XAPP625 (v1.0) March 12, 2002

SDI: Video Standard Detector and Flywheel Decoder

Author: John Snow

Summary

The SMPTE 259M Serial Digital Interface (SDI) standard describes how to transport standard-definition digital video serially over video coax cable. SDI is commonly used as the video transportation backbone of most broadcast studios and video production centers. This application note describes implementations of a video standard detector and a flywheel video decoder, suitable for use with Xilinx FPGAs.

Introduction

This is one in a series of application notes describing SDI implementation in Xilinx FPGAs. Figure 1 is a block diagram showing correlation between the various application notes and the elements of the SDI link.

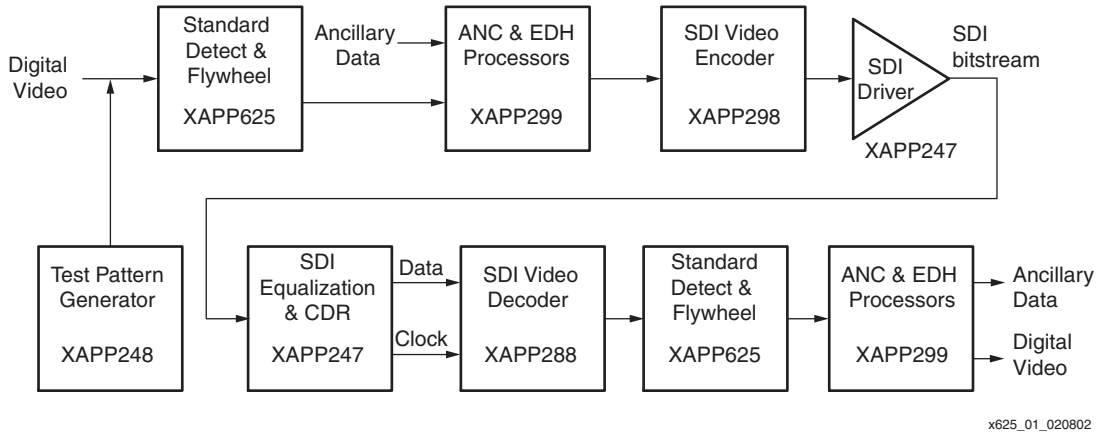

Figure 1: **SDI Block Diagram and Application Notes**

Before transmission over an SDI link, digital video is usually processed to insert error detection checkwords. These checkwords allow the receiver to detect transmission errors. Ancillary data packets can also be inserted into the inactive (blanked) portions of the video to carry non-video data, such as digital audio. At the receiving end of the SDI link, the digital video is again processed to detect transmission errors, extract ancillary data, or insert additional types of ancillary data.

SDI is compatible with a variety of different digital video standards. Because the locations of the error detection and ancillary data packets vary with the digital video standard, a video processor must know which video standard is currently being processed. Most digital video processors have a video standard detector examining the video stream to automatically determine the video standard.

In addition to determining the video standard, a video processor must also synchronize itself to the input video stream. It must know the vertical and horizontal position of the current video sample. The video decoder function synchronizes to the input video stream and keeps running counts of the current video line number and current horizontal position. With this information, the video processor knows when to insert or extract the error detection checkwords and

ancillary data packets. A special type of video decoder, the flywheel video decoder, is often used in video processors because it provides immunity from noisy, error prone, or briefly interrupted input video streams.

Digital Video Standards

There are many different video standards, both analog and digital. Today, most broadcast studios and video production centers use component digital video when creating, storing, and transporting video. Component digital video can be readily compressed using digital video compression standards. It can also be encoded into analog composite video for broadcast.

SDI supports a variety of standard-definition digital video standards. The ANSI/SMPTE 259M-1997 SDI [1] standard defines how to serially transport the digital standards listed in Table 1. The documents listed describe the parallel form of these video standards, and the SDI standard describes how to convert the parallel video data to a serial format.

Table 1: **SDI Supported Digital Video Standards**

Standard	Description	Serial Bit Rate
SMPTE 125M[2] & ITU-R BT.601-5[3]	NTSC & PAL 4x3 aspect ratio 4:2:2 component digital video	270 Mb/s
SMPTE 267M[4]	NTSC 16x9 aspect ratio 4:2:2 component digital video	360 Mb/s
SMPTE 244M[5]	NTSC Composite Digital Video	143 Mb/s
IEC 61179 [6] & EBU Tech. 3280-E [7]	PAL Composite Digital Video	177 Mb/s

SDI is compatible with both component and composite digital video. Component digital video is very commonly used in the broadcast industry, but composite digital video is not as common. The reference designs accompanying this application note support only component digital video.

Since the original SDI specification was introduced, it has been adapted to accommodate some additional digital video standards. Among these are NTSC and PAL 4:4:4:4 component digital video standards. The parallel format for the NTSC and PAL 4:4:4:4 video standards are described in SMPTE RP 174-1993 [8] and ITU-R BT.799-3 [9]. The serial data rate for both 4:4:4:4 digital component video standards is 540 Mb/sec. The SDI specification was extended to cover the 540 Mb/sec data rate by SMPTE 344M-2000 [10].

Xilinx application notes **XAPP248** and **XAPP286** contain descriptions of the 4 x 3 aspect ratio, 4:2:2 component digital video standards. The NTSC and PAL 16x9 aspect ratio 4:2:2 component digital video standards are very similar to the 4 x 3 aspect ratio standards and simply have more samples per line. The 4:4:4:4 standards differ more significantly from the 4:2:2 standards, as described below.

The 4:2:2 digital video standards use two data words per video sample. Each video sample contains a luma word (Y) and one chroma word. Consecutive video samples alternate between containing a blue color-difference chroma word (Cb) and a red color-difference chroma word (Cr).

In contrast, the 4:4:4:4 digital video standards have four data words per video sample. These standards support either the YCbCr color space or the RGB color space. If YCbCr is used, each video sample contains words for the Y, Cb, and Cr components plus a fourth auxiliary component designated as A. If the RGB color space is used, each video sample contains a word for each red, green, blue, and A component. The A component typically carries the key channel, indicating the transparency of the sample. In the RGB color space, this key channel is often called the alpha channel.

The format of the XYZ word of the timing reference signal (TRS) symbol is different for the 4:4:4:4 standards. A flag bit called S has been added to indicate the color space used in the

video stream. The S bit is low for RGB and high for YCbCr. The format of the XYZ word for the 4:4:4:4 TRS symbol is shown in Table 2.

Table 2: **XYZ Word Format for the 4:4:4:4 TRS Symbol**

Bit	9	8	7	6	5	4	3	2	1	0
TRS Symbol	1	F	V	H	S	P4	P3	P2	P1	0

The bits labeled P4 through P1 are protection bits calculated in the following manner:

$P4 = V \oplus V \oplus H$

$P3 = F \oplus V \oplus S$

$P2 = V \oplus H \oplus S$

$P1 = F \oplus H \oplus S$

Video Standard Detection

A video processor can be designed to support several different standards of digital video. To properly process the video stream, the processor must first determine the video standard of the input video stream.

Until very recently, digital video streams did not carry any explicit identification information to indicate the video standard. In 2001, the SMPTE 352M [11] standard was released. This standard describes a standard ancillary data packet carrying "video payload" identification information. Once widely adopted, this standard will simplify the process of detecting the video standard of a digital video stream. The video processor will simply be able to look for and decode the identification information in the ancillary data packet.

For now, however, more traditional methods of video standard detection must still be used. The video processor must determine the video standard by examining the timing of the video stream. All digital video standards supported by SDI contain TRS symbols. These symbols occur in the video stream whenever the video timing signals change. There are three timing signals in the TRS symbol called F, V, and H. The F bit indicates the current field (odd or even). The V bit is asserted during the vertical blanking interval of each field. The H bit is asserted during the horizontal blanking interval of each line.

Each of the six component digital video standards supported by this application note contains a different number of data words on a line of video. A video standard detector finds the TRS symbols marking the beginning of each video line and counts the number of words between those symbols, then compares the results against the known video standards. Table 3 shows the number of samples on a line of video for each of the video standards supported by this application note.

Table 3: **Words Per Video Line**

NTSC/PAL	Sampling Scheme	Aspect Ratio	Words Per Line
NTSC	4:2:2	4:3	1716
NTSC	4:2:2	16:9	2288
NTSC	4:4:4:4	4:3	3432
PAL	4:2:2	4:3	1728
PAL	4:2:2	16:9	2304
PAL	4:4:4:4	4:3	3456

Most video standard detectors require some number of consecutive lines to contain the same number of samples before reporting that the video standard has been detected. This same function can be used to provide noise immunity by preventing the video standard detector from

inadvertently switching to a new video standard when receiving a few lines containing the incorrect number of words, possibly caused by noise in the video stream.

If the video processor supports composite digital video, then it must determine whether the video stream is composite or component digital video. This can be done by examining the fourth word of any TRS symbol in the video stream. For composite digital video, all the bits of this word are zero. For component digital video, the most significant bit (bit 9) of the fourth word is always a one. Therefore, by simply looking at bit 9 of the fourth word of any TRS symbol, a video standard detector can determine whether the video stream contains composite or component digital video.

This application note does not support composite digital video. The video standard detector in the reference design determines the type of video stream, and simply stays "unlocked" when it finds composite video.

Flywheel Video Decoder

Basic Video Decoding

Inserting and extracting information, such as ancillary data packets, requires the video processor to know exactly the current line number and horizontal position of the sample being received and processed. The primary purpose of the video decoder is to synchronize to the incoming video stream and provide the current horizontal and vertical position to other modules in the video processor.

To find the current horizontal position, a video decoder watches for a start of active video (SAV) TRS symbol. The sample immediately after the last word of the SAV symbol is the first sample in the active portion of the line (sample 0).

To find the current vertical position, the video decoder must watch for a transition of the F bit, indicating the beginning of a new field. When a field transition occurs, the video decoder can determine the current line number if it also knows the video standard. Table 4 shows the starting line numbers of each field for both NTSC and PAL video.

Table 4: **Field Starting Line Numbers**

NTSC/PAL	Odd Field Starting Line Number (F = 0)	Even Field Starting Line Number (F = 1)
NTSC	4	266
PAL	1	313

Using a Flywheel for Noise Immunity

A flywheel video decoder is often used in video processors to provide immunity from noise and interruptions in the input video stream. Flywheel video decoders use the same techniques just described to synchronize to the input video stream. Once synchronized, however, the flywheel generates its own video timing for the video stream. It continuously compares its internally generated video timing information with the input video stream. When a difference occurs, the flywheel decoder does not immediately resynchronize with the input video stream. Instead, it continues to generate video timing unchanged, as if it had momentum carrying it forward. If the input video stream contains only a few corrupted data words, it usually resumes in sync with the flywheel. However, if synchronization is lost because the video stream was switched to a different source or standard, the flywheel eventually determines that it must resynchronize with the incoming video stream.

Not only does the flywheel decoder provide noise immunity for the video decoder function, but it also can provide several other valuable functions.

The video timing information produced by the flywheel decoder can be used to repair damaged or invalid TRS symbols in the input video stream. Because the flywheel is generating video timing information, the video processor can generate correct TRS symbols. These can be

inserted in place of the TRS symbols in the input video stream, thereby, repairing any TRS symbols that have been corrupted by noise.

The flywheel decoder continues to generate and insert TRS symbols into the video stream even when the input video stream is interrupted. Obviously, the visual information in the resulting video stream is invalid. But, the timing information contained in the TRS symbols is valid. This is useful because it keeps all downstream video equipment synchronized.

Synchronous Switching Considerations

SDI video streams are often sent through video routers. In most cases, broadcast studios take care to insure synchronization of the various video streams into the router. The input video streams usually are synchronized to the same video line. However, the video streams are not always synchronized to precisely the same horizontal position on the line.

When a router switches between these synchronous video sources, the receiving equipment sometimes detects some small horizontal offset of the EAV symbol on the line where the switch occurs. Normally, a flywheel decoder would ignore this EAV offset until it detected the offset occurring repeatedly over some number of consecutive lines. Only then would the flywheel resynchronize. However, when switching between closely synchronized video sources, it is better for the flywheel decoder to instantly resynchronize.

SMPTE recommended practice RP 168-1993 defines one line per field when synchronous switching is allowed to occur.[12] Table 5 shows the synchronous switching lines for both fields of both the NTSC and PAL video standards. These lines were carefully chosen to minimize disturbances to timing and other vital data. Other digital video standards forbid the placement of critical information on these synchronous switching lines, since these lines are subject to corruption during the switch.

Table 5: **RP 168 Synchronous Switching Line Numbers**

NTSC/PAL	Odd Field Synchronous Switching Line Number	Even Field Synchronous Switching Line Number
NTSC	10	273
PAL	6	319

According to RP 168, a synchronous switch must occur only during a window of a few hundred samples located in about the middle of the active portion of the synchronous switching line. This insures that the switch occurs well after the SAV symbol and well before the EAV symbol, thus minimizing the chance that these important video timing signals will be corrupted by the switch.

A video flywheel decoder should accommodate horizontal offsets that occur on these synchronous switching lines and immediately resynchronize to the incoming video stream, if such an offset is detected. If a vertical offset or field difference is detected on a synchronous switching line, the switch is asynchronous and the flywheel should implement its normal resynchronization process.

Tolerating an Early Falling Transition of the V Bit

The current standards for NTSC component digital video, SMPTE 125M-1995 and ITU-R BT.601-5, require the V bit to transition from a High to a Low on lines 20 and 283, marking the end of the vertical blanking interval. Earlier versions of the NTSC component digital video specifications, however, allowed the V bit to fall Low on any line from 10 to 20 for the odd field and 273 to 283 for the even field.

The current specifications recommend tolerance of early V bit transitions to allow for compatibility with video equipment designed to earlier versions of the specifications. Since the video flywheel decoder is generating its own version of the V bit, it may detect a discrepancy in the V bit on video streams generated by older equipment. The flywheel decoder should tolerate

these discrepancies, but only on those lines where the V bit was permitted to transition early in the previous standards.

PAL component digital video specifications have always precisely specified the line when the V bit should transition, so this does not apply to the PAL standards.

Reference Design

The reference design contains a top-level module called *video_decode*. This module is a wrapper around three modules: the video standard detector (*autodetect*), the flywheel video decoder (*flywheel*), and a preprocessor module that is used to examine the video stream for TRS symbols and other special patterns (*trs_detect*). Figure 2 is a block diagram of the *video_decode* module.

There may be some cases where the *autodetect* module is not required. If an application always processes the same video standard, or if the video standard is provided some other way, by a front panel selector switch for example, the autodetect function can be eliminated from the design. In these cases, the `std_in` inputs to the *flywheel* module should be hardwired to the video standard or controlled by some external function. The `std_locked` signal should always be asserted. The `rx_xyz_err` input of the *flywheel* module should be connected to either the `rx_xyz_err` or the `rx_xyz_err_4444` output of the *trs_detect* module, depending on the video standard. Finally, the `s4444` input of the *flywheel* module should be correctly controlled if one of the 4:4:4:4 formats is selected.

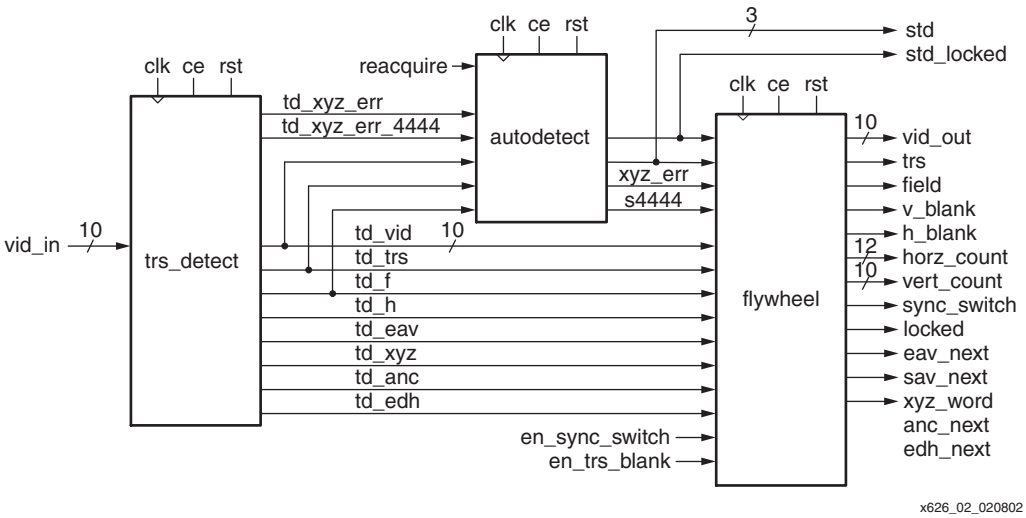

Figure 2: **Video Decoder Block Diagram**

The *video_decode* module delays the video stream by six clock cycles. Figure 3 shows the timing relationships between many of the output signals of the *video_decode* module. The diagram shows an EAV symbol followed immediately by the first part of an ANC packet. If the ANC packet were an EDH packet, then the `edh_next` signal would be asserted at the same time as the `anc_next` signal. The diagram ends with an SAV symbol.

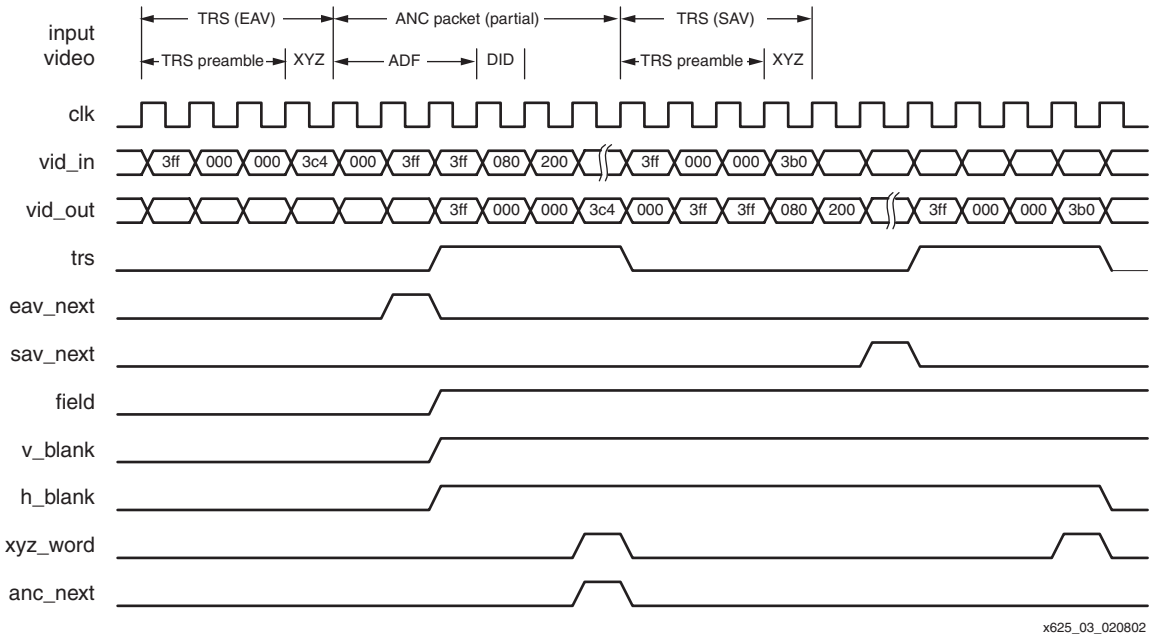

Figure 3: Timing Diagram for `video_decode`

Trs_detect Module

The *trs_detect* module performs some preliminary parsing of the video stream, looking for certain special word patterns. This module has a four clock deep register pipeline that delays the video while the module examines it. Figure 4 is a block diagram of the *trs_detect* module.

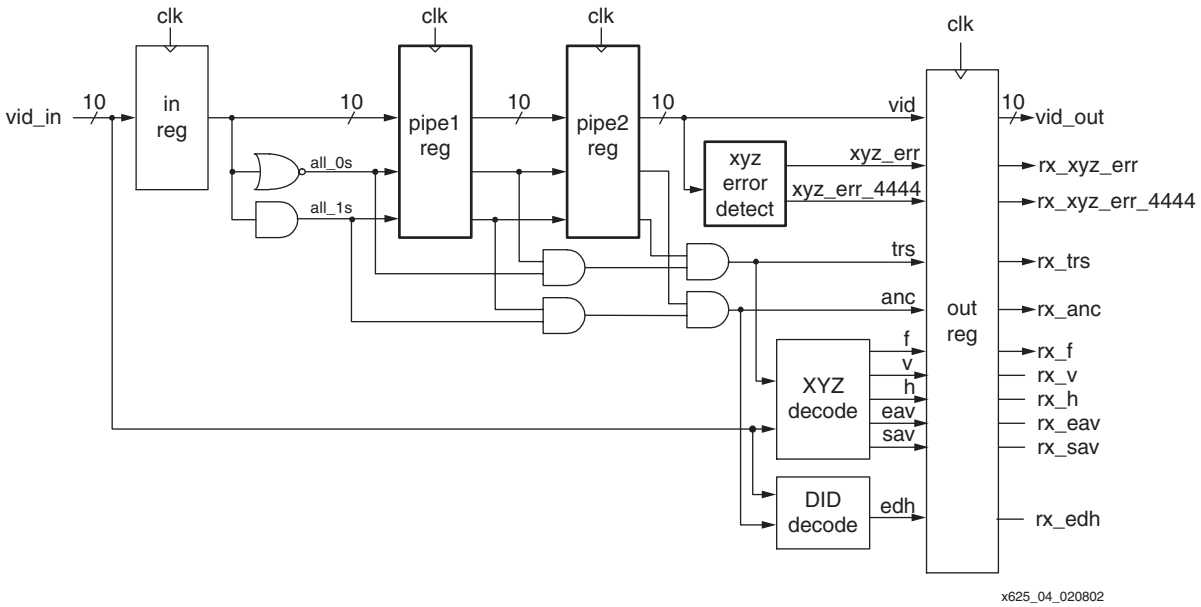

Figure 4: ***trs_detect* Block Diagram**

The *trs_detect* module performs the following functions:

- The *trs_detect* module detects TRS symbols occurring in the video stream. TRS symbols are four words long. The first three words are a pattern unique in the video stream: `3FFh`, `000h`, and `000h`. When this pattern is detected, the *trs_detect* module asserts the `rx_trs` signal while it outputs the first word of the TRS symbol.
- If a TRS symbol is detected, the *trs_detect* module decodes the fourth word of the TRS

symbol, called the XYZ word. It asserts the `rx_xyz` signal as it outputs the XYZ word. It also asserts either the `rx_eav` or `rx_sav` signals, depending on whether the TRS symbol is an EAV or SAV. The `rx_eav` and `rx_sav` signals are only asserted when the first word of a TRS symbol is output from the *trs_detect* module (when `rx_trs` is asserted). These signals provide a look-ahead function for the video processor, indicating whether the TRS symbol is an EAV or SAV before the XYZ word of the TRS symbol appears in the output video stream from the *trs_detect* module.

- The *trs_detect* module checks the protection bits of the XYZ word to determine if it contains an error. The module asserts the `rx_xyz_err` signal if the XYZ word contains an error. This signal is only valid if the video standard is one of the 4:2:2 standards. A different error signal, rx_xyz_err_4444 indicates the detection of an error in the XYZ word for the 4:4:4:4 video standards. Because the *trs_detect* module does not know which video standard is being received, it always examines the XYZ word for errors in both formats. These two error signals are only valid when the `rx_xyz` signal is asserted.
- The *trs_detect* module latches the F, V, and H bits from the TRS symbol's XYZ word. These latched bits are output from the *trs_detect* module as the `rx_f`, `rx_v`, and `rx_h` signals. These signals remain valid until the next TRS symbol is detected. These signals always transition at the beginning of the TRS symbol.
- The *trs_detect* module detects ancillary data (ANC) packets. An ANC packet begins with a three-word ancillary data flag (ADF). Similar to the TRS symbol, the ADF is unique in the video stream. The three words of the ADF are `000h`, `3FFh`, and `3FFh`. When an ADF is detected, the *trs_detect* module asserts the `rx_anc` signal during the first word of the ADF.
- When an ADF is detected, the *trs_detect* module examines the word immediately after the ADF to determine if the ANC packet is an error detection and handling (EDH) packet. The word immediately following the ADF in an ANC packet is called the Data Identification word (DID) identifying the type of packet. EDH packets have a DID value of `1F4h`. If an EDH packet is found, the *trs_detect* module asserts the `rx_edh` signal during the first ADF word of the packet (when `rx_anc` is asserted).

Figure 5 shows a timing diagram of the inputs and outputs of the *trs_detect* module. It shows how the input video stream is delayed by four clock cycles before coming out of the module. An EAV TRS symbol is shown going into the *trs_detect* module. An ancillary data packet immediately follows the TRS symbol. The signals in parenthesis have the same timing as the signals listed above them.

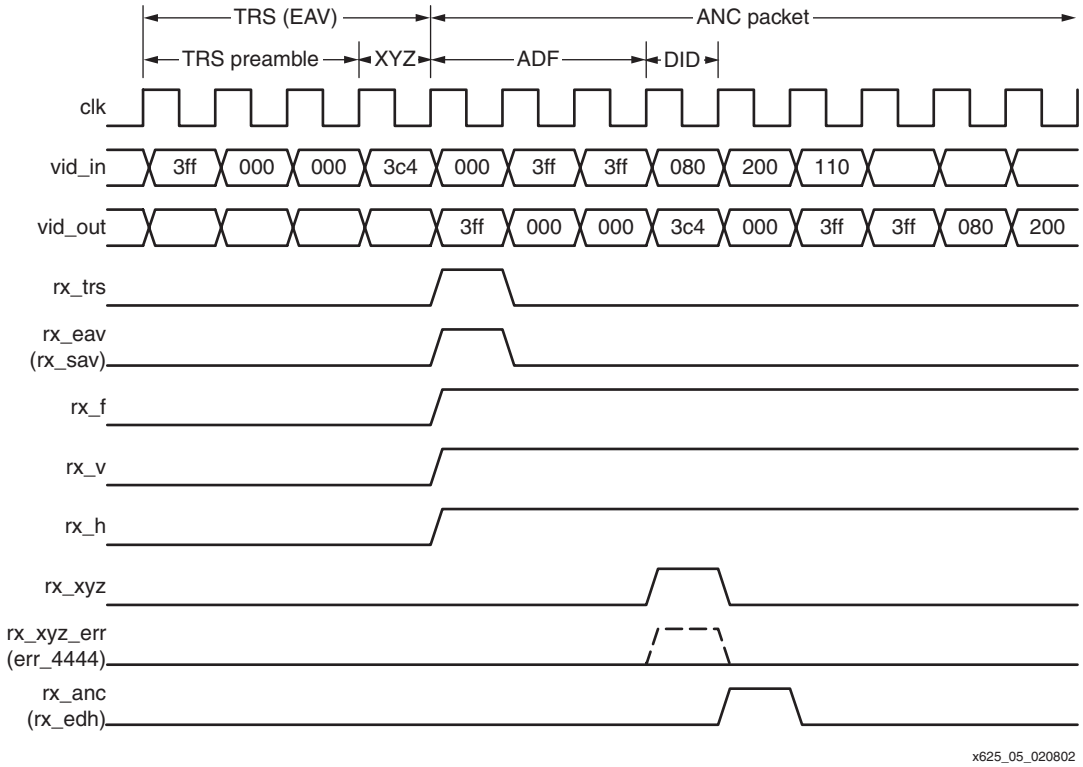

Figure 5: ***trs_detect*** **Module Timing**

When the *trs_detect* module is looking for TRS symbols and ANC packets, it only examines the eight most significant bits of the video word to determine if the word contains all zeros or all ones. This is to provide compatibility with 8-bit video equipment. The digital video standards state that, when checking for TRS and ADF words, the least two significant bits should be ignored.

Autodetect Module

The *autodetect* module implements a video standard detector. This module examines the input video stream and decoded information from the *trs_detect* module and determines the video standard. Figure 6 is a block diagram of the *autodetect* module.

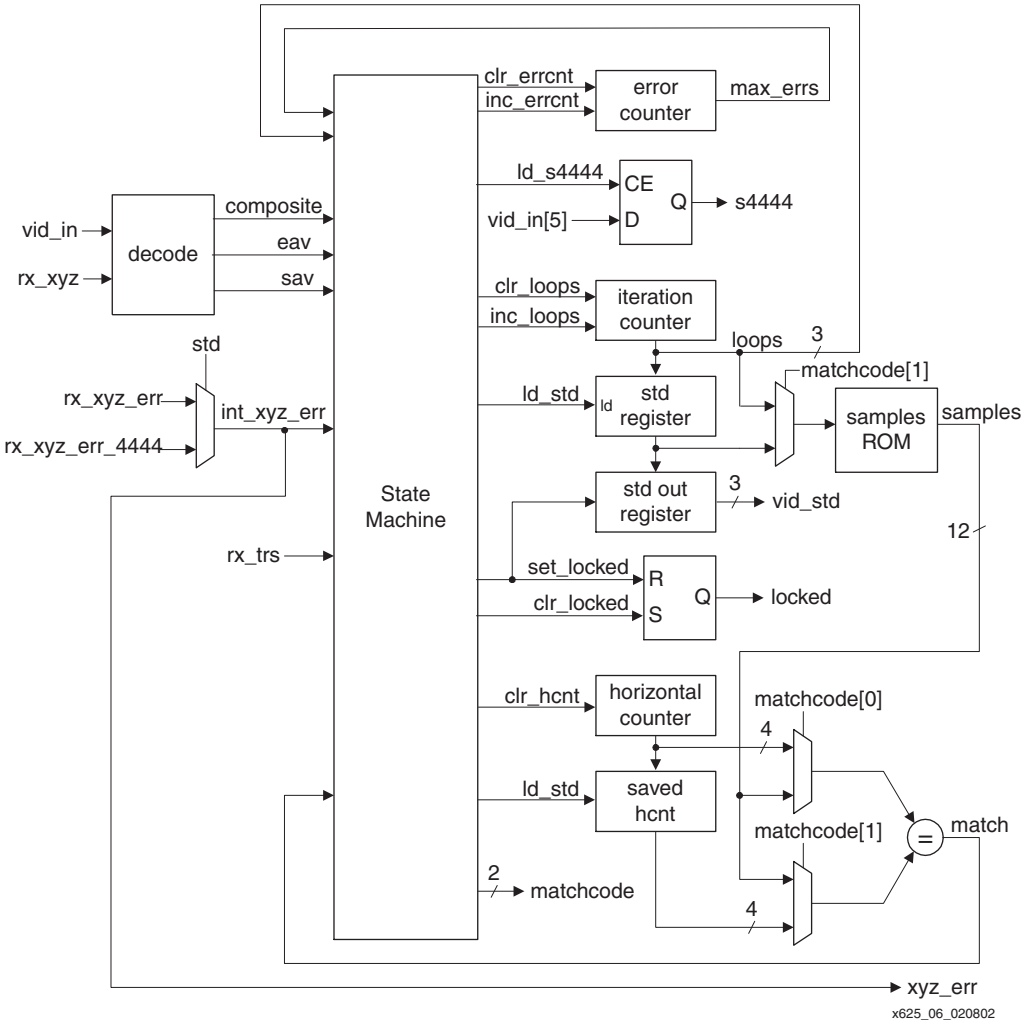

Figure 6: ***autodetect* Block Diagram**

The *autodetect* module is based on a finite state machine (FSM). The FSM has two main loops, the ACQUIRE loop shown in Figure 7 and the LOCKED loop shown in Figure 8. The ACQUIRE loop tries to match the input video stream with one of the standards supported by the module. Once the video standard is determined, the FSM enters the LOCKED loop where it continuously monitors the input video stream for a change in the video standard.

To determine the video standard, the FSM determines the number of words between SAV symbols. Each video standard supported by the module has a unique number of words per video line. When the first SAV is detected, the horizontal counter is cleared (state ACQ3). The horizontal counter increments every clock cycle, counting the number of words on the video line. When the next SAV symbol is found, the horizontal counter value is captured in the `saved_hcnt` register (state ACQ4). The value in the `saved_hcnt` register is compared to the horizontal position of the six subsequent SAV symbols (state ACQ7). If the SAV occurs at the same horizontal position on each of these lines, then the FSM assumes that the video is stable and it has found the correct number of words per line. If the number of words on a line varies from the value in the `saved_hcnt` register during the acquisition process, the entire process is restarted.

After finding the number of words per line in the input video stream, the FSM compares this word count with the word counts of each of the supported video standards. The Samples ROM contains the word counts for each supported video standard. In state ACQ6, the FSM cycles through each entry in the ROM by incrementing the iteration counter and comparing the output of the ROM with the value in the `saved_hcnt` register. If a match is found, the value of the iteration counter is captured in the `std` register and is used as the output video standard code (`vid_std`). If no match is found after searching all the entries in the ROM, the FSM restarts the acquisition process.

After the FSM has acquired the video standard, it moves to the LOCKED loop. In this loop, the FSM determines if the number of words on each video line in the input video stream is correct for the current video standard. It also checks for errors in XYZ words of the TRS symbols. If an incorrect number of words is found on a line or an XYZ word error occurs, an error counter increments. When the number of consecutive video lines with errors exceeds the MAX_ERRS value, the FSM returns to the ACQUIRE loop to reacquire the standard.

By requiring that errors occur on some number of consecutive lines before beginning to reacquire the video standard, the FSM provides some noise immunity for the video standard detection function. It also, however, increases the amount of time required for a new video standard to be detected.

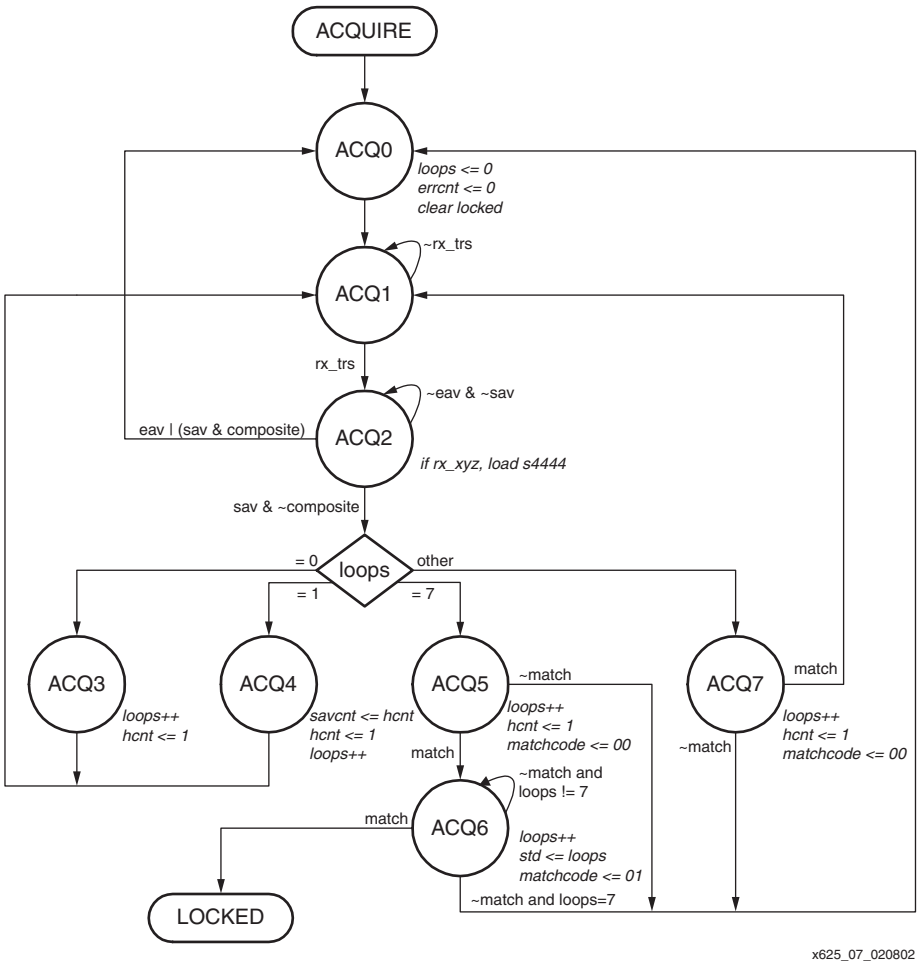

Figure 7: ***autodetect*** **FSM ACQUIRE Loop**

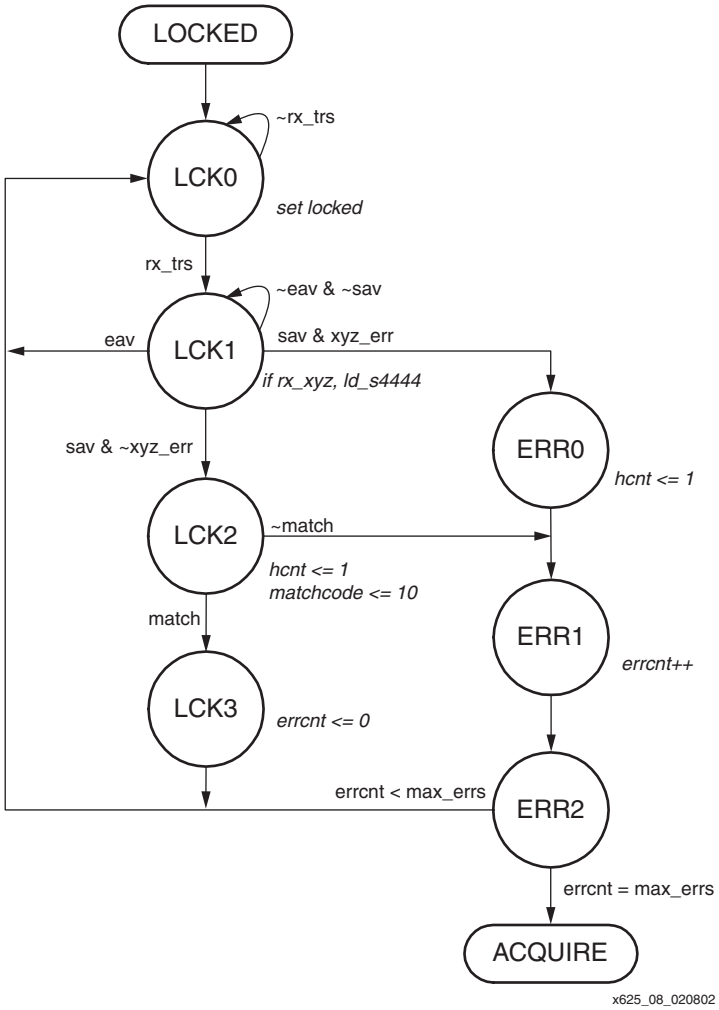

Figure 8: ***autodetect*** **FSM LOCKED Loop**

Flywheel Module

The *flywheel* module implements a flywheel video decoder. This module provides video timing in the presence of noisy, error prone, or interrupted input video data. After the *autodetect* module has determined the video standard of the input video stream, the flywheel synchronizes to the input video stream. Once synchronized, the flywheel generates video timing that should correspond to the timing of the input video stream. If the flywheel detects mismatches between the input video stream and its internally generated video timing on four lines over a window of eight video lines, it will begin to resynchronize. The flywheel provides noise immunity by requiring a significant number of errors to occur before resynchronizing.

The flywheel generates and inserts TRS symbols into the video stream, overwriting the data in the input video stream where the TRS symbols occur. This will repair any damaged or erroneous TRS symbols appearing in the input video stream. However, this can cause multiple copies of a TRS to appear in the resulting video stream if the TRS in the input video stream does not occur at the same time as the flywheel generated TRS. To prevent this, the flywheel implements TRS blanking. The flywheel generates black-level video values and inserts them in place of a TRS in the input video stream, if the TRS does not occur at the same time as the TRS generated by the flywheel.

The flywheel reference design takes less than two fields to synchronize to a new input video stream. Because the flywheel must look for the start of a new field to synchronize vertically, the actual time to synchronize is anywhere from a few lines to a little over one field, depending on where the first field transition occurs in the input video stream.

Figure 9 is a block diagram of the *flywheel* module. The flywheel contains four modules implementing the state machine (*fly_fsm*), the field functions (*fly_field*), the vertical functions (*fly_vert*), and the horizontal functions (*fly_horz*).

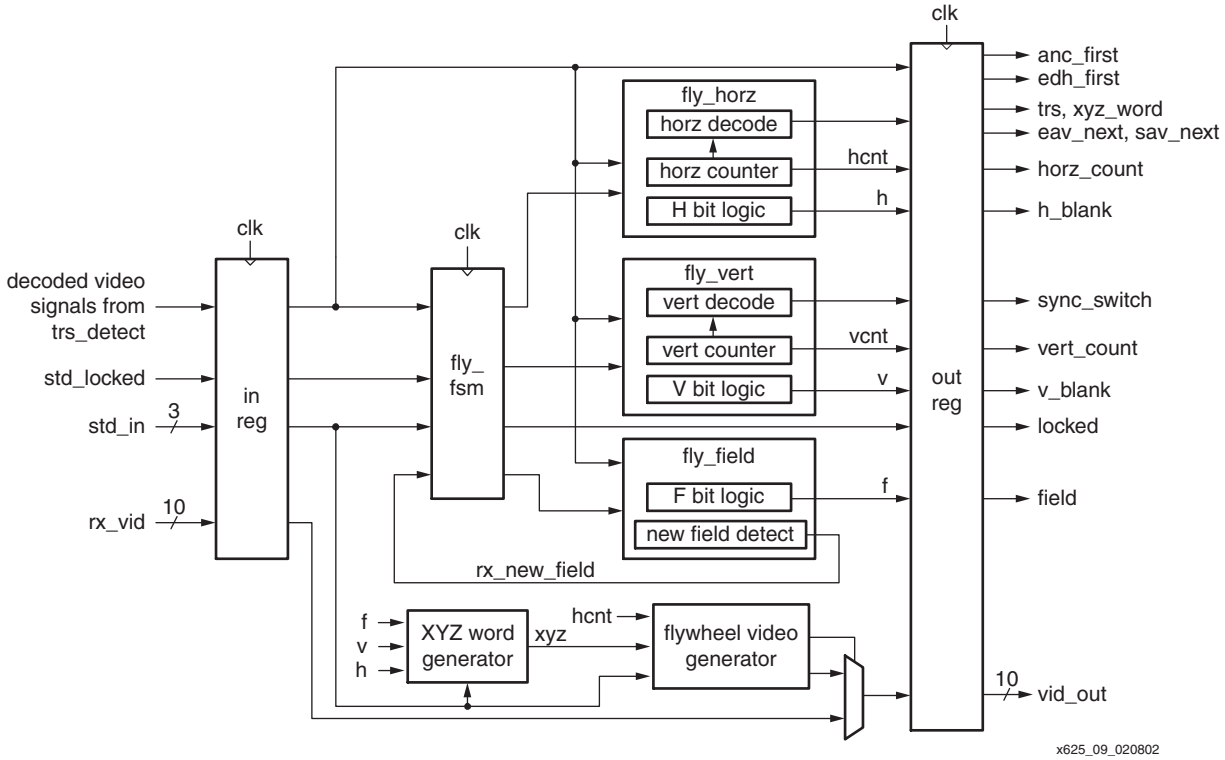

Figure 9: ***flywheel* Block Diagram**

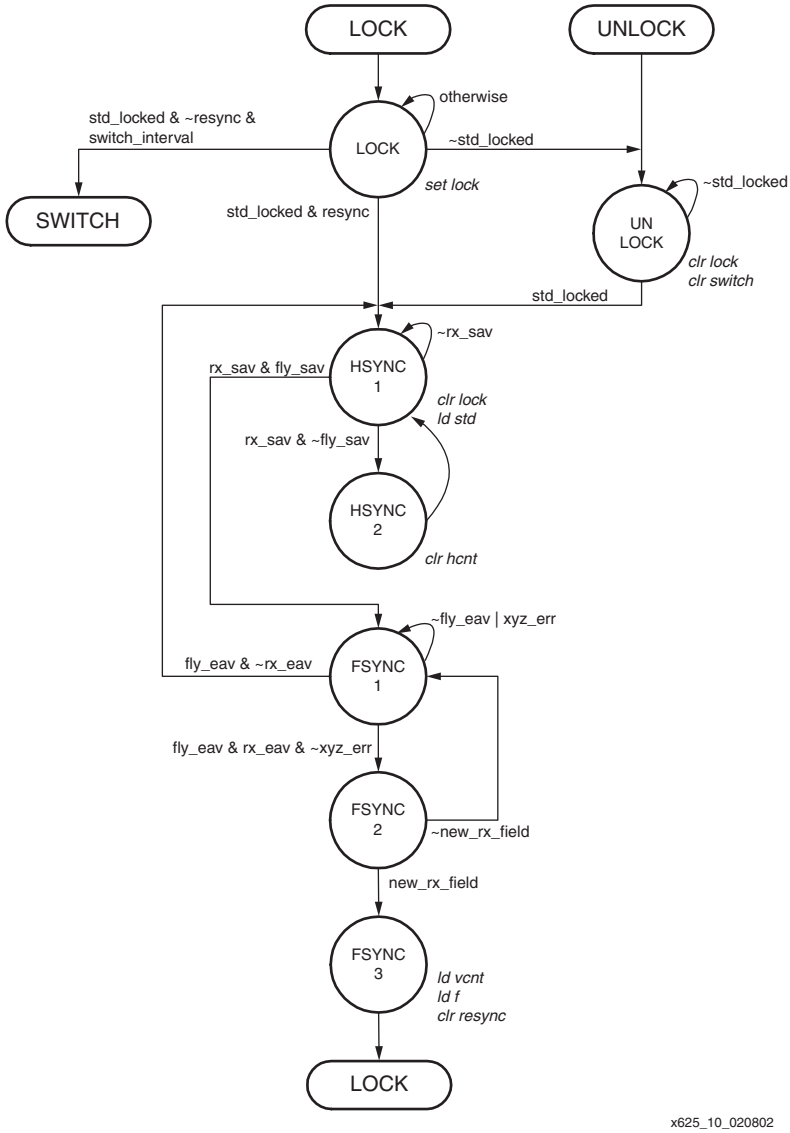

Figure 10: ***flywheel*** **FSM State Diagram Main Loop**

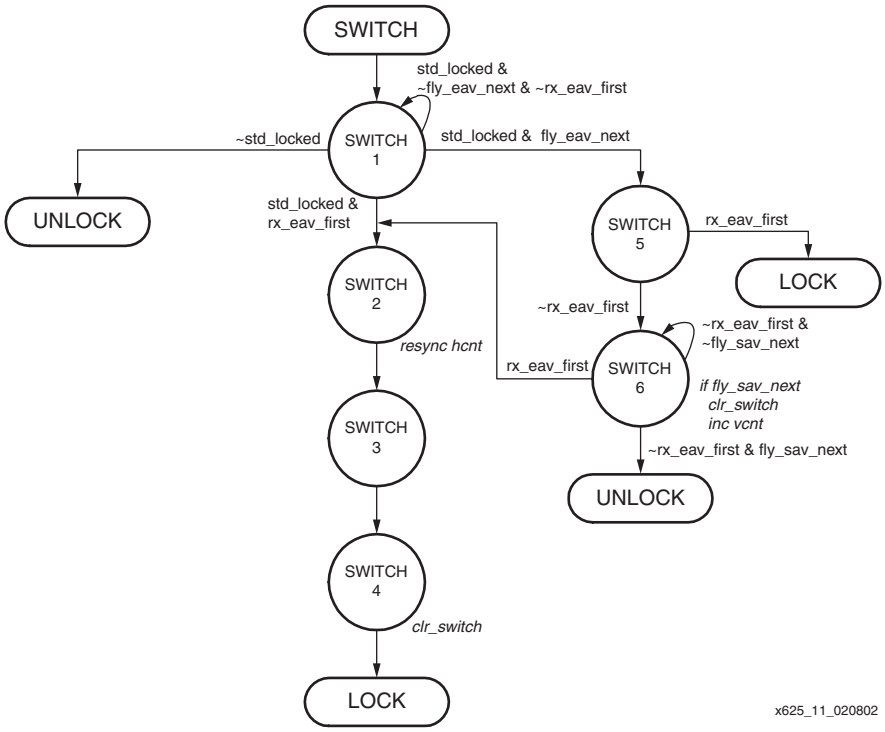

Figure 11: ***flywheel*** **FSM State Diagram Synchronous Switching**

The FSM of the flywheel begins in the UNLOCK state. It remains in this state until the *autodetect* module signals detection of the input video standard by asserting the `std_locked` signal. The FSM then attempts to synchronize to the input video stream.

To synchronize to the input video stream, the FSM first synchronizes horizontally by looking for SAV symbols in the input video stream. The FSM resets the horizontal counter located in the *fly_horz* module whenever an SAV is received. This is repeated until the flywheel's generated SAV occurs at the same time as the SAV in the input video stream. When the positions of the SAV symbols match, the FSM has synchronized the horizontal counter to the input video stream.

Next, the FSM attempts to synchronize vertically by waiting for a transition of the field bit (F) in the input video stream. The *fly_field* module contains field transition detection logic. This logic captures the F bit from every EAV in the input video stream and compares it to the F bit from the previous EAV. When the F bit changes, the start of a new field has been found and the *fly_field* module asserts the `new_rx_field` signal, causing the state machine to load the vertical counter. The value loaded into the vertical counter is determined by the current video standard and by the value of the F bit.

After the vertical counter has been loaded, the flywheel is synchronized to the input video stream. The FSM moves to the LOCK state and asserts the locked output. Once locked, the FSM remains locked until it detects differences between its internally generated TRS symbols and the TRS symbols in the input video stream on four lines over a rolling eight-line window. When too many errors are detected, the FSM negates the locked signal and repeats the synchronization process.

Once per field, on the synchronous switching line, the FSM moves to the SWITCH1 state to check for a synchronous switch. In this state, the FSM determines if the EAV in the input video stream and the internally generated EAV occur at the same time. If they do not, the FSM reloads the horizontal counter to match the position of the EAV symbol in the input video stream.

The FSM contains a fail-safe mechanism to allow it to continue to generate valid video timing even if the input video stream does not contain an EAV during the synchronous switching line.

Without this fail-safe mechanism, the FSM would become stuck waiting for an EAV in the input video stream. The fail-safe mechanism is found in state SWITCH6. In this state, the FSM has already determined that the input EAV is overdue. If no EAV is received before it is time to generate an SAV, the FSM gives up on the synchronous switch and proceeds to the UNLOCK state. This is considered to be a failed synchronous switching event.

During the synchronous switching lines, the flywheel is designed to pass the EAV from the input video stream directly through to the output video stream. In the case of a failed synchronous switch, the input video stream does not contain an EAV. In this case, there will not be an EAV symbol in the output video on the failed synchronous switching line.

The flywheel module generates an output signal called `sync_switch`. This signal is asserted when the current video line contains the synchronous switching point. In an SDI receiver design, this signal should be used to disable TRS filtering in the SDI receiver's framer function. TRS filtering generally forces the framer to receive at least two consecutive TRS symbols at a new bit offset in the serial data stream before reframing to this new offset. During the synchronous switching interval, the framer should immediately reframe if an offset is detected in the TRS symbol.

When the FSM moves to the UNLOCK state due to losing synchronization with the input video stream or because of a failed synchronous switching event, the flywheel continues to generate valid TRS symbols based on its internal timing until it regains synchronization.

The flywheel module is designed to tolerate early V-bit transitions. On those lines when the V bit is permitted to be either High or Low, the flywheel ignores the V bit when comparing its internally generated TRS symbols with the TRS symbols in the input video stream. This only applies to the NTSC standards. For all PAL video standards, the V bit is always checked.

Testbench

The `test_vid_decode.v` and `test_vid_decode.vhd` files each contain a testbench for the video decoder. The testbench contains an instance of the *video_decode* module and video generator module called *multigen*.

The *multigen* module generates video for each of the six digital component video standards supported by the video decoder. It also supports the option of generating an early transition on the V bit for the NTSC standards.

The video generated by the *multigen* module is connected to the input of the video decoder. It is also delayed by an amount equal to the latency of the video decoder and then compared with the output of the video decoder to detect any differences. The comparison is only done when the video decoder indicates that it is locked to the video standard. The comparison code also ignores legal early transitions of the V bit and differences during the synchronous switching interval.

The testbench cycles through all six supported video standards, determining if the video decoder recognizes and locks to each of them. During the NTSC 4:2:2 test, the testbench also performs tests of various other features of the video decoder: tolerance of early V-bit transitions, TRS blanking, synchronous switching, and generation of TRS symbols when the input video stream is interrupted. The code in the testbench that tests these features can be moved to allow testing of these features during any of the video standards.

Reference Design Results

Table 6 shows the results after place and route of the various modules implemented in this application note. Results are given for the top-level *video_decode* module and individually for the three main blocks that make up the video decoder. All results were obtained using the Verilog versions of the designs with Xilinx ISE version 4.1i using XST as the synthesis tool. Results using the VHDL files are not shown but are essentially identical. Virtex-II results are for a -5 speed grade device. Spartan II results are for a -6 speed grade device. The reference design files are available on the Xilinx FTP site at:

ftp://ftp.xilinx.com/pub/applications/xapp/xapp625.zip

The video decoder runs at the word rate of the video interface. The highest word rate required for the six supported video standards is 54 MHz. As shown in Table 6, this is easily achievable with Xilinx FPGAs.

Table 6: **Reference Design Results**

	Optimized for Area			Optimized for Speed		
Design Name	**Size LUTs/FFs**	**Speed: Virtex-II**	**Speed: Spartan II**	**Size LUTs/FFs**	**Speed: Virtex-II**	**Speed: Spartan II**
trs_detect.v	30/60	140 MHz	110 MHz	35/56	200 MHz	120 MHz
autodetect.v	124/55	95 MHz	60 MHz	122/55	110 MHz	70 MHz
flywheel.v	211/127	75 MHz	60 MHz	215/131	100 MHz	80 MHz
video_decode.v	363/235	75 Mhz	60 MHz	372/237	100 MHz	70 MHz

Conclusions

In an SDI transmission link, digital video is normally preprocessed prior to transmission to insert error detection checkwords and other types of ancillary data. At the receiving end of the SDI link, the data is again processed to check for transmission errors and possibly to extract the ancillary data. In order to carry out these functions, a digital video processor must be synchronized to the video stream, to determine the fixed locations of these types of data packets. This is the job of the video decoder. A flywheel video decoder adds noise immunity features to the basic video decoder. Most video processors also include a video standard detector, to automatically determine the standard of the input video stream.

These two functions can be used for more than just SDI-related video processing. Most video processing features require the two things provided by these functions: the standard of the input video stream and the current horizontal and vertical position of the input video stream.

References

1. ANSI/SMPTE 259M-1997, SMPTE Standard for Television - 10-Bit 4:2:2 Component and 4fsc Composite Digital Signals - Serial Digital Interface (The Society of Motion Picture standards and recommended practices referenced in this application note can be purchased at the SMPTE web site: **http://www.smpte.org**.
2. ANSI/SMPTE 125M-1995, SMPTE Standard for Television - Component Video Signal 4:2:2 - Bit-Parallel Digital Interface.
3. ITU-R BT.601-5, Studio Encoding Parameters of Digital Television for Standard 4:3 and Wide-Screen 16:9 Aspect Ratios (International Telecommunication Union). The ITU-R standards referenced in this application note can be purchased from the International Telecommunication Union at: **http://www.itu.int/itudoc/itu-r/rec/bt/**.
4. ANSI/SMPTE 267m-1995, SMPTE Standard for Television - Bit-Parallel Digital Interface - Component Video Signal 4:2:2 16 x 9 Aspect Ratio.
5. ANSI/SMPTE 244M-1995, SMPTE Standard for Television - System M/NTSC Composite Video Signals - Bit-Parallel Digital Interface.
6. IEC 61179, Helical-scan digital composite video cassette recording system using 19mm magnetic tape, format D2 (International Electrotechnical Commission). This standard can be purchased from the International Electrotechnical Commission at: **http://www.iec.ch/webstore**.
7. EBU Tech. 3280-E, Specification of interfaces for 625-line digital PAL signals (European Broadcasting Union). This standard can be downloaded from the EBU web site at **http://www.ebu.ch**.
8. RP 174-1993, SMPTE Recommended Practice - Bit Parallel Digital Interface for 4:4:4:4 Component Video Signal (Single Link).

9. SMPTE 344M-2000, SMPTE Standard for Television - 540 Mb/s Serial Digital Interface.
10. ITU-R BT.799-3, Interfaces for Digital Component Video Signals in 525-Line and 625-Line Television Systems Operating at the 4:4:4 Level of Recommendation ITU-R BT.601 (Part A).
11. SMPTE 352M-2001 SMPTE Standard for Television - Video Payload Identification for Digital Interfaces.
12. RP 168-1993, SMPTE Recommended Practice - Definition of Vertical Interval Switching Point for Synchronous Video Switching.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/12/02	1.0	Initial Xilinx release