XAPP680 (v1.0) November 25, 2003

HD-SDI Transmitter Using Virtex-II Pro RocketIO Multi-Gigabit Transceivers

Author: John F. Snow

Summary

The High-Definition Serial Digital Interface (HD-SDI) standard describes how to transport high-definition (HD) digital video serially over video coax cable. HD-SDI is used to connect HD video equipment in broadcast studios and video production centers. It is an evolution of the popular SDI standard that is widely used to transport standard-definition (SD) digital video in the broadcast industry.

The flexibility of RocketIO™ multi-gigabit transceivers available in the Virtex-II Pro™ family devices combined with the programmable logic of Virtex-II Pro FPGAs makes it possible to implement HD-SDI interfaces. Because every Virtex-II Pro FPGA has multiple RocketIO transceivers, it is possible to integrate multiple HD-SDI interfaces into one Virtex-II Pro device along with other video processing functions.

This application note describes the electrical specifications for HD-SDI transmitters and the HD-SDI data format. It also presents several implementation examples and reference designs for an HD-SDI transmitter implemented using the Virtex-II Pro FPGA.

Introduction

Use of HD-SDI, defined by the SMPTE 292M standard, is increasing rapidly in broadcast studios and video production centers as the broadcast industry ramps up support for HDTV broadcasting [Ref 1].

HD-SDI builds upon the widely used SDI standard used to transport SD digital video. It is now common to refer to the older SDI standard as SD-SDI to differentiate it from HD-SDI. The SD-SDI and HD-SDI standards share the same electrical characteristics and encoding scheme. However, HD-SDI uses a higher bit rate to accommodate the higher bandwidth requirements of uncompressed HD digital video signals. Because SD-SDI and HD-SDI share common electrical characteristics, it is possible to build video equipment that can support both standards through a single connection.

This application note discusses the HD-SDI transmitter. A companion application note, XAPP681, describes how to implement the HD-SDI receiver [Ref 2]. Other existing Xilinx application notes cover the SD-SDI standard [Ref 3]. Forthcoming Xilinx application notes will describe how to use the RocketIO transceivers to implement multi-rate capable interfaces supporting both HD-SDI and SD-SDI [Refs 4, 5]. Another related application note describes the implementation of an HD digital video pattern generator [Ref 6].

The HD-SDI standard supports both coax cable and optical fiber interfaces. So far, coax cable is the more popular of the two interfaces due to lower cost and commonality with SD-SDI. This application note only discusses the implementation details for the coaxial cable interface. However, since the data formats and encoding schemes for the optical interface option are identical to the coaxial interface option, the reference designs presented in this application note are directly applicable to implementing an HD-SDI transmitter with an optical fiber interface.

As of this writing, there is a proposal for a new standard, SMPTE 372M, defining a dual-link HD-SDI interface. This proposal uses two HD-SDI interfaces to provide twice the bandwidth, allowing higher bandwidth video formats to be supported. This proposed new standard is not

specifically addressed in this application note, but the HD-SDI reference designs described here can be used as the building blocks for implementing a dual-link HD-SDI interface.

HD-SDI Data Format

This section describes the data format used by HD-SDI, including a discussion of the various video formats that are supported, how those video formats are mapped onto HD-SDI, and how the final bitstream is encoded for transmission over the coax cable.

HD-SDI Supported Video Formats

The SMPTE 292M standard identifies 13 different video formats that can be transported using HD-SDI. Table 1 shows these 13 HD-SDI compatible video formats.

Table 1: **HD-SDI Compatible Video Formats from SMPTE 292M**

SMPTE Standard	260M		295M	274M								296M	
Format Designation	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**
Format[1]	1035i	1035i	1080i	1080i	1080i	1080i	1080p	1080p	1080p	1080p	1080p	720p	720p
Frame Rate (Hz)	30	30/M	25	30	30/M	25	30	30/M	25	24	24/M	60	60/M
Sample Rate (MHz)	74.25	74.25 /M	74.25	74.25	74.25 /M	74.25	74.25	74.25 /M	74.25	74.25	74.25 /M	74.25	74.25 /M
Active Samples/Line & Active Lines/Frame (words x lines)[2]	1920 x 1035	1920 x 1035	1920 x 1080	1920 x 1080	1920 x 1080	1920 x 1080	1920 x 1080	1920 x 1080	1920 x 1080	1920 x 1080	1920 x 1080	1280 x 720	1280 x 720
Total Samples/Line & Total Lines/Frame (words x lines)[2]	2200 x 1125	2200 x 1125	2376 x 1250	2200 x 1125	2200 x 1125	2640 x 1125	2200 x 1125	2200 x 1125	2640 x 1125	2750 x 1125	2750 x 1125	1650 x 750	1650 x 750

1. The format designations follow the industry practice of using the number of active lines per frame plus either the letter “i” indicating interlaced scan or the letter “p” indicating progressive scan. Thus, a format listed as 1080i has 1080 active lines per frame and is interlaced, while a format given as 720p has 720 active lines per frame and is progressive scan.
2. The active samples per line and total samples per line shown are 2-word samples, one word of Y and one word of C. If there are 1920 active samples in a line, then there are 3840 10-bit active words per line after the channels have been interleaved.

As shown in Table 1, some frame rates and sample rates include a divisor called M. The value of M is exactly 1.001. Therefore, the frame rates for the two 1035i formats of SMPTE 260M are 30 Hz and 30 Hz / 1.001 (or approximately 29.97 Hz). The corresponding sample rates for these two standards are 74.25 MHz and 74.25 MHz / 1.001 (approximately 74.1758 MHz), respectively.

The reason for the M divisor is that some of the video formats are defined for true 60 Hz video rates (or 60 Hz derivative rates such as 30 Hz, 25 Hz, and 24 Hz) while others are defined for the 59.94 Hz video rates and derivatives commonly used in North America. The NTSC TV standard used in North America historically has used a field rate of 59.94 Hz, which is carried forward into the HDTV video formats. The major implication for HD-SDI is that there are two different bit rates supported by the HD-SDI standard: 1.485 Gbps and 1.485 / M Gbps.

All video formats in Table 1 are 10-bit 4:2:2 component video. They use the YCbCr color space. The chroma (Cb and Cr) information is sampled at half the rate of the luma (Y) information. Each video sample is defined by two 10-bit words, one luma word and one chroma word. The chroma word of each sample is either a Cb or a Cr word. Consecutive samples alternate between Cb or Cr.

SMPTE recommended practice RP 211 adds five additional “segmented frame” video formats that are compatible also with HD-SDI. RP 211 takes the five SMPTE 274M 1080p standards and defines a segmented frame format corresponding to each of them. These segmented frame video formats have the same number of words and lines and the same frame rates as their 1080p counterparts. The primary difference is that the single progressive frame is divided

or segmented into two fields, consisting of even lines in one field and odd lines in the other. Although this seems to be the same as interlaced video, there is a fundamental difference between interlaced video and segmented frame video. In interlaced video, the source image is rescanned for each field, so the two fields represent images that are separated in time. The two fields of a segmented frame format are both taken from the same progressive scan frame of the image. The two segmented fields represent the same image and are not separated in time.

The motivation behind defining the segmented frame formats was that HD equipment, particularly recorders, capable of handling the 1080p formats was almost non-existent early in the development of HD broadcasting. However, the 1080p format is an ideal format to use as a master format since it is easy to convert 1080p video to all the other video formats without artifacts. The segmented frame formats (1080sF) retain all the information of the 1080p format, but make the actual video stream appear to be an 1080i (interlaced) video stream. Video equipment and recorders supporting 1080i are more available than those supporting 1080p. Since 1080sF looks just like 1080i, a video recorder designed for 1080i can record the 1080sF video format.

Table 2 shows the five segmented frame video formats defined by SMPTE RP 211. The last row of the table shows the 1080p formats from Table 1 that correspond to each 1080sF format. In terms of data format, 1080sF formats with frame rates of 30, 30 / M, and 25 are indistinguishable from the 1080i formats D, E, and F, respectively. The 1080sF formats with frame rates of 24 and 24 / M have no “look-alike” 1080i formats.

Table 2: **Segmented Frame Video Formats from RP 211**

Format	1080sF				
Frame Rate (Hz)	30	30/M	25	24	24/M
Sample Rate (MHz)	74.25	74.25/M	74.25	74.25	74.25/M
Active Samples/Line & Active Lines/Frame (words x lines)	1920 x 1080	1920 x 1080	1920 x 1080	1920 x 1080	1920 x 1080
Total Samples/Line & Total Lines/Frame (words x lines)	2200 x 1125	2200 x 1125	2640 x 1125	2750 x 1125	2750 x 1125
Corresponding 1080p Format in Table 1	G	H	I	J	K
“Look-alike” 1080i Format in Table 1	D	E	F	None	None

HD-SDI Data Format

The high-definition video formats supported by HD-SDI keep the luma and chroma information in separate channels. The parallel video interfaces for these video standards carry the Y and C channels on separate wires. HD-SDI treats these two channels separately, but then interleaves them before encoding is done.

Each video line sent over an HD-SDI interface is divided into four areas as shown in Figure 1. The line begins with an EAV (end of active video) timing reference. The EAV includes the line number of the current line and a CRC value for the previous line. Following the EAV is the horizontal blanking interval. The horizontal blanking interval contains either blanking level video or ancillary data carrying information such as embedded digital audio. Following the horizontal blanking interval is the SAV (start of active video) timing reference followed immediately by the active portion of the line. The active portion of the line contains active video samples (two words per sample) if the line is an active line or it contains blanking level or ancillary data if the line is part of the vertical blanking interval.

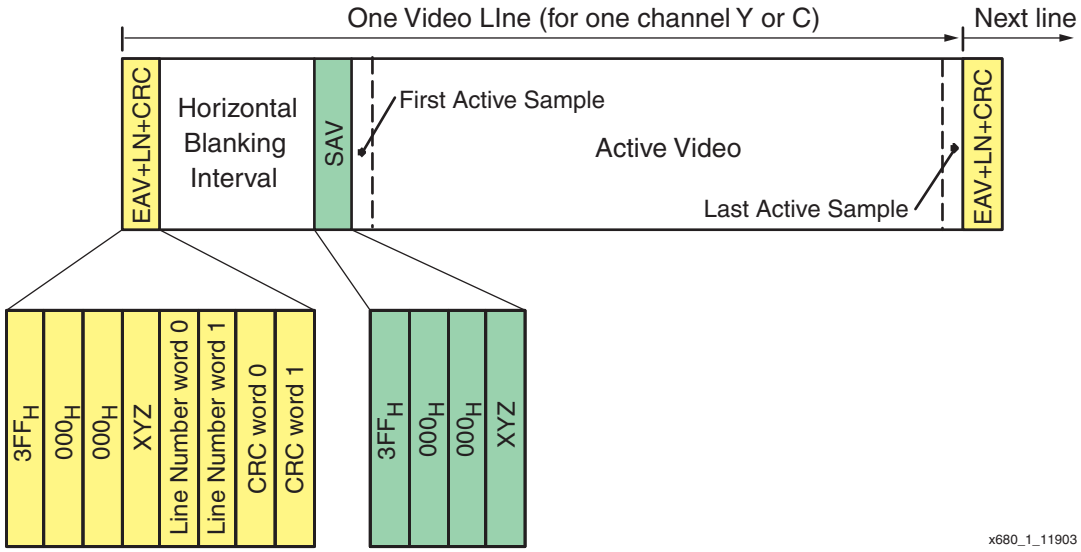

Figure 1: **HD-SDI Line Format**

Figure 1 shows just one of the two channels, either Y or C. Each channel has its own EAV, horizontal blanking, SAV, and active areas. The two channels are considered to be synchronous. For example, the Y word of sample zero is present on the Y channel at the same time that the C word of sample zero is present on the C channel.

The number of words in the horizontal blanking interval and the active portions of each video line are dependent upon the video format being transported. Refer to Table 1 for the number of words in these two regions for any supported video format.

The formats of the EAV and SAV timing references are shown in Figure 1. The first three words of these timing references are always fixed values: $3FF_H$, 000_H, 000_H. This sequence of three words is unique in the video stream and can occur only at the beginning of an EAV or SAV. The fourth word of the timing reference is commonly called the XYZ word and contains various flags that are used to describe the attributes of the current line. One bit in the XYZ word (the H bit) distinguishes between EAV and SAV. The format of the XYZ word is shown in Figure 2.

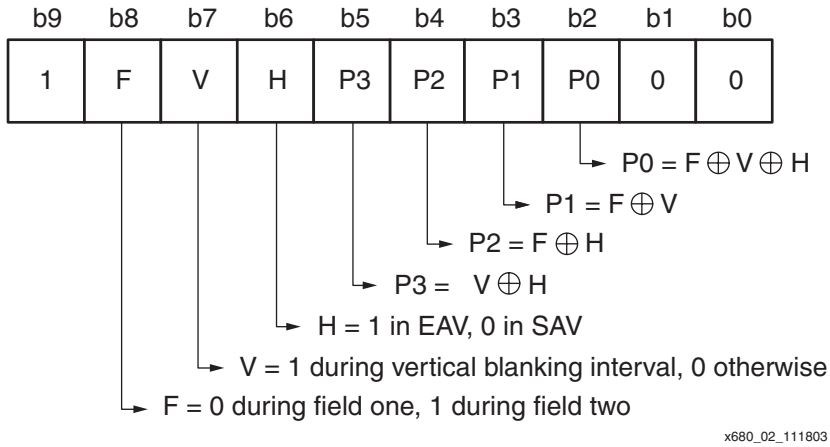

Figure 2: **XYZ Word Format**

The EAV timing reference is followed by a two-word line number field called LN. The line number, required by HD-SDI, is used by the receiving equipment to synchronize more quickly to the video stream. The 11-bit line number value is encoded into the two 10-bit words as shown in Figure 3. Note that each channel, Y and C, has separate LN fields. The values of the line number for both Y and C should always be the same. Line numbers are assigned

sequentially beginning with 1. The definition of which line in the frame is line 1 is video format dependent as specified in the SMPTE standard for each video format.

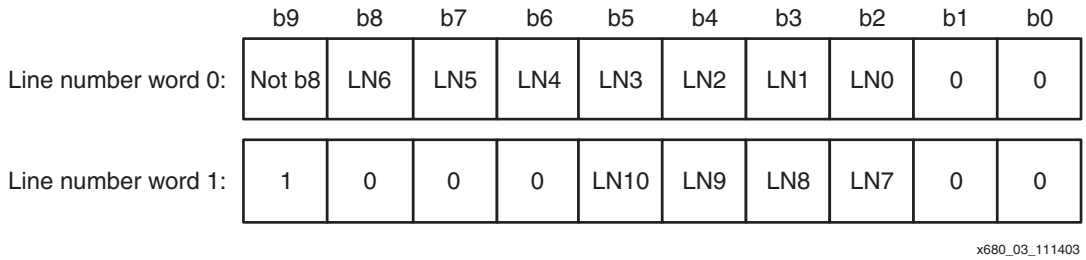

Figure 3: **Line Number Format**

Two words that immediately follow the second LN word contain a cyclic redundancy code (CRC) for the previous line. The HD-SDI receiver uses the CRC value to detect transmission errors. As with the LN field, the Y and C channels each have their own CRC values. The CRC values for the two channels are calculated separately. Each CRC value is calculated using the following CRC polynomial:

$$CRC = X^{18} + X^5 + x^4 + 1$$

To calculate the CRC for a line, the initial CRC value is cleared to zero then the CRC is formed by including the first word of the active video portion of the line through and including the second word of the LN field. Figure 4 shows the words included in each CRC calculation. Note that the CRC calculation does not include the two CRC words, the horizontal blanking interval, or the SAV. The SAV has its own parity bits that can be used to detect a corrupted SAV. Ancillary data packets inserted into the horizontal blanking interval usually have a CRC or checksum for each packet.

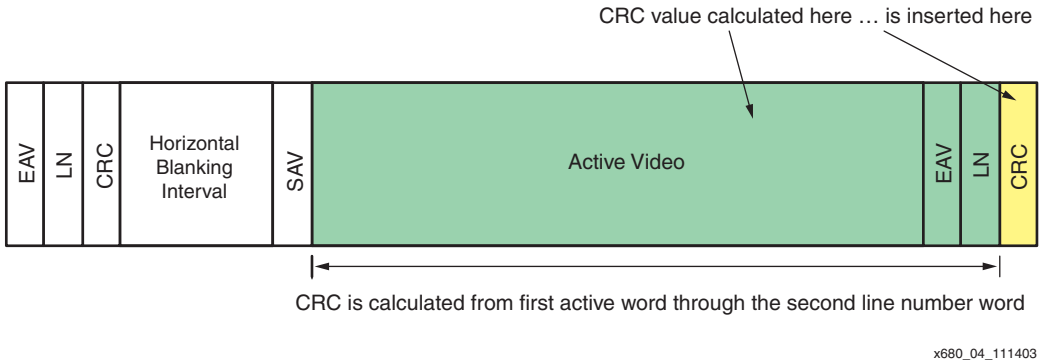

Figure 4: **CRC Calculation**

Figure 5 shows how the 18-bit CRC value is formatted into two 10-bit words when included into the HD-SDI video stream.

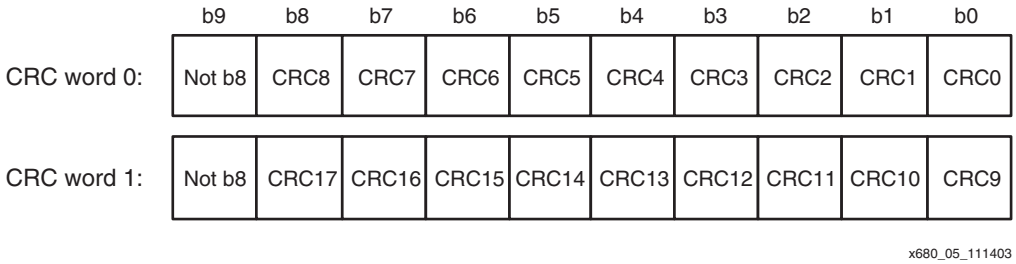

Figure 5: **CRC Format**

Channel Interleaving

After the line number and CRC fields are inserted into the two channels and prior to encoding the data, the Y and C channels are interleaved together as shown in Figure 6. For each sample, the word from the C channel is always sent first followed by the word from the Y channel.

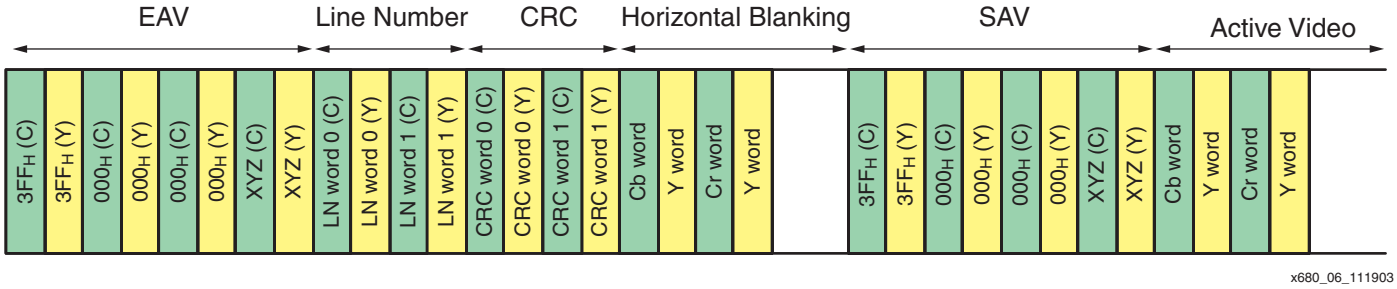

Figure 6: **Interleaved Data Stream**

Encoding

The two-stage encoding scheme used by HD-SDI is identical to the one used by SD-SDI. First, a pseudo-random scrambler is used to scramble the data. Next, the non-return to zero (NRZ) data from the scrambler is converted to non-return to zero inverted (NRZI). The pseudo-random scrambler is very much like a linear feedback shift register (LFSR). The polynomial implemented by the scrambler is shown below as G_1, and the polynomial implemented by the NRZ-to-NRZI converter is shown as G_2:

$$G_1(X) = X^9 + X^4 + 1$$

$$G_2(X) = X + 1$$

NRZI bitstreams, such as HD-SDI, have an interesting property: they are polarity free. This means that the HD-SDI bitstream can be inverted between the transmitter and the receiver, and the receiver still correctly receives and decodes the data.

Figure 7 shows conceptually how the encoder would work if implemented serially. The boxes represent flip-flops and the ⊕ symbols represent XOR gates. In the reference designs given in this application note, the SDI encoder is implemented in a parallel fashion, encoding 20 bits per clock cycle.

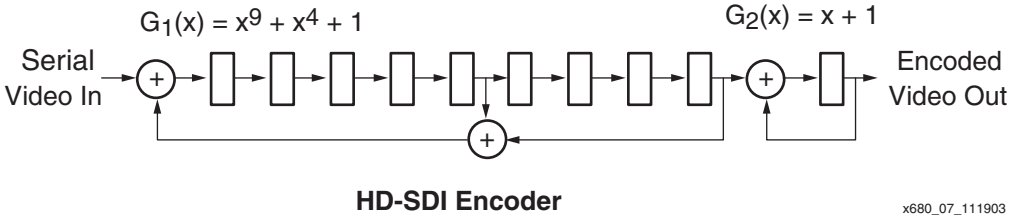

Figure 7: **HD-SDI Encoder Serial Implementation**

HD-SDI Transmitter Requirements

This section describes the electrical requirements for the HD-SDI coaxial interface.

Electrical Requirements

The SMPTE 292M HD-SDI coaxial interface specifies the use of 75-ohm coax cable with BNC connectors. The BNC connectors used for HD-SDI usually have an impedance of 75 ohms, although the specification does permit the use of 50-ohm connectors. When 75-ohm BNC connectors are used, they must be mechanically compatible with the 50-ohm BNC connector type defined by IEC 60169-8 [Ref 7].

The HD-SDI transmitter is unbalanced (singled-ended) with a source impedance of 75 ohms. The transmitter must have a return loss of at least 15dB over a frequency range of 5 MHz to 1.485 GHz. Return loss is a measurement of the amount of signal that is absorbed by the transmitter when a reflected wave reaches the transmitter. The higher the return loss, the lower the amount of the reflected wave that is, in turn, reflected back to the receiver.

The transmitter signal amplitude is required to be 800mV ±10%. The DC offset must be 0.0V ±0.5V. This DC offset requirement implies that the transmitter will be AC coupled to the coax cable.

The rise and fall times (20% to 80% amplitude) are required to be no slower than 270 ps. They cannot differ from each other by more than 100 ps. The overshoot of the rising and falling edges cannot exceed 10% of the amplitude of the signal.

Jitter Requirements

The SMPTE 292M specification places certain output jitter requirements on the HD-SDI transmitter. SMPTE 292M separates jitter into two bands called timing jitter and alignment jitter. Timing jitter includes all jitter frequency components from 10 Hz to 148.5 MHz. Alignment jitter is high-frequency jitter and begins at 100 kHz and goes to 148.5 MHz. As shown in Figure 8, both the timing and alignment jitter pass bands roll off at 20 dB / decade above and below the frequency limits just mentioned. Note that alignment jitter is a subset of the timing jitter.

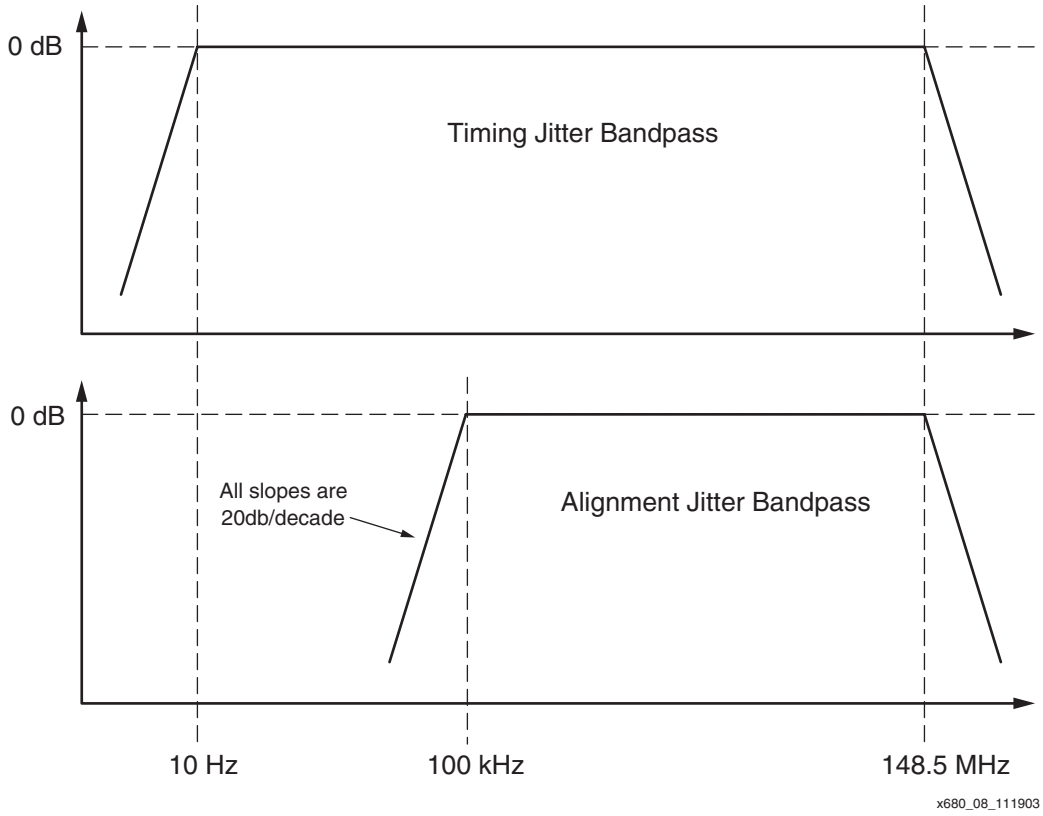

Figure 8: **HD-SDI Transmitter Output Jitter Bands**

The HD-SDI specification requires that the transmitter must have a timing jitter output of less than 1.0 UI(1). Also, the total peak-to-peak alignment jitter must be less than 0.2 UI.

1. The term UI stands for Unit Interval and is equal to one bit period. For an HD-SDI interface running at 1.485 Gbps, 1.0 UI is equal to about 673 ps.

Usually, video equipment with HD-SDI interfaces have output jitter specifications well below the jitter requirements given in the HD-SDI spec, which allows for more jitter margin in the complete HD-SDI link. Good quality HD-SDI transmitters have timing jitter specifications of under 0.15 UI and alignment jitter specifications approaching 0.1 UI.

Cable Driver

The HD-SDI coaxial interface is single-ended, not differential. The RocketIO output driver is a differential driver. Trying to use just one side of the RocketIO output to drive the cable, even if the unused side is terminated to a 75-ohm load, probably will not result in acceptable results. Trying to meet all the other electrical requirements of the HD-SDI transmitter, such as return loss and rise and fall times, by driving the coax cable directly from the RocketIO can be challenging. The RocketIO transmitter was designed to drive PCB traces that are, at most, about one meter long, whereas the HD-SDI specification allows for coax cable lengths of up to 100 meters. Therefore, Xilinx recommends the use of a cable driver specifically designed to meet the HD-SDI standards to interface the RocketIO output to the video coax cable.

The Gennum GS1528 cable driver is an example of such an HD-SDI compliant cable driver. The GS1528 can meet the requirements of both SD-SDI and HD-SDI. It has a rate control input that dynamically selects between HD and SD modes. This control signal changes the slew rate of the cable driver since the rise and fall time requirements of SD-SDI differ from those of HD-SDI.

This GS1528 cable driver has 3.3V LVPECL differential inputs. The CML output of the RocketIO transceiver is a 2.5V driver and is not LVPECL compatible. AC coupling must be used between the RocketIO driver and the input of the GS1528 in order to shift the signal levels from the 2.5V CML levels to the LVPECL levels. Figure 9 shows an example of interfacing the GS1528 to a RocketIO transmitter. Large AC coupling capacitor values, in the range of 1 µF to 4.7 µF, must be used in order to successfully pass the worst-case waveforms that can be generated by the HD-SDI and SD-SDI encoders. The rate control input pin of the GS1528 must be Low for HD-SDI. The RocketIO transceiver's transmitter termination voltage (VTTX) should be set to 2.5V. The output impedance of the transceiver should be set to 50 Ω.

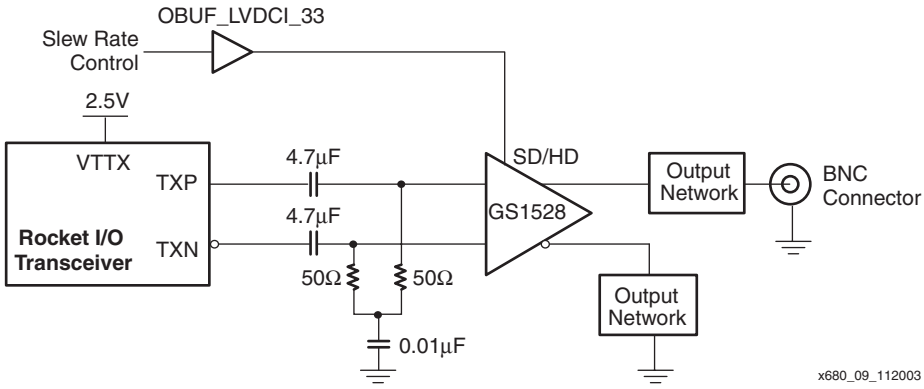

Figure 9: **Interfacing the GS1528 Cable Driver to the RocketIO Transmitter**

Figure 9 shows only the interface between the RocketIO transceiver and the GS1528. Consult the GS1528 data sheet for details on connecting the GS1528 to the BNC connector [Ref 8].

Meeting the 15dB return loss requirement on the output of the cable driver requires careful layout of the cable driver and the output network between the cable driver and the BNC connector. Xilinx recommends that you carefully follow the recommendations from the cable driver manufacturer.

Clocks

One of the most important aspects of implementing an HD-SDI transmitter using the RocketIO transceivers is providing the right clocks to the transceivers. The RocketIO transceiver requires two types of clocks, *reference* clocks and *user* clocks. The reference clocks are used to generate the bit-rate clock for the serializer. The user clocks are used to clock data from the fabric of the FPGA into the RocketIO transceiver.

The following sections contain descriptions of the clocking requirements of the RocketIO transceivers oriented towards implementing HD-SDI interfaces. More details about the clocking requirements of the RocketIO transceivers can be found in the *RocketIO Transceiver User Guide* [Ref 9].

For most HD-SDI transmitter applications, both the reference clock and user clocks are all derived from the parallel video clock running at either 74.25 MHz or 74.1758 MHz.

Reference Clocks

The reference clocks provide low-jitter frequency reference sources for the RocketIO transceiver. The RocketIO transceiver multiplies the selected reference clock by 20 to obtain a bit-rate clock for the transmitter's serializer. Any jitter below about 10 MHz present on the reference clock is passed through to the transmitter output with little attenuation. Thus it is very important to use a low-jitter reference clock source.

As shown in Figure 10, each RocketIO transceiver has four reference clock inputs, although only one reference clock can be selected at a time. The four reference clocks are BREFCLK, BREFCLK2, REFCLK, and REFCLK2. Inside every RocketIO transceiver is a set of MUXes that select one of these four reference clocks. The MUXes are controlled by two signals:

- The REF_CLK_V_SEL signal is a configuration attribute of the RocketIO transceiver and can be changed only by reconfiguration.
- The REFCLKSEL signal is an input port to the RocketIO transceiver and can be controlled dynamically.

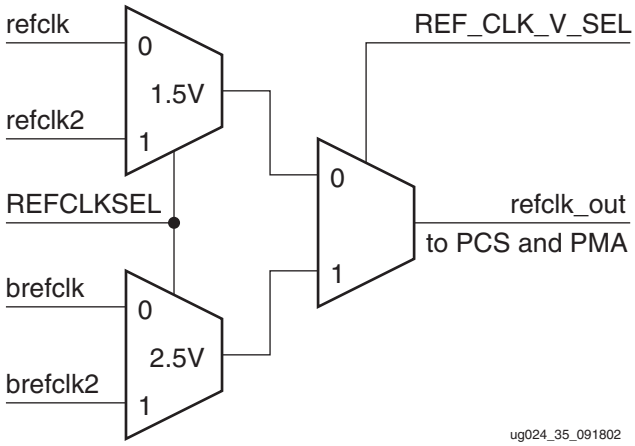

Figure 10: **REFCLK MUXes**

Keep in mind that although there are four reference clock inputs, you can switch only between two of them without reconfiguring the RocketIO transceiver. At configuration time, you must choose between using the REFCLKs or the BREFCLKs.

There is a significant difference between the REFCLKs and the BREFCLKs. The BREFCLK inputs provide the lowest jitter input path for the reference clocks, but at the cost of limited flexibility. The REFCLK inputs offer more flexibility, but at the cost of increased jitter.

The BREFCLK inputs use specific pins on the FPGA, and they can only use 2.5V differential I/O standards. The two BREFCLKs have dedicated routing resources inside the FPGA from the IOBs to the RocketIO transceivers. Every Virtex-II Pro device has one BREFCLK input and one

BREFCLK2 input on the top edge of the FPGA, and these inputs are only connected to the RocketIO transceivers along the FPGA's top edge. Likewise, the bottom edge of the FPGA has one BREFCLK input and one BREFCLK2 input, and these inputs are connected only to the RocketIO transceivers on the bottom edge of the FPGA.

On the other hand, the REFCLK inputs do not use specific FPGA pins. They are connected through programmable routing resources inside the FPGA. The REFCLKs can be used only when running the RocketIO transceivers at rates less than 2.5 Gbps. The BREFCLK inputs must be used for bit rates above 2.5 Gbps, but can be used at slower bit rates.

Because HD-SDI has two different bit rates, the RocketIO transceiver must have reference clocks for each of the supported bit rates. A reference clock source, external to the FPGA, must provide both 74.25 MHz and 74.25 / 1.001 MHz reference clock frequencies to the FPGA if both HD-SDI bit rates are supported. These two frequencies could come into the FPGA as one signal with something external to the FPGA selecting between them, or they could come into the FPGA as two separate clock sources, in which case the RocketIO reference clock MUX would be used to select between them.

It is very important that the reference clock has very low jitter. If this clock has too much jitter, the output jitter of the RocketIO transceivers might exceed the HD-SDI specifications. At HD-SDI speeds, Xilinx recommends that the peak-to-peak jitter of the reference clock should be less than 100 ps. On the Xilinx SDV demo board, the used reference clock sources had less than 40 ps peak-to-peak jitter.

If the reference clock MUX in the RocketIO is used to select between the two HD-SDI reference clock frequencies, you are highly recommended to use the BREFCLK inputs rather than the REFCLK inputs. Because the two HD-SDI reference clock frequencies differ by just 74.25 kHz, our experience has shown that these two clock sources tend to mix (heterodyne) if the REFCLK inputs are used. This results in excessive output jitter on the transmitter output with a large jitter component at 74.25 kHz. However, if the BREFCLK inputs are used, this mixing does not occur and the output jitter is much lower.

If only a single reference clock input is required for the RocketIO transceiver, such as when an external frequency synthesizer generates the reference clocks, then either the BREFCLK or REFCLK inputs can be used and will produce about the same amount of output jitter.

One further recommendation is to use clock sources with differential LVDS or LVPECL outputs. The use of differential I/O standards results in significantly less jitter than when single-ended I/O standards are used for the reference clocks. In fact, the BREFCLK inputs must be differential.

User Clocks

The user clocks load data into the RocketIO transceiver from the fabric of the FPGA. Each RocketIO transceiver requires two user clocks on the transmitter side called TXUSRCLK and TXUSRCLK2. Each RocketIO transceiver also has two user clocks for the receiver called RXUSRCLK and RXUSRCLK2. If the receiver is not used, RXUSRCLK and RXUSRCLK2 still should be driven with valid clock signals. The easiest thing to do in this case is to connect RXUSRCLK to TXUSRCLK and RXUSRCLK2 to TXUSRCLK2.

TXUSRCLK always must be frequency-locked to the selected reference clock. There is no required phase relationship between TXUSRCLK and the selected reference clock, but they must have the same frequency.

The frequency and phase relationships between TXUSRCLK and TXUSRCLK2 depend on the width of the TXDATA input port of the RocketIO transceiver. For HD-SDI, it is usually most convenient to use a 20-bit wide TXDATA port because this matches the data word width of HD-SDI (10 bits of Y and 10 bits of C). When using a 20-bit TXDATA port, TXUSRCLK2 must have the same frequency and phase as TXUSRCLK (simply connect TXUSRCLK and TXUSRCLK2 to the same clock source). Consult the *RocketIO Transceiver User Guide* for TXUSRCLK2 requirements when other TXDATA port widths are used.

The user clocks do not have to be low jitter. They are often connected to outputs of digital clock managers (DCMs) due to frequency and phase relationship requirement between TXUSRCLK and TXUSRCLK2 when using TXDATA port widths other than 20 bits.

When using two reference clock sources for HD-SDI, it is necessary to change the frequency of the TXUSRCLK and TXUSRCLK2 inputs when the reference clock source is changed. Figure 11 shows an example of the RocketIO clock connections when a single reference clock input is used and jitter reduction is being done on the reference clock by an external PLL. Figure 12 shows how to connect the reference clocks and user clocks when two clock sources are used.

Jitter Reduction

In many cases, the parallel video clock associated with the input parallel video stream might have too much jitter to be used directly as a reference clock to the RocketIO transceiver. For example, the SMPTE parallel interface standards for HD video, such as SMPTE 274M, are allowed to have jitter as high as 2 ns peak-to-peak on the parallel video clock. This is more than 10 times the acceptable amount of jitter for the reference clocks for the RocketIO transceiver. The SMPTE 292M standard specifically warns that jitter reduction must be implemented on this type of high-jitter parallel video interface before the video can be transmitted using HD-SDI.

Also jitter reduction is usually required when connecting the output of an HD-SDI receiver to an HD-SDI transmitter. The HD-SDI receiver recovers a clock from the incoming HD-SDI bitstream. Since the input bitstream might have a considerable amount of jitter after travelling down a long coax cable, the recovered clock also might have a great deal of jitter. It usually is not possible to directly use the recovered clock from receiver as the reference clock to the transmitter without reducing the jitter on the recovered clock.

Figure 11 shows a typical implementation of jitter reduction. The high-jitter parallel video clock passes through a PLL that is optimized for the desired jitter reduction characteristics. The high-jitter video data is written into an asynchronous FIFO using the high-jitter clock. Then the data is read out of the FIFO using the low-jitter clock from the PLL, thereby resynchronizing the data to the low-jitter clock.

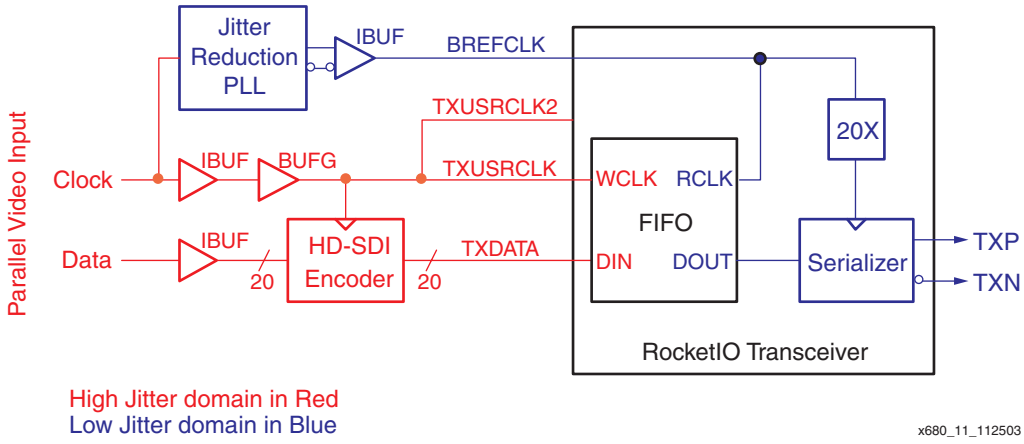

Figure 11: **Jitter Reduction**

The RocketIO transmitter has a shallow FIFO on its input data path that is perfectly suited for implementing the asynchronous FIFO function shown in Figure 11. However, the Virtex-II Pro FPGA does not contain a PLL that can be used for clock jitter reduction. Thus, it is necessary to use an external PLL to implement jitter reduction on the video clock in those cases where jitter reduction is required. Any PLL circuit used for jitter reduction must produce an output that has 100 ps peak-to-peak of jitter, at most.

Reference Designs

The reference design for this application note can be downloaded from xapp680.zip. The reference design is described at a top level in the following section. Detailed information about the reference design can be found in "Appendix: Reference Design Details."

Most of the HD-SDI transmitter is contained in the module called hdsdi_tx. This module contains all of the formatting and encoding logic necessary for the HD-SDI transmitter, but it does not contain the RocketIO transceiver. The RocketIO transceiver is contained in another module called hdsdi_rio. The reason for keeping the RocketIO module separate from the rest of the HD-SDI transmitter is so that the receiver side of the transceiver can also be connected to an HD-SDI receiver module, if needed.

An example of how to connect the hdsdi_tx and hdsdi_rio modules is given in the file named sdv_hdsdi_tx. This HD-SDI transmitter example was designed specifically for the Xilinx SDV demo board. In this example, the video source for the HD-SDI transmitter comes from a video pattern generator module called multigenHD. The multigenHD module is from Xilinx application note XAPP682. Figure 12 is a block diagram of the sdv_hdsdi_tx design.

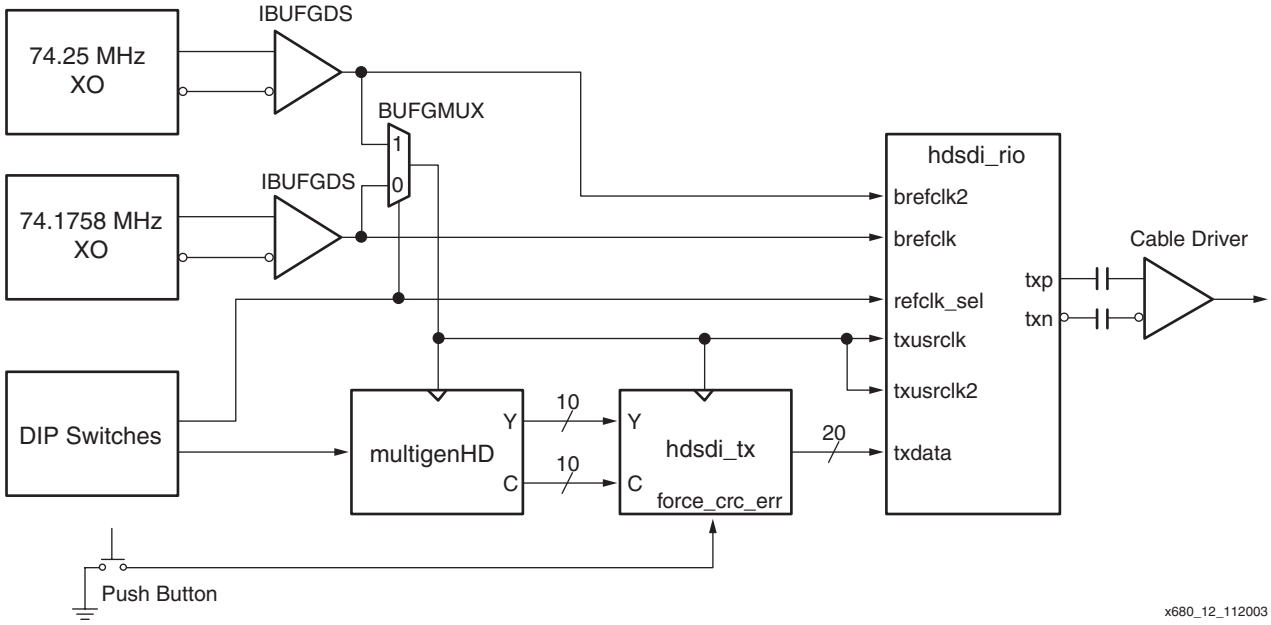

Figure 12: **Xilinx SDV Board HD-SDI Transmitter Reference Design**

Figure 13 shows a block diagram of the main HD-SDI transmitter module, hdsdi_tx. This module contains three submodules described in the following paragraphs.

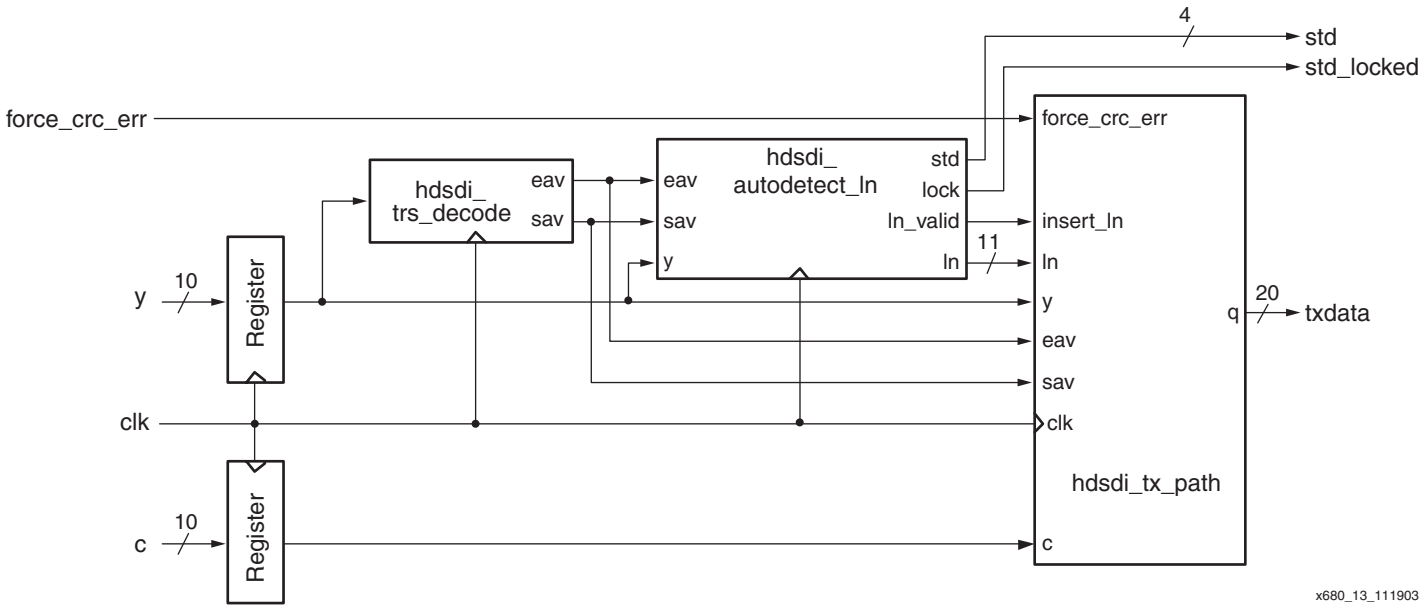

Figure 13: **hdsdi_tx Block Diagram**

The hdsdi_trs_decode module examines the incoming video stream and identifies when EAV and SAV timing references occur. The EAV and SAV signals from this module are used by the other modules in the HD-SDI transmitter for such things as video format detection, line number insertion, and CRC generation and insertion.

The hdsdi_autodetect_ln module examines the incoming video stream and determines the format of the video. This module can identify all video formats listed in Table 1 and Table 2. Once the module determines the format of the video, it begins generating line number information that can be inserted into the video as required by the HD-SDI standard. This module does not insert the line number information. Insertion is done in the hdsdi_tx_path module. The hdsdi_autodetect_ln module is optional. If the video already contains line number information, then this module is not required.

Note that in the sdv_hdsdi_tx reference design, the multigenHD video pattern generator produces line numbers but does not insert them into the video. The multigenHD module also provides essentially the same video timing information that hdsdi_trs_decode provides. The hdsdi_trs_decode and hdsdi_autodetect_ln modules really are not necessary when the video is provided by a module such as multigenHD. However, these modules are included in this reference design to illustrate how they are used when video timing and line number information is not available on the input of the HD-SDI transmitter.

The third module in hdsdi_tx is called hdsdi_tx_path (see Figure 14). This module contains the main data path of the HD-SDI transmitter. In a design where line number information and video timing is provided at the input to the HD-SDI transmitter, the hdsdi_autodetect_ln and hdsdi_trs_decode modules could be eliminated and only hdsdi_tx_path would be required.

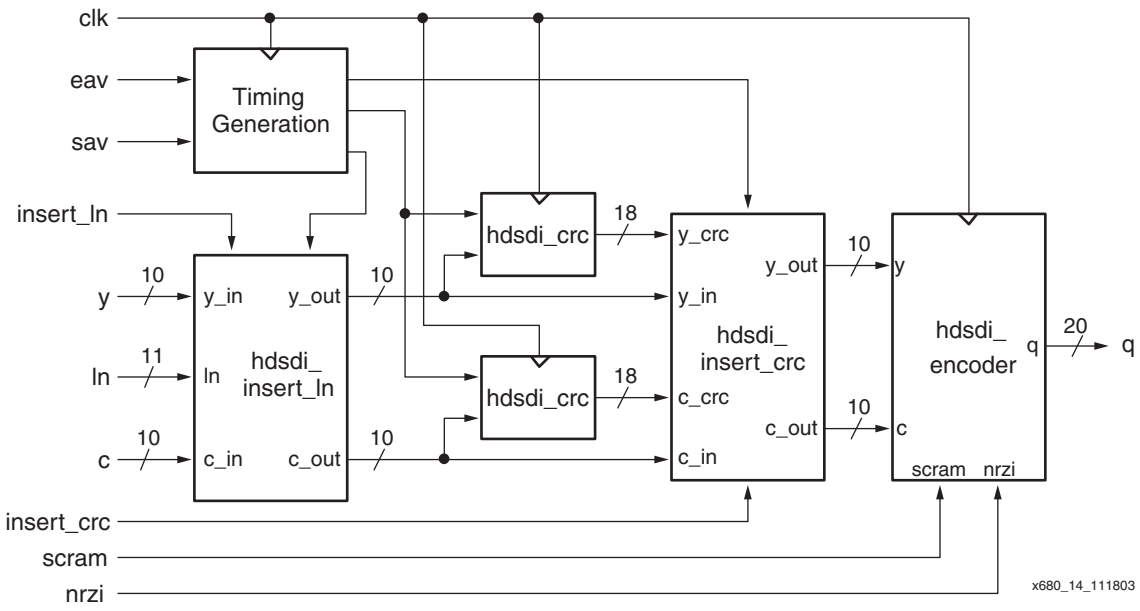

Figure 14: **hdsdi_tx_path Block Diagram**

The hdsdi_tx_path module performs four main functions: line number insertion, CRC generation, CRC insertion, and encoding. The line number information is formatted and inserted into the Y and C channels by the hdsdi_insert_ln module. Once line number information is inserted, a CRC is computed for each channel by the hdsdi_crc modules, and then the CRCs are formatted and inserted into the Y and C channels by the hdsdi_insert_crc module. After CRC insertion, the video is encoded by the hdsdi_encoder module. After encoding, the data is sent to the hdsdi_rio module for transmission.

Figure 15 shows the block diagram of the hdsdi_rio module. This module is wrapped around the RocketIO transceiver primitive (GT_CUSTOM). In addition to the RocketIO primitive, the module contains bit swap functions on the input and output data ports of the RocketIO primitive and a reset delay circuit for the RocketIO transceiver.

The RocketIO transceiver transmits the MSB of the TXDATA port first. Likewise, the RocketIO receiver outputs the first bit it receives on the MSB of the RXDATA output port. HD-SDI always transmits the LSB first, just the opposite of how the RocketIO transceiver operates. The 20-bit output of the HD-SDI encoder must be bit swapped before being connected to the TXDATA port of the RocketIO primitive so that the LSB of the encoder is connected to the MSB of the RocketIO transceiver's TXDATA port. The RocketIO receiver's RXDATA output port also is bit swapped before leaving the hdsdi_rio module.

The TXRESET input of the GT_CUSTOM primitive must remain asserted for at least two TXUSRCLK cycles after all clock inputs become stable. The hdsdi_rio primitive contains some logic to keep the TXRESET input asserted until several clock cycles after the dcm_locked input becomes asserted. This input is called dcm_locked because it usually is driven from the LOCKED output of the DCM generating the TXUSRCLK and RXUSRCLK clock inputs to the RocketIO transceiver. If a DCM is not used to generate these clock signals, then the dcm_locked input can be connected to another appropriate signal that indicates when the clocks are stable or it can be tied High if the clocks are always running.

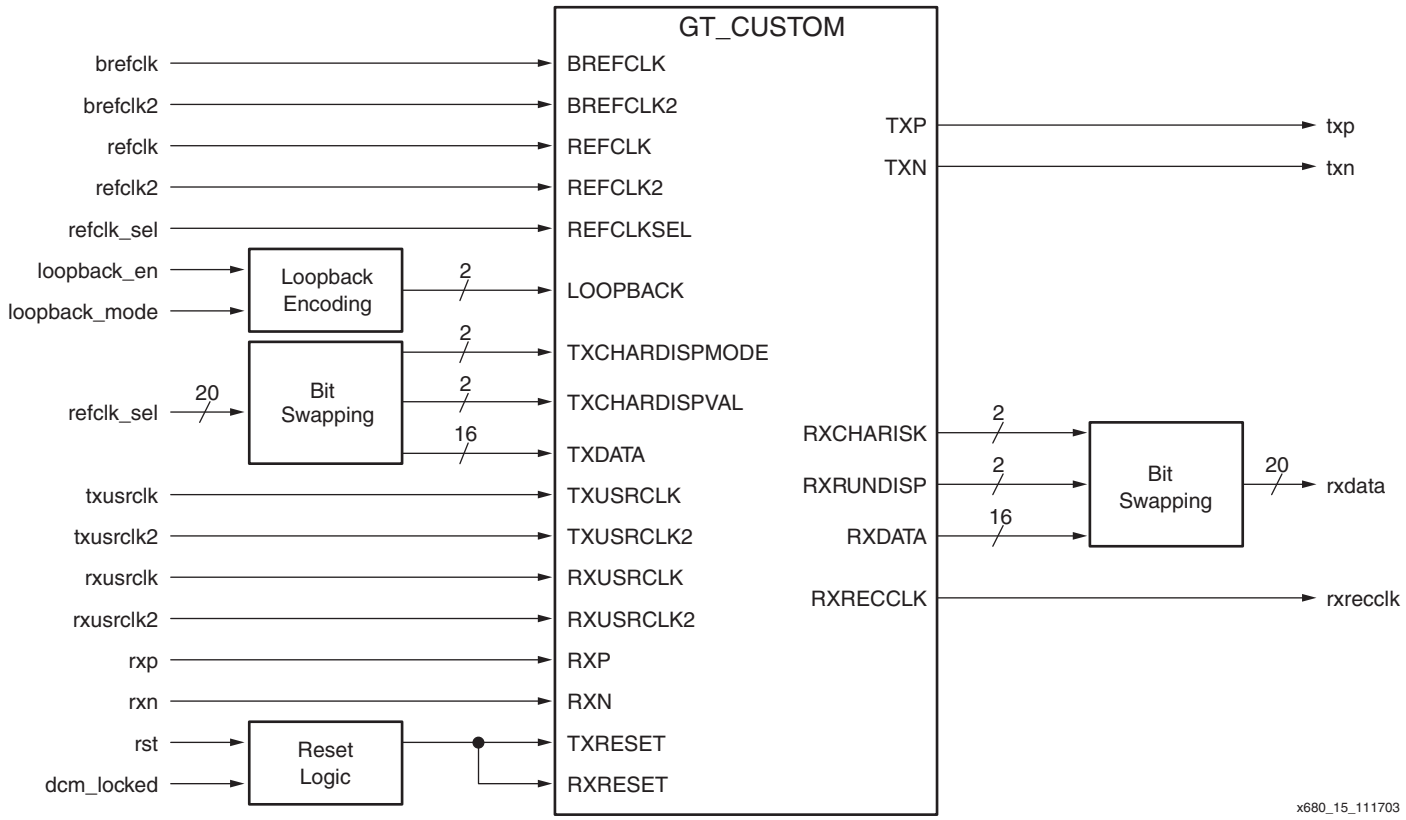

Figure 15: **hdsdi_rio Module**

Jitter Performance

The output jitter of the HD-SDI transmitter is dependent upon the amount of jitter present on the reference clock, noise coupled into the reference clock from other sources (such as the other reference clock input), the intrinsic jitter of the RocketIO transceiver, and jitter added by the cable driver and board layout.

The Xilinx SDV demo board [Ref 10] was used to measure the typical transmitter output jitter of an HD-SDI transmitter implemented using the Virtex-II Pro device. This board uses low-jitter crystal oscillators for reference clocks to the RocketIO transceiver. The demo board uses a Gennum GS1528 cable driver to interface the RocketIO transceiver to the BNC connector.

The typical HD-SDI transmitter jitter values that were measured on the SDV demo board were verified on over 30 different boards using several different HD-SDI waveform analyzers including the Tektronix WFM700 and the SyntheSys Research HD292 and MVA3000 analyzers. Figure 16 is the eye diagram display from a SyntheSys Research MVA3000 analyzer showing the output waveform and jitter measurements from a Xilinx SDV demo board running the HD-SDI transmitter reference design.

All jitter measurements were made with the SDV demo board connected to the analyzer using a 1-meter long, 75-ohm coax cable. The HD-SDI transmitter was sending a 75% color bar pattern. The jitter values reported in Table 3 were the worst numbers measured across all 30 boards tested under room temperature conditions and nominal voltage conditions. These jitter numbers have been verified even when the FPGA was filled so that about 95% of the fabric, block RAM, and multipliers were being toggled actively and random data was output on about 50% of the IOBs. The reference clocks to the RocketIO transceiver had no more than 40 ps peak-peak of jitter.

Table 3: **Typical HD-SDI Transmitter Output Jitter Values**

Jitter Type	Value
Timing Jitter	0.133 UI
Alignment Jitter	0.107 UI

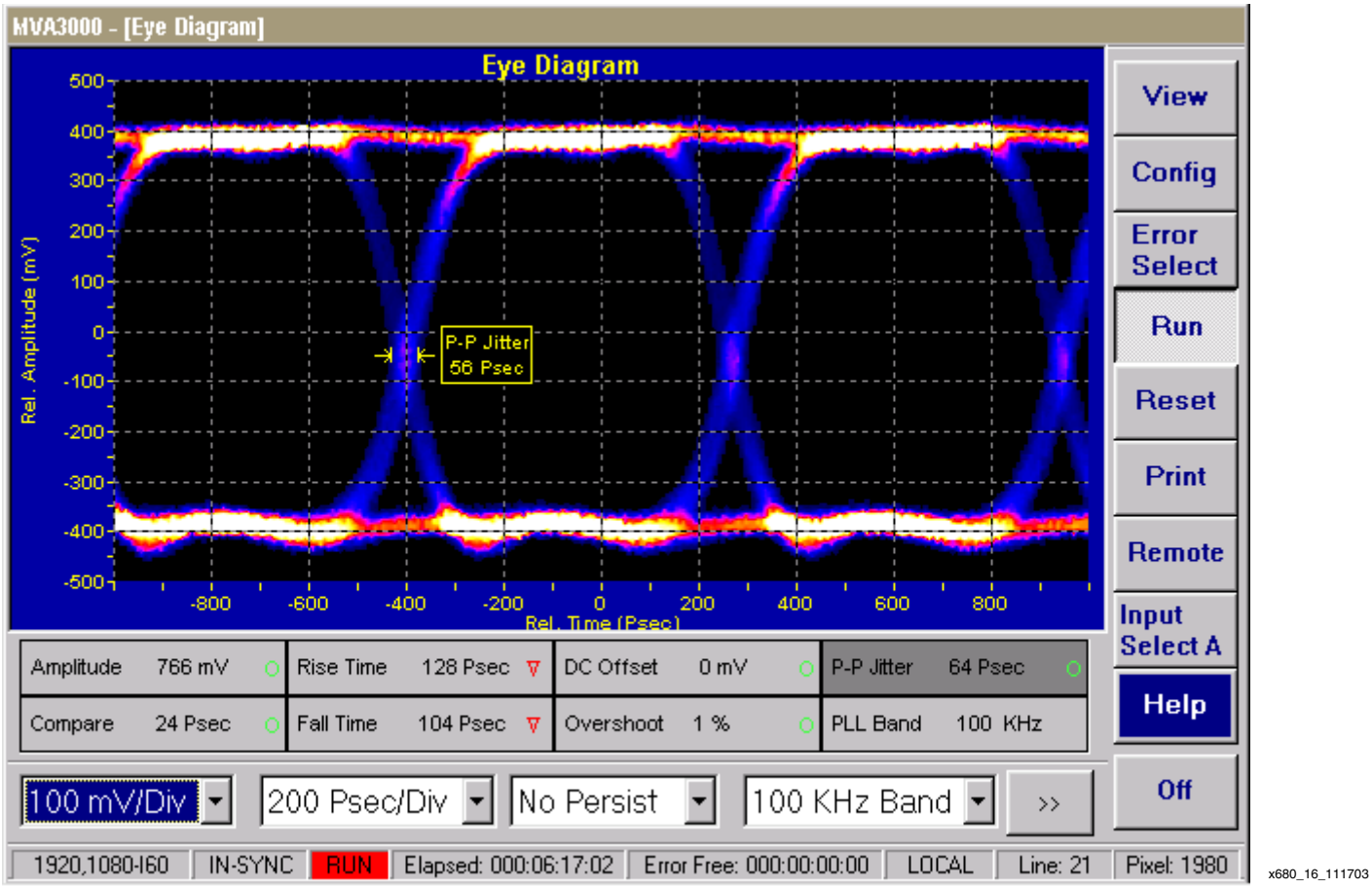

Figure 16: **Xilinx SDV Demo Board Eye Diagram Measurements**

Reference Design Size

Table 4 shows the FPGA resources used by the HD-SDI transmitter reference design. Two implementation sizes are shown, one with line number generation and insertion and one without. As can be seen by the results, the line number generation and insertion accounts for about half of the size of the design when included.

In both cases, the results were obtained using XST running under ISE 6.1. Area optimization was used. Both designs met the necessary timing constraints using a Virtex-II Pro –5 speed grade device.

The multigenHD video pattern generator module, included in the sdv_hdsdi_tx reference design is not included in the reference design results shown in Table 4.

Table 4: **Reference Design Implementation Sizes**

Reference Design	FFs	LUTs	Slices
With LN generation and insertion	197	313	190
Without LN generation and insertion	126	144	85

Conclusion

This application note describes the implementation details of an HD-SDI transmitter using the RocketIO transceivers available in the Virtex-II Pro FPGA family. The RocketIO transceivers combined with an HD-SDI encoder and other support functions built in the fabric of the FPGA can easily implement an HD-SDI transmitter.

The HD-SDI transmitter reference design module requires very few resources in the FPGA, thus making it quite easy to implement multiple HD-SDI interfaces in even the smallest member of the Virtex-II Pro family or to integrate video processing functions and an HD-SDI transmitter all in the same part.

References

The following references provide related information for this application note:

1. All the SMPTE standards referenced in this application note are available from The Society of Motion Picture and Television Engineers. These standards can be purchased at the SMPTE website: http://www.smpte.org.
2. Xilinx application note XAPP681 – HD-SDI Receiver Using Virtex-II Pro RocketIO Transceivers (planned for future release)
3. The Xilinx SD-SDI applications notes are:
   - XAPP247 – Serial Digital Interface (SDI) Physical Layer (planned for future release)
   - XAPP288 – Serial Digital Interface (SDI) Video Decoder
   - XAPP298 – Serial Digital Interface (SDI) Video Encoder
   - XAPP299 – Serial Digital Interface (SDI) Ancillary Data and EDH Processors
   - XAPP625 – SDI: Video Standard Detector and Flywheel Decoder
4. Xilinx application note XAPP683 – Multi-Rate HD/SD-SDI Transmitter using Virtex-II Pro RocketIO Transceivers (planned for future release)
5. Xilinx application note XAPP684 – Multi-Rate HD/SD-SDI Receiver using Virtex-II Pro RocketIO Transceivers (planned for future release)
6. Xilinx application note XAPP682 – HDTV Video Pattern Generator (planned for future release)
7. IEC 60169-8 (1978-01), Radio Frequency Connectors, Part 8: R.F. Coaxial Connectors with Inner Diameter of Outer Conductor 6.5mm (0.256 in) with Bayonet Lock – Characteristic Impedance 50 Ohms (Type BNC)
8. The Gennum GS1528 data sheet is located at http://www.gennum.com/vb/pdffiles/16632DOC.pdf
9. UG024: *RocketIO Transceiver User Guide*
10. The Xilinx SDV Demo board is available from Cook Technologies (part number CTXIL103). Further information is available at http://www.cook-tech.com.

Appendix: Reference Design Details

This appendix contains detailed design information for the hdsdi_autodetect_ln module and the hdsdi_encoder module.

Video Format Detection and Line Number Generation

If the video entering the HD-SDI transmitter does not contain line numbers after each EAV, then it necessary for the HD-SDI transmitter to decode the video stream and generate and insert line numbers into the video prior to transmission.

The module hdsdi_autodetect_ln is an HD video decoder. It can determine the incoming video format and generate line numbers. This module does not insert the line numbers. Line number insertion is done by the hdsdi_insert_ln module inside of hdsdi_tx_path.

The hdsdi_autodetect_ln module contains a finite state machine (FSM) that analyzes the incoming video and determines which video format the input video stream matches. The FSM counts the number of words per line and the number of lines per field (or frame for progressive scan formats). The FSM then compares these word and line counts to the known video formats. If it finds a match, it generates a 4-bit code indicating which video format was detected.

Once the FSM is locked to a video format, it will generate line numbers for that video format. The line number generator looks for the first active line of a field. The first active line of a field is easy to detect because the V bit in the EAV symbol transitions from a 1 in the last line of the video blanking interval to a 0 in the first active line. When the first active line of a field is detected, the line number counter is loaded from a small ROM that contains the first active line number of each field for all of the known video formats. Once loaded from this ROM, this line counter increments every time an EAV is detected unless the counter has reached the last line of the frame, in which case it is reloaded with a value of 1. Another ROM provides the maximum line number for each of the known video formats, so that the line counter can rollover to 1 at the appropriate time.

Figure 17 shows a block diagram of the hdsdi_autodetect_ln module, and Figure 18 is the state diagram for the FSM.

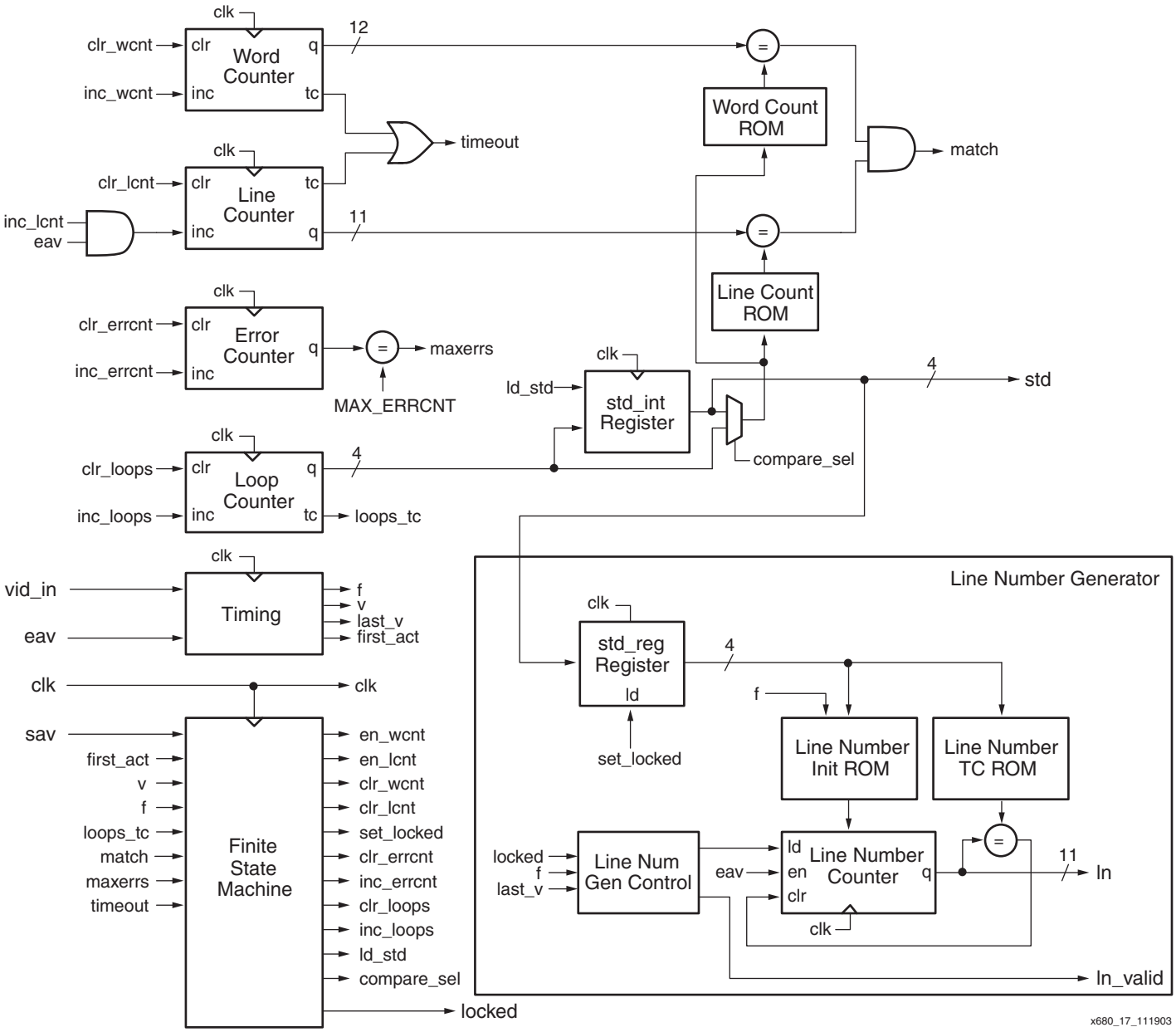

Figure 17: **hdsdi_autodetect_ln Module**

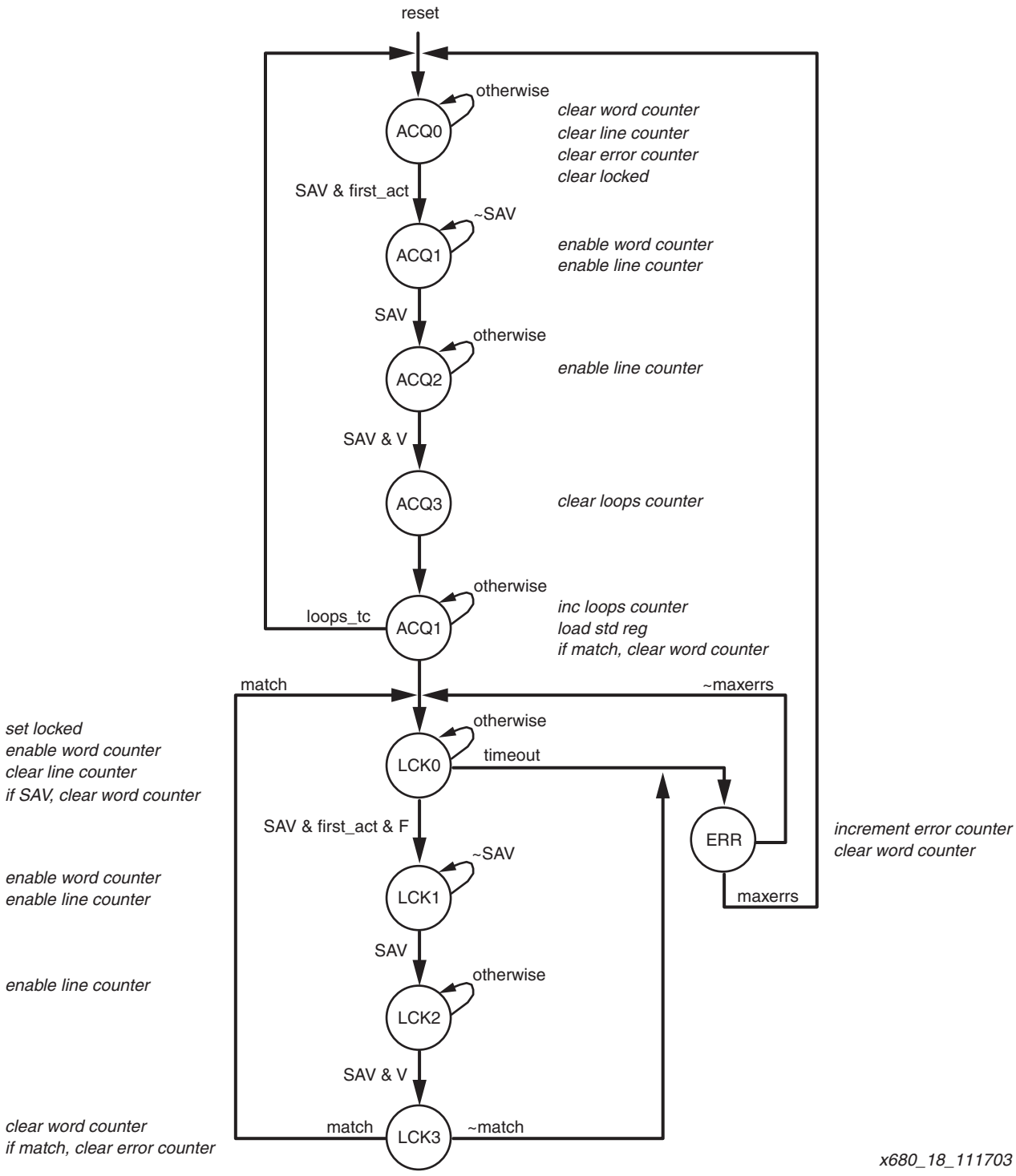

Figure 18: **hdsdi_autodetect_ln State Diagram**

The FSM has two main loops, the acquire loop (ACQ) and the locked loop (LCK). Initially, the FSM starts in the ACQ loop, trying to match the video to a known format. As soon as the state machine sees the XYZ word of an SAV, and the current line is the first active line of a field (as indicated by the V bit transitioning from High to Low), the state machine enables the word and line counters. The word counter increments every clock cycle and the line counter increments once per line. Words are counted until the XYZ word of the next SAV, at which time the word counter will contain the total number of words per video line. Line counting continues until the

next vertical blanking interval begins (V goes High), at which time the line counter contains the active lines per field (or frame if the video is progressive).

After counting the number of words and lines, the FSM moves to state ACQ4 where it sequentially compares the measured words and line counts to the corresponding values for each known video format. The “loops” counter is at zero when the FSM enters ACQ4. This counter provides the address to the ROMs that contain the word and line counts for each known format. The FSM stays in ACQ4, incrementing the loops counter to sequence through all of the known video formats, comparing the acquired words and lines value to the known values of each format, until either a match is found or the loops counter reaches its terminal count. If a match is found, the std register is loaded with the value of the loops counter so that it contains the code indicating the matching video format and the FSM moves to the LCK loop. If no match is found and the loops counter reaches its terminal count, the FSM moves back to state ACQ0 and begins the process again.

The LCK loop is similar to the ACQ loop. In the LCK loop, the FSM continuously counts the number of words per line and active lines per field and compares them to the known values for the current video format. If a mismatch is encountered, the FSM moves to the ERR state and the error counter is incremented. If the error counter reaches the MAX_ERRS value, then the FSM returns to the ACK loop, to acquire the new format.

Note that the error counter is reset in state LCK3 if a successful match is made between the measured words and lines and the expected values. Thus, the FSM must see MAX_ERRS consecutive fields with errors before it return to the ACQ loop. In some cases, other modules within the design may know when the video format changes. In this case, it is possible to immediately move the FSM to the ACQ loop in order to avoid the MAX_ERRS fields latency that the FSM normally waits before moving to the ACQ loop. This is done by asserting the reacquire input to the FSM. This input forces the FSM to the ACQ0 state whenever it is asserted High.

The hdsdi_autodetect_ln module can detect all the video formats shown in Table 1 and Table 2. However, several of the 1080sF formats cannot be distinguished from their “look-alike” 1080i formats by word and line counting. For the purpose of generating line numbers, however, it is not necessary to distinguish between the 1080sF formats and their corresponding 1080i formats.

HD-SDI Encoder

HD-SDI encoding is implemented in the hdsdi_encoder module. This encoder has control inputs that enable and disable both parts of the encoding algorithm: the scrambler and the NRZ-to-NRZI converter. For normal operation, these enable inputs should always be High. The scrambler and the NRZ-to-NRZI converter can be disabled for diagnostic purposes. For instance, if both of them are disabled, the video stream is directly serialized without any encoding, making it easier to determine if other portions of the transmitter are working correctly in simulation or by using a logic analyzer.

As described earlier in this application note, the HD-SDI bitstream is interleaved and alternately contains a 10-bit word from the C channel followed by a 10-bit word from the Y channel. If the HD-SDI encoder ran at 2X the HD-SDI word rate (108.5 MHz or 108.5 / 1.001 MHz) then a single 10-bit encoder module could be used along with a MUX alternately feed C and Y channel words into the encoder. However, the hdsdi_encoder module has 20-bit input and output data paths and runs at the HD-SDI word rate. As shown in Figure 19, this module contains two instances of a 10-bit encoder module called smpte_encoder. One smpte_encoder module encodes the C channel and the other encodes the Y channel. These two modules are interconnected so that encoded bits from the C channel encoder affect the encoding of the Y channel word (remember that each Y word is sent after the corresponding C word and the encoding results from the C word affect the encoding of the Y word). Likewise, the encoded results from the Y word are saved in a register and affect the encoding of the C word during the next clock cycle.

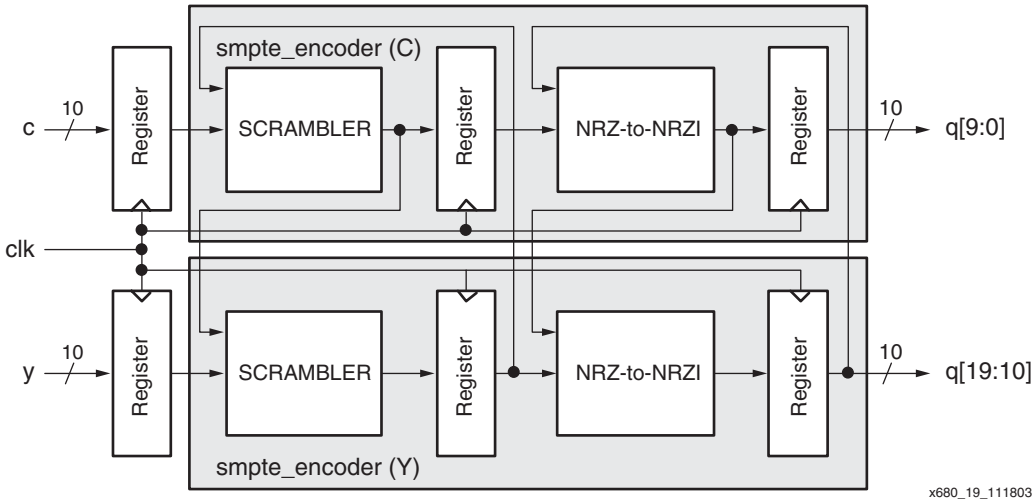

Figure 19: **hdsdi_encoder Module**

The smpte_encoder module implements the HD-SDI scrambling and NRZ-to-NRZI encoding algorithms. It operates at the word rate and encodes one 10-bit word per clock cycle. The smpte_encoder module has a two clock cycle latency and each 10-bit video word is processed in two stages. The first stage of the encoder is the scrambler and the second stage is the NRZ-to-NRZI converter. Figure 20 is a block diagram of the smpte_encoder module.

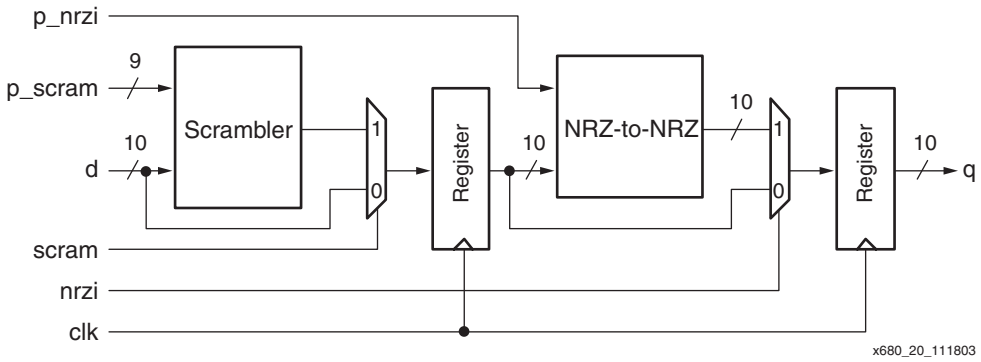

Figure 20: **smpte_encoder Module**

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/25/03	1.0	Initial Xilinx release.