XAPP592 (v2.0) July 14, 2014

Implementing SMPTE SDI Interfaces with Kintex-7 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 SD/HD/3G-SDI LogiCORE IP with the Kintex®-7 FPGA GTX transceivers to form a complete SDI interface. This application note also provides several example SDI designs that run on the Xilinx Kintex-7 FPGA KC705 evaluation board.

Terms used in this document are explained in the Glossary, page 63. Titles of SMPTE reports and standards are listed in References, page 66, 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 Kintex-7 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: 1.485 Gb/s and 1.485/1.001 Gb/s in HD-SDI mode and 2.97 Gb/s and 2.97/1.001 Gb/s in 3G-SDI mode
- 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 bit rate signal or a 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 SD/HD/3G-SDI core with the necessary connections between them. This file simplifies the process of creating an SDI interface.

In this document, the following terms are used. The *SDI core* refers to the SMPTE SD/HD/3G-SDI core generated by the CORE Generator™ tool or the Vivado® tool IP catalog. The *control module* is a module that implements the various device-specific functions required when using the GTX to implement an SDI interface using the SMPTE SD/HD/3G-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 SD/HD/3G-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 a single GTX transceiver generated by the 7 Series FPGAs Transceivers Wizard that is available in the CORE Generator and IP Catalog tools. The *GTX common wrapper* is a wrapper file containing the QPLL for the GTX Quad also generated by the 7 Series FPGAs Transceivers Wizard when the GTX wrapper is generated.

The SDI wrapper includes one instance of a control module and one instance of an SMPTE SD/HD/3G-SDI core. The SMPTE SD/HD/3G-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 SMPTE SD/HD/3G-SDI core are optimized away during synthesis.

This application note includes two example demonstration applications using the SDI core. These applications run on the KC705 evaluation board. An inrevium SDI FPGA mezzanine card (FMC) is also required to provide the SDI physical interfaces.

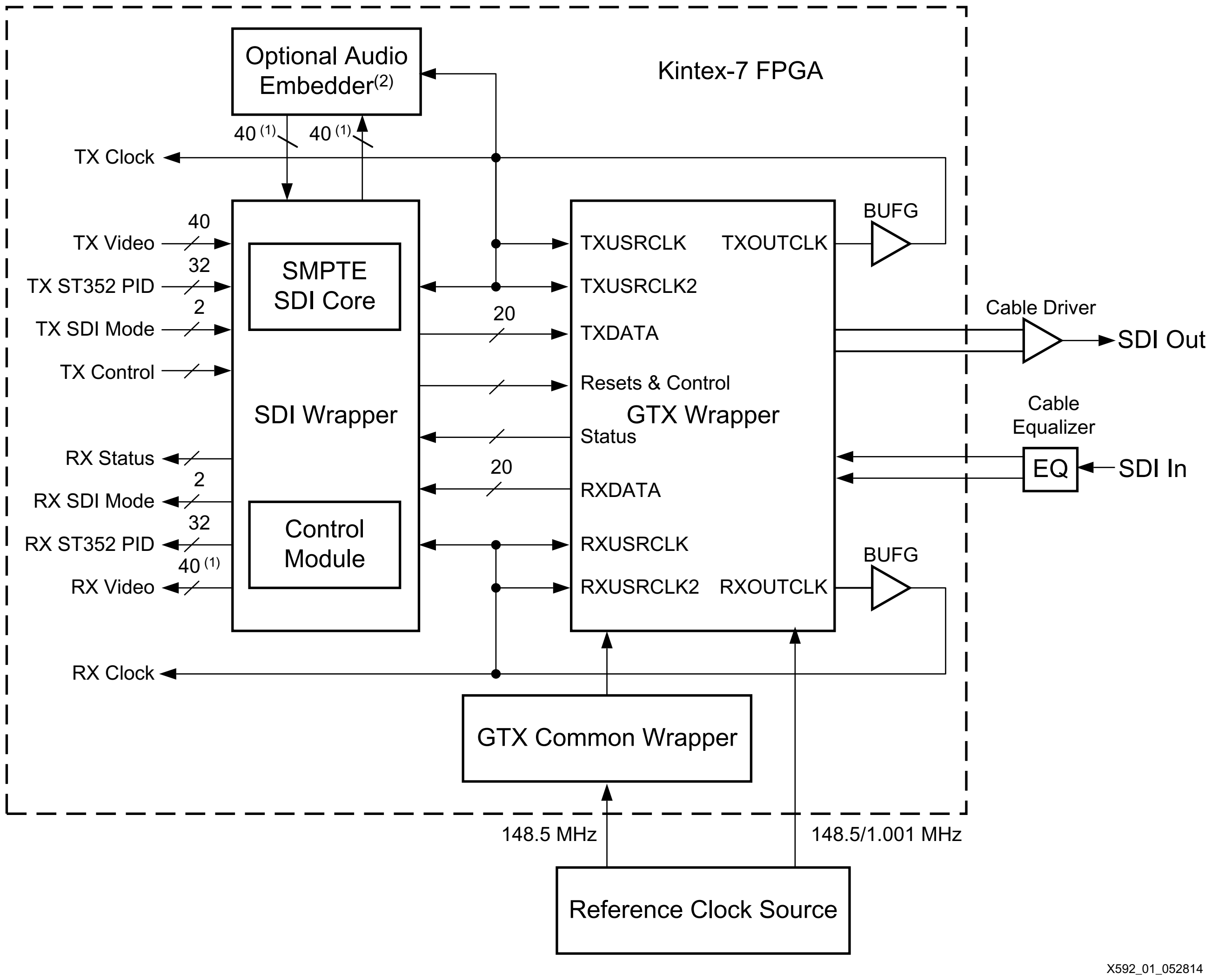


Figure 1: **Block Diagram of Complete SDI RX/TX Interface**

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 SD/HD/3G-SDI core nor with this application note.

Using Kintex-7 GTX Transceivers for SDI Interfaces

The information in this section is intended to supplement, not replace, the information in *7 Series GTX/GTX Transceivers User Guide* (UG476) [Ref 1]. 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 GTX/GTX Transceivers User Guide* (UG476) 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, 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.

There are several clocks required 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 transceiver Quad to generate serial clocks for the receiver and transmitter sections of each transceiver. As described in more detail in the GTX Transceiver Reference Clocks section, the serial bit rate of the GTX transmitter is an integer 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, you must determine how to generate the transmitter reference clock so that it is frequency-locked exactly with the data rate of the video stream being transmitted.

The GTX transmitter outputs a clock on its txoutclk port 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 (BUFG) or horizontal (BUFH) 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 clock other than one derived directly from txoutclk as the clock source for 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 clock 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 underflow or overflow, resulting in errors in the serial bitstream generated by the GTX transmitter. Consequently, the data rate of the data stream entering the txdata port of the GTX transmitter (as 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 can receive bit rates that are up to ±1,250 ppm away from the nominal bit rate as set by the reference clock frequency. This 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 in this application note, 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 horizontal 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 module in the SDI wrapper where it is used for timing purposes. Xilinx recommends that the frequency of this clock be at least 10 MHz. The frequency of this clock does not require any specific relationship relative to other clocks or data rates of the SDI application. This clock must not change frequencies when the SDI mode changes. It must always remain running at the same nominal frequency at all times. It also must never stop while the SDI application is active. This clock can be used for all SDI interfaces in the device.

GTX Transceiver Reference Clocks

Kintex-7 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 the 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 clock and data recovery (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. Because the two bit rates of HD-SDI differ by exactly 1000 ppm and likewise the two 3G-SDI bit rates differ by exactly 1000 ppm, it is possible to receive all five SDI bit rates using a single reference clock frequency while still providing a sufficient amount of ppm offset margin.

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 transceiver reference clocks for SDI applications is very application-specific. The receiver reference clock source can be a local oscillator because it does not need to match the incoming SDI bit rate exactly. However, because the GTX transmitter line rate is always an integer multiple of the reference clock frequency, the frequency of the transmitter reference clock must be exactly related to the data rate of the

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.

transmitted data. Most often, the transmitter reference clocks are generated by genlock PLLs, thereby deriving the GTX transmitter line rate from the studio video reference signal. In some cases, such as the SDI pass-through demonstration included with this application note, the transmitter line rate is derived from the recovered clock of the GTX receiver that is receiving the 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.

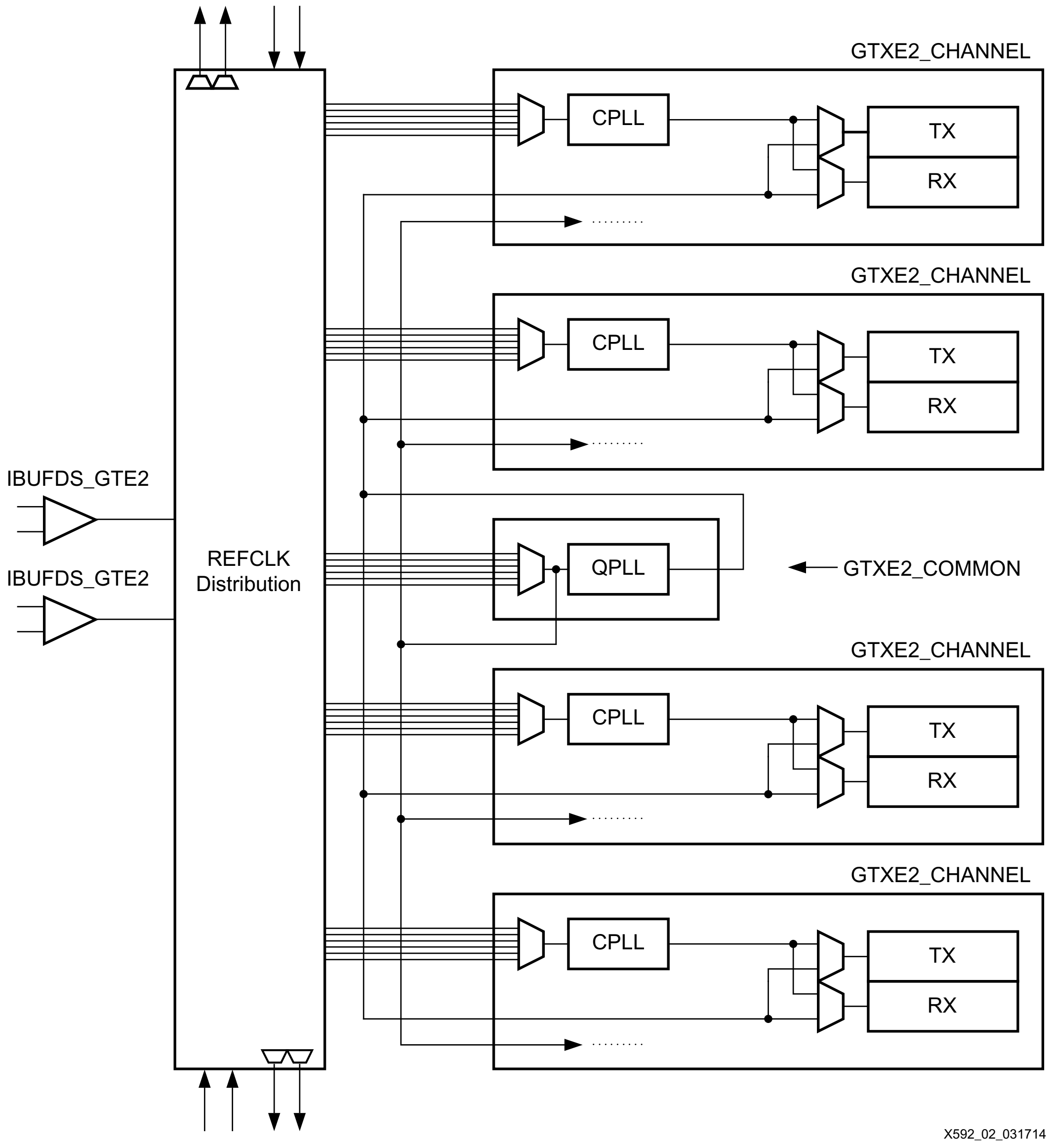


Figure 2: **Kintex-7 FPGA GTX Transceiver Quad Configuration**

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.

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. And 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 and 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.

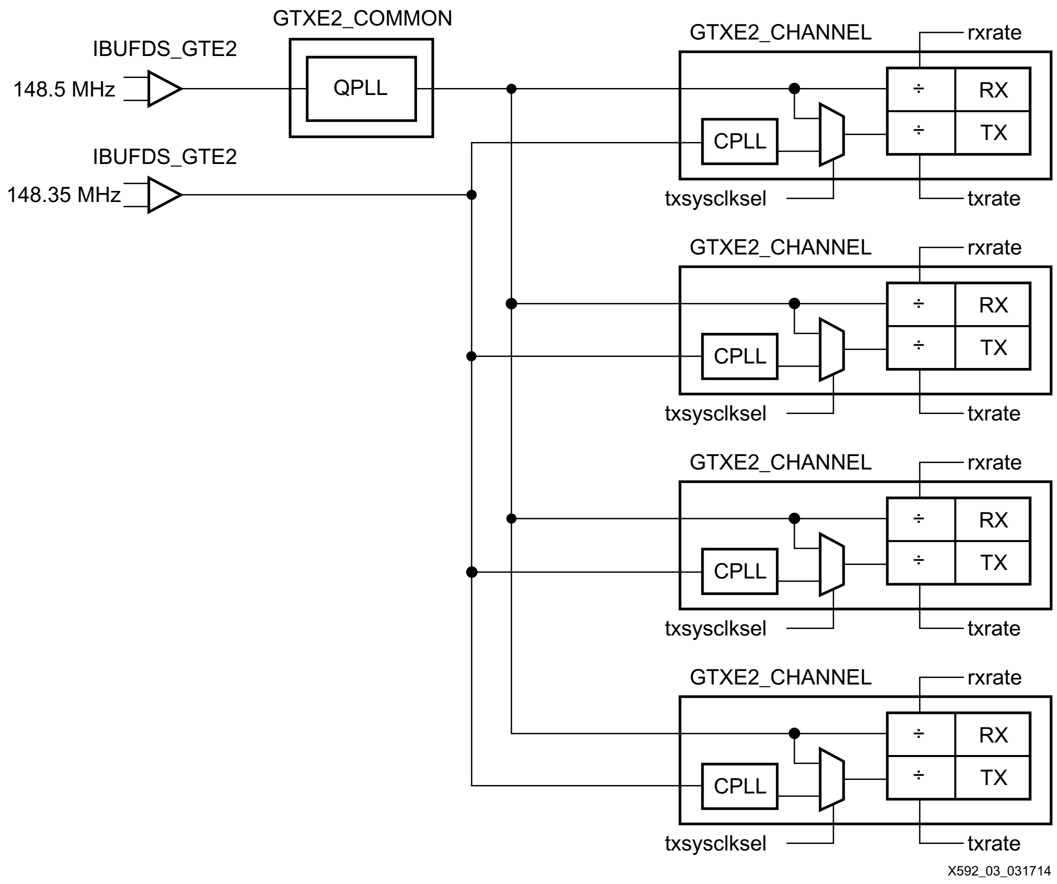


Figure 3: **Typical GTX Reference Clock Implementation for SDI**

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 a perfectly legal and viable configuration and can be implemented if desired.

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 Kintex-7 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 Quad above and the Quad 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 obviously 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.

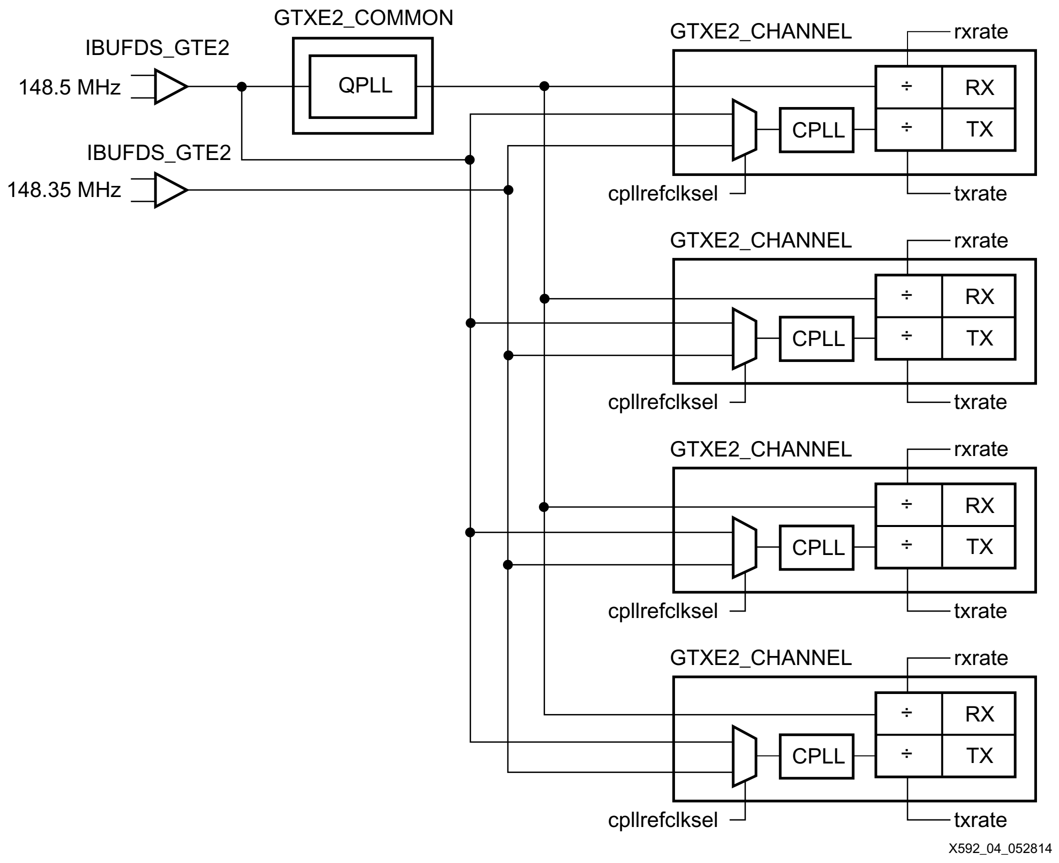


Figure 4: **Alternative Reference Clock Implementation for SDI**

Resets

The GTX transceiver has very specific reset requirements as described in the *7 Series GTX/GTX Transceivers User Guide* (UG476) [Ref 1]. The GTX transceiver requires careful coordination of resets of the PLLs, GTX transceiver resets (gttxreset and gtrxreset), dynamic changes of some GTX transceiver ports such as txrate, and dynamic changes of GTX transceiver attributes through the DRP. Without proper coordination of all of these events, it is possible for the GTX to fall into a state in which it does not function properly for SDI, a situation from which the only possible recovery is to reconfigure the FPGA. The control module supplied with this application note enforces all of these requirements to ensure proper operation of the GTX transceiver.

The user application should never directly control the GTX inputs gttxreset and gtrxreset. To ensure proper operation of the GTX transceiver, these GTX transceiver inputs must only be controlled by the SDI control module. The user application can request GTX resets using the various reset inputs of the control module. These reset requests are handled at the next appropriate time by the control module, coordinating the resets with other GTX transceiver actions so that they do not interfere.

GTX Transceiver Initialization Sequence

Immediately following FPGA configuration, the SDI control module executes initialization sequences for the QPLL, the CPLLs, and the RX and TX units of the GTX transceiver. The control module has separate state machines that execute the following initialization sequence separately for the RX and the TX portions of the GTX transceiver. The sequence described here is for the RX. The TX initialization sequence is identical except that the transmitter and CPLL ports replace the receiver and QPLL ports.

1. After waiting at least 500 ns following completion of FPGA configuration, assert the qpllreset and gtrxreset signals.
2. Wait until the rx_refclk_stable input is asserted, then negate the qpllreset.
3. Wait until the qplllock signal is asserted, then negate the gtrxreset signal.
4. Wait until the rxresetdone signal is asserted, then indicate that the initialization sequence is complete.

Also, the GTX txuserrdy and rxuserrdy inputs must be properly controlled. The SDI wrapper generates both of these signals. It asserts txuserrdy five txusrclk cycles after gttxreset is negated. Likewise, it asserts rxuserrdy five rxusrclk cycles after gtrxreset is negated. In step 2, step 3, and step 4 of the initialization sequence where the sequence is waiting on a condition to be satisfied, a timeout counter is running. If the timeout counter expires before the wait condition is satisfied, the state machine moves to a timeout state where it increments a retry counter and then cycles back in the initialization sequence and resumes the sequence. If the retry counter reaches its maximum count due to numerous timeouts, the initialization sequence fails and the state machine moves to a fail state, indicating failure of the initialization sequence. The maximum number of retries allowed is controlled by a parameter of the SDI wrapper.

PLL Resets

Besides being reset during the initialization sequences that run automatically after FPGA configuration, a QPLL or CPLL must also be reset whenever there is a change in frequency or interruption of the reference clock supplied to that PLL. The reset is required to force the PLL to relock to the reference clock. The qpllreset input of the GTX common wrapper and the cpllreset input of the GTX wrapper are controlled by the SDI control module to implement the PLL resets. The user application should not assert the PLL resets directly. The SDI control module, alone, should control the PLL resets. However, it is up to the user application to determine when PLL resets are required. When a PLL must be reset, the application must request that the SDI control module reset the PLL and all of the GTX RX and/or TX units using the serial clock from that PLL. The SDI wrapper has a rx_pllreset output and a tx_pllreset output. These are used to control the qpllreset input of the GTX common wrapper and the cpllreset input of the GTX wrapper. If a PLL is used by only one RX or one TX unit, it is a simple matter to connect the correct rx_pllreset or tx_pllreset output of the SDI wrapper to the corresponding PLL reset input port. But, when a PLL provides the serial clock to multiple RX and/or TX units, it is more complicated. See the GTX PLL Usage Models for SDI Applications, page 11 section for more details.

The SDI wrapper has two inputs that the application should use to request a full reset of the GTX RX (rx_gtx_full_reset) and the GTX TX (tx_gtx_full_reset). Asserting either of these inputs causes the appropriate reset state machine in the control module to execute the full initialization sequence of the RX or TX section of the GTX, including resetting the associated PLL. The user application must properly control the rx_gtx_full_reset and tx_gtx_full_reset inputs so that these initialization sequences are done whenever there is an interruption or change in the reference clock used by the PLL.

It is up to the user application to properly control the rx_refclk_stable and tx_refclk_stable inputs to the control module. These must be asserted only when the reference clocks to the PLLs are stable. As previously described, the initialization sequences wait until these inputs are asserted before negating the PLL resets. Negating the rx_refclk_stable or tx_refclk_stable inputs does not initiate a reset of the associated PLL. PLL resets are only initiated by asserting the rx_gtx_full_reset and tx_gtx_full_reset inputs to the control module. The rx_refclk_stable and tx_refclk_stable are only effective in delaying completion of the reset sequence after the initialization sequence has been started by assertion of rx_gtx_full_reset or tx_gtx_full_reset.

GTX TX Resets

There are three conditions that require the TX portion of the GTX transceiver to be reset:

- Whenever the PLL that supplies the serial clock to the GTX TX is reset, the gttxreset port must be used to reset the TX section. This is done automatically after FPGA configuration by the SDI control module and whenever the user application asserts the tx_gtx_full_reset to the SDI wrapper, causing both the PLL and the GTX TX to be reset.
- The GTX gttxreset input must be asserted during dynamic changes of the txsysclksel port. The txsysclksel port is used to select between the QPLL or the CPLL as the serial clock source for the GTX TX. Each GTX transceiver has its own txsysclksel port and can independently switch its serial clock source between the two PLLs. The txsysclksel port should not be controlled directly by the application. The SDI control module dynamically changes the txsysclksel port of a GTX transceiver in response to changes on its tx_m input. When the control module detects a change on its tx_m input, it first asserts the gttxreset signal, then changes txsysclksel, and then negates gttxreset. The sequence is complete after the GTX transceiver asserts its txresetdone output. At that point, the SDI control module indicates completion of the txsysclksel change by asserting its tx_change_done output.
- The GTX TX is automatically reset by the GTX transceiver itself whenever its txrate input port dynamically changes. The txrate port controls the serial clock divider for the GTX TX. The user application should not change the txrate port directly. The SDI control module changes the txrate port, when appropriate, in response to changes on its tx_mode input port.

The SDI wrapper has three reset inputs for the TX section:

tx_rst	When asserted High, this input resets the SDI TX datapath in the SDI core.
tx_gtx_full_reset	When asserted High, this input resets both the PLL associated with the TX and then the TX section of the GTX transceiver (gttxreset). These two resets are sequenced so that the gttxreset does not complete until after the PLL reset is complete and the PLL is locked to its reference clock.
tx_gtx_reset	When asserted High, this input resets only the TX section of the GTX transceiver (gttxreset). If the PLL is not locked when the gttxreset sequence begins, the gttxreset sequence does not complete until after the PLL is locked

GTX RX Resets

As with the TX section, the user application should rely on the SDI control module to carefully coordinate all of the RX reset and dynamic change activities described here to prevent them from interfering with each other.

These conditions require resets of the GTX RX section:

- Whenever the PLL that supplies the serial clock to the GTX RX (usually the QPLL) is reset, the gtrxreset port must be used to reset the RX section. This is done automatically after FPGA configuration by the SDI control module and whenever the user application asserts the rx_gtx_full_reset to the SDI wrapper, causing both the PLL and the GTX RX to be reset.
- Changes in the SDI mode between SD-SDI, HD-SDI, and 3G-SDI require changes to one or more of the following three things: the rxcdrhold port, the rxrate port, and the RXCDR_CFG attribute. The RXCDR_CFG attribute is changed through the DRP. The rxcdrhold port must be asserted High when the RX SDI mode is SD-SDI and Low in other SDI modes. The RXCDR_CFG attribute is modified when switching into either HD-SDI or 3G-SDI modes to optimize the CDR for the current line rate. The rxrate port controls the serial clock divider for the GTX RX. The GTX RX must be reset using the gtrxreset port after dynamic changes are made to any of these three things. If more than one of these

things changes during the same SDI mode change sequence, only a single gtrxreset is required after all changes have been made.

The SDI wrapper has three reset inputs for the RX section:

- rx_rst: When asserted High, this input resets the SDI RX datapath in the SDI core.
- rx_gtx_full_reset: When asserted High, this input resets both the PLL associated with the RX and then the RX section of the GTX transceiver (gtrxreset). These two resets are sequenced so that the gtrxreset does not complete until after the PLL reset is complete and the PLL is locked to its reference clock.
- rx_gtx_reset: When asserted High, this input resets only the RX section of the GTX transceiver (gtrxreset). If the PLL is not locked when the gtrxreset sequence begins, the gtrxreset sequence does not complete until after the PLL is locked.

GTX PLL Usage Models for SDI Applications

This section describes several typical configurations of PLLs and transceivers used in SDI applications. Not every possible configuration is described, but the configurations shown here are sufficient to describe the proper connection of the PLL reset and locked signals. The SDI wrapper has two parameters that specify which TX serial clock sources come from the QPLL and which come from the CPLL. These attributes do not control the routing of PLL clocks. They are only used to determine the value to drive onto the GTX wrapper txsysclksel port based on the current value of tx_m. These two parameters are integers and must be assigned values as described here:

- The TX_CLK0_QPLL parameter must be set to 1 if the QPLL is the clock source for the GTX TX when the tx_m input to the SDI wrapper is Low. This parameter must be set to 0 if the CPLL is the clock source for the GTX TX when tx_m is Low.
- The TX_CLK1_QPLL parameter must be set to 1 if the QPLL is the clock source for the GTX TX when the tx_m input to the SDI wrapper is High. This parameter must be set to 0 if the CPLL is the clock source for the GTX TX when tx_m is High.

Both parameters are static. There are two parameters to support dynamic switching of the TX between the CPLL and the QPLL using the tx_m port of the SDI wrapper. TX_CLK0_QPLL is used when tx_m is Low and TX_CLK1_QPLL is used when tx_m is High. In applications where the TX is not dynamically switched between the QPLL and the CPLL, set both TX_CLK0_QPLL and TX_CLK1_QPLL to 1 if the QPLL is always the TX serial clock source and to 0 if the CPLL is always the TX serial clock source.

Usage Model 1: A Single Transceiver Is Active in the Quad, RX Clocked by QPLL, TX Clocked by Both QPLL and CPLL

In this usage model, shown in Figure 5, there is a single transceiver active in the Quad with the RX serial clock provided by the QPLL and the GTX TX dynamically switched between the QPLL and the CPLL. In this case, the RX portion of the SDI wrapper controls the QPLL reset and the TX portion controls the CPLL reset. The TX section must, however, observe the locked status of both the QPLL and the CPLL because both PLLs must be locked before the gttxreset cycle completes.

The following connections must be made:

- The gtx_rxpllreset output of the SDI wrapper must be connected to the qpllreset port of the GTX common wrapper.
- The gtx_txpllreset output of the SDI wrapper must be connected to the cpllreset port of the GTX wrapper.
- The gtx_rxplllock input of the SDI wrapper must be connected to the qplllock output of the GTX common wrapper.
- The gtx_txplllock input of the SDI wrapper must be driven by the logical OR of the GTX common wrapper qplllock output and the GTX wrapper cplllock output.

- The rx_refclk_stable input of the SDI wrapper must be asserted High only when the reference clock source to the QPLL is stable.
- The tx_refclk_stable input of the SDI wrapper must be asserted High only when the reference clock source to the CPLL is stable.
- The tx_m input port of the SDI wrapper controls dynamic switching of the TX serial clock source by controlling the gtx_txsysclksel output of the SDI wrapper which must be connected to the txsysclksel port of the GTX wrapper.
- The TX_CLK0_QPLL and TX_CLK1_QPLL parameters of the SDI wrapper must be set appropriately based on how the tx_m port is used to select between the QPLL and the CPLL. Typically, TX_CLK0_QPLL is set to 1 and TX_CLK1_QPLL is set to 0. This configures tx_m to select the QPLL as the TX serial clock source when tx_m is Low and the CPLL when tx_m is High.
- When the QPLL needs to be reset due to a reference clock change or interruption, assert the rx_gtx_full_reset input of the SDI wrapper to reset both the QPLL and the GTX RX. Also assert the tx_gtx_reset input of the SDI wrapper to reset the GTX TX without resetting the CPLL.
- When the CPLL needs to be reset due to a reference clock change or interruption, assert the tx_gtx_full_reset input of the SDI wrapper to reset both the CPLL and the GTX TX.

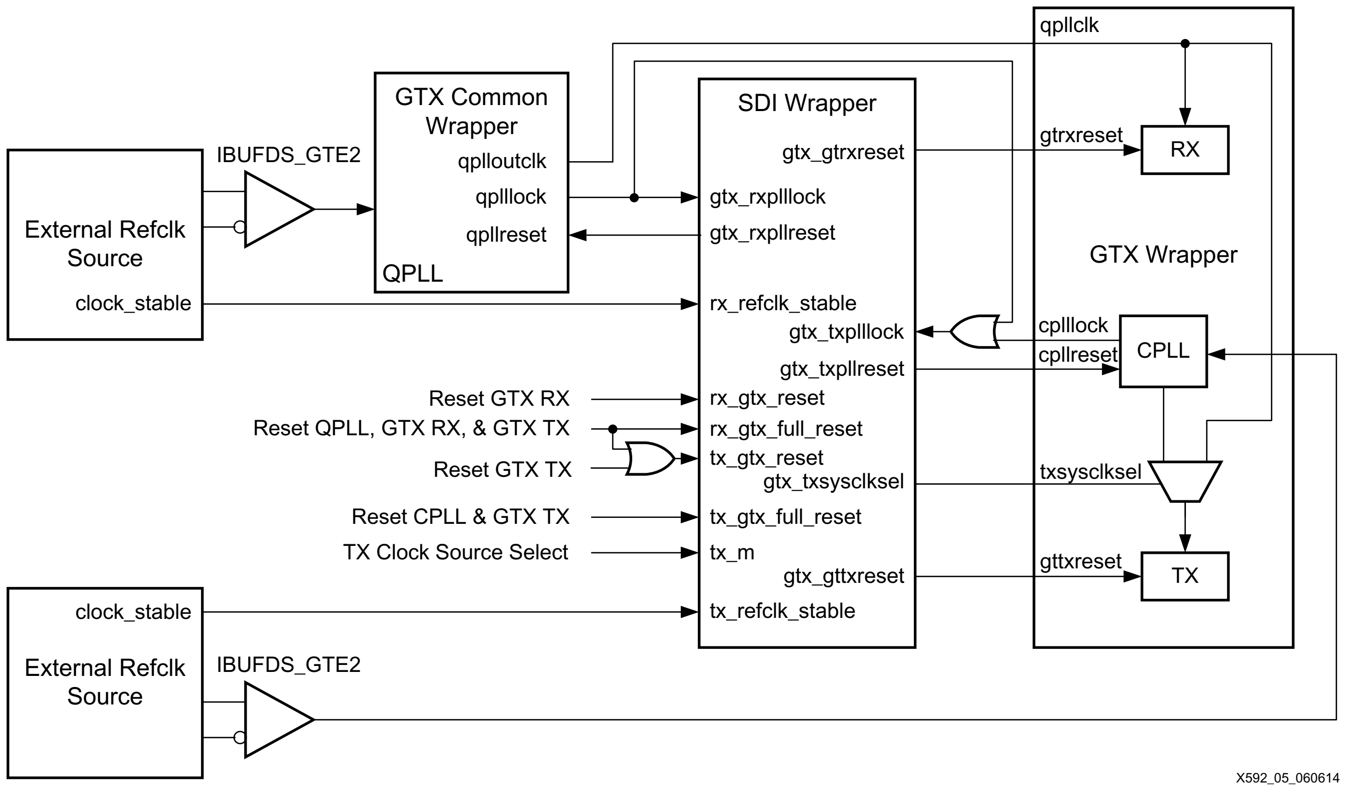


Figure 5: **PLL Usage Model 1**

Usage Model 2: A Single Transceiver Is Active in the Quad, RX Clocked by the QPLL, TX Clocked by the CPLL

In this usage model, shown in Figure 6, there is a single transceiver active in the Quad with the GTX RX clocked by the QPLL and the GTX TX clocked by the CPLL.

The following connections must be made:

- The gtx_rxpllreset output of the SDI wrapper must be connected to the qpllreset port of the GTX common wrapper.

- The gtx_txpllreset output of the SDI wrapper must be connected to the cpllreset port of the GTX wrapper.
- The gtx_rxplllock input of the SDI wrapper must be driven by the qplllock output of the GTX common wrapper.
- The gtx_txplllock input of the SDI wrapper must be driven by the cplllock output of the GTX wrapper.
- The rx_refclk_stable input of the SDI wrapper must be asserted High only when the reference clock source to the QPLL is stable.
- The tx_refclk_stable input of the SDI wrapper must be asserted High only when the reference clock source to the CPLL is stable.
- The txsysclksel input port of each GTX wrapper must be driven with a value of `2'b00` to permanently select the CPLL as the TX serial clock source. The gtx_txsysclksel output port of the SDI wrapper is left unconnected.
- The TX_CLK0_QPLL and TX_CLK1_QPLL parameters of the SDI wrapper must be set to 0.
- The tx_m input port of the SDI wrapper is not used and should be driven Low.
- When the QPLL needs to be reset due to a reference clock change or interruption, assert the rx_gtx_full_reset input of the SDI wrapper to reset both the QPLL and the GTX RX.
- When the CPLL needs to be reset due to a reference clock change or interruption, assert the tx_gtx_full_reset input of the SDI wrapper to reset both the CPLL and the GTX TX.

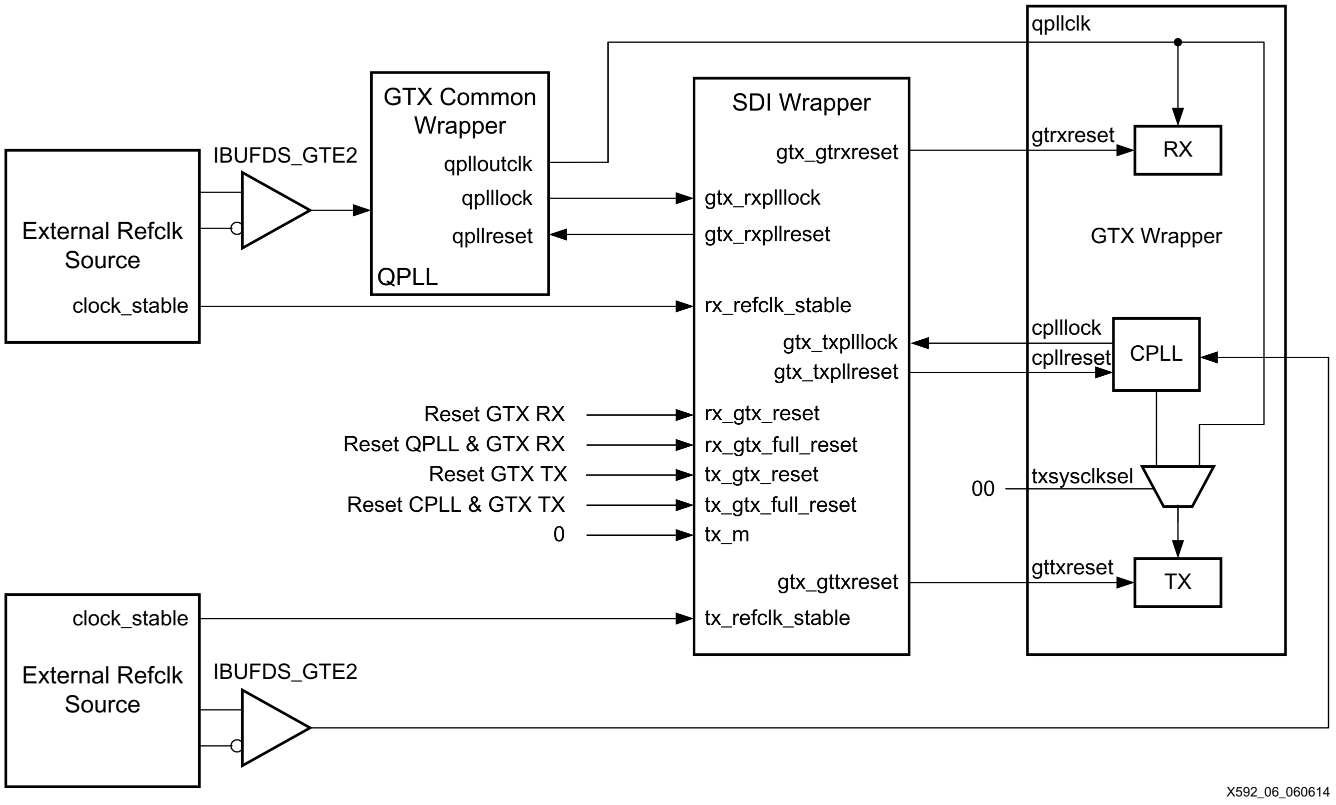


Figure 6: **PLL Usage Model 2**

Usage Model 3: Multiple Transceivers Are Active in the Quad, All RX Clocked by the QPLL, Each TX Dynamically Switched between the QPLL and the CPLL

In this usage model, shown in Figure 7, there are multiple transceivers active in the Quad. All GTX receivers are clocked by the QPLL. All of the GTX transmitters can be independently switched between the QPLL and their own CPLL. All of the CPLLs use the same reference clock. This model conforms to the standard usage model shown in Figure 3.

In this usage model, one SDI wrapper is chosen as the QPLL master and controls the gtx_qpllreset port of the GTX common wrapper. The other SDI wrappers do not control the QPLL reset, but they do monitor the qplllock output of the GTX common wrapper so that the reset sequences of the GTX transceiver do not proceed until the QPLL is locked.

The following connections must be made:

- The gtx_rxpllreset output of the SDI wrapper designated as the QPLL master must be connected to the qpllreset port of the GTX common wrapper. The gtx_rxpllreset outputs of the other SDI wrappers in the Quad are left unconnected.
- The gtx_txpllreset output of each SDI wrapper must be connected to the associated GTX wrapper cpllreset port.
- The gtx_rxpllock input of every SDI wrapper must be driven by the qplllock output of the GTX common wrapper.
- The gtx_txplllock input of each SDI wrapper must be driven by the logical OR of the qplllock output of the GTX common wrapper and the associated GTX wrapper cplllock output.
- The rx_refclk_stable input of the QPLL master SDI wrapper must be asserted High only when the reference clock source to the QPLL is stable. The rx_reflk_stable inputs of the other SDI wrappers must be permanently wired High.
- The tx_refclk_stable input of each SDI wrapper must be asserted High only when the CPLL reference clock source is stable.
- The tx_m input port of each SDI wrapper controls dynamic switching of the TX serial clock source in the associated GTX transceiver by controlling the gtx_txsysclksel output of the SDI wrapper which must be connected to the txsysclksel port of the associated GTX wrapper.
- The TX_CLK0_QPLL and TX_CLK1_QPLL parameters of each SDI wrapper must be set appropriately based on how the tx_m port is used to select between the QPLL and the CPLL. Typically, TX_CLK0_QPLL is set to 1 and TX_CLK1_QPLL is set to 0. This configures tx_m to select the QPLL as the TX serial clock source when tx_m is Low and the CPLL when tx_m is High.
- When the QPLL needs to be reset due to a reference clock change or interruption, assert the rx_gtx_full_reset input of all SDI wrappers. The QPLL master SDI wrapper resets the QPLL and all GTX RX units are reset. Also, assert the tx_gtx_reset input of all SDI wrappers to reset the GTX TX units.
- When the CPLL reference clock source changes or is interrupted, all CPLLs using that reference clock must be reset by asserting the tx_gtx_full_reset port of all the SDI wrappers.

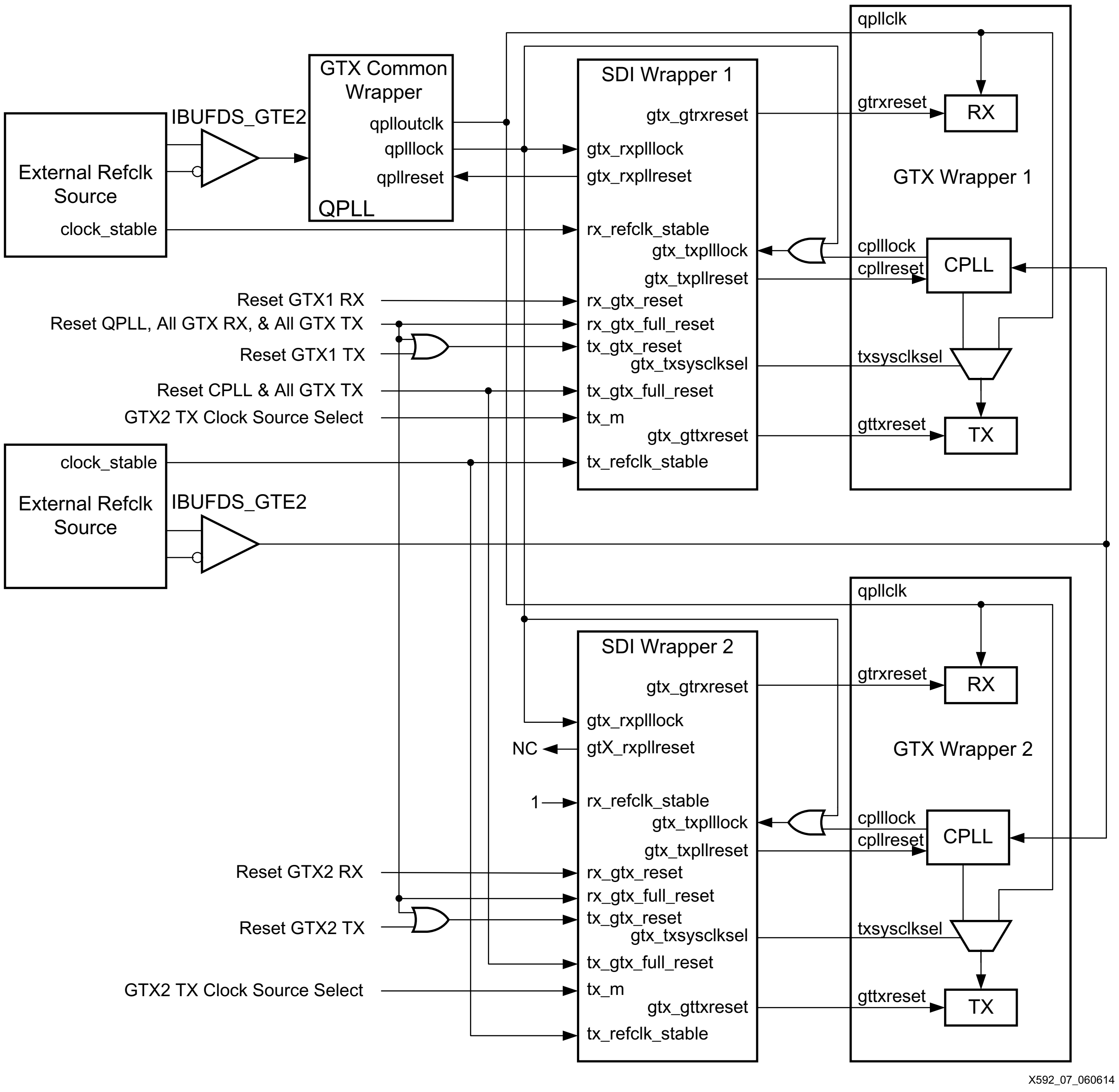


Figure 7: **PLL Usage Model 3**

Usage Model 4: Multiple Transceivers Are Active in a Quad, All RX Use the QPLL, All TX Use Their Own CPLL

In this usage model, shown in Figure 8 , there are multiple transceivers active in the Quad. All of the receivers are clocked by the QPLL. Each transmitter is clocked only by its associated CPLL. Each CPLL has its own reference clock source.

This usage model covers a very common case where multiple transceivers are active in the Quad, all implementing SDI interfaces. All the active GTX RX units in the Quad use the serial clock from the QPLL. All the GTX TX units use the serial clock from their associated CPLLs. In this usage model, one SDI wrapper is designated as the QPLL master and controls the gtx_qpllreset port of the GTX common wrapper. The other SDI wrappers do not control the QPLL reset, but they do monitor the QPLL locked output of the GTX common wrapper.

The following connections must be made:

- The gtx_rxpllreset output of the QPLL master SDI wrapper must be connected to the qpllreset port of the GTX common wrapper. The gtx_rxpllreset outputs of the other SDI wrapper are left unconnected.
- The gtx_txpllreset output of each SDI wrapper must be connected to the cpllreset port of the associated GTX wrapper.
- The gtx_rxplllock input of every SDI wrapper must be driven by the qplllock output of the GTX common wrapper.
- The gtx_txplllock input of each SDI wrapper must be driven by the cplllock output of the associated GTX wrapper.
- The rx_refclk_stable input of the QPLL master SDI wrapper must be asserted High only when the reference clock source to the QPLL is stable. The rx_refclk_stable inputs of the other SDI wrappers must be wired High.
- The tx_refclk_stable input of each SDI wrapper must be asserted High only when the reference clock source to the associated transceiver CPLL is stable.
- The txsysclksel port of each GTX wrapper must be wired to `2'b00` to permanently select the CPLL as the TX serial clock source. The gtx_txsysclksel output port of the SDI wrapper is left unconnected.
- The TX_CLK0_QPLL and TX_CLK1_QPLL parameters of every SDI wrapper must be set to 0.
- The tx_m input port of every SDI wrapper is not used and should be wired Low.
- When the QPLL needs to be reset due to a reference clock change or interruption, assert the rx_gtx_full_reset input of every SDI wrapper to reset both the QPLL and all of the GTX RX.
- When the CPLL of a particular transceiver needs to be reset due to a reference clock change or interruption, assert the tx_gtx_full_reset input of the associated SDI wrapper to reset both the CPLL and the GTX TX.

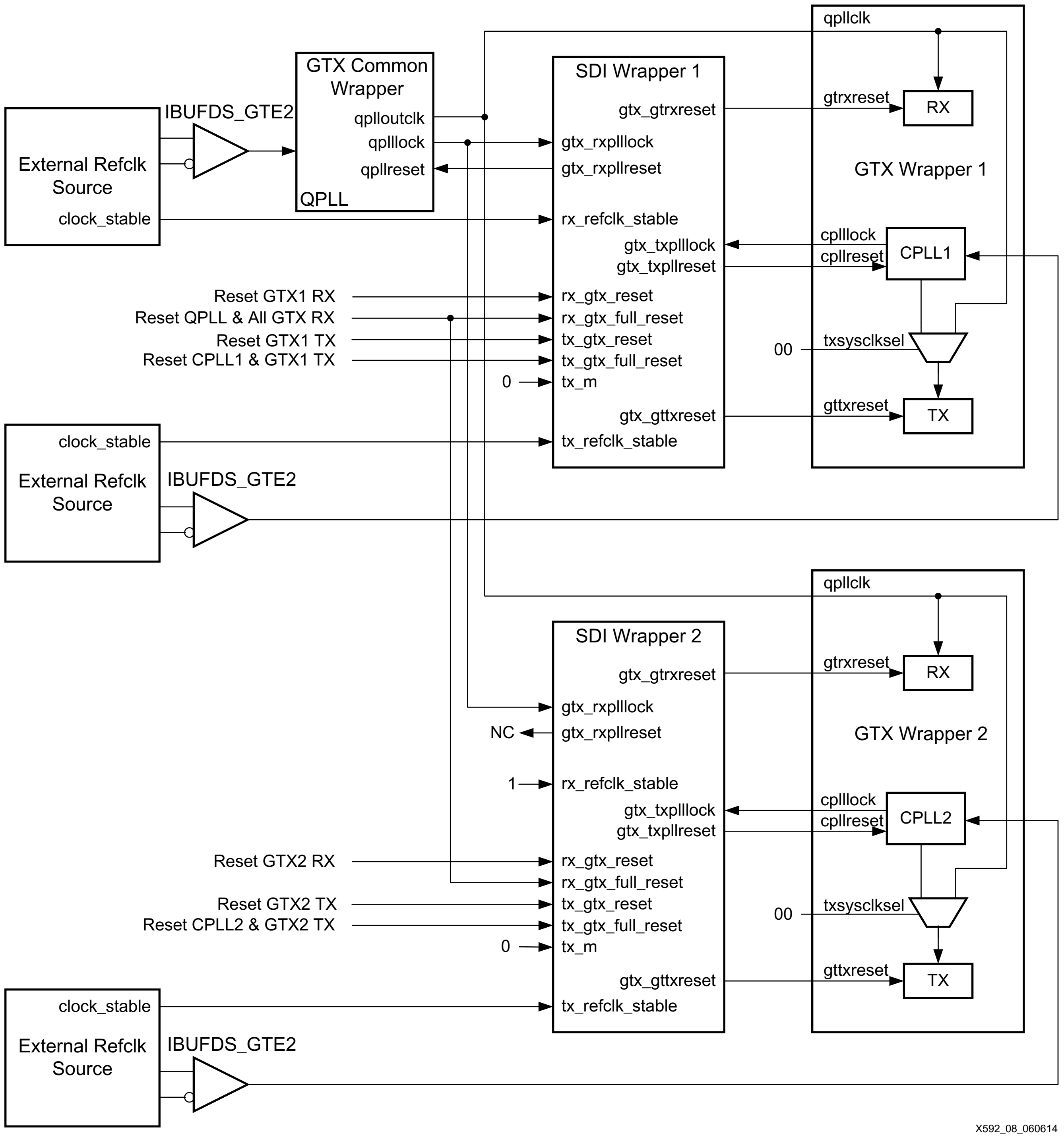


Figure 8: **PLL Usage Model 4**

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 9.

Important: The capacitance values of the AC coupling capacitors between the outputs of the external SDI cable equalizer and the serial inputs of the GTX RX must be large enough to pass the SDI pathological signals without significant signal droop. AC coupling capacitors with values of at least 1.0 µF are required and 4.7 µF capacitors are recommend. Some new generation SDI cable equalizers default to 600 mV differential swing on their outputs instead of the traditional 800 mV differential swing. When using an equalizer with 600 mV differential swing, even larger AC coupling capacitors might be required. With a cable equalizer set for 600 mV differential output swing, 10 µF AC coupling capacitors might be required for reliable reception of SD-SDI signals. It is recommended that cable equalizers be configured with 800 mV differential swing.

The differential inputs of the GTX RX have built-in differential termination. As described in *7 Series GTX/GTX Transceivers User Guide* (UG476) [Ref 1], 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.

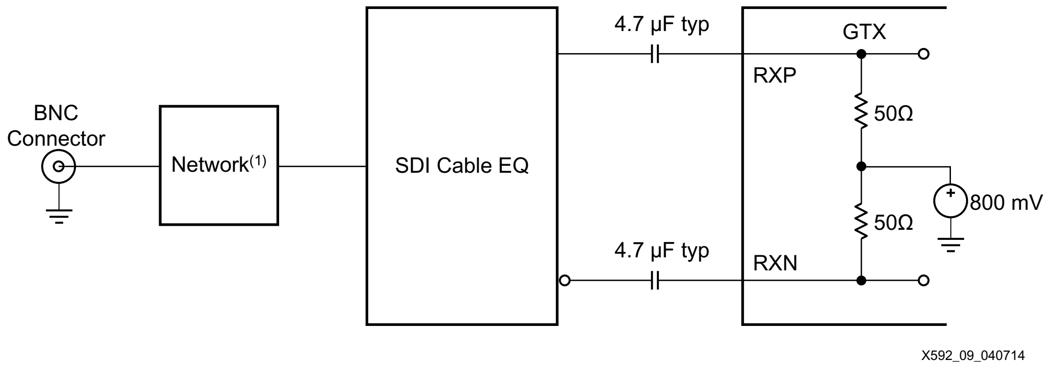


Figure 9: **Interfacing an SDI Cable Equalizer to the GTX Receiver Inputs**

Notes relevant to Figure 9:

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 10. 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.

Important: The capacitance values of the AC coupling capacitors between the GTX TX serial outputs and the inputs of the SDI cable driver must be large enough to pass the SDI pathological signals without significant signal droop. AC coupling capacitors with values of at least 1.0 µF are required and 4.7 µF capacitors are recommended.

Note: The capacitance values of the AC coupling capacitors between the GTX TX serial outputs and the inputs of the SDI cable driver must be large enough to pass the SDI pathological signals without significant signal droop. AC coupling capacitors with values of at least 1.0 µF are required and 4.7 µF capacitors are recommended.

Caution! The capacitance values of the AC coupling capacitors between the GTX TX serial outputs and the inputs of the SDI cable driver must be large enough to pass the SDI pathological signals without significant signal droop. AC coupling capacitors with values of at least 1.0 µF are required and 4.7 µF capacitors are recommended.

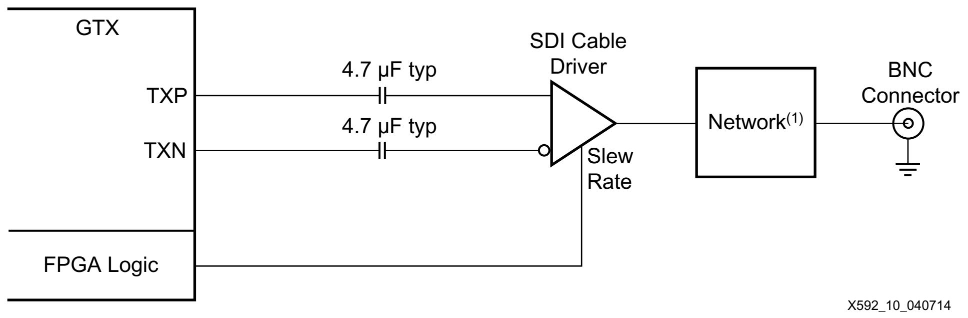


Figure 10: **Interfacing an SDI Cable Driver to the GTX Transmitter Outputs**

Notes relevant to Figure 10:

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 clock and data recovery (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 the Xilinx application note *Dynamically Programmable DRU for High-Speed Serial I/O* (XAPP875) [Ref 2] that has been optimized for recovering 270 Mb/s SD-SDI bit streams from 11X oversampled data. The general purpose DRU described in XAPP875 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.

SMPTE ST 259 (the SD-SDI standard) [Ref 3] 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* (XAPP875) [Ref 2]. 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. Note that the 540 Mb/s SD-SDI bit rate specified by SMPTE ST 344 [Ref 4] 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 XAPP875 DRU to receive the 540 Mb/s ST 344 signal using 5.5X oversampling so that the standard SDI reference clock frequency can be used.

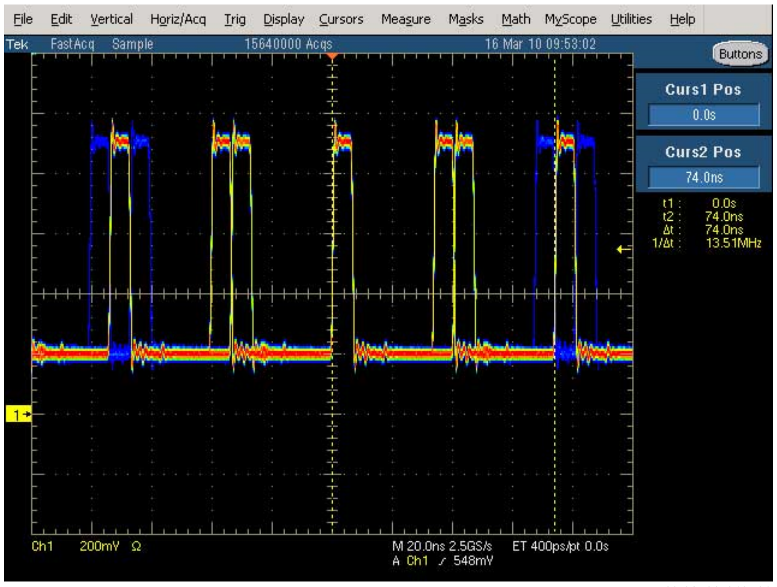
Receiving the additional SD-SDI bit rates also requires modifications to the SDI RX rate detector that controls the locking of the SDI RX by searching sequentially through all SDI bit rates until the receiver locks. The rate detection algorithm is implemented in the `triple_sdi_rx_autorate.v` or `triple_sdi_rx_autorate.vhd` file supplied with the SMPTE SD/HD/3G-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 11 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.

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. And 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, occasionally, the DRU 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 file `dru_sim.v` the is included with this application note, provides a simplified simulation model of the DRU. This file can be used during simulation to replace the `dru.ngc` file. However, do not use this simulation model in a design intended for use in the actual FPGA as this model does not support any variation in frequency between the GTX RX reference clock and the SD-SDI bit stream.


File Edit Vertical Horiz/Acq Trig Display Cursors Measure Masks Math MyScope Utilities Help
Tek FastAcq Sample 15640000 Acqs 16 Mar 10 09:53:02
Buttons
Curs1 Pos
0.0s
Curs2 Pos
74.0ns
t1 : 0.0s
t2 : 74.0ns
Δt : 74.0ns
1/Δt : 13.51MHz
Ch1 200mV Ω
M 20.0ns 2.5GS/s ET 400ps/pt 0.0s
A Ch1 ⟋ 548mV

X592_11_040714

Figure 11: **Oscilloscope Capture of SD-SDI Clock Enable**

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, then 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. Please contact Xilinx technical support for inquiries about the PICXO technique and SDI.) The bandwidth of the mixed-mode clock manager (MMCM) in the Kintex-7 FPGA 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 Texas Instruments 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 a 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. And, 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 option 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 FPGA 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 Kintex-7 FPGA MMCM. But, there is a way to generate a recovered SD-SDI clock using a spare GTX transmitter. (See Figure 12.)

The control module's recclk_txdata port can 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 Kintex-7 FPGA, with appropriate care to properly terminate the GTX transceiver TX outputs and translate them to LVDS. Then the 270 MHz clock can be used in whatever manner is required in the FPGA. 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.

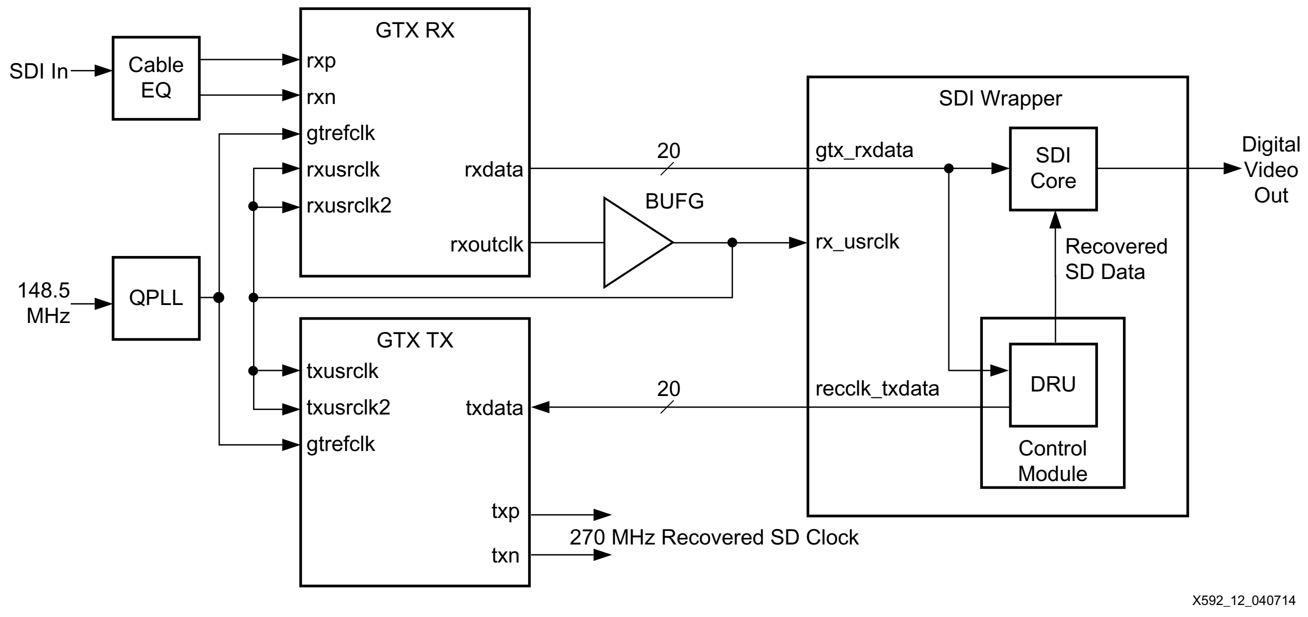


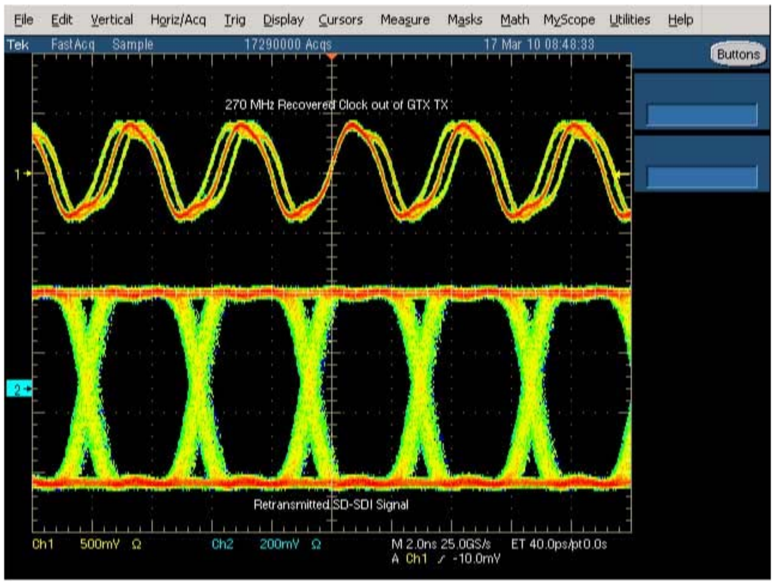
Figure 12: **Using a GTX TX to Generate an SD-SDI Recovered Clock**

The GTX TX that is used to generate the recovered SD-SDI clock does not have to be configured for SDI. It just needs to be configured to always run a 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 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 337ps). This can be seen in Figure 13. 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 13 is the SD-SDI that is being retransmitted by another GTX TX.

Note that 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, 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. And, it indicates which SDI mode the RX is locked to on its sdi_mode output port.



X592_13_040714

Figure 13: **Recovered SD-SDI Clock from GTX Transceiver**

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. And 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.

Note that 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 Kintex-7 FPGA

There are several steps required to implement an SDI interface in a Kintex-7 FPGA design. Those steps are:

1. Generate a GTX wrapper and GTX common wrapper using the 7 Series FPGAs Transceivers Wizard.
2. Generate the SMPTE SD/HD/3G-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. With this version of the application note, the top level SDI wrapper includes instances of a single GTX transceiver wrapper along with the SDI core and the SDI control module, making it easier to instance a complete SDI interface. The GTX common wrapper does need to be instantiated separately and connected to all SDI wrappers in the same Quad.
4. Add the `dru.ngc` file to the Vivado tools project (see the `readme.txt` file in `xapp592.zip` for more information).
5. Apply proper timing constraints for the SDI interface.

Generating the GTX Wrapper

Use the 7 Series FPGAs Transceivers Wizard to generate a GTX wrapper. Beginning with version 3.0 of the wizard, the GTXE2_COMMON block is not included in the GTX wrapper, but is instanced in a separate wrapper called the GTX common wrapper.

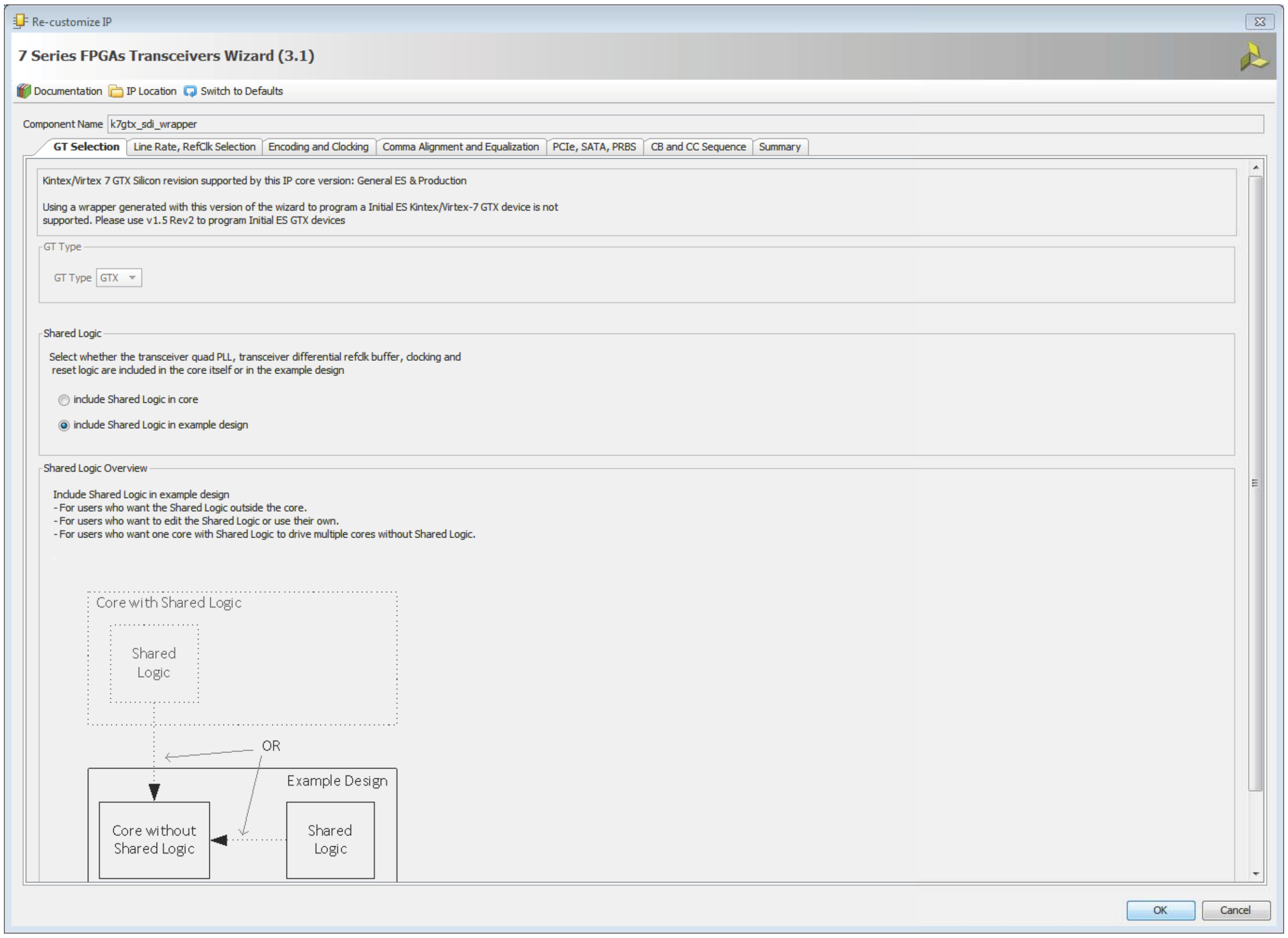
The GTX wrapper generated by the wizard is actually a hierarchy of wrapper levels. Beginning with version 3.0 of the wizard, the upper level wrappers contain additional reset logic that is not compatible with SDI operation. So, only the lowest level GTX wrapper file is actually useful for SDI applications. The lowest level GTX wrapper always contains a single GTXE2_CHANNNEL instance. The easiest way to generate and use the GTX wrapper is to use the wizard to generate just a single transceiver and then instantiate the lowest level GTX wrapper multiple times in the application, once for each GTX transceiver that is used for SDI. This version of the application note has the GTX wrapper instantiated in the SDI wrapper, but you should always check that this instance matches the GTX wrapper that is generated by the wizard. Ports on the GTX wrapper tend to differ slightly between versions of the wizard. Also, the GTX common wrapper must be instantiated as many times as necessary, once for each GTX Quad containing transceivers implementing SDI interfaces. If only the CPLL is being used for serial clocks to the GTX transceivers, the GTX common wrapper does not need to be instantiated at all because it only contains the QPLL. The two SDI demonstration applications supplied with this application note provide examples of how to instantiate the GTX wrapper and the GTX common wrapper.

The following information details exactly the steps required to generate the GTX wrapper using the wizard version 3.1 from the Vivado IP catalog.

Because the top level GTX wrapper is not used in the SDI application, it is best not to generate the GTX wrapper in the same Vivado project as the SDI application. Run Vivado tools and create a new project just for the purpose of generating the GTX wrapper for SDI. After the GTX wrapper is generated, only those GTX wrapper files that are needed for SDI should be added to the actual SDI Vivado project. Always specify the same Kintex-7 FPGA device in the GTX wrapper Vivado project and in the SDI Vivado project.

After creating the GTX wrapper Vivado project, open the IP catalog. The 7 Series FPGAs Transceivers Wizard is found in the I/O Interfaces folder in the top-level FPGA Features and Design folder of the Vivado IP catalog. Find the wizard in the IP catalog and double-click to launch the wizard.

The wizard launches with the GT Selection tab open as shown in Figure 14. Above the tabs is a text field called Component Name. The name entered here is used as the name for the GTX wrapper file and the name of the GTX component. In this example, the component name is **k7gtx_sdi_wrapper.** This matches the GTX wrapper name used in the two demonstrations applications and is also the name of the GTX wrapper instance in the SDI wrapper. For compatibility with that SDI wrapper, use **k7gtx_sdi_wrapper** as the GTW wrapper name, otherwise the GTX wrapper instance name in the SDI wrapper needs to be edited.



Component Name k7gtx_sdi_wrapper

Kintex/Virtex 7 GTX Silicon revision supported by this IP core version: General ES & Production

Using a wrapper generated with this version of the wizard to program a Initial ES Kintex/Virtex-7 GTX device is not supported. Please use v1.5 Rev2 to program Initial ES GTX devices

GT Type

GT Type GTX

Shared Logic

Select whether the transceiver quad PLL, transceiver differential refclk buffer, clocking and reset logic are included in the core itself or in the example design

include Shared Logic in core [✓ include Shared Logic in example design]

Shared Logic Overview

Include Shared Logic in example design
- For users who want the Shared Logic outside the core.
- For users who want to edit the Shared Logic or use their own.
- For users who want one core with Shared Logic to drive multiple cores without Shared Logic.

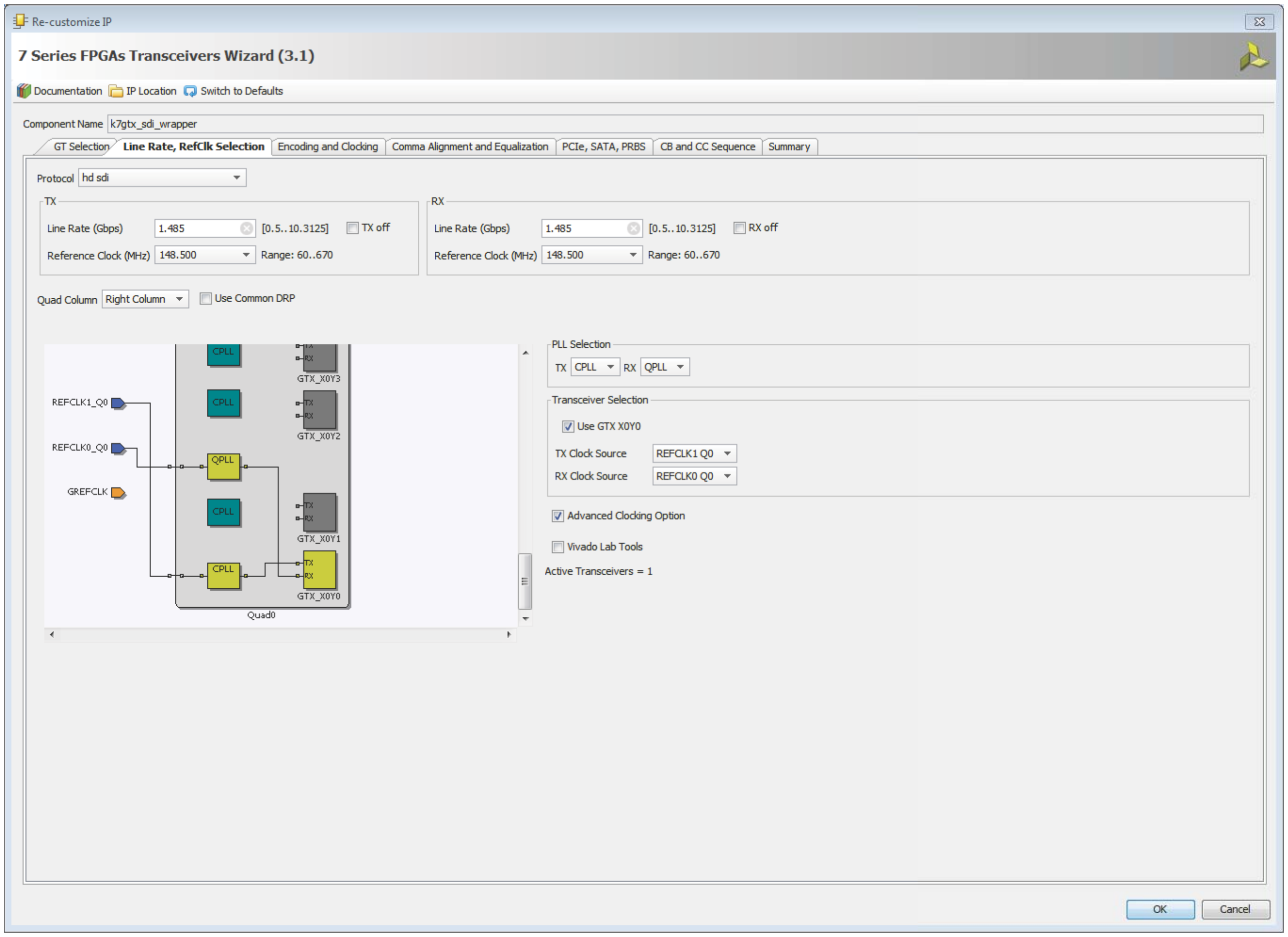
X592_14_040814

Figure 14: **7 Series FPGAs Transceivers Wizard - GT Selection Tab**

Select the type of transceiver used in the GT Type pull-down menu in the GT Selection tab., the type of transceiver used must be specified. For Kintex-7 devices, only the GTX transceiver is available, so only GTX transceivers can be selected and the GT Type selection menu is grayed out in Figure 14.

In the **Shared Logic** section, select **Include Shared Logic in example design.** When moving from tab to tab, click the tabs located under the Component Name field. Do not click OK until all tabs have been correctly set up. The **OK** button closes the wizard.

Select the **Line Rate, RefClk Selection** tab, shown in Figure 15. In the **Protocol** pull-down menu, select **hd sdi.** This sets most options in the wizard to the correct values for operation with the SDI core. Choose **hd sdi** even if creating a wrapper to be used with some other SDI protocol such as 3G-SDI or SD-SDI. The SDI core expects the GTX wrapper to be created using the **hd sdi** protocol settings and dynamically changes ports and attributes properly when other SDI protocols are selected during operation.



Re-customize IP

7 Series FPGAs Transceivers Wizard (3.1)

Documentation | IP Location | Switch to Defaults

Component Name: k7gtx_sdi_wrapper

GT Selection | Line Rate, RefClk Selection | Encoding and Clocking | Comma Alignment and Equalization | PCIe, SATA, PRBS | CB and CC Sequence | Summary

Protocol: hd sdi

TX
- Line Rate (Gbps): 1.485 [0.5..10.3125] ☐ TX off
- Reference Clock (MHz): 148.500 Range: 60..670

RX
- Line Rate (Gbps): 1.485 [0.5..10.3125] ☐ RX off
- Reference Clock (MHz): 148.500 Range: 60..670

Quad Column: Right Column ☐ Use Common DRP

PLL Selection
- TX: CPLL RX: QPLL

Transceiver Selection
- ☑ Use GTX X0Y0
- TX Clock Source: REFCLK1 Q0
- RX Clock Source: REFCLK0 Q0

☑ Advanced Clocking Option

☐ Vivado Lab Tools

Active Transceivers = 1

OK | Cancel

x592_15_061314

Figure 15: **7 Series FPGAs Transceivers Wizard - Line Rate, RefClk Selection Tab**

As shown in Figure 15, the **Line Rate** for both the TX and RX should already be set to 1.485 Gb/s after the **hd sdi** protocol has been selected. These line rates must be used in the wizard no matter what SDI bit rates will actually be supported in the SDI application. Set the **Reference Clock** frequency for both the TX and RX to the desired value, typically 148.5 MHz. This sets the line rate to 1.485 Gb/s and the reference clock frequency to 148.5 MHz for both RX and TX. Do not change the line rate to 1.485/1.001 Gb/s or the reference clock frequency to 148.5/1.001 MHz. The SDI control module takes care of switching to the 1/1.001 rates from the 1/1 rates. The control module also takes care of dynamically switching to the other line rates of 2.97 Gb/s for 3G-SDI and 270 Mb/s for SD-SDI. The line rate specified on this tab should always be 1.485 Gb/s. Alternative reference clock frequencies can be chosen on this tab, but only choose from those that are available in the Reference Clock pull-down lists.

The TX off and RX off check boxes allow the creation of GTX wrappers with only transmitters (by selecting **RX off**) or only receivers (by selecting **TX off**). In this example, neither of these options is selected.

The Quad column does not matter in this case, so just leave it to its default value.

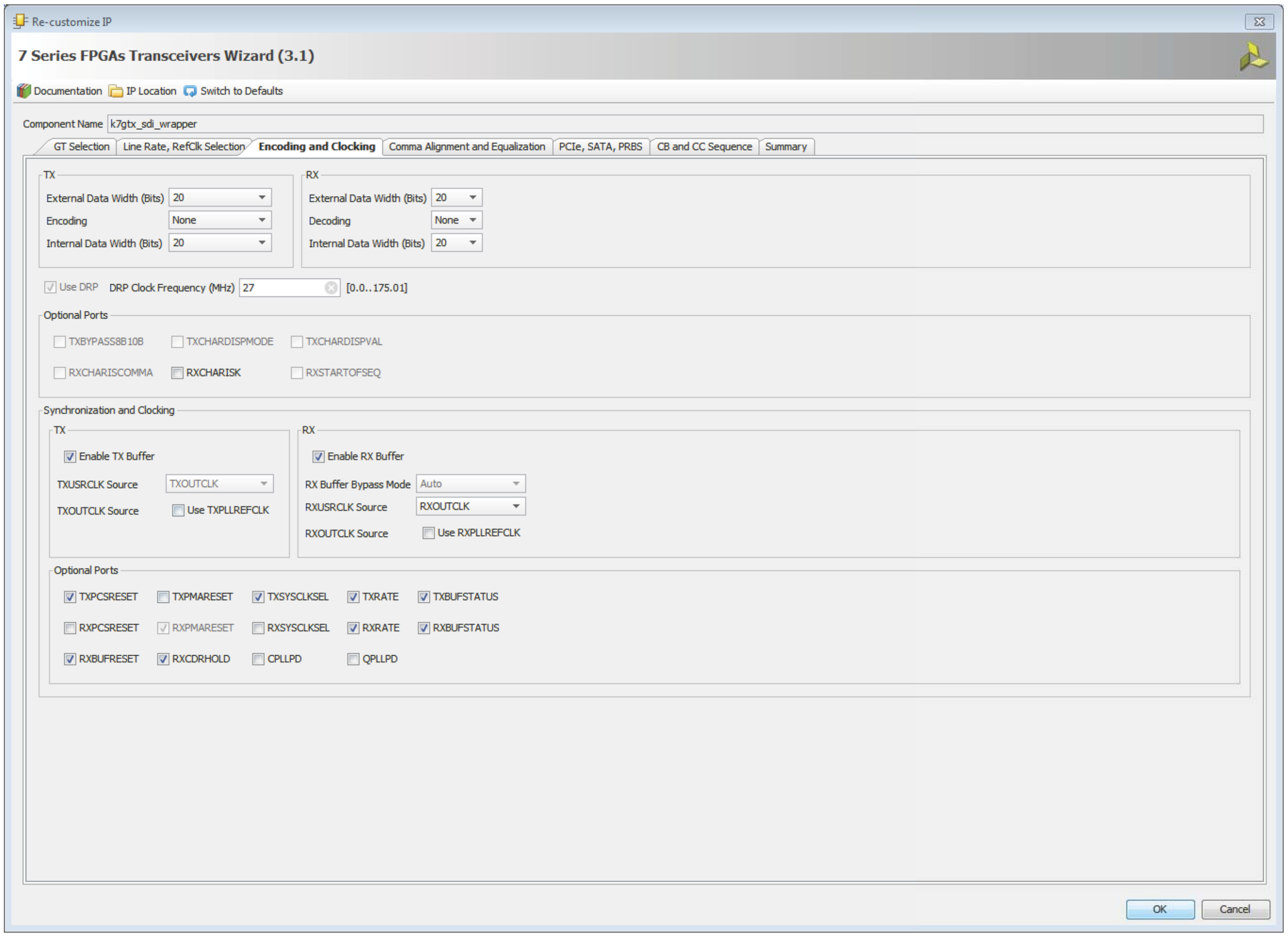
Use Common DRP is usually not selected for SDI applications.

The bottom section of the **Line Rate, RefClk Selection** tab allows you to choose which GTX transceivers and Quads are included in the top-level GTX wrapper. It also allows you to choose the reference clocks used by the PLLs and which PLL supplies the serial clock to each transceiver. For SDI applications, always generate a GTX wrapper with a single GTX transceiver. It does not matter which transceiver is selected and using the single transceiver that is selected by default is the easiest.

In this example, the RX unit is clocked by the QPLL which uses REFCLK0 Q1 as its reference clock. The TX unit is clocked by the CPLL referenced to REFCLK1 Q1. The wizard does not explicitly handle the case where TX units are dynamically switched between the QPLL and the CPLL. The SDI control module takes care of the control for this dynamic switching. But, to build a GTX wrapper with all the PLLs active and connected properly for dynamic switching of the TX between the QPLL and the CPLL, assign the QPLL as the RX clock source and the CPLL as the TX clock source, and assign different reference clocks to the QPLL and the CPLL as shown in Figure 15. In cases where the QPLL is not being used and only the CPLL is used, use the CPLL as the reference clock source to both the RX and the TX units.

Enable the **Advanced Clocking Option**.

Select the **Encoding and Clocking** tab, shown in Figure 16.



Component Name: k7gtx_sdi_wrapper

TX
- External Data Width (Bits): 20
- Encoding: None
- Internal Data Width (Bits): 20

RX
- External Data Width (Bits): 20
- Decoding: None
- Internal Data Width (Bits): 20

[✓] Use DRP   DRP Clock Frequency (MHz): 27 [0.0..175.01]

Optional Ports
- [ ] TXBYPASS8B10B   [ ] TXCHARDISPMODE   [ ] TXCHARDISPVAL
- [ ] RXCHARISCOMMA   [ ] RXCHARISK   [ ] RXSTARTOFSEQ

Synchronization and Clocking

TX
- [✓] Enable TX Buffer
- TXUSRCLK Source: TXOUTCLK
- TXOUTCLK Source: [ ] Use TXPLLREFCLK

RX
- [✓] Enable RX Buffer
- RX Buffer Bypass Mode: Auto
- RXUSRCLK Source: RXOUTCLK
- RXOUTCLK Source: [ ] Use RXPLLREFCLK

Optional Ports
- [✓] TXPCSRESET   [ ] TXPMARESET   [✓] TXSYSCLKSEL   [✓] TXRATE   [✓] TXBUFSTATUS
- [ ] RXPCSRESET   [✓] RXPMARESET   [ ] RXSYSCLKSEL   [✓] RXRATE   [✓] RXBUFSTATUS
- [✓] RXBUFRESET   [✓] RXCDRHOLD   [ ] CPLLPD   [ ] QPLLPD

X592_16_040814

Figure 16: **7 Series FPGAs Transceivers Wizard - Encoding and Clocking Tab**

For both the TX and the RX section, the **External Data Width** must be 20 and the **Internal Data Width** must be **20**. The TX **Encoding** and the RX **Decoding** must be **None**.

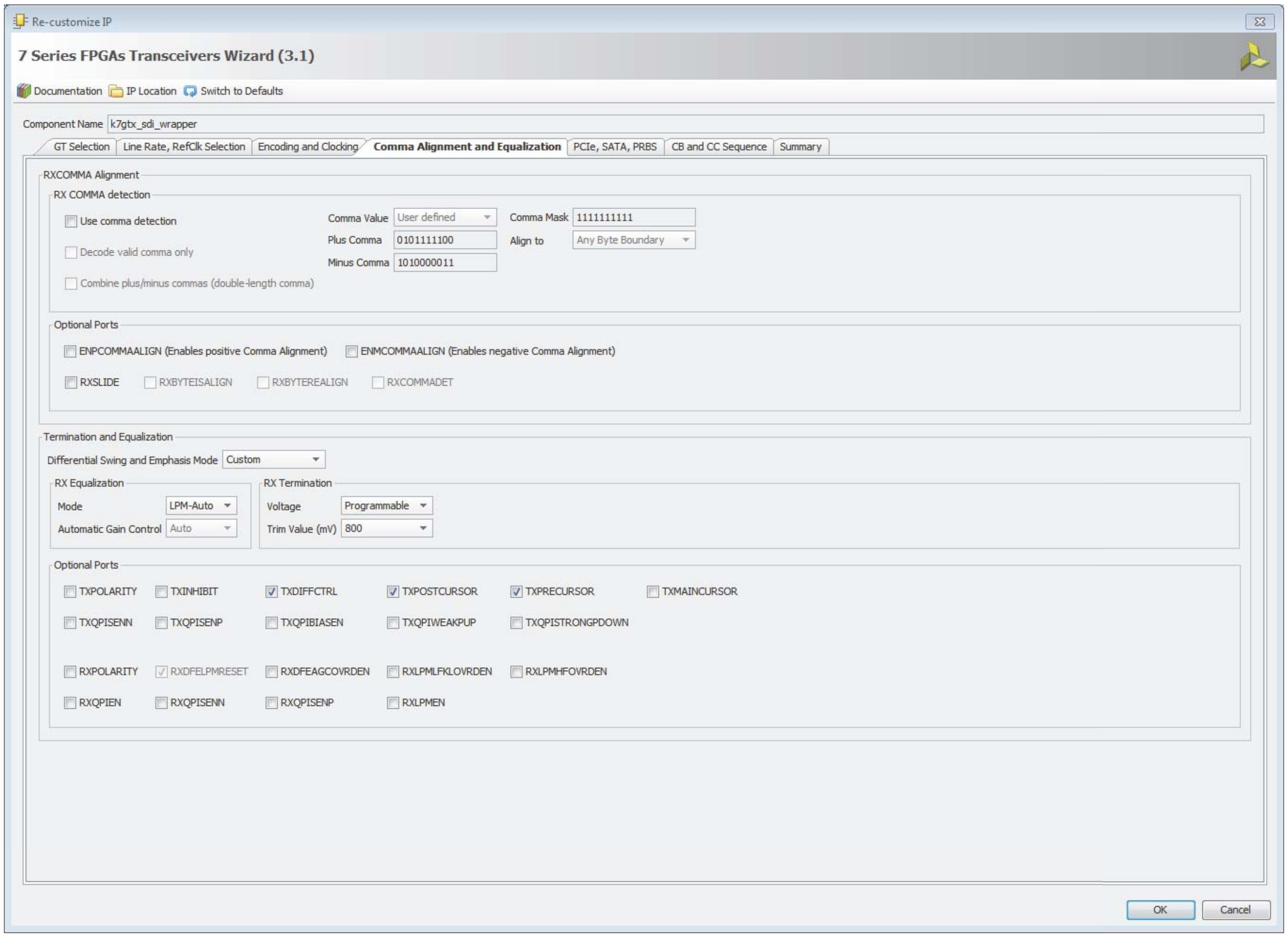
Use DRP is always selected and cannot be deselected. Set the DRP frequency to the nominal frequency of the clock connected to the GTX drpclk port.

None of the optional ports in the top section, under the DRP frequency selection, are required for SDI.

It is highly recommended that the RX and TX buffers be used for SDI applications. Thus, you should select **Enable TX Buffer** and **Enable RX Buffer**. The **TXUSRCLK Source** is set to **TXOUTCLK** and cannot be changed. The **RXUSRCLK Source** must be set to **RXOUTCLK**, as shown in Figure 16.

In the bottom Optional Ports section, the following ports are required for SDI applications: TXPCSRESET, TXRATE, TXBUFSTATUS, RXRATE, RXBUFSTATUS, RXBUFRESET, and RXCDRHOLD. If the application requires that the TX units dynamically switch between the QPLL and the CPLL, then the TXSYSCLKSEL port is also required. It is recommended that the TXSYSCLSEL port always be selected and, if dynamic switching of the TX is not required, the TXSYSCLKSEL port can be hardwired to select either the QPLL or the CPLL as the serial clock source.

Select the **Comma Alignment and Equalization** tab, shown in Figure 17. In the **RXCOMMA Alignment** section of this tab, **Use COMMA detection** and the **RXSLIDE** port must not be selected. Comma detection and the RXSLIDE features are not used for SDI.



Component Name: k7gtx_sdi_wrapper

RXCOMMA Alignment

RX COMMA detection

- [ ] Use comma detection
- [ ] Decode valid comma only
- [ ] Combine plus/minus commas (double-length comma)

Comma Value	User defined	Comma Mask	1111111111
Plus Comma	0101111100	Align to	Any Byte Boundary
Minus Comma	1010000011		

Optional Ports

- [ ] ENPCOMMAALIGN (Enables positive Comma Alignment)
- [ ] ENMCOMMAALIGN (Enables negative Comma Alignment)
- [ ] RXSLIDE
- [ ] RXBYTEISALIGN
- [ ] RXBYTEREALIGN
- [ ] RXCOMMADET

Termination and Equalization

Differential Swing and Emphasis Mode: Custom

RX Equalization		RX Termination	
Mode	LPM-Auto	Voltage	Programmable
Automatic Gain Control	Auto	Trim Value (mV)	800

Optional Ports

☐ TXPOLARITY	☐ TXINHIBIT	✓ TXDIFFCTRL	✓ TXPOSTCURSOR	✓ TXPRECURSOR	☐ TXMAINCURSOR
☐ TXQPISENN	☐ TXQPISENP	☐ TXQPIBIASEN	☐ TXQPIWEAKPUP	☐ TXQPISTRONGPDOWN	
☐ RXPOLARITY	✓ RXDFELPMRESET	☐ RXDFEAGCOVRDEN	☐ RXLPMLFKLOVRDEN	☐ RXLPMHFOVRDEN	
☐ RXQPIEN	☐ RXQPISENN	☐ RXQPISENP	☐ RXLPMEN		

Figure 17: **7 Series FPGAs Transceivers Wizard - Comma Alignment and Equalization Tab**

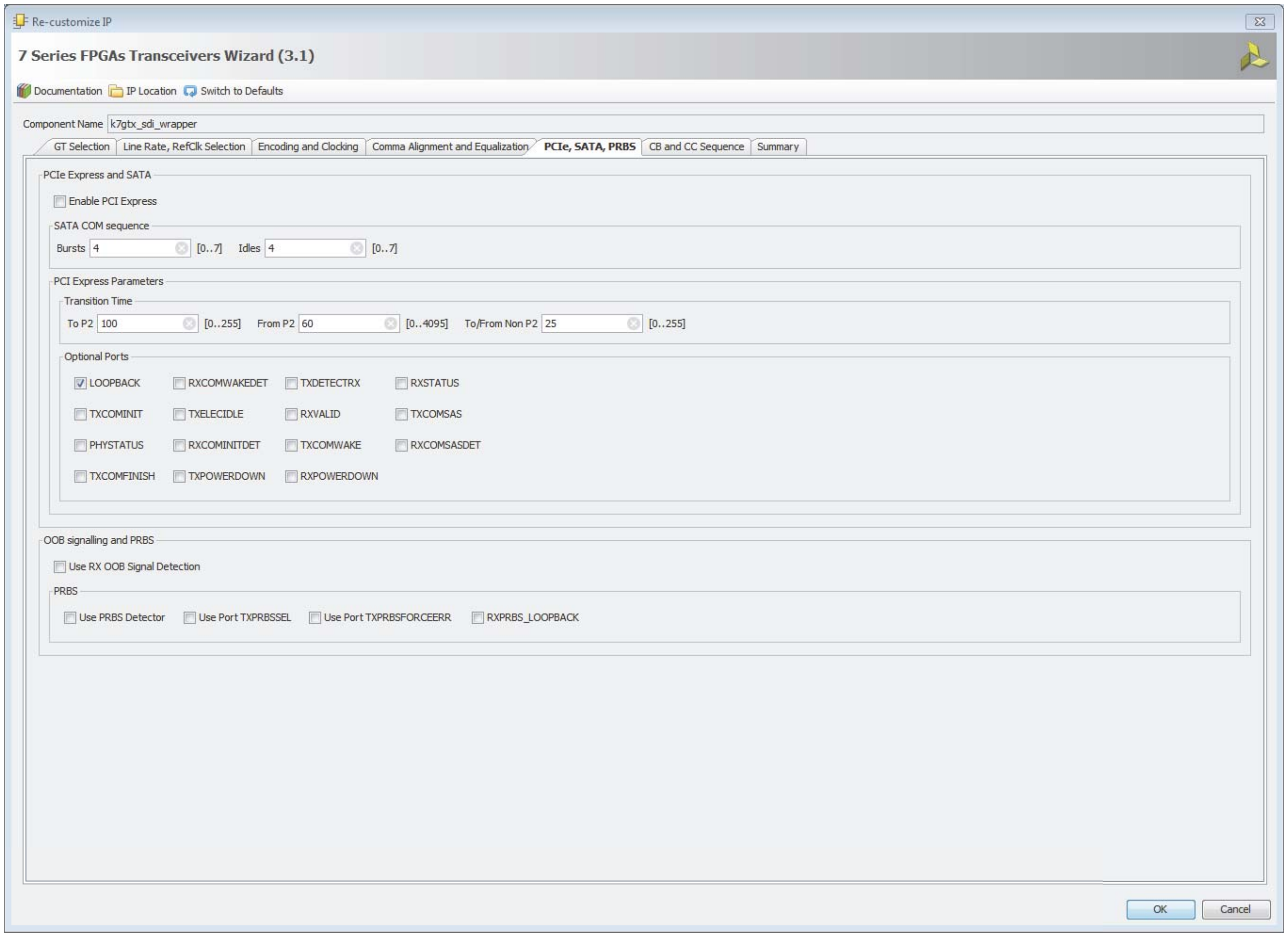
The settings in the **Termination and Equalization** section must be set to the values shown in Figure 17. The **Differential Swing and Emphasis Mode** must be set to **Custom, RX Equalization Mode** must be set to **LPM-Auto**, the **RX Termination Voltage** must be set to **Programmable**, and the **Trim Value** must be set to **800 mV**.

In the Optional Ports section, any of these ports can be enabled or disabled depending on the application requirements. The TXDIFFCTRL port is typically enabled, allowing the application to set the output swing of the TX to match the input voltage requirements of external SDI cable driver. The TXPOSTCURSOR and TXPRECURSOR ports can be selected if these ports are needed to improve the integrity of the signal from the TX to the external SDI cable driver.

Select the **PCIE, SATA, PRBS** tab, shown in Figure 18. Most of the options on this page are not relevant to SDI and should be left in their default values. There are a few ports in the Optional Ports section that can be useful for SDI applications.

The LOOPBACK port is selected by default. This port allows for dynamic selection of various loopback modes where the data being transmitted by the GTX TX is looped back to the GTX RX in the same transceiver. The loopback modes can be useful for debugging purposes, but generally are not used in production applications.

The TXPOWERDOWN and RXPOWERDOWN ports allow the TX and RX to be dynamically powered down to save power.


Re-customize IP

7 Series FPGAs Transceivers Wizard (3.1)

Documentation | IP Location | Switch to Defaults

Component Name: k7gtx_sdi_wrapper

Tabs: GT Selection | Line Rate, RefClk Selection | Encoding and Clocking | Comma Alignment and Equalization | **PCIe, SATA, PRBS** | CB and CC Sequence | Summary

PCIe Express and SATA

- [ ] Enable PCI Express

SATA COM sequence

Bursts: 4 [0..7] Idles: 4 [0..7]

PCI Express Parameters

Transition Time

To P2: 100 [0..255] From P2: 60 [0..4095] To/From Non P2: 25 [0..255]

Optional Ports

[✓] LOOPBACK	[ ] RXCOMWAKEDET	[ ] TXDETECTRX	[ ] RXSTATUS
[ ] TXCOMINIT	[ ] TXELECIDLE	[ ] RXVALID	[ ] TXCOMSAS
[ ] PHYSTATUS	[ ] RXCOMINITDET	[ ] TXCOMWAKE	[ ] RXCOMSASDET
[ ] TXCOMFINISH	[ ] TXPOWERDOWN	[ ] RXPOWERDOWN	

OOB signalling and PRBS

- [ ] Use RX OOB Signal Detection

PRBS

[ ] Use PRBS Detector [ ] Use Port TXPRBSSEL [ ] Use Port TXPRBSFORCEERR [ ] RXPRBS_LOOPBACK

OK | Cancel

UG592_18_040814

Figure 18: **7 Series FPGAs Transceivers Wizard - PCIe, SATA, PRBS Tab**

On the **CB and CC Sequence** tab, **Use Channel Bonding** and **Use Clock Correction** must not be selected. The **Summary** tab provides a summary of the selections made on the other tabs. To generate the GTX wrapper, click **OK** and then click **Generate** when the next menu opens.

The wizard generates several files, some that are required for an SDI application plus several other example files that should not be used for SDI. The files that are used all start with the component name that you assigned to the GTX wrapper in the wizard. The required files are:

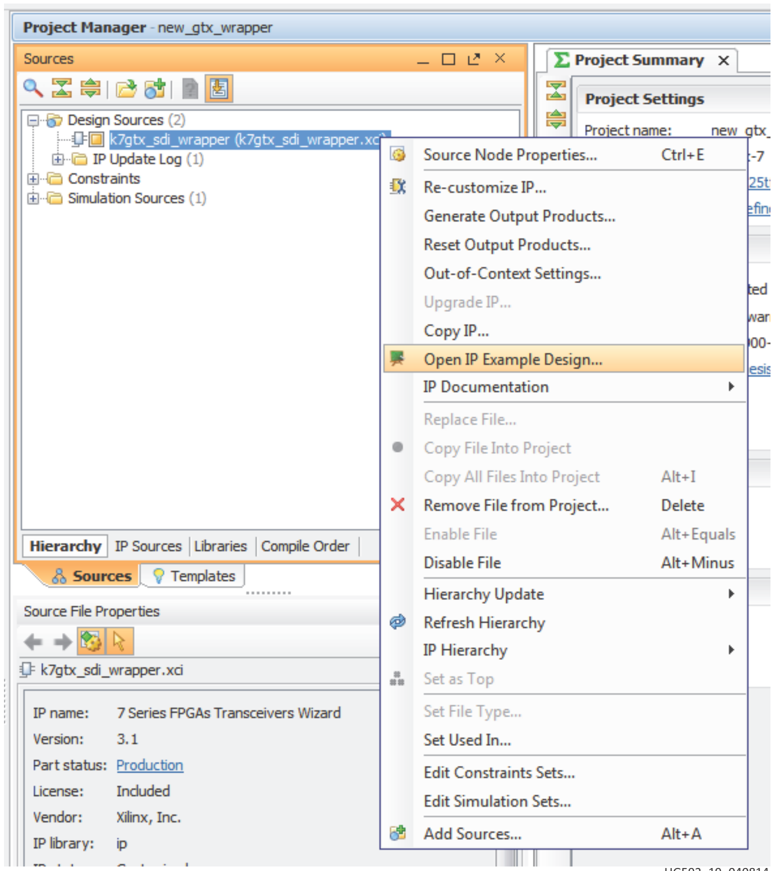
`<component_name>_gt.v/vhd`	lowest level GTX wrapper
`<component_name>_common.v/vhd`	GTX common wrapper

If the Vivado project name is gtx_wrapper, Verilog is selected as the default language, and the component name given to the GTX wrapper is k7gtx_sdi_wrapper, then the paths to the necessary files would be:

```
gtx_wrapper/k7gtx_sdi_wrapper_example.srcs/sources_1/ip/k7gtx_sdi_
wrapper/k7gtx_sdi_wrapper.v
```

```
gtx_wrapper/k7gtx_sdi_wrapper_example.srcs/sources_1/imports/suppo
rt/k7gtx_sdi_wrapper_common.v
```

The support directory and the GTX common wrapper located in the support directory are not automatically created when you generate the GTX wrapper using the wizard. You must right-click the SDI wrapper item in the **Sources** panel and choose **Open IP Example Design** as shown in Figure 19.



IP name:	7 Series FPGAs Transceivers Wizard
Version:	3.1
Part status:	Production
License:	Included
Vendor:	Xilinx, Inc.
IP library:	ip

Figure 19: **Generating the Support Directory**

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

Use the Vivado IP catalog to generate the SMPTE SD/HD/3G-SDI core. Do not use the older Triple-Rate SDI core. That core is only for Virtex®-6 FPGAs. The SMPTE SD/HD/3G-SDI core is the generic SDI core that works with 7 series FPGA 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. Note that 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 SDI wrapper. So, 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 wrapper and the SDI wrapper 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 GTX wrappter, 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_wrapper.v`. In addition to the GTX wrapper and the SDI core, it also instances the following files:

- `x7gtx_sdi_control.v` or `.vhd`
- `x7gtx_reset_control.v` or `.vhd`
- `x7gtx_sdi_drp_control.v` or `.vhd`
- `sdi_rate_detect.v` or `.vhd`
- `dru_bshift10to10.v` or `.vhd`
- `dru_maskencoder.v` or `.vhd`
- `dru_control.v` or `.vhd`
- `dru_rot20.v` or `.vhd`
- `dru.v` (for 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_wrapper.v` file, the `dru.v` file must be included in the project.

When using 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` file that is included with this application note in a design intended to be used in the actual FPGA. This file is for simulation purposes only. Using it in an actual hardware implementation results in the SDI receiver that is not able to correctly receive SD-SDI signals. For simulation purposes, the `dru_sim.v` file 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 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 SD/HD/3G-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 common 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 3G-SDI data ready signal (either tx_din_rdy or rx_dout_rdy_3G).

Most of 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* (PG071) [Ref 5].

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 is not correct and the RX bit rate detection circuit might not function.
drpclk	In	1	This input must be connected to a fixed-frequency free running clock. This clock is used to clock the DRP of the GTX and SDI control logic associated with the DRP. The period of this clock must be specified by the parameter DRPCLK_PERIOD. In most cases, the same clock can be used to drive the drpclk and clk ports.
qpllclk	In	1	If the QPLL is being used to clock the GTX transceiver, connect this port to the QPLLOUTCLK_OUT port of the GTX common wrapper. If the QPLL is not being used, tie this port Low.
qpllrefclk	In	1	If the QPLL is being used to clock the GTX transceiver, connect this port to the QPLLOUTREFCLK_OUT port of the GTX common wrapper. If the QPLL is not being used, tie this port Low.
qpllreset	Out	1	If the QPLL is being used to clock the GTX transceiver, connect this port to the QPLLRESET_IN port of the GTX common wrapper. This port can be left unconnected if the QPLL is not being used. Only one SDI wrapper can drive the GTX common wrapper's QPLLRESET_IN port. If multiple transceivers in a Quad are being used for SDI, one must be selected as the QPLL master and that wrapper must drive the GTX common wrapper QPLLRESET_IN port.
qplllock	In	1	If the QPLL is being used to clock the GTX, connect this port to the QPLLLOCKOUT_OUT port of the GTX common wrapper. If the QPLL is not being used, this port must be tied High. If multiple transceivers in a Quad are being used for SDI, connect the GTX common wrapper QPLLLOCK_OUT port to the qplllock_in port of all SDI wrappers in that Quad.
cpllrefclk	In	1	Connect this port to the O port of the IBUFDS_GTE2 that is providing the CPLL reference clock for this transceiver.
cplllock	Out	1	This output is High when the transceiver CPLL is locked to the reference clock.
rxp	In	1	This port is internally connected directly to the rxp port of the GTX receiver. It must be connected to an input port at the top level of the design.
rxn	In	1	This port is internally connected directly to the rxn port of the GTX receiver. It must be connected to an input port at the top level of the design.

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

Port Name	I/O	Width	Description
txp	In	1	This port is internally connected directly to the txp port of the GTX transmitter. It must be connected to an output port at the top level of the design.
txn	In	1	This port is internally connected directly to the txn port of the GTX transmitter. It must be connected to an output port at the top level of the design.
Receive Ports			
rx_rst	In	1	This port resets the RX portion of the SDI core, but does not reset the GTX RX. This synchronous reset input can normally be hardwired Low because a reset is not required. After FPGA configuration, the SMPTE SD/HD/3G-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_out	In	1	This output is driven internally by a BUFG. The input to the BUFG is driven by the rxoutclk port of the GTX transceiver. Unless otherwise indicated, all ports on the wrapper that are prefixed with rx_ are synchronous with this clock. The frequency of this clock is 148.5 MHz (or 148.5/1.001 MHz) for 3G-SDI and SD-SDI modes and 74.25 MHz (or 74.25/1.001 MHz) for SD-SDI mode.
rx_gtx_reset	In	1	When this input is asserted High, a full GTX RX reset sequence is initiated. First, if the qpllreset output of this module is connected to a GTX PLL reset input, that PLL is reset. After the PLL locks to the reference clock input, the GTX RX is reset using the GTX transceiver gtrxreset port. Completion of this reset sequence is indicated by assertion of the rx_change_done output. This signal connected to this input must be synchronous with the drpclk clock.
rx_gtx_reset	In	1	When this input is asserted High, the GTX RX is reset using the GTX transceiver gtrxreset port. If the PLL providing the serial clock to the GTX RX is not locked, the gtrxreset sequence does not complete until that PLL is locked. Completion of this reset sequence is indicated by assertion of the rx_change_done output. This signal connected to this input must be synchronous with the drpclk clock.
rx_refclk_stable	In	1	This input is used by the RX initialization logic to keep the PLL that is providing the serial clock to the GTX RX in reset until the PLL reference clock is stable. If this SDI wrapper is controlling the PLL reset, then the rx_refclk_stable input must be kept Low until the PLL reference clock is stable. This input does not initiate a PLL reset. It only delays completion of the PLL reset sequence initiated by the rx_gtx_full_reset input until the rx_refclk_stable input is High. This input is treated as an asynchronous input.

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

Port Name	I/O	Width	Description
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.
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.

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

Port Name	I/O	Width	Description
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 3 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 4 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).
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.

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

Port Name	I/O	Width	Description
rx_a_vpid	Out	32	All four user data bytes of the SMPTE ST 352 [Ref 6] 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 6] 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 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.
rx_crc_err_a	Out	1	This output is asserted High when a 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

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

Port Name	I/O	Width	Description
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.
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 5 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 both rx_edh_clr_errcnt and rx_ce_sd are both High.

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

Port Name	I/O	Width	Description
rx_edh_ap	Out	1	This output is asserted High when the active picture 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 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 6 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 6 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 6 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 7 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_change_done	Out	1	This output is Low during those periods when the GTX RX is being initialized, reset, or when it is being dynamically switched between SDI modes. If the initialization, reset, or dynamic change sequence completes successfully, the rx_change_done output is asserted High to indicate successful completion. This output is synchronous with the drpclk.
rx_change_fail	Out	1	Under normal conditions, this output is always Low. It only goes High if the control module is unsuccessful in completing a GTX RX initialization, reset, or SDI mode change sequence. If such a failure occurs, the rx_change_fail port is asserted High and the rx_change_fail_code port indicates the nature of the failure. If a failure occurs, the GTX RX must be reset using the rx_gtx_full_reset input. This output is synchronous with the drpclk.
rx_change_fail_code	In	3	When the rx_change_fail port is High, this port indicates the nature of the sequence failure. See Table 8 for encoding of this port. This output is synchronous with the drpclk.
drpbusy	Out	1	This status output is High when the GTX DRP is being used by the SDI control logic.
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.

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

Port Name	I/O	Width	Description
tx_usrclk_out	Out	1	This output is driven by a BUFG inside the SDI wrapper. The input of the BUFG is driven by the txoutclk port of the GTX wrapper. The frequency of this clock is 148.5 MHz (or 148.5/1.001 MHz) in 3G-SDI and SD-SDI modes and 74.25 MHz (or 74.25/1.001 MHz) in HD-SDI mode. Unless otherwise noted, all SDI wrapper ports prefixed by tx_ are synchronous with this clock.
tx_gtx_full_reset	In	1	When this input is asserted High, a full GTX TX reset sequence is initiated. First, the CPLL is reset. After the CPLL locks to the reference clock input, the GTX RX is reset using the GTX transceiver gttxreset port. Completion of this reset sequence is indicated by assertion of the tx_change_done output. This signal connected to this input must be synchronous with the drpclk clock.
tx_gtx_reset	In	1	When this input is asserted High, the GTX TX is reset using the GTX transceiver gttxreset port. If the QPLL and CPLL providing the serial clocks to the GTX TX are not locked, the gttxreset sequence does not complete until both PLLs are locked. Completion of this reset sequence is indicated by assertion of the tx_change_done output. This signal connected to this input must be synchronous with the drpclk clock.
tx_refclk_stable	In	1	This input is used by the TX initialization logic to keep the CPLL that is providing the serial clock to the GTX TX in reset until the CPLL reference clock is stable. The tx_refclk_stable input must be kept Low until the CPLL reference clock is stable. This input does not initiate a CPLL reset. It only delays completion of the CPLL reset sequence initiated by the tx_gtx_full_reset input until the tx_refclk_stable input is High. This input is treated as an asynchronous input.
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 or 148.5/1.001 MHz in 3G-SDI mode, 74.25 or 74.25/1.001 MHz in HD-SDI mode, and 27 MHz in SD-SDI mode. 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.

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

Port Name	I/O	Width	Description
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 for level B (High).
tx_m	In	1	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.
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 the 3G-SDI standards. If the data streams entering the SDI TX input ports do not have CRC values, then 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 numbers 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 the 3G-SDI standards. If the data streams entering the SDI TX input ports do not have line number words already embedded, then 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 6] 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.

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

Port Name	I/O	Width	Description
tx_overwrite_vpid	In	1	If this input is High and the tx_insert_vpid port is High, SMPTE ST 352 [Ref 6] 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 7] 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
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.

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

Port Name	I/O	Width	Description
tx_line_a	In	11	The current line number must be provided to the module through this port if either ST 352 [Ref 6] 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 than 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.
tx_vpid_byte1	In	8	The value on this port is inserted as the first user data word of the ST 352 packet [Ref 6]. 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 6]. 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 6]. 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.

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

Port Name	I/O	Width	Description
tx_vpid_byte4a	In	8	The value on this port is inserted as the fourth user data word of the ST 352 packet. [Ref 6]. 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	This value on this port is inserted as the fourth user data word of ST 352 packets [Ref 6] 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 6] 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, a ST 352 packet [Ref 6] 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.
tx_vpid_line_f2_en	In	1	This input controls whether or not ST 352 packets [Ref 6] 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 6] 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.

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

Port Name	I/O	Width	Description
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 6]. 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 6]. 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 6]. 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.
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 6]. 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.

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

Port Name	I/O	Width	Description
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 6]. 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.
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.

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

Port Name	I/O	Width	Description
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_change_done	Out	1	This output is Low during those periods when the GTX TX is being initialized, reset, or when it is being dynamically switched between SDI modes. If the initialization, reset, or dynamic change sequence completes successfully, the tx_change_done output is asserted High to indicate successful completion. This output is synchronous with the drpclk.
tx_change_fail	Out	1	Under normal conditions, this output is always Low. It only goes High if the control module is unsuccessful in completing a GTX TX initialization, reset, or SDI mode change sequence. If such a failure occurs, the tx_change_fail port is asserted High and the tx_change_fail_code port indicates the nature of the failure. If a failure occurs, the GTX TX must be reset using the tx_gtx_full_reset input. This output is synchronous with the drpclk.
tx_change_fail_code	In	3	When the tx_change_fail port is High, this port indicates the nature of the sequence failure. See Table 9 for encoding of this port. This output is synchronous with the drpclk.

Table 2 lists the parameters that can be applied to the SDI wrapper.

Table 2: **SDI Wrapper Parameter List**

Name	Type	Default	Description
FXDCLK_FREQ	Integer	27000000	This parameter specifies the frequency, in Hz, of the fixed frequency clock on the clk port of the SDI 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 function correctly.
DRPCLK_PERIOD	Integer	37	This parameter specifies the period, (in ns) of the clock that is driving the SDI wrapper's drpclk port. Round all non-integer values down to the nearest integer. The nominal period of this clock must be specified correctly so that the control module can generate delays in the GTX initialization sequences based on the period of this clock.
PLLLOCK_TIMEOUT_PERIOD	Integer	2000000	This parameter specifies the duration of the PLL lock timeout period in ns. If, after being reset, a PLL does not assert its lock signal within this period of time, the control module times out and retries the PLL reset sequence. The default value is equivalent to 2 ms.

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

Name	Type	Default	Description
RESET_TIMEOUT_PERIOD	Integer	500000	This parameter specifies the duration of the GTX transceiver reset timeout period in ns. If, after being reset, the GTX transceiver does not assert its rxresetdone or txresetdone within this period of time, the control module times out and retries the GTX transceiver reset sequence. The default value is equivalent to 500 µs.
TIMEOUT_CNTR_BITWIDTH	Integer	16	This parameter specifies the width in bits of the timeout counter used during the PLL and GTX transceiver reset sequences. The width of this counter must be sufficient to count up to the maximum timeout periods specified by PLLLOCK_TIMEOUT_PERIOD and RESET_TIMEOUT_PERIOD based on the period specified by DRPCLK_PERIOD. For example, the default value of 16 bits is sufficient for timeout periods of up to about 2.4 ms with the default DRPCLK_PERIOD of 37 which is larger than the default values of both PLLOCK_TIMEOUT_PERIOD and RESET_TIMEOUT_PERIOD.
RETRY_CNTR_BITWIDTH	Integer	8	This parameter specifies the width in bits of the retry counter. The retry counter counts the number of retry cycles attempted to complete a GTX RX or TX initialization or reset sequence or a dynamic change of the GTX transceiver txrate or txsysclksel ports. If the retry counter reaches its maximum value of all ones, the sequence is considered to have failed. Thus, this parameter specifies the number of retries that are permitted before the control module gives up on the sequence. The default value of 8 allows for 255 retry cycles.
CPLL_REFCLK_PORT	String	"REFCLK1"	This parameter specifies which reference clock input should be used by the CPLL. Legal values are: "REFCLK0", "REFCLK1", "GREFCLK", "NORTH0", "NORTH1", SOUTH0", and "SOUTH1".
TX_CLK0_QPLL	Integer	1	If the QPLL is the serial clock source to the GTX TX when the tx_m port is Low, this parameter must be assigned a value of 1. If the CPLL is the serial clock source to the GTX TX when the tx_m port is Low, this parameter must be assigned a value of 0.
TX_CLK0_QPLL	Integer	0	If the QPLL is the serial clock source to the GTX TX when the tx_m port is High, this parameter must be assigned a value of 1. If the CPLL is the serial clock source to the GTX TX when the tx_m port is High, this parameter must be assigned a value of 0.
GT_SIM_GTRESET_SPEEDUP	String	"FALSE"	This parameter is passed directly to the identically named parameter of the GTX wrapper. A value of "TRUE" causes a reduction in simulation time by not simulating the full reset sequences. A value of "FALSE" causes the full reset sequences to be simulated. This parameter has no effect on operation in hardware, only in simulation.

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 6]. 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 3. 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 3: **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 4. The encoding of the frame rate matches the encoding used in the picture rate field of SMPTE ST 352 video payload ID packets [Ref 6]. 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 4: **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 asserted 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 5 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 5: **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, see the SMPTE *Error Detection Checkwords and Status Flags for Use in Bit-Serial Digital Interfaces for Television* document (RP 165) [Ref 12].

- 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, if enabled, by the error counter, 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. And 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 is shown in Table 6.

Table 6: **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 7.

Table 7: **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

GTX Initialization and Reset and Change Sequence Failure Codes

If a failure occurs during a GTX RX initialization or reset sequence or during a dynamic change of the RX SDI mode, the rx_change_fail port will be asserted high and a failure code will be output on the rx_change_fail_code port. A sequence only ends in failure after it has been retried the maximum number of times allowed by the retry counter. The maximum number of retries is controlled by the width of the retry counter as specified by the RETRY_CNTR_BITWIDTH parameter. The number of retries attempted is:

$$\text{Retries} = 2^{\text{RETRY\_CNTR\_BITWIDTH}} - 1 \quad \textit{Equation 1}$$

The encoding of the rx_change_fail port is shown in Table 8.

Table 8: **rx_change_fail_code Port Encoding**

Code	Description
0	This code indicates that the PLL failed to lock to its reference clock within the allowed time or the GTX transceiver failed to assert rxresetdone within the allowed period of time following a gtrxreset.
1	This code indicates that the DRP arbiter was not able to grant control of the DRP to the gtrxreset state machine in the GTX wrapper to do a gtrxreset sequence because the DRP was constantly busy. Such a failure should only occur if there is an issue with the x7gtx_sdi_drp_control module which prevents it from giving up the DRP.
2	This failure code is reserved.
3	This failure code is reserved.
4	This failure code is reserved.
5	When a change of the RX SDI mode occurs that requires changing the RXCDR_CFG attribute in the GTX transceiver, the x7gtx_sdi_drp_control module attempts to do a series of DRP write cycles to change that attribute. If any of these write cycles is not acknowledged by the GTX transceiver by asserting the drprdy port within the given amount of time, the entire sequence is aborted and retried up to the maximum allowed number of retries. If the RXCDR_CFG attribute cannot be modified correctly after the maximum number of retires, this failure code is asserted.
6	This failure code is reserved.
7	When the RX SDI mode changes to a mode other than 3G-SDI but rxrate does not need to change (e.g., a change from SD-SDI mode to 3G-SDI mode), the x7gtx_sdi_drp_control module requests a gtrxreset to reset the CDR. If the GTX wrapper does not respond to this gtrxreset request within the allowed amount of time, including retries, this failure code is asserted.

Any sequence failure that results in the rx_change_fail port going High causes the GTX RX control logic in the SDI wrapper to stop in a failure condition. The GTX RX might still receive an SDI signal, but does not dynamically switch between SDI modes as it normally would. A full reset of the GTX RX, by asserting rx_gtx_full_reset High, is required to attempt to resolve the failed condition. Repeated failures most likely indicate an issue with the design of the application.

If a failure occurs during a GTX TX initialization or reset sequence or during a dynamic change of the GTX transceiver txrate or txsysclksel ports, the tx_change_fail port is asserted High and a failure code is output on the rx_change_fail port. As with the RX side, sequences only fail after the maximum number of retries has been attempted. The encoding of the tx_change_fail_code port is shown in Table 9.

Table 9: **tx_change_fail_code Port Encoding**

Code	Description
0	This failure code is reserved.
1	During a full reset sequence or the GTX initialization sequence, this failure code indicates that the PLL providing the serial clock to the GTX TX failed to assert its plllock signal within the given amount of time, including retries, after being reset.
2	During the GTX initialization sequence, a GTX full reset sequence, or a gttxreset sequence requested by the application, this failure code indicates that the GTX transceiver failed to negate its txresetdone signal within the given amount of time, including retries, after assertion of gttxreset. This indicates a failure of the GTX transceiver to respond to the assertion of gttxreset.

Table 9: **tx_change_fail_code Port Encoding *(Cont'd)***

Code	Description
3	During the GTX initialization sequence, a GTX full reset sequence, or a gttxreset sequence requested by the application, this failure code indicates that the GTX transceiver failed to assert its txresetdone signal within the given amount of time, including retries, after a gttxreset.
4	This failure code indicates that the GTX transceiver failed to indicate successful completion of a txrate change by asserting its txratedone output within the allowed amount of time, including retries.
5	When the application requests a dynamic change of txsysclksel by changing the tx_m input of the SDI wrapper, gttxreset is asserted prior to the change of txsysclksel. If the GTX transceiver fails to negate its txresetdone output in response to the assertion of gttxreset within the allowed amount of time, including retries, the txsysclksel change sequence fails with this failure code.
6	During a dynamic change of txsysclksel, gttxreset is asserted. At the end of the sequence, gttxreset is negated. If the GTX transceiver fails to assert the txresetdone output within the allowed amount of time, including retries, after gttxreset is negated, the txsysclksel change sequence fails with this failure code.
7	This failure code is reserved.

SDI Timing Constraints

For the SDI wrapper and the SDI core, only the periods of the clocks need to be constrained. These are the clocks applied to the clk and drpclk ports of the SDI wrapper and the tx_outclk and rx_outclk signals inside the SDI wrapper.

The tx_outclk and rx_outclk clocks are usually constrained to 148.5 MHz, sometimes rounded up to 150 MHz.

Vivado tools consider all clocks to be related, unless told otherwise. The various clocks of the SDI wrapper are generally unrelated, so a constraint is required to specify that these clocks are not related.

See the timing constraints files of the example SDI demonstration provided with this application note for examples of setting these constraints.

Example SDI Demonstrations

Two example SDI demonstration applications are included with this app note. Verilog source code for both demos is provided. And pre-generated FPGA configuration files are also provided for both demonstrations that can be loaded onto a Xilinx Kintex-7 FPGA KC705 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 FMC connector of the KC705 board. The inrevium FMC also provides SDI-specific clock sources which 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 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. And the video data received by each SDI RX can be captured and viewed using an 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 118 of the Kintex-7 FPGA. 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 118 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. You can configure it 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 ChipScope Pro analyzer.

Figure 20 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 just a single SDI connector and a bidirectional SDI physical interface.

The inrevium SDI FMC provides the 148.5 MHz and 148.5/1.001 MHz reference clocks for the GTX transcievers.

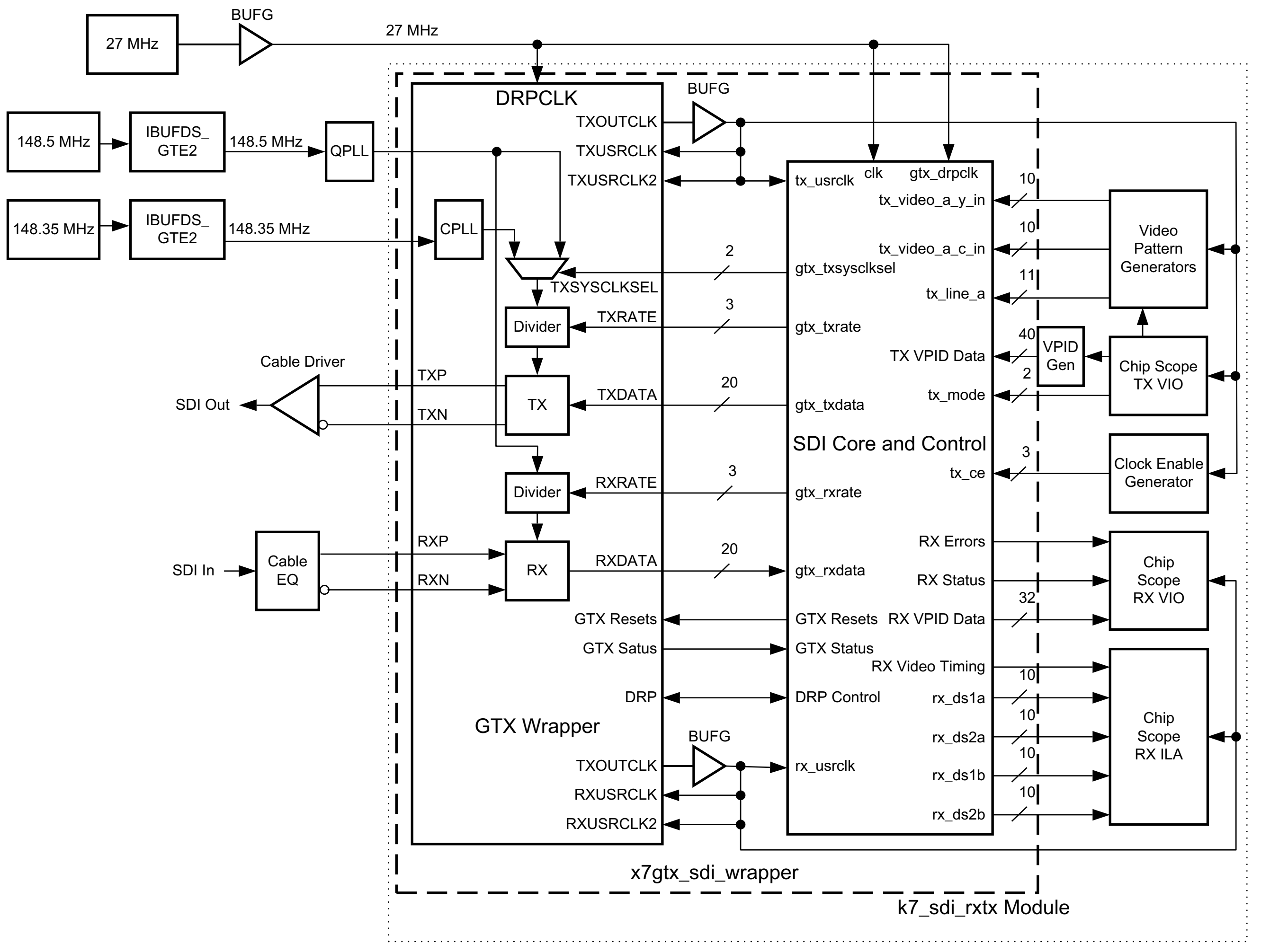


Figure 20: **Quad SDI Block Diagram**

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 k7_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 Kintex-7 FPGA KC705 Evaluation Kit
- inrevium TB-FMCH-3GSDI2A SDI FPGA mezzanine card (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 KC705 board as shown in Figure 21.

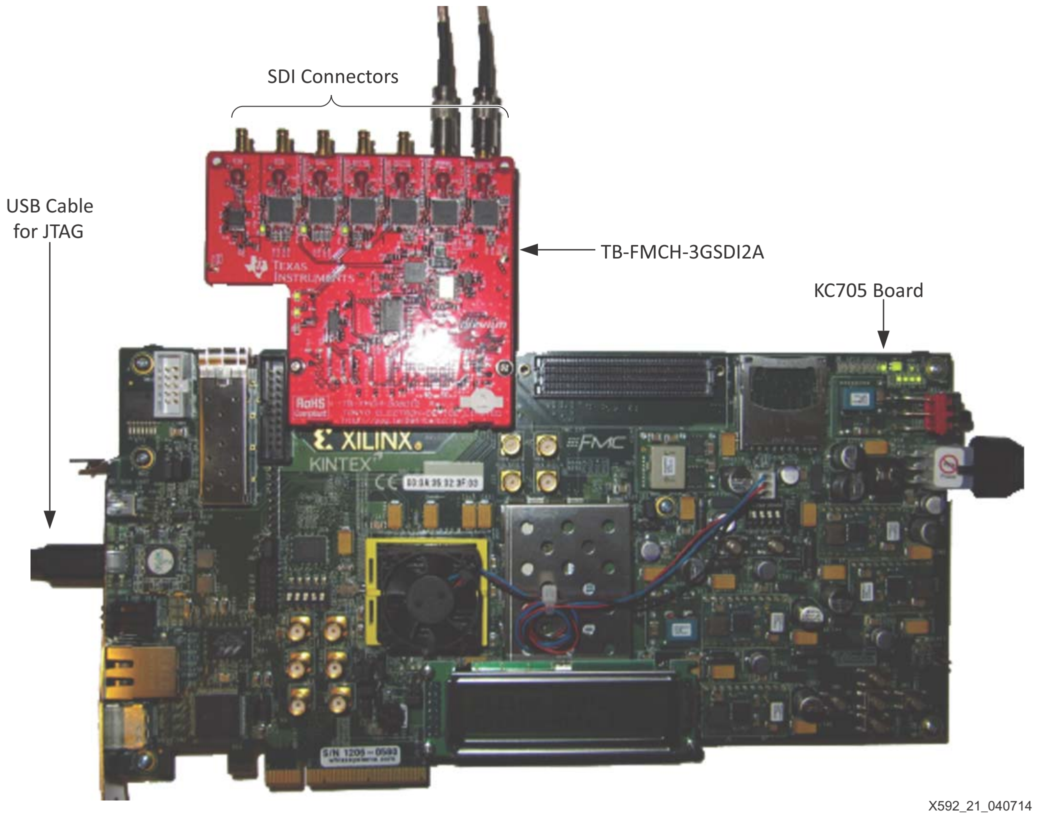
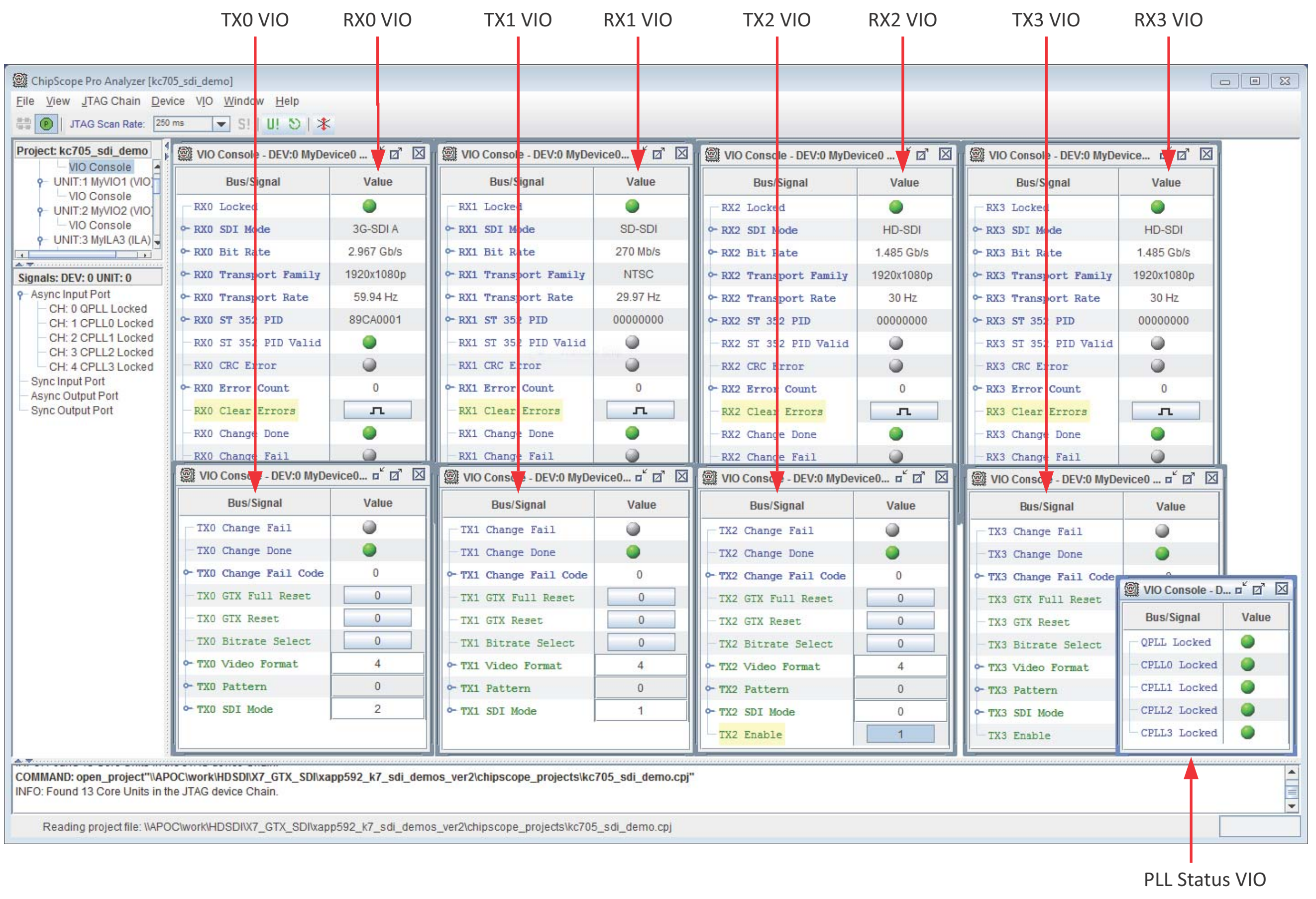


Figure 21: **KC705 Board with TB-FMCH-3GSDI2A Board Connected**

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 KC705 board must be connected to a PC with ChipScope Pro analyzer installed by the USB JTAG cable provided with the KC705 board.

The file called `kc705_sdi_demo.bit` provided with this application note must be loaded into the Kintex-7 FPGA on the KC705 board using ChipScope Pro analyzer. After the BIT file is loaded into the FPGA, the `kc705_sdi_demo.cpj` ChipScope analyzer project file must be opened in ChipScope Pro analyzer. When the project is opened, it looks like Figure 22. There are eight VIO windows, one for each RX and TX in the application. Not visible in Figure 22 are four ILA waveform windows, one for each receiver in the demonstration.



Bus/Signal	Value
RX0 Locked	🟢
RX0 SDI Mode	3G-SDI A
RX0 Bit Rate	2.967 Gb/s
RX0 Transport Family	1920x1080p
RX0 Transport Rate	59.94 Hz
RX0 ST 352 PID	89CA0001
RX0 ST 352 PID Valid	🟢
RX0 CRC Error	⚪
RX0 Error Count	0
RX0 Clear Errors	⎍
RX0 Change Done	🟢
RX0 Change Fail	⚪

Bus/Signal	Value
RX1 Locked	🟢
RX1 SDI Mode	SD-SDI
RX1 Bit Rate	270 Mb/s
RX1 Transport Family	NTSC
RX1 Transport Rate	29.97 Hz
RX1 ST 352 PID	00000000
RX1 ST 352 PID Valid	⚪
RX1 CRC Error	⚪
RX1 Error Count	0
RX1 Clear Errors	⎍
RX1 Change Done	🟢
RX1 Change Fail	⚪

Bus/Signal	Value
RX2 Locked	🟢
RX2 SDI Mode	HD-SDI
RX2 Bit Rate	1.485 Gb/s
RX2 Transport Family	1920x1080p
RX2 Transport Rate	30 Hz
RX2 ST 352 PID	00000000
RX2 ST 352 PID Valid	⚪
RX2 CRC Error	⚪
RX2 Error Count	0
RX2 Clear Errors	⎍
RX2 Change Done	🟢
RX2 Change Fail	⚪

Bus/Signal	Value
RX3 Locked	🟢
RX3 SDI Mode	HD-SDI
RX3 Bit Rate	1.485 Gb/s
RX3 Transport Family	1920x1080p
RX3 Transport Rate	30 Hz
RX3 ST 352 PID	00000000
RX3 ST 352 PID Valid	⚪
RX3 CRC Error	⚪
RX3 Error Count	0
RX3 Clear Errors	⎍
RX3 Change Done	🟢
RX3 Change Fail	⚪

Bus/Signal	Value
TX0 Change Fail	⚪
TX0 Change Done	🟢
TX0 Change Fail Code	0
TX0 GTX Full Reset	0
TX0 GTX Reset	0
TX0 Bitrate Select	0
TX0 Video Format	4
TX0 Pattern	0
TX0 SDI Mode	2

Bus/Signal	Value
TX1 Change Fail	⚪
TX1 Change Done	🟢
TX1 Change Fail Code	0
TX1 GTX Full Reset	0
TX1 GTX Reset	0
TX1 Bitrate Select	0
TX1 Video Format	4
TX1 Pattern	0
TX1 SDI Mode	1

Bus/Signal	Value
TX2 Change Fail	⚪
TX2 Change Done	🟢
TX2 Change Fail Code	0
TX2 GTX Full Reset	0
TX2 GTX Reset	0
TX2 Bitrate Select	0
TX2 Video Format	4
TX2 Pattern	0
TX2 SDI Mode	0
TX2 Enable	1

Bus/Signal	Value
TX3 Change Fail	⚪
TX3 Change Done	🟢
TX3 Change Fail Code	
TX3 GTX Full Reset	
TX3 GTX Reset	
TX3 Bitrate Select	
TX3 Video Format	
TX3 Pattern	
TX3 SDI Mode	
TX3 Enable	

Bus/Signal	Value
QPLL Locked	🟢
CPLL0 Locked	🟢
CPLL1 Locked	🟢
CPLL2 Locked	🟢
CPLL3 Locked	🟢

```
COMMAND: open_project"\\APOC\work\HDSDI\X7_GTX_SDI\xapp592_k7_sdi_demos_ver2\chipscope_projects\kc705_sdi_demo.cpj"
INFO: Found 13 Core Units in the JTAG device Chain.
```

Reading project file: \\APOC\work\HDSDI\X7_GTX_SDI\xapp592_k7_sdi_demos_ver2\chipscope_projects\kc705_sdi_demo.cpj

X592_22_040714

Figure 22: **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 23.

The first three items at the top of the TX VIO window indicate the status of the last GTX TX initialization or dynamic change sequence. If the last sequence completed normally, the Change Done indicator is green. If the last sequence failed, the Change Fail indicator is red and the Change Failure Code indicates the cause of the failure as shown in Table 9.

At the bottom of the TX VIO window are two buttons that reset the GTX TX. The TX GTX Full Reset button resets both the CPLL and the GTX TX unit. The TX GTX Reset button resets just the GTX TX unit and not the CPLL.

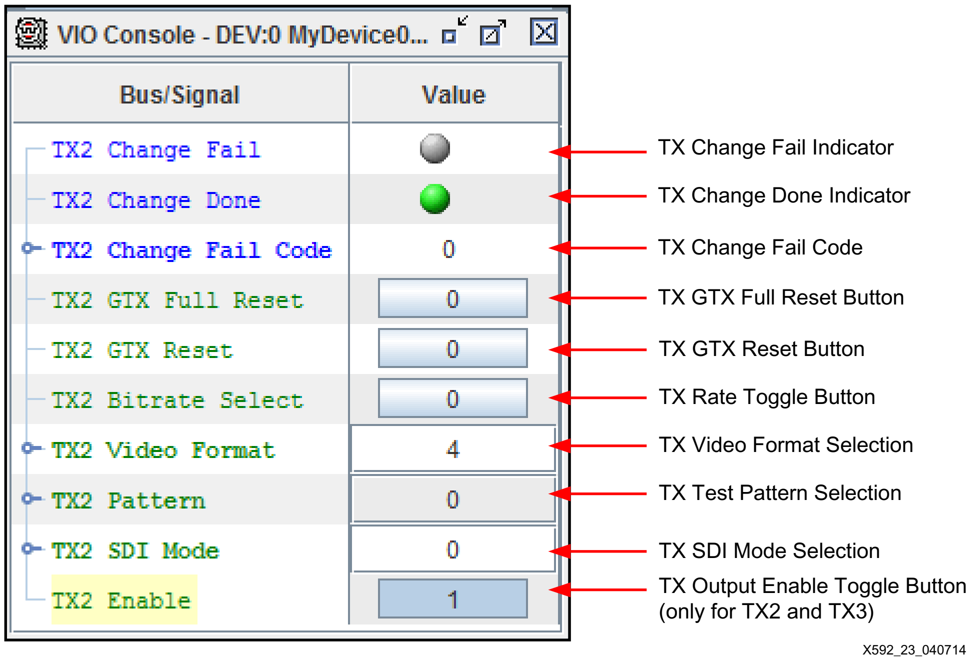


Figure 23: **Quad SDI Demonstration TX Control Window**

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 10.

Table 10: **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

TX2 and TX3 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. Note, however, that 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.

TX0 and TX1 have connectors that are separate from the receivers of those two channels. Thus, the TX VIO control windows for TX0 and TX1 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 24 shows the VIO window for one of the receivers.

The RX Locked Indicator is green when the SDI RX is locked to the incoming SDI signal, 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.

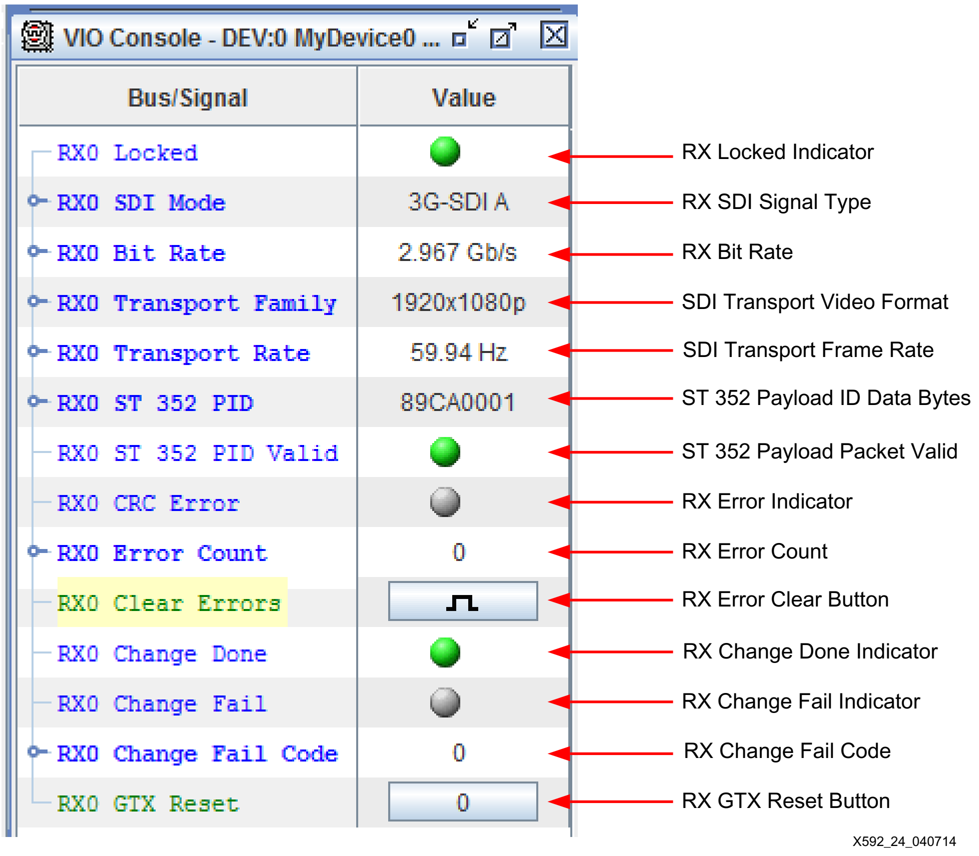
The SDI Transport Video Format provides information about the video transport that has been detected in the SDI signal. The SDI Transport Frame Rate is the frame rate of the video transport that has been detected in the SDI signal. Both of these refer to the transport structure, not necessarily the picture 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 6]. They are shown with byte 1 on the left and byte 3 on the right. They 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, 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 once the SDI RX has fully locked to the new SDI signal, the error count typically is not zero.

The RX Change Done indicator is green under normal operating conditions. It is gray if the RX is currently doing a dynamic change operation such as a reset or SDI mode change. The RX Change Fail indicator is gray under normal conditions and is red if an error has occurred during a dynamic change operation such as a reset or SDI mode change. If the RX Change Fail indicator is red, the cause of the failure is indicated by the RX Change Fail Code value. See Table 8 for descriptions of the failure codes.

The RX GTX Reset button resets the RX section of the GTX.



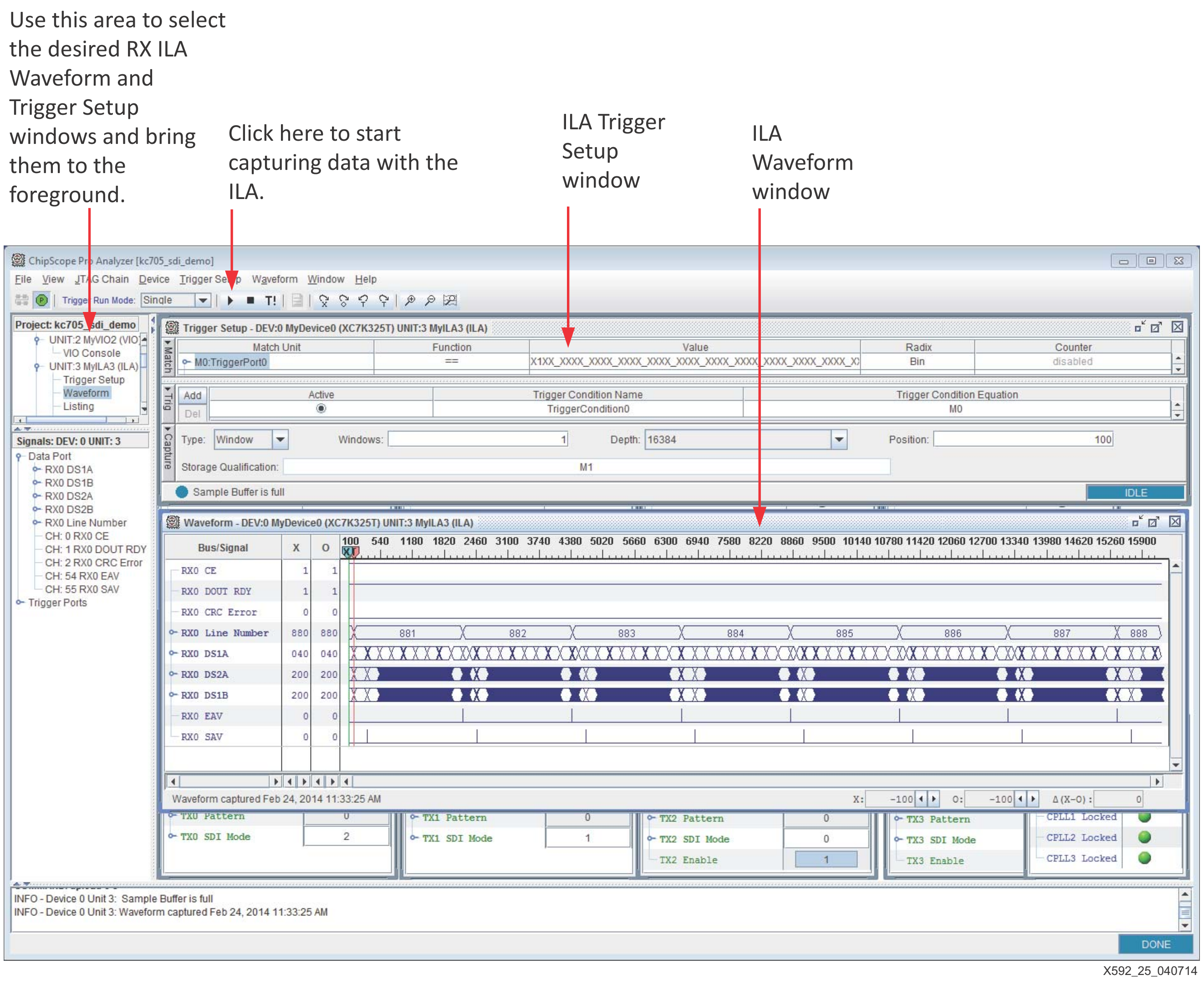
Bus/Signal	Value	
RX0 Locked	●	RX Locked Indicator
RX0 SDI Mode	3G-SDI A	RX SDI Signal Type
RX0 Bit Rate	2.967 Gb/s	RX Bit Rate
RX0 Transport Family	1920x1080p	SDI Transport Video Format
RX0 Transport Rate	59.94 Hz	SDI Transport Frame Rate
RX0 ST 352 PID	89CA0001	ST 352 Payload ID Data Bytes
RX0 ST 352 PID Valid	●	ST 352 Payload Packet Valid
RX0 CRC Error	●	RX Error Indicator
RX0 Error Count	0	RX Error Count
RX0 Clear Errors	⎍	RX Error Clear Button
RX0 Change Done	●	RX Change Done Indicator
RX0 Change Fail	●	RX Change Fail Indicator
RX0 Change Fail Code	0	RX Change Fail Code
RX0 GTX Reset	0	RX GTX Reset Button

Figure 24: **Quad SDI Demonstration RX Status Window**

Figure 25 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 Figure 25. UNIT 3 is the ILA for RX0, UNIT 6 is the ILA for RX1, UNIT 9 is the ILA for RX2, and UNIT 12 is the ILA for RX3.

The **Trigger Setup** window can be used to change the trigger point and storage qualification. There are two match units and 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 26 to initiate a capture by the ILA. The capture buffer is large enough to capture multiple lines of video.


Use this area to select the desired RX ILA Waveform and Trigger Setup windows and bring them to the foreground.
Click here to start capturing data with the ILA.
ILA Trigger Setup window
ILA Waveform window
ChipScope Pro Analyzer [kc705_sdi_demo]
Trigger Setup - DEV:0 MyDevice0 (XC7K325T) UNIT:3 MyILA3 (ILA)
Waveform - DEV:0 MyDevice0 (XC7K325T) UNIT:3 MyILA3 (ILA)
IDLE
DONE
X592_25_040714

Waveform captured Feb 24, 2014 11:33:25 AM

```
INFO - Device 0 Unit 3:  Sample Buffer is full
INFO - Device 0 Unit 3: Waveform captured Feb 24, 2014 11:33:25 AM
```

Figure 25: **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 26 is a block diagram of this 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 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.

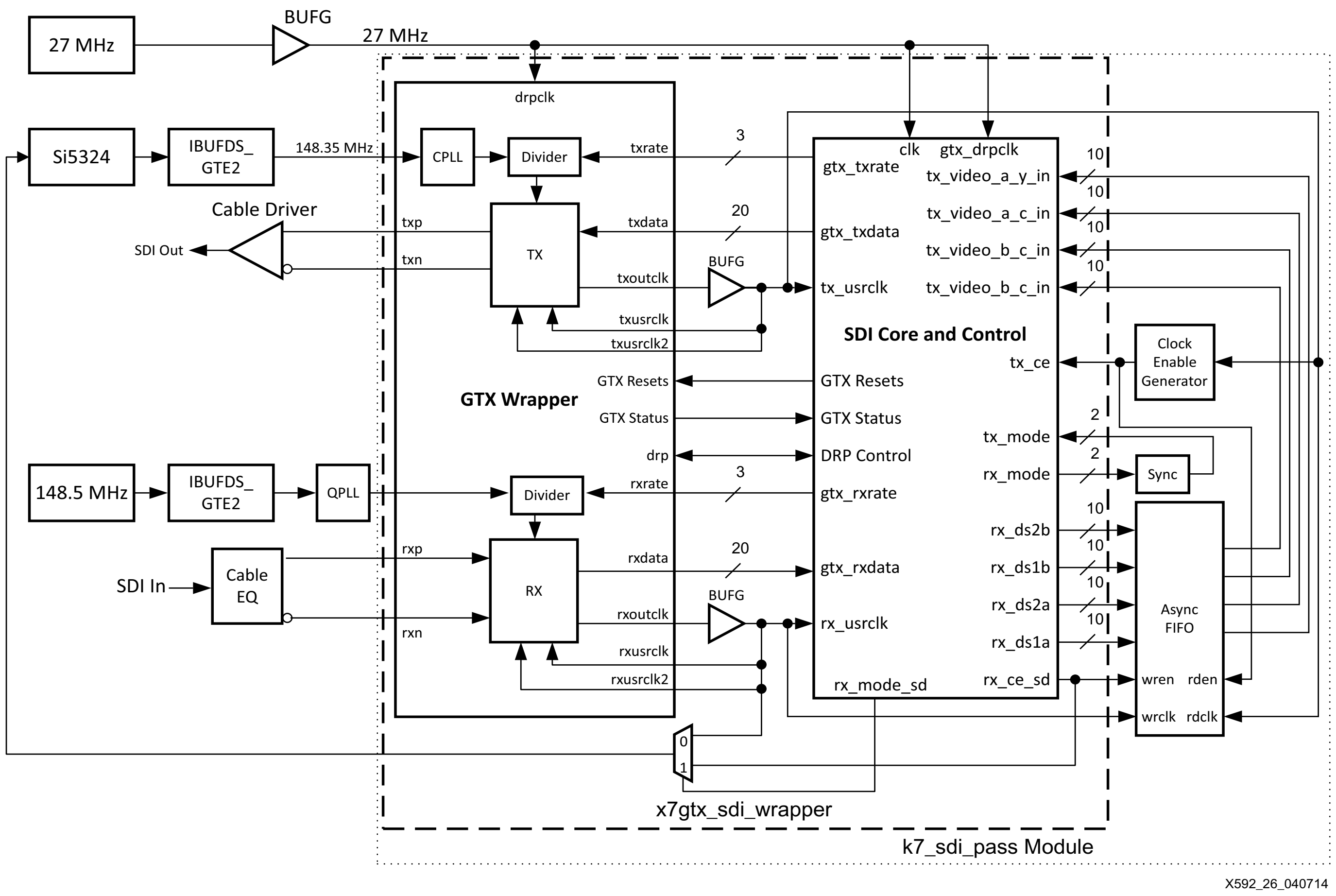


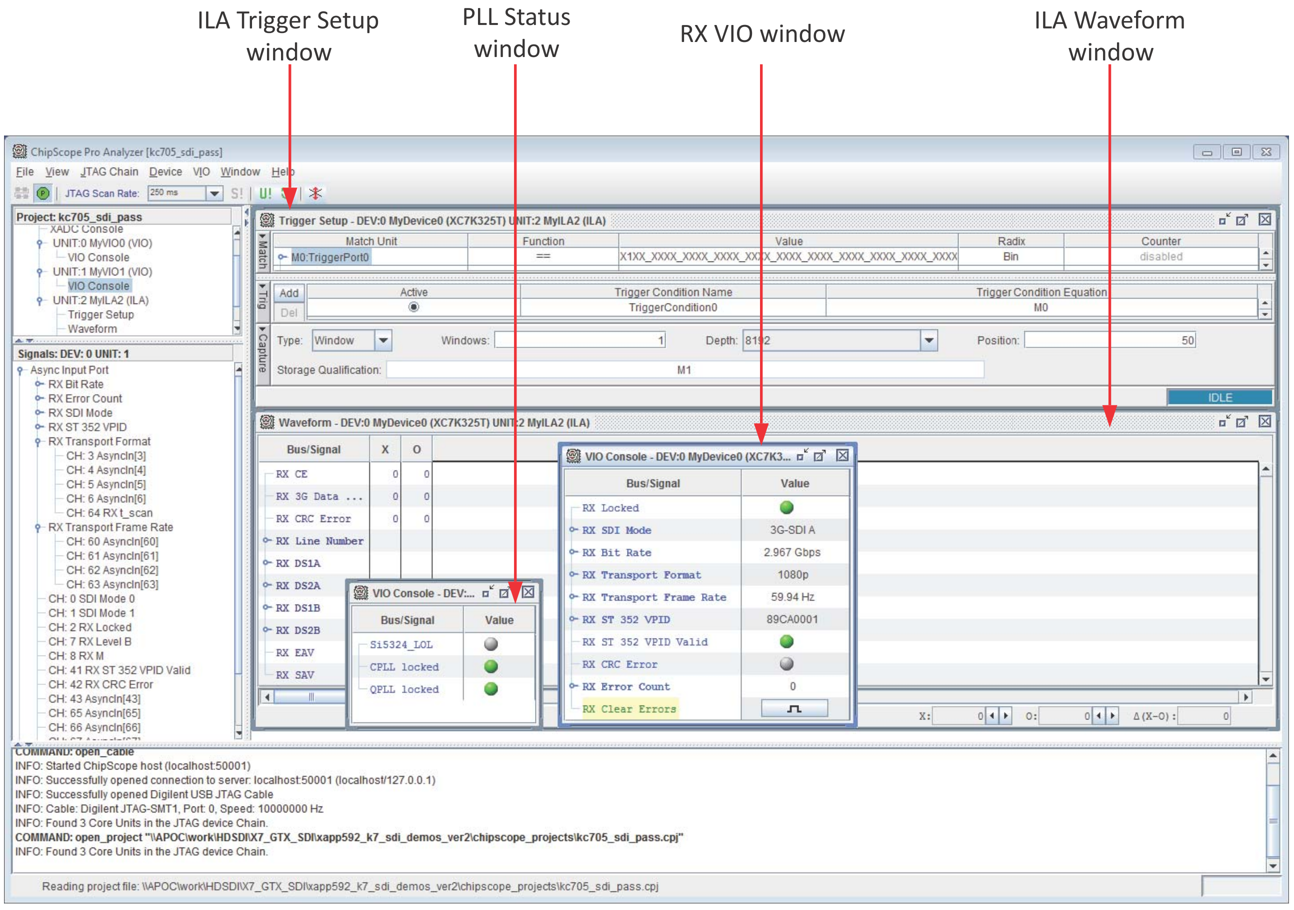
Figure 26: **SDI Pass-Through Demonstration**

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

- Xilinx Kintex-7 FPGA KC705 Evaluation Kit
- inrevium TB-FMCH-3GSDI2A SDI FPGA mezzanine card (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 KC705 board's JTAG USB connector

The inrevium SDI FMC must be connected to the HPC FMC connector on the KC705 board as shown in Figure 21. The only active SDI connectors on the inrevium board are CH0-RX and CH0-TX. A SDI signal source must be connected to the CH0-RX connector. The SDI signal is retransmitted on the CH0-TX connector.

The file called `kc705_sdi_pass_demo.bit` provided with this application note must be loaded into the Kintex-7 FPGA on the KC705 board. After the BIT file is loaded into the FPGA, the `kc705_sdi_pass_demo.cpj` ChipScope analyzer project file can be opened in 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 27.


ILA Trigger Setup window
PLL Status window
RX VIO window
ILA Waveform window
ChipScope Pro Analyzer [kc705_sdi_pass]
Trigger Setup - DEV:0 MyDevice0 (XC7K325T) UNIT:2 MyILA2 (ILA)
Waveform - DEV:0 MyDevice0 (XC7K325T) UNIT:2 MyILA2 (ILA)
VIO Console - DEV:0 MyDevice0 (XC7K3...
VIO Console - DEV:...
X592_27_040714

Bus/Signal	Value
Si5324_LOL	●
CPLL locked	●
QPLL locked	●

Bus/Signal	Value
RX Locked	●
RX SDI Mode	3G-SDI A
RX Bit Rate	2.967 Gbps
RX Transport Format	1080p
RX Transport Frame Rate	59.94 Hz
RX ST 352 VPID	89CA0001
RX ST 352 VPID Valid	●
RX CRC Error	●
RX Error Count	0
RX Clear Errors	

```
COMMAND: open_cable
INFO: Started ChipScope host (localhost:50001)
INFO: Successfully opened connection to server: localhost:50001 (localhost/127.0.0.1)
INFO: Successfully opened Digilent USB JTAG Cable
INFO: Cable: Digilent JTAG-SMT1, Port: 0, Speed: 10000000 Hz
INFO: Found 3 Core Units in the JTAG device Chain.
COMMAND: open_project "\\APOC\work\HDSDI\X7_GTX_SDI\xapp592_k7_sdi_demos_ver2\chipscope_projects\kc705_sdi_pass.cpj"
INFO: Found 3 Core Units in the JTAG device Chain.

Reading project file: \\APOC\work\HDSDI\X7_GTX_SDI\xapp592_k7_sdi_demos_ver2\chipscope_projects\kc705_sdi_pass.cpj
```

Figure 27: **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 24. See the description of the fields and controls for the window given in that section describing that demonstration. Note, 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 ChipScope Pro analyzer. The pass-through SDI interface is fully functional even when ChipScope Pro analyzer is not used to observe the SDI RX.

FPGA Resource Usage

Table 11 shows the FPGA resources required by an SDI interface with a Kintex-7 GTX transceiver. The resource usage includes all the modules required to implement the interface, including the SMPTE SD/HD/3G-SDI core and the SDI wrapper. Resource usage is shown for various common configurations.

The results shown were achieved with Vivado Design Suite 2013.4.

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

Typically, one global or regional clock is required for each SDI TX and for each 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 11: **Kintex-7 GTX SDI Interface FPGA Resource Usage**

Reference Design	LUTs	FFs
SDI RX with EDH processor and TX	3650	2955
SDI RX without EDH processor and TX	3077	2425
SDI RX with EDH processor	2300	1750
SDI RX without EDH processor	1725	1395
SDI TX	1445	1087

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 Kintex-7 devices is typically not a problem.

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. 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 coordinates 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 12 lists a glossary of terms used in this application note.

Table 12: **Glossary**

Term	Definition
3G-SDI	Common name for SMPTE ST 424, the 3 Gb/s serial digital interface [Ref 13]. 3G-SDI supports three mapping modes defined in ST 425-1 called 3G-SDI level A, level B-DL, and level B-DS. See *Source Image Format and Ancillary Data Mapping for the 3 Gb/s Serial Interface* (ST 425-1) [Ref 14] 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 15].
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 12].
Embedded audio	Generally refers to digital audio that is carried as ancillary data in an SDI signal.

Table 12: **Glossary *(Cont'd)***

Term	Definition
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 16].
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 multilink set is called a link. SMPTE *Dual Link 1.5 Gb/s Digital Interface for 1920 x 1080 and 2048 x 1080 Picture Formats* (ST 372) [Ref 7] defines how to transport some higher bandwidth video formats on two HD-SDI links. Multilink 3G-SDI standards in the ST 425-x family are currently under development by SMPTE [Ref 14]. 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 6]. 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, color space, and so on) and the type of SDI interface used to transport that payload. In multilink interfaces, the payload ID also contains bits that distinguish between the individual links.
Progressive	A non-interlaced video scanning system. All lines of the progressive frame belong to the same picture.
Serial Digital Interface (SDI)	Originally referred to as SMPTE *Television – SDTV Digital Signal/Data – Serial Digital Interface* (ST 259) [Ref 3], 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*.
SD-SDI	Common name for SMTPE *Television – SDTV Digital Signal/Data – Serial Digital Interface* (ST 259) [Ref 3], the standard-definition serial digital interface.
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.

Table 12: **Glossary *(Cont'd)***

Term	Definition
Synchronous switching (point, interval, line)	SMPTE *Definition of Vertical Switching Point for Synchronous Video Switching* (RP 168) [Ref 17] 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*.
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.
Timing reference signal (TRS)	A generic term referring to both EAV and SAV sequences.
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=192180

Reference Design Matrix

The reference design matrix is shown in Table 13.

Table 13: **Reference Design Matrix**

Parameter	Description
General	
Developer name	John Snow
Target devices	All Kintex-7 FPGA devices
Source code provided	Yes
Source code format	Verilog
Design uses code/IP from existing Xilinx application note/reference designs, the CORE Generator tool, or third party	Yes. IP cores from Vivado IP catalog
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	Vivado synthesis
Implementation software tools/versions used	Vivado Design Suite 2013.4

Table 13: **Reference Design Matrix *(Cont'd)***

Parameter	Description
Static timing analysis performed	Yes
Hardware Verification	
Hardware verified	Yes
Hardware platform used for verification	KC705 and TB-FMCH-3GSDI2A

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 Kintex-7 GTX transceivers to implement SDI interfaces compatible with the SMPTE SD-SDI, HD-SDI, and 3G-SDI standards. The Kintex-7 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 Kintex-7 FPGA design.

References

This section lists the references available from Xilinx (www.xilinx.com) or from the Society of Motion Picture and Television Engineers (www.smpte.org).

1. *7 Series GTX/GTX Transceivers User Guide* (UG476)
2. *Dynamically Programmable DRU for High-Speed Serial I/O* (XAPP875)
3. *Television – SDTV Digital Signal/Data – Serial Digital Interface* (SMPTE ST 259)
4. *Television – 540 Mb/s Serial Digital Interface* (SMPTE ST 344)
5. *Society of Motion Picture and Television Engineers (SMPTE) SD/HD/3G-SDI Product Guide* (PG071)
6. *Payload Identification Codes for Serial Digital Interfaces* (SMPTE ST 352)
7. *Dual Link 1.5 Gb/s Digital Interface for 1920 x 1080 and 2048 x 1080 Picture Formats* (ST 372)
8. *Television – 1920 x 1080 Image Sample Structure, Digital Representation and Digital Timing Reference Sequences for Multiple Picture Rates* (SMPTE ST 274)
9. *1280 x 270 Progressive Image 4:2:2 and 4:4:4 Sample Structure -- Analog and Digital Representations and Analog Interface* (SMPTE ST 296)
10. *2048 x 1080 Digital Cinematography Production Image FS/709 Formatting for Serial Digital Interface* (SMPTE 2048-2)
11. *Television –1920 x 1080 50-Hz - Scanning and Interface* (SMPTE 295)
12. *Error Detection Checkwords and Status Flags for Use in Bit-Serial Digital Interfaces for Television* (RP 165)
13. *Television – 3 Gb/s Signal/Data Serial Interface* (SMPTE ST 424)
14. *Source Image Format and Ancillary Data Mapping for the 3 Gb/s Serial Interface* (ST 425-1)
15. *Television – Ancillary Data Packet and Space Formatting* (SMPTE ST 291)
16. *1.5 Gb/s Signal/Data Serial Interface (*SMPTE ST 292-1)
17. *Definition of Vertical Switching Point for Synchronous Video Switching* (RP 168)
18. *Kintex-7 FPGAs Data Sheet: DC and Switching Characteristics* (DS182)

Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
09/06/2012	1.0	Initial Xilinx release.
02/07/2013	1.1	The document was updated to support ISE® Design Suite 14.4 and Vivado Design Suite 2012.4. Removed LPM-Manual as a possible GTX RX EQ mode for SDI. LPM-Manual has been deprecated from the GTX wizard. TED was changed to inrevium throughout. In Table 1, the rx_crc_err_a port description changed. Figure 11, 7 Series FGPAs Transceivers Wizard - Page 1 through Figure 15, 7 Series FPGAs Transceivers Wizard - Page 5 were replaced. Board TB-FMCH-3GSDI2 changed to TB-FMCH-3GSDI2A. In Table 10 in the TX Video Format 4 row, in the 3G-SDI (SDI Mode = 2) / TX Bit Rate = 1 column, 1080p 59.97 Hz changed to 1080p 59.94 Hz,
07/14/2014	2.0	The document was updated to support Vivado Design Suite 2013.4 and later. The GTX wrappers generated by the 7 Series FPGAs Transceivers Wizard changed significantly from earlier versions of Vivado, requiring extensive changes to the SDI wrapper and the SDI demo applications supplied with this application note. Changes to the GTX wrappers generated by the 7 Series FPGAs Transceivers Wizard resulted in the following changes to the document: The QPLL must now be instantiated separately from the transceivers. The QPLL is instantiated in the new GTX common wrapper. References to the GTX common wrapper were added throughout the document. The GTX wrapper generated by the transceiver wizard now has all lowercase port names, rather than all uppercase. GTX wrapper port names have been changed to all lowercase throughout the document. GTX transceivers are now instantiated individually rather than in groups and the document has been modified to explain how this is done. Generating the GTX Wrapper has been rewritten to be compatible with the extensive changes that have occurred with the 7 Series FPGAs Transceivers Wizard. The SDI wrapper now implements all required details of the GTX initialization sequence including generating resets for the QPLL and CPLLs. The GTX Transceiver Initialization Sequence, PLL Resets, GTX TX Resets, and GTX RX Resets sections have been rewritten and a new GTX PLL Usage Models for SDI Applications section has been added to describe these changes. A new GTX Initialization and Reset and Change Sequence Failure Codes section has been added to document the failure codes that are now output by the SDI wrapper when errors occur during GTX initialization, resets, and dynamic change sequences. These failure codes are now visible in the ChipScope Pro analyzer VIO windows of the demos as described in the Example SDI Demonstrations section. The SDI Electrical Interface section was modified to add some information about using GTX transceivers with new generation SDI cable equalizers that default to 600 mV output swing. Table 1 has been updated with the changes made to various ports on the SDI wrapper. Table 2 has been updated due to changes made to the SDI wrapper parameters. Table 11 has been updated with the latest FPGA resource usage numbers for the SMPTE SD/HD/3G-SDI core.

Notice of Disclaimer