Altera LVDS SERDES Megafunction User Guide

2013.11.29

ug_altera_lvds

Subscribe Feedback

You can configure the features of Altera LVDS SERDES megafunction through the MegaWizard® Plug-In Manager in the Quartus® II software. The Altera LVDS SERDES megafunction configures the serializer/deserializer (SERDES) and dynamic phase alignment (DPA) blocks. The megafunction also supports LVDS channels placement, legality checks, and LVDS channel-related rule checks.

The Altera LVDS SERDES megafunction is only available for Arria® 10 devices. For Arria V, Cyclone® V, and Stratix® V devices, follow the steps in **Migrating Your ALTLVDS_TX and ALTLVDS_RX Megafunctions** on page 2 to migrate your IP.

Related Information

- **LVDS SERDES Transmitter/Receiver (ALTLVDS_TX and ALTLVDS_RX) Megafunctions User Guide**

Features

The Altera LVDS SERDES megafunction feature includes the ALTLVDS_RX and ALTLVDS_TX megafunctions features supported in Stratix V devices, such as:

- Parameterizable data channel widths
- Parameterizable serializer/deserializer (SERDES) factors
- Registered input and output ports
- PLL control signals
- Dynamic phase alignment (DPA) mode
- Soft clock data recovery (CDR) mode

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 megafunctions of Arria V, Cyclone V, and Stratix V devices to the Altera LVDS SERDES megafunction of Arria 10 devices.

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

Note: Some megafunctions only support the IP migration flow in specific modes. If your megafunction is in a mode that is not supported, you may need to run the MegaWizard Plug-In Manager for the Altera LVDS SERDES megafunction and configure the megafunction manually.

101 Innovation Drive, San Jose, CA 95134

Migrating Your ALTLVDS_TX and ALTLVDS_RX Megafunctions

To migrate your ALTLVDS_TX and ALTLVDS_RX megafunctions, follow these steps:

1. Open your ALTLVDS_TX or ALTLVDS_RX megafunction in the MegaWizard Plug-In Manager.
2. In the **Currently selected device family**, select **Arria 10**.
3. Click **Finish** to open the Altera LVDS SERDES MegaWizard Plug-In Manager. The MegaWizard Plug-In Manager configures the Altera LVDS SERDES settings similarly to the ALTLVDS_TX or ALTLVDS_RX megafunction settings.
4. If there are any incompatible settings between the two, select **new supported settings**.
5. Click **Finish** to regenerate the megafunction.
6. Replace your ALTLVDS_TX or ALTLVDS_RX megafunction instantiation in RTL with the Altera LVDS SERDES megafunction.

Note: The Altera LVDS SERDES megafunction port names may not match the ALTLVDS_TX or ALTLVDS_RX megafunction port names, so simply changing the megafunction name in the instantiation is not sufficient.

Comparison with Stratix V Devices

The Altera LVDS SERDES megafunction has similar features to the Stratix V SERDES feature. The key difference is the clock network and the ubiquitous RX and TX resource in LVDS I/O banks.

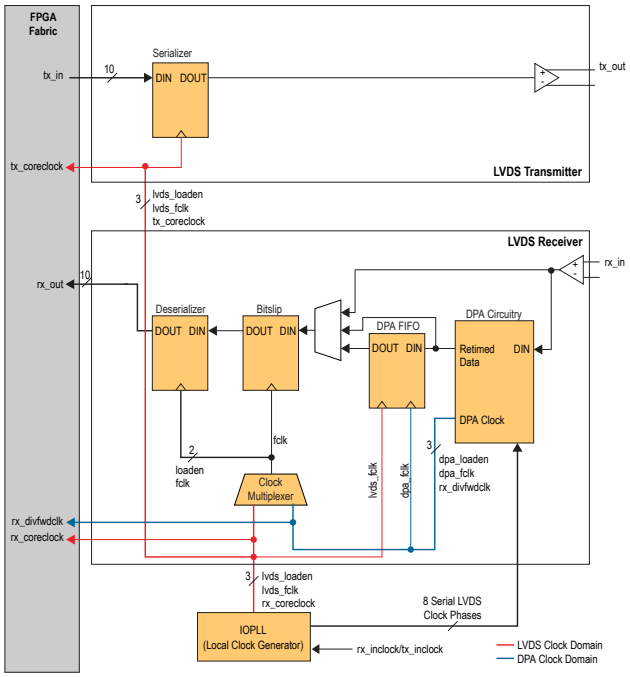
Table 1: Arria 10 and Stratix V Devices Feature Comparison

Features	Arria 10 Devices	Stratix V Devices
Operation Frequency Range	150 MHz - 1.6 GHz	150 Mhz - 1.6 GHz
Serialization/Deserialization Factors	3 to 10	3 to 10
Regular DPA and non-DPA mode	Yes	Yes
Clock Forwarding for Soft-CDR	Yes	Yes
RX Resource	Every I/O pair	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	8
I/O Standard	True LVDS	True LVDS, pseudo-differential output

Functional Description

A single Altera LVDS SERDES 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. You can configure the Altera LVDS SERDES channel as a receiver or a transmitter for a single differential I/O. 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 Altera LVDS SERDES channel in the I/O bank.

Figure 1: Altera LVDS SERDES Channel Diagram



Each Altera LVDS SERDES channel can be broken down into the following paths, with seven functional units:

Path	Block	Modes	Clock Domain
TX Data Path	Serializer	TX mode	LVDS
RX Data Path	DPA Circuitry	DPA FIFO and Soft-CDR modes	DPA
	DPA FIFO	DPA-FIFO mode	LVDS-DPA domain crossing
	Bitslip	All RX modes	LVDS
	Deserializer	All RX modes	LVDS

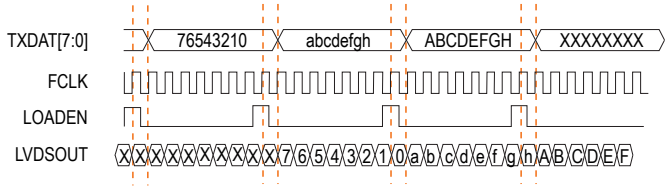
Path	Block	Modes	Clock Domain
Clock Generation and Multiplexers	Local Clock Generator	Soft-CDR mode	Generates PCLK and LOADEN in these modes
	SERDES Clock Multiplexers	All modes	Selects LVDS clock sources for all modes

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 `loaden` clock is provided alongside the LVDS fast clock, to enable these capture registers once per coreclock period. After the data is captured, the data is then loaded into a shift register, which shifts the LSB towards the MSB, one bit per fast clock cycle. The MSB of the shift register feeds the LVDS output buffer; hence, higher order bits precede lower order bits in the output bitstream.

The following figure shows the serializer waveform.

Figure 2: LVDS x8 Serializer Waveform



Signal	Description
`txdat[7:0]`	Data to be serialized (supported serialization factors are 3 -10).
`fclk`	Clock used for transmitter.
`loaden`	Enable signal for serialization.
`lvdsout`	LVDS data stream, output from the Altera LVDS SERDES channel.

DPA FIFO

In DPA-FIFO mode, the DPA FIFO synchronizes the retimed data to the high-speed LVDS clock domain. Because the DPA clock may shift phase during the initial lock period, the FIFO must be held in reset state until the DPA locks; otherwise, there may be a data run-through condition due to the FIFO write pointer creeping up to the read pointer.

Bitslip

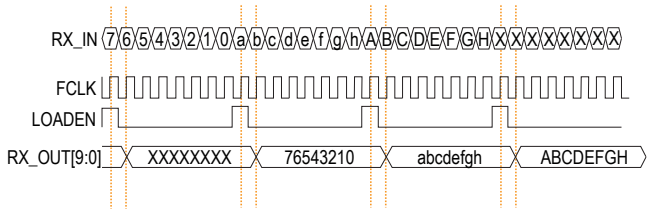
Bitslip circuitry is used to insert latencies in increments of one `fclk` cycle and for data word alignment. The data is slipped one bit for every pulse of the `rx_bitslip_ctrl` signal. You must wait at least two core clock cycles before checking if the data is aligned because it will take at least two core clock cycles to purge the undefined data.

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

Deserializer

The deserializer consists of shift registers. The deserialization factor determines the depth of the shift registers. The `loaden` is a pulse with a frequency of the `fclk` divided by the deserialization factor. The deserializer converts a 1-bit serial data stream into a parallel data stream based on the deserialization factor.

Figure 3: LVDS x8 Deserializer Waveform



Signal	Description
`rx_in`	LVDS data stream, input to the Altera LVDS SERDES channel.
`fclk`	Clock used for receiver.
`loaden`	Enable signal for deserialization.
`rx_out[9:0]`	Deserialized data.

Operation Modes

You can configure the Altera LVDS SERDES megafunction in one of the operation modes listed in the following table:

Table 2: Operation Modes for the Altera LVDS SERDES Megafunction

Mode	Description
Transmitter Mode	The megafunction configures the SERDES block as a serializer. A PLL generates the fast clock (`fclk`) and load enable (`loaden`).

Mode	Description
DPA-FIFO Mode	The DPA block selects an optimal phase to sample incoming data from a set of eight DPA clocks running at the `fclk` frequency, each 45° out of phase. The DPA FIFO, a circular buffer, samples the incoming data with the selected DPA clock and forwards the data to LVDS clock domain. The data released from the DPA-FIFO is then sampled at the bitslip circuitry, where it is lagged, and thus, realigned to match the desired word boundary when it is deserialized. To avoid clock metastability issues, after FIFO resets, wait for two core clock cycles before resetting the bitslip. **Note:** All RX channels must be placed in one I/O bank, which supports up to 24 channels only.
Soft-CDR Mode	In this mode, the optimal DPA clock (DPACLK) is forwarded into the LVDS clock domain, where it is used as the FCLK. The local clock generator produces `rx_divfwdclk` which will be forwarded to the core through a PCLK network. Note, there is a limitation of the number of soft-CDR channels due to PCLK usage. **Note:** RX interfaces must be placed in one I/O bank, and each bank only has 12 PCLK resources, hence 12 soft-CDR channels. **Note:** For actual soft-CDR supported channel, refer to the respective device pin out list.
Non-DPA Mode	In this mode, you must ensure the correct clock-data alignment, as the incoming data is captured at the bitslip with the FCLK. The DPA and DPA-FIFO are bypassed. As in the transmitter mode, the FCLK is provided by a PLL.

Initialization and Reset

This section describes the initialization and reset aspects, using control characters. This section also provides a recommended initialization and reset flow for the Altera LVDS SERDES megafunction.

Initializing the Altera LVDS SERDES Megafunction

With the Altera LVDS SERDES megafunction, the PLL is locked to the reference clock prior to implementing the SERDES blocks for data transfer. The PLL starts to lock to the reference clock during device initialization. The PLL is operational when the PLL achieves lock during user mode. If the clock reference is not stable during device initialization, the PLL output clock phase shifts becomes corrupted.

When the PLL output clock phase shifts are not set correctly, the data transfer between the high-speed LVDS domain and the low-speed parallel domain might not be successful, which leads to data corruption. Assert the `pll_areset` port for at least 10 ns, and then deassert the `pll_areset` port and wait until the PLL lock becomes stable. After the PLL lock port asserts and is stable, the SERDES blocks are ready for operation.

When using DPA, further steps are required for initialization and reset recovery. The DPA circuit samples the incoming data and finds the optimal phase tap from the PLL to capture data on a receiver 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. Use the rx_dpa_reset port to keep the DPA in reset until the PLL lock signal is asserted and stable.

The rx_dpa_locked signal asserts when the DPA has found the optimal phase tap.

Note: Altera recommends asserting the rx_fifo_reset port after the rx_dpa_locked signal asserts, and then deassert the rx_fifo_reset port to begin receiving data.

Each time the DPA shifts the phase taps during normal operation to track variations between the relationship of the reference clock source and the data, the timing margin for the data transfer between clock domains is reduced.

The Altera LVDS SERDES megafunction asserts the rx_dpa_locked port upon initial DPA lock. When you enable the **Enable DPA loss of lock on one change** option, the rx_dpa_locked port deasserts after one change in phase. If this option is disabled, the rx_dpa_locked signal will deassert after two phase changes in the same direction.

Note: Altera recommends using the data checkers to ensure data accuracy.

Resetting the DPA

When the data becomes corrupted, you must reset the DPA circuitry using the rx_dpa_reset port and rx_fifo_reset port.

Assert the rx_dpa_reset port to reset the entire DPA block. This requires the DPA to be trained before it is ready for data capture.

Note: Altera recommends toggling the rx_fifo_reset port after rx_dpa_locked is asserted. This ensures the synchronization FIFO is set with the optimal timing to transfer data between the DPA and high-speed LVDS clock domains.

Assert the rx_fifo_reset port to reset only the synchronization FIFO. This allows you to continue system operation without having to re-train the DPA. Using this port can fix data corruption because it resets the FIFO; however, it does not reset the DPA circuit.

When the DPA is locked, the Altera LVDS SERDES block is ready to capture data. The DPA finds the optimal sample location to capture each bit. The next step is to set up the word boundary using custom logic to control the rx_bitslip_ctrl port on a channel-by-channel basis.

The bitslip circuit can be reset using the rx_bitslip_reset port. This circuit can be reset anytime and is not dependent on the PLL or DPA circuit operation.

Aligning the Word Boundaries

To align the word boundaries, it is useful to have control characters in the data stream so that your logic can have a known pattern to search for. You can compare the data received for each channel, compare to the control character you are looking for, then pulse the rx_bitslip_ctrl port as required until you successfully receive the control character.

Note: Altera recommends setting the bitslip rollover count to the deserialization factor or higher, which allows enough depth in the bitslip circuit to roll through an entire word if required.

If you do not have control characters in the received data, you need a deterministic relationship between the reference clock and data to predict the word boundary using timing simulation or laboratory measurements. This applies only for 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. For example, if the data

rate is 800 Mbps, and the deserialization factor is 8, the PLL requires a 100-MHz reference clock. 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. Using timing simulation, or lab measurements, monitor the parallel words received and determine how many pulses are required on the `rx_bitslip_ctrl` port to set your word boundaries. You can create a simple state machine to apply the required number of pulses when you enter user mode, or anytime you reset the PLL.

Note: When using 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. Thus, there can be a ± 1-bit of variance in the serial bit where the DPA initially locks. If there are no training patterns or control characters available in the serial bit stream to use for word alignment, Altera recommends using non-DPA mode.

Recommended Initialization and Reset Flow

Altera recommends that you follow these steps to initialize and reset the Altera LVDS SERDES megafunctions:

1. During entry into user mode, or anytime in user mode operation when the interface requires a reset, assert the `pll_areset` and `rx_dpa_reset` ports.
2. Deassert the `pll_areset` port and monitor the `pll_locked` port. For non-DPA mode, skip to **step 7**.
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.) Refer to the respective device data sheet for DPA lock time specifications.
5. Wait for the `rx_dpa_locked` port to assert.
6. Assert `rx_fifo_reset` for at least one parallel clock cycle, and then de-assert `rx_fifo_reset`.
7. Assert the `rx_bitslip_reset` port for at least one parallel clock cycle, and then deassert the `rx_bitslip_reset` port.
8. Begin word alignment by applying pulses as required to the `rx_bitslip_ctrl` port.
9. When the word boundaries are established on each channel, the interface is ready for operation.

Timing

The timing for Arria 10 devices is preliminary for the Quartus II software version 13.1. As a result, you should remove unoptimized periphery paths. An example **.sdc** file (**altera_lvds.sdc**) is generated as part of the design example. The following section is provided solely for information. Further documentation on setting constraints in the TimeQuest timing analyzer will be provided in future releases.

Receiver Skew Margin and Transmitter Channel-to-Channel Skew

Changes in the system environment, such as temperature, media (cable, connector, or PCB), and loading, affect the receiver's setup and hold times; internal skew affects the sampling ability of the receiver.

In non-DPA mode, use receiver skew margin (RKSM), receiver channel-to-channel skew (RCCS), and sampling window (SW) specifications to analyze the timing for high-speed source-synchronous differential signals in the receiver data path. The following equation shows the relationship between RSKM, RCCS, and SW.

Figure 4: RSKM

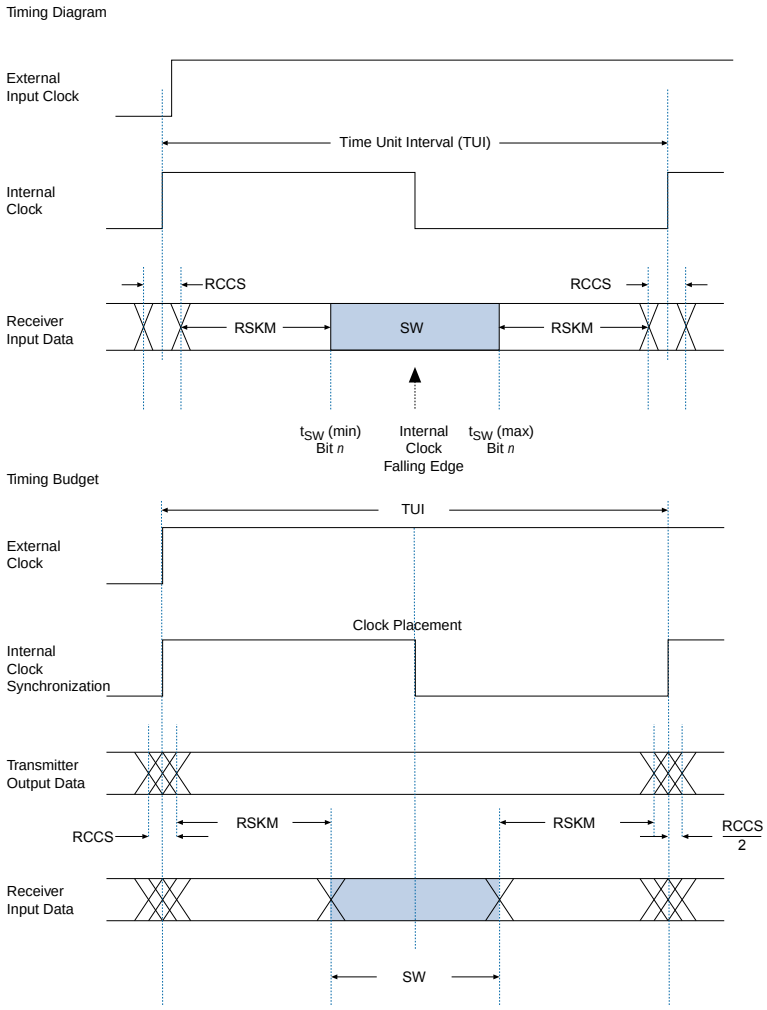
$$RSKM = \frac{TUI - SW - RCCS}{2}$$

Where:

- RSKM—is the timing margin between the receiver's clock input and the data input SW.
- Time unit interval (TUI)—is the time period of the serial data ($1/f_{MAX}$). Also known as the LVDS period in the TimeQuest Timing Analyzer section in the Quartus II Compilation Report.
- SW—is the period of time that the input data must be stable to ensure that data is successfully sampled by the LVDS receiver. The SW is a device property and varies with device speed grade.
- RCCS— is the timing difference between the fastest and slowest input transitions, including t_{CO} variations and clock skew. Specify RCCS by applying minimum and maximum `set_input_delay` constraints to the receiver inputs, where RCCS is the difference between the maximum and minimum value.

The following figure shows the relationship between the RSKM, RCCS, and SW.

Figure 5: Differential High-Speed Timing Diagram and Timing Budget for Non-DPA Mode



You must calculate the RSKM value to decide whether you can properly sample the data by the LVDS receiver with the given data rate and device. A positive RSKM value indicates the LVDS receiver can properly sample the data; a negative RSKM value indicates the receiver cannot properly sample the data.

The following example shows the RSKM calculation.

Data Rate: 1 Gbps, Board channel-to-channel skew = **200 ps**

RCCS = **100 ps** (pending characterization)

SW = **300 ps** (pending characterization)

TUI = **1000 ps**

Total RCCS = RCCS + Board channel-to-channel skew= 100 ps + 200 ps

= **300 ps**

RSKM= **TUI** - **SW** - **RCCS**

= **1000 ps - 300 ps - 300 ps**

= **400 ps > 0**

Because the RSKM > 0 ps, receiver non-DPA mode must work correctly.

Parameter Settings

You can parameterize the megafunctions using the MegaWizard® Plug-In Manager or the command-line interface (CLI).

Note: Altera recommends that you configure the megafunctions using the MegaWizard Plug-In Manager.

Table 3: Altera LVDS SERDES Parameter Settings

Tab	Parameter Settings	Value	Description
General Settings	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 • 1 to 12 for RX Soft-CDR	Number of serial channels in the interface.
	Data rate	150 to 1600.0 (functional mode dependent)	Data rate of each channel in Mbps.
	SERDES factor	3, 4, 5, 6, 7, 8, 9, and 10	Deserialization/serialization factor for the interface.
	Use clock-pin drive	—	When enabled, the megafunction bypasses the PLL and the interface is driven with a clock pin. **Note:** This feature is not supported in the current version of the Quartus II software.

Tab	Parameter Settings	Value	Description
PLL Settings	Use external PLL	—	When enabled, you must generate the Altera PLL megafunction to interface with the Altera LVDS SERDES megafunction. This option allows you to access all of the available clocks from the PLL, as well as use advanced PLL features such as clock switchover, bandwidth presets, dynamic phase stepping, and dynamic reconfiguration. **Note:** This feature is not supported in the current version of the Quartus II software.
	Desired inclock frequency	—	Allows you to specify `inclock` frequency in MHz
	Actual inclock frequency	—	Specifies the closest `inclock` frequency to the desired frequency that can source the interface.
	FPGA fabric speed grade	2 to 4	Allows you to specify the FPGA fabric speed grade which determines the operation range of the PLL.
	Enable pll_locked port	—	When enabled, the Altera LVDS SERDES megafunction enables the `pll_locked` port, which is asserted when the internal PLL locks onto the `inclock` signal.
	Enable pll_areset port	—	This required port enables you to reset the Altera LVDS SERDES megafunction interface.
	Core clock resource type	—	Specifies which clock network the Altera LVDS SERDES megafunction should export an internally generated coreclock onto. **Note:** This feature is not supported in the current version of the Quartus II software.

Tab	Parameter Settings	Value	Description
Receiver Settings	**Bitslip Settings**		
	Enable bitslip mode	—	When enabled, this parameter adds a bitslip block to the data path of the receiver. Every assertion of `rx_bitslip_ctrl` adds one bit of serial latency to the data path of the specified channel.
	Enable rx_bitslip_reset port	—	When enabled, user logic drives the `rx_bitslip_reset` port which you can use to reset the bitslip of each channel independently.
	Enable rx_bitslip_max port	—	When enabled, the Altera LVDS SERDES megafunction drives the `rx_bitslip_max` port. When asserted high, the next assertion of `rx_bitslip_ctrl` resets the serial latency of the bitslip to zero.
	Bitslip rollover value	3, 4, 5, 6, 7, 8, 9, 10, 11	Allows you to specify the depth of the bitslip block. Altera recommends setting this parameter to a value equal to or greater than the deserialization factor. The default value is 10.

Tab	Parameter Settings	Value	Description
Receiver Settings	**DPA Settings**		
	Enable rx_dpa_reset port	—	When enabled, the user logic drives the `rx_dpa_reset` port which you can use to reset DPA logic of each channel independently.
	Enable rx_dpa_locked port	—	When enabled, the Altera LVDS SERDES megafunction drives the `rx_dpa_locked` port. The DPA logic asserts the `rx_dpa_locked` signal when the signal settles on an ideal phase for that given channel. The `rx_dpa_locked` port will de-assert if the DPA moves two phases in the same direction. If Enable DPA loss of lock on one change is enabled, the `rx_dpa_locked` port will de-assert if the DPA moves one phase. The `rx_dpa_locked` will still toggle when `rx_dpa_hold` is asserted, and should be ignored by user logic when `rx_dpa_hold` is asserted.
	Enable rx_fifo_reset port	—	When enabled, user logic drives the `rx_fifo_reset` port which you can use to reset the DPA-FIFO block.
	Enable rx_dpa_hold port	—	When enabled, user logic drives the `rx_dpa_hold` port. Use this port to prevent the DPA from changing phase taps. Altera recommends using this port after DPA initially locks for interfaces that cannot meet the minimum required data transition density once DPA training is complete.

Tab	Parameter Settings	Value	Description
Receiver Settings	Enable DPA loss of lock on one change	—	When enabled, the Altera LVDS SERDES megafunction will drive the `rx_dpa_locked` signal low when the DPA changes phase selection from the initially locked position. The Altera LVDS SERDES megafunction will drive the `rx_dpa_locked` signal high if the DPA changes the phase selection back to the initial locked position. When disabled, the Altera LVDS SERDES megafunction will drive the `rx_dpa_locked` signal low when the DPA moves two phases in the same direction away from the initial locked position. The Altera LVDS SERDES megafunction will drive the `rx_dpa_locked` signal high if the DPA changes the phase selection to be within one phase or same phase as the initial locked position. A de-assertion of `rx_dpa_locked` does not indicate the data is invalid, it indicates the DPA has changed phase taps to track variations between the inclock and `rx_in` data. Altera recommends using data checkers to verify data accuracy.

Tab	Parameter Settings	Value	Description
Receiver Settings	Enable DPA alignment only to rising edges of data	—	When enabled, DPA logic counts the rising edges on the incoming serial data only. When disabled, DPA logic counts the rising and falling edges. **Note:** This port is only recommended for use in high jitter systems, and Altera recommends this port be disabled in typical applications.
	(Simulation only) Specify PPM drift on the recovered clock(s)	—	Specifies the amount of phase drift the ALTERA_LVDS simulation model should add to the recovered `rx_divfwdclks`. **Note:** This feature is not supported in the current version of the Quartus II software.
Receiver Settings	**Non-DPA Settings**		
	Desired receiver inclock phase shift (degrees):	—	Allows you to specify the phase relationship between the incoming serial data and `inclock` in degrees.
	Actual receiver inclock phase shift (degrees)	Legal values are dependent on the fclk and inclock frequencies. Refer to **Setting the Receiver Input Clock Parameters** on page 17.	Specifies the closest achievable receiver inclock phase shift to the desired receiver inclock phase shift.

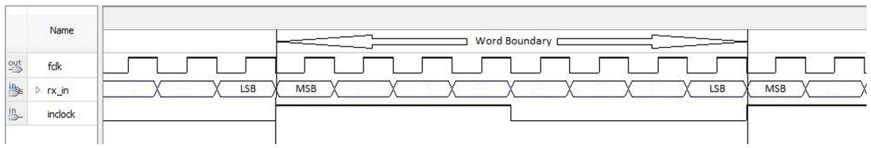
Tab	Parameter Settings	Value	Description
Transmitter Settings	Enable tx_coreclock port	—	When enabled, the Altera LVDS SERDES megafunction drives the `tx_coreclock` port which you can use to drive the core logic feeding the transmitter.
	Enable tx_outclock port	—	When enabled, the Altera LVDS SERDES megafunction drives the `tx_outclock` port through an LVDS transmitter. The tx_outclock frequency is dependent on the setting for the tx_outclock division factor parameter. The tx_outclock phase is dependent on the Desired tx_outclock phase shift parameter.
	Desired tx_outclock phase shift (degrees)	Refer to the **Setting the Transmitter Output Clock Parameters** on page 19.	Allows you to specify the phase relationship between the outclock and outgoing serial data in degrees.
	Actual tx_outclock phase shift (degrees)	Legal values are dependent on the fclk and tx_outclock frequencies. **Setting the Transmitter Output Clock Parameters** on page 19.	Specifies the closest achievable tx_outclock phase shift to the desired tx_outclock phase shift.
	Tx_outclock division factor	Legal values are dependent on the serialization factor.	Allows you to specify the ratio of the fast clock frequency to the `outclock` frequency (for example, the maximum number of serial transitions per outclock cycle).
Clock Resource Summary	Clock Parameters	—	Specifies the characteristics of all the clocks required by the currently specified interface.
	Clock Parameter Values	—	Specifies the values of each clock characteristics required by the currently specified interface.

Setting the Receiver Input Clock Parameters

When using non-DPA mode, in order for the source synchronous data to be properly sampled by the SERDES receiver, you must specify the inclock relationship to the `rx_in` data. To do so, type a value in the **Desired receiver inclock phase shift (degrees)** parameter. Legal values are evenly divisible by 45. If you enter an illegal value, the actual phase shift will appear in **Actual receiver inclock phase shift (degrees)**.

For rising inclock edge aligned interfaces to the `rx_in` data (**Figure 6**), select 0° as the desired receiver clock phase shift. The PLL will be set with the required phase shift on `fclk` to center it at the SERDES receiver.

Figure 6: 0° Edge Aligned inclock x8 Deserializer Waveform With Single Rate Clock



The phase shift you specify will be relative to the `fclk` which operates at the serial data rate. Phase shift values between 0° and 360° are used to specify the rising edge of the inclock within a single bit period. The maximum phase shift value is determined by the following equation:

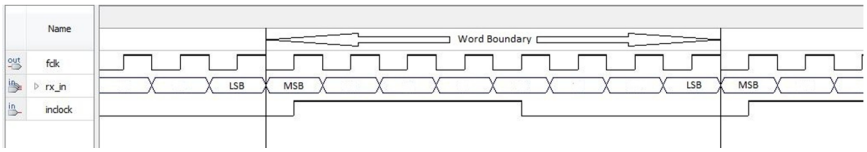
(Number of `fclk` periods per inclock period x 360) - 1

Specifying phase shift values greater than 360° will change the MSB location within the parallel data.

Note: By default, the MSB from the serial data will not be the MSB on the parallel data. You can use bitslip to set the proper word boundary on the parallel data. Refer to **Aligning the Word Boundaries** for more details.

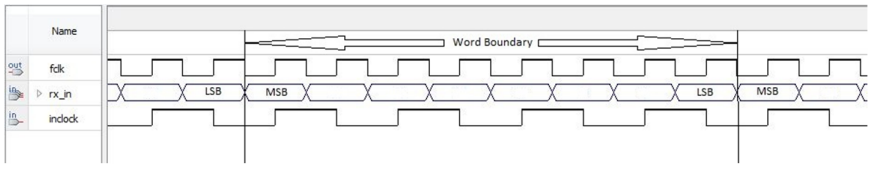
To specify a center aligned inclock to `rx_in` relationship (**Figure 7**), enter a phase shift value of 180° for the **Desired receiver inclock phase shift (degrees)** parameter.

Figure 7: 180° Center Aligned inclock x8 Deserializer Waveform With Single Rate Clock



The phase shift value you enter to specify the inclock to `rx_in` relationship is independent of the inclock frequency. To specify a center aligned DDR inclock to `rx_in` relationship (**Figure 8**), enter a phase shift value of 180° for the **Desired receiver inclock phase shift (degrees)** parameter.

Figure 8: 180° Center Aligned inclock x8 Deserializer Waveform With DDR Clock



Setting the Transmitter Output Clock Parameters

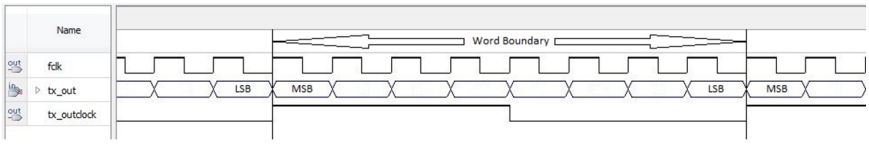
The `tx_outclock` relationship to the `tx_out` data is specified with two parameters:

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

These parameters set the phase and frequency of the `tx_outclock` based on the `fclk` 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 `fclk`. You can set the `tx_outclock` frequency using the available division factors from the drop-down list.

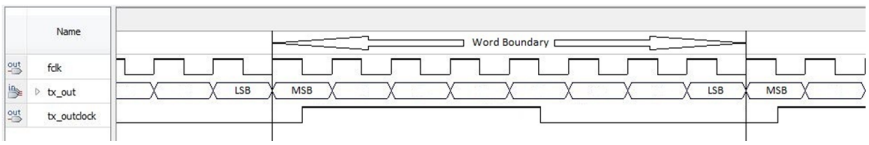
Use 0° to specify the `tx_outclock` phase to be rising edge aligned to the MSB of the serial data on `tx_out` (**Figure 9**).

Figure 9: 0° Edge Aligned tx_outclock x8 Serializer Waveform with Division Factor of 8



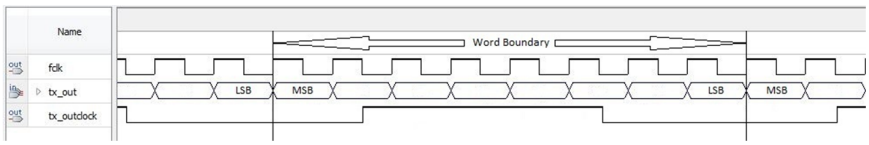
Use 180° to specify the `tx_outclock` phase to center aligned to the MSB of the serial data on `tx_out` (**Figure 10**).

Figure 10: 180° Center Aligned tx_outclock x8 Serializer Waveform with Division Factor of 8



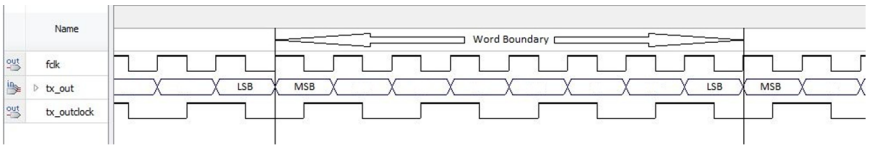
Phase shift values of 0° through 315° will position the rising edge of the `tx_outclock` within the MSB of the `tx_out` data. Phase shift values beginning with 360° will position the rising edge of the `tx_outclock` in serial bits after the MSB. The available number of 45° increment phase shift values you can enter for the **Desired tx_outclock phase shift (degrees)** parameter is equal to x8 serialization factor. For example, a phase shift of 540° will position the rising edge in the center of the bit after the MSB (**Figure 11**).

Figure 11: 540° Center Aligned tx_outclock x8 Serializer Waveform with Division Factor of 8



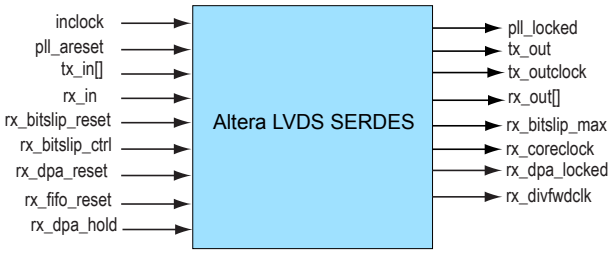
Use the **Tx_outclock division factor** drop-down list to set the `tx_outclock` frequency. **Figure 12** shows a x8 serialization factor using a 180° phase shift with a `tx_outclock` division factor of 2 (DDR clock and data relationship).

Figure 12: 180° Center Aligned tx_outclock x8 Serializer Waveform with Division Factor of 2



Ports

Figure 13: Altera LVDS SERDES Megafunction Ports



The following tables list the input and output ports for the Altera LVDS SERDES megafunction.

Note: *N* represents the LVDS interface width and the number of serial channels while *J* represents the SerDes factor of the interface.

Table 4: Common TX and RX Ports

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 Altera LVDS SERDES and PLL.
pll_locked	1	Output	Control	Asserted when internal PLL is locked.

Table 5: RX Ports

Signal Name	Width	Direc-tion	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 (bitslip).
rx_bitslip_ctrl	*N*	Input	Control	Positive-edge triggered increment for bitslip circuitry. Each assertion adds one bit of latency to the received bitstream.

Signal Name	Width	Direc-tion	Type	Description
rx_dpa_hold	*N*	Input	Control	Asynchronous, active-high signal prevents the DPA circuitry from switching to a new clock phase on the target channel. When held high, the selected channel(s) hold their current phase setting. When held low, the DPA block on selected channel(s) 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 and FIFO blocks. Minimum pulse width is 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 is one parallel clock period. Applicable in DPA-FIFO mode only.
rx_out	*N*J*	Output	Data	Receiver parallel data output. Synchronous to rx_coreclock in (DPA-FIFO and non-DPA modes). In soft-CDR mode, each channel has parallel data synchronous to its rx_divfwdclk.
rx_bitslip_max	*N*	Output	Control	Bitslip 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 (excluding soft-CDR) provided by the PLL. Not available when using an external PLL. **Note:** The external PLL feature is not supported in the current version of the Quartus II software.

Signal Name	Width	Direc-tion	Type	Description
rx_divfwdclk	*N*	Output	Clock	The per channel, divided clock with the ideal DPA phase. The recovered slow clock for a given channel. Applicable in soft-CDR mode only. Because each channel may have a different ideal sampling phase, the rx_divfwdclks may not be edge-aligned with each other. 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. Applicable in DPA-FIFO and soft-CDR modes only.

Table 6: TX Ports

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 via the TX data path). Source-synchronous with tx_out.
tx_coreclock	1	Output	Clock	The clock that drives the core logic feeding the serializer. Not available in the external PLL mode. **Note:** The external PLL feature is not supported in the current version of the Quartus II software.

Design Example

The Altera LVDS SERDES megafunction can generate a design example that matches the same configuration chosen for the megafunction. The design example is a simple design that does not target any specific application; however you can use the design example as a reference on how to instantiate the megafunction and what behavior to expect in a simulation.

Generating Design Example

During generation, the **Generation** dialog box displays the option to generate a design example. Turn on the **Generate Example Design** option.

The software generates the **<instance>_example_design** directory along with the megafunction, where **<instance>** is the name of your megafunction.

The **<instance>_example_design** directory contains two TCL scripts:

- `- make_qii_design.tcl`
- `- make_sim_design.tcl`

Generating Quartus Design Example

The `make_qii_design.tcl` generates a synthesizable design example along with a Quartus project, ready for compilation.

To generate synthesizable design example, run the following script at the end of IP generation:

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

To specify an exact device to use, run the following script:

```
quartus_sh -t make_qii_design.tcl [device_name]
```

This script generates a **qii** directory containing a project called **ed_synth.qpf**. You can open and compile this project with the Quartus II software.

Generating Simulation Design Example

The `make_sim_design.tcl` generates a simulation design example along with tool-specific scripts to compile and elaborate the necessary files.

To generate a simulation design example, run the following script at the end of the megafunction generation:

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

To generate simulation design example for a VHDL-only simulator, run the following script:

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

This script generates a **sim** directory containing one subdirectory for each supported simulation tools. Each subdirectory contains the specific scripts to run simulation with the corresponding tool.

The simulation design example is made of a driver connected to the generated megafunction. The driver generates random traffic and internally checks the legality of the outgoing data.

Document Revision History

The following table lists the revision history for this document.

Table 7: Document Revision History

Date	Version	Changes
November, 2013	2013.11.29	Initial release.