LVDS SERDES Intel® FPGA IP User Guide

Intel® Arria® 10 and Intel® Cyclone® 10 GX Devices

Updated for Intel® Quartus® Prime Design Suite: **21.1**

IP Version: **20.0.0**

Subscribe

Send Feedback

ug_altera_lvds | 2021.03.29

Latest document on the web: **PDF** | **HTML**

Contents

LVDS SERDES Intel® FPGA IP User Guide: Intel® Arria® 10 and Intel® Cyclone® 10 GX Devices

The LVDS SERDES IP core configures the serializer/deserializer (SERDES) and dynamic phase alignment (DPA) blocks. The IP core also supports LVDS channel placements, legality checks, and LVDS channel-related rule checks.

The LVDS SERDES IP core is available for Intel Arria® 10 and Intel Cyclone® 10 GX devices only. If you are migrating designs from Stratix® V, Arria V, or Cyclone V devices, you must migrate the ALTLVDS_TX and ALTLVDS_RX IP cores.

Related Information

- Migrating Your ALTLVDS_TX and ALTLVDS_RX IP Cores on page 31
- LVDS SERDES Transmitter/Receiver IP Cores User Guide
  Provides more information about the ALTLVDS_TX and ALTLVDS_RX IP cores.
- High-Speed I/O Specifications, Intel Arria 10 Device Datasheet
- High-Speed I/O Specifications, Intel Cyclone 10 GX Device Datasheet
- Introduction to Intel FPGA IP Cores
  Provides general information about all Intel FPGA IP cores, including parameterizing, generating, upgrading, and simulating IP cores.
- Creating Version-Independent IP and Qsys Simulation Scripts
  Create simulation scripts that do not require manual updates for software or IP version upgrades.
- Project Management Best Practices
  Guidelines for efficient management and portability of your project and IP files.
- LVDS SERDES Intel FPGA IP User Guide Archives on page 40
  Provides a list of user guides for previous versions of the LVDS SERDES Intel FPGA IP.

Release Information

Intel FPGA IP versions match the Intel Quartus® Prime Design Suite software versions until v19.1. Starting in Intel Quartus Prime Design Suite software version 19.2, Intel FPGA IP has a new versioning scheme.

The Intel FPGA IP version (X.Y.Z) number can change with each Intel Quartus Prime software version. A change in:

- X indicates a major revision of the IP. If you update the Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

Table 1. LVDS SERDES Intel FPGA IP Core Current Release Information

Item	Description
IP version	20.0.0 (1)
Intel Quartus Prime	21.1
Release Date	2021.03.29

LVDS SERDES IP Core Features

The LVDS SERDES IP core includes features for the LVDS receiver and transmitter. You can use the Intel Quartus Prime parameter editor to configure the LVDS SERDES IP core.

Among the features of the LVDS SERDES IP core:

- Parameterizable data channel widths
- Parameterizable SERDES factors
- Registered input and output ports
- PLL control signals
- Non-DPA mode
- DPA mode
- Soft clock data recovery (CDR) mode

Related Information

- UI/O and High Speed I/O in Intel Arria 10 Devices, Intel Arria 10 Core Fabric and General Purpose I/Os Handbook
  Provides more information about using the LVDS SERDES blocks in Intel Arria 10 devices.
- I/O and High Speed I/O in Intel Cyclone 10 GX Devices, Intel Cyclone 10 GX Core Fabric and General Purpose I/Os Handbook
  Provides more information about using the LVDS SERDES blocks in Intel Cyclone 10 GX devices.

LVDS SERDES IP Core Functional Modes

The LVDS SERDES IP core can function in transmitter or receiver modes.

(1) For Intel Arria 10 and Intel Cyclone 10 GX devices, you do not need to recompile the LVDS SERDES IP if you are updating to version 20.0.0 from the previous version.

Note: Place all RX channels in one I/O bank. Each I/O bank supports up to 24 channels.

Table 2. Functional Modes of the LVDS SERDES IP Core

All functional modes in this table support SERDES factors of 3 to 10.

Functional Mode	Description
Transmitter (**TX**)	In the transmitter mode, the SERDES block acts as a serializer. A PLL generates the following signals: • `fast_clock` • `load_enable`
Non-DPA Receiver (**RX Non-DPA**)	In the RX non-DPA mode, The SERDES block acts as a deserializer that bypasses the DPA and DPA-FIFO. A PLL generates the `fast_clock` signal. Because the incoming data is captured at the bitslip with the `fast_clock` signal, you must ensure the correct clock–data alignment.
DPA-FIFO Receiver (**RX DPA-FIFO**)	In the RX DPA-FIFO mode, the SERDES block acts as a deserializer that uses the DPA block. The DPA block uses a set of eight DPA clocks to select the optimal phase for sampling data. These DPA clocks run at the `fast_clock` frequency with each clock phase-shifted 45° apart. The DPA-FIFO, a circular buffer, samples the incoming data with the selected DPA clock and forwards the data to LVDS clock domain. The bitslip circuitry then samples the data and inserts latencies to realign the data to match the desired word boundary of the deserialized data.
Soft-CDR Receiver (**RX Soft-CDR)**	In the RX soft-CDR mode, the IP core forwards the optimal DPA clock (DPACLK) into the LVDS clock domain as the `fast_clock` signal. The IP core forwards the `rx_divfwdclk`, produced by the local clock generator, to the core through a PCLK network. Because you must place RX interfaces in one I/O bank and each bank has only 12 PCLK resources, there are only 12 soft-CDR channels available. To find out which pin pairs can support soft-CDR channels in each bank, refer to the device pin out file. In the device pin out file, the "Dedicated Tx/Rx Channel" column lists the available LVDS pin pairs in a LVDS<*bank number*>_<*pin pair*><*p or n*> format. If the value of <*pin pair*> is an even number, the pin pair supports soft-CDR mode.

Related Information

- Intel Arria 10 Pin-Out Files, Documentation: Pin-Out Files for Intel FPGAs
- Intel Cyclone 10 GX Pin-Out Files, Documentation: Pin-Out Files for Intel FPGAs
- Transmitter Blocks in Intel Arria 10 Devices, Intel Arria 10 Core Fabric and General Purpose I/Os Handbook
- Transmitter Blocks in Intel Cyclone 10 GX Devices, Intel Cyclone 10 GX Core Fabric and General Purpose I/Os Handbook
- Receiver Modes in Intel Arria 10 Devices, Intel Arria 10 Core Fabric and General Purpose I/Os Handbook
- Receiver Modes in Intel Cyclone 10 GX Devices, Intel Cyclone 10 GX Core Fabric and General Purpose I/Os Handbook

LVDS SERDES IP Core Functional Description

You can configure each LVDS SERDES IP core channel as a receiver or a transmitter for a single differential I/O.

Each LVDS SERDES IP core channel contains a SERDES, a bitslip block, DPA circuitry for all modes, a high-speed clock tree (LVDS clock tree) and forwarded clock signal for soft-CDR mode. Therefore, an *n*-channel LVDS interface contains *n*-serdes_dpa blocks.

The I/O PLLs drive the LVDS clock tree, providing clocking signals to the LVDS SERDES IP core channel in the I/O bank.

Figure 1. LVDS SERDES Channel Diagram

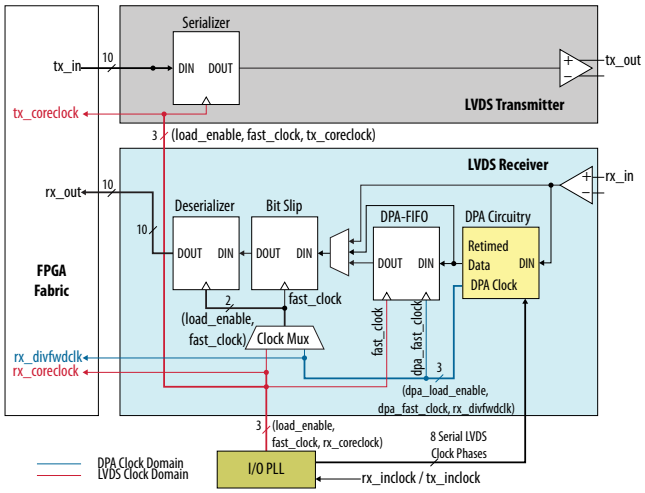


Table 3. LVDS SERDES IP Core Channel Paths and Functional Units

This table lists the paths and seven functional units in each LVDS SERDES IP core channel.

Path	Block	Mode	Clock Domain
TX Data Path	Serializer	TX	LVDS
RX Data Path	DPA	• DPA-FIFO • Soft-CDR	DPA
	DPA FIFO	DPA-FIFO	LVDS–DPA domain crossing
	• Bitslip • Deserializer	• Non-DPA • DPA-FIFO	LVDS
		Soft CDR	DPA
Clock Generation and Multiplexers	Local Clock Generator	Soft-CDR	Generates PCLK and `load_enable` in these modes
	SERDES Clock Multiplexers	All	Selects LVDS clock sources for all modes

Related Information

- Differential Transmitter in Intel Arria 10 Devices, Intel Arria 10 Core Fabric and General Purpose I/Os Handbook
- Differential Transmitter in Intel Cyclone 10 GX Devices, Intel Cyclone 10 GX Core Fabric and General Purpose I/Os Handbook
- Differential Receiver in Intel Arria 10 Devices, Intel Arria 10 Core Fabric and General Purpose I/Os Handbook
- Differential Receiver in Intel Cyclone 10 GX Devices, Intel Cyclone 10 GX Core Fabric and General Purpose I/Os Handbook

Serializer

The serializer consists of two sets of registers.

The first set of registers captures the parallel data from the core using the LVDS fast clock. The `load_enable` clock is provided alongside the LVDS fast clock, to enable these capture registers once in each `coreclock` period.

After the data is captured, it is loaded into a shift register that shifts the LSB towards the MSB at one bit per fast clock cycle. The MSB of the shift register feeds the LVDS output buffer. Therefore, higher order bits precede lower order bits in the output bitstream.

Figure 2. LVDS x8 Serializer Waveform

This figure shows the waveform specific to serialization factor of eight.

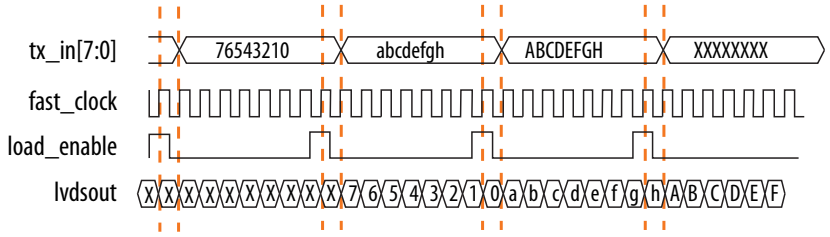


Table 4. LVDS Serializer Signals

Signal	Description
`tx_in[7:0]`	Data for serialization (Supported serialization factors: 3–10)
`fast_clock`	Clock for the transmitter
`load_enable`	Enable signal for serialization
`lvdsout`	LVDS output data stream from the LVDS SERDES IP core channel

Related Information

- Transmitter Blocks in Intel Arria 10 Devices, Intel Arria 10 Core Fabric and General Purpose I/Os Handbook
- Transmitter Blocks in Intel Cyclone 10 GX Devices, Intel Cyclone 10 GX Core Fabric and General Purpose I/Os Handbook

DPA FIFO

In DPA-FIFO mode, the DPA FIFO synchronizes the re-timed data to the high-speed LVDS clock domain.

The DPA clock may shift phase during the initial lock period. To avoid data run-through condition caused by the FIFO write pointer creeping up to the read pointer, hold the FIFO in reset state until the DPA locks.

Related Information

- DPA Block, Intel Arria 10 Core Fabric and General Purpose I/Os Handbook
- DPA Block, Intel Cyclone 10 GX Core Fabric and General Purpose I/Os Handbook

- Guideline: Pin Placement for DPA-Enabled Differential Channels, Intel Arria 10 Core Fabric and General Purpose I/Os Handbook
- Guideline: Pin Placement for DPA-Enabled Differential Channels, Intel Cyclone 10 GX Core Fabric and General Purpose I/Os Handbook

Bitslip

Use bitslip circuitry to insert latencies in increments of one fast clock cycle for data word alignment.

The data slips one bit for every pulse of the `rx_bitslip_ctrl` signal. Because it takes at least two core clock cycles to clear the undefined data, wait at least four core clock cycles before checking if the data is aligned.

After enough bitslip signals are sent to rollover the bitslip counter, the `rx_bitslip_max` status signal is asserted after four core clock cycles to indicate that the bitslip counter rollover point has reached its maximum counter value.

Related Information

- Data Realignment Block (Bit Slip), Intel Arria 10 Core Fabric and General Purpose I/Os Handbook
- Data Realignment Block (Bit Slip), Intel Cyclone 10 GX Core Fabric and General Purpose I/Os Handbook

Deserializer

The deserializer consists of shift registers. The deserialization factor determines the depth of the shift registers. The deserializer converts a 1-bit serial data stream into a parallel data stream based on the deserialization factor.

The `load_enable` is a pulse signal with a frequency equivalent to the fast clock divided by the deserialization factor.

Figure 3. LVDS x8 Deserializer Waveform

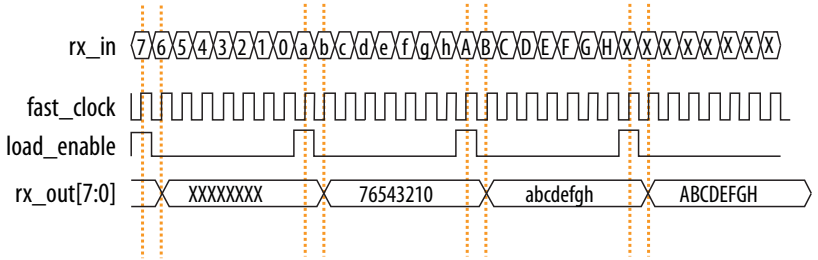


Table 5. LVDS Deserializer Signals

Signal	Description
`rx_in`	LVDS input data stream to the LVDS SERDES IP core channel
`fast_clock`	Clock for the receiver
`load_enable`	Enable signal for deserialization
`rx_out[7:0]`	Deserialized data

Related Information

- Deserializer, Intel Arria 10 Core Fabric and General Purpose I/Os Handbook
- Deserializer, Intel Cyclone 10 GX Core Fabric and General Purpose I/Os Handbook

LVDS SERDES IP Core Initialization and Reset

During device initialization, the clock reference must be stable while the PLL is locking to it to avoid corruption of the PLL output clock phase shifts. If the PLL output clock phase shifts are incorrect, data transfer between the high-speed LVDS and low-speed parallel domain can fail and causes corrupted data.

After you have initialized the IP core in DPA or non-DPA mode, you can perform word boundaries alignment using the bitslip control signal.

Initializing the LVDS SERDES IP Core in Non-DPA Mode

The PLL is operational after it achieves lock in user mode. Before transferring data using SERDES block with the LVDS SERDES IP core, ensure that the PLL is locked to the reference clock.

Intel recommends that you follow these steps to initialize the LVDS SERDES IP core in non-DPA mode:

1. During entry into user mode, assert the `pll_areset` signal for at least 10 ns.

   You can also perform this step at any time in user mode operation to reset the interface.
2. After at least 10 ns, deassert the `pll_areset` signal and monitor the `pll_locked` port.

After the PLL lock port asserts and becomes stable, the SERDES blocks are ready for operation.

After the initialization, you can proceed to align the word boundaries (bitslip).

Related Information

- Word Boundaries Alignment on page 11
- Aligning Word Boundaries on page 12

Initializing the LVDS SERDES IP Core in DPA Mode

The DPA circuit samples the incoming data and determines the optimal phase tap from the PLL to capture data at the receiver on a channel-by-channel basis. If the PLL has not locked to a stable clock source, the DPA circuit might lock prematurely to a non-ideal phase tap.

Before the PLL lock is stable, use the `rx_dpa_reset` signal to keep the DPA in reset. When the DPA has determined the optimal phase tap, the `rx_dpa_locked` signal asserts. The LVDS SERDES IP core asserts the `rx_dpa_locked` port at the initial DPA lock. If you turn on the **Enable DPA loss of lock on one change** option, the `rx_dpa_locked` port deasserts after one phase change. If you turn off this option, the `rx_dpa_locked` signal deasserts after two phase changes in the same direction.

Intel recommends that you follow these steps to initialize and reset the LVDS SERDES IP core in DPA mode:

1. During entry into user mode, assert the `pll_areset` and `rx_dpa_reset` signals. Keep the `pll_areset` signal asserted for at least 10 ns.

   You can also perform this step at any time in user mode operation to reset the interface.
2. After at least 10 ns, deassert the `pll_areset` signal and monitor the `pll_locked` port.
3. Deassert the `rx_dpa_reset` port after the `pll_locked` port becomes asserted and stable.
4. Apply the DPA training pattern and allow the DPA circuit to lock.

   If a training pattern is not available, any data with transitions is required to allow the DPA to lock. For the DPA lock time specification, refer to the related information.
5. After the `rx_dpa_locked` signal asserts, assert the `rx_fifo_reset` signal for at least one parallel clock cycle.
6. To start receiving data, deassert the `rx_fifo_reset` signal.

During normal operation, every time the DPA shifts the phase taps to track variations between the reference clock source and the data, the data transfer timing margin between clock domains is reduced.

Note: To ensure data accuracy, Intel recommends that you use the data checkers.

After the initialization, you can proceed to align the word boundaries (bitslip).

Related Information

- Resetting the DPA on page 10
- Word Boundaries Alignment on page 11
- Aligning Word Boundaries on page 12
- DPA Lock Time Specifications, Intel Arria 10 Device Datasheet
- DPA Lock Time Specifications, Intel Cyclone 10 GX Device Datasheet
- LVDS Soft-CDR/DPA Sinusoidal Jitter Tolerance Specifications, Intel Arria 10 Device Datasheet
- LVDS Soft-CDR/DPA Sinusoidal Jitter Tolerance Specifications, Intel Cyclone 10 GX Device Datasheet

Resetting the DPA

If data corruption occurs, reset the DPA circuitry.

1. Assert the `rx_dpa_reset` signal to reset the entire DPA block. After you reset the entire DPA block, the DPA must be retrained before capturing data.

You can also fix data corruption by resetting only the synchronization FIFO without resetting the DPA circuit, which means that system operation continues without having to retrain the DPA. To reset just the synchronization FIFO, assert the `rx_fifo_reset` signal.

2. After `rx_dpa_locked` asserts, the LVDS SERDES IP core is ready to capture data. The DPA finds the optimal sample location to capture each bit.

   Intel recommends that you toggle the `rx_fifo_reset` signal after `rx_dpa_locked` asserts. Toggling `rx_fifo_reset` ensures that the synchronization FIFO is set with the optimal timing to transfer data between the DPA and the high-speed LVDS clock domains.

3. Using custom logic to control the `rx_bitslip_ctrl` signal on a channel-by-channel basis, set up the word boundary.

   You can reset the bit slip circuit at any time, independent of the PLL or DPA circuit operation. To reset the bit slip circuit, use the `rx_bitslip_reset` signal.

Related Information

- Initializing the LVDS SERDES IP Core in DPA Mode on page 9
- DPA Lock Time Specifications, Intel Arria 10 Device Datasheet
- DPA Lock Time Specifications, Intel Cyclone 10 GX Device Datasheet
- LVDS Soft-CDR/DPA Sinusoidal Jitter Tolerance Specifications, Intel Arria 10 Device Datasheet
- LVDS Soft-CDR/DPA Sinusoidal Jitter Tolerance Specifications, Intel Cyclone 10 GX Device Datasheet

Word Boundaries Alignment

You can perform word boundaries alignment with or without control characters in your data stream. If there are no training patterns or control characters available in the serial bit stream to use for word alignment, Intel recommends that you use the non-DPA mode.

Aligning with Control Characters

By adding control characters in the data stream, your logic can search for a known pattern to align the word boundaries. You can compare the received data for each channel, and then pulse the `rx_bitslip_ctrl` signal as required until you receive the control character.

Note: Intel recommends that you set the bit slip rollover count to the deserialization factor, or higher. This setting allows enough depth in the bit slip circuit to roll through an entire word, if required.

Aligning without Control Characters

Without control characters in the data stream, you need a deterministic relationship between the reference clock and the data. With the deterministic relationship, you can predict the word boundary using timing simulation or laboratory measurement. You can only use deterministic relationship in non-DPA mode.

The only way to ensure a deterministic relationship on the default word position in the SERDES when the device powers up, or anytime the PLL is reset, is to have a reference clock equal to the data rate divided by the deserialization factor. This is important because the PLL locks to the rising edge of the reference clock. If you have one rising edge on the reference clock per serial word received, the deserializer always starts at the same position.

For example, if the data rate is 800 Mbps and the deserialization factor is 8, the PLL requires a 100-MHz reference clock.

Using timing simulation, or lab measurements, monitor the parallel words received and determine how many pulses of the `rx_bitslip_ctrl` are required to set your word boundaries. You can create a simple state machine to apply the required number of pulses after you enter user mode or at any time after you reset the PLL.

Note: If you are using the DPA or soft-CDR modes, the word boundary is not deterministic. The initial training of the DPA allows it to move forward or backward in phase relative to the incoming serial data. Therefore, there can be a ±1 bit of variance in the serial bit where the DPA locks initially.

Related Information

- Initializing the LVDS SERDES IP Core in Non-DPA Mode on page 9
- Initializing the LVDS SERDES IP Core in DPA Mode on page 9
- Aligning Word Boundaries on page 12

Aligning Word Boundaries

After initializing the LVDS SERDES IP core in DPA or non-DPA mode, perform these steps to align the word boundaries.

1. Assert the `rx_bitslip_reset` port for at least one parallel clock cycle, and then deassert the `rx_bitslip_reset` port.
2. Begin word alignment by applying pulses as required to the `rx_bitslip_ctrl` port.

After the word boundaries are established on each channel, the interface is ready for operation.

Related Information

- Initializing the LVDS SERDES IP Core in Non-DPA Mode on page 9
- Initializing the LVDS SERDES IP Core in DPA Mode on page 9
- Word Boundaries Alignment on page 11

LVDS SERDES IP Core Signals

Table 6. Common LVDS SERDES IP Core TX and RX Signals

Signal Name	Width	Direction	Type	Description
inclock	1	Input	Clock	PLL reference clock
pll_areset	1	Input	Reset	Active-high asynchronous reset to all blocks in LVDS SERDES IP core and PLL
pll_locked	1	Output	Control	Asserts when internal PLL locks

Table 7. LVDS SERDES IP Core RX Signals

In this table, *N* represents the LVDS interface width and the number of serial channels while *J* represents the SERDES factor of the interface.

Signal Name	Width	Direction	Type	Description
rx_in	*N*	Input	Data	LVDS serial input data
rx_bitslip_reset	*N*	Input	Reset	Asynchronous, active-high reset to the clock-data alignment circuitry (bit slip)
rx_bitslip_ctrl	*N*	Input	Control	• Positive-edge triggered increment for bit slip circuitry • Each assertion adds one bit of latency to the received bit stream
rx_dpa_hold	*N*	Input	Control	• Asynchronous, active-high signal that prevents the DPA circuitry from switching to a new clock phase on the target channel — Held high—selected channels hold their current phase setting — Held low—the DPA block on selected channels monitors the phase of the incoming data stream continuously and selects a new clock phase when needed • Applicable in DPA-FIFO and soft-CDR modes only
rx_dpa_reset	*N*	Input	Reset	• Asynchronous, active-high reset to DPA blocks • Minimum pulse width: one parallel clock period • Applicable in DPA-FIFO and soft-CDR modes only
rx_fifo_reset	*N*	Input	Reset	• Asynchronous, active-high reset to FIFO block • Minimum pulse width: one parallel clock period • Applicable in DPA-FIFO mode only
rx_out	*N***J*	Output	Data	Receiver parallel data output • DPA-FIFO and non-DPA modes—synchronous to rx_coreclock. • Soft-CDR mode—each channel has parallel data synchronous to its rx_divfwdclk
rx_bitslip_max	*N*	Output	Control	• Bit slip rollover signal • High when the next assertion of rx_bitslip_ctrl resets the serial bit latency to 0
rx_coreclock	1	Output	Clock	• Core clock for RX interfaces provided by the PLL • Not available if you use an external PLL
rx_divfwdclk	*N*	Output	Clock	The per channel and divided clock with the ideal DPA phase • This is the recovered slow clock for a given channel • Applicable in soft-CDR mode only

continued...

Signal Name	Width	Direction	Type	Description
				The rx_divfwdclk signals may not be edge-aligned with each other because each channel may have a different ideal sampling phase. Each rx_divfwdclk must drive the core logic with data from the same channel.
rx_dpa_locked	*N*	Output	Control	Asserted when the DPA block selects the ideal phase • Driven by the LVDS SERDES IP core • Asserts when the signal settles on an ideal phase for that given channel • Deasserts in one of these conditions: — The DPA moves one phase — The DPA moves two phases in the same direction • Applicable in DPA-FIFO and soft-CDR modes only Ignore all toggling of the rx_dpa_locked signal after rx_dpa_hold asserts.

Table 8. LVDS SERDES IP Core TX Signals

In this table, *N* represents the LVDS interface width and the number of serial channels while *J* represents the SERDES factor of the interface.

Signal Name	Width	Direction	Type	Description
tx_in	*N***J*	Input	Data	Parallel data from the core
tx_out	*N*	Output	Data	LVDS serial output data
tx_outclock	1	Output	Clock	• External reference clock (sent off-chip through the TX data path) • Source-synchronous with tx_out
tx_coreclock	1	Output	Clock	• Drives the core logic feeding the serializer • This signal is a feedthrough of the ext_coreclock input

Table 9. External PLL Signals for LVDS SERDES IP Core

For instructions on setting the frequencies, duty cycles, and phase shifts of the required PLL clocks for external PLL mode, refer to the **Clock Resource Summary** tab in the IP Parameter Editor.

Signal Name	Width	Direction	Type	Description
ext_fclk	1	Input	Clock	LVDS fast clock • Used for serial data transfer • Required in all modes For more information about connecting this port with the signal from the IOPLL Intel FPGA IP, refer to the related information.
ext_loaden	1	Input	Clock	LVDS load enable • Used for parallel load • Not required in RX soft-CDR mode For more information about connecting this port with the signal from the IOPLL IP core, refer to the related information.
ext_coreclock	1	Input	Clock	• Drives the core logic feeding the serializer (TX) or receiving from the deserializer (RX) • Present in RX soft-CDR mode, even though the RX core registers are clocked by rx_divfwdclk.

continued...

Signal Name	Width	Direction	Type	Description
ext_vcoph[7:0]	8	Input	Clock	• Provides the VCO clocks to the DPA circuitry for optimal phase selection • Required for RX DPA-FIFO and RX soft-CDR modes For more information about connecting this port with the signal from the IOPLL IP core, refer to the related information.
ext_pll_locked	1	Input	Data	PLL lock signal Required for RX DPA-FIFO and RX Soft-CDR modes only
ext_tx_outclock_fclk	1	Input	Clock	Phase-shifted version of fast clock Required for TX outclock phase shifts that are not multiples of 180°
ext_tx_outclock_loaden	1	Input	Clock	Phase-shifted version of load_enable Required for TX outclock phase shifts that are not multiples of 180°

Related Information

LVDS Interface with External PLL Mode on page 31

LVDS SERDES IP Core Parameter Settings

You can parameterize the LVDS SERDES IP core using the Intel Quartus Prime parameter editor.

LVDS SERDES IP Core PLL Settings

Table 10. PLL Settings Tab

Parameter	Value	Description
Use external PLL	On, Off	Turn on to use an external PLL: • The IP core does not instantiate a local PLL. • The IP core creates a series of clock connections with the "ext" prefix. Connect these ports to an externally generated PLL. • For details about how to configure the external PLL, refer to the **Clock Resource Summary** tab of the parameter editor. This option allows you to access all of the available clocks from the PLL and use advanced PLL features such as clock switchover, bandwidth presets, dynamic phase stepping, and dynamic reconfiguration. *Note:* You must turn on this option if you want to place combined LVDS transmitter and receiver interfaces in the same I/O bank.
Desired inclock frequency	—	Specifies the inclock frequency in MHz.
Actual inclock frequency	—	Displays the closest inclock frequency to the desired frequency that can source the interface.

continued...

Parameter	Value	Description
FPGA/PLL speed grade	—	Specifies the FPGA/PLL speed grade which determines the operation range of the PLL.
Enable pll_areset port	On, Off	Turn on to expose the `pll_areset` port. You can use the `pll_areset` signal to reset the entire LVDS interface.
Core clock resource type	—	Specifies onto which clock network the IP core exports an internally generated `coreclock`. *Note:* This feature will be supported in a future version of the Intel Quartus Prime software. Currently, use QSF assignments to manually specify this parameter.

Related Information

- LVDS Interface with External PLL Mode on page 31
- PLLs and Clocking for Intel Arria 10 Devices, Intel Arria 10 Core Fabric and General Purpose I/Os Handbook
  Provides more information about clocking differential transmitters and receivers in Intel Arria 10 devices.
- PLLs and Clocking for Intel Cyclone 10 GX Devices, Intel Cyclone 10 GX Core Fabric and General Purpose I/Os Handbook
  Provides more information about clocking differential transmitters and receivers in Intel Cyclone 10 GX devices.

LVDS SERDES IP Core Receiver Settings

Table 11. Receiver Settings Tab—Bitslip Settings

Parameter	Value	Description
Enable bitslip mode	On, Off	Turn on to add a bit slip block to the receiver data path and expose the `rx_bitslip_ctrl` port (one input per channel). Every assertion of the `rx_bitslip_ctrl` signal adds one bit of serial latency to the data path of the specified channel.
Enable rx_bitslip_reset port	On, Off	Turn on to expose the `rx_bitslip_reset` port (one input per channel) that you can use to reset the bit slip.
Enable rx_bitslip_max port	On, Off	Turn on to expose the `rx_bitslip_max` port (one output per channel). When asserted, the next rising edge of `rx_bitslip_ctrl` resets the latency of the bit slip to zero.
Bitslip rollover value	Deserialization factor	Specifies the maximum latency that the bit slip can inject. When the bit slip reaches the specified value, it rolls over and the `rx_bitslip_max signal` asserts. The rollover value is set automatically to the deserialization factor.

Table 12. Receiver Settings Tab—DPA Settings

Parameter	Value	Description
Enable rx_dpa_reset port	On, Off	Turn on to expose the `rx_dpa_reset` port that you can use to reset the DPA logic of each channel independently.

continued...

Parameter	Value	Description
		(Formerly known as `rx_reset`.)
Enable rx_fifo_reset port	On, Off	Turn on to use your logic to drive the `rx_fifo_reset` port to reset the DPA-FIFO block.
Enable rx_dpa_hold port	On, Off	Turn on to expose the `rx_dpa_hold` input port (one input per channel). If set high, the DPA logic in the corresponding channel does not switch sampling phases. (Formerly known as `rx_dpll_hold`.)
Enable DPA loss of lock on one change	On, Off	• On—the IP core drives the `rx_dpa_locked` signal low when the DPA changes phase selection from the initially locked position. When the DPA changes the phase selection back to the initial locked position, the IP core drives the `rx_dpa_locked` signal high. • Off—the IP core drives the `rx_dpa_locked` signal low when the DPA moves two phases in the same direction away from the initial locked position. When the DPA changes the phase selection to be within one phase or same phase as the initial locked position, the IP core drives the `rx_dpa_locked` signal high. Deassertion of `rx_dpa_locked` does not indicate that the data is invalid. Instead, it indicates that the DPA has changed phase taps to track variations between the `inclock` and `rx_in` data. Intel recommends that you use data checkers to verify data accuracy.
Enable DPA alignment only to rising edges of data	On, Off	• On—DPA logic counts only the rising edges of the incoming serial data • Off—DPA logic counts the rising and falling edges *Note:* Intel recommends that you use this port only for high jitter systems and turn it off for typical applications.
(Simulation only) Specify PPM drift on the recovered clock(s)	—	Specifies the amount of phase drift the LVDS SERDES IP core simulation model should add to the recovered `rx_divfwdclks`. *Note:* This feature will be supported in a future version of the Intel Quartus Prime software.

Table 13. Receiver Settings Tab—Non-DPA Settings

Parameter	Value	Description
Desired receiver inclock phase shift (degrees)	—	Specifies, in degrees of the LVDS fast clock, the ideal phase delay of the `inclock` with respect to transitions in the incoming serial data. For example, specifying 180° implies that the `inclock` is center aligned to the incoming data.
Actual receiver inclock phase shift (degrees)	Depends on the `fast_clock` and `inclock` frequencies. Refer to the related information.	Specifies the closest achievable receiver `inclock` phase shift to the desired receiver `inclock` phase shift.

Related Information

Receiver Input Clock Parameters Setup on page 18

Receiver Input Clock Parameters Setup

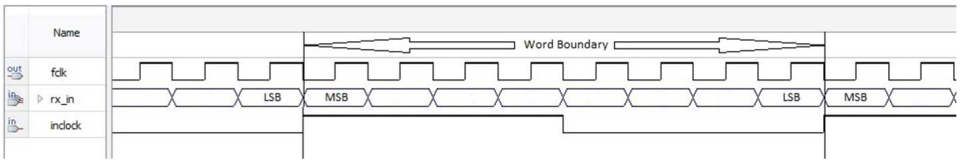
To sample the source-synchronous data using the SERDES receiver in non-DPA mode, you must specify the phase relationship between the `inclock` and the `rx_in` data.

You can specify the `inclock` to `rx_in` phase relationship value in the **Desired receiver inclock phase shift (degrees)** parameter setting. The value must be evenly divisible by 45. If the value is not divisible by 45, the actual phase shift appears in the **Actual receiver inclock phase shift (degrees)** parameter setting.

Edge-Aligned `inclock` to `rx_in`

For rising `inclock` edge-aligned to the `rx_in` data, specify 0° as the desired receiver clock phase shift. Specifying 0° phase shift sets the PLL with the required phase shift from `fast_clock` to center it at the SERDES receiver.

Figure 4. 0° Edge-Aligned `inclock` x8 Deserializer Waveform with Single Rate Clock



The phase shift you specify is relative to the `fast_clock`, which operates at the serial data rate. Use phase shift values between 0° and 360° to specify the rising edge of the `inclock` within a single bit period. If you specify phase shift values greater than 360°, the MSB location within the parallel data changes.

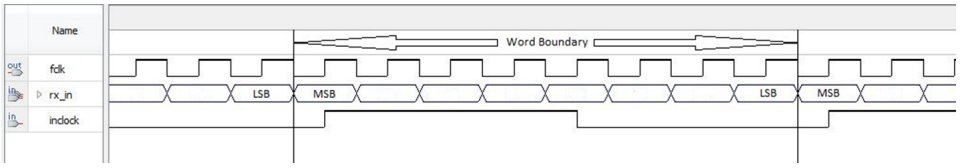
This equation determines the maximum phase shift value: (Number of `fast_clock` periods per `inclock` period x 360) – 1.

Note: By default, the MSB from the serial data is not the MSB of the parallel data. You can use bit slip to set the proper word boundary on the parallel data.

Center-Aligned `inclock` to `rx_in`

To specify a center-aligned relationship between `inclock` and `rx_in`, specify a 180° phase shift.

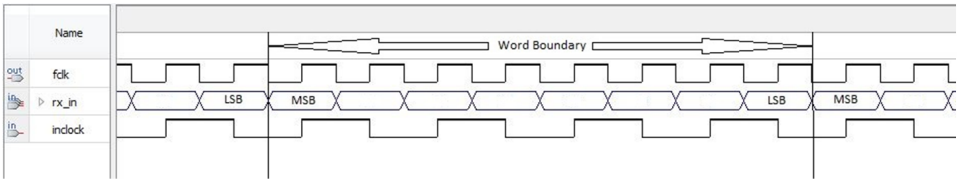
Figure 5. 180° Center-Aligned `inclock` x8 Deserializer Waveform with Single Rate Clock



The `inclock` to `rx_in` phase shift relationship you specify is independent of the `inclock` frequency.

To specify a center-aligned DDR `inclock` to `rx_in` relationship, specify a 180° phase shift.

Figure 6. 180° Center Aligned `inclock` x8 Deserializer Waveform with DDR Clock



Related Information

- LVDS SERDES IP Core Receiver Settings on page 16
- Word Boundaries Alignment on page 11

LVDS SERDES IP Core Transmitter Settings

Table 14. Transmitter Settings Tab

Parameter	Value	Description
TX core registers clock	• tx_coreclock • inclock	Selects the clock that clocks the core registers: • **tx_coreclock**—selects the `tx_coreclock` as the clock source. • **inclock**—select the PLL `refclk` as the clock source. The `refclk` frequency must be equal to the data rate divided by the serialization factor. This parameter is available only in the TX functional mode.
Enable tx_coreclock port	On, Off	Turn on to expose the `tx_coreclock` port that you can use to drive the core logic feeding the transmitter. The `tx_coreclock` signal is a feedthrough of the `ext_coreclock` input. Intel recommends that you use the `tx_coreclock` output signal if it is requested. *Note:* This option is disabled if the **Use external PLL** option in the **PLL Settings** tab is turned on. To turn the **Enable tx_coreclock port** option on or off, turn off **Use external PLL** option first. After making changes to **Enable tx_coreclock port**, you can turn **Use external PLL** back on.
Enable tx_outclock port	On, Off	Turn on to expose the `tx_outclock` port. • The `tx_outclock` port frequency depends on the setting for the **tx_outclock division factor** parameter. • The `tx_outclock` port phase depends on the **Desired tx_outclock phase shift** parameter. Turning on this parameter reduces the maximum number of channels per TX interface by one channel.
Desired tx_outclock phase shift (degrees)	Refer to related information.	Specifies the phase relationship between the `outclock` and outgoing serial data in degrees of the LVDS fast clock.
Actual tx_outclock phase shift (degrees)	Depends on `fast_clock` and `tx_outclock` frequencies. Refer to related information.	Displays the closest achievable `tx_outclock` phase shift to the desired `tx_outclock` phase shift.
Tx_outclock division factor	Depends on the serialization factor.	Specifies the ratio of the fast clock frequency to the `outclock` frequency. For example, the maximum number of serial transitions per `outclock` cycle.

Related Information

Setting the Transmitter Output Clock Parameters on page 20

Setting the Transmitter Output Clock Parameters

You can specify the relationship of `tx_outclock` to the `tx_out` data using these parameters:

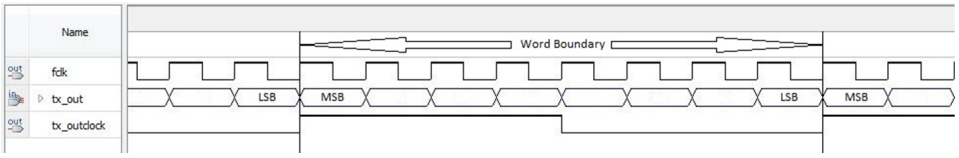
- **Desired tx_outclock phase shift (degrees)**
- **Tx_outclock division factor**

The parameters set the phase and frequency of the `tx_outclock` based on the `fast_clock`, which operates at the serial data rate. You can specify the desired `tx_outclock` phase shift relative to the `tx_out` data at 45° increments of the `fast_clock`. You can set the `tx_outclock` frequency using the available division factors from the drop-down list.

Edge-Aligned `tx_outclock` to `tx_out`

For rising `tx_outclock` edge-aligned to the MSB of the serial data on `tx_out`, specify 0° phase shift.

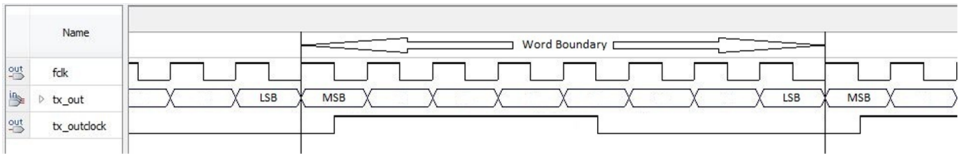
Figure 7. 0° Edge Aligned `tx_outclock` x8 Serializer Waveform with Division Factor of 8



Center-Aligned `tx_outclock` to `tx_out`

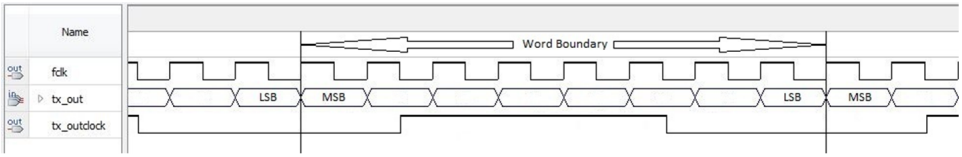
To specify center-aligned relationship between `tx_outclock` and the MSB of the serial data on `tx_out`, specify 180° phase shift.

Figure 8. 180° Center Aligned `tx_outclock` x8 Serializer Waveform with Division Factor of 8



- Phase shift values from 0° to 315° position the rising edge of `tx_outclock` within the MSB of the `tx_out` data.
- Phase shift values starting from 360° position the rising edge of `tx_outclock` in serial bits after the MSB. For example, a phase shift of 540° positions the rising edge in the center of the bit after the MSB.

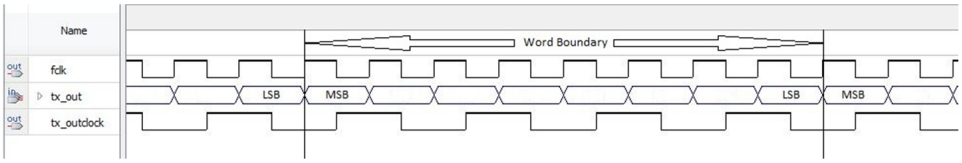
Figure 9. 540° Center Aligned `tx_outclock` x8 Serializer Waveform with Division Factor of 8



Use the **Tx_outclock division factor** drop-down list to set the `tx_outclock` frequency.

Figure 10. 180° Center Aligned tx_outclock x8 Serializer Waveform with Division Factor of 2

This figure shows a x8 serialization factor using a 180° phase shift with a `tx_outclock` division factor of 2 (DDR clock and data relationship).



Related Information

LVDS SERDES IP Core Transmitter Settings on page 19

LVDS SERDES IP Core Clock Resource Summary

The **Clock Resource Summary** tab lists the required frequencies, phase shifts, and duty cycles of the required clocks, and instructions for connections. You can refer to this tab for information about configuring and connecting an external PLL to the LVDS SERDES IP core.

LVDS SERDES IP Core General Settings

Table 15. General Settings Tab

Parameter	Value	Description
Functional mode	• TX • RX Non-DPA • RX DPA-FIFO • RX Soft-CDR	Specifies the functional mode of the interface.
Number of channels	• 1 to 72 for TX • 1 to 24 for RX Non-DPA • 1 to 24 for RX DPA-FIFO • 1 to 12 for RX Soft-CDR	Specifies the number of serial channels in the interface. • If you use a dedicated reference clock for the TX, RX non-DPA, or RX DPA-FIFO, you must use one of the channels for the `refclk` pin. Use a dedicated reference clock to reduce jitter. • If you use a transmitter output clock, you must use one of the channels for the `tx_outclock` pin. For an LVDS RX design, place the `refclk` pin on the same I/O bank as the receiver.

continued...

Parameter	Value	Description
		For an LVDS TX design: • For an interface with less than 23 channels (standalone), each interface requires a `refclk` pin on the same I/O bank. • For an interface with more than 23 channels, channels 23 to 71 can share one `refclk` input.
Data rate	150.0 to 1600.0	Specifies the data rate (in Mbps) of a single serial channel. The value is dependent on the **Functional mode** parameter settings.
SERDES factor	3, 4, 5, 6, 7, 8, 9, and 10	Specifies the serialization rate or deserialization rate for the LVDS interface.
Use backwards-compatible port names	On, Off	Turn on to use legacy top-level names that are compatible with the ALTLVDS_TX and ALTLVDS_RX IP cores.

LVDS SERDES IP Core Timing

Use the Intel Quartus Prime software from version 14.0.a10 onwards to generate the required timing constraint to perform proper timing analysis of the LVDS SERDES IP core in Intel Arria 10 and Intel Cyclone 10 GX devices.

Table 16. LVDS SERDES IP Core Timing Components

Timing Component	Description
Source Synchronous Paths	The source synchronous paths are paths where clock and data signals are passed from the transmitting devices to the receiving devices. For example: • FPGA/LVDS/TX to external receiving device transmitting • External transmitting device to FPGA/non-DPA mode/LVDS/RX receiving path
Dynamic Phase Alignment Paths	A DPA block registers the I/O capture paths in soft-CDR and DPA-FIFO modes. The DPA block dynamically chooses the best phase from the PLL VCO clocks to latch the input data.
Internal FPGA Paths	The internal FPGA paths are the paths inside the FPGA fabric: • LVDS RX hardware to core registers paths • Core registers to LVDS TX hardware paths • Others core registers to core registers path The Timing Analyzer reports the corresponding timing margins.

Table 17. LVDS SERDES Timing Constraint Files

This table lists the timing files generated by the LVDS SERDES IP core. Use these files for successful timing analysis of the LVDS SERDES IP core. You can find these files in the *<variation_name>* directory.

File Name	Description
`<variation_name>_altera_lvds_core20_<quartus_version>_<random_id>.sdc`	This **.sdc** file allows the Intel Quartus Prime Fitter to optimize timing margins with timing-driven compilation. The file also allows the Timing Analyzer to analyze the timing of your design. The IP core uses the **.sdc** for the following operations: • Creating clocks on PLL inputs • Creating generated clocks • Calling `derive_clock_uncertainty` • Creating proper multi-cycle constraints You can locate this file in the **.qip** generated during IP generation.
`sdc_util.tcl`	This **.tcl** file is a library of functions and procedures that the **.sdc** uses.

Related Information

- Source-Synchronous Timing Budget, Intel Arria 10 Core Fabric and General Purpose I/Os Handbook
- Source-Synchronous Timing Budget, Intel Cyclone 10 GX Core Fabric and General Purpose I/Os Handbook

I/O Timing Analysis

The LVDS I/O standard enables high-speed transmission of data, resulting in better overall system performance. To take advantage of fast system performance, you must analyze the timing for these high-speed signals. Timing analysis for the differential block is different from traditional synchronous timing analysis techniques.

Receiver Timing Analysis in Soft-CDR and DPA-FIFO Modes

The DPA hardware dynamically captures the received data in soft-CDR and DPA-FIFO modes. For these modes, the Timing Analyzer does not perform static I/O timing analysis.

Receiver Timing Analysis in Non-DPA Mode

In non-DPA mode, use RSKM, TCCS, and sampling window (SW) specifications for high-speed source-synchronous differential signals in the receiver data path.

To obtain accurate RSKM results in the Timing Analyzer, add this line of code to your `.sdc` to specify the RCCS value: `set ::RCCS <RCCS value in nanoseconds>`. For example, `set ::RCCS 0.0`.

Transmitter Timing Analysis

For LVDS transmitters, the Timing Analyzer provides the transmitter channel-to-channel skew (TCCS) value in the TCCS report (`report_TCCS`) in the Intel Quartus Prime compilation report, which shows TCCS values for serial output ports. You can also get the TCCS value from the device datasheet.

TCCS is the maximum skew observed across the channels of data and TX output clock —the difference between the fastest and slowest data output transitions, including the T_{CO} variation and clock skew.

Related Information

- Receiver Skew Margin for Non-DPA Mode, Intel Arria 10 Core Fabric and General Purpose I/Os Handbook
- Receiver Skew Margin for Non-DPA Mode, Intel Cyclone 10 GX Core Fabric and General Purpose I/Os Handbook
- Assigning Input Delay to LVDS Receiver Using TimeQuest Timing Analyzer, Intel Arria 10 Core Fabric and General Purpose I/Os Handbook
- Assigning Input Delay to LVDS Receiver Using TimeQuest Timing Analyzer, Intel Cyclone 10 GX Core Fabric and General Purpose I/Os Handbook
- Transmitter Channel-to-Channel Skew, Intel Arria 10 Core Fabric and General Purpose I/Os Handbook
- Transmitter Channel-to-Channel Skew, Intel Cyclone 10 GX Core Fabric and General Purpose I/Os Handbook

- High-Speed I/O Specifications, Intel Arria 10 Device Datasheet
  Provides the TCCS value for Intel Arria 10 devices.
- High-Speed I/O Specifications, Intel Cyclone 10 GX Device Datasheet
  Provides the TCCS value for Intel Cyclone 10 GX devices.
- KDB: Why is the LVDS Receiver Package Skew Compensation report missing in Intel Quartus Prime Pro Edition software?

Obtaining RSKM Report

For LVDS receivers, the Intel Quartus Prime software generates the RSKM report that provides the SW, TUI or LVDS period, and RSKM values for the non-DPA mode.

To obtain the RSKM report (`report_rskm`), follow these steps:

1. On the Intel Quartus Prime menu, select **Tools ➤ Timing Analyzer**.
   The **Timing Analyzer** window appears.
2. On the Timing Analyzer menu, select **Reports ➤ Device Specific ➤ Report RSKM**.

Related Information

KDB: Why is the LVDS Receiver Package Skew Compensation report missing in Intel Quartus Prime Pro Edition software?

Obtaining TCCS Report

For LVDS transmitters, the Intel Quartus Prime software generates the TCCS report that provides the TCCS values for serial output ports.

To obtain the TCCS report (`report_tccs`), follow these steps:

1. On the Intel Quartus Prime menu, select **Tools ➤ Timing Analyzer**.
   The **Timing Analyzer** window appears.
2. On the Timing Analyzer menu, select **Reports ➤ Device Specific ➤ Report TCCS**.

FPGA Timing Analysis

When you generate the LVDS SERDES IP core, the IP core generates the SERDES hardware clock settings and the core clock for IP core timing analysis.

Table 18. Clocks for the Transmitter and Receiver in Non-DPA and DPA-FIFO Modes

Because the frequency of LVDS fast clock is higher than the user core clock by the serialization factor, the IP also creates multicycle path constraints for proper timing analysis at the SERDES–core interface.

Clock	Clock Name
Core clock	`<pll_instance_name>_*_outclk[*]`
LVDS fast clock	`<pll_instance_name>_*_lvds_clk[*]`

Table 19. Clock for the Receiver in Soft-CDR Mode

Clock	Clock Name
Core clock	<*lvds_instance_name*>_core_ck_name_<*channel_num*>
DPA fast clock	<*lvds_instance_name*>_dpa_ck_name_<*channel_num*>

To ensure proper timing analysis, instead of multicycle constraints, the IP core creates clock settings at `rx_out` in the following format:

- For rising edge data—
  <*lvds_instance_name*>_core_data_out_<*channel_num*>_<*bit*>
- For falling edge data—
  <*lvds_instance_name*>_core_data_out_<*channel_num*>_<*bit*>_neg

With these proper clock settings, the Timing Analyzer can correctly analyze the timing of the LVDS SERDES–Core interface transfer and within the core transfer.

Timing Analysis for the External PLL Mode

If you enable the **Use external PLL** parameter in the **PLL Settings** tab, the IP generation does not create clock settings for the PLL input and output. You must ensure the PLL clock settings are correct.

Some of the SERDES constraints are derived from the PLL clocks. Therefore, the external PLL clock settings must be generated before the LVDS SERDES IP core clock settings. In you project's `.qsf`, ensure that the line for the IOPLL IP core's `.qip` appears before the line for the LVDS SERDES IP core's `.qip`.

Add the following line in your `.sdc` file to ensure all the PLL clocks are derived correctly.

```
derive_pll_clocks -create_base_clocks
```

Related Information

Knowledge Base page to work around Intel Quartus Prime error during timing analysis.

Timing Closure Guidelines for Internal FPGA Paths

Closing timing at the internal FPGA paths is challenging for an LVDS SERDES design with high frequency and low SERDES factor.

If you observe setup violation from core registers to LVDS transmitter hardware, check the **TX core registers clock** parameter:

- If the parameter is set to `inclock`, consider changing it to `tx_coreclock`. Core registers that use `tx_coreclock` have less clock delay. Because of the PLL compensation delay on the `tx_coreclock` path, there is less source clock delay and more setup slack for the transfer.
- If the parameter is set to `tx_coreclock`, consider lowering the data rate or increasing the SERDES factor to reduce the core frequency requirement and provide more setup slack.

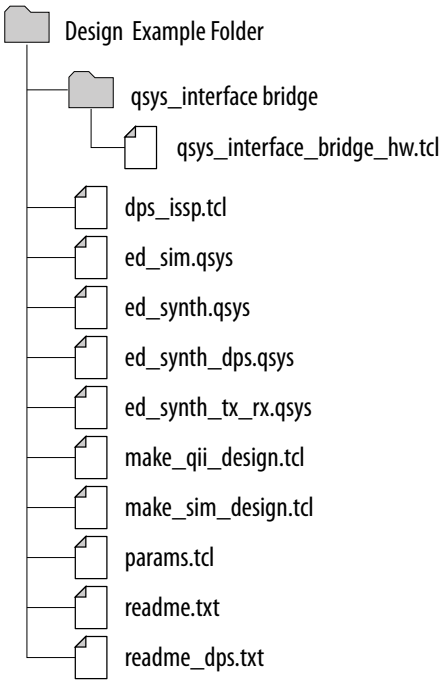
If you observe hold violation from the LVDS receiver to core registers, consider checking the setup slack of the transfer. If there is ample setup slack, you can attempt to over-constraint the hold for the transfer. Normally, the Fitter attempts to correct the hold violation by adding delay. Under certain circumstances, the Fitter may have calculated that adding more delay for avoiding hold violation at the fast corner can negatively affect setup at the slow corner.

LVDS SERDES IP Core Design Examples

The LVDS SERDES IP core can generate several design examples that match your IP configuration in the parameter editor. You can use these design examples as references for instantiating the IP core and the expected behavior in simulations.

You can generate the design examples from the LVDS SERDES IP core parameter editor. After you have set the parameters that you want, click **Generate Example Design**. The IP core generates the design example source files in the directory you specify.

Figure 11. Source Files in the Generated Design Example Directory



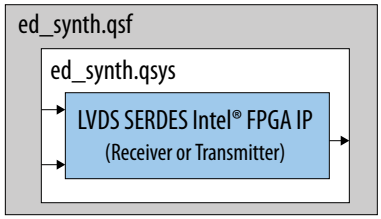
LVDS SERDES IP Core Synthesizable Intel Quartus Prime Design Examples

The synthesizable design example is a compilation-ready Platform Designer system that you can include in an Intel Quartus Prime project.

The design example uses the parameter settings you configured in the IP core parameter editor:

- Basic LVDS SERDES IP core system with transmitters or receivers
- LVDS SERDES IP core system with transmitters or receivers connected to an external PLL

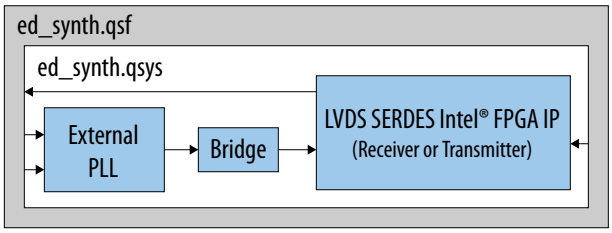
Figure 12. Basic LVDS SERDES IP Core System with Internal PLL



If you configured the IP core to use an external PLL, the generated design example connects a properly configured IOPLL Intel FPGA IP.

Figure 13. LVDS SERDES IP Core System with External PLL

In this figure, a qsys_interface_bridge provides Platform Designer connections between the IOPLL IP core and the LVDS SERDES IP core. For simplicity, this bridge is not shown in the other figures.



To demonstrate how to configure the PLL, the design example also provides the `lvds_external_pll.qsys` Platform Designer file containing a standalone version of the IOPLL IP core configured to work as an external PLL. You can use `lvds_external_pll.qsys`, modified or unmodified, to build an LVDS design with external PLL.

Generating and Using the Design Example

To generate the synthesizable Intel Quartus Prime design example from the source files, run the following command in the design example directory:

```
quartus_sh -t make_qii_design.tcl -system ed_synth
```

The TCL script creates a `qii` directory that contains the `ed_synth.qpf` project file. You can open and compile this project in the Intel Quartus Prime software.

For more information about `make_qii_design.tcl` arguments, run the following command:

```
quartus_sh -t make_qii_design.tcl -help
```

Related Information

LVDS Interface with External PLL Mode on page 31

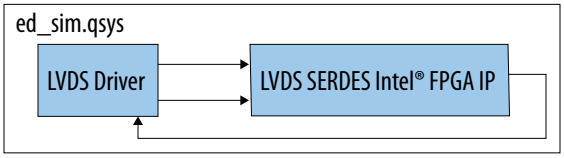
LVDS SERDES IP Core Simulation Design Example

The simulation design example uses your LVDS SERDES IP core parameter settings to build the IP instance connected to a non-synthesizable simulation driver.

Using the design example, you can run a simulation using a single command, depending on the simulator that you use. The simulation demonstrates how you can use the LVDS SERDES IP core.

Note: The non-synthesizable simulation driver works for the transmitter or receiver mode. However, to function in any receiver mode, the driver requires bitslip.

Figure 14. LVDS SERDES IP Core Simulation



Generating and Using the Design Example

To generate the simulation design example from the source files for a Verilog simulator, run the following command in the design example directory:

```
quartus_sh -t make_sim_design.tcl VERILOG
```

To generate the simulation design example from the source files for a VHDL simulator, run the following command in the design example directory:

```
quartus_sh -t make_sim_design.tcl VHDL
```

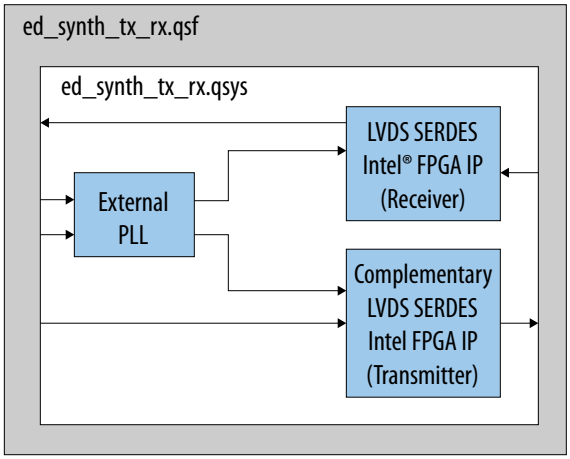
The TCL script creates a `sim` directory that contains subdirectories—one for each supported simulation tool. You can find the scripts for each simulation tool in the corresponding directories.

Combined LVDS SERDES IP Core Transmitter and Receiver Design Example

The combined transmitter and receiver design example uses your LVDS SERDES IP core parameter settings and adds a complementary transmitter or receiver interface. Both interfaces are connected to the same external PLL. You can use the design example to see how to connect the transmitter and receiver interfaces.

If your LVDS SERDES IP core configuration implements a transmitter, the design example adds a DPA-FIFO receiver. If your LVDS SERDES IP core configuration implements any of the receiver interfaces, the design example adds a transmitter.

Figure 15. Combined LVDS SERDES Transmitter and Receiver



Generating and Using the Design Example

To generate the combined transmitter and receiver design example from the source files, run the following command in the design example directory:

```
quartus_sh -t make_qii_design.tcl -system ed_synth_tx_rx
```

The TCL script creates a `qii_ed_synth_tx_rx` directory that contains the `ed_synth_tx_rx.qpf` project file. You can open and compile this project in the Intel Quartus Prime software.

For more information about make_qii_design.tcl arguments, run the following command:

```
quartus_sh -t make_qii_design.tcl -help
```

LVDS SERDES IP Core Dynamic Phase Shift Design Example

The dynamic phase shift design example provides you live control over the PLL clock shifts in an LVDS design through a flexible TCL script interface.

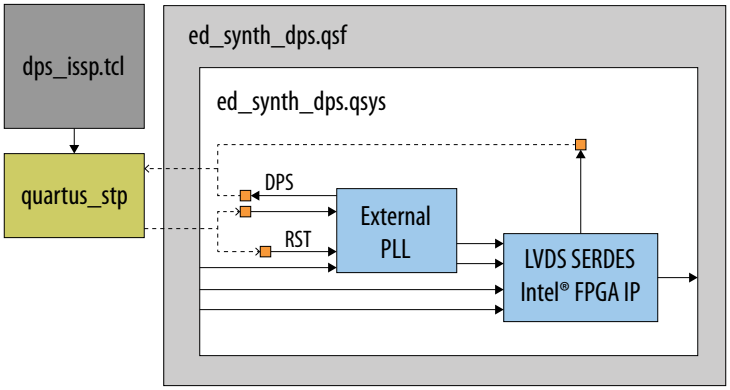
You can use this example in LVDS-specific applications such as debugging non-DPA receiver capture where you can repeatedly shift the capture clock to find the best operational phase shift.

You can also use the design example as a general example of using the In-System Sources and Probes feature with Signal Tap to interface with your hardware through TCL scripting. This method allows you to use manual switches to test a board without being physically present.

The dynamic phase shift design example uses LVDS SERDES IP core parameter settings and connects the IP core to an external PLL. The PLL has an exposed dynamic phase shift interface that connects to in-system sources and probes. This connection allows you to control the PLL using the In-System Sources and Probes editor or the provided TCL script in conjunction with Signal Tap.

A part of the LVDS SERDES IP core in the design example is also connected to the in-system sources and probes. The provided TCL script shows an example of how you can shift a selected PLL clock and also provides you some utility functions. You can use this example script as a start towards accomplishing the testing function that you want.

Figure 16. LVDS SERDES IP Core Dynamic Phase Shift



Generating and Using the Design Example

To generate the combined dynamic phase shift design example from the source files, run the following command in the design example directory:

```
quartus_sh -t make_qii_design.tcl -system ed_synth_dps
```

The TCL script creates a `qii_ed_synth_dps` directory that contains the `ed_synth_dps.qpf` project file. You can open and compile this project in the Intel Quartus Prime software.

To use the provided TCL script to control the in-system sources and probes, run the following command:

```
quartus_stp -t dps_issp.tcl qii_ed_synth_dps/ed_synth_dps
```

Note: For the control to work, you must first program the FPGA.

For more information about make_qii_design.tcl arguments, run the following command:

```
quartus_sh -t make_qii_design.tcl -help
```

Related Information

- Design Debugging Using In-System Sources and Probes
  Provides more information about the In-System Sources and Probes Editor.
- Tcl Interface for the In-System Sources and Probes Editor
  Provides more information about using the Tcl interface to automate the In-System Sources and Probes Editor.
- AN 728: I/O PLL Reconfiguration and Dynamic Phase Shift for Intel Arria 10 Devices
  Provides more information about the PLL dynamic phase shift.

Additional LVDS SERDES IP Core References

IP Migration Flow for Arria V, Cyclone V, and Stratix V Devices

The IP migration flow allows you to migrate the ALTLVDS_TX and ALTLVDS_RX IP cores of Arria V, Cyclone V, and Stratix V devices to the LVDS SERDES IP core of Intel Arria 10 and Intel Cyclone 10 GX devices.

This IP migration flow configures the LVDS SERDES IP core to match the settings of the ALTLVDS_TX and ALTLVDS_RX IP cores, allowing you to regenerate the IP core.

Note: Some IP cores support the IP migration flow in specific modes only. If your IP core is in a mode that is not supported, you may need to run the IP Parameter Editor for the LVDS SERDES IP core and configure the IP core manually.

Migrating Your ALTLVDS_TX and ALTLVDS_RX IP Cores

To migrate your ALTLVDS_TX and ALTLVDS_RX IP cores to the LVDS SERDES Intel FPGA IP, follow these steps:

1. Open your ALTLVDS_TX or ALTLVDS_RX core in the IP parameter editor.
2. In the **Currently selected device family**, select **Arria 10** or **Cyclone 10 GX**.
3. Click **Finish** to open the LVDS SERDES IP core parameter editor. The parameter editor configures the LVDS SERDES IP core settings similar to the ALTLVDS_TX or ALTLVDS_RX IP core settings.
4. If there are any incompatible settings between the two IP cores, select **new supported settings**.
5. Click **Finish** to regenerate the IP core.
6. Replace your ALTLVDS_TX or ALTLVDS_RX IP core instantiation in RTL with the LVDS SERDES IP core.

Note: The LVDS SERDES IP core port names may not match the ALTLVDS_TX or ALTLVDS_RX IP core port names. Therefore, simply changing the IP core name in the instantiation may not be sufficient.

LVDS Interface with External PLL Mode

The LVDS SERDES IP core parameter editor provides an option for implementing the LVDS interface with the **Use External PLL** option. With this option enabled you can control the PLL settings, such as dynamically reconfiguring the PLL to support different data rates, dynamic phase shift, and other settings.

If you enable the **Use External PLL** option with the LVDS SERDES IP core transmitter and receiver, the following signals are required from the IOPLL Intel FPGA IP:

- Serial clock (fast clock) input to the SERDES of the LVDS SERDES IP core transmitter and receiver
- Load enable to the SERDES of the LVDS SERDES IP core transmitter and receiver
- Parallel clock (core clock) used to clock the transmitter FPGA fabric logic and parallel clock used for the receiver
- Asynchronous PLL reset port of the LVDS SERDES IP core receiver
- PLL VCO signal for the DPA and soft-CDR modes of the LVDS SERDES IP core receiver

The **Clock Resource Summary** tab in the LVDS SERDES IP core parameter editor provides the details for the signals in the preceding list.

You must instantiate an IOPLL IP core to generate the various clocks and load enable signals. You must configure these settings in IOPLL IP core parameter editor:

- **LVDS External PLL** options in the **Settings** tab
- **Output Clocks** options in the **PLL** tab
- **Compensation Mode** option in the **PLL** tab

Table 20. Compensation Mode Setting to Generate IOPLL IP Core

When you generate the IOPLL IP core, use the PLL setting in this table for the corresponding LVDS functional mode.

LVDS Functional Mode	IOPLL IP Core Setting
TX, RX DPA, RX Soft-CDR	Direct mode
RX non-DPA	LVDS compensation mode

Related Information

- LVDS SERDES IP Core PLL Settings on page 15
- LVDS SERDES IP Core Synthesizable Intel Quartus Prime Design Examples on page 26
- LVDS SERDES IP Core Signals on page 13

IOPLL IP Core Signal Interface with LVDS SERDES IP Core

Table 21. Signal Interface between IOPLL and LVDS SERDES IP cores

This table lists the signal interface between the output ports of the IOPLL IP core and the input ports of the LVDS SERDES IP core transmitter and receiver.

From the IOPLL IP core	To the LVDS SERDES IP core transmitter	To the LVDS SERDES IP core receiver
`lvds_clk[0]` (serial clock output signal)	`ext_fclk` (serial clock input to the transmitter)	`ext_fclk` (serial clock input to the receiver)

continued...

From the IOPLL IP core	To the LVDS SERDES IP core transmitter	To the LVDS SERDES IP core receiver
• Configure this signal using `outclk0` in the PLL. • Select **Enable LVDS_CLK/ LOADEN 0** or **Enable LVDS_CLK/ LOADEN 0 & 1** option for the **Access to PLL LVDS_CLK/ LOADEN output port** setting. In most cases, select **Enable LVDS_CLK/LOADEN 0**. The serial clock output can only drive `ext_fclk` on the LVDS SERDES IP core transmitter and receiver. This clock cannot drive the core logic.		
`loaden[0]` (load enable output) • Configure this signal using `outclk1` in the PLL. • Select **Enable LVDS_CLK/ LOADEN 0** or **Enable LVDS_CLK/ LOADEN 0 & 1** option for the **Access to PLL LVDS_CLK/ LOADEN output port** setting. In most cases, select **Enable LVDS_CLK/LOADEN 0**.	`ext_loaden` (load enable to the transmitter)	`ext_loaden` (load enable for the deserializer) This signal is not required for LVDS receiver in soft-CDR mode.
`outclk2` (parallel clock output)	`ext_coreclock` (parallel core clock)	`ext_coreclock` (parallel core clock)
`locked`	—	`pll_areset` (asynchronous PLL reset port)
`phout[7:0]` • This signal is required only for LVDS receiver in DPA or soft-CDR mode. • Configure this signal by turning on **Specify VCO frequency** in the PLL and specifying the **VCO frequency** value. • Turn on **Enable access to PLL DPA output port**.	—	`ext_vcoph` This signal is required only for LVDS receiver in DPA or soft-CDR mode.

IOPLL Parameter Values for External PLL Mode

The following examples show the clocking requirements to generate output clocks for LVDS SERDES IP core using the IOPLL IP core. The examples set the phase shift with the assumption that the clock and data are edge aligned at the pins of the device.

Note: For other clock and data phase relationships, Intel recommends that you first instantiate your LVDS SERDES IP core interface without using the external PLL mode option. Compile the IP cores in the Intel Quartus Prime software and take note of the frequency, phase shift, and duty cycle settings for each clock output. Enter these settings in the IOPLL IP core parameter editor and then connect the appropriate output to the LVDS SERDES IP cores.

Table 22. Example: Generating Output Clocks Using an IOPLL IP core (Receiver in Non-DPA Mode)

This table lists the parameter values that you can set in the IOPLL IP core parameter editor to generate three output clocks using an IOPLL IP core if you are using the non-DPA receiver.

Parameter	outclk0 (Connects as `lvds_clk[0]` to the `ext_fclk` port of LVDS SERDES IP core transmitter or receiver)	outclk1 (Connects as `loaden[0]` to the `ext_loaden` port of LVDS SERDES IP core transmitter or receiver)	outclk2 (Used as the core clock for the parallel data registers for both transmitter and receiver, and connects to the `ext_coreclock` port of LVDS SERDES IP core)
Frequency	data rate	data rate/serialization factor	data rate/serialization factor
Phase shift	180°	[(deserialization factor – 1)/ deserialization factor] x 360°	180/serialization factor (outclk0 phase shift divided by the serialization factor)
Duty cycle	50%	100/serialization factor	50%

The calculations for phase shift, using the RSKM equation, assume that the input clock and serial data are edge aligned. Introducing a phase shift of 180° to sampling clock (outclk0) ensures that the input data is center-aligned with respect to the outclk0, as shown in the following figure.

Figure 17. Phase Relationship for External PLL Interface Signals

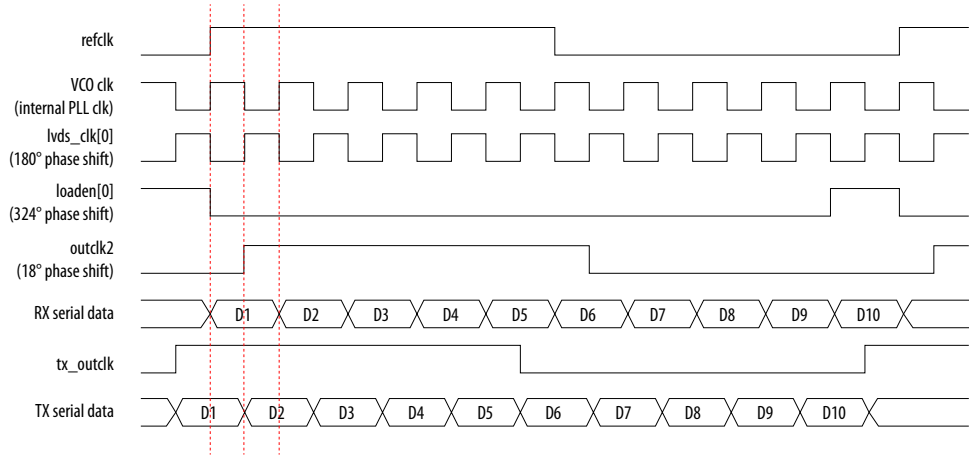


Table 23. Example: Generating Output Clocks Using an IOPLL IP core (Receiver in DPA or Soft-CDR Mode)

This table lists the parameter values that you can set in the IOPLL IP core parameter editor to generate four output clocks using an IOPLL IP core if you are using the DPA or soft-CDR receiver. The `locked` output port of IOPLL IP core must be inverted and connected to the `pll_areset` port of the LVDS SERDES IP core if you are using the DPA or soft-CDR receiver.

Parameter	outclk0 (Connects as `lvds_clk[0]` to the `ext_fclk` port of LVDS SERDES IP core transmitter or receiver)	outclk1 (Connects as `loaden[0]` to the `ext_loaden` port of LVDS SERDES IP core transmitter or receiver) Not required for the soft-CDR receiver.	outclk2 (Used as the core clock for the parallel data registers for both transmitter and receiver, and connects to the `ext_coreclock` port of LVDS SERDES IP core)	VCO Frequency (Connects as `phout[7:0]` to the `ext_vcoph[7:0]` port of LVDS SERDES IP core)
Frequency	data rate	data rate/serialization factor	data rate/serialization factor	data rate
Phase shift	180°	[(deserialization factor - 1)/deserialization factor] x 360°	180/serialization factor (outclk0 phase shift divided by the serialization factor)	—
Duty cycle	50%	100/serialization factor	50%	—

Table 24. Example: Generating Output Clocks Using a Shared IOPLL IP core for Transmitter Spanning Multiple Banks Shared with Receiver Channels (Receiver in DPA or Soft-CDR Mode)

This table lists the parameter values that you can set in the IOPLL IP core parameter editor to generate six output clocks using an IOPLL IP core. Use these settings if you use transmitter channels that span multiple banks shared with receiver channels in DPA or soft-CDR mode. The `locked` output port of IOPLL IP core must be inverted and connected to the `pll_areset` port of the LVDS SERDES IP core if you are using the DPA or soft-CDR mode.

Parameter	outclk0 (Connects as `lvds_clk[0]` to the `ext_fclk` port of LVDS SERDES IP core receiver)	outclk1 (Connects as `loaden[0]` to the `ext_loaden` port of LVDS SERDES IP core receiver) Not required for the soft-CDR receiver.	outclk4 (Used as the core clock for the parallel data registers for both transmitter and receiver, and connects to the `ext_coreclock` port of LVDS SERDES IP core)	VCO Frequency (Connects as `phout[7:0]` to the `ext_vcoph[7:0]` port of LVDS SERDES IP core)
	outclk2 (Connects as `lvds_clk[1]` to the `ext_fclk` port of LVDS SERDES IP core transmitter)	outclk3 (Connects as `loaden[1]` to the `ext_loaden` port of LVDS SERDES IP core transmitter)		
Frequency	data rate	data rate/serialization factor	data rate/serialization factor	data rate
Phase shift	180°	[(deserialization factor - 1)/deserialization factor] x 360°	180/serialization factor (outclk0 phase shift divided by the serialization factor)	—
Duty cycle	50%	100/serialization factor	50%	—

Connection between IOPLL IP Core and LVDS SERDES IP Core in External PLL Mode

Figure 18. Non-DPA LVDS Receiver Interface with IOPLL IP Core in External PLL Mode

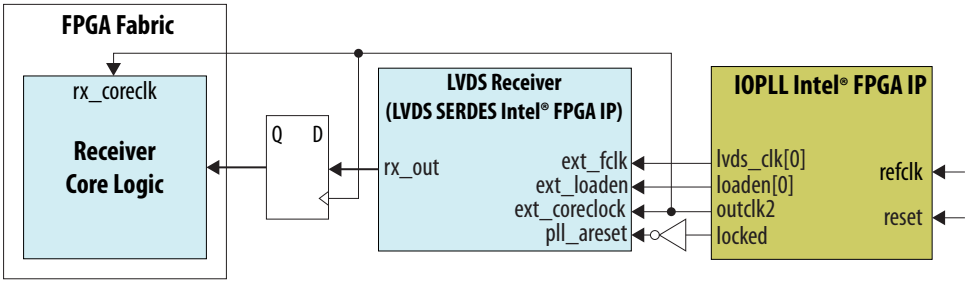


Figure 19. DPA LVDS Receiver Interface with the IOPLL IP Core in External PLL Mode

Invert the `locked` output port and connect it to the `pll_areset` port.

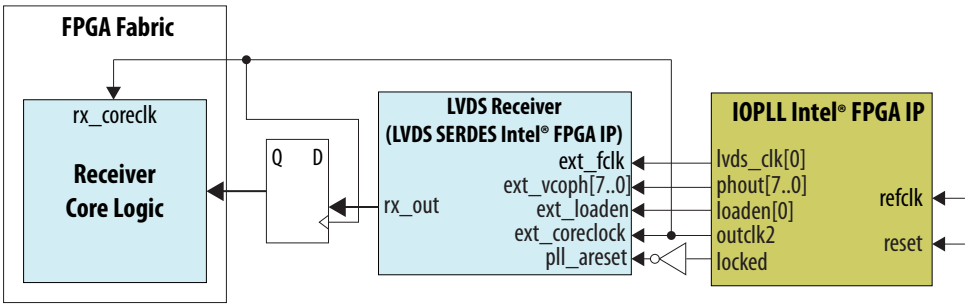


Figure 20. Soft-CDR LVDS Receiver Interface with the IOPLL IP Core in External PLL Mode

Invert the `locked` output port and connect it to the `pll_areset` port.

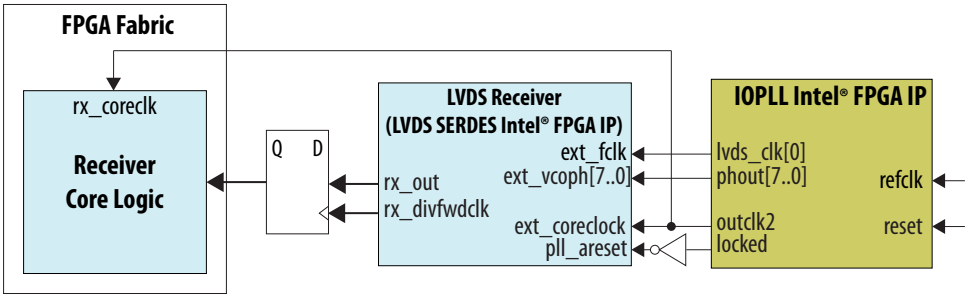
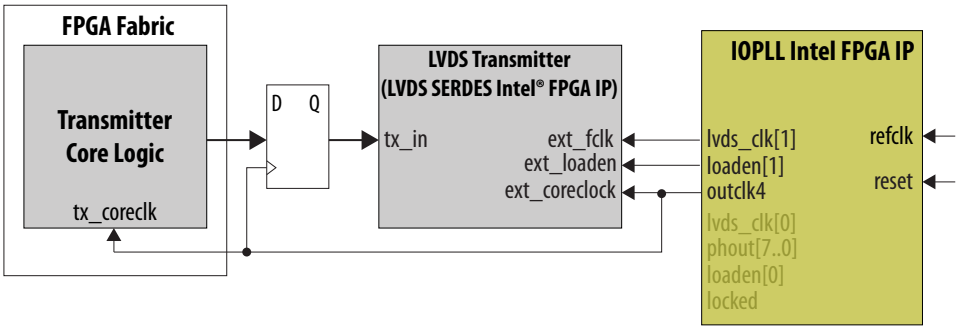


Figure 21. LVDS Transmitter Interface with the IOPLL IP Core in External PLL Mode

Connect the I/O PLL `lvds_clk[1]` and `loaden[1]` ports to the `ext_fclk` and `ext_loaden` ports of the LVDS transmitter.



The `ext_coreclock` port is automatically enabled in the LVDS SERDES IP core in external PLL mode. The Intel Quartus Prime compiler outputs error messages if this port is not connected as shown in the preceding figures.

LVDS Transmitters and Receivers in the Same I/O Bank

If you want to place both LVDS transmitter and receiver interfaces in the same I/O bank, you can use the LVDS SERDES IP core with an external PLL.

- To use an external PLL, in the LVDS SERDES IP parameter editor, turn on the **Use external PLL** option.
- You can generate two instances of the LVDS SERDES IP—a receiver and a transmitter.
- In each instance, you can use up to the following number of channels:
  - 71 transmitters
  - 23 DPA or non-DPA receivers
  - 12 soft-CDR receivers
- Connect the same PLL to both the transmitter and receiver instances.

Figure 22. LVDS Interface with the IOPLL IP (Non-DPA Mode)

This figure shows the connections you need to make between the IOPLL IP and the LVDS SERDES IP in external PLL mode if you are using non-DPA mode.

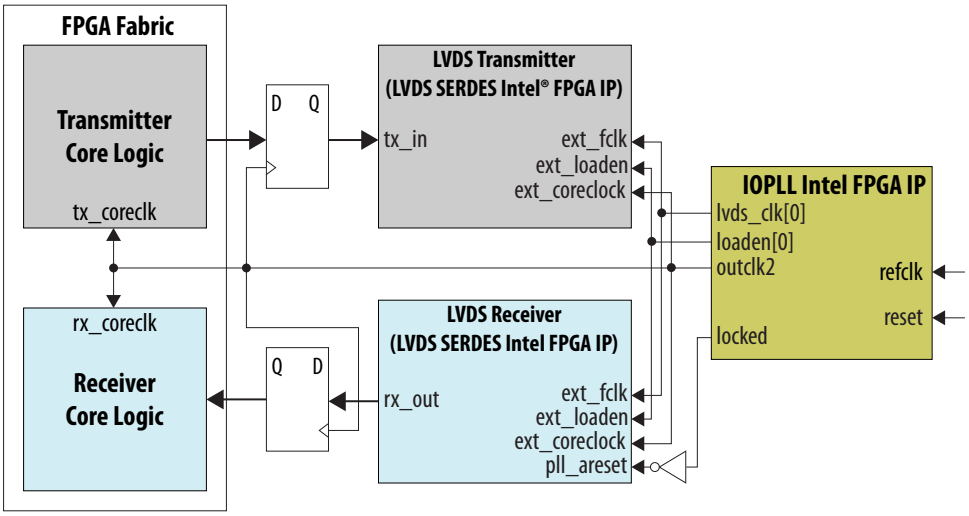


Figure 23. LVDS Interface with the IOPLL IP (DPA Mode)

This figure shows the connections you need to make between the IOPLL IP and the LVDS SERDES IP in external PLL mode if you are using DPA. Invert the `locked` output port and connect it to the `pll_areset` port.

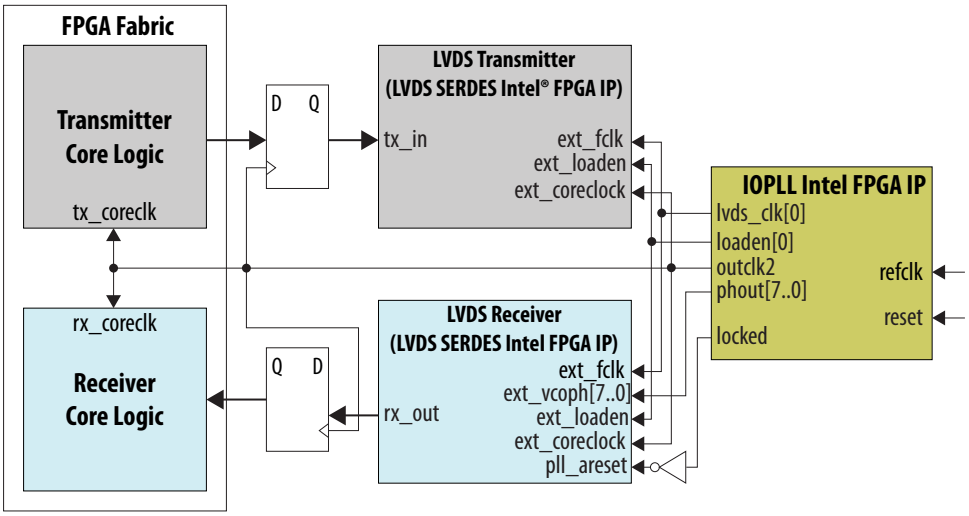
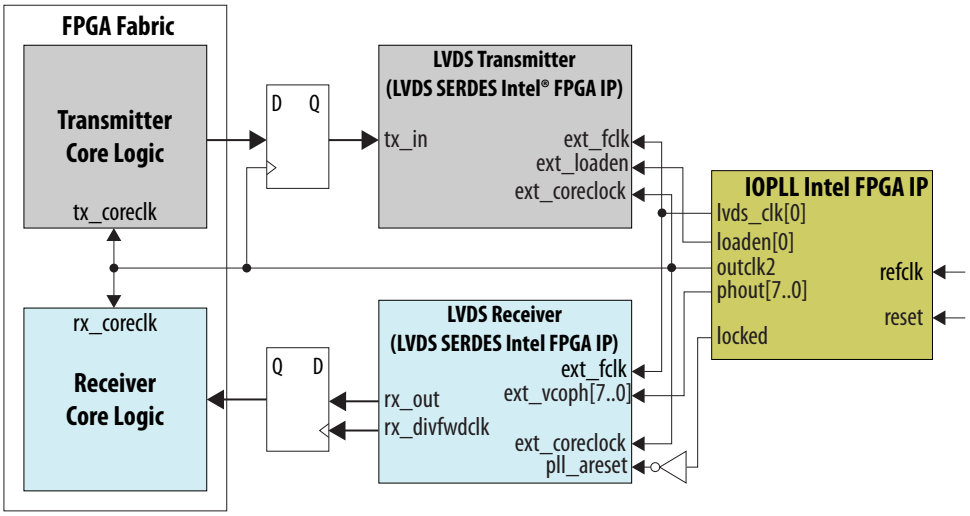


Figure 24. LVDS Interface with the IOPLL IP (Soft-CDR Mode)

This figure shows the connections you need to make between the IOPLL IP and the LVDS SERDES IP core if you are using soft-CDR mode. Invert the `locked` output port and connect it to the `pll_areset` port.



Comparison of LVDS SERDES IP Core with Stratix V SERDES

The LVDS SERDES IP core has similar features to the Stratix V SERDES. The key differences are the clock network and the ubiquitous RX and TX resource in LVDS I/O banks.

Table 25. Intel Arria 10/Intel Cyclone 10 GX and Stratix V Devices Feature Comparison

Features	Intel Arria 10/Intel Cyclone 10 GX Devices	Stratix V Devices
Operation Frequency Range	150 MHz - 1.6 GHz(2)	
Serialization/Deserialization Factors	3 to 10	
Regular DPA and non-DPA mode	Supported	
Clock Forwarding for Soft-CDR	Supported	
RX Resource	Every I/O pair (Every two I/O pairs for CDR)	Every two I/O pairs on every side without HSSI transceivers
TX Resource	Every I/O pair	Every two I/O pairs every side without HSSI transceivers
PLL Resource	TX channels can span three adjacent banks, driven by the IOPLL in the middle bank. RX channels are driven by the IOPLL in the same bank.	RX and TX channels placed on one edge can be driven by the corner or center PLL.
Number of DPA Clock Phase	8	
I/O Standard	True LVDS	True LVDS, pseudo-differential output

(2) The supported operation frequency range depends on the device, speed grade, and SERDES factor. Refer to the relevant device datasheet.

Related Information

- Stratix V Device Datasheet
- Intel Cyclone 10 GX Device Datasheet
- Intel Arria 10 Device Datasheet

LVDS SERDES Intel FPGA IP User Guide Archives

IP versions are the same as the Intel Quartus Prime Design Suite software versions up to v19.1. From Intel Quartus Prime Design Suite software version 19.2 or later, IP cores have a new IP versioning scheme.
If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
20.2	LVDS SERDES Intel FPGA IP User Guide: Intel Arria 10 and Intel Cyclone 10 GX Devices
19.3.0	LVDS SERDES Intel FPGA IP User Guide: Intel Arria 10 and Intel Cyclone 10 GX Devices
19.1	LVDS SERDES Intel FPGA IP User Guide: Intel Arria 10 and Intel Cyclone 10 GX Devices
18.1	LVDS SERDES Intel FPGA IP User Guide: Intel Arria 10 and Intel Cyclone 10 GX Devices
18.0	LVDS SERDES Intel FPGA IP User Guide: Intel Arria 10 and Intel Cyclone 10 GX Devices
17.1	Intel FPGA LVDS SERDES IP Core User Guide
17.0	Altera LVDS SERDES IP Core User Guide
16.0	Altera LVDS SERDES IP Core User Guide
15.1	Altera LVDS SERDES IP Core User Guide
14.1	Altera LVDS SERDES IP Core User Guide
13.1	Altera LVDS SERDES Megafunction User Guide

Document Revision History for LVDS SERDES Intel FPGA IP User Guide: Intel Arria 10 and Intel Cyclone 10 GX Devices

Document Version	Intel Quartus Prime Version	IP Version	Changes
2020.03.29	21.1	20.0.0	Updated the LVDS SERDES IP version number.
2020.09.25	20.2	19.4.0	Removed the **Use clock-pin drive** parameter from the LVDS SERDES IP core general settings.
2020.07.10	20.2	19.4.0	• Added link to the KDB article about missing RSKM report in Intel Quartus Prime Pro Edition in the section about I/O timing analysis. • Updated the footnote in the *Comparison of LVDS SERDES IP Core with Stratix V SERDES* topic to clarify that the operation frequency range depends on the product line, speed grade, and SERDES factor.
2020.05.06	19.4	19.3.0	Added the *Tcl error: ERROR: Argument <clk_object> is a collection with more than one object. Specify a collection with one object. while executing "get_clock_info -period [get_clocks [lindex $fclk_setting_name 0]]* KDB link in the *Timing Analysis for the External PLL Mode* topic.

continued...

Document Version	Intel Quartus Prime Version	IP Version	Changes
2020.03.10	19.4	19.3.0	• Added *Release Information* topic. • Updated the *Timing Analysis for the External PLL Mode* topic with a command to include in the `.sdc` file to derive all PLL clocks.
2019.05.03	19.1	19.1	• Moved the *Usage Modes Summary of the LVDS SERDES* table to the following documents: — I/O and High Speed I/O in Intel Arria 10 Devices chapter of the *Intel Arria 10 Core Fabric and General Purpose I/Os Handbook* — I/O and High Speed I/O in Intel Cyclone GX 10 Devices chapter of the *Intel Cyclone GX 10 Core Fabric and General Purpose I/Os Handbook* • Updated the table that lists the functional modes of the LVDS SERDES IP core to specify that all functional modes support SERDES factors of 3 to 10.
2019.01.30	18.1	18.1	Added *Usage Modes Summary of the LVDS SERDES* table in *LVDS IP Core Features* topic.
2018.12.05	18.1	18.1	• Updated the topic about the timing analysis for the external PLL mode to improve clarity. • Updated the topic about the simulation design example to add a note about the non-synthesizable simulation driver. • Renamed "TimeQuest Timing Analyzer" to "Timing Analyzer". • Renamed "SignalTap" to "Signal Tap".
2018.09.06	18.0	18.0	• Removed `ext_loaden` signal in figures showing the LVDS receiver in soft-CDR mode. • Specified that connecting the IOPLL loaden signal to the LVDS receiver ext_loaden signal is not required for LVDS receivers in soft-CDR mode. • Updated the figures descriptions in the guideline topic about using LVDS transmitters and receivers in the same I/O bank to clarify that the figures show connections that you need to make. • Updated the synthesizable design example topic to improve clarity. • Updated the names of the following IP cores: — Intel FPGA LVDS SERDES to LVDS SERDES Intel FPGA IP — Intel FPGA IOPLL to IOPLL Intel FPGA IP • Updated the document title.

Date	Version	Changes
November 2017	2017.11.06	• Corrected typographical error in the example showing the parameter values to generate output clock in external PLL mode by updating "c0" to "outclk0". • Added more description for the **Enable tx_coreclock port** parameter option to describe how configure it in external PLL mode. • Updated the description of the `tx_coreclock` signal. • Added Intel Cyclone 10 GX device support.

continued...

Date	Version	Changes
		• Renamed the IP core from "Altera LVDS SERDES" to "LVDS SERDES". • Specified that in Intel Arria 10 devices, the maximum operation frequency for SERDES factor 3 is 1.25 GHz. • Restructured the information in the topic about connecting the external PLL to the LVDS receiver and transmitter. Moved some of the information to the topic about using external PLL for combined LVDS transmitters and receivers in the same I/O bank.
May 2017	2017.05.08	• Updated the topic about the LVDS interface with external PLL mode to clarify that the **Clock Resource Summary** tab in the LVDS SERDES IP core parameter editor provides the details for the signals required from the IOPLL IP core. • Updated the description for the **Number of channels** parameter in the table listing the LVDS SERDES **General Settings** tab to improve clarity and specify the placement of the `refclk` and `tx_outclock` pins. • Rebranded as Intel.
August 2016	2016.08.05	• Updated the topics about using the LVDS interface with external PLL mode. The update adds examples and connection diagrams for using transmitter channels that span multiple banks and shared with receiver channels in DPA and soft-CDR modes. • Restructured the section about IP core initialization and reset to simplify and improve clarity.
December 2015	2015.12.14	• Rewrote and restructured the document to improve clarity and for ease of reference. • Updated the signal names generated by the LVDS SERDES internal PLL. • Removed the I/O timing analysis topics and added links to the relevant topics in the *Arria 10 Core Fabric and General Purpose I/Os Handbook*. • Updated the core clock cycles to wait before checking if the data is aligned for bitslip from five cycles to four cycles. • Updated the number of core clock cycles the `rx_bitslip_max` signal is asserted after rollover from five cycles to four cycles. • Removed the statement about waiting two core clock cycles before resetting the bitslip after FIFO resets. • Updated the section about design examples. The LVDS SERDES IP core now provides more design examples. • Added topics about creating LVDS interfaces using external PLL in the additional references section. • Updated the bitslip rollover value in the topic about receiver settings. The bitslip rollover value is now set automatically to the deserialization factor. • Added related information links from several topics to relevant topics in the Intel Arria 10 device handbook and datasheet.
August, 2014	2014.08.18	• Clarified that you must wait five core clock cycles before checking if the data is aligned for bitslip circuitry. • Changed the `rx_out[9:0]` signal to `rx_out[7:0]` for the deserializer. • Clarified that if one of the pins is taken for the `refclk`, then the value is 1 to 71 for TX and 1 to 23 for RX. This change is implemented for the **Number of channels** parameter. • Clarified that if one of the pins is taken for the `tx_outclock`, then the value is 1 to 71 for TX. This change is implemented for the **Number of channels** parameter. • Added a new parameter (**Use backwards-compatible port names**). • The **Use external PLL** is supported in the 14.0a10 release. The Clock Resource Summary tab guides you to configure your external PLL.

continued...

Date	Version	Changes
		• Removed the **Enable pll_locked port** and **Enable rx_dpa_locked port** parameters. • Added the external PLL signals. • Added timing information.
November, 2013	2013.11.29	Initial release.