8b/10b PCS Core User Guide

UG-CORE-8B-10B-PCS-v1.0
May 2025
www.elitestek.com

Contents

Introduction

The 8b/10b Physical Coding Sublayer (PCS) core is designed to provide encoding and decoding capability for high-speed serial data transmission, in compliance with the industrial standard of 8b/10b line encoding scheme.

The core is designed to work with the user logic and higher layer protocol, and directly with the FPGA's built-in high-speed transceiver blocks, providing a general-purpose 8b/10b physical coding sublayer suitable for various serial communication protocols.

Features

The 8b/10b PCS core includes the following features:

- Supports transceiver interface widths of 20 bits and 40 bits
- Supports x1, x2, and x4 lane bonding modes
- Supports per-lane configuration with independent Tx and Rx paths
- Supports 8b/10b encoding (Tx) and decoding (Rx) with disparity control
- Configurable 10-bit character pattern for word alignment
- Detects code violations and disparity errors in the receive path
- Optional input and output pipeline registers at the encoder, decoder, and word aligner stages

Device Support

Refer to the TJ-Series Selector Guide and TP-Series Selector Guide to get the latest device list that supports transceivers.

Resource Utilization and Performance

Note: The resources and performance values provided are based on some of the supported FPGAs. These values are just guidance and can change depending on the device resource utilization, design congestion, and user design.

Table 1: TJ-Series Resource Utilization and Performance

FPGA	BW	Mode	Bonding Mode	Flop Count	LUT Count	BRAM Count	DSP Count	f_{MAX} (MHz)[1]	Efinity Version[2]
TJ375 N1156X C4	40	Tx_FIFO_Rx_FIFO	x1	1,366	4,298	0	0	312.5	2025.1
	40	Tx_FIFO_Rx_FIFO	x4	1,304	4,311	0	0	312.5	
	40	Tx_FIFO_Rx_Register	x1	1,619	4,292	0	0	312.5	
	20	Tx_FIFO_Rx_FIFO	x1	732	1,969	0	0	312.5	
	20	Tx_FIFO_Rx_FIFO	x4	732	1,945	0	0	312.5	
	20	Tx_FIFO_Rx_Register	x1	812	1,972	0	0	312.5	

Table 2: TP-Series Resource Utilization and Performance

FPGA	BW	Mode	Bonding Mode	Flop Count	LUT Count	BRAM Count	DSP Count	f_{MAX} (MHz)[1]	Efinity Version[2]
TP100 N676X C4	40	Tx_FIFO_Rx_FIFO	x1	1,366	4,298	0	0	312.5	2025.1
	40	Tx_FIFO_Rx_FIFO	x4	1,304	4,311	0	0	312.5	
	40	Tx_FIFO_Rx_Register	x1	1,619	4,292	0	0	270.0	
	20	Tx_FIFO_Rx_FIFO	x1	732	1,969	0	0	312.5	
	20	Tx_FIFO_Rx_FIFO	x4	732	1,945	0	0	312.5	
	20	Tx_FIFO_Rx_Register	x1	812	1,972	0	0	312.5	

Release Notes

You can refer to the IP Core Release Notes for more information about the IP core changes. The IP Core Release Notes are available on the Efinity Downloads page under each Efinity software release version.

Note: You must be logged in to the Support Center to view the IP Core Release Notes.

(1) Using default parameter settings.
(2) Using Verilog HDL.

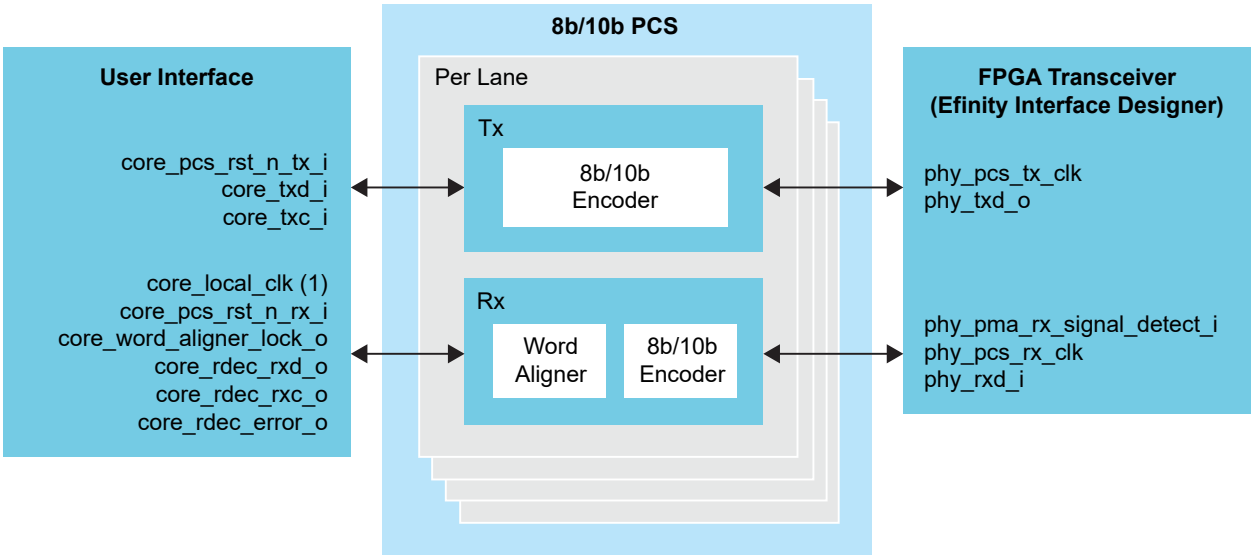
Functional Description

The 8b/10b PCS core consists of the following primary functional components:

- Transmit (Tx) path:
  - Performs 8b/10b encoding of data to convert 8-bit input symbols into 10-bit transmission characters.
  - Maintains running disparity to ensure a balanced number of 1s and 0s in the serial bit stream.
- Receive (Rx) Path:
  - Includes a predefined 10-bit character detection mechanism used to achieve word alignment.
  - Detects and flags invalid code groups or disparity errors to support error handling in the user logic.
  - Performs 8b/10b decoding of data to convert received 10-bit code groups back into 8-bit data and control information.

The core supports up to four independent lanes, each with dedicated transmit (Tx) and receive (Rx) data paths. Each lane operates independently, and lane activation is controlled through a user-configurable lane enable parameter. This design allows for scalable deployment across single-lane or multi-lane system architectures.

Figure 1: 8b/10b PCS Core Block Diagram



Notes: 1) *core_local_clk* is reserved and not implemented yet in early access.

Ports

Table 3: Clock and Reset Port

Signal	Direction	Bus Width	Clock Domain	Description
I<0-3>_phy_pcs_tx_clk	Input	1	N/A	TX clock source from PHY.
I<0-3>_phy_pcs_rx_clk	Input	1	N/A	Recovered clock from PHY.
I<0-3>_core_local_clk	Input	1	N/A	Local clock for deskew FIFO and elastic buffer. **Note:** Not used in early access.
I<0-3>_core_pcs_rst_n_tx_i	Input	1	Async	PCS TX reset.
I<0-3>_core_pcs_rst_n_rx_i	Input	1	Async	PCS RX reset.
I<0-3>_phy_pma_rx_signal_ detect_i	Output	1	Async	PMA receiver signal detect. Asserted high upon detection of a high-speed signal on the RX differential inputs.

Table 4: Tx Port

Signal	Direction	Bus Width	Clock Domain	Description
I<0-3>_core_txd_i	Input	(L<0-3>_BW/10)*8	I<0-3>_phy_pcs_tx_clk	Transmit data from user logic.
I<0-3>_core_txc_i	Input	L<0-3>_BW/10	I<0-3>_phy_pcs_tx_clk	Control signal indicating special characters from user logic.
I<0-3>_phy_txd_o	Output	L<0-3>_BW	I<0-3>_phy_pcs_tx_clk	Encoded data to be transmitted to PHY. See Note no. 1 for TXD bit mapping for TJ/TP PHY.

Table 5: Rx Port

Signal	Direction	Bus Width	Clock Domain	Description
I <0-3> _phy_rxd_i	Input	L<0-3>_BW	I<0-3>_phy_pcs_rx_clk	Receive data from PHY. See Note no. 1 for RXD bit mapping for TJ/TP PHY.
I <0-3> _core_word_ aligner_lock_o	Output	1	I<0-3>_phy_pcs_rx_clk	Indicates word aligner has successfully locked to alignment character.
I <0-3> _core_rdec_ rxd_o	Output	(L<0-3>_BW/10)*8	I<0-3>_phy_pcs_rx_clk	Decoded data.
I<0-3> _core_rdec_ rxc_o	Output	L<0-3>_BW/10	I<0-3>_phy_pcs_rx_clk	Decoded control signals.
I<0-3> _core_rdec_ error_o	Output	L<0-3>_BW/10	I<0-3>_phy_pcs_rx_clk	Decoder error.

Note:

1. TXD and RXD mapping for TJ-Series/TP-Series PHY.

L<0-3>_BW	TXD Output Mapping	RXD Input Mapping
20	TXD = {44'h0, l<0-3>_phy_txd_o}	l<0-3>_phy_rxd_i = RXD[19:0]
40	TXD = {12'h0, l<0-3>_phy_txd_o[39:20], 12'h0, l<0-3>_phy_txd_o[19:0]}	l<0-3>_phy_rxd_i = {RXD[51:32], RXD[19:0]}

   TXD is 64bits transmit data output to PHY; RXD is 64bits receive data input from PHY.

2. As this is an early access release, the current port list is about to 80 % finalized. Some changes may still occur as the design is refined towards final release.

Clock Sources

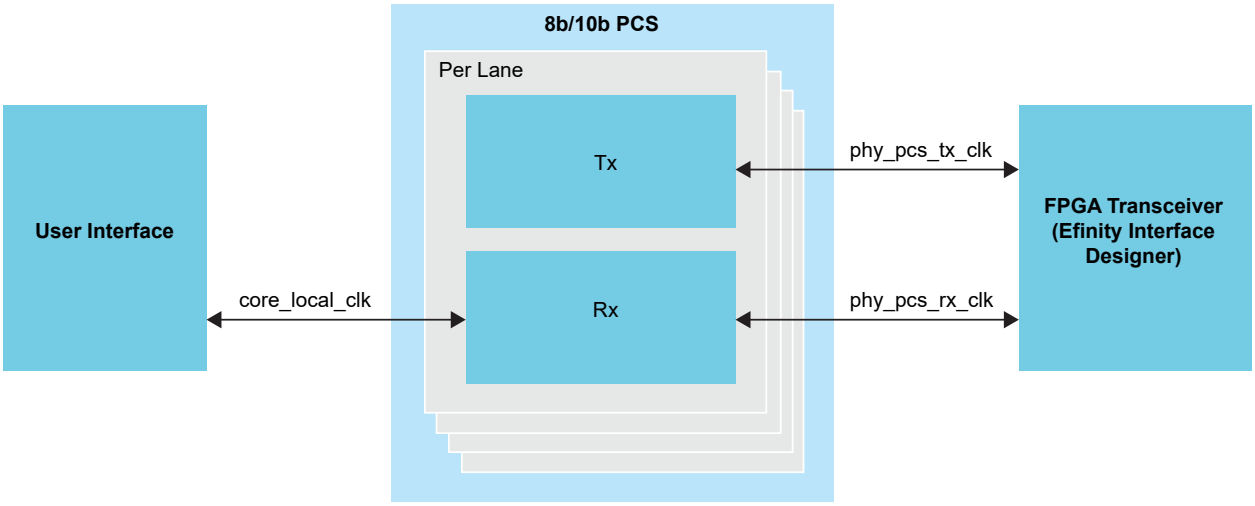
In 8b/10b PCS core, each lane requires three distinct clock sources to support the transmit, receive, and internal data handling functions:

- *phy_pcs_tx_clk*—Transmit clock source from the PHY. This clock drives the transmit (TX) path logic, which includes the 8b/10b encoder.
- *phy_pcs_rx_clk*—Recovered clock from the PHY for RX data capture. This clock is used to clock the receive (RX) path logic, which includes the decoder and the word aligner.
- *core_local_clk*—Local core clock used internally for deskew FIFOs and elastic buffers.

Note: This clock is reserved and not used in the early access version.

For each enabled lane (as configured by the lane enable parameter), the clock signals must be provided.

Figure 2: 8b/10b PCS Core Clock Block Diagram



In bonded configurations, of x2 or x4 modes, multiple lanes share a common clock source. Proper clock grouping ensures data alignment and timing consistency across lanes. However, the individual clock signal ports remain per-lane to support flexible grouping. **Table 6: Clock Connections** on page 8 outlines the clock source configuration based on the bonding mode and lane grouping.

Table 6: Clock Connections

Bonding Mode	Lane Grouping	Required Clock Connections
x1	Independent – L0, L1, L2, L3	Each lane operates independently. Each lane requires its own clock: `l0_phy_pcs_<tx/rx>_clk`,`l1_phy_pcs_<tx/rx>_clk`, `l2_phy_pcs_<tx/rx>_clk`, `l3_phy_pcs_<tx/rx>_clk`
x2	Optional per group – L0 + L1, L2 + L3	Lanes operate in pairs and share clocks within each pair. `l0_phy_pcs_<tx/rx>_clk` shared by L0 & L1. `l2_phy_pcs_<tx/rx>_clk` shared by L2 & L3.
x4	Fully bonded – L0 + L1 + L2 + L3	All lanes are grouped together. Only `l0_phy_pcs_<tx/rx>_clk` is required, shared across all lanes.

Mixed Configuration Example

In configurations where lanes are grouped differently, e.g., x2 + x1, the clock signals must still follow the grouping rules. Example:

- If L0 and L1 are bonded in **x2 mode**, both share the same clock:
  - `l0_phy_pcs_<tx/rx>_clk`
- If L2 and L3 operate independently in **x1 mode**, each has their own clock:
  - `l2_phy_pcs_<tx/rx>_clk`
  - `l3_phy_pcs_<tx/rx>_clk`

Reset Signals

In 8b/10b PCS core, each lane has a dedicated set of reset-related signals to manage the initialization and control of its transmit and receive paths:

- *core_pcs_rst_n_tx_i*—Active-low asynchronous reset for the Tx path.
- *core_pcs_rst_n_rx_i*—Active-low asynchronous reset for the Rx path.
- *phy_pma_rx_signal_detect_i*—PMA signal detect input, asserted high when a valid high-speed signal is detected on the RX differential pair.

These 3 reset signals are asynchronous, but the deassertion of each reset signal is synchronous.

Note: The PCS reset signals, `core_pcs_rst_n_<tx/rx>_i` should be released after the power-up handshake with the FPGA transceiver. This sequence is outside the scope of the 8b/10b PCS core and must be handled by the user before enabling the PCS logic. Refer to the power-up sequence chapter in the **Titanium PMA Direct User Guide** for further details on the power-up sequence of the FPGA transceiver.

Internally, the Rx path reset is gated by both the `core_pcs_rst_n_rx_i` and the `phy_pma_rx_signal_detect_i`. The RX logic becomes active when both signals are high, indicating that reset has been released and a valid input signal is present.

The Tx and Rx paths incorporate internal reset synchronizers to ensure safe reset deassertion across clock domains. As a result, the reset signals are not applied instantaneously in the logic. After asserting the reset inputs for either the Tx or Rx path, users must wait for 3 clock cycles before asserting any valid control or data signals. This delay ensures that the internal reset synchronization has been completed and that the 8b/10b PCS core is ready for normal operation.

In bonded configurations of x2 or x4 modes, reset signals across grouped lanes must be driven from the same source to ensure consistent and deterministic reset behavior. However, each lane maintains its individual reset input ports to support flexible grouping and independent lane control in non-bonded (x1) or mixed configurations. **Table 7: Reset Signals** on page 10 summarizes the reset signal requirements for each bonding mode and lane configuration.

Table 7: Reset Signals

Bonding Mode	Lane Grouping	Required Clock Connections
x1	Independent – L0, L1, L2, L3	Each lane can have its own reset: `l0_core_pcs_rst_n_<tx/rx>_i`,`l1_core_pcs_rst_n_<tx/rx>_i`, `l2_phy_pcs_rst_n_<tx/rx>_i`, `l3_phy_pcs_rst_n_<tx/rx>_i`.
x2	Optional per group – L0 + L1, L2 + L3	`l0_core_pcs_rst_n_<tx/rx>_i` and `l1_core_pcs_rst_n_<tx/rx>_i` must come from the same source. `l2_core_pcs_rst_n_<tx/rx>_i` and `l3_core_pcs_rst_n_<tx/rx>_i` must come from the same source.
x4	Fully bonded – L0 + L1 + L2 + L3	All reset signals must be driven from the same source.

Transmit Path (Tx)

8b/10b Encoder

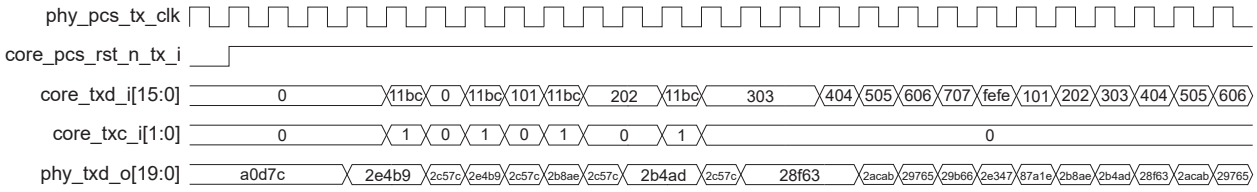
The 8b/10b encoder has the following functions and characteristics:

- Encoder input data widths of 16 bits or 32 bits into multiple 10-bit encoded symbols, following standard 8b/10b encoding practices.
- The module features a wrapper that instantiates multiple 8b/10b encoders internally, allowing for configurable output widths: either 20 bits or 40 bits, based on the bit word (BW) parameter selection.
- Wrapper block:
  - The wrapper manages the configuration and instantiation of multiple 8b/10b encoder instances based on the BW parameter.
  - For BW = 20:
    - Instantiates two 8b/10b encoders to process two 8-bit data words, producing a 20-bit encoded output.
  - For BW = 40:
    - Instantiates four 8b/10b encoders to process four 8-bit data words, producing a 40-bit encoded output.
- Encoder Instances:
  - Each 8b/10b encoder instance is responsible for encoding an 8-bit data word or control character into a 10-bit code group.
  - For comprehensive details on 8b/10b encoding, including the mapping of 8-bit data words and control characters to 10-bit code groups, refer to the IEEE 802.3 standard.

Tx Path Timing Diagram

The following timing diagram illustrates the flow of data from the input to the encoded output.

Figure 3: Tx Path Timing Diagram



In the **Figure 3: Tx Path Timing Diagram** on page 11, the `core_pcs_rst_n_tx_i` is asserted (set high) to release the Tx logic from reset. However, you must wait at least 3 clock cycles after the `core_pcs_rst_n_tx_i` is asserted before driving the valid data to the `core_txd_i` and `core_txc_i` because of internal reset synchronizers.

The encoder begins processing and transmitting encoded symbols when the valid data is presented to the `core_txd_i` and `core_txc_i`. The encoded output, `phy_txd_o` becomes valid after a few cycles of internal processing latency.

Note: Latency from the input to the encoded output may vary depending on the user parameters. Refer to **Latency** on page 15 for the formula and examples.

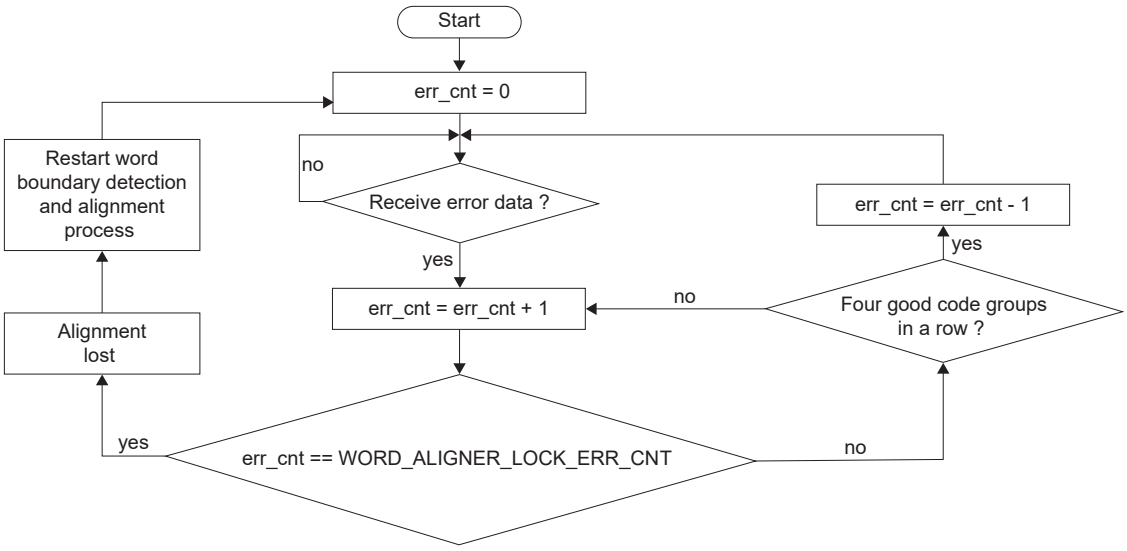
Receive Path (Rx)

Word Aligner

In 8b/10b PCS core, the word aligner has the following functions and characteristics:

- When data is received, the bits are just a continuous stream. The receiver does not know where each 10-bit symbol starts. Therefore, decoding without alignment may cause the decoding of wrong bits, resulting in garbage data.
- This is designed to detect and lock onto a user-defined 10-bit alignment character within a deserialized serial data stream, ensuring proper word boundary detection before decoding operations.
- Each incoming bit word (BW) is scanned bit-by-bit for a valid alignment character. Valid patterns are defined as:
  - Positive alignment character, `COM_CHAR_P` – Default: 10'h283 (K28.5)
  - Negative alignment character, `COM_CHAR_N` – Default: 10'h17c (K28.5)
- To achieve a lock, the module must observe:
  - Multiple alignment-bearing cycles—Cycles containing at least one instance of `COM_CHAR_P` or `COM_CHAR_N` with no decode error between them.
    - If any decode error is detected during this phase:
      - The current alignment attempt is discarded.
      - The word aligner resumes scanning for a new valid alignment character.
    - These alignment-bearing cycles do not have to occur consecutively; they may interleave with cycles containing other (non-alignment) characters.
- Once a clean sequence of alignment-bearing cycles is observed without error, the word aligner asserts `word_aligner_lock_o`.
- Users must ignore RX output data until word_aligner_lock_o is high, as output data may be misaligned and invalid during this period.
- Valid data output is guaranteed only after the word aligner achieves lock.
- To ensure robustness against invalid 10-bit symbols and disparity errors caused by link instability, the 8b/10b PCS core includes a built-in error handling mechanism:

Figure 4: Error Handling Mechanism Flow



  - Each decode error increments an internal error counter (`err_cnt`).
  - If no errors occur for 4 consecutive cycles, the counter decrements by 1.
  - If `err_cnt` reaches a threshold (`WORD_ALIGNER_LOCK_ERR_CNT`, default 4), the word aligner considers the link unstable and automatically exits the locked

state. The signal `core_word_aligner_lock_o` is deasserted, and the word aligner resets to its initial state, and restart the alignment process.
  - The word aligner will then wait for next valid alignment character from PHY. Once alignment is reacquired, `core_word_aligner_lock_o` is reasserted.

8b/10b Decoder

The 8b/10b decoder has the following functions and characteristics:

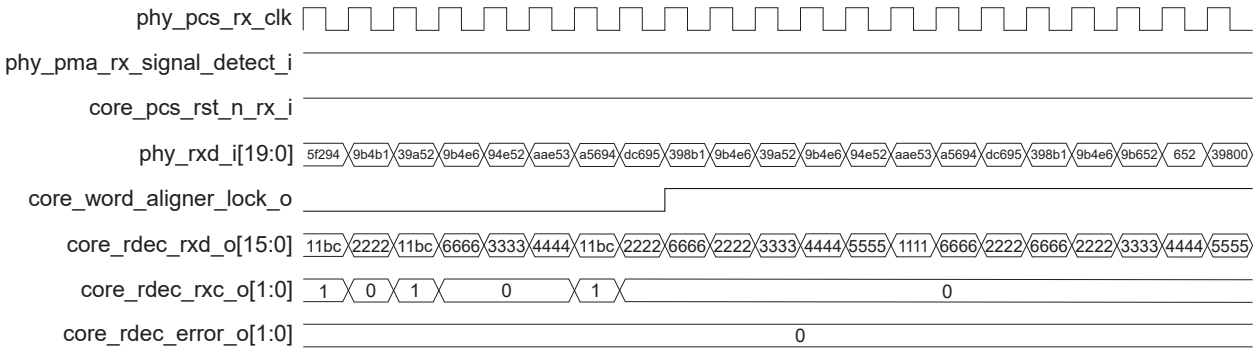
- Decode input data widths of either 20 bits or 40 bits into multiple 8-bit data symbols, aligning with standard 8b/10b decoding practices.
- The module features a wrapper that instantiates multiple 8b/10b decoders internally, allowing configurable input widths: either 20 bits or 40 bits, based on parameter selection, `BW`.
- Wrapper block:
  - The wrapper manages the configuration and instantiation of multiple 8b/10b decoders based on the parameter `BW`.
  - For a 20-bit input (`BW = 20`):
    - Instantiates two 8b/10b decoders to process two 10-bit encoded symbols, producing a 16-bit decoded output.
  - For a 40-bit input (`BW = 40`):
    - Instantiates four 8b/10b decoders to process four 10-bit encoded symbols, producing a 32-bit decoded output.
- Decoder instances:
  - Each 8b/10b decoder instance decodes a 10-bit encoded symbol into an 8-bit data or control character.
  - The decoding process follows the 8b/10b encoding scheme defined in the IEEE 802.3 standard.
  - The wrapper coordinates the outputs of these decoder instances to reconstruct the original data word.

Rx Path Timing Diagram

Alignment Acquisition Timing Diagram (Before Lock)

The following timing diagram shows how the Rx path searches for characters alignment and waits for error-free cycles before asserting `word_aligner_lock_o`.

Figure 5: Alignment Acquisition Timing Diagram (Before Lock)



Initially, the `core_pcs_rst_n_rx_i` and the `phy_pma_rx_signal_detect_i` are asserted, allowing the Rx logic to begin processing input symbols. The word aligner starts scanning for alignment-bearing cycles to establish word boundaries. As valid alignment characters are detected and no decode errors occur, the word aligner approaches lock.

However, if a decode error is detected before lock is achieved, the internal alignment character tracking is reset, and `word_aligner_lock_o` remains deasserted. The word aligner resumes scanning for a new alignment character sequence. Until `word_aligner_lock_o` is asserted, the Rx path output is not guaranteed to be valid.

The word aligner asserts `word_aligner_lock_o` when a clean sequence of alignment-bearing cycles is observed without errors. From this point onward,

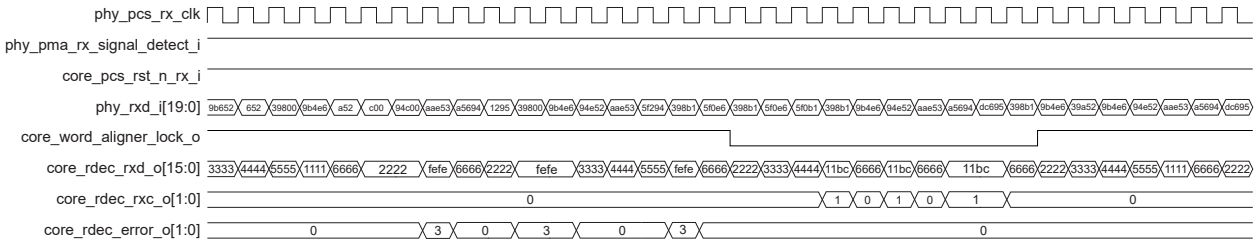
- `core_rdec_rxd_o` and `core_rdec_rxc_o` are valid and aligned.
- `core_rdec_error_o` remains low (assuming error-free input).
- The user may safely begin using the Rx output data.

Note: Latency from the input to the encoded output may vary depending on the user parameters. Refer to **Latency** on page 15 for the formula and examples.

Lock Loss and Recovery Timing Diagram (After Lock)

The following timing diagram illustrates how decode errors after alignment cause the word aligner to drop lock and re-enter the search state until a stable realignment is achieved.

Figure 6: Lock Loss and Recovery Timing Diagram (After Lock)



Under normal conditions, when `word_aligner_lock_o` is high, the output `core_rdec_rxd_o` and `core_rdec_rxc_o` are valid, and `core_rdec_error_o` remains all-zero.

If non-zero values appear on core_rdec_error_o across multiple cycles, it indicates persistent decode errors in the incoming stream. This may be caused by corrupted or noisy input data. When such errors accumulate, the PCS increments an internal error counter.

If 4 consecutive error-free cycles occur, the counter decrements by 1. If decode errors persist and the internal counter reaches the user-defined threshold (`WORD_ALIGNER_LOCK_ERR_CNT`, default = 4), the core deasserts `word_aligner_lock_o`, indicating a loss of alignment.

After a lock is dropped, the Rx path re-enters alignment character detection and error monitoring. During this time, `core_rdec_rxd_o` and `core_rdec_rxc_o` may be unstable or invalid. The `word_aligner_lock_o` is reasserted, and valid decoding resumes when a new alignment is established (based on valid alignment character sequences and error-free monitoring).

Latency

Transmit Path (Tx)

The latency of the 8b/10b encoder determines the number of clock cycles between a valid input and its corresponding encoded output. This latency depends on the parameters:

Tx Latency = `ENC_INPUT_REG` + 1

Receive Path (Rx)

After the word aligner has successfully locked to `COM_CHAR`, the latency for Rx Path determines the number of clock cycles between `phy_rxd_i` and its corresponding decoded output `core_rdec_rxd_o`. This latency depends on the parameters:

Rx Latency = `DEC_INPUT_REG` + `DEC_OUTPUT_REG` + `WA_INPUT_REG` + `WA_OUTPUT_REG` + 2 + 2*`OPTIMIZE_FOR_TIMING` + $B_{boundary}$

$B_{boundary}$ is a potential +1 cycle variation in latency due to word boundary realignment. However, the received data may not always be naturally aligned to the boundary because of serialization and deserialization processes. This misalignment introduces variations in latency, which must be considered in the system design.

Table 8: Examples of Latency in the Rx Path on page 15 shows latency examples of the Rx path.

Table 8: Examples of Latency in the Rx Path

Parameters				Latency (Cycles)	
DEC_INPUT_REG	DEC_OUTPUT_REG	WA_INPUT_REG	WA_OUTPUT_REG	OPTIMIZE_FOR_TIMING = 0	OPTIMIZE_FOR_TIMING = 1
0	1	0	1	4 or 5	6 or 7
0	1	1	0	4 or 5	6 or 7
0	1	1	1	5 or 6	7 or 8
1	1	1	1	6 or 7	8 or 9

Customizing the 8b/10b PCS

Table 9: 8b/10b PCS Core Parameters (General Tab)

Parameter	Options	Description
L<0-3>_EN	0, 1	Enable lane 0. Default: 1
L<0-3>_BW	20, 40	Transceiver width. Default: 20
L<0-3>_MODE	Tx_FIFO_Rx_FIFO, Tx_FIFO_Rx_Register, Tx_FIFO, Rx_FIFO	Mode. Default: Tx_FIFO_Rx_FIFO
L<0-3>_BONDING_MODE	x1, x2, x4	Bonding mode. x4 mode: • Enable all 4 lanes: L0, L1, L2, and L3. — Set `L<0-3>_EN = 1` for all four lanes. • All other lane parameters must be identical across all 4 lanes. x2 mode: • Enable 2 lanes, typically L0 and L1 or L2 and L3. — Set `L<0-3>_EN = 1` for the selected lanes. • All other lane parameters must be identical for both lanes. **Note:** Any mismatch in parameters across lanes in bundle mode can result in incorrect behavior, and such configuration errors are not detected or guarded by the RTL. Default: x1
L<0-3>_ENC_INPUT_REG	0, 1	Enables registered input at 8b/10b encoder. Default: 0
L<0-3>_DEC_INPUT_REG	0, 1	Enables registered input at 8b/10b decoder. Default: 0
L<0-3>_DEC_OUTPUT_REG	0, 1	Enables registered output at 8b/10b decoder. Default: 1
L<0-3> _WA_INPUT_REG	0, 1	Enables registered input at word aligner. **Note:** This parameter is ignored when `L<0-3>_MODE == Tx_FIFO_Rx_Register`. In this mode, the registered input is always enabled internally regardless of the value of this parameter. Default: 1
L<0-3> _WA_OUTPUT_REG	0, 1	Enables registered output at word aligner. Default: 1
L<0-3>_COM_CHAR_P	0x0 - 0x3FF	Positive disparity 10-bit K-character for comma alignment. **Note:** Refer to IEEE 8b/10b encoding specification for valid 8b/10b K-character (control code) Default: 0x283

Parameter	Options	Description
L<0-3>_COM_CHAR_N	0x0 - 0x3FF	Negative disparity 10-bit K-character for comma alignment. **Note:** Refer to IEEE 8b/10b encoding specification for valid 8b/10b K-character (control code) Default: 0x17C
L<0-3>_WORD_ALIGNER_ LOCK_ERR_CNT	1 – 31	Threshold value for error handling in word aligner. Determines how many errors cause loss of alignment. Default: 4
OPTIMIZE_FOR_TIMING	0, 1	Adds extra pipeline stages to improve timing closure at the cost of 2 cycles of additional latency. Default: 0

L<0-3>_FIFO_CLK_SEL	0,1	Rx clock selection.
		`0 – l<0-3>_phy_pcs_rx_clk`
		`1 – l<0-3>_core_local_clk`
		Default: 0
		Note: Not used in early access.

8b/10b PCS Testbench

Note: You must include all **.sv** files generated in the **/testbench** directory in your simulation.

The 8b/10b PCS core includes a simulation testbench that simulates the PCS operation.

Elitestek provides a simulation script for you to run the testbench quickly using the Questasim software. To run the Questasim testbench script, run `Make all` in a terminal application. You must have Questasim installed on your computer to use these scripts.

The testbench is set to the following settings:

- L0 enabled
- 40 bits
- Tx_FIFO_Rx_FIFO mode
- x1 mode

All unspecified parameters remain at their default settings.

Revision History

Table 10: Revision History

Date	Document Version	IP Version	Description
May 2025	1.0	1.0	Early release. (DOC-2470)