XILINX®

XAPP247 (v1.0) December 02, 2003

# Serial Digital Interface (SDI) Physical Layer Implementation

Author: John F. Snow

## Summary

The Serial Digital Interface (SDI) standard describes how to transport standard-definition digital video serially over coax cable. Equipment based on the SDI standard is commonly used in broadcast studios and video production centers.

This application note describes implementations of the SDI physical layer using Xilinx FPGAs. The main topics covered by this application note are: cable equalization, clock and data recovery, jitter reduction, clock multiplication for the transmitter, and cable drivers.

## Introduction

SDI is defined by the SMPTE 259M and the ITU-R BT.656 standards.[1,2] SDI is widely used in broadcast studios and video production centers to transport digital video serially over coax cable. SMPTE 259M defines four standard bit rates for SDI ranging from 143 Mb/s to 360 Mb/s with the 270 Mb/s bit rate being, by far, the most common. Another standard, SMPTE 344M, adds a 540 Mb/s bit rate for SDI.[1] However, use of the 540 Mb/s bit rate is not yet widespread.

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

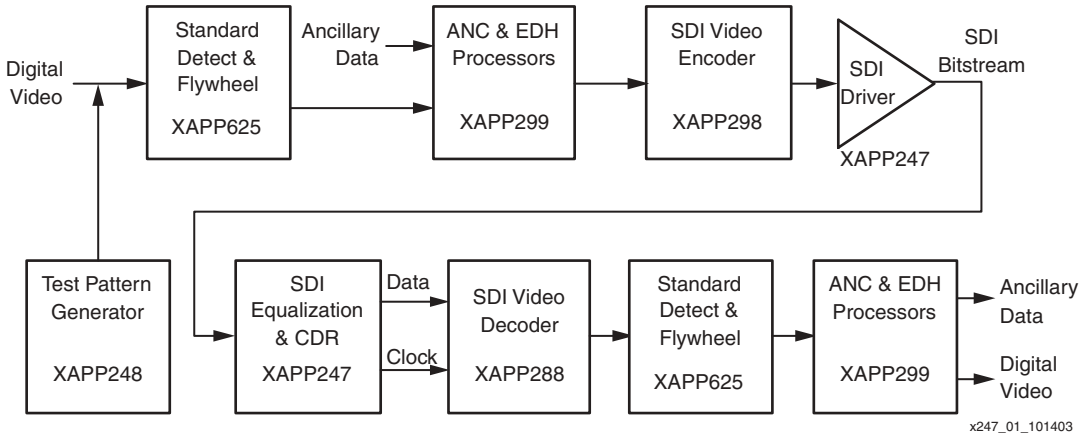



*Figure 1:* **SDI Block Diagram and Application Notes**

The other application notes cover SDI encoding, decoding, error detection, and ancillary data handling. This application note focuses on the physical layer and the details of sending and receiving the encoded SDI bitstream. Five different aspects of the physical layer are covered in this application note. The block diagram shown in Figure 2 shows where these topics fit in a piece of SDI equipment designed to receive and then retransmit an SDI bitstream (SDI pass-through).

- *Cable Equalization*: SDI receivers use adaptive cable length equalization to compensate for signal loss in the coax cable. An external cable equalizer can be used to equalize the

bitstream before it is received by the FPGA. This application note discusses techniques for interfacing cable equalizers to Xilinx FPGAs.

- *Clock & Data Recovery (CDR)*: After equalization, the SDI receiver must recover the data from the bitstream. Asynchronous data recovery is usually done by oversampling the bitstream and then looking for transitions. The data recovery unit tries to sample each bit in the middle of the bit period and as far away from the bit transitions as possible. Typically, a Phase-Locked Loop (PLL) is also used with the data recovery unit to recover the clock. However, in some cases, it can be assumed that the receiver and transmitter are running at the same frequency. If this is the case, then the receiver might not need to recover the clock. Instead, data-only recovery techniques can be used. This application note discusses several clock and data recovery and data-only recovery techniques suitable for receiving SDI bitstreams with Xilinx FPGAs.
- *Jitter Reduction*: Parallel digital video supplied to an SDI transmitter either from an external video source or from an SDI receiver can contain a significant amount of jitter. An SDI transmitter is required to send the SDI bitstream with very little jitter. This may require the transmitter to reduce the amount of jitter on the video stream before transmission.
- *Clock Multiplication*: The SDI transmitter needs a bit-rate clock for its serializer. This usually requires the transmitter to multiply the incoming word-rate video clock by 10 to obtain the bit-rate clock. This multiplication must be done without adding too much jitter.
- *Cable Driver*: This application note describes a method of interfacing Xilinx FPGAs to video coax cables.

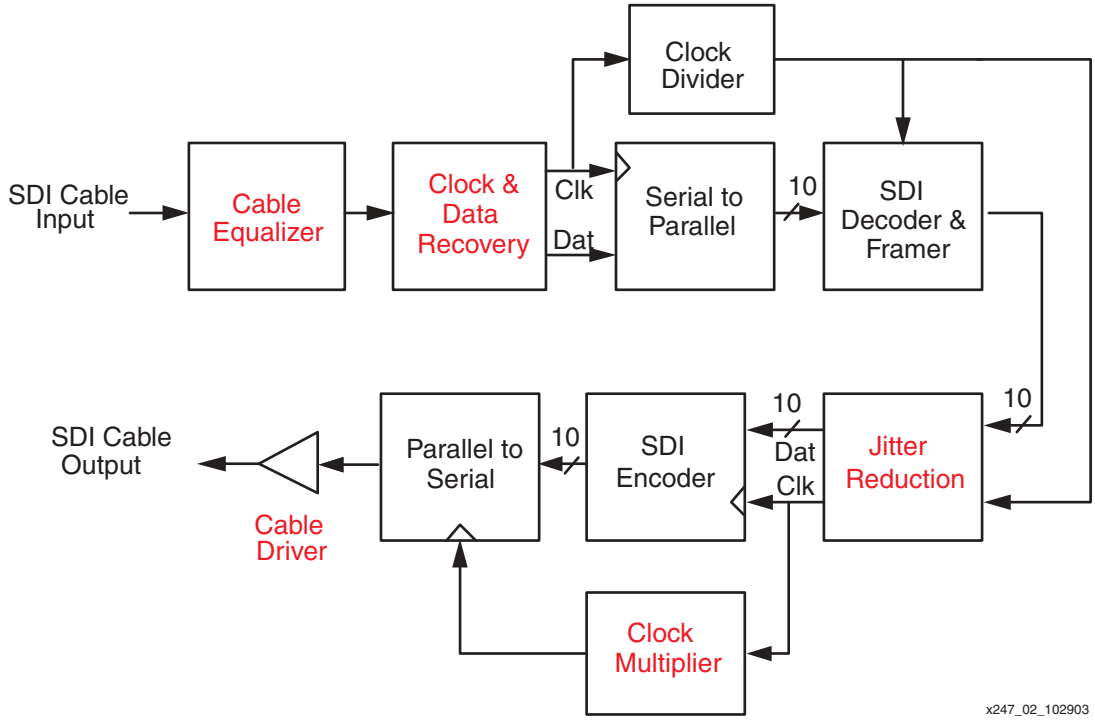



*Figure 2:* **SDI Pass-Through Block Diagram**

The reference designs associated with this application note have been tested on the Xilinx SDV demo board. This demo board is available from Cook Technologies (part number CTXIL103). Information is available at www.cook-tech.com.

This application note is focused on implementations of SDI interfaces built using the fabric of Xilinx FPGAs. However, it is also possible to implement SDI receivers and transmitters using the Rocket IO transceivers in Virtex-II Pro devices. Future Xilinx application notes XAPP683 and XAPP684) will describe how to implement SDI using Virtex-II Pro Rocket IO transceivers.

## Measuring SDI Transmitter and Receiver Performance

Before discussing how to implement SDI interfaces using Xilinx FPGAs, it is useful to have an understanding of the metrics used by the video industry to evaluate the performance of SDI transmitters and receivers. A very informative discussion of measuring SDI interface performance can be found in EBU Technical Document 3283 1996.[3]

The quality of an SDI transmitter is measured in two primary ways: the electrical performance of the output driver and the transmitter's output jitter. The transmitter's output driver must meet all of the electrical specifications defined in the SDI standard. These requirements are listed in the "SDI Cable Driver" section of this application note. The transmitter must also produce an output bitstream that has less than 0.2 UI(1) of output jitter. Lower transmitter output jitter results in more jitter margin in the SDI link.

SDI receiver performance is usually measured in three areas:

- Tolerance of waveform attenuation and distortion caused by long runs of coax cable
- Tolerance of jitter present on the input bitstream
- Tolerance of the SDI pathological waveforms.

The use of an SDI adaptive cable length equalizer takes care of the signal attenuation and phase distortion introduced by the coax cable.

Input jitter tolerance is the ability of the clock and data recovery unit to correctly receive an SDI bitstream with a significant amount of jitter distortion. The SMPTE 259M specification does not contain any requirement for how much input jitter must be tolerated by an SDI receiver. However, the standard does allow an SDI transmitter to have as much as 0.2 UI peak-to-peak of output jitter. Additional jitter is added by various sources such as reflections caused by impedance mismatches on the PCB board, connectors, and cable. An SDI receiver should be able to tolerate some amount of input jitter above the 0.2 UI allowed at the transmitter output. A good SDI receiver is usually capable of tolerating at least 0.5 UI of input jitter.(2)

The pathological waveforms are defined by the SMPTE recommend practice RP 178-1996.[1] These are two different worst-case waveforms that can be created by an SDI encoder and must be tolerated by SDI receivers.

One pathological waveform is poorly DC balanced and stresses the cable equalizer. This pattern is the top waveform in Figure 3. The waveform has one high bit followed by 19 low bits. This pattern can be repeated for the entire active portion of one video line. The opposite polarity is equally possible (one low bit followed by 19 high bits). A properly designed cable equalizer must be able to deal with this waveform.

The second pathological waveform is a square wave with a period 40 bits long as shown in Figure 3. This square wave can repeat for the entire active portion of one line of video. Because of its relatively low density of transitions, this waveform can cause problems for the PLL in the CDR section of the SDI receiver. A properly designed CDR unit must be able to stay locked to the bitstream in the presence of this waveform.

1. UI stands for Unit Interval and is a term commonly used when discussing high-speed serial interfaces. One UI is equal to the time required to transmit one bit across the serial interface. At 270 Mb/s, 1.0UI is approximately 3.7ns and 0.2 UI is 0.2 * 3.7 ns ≈ 740 ps. The UI value is relative to the bit rate. At 360 Mb/s, 0.2 UI is about 556 ps.
2. The input jitter tolerance of an SDI receiver usually varies with the frequency of the jitter. An SDI receiver can usually tolerate more low frequency jitter than high frequency jitter. Typically, an SDI receiver can tolerate several UI of jitter when the sinusoidal jitter frequency is below 1 kHz. The jitter tolerance drops to something less than 1.0 UI as the jitter frequency increases above 10 kHz.

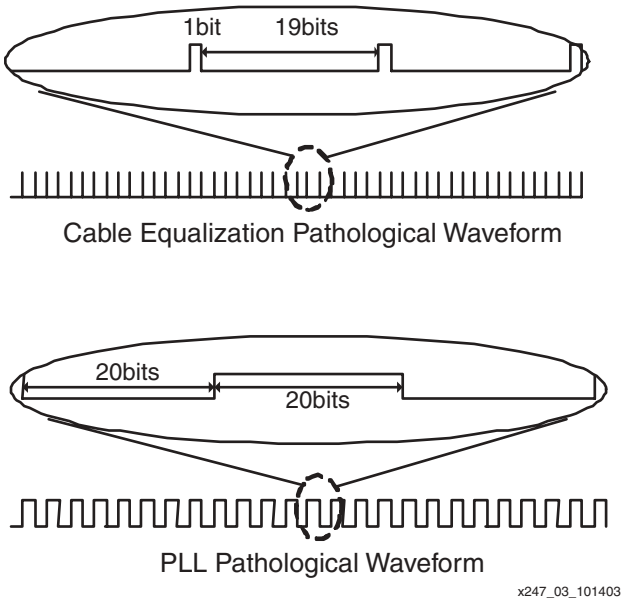



*Figure 3:* **Pathological Waveforms**

Another type of jitter measurement applies when a piece of equipment receives an SDI bitstream and then retransmits this bitstream. The “jitter transfer function” of such a piece of equipment measures how much jitter present on the input SDI bitstream is transferred through the equipment to the output SDI bitstream for each different frequency component in the jitter spectrum. The jitter transfer function is very dependent upon the ability of the system to reduce jitter present on the input bitstream before re-transmission. Jitter reduction techniques, such as those described in the “Jitter Reduction” section of this application note, may be used to modify the jitter transfer function of the system.

## SDI Cable Equalization

The SDI standard allows digital video to be transmitted over long runs of video coax cable, up to 300 meters. Even with high quality cable, the signal will be significantly attenuated and distorted after passing through 300 meters of cable. The attenuation is frequency dependent, with the higher frequency components attenuated more than the lower frequency components of the signal. The coax cable will also introduce frequency dependent phase shifting with the higher frequency components phase shifted more than the lower frequency components.

The SDI standard requires that SDI receivers must be capable of working with input signals that have been attenuated by as much as 30 dB at one half the clock rate of the bitstream. To compensate for this amount of signal loss in the cable and to compensate for the frequency dependent phase shift introduced by the cable, SDI receivers usually employ an adaptive cable length equalizer circuit. An adaptive cable length equalizer actively monitors the incoming signal and compensates for signal attenuation and phase shift caused by any cable length up to the maximum allowed.

SDI compliant adaptive cable length equalizers are not currently available in Xilinx FPGAs. It is recommended that an external SDI equalization circuit be used between the cable connector and the FPGA.

Figure 4 shows a National Semiconductor CLC014 SDI equalizer interfaced to a Xilinx Virtex-II Pro FPGA. The CLC014, like most SDI equalization chips, has differential PECL outputs. The peak-to-peak differential signal swing is compatible with some of the LVDS input standards supported by many Xilinx FPGAs. However, the common mode voltage is above the maximum allowed by low voltage Xilinx FPGAs, such as Virtex-II and Virtex-II Pro. In Figure 4, AC coupling is used to remove the DC offset from the signal. The parallel termination resistor

networks on each leg of the LVDS input to the FPGA not only terminate the signal, but also bias the FPGA inputs to a common mode voltage that is compatible with the LVDS input buffers. The Virtex-II and Virtex-II Pro FPGAs have termination resistors for the LVDS input buffer built into the IOBs. These can be used instead of external resistor networks to terminate and bias the AC coupled signals from the equalizer.

The CLC014 generates a carrier detect (CD) signal. CD is asserted when a received signal is detected. In the example in Figure 4, the CD output of the CLC014 is connected to an FPGA input. A voltage divider is used to make the CD voltage level compatible with the FPGA input.

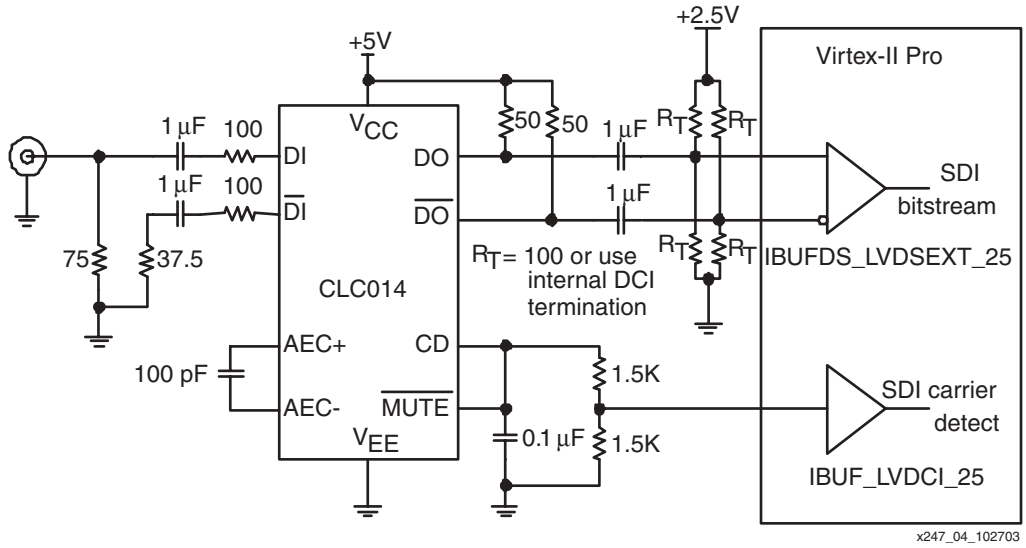



*Figure 4:* **Cable Equalization Example**

The Xilinx SDV demo board uses the CLC014 cable equalizer as shown in Figure 4. This design has been tested with cable lengths in excess of 300 meters.(1)

## Jitter Reduction

Prior to sending video data using an SDI transmitter, it might be necessary to reduce the amount of jitter present on the video stream. Digital video is usually provided to the SDI transmitter as 8-bit or 10-bit parallel video data words with an accompanying video clock running at the video word rate. Depending on the source of this video, the video clock can have a large amount of jitter. The SMPTE and ITU parallel video standards allow a large amount of jitter on the parallel video clock. For example, SMPTE 125M, a standard that defines an SDI compatible parallel video format for 4:2:2 component digital video, allows for as much as ±3 ns peak-to-peak clock jitter.[1] In contrast, the SDI standard requires that the bitstream generated by the SDI transmitter have no more than 0.2 UI peak-to-peak jitter (about 740 ps at 270 Mb/s). Simply multiplying the parallel video clock by 10 to obtain a bit-rate clock without also implementing jitter reduction would result in a bit-rate clock with as much as 0.8 UI of jitter, four times more than the entire output jitter budget of the SDI transmitter.

There are many sources of jitter in the digital video stream. Analog video can be passed through a video decoder to convert it to digital video. The video decoder may generate the digital video clock using a PLL that locks to the horizontal sync pulses present in the analog video. These circuits can produce a video clock with significant jitter.

Jitter will also be present on the video stream recovered by an SDI receiver. SDI receivers usually are able to correctly receive SDI bitstreams that have 0.5 UI of jitter or more. The

1. Cable length testing of the SDI receiver was conducted on the Xilinx SDV demo board using a National Semiconductor CLC014 cable equalizer combined with the XAPP250-based SDI receiver checking for EDH errors.

recovered clock generated by the receiver's CDR circuit will usually contain much of the jitter present on the incoming bitstream. So, in most cases, it is not possible to use the recovered clock from an SDI receiver directly as the clock for an SDI transmitter.

In all of these cases, the SDI transmitter should implement jitter reduction on the video before transmission. The classic jitter reduction technique for digital video is shown Figure 5. The video clock is passed through a PLL designed to reduce jitter. The input video clock is also used to write the video data into an asynchronous FIFO. The data is read out of the FIFO using the low-jitter clock from the PLL. The jitter has been removed from the clock and the data has been resynchronized to the low-jitter clock. This technique is known as "reclocking".

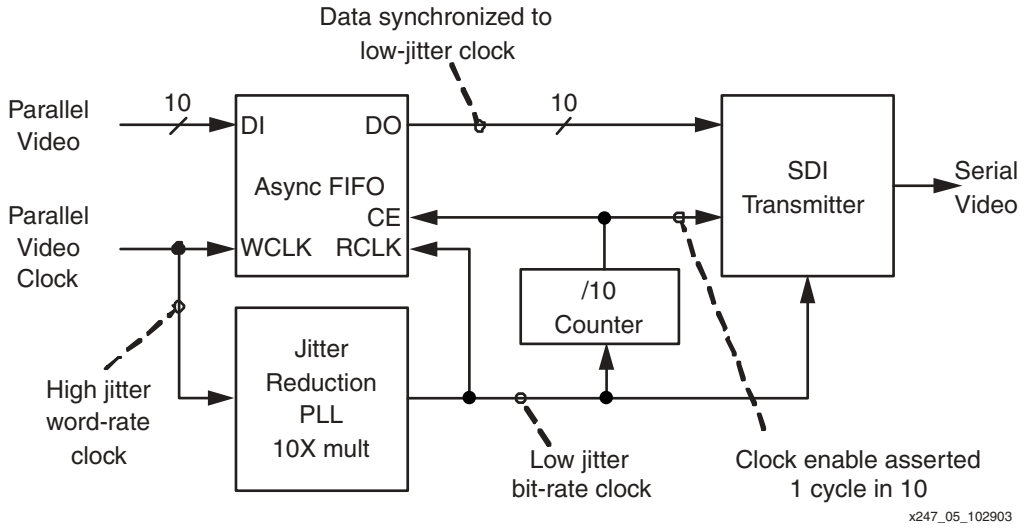



*Figure 5:* **Jitter Reduction**

The SDI transmitter must generate a bit-rate clock by multiplying the word-rate video clock by ten. Often, the PLL used to multiply the clock can also be designed to implement jitter reduction. In Figure 5, the PLL multiplies the high-jitter clock by 10 and also reduces the jitter on the clock. The bit-rate clock is used by the transmitter to serialize the data for transmission. The asynchronous FIFO and the input stages of the SDI transmitter must run at the word rate. The low-jitter bit-rate clock produced by the PLL can be divided by ten to produce a low-jitter word-rate clock. Or, as shown in Figure 5, a counter can generate a clock enable for one clock cycle out of every ten cycles for the FIFO and SDI transmitter.

The Xilinx Coregen tool can produce asynchronous FIFOs well suited for this jitter reduction application. The FIFO produced by Coregen can be implemented in either distributed RAM for shallow FIFOs or in block RAM when deeper FIFOs are required. In a jitter reduction application, the FIFO usually is not very deep. An asynchronous FIFO that is 10-bits wide and 15 locations or less deep is usually sufficient and uses very few FPGA resources when implemented in distributed RAM in Xilinx FPGAs.

A PLL with a narrow input bandwidth is required to implement clock jitter reduction. A DCM or DLL in Xilinx FPGAs cannot be used for jitter reduction. These digital circuits effectively pass any jitter present on the input clock directly through to the output clock.

It is possible to build a PLL in a Xilinx FPGA using an external VCO or VCXO as described in the "Internal CDR" section of this application note. A PLL built using this technique can act as both a clock multiplier and a jitter reducer to meet the needs of the SDI transmitter.

## Clock Multiplication

Every SDI transmitter needs to have both a word-rate clock and a bit-rate clock. The bit-rate clock runs at 10X the word-rate and is used to clock the serializer. It is important that the clock

multiplier be capable of producing the bit-rate clock with low jitter. Jitter on the bit-rate clock becomes jitter on the SDI output bitstream.

As mentioned in the previous section, the jitter reduction PLL can often also be used to implement the clock multiplier for the bit-rate clock.

It is also possible to use the clock synthesis capability of the DCM in Virtex-II Pro to create the bit-rate clock for the SDI transmitter. The Virtex-II Pro DCM synthesizes an output clock that can be as fast as 360 MHz in the slowest speed grade Virtex-II Pro devices. The DCM in Virtex-II, however, should not be used for generating the SDI transmitter bit-rate clock. This older DCM design produces too much jitter in frequency synthesis mode.

When using the Virtex-II Pro DCM in this application, the 27 MHz clock cannot simply be multiplied by 10 to produce a 270 MHz bit-rate clock. For all current speed grades of Virtex-II Pro devices, the DCM must be in high frequency mode to output a 270 MHz clock on the CLKFX output. However, in high frequency mode, the input clock must be at least 50 MHz when using the CLKFX output.

There are two solutions to this problem. First, if a 2X version of the word-rate clock (54 MHz) is available to the FPGA, this can be used as the CLKIN to the DCM with the DCM in high frequency mode. The DCM can multiply this clock by 5 to produce a 270 MHz clock. It is not viable to cascade two DCMs, one multiplying the input clock by 2 and the second by 5. Cascading DCMs will produce too much jitter. Using a 54 MHz reference clock and using the DCM as a 5X multiplier, the typical output jitter was measured on the SDI transmitter of the Xilinx SDV demo board of less than 0.1 UI with a 270 Mb/s SDI bitstream.(1)

If a 2X word-rate clock is not available, then the DDR output capabilities of the Virtex-II Pro FPGA can be used as shown in Figure 6. In this example, the DCM is in low frequency mode and the 27 MHz clock is connected to the CLKIN of the DCM. The DCM multiplies CLKIN by 5X to produce a 135 MHz clock (one-half the bit-rate). The SDI transmitter's serializer is clocked by the 135 MHz clock and shifts two bits every clock cycle. The two output bits from the serializer connect to the data inputs of a FDDRRSE DDR output primitive. The DDR primitive uses two phases of the bit-rate clock to multiplex the two bits together to produce a 270 Mb/s bitstream.

The Load Enable Logic block in Figure 6 generates a load enable to the serializer. The load enable signal is asserted for one clock cycle out of every five cycles of the 135 MHz clock. The load enable to the serializer is asserted during the clock cycle immediately after the rising edge of the 27 MHz clock.

This DDR technique also works for 360 MHz bit rates by multiplying the 36 MHz word-rate clock by 5 to produce a 180 MHz clock. It can also be used for 540 MHz bit rates, but the DCM must be in high frequency mode for the CLKFX output to run at 270 MHz.

Using this DDR technique on the Xilinx SDV Demo Board, we have measured typical SDI transmitter jitter numbers of less than 0.1 UI for both timing and alignment jitter at 270 Mb/s. Keep in mind, however, that the DCM will not do jitter reduction. The word-rate clock supplied to the CLKIN input of the DCM must have little jitter in order to produce a bit-rate clock with low jitter and, subsequently, a low-jitter SDI output bitstream.

In Figure 6, the Q0 out of the serializer is the LSB and must be sent by the transmitter before Q1. This requires connecting Q0 to the D1 input of the FDDRRSE block and Q1 to the D0 input as shown.

1. All SDI transmitter output jitter measurements were conducted using the Xilinx SDV demo board. The FPGA was interfaced to a 75-ohm BNC connector through a National Semiconductor CLC001 cable driver. The reference clock into the DCM clock multiplier was a crystal oscillator with less than 40ps pk-pk jitter. The jitter measurements were made with a Tektronix WFM700A analyzer connected to the output of the SDV demo board with 1 meter of RG179B/U coax cable.

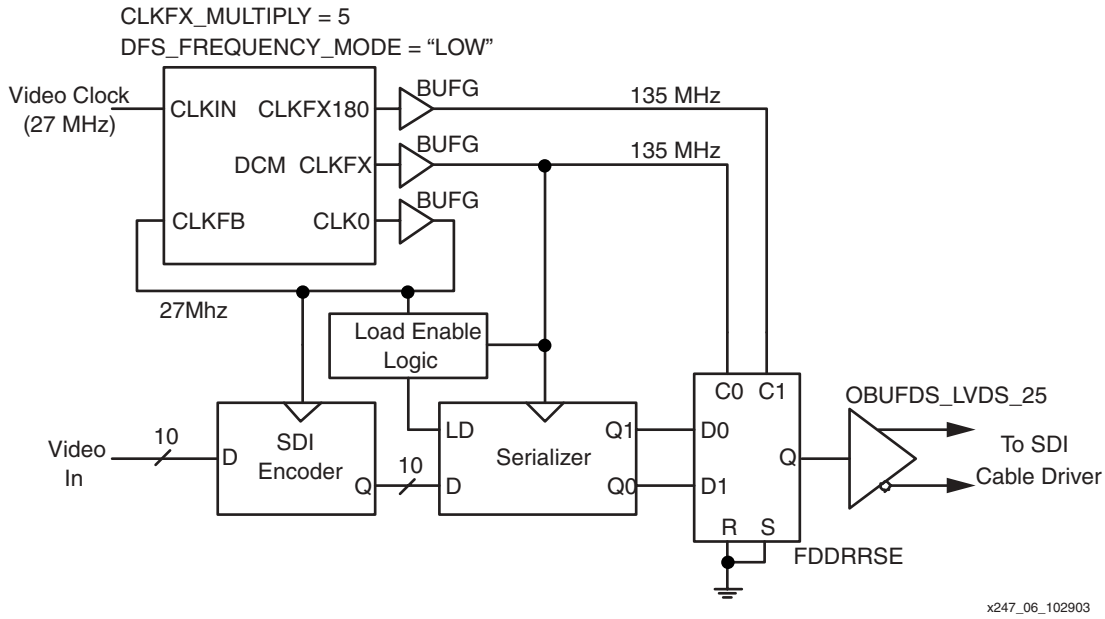



*Figure 6:* **Using DCM and DDR for SDI Serializer**

## SDI Cable Driver

The SMPTE 259M SDI standard requires that the signal produced by an SDI transmitter meet the following electrical specifications:

Unbalanced (single-ended) output with a source impedance of 75-ohms and a return loss(1) of at least 15 dB over a frequency range of 5 MHz to the clock frequency of the signal being transmitted.

- A peak-to-peak amplitude of 800 mV ± 10%.
- DC offset of the mid-amplitude point of the signal of 0.0V ± 0.5V. (This means that the transmitter will be AC coupled to the cable.)
- Rise and fall times (between 20% and 80% of the signal amplitude): 0.4 ns min and 1.5 ns max. Rise and fall times must not differ by more than 0.5 ns.
- Overshoot of the rising and falling edges shall not exceed 10% of the amplitude.

While it might be possible to AC couple the output of a Xilinx FPGA to the SDI coax cable using one of the existing Xilinx FPGA supported I/O standards, meeting all of the SDI electrical requirements is not a simple matter. It is much easier to use an external cable driver chip specifically designed to meet the electrical specifications of the SDI standard. Several vendors make SDI cable drivers.

With signaling rates as high as 360 MHz (or 540 MHz with SMPTE 344M), it is highly recommended that an LVDS output pair be used to drive the SDI bitstream out of the Xilinx FPGA. Using an LVDS output buffer produces the lowest jitter signal out of the FPGA. The National Semiconductor CLC001 is an SDI-compliant cable driver that has LVDS inputs. It is easy to interface the CLC001 to any Xilinx FPGA that supports the LVDS I/O standard. Figure 7 shows an example of using the National CLC001 with a Xilinx FPGA.[4].

1. Return loss is a measurement of the amount of signal absorbed when a wave reflected by the receiver arrives back at the transmitter output. Higher return loss numbers are better since less of the reflected wave is reflected back to the receiver. Excessive reflections on the cable can cause waveform distortion (jitter), interfering with the SDI receiver's ability to correctly receive the signal.

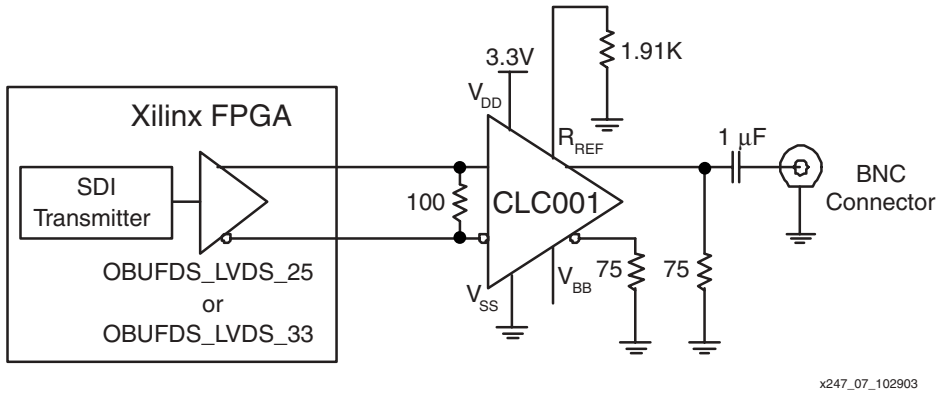



*Figure 7:* **SDI Cable Driver Example**

Note that the AC coupling capacitor used on the output of the CLC001 must be at least 1 µF in order to successfully pass the SDI pathological waveforms. The pathological waveforms are discussed in the "Clock and Data Recovery" section.

By using an SDI compliant cable driver and following the layout and circuit recommendations from the cable driver manufacturer, an SDI transmitter fully compliant with the SDI transmitter electrical specifications can easily be implemented.

## Clock and Data Recovery

The SDI bitstream is encoded using a pseudo-random scrambler followed by a NRZ-to-NRZI conversion. Figure 8 shows how the SDI encoding and decoding algorithms are defined.

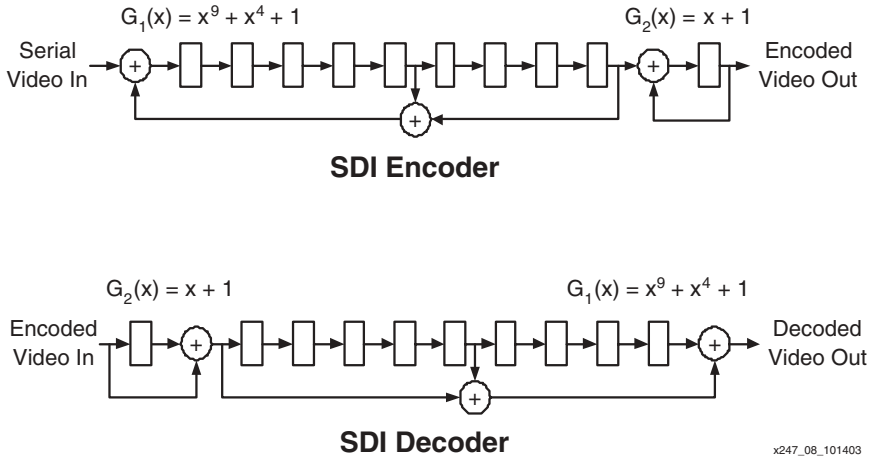



*Figure 8:* **SDI Encoding and Decoding**

The usual way to recover the data from SDI bitstream is to use a PLL-based CDR circuit. The PLL locks to the clock frequency of the incoming bitstream and provides a recovered bit-rate clock running at the clock rate of the bitstream. The data is recovered from the bitstream and is synchronized with the recovered clock.

After the data has been recovered, it is decoded using the inverse of the encoding algorithm. Finally, a "framer" must determine where the 10-bit word boundaries occur in the decoded bitstream. SDI decoder and framer circuits are described in Xilinx application note XAPP288.[5]

Two different CDR techniques for Xilinx FPGA-based SDI receivers are described in the following sections. The "External CDR" section uses an external CDR chip designed specifically for SDI. The "Internal CDR" section is based on Xilinx application note XAPP250 and uses an inexpensive external VCO along with logic in the fabric of a Xilinx FPGA to implement a PLL-based CDR circuit.[6]

In some cases, it might not be necessary to do clock recovery in the SDI receiver. A data recovery technique for SDI is described in the "Data Recovery Only" section.

### External CDR

Several companies produce CDR devices designed specifically for SDI. The National Semiconductor CLC016 is an example of such a device. It supports the four standard bit rates of SMPTE 259M: 143 Mb/s, 177 Mb/s, 270 Mb/s, and 360 Mb/s. Devices such as the CLC016 are often called "reclockers," because they generate a recovered clock and a copy of the original bitstream that has been "reclocked" or synchronized to the recovered clock.

The differential outputs of the CLC016 are compatible with PECL signaling. The common mode voltage of the CLC016's differential outputs is too high to be used directly by low-voltage Xilinx FPGAs. Level shifting is required to reduce the common mode voltage of these outputs to a level compatible with the FPGA. This can be done by AC coupling the outputs of the CLC016 to the FPGA as shown in Figure 9.

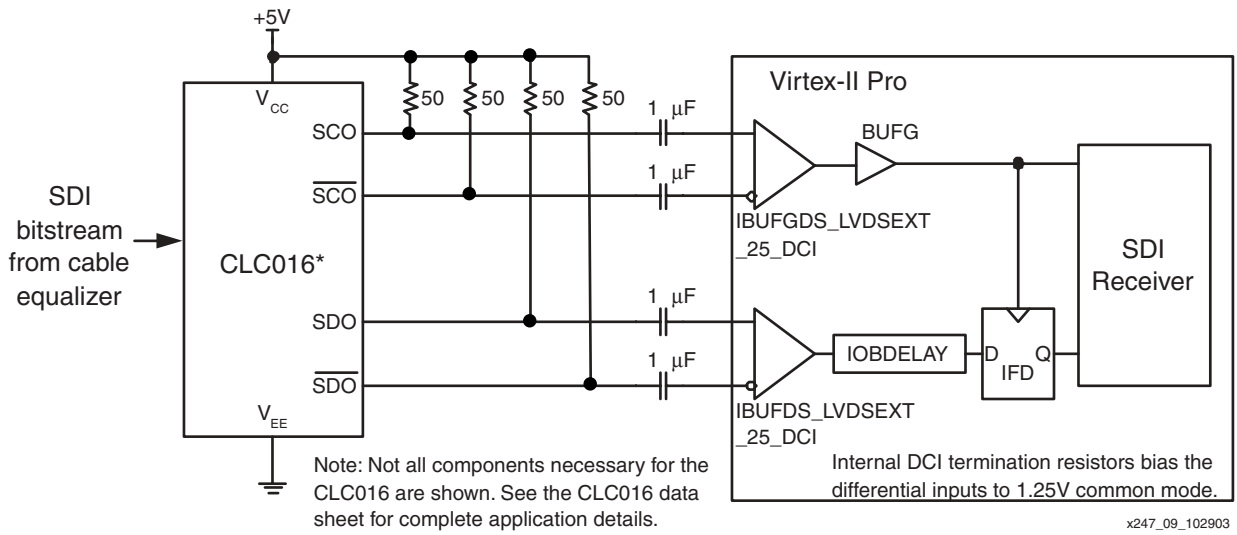



*Figure 9:* **Interfacing an External SDI Reclocker to Xilinx FPGAs**

In Figure 9, a Xilinx Virtex-II Pro FPGA is interfaced to the CLC016. The two differential input pairs into the FPGA use the LVDSEXT_25_DCI standard. This standard supports differential input voltage swings up to 1V, sufficient to handle the 800 mV differential voltage swings of the CLC016. AC coupling is used to remove the DC offset on the outputs of the CLC016. The DCI input termination for the LVDS inputs of the FPGA biases each leg of the inputs to half the VCCO voltage, providing a suitable common mode voltage level on the LVDS inputs. External termination resistors can be used instead of the internal DCI termination, if desired. The AC coupling capacitors must be at least 1 µF in order to pass the pathological waveforms.

The recovered clock from the CLC016 is connected to a BUFG and distributed globally in the FPGA. The serial data comes into the LVDS buffer and passes through the IOBDELAY element in the IOB before being clocked into the input flip-flop (IFD) in the IOB. The IOBDELAY element matches the clock delay through the BUFG and global clock distribution network so that the setup and hold requirements of the IFD flip-flop are satisfied.

The input jitter tolerance of a solution using an external SDI CDR chip is dictated primarily by the specifications of the CDR chip.

### Internal CDR

An external SDI CDR chip usually supports multiple bit rates. However, in many cases, an SDI receiver only needs to support a single bit rate, usually 270 Mb/s. This rate supports 4:2:2 component digital video for both NTSC and PAL. An SDI receiver or transmitter that only

supports a single-bit rate is permitted by the SDI standard. Many pieces of video equipment with SDI interfaces are designed to support only the 270 Mb/s rate.

The main advantage of designing an SDI receiver supporting a single bit rate is cost. It is usually less expensive to implement a CDR circuit that only runs at one bit rate instead of multiple rates.

Xilinx application note XAPP250 describes a technique that can be used to implement a 270 Mb/s CDR circuit using the fabric of a Xilinx FPGA with the addition of a few inexpensive external components. The external components are a voltage controlled oscillator (VCO) and a loop filter to generate the control voltage into the VCO. Using these external components, a complete CDR circuit can be implemented in the fabric of the FPGA.

Figure 10 shows a block diagram of an SDI receiver using XAPP250 for the CDR function. Blocks in the shaded portion of the drawing are internal to the FPGA.

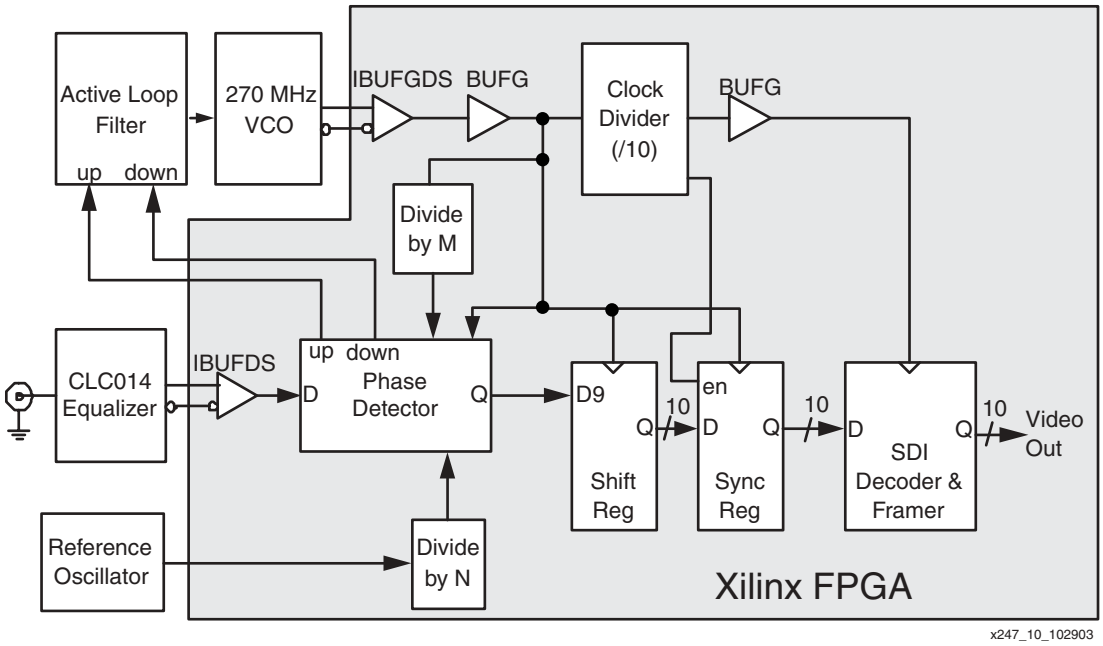



*Figure 10:* **Internal CDR for SDI Receiver**

Initially, the phase detector will control the frequency of the VCO (divided by the M divider) to match the frequency of the reference oscillator divided by the N divider. After locking to the reference clock, the phase detector attempts to match the VCO to the frequency of the video bitstream coming from the cable equalizer. Using a reference clock to “spin-up” the PLL and to lock it when the input bitstream is missing provides several benefits. First, it allows the PLL to lock to the input bitstream faster. Second, it keeps the PLL running at 270 MHz even when the bitstream is missing since the PLL will revert to locking to the reference clock when the bitstream stops.

The clock generated by the VCO is a bit-rate clock and is distributed by the global clock distribution network of the FGPA. A 10-bit shift register, clocked by the bit-rate clock, deserializes the data recovered by the phase detector, generating a 10-bit data word.

The VCO clock is divided by 10 to create a word-rate clock. The clock divider also generates a clock enable to the sync register. This register loads the 10-bit parallel data word from the shift register once every ten cycles of the VCO clock. The output of the sync register is fed into the SDI decoding and framing unit, clocked by the word-rate clock. The sync register is loaded at approximately the same time as the falling edge of the word-rate clock. This provides about half the word-rate clock period for data set up into SDI decoder. The other half of the word-rate clock period provides hold time on the data.

The CDR circuit described in XAPP250 uses a passive loop filter to generate the control voltage for the VCO. The loop filter converts two outputs of the FPGA into an analog control voltage to control the frequency of the VCO. In Figure 10, an active loop filter replaces the passive loop filter described in XAPP250. Active loop filters are generally superior to passive loop filters.

Figure 11 shows the active loop filter and VCO used on the Xilinx SDV demo board to implement the 270 MHz VCO for the SDI CDR circuit. The loop filter acts as an integrator, increasing the output control voltage when FASTER pulses are received from the phase detector in the FPGA and decreasing the control voltage when SLOWER pulses are received. The FASTER and SLOWER inputs to the loop filter are driven by 3.3V outputs from the FPGA.

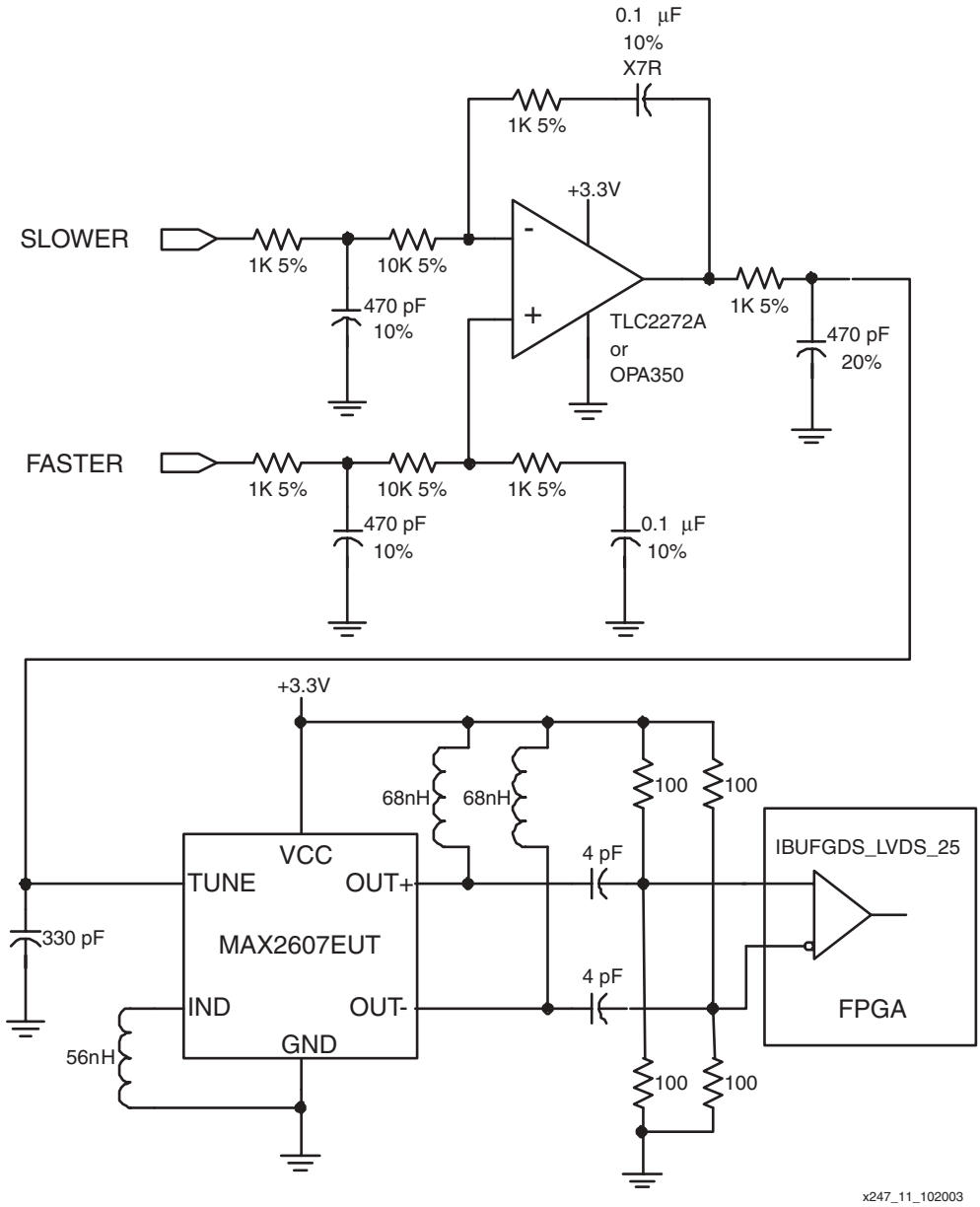



*Figure 11:* **270 MHz VCO for SDI CDR**

The active loop filter in Figure 11 has been optimized to provide good input jitter tolerance characteristics. Figure 12 shows the input jitter tolerance of an SDI receiver using the VCO and loop filter shown in Figure 11.(1) The horizontal axis of the graph is the frequency of the jitter that has been added to the bitstream. The vertical axis is the amplitude of the jitter in UI. To make these measurements, the SDI bitstream is modulated with sinusoidal waveform of a certain frequency. This adds sinusoidal jitter to the bitstream at that particular frequency. The amplitude of the jitter is increased until the SDI receiver fails to accurately receive the signal. This point is recorded on the chart and indicates the input jitter tolerance of the receiver to jitter of that particular frequency. By measuring the input jitter tolerance of the SDI receiver at various jitter frequencies, a graph such as Figure 12 is generated.

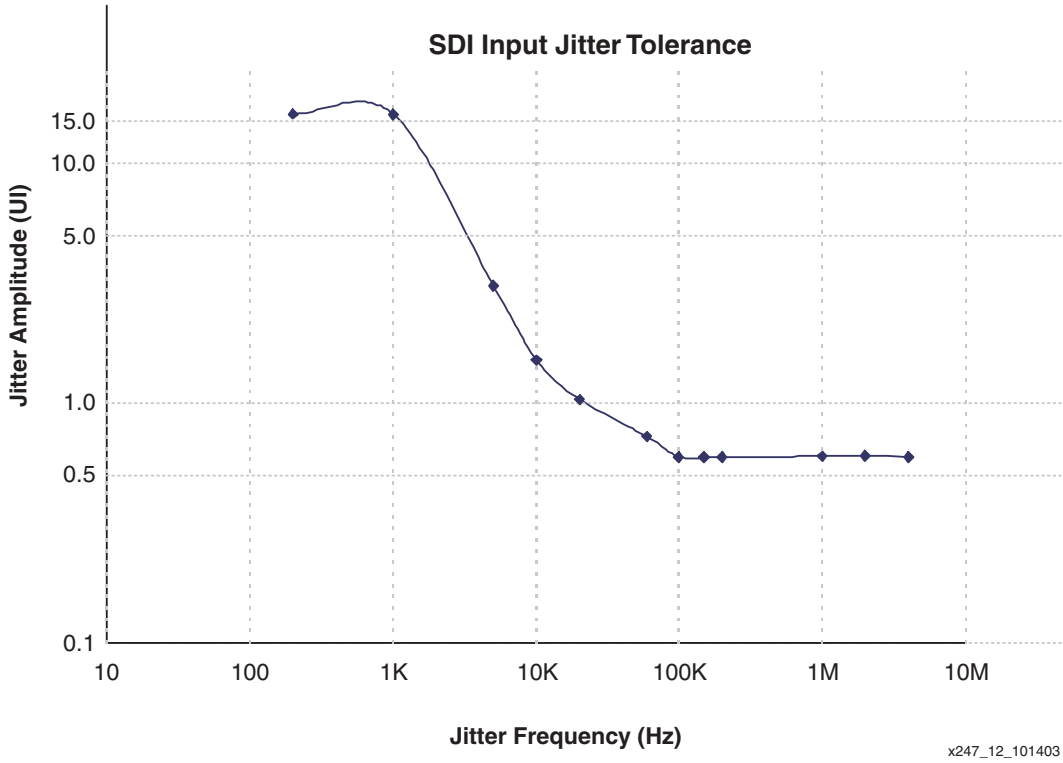



*Figure 12:* **XAPP250 Input Jitter Tolerance**

Figure 12 shows that for jitter frequencies below 1 kHz, the CDR circuit can tolerate more than 15 UI of jitter. The jitter tolerance rolls off above 1 kHz then stabilizes at about 0.65 UI of input jitter tolerance for jitter frequencies above 100 kHz.

### Data Recovery Only

In some cases, it may not be necessary to implement clock recovery for an SDI receiver. If the receiver and transmitter always run at the same frequency, then clock recovery is not necessary and data recovery only techniques can be used. Even if the transmitter and receiver vary slightly in frequency, various synchronization techniques can be used to make up for these minor frequency variations on a periodic basis.

Data recovery only could be used, for example, for an SDI link between a receiver and transmitter located in the same piece of equipment. Perhaps the receiver and transmitter are on different circuit boards in the same chassis. In this case, the transmitter and receiver would

1. The input jitter tolerance graph shown for the XAPP250 style CDR was measured using the Xilinx SDV demo board. The external loop filter and VCO are implemented on this board as shown in Figure 11. The measurements were taken using a PRBS pattern using the equipment described in "Appendix A: Test Equipment.".

probably have access to the same video clock and data recovery techniques would work very well.

Another example occurs in broadcast studios where a studio time base signal, usually called “house sync,” is distributed to all the video equipment in the studio. Each piece of equipment uses this time base signal to generate local video timing signals including the video clock. In this scheme, a transmitter and receiver in different locations in the same broadcast studio will have video clocks that are derived from the common time base and will, over time, be running at exactly the same frequency.

Xilinx has a several application notes that describe different techniques for implementing data recovery units in Xilinx FPGAs. These techniques could be used in SDI receiver applications where clock recovery is not required.

Xilinx application note XAPP224 describes a technique for implementing data recovery in Virtex-II or Virtex-II Pro FPGAs.[7] This technique supports bit rates up to about 420 Mb/s. XAPP224 implements an oversampling technique and does not require that the input bitstream have any particular phase relationship to the local reference clock.

The primary disadvantage of XAPP224 is that it requires two DCMs to create different phases of the reference clock for oversampling the bitstream. However, the clocks generated by the DCMs can support multiple SDI receivers in the same FPGA. Asynchronous data recovery techniques like the one described in Xilinx application note XAPP671 can be used to implement data recovery for SDI without requiring any DCMs.[8]

Both of these data recovery techniques allow for minor variations in the clock between the transmitter and the receiver. If it is known that, on average, the transmit and receive clocks are exactly the same frequency, a shallow FIFO is sufficient to smooth out the minor variations in the rate at which data is recovered. The data recovery unit will normally produce 10 bits of data every word-rate clock cycle. However, occasionally the data recovery unit recovers either 9 or 11 bits in one word-rate clock cycle. By buffering the recovered data bits with a shallow FIFO, these minor variations are averaged out.

If latency in the SDI receiver is not an issue for a particular application, then it is possible to use a data recovery technique combined with a VCO to regenerate the video clock, in effect implementing clock and data recovery without a PLL. This technique is shown in Figure 13.

The data from the data recovery unit is written to an asynchronous FIFO. The block RAMs in Xilinx FPGAs are well suited for implementing asynchronous FIFOs. Initially, a controller allows the FIFO to become half full before any data words are read from the FIFO. After starting to read the FIFO, the controller monitors the data level in the FIFO and attempts to keep the FIFO half full by speeding up or slowing down the VCO that is generating the word-rate clock used to read the FIFO.

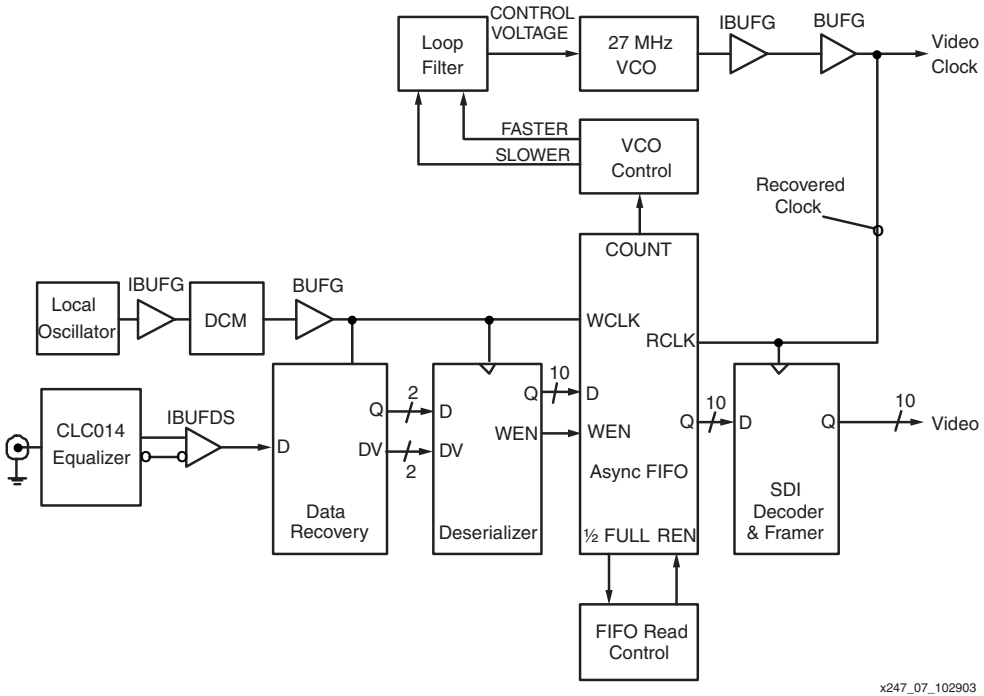
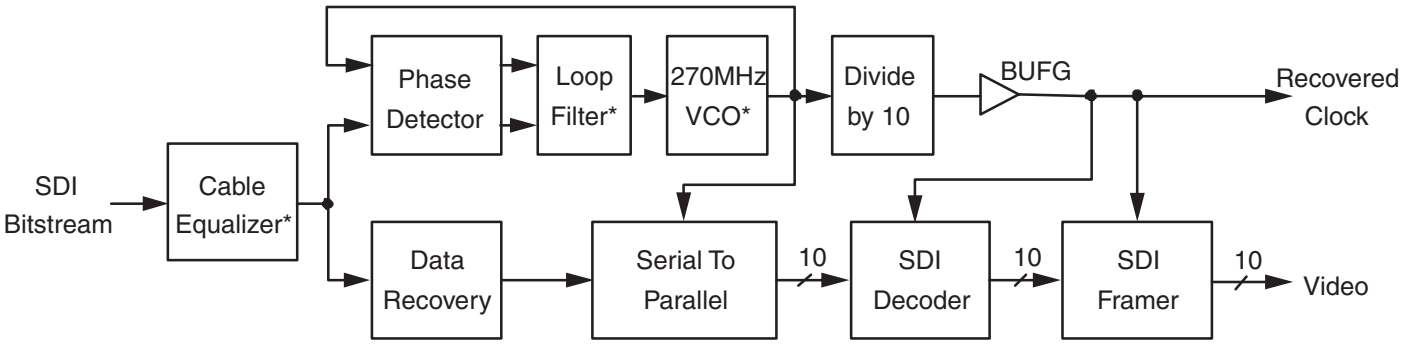



*Figure 13:* **XAPP224 and VCO Implement CDR**

## Reference Designs

Three reference designs are provided with this application note: two SDI receivers and one SDI transmitter. One SDI receiver is based on XAPP250. The other receiver is intended for use with an external SDI reclocker.

Figure 14 is a block diagram of the XAPP250-based SDI receiver reference design. The CDR unit uses a clock recovery PLL built in the fabric of the FPGA. An external VCO and loop filter must be provided. Details of the VCO and loop filter are shown in Figure 11.



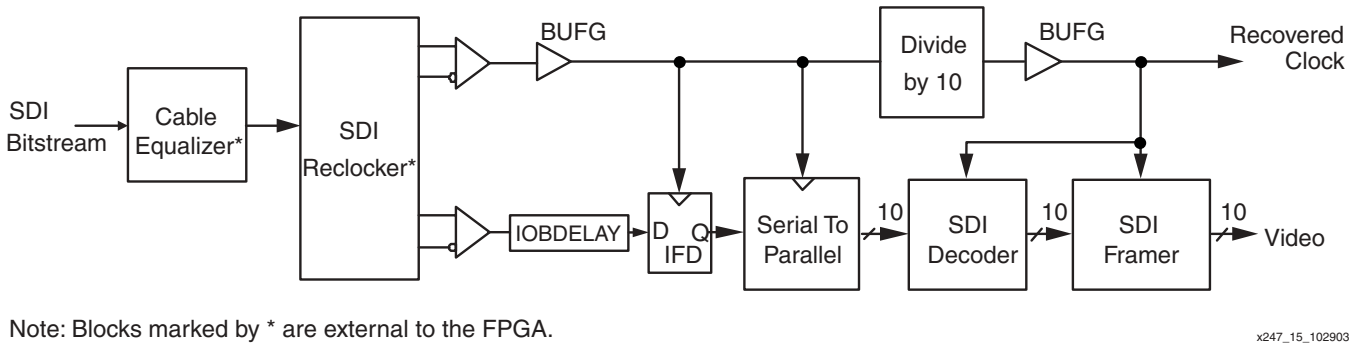

Blocks marked by * are external to the FPGA.

*Figure 14:* **XAPP250-Based SDI Receiver Reference Design**

The second SDI receiver reference design uses an external reclocker for CDR. See Figure 15. The reclocker provides a recovered clock on a differential signal pair, and it provides the serial bitstream resynchronized to the recovered clock on another differential signal pair.

In the FPGA, the recovered clock is distributed through the global clock distribution network. The serial data is delayed by an IOBDELAY element. The IOBDELAY matches the delay added to the clock by the BUFG and clock distribution network. After the IOBDELAY, the serial data is clocked into the input flip-flip in the IOB. The serial bitstream from the flip-flop is deserialized

and then decoded and framed.The recovered clock is divided by 10 by a ring counter to provide a word-rate recovered clock.




Note: Blocks marked by * are external to the FPGA.

*Figure 15:* **SDI Receiver Reference Design for External SDI Reclocker**

The SDI transmitter reference design uses a serializer implemented in the fabric of the FPGA and is identical to the SDI transmitter design shown in Figure 6. A DCM takes the 27 MHz word-rate clock and produces two phases of a 135 MHz clock that are 180 degrees out of phase from each other. A DDR primitive on the output of the serializer creates the 270 MHz SDI bitstream.

Table 1 shows the logic resources required to implement the reference designs. The SMPTE descrambler module and the framer module were taken directly from the XAPP288 reference designs. In addition to the logic shown, both designs required one MULT18X18 primitive used by the barrel shifter in the par_framer_mult module. A framer design without a multiplier is also available in XAPP288 and is slightly larger (see par_framer results).

The reference designs were targeted at Virtex-II Pro XC2VP4-5 and Virtex-II XC2V250-4 devices. In all cases, the synthesis was done using XST with the ISE 5.2i tools. Both devices met all timing constraints for operation at SDI bit rates up to 360 Mb/s.

The reference designs presented here are basic SDI transmitters and receivers and do not include more advanced features, such as EDH insertion and error detection, video standard detection, and flywheel video decoder. These advanced features can be added to the basic designs. The implementation sizes of these features can be found in application notes, XAPP299 and XAPP625.

The reference designs can be downloaded here.

*Table 1:* **Reference Design Implementation Results**

| Logic Resource | Size LUTs | Size FFs | Other |
|---|---|---|---|
| SDI Rx with external CDR (par_framer) | 132 | 103 | |
| SDI Rx with external CDR (par_framer_mult) | 78 | 109 | 1 MULT18X18 |
| SDI Rx with XAPP250 CDR (par_framer) | 352 | 256 | |
| SDI Rx with XAPP250 CDR (par_framer_mult) | 270 | 262 | 1 MULT18X18 |
| SDI Tx | 31 | 43 | 1 DCM |

## Conclusions

The SDI application notes from Xilinx describe how to implement a complete SDI interface using Xilinx FPGAs. This application note focused on the details of implementing the physical layer of the SDI interface.

Using Xilinx FPGAs to implement SDI interfaces in video equipment can provide many benefits. Xilinx FPGAs are large enough to allow multiple SDI interfaces to be implemented in the same FPGA device, reducing the number of devices required. Even in video equipment with only a single SDI interface, the FPGA can provide higher levels of integration by providing additional processing functions for the video prior to transmission over the SDI interface or after reception.

## 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. EBU (European Broadcasting Union) Technical Document tech 3283 1996: Measurement in Digital Component Television Studios, available at: http://www.ebu.ch.
4. National Semiconductor: CLC001 Data Sheet.
5. Xilinx application note, *XAPP288: Serial Digital Interface (SDI) Video Encoder and Test Generator* by John F. Snow.
6. Xilinx application note, *XAPP250: Clock and Data Recovery with Coded Data Streams* by Leonard Dieguez.
7. Xilinx application note, *XAPP224: Data Recovery* by Nick Sawyer.
8. Xilinx application note, *XAPP671: High Speed Data Recovery using Asynchronous Data Capture Techniques* by Catalin Baetoniu and Tze Yi Yeoh.

# Appendix A: Test Equipment

## Test Equipment Used for Receiver Input Jitter Tolerance Measurements

All input jitter tolerance tests were made using the SDI receiver on the Xilinx SDV demo board (CTXIL103 from Cook Technologies).

Various test equipment was required to generate and measure the input jitter tolerance, including:

HP71603B Bit-Error Rate Tester for generating and verifying high-speed serial data. Includes the following modules:

- HP70311A Clock Source
- HP70841B Pattern Generator
- HP70842B Error Detector

Two HP8648D Signal generators to provide an accurate, low-jitter reference clock:

- Agilent 71501C jitter analysis system to add periodic jitter to the serial stream for receiver jitter tolerance testing
- Agilent 86100A Digital Communications Analyzer for measuring jitter, signal amplitude, eye width, and other signal characteristics.

## Test Equipment Used for Transmitter Output Jitter Measurements

All transmitter output jitter measurements were made using the SDI transmitter on the Xilinx SDV demo board.

The transmitter output jitter was measured using a Tektronix WFM700A Waveform Monitor.

# Revision History

The following table shows the revision history for this document.

| Date | Version | Revision |
|---|---|---|
| 12/02/03 | 1.0 | Initial Xilinx release. |