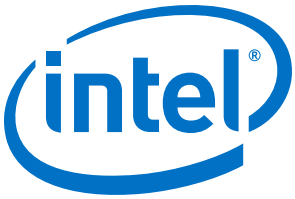


# Intel® Stratix® 10 High-Speed LVDS I/O User Guide

Updated for Intel® Quartus® Prime Design Suite: **19.4**

Subscribe
Send Feedback

**UG-S10LVDS | 2020.01.03**
Latest document on the web: PDF | HTML

# Contents

**1. Intel® Stratix® 10 High-Speed LVDS I/O Overview....4**
1.1. Intel Stratix 10 LVDS SERDES Usage Modes....4
1.2. Intel Stratix 10 LVDS Channels Support....5
1.3. Intel Stratix 10 GPIO Banks, SERDES, and DPA Locations....5
**2. Intel Stratix 10 High-Speed LVDS I/O Architecture and Features....7**
2.1. Intel Stratix 10 LVDS SERDES I/O Standards Support....7
2.2. LVDS Transmitter Programmable I/O Features....8
2.2.1. Programmable Pre-Emphasis....8
2.2.2. Programmable Differential Output Voltage....9
2.3. SERDES Circuitry....10
2.4. Differential Transmitter in Intel Stratix 10 Devices....11
2.4.1. Transmitter Blocks in Intel Stratix 10 Devices....11
2.4.2. Serializer Bypass for DDR and SDR Operations....11
2.5. Differential Receiver in Intel Stratix 10 Devices....12
2.5.1. Receiver Blocks in Intel Stratix 10 Devices....13
2.5.2. Receiver Modes in Intel Stratix 10 Devices....17
**3. Stratix 10 High-Speed LVDS I/O Design Considerations....20**
3.1. PLLs and Clocking for Intel Stratix 10 Devices....20
3.1.1. Clocking Differential Transmitters....20
3.1.2. Clocking Differential Receivers....21
3.1.3. Guideline: LVDS Reference Clock Source....22
3.1.4. Guideline: Use PLLs in Integer PLL Mode for LVDS....22
3.1.5. Guideline: Use High-Speed Clock from PLL to Clock LVDS SERDES Only....22
3.1.6. Guideline: Pin Placement for Differential Channels....22
3.1.7. LVDS Interface with External PLL Mode....25
3.2. Source-Synchronous Timing Budget....30
3.2.1. Differential Data Orientation....31
3.2.2. Differential I/O Bit Position....31
3.2.3. Transmitter Channel-to-Channel Skew....32
3.2.4. Receiver Skew Margin for Non-DPA Mode....33
3.3. Guideline: LVDS SERDES IP Core Instantiation....34
3.4. Guideline: LVDS SERDES Pin Pairs for Soft-CDR Mode....35
3.5. Guideline: LVDS Transmitters and Receivers in the Same I/O Bank....35
3.5.1. Using the Duplex Feature....35
3.5.2. Using an External PLL....36
3.6. Guideline: LVDS SERDES Limitation for Intel Stratix 10 GX 400, SX 400, and TX 400....37
**4. Intel Stratix 10 High-Speed LVDS I/O Implementation Guides....38**
4.1. LVDS SERDES Intel FPGA IP....38
4.1.1. LVDS SERDES IP Core Features....38
4.1.2. LVDS SERDES IP Core Functional Modes....39
4.1.3. LVDS SERDES IP Core Functional Description....40
4.2. LVDS SERDES IP Core Initialization and Reset....43
4.2.1. Initializing the LVDS SERDES IP Core in Non-DPA Mode....43
4.2.2. Initializing the LVDS SERDES IP Core in DPA Mode....43
4.2.3. Resetting the DPA....44

4.2.4. Word Boundaries Alignment....45
4.3. LVDS SERDES IP Core Timing....46
4.3.1. I/O Timing Analysis....47
4.3.2. FPGA Timing Analysis....48
4.3.3. Timing Analysis for the External PLL Mode....49
4.3.4. Timing Closure Guidelines for Internal FPGA Paths....49
4.3.5. Guideline: Use Clock Phase Alignment Block to Improve Timing Closure....49
4.4. LVDS SERDES IP Core Design Examples....50
4.4.1. LVDS SERDES IP Core Synthesizable Intel Quartus Prime Design Examples....50
4.4.2. LVDS SERDES IP Core Simulation Design Example....52
4.4.3. Combined LVDS SERDES IP Core Transmitter and Receiver Design Example....52
4.4.4. LVDS SERDES IP Core Dynamic Phase Shift Design Example....53
4.5. IP Migration Flow for Arria V, Cyclone V, and Stratix V Devices....54
4.5.1. Migrating Your ALTLVDS_TX and ALTLVDS_RX IP Cores....55

**5. LVDS SERDES Intel FPGA IP References....56**
5.1. LVDS SERDES IP Core Parameter Settings....56
5.1.1. LVDS SERDES IP Core General Settings....56
5.1.2. LVDS SERDES IP Core PLL Settings....57
5.1.3. LVDS SERDES IP Core Receiver Settings....58
5.1.4. LVDS SERDES IP Core Transmitter Settings....61
5.1.5. LVDS SERDES IP Core Clock Resource Summary....63
5.2. LVDS SERDES IP Core Signals....63
5.3. Comparison of LVDS SERDES Intel FPGA IP with Stratix V SERDES....66

**6. Intel Stratix 10 High-Speed LVDS I/O User Guide Archives....67**

**7. Document Revision History for the Intel Stratix 10 High-Speed LVDS I/O User Guide..68**

# 1. Intel® Stratix® 10 High-Speed LVDS I/O Overview

The Intel® Stratix® 10 device family supports high-speed LVDS protocols through the LVDS I/O banks, the LVDS SERDES Intel FPGA IP, and the GPIO Intel FPGA IP.

Intel Stratix 10 devices support LVDS on all LVDS I/O banks:

- All LVDS I/O banks support true LVDS input with $R_D$ OCT and true LVDS output buffer.
- The devices do not support emulated LVDS channels.
- The devices support true differential I/O reference clock for the I/O PLL that drives the serializer/deserializer (SERDES).
- You can use each LVDS I/O pins pair as LVDS receiver or LVDS transmitter.
- The LVDS SERDES IP core can place transmitter and receiver channels in the same I/O bank by using the **Duplex Feature** option.

### Related Information

- High-Speed I/O Specifications, Intel Stratix 10 Device Datasheet
  Lists the performance specifications of the SERDES in different modes.
- Document Revision History for the Intel Stratix 10 High-Speed LVDS I/O User Guide on page 68
- LVDS SERDES Intel FPGA IP on page 38
- GPIO Intel FPGA IP, Intel Stratix 10 General Purpose I/O User Guide
- Intel Stratix 10 High-Speed LVDS I/O User Guide Archives on page 67
  Provides a list of user guides for previous versions of the LVDS SERDES Intel FPGA IP.

## 1.1. Intel Stratix 10 LVDS SERDES Usage Modes

**Table 1. Usage Modes Summary of the Intel Stratix 10 LVDS SERDES**

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

| Usage Mode | Quick Guideline |
|---|---|
| Transmitter | In this mode, the SERDES block acts as a serializer. |
| DPA Receiver | • This mode is useful for source-synchronous clocking applications.<br>• The dynamic phase alignment block (DPA) automatically adjusts the clock phase to achieve optimal data-to-clock skew. |

*continued...*

| Usage Mode | Quick Guideline |
|---|---|
| Non-DPA Receiver | • This mode is useful for source-synchronous clocking applications.<br>• You must manage the data-to-clock skew. |
| Soft-CDR Receiver | • The soft clock data recovery (soft-CDR) mode is useful for asynchronous clocking applications.<br>• An asynchronous clock drives the LVDS SERDES IP core. The IP core outputs a recovered clock from the received data. |
| Bypass the SERDES | You can bypass the serializer to use SERDES factor of 2 by using the GPIO IP core:<br>• Single data rate (SDR) mode—you do not require clocks.<br>• Double data rate (DDR) mode—useful for slow source-synchronous clocking applications. |

## 1.2. Intel Stratix 10 LVDS Channels Support

The LVDS channels available vary among Intel Stratix 10 devices. In Intel Stratix 10 devices, you can use the LVDS I/O pin pairs as LVDS transmitter or receiver channels.

Refer to the Intel Stratix 10 device pin-out files for the LVDS channels counts.

### Related Information

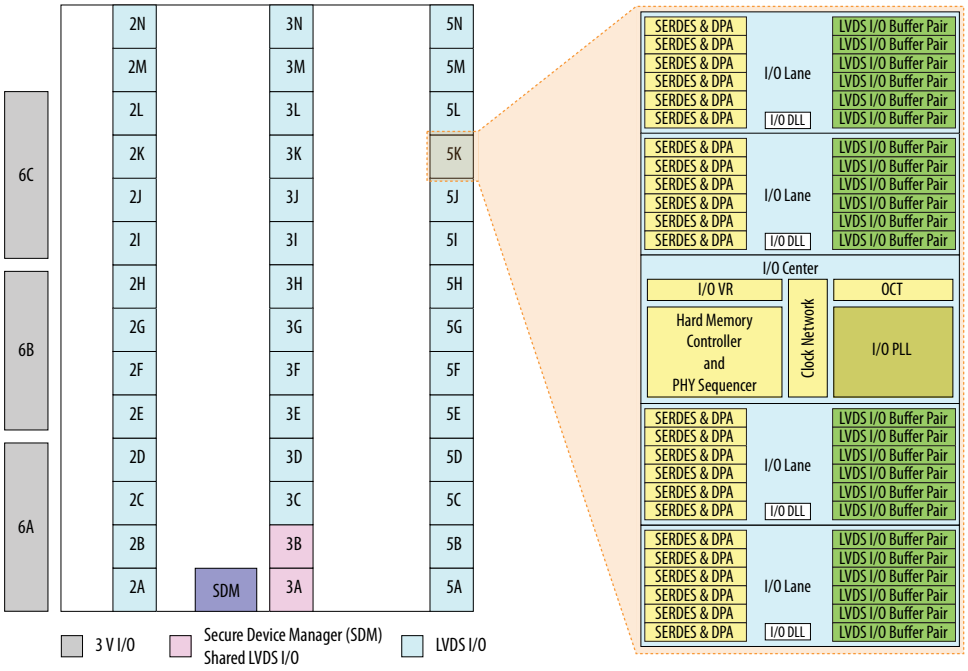
Intel Stratix 10 Device Pin-Out Files

## 1.3. Intel Stratix 10 GPIO Banks, SERDES, and DPA Locations

The I/O banks are located in I/O columns. Each I/O bank contains its own PLL, dynamic phase alignment (DPA), and SERDES circuitries.

**Figure 1. I/O Bank Structure with I/O PLL, DPA, and SERDES**

This figure shows an example of I/O banks in one Intel Stratix 10 device. The I/O banks availability and locations vary among Intel Stratix 10 devices.



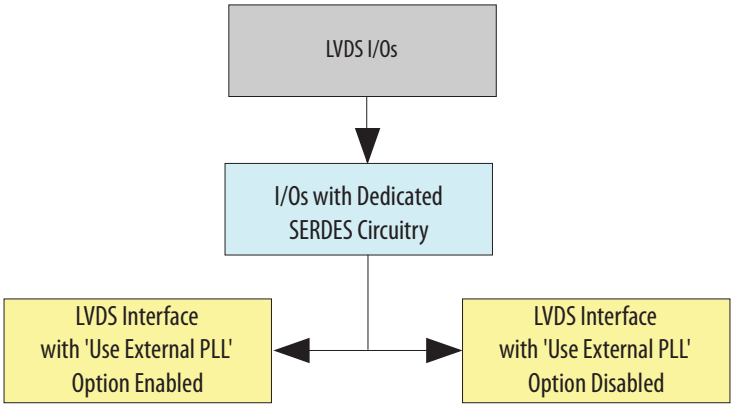
### Related Information

- Secure Device Manager
- Intel Stratix 10 Device Pin-Out Files
  Provides pin-out files that lists LVDS I/O banks locations and availability in different devices and packages.

# 2. Intel Stratix 10 High-Speed LVDS I/O Architecture and Features

The high-speed differential I/O interfaces and DPA features in Intel Stratix 10 devices provide advantages over single-ended I/Os and contribute to the achievable overall system bandwidth. Intel Stratix 10 devices support the LVDS, mini-LVDS, and reduced swing differential signaling (RSDS) differential I/O standards.

**Figure 2. I/O Bank Support for Dedicated SERDES Circuitry in Intel Stratix 10 Devices**




## 2.1. Intel Stratix 10 LVDS SERDES I/O Standards Support

**Table 2. Intel Stratix 10 SERDES Transmitter and Receiver High-Speed I/O Standards Support**

| I/O Standard | Intel Quartus® Prime Software I/O Assignment Value |
|---|---|
| True LVDS | LVDS |
| mini-LVDS | mini-LVDS |
| RSDS | RSDS |

## 2.2. LVDS Transmitter Programmable I/O Features

You can program some features of the I/O buffers according to your LVDS design requirements.

**Table 3. Summary of Supported Intel Stratix 10 LVDS Transmitter Programmable I/O Features and Settings**

| Feature | Setting | Assignment Name | Supported I/O Standards |
|---|---|---|---|
| Pre-Emphasis | 0 (disabled), 1 (enabled). Default is 1. | Programmable Pre-emphasis | • LVDS<br>• RSDS<br>• Mini-LVDS |
| Differential Output Voltage | 0 (low), 1 (medium low), 2 (medium high), 3 (high). Default is 2. | Programmable Differential Output Voltage ($V_{OD}$) | |

**Related Information**

High-Speed I/O Specifications, Intel Stratix 10 Device Datasheet
Lists the performance specifications of the SERDES in different modes.

### 2.2.1. Programmable Pre-Emphasis

The $V_{OD}$ setting and the output impedance of the driver set the output current limit of a high-speed transmission signal. At a high frequency, the slew rate may not be fast enough to reach the full $V_{OD}$ level before the next edge, producing pattern-dependent jitter. With pre-emphasis, the output current is boosted momentarily during switching to increase the output slew rate.

Pre-emphasis increases the amplitude of the high-frequency component of the output signal, and thus helps to compensate for the frequency-dependent attenuation along the transmission line. The overshoot introduced by the extra current happens only during a change of state switching to increase the output slew rate and does not ring, unlike the overshoot caused by signal reflection. The amount of pre-emphasis required depends on the attenuation of the high-frequency component along the transmission line.

**Figure 3. Programmable Pre-Emphasis**

This figure shows the LVDS output with pre-emphasis.

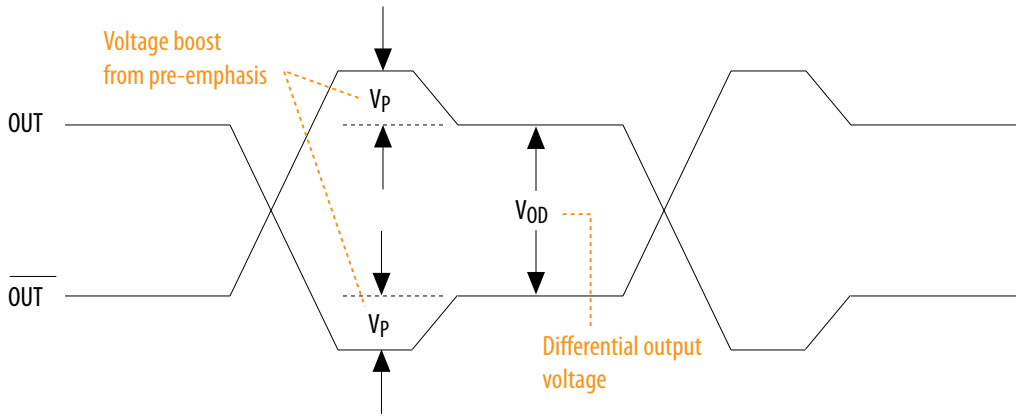


**Table 4. Intel Quartus Prime Software Assignment Editor—Programmable Pre-Emphasis**

This table lists the assignment name for programmable pre-emphasis and its possible values in the Intel Quartus Prime software Assignment Editor.

| Field | Assignment |
|---|---|
| To | `tx_out` |
| Assignment name | Programmable Pre-emphasis |
| Allowed values | 0 (disabled), 1 (enabled). Default is 1. |

## 2.2.2. Programmable Differential Output Voltage

The programmable $V_{OD}$ settings allow you to adjust the output eye opening to optimize the trace length and power consumption. A higher $V_{OD}$ swing improves voltage margins at the receiver end, and a smaller $V_{OD}$ swing reduces power consumption. You can statically adjust the $V_{OD}$ of the differential signal by changing the $V_{OD}$ settings in the Intel Quartus Prime software Assignment Editor.

**Figure 4. Differential $V_{OD}$**

This figure shows the $V_{OD}$ of the differential LVDS output.

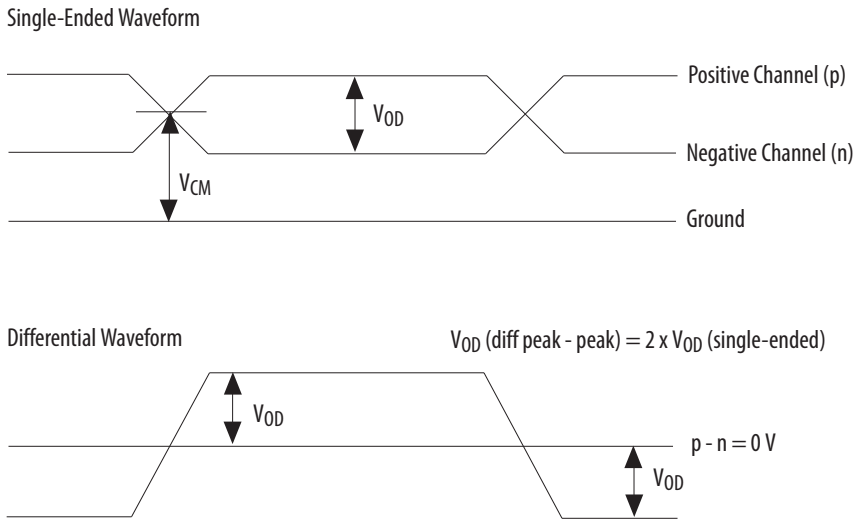



**Table 5. Intel Quartus Prime Software Assignment Editor—Programmable $V_{OD}$**

This table lists the assignment name for programmable $V_{OD}$ and its possible values in the Intel Quartus Prime software Assignment Editor.

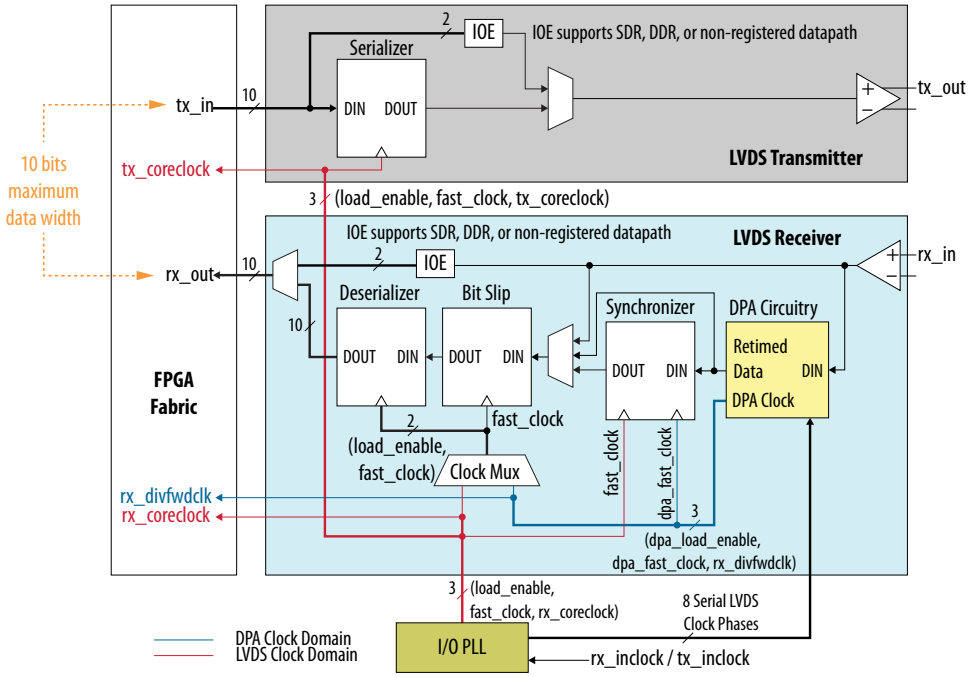
| Field | Assignment |
|---|---|
| To | `tx_out` |
| Assignment name | Programmable Differential Output Voltage ($V_{OD}$) |
| Allowed values | 0 (low), 1 (medium low), 2 (medium high), 3 (high). Default is 2. |

## 2.3. SERDES Circuitry

Each LVDS I/O channel in Intel Stratix 10 devices has built-in serializer/deserializer (SERDES) circuitry that supports high-speed LVDS interfaces. You can configure the SERDES circuitry to support source-synchronous communication protocols such as RapidIO®, XSBI, serial peripheral interface (SPI), and asynchronous protocols.

**Figure 5. SERDES**

This figure shows a transmitter and receiver block diagram for the LVDS SERDES circuitry with the interface signals of the transmitter and receiver data paths. The figure shows a shared PLL between the transmitter and receiver. If the transmitter and receiver do not share the same PLL, you require two I/O PLLs. In single data rate (SDR) and double data rate (DDR) modes, the data widths are 1 and 2 bits, respectively.




The LVDS SERDES Intel FPGA IP transmitter and receiver require various clock and load enable signals from an I/O PLL. The Intel Quartus Prime software configures the PLL settings automatically. The software is also responsible for generating the various clock and load enable signals based on the input reference clock and selected data rate.

*Note:* For the maximum data rate supported by the Intel Stratix 10 devices, refer to the device datasheet.

### Related Information

High-Speed I/O Specifications, Intel Stratix 10 Device Datasheet
Lists the performance specifications of the SERDES in different modes.

# 2.4. Differential Transmitter in Intel Stratix 10 Devices

**Table 6. Dedicated Circuitries and Features of the Differential Transmitter**

| Dedicated Circuitry / Feature | Description |
|---|---|
| Differential I/O buffer | Supports LVDS, mini-LVDS, and RSDS |
| SERDES | 3 to 10-bit wide serializer |
| Phase-locked loops (PLLs) | Clocks the load and shift registers |
| Programmable $V_{OD}$ | Static |
| Programmable pre-emphasis | Boosts output current |

**Related Information**

LVDS SERDES IP Core Signals on page 63

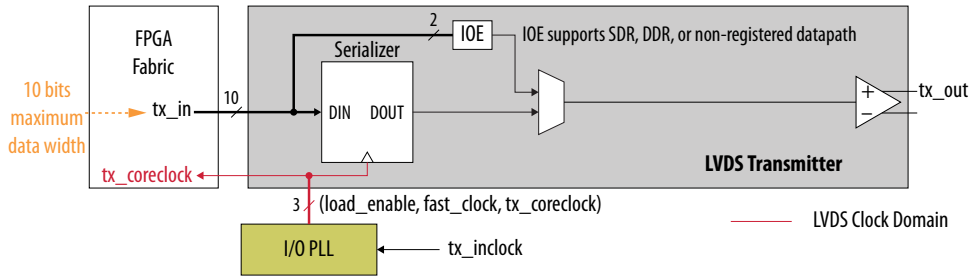
## 2.4.1. Transmitter Blocks in Intel Stratix 10 Devices

The dedicated circuitry consists of a true differential buffer, a serializer, and I/O PLLs that you can share between the transmitter and receiver. The serializer takes up to 10 bits wide parallel data from the FPGA fabric, clocks it into the load registers, and serializes it using shift registers that are clocked by the I/O PLL before sending the data to the differential buffer. The MSB of the parallel data is transmitted first.

*Note:* The PLL that drives the LVDS SERDES channel must operate in integer PLL mode. You do not need a PLL if you bypass the serializer.

**Figure 6. LVDS Transmitter**

This figure shows a block diagram of the transmitter. In SDR and DDR modes, the data width is 1 and 2 bits, respectively.




**Related Information**

- LVDS SERDES IP Core Signals on page 63
- Guideline: Use PLLs in Integer PLL Mode for LVDS on page 22

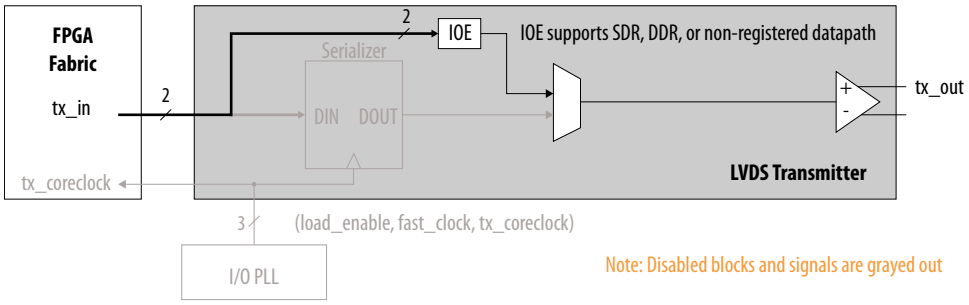
## 2.4.2. Serializer Bypass for DDR and SDR Operations

The I/O element (IOE) contains two data output registers that can each operate in either DDR or SDR mode.

You can bypass the serializer to support DDR (x2) and SDR (x1) operations to achieve a serialization factor of 2 and 1, respectively. The deserializer bypass is supported through the GPIO Intel FPGA IP.

**Figure 7. Serializer Bypass**

This figure shows the serializer bypass path.




- In SDR mode:
  - The IOE data width is 1 bit.
  - Registered output path requires a clock.
  - Data is passed directly through the IOE.
- In DDR mode:
  - The IOE data width is 2 bits.
  - The GPIO IP core requires a clock.
  - `tx_inclock` clocks the IOE register.

## 2.5. Differential Receiver in Intel Stratix 10 Devices

The receiver has a differential buffer and I/O PLLs that you can share among the transmitter and receiver, a DPA block, a synchronizer, a data realignment block, and a deserializer. The differential buffer can receive LVDS, mini-LVDS, and RSDS signal levels. You can statically set the I/O standard of the receiver pins to LVDS, mini-LVDS, or RSDS in the Intel Quartus Prime software Assignment Editor.

*Note:* The PLL that drives the LVDS SERDES channel must operate in integer PLL mode. You do not need a PLL if you bypass the deserializer

**Table 7. Dedicated Circuitries and Features of the Differential Receiver**

| Dedicated Circuitry / Feature | Description |
|---|---|
| Differential I/O buffer | Supports LVDS, mini-LVDS, and RSDS |
| SERDES | Up to 10-bit wide deserializer |
| Phase-locked loops (PLLs) | Generates different phases of a clock for data synchronizer |
| Data realignment (Bit slip) | Inserts bit latencies into serial data |
| DPA | Chooses a phase closest to the phase of the serial data |
| Synchronizer (FIFO buffer) | Compensate for phase differences between the data and the receiver's input reference clock |
| Skew adjustment | Manual |
| On-chip termination (OCT) | 100 Ω in LVDS I/O standards |

**Related Information**

Guideline: Use PLLs in Integer PLL Mode for LVDS on page 22

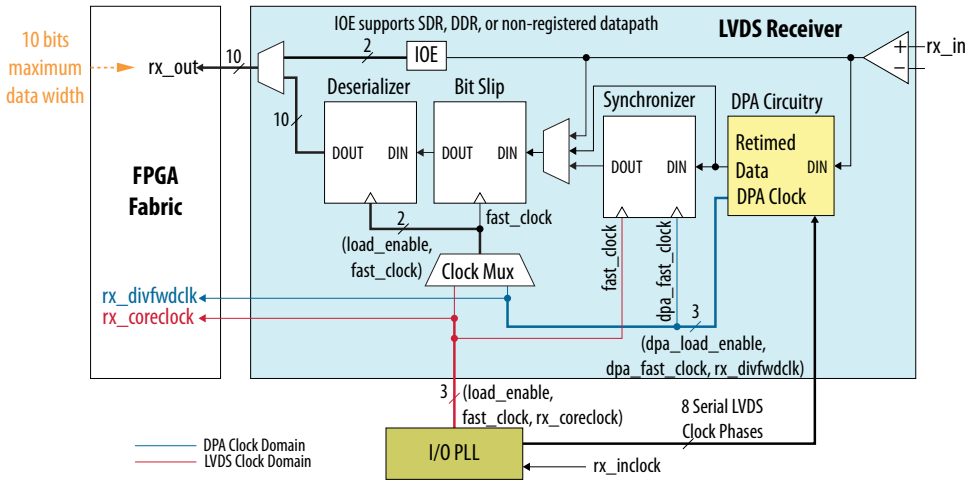
## 2.5.1. Receiver Blocks in Intel Stratix 10 Devices

The Intel Stratix 10 differential receiver has the following hardware blocks:

- DPA block
- Synchronizer
- Data realignment block (bit slip)
- Deserializer

**Figure 8. Receiver Block Diagram**

This figure shows the hardware blocks of the receiver. In SDR and DDR modes, the data width from the IOE is 1 and 2 bits, respectively. The deserializer includes shift registers and parallel load registers, and sends a maximum of 10 bits to the internal logic.



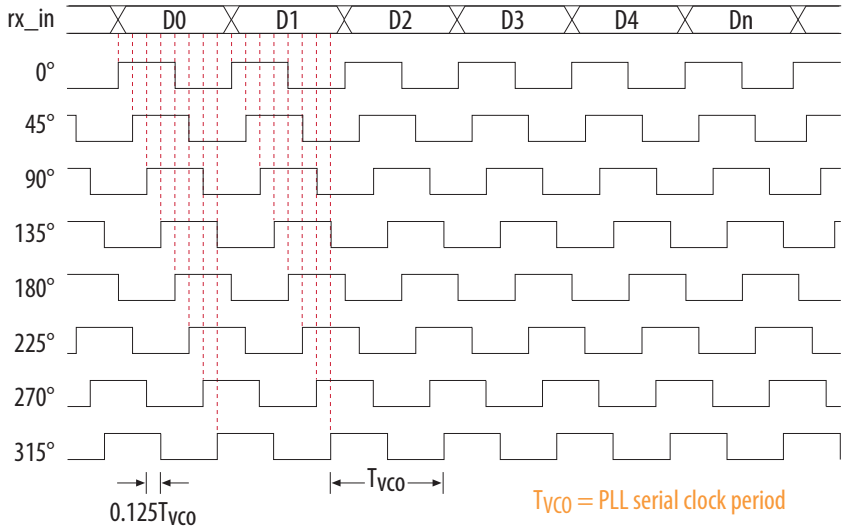

### 2.5.1.1. DPA Block

The DPA block takes in high-speed serial data from the differential input buffer and selects one of the eight phases that the I/O PLLs generate to sample the data. The DPA chooses a phase closest to the phase of the serial data. The maximum phase offset between the received data and the selected phase is 1/8 unit interval (UI)[(1)], which is the maximum quantization error of the DPA. The eight phases of the clock are equally divided, offering a 45° resolution.

(1) The unit interval is the period of the clock running at the serial data rate (fast clock).

**Figure 9. DPA Clock Phase to Serial Data Timing Relationship**

This figure shows the possible phase relationships between the DPA clocks and the incoming serial data.




The DPA block continuously monitors the phase of the incoming serial data and selects a new clock phase if it is required. You can prevent the DPA from selecting a new clock phase by asserting the optional `rx_dpa_hold` port, which is available for each channel.

DPA circuitry does not require a fixed training pattern to lock to the optimum phase out of the eight phases. After reset or power up, the DPA circuitry requires transitions on the received data to lock to the optimum phase. An optional output port, `rx_dpa_locked`, is available to indicate an initial DPA lock condition to the optimum phase after power up or reset. Use data checkers such as a cyclic redundancy check (CRC) or diagonal interleaved parity (DIP-4) to validate the data.

An independent reset port, `rx_dpa_reset`, is available to reset the DPA circuitry. You must retrain the DPA circuitry after reset.

*Note:* The DPA block is bypassed in non-DPA mode.

### 2.5.1.2. Synchronizer

The synchronizer is a one-bit wide and six-bit deep FIFO buffer that compensates for the phase difference between `dpa_fast_clock`—the optimal clock that the DPA block selects—and the `fast_clock` that the I/O PLLs produce. The synchronizer can only compensate for phase differences, not frequency differences, between the data and the receiver's input reference clock.

An optional port, `rx_fifo_reset`, is available to the internal logic to reset the synchronizer. The synchronizer is automatically reset when the DPA first locks to the incoming data. Intel recommends using `rx_fifo_reset` to reset the synchronizer when the data checker indicates that the received data is corrupted.

*Note:* The synchronizer circuit is bypassed in non-DPA and soft-CDR mode.

### 2.5.1.3. Data Realignment Block (Bit Slip)

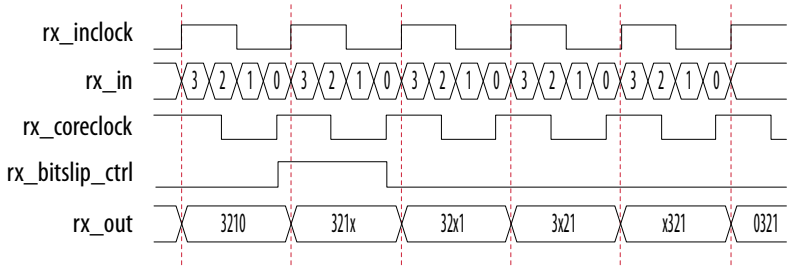
Skew in the transmitted data along with skew added by the link causes channel-to-channel skew on the received serial data streams. If you enable the DPA, the received data is captured with different clock phases on each channel. This difference may cause misalignment of the received data from channel to channel. To compensate for this channel-to-channel skew and establish the correct received word boundary at each channel, each receiver channel has a dedicated data realignment circuit that realigns the data by inserting bit latencies into the serial stream.

An optional `rx_bitslip_ctrl` port controls the bit insertion of each receiver independently controlled from the internal logic. The data slips one bit on the rising edge of `rx_bitslip_ctrl`. The requirements for the `rx_bitslip_ctrl` signal include the following items:

- The minimum pulse width is one period of the parallel clock in the logic array.
- The minimum low time between pulses is one period of the parallel clock.
- The signal is an edge-triggered signal.
- The valid data is available four parallel clock cycles after the rising edge of `rx_bitslip_ctrl`.

**Figure 10. Data Realignment Timing**

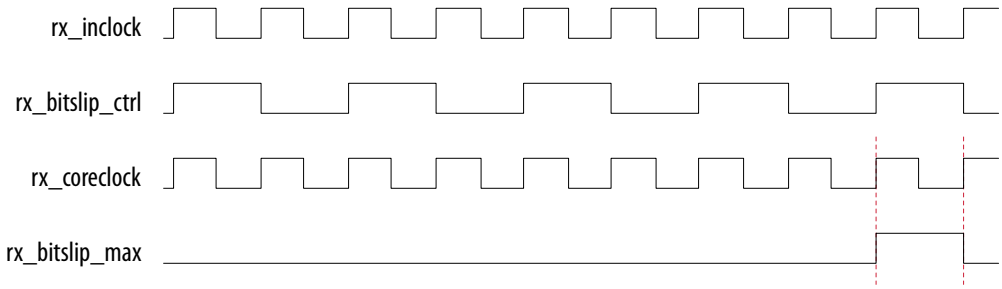
This figure shows receiver output (`rx_out`) after one bit slip pulse with the deserialization factor set to **4**.




The data realignment circuit has a bit slip rollover value set to the deserialization factor. An optional status port, `rx_bitslip_max`, is available to the FPGA fabric from each channel to indicate the reaching of the preset rollover point.

**Figure 11. Receiver Data Realignment Rollover**

This figure shows a preset value of four bit cycles before rollover occurs. The `rx_bitslip_max` signal pulses for one `rx_coreclock` cycle to indicate that rollover has occurred.



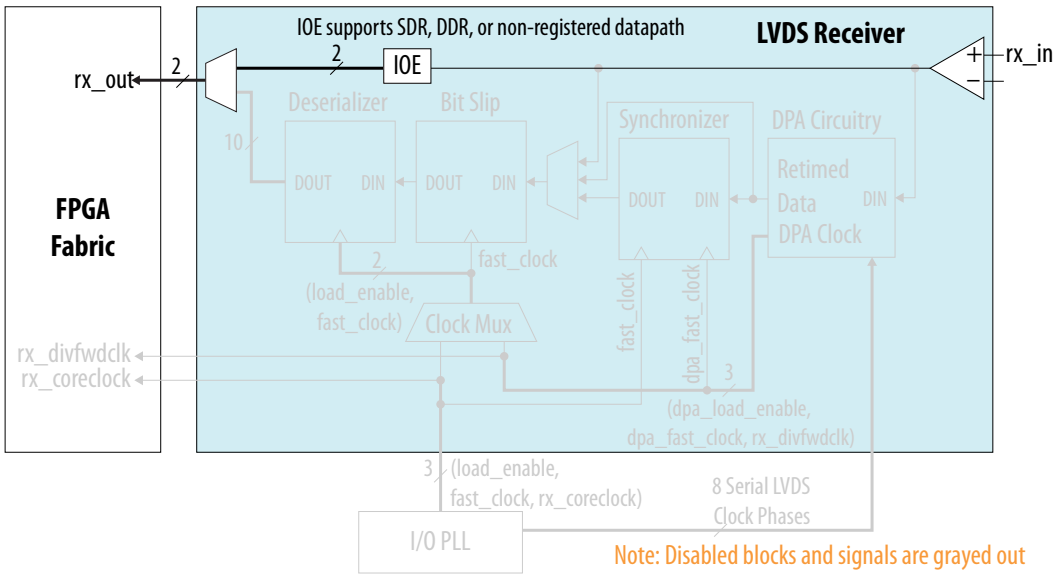
### 2.5.1.4. Deserializer

You can statically set the deserialization factor to x3, x4, x5, x6, x7, x8, x9, or x10 by using the Intel Quartus Prime software.

The IOE contains two data input registers that can operate in DDR or SDR mode. You can bypass the deserializer to support DDR (x2) and SDR (x1) operations. The deserializer bypass is supported through the GPIO IP core.

**Figure 12. Deserializer Bypass**

This figure shows the deserializer bypass path.




- If you bypass the deserializer in SDR mode:
  - The IOE data width is 1 bit.
  - Registered input path requires a clock.
  - Data is passed directly through the IOE.
- If you bypass the deserializer in DDR mode:
  - The IOE data width is 2 bits.
  - The GPIO IP core requires a clock.
  - `rx_inclock` clocks the IOE register. The clock must be synchronous to `rx_in`.
  - You must control the data-to-clock skew.

You cannot use the DPA and data realignment circuit when you bypass the deserializer.

## 2.5.2. Receiver Modes in Intel Stratix 10 Devices

The Intel Stratix 10 devices support the following receiver modes:

- Non-DPA mode
- DPA mode
- Soft-CDR mode

*Note:* If you use DPA mode, follow the recommended initialization and reset flow. The recommended flow ensures that the DPA circuit can detect the optimum phase tap from the PLL to capture data on the receiver.

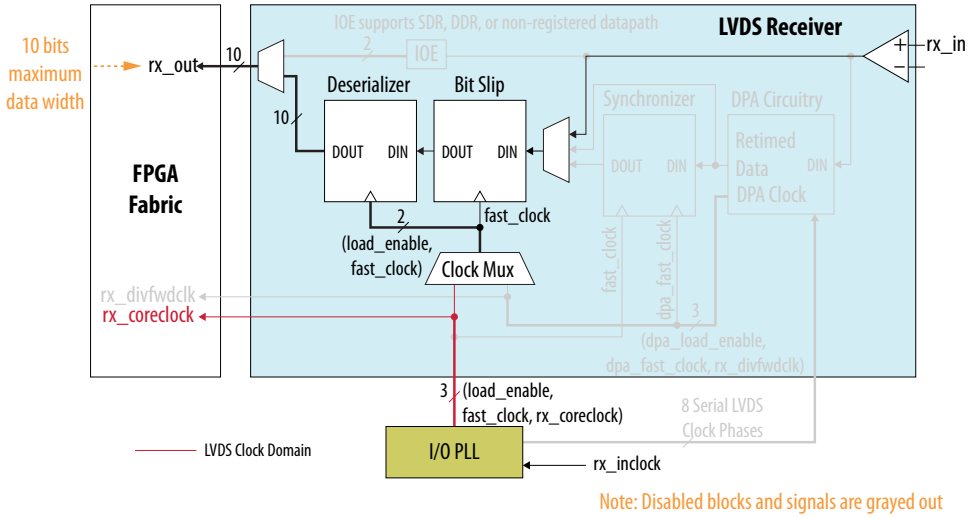
### 2.5.2.1. Non-DPA Mode

The non-DPA mode disables the DPA and synchronizer blocks. Input serial data is registered at the rising edge of the serial `fast_clock` clock that is produced by the I/O PLLs.

The `fast_clock` clock that is generated by the I/O PLLs clocks the data realignment and deserializer blocks.

**Figure 13. Receiver Datapath in Non-DPA Mode**

This figure shows the non-DPA datapath block diagram.

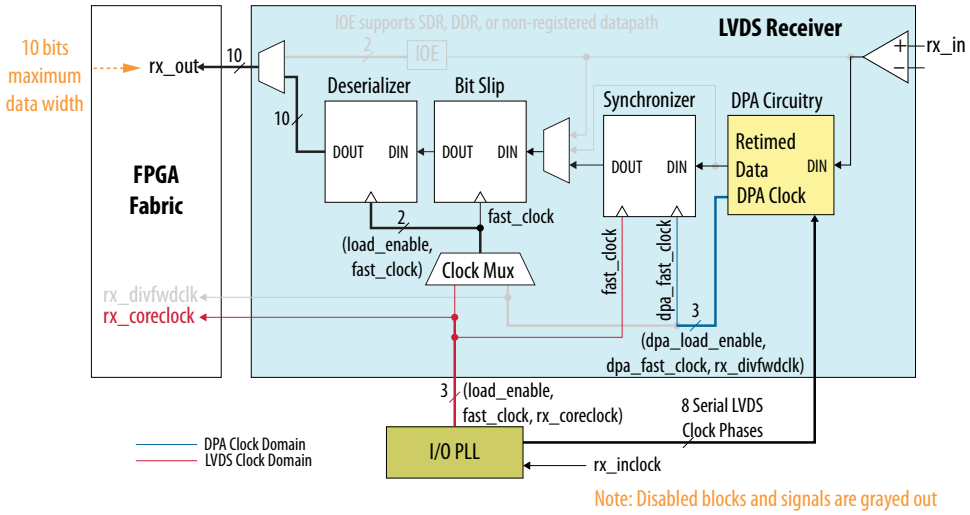



### 2.5.2.2. DPA Mode

The DPA block chooses the best possible clock (`dpa_fast_clock`) from the eight fast clocks that the I/O PLL sent. This serial `dpa_fast_clock` clock is used for writing the serial data into the synchronizer. A serial `fast_clock` clock is used for reading the serial data from the synchronizer. The same `fast_clock` clock is used in data realignment and deserializer blocks.

**Figure 14. Receiver Datapath in DPA Mode**

This figure shows the DPA mode datapath. In the figure, all the receiver hardware blocks are active.



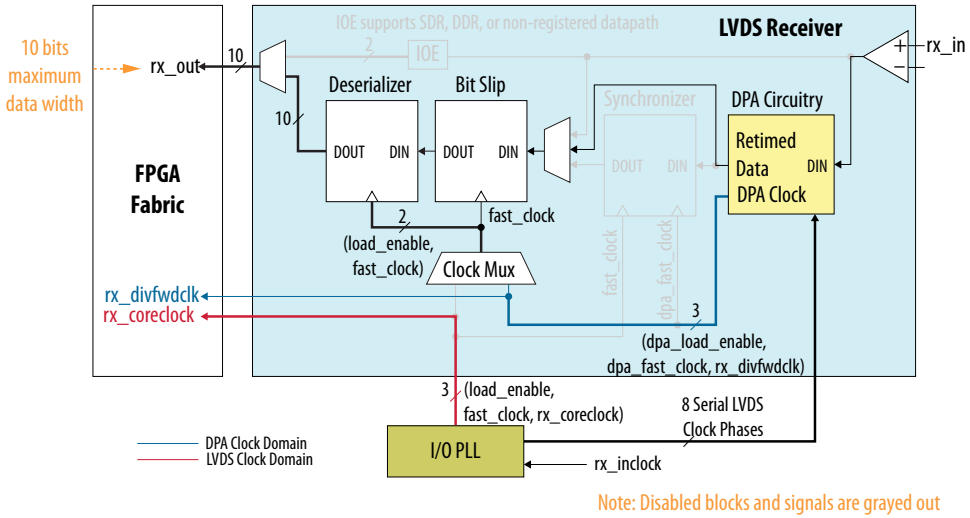

*Note:* In DPA mode, you must place all receiver channels of an LVDS instance in one I/O bank. Because each I/O bank has a maximum of 24 LVDS I/O buffer pairs, each LVDS instance can support a maximum of 24 DPA channels.

### 2.5.2.3. Soft-CDR Mode

The Intel Stratix 10 LVDS channel offers the soft-CDR mode to support the GbE and SGMII protocols. A receiver PLL uses the local clock source for reference.

**Figure 15. Receiver Datapath in Soft-CDR Mode**

This figure shows the soft-CDR mode datapath.



In soft-CDR mode, the synchronizer block is inactive. The DPA circuitry selects an optimal DPA clock phase to sample the data. This clock is used for bit slip operation and deserialization. The DPA block also forwards the selected DPA clock, divided by the deserialization factor called `rx_divfwdclk`, to the FPGA fabric, along with the deserialized data. This clock signal is put on the periphery clock (PCLK) network.

If you use the soft-CDR mode, do not assert the `rx_dpa_reset` port after the DPA has trained. The DPA continuously chooses new phase taps from the PLL to track parts per million (PPM) differences between the reference clock and incoming data.

You can use every LVDS channel in soft-CDR mode and drive the FPGA fabric using the PCLK network in the Intel Stratix 10 device family. In soft-CDR mode, the `rx_dpa_locked` signal is not valid because the DPA continuously changes its phase to track PPM differences between the upstream transmitter and the local receiver input reference clocks. However, you can use the `rx_dpa_locked` signal to determine the initial DPA locking conditions that indicate the DPA has selected the optimal phase tap to capture the data. The `rx_dpa_locked` signal is expected to deassert when operating in soft-CDR mode. The parallel clock, `rx_coreclock`, generated by the I/O PLLs, is also forwarded to the FPGA fabric.

*Note:* In soft-CDR mode, you must place all receiver channels of an LVDS instance in one I/O bank. Because each I/O bank has a maximum of 12 PCLK resources, each LVDS instance can support a maximum of 12 soft-CDR channels.

# 3. Stratix 10 High-Speed LVDS I/O Design Considerations

Follow the design considerations in this section when you are designing LVDS interfaces that use the SERDES circuitry in Intel Stratix 10 devices. Unless noted otherwise, these guidelines apply to all variants of the device family.

## 3.1. PLLs and Clocking for Intel Stratix 10 Devices

To generate the parallel clocks (`rx_coreclock` and `tx_coreclock`) and high-speed clocks (`fast_clock`), the Intel Stratix 10 devices provide I/O PLLs in the high-speed differential I/O receiver and transmitter channels.

### 3.1.1. Clocking Differential Transmitters

The I/O PLL generates the load enable (`load_enable`) signal and the `fast_clock` signal (the clock running at serial data rate) that clocks the load and shift registers. You can statically set the serialization factor to x3, x4, x5, x6, x7, x8, x9, or x10 using the Intel Quartus Prime software. The load enable signal is derived from the serialization factor setting.

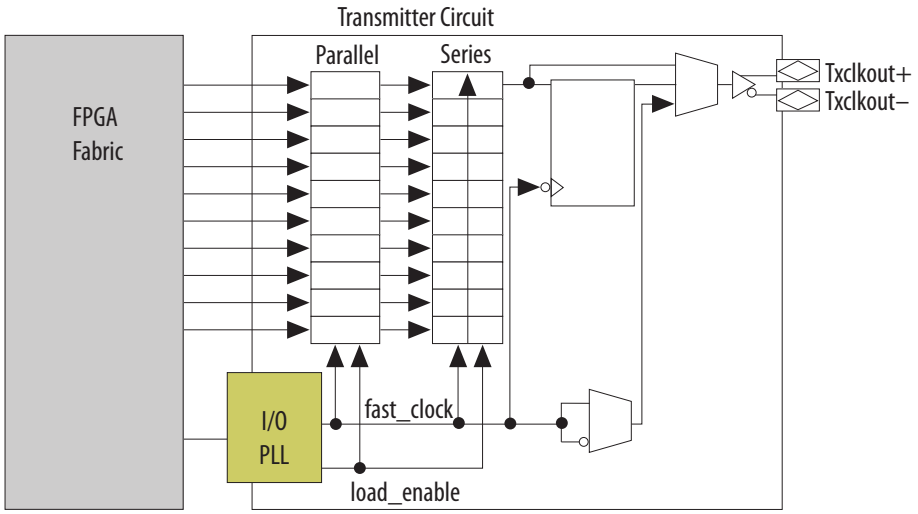
You can configure any Intel Stratix 10 transmitter data channel to generate a source-synchronous transmitter clock output. This flexibility allows the placement of the output clock near the data outputs to simplify board layout and reduce clock-to-data skew.

Different applications often require specific clock-to-data alignments or specific data-rate-to-clock-rate factors. You can specify these settings statically in the Intel Quartus Prime parameter editor:

- The transmitter can output a clock signal at the same rate as the data—with a maximum output clock frequency that each speed grade of the device supports.
- You can divide the output clock by a factor of 1, 2, 4, 6, 8, or 10, depending on the serialization factor.
- You can set the phase of the clock in relation to the data at 0° or 180° (edge- or center-aligned). The I/O PLLs provide additional support for other phase shifts in 45° increments.
- If the `tx_outclock` has a phase shift that is not a multiple of 180°, you can only place each LVDS SERDES Intel FPGA IP transmitter interface within a single I/O bank.

**Figure 16. Transmitter in Clock Output Mode**

This figure shows the transmitter in clock output mode. In clock output mode, you can use an LVDS channel as a clock output channel.




**Related Information**

LVDS SERDES IP Core Transmitter Settings on page 61

## 3.1.2. Clocking Differential Receivers

The I/O PLL receives the external clock input and generates different phases of the same clock. The DPA block automatically chooses one of the clocks from the I/O PLL and aligns the incoming data on each channel.

The synchronizer circuit is a 1-bit wide by 6-bit deep FIFO buffer that compensates for any phase difference between the DPA clock and the data realignment block. If necessary, the user-controlled data realignment circuitry inserts a single bit of latency in the serial bit stream to align to the word boundary. The deserializer includes shift registers and parallel load registers, and sends a maximum of 10 bits to the internal logic.

The physical medium connecting the transmitter and receiver LVDS channels may introduce skew between the serial data and the source-synchronous clock. The instantaneous skew between each LVDS channel and the clock also varies with the jitter on the data and clock signals as seen by the receiver. The three different modes —non-DPA, DPA, and soft-CDR—provide different options to overcome skew between the source synchronous clock (non-DPA, DPA) /reference clock (soft-CDR) and the serial data.

Non-DPA mode allows you to statically select the optimal phase between the source synchronous clock and the received serial data to compensate skew. In DPA mode, the DPA circuitry automatically chooses the best phase to compensate for the skew between the source synchronous clock and the received serial data. Soft-CDR mode provides opportunities for synchronous and asynchronous applications for chip-to-chip and short reach board-to-board applications for SGMII protocols.

*Note:* Only the non-DPA mode requires manual skew adjustment.

### 3.1.3. Guideline: LVDS Reference Clock Source

The LVDS SERDES IP core accepts two reference clock input sources. Whichever reference clock source you select, you must ensure timing closure.

**Table 8. LVDS Reference Clock Source**

| Reference Clock Input Source | Description | Reference Clock Promotion |
|---|---|---|
| Dedicated reference clock input within the same I/O bank. | This reference clock input source is the best choice to avoid performance and timing closure issues. | Do not manually promote the reference clock. |
| Reference clock input from other I/O banks. | This source must come from another I/O bank and not from other sources such as the hard processor system (HPS), IOPLL IP, or other IPs. | You must manually promote the reference clock. |

To manually promote the reference clock, include this statement in your Intel Quartus Prime settings file (`.qsf`):

```
set_instance_assignment -name GLOBAL_SIGNAL GLOBAL_CLOCK -to <name of top-level reference clock input port>
```

### 3.1.4. Guideline: Use PLLs in Integer PLL Mode for LVDS

Each I/O bank has its own PLL (I/O PLL) to drive the LVDS channels. These I/O PLLs operate in integer mode only.

### 3.1.5. Guideline: Use High-Speed Clock from PLL to Clock LVDS SERDES Only

The high-speed clock generated from the PLL is intended to clock the LVDS SERDES circuitry only. Do not use the high-speed clock to drive other logic because the allowed frequency to drive the core logic is restricted by the PLL $F_{OUT}$ specification.

For more information about the $F_{OUT}$ specification, refer to the device datasheet.

**Related Information**

PLL Specifications, Intel Stratix 10 Device Datasheet

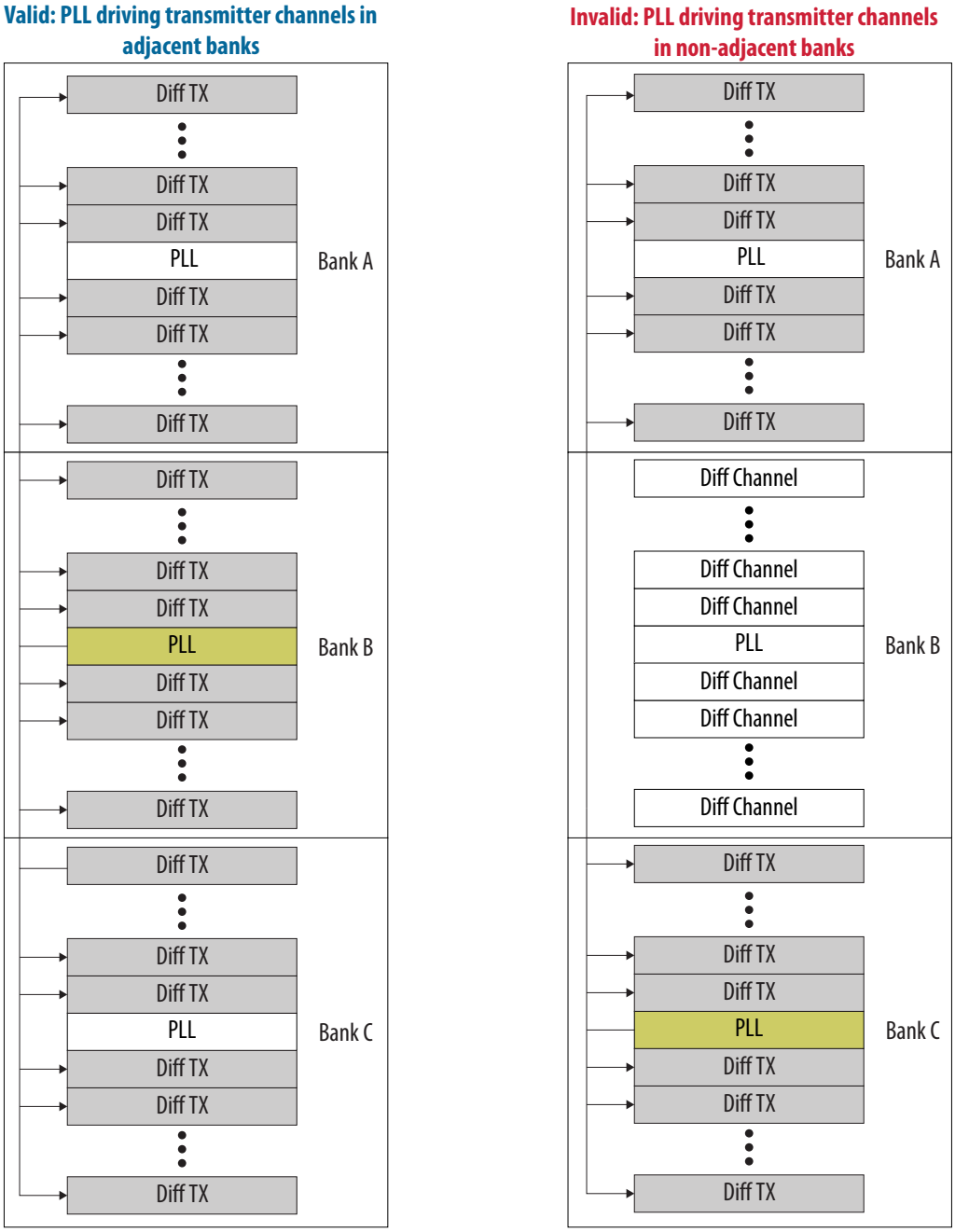
### 3.1.6. Guideline: Pin Placement for Differential Channels

Each I/O bank contains its own PLL. The I/O bank PLL can drive all receiver and transmitter channels in the same bank, and transmitter channels in adjacent I/O banks. However, the I/O bank PLL cannot drive receiver channels in another I/O bank or transmitter channels in non-adjacent I/O banks.

#### PLLs Driving Differential Transmitter Channels

For differential transmitters, the PLL can drive the differential transmitter channels in its own I/O bank and adjacent I/O banks. However, the PLL cannot drive the channels in a non-adjacent I/O bank.

**Figure 17. PLLs Driving Differential Transmitter Channels**



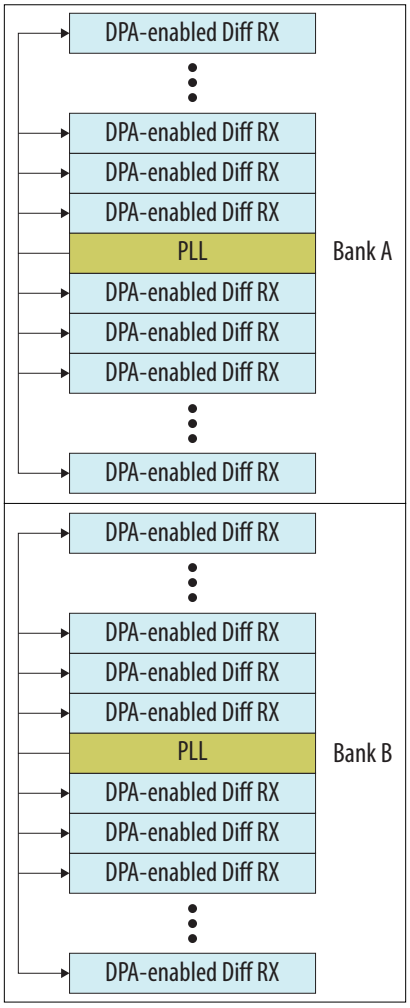

## PLLs Driving DPA-Enabled Differential Receiver Channels

For differential receivers, the PLL can drive all channels in the same I/O bank but cannot drive across banks.

Each differential receiver in an I/O bank has a dedicated DPA circuit to align the phase of the clock to the data phase of its associated channel. If you enable a DPA channel in a bank, you can assign the unused I/O pins in the bank to single-ended or differential I/O standards that has the same VCCIO voltage level used by the bank.

DPA usage adds some constraints to the placement of high-speed differential receiver channels. The Intel Quartus Prime compiler automatically checks the design and issues error messages if there are placement guidelines violations. Adhere to the guidelines to ensure proper high-speed I/O operation.

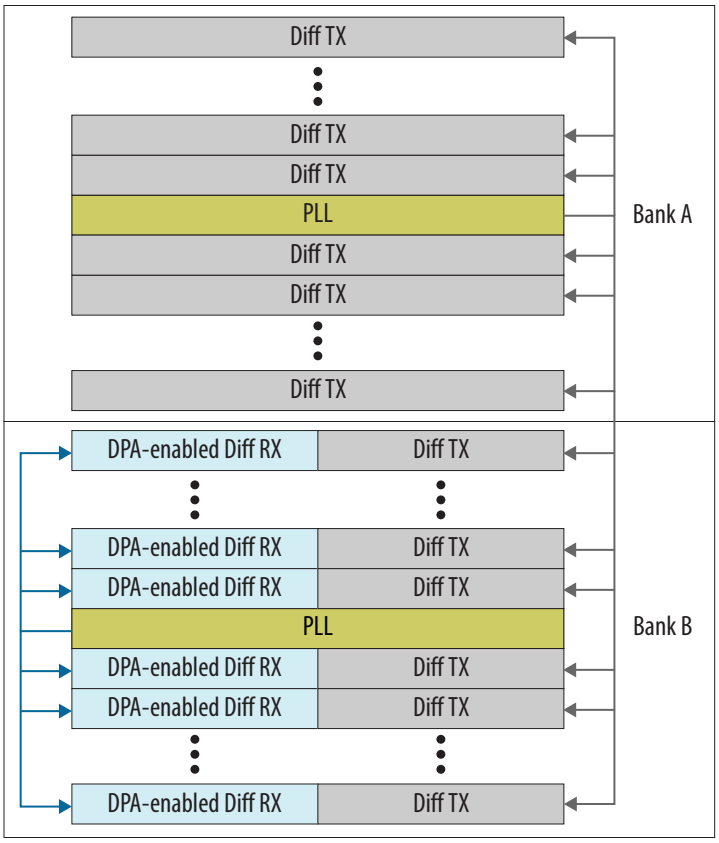
**Figure 18. PLLs Driving DPA-Enabled Differential Receiver Channels**




### PLLs Driving DPA-Enabled Differential Receiver and Transmitter Channels in LVDS Interface Spanning Multiple I/O Banks

If you use both differential transmitter and DPA-enabled receiver channels in a bank, the PLL can drive the transmitters spanning multiple adjacent I/O banks, but only the receivers in its own I/O bank.

**Figure 19. PLLs Driving DPA-Enabled Differential Receiver and Transmitter Channels Across I/O Banks**




## 3.1.7. 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 9. 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 |

*Note:* If you are using an external PLL for a wide transmitter interface that spans multiple I/O banks, only the second pair of clocks (indexed by "[1]") from the external PLL is valid.

**Related Information**

- Timing Analysis for the External PLL Mode on page 49
- Guideline: LVDS Transmitters and Receivers in the Same I/O Bank on page 35
- Combined LVDS SERDES IP Core Transmitter and Receiver Design Example on page 52
- LVDS SERDES IP Core PLL Settings on page 57

### 3.1.7.1. IOPLL IP Core Signal Interface with LVDS SERDES IP Core

**Table 10. 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 or receiver. The required signal interfaces differ if you turn on the Clock Phase Alignment (CPA) feature of the LVDS SERDES IP core.

| From the IOPLL IP core | To the LVDS SERDES IP core transmitter or receiver | |
|---|---|---|
| | Without CPA | With CPA |
| `lvds_clk[0]` (serial clock output signal)<br>• Configure this signal using `outclk0` in the PLL.<br>• 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**.<br>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. | `ext_fclk` (serial clock input to the transmitter or receiver) | `ext_fclk` (serial clock input to the transmitter or receiver) |
| `loaden[0]` (load enable output)<br>• Configure this signal using `outclk1` in the PLL.<br>• 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 or receiver)<br>This signal is not required for LVDS receiver in soft-CDR mode. | `ext_loaden` (load enable to the transmitter or receiver)<br>This signal is not required for LVDS receiver in soft-CDR mode. |

*continued...*

| From the IOPLL IP core | To the LVDS SERDES IP core transmitter or receiver | |
|---|---|---|
| | Without CPA | With CPA |
| `outclk4` (parallel clock output)<br>This clock is not required if you turn on **Use the CPA block for improved periphery-core timing**. | `ext_coreclock` (parallel core clock) | — |
| `locked` | — | `ext_pll_locked` |
| `reset` | `pll_areset` (asynchronous PLL reset port) | `pll_areset` (asynchronous PLL reset port) |
| `phout[7:0]`<br>• This signal is required if `ext_vcoph[7:0]` is required.<br>• Configure this signal by turning on **Specify VCO frequency** in the PLL and specifying the **VCO frequency** value.<br>• Turn on **Enable access to PLL DPA output port**. | `ext_vcoph[7:0]`<br>This signal is required only for LVDS receiver in DPA or soft-CDR mode. | `ext_vcoph[7:0]`<br>This signal is required for all transmitter or receiver modes. |

**Related Information**

Clock Phase Alignment on page 42
Provides more information about the CPA feature of the LVDS SERDES IP core, its required conditions, and the resultant core clock duty cycles.

### 3.1.7.2. 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 11. 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) | outclk4 (2) (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<br>(outclk0 phase shift divided by the serialization factor) |
| Duty cycle | 50% | 100/serialization factor | 50% |

(2) Not required if your turn on **Use the CPA block for improved periphery-core timing**.

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 20. Phase Relationship for External PLL Interface Signals**

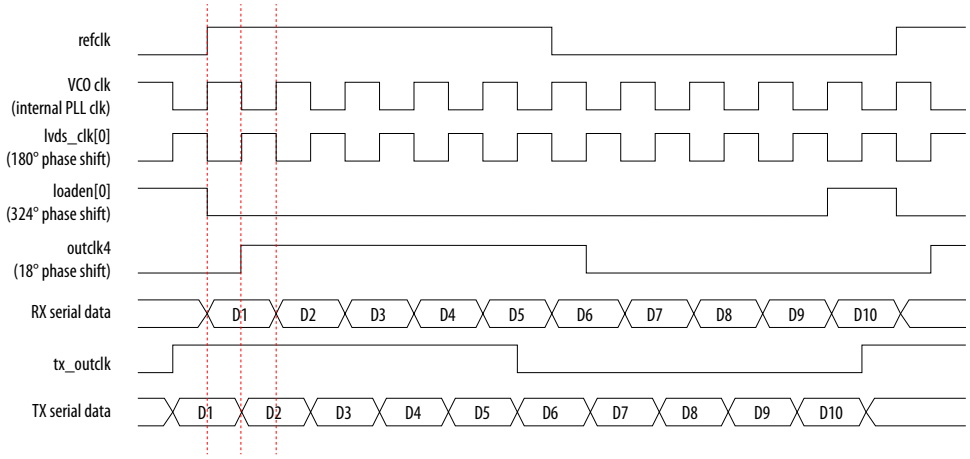



**Table 12. Example: Generating Output Clocks Using an IOPLL IP core (Receiver in DPA or Soft-CDR Mode)**

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.

| 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. | outclk4[2] (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 13. 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.

| 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(2) (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% | — |

**Related Information**

- LVDS SERDES IP Core PLL Settings on page 57
- LVDS SERDES IP Core Signals on page 63
- LVDS SERDES IP Core Clock Resource Summary on page 63
- LVDS SERDES IP Core Clock Resource Summary on page 63

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

**Figure 21. Non-DPA or DPA LVDS Receiver Interface with the IOPLL IP Core in External PLL Mode**

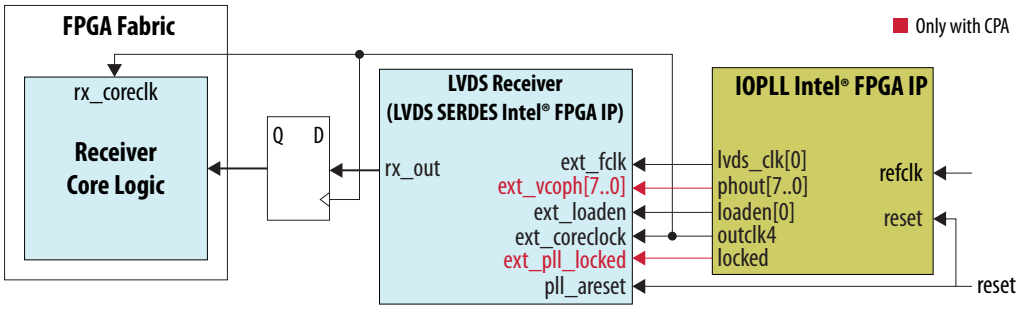


**Figure 22. Soft-CDR LVDS Receiver Interface with the IOPLL IP Core in External PLL Mode**

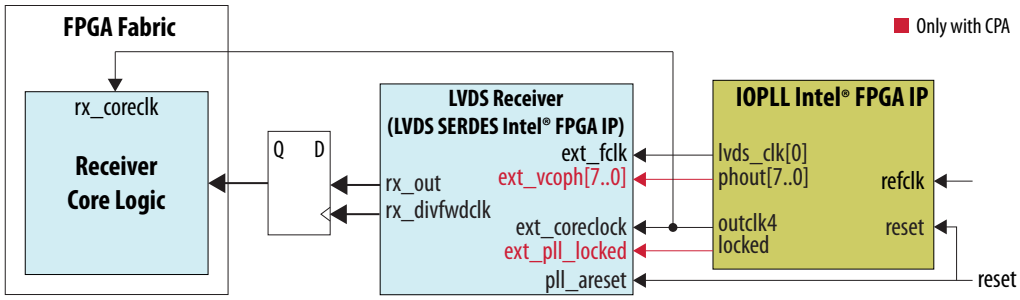
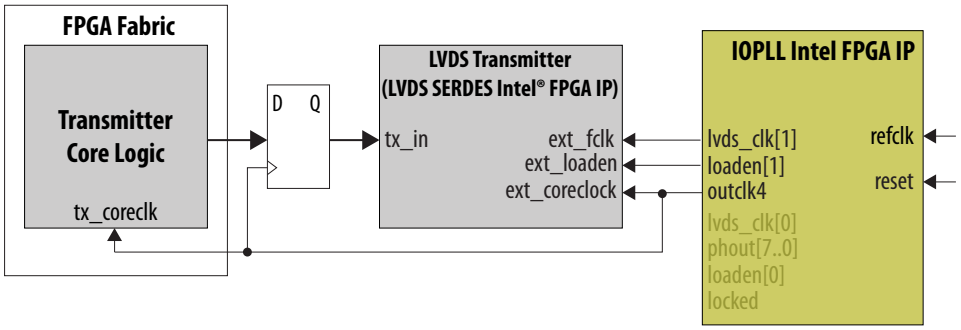



**Figure 23. LVDS Transmitter Interface with the IOPLL IP Core in External PLL Mode**

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.

## 3.2. Source-Synchronous Timing Budget

The topics in this section describe the timing budget, waveforms, and specifications for source-synchronous signaling in the Intel Stratix 10 device family.

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.

The basis of the source synchronous timing analysis is the skew between the data and the clock signals instead of the clock-to-output setup times. High-speed differential data transmission requires the use of timing parameters provided by IC vendors and is strongly influenced by board skew, cable skew, and clock jitter.

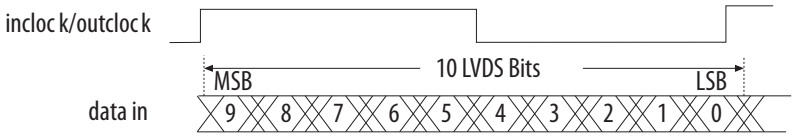
This section defines the source-synchronous differential data orientation timing parameters, the timing budget definitions for the Intel Stratix 10 device family, and how to use these timing parameters to determine the maximum performance of a design.

## 3.2.1. Differential Data Orientation

There is a set relationship between an external clock and the incoming data. For operations at 1 Gbps and a serialization factor of 10, the external clock is multiplied by 10. You can set phase-alignment in the PLL to coincide with the sampling window of each data bit. The data is sampled on the falling edge of the multiplied clock.

**Figure 24. Bit Orientation in the Intel Quartus Prime Software**

This figure shows the data bit orientation of the x10 mode.



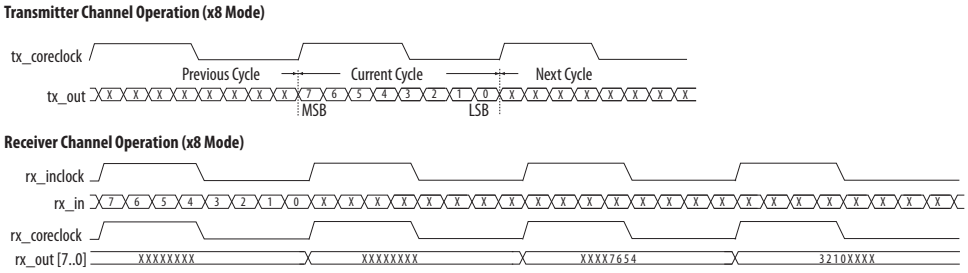

## 3.2.2. Differential I/O Bit Position

Data synchronization is necessary for successful data transmission at high frequencies.

**Figure 25. Bit-Order and Word Boundary for One Differential Channel**

This figure shows the data bit orientation for a channel operation and is based on the following conditions:

- The serialization factor is equal to the clock multiplication factor.
- The phase alignment uses edge alignment.
- The operation is implemented in hard SERDES.




Note: These waveforms are only functional waveforms and do not convey timing information

For other serialization factors, use the Intel Quartus Prime software tools to find the bit position within the word.

### 3.2.2.1. Differential Bit Naming Conventions

**Table 14. Differential Bit Naming**

This table lists the conventions for differential bit naming for 18 differential channels. The MSB and LSB positions increase with the number of channels used in a system.

| Receiver Channel Data Number | Internal 8-Bit Parallel Data | |
|---|---|---|
| | MSB Position | LSB Position |
| 1 | 7 | 0 |
| 2 | 15 | 8 |
| 3 | 23 | 16 |

*continued...*

| Receiver Channel Data Number | Internal 8-Bit Parallel Data | |
|---|---|---|
| | MSB Position | LSB Position |
| 4 | 31 | 24 |
| 5 | 39 | 32 |
| 6 | 47 | 40 |
| 7 | 55 | 48 |
| 8 | 63 | 56 |
| 9 | 71 | 64 |
| 10 | 79 | 72 |
| 11 | 87 | 80 |
| 12 | 95 | 88 |
| 13 | 103 | 96 |
| 14 | 111 | 104 |
| 15 | 119 | 112 |
| 16 | 127 | 120 |
| 17 | 135 | 128 |
| 18 | 143 | 136 |

## 3.2.3. Transmitter Channel-to-Channel Skew

The receiver skew margin calculation uses the transmitter channel-to-channel skew (TCCS)—an important parameter based on the Intel Stratix 10 transmitter in a source-synchronous differential interface:

- TCCS is the difference between the fastest and slowest data output transitions, including the $T_{CO}$ variation and clock skew.
- For LVDS transmitters, the Timing Analyzer provides the 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.

For the Intel Stratix 10 devices, perform PCB trace compensation to adjust the trace length of each LVDS channel to improve channel-to-channel skew when interfacing with non-DPA receivers at data rate above 840 Mbps. The Intel Quartus Prime software Fitter Report panel reports the amount of delay you must add to each trace for the Intel Stratix 10 device. You can use the recommended trace delay numbers published under the LVDS Transmitter/Receiver Package Skew Compensation panel and manually compensate the skew on the PCB board trace to reduce channel-to-channel skew, thus meeting the timing budget between LVDS channels.

### Related Information

Obtaining TCCS Report on page 48

## 3.2.4. Receiver Skew Margin for Non-DPA Mode

Different modes of LVDS receivers use different specifications, which can help in deciding the ability to sample the received serial data correctly.

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

**Related Information**

- I/O Timing Analysis on page 47
- Obtaining RSKM Report on page 48
- Obtaining TCCS Report on page 48

### 3.2.4.1. RSKM Equation

The RSKM equation expresses the relationship between RSKM, TCCS, and SW.

**Figure 26. RSKM Equation**

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

Conventions used for the equation:

- RSKM—the timing margin between the clock input of the receiver and the data input sampling window, and the jitter induced from core noise and I/O switching noise.
- Time unit interval (TUI)—time period of the serial data.
- SW—the period of time that the input data must be stable to ensure that the LVDS receiver samples the data successfully. The SW is a device property and varies according to device speed grade.
- TCCS—the timing difference between the fastest and the slowest output edges across channels driven by the same PLL. The TCCS measurement includes the $t_{CO}$ variation, clock, and clock skew.

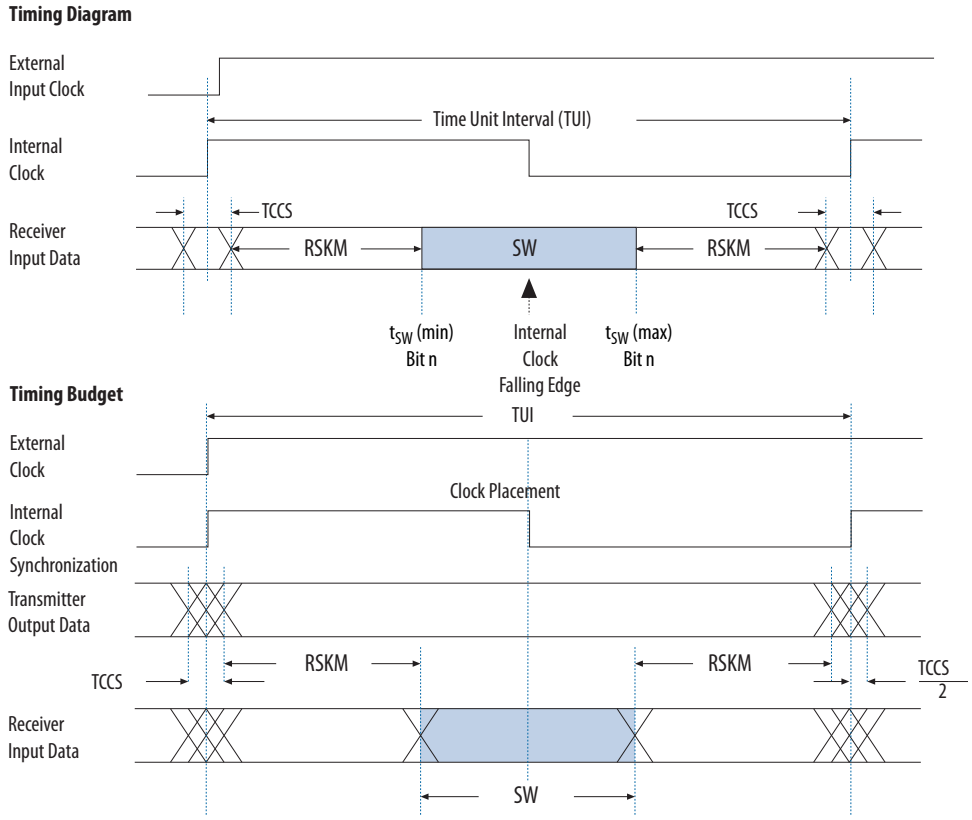
*Note:* If there is additional board channel-to-channel skew, consider the total receiver channel-to-channel skew (RCCS) instead of TCCS. Total RCCS = TCCS + board channel-to-channel skew.

You must calculate the RSKM value, based on the data rate and device, to determine if the LVDS receiver can sample the data:

- A positive RSKM value, after deducting transmitter jitter, indicates that the LVDS receiver can sample the data properly.
- A negative RSKM value, after deducting transmitter jitter, indicates that the LVDS receiver cannot sample the data properly.

**Figure 27. Differential High-Speed Timing Diagram and Timing Budget for Non-DPA Mode**

This figure shows the relationship between the RSKM, TCCS, and the SW of the receiver.




### 3.2.4.2. Example: RSKM Calculation

This example shows the RSKM calculation for FPGA devices at 1 Gbps data rate with a 200 ps board channel-to-channel skew.

- TCCS = 100 ps
- SW = 300 ps
- TUI = 1000 ps
- Total RCCS = TCCS + Board channel-to-channel skew = 100 ps + 200 ps = 300 ps
- RSKM = (TUI – SW – RCCS) / 2 = (1000 ps – 300 ps – 300 ps) / 2 = 200 ps

If the RSKM is greater than 0 ps after deducting transmitter jitter, the non-DPA receiver will work correctly.

## 3.3. Guideline: LVDS SERDES IP Core Instantiation

For all LVDS SERDES IP core functional modes, if you turn on the **Use External PLL** option, you can instantiate multiple LVDS SERDES IP core instances per I/O bank. Otherwise, using the LVDS SERDES IP core internal PLL, you can instantiate only one LVDS SERDES IP core instance in each I/O bank.

## 3.4. Guideline: LVDS SERDES Pin Pairs for Soft-CDR Mode

You can use only specific LVDS pin pairs in soft-CDR mode. Refer to the pinout file of each device to determine the LVDS pin pairs that support the soft-CDR mode.

**Related Information**

Intel Stratix 10 Device Pin-Out Files

## 3.5. Guideline: 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 turn on the duplex feature of the LVDS SERDES IP core or use an external PLL.

**Related Information**

- LVDS Interface with External PLL Mode on page 25
- Combined LVDS SERDES IP Core Transmitter and Receiver Design Example on page 52
- LVDS SERDES IP Core PLL Settings on page 57

### 3.5.1. Using the Duplex Feature

The Intel Stratix 10 device does not support PLL merging. However, the duplex feature allows placing transmitters and receivers in the same I/O bank. The duplex feature is a simpler option compared to using an external PLL.

- To use the duplex feature, turn on the **Duplex Feature** option of the LVDS SERDES IP core.
- The number of transmitter and receiver channels in the IP core instance is the same.
- The limitation is that you can create only up to 11 transmitter and 11 receiver channels in each LVDS SERDES IP core instance.
- The LVDS SERDES IP core sets up PLL sources for the transmitters and receivers in the IP core instance.

## 3.5.2. Using 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
- Generate the IOPLL Intel FPGA IP and ensure that the `.qsf` file lists the IOPLL IP before the LVDS SERDES IP. This order is required for your design to compile with the proper clock constraints.
- Connect the same PLL to both the transmitter and receiver instances.

**Figure 28. LVDS Interface with the IOPLL IP (Non-DPA or 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.

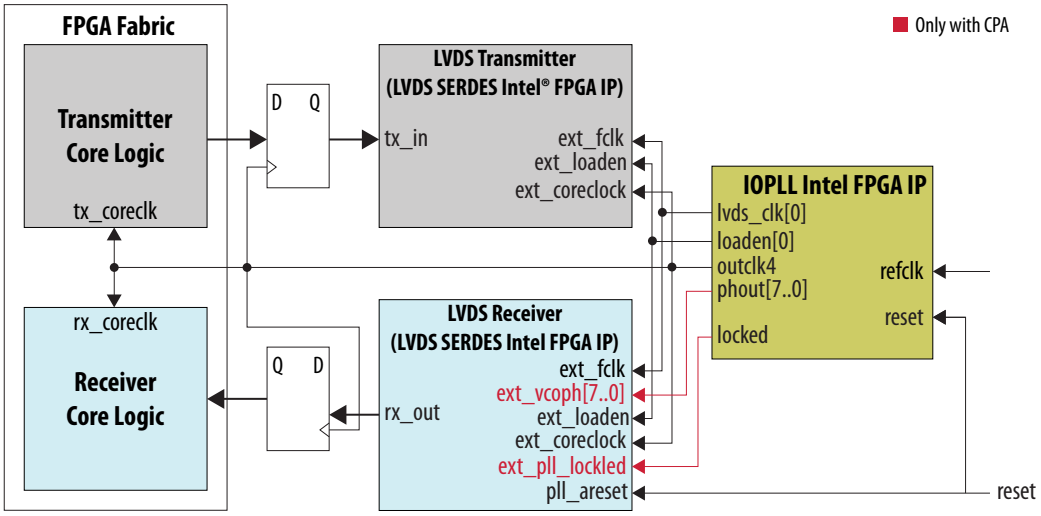
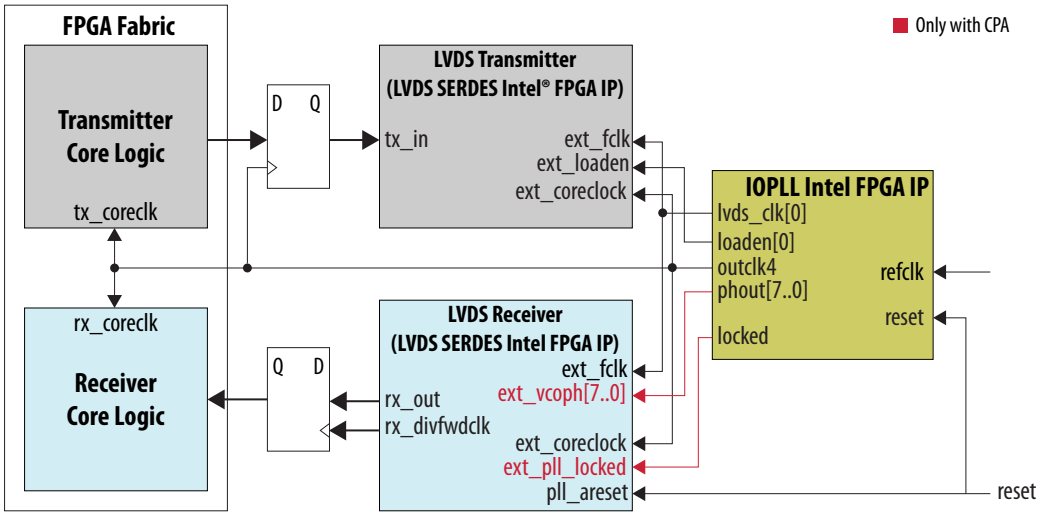


**Figure 29. LVDS Interface with the IOPLL IP (Soft-CDR Mode)**

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.




## 3.6. Guideline: LVDS SERDES Limitation for Intel Stratix 10 GX 400, SX 400, and TX 400

The LVDS SERDES Intel FPGA IP does not support the following I/O banks:

- Intel Stratix 10 GX 400 and SX 400 devices—I/O banks 3A, 3C, and 3D
- Intel Stratix 10 TX 400 devices—I/O banks 3A and 3D

# 4. Intel Stratix 10 High-Speed LVDS I/O Implementation Guides

You can implement your high-speed LVDS I/O design in the Intel Quartus Prime software. The software contains tools for you to create and compile your design, and configure your device.

The Intel Quartus Prime software allows you to prepare for device migration, set pin assignments, define placement restrictions, setup timing constraints, and customize IP cores. For more information about using the Intel Quartus Prime software, refer to the related information.

**Related Information**

Intel Quartus Prime Design Software

## 4.1. LVDS SERDES Intel FPGA IP

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.

With the LVDS SERDES IP core, you can implement these types of LVDS applications:

- Transmitter-only applications
- Receiver-only applications
- Applications with a mix of transmitters and receivers

*Note:* 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**

- 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 Platform Designer 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.

### 4.1.1. 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
- Duplex mode—transmitters and receivers in the same I/O bank
- Clock phase alignment (CPA) block

## 4.1.2. LVDS SERDES IP Core Functional Modes

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

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

**Table 15. 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.<br>A PLL generates the following signals:<br>• `fast_clock`<br>• `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.<br>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.<br>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.<br>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.<br>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. |
| Duplex (**Duplex Feature**) | In the duplex mode, the IP core automatically enables the transmitters. You can select the receiver mode to use. The number of transmitters and receivers are the same.<br>The duplex mode allows the IP core to place receivers and transmitters in the same I/O bank. You can enable up to 11 transmitter and 11 receiver channels.<br>If you enable the duplex mode, the external PLL mode is disabled. |

**Related Information**

Intel Stratix 10 Device Pin-Out Files

## 4.1.3. 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 30. LVDS SERDES Channel Diagram**

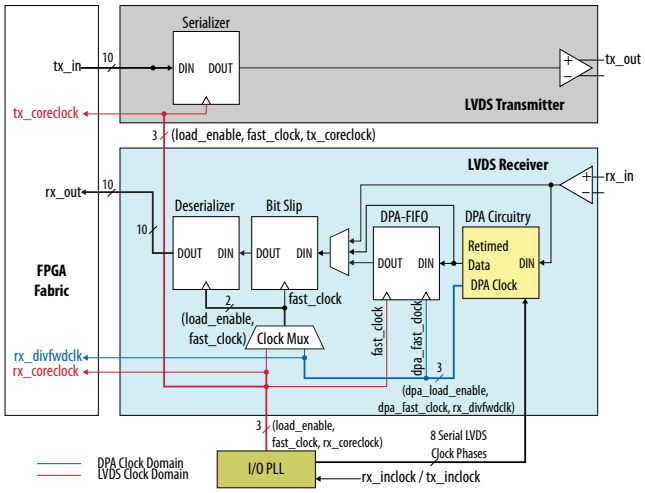



**Table 16. LVDS SERDES IP Core Channel Paths and Functional Units**

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<br>• Soft-CDR | DPA |
| | DPA FIFO | DPA-FIFO | LVDS–DPA domain crossing |
| | • Bitslip<br>• Deserializer | • Non-DPA<br>• 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 |

### 4.1.3.1. 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 31. LVDS x8 Serializer Waveform**

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

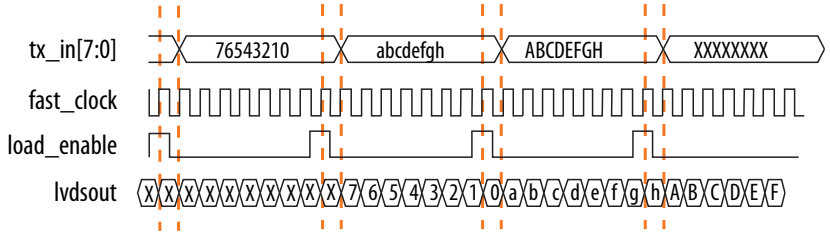



**Table 17. LVDS Serializer Signals**

| Signal | Description |
|---|---|
| `tx_in[7:0]` | Data for serialization<br>(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 |

### 4.1.3.2. 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.

### 4.1.3.3. 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.

### 4.1.3.4. 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 32. LVDS x8 Deserializer Waveform**

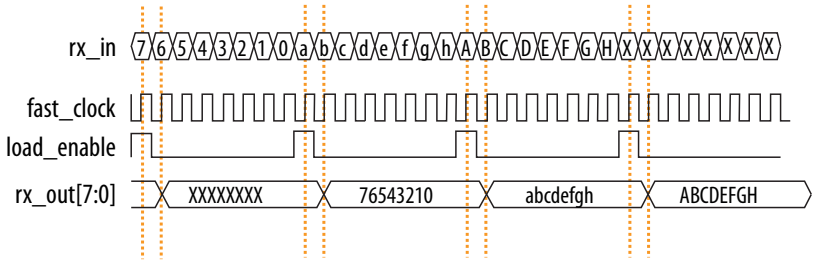



**Table 18. LVDS Deserializer Signals**

| 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 |

### 4.1.3.5. Clock Phase Alignment

The CPA block helps improve timing closure between the periphery and the core. To use this feature, turn on the **Use the CPA block for improved periphery-core timing** option in the LVDS SERDES IP core parameter editor.

If you turn on the option, the LVDS SERDES IP core uses the CPA block to phase-align the core clock and the load enable clock.

**Table 19. Core Clock Duty Cycles when Using the CPA Feature**

This table lists the actual core clock duty cycles if you turned on the **Use the CPA block for improved periphery-core timing** feature. The CPA feature works best for SERDES factors 4 and 8 where the core clock duty cycle remains 50%.

| SERDES Factor | Actual Core Clock Duty Cycle |
|---|---|
| 3 | 66.6% |
| 4 | 50% |
| 5 | 40% |
| 6 | 33% |
| 7 | 40% |

*continued...*

| SERDES Factor | Actual Core Clock Duty Cycle |
|---|---|
| 8 | 50% |
| 9 | 40% |
| 10 | 40% |

The **Use the CPA block for improved periphery-core timing** option is available for any selectable SERDES factor under the following conditions:

- The IP core functional mode is **TX**, **RX Non-DPA**, or **RX DPA-FIFO**.
- The `tx_outclock` phase shift is a multiple of 180°.

## 4.2. 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.

### 4.2.1. 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 45
- Aligning Word Boundaries on page 46

### 4.2.2. 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**

- Word Boundaries Alignment on page 45
- Aligning Word Boundaries on page 46
- Resetting the DPA on page 44
- DPA Lock Time Specifications, Intel Stratix 10 Device Datasheet
- LVDS Soft-CDR/DPA Sinusoidal Jitter Tolerance Specifications, Intel Stratix 10 Device Datasheet

## 4.2.3. 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 43
- DPA Lock Time Specifications, Intel Stratix 10 Device Datasheet
- LVDS Soft-CDR/DPA Sinusoidal Jitter Tolerance Specifications, Intel Stratix 10 Device Datasheet

## 4.2.4. 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 43
- Initializing the LVDS SERDES IP Core in DPA Mode on page 43
- Aligning Word Boundaries on page 46

### 4.2.4.1. 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 43
- Initializing the LVDS SERDES IP Core in DPA Mode on page 43
- Word Boundaries Alignment on page 45

## 4.3. LVDS SERDES IP Core Timing

Use the Intel Quartus Prime software to generate the required timing constraint to perform proper timing analysis of the LVDS SERDES IP core in Intel Stratix 10 devices.

**Table 20. 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: |

*continued...*

| Timing Component | Description |
|---|---|
| | • FPGA/LVDS/TX to external receiving device transmitting<br>• 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:<br>• LVDS RX hardware to core registers paths<br>• Core registers to LVDS TX hardware paths<br>• Others core registers to core registers path<br>The Timing Analyzer reports the corresponding timing margins. |

**Table 21. 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.<br>The IP core uses the **.sdc** for the following operations:<br>• Creating clocks on PLL inputs<br>• Creating generated clocks<br>• Calling `derive_clock_uncertainty`<br>• Creating proper multi-cycle constraints<br>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. |

## 4.3.1. 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.

### 4.3.1.1. 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**.

### 4.3.1.2. 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**.

## 4.3.2. 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 22. 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 23. 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.

## 4.3.3. 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`.

**Related Information**

LVDS Interface with External PLL Mode on page 25

## 4.3.4. 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.

## 4.3.5. Guideline: Use Clock Phase Alignment Block to Improve Timing Closure

For large clock networks, skew added to the core clock through the clock network can affect timing closure. To improve timing closure between the periphery and the core, turn on the **Use the CPA block for improved periphery-core timing** feature.

**Related Information**

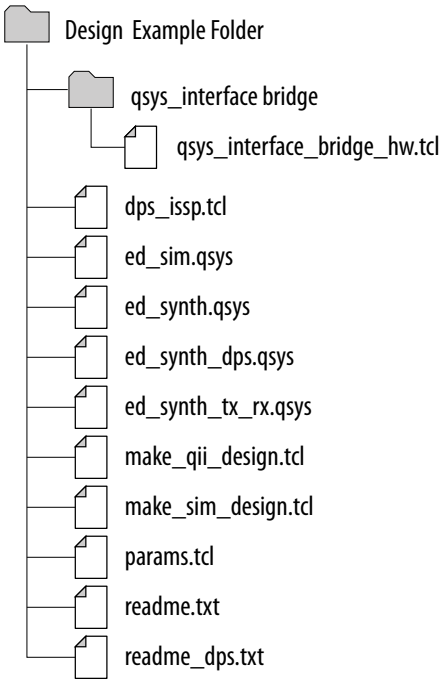
Clock Phase Alignment on page 42
Provides more information about the CPA feature of the LVDS SERDES IP core, its required conditions, and the resultant core clock duty cycles.

## 4.4. 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 33. Source Files in the Generated Design Example Directory**



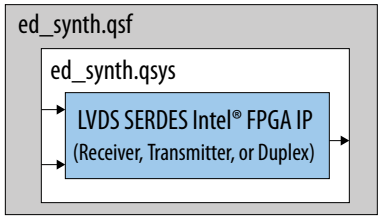

### 4.4.1. LVDS SERDES IP Core Synthesizable Intel Quartus Prime Design Examples

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

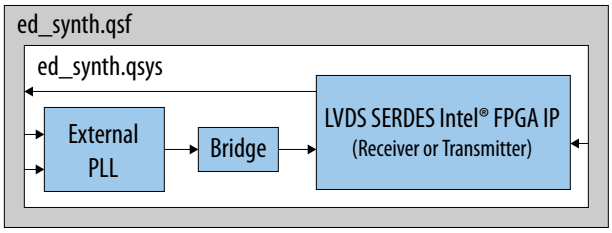
**Figure 34. Basic LVDS SERDES IP Core System with Internal PLL**




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

**Figure 35. 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
```

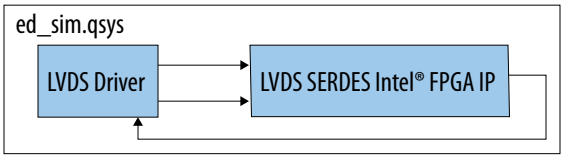
## 4.4.2. LVDS SERDES IP Core Simulation Design Example

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 36. 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.

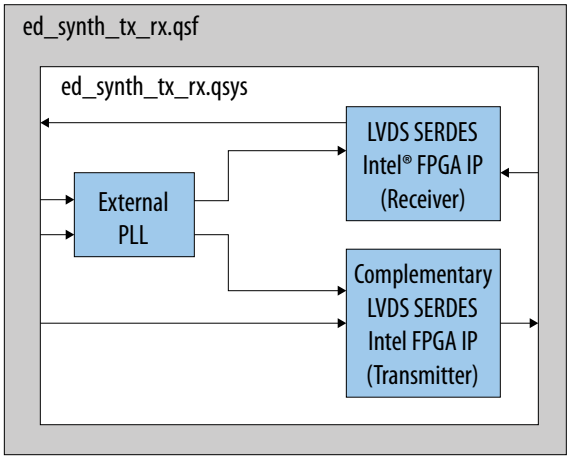
## 4.4.3. Combined LVDS SERDES IP Core Transmitter and Receiver Design Example

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.

*Note:* The combined transmitter and receiver design example does not support the duplex mode. If your LVDS SERDES IP core uses the **Duplex Feature** mode, ignore the `ed_synth_tx_rx.qsys` file generated by the **Generate Example Design** command.

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 37. 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
```

## 4.4.4. 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.

*Note:* The dynamic phase shift design example does not support the duplex mode. If your LVDS SERDES IP core uses the **Duplex Feature** mode, ignore the `ed_synth_dps.qsys` file generated by the **Generate Example Design** command.

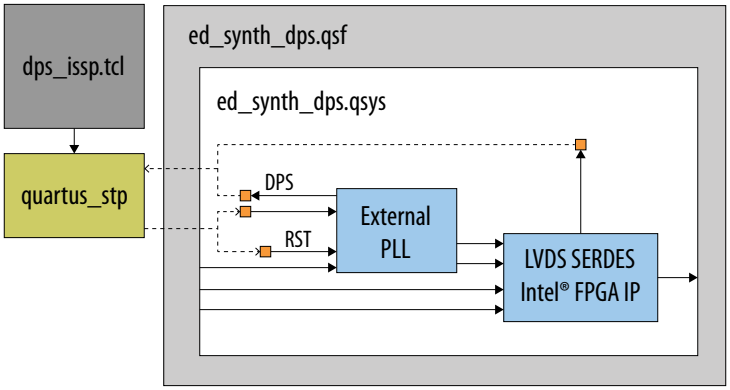
You can use this example in LVDS-specific applications such as debugging non-DPA receiver capture where you can repeatedly shift the capture clock to find the best operational phase shift.

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 38. 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
```

## 4.5. 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 Intel FPGA IP of Intel Stratix 10 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.

### 4.5.1. 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 **Stratix 10**.
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.

# 5. LVDS SERDES Intel FPGA IP References

You can set various parameter settings for the LVDS SERDES IP core to customize its behaviors, ports, and signals.

The Intel Quartus Prime software generates your customized LVDS SERDES IP core according to the parameter options that you set in the parameter editor.

## 5.1. LVDS SERDES IP Core Parameter Settings

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

### 5.1.1. LVDS SERDES IP Core General Settings

**Table 24. General Settings Tab**

| Parameter | Value | Description |
|---|---|---|
| **Duplex Feature** | On, Off | Turn on to allow transmitter and receiver channels in the same I/O bank.<br>• The number of transmitter and receiver channels in the same I/O bank is the same.<br>• The number of channels is limited to 11 transmitters and 11 receivers.<br>• **Use external PLL** option is disabled. |
| **Functional mode** | • TX<br>• RX Non-DPA<br>• RX DPA-FIFO<br>• RX Soft-CDR | Specifies the functional mode of the interface.<br>The **TX** option is not available if you turn on the **Duplex Feature** option. In duplex mode, transmitter channels are created by default. |
| **Number of channels** | • 1 to 72 for TX<br>• 1 to 24 for RX Non-DPA<br>• 1 to 24 for RX DPA-FIFO<br>• 1 to 12 for RX Soft-CDR<br>• 1 to 11 TX and RX if **Duplex Feature** is turned on | Specifies the number of serial channels in the interface.<br>• 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.<br>• If you use a transmitter output clock, you must use one of the channels for the `tx_outclock` pin.<br>For an LVDS RX design, place the `refclk` pin on the same I/O bank as the receiver.<br>For an LVDS TX design:<br>• For an interface with less than 23 channels (standalone), each interface requires a `refclk` pin on the same I/O bank.<br>• For an interface with more than 23 channels, channels 23 to 71 can share one `refclk` input.<br>In **Duplex Feature** mode, this value specifies the number of channels each for the transmitter and receiver. For example, if you specify 11 channels, the IP core uses 22 channels in the I/O bank. |

*continued...*

| Parameter | Value | Description |
|---|---|---|
| **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 clock-pin drive** | On, Off | Turn on to bypass the PLL and drive the interface with a clock pin.<br>*Note:* This feature will be supported in a future version of the Intel Quartus Prime software. |
| **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. |
| **Use the CPA block for improved periphery-core timing** | On, Off | Turn on to improve timing closure between the periphery and core. The IP core uses the clock phase alignment (CPA) block to phase-align the core clock and load enable clock. The option is available for any selectable SERDES factor if:<br>• **Functional mode** is **TX**, **RX Non-DPA**, or **RX DPA-FIFO**.<br>• **Desired tx_outclock phase shift (degrees)** parameter is a multiple of 180°. |

## 5.1.2. LVDS SERDES IP Core PLL Settings

### Table 25. PLL Settings Tab

| Parameter | Value | Description |
|---|---|---|
| **Use external PLL** | On, Off | Turn on to use an external PLL:<br>• The IP core does not instantiate a local PLL.<br>• The IP core creates a series of clock connections with the "ext" prefix. Connect these ports to an externally generated PLL.<br>• For details about how to configure the external PLL, refer to the **Clock Resource Summary** tab of the parameter editor.<br>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.<br>*Note:* If you want to place combined LVDS transmitters and receivers in the same I/O bank using two LVDS SERDES IP core instances, you must turn on this option. You can also place combined transmitters and receivers in the same I/O bank by turning on the **Duplex Feature** option in the **General Settings** tab. If you turn on **Duplex Feature**, the **Use external PLL** option is disabled. |
| **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. |
| **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`. |

***continued...***

| Parameter | Value | Description |
|---|---|---|
| | | *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**

- IOPLL Parameter Values for External PLL Mode on page 27
- LVDS SERDES IP Core Clock Resource Summary on page 63

## 5.1.3. LVDS SERDES IP Core Receiver Settings

**Table 26. 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).<br>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).<br>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.<br>When the bit slip reaches the specified value, it rolls over and the `rx_bitslip_max signal` asserts.<br>The rollover value is set automatically to the deserialization factor. |

**Table 27. 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.<br>(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).<br>If set high, the DPA logic in the corresponding channel does not switch sampling phases.<br>(Formerly known as `rx_dpll_hold`.) |

*continued...*

| Parameter | Value | Description |
|---|---|---|
| **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.<br>• 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.<br>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.<br>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<br>• Off—DPA logic counts the rising and falling edges<br>*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`.<br>*Note:* This feature will be supported in a future version of the Intel Quartus Prime software. |

**Table 28. 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 59

### 5.1.3.1. Receiver Input Clock Parameters Setup

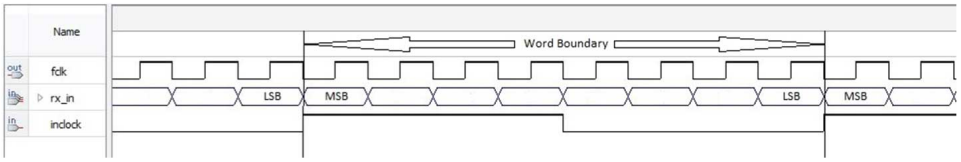
To sample the source-synchronous data using the SERDES receiver in non-DPA mode, you must specify the phase relationship between the `inclock` and the `rx_in` data.

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 39. 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.

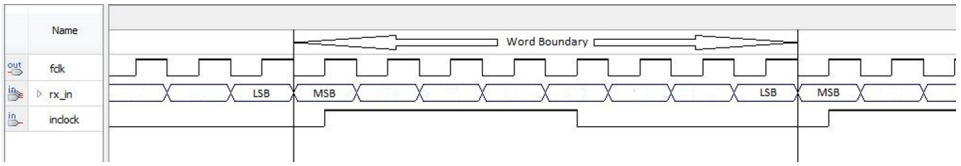
This equation determines the maximum phase shift value: (Number of fast_clock periods per inclock period x 360) – 1.

*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.

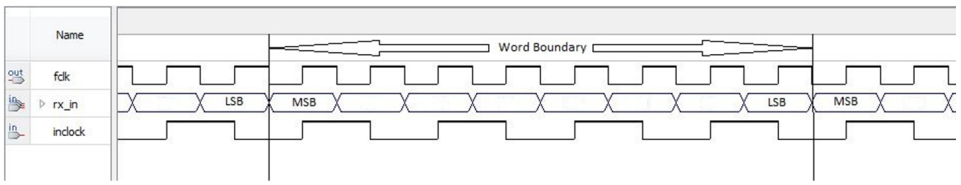
**Figure 40. 180° Center-Aligned inclock x8 Deserializer Waveform with Single Rate Clock**




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 41. 180° Center Aligned inclock x8 Deserializer Waveform with DDR Clock**



## 5.1.4. LVDS SERDES IP Core Transmitter Settings

### Table 29. Transmitter Settings Tab

| Parameter | Value | Description |
|---|---|---|
| **TX core registers clock** | • tx_coreclock<br>• inclock | Selects the clock that clocks the core registers:<br>• **tx_coreclock**—selects the `tx_coreclock` as the clock source.<br>• **inclock**—select the PLL `refclk` as the clock source. The `refclk` frequency must be equal to the data rate divided by the serialization factor.<br>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.<br>• If the **Use the clock phase alignment block for improved periphery-core timing for even SERDES factors** option in the **General Settings** tab is turned off, the `tx_coreclock` signal is a feedthrough of the `ext_coreclock` input.<br>• If the **Use the clock phase alignment block for improved periphery-core timing for even SERDES factors** option in the **General Settings** tab is turned on, the `tx_coreclock` signal is a phase-aligned core clock signal generated by the `loaden`.<br>Intel recommends that you use the `tx_coreclock` output signal if it is requested.<br>*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.<br>• The `tx_outclock` port frequency depends on the setting for the **tx_outclock division factor** parameter.<br>• The `tx_outclock` port phase depends on the **Desired tx_outclock phase shift** parameter.<br>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**

- Clocking Differential Transmitters on page 20
- Setting the Transmitter Output Clock Parameters on page 62

### 5.1.4.1. Setting the Transmitter Output Clock Parameters

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

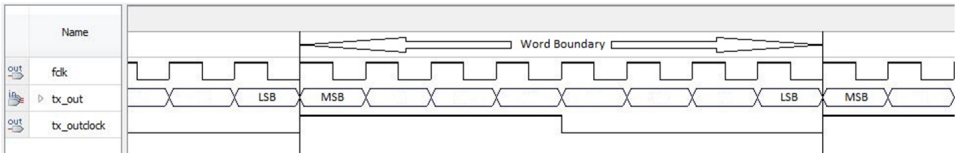
- **Desired tx_outclock phase shift (degrees)**
- **Tx_outclock division factor**

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.

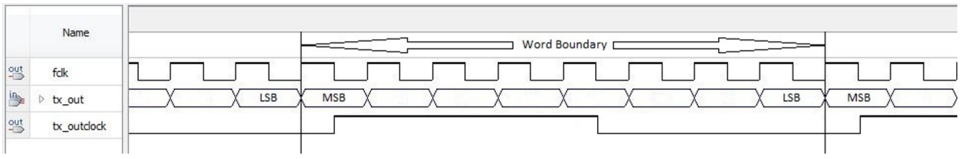
**Figure 42. 0° Edge Aligned `tx_outclock` x8 Serializer Waveform with Division Factor of 8**




#### Center-Aligned `tx_outclock` to `tx_out`

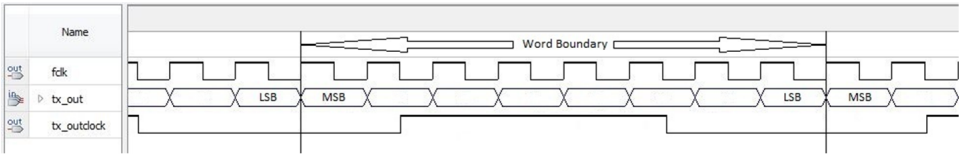
To specify center-aligned relationship between `tx_outclock` and the MSB of the serial data on `tx_out`, specify 180° phase shift.

**Figure 43. 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.

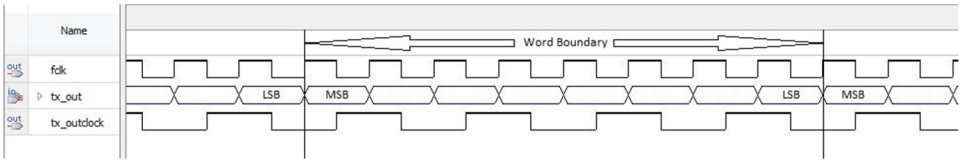
**Figure 44. 540° Center Aligned `tx_outclock` x8 Serializer Waveform with Division Factor of 8**




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

**Figure 45. 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 61

### 5.1.5. 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.

**Related Information**

IOPLL Parameter Values for External PLL Mode on page 27

## 5.2. LVDS SERDES IP Core Signals

**Table 30. 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 31. 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<br>• 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<br>— Held high—selected channels hold their current phase setting<br>— 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<br>• Applicable in DPA-FIFO and soft-CDR modes only |
| rx_dpa_reset | *N* | Input | Reset | • Asynchronous, active-high reset to DPA blocks<br>• Minimum pulse width: one parallel clock period<br>• Applicable in DPA-FIFO and soft-CDR modes only |
| rx_fifo_reset | *N* | Input | Reset | • Asynchronous, active-high reset to FIFO block<br>• Minimum pulse width: one parallel clock period<br>• Applicable in DPA-FIFO mode only |
| rx_out | *N***J* | Output | Data | Receiver parallel data output<br>• DPA-FIFO and non-DPA modes—synchronous to rx_coreclock.<br>• Soft-CDR mode—each channel has parallel data synchronous to its rx_divfwdclk |
| rx_bitslip_max | *N* | Output | Control | • Bit slip rollover signal<br>• 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<br>• Not available if you use an external PLL |
| rx_divfwdclk | *N* | Output | Clock | The per channel and divided clock with the ideal DPA phase<br>• This is the recovered slow clock for a given channel<br>• 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<br>• Driven by the LVDS SERDES IP core<br>• Asserts when the signal settles on an ideal phase for that given channel<br>• Deasserts in one of these conditions:<br>— The DPA moves one phase<br>— The DPA moves two phases in the same direction<br>• Applicable in DPA-FIFO and soft-CDR modes only<br>Ignore all toggling of the `rx_dpa_locked` signal after `rx_dpa_hold` asserts. |

## Table 32. 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)<br>• Source-synchronous with `tx_out` |
| `tx_coreclock` | 1 | Output | Clock | Drives the core logic feeding the serializer<br>• If the clock phase alignment block is off, this signal is a feedthrough of the `ext_coreclock` input.<br>• If the clock phase alignment block is on, this signal is a phase-aligned core clock signal generated by the `loaden`. |

## Table 33. 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<br>• Used for serial data transfer<br>• Required in all modes<br>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<br>• Used for parallel load<br>• Not required in RX soft-CDR mode<br>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)<br>• 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<br>• Required for RX DPA-FIFO and RX soft-CDR modes<br>• Required for all supported modes if you turn on **Use the CPA block for improved periphery-core timing**<br>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<br>• Required for RX DPA-FIFO and RX Soft-CDR modes<br>• Required for all supported modes if you turn on **Use the CPA block for improved periphery-core timing** |
| ext_tx_outclock_fclk | 1 | Input | Clock | Phase-shifted version of fast clock<br>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<br>Required for TX outclock phase shifts that are not multiples of 180° |

**Related Information**

- IOPLL Parameter Values for External PLL Mode on page 27
- LVDS SERDES IP Core Clock Resource Summary on page 63

## 5.3. Comparison of LVDS SERDES Intel FPGA IP 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 34. Intel Stratix 10 and Stratix V Devices Feature Comparison**

| Features | Intel Stratix 10 Devices | Stratix V Devices |
|---|---|---|
| Operation Frequency Range | 150 MHz - 1.6 GHz | |
| 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<br>(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.<br>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 |

# 6. Intel Stratix 10 High-Speed LVDS I/O User Guide Archives

If an IP core version is not listed, the user guide for the previous IP core version applies.

| IP Core Version | User Guide |
|---|---|
| 19.2 | Intel Stratix 10 High-Speed LVDS I/O User Guide |
| 19.1 | Intel Stratix 10 High-Speed LVDS I/O User Guide |
| 18.1 | Intel Stratix 10 High-Speed LVDS I/O User Guide |
| 18.0 | Intel Stratix 10 High-Speed LVDS I/O User Guide |
| 17.1 | Intel Stratix 10 High-Speed LVDS I/O User Guide |

# 7. Document Revision History for the Intel Stratix 10 High-Speed LVDS I/O User Guide

| Document Version | Intel Quartus Prime Version | Changes |
|---|---|---|
| 2020.01.03 | 19.4 | • Updated the programmable pre-emphasis diagram to remove the word "peak-peak".<br>• Added guideline topic about LVDS SERDES limitation for Intel Stratix 10 GX 400, SX 400, and TX 400 devices. |
| 2019.07.10 | 19.2 | Corrected the clock connection to the register and `rx_coreclock` in the figure showing the connection for non-DPA or DPA receiver interface with the IOPLL IP core in external PLL mode. |
| 2019.05.02 | 19.1 | • Moved the LVDS SERDES usage modes summary table into its own topic.<br>• Updated the description of the LVDS SERDES usage modes table to improve accuracy.<br>• 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. |
| 2019.02.26 | 18.1 | Updated the guidelines for the LVDS interface with external PLL mode:<br>• Combined the figures for the non-DPA and DPA modes.<br>• Marked in the figures the ports that are available in CPA mode only.<br>• Updated the source for the LVDS SERDES IP reset signal.<br>• Updated the connection of the `locked` signal from the IOPLL IP to the `ext_pll_locked` port of the LVDS SERDES IP. |
| 2019.01.14 | 18.1 | Removed statement that says that the programmable $V_{OD}$ value of "0" is not available for the LVDS I/O standard. |
| 2018.11.12 | 18.1 | • Updated the table listing the dedicated circuitries and features of the differential transmitter to clarify that the serializer width is from 3-bits to 10-bits.<br>• Updated the guideline about the LVDS reference clock source to include support for reference clock input from other I/O banks.<br>• Removed `ext_loaden` signal in figures showing the LVDS receiver in soft-CDR mode.<br>• Specified that connecting the IOPLL loaden signal to the LVDS receiver ext_loaden signal is not required for LVDS receivers in soft-CDR mode.<br>• Removed restriction of using the CPA block while the external PLL option is turned on.<br>• Updated the topic about the timing analysis for the external PLL mode to improve clarity. |

*continued...*

| Document Version | Intel Quartus Prime Version | Changes |
|---|---|---|
| | | • Updated the topic about the simulation design example to add a note about the non-synthesizable simulation driver.<br>• Renamed "TimeQuest Timing Analyzer" to "Timing Analyzer".<br>• Renamed "SignalTap" to "Signal Tap". |
| 2018.08.06 | 18.0 | • Clarified that all LVDS SERDES IP usage modes support SERDES factors of 3 to 10.<br>• Clarified that unused pins within an I/O bank with DPA feature enabled can be assigned to single ended or differential I/O standards that has the same VCCIO voltage level used by the bank in *Guideline: Pin Placement for Differential Channels* section.<br>• Removed LVDS channels count tables in *Intel Stratix 10 LVDS Channels Support* topic and added link to Intel Stratix 10 pin-out files.<br>• Removed the "pending characterization" labels in the topic showing the example for the RSKM calculation.<br>• Updated the list of LVDS SERDES IP core features to include the CPA block.<br>• Updated outclk2 to outclk4 in all examples of using the LVDS interfaces in external PLL mode.<br>• Updated tables and examples for IOPLL and LVDS SERDES IP cores signals in external PLL mode to include information about using the IP cores with the CPA block turned on.<br>• Updated the LVDS SERDES IP core instantiation guideline to specify that you can use multiple LVDS SERDES IP cores instance per I/O bank in any functional mode by using an external PLL.<br>• Corrected typograpical error—changed `tx_inclock` to `rx_inclock`—in the topic about the deserializer.<br>• Updated the figures descriptions in the guideline topic about using external PLL to use LVDS transmitters and receivers in the same I/O bank to clarify that the figures show connections that you need to make.<br>• Added topic about the CPA block under the Functional Description section. Moved information from the CPA feature guideline topic to this new topic.<br>• Updated the guideline topic about using the CPA feature to move information to a new CPA topic. Added link to the new topic.<br>• Updated the synthesizable design example topic to improve clarity and add duplex mode.<br>• Corrected the combined receiver and transmitter design example topic to specify that it creates an external PLL. The combined transmitter and receiver design example does not support the duplex feature.<br>• Updated the dynamic phase shift design example topic to specify that the design example does not support the duplex feature.<br>• Updated the LVDS SERDES IP core general settings reference topic to clarify the number of channels in the **Duplex Feature** mode and to update the CPA feature parameter name.<br>• Updated the names of the following IP cores:<br>— Intel FPGA LVDS SERDES to LVDS SERDES Intel FPGA IP<br>— Intel FPGA IOPLL to IOPLL Intel FPGA IP<br>— Intel FPGA GPIO to GPIO Intel FPGA IP |

| Date | Version | Changes |
|---|---|---|
| November 2017 | 2017.11.06 | • Added the duplex feature option that allows you to place transmitters and receivers in the same I/O bank using a single instance of the LVDS SERDES IP core.<br>• Removed the HF50 package from all Intel Stratix 10 devices.<br>• Added package SF48 to Intel Stratix 10 TX 1650 and TX 2100 devices.<br>• Removed Intel Stratix 10 TX 4500 and TX 5500 devices. |

***continued...***

| Date | Version | Changes |
|---|---|---|
| | | • Added Intel Stratix 10 MX devices.<br>• Updated descriptions of the tables that list the LVDS channels support to specify that the LVDS channels counts include dedicated clock pins.<br>• Changed all instances of the following IP names:<br>— Altera LVDS SERDES to Intel FPGA LVDS SERDES<br>— Altera IOPLL to Intel FPGA IOPLL<br>— Altera GPIO to Intel FPGA GPIO<br>• Renamed "Qsys" to Platform Designer.<br>• Removed the statement about selecting the rising edge option in the parameter editor for the **RX Non-DPA** mode.<br>• Updated the topic about clocking differential transmitters to improve clarity about transmitter placement limitation in relations to the `tx_outclock` phase shift.<br>• Restructured the information in the topic about connecting the external PLL to the LVDS receiver and transmitter. Moved some of the information to the guideline topic about using external PLL for combined LVDS transmitters and receivers in the same I/O bank.<br>• Rewrote the guideline topic about using external PLL for combined LVDS transmitters and receivers in the same I/O bank. The topic now describes using either an external PLL or the duplex feature of the LVDS SERDES IP core.<br>• Added quick guidelines about using the SERDES in the topic that provides an overview of the high-speed LVDS I/O.<br>• Updated the note about driving LVDS channels with the PLL in integer PLL mode to clarify that you do not need a PLL if you bypass the SERDES.<br>• Updated the topic about the serializer bypass for DDR and SDR operation to add more information about clocks to the IOE.<br>• Updated the topic about the deserializer to add more information about bypassing the deserializer.<br>• Removed the statement about SDR and DDR data width from the figures that show the receiver datapath in non-DPA, DPA, and soft-CDR modes.<br>• Corrected typographical error in the example showing the parameter values to generate output clock in external PLL mode by updating "c0" to "outclk0".<br>• Added more description for the **Enable tx_coreclock port** parameter option to describe how configure it in external PLL mode, and the effect of turning on the clock phase alignment block.<br>• Updated the description of the `tx_coreclock` signal.<br>• Removed the *RSKM Report for LVDS Receiver* and *Assigning Input Delay to LVDS Receiver Using TimeQuest Timing Analyzer* topics and added a related link to *Obtaining RSKM Report* topic instead.<br>• Updated the topic about the combined transmitter and receiver design example to specify that the design example uses the duplex mode feature.<br>• Added guideline topic about LVDS reference clock source.<br>• Added a note about using external PLL with a wide transmitter interface that spans multiple I/O banks.<br>• Updated the **Use the CPA block for improved periphery-core timing for even SERDES Factors** IP core parameter option to update the label and specify that it is now available for any selectable SERDES factor. |
| May 2017 | 2017.05.08 | • Updated the timing diagram that shows the DPA clock phase to serial data timing relationship to align the clock phases with the data.<br>• 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 GPIO IP core.<br>• Added guideline topic about using external PLL for LVDS transmitter and receiver interfaces combined in an I/O bank. |

*continued...*

| Date | Version | Changes |
|---|---|---|
| | | • Added guideline topic about using the clock phase alignment block to improve periphery-core timing.<br>• 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.<br>• Added the "Use the clock phase alignment block for improved periphery-core timing for even SERDES factors" IP core parameter option. |
| February 2017 | 2017.02.13 | Removed the SF48 package from the Intel Stratix 10 TX 1650 and TX 2100 devices. |
| October 2016 | 2016.10.31 | Initial release. |