XAPP684 (v2.0) September 22, 2004

Multi-Rate HD/SD-SDI Receiver Using Virtex-II Pro RocketIO Multi-Gigabit Transceivers

Author: John F. Snow

Summary

The SD-SDI standard is widely used in broadcast studios and video production centers to transport standard definition (SD) digital video serially over video coax cable. The HD-SDI standard is similar, but transports high-definition (HD) digital video. The SD-SDI and HD-SDI standards are similar enough that it is possible to implement interfaces for video equipment that support both standards through the same connector.

This application note describes how to use the RocketIO™ multi-gigabit transceivers available in the Virtex-II Pro™ family of FPGA devices to implement a receiver that can support both SD-SDI and HD-SDI. The flexibility of the RocketIO transceivers, combined with the programmable logic of the Virtex-II Pro devices, makes it possible to implement multi-rate SDI interfaces.

Introduction

The SD-SDI standard, traditionally known simply as SDI, is defined by the SMPTE 259M standard [Ref 1] and also by the equivalent ITU-R BT.656 standard [Ref 2]. The SMPTE 292M standard defines HD-SDI. SD-SDI is widely used in broadcast studios and video production centers today. Use of HD-SDI is increasing rapidly as the broadcast industry ramps up support for HDTV broadcasting.

Throughout this document, the term SD-SDI is used to refer to the SD version of the interface standard. SDI is used to refer generically to both SD-SDI and HD-SDI. And, multi-rate SDI is used to refer to support of both standards.

This application note focuses specifically on the implementation of a multi-rate SDI receiver using the RocketIO transceivers available in the Virtex-II Pro FPGA family. A companion application note, XAPP683, describes the implementation of multi-rate SDI transmitters, also using RocketIO transceivers [Ref 3].

This application note refers to the Xilinx SD-SDI application note set [Ref 4] and the Xilinx HD-SDI receiver application note, XAPP681 [Ref 5]. Please refer to these application notes for more information.

The RocketIO transceivers are designed to support serial bit rates from 622 Mbps to 3.125 Gbps. HD-SDI, with bit rates of approximately 1.5 Gbps, falls within this range. However, all of the current SD-SDI bit rates, ranging from 143 Mbps to 540 Mbps, are well below the range supported by the RocketIO transceivers. Therefore, the main focus of this application note is on a technique that allows the RocketIO receiver to support these slower bit rates without violating any of the specifications of the RocketIO transceivers. XAPP681 describes how to use the RocketIO transceiver to build an HD-SDI receiver. The combination of the SD-SDI reception technique described here and the HD-SDI receiver from XAPP681 results in a multi-rate SDI receiver design. SD-SDI only receivers can also be built using the RocketIO transceivers or they can be implemented in the fabric of the FPGA as described in the SD-SDI application note set.

The reference design presented here has been tested and verified on the Xilinx SDV demo board [Ref 6].

SD-SDI and HD-SDI Similarities and Differences

The basic electrical specifications of HD-SDI and SD-SDI are virtually identical, making it possible to design multi-rate SDI interfaces that support both standards. Both standards have the same basic electrical interface: a singled-ended signal with an 800 mV peak-to-peak swing centered around 0.0V. Both use 75Ω coaxial cable and BNC connectors. And, both use the same encoding algorithm.

The most obvious difference between HD-SDI and SD-SDI is the much higher bit rate required to support HD video. HD-SDI has two bit rates: 1.485 Gbps and 1.485/1.001 Gbps. SD-SDI bit rates range from 143 Mbps to 540 Mbps.

Another difference is that HD video is 20 bits wide with two 10-bit channels, one for chroma and one for luma. An HD-SDI transmitter encodes and transmits 20 bits for every video clock cycle, whereas SD-SDI only encodes and transmits 10 bits per video clock cycle.

Multi-Rate SDI Receiver Functions

This section describes the basic functions implemented by a multi-rate SDI receiver. Figure 1 is a block diagram of a typical multi-rate SDI receiver. The following sections describe the basic functions of the receiver.

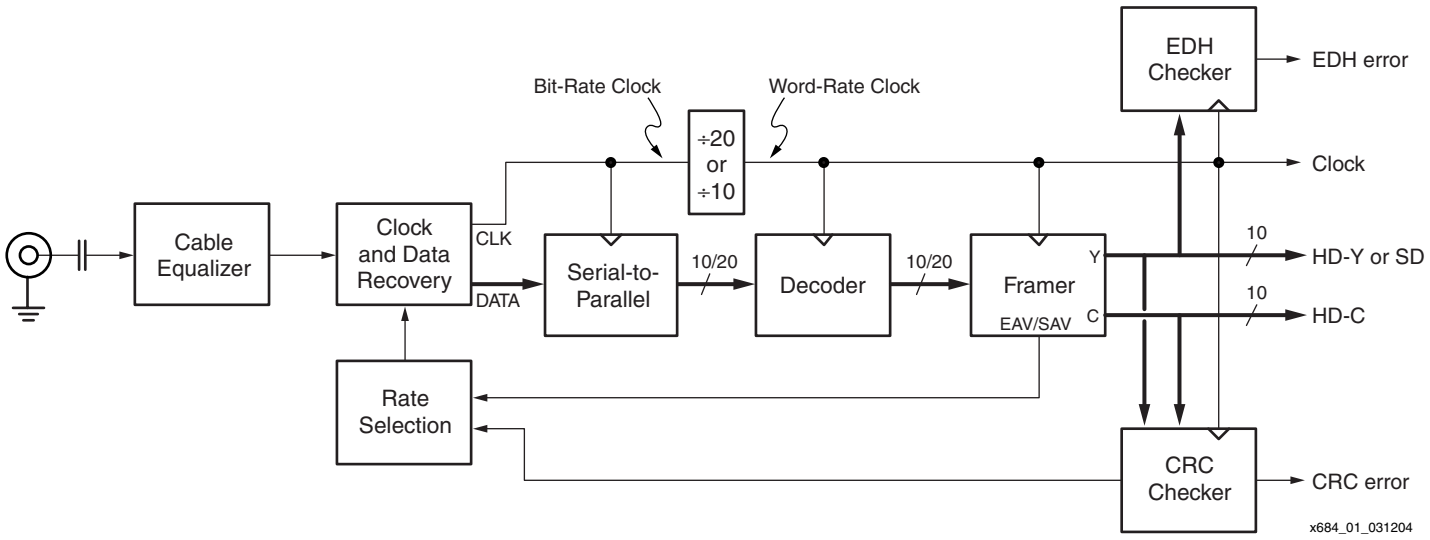

Figure 1: **Multi-rate SDI Receiver Block Diagram**

Cable Equalization

The HD-SDI standard allows HD digital video to be sent over video coax cables up to 100 meters in length. SD-SDI, with its slower bit rates, allows maximum cable lengths of up to 300 meters.

The coax cable causes frequency-dependent attenuation of the signal, where the higher frequency components of the signal are attenuated more than the lower frequency components. The coax cable also causes frequency-dependent phase distortion, where the higher frequency components are phase shifted more than the lower frequency components. After passing through long coax cables, the SDI signal is severely distorted and attenuated. The receiver must compensate for this attenuation and distortion before attempting to recover the signal.

Cable length equalization is used to compensate for the attenuation and distortion introduced by the coax cable. Typically, an adaptive cable length equalizer is used in SDI receivers. Such an equalizer actively monitors the amount of attenuation and distortion present on the incoming signal and applies the correct amount of equalization to the signal. The cable length can be changed without requiring a change to the equalizer, as would be the case if fixed length equalization were used.

Clock and Data Recovery

After cable equalization, the SDI receiver recovers the clock and data from the SDI bitstream. This is typically done with a PLL-based clock and data recovery (CDR) unit. A recovered clock is often required for the receiver because SDI does not support clock correction to allow the incoming bitstream to be easily resynchronized to a local reference clock. Instead, the recovered clock from the CDR unit is typically used to clock all the SDI receiver logic downstream from the CDR unit.

Deserialization

After the CDR unit, the serial bitstream is typically deserialized to produce a parallel data stream. While it is possible to implement SDI decoding and framing in a serial fashion, doing so at the HD-SDI clock rates of 1.485 Gbps is not possible in today's FPGA technology. Instead, the HD-SDI bitstream is deserialized into 20-bit words for processing by the decoder and framer. The serial clock is divided by 20 to derive a word-rate clock for the downstream receiver functions. For SD-SDI, it is usually more convenient to deserialize the bitstream into 10-bit words and divide the clock by 10.

Decoding

Both HD-SDI and SD-SDI use the same two-stage encoding scheme. The first stage performs pseudorandom scrambling and the second stage performs non-return-to-zero (NRZ) to NRZ-inverted (NRZI) conversion. After recovering the data, the SDI receiver must decode the data by reversing the two encoding steps—first converting the NRZI data to NRZ and then descrambling the data. Figure 2 shows conceptually how the SDI bitstream is decoded in a serial manner.

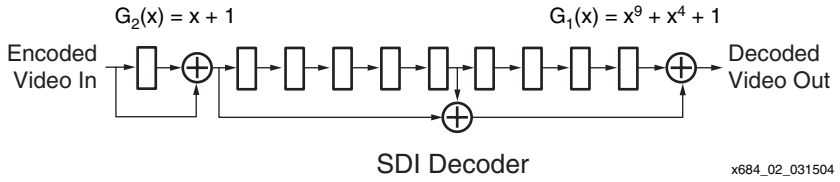

Figure 2: **SDI Decoding Algorithm**

The RocketIO transceivers have built-in 8B/10B decoders. However, they do not have SDI decoders. So, the recovered data from the RocketIO transceiver must bypass the decoding logic built into the RocketIO transceiver and be sent directly to the transceiver's RXDATA port still encoded. An SDI decoder built in the fabric of the FPGA implements the SDI decoding algorithm on the recovered data. The data is decoded in a parallel manner—20 bits are decoded per clock cycle for HD-SDI and 10 bits per clock cycle for SD-SDI.

Framing

The data words from the SDI decoder are not word aligned. The CDR unit has no concept of where the video sample boundaries lie in the continuous stream of incoming bits. The decoder does not care where the video sample boundaries lie since it can decode the data without this information. However, after decoding, it is necessary to identify the sample boundaries and realign the data. This process of realigning the data is called framing.

The framer in the SDI receiver monitors the decoded data and looks for the bit sequences that mark the beginning of the timing references. There are two timing references per video line: the end-of-active video (EAV) and the start-of-active video (SAV). Both the EAV and SAV have the same format and are four 10-bit words long. The first three words are always fixed values. The first word of the timing reference is a word of all ones and has a hex value of 3FFh. The second and third words of the timing reference are made up of all zeros (000h). The fourth word of the

timing reference is called the XYZ word. Figure 3 shows the format of the XYZ word of the timing reference.

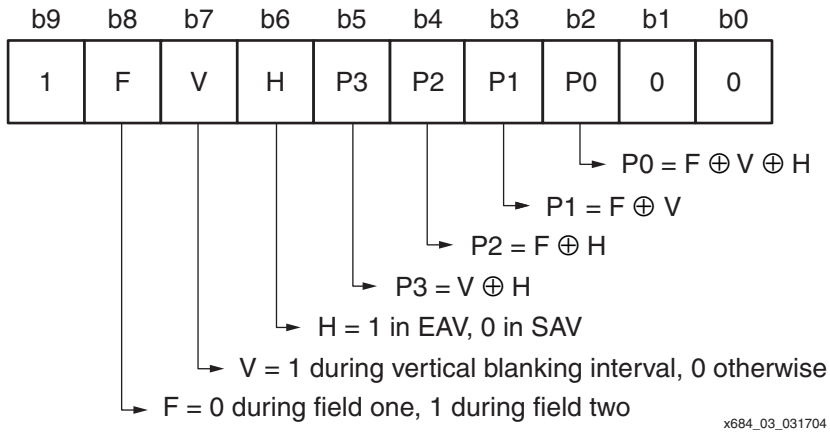

Figure 3: **XYZ Word Format**

In HD-SDI video streams, there are two separate 10-bit channels called the luma (Y) channel and the chroma (C) channel. Each channel has its own set of timing references. The channels are considered to be synchronous so that the first word of the EAV, for example, would appear on both the Y channel and C channel simultaneously.

Before transmission by the HD-SDI transmitter, the Y and C channels are interleaved so that a C word is transmitted first followed immediately by the corresponding Y word. Figure 4 shows the details of this interleaving.

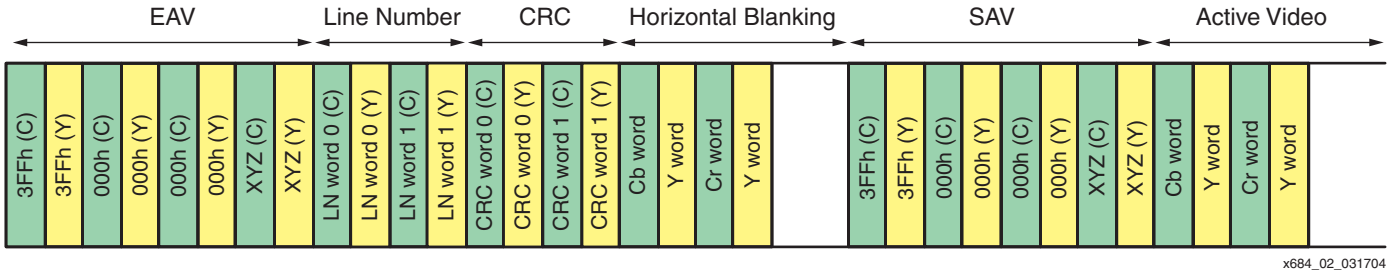

Figure 4: **HD-SDI Interleaved Data Stream**

In SD-SDI video, there are also Y and C words for each sample, but the components are always interleaved and there is a single timing reference for the entire SD-SDI video stream, as shown in Figure 5.

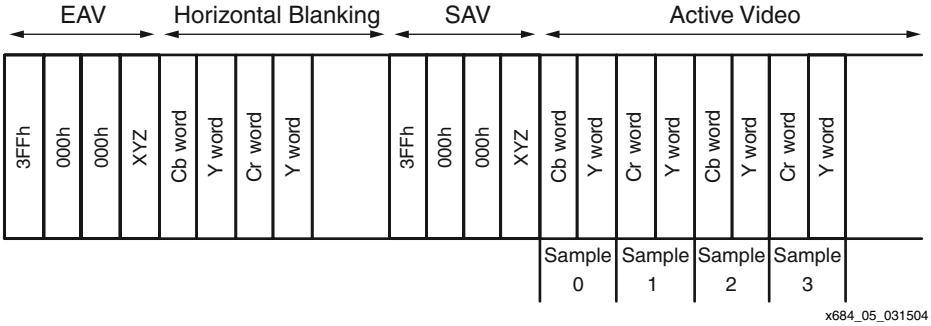

Figure 5: **SD-SDI Data Stream**

The framer looks for the unique 3FFh, 000h, 000h sequence that marks the beginning of the EAV and SAV sequences. Only this unique pattern can be used as a reference point for realigning the data. For SD-SDI, the framer can simply look for this sequence, but for HD-SDI, due to the interleaving of the Y and C channels, the timing reference actually looks like this: 3FFh, 3FFh, 000h, 000h, 000h, 000h. A multi-rate SDI framer must be capable of detecting the timing reference sequence for both SD-SDI and HD-SDI.

The framer must look for the timing reference sequence pattern beginning at any possible bit position in the data words from the decoder. Once the pattern is identified, the framer knows the bit offset of the first bit of each data word. A barrel shifter is used to realign the data to proper word boundaries.

For more details about the framing function, refer to XAPP681.

Error Checking

Error checking is done much differently in HD-SDI than it is in SD-SDI. Error checking in SD-SDI was added after the original SD-SDI specification was released. As a result, error checking in SD-SDI is quite cumbersome. In contrast, error checking was implementing from the beginning in HD-SDI in a much more elegant and robust manner.

In HD-SDI, the transmitter calculates two 18-bit CRC values for each video line, one for the Y channel and one for the C channel. Each CRC value is separately formatted into two 10-bit words and placed into the video stream two words after the end of the EAV. The two words between the EAV and the CRC contain a value indicating the line number. XAPP680 has details of how the CRC values are computed for each line. The starting and ending positions are all related to the positions of the EAV and SAV on each line, making it easy to compute the CRC values and to locate the position where the CRC values are located in the video stream. The transmitter and receiver don't need to have the details of the video format, such as number of words per line and lines per frame, in order to generate or check the CRC values on each line.

In SD-SDI, the transmitter calculates two 16-bit CRC values for each frame of video. One CRC value is called the full-field (FF) CRC and is calculated for all words in most (but not all) of the video lines of a field. The other CRC value is called the active-picture (AP) CRC and is calculated on just the active portion of the field. These two CRC values, plus some additional error flags, are formatted into a 23-word long packet called the error detection and handling (EDH) packet. The EDH packet must be located immediately prior to the SAV on a specific video line in each field. The line where the EDH packet is located depends on the video format. Details of how EDH packets are calculated can be found in XAPP299.

In order to generate or check the EDH packet in an SD-SDI video stream, the transmitter or receiver must first determine the format of the video since the details of how the CRC values are calculated and the position of the EDH packet are different for each video format. This greatly complicates the process of error checking in SD-SDI. In fact, the EDH processor function accounts for almost half of the logic in the multi-rate SDI receiver reference design.

Because there is no commonality between HD-SDI and SD-SDI error checking, a multi-rate receiver generally needs two separate error checking modules, one for HD and one for SD.

Rate Selection

When implementing a multi-rate SDI receiver, the RocketIO transceiver requires a separate reference clock frequency for each bit rate that is to be received. For a typical SDI receiver capable of receiving both HD-SDI bit rates plus the 270 Mbps SD-SDI bit rate, three reference clock frequencies are needed for the RocketIO transceiver.

In some cases, the SDI receiver can simply be told which bit rate to expect, making selection of the proper reference clock relatively straightforward. However, in most cases, SDI receivers are expected to automatically detect the frequency of the bitstream. This is the function of the

rate selection block in Figure 1. It forms a feedback loop, monitoring the data being received for errors and cycling through the reference clocks until the receiver locks to the bitstream.

Implementing the Multi-Rate SDI Receiver

This section details how to implement a multi-rate SDI receiver using the RocketIO transceivers in the Virtex-II Pro FPGAs.

Cable Equalization

As previously described, an SDI receiver usually has an adaptive cable length equalizer to compensate for attenuation and distortion of the signal caused by long runs of coax cable. The RocketIO transceivers in the Virtex-II Pro FPGA do not include adaptive cable length equalizers. So, an external cable equalizer must be used to interface the SDI cable to the RocketIO transceiver. As a side benefit, the cable equalizer also converts the single-ended SDI signal into a differential signal. The CML inputs of the RocketIO receiver require a differential input signal. Most multi-rate SDI cable equalizers currently available have 3.3V LVPECL outputs that are not directly compatible with the 2.5V CML inputs of the RocketIO transceiver. AC coupling can be used to interface the LVPECL outputs of the cable equalizer to the CML inputs of the RocketIO transceiver. Figure 6 shows a typical AC coupled interface between a Gennum GS1524 cable equalizer and a RocketIO transceiver.

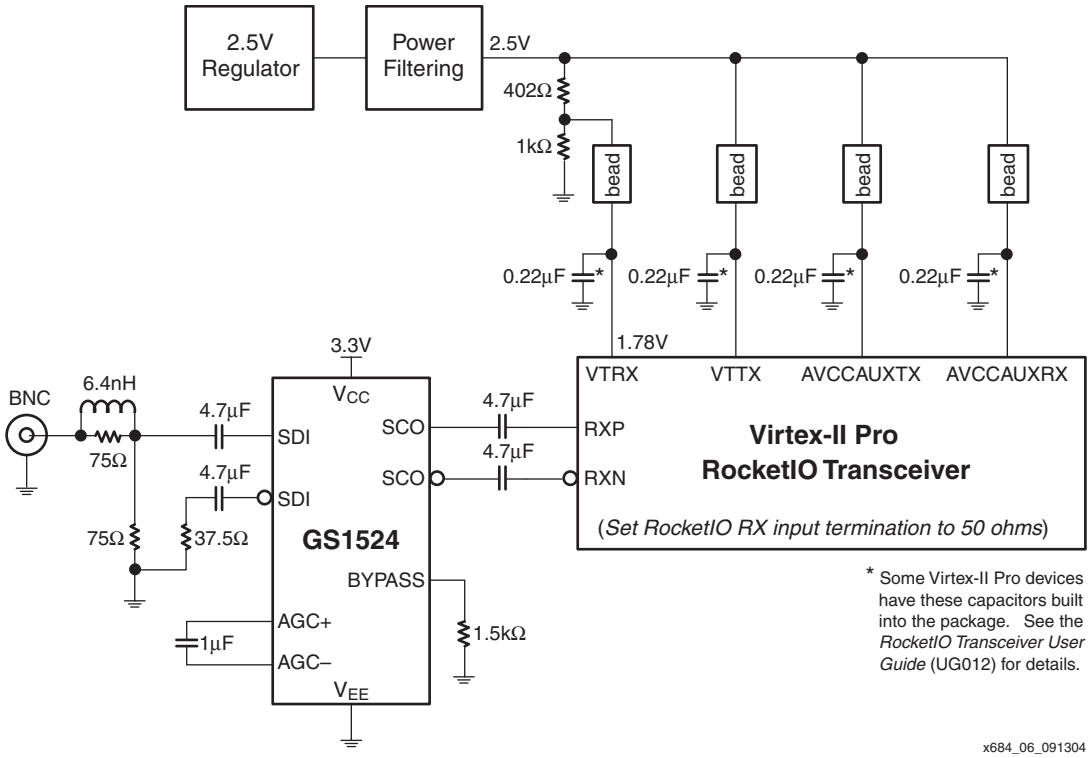

Figure 6: **Interfacing a Cable Equalizer to the RocketIO Transceiver**

There are several important details in Figure 6:

- The recommendations given in the GS1524 data sheet [Ref 7] must be followed for the interface network between the BNC cable connector and the GS1524's input.
- The AC coupling capacitors between the GS1524 and the RocketIO receiver must be in the 1 µF to 10 µF range to pass the SDI pathological waveforms without too much voltage droop. Typically, 4.7 µF capacitors are used.
- The input impedance of the RocketIO transceiver should be set equal to impedance of the circuit board traces between the cable equalizer and the transceiver's inputs. Normally this is 50Ω.

- As described in the RocketIO Transceiver User Guide [Ref 8], when using AC coupling, the RocketIO receiver termination voltage (VTRX) must be between 1.6V and 1.8V. As shown in Figure 6, the required termination voltage can be generated from 2.5V by using a voltage divider network. The resistor values shown are sized to supply the termination voltage to a single RocketIO transceiver, so this resistor network must be duplicated for each RocketIO transceiver used as an SDI receiver.
- Some Virtex-II Pro devices have internal power filter capacitors for the VTRX, VTTX, AVCCAUXRX, and AVCCAUXTX signals of each RocketIO transceiver. Consult the *RocketIO Transceiver User Guide* for more information.
- It is absolutely essential that all guidelines given in the *RocketIO Transceiver User Guide* for layout, bypass capacitors, and power regulation and filtering be observed. Do not attempt to provide power to the RocketIO transceiver from a switching regulator.

Receiving SD-SDI Bitstreams with the RocketIO Transceiver

The original SD-SDI standard supports four bit rates:

- 143 Mbps for NTSC digital *composite* video
- 177.3 Mbps for PAL digital *composite* video
- 270 Mbps for NTSC and PAL digital *component* video
- 360 Mbps for NTSC and PAL 16:9 aspect ratio digital *component* video

In addition, a more recent document (SMPTE 344M) adds a 540 Mbps bit rate compatible with SD-SDI.

The digital composite video rates of 143 Mbps and 177.3 Mbps are rarely used. The most commonly used bit rate, by far, is 270 Mbps. It is quite common for a piece of video equipment to support only the 270 Mbps bit rate through its SDI interface. It is also common for equipment to support both 270 Mbps and 360 Mbps. The 540 Mbps bit rate is relatively new and not widely supported, yet.

All of these bit rates are well below the 622 Mbps minimum bit rate supported by RocketIO transceivers in Virtex-II Pro devices. Therefore, it is necessary to work around this minimum bit rate limitation in order to support SD-SDI with the RocketIO transceivers.

As described in XAPP683, SD-SDI bitstreams can be transmitted with the RocketIO transceiver by simply over-clocking the transceiver at some multiple of the SD-SDI bit rate and then sending each encoded bit multiple times.

Trying to receive SD-SDI bitstreams by simply over-clocking the RocketIO transceiver and relying on the CDR unit of the transceiver to lock to a harmonic of the bitstream frequency does not work. The primary issue is run length limitations. SD-SDI has a maximum run length of 39 consecutive bits without a transition.(1) This is well within the RocketIO transceiver's maximum run length limit of 75 bits. However, when over-sampling the bitstream by 3X, the minimum over-sampling rate sufficient to meet the minimum frequency requirements of the RocketIO transceiver with a 270 Mbps input bitstream, the 39-bit long maximum run length of the SD-SDI bitstream appears to the RocketIO transceiver as being three times as long (117 bits long). This clearly exceeds the maximum run length limit of the RocketIO transceiver by more than 50%. Thus, the CDR unit in the RocketIO transceiver does not maintain lock with a 270 Mbps SD-SDI bitstream when simple over-sampling is attempted.

The CDR unit in the RocketIO transceiver was not designed to work with bitstreams with the frequency and characteristics of SD-SDI. To overcome this, Xilinx has developed an over-sampling technique that can be used with the RocketIO transceivers. This technique does not rely on the CDR section of the RocketIO transceiver for either clock or data recovery. The

1. SD-SDI bitstreams carrying PAL digital *composite* ($4f_{SC}$) video can have a maximum run length of 40 bits. However, this application note is mainly targeted at digital *component* video, which has a worst case SD-SDI run length of 39 bits.

receiver section of the RocketIO transceiver is used as an asynchronous sampler. Data recovery is done in a data recovery unit (DRU) implemented in the fabric of the FPGA. The PLL in the CDR section of the RocketIO transceiver does not have to lock to the bitstream frequency for this technique to work. Therefore, the SD-SDI run lengths are not an issue for the RocketIO transceiver when using this technique.

The DRU takes the raw, over-sampled bitstream data from the output of the RocketIO transceiver. It scans the over-sampled data, looking for bit transitions. From these transitions, it determines the optimum place to sample each bit, producing a recovered bitstream on its output.

The over-sampling technique allows the RocketIO transceivers in the Virtex-II Pro devices to receive SD-SDI bitstreams of 360 Mbps and slower. It does not, however, work for 540 Mbps bitstreams. The minimum supported 8X over-sampling rate at 540 Mbps exceeds the maximum bit rate supported by the RocketIO transceivers in the Virtex-II Pro.

There are two versions of the DRU. Version 1 synthesizes a recovered 27 MHz clock by dividing a global 270 MHz clock by 10. Version 2 generates a clock enable output. This clock enable is used with a global clock to allow all downstream logic in the SDI receiver to run at the 27 MHz SD rate.

DRU Version 1

Figure 7 shows an example of how the RocketIO transceiver and the DRU (version 1) work together to receive a 270 Mbps SD-SDI bitstream. In this example, the RocketIO transceiver is over-sampling the bitstream by 8X (2.16 GHz sample rate).

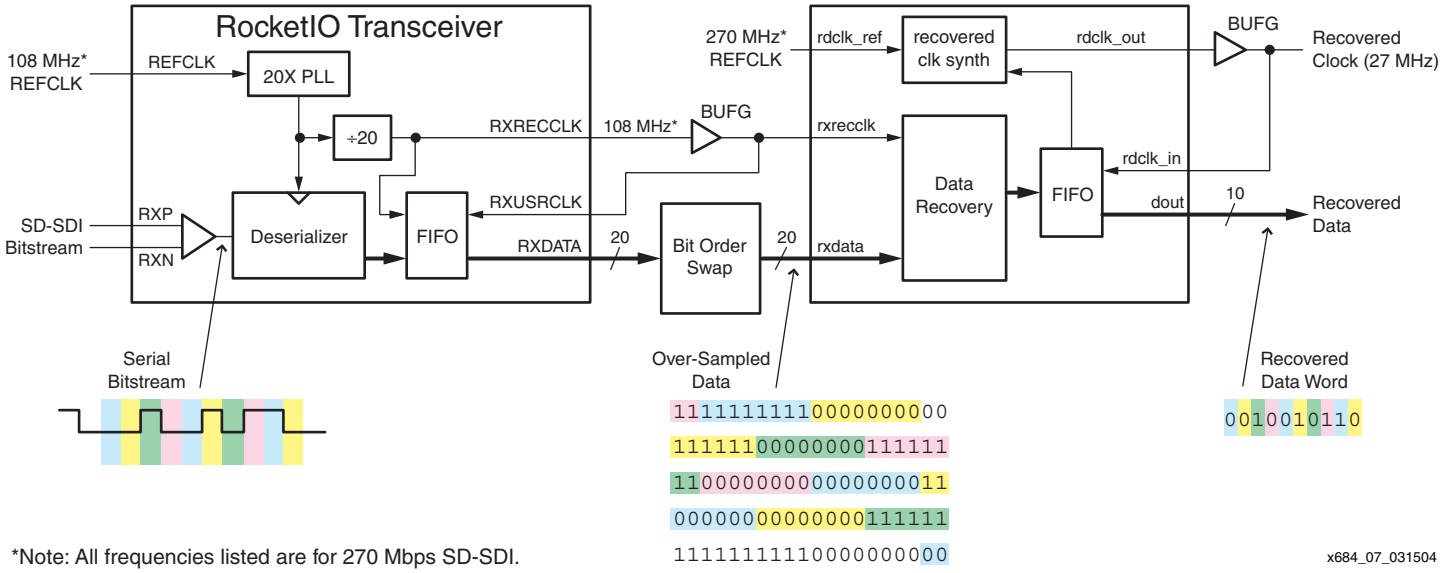

Figure 7: **RocketIO Transceiver and Data Recovery Unit**

The Xilinx DRU used in this application can work at sampling rates between 8 and 11 times the bitstream frequency. The RocketIO transceiver multiplies the selected reference clock by 20, so the correct RocketIO transceiver reference clock frequency is calculated as:

$$\text{reference clock freq} = \text{bitstream freq} \times \text{over-sample rate} / 20 \qquad \text{Eq. 1}$$

For example, if 8X over-sampling is used on a 270 Mbps bitstream, the reference clock must be 108 MHz. With a 108 MHz reference clock, the RocketIO transceiver is actually running at 2.16 Gbps—that is, it is sampling the bitstream at a 2.16 GHz rate. At the time of this writing, -5 speed grade Virtex-II Pro parts have a maximum bit rate speed for their RocketIO transceivers of 2.0 Gbps. However, since January 2004, selected -5 speed grade Virtex-II Pro devices can

be obtained from Xilinx that have RocketIO transceivers tested to 2.5 Gbps, allowing the -5 speed grade devices to receive 270 Mbps SD-SDI bitstreams.

Version 1 of the DRU synthesizes a word-rate recovered clock called rdclk. To synthesize this clock, the DRU requires a reference clock (rdclk_ref) running at 10X the word-rate. This clock can come from a local oscillator and does not have to be frequency locked to the bitstream. The DRU normally divides rdclk_ref by 10 to produce the rdclk word-rate clock. However, since rdclk_ref is usually not running at exactly the same frequency as the bitstream, the DRU must occasionally make up for the slight differences between the bitstream frequency and the rdclk_ref frequency. So, occasionally, the recovered clock is 9 or 11 rdclk_ref cycles long, instead of the normal 10 cycles. This keeps the recovered clock in step with the rate that the data is being recovered by the DRU. The recovered data from the DRU always changes synchronously with the rising edge of rdclk, even when the period of rdclk is adjusted by the DRU as shown in Figure 8.

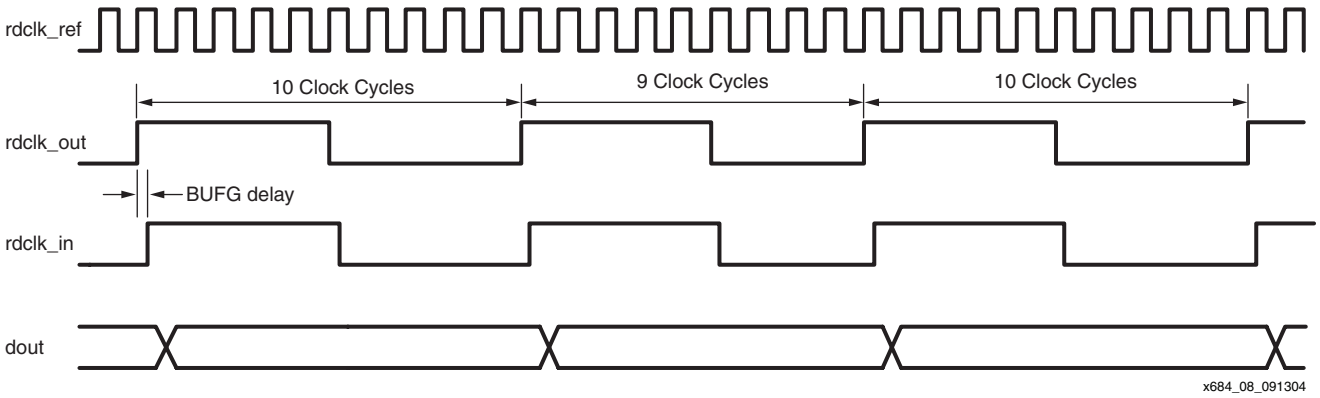

Figure 8: **Synthesis of SD-SDI Recovered Clock (DRU Version 1)**

The rdclk_ref signal doesn't have to come from a separate oscillator. In fact, since the frequency of this clock and the frequency of the reference clock required by the RocketIO transceiver are related, rdclk_ref can be generated in a DCM by multiplying the RocketIO transceiver reference clock by some factor. For example, in Figure 11, a single DCM generates both the 108 MHz RocketIO transceiver reference clock and the 270 MHz clock for the DRU from a 54 MHz source.

DRU Version 2

Version 2 of the DRU does not synthesize a recovered clock. Instead, it requires a reference clock called rdclk, which it divides by a set factor to produce a clock enable output called rdclkDvEn. If, for example, rdclk is running at 108 MHz, the DRU would divide it by 4 and assert the rdclkDvEn for one cycle of every four cycles of rdclk. The rdclkDvEn clock enable and rdclk can then be used to clock logic downstream from the DRU at 27 MHz. rdclk can be any integer multiple of the 27 MHz SD word rate from 4 to 10. Half-integer multiples (4.5X, 5.5X, etc) are also supported. This version of the DRU corrects for differences between the local rdclk frequency and the recovered data rate by sometimes inserting or removing one clock cycle between rdclkDvEn assertions. For example, if a clock divider of 4 is being used, occasionally, there are 2 or 4 rdclk cycles between assertions of rdclkDvEn, rather than the normal 3. The data out of the DRU changes synchronously with the rising edge of rdclk when rdclkDvEn is asserted. Figure 9 shows the timing of the version 2 DRU.

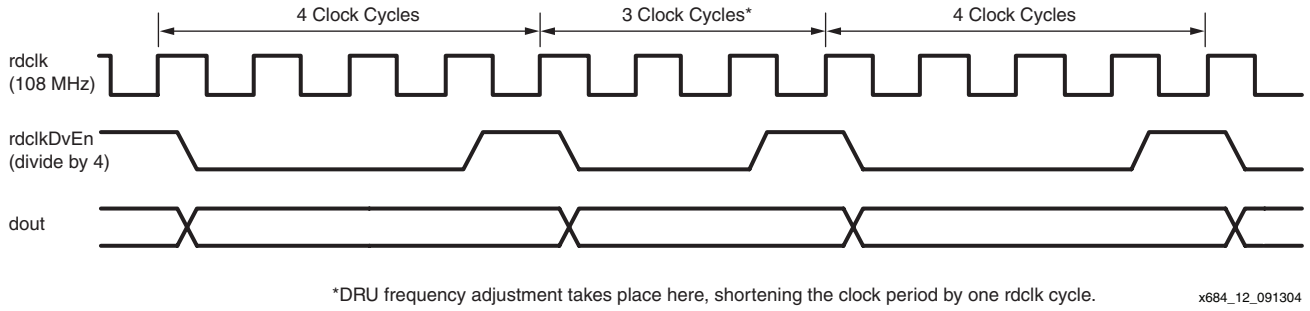

Figure 9: **Clock Enable Timing (DRU Version 2)**

The rdclk reference clock can come from a local oscillator that is totally independent of the recovered clock from the RocketIO transceiver. However, one convenient way to use the version 2 DRU is to use the recovered clock (RXRECCLK) from the RocketIO transceiver as the rdclk reference clock. See the reference design description for more details.

RocketIO Transceiver Clocks

The RocketIO transceiver requires two types of clocks: reference clocks and user clocks. The reference clocks are used by the RocketIO transceiver as a reference for the CDR PLL. The user clocks are used to clock data out of the RocketIO transceiver and into the fabric of the FPGA. In addition, the RocketIO receiver also produces a recovered clock, called RXRECCLK.

The following sections describe the clocking requirements of the RocketIO transceivers oriented towards implementing multi-rate SDI interfaces. More details about the clocking requirements of the RocketIO transceivers can be found in the RocketIO Transceiver User Guide [Ref 8].

Reference Clocks

The RocketIO transceiver uses reference clocks for two different purposes:

1. In the transmitter, the reference clock provides a low-jitter frequency reference that the transmitter multiplies by 20 to obtain a bit-rate clock for the transmitter's serializer.
2. In the receiver, the reference clock is used to spin up the CDR unit so that it quickly locks to the incoming bitstream. After the CDR unit is locked to the bitstream, the frequency of the PLL is constantly compared to the frequency of the reference clock to determine if the PLL is maintaining lock to a valid bitstream frequency.

The reference clocks are required to be 1/20th the frequency of the bitstream ±100 ppm.

For HD-SDI, there must be two reference clock frequencies, one for each of the two standard HD-SDI bitstream frequencies. The RocketIO transceiver needs reference clocks of 74.25 MHz and 74.25/1.001 MHz for HD-SDI. For SD-SDI, the reference clock determines the rate at which the bitstream is sampled by the RocketIO transceiver. The reference clock frequency calculation is shown in Equation 1.

Each RocketIO transceiver has four reference clock inputs from which a single reference clock is selected. There are two pairs of reference clocks called REFCLK and BREFCLK. The difference between the pairs of reference clocks is that the BREFCLK inputs are designed for lowest possible jitter. The BREFCLKs can only come from certain special IOBs. By contrast, the REFCLKs can come from any IOB or from anywhere in the FPGA.

As shown in Figure 10, a set of MUXes selects one active reference clock from the four reference clock inputs. An input signal called REFCLKSEL chooses one reference clock from each pair. A final MUX, controlled by a RocketIO attribute called REF_CLK_V_SEL, chooses between the REFCLK or BREFCLK input.

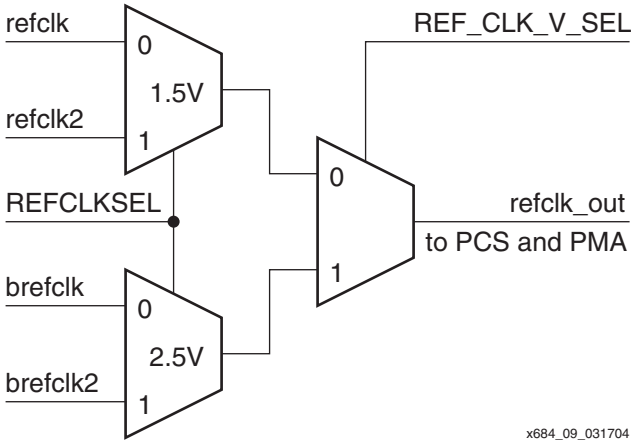

Figure 10: **Reference Clock Selection**

The REF_CLK_V_SEL attribute is initialized when the FPGA is configured and normally cannot be changed until the FPGA is configured again. So, in normal operation, the RocketIO transceiver is really limited to just two reference clock inputs that can be selected dynamically by the REFCLKSEL input. Since supporting two HD-SDI rates plus one SD-SDI rate requires the use of three reference clocks, a work-around to this limitation must be employed.

There are several possible solutions. The simplest solution is to multiplex two of the reference clocks together externally to the RocketIO transceiver or externally to the FPGA. Another solution is to reconfigure the RocketIO transceiver dynamically to change the REF_CLK_V_SEL attribute.

XAPP660 describes a method of using the embedded PowerPC processor available in most Virtex-II Pro devices to reconfigure one or more RocketIO transceivers dynamically. Since a PowerPC is not available in the smallest Virtex-II Pro device, a PicoBlaze soft processor can be used to reconfigure the RocketIO transceivers in this smallest device. There are some disadvantages to reconfiguring the RocketIO transceiver as opposed to MUXing the reference clocks externally. First, it takes several hundred microseconds for the PowerPC to reconfigure the RocketIO transceiver. Thus, the switching time of the external MUX solution is much faster. Second, if the PowerPC is being used for other purposes, then it usually cannot also implement the RocketIO reconfiguration.

The external MUX solution also has a disadvantage when multiple SDI interfaces are implemented in the same FPGA. Unless all the SDI interfaces can always run at the same HD-SDI bit rate, then a separate MUX is required for each RocketIO transceiver used as an SDI interface. Also, care must be taken that the two closely related HD-SDI reference clock frequencies do not mix (heterodyne) in the MUX, causing excessive jitter on the reference clock.

In the reference design section, an external frequency synthesizer generates the two HD-SDI reference frequencies based on a control signal from the RocketIO transceiver. This solution is has the same advantages and disadvantages as the external MUX solution.

Since the CDR unit in the RocketIO transceiver is not really used for receiving SD-SDI, the normal reference clock requirements are not directly applicable. We recommend that the SD-SDI reference clock have no more than 200 ps peak-to-peak of jitter and that it be within ±1000 ppm of the calculated reference clock frequency. If the reference clock jitter is excessive, the PLL in the transceiver might not lock to the reference clock, causing the transceiver to sample the bitstream improperly.

User Clocks

The user clocks clock data out of the RocketIO transceiver and into the fabric of the FPGA. Each transceiver requires two user clocks on the receiver side called RXUSRCLK and

RXUSRCLK2. Each transceiver also has two user clocks for the transmitter side called TXUSRCLK and TXUSRCLK2. If the transmitter portion of the transceiver is not used, the TXUSRCLK and TXUSRCLK2 inputs must still be driven with valid clock signals. In this case, simply connect TXUSRCLK to RXUSRCLK and TXUSRCLK2 to RXUSRCLK2.

RXUSRCLK is the clock signal that clocks data out of the RocketIO transceiver. The receiver's output ports, such as RXDATA, change synchronously with the rising edge of RXUSRCLK. For HD-SDI, the frequency of RXUSRCLK is equal to the word-rate of the HD-SDI interface, either 74.25 MHz or 74.25/1.001 MHz. For SD-SDI, the frequency of RXUSRCLK is equal to reference clock frequency.

The frequency and phase relationships between RXUSRCLK and RXUSRCLK2 depend on the width of the RXDATA port of the RocketIO transceiver. For HD-SDI, a 20-bit RXDATA port is convenient to use because it matches the data word width of HD-SDI (10 bits of Y and 10 bits of C). The DRU used for SD-SDI expects the RXDATA port of the RocketIO transceiver to be 20 bits wide. When using a 20-bit wide output data path from the RocketIO transceiver, RXUSRCLK2 must have the same frequency and phase as RXUSRCLK (simply connect RXUSRCLK and RXUSRCLK2 to the same clock signal).

In serial protocols that have clock correction capability, the RXUSRCLK and RXUSRCLK2 signals usually are derived from the same source as the reference clock. The RocketIO transceiver's clock correction capability is used to occasionally insert or remove idle characters to compensate for the minor differences between the actual clock frequency of the incoming bitstream and the frequency of the local reference clock.

SDI does not support clock correction. Therefore, deriving the transceiver's user clocks from the reference clock would quickly result in an overflow or underflow condition on the output data port of the RocketIO receiver because RXUSRCLK and RXUSRCLK2 would have a slightly different frequency than the bitstream.

When implementing the HD-SDI receiver, the recovered clock (RXRECCLK) from the RocketIO receiver is used as the source of RXUSRCLK and RXUSRCLK2. When connected in this manner, RXUSRCLK and RXUSRCLK2 always run at the same frequency as the CDR PLL in the RocketIO transceiver. Thus, underflow and overflow conditions are prevented.

For SD-SDI, RXUSRCLK and RXUSRCLK2 should also be driven by RXRECCLK. For SD-SDI over-sampling, the PLL in the RocketIO transceiver is either locked to a harmonic of the SD-SDI bitstream or it is locked to the reference clock input. In either case, the RXRECCLK, which is derived from the PLL, indicates the rate at which the over-sampled data is being captured by the transceiver. RXRECCLK is used to clock data out of the transceiver, by connecting it to RXUSRCLK and RXUSRCLK2, and to clock data into the SD-SDI DRU.

Reference Design

The reference design for this application note can be downloaded from the Xilinx website at http://www.xilinx.com/bvdocs/appnotes/xapp684.zip. The multi-rate SDI reference design supports the two HD-SDI bit rates and 270 Mbps SD-SDI.

A high-level description of the reference design is given in the following section. Detailed information about the reference design can be found in “Appendix A: Reference Design Details.”

Multi-Rate SDI Reference Design

There are two versions of the reference design, one for each version of the DRU. Figure 11 shows the top level of the multi-rate SDI receiver reference design for version 1 of the DRU. This reference design was designed to run on the Xilinx SDV demo board, but can easily be adapted to other implementations. The top-level module is called sdv_multi_sdi_rx_v1. It contains the clock generation and distribution, the RocketIO module, the SD-SDI DRU, and a module called multi_sdi_rx containing the bulk of the multi-rate receiver logic.

Figure 12 shows the top level of the multi-rate SDI receiver design for version 2 of the DRU. It is nearly identical to version 1, with only minor changes involving the clocking of the DRU and downstream logic. Note that when using version 1 of the DRU, two global clock buffers are usually required per SDI receiver channel. With version 2, however, only one global clock is used by the receiver.

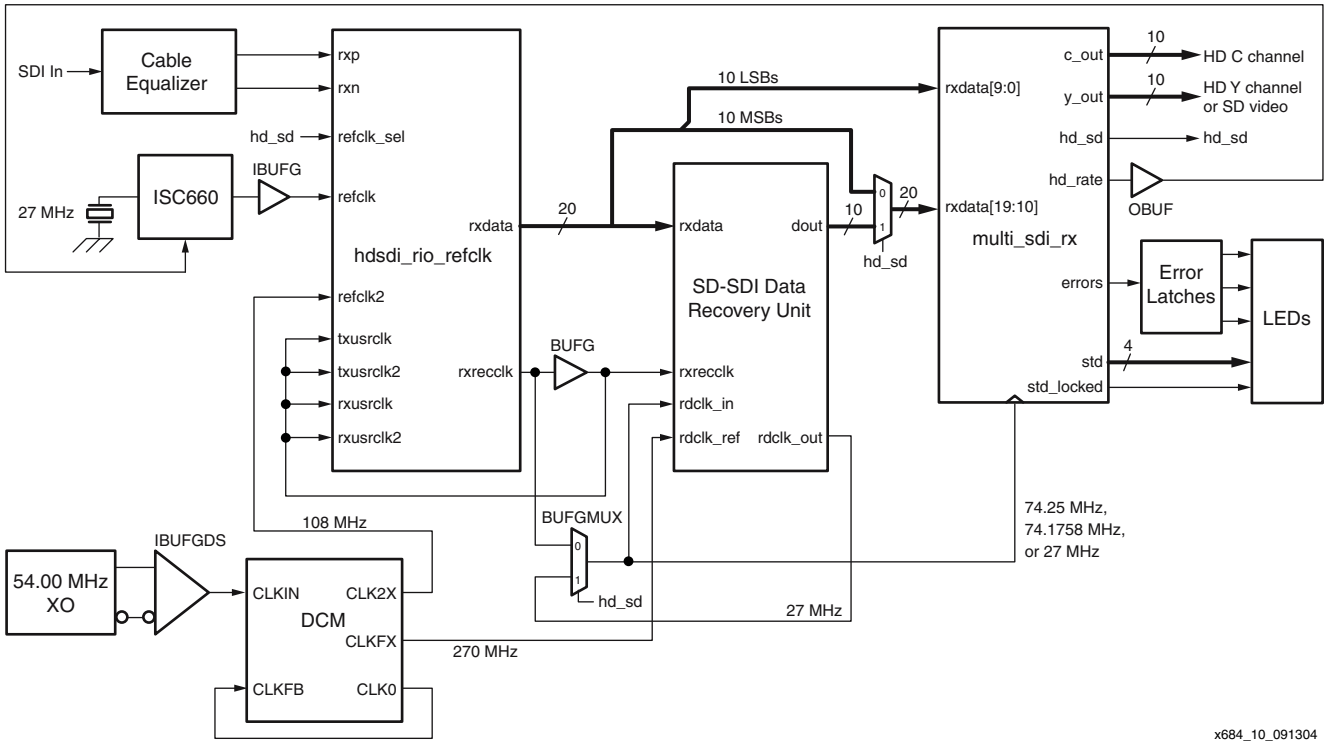

Figure 11: **Xilinx SDV Demo Board Multi-Rate SDI Receiver Reference Design (DRU Version 1)**

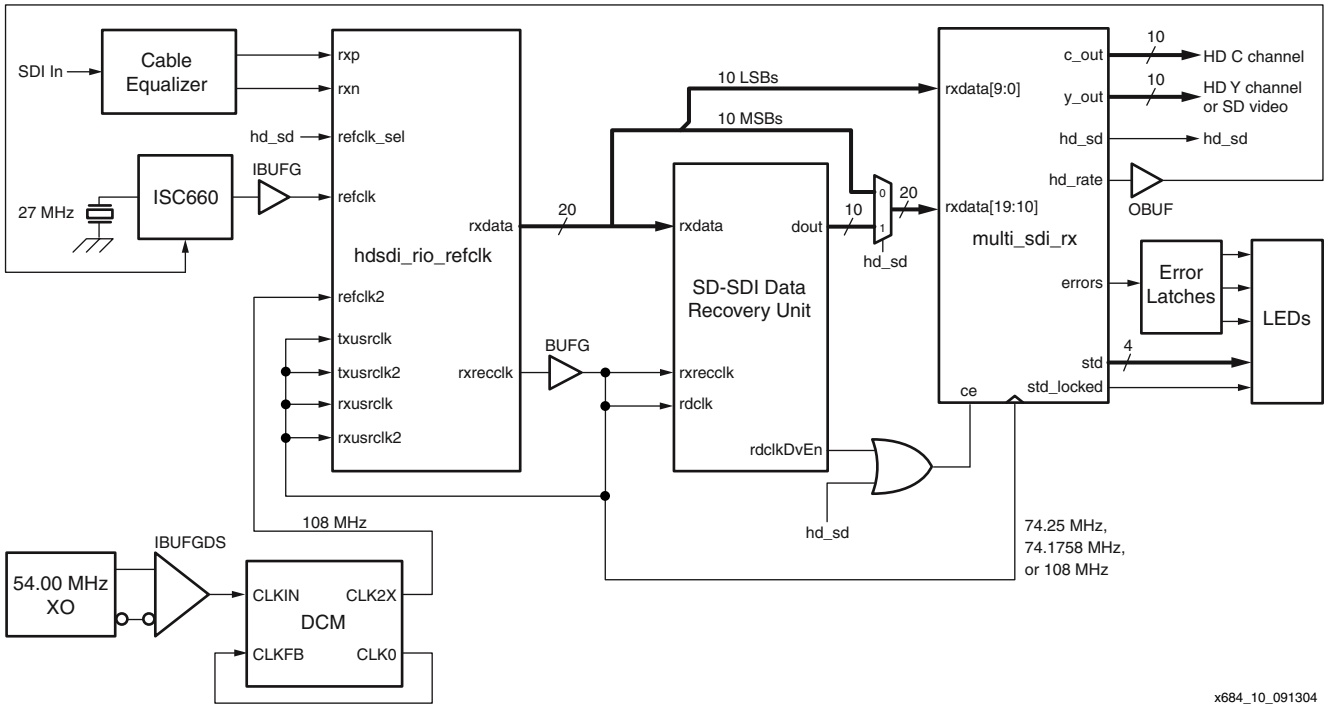

Figure 12: **Xilinx SDV Demo Board Multi-Rate SDI Receiver Reference Design (DRU Version 2)**

On the Xilinx SDV demo board, there are no provisions for bringing the received parallel video out of the Virtex-II Pro FPGA. So, the received video is simply checked for CRC and EDH errors to determine correct reception with LEDs indicating the error status.

Clocks (DRU Version 1)

On the SDV demo board, an ICS660 clock synthesizer is used to generate the two HD-SDI reference clocks of 74.25 MHz and 74.1758MHz from a 27 MHz reference crystal. For new designs, the newer ICS664-01 and ICS664-02 are recommended because they have better jitter performance than the ICS660. The ICS664-02 has a differential output and the ICS664-01 has a single-ended output. An output from the multi_sdi_rx module, called hd_rate, selects which of the two HD-SDI reference clocks is to be generated by the ICS660.

To recover 270 Mbps SD-SDI bitstreams with 8X over-sampling, the RocketIO module needs a 108 MHz reference clock. Ideally, a 108 MHz oscillator would be provided. However, the SDV demo board lacks such an oscillator. Instead, a 54 MHz oscillator is used. This oscillator is connected to the CLKIN of a DCM. The CLK2X output of the DCM provides the 108 MHz reference clock to the RocketIO. The CLKFX output of the DCM provides the 270 MHz rdclk_ref signal needed by the SD-SDI DRU by multiplying the 54 MHz input clock by 5.

While driving the RocketIO transceiver's reference clock input with a DCM is not normally recommended, keep in mind that, in this application, the CDR PLL is not used for clock and data recovery. Our experimentation with this configuration has shown that jitter on the RocketIO reference clock does affect the SD-SDI DRU and providing a low-jitter reference clock for the RocketIO provides better input jitter tolerance for the DRU. Using the 2X clock output of a DCM, given a very low-jitter clock source as the reference to the DCM, does produce satisfactory results. However, do not attempt to use the CLKFX output of the DCM for the RocketIO transceiver reference clock. The CLKFX output has too much jitter and will not work.

None of the outputs of the DCM need to be buffered by a BUFG buffer. The CLK0 and CLK2X outputs only drive single loads and the CLKFX output only drives 12 loads in the DRU.

The HD reference clock from the ICS660 is connected to the REFCLK2 input of the RocketIO transceiver. The 108 MHz SD reference clock from the DCM is connected to the REFCLK input of the RocketIO transceiver. An output from the multi_sdi_rx module, called hd_sd, is connected to the REFCLKSEL input of the Rocket transceiver to select between these two reference clock sources.

The RXRECCLK output of the RocketIO transceiver runs at the word rate of the HD video stream when receiving HD-SDI. In HD mode, a BUFGMUX passes RXRECCLK onto the multi_sdi_rx module as the receiver clock. In SD mode, the BUFGMUX passes the recovered clock from the SD-SDI DRU (rdclk_out) to the multi_sdi_rx module.

In SD mode, RXRECCLK is about the same frequency as the SD reference clocks (108 MHz), although its frequency can vary slightly as the transceiver's CDR PLL attempts to lock to the bitstream. RXRECCLK is buffered by a BUFG and sent to the rxrecclk input of the SD-SDI DRU to clock the data from the RXDATA output port of the transceiver into the DRU.

For both HD and SD modes, the buffered RXRECCLK output from the transceiver is connected to the four user clock inputs of the RocketIO transceiver. The data from the RXDATA port changes synchronously with the rising edge of the buffered RXRECCLK signal.

Clocks (DRU Version 2)

Version 1 of the DRU needs a high-frequency rdclk_ref so that the output phase jitter caused by the clock correction in the DRU can be kept to a minimum. Version 2 of the DRU does not have this requirement. The rdclk input to the DRU can be driven by a clock running at any integer or half-integer multiple of the 27 MHz SD clock rate from 4X to 10X.

The clock source for rdclk is also used to clock all downstream logic in the SD receiver. The rdclkDvEn output from the DRU is used in conjunction with rdclk to cause this downstream logic to run at 27 MHz when running in SD mode.

Usually, the most convenient way to use the version 2 DRU is use the RXRECCLK output of the RocketIO transceiver, buffered by a BUFG, as the rdclk source. This global RXRECCLK signal can then drive all downstream receiver logic.

This is particularly convenient when building a multi-rate receiver. In HD mode, RXRECCLK runs at exactly the word rate of the HD data being received by the RocketIO transceiver, and is the correct frequency needed to clock the downstream receiver logic. In SD mode, RXRECCLK runs at the same frequency as the REFCLK input to the RocketIO transceiver. The DRU generates a clock enable output that causes the downstream receiver logic, clocked by RXRECLK, to run at the 27 MHz SD word rate.

This eliminates the need for a global 270 MHz reference clock, and also eliminates the need for the BUFGMUX used in the version 1 reference design to multiplex the RXRECCLK from the RocketIO with the synthesized 27 MHz recovered clock from the DRU.

RocketIO Transceiver

The RocketIO transceiver primitive is included in the hdsdi_rio_refclk module. This module is a wrapper around the RocketIO GT_CUSTOM primitive. The module includes the bit swap functions on the transceiver's TXDATA input port and RXDATA output port. The bit swappers reverse the order of the input and output data vectors, compensating for the fact that the RocketIO transceiver sends and receives the MSB first, while SDI requires that the LSB be sent and received first.

SD-SDI DRU

At the current time, the source code for the DRU is not being provided. Instead, precompiled .ngc files for the DRU are provided in the reference design.

There are four `.ngc` files for the version 1 DRU, one for each oversampling rate from 8X to 10X. The file names are `oversample_DRU_nX`, where `n` is replaced by the oversampling rate.

There are 52 `.ngc` files for the version 2 DRU, 13 for each of the four supported oversampling rates. For each oversampling rate, there is a separate file for each rdclk division factor. The files are named `oversample_DRU_nX_clkdv_m`, where `n` represents the oversampling rate and `m` represents the clock divider used to produce the rdclkDvEn clock enable output. For example, the file `oversample_DRU_11X_clkdv5_5` is a DRU supporting 11X oversampling and a clock divider of 5.5. Table 1 shows all the various version 2 DRU files and indicates the RocketIO reference clock frequencies and DRU rdclk frequencies required for each when running at the standard 270 Mbps SD-SDI bit rate.

Table 1: **Version 2 DRU Filenames**

Oversample Rate	Clock Divider	Filename	RocketIO REFCLK	DRU RDCLK
8X	4	`oversample_DRU_8X_clkdv4`	108.0 MHz	108.0 MHz
8X	4.5	`oversample_DRU_8X_clkdv4_5`	108.0 MHz	121.5 MHz
8X	5	`oversample_DRU_8X_clkdv5`	108.0 MHz	135.0 MHz
8X	5.5	`oversample_DRU_8X_clkdv5_5`	108.0 MHz	148.5 MHz
8X	6	`oversample_DRU_8X_clkdv6`	108.0 MHz	162.0 MHz
8X	6.5	`oversample_DRU_8X_clkdv6_5`	108.0 MHz	175.5 MHz
8X	7	`oversample_DRU_8X_clkdv7`	108.0 MHz	189.0 MHz
8X	7.5	`oversample_DRU_8X_clkdv7_5`	108.0 MHz	202.5 MHz

Table 1: **Version 2 DRU Filenames *(Continued)***

Oversample Rate	Clock Divider	Filename	RocketIO REFCLK	DRU RDCLK
8X	8	oversample_DRU_8X_clkdv8	108.0 MHz	216.0 MHz
8X	8.5	oversample_DRU_8X_clkdv8_5	108.0 MHz	229.5 MHz
8X	9	oversample_DRU_8X_clkdv9	108.0 MHz	243.0 MHz
8X	9.5	oversample_DRU_8X_clkdv9_5	108.0 MHz	256.5 MHz
8X	10	oversample_DRU_8X_clkdv10	108.0 MHz	270.0 MHz
9X	4	oversample_DRU_9X_clkdv4	121.5 MHz	108.0 MHz
9X	4.5	oversample_DRU_9X_clkdv4_5	121.5 MHz	121.5 MHz
9X	5	oversample_DRU_9X_clkdv5	121.5 MHz	135.0 MHz
9X	5.5	oversample_DRU_9X_clkdv5_5	121.5 MHz	148.5 MHz
9X	6	oversample_DRU_9X_clkdv6	121.5 MHz	162.0 MHz
9X	6.5	oversample_DRU_9X_clkdv6_5	121.5 MHz	175.5 MHz
9X	7	oversample_DRU_9X_clkdv7	121.5 MHz	189.0 MHz
9X	7.5	oversample_DRU_9X_clkdv7_5	121.5 MHz	202.5 MHz
9X	8	oversample_DRU_9X_clkdv8	121.5 MHz	216.0 MHz
9X	8.5	oversample_DRU_9X_clkdv8_5	121.5 MHz	229.5 MHz
9X	9	oversample_DRU_9X_clkdv9	121.5 MHz	243.0 MHz
9X	9.5	oversample_DRU_9X_clkdv9_5	121.5 MHz	256.5 MHz
9X	10	oversample_DRU_9X_clkdv10	121.5 MHz	270.0 MHz
10X	4	oversample_DRU_10X_clkdv4	135.0 MHz	108.0 MHz
10X	4.5	oversample_DRU_10X_clkdv4_5	135.0 MHz	121.5 MHz
10X	5	oversample_DRU_10X_clkdv5	135.0 MHz	135.0 MHz
10X	5.5	oversample_DRU_10X_clkdv5_5	135.0 MHz	148.5 MHz
10X	6	oversample_DRU_10X_clkdv6	135.0 MHz	162.0 MHz
10X	6.5	oversample_DRU_10X_clkdv6_5	135.0 MHz	175.5 MHz
10X	7	oversample_DRU_10X_clkdv7	135.0 MHz	189.0 MHz
10X	7.5	oversample_DRU_10X_clkdv7_5	135.0 MHz	202.5 MHz
10X	8	oversample_DRU_10X_clkdv8	135.0 MHz	216.0 MHz
10X	8.5	oversample_DRU_10X_clkdv8_5	135.0 MHz	229.5 MHz
10X	9	oversample_DRU_10X_clkdv9	135.0 MHz	243.0 MHz
10X	9.5	oversample_DRU_10X_clkdv9_5	135.0 MHz	256.5 MHz
10X	10	oversample_DRU_10X_clkdv10	135.0 MHz	270.0 MHz
11X	4	oversample_DRU_11X_clkdv4	148.5 MHz	108.0 MHz
11X	4.5	oversample_DRU_11X_clkdv4_5	148.5 MHz	121.5 MHz
11X	5	oversample_DRU_11X_clkdv5	148.5 MHz	135.0 MHz
11X	5.5	oversample_DRU_11X_clkdv5_5	148.5 MHz	148.5 MHz

Table 1: **Version 2 DRU Filenames *(Continued)***

Oversample Rate	Clock Divider	Filename	RocketIO REFCLK	DRU RDCLK
11X	6	oversample_DRU_11X_clkdv6	148.5 MHz	162.0 MHz
11X	6.5	oversample_DRU_11X_clkdv6_5	148.5 MHz	175.5 MHz
11X	7	oversample_DRU_11X_clkdv7	148.5 MHz	189.0 MHz
11X	7.5	oversample_DRU_11X_clkdv7_5	148.5 MHz	202.5 MHz
11X	8	oversample_DRU_11X_clkdv8	148.5 MHz	216.0 MHz
11X	8.5	oversample_DRU_11X_clkdv8_5	148.5 MHz	229.5 MHz
11X	9	oversample_DRU_11X_clkdv9	148.5 MHz	243.0 MHz
11X	9.5	oversample_DRU_11X_clkdv9_5	148.5 MHz	256.5 MHz
11X	10	oversample_DRU_11X_clkdv10	148.5 MHz	270.0 MHz

The DRU provides a 10-bit recovered data word on its output. This 10-bit vector is multiplexed with the 20-bit RXDATA word from the RocketIO module and connected to the 20-bit data input of the multi_sdi_rx module. The 10-bit word from the SD-SDI DRU must be connected to the 10 most significant bits of the multi_sdi_rx module's 20-bit d input port in SD mode.

multi_sdi_rx

The multi_sdi_rx module contains all of the logic needed to descramble and frame the HD or SD video data and to check the video for errors.

The multi_sdi_decoder module descrambles the 20-bit HD data and the 10-bit SD data. The output of the decoder module is normally connected to the multi_sdi_framer module, although the decoder module can be bypassed for debugging purposes by asserting the dec_bypass input.

The multi_sdi_framer module implements the framing algorithm for both HD-SDI and SD-SDI. The Y channel output from the framer carries the 10-bit SD video when running in SD-SDI mode.

The framer resynchronizes immediately to a new EAV or SAV position if the frame_en input is asserted. The frame_en input can be used to implement various filtering functions to filter out single or multiple anomalous EAV or SAV sequences. One simple approach is to connect the nsp output of the multi_sdi_framer module to the frame_en input. The nsp output becomes asserted if the framer detects a timing reference signal at a new starting bit position when frame_en is not asserted and it stays asserted until the framer is synchronized to the bitstream. By connecting nsp to frame_en, a single anomalous timing reference does not cause the framer to resynchronize, but two consecutive timing references at a new starting position does cause the framer to resynchronize.

Note that there are two different implementations of the framer module provided in the reference design: multi_sdi_framer and multi_sdi_framer_mult. The latter uses several MULT18X18 multiplier blocks, available in the Virtex-II Pro, to implement the barrel shifter, reducing the amount of FPGA fabric required for the framer design.

In HD-SDI mode, the hdsdi_rx_crc module checks the video from the framer for CRC errors. This module also captures the line numbers embedded in the video stream and provides them on output ports. This module is fully described in XAPP681.

The hdsdi_autodetect_ln module detects the current format of the HD video and provides a 4-bit code indicating which video format is being received. Refer to XAPP681 for a detailed description of this module.

In SD-SDI mode, the EDH processor from XAPP299 is used to detect the SD video format and to check the video stream for EDH errors. Refer to XAPP299 for a detailed description of this module. For simplicity, many of the inputs and outputs of the EDH processor are not used in this design. All of the various function of the EDH processor are available by simply connecting the appropriate inputs and outputs.

A set of MUXes selects the appropriate HD or SD signals to drive the outputs of the multi_sdi_rx module. The various outputs of the framer module, such as y, trs, xyz, eav, and sav, are valid for both SD-SDI and HD-SDI, but in SD-SDI mode these MUXes select outputs from the EDH processor instead of from the framer. This is because the EDH processor delays the video stream (and modifies the video stream to correct EDH and TRS errors). In order to have the various timing signals, such as eav and sav, line up with the delayed video from the EDH processor, these signals are taken from the EDH processor rather than the framer in SD-SDI mode.

The multi_sdi_rx_autorate module is responsible for selecting the correct reference clock for the RocketIO transceiver to match the incoming bitstream frequency. It generates the hd_sd and hd_rate signals that, at the top level of the design, select between the various reference clocks for the RocketIO transceiver. This module cycles through the two HD-SDI bit rates and the 270 Mbps SDI bit rate until valid data is detected coming from the framer module.

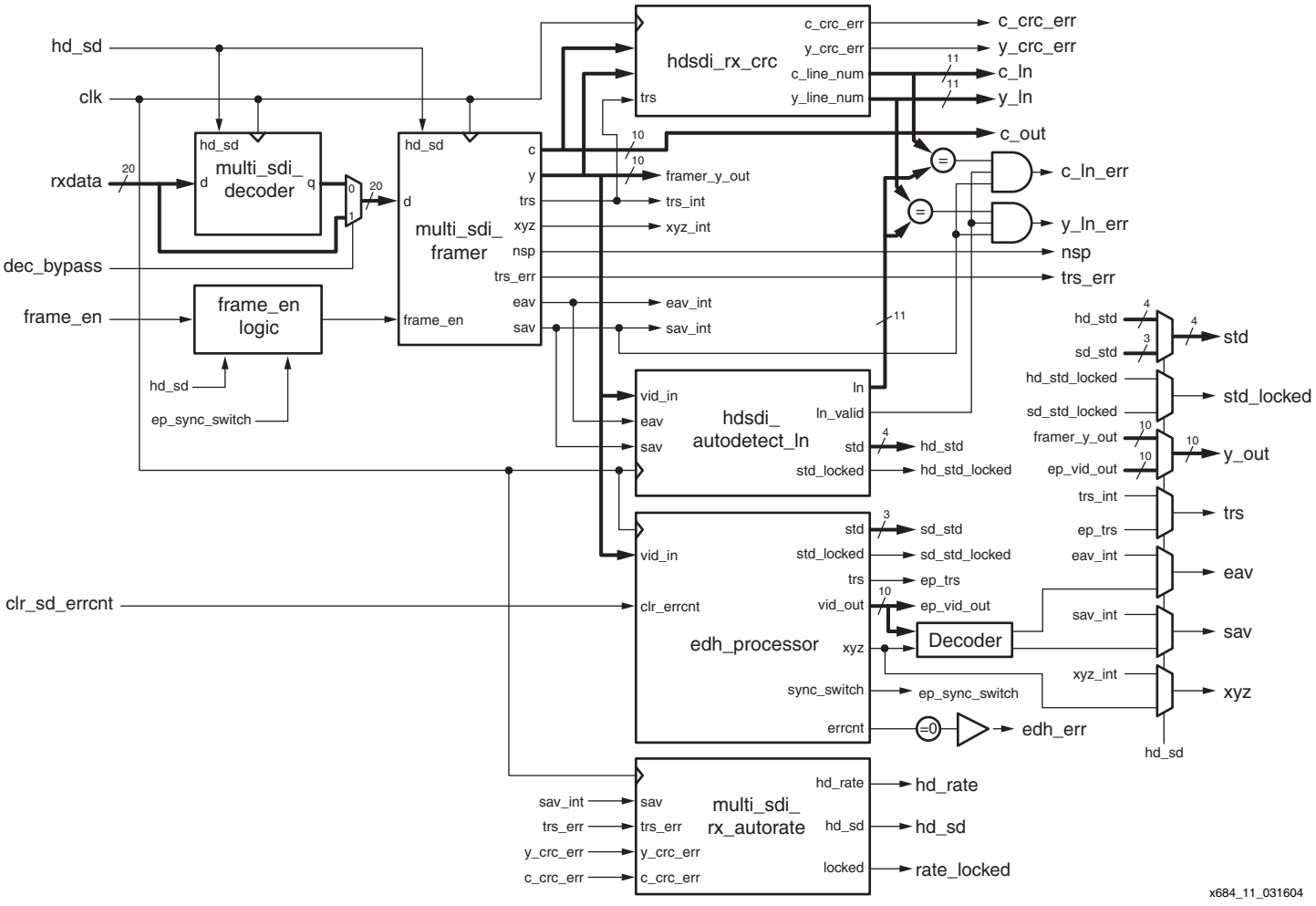

Figure 13: **multi_sdi_rx Block Diagram**

Results

The results shown in Table 2 were obtained using XST running under ISE 6.2. Area optimization was used. All designs were able to meet the necessary timing constraints using a Virtex-II Pro -5 speed grade device both HD-SDI bit rates and for 270 Mbps SD-SDI. In order to support 360 Mbps SD-SDI, the minimum speed grade required is -6, because 8X over-sampling requires RocketIO transceivers capable of running at 2.88 Gbps.

About half of the FPGA resources used by the receiver design are consumed by the SD-SDI EDH processor, even with many of the EDH processor features left unused and optimized away by the tools. To illustrate this, a multi-rate receiver with all the features except the EDH processor is also shown in Table 2.

Table 2: **Reference Design Implementation Sizes**

Reference Design	FFs	LUTs	MULT18X18s
Multi-rate SDI Rx for SDV demo board using multi_sdi_framer module	945	1685	0
Multi-rate SDI Rx for SDV demo board using multi_sdi_framer_mult module	945	1602	6
Multi-rate SDI Rx for SDV board without EDH processor using multi_sdi_framer_mult module	517	896	6

Conclusions

This application note describes a technique that can be used to allow SD-SDI bitstreams to be received using the RocketIO multi-gigabit transceivers in the Virtex-II Pro FPGA family. By combining this technique with the HD-SDI receiver design described in XAPP681, a multi-rate SDI receiver can be implemented that supports both SD-SDI and HD-SDI.

All Virtex-II Pro devices contain multiple RocketIO transceivers, making it possible to implement multiple multi-rate SDI receivers in a single Virtex-II Pro device. For applications that require multiple SDI receivers, this can provide a high level of integration, saving board space, power, and reduce cost as compared to discrete multi-rate SDI receiver implementations.

References

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 web site: http://www.smpte.org.
2. The ITU-R BT.601-5 standard can be purchased from the International Telecommunication Union at http://www.itu.int/itudoc/itu-r/rec/bt/.
3. Xilinx application note XAPP683, "Multi-Rate HD/SD-SDI Transmitter Using Virtex-II Pro RocketIO Multi-Gigabit Transceivers" by John F. Snow
4. The Xilinx SD-SDI applications notes are: XAPP247, "SDI Physical Layer"; XAPP288, "SDI Video Decoder"; XAPP298, "SDI Video Encoder"; XAPP299, "SDI Ancillary Data and EDH Processors"; and XAPP625, "Video Standard Detector and Flywheel Decoder".
5. XAPP681, "HD-SDI Receiver Using Virtex-II Pro RocketIO Multi-Gigabit Transceivers" by John F. Snow.
6. The Xilinx SDV Demo board is available from Cook Technologies (part number CTXIL103). Further information is available at http://www.cook-tech.com.
7. The Gennum GS1524 data sheet is found at: http://www.gennum.com/vb/pdffiles/14976DOC.pdf.
8. UG024, *RocketIO Transceiver User Guide*, Xilinx, Inc.
9. XAPP682, "HDTV Video Pattern Generator" by John F. Snow.
10. XAPP248, "Digital Video Test Pattern Generators" by John F. Snow.

Appendix A: Reference Design Details

This appendix provides more details about the multi_sdi_decoder, multi_sdi_framer, and multi_sdi_rx_autorate modules.

multi_sdi_decoder

This module (see Figure A-1) implements the SDI decoder algorithm for both SD-SDI and HD-SDI. It is based on the hdsdi_decoder module from XAPP681. In fact, the only differences are MUXes (labeled Bit Selection) preceding the NRZI-to-NRZ converter and descrambler sections that select a different set of bits for use in SD-SDI mode.

The SD-SDI input data is only 10 bits wide and is required to be placed on the most-significant 10 bits of the d input port. The MUXes select the proper set of bits to perform the NRZI-to-NRZ and descrambling functions on a 10-bit vector instead of the 20-bit wide HD-SDI vector.

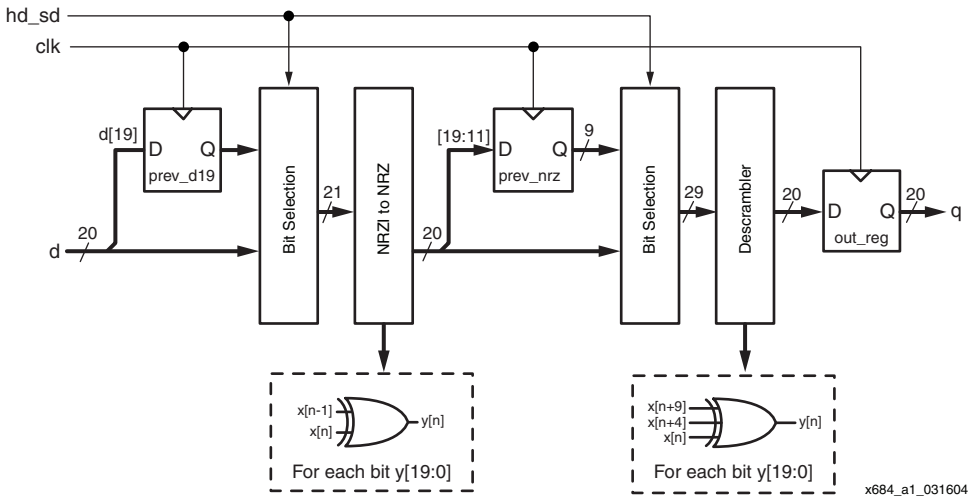

Figure A-1: **multi_sdi_decoder Block Diagram**

multi_sdi_framer

This module (see Figure A-2) implements the SDI framer function for both SD-SDI and HD-SDI. It is based on the hdsdi_framer module from XAPP681. The only differences are:

- A minor change to the TRS detect logic to mask out detection of a TRS in the least-significant 10 bits of the input data during SD mode
- And, a MUX in front of the barrel shifter to select the proper input vector to the barrel shifter based on the hd_sd input signal.

An alternate version of the framer, called multi_sdi_framer_mult, uses six MULT18X18 blocks to implement the barrel shifter, providing some savings in the amount of FPGA fabric consumed by the framer module.

Refer to XAPP681 for more complete descriptions of the framer modules.

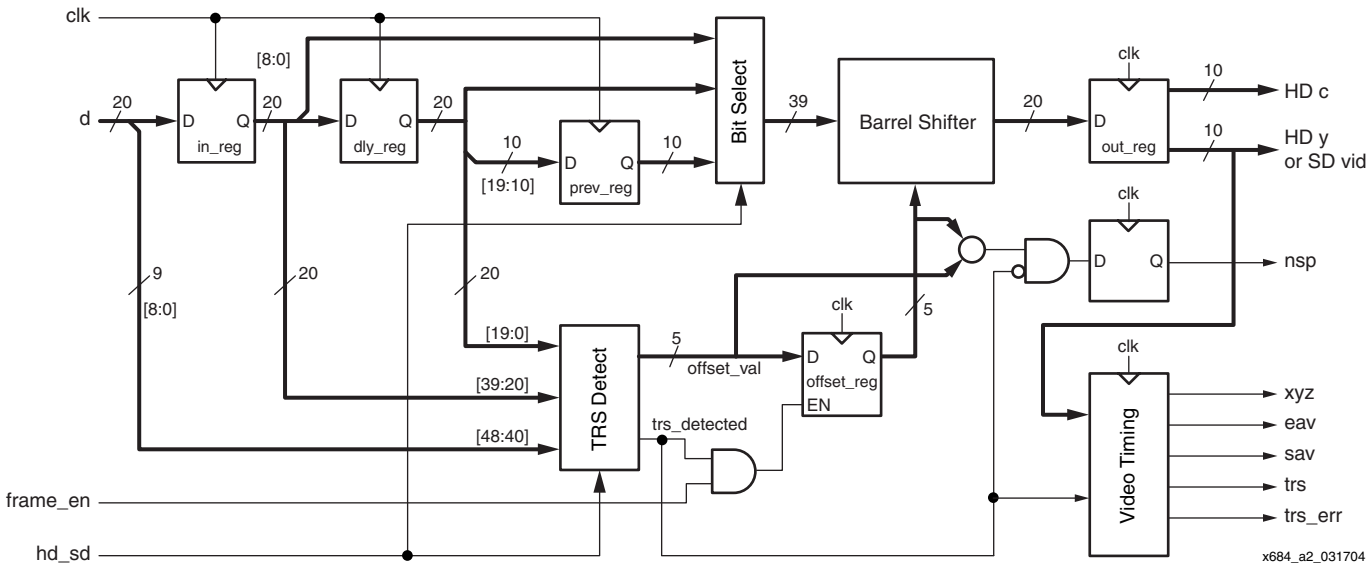

Figure A-2: **multi_sdi_framer Block Diagram**

multi_sdi_rx_autorate

This module selects the correct bit rate for the multi-rate SDI receiver, cycling through the possible reference clock sources to the RocketIO transceiver until the receiver locks to the incoming bitstream. This module is similar to the hdsdi_autorate_rx module from XAPP681. The XAPP681 design simply switches between the two HD-SDI bit rates while the design used here also includes the 270 Mbps SD-SDI bit rate.

The module relies on missing or erroneous SAVs and HD-SDI CRC errors to determine when the receiver is locked to the incoming bitstream. As described in XAPP681, SAV errors alone are not sufficient for distinguishing whether the correct HD-SDI reference clock is selected. However, SAV errors are sufficient for determining if the correct selection is made between HD-SDI and SD-SDI.

The module consists of a state machine plus an error counter that tracks the number of consecutive video lines with erroneous or missing SAVs or CRC errors (HD-SDI only). When the maximum error threshold is exceeded, the state machine switches to the next reference clock in the cycle and tries again. The design allows the use of different maximum error thresholds depending on whether the receiver is already locked to the bitstream or whether it is unlocked and seeking the correct reference clock. This allows a bigger error threshold to be applied in the locked case to prevent accidentally unlocking in the presence of a burst of noise. It also allows a smaller error threshold to be applied to the unlocked case to make the search process quicker. The error thresholds are defined by the MAX_ERRS_LOCKED and MAX_ERRS_UNLOCKED parameters in the module, making it easy to modify these error thresholds. When modifying these parameters, make sure that the width of the error counter, defined by the ERRCNT_WIDTH parameter, is sufficient to handle the largest threshold value.

Figure A-3 is the state diagram for the finite state machine in the multi_sdi_rx_autorate module.

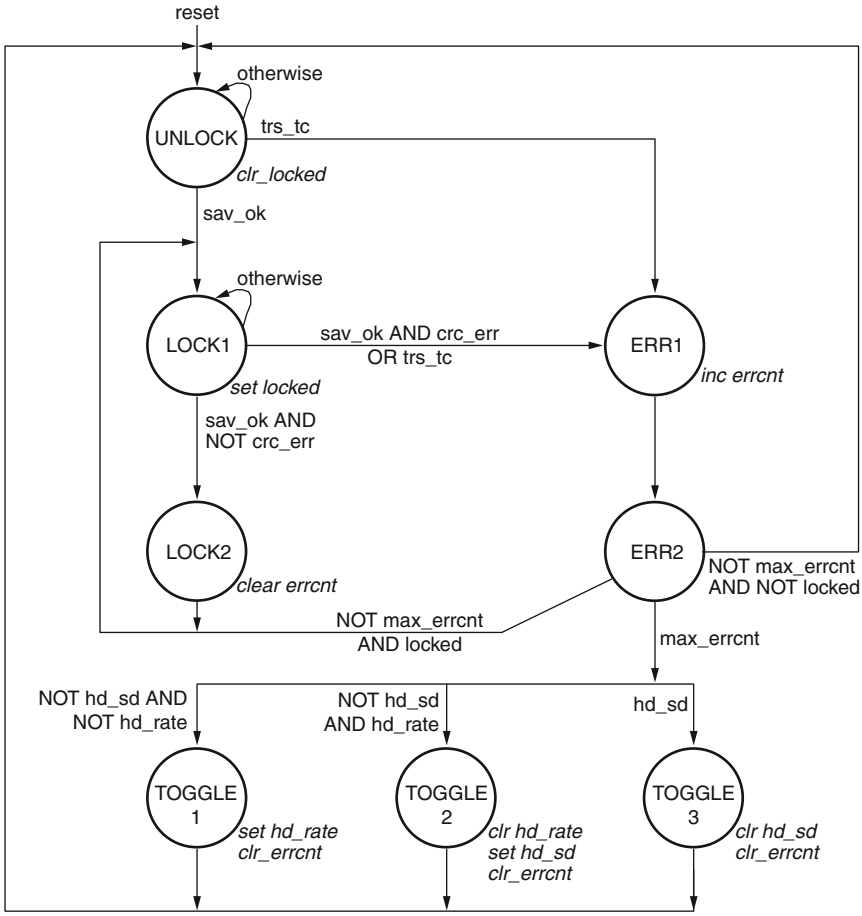

Figure A-3: **multi_sdi_rx_autorate State Diagram**

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/19/04	1.0	Initial Xilinx release.
09/22/04	2.0	• Added new text, together with new Figure 9, Figure 12, and Table 1, differentiating two methods of implemeting the data recovery unit (DRU). • Corrected name of bottom waveform in Figure 8 to **dout**. • Added text to Figure 6 and to section "Cable Equalization" noting that some Virtex-II Pro devices have internal power filter capacitors for the VTRX, VTTX, AVCCAUXRX, and AVCCAUXTX signals of each RocketIO transceiver.