Implementing Deterministic Latency for CPRI and OBSAI Protocols in Altera Devices

AN-610-2.1 Application Note

This application note describes the transceiver configuration and clocking scheme to implement deterministic latency, and how you can implement deterministic latency with the transceivers in the Stratix® IV, HardCopy® IV, Arria® II, and Cyclone® IV devices for the following protocols :

- Common Public Radio Interface (CPRI)
- Open Base Station Architecture Initiative Reference Point 3-01 (OBSAI RP3-01)

Based on the implementation in this application note, you can create your designs for proprietary CPRI, OBSAI RP3-01, or other interfaces requiring deterministic latency.

This application note covers the following topics:

- "Overview of the CPRI and OBSAI RP3-01 Protocols" on page 1
- "Transceiver Support for CPRI and OBSAI RP3-01 Applications" on page 2
- "Implementing Deterministic Latency for CPRI and OBSAI RP3-01 Interfaces" on page 3
- "Transceiver Channel Instantiation" on page 5
- "Input Reference Clocks and Transmit Side Clock Generation" on page 6
- "Interface Clocking" on page 9
- "Phase Detector Logic" on page 11
- "Design Considerations for Auto-Rate Negotiation" on page 13

Overview of the CPRI and OBSAI RP3-01 Protocols

The CPRI and OBSAI RP3-01 protocols are point-to-point, high-speed serial interfaces in wireless applications for connecting base station component and remote radio heads.

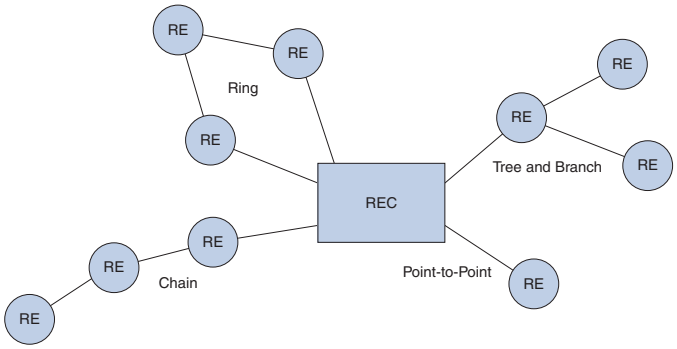
- Single-hop connections—CPRI connects the radio equipment (RE) modules to a radio equipment controller (REC). The REC is always the master port and the RE module is the slave port.
- Multi-hop connections—For high-bandwidth, the CPRI connections can be used to chain multiple RE modules to a single REC. For links between two RE modules, the RE port closest to the REC becomes the master port.

101 Innovation Drive
San Jose, CA 95134
www.altera.com

Subscribe

Figure 1 shows various CPRI topologies.

Figure 1. CPRI Topologies



In the CPRI and OBSAI RP3-01 specifications, the requirements for the accuracy of round-trip delay measurements is stringent. For example, in the CPRI specification, the round-trip delay measurement accuracy—excluding the cable—must be within ±16.276 ns for single- and multi-hop connections. In multi-hop connections, the allowed delay uncertainty is accumulated over the number of hops in the connection.

To reduce your development time, Altera offers a complete and easy to use intellectual property (IP) core for building a CPRI v4.1 system that includes the transceiver.

For more information about the Altera® CPRI IP solution, refer to the Altera CPRI IP web page.

Transceiver Support for CPRI and OBSAI RP3-01 Applications

The Stratix IV, HardCopy IV, Arria II, and Cyclone IV devices include embedded transceivers with deterministic latency features that comply with the delay accuracy requirements of the CPRI and OBSAI RP3-01 specifications. The deterministic latency features enable you to accurately compute the transceiver datapath latencies when implementing the interfaces.

Table 1 lists the data rates supported for the CPRI and OBSAI RP3-01 implementations using the transceivers in the Stratix IV, HardCopy IV, Arria II, and Cyclone IV devices.

Table 1. Supported Data Rates for CPRI and OBSAI RP3-01 Implementations (Part 1 of 2)

Protocol	Data Rate (Mbps)	Stratix IV	HardCopy IV	Arria II	Cyclone IV
CPRI	614.4	✓	✓	✓	✓
	1228.8	✓	✓	✓	✓
	2457.6	✓	✓	✓	✓
	3072	✓	✓	✓	✓
	4915.2	✓	✓	✓	—
	6144	✓	✓	✓	—

Table 1. Supported Data Rates for CPRI and OBSAI RP3-01 Implementations (Part 2 of 2)

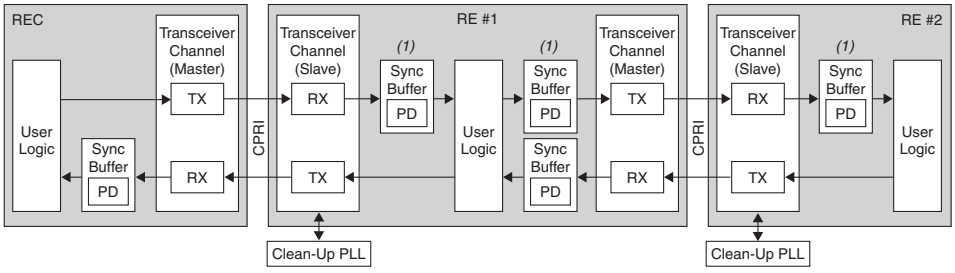
Protocol	Data Rate (Mbps)	Stratix IV	HardCopy IV	Arria II	Cyclone IV
OBSAI RP3-01	768	✓	✓	✓	✓
	1536	✓	✓	✓	✓
	3072	✓	✓	✓	✓
	6144	✓	✓	✓	—

For information about the supported data rate for implementing deterministic latency on other proprietary protocols, refer to the respective device family datasheet.

Implementing Deterministic Latency for CPRI and OBSAI RP3-01 Interfaces

This section describes the methods to configure the transceiver channels with deterministic latency features that are required for implementing the CPRI and OBSAI RP3-01 interfaces. Figure 2 shows an overview of CPRI implementation using the transceivers.

Figure 2. Overview of CPRI Implementation Using Transceivers



Note to Figure 2:

(1) The synchronization buffer with phase detector is not required for transceiver implementation with PLL PFD feedback.

On the transceiver channels, use the Deterministic Latency functional mode. In this mode, the transmitter channel is configured without delay uncertainty. On the `tx_clkout` port, the datapath latency is fixed relative to the core fabric interface clock.

To fix the transmitter channel datapath latency relative to the transmitter phase-locked loop (PLL) input reference clock on the `pll_inclk` port, enable the PLL phase frequency detector (PFD) feedback. Using the PLL PFD feedback simplifies port implementations in remote radio heads—effectively eliminating the need for additional logic in the core fabric for implementing a synchronization buffer with a phase detector.

For information about the port implementations requirement in remote radio heads without PLL PFD feedback, refer to "Interface Clocking" on page 9. For information about the phase detector implementation, refer to "Phase Detector Logic" on page 11.

In CPRI, the PLL PFD feedback is optional for REC port implementation. To simplify your interface design, Altera recommends that you enable the PLL PFD feedback for RE port implementation.

For the receiver, the channel datapath is configured without latency uncertainty. In the word aligner block, the latency variation from the link synchronization function is deterministic with the `rx_bitslipboundaryselectout` port.

Optionally, you can fix the round-trip transceiver latency for port implementation in the remote radio head to compensate the latency variation in the word aligner block using the `tx_bitslipboundaryselect` port. The `tx_bitslipboundaryselect` port is available to control the amount of bits to be slipped in the transmitter serial data stream.

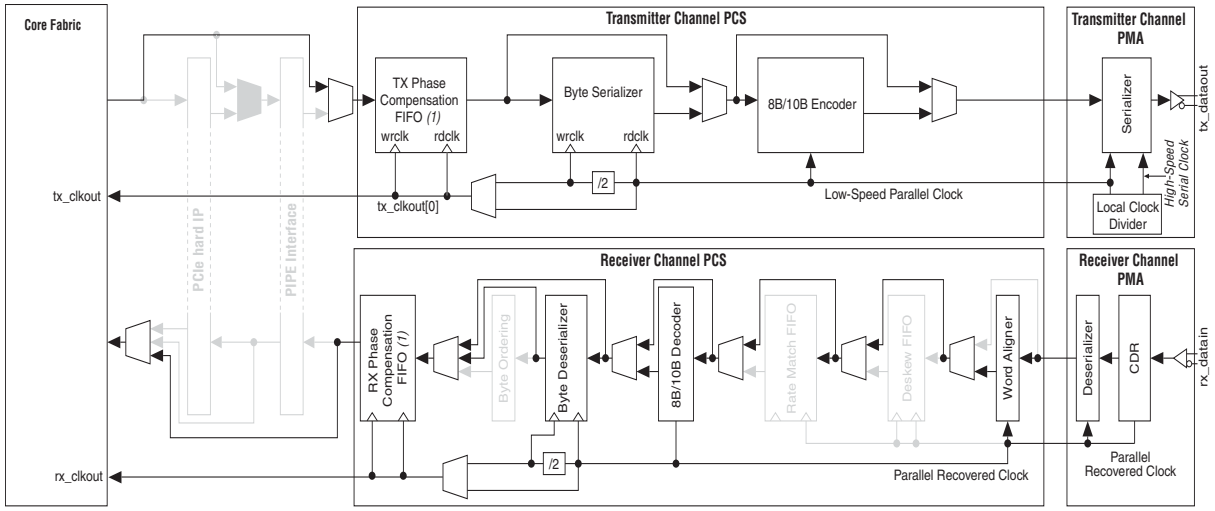
You can also use the `tx_bitslipboundaryselect` port to round up to a whole number the round-trip latency cycles. If you use the byte deserializer in the transceiver, you must create additional logic in the core fabric to determine if the comma byte is received in the lower or upper byte of the word. The delay is dependent on which word the comma byte appears.

The total transmitter and receiver channel datapath latencies are computed using the following equations:

- Total transmitter channel datapath latency = transmitter fixed latency + `tx_bitslipboundaryselect` delay
- Total receiver channel datapath latency = receiver fixed latency + `rx_bitslipboundaryselectout` delay + byte deserializer delay

Figure 3 shows the transceiver configuration in the Deterministic Latency mode. In the receiver channel, the recovered clock is available in the core fabric to capture the receiver data from the `rx_dataout` port. The x1 single and x4 bonded channel configurations are supported in this mode.

Figure 3. Transceiver Configuration in Deterministic Latency Mode



Note to Figure 3:

(1) The TX and RX phase compensation FIFOs are configured to register mode.

Table 2 lists the transceiver configurations for the supported CPRI and OBSAI RP3-01 data rates.

Table 2. FPGA Fabric–Transceiver Interface Clock Rates for Supported Line Rates and Channel Widths

Protocol	Line Rate (Mbps)	Stratix IV, HardCopy IV, Arria II GZ Interface Clock Rates (MHz)			Arria II GX Interface Clock Rates (MHz)		Cyclone IV Interface Clock Rates (MHz)	
		Channel Width *(1)*			Channel Width *(1)*		Channel Width *(1)*	
		8/10	16/20 *(2)*	32/40 *(3)*	8/10	16/20 *(2)*	8/10	16/20 *(4)*
CPRI	614.4	61.44	30.72 *(4)*	—	61.44	30.72 *(4)*	61.44	30.72
	1228.8	122.88	61.44	30.72	122.88	61.44	122.88	61.44
	2457.6	245.76	122.88	61.44	246.76 *(6)*	122.88	—	122.88
	3072	307.2	153.6	76.8	307.2	153.6	—	153.6
	4915.2	—	245.76 *(5)*	122.88	—	245.76 *(5)*	—	—
	6144	—	307.2 *(5)*	153.6	—	307.2 *(5)*	—	—
OBSAI RP3	768	76.8	38.4	—	76.8	38.4 *(4)*	76.8	38.4
	1536	153.6	76.8	38.4 *(4)*	153.6	76.8	153.6	76.8
	3072	307.2	153.6	76.8	307.2	153.6	—	153.6
	6144	—	307.2 *(5)*	153.6	—	307.2 *(5)*	—	—

Notes to Table 2:

(1) The 8/16/32 bit channel widths are supported with 8B/10B encoder/decoder and the 10/20/40 bits without 8B1/0B encoder/decoder. The ALTGX megafunction automatically enables the 8B/10B encoder/decoder according to the channel width selection.

(2) Supported in double-width mode or with the byte serializer/deserializer block.

(3) Supported in double-width mode and with the byte serializer/deserializer block.

(4) Supported with the byte serializer/deserializer block only.

(5) Supported in double-width mode only.

(6) Supported only for Arria II GX devices in the I3 speed grade.

Transceiver Channel Instantiation

You can use the MegaWizard™ Plug-In Manager in the Quartus® II software to create a transceiver configuration which implements the CPRI or OBSAI RP3-01 interface with the deterministic latency features using the ALTGX megafunction.

In this example using the Stratix IV GX device, an RE slave port (transmit and receive) is implemented with the CPRI link running at 6.144 Gbps and using the 8B/10B encoder and decoder blocks.

For more information about using the MegaWizard Plug-In Manager, refer to the *Megafunction Overview User Guide*.

Configuring ALTGX Megafunction for CPRI Implementation

Navigate through the **MegaWizard Plug-In Manager** and specify the necessary options and settings. Table 3 lists the specific values for CPRI implementation with the deterministic latency feature.

Table 3. Specific ALTGX Megafunction Options for CPRI Implementation with Deterministic Latency

Option	Settings
Which megafunction would you like to customize	Expand **I/O** and select **ALTGX**.
Which protocol will you be using?	Select **Deterministic Latency**.
Which subprotocol will you be using?	Select **X1**.
What is the operation mode?	Select **Receiver and Transmitter**.
What is the number of channels?	Select **1**.
What is the deserializer block width?	Select **Double (valid data rates: > 1.000 Gbps)**.
What is the channel width?	Select **32** bits.
What is the effective data rate?	Type `6144`. *(1)*
What is the input clock frequency?	Select **153.6** MHz.
Enable PLL phase frequency detector(PFD) feedback to compensate latency uncertainty in Tx dataout and Tx clkout paths relative to the reference clock *(2)*	Turn on. *(3)*
Enable Tx Phase Comp FIFO in register mode	Turn on. *(4)*
Use manual word alignment mode	Select this option.

Notes to Table 3:

(1) For information about supported transceiver configurations at other data rates for the CPRI and OBSAI RP3-01 interfaces, refer to Table 2 on page 5.

(2) To select this option, the input clock frequency provided to `pll_inclk` must be the same as the transmitter interface clock frequency on the `tx_clkout` port. In this example, the `pll_inclk` frequency of 153.6 MHz is the same as the `tx_clkout` frequency, as listed in Table 2 on page 5.

(3) The PLL PFD feedback is not supported if you use the ATX PLL in the Stratix IV and HardCopy IV devices.

(4) If this option is turned off, the FIFO contributes one or two clock cycles of latency uncertainty.

For more information about the options and settings of the ALTGX megafunction in the MegaWizard Plug-In Manager, refer to the *ALTGX Transceiver Setup Guide for Stratix IV Devices*.

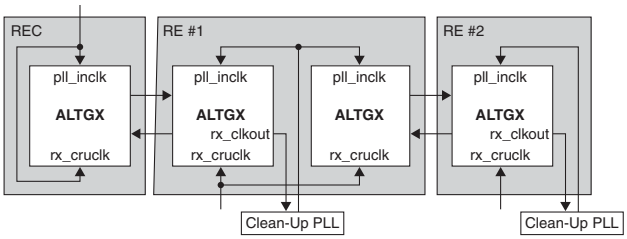
For information about design considerations when implementing auto-rate negotiation, refer to "Design Considerations for Auto-Rate Negotiation" on page 13.

Input Reference Clocks and Transmit Side Clock Generation

This section describes the input reference clock connections and schemes for the transmit side clock generation. The clocking requirements are different for port implementations in base station components and remote radio heads. For example, the CPRI specification requires any RE to reuse a transmit clock traceable to the REC on its master ports—which can be the receiver recovered clock from the RE slave port.

Figure 4 shows the methods of implementing input reference clocking for CPRI port implementations in REC and REs (in single-hop and multi-hop connections).

Figure 4. Input Reference Clocking in REC and REs ***(Note 1)***



Note to Figure 4:

(1) The input clock ports shown in the diagram are applicable for Stratix IV, HardCopy IV, and Arria II devices only.

If you configure the Cyclone IV GX device in the Deterministic Latency mode, the device provides an input reference clock port for each transmitter PLL and multipurpose PLL (MPLL) that clocks the CDR.

For the REC, use a common input reference clock source for the transmitter PLL and the CDR. In any RE module, before feeding the receiver recovered clock from the slave port into the transmitter PLL input reference clocks of the slave port and master port (in a multi-hop connection), send it to an external clean-up PLL. You need the external clean-up PLL to reduce the phase noise of the receiver recovered clock.

Use the following guidelines when selecting the appropriate external clean-up PLL for port implementation in the RE:

- Choose an external clean-up PLL device with a sufficient input and output frequency range to handle auto-rate negotiation scenarios in your application.
- Ensure that the output clock from the clean-up PLL complies with the TX REFCLK phase noise requirement, as specified in the respective device family data sheets. An example of a suitable clean-up PLL device for such implementation is the CDCL6010 from Texas Instruments.

Use the following Synopsis Design Contraints (SDC) commands if you enable the PLL PFD feedback feature for the RE implementation with an external clean-up PLL:

- On the output clock pin to the external clean-up PLL, if the RX recovered clock feeds the external clean-up PLL with no clock multiplication or division:

```
create_generated_clock -divide_by 1 -multiply_by 1 -source <RX PCS recovered clock> <clkout pin>
```

- On the input clock pin from the external clean-up PLL, if there is no clock multiplication or division from the external clean-up PLL:

```
create_generated_clock -source <clkout pin> <clkin pin>

set_clock_latency -source -early <min board + external PLL delay> <clkin pin>

set_clock_latency -source -late <max board + external PLL delay> <clkin pin>
```

Use any of the supported input reference clocking methods to the transmitter PLL and the CDR.

For more information about the supported input reference clocking methods in each device family, refer to the following documents:

- The *Transceiver Clocking in Stratix IV Devices* chapter in the *Stratix IV Device Handbook*
- The *Transceiver Architecture in HardCopy IV Devices* chapter in the *HardCopy IV Device Handbook*
- The *Transceiver Clocking in Arria II Devices* chapter in the *Arria II Device Handbook*
- The *Cyclone IV Transceiver Architecture* chapter in the *Cyclone IV Device Handbook*

The transmitter PLL utilization is dependent on PLL PFD feedback usage. If you enable the feedback function, you need a PLL for each transmitter channel to compensate the latency uncertainty in each channel.

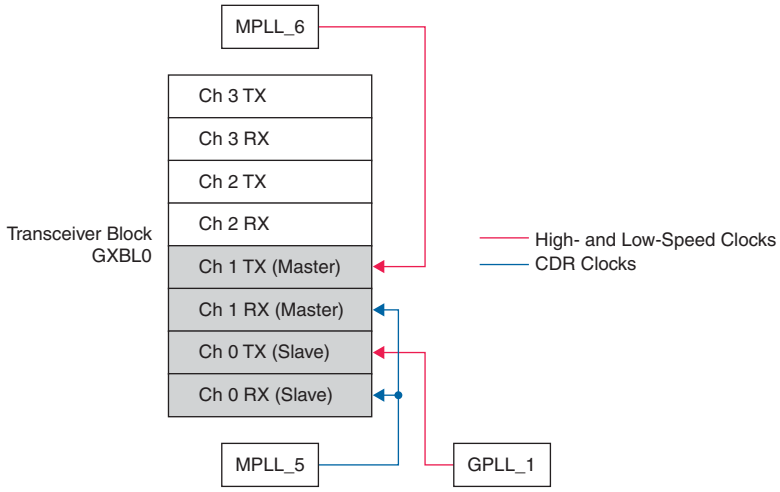
For example, in a multi-hop RE implementation, in which you enable the PLL PFD feedback, you require two PLLs:

- For the Stratix IV, HardCopy IV, or Arria II devices, you need a transceiver block and both CMU PLLs of the transceiver block.
- For the Cyclone IV devices, you need a transceiver block and three left-side PLLs, as shown in Figure 5.

Additionally, in x4 channel configurations, four transmit channels can be bonded together to share a PLL with a common latency. All of the bonded channels must run at the same data rate, which may limit auto-rate negotiation options.

Figure 5 shows an example of a transmit side clock generation and CDR clocking for a multi-hop RE module using the Cyclone IV GX device with PLL PFD feedback enabled.

Figure 5. Transmit Side Clock Generation and CDR Clocking for Multi-Hop RE *(Note 1)*



Note to Figure 5:

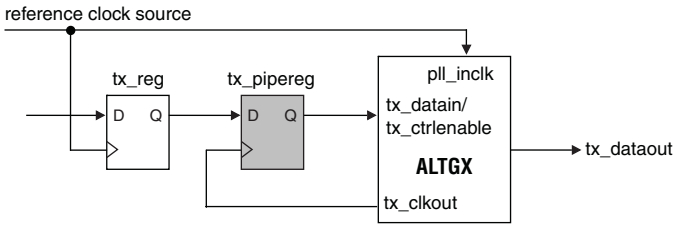
(1) This example assumes that the channels that implement slave and master ports are running at the same data rate.

Interface Clocking

The core fabric–transceiver interface clocking requirements depend on the PLL PFD feedback usage and the core fabric–transceiver interface clock frequency.

The PLL PFD feedback enables the transmitter channel datapath latency to be fixed relative to the input reference clock on the `pll_inclk` port by ensuring a deterministic path between clocks from the `tx_clkout` and `pll_inclk` ports. Use the clock from the `pll_inclk` port to clock core registers that are sending data to the transmitter channel, as shown in Figure 6.

Figure 6. Core Fabric–Transmitter Interface Clocking with the PLL PFD Feedback *(Note 1)*



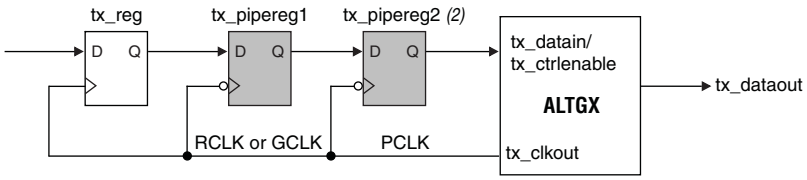
Note to Figure 6:

(1) Registers in the shaded block are optionally used to achieve timing closure.

Without the PLL PFD feedback, the transmitter channel datapath latency is fixed relative to the interface clock on the `tx_clkout` port. Use the clock from the `tx_clkout` port to clock core registers sending data to the transmitter channel, as shown in Figure 7.

Without the PLL PFD feedback, delay uncertainty exists between clocks from the `tx_clkout` and `pll_inclk` ports.

Figure 7. Core Fabric–Transmitter Interface Clocking without the PLL PFD Feedback *(Note 1)*



Notes to Figure 7:

(1) Registers in the shaded block are optionally used to achieve timing closure.

(2) Additional timing constraint is required when the `tx_pipereg2` registers are used.

Depending on the implementation, there may be delay uncertainty in transferring data between the receiver and transmitter clock domains. The receiver recovered clock may have an unknown phase relationship with the transmitter clock, resulting in a delay uncertainty that may exceed the required accuracy for the round-trip delay measurement limit of the interface.

For example, delay uncertainty occurs in a CPRI REC port implementation (without PLL PFD feedback) between clocks from the rx_clkout and tx_clkout ports. The delay uncertainty also occurs in the CPRI RE slave port implementation (without PLL PFD feedback) between clocks from the pll_inclk (derived from rx_clkout through the external clean-up PLL) and tx_clkout ports. To overcome this delay uncertainty, create an additional logic in the core fabric to implement a synchronization buffer with a phase detector to determine the phase difference. Include the measured delay from the phase difference into the total roundtrip latency computation. For information about the phase detector implementation, refer to "Phase Detector Logic" on page 11.

In certain implementations at high core fabric–transceiver interface clock frequencies, you may not achieve timing closure when you are interfacing core registers to the transmitter and receiver channels.

Use the following methods only if you are not able to achieve timing when interfacing core registers to the transmitter and receiver channels because each stage of the pipeline registers adds a latency of one clock cycle to the transmit datapath.

To comply with the core fabric–transceiver interface timing requirement, use the methods described for these scenarios:

- "Transmitter Channel with PLL PFD Feedback"
- "Transmitter Channel without PLL PFD Feedback"
- "Receiver Channel"

For more information about the methods described in the "Transmitter Channel without PLL PFD Feedback" and the "Receiver Channel" scenarios, refer to the *Achieving Timing Closure in Basic (PMA Direct) Functional Mode* application note.

Transmitter Channel with PLL PFD Feedback

Pipeline the data with intermediate registers (tx_pipereg, as shown in Figure 6) using the clock from the tx_clkout port.

Include the following assignment into the Quartus II Settings File (**.qsf**), which assigns non-global routing to the clock from the tx_clkout port:

```
set_instance_assignment -name GLOBAL_SIGNAL OFF -from
gxb_rec_fb*tx_clkout_int_wire[0] -to tx_pipereg*
```

In the assignment, gxb_rec_fb is the transceiver instance that provides the tx_clkout port, and tx_pipereg are the registers used to pipeline the data to the transmitter channel.

Transmitter Channel without PLL PFD Feedback

Pipeline the data with negative-edge triggered intermediate registers (tx_pipereg1 and tx_pipereg2 in Figure 7) using the clock from tx_clkout port.

Include the following constraint in the Synopsys Design Constraints File (**.sdc**), which adjusts the setup requirement between the pipeline registers to transmitter channel:

```
set_multicycle_path -setup -from [get_registers tx_pipereg2*] 2
```

The tx_pipereg2 registers are the last stage of the pipeline registers interfacing to the transmitter channel.

For more information about the methods described in this scenario, refer to the *Achieving Timing Closure in Basic (PMA Direct) Functional Mode* application note.

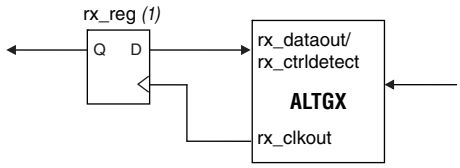
Receiver Channel

Include the following command in the **.sdc**, which adjusts the setup requirement between the receiver channel to capture registers, as shown in Figure 8:

```
set_multicycle_path -setup -from [get_registers rx_reg*] 0
```

The rx_reg registers are the registers that are used to capture data from the receiver channel.

Figure 8. Core Fabric–Receiver Interface Clocking



Note to Figure 8:

(1) To achieve timing closure, additional timing constraint is optional.

For more information about the method described in this scenario, refer to the *Achieving Timing Closure in Basic (PMA Direct) Functional Mode* application note.

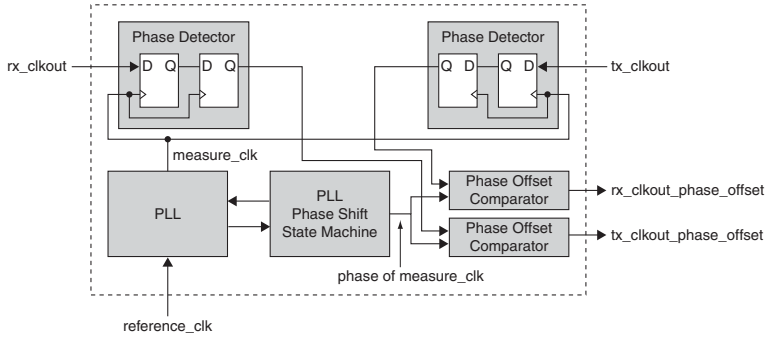
Phase Detector Logic

This section provides an example of a phase detector logic implementation to measure the phase difference between the clock domains. Use the measured phase difference to calculate the total latency of the interface datapath.

The phase detection design in this example takes advantage of the PLL dynamic phase shifting feature that is supported in the Stratix IV, HardCopy IV, Arria II, and Cyclone IV devices. The logic in this design continuously adjusts the phase of a measurement clock to determine the relative phase offsets between the desired clock domains.

Figure 9 shows a block diagram of the phase detection design that measures the phase difference between two clock domains—for example, between clocks from `rx_clkout` and `tx_clkout` if you use the design in an RE slave port without PLL PFD feedback.

Figure 9. Block Diagram of an Example Phase Detection Design



The following list describes the implementation of each block in Figure 9:

- **PLL**—The PLL (a core fabric PLL) takes a reference clock and generates a measurement clock with dynamic phase shift capability. The required phase shift step resolution depends on the system requirements. A smaller phase shift step resolution provides a higher level of delay accuracy but requires longer duration to step through all the steps in a clock period.
- **Phase Detector**—The phase detector comprises a set of pipeline registers that samples the target clock with the measurement clock from the PLL. You require a phase detector for each clock domain that requires measurement.
- **PLL Phase Shift State Machine**—The state machine controls the continuous phase increments of the measurement clock from the PLL and outputs the phase of the measurement clock.
- **Phase Offset Comparator**—The comparator finds and reports the phase offsets that match the first rising edge of a target clock from the phase detector output and the measurement clock from the PLL. You require a comparator for each clock domain that requires measurement.

With the outputs from the phase detection logic, you can compute the latency between two clock domains. Using the example illustrated in Figure 9, the latency between clocks from `rx_clkout` and `tx_clkout` is computed as follows:

> Latency between clocks from `rx_clkout` and `tx_clkout` = |phase offsets for `rx_clkout` – phase offsets for `tx_clkout`| × phase shift step resolution

For more information about implementing dynamic phase shifting, refer to the following documents:

- Stratix IV, HardCopy IV, and Arria II devices—The *PLL Dynamic Phase Shifting in the Quartus II Software* section in the *AN 454: Implementing PLL Reconfiguration in Stratix III and Stratix IV Devices* application note.
- Cyclone IV devices—The *Implementing PLL Dynamic Phase Shifting in Quartus II Software* section in the *Implementing PLL Reconfiguration in Cyclone III Devices* application note.

To facilitate data transfer between the clock domains, create additional logic—for example, a synchronization buffer. Include the additional latencies from the data transfer logics to the total latency calculation.

Design Considerations for Auto-Rate Negotiation

If you are implementing auto-rate negotiation capability for the CPRI or OBSAI RP3-01 interfaces, use the following guidelines:

- Configure the transceiver with the highest data rate intended for the device and use the supported dynamic reconfiguration features to switch to lower data rates.
- Perform timing closure with the transceiver channel configuration at the highest data rate intended for the port implementation.
- When using the PLL PFD feedback, ensure that the clocks from the `pll_inclk` and `tx_clkout` ports have the same frequency at each reconfigured data rate.
- Take advantage of the reconfiguration features on the TX local divider and TX PLL switching in the Stratix IV, HardCopy IV, and Arria II devices to enable the implementation of up to four transmitter channels in a transceiver block with independent data rate reconfiguration capability.
- Take advantage of the reconfiguration feature on the RX local divider in the Cyclone IV device to enable the implementation of up to four receiver channels in a transceiver block with independent data rate reconfiguration capability that switches between two data rates (in multiples of two).
- Use bonded (x4) channel configuration if you require the PLL PFD feedback for all four transmitter channels. Create oversampling logics in the core fabric for the auto-rate negotiation on the transmit side.

For more information about dynamic reconfiguration implementation when using the transceivers, refer to the following documents:

- The *Dynamic Reconfiguration in Stratix IV Devices* chapter in the *Stratix IV Device Handbook*
- The *HardCopy IV GX Dynamic Reconfiguration* chapter in the *HardCopy IV Device* Handbook
- The *AN 558: Implementing Dynamic Reconfiguration in Arria II Devices* application note
- The *Implementing Dynamic Reconfiguration in Cyclone IV GX Devices* application note

Table 4 on page 14 and Table 5 on page 16 list the example configurations and clocking schemes in various implementation scenarios that require auto-rate negotiation for the Stratix IV, HardCopy IV, Arria II, and Cyclone IV devices.

Table 4. Auto-Rate Negotiation Implementation Scenarios using the Stratix IV, HardCopy IV, and Arria II Devices (Part 1 of 2)

Scenarios	Channel Utilization	Reconfiguration Option	Details
Dedicated CMU PLL per channel	Up to two independent x1 duplex channels in a transceiver block, with each channel using a dedicated CMU PLL	**Channel and CMU PLL reconfiguration**	Perform negotiation to the desired data rate by reconfiguring the specific receiver channel and the CMU PLL (affects the transmitter only). Ensure that the input reference clock frequency is the same as the clock from the `tx_clkout` port at each reconfigured data rate. This option supports the PLL PFD feedback for each transmitter channel.
Shared CMU PLL—without the PLL PFD feedback path support	Up to four independent x1 duplex channels in a transceiver block, with the channels sharing one or two CMUs	**Channel and CMU PLL reconfiguration**	Perform negotiation to the desired data rate by reconfiguring the specific receiver channel.
		Data rate division in TX	Perform negotiation to related data rates (in multiples of /1, /2, or /4 of each other) by reconfiguring the TX local clock divider of the specific transmitter channel. For example: ■ From 4915.2 Mbps to 2457.6 Mbps, or 1228.8 Mbps ■ From 6144 Mbps to 3072 Mbps Enable the channels to share the same CMU PLL and perform negotiation independently without affecting each other while listening to the same CMU PLL.
		Channel reconfiguration with TX PLL select	Perform negotiation to the unrelated data rates (not in multiples of /1, /2, or /4 of each other) by reconfiguring the specific transmitter channel to select clocks from another CMU PLL. For example: ■ From 6144 Mbps to 4915.2 Mbps ■ From 4915.2 Mbps to 3072 Mbps This option uses two CMUs—one with the initial line rate clock settings and the other with the negotiated lower line rate clock settings. Reconfiguration at the specific transmitter channel does not affect the other channels that listen to either one of the CMU PLLs. Use with the **Data rate division in TX reconfiguration** option for greater negotiation flexibility. For example: ■ From 6144 Mbps to 4915.2 Mbps, then to 3072 Mbps ■ From 4915.2 Mbps to 3072 Mbps, then to 2457.6 Mbps

Table 4. Auto-Rate Negotiation Implementation Scenarios using the Stratix IV, HardCopy IV, and Arria II Devices (Part 2 of 2)

Scenarios	Channel Utilization	Reconfiguration Option	Details
Shared CMU PLL—with the PLL PFD feedback support	Four duplex channels bundled in (x4) bonded mode, with all channels sharing the CMU0 PLL	**Channel and CMU PLL reconfiguration**	Perform negotiation to the desired line rate by reconfiguring the specific receiver channel. Do not implement oversampling logic on the receiver datapath.
		Implement oversampling in your user logic	Implement variable oversampling (sends the same bit multiple times) in the user logic at each transmitter path with the CMU0 PLL clock set at the highest data rate intended for the device. Implement 8B/10B encoding in the user logic, which selects 10, 20, or 40 bits channel width to bypass the 8B/10B encoder in transceiver. Perform negotiation to the data rates that are multiples of each other (/2, /4, or /8) by enabling the appropriate oversampling path in the user logic. Negotiation at a specific transmitter channel does not affect other channels in the bundle. This option supports the PLL PFD feedback path for each transmitter, compensating the transmitter uncertainty in the four bonded channels at the same time to the CMU0 PLL.

Table 5. Auto-Rate Negotiation Implementation Scenarios using Cyclone IV Devices

Scenarios	Channel Utilization	Reconfiguration Option	Details
Dedicated PLLs per channel	Up to two independent x1 duplex channels in a transceiver block, using two PLLs for each channel (one PLL each for the transmitter and receiver) with the PLL PFD feedback, and one PLL for each channel without the PLL PFD feedback (the PLL is shared by the transmitter and receiver)	**PLL reconfiguration**	Perform negotiation to the desired data rate by reconfiguring the specific PLL that clocks the target transmitter or receiver channel, or both, when transmitter and receiver listens to the same PLL. This option supports PLL PFD feedback for each transmitter channel.
Shared PLL—with the PLL PFD feedback support	Four duplex channels bundled in (x4) bonded mode, with all the transmitter channels sharing one PLL and receiver channels listening to another PLL	**RX local divider reconfiguration**	Perform negotiation to data rates that are in multiples of /2 of each other by reconfiguring the RX local divider at the specific receiver channel. For example: ■ From 2457.6 Mbps to 1228.8 Mbps Reconfiguration at the specific receiver channel does not affect the other receiver channels in the bundle. Do not implement oversampling logic on the receiver datapath.
		Implement oversampling in your user logic	Implement variable oversampling (sending the same bit multiple times) in the user logic at each transmitter path with the PLL clock settings at the highest line rate intended for the device. Implement 8B/10B encoding in the user logic, which selects 10/20 bits channel width to bypass the 8B/10B encoder in the transceiver. Perform negotiation to the line rates that are in multiples of /2 of each other by enabling the appropriate oversampling path in the user logic. Negotiation at a specific transmitter does not affect the other channels in the bundle. This option supports the PLL PFD feedback path for each transmitter, compensating transmitter uncertainty in the four bonded channels at the same time to the PLL.

Document Revision History

Table 6 shows the revision history for this document.

Table 6. Document Revision History

Date	Version	Changes
December 2016	2.1	Updated Table 2.
July 2012	2.0	■ Added Arria II GZ support. ■ Added information about timing constraints for PLL PFD feedback. ■ Updated links to external documents. ■ Updated Figure 3.
July 2010	1.0	Initial release.