XAPP1248 (v1.4) February 1, 2018

Implementing SMPTE SDI Interfaces with UltraScale GTH Transceivers

Author: SaiRam Nedunuri, Killivalavan Kaliyamoorthy

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. 6G-SDI and 12G-SDI, collectively referred to as UHD-SDI, are recent extensions to the SDI family of standards that provide increased bandwidth to transport Ultra HD (UHD) video formats and higher frame rate HD video formats.

The Xilinx® LogiCORE™ IP SMPTE UHD-SDI core is a generic UHD-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 UHD-SDI LogiCORE IP with the UltraScale™ GTH transceivers to form a complete UHD-SDI interface. This application note also provides an example SDI design that runs on the Xilinx UltraScale FPGA KCU105 evaluation board.

Introduction

The Xilinx LogiCORE IP SMPTE UHD-SDI core (UHD-SDI core) can be connected to an UltraScale GTH transceiver to implement an SDI interface capable of supporting the SMPTE SD-SDI, HD-SDI, 3G-SDI, 6G-SDI, and 12G-SDI standards. The UHD-SDI core and GTH 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 as Verilog source code.

The term *SDI* is used generically to refer to the SMPTE family of interface standards including SD-SDI, HD-SDI, 3G-SDI, 6G-SDI, and 12G-SDI in this application note. Refer to the Glossary for a list of terms used in this application note.

UltraScale GTH transceivers support all SDI bit rates up to and including 12G-SDI. CPLL usage imposes a bit rate limitation of up to 6G-SDI for -1 speed grade and 12G-SDI for -2 and -3 speed grades. Refer to the *GTH Transceiver Switching Characteristics* section of *Kintex UltraScale Architecture Data Sheet: DC and AC Switching Characteristics* [Ref 16] for the maximum line rates supported by the GTH transceivers for each combination of speed grade and device package.

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

- reset logic for the GTH transceiver
- dynamic switching of the GTH receiver (RX) and transmitter (TX) serial clock dividers to support the five SDI standards
- dynamic TX reference clock switching to support the two different bit rates in each of the HD-SDI, 3G-SDI, 6G-SDI, and 12G-SDI standards:
  - 1.485 Gb/s and 1.485/1.001 Gb/s in HD-SDI mode
  - 2.97 Gb/s and 2.97/1.001 Gb/s in 3G-SDI mode
  - 5.94 Gb/s and 5.94/1.001 Gb/s in 6G-SDI mode
  - 11.88 Gb/s and 11.88/1.001 Gb/s in 12G-SDI mode
- dynamic RX reference clock switching to support the two different bit rates in each of the 12G-SDI standard:
  - 11.88 Gb/s and 11.88/1.001 Gb/s in 12G-SDI mode
- dynamic switching of the GTH RXDATA and TXDATA port widths
  - 20-bit RXDATA and TXDATA ports for SD-SDI, HD-SDI, and 3G-SDI modes
  - 40-bit RXDATA and TXDATA ports for 6G-SDI and 12G-SDI modes
- data recovery unit for recovering data in SD-SDI mode
- RX bit rate detection to determine if the RX is receiving integer frame-rate signals (line rates such as 1.485 Gb/s and 2.97 Gb/s) or fractional frame-rate signals (line rates such as 1.485/1.001 Gb/s and 2.97/1.001 Gbs)

This application note provides a wrapper file that includes an instance of the control module for the GTH transceiver, one GTH channel instance, and the UHD-SDI core with necessary connections between them to simplify the creation of an SDI interface.

Figure 1 shows a simplified block diagram of the SDI interface.

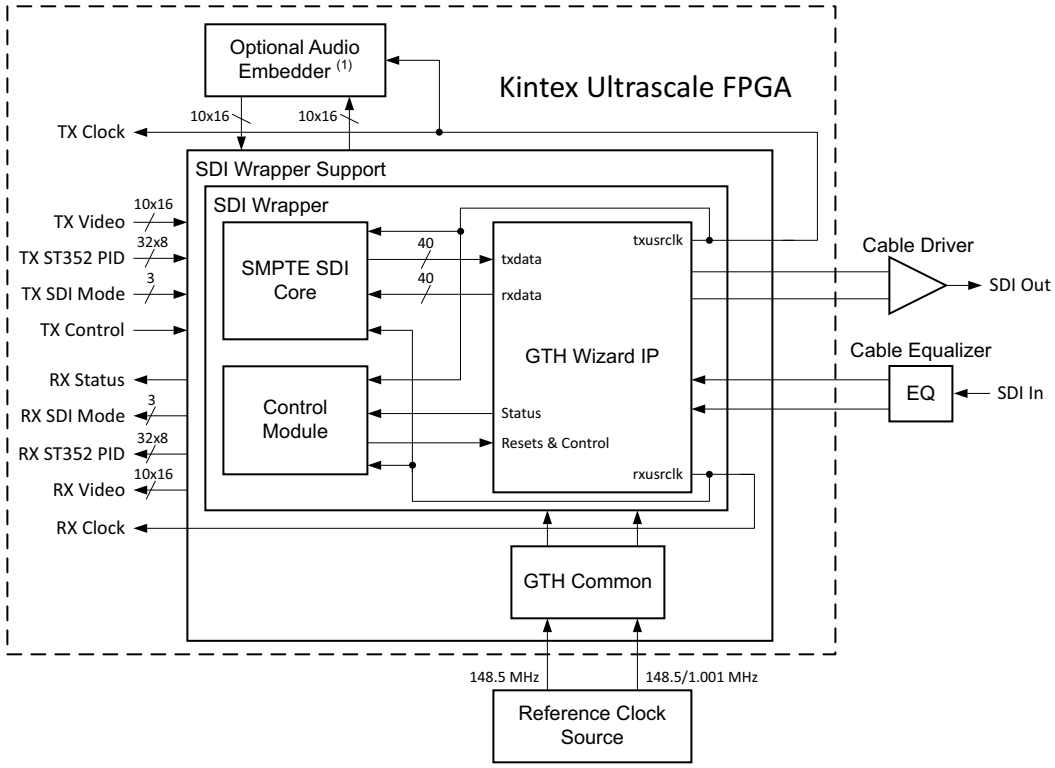


Figure 1: **Block Diagram of a Typical SDI RX/TX Interface**

In Figure 1:

- SMPTE SDI Core refers to the SMPTE UHD-SDI core that is available in the Vivado® IP catalog.
- Control Module is a module that implements the various device-specific functions required when using the GTH transceiver to implement an SDI interface using the SMPTE UHD-SDI core. The control module is supplied in source code form with this application note.
- GTH Wizard IP is an UltraScale Transceiver Wizard module that includes an instance of a single `GTHE3_CHANNEL` transceiver and its corresponding control module. This wrapper is generated by the UltraScale FPGAs Transceiver Wizard which is available in the Vivado IP catalog.
- SDI Wrapper is a wrapper module that instantiates and interconnects the UHD-SDI core, GTH wizard IP, and the control module. The SDI wrapper is supplied in source code form with this application note.
- SDI Wrapper Support module contains one SDI Wrapper instance and a `GTHE3_COMMON` primitive for a GTH quad. This wrapper is meant to be instantiated once per quad, and its QPLL clock, reference clock, and lock outputs should be connected to the SDI wrapper residing in the same quad but in a different channel. If the QPLL is not used in the SDI application, this wrapper is not required.
- The optional audio embedder is a separate core and not included with the UHD-SDI core or this application note.

Features

The *SMPTE UHD-SDI Product Guide* [Ref 17] lists all the features of the UHD-SDI core and SMPTE standards supported by the core, and also includes timing diagrams that show the input and output timing of the core in the various SDI modes.

This application note uses the term *elementary data streams* to refer to an SDI data stream that is not multiplexed. For example, an HD-SDI signal consists of two elementary data streams, usually referred to as the Y and C data streams, that are multiplexed together onto the virtual 10-bit HD-SDI interface. Likewise, a 3G-SDI level A signal also consists of two elementary data streams, called data stream 1 and data stream 2, that are multiplexed together onto the 10-bit virtual 3G-SDI interface. However, a 3G-SDI level B-DS signal consists of four elementary data streams, a Y and a C data stream for each of the HD-SDI signals that are aggregated together onto the 3G-SDI level B interface. These four elementary streams get interleaved in a 4-way multiplex onto the 10-bit virtual 3G-SDI interface. With the introduction of 6G-SDI and 12G-SDI, up to 16 elementary data streams may be interleaved onto a single SDI interface. These data streams are labeled ds1 through ds16 in this application note and in the port names of the UHD-SDI core and the UHD-SDI wrapper.

The UHD-SDI core transmitter only accepts and the receiver only outputs elementary, non-multiplexed, data streams on the respective data stream inputs and outputs. The multiplexing and de-multiplexing of data streams occurs internally in the UHD-SDI core and does not need to be addressed outside of the core, with the exception of SD-SDI. The ST 259 SD-SDI standard defines a single data stream that carries both the Y and C components. This is considered to be an elementary data stream by the UHD-SDI core because multiple end active video (EAV) and start active video (SAV) are not interleaved.

The UHD-SDI core does not map between native video formats and elementary data streams. The user application must perform any necessary mapping of video to elementary data streams prior to providing those streams to the UHD-SDI transmitter and must reconstruct the video image from the elementary streams output by the UHD-SDI receiver. No mapping is necessary for all video formats on SD-SDI and single-link HD-SDI, and for 1080p 50, 59.94, and 60Hz 4:2:2 YCBCR 10-bit video on 3G-SDI level A because there is a one-to-one correspondence between the data streams of these formats and the elementary data streams into and out of the UHD-SDI core. This is also true for 3G-SDI level B-DS, the dual stream mode where two HD-SDI video formats are aggregated onto a single 3G-SDI interface. For dual-link HD-SDI, 3G-SDI level B-DL, multi-link 3G-SDI, 6G-SDI, and 12G-SDI, mapping of the video formats to and from elementary data streams is required, and is not done in the UHD-SDI core.

For 6G-SDI, the UHD-SDI core supports up to 8 elementary data streams. For 12G-SDI, the UHD-SDI core supports up to 16 elementary data streams. In the SMPTE 6G-SDI and 12G-SDI mapping documents, the term *data streams* refers to both multiplexed and non-multiplexed (elementary) data streams and care must be used when interpreting these documents to determine how many elementary data streams are used by each mapping method. Depending on the data format being transported, either four or eight elementary data streams are interleaved together on a 6G-SDI interface, and either eight or sixteen elementary data streams are interleaved together on a 12G-SDI interface. The 16-way interleave only occurs in dual link 12G-SDI. The UHD-SDI TX must be told how many streams are active on its input using the port

called tx_mux_pattern. The UHD-SDI RX automatically determines how many elementary data streams are present in the incoming SDI signal, demultiplexes the data stream appropriately, and indicates on the rx_active_streams port how many elementary data streams are present in the incoming signal.

Using UltraScale GTH Transceivers for SDI Interfaces

This section supplements the *UltraScale Architecture GTH Transceivers User Guide* (UG576) [Ref 13], and highlights features and operating requirements of the GTH transceivers that are important for UHD-SDI applications.

This application note uses the same naming convention in the *UltraScale Architecture GTH Transceivers User Guide* (UG576) [Ref 13] for the GTH transceiver ports, which uses only the base name of a port. When the UltraScale FPGAs Transceiver Wizard is used to create a GTH wizard module, all input ports names have a suffix of _in and all outputs have a suffix of _out. For example, a port named txpllclksel would be txpllclksel_in.

Several clocks are required in applications that use GTH 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. GTH transceivers require reference clocks to operate. The reference clocks are used by phase-locked loops (PLLs) in the GTH transceiver quad to generate serial clocks for the receiver and transmitter sections of each transceiver. As described in more detail in GTH Transceiver Reference Clocks, the serial bit rate of the GTH transmitter is an integer multiple of the reference clock frequency that 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 GTH 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 GTH transmitter clocking is handled by the Transmitter User Clocking Network Helper Block when enabled during GT IP generation from UltraScale FPGAs Transceiver Wizard. The txusrclk and txusrclk2 output is driven by a BUFG_GT within the helper block and its frequency is equal to the word rate of the data that must enter the txdata port of the GTH transmitter. The txusrclk and txusrclk2 are generated in the GTH transmitter by dividing the serial clock from the PLL down to the word rate. Refer to the *UltraScale FPGAs Transceivers Wizard* [Ref 15] for more details on Transmitter User Clocking Network Helper Block.

The GTH receiver reference clock does not need an exact relationship with the bit rate of the incoming SDI signals. This is because the clock and data recovery (CDR) unit in the GTH receiver can receive bit rates that are up to ±1,250 ppm (≤ 6.6 Gb/s) or ±200 ppm (> 8.0 Gb/s) away from the nominal bit rate as set by the reference clock frequency. This allows the receiver reference clock to be generated by local oscillators that have no exact frequency relationship to the incoming SDI signal. The GTH receiver generates a recovered clock (rxoutclk) that is frequency-locked to the incoming SDI bit rate. These clocks are output as rxusrclk and rxusrclk2 ports of the Receiver User Clocking Network Helper Block from the GTH Wizard IP and are driven by BUFG_GT. As is described in Generating an SD-SDI Recovered Clock,

rxusrclk and rxusrclk2 are true recovered clocks when receiving all SDI line rates except when receiving SD-SDI signals.

One additional clock is required for SDI applications – a free-running, fixed-frequency clock used as the clock for the dynamic reconfiguration port (DRP) of the GTH transceiver. This clock is also usually supplied to the control module in the SDI wrapper for timing purposes. The valid frequency range for this clock is stated in the UltraScale FPGAs Transceiver Wizard and normally ranges from 3.125 to 200 MHz. The frequency of this clock does not require any specific relationship to other clocks or data rates of the SDI application. This clock must not change frequencies when the SDI mode changes. It must remain running at the same nominal frequency at all times and never stop while the SDI application is active. This clock can be used for all SDI interfaces in the device.

The frequency of the rxusrclk and txusrclk depend on the SDI mode and the width of the GTH transceiver's rxdata and txdata ports. This relationship is fixed by the architecture of the GTH transceiver. The receiver and the transmitter both use clock enables to throttle the data stream transfer data rate because, in some cases, the data rate on the data streams is less than the frequency of the clock. Table 1 shows the relationships between SDI mode, number of active data streams, rxdata/txdata port widths, rxoutclk/txoutclk frequencies, and clock enable cadences. The clock enable cadences are given in number of clocks between assertions of the clock enable over two data word cycles where 1/1 means that the clock enable is asserted every clock cycle, 2/2 indicates assertion every other clock cycle (50% duty cycle), 4/4 indicates assertion every fourth clock cycle (25% duty cycle), and 5/6 indicates that the clock enable alternates between assertion every 5 or 6 clock cycles, to average once every 5.5 clock cycles (one instance of 5 clock cycles between High pulses on the clock enabled followed by one instance of 6 clock cycles between High pulses on the clock enable, with this pattern repeating).

Table 1: **Clock Frequencies and Clock Enable Requirements**

SDI-Mode	Active Data Streams	RX/TXDATA Bit Width	RX/TXOUTCLK Frequency	Clock enable
SD-SDI	1	20	148.5 MHz	5/6
HD-SDI	2	20	74.25 or 74.25/1.001 MHz	1/1
3G-SDI A	2	20	148.5 or 148.5/1.001 MHz	1/1
3G-SDI B	4	20	148.5 or 148.5/1.001 MHz	2/2
6G-SDI	4	40	148.5 or 148.5/1.001 MHz	1/1
6G-SDI	8	40	148.5 or 148.5/1.001 MHz	2/2
12G-SDI	8	40	297 or 297/1.001 MHz	2/2
12G-SDI	16	40	297 or 297/1.001 MHz	4/4

GTH Transceiver Reference Clocks

UltraScale GTH transceivers are grouped into quads. Each quad contains four GTHE3_CHANNEL transceiver primitives and one GTHE3_COMMON primitive containing two quad PLLs (QPLL0 and QPLL1) as shown in Figure 2. The clock generated by the QPLL0 and QPLL1 is distributed to all four transceivers in the quad. Each GTHE3_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 or both QPLL0 and QPLL1 or the CPLL as its clock source. Furthermore, any RX or TX unit can dynamically switch its clock source between QPLL0, QPLL1, and CPLL. This configuration and the dynamic switching capability are particularly useful for SDI applications.

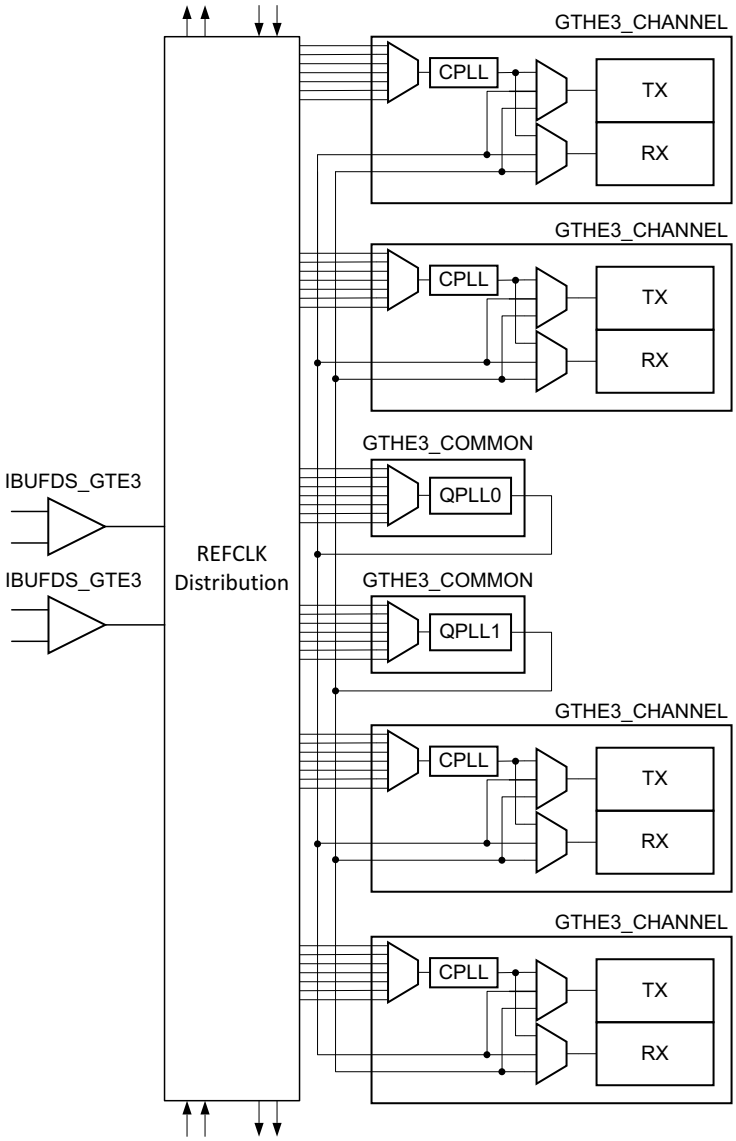


Figure 2: **UltraScale GTH Transceiver Quad Configuration**

Typical UHD-SDI applications require the GTH transceivers to support nine different bit rates:

- 270 Mb/s for SD-SDI
- 1.485 Gb/s for HD-SDI
- 1.485/1.001 Gb/s for HD-SDI
- 2.97 Gb/s for 3G-SDI
- 2.97/1.001 Gb/s for 3G-SDI
- 5.94 Gb/s for 6G-SDI
- 5.94/1.001 Gb/s for 6G-SDI
- 11.88 Gb/s for 12G-SDI
- 11.88/1.001 Gb/s for 12G-SDI

The CDR unit in the RX section of the GTH transceiver can support receiving bit rates that are up to +/-1250 ppm from the reference frequency at bit rates less than 6.6 Gb/s. HD-SDI, 3G-SDI, 6G-SDI, and 12G-SDI each have two bit rates that differ by exactly 1000 ppm. For HD-SDI, 3G-SDI, and 6G-SDI, both bit rates can be received using a single reference clock frequency. That same reference clock frequency can also support reception of SD-SDI. Thus, for all SDI modes except 12G-SDI, a single RX reference clock frequency is required. However, at 12G-SDI rates, the CDR unit has only ±200ppm tolerance relative to the reference clock frequency. Thus two different reference clock frequencies are needed to receive the two 12G-SDI bit rates. These two reference clock frequencies are typically 148.5 MHz to receive 11.88 Gb/s and 148.5/1.001 MHz to receive 11.88/1.001 Gb/s.

Therefore, most SDI applications provide two separate reference clocks to the GTH quad. Usually, the supplied reference frequency pair are 148.5 MHz and 148.5/1.001 MHz. This application note always refers to the reference clock frequency pair 148.5 MHz and 148.5/1.001 MHz.

The source of the GTH transceiver reference clocks is application specific. The receiver reference clock source can be a local oscillator because it does not need to exactly match the incoming SDI bit rate. However, because the GTH transmitter line rate is always a multiple of the reference clock frequency, the frequency of the transmitter reference clock must be exactly related to the data rate of the transmitted data. Most often, the transmitter reference clocks are generated by generator locking (genlock) PLLs, thereby deriving the GTH transmitter line rate from the studio video reference signal. In some cases, such as the SDI pass-through connection, the transmitter line rate is derived from the recovered clock of the GTH 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.

In a typical UHD-SDI application, the two reference clocks are connected to the QPLL0 and QPLL1. The RX and TX units of each transceiver in the quad dynamically switch between the PLL clocks, depending on the bit rate that is required at the moment. The GTH `txsysclksel` and `rxsysclksel` ports are used to select the TX and RX units serial clock source between the PLLs. This common configuration for SDI applications is shown in Figure 3. In Figure 3, multiplexers that are not used dynamically in the implementation have been replaced with wires and the reference clock routing between quads is not shown. It is also possible to the connect

one reference clock to CPLL and the other to QPLL0/1 provided that only one 12G-SDI bit rate is supported.

Also, each GTH RX and TX unit has a serial clock divider that divides the selected clock 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 lines rates by using different serial clock divider values. This is very useful for SDI interfaces because the 3G-SDI, 6G-SDI and 12G-SDI bit rates are exactly twice as fast the HD-SDI, 3G-SDI and 6G-SDI bit rates respectively. And, for 270 Mb/s SD-SDI, the GTH transceiver runs at the 3G-SDI line rate using 11X oversampling techniques. The ability of the RX and TX units to locally divide the clock source by four divisors that differ by a factor of two is important, allowing reception and 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 through the DRP by using the `RXOUT_DIV` and `TXOUT_DIV` attributes.

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

- The receivers can receive all SDI bit rates when using QPLL0 and QPLL1 to provide the serial clock derived from that reference clocks to all receivers in the quad.
- The transmitters have the flexibility to dynamically switch between the clocks from QPLL0 and QPLL1 to get both frequencies 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 run at different SDI bit rates and can dynamically switch between bit rates without disrupting the other RX or TX units.
- For genlock applications, modern genlock PLLs usually can simultaneously provide both required reference clock frequencies from the synchronization reference input signal.

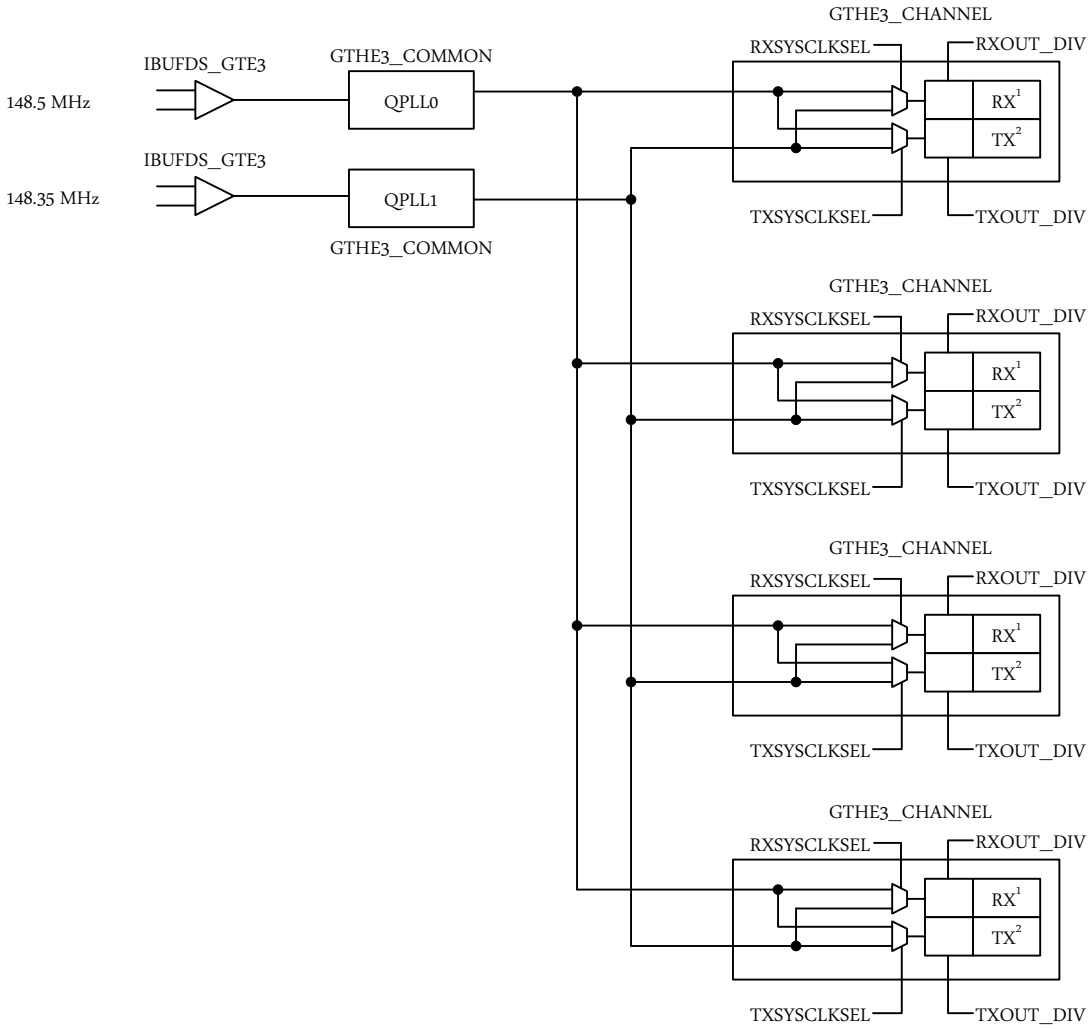


Figure 3: **Typical GTH Reference Clock Implementation for SDI**

Notes:

1. GTH RX interface and internal bit width are dynamically changed through RX_DATA_WIDTH and RX_INT_DATAWIDTH DRP attributes depending on the current SDI Mode and data stream inter-leaving pattern.
2. GTH TX interface and internal bit width are dynamically changed through TX_DATA_WIDTH and TX_INT_DATAWIDTH DRP attributes depending on the current SDI Mode and data stream inter-leaving pattern.

In some SDI applications, it might be necessary for different SDI transmitters to run at slightly different bit rates although 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 transmit at the same nominal bit rate have bit rates that differ by a few ppm. Supporting such applications is possible with the UltraScale GTH 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 GTH reference clock inputs is limited. There are two reference clock inputs per GTH quad. A quad can use reference clocks. Thus, it is possible to provide some GTH quads in the device with five different reference clock frequencies (one for the RX and four for the four TX units), but overall, there are not enough reference clock inputs for every GTH TX in the device to have its own reference clock. The phase interpolator controlled crystal or Xtal oscillator (PICXO) technique can be very

useful in these cases because it allows a GTH TX to be pulled by a few hundred ppm away from the frequency of its serial clock. Thus, applications where the bit rate of each SDI TX must be individually locked to the bit rate of the received SDI signal can be implemented by using common reference clocks as shown in Figure 3 then using the PICXO technique with each GTH TX to set the exact bit rate of each SDI transmitter individually. This application note does not cover the PICXO technique. For further information about using PICXO, contact Xilinx Technical Support at the Xilinx Support web page.

Resets

The GTH transceiver has very specific reset requirements as described in the *UltraScale Architecture GTH Transceivers User Guide* [Ref 13]. The GTH transceiver requires careful coordination of resets of the PLLs, GTH transceiver resets (`gttxreset` and `gtrxreset`). Such coordination is simplified when the GTH transceiver is generated using the UltraScale FPGAs Transceiver Wizard with the clocking network with the reset controller helper blocks enabled. The reset controller helper block handles the complicated GTH transceiver reset sequence and the control module supplied with this application note handles the reset assertions for all UHD-SDI core configuration updates to ensure proper operation of the GTH transceiver.

GTH TX Resets

The UltraScale FPGAs Transceiver Wizard offers three ways to reset the TX portion of the GTH Transceiver.

- `gtwiz_reset_all_in`: Asserted High. User signal to reset both TX and RX portions phase-locked loops (PLLs) and active data directions of GTH transceiver. This reset is affects for TX and RX GTH portions hence normally asserted during startup condition.
- `gtwiz_reset_tx_pll_and_datapath_in`: Asserted High. User signal to reset the TX data direction and associated PLLs of GTH transceiver. This reset is particularly useful if the reference clock to the TX PLL changes.
- `gtwiz_reset_tx_datapath_in`: Asserted High. User signal to reset the TX data direction of transceiver primitives. This reset is asserted for SDI TX application when at least one of the `tx_mode`, `tx_m` and tx_mux_pattern ports change.

Note: All the resets mentioned in this application note are asynchronous resets. Refer to the *Logicore IP UltraScale FPGAs Transceivers Wizard Product Guide* [Ref 15] for more information.

For use cases that use one of the QPLLs and one CPLL, the two PLL types have different operating frequency ranges. For SDI applications, the serial clocks from the QPLLs are twice the frequency of the serial clock from the CPLL. Thus, when the `tx_m` input port of the SDI wrapper changes to request a dynamic switch of the GTH TX between the two PLLs, a dynamic change of the serial clock divider through the `TXOUT_DIV` DRP attribute must also be done at the same time if the transmitter remains in the same SDI mode. For example, when switching from an HD-SDI bit rate of 1.485 Gb/s using the QPLL as the serial clock source to an HD-SDI bit rate of 1.485/1.001 Gb/s using the CPLL as the serial clock source, both the `txsysclksel` port and `TXOUT_DIV` DRP attribute must be changed. However, if the SDI mode, as selected by the `tx_mode` input port of the SDI wrapper, changes at the same time as the `tx_m` port, the serial clock divider may or may need to be changed. For example, if changing from HD-SDI mode

using the CPLL to the 3G-SDI mode using the QPLL, the txrate port does not need to change because changing from the CPLL to the QPLL inherently increases the serial clock frequency and the resulting line rate by a factor of two.

The tx_mode port dictates the data width of the GTH transceiver. For example, at 6G-SDI or 12G-SDI, the GTH internal and user interface data width have to be changed to 4 bytes and 40 bits, respectively. Same parameters are set to 2 bytes and 20 bits for the lower bit rates. The data widths of the UHD-SDI TX and TX portion of the GTH transceiver should always match. This is done in the GTH transceiver by modifying the RX_DATA_WIDTH and RX_INT_DATAWIDTH DRP attributes which affects the interface and internal data width, respectively.

Because tx_mode and tx_m are separate input ports to the SDI wrapper, when one of these ports changes, a small settling delay is implemented before the txsysclksel port, TXOUT_DIV, RX_DATA_WIDTH and RX_INT_DATAWIDTH DRP attributes are dynamically changed. This settling delay allows a short window of time for the other port to also change before the TX control logic decides whether these port and DRP attributes need to change.

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

- tx_rst_in: When asserted High, this input resets the SDI TX data path in the UHD-SDI core, TX controller module and TX portion of GTH transceiver.
- gth_wiz_reset_tx_pll_and_datapath_in: When asserted High, this input resets both the PLL associated with the TX and then the TX section of the GTH transceiver.

GTH 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.

The UltraScale FPGAs Transceiver Wizard offers three ways to reset the RX portion of the GTH Transceiver.

- gtwiz_reset_all_in: Asserted High. User signal to reset both TX and RX portions PLLs and active data directions of GTH transceiver. This reset affects TX and RX GTH portions hence normally asserted during startup condition.
- gtwiz_reset_rx_pll_and_datapath_in: Asserted High. User signal to reset the RX data direction and associated PLLs of GTH transceiver. This reset is particularly useful if the reference clock to the RX PLL changes.
- gtwiz_reset_rx_datapath_in: Asserted High. User signal to reset the RX data direction of transceiver primitives. This reset is asserted for SDI RX application ones when at least one of the rx_mode, rx_m and rx_active_streams ports change.

Note: All the resets mentioned in this application note are asynchronous resets. Refer to the *Logicore IP UltraScale FPGAs Transceivers Wizard Product Guide* [Ref 15] for more information.

All bit rates (0 ppm and 1000 ppm) from SD-SDI to 6G-SDI can be supported by one CPLL or QPLL because these PLLs offer ±1250 ppm for bit rates ≤ 6.6 Gb/s. Conversely, for 12G-SDI, two PLLs (CPLL and one of QPLL or QPLL0 and QPLL1) of GTH transceiver are required to support the

two bit rates. This means that for 12G-SDI applications, rxsysclksel must change when switching from 11.88 Gb/s to 11.88/1.001 Gb/s, and vice versa.

Changes in the SDI mode between SD-SDI, HD-SDI, 3G-SDI, 6G-SDI, and 12G-SDI require changes to one or more of the following:

- controlling the rxcdrhold port
- enabling or disabling equalization (LPM and DFE)
- updating the RXCDR_CFG attribute based on SDI mode
- RXOUT_DIV, RX_DATA_WIDTH, and RX_INT_DATA_WIDTH attributes.

The rxcdrhold port must be asserted High when the RX SDI mode is SD-SDI. The LPM and DFE must be disabled for SD-SDI and enabled for other SDI line rates. The RXCDR_CFG2 attribute is modified when switching into HD-SDI, 3G-SDI, 6G-SDI, and 12G-SDI modes to optimize the CDR for the current line rate. The RXOUT_DIV attribute control the serial clock divider for the GTH RX. The GTH RX must be reset using the gtwiz_reset_rx_datapath_in port of GT Wizard after dynamic changes are made to any of these four things. If more than one of these things changes during the same SDI mode change sequence, only a single gtwiz_reset_rx_datapath_in is required after all changes have been made.

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

- rx_rst_in: When asserted High, this input resets the SDI RX data path in the UHD-SDI core, RX controller module and RX portion of GTH transceiver.
- gth_wiz_reset_rx_pll_and_datapath_in: When asserted High, this input resets both the PLL associated with the RX and then the RX section of the GTH transceiver.

GTH PLL Use 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 sufficiently describe the proper connection of the PLL reset and locked signals.

The SDI wrapper has four static parameters that specify which serial clock sources come from the QPLLs and which come from the CPLL. These attributes do not control the routing of PLL clocks and are only used to calculate the correct RX and TX serial clock divider values and, for the TX, the value to drive onto the GTH Wizard IP rxpllclksel_in and txpllclksel_in ports based on the current value of rx_m and tx_m respectively. These four parameters are two bit binary values and must be assigned values as described here:

- The RXPLLCLKSEL_RX_M_0 parameter must be set to 2'b00 (CPLL) or 2'b11 (QPLL0) or 2'b10 (QPLL1) depending on the clock source for the GTH RX when rx_m is Low.
- The RXPLLCLKSEL_RX_M_1 parameter must be set to 2'b00 (CPLL) or 2'b11 (QPLL0) or 2'b10 (QPLL1) depending on the clock source for the GTH RX when rx_m is High and rx_mode is 3'b110 (12G 11.88/1.001 Gb/s).
- The TXPLLCLKSEL_TX_M_0 parameter must be set to 2'b00 (CPLL) or 2'b11 (QPLL0) or 2'b10 (QPLL1) depending on the clock source for the GTH TX when tx_m is Low.

- The TXPLLCLKSEL_TX_M_1 parameter must be set to 2'b00 (CPLL) or 2'b11 (QPLL0) or 2'b10 (QPLL1) depending on the clock source for the GTH TX when tx_m is High.

There are two parameters for the RX clock to support dynamic switching of the RX between the two PLL clock sources using the rx_m port of the SDI wrapper. RXPLLCLKSEL_RX_M_0 is used to drive the rxpllclksel_in of GT Wizard IP when tx_m is Low and RXPLLCLKSEL_RX_M_1 is used when rx_m is High and rx_mode is 3'b110 (12G-SDI /1.001). In applications where the RX PLL is not dynamically switched, set the same value for both RXPLLCLKSEL_RX_M_0 and RXPLLCLKSEL_RX_M_1 depending on clock source to RX PLL.

Similar to RX, there are two parameters for the TX clock to support dynamic switching of the TX between the two PLL clock sources using the tx_m port of the SDI wrapper. TXPLLCLKSEL_TX_M_0 is used to drive the txpllclksel_in when tx_m is Low and TXPLLCLKSEL_TX_M_1 is used when tx_m is High. In applications where the TX PLL is not dynamically switched, set the same value for both TXPLLCLKSEL_TX_M_0 and TXPLLCLKSEL_TX_M_1 depending on clock source to TX PLL.

Use Model 1: A Single Transceiver Active in the Quad, RX and TX are Dynamically Clocked by QPLL0 and QPLL1

In this use model, shown in Figure 4, there is a single transceiver active in the quad with the RX and TX serial clocks provided by either QPLL0 or QPLL1. This use model is the recommended clocking if both 12G-SDI bit rates are supported. This use model is the recommended clocking for both SDI TX and SDI RX if either 12G-SDI integer bit rate or 12G-SDI fractional bit rate support but not both is required. CPLL is required to implement different protocol other than SDI in other transceivers of the same quad.

The following connections and configurations must be made:

- Connect the reference clocks, 148.5 MHz and 148.35 MHz to the gth_qpll0_refclk_p/n_in and gth_qpll1_refclk_p/n_in ports respectively.
- The gth_cpll_refclk_p_in and gth_cpll_refclk_n_in ports must be connected to zero.
- The gth_drpclk_in must be connected to clock specified during GTH Wizard IP generation, in this application note it is 27 MHz.
- gth_wiz_reset_tx_pll_and_datapath_in and gth_wiz_reset_rx_pll_and_datapath_in input ports must be Low only when the reference clock source to the QPLL0 and QPLL1 are stable.
- The RXPLLCLKSEL_RX_M_0 parameter of the SDI Wrapper Support must be set to 2'b11 (QPLL0).
- The RXPLLCLKSEL_RX_M_1 parameter of the SDI Wrapper Support must be set to 2'b10 (QPLL1).
- The TXPLLCLKSEL_TX_M_0 parameter of the SDI Wrapper Support must be set to 2'b11 (QPLL0).
- The TXPLLCLKSEL_TX_M_1 parameter of the SDI Wrapper Support must be set to 2'b10 (QPLL1).

- When the QPLL0 must be reset due to a reference clock change or interruption, assert the `gth_qpll0_reset_in` input of the SDI Wrapper Support.
- When the QPLL1 must be reset due to a reference clock change or interruption, assert the `gth_qpll1_reset_in` input of the SDI Wrapper Support.

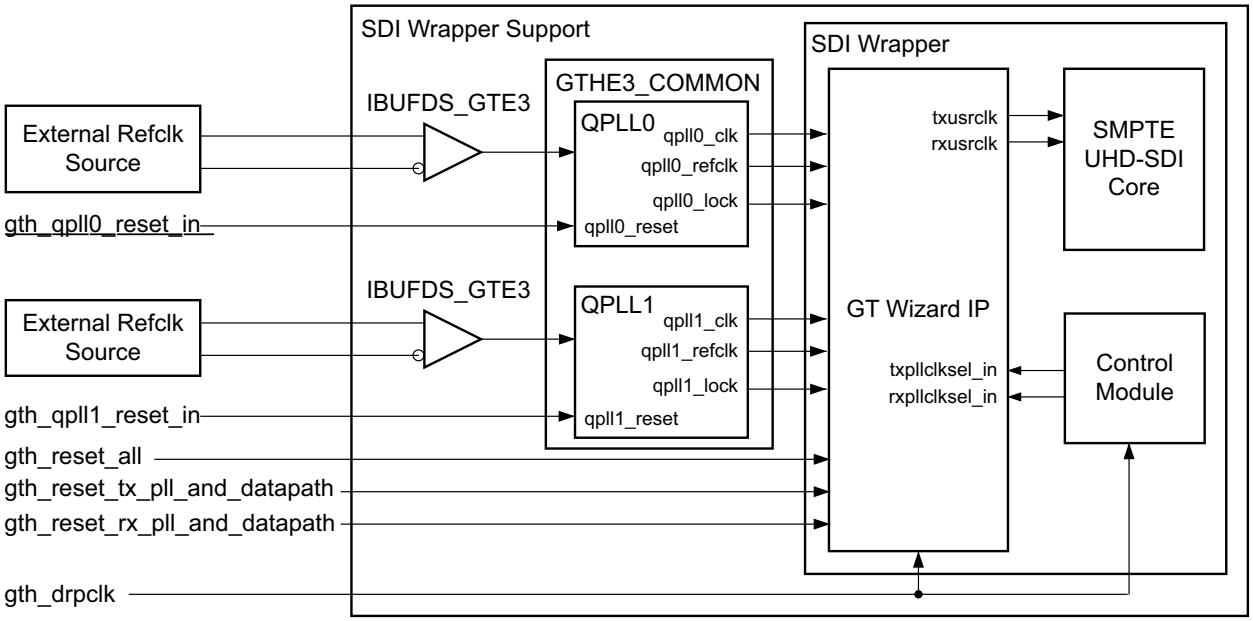


Figure 4: **PLL Use Model 1 and Model 2**

Use Model 2: A Single Transceiver Active in the Quad, RX Clocked by the QPLL1, TX Clocked by the QPLL0

In this use model, shown in Figure 4, there is a single transceiver active in the quad with the GTH RX clocked by the QPLL1 and the GTH TX clocked by the QPLL0. This use model is the recommended clocking, if either 12G-SDI integer bit rate or 12G-SDI fractional bit rate support but not both is required for both SDI TX and SDI RX. The CPLL is required to implement different protocol other than SDI in other transceivers of the same quad.

The following connections must be made:

- Connect the reference clocks, to the `gth_qpll0_refclk_p/n_in` and `gth_qpll1_refclk_p/n_in` ports.
- The `gth_cpll_refclk_p_in` and `gth_cpll_refclk_n_in` ports must be connected to zero.
- The `gth_drpclk_in` must be connected to clock specified during GTH Wizard IP generation, in this application note it is 27 MHz.
- The `gth_wiz_reset_tx_pll_and_datapath_in` input port must be Low only when the reference clock source to the QPLL0 is stable.
- The `gth_wiz_reset_rx_pll_and_datapath_in` input port must be Low only when the reference clock source to the QPLL1 is stable.

- The RXPLLCLKSEL_RX_M_0 parameter of the SDI Wrapper Support must be set to 2'b10 (QPLL1).
- The RXPLLCLKSEL_RX_M_1 parameter of the SDI Wrapper Support must be set to 2'b10 (QPLL1).
- The TXPLLCLKSEL_TX_M_0 parameter of the SDI Wrapper Support must be set to 2'b11 (QPLL0).
- The TXPLLCLKSEL_TX_M_1 parameter of the SDI Wrapper Support must be set to 2'b11 (QPLL0).
- When the QPLL0 must be reset due to a reference clock change or interruption, assert the gth_qpll0_reset_in input of the SDI Wrapper Support.
- When the QPLL1 must be reset due to a reference clock change or interruption, assert the gth_qpll1_reset_in input of the SDI Wrapper Support.

Use Model 3: Multiple Transceivers Active in the Quad, All RX and TX are Dynamically Clocked by QPLL0 and QPLL1

In this use model, shown in Figure 5, there are multiple transceivers active in the quad. All GTH receivers are clocked by the QPLL. All of the GTH transmitters can be independently switched between the QPLL0 and QPLL1. This model conforms to the standard use model shown in Figure 3.

In this use model, one SDI Wrapper Support is instantiated which contains the GTHE3 Common Primitive and corresponding differential clock buffers. Multiple SDI Wrappers are instantiated (up to three) for other SDI channels.

This use model covers a very common case where multiple transceivers are active in the quad, all implementing SDI interfaces. All the active GTH RX and TX units in the quad use the serial clock from the QPLL0 or QPLL1. This usage module is shown in Figure 5. This multi-link use model is the recommended clocking for both SDI TX and SDI RX if both 12G-SDI bit rates (integer and fractional) support is required. CPLL is required to implement different protocol other than SDI in other transceivers of the same quad if less than four transceivers are used.

In this use model, the SDI Wrapper Support is designated as the QPLL0 and QPLL1 master and controls the QPLL0RESET and QPLL1RESET ports of the GTH Common Primitive. The SDI Wrappers do not control the QPLL reset, but they do monitor the QPLL0 and QPLL1 locked output of the SDI Wrapper Support.

The following connections must be made:

- Connect the reference clocks, 148.5 MHz and 148.35 MHz to the gth_qpll0_refclk_p/n_in and gth_qpll1_refclk_p/n_in ports respectively.
- The gth_cpll_refclk_p_in and gth_cpll_refclk_n_in ports must be connected to zero.
- The gth_drpclk_in must be connected to clock specified during GTH Wizard IP generation, in this application note it is 27 MHz.

- `gth_wiz_reset_tx_pll_and_datapath_in` and `gth_wiz_reset_rx_pll_and_datapath_in` input ports must be Low only when the reference clock source to the QPLL0 and QPLL1 are stable.
- The `RXPLLCLKSEL_RX_M_0` parameter of the SDI Wrapper Support and SDI Wrapper must be set to 2'b11 (QPLL0).
- The `RXPLLCLKSEL_RX_M_1` parameter of the SDI Wrapper Support and SDI Wrapper must be set to 2'b10 (QPLL1).
- The `TXPLLCLKSEL_TX_M_0` parameter of the SDI Wrapper Support and SDI Wrapper must be set to 2'b11 (QPLL0).
- The `TXPLLCLKSEL_TX_M_1` parameter of the SDI Wrapper Support and SDI Wrapper must be set to 2'b10 (QPLL1).
- When the QPLL0 must be reset due to a reference clock change or interruption, assert the `gth_qpll0_reset_in` input of the SDI Wrapper Support.
- When the QPLL1 must be reset due to a reference clock change or interruption, assert the `gth_qpll1_reset_in` input of the SDI Wrapper Support.
- The SDI Wrapper Support `qpll0/1_clk`, `qpll0/1_refclk` and `qpll0/1_lock` output ports must be connected to their corresponding ports in the SDI Wrapper.

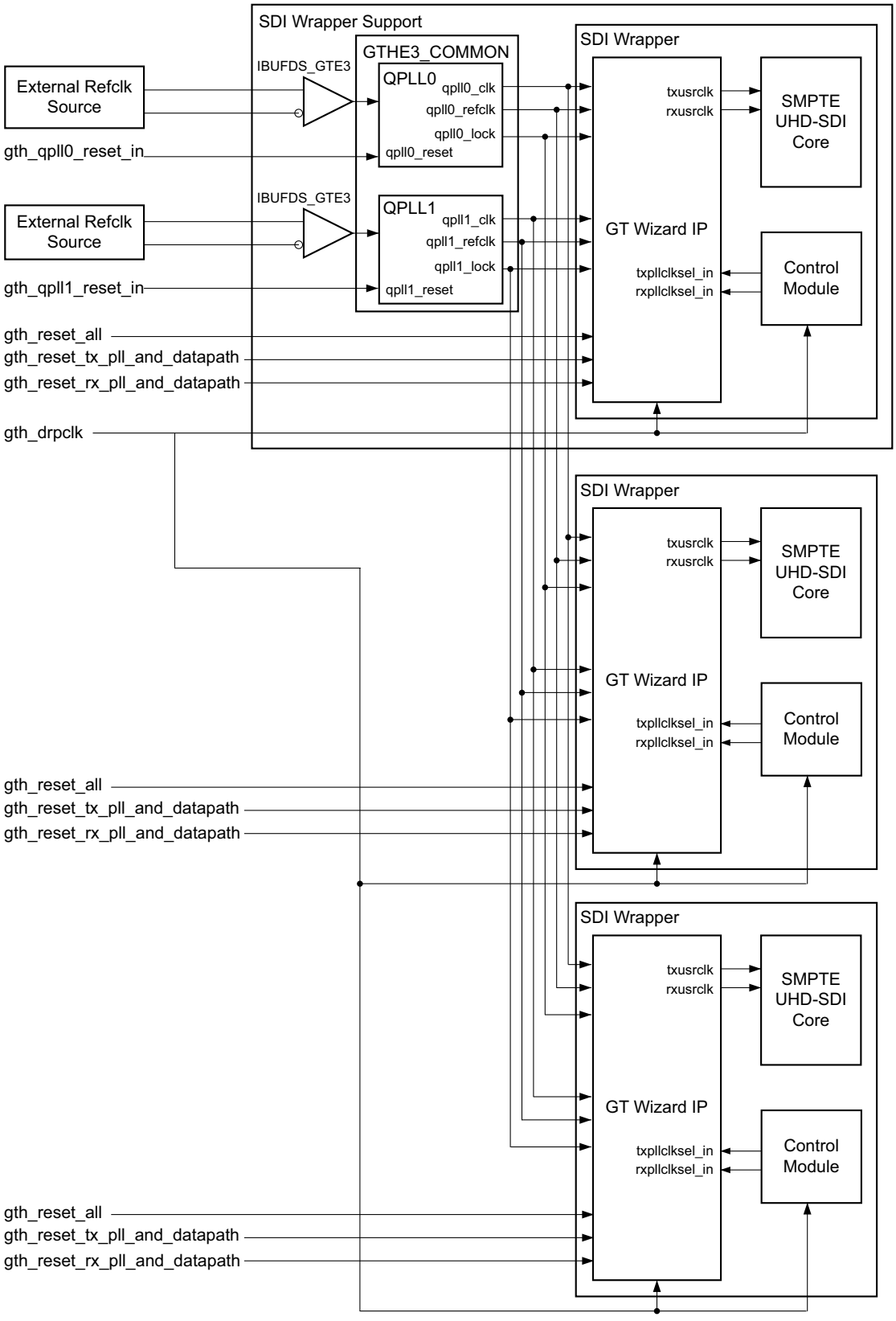


Figure 5: **PLL Use Model 3 and Model 4**

Use Model 4: Multiple Transceivers are Active in a Quad, All RX Use the QPLL1, All TX Use the QPLL0

In this use model, shown in Figure 5, there are multiple transceivers active in the quad. All of the receivers are clocked by the QPLL1. Each transmitter is clocked only by QPLL0. This multi-link use model is the recommended clocking for SDI TX and SDI RX if either 12G-SDI integer bit rate or 12G-SDI fractional bit rate support but not both is required. CPLL is required to implement different protocol other than SDI in other transceivers of the same quad if less than four transceivers are used.

The following connections must be made:

- Connect the reference clocks, to the `gth_qpll0_refclk_p/n_in` and `gth_qpll1_refclk_p/n_in` ports.
- The `gth_cpll_refclk_p_in` and `gth_cpll_refclk_n_in` ports must be connected to zero.
- The `gth_drpclk_in` must be connected to clock specified during GTH Wizard IP generation, in this application note it is 27 MHz.
- The `gth_wiz_reset_tx_pll_and_datapath_in` input port must be Low only when the reference clock source to the QPLL0 is stable.
- The `gth_wiz_reset_rx_pll_and_datapath_in` input port must be Low only when the reference clock source to the QPLL1 is stable.
- The `RXPLLCLKSEL_RX_M_0` parameter of the SDI Wrapper Support and SDI Wrapper must be set to 2'b10 (QPLL1).
- The `RXPLLCLKSEL_RX_M_1` parameter of the SDI Wrapper Support and SDI Wrapper must be set to 2'b10 (QPLL1).
- The `TXPLLCLKSEL_TX_M_0` parameter of the SDI Wrapper Support and SDI Wrapper must be set to 2'b11 (QPLL0).
- The `TXPLLCLKSEL_TX_M_1` parameter of the SDI Wrapper Support and SDI Wrapper must be set to 2'b11 (QPLL0).
- When the QPLL0 must be reset due to a reference clock change or interruption, assert the `gth_qpll0_reset_in` input of the SDI Wrapper Support.
- When the QPLL1 must be reset due to a reference clock change or interruption, assert the `gth_qpll1_reset_in` input of the SDI Wrapper Support.
- The SDI Wrapper Support `qpll0/1_clk`, `qpll0/1_refclk` and `qpll0/1_lock` output ports must be connected to their corresponding ports in the SDI Wrapper.

Use Model 5: A Single Transceiver Active in the Quad, RX Is Clocked by QPLL1 and TX Is Dynamically Clocked by QPLL1 and CPLL

In this use model, shown in Figure 6, there is a single transceiver active in the quad with the RX serial clock provided by QPLL1 and TX serial clock provided by either QPLL1 or CPLL. This use model is the recommended clocking for both SDI TX and SDI RX if either 12G-SDI integer bit rate or 12G-SDI fractional bit rate support but not both is required. QPLL0 is required to implement different protocol other than SDI in other transceivers of the same quad.

The following connections must be made:

- Connect one reference clock to the `gth_qpll1_refclk_p_in` and `gth_qpll1_refclk_n_in` ports.
- Connect one reference clock to the `gth_cpll_refclk_p_in` and `gth_cpll_refclk_n_in` ports.
- The `gth_qpll0_refclk_p_in` and `gth_qpll0_refclk_n_in` ports must be connected to zero.
- The `gth_drpclk_in` must be connected to clock specified during GTH Wizard IP generation, in this application note it is 27 MHz.
- The `gth_wiz_reset_tx_pll_and_datapath_in` input port must be Low only when the reference clock source to the QPLL1 and CPLL are stable.
- The `gth_wiz_reset_rx_pll_and_datapath_in` input port must be Low only when the reference clock source to the QPLL1 is stable.
- The `RXPLLCLKSEL_RX_M_0` parameter of the SDI Wrapper Support must be set to 2'b10 (QPLL1).
- The `RXPLLCLKSEL_RX_M_1` parameter of the SDI Wrapper Support must be set to 2'b10 (QPLL1).
- The `TXPLLCLKSEL_TX_M_0` parameter of the SDI Wrapper Support must be set to either 2'b10 (QPLL1) or 2'b00 (CPLL).
- The `TXPLLCLKSEL_TX_M_1` parameter of the SDI Wrapper Support must be set to either 2'b00 (CPLL) or 2'b10 (QPLL1) depending on the reference clock connection and which is not used on `TXPLLCLKSEL_TX_M_0`.
- When the QPLL1 must be reset due to a reference clock change or interruption, assert the `gth_qpll1_reset_in` input of the SDI Wrapper Support.

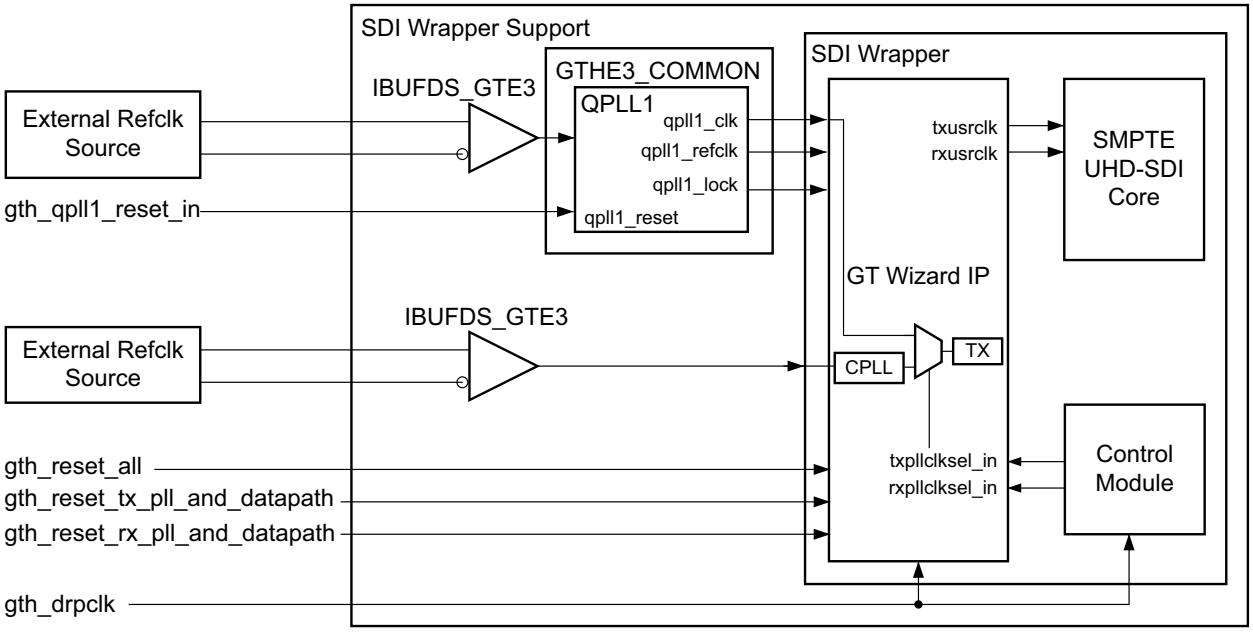


Figure 6: **PLL Use Model 5**

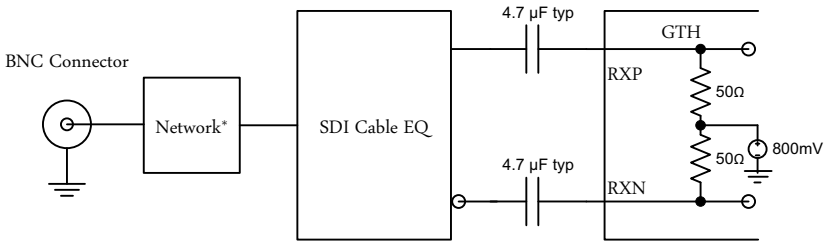
SDI Electrical Interface

External SDI cable equalizers and cable drivers are required to convert the serial signals into and out of the GTH 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 GTH transceiver. Appropriate SDI cable equalizers are available from several manufacturers. The differential outputs of these cable equalizers usually must be AC-coupled to the GTH receiver input signals due to common mode voltage differences. An example of interfacing a typical SDI cable equalizer to a GTH receiver is shown in Figure 7.

IMPORTANT: *The capacitance values of the AC coupling capacitors between the outputs of the external SDI cable equalizer and the serial inputs of the GTH 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 recommended.*

The differential inputs of the GTH RX have built-in differential termination. As described in *UltraScale Architecture GTH Transceivers User Guide* [Ref 13], RX Termination Use Mode 3 is the recommended termination mode for the GTH RX inputs in SDI applications. The GTH internal programmable termination voltage should be set to 800 mV for SDI applications.



*Consult the SDI Cable EQ manufacturer's information for the network between the SDI Cable EQ and the BNC connector.

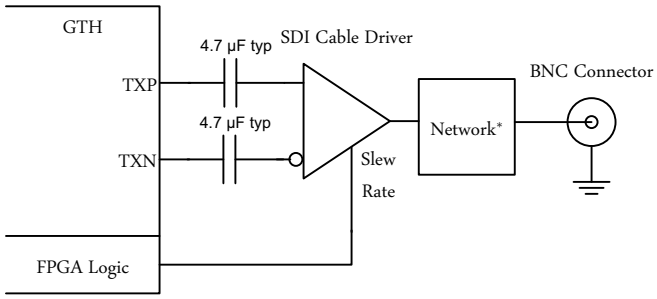
Figure 7: **Interfacing a SDI Cable Equalizer to the GTH Receiver Inputs**

Notes:

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 GTH transmitter are connected to the inputs of a SDI cable driver, usually with AC coupling as shown in Figure 8. The cable driver converts the differential signal from the GTH 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, 3G-SDI, 6G-SDI, and 12G-SDI. The slew rate control input of the SDI cable driver is typically controlled by the FPGA, however 12G-SDI FMC expansion card used with this application note internally controls the slew rate of the cable driver. The control module supplied with this application note generates a slew rate control signal for use with the external SDI cable driver for other use cases.

IMPORTANT: *The capacitance values of the AC coupling capacitors between the GTH TX serial outputs and the inputs of the external 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.*



*Consult the SDI Cable Driver manufacturer's information for the network between the SDI Cable Driver and the BNC connector.

Figure 8: **Interfacing a SDI Cable Driver to the GTH Transmitter Outputs**

Notes:

1. Consult the SDI cable EQ manufacturer's information for the network between the SDI cable EQ 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 GTH RX. To receive 270 Mb/s SD-SDI, the GTH RX is used as an asynchronous oversampler to sample the SD-SDI bit stream at 11 times 270 Mb/s (2.97 G samples/sec) without regard to where bit transitions occur. The CDR unit in the GTH RX is locked to the reference clock by asserting the GTH transceiver's `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.

When receiving an SD-SDI signal, the auto adaptation feature of the low power mode (LPM) and Decision feedback equalization (DFE) equalizer must be disabled. The long run lengths at the slow bit rate cause problems for the equalizers. The LPM auto adaptation feature is disabled by asserting the following ports of the `GTHE3_CHANNEL` primitive High:

- RXLPMGCOVRDEN
- RXLPMHFOVRDEN
- RXLPMLFKLOVRDEN
- RXLPMOSOVRDEN
- RXOSOVRDEN

The DFE equalization is disabled by asserting the following ports of the `GTHE3_CHANNEL` primitive High:

- RXDFEAGCOVRDEN
- RXDFELFOVRDEN
- RXDFETAP2OVRDEN
- RXDFETAP3OVRDEN
- RXDFETAP4OVRDEN
- RXDFETAP5OVRDEN
- RXDFETAP6OVRDEN
- RXDFETAP7OVRDEN
- RXDFETAP8OVRDEN
- RXDFETAP9OVRDEN
- RXDFETAP10OVRDEN
- RXDFETAP11OVRDEN
- RXDFETAP12OVRDEN
- RXDFETAP13OVRDEN
- RXDFETAP14OVRDEN

- RXDFETAP15OVRDEN
- RXDFEUTOVRDEN

With UltraScale FPGAs Transceiver Wizard, these ports are not enabled by default on the GTH Wizard IP and need to be manually enabled. These ports can be found in the Structural Options tab of the wizard with `_in` suffixes as port names.The easiest thing to do is to connect the `rxcdrhold_in` port of the GTH wrapper to these ports of the GTH Wizard IP. The `rxcdrhold_in` port is driven High by the SDI control logic when the receiver is in SD-SDI mode, so these three ports are driven High in SD-SDI mode if connected in this manner.

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

The DRU provided with this application note is described in the Xilinx application note *Clock and Data Recovery Unit based on 20-Bit-Wide Oversampled Data* [Ref 18]. That application note does describe the theory of operation of the DRU, but is not necessary for use of the DRU in the UHD-SDI reference design.

SMPTE ST 259 (the SD-SDI standard) specifies several other bit rates besides 270 Mb/s. The DRU is instantiated into the SDI control module so as to support only 11x oversampling of 270 Mb/s serial data. However, if other SD-SDI bit rates need to be supported by the application, the DRU can used to receive those bit rates as well. 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 is within the supported line rate range of the GTH transceiver and thus the GTH RX does not need to use the DRU to receive it. However, receiving the 540 Mb/s bit rate without the DRU requires a different reference clock frequency than is used for the other SDI bit rates. Thus, it is usually more convenient to use the DRU to receive the 540 Mb/s ST 344 signal using 5.5X oversampling so that the standard SDI reference clock frequency can be used. Xilinx does not have an example design supporting additional SD-SDI bit rates.

The DRU does not recover a clock and, because the CDR unit in the GTH RX is locked to its reference clock, the `rxusrclk` is not locked to the incoming bit rate in SD-SDI mode. The DRU produces a data strobe in `rxusrclk` clock domain indicating when a 10-bit data word is ready on its output. This data strobe is used by the UHD-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 `rxusrclk` clock from the GTH. The `rx_ce_out` output of the `v_smpte_uhdsdi_rxtx` wrapper during SD-SDI operations 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 must make up for the slight difference between the actual SD-SDI bit rate and the frequency of the local reference clock provided to the PLL used by the GTH RX.

Figure 9 shows a screen capture from an oscilloscope showing the 27 MHz `rx_ce_out` port during SD-SDI operation.

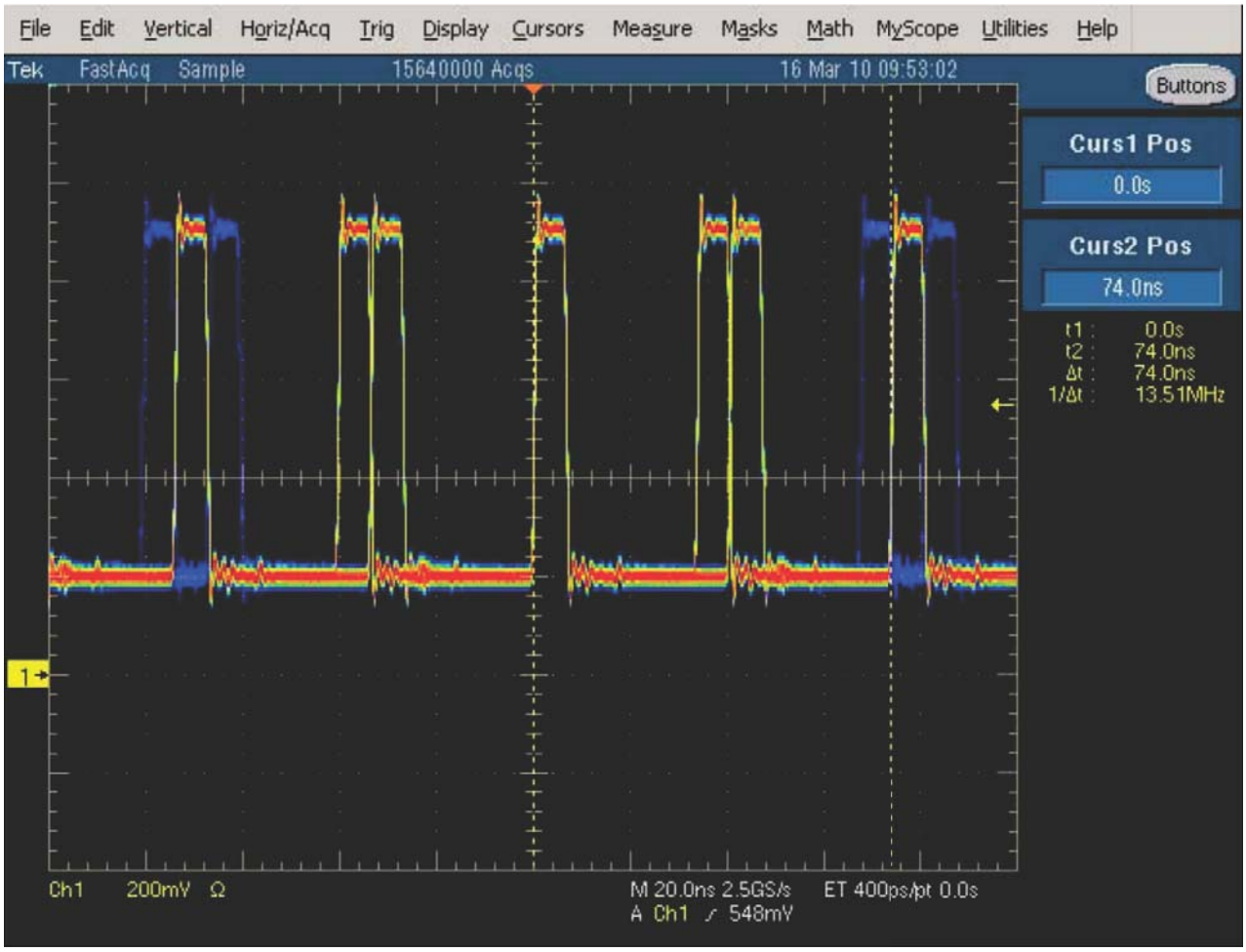


Figure 9: **Oscilloscope Capture of SD-SDI Clock Enable**

The scope is triggered on the rising edge of `rx_ce_out` 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 GTH receiver. The `rx_ce_out` 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_out` port.

The two pulses at the far right and far left of the trace are nominally 11 clock cycles from the center pulse because of the 5|6|5|6 cadence. The nominal position is marked by the yellow and red pulse. For the far right pulse, the dashed yellow vertical cursor marks the position that is 11 clock cycles from the rising edge of the center pulse. The nominal location of the central yellow/red pulses are surrounded on either side by blue pulses indicating that, occasionally, the DRU must make the period of the `rx_ce_out` 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. Refer to the *SMPTE UHD-SDI Product Guide* [Ref 17] for more information.

The SD-SDI DRU is supplied with this application note as an encrypted, pre-generated file called `nidru_20_wrapper.vhd`. The encryption used on the DRU is compatible with most synthesis and simulation software.

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 GTH TX. To transmit the SD-SDI signal, the GTH TX is configured for a line rate of 2.97 Gb/s. The UHD-SDI core replicates each bit to be transmitted 11 times so that the data out of the UHD-SDI core and into the `gth_txn_out` port of the GTH Wizard IP contains 11 consecutive copies of each bit. The resulting signal output by the GTH TX is a valid 270 Mb/s SD-SDI signal.

Generating an SD-SDI Recovered Clock

In SD-SDI mode, the `rxusrclk` of the GTH RX is technically not 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_out` 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_out` port may be sufficient as a recovered clock. Typically, this signal is used as a clock enable to downstream modules that are clocked with the `rxusrclk` from the GTH RX, similar to how the SDI data path in the UHD-SDI core works by using the `rx_ce_out` port as a clock enable.

If the received video data is to be retransmitted as an SD-SDI signal using a GTH 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 PLL generating the serial clock for the GTH TX. Furthermore, the frequency of the recovered clock must be 148.5 MHz so that the GTH TX can use 11x oversampling to transmit the 270 Mb/s SD-SDI data. This requires the use of an external, low-bandwidth PLL that can perform jitter attenuation. The bandwidth of the mixed-mode clock manager (MMCM) in the UltraScale is too high to adequately filter out the large amounts of low frequency jitter present on the `rx_ce_out` port from the SDI receiver. The Texas Instruments LMH1983 and the Silicon Labs Si5328 can both perform this function. Both of these devices can take in the `rx_ce_out` port as a 27 MHz reference and multiply it up to 148.5 MHz while also filtering out the jitter. The resulting clock is suitable for use as a reference clock for the GTH TX. The pass-through demo included with this application note uses a Si5328 to generate a 148.5 MHz reference clock for the GTH TX from the 27 MHz `rx_ce_out` port in exactly this manner in SD-SDI mode. When retransmitting HD-SDI, 3G-SDI, 6G-SDI, or 12G-SDI, the same Si5328 is reprogrammed to filter jitter from the `rxusrclk` output of the GTH RX, doubling its frequency in the case of HD-SDI, thereby producing a low-jitter 148.5 MHz reference clock for the GTH 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 a SD-SDI recovered clock.

A recovered clock is sometimes required to drive external video application-specific standard product (ASSP) devices. In SD-SDI mode, such a clock probably must have a frequency of

27 MHz and have lower jitter than is present on the rx_ce_out port, but does not need to have very low jitter as is the case when producing a GTH TX reference clock. The previously-mentioned techniques can be used, but it may be preferable to generate such a recovered clock entirely in the FPGA without requiring external components. Unfortunately, the jitter on the rx_ce_out port is too high to allow it to be used directly as a reference clock input to the UltraScale MMCM. But, there is a way to generate a recovered SD-SDI clock using a spare GTH TX as shown in Figure 10.

The control module's recclk_txdata port can be connected to the gtwiz_userdata_tx_in port of a extra GTH TX of GTH Wizard IP. The GTH TX must use the same reference clock as the GTH RX that is receiving the SDI input signal. The rxusrclk can be routed to gtrefclk0_in of the GT Wizard IP and the txpllclksel_in must be set to use CPLL. The GTH TX must be configured for a line rate of 2.97 Gb/s with no encoding and with a 20-bit gtwiz_userdata_tx_in port.

When configured in this manner, the serial output of the GTH TX 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 GTH TX serial output pins can be connected to a global or regional clock LVDS input of the UltraScale FPGA, with appropriate care to properly terminate the CML 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 data paths. The signal has low enough jitter that it can be used as a reference clock to a MMCM.

The recclk_txdata port of the DRU is not wired from the SDI control module to an output port of the SDI wrapper. However, if an application must use this feature, the SDI wrapper can be edited to add this output port.

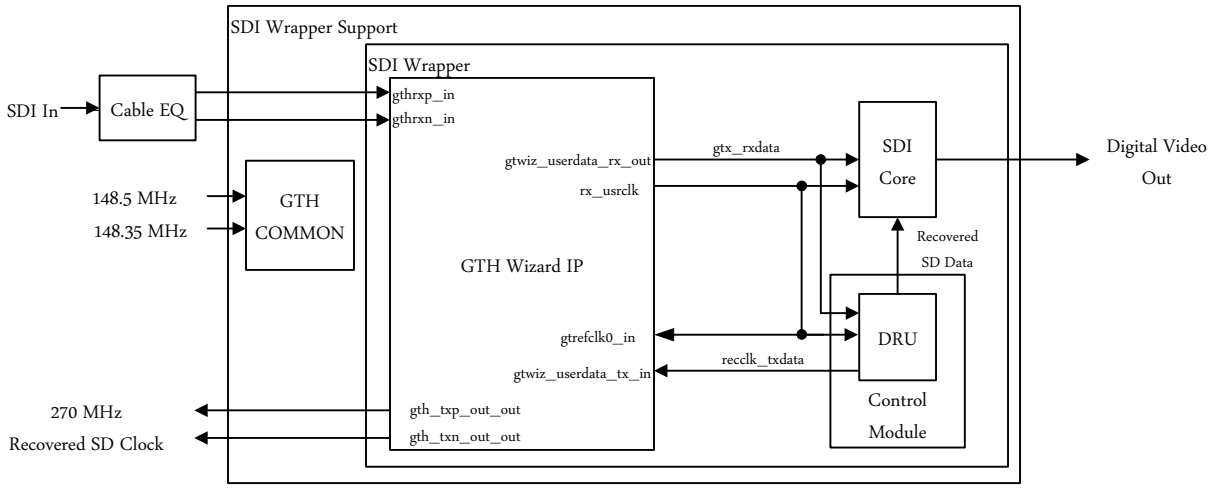


Figure 10: **Using a GTH TX to Generate a SD-SDI Recovered Clock**

The GTH TX that is used to generate the recovered SD-SDI clock does not have to be configured for SDI. It only must be configured to always run a 2.97 Gb/s with no encoding. The data supplied to the gtwiz_userdata_tx_in port of the GTH Wizard IP from the recclk_txdata port of the control module creates a 270 MHz clock on the GTH TX serial output pins. The edges of the generated clock move around by ± 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 GTH TX is ±337ps plus whatever jitter is inherent in the GTH TX output signal (1 bit time at 2.97 Gb/s is 337 ps), seen in Figure 11.

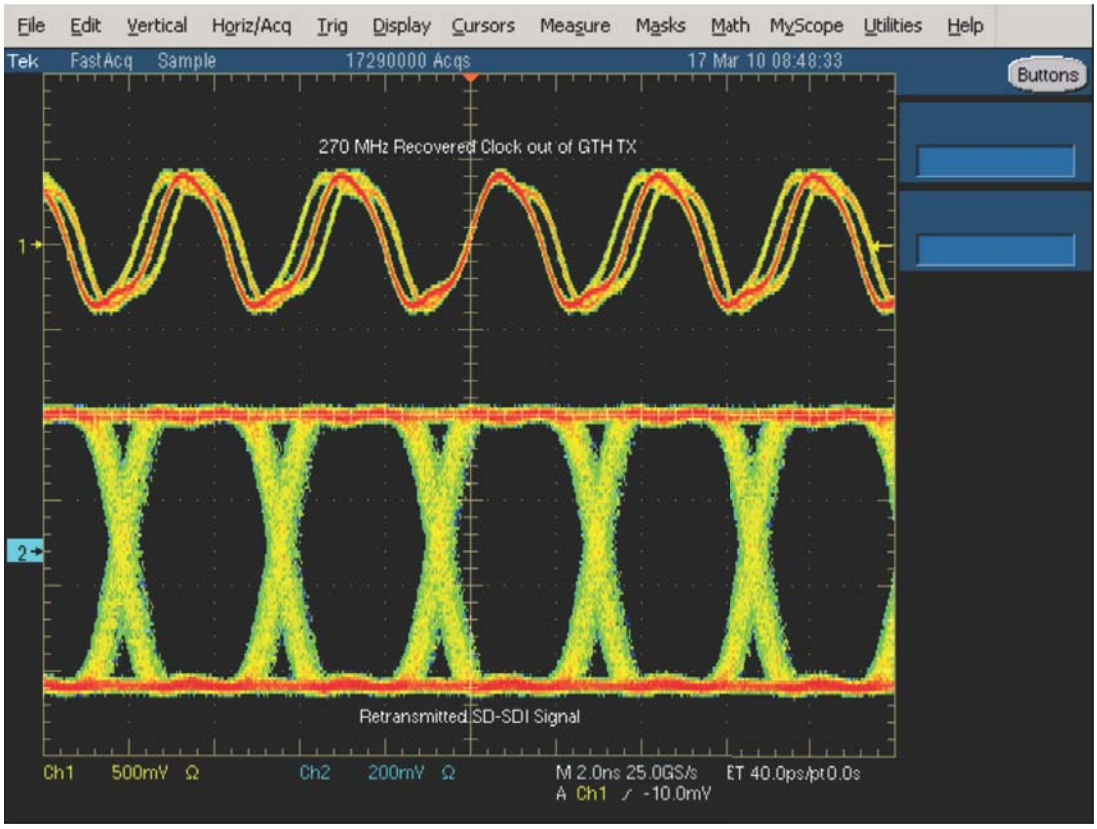


Figure 11: **Recovered SD-SDI Clock from GTH Transceiver**

In Figure 11, the top trace is the 270 MHz clock generated by the GTH 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, it is easy to see the ±-337ps cycle-to-cycle jitter as these rising edges each have three discrete rising points. The bottom trace is the SD-SDI that is being retransmitted by another GTH TX.

Automatic RX SDI Mode Detection

The UHD-SDI core can automatically determine the SDI mode (SD, HD, 3G, 6G, or 12G-SDI) of the SDI signal coming into the GTH RX. When it is not locked to the current SDI input signal, the UHD-SDI core sequences the GTH RX through the five different SDI modes until it detects recognizably good SDI data on the `rxdata` output port of the GTH. At that point, the UHD-SDI core indicates that the GTH CDR is locked to the SDI signal by asserting its `rx_mode_locked_out` port. And, it indicates which SDI mode the RX is locked to on its `sdi_mode_out` port.

It is important to understand that the `rx_mode_locked` signal is an indication of whether or not the UHD-SDI core thinks that the GTH RX is locked to the SDI signal, and nothing more. It is really just an indication of whether the UHD-SDI core's mode search state machine is still

searching for the correct SDI mode or not. Because of this, rx_mode_locked should not be considered as an absolute indicator as to the locked status of the UHD-SDI RX.

When the GTH RX is not locked to the input SDI signal and the UHD-SDI core is actively controlling GTH RX in an effort to determine the correct SDI mode, the rx_mode_locked signal may briefly become asserted. This happens if the incoming data randomly appears to be a valid SAV sequence. If an SAV sequence is detected, the UHD-SDI core asserts rx_mode_locked and pause the mode search expecting more good data to be received. If, however, good data is not received within a specific timeout period, the rx_mode_locked signal is negated and the SDI mode search resumes.

The SDI mode search algorithm only attempts to lock to SDI modes that are enabled by the rx_mode_en_in port of the UHD-SDI wrapper. This 6-bit port has unary bits that enable HD-SDI (bit 0), SD-SDI (bit 1), 3G-SDI (bit 2), 6G-SDI (bit 3), 12G-SDI at 11.88 Gb/s (bit 4), and 12G-SDI at 11.88/1.001 Gb/s (bit 5). Because the GTH RX must be configured with different reference clock frequencies for the two 12G-SDI line rates, the two 12G-SDI line rates are treated as different SDI modes by the mode search algorithm. And, because there are separate enable bits on the rx_mode_en_in port, it is possible to specify that only one of the two 12G-SDI line rates should be included in the mode search. This is useful for those applications where it is undesirable to have the reference clock frequency of the QPLL frequently changed as the GTH RX scans through the two 12G-SDI line rates.

The rx_mode_en_in port can be changed dynamically. However, if the UHD-SDI RX is already locked to a mode that becomes disabled by dynamically clearing its bit on the rx_mode_en_in port, this does not automatically kick the UHD-SDI RX out of that mode. The UHD-SDI RX remains locked in the SDI mode until the input SDI signal changes or the UHD-SDI RX is reset, forcing the SDI mode search algorithm to try and identify the SDI mode using the new settings of the rx_mode_en_in port.

It is possible to disable the automatic SDI mode search algorithm of the UHD-SDI core. The mode search algorithm is only enabled when the rx_mode_detect_en_in port is High. If this port is Low, then the UHD-SDI RX must be told what SDI mode to operate in through the rx_forced_mode_in port. When rx_mode_detect_en_in is Low and the SDI mode search algorithm is disabled, the SDI RX is in the mode specified by the rx_forced_mode_in port and the rx_mode_locked output is always High. Thus, rx_mode_locked cannot be used as a locked indicator or a data valid indicator in this mode. When the mode search algorithm is disabled, dynamic changes on rx_forced_mode_in causes the SDI control logic to dynamically change the settings of the GTH RX as necessary for the new SDI mode.

RX Bit Rate Detection

The UHD-SDI core can automatically determine the SDI mode (SD-SDI, HD-SDI, 3G-SDI, 6G-SDI, and 12G-SDI) of the SDI signal received by the GTH RX. When it is not locked to the current SDI input signal, the UHD-SDI core sequences the GTH RX through the five different SDI modes until it detects recognizably valid SDI data on the gtwiz_userdata_rx_out output port of the GTH Wizard IP. At that point, the UHD-SDI core indicates that it is locked to the SDI signal by asserting its rx_mode_locked output. It also indicates which SDI mode the RX is locked to on its rx_mode output port. In HD-SDI, 3G-SDI, and 6G-SDI modes, the GTX RX gives no indication of whether it is receiving an integer frame rate or a fractional frame rate SDI signal.

This means it cannot differentiate between 1.485 Gb/s and 1.485/1.001 Gb/s in HD-SDI mode, for example.

However, when the UHD-SDI core is in HD-SDI, 3G-SDI, 6G-SDI, and 12G-SDI mode, it has no way of determining if the bit rate of the input SDI signal is bitrate/1 or bitrate/1.001 (e.g for 6G-SDI; 5.94 Gb/s or 5.94/1.001 Gb/s). However, the control module supplied with this application note contains a bit rate detector that can distinguish between 1.485 Gb/s and 1.485/1.001 Gb/s; between 2.97 Gb/s and 2.97/1.001 Gb/s; between 5.94 Gb/s and 5.94/1.001 Gb/s and between 11.88 Gb/s and 11.88/1.001 Gb/s. The SDI wrapper output port `rx_m_out` is Low when the input SDI signal's bit rate is bitrate/1. And, `rx_m_out` is High when the input SDI signal's bit rate is bitrate/1.001.

For the bit rate detection feature to work, the SDI wrapper must be supplied with a fixed-frequency clock on its `rx_fxdclk_in` input port. It is recommended that the frequency of this clock be at least 10 MHz. If the frequency is over 150 MHz, it may be difficult to meet timing in the bit rate detection logic. The SDI wrapper has a parameter called `FXDCLK_FREQ` that must be used to specify the frequency of the clock connected to the `rx_fxdclk_in` port. The value of `FXDCLK_FREQ` must be set equal to the frequency of the fixed frequency clock in Hz.

Implementing an SDI Interface in an UltraScale Device

To implement an SDI interface in an UltraScale FPGA design, perform the following steps:

1. Generate GTH Wizard IP using the UltraScale FPGAs Transceiver Wizard found in the Vivado IP catalog.
2. Generate the SMPTE UHD-SDI from the Vivado IP catalog.
3. Instance the files `kugth_uhdsdi_wrapper_support`, `v_smpte_uhdsdi_wrapper`, and associated files.
4. Apply proper timing constraints the SDI interfaces.

Generating the GTH Wizard IP

Use the UltraScale FPGAs Transceiver Wizard to generate the GTH Wizard IP.

The GTH Wizard IP generated by the Wizard is actually a hierarchy of wrapper levels that optionally include the `GTH COMMON` instance and helper logic for GTH TX and RX clocking, GTH reset and data width sizing. For UHD-SDI applications, all the helper logic are suggested to be included with the GTH Wizard IP. The `GTH COMMON` must be excluded from the GTH Wizard IP since it's already being instantiated in the SDI Wrapper Support module. Each instance of GTH Wizard IP is LOC'ed to a specific `GTHE3_CHANNEL` location thus multiple GTH Wizard IP need to be generated for as many SDI channels in the design. Also, the SDI Wrapper Support module must be instantiated as many times as necessary, once for each GTH quad containing transceivers implementing SDI interfaces. If only the CPLL is used to clock the GTH transceivers, then the SDI Wrapper Support module doesn't need to be instantiated. An instance of `IBUFDS_GTE3` primitive must however be instantiated to bring the differential reference clock to the CPLL. The SDI demonstration applications supplied with this application note provide examples of how to instantiate the multiple GTH Wizard IP in the SDI Wrapper module.

The following information details exactly the steps required to generate the GTH wrapper using the Wizard from the Vivado IP catalog.

In the Vivado project, open the IP catalog. The UltraScale FPGAs Transceiver Wizard is found in the IO 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 on it to launch the Wizard.

Version 1.5 of the Wizard does not contain a protocol template for 6G-SDI and 12G-SDI. It however comes with HD-SDI and 3G-SDI presets, the 3G-SDI presets are to be used as a baseline moving forward. The instructions given in this section describe how to create a GTH wrapper with all the proper settings and ports necessary for implementing 6G-SDI and 12G-SDI interfaces. In the future, an SDI template will be added to the GTH wrapper.

The Wizard launches with the **Basic** tab open as shown in Figure 12. Above the tabs is a text field called **Component Name**. The name entered here is used as the name for the GTH wrapper file and the name of the GTH component. In this example, the component name is **v_smpte_uhdsdi_gtwiz_x0y16** ("_x0y16" designates the `GTHE3_CHANNEL` location).

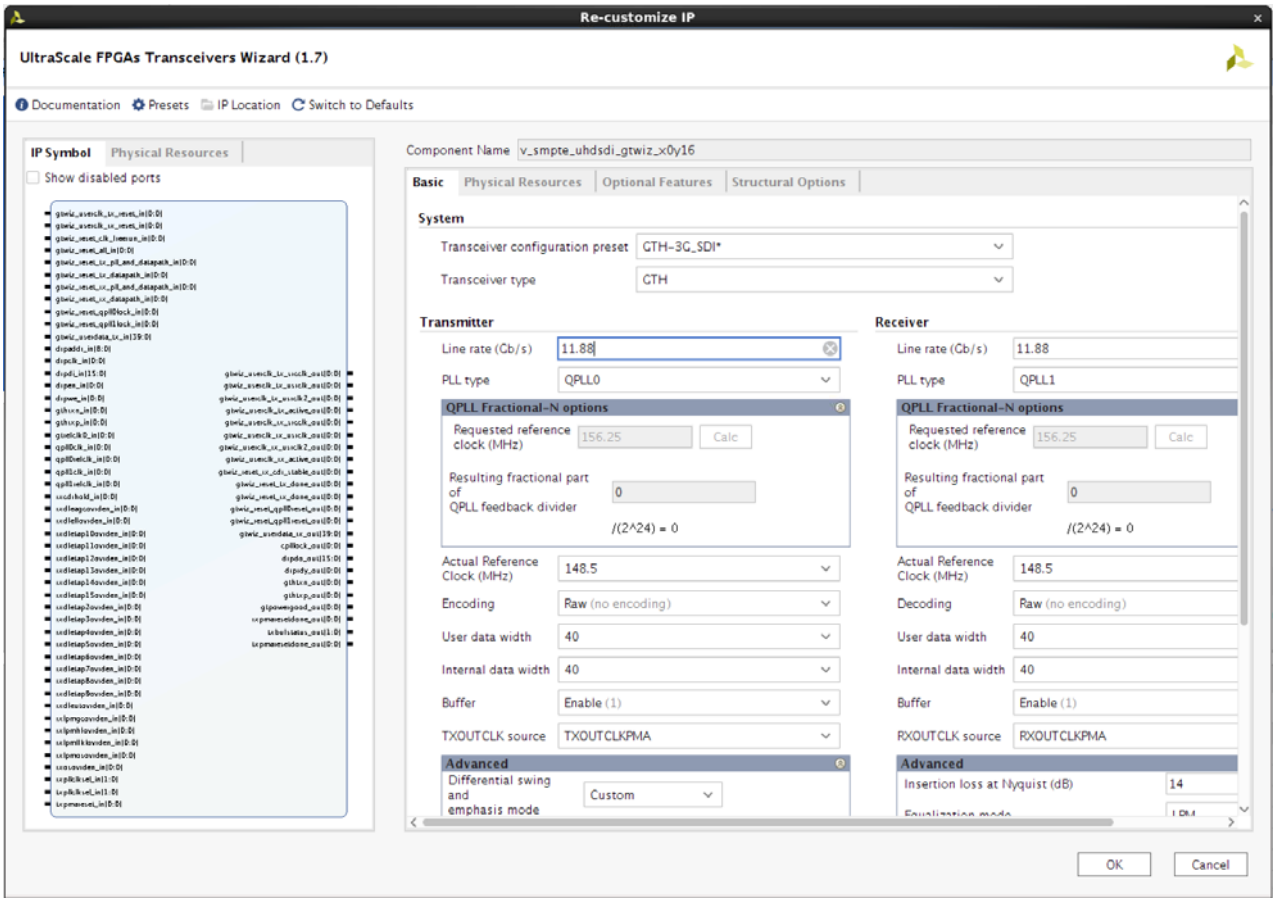


Figure 12: **UltraScale FPGAs Transceiver Wizard - Basic Tab**

Select the **GTH:3G-SDI** preset in from the list in the **Transceiver configuration preset** pull-down menu. This sets all basic settings in the Wizard for 3G-SDI operation. This preset is used as baseline to change the Wizard settings for 6G-SDI and 12G-SDI applications.

The **Line rate (Gb/s)** fields of Transmitter and Receiver, can be set to 5.94 for 6G-SDI and 11.88 for 12G-SDI applications. In this example the **PLL type** for the Transmitter is using QPLL0 as default clock source while Receiver is using QPLL1, the SDI controller module however dynamically switches between two clocks sources depending which SDI mode the TX and RX need to operate on. The CPLL can also be used as clock source for either TX and RX.

Set the **Reference clock (MHz)** frequency for both the TX and RX to the desired value, typically 148.5 MHz. Encoding should be set to Raw.

The **User data width** and **Internal data width** drop down menus are set according to the SDI interleaving pattern to be used. Typically, it is set to 40 for 6G-SDI and 12G-SDI applications. The SDI Controller dynamically change the GTH data width between 20 and 40 depending which SDI mode the TX and RX need to operate on. The User data width and Internal data width can also be set to 20 provided that the GTH is only supporting SD-SDI, HD-SDI and 3G-SDI.

Ensure that Buffer is enabled and that `TXOUTCLKPMA` and `RXOUTCLKPMA` are selected for `TXOUTCLK` source and `RXOUTCLK` source respectively.

In receiver **Advanced** menu, make sure that **Programmable termination voltage (mV)** is 800 and **Equalization mode** is LPM.

When moving from tab to tab, click on the tabs located under the **Component Name** field. Do not click the **OK** button until all tabs have been correctly setup. The **OK** button closes the Wizard.

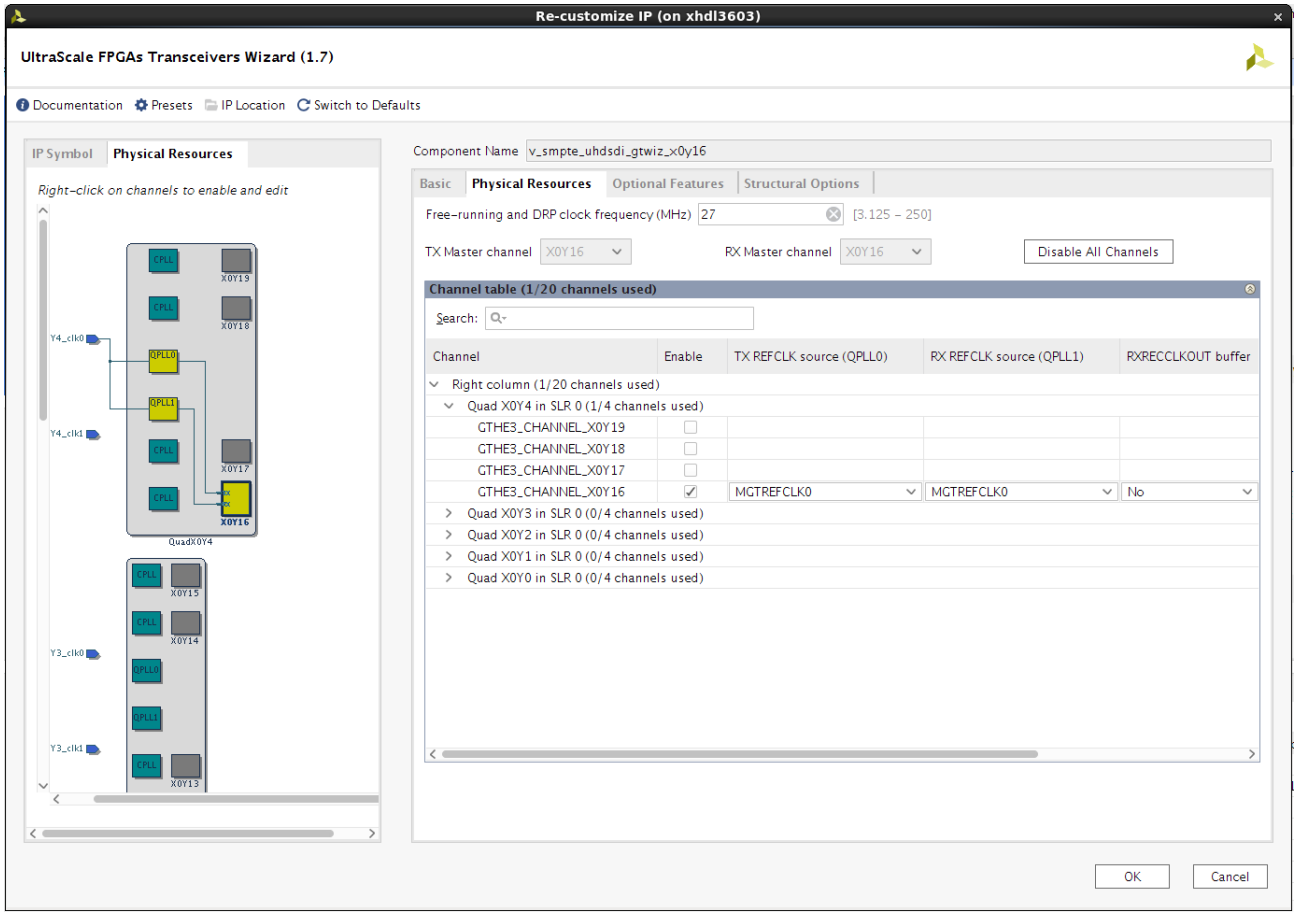


Figure 13: **UltraScale FPGAs Transceiver Wizard - Physical Resources Tab**

Go to the **Physical Resources** tab, shown in Figure 13. Set the desired **Free-running and DRP clock frequency (MHz)** field, in this example it is set to 27.

Select the target `GTHE3_CHANNEL` to be activated and make sure that only one `CHANNEL` is enabled per GTH Wizard IP instance. In this example, the RX unit uses the QPLL1 which uses `MGTREFCLK1` as its reference clock. The TX unit uses the QPLL0 referenced to `MGTREFCLK0`. The wizard does not explicitly handle the case where TX units are dynamically switched between the QPLL0 and the QPLL1. The SDI control module takes care of the control for this dynamic switching. But, to build a GTH wrapper with all the PLLs active and connected properly for dynamic switching of the TX between the QPLL0 and the QPLL1.

Optional Features tab shown in Figure 14, needs no modification, just ensure that both **Reset receiver elastic buffer on rate change** and **Reset Transmitter buffer on rate change** are enabled.

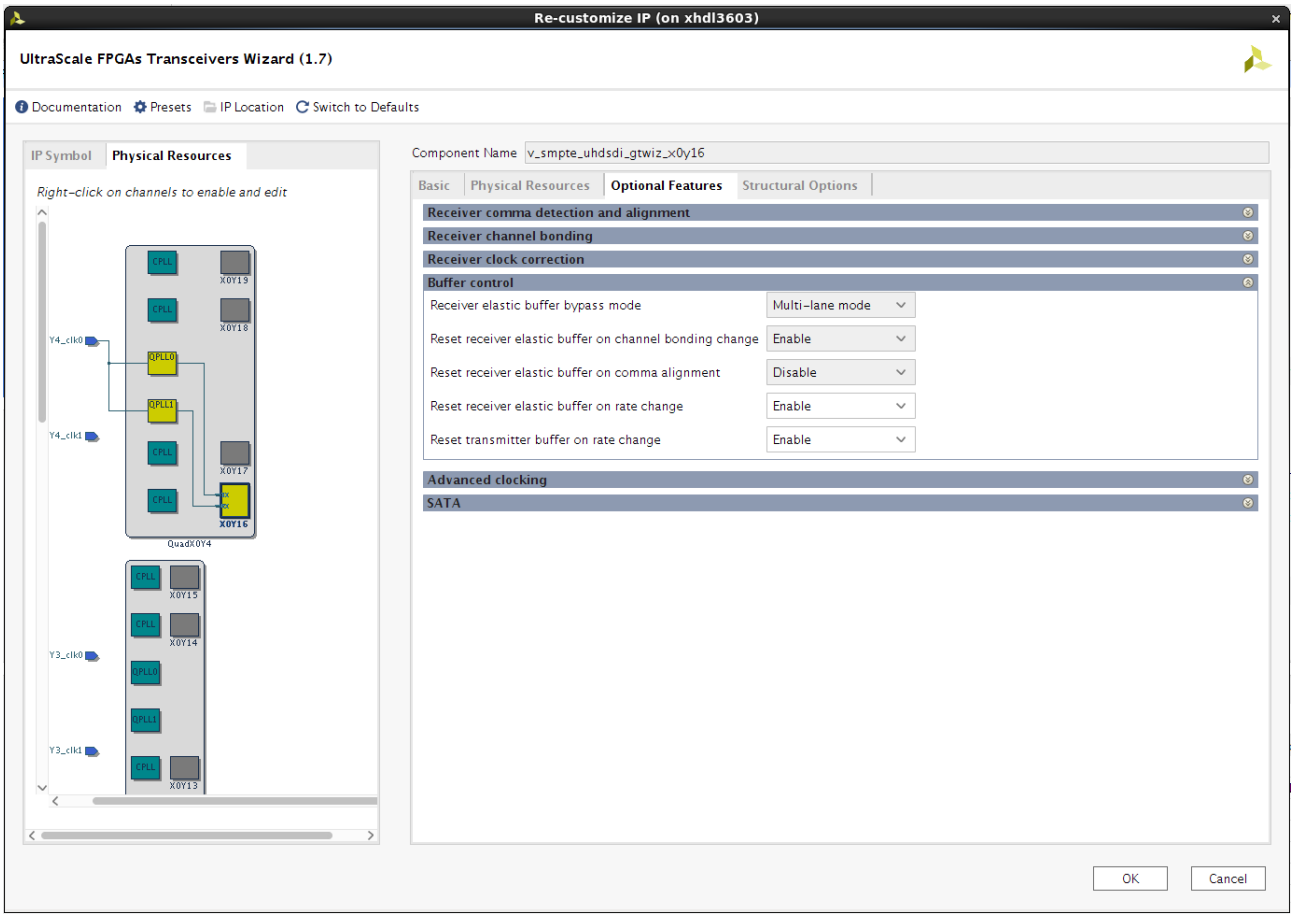


Figure 14: **UltraScale FPGAs Transceiver Wizard - Optional Features Tab**

Go to the **Structural Options** tab, shown in Figure 15.

In the **Simplify transceiver usage by organizing resources and helper blocks** section, set the **Include transceiver COMMON** in the field to **Example Design** and set the rest of the options to **Core**.

Expand the **All Ports** of the **Expose additional ports by functionality, for advanced feature usage** section. Enable the following ports in the **Inputs** sub-section

- drpaddr_in
- drpclk_in
- drpdi_in
- drpen_in
- drpwe_in
- gtrefclk0_in
- rxcdrhold_in
- rxdfeagcovrden_in
- rxdfelfovrden_in

- rxdfetap2ovrden_in
- rxdfetap3ovrden_in
- rxdfetap4ovrden_in
- rxdfetap5ovrden_in
- rxdfetap6ovrden_in
- rxdfetap7ovrden_in
- rxdfetap8ovrden_in
- rxdfetap9ovrden_in
- rxdfetap10ovrden_in
- rxdfetap11ovrden_in
- rxdfetap12ovrden_in
- rxdfetap13ovrden_in
- rxdfetap14ovrden_in
- rxdfetap15ovrden_in
- rxdfeutovrden_in
- rxlpmgcovrden_in
- rxlpmhfovrden_in
- rxlpmlfklovrden_in
- rxlpmosovrden_in
- rxosovrden_in
- rxpllclksel_in
- txpllclksel_in

Enable the following in **Outputs** sub-section

- cplllock_out
- drpdo_out
- drprdy_out

Some ports can also be enabled for debugging purposes such as `loopback_in`, `rxelecidlemode_in`, `txelecidlemode_in`, `txpostcursor_in`, and `txprecursor_in`.

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

The `rxelecidlemode_in` and `txelecidlemode_in` ports allow the TX and RX to be dynamically idled to save power.

The `txpostcursor_in` and `txprecursor_in` 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.

The GTH wrapper is generated by clicking the **OK** button and then the **Generate** button when the next menu opens.

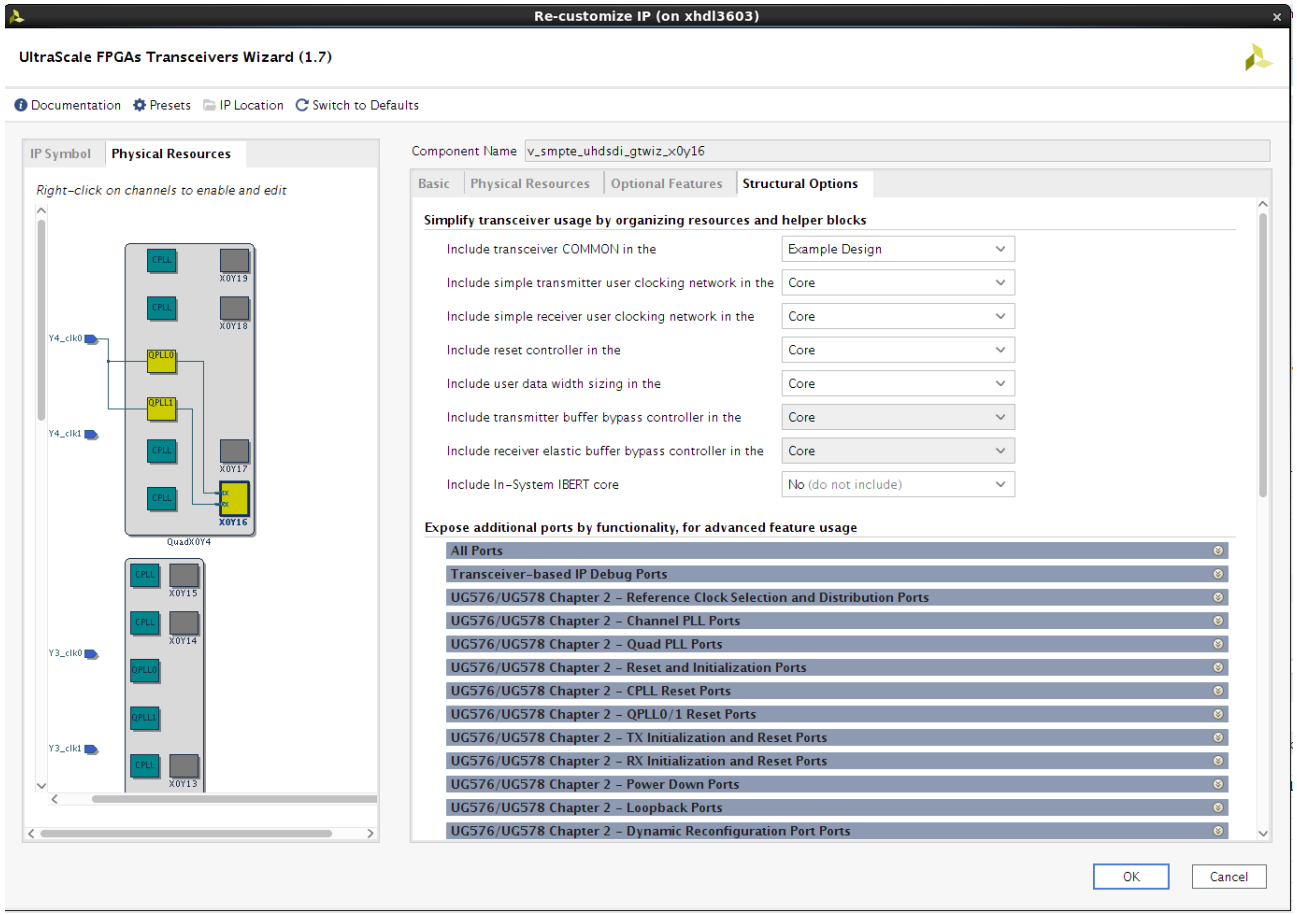


Figure 15: **UltraScale FPGAs Transceiver Wizard - Structural Options Tab**

Generate the SMPTE UHD-SDI IP Core

Use the Vivado IP catalog to generate the SMPTE UHD-SDI core. The SMPTE UHD-SDI core is located in the **Video & Image Processing** folder of the IP catalog.

The UHD-SDI core is a source code core, not a precompiled core. When the UHD-SDI core is generated, a folder is created containing the source code files for the UHD-SDI core in Verilog.

The options available when generating the UHD-SDI core is whether or not to include the error detection and handling (EDH) processor for the RX section and maximum line rate the core supports.

Choosing the Maximum Line Rate affects the maximum SDI data streams (DS) that is activated in the IP. Selecting **3G-SDI** activates 4 DS; **6G-SDI** and **12G-SDI 8DS** activates 8 DS; and **12G-SDI 16DS** activates 16 DS.

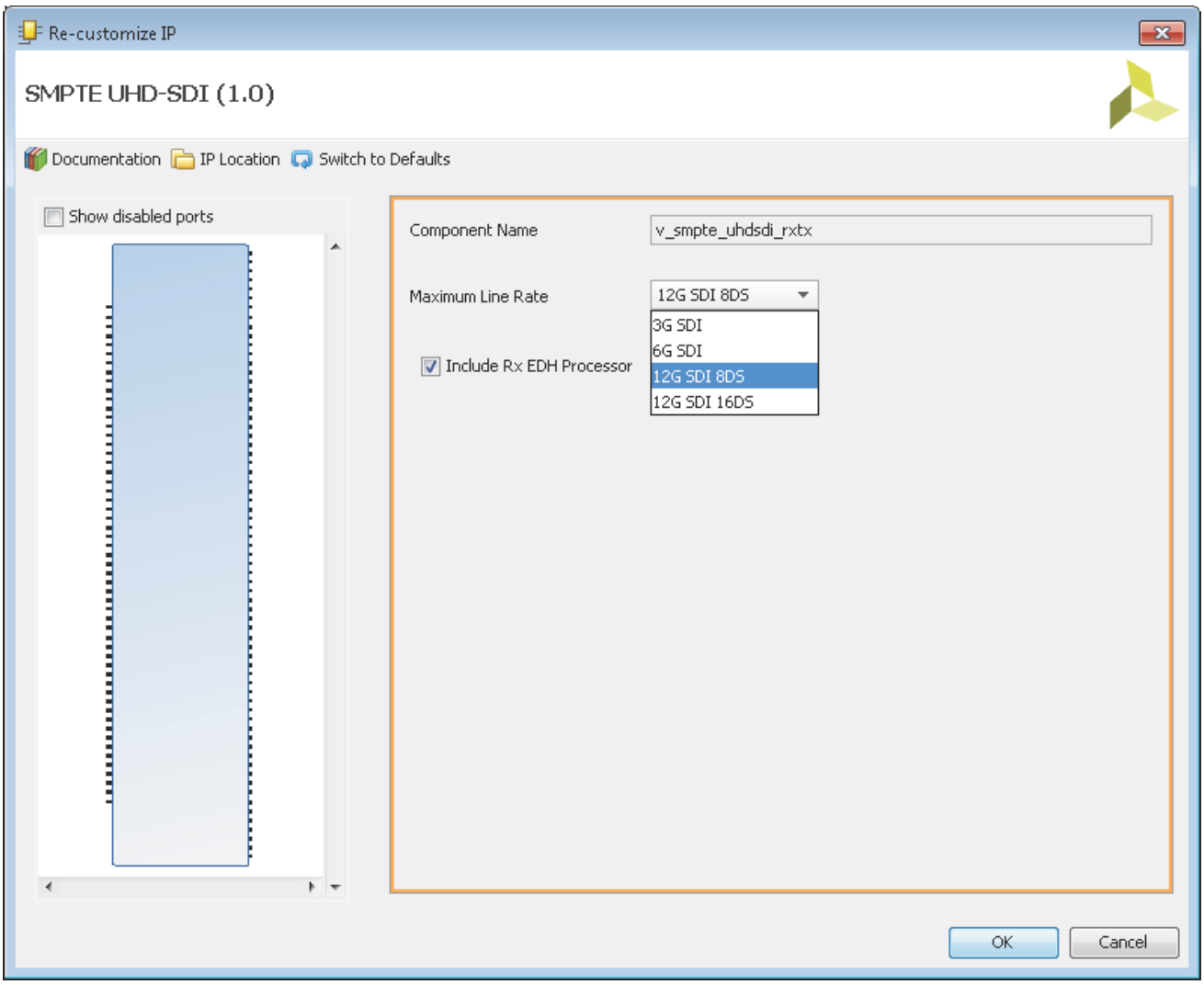


Figure 16: **SMPTE UHD-SDI IP Dialog**

Instantiating the UHD-SDI Wrappers

There are two main UHD-SDI wrappers in this application note, the SDI Wrapper Support (`kugth_uhdsdi_wrapper_support.v`) and SDI Wrapper (`kugth_uhdsdi_wrapper.v`), shown in Figure 5. The SDI Wrapper Support is needed when QPLL0 and/or QPLL1 is/are used by the SDI Wrapper and is instantiated once per quad only.

The SDI Wrapper Support and/or SDI Wrapper need to be instantiated and interconnected in your design. It is possible to implement the SDI interface without the SDI wrapper supplied with this application note, but the wrapper simplifies things because it interconnects the SMPTE UHD-SDI Core, Control Module, and one channel of GT Wizard IP. If the wrapper is not used, you must make all of these connections. There are alternative SDI wrappers file called `kugth_uhdsdi_<line rate>_norxedh_wrapper.v` that should be used when the UHD-SDI core is generated without the RX EDH processor.

There are 24 wrappers included in the reference design, instantiation and usage depend entirely upon the UHD-SDI core configuration. This reference design uses the wrappers in **bold**.

SDI 4-Channel Wrapper per quad:

- kugth_uhdsdi_12g_16s_4ch_wrapper.v
- kugth_uhdsdi_12g_16s_norxedh_4ch_wrapper.v
- **kugth_uhdsdi_12g_8s_4ch_wrapper.v**
- kugth_uhdsdi_12g_8s_norxedh_4ch_wrapper.v
- kugth_uhdsdi_6g_4ch_wrapper.v
- kugth_uhdsdi_6g_norxedh_4ch_wrapper.v
- kugth_uhdsdi_3g_4ch_wrapper.v
- kugth_uhdsdi_3g_norxedh_4ch_wrapper.v

SDI Wrapper Support

- kugth_uhdsdi_12g_16s_wrapper_support.v
- kugth_uhdsdi_12g_16s_norxedh_wrapper_support.v
- **kugth_uhdsdi_12g_8s_wrapper_support.v**
- kugth_uhdsdi_12g_8s_norxedh_wrapper_support.v
- kugth_uhdsdi_6g_wrapper_support.v
- kugth_uhdsdi_6g_norxedh_wrapper_support.v
- kugth_uhdsdi_3g_wrapper_support.v
- kugth_uhdsdi_3g_norxedh_wrapper_support.v

SDI Wrapper

- ugth_uhdsdi_12g_16s_wrapper.v
- kugth_uhdsdi_12g_16s_norxedh_wrapper.v
- **kugth_uhdsdi_12g_8s_wrapper.v**
- kugth_uhdsdi_12g_8s_norxedh_wrapper.v
- kugth_uhdsdi_6g_wrapper.v
- kugth_uhdsdi_6g_norxedh_wrapper.v
- kugth_uhdsdi_3g_wrapper.v
- kugth_uhdsdi_3g_norxedh_wrapper.v

In addition to the UHD-SDI core, the SDI Wrapper instantiates the following files:

- kugth_uhdsdi_control.v
- kugth_uhdsdi_drp_control.v
- kugth_uhdsdi_drp_control_fsm.v
- kugth_uhdsdi_rx_control.v
- kugth_uhdsdi_tx_control.v
- sync_block.v
- uhdsdi_rate_detect.v
- bs_flex_v_1.vhd
- nidru_20_v_6.vhd
- nidru_20_wrapper.vhd

IMPORTANT: *The SDI Wrapper contains an instance of the SMPTE UHD-SDI core. The SDI Wrapper must be edited so that the name given to the UHD-SDI core when it is generated is used where the core is instanced in the SDI wrapper. This can be avoided by using the component name v_smpte_uhdsdi_rxtx when generating the SMPTE UHD-SDI core.*

IMPORTANT: *The SDI Wrapper may contain multiple instances of GTH Wizard IP for multiple SDI channel design. Specific GTH Wizard IP is targeted by the use of XY_SITE generic in the SDI Wrapper and is used in the Verilog generate statement. The SDI Wrapper must be edited to handle each channel instance.*

Table 2 describes all of the ports of the SDI Wrapper. This port list is similar to the port list of the UHD-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 GTH 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 `txusrclk` or `rxusrclk`) 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_sd_ce_in` or `rx_ce_out`) is asserted. In 3G-SDI level B mode, a sample period is two clock cycles long as controlled by the assertion of `rx_ce_out` port.

Most of the RX and TX ports in this list are wired directly to the ports of the same name plus suffix of `_in` or `_out` on the UHD-SDI core that is instantiated inside the SDI wrapper. Timing diagrams of the video and video timing signals can be found in the *LogiCORE IP SMPTE UHD-SDI Product Guide LogiCORE IP* [Ref 17].

Table 2: **SDI Wrapper Port List**

Port Name	I/O	Width	Description
Receive Ports			
rx_fxdclk_in	in	1	Fixed frequency clock SDI RX bit rate detection
rx_rst_in	In	1	Synchronous reset input. This reset is synchronous to gth_drpclk_in port
rx_usrclk_out	out	1	GTH rxusrclk clock output. This port is also signal fed into rx_clk port of the UHD-SDI core
rx_mode_detect_rst_in	In	1	Synchronous reset that resets only the SDI mode detect search function. The SDI mode detect search function is only reset when rx_mode_detect_rst_in is High when rx_ce_out is High on a rising edge of rx_usrclk_out.
rx_mode_en_in	In	6	This port has unary bits to enable reception of each of the five SDI modes:
			Bit 0 enables HD-SDI mode
			Bit 1 enables SD-SDI mode
			Bit 2 enables 3G-SDI mode
			Bit 3 enables 6G-SDI mode
			Bit 4 enables 12G-SDI 11.88 Gb/s mode
			Bit 5 enables 12G-SDI 11.88/1.001 Gb/s mode
			When a bit is High, the corresponding SDI mode is enabled. When a bit is low, the receiver does not attempt to detect incoming SDI signals of that mode. Disabling unused SDI modes using these bits decrease the amount of time it takes for the receiver to lock to the incoming signal when it changes modes.
rx_mode_detect_en_in	In	1	This port enables the SDI mode detection feature when High. When enabled, the SDI mode detector controls the receiver to search for and lock to the incoming SDI data stream. When disabled, the user application must tell the SDI receiver what SDI mode to operate in using the rx_forced_mode_in port.
rx_forced_mode_in	In	3	When the rx_mode_detect_en_in input is Low, disabling the automatic SDI mode detection feature, the receiver operates in the SDI mode specified by the value on the rx_forced_mode_in port.
			000 = HD
			001 = SD
			010 = 3G
			100 = 6G
			101 = 12G 11.88 Gb/s
			110 = 12G 11.88/1.001 Gb/s

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

Port Name	I/O	Width	Description
rx_mode_out	Out	3	This output port indicates the current SDI mode of the receiver:
			000 = HD
			001 = SD
			010 = 3G
			100 = 6G
			101 = 12G 11.88 Gb/s
			110 = 12G 11.88/1.001 Gb/s
			When the receiver is not locked, the rx_mode port changes values as the receiver searches for the correct SDI mode. During this time, the rx_mode_locked output is Low. When the receiver detects the correct SDI mode, the rx_mode_locked output goes High.
rx_mode_hd_out	Out	1	High when RX is locked in HD-SDI mode
rx_mode_sd_out	Out	1	High when RX is locked in SD-SDI mode
rx_mode_3g_out	Out	1	High when RX is locked in 3G-SDI mode
rx_mode_6g_out	Out	1	High when RX is locked in 6G-SDI mode
rx_mode_12g_out	Out	1	High when RX is locked in 12G-SDI mode (either bit rate)
rx_mode_locked_out	Out	1	When this output is Low, the receiver is actively searching for the SDI mode that matches the input data stream. During this time, the rx_mode_out port changes frequently. When the receiver locks to the correct SDI mode, the rx_mode_locked_out output goes High.
			When the SDI mode detect function is disabled (rx_mode_detect_en_in = Low), this output is always asserted High. In this case, it is not a reliable indicator of whether or not the SDI receiver is locked to the incoming SDI signal.
rx_bit_rate_out	Out	1	This is the bit rate output of the v_smpte_uhdsdi_rate_detect.v module. This port is the signal that goes into the rx_bit_rate port of the UHD-SDI core.
			HD-SDI mode:
			rx_m_out = 0: Bit rate = 1.485 Gb/s
			rx_m_out = 1: Bit rate = 1.485/1.001 Gb/s
			3G-SDI mode:
			rx_m_out = 0: Bit rate = 2.97 Gb/s
			rx_m_out = 1: Bit rate = 2.97/1.001 Gb/s
			6G-SDI mode:
			rx_m_out = 0: Bit rate = 5.94 Gb/s
			rx_m_out = 1: Bit rate = 5.94/1.001 Gb/s
			12G-SDI mode:
			rx_m_out = 0: Bit rate = 11.88 Gb/s
			rx_m_out = 1: Bit rate = 11.88/1.001 Gb/s

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

Port Name	I/O	Width	Description
rx_t_locked_out	Out	1	This output is High when the transport detection function in the receiver has identified the transport format of the SDI signal.
rx_t_family_out	Out	4	This output indicates which family of video signals is being used as the transport of 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 rransported. It only identifies the transport characteristics. See Table 3 for the encoding of this port.
rx_t_rate_out	Out	4	This output indicates the frame rate of the transport. This is not necessarily the same as the frame rate of the actual picture. This output is only valid when rx_t_locked is High. See Table 4 for the encoding of this port.
rx_t_scan_out	Out	1	This output indicates whether the transport 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	Out	1	In 3G-SDI mode, this output is asserted High when the input signal is level B and Low when it is level A. This output is only valid when rx_mode_3g is High.
rx_ce_out	Out	NUM_RX_CE	This is the RX clock enable output. There are NUM_RX_CE copies of this clock enable on this port. These clock enables are valid in all SDI modes. In SD mode, the CEs have a nominal 5|6|5|6 cadence. In HD and 3GA modes, the CEs are always High. In 3GB mode, the CEs have a 50% duty cycle. In 6G, the duty cycle can by 100% or 50% depending on how may data streams are interleaved onto the signal. In 12G, the duty cycle can be 50% or 25% depending on how many data streams are interleaved onto the signal. This port replaces the rx_ce_sd and rx_dout_rdy_3g ports of the old core and combines their functionality by being correct for all SDI modes.
rx_active_streams_out	Out	3	This port indicates the number of data streams that are active for the current video format being received. The number of active data streams is 2^active_streams. 000: 1 active stream 001: 2 active streams 010: 4 active streams 011: 8 active streams 100: 16 active streams
rx_line_0_out	Out	11	Captured line number from data stream 1 is output here. Not valid in SD-SDI mode.
rx_line_1_out	Out	11	Captured line number from data stream 3 is output here. Only valid if 4 or more data streams are active.
rx_line_2_out	Out	11	Captured line number from data stream 5 is output here. Only valid if 8 or more data streams are active.
rx_line_3_out	Out	11	Captured line number from data stream 7 is output here. Only valid if 8 or more data streams are active.
rx_line_4_out	Out	11	Captured line number from data stream 9 is output here. Only valid if 16 data streams are active.

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

Port Name	I/O	Width	Description
rx_line_5_out	Out	11	Captured line number from data stream 11 is output here. Only valid if 16 data streams are active.
rx_line_6_out	Out	11	Captured line number from data stream 13 is output here. Only valid if 16 data streams are active.
rx_line_7_out	Out	11	Captured line number from data stream 15 is output here. Only valid if 16 data streams are active.
rx_st352_0_out	Out	32	The ST 352 payload ID packet data bytes captured from ds1 are output here.
rx_st352_0_valid_out	Out	1	High when rx_st352_0 is valid.
rx_st352_1_out	Out	32	The ST 352 payload ID packet data bytes captured from ds3 are output here. In 3G-SDI level A mode, the ST 352 payload ID packet data bytes from ds2 are output here.
rx_st352_1_valid_out	Out	1	High when rx_st352_1 is valid.
rx_st352_2_out	Out	32	The ST 352 payload ID packet data bytes captured from ds5 are output here.
rx_st352_2_valid_out	Out	1	High when rx_st352_2 is valid.
rx_st352_3_out	Out	32	The ST 352 payload ID packet data bytes captured from ds7 are output here.
rx_st352_3_valid_out	Out	1	High when rx_st352_3 is valid.
rx_st352_4_out	Out	32	The ST 352 payload ID packet data bytes captured from ds9 are output here.
rx_st352_4_valid_out	Out	1	High when rx_st352_4 is valid.
rx_st352_5_out	Out	32	The ST 352 payload ID packet data bytes captured from ds11 are output here.
rx_st352_5_valid_out	Out	1	High when rx_st352_5 is valid.
rx_st352_6_out	Out	32	The ST 352 payload ID packet data bytes captured from ds13 are output here.
rx_st352_6_valid_out	Out	1	High when rx_st352_6 is valid.
rx_st352_7_out	Out	32	The ST 352 payload ID packet data bytes captured from ds15 are output here.
rx_st352_7_valid_out	Out	1	High when rx_st352_7 is valid.
rx_crc_err_out	Out	16	These 16 bits are the CRC error indicator for each data stream output. Bit 0 is the CRC error indicator for data stream 1, bit 1 for data stream 2, etc. When a CRC is detected on a line, the CRC error bit corresponding to that data stream becomes asserted starting a few clock cycles after the last CRC word is output on the data stream ports following the EAV that ends the line containing the error. The CRC error bit remains asserted for one line time. These bits are not valid in SD-SDI mode.
rx_ds1_out	Out	10	Data stream 1 output. In SD mode this is interleaved Y/C. In HD and 3GA modes, this is the Y channel. In 3GB mode, this is the link A Y channel. In 6G and 12G modes, this is ds1. Same as the rx_ds1a output port of previous core.

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

Port Name	I/O	Width	Description
rx_ds2_out	Out	10	Data stream 2 output. Not used in SD mode. In HD and 3GA modes, this is the C channel. In 3GB mode, this is the link A C channel. In 6G and 12G modes, this is ds2. Same as the rx_ds2a port of previous core.
rx_ds3_out	Out	10	Data stream 3 output. Not used in SD, HD, and 3GA modes. In 3GB mode, this is the link B Y channel. In 6G and 12G modes this is ds3. Same as the rx_ds1b port of previous core.
rx_ds4_out	Out	10	Data stream 4 output. Not used in SD, HD, and 3G level A modes. In 3G level B mode, this is the link B C channel. In 6G and 12G modes this is ds4.
rx_ds5_out	Out	10	Data stream 5 output. Only used in 6G and 12G modes.
rx_ds6_out	Out	10	Data stream 6 output. Only used in 6G and 12G modes.
rx_ds7_out	Out	10	Data stream 7 output. Only used in 6G and 12G modes.
rx_ds8_out	Out	10	Data stream 8 output. Only used in 6G and 12G modes.
rx_ds9_out	Out	10	Data stream 9 output. Only used in 12G mode when 16 data streams are active.
rx_ds10_out	Out	10	Data stream 10 output. Only used in 12G mode when 16 data streams are active.
rx_ds11_out	Out	10	Data stream 11 output. Only used in 12G mode when 16 data streams are active.
rx_ds12_out	Out	10	Data stream 12 output. Only used in 12G mode when 16 data streams are active.
rx_ds13_out	Out	10	Data stream 13 output. Only used in 12G mode when 16 data streams are active.
rx_ds14_out	Out	10	Data stream 14 output. Only used in 12G mode when 16 data streams are active.
rx_ds15_out	Out	10	Data stream 15 output. Only used in 12G mode when 16 data streams are active.
rx_ds16_out	Out	10	Data stream 16 output. Only used in 12G mode when 16 data streams are active.
rx_eav_out	Out	1	This output is asserted High when the XYZ word of an EAV is present on the data stream output ports.
rx_sav_out	Out	1	This output is asserted High when the XYZ word of a SAV is present on the data stream output ports.
rx_trs_out	Out	1	This output is asserted High while the four consecutive words of any EAV or SAV are present on the data stream output ports, starting
rx_edh_errcnt_en_in	In	16	This input controls which EDH error conditions increments the rx_edh_errcnt counter. See Table 5 for more details. (1)
rx_edh_clr_errcnt_in	In	1	When High, this input clears the rx_edh_errcnt counter. This input port must be High during the same clock cycle when rx_ce_sd is also High to clear the error counter.(1)
rx_edh_ap_out	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.(1)

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

Port Name	I/O	Width	Description
rx_edh_ff_out	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.(1)
rx_edh_anc_out	Out	1	This output is asserted High when an ancillary data packet checksum error is detected.(1)
rx_edh_ap_flags_out	Out	5	The active picture error flag bits from the most recently received EDH packet are output on this port. See Table 4 for encoding of this port. See Table 6 for more details. (1)
rx_edh_ff_flags_out	Out	5	The full frame error flag bits from the most recently received EDH packet are output on this port. See Table 4 for encoding of this port. See Table 6 for more details. (1)
rx_edh_anc_flags_out	Out	5	The ancillary error flag bits from the most recently received EDH packet are output on this port. See Table 4 for encoding of this port. See Table 6 for more details. (1)
rx_edh_packet_flags_out	Out	4	This port outputs four error flags related to the most recently received EDH packet. See Table 5 for encoding of this port. See Table 7 for more details. (1)
rx_edh_errcnt_out	Out	16	This is the SD-SDI EDH error counter. It increments once per field when any of the error conditions enabled by the rx_edh_err_en port occur during that field. (1)
rx_change_done_out	Out	1	This output is Low during those periods when the GTH 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_out output is asserted High to indicate successful completion. This output is synchronous with the gth_drpclk_in.
rx_change_fail_out	Out	1	Under normal conditions, this output is always Low. It only goes High if the control module is unsuccessful in completing a GTH RX initialization, reset, or SDI mode change sequence. If such a failure occurs, the rx_change_fail_out port is asserted High and the rx_change_fail_code_out port indicates the nature of the failure. If a failure occurs, the GTH RX must be reset using the rx_rst_in and gth_wiz_reset_rx_pll_and_datapath_in. This output is synchronous with the gth_drpclk.
rx_change_fail_code_out	Out	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 gth_drpclk_in.
Transmit Ports			
tx_rst_in	In	1	This is a synchronous reset input. It resets the transmitter when High. To fully reset the transmitter, the tx_ce_in, tx_sd_ce_in, and tx_edh_ce_in inputs must be High when tx_rst_in is asserted.
tx_usrclk_out	out	1	GTH txusrclk clock output. This port is also signal fed into tx_clk port of the UHD-SDI core

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

Port Name	I/O	Width	Description
tx_ce_in	In	1	This is the clock enable input for the main portion of the transmitter data path. This is somewhat equivalent to the tx_din_rdy port of the old core. It must be High in SD, HD, and 3GA modes. In 3GB mode, it must have a 50% duty cycle. In 6G and 12G modes, it must have a 100% duty cycle when 4 streams are interleaved, 50% duty cycle when 8 streams are interleaved, and 25% duty cycle when all 16 data streams are interleaved.
tx_sd_ce_in	In	1	This is the clock enable for SD-SDI mode. It must have exactly a 5|6|5|6 cadence in SD-SDI mode and must be High in all other modes.
tx_edh_ce_in	In	1	This is the clock enable for the TX EDH processor. In SD-SDI mode, it must be exactly equal to the tx_sd_ce port with its 5|6|5|6 cadence. It must be phase aligned with tx_sd_ce_in. In all other modes, this ce can be driven Low to reduce the power that would be consumed by the EDH processor.
tx_mode_in	In	3	This input port is used to select the transmitter SDI mode: 000 = HD 001 = SD 010 = 3G 100 = 6G 101 = 12G All other values are reserved.
tx_m_in	In	1	This port is used to select which reference clock to use. By convention, 0 = select 148.35 MHz refclk, 1 = select 148.5 MHz refclk. However, this distinction is entire governed by the frequency of the PLLs and the values set to TXPLLCLKSEL_TX_M_0 and TXPLLCLKSEL_TX_M_1 parameters in Table 2.
tx_insert_crc_in	In	1	When this input is High, the transmitter generates and inserts CRC values into the data streams for each video line in all modes except SD-SDI. When this input is Low, CRC values are not inserted into the data streams. This input is ignored in SD-SDI mode.
tx_insert_ln_in	In	1	When this input is High, the transmitter inserts line numbers into all active data streams after the EAV of each video line. The line numbers must be supplied on the tx_line_chX_in input ports of all active data stream pairs. When this input is Low, line numbers are not inserted. This input is ignored in SD-SDI mode.
tx_insert_st352_in	In	1	When this input is High, ST 352 packets are inserted into the data streams, otherwise the packets are not inserted. ST 352 packets are mandatory in 3G, 6G, and 12G modes and optional in HD and SD modes.
tx_overwrite_st352_in	In	1	If this input is High, ST 352 packets already present in the data streams are overwritten. If this input is Low, existing ST 352 packets are not overwritten.
tx_insert_edh_in	In	1	When this input is High, the transmitter generates and inserts EDH packets into every field in SD-SDI mode. When this input is Low, EDH packets are not inserted. This input is ignored in all modes except SD-SDI mode.

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

Port Name	I/O	Width	Description
tx_mux_pattern_in	In	3	This specifies the data stream interleaving pattern to be used. 000 = SD, HD, and 3G level A 001 = 3G level B 010 = 8 stream interleave in 6G and 12G modes 011 = 4 stream interleave in 6G mode 100 = 16 stream interleave in 12G mode
tx_insert_sync_bit_in	In	1	In 6G and 12G modes, when this port is High, the sync bit insertion function is enabled for run length mitigation. For compliance with the ST 2081 and ST 2082 standards, sync bit insertion must be enabled. However, some early implementations of 6G-SDI and 12G-SDI receivers do not support sync bit insertion and when transmitting signals to those devices, sync bit insertion can be disabled when setting this port Low.
tx_line_0_in	In	11	Current line number for data streams 1 & 2
tx_line_1_in	In	11	Current line number for data streams 3 & 4
tx_line_2_in	In	11	Current line number for data streams 5 & 6
tx_line_3_in	In	11	Current line number for data streams 7 & 8
tx_line_4_in	In	11	Current line number for data streams 9 & 10
tx_line_5_in	In	11	Current line number for data streams 11 & 12
tx_line_6_in	In	11	Current line number for data streams 13 & 14
tx_line_7_in	In	11	Current line number for data streams 15 & 16
tx_st352_line_f1_in	In	11	The ST 352 packets are inserted into 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 packets are inserted. The input value must be valid during the entire HANC interval. If tx_insert_st352 is Low, this input is ignored.
tx_st352_line_f2_in	In	11	For interlace video, ST 352 packets are inserted on the line number in field 2 indicated by this value. For progressive video, this input port must be disabled by driving the tx_st352_f2_en port Low. The input value on this port must be valid during the entire HANC interval. This port is ignored if either tx_insert_st352 or tx_st352_f2_en are Low.
tx_st352_f2_en_in	In	1	This input controls whether ot not ST 352 packets are inserted on the line indicated by tx_vpid_line_f2. For interlaced video, this input must be High if ST 352 packet insertion is enabled. For progressive video, this input must be Low if ST 352 packet insertion is enabled. If ST 352 packet insertion is disabled (tx_insert_st352 = Low), this port is ignored.
tx_st352_data_0_in	In	32	The four data bytes of the ST352 packet to be inserted into ds1 when tx_insert_st352 is High. The data bytes are ordered like this: {byte4, byte3, byte2, byte1}.
tx_st352_data_1_in	In	32	The four data bytes of the ST352 packet to be inserted into ds3 when tx_insert_st352 is High. In 3GA mode, this port specifies the data bytes that is inserted into the ST352 packet of ds2.
tx_st352_data_2_in	In	32	The four data bytes of the ST352 packet to be inserted into ds5 when tx_insert_st352 is High.

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

Port Name	I/O	Width	Description
tx_st352_data_3_in	In	32	The four data bytes of the ST352 packet to be inserted into ds7 when tx_insert_st352 is High.
tx_st352_data_4_in	In	32	The four data bytes of the ST352 packet to be inserted into ds9 when tx_insert_st352 is High.
tx_st352_data_5_in	In	32	The four data bytes of the ST352 packet to be inserted into ds11 when tx_insert_st352 is High.
tx_st352_data_6_in	In	32	The four data bytes of the ST352 packet to be inserted into ds13 when tx_insert_st352 is High.
tx_st352_data_7_in	In	32	The four data bytes of the ST352 packet to be inserted into ds15 when tx_insert_st352 is High.
tx_ds1_in	In	10	Data stream 1 input: SD=Y/C, HD=Y, 3GA=DS1(Y), 3GB=AY, 6G/12G=DS1
tx_ds2_in	In	10	Data stream 2 input: HD=C, 3GA=DS2(C), 3GB=AC, 6G/12G=DS2
tx_ds3_in	In	10	Data stream 3 input: 3GB=BY, 6G/12G=DS3
tx_ds4_in	In	10	Data stream 4 input: 3GB=BC, 6G/12G=DS4
tx_ds5_in	In	10	Data stream 5 input: 6G/12G=DS5
tx_ds6_in	In	10	Data stream 6 input: 6G/12G=DS6
tx_ds7_in	In	10	Data stream 7 input: 6G/12G=DS7
tx_ds8_in	In	10	Data stream 8 input: 6G/12G=DS8
tx_ds9_in	In	10	Data stream 9 input: 12G=DS9
tx_ds10_in	In	10	Data stream 10 input: 12G=DS10
tx_ds11_in	In	10	Data stream 11 input: 12G=DS11
tx_ds12_in	In	10	Data stream 12 input: 12G=DS12
tx_ds13_in	In	10	Data stream 13 input: 12G=DS13
tx_ds14_in	In	10	Data stream 14 input: 12G=DS14
tx_ds15_in	In	10	Data stream 15 input: 12G=DS15
tx_ds16_in	In	10	Data stream 16 input: 12G=DS16
tx_ds1_st352_out	Out	10	This is the data stream 1 (DS1) output data stream after the ST 352 packet insertion module. The data stream is output at this point for the application to embed other ANC data.
tx_ds2_st352_out	Out	10	This is the DS2 output data stream for ANC insertion.
tx_ds3_st352_out	Out	10	This is the DS3 output data stream for ANC insertion.
tx_ds4_st352_out	Out	10	This is the DS4 output data stream for ANC insertion.
tx_ds5_st352_out	Out	10	This is the DS5 output data stream for ANC insertion.
tx_ds6_st352_out	Out	10	This is the DS6 output data stream for ANC insertion.
tx_ds7_st352_out	Out	10	This is the DS7 output data stream for ANC insertion.
tx_ds8_st352_out	Out	10	This is the DS8 output data stream for ANC insertion.
tx_ds9_st352_out	Out	10	This is the DS9 output data stream for ANC insertion.
tx_ds10_st352_out	Out	10	This is the DS10 output data stream for ANC insertion.

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

Port Name	I/O	Width	Description
tx_ds11_st352_out	Out	10	This is the DS11 output data stream for ANC insertion.
tx_ds12_st352_out	Out	10	This is the DS12 output data stream for ANC insertion.
tx_ds13_st352_out	Out	10	This is the DS13 output data stream for ANC insertion.
tx_ds14_st352_out	Out	10	This is the DS14 output data stream for ANC insertion.
tx_ds15_st352_out	Out	10	This is the DS15 output data stream for ANC insertion.
tx_ds16_st352_out	Out	10	This is the DS16 output data stream for ANC insertion.
tx_ds1_anc_in	In	10	Data stream 1 (DS1) input from the application's ANC inserter. This port is only used if the tx_use_anc_in port is High.
tx_ds2_anc_in	In	10	Data stream 2 (DS2) input from the application's ANC inserter. This port is only used if the tx_use_anc_in port is High.
tx_ds3_anc_in	In	10	Data stream 3 (DS3) input from the application's ANC inserter. This port is only used if the tx_use_anc_in port is High.
tx_ds4_anc_in	In	10	Data stream 4 (DS4) input from the application's ANC inserter. This port is only used if the tx_use_anc_in port is High.
tx_ds5_anc_in	In	10	Data stream 5 (DS5) input from the application's ANC inserter. This port is only used if the tx_use_anc_in port is High.
tx_ds6_anc_in	In	10	Data stream 6 (DS6) input from the application's ANC inserter. This port is only used if the tx_use_anc_in port is High.
tx_ds7_anc_in	In	10	Data stream 7 (DS7) input from the application's ANC inserter. This port is only used if the tx_use_anc_in port is High.
tx_ds8_anc_in	In	10	Data stream 8 (DS8) input from the application's ANC inserter. This port is only used if the tx_use_anc_in port is High.
tx_ds9_anc_in	In	10	Data stream 9 (DS9) input from the application's ANC inserter. This port is only used if the tx_use_anc_in port is High.
tx_ds10_anc_in	In	10	Data stream 10 (DS10) input from the application's ANC inserter. This port is only used if the tx_use_anc_in port is High.
tx_ds11_anc_in	In	10	Data stream 11 (DS11) input from the application's ANC inserter. This port is only used if the tx_use_anc_in port is High.
tx_ds12_anc_in	In	10	Data stream 12 (DS12) input from the application's ANC inserter. This port is only used if the tx_use_anc_in port is High.
tx_ds13_anc_in	In	10	Data stream 13 (DS13) input from the application's ANC inserter. This port is only used if the tx_use_anc_in port is High.
tx_ds14_anc_in	In	10	Data stream 14 (DS14) input from the application's ANC inserter. This port is only used if the tx_use_anc_in port is High.
tx_ds15_anc_in	In	10	Data stream 15 (DS15) input from the application's ANC inserter. This port is only used if the tx_use_anc_in port is High.
tx_ds16_anc_in	In	10	Data stream 16 (DS16) input from the application's ANC inserter. This port is only used if the tx_use_anc_in port is High.
tx_use_anc_in	In	1	When Low, the data streams out of the ST352 packet insertion function are routed internally to the TX output channels. When High, the TX output channels accept data streams from the tx_ds[16:1]_anc_in ports.

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

Port Name	I/O	Width	Description
tx_ce_align_err_out	Out	1	This output indicates problems with the 5|6|5|6 clock cycle cadence of the tx_sd_ce input in SD-SDI mode. In SD-SDI mode, the tx_sd_ce signal must follow a regular 5|6|5|6 clock cycle cadence. If it does not, the SD-SDI serial stream is formed incorrectly. The tx_ce_align_err output goes High if the cadence is incorrect. This port is only valid in SD-SDI mode and only if tx_sd_bitrep_bypass is Low.
tx_slew_out	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	Out	1	This output is Low during those periods when the GTH TX is being initialized or reset or the GTH DRP registers or txsysclksel ports are being dynamically changed. If the sequence completes successfully, the tx_change_done_out output is asserted High to indicate successful completion. This output is synchronous with the gth_drpclk_in.
tx_change_fail_out	Out	1	Under normal conditions, this output is always Low. It only goes High if the control module is unsuccessful in completing a GTH TX initialization or reset sequence or a dynamic change of the GTH DRP or txsysclksel ports. If such a failure occurs, the tx_change_fail_out port is asserted High and the tx_change_fail_code port indicates the nature of the failure. If a failure occurs as indicated by tx_change_fail_out going High, a full reset must be done using the tx_rst_in and gth_wiz_reset_tx_pll_and_datapath_in. This output is synchronous with the gth_drpclk_in.
tx_change_fail_code_out	Out	3	When the tx_change_fail port is High, this port indicates the nature of the failure. See Table 9 for encoding of this port. This output is synchronous with the gth_drpclk_in.
DRP Controller Ports			
drp_fail_out	Out	1	Under normal conditions, this output is always Low. It only goes High if the control module is unsuccessful in completing a GTH DRP transaction. If such a failure occurs, the drp_fail_out port is asserted High and thedrp_fail_cnt_out port incremented. If a failure occurs as indicated by drp_fail_out going High, a full GTH reset must be done using the gth_wiz_reset_all_in port. This output is synchronous with the gth_drpclk_in.
drp_fail_cnt_out	Out	8	This port indicates the count of how many DRP transaction attempts have failed.
GTH Ports for SDI Wrapper Support			
gth_wiz_reset_all_in	In	1	User signal to reset the phase-locked loops (PLLs) and active data directions of transceiver primitives. An active-High, asynchronous pulse of at least one gth_drpclk_in period in duration initializes the process.
gth_wiz_reset_tx_pll_and_datapath_in	In	1	User signal to reset the transmit data direction and associated PLLs of transceiver primitives. An active-High, asynchronous pulse of at least one gth_drpclk_in period in duration initializes the process.

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

Port Name	I/O	Width	Description
gth_wiz_reset_rx_pll_and_datapath_in	In	1	User signal to reset the receive data direction and associated PLLs of transceiver primitives. An active-High, asynchronous pulse of at least one gth_drpclk_in period in duration initializes the process.
gth_wiz_txresetdone_out	Out	1	Active-High indication that the transmitter reset sequence of transceiver primitives has completed. This output is synchronous with the tx_usrclk_out.
gth_wiz_rxresetdone_out	Out	1	Active-High indication that the receiver reset sequence of transceiver primitives has completed. This output is synchronous with the rx_usrclk_out.
gth_drpclk_in	In	1	DRP clock to GTH. Normally, this port is driven by same clock as rx_fxdclk_in.
gth_qpll0_refclk_p_in	In	1	This port must be connected to either MGTREFCLK0P or MGTREFCLK1P FPGA input ports.This port drives the I pin of the IBUFDS_GTE3 primitive.
gth_qpll0_refclk_n_in	In	1	This port must be connected to either MGTREFCLK0N or MGTREFCLK1N FPGA input ports.This port drives the IB pin of the IBUFDS_GTE3 primitive.
gth_qpll0_reset_in	In	1	Active-High reset input to the QPLL0RESET pin of GTHE3_COMMON primitive
gth_qpll0_clk_out	Out	1	This should be connected to gth_qpll0_clk_in port of SDI Wrapper. Clock output from the QPLL0OUTCLK port of GTHE3_COMMON primitive.
gth_qpll0_refclk_out	Out	1	This should be connected to gth_qpll0_refclk_in port of SDI Wrappert. Clock output from the QLL0OUTREFCLK port of GTHE3_COMMON primitive.
gth_qpll0_lock_out	Out	1	This should be connected to gth_qpll0_lock_in port of SDI Wrapper. This active-High lock indicator of QPLL0 from the QPLL0LOCK port of GTHE3_COMMON
gth_qpll1_refclk_p_in	In	1	This port must be connected to either MGTREFCLK0P or MGTREFCLK1P FPGA input ports.This port drives the I pin of the IBUFDS_GTE3 primitive.
gth_qpll1_refclk_n_in	In	1	This port must be connected to either MGTREFCLK0N or MGTREFCLK1N FPGA input ports.This port drives the IB pin of the IBUFDS_GTE3 primitive.
gth_qpll1_reset_in	In	1	Active-High reset input to the QPLL1RESET pin of GTHE3_COMMON primitive
gth_qpll1_clk_out	Out	1	This should be connected to gth_qpll1_clk_in port of SDI Wrapper. Clock output from the QPLL1OUTCLK port of GTHE3_COMMON primitive.
gth_qpll1_refclk_out	Out	1	This should be connected to gth_qpll1_refclk_in port of SDI Wrappert. Clock output from the QLL1OUTREFCLK port of GTHE3_COMMON primitive.
gth_qpll1_lock_out	Out	1	This should be connected to gth_qpll1_lock_in port of SDI Wrapper. This active-High lock indicator of QPLL1 from the QPLL1LOCK port of GTHE3_COMMON

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

Port Name	I/O	Width	Description
gth_cpll_refclk_out	Out	1	This port is meant to be connected to the gth_cpll_refclk_in port of SDI Wrapper. Clock output from from a IBUFDS_GTE3 primitive.
gth_cpll_lock_out	Out	1	This active-High frequency lock output from the CPLLLOCK port of GTHE3_CHANNEL
gth_rxn_in	In	1	This port connects to the GTHRXN diferrential input pin of GTHE3_CHANNEL primitive
gth_rxp_in	In	1	This port connects to the GTHRXP diferrential input pin of GTHE3_CHANNEL primitive
gth_txn_out	Out	1	This port connects to the GTHTXN diferrential output pin of GTHE3_CHANNEL primitive
gth_txp_out	Out	1	This port connects to the GTHYXP diferrential output pin of GTHE3_CHANNEL primitive
GTH Ports for SDI Wrapper			
gth_wiz_reset_all_in	In	1	User signal to reset the phase-locked loops (PLLs) and active data directions of transceiver primitives. An active-High, asynchronous pulse of at least one gth_drpclk_in period in duration initializes the process.
gth_wiz_reset_tx_pll_and_datapath_in	In	1	User signal to reset the transmit data direction and associated PLLs of transceiver primitives. An active-High, asynchronous pulse of at least one gth_drpclk_in period in duration initializes the process.
gth_wiz_reset_rx_pll_and_datapath_in	In	1	User signal to reset the receive data direction and associated PLLs of transceiver primitives. An active-High, asynchronous pulse of at least one gth_drpclk_in period in duration initializes the process.
gth_wiz_txresetdone_out	Out	1	Active-High indication that the transmitter reset sequence of transceiver primitives has completed. This output is synchronous with the tx_usrclk_out.
gth_wiz_rxresetdone_out	Out	1	Active-High indication that the receiver reset sequence of transceiver primitives has completed. This output is synchronous with the rx_usrclk_out.
gth_drpclk_in	In	1	DRP clock to GTH. Normally, this port is driven by same clock as rx_fxdclk_in.
gth_qpll0_clk_in	In	1	This should be connected to gth_qpll0_clk_out port of SDI Wrapper Support. Clock input from the QPLL0OUTCLK port of GTHE3_COMMON primitive.
gth_qpll0_refclk_in	In	1	This should be connected to gth_qpll0_refclk_out port of SDI Wrapper Support. Clock input from the QPLL0OUTREFCLK port of GTHE3_COMMON primitive.
gth_qpll0_lock_in	In	1	This should be connected to gth_qpll0_lock_out port of SDI Wrapper Support. This active-High lock indicator of QPLL0 from the QPLL0LOCK port of GTHE3_COMMON

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

Port Name	I/O	Width	Description
gth_qpll1_clk_in	In	1	This should be connected to gth_qpll1_clk_out port of SDI Wrapper Support. Clock input from the QPLL1OUTCLK port of GTHE3_COMMON primitive.
gth_qpll1_refclk_in	In	1	This should be connected to gth_qpll1_refclk_out port of SDI Wrapper Support. Clock input from the QPLL1OUTREFCLK port of GTHE3_COMMON primitive.
gth_qpll1_lock_in	In	1	This should be connected to gth_qpll1_lock_out port of SDI Wrapper Support. This active-High lock indicator of QPLL1 from the QPLL1LOCK port of GTHE3_COMMON
gth_cpll_refclk_in	In	1	Clock input for GTREFCLK0 of GTHE3_CHANNEL primitive. Normally this port is driven by a clock from IBUFDS_GTE3 primitive.
gth_cpll_lock_out	Out	1	This active-High frequency lock output from the CPLLLOCK port of GTHE3_CHANNEL
gth_rxn_in	In	1	This port connects to the GTHRXN diferrential input pin of GTHE3_CHANNEL primitive
gth_rxp_in	In	1	This port connects to the GTHRXP diferrential input pin of GTHE3_CHANNEL primitive
gth_txn_out	Out	1	This port connects to the GTHTXN diferrential output pin of GTHE3_CHANNEL primitive
gth_txp_out	Out	1	This port connects to the GTHYXP diferrential output pin of GTHE3_CHANNEL primitive

Notes:

1. The RX ports related to the EDH processor are not present on the SMPTE core when the core is generated without the RX EDH processor, an option allowed in the UHD-SDI core GUI. If the RX EDH processor is not included in the UHD-SDI core, the kugth_uhdsdi_<line rate>_wrapper.v SDI wrapper file should not be used as it has all the ports to support the RX EDH processor. Instead, the kugth_uhdsdi_<line rate>_norxedh_wrapper.v SDI wrapper file should be used.

Table 3 lists the parameters that can be applied to the SDI wrapper.

Table 3: **SDI Wrapper Parameter List**

Name	Type	Default	Description
UHD-SDI GTH TX Controller Parameters			
TXPLLCLKSEL_TX_M_0	binary	2'b11	This parameter specifies the value driven to txpllclksel pin of GTHE3_CHANNEL when tx_m_in is Low. Valid values are 2'b00 (CPLL), 2'b11 (QPLL0) and 2'b10 (QPLL1)
TXPLLCLKSEL_TX_M_1	binary	2'b10	This parameter specifies the value driven to txpllclksel pin of GTHE3_CHANNEL when tx_m_in is High. Valid values are 2'b00 (CPLL), 2'b11 (QPLL0) and 2'b10 (QPLL1)

Table 3: **SDI Wrapper Parameter List *(Cont'd)***

Name	Type	Default	Description
UHD-SDI GTH RX Controller Parameters			
RX_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 functions correctly.
RXPLLCLKSEL_TX_M_0	binary	2'b11	This parameter specifies the value driven to rxpllclksel pin of GTHE3_CHANNEL for all rx_mode_out value except 3'b110. Valid values are 2'b00 (CPLL), 2'b11 (QPLL0) and 2'b10 (QPLL1)
RXPLLCLKSEL_TX_M_1	binary	2'b10	This parameter specifies the value driven to rxpllclksel pin of GTHE3_CHANNEL when rx_m_out is High and rx_mode_out is 3'b110. Valid values are 2'b00 (CPLL), 2'b11 (QPLL0) and 2'b10 (QPLL1)
GTH Wizard IP Parameters			
XY_SITE	string	"x0y16"	This parameter specifies the GTH Wizard IP instance location in the FPGA

Video Transport Detector Ports

The RX section of the UHD-SDI core has a 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. 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 which video format family of the transport in the SDI signal. The encoding of this output port is shown in Table 4. The transport detection unit also determines whether the SDI transport is interlaced or progressive and reports this on the `rx_t_scan` output port.

Table 4: **rx_t_family_out Encoding**

rx_t_family	Transport Video Format	Active Pixels
0000	SMPTE ST 274	1920 x 1080
0001	SMPTE ST 296	1280 x 720
0010	SMPTE ST 2048-2	2048 x 1080
0011	SMPTE ST 295	1920 x 1080

Table 4: **rx_t_family_out Encoding *(Cont'd)***

rx_t_family	Transport Video Format	Active Pixels
1000	NTSC	720 x 486
1001	PAL	720 x 486
1111	Unknown	
Others	Reserved	

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

Table 5: **rx_t_rate_out Encoding**

rx_t_rate_out	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

Notes:

1. 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 simply 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_out` port of the SDI wrapper. The counter is cleared by asserted `rx_edh_clr_errcnt_in` High. The user can specify which types of errors are counted by this counter using the `rx_edh_errcnt_en_in` port. This port has 16 unary bits that enable and disable 16 different error types. Any bit that is High enables the corresponding error to be counted by the error counter. Any bit that is Low disables the

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

Table 6: **rx_edh_errcnt_en_in Bits**

Bit #	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

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.

Each ANC, FF, and AP error condition set has five individual error flags. All flags are asserted High to indicate an error condition. For a complete description of the EDH, EDA, IDH, IDA, and UES error flags in the EDH packet, refer to the SMPTE RP 165 document.

- 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_out, rx_edh_ap_out, and rx_edh_ff_out 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_out 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_out, rx_edh_ap_flags_out, and rx_edh_ff_flags_out 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_out, rx_edh_ap_out, and rx_edh_ff_out ports. For example, the EDH flag (bit 0) of the rx_edh_ap_flags_out port indicates that the AP EDH flag was set in the last received EDH packet. However, the rx_edh_ap_out 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_out, rx_edh_ap_flags_out, and rx_edh_ff_flags_out ports are each five bits wide; the encoding of all three ports are identical and is shown in Table 7.

Table 7: **Encoding of rx_edh_anc_flags_out, rx_edh_ap_flags_out, and rx_edh_ff_flags_out Ports**

Bit #	Error
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_out port. The encoding of this port is shown in Table 8.

Table 8: **Encoding of rx_edh_packet_flags Port**

Bit #	Error
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

GTH Initialization and Reset and Change Sequence Failure Codes

If a failure occurs during a GTH RX initialization or reset sequence or during a dynamic change of the RX SDI mode, the rx_change_fail_out port is asserted High and a failure code is output on the rx_change_fail_code_out port. A sequence only ends in failure once 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

RX_RETRY_CNTR_MSB parameter or generic of the v_smpte_uhdsdi_kugth_control.v in the SDI Wrapper module. The number of retries attempted is:

$$\text{Retries} = 2^{\text{RX\_RETRY\_CNTR\_MSB}} - 1$$

The encoding of the rx_change_fail_out port is shown in Table 9.

Table 9: **rx_change_fail_code_out Port Encoding**

Code	Description
0	Reserved
1	When a change of the RX SDI mode is requested that requires changing the RXCDR_CFG2 attribute in the GTH transceiver, the v_smpte_uhdsdi_kugth_control module attempts to do a DRP write cycle to change that attribute. If the v_smpte_uhdsdi_kugth_drp control module detected a mismatch between the written RXCDR_CFG2 value and its actual content after retries, the sequence fails with this failure code
2	When a change of the RX SDI mode is requested that requires changing the RXOUT_DIV attribute in the GTH transceiver, the v_smpte_uhdsdi_kugth_control module attempts to do a DRP write cycle to change that attribute. If the v_smpte_uhdsdi_kugth_drp control module detected a mismatch between the written RXOUT_DIV value and its actual content after retries, the sequence fails with this failure code
3	The gtwiz_reset_rx_datapath_in port of the GTH Wizard IP is asserted after completing a series for DRP and GTH port during a dynamic change to reset the GTH RX portion. If the gtwiz_reset_rx_done_out port of GTH Wizard IP failed to assert after retries, the sequence fails with this failure code
4	When a change of the RX SDI mode is requested that requires changing the RXDATA_WIDTH attribute in the GTH transceiver, the v_smpte_uhdsdi_kugth_control module attempts to do a DRP write cycle to change that attribute. If the v_smpte_uhdsdi_kugth_drp control module detected a mismatch between the written RXDATA_WIDTH value and its actual content after retries, the sequence fails with this failure code
5	When a change of the RX SDI mode is requested that requires changing the RXINT_DATAWIDTH attribute in the GTH transceiver, the v_smpte_uhdsdi_kugth_control module attempts to do a DRP write cycle to change that attribute. If the v_smpte_uhdsdi_kugth_drp control module detected a mismatch between the written RXINT_DATAWIDTH value and its actual content after retries, the sequence fails with this failure code
6	Reserved
7	Reserved

Any sequence failure that results in the rx_change_fail_out port going High causes the GTH RX control logic in the SDI wrapper to stop in a failure condition. The GTH RX may continue to receive an SDI signal, but does not dynamically switch between SDI modes as it normally would. If a failure occurs as indicated by rx_change_fail_out going High, a GTH RX full reset must be done using the rx_rst_in and

gth_wiz_reset_rx_pll_and_datapath_in. This output is synchronous with the gth_drpclk_in. Repeated failures most likely indicate a problem with the design of the application.

If a failure occurs during a GTH TX initialization or reset sequence or during a dynamic change of the TX SDI mode, the tx_change_fail_out port is asserted High and a failure code is output on the tx_change_fail_code_out port. A sequence only ends in failure once 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 TX_RETRY_CNTR_MSB parameter or generic of the v_smpte_uhdsdi_kugth_control.v in the SDI Wrapper module. The number of retries attempted is:

$$\text{Retries} = 2^{\text{TX\_RETRY\_CNTR\_MSB}} - 1$$

The encoding of the tx_change_fail_code port is shown in Table 10.

Table 10: **tx_change_fail_code_out Port Encoding**

Code	Description
0	Reserved
1	When a change of the TX SDI mode is requested that requires changing the TXDATA_WIDTH attribute in the GTH transceiver, the v_smpte_uhdsdi_kugth_control module attempts to do a DRP write cycle to change that attribute. If the v_smpte_uhdsdi_kugth_drp control module detected a mismatch between the written TXDATA_WIDTH value and its actual content after retries, the sequence fails with this failure code
2	When a change of the TX SDI mode is requested that requires changing the TXINT_DATAWIDTH attribute in the GTH transceiver, the v_smpte_uhdsdi_kugth_control module attempts to do a DRP write cycle to change that attribute. If the v_smpte_uhdsdi_kugth_drp control module detected a mismatch between the written TXINT_DATAWIDTH value and its actual content after retries, the sequence fails with this failure code
3	When a change of the TX SDI mode is requested that requires changing the TXOUT_DIV attribute in the GTH transceiver, the v_smpte_uhdsdi_kugth_control module attempts to do a DRP write cycle to change that attribute. If the v_smpte_uhdsdi_kugth_drp control module detected a mismatch between the written TXOUT_DIV value and its actual content after retries, the sequence fails with this failure code
4	The gtwiz_reset_tx_datapath_in port of the GTH Wizard IP is asserted after completing a series for DRP and GTH port during a dynamic change to reset the GTH TX portion. If the gtwiz_reset_tx_done_out port of GTH Wizard IP failed to assert after retries, the sequence fails with this failure code
5	Reserved
6	Reserved
7	Reserved

SDI Timing Constraints

For the SDI Wrapper Support and SDI Wrapper, only the periods of the FPGA IOB ports for GTH reference clocks (MGTREFCLK0/1P/N) and the GTH DRP clock need to be constrained.

The rxusrclk and txusrclk clocks no longer need to be constrained as they are automatically constrained within the GTH Wizard IP and are constrained depending on the maximum target line rate. For 6G-SDI and below, it is usually 148.5 MHz; for 12G-SDI it usually 297 MHz.

The UHD-SDI wrapper contains the NI-DRU used to recover data in SD-SDI mode. The NI-DRU only runs in SD-SDI mode and in that mode, the RXOUTCLK has a frequency of 148.5 MHz. In applications that support 12G-SDI, the RXOUTCLK is constrained to 297 MHz and the NI-DRU usually does not meet timing at 297 MHz. But, it doesn't need to because it is only active when RXOUTCLK is 148.5 MHz. An additional set of constraints can be applied to the NI-DRU so that the NI-DRU is constrained to 148.5 MHz while the rest of the RX section is constrained to 297 MHz. The following two constraints are used in the example design to accomplish this:

```
set_property KEEP_HIERARCHY true [get_cells \
uhdsdi_demo/sdi_4ch_rxtx/genblk1[0].sdi_wrapper_support/sdi_wrapper/uhdsdi_kugth_ctrl/NIDRU]

create_generated_clock -name nidru_clk0 -source [get_pins -of [get_clocks rxoutclk_out[0]]] \
-divide_by 2 [get_pins \
uhdsdi_demo/sdi_4ch_rxtx/genblk1[0].sdi_wrapper_support/sdi_wrapper/uhdsdi_kugth_ctrl/NIDRU/ \
Inst_dru/CLK]
```

A KEEP_HIERARCHY constraint is applied to the NI-DRU module so that the clock name that must be identified in the next constraint is not changed by synthesis. The get_cells portion of this constraint uses a path to the NI-DRU of SDI/GTH Control/NIDRU. In the example design, the UHD-SDI wrapper has an instance name of SDI. Change the SDI portion of this path to the instance name of the UHD-SDI wrapper in your application. The KEEP_HIERARCHY constraint is only applied for synthesis and does not apply for implementation so it does not interfere with any optimizations that the implementation tool may perform.

The create_generated_clock constraint creates a hierarchical clock just for the NI-DRU. This is not a physically separate clock. It is a logical clock used for timing analysis only. The NI-DRU is still driven by the RXOUTCLK of the GTH. This constraint tells the timing analyzer that the clock connected to the NI-DRU's CLK port is derived from the GTH RXOUTCLK but is half the maximum frequency. RXOUTCLK is constrained to 297 MHz so the NI-DRU is constrained to 148.5 MHz.

Vivado considers all clocks to be related unless specified. 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

The example SDI demonstration application is included with this application note. The source code for this demonstration is provided in Verilog only. Instructions for building these demonstrations using the Vivado IDE are included in the readme.txt file located in the `xapp1248-smpte-sdi-interfaces-ultrascale-gth-transceivers.zip` file along with the source code. Pregenerated FPGA configuration files are also provided for the demonstration that can be loaded onto an UltraScale FPGA KCU105 evaluation board. This demonstration require an inrevium TB-FMCH-12GSDI FMC, which provides the SDI cable drivers and SDI cable equalizers connected to the FMC connector of the KCU105 board. The inrevium FMC also provides SDI-specific clock sources that are used as reference clocks for the GTH transceivers.

SDI Demonstration

This demonstration application includes a pair of SDI RX and SDI TX interfaces on KCU105 evaluation board. It requires a Fidus 12G-SDI FMC board connected to the HPC FMC connector of the KCU105 board. The example design has a single UHD-SDI transmitter driven by a test pattern generator. It supports operation at SD-SDI, HD-SDI, 3G-SDI (levels A and B), 6G-SDI, and 12G-SDI. The UHD-SDI transmitter is controlled by a Vivado Analzyer VIO module. The example design also has a single UHD-SDI receiver which can operate in the same modes as the transmitter. The status of the UHD-SDI receiver is monitored by a Vivado Analyzer VIO module. The data streams, line numbers, and video timing signals output by the UHD-SDI receiver are captured by a Vivado Analyzer ILA module and can be inspected in the Vivado Analyzer tool.

The SDI TX is driven by a Video Test Pattern Generator. The SDI mode, the video format and video pattern of SDI TX can be selected using the Vivado VIO window in the Vivado Hardware Manager. The status of the SDI RX can be monitored using another Vivado VIO window and the video data received by the SDI RX can be captured and viewed using a Vivado ILA window.

The inrevium SDI FMC board has five connectors for the SDI interfaces. The two connectors at the right most (Figure 18) are the only unidirectional SDI interfaces, the right-most being the CH0 TX and the other CH0 RX. These connectors are the only ones which are used in this demonstration. The second, third, and fourth SDI interfaces have only a single connector each, CH1, CH2 and CH3. These are bidirectional interfaces and can be controlled by `F_CHn_DIR` pins of the FMC card.

Figure 17 is a block diagram of the demonstration, showing SDI channel 0 which is connected to the first GTH transceiver in the quad.

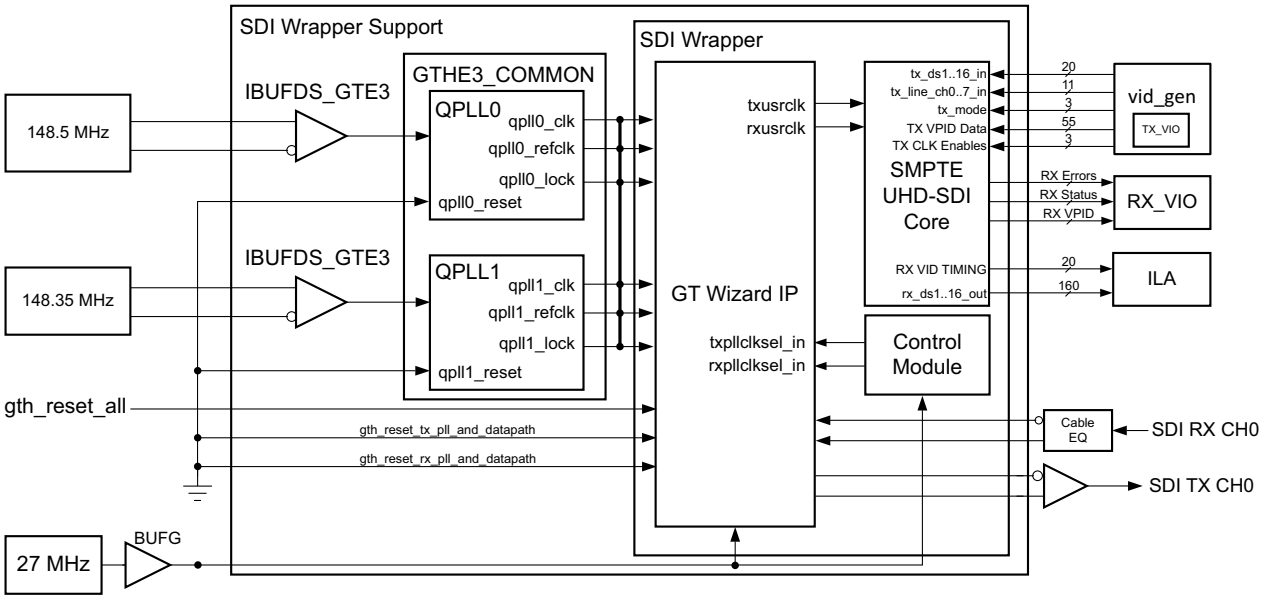


Figure 17: **UHD-SDI Example Design Block Diagram**

The inrevium SDI FMC board has 148.5 MHz and 148.5/1.001 MHz oscillators which this demo uses to supply reference clocks to the QPLL0 and the QPLL1 that goes to each transceiver. The 148.5 MHz reference clock is used by the QPLL0 and the 148.5/1.001 MHz reference clock is used by the QPLL1. The GTH transmitters are dynamically switched between the serial clock from the QPLL0 and the QPLL1 to support all SDI bit rates.

The LMH1983 device on the inrevium board supplies a 27 MHz clock to the UltraScale FPGA that is used for the DRP clock and fixed frequency clock required by the control module.

To make it simplify scaling the SDI interface up to four in this demo, a four SDI channel wrapper (`kugth_uhdsdi_4ch_wrapper.v`) was created which instantiates one SDI Wrapper Support and three SDI Wrapper. The video generator, main, and RX Vivado VIOs were all placed inside a Verilog Generate statement to easily increase the number of channels.

The following are required to run the quad SDI demonstration:

- Xilinx Kintex® UltraScale FPGA KCU105 Evaluation Kit
- inrevium Fidus TB-FMCH-12GSDI SDI FMC
- 2 HD-BNC to BNC converter cables
- SDI signal source
- SDI signal sink (waveform monitor or other device to view signal from SDI transmitters)
- Xilinx Vivado IDE

The inrevium SDI FMC board must be connected to the FMC HPC connector on the KCU105 board as shown in Figure 18.

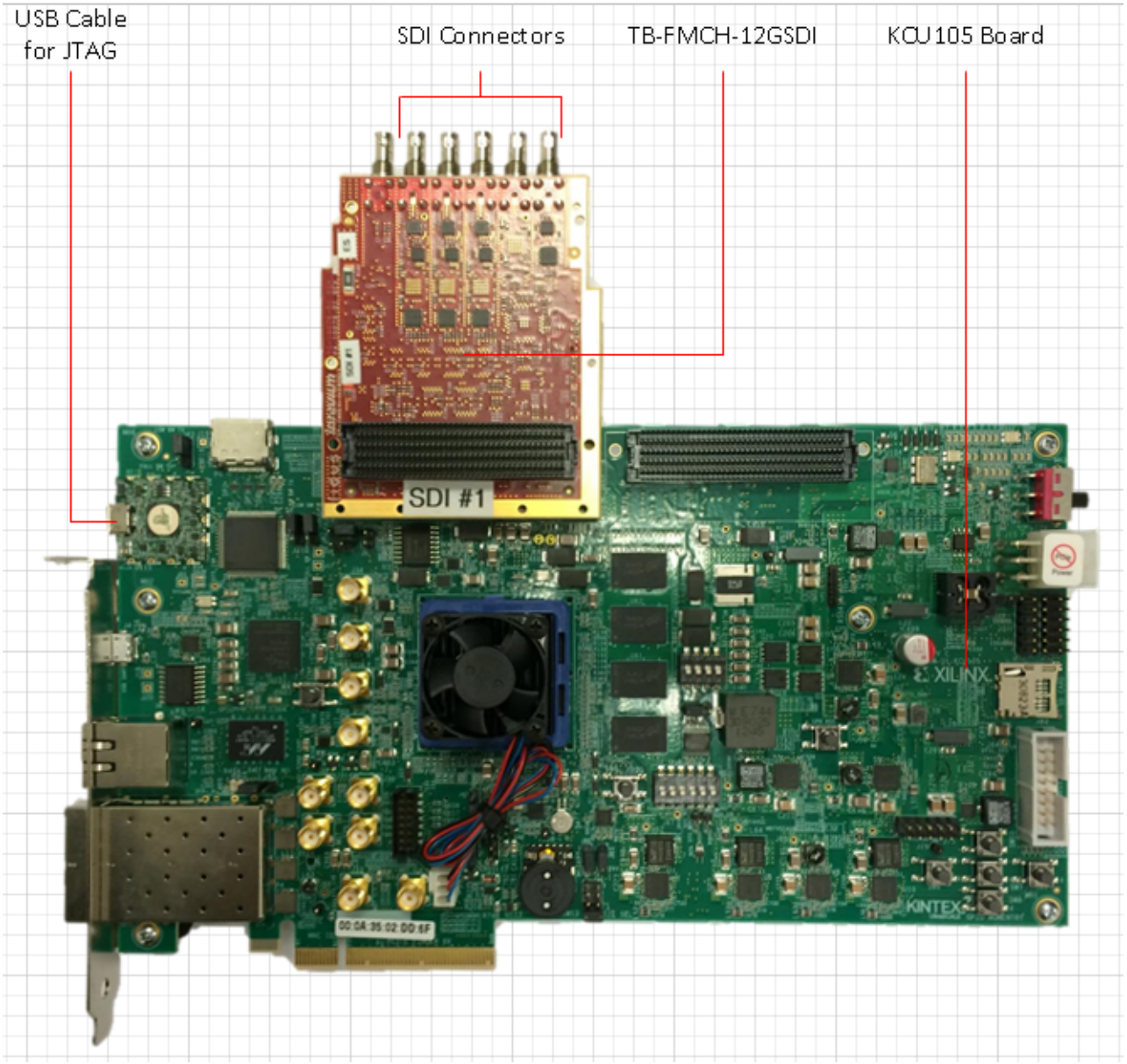


Figure 18: **KCU105 Board with TB-FMCH-12GSDI Board Connected**

The Vivado Hardware Manager must be opened to control the SDI transmitters and to look at the status and received data from the SDI receivers via the VIOs. The KCU105 board must be connected to a PC by the USB JTAG cable provided with the board.

The configuration file named `kcu105_uhdsdi_demo.bit` is provided with this application note must be loaded into the Kintex UltraScale FPGA on the KCU105 board using Vivado Hardware Manager. After this, the hardware (LTX) configuration file is loaded where three `HW_VIOs` and an `HW_ILA` automatically open. A Vivado project file (`bit_files.xpr`) is provided with this application note so that the `HW_VIOs` appear as shown in Figure 19 instead of default HEX or binary view. Follow the instructions in Configuring FPGA with Precompiled Bit File to open the `bit_files.xpr` and to download the precompiled bitstream.

Configuring FPGA with Precompiled Bit File

1. Unzip xapp1248.zip.
2. Connect to the KCU105 via the UART USB port.
3. Power on the KCU105.
4. Connect to the KCU105 System Controller and set the VADJ to 1.8V.

   Note, the single microUSB connector provides access to both the Zynq system controller's UART and to the UltraScale FPGA's UART. In the Windows Device Manager, the Enhanced COM port associated with the CP210x, is the one connected to the System Controller.

   Open a Terminal window (115200, 8, N, 1) and set the COM port to the one communicating with the KCU105 System Controller. (Note, the single microUSB connector provides access to both the Zynq system controller's UART and to the UltraScale FPGA's UART.) In the Windows Device Manager, the Enhanced COM port associated with the CP210x, is the one connected to the System Controller.

   After the UART terminal is connected, power cycle the KCU105 to refresh the System Controller Menu in the UART terminal. Select the following option in the System Controller Menu.

   Adjust FPGA Mezzanine Card (FMC) Settings

   then in the next menu, select:

   Set FMC VADJ to 1.8V

5. Look for the VADJ power good on DS19 LED located near the power switch on the KCU105 board.
6. Connect up the KCU105 via the JTAG USB port.
7. In Vivado TCL Console, enter the following sequentially:
   a. cd <unzip_dir>\ready_for_download
   b. source bit_files.tcl
8. Wait for project load and the FPGA to program.

Note: If the UHD-SDI RX is not locking, make sure that the VADJ power to FMCH port is at 1.8V VADJ power good is indicated by DS19 LED located near the power switch of the KCU105 board. If the LED is off, the VADJ power can be set through KCU105's system controller's UART interface.

Demo Status LED

- GPIO_LED_0- RX locked to SD-SDI mode
- GPIO_LED_1- RX locked to HD-SDI mode
- GPIO_LED_2- RX locked to 3G-SDI mode
- GPIO_LED_3- RX locked to 6G-SDI mode

- GPIO_LED_4- RX locked to 12G-SDI mode
- GPIO_LED_5- RX bitrate indicator
- GPIO_LED_6- RX change done indicator
- GPIO_LED_7- FMC Initialization Done

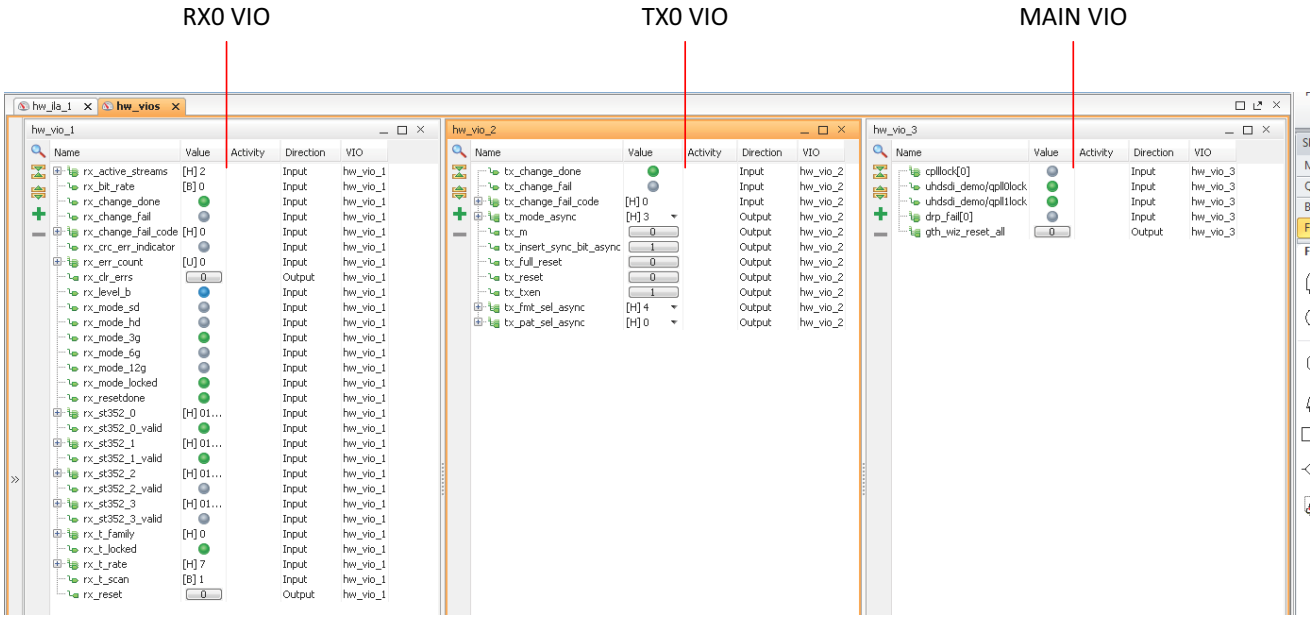


Figure 19: **Vivado Hardware Manager Main and CH0 VIO View**

To observe the signal being generated by the SDI transmitter, an SDI waveform monitor or SDI display device must be connected to the output of the CH0 TX, or the SDI transmitter output can be connected back to the CH0 RX input on the inrevium FMC with a cable. The SDI connectors on the inrevium SDI FMC board are not standard BNC connectors, so adapter cables are required to go from HD-BNC connectors to regular BNC connectors.

Each SDI transmitter has a VIO control window. The VIO control window for TX0 is shown in Figure 20.

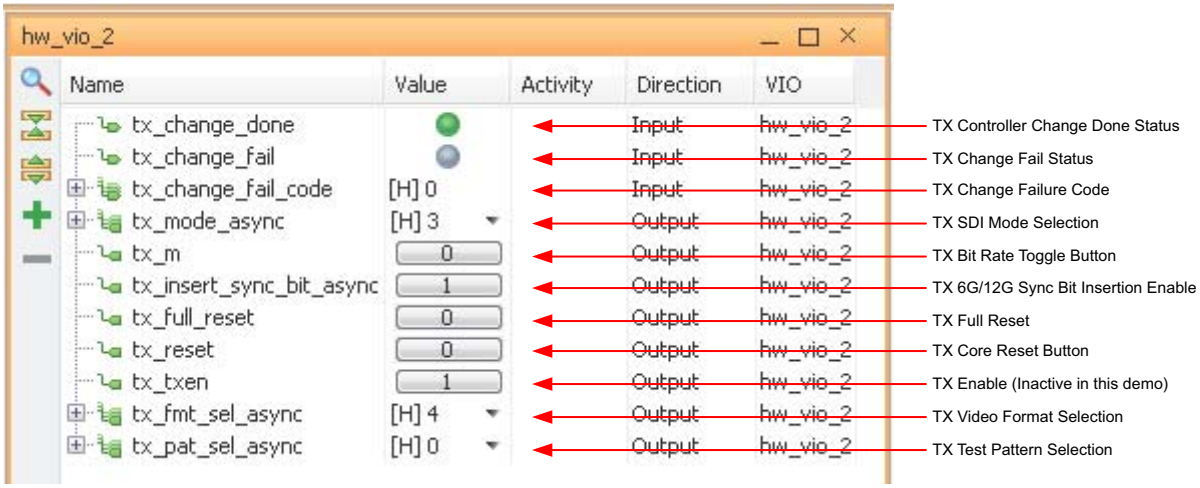


Figure 20: **SDI Demonstration TX0 VIO Control Window**

The three items at the top of the TX VIO window indicate the status of the last GTH TX initialization or dynamic change sequence. If the last sequence completed normally, the Change Done indicator is High. 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.

The `tx_resetdone` indicator shows the status of the GTH Wizard IP `gth_wiz_txresetdone_out` output port. During normal operation, this indicator is High.

The TX Bit Rate toggle button, TX Video Format selection field, and the TX SDI Mode selection field work together to select the format of the SDI signal generated by the SDI transmitter as shown in Table 11.

Table 11: **Quad SDI Demonstration TX Video Format Selection**

TX Video Format	SD-SDI (SDI Mode = 1)	HD-SDI (SDI Mode = 0)		3G-SDI Level A (SDI Mode = 2)		3G-SDI Level B (SDI Mode = 3)		6G-SDI (SDI Mode = 4)		12G-SDI (SDI Mode = 5)	
		TX Bit Rate = 0	TX Bit Rate = 1	TX Bit Rate = 0	TX Bit Rate = 1	TX Bit Rate = 0	TX Bit Rate = 1	TX Bit Rate = 0	TX Bit Rate = 1	TX Bit Rate = 0	TX Bit Rate = 1
0	NTSC	720p 50 Hz									
1	PAL	1080pSF 24 Hz	1080pSF 23.98 Hz								
2	NTSC	1080i 60 Hz	1080i 59.94 Hz								
3	PAL	1080i 50 Hz									
4	NTSC	1080p 30 Hz	1080p 29.97 Hz	1080p 60 Hz	1080p 59.94 Hz	1080p 30 Hz	1080p 29.97 Hz	2160p 30 Hz	2160p 29.97 Hz	2160p 30 Hz	2160p 29.97 Hz
5	PAL	1080p 25 Hz		1080p 50 Hz		1080p 25 Hz		2160p 25 Hz		2160p 25 Hz	
6	NTSC	1080p 24 Hz	1080p 23.98 Hz								
7	PAL	720p 60 Hz	720p 59.94 Hz								

The TX Video Pattern value selects the video test pattern generated by the Video Test 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 check field
- 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 check field

In addition to the tx_mode_in values specified in Table 1, TX_MODE can also be set to 3'b011 to stream 3G-SDI Level B patterns.

Each SDI receiver has a VIO window to monitor the status of the receiver and an ILA window through which the video data received by the SDI RX and can be viewed. Figure 21 shows the VIO window for RX0.

The three items at the top of the RX VIO window indicate the status of the last GTH RX initialization or dynamic change sequence. If the last sequence completed normally, the Change Done indicator is High. 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.

The RX Error Indicator is High (red) if any CRC or EDH error has been detected and Low (gray) 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 relock to the signal. However, the error counter is automatically cleared early in the process of locking to the new SDI signal. Thus, after the SDI RX has fully locked to the new SDI signal, the error count is typically not zero.

The RX Level B Indicator is High (Blue) when RX is receiving 3G-SDi Level B signal, it is Low (gray) otherwise.

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

The RX SDI Mode shows the rx_mode_out current value according to Table 1.

The RX Locked Status is High (green) when the SDI RX is locked to the incoming SDI signal and Low (gray) when it is not locked.

The RX Reset Done Indicator is High (green) when GTH Wizard IP completed a GTH RX reset sequence.

The RX Video Family, RX Frame Rate Transport, and RX Scan Mode provide information about the video that has been detected and can be decoded using Table 1.

The ST 352 Payload ID Data Bytes are the four data bytes of the ST 352 payload ID packet. They are shown with byte 1 on the left and byte 4 on the right. They are only valid when the ST 352 Payload Packet Valid indicator is green.

The RX Controller Reset Button triggers a reset routine in the RX Controller Module.

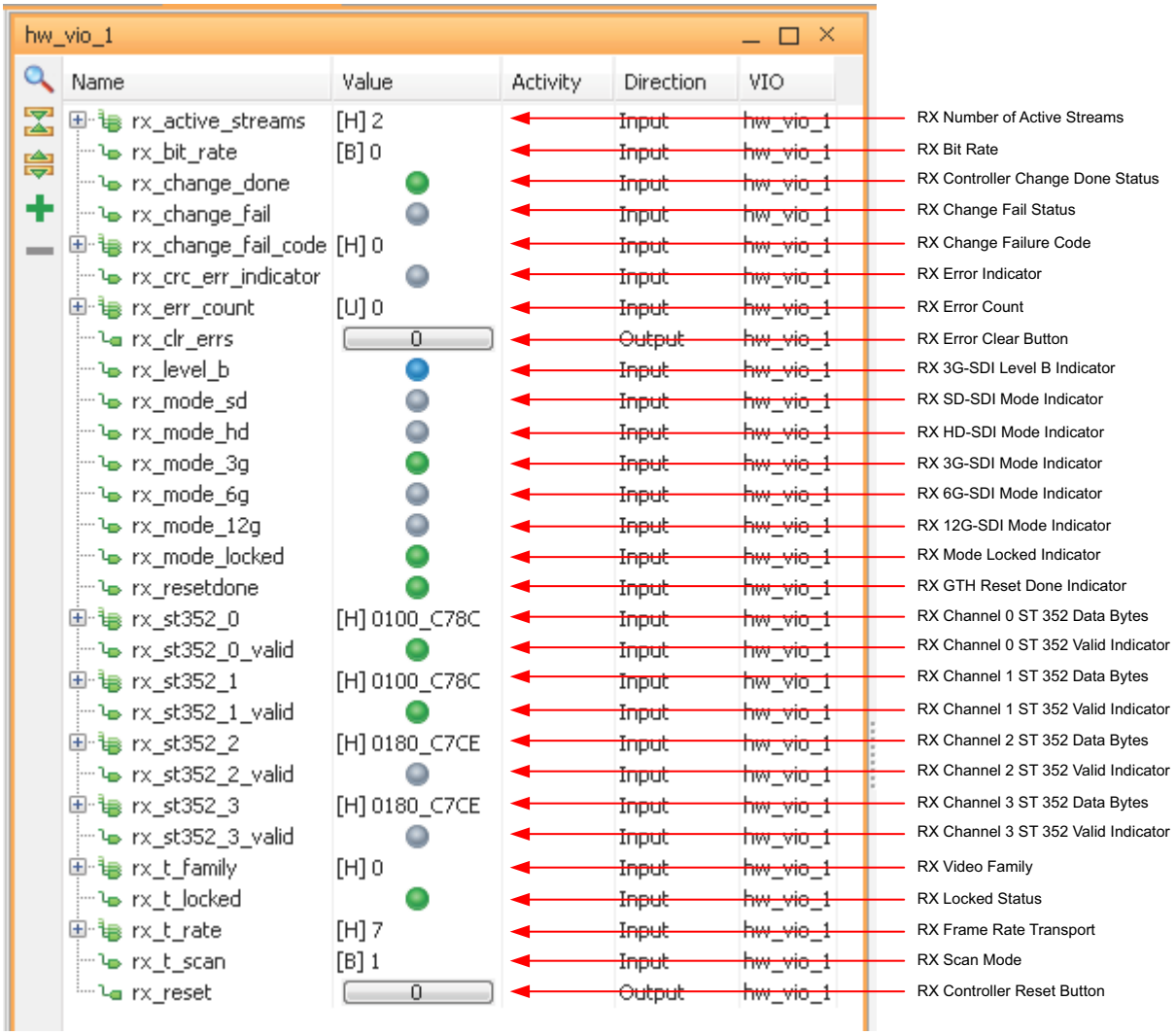


Figure 21: **SDI Demonstration RX Status Window**

Figure 22 shows an ILA window screen shot of an incoming 12G-SDI stream. Refer to *Vivado Design Suite Tutorial: Programming and Debugging* [Ref 14] for ILA usage.

Figure 22: **Vivado ILA to View RX Data in the SDI Demonstration**

Reference Design

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

https://secure.xilinx.com/webreg/clickthrough.do?cid=390742, registration required.

Tool Flow and Verification

The following checklist indicates the tool flow and verification procedures used for the provided reference design.

Table 12: **Reference Design Checklist**

Parameter	Description
General	
Developer name	Gilbert Magnaye, John Snow
Target devices	UltraScale devices with GTH transceivers
Source code provided	Yes
Source code format	Verilog
Design uses code/IP from existing Xilinx application note/reference design, IP catalog, 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 2017.4
Implementation software tools/version used	Vivado 2017.4
Static timing analysis performed	Yes
Hardware Verification	
Hardware verified	Yes
Hardware platform used for verification	KCU105 and TB-FMCH-12GSDI boards

Requirements

Hardware

- Xilinx Kintex UltraScale FPGA KCU105 Evaluation Kit
- inrevium Fidus TB-FMCH-12GSDI SDI FMC
- 2 HD-BNC to BNC converter cables
- SDI signal source
- SDI signal sink (waveform monitor or other device to view signal from SDI transmitters)

Software

- Xilinx Vivado Design Suite

Reference Design Steps

Compiling SDI Demonstration

Compiling the reference design is done in four steps and takes approximately 30 minutes to complete. To begin the compilation, perform the following steps:

1. Unzip xapp1248.zip.
2. Open Vivado 2017.4 or higher.
3. In Vivado TCL Console, Key-in the following sequentially.
   a. `cd <unzip_dir>\xapp1248`
   b. `source kcu105_uhdsdi_demo_script.tcl`
4. Wait for project compilation to finish.

The `kcu105_uhdsdi_demo_script` TCL executes the following steps to complete the bitstream generation.

1. Creating Project.
2. Importing RTL sources.
3. Adding design constraint files.
4. Generating Xilinx IPs.
   a. tx_vio
   b. rx_vio

c. rx_ila

d. GT Wizard IP for x0y16 (v_smpte_uhdsdi_gtwiz_x0y16)

5. Build IPI Subsystem for inrevium 12G-SDI FMC Card control.
6. Running compilation.

inrevium 12G-SDI FMC Card Controller

This application note is supplied with IP Integrator based FMC controller to provide access and control to the I2C and SPI devices in the FMC card. The controller is instantiated in the project hierarchy as `system_basic`, and is composed of SPI, IIC GPIO IPs, and basic IP components to run a basic MicroBlaze™ application. The GPIO's main purpose is to allow FMC channel selection during configuration and to indicate initialization done state.

There are three stages in the initialization process, first is the clock switch configuration which selects the two on-board crystal oscillator to supply 148.35 MHz and 148.5 MHz reference clocks to the GTH. Next is the generic SPI devices (cable driver, equalizer and reclocker) initialization, such as setting of output swing, input equalization factor and muting of one of two SDI cable driver outputs. Third is the SPI devices initialization based on the silicon version. Macom, the manufacturer of the cable equalizer and reclocker for the FMC card has published errata to their devices. Special SPI register initialization is needed for different silicon version (denoted by die ID) to ensure error free SDI reception. So far, errata 235x4-ERR-001-A, 23145-ERR-001-A, 23145-ERR-001-C, 23145-ERR-001-D and Errata M235x4-ERR-001-C were considered when the FMC controller was written.

There is a UART GUI interface to allow more flexibility on controlling the 12G-SDI FMC card. Users can selectively make register modifications to the I2C and SPI devices in the FMC card by following the instructions in the GUI shown below:

```
-----------------------
--  FIDUS Main Menu  --
-----------------------

 Select option
 1 = Re-Init
 2 = IIC Dev Select
 3 = SPI CH0 Select
 4 = SPI CH1 Select
 5 = SPI CH2 Select
 6 = SPI CH3 Select
 ? = help
------------------
>
```

Recompiling FMC Controller SDK Project

The SDK environment must be prepared after the completion of `kcu105_uhdsdi_demo_script.tcl` script. This is done by exporting the project's hardware information and importing the SDK source codes.

1. Export hardware: In Vivado 2017.4 select, **File>Export>Export Hardware**.
   a. In the Export Hardware pop-up window, check **Include bitstream** option.
   b. Set the Export to: `<unzip_dir>\xapp1248\srcs\fidus_fmc_ctlr\SW`.
2. Launch Xilinx SDK 2017.4 select, **File>Launch SDK**.
   a. Set both **Exported location** and **Workspace** fields to:

      `<unzip_dir>\xapp1248\srcs\fidus_fmc_ctlr\SW`

   b. In SDK, create a new Board Support Package; **File>New>Board Support Package**. Assign Project Name as **fidus_fmc_ctlr_bsp** then click **Finish**.
   c. In **Board Support Package Settings**, click **OK**.
3. Import SDK Sources: In SDK 2017.4 select, **File>Import**.
   a. In the **Import** pop-up window, select **General->Existing Projects into Workspace**.
   b. Click **Next**.
   c. Click on **Browse...** button, and ensure that it points to the corresponding folders:

      `<unzip_dir>\xapp1248\srcs\fidus_fmc_ctlr\SW`

   d. Click **OK**.
   e. Make sure **fidus_fmc_ctlr** is checked.
   f. Click **Finish**.
4. Assign **fidus_fmc_ctlr_bsp** to **fidus_fmc_ctlr**.
   a. In SDK, right click on **fidus_fmc_ctlr** folder.
   b. Click on **Change Referenced BSP**.
   c. Select **fidus_fmc_ctlr_bsp** and click **OK**.

FPGA Resource Usage

Table 13 shows the FPGA resources required for an SDI interface with a Kintex UltraScale GTH transceiver. The resource usage includes all the modules required to implement the interface included in one SDI Wrapper Support instance. Resource usage is shown for various common configurations.

The results shown were achieved with Vivado 2017.4.

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

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 in the SDI wrapper. This fixed frequency clock is usually also used as the GTH DRP clock. Only one such fixed frequency global clock is required regardless of the number of SDI interfaces are implemented in the FPGA.

Table 13: **Kintex UltraScale GTH SDI Interface FPGA Resource Usage**

UHD-SDI IP and Wrapper Configuration		FF	LUT	Memory LUT	BUFG
Max Line Rate	UHD-SDI Core				
3G-SDI	RX/TX with EDH processor	5658	6488	140	2
	RX/TX without EDH processor	5245	5983	139	2
6G-SDI	RX/TX with EDH processor	6386	6870	140	2
	RX/TX without EDH processor	5973	6319	139	2
12G-SDI 8 Data Streams	RX/TX with EDH processor	6387	7290	140	2
	RX/TX without EDH processor	5974	6760	139	2
12G-SDI 16 Data Streams	RX/TX with EDH processor	6851	7450	140	2
	RX/TX without EDH processor	6438	6911	139	2

Constraints

Example constraint files are provided with the reference designs and can be used as examples of the timing and placement constraints required for SDI interfaces. Typically, the only timing requirements are period constraints on the GTH transceiver reference clock IOB pins and the fixed frequency clock used for the DRPCLK and the SDI Wrapper's `rx_fxdclk_in` port. The GTH reference clocks constraints should specify the period of these clocks as 148.5 MHz (often rounded up to 150 MHz). I/O placements and clock constrains for the GTH transceivers are already specified within each GTH Wizard IP.

Conclusion

This document describes how to use the SMPTE UHD-SDI core and the UltraScale GTH transceivers to implement SDI interfaces compatible with the SMPTE SD-SDI, HD-SDI, 3G-SDI, 6G-SDI, and 12G-SDI standards. The UltraScale GTH 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 UltraScale FPGA design.

Glossary

Table 14 describes the terms used in this application note.

Table 14: **Terms and Definitions**

Name	Description
12G-SDI	Common name for SMPTE ST 2082-1, the 12Gb/s serial digital interface.
3G-SDI	Common name for SMPTE ST 424, the 3Gb/s serial digital interface. 3G-SDI supports three mapping modes defined in ST 425-1 called 3G-SDI level A, level B-DL, and level B-DS. Refer to ST 425-1 for details about these mapping modes.
6G-SDI	Common name for SMPTE ST 2081-1, the 6Gb/s serial digital interface.
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 ST 291-1.
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 RP 165.
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 ST 292-1 1.5 Gb/s serial digital interface.
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.

Table 14: **Terms and Definitions *(Cont'd)***

Name	Description
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 ST 372 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. The 3G-SDI level B-DL transport carries both link 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 ST 352. 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, etc) 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 SMPTE ST 259, 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 ST 259, 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.
Synchronous switching (point, interval, line)	SMPTE RP 168 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.

References

Available from the Society of Motion Picture and Television Engineers (www.smpte.org):

1. *RP 165: Error Detection Checkwords and Status Flags for Use in Bit-Serial Digital Interfaces for Television*
2. *RP 168: Definition of Vertical Switching Point for Synchronous Video Switching*
3. *ST 259: Television - SDTV Digital Signal/Data - Serial Digital Interface*
4. *ST 291-1: Television - Ancillary Data Packet and Space Formatting*
5. *ST 292-1: 1.5 Gb/s Signal/Data Serial Interface*
6. *ST 344: Television - 540 Mb/s Serial Digital Interface*
7. *ST 352: Payload Identifier Codes for Serial Digital Interfaces*
8. *ST 372: Dual Link 1.5 Gb/s Digital Interface for 1920x1080 and 2048 x 1080 Picture Formats*
9. *ST 424: Television - 3 Gb/s Signal/Data Serial Interface*
10. *ST 425-1: Source Image Format and Ancillary Data Mapping for the 3Gb/s Serial Interface*
11. *ST 2081-1: 6Gb/s Signal/Data Serial Interface - Electrical*
12. *ST 2082-1: 12Gb/s Signal/Data Serial Interface - Electrical*

Available from Xilinx (www.xilinx.com):

13. *UltraScale Architecture GTH Transceivers User Guide* (UG576)
14. *Vivado Design Suite Tutorial: Programming and Debugging* (UG936)
15. *UltraScale FPGAs Transceivers Wizard* (PG182)
16. *Kintex UltraScale Architecture Data Sheet: DC and AC Switching Characteristics* (DS892)
17. *LogiCORE IP SMPTE UHD-SDI Product Guide* (PG205)
18. *Clock and Data Recovery Unit based on 20-Bit-Wide Oversampled Data* (XAPP1240)

Revision History

The following table shows the revision history for this document.

Date	Version	Changes
02/01/2018	1.4	Updated PLL use models.
06/08/2016	1.3	Updated hardware components.
08/14/2015	1.2	Updated Table 11.
06/18/2015	1.1	Text edits.
04/01/2015	1.0	Initial Xilinx release.

Please Read: Important Legal Notices