C-Cube CL550 and Xilinx XC3020A ISA-based Motion-JPEG Codec

November 1994

Application Note By LOUIS W. SHAY

Introduction

This design is the result of a recent collaborative effort between C-Cube Microsystems, Auravision Corporation (Fremont, CA), Xilinx Corporation (San Jose, CA), and Ring Zero Systems (San Mateo, CA). The design is a Motion-JPEG video codec for ISA bus PC platforms based on the CL550 JPEG, which features a direct hardware interface to the Auravision VxP500 Video Processor. The CL550 codec installs as a daughter card on a stock Auravision VxP500 evaluation board to provide real-time JPEG video or still-image compression functions.

The VxP500 is among the first in a new generation of video processors that have built-in support for video compression engines such as the CL550. These support functions include input image pre-scaling, cropping, and buffering, zoom functions for playback, and a direct-access compression interface. The CL550 uses all of the VxP500's support features to achieve high-quality compressed video for a modest implementation cost. Ring Zero Systems, a leading supplier of Windows device drivers for the graphics and multimedia industry, contributed the software drivers that ties together the video processing chain, providing transparent support for off-the-shelf Windows software packages like Adobe Premiere™.

This document describes the CL550 daughter card hardware design in detail. The design can be taken as-is, or it can be customized and cost-reduced with a minimum of engineering effort. In writing this document, care was taken to emphasize fundamental design concepts and techniques as opposed to specific implementation. If you are designing for a platform other than the ISA PC, you will find that many of these concepts extend to any target hardware environment.

The remainder of this document includes:

- Functional Overview/Theory of Operation
- Program Interface Specification
- Interface Logic Design Details
- Daughter Card Schematics
- FPGA Schematics

Design Highlights

- Based on C-Cube CL550-30 (MQUAD) processor clocked at 25 MHz and low cost Xilinx XC3020A FPGA.
- Compact design, minimal parts count (only 11 IC's).
- Xilinx FPGA interface logic design for low-cost, easy customization and quick time-to-market.
- Real time compression of video data or still images at resolutions to 320 x 240, 30 fields per second, and sustained compressed data rates to 500 KB/second (system dependent).

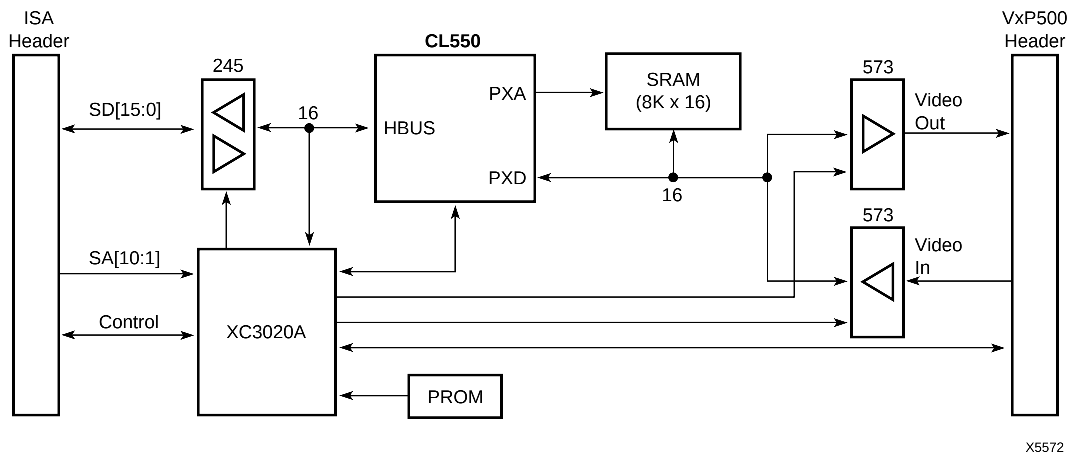

Figure 1. CL550 Motion-JPEG Daughter Card Block Diagram

- Designed to support Microsoft's Video-For-Windows architecture; works well with popular Windows-based video editing applications.
- Program-I/O based driver architecture gives higher performance than DMA while eliminating the extra cost and compatibility problems associated with typical DMA implementations.
- Uses all of the VxP500's input scaling, cropping, frame buffering, and capture features for higher quality video input on compression. This makes a noticeable difference in image quality over other implementations in it's cost range.
- Hardware zoom on playback via the VxP500 allows up to full-screen video playback from 320x240 or smaller decompressed frames.

Functional Overview / Theory of Operation

The codec design provides four key functions required by video capture and editing applications.

1) Compression from live video input (capture) for real-time storage to disk.
2) Decompression of images and/or video to display overlay (playback).
3) Decompression of image frames back to memory for processing/editing.
4) Compression of processed/edited image frames back to storage.

Each of these functions must be supported by the hardware codec to achieve a level of speed practical for the PC user. Points 1 and 2 are obvious to most designers. Points 3 and 4 are not as obvious but no less important, considering their use in editing applications.

For example, let's say we have two compressed video streams A and B. We want to mix them using a dissolve effect, and store the result to stream C. To the editing application, this means:

- Decompress frame A to memory,
- decompress frame B to memory,
- process A,B to frame C,
- compress frame C to storage.

To the codec, this means two decompresses and one compress for every frame in the stream. Hardware codecs that do not easily support these functions must resort to software-based compression, wasting an otherwise good codec and a lot of the user's time. Not only does this design support memory-to-memory compression/decompression, it does it for zero added cost - a direct benefit of this architecture.

Figure 1 shows a block diagram of the CL550 daughter card. Figure 2 is a simplified system block diagram, showing the Auravision VxP500 system and the datapaths that connect it to the CL550. For the CL550, there are three datapaths required:

- Host interface (CL550-to-ISA),
- Video capture port (VxP500-to-CL550), and
- Video playback port (CL550-to-VxP500).

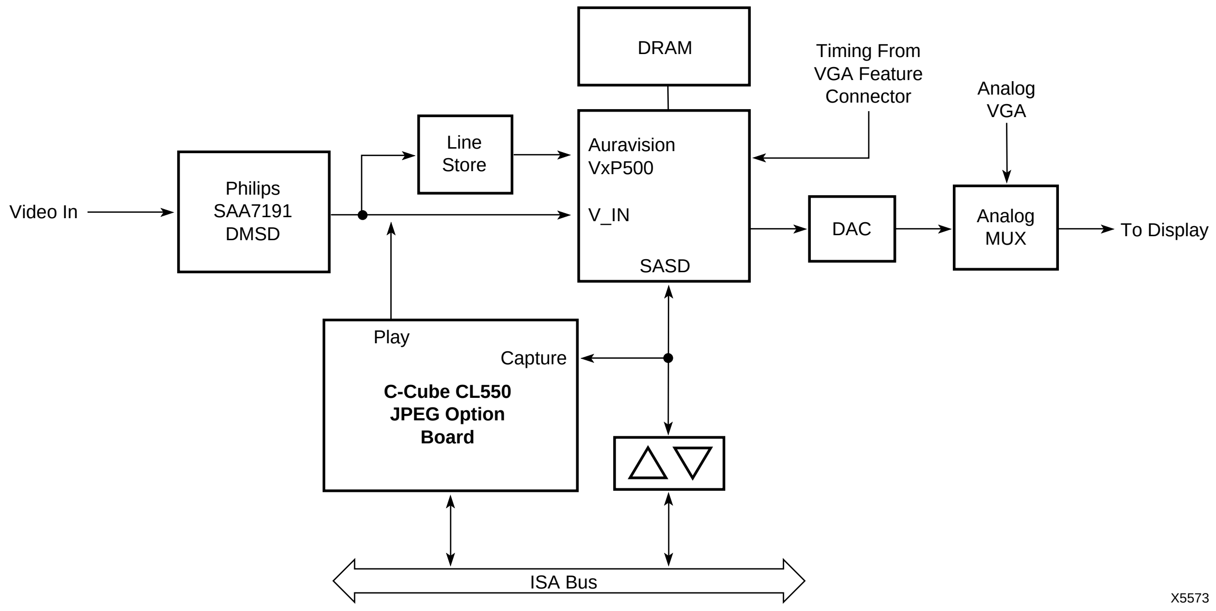

Figure 2. Simplified Video System Block Diagram

All datapaths are 16-bits wide, and the pixel format for compression/playback is YUV4:2:2, exclusively. The decision to use YUV4:2:2 format over RGB is two-fold. First, JPEG is much more efficient when compressing YUV4:2:2 over an equivalent RGB image, as YUV4:2:2 allows individual handling of luminance and chrominance components. Secondly, though the CL550 supports real-time RGB-to-YUV4:2:2 conversion, doing so would require 24-bit wide pixel input and output ports. Since the YUV4:2:2 format is common to both the CL550 and the VxP500, it requires the least amount of hardware to accommodate.

The decision to use YUV4:2:2 over YUV4:1:1 is simple: quality. For example, zooming a YUV4:1:1 format frame from 320 pixels/line to 640 pixels/line results in 8 times replication in the chrominance components, which is not very sharp - to say the least. Although the VxP500 provides real time YUV4:1:1-to-YUV4:2:2 conversion and we could theoretically use it to achieve lower cost in the video decoder, we make no attempt to support it here. Remember, you can only expect to get out of a codec what you first put into it. If you're using a high-quality JPEG codec like the CL550, then it's well worth the effort to start with a high quality image. For this design, YUV4:2:2 gives the optimal balance between system cost and image quality.

Capture/Compression From Live Input

For video compression, the Auravision VxP500 chip acquires incoming video by way of its V_IN port from an NTSC/PAL decoder such as the Philips SAA7191 or equivalent. Acquired frames are cropped, scaled, and then stored in the VxP500's DRAM to await compression by the CL550. When the system is ready to accept the compressed frame, the CL550 inputs the buffered frame from the VxP500's high-speed capture interface at up to 12.5 million pixels/second, passing that data to the host via the ISA port under CPU control.

Figure 3 shows a simplified flow diagram of the video compression/capture process, which is actually a chain of processes consisting of acquisition, cropping, scaling, buffering, and lastly, compression by the CL550 to the system. Where acquisition and cropping are obvious requirements for most designers, scaling and buffering are not. If ignored, this will greatly limit the potential of the JPEG compression system. It is worthwhile to consider these points further.

First, let's consider image pre-scaling support. In the ISA system environment, disk I/O rates are generally limited to about 500 KBytes/second. Pre-scaling of the incoming video prior to compression and zooming out on playback results in much higher image quality than using straight JPEG on full-resolution (640x480) frames. For example,

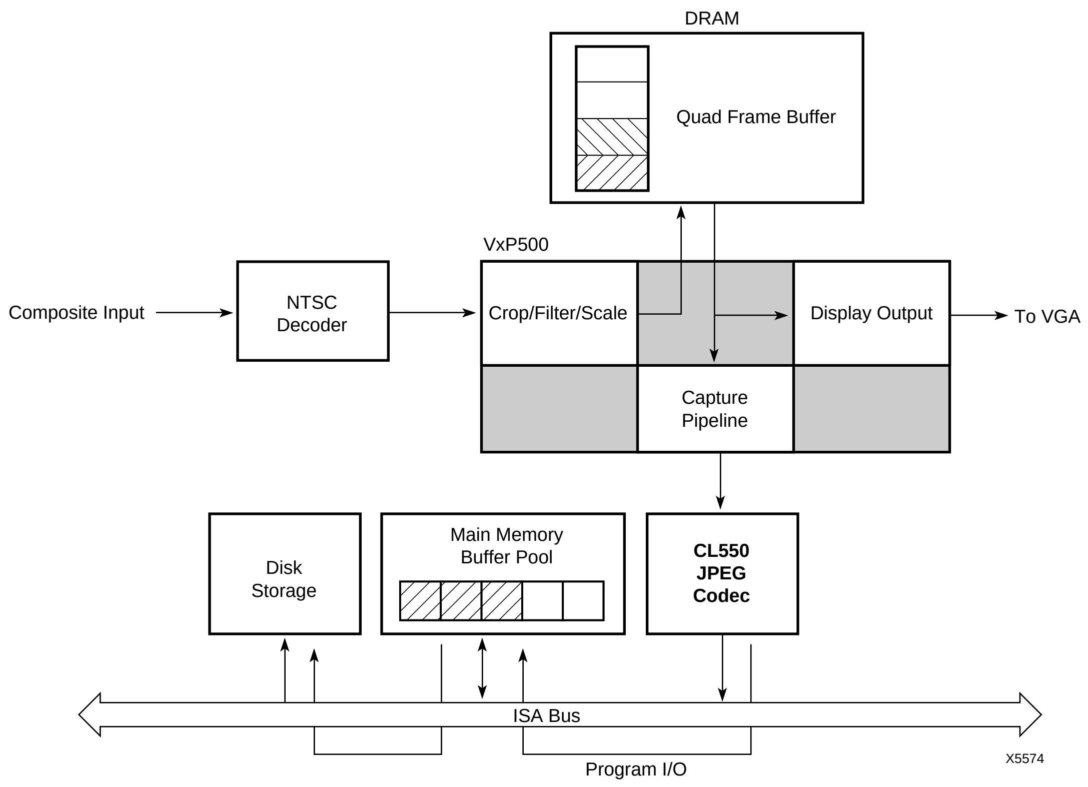

Figure 3. Flow Diagram of Compression From Live Video Input

at full resolution the incoming raw pixel stream averages over 18 MBytes/s (640x480 pixel/frame x 30 frames/s x 2 Bytes/pixel). To support real-time compression at a compressed data rate below 500 KB/sec would require a JPEG compression ratio of almost 40:1. The problem here is that JPEG video quality suffers dramatically at rates above 20:1. The output video quality in this case will be unacceptable to most users.

A better solution results from scaling the video images prior to compression, with optional zoom to full-screen on playback. In that case, the VxP500's pre-scaling function is used to reduce the resolution to 320x240 at 30 frames/s, cutting the raw pixel rate to less than 5 MBytes/second. This reduced rate allows JPEG compression ratios in the range of 10:1, which is optimal for the JPEG algorithm. This trade-off works well, maintaining high video quality at manageable disk I/O rates.

Another factor in determining video quality is the type of scaling used. There are several ways to scale the video, from simple pixel dropping to advanced multi-tap filtering approaches. For JPEG compression, it is preferable to use a multi-tap filtering approach. Not only is pixel dropping bad in terms of image quality compared to multi-tap filtering, it may introduce extra frequency components in the image as a result of under-sampling (aliasing). These extra frequency components will show up in the compressed data as extra overhead, giving lower compression efficiency. In the decompressed data these show up as increased artifacts, giving lower quality. Luckily for the CL550, the VxP500 provides an advanced 4-tap FIR horizontal scaling function, a key benefit for this architecture.

The other major benefit of the VxP500 for the CL550 is buffering of the pre-scaled frames. Because the PC is not a real-time platform, it will only accept data when it is ready and willing to do so. Incoming video data, however, is real-time. If the system is not ready for it, then it must somehow be buffered. The VxP500 handles this by being able to queue up to four frames at 320x240 resolution in its DRAM buffer, passing frames along for compression as the CL550 requests them. This allows the JPEG codec to sustain fairly long periods of system latency without losing frames, and it also allows the CL550 to operate with no additional memory. All buffering in this design is provided by the VxP500 and by available system memory, allowing absolute minimum cost in the JPEG compression system.

The next issue that needs to be addressed is fast codec access to the frames in the buffer. The CL550 is able to compress pixels at peak rates up to 12.5 million pixels/second at a clock rate of 25 MHz. In order to minimize time in the interrupt driver loop, it must be able to run at it's full rate. To support this, the VxP500 provides a direct high-speed codec interface port (mux'ed with the ISA bus signals). The capture interface is asynchronous, and allows the CL550 to stop and start as needed to support system overhead. Hardware required to connect the VxP to the CL550 for compression is minimal, consisting of a '573-type latch and a small amount of handshake logic contained in the FPGA.

Once the pixel data is transferred to the CL550, a program-I/O based device driver is used to unload the compressed output from the CL550 and transfer it to the system. This is a departure from earlier CL550 designs that used DMA, and it eliminates the DMA compatibility problems that are common with most ISA motherboards and adapter cards.

This also eliminates the additional logic required to manage DMA transfers across the ISA bus. In fact, this approach has proven to be slightly faster. Using a 486-66 motherboards at the standard ISA clock rate (8.3 MHz), the driver is able to sustain rates up to 500 KBytes/second to disk, consuming on average no more than 50% of the CPU in the driver. Typical time in the interrupt driver is 10 ms out of 33 ms each frame, with nominal compressed frame sizes in the range of 5K-15K Bytes. Some would argue that it is unjust to use the CPU for that amount of time. In reality, the system is being pushed to its full limit during an actual capture or playback operation. There is little disk bandwidth left to be doing anything else. From that point of view, use of the CPU is more than justified. It saves money and enhances system compatibility in the JPEG codec.

Video Decompression/Playback Operation

For decompression and playback of frames from the CL550, the VxP500 does not provide a dedicated port as it does in the capture direction. Instead, when decompressing video, the CL550 drives its output pixel data to the VxP500's VIN port, and shares a common video bus with the Philips 7191. A multiplexer on the base board is used to select between the 7191 and the CL550 as the current video source. There are some key differences, however, in the way the CL550 outputs data relative to the Philips 7191 that the VxP500 makes provisions for.

- The CL550's decompressed images are not full-size like standard video frames, as they were cropped and scaled prior to being compressed. The CL550's line width can therefore be as small as 80 pixels in some cases.
- The CL550's decompressed frames are presented in a progressive-scan fashion instead of interlaced-scan fashion as are standard video frames.
- The CL550's VSYNC- output signal is active low, where the Philips 7191's VSYNC output is active high.
- Most importantly, the CL550's output is asynchronous, stopping and starting on any pixel boundary as opposed to the 7191, whose output is constant and

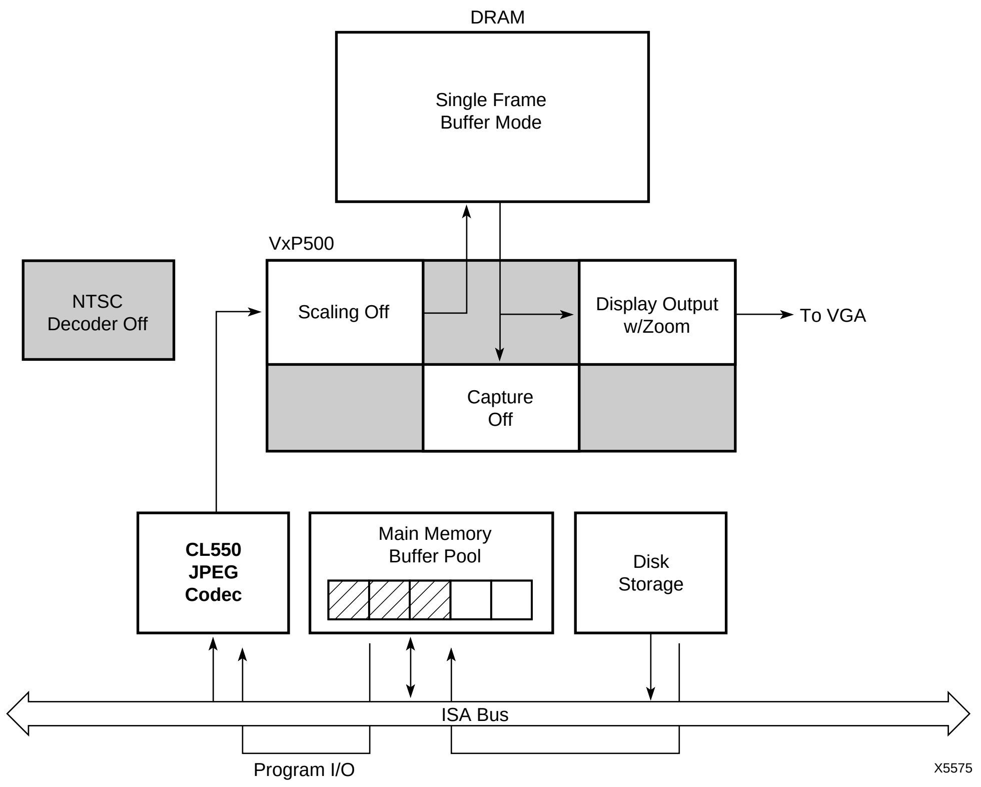

Figure 4. Video Decompression Flow Diagram

synchronous. By being asynchronous, the CL550 requires no extra memory to buffer compressed data. If the host is slow, the CL550 simply stops until the next word of data is available. The VxP500 is able to accommodate this operation by not displaying the incoming frame until it has been completely acquired in its overlay buffer. Figure 4 shows a simplified flow diagram of a video playback operation.

Still-Image Compression/Decompression Operation
Still-image compression is the process by which a RGB or YUV source image in memory is compressed using the CL550. For example, to compress a 24-bit RGB image in main memory, the source image in memory is converted to YUV4:2:2 format by the driver software and loaded to the VxP500’s overlay buffer by way of the ISA bus. Once in the overlay buffer, the CL550 is used to compress the image through the VxP500’s hardware capture port. The speed at which the image is compressed is largely dependent on the time it takes to load the raw pixels into the VxP500's frame buffer by way of the ISA bus. As the compressed data is much less, it takes far less time to transfer. Even with the ISA bus limitation, the compression operation is still much faster than software-only approaches, greatly assisting editing and effects generation operations.

One drawback with this approach is that the size of the bit map that can be compressed at one time is limited by the size of the overlay buffer. However, it is possible to compress large images in successive strips or tiles, as long as the scan order required by JPEG is maintained.

Decompression of still images is accomplished in the reverse order of the compression case. First the compressed image is decompressed by the CL550 into the VxP500’s overlay buffer. This image is then unloaded via the ISA bus, converting to RGB in software if needed.

System Limitations

Though this design is advanced in many ways, this architecture does have limitations that system designers and programmers should be aware of. These points are listed below.

VxP500 DRAM Bandwidth Limitations. The VxP500 achieves lower overall system cost by using DRAM instead of VRAM for it’s overlay buffer. The trade-off is that the display refresh must use the DRAM’s random access port. This leaves less overall bandwidth for the incoming source video and outgoing video to the codec. This limitation will be most severe during compression, where three resources, display, input, and capture, contend for DRAM access.

When using the 48-bit wide frame buffer mode, the total peak DRAM bandwidth available is just below 100 MBytes/s. The maximum sustainable data rate lies in the range of 80-90 MBytes/s. The arbitration priority for the memory is display refresh, video input, and then capture output. Let's consider some cases here.

For a display resolution of 640x480 and a refresh rate of 60 Hz, approximately 50 MBytes/s of the available DRAM bandwidth are consumed for display refresh. The input pipeline consumes another 6.75 MBytes/s. At this rate, there is enough bandwidth remaining to sustain the capture data flow at the CL550's full 25 MByte/sec peak pixel rate with no problems. On the average, the CL550 will be operating in the 10-20 MByte/sec rate, depending on the rate of compression. Lowering the compression level reduces the average pixel rate.

The problem occurs at 800x600 resolution. In this case the refresh load on the DRAM jumps to nearly 80 MBytes/s, leaving little bandwidth to sustain the CL550’s capture rate. At times during the capture sequence, the VxP500’s internal capture data FIFO may starve for data, and the SDDIR signal will go low, causing the /STALL signal to be asserted on the CL550. This increases the length of time the driver must spend compressing the image to main memory. In the worst case, frames may be dropped by the driver in order to maintain time synchronization in the video.

There are only two possible solutions to the DRAM bandwidth bottleneck issue:

• Disable display output during capture. This gives plenty of bandwidth, but users usually like to see what they are recording.

• Use horizontal zoom to reduce the number of pixels per line required by the display. This is little better than shutting off the display, but only a portion of the frame being captured is actually displayed.

System Bandwidth Limitations. For most ISA PC systems incorporating a 486 or better CPU, the compressed throughput will be a function of disk transfer rate. A prototype unit was run on a Compudyne 486DX2-50 with 256K cache, 8 MBytes RAM, and a Western Digital 420Mbyte IDE drive, with the bus clock set to 8.3 MHz. This system had no problems running at up to 300 KB/second average rate to disk. At around 350-400 KB/s, we see occasional frame drops, but they are distributed in such a way as to be few and far between, which is a direct benefit of the VxP500’s quad buffer mode.

In general, the target “top-end” for this design is about 400-500 KBytes second on the fastest ISA platforms. That equates to about 50% CPU time in the codec service routine, which is about as high as we want to go. At that rate, video quality is surprisingly good.

The critical item for system bandwidth is drive throughput: The faster the better. When selecting a drive, specify maximum sustained transfer rate as opposed to access time, as that is the critical parameter for sustained real-time capture/play throughput. Although SCSI drives may offer a slight performance advantage, all of our prototype systems were developed using off-the-shelf IDE drives.

Context Switching Issues. Although dedicated specialty devices like the CL550 and the VxP500 are beneficial in terms of relieving the load on the system CPU, their remoteness to the CPU introduces context-switching issues that programmers should be aware of. Normally, a software designer could start any number of compression, decompression, and video handling tasks at the same time. For example, a Windows-based editing application may think that it is perfectly legal (and it is) to open a compression driver and a decompression driver at the same time. In reality, though, at the machine level the operating system would be switching between these tasks to give the appearance of simultaneous operation.

Hardware-accelerated video computing systems, however, are quite different, because a video computing system is really a chain of processors running end-to-end. Coordinating all of these processing stages at the driver level is a real challenge for the programmer. The main concepts that we stress to programmers is that it is not always possible to: (a) interrupt the processing chain and restore it's state later, and (b) know the exact status of the processing chain at a particular instant in time.

For this particular design, as will be the case in many accelerator-based designs, the compression and decompression processes are NOT re-entrant, and cannot operate concurrently in the same hardware system. It is not possible to save the state of the real-time processing chain and re-store it some time later. Once an operation is started at the hardware level, it must be allowed to finish or it must be aborted altogether. Developers must carefully schedule incoming commands within the driver in order to support applications that open concurrent processes.

Interface Logic Design

This section details the interface logic used to drive all control signals on the Option Card. For specific technical information on each of the devices used in this design, readers should obtain the *C-Cube CL550 JPEG Compression Processor User’s Manual, the Auravision VxP500 Video Processor Databook, and the Xilinx Programmable Logic Databook.*

All interface logic for this design was implemented using a single Xilinx XC3020A™ FPGA. The 3020A is the smallest in the Xilinx 3K series, and has just enough density to implement all required logic functions for the daughter card. These logic functions include:

- Clock Generation
- ISA Bus Interface
- VXP500 Compression Interface Control
- VXP500 Playback Interface Control

In addition to high integration, the design is partitioned in a modular fashion for easy customization. If more density is needed, designers can use the XC3030A™ or XC3042A™ devices without having to change package type or pinout. For example, a designer can take the ISA interface from this design and quickly drop it into any other design that uses the CL550 on the ISA bus.

Figure 5 shows a signal-level block diagram of the logic modules that make up the interface between the ISA bus, the CL550, and the VxP500. Refer to the board-level and FPGA schematics for specific implementation details.

CL550 Clock Generation

Figure 6 shows the waveforms for the clock generator. In this design, all clock signals are derived from the VxP500's MEMCLK signal, which eliminates the need for an oscillator on the daughter card. The MEMCLK input is 50.0 MHz, with a 50% duty cycle. From this input, three clocks are derived. These are PXCLK, PXPHSE, and HBCLK. PXCLK is divided from MEMCLK and runs at 25.0 MHz. PXPHSE is divided from PXCLK and runs at 12.5 MHz, but phased such that the PXPHSE transitions are coincident with the high-to-low transitions on PXCLK.

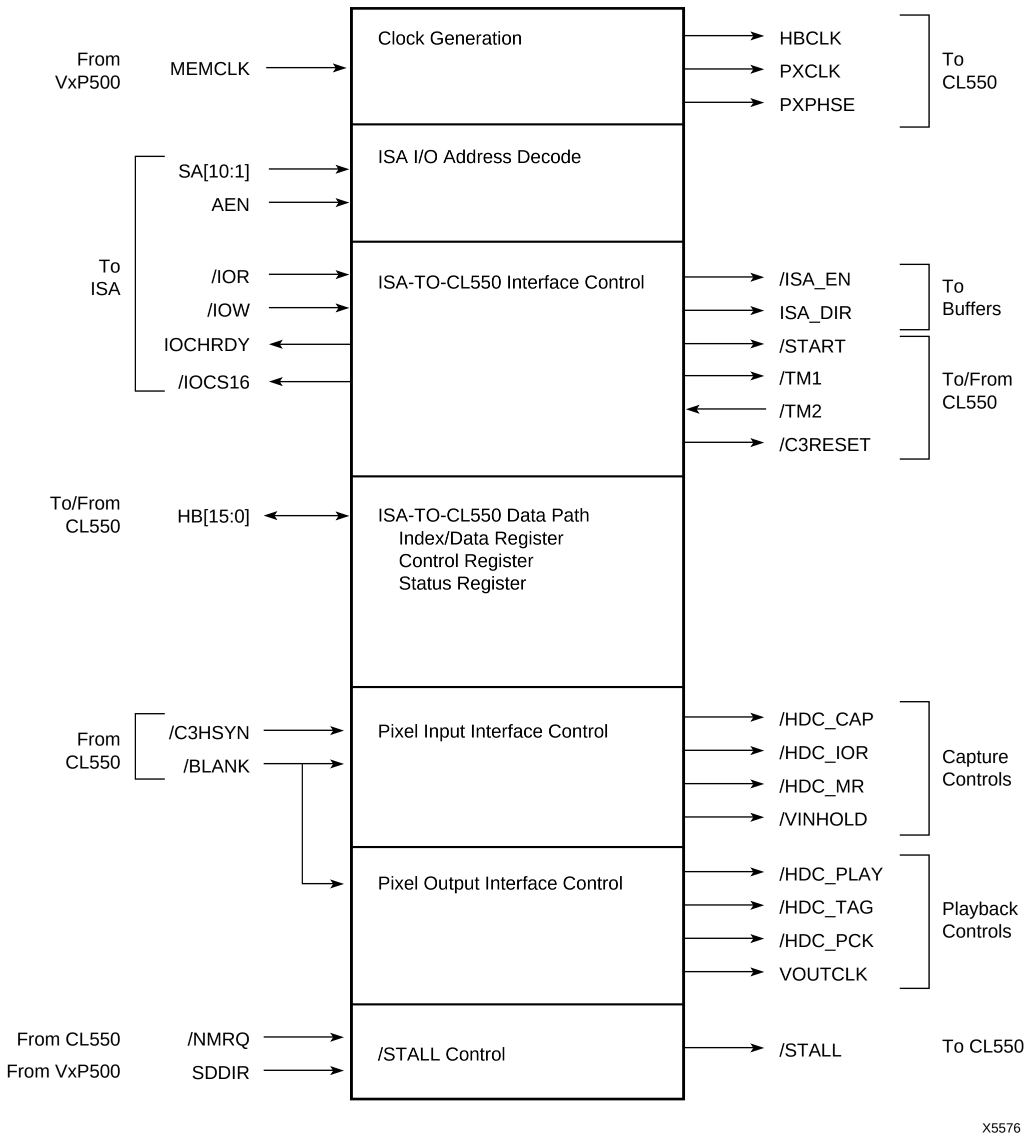

Figure 5. Interface Logic and Signal Diagram

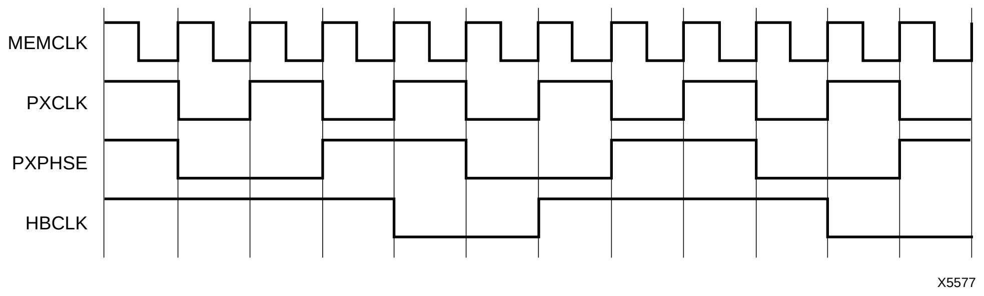

Figure 6. Clock Timing

Lastly, HBCLK is obtained by using a divide-by-three of the PXCLK signal, and runs at 8.33 MHz. The HBCLK duty cycle is 66/33 (two PXCLK's high, one PXCLK low). Refer to the FPGA schematic for implementation details.

ISA I/O Address Decode

The CL550 codec module is mapped into ISA I/O address space in four consecutive 16-bit I/O ports, called:

- C3Data
- C3Index
- C3Control/C3Status
- C3Codec

The C3Data port is used for reading and writing CL550 device registers, with the register address being held in the C3Index port. The C3Control port controls board-level functions, and the C3Status register allows programs to check board-level status information. The C3Codec port is directly indexed to the CL550's Codec register for fast access. A full description of the program interface is given in the "Program Interface" Section.

ISA Interface Control

This block of logic is used to control the transfer of data between the ISA bus and the CL550. Four basic types of transfers are supported:

- *Port Write Transfers* - used to load the C3Index and C3Control ports.
- *Port Read Transfers* - used to read the C3Status port and can be used for reading back the C3Index register as a diagnostic check.
- *CL550 Write Cycles* - used when writing to the C3Data or C3Codec ports.
- *CL550 Read Cycles* - used when reading from the C3Data or C3Codec ports.

Figure 7 illustrates the timing of the Port Write operation. The Port Write begins when a valid I/O address appears on SA[10:1]. In response to this address, the address decoder generates the internal signal /550_INDX or /CTL_STS (point 1 in the Figure 7). Next, the ISA bus write command strobe is asserted by the system. In response to /IOW, the IOCHRDY line is immediately driven low (point 2), so that the command strobe can be synchronized to the HBCLK (LCMD, point 3). Once LCMD is active, a state machine, CP[1:0] is activated. This state machine generates a synchronous pulse of one HBCLK duration on CP0. CP0 is used to create data latching strobes IDX_WP, CTL_WP (point 4). When CP1 goes high, the data has been latched and the IOCHRDY line is released, terminating the transaction (point 5).

Port read timing is shown in Figure 8. This operation is very similar to the Port Write, except that here the /XIL_OE signal is asserted to enable the FPGA's output drivers to drive data onto the HB[15:0] bus, as shown.

CL550 register write timing is illustrated in Figure 9. There are two variations of this transaction, one is used when writing to the C3Data port, and the other for writing to the C3Codec port. The C3Data port can be used to access all CL550 device registers using the index stored in the C3Index port, whereas the C3Codec port is hard-wired directly to the CL550's codec register. When accessing the C3Data port, the cycle starts out much the same as the Port Write, except now, when CP0 is active, the following events occur:

- /START is asserted to begin the register write transaction to the CL550.
- /TM1 is driven low from hi-z to indicate the direction is ISA-to-CL550.
- /XIL_OE is activated (and /ISA_EN negated) so that the pre-stored register index (0x8000-0xFFFC) is driven onto the HB[15:0] bus.

On the next HBCLK cycle, the CL550 inserts one idle cycle (point 5), followed by an acknowledge cycle (point 6). When TM2 is sampled low, the signal LTM2 asserts, releasing the IOCHRDY line (point 7), allowing the ISA cycle to terminate. The C3Codec write cycle timing is almost identical to this, except during the /START cycle (point 4), no index value is written. Rather, the bus line HB15 signal is driven low to automatically address the CL550's Codec register (indexes 0x0000-0x7FFC). The auto-index feature allows faster access to the Codec register, which improves performance in the driver.

I/O Port Read transactions from the CL550 are illustrated in Figure 10. Again, there are two variations of the transfer, with the C3Data port used to read any CL550 register and the C3Codec port being directly wired to the Codec register. The transaction begins in the same fashion as the write, except that the ISA transceiver direction is reversed. During the /START pulse, TM1 is at a high level to initiate a CL550 register read cycle. When the CL550 asserts /TM2 low (point 5), the valid read data is latched into the C3Index register, overwriting the previously stored index. As the CL550's output bus releases (point 6), the FPGA is used to drive the latched data back onto the HBUS through the end of the /IOR strobe, as shown.

Capture Interface Control

This section describes the operation of the pixel interface between the VxP500 and the CL550 during the compression process. Refer to the board-level schematics for the specific signal connections.

The VxP500 provides a high-bandwidth, asynchronous pixel port that is designed to support compression devices like the CL550. A block of logic within the FPGA is dedicated to managing the flow of pixel data across the interface. The hardware capture port is multiplexed with the ISA data bus. When the VxP500 is in hardware capture mode, the signals SASD[15:0] become pixel data outputs to the hardware codec, with SASD[7:0] carrying the luminance component (Y), and SASD[15:8] carrying the chrominance components (U and V). For this design, the pixel data is always assumed to be in the YUV4:2:2 format, transferred in the order YU, YV, YU, YV...and so on.

In addition to the SASD[15:0] bus, three pixel transfer control signals are multiplexed with the ISA bus control signals /MEMRD, SDDIR, and /IOR. In hardware capture mode, the signal /MEMRD becomes the pixel output strobe. Each low-to-high transition on the /MEMRD input will cause the next pixel to appear on the SASD bus from the VxP500's capture FIFO. The signal SDDIR, when high, indicates to the codec that the next pixel is ready for reading by the CL550. When SDDIR is low, then /STALL must be asserted to hold off the CL550 until the next pixel is ready. When all the pixels have been transferred, the

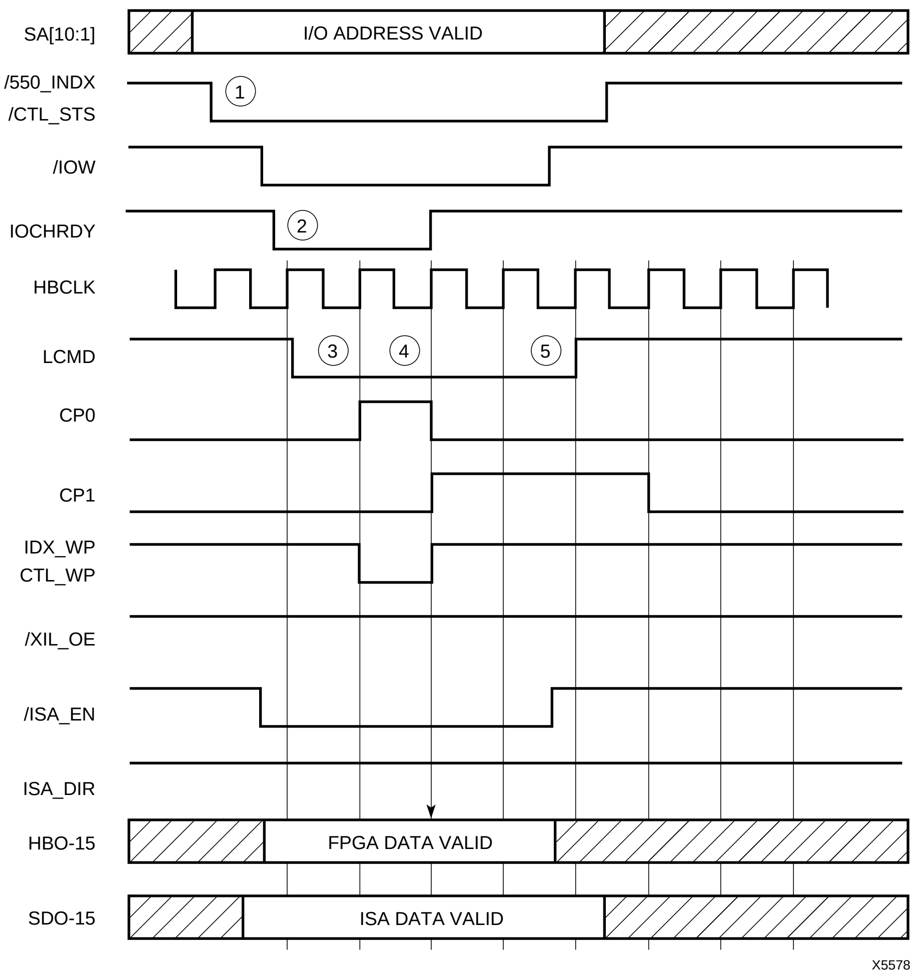

Figure 7. C3Index, C3Control Port Write Timing

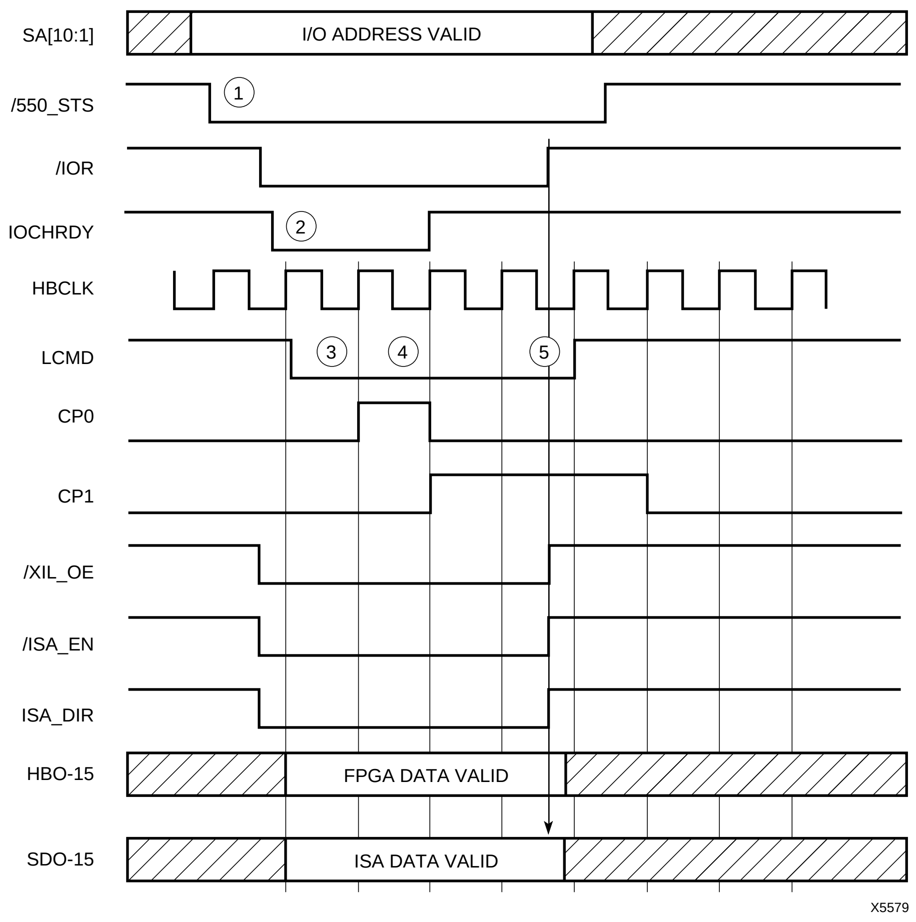

Figure 8. C3Status Port Read Timing

VxP500 is released from hardware capture mode by a low-to-high transition of the /IOR input.

The VxP500 is placed into hardware capture mode by software, using ISA bus write operations. However, once in hardware capture mode, the VxP500 no longer responds to ISA I/O accesses. For this reason, /IOR is used as a hardware release. Because the VxP500's input signals /MEMRD and /IOR are shared with the ISA bus and the codec, they are multiplexed using a '157 type mux. Here, the ISA's /MEMRD is multiplexed with the signal /HDC_MR, and the ISA's /IOR signal is multiplexed with the signal /HDC_IOR. The mux is controlled by the hardware codec via the signal /HDC_CAP. /HDC_CAP, when low, gates the VxP500's control inputs to the /HDC_MR and /HDC_IOR outputs from the codec. /HDC_CAP has a pull-up resistor on the base board to allow operation without the codec option card.

The compression process begins by initializing the VxP500's capture pipeline, defining a window in the overlay buffer that is to be compressed. Once the initialization is complete, the VxP500 is programmed to enter hardware compression mode, and the mux is switched by writing to C3Control port bit 1 (HDC_CAP). At that point, the CL550 can be started for compression. The CL550 signals used to control the transfers are /HSYNC (referred to as /C3HS), /BLANK, PXPHSE, PXCLK, and /NMRQ. From these signals, all other control signals can be derived.

Transfer timing for the hardware capture port is illustrated in Figure 11. The capture sequence is initiated with the VxP500 in hardware capture mode with the first pixel of the image present on the SASD bus. Each time the signal /HDC_MR goes from low to high, the next pixel will appear a maximum of 80 ns later. In order to insure that the second pixel is out on time, /HDC_MR must be asserted prior to the first /PXIN strobe. To do this, the /HSYNC signal is used. With the CL550's HDelay register programmed for one pixel blanking, the /HSYNC signal will assert exactly one pixel time ahead of /BLANK going from low to high. /HSYNC is also used to stop requesting pixels exactly one pixel before blank goes high to low. This requires the HSync register to be programmed exactly with respect to the width of the image. The HPeriod register is programmed such that there is blanking time at the end of each line, so that the capture width adjust logic can remove excess pixels. The capture width adjust mechanism is described in the next section. See the FPGA schematic for the specific logic implementation.

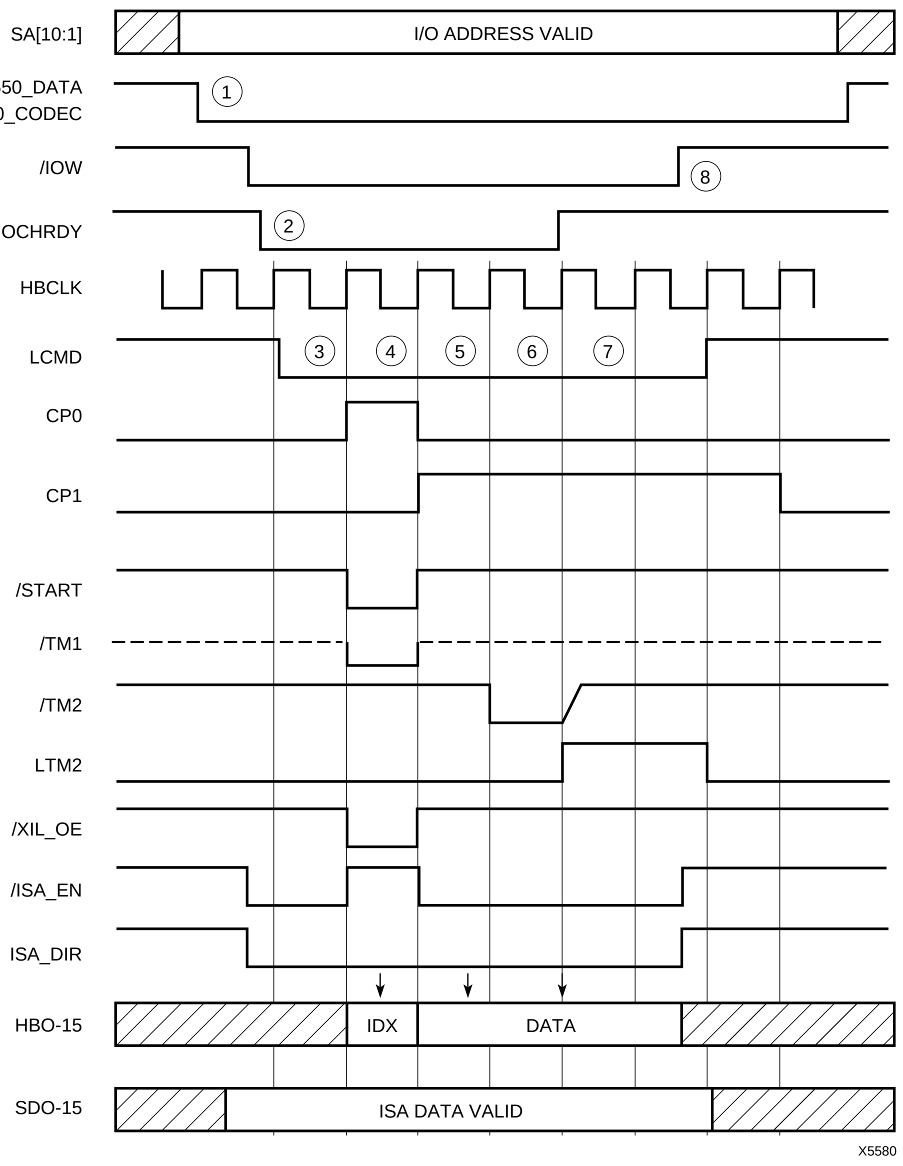

Figure 9. C3Data, C3Codec Write Timing

Capture Width Adjust Logic.
One of the unique aspects of designing for the VxP500 is that when using the 48-bit wide memory mode with YUV4:2:2 capture format, the capture window width must be a multiple of six, as this is the size of the VxP500's minimum pixel group for that frame buffer arrangement. (See the Auravision VxP500 Databook for more information.) This creates a problem for the CL550, whose compression window width must be an even multiple of 16 pixels when compressing YUV4:2:2 video. For example, let's say we want to compress an image that is 320 pixels wide. To do this, the VxP500's capture window must be set to 324 pixels, as this is the next higher multiple of six above 320. In this instance, there are four excess pixels per line that must somehow be removed from the VxP500's capture FIFO without being taken by the CL550. To correct for this difference in widths, a special state machine had to be designed to trim the excess pixels from the VxP500's capture FIFO at the end of each compressed line while the CL550 is between lines. For all CL550 image widths supported in this design (all multiples of 16 from 32 to 320), there will be either zero, two, or four excess pixels per line that need to be trimmed by this state machine. The pixels are removed during the CL550's horizontal blanking interval, after the /BLANK signal goes high-to-low. Figure 12 illustrates this operation when four excess pixels are removed. To accommodate the width adjust logic, the CL550's HDelay and HPeriod registers must be set exactly to achieve proper signal timing. Refer to the compression code examples for the specific formulas.

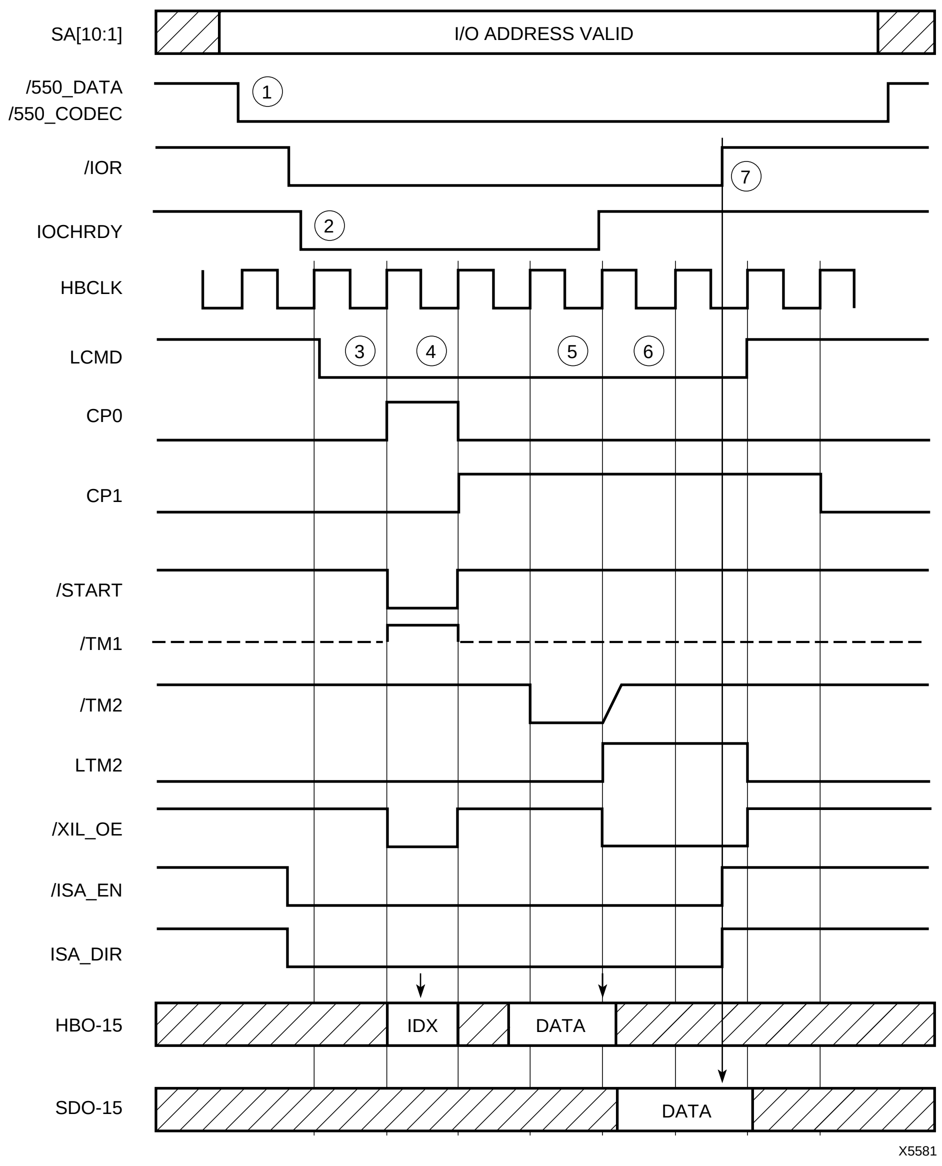

Figure 10. C3Data, C3Codec Read Timing

The state machine is configured via two bits in the C3Control register, PADCTL_0 and PADCTL_1. Programming of these bits is described in the “Program Interface” Section. The state sequence for the width adjuster is shown in Figure 13. Refer to the FPGA schematic for implementation details.

Decompression/Playback Interface Control

This section describes the operation of the decompression interface to the VxP500. Refer to the board-level schematics for specific signal connections.

For decompression of video or images to the VxP500's overlay buffer, the CL550 writes YUV4:2:2 format pixel data to the VxP500’s VIN port. The VIN bus is shared with the NSTC decoder, which is placed into high-impedance state by software before enabling the CL550’s output port. A '157 type multiplexer on the base-board is used to multiplex three VxP500 control signals. These are VIN_CLK, VIN_VS, and VIN_HS. The signal /HDC_PLAY is used to switch between the NTSC decoder and the CL550 codec board. /HDC_PLAY is pulled high with a resistor to allow operation without the codec daughter board. When /HDC_PLAY is asserted by programming the C3Control port bit 0 (HDC_PLAY) to one, the signals VIN_CLK, VIN_VS, and VIN_HS are connected to HDC_PCK, HDC_VSYN, and /BLANK, respectively. A fourth VXP500 control input, /VIN_TAGN is connected directly to the signal /HDC_TAG. /VIN_TAGN is pulled low using a 2K ohm resistor to allow operation without the daughter board.

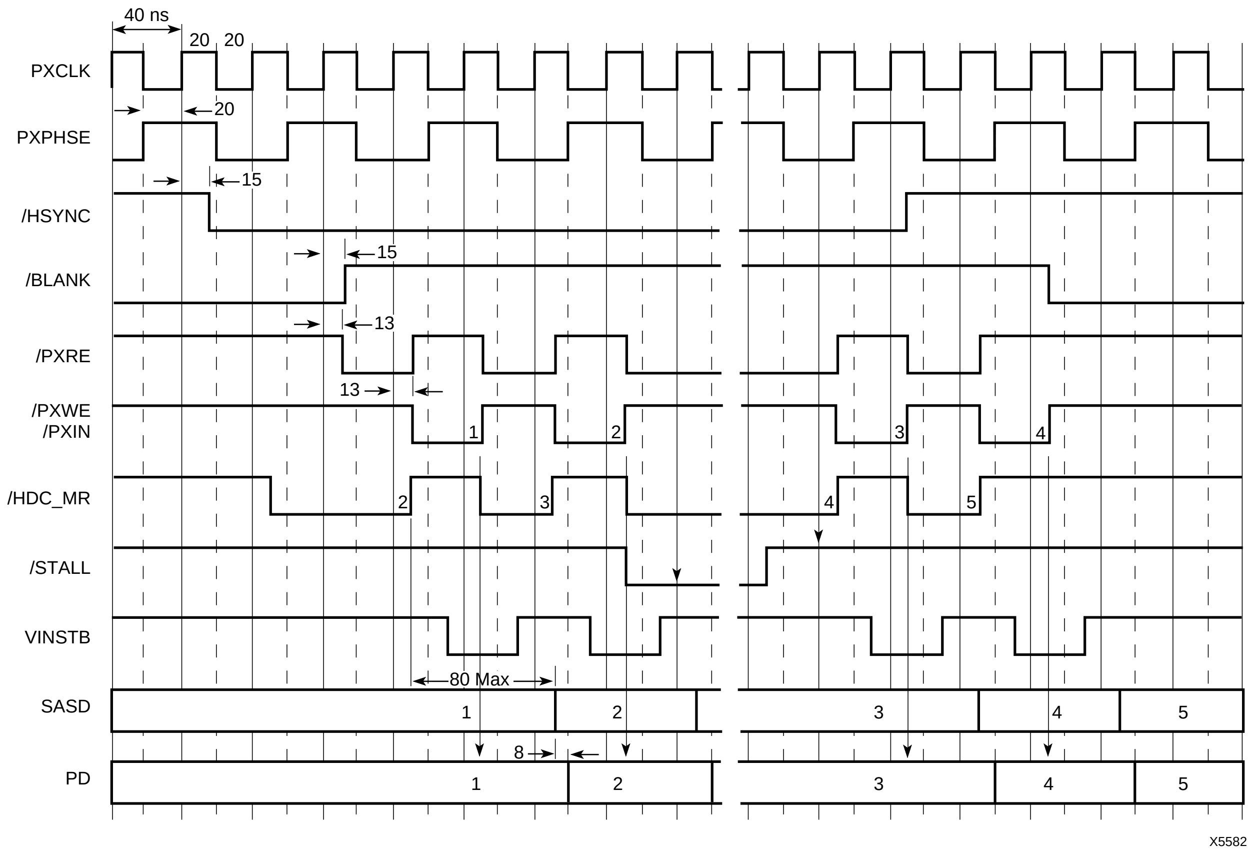

Figure 11. VXP500-to-CL550 Video Capture Timing

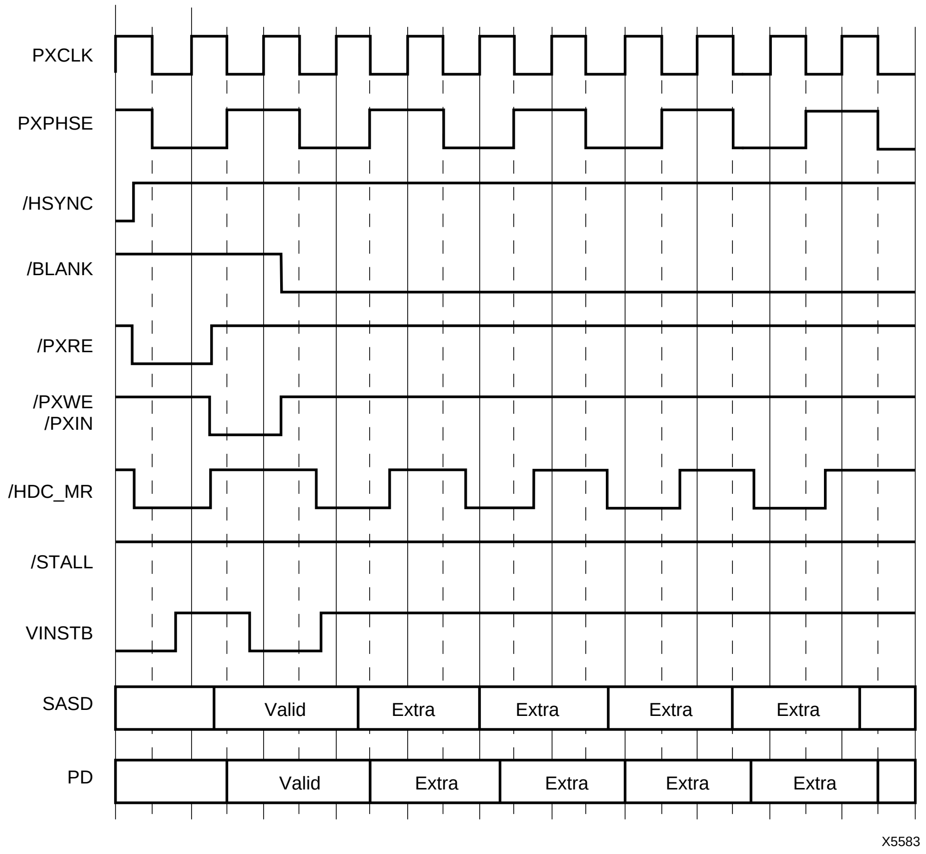

Figure 12. Capture Width Adjust Operation to Trim Four Excess Pixels

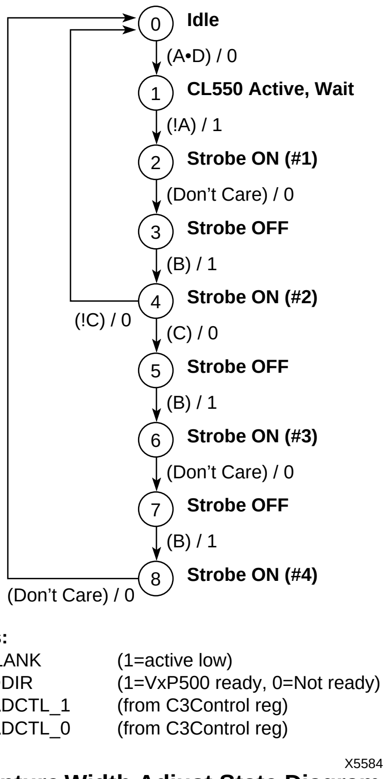

Figure 13. Capture Width Adjust State Diagram

The CL550 transfers pixels in a different manner than the NTSC decoder, and the VxP500 needs to be programmed differently in order to receive the pixel data correctly:

- The CL550 has an active-low VSYNC output, as opposed to the Philips VREF, which is active-high.
- The CL550 outputs a rectangular image in progressive-scan order, as opposed to interlaced-scan order.
- The CL550's output image can vary in width and height, depending on what was compressed originally.
- The CL550, being a slave to the system, may stop sending pixels at any time, depending on the ability of the host to provide compressed data to the codec. In contrast, the Philips video stream is fixed and synchronous.

Once the VxP500 is initialized to accept CL550 input, the next step in the decompression process is to disable the Philips output buffers by writing to it's output control register. This action places the 7191's pixel data outputs in a high-impedance state. At this time, the hardware codec's output drivers can be enabled by writing to the C3Control port. When the HDC_PLAY bit is programmed to 1, a block of logic in the FPGA synchronizes the switching of the mux with the LL3 clock, so as to produce no glitches on the VxP500's VIN_CLK line. This logic block waits for LL3 to reach a known state before switching the MUX. Once switched, the VIN_CLK input is then driven by the HDC_PCK output of the FPGA.

At this point, the CL550 is initialized and started for decompression. The first event that occurs is the assertion of the /HDC_VS signal by the CL550. When /BLANK goes high, valid pixels are clocked into the VxP500 on the rising edge of HDC_PCK. When the /STALL input to the CL550 asserts, temporarily stopping the flow of pixels, /HDC_TAG is driven high to prevent the VxP500 from clocking in bad pixel data. Timing of the decompression interface is shown in Figure 14. Refer to the FPGA schematics for the specific logic implementation.

Program Interface

This section describes the I/O port definitions for the JPEG codec design. Refer to the CL550/CL560 Databook and available CL550 programming examples for more information on accessing and programming the CL550 registers.

Base Address Decode

The address decode block provides a full 16-bit I/O address decode of ISA address lines SA[15:1]. Two pin jumpers on the board allow the user to select between one of four base addresses which are embedded into programmable logic. Table 1 shows the base I/O address settings for the compression card. Base address decode logic is implemented using programmable logic, so these addresses can be changed to suit particular applications.

Table 1. Base I/O Address Configurations

JP2	JP1	Base Address
Closed	Closed	0x380
Closed	Closed	0x3A0
Open	Closed	0x3C0
Open	Open	0x3E0

I/O Port Definitions

The program interface to the compression board consists of four 16-bit I/O ports, at offsets of 0, 2, 4, and 6 from the selected base address. These ports are described below.

Base+0 CL550 Host Data Port (C3Data)
This 16-bit read/write port is used to pass data to and from the CL550's registers. These data include programming/status information and JPEG-compressed data. Before accessing this port, a valid register address must be loaded to the CL550 Register Index Port

Base+2 CL550 Register Index Port (C3Index)
This 16-bit register, internal to the Xilinx, stores the address of the CL550 register being addressed. To access a CL550 register, the program must first load the CL550 register

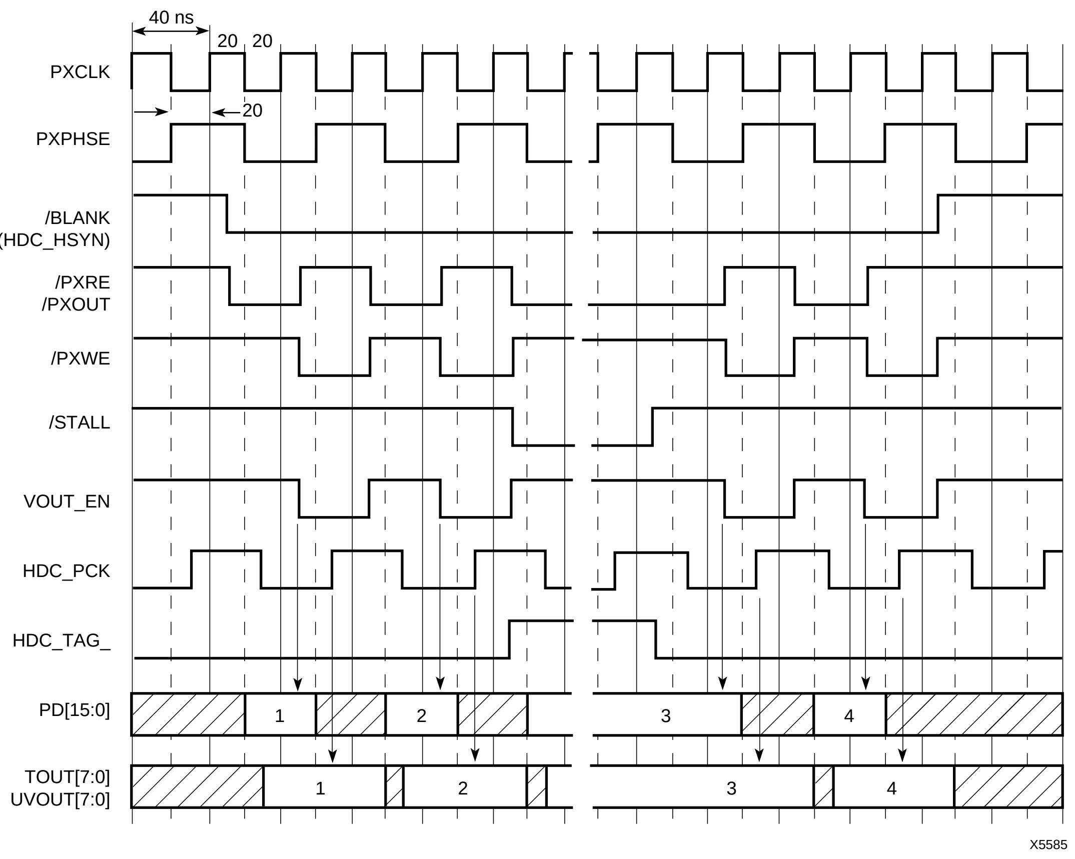

Figure 14. CL550-to-VxP500 Playback Timing

address into the index port, then read from or write to the host data port. The value written to this register can be read back, but it is volatile. Any read operation to the CL550 Host Data Port or the Codec port destroys the previously loaded index.

Base+4 Board-Level Control/Status Port (C3Control/ C3Status)

This port, internal to the FPGA, is a dual-function port. Writes are directed to the C3Control Port, and reads are directed to the C3Status port. Bit definitions for the ports are listed below.

C3Control Port (write only)

This port is used to configure the board-level logic functions.

Bit 0:PLAY_EN

When this bit is set to one, the video output port will be enabled. ALWAYS be sure to disable the outputs of the 7191 via the I2C prior to setting this bit. (Set the SAA7191's OEY and OEC control bits to zero to allow FEIN-bus control.) *Damage to the hardware could result if this rule is not followed.*

Bit 1:HDC_CAP

This bit, when set to 1, enables the video capture port. HDC_CAP has priority over PLAY_EN such that if PLAY_EN = 1 and HDC_CAP = 1, the output pixel port will be disabled, and PLAY_ON (C3Status port) will read zero.

Bit 2:HDC_IOR

This bit is used with compression, and controls the release of the VxP500 capture interface. When the VxP500 is in capture mode, this bit is programmed to oneand then back to zero in order to release the VxP500 capture port.

Bit 3: C3RESET

A one in this bit drives a hard reset to the CL550. A zero to this register bit clears reset.

Bit 4: PADCTL_0

Bit 5: PADCTL_1

This bit field is used to enable the capture width adjust mechanism. This mechanism must be used in cases where the VxP500's frame buffer is configured as 3-way interleave

(48-bit wide), YUV422 format. In this case, all VxP500 output frames for compression will be padded to the next multiple of 6 pixels wide. Therefore, in order to support standard widths 160, 320, etc., the capture width adjust is used to remove the excess padding pixels from each line. The programming of PADCTL_1:0 is then dependent on the remainder of width/6, as shown.

PADCTL 1:0	Function
0 0	No Function. Use when width%6=0
0 1	Remove two pixels/line. Use when width%6=4
1 0	No Function – undefined
1 1	Remove four pixels/line. Use when width%6=2

Bits 6-15:Undefined

C3Status Port (read only)

This port is used to obtain board-level status information.

Bit 0: PLAY_ON

A one in this bit indicates that the codec modules output pixel port is enabled and that the video input mux is switched to the codec instead of the input video decoder.

Bit 1: SDDIR

When this bit is one, the VxP500's capture pipeline contains data and is ready for the compression to start. Do not enable the CL550 until this bit is 1, as STALL- is generated whenever SDDIR is low.

Bit 2: C3DRQ-

This bit reflects the state of the CL550's DRQ- signal. Zero in this bit indicates DRQ- is asserted low. This bit has no logic function relative to the hardware interface and is not currently used in the logic. This signal has been provided for software drivers to get faster access to CL550 flags status via the DRQ- signal. With this approach, the CL550's codec status can be checked using only one I/O read as opposed to the traditional write-index/read-CL550 protocol (50% faster access). Programmers should note that DRQ- must be initialized via the DRQ Mask Register before it can be used for polling. As this signal is a multi-purpose status output, the programmer is responsible for it's use. For example, on compression, you could program the CL550's DRQ- mask register with 0x2040 (CodecNotBusy & 1/2_Full). When C3DRQ- reads zero, the codec is ready.

Bits 3-15:Undefined.

Base+6 CL550 CODEC Data Port (C3Codec)

This port is automatically indexed to the CL550's Codec Data port, so reads or writes to this port do not require an Index value to first be written. This approach results in highest possible CL550-to-ISA transfer rates when using the program-I/O based drivers.

Programmers should read/write this register a minimum of two times per flags read in order to assure 32-bit alignment of the compressed frames. On compression, it is possible to read up to 4 longwords (8 reads to the codec port) provided that the CL550's internal FIFO is at least half full. The same applies on decompression, provided that the FIFO is below 1/2 full before the first codec write. Also note that extra ISA wait states may be incurred when writing the 2nd, 3rd, or 4th longwords of a group, depending on codec status, but these wait states should not cause problems in the system.

Schematics

Schematic diagrams for the CL550 daugther board and for the Xilinx FPGA are provided here for your reference. A schematic diagram for the Auravision VxP500 Evaluation Board can be found in the *Auravision VxP500 Databook.* Contact Auravision Corporation for more information on the VxP500 or the base board design.

Schematics were entered using OrCAD 4.12 for DOS. The Xilinx XACT 4.0 and OrCAD interface tools were used for routing the FPGA. To obtain design information on PC diskettes, contact:

C-Cube Microsystems
JPEG Products Group
1778 McCarthy Boulevard
Milpitas, CA 95035
Phone: (408) 944-6300
Fax: (408) 944-6314

Additional Information

If you wish additional information on Xilinx, contact the sales office nearest you from the list on back. For information regarding the Auravision or Ring Zero products involved, contact the companies directly:

Auravision Corporation
47865 Fremont Blvd
Fremont, CA 94538
(510) 252-6800

Ring Zero Systems
1650 S. Amphlett Blvd. #300
San Mateo, CA 94402
(415) 349-0600