XAPP1092 (v1.0) July 8, 2013

Implementing SMPTE SDI Interfaces with Zynq-7000 AP SoC GTX Transceivers

Author: John Snow

Summary

The Society of Motion Picture and Television Engineers (SMPTE) serial digital interface (SDI) family of standards is widely used in professional broadcast video equipment. These interfaces are used in broadcast studios and video production centers to carry uncompressed digital video, along with embedded ancillary data such as multiple audio channels.

The Xilinx® SMPTE SD/HD/3G-SDI LogiCORE™ IP is a generic SDI receive/transmit datapath that does not have any device-specific control functions. This application note provides a module containing control logic to couple the SMPTE SDI LogiCORE IP with the Zynq®-7000 All Programmable SoC GTX transceivers to form a complete SDI interface. This application note also provides several example SD/HD/3G-SDI designs that run on the Xilinx Zynq-7000 AP SoC ZC706 evaluation board.

Terms used in this document are explained in the Glossary, page 53. Titles of SMPTE reports and standards are listed in References, page 56, and referred to by SMPTE document number in text.

Introduction

The Xilinx SMPTE SD/HD/3G-SDI LogiCORE IP (called the *SDI core* in the rest of this document) can be connected to a Zynq-7000 SoC GTX transceiver to implement an SDI interface capable of supporting the SMPTE SD-SDI, HD-SDI, and 3G-SDI standards. The SDI core and GTX transceiver must be supplemented with some additional logic to connect them together to implement a fully functional SDI interface. This application note describes this additional control and interface logic and provides the necessary control and interface modules in both Verilog and VHDL source code.

The primary functions of the device-specific control logic are:

- Reset logic for the GTX transceiver
- Dynamic switching of the GTX RX and TX serial clock dividers to support the three SDI standards
- Dynamic TX reference clock switching to support the two different bit rates in each of the HD-SDI and 3G-SDI standards:
  - HD-SDI mode: 1.485 Gb/s and 1.485/1.001 Gb/s
  - 3G-SDI mode: 2.97 Gb/s and 2.97/1.001 Gb/s
- Data recovery unit for recovering data in SD-SDI mode
- RX bit rate detection used to determine if the RX is receiving a 1/1 or 1/1.001 bit rate signal

Also supplied with this application note is a wrapper file that contains an instance of the control module for the GTX transceiver and an instance of the SMPTE SDI core with the necessary connections between them. This file simplifies the process of creating an SDI interface.

The following terms are used in this document:

- The *SDI core* refers to the SMPTE SD/HD/3G-SDI core generated by the CORE Generator™ tool or the Vivado™ IP catalog.

- The *control module* is a module that implements the various device-specific functions required when using the GTX transceiver to implement an SDI interface using the SMPTE SDI core. The control module is supplied in source code form with this application note.
- The *SDI wrapper* is a wrapper module that instances and interconnects the SMPTE SDI core and the control module. The SDI wrapper is supplied in source code form with this application note. Figure 1 is a simplified block diagram of how the various pieces fit together to form an SDI interface.
- The *GTX wrapper* is a wrapper file for GTX transceivers generated by the 7 Series FPGAs Transceivers Wizard that is available in the CORE Generator and IP Catalog tools.

The SDI wrapper includes one instance of a control module and one instance of an SDI core. The SDI core includes both an SDI RX and an SDI TX datapath. The wrapper module is usually connected to the GTX RX and TX units in the same GTX transceiver, but this does not have to be the case. The RX and TX units of different GTX transceivers can be connected to the same SDI wrapper. If only an SDI RX or only an SDI TX is required, the unused portions of the control module and the SDI core are optimized away during synthesis.

This application note includes two example demonstration applications using the SDI core. These applications run on the ZC706 evaluation board. An inrevium SDI FPGA mezzanine card (FMC) is also required to provide the SDI physical interfaces.

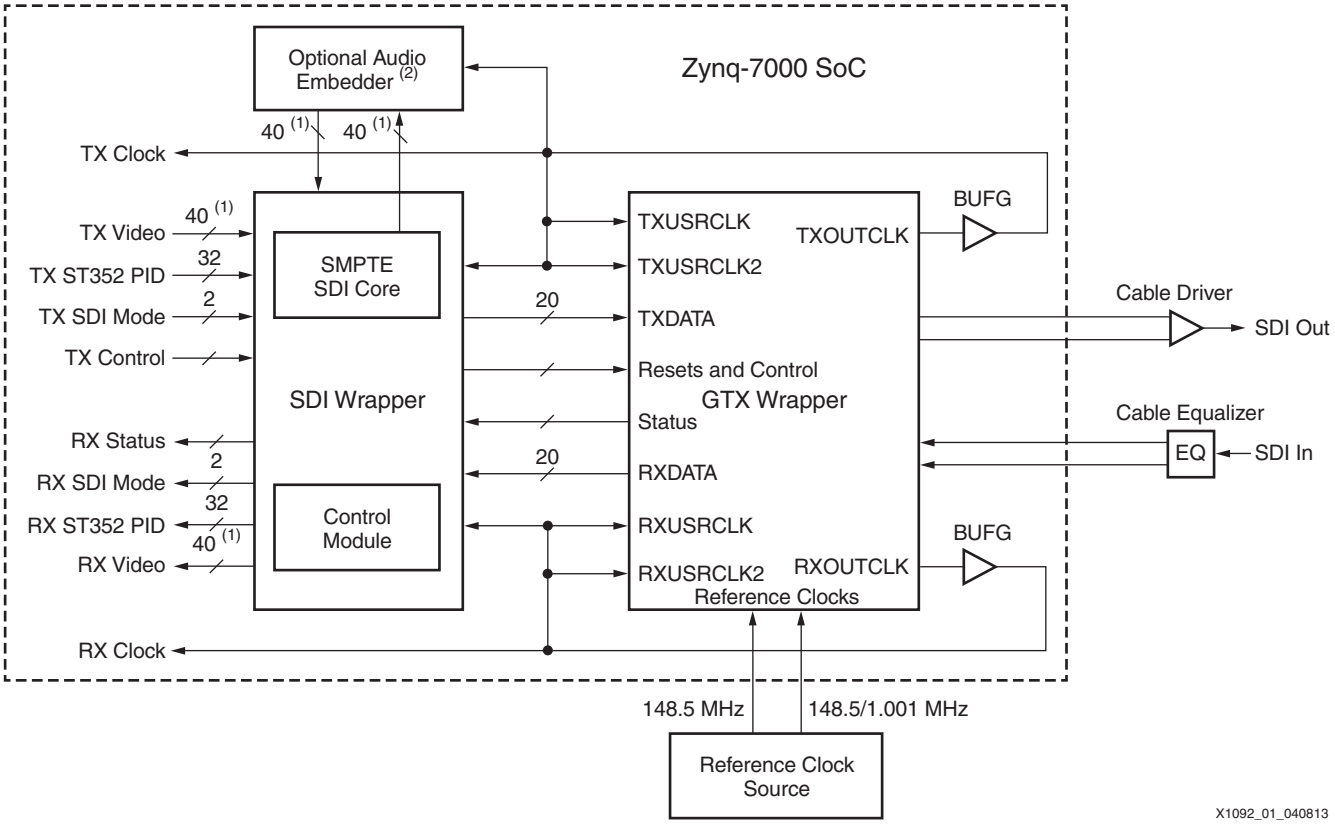


Figure 1: **Block Diagram of Complete SDI RX/TX Interface**

Notes relevant to Figure 1:

1. These 40-bit buses are actually four buses, each of which is 10 bits wide, and each carrying a different SDI data stream. The number of active data streams, and therefore buses, varies depending on the SDI mode. For example, in SD-SDI mode, only one 10-bit data stream is active and in HD-SDI mode, two 10-bit data streams are active.

2. The optional audio embedder is a separate core and is not included with the SMPTE SDI core or with this application note.

Using Zynq-7000 SoC GTX Transceivers for SDI Interfaces

The information in this section is intended to supplement, not replace, the information in *7 Series FPGAs GTX/GTH Transceivers User Guide* (UG476). This information highlights features of the GTX transceivers that are of particular importance for SDI applications.

In this document, the naming convention used in the *7 Series FPGAs GTX/GTH Transceivers User Guide* for the GTX transceiver ports is followed. This convention is to use only the base name of a port. When the 7 Series FPGAs Transceivers Wizard is used to create a GTX wrapper, the port names on the wrapper add a prefix to the base port name with *GT0_* for the first transceiver in the wrapper, *GT1_* for the second transceiver in the wrapper, and so on. Additionally, all input ports have a suffix of *_IN* and all outputs have a suffix of *_OUT*. For example, when a port named TXRATE is discussed in this document, the actual name of that port in the GTX wrapper would be GT0_TXRATE_IN for the TXRATE port of the first GTX wrapper, GT1_TXRATE_IN for the second GTX wrapper, and so on.

There are a variety of clocks in applications using GTX transceivers. The SDI protocol, which does not allow for clock correction by stuffing and removing extra data in the data stream, requires careful attention to how these clocks are generated and used in the application. GTX transceivers require reference clocks to operate. The reference clocks are used by phase-locked loops (PLLs) in the GTX Quad to generate serial clocks for the receiver and transmitter sections of each transceiver. As described in more detail in GTX Reference Clocks, page 4, the serial bit rate of the GTX transmitter is an exact multiple of the reference clock frequency it is using. Furthermore, the data rate of the video provided to the input of the SDI transmitter datapath must also exactly match (or be a specific multiple of) the frequency of the reference clock used by the GTX transmitter. Consequently, the designer must determine how to generate the transmitter reference clock so that it is frequency-locked exactly with the data rate of the video stream to be transmitted.

The GTX transmitter outputs a clock on its TXOUTCLK at a frequency that is exactly equal to the word rate of the data that must enter the TXDATA port of the GTX transmitter. The TXOUTCLK is generated in the GTX transmitter by dividing the serial clock from the PLL down to the word rate. In most applications, the TXOUTCLK from the GTX transmitter is buffered by a global or regional clock buffer and then used to clock the SDI transmitter datapath and the TXUSRCLK and TXUSRCLK2 clock inputs of the GTX transmitter. It is possible to use a different clock to clock the SDI transmitter datapath and the TXUSRCLK and TXUSRCLK2 ports of the GTX transmitter. A shallow TX buffer in the GTX transmitter does allow for phase differences between the data entering the TXDATA port and the internal clocking of the GTX transmitter. However, any frequency difference between the incoming data and the internal clock frequency of the GTX transmitter (as represented by TXOUTCLK) quickly causes the TX buffer to under- or overflow resulting in errors in the serial bit stream generated by the GTX transmitter. Consequently, the data rate of the stream entering the TXDATA port of the GTX transmitter (represented by the frequency of the TXUSRCLK and TXUSRCLK2 clocks) and the internal data rate of the GTX transmitter (as set by the transmitter reference clock and represented by the frequency of TXOUTCLK) must match exactly.

The GTX receiver reference clock, however, does not need an exact relationship with the bit rate of the incoming SDI signal. This is because the clock and data recovery (CDR) unit in the GTX receiver has a fairly wide range of bit rates that it can receive. That range is centered on the frequency of the reference clock, but there is not an exact frequency relationship required as there is with the transmitter. Not having an exact relationship allows the receiver reference clock to be generated by a local oscillator that has no exact frequency relationship to the incoming SDI signal. The GTX receiver generates a recovered clock that is frequency-locked to the incoming SDI bit rate. This clock is output on the RXOUTCLK port of the GTX transceiver. As is described in more detail later, RXOUTCLK is a true recovered clock when receiving HD-SDI and 3G-SDI signals, but not when receiving SD-SDI signals. Typically, RXOUTCLK is

buffered by a global or regional clock buffer and then applied to the RXUSRCLK and RXUSRCLK2 ports of the GTX receiver and used as the clock for the SDI receiver datapath.

One additional clock is required for SDI applications. This is a free-running, fixed-frequency clock that is used as the clock for the dynamic reconfiguration port (DRP) of the GTX transceiver. This same clock is also usually supplied to the control logic in the SDI wrapper where it is used for timing purposes. This clock can be any frequency up to the maximum GTX transceiver DRP clock frequency for the device being used. Xilinx recommends that the frequency of this clock be at least 10 MHz. The frequency of this clock does not require a relationship with any of the other clocks or data rates of the SDI application. This clock must not change frequencies when the SDI mode changes. It must remain exactly the same frequency at all times. It also must never stop. This clock can be used for all SDI interfaces in the device.

GTX Reference Clocks

Zynq-7000 SoC GTX transceivers are grouped into Quads. Each Quad contains four GTXE2_CHANNEL transceiver primitives and one GTXE2_COMMON primitive containing a Quad PLL (QPLL) as shown in Figure 2. The clock generated by the QPLL is distributed to all four transceivers in the Quad. Each GTXE2_CHANNEL has its own PLL called the channel PLL (CPLL), which can provide a clock to the RX and TX of that transceiver only. Each RX and TX unit in the Quad can be individually configured to use either the QPLL or the CPLL as its clock source. Furthermore, any RX or TX unit can dynamically switch its clock source between the QPLL and the CPLL. This configuration and the dynamic switching capability are particularly useful for SDI applications.

Typical SDI applications require GTX transceivers to support five different bit rates:

- 270 Mb/s for SD-SDI
- 1.485 Gb/s for HD-SDI
- 1.485/1.001 Gb/s (~1.4835 Gb/s) for HD-SDI
- 2.97 Gb/s for 3G-SDI
- 2.97/1.001 Gb/s (~2.967 Gb/s) for 3G-SDI

The CDR unit in the RX section of the GTX transceiver can support receiving bit rates that are up to ±1250 ppm from the reference frequency. This makes it possible to receive all five of the SDI bit rates using a single reference clock frequency.

The TX section of the GTX transceiver, however, requires two different reference frequencies to support all five SDI bit rates. This is because the transmitters, in general, can only transmit at an exact integer multiple of the supplied reference clock frequency[1]. Therefore, most SDI applications provide two separate reference clocks to the GTX Quad. One of those clocks is used as the RX reference clock, and both of them are used as TX reference clocks. Usually, the supplied reference frequency pair are 148.5 MHz and 148.5/1.001 MHz or 74.25 MHz and 74.25/1.001 MHz.

The source of the GTX reference clocks is very application specific. The receiver reference clock can be a local oscillator because it does not need to match the incoming SDI bit rate exactly. However, because the GTX transmitter always runs at an exact multiple of the reference clock frequency, the frequency of the transmitter reference clock must be exactly related to the data rate of the transmitted data (unless the PICXO technique is used). Most often, the transmitter reference clocks are generated by genlock PLLs, thereby deriving the GTX transmitter bit rate from the studio video reference signal. In some cases, such as the SDI

1. Using a technique called *phase interpolator controlled oscillator* (PICXO), the bit rate of a GTX TX can be “pulled” by plus or minus a few hundred ppm from an exact integer multiple of the reference clock frequency. However, the pull range of the GTX TX using the PICXO technique is not sufficient to span both HD-SDI bit rates or both 3G-SDI bit rates using a single reference clock frequency.

pass-through demonstration included with this application note, the transmitter bit rate is derived from the recovered clock of a GTX receiver that is receiving an SDI signal. In such cases, an external PLL is required to reduce the jitter on the recovered clock before using it as the transmitter reference clock.

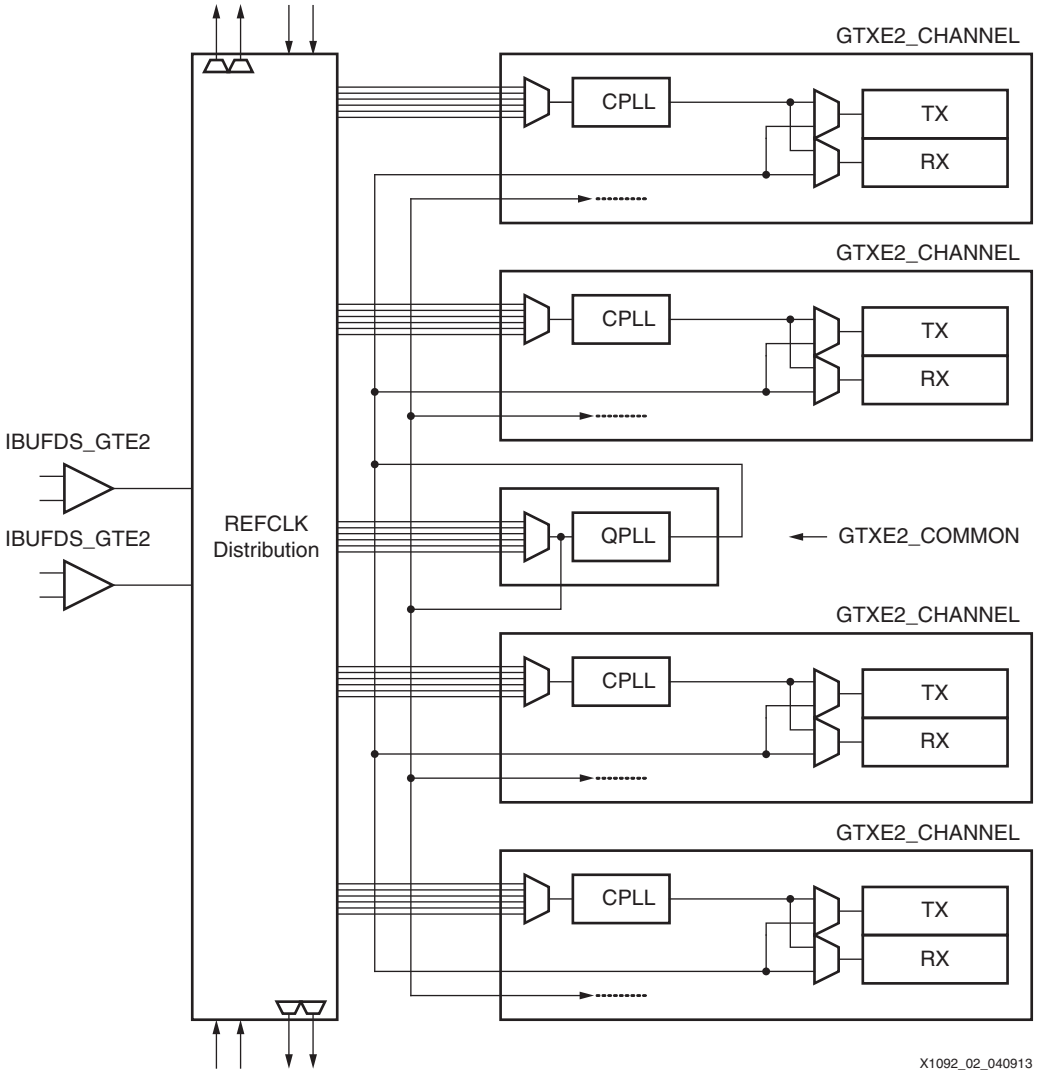


Figure 2: **Zynq-7000 SoC GTX Transceiver Quad Configuration**

In a typical SDI application, one of these reference clocks is connected to the QPLL and the other is connected to all the CPLLs in the Quad. It does not matter which one is used for the QPLL reference clock and which is used for the CPLL reference clock. The RX units of each transceiver in the Quad are configured to always use the clock from the QPLL. The TX units can dynamically switch between the QPLL clock and the local CPLL clock, depending on the bit rate that is required at the moment. The GTX TXSYSCLKSEL port is used to select the TX unit's clock source between the QPLL and the CPLL. This common configuration for SDI applications is shown in Figure 3. In this figure, MUXes that are not used dynamically in the implementation have been replaced with wires and the reference clock routing between Quads is not shown.

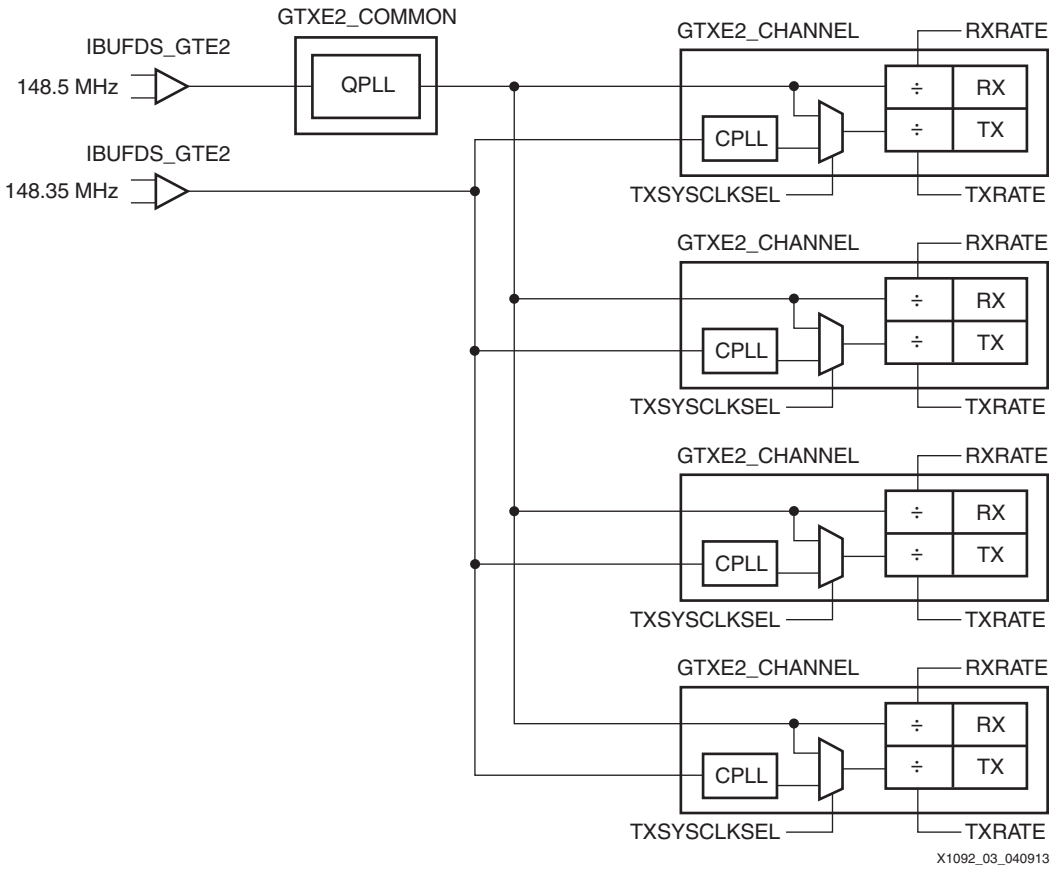


Figure 3: **Typical GTX Reference Clock Implementation for SDI**

Additionally, each GTX RX and TX unit has a serial clock divider that divides the selected clock (QPLL or CPLL) by several selectable integer powers of two. This allows, for example, all of the RX units in the Quad to use the same clock frequency from the QPLL but operate at different line rates by using different serial clock divider values. This is very useful for SDI interfaces because the 3G-SDI bit rates are exactly twice as fast the HD-SDI bit rates. For 270 Mb/s SD-SDI, the GTX transceiver runs at the 3G-SDI line rate using 11X oversampling techniques. Thus, by using two divisors that differ by a value of two locally in each RX unit, reception of all the SDI bit rates is supported by a single RX clock frequency from the QPLL. The ability of the TX units to also locally divide the clock source by two divisors that differ by a factor of two is also important, allowing transmission of all SDI bit rates using just two reference clock frequencies. The serial clock divider value of each RX and TX unit can be changed dynamically using the RXRATE and TXRATE ports of each GTX transceiver.

The configuration shown in Figure 3 is an optimal solution for most SDI applications for several reasons:

- The receivers can receive all SDI bit rates from one fixed reference clock frequency, and the QPLL provides that reference clock to all receivers in the Quad.
- The transmitters have the flexibility to dynamically switch between the QPLL and the CPLL to get both reference clocks they need to transmit all supported SDI bit rates.
- All four receivers and all four transmitters in the Quad are fully independent. They can each be running at different SDI bit rates and can dynamically switch between bit rates without disrupting the other RX or TX units.
- For genlocked applications, modern genlock PLLs usually can simultaneously provide both required reference clock frequencies from the synchronization reference input signal.

The flexibility of the reference clock routing structure in the GTX Quad does allow other PLL clocking configurations. For example, as shown in Figure 4, it would also be possible to provide both reference clocks to the clock selection MUX at the input of each CPLL. The CPLL could then dynamically switch its clock source between the two reference clock frequencies rather than switching TX between the CPLL and the QPLL. This configuration, however, has the disadvantage of requiring a reset of the CPLL and the resulting re-lock time of the CPLL each time the CPLL's reference clock frequency is switched. Thus, the configuration shown in Figure 3 has a faster switching time between TX bit rates and is the configuration supported by the control module supplied with this application note. While the configuration shown in Figure 4 is not directly supported by the supplied control module, it is an allowed and viable configuration and can be implemented if desired.

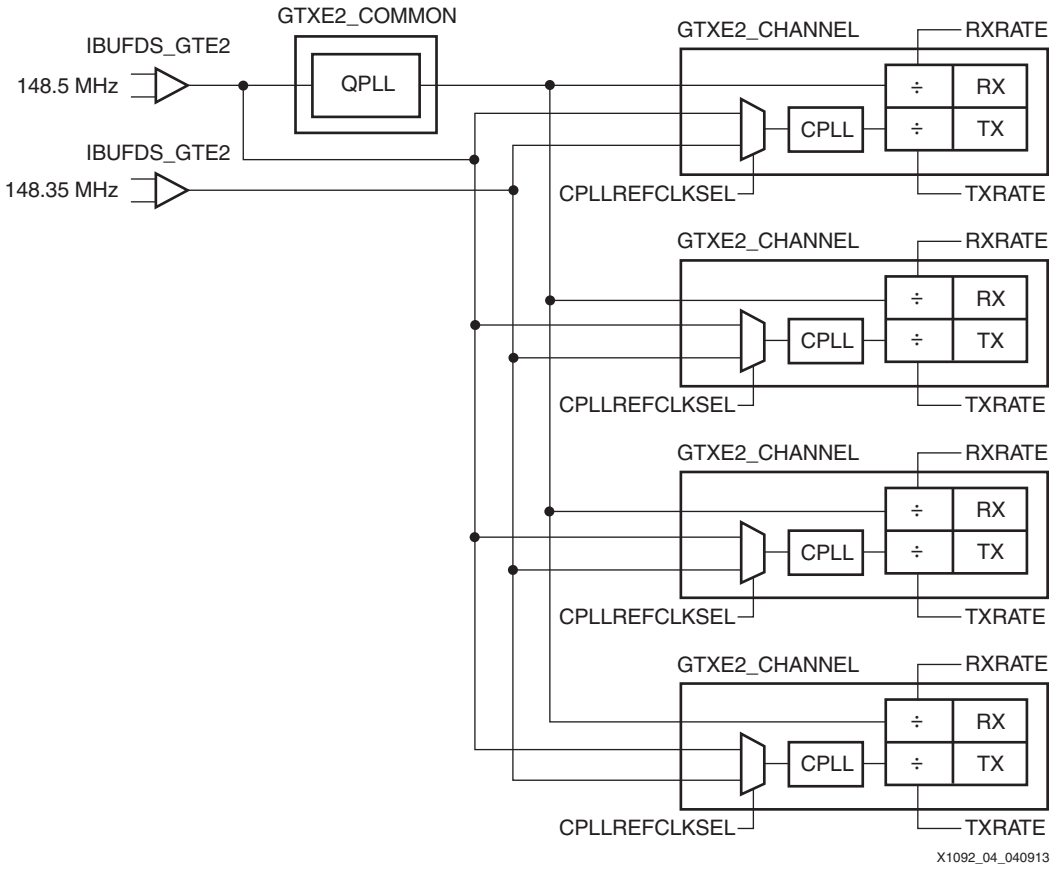


Figure 4: **Alternative Reference Clock Implementation for SDI**

In some applications, it might be necessary for SDI transmitters in the same Quad to be running at slightly different bit rates even though they are transmitting at the same nominal bit rate. This is often the case with SDI routers where the bit rate of each TX must exactly match the bit rate of the SDI signal received by the SDI RX to which the TX is currently connected. In these cases, two transmitters that are transmitting at the same nominal bit rate, in fact, have bit rates that differ by a few ppm. Supporting such applications is possible with the Zynq-7000 SoC GTX Quad architecture because each TX unit has exclusive use of its own CPLL. But to accomplish this, each CPLL must be provided with its own individual reference clock frequency, and the number of GTX reference clock inputs are limited. There are two reference clock inputs per Quad. A Quad can use reference clocks from the Quads above and below. Thus, it is possible to provide some GTX Quads in the device with five different reference clock frequencies (one for the RX and four for the four TX units), but overall, there are not enough reference clock inputs to allow every GTX TX in the device to have its own reference clock. The PICXO technique can be very useful in these cases because it allows a GTX TX to be pulled by a few

hundred ppm away from the frequency of its reference clock. Thus, applications where the bit rate of each SDI TX needs to be individually locked to the bit rate of a received SDI signal can be implemented by using common reference clocks as in Figure 3 and then using the PICXO technique with each GTX TX to set the exact bit rate of each SDI TX individually. This application note does not cover the PICXO technique. For further information about using PICXO, contact Xilinx technical support.

Resets

Refer to *7 Series FPGAs GTX/GTH Transceivers User Guide* (UG476) for complete information about the various resets available on the GTX transceiver. The GTX transceiver resets can be divided into two basic categories: PLL resets and transceiver resets.

PLL Resets

The QPLL and the active CPLLs in each Quad each have their own reset input ports on the GTX wrapper. The control logic supplied by this application note does not generate PLL resets because these resets are very application specific. Each PLL must be reset after FPGA configuration after the reference clock source to the PLL is stable. Interruptions or changes in the reference clock to a PLL also require the PLL to be reset after the clock is once again stable. It is recommended that these PLLs not be reset until at least 500 ns after configuration of the programmable logic (PL) portion of the SoC.

Transceiver Resets

The GTX transceiver has two reset modes called sequential mode and single mode. Sequential mode is the easiest to use and is the mode supported by the control module supplied with this application note. In sequential mode, when any reset input is asserted, the reset state machine in the transceiver automatically asserts all other required transceiver resets in sequential order.

The control module generates three reset signals for the GTX transceiver: GTTXRESET, GTRXRESET, and TXPMARESET. GTTXRESET resets all portions of the transceiver's TX unit. Likewise, GTRXRESET resets all portions of the RX unit. A complete reset of the TX and RX units using these two resets must be done once after configuration. These resets must be initiated after the PLLs have been reset and are locked to their reference clocks. Separate but identical state machines in the control module monitor the PLL reset and PLL locked signals for the RX and the TX. These state machines assert the GTTXRESET or GTRXRESET signals when they detect that the PLL that supplies the clock to the TX or RX unit has been reset. The state machine then waits for the PLL to assert its locked signal while observing a minimum assertion period for the reset signal. This minimum assertion period is controlled by the PLLLOCKDLY parameter/generic. After the PLL indicates it is locked, the GTTXRESET or GTRXRESET is negated. The recommended minimum period of these resets is one cycle of the reference clock supplied to that unit. By default, these resets are asserted for 16 cycles of the fixed frequency clock applied to the clk port of the SDI wrapper. The PLLLOCKDLY parameter/generic specifies the width of a binary counter. Thus, the minimum assertion period of the these resets is given by Equation 1.

$$\text{Minimum GT[R/T]XRESET period} = \frac{2^{PLLLOCKDLY}}{\text{clk frequency}} \qquad \textit{Equation 1}$$

The GTX TXPMARESET port resets the PMA portion of the TX unit. In SDI applications, when the TXSYSCLKSEL port is changed dynamically to switch the TX unit's clock between the CPLL and the QPLL, TXPMARESET must be asserted during the time that TXSYSCLKSEL changes. TXPMARESET must be asserted before changing TXSYSCLKSEL and must remain asserted until after TXSYSCLKSEL has changed values. The control module contains the necessary logic to sequence the TXPMARESET and TXSYSCLKSEL signals whenever such a dynamic clock change is requested by a change on the SDI wrapper's tx_m port.

The SDI control module also briefly asserts TXPMARESET for a given period of time after the txplllock input to the SDI wrapper becomes asserted. This is useful in cases where a PLL is used to generate the TX reference clock and the output frequency of the PLL might continue to

fluctuate for some time even after it indicates that it is locked. For example, in the pass-through example application included with this application note, a Si5324 digital PLL is used to reduce jitter on the clock recovered by the GTX RX to provide a clean reference clock to the GTX TX. The recovered clock changes frequencies whenever the bit rate of the input SDI signal changes, and the Si5324 might take some time to re-lock after the clock changes frequencies. The TXPMARESET helps ensure that the GTX TX returns to normal operation when the reference clock to the QPLL or CPLL changes in such a manner. The delay between when the txplllock input to the SDI wrapper is asserted until the control module pulses its gtx_txpmareset output is controlled by a parameter/generic to the SDI wrapper named TXPMARESETDLY_MSB. This delay is related to the frequency of the fixed frequency clock on the clk port of the SDI wrapper, as given in Equation 2.

$$\text{TXPMARESETdelay} = \frac{2^{\text{TXPMARESETDLY\_MSB} + 1}}{\text{clk frequency}} \qquad \textit{Equation 2}$$

Some applications might have additional cases where GTTXRESET, GTRXRESET, or TXPMARESET need to be asserted. The reset outputs of the control module can usually simply be ORed with the application's reset conditions before being connected to the reset input ports of the GTX wrapper.

Depending on the application's requirements, other GTX resets might be required. One such reset is the TXPCSRESET. It is recommended that TXPCSRESET be asserted to reset the TX buffer whenever the TX buffer underflows or overflows, as indicated by bit 1 of the GTX TXBUFSTATUS output port. In the example SDI applications provided with this application note, bit 1 of TXBUFSTATUS is always wired directly to the TXPCSRESET input port of the GTX for this reason.

The GTX transceiver has two inputs called RXUSERRDY and TXUSERRDY. These inputs are used by the transceiver's internal reset state machine to determine when to reset the PCS sections of the RX and TX. The control module supplied with this application note does not generate the RXUSERRDY and TXUSERRDY signals. However, the SDI wrapper module, which includes the control module and an instance of the SDI core, does include some simple example logic for generating these signals. The logic in the wrapper that generates the TXUSERRDY signal is just a 5-bit shift register that keeps TXUSERRDY Low until the fifth TXUSRCLK signal occurs. The logic for RXUSERRDY is identical and keeps RXUSERRDY Low until the fifth RXUSRCLK occurs. This is the minimum recommended logic for generating these signals and has proven to be sufficient for most SDI applications. However, users should always refer to the requirements for RXUSERRDY and TXUSERRDY as described in *7 Series FPGAs GTX/GTH Transceivers User Guide* (UG476) and include logic to control these signals appropriately if the example logic in the wrapper is not sufficient for a particular application.

SDI Core Resets

The SDI wrapper has reset inputs for the RX section (rx_rst) and the TX section (tx_rst). The rx_rst input is a synchronous reset in the rx_usrclk clock domain. To fully reset the RX section of the SDI core, both the rx_ce_sd and the rx_din_rdy_3G inputs of the SDI wrapper must be High while rx_rst is High. Asserting rx_rst under these conditions fully resets the RX section of the SDI core. This includes the state machine that controls the SDI mode search algorithm that sequences the GTX RX and SDI RX through the various SDI modes until lock is achieved. Thus, care must be taken when using rx_rst that it not be asserted indiscriminately when not needed. Otherwise, the SDI RX might never lock or might lock much more slowly than it otherwise could.

Most applications do not even need to assert the rx_rst signal. The SDI RX section comes out of FPGA configuration in a working state. If the illegal state recovery logic of the various finite state machines in the SDI RX section are not optimized away during synthesis, the SDI RX is very robust at recovering from almost any sort of abnormal condition. Different synthesizers have their own unique ways of specifying whether or not the illegal state recovery of finite state machines is left intact or optimized away. For the ISE® Design Suite, setting the XST setting

SAFE_IMPLEMENTATION to **TRUE** ensures that illegal state recovery is included with the state machines. It is recommended that **SAFE_IMPLEMENTATION** or its equivalent be enabled in all applications using the SDI core. The default for the ISE tools is for **SAFE_IMPLEMENTATION** to be set to **FALSE**, so this should be changed when processing an SDI application. The example SDI applications included with this application note hardwire rx_rst Low. The BIT files for these example applications were generated in the ISE tools with **SAFE_IMPLEMENTATION** set to **TRUE**. These applications are very robust, showing that, in most cases, rx_rst can simply be hardwired Low.

The tx_rst input is a synchronous reset in the tx_usrclk clock domain. To fully reset the TX section of the SDI core, both tx_ce and tx_din_rdy inputs of the SDI wrapper must be High while tx_rst is High. As with the RX section described in the previous paragraph, the TX section of the SDI core emerges from FPGA configuration in working condition and, as long as the illegal state recovery of the finite state machines is not optimized out, the TX section is very robust. The tx_usrclk is generally driven by the GTX TXOUTCLK through a global clock buffer. TXOUTCLK can have erratic timing whenever TXSYSCLKSEL is dynamically changed to switch the TX clock source between the QPLL and the CPLL and when TXRATE is dynamically changed to change the serial clock divider. While the SDI core's TX section can automatically recover from these abnormal timing conditions on tx_usrclk, often the portions of the application that are clocked by this same clock might not be so robust. This is true of both of the SDI demonstration applications and they both take great care to protect logic clocked by TXOUTCLK when it is known that TXSYSCLKSEL and TXRATE are changing.

In both example SDI applications, there is an 11-bit shift register that is initialized with a pattern of `00000100001`. This shift register is part of the application, not part of the SDI wrapper or SDI core, and it generates the SD-SDI TX clock enable with a 5/6/5/6 clock cycle cadence as required by the TX section of the SDI core when transmitting SD-SDI with the Zynq-7000 SoC GTX transceivers. The shift register shifts right by one bit every tx_usrclk clock cycle, and the LSB of the shift register is wrapped around to the MSB input of the shift register so that this pattern continuously circulates in the shift register. The pattern in the shift register often becomes corrupted when the TXOUTCLK is abnormal because the TXSYSCLKSEL or TXRATE inputs to the GTX are dynamically changed. Thus, whenever either TXSYSCLKSEL or TXRATE are changed, the application code, not the SDI wrapper or SDI core, resets this shift register, restoring the correct pattern required for TX clock enable generation. To be safe, the tx_rst port of the SDI wrapper is also asserted when the shift register is reset, only to ensure that all state machines in the SDI core's TX go to their initial conditions.

Dynamic Bit Rate Switching

Two types of dynamic switching are required to support the various SDI bit rates: switching the serial clock divider and switching the serial clock source. For the SDI RX, only the serial clock divider needs to switch dynamically. For the SDI TX, both types of dynamic switching are used.

Figure 5 is a block diagram view of the signal connections associated with dynamic rate switching.

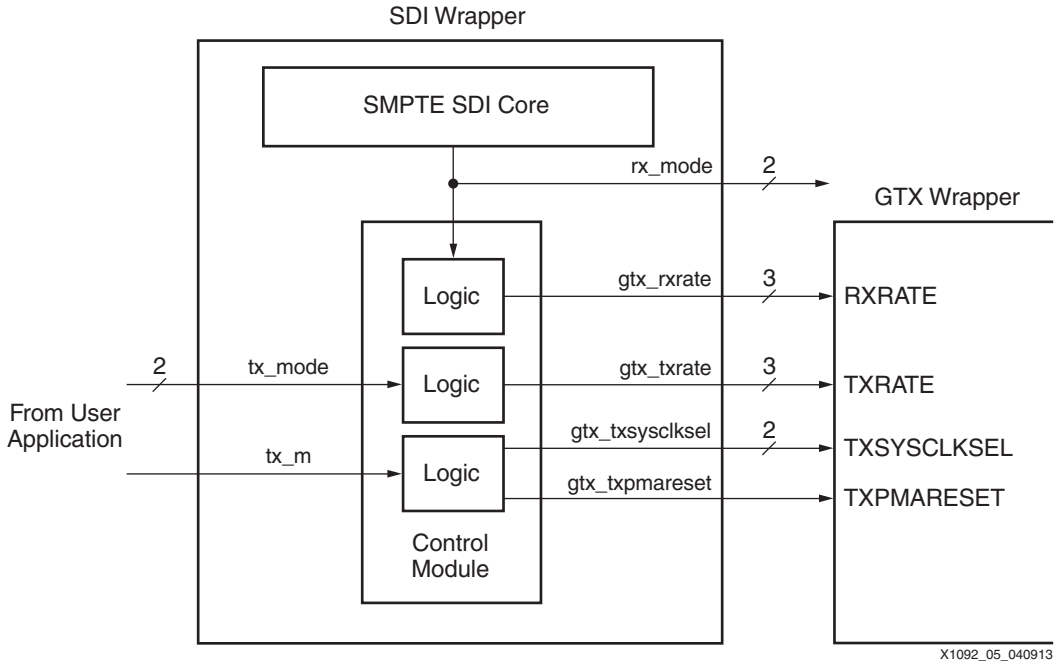


Figure 5: **Signal Connections for Dynamic Rate Switching**

Dynamically Switching the Serial Clock Dividers

Dynamically switching the serial clock dividers changes the line rate of the RX or TX between 3G-SDI operation (~3 Gb/s) and HD-SDI operation (~1.5 Gb/s). For SD-SDI, the GTX RX and TX are configured for the 3G-SDI line rate, and 11X oversampling is used. Each RX and TX unit in a GTX Quad has its own serial clock divider and control port. The serial clock divider of the RX unit is controlled by its RXRATE port. The serial clock divider of a TX unit is controlled by its TXRATE port.

The control module contains logic to control the RXRATE and TXRATE ports of a GTX transceiver through the SDI wrapper's gtx_rxrate and gtx_txrate output ports. The gtx_txrate output port changes in response to a change on the tx_mode input port of the control module. The gtx_rxrate output port changes in response to a change on the rx_mode input port of the control module. In the SDI wrapper, the rx_mode port of the control module is connected internally to the source of the rx_mode signal, the SMPTE SDI core.

In the following discussion, all frequencies are given assuming the GTX is operating at line rates of 1.485 Gb/s or 2.97 Gb/s. When the PLL is locked to a 148.5/1.001 MHz or 74.25/1.001 MHz reference clock, all frequencies would be divided by 1.001.

The frequency of the serial clock out of the PLL divider must be half the line rate. For HD-SDI, the serial clock frequency must be 742.5 MHz. For SD-SDI and 3G-SDI, the serial clock frequency must be 1.485 GHz.

The operating range of the CPLL in the Zynq-7000 SoC GTX transceiver is 1.6 GHz to 3.3 GHz. For SDI applications, the CPLL must operate at 2.97 GHz. So, for HD-SDI bit rates, the serial clock divider must be set to divide by 4, resulting in a serial clock frequency of 742.5 MHz. For SD-SDI and 3G-SDI bit rates, the serial clock divider must be set to divide by 2, resulting in a serial clock frequency of 1.485 GHz.

The QPLL in the Zynq-7000 SoC GTX Quad has two operating ranges. The lower range is from 5.93 GHz to 8 GHz or lower in some packages. The upper range is from 9.8 GHz to some upper limited determined by speed grade and package type. For SDI applications, the lower range should be used, and the QPLL should be operated at 5.94 GHz or 5.94/1.001 GHz, almost exactly at the 5.93 GHz lower limit of the lower range. In fact, this lower limit of the QPLL operating range is specifically set to support SDI applications. The clock from the QPLL is

always divided by two before leaving the QPLL block. Thus, when using the lower QPLL range, the clock into the serial clock divider of an RX or TX unit from the QPLL is actually 2.97 GHz, the same frequency as the clock from the CPLL. Thus, the same two serial clock divider values of 4 for HD-SDI and 2 for SD-SDI and 3G-SDI apply when the QPLL clock is used in its lower operating range. If the QPLL is operating in its upper range, the clock is twice as fast, so a serial clock divider value of 8 must be used in HD-SDI mode and 4 in SD-SDI and 3G-SDI mode.

For the control module to generate the correct RXRATE and TXRATE values, it must know what range the PLL is operating in. This information is supplied to the control module on its gtx_rxpllrange and gtx_txpllrange ports. When using the QPLL in its lower operating range or the CPLL, the gtx_rxpllrange and gtx_txpllrange ports can simply be set Low all of the time. This is the normal mode of operation for SDI applications using the Zynq-7000 SoC GTX transceivers.

If the QPLL is operating in its upper range, any unit using a clock from the PLL must have its corresponding control module range port (gtx_rxpllrange or gtx_txpllrange) set High. Furthermore, if a unit is dynamically switching between using the QPLL operating in the upper range and the CPLL, the value on the range port must change dynamically too. Using Figure 3 as an example and with the QPLL operating in its upper range at a frequency of 11.88 GHz, the gtx_rxpllrange port of each transceiver's control module would be permanently wired High because the QPLL is always used as the clock source for the GTX RX. The GTX TX is switched dynamically between the QPLL and the CPLL. Thus, the gtx_txpllrange port of each transceiver's control module must be dynamically switched. When the GTX TX is using the clock from the QPLL, the gtx_txpllrange port must be High. When the GTX TX is using the clock from the CPLL, the gtx_txpllrange port must be Low. This only applies when using the QPLL in its upper operating range. Normally, for SDI applications, the QPLL operates in its lower range and the gtx_rxpllrange and gtx_txpllrange input ports of the control module (and the SDI wrapper) must be Low all of the time.

Dynamically Switching the TX Clock Source

In applications where the GTX clock sources are configured as shown in Figure 3, the GTX TX units must dynamically switch their clock source between the QPLL and the CPLL to support all of the SD bit rates. In Figure 3, a TX unit uses the QPLL clock to transmit HD-SDI at 1.485 Gb/s, 3G-SDI at 2.97 Gb/s, and SD-SDI at 270 Mb/s. The TX unit uses the CPLL clock to transmit HD-SDI at 1.485/1.001 Gb/s and 3G-SDI at 2.97/1.001 Gb/s.

The GTX TXSYSCLKSEL port controls the clock MUX that selects between the QPLL and the CPLL. This is actually a two-bit port, and the two bits of the port control different MUXes. But, for SDI applications, both bits of the TXSYSCLKSEL port should always be the same and should switch together.

To ensure reliable recovery of the GTX TX after dynamically switching its clock source, the GTX TXPMARESET signal must be asserted before the TXSYSCLKSEL port is changed and must remain asserted for a short period of time after the TXSYSCLKSEL port is changed. The control module contains the necessary logic to control the TXSYSCLKSEL and TXPMARESET ports of the GTX for dynamic switching of the TX clock source. The SDI wrapper's gtx_txsysclksel output port should be connected to the TXSYSCLKSEL port of the GTX wrapper. The SDI wrapper's gtx_txpmareset output port should be connected to the TXPMARESET port of the GTX wrapper. The control module properly sequences the TXPMARESET and TXSYSCLKSEL ports to dynamically switch the clock source between the QPLL and the CPLL in response to a change in value on the SDI wrapper's tx_m port. The values driven to the TXSYSCLKSEL port of the GTX for each value of tx_m are controlled by two parameters/generics to the SDI wrapper called TXSYSCLKSEL_M_0 and TXSYSCLKSEL_M_1. A value of **00** on TXSYSCLKSEL selects the CPLL, and a value of **11** selects the QPLL. Traditionally, Xilinx has used a Low on tx_m to select the 148.5 MHz reference and High on tx_m to select the 148.5/1.001 MHz reference. So, in an application configured as shown in Figure 3 where the QPLL is referenced to 148.5 MHz and the CPLL is referenced to 148.5/1.001 MHz, the SDI wrapper's TXSYSCLKSEL_M_0 parameter/generic

would be assigned a value of **11** to select the QPLL when tx_m is Low and the TXSYSCLKSEL_M_1 parameter/generic would be assigned a value of **00** to select the CPLL when tx_m is High. In fact, the default value of TXSYSCLKSEL_M_0 is **11** and the default value of TXSYSCLKSEL_M_1 is **00**.

Of course, not all applications require dynamic switching of the TX clock source between the QPLL and the CPLL. In those cases, the tx_m port and the two parameters/generics can be assigned to appropriate values to permanently select either the QPLL or the CPLL as the clock source for the GTX TX. Or, the TXSYSCLKSEL port of the GTX wrapper can be hardwired to **00** to select the CPLL or **11** to select the QPLL, and the gtx_txsysclksel output port of the SDI wrapper can be left disconnected.

SDI Electrical Interface

External SDI cable equalizers and cable drivers are required to convert the serial signals into and out of the GTX transceivers to SDI electrical standards.

An external SDI cable equalizer must be used to convert the single-ended 75Ω SDI signal to a 50Ω differential signal compatible with the receiver input signal requirements of the GTX transceiver. Appropriate SDI cable equalizers are available from several manufacturers. The differential outputs of these cable equalizers usually must be AC-coupled to the GTX receiver input signals. An example of interfacing a typical SDI cable equalizer to a GTX receiver is shown in Figure 6. Capacitance values of the coupling capacitors must be large enough to pass the SDI pathological signals without significant signal droop. Typical values used are between 1 µF and 4.7 µF.

The differential inputs of the GTX RX have built-in differential termination. As described in *7 Series FPGAs GTX/GTH Transceivers User Guide* (UG476), RX Termination Use Mode 3 is the recommended termination mode for the GTX RX inputs in SDI applications. The GTX internal programmable termination voltage should be set to 800 mV for SDI applications.

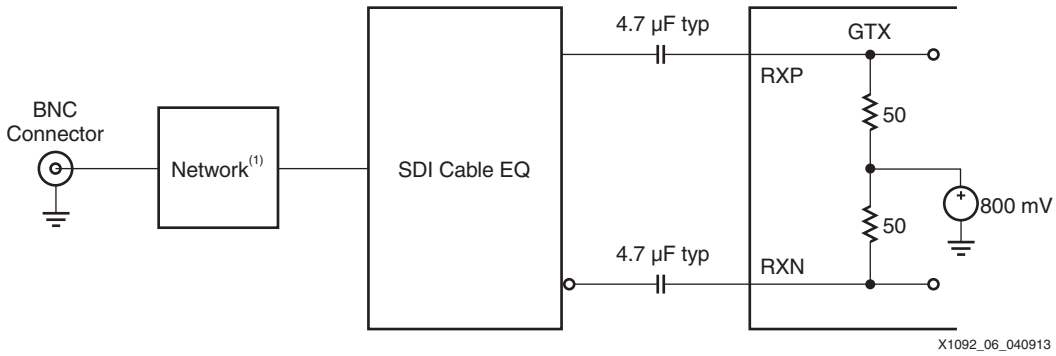


Figure 6: **Interfacing an SDI Cable Equalizer to the GTX Receiver Inputs**

Notes relevant to Figure 6:

1. Consult the SDI cable EQ manufacturer's information for the network between the SDI cable EQ and the BNC connector.

Similarly, the differential serial outputs of the GTX transmitter are connected to the inputs of an SDI cable driver, usually with AC coupling as shown in Figure 7. The cable driver converts the differential signal from the GTX transmitter into a single-ended signal with electrical characteristics meeting the SDI standards. SDI cable drivers typically have a slew rate control input that sets the slew rate of the cable driver. The slew rate requirements for SD-SDI are significantly different than the slew rate requirements for HD-SDI and 3G-SDI. The slew rate control input of the SDI cable driver is typically controlled by the FPGA. The control module supplied with this application note generates a slew rate control signal for use with the external SDI cable driver.

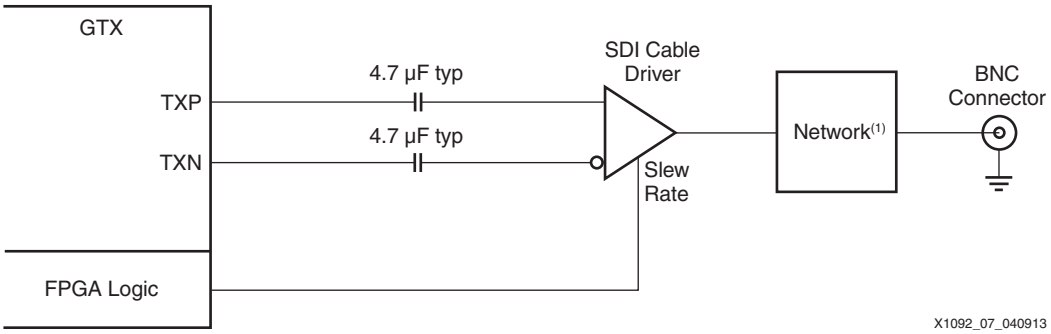


Figure 7: **Interfacing an SDI Cable Driver to the GTX Transmitter Outputs**

Notes relevant to Figure 7:

1. Consult the SDI cable driver manufacturer's information for the network between the SDI cable driver and the BNC connector.

SD-SDI Considerations

Receiving SD-SDI

The 270 Mb/s bit rate of SD-SDI is below the minimum line rate supported by the GTX RX. To receive 270 Mb/s SD-SDI, the GTX RX is used as an asynchronous oversampler to sample the SD-SDI bit stream at 11 times 270 Mb/s (2.97 gigasamples per second (GSPS)) without regard to where bit transitions occur. The CDR unit in the GTX RX is locked to the reference clock by asserting the GTX RXCDRHOLD input port High. This prevents the CDR from trying to lock to the slow SD-SDI signal and results in more uniform oversampling of the SD-SDI signal.

A data recovery unit (DRU), implemented in the programmable logic of the FPGA, examines the oversampled SD-SDI data from the GTX RX, determines the best sample to use for each bit, and outputs the recovered data. This DRU is not part of the SDI core, but is provided as part of this applications note's control module.

The DRU provided with this application note is a version of the DRU described in *Dynamically Programmable DRU for High-Speed Serial I/O* (XAPP875) that has been optimized for recovering 270 Mb/s SD-SDI bit streams from 11X oversampled data. The general-purpose DRU described in *Dynamically Programmable DRU for High-Speed Serial I/O* can recover data using many different oversampling factors and, as a result, is larger and uses more FPGA resources than the optimized version provided here for use with the SDI core.

The SD-SDI standard SMPTE ST 259 [Ref 4] specifies several other bit rates besides 270 Mb/s. The optimized DRU supplied with this application note only supports 270 Mb/s because the vast majority of SDI interfaces only need to support the 270 Mb/s SD-SDI bit rate. However, if other SD-SDI bit rates need to be supported by the application, the optimized DRU can be replaced with the DRU from *Dynamically Programmable DRU for High-Speed Serial I/O*. Because that DRU supports fractional oversampling factors, it is possible to receive the other SD-SDI bit rates without requiring any additional RX reference clock frequencies. The 540 Mb/s SD-SDI bit rate specified by SMPTE ST 344 [Ref 5] is within the supported line rate range of the GTX transceiver, and thus the GTX RX does not need to use the DRU to receive it. However, receiving the 540 Mb/s bit rate without the DRU requires a different reference clock frequency than is used for the other SDI bit rates. Thus, it is usually more convenient to use the DRU from *Dynamically Programmable DRU for High-Speed Serial I/O* to receive the 540 Mb/s ST 344 signal using 5.5X oversampling so that the standard SDI reference clock frequency can be used.

Receiving the additional SD-SDI bit rates also requires modifications to the SDI RX rate detector that controls the locking of the SDI RX by searching sequentially through all SDI bit

rates until the receiver locks. The rate detection algorithm is implemented in the `triple_sdi_rx_autorate.v` or `triple_sdi_rx_autorate.vhd` file supplied with the SMPTE SDI core. Xilinx does not provide an equivalent module that supports the additional SD-SDI bit rates.

The DRU does not recover a clock and, because the CDR unit in the GTX RX is locked to its reference clock, the RXOUTCLK is not locked to the incoming bit rate in SD-SDI mode. The DRU does produce a data strobe indicating when a 10-bit data word is ready on its output. This data strobe is used by the SDI core to generate a clock enable that is asserted at a 27 MHz rate, typically with a 5/6/5/6 cadence relative to the RXOUTCLK clock from the GTX. The rx_ce_sd output of the SDI wrapper is derived from the DRU data strobe and has the same cadence. Occasionally the cadence of the DRU data strobe and the rx_ce_sd signal varies from the typical 5/6/5/6 cadence. This occurs when the DRU needs to make up for the slight difference between the actual SD-SDI bit rate and the frequency of the local reference clock provided to the GTX RX.

Figure 8 is a screen capture from an oscilloscope showing the 27 MHz rx_ce_sd signal. The scope is triggered on the rising edge of rx_ce_sd at the center of the screen. The scope is in infinite persistence mode and the waveform was allowed to accumulate for several minutes. The waveform is temperature-coded from red (indicating the most common position of the signal) to blue (indicating the least common position). The incoming SD-SDI signal that was used to create this screen capture was asynchronous to the local reference clock used by the GTX receiver. The rx_ce_sd pulses on either side of the center pulse are always 5 or 6 clock cycles away from the center pulse because of the 5/6/5/6 cadence of the rx_ce_sd signal.

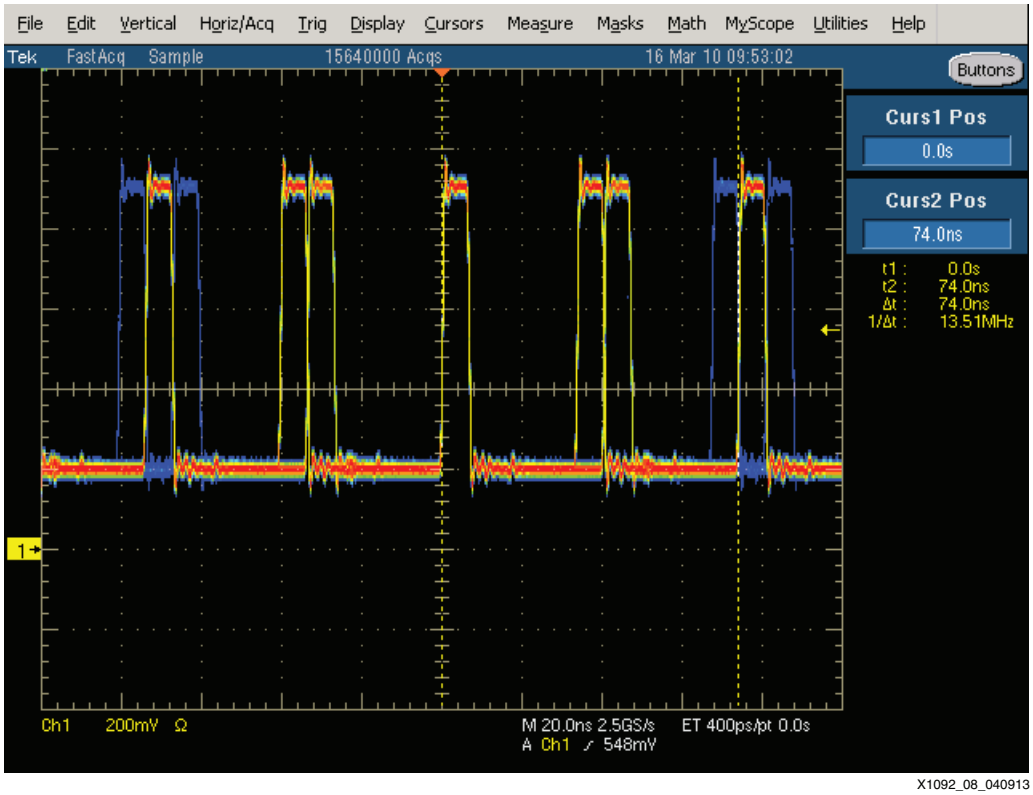


Figure 8: **Oscilloscope Capture of SD-SDI Clock Enable**

The two pulses at the far right and far left of the trace are nominally 11 clock cycles from the center pulse because of the 5/6/5/6 cadence. The nominal position is marked by the yellow and red pulse. For the far right pulse, the dashed yellow vertical cursor marks the position that is 11 clock cycles from the rising edge of the center pulse. The nominal locations of the central

yellow/red pulses are surrounded on either side by blue pulses indicating that the DRU occasionally needs to make the period of the rx_ce_sd cycle either 10 clock cycles or 12 clock cycles long to compensate for the frequency differences between the local reference clock and the incoming SD-SDI signal.

The SD-SDI DRU is supplied with this application note as an encrypted, pre-generated file called `dru.ngc`. It is not possible to do any simulation of a design using the `dru.ngc` file because of its encryption. However, the files `dru_sim.v` and `dru_sim.vhd` included with this application note provide simplified simulation models of the DRU. Either of these files can be used during simulation to replace the `dru.ngc` file. However, these simulation models should not be used in a design intended for use in the actual FPGA because the models do not support any variation in frequency between the GTX RX reference clock and the SD-SDI bit stream.

Transmitting SD-SDI

As with reception of SD-SDI, transmission of the slow 270 Mb/s SD-SDI bit rate is not directly supported by the GTX TX. To transmit the SD-SDI signal, the GTX TX is configured for a line rate of 2.97 Gb/s. The SDI core replicates each bit to be transmitted 11 times so that the data out of the SDI core and into the TXDATA port of the GTX TX contains 11 consecutive copies of each bit. The resulting signal output by the GTX TX is a valid 270 Mb/s SD-SDI signal.

Generating an SD-SDI Recovered Clock

In SD-SDI mode, the RXOUTCLK of the GTX RX is not really a recovered clock because the CDR unit is locked to the frequency of the reference clock, not to the SD-SDI bit stream. The only signal available that actually indicates the data rate of the incoming SD-SDI bit stream is the 27 MHz rx_ce_sd output of the SDI wrapper.

For some video applications, particularly those that do not need to retransmit the recovered video over an SDI interface, the rx_ce_sd signal might be sufficient as a recovered clock. Typically, this signal is used as a clock enable to downstream modules that are clocked with the RXOUTCLK from the GTX receiver. This is how the SDI datapath in the SDI core works – using the rx_ce_sd signal as a clock enable.

If the received video data is to be retransmitted as an SD-SDI signal using a GTX TX, a low-jitter recovered clock is required. The recovered clock must have low enough jitter that it can be used as a reference clock for the GTX transmitter PMA PLL. Furthermore, the frequency of the recovered clock must be 74.25 MHz or 148.5 MHz so that the GTX transmitter can use 11X oversampling to transmit the 270 Mb/s SD-SDI data. This requires the use of an external, low-bandwidth PLL or use of the PICXO technique. (The PICXO technique is not covered in this application note. Contact Xilinx technical support for inquiries about the PICXO technique and SDI.) The bandwidth of the mixed-mode clock manager (MMCM) in the Zynq-7000 SoC is too high to adequately filter out the large amounts of low-frequency jitter present on the rx_ce_sd signal from the SDI receiver. The National Semiconductor LMH1983 and the Silicon Labs Si5324 PLLs can both perform this function. Both of these devices can take in the rx_ce_sd signal as a 27 MHz reference and multiply it up to either 74.25 MHz or 148.5 MHz while also filtering out the jitter. The resulting clock is suitable for use as a reference clock for the GTX TX. The pass-through demonstration included with this application note uses an Si5324 to generate a 148.5 MHz reference clock for the GTX TX from the 27 MHz rx_ce_sd signal in exactly this manner in SD-SDI mode. When retransmitting either HD-SDI or 3G-SDI, the same Si5324 is reprogrammed to filter jitter from the RXOUTCLK output of the GTX RX, doubling its frequency in the case of HD-SDI, thereby producing a low-jitter 148.5 MHz reference clock for the GTX TX.

Another alternative is to use an external genlock PLL and lock it to the video sync signals from the recovered video. The output of the genlock PLL is an SD-SDI recovered clock.

Sometimes a recovered clock is required to drive external video application-specific standard product (ASSP) devices. In SD-SDI mode, such a clock probably needs to have a frequency of 27 MHz and have lower jitter than is present on the rx_ce_sd signal, but does not need to have very low jitter as is the case when producing a GTX TX reference clock. The techniques

mentioned previously can be used, but it might be preferable to generate such a recovered clock entirely in the SoC without requiring external components. Unfortunately, the jitter on the rx_ce_sd signal is too high to allow it to be used directly as a reference clock input to the MMCM. But, there is a way to generate a recovered SD-SDI clock using a spare GTX transmitter, as shown in Figure 9.

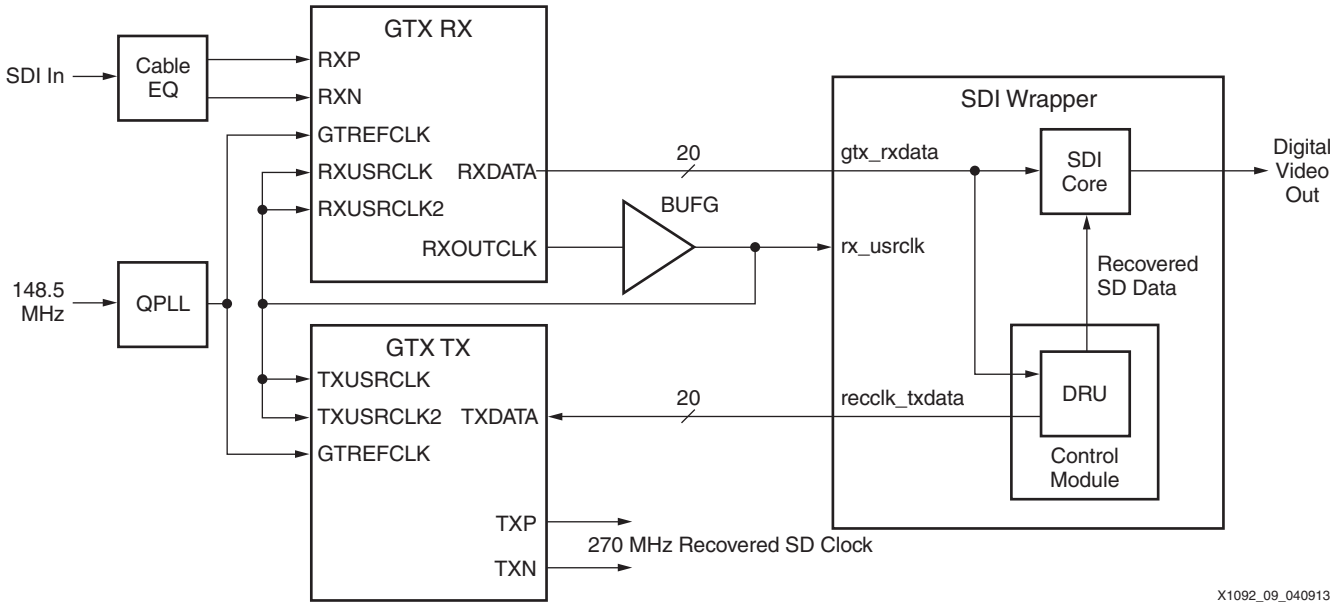


Figure 9: **Using a GTX TX to Generate an SD-SDI Recovered Clock**

The control module's recclk_txdata port can be connected to the TXDATA port of a spare GTX transmitter. The GTX TX must use the same reference clock as the GTX RX that is receiving the SDI input signal. The TXUSRCLK and TXUSRCLK2 ports of the GTX TX must be connected to the same clock that is driving the RXUSRCLK and RXUSRCLK2 ports of the GTX TX and the rx_usrclk port of the SDI wrapper. The GTX TX must be configured for a line rate of 2.97 Gb/s with no encoding and with a 20-bit TXDATA port.

When configured in this manner, the serial output of the GTX transmitter is a 270 MHz clock that is frequency-locked to the incoming SD-SDI signal. In other words, it is a true recovered clock for SD-SDI. The GTX transmitter serial output pins can be connected to a global or regional clock LVDS input of the Zynq-7000 SoC, with appropriate care to properly terminate the GTX transceiver TX outputs and translate them to LVDS. The 270 MHz clock can then be used in whatever manner is required in the SoC. For example, it can be divided by 10 to get a 27 MHz recovered clock to drive internal or external video datapaths. The signal has low enough jitter that it can be used as a reference clock to an MMCM.

The recclk_txdata port of the DRU is not wired from the control module to an output port in the SDI wrapper supplied with this application note. However, if an application needs to use this feature, it is a simple matter to edit the SDI wrapper to add this output port.

The GTX TX that is used to generate the recovered SD-SDI clock does not have to be configured for SDI. It only needs to be configured to always run at 2.97 Gb/s with no encoding. The data supplied to the TXDATA port of the GTX from the recclk_txdata port of the control module creates a 270 MHz clock on the GTX TX serial output pins. The edges of the generated clock move around by plus or minus one bit time of the 2.97 Gb/s line rate to modify the frequency of the output signal so as to exactly match with the bit rate of the input SD-SDI signal. Thus, the cycle-to-cycle jitter on the 270 MHz clock generated by the GTX TX is ±337 ps plus whatever jitter is inherent in the GTX TX output signal (1 bit time at 2.97 Gb/s is 337 ps). This can be seen in Figure 10. The top trace is the 270 MHz clock generated by the GTX TX. The scope was triggered on the rising edge of the recovered clock at the center of the screen.

Looking at the rising edges of the cycles on either side of the trigger point, the ±337 ps cycle-to-cycle jitter is clearly seen because these rising edges each have three discrete rising points. The bottom trace in Figure 10 is the SD-SDI that is being retransmitted by another GTX TX.

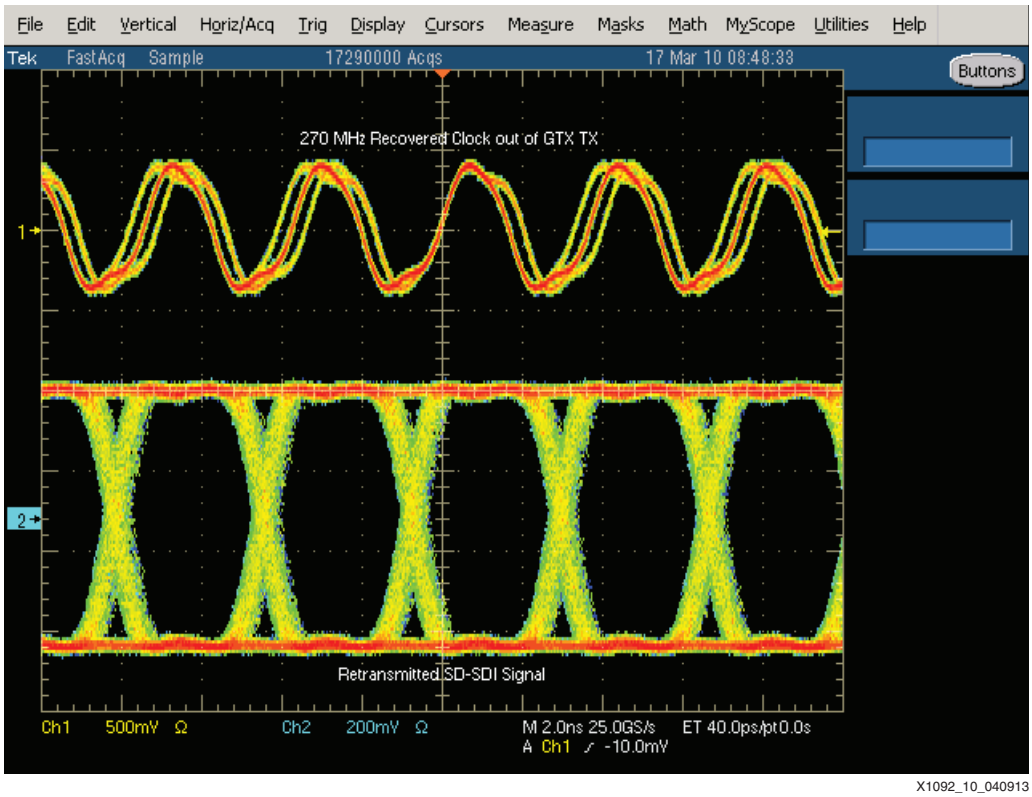


Figure 10: **Recovered SD-SDI Clock from GTX Transceiver**

The recclk_txdata port is not output from the SDI wrapper. This is because this port is not used by most SDI applications. If needed, the SDI wrapper can be edited to add a new port and connect it to the recclk_txdata port of the control module.

RX Bit Rate Detection

The SDI core can automatically determine the SDI mode (SD-SDI, HD-SDI, or 3G-SDI) of the SDI signal coming into the GTX RX. When it is not locked to the current SDI input signal, the SDI core sequences the GTX RX through the three different SDI modes until it detects recognizably good SDI data on the RXDATA output port of the GTX. At that point, the SDI core indicates that it is locked to the SDI signal by asserting its rx_mode_locked output. It also indicates which SDI mode the RX is locked to on its sdi_mode output port.

However, when the SDI core is in HD-SDI mode, it has no way of determining if the bit rate of the input SDI signal is 1.485 Gb/s or 1.485/1.001 Gb/s. Likewise, in 3G-SDI mode, the SDI core cannot determine whether the bit rate of the input SDI signal is 2.97 Gb/s or 2.97/1.001 Gb/s. The control module supplied with this application note, however, contains a bit rate detector that can distinguish between 1.485 Gb/s and 1.485/1.001 Gb/s, and between 2.97 Gb/s and 2.97/1.001 Gb/s. The SDI wrapper output port rx_bit_rate is Low when the input SDI signal's bit rate is either 1.485 Gb/s or 2.97 Gb/s. The rx_bit_rate is High when the input SDI signal's bit rate is either 1.485/1.001 Gb/s or 2.97/1.001 Gb/s.

For the bit rate detection feature to work, the SDI wrapper must be supplied with a fixed-frequency clock on its clk input port. It is recommended that the frequency of this clock be at least 10 MHz. If the frequency is over 150 MHz, it might be difficult to meet timing in the bit

rate detection logic. The SDI wrapper has a parameter/generic called FXDCLK_FREQ that must be used to specify the frequency of the clock connected to the clk port. The value of FXDCLK_FREQ must be set equal to the frequency of the fixed-frequency clock in Hz.

The SDI wrapper uses the fixed-frequency clock for other purposes besides RX bit rate detection. Thus, even if the bit rate detection feature is not used in a particular application, a fixed-frequency clock must be supplied to the clk port of the SDI wrapper.

Implementing an SDI Interface in a Zynq-7000 SoC

There are several steps required to implement an SDI interface in a Zynq-7000 SoC design. Those steps are:

1. Generate a GTX wrapper using the 7 Series FPGAs Transceivers Wizard.
2. Generate the SMPTE SDI LogiCORE IP using the CORE Generator tool or the Vivado IP catalog.
3. Instance the GTX wrapper and the SDI wrapper from this application note into the application.
4. Put the `dru.ngc` file from this application note into the ISE tools project directory or add it to the Vivado tools project (see the `readme.txt` file in `xapp1092.zip` for more information).
5. Apply proper timing constraints for the SDI wrapper.

Generating the GTX Wrapper

Use the 7 Series FPGAs Transceivers Wizard, called the *Wizard* for the rest of this document, to generate a GTX wrapper. With previous generations of Xilinx GT transceivers, it was possible to create a GT wrapper using the Wizard that contained a single GT transceiver and then instance that wrapper multiple times even if the transceivers were in the same GT tile. With the 7 series GTX transceiver, this is no longer possible. The user must generate a GTX wrapper that contains exactly the number of transceivers that the application uses in the GTX Quad tile or multiple Quad tiles. This is because the Wizard always instances a GTXE2_COMMON in the GTX transceiver, and there is only one GTXE2_COMMON block per GTX Quad. Thus, if the user tries to place two GTX wrappers (each with a GTXE2_COMMON block in them) in the same GTX Quad, the tools flag this as an error.

For those transceivers that are used to implement SDI interfaces, select the **hd sdi** protocol template. This chooses the correct GTX port set and the attributes needed to support SDI. It is highly recommended that GTX wrappers used for SDI applications be generated with the hd sdi protocol template and not with the 3g sdi or sd sdi protocol templates.

The following information details exactly the steps required to generate the GTX wrapper using the Wizard version 2.4. This example creates a GTX wrapper that includes all four GTX transceivers in one GTX Quad all configured for SDI.

The 7 Series FPGAs Transceivers Wizard is found in the **IO Interfaces** folder in the top-level **FPGA Features and Design** folder.

Figure 11 shows the first page of the Wizard. On this page, select **hd sdi** from the **Protocol template** drop-down menu. This sets the line rate to 1.485 Gb/s, and the reference clock frequencies default to 148.5 MHz. If a different reference clock frequency such as 74.25 MHz is required, the reference clock drop-down menus for the TX and RX show the only reference clock frequencies that are supported. Even when one of the reference clocks is actually 148.5/1.001 MHz, both reference clock frequencies should be set to 148.5 MHz and both line rates set to 1.485 Gb/s. The control module takes care of dynamically changing the line rate to 2.97 Gb/s for 3G-SDI and HD-SDI modes. Thus, the line rate should always be set to 1.485 Gb/s in the Wizard.

Figure 11: **7 Series FPGAs Transceivers Wizard - Page 1**

When implementing only an SDI receiver or only an SDI transmitter, the **TX off** or **RX off** check boxes on this page can be used to disable the unneeded portion of the transceiver. The ports for the unused portion of the transceiver do not appear on the GTX wrapper and that portion of the transceiver is powered down. However, the **TX off** and **RX off** selections apply to all transceivers included in the GTX wrapper. Thus, if some transceivers need both RX and TX and others need only the RX or only the TX, the **TX off** and **RX off** check boxes cannot be used. In these cases, the GTX wrapper must be created with the RX and TX sections of all transceivers enabled. The RX and TX units that are not needed can be individually powered down by selecting the **RXPOWERDOWN** and **TXPOWERDOWN** ports from the **Optional Ports** list on page 5 of the Wizard (Figure 15). This adds the ports that enable the power down mode for the RX and TX to each transceiver in the wrapper.

Move to page 2 of the Wizard by clicking the **Next >** button at the bottom of the page. Page 2 of the Wizard allows the user to select which transceivers are included in the GTX wrapper (see Figure 12).

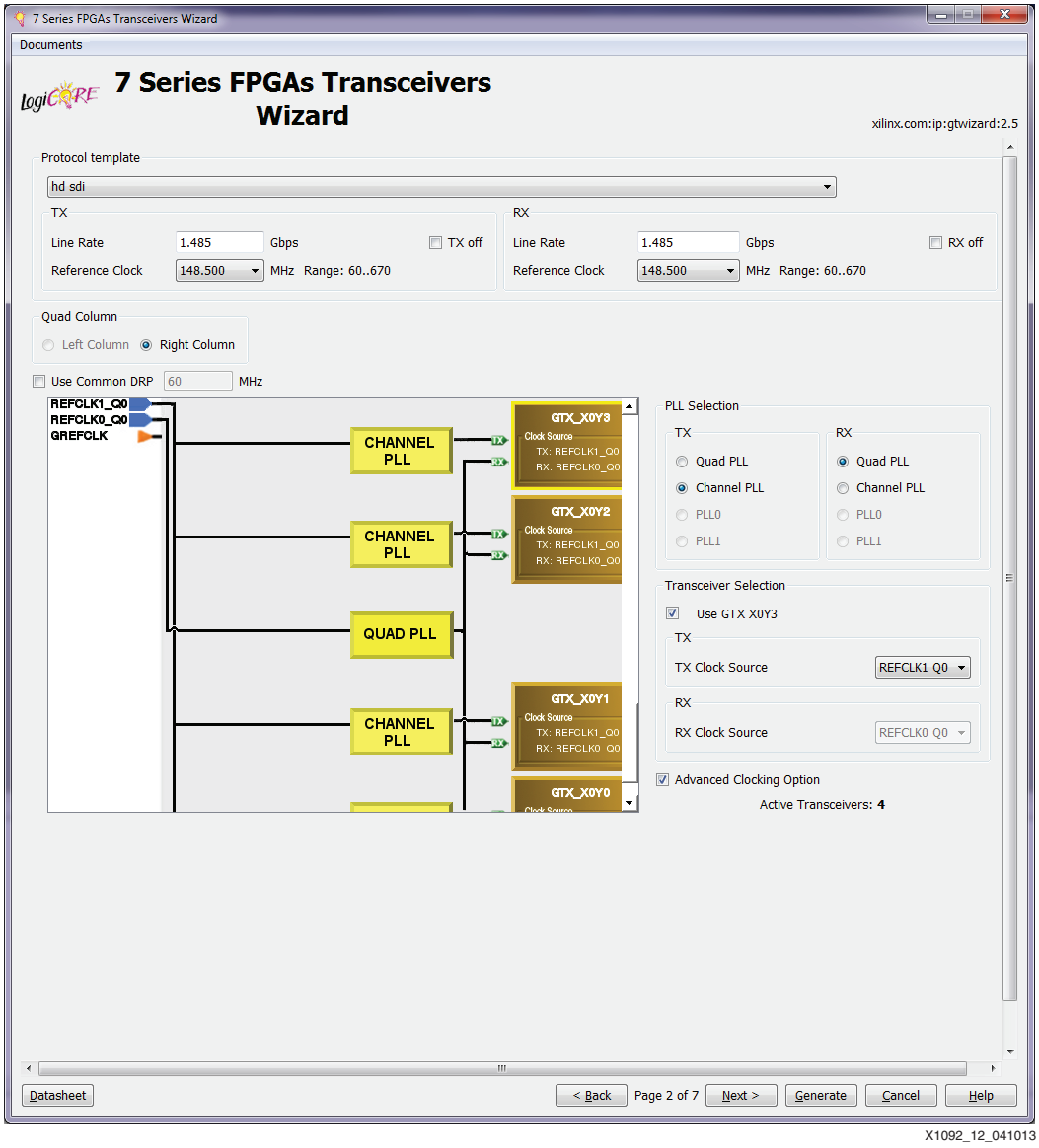


Figure 12: **7 Series FPGA Transceivers Wizard - Page 2**

Usually, the Common DRP (the DRP for the GTXE2_COMMON block containing the QPLL) is not required for SDI applications, so the **Use Common DRP** check box can be unchecked.

Select the GTX transceivers to be included in the GTX wrapper. Also select the PLL for the RX and TX portions of each transceiver. Typically, QPLL is used for all receivers and CPLLs are used for each transmitter. The reference clock source for each PLL can also be made here.

The PLL selection affects how the GTX wrapper is created. For the RX section, which typically has a static PLL selection in SDI applications, the attribute that specifies whether the clock comes from the QPLL or the CPLL is determined by the PLL selection made on this page for each transceiver. Typically, the QPLL is used as the clock source for all RX units in SDI applications. For the TX section, many SDI applications require dynamic switching between the QPLL and the CPLL as controlled by the TXSYSCLKSEL port. Thus, the PLL selection for the TX units does not really matter as far as the GTX wrapper is concerned. However, if an SDI application does not require dynamic switching between the QPLL and the CPLL, the PLL to be used for each TX unit should be specified correctly on this page, and the TXSYSCLKSEL port should be unselected on page 3 of the Wizard (Figure 13). When this is done, the PLL assignments made in the Wizard are statically implemented in the GTX wrapper.

The reference clock sources to the PLLs made on this page do not affect the way the GTX wrapper is created. They only affect the example design that is also created by the Wizard. If the example design is to be used as the starting point for the application, the reference clock sources should be set correctly. Otherwise, it does not matter.

Move to page 3 of the Wizard by clicking the **Next >** button at the bottom of the page (Figure 13). The external and internal data widths are already correctly set to **20 Bits** and the encoding and decoding are already correctly set to **None**.

Figure 13: **7 Series FPGA Transceivers Wizard - Page 3**

The control module does require a DRP port on each transceiver to dynamically change some of the transceiver's attributes when the transceiver is switched between SDI modes. Thus, the **Use DRP** check box must be selected, and the frequency of the DRPCLK must be correctly specified. In this example, the DRPCLK frequency is set to **74.25 MHz**. Any free-running clock available in the design can be used as the DRPCLK as long as it meets the frequency requirements as specified by FGTXDRPCLK in *Zynq-7000 All Programmable SoC (XC7Z030, XC7Z045, and XC7Z100): DC and AC Switching Characteristics* (DS191). The frequency of the clock source driving the DRPCLK should not change when the SDI mode changes. It should

remain at a constant, fixed frequency at all times, and that frequency must be specified here on Page 3 of the Wizard.

Xilinx recommends that the TX and RX buffers both be enabled. Bypassing these buffers is not supported by the control module included with this application note.

All optional ports selected on this page by default when the hd sdi protocol template was chosen are required for SDI applications with one possible exception. **TXSYSCLKSEL** is normally used to allow the TX unit to be dynamically switched between the QPLL and the CPLL. If this is not required, the **TXSYSCLKSEL** port can be unselected, and the PLL selection for the TX units in the wrapper are specified on page 2 of the Wizard (Figure 12).

Move to page 4 of the Wizard by clicking the **Next >** button at the bottom of the page (Figure 14).

Figure 14: **7 Series FPGAs Transceivers Wizard - Page 4**

Verify that all selections on this page are as shown in Figure 14. In particular, **Use comma detection** and the **RXSLIDE** port should not be selected. Some versions of the Wizard (including version 2.4) have a bug that causes the **Use comma detection** and **RXSLIDE** selections to be re-enabled when certain changes are made on other pages of the Wizard. So

always check that **Use comma detection** and **RXSLIDE** are not selected after making any changes to settings on other pages of the Wizard.

Xilinx recommends that **Equalization Mode** be set to **LPM-Auto** for SDI applications.

The **RX Termination** must use the **Programmable** termination voltage with **RX Termination Voltage** set to **800 mV.**

The **TXDIFFCTRL** optional port can be enabled if control of the differential swing at the output of the GTX TX is required. Likewise, **TXPOSTCURSOR** and **TXPRECURSOR** can be enabled if those ports are required to improve the signal quality between the GTX TX output and the input of the SDI cable driver.

Move to page 5 of the Wizard by clicking the **Next >** button at the bottom of the page (Figure 15).

Figure 15: **7 Series FPGAs Transceivers Wizard - Page 5**

If loopback testing of the transceivers is desired, the optional **LOOPBACK** port can be enabled. The **TXPOWERDOWN** and **RXPOWERDOWN** ports can be enabled if the applications need power-down control over each individual RX and TX unit. Otherwise, none of the optional ports should be selected.

At this point, all selections necessary for creating a GTX wrapper for SDI applications have been made. Page 6 of the Wizard does not apply to SDI, and page 7 is just a summary page. The user can either move through those pages using the **Next >** button or generate the GTX wrapper from this page using the **Generate** button at the bottom of the page.

The Wizard generates a number of files. The two important files are called `<component_name>.v/vhd` and `<component_name>_gt.v/vhd`. In this example, the component name is `z7gtx_sdi_wrapper`, as can be seen near the top of page 1 of the Wizard in Figure 11. In this example, Verilog was set as the preferred language. Thus, the two important files created by the Wizard are named `z7gtx_sdi_wrapper.v` and `z7gtx_sdi_wrapper_gt.v`.

The `<component_name>` file is the top-level wrapper file, the one that must be instantiated into the SDI application. The `<component_name>_gt` file is a wrapper file for a single GTX transceiver. It is instantiated one or more times in the `<component_name>.v` wrapper file depending on how many transceivers are included in the GTX wrapper.

Note: The example design generated by the Wizard is not particularly useful for SDI applications. The example designs provided by this application note are usually better starting points for an SDI application.

Generating the SMPTE SD/HD/3G-SDI LogiCORE IP

Use the CORE Generator tool or the Vivado IP catalog to generate the SMPTE SD/HD/3G-SDI core. Do not use the older Triple-Rate SDI core, which is only for Virtex®-6 FPGAs. The SMPTE SD/HD/3G-SDI core is the generic SDI core that works with 7 series devices.

The SMPTE SD/HD/3G-SDI core is a source code core, not a precompiled core. When the CORE Generator tool generates the SMPTE SD/HD/3G-SDI core, it delivers a set of source code files in either Verilog or VHDL, depending on project's preferred language setting.

The only option available when generating the SMPTE SD/HD/3G-SDI core is whether or not to include the error detection and handling (EDH) processor for the RX section. Even if the RX EDH processor is not included, the SDI core has all RX EDH ports, but they are inactive.

The SMPTE SD/HD/3G-SDI core is instantiated in the SDI wrapper. Thus, if the SDI wrapper is used, the SMPTE SD/HD/3G-SDI core itself does not need to be directly instantiated in the application.

Instancing the GTX and SDI Wrappers

The GTX and SDI wrappers need to be instantiated and interconnected in the user design. It is possible to implement the SDI interface without the SDI wrapper supplied with this application note, but the wrapper makes things easier because it interconnects the SDI control module and the SDI core. If the wrapper is not used, the user must make all of these connections. The SDI wrapper file is called `x7gtx_sdi_rxtx_wrapper.v` (Verilog version) or `x7gtx_sdi_rxtx_wrapper.vhd` (VHDL version). In addition to the SDI core, it also instances these files:

- `x7gtx_sdi_control.v/vhd`
- `x7gtx_reset_control.v/vhd`
- `x7gtx_sdi_drp_control.v/vhd`
- `sdi_rate_detect.v/vhd`
- `dru_bshift10to10.v/vhd`
- `dru_maskencoder.v/vhd`
- `dru_control.v/vhd`
- `dru_rot20.v/vhd`
- `dru.v` (Verilog only)

Add the dru.ngc File to the Project

The dru.v file is an empty module which, in Verilog, specifies the ports on the precompiled dru.ngc file. When using the x7gtx_sdi_rxtx_wrapper.v file, the dru.v file must be included in the project. When using the x7gtx_sdi_rxtx_wrapper.vhd VHDL file, the component definition serves the same purpose as the dru.v file, thus the dru.v file is not required.

When using the ISE tools, the dru.ngc file included with this application note must be moved into the ISE tools project directory where the tools can find and include it in the design. When using the Vivado tools, the dru.ngc file must be added to the project as a source file just like adding any of the Verilog or VHDL files. The dru.ngc file is the pre-generated and encrypted DRU module.

> ***Caution!*** Do not use the dru_sim.v or dru_sim.vhd files that are included with this application note in a design intended to be used in the actual FPGA. These files are for simulation purposes only. Using them in an actual hardware implementation results in an SDI receiver that is not able to correctly receive SD-SDI signals. For simulation purposes, the dru_sim.v/vhd files can be added to the design instead of the dru.v file and the dru.ngc file.

IMPORTANT: The SDI wrapper contains an instance of the SMPTE SD/HD/3G-SDI core. The SDI wrapper must be edited so that the name given to the SDI core (when it is generated using the CORE Generator tool or the Vivado IP catalog) is used where the core is instanced in the SDI wrapper. This can be avoided by using the component name *smpte_sdi* when generating the SMPTE SDI core.

Table 1 describes all of the ports of the SDI wrapper. This port list is similar to the port list of the SDI core itself, but there are some differences. Also refer to the example SDI applications provided with this application note for examples of how to interconnect the GTX and SDI wrappers.

Some signals are described as being asserted for some number of video sample periods. A video sample period lasts for differing numbers of cycles of the appropriate clock (either tx_usrclk or rx_usrclk) depending on the SDI mode. In HD-SDI and 3G-SDI level A modes, a sample period lasts one clock cycle. In SD-SDI mode, a sample period is either 5 or 6 clock cycles long and begins and ends with the rising edge of the clock when the clock enable (either tx_ce or rx_ce_sd) is asserted. In 3G-SDI level B mode, a sample period is two clock cycles long as controlled by the assertion of the 3G-SDI data ready signal (either tx_din_rdy or rx_dout_rdy_3G).

Most of the RX and TX ports in this list are wired directly to the ports of the same name on the SDI core that is instantiated inside the SDI wrapper. Timing diagrams of the video and video timing signals can be found in *Society of Motion Picture and Television Engineers (SMPTE) SD/HD/3G-SDI Product Guide* [Ref 6].

Table 1: **SDI Wrapper Port List**

Port Name	I/O	Width	Description
clk	In	1	This input must be connected to a fixed-frequency free running clock. This clock is used by the SDI wrapper for various timing purposes. The frequency of this clock must be specified by the parameter/generic FXDCLK_FREQ. If the clock frequency does not closely match the frequency specified by FXDCLK_FREQ, the timing delays generated by the wrapper are not correct and the RX bit rate detection circuit might not function.

Table 1: **SDI Wrapper Port List *(Cont'd)***

Port Name	I/O	Width	Description
Receive Ports			
rx_rst	In	1	This synchronous reset input can normally be hardwired Low because a reset is not required. After FPGA configuration, the SMPTE SDI core is in a fully operational mode and does not require a reset. Both rx_ce_sd and rx_din_rdy_3G must be High when rx_rst is High to completely reset the receiver. Asserting rx_rst also resets the state machine that controls the automatic SDI mode lock detector. Do not assert rx_rst just because the SDI RX is not locked, otherwise the SDI RX never locks.
rx_usrclk	In	1	This input connects to the recovered clock from the transceiver's RX unit: the RXOUTCLK output of the GTX transceiver buffered by a global clock buffer. The clock frequency must be 148.5 MHz (or 148.5/1.001 MHz) for 3G-SDI and SD-SDI modes. It must be 74.25 MHz (or 74.25/1.001 MHz) for HD-SDI mode. All input and outputs of the wrapper prefixed with ***rx***_ are synchronous with this clock.
rx_frame_en	In	1	This input enables the SDI framer function. When this input is High, the framer automatically readjusts the output word alignment to match the alignment of each timing reference signal (TRS): end of active video (EAV), or start of active video (SAV). Normally, this input should always be High. However, if controlled properly, this input can be used to implement TRS alignment filtering. For example, if the rx_nsp output is connected to the rx_frame_en input, the framer ignores a single misaligned TRS, keeping the existing word alignment until the new word alignment is confirmed by a second matching TRS. If a TRS alignment filtering scheme is employed, it is very important to turn off any TRS filtering during the synchronous switching lines by driving the rx_frame_en input High on the synchronous switching lines.
rx_mode_en	In	3	This port has unary bits to enable reception of each of the three SDI modes: • Bit 0 enables HD-SDI mode • Bit 1 enables SD-SDI mode • Bit 2 enables 3G-SDI mode When a bit is High, the corresponding SDI mode is included in the search for the correct SDI mode when the SDI RX is not locked to the incoming signal. When a bit is Low, the SDI RX does not attempt to detect incoming SDI signals of that mode. Disabling unused SDI modes using these bits decreases the amount of time it takes for the SDI RX to lock to the incoming signal when it changes modes.

Table 1: **SDI Wrapper Port List *(Cont'd)***

Port Name	I/O	Width	Description
rx_mode	Out	2	This output port indicates the current SDI mode of the SDI RX: • 00 = HD-SDI • 01 = SD-SDI • 10 = 3G-SDI When the receiver is not locked, the rx_mode port changes values as the SDI RX searches for the correct SDI mode. During this time, the rx_mode_locked output is Low. When the SDI RX detects the correct SDI mode, the rx_mode_locked output goes High and the mode of the incoming SDI signal is indicates by the rx_mode port.
rx_mode_HD rx_mode_SD rx_mode_3G	Out	1	These three output ports are decoded versions of the rx_mode port. Unlike the rx_mode port, which changes continuously as the SDI RX seeks to identify and lock to the incoming signal, these outputs are all forced Low when the SDI RX is not locked. The output matching the current SDI mode of the SDI RX is High when rx_mode_locked is High.
rx_mode_locked	Out	1	When this output is Low, the SDI RX is actively searching for the SDI mode that matches the input data stream. During this time, the rx_mode output port changes frequently. When the SDI RX locks to the correct SDI mode, the rx_mode_locked output goes High.
rx_bit_rate	Out	1	This output port indicates which bit rate is being received in HD-SDI and 3G-SDI modes as shown below. This output is not valid in SD-SDI mode. HD-SDI mode: • rx_bit_rate = 0: Bit rate = 1.485 Gb/s • rx_bit_rate = 1: Bit rate = 1.485/1.001 Gb/s 3G-SDI mode: • rx_bit_rate = 0: Bit rate = 2.97 Gb/s • rx_bit_rate = 1: Bit rate = 2.97/1.001 Gb/s
rx_t_locked	Out	1	This output is High when the transport detection function in the SDI RX has identified the transport format of the SDI signal.
rx_t_family	Out	4	This output indicates which family of video signals is being used as the transport signal on the SDI interface. This output is only valid when rx_t_locked is High. This port does not necessarily identify the video format of the picture being transported. It only identifies the transport characteristics. See Table 4 for the encoding of this port.
rx_t_rate	Out	4	This output indicates the frame rate of the SDI transport signal. This is not necessarily the same as the frame rate of the actual picture. See Table 5 for the encoding of this port. This output is only valid when rx_t_locked is High.
rx_t_scan	Out	1	This output indicates whether the SDI transport signal is interlaced (Low) or progressive (High). This is not necessarily the same as the scan mode of the actual picture. This output is only valid when rx_t_locked is High.
rx_level_b_3G	Out	1	This output is asserted High when the input 3G-SDI signal is level B and Low when it is 3G-SDI level A. This output is only valid when the SDI RX is locked to a 3G-SDI signal (when rx_mode_3G is High).

Table 1: **SDI Wrapper Port List *(Cont'd)***

Port Name	I/O	Width	Description
rx_ce_sd	Out	1	This output is a clock enable for SD-SDI mode. This output is asserted, on average, one cycle of rx_usclk out of every 5.5 cycles in SD-SDI mode. The SD-SDI data stream output on the rx_ds1a port and the RX video timing signals (rx_trs, rx_eav, and rx_sav) are only valid when rx_ce_sd is High in SD-SDI mode. In other SDI modes, rx_ce_sd is always High.
rx_nsp	Out	1	When this output is High, it indicates that the SDI framer has detected a TRS (EAV or SAV) at a new word alignment. If rx_frame_en is High, this output is only asserted for one video sample period. If rx_frame_en is Low, this output remains High until the framer is allowed to realign to the new TRS alignment (by the assertion of rx_frame_en during the occurrence of a TRS).
rx_line_a	Out	11	The current line number captured from the LN words of the Y data stream of the SDI input signal is output on this port. This output is valid in HD-SDI and 3G-SDI modes, but not in SD-SDI mode. In 3G-SDI level B mode, the output value is the line number from the Y data stream of link A or HD-SDI signal 1. For any case where the interface line number is not the same as the picture line number, such as for 1080p 60 Hz carried on 3G-SDI level B or dual link HD-SDI, the output value on this port is the interface line number, not the picture line number.
rx_a_vpid	Out	32	All four user data bytes of the SMPTE ST 352 [Ref 7] payload ID packet from data stream 1 are output on this port in this format: MS byte to LS byte: byte4, byte3, byte2, byte1. This output port is valid only when rx_a_vpid_valid is High. This port is potentially valid in any SDI mode, but only if there are ST 352 packets embedded in the SDI signal. In 3G-SDI level A mode, the output data is the ST 352 data bytes captured from data stream 1 (luma). In 3G-SDI level B mode, the output data is the ST 352 data bytes captured from data stream 1 of link A (dual link streams,) or HD-SDI signal 1 (dual HD-SDI signals).
rx_a_vpid_valid	Out	1	This output is High when rx_a_vpid is valid.
rx_b_vpid	Out	32	All four user data bytes of the SMPTE ST 352 [Ref 7] payload ID packet from data stream 2 are output on this port in this format: MS byte to LS byte: byte4, byte3, byte2, byte1. This output is valid only in 3G-SDI mode and only when rx_b_vpid_valid is High. In 3G-SDI level A mode, the output data is the ST 352 data bytes captured from data stream 2 (chroma). In 3G-SDI level B mode, the output data is the ST 352 data bytes captured from data stream 1 of link B (dual link streams) or HD-SDI signal 2 (dual HD-SDI signals).
rx_b_vpid_valid	Out	1	This output is High when rx_b_vpid is valid.

Table 1: **SDI Wrapper Port List *(Cont'd)***

Port Name	I/O	Width	Description
rx_crc_err_a	Out	1	This output is asserted High when a cyclic redundancy check (CRC) error is detected on the previous video line. For 3G-SDI level B mode, this output indicates CRC errors on data stream 1 only. There is a second output called rx_crc_err_b that indicates CRC errors on data stream 2 for 3G-SDI level B mode. Neither CRC error output is valid in SD-SDI mode. The CRC error outputs are asserted High for one video line time when a CRC error has been detected on the previous video line. There is a six or seven video sample period latency, depending on the SDI mode, from the video sample in which the rx_eav signal is asserted until the rx_crc_err_a signal changes values.
rx_ds1a	Out	10	The recovered SDI data stream 1 is output on this port. The contents of this data stream are dependent on the SDI mode: • SD-SDI: Multiplexed Y/C_B/C_R components • HD-SDI: Y component • 3G-SDI level A: Data stream 1 • 3G-SDI level B-DL: Data stream 1 of link A • 3G-SDI level B-DS: Y component of HD-SDI signal 1
rx_ds2a	Out	10	The recovered SDI data stream 2 is output on this port. The contents of this data stream are dependent on the SDI mode: • SD-SDI: Not used • HD-SDI: Interleaved C_B and C_R components • 3G-SDI level A: Data stream 2 • 3G-SDI level B-DL: Data stream 2 of link A • 3G-SDI level B-DS: Interleaved C_B and C_R components of HD-SDI signal 1
rx_eav	Out	1	This output is asserted High for one video sample period when the XYZ word of an EAV is present on the data stream output ports.
rx_sav	Out	1	This output is asserted High for one video sample period when the XYZ word of an SAV is present on the data stream output ports.
rx_trs	Out	1	This output is asserted High for four consecutive video sample periods as all four words of an EAV or SAV, starting with the `3FF` word and continuing through the XYZ word, are output on the data stream ports.
rx_line_b	Out	11	This output port is only valid in 3G-SDI level B mode, and outputs the line number for the Y data stream of link B or HD-SDI signal 2. For any case where the interface line number is not the same as the picture line number, the line number output on this port is the interface line number, not the picture line number.
rx_dout_rdy_3G	Out	1	In 3G-SDI level B mode, the output data rate is 74.25 MHz, but the rx_usrclk frequency is 148.5 MHz. The rx_dout_rdy_3G output is asserted every other cycle of rx_usrclk in 3G-SDI level B mode. When this output is High, the data stream and video timing outputs are valid. This output is always High in all other SDI modes, allowing it to be used as a clock enable to downstream modules.

Table 1: **SDI Wrapper Port List *(Cont'd)***

Port Name	I/O	Width	Description
rx_crc_err_b	Out	1	This is the CRC error indicator valid only in 3G-SDI level B mode. It indicates that a CRC error was detected on link B for 3G-SDI B-DL signals and HD-SDI signal 2 for 3G-SDI level B-DS signals. This output has the same timing as the rx_crc_err_a signal.
rx_ds1b	Out	10	This output is only valid in 3G-SDI level B mode. The data stream output on this port is: • 3G-SDI level B-DL: Data stream 1 of link B • 3G-SDI level B-DS: Y component of HD-SDI signal 2
rx_ds2b	Out	10	This output is only valid in 3G-SDI level B mode. The data stream output on this port is: • 3G-SDI level B-DL: Data stream 2 of link B • 3G-SDI level B-DS: Interleaved C_B and C_R components of HD-SDI signal 2
rx_edh_errcnt_en	In	16	This input controls which EDH error conditions increment the EDH error counter. See Table 6 for more details.
rx_edh_clr_errcnt	In	1	When High, this input clears the EDH error counter. The EDH error counter is cleared on the rising edge of rx_usrclk only if rx_edh_clr_errcnt and rx_ce_sd are both High.
rx_edh_ap	Out	1	This output is asserted High when the active picture (AP) CRC calculated for the previous field does not match the AP CRC value in the EDH packet.
rx_edh_ff	Out	1	This output is asserted High when the full field (FF) CRC calculated for the previous field does not match the FF CRC value in the EDH packet.
rx_edh_anc	Out	1	This output is asserted High when an ancillary data packet checksum error is detected.
rx_edh_ap_flags	Out	5	The active picture error flag bits from the most recently received EDH packet are output on this port. See Table 7 for more information.
rx_edh_ff_flags	Out	5	The full field error flag bits from the most recently received EDH packet are output on this port. See Table 7 for more information.
rx_edh_anc_flags	Out	5	The ancillary error flag bits from the most recently received EDH packet are output on this port. See Table 7 for more information.
rx_edh_packet_flags	Out	4	This port outputs four error flags related to the most recently received EDH packet. See Table 8 for more information.
rx_edh_errcnt	Out	16	This is the SD-SDI EDH error counter. It increments once each field when any of the error conditions enabled by the rx_edh_err_en port occur.
rx_pllrange	In	1	This input specifies the operating range of the PLL that is supplying the reference clock to the GTX RX. For the CPLL, this must always be 0. For the QPLL, this must be 0 for the low QPLL operating range and 1 for the high QPLL operating range. The low QPLL operating range is usually used for SDI applications.

Table 1: **SDI Wrapper Port List *(Cont'd)***

Port Name	I/O	Width	Description
Transmit Ports			
tx_rst	In	1	This is a synchronous reset input. It resets the transmit section when High. To fully reset the transmitter, both tx_ce and tx_din_rdy must be High when tx_rst is High.
tx_usrclk	In	1	This input clocks the SDI transmitter datapath. This input must be connected to the same clock source that is driving the TXUSRCLK port of the GTX wrapper. It must have a frequency of 74.25 MHz or 74.25/1.001 MHz for HD-SDI and 148.5 MHz or 148.5/1.001 MHz for 3G-SDI and SD-SDI. The combination of the tx_usrclk frequency and tx_ce must result in a 27 MHz data rate in SD-SDI mode. All inputs and outputs of the SDI wrapper prefixed with ***tx***_ are synchronous with this clock.
tx_ce	In	3	The combination of the tx_usrclk frequency and tx_ce must clock the SDI core's transmitter datapath at the word rate (not necessarily the video sample rate) of the current SDI mode: 148.5 MHz or 148.5/1.001 MHz in 3G-SDI mode, 74.25 MHz or 74.25/1.001 MHz in HD-SDI mode, and 27 MHz in SD-SDI mode. The tx_ce must always be High in HD-SDI and 3G-SDI modes. In SD-SDI mode, tx_ce must be asserted at a 27 MHz rate with a mandatory 5/6/5/6 clock cycle cadence. Three identical copies of the clock enable signal must be provided on the three bits of this port. Three input bits are provided to make it easier to meet timing. If these three inputs are all driven by the same flip-flop, the loading on the single clock enable signal might be too high to meet timing. In those cases, duplicate copies of the clock enable signal can be created by multiple flip-flops each driving a different bit of the tx_ce input port.
tx_din_rdy	In	1	In SD-SDI, HD-SDI, and level A 3G-SDI modes, this input must be kept High at all times. In level B 3G-SDI mode, this input must be asserted every other clock cycle.
tx_mode	In	2	This input port is used to select the SDI transmitter's mode: • `00` = HD-SDI (including dual link HD-SDI) • `01` = SD-SDI • `10` = 3G-SDI • `11` = Invalid
tx_level_b_3G	In	1	In 3G-SDI mode, this input determines whether the module is configured for level A (Low) or level B (High).
tx_m	In	1	This port is used to select which PLL clock is used by the GTX TX. This input causes the SDI wrapper's gtx_txsysclksel output port to change in order to change the GTX TX PLL clock select MUX. By convention, tx_m = Low selects the 1/1.000 bit rates and tx_m = High selects the 1/1.001 bit rates. See Dynamically Switching the TX Clock Source, page 12 for more details.

Table 1: **SDI Wrapper Port List *(Cont'd)***

Port Name	I/O	Width	Description
tx_insert_crc	In	1	When this input is High, the SDI TX generates and inserts CRC values on each video line in HD-SDI and 3G-SDI modes. When this input is Low, CRC values are not generated and inserted. This input is ignored in SD-SDI mode. CRC values are required by both the HD-SDI and 3G-SDI standards. If the data streams entering the SDI TX input ports do not have CRC values, this input should be asserted High. If the data streams entering the SDI TX input ports already have CRC values, the existing CRC values are replaced by newly calculated CRC values if tx_insert_crc is asserted High.
tx_insert_ln	In	1	When this input is High, the transmitter inserts line number words after the EAV in each video line. The line number must be supplied on the tx_line_a and tx_line_b input ports. This input is ignored in SD-SDI mode. Line numbers are required by both the HD-SDI and 3G-SDI standards. If the data streams entering the SDI TX input ports do not have line number words already embedded, this input should be asserted High, and valid line numbers must be supplied on the tx_line_a and tx_line_b ports. If the data streams entering the SDI TX input port already have line numbers embedded, those line numbers are overwritten if tx_insert_ln is High.
tx_insert_edh	In	1	When this input is High, the transmitter generates and inserts EDH packets in every field in SD-SDI mode. When this input is Low, EDH packets are not inserted. This input is ignored in HD-SDI and 3G-SDI modes. EDH packets are optional but commonly used in SD-SDI mode and are never used in HD-SDI and 3G-SDI modes. If the SD-SDI data stream entering the SDI TX already has an EDH packet embedded, it is overwritten with a new packet if tx_insert_edh is High.
tx_insert_vpid	In	1	When this input is High, SMPTE ST 352 [Ref 7] packets are inserted into the data streams, otherwise the packets are not inserted. ST 352 packets are mandatory in 3G-SDI and dual link HD-SDI modes and optional in HD-SDI and SD-SDI modes.
tx_overwrite_vpid	In	1	If this input is High and the tx_insert_vpid port is High, SMPTE ST 352 [Ref 7] packets already present in the data streams are overwritten with new ST 352 packets. If this input is Low, existing ST 352 packets are not overwritten. When transporting ST 372 [Ref 12] dual-link data streams on a 3G-SDI level B interface, existing ST 352 packets in the data streams must be updated to indicate that the interface is 3G-SDI rather than HD-SDI mode.
tx_video_a_y_in	In	10	This is the SDI data stream A Y input to the SDI TX. The data on this port depends on the SDI mode: • SD-SDI: Multiplexed Y/C data stream • HD-SDI: Y component • 3G-SDI level A: Data stream 1 • Dual-link HD-SDI or 3G-SDI level B-DL: Data stream 1 of link A • 3G-SDI level B-DS: Y component of HD-SDI signal 1

Table 1: **SDI Wrapper Port List *(Cont'd)***

Port Name	I/O	Width	Description
tx_video_a_c_in	In	10	This is the SDI data stream A C input to the SDI TX. The data on this port depends on the SDI mode: • SD-SDI: Unused • HD-SDI: Interleaved C_B and C_R components • 3G-SDI level A: Data stream 2 • Dual-link HD-SDI or 3G-SDI level B-DL: Data stream 2 of link A • 3G-SDI level B-DS: Interleaved C_B and C_R components of HD-SDI signal 1
tx_video_ b_y_in	In	10	This is the SDI data stream B Y input to the SDI TX. The data stream on this port depends on the SDI mode: • Dual-link HD-SDI or 3G-SDI level B-DL: Data stream 1 of link B • 3G-SDI level B-DS: Y component of HD-SDI signal 2 For other SDI modes, this input port is unused.
tx_video_b_c_in	In	10	This is the SDI data stream B C input to the SDI TX. The data stream on this port depends on the SDI mode: • Dual-link HD-SDI or 3G-SDI level B-DL: Data stream 2 of link B • 3G-SDI level B-DS: Interleaved C_B and C_R components of HD-SDI signal 2 For other SDI modes, this input port is unused.
tx_line_a	In	11	The current line number must be provided to the module through this port if either ST 352 [Ref 7] VPID packet insertion is enabled (tx_insert_vpid = High) or if HD-SDI and 3G-SDI line number insertion is enabled (tx_insert_ln = High). SD-SDI only uses 10-bit line numbers, so bit 10 of this port must be 0 in SD-SDI mode if ST 352 VPID packet insertion is enabled in SD-SDI mode. Line number insertion is never done in SD-SDI mode so this input port is only used for ST 352 VPID packet insertion in SD-SDI mode. The line number must be valid at least one clock cycle before the start of the HANC space (by the XYZ word of the EAV) and must remain valid during the entire HANC interval. This input is the only line number input used for SD-SDI, HD-SDI, and 3G-SDI level A modes. For 3G-SDI level B mode, a second line number input port, tx_line_b, is also provided. For video formats where the picture line number is different from the transport line number, the value supplied on this port must be the transport line number.
tx_line_b	In	11	This is the second line number input port and is used only for 3G-SDI level B mode. This additional line number port allows the two separate HD-SDI signals to be vertically unsynchronized in level B-DS mode. When using either 3G-SDI level B-DL or B-DS, this port must be given a valid line number input. In 3G-SDI level B-DL mode, the value on this input port must be the same as the value on the tx_line_a port. This input port has the same timing and other requirements described for tx_line_a.

Table 1: **SDI Wrapper Port List *(Cont'd)***

Port Name	I/O	Width	Description
tx_vpid_byte1	In	8	The value on this port is inserted as the first user data word of the ST 352 packet [Ref 7]. It must be valid during the entire HANC interval of the lines that are to contain the ST 352 packet if ST 352 packets are being inserted or overwritten.
tx_vpid_byte2	In	8	The value on this port is inserted as the second user data word of the ST 352 packet [Ref 7]. It must be valid during the entire HANC interval of the lines that are to contain the ST 352 packet if ST 352 packets are being inserted or overwritten.
tx_vpid_byte3	In	8	The value on this port is inserted as the third user data word of the ST 352 packet [Ref 7]. It must be valid during the entire HANC interval of the lines that are to contain the ST 352 packet if ST 352 packets are being inserted or overwritten.
tx_vpid_byte4a	In	8	The value on this port is inserted as the fourth user data word of the ST 352 packet. [Ref 7]. This word is used for the ST 352 packets inserted into SD-SDI, HD-SDI, and 3G-SDI level A data streams. For 3G-SDI level B and dual-link HD-SDI modes, this value is used for the ST 352 packet inserted into data stream 1 of link A only. This input must be valid during the entire HANC interval of the lines that are to contain the ST 352 packet if ST 352 packets are being inserted or overwritten. Separate values are allowed for byte 4 for link A and link B because this byte contains the link ID bits which must be different on link A than on link B.
tx_vpid_byte4b	In	8	The value on this port is inserted as the fourth user data word of ST 352 packets [Ref 7] inserted in the data stream 1 of link B for 3G-SDI level B and dual-link HD-SDI modes only. This input value is not used for SD-SDI, HD-SDI, or 3G-SDI level A modes. This input must be valid during the entire HANC interval of the lines that are to contain the ST 352 packet if ST 352 packets are being inserted or overwritten.
tx_vpid_line_f1	In	11	The ST 352 packet [Ref 7] is inserted in the HANC space of the line number specified by this input port. For interlaced video, this input port specifies a line number in field 1. For progressive video, this specifies the only line in the frame where the packet is inserted. The input value must be valid during the entire HANC interval. If tx_insert_vpid is Low, this input is ignored.
tx_vpid_line_f2	In	11	For interlaced video, an ST 352 packet [Ref 7] is inserted on the line number in field 2 indicated by this value. For progressive video, insertion of ST 352 packets on the line specified by this input port must be disabled by holding the tx_vpid_line_f2_en port Low. The input value must be valid during the entire HANC interval. This input is ignored if either tx_insert_vpid or tx_vpid_line_f2_en are Low.

Table 1: **SDI Wrapper Port List *(Cont'd)***

Port Name	I/O	Width	Description
tx_vpid_line_f2_en	In	1	This input controls whether or not ST 352 packets [Ref 7] are inserted on the line indicated by tx_vpid_line_f2. For interlaced video, this input must be High. For progressive video, this input must be Low. For progressive video transported on an interlaced transport such as 1080p 60 Hz transported by either 3G-SDI level B-DL or dual-link HD-SDI, ST 352 packets [Ref 7] must be inserted into both fields of the interlaced transport, so this input must be High in these cases. This input must be valid during the entire HANC interval. This input is ignored if tx_insert_vpid is Low.
tx_ds1a_out	Out	10	This is the link A data stream 1 output. The data stream output on this port comes from the ST 352 packet insertion module [Ref 7]. If the application needs to insert ancillary data packets, they should be inserted into the data stream output on this port so that ST 352 packets have already been inserted into the data streams. The resulting data stream after ancillary data insertion by the application should then be supplied to the tx_ds1a_in port. The data on this port depends on the SDI mode: • SD-SDI: Interleaved Y/C data stream • HD-SDI: Y component • 3G-SDI level A: Data stream 1 • Dual-link HD-SDI or 3G-SDI level B-DL: Data stream 1 of link A • 3G-SDI level B-DS: Y component of HD-SDI signal 1
tx_ds2a_out	Out	10	This is the link A data stream 2 output. The data stream output on this port comes from the ST 352 packet insertion module [Ref 7]. If the application needs to insert ancillary data packets, they should be inserted into the data stream output on this port so that ST 352 packets have already been inserted into the data streams. The resulting data stream after ancillary data insertion by the application should then be supplied to the tx_ds2a_in port. The data on this port depends on the SDI mode: • HD-SDI: Interleaved CB/CR component • Dual-link HD-SDI or 3G-SDI level B-DL: Data stream 2 of link A • 3G-SDI level B-DS: Interleaved CB/CR component data stream of HD-SDI signal 1
tx_ds1b_out	Out	10	This is the link B data stream 1 output. The data stream output on this port comes from the ST 352 packet insertion module [Ref 7]. If the application needs to insert ancillary data packets, they should be inserted into the data stream output on this port so that ST 352 packets have already been inserted into the data streams. The resulting data stream after ancillary data insertion by the application should then be supplied to the tx_ds1b_in port. The data on this port depends on the SDI mode: • Dual-link HD-SDI or 3G-SDI level B-DL: Data stream 1 of link B • 3G-SDI level B-DS: Y component of HD-SDI signal 2 For other SDI modes, this output port is unused.

Table 1: **SDI Wrapper Port List *(Cont'd)***

Port Name	I/O	Width	Description
tx_ds2b_out	Out	10	This is the link B data stream 2 output. The data stream output on this port comes from the ST 352 packet insertion module [Ref 7]. If the application needs to insert ancillary data packets, they should be inserted into the data stream output on this port so that ST 352 packets have already been inserted into the data streams. The resulting data stream after ancillary data insertion by the application should then be supplied to the tx_ds2b_in port. • Dual-link HD-SDI or 3G-SDI level B carrying dual-link HD-SDI: Data stream 2 of link B • 3G-SDI level B carrying dual HD-SDI signals: Interleaved CB/CR component of HD-SDI signal 2 For other SDI modes, this input port is unused.
tx_use_dsin	In	1	This input controls the source of the data streams sent by the SDI TX. When this input is High, the sources of the transmitted data streams are the tx_ds1a_in, tx_ds2a_in, tx_ds1b_in, and tx_ds2b_in input ports. When this input is Low, the source of the transmitted data streams are internal to the core, coming directly from the ST 352 packet inserter [Ref 7]. When the application needs to do ancillary data insertion, the tx_use_dsin port is set High to allow the application to modify the data streams and provide the modified data streams to the transmitter on the tx_dsxx_in ports. When no ancillary data insertion is required, the tx_use_dsin input is set Low and the tx_dsxx_in ports are ignored.
tx_ds1a_in	In	10	This is the link A data stream 1 input. This port is ignored if tx_use_dsin is Low. If tx_use_dsin is High, this port must supply a data stream to be transmitted. The data stream supplied input to this port depends on the SDI mode: • SD-SDI: Interleaved Y/C data stream • HD-SDI: Y component • 3G-SDI level A: Data stream 1 • Dual-link HD-SDI or 3G-SDI level B-DL: Data stream 1 of link A • 3G-SDI level B-DS: Y component of HD-SDI signal 1
tx_ds2a_in	In	10	This is the link A data stream 2 input. This port is ignored if tx_use_dsin is Low. If tx_use_dsin is High, this port must supply a data stream to be transmitted. The data stream input to this port depends on the SDI mode: • HD-SDI: Interleaved CB/CR component • Dual-link HD-SDI or 3G-SDI level B-DL: Data stream 2 of link A 3G-SDI level B-DS: Interleaved CB/CR component data stream of HD-SDI signal 1
tx_ds1b_in	In	10	This is the link B data stream 1 input. This port is ignored if tx_use_dsin is Low. If tx_use_dsin is High, this port must supply a data stream to be transmitted. The data stream input to this port depends on the SDI mode: • Dual-link HD-SDI or 3G-SDI level B-DL: Data stream 1 of link B • 3G-SDI level B-DS: Y component of HD-SDI signal 2 For other SDI modes, this input port is unused.

Table 1: **SDI Wrapper Port List *(Cont'd)***

Port Name	I/O	Width	Description
tx_ds2b_in	In	10	This is the link B data stream 2 input. This port is ignored if tx_use_dsin is Low. If tx_use_dsin is High, this port must supply a data stream to be transmitted. The data stream input to this port depends on the SDI mode: • Dual-link HD-SDI or 3G-SDI level B carrying dual-link HD-SDI: Data stream 2 of link B • 3G-SDI level B carrying dual HD-SDI signals: Interleaved CB/CR component of HD-SDI signal 2 For other SDI modes, this input port is unused.
tx_ce_align_err	Out	1	This output indicates problems with the 5/6/5/6 clock cycle cadence on the tx_ce clock enable inputs in SD-SDI mode. In SD-SDI mode, the tx_ce signals must follow a regular 5/6/5/6 clock cycle cadence. If they do not, the SD-SDI bit stream is formed incorrectly. The tx_ce_align_err goes High if the cadence is incorrect. This port is only valid in SD-SDI mode.
tx_slew	Out	1	This output is designed to control the slew rate signal of the external SDI cable equalizer. It is High when the TX mode is SD-SDI, otherwise it is Low.
tx_pllrange	In	1	This input specifies the operating range of the PLL that is supplying the reference clock to the GTX TX. For the CPLL, this must always be 0. For the QPLL, this must be 0 for the low QPLL operating range and 1 for the high QPLL operating range. The low QPLL operating range is usually used for SDI applications.
Ports that Connect to the GTX RX			
gtx_rxdata	In	20	Connect this port to the RXDATA port of the GTX transceiver.
gtx_rxpllreset	In	1	Connect this port to the signal connected to either QPLLRESET or CPLLRESET of the GTX transceiver.
gtx_rxplllock	In	1	Connect this port to either QPLLLOCK or CPLLLOCK of the GTX wrapper.
gtx_rxresetdone	In	1	Connect this port to the RXRESETDONE port of the GTX transceiver.
gtx_gtrxreset	Out	1	Connect this port to the GTRXRESET port of the GTX transceiver.
gtx_rxuserrdy	Out	1	Connect this port to the RXUSERRDY port of the GTX transceiver.
gtx_rxrate	Out	3	Connect this port to the RXRATE port of the GTX transceiver.
gtx_rxcdrhold	Out	1	Connect this port to the RXCDRHOLD port of the GTX transceiver.
gtx_drpclk	In	1	Connect this port to the clock that is driving the DRPCLK port of the GTX transceiver.
gtx_drprdy	In	1	Connect this port to the DRPRDY port of the GTX transceiver.
gtx_drpaddr	Out	10	Connect this port to the DRPADDR port of the GTX transceiver.

Table 1: **SDI Wrapper Port List *(Cont'd)***

Port Name	I/O	Width	Description
gtx_drpdi	Out	16	Connect this port to the DRPDI port of the GTX transceiver.
gtx_drpen	Out	1	Connect this port to the DRPEN port of the GTX transceiver.
gtx_drpwe	Out	1	Connect this port to the DRPWE port of the GTX transceiver.
Ports that Connect to the GTX TX			
gtx_txdata	Out	20	Connect this port to the TXDATA port of the GTX transceiver.
gtx_txpllreset	In	1	Connect this port to the signal driving the reset of the PLL supplying the clock to the GTX TX, either QPLLRESET or CPLLRESET of the GTX transceiver. If the GTX TX can dynamically switch between the QPLL and the CPLL, the signal driving this port must be asserted High when either of the PLLs are reset (e.g., logical OR of QPLLRESET and CPLLRESET).
gtx_txplllock	In	1	Connect this port to the PLL lock signal of the PLL supplying the clock to the GTX TX, either QPLLLOCK or CPLLLOCK of the GTX wrapper. If the GTX TX can dynamically switch between the QPLL and the CPLL, the signal driving this port must only be asserted High when both the QPLL and the CPLL are locked (e.g., logical AND of QPLLLOCK and CPLLLOCK).
gtx_gttxreset	Out	1	Connect this port to the GTTXRESET port of the GTX transceiver.
gtx_txuserrdy	Out	1	Connect this port to the TXUSERRDY port of the GTX transceiver.
gtx_txpmareset	Out	1	Connect this port to the TXPMARESET port of the GTX transceiver.
gtx_txrate	Out	3	Connect this port to the TXRATE port of the GTX transceiver.
gtx_txsysclksel	Out	2	If the clock source of the GTX TX needs to be dynamically switched between the QPLL and the CPLL, connect this port to the TXSYSCLKSEL port of the GTX transceiver.

Table 2 lists the parameters that can be applied to the Verilog version of the SDI wrapper. Table 3 lists the generics that can be applied to the VHDL version of the SDI wrapper.

Table 2: **SDI Wrapper Verilog Parameter List**

Name	Type	Default	Description
FXDCLK_FREQ	Integer	27000000	Specifies the frequency, in Hz, of the fixed-frequency clock on the clk port of the GTX wrapper. The nominal frequency of this clock must be correctly specified so that the portions of the control module that depend on this clock for timing purposes functions correctly.
TXPMARESETDLY_MSB	integer	15	Specifies the MSB of the TXPMARESET delay counter. See Transceiver Resets, page 8 for more details.

Table 2: **SDI Wrapper Verilog Parameter List *(Cont'd)***

Name	Type	Default	Description
PLLLOCKDLY	integer	4	Specifies the width of the PLL lock delay counter. See Transceiver Resets, page 8 for more details.
TXSYSCLKSEL_M_0	2-bit value	2'b11	Specifies the value output on the gtx_txsysclksel port when the tx_m is Low.
TXSYSCLKSEL_M_1	2-bit value	2'b00	Specifies the value output on the gtx_txsysclksel port when the tx_m is High.

Table 3: **SDI Wrapper VHDL Generic List**

Name	Type	Default	Description
FXDCLK_FREQ	integer	27000000	Specifies the frequency, in Hz, of the fixed-frequency clock on the clk port of the GTX wrapper. The nominal frequency of this clock must be correctly specified so that the portions of the control module that depend on this clock for timing purposes functions correctly.
TXPMARESETDLY_MSB	integer	15	Specifies the MSB of the TXPMARESET delay counter. See Transceiver Resets, page 8 for more details.
PLLLOCKDLY	integer	4	Specifies the width of the PLL lock delay counter. See Transceiver Resets, page 8 for more details.
TXSYSCLKSEL_M_0	std_logic_vector (1 down to 0)	11	Specifies the value output on the gtx_txsysclksel port when the tx_m is Low.
TXSYSCLKSEL_M_1	std_logic_vector (1 down to 0)	00	Specifies the value output on the gtx_txsysclksel port when the tx_m is High.

Video Transport Detector Ports

The RX section of the SDI core has an SDI transport format detector. This function examines the timing of the video transport in the SDI data streams and determines which video format is being received. The operation of this function is not dependent on the presence of ST 352 payload ID packets [Ref 7]. This function determines the transport format, not the picture format. Usually these are the same, but not always. For example, when 1080p 50 Hz video is transported on 3G-SDI level B-DL, the video transport is actually 1080i 50 Hz—the transport is interlaced, but the picture is progressive.

The rx_t_family output port provides a 4-bit code indicating the video format family of the transport in the SDI signal. The encoding of this output port is shown in Table 4. The transport detection unit also determines whether the SDI transport is interlaced or progressive and reports this on the rx_t_scan output port.

Table 4: **rx_t_family Encoding**

rx_t_family	Transport Video Format	Active Pixels
0000	SMPTE ST 274 [Ref 8]	1920 x 1080
0001	SMPTE ST 296 [Ref 9]	1280 x 720
0010	SMPTE 2048-2 [Ref 10]	2048 x 1080
0011	SMPTE 295 [Ref 11]	1920 x 1080
1000	NTSC	720 x 486
1001	PAL	720 x 576
1111	Unknown	
Others	Reserved	

The transport detector also determines the frame rate of the transport in the SDI signal. The rx_t_rate port indicates the frame rate of the transport signal as shown in Table 5. The encoding of the frame rate matches the encoding used in the picture rate field of SMPTE ST 352 video payload ID packets [Ref 7]. However, the rx_t_rate shows the transport frame rate, not the picture rate. Also, the rx_t_rate port value is always the frame rate, even for interlaced transports.

Table 5: **rx_t_rate Encoding**

rx_t_rate	Frame Rate
0000	None
0010	23.98 Hz
0011	24 Hz
0100	47.95 Hz
0101	25 Hz
0110	29.97 Hz
0111	30 Hz
1000	48 Hz
1001	50 Hz
1010	59.94 Hz
1011	60 Hz
Others	Reserved

It can take the transport format detector up to two video frames to identify the transport format after the SDI RX locks to the SDI signal.

SD-SDI RX EDH Processor

The SDI receiver can, optionally, include an EDH processor for detecting receiver errors in SD-SDI mode. The EDH processor does not update EDH packets in the SD-SDI data stream. It just reports any errors found and also captures the error flags from each EDH packet.

The EDH processor has a 16-bit counter that counts the number of fields with errors. The current error count is output on the rx_edh_errcnt port of the SDI wrapper. The counter is cleared by asserting rx_edh_clr_errcnt High. The user can specify which types of errors are counted by this counter using the rx_edh_errcnt_en port. This port has 16 unary bits that enable and disable 16 different error types. Any bit that is High enables the corresponding error to be counted by the error counter. Any bit that is Low disables the corresponding error. If

multiple errors occur in the same field, the EDH error counter only increments by one. Table 6 shows the encoding of the bits on the rx_edh_errcnt_en port.

The ANC error conditions are associated with errors in the *ancillary* data packets. The FF error conditions are associated with errors detected by the *full field* CRC. The AP error conditions are associated with errors detected by the *active picture* CRC. The EDH packet checksum error indicates a checksum error was found within the EDH packet itself.

Table 6: **rx_edh_errcnt_en Bits**

Bit Number	Error
0	ANC EDH error
1	ANC EDA error
2	ANC IDH error
3	ANC IDA error
4	ANC UES error
5	FF EDH error
6	FF EDA error
7	FF IDH error
8	FF IDA error
9	FF UES error
10	AP EDH error
11	AP EDA error
12	AP IDH error
13	AP IDA error
14	AP UES error
15	EDH packet checksum error

Each ANC, FF, and AP error condition set has five individual error flags. All flags are asserted High to indicate an error condition. For a complete description of the EDH, EDA, IDH, IDA, and UES error flags in the EDH packet, refer to the SMPTE *Error Detection Checkwords and Status Flags for Use in Bit-Serial Digital Interfaces for Television* document (RP 165) [Ref 13].

- EDH error: This error condition occurs when the EDH processor detects a CRC error (checksum error for ANC packets) in a field. For example, the FF EDH error flag indicates an error was detected by the full field CRC.
- EDA error: This error condition occurs when the EDA or EDH flags of the received EDH packet are asserted.
- IDH error: This error condition is not supported by the RX EDH processor.
- IDA error: This error condition occurs when the IDA or IDH flags of the received EDH packet are asserted.
- UES error: This error condition occurs when the UES flag in the received EDH packet is asserted.

In addition to being counted by the error counter, if enabled, any detected ANC EDH, AP EDH, and FF EDH errors are also indicated by assertion of the rx_edh_anc, rx_edh_ap, and rx_edh_ff ports, respectively. Thus, the rx_edh_anc port is asserted whenever a checksum error is detected in an ancillary data packet. The rx_edh_ap port is asserted when the calculated active picture CRC does not match the AP CRC in the EDH packet. The rx_edh_ff

port is asserted when the calculated full field CRC does not match the FF CRC in the EDH packet.

The RX EDH processor also outputs the ANC, AP, and FF error flags from the EDH packet on the rx_edh_anc_flags, rx_edh_ap_flags, and rx_edh_ff_flags ports, respectively. These output ports are exact copies of the flags found in the last received EDH packet. Thus, they differ from the detected errors used to increment the error counter and output on the rx_edh_anc, rx_edh_ap, and rx_edh_ff ports. For example, the EDH flag (bit 0) of the rx_edh_ap_flags port indicates that the AP EDH flag was set in the last received EDH packet. However, the rx_edh_ap port indicates that the active picture CRC calculated locally by the EDH processor does not match the AP CRC value in the EDH packet. The rx_edh_anc_flags, rx_edh_ap_flags, and rx_edh_ff_flags ports are each five bits wide. The encoding of all three ports are identical and shown in Table 7.

Table 7: **Encoding of rx_edh_anc_flags, rx_edh_ap_flags, and rx_edh_ff_flags Ports**

Bit Number	Flag
0	EDH
1	EDA
2	IDH
3	IDA
4	UES

The RX EDH processor also produces four error flags related to the format and contents of the EDH packet itself. These error flags are output on the rx_edh_packet_flags port. The encoding of this port is shown in Table 8.

Table 8: **Encoding of rx_edh_packet_flags Port**

Bit Number	Flag
0	EDH packet is missing
1	Parity error in user data words of EDH packet
2	Checksum error in EDH packet
3	Format error in EDH packet, such as invalid data count

SDI Timing Constraints

The timing constraint requirements are quite simple for the SDI wrapper and SMPTE SDI core. Only the periods of the clocks need to be constrained. These are the clocks applied to the clk, rx_usrclk, tx_usrclk, and gtx_drpclk ports of the SDI wrapper. See the constraints files of the example SDI applications provided with this application note for examples of setting these constraints.

Example SDI Demonstrations

Two example SDI demonstration applications are included with this application note. The source code for these demonstrations is provided in Verilog only. Instructions for building these demonstrations using either ISE or Vivado are included in the `readme.txt` file located in the `xapp1092.zip` file along with the source code. Pre-generated FPGA configuration files are also provided for both demonstrations that can be loaded onto a Zynq-7000 SoC ZC706 evaluation board. These demonstrations require an inrevium TB-FMCH-3GSDI2A FMC, which provides the SDI cable drivers and SDI cable equalizers connected to the HPC FMC1 connector of the ZC706 board. The inrevium FMC also provides SDI-specific clock sources that are used as reference clocks for the GTX transceivers.

Quad SDI Demonstration

This demonstration application includes four SDI RX interfaces and four SDI TX interfaces that are all independent.

Each SDI TX is driven by a video pattern generator. The SDI mode, video format, and video pattern of each SDI TX can independently be selected using virtual I/O (VIO) windows in the ChipScope™ Pro analyzer tool.

The status of each SDI RX can be monitored using a VIO window in the ChipScope Pro analyzer. The video data received by each SDI RX can be captured and viewed using an integrated logic analyzer (ILA) window in the ChipScope Pro analyzer.

The inrevium SDI FMC has six connectors for the SDI interfaces. The connectors labeled CH0-RX and CH0-TX are separate connectors for GTX0 in Quad 109 of the Zynq-7000 SoC. The CH1-RX and CH1-TX connectors are separate connectors for GTX1 in the same Quad. Because both of these transceivers have separate connectors for their RX and TX sides, they can both receive and transmit at the same time. The other two transceivers in Quad 109 each only have a single connector on the inrevium SDI FMC. These two connectors, labeled CH2 (for GTX2) and CH3 (for GTX3), are bidirectional. They can each be configured to receive or transmit. The demonstration has a control for each of them, available in a VIO window of the ChipScope Pro analyzer, to specify whether they are configured for receive or transmit. Thus, it is possible to run the demonstration with four SDI receivers or four SDI transmitters, but not all at the same time. The demonstration can be configured for four SDI receivers and two SDI transmitters, two SDI receivers and four SDI transmitters, or three SDI receivers and three SDI transmitters. This selection can be changed dynamically using the ChipScope Pro analyzer.

Figure 16 is a block diagram of the demonstration showing just one of the SDI channels. All four SDI channels are identical with the exception noted above that channels 2 and 3 have only a single SDI connector and a bidirectional SDI physical interface.

The inrevium SDI FMC has 148.5 and 148.3516 MHz oscillators, which this demonstration uses to supply the two GTX reference clock frequencies needed to support all SDI bit rates. The 148.5 MHz reference clock is also divided by two by the IBUFDS_GTE2 on its ODIV2 output to provide a 74.25 MHz global clock that is used as the DRPCLK for the GTX transceivers and as the fixed-frequency clock required by the control module in the SDI wrapper.

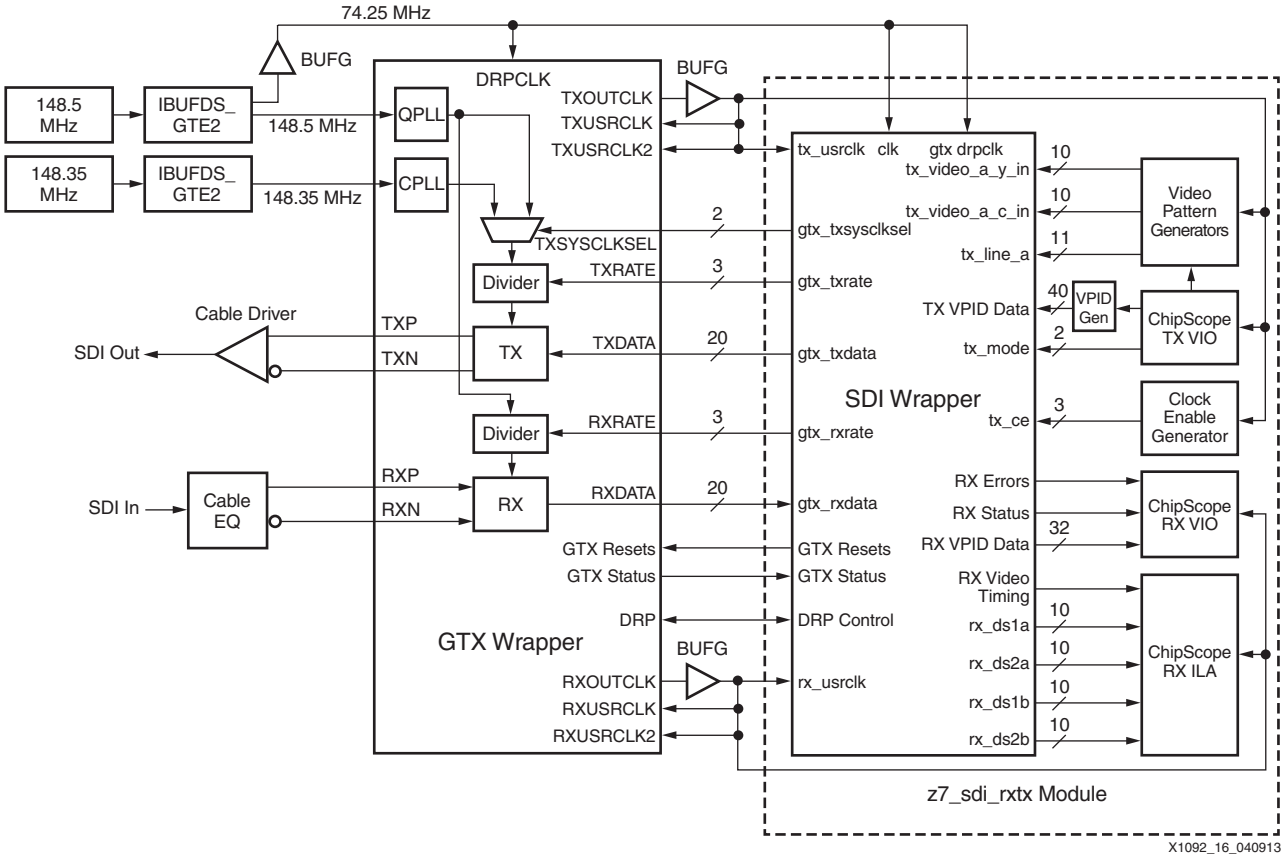


Figure 16: **Quad SDI Block Diagram**

To make it easier to replicate the SDI interface four times in this demonstration, the SDI wrapper, video pattern generators, TX clock enable generator, ChipScope analyzer VIO and ILA modules, and other miscellaneous logic are contained in a module called `z7_sdi_rxtx`. This module is instanced four times in the top-level module of the design.

The following are required to run the Quad SDI demonstration:

- Xilinx Zynq-7000 SoC ZC706 Evaluation Kit
- inrevium TB-FMCH-3GSDI2A SDI FMC
- DIN 1.0/2.3 to BNC converter cables (usually supplied with the TB-FMCH-3GSDI2A)
- SDI signal source
- SDI signal sink (waveform monitor or other device to view signal from SDI transmitters)
- PC with ChipScope Pro analyzer installed

The inrevium SDI FMC must be connected to the HPC FMC connector on the ZC706 board as shown in Figure 17.

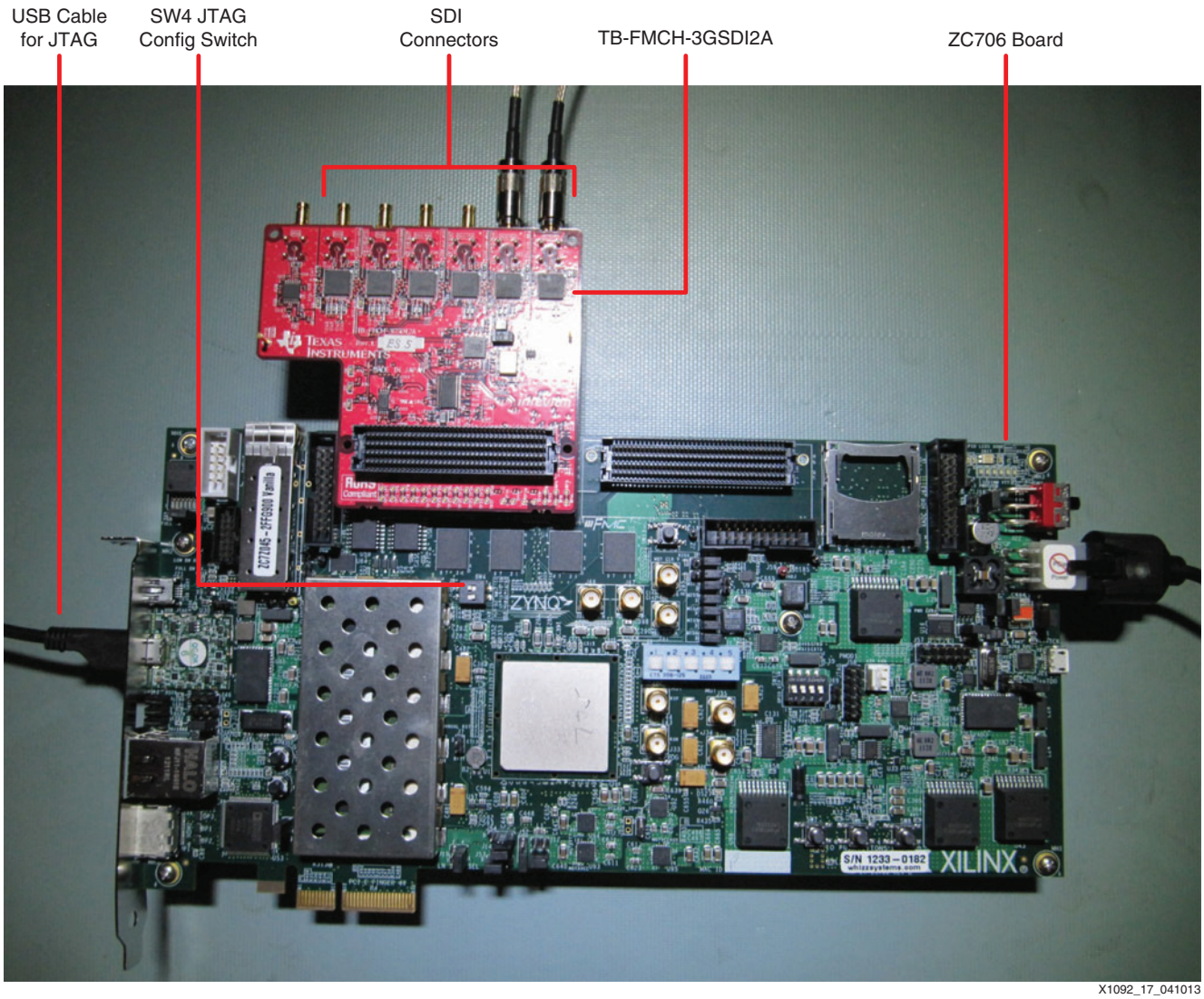


Figure 17: **ZC706 Board with TB-FMCH-3GSDI2A Board Connected**

ChipScope Pro analyzer is required to run this demonstration. It is used to control the SDI transmitters and to look at the status and received data from the SDI receivers. The ZC706 board must be connected to a PC with ChipScope Pro analyzer installed by the USB JTAG cable provided with the ZC706 board. SW4 on the ZC706 board must be correctly set to use the Digilent USB-to-JTAG interface (switch 1 = 0 and switch 2 = 1).

The file called `zc706_sdi_demo.bit` provided with this application note must be loaded into the PL portion of the Zynq-7000 SoC on the ZC706 board using the ChipScope Pro analyzer. After the BIT file is loaded into the PL, the `zc706_sdi_demo.cpj` ChipScope analyzer project file must be opened in ChipScope Pro analyzer. When the project is opened, it looks like Figure 18. There are eight VIO windows, one for each RX and TX in the application. Not visible in Figure 18 are four ILA waveform windows, one for each receiver in the demonstration.

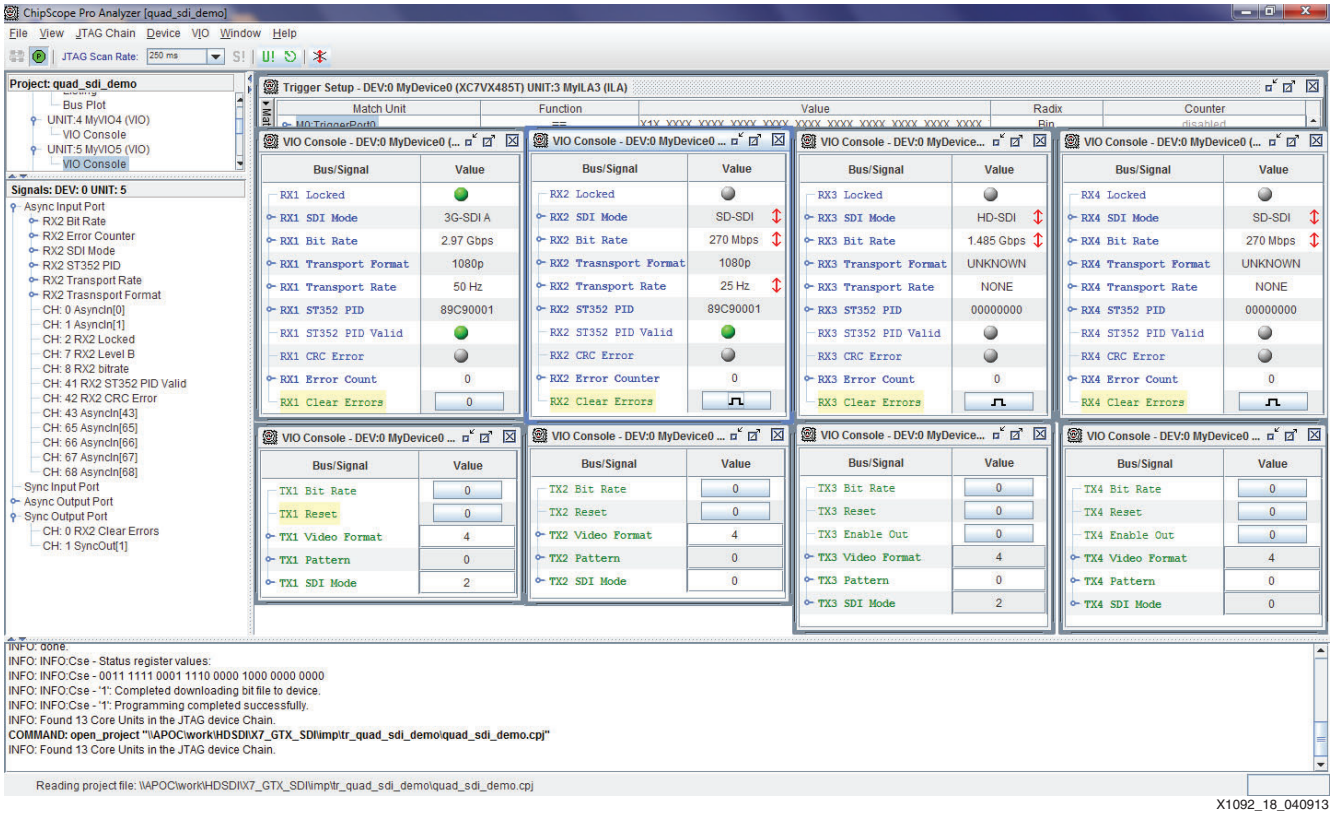


Figure 18: **ChipScope Pro Analyzer with Quad SDI Project Opened**

To observe the signal being generated by an SDI transmitter, an SDI waveform monitor or other SDI display device must be connected to the output of the SDI TX. The SDI connectors on the inrevium SDI FMC are not standard BNC cables, so adapter cables are required to go from these DIN 1.0/2.3 connectors to regular BNC connectors.

Each of the four transmitters in the demonstration has a VIO control window like the one shown in Figure 19. While the SDI connectors on the inrevium SDI FMC are labeled as CH0 through CH3, the ILA and VIO windows in the ChipScope Pro analyzer show the receivers and transmitters being numbered as RX1 through RX4 and TX1 through TX4. RX1 and TX1 correspond to the CH0-RX and CH0-TX connectors.

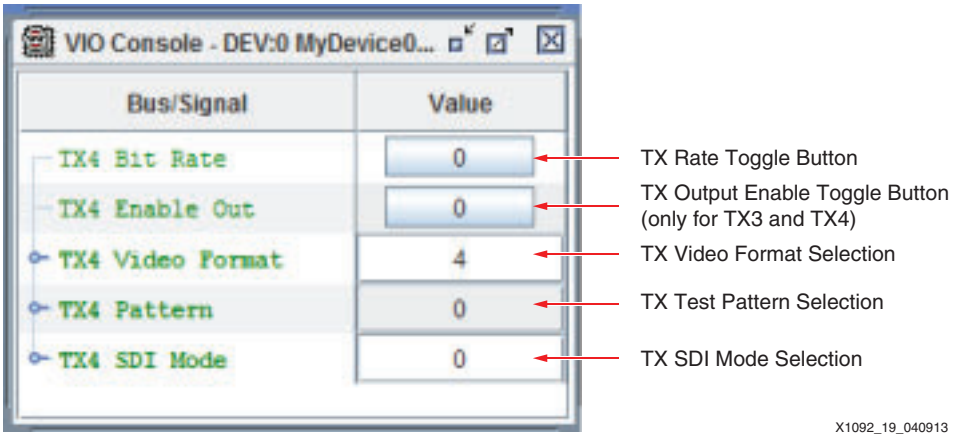


Figure 19: **Quad SDI Demonstration TX Control Window**

The TX Bit Rate toggle button, TX Video Format section field, and the TX SDI Mode selection field work together to set the format of the SDI signal generated by the SDI transmitter as shown in Table 9.

Table 9: **Quad SDI Demonstration TX Video Format Selection**

TX Video Format	HD-SDI (SDI Mode = 0)		3G-SDI (SDI Mode = 2)		SD-SDI (SDI Mode = 1)
	TX Bit Rate = 0	TX Bit Rate = 1	TX Bit Rate = 0	TX Bit Rate = 1	
0	720p 50 Hz	Not Valid	Not Valid	Not Valid	NTSC
1	1080pSF 24 Hz	1080pSF 23.98 Hz	Not Valid	Not Valid	PAL
2	1080i 60 Hz	1080i 59.94 Hz	Not Valid	Not Valid	NTSC
3	1080i 50 Hz	Not Valid	Not Valid	Not Valid	PAL
4	1080p 30 Hz	1080p 29.97 Hz	1080p 60 Hz	1080p 59.94 Hz	NTSC
5	1080p 25 Hz	Not Valid	1080p 50 Hz	Not Valid	PAL
6	1080p 24 Hz	1080p 23.98 Hz	Not Valid	Not Valid	NTSC
7	720p 60 Hz	720p 59.94 Hz	Not Valid	Not Valid	PAL

The TX Video Pattern value selects the video test pattern generated by the video pattern generator driving the SDI TX. In HD-SDI and 3G-SDI mode, three test patterns are available:

- 0 = SMPTE RP 219 color bars
- 1 and 3 = SDI pathological checkfield
- 2 = 75% color bars

In SD-SDI mode, two test patterns are available:

- 0 and 2 = SMPTE EG 1 color bars
- 1 and 3 = SDI pathological checkfield

TX3 and TX4 are connected to their respective connectors through bidirectional SDI interfaces. The VIO window for these two transmitters have an extra toggle button labeled **TX Enable Out**. When the **TX Enable Out** button of these transmitters is 0, the transmitter is disabled and the channel is running in receive mode. When the **TX Enable Out** button is 1, the transmitter is enabled. If the transmitter is enabled, the receiver of the same channel is still active and it receives the SDI signal being transmitted by the transmitter. However, the SDI cable driver automatically is powered down if it does not detect a properly terminated cable attached to the connector. If the SDI cable driver is powered down, the SDI receiver is not able to receive the signal from the transmitter because this loopback happens at the output of the SDI cable driver.

TX1 and TX2 have connectors that are separate from the receivers of those two channels. Thus, the TX VIO control windows for TX1 and TX2 do not have the **TX Enable Out** buttons.

The SDI receivers each have a VIO window to monitor the status of the receiver and an ILA window through which the actual video data received by the SDI RX and captured by the ChipScope Pro analyzer ILA can be viewed. Figure 20 shows the VIO window for one of the receivers.

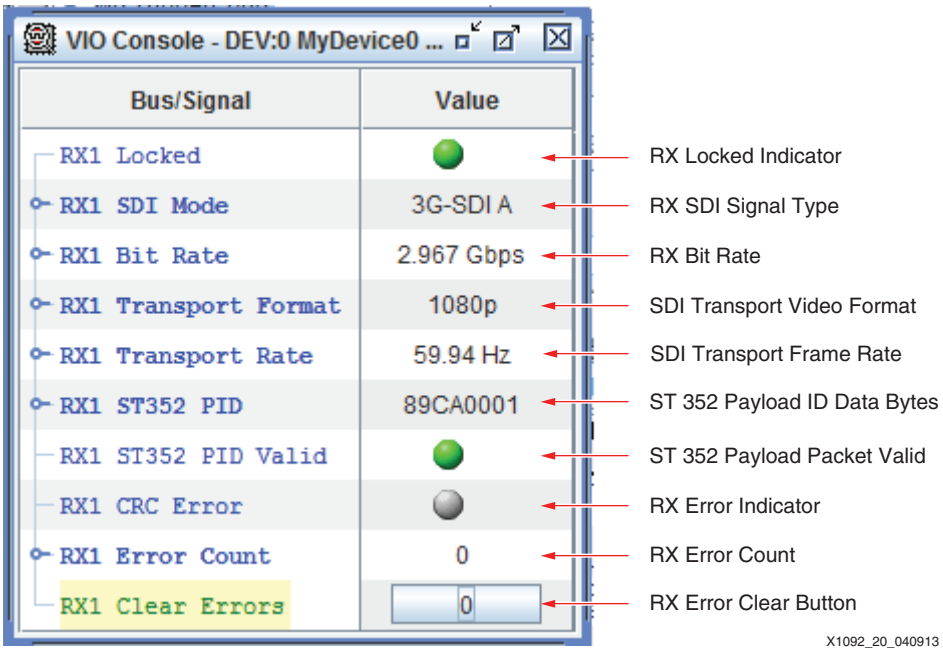


Figure 20: **Quad SDI Demonstration RX Status Window**

The RX Locked Indicator is green when the SDI RX is locked to the incoming SDI signal, and grey when it is not locked.

The RX SDI Signal Type shows the type of SDI signal being received: SD-SDI, HD-SDI, 3G-SDI level A, or 3G-SDI level B. This field does not distinguish between 3G-SDI levels B-DL and B-DS.

The RX Bit Rate shows the bit rate of the SDI signal being received.

The SDI Transport Video Format provides information about the video transport that has been detected in the SDI signal. The SDI Transport Frame Rate is the frame rate of the video transport that has been detected in the SDI signal. Both of these refer to the transport structure, not necessarily the picture format. For example, if the signal is 1080p 50 Hz carried on a 3G-SDI level B-DL interface, the transport would be detected and reported as 1080i 25 Hz (frame rate).

The ST 352 Payload ID Data Bytes are the four data bytes of the ST 352 payload ID packet [Ref 7]. They are shown with byte 1 on the left and byte 3 on the right, and are only valid when the ST 352 Payload Packet Valid indicator is green.

The RX Error Indicator is red if any CRC or EDH error has been detected, and grey if no errors have been detected. After an error has been detected, this indicator stays red until it is manually reset by clicking the **RX Error Clear** button. The RX Error Count is an integer count of the number of CRC (HD-SDI and 3G-SDI modes) or EDH errors (SD-SDI mode only) received since the counter was last cleared. The error counter is manually cleared by clicking the **RX Error Clear** button. The error counter is also cleared automatically when the incoming SDI signal changes bit rates and the SDI RX has to re-lock to the signal. However, the error counter is automatically cleared early in the process of locking to the new SDI signal. Thus, after the SDI RX has fully locked to the new SDI signal, the error count typically is not zero.

Figure 21 illustrates how to use the ChipScope Pro analyzer ILA to view the data being received by an SDI receiver. Each receiver has an ILA connected to its outputs. To use one of these ILAs, its trigger setup and waveform windows must be brought to the foreground in the ChipScope Pro analyzer window. One way to do that is to click the **Trigger Setup** and **Waveform** items under the appropriate UNIT in the Project panel in the upper left, as shown in

the figure. UNIT 3 is the ILA for RX1, UNIT 6 is the ILA for RX2, UNIT 9 is the ILA for RX3, and UNIT 12 is the ILA for RX4.

The **Trigger Setup** window can be used to change the trigger point and storage qualification. There are two match units. Typically, match unit M0 is used to trigger the ILA capture and match unit M1 is used to qualify the data storage, usually when the clock enable is High so that, in SD-SDI mode, only valid data words are captured.

With either the **Trigger Setup** window or the **Waveform** window for the desired receiver selected, click the triangular **play** button as shown in Figure 21 to initiate a capture by the ILA. The capture buffer is large enough to capture multiple lines of video.

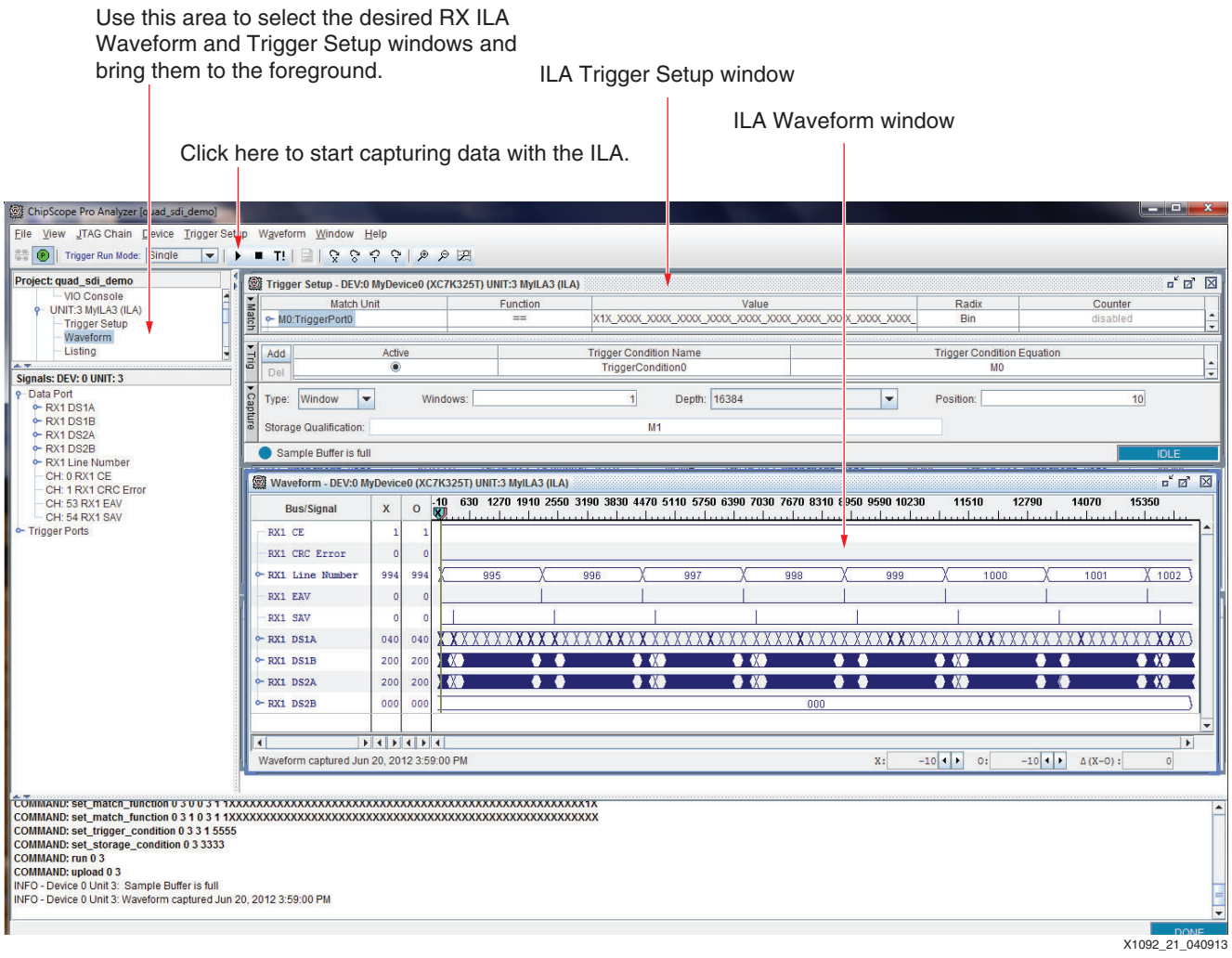


Figure 21: **Using the ChipScope ILA to View RX Data in the Quad SDI Demonstration**

SDI Pass-Through Demonstration

The second SDI demonstration has one SDI RX and one SDI TX connected together in a pass-through configuration such that the TX always retransmits the data received by the RX. Figure 22 is a block diagram of this demonstration.

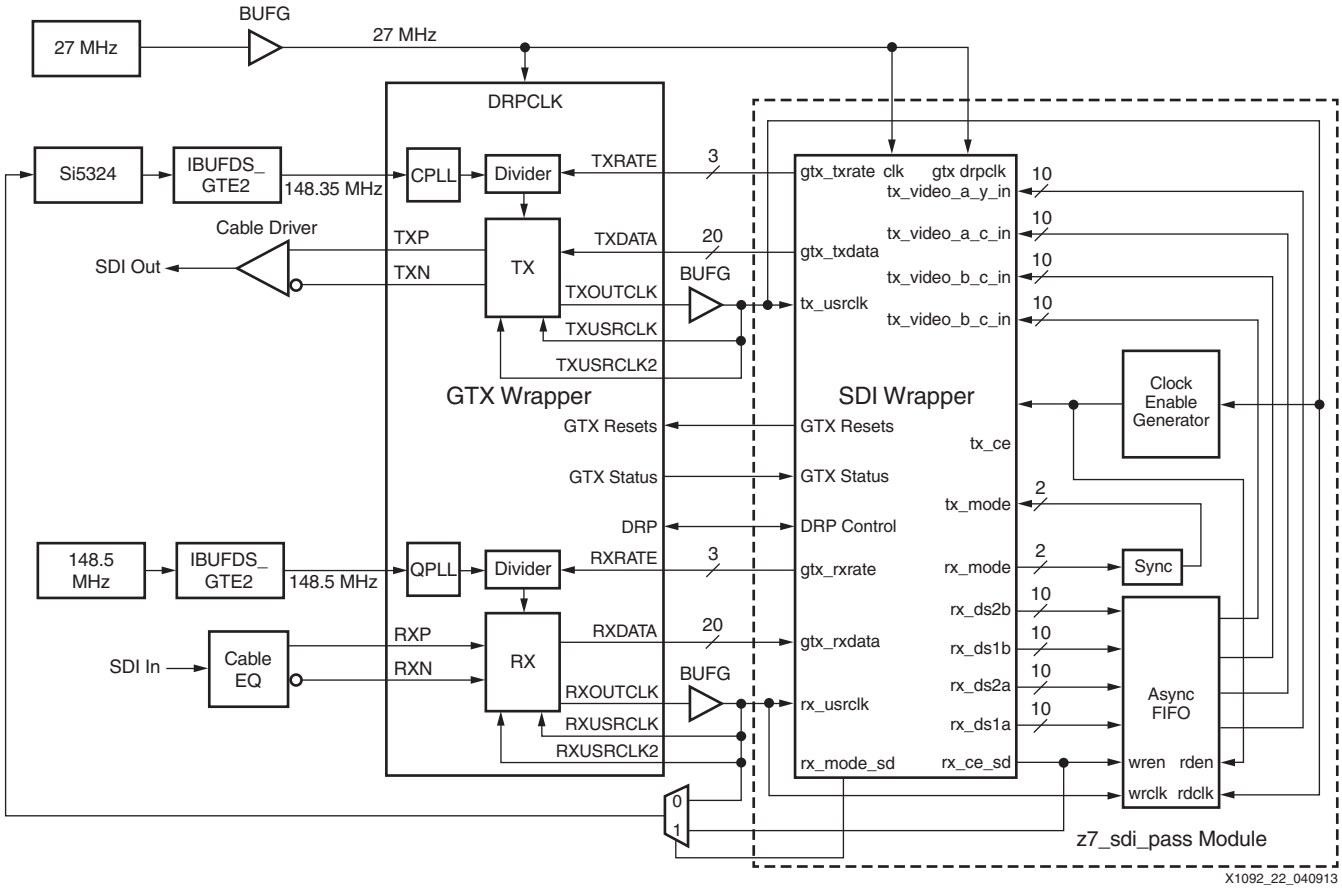


Figure 22: **SDI Pass-Through Demonstration**

The QPLL is locked to a 148.5 MHz reference clock and provides the serial clock to the GTX RX unit. The data from the GTX RX goes through the SDI RX datapath and then into an asynchronous FIFO. The FIFO moves the data from the RX clock domain (rx_usrclk) to the TX clock domain (tx_usrclk). In HD-SDI and 3G-SDI modes, the recovered clock from the GTX RX (RXOUTCLK) is sent to a Si5324 digital PLL for jitter reduction and then used as the reference clock to the CPLL. In SD-SDI mode, RXOUTCLK is not a recovered clock and cannot be used to generate the TX reference clock. Instead, the 27 MHz SD-SDI RX clock enable (rx_ce_sd) is sent to the Si5324 which multiplies it to 148.5 MHz while also filtering the jitter. The CPLL is locked to the clock from the Si5324 and provides the serial clock to the GTX TX. Data is read from the asynchronous FIFO in the TX clock domain and enters the SDI TX datapath. The resulting SDI data from the SDI TX datapath goes into the GTX TX for serialization.

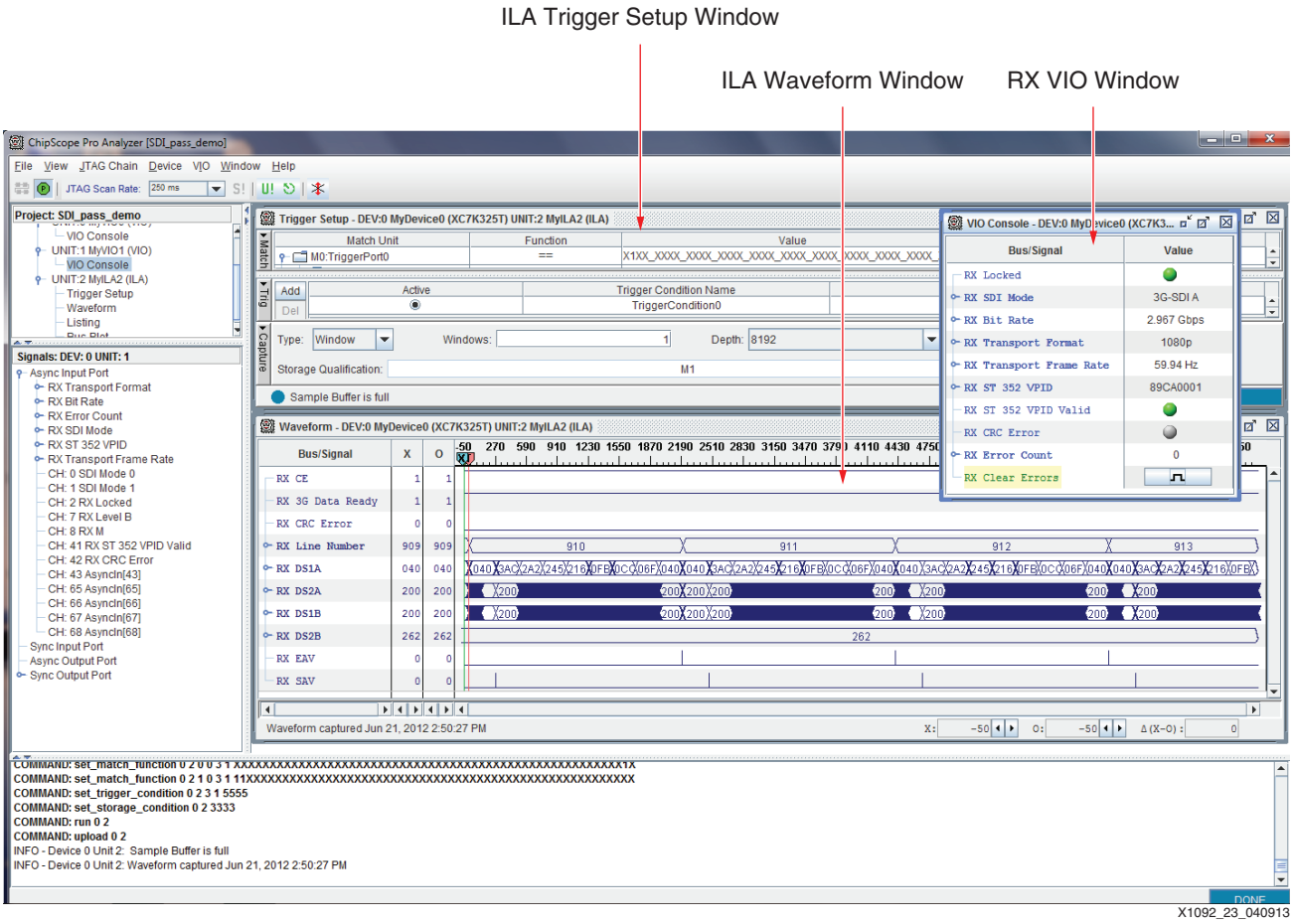
The following items are required to run the SDI pass-through demonstration:

- Xilinx Zynq-7000 SoC ZC706 Evaluation Kit
- inrevium TB-FMCH-3GSDI2A SDI FMC
- DIN 1.0/2.3 to BNC converter cables
- SDI signal source
- SDI signal sink (waveform monitor or other device to view signal from SDI transmitters)
- (Optional) A PC with ChipScope Pro analyzer installed and connected to the ZC706 board's JTAG USB connector

The inrevium SDI FMC must be connected to the HPC FMC connector on the ZC706 board as shown in Figure 17. The only active SDI connectors on the inrevium board are CH0-RX and

CH0-TX. An SDI signal source must be connected to the CH0-RX connector. The SDI signal is retransmitted on the CH0-TX connector.

The file called `zc706_sdi_pass_demo.bit` provided with this application note must be loaded into the PL portion of the Zynq-7000 SoC on the ZC706 board. After the BIT file is loaded into the FPGA, the `zc706_sdi_pass_demo.cpj` ChipScope analyzer project file can be opened in the ChipScope Pro analyzer to observe the status of the SDI RX and to capture and observe data received by the SDI RX as shown in Figure 23.



X1092_23_040913

Figure 23: **Pass-Through Demonstration ChipScope Analyzer Window**

The VIO window shows the status of the SDI RX. This window is identical to the RX status VIO window in the Quad SDI demonstration shown in Figure 20. Refer to the description of the fields and controls for the window given in that section describing that demonstration. It should be noted, however, that the SDI pass-through demonstration only has a single RX status VIO window because there is only one active SDI RX.

Likewise, there is a single ILA used to capture and observe the data from the SDI RX. It, too, functions just like the SDI RX ILA in the Quad SDI demonstration.

The SDI pass-through demonstration can be run without the ChipScope Pro analyzer. The pass-through SDI interface is fully functional even when the ChipScope Pro analyzer is not used to observe the SDI RX.

FPGA Resource Usage

Table 10 shows the FPGA resources required by an SDI interface with a Zynq-7000 SoC GTX transceiver. The resource usage includes all the modules required to implement the interface, including the SMPTE SDI core and the SDI wrapper. Resource usage is shown for various common configurations.

The results shown were achieved with **XST** and **MAP** set to **optimize for area**. The XST **Safe Implementation** property was set to **Yes**.

The SDI receiver and transmitter interface designs do not use any MMCM clock managers. They also do not require any block RAMs or DSP48E1 slices.

Typically, one global or regional clock is required for each SDI TX and SDI RX. In addition, one fixed-frequency global clock is required for timing purposes in the SDI wrapper. This fixed-frequency clock is usually also used as the GTX DRP clock. Only one such fixed-frequency global clock is required no matter how many SDI interfaces are implemented in the FPGA.

Table 10: **Zynq-7000 SoC GTX SDI Interface FPGA Resource Usage**

Reference Design	LUT-Flip-Flop Pairs	LUTs	Flip-Flops
SDI RX with EDH processor and TX	4,149	3,482	2,430
SDI RX without EDH processor and TX	3,554	2,979	2,050
SDI RX with EDH processor	2,820	2,387	1,569
SDI RX without EDH processor	2,200	1,897	1,189
SDI TX	1,313	1,101	860

Constraints

Because of the use of 20-bit RX and TX datapaths to and from the GTX transceiver, the maximum clock frequency used in these designs is 148.5 MHz. Meeting timing in the slowest speed grade Zynq-7000 SoCs is typically not a problem.

The RXOUTCLK and TXOUTCLK generated by the GTX can, under some conditions, have erratic timing when the input SDI bit stream stops and restarts or when the GTX is switched between SDI modes. These periods of erratic timing can cause problems for sequential control logic, such as finite state machines. Most synthesis tools, by default, optimize away illegal state recovery for finite state machines. The erratic timing of the GTX clocks can cause finite state machines to go into illegal states. Thus, it is highly recommended that the synthesis tool be forced to generate “safe” implementations of finite state machines that include illegal state recovery.

Example constraint files are supplied with the reference designs and can be used as examples of the timing and placement constraints required for SDI interfaces. For timing, generally all that is required are period constrains on the RXOUTCLK and TXOUTCLK clocks from the GTX transceiver. These constraints should specify the period of these clocks as 148.5 MHz. For placement, all that is required is to constrain the GTX transceivers to their desired locations by constraining the RXP/RXN and TXP/TXN pins or using the XY coordinate system to constrain the actual location of the GTX transceiver itself. All GTX transceivers that are instanced in the same GTX wrapper must be constrained to be in the same GTX Quad tile.

Glossary

Table 11 lists a glossary of terms used in this application note.

Table 11: **Glossary**

Term	Definition
3G-SDI	Common name for SMPTE ST 424, the 3 Gb/s serial digital interface [Ref 14]. 3G-SDI supports three mapping modes defined in ST 425-1 called 3G-SDI level A, level B-DL, and level B-DS. Refer to *Source Image Format and Ancillary Data Mapping for the 3 Gb/s Serial Interface* (ST 425-1) [Ref 15] for details about these mapping modes.
Ancillary (ANC) data	Non-video data embedded in portions of the SDI data stream not used for active picture data. One very common type of ANC data is embedded audio. ANC data must be formatted into ancillary data packets, as specified by SMPTE *Television – Ancillary Data Packet and Space Formatting* (ST 291) [Ref 16].
Data stream	The actual data into and out of the SDI interface. The data stream must be formatted according to the transport data structure as it enters and exits the SDI interface.
EDH	The error detection and handling protocol for SD-SDI as defined by SMPTE *Error Detection Checkwords and Status Flags for Use in Bit-Serial Digital Interfaces for Television* (RP 165) [Ref 13].
Embedded audio	Generally refers to digital audio that is carried as ancillary data in an SDI signal.
End of active video (EAV)	In SDI compatible data streams, the EAV is a sequence of four words, unique in the data steam, marking the end of the active portion of the line and start of the horizontal blanking interval. Each video line is considered to begin with the first word of the EAV.
HD-SDI	Common name for the SMPTE *1.5 Gb/s Signal/Data Serial Interface (*ST 292-1) [Ref 17].
Interlaced	A video scanning system in which the video frame is divided into two sequential fields. Field one consists of the odd lines and field two consist of the even lines that are displayed between the odd lines of field one. The two fields represent different pictures displaced in time by one half of the frame time.
Link	If the picture's bandwidth exceeds the capacity of the serial digital interface, two or more serial digital interfaces can be ganged together to increase the bandwidth to transport the picture. Each separate serial digital interface of a multi-link set is called a link. SMPTE *Dual Link 1.5 Gb/s Digital Interface for 1920 x 080 and 2048 x 1080 Picture Formats* (ST 372) [Ref 12] defines how to transport some higher bandwidth video formats on two HD-SDI links. Multi-link 3G-SDI standards in the ST 425-x family are currently under development by SMPTE [Ref 15]. The 3G-SDI level B-DL transport carries both links of a dual-link HD-SDI (ST 372) pair on one 3G-SDI interface. Each of the two HD-SDI signals carried by 3G-SDI level B-DL is still called a link.
Payload ID	Sometimes called the Video Payload ID (VPID), the payload ID is an ancillary data packet defined by SMPTE *Payload Identifier Codes for Serial Digital Interfaces* (ST 352) [Ref 7]. The four data words of the ST 352 payload ID packet identify both the nature of the video picture (video format, frame rate, scanning structure, and color space) and the type of SDI interface used to transport that payload. In multi-link interfaces, the payload ID also contains bits that distinguish between the individual links.

Table 11: **Glossary *(Cont'd)***

Term	Definition
Progressive	A non-interlaced video scanning system. All lines of the progressive frame belong to the same picture.
SD-SDI	Common name for SMTPE *Television – SDTV Digital Signal/Data – Serial Digital Interface* (ST 259) [Ref 4], the standard-definition serial digital interface.
Serial Digital Interface (SDI)	Originally referred to as SMPTE *Television – SDTV Digital Signal/Data – Serial Digital Interface* (ST 259) [Ref 4], the standard-definition serial digital interface. With the advent of HD-SDI and 3G-SDI, ST 259 is now often called SD-SDI to avoid confusion. This document uses the term *SDI* to generically refer to SD-SDI, HD-SDI and 3G-SDI. When referring specifically to ST 259, this document always uses the term *SD-SDI*.
SMPTE	Society of Motion Picture and Television Engineers.
Start of active video (SAV)	In SDI-compatible data streams, the SAV is a sequence of four words, unique in the data stream, marking the end of the horizontal blanking interval and the start of the active video portion of the line. The first active video sample of a line, usually called sample 0, occurs immediately after the SAV.
Synchronous switching (point, interval, line)	SMPTE *Definition of Vertical Switching Point for Synchronous Video Switching* (RP 168) [Ref 18] defines the point(s) in a video frame where it is permissible to switch between synchronous video sources. This is often called the synchronous switching point but is actually defined as an interval, a portion of a line, rather than an exact point on a line. The line that contains the synchronous switching interval is sometimes called the *synchronous switching line*.
Timing reference signal (TRS)	A generic term referring to both EAV and SAV sequences.
Transport	The data structure of an interface data stream or streams. The transport data structure defines the EAV and SAV sequences used to carry video timing information.
XYZ	The fourth word of each EAV and SAV is called the XYZ word. This word carries the horizontal (H), vertical (V), and field (F) bits that indicate the video timing. The XYZ word also contains some protection bits that allow detection of errors in the XYZ word.

Reference Design

The reference design files for this application note can be downloaded from:

https://secure.xilinx.com/webreg/clickthrough.do?cid=323535

The reference design checklist is shown in Table 12.

Table 12: **Reference Design Checklist**

Parameter	Description
General	
Developer name	John Snow
Target devices	Zynq-7000 SoCs with GTX transceivers
Source code provided	Yes
Source code format	Verilog

Table 12: **Reference Design Checklist *(Cont'd)***

Parameter	Description
Design uses code/IP from existing Xilinx application note/reference designs, the CORE Generator tool, or third party	Yes, IP cores from CORE Generator tool
Simulation	
Functional simulation performed	No
Timing simulation performed	No
Test bench used for functional and timing simulations	None
Test bench format	N/A
Simulator software/version used	N/A
SPICE/IBIS simulations	N/A
Implementation	
Synthesis software tools/version used	XST in ISE Design Suite 14.5
Implementation software tools/versions used	ISE Design Suite 14.5
Static timing analysis performed	Yes
Hardware Verification	
Hardware verified	Yes
Hardware platform used for verification	ZC706 and TB-FMCH-3GSDI2A boards

The `readme.txt` file describes the directory structure of the files that come with the ZIP file.

Conclusion

This document describes how to use the SMPTE SD/HD/3G-SDI core and the Zynq-7000 SoC GTX transceivers to implement SDI interfaces compatible with the SMPTE SD-SDI, HD-SDI, and 3G-SDI standards. The Zynq-7000 SoC GTX device-specific control logic necessary to use the transceivers in SDI applications is included with this application note. Also included are two example SDI demonstration applications providing detailed examples of SDI implementations in a Zynq-7000 SoC design.

References

This section lists the references used in this document.

1. *7 Series FPGAs GTX/GTH Transceivers User Guide* (UG476)
2. *Dynamically Programmable DRU for High-Speed Serial I/O* (XAPP875)
3. *Zynq-7000 All Programmable SoC (XC7Z030, XC7Z045, and XC7Z100): DC and AC Switching Characteristics* (DS191)

The following references are available from the Society of Motion Picture and Television Engineers (www.smpte.org):

4. ST 259, *Television – SDTV Digital Signal/Data – Serial Digital Interface*
5. ST 344, *Television – 540 Mb/s Serial Digital Interface*
6. PG071, *Society of Motion Picture and Television Engineers (SMPTE) SD/HD/3G-SDI Product Guide*
7. ST 352, *Payload Identification Codes for Serial Digital Interfaces*
8. ST 274: *Television – 1920 x 1080 Image Sample Structure, Digital Representation and Digital Timing Reference Sequences for Multiple Picture Rates*

9. ST 296: *1280 x 270 Progressive Image 4:2:2 and 4:4:4 Sample Structure — Analog and Digital Representations and Analog Interface*
10. ST 2048-2: *2048 x 1080 Digital Cinematography Production Image FS/709 Formatting for Serial Digital Interface*
11. T 295: *Television – 1920 x 1080 50-Hz - Scanning and Interface*
12. ST 372, *Dual Link 1.5 Gb/s Digital Interface for 1920 x 1080 and 2048 x 1080 Picture Formats*
13. RP 165, *Error Detection Checkwords and Status Flags for Use in Bit-Serial Digital Interfaces for Television*
14. ST 424, *Television – 3 Gb/s Signal/Data Serial Interface*
15. ST 425-1, *Source Image Format and Ancillary Data Mapping for the 3 Gb/s Serial Interface*
16. ST 291, *Ancillary Data Packet and Space Formatting*
17. ST 292-1, *1.5 Gb/s Signal/Data Serial Interface*
18. RP 168, *Definition of Vertical Switching Point for Synchronous Video Switching*

Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
07/08/2013	1.0	Initial Xilinx release.

Notice of Disclaimer

Automotive Applications Disclaimer