Intel Agilex® 5 LVDS SERDES User Guide

Updated for Intel® Quartus® Prime Design Suite: **23.2.1**

Send Feedback

766906
2023.08.11

Preliminary Documentation – Subject to Change

david@terasic.com 20420503;006819 Chung-Chieh Wei

Contents

Preliminary Documentation – Subject to Change

1. Intel Agilex® 5 LVDS SERDES Overview

The Intel Agilex® 5 I/O system includes four types of I/O interfaces—high-speed I/O (HSIO), high-voltage I/O (HVIO), Secure Device Manager (SDM) I/O, and Hard Processor System (HPS) I/O. Each I/O interface caters to different interfacing requirements.

Intel Agilex 5 devices support LVDS serializer/deserializer (SERDES) through the True Differential Signaling I/Os in the HSIO banks. The true differential I/Os are capable of supporting LVDS interfaces, including subsets such as:

- RSDS
- Mini-LVDS
- SLVS
- Any I/O standards using equivalent electrical specifications

Intel Agilex 5 devices support SERDES in all HSIO banks with the following features:

- Configurable transmitter or receiver on all I/O pins
- Serialize and deserialize functions up to 1.6 Gbps.
- Clock data recovery (CDR) function on specific differential channel
- Configurable 100 Ω differential on-chip termination (OCT R_D)
- Serialize or deserialize factors of 4 and 8(1)
- I/O standards support for the LVDS SERDES:
  - Transmitter—True Differential Signaling I/O standard at 1.3 V
  - Receiver:
    - DPA mode—True Differential Signaling and SLVS-400 I/O standards
    - Non-DPA mode—True Differential Signaling I/O standard only

1.1. LVDS SERDES Usage Modes

You can use the Intel Agilex 5 LVDS SERDES through the LVDS SERDES Intel® FPGA IP. The LVDS SERDES IP supports four SERDES functional modes.

(1) Serialization factor of 8 is available only in Intel Agilex 5 FPGAs production devices.

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Preliminary Documentation – Subject to Change

Table 1. Summary of the Intel Agilex 5 LVDS SERDES Usage Modes

All usage modes in this table support SERDES factors of 4 and 8.

Functional Mode	Description
Transmitter (**TX**)	• The SERDES block acts as a serializer. • A PLL generates these signals: — `fast_clock` — `load_enable`
Non-DPA receiver (**RX Non-DPA**)	• The SERDES block acts as a deserializer that bypasses the DPA and DPA-FIFO. • A PLL generates the `fast_clock` signal. • The SERDES captures the incoming data at the bit slip with the `fast_clock` signal. Therefore, you must ensure the correct clock–data alignment.
DPA-FIFO receiver (**RX DPA-FIFO**)	• The SERDES block acts as a deserializer that uses the DPA block. • The DPA block uses a set of eight DPA clocks to select the optimal phase for sampling data. — The 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 bit slip 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**)	• The LVDS SERDES IP forwards these clocks: — The optimal DPA clock (DPACLK) into the LVDS clock domain as the `fast_clock` signal. — The `rx_divfwdclk`, produced by the local clock generator, to the device core. • Each bank has only 12 soft-CDR channels available. Refer to the device pin-out files to determine which pin pairs can support soft-CDR channels in each bank.

Preliminary Documentation – Subject to Change

2. Intel Agilex 5 LVDS SERDES Architecture

Each HSIO bank in Intel Agilex 5 devices consists of two sub-banks. Each sub-bank contains its own V_{CCIO_PIO}, PLL, dynamic phase alignment (DPA), and SERDES circuitry blocks.

You can configure each SERDES channel as a transmitter or a receiver.

Table 2. Differential Pairs and Channel Mode Support in Each Bank and Sub-Bank

This table lists the number of SERDES channels supported. Refer to the device pin-out files for the exact location of the SERDES and Soft-CDR pins.

Total Transmitter or Receiver Pairs Per Bank	Channel Mode	Maximum Pairs Per Sub-Bank	
		Top Index Sub-Bank	Bottom Index Sub-Bank
47(2)	Transmitter	24	24
	DPA	24	24
	Non-DPA	24	24
	Soft-CDR	4	8

Related Information

I/O PLLs Driving LVDS Transmitter and Receiver Channels on page 58

2.1. Intel Agilex 5 HSIO Banks, SERDES, and DPA Locations

The HSIO banks are located at the top and bottom I/O bank rows.

(2) One LVDS SERDES pair is used for the reference clock.

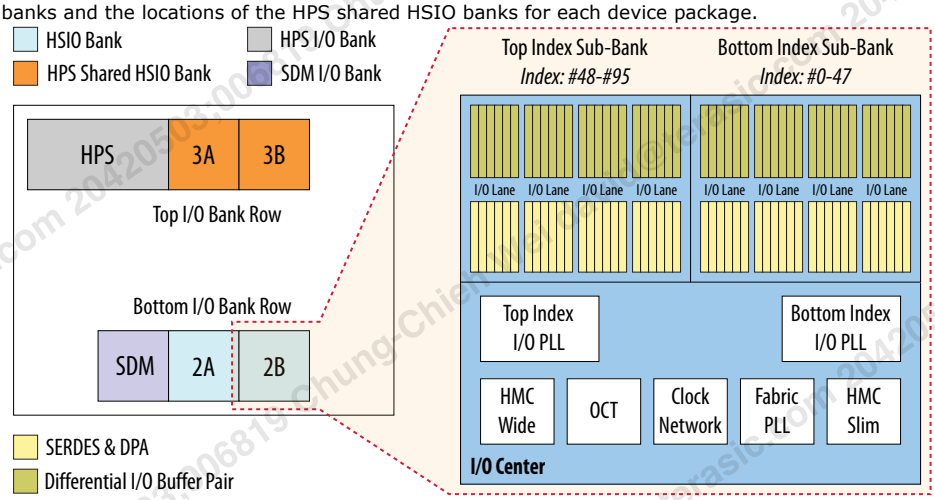
Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Preliminary Documentation – Subject to Change

Figure 1. Intel Agilex 5 HSIO Bank Structure (Die Top View)

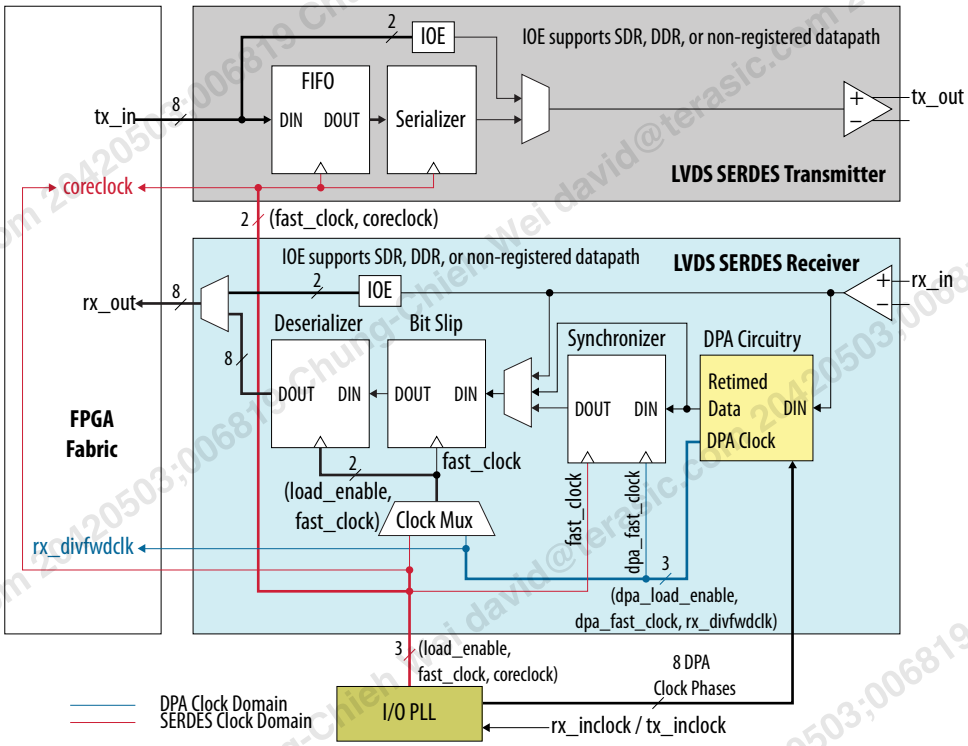
This figure shows the HSIO bank structure of the Intel Agilex 5 device. The figure shows the view of the die as shown in the Intel Quartus® Prime **Chip Planner**. In the **Pin Planner**, this corresponds to the "Bottom View". Different device packages have different number of HSIO banks. Refer to the device pin-out files for available HSIO banks and the locations of the HPS shared HSIO banks for each device package.



2.2. SERDES Blocks, Modes, and Clock Domains

Figure 2. SERDES Circuitry

This figure shows a transmitter and receiver block diagram for the SERDES circuitry with the interface signals of the transmitter and receiver data paths. The figure shows a transmitter and a receiver sharing an I/O PLL as they are in the same sub-bank and using the same I/O PLL resource. In single data rate (SDR) and double data rate (DDR) modes, the data widths are 1 and 2 bits, respectively.



Preliminary Documentation – Subject to Change

Table 3. Supported Modes, Blocks, and Clocks for the Data Paths

Data Path	Mode	Block	Clock Domain
Transmitter	**TX**	Serializer	SERDES clock domain
Receiver	**DPA-FIFO**	DPA	DPA clock domain
		Synchronizer	DPA-SERDES clock domain crossing
		Bit Slip	SERDES clock domain
		Deserializer	SERDES clock domain
	Non-DPA	DPA	Not used
		Synchronizer	Not used
		Bit Slip	SERDES clock domain
		Deserializer	SERDES clock domain
	Soft-CDR	DPA	DPA clock domain
		Bit Slip	DPA clock domain
		Deserializer	DPA clock domain

3. Intel Agilex 5 LVDS SERDES Transmitter

The Intel Agilex 5 LVDS SERDES transmitters are dedicated circuitries.

Each dedicated transmitter circuitry consists of:

- A transmitter buffer
- A serializer
- PLL shared with other SERDES within the same I/O bank

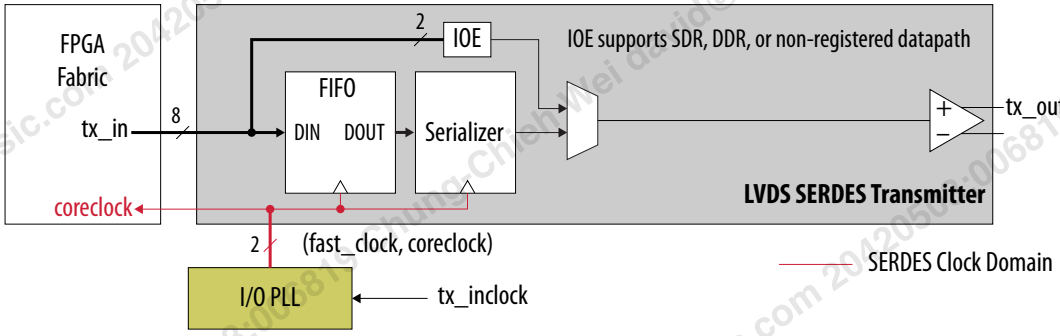
Table 4. Dedicated Circuitry and Features of the LVDS SERDES Transmitter

Dedicated Circuitry / Feature	Description
Differential I/O buffer	Supports True Differential Signaling I/O standard (at 1.3 V V_{CCIO_PIO} only), which is compatible with LVDS, RSDS, SLVS, and Mini-LVDS.
Serializer	4-bit or 8-bit(3) wide serializer
Phase-locked loops (PLLs)	Clocks the registers
Programmable V_{OD}	Adjusts the output voltage swing
Programmable pre-emphasis	Boosts output current

3.1. LVDS SERDES Transmitter Blocks

In the Intel Agilex 5 LVDS SERDES transmitter, the serializer receives up to 8 bits(3) wide parallel data from the FPGA fabric.

Figure 3. LVDS SERDES Transmitter



(3) Serialization factor of 8 is available only in Intel Agilex 5 FPGAs production devices.

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

- The serializer clocks the data into the registers and serializes the data using a multiplexer.
- The I/O PLL that drives the data to the differential buffer clocks the shift registers.
- The multiplexer transmits the MSB of the parallel data first.

Note: The PLL that drives the SERDES channel must operate in integer PLL mode.

3.2. Serializer

The serializer consists of two sets of registers. The first set of registers (FIFO) captures the parallel data from the core using the LVDS fast clock and then transfers the data to the serializer block. The MSB of the serializer feeds the LVDS output buffer. Consequently, higher order bits precede lower order bits in the output bitstream.

Figure 4. LVDS SERDES ×8 Serializer Bit Position

This figure shows the waveforms specific to the serialization factor of 8. These are functional waveforms and do not convey timing information.

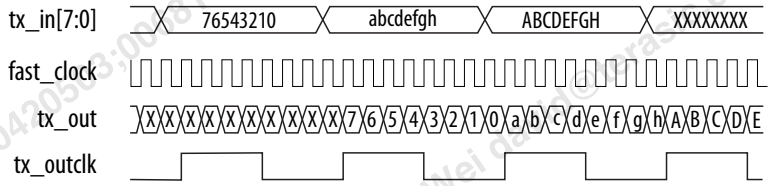


Table 5. LVDS SERDES Serializer Signals

Signal	Description
`tx_in[7:0]`	Data for serialization (Supported serialization factors: 4 and 8[4])
`fast_clock`	Clock for the transmitter
`tx_out`	LVDS output data stream

3.2.1. Serializer Bypass for DDR and SDR Operations

The I/O element (IOE) contains two data output registers. Each data output register can operate in double data rate (DDR) or single data rate (SDR) mode. Use the GPIO Intel FPGA IP to bypass the serializer and operate in DDR and SDR modes.

(4) Serialization factor of 8 is available only in Intel Agilex 5 FPGAs production devices.

Preliminary Documentation – Subject to Change

Figure 5. Serializer Bypass

This figure shows the serializer bypass path.

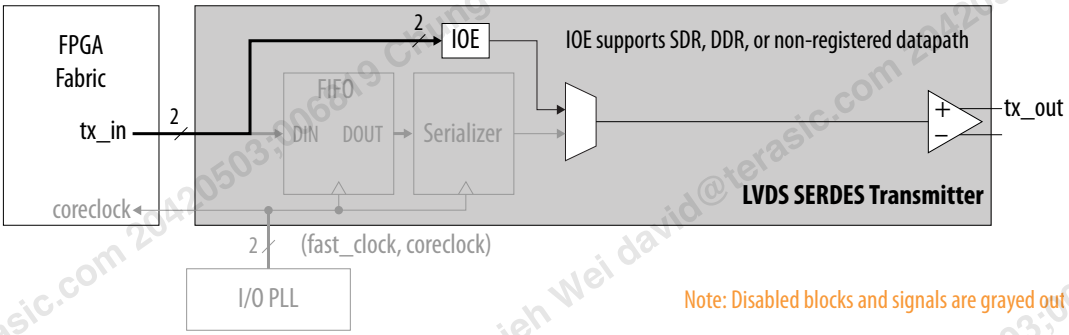


Table 6. SDR and DDR Transmitter Modes

Mode	Description
SDR (×1)	• The IOE data width is 1 bit. • Serialization factor of 1. • Registered output path requires a clock. • Data is passed directly through the IOE.
DDR (×2)	• The IOE data width is 2 bits. • Serialization factor of 2. • The GPIO IP requires a clock. • `tx_inclock` clocks the IOE register.

3.2.2. Differential I/O Bit Position

Table 7. Differential Bit Naming

This table lists the differential bit naming conventions for 12 differential channels. The MSB and LSB positions increase with the number of channels a system uses.

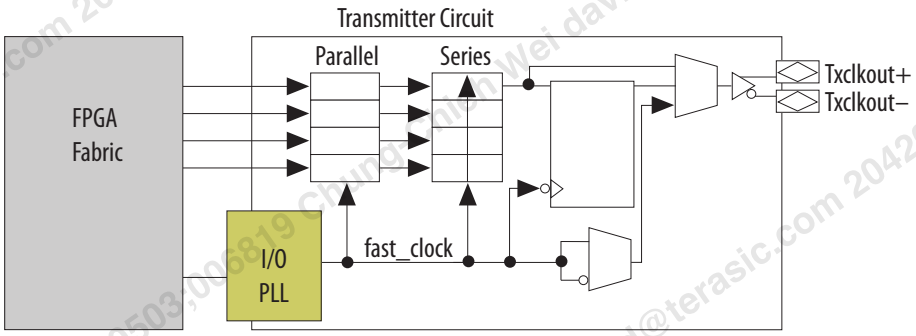
Transmitter Channel Data Number	Internal 8-Bit Parallel Data	
	MSB Position	LSB Position
1	7	0
2	15	8
3	23	16
4	31	24
5	39	32
6	47	40
7	55	48
8	63	56
9	71	64
10	79	72
11	87	80
12	95	88

3.3. Clocking the Differential Transmitters

The I/O PLL generates the `fast_clock` signal. The `fast_clock` signal clocks the load and shift registers, and runs at the serial data rate.

You can configure any Intel Agilex 5 transmitter data channel to generate a source-synchronous transmitter clock output. This allows placement of the output clock near the data outputs to simplify board layout and reduce clock-to-data skew.

Figure 6. Transmitter in Clock Output Mode



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 LVDS SERDES Intel FPGA IP parameter editor:

- The transmitter can output a clock signal at the same rate as the data with a maximum output clock frequency supported by the device speed grade.
- You can divide the output clock by a factor of 4 or 8, depending on the serialization factor.
- You can set the phase of the clock in relation to the data at 0° (edge-aligned) or 180° (center-aligned). The I/O PLLs provide additional support for other phase shifts in 45° increments.

Preliminary Documentation – Subject to Change

4. Intel Agilex 5 LVDS SERDES Receiver

The Intel Agilex 5 LVDS SERDES receivers are dedicated circuitries.

Table 8. Dedicated Circuitry and Features of the LVDS SERDES Receiver

Dedicated Circuitry / Feature	Description
Differential I/O buffer	Supports I/O standards compatible with LVDS, RSDS, SLVS, Mini-LVDS, and LVPECL: • DPA mode—True Differential Signaling and SLVS-400 I/O standards • Non-DPA mode—True Differential Signaling I/O standard only
Phase-locked loops (PLLs)	Generates different phases of a clock for data synchronizer
Data realignment (bit slip)	Inserts bit latencies into serial data
Dynamic phase alignment (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
On-chip termination (OCT)	100 Ω in True Differential Signaling I/O standards

4.1. LVDS SERDES Receiver Blocks

The True Differential Signaling buffers can interface with LVDS, mini-LVDS, RSDS, and LVPECL compatible signaling. You can statically set the I/O standard of the receiver pins to True Differential Signaling in the Intel Quartus Prime Assignment Editor or `.qsf` file.

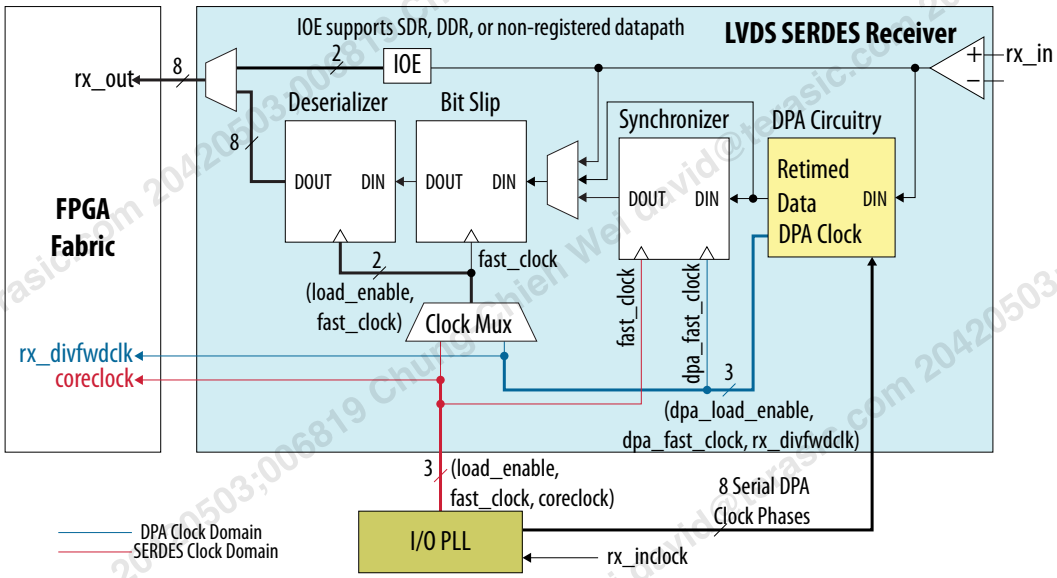
Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Preliminary Documentation – Subject to Change

Figure 7. 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 8 bits to the internal logic.



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

4.1.1. Dynamic Phase Alignment Block

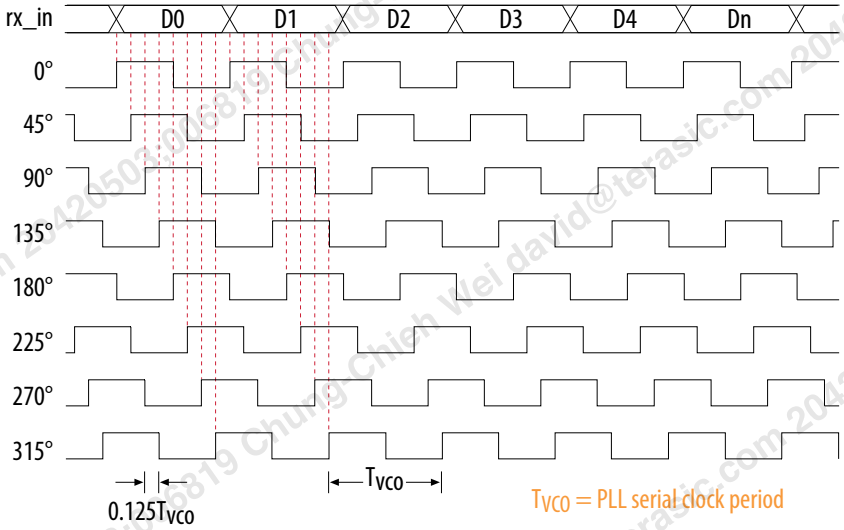
The DPA block receives high-speed serial data from the differential input buffer. To sample the data, the DPA block selects one of the eight phases generated by the I/O PLLs.

The DPA block selects a phase closest to the phase of the serial data. The maximum phase offset between the received data and the selected phase is $\frac{1}{8}$ unit interval (UI)[(5)], which is the maximum quantization error of the DPA. The eight phases of the clock divides equally, offering a 45° resolution.

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

Figure 8. 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 when required. To prevent the DPA from selecting a new clock phase, assert the optional `rx_dpa_hold` signal available for each channel.

The 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. The optional `rx_dpa_locked` output signal indicates an initial DPA lock condition to the optimum phase after power up or reset. To validate the data, use data checkers such as a cyclic redundancy check (CRC) or diagonal interleaved parity (DIP-4).

The independent `rx_dpa_reset` reset signal resets the DPA circuitry. After a reset, retrain the DPA circuitry.

Note: The receiver bypasses the DPA block in the non-DPA receiver mode.

4.1.2. Synchronizer (DPA FIFO)

The synchronizer is a 1-bit wide and 6-bit deep FIFO buffer that compensates for the phase difference between `dpa_fast_clock` from the DPA block and the `fast_clock` that the I/O PLLs produce.

The synchronizer can compensate only for phase differences, not frequency differences, between the data and the input reference clock of the receiver.

The optional `rx_fifo_reset` signal resets the synchronizer. The synchronizer resets automatically when the DPA block first locks to the incoming data. If your data checker indicates corrupt received data, use `rx_fifo_reset` to reset the synchronizer.

Note: The receiver bypasses the synchronizer circuit in non-DPA and soft-CDR modes.

4.1.3. 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 the 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.

The optional `rx_bitslip_ctrl` signal controls the bit insertion of each receiver that is independently controlled from the internal logic. The data slips one bit on the rising edge of `rx_bitslip_ctrl`.

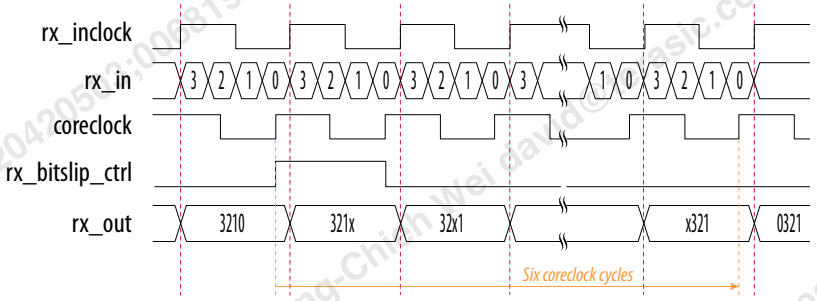
The `rx_bitslip_ctrl` signal has the following requirements:

- 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 six parallel clock cycles after the rising edge of `rx_bitslip_ctrl`.

The MSB from the serial data is not the MSB of the parallel data. You can use the bit slip to set the proper word boundary on the parallel data.

Figure 9. Data Realignment Timing

This figure shows the receiver output (`rx_out`) signal after a 1-bit slip pulse with the deserialization factor of 4.

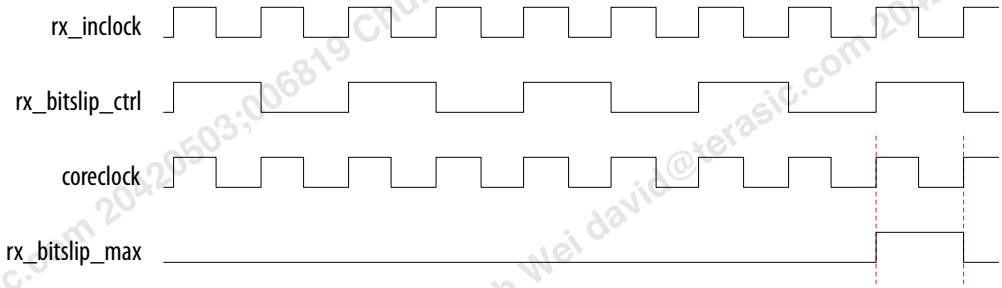


The bit slip rollover value of the data realignment circuit is set to the deserialization factor. An optional `rx_bitslip_max` status signal, available to the FPGA fabric from each channel, indicates arrival to the preset rollover point.

Preliminary Documentation – Subject to Change

Figure 10. Receiver Data Realignment Rollover

This figure shows a preset value of 4-bit cycles before the rollover occurs. The `rx_bitslip_max` signal pulses for one `coreclock` cycle to indicate that rollover has occurred.



4.1.4. Deserializer

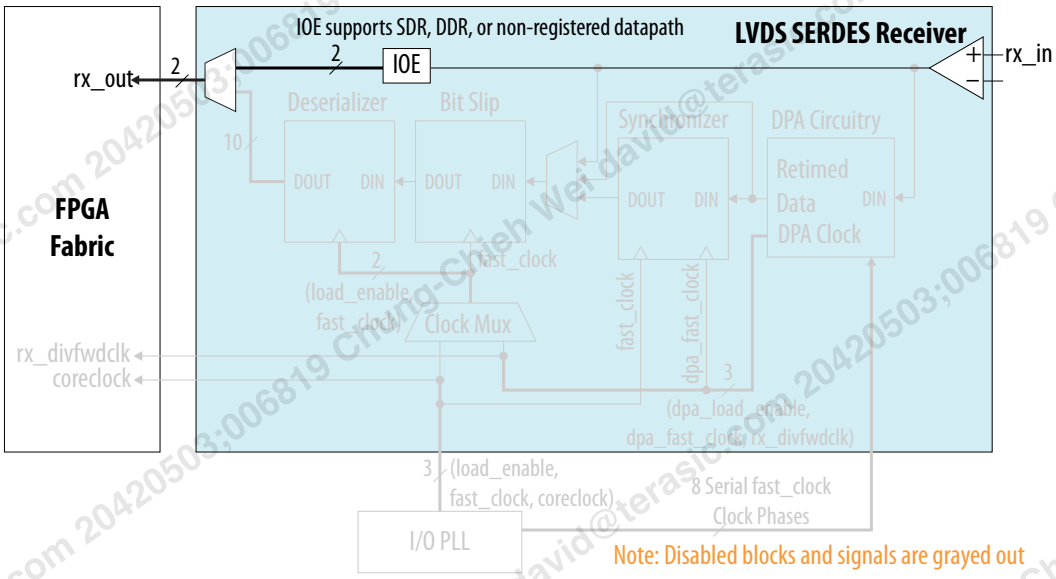
The deserializer includes shift registers and parallel load registers. The deserializer sends a maximum of 8 bits to the internal logic. You can statically set the deserialization factor from ×4 to ×8 in the LVDS SERDES Intel FPGA IP parameter editor.

The I/O element (IOE) contains two data input registers. Each data input register can operate in double data rate (DDR) or single data rate (SDR) mode. Use the GPIO Intel FPGA IP to bypass the serializer and operate in DDR and SDR modes.

If you bypass the deserializer, you cannot use the DPA block and data realignment circuit.

Figure 11. Deserializer Bypass

This figure shows the deserializer bypass path.



Preliminary Documentation – Subject to Change

Table 9. SDR and DDR Receiver Modes

Mode	Description
SDR (×1)	• The IOE data width is 1 bit. • Deserialization factor of 1. • Registered input path requires a clock. • Data is passed directly through the IOE.
DDR (×2)	• The IOE data width is 2 bits. • Deserialization factor of 2. • The GPIO IP 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.

4.2. Clocking the LVDS SERDES Receivers

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

The synchronizer circuit compensates for any phase difference between the DPA clock and the data realignment block. When necessary, the data realignment circuitry, which you control, inserts a single or multiple bits of latency in the serial bit stream to align the data to the word boundary.

The physical medium connecting the transmitter and receiver SERDES channels may introduce skew between the serial data and the source-synchronous clock. The instantaneous skew between each SERDES channel and the clock also varies with the jitter on the data and clock signals as seen by the receiver.

The different modes provide different options to compensate the skew between the source synchronous or reference clock, and the serial data:

- Non-DPA mode—you can statically select the optimal phase between the source synchronous clock and the received serial data.
- DPA mode—the DPA circuitry automatically selects the best phase 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.

4.2.1. Receiver Input Clock Parameters Settings

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

Preliminary Documentation – Subject to Change

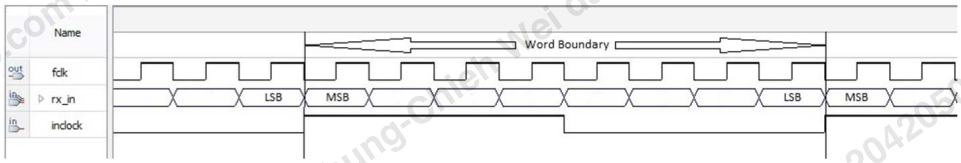
Related Information

LVDS SERDES Intel FPGA IP Receiver Settings on page 36

4.2.1.1. 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 12. 0° Edge-Aligned inclock ×8 Deserializer Waveform with a 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.

Equation 1. Maximum Phase Shift Value

This equation determines the maximum phase shift value.

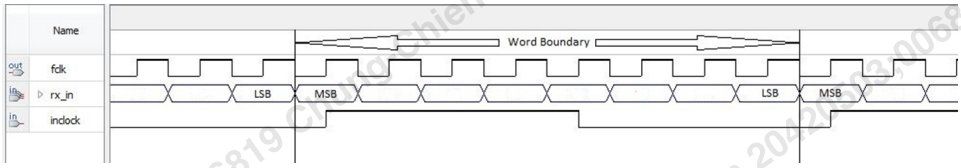
(Number of fast_clock periods per inclock period × 360) – 1

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

4.2.1.2. Center-Aligned inclock to rx_in

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

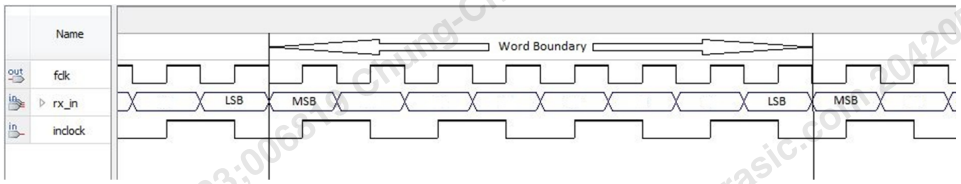
Figure 13. 180° Center-Aligned inclock ×8 Deserializer Waveform with a Single Rate Clock



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 14. 180° Center Aligned `inclock` ×x8 Deserializer Waveform with a DDR Clock



4.3. LVDS SERDES Receiver Modes

Intel Agilex 5 devices support three receiver modes.

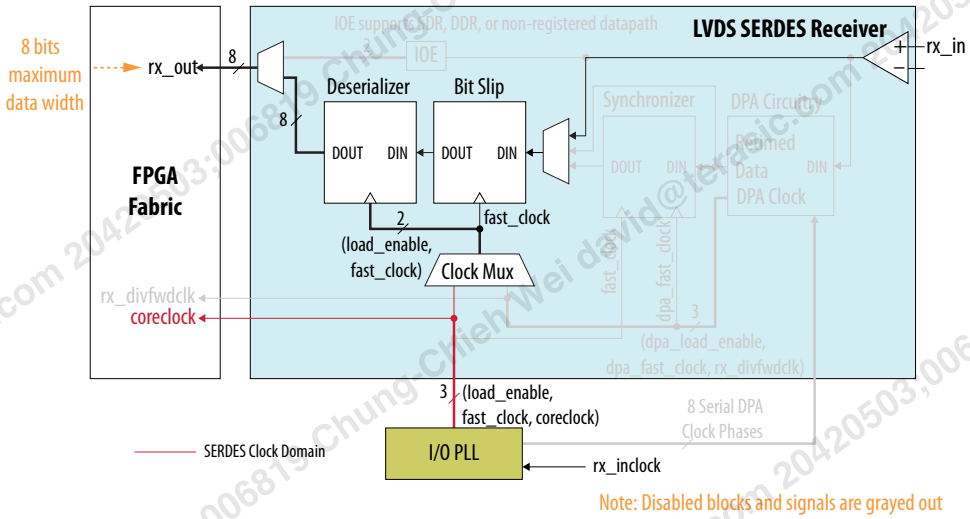
- Non-DPA mode
- DPA mode
- Soft-CDR mode

4.3.1. Non-DPA Mode

The non-DPA mode disables the DPA and synchronizer blocks. The receiver registers the input serial data at the rising edge of the serial `fast_clock` clock.

The I/O PLL generates the `fast_clock` clock signal. The `fast_clock` signal clocks the data realignment and deserializer blocks.

Figure 15. Receiver Data Path Block Diagram—Non-DPA Mode



4.3.2. DPA Mode

The DPA block selects the best possible `dpa_fast_clock` from the eight fast clock signals generated by the I/O PLL.

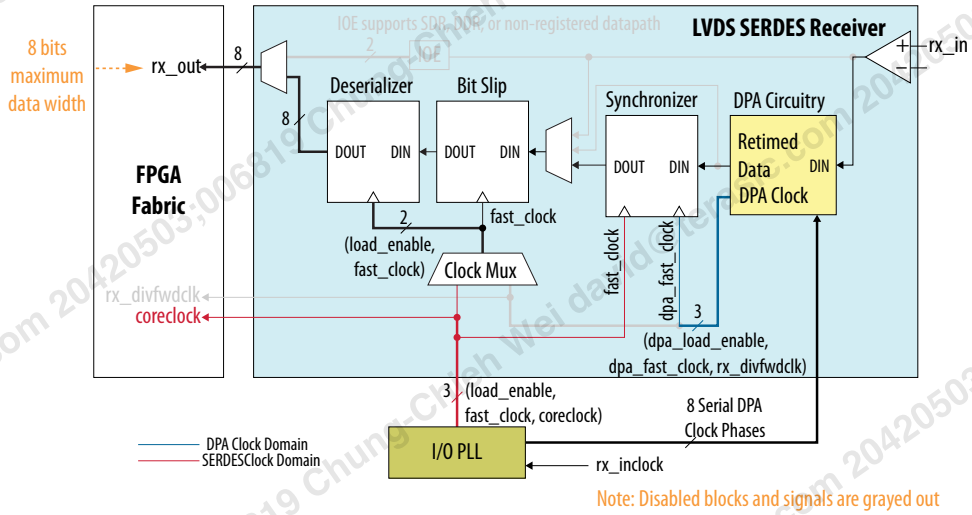
The receiver uses these serial clock signals for the following functions:

- `dpa_fast_clock`— writing serial data into the synchronizer
- `fast_clock`—reading serial data from the synchronizer, data realignment, and deserializer blocks

In DPA mode, the DPA FIFO synchronizes the retimed data to the LVDS SERDES clock domain. The DPA clock may shift the phase during the initial lock period. To avoid data run-through conditions caused by the FIFO write pointer creeping up to the read pointer, hold the FIFO in reset state until the DPA locks.

Figure 16. Receiver Data Path Block Diagram—DPA Mode

In this figure, all the receiver hardware blocks are active.

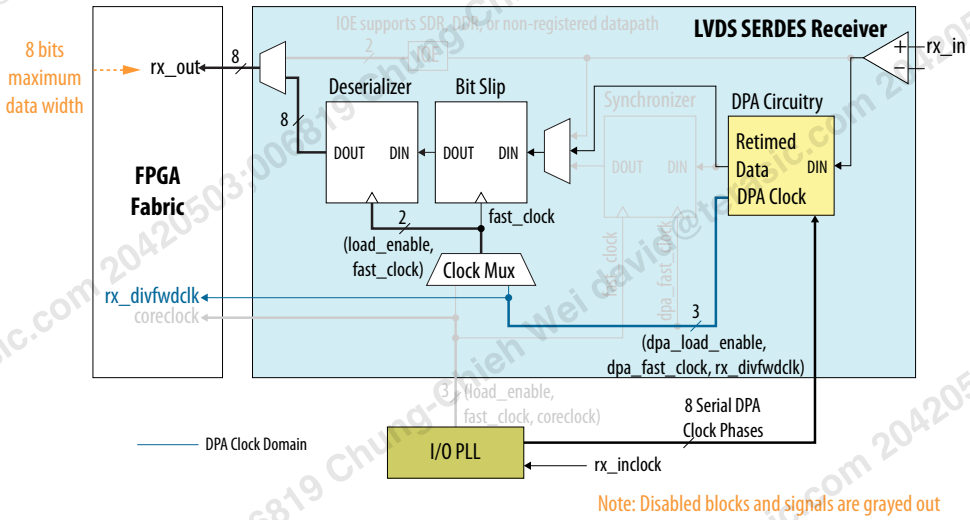


Note: In DPA mode, you can place receiver channels of a SERDES instance in both I/O sub-banks. However, you must drive the channels in each sub-bank with different PLLs.

4.3.3. Soft-CDR Mode

The Intel Agilex 5 SERDES channel offers the soft-CDR mode to support the GbE and SGMII protocols. A receiver PLL uses the local clock source for reference.

Figure 17. Receiver Data Path Block Diagram—Soft-CDR Mode



In the soft-CDR mode, the synchronizer block is inactive. The DPA circuitry selects an optimal DPA clock phase to sample the data. The receiver uses the selected clock for bit slip operation and deserialization.

Additionally, the DPA block divides the selected DPA clock by the deserialization factor and forwards it as `rx_divfwdclk` together with the deserialized data to the FPGA fabric. The `rx_divfwdclk` clock signal resides in the periphery clock (PCLK) network.

If you use the soft-CDR mode, do not assert the `rx_dpa_reset` signal after the DPA has been 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.

In the soft-CDR mode, the `rx_dpa_locked` signal is invalid 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 deasserts when the receiver operates in the soft-CDR mode. The receiver also forwards the `rx_divfwclk` parallel clock, generated from the DPA clock, to the FPGA fabric.

Note: In the soft-CDR mode, you must place all receiver channels of a SERDES instance in one I/O sub-bank. Refer to the related information for the number of soft-CDR channels supported in each sub-bank. Refer to device pin-out files to identify the locations pin locations that support the soft-CDR mode.

Related Information

Intel Agilex 5 LVDS SERDES Architecture on page 6
Lists the number of channels with soft-CDR support in the top index and bottom index sub-banks.

Preliminary Documentation – Subject to Change

5. Intel Agilex 5 High-Speed LVDS I/O Implementation Guide

You can implement your high-speed LVDS I/O design using the LVDS SERDES Intel FPGA IP 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 the LVDS SERDES IP.

5.1. LVDS SERDES Intel FPGA IP

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

5.1.1. Release Information

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

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

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

Table 10. LVDS SERDES Intel FPGA IP Core Current Release Information

Item	Description
IP version	20.0.1
Intel Quartus Prime	23.2.1
Release Date	2023.08.11

5.1.2. LVDS SERDES Intel FPGA IP Features

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

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Preliminary Documentation – Subject to Change

The LVDS SERDES IP provides the following features for you to implement your LVDS I/O design:

- 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

5.1.3. LVDS SERDES IP Usage Modes

You can implement four usage modes using the LVDS SERDES IP.

- Transmitter—specify the **Number of TX channels** to generate the IP as a transmitter.
- Non-DPA receiver—specify the **Number of RX channels** and select the **RX Non-DPA** option to generate the IP as non-DPA receiver.
- DPA receiver—specify the **Number of RX channels** and select the **RX DPA-FIFO** option to generate the IP as DPA receiver.
- Soft CDR receiver—specify the **Number of RX channels** and select the **RX Soft-CDR** option to generate the IP as soft-CDR receiver.

Each HSIO sub-bank can support one IP instance with a maximum of 12 transmitter and 12 receiver channels. For designs with more than 12 channels, you must generate a new LVDS SERDES IP instance and place it in a new HSIO sub-bank.

Table 11. Supported Usage Modes with Number of IP Instances in an I/O Sub-Bank

This table lists the supported number of LVDS SERDES IP instances based on the usage modes and PLL configurations for one HSIO sub-bank.

Number of Channels	Usage Modes	PLL Configuration	Number of IP Instances
1–47 Maximum of 24 transmitter and receiver channels combination per sub-bank.	Transmitters and receivers	• Internal PLL—you can drive both transmitters and receivers in a single LVDS SERDES instance with an internal PLL • External PLL	2
1–23	Transmitters	External PLL	1
		Internal PLL	1
1–23	Receivers	External PLL	1
		Internal PLL	1

5.1.4. Planning the LVDS Interface

In Intel Agilex 5 devices, you must plan your LVDS interface before setting up the LVDS SERDES IP, especially the placement of the pins.

Preliminary Documentation – Subject to Change

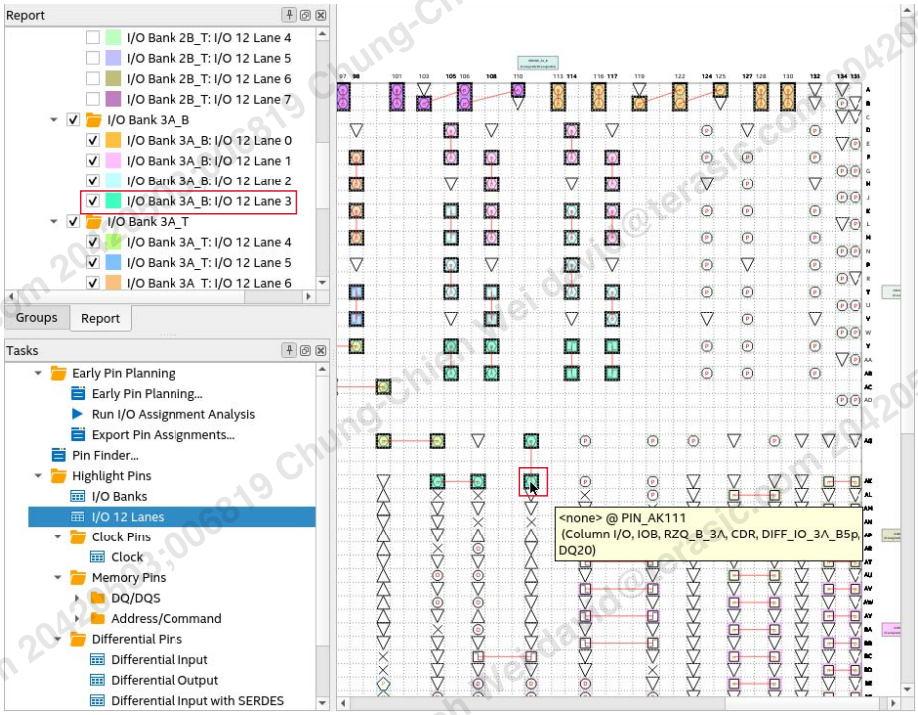
The purpose of the following steps is to identify the location of the pin using the Intel Quartus Prime Pin Planner and Chip Planner tools. Knowing the location of the pins helps you place the LVDS channel bytes in I/O lanes using the Intel Quartus Prime Interface Planner tool.

1. Plan the interface width, the combination of receiver and transmitter channels, and where to place the pins.
   - An LVDS interface can span across sub-banks but must be within the same HSIO bank. Refer to the related information for valid and invalid scenarios.
   - The channels of the LVDS interface can be any differential pin pair combinations within the bank and do not need to be sequential.
   - In each HSIO bank, reserve a pair of pins for the reference clock.
   - Refer to the related information about the placement restrictions for differential pins. The placement restrictions guidelines list the pins using the pin index numbers within the I/O bank.
2. Based on the HSIO bank and pin index numbers you plan to use, refer to the device pin out file to determine the pin names.

   Note: The following steps are optional but recommended. If you are familiar with the device layout, you can use the Intel Quartus Prime Interface Planner to find the pin location and subsequently the respective I/O lane. You need to run **Analysis & Synthesis** before you can use the Interface Planner. Refer to HSIO Pin Index Number and Respective Channel Pin Selection on page 32 to determine which pin channel to select based on the pin index number and I/O lane you plan to use.
3. In the Intel Quartus Prime Pin Planner tool, under the **Tasks** window, double-click **Highlight Pins ➤ I/O 12 Lanes**.
   The Pin Planner highlights and color-codes the pins according to the I/O lanes.
4. Using the pin name, locate the pin you want and take note of the I/O lane.

   You need these information later to select the channel pin when setting up the LVDS SERDES IP parameters.

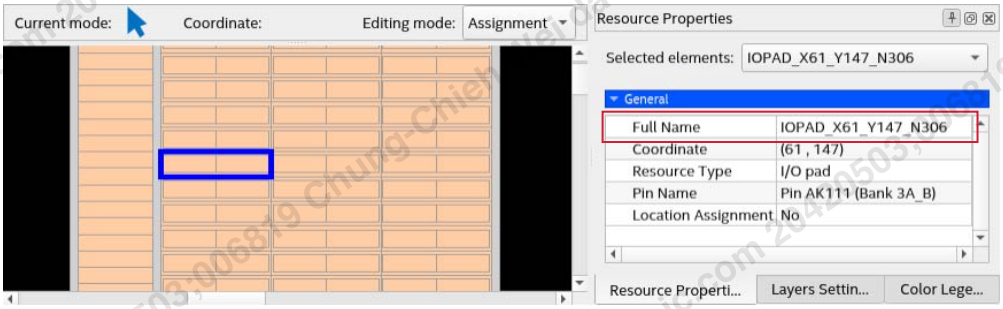
Figure 18. Pin Planner Showing the Pins According to I/O Lanes



5. Right-click the pin, and select **Locate Node ➤ Locate in Chip Planner**. The Chip Planner displays the full name and coordinate of the pin.
6. Take note of the full name of the pin.

   You need this information later to identify the I/O lane where you want to place the channel byte you select in the **Pin Settings** tab in the LVDS SERDES IP parameter editor.

Figure 19. Chip Planner Showing the Pin Full Name and Coordinate



7. Repeat from step 4 on page 25 for one pin in each I/O lane.

   You only need to locate one pin from each I/O lane.

After completing the procedure, you have a list of pins you want to use and the coordinate of at least one pin in each I/O lane. The pin coordinate helps you to identify its I/O lane in the Intel Quartus Prime Interface Planner.

The next steps are:

- Configure and generate your LVDS SERDES IP.
- Instantiate the LVDS SERDES IP in your design.
- Run **Analysis & Synthesis** on your project.
- Use the Intel Quartus Prime Interface Planner to place your LVDS channel bytes in I/O lanes.

Related Information

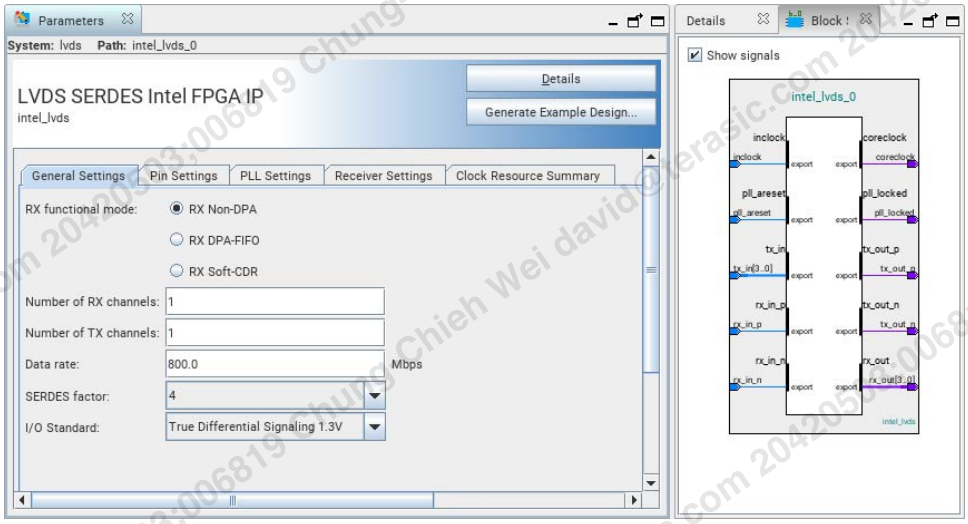
- Generating the LVDS SERDES Intel FPGA IP on page 27
- LVDS SERDES Intel FPGA IP Pin Settings on page 31
- HSIO Pin Index Number and Respective Channel Pin Selection on page 32
- Placement Restrictions for True Differential and Single-Ended I/O Standards in the Same or Adjacent HSIO Bank on page 60
- Placing Channel Bytes in I/O Lanes on page 33

5.1.5. Generating the LVDS SERDES Intel FPGA IP

Using the LVDS SERDES Intel FPGA IP parameter editor, you can customize the IP settings and generate the IP variant files, simulation testbench, and HDL instantiation template.

Before you begin, create or open an Intel Quartus Prime project. You should also plan your LVDS interface pin placements. Refer to the related information.

Figure 20. LVDS SERDES Intel FPGA IP Parameter Editor



1. In the **IP Catalog** window, double-click **LVDS SERDES Intel FPGA IP**. The **Parameter Editor** window appears.
2. Specify a top-level name for your new IP variant and click **Create**.

   Do not include space and special characters in the name and file path.
3. Set the values in the **General Settings**, **Pin Settings**, **PLL Settings**, and **Receiver Settings** tabs.

- Setting up the **Pin Settings** tab requires you to plan your LVDS interface in advance. Refer to the related information.
- The **System Messages** tab displays errors and warning for the parameters settings. Refer to the related information for the valid parameter values.

4. From the **Parameter Editor** menu, select **File ➤ Save**.
   The parameter editor saves the IP variant settings in the *<your_ip>*.ip file.
5. To generate the IP variant HDL files:
   a. Click **Generate HDL**.
      The **Generation** window appears.
   b. Specify the output file generation options and click **Generate**.
      The parameter editor generates the synthesis and simulation files as you specified, and automatically adds the .ip file of the variant to your project.
   c. Click **Close**.
6. To generate a simulation testbench:
   a. From the **Parameter Editor** menu, select **Generate ➤ Generate Testbench System**.
   b. Specify the testbench generation options and click **Generate**.
   c. Click **Close**.
7. To generate an HDL instantiation template that you can copy and paste into your text editor:
   a. From the **Parameter Editor** menu, select **Generate ➤ Show Instantiation Template**.
   b. Select the **HDL Language**.
      The code template appears in the **Example HDL** box.
   c. Click **Copy** and then click **Close**.

After generating and instantiating your IP variant, assign appropriate pins to connect the ports.

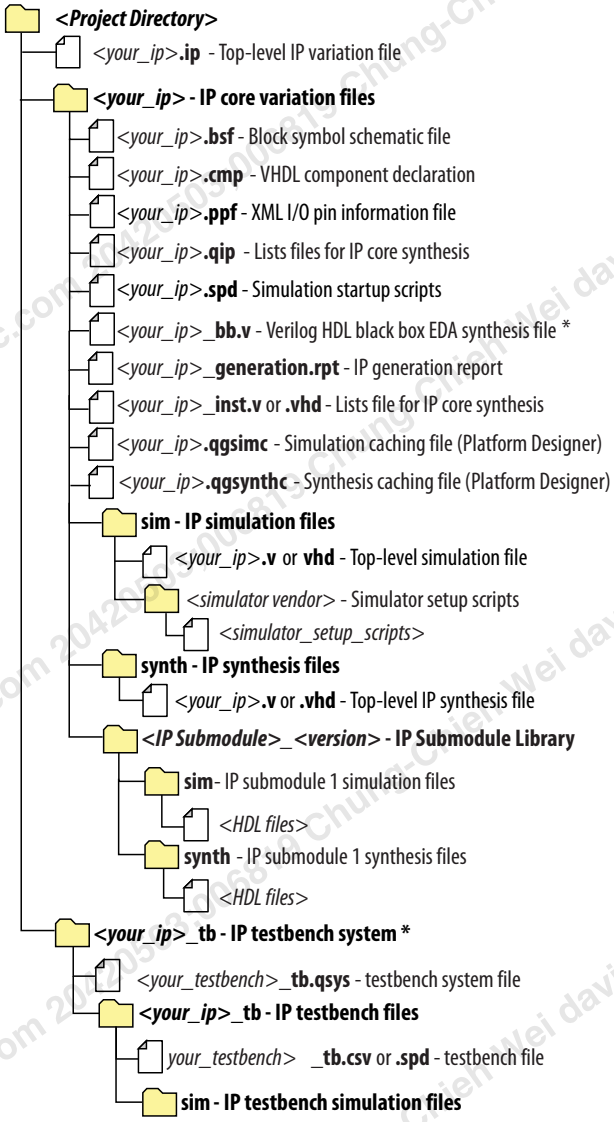
Related Information

- Planning the LVDS Interface on page 24
- LVDS SERDES Intel FPGA IP General Settings on page 31
- LVDS SERDES Intel FPGA IP Pin Settings on page 31
- LVDS SERDES Intel FPGA IP PLL Settings on page 36
- LVDS SERDES Intel FPGA IP Receiver Settings on page 36

5.1.5.1. Intel FPGA IP Generation Output

The Intel Quartus Prime software generates the following output file structure for individual IPs that are not part of a Platform Designer system.

Figure 21. Individual IP Generation Output



* If supported and enabled for your IP core variation.

Table 12. Output Files of Intel FPGA IP Generation

File Name	Description
<your_ip>.ip	Top-level IP variation file that contains the parameterization of an IP in your project. If the IP variation is part of a Platform Designer system, the parameter editor also generates a .qsys file.
<your_ip>.cmp	The VHDL Component Declaration (.cmp) file is a text file that contains local generic and port definitions that you use in VHDL design files.
<your_ip>_generation.rpt	IP or Platform Designer generation log file. Displays a summary of the messages during IP generation.

continued...

File Name	Description
<*your_ip*>.qgsimc (Platform Designer systems only)	Simulation caching file that compares the .qsys and .ip files with the current parameterization of the Platform Designer system and IP. This comparison determines if Platform Designer can skip regeneration of the HDL.
<*your_ip*>.qgsynth (Platform Designer systems only)	Synthesis caching file that compares the .qsys and .ip files with the current parameterization of the Platform Designer system and IP. This comparison determines if Platform Designer can skip regeneration of the HDL.
<*your_ip*>.csv	Contains information about the upgrade status of the IP component.
<*your_ip*>.bsf	A symbol representation of the IP variation for use in Block Diagram Files (.bdf).
<*your_ip*>.spd	Input file that ip-make-simscript requires to generate simulation scripts. The .spd file contains a list of files you generate for simulation, along with information about memories that you initialize.
<*your_ip*>.ppf	The Pin Planner File (.ppf) stores the port and node assignments for IP components you create for use with the Pin Planner.
<*your_ip*>_bb.v	Use the Verilog blackbox (_bb.v) file as an empty module declaration for use as a blackbox.
<*your_ip*>_inst.v or _inst.vhd	HDL example instantiation template. Copy and paste the contents of this file into your HDL file to instantiate the IP variation.
<*your_ip*>.regmap	If the IP contains register information, the Intel Quartus Prime software generates the .regmap file. The .regmap file describes the register map information of master and slave interfaces. This file complements the .sopcinfo file by providing more detailed register information about the system. This file enables register display views and user customizable statistics in System Console.
<*your_ip*>.svd	Allows HPS System Debug tools to view the register maps of peripherals that connect to HPS within a Platform Designer system. During synthesis, the Intel Quartus Prime software stores the .svd files for slave interface visible to the System Console masters in the .sof file in the debug session. System Console reads this section, which Platform Designer queries for register map information. For system slaves, Platform Designer accesses the registers by name.
<*your_ip*>.v <*your_ip*>.vhd	HDL files that instantiate each submodule or child IP for synthesis or simulation.
mentor/	Contains a msim_setup.tcl script to set up and run a ModelSim* simulation.
aldec/	Contains a Riviera-PRO* script rivierapro_setup.tcl to setup and run a simulation.
/synopsys/vcs /synopsys/vcsmx	Contains a shell script vcs_setup.sh to set up and run a VCS* simulation. Contains a shell script vcsmx_setup.sh and synopsys_sim.setup file to set up and run a VCS MX simulation.
/xcelium	Contains an Xcelium* Parallel simulator shell script xcelium_setup.sh and other setup files to set up and run a simulation.
/submodules	Contains HDL files for the IP submodule.
<*IP submodule*>/	Platform Designer generates /synth and /sim sub-directories for each IP submodule directory that Platform Designer generates.

5.1.6. LVDS SERDES Intel FPGA IP Parameter Settings

You can set the parameter settings for the LVDS SERDES IP in the Intel Quartus Prime software.

5.1.6.1. LVDS SERDES Intel FPGA IP General Settings

Table 13. General Settings Tab

Parameter	Value	Description
RX functional mode	• **RX Non-DPA** • **RX DPA-FIFO** • **RX Soft-CDR**	Specifies the functional mode of the receiver interface. Default is **RX Non-DPA**. These options are not available if **Number of RX channels** is `0`.
Number of RX channels	• RX Non-DPA—`0` to 47 • RX DPA-FIFO—`0` to 47 • RX Soft-CDR—`0` to 12	Specifies the number of receiver channels in the interface. Default is `1`. Place the `refclk` pin on the same I/O bank as the receiver.
Number of TX channels	`0` to 47	Specifies the number of transmitter channels in the interface.
Data rate	`600.0` to `1250.0`	Specifies the data rate (in Mbps) of a single serial channel. Default is `800.0`.
SERDES factor	• RX—**4** or **8** • TX—**4**	Select the rate of serialization and deserialization for the LVDS interface. Default is 4. *Note:* Serialization factor of 8 is available only in Intel Agilex 5 FPGAs production devices.
I/O Standard	• **True Differential Signaling 1.3V** • **True Differential Signaling 1.2V** • **True Differential Signaling 1.1V** • **True Differential Signaling 1.05V** • **SLVS 1.2V** • **SLVS 1.1V**	Select the I/O standard of the LVDS interface.

Related Information

Generating the LVDS SERDES Intel FPGA IP on page 27

5.1.6.2. LVDS SERDES Intel FPGA IP Pin Settings

The number of pin settings available depends on the **Number of RX channels** and **Number of TX channels** you specify in the **General Settings** tab. Each byte and pin pair combination must be unique in the LVDS interface across receivers and transmitters. The combination of the channel byte and pins, and which I/O lane you place the channel byte, determines the pins locations within the I/O bank.

Before configuring the **Pin Settings** tab, Intel recommends that you first plan your LVDS interface channels placement. Refer to the related information.

Table 14. Pin Settings Tab—RX Pin Settings

Parameter	Value	Description
RX channel *n* byte	**00** to **07**	Select the byte reference to use for the RX channel.

continued...

Parameter	Value	Description
		Use the Intel Quartus Prime Interface Planner tool to place the byte in an I/O lane. Refer to the related information for differential pin placement restrictions.
RX channel *n* pin	• **0001** • **0203** • **0405** • **0607** • **0809** • **1011**	Select the pin pair reference within the byte. Refer to the related information for which pin pair to select based on which pin index numbers and I/O lane you want to place the channel.

Table 15. Pin Settings Tab—TX Pin Settings

Parameter	Value	Description
TX channel *n* byte	**00** to **07**	Select the byte reference to use for the TX channel. Use the Intel Quartus Prime Interface Planner tool to place the byte in an I/O lane. Refer to the related information for differential pin placement restrictions.
TX channel *n* pin	• **0001** • **0203** • **0405** • **0607** • **0809** • **1011**	Select the pin pair reference within the byte. Refer to the related information for which pin pair to select based on which pin index numbers and I/O lane you want to place the channel.

Related Information

- HSIO Pin Index Number and Respective Channel Pin Selection on page 32
- Placement Restrictions for True Differential and Single-Ended I/O Standards in the Same or Adjacent HSIO Bank on page 60
- Intel Agilex 5 HSIO Banks, SERDES, and DPA Locations on page 6
- Planning the LVDS Interface on page 24
- Generating the LVDS SERDES Intel FPGA IP on page 27

5.1.6.2.1. HSIO Pin Index Number and Respective Channel Pin Selection

Select the channel pin in the **Pin Settings** tab based on the pin index number within the HSIO bank. The placement of the pin pair on the I/O bank depends on which I/O lane you place the specific channel byte.

Table 16. Channel Pin to Select in Pin Settings Tab for Each HSIO Bank Pin Index Number Pair

Pin Index Pair within HSIO Bank								Channel Pin to Select
Bottom Index Sub-Bank				Top Index Sub-Bank				
Lane 0	Lane 1	Lane 2	Lane 3	Lane 4	Lane 5	Lane 6	Lane 7	
0/1	12/13	24/25	36/37	48/49	60/61	72/73	84/85	0001
2/3	14/15	26/27	38/39	50/51	62/63	74/75	86/87	0203
4/5	16/17	28/29	40/41	52/53	64/65	76/77	88/89	0405

continued...

Preliminary Documentation – Subject to Change

Pin Index Pair within HSIO Bank								Channel Pin to Select
Bottom Index Sub-Bank				Top Index Sub-Bank				
Lane 0	Lane 1	Lane 2	Lane 3	Lane 4	Lane 5	Lane 6	Lane 7	
6/7	18/19	30/31	42/43	54/55	66/67	78/79	90/91	0607
8/9	20/21	32/33	44/45	56/57	68/69	80/81	92/93	0809
10/11	22/23	34/35	46/47	58/59	70/71	82/83	94/95	1011

For example, you select **01** in **RX channel 0 byte** box and **0203** in **RX channel 0 pin** box. If you place byte 01 in lane 3 of the HSIO bank, channel 0 location is pins with index numbers 38 and 39 within the bottom index sub-bank.

Related Information

- LVDS SERDES Intel FPGA IP Pin Settings on page 31
- Intel Agilex 5 HSIO Banks, SERDES, and DPA Locations on page 6
- Planning the LVDS Interface on page 24

5.1.6.2.2. Placing Channel Bytes in I/O Lanes

Use the Intel Quartus Prime Interface Planner to place the channel bytes in I/O lanes. Alternatively, you can use the `.qsf` file to assign the channel bytes to I/O lanes. The placement of the bytes determine the channel pins locations.

Before you begin:

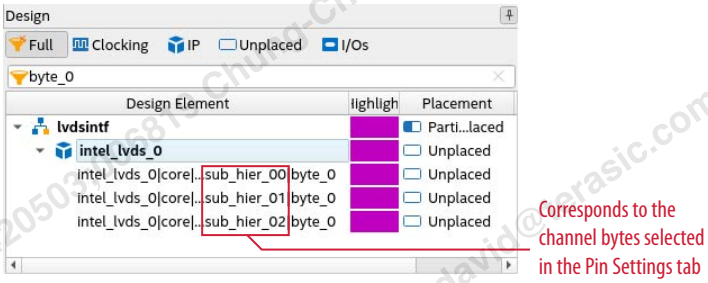
1. Optionally, plan your LVDS interface and note the full pin name and coordinate of one pin from each I/O lane. Refer to the related information.
2. Set up the **Pin Settings** tab.
3. Run **Analysis & Synthesis** for your project.

In the following steps, you use the pin name and coordinate to determine the I/O lane of the pin in the **Interface Planner** window and place the channel byte to the I/O lane.

1. Open and initialize the Intel Quartus Prime Interface Planner.
   a. From the Intel Quartus Prime menu, select **Tools ➤ Interface Planner**. The **Interface Planner** window displays.
   b. From the **Interface Planner** menu, select **Plan ➤ Initialize Interface Planner**. Wait for the initialization to complete.
   c. From the **Interface Planner** menu, select **Plan ➤ Update Plan**. The **Assignments Applied** window appears.
   d. Click **OK**.
2. Switch to the **Plan** tab.
3. Find the byte elements of your LVDS SERDES IP.
   a. In **Design Element Filter** box, enter `byte_0`.
   b. Expand your LVDS SERDES IP instance under the **Design Element** column.

   The **Design** section lists design elements that corresponds to the channel bytes you selected in the **Pin Settings** tab of the LVDS SERDES IP parameter editor.

Figure 22. Interface Planner Listing the Channel Bytes Instances



4. Right click a byte element and select **Generate Legal Locations for Selected Element**.
   The **Legal Locations** window displays.
5. Find the I/O lane of the pins you plan for your LVDS channels.
   a. Refer to the pin full name and coordinate you noted when planning the LVDS interface.

      Alternatively, you can zoom in to the HSIO bank you want in the Interface Planner Chip View and look for the pin name. Once you select the pin, the **Device Location ➤ gid name** field under the **Info** section of the Interface Planner displays the full pin name.

   b. In the **Filter** box under **Legal Locations**, enter the coordinate of the pin to make it easier to find your I/O lane.

      For example, the pin full name is "IOPAD_X61_Y147_N306". Enter X61_Y147.

   c. From the filtered list, select the I/O lane with the closest number lower than the number in the last section of the pin full name.

      For example, the number in the last section of the pin full name is "306". In the filtered list, select **BYTE_X61_Y147_N291**.
6. Right-click the selected legal location and select **Place at Selected**.

The **Placement** column for selected row under the **Design** section updates with the selected resource.

You can verify the I/O lane and channel pins placement by using the Interface Planner to locate the I/O lane number and pin index number (as shown in the Pin Planner).

Related Information

Planning the LVDS Interface on page 24

5.1.6.2.3. Locating the I/O Lane and Channel Pin through the Interface Planner

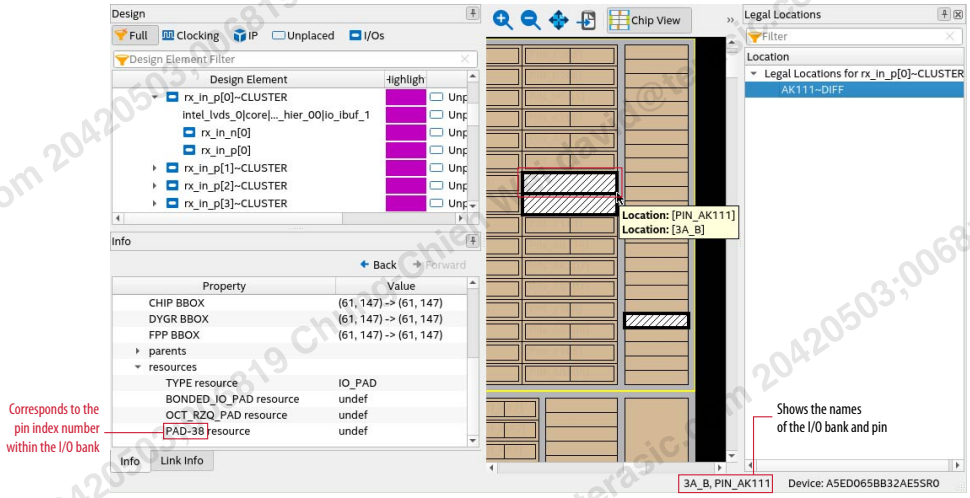
After you have assigned the channel byte to an I/O lane, you can refer to information in the Interface Planner to verify the I/O lane number and pin index numbers.

1. Select the channel byte design element (already placed) under the **Design** section.
2. Refer to the **Device Location ➤ resources** field in the **Info** tab under the **Info** section.

The number in the byte resource corresponds to the I/O lane number shown in the Pin Planner. For example, if the resource is **BYTE-3**, this corresponds to **I/O Lane 3**.

3. To determine the I/O pins of the LVDS channels and the respective I/O bank:
   a. Clear the **Design Element Filter** box under the **Design** section.
   b. Switch to the **Link Info** tab.
   c. Click the hyperlink for the channel in the **Fanin ➤ IO_CLUSTER** category.
      For example, to verify the pins for receiver channel 0, click **rx_in_p[0]~CLUSTER**.
      The list under the **Design** section selects the design channel design element.
   d. Clear the **Filter** box under the **Legal Locations** section.
   e. Right-click the selected design element row and select **Generate Legal Locations for Selected Element**.
      The **Legal Locations** section displays the differential pin pair.
   f. Right-click the differential pin pair in the **Legal Locations** section and select **Zoom to Selected**.
      The chip view zooms to the pin pair and highlight them.
   g. Hover your mouse cursor at the edge of the selected I/O pad to view the bank and pin names.
4. Click at the edge of the selected I/O pad to display the resource property under the **Info** section.
   The **Device Location ➤ resources** category displays the pad number that corresponds to the pin index number as shown in the Pin Planner and pin out files.

Figure 23. Interface Planner Showing the Bank Name, Pin Name, and Pin Index Number



You can compare how the assigned pins correspond to your channel selection in the **Pin Settings** tab of the LVDS SERDES IP parameter editor. For example, if you selected pin pair "0203" as the channel for byte 0, and then placed byte 0 to I/O lane 3 of bank 3A_B, the pins assigned to the channel has the index numbers as listed in Table 16 on page 32.

5.1.6.3. LVDS SERDES Intel FPGA IP PLL Settings

Table 17. PLL Settings Tab

Parameter	Value	Description
Use external PLL	• On • Off	Turn on to use an external PLL: • The IP does not instantiate a local PLL. • The IP creates a series of clock connections with the "ext" prefix. Connect these ports to an externally generated PLL. • For details about how to configure the external PLL, refer to the **Clock Resource Summary** tab of the parameter editor. Default is Off. 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.
Desired inclock frequency	—	Specifies the `inclock` frequency in MHz. Default is `100.0`.
Actual inclock frequency	—	Displays the closest `inclock` frequency to the desired frequency that can source the interface. The displayed value changes according to the **Desired inclock frequency** parameter value.
FPGA/PLL speed grade	—	Displays the FPGA/PLL speed grade, which determines the operation range of the PLL. The displayed value is based on the device selected in your project.

Related Information

Generating the LVDS SERDES Intel FPGA IP on page 27

5.1.6.4. LVDS SERDES Intel FPGA IP Receiver Settings

The parameter options in the **Receiver Settings** tab are available if you select the **RX Non-DPA**, **RX DPA-FIFO**, or **RX Soft-CDR** functional mode in the **General Settings** tab.

Table 18. 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). Default is Off. Every assertion of the `rx_bitslip_ctrl` signal adds one bit of serial latency to the data path of the specified channel. *Note:* You must enable this parameter for the IP simulation driver to function correctly.
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. Default is Off. This setting is available if you turn on **Enable bitslip mode**.
Enable rx_bitslip_max port	• On • Off	Turn on to expose the `rx_bitslip_max` port (one output per channel). Default is Off.

continued...

Preliminary Documentation – Subject to Change

Parameter	Value	Description
		When asserted, the next rising edge of `rx_bitslip_ctrl` resets the latency of the bit slip to zero. This setting is available if you turn on **Enable bitslip mode**.

Table 19. 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. (Formerly known as `rx_reset`.) Default is Off. This setting is available if you select **RX DPA-FIFO** or **RX Soft-CDR** in **RX functional mode**.
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. Default is Off. This setting is available if you select **RX DPA-FIFO** in **RX functional mode**.
Enable rx_dpa_hold port	• On • Off	Turn on to expose the `rx_dpa_hold` input port (one input per channel). If set high, the DPA logic in the corresponding channel does not switch sampling phases. (Formerly known as `rx_dpll_hold`.) Default is Off. This setting is available if you select **RX DPA-FIFO** in **RX functional mode**.

Table 20. 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 `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.
RCCS (ps)	—	Specifies the RCCS value in picoseconds. Depending on the order the Intel Quartus Prime software reads the project `.sdc` files, this value may override any RCCS value you specify in the `.sdc` file. To avoid uncertainty, Intel recommends that you specify the RCCS value in only one of the locations.

Related Information

- Receiver Input Clock Parameters Settings on page 18
- Generating the LVDS SERDES Intel FPGA IP on page 27

5.1.6.5. LVDS SERDES Intel FPGA IP Clock Resource Summary

The **Clock Resource Summary** tab lists the required frequencies, phase shifts, duty cycles of the required clocks, instructions for connections, and compensation mode that you need to set in the IOPLL Intel FPGA IP.

5.1.7. LVDS SERDES Intel FPGA IP Signals

Table 21. Common LVDS SERDES IP 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 and PLL. *Note:* This signal must always be connected to the reset logic.
pll_locked	1	Output	Control	Asserts when internal PLL locks.

Table 22. 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_p • rx_in_n	*N*	Input	Data	LVDS serial input data differential pair.
rx_bitslip_reset	*N*	Input	Reset	Asynchronous, active-high reset to the clock-data alignment circuitry (bit slip).
rx_bitslip_ctrl	*N*	Input	Control	• Positive-edge triggered increment for bit slip circuitry. • Each assertion adds one bit of latency to the received bit stream.
rx_dpa_hold	*N*	Input	Control	• Asynchronous, active-high signal that prevents the DPA circuitry from switching to a new clock phase on the target channel. — Held high—selected channels hold their current phase setting. — Held low—the DPA block on selected channels monitors the phase of the incoming data stream continuously and selects a new clock phase when needed. • Applicable in DPA-FIFO mode only.
rx_dpa_reset	*N*	Input	Reset	• Asynchronous, active-high reset to DPA blocks. • Minimum pulse width: one parallel clock period. • Applicable in DPA-FIFO and soft-CDR modes only.
rx_fifo_reset	*N*	Input	Reset	• Asynchronous, active-high reset to FIFO block. • Minimum pulse width: one parallel clock period. • Applicable in DPA-FIFO mode only.
rx_out	*N*×*J*	Output	Data	Receiver parallel data output. • DPA-FIFO and non-DPA modes—synchronous to coreclock. • Soft-CDR mode—each channel has parallel data synchronous to its rx_divfwdclk.

continued...

Preliminary Documentation – Subject to Change

Signal Name	Width	Direction	Type	Description
rx_bitslip_max	N	Output	Control	• Bit slip rollover signal. • High when the next assertion of rx_bitslip_ctrl resets the serial bit latency to 0.
coreclock	1	Output	Clock	• Core clock for RX interfaces provided by the LVDS SERDES Intel FPGA IP. • Applicable in Non-DPA and DPA-FIFO modes only.
rx_divfwdclk	N	Output	Clock	The per channel and divided clock with the ideal DPA phase. • This is the recovered slow clock for a given channel. • Applicable in soft-CDR mode only. The rx_divfwdclk signals may not be edge-aligned with each other because each channel may have a different ideal sampling phase. Each rx_divfwdclk must drive the core logic with data from the same channel.
rx_dpa_locked	N	Output	Control	Asserted when the DPA block selects the ideal phase. • Driven by the LVDS SERDES IP. • Asserts when the signal settles on an ideal phase for that given channel. • Deasserts in one of these conditions: — The DPA moves one phase. — The DPA moves two phases in the same direction. • Applicable in DPA-FIFO and soft-CDR modes only. Ignore all toggling of the rx_dpa_locked signal after rx_dpa_hold asserts.

Table 23. LVDS SERDES IP 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_p • tx_out_n	N	Output	Data	Serial output data differential pair.
coreclock	1	Output	Clock	Drives the core logic feeding the serializer.

Table 24. External PLL Signals for LVDS SERDES IP

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_outclock_periph[1:0]	2	Input	Clock	Fast clock. • Used for serial data transfer. • Required in all modes. Connect both ports to the IOPLL Intel FPGA IP lvds_clk[1:0] ports. For more information about connecting this port with the signal from the IOPLL Intel FPGA IP, refer to the related information.
ext_pll_1_outclock2	1	Input	Clock	• Input clock to the LVDS SERDES Intel FPGA IP. • Required for RX soft-CDR mode.

continued...

Preliminary Documentation – Subject to Change

Signal Name	Width	Direction	Type	Description
ext_phout[7:0]	8	Input	Clock	• Provides the VCO clocks to the DPA circuitry for optimal phase selection. • Required for all functional modes For more information about connecting this port with the signal from the IOPLL IP core, refer to the related information.
ext_phout_periph	1	Input	Clock	Input for the single VCO clock routed throughout the periphery. For more information about connecting this port with the phout_periph signal from the IOPLL IP core, refer to the related information.
ext_pll_locked	1	Input	Data	PLL lock signal. This signal indicates when external PLL is locked. It does not indicate if SERDES is ready for initialization.

Related Information

- IOPLL IP Signal Interface with LVDS SERDES IP on page 41
- Placing LVDS Transmitters and Receivers with Identical Data Rates in the Same HSIO Sub-Bank on page 62

5.2. LVDS Interface with External PLL Mode

The LVDS SERDES IP parameter editor provides an option to implement the LVDS interface with the **Use External PLL** option. With this option turned on 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 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 receiver
- PLL VCO signal for the transmitter and DPA receiver modes of the LVDS SERDES IP

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

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

- In the **Settings** tab, specify the **LVDS External PLL** settings.
- In the **PLL** tab:
  - Set the **Output Clocks** settings.
  - Select the **Compensation Mode** according to the following table.

Preliminary Documentation – Subject to Change

Table 25. Compensation Mode Setting to Generate IOPLL IP

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

LVDS Functional Mode	IOPLL IP Compensation Mode
TX (without RX non-DPA), RX DPA, RX Soft-CDR	**direct**
RX non-DPA (with or without TX)	**lvds**

5.2.1. IOPLL IP Signal Interface with LVDS SERDES IP

Table 26. Signal Interface between IOPLL and LVDS SERDES IPs

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

From the IOPLL IP	To the LVDS SERDES IP Transmitter or Receiver
`outclock_periph[1:0]` (serial clock output signal) • Configure this signal using `outclk0` in the PLL. • Turn on **Enable access to I/O Bank clock ports** option under the **LVDS External PLL** section. The serial clock output can only drive `ext_outclock_periph[1:0]` on the LVDS SERDES IP transmitter and receiver. This clock cannot drive the core logic.	`ext_outclock_periph[1:0]` (serial clock input to the transmitter or receiver)
`outclk_2` (parallel clock output)	`ext_pll_1_outclock2` (core clock to the LVDS SERDES Intel FPGA IP)
`locked`	`ext_pll_locked` This signal indicates when external PLL is locked. It does not indicate if SERDES is ready for initialization.
`rst`	`pll_areset` (asynchronous PLL reset port)
`phout[7:0]` • This signal is required if `ext_vcoph[7:0]` is required. • Configure this signal by turning on **Specify VCO frequency** in the PLL and specifying the **Desired VCO Frequency** value. • Turn on **Enable access to PLL DPA output port**.	`ext_vcoph[7:0]` This signal is required for all transmitter or receiver modes.

Related Information

- LVDS SERDES Intel FPGA IP Signals on page 38
- Placing LVDS Transmitters and Receivers with Identical Data Rates in the Same HSIO Sub-Bank on page 62

5.2.2. IOPLL Parameter Values for External PLL Mode

These examples show the clocking requirements to generate output clocks for LVDS SERDES IP using the IOPLL IP. 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 interface without using the external PLL mode option. Compile the IPs 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 parameter editor and then connect the appropriate output to the LVDS SERDES IPs.

Table 27. Example: Generating Output Clocks Using an IOPLL IP (Receiver in Non-DPA Mode)

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

Parameter	outclk0 (Connects as outclk_periph[0] to the ext_outclk_periph[0] port of LVDS SERDES IP transmitter or receiver)	outclk1 (Connects as outclk_periph[1] to the ext_outclk_periph[1] port of LVDS SERDES IP transmitter or receiver)	outclk2 (Used as the core clock for the parallel data registers for both transmitter and receiver, and connects to the ext_pll_1_outclock2 port of LVDS SERDES IP)
Frequency	data rate	data rate/serialization factor	data rate/serialization factor
Phase shift	180°	[(deserialization factor – 1)/ deserialization factor] × 360°	180/serialization factor (outclk0 phase shift divided by the serialization factor)
Duty cycle	50%	100/serialization factor	50%

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

Figure 24. Phase Relationship for External PLL Interface Signals

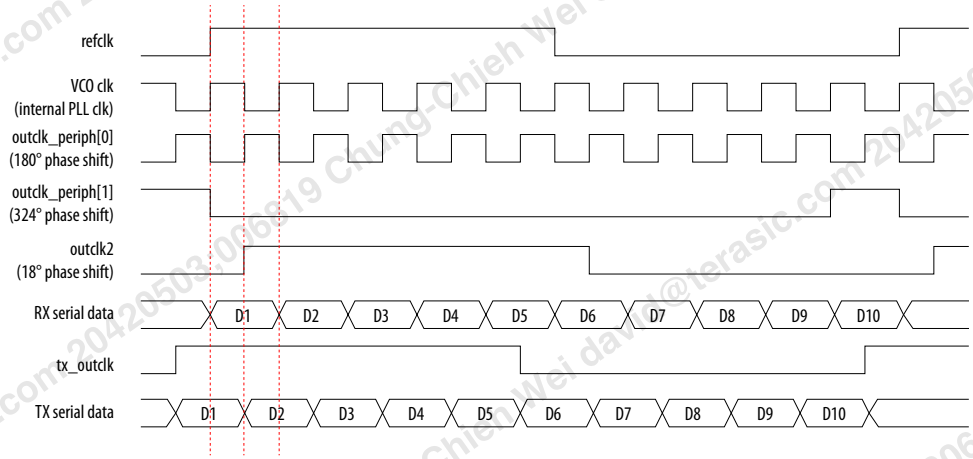


Table 28. Example: Generating Output Clocks Using an IOPLL IP (Receiver in DPA or Soft-CDR Mode)

This table lists the parameter values that you can set in the IOPLL IP parameter editor to generate four output clocks using an IOPLL IP if you are using the DPA or soft-CDR receiver.

Parameter	outclk0 (Connects as outclk_periph[0] to the ext_outclk_periph[0] port of LVDS SERDES IP transmitter or receiver)	outclk1 (Connects as outclk_periph[1] to the ext_outclk_periph[1] port of LVDS SERDES IP transmitter or receiver) Not required for the soft-CDR receiver.	outclk2 (Used as the core clock for the parallel data registers for both transmitter and receiver, and connects to the ext_pll_1_outclock2 port of LVDS SERDES IP)	VCO Frequency (Connects as phout[7:0] to the ext_phout[7:0] port of LVDS SERDES IP)
Frequency	data rate	data rate/serialization factor	data rate/serialization factor	Use the value recommended by the LVDS and IOPLL IPs
Phase shift	180°	[(deserialization factor – 1)/deserialization factor] × 360°	180/serialization factor (outclk0 phase shift divided by the serialization factor)	—
Duty cycle	50%	100/serialization factor	50%	—

Table 29. Example: Generating Output Clocks Using a Shared IOPLL IP for Transmitter and Receiver Channels (Receiver in DPA or Soft-CDR Mode)

This table lists the parameter values that you can set in the IOPLL IP parameter editor to generate six output clocks using an IOPLL IP. Use these settings if you use transmitter channels shared with receiver channels in DPA or soft-CDR mode. In this example, the transmitter and receiver interfaces are using different data rates. If the data rates are the same, you can share outclk0 and outclk1 for both the transmitter and receiver, and do not need outclk3 and outclk4.

Parameter	outclk0 (Connects as outclk_periph[0] to the ext_outclk_periph[0] port of LVDS SERDES IP receiver)	outclk1 (Connects as outclk_periph[1] to the ext_outclk_periph[1] port of LVDS SERDES IP receiver) Not required for the soft-CDR receiver.	outclk2 (Used as the core clock for the parallel data registers for both transmitter and receiver, and connects to the ext_pll_1_outclock2 port of LVDS SERDES IP)	VCO Frequency (Connects as phout[7:0] to the ext_phout[7:0] ports of LVDS SERDES IP)
	outclk3 (Connects as outclk_3 to the ext_outclock_periph[0] port of LVDS SERDES IP transmitter)	outclk4 (Connects as outclk_4 to the ext_outclock_periph[1] port of LVDS SERDES IP transmitter)		
Frequency	data rate	data rate/serialization factor	data rate/serialization factor	data rate
Phase shift	180°	[(deserialization factor – 1)/deserialization factor] × 360°	180/serialization factor (outclk0 phase shift divided by the serialization factor)	—
Duty cycle	50%	100/serialization factor	50%	—

5.2.3. Connection between IOPLL IP and LVDS SERDES IP in External PLL Mode

Figure 25. Non-DPA or DPA LVDS Receiver Interface with the IOPLL IP without LVDS Transmitter in the Same Sub-Bank

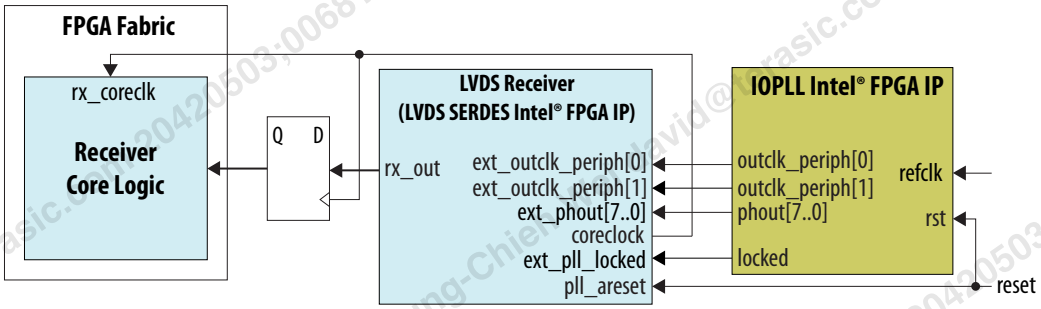


Figure 26. Non-DPA or DPA LVDS Receiver Interface with the IOPLL IP with LVDS Transmitter in the Same Sub-Bank

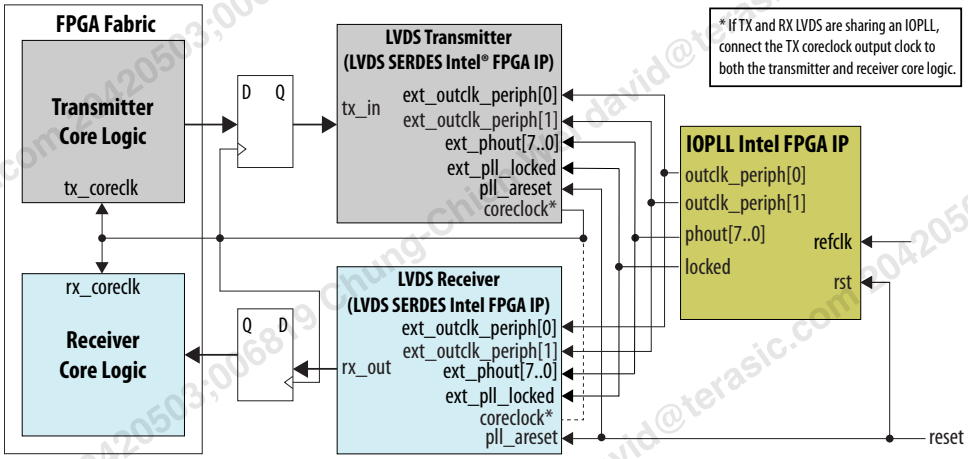


Figure 27. Soft-CDR LVDS Receiver Interface with the IOPLL IP without LVDS Transmitter in the Same Sub-Bank

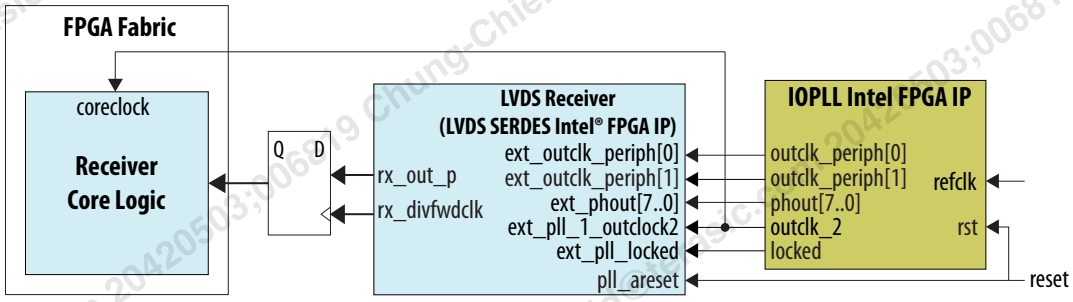


Figure 28. Soft-CDR LVDS Receiver Interface with the IOPLL IP with LVDS Transmitter in the Same Sub-Bank

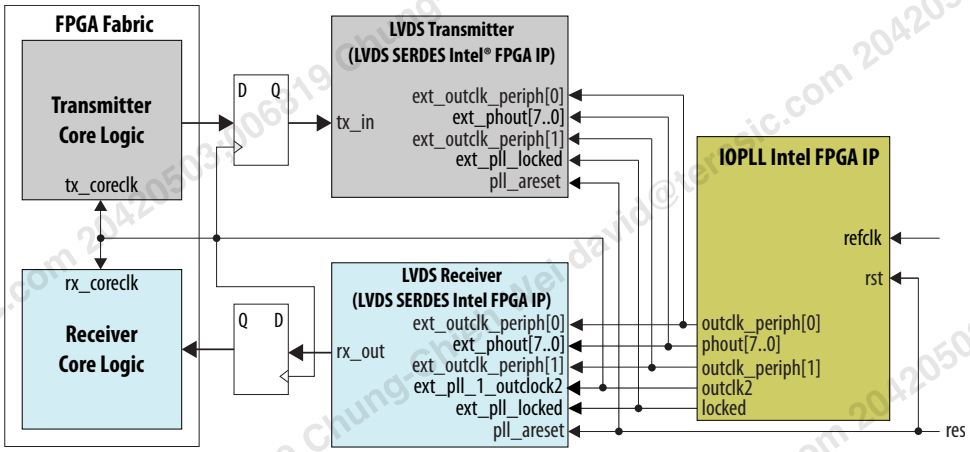
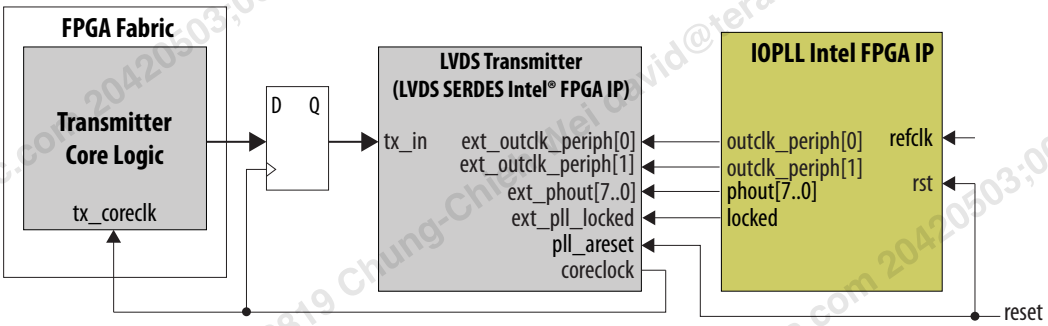


Figure 29. LVDS Transmitter Interface with the IOPLL IP



In the external PLL mode, the LVDS SERDES IP automatically turns on the `ext_pll_1_outclock2` port. If you do not connect the `ext_pll_1_outclock2` port as shown in the preceding figures, the Intel Quartus Prime compiler outputs error messages.

Related Information

Placing LVDS Transmitters and Receivers with Identical Data Rates in the Same HSIO Sub-Bank on page 62

5.3. LVDS SERDES IP 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 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 in DPA or non-DPA mode, you can perform word boundaries alignment using the bit slip control signal.

5.3.1. Initializing the LVDS SERDES IP 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, ensure that the PLL is locked to the reference clock.

Intel recommends that you follow these steps to initialize the LVDS SERDES IP 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 initialization, you can proceed to align the word boundaries (bit slip).

5.3.2. Initializing the LVDS SERDES IP 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 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 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. 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.
4. After the `rx_dpa_locked` signal asserts, assert the `rx_fifo_reset` signal for at least one parallel clock cycle.
5. 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 (bit slip).

5.3.3. 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 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.

5.3.4. 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

Preliminary Documentation – Subject to Change

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` you require 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.

5.3.4.1. Aligning Word Boundaries

After initializing the LVDS SERDES IP 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.

Preliminary Documentation – Subject to Change

6. Intel Agilex 5 LVDS SERDES Timing

The true differential 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.

6.1. LVDS SERDES Intel FPGA IP Timing

Use the Intel Quartus Prime software to generate the required timing constraint to perform proper timing analysis of the LVDS SERDES IP in Intel Agilex 5 devices.

Table 30. LVDS SERDES IP Timing Components

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

Table 31. LVDS SERDES IP Timing Constraint Files

This table lists the timing files generated by the LVDS SERDES IP. Use these files for successful timing analysis of the LVDS SERDES IP. You can find these files in the `<variation_name>/intel_lvds_core10_ph2_191/synth` directory.

File Name	Description
`<variation_name>_intel_lvds_core10_ph2_191_<random_id>.sdc`	This **.sdc** file allows the Intel Quartus Prime Fitter to optimize timing margins with timing-driven compilation. The file also allows the Timing Analyzer to analyze the timing of your design. The IP uses the **.sdc** for the following operations: • Creating clocks on PLL inputs • Creating generated clocks • Calling `derive_clock_uncertainty` • Creating proper multi-cycle constraints You can locate this file in the **.qip** generated during IP generation.
`sdc_util.tcl`	This **.tcl** file is a library of functions and procedures that the **.sdc** uses.

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Preliminary Documentation – Subject to Change

6.1.1. I/O Timing Analysis

Table 32. Timing Analysis for Different LVDS SERDES IP Modes

Mode	Timing Analysis
TX	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. The TCCS report lists the 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 transmitter output clock—the difference between the fastest and slowest data output transitions, including the T_{CO} variation and clock skew.

RX Non-DPA	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>`
	Example:
	`set ::RCCS 0.0`

RX DPA-FIFO	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.
RX Soft-CDR	

6.1.1.1. Obtaining RSKM Report

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

Before you begin, compile your project and ensure that the compilation is successful.

1. From the Intel Quartus Prime menu, select **Tools ➤ Timing Analyzer**. The **Timing Analyzer** window appears.
2. In the **Task** pane of the Timing Analyzer window, double-click **Update Timing Netlist**.
3. From the Timing Analyzer menu, select **Reports ➤ IP Specific ➤ Report RSKM**.

6.1.1.2. Obtaining TCCS Report

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

Before you begin, compile your project and ensure that the compilation is successful.

1. From the Intel Quartus Prime menu, select **Tools ➤ Timing Analyzer**. The **Timing Analyzer** window appears.
2. In the **Task** pane of the Timing Analyzer window, double-click **Update Timing Netlist**.
3. From the Timing Analyzer menu, select **Reports ➤ IP Specific ➤ Report TCCS**.

6.1.2. FPGA Timing Analysis

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

Table 33. 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	<*lvds_instance_name*>|cpa_clk
LVDS fast clock	• Receiver—<*lvds_instance_name*>|p2c_fa_div_clk_<*byte_num*> • Transmitter—<*lvds_instance_name*>|c2p_fa_div_clk_<*byte_num*>

Table 34. Clock for the Receiver in Soft-CDR Mode

Clock	Clock Name
Core clock	<*lvds_instance_name*>|cpa_clk
DPA fast clock	<*lvds_instance_name*>|dpa_core_clk_<*byte_num*>_<*pin_num*>

6.1.3. 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.

The Intel Quartus Prime software derives some of the SERDES constraints from the PLL clocks. Therefore, the Intel Quartus Prime software must generate the external PLL clock settings before the LVDS SERDES IP clock settings. In the `.qsf` of your project, ensure that the line for the `.ip` file of the IOPLL IP appears before the line for the `.ip` file of the LVDS SERDES IP.

6.2. LVDS SERDES Source-Synchronous Timing Budget

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.

6.2.1. Transmitter Channel-to-Channel Skew

The receiver skew margin calculation uses the transmitter channel-to-channel skew (TCCS)—an important parameter based on the FPGA 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 SERDES transmitters, the Timing Analyzer provides the TCCS value in the TCCS report (`report_TCCS`) in the Intel Quartus Prime compilation report. The TCCS report lists the TCCS values for serial output ports.
- You can also get the TCCS value from the device datasheet.

Preliminary Documentation – Subject to Change

Perform PCB trace compensation to adjust the trace length of each SERDES channel to improve channel-to-channel skew when interfacing with non-DPA receivers at data rate above 840 Mbps.

The Intel Quartus Prime Fitter report lists the amount of delay you must add to each trace.

The Transmitter/Receiver Package Skew Compensation report lists the recommended trace delay numbers. Using these numbers, you can manually compensate the skew on the PCB board trace to reduce the channel-to-channel skew and meet the timing budget between the SERDES channels.

6.2.2. Receiver Skew Margin

Different modes of SERDES receivers use different specifications, which determine the ability to sample the received serial data correctly.

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

Equation 2. RSKM Equation

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

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

Table 35. RSKM Equation Conventions

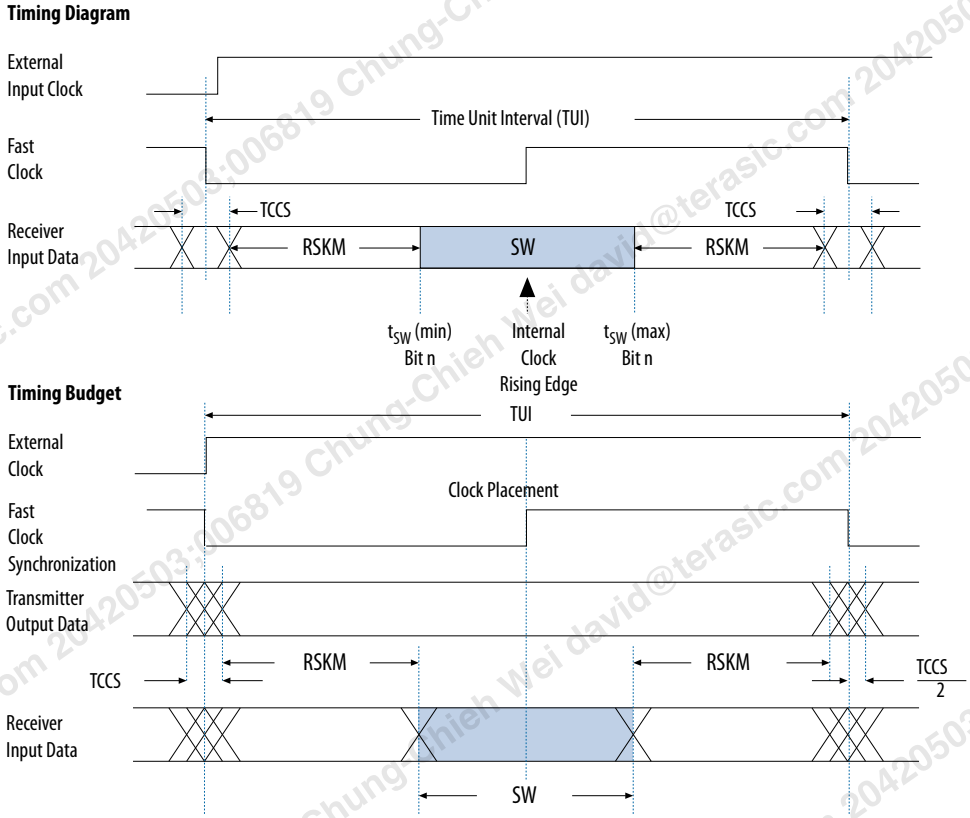
Symbol	Description
RSKM	The timing margin between the clock input of the receiver and the data input sampling window, and the jitter induced from core noise and I/O switching noise.
Time unit interval (TUI)	The time period of the serial data.
SW	The period of time that the input data must be stable to 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 SERDES receiver can sample the data:

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

Figure 30. Differential High-Speed Timing Diagram and Timing Budget



Example 1. RSKM Calculation Example

This example shows the RSKM calculation for Intel Agilex 5 devices at 1 Gbps data rate with a 200 ps board channel-to-channel skew.

Equation 3. RSKM Calculation

$$TCCS = 100 \text{ ps} \quad SW = 300 \text{ ps} \quad TUI = 1000 \text{ ps}$$

$$\begin{aligned} \text{Total } RCCS &= TCCS + \text{board channel-to-channel skew} \\ &= 100 \text{ ps} + 200 \text{ ps} \\ &= 300 \text{ ps} \end{aligned}$$

$$\begin{aligned} RSKM &= \frac{TUI - SW - RCCS}{2} \\ &= \frac{1000 \text{ ps} - 300 \text{ ps} - 300 \text{ ps}}{2} \\ &= 200 \text{ ps} \end{aligned}$$

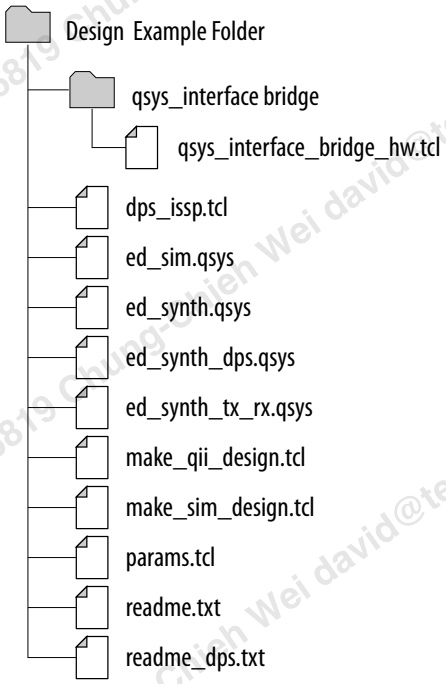
The non-DPA receiver works correctly if the RSKM is greater than 0 ps after deducting transmitter jitter.

7. LVDS SERDES Intel FPGA IP Design Examples

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

Figure 31. Source Files in the Generated Design Example Directory



7.1. LVDS SERDES IP 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.

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

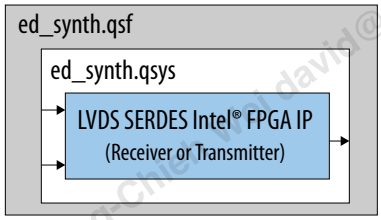
ISO 9001:2015 Registered

Preliminary Documentation – Subject to Change

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

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

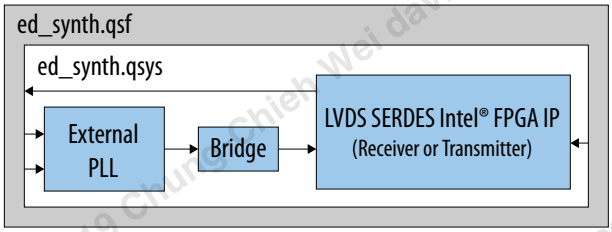
Figure 32. Basic LVDS SERDES IP System with Internal PLL



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

Figure 33. LVDS SERDES IP System with External PLL

In this figure, a qsys_interface_bridge provides Platform Designer connections between the IOPLL IP and the LVDS SERDES IP. 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 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
```


Preliminary Documentation – Subject to Change

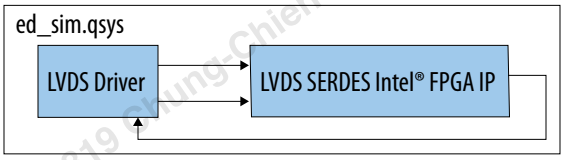
7.2. LVDS SERDES IP Simulation Design Example

The simulation design example uses your LVDS SERDES IP 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.

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

Figure 34. LVDS SERDES IP 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.

8. Intel Agilex 5 LVDS SERDES Design Guidelines

Different functions of the SERDES have different guidelines, placement restrictions, connection requirements, and clocking requirements.

8.1. Use PLLs in Integer PLL Mode for LVDS

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

8.2. Use High-Speed Clock from PLL to Clock SERDES Only

The high-speed clock generated from the PLL is intended to clock the 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.

8.3. Pin Placement for Differential Channels

Each Intel Agilex 5 HSIO sub-bank contains its own PLL. The PLL can drive all receiver and transmitter channels in the same bank. You must use the dedicated clock pins to drive the LVDS PLLs. Each bank supports driving two PLLs from the same bank with a single reference clock.

Pins Arrangement in the HSIO Bank

In the device pin out files, the following pin index numbers indicate the location of the pins in a single HSIO bank:

- 0 to 47—bottom index sub-bank
- 48 to 95—top index sub-bank

PLLs Driving DPA-Enabled Differential Channels

- For differential channels, the PLL can drive all channels in the same I/O bank but cannot drive channels in other 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.
- 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.

Related Information

I/O PLLs Driving LVDS Transmitter and Receiver Channels on page 58

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Preliminary Documentation – Subject to Change

8.3.1. I/O PLLs Driving LVDS Transmitter and Receiver Channels

The following figures show valid and invalid scenarios of the HSIO bank PLLs driving the transmitter and receiver channels.

Figure 35. I/O PLL Driving Transmitter or Receiver Channels in the Same Sub-Bank

The I/O PLL can drive the transmitter or receiver channels in its own sub-bank.

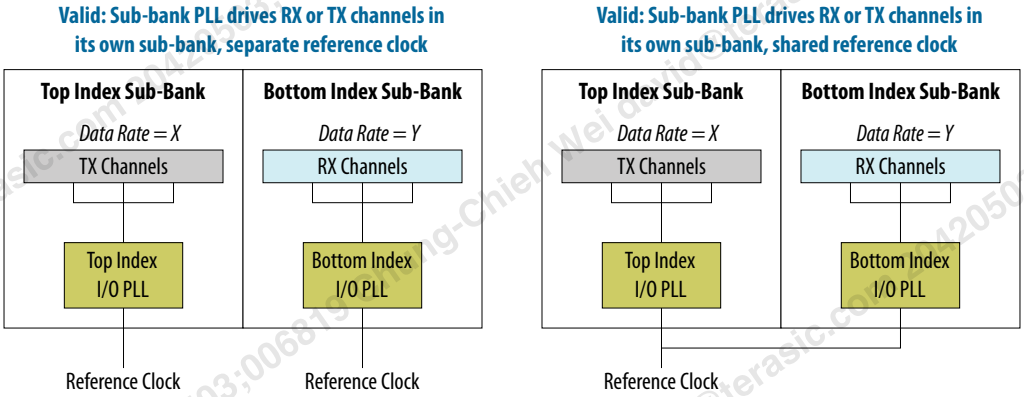
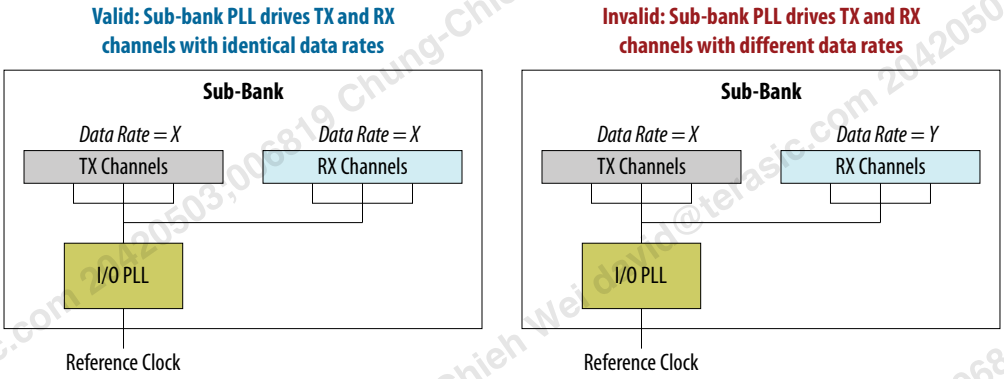


Figure 36. I/O PLL Driving Mixed Receiver and Transmitter Channels in the Same Sub-Bank

The I/O PLL can drive transmitter and receiver channels in the same sub-bank if the channels have identical data rates. If the transmitter and receiver channels have different data rates, you need another PLL.



Preliminary Documentation – Subject to Change

Figure 37. I/O PLL Driving Transmitter or Receiver Channels Across Sub-Banks

The I/O PLL can drive channels across sub-banks if the channels have identical data rates.

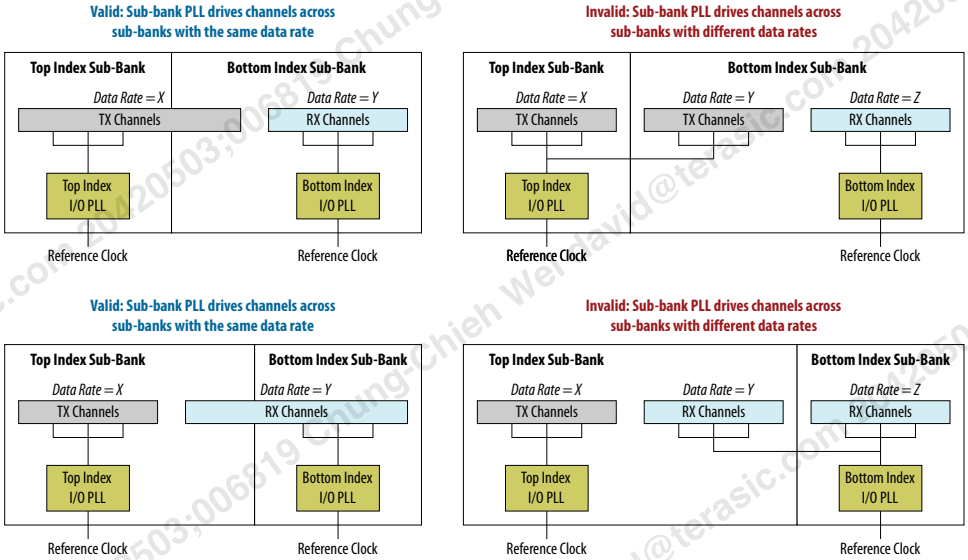
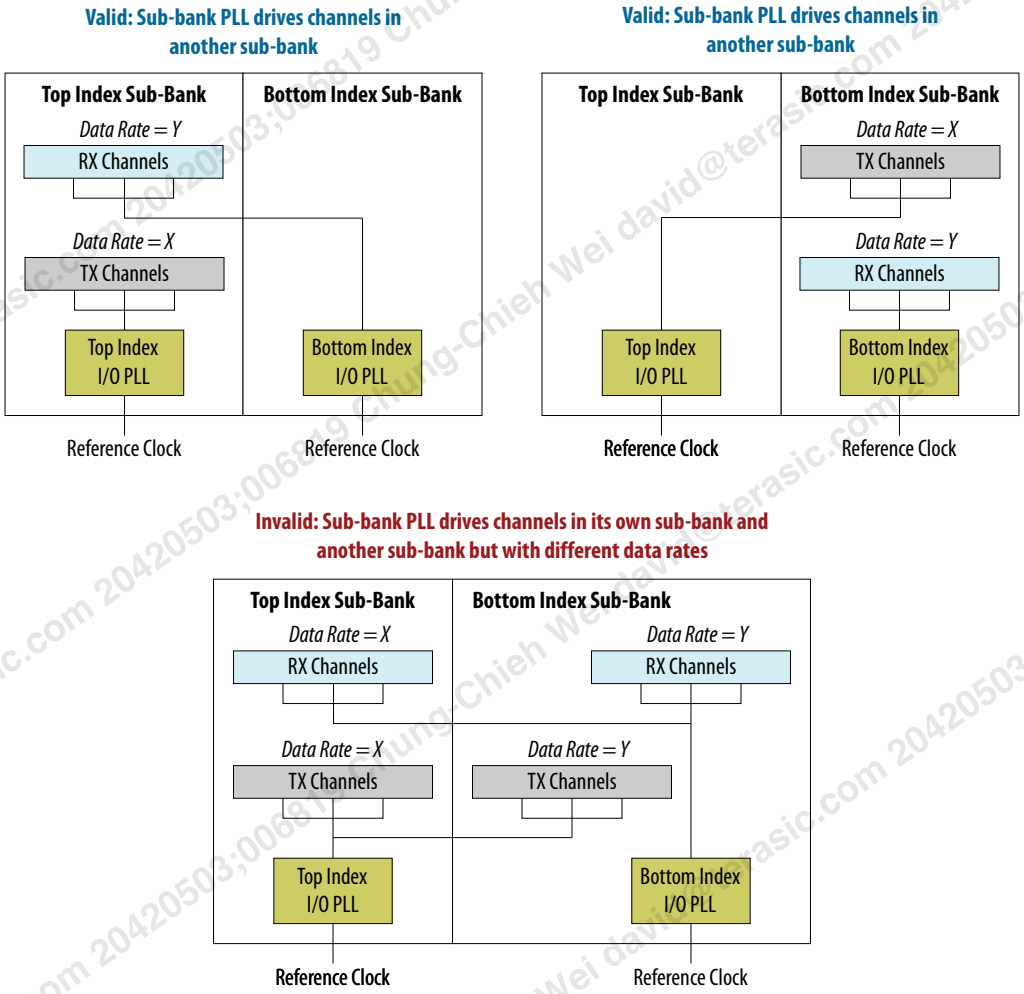


Figure 38. I/O PLL Driving Transmitter or Receiver Channels in Another Sub-Bank

The I/O PLL can drive channels in another sub-bank. In this valid scenario, you may have to use a reference clock tree. However, the I/O PLL cannot drive channels in different sub-banks with different data rates.



8.3.2. Placement Restrictions for True Differential and Single-Ended I/O Standards in the Same or Adjacent HSIO Bank

If you use true differential I/O standards and single-ended I/O standards in the same or adjacent HSIO banks, adhere to the following placement guidelines.

- Do not place true differential and toggling single-ended I/O standards in the combinations of locations listed in the following tables.
- From version 23.4, the Intel Quartus Prime software issues the following errors:
  - Compilation error—violation of the same bank placement restriction.
  - Critical warning—assignment of true differential I/O standards to pin pairs with the following pin index numbers: 0 and 1, 6 and 7, 88 and 89, and 94 and 95.

Table 36. Restricted Pin Placement Combinations for True Differential and Single-Ended I/O Standards in the Same HSIO Bank

This table lists the combinations of pins and I/O standards not allowed in the same HSIO bank. Examples:

- If you place a true differential I/O standard in pin pair 10 and 11, do not place single-ended I/O standards in pins 8 or 19.
- If you place a single-ended I/O standard in pin 57 or 67, do not place a true differential I/O standard in pin pair 58 and 59.

Combinations Not Allowed (Pin Index Number)		Combinations Not Allowed (Pin Index Number)		Combinations Not Allowed (Pin Index Number)		Combinations Not Allowed (Pin Index Number)	
True Differential Pin Pair	Single-Ended Pin	True Differential Pin Pair	Single-Ended Pin	True Differential Pin Pair	Single-Ended Pin	True Differential Pin Pair	Single-Ended Pin
0 and 1	3, 4	24 and 25	27, 28	48 and 49	51, 52	72 and 73	75, 76
2 and 3	0	26 and 27	16, 24	50 and 51	40, 48	74 and 75	64, 72
4 and 5	1, 15	28 and 29	25, 39	52 and 53	49, 63	76 and 77	73, 87
6 and 7	9	30 and 31	22, 32	54 and 55	46, 56	78 and 79	70, 80
8 and 9	6, 11	32 and 33	31, 34	56 and 57	55, 58	80 and 81	79, 82
10 and 11	8, 19	34 and 35	33, 43	58 and 59	57, 67	82 and 83	81, 90
12 and 13	14, 17	36 and 37	38, 41	60 and 61	62, 65	84 and 85	86, 89
14 and 15	5, 12	38 and 39	29, 36	62 and 63	53, 60	86 and 87	77, 84
16 and 17	13, 26	40 and 41	37, 50	64 and 65	61, 74	88 and 89	85
18 and 19	10, 21	42 and 43	35, 45	66 and 67	59, 69	90 and 91	83, 92
20 and 21	18, 23	44 and 45	42, 47	68 and 69	66, 71	92 and 93	91, 94
22 and 23	20, 30	46 and 47	44, 54	70 and 71	68, 78	94 and 95	93

Table 37. Restricted Pin Placement Combinations for True Differential and Single-Ended I/O Standards in Adjacent HSIO Banks

This table lists the combinations of pins and I/O standards not allowed in adjacent HSIO banks. Examples:

- If you place a true differential I/O standard in pin pair 6 and 7 of bank 3B, do not place single-ended I/O standards in pin 95 of bank 3A.
- If you place a single-ended I/O standard in pin 2 of bank 2B, do not place a true differential I/O standard in pin pair 88 and 89 of bank 2A.

Combinations Not Allowed (Pin Index Number)				Combinations Not Allowed (Pin Index Number)			
True Differential		Single-Ended		True Differential		Single-Ended	
Bank	Pin Pair	Bank	Pin	Bank	Pin Pair	Bank	Pin
3A	88 and 89	3B	2	2A	88 and 89	2B	2
	94 and 95		7		94 and 95		7
3B	2 and 3	3A	88	2B	2 and 3	2A	88
	6 and 7		95		6 and 7		95

Refer to the related information for the figure showing the locations of the HSIO banks.

Related Information

- LVDS SERDES Intel FPGA IP Pin Settings on page 31
- Intel Agilex 5 HSIO Banks, SERDES, and DPA Locations on page 6
- Planning the LVDS Interface on page 24

8.4. SERDES Pin Pairs for Soft-CDR Mode

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

8.5. Placing LVDS Transmitters and Receivers with Identical Data Rates in the Same HSIO Sub-Bank

If you want to place both LVDS transmitter and receiver interfaces in the same HSIO sub-bank, use 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:
  - 23 transmitters, with one `inclock` for the I/O PLL
  - 23 DPA or non-DPA receivers, with one `inclock` for the I/O PLL
  - 8 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. You can either use the `coreclock` from the LVDS transmitter instance or the `coreclock` from the LVDS receiver instance to clock your core logic. For the **RX Soft-CDR** mode, connect the `coreclock` of the LVDS transmitter instance to the `ext_pll_1_outclock2` port of the LVDS receiver instance.
- Set the I/O standard for the `refclk` port of the IOPLL IP to be compatible with the I/O standard used by the LVDS SERDES IP.
- If you are using an external PLL, ensure that the PLL outclock frequency is the same as the LVDS data rate.
- For DPA receiver LVDS interface that uses more than 4 bytes, use two external PLLs.

Related Information

- Connection between IOPLL IP and LVDS SERDES IP in External PLL Mode on page 44
  Provides connection guidelines for the LVDS SERDES IP in the external PLL mode.
- I/O PLLs Driving LVDS Transmitter and Receiver Channels on page 58

Preliminary Documentation – Subject to Change

9. Intel Agilex 5 LVDS SERDES Troubleshooting Guidelines

These debug guidelines are initial debug actions and do not necessarily resolve the failures in your designs.

Table 38. LVDS SERDES I/O Debug Guidelines

This table lists the failure symptoms and the associated debug actions that you can take to identify the failure areas when designing LVDS SERDES systems with Intel Agilex 5 devices.

Failure Symptoms	Recommended Debug Actions
`pll_locked` signal is unable to assert	• Ensure that the `pll_areset` signal is deasserted. • If you are using the LVDS SERDES IP with the external PLL mode, ensure that the desired PLL IP settings match the recommended settings from the LVDS SERDES IP **Clock Resource Summary** tab. *Note:* Use simple PLL settings when debugging this failure to limit the failure scope. Ensure that the external PLL automatic switchover and dynamic reconfiguration modes are disabled. Check `refclk` signal quality if you detect any violation. • Switch to use internal PLL to verify if the same failure occurs when using internal PLL.
`rx_dpa_locked` signal is unable to assert	• Ensure that the `pll_locked` signal is asserted and the `rx_dpa_reset` is deasserted. It is important to ensure that the PLL is able to lock to confirm that the LVDS SERDES IP input clock frequency is correct. • Provide a training pattern to the DPA block with a toggling signal that conforms to the LVDS input buffer specification.
Random bit error occurs at LVDS receiver parallel data out bus	• Ensure that R_D termination is applied. You can enable OCT R_D using the assignment editor in the Intel Quartus Prime software or place an on-board 100 Ω resistor termination. • Measure the `rx_in_p` and `rx_in_n` signal voltages and ensure that the voltages conforms to the V_{ID} and V_{ICM} requirements. • If the `rx_in_p` and `rx_in_n` signals have jitter, ensure that the signals have sufficient data valid window that conforms to the sampling window requirements. • Re-initialize the LVDS receiver reset sequence and ensure that: — The `pll_locked` signal is asserted. — The `rx_dpa_locked` signal is asserted. — The `rx_fifo_reset` signal is deasserted after FIFO reset (for DPA FIFO mode only). — The `rx_divfwdclk` (soft-CDR mode only) and `coreclock` signals have the correct clock frequencies ($\text{Expected Frequency} = \frac{\text{Data Rate}}{\text{SERDES Factor}}$).
LVDS receiver parallel data out is not matching a training pattern	Assert the `rx_bitslip_ctrl` signal for one clock cycle to add bit latency to the received bitstream. Continue to assert the signal until you see the expected pattern at the `rx_out` bus.
The `rx_bitslip_max` signal asserts before it reaches the bit slip rollover value	• Check the bit slip rollover value in the LVDS SERDES IP. Set the rollover value based on $\text{Deserialization Factor} - 1$. • Assert the `rx_bitslip_reset` signal before starting the bit slip and the reset must hold for at least one parallel clock cycle (based on `coreclock` or `rx_divfwdck`).

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Preliminary Documentation – Subject to Change

10. Document Revision History for the Intel Agilex 5 LVDS SERDES User Guide

Document Version	Changes
2023.08.11	• Changed the term "high-speed SERDES" to "LVDS SERDES" • Updated support for serialization factor of 8. • Updated support for SLVS-400 in non-DPA mode. • Changed `tx_coreclock` and `rx_coreclock` to `coreclock`. • Added LVDS SERDES Intel FPGA IP information. • Updated guidelines for pin placement for differential channels. • Added topic about PLLs driving LVDS transmitter and receiver channels. • Updated topic about placing LVDS transmitters and receivers in the same bank. • Added information about timing. • Added information about design examples. • Added troubleshooting guidelines. • Added topics related to planning your LVDS interface pins and selecting bytes and pins in the **Pin Settings** tab. • Added topic about placement restrictions if you mix true differential and single-ended I/O standards in the same or adjacent HSIO banks.
2023.02.10	Initial release.

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Preliminary Documentation – Subject to Change