Explore more resources
Altera® Design Hub

altera
An Intel Company

Serial Lite IV Intel® FPGA IP User Guide

Updated for Quartus® Prime Design Suite: **25.1**

IP Version: **5.5.2**

Contents

Send Feedback

1. About the Serial Lite IV Intel® FPGA IP User Guide

This document describes IP features, architecture description, steps to generate, and guidelines to design the Serial Lite IV Intel® FPGA IP using the E-tile transceivers in Stratix® 10 (TX and MX series) and Agilex™ 7 (F-series) devices.

Intended Audience

This document is intended for the following users:

- Design architects to make IP selection during the system-level design planning phase
- Hardware designers when integrating the IP into their system-level design
- Validation engineers during the system-level simulation and hardware validation phases

Related Documents

The following table lists other reference documents that are related to the Serial Lite IV Intel FPGA IP.

Table 1. Related Documents

Reference	Description
Serial Lite IV Stratix 10 FPGA IP Design Example User Guide	This document provides generation, usage guidelines, and functional description of the Serial Lite IV Intel FPGA IP design examples in Stratix 10 devices.
Serial Lite IV Agilex 7 FPGA IP Design Example User Guide	This document provides generation, usage guidelines, and functional description of the Serial Lite IV Intel FPGA IP design examples in Agilex 7 devices.
E-tile Hard IP User Guide: E-tile Hard IP for Ethernet and E-Tile CPRI PHY Intel FPGA IPs	This document describes the features, functionality, and guidelines of the E-Tile Hard IP for Ethernet and E-Tile CPRI PHY Intel FPGA IPs in Stratix 10 devices.
Stratix 10 Device Data Sheet	This document describes the electrical characteristics, switching characteristics, configuration specifications, and timing for Stratix 10 devices.
Agilex 7 FPGAs and SoCs Device Data Sheet: F-Series and I-Series	This document describes the electrical characteristics, switching characteristics, configuration specifications, and timing for Agilex 7 devices.
E-Tile Transceiver PHY User Guide	This document describes the features, functionality, and guidelines of the E-tile transceiver PHY in Stratix 10 devices.

Acronyms and Glossary

Table 2. Acronym List

Acronym	Expansion
CW	Control Word
RS-FEC	Reed-Solomon Forward Error Correction
PMA	Physical Medium Attachment
TX	Transmitter
RX	Receiver
PAM4	Pulse-Amplitude Modulation 4-Level
NRZ	Non-return-to-zero
PCS	Physical Coding Sublayer
MII	Media Independent Interface
XGMII	10 Gigabit Media Independent Interface

2. Serial Lite IV Intel FPGA IP Overview

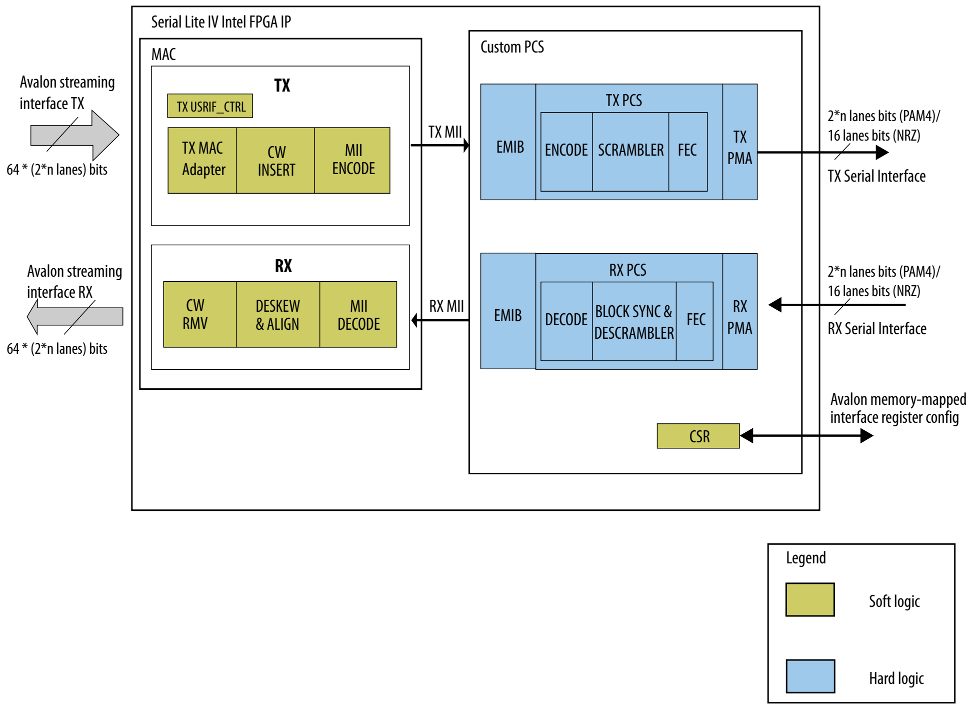
Serial Lite IV Intel FPGA IP is suitable for high bandwidth data communication for chip-to-chip, board-to-board, and backplane applications.

The Serial Lite IV Intel FPGA IP incorporates a media access control (MAC), physical coding sublayer (PCS), and physical media attachment (PMA) block. The IP supports data transfer up to 56 Gbps per lane with a maximum of eight PAM4 lanes in a single link or 28 Gbps per lane with a maximum of 16 NRZ lanes. This protocol offers high bandwidth, low overhead frames, low I/O count, and supports high scalability in both numbers of lanes and speed. The IP is easily reconfigurable with support of a wide range of data rates with Ethernet PCS mode of the E-tile transceiver. It also supports reference clocks provided from separate clock chips or oscillators with a tolerance of ±100 ppm clock variation between the different clock chips or oscillators.

This IP supports two transmission modes:

- Basic mode—This is a pure streaming mode where data is sent without the start-of-packet, empty cycle, and end-of-packet to increase bandwidth. The IP takes the first valid data as the start of a burst.
- Full mode—This is a packet transfer mode. In this mode, the IP sends a burst and a sync cycle at the start and end of a packet as delimiters.

Figure 1. Serial Lite IV High Level Block Diagram



You can generate Serial Lite IV Intel FPGA IP design examples to learn more about the IP features. Refer to *Serial Lite IV Stratix 10 FPGA IP Design Example User Guide* and *Serial Lite IV Agilex 7 FPGA IP Design Example User Guide*.

Related Information

- Functional Description on page 11
- Serial Lite IV Intel Stratix 10 FPGA IP Design Example User Guide
- Serial Lite IV Intel Agilex 7 FPGA IP Design Example User Guide

2.1. Release Information

Intel FPGA IP versions match the Quartus® Prime Design Suite software versions until v19.1. Starting in 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 Quartus Prime software version. A change in:

- X indicates a major revision of the IP. If you update the 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 3. Serial Lite IV Intel FPGA IP Release Information

Item	Description
IP Version	5.5.2
Quartus Prime Version	25.1
Release Date	2025.04.07
Ordering Code	IP-SLITE4

2.2. Supported Features

The following table lists the features available in Serial Lite IV Intel FPGA IP:

Table 4. Serial Lite IV Intel FPGA IP Features

Feature	Description
Data Transfer	• Supports up to 56 Gbps per lane with a maximum of eight PAM4 lanes in a single link. • Supports up to 28 Gbps per lane with a maximum of 16 NRZ lanes. • Supports continuous streaming (Basic) or packet (Full) modes. • Supports low overhead frame packets. • Supports byte granularity transfer for every burst size. • Supports user-initiated or automatic lane alignment. • Supports programmable alignment period.
PCS	• Uses hard IP logic that interfaces seamlessly to Stratix 10 and Agilex 7 E-tile transceivers for soft logic resource reduction. • Supports PAM4 modulation mode for 100GBASE-KP4 specification. RS-FEC is always enabled in this modulation mode. • Supports NRZ with optional RS-FEC modulation mode. • Supports 64b/66b encoding decoding.
Error Detection and Handling	• Supports CRC error checking on TX and RX data paths. • Supports RX link error checking. • Supports RX PCS error detection.
Interfaces	• Supports only full duplex packet transfer with independent links. • Uses point-to-point interconnect to multiple FPGA devices with low transfer latency. • Supports user-defined commands.

2.3. IP Version Support Level

The Quartus Prime software and Intel FPGA device support for the Serial Lite IV Intel FPGA IP is as follows:

Table 5. IP Version and Support Level

Quartus Prime	Device	IP Version	Simulation	Compilation	Hardware Design
25.1	Stratix 10 E-tile transceivers	5.5.2	√	√	√
	Agilex 7 E-tile transceivers	5.5.2	√	√	√

2.4. Device Speed Grade Support

The Serial Lite IV Intel FPGA IP supports the following speed grades for Stratix 10 and Agilex 7 E-tile devices:

- Transceiver speed grade: -1, -2, and -3
- Core speed grade: -1, -2, and -3

2.5. Resource Utilization and Latency

The resources and latency for the Serial Lite IV Intel FPGA IP were obtained from the Quartus Prime Pro Edition software version 21.1.

Table 6. Stratix 10 Serial Lite IV Intel FPGA IP Resource Utilization

The latency measurement is based on the round trip latency from the TX core input to the RX core output.

Variant	Number of Data Lanes	Mode	RS-FEC	ALM	Latency (TX core clock cycle)
28 Gbps NRZ	16	Basic	Disabled	16,171	80
	16	Full	Disabled	16,724	82
	16	Basic	Enabled	15,383	239
	16	Full	Enabled	15,771	240
56 Gbps PAM4	8	Basic	Enabled	11,197	154
	8	Full	Enabled	11,591	152

Table 7. Agilex 7 Serial Lite IV Intel FPGA IP Resource Utilization

The latency measurement is based on the round trip latency from the TX core input to the RX core output.

Variant	Number of Data Lanes	Mode	RS-FEC	ALM	Latency (TX core clock cycle)
28 Gbps NRZ	16	Basic	Disabled	16,480	80
	16	Full	Disabled	16,896	82
	16	Basic	Enabled	15,173	239
	16	Full	Enabled	15,534	240
56 Gbps PAM4	8	Basic	Enabled	11,356	154
	8	Full	Enabled	11,448	152

2.6. Bandwidth Efficiency

Table 8. Bandwidth Efficiency

Variables	Settings					
XCVR Mode	PAM4		NRZ			
Streaming Mode	Full	Basic	Full		Basic	
RS-FEC	Enabled	Enabled	Disabled	Enabled	Disabled	Enabled
Serial interface bit rate in Gbps (RAW_RATE)	56.0	56.0	28.0	28.0	28.0	28.0

continued...

Variables	Settings					
Burst size of a transfer in number of word (BURST_SIZE)[1]	2,048	4,194,304	2,048	2,048	4,194,304	4,194,304
Alignment period in clock cycle (SRL4_ALIGN_PERIOD)	4,096	4,096	4,096	4,096	4,096	4,096
64/66b encode	0.96969697	0.96969697	0.96969697	0.96969697	0.96969697	0.96969697
Overhead of a burst size in number of word (BURST_SIZE_OVHD)	2	0	2[2]	2[2]	0[3]	0[3]
Alignment marker period in clock cycle (ALIGN_MARKER_PERIOD)	81,915	81,915	81,916	81,916	81,916	81,916
Alignment marker width in clock cycle (ALIGN_MARKER_WIDTH)	5	5	0	4	0	4
Bandwidth efficiency[4]	0.96821788	0.96916433	0.96827698	0.96822967	0.96922348	0.96917616
Effective rate (Gbps) [5]	54.2202012	54.27320236	27.11175544	27.11043076	27.13825744	27.13693248
Maximum user clock frequency (MHz) [6]	423.59532225	424.00939437	423.62117875	423.6004806	424.0352725	424.01457

Related Information

Link Rate and Bandwidth Efficiency Calculation on page 36

(1) The BURST_SIZE for Basic mode approaches infinity, hence a large number is used.

(2) In Full mode, the BURST_SIZE_OVHD size is inclusive of the START/END paired Control Words in a data stream.

(3) For Basic mode, BURST_SIZE_OVHD is 0 because there is no START/END during streaming.

(4) Refer to *Link Rate and Bandwidth Efficiency Calculation* for bandwidth efficiency calculation.

(5) Refer to *Link Rate and Bandwidth Efficiency Calculation* for effective rate calculation.

(6) Refer to *Link Rate and Bandwidth Efficiency Calculation* for maximum user clock frequency calculation.

3. Functional Description

Serial Lite IV Intel FPGA IP consists of MAC and custom PCS. The MAC communicates with the custom PCS through MII interfaces.

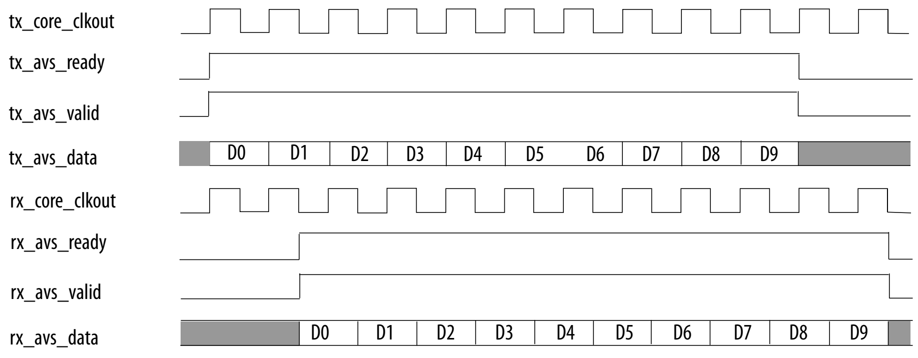
The IP supports two modulation modes:

- PAM4—Provides 2, 4, 6, or 8 number of lanes for selection. A PCS block in PAM4 modulation mode contains four Ethernet channels. The IP always instantiates two PCS channels for each lane in PAM4 modulation mode.
- NRZ—Provides 1 to 16 number of lanes for selection. In this modulation mode, each PCS block supports up to a maximum of four Ethernet channels.

Each modulation mode supports two data modes:

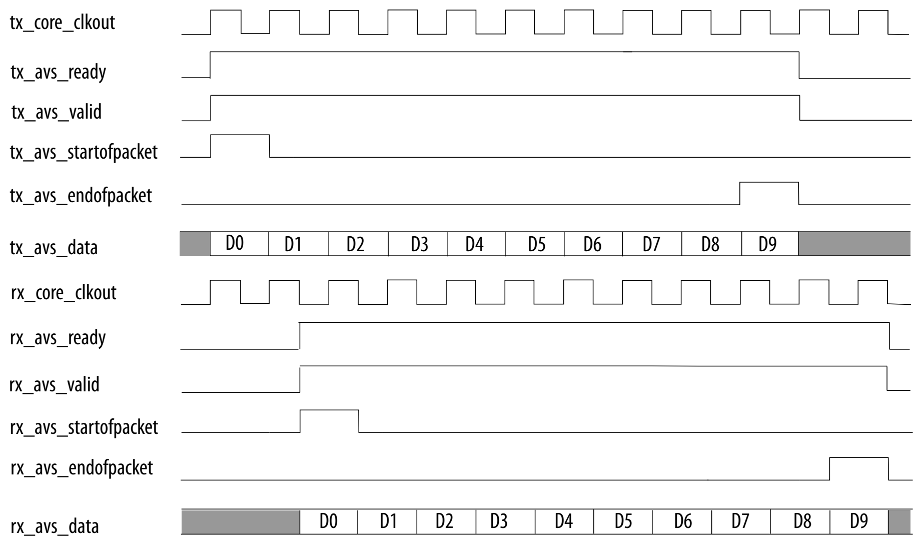
- Basic mode—This is a pure streaming mode where data is sent without the start-of-packet, empty cycle, and end-of-packet to increase bandwidth. The IP takes the first valid data as the start of a burst.

Figure 2. Basic Mode Data Transfer



- Full mode—This is the packet mode data transfer. In this mode, the IP sends a burst and a sync cycle at the start and the end of a packet as delimiters.

Figure 3. Full Mode Data Transfer



Related Information

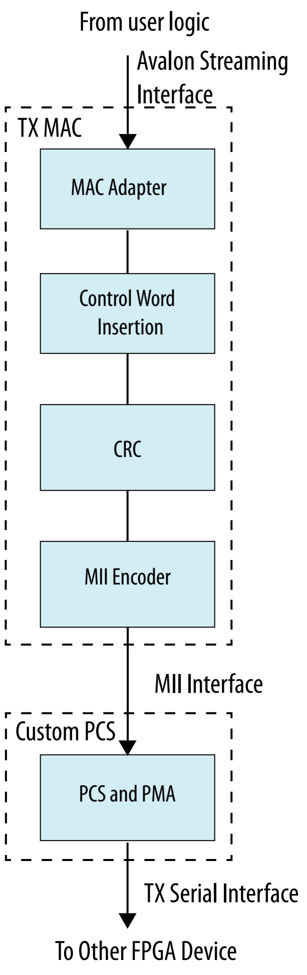
- Serial Lite IV Intel FPGA IP Overview on page 6
- Serial Lite IV Intel Stratix 10 FPGA IP Design Example User Guide
- Serial Lite IV Intel Agilex 7 FPGA IP Design Example User Guide
- E-Tile Transceiver PHY User Guide
- Ethernet Link Inspector User Guide for Intel Stratix 10 Devices
  Information about the Ethernet Link Inspector, an inspection tool that can continuously monitor an Ethernet link.

3.1. TX Datapath

The TX datapath consists of the following components:

- MAC adapter
- Control word insertion block
- CRC
- MII encoder
- PCS block
- PMA block

Figure 4. TX Datapath



3.1.1. TX MAC Adapter

The TX MAC adapter controls the data transmission to the user logic using the Avalon® streaming interface. This block supports user-defined information transmission and flow control.

Transferring User-defined Information

In Full mode, the IP provides the `tx_is_usr_cmd` signal that you can use to initiate user-defined information cycle such as XOFF/XON transmission to the user logic. You can initiate the user-defined information transmission cycle by asserting this signal and transfer the information using `tx_avs_data` along with the assertion of `tx_avs_startofpacket` and `tx_avs_valid` signals. The block then deasserts the `tx_avs_ready` for two cycles.

Note: The user-defined information feature is available only in Full mode.

Flow Control

There are conditions where the TX MAC is not ready to receive data from the user logic such as during link re-alignment process or when there is no data available for transmission from the user logic. To avoid data loss due to these conditions, the IP uses the `tx_avs_ready` signal to control the data flow from the user logic. The IP deasserts the signal when the following conditions occur:

- When `tx_avs_startofpacket` is asserted, `tx_avs_ready` is deasserted for one clock cycle.
- When `tx_avs_endofpacket` is asserted, `tx_avs_ready` is deasserted for one clock cycle.
- When any paired CWs is asserted `tx_avs_ready` is deasserted for two clock cycles.
- When RS-FEC alignment marker insertion occurs at the custom PCS interface, `tx_avs_ready` is deasserted for four clock cycles.
- Every 17 Ethernet core clock cycles in PAM4 modulation mode and every 33 Ethernet core clock cycles in NRZ modulation mode. The `tx_avs_ready` is deasserted for one clock cycle.
- When user logic deasserts `tx_avs_valid` during no data transmission.

The following timing diagrams are examples of TX MAC adapter using `tx_avs_ready` for data flow control.

Figure 5. Flow Control with `tx_avs_valid` Deassertion and START/END Paired CWs

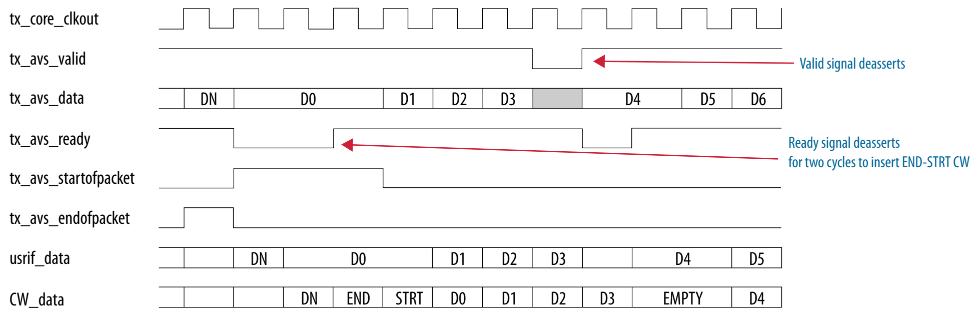


Figure 6. Flow Control with Alignment Marker Insertion

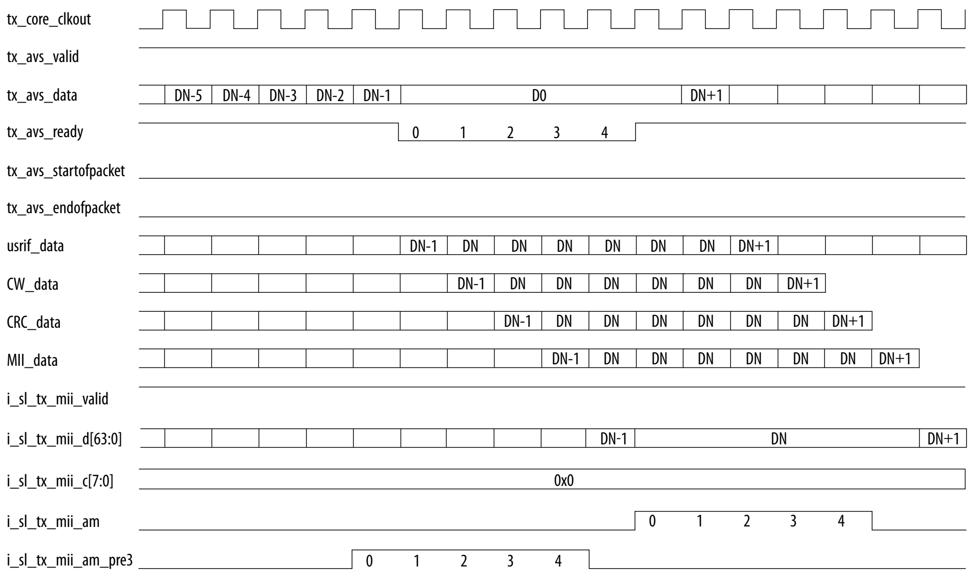
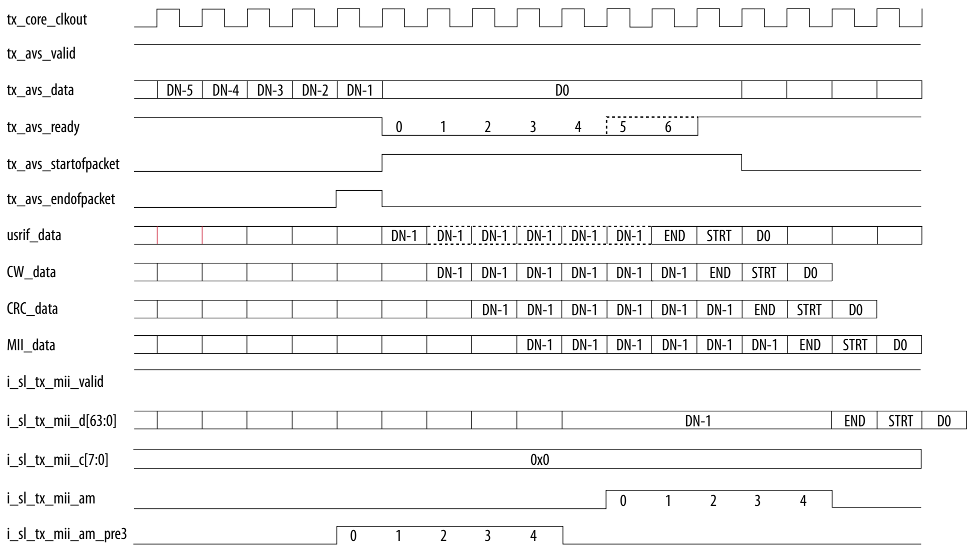


Figure 7. Flow Control with START/END Paired CWs Coincide with Alignment Marker Insertion



3.1.2. Control Word (CW) Insertion

The Serial Lite IV Intel FPGA IP constructs CWs based on the input signals from the user logic. The CWs indicate packet delimiters, transmission status information or user data to the PCS block and they are derived from XGMII control codes.

The following table shows the description of the supported CWs:

Table 9. Description of Supported CWs

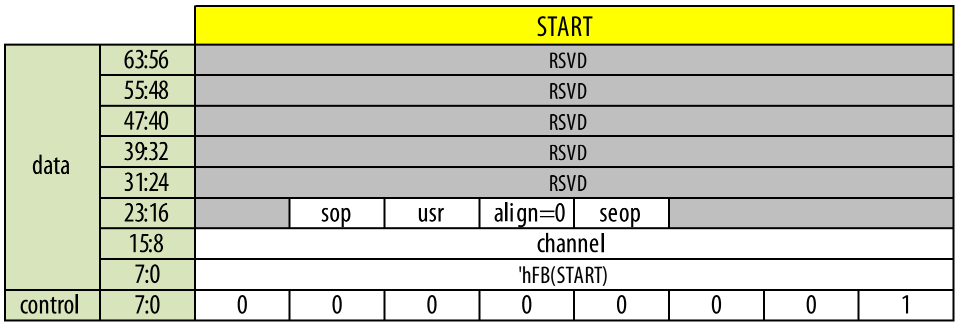
CW	Number of Words (1 word = 64 bits)	In-band	Description
START	1	Yes	Start of data delimiter.
END	1	Yes	End of data delimiter.
ALIGN	2	Yes	Control word (CW) for RX alignment.
EMPTY_CYC	2	Yes	Empty cycle in a data transfer.
IDLE	1	No	IDLE (out of band).
DATA	1	Yes	Payload.

Table 10. CW Field Description

Field	Description
RSVD	Reserved field. May be used for future extension. Tied to 0.
`num_valid_bytes_eob`	Number of valid bytes in the last word (64-bit). This is a 3-bit value. • 3'b000: 8 bytes • 3'b001: 1 byte • 3'b010: 2 bytes • 3'b011: 3 bytes • 3'b100: 4 bytes • 3'b101: 5 bytes • 3'b110: 6 bytes • 3'b111: 7 bytes
EMPTY	Number of non-valid words at the end of a burst.
`eop`	Indicates the RX Avalon streaming interface to assert an end-of-packet signal.
`sop`	Indicates the RX Avalon streaming interface to assert a start-of-packet signal.
`seop`	Indicates the RX Avalon streaming interface to assert a start-of-packet and an end-of-packet in the same cycle.
`align`	Check RX alignment.
CRC32	The values of computed CRC.
`usr`	Indicates that the control word (CW) contains user-defined information.

3.1.2.1. Start-of-burst CW

Figure 8. Start-of-burst CW Format



		START							
data	63:56	RSVD							
	55:48	RSVD							
	47:40	RSVD							
	39:32	RSVD							
	31:24	RSVD							
	23:16		sop	usr	align=0	seop			
	15:8	channel							
	7:0	'hFB(START)							
control	7:0	0	0	0	0	0	0	0	1

In Full mode, you can insert the START CW by asserting the `tx_avs_startofpacket` signal. When you assert only the `tx_avs_startofpacket` signal, the sop bit is set. When you assert both the `tx_avs_startofpacket` and `tx_avs_endofpacket` signals, the seop bit is set.

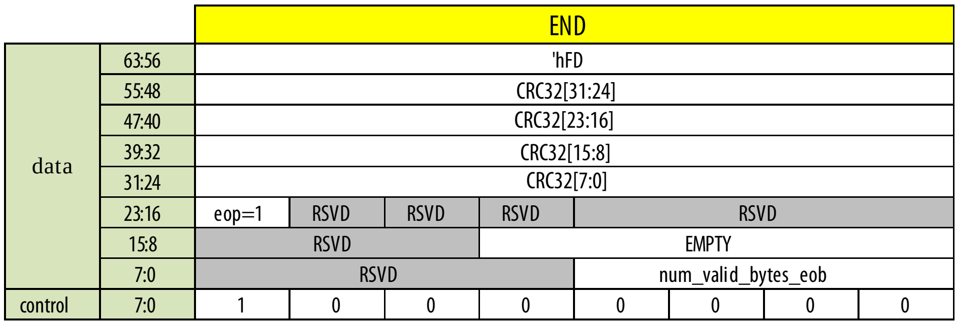
Table 11. START CW Field Values

Field	Value
`sop/seop`	1
`usr` [(7)]	Depending on the `tx_is_usr_cmd` signal: • 1: When `tx_is_usr_cmd = 1` • 0: When `tx_is_usr_cmd = 0`
`align`	0

In Basic mode, the MAC sends a START CW after the reset is deasserted. If no data is available, the MAC continuously sends EMPTY_CYC paired with END and START CWs until you start sending data.

3.1.2.2. End-of-burst CW

Figure 9. End-of-burst CW Format



		END							
data	63:56	'hFD							
	55:48	CRC32[31:24]							
	47:40	CRC32[23:16]							
	39:32	CRC32[15:8]							
	31:24	CRC32[7:0]							
	23:16	eop=1	RSVD	RSVD	RSVD	RSVD			
	15:8	RSVD			EMPTY				
	7:0	RSVD				num_valid_bytes_eob			
control	7:0	1	0	0	0	0	0	0	0

(7) This is supported only in Full mode.

The MAC inserts the END CW when the `tx_avs_endofpacket` is asserted. The END CW contains the number of valid bytes at the last data word and the CRC information.

The CRC value is a 32-bit CRC result for the data between the START CW and the data word before the END CW.

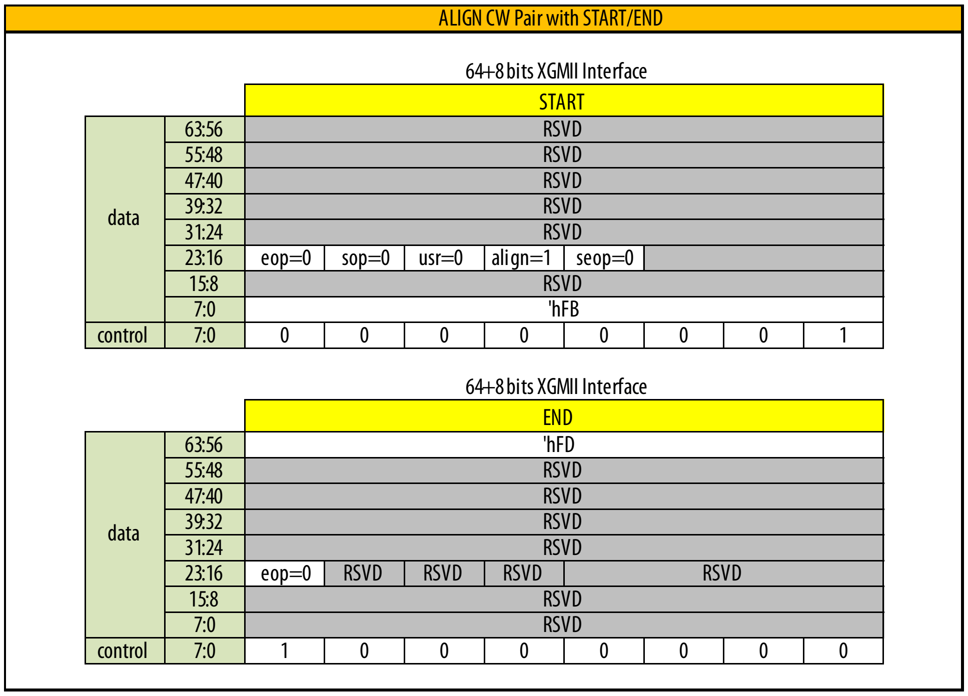
The following table shows the values of the fields in END CW.

Table 12. END CW Field Values

Field	Value
`eop`	1
`CRC32`	CRC32 computed value.
`num_valid_bytes_eob`	Number of valid bytes at the last data word.

3.1.2.3. Alignment Paired CW

Figure 10. Alignment Paired CW Format



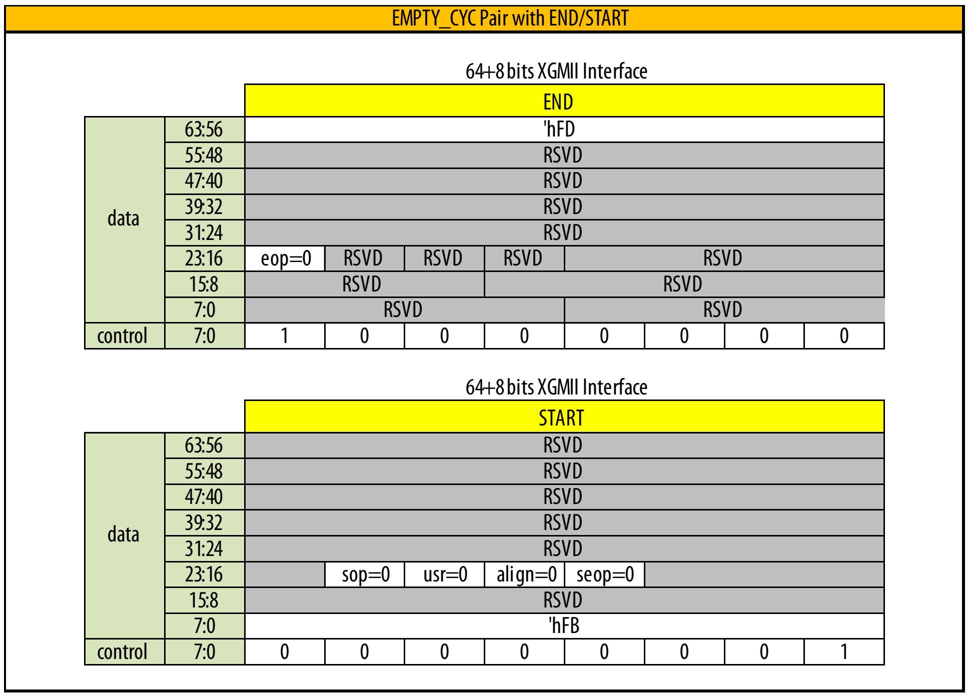
The ALIGN CW is a paired CW with START/END or END/START CWs. You can insert the ALIGN paired CW by either asserting the `tx_link_reinit` signal, set the Alignment Period counter, or initiating a reset. When the ALIGN paired CW is inserted, the align field is set to 1 to initiate the receiver alignment block to check data alignment across all lanes.

Table 13. ALIGN CW Field Values

Field	Value
align	1
eop	0
sop	0
usr	0
seop	0

3.1.2.4. Empty-cycle CW

Figure 11. Empty-cycle CW Format



When you deassert `tx_avs_valid` for two clock cycles during a burst, the MAC inserts an EMPTY_CYC CW paired with END/START CWs. You can use this CW when there is no data available for transmission momentarily.

When you deassert `tx_avs_valid` for one cycle, the IP deasserts `tx_avs_valid` for twice the period of `tx_avs_valid` deassertion to generate a pair of END/START CWs.

Table 14. EMPTY_CYC CW Field Values

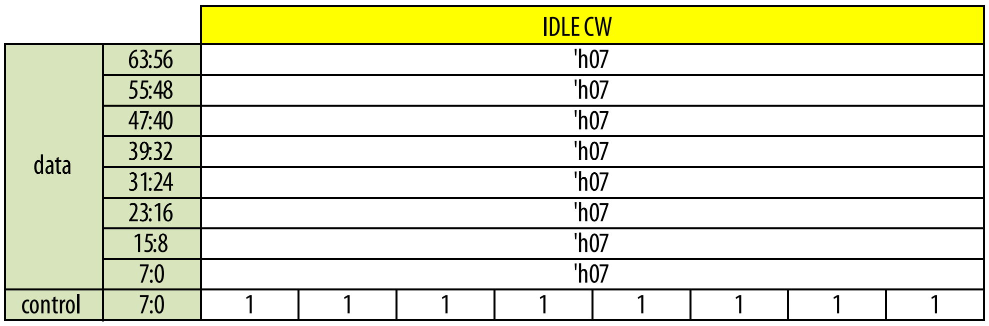
Field	Value
align	0
eop	0

continued...

Field	Value
sop	0
usr	0
seop	0

3.1.2.5. Idle CW

Figure 12. Idle CW Format



		IDLE CW							
data	63:56	'h07							
	55:48	'h07							
	47:40	'h07							
	39:32	'h07							
	31:24	'h07							
	23:16	'h07							
	15:8	'h07							
	7:0	'h07							
control	7:0	1	1	1	1	1	1	1	1

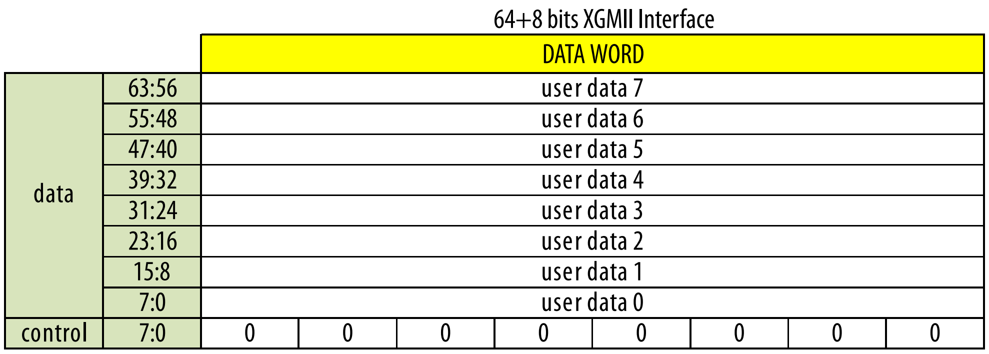
The MAC insert the `IDLE` CW when there is no transmission. During this period, the `tx_avs_valid` signal is low.

You can use the `IDLE` CW when a burst transfer has completed or the transmission is in an idle state.

3.1.2.6. Data Word

The data word is the payload of a packet. The XGMII control bits are all set to 0 in data word format.

Figure 13. Data Word Format



64+8 bits XGMII Interface

		DATA WORD							
data	63:56	user data 7							
	55:48	user data 6							
	47:40	user data 5							
	39:32	user data 4							
	31:24	user data 3							
	23:16	user data 2							
	15:8	user data 1							
	7:0	user data 0							
control	7:0	0	0	0	0	0	0	0	0

3.1.3. TX CRC

You can enable the TX CRC block using the **Enable CRC** parameter in the IP Parameter Editor. This feature is supported in both Basic and Full modes.

The MAC adds the CRC value to the END CW by asserting the `tx_avs_endofpacket` signal. In the BASIC mode, only the `ALIGN` CW paired with END CW contains a valid CRC field.

The TX CRC block interfaces with the TX Control Word Insertion and TX MII Encode block. The TX CRC block computes the CRC value for 64-bit value per-cycle data starting from the START CW up to the END CW.

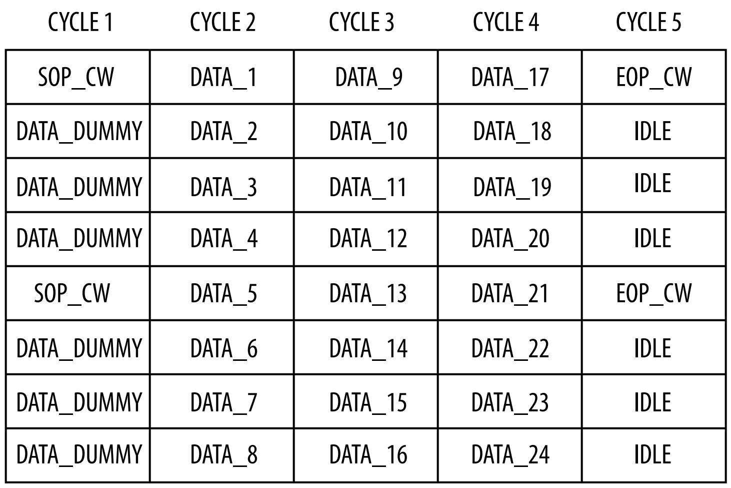
You can assert the `crc_error_inject` signal to intentionally corrupt data in a specific lane to create CRC errors.

3.1.4. TX MII Encoder

The TX MII encoder handles the packet transmission from the MAC to the TX PCS.

In PAM4 mode, a custom PCS always contains four Ethernet channels. Therefore the MII bus data pattern in PAM4 mode is different than the MII bus data pattern in NRZ mode. The following figure shows the data pattern on the 8-bit MII bus in PAM4 modulation mode. The START and END CW appear once in every four MII lanes.

Figure 14. PAM4 Modulation Mode MII Data Pattern



CYCLE 1	CYCLE 2	CYCLE 3	CYCLE 4	CYCLE 5
SOP_CW	DATA_1	DATA_9	DATA_17	EOP_CW
DATA_DUMMY	DATA_2	DATA_10	DATA_18	IDLE
DATA_DUMMY	DATA_3	DATA_11	DATA_19	IDLE
DATA_DUMMY	DATA_4	DATA_12	DATA_20	IDLE
SOP_CW	DATA_5	DATA_13	DATA_21	EOP_CW
DATA_DUMMY	DATA_6	DATA_14	DATA_22	IDLE
DATA_DUMMY	DATA_7	DATA_15	DATA_23	IDLE
DATA_DUMMY	DATA_8	DATA_16	DATA_24	IDLE

The following figure shows the data pattern on the 8-bit MII bus in NRZ modulation mode. The START and END CW appear in every MII lanes.

Figure 15. NRZ Modulation Mode MII Data Pattern

CYCLE 1	CYCLE 2	CYCLE 3	CYCLE 4	CYCLE 5
SOP_CW	DATA_1	DATA_9	DATA_17	EOP_CW
SOP_CW	DATA_2	DATA_10	DATA_18	EOP_CW
SOP_CW	DATA_3	DATA_11	DATA_19	EOP_CW
SOP_CW	DATA_4	DATA_12	DATA_20	EOP_CW
SOP_CW	DATA_5	DATA_13	DATA_21	EOP_CW
SOP_CW	DATA_6	DATA_14	DATA_22	EOP_CW
SOP_CW	DATA_7	DATA_15	DATA_23	EOP_CW
SOP_CW	DATA_8	DATA_16	DATA_24	EOP_CW

3.1.5. TX PCS and PMA

The Serial Lite IV Intel FPGA IP uses the custom PCS variant in the E-tile Hard IP for Ethernet Intel FPGA IP.

For more information about the custom PCS variant from the E-tile Hard IP for Ethernet Intel FPGA IP, refer to *E-tile Hard IP User Guide: E-tile Hard IP for Ethernet and E-Tile CPRI PHY Intel FPGA IPs*.

Related Information

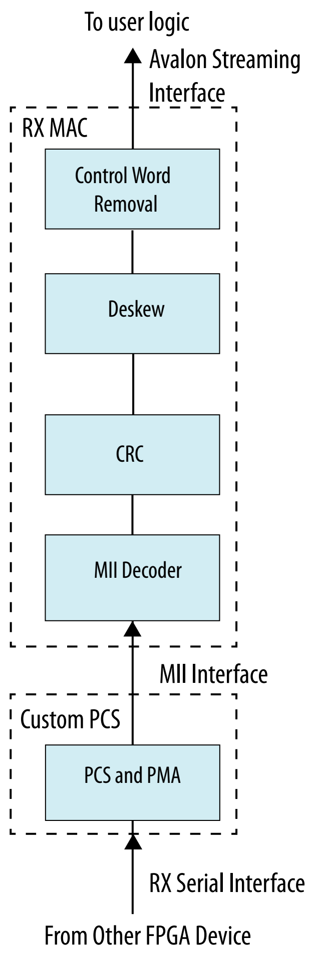
About the E-tile Hard IP for Ethernet Intel FPGA IP Core

3.2. RX Datapath

The RX datapath consists of the following components:

- PMA block
- PCS block
- MII decoder
- CRC
- Deskew block
- Control Word removal block

Figure 16. RX Datapath



3.2.1. RX PCS and PMA

The Serial Lite IV Intel FPGA IP uses the custom PCS variant in the E-tile Hard IP for Ethernet Intel FPGA IP.

For more information about the custom PCS variant from the E-tile Hard IP for Ethernet Intel FPGA IP, refer to the *E-tile Hard IP User Guide: E-tile Hard IP for Ethernet and E-Tile CPRI PHY Intel FPGA IPs*.

Related Information

About the E-tile Hard IP for Ethernet Intel FPGA IP Core

3.2.2. RX MII Decoder

This block identifies if incoming data contains control word and alignment markers.

The RX MII decoder outputs data in the form of 1-bit valid, 1-bit marker indicator, 1-bit control indicator, and 64-bit data per lane.

3.2.3. RX CRC

You can enable the TX CRC block using the **Enable CRC** parameter in the IP Parameter Editor. This feature is supported in both Basic and Full modes.

The RX CRC block interfaces with the RX Control Word Removal and RX MII Decoder blocks. The IP asserts `rx_crc_error` signal when a CRC error occurs.

The IP deasserts the `rx_crc_error` at every new burst. It is an output to the user logic for user logic error handling.

3.2.4. RX Deskew

The RX deskew block detects the alignment markers for each lane and re-aligns the data before sending it to the RX CW removal block.

You can choose to let the IP core to align the data for each lane automatically when an alignment error occurs by setting the **Enable Auto Alignment** parameter in the IP parameter Editor. If you disable the automatic alignment feature, the IP core asserts the `rx_error` signal to indicate alignment error. You must assert the `rx_link_reinit` to initiate the lane alignment process when a lane alignment error occurs.

The RX deskew detects the alignment markers based on a state machine. The following diagram shows the states in the RX deskew block.

Figure 17. RX Deskew Lane Alignment State Machine with Auto Alignment Enabled Flow Chart

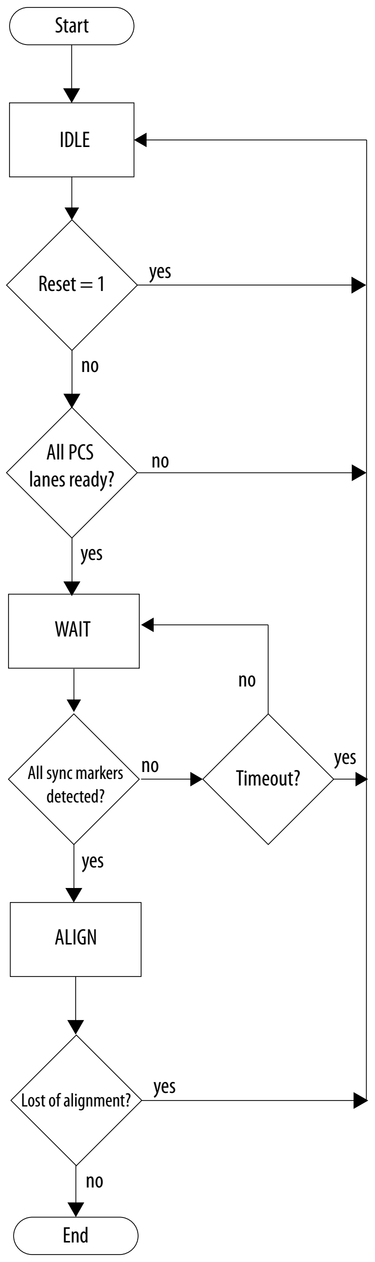
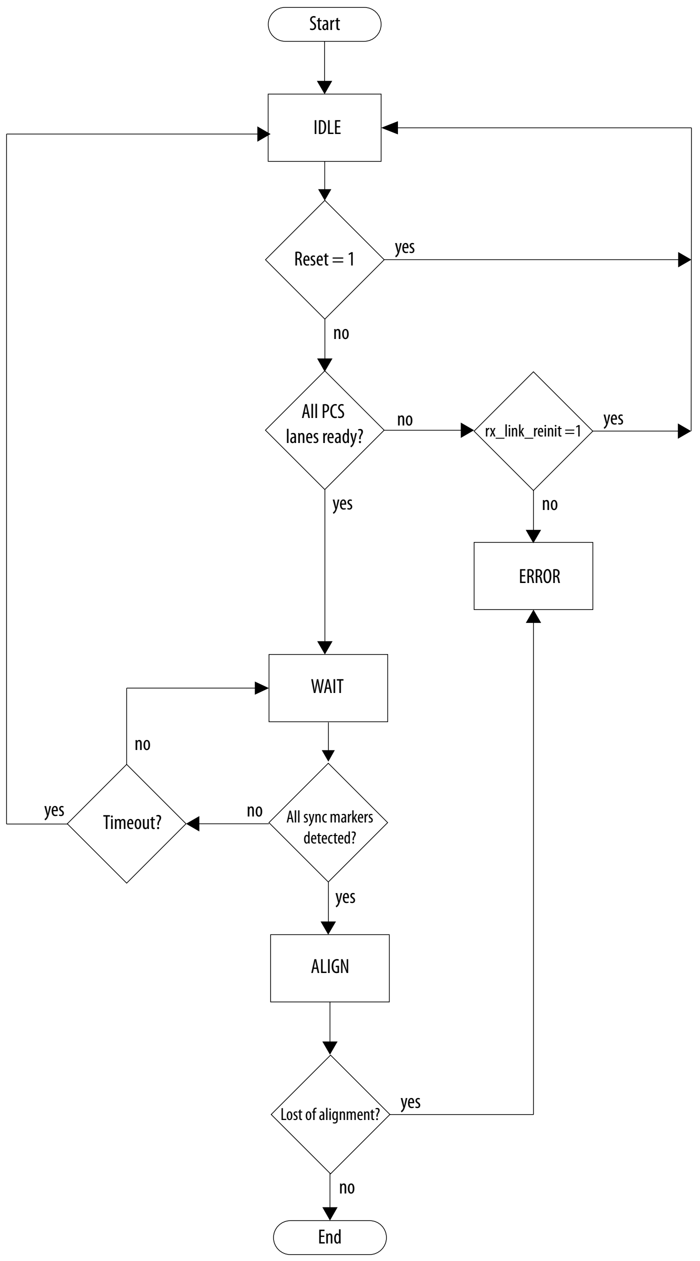


Figure 18. RX Deskew Lane Alignment State Machine with Auto Alignment Disabled Flow Chart



1. The alignment process starts with the IDLE state. The block moves to WAIT state when all PCS lanes are ready and `rx_link_reinit` is deasserted.
2. In WAIT state, the block checks all detected markers are asserted within the same cycle. If this condition is true, the block moves to the ALIGNED state.
3. When the block is in the ALIGNED state, it indicates the lanes are aligned. In this state, the block continues to monitor lane alignment and check if all markers are present within the same cycle. If at least one marker is not present in the same cycle and the **Enable Auto Alignment** parameter is set, the block goes to the

IDLE state to re-initialize the alignment process. If **Enable Auto Alignment** is not set and at least one marker is not present in the same cycle, the block goes to ERROR state and waits for the user logic to assert `rx_link_reinit` signal to initiate lane alignment process.

Figure 19. Lane Realignment with Enable Auto Alignment Enabled

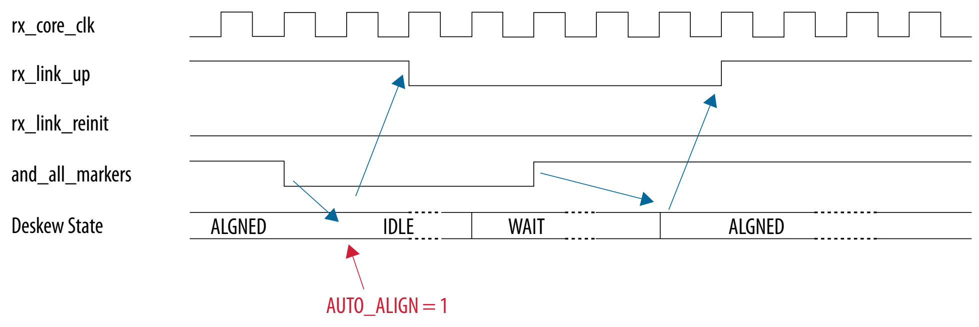
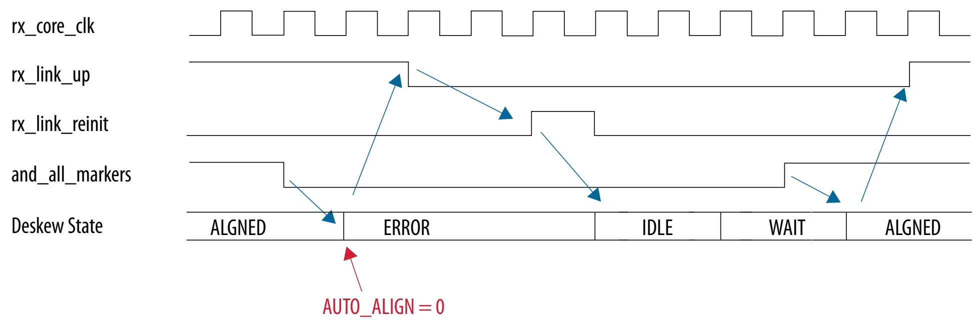


Figure 20. Lane Realignment with Enable Auto Alignment Disabled



3.2.5. RX CW Removal

This block decodes the CWs and sends data to the user logic using the Avalon streaming interface after the removal of the CWs.

When there is no valid data available, the RX CW removal block deasserts the `rx_avs_valid` signal.

In FULL mode, if the user bit is set, this block asserts the `rx_is_usr_cmd` signal and the data in the first clock cycle is used as user-defined information or command.

When `rx_avs_ready` deasserts and `rx_avs_valid` asserts, the RX CW removal block generates an error condition to the user logic.

The Avalon streaming signals related to this block are as follow:

- `rx_avs_startofpacket`
- `rx_avs_endofpacket`
- `rx_avs_channel`
- `rx_avs_empty`
- `rx_avs_data`

- `rx_avs_valid`
- `rx_num_valid_bytes_eob`
- `rx_is_usr_cmd` (only available in Full mode)

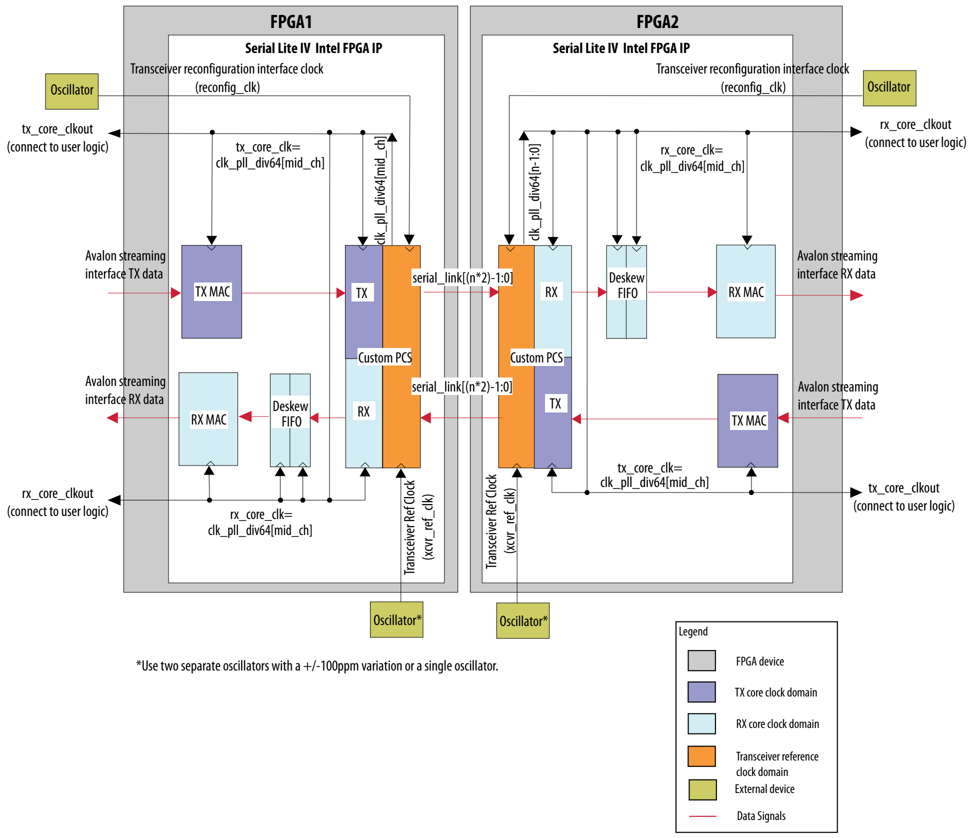
3.3. Serial Lite IV Intel FPGA IP Clock Architecture

The Serial Lite IV Intel FPGA IP has four clock inputs which generate clocks to different blocks:

- Transceiver reference clock (`xcvr_ref_clk`)—Input clock from external clock chips or oscillators which generates clocks for TX MAC, RX MAC, and TX and RX custom PCS blocks. The IP supports reference clocks provided from separate clock chips or oscillators with a tolerance of ±100 ppm clock variation between the different clock chips or oscillators. Refer to *Parameters* for supported frequency range.
- TX core clock (`tx_core_clk`)—This clock is derived from transceiver PLL (`clk_pll_div64[mid_ch]`) in the custom PCS and is used for TX custom PCS interface and TX MAC. This clock is also an output clock from the IP to connect to the TX user logic.
- RX core clock (`rx_core_clk`)—This clock is derived from the transceiver PLL (`clk_pll_div64[mid_ch]`) in the custom PCS and is used for RX custom PCS interface, RX deskew FIFO, and RX MAC. This clock is also an output clock from the IP to connect to the RX user logic.
- Clock for transceiver reconfiguration interface (`reconfig_clk`)—input clock from external clock circuits or oscillators which generates clocks for custom PCS and RS-FEC reconfiguration interface in both TX and RX datapaths. The clock frequency is 100 to 162 MHz. For more about custom PCS and RS-FEC reconfiguration interface, refer to *E-tile Hard IP User Guide: E-tile Hard IP for Ethernet and E-Tile CPRI PHY Intel FPGA IPs.*

The following block diagram shows Serial Lite IV Intel FPGA IP clock domains and the connections within the IP.

Figure 21. Serial Lite IV Intel FPGA IP Clock Architecture


FPGA1
FPGA2
Serial Lite IV Intel FPGA IP
Transceiver reconfiguration interface clock (reconfig_clk)
Oscillator
tx_core_clkout (connect to user logic)
rx_core_clkout (connect to user logic)
tx_core_clk= clk_pll_div64[mid_ch]
rx_core_clk= clk_pll_div64[mid_ch]
clk_pll_div64[mid_ch]
clk_pll_div64[n-1:0]
Avalon streaming interface TX data
Avalon streaming interface RX data
TX MAC
RX MAC
Deskew FIFO
TX
RX
Custom PCS
serial_link[(n*2)-1:0]
Transceiver Ref Clock (xcvr_ref_clk)
Oscillator*
*Use two separate oscillators with a +/-100ppm variation or a single oscillator.
Legend
FPGA device
TX core clock domain
RX core clock domain
Transceiver reference clock domain
External device
Data Signals

Related Information

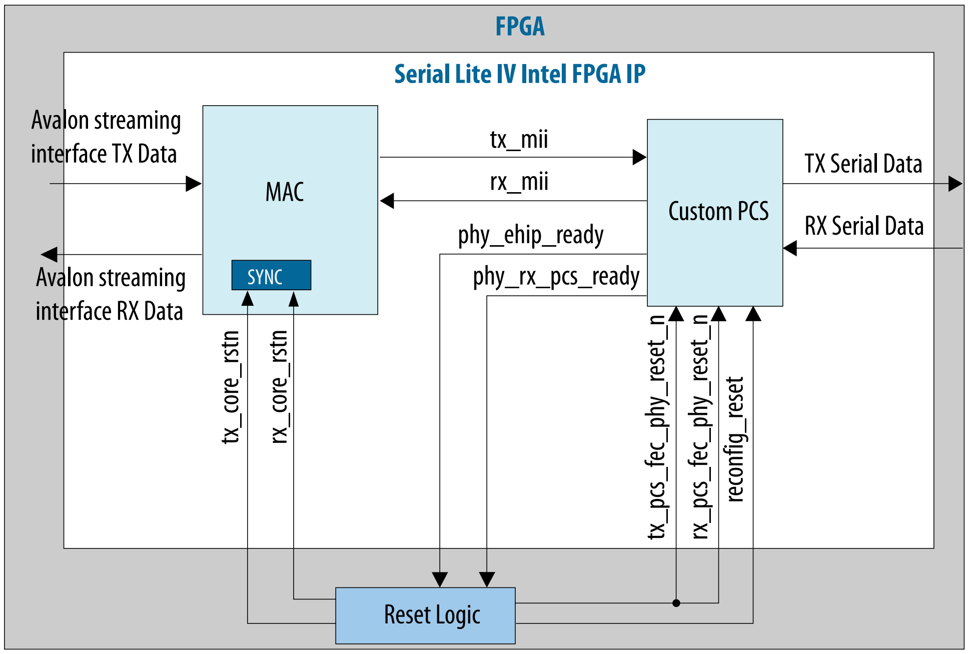
- E-Tile Transceiver PHY User Guide: Ports and Parameters
- E-tile Hard IP User Guide: E-Tile Hard IP for Ethernet and E-Tile CPRI PHY Intel FPGA IPs
  More information about Ethernet reconfiguration interfaces.
- Parameters on page 45

3.4. Reset and Link Initialization

The MAC, custom PCS, and reconfiguration blocks have different reset signals:

- TX and RX MAC blocks use `tx_core_rst_n` and `rx_core_rst_n` reset signals.
- TX and RX RS-FEC blocks use `tx_pcs_fec_phy_reset_n` and `rx_pcs_fec_phy_reset_n` reset signals.
- Reconfiguration block uses the `reconfig_reset` reset signal.

Figure 22. Reset Architecture



Related Information

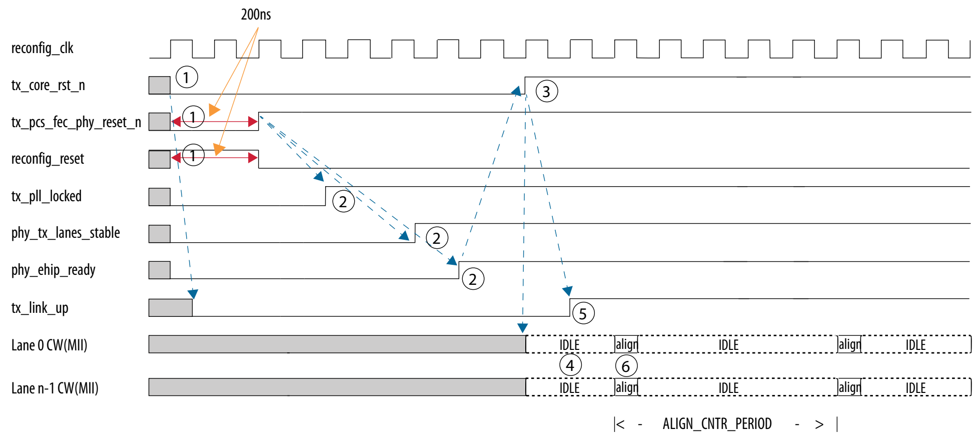
- Reset Guidelines on page 54
- Serial Lite IV Intel Stratix 10 FPGA IP Design Example User Guide: Serial Lite IV IP Toolkit
- Serial Lite IV Intel Agilex 7 FPGA IP Design Example User Guide

3.4.1. TX Reset and Initialization Sequence

The TX reset sequence for Serial Lite IV Intel FPGA IP is as follows:

1. Assert `tx_pcs_fec_phy_reset_n`, `tx_core_rst_n`, and `reconfig_reset` simultaneously to reset the custom PCS, MAC, and reconfiguration blocks. Release the custom PCS (`tx_pcs_fec_phy_reset_n`) and reconfiguration reset (`reconfig_reset`) after 200 ns to ensure the blocks are properly reset.
2. The IP then asserts the `phy_tx_lanes_stable`, `tx_pll_locked`, and `phy_ehip_ready` signals after the custom PCS reset is released, to indicate the TX PHY is ready for transmission.
3. The `tx_core_rst_n` signal deasserts after `phy_ehip_ready` signal goes high.
4. The IP starts transmitting `IDLE` characters on the MII interface once the MAC is out of reset. There is no requirement for TX lane alignment and skewing because all lanes use the same clock.
5. While transmitting `IDLE` characters, the MAC asserts the `tx_link_up` signal.
6. The MAC then starts transmitting `ALIGN` paired with `START`/`END` or `END`/`START` CWs at a fixed interval to initiate the lane alignment process of the connected receiver.

Figure 23. TX Reset and Initialization Timing Diagram

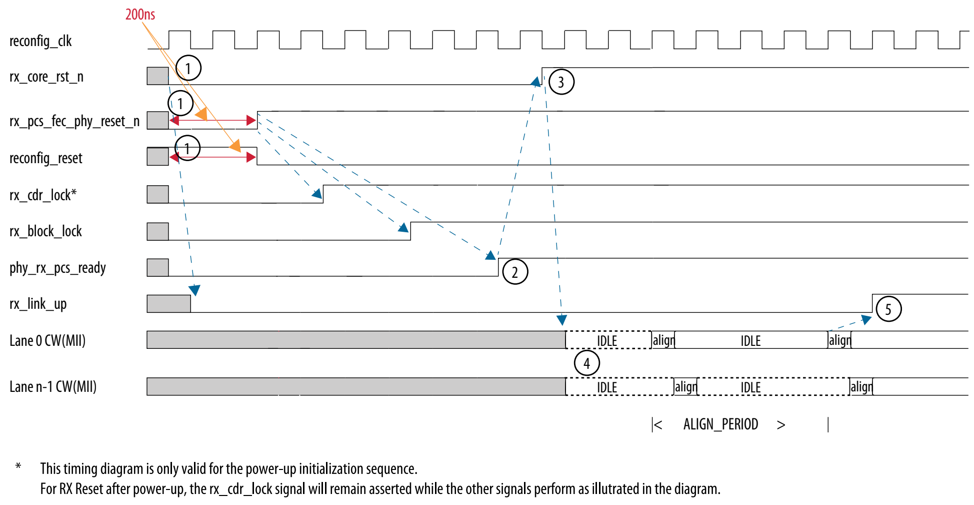


3.4.2. RX Reset and Initialization Sequence

The RX reset sequence for Serial Lite IV Intel FPGA IP is as follows:

1. Assert `rx_pcs_fec_phy_reset_n`, `rx_core_rst_n`, and `reconfig_reset` simultaneously to reset the custom PCS, MAC, and reconfiguration blocks. Release the custom PCS (`rx_pcs_fec_phy_reset_n`) and reconfiguration reset (`reconfig_reset`) after 200 ns to ensure the blocks are properly reset.
2. The IP then asserts the `phy_rx_pcs_ready` signal after the custom PCS reset is released, to indicate RX PHY is ready for transmission.
3. The `rx_core_rst_n` signal deasserts after `phy_rx_pcs_ready` signal goes high.
4. The IP starts the lane alignment process after the RX MAC reset is released and upon receiving `ALIGN` paired with START/END or END/START CWs.
5. The RX deskew block asserts the `rx_link_up` signal once alignment for all lanes has complete.
6. The IP then asserts the `rx_link_up` signal to the user logic to indicate that the RX link is ready to start data reception.

Figure 24. RX Reset and Initialization Timing Diagram


200ns
reconfig_clk
rx_core_rst_n
rx_pcs_fec_phy_reset_n
reconfig_reset
rx_cdr_lock*
rx_block_lock
phy_rx_pcs_ready
rx_link_up
Lane 0 CW(MII)
Lane n-1 CW(MII)
1
2
3
4
5
IDLE
align
|< ALIGN_PERIOD > |
* This timing diagram is only valid for the power-up initialization sequence.
For RX Reset after power-up, the rx_cdr_lock signal will remain asserted while the other signals perform as illutrated in the diagram.

3.4.3. PMA Adaptation Flow

The PMA block in the Serial Lite IV Intel FPGA IP uses the same PMA adaptation flow as the E-Tile Hard IP for Ethernet Intel FPGA IP. Refer to the *Ethernet Adaptation Flow with Non-external AIB Clocking* section in *E-tile Hard IP User Guide: E-Tile Hard IP for Ethernet and E-Tile CPRI PHY Intel FPGA IPs* to trigger the PMA adaptation flow for the Serial Lite IV Intel FPGA IP.

Table 15. Signal Mapping Between Serial Lite IV Intel FPGA IP and E-Tile Hard IP for Ethernet Intel FPGA IP

Signal Name (Serial Lite IV Intel FPGA IP)	Equivalent Signal Name (E-Tile Hard IP for Ethernet Intel FPGA IP)	
	NRZ Mode (10GE/25GE)	PAM4 Mode (100GE)
`tx_pcs_fec_phy_reset_n`	`i_sl_tx_rst_n`	`i_tx_rst_n`
`rx_pcs_fec_phy_reset_n`	`i_sl_rx_rst_n`	`i_rx_rst_n`
`reconfig_reset`	`i_reconfig_reset`	`i_reconfig_reset`

Related Information

E-tile Hard IP User Guide: E-Tile Hard IP for Ethernet and E-Tile CPRI PHY Intel FPGA IPs
More information about PMA adaptation flow.

3.4.3.1. Starting the PMA Adaptation Flow

For E-tile Serial Lite IV Intel FPGA IP core variations, perform the following steps to start the PMA adaptation:

1. Assert the TX, RX Digital Resets (`tx_pcs_fec_phy_reset_n`, `rx_pcs_fec_phy_reset_n`, `tx_core_rst_n`, `rx_core_rst_n`) and reconfiguration reset (`reconfig_reset`) signals. Release the reconfiguration reset (`reconfig_reset`) signal.
2. Trigger the PMA analog reset and re-load the initial PMA settings.

   a. Write 0x200[7:0] = 0x00.
   b. Write 0x201[7:0] = 0x00.
   c. Write 0x202[7:0] = 0x00.
   d. Write 0x203[7:0] = 0x81.
   e. Read 0x207 until it becomes 0x80. This indicates that the operation completed successfully.
   f. Write 0x91[0] = 1 to load the initial settings in the programming file.

3. Apply the control status registers (CSR) Reset (`csr_phy_reset_n`) signal and de-assert the CSR Reset (`csr_phy_reset_n`) signal.
4. De-assert the TX Digital reset (`tx_pcs_fec_phy_reset_n`) and (`tx_core_rst_n`) signals.
5. Configure the Attenuation Value (VOD) (skip this step if using internal serial loopback).
   a. Write 0x84[7:0] = 0x01.
   b. Write 0x85[7:0] = 0x40.
   c. Write 0x86[7:0] = 0x15.
   d. Write 0x87[7:0] = 0x00.
   e. Write 0x90[0] = 1’b1.
   f. Read 0x8A[7]. It should be 1.
   g. Read 0x8B [0] until it changes to 0.
   h. Write 0x8A[7] to 1 to clear the 0x8A[7] flag.
6. Set the operation mode, enable Internal Serial Loopback Mode via PMA attribute code. Attribute = 0x8, data = 0x101.
   a. Write 0x84[7:0] = 0x01.
   b. Write 0x85[7:0] = 0x01.
   c. Write 0x86[7:0] = 0x08.
   d. Write 0x87[7:0] = 0x00.
   e. Write 0x90[0] = 1’b1.
   f. Read 0x8A[7]. It should be 1.
   g. Read 0x8B[0] until it changes to 0.
   h. Write 0x8A[7] to 1'b1 to clear the 0x8A[7] value.
7. Perform Initial Adaptation in Serial Loopback Mode. Attribute = 0xA, data = 0x1.
   a. Write 0x84[7:0] = 0x01.
   b. Write 0x85[7:0] = 0x00.
   c. Write 0x86[7:0] = 0x0A.
   d. Write 0x87[7:0] = 0x00.
   e. Write 0x90[0] = 1'b1.

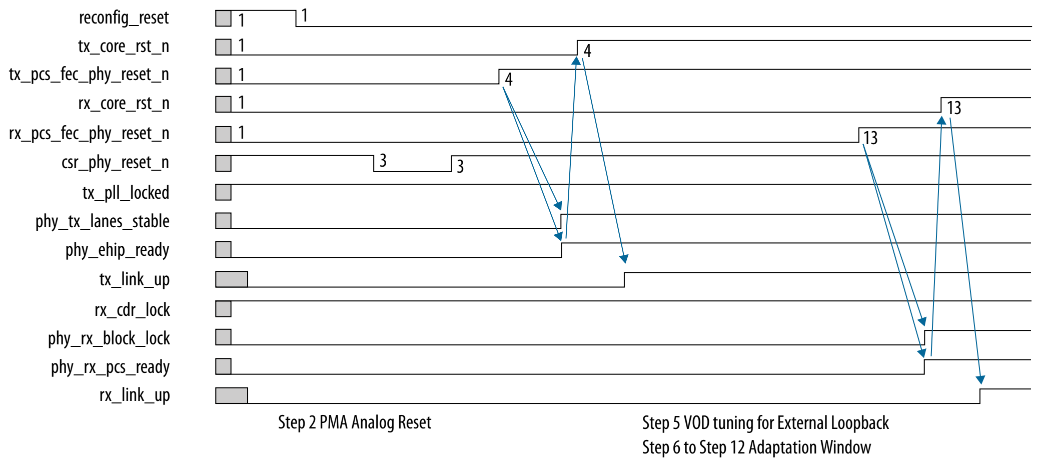
   f. Read 0x8A[7]. It should be 1.
   g. Read 0x8B[0] until it changes to 0.
   h. Write 0x8A[7] to 1'b1 to clear the 0x8A[7] value.

8. Verify that the initial adaptation status is complete using interrupt code. Attribute = 0x0126, data = 0x0B00.
   a. Write 0x84[7:0] = 0x00.
   b. Write 0x85[7:0] = 0x0B.
   c. Write 0x86[7:0] = 0x26.
   d. Write 0x87[7:0] = 0x01.
   e. Write 0x90[0] = 1'b1.
   f. Read 0x8A[7]. It should be 1.
   g. Read 0x8B[0] until it changes to 0.
   h. Write 0x8A[7] to 1'b1 to clear the 0x8A[7] value.
   i. Read register 0x88. Repeat Step 8 until 0x88[0] = 0, to confirm that adaptation is no longer in process.
9. Enable mission mode and disable internal serial loopback (skip this step if using internal serial loopback). Attribute = 0x8, data = 0x100.
   a. Write 0x84[7:0] = 0x00.
   b. Write 0x85[7:0] = 0x01.
   c. Write 0x86[7:0] = 0x08.
   d. Write 0x87[7:0] = 0x00.
   e. Write 0x90[0] = 1'b1.
   f. Read 0x8A[7]. It should be 1.
   g. Read 0x8B[0] until it changes to 0.
   h. Write 0x8A[7] to 1'b1 to clear the 0x8A[7] value.
10. Run initial adaptation via PMA attribute code (skip this step if using internal serial loopback). Attribute = 0xA, data = 0x1.
   a. Write 0x84[7:0] = 0x01.
   b. Write 0x85[7:0] = 0x00.
   c. Write 0x86[7:0] = 0x0A.
   d. Write 0x87[7:0] = 0x00.
   e. Write 0x90[0] = 1'b1.
   f. Read 0x8A[7]. It should be 1.
   g. Read 0x8B[0] until it changes to 0.
   h. Write 0x8A[7] to 1'b1 to clear the 0x8A[7] value.
11. Verify that the initial adaptation status is complete using interrupt code (skip this step if using internal serial loopback). Attribute = 0x0126, data = 0x0B00.
   a. Write 0x84[7:0] = 0x00.
   b. Write 0x85[7:0] = 0x0B.
   c. Write 0x86[7:0] = 0x26.

d. Write 0x87[7:0] = 0x01.
e. Write 0x90[0] = 1'b1.
f. Read 0x8A[7]. It should be 1.
g. Read 0x8B[0] until it changes to 0.
h. Write 0x8A[7] to 1'b1 to clear the 0x8A[7] value.
i. Read register 0x88. Repeat Step 11 until 0x88[0] = 0, to confirm that adaptation is no longer in process.

12. Perform Continuous adaptation via PMA attribute code (skip this step if using internal serial loopback). Attribute = 0xA, data = 0x6.
    a. Write 0x84[7:0] = 0x06.
    b. Write 0x85[7:0] = 0x00.
    c. Write 0x86[7:0] = 0x0A.
    d. Write 0x87[7:0] = 0x00.
    e. Write 0x90[0] = 1'b1.
    f. Read 0x8A[7]. It should be 1.
    g. Read 0x8B[0] until it changes to 0.
    h. Write 0x8A[7] to 1'b1 to clear the 0x8A[7] value.
13. De-assert RX Digital reset (`rx_pcs_fec_phy_reset_n`) and (`rx_core_rst_n`) signals.
14. Verify that the link status signals (`tx_link_up`) and (`rx_link_up`) transitions high. If the link status signals are not high, repeat all the steps above.
15. Send packets.

Figure 25. Reset and Adaptation Flow Sequence



3.5. Link Rate and Bandwidth Efficiency Calculation

The Serial Lite IV Intel FPGA IP bandwidth efficiency calculation is as below:

Bandwidth efficiency = 64/66 * (burst_size - burst_size_ovhd)/burst_size * [align_marker_period / (align_marker_period + align_marker_width)] * [(srl4_align_period - 2) / srl4_align_period]

Table 16. Bandwidth Efficiency Variables Description

Variable	Description
raw_rate	This is the bit rate achieved by the serial interface. raw_rate = SERDES width * transceiver clock frequency Example: raw_rate = 64 * 402.812500 Gbps = 25.78 Gbps
burst_size	Value of burst size. To calculate average bandwidth efficiency, use common burst size value. For maximum rate, use maximum burst size value.
burst_size_ovhd	The burst size overhead value. In Full mode, the burst_size_ovhd value is referring to the START and END paired CWs. In Basic mode, there is no burst_size_ovhd because there is no START and END paired CWs.
align_marker_period	The value of the period where an alignment marker is inserted. The value is 81920 clock cycle for compilation and 1280 for fast simulation. This value is obtained from the PCS hard logic.
align_marker_width	The number of clock cycles where a valid alignment marker signal is held high.
srl4_align_period	The number of clock cycles between two alignment markers. You can set this value using the **Alignment Period** parameter in the IP Parameter Editor.

The link rate calculations are as below:

Effective rate = bandwidth efficiency * raw_rate

You can get the maximum user clock frequency with the following equation. The maximum user clock frequency calculation assumes continuous data streaming and no IDLE cycle occurs at the user logic. This rate is important when designing the user logic FIFO to avoid FIFO overflow.

Maximum user clock frequency = effective rate / 64

4. Getting Started

4.1. Installing and Licensing Intel FPGA IP Cores

The Quartus Prime software installation includes the Intel FPGA IP library. This library provides many useful IP cores for your production use without the need for an additional license. Some Intel FPGA IP cores require purchase of a separate license for production use. The Intel FPGA IP Evaluation Mode allows you to evaluate these licensed Intel FPGA IP cores in simulation and hardware, before deciding to purchase a full production IP core license. You only need to purchase a full production license for licensed Intel IP cores after you complete hardware testing and are ready to use the IP in production.

The Quartus Prime software installs IP cores in the following locations by default:

Figure 26. IP Core Installation Path

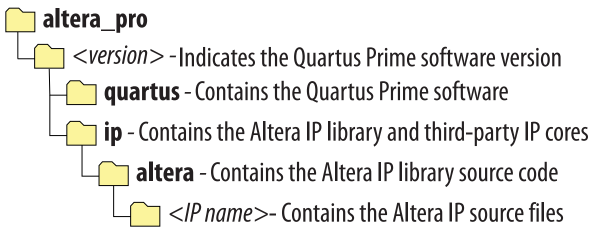

altera_pro
<version> - Indicates the Quartus Prime software version
quartus - Contains the Quartus Prime software
ip - Contains the Altera IP library and third-party IP cores
altera - Contains the Altera IP library source code
<IP name>- Contains the Altera IP source files

Table 17. IP Core Installation Locations

Location	Software	Platform
`<drive>:\intelFPGA_pro\quartus\ip\altera`	Quartus Prime Pro Edition	Windows*
`<home directory>:/intelFPGA_pro/quartus/ip/altera`	Quartus Prime Pro Edition	Linux*

Note: The Quartus Prime software does not support spaces in the installation path.

4.1.1. Intel FPGA IP Evaluation Mode

The free Intel FPGA IP Evaluation Mode allows you to evaluate licensed Intel FPGA IP in simulation and hardware before purchase. Intel FPGA IP Evaluation Mode supports the following evaluations without additional license:

- Simulate the behavior of a licensed Intel FPGA IP in your system.
- Verify the functionality, size, and speed of the IP quickly and easily.
- Generate time-limited device programming files for designs that include IPs.
- Program a device with your IP and verify your design in hardware.

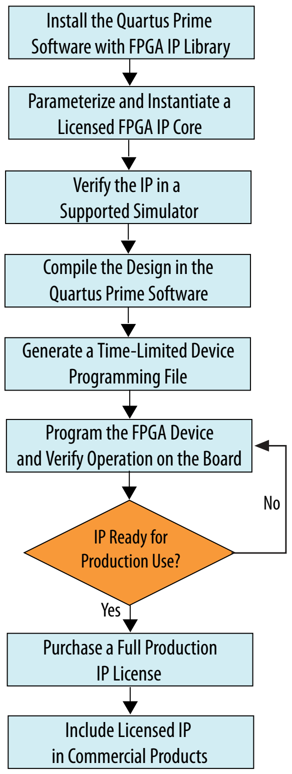
Intel FPGA IP Evaluation Mode supports the following operation modes:

- **Tethered**—Allows running the design containing the licensed Intel FPGA IP indefinitely with a connection between your board and the host computer. Tethered mode requires a serial JTAG cable connected between the JTAG port on your board and the host computer, which is running the Quartus Prime Programmer for the duration of the hardware evaluation period. The Programmer only requires a minimum installation of the Quartus Prime software, and requires no Quartus Prime license. The host computer controls the evaluation time by sending a periodic signal to the device via the JTAG port. If all licensed IPs in the design support tethered mode, the evaluation time runs until any IP evaluation expires. If all of the IPs support unlimited evaluation time, the device does not time-out.
- **Untethered**—Allows running the design containing the licensed IP for a limited time. The IP reverts to untethered mode if the device disconnects from the host computer running the Quartus Prime software. The IP also reverts to untethered mode if any other licensed IP in the design does not support tethered mode.

When the evaluation time expires for any licensed Intel FPGA IP in the design, the design stops functioning. All IPs that use the Intel FPGA IP Evaluation Mode time out simultaneously when any IP in the design times out. When the evaluation time expires, you must reprogram the FPGA device before continuing hardware verification. To extend use of the IP for production, purchase a full production license for the IP.

You must purchase the license and generate a full production license key before you can generate an unrestricted device programming file. During Intel FPGA IP Evaluation Mode, the Compiler only generates a time-limited device programming file (<*project name*>`_time_limited.sof`) that expires at the time limit.

Figure 27. Intel FPGA IP Evaluation Mode Flow


Install the Quartus Prime Software with FPGA IP Library
Parameterize and Instantiate a Licensed FPGA IP Core
Verify the IP in a Supported Simulator
Compile the Design in the Quartus Prime Software
Generate a Time-Limited Device Programming File
Program the FPGA Device and Verify Operation on the Board
IP Ready for Production Use?
No
Yes
Purchase a Full Production IP License
Include Licensed IP in Commercial Products

Note: Refer to each IP's user guide for parameterization steps and implementation details.

Altera licenses IPs on a per-seat, perpetual basis. The license fee includes first-year maintenance and support. You must renew the maintenance contract to receive updates, bug fixes, and technical support beyond the first year. You must purchase a full production license for Intel FPGA IPs that require a production license, before generating programming files that you may use for an unlimited time. During Intel FPGA IP Evaluation Mode, the Compiler only generates a time-limited device programming file (<*project name*>`_time_limited.sof`) that expires at the time limit. To obtain your production license keys, visit the Self-Service Licensing Center.

The Altera Software License Agreements govern the installation and use of licensed IPs, the Quartus Prime design software, and all unlicensed IPs.

Related Information

- FPGA Licensing Support Center
- Introduction to Altera Software Installation and Licensing

4.2. Specifying the IP Parameters and Options

The IP parameter editor allows you to quickly configure your custom IP variation. Use the following steps to specify IP options and parameters in the Quartus Prime Pro Edition software.

1. If you do not already have an Quartus Prime Pro Edition project in which to integrate your Serial Lite IV Intel FPGA IP, you must create one.
    a. In the Quartus Prime Pro Edition, click **File ➤ New Project Wizard** to create a new Quartus Prime project, or **File ➤ Open Project** to open an existing Quartus Prime project. The wizard prompts you to specify a device.
    b. Specify the device family **Stratix 10** or **Agilex 7** and select a production E-tile device that meets the speed grade requirements for the IP.
    c. Click **Finish**.
2. In the IP Catalog, locate and select **Serial Lite IV**. The **New IP Variation** window appears.
3. Specify a top-level name for your new custom IP variation. The parameter editor saves the IP variation settings in a file named <*your_ip*>`.ip`.
4. Click **OK**. The parameter editor appears.
5. Specify the parameters for your IP variation. Refer to Parameters on page 45 for information about Serial Lite IV Intel FPGA IP parameters.
6. Optionally, to generate a simulation testbench or compilation and hardware design example, follow the instructions in the *Design Example User Guide*.
7. Click **Generate HDL**. The **Generation** dialog box appears.
8. Specify output file generation options, and then click **Generate**. The IP variation files generate according to your specifications.
9. Click **Finish**. The parameter editor adds the top-level `.ip` file to the current project automatically. If you are prompted to manually add the `.ip` file to the project, click **Project ➤ Add/Remove Files in Project** to add the file.
10. After generating and instantiating your IP variation, make appropriate pin assignments to connect ports and set any appropriate per-instance RTL parameters.

4.3. Generated File Structure

The Quartus Prime Pro Edition software generates the following IP core output file structure.

For information about the file structure of the design example, refer to the *Serial Lite IV Stratix 10 FPGA IP Design Example User Guide* and *Serial Lite IV Agilex 7 FPGA IP Design Example User Guide*.

Figure 28. Serial Lite IV IP Core Generated Files

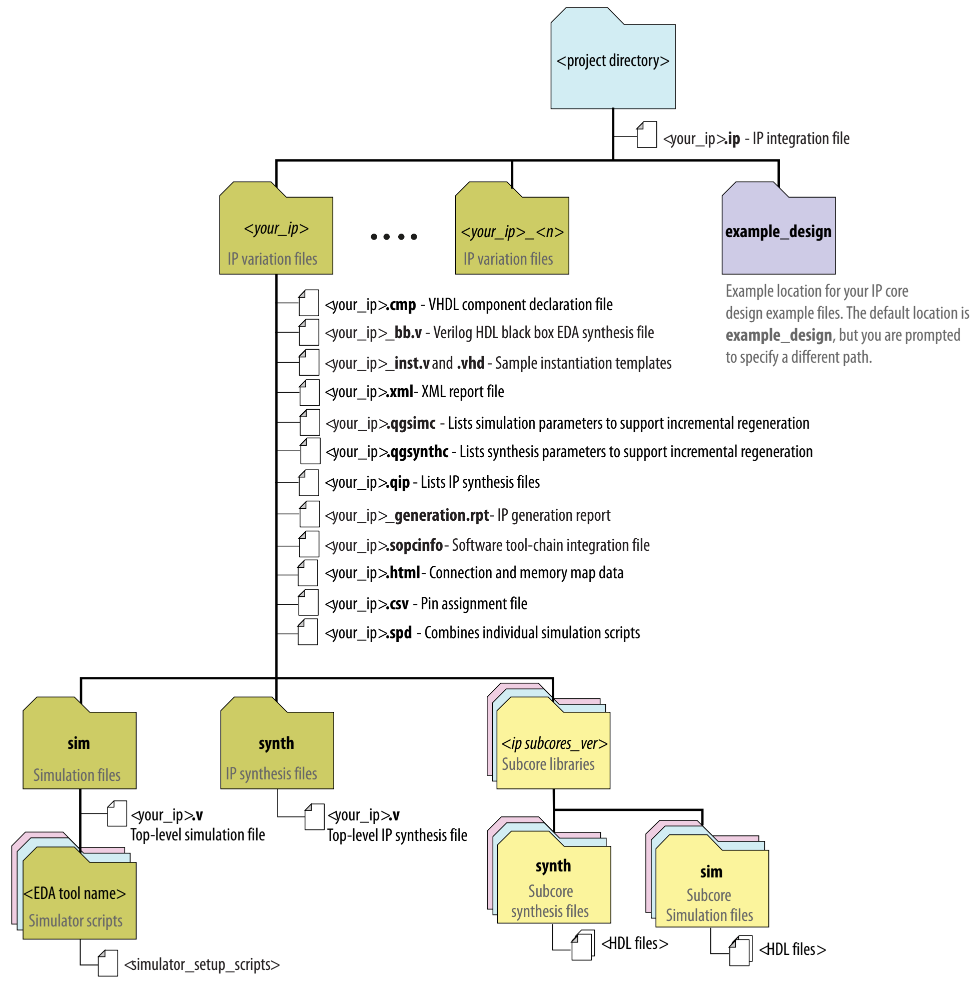

<project directory>
<your_ip>.ip - IP integration file
<your_ip>
IP variation files
<your_ip>_<n>
IP variation files
example_design
Example location for your IP core design example files. The default location is example_design, but you are prompted to specify a different path.
<your_ip>.cmp - VHDL component declaration file
<your_ip>_bb.v - Verilog HDL black box EDA synthesis file
<your_ip>_inst.v and .vhd - Sample instantiation templates
<your_ip>.xml- XML report file
<your_ip>.qgsimc - Lists simulation parameters to support incremental regeneration
<your_ip>.qgsynthc - Lists synthesis parameters to support incremental regeneration
<your_ip>.qip - Lists IP synthesis files
<your_ip>_generation.rpt- IP generation report
<your_ip>.sopcinfo- Software tool-chain integration file
<your_ip>.html- Connection and memory map data
<your_ip>.csv - Pin assignment file
<your_ip>.spd - Combines individual simulation scripts
sim
Simulation files
synth
IP synthesis files
<ip subcores_ver>
Subcore libraries
<your_ip>.v
Top-level simulation file
<your_ip>.v
Top-level IP synthesis file
<EDA tool name>
Simulator scripts
<simulator_setup_scripts>
synth
Subcore synthesis files
sim
Subcore Simulation files
<HDL files>
<HDL files>

Table 18. Serial Lite IV IP Core Generated Files

File Name	Description
`<your_ip>.ip`	The Platform Designer system or top-level IP variation file. *<your_ip>* is the name that you give your IP variation.
`<your_ip>.cmp`	The VHDL Component Declaration (`.cmp`) file is a text file that contains local generic and port definitions that you can use in VHDL design files.
`<your_ip>.html`	A report that contains connection information, a memory map showing the address of each slave with respect to each master to which it is connected, and parameter assignments.
`<your_ip>_generation.rpt`	IP or Platform Designer generation log file. A summary of the messages during IP generation.
`<your_ip>.qgsimc`	Lists simulation parameters to support incremental regeneration.
`<your_ip>.qgsynthc`	Lists synthesis parameters to support incremental regeneration.
`<your_ip>.qip`	Contains all the required information about the IP component to integrate and compile the IP component in the Quartus Prime software.

continued...

File Name	Description
<*your_ip*>.sopcinfo	Describes the connections and IP component parameterizations in your Platform Designer system. You can parse its contents to get requirements when you develop software drivers for IP components. Downstream tools such as the Nios® II tool chain use this file. The `.sopcinfo` file and the `system.h` file generated for the Nios II tool chain include address map information for each slave relative to each master that accesses the slave. Different masters may have a different address map to access a particular slave component.
<*your_ip*>.csv	Contains information about the upgrade status of the IP component.
<*your_ip*>.spd	Required input file for `ip-make-simscript` to generate simulation scripts for supported simulators. The `.spd` file contains a list of files generated for simulation, along with information about memories that you can initialize.
<*your_ip*>_bb.v	You can use the Verilog black-box (`_bb.v`) file as an empty module declaration for use as a black box.
<*your_ip*>_inst.v or _inst.vhd	HDL example instantiation template. You can copy and paste the contents of this file into your HDL file to instantiate the IP variation.
<*your_ip*>.regmap	If IP contains register information, `.regmap` file generates. 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 enables register display views and user customizable statistics in the System Console.
<*your_ip*>.svd	Allows hard processor system (HPS) System Debug tools to view the register maps of peripherals connected to HPS in a Platform Designer system. During synthesis, the `.svd` files for slave interfaces visible to System Console masters are stored in the `.sof` file in the debug section. System Console reads this section, which Platform Designer can query for register map information. For system slaves, Platform Designer can access the registers by name.
<*your_ip*>.v or <*your_ip*>.vhd	HDL files that instantiate each submodule or child IP core for synthesis or simulation.
mentor/	Contains a ModelSim* or QuestaSim* script `msim_setup.tcl` to set up 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 a shell script `xcelium_setup.sh` and other setup files to set up and run Xcelium* simulation
submodules/	Contains HDL files for the IP core submodules.
<*child IP cores*>/	For each generated child IP core directory, Platform Designer generates `synth/` and `sim/` sub-directories.

4.4. Simulating Intel FPGA IP Cores

The Quartus Prime software supports IP core RTL simulation in specific EDA simulators. IP generation optionally creates simulation files, including the functional simulation model, any testbench (or example design), and vendor-specific simulator setup scripts for each IP core. You can use the functional simulation model and any testbench or example design for simulation. IP generation output may also include scripts to compile and run any testbench. The scripts list all models or libraries you require to simulate your IP core.

The Quartus Prime software provides integration with many simulators and supports multiple simulation flows, including your own scripted and custom simulation flows. Whichever flow you choose, IP core simulation involves the following steps:

1. Generate IP HDL, testbench (or example design), and simulator setup script files.
2. Set up your simulator environment and any simulation scripts.
3. Compile simulation model libraries.
4. Run your simulator.

4.4.1. Simulating and Verifying the Design

By default, the parameter editor generates simulator-specific scripts containing commands to compile, elaborate, and simulate Intel FPGA IP models and simulation model library files. You can copy the commands into your simulation testbench script, or edit these files to add commands for compiling, elaborating, and simulating your design and testbench.

Table 19. Intel FPGA IP Core Simulation Scripts

Simulator	File Directory	Script
ModelSim	*<variation name>***/sim/mentor**	**msim_setup.tcl** [8]
QuestaSim		
VCS	*<variation name>***/sim/synopsys/vcs**	**vcs_setup.sh**
VCS MX	*<variation name>***/sim/synopsys/vcsmx**	**vcsmx_setup.sh** **synopsys_sim.setup**
Riviera-PRO*	*<variation name>***/sim/aldec**	**rivierapro_setup.tcl**
Xcelium	*<variation name>***/sim/xcelium**	**xcelium_setup.sh**

4.5. Synthesizing IP Cores in Other EDA Tools

Optionally, use another supported EDA tool to synthesize a design that includes Intel FPGA IP cores. When you generate the IP core synthesis files for use with third-party EDA synthesis tools, you can create an area and timing estimation netlist. To enable generation, turn on **Create timing and resource estimates for third-party EDA synthesis tools** when customizing your IP variation.

The area and timing estimation netlist describes the IP core connectivity and architecture, but does not include details about the true functionality. This information enables certain third-party synthesis tools to better report area and timing estimates. In addition, synthesis tools can use the timing information to achieve timing-driven optimizations and improve the quality of results.

The Quartus Prime software generates the `<`*`variant name`*`>_syn.v` netlist file in Verilog HDL format, regardless of the output file format you specify. If you use this netlist for synthesis, you must include the IP core wrapper file *<variant name>*`.v` or *<variant name>* `.vhd` in your Quartus Prime project.

(8) If you did not set up the EDA tool option— which enables you to start third-party EDA simulators from the Quartus Prime software—run this script in the ModelSim or QuestaSim simulator Tcl console (not in the Quartus Prime software Tcl console) to avoid any errors.

4.6. Compiling the Full Design

You can use the **Start Compilation** command on the Processing menu in the Quartus Prime Pro Edition software to compile your design.

5. Parameters

Table 20. IP

Parameter	Value	Default	Description
General Design Options			
XCVR Mode	• **PAM4** • **NRZ**	PAM4	Select the PCS modulation mode.
Number of lanes	• For PAM4 mode: — **2** — **4** — **6** — **8** • For NRZ mode: — **1 to 16**	2	Select the number of lanes.
Transceiver reference clock frequency	• For PAM4 mode: — **156.25 MHz** — **312.5 MHz** • For NRZ mode: — **138.888888 MHz to 500 MHz**	156.25 MHz	Specifies the transceiver's reference clock frequency.
Preserve unused transceiver channels for PAM4	Enable Disable	Disable	Turn on to preserve unused transceiver channels for PAM4 mode.
Reference clock frequency for preserved channels	125 MHz to 500 MHz	156 MHz	Set the clock frequency of reference clock for the unused preserved channels. If used, this value must be the same as the reference clock frequency that you set for other unused preserved channels in the same tile. This option is only applicable when you turn on **Preserve unused transceiver channels for PAM4**.
User Interface			
Enable Auto Alignment	Enable Disable	Disable	Turn on to enable automatic lane alignment feature.
Enable RSFEC	Enable Disable	Enable	Turn on to enable the RS-FEC feature. For PAM4 PCS modulation mode, RS-FEC is always enabled.
Enable CRC	Enable Disable	Disable	Turn on to enable CRC error detection and correction.
Alignment Period	128-65536	128	Specifies the alignment marker period. The value must be x^2.
Streaming Mode	• **Full** • **Basic**	Full	Select the data streaming for the IP. Full: This mode sends a start-of-packet and end-of-packet cycle within a frame.

continued...

Parameter	Value	Default	Description
			Basic: This is a pure streaming mode where data is sent without a start-of-packet, empty, and end-of-packet to increase bandwidth.
Transceiver data rate	• For PAM4 mode: — **32.5 Gbps** — **40.0 Gbps** — **53.125 Gbps** — **56.0 Gbps** • For NRZ with RS-FEC disabled mode: — **9.92 Gbps** to **28.0 Gbps** • For NRZ with RS-FEC enabled mode: — **10.000 Gbps** to **28.0 Gbps**	53.125 Gbps (PAM4) 25.0 Gbps (NRZ)	Specifies the effective data rate at the output of the transceiver incorporating transmission and other overheads. The value is calculated by the IP by rounding up to 1 decimal place in Gbps unit.

IP Debug and Phy Dynamic Reconfiguration

Table 21. Native Transceiver Phy

Parameter	Value	Default	Description
Dynamic Reconfiguration			
Enable dynamic reconfiguration	—	Enable	Turn on to enable dynamic reconfiguration interface of Transceiver Native PHY.
Enable Native PHY Debug Master Endpoint	Disable Enable	Disable	Turn on to enable the Native PHY Debug Master Endpoint and Optional Reconfiguration Logic Parameters of Transceiver Native PHY.
Optional Reconfiguration Logic			
Enable capability registers	Disable Enable	Disable	Turn on to enable capability register of Transceiver Native PHY, which provide high level information about the transceiver PLL configuration.
Set user-defined IP identifier	—	0	Sets a user-defined numeric identifier that can be read from the user-identifier offset when the capability registers are enabled. You must enable the **Enable capability registers** parameter to change the value for this parameter.
Enable control and status registers	Disable Enable	Disable	Turn on to enable control and status registers of Transceiver Native PHY.

For parameters in the **PMA Adaptation** tab, refer to the PMA Adaptation topic in the *E-Tile Transceiver PHY User Guide*.

Related Information

- E-Tile Transceiver PHY User Guide: PMA Parameters
  Information about PMA Adaptation parameters.
- E-Tile Transceiver PHY User Guide: Dynamic Reconfiguration Examples
  Information about configuring PMA parameters.
- Serial Lite IV Intel FPGA IP Clock Architecture on page 28
- Clock Signals on page 47

6. Serial Lite IV Intel FPGA IP Interface Signals

6.1. Clock Signals

Table 22. Clock Signals

Name	Width	Direction	Description
`tx_core_clkout`	1	Output	TX core clock for the TX custom PCS interface, TX MAC and user logics in the TX datapath. This clock is generated from the custom PCS block.
`rx_core_clkout`	1	Output	RX core clock for the RX custom PCS interface, RX deskew FIFO, RX MAC and user logics in the RX datapath. This clock is generated from the custom PCS block.
`xcvr_ref_clk`	1	Input	Transceiver reference clock. The IP supports reference clocks provided from separate clock chips or oscillators with a tolerance of ±100 ppm clock variation between the different clock chips or oscillators. Refer to *Parameters* for supported frequency range.
`reconfig_clk`	1	Input	Input clock for transceiver reconfiguration interface. The clock frequency is 100 to 162 MHz. Connect this input clock signal to external clock circuits or oscillators.
`xcvr_ref_clk_1`	1	Input	Transceiver reference clock used for preservation of unused transceiver channels. This clock is only applicable in PAM4 mode when you turn on **Preserve unused transceiver channels for PAM4** in the IP parameter editor. In NRZ mode, this clock is available by default.

Related Information

Parameters on page 45

6.2. Reset Signals

Table 23. Reset Signals

Name	Width	Direction	Clock Domain	Description
`tx_core_rst_n`	1	Input	Asynchronous	Active-low reset signal. Resets the Serial Lite IV TX MAC.
`rx_core_rst_n`	1	Input	Asynchronous	Active-low reset signal. Resets the Serial Lite IV RX MAC.
`tx_pcs_fec_phy_reset_n`	1	Input	Asynchronous	Active-low reset signal. Resets the Serial Lite IV TX custom PCS.
`rx_pcs_fec_phy_reset_n`	1	Input	Asynchronous	Active-low reset signal.

continued...

Name	Width	Direction	Clock Domain	Description
				Resets the Serial Lite IV RX custom PCS.
`reconfig_reset`	• lane/2 (PAM4 mode) • lane/4 (NRZ mode)	Input	`reconfig_clk`	Active-high reset signal. Resets the Avalon memory-mapped interface reconfiguration block.
`csr_phy_reset_n`	1	Input	Asynchronous	Active-low hard global reset signal. Resets the TX PCS, RX PCS, transceivers (transceiver configuration registers and interface), and reconfiguration registers. This reset leads to the deassertion of the `phy_tx_lanes_stable` and `phy_rx_pcs_ready` output signals.

6.3. MAC Signals

Table 24. TX MAC Signals

In this table, N represents the number of lanes set in the IP parameter editor.

Name	Width	Direction	Clock Domain	Description
`tx_avs_ready`	1	Output	`tx_core_clkout`	Avalon streaming signal. When asserted, indicates that the TX MAC is ready to accept data.
`tx_avs_data`	• (64**N*)*2 (PAM4 mode) • 64**N* (NRZ mode)	Input	`tx_core_clkout`	Avalon streaming signal. TX data.
`tx_avs_channel`	8	Input	`tx_core_clkout`	Avalon streaming signal. The channel number for data being transferred on the current cycle. This signal is not available in Basic mode.
`tx_avs_valid`	1	Input	`tx_core_clkout`	Avalon streaming signal. When asserted, indicates the TX data signal is valid.
`tx_avs_startofpacket`	1	Input	`tx_core_clkout`	Avalon streaming signal. When asserted, indicates the start of a TX data packet. Assert for only a single clock cycle for each packet. This signal is not available in Basic mode.
`tx_avs_endofpacket`	1	Input	`tx_core_clkout`	Avalon streaming signal. When asserted, indicates the end of a TX data packet. Assert for only a single clock cycle for each packet. This signal is not available in Basic mode.
`tx_avs_empty`	5	Input	`tx_core_clkout`	Avalon streaming signal. Indicates the number of non-valid words in the final burst of the TX data. This signal is not available in Basic mode.

continued...

Name	Width	Direction	Clock Domain	Description
`tx_num_valid_bytes_eob`	4	Input	`tx_core_clkout`	Indicates the number of valid bytes in the last word of the final burst. This signal is not available in Basic mode.
`tx_is_usr_cmd`	1	Input	`tx_core_clkout`	When asserted, this signal initiate a user-defined information cycle. Assert this signal at the same clock cycle as `tx_startofpacket` assertion. This signal is not available in Basic mode.
`tx_link_up`	1	Output	`tx_core_clkout`	When asserted, indicates the TX data link is ready for data transmission.
`tx_link_reinit`	1	Output	`tx_core_clkout`	When asserted, this signal initiates lanes re-alignment. Assert this signal for one clock cycle to trigger the MAC to send `ALIGN` CW.
`crc_error_inject`	*N*	Input	`tx_core_clkout`	When asserted, the MAC injects a CRC32 error to selected lanes.
`tx_error`	5	Output	`tx_core_clkout`	Not used.

The following timing diagram shows an example of TX data transmissions of 10 words from user logic across 10 TX serial lanes.

Figure 29. TX Data Transmission Timing Diagram

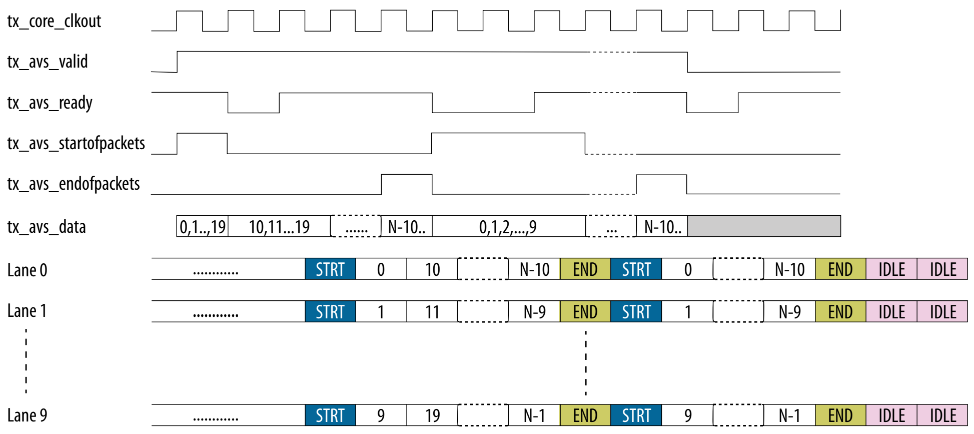


Table 25. RX MAC Signals

In this table, N represents the number of lanes set in the IP parameter editor.

Name	Width	Direction	Clock Domain	Description
`rx_avs_ready`	1	Input	`rx_core_clkout`	Avalon streaming signal. When asserted, indicates that the user logic is ready to accept data.
`rx_avs_data`	(64**N*)*2 (PAM4 mode) 64**N* (NRZ mode)	Output	`rx_core_clkout`	Avalon streaming signal. RX data.
`rx_avs_channel`	8	Output	`rx_core_clkout`	Avalon streaming signal. The channel number for data being received on the current cycle.

continued...

Name	Width	Direction	Clock Domain	Description
				This signal is not available in Basic mode.
`rx_avs_valid`	1	Output	`rx_core_clkout`	Avalon streaming signal. When asserted, indicates the RX data signal is valid.
`rx_avs_startofpacket`	1	Output	`rx_core_clkout`	Avalon streaming signal. When asserted, indicates the start of an RX data packet. Assert for only a single clock cycle for each packet. This signal is not available in Basic mode.
`rx_avs_endofpacket`	1	Output	`rx_core_clkout`	Avalon streaming signal. When asserted, indicates the end of an RX data packet. Assert for only a single clock cycle for each packet. This signal is not available in Basic mode.
`rx_avs_empty`	5	Output	`rx_core_clkout`	Avalon streaming signal. Indicates the number of non-valid words in the final burst of the RX data. This signal is not available in Basic mode.
`rx_num_valid_bytes_eob`	4	Output	`rx_core_clkout`	Indicates the number of valid bytes in the last word of the final burst. This signal is not available in Basic mode.
`rx_is_usr_cmd`	1	Output	`rx_core_clkout`	When asserted, this signal initiate a user-defined information cycle. Assert this signal at the same clock cycle as `tx_startofpacket` assertion. This signal is not available in Basic mode.
`rx_link_up`	1	Output	`rx_core_clkout`	When asserted, indicates the RX data link is ready for data reception.
`rx_link_reinit`	1	Input	`rx_core_clkout`	When asserted, this signal initiates lanes re-alignment. If you disable **Enable Auto Alignment**, assert this signal for one clock cycle to trigger the MAC to re-align the lanes. If the **Enable Auto Alignment** is set, the MAC re-align the lanes automatically. Do not assert this signal when **Enable Auto Alignment** is set.
`rx_error`	(*N**2*2)+3 (PAM4 mode) (*N**2)*3 (NRZ mode)	Output	`rx_core_clkout`	When asserted, indicates error conditions occur in the RX datapath. • [(*N**2+2):*N*+3] = Indicates PCS error for specific lane. • [*N*+2] = Indicates alignment error. Re-initialize lane alignment if this bit is asserted. • [*N*+1]= Indicates data is forwarded to the user logic when user logic is not ready. • [*N*] = Indicates loss of alignment. • [(*N*-1):0] = Indicates the data contains CRC error.

6.4. Transceiver Reconfiguration Signals

Table 26. PCS Reconfiguration Signals

In this table, *N* represents the number of lanes set in the IP parameter editor.

Name	Width	Direction	Clock Domain	Description
`phy_reconfig_read`	*N*/2 (PAM4 mode) *N* (NRZ mode)	Input	`reconfig_clk`	PCS reconfiguration read command signals.
`phy_reconfig_write`	*N*/2 (PAM4 mode) *N* (NRZ mode)	Input	`reconfig_clk`	PCS reconfiguration write command signals.
`phy_reconfig_address`	21*(*N*/2) (PAM4 mode) 19*N (NRZ mode)	Input	`reconfig_clk`	Specifies PCS reconfiguration Avalon memory-mapped interface address in a selected lane. For NRZ mode, each lane has 19 bits and lane 0 address starts from `phy_reconfig_address[18:0]`. Example, for a 4-lane NRZ design: • `phy_reconfig_address[18:0]` = address for lane 0. • `phy_reconfig_address[37:19]` = address for lane 1. • `phy_reconfig_address[56:38]` = address for lane 2. • `phy_reconfig_address[75:57]` = address for lane 3. For PAM4 mode, the lower 19 bits within the 21-bit bus specify the address and the upper 2 bits specify the lane. Example, for a 4-lane PAM4 design: • `phy_reconfig_address[18:0]` = address for lane 0 and 1, `phy_reconfig_address[20:19]` = lane number 0 and 1. • `phy_reconfig_address[39:21] = address for lane 2 and 3, phy_reconfig_address[41:40]` = lane number 2 and 3.
`phy_reconfig_readdata`	32*(*N*/2) 32**N* (NRZ mode)	Output	`reconfig_clk`	Specifies PCS reconfiguration data to be read by a ready cycle in a selected lane.
`phy_reconfig_waitrequest`	*N*/2 (PAM4 mode) *N* (NRZ mode)	Output	`reconfig_clk`	Represents PCS reconfiguration Avalon memory-mapped interface stalling signal in a selected lane.
`phy_reconfig_writedata`	32*(*N*/2) (PAM4 mode) 32**N* (NRZ mode)	Input	`reconfig_clk`	Specifies PCS reconfiguration data to be written on a write cycle in a selected lane.
`phy_reconfig_readdata_valid`	*N*/2 (PAM4 mode) *N* (NRZ mode)	Output	`reconfig_clk`	Specifies PCS reconfiguration received data is valid in a selected lane.

Table 27. PMA Reconfiguration Signals

In this table, N represents the number of lanes set in the IP parameter editor.

Name	Width	Direction	Clock Domain	Description
`xcvr_reconfig_read`	*N**2 (PAM4 mode) *N* (NRZ mode)	Input	`reconfig_clk`	PMA reconfiguration read command signals.
`xcvr_reconfig_write`	*N**2 (PAM4 mode) *N* (NRZ mode)	Input	`reconfig_clk`	PMA reconfiguration write command signals.
`xcvr_reconfig_address`	19**N**2 (PAM4 mode) 19**N* (NRZ mode)	Input	`reconfig_clk`	Specifies PMA Avalon memory-mapped interface address in a selected lane. In both PAM4 ad NRZ modes, each lane has 19 bits and lane 0 address starts from `xcvr_reconfig_address[18:0]`. Example, for a 4-lane design: • `xcvr_reconfig_address[18:0]` = address for lane 0. • `xcvr_reconfig_address[37:19]` = address for lane 1. • `xcvr_reconfig_address[56:38]` = address for lane 2. • `xcvr_reconfig_address[75:57]` = address for lane 3.
`xcvr_reconfig_readdata`	8**N**2 (PAM4 mode) 8**N* (NRZ mode)	Output	`reconfig_clk`	Specifies PMA data to be read by a ready cycle in a selected lane.
`xcvr_reconfig_waitrequest`	*N**2 (PAM4 mode) *N* (NRZ mode)	Output	`reconfig_clk`	Represents PMA Avalon memory-mapped interface stalling signal in a selected lane.
`xcvr_reconfig_writedata`	8**N**2 (PAM4 mode) 8**N* (NRZ mode)	Input	`reconfig_clk`	Specifies PMA data to be written on a write cycle in a selected lane.

Table 28. RS-FEC Reconfiguration Signals

Name	Width	Direction	Clock Domain	Description
`rsfec_reconfig_read`	1	Input	`reconfig_clk`	RS-FEC reconfiguration read command signal.
`rsfec_reconfig_write`	1	Input	`reconfig_clk`	RS-FEC reconfiguration write command signal.
`rsfec_reconfig_address`	11 + lane offset	Input	`reconfig_clk`	Specifies RS-FEC reconfiguration Avalon memory-mapped interface address. For PAM4 mode: • If *N*/2 = 1, lane offset is 0 • If *N*/2 = 2, lane offset is 1 • If *N*/2 = 3, lane offset is 2 • If *N*/2 = 4, lane offset is 2 For NRZ mode: • For number of lanes from 1 to 4, the lane offset is 0. • For number of lanes from 5 to 8, the lane offset is 1. • For number of lanes from 9 to 16, the lane offset is 2.

continued...

Name	Width	Direction	Clock Domain	Description
`rsfec_reconfig_readdata`	8	Output	`reconfig_clk`	Specifies RS-FEC reconfiguration data to be read by a ready cycle in a selected lane.
`rsfec_reconfig_waitrequest`	1	Output	`reconfig_clk`	Represents RS-FEC reconfiguration Avalon memory-mapped interface stalling signal in a selected lane.
`rsfec_reconfig_writedata`	8	Input	`reconfig_clk`	Specifies RS-FEC reconfiguration data to be written on a write cycle in a selected lane.

6.5. PMA Signals

Table 29. PMA Signals

In this table, N represents the number of lanes set in the IP parameter editor.

Name	Width	Direction	Clock Domain	Description
`phy_tx_lanes_stable`	*N**2 (PAM4 mode) *N* (NRZ mode)	Output	Asynchronous	When asserted, indicates TX datapath is ready to send data.
`tx_pll_locked`	*N**2 (PAM4 mode) *N* (NRZ mode)	Output	Asynchronous	When asserted, indicates the TX PLL has achieved lock status.
`phy_ehip_ready`	*N**2 (PAM4 mode) *N* (NRZ mode)	Output	Asynchronous	When asserted, indicates that the custom PCS has completed internal initialization and ready for transmission. This signal asserts after `tx_pcs_fec_phy_reset_n` and `tx_pcs_fec_phy_reset_n`are deasserted.
`tx_serial_data`	*N**2 (PAM4 mode) *N* (NRZ mode)	Output	TX serial clock	TX serial pins. For PAM4 mode, you must only use the even channels (0, 2, 4, ... , (N-1)*2) as the odd channels are inactive.
`rx_serial_data`	*N**2 (PAM4 mode) *N* (NRZ mode)	Input	RX serial clock	RX serial pins. For PAM4 mode, you must only use the even channels (0, 2, 4, ... , (N-1)*2) as the odd channels are inactive.
`phy_rx_block_lock`	*N**2 (PAM4 mode) *N* (NRZ mode)	Output	Asynchronous	When asserted, indicates that the 66b block alignment has completed for the lanes.
`rx_cdr_lock`	*N**2 (PAM4 mode) *N* (NRZ mode)	Output	Asynchronous	When asserted, indicates that the recovered clocks are locked to data.
`phy_rx_pcs_ready`	*N**2 (PAM4 mode) *N* (NRZ mode)	Output	Asynchronous	When asserted, indicates that the RX lanes of the corresponding Ethernet channel are fully aligned and ready to receive data.
`phy_rx_hi_ber`	*N**2 (PAM4 mode) *N* (NRZ mode)	Output	Asynchronous	When asserted, indicates that the RX PCS of the corresponding Ethernet channel is in a HI BER state.

7. Designing with Serial Lite IV Intel FPGA IP

7.1. Reset Guidelines

Follow these reset guidelines to implement your system-level reset.

- Tie `tx_pcs_fec_phy_reset_n` and `rx_pcs_fec_phy_reset_n` signals together on the system level in order to reset the TX and RX PCS simultaneously.
- Assert `tx_pcs_fec_phy_reset_n`, `rx_pcs_fec_phy_reset_n`, `tx_core_rst_n`, `rx_core_rst_n`, and `reconfig_reset` signals at the same time. Refer to *Reset and Link Initialization* for more information about the IP reset and initialization sequences.
- Hold `tx_pcs_fec_phy_reset_n`, and `rx_pcs_fec_phy_reset_n` signals low, and `reconfig_reset` signal high for at least 200 ns to properly reset the custom PCS and the reconfiguration blocks.
- To achieve fast link-up between FPGA devices, reset the connected Serial Lite IV Intel FPGA IPs at the same time. Refer to *Serial Lite IV Stratix 10 FPGA IP Design Example User Guide* for information about monitoring the IP TX and RX link using the toolkit.

Related Information

- Reset and Link Initialization on page 29
- Serial Lite IV Intel Stratix 10 FPGA IP Design Example User Guide: Serial Lite IV IP Toolkit
- Serial Lite IV Intel Agilex 7 FPGA IP Design Example User Guide

7.2. Error Handling Guidelines

The following table lists the error handling guidelines for error conditions which may occur with the Serial Lite IV Intel FPGA IP design.

Table 30. Error Condition and Handling Guidelines

Error Condition	Guidelines
One or more lanes cannot establish communication after a given time frame.	Implement a time-out system to reset the link at the application level.
A lane loses communication after communication is established.	This may happen after or during the data transfer phases. Implement a link loss detection at the application level and reset the link.
A lane loses communication during the deskew process.	Implement link reinitialization process for the erroneous lane. You must ensure that the board routing does not exceed 320 UI.
Loss lane alignment after all lanes have been aligned.	This may happen after or during data transfer phases. Implement a lane alignment loss detection at the application level to restart the lane alignment process.

Related Information

- Reset and Link Initialization on page 29
- Serial Lite IV Intel Agilex 7 FPGA IP Design Example User Guide
  More information about link debugging sequence when debugging your design for Intel Agilex 7 devices.
- Serial Lite IV Intel Stratix 10 FPGA IP Design Example User Guide
  More information about link debugging sequence when debugging your design for Intel Stratix 10 devices.

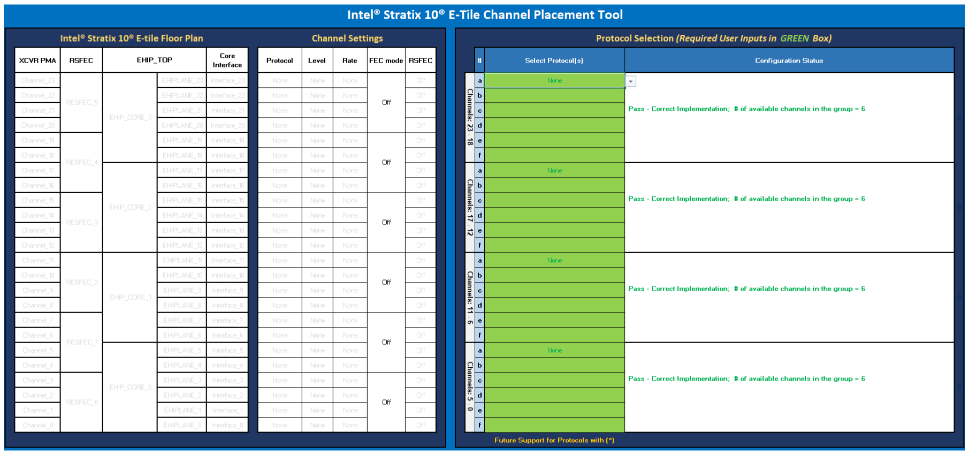
7.3. E-Tile Channel Placement Tool

Use the *Stratix 10 E-Tile Channel Placement Tool* to plan your channel placement for Serial Lite IV Intel FPGA IP.

E-tile supports Data Centers, 5G networks, Smart Grid, and other market segments. Ethernet, CPRI, and OTN are the backbone of these emerging and traditional technologies. The *E-Tile Channel Placement Tool*, in conjunction with the *Device Family Pin Connection Guidelines*, allows you to swiftly plan protocol placements in the product prior to reading comprehensive documentation and implementing designs in Quartus Prime software.

The Excel-based *E-Tile Channel Placement Tool*, supplemented with **Instructions**, **Legend**, and **Revision** tabs. Refer to the related information for the download link.

Figure 30. E-Tile Channel Placement Tool



Intel® Stratix 10® E-Tile Channel Placement Tool

Intel® Stratix 10® E-tile Floor Plan / Channel Settings

XCVR PMA	RSFEC	EHIP_TOP		Core Interface	Protocol	Level	Rate	FEC mode	RSFEC
Channel_23	RESFEC_5	EHIP_CORE_3	EHIPLANE_23	Interface_23	None	None	None	Off	Off
Channel_22			EHIPLANE_22	Interface_22	None	None	None		Off
Channel_21			EHIPLANE_21	Interface_21	None	None	None		Off
Channel_20			EHIPLANE_20	Interface_20	None	None	None		Off
Channel_19	RESFEC_4		EHIPLANE_19	Interface_19	None	None	None	Off	Off
Channel_18			EHIPLANE_18	Interface_18	None	None	None		Off
Channel_17		EHIP_CORE_2	EHIPLANE_17	Interface_17	None	None	None		Off
Channel_16			EHIPLANE_16	Interface_16	None	None	None		Off
Channel_15	RESFEC_3		EHIPLANE_15	Interface_15	None	None	None	Off	Off
Channel_14			EHIPLANE_14	Interface_14	None	None	None		Off
Channel_13			EHIPLANE_13	Interface_13	None	None	None		Off
Channel_12			EHIPLANE_12	Interface_12	None	None	None		Off
Channel_11	RESFEC_2	EHIP_CORE_1	EHIPLANE_11	Interface_11	None	None	None	Off	Off
Channel_10			EHIPLANE_10	Interface_10	None	None	None		Off
Channel_9			EHIPLANE_9	Interface_9	None	None	None		Off
Channel_8			EHIPLANE_8	Interface_8	None	None	None		Off
Channel_7	RESFEC_1		EHIPLANE_7	Interface_7	None	None	None	Off	Off
Channel_6			EHIPLANE_6	Interface_6	None	None	None		Off
Channel_5		EHIP_CORE_0	EHIPLANE_5	Interface_5	None	None	None		Off
Channel_4			EHIPLANE_4	Interface_4	None	None	None		Off
Channel_3	RESFEC_0		EHIPLANE_3	Interface_3	None	None	None	Off	Off
Channel_2			EHIPLANE_2	Interface_2	None	None	None		Off
Channel_1			EHIPLANE_1	Interface_1	None	None	None		Off
Channel_0			EHIPLANE_0	Interface_0	None	None	None		Off

Protocol Selection *(Required User Inputs in GREEN Box)*

	#	Select Protocol(s)	Configuration Status
Channels: 23 - 18	a	None	Pass - Correct Implementation; # of available channels in the group = 6
	b		
	c		
	d		
	e		
	f		
Channels: 17 - 12	a	None	Pass - Correct Implementation; # of available channels in the group = 6
	b		
	c		
	d		
	e		
	f		
Channels: 11 - 6	a	None	Pass - Correct Implementation; # of available channels in the group = 6
	b		
	c		
	d		
	e		
	f		
Channels: 5 - 0	a	None	Pass - Correct Implementation; # of available channels in the group = 6
	b		
	c		
	d		
	e		
	f		

Future Support for Protocols with (*)

Related Information

- E-Tile Channel Placement Tool
- Stratix 10 Device Family Pin Connection Guidelines
- Agilex 7 Device Family Pin Connection Guidelines

8. Serial Lite IV Intel FPGA IP Registers

There is no register available for the Serial Lite IV MAC. However, the IP utilizes the RS-FEC and PMA registers in the custom PCS. The following table lists the base addresses for the RS-FEC and PMA registers.

Table 31. RS-FEC and PMA Register Base Address

Register Type	Address Range
TX and RX RS-FEC registers	0x000–0x2FF
PMA capability registers	0x40000–0x40144
PMA Avalon memory-mapped interface registers	0x000–0x207

For information on RS-FEC and PMA registers, refer to the *E-Tile Transceiver PHY User Guide*.

Related Information

- E-Tile Transceiver PHY User Guide: RS-FEC Registers
- E-Tile Transceiver PHY User Guide: PMA Register Map

9. Serial Lite IV Intel FPGA IP User Guide Archives

For the latest and previous versions of this user guide, refer to Serial Lite IV Intel FPGA IP User Guide. If an IP or software version is not listed, the user guide for the previous IP or software version applies.

IP versions are the same as the Quartus Prime Design Suite software versions up to v19.1. From Quartus Prime Design Suite software version 19.2 or later, IP cores have a new IP versioning scheme.

10. Document Revision History for the Serial Lite IV Intel FPGA IP User Guide

Document Version	Quartus Prime Version	IP Version	Changes
2025.05.10	25.1	5.5.2	• Updated Tables: — *Serial Lite IV Intel FPGA IP Release Information* — *IP Version and Support Level* — *PMA Signals*
2023.10.02	23.3	5.1.0	• Updated product family name to "Intel Agilex® 7". • Updated Table: *Serial Lite IV Intel FPGA IP Release Information*. • Updated Table: *IP Version and Support Level*.
2023.02.10	22.4	4.0.0	• Added topic: *Starting the PMA Adaptation Flow*. • Updated Table: *Reset Signals* • Updated Table: *Serial Lite IV Intel FPGA IP Release Information* • Updated Table: *IP Version and Support Level*
2022.11.11	21.3	1.3.1	• Updated Figure: *RX Reset and Initialization Timing Diagram* • Updated *Serial Lite IV Intel FPGA IP User Guide Archives*
2021.12.01	21.3	1.3.1	• Added new Table: *Signal Mapping Between Serial Lite IV Intel FPGA IP and E-Tile Hard IP for Ethernet Intel FPGA IP*. • Corrected the **Transceiver data rate** values for NRZ with RS-FEC enabled mode in Table: *IP* from "**10.3125 Gbps** to **28.0 Gbps**" to "**10.000 Gbps** to **28.0 Gbps**".
2021.11.01	21.3	1.3.1	Added support for QuestaSim simulator.
2021.10.04	21.2	1.3.1	• Updated the guideline for error condition during deskew process in Table: *Error Condition and Handling Guidelines*.
2021.08.18	21.2	1.3.1	• Updated the following topics: — *TX Reset and Initialization Sequence* — *RX Reset and Initialization Sequence* • Updated the following Figures: *RX Reset and Initialization Timing Diagram* and *TX Reset and Initialization Timing Diagram*. • Corrected the description for `reconfig_reset` to clarify that this signal is an active-high reset signal in Table: *Reset Signals*. • Removed support for NCSim in the following tables: — Table: *Serial Lite IV IP Core Generated Files* — Table: *Intel FPGA IP Core Simulation Scripts*

continued...

Document Version	Quartus Prime Version	IP Version	Changes
2021.04.09	21.1	1.3.1	• Updated *RX Reset and Initialization Sequence*. • Made minor editorial edits to the document.
2021.04.01	21.1	1.3.1	• Added new parameters to Table: *IP*: — **Preserve unused transceiver channels for PAM4** — **Reference clock frequency for preserved channels** • Updated the **Transceiver data rate** value for PAM4 mode in Table: *IP* from **32.0 Gbps** to **32.5 Gbps**. • Added a new interface signal—`xcvr_ref_clk_1`. • Updated the following tables: — *Serial Lite IV Intel FPGA IP Release Information* — *IP Version and Support Level*
2020.03.12	19.4	1.2.0	• Updated effective rates for PAM4 transceiver mode in the *Bandwidth Efficiency* table. • Removed the effective rate calculation for PAM4 in the *Link Rate and Bandwidth Efficiency Calculation* topic. • Added *PMA Adaptation Flow* topic and link to the *Ethernet Adaptation Flow with Non-external AIB Clocking* in the *E-tile Hard IP User Guide: E-Tile Hard IP for Ethernet and E-Tile CPRI PHY Intel FPGA IPs* user guide.
2020.01.23	19.4	1.2.0	• Added ALM count and IP latency for NRZ with RS-FEC enabled mode in the *Stratix 10 Serial Lite IV Intel FPGA IP Resource Utilization* and *Agilex 7 Serial Lite IV Intel FPGA IP Resource Utilization* tables. • Added bandwidth efficiency values for NRZ with RS-FEC enabled mode in the *Bandwidth Efficiency* table. • Updated the effective rate calculations for both NRZ and PAM4 features in the *Link Rate and Bandwidth Efficiency Calculation* topic. • Updated values for the following parameters in the *Parameters* topic: — **Transceiver data rate** — **Set user-defined IP identifier** • Updated `rsfec_reconfig_address` signals description for NRZ mode. • Added description to decode addresses and lane numbers for the `phy_reconfig_address` and `xcvr_reconfig_address` signals.
2019.09.30	19.3	1.1.0	• Changed IP name to Serial Lite IV Intel FPGA IP. • Added support for Agilex 7 devices. • Added support for NRZ mode. • Added information for CRC feature. • Updated PCS, PMA and RS-FEC reconfiguration signals width.
2019.07.01	19.2	1.0.0	Initial release.