

Introduction

This application note describes the Serial Digital Interface (SDI) Flywheel Video Decoder reference design based on the current Altera® SDI MegaCore® function. The SDI flywheel video decoder is used in video processors to handle real video signal streaming without noise or interruptions. The SDI flywheel video decoder synchronizes its internally-generated video timing to the incoming video stream, and provides the horizontal and vertical timings to the other blocks in the video processor. After synchronizing, the SDI flywheel video decoder continuously compares its internal timing information with the input video stream. Whenever a difference occurs, the SDI flywheel video decoder does not immediately resynchronize with the input video stream, but instead, continues to generate video timing. The internal timing usually resumes in sync with the SDI flywheel video decoder if the input video stream contains only a few corrupted data words. If the video stream is switched to a different source or standard causing the synchronization to go off, the SDI flywheel video decoder resynchronizes with the incoming video stream.

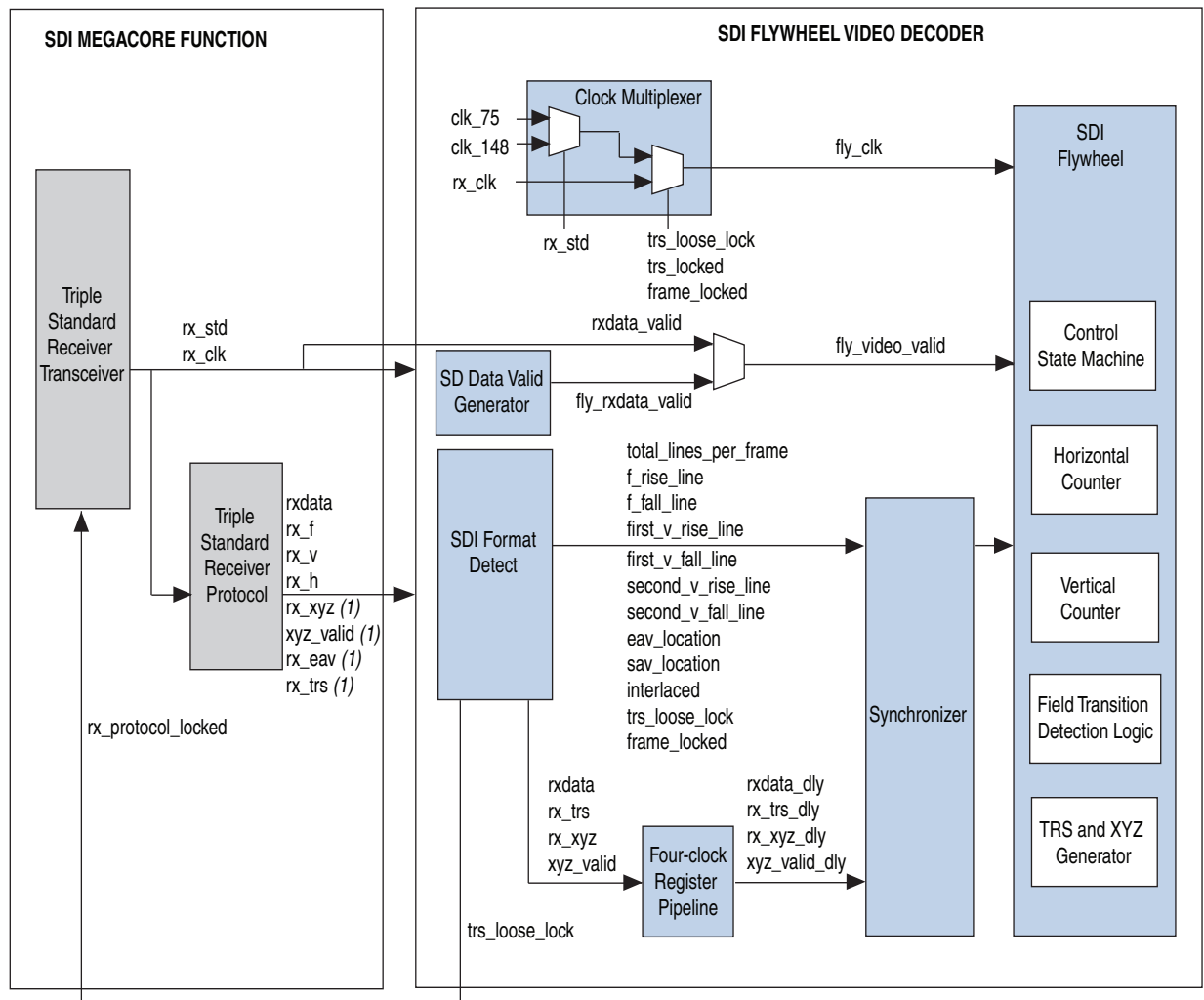
The following sections describe the functional description, interface signals, and steps to get started using the reference design.

Functional Description

Figure 1 on page 1–2 shows the block diagram of the SDI top-level block (ALT_SDI_FLY_TOP) that wraps around the following three blocks:

- SDI Triple Standard Receiver Transceiver
- SDI Triple Standard Receiver Protocol
- SDI Flywheel Video Decoder

Figure 1. SDI Top-Level Block Diagram

**Note to Figure 1:**

(1) These signals are implemented as output ports in version 9.0 of the SDI MegaCore function.

SDI Triple Standard Receiver Transceiver

This reference design uses a split transceiver because the transceiver has the input `rx_protocol_locked` port. The `rx_protocol_locked` port is the input to the transceiver control logic. When active, this port indicates to the transceiver control logic that the protocol blocks are locked to the incoming video stream, and stops the transceiver search algorithm at the current rate. When inactive, this port restarts the transceiver search algorithm.

The `trs_loose_lock` signal from the SDI format detect block controls the locking of the transceiver. A single, valid time reference signal (TRS) asserts the `trs_loose_lock` signal indicating that the receiver is acquiring valid SDI samples. This signal is deasserted when a specified number of consecutive missing end of active videos (EAV) is detected.

The triple-rate detection depends on whether the transceiver is running in 3G-SDI or SD-SDI mode. If there is no SDI signal, the core sets the transceiver in 3G-SDI or SD-SDI mode to detect the rate. If HD-SDI mode is detected, the core resets the transceiver to 3G-SDI or SD-SDI mode to detect the rate, and then reprograms back to HD-SDI mode so that the transceiver can be correctly received.

SDI Triple Standard Receiver Protocol

The SDI triple standard receiver protocol block (SDI_TR_RX_PROTO) examines the video stream for TRS symbols. The block also indicates that the current word is either an EAV or a start of active video (SAV), decodes the XYZ word, and outputs the bit signals (F, V, H).

SDI Flywheel Video Decoder

The SDI flywheel video decoder comprises the following blocks:

- SDI Format Detect
- SD Data Valid Generator
- Clock Multiplexer
- Four-Clock Register Pipeline
- Synchronizer
- SDI Flywheel

SDI Format Detect

The SDI triple standard receiver protocol block already has an SDI format code. The SDI format detect block (SDI_FORMAT_DETECT) is enhanced from the existing SDI format code so that you can edit and modify the line and frame timing details.

The SDI format detect block monitors the line and frame timing of an incoming SDI stream. This block generates various signals (for example, `trs_loose_lock`, `trs_locked`, and `frame_locked`) that indicate whether the receiver is locked or unlocked to the stream. The SDI format detect block also decodes frame information such as `f_rise_line`, `f_fall_line`, and so on.



Because the existing SDI format code is not used, ignore the `rx_status[4:3]` bits about the locked signals generated by the code.

SD Data Valid Generator

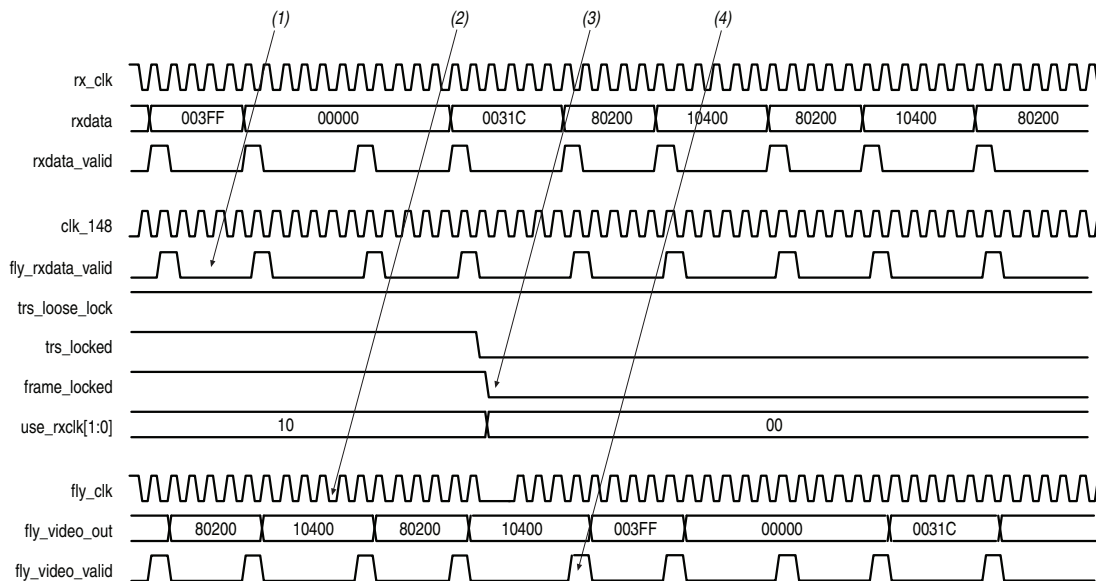
The triple-standard variant of the SDI MegaCore function requires 11 times oversampling to receive the SD-SDI data. For this oversampling scheme, the data valid pattern is 1 high cycle, 4 low cycles, 1 high cycle, and 5 low cycles. When the receiver is unlocked to the incoming SD-SDI data, the SD data valid generator block (SD_DATA_VALID_GEN) generates the data valid signal. This data signal allows the SDI flywheel block to continue generating SD-SDI black-level video data at the previous SD rate. The data valid signal is synchronous to the local phase-locked loop (PLL) clock at 148.5 MHz. Refer to [Figure 2 on page 1–4](#) for the data valid switchover timing diagram.

Clock Multiplexer

The clock multiplexer block (CLOCK_MUX) performs clock switchover. When `trs_loose_lock`, `trs_locked`, and `frame_locked` signals are high, indicating that the receiver is locked to the incoming data, choose the transceiver-recovered clock (`rx_clk`) as the clock source for the SDI flywheel block. When any of the locked signals is low, indicating that the receiver is unlocked to the incoming data, choose the local PLL clock based on the currently received video rate.

The timing diagram in [Figure 2](#) shows the clock and data valid switchover when the receiver is unlocked to the incoming SD-SDI input stream.

Figure 2. Clock and Data Valid Switchover



Notes to Figure 2:

- (1) The SD data valid generator block generates `fly_rxdata_valid` signal synchronous to local PLL clock (`clk_148`) at 148.5 MHz.
- (2) The SDI flywheel block uses `rx_clk` as a clock source and `rxdata_valid` as the video output valid signal when the receiver is locked to the incoming SD-SDI.
- (3) When the receiver is unlocked to the incoming SD-SDI, clock and data valid switchover happens.
- (4) The SDI flywheel block uses `clk_148` as a clock source and internally generated `fly_rxdata_valid` as the video output valid signal.

Although the `trs_locked` signal indicates that the receiver is locked to the incoming data and `rx_clk` is stable, the SDI flywheel block waits for the `frame_locked` signal to be asserted before selecting `rx_clk`. When the `frame_locked` signal is asserted, the SDI flywheel block initiates the SDI flywheel control state machine to synchronize to the input video stream. The line and frame timing details are important and useful to the SDI flywheel block for synchronization, and the details are only valid when the `frame_locked` signal is asserted.

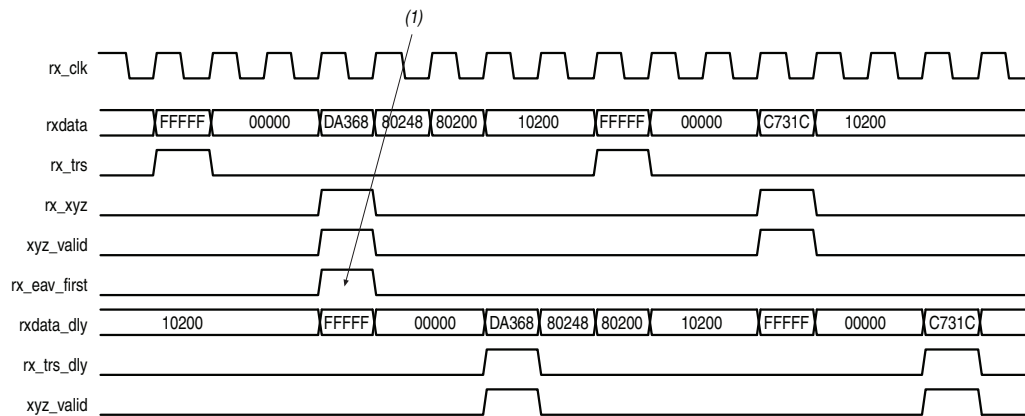
Four-Clock Register Pipeline

The four-clock register pipeline (4PIPE_REG) delays the input video stream and the XYZ decoded signals (`rxdata`, `rx_trs`, `rx_xyz`, and `xyz_valid`) by four clock cycles. By delaying the input video stream and the XYZ decoded signals, the clock register pipeline aligns the `rx_eav` signal from the SDI triple standard receiver protocol block with the first word of the TRS (EAV).

The first three words, 3FF, 000, 000, form a unique pattern in the video stream. The `rx_eav` signal is now considered `rx_eav_first`, as the signal is now asserted on the first word, 3FF, of the delayed input video stream.

The timing diagram in [Figure 3](#) shows how the input video stream and the XYZ decoded signals are delayed by four clock cycles before going to the SDI flywheel block.

Figure 3. Delayed Input Video Stream and XYZ Decoded Signals



Note to Figure 3:

- (1) The `rx_eav_first` signal now accompanies the first word of the TRS (EAV). This signal is identical to the `rx_eav` signal that comes from the SDI triple standard receiver protocol block.

Synchronizer

The synchronizer block (`SYNC_TO_FLY_CLK`) synchronizes the data to the `fly_clk` domain for metastability protection. The delayed version of the `frame_locked` signal is used as an enable line to ensure the SDI flywheel block gets stable line and frame timing details during synchronization.

SDI Flywheel

The SDI flywheel block provides video timing in the presence of noisy or interrupted input video data. When the SDI format detect block asserts the `frame_locked` signal to indicate that the receiver has locked to the input video stream, the SDI flywheel block synchronizes its internal data to the input video stream. After the synchronization, the generated video timing corresponds to the input video stream timing. The SDI flywheel block provides noise immunity by requiring a significant number of consecutive missing EAVs to occur before synchronizing again.

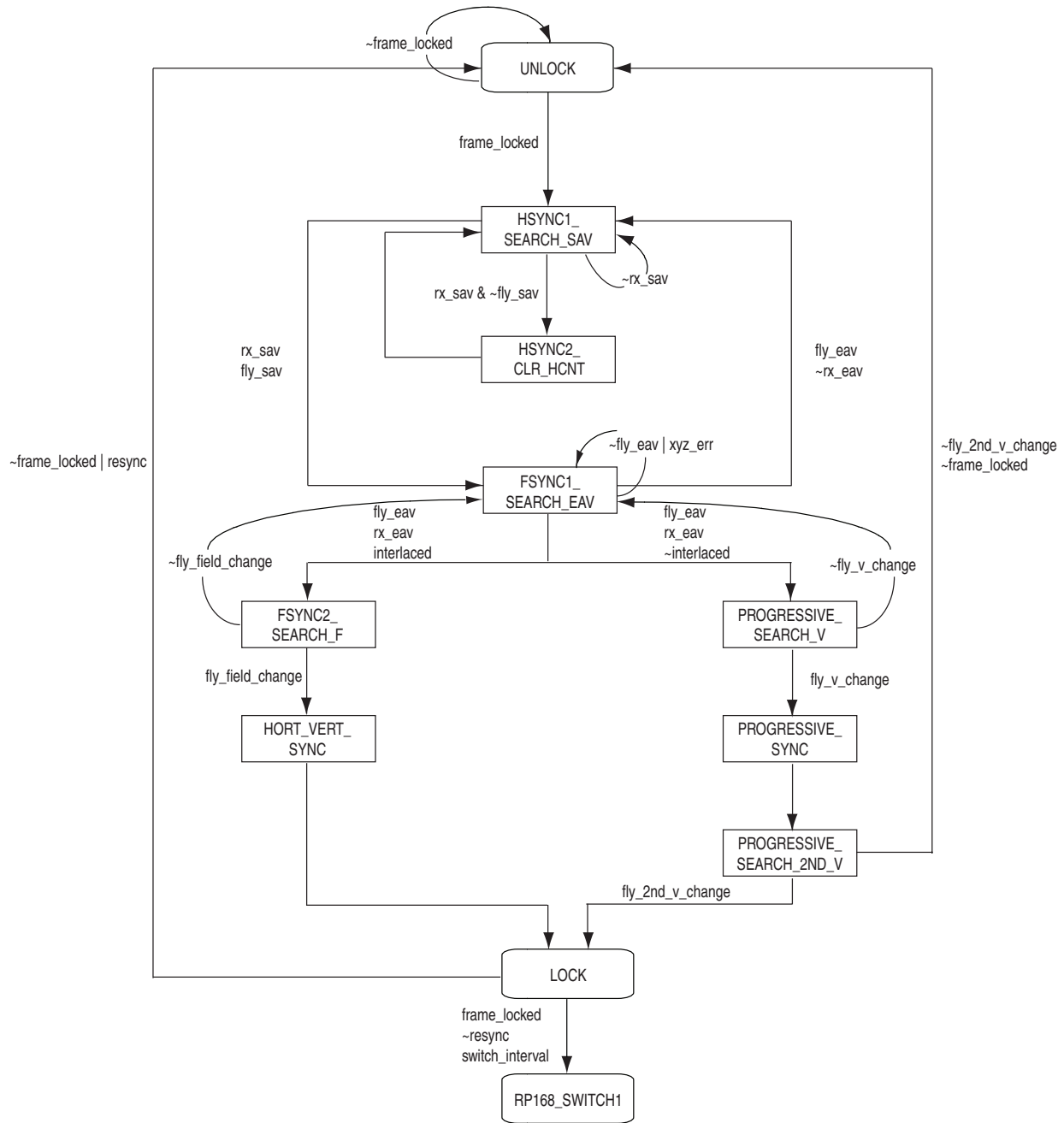
The SDI flywheel block generates and inserts the TRS symbols into the video stream. The block overwrites the data in the input video stream, where the TRS symbols occur, to repair any damaged TRS symbols in the input video stream. If the completely missing TRS symbols are more than the threshold you specified, the SDI flywheel block generates black-level video data and inserts them in place of a TRS symbol in the input video stream at a previous timing.

The SDI flywheel block contains the following sub-blocks:

- Horizontal counter (SDI_FLY_HCNT)— increments at each valid word. It also generates the TRS symbol signal (`fly_trs`), the XYZ symbol signal (`fly_xyz`), and the horizontal blanking signal (`fly_h_blank`).
- Vertical counter (SDI_FLY_VCNT)— increments when an EAV is detected. It also generates the vertical blanking signal (`fly_v_blank`).
- Field transition detection logic (SDI_FLY_FCNT)— contains the field transition detection logic for interlaced frame. This logic captures the F bit from every EAV in the input video stream and compares the bit to the F bit from the previous EAV. When the F bit changes, this block detects the start of a new field and asserts the `fly_field_change` signal. It also generates the F bit (`fly_f`).
- TRS and XYZ generator (SDI_FLY_TRS_XYZ_GEN)— generates TRS symbols (3FF, 000, 000, XYZ) and the XYZ word based on the flywheel video decoder's internal timing, and inserts them into the video stream. It generates black-level video data when the receiver is unlocked to the input video stream.
- Control state machine (SDI_FLY_CTRL_FSM)— controls the operation of the flywheel video decoder. This state machine is divided into the main loop and the synchronous loop.

Figure 4 shows the function of the main loop.

Figure 4. Main Loop



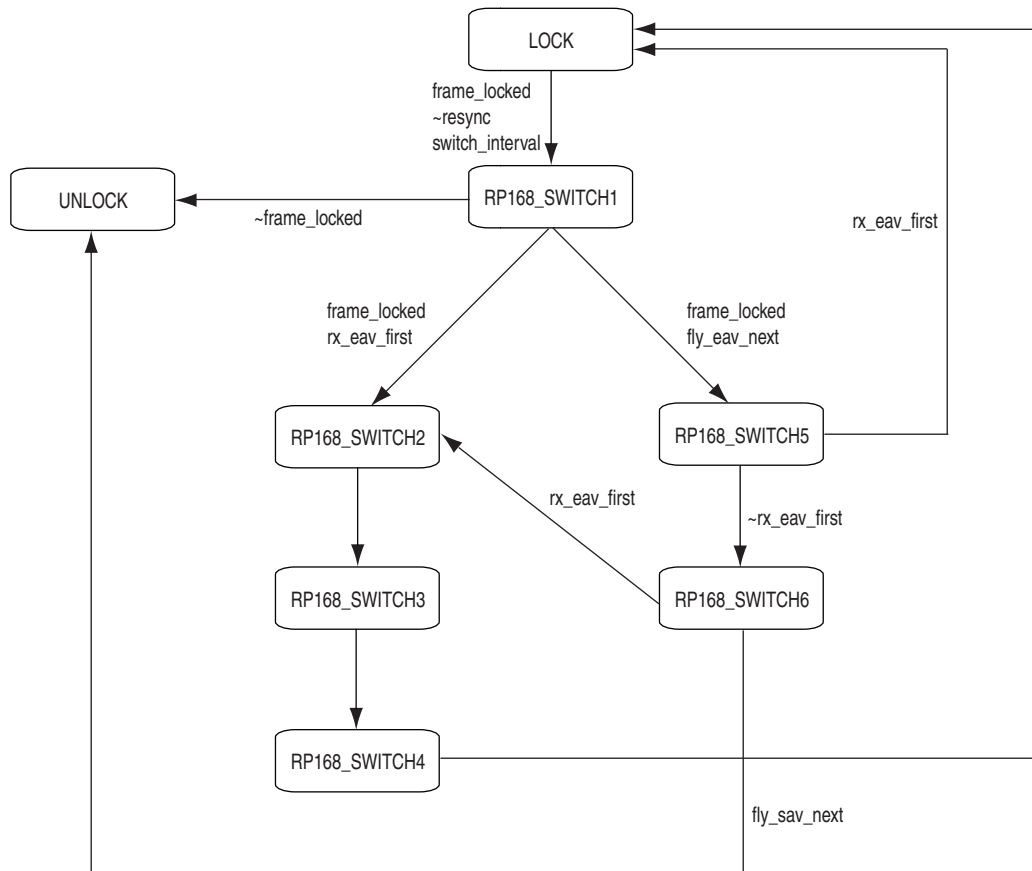
The following steps explain the process in Figure 4.

1. The finite state machine (FSM) starts in the UNLOCK state. It remains in this state until the SDI format detect block asserts the `frame_locked` signal to indicate that the receiver has locked to the input video stream. The FSM then attempts to synchronize to the input video stream. The FSM takes less than two fields (or less than one frame) to synchronize to an input video stream.

2. In the `HSYNC1_SEARCH_SAV` state, the FSM first synchronizes horizontally by looking for SAV symbols in the input video stream. Whenever an SAV is received, the FSM clears the flywheel video decoder's horizontal counter in the horizontal counter block. The clearing is repeated until the flywheel video decoder generates the SAV at the same timing of the SAV in the input video stream. When the positions of the flywheel video decoder's SAV and the input's SAV match, the FSM synchronizes horizontally to the input video stream.
3. The FSM transitions to the `FSYNC1_SEARCH_EAV` state and attempts to synchronize vertically. The FSM looks for the start of a new field to synchronize vertically in one of the following ways:
 - For interlaced frame, the FSM waits for the `fly_field_change` signal from the field transition detection logic block. When a new field is detected, the state machine loads the vertical counter with the value determined based on the line and frame timing.
 - For progressive frame, the FSM waits for a falling V bit to synchronize vertically. When a falling V bit is detected by the vertical counter block, the state machine loads the vertical counter with the value determined based on the line and frame timing.
4. After the vertical counter is loaded, the flywheel video decoder is synchronized to the input video stream. The FSM moves to the `LOCK` state and asserts the `fly_locked` signal. The FSM remains locked until the SDI format detect block detects inconsistent F and V bits in the input video stream, or consecutive missing EAVs more than the threshold you specified. When the inconsistent bits or missing EAVs are detected, the receiver is unlocked to the input video stream and the FSM moves to the `UNLOCK` state.
5. When the FSM moves to the `UNLOCK` state, the flywheel video decoder continues to generate valid TRS symbols based on its internal timing until the flywheel resynchronizes. The FSM does not negate the `fly_locked` signal until the SDI format detect block detects a frame format change in the new input video stream later. It is useful not to negate the `fly_locked` signal because it keeps all downstream video equipment synchronized.

Figure 5 shows the function of the synchronous switching loop.

Figure 5. Synchronous Switching Loop



The following steps explain the process in Figure 5.

1. When the synchronous video sources are switched, it disturbs the horizontal timing and the alignment of the stream. However, the vertical timing remains synchronized.



The SMPTE RP168 defines that synchronous switching is allowed to occur at one line per field. The SDI flywheel video decoder accommodates the horizontal offsets that occur on these synchronous switching lines, and immediately resynchronizes to the incoming video stream, if such an offset is detected.

2. During the synchronous switching, the FSM moves to the RP168_SWITCH1 state when you assert the en_sync_switch pulse. In this state, the FSM determines if its internal EAV and the EAV in the input video stream 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.

Refer to Figure 17 and Figure 19 for examples of synchronous switching loop.

Interface Signals

Table 1 shows the input and output ports of the SDI top-level block.

Table 1. Interface Signals (Part 1 of 3)

Port	Width	Direction	Description
rst	1	Input	Reset signal, which holds the receiver and flywheel video decoder in reset. It must be synchronous to the rx_serial_refclk clock domain.
rx_serial_refclk	1	Input	Transceiver training clock for SDI triple standard.
clk_148	1	Input	Local PLL clock (148.5 MHz).
clk_75	1	Input	Local PLL clock (74.25 MHz).
gxb2_cal_clk	1	Input	Calibration clock for Stratix II GX transceivers.
sdi_rx	1	Input	Serial Input.
enable_sd_search	1	Input	Enables search for SD signal in triple-standard mode.
enable_hd_search	1	Input	Enables search for HD signal in triple-standard mode.
enable_3g_search	1	Input	Enables search for 3G signal in triple-standard mode.
en_sync_switch	1	Input	Enables aligner, format detect and flywheel blocks to realign immediately so that the downstream is completely non disruptive.
sdi_reconfig_clk	1	Input	Clock input for the embedded transceiver instance.
sdi_reconfig_togxb	4	Input	Data input for the embedded transceiver instance. The Stratix II GX DPRIO bit, sdi_reconfig_togxb[3] is unused in this reference design.
sdi_reconfig_fromgxb	17	Output	Data output from embedded transceiver instance. The Stratix II GX DPRIO bit, sdi_reconfig_fromgxb[16:1] is unused in this reference design.
sdi_start_reconfig	1	Output	Request from the SDI MegaCore function to start reconfiguration.
sdi_reconfig_done	1	Input	Indicates that reconfiguration has finished.
rx_clk	1	Output	Transceiver clock data recovery (CDR) clock.
rx_std (1)	2	Output	Receive video standard. 00 = SD, 01 = HD, 10 = 3G
rx_status (1)	11	Output	Receiver status: rx_status[0] —transceiver PLL locked rx_status[1] —receiver in reset rx_status[2] —alignment locked (a TRS detected and word alignment performed) rx_status[10:3] —unused in this reference design
rx_ln (1)	22	Output	Receiver line number output. Bits 11 to 21 are unused in this reference design.

Table 1. Interface Signals (Part 2 of 3)

Port	Width	Direction	Description
trs_loose_lock (1)	1	Output	The transceiver control state machine uses this signal to determine if the core should be reset or if the core should search for another video standard. A single and valid TRS asserts this signal. A number of consecutive missing EAVs (user programmable) deasserts this signal. When asserted, the transceiver control state machine locks and stops the transceiver search algorithm at current rate. When deasserted, the transceiver control state machine resets the transceiver (reprogrammed to 3G mode) and restarts the search algorithm.
trs_locked (1)	1	Output	Six consecutive lines of the same EAV and SAV timing asserts this signal.
frame_locked (1)	1	Output	Consistent rising or falling edge on the F or V bits over multiple frames asserts this signal.
rx_ap (1)	2	Output	Receiver active picture synchronization output. Bit 1 is unused in this reference design.
fly_clk	1	Output	Flywheel output clock. When trs_loose_lock, trs_locked, and frame_locked are high, this output clock is equivalent to rx_clk. Otherwise, it is equivalent to clk_148 or clk_75 based on the receiver video standard.
fly_trs (2)	1	Output	Flywheel TRS flag. Asserted during TRS symbol (3FF, 000, 000, XYZ).
fly_eav_next (2)	1	Output	Flywheel EAV indicator flag. Asserted when the next word is the first word of EAV.
fly_sav_next (2)	1	Output	Flywheel SAV indicator flag. Asserted when the next word is the first word of SAV.
fly_xyz_word (2)	1	Output	Flywheel XYZ word flag. Asserted when the current word is the XYZ word of a TRS.
fly_video_out (2)	20	Output	Flywheel output video stream.
fly_video_valid (2)	1	Output	Flywheel output video stream valid signal.
fly_f (2)	1	Output	Flywheel frame synchronization output.
fly_v_blank (2)	1	Output	Flywheel vertical synchronization output.
fly_h_blank (2)	1	Output	Flywheel horizontal synchronization output.
fly_vcount (2)	14	Output	Flywheel current vertical count.
fly_hcount (2)	11	Output	Flywheel current horizontal count.
fly_locked (2)	1	Output	Flywheel synchronization and locked signal. Asserted when the flywheel video decoder is synchronized to the input video stream. Deasserted when the receiver detects a format change in the new input video stream.
fly_std (2)	2	Output	Flywheel video standard. 00 = SD, 01 = HD, 10 = 3G

Table 1. Interface Signals (Part 3 of 3)

Port	Width	Direction	Description
format_change (2)	1	Output	Flywheel format change indicator signal. Asserted when the frame format of the new input video stream does not match with the flywheel video decoder's internally generated frame format.

Notes to Table 1:

- (1) Synchronous to rx_clk domain.
- (2) Synchronous to fly_clk domain.

Getting Started

This section discusses the requirements and related procedures to demonstrate the SDI flywheel video decoder reference design with the Stratix II GX Audio Video development board. This section contains the following topics:

- [Hardware and Software Requirements](#)
- [Directory Structure](#)
- [Hardware Setup](#)
- [Demonstrate the Reference Design](#)

Hardware and Software Requirements

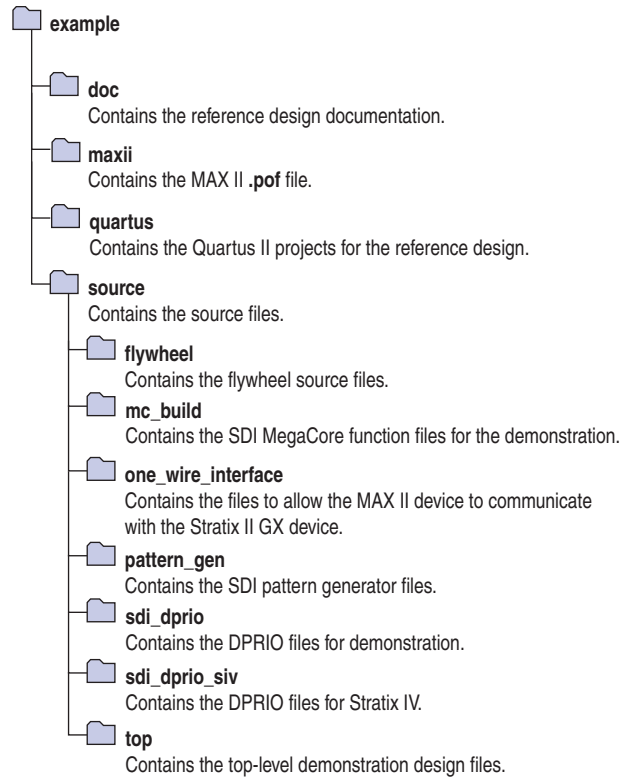
The demonstration requires the following hardware and software:

- Stratix® II GX Audio Video development board
- SDI MegaCore function, version 9.0 or later
- Quartus® II software, version 9.0 or later

Directory Structure

Figure 6 shows the directory structure of the reference design.

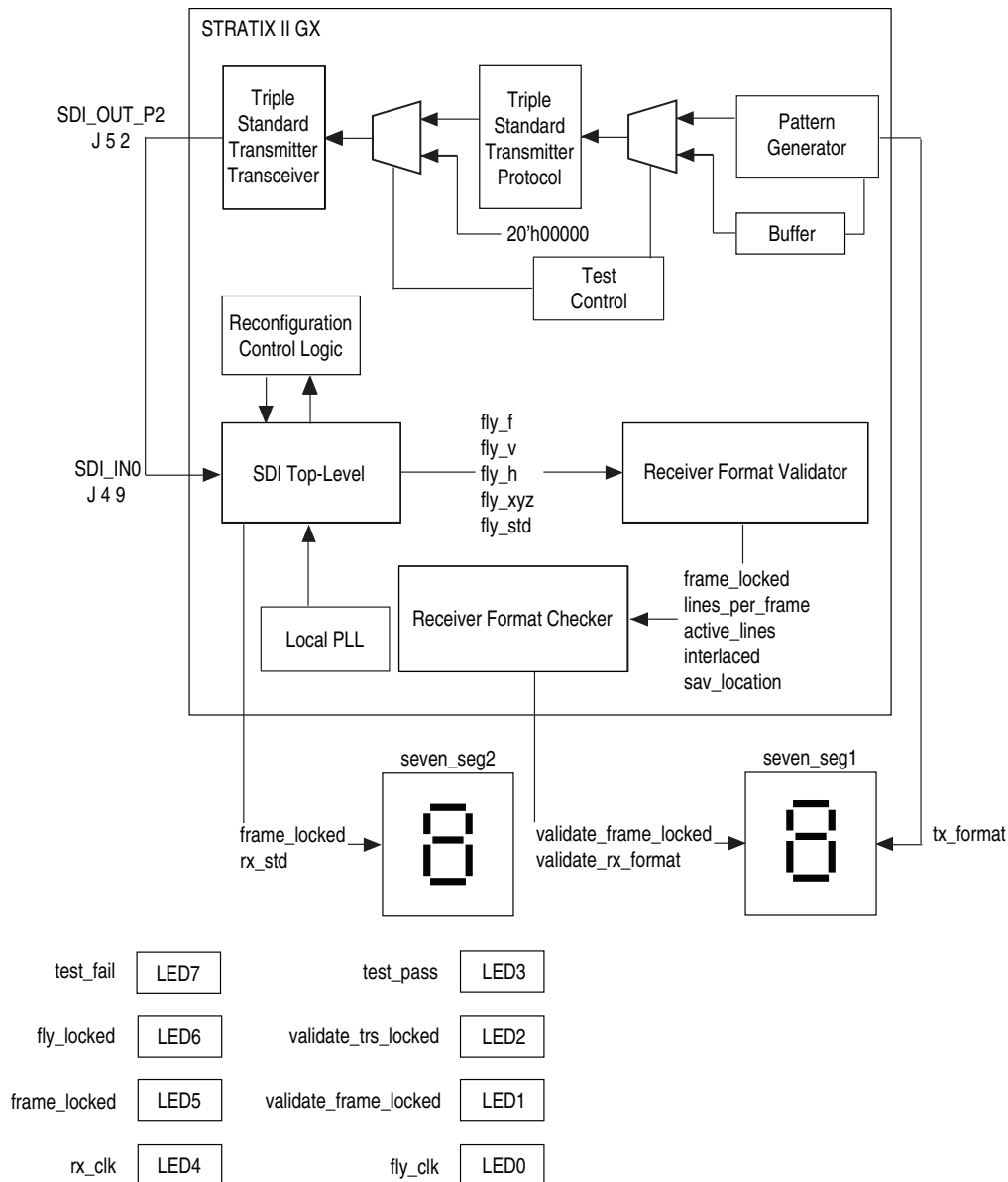
Figure 6. Directory Structure



Hardware Setup

Figure 7 shows the hardware system setup for the demonstration.

Figure 7. Hardware System Setup



SDI Top-Level

The SDI top-level block (ALT_SDI_FLY_TOP) that wraps around the SDI triple standard transmitter, SDI triple standard transmitter protocol and the SDI flywheel video decoder.

Pattern Generator

The pattern generator generates one of the following test patterns:

- 2.97-Gbps 3G-SDI

- 1.485-Gbps HD-SDI
- 270-Mbps SD-SDI

The test pattern generator is customized to generate various formats as shown in [Table 2](#).

Table 2. Formats Generated in Test Pattern Generator

SMPTE	Format	Words/Active Line	Active Line/Frame	Words/Total Line	Total Line/Frame	Frame Rate
260 M	1035i	1,920	1,035	2,200	1,125	30
274 M	1080i	1,920	1,080	2,200	1,125	30
				2,640	1,125	25
	1080p	1,920	1,080	2,200	1,125	60
				2,640	1,125	50
				2,200	1,125	30
				2,640	1,125	25
				2,750	1,125	24
295 M	1080i	1,920	1,080	2,376	1,250	25
296 M	720p	1,280	720	1,650	750	60
				1,980	750	50
				3,300	750	30
				3,960	750	25
				4,125	750	24
125 M	NSTC	1,440	487	1,716	525	30
	PAL	1,440	576	1,728	625	25

SDI Triple Standard Transmitter

The SDI triple standard transmitter takes the input from the pattern generator and generates the same test pattern: 2.97-Gbps 3G-SDI, 1.485-Gbps HD-SDI, or 270-Mbps SD-SDI. You use the split transceiver and protocol configuration to transmit all zeros during dead-time period when testing for the RP168 feature.

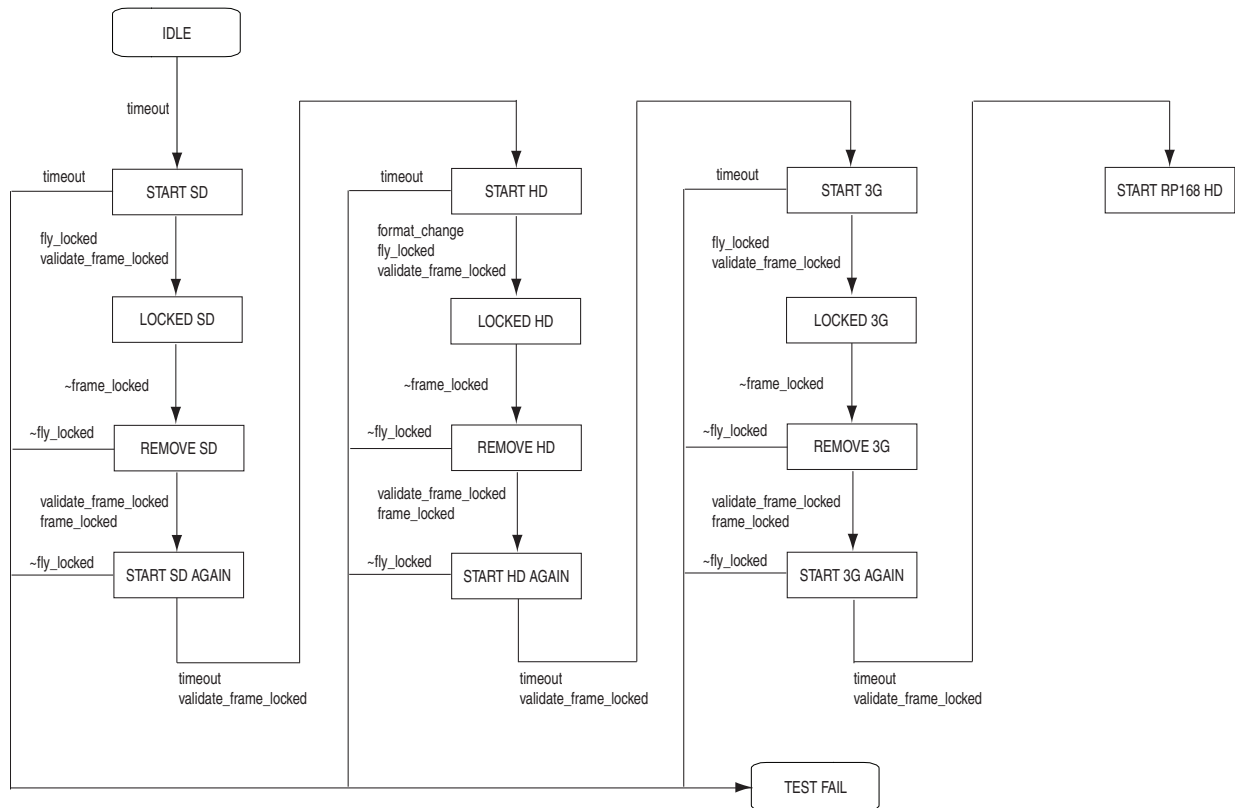
Test Control

The test control block is based on an FSM. This FSM has two main loops:

- Asynchronous switching loop
- Synchronous switching loop

Figure 8 shows the asynchronous switching loop in the FSM.

Figure 8. Asynchronous Switching Loop Diagram



The asynchronous switching loop cycles through transmitting SD-SDI, HD-SDI, and 3G-SDI to determine whether the SDI top-level block (*ALT_SDI_FLY_TOP*) recognizes and locks to each video standard. In the system top-level file (*tr_sdi.v*), the following default settings for transmit format for each video standard are used:

```

// user specified Transmit format
parameter [3:0] TX_FORMAT_SD = FORMAT_PAL;
parameter [3:0] TX_FORMAT_HD = FORMAT_GH;
parameter [3:0] TX_FORMAT_3G = FORMAT_I_3G;
  
```

You can choose to transmit any format supported for each video standard by changing the following TX_FORMAT_XX parameters:

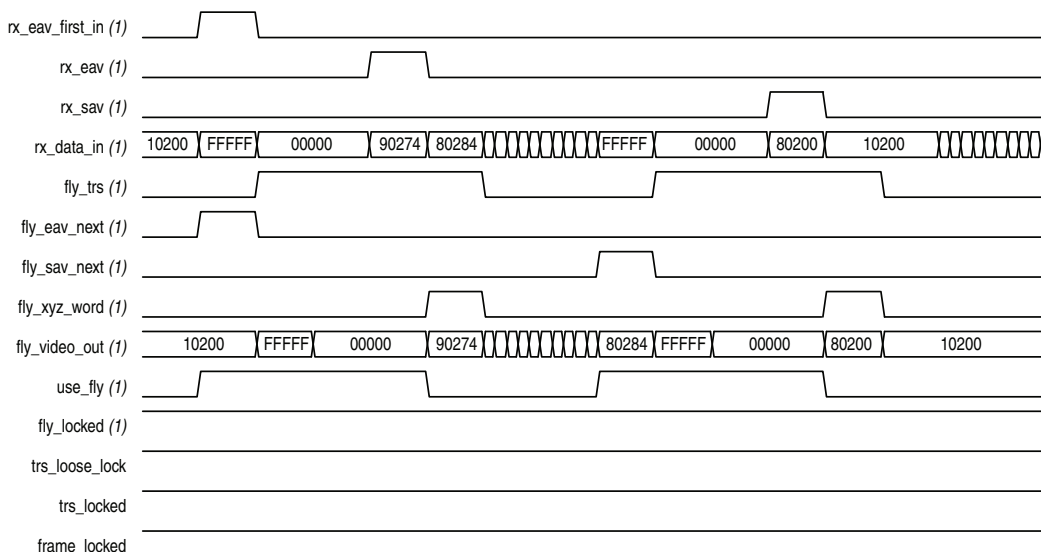
```
// This group of parameters defines the transmit format that supported
parameter [3:0] FORMAT_NTSC = 4'b0000; // 25M SD
parameter [3:0] FORMAT_PAL  = 4'b0001; // 125M SD
parameter [3:0] FORMAT_AB   = 4'b0010; // 260M HD 1035i30
parameter [3:0] FORMAT_C    = 4'b0011; // 295M HD 1080i25
parameter [3:0] FORMAT_DE   = 4'b0100; // 274M HD 1080i30
parameter [3:0] FORMAT_F    = 4'b0101; // 274M HD 1080i25
parameter [3:0] FORMAT_GH   = 4'b0110; // 274M HD 1080p30
parameter [3:0] FORMAT_I    = 4'b0111; // 274M HD 1080p25
parameter [3:0] FORMAT_JK   = 4'b1000; // 274M HD 1080p24
parameter [3:0] FORMAT_LM   = 4'b1001; // 296M HD 720p60
parameter [3:0] FORMAT_N    = 4'b1010; // 296M HD 720p50
parameter [3:0] FORMAT_O    = 4'b1011; // 296M HD 720p30
parameter [3:0] FORMAT_G_3G = 4'b1100; // 3G 1080p60
parameter [3:0] FORMAT_I_3G = 4'b1101; // 3G 1080p50
```

This test also shows how the flywheel video decoder generates TRS symbols when the input video is interrupted or removed. The transceiver control state machine uses the presence or absence of TRS signals in the stream to determine if the SDI is being correctly received. A single, valid TRS indicates to the transceiver control state machine that the receiver is acquiring some valid SDI samples. This signal is only deasserted when the state machine does not detect any TRS sequences more than the threshold you specified. This flywheel video decoder tolerates the missing EAVs without resetting the transceiver. You can set the threshold by changing TOTAL_CONSECUTIVE_MISSING_EAV parameter in the system top-level file (**tr_sdi.v**):

```
// This parameter allows the missing EAVs to be tolerated
parameter [7:0] TOTAL_CONSECUTIVE_MISSING_EAV = 8'd4;
```

The FSM of the test control block begins in IDLE state. After time-out, the FSM starts the test by configuring the pattern generator and the SDI transmitter to transmit the SD-SDI output. After the SDI top-level block recognizes and locks to the incoming stream, and the downstream logic receiver format validator (RX_FORMAT_VALIDATE) and receiver format checker (RX_FORMAT_CHECKER) blocks detect and validate the frame format generated by the SDI top-level block; the FSM moves to the LOCKED_SD state.

The timing diagram in [Figure 9 on page 1–18](#) shows when the flywheel video decoder is locked and synchronized to the incoming input stream. The use_fly signal indicates that the flywheel video decoder replaces the TRS symbols in the input stream with its internally generated TRS symbols; repairing any TRS symbols corrupted by noise in the process.

Figure 9. Flywheel Locked and Synchronized to the Incoming Input Stream**Note to Figure 9:**

(1) The input and output ports for the embedded SDI flywheel block.

Table 3 shows the 13-bit register (`result_reg`) that the FSM uses to record the additional test status.

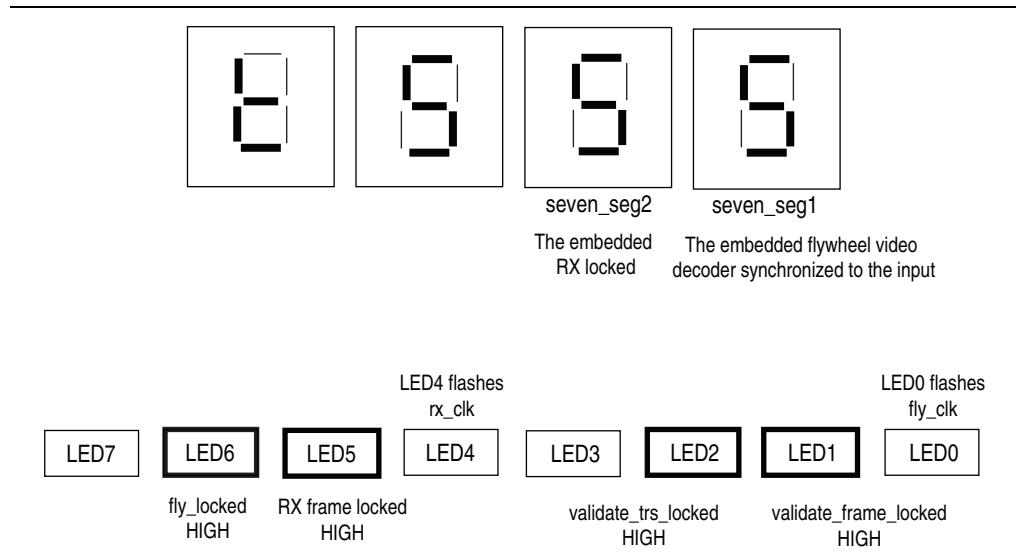
Table 3. 13-bit Register

Bits	Description
0	For SD-SDI. Indicates that the SD RX format generated by the SDI top-level block matches with the TX format.
1	For SD-SDI. Indicates that the SDI top-level block continues generating at previous SD timing even when input is removed.
2	For SD-SDI. Indicates that zero interrupt is detected at SDI top-level block output in terms of the locked signal (for example, the <code>fly_locked</code> signal remains asserted) when switching between same standard and same format.
3	For HD-SDI. Indicates that the HD RX format generated by the SDI top-level block matches with the TX format.
4	For HD-SDI. Indicates that the SDI top-level block continues generating at previous HD timing even when input is removed.
5	For HD-SDI. Indicates that zero interrupt is detected at SDI top-level block output in terms of the locked signal (for example, the <code>fly_locked</code> signal remains asserted) when switching between same standard and same format.
6	For 3G-SDI. Indicates that the 3G RX format generated by the SDI top-level block matches with the TX format.
7	For 3G-SDI. Indicates that the SDI top-level block continues generating at previous 3G timing even when input is removed.
8	For 3G-SDI. Indicates that zero interrupt is detected at SDI top-level block output in terms of the locked signal (for example, the <code>fly_locked</code> signal remains asserted) when switching between same standard and same format.

Table 3. 13-bit Register

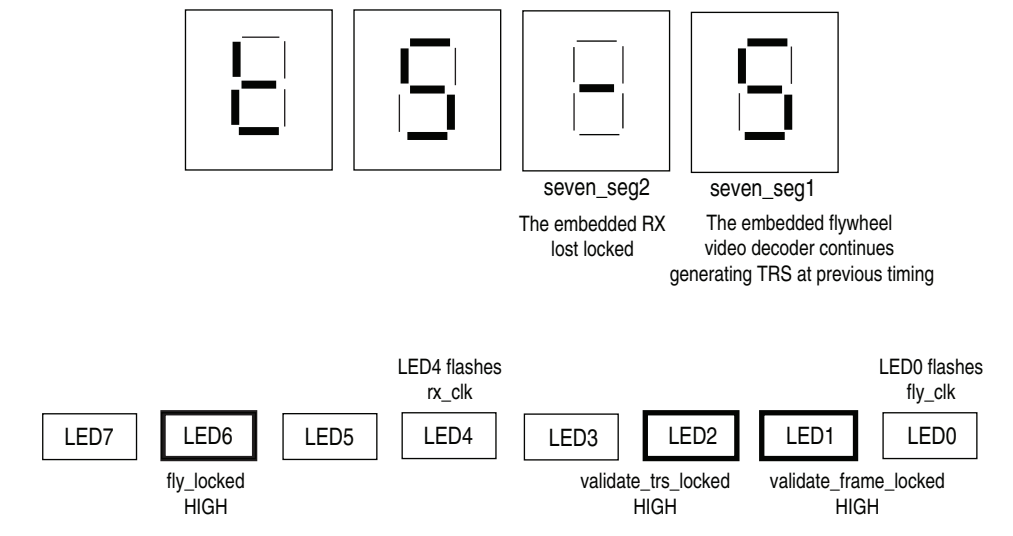
Bits	Description
9	For HD-SDI. Indicates that zero interrupt (<code>fly_locked</code> and <code>frame_locked</code> signals remain asserted) is detected during dead-time period.
10	For HD-SDI. Indicates that zero interrupt (<code>fly_locked</code> and <code>frame_locked</code> signals remain asserted) is detected after synchronous switching.
11	For 3G-SDI. Indicates that zero interrupt (<code>fly_locked</code> and <code>frame_locked</code> signals remain asserted) is detected during dead-time period.
12	For 3G-SDI. Indicates that zero interrupt (<code>fly_locked</code> and <code>frame_locked</code> signals remain asserted) is detected after synchronous switching.

Figure 10 shows what the board displays when the FSM is in LOCKED_SD state.

Figure 10. FSM in LOCKED_SD State

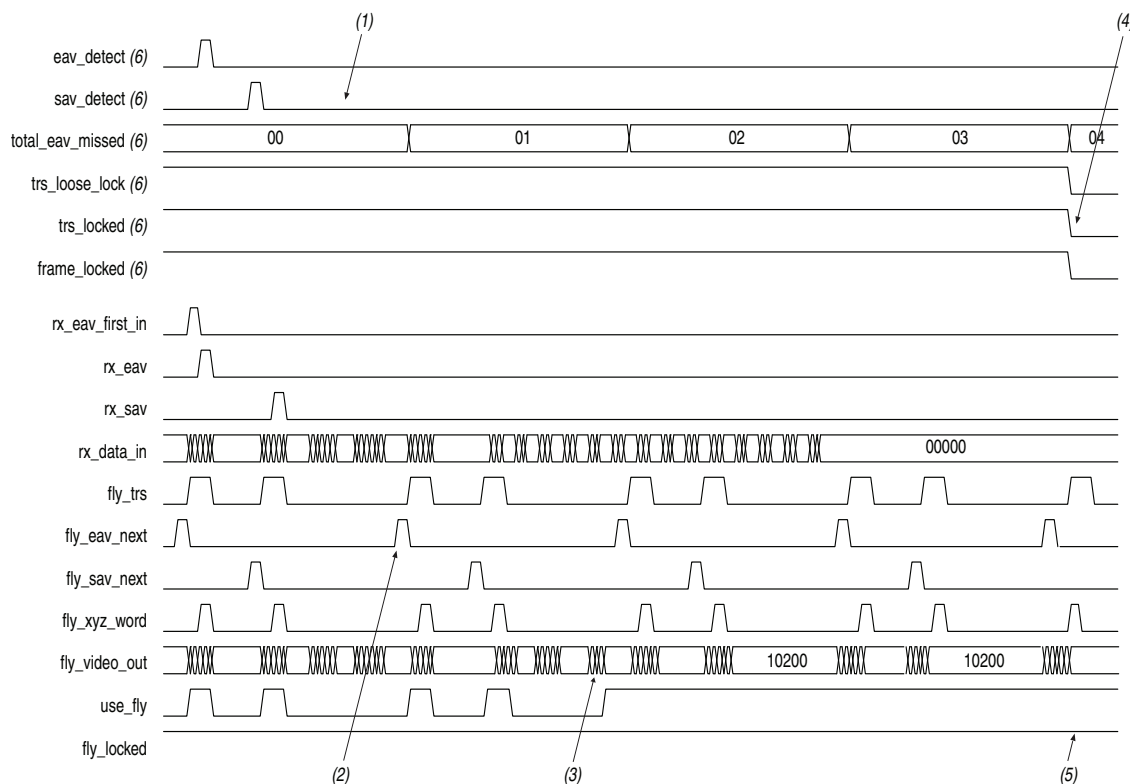
The FSM remains in LOCKED_SD state until you unplug the cable. Then, the FSM moves to REMOVE_SD state and waits for you to reconnect the cable. Figure 11 on page 1–20 shows what the board displays when you remove the input.

Figure 11. FSM in REMOVE_SD State



The timing diagram in [Figure 12](#) shows how the flywheel video decoder continues to generate and insert TRS symbols into the input stream when the input stream is interrupted or removed.

Figure 12. Flywheel Video Decoder Continues Generating TRS Symbols



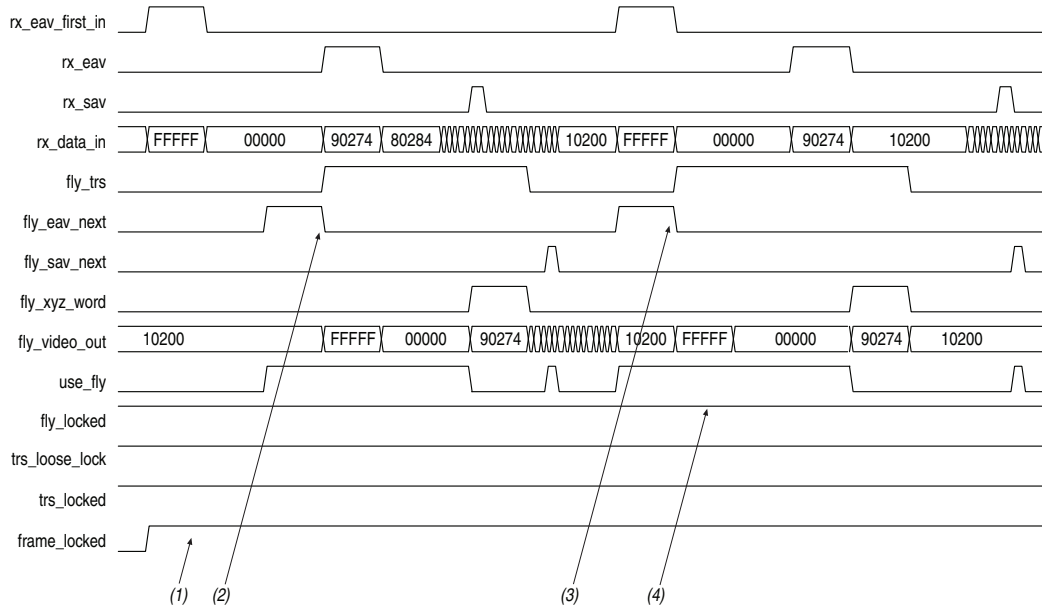
Notes to Figure 12:

- (1) The input video is removed.
- (2) The flywheel video decoder continues to generate TRS at previous timing. The visual information in the resulting video stream is now invalid but the timing information in the TRS symbols is valid.
- (3) At the next EAV, the flywheel video decoder starts generating black-level video data instead of using the incoming stream (`use_fly` signal remains asserted).
- (4) The locked signals of the embedded format detect block is deasserted when four consecutive missing EAVs are detected. In this reference design, `TOTAL_CONSECUTIVE_MISSING_EAV` is set to 8 * d4.
- (5) The flywheel video decoder continues to generate black-level video data at previous timing while the `fly_locked` signal remains asserted.
- (6) Signals from the embedded SDI format detect block.

After you reconnect the cable, the embedded flywheel video decoder resynchronizes to the new incoming input. The FSM moves to the `START_SD_AGAIN` state. Now, the board displays the output shown in [Figure 10](#). The flywheel video decoder does not deassert the `fly_locked` signal when you remove and reconnect the input with the same standard and format. During this period, the FSM continuously monitors the `fly_locked` signal.

The timing diagram in [Figure 13](#) shows how the flywheel video decoder synchronizes to the new incoming input stream when it switches between the same video standard and format.

Figure 13. Flywheel Video Decoder Synchronizes —Switching between Same Video Standard and Format



Notes to Figure 13:

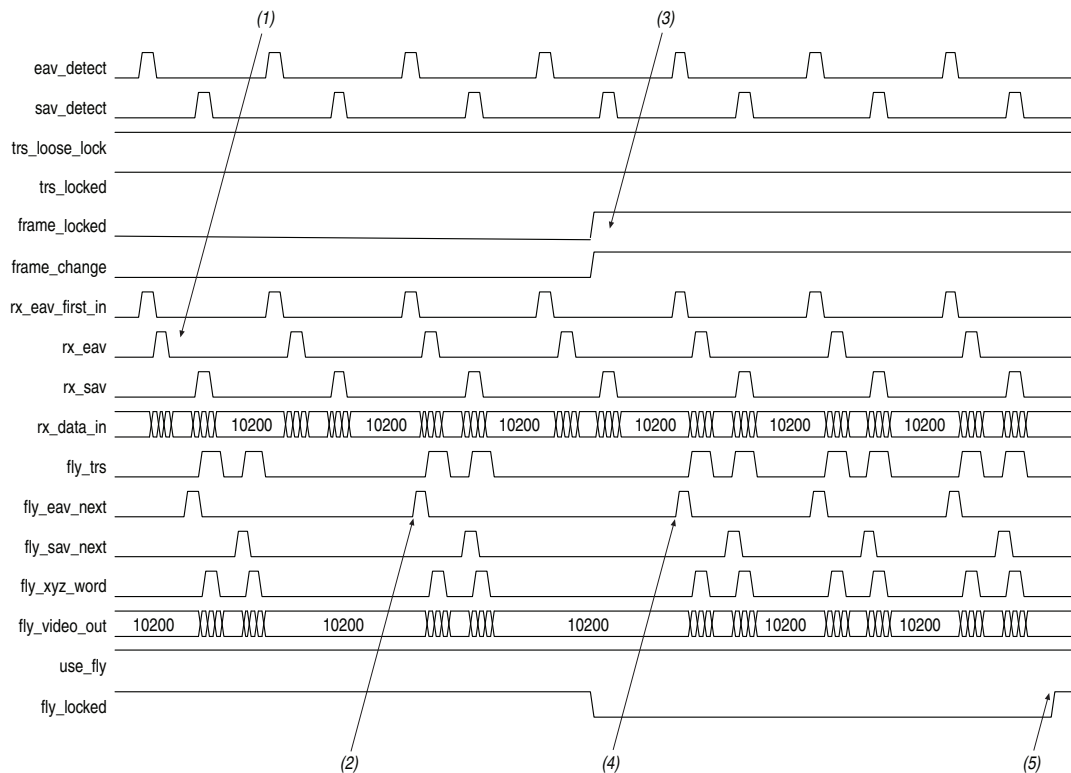
- (1) Once the embedded format detect block has detected a consistent sync pulse (F, V, H) over a number of lines or frames, `frame_locked` is asserted and the flywheel control state machine starts resynchronizing to the new incoming input stream.
- (2) Before the flywheel video decoder synchronizes to the new horizontal timing, it continues to generate at the previous horizontal timing.
- (3) The flywheel video decoder immediately synchronizes to the new horizontal timing at next EAV.
- (4) The `fly_locked` signal remains high if there is no change in the received video format.

Until this synchronization state, the SDI top-level block synchronizes to the incoming input, generates TRS symbols when the input is interrupted or removed, and produces zero interrupt (`fly_locked` signal remains asserted) when switching between the same video standard and format.

Next, the FSM transitions to the `START_HD` state after a specified time. The `START_HD` state configures the pattern generator and the SDI TX to transmit HD-SDI test pattern. The FSM detects the `format_change` signal that is asserted by the SDI top-level block when there is a change in the video format.

The timing diagram in [Figure 14](#) shows how the flywheel video decoder synchronizes to the new incoming input stream when it switches between different video standard and format.

Figure 14. Flywheel Synchronizes —Switching between Different Video Standard and Format



Notes to [Figure 14](#):

- (1) The new incoming input at 3G rate.
- (2) The flywheel video decoder is still generating black-level video data at the previous HD rate.
- (3) The format detect block detects a consecutive valid frame and asserts the `frame_locked` signal for the flywheel video decoder to resynchronize. The `fly_locked` signal is deasserted as the received video format changes.
- (4) The flywheel video decoder synchronizes to the incoming 3G input at the next incoming EAV.
- (5) The flywheel video decoder synchronizes and locks to the incoming 3G input.

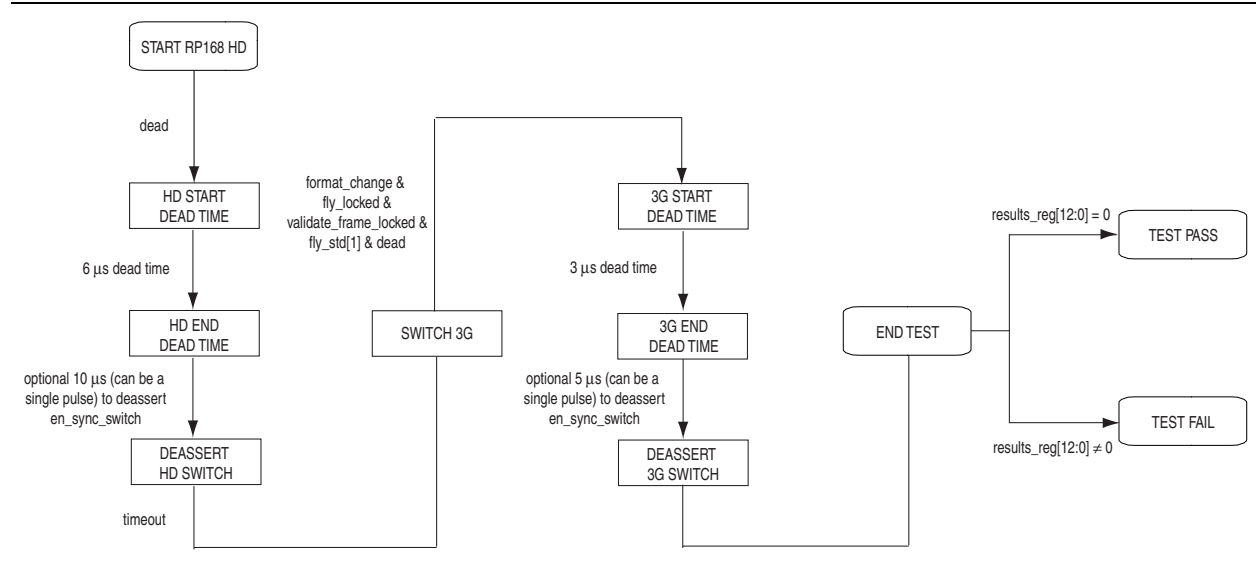
Once the SDI top-level block locks to the HD input and verifies its generated format with the transmit format, the FSM moves to the `LOCKED_HD` state indicating that asynchronous switching with different video standard and format (for example, from SD PAL to HD 1080p30) occurs. The FSM remains at this state until you unplug the cable. The FSM then moves to the `REMOVE_HD` state and waits for you to reconnect the cable.

The FSM repeats the same test sequence for 3G-SDI (refer to [Figure 8](#) on page 1–16). Once you remove and reconnect the cable for the 3G-SDI test case, the FSM enters the synchronous switching loop (refer to [Figure 15](#) on page 1–24). This loop tests for the RP168 feature by switching the same video format during switching line. The switching is done completely non disruptive and the downstream user logic is not aware of this change. The `fly_locked` and `frame_locked` signals remain asserted during and after the synchronous switching.

Synchronous Switching Loop

Figure 15 on page 1–24 shows the synchronous switching loop in the FSM.

Figure 15. Synchronous Switching Loop Diagram

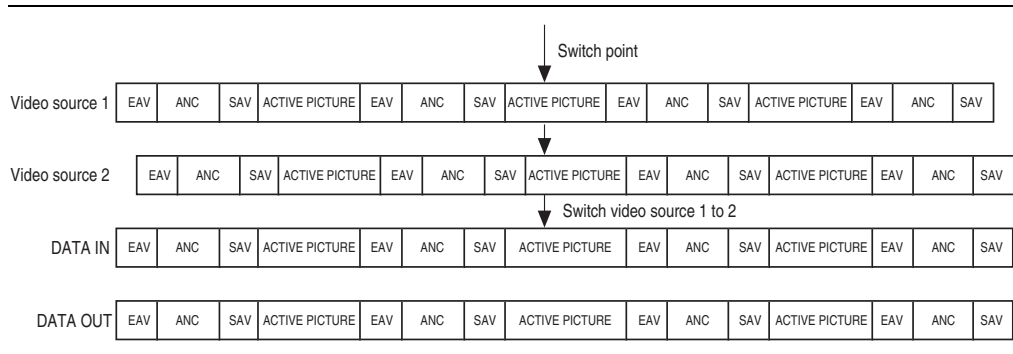


The synchronous switching loop cycles through transmitting HD-SDI and 3G-SDI similar to the asynchronous switching loop. The FSM starts the dead-time period and moves to the HD_START_DEAD_TIME state. During the dead-time period, the FSM controls the SDI triple standard transmitter block to transmit all zeros to mimic the signal loss situation on the receiver side.

For HD 1080p30, the dead-time period is 6μs and the signal loss occurs at line 7 and switching area between 625 and 1,070 pixels.

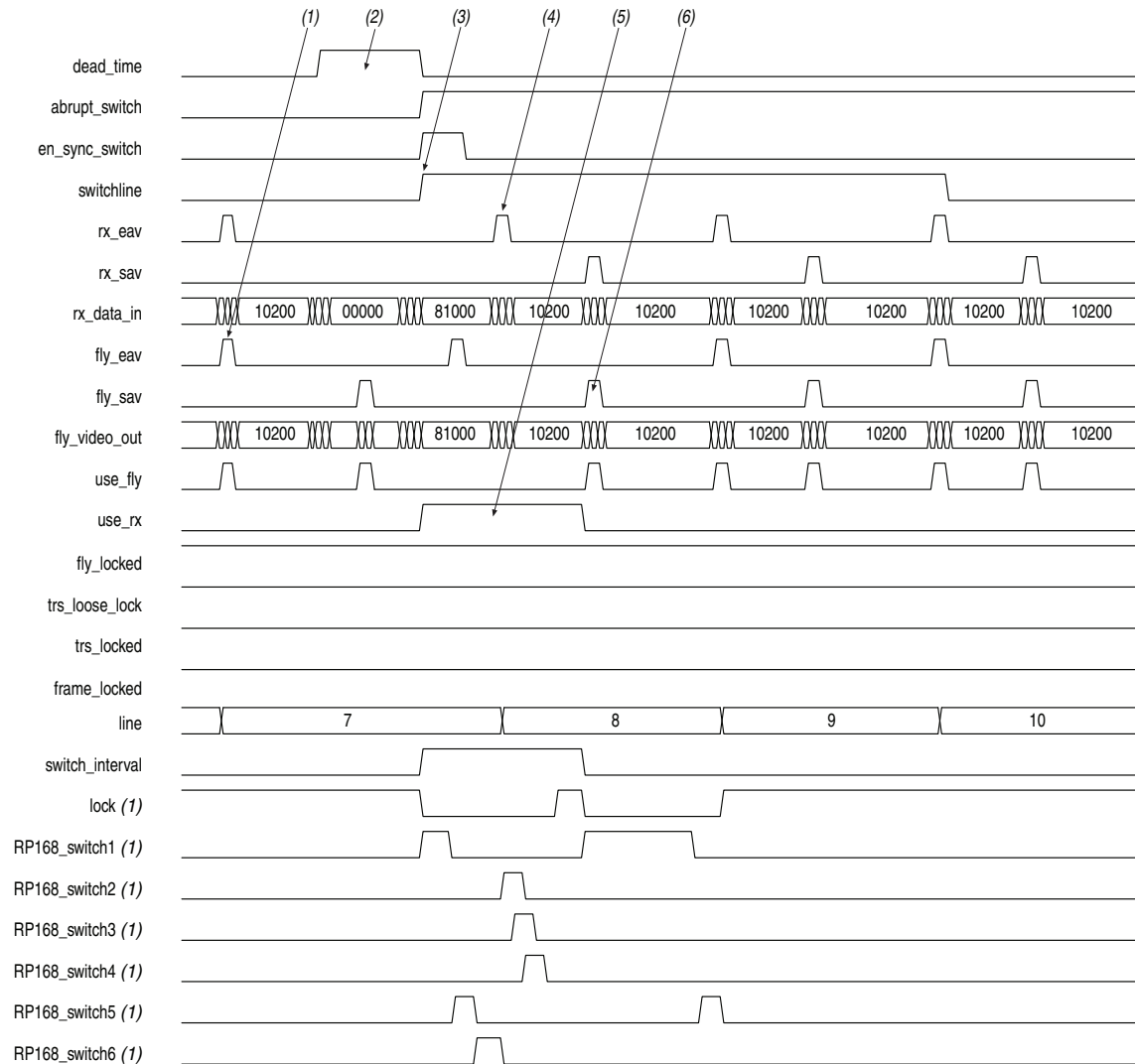
After 6μs, the FSM moves to the HD_END_DEAD_TIME state and performs abrupt switching. The machine controls the SDI triple standard transmitter protocol block to use the delayed version of the TX data from the pattern generator. The SDI top-level block receives a delayed version of the input stream after the dead-time period. The FSM mimics a scenario of switching from video source 1 to video source 2 where the stream of video source 2 is later by a few clock cycles as shown in Figure 16. The FSM also generates an en_sync_switch pulse for the SDI top-level block.

Figure 16. Switch from Video Source 1 to Video Source 2



The switching happens from video source 1 to video source 2 where the `rx_eav` signal of video source 2 is asserted later than the `fly_eav` signal. The timing diagram in Figure 17 shows how the switching takes place.

Figure 17. HD Synchronous Switching from Video Source 1 to Video Source 2



Notes to Figure 17:

- (1) The flywheel video decoder locks to the incoming HD input.
- (2) 6us dead-time period for HD case. All zeros are received by the receiver but the flywheel video decoder continues to generate TRS timing.
- (3) End of dead-time period and video switch point. The flywheel video decoder latches the asserted `en_sync_switch` pulse to three lines (`switchline`) so that the flywheel video decoder realigns to a new TRS alignment immediately without the locked signals.
- (4) The new EAV is asserted later than the `fly_eav`. Disturbance of the horizontal timing and alignment of the stream.
- (5) During the synchronous switching line (`switch interval` is high), the flywheel video decoder is designed to pass the EAV directly from the input video stream through to the output video stream.
- (6) The flywheel video decoder immediately resynchronizes to the new horizontal timing at the next SAV.

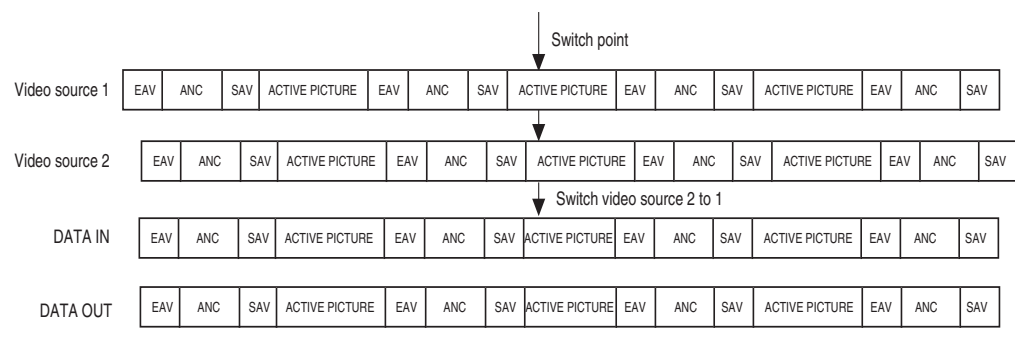
The FSM continuously monitors the `fly_locked` and `frame_locked` signals during and after the synchronous switching before it moves to the `SWITCH_3G` state.

Next, the FSM moves to the SWITCH_3G state. After the SDI top-level block locks to the 3G input and verifies its generated format with the transmit format, the machine starts the dead-time period. The FSM controls the SDI triple standard transmitter block to transmit all zeros to mimic the signal loss situation at the receiver side.

For 3G 1080p50, the dead-time period is 3 μ s and the signal loss occurs at line 7 and switching area between 625 and 1070 pixels.

After 3 μ s, the FSM moves to the END_3G_DEAD_TIME state and performs abrupt switching. The FSM controls the pattern generator to generate the next EAV earlier than the previous timing by mimicking the scenario of switching from video source 2 to video source 1 where the stream of video source 2 is later by few clock cycles as shown in Figure 18.

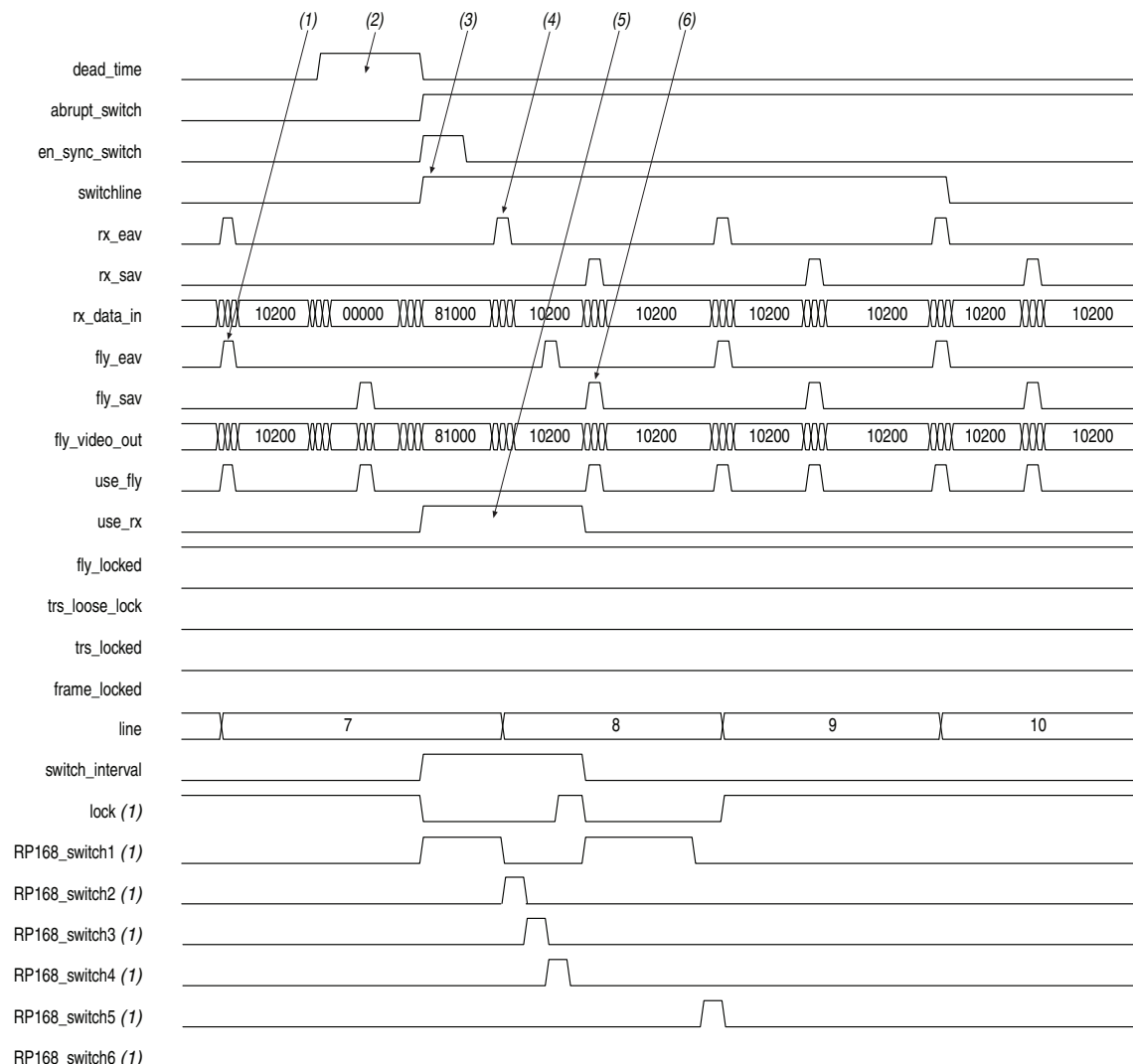
Figure 18. Switch from Video Source 2 to Video Source 1



The switching happens from video source 2 to video source 1 where the rx_eav signal of video source 1 is asserted earlier than the fly_eav signal.

The timing diagram in Figure 19 on page 1–27 shows how the switching takes place.

Figure 19. 3G Synchronous Switching from Video Source 2 to Video Source 1



Notes to Figure 19:

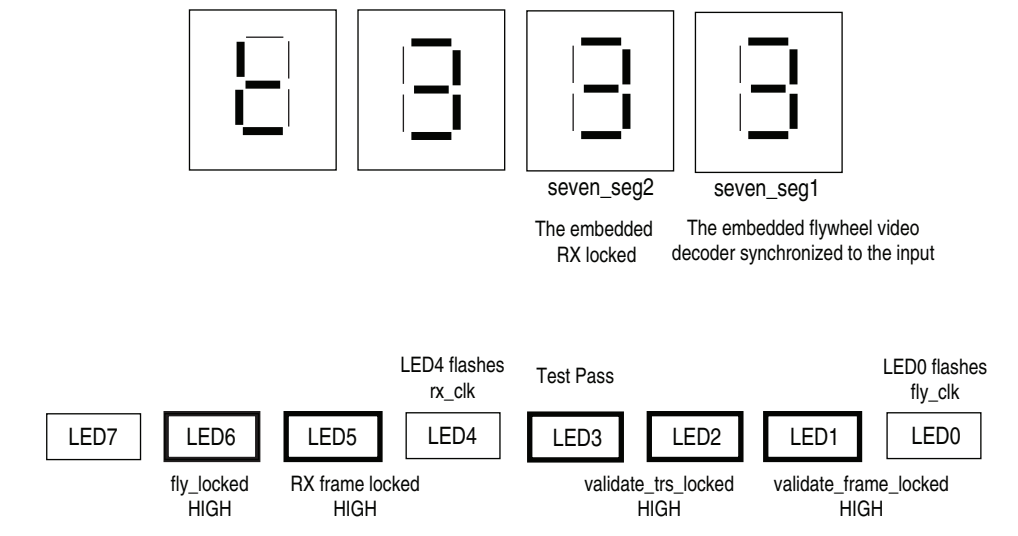
- (1) The flywheel video decoder locks to incoming 3G input.
- (2) 3 μ s dead-time period for 3G case. All zeros are received by the receiver but the flywheel video decoder continues to generate TRS timing.
- (3) End of dead time period and video switch point. The flywheel video decoder latches the `en_sync_switch` pulse to three lines (`switchline`) so that the flywheel video decoder realigns to a new TRS alignment immediately without the locked signals.
- (4) The new EAV is asserted earlier than the `fly_eav` signal. Disturbance of the horizontal timing and alignment of the stream.
- (5) During the synchronous switching line (`switch interval` is high), the flywheel video decoder is designed to pass the EAV directly from the input video stream through to the output video stream.
- (6) The flywheel video decoder immediately resynchronizes to the new horizontal timing at the next SAV.

The machine also generates a necessary `en_sync_switch` pulse for the SDI top-level block. Again, the FSM continuously monitors the `fly_locked` and `frame_locked` signals during and after the synchronous switching.

The synchronous switching loop checks if the SDI top-level block resynchronizes the horizontal timing immediately after a switch has taken place to account for the horizontal disturbance caused by synchronous switching.

Finally, the FSM ends the process by checking the status of all the tests. LED3 illuminates if all tests pass, and LED7 illuminates if one or more tests fails as shown in Figure 20.

Figure 20. End of Test With All Test Passed



Local PLL

The local PLL block (FLY_CLK_GEN) supplies 74.25 MHz (HD-SDI) and 148.5 MHz (SD-SDI and 3G-SDI) for the SDI top-level block. The SDI top-level block only uses this local clock when the receiver is unlocked to the input video stream.

Reconfiguration Control Logic

The reconfiguration control logic block (ALT2GXB_RECONFIG) handles the reconfiguration of the receiver part of the SDI top-level block.

Receiver Format Validator

The receiver format validator block (RX_FORMAT_VALIDATE) is identical to the format detect block embedded in the SDI top-level block. The receiver format validator block monitors the line and frame timing of the SDI stream generated by the SDI top-level block. It generates various locked signals to indicate whether the received stream is locked or unlocked. It also decodes the frame format information for the receiver format checker block.

Receiver Format Checker

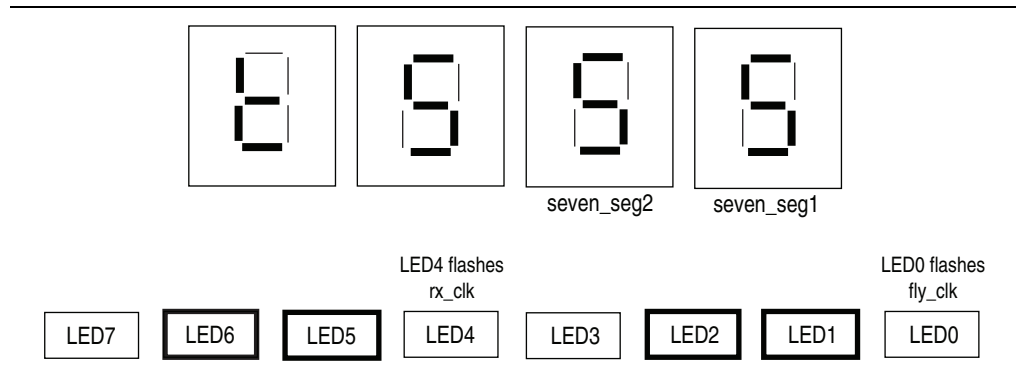
The receiver format checker block (RX_FORMAT_CHECKER) determines the video format of the SDI stream generated by the SDI top-level block based on the line and frame timing decoded by the receiver format validator block. It compares the received video format with the TX format.

Demonstrate the Reference Design

To run the reference design, perform the following steps:

1. Set up the board connections. Connect the test pattern transmitter output, SDI_OUT_P2 (BNC J52), to the receiver input, SDI_IN0 (BNC J49), of the SDI top-level block.
2. Launch the Quartus II software.
 - a. On the File menu, click **Open Project**, navigate to `/quartus/tr_sdi.qpf`, and click **Open**.
 - b. On the Processing menu, click **Start Compilation**.
3. Download the Quartus II generated SRAM Object File (.sof), `/quartus/tr_sdi.sof`.
4. When the LEDs appear as shown in [Figure 21](#), remove the cable from the BNC J49 connector.

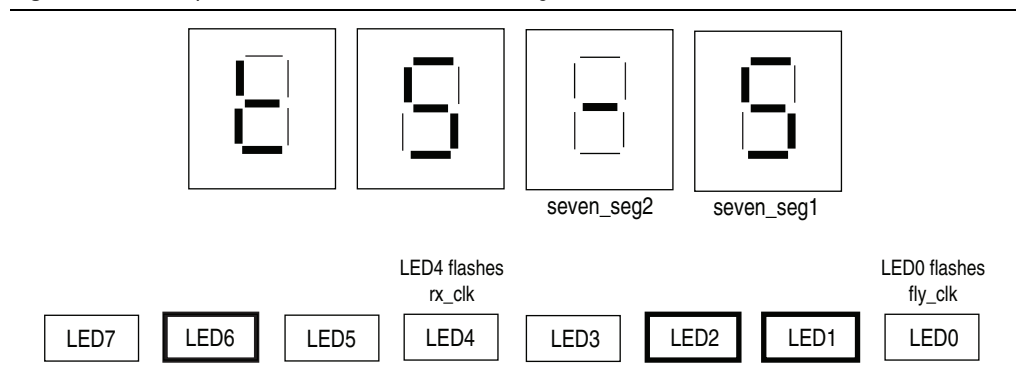
Figure 21. Locked to SD-SDI



Remove the cable only after three seconds upon startup.

5. Remove the cable from the BNC J49 connector. The LEDs appear as shown in [Figure 22](#).

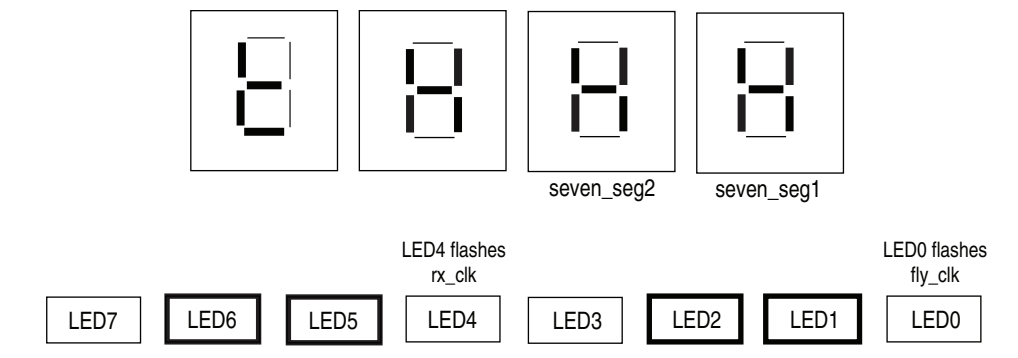
Figure 22. SDI Top-Level Block Continues Generating Data at Previous SD Rate




6. Reconnect the cable to the BNC J49 connector and wait for the LEDs to appear as shown in [Figure 21](#).

7. After a few seconds, the test cycles through to transmit HD-SDI. The LEDs appear as shown in [Figure 23](#).

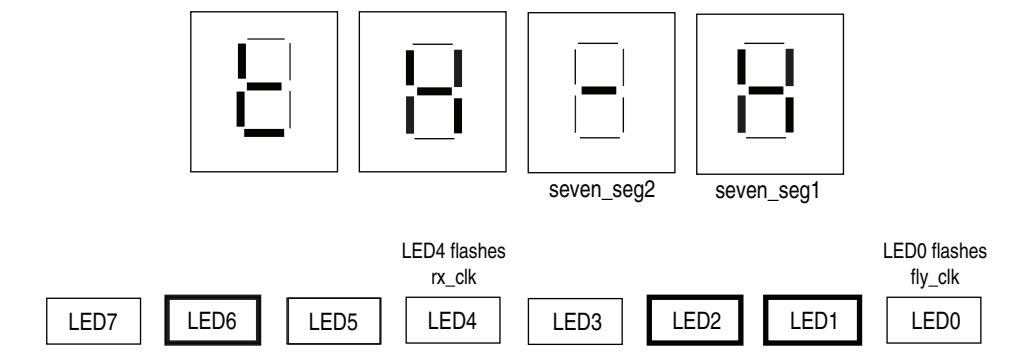
Figure 23. LOCKED to HD-SDI



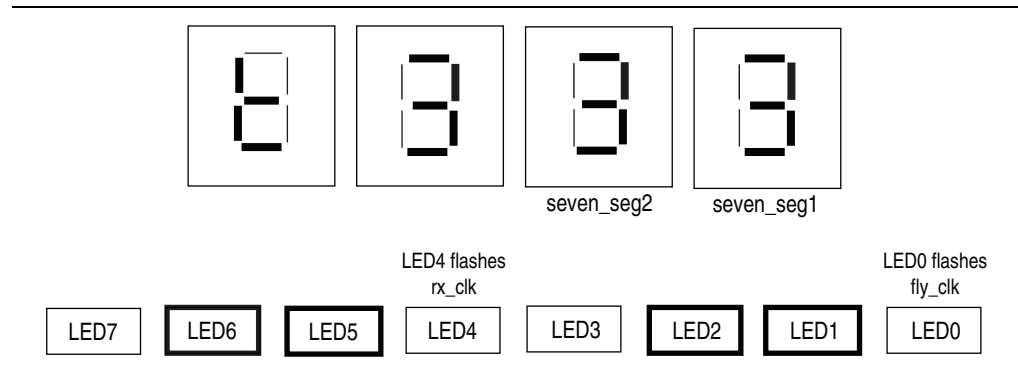
 If the LEDs do not appear as shown in [Figure 23](#), remove the cable and reconnect it.

8. Remove the cable from the BNC J49 connector and wait for the LEDs to appear as shown in [Figure 24](#).

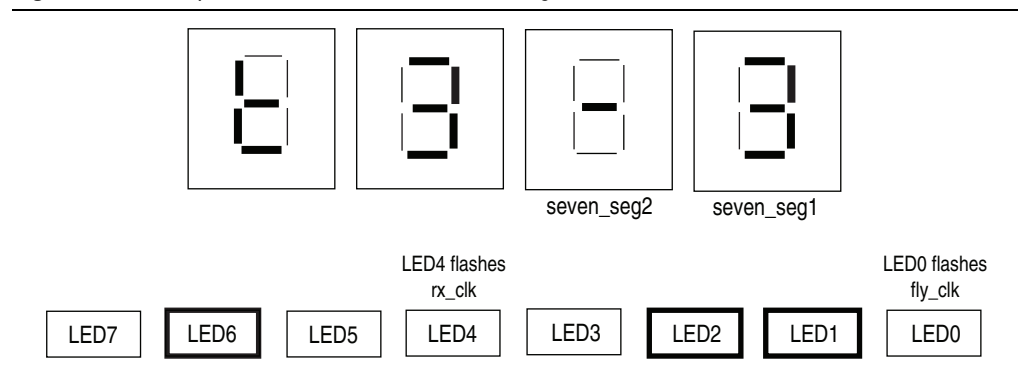
Figure 24. SDI Top-Level Block Continues Generating Data at Previous HD Rate



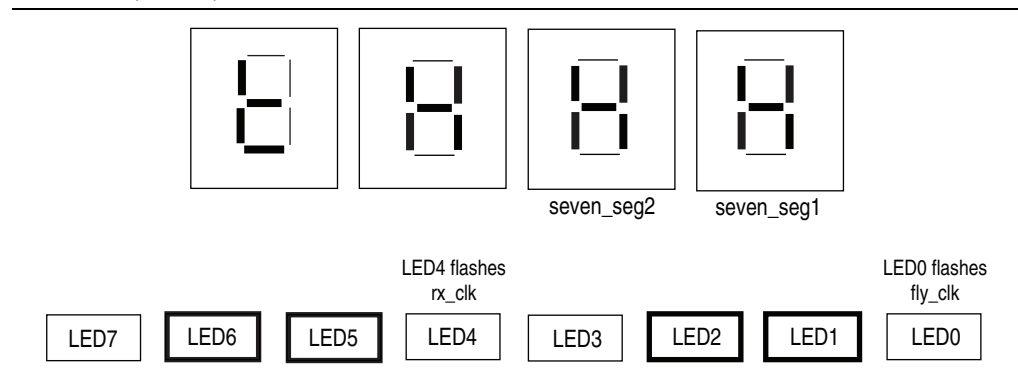
9. Next, reconnect the cable to the BNC J49 connector and wait for the LEDs to appear as shown in [Figure 23](#) again.
10. After a few seconds, the test cycles through to transmit 3G-SDI. Now, the LEDs appear as shown in [Figure 25](#).

Figure 25. Locked to 3G-SDI

11. Remove the cable from the BNC J49 connector and the LEDs appear as shown in [Figure 26](#).

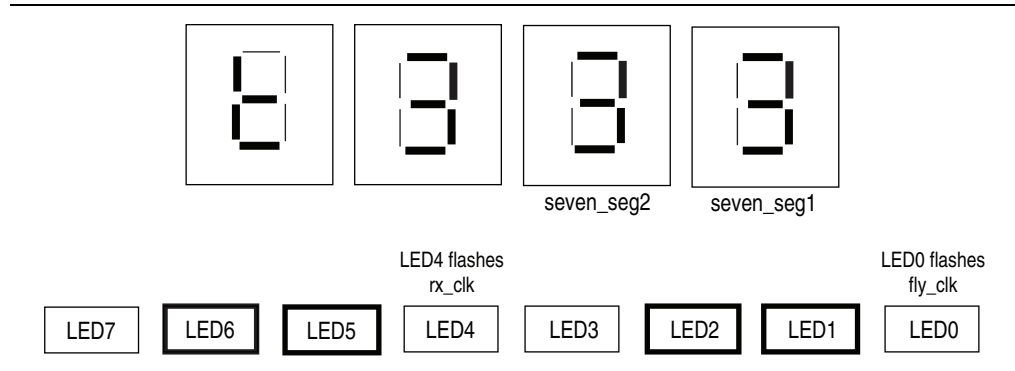
Figure 26. SDI Top-Level Block Continues Generating Data at Previous 3G Rate

12. Next, reconnect the cable to the BNC J49 connector and wait for the LEDs to appear as shown in [Figure 25](#) again.
13. After a few seconds, the test enters the synchronous switching loop (RP168) by first transmitting HD-SDI. Now, the LEDs appear as shown in [Figure 27](#).

Figure 27. (RP 168) Locked to HD-SDI

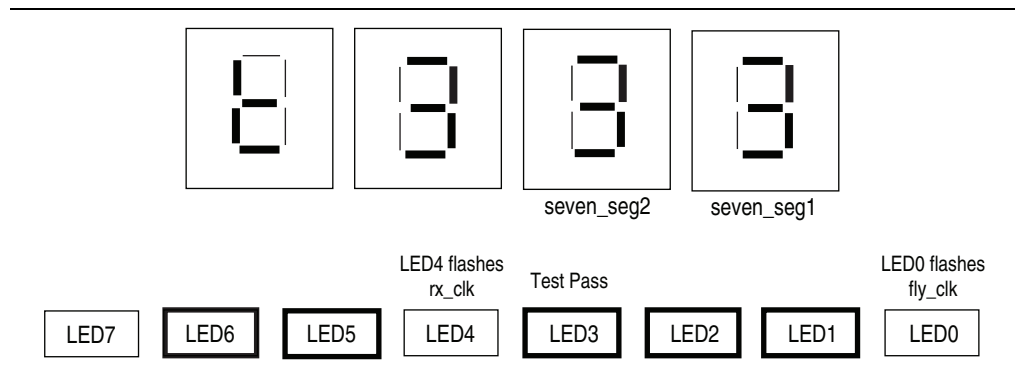
14. The test performs synchronous switching automatically. The `fly_locked` and `frame_locked` signals from the SDI top-level block are monitored continuously during and after the synchronous switching. The test status is recorded and checked at the end of test.
15. The test proceeds to transmit 3G-SDI, and the LEDs appear as shown in [Figure 28](#).

Figure 28. (RP 168) Locked to 3G-SDI



16. Similarly, the test performs synchronous switching automatically. The `fly_locked` and `frame_locked` signals from the SDI top-level block are monitored continuously during and after the synchronous switching. The test status is recorded and checked at the end of test.
17. When the test is successful, the LEDs appear as shown in [Figure 29](#).

Figure 29. Test Pass



Revision History

Table 4 shows the revision history for this application note.

Table 4. Document Revision History

Date and Revision	Changes Made	Summary of Changes
May 2009, version 1.0	Initial Release.	—



101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support
www.altera.com/support

Copyright © 2009 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.



I.S. EN ISO 9001