least 5 nanoseconds after IXMADL rises.

Host bus timing for VIPeR rev B silicon is a superset of the host bus timings of VIPeR Rev.A silicon. PORI pin functionality, PRED<0> and PBLUE<0> is added in Rev.B silicon which allows different bus timings to be programmed on the video board, via jumper settings. System designer either leaves jumpers on these PORI pins out, in which case the internal pull ups take effect, or installs jumpers to resistive ground, effectively toggling the POR state set within the VIPeR. These settings add flexibility for video board designs with different layouts, and different pin timings. Also, ISABUS input has changed its functionality between Rev.A and Rev.B. In Rev.B, it is used to determine the rising or falling edge of XMADL to use to latch the data (on BDB<15:0>)into VIPeR during write accesses.

There are 8 basic modes of operation chosen by high or low values programmed into the 2 PORI pins on reset, and the logic state of ISABUS input pin. In addition, polarity of the values set on reset are effected by the PCIBUSL input polarity as follows:

EARLYADDRESS = PCIBUSL (exclusive-OR'd) with latched, PORI PBLUE<0> value
EARLYDATA = PCIBUSL (exclusive-OR'd) with latched, PORI PRED<0> value
LREDGE IXMAD = ISABUS

LREDGE IXMAD means data is written into VIPeR on rising edge of IXMADL for host CPU writes, when ISABUS=1. Data is written into VIPeR some time after the falling edge of IXMADL when ISABUSL=0.

Mode#	EARLYADDRESS	EARLYDATA	LREDGE IXMAD
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	0
4	0	0	1
5	0	1	1
6	1	0	1
7	1	1	1

All 8 modes are available for VIPeR Rev.B silicon. Mode 0 only is available with VIPeR Rev.A silicon.

Following is timing specification in all 8 modes.

4.1.1 Host Write Access

	mode0	mode1	mode2	mode3
address setup wrt falling edge IXMADL	+6ns	+6ns	2ns	2ns
address hold wrt falling edge IXMADL	0ns	0ns	4ns	4ns
data setup wrt falling edge IXMADL	5ns	1ns	5ns	1ns
data hold wrt falling edge IXMADL	11ns	7ns	11ns	7ns

	mode4	mode5	mode6	mode7
address setup wrt falling edge IXMADL	+6ns	+6ns	2ns	2ns
address hold wrt falling edge IXMADL	0ns	0ns	4ns	4ns
data setup wrt rising edge IXMADL	-2ns	+6ns	-2ns	+6ns
data hold wrt rising edge IXMADL	4ns	0ns	4ns	0ns

4.1.2 Host Read Access

In each case, address setup/hold is identical to host write access timings. Since VIPeR is delivering data, on IXMADL low transition of appropriate functional cycle, only relevent timing is when VIPeR delivers read data back to BDB<15:0> (or BDB<7:0>).

This time is 8 (+/- 3) nanoseconds after falling edge of IXMADL. This is the same in all 8 timing modes.

**RECOMMENDATION: mode 5 is recommended as default.

For PCI and VESA Local bus host bus systems, set

PCIBUSL=0
PBLUE<0> jumpered through 33ohm resistor to gnd
PRED<0> floating high
ISABUS=1.

For ISA bus host bus systems, set

PCIBUSL=1
PBLUE<0> jumpered through 33ohm resistor to gnd
PRED<0> floating high
ISABUS=1.

4.1.3 IXMADL Timing

Maximum effective frequency: 1/2 SCLK.
Minimum IXMADL high time: 1 SCLK time duration
Minimum IXMADL low time: 1 SCLK time duration

4.1.4 MEMWL Timing

4.2 Image Data Port Signal Timing

The Image Data Port is composed of the following 13 output signals:

IXDATA<7:0>
IXCMDL
IXMSK
IXAEN
IXLS
IXFS
IXOF

and the input IXRDY signal.

See Section 2.0.10 for a functional description of the VIPeR Image Data Port unit, and Section 3.2 for a description of these pins.

4.2.1 IXDATA<7:0>, IXMSK Timing

IXDATA<7:0> switches 10 (+/-3) nanoseconds after rising edge of SCLK
IXMSK switches 8 (+/-2) nanoseconds after rising edge of SCLK

Both IXDATA<7:0> and IXMSK are latched by the W32/W32i/W32p on the rising edge of IXCMDL. Timing skew of IXDATA<7:0> and IXMSK will generally track together. That is, if IXDATA<7:0> switches earlier than the nominal 10ns after SCLK, IXMSK will also switch earlier than its nominal 8ns after SCLK. Therefore, IXMSK will always toggle 2 (±1) nanosecond before IXDATA<7:0>.

4.2.1 IXCMDL Timing

Based on W32/W32i/W32p requirements for IXDATA<7:0>, IXMSK versus IXCMDL, and the IMA connector board design, IXCMDL timing can be changed via VIPeR register programming as follows.

The nominal delay for IXCMDLL versus SCLK is tuned via programming bits (7,5:0) in Configuration Register A. These bits choose from positive and negative polarity, and divide by two or direct, different timing versions of SCLK, as described below. Result of the clock chosen creates an internal signal called CMDCLK. IXMADL toggles 9 (+/- 2) nanoseconds after CMDCLK rises.

IXCMDL timing is controlled by setting bits (7, 5:0) in Configuration Register A, address x02, as follows:

bit <7> = CMDPOL
bits<5:3> = CMDCK<2:0>
bits<2:0> = CMDCKDIV2<2:0>

CMDCKDIV2<2:0> - creates internal signals D2CLK, ID2CLK, by first choosing an internal clock signal (D2GENERATE) by following mux select.

CMDCKDIV2<2:0>	D2GENERATE
000	reserved
001	SCLK
010	ASCLK
011	DSCLK
100	reserved
101	ISCLK
110	IASCLK
111	IDSCLK

Then, D2GENERATE clocks a "divide by 2" flip flop circuit. The output of this flip flop is D2CLK

ID2CLK always equals inverted D2CLK

D2CLK and ID2CLK are toggle states: one SCLK cycle on, one SCLK cycle off, etc. Phasing of the toggle points are determined by D2GENERATE.

CMDCK<2:0> - generates the internal signal CMDCLK by following mux select:

CMDCK<2:0>	CMDCLK
000	D2CLK
001	SCLK
010	ASCLK
011	DSCLK
100	ID2CLK
101	ISCLK
110	IASCLK
111	IDSCLK

CMDPOL- chooses the polarity of IXCMDL output, by following:

CMDPOL=0, IXCMDL= CMDCLK "ANDed with" IXRDY (synchronized ready signal from W32p)

CMDPOL=1, IXCMDL= CMDCLK "OR'd with" inverted IXRDY (synchronized ready signal from W32p)

Other internal signals used above are:

SCLK - the main internal clock: buffered version of 50MHz input clock
ASCLK - internal "early" clock. 1.0 to 1.5ns EARLIER than SCLK
DSCLK - internal "delayed" clock. 1.0 to 1.5ns LATER than SCLK
ISCLK - inverted SCLK

IASCLK - inverted ASCLK
IDSCLK - inverted DSCLK

Programming example:

Setting CMDPOL=0, RDYCK=xxx, CMDCK=001, leads to IXCMDL timing similar to the IXDATA outputs. Changing CMDCK setting to 011, now delays IXCMDL by 1.0 to 1.5 nanoseconds. This delay is due to the difference between CMDCLK referenced to SCLK versus CMDCLK referenced to DSCLK.

CMDCK<2:0>	CMDCKDIV2<2:0>	CMDPOL	SCLK → IXCMDL delay(ns)
000	001	0	10 (+/- 2)
000	010	0	9 (+/- 2)
000	011	0	11 (+/- 2)
000	101	0	20 (+/- 2)
000	010	0	19 (+/- 2)
000	111	0	21 (+/- 2)
001	XXX	0	9 (+/- 2)
010	XXX	0	8 (+/- 2)
011	XXX	0	10 (+/- 2)
100	001	0	20 (+/- 2)
100	010	0	19 (+/- 2)
100	011	0	1 (+/- 2)
100	101	0	10 (+/- 2)
100	010	0	9 (+/- 2)
100	111	0	11 (+/- 2)
101	XXX	0	19 (+/- 2)
110	XXX	0	18 (+/- 2)
111	XXX	0	20 (+/- 2)

** for each of the values listed above, changing cmdpol to 1 has the effect of changing the SCLK--> IXCMDL time by exactly 1/2 SCLK time.

NOTE: CMDCK<2:0>=000 or 100 chooses DSCLK or IDSCLK (divide by two) versions of the internal SCLK. Timing is as specified above, however, it should be noted, that the maximum IXMADL frequency in these modes is 1/2 SCLK, instead of SCLK frequency.

4.2.2 Other Output Signal Timing

IXAEN - static output; a reflection of programmed bit 0 in Control Register.

IXLS - toggles 4 (+/- 2) nanoseconds after rising edge of SCLK. When activated, remains high for 7 SCLK cycles

IXFS - timing is function of programmed state. If in CPUSYNC mode, IXFS toggles 4 (+/- 2) nanoseconds after the SCLK immediately following the toggling of CPUFSL in Control Register. In ADC input mode, sync separator enabled mode, IXFS reflects the FSINL input, toggling 5 (+/- 2) nanoseconds after FSINL toggles. In ADC input, sync separator bypass mode, IXFS toggles 5 (+/- 2) nanoseconds after SCLK.

IXOF - similar to IXFS in all modes

4.2.3 IXRDY Input Signal Timing

IXRDY is an asynchronous input to the VIPeR. Minimum pulse duration is 10 nanoseconds.

4.2.4 Minimum Pulse Width Duration

IXDATA<7:0> - minimum valid time: SCLK period minus 6ns
IXMSK - minimum valid time: SCLK period minus 6ns
IXCMDL - minimum valid time: 1/2 SCLK period minus 1ns

IXAEN - N/A
IXLS - minimum valid time: 7 SCLK periods minus 4ns
IXFS - minimum valid time: FSINL duration minus 2ns
IXOF - minimum valid time: OFIN duration minus 2ns

IXRDY - 10ns

4.3 ADC Interface Timing

The ADC Interface is composed of RED<7:0>, GREEN<7:0>, BLUE<7:0>, and LLCLKIN inputs: ADCCLK and ADCEN outputs.

RED<7:0>, GREEN<7:0>, BLUE<7:0> data input to VIPER is latched with respect to LLCLKIN signal. Data can be latched with respect to EITHER the rising or falling edge of LLCLKIN, as programmed by ADCLATPOL in Configuration Register B. If this bit is set to 1, the falling edge of LLCLKIN is used to latch input video data on the 24 bit ADC inputs. Set to 0 and the rising edge of LLCLKIN is used.

LLCLKIN has a maximum frequency of 1/2 SCLK frequency. Duty cycle must remain within 40/60 or 60/40 (percent high/low) times. It can operate either synchronously or asynchronously with SCLK. In either programmed case (rising or falling edge), the data has a setup time of ZERO nanoseconds, and a hold time requirement of 8 nanoseconds.

ADCCLK is a VIPER output whose polarity is also programmable via ADCCLKPOL in Configuration Register B. This output signal is derived from LLCLKIN. It toggles 5 (+/- 3) nanoseconds after LLCLKIN.

The ADCEN output is static, reflecting the programmed logic state of register ADCEN, bit 5 in Configuration Register B.

4.4 Sync Signal Timing

The sync signals are the inputs: CSYNCL, FSINL, LSINL, OFIN, and the outputs: VSYNCL, HSYNCL, FIELD, CLKSYNC.

CSYNCL, FSINL, LSINL, and OFIN have no timing spec versus any of the input clocks. They can operate synchronously, or asynchronously with either SCLK or LLCLKIN.

CSYNC contains the horizontal and vertical synchronization information from an external video A/D component. It may also contain field information for interlaced video sources. Typical CSYNC signals look like figure 4.4.1.

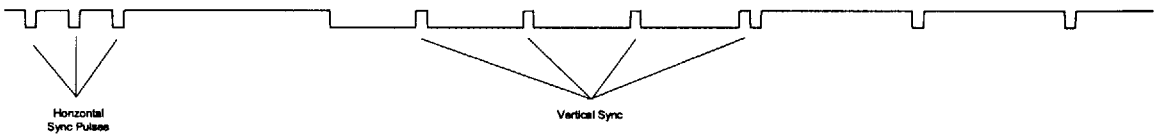


Figure 4.4.1 CSYNC Signal

Minimum low time for the horizontal sync pulses on CSYNC is 2*SCLK. Maximum low time, and min/max high times for the horizontal sync pulses on CSYNC is determined by the video format requirements: horizontal frequency, active video display times, front/back porches, etc.

Minimum low time for the vertical sync on CSYNC is determined by programmed settings in the Vertical Sync Detector Width Register. Value is programmed in SCLK cycles minus 1.

FSINL, LSINL, and OFIN contain vertical, horizontal, and field information when the internal VIPeR sync separator is bypassed (by setting SSBYPASS in Configuration Register B). Minimum active time for FSINL and OFIN is 5*SCLK. Minimum active time for LSINL is 3*SCLK. Maximum times are determined by the video format requirements: horizontal frequency, active video display times, front/back porches, etc.

The VSYNCL, HSYNCL, and FIELD outputs have different timing references based on the programmed sync source selected. There are three sources: the CPU sync signal in Control register, in CPU input video mode, the sync separator generated in ADC video in and SSBYPASS=0 mode, and the FSINL/LSINL/OFIN inputs in ADC video and SSBYPASS=1 mode.

For the first, timing of VSYNCL/HSYNCL/FIELD is determined by the timing of the register write to the Control Register. This happens asynchronously to SCLK. For the second, timing of VSYNCL/HSYNCL/FIELD is referenced from SCLK. Toggling occurs 8 (+/- 4) nanoseconds after rising edge of SCLK. For the third, VSYNCL/HSYNCL/FIELD is referenced from the FSINL/LSINL/OFIN inputs. Outputs toggle 6 (+/- 4) nanoseconds after the corresponding input transitions.

The CLKSYNC output toggles in response to falling edge of the currently selected live video horizontal sync input: CSYNC if SSBYPASS=0, LSINL if SSBYPASS=1. The polarity is programmable via CLKSYNCPOL in Configuration Register B. CLKSYNC toggles 25 (+/- 10) nanoseconds after CSYNC or LSINL. It remains active for 20 (+/- 8) nanoseconds. See Figure 4.4.2

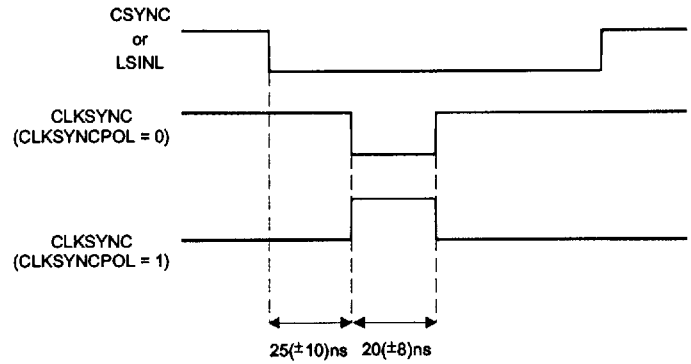


Figure 4.4.2 CLKSYNC Timing

4.5 Processed Data Output Timing

The processed data output signals are PRED<7:0>, PGREEN<7:0>, PBLUE<7:0>, (hereafter known as PR/PG/PB) and PSTBL.

PR/PG/PB are the 24 bit output pixel data signals. PSTBL is the data valid strobe.

All signals are referenced to SCLK.

PSTBL toggles 11 (± 5) nanoseconds after the rising edge of SCLK PR/PG/PB toggles 13 (± 6) nanoseconds after the rising edge of SCLK

PSTBL hold time after the rising edge of SCLK is 4 (± 2) nanoseconds PSTBL setup time before next rising edge of SCLK is SCLK - [11 (± 5)] nanoseconds

PR/PG/PB hold time after rising edge of SCLK is 4 (± 2) nanoseconds PR/PG/PB setup time before next rising edge of SCLK is SCLK - [13 (± 6)] nanoseconds

Figure 4.5.1 shows the timing of these signals in maximum pixel output rate mode (as during video vertical expand during generated video lines), and in slower output rate modes.

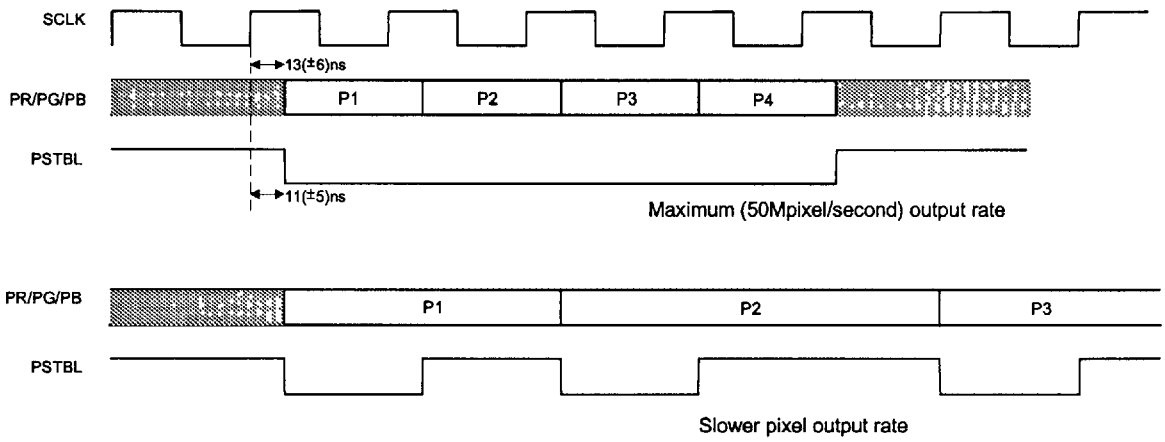


Figure 4.5.1 Processed Data Output Timing

4.6 Video Control Port Timing

The video control port I/O signals are VCPORT<7:0>.

They have no timing specification versus any of the input clocks, SCLK, or LLCLKIN. These bits are static: outputting data written into the VCPORT register when configured as output, accepting logic state into a host CPU readable register when configured as input.

VCPORT output configured pins change 8 (+/- 4) nanoseconds after the VCPORT register has been written with different data, from the host CPU. Timing of this register bit writing is referenced to the host bus address/data input strobe, IXMADL.

Given that the maximum host access rate is 1/2 SCLK, the VCPORT register bits can change every host access cycle. Direction (input or output) of each of the VCPORT pins can also change at every host access cycle. Typical uses of the VCPORT, however, will set different bits of VCPORT<7:0> as either input or output, for entire applications. That is, the input/output nature of the pins will not change.

External logic changes on VCPORT bits (configured as inputs) is available for host CPU reads 5 (+/- 2) nanoseconds after the logic change on the video board. The video board logic change, and the host access timing is presumably asynchronous.

Data driven out of VCPORT pins (configured as outputs) does toggle synchronously with the host CPU access. Exact timing of the data change, with respect to IXMADL, is a function of the host bus timing programmed state, (see Section 4.1). VCPORT pins toggle 8 (+/- 4) nanoseconds after the VCPORT register has been written.

Choices for writing of the VCPORT register are:

ISABUS=0: 7 (+/- 3) nanoseconds after falling edge of IXMADL (with MEML=0)
ISABUS=1: 2 (+/- 2) nanoseconds after rising edge of IXMADL (with MEML=0)

4.7 Audio Interface Timing

The Audio Interface signals are AUD<7:0> inputs, and AUDCLK input.

They have no timing specification versus any of the input clocks, SCLK, or LLCLKIN.

The AUDCLK input must toggle at a rate no greater than 1/2 SCLK. Data on AUD<7:0> is latched into VIPeR on every rising edge of AUDCLK. In design of an audio interface, care must be given to output data rates on the Image Data Port. Programmable register settings define how much of the audio data will be output per video scan line. In order to ensure no data FIFO overruns, the AUDCLK frequency should be no greater than the maximums set by the following three equations.

1. AUDCLK < 1/2 SCLK

2. AUDCLK frequency < (Audio Bytes per Line) * (Video Scan Line frequency)

    where: (Audio Bytes per Line) = (Maximum Bytes per Line) - (Video Bytes per Line)

3. AUDCLK frequency < (128) * (Video Scan Line frequency)

The duty cycle of AUDCLK should be between 80/20 and 20/80 (percent high/low time).

AUD<7:0> data input is referenced to rising edge of AUDCLK.

AUD<7:0> setup time is 4 nanoseconds before rising edge of AUDCLK.
AUD<7:0> hold time is 4 nanoseconds after rising edge of AUDCLK.

4.8 Mask DRAM Interface Timing

The Mask DRAM Interface signals are outputs:

MSKADD<8:0>
MSKRASL
MSKCASL
MSKWRL
MSKOEL

and the I/O signals MSKD<3:0>.

MSKADD<8:0>, MSKRASL, MSKCASL, (output) MSKD<3:0> have timing referenced to SCLK.

Rising edges of MSKWRL, and MSKOEL, have timing referenced to rising edge of SCLK. Falling edges of MSKWRL, and MSKOEL, have timing referenced to IXMADL, host bus address/data phase indicator, as these are effected by host CPU access. In sequential writes, with CPU not toggling write or output enable, then falling edges are also referenced to SCLK.

Figures 4.8.1 and 4.8.2 show functional timing diagrams of mask dram signal operation.

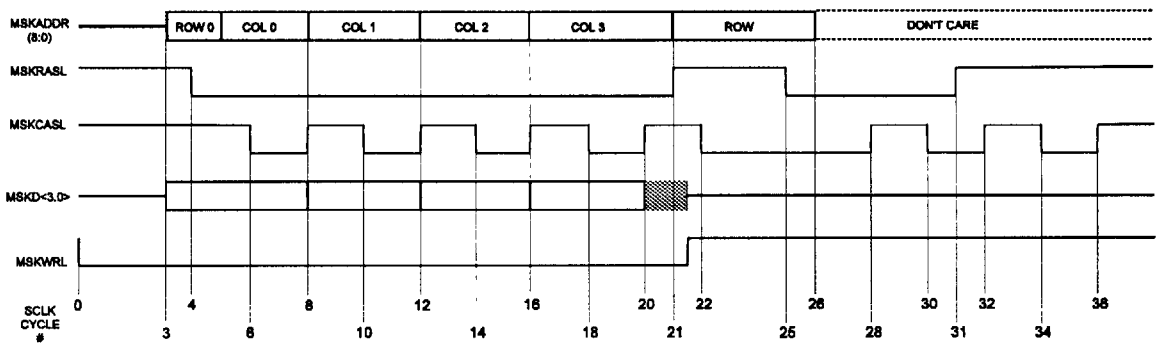


Figure 4.8.1 Mask DRAM Functional Write Timing Diagram

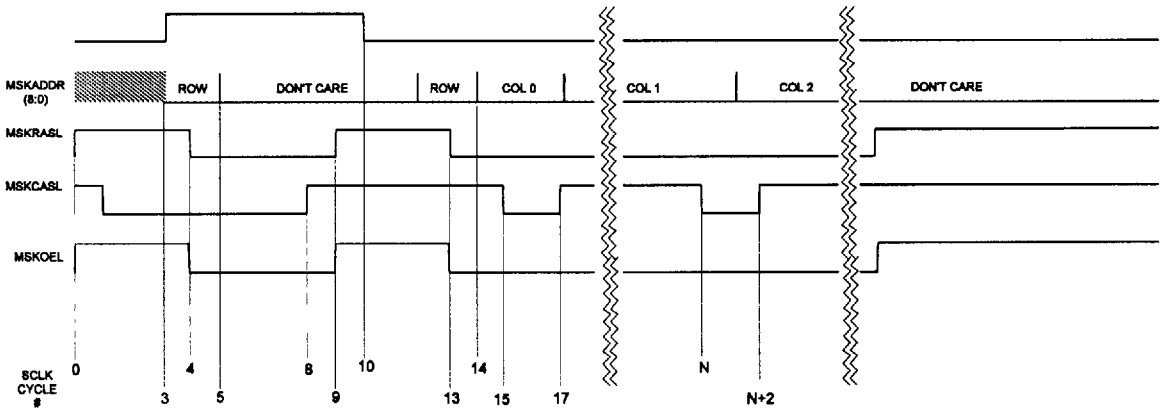


Figure 4.8.2 Mask DRAM Functional Read Timing (Active Display Time)

Toggle times for output signals, with respect to SCLK, are as follows:

MSKADDR<8:0>	toggles 13 (±5) nanoseconds after rising edge SCLK
MSKRASL	toggles 11 (+2/ -4) nanoseconds after rising edge SCLK
MSKCASL	toggles 11 (+2/ -4) nanoseconds after rising edge SCLK
MSKWRL	toggles 13 (±5) nanoseconds after rising edge SCLK
MSKOEL	toggles 14 (±5) nanoseconds after rising edge SCLK

When writing data to external overlay DRAM, MSKD<3:0> I/O lines are driven out of VIPeR with following timing:

MSKD<3:0> toggles 14 (+5/ -8) nanoseconds after rising edge SCLK

Upon reading data from external overlay DRAM, during video active display times, MSKD<3:0> is read back by VIPeR. Required setup/hold times for the data are specified with respect to the MSKCASL output.

MSKD<3:0> data setup = 4 nanoseconds before rising edge of MSKCASL
MSKD<3:0> data hold = 4 nanoseconds after rising edge of MSKCASL

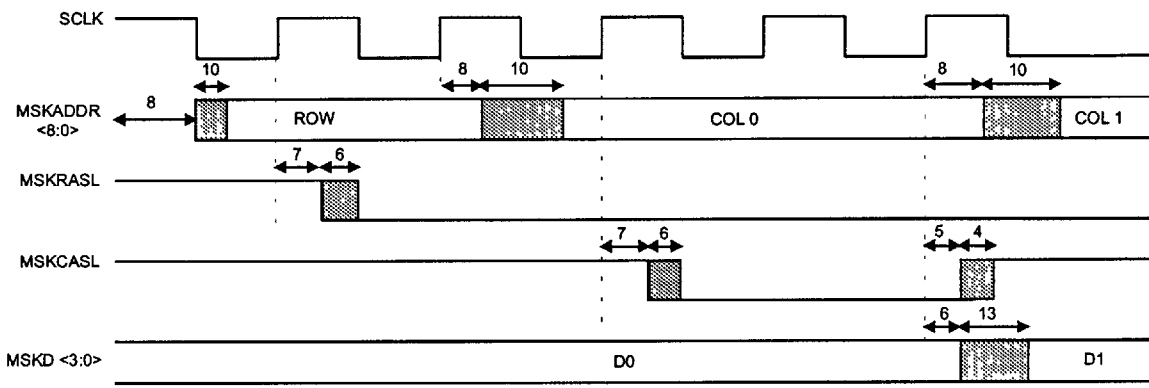


Figure 4.8.3 Mask DRAM Signal Timing

5.0 VIPeR Registers

5.1 Overview

All VIPeR registers are accessed with 16 bit memory operations. They occupy the W32x's (8k) external device address space.

VIPeR REGISTER ACCESS. All bits not listed are RESERVED (e.g. bits 15:12 in register 0x26, and all bits in register 0x06). Upon readback, all reserved bits will return 0. The only exception to this is register 0x00. The upper byte, 15:8, returns the VIPeR signature value upon readback.

Following are the VIPeR registers. They are located at address equal to the W32x's base external device address (xBE000), plus the register offset addresses listed below.

Register Offset Address	Accessible Bits	Register Name
0x00	15:0 readable, 7:0 writable	Control Register
0x02	15:0 R/W	Configuration Register A
0x04	15:0 R/W	Configuration Register B
0x06	reserved address	Configuration Register C
0x08	7:0 R/W	Status Register A
0x0A	7:0 R/W	Status Register B
0x0C	7:0 R/W	Status Register C
0x0E	reserved address	Status Register D
0x10	11:0 R/W	Horizontal Sync Width
0x12	11:0 R/W	Horizontal Sync Window Closed
0x14	11:0 R/W	Horizontal Sync Window Open
0x16	11:0 R/W	Vertical Sync Detector Width
0x18	11:0 R/W	Vertical Sync Width
0x1A	11:0 R/W	Vertical Sync Window Closed
0x1C	11:0 R/W	Vertical Sync Window Open
0x1E	11:0 R/W	Field Detector Width
0x20	9:0 R/W	Pixels per Line
0x22	9:0 R/W	Lines per Field
0x24	6:0 R/W	Framing Control
0x26	11:0 R/W	Video Bytes per Line
0x28	11:0 R/W	Maximum Bytes per Line
0x2A	11:0 R/W	X-delay
0x2C	11:0 R/W	Y-delay
0x2E	7:0 R/W	Video Control Port
0x30	15:0 R/W	X-coefficient
0x32	10:8,5:0 R/W	X-K factor
0x34	15:0 R/W	Y-coefficient
0x36	10:8,5:0 R/W	Y-K factor
0x38	11:0 R/W	Overlay Mask Control
0x3A-0x3FF	reserved addresses	
0x400-0x7FF	15:0, write only: VIPeR mask and pixel data **	

** with CPUMSKWRL, bit 11 in address 0x38 low, writes to addresses 0x400-0x7FF will be directed, by the VIPeR, to the overlay mask RAM.

** with CPUMSKWRL, bit 11 in address 0x38 high, writes to addresses 0x400-0x7FF will be interpreted, by the VIPeR, as sequential pixel data.

5.2 Register Descriptions

NOTE: following register descriptions apply to Rev.B VIPeR directly. Bit names without a * prefix apply to VIPeR Rev.B and Rev.A. Bit names marked with an *, are Rev.B only. These bits are reserved, reset to 0 in VIPeR Rev.A silicon, and the functionality the bits enable is not available in Rev.A silicon.

5.2.1 Control Register

Address 0x00

Bit	Description	Access
15:8	Reserved.	RO
7	CPUSYNC.	RW
6	Gate horizontal sync. (GSYNCSEL)	RW
5	Low active, frame sync. (CPUFSL)	RW
4	Low active, line sync. (CPULSL)	RW
3	Data readback timing BDB<15:0>. (*PREREAD)	RW
2	Tri-stateable output driver enable. (*RGBOUTen)	RW
1	Data readback timing BDB<7:0>. (*READHI)	RW
0	Image data port enable. (IXAEN)	RW

Bit **Description**

15:8 VIPeR signature. When read, returns value 0xCB, for Rev.B VIPeR. Returns value 0xBB for Rev.A VIPeR.

7 VIPeR sync information is taken from the programmed register bits, CPUFSL, CPULSL, and CPUOF, instead of the either the external sync inputs, CSYNC, LSIN, FSIN, OFIN, or the outputs of the sync separator. In addition, pixel input data from the host (BDB<15:0>) bus is used if CPUSYNC=1. If CPUSYNC is reset to 0, VIPeR accepts pixel data from the ADC (Red<7:0>,Green<7:0>,Blue<7:0>) inputs, as well as sync information from either the CSYNC or LSIN*/FSIN* inputs.

6 When set to 1, the internal horizontal sync is enabled only after the Y-delay count has been reached; and vertical active display lines are enabled.

When reset to 0, the internal horizontal sync follows the horizontal sync as determined directly by either the CPULSL, LSIN, or sync separator (whichever is enabled).

5 When reset to 0, resets the X processing, Y processing, and image port FIFO control counters. Signal is ignored if CPUSYNCenable is reset to 0.

4 When reset to 0, resets the X processing, and pulses the Y processing control counters. Signal is ignored if CPUSYNCenable is reset to 0.

3 Effects timing of readback data on BDB<15:0>. When set to 1, register readback data from VIPeR is driven onto BDB<15:0> one IXMAD* cycle after the IXMAD* cycle when a valid address is presented. Timing is also 5 nanoseconds after this later IXMAD* goes low.

When reset to 0, register readback data from VIPeR is driven onto BDB<15:0> as soon as a valid address is presented, and 5 nanoseconds after IXMAD* goes low. See VIPeR BDB bus timing diagram.

5.2.1 Control Register (cont'd)

Bit	Description
2	Enables tristatable output drivers on PRED<7:0>, PGREEN<7:0>, PBLUE<7:0>, and PSTBL. When set to 1, these processed video outputs are enabled: when reset to 0, these signals are left in tristate mode.
1	Effects readback data returned onto BDB<7:0>. When set to 1, the high byte of internal VIPeR register data <15:8> is driven onto BOTH BDB<15:8> and BDB<7:0>. This mode is useful when in PCI mode, and BDB<7:0> are only host interface bits connected to the W32x, DD<7:0> bus. When reset to 0, VIPeR register data <15:0> is driven directly to BDB<15:0> upon register reads. (Timing effected by PREREAD signal described above).
0	When set to 1, enables the tristate output buffers of the image port: IXDATA<7:0>, IMCMDL, IXMSK, IXFS, IXLS, IXOF. **NOTE:** Bits 3:1 are reserved, set to 0 in VIPeR Rev.A silicon. The preread function when set to 1, the ability to disable the processed video outputs, and the readhi=1 capability, are not available in Rev.A.

5.2.2 Configuration Register A

Address 0x02

Bit	Description	Access
15	Image mask bit polarity. (IXMSKPOL)	RW
14	Overlay Mask DRAM. (IXMSKEN)	RW
13	CPU odd field indicator. (CPUOF)	RW
12	Image data port IXOF polarity. (IXOFPOL)	RW
11	16-bit color data output. (OUT64K)	RW
10	24-bit color data output. (OUT24)	RW
9	16-bit input color mode interpretation. (CPU64KIN)	RW
8	24-bit input color mode interpretation. (CPU24IN)	RW
7	IXCMDL output signal polarity. (CMDPOL)	RW
6	Overlay mask control, > 512 output lines per field. (*G512LINES)	RW
5:3	IXCMDL output signal time and phasing. (CMDCK<2:0>)	RW
2:0	IXCMDL time and phasing. (CMDCKDIV2)	RW

Bit	Description
15	When set to 1, the IXMSK output is inverted version of the data programmed in the overlay mask DRAM. When reset to 0, the IXMSK output reflects the data programmed in the overlay mask DRAM.
14	When set to 1, the VIPeR accesses the overlay mask DRAM, and outputs mask data on IXMSK corresponding to data programmed in the overlay DRAM, as well as the programmed IXMSKPOL bit. When reset to 0, VIPeR does no accesses to the overlay mask DRAM, neither writes, nor reads nor refreshes. A constant value (equal to the value programmed in IXMSKPOL) is held on the IXMSK output pin.

5.2.2 Configuration Register A (cont'd)

Bit **Description**

13 When CPUSYNCenable is set to 1, CPUOF is used by Y processing control, and is output on IXOF signal, determining current field (1 for odd, 0 for even field). Signal is ignored if CPUSYNCenable is reset to 0.

12 When set to 1, the internal field signal is inverted. This internal field signal effects the Y processing control, and the IXOF output.

When reset to 0, the VIPeR internal field signal (derived from either CPUOF or OFIN input) is unaffected.

11 When set to 1, VIPeR outputs 64K (16 bit, 565) color mode data on both image port and the processed data outputs. If the input data is 24 bit per pixel, lower order bits are just dropped internally before outputting the 565. (Scaling is done at 24bpp precision internally, before the output lower order bits are dropped).

When reset to 0, and OUT24 is also 0, VIPeR outputs 32K (15 bit, 555) color mode data on both image port and the processed data outputs. Similarly, lower order bits are dropped at the final output if the input data is 24 or 16bpp.

10 When set to 1, VIPeR outputs 16M (24 bit, 888) color mode data on both the image port and the processed data outputs. If the input data is 15 or 16 bits per pixel, the output data is padded with zeros in the least significant bits. (Scaling is done at 24bpp precision internally, however, the 15 or 16bpp source is padded with zeros at the input to the scaling circuitry to create the 24bpp).

9 This bit must be set in the YUV422 mode. When set to 1, CPUSYNC set to 1, and CPU24IN reset to 0, VIPeR interprets input data from the CPU in 64K (16 bit per pixel, 565) color mode. The input color mode is determined by CPU64KIN, CPU24IN, and CPUSYNC. See table 5.2.1 below for input mode.

8 When set to 1, and CPUSYNC set to 1, VIPeR interprets input data from the CPU in 16M (24 bit per pixel, 888) color mode. When both CPU24IN and CPU64KIN reset to 0, (with CPUSYNC set to 1, VIPeR interprets input data from the CPU in 32K (15 bit per pixel, 555) color mode. See table 5.2.1 below for input mode.

Table 5.2.1: Input Color Mode Determination

CPUSYNC	CPU24IN	CPU64KIN	Input Mode
1	0	0	15 bit/pixel (555 mode)
1	0	1	16 bit/pixel (565 mode) / YUV
1	1	X	24 bit/pixel (888 mode)
0	X	X	CPU data ignored: input from ADC (RED<7:0>, GREEN<7:0>, BLUE<7:0> inputs)

7 When reset to 0, IXCMDL follows the timing phase determined directly by CMDCK<2:0> and CMDCKDIV2<2:0>, described below. When set to 1, IXCMDL is the inverted version.

6 Used to enable overlay masking of greater than 512 lines per field. When set to 1, optional overlay mask DRAM read address high column address bit is forced to 1 when the internal line count is greater than 511. See the data organization and split mask modes of the External Mask DRAM Control, Section 2.6, of this document for fuller explanation.

5.2.2 Configuration Register A (cont'd)

Bit	Description
5:3	These bits are used in conjunction with CMDCKDIV2<2:0> to determine time and phasing of the IXCMDL output signal, with respect to the 50MHz input system clock, SCLK.
2:0	These bits are used in conjunction with CMDCK<2:0> to determine time and phasing of IXCMDL. Table 5.2.1 gives a brief summary of the IXCMDL timing as a function of these bits. A fuller explanation is giving in Section 4.2, Image Port Data Signal Timing. **NOTE:** Bit 6 is reserved, set to 0 in VIPeR Rev.A silicon. The VIPeR Rev.A overlay mask feature supports video windows of up to 512 lines/field only.

5.2.3 Configuration Register B

Address 0x04

Bit	Description	Access
15	Y expand mode. (YEXP)	RW
14	Y crush mode. (YCRUSH)	RW
13	Y scaler coefficient offset enable. (*YCOFIOFFEN)	RW
12	Field polarity inverter. (*YCOFIELDPOL)	RW
11	Audio data output enable. (AUDIOEN)	RW
10	Y input data in Least Significant Byte lane. (*YINLSB)	RW
9	YUV422 to RGB conversion. (YUV422A)	RW
8	YUV422 to RGB conversion. (YUV422B)	RW
7	YUV to RGB conversion enable. (YUVEN)	RW
6	Internal Sync Separator Bypass. (SSBYPASS)	RW
5	ADC enable output pin. (ADCenable)	RW
4	Clock Sync Polarity. (CLKSYNCPOL)	RW
3	Invert CSYNC input. (*INVCSYNC)	RW
2	ADC clock divide by 2. (ADCCLKDIV2)	RW
1	ADC latch polarity. (ADCLATPOL)	RW
0	ADC clock polarity. (ADCCLKPOL)	RW

Bit	Description
15	Should be set to 1 if the desired Y output size (number of output lines per field) is greater than the Y input size (number of input lines per field). Set to zero if Y output size is less than or equal to the Y input size.
14	Should be set to 1 if the desired Y output size (number of output lines per field) is less than the Y input size (number of input lines per field). Set to zero if Y output size is greater than or equal to the Y input size. **NOTE:** Allowable values for YEXP, YCRUSH are 01, and 10. Setting YEXP, YCRUSH to either 00, or 11 will result in unreliable operation. In addition, setting YEXP, YCRUSH incorrectly, for example, 10 when Y output size is less than the Y input size (as determined by data delivery rates and programmed lines per field, register 0x22) will result in unreliable operation.

13 When set to 1, the Y processing coefficient generator begins every other field with a constant offset, rather than the default 0.

When reset to 0, the Y processing coefficient generator begins each field with the default 0 value. Enabling this bit allows the Y scaler unit to compensate for the top line shifting inherent in odd/even interlaced input fields.

12 When set to 1, the Y scaler coefficient offset will be used, if enabled by setting YCOFIOFFEN, for even fields. The offset will be 0 for odd fields.

When reset to 0, and YCOFIOFFEN set to 1, the Y scaler coefficient offset will be used for odd fields, and the offset will be 0 for even.

11 When set to 1, allows the VIPeR to add whatever audio data has accumulated during a scan line time, to the end of the video scan line data, at the Image Port data outputs. The amount of audio data output on the IMA per scan line is limited by how much has accumulated, and the number of audio bytes programmed. This programming is done indirectly by setting Maximum Bytes per Line, register 0x028, to be equal to the Video Bytes per Line, register 0x26, plus the desired number of audio bytes per line. A fuller explanation of the audio data feature of the VIPeR is given in Audio Data, Section 2.8.

10 Used when in operating in YUV422 mode, allows for byte lane switching of input data. When YINLSB is set to 1, VIPeR assumes Y data is in the low byte, and U/V is in the high byte.

When reset to zero, VIPeR assumes high byte of 16 bit input data is Y, and low byte of 16-bit input data is U/V information.

9 When set to 0, VIPeR performs YUV444 to RGB conversion.

When set to 1, VIPeR performs YUV422 to RGB conversion. Whether it performs standard YUV or YUV ITT conversion is dependent on the status of bit 8 of same register Configuration Register B).

8 When set to 0, VIPeR performs YUV422 to RGB conversion.

When set to 1, VIPeR performs YUV422 ITT to RGB conversion.

YUV	Y0<7:0>,U0<7:0>
	Y1<7:0>,V0<7:0>
	Y2<7:0>,U2<7:0>
	Y3<7:0>,V2<7:0>
	Y4<7:0>,U4<7:0>
	Y5<7:0>,V4<7:0>
YUV ITT	Y0<7:0>,U0<7:4>,V0<7:4>
	Y1<7:0>,U0<3:0>,V0<3:0>
	Y2<7:0>,U2<7:4>,V2<7:4>
	Y3<7:0>,U2<3:0>,V2<3:0>
	Y4<7:0>,U4<7:4>,V4<7:4>
	Y5<7:0>,U4<3:0>,V4<3:0>

5.2.3 Configuration Register B (cont'd)

YUV to RGB Selection Guide

Bit 9	Bit 8	Bit 7	
0	x	1	YUV444
1	0	1	YUV422
1	1	1	YUV422 ITT
x	x	0	RGB

NOTE: Allowable values of YUV422A, YUV422B are 01, 10, and 11. A programmed value of 00 is reserved for future VIPeR iterations and could result in unreliable operation. If a value of 01 is programmed and YUVEN is set to 1, VIPeR will perform YUV to RGB conversion assuming un-subsampled (YUV444 mode) input formats.

Bit **Description**

7 When set to 1, VIPeR performs YUV to RGB conversion on the input data, assuming input format specified by YUV422A and YUV422B above. When reset to 0, VIPeR does no data color space conversion.

6 When set to 1, and when in ADC input mode (CPUSYNC reset to 0), the VIPeR uses the FSINL, LSINL, and OFIN directly, unaffected by the sync separator.

When reset to 0, and in ADC input mode, VIPeR derives its line/frame/field information from the CSYNC input pin, modulated by the sync separator filter logic.

NOTE: the sync separator registers, addresses 0x10 through 0x1E must be programmed for the expected sync input characteristics, even if the sync separator is bypassed by setting SSBYPASS to 1.

5 This bit has no effect on VIPeR operation except that its value is seen at the ADCEN output pin.

4 When reset to 0, VIPeR uses the falling edge of CSYNC (or LSINL if CPUSYNC mode is chosen) to initiate horizontal synchronization activity. When set to 1, VIPeR uses the rising edge.

3 When set to 1, inverts polarity of the input CSYNC signal within the VIPeR. When reset to 0, leaves CSYNC input unaffected.

2 When reset to 0, the ADCCLK output is the same frequency as the input LLCLKIN. When set to 1, the ADCCLK is half the frequency of the input LLCLKIN.

NOTE: The ADCCLK output is used internally by VIPeR to latch ADC input data.

1 When set to 1, VIPeR latches ADC input data when ADCCLK falls.

When reset to 0, the VIPeR latches ADC input data when the output ADCCLK signal rises.

NOTE: ADCCLK output is derived from LLCLKIN, modulated by ADCCLKDIV2 and ADCCLKPOL.

0 When set to 1, the ADCCLK is the inverted version of the result of ADCCLKDIV2 and LLCLKIN. That is, if ADCCLKDIV2 is 0, and ADCLATPOL is 1, then ADCCLK is the inverted version of LLCLKIN. If both bits are 1, then ADCCLK will be the inversion of LLCLKIN divided by 2.

When reset to 0, the ADCCLK remains in phase with the LLCLKIN.

NOTE: Bits 13,12,10,3 are reserved, set to 0 in VIPeR Rev.A silicon. The Y coefficient offset and field polarity are as if they were both programmed 0, (each field starts with Y coefficient generator offset=0). YINLSB behaves, in Rev.A, as if it were set to 0, (Y always in MSB, U/V always in LSB). The invert csync option is internally available in Rev.A. CSYNC input is used unaffected by VIPeR Rev.A.

5.2.4 Configuration Register C

Address 0x06

All bits reserved: set to 0.

5.2.5 Status Register A

Address 0x08

Bit	Description	Access
15:8	Reserved (always return 0x00 when read).	RO
7	Input state. (IXRDY status)	RO
6	*LBUS input pin state. (LBUSL)	RO
5	PCIBUS* input pin state. (PCIBUSL)	RO
4	ISABUS input pin state. (ISABUS)	RO
3	OFIN input pin state. (OFIN)	RO
2	LSIN* input pin state.(LSINL)	RO
1	FSIN* input pin state. (FSINL)	RO
0	CSYNC input pin state. (CSYNC)	RO

Bit **Description**

7 Realtime state of the input (from W32x) IXRDY signal: part of the Image Data Port.

6 Reflects state of LBUS* input pin.

NOTE: LBUS* input pin has different functionality between Rev.A and Rev.B VIPeR. See pinout section for description.

5 Reflects state of PCIBUS* input pin.

NOTE: PCIBUS* input pin has different functionality between Rev.A and Rev.B VIPeR. See pinout section for description.

4 Reflects state of ISABUS input pin.

NOTE: ISABUS input pin has different functionality between Rev.A and Rev.B VIPeR. See pinout section for description.

3 Reflects state of OFIN input pin.

2 Reflects state of LSIN* input pin.

1 Reflects state of FSIN* input pin.

0 Reflects state of CSYNC input pin.

5.2.6 Status Register B

Address 0x0A

Bit	Description	Access
15:8	Reserved (always return 0x00 when read).	RO
7	W32 interface FIFO empty flag. (EMPTY)	RO
6	Vertical scaling horizontal sync output. (YHSYNCL)	RO
5	Reserved (always returns 0 when read).	RO
4	Scaler (X and Y directions) completion bit. (YXDONL)	RO
3	Sync Separator Interlace flag. (SSIF)	RO
2	Sync Separator Odd Field. (SSOF)	RO
1	Sync Separator horizontal sync. (SSHSL)	RO
0	Sync Separator vertical sync. (SSVSL)	RO

Bit	Description
7	Returns value of 1 when Image Data Port FIFO is empty. This can be used by software to regulate the delivering of video data lines.
6	Value of internal horizontal sync signal, output from the scaler unit (and input to the Image Data Port FIFO). Signal is low when the VIPeR scaler (X and Y directions) is seeing an active internal horizontal sync pulse.
4	The Signal is low when the VIPeR scaler, (X and Y directions) has completed processing of the current line. It is set high again by either the next horizontal or vertical sync pulse.
3	Internal sync separator has detected framing information in the composite sync input.
2	Reflects the realtime status of the sync separator field signal.
1	Reflects the realtime status of the sync separator horizontal sync output.
0	Reflects the realtime status of the sync separator vertical sync output.

5.2.7 Status Register C

Address 0x0C

Bit	Description	Access
15:8	Reserved (always return 0x00 when read).	RO
7	Image Data Port Line Sync. (IXLS)	RO
6	Image Data Port Frame Sync. (IXOF)	RO
5:0	Frame Count. (FC<5:0>)	RO

Bit	Description
7	Reflects the realtime status of the image data port line sync.
6	Reflects the realtime status of the image data port odd field signal. The polarity depends on the IXPOL in Configuration A (normally set active high).
5:0	These bits reflect the number of the last complete frame written to the Image Data Port. It is controlled by the Framing Control Register, 0x24. See 5.2.18 for a description of this register.

5.2.8 Status Register D

Address 0x0E

All bits reserved: set to 0.

5.2.9 Sync Separator Horizontal Sync Width Register

Address 0x10

Bit	Description	Access
15:12	Reserved (always return 0x00 when read).	RO
11:0	Sync Separator sync pulse width parameter. (SSHSW)	RW

Bit	Description
11:0	Determines duration of internal horizontal sync pulse as started by either detecting an horizontal sync on CSYNC input, or by generating an internal horizontal sync based on programmable time out waiting for one.

Window is started by either arrival of a valid hsync pulse on CSYNC input (during the horizontal window open time) or by the timing out of the horizontal window open width register (SSHWO). If the SSHWO times out before a valid hsync arrives on CSYNC input, the VIPeR will generate an hsync, and restart the SSHSW window timer.

Horizontal sync width is programmed in system clocks minus 1. For example, to program 4.7 microseconds at 50MHz SCLK, SSHSW = [(4.7us/20ns)-1] = 234 decimal= 0xEA.

5.2.10 Sync Separator Horizontal Window Closed Width Register

Address 0x12

Bit	Description	Access
15:12	Reserved (always return 0x00 when read).	RO
11:0	Sync Separator horizontal window closed register. (SSHWC)	RW

Bit	Description
11:0	A programmable counter defining the time during which the sync separator will ignore any (presumably noise) horizontal sync pulses on CSYNC input.

Horizontal window closed (SSHWC) is started by the completion of the SSHSW timer. Completion of this SSHWC timer starts the horizontal window open width register (SSHSO). Programmed in system clocks minus 1. For example, to program 57.6 microseconds at 50MHz SCLK, SSHWC = [(57.6us/20ns)-1] = 2879 decimal=0xB3F.

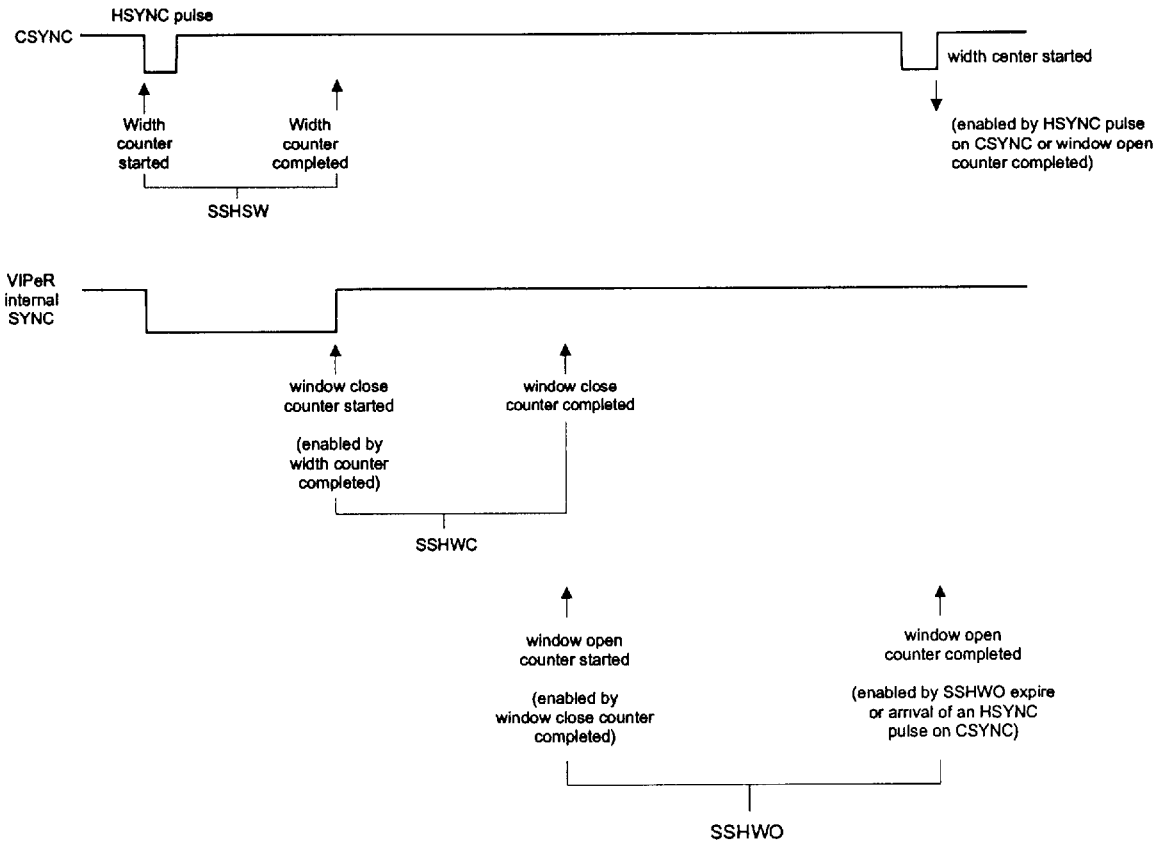
5.2.11 Sync Separator Horizontal Window Open Width Register

Address 0x14

Bit	Description	Access
15:12	Reserved (always return 0x00 when read).	RO
11:0	Sync Separator horizontal window open register. (SSHWO)	RW

Bit	Description
11:0	A programmable counter defining the time during which the sync separator will accept a horizontal sync pulse on CSYNC as a legitimate input horizontal sync. Not finding a valid horizontal sync pulse on the CSYNC input during this window open period, the VIPeR sync separator will generate one of its own, reset itself and start the horizontal sync width timer (SSHSW). If a horizontal sync pulse is found on CSYNC input during the window open time, the VIPeR will use it, close the SSHWO window immediately, and start the SSHSW timer. A fuller explanation of these three registers is given in Section 2.3, Sync Separator functionality, earlier in this document. Also, see the time line below.

Horizontal Sync Separator Time Line



5.2.12 Sync Separator Vertical Sync Detector Width Register

Address 0x16

Bit	Description	Access
15:12	Reserved (always return 0x00 when read).	RO
11:0	Sync Separator vertical sync detector width parameter. (SSVSDW)	RW

Bit	Description
11:0	Defines, in system clocks minus 1, the amount of time the CSYNC input must be low to be interpreted as a valid vertical sync, by the VIPeR.

5.2.13 Sync Separator Vertical Sync Width Register

Address 0x18

Bit	Description	Access
15:12	Reserved (always return 0x00 when read).	RO
11:0	Sync Separator vertical pulse width parameter. (SSVSW)	RW

Bit	Description
11:0	Determines duration of internal vertical sync pulse as started by either detecting a valid vertical sync on CSYNC input, or by generating an internal vertical sync based on programmable time out waiting for one. Window is started by either arrival of a valid vertical sync on CSYNC input (during the vertical window open time) or by the timing out of the vertical window open width register (SSVWO). If the SSVWO times out before a valid vsync arrives on CSYNC input, the VIPeR will generate a vsync, and restart the SSVSW window timer. Vertical sync width is programmed in horizontal sync pulses minus 1.

5.2.14 Sync Separator Vertical Window Closed Width Register

Address 0x1A

Bit	Description	Access
15:12	Reserved (always return 0x00 when read).	RO
11:0	Sync Separator vertical window closed register. (SSVWC)	RW

Bit	Description
11:0	A programmable counter defining the time during which the sync separator will ignore any (presumably erroneous) vertical syncs on CSYNC input. Vertical window closed (SSVWC) is started by the completion of the SSVSW timer. Completion of this SSVWC timer starts the vertical window open width register (SSVSO). Programmed in horizontal sync pulses minus 1.

5.2.15 Sync Separator Vertical Window Open Width Register

Address 0x1C

Bit	Description	Access
15:12	Reserved (always return 0x00 when read).	RO
11:0	Sync Separator vertical window open register. (SSVWO)	RW

Bit	Description
11:0	A programmable counter defining the time during which the sync separator will accept a vertical sync on CSYNC as a legitimate input vertical sync. Not finding a valid vertical sync on the CSYNC input during this window open period, the VIPeR sync separator will generate one of its own, reset itself and start the vertical sync width timer (SSVSW). If a vertical sync is found on CSYNC input during the window open time, the VIPeR will use it, close the SSVWO window immediately, and start the SSVSW timer.

5.2.15 Sync Separator Field Detection Window Width Register

Address 0x1E

Bit	Description	Access
15:12	Reserved (always return 0x00 when read).	RO
11:0	Sync Separator Field detection window width register. (SSFDWW)	RW

Bit	Description
11:0	These bits comprise a programmable counter defining the number of horizontal sync pulses needed before the FIELD output toggles (in interlaced mode). It is programmed in horizontal sync pulses minus 1.

5.2.16 Pixels Per Line Register

Address 0x20

Bit	Description	Access
15:10	Reserved (always return 0x00 when read).	RO
9:0	Pixels per line. (PPL)	RW

Bit	Description
9:0	Defines the number of pixels the VIPeR will output per line. Program desired value minus 1.

5.2.17 Lines Per Field Register

Address 0x22

Bit	Description	Access
15:10	Reserved (always return 0x00 when read).	RO
9:0	Lines per field. (LPF)	RW

Bit	Description
9:0	Defines the number of video lines the VIPeR will output per field. Program desired value minus 1.

5.2.18 Framing Control Register

Address 0x24

Bit	Description	Access
15:7	Reserved (always return 0x00 when read).	RO
6	Frame count reset.	RW
5:0	Frame count.	RW

Bit	Description
6	When set to 1, enables the frame count functionality described immediately below. When reset to zero, has the effect of forcing the frame count value to 00.
5:0	Frame counter value for strip mode operation. Defines the number of VIPeR's frame sync signals to suppress before finally letting one emerge on the IXFS signal. This reduces the number of frame syncs seen by the graphics controller, but does NOT suppress the video data associated with each field. The result is a "strip" of FC+1 fields of video written (and potentially displayed) one directly on top of the previous in the W32x graphics frame buffer. The number FC, is the programmed frame count value. For example, programming 0x40 in the framing control register has the effect of passing every field of video with its corresponding frame sync. This is the normal, one video window mode. Programming 0x41 has the effect of suppressing every other frame sync signal. If they are written to visible regions of the W32x frame buffer, the effect will be to see two video windows, one directly on top of the other. If the video lines associated with the second window go out of range of the W32x visible range, programming 0x41 in the framing control register is a very useful way to de-interlace the video: capturing only odd video fields and dropping the even ones.

5.2.19 Video Bytes Per Line Register

Address 0x26

Bit	Description	Access
15:11	Reserved (always return 0x00 when read).	RO
10:0	Video Bytes per Line. (VBPL)	RW

Bit	Description
10:0	Defines the number of video bytes output from the VIPeR, on the Image Data Port, per scan line. Program desired value minus 1. Programmed value should be consistent with the value programmed in the pixels per line register, and the output bit/pixel format programmed in configuration register A. For example, if 640 pixels per line are desired, in 16 bit/pixel format, 639 should be programmed in the pixels per line register, and 1279 should be programmed in the video bytes per line register. 1279 = [(640 pixels/line)*(2 bytes/pixel)] -1

5.2.20 Maximum Bytes Per Line Register

Address 0x28

Bit	Description	Access
15:11	Reserved (always return 0x00 when read).	RO
10:0	Maximum Bytes per Line. (MBPL)	RW

Bit	Description
10:0	Defines the total number of bytes output from the VIPeR, on the Image Data Port, per scan line. If no audio data is desired, the maximum bytes per line should be equal to the video bytes per line. If audio data is to be appended to the video scan output line (see Section 2.8 on Audio Data), setting the maximum bytes line to some number N greater than the maximum video bytes per line, determines the maximum number (N) bytes of audio data that can be output on Image Data Port, after completion of the video data. Care must be taken here so that enough time is left for all the video pixels to be output completely, and received by the W32x. That is, the maximum bytes per line must not be programmed to a value larger than the horizontal sync frequency minus the X delay.

5.2.21 X delay

Address 0x2A

Bit	Description	Access
15:11	Reserved (always return 0x00 when read).	RO
10:0	Horizontal active display delay time. (Xdelay)	RW

Bit	Description
10:0	VIPeR begins accepting and processing video data (Xdelay - 1) SCLK cycles after receiving the horizontal sync pulse. This processing is also gated by the Y delay counter also having timed out, (see Y delay register immediately following).

5.2.22 Y delay

Address 0x2C

Bit	Description	Access
15:11	Reserved (always return 0x00 when read).	RO
10:0	Vertical active display delay time. (Ydelay)	RW

Bit	Description
10:0	VIPeR begins accepting and processing video lines (Ydelay - 1) horizontal sync cycles after receiving the vertical sync.

For example, if the programmer sets Xdelay equal 110, and Ydelay equal 21, then active video will begin 109 SCLK cycles after the 20th horizontal sync pulse, AFTER vertical sync.

5.2.23 Video Control Port Register

Address 0x2E

Bit	Description	Access
15:8	Video Control Port configuration. (VCPConfig<7:0>)	RW
7:0	Video Control Port register locations. (VCPreg<7:0>)	RW

Bit	Description
15:8	These 8 bits control the direction (input or output) of the corresponding 8 video control port pins. When set to 1, the corresponding pin is set to output: 0 it's an input. These 8 configuration bits default to 0 on reset.
7:0	When written, data on these 8 bits are stored in an internal register. If the corresponding VCPConfig bit is set to 1, data stored in that internal register is driven at the corresponding VC pin. When the VCPConfig bit is set to 0, the pin is an input. Reading this VCP port register, in that case, will reflect the logic state on that corresponding pin.

NOTE: reading bits which are configured as outputs will yield unreliable results. Attempting to write a bit which is configured as input (by VCPConfig) will not drive the attempted write data externally.

A fuller explanation of the Video Control Port timing and functionality is given in the Video Control Port section, 2.9.

5.2.24 X Scaling Coefficient Register

Address 0x30

Bit	Description	Access
15:0	Undocumented bits. (Xscale)	RW

Bit	Description
15:0	This register contributes to the control of the horizontal scaling mechanism. It is calculated by the device driver based on desired input and output horizontal resolutions.

5.2.25 XK Scaling Coefficient Register

Address 0x32

Bit	Description	Access
15:0	Undocumented bits. (XKscale)	RW

Bit	Description
15:0	This register contributes to the control of the horizontal scaling mechanism. It is calculated by the device driver based on desired input and output horizontal resolutions.

5.2.26 Y Scaling Coefficient Register

Address 0x34

Bit	Description	Access
15:0	Undocumented bits. (Yscale)	RW

Bit	Description
15:0	This register contributes to the control of the vertical scaling mechanism. It is calculated by the device driver based on desired input and output vertical resolutions.

5.2.27 YK Scaling Coefficient Register

Address 0x36

Bit	Description	Access
15:0	Undocumented bits. (YKscale)	RW

Bit	Description
15:0	This register contributes to the control of the vertical scaling mechanism. It is calculated by the device driver based on desired input and output vertical resolutions.

5.2.28 Mask Control Register

Address 0x38

Bit	Description	Access
15:12	Reserved (always return 0x00 when read)	RO
11	External mask DRAM write enable. (CPUMSKWRL)	RW
10	Mask DRAM column address enable. (MSKCOL8)	RW
9	Reserved (always returns 0 when read)	RO
8:0	CPUROW address for mask DRAM writes. (CPUROW)	RW

Bit **Description**

11 When reset to 0, the VIPeR interprets any host register writes to addresses 0x400 thru 0x7FF as writes to the external mask DRAM. When set to 1, the VIPeR will do no writes to the external mask DRAM.

Host writes to 0x400 thru 0x7FF (if CPUMSKWRL is 0), initiate write cycles to the mask DRAM. When doing a write cycle, 16 bits of data are written out to a series of 4 sequential column addresses: four bits at a time. The row address is the current row address stored in the CPUROW bits listed in this register below. The column address comes from the lower 7 bits of the host presented write address (in the 0x400 thru 0x7FF range). These lower 7 bits of host address map to the upper 7 (of 9 total) column address bits presented to the mask DRAM. The lower 2 bits of column address are counted up from 00 thru 11 in the sequencing of four 4 bit writes which make up the 16 bit write to the external mask DRAM.

10 Forces the most significant bit of the mask DRAM column address to 1 upon data readout during video active display times, when MSKCOL8 is set to 1. When MSKCOL8 is reset to 0, the entire mask DRAM column (read) address follows the internal pixel counter.

Setting MSKCOL8 to 1, has the effect of "splitting" the mask DRAM into two halves; each of 512 lines by 1024 pixels size.

8:0 **NOTE:** similar to the column addresses during active video display time: row addresses on data readout reflect the internal line counter in the VIPeR.

Host accesses to the mask DRAM are write only. VIPeR reads the mask DRAM (and pairs up the mask data with the corresponding video pixels) during active video display times; however, there is no direct mechanism for the host to read the mask DRAM through the VIPeR. Readback is attainable indirectly: in conjunction with the Image Data Port, and the W32x frame buffer memory.

A fuller explanation of the working and timing of the mask DRAM control is given in Section 2.1, External Mask DRAM control.

5.2.29 Reserved Registers

Addresses 0x3A-0x3FF

Bit	Description	Access
15:0	Reserved (always return 0x00 when read).	RO

5.2.30 Host Access Registers

Addresses 0x400-0x7FF

Bit	Description	Access
15:0	Pixel or mask DRAM data to be written. (DATA<15:0>)	WO

Bit **Description**

15:0 If CPUMSKWRL is 1, data written to 0x400-0x7FF is interpreted as input pixel data, if VIPeR is also programmed for CPU playback (by setting CPUSYNC=1). Format of the data is determined further, by the which input bit/pixel format is chosen. If 15 or 16 bit/pixel is chosen, 16 bits of data are accepted by the VIPeR for any access in the 0x400 thru 0x7FF address range. If 24 bit/pixel is chosen, the data is assumed to correspond to the lower 16 bits, if the host address presented is even. The 16 bit input data is assumed to be (don't care in the upper byte) and the upper byte of the 24 bit input pixel (in the lower byte of the host data transfer) if the host address is odd.

If CPUMSKWRL is 0, data written to 0x400-0x7FF is interpreted as mask DRAM data. The lower 7 bits of the host address are used as bits <8:2> of the column address during the actual write to the mask DRAM. The remaining column address bits, and the row address are gotten from register data previously written to the mask control register, 0x38, described above.

6.0 Summary of Rev.A versus Rev.B VIPeR Differences

VIPeR Rev.A and Rev.B are pin and register compatible. However, there are a modest number of changes between the Rev.A and Rev.B VIPeR chips. These fall into 3 classes:

- pin definition changes
- new features,
- Rev.A bug fixes

Each is described in the following sections.

6.1 Pin Definition Changes

Viper pinout identical (A-step to B-step), EXCEPT:

1. Function of pins ISABUS, and LBUS* have changed.

Rev.A definition:

ISABUS = ISA host CPU bus connection

ISABUS=1: 16 bit buffered data bus host bus connection
MEMW* input is low active
ISABUS=0: 8 bit DD host bus connection
MEMW* input is high active

LBUS* = VESA Local bus mode

LBUS*=1: during host access address, DD<1> used directly as the VIPeR address bit<1>
LBUS*=0: during host access address, DD<1> and DD<0> are ANDed together internally, and used as the VIPeR address bit<1>

Rev.B definition:

LBUS* = invert reset

LBUS*=1: input RESET* is inverted internally
LBUS*=0: input RESET* is not inverted, but used directly

ISABUS = delay latching of data on BDB during VIPeR register access

ISABUS=0: data on BDB<15:0> is latched 6 (+/-2) ns after falling edge IXMAD
ISABUS=1: data on BDB<15:0> is latched 0 (+/-2) ns after rising edge IXMAD

2. PRED<0>, PBLUE<0> outputs have additional functionality.

Upon RESET going inactive, Rev.B latches values on PRED<0> and PBLUE<0> and uses these two values to control the choice of BDB address and data register access timing. Programmable with jumperable weak pulldowns on PRED<0> and PBLUE<0>; latching of values on these pins leads to generation of two internal signals (in combination with PCIBUS* input), EARLYADDRESS and EARLYDATA. Values of these are determined by:

EARLYADDRESS = PCIBUS* (exclusive-OR'd) with latched PBLUE<0>
EARLYDATA = PCIBUS* (exclusive-OR'd) with latched PRED<0>

if EARLYADDRESS = 0: BDB address timing same as VIPeR-A wrt IXMAD
if EARLYADDRESS = 1: internal BDB address path sped up ~6ns (expected address later on BDB wrt IXMAD)

if EARLYDATA = 0: BDB data latched with same timing as VIPeR-A
if EARLYDATA = 1: internal BDB data path up ~6ns (expected data later on BDB wrt IXMAD)

NOTE: See timing diagram in Section 4.1 for details of host bus timings, and how they are quantitatively effected by PRED<0> and PBLUE<0>

6.2 New features: Rev.B versus Rev.A VIPeR

1. Maximum output horizontal resolution increase from 640 to 720 pixels/line

2. Independent internal BDB bus address and data timing skew control which adds flexibility for working with different BDB, IXMAD bus timings. See Sections 6.1 and 4.1 for detailed description of timings.

3. Tristatable outputs for 24-bit digital RGB outputs (and PSTB). New register bit, RGBOUTEN, bit 2 in Control Register controls this feature. See Section 5.0 for a fuller description of this new feature.

4. Assorted other new video controls enabled by setting following register control signals. See Section 5.0 for full description.

- GSYNCSEL:	gates internal horizontal sync signals with vertical active display time
- PREREAD:	change the timing of VIPeR register reads
- RGBOUTEN:	Enables the processed data (RGB) output pins
- READHI:	Allow reading of high byte of 16 bit register data, onto the 8 bit DD host bus
- G512LINES:	Allow for full masking capability for video display formats of over 512 lines/field (like PAL)
- YCOFIOFFEN:	Preset vertical processing control for sharper scaling of interlaced video sources.
- YCOFIELDPOL:	Preset vertical processing control for sharper scaling of interlaced video sources.
- YINLSB:	Allow for byte lane switching of luminance/color information in YUV modes
- INVCSYNC:	Internal polarity selection of input CSYNC pin

6.3 VIPeR Rev.A operational discrepancies corrected in VIPeR Rev.B

1. IXMAD# should be held high (externally) during reset time.
   - "OR" IXMAD# with reset active

2. IXMAD# must be delayed ~5ns in Rev.A DD bus mode. Timing of host data writes is very touchy in Rev.A VIPeR.

3. Decode of addressing in Rev.A DD bus mode is done incorrectly.

4. 16 bit per pixel Image Port output does not work in Rev.A. 15 and 24-bit/pixel modes do work.

5. Unable to read back high data byte in Rev.A DD bus mode.

6. Mask overlay bug in first full line of every frame. (Rev.A not accessing optional mask DRAM correctly until second video line)

7. Audio FIFO output disabled (unable to read out audio data from Rev.A)

8. Register readback timing problem. Rev.A returns data too slowly to be latched into W32p GAL required for ISA, PAL with different equations for Local, PCI bus modes.

Index

L

M

N

O

P

R

S

V

Y