



Gowin GW1N-2 Hardened MIPI D-PHY RX

User Guide

IPUG778-1.2E, 09/26/2025

Copyright © 2025 Guangdong Gowin Semiconductor Corporation. All Rights Reserved.

GOWIN is a trademark of Guangdong Gowin Semiconductor Corporation and is registered in China, the U.S. Patent and Trademark Office, and other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders. No part of this document may be reproduced or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written consent of GOWINSEMI.

Disclaimer

GOWINSEMI assumes no liability and provides no warranty (either expressed or implied) and is not responsible for any damage incurred to your hardware, software, data, or property resulting from usage of the materials or intellectual property except as outlined in the GOWINSEMI Terms and Conditions of Sale. GOWINSEMI may make changes to this document at any time without prior notice. Anyone relying on this documentation should contact GOWINSEMI for the current documentation and errata.

Revision History

Date	Version	Description
06/01/2021	1.0E	Initial version published.
07/07/2021	1.01E	MIPI D-PHY Data Rates chart added.
02/15/2022	1.1E	● The note in Table 3-1 added. ● “Select clock source for HS lane output data” option removed from the configuration interface.
09/26/2025	1.2E	● “Table 3-1 MIPI_DPHY_RX Port Description” optimized. ● “3.4 Typical Timing” added.

Contents

Contents	i
List of Figures	ii
List of Tables	iii
1 About This Guide	1
1.1 Purpose	1
1.2 Related Documents	1
1.3 Terminology and Abbreviations	2
1.4 Support and Feedback	2
2 Overview	3
2.1 Features	3
2.2 Functional Description	4
3 MIPI D-PHY RX Primitives	5
3.1 Port Diagram	5
3.2 Port Description	7
3.3 Parameter Description	11
3.4 Typical Timing	11
3.5 Primitive Instantiation	12
4 MIPI D-PHY RX Configuration and Generation	20
4.1 MIPI D-PHY RX Configuration	20
4.2 MIPI D-PHY RX Generation File	22
Appendix A MIPI D-PHY Data Rates	23

List of Figures

Figure 2-1 MIPI D-PHY RX Diagram	4
Figure 3-1 MIPI_DPHY_RX Port Diagram.....	6
Figure 3-2 D-PHY HS Burst Data Transmission	11
Figure 3-3 MIPI D-PHY RX Typical Timing	12
Figure 4-1 MIPI D-PHY RX Configuration	21

List of Tables

Table 1-1 Terminology and Abbreviations	2
Table 3-1 MIPI_DPHY_RX Port Description	7
Table 3-2 MIPI_DPHY_RX Parameter Description.....	11
Table A-1 MIPI D-PHY Data Rates	24

1 About This Guide

1.1 Purpose

The purpose of Gowin GW1N-2 Hardened MIPI D-PHY RX User Guide is to help you learn the usage and features of hard MIPI D-PHY RX core by providing the descriptions of functions, ports, configuration, etc.

1.2 Related Documents

1. The latest user guides are available on the GOWINSEMI Website. You can find the related documents at www.gowinsemi.com:
 - [DS100, GW1N series of FPGA Products Data Sheet](#)
 - [UG103, GW1N series of FPGA Products Package and Pinout User Guide](#)
 - [UG171, GW1N-2 Pinout](#)
2. MIPI Alliance Specification for D-PHY, Version 1.2 August 2014, www.mipi.org

1.3 Terminology and Abbreviations

The terminology and abbreviations used in this manual are as shown in Table 1-1.

Table 1-1 Terminology and Abbreviations

Terminology and Abbreviations	Meaning
CSI	Camera Serial Interface
DSI	Display Serial Interface
HS	High Speed
I/O	Input/Output
LP	Low Power
MIPI	Mobile Industry Processor Interface
ODT	On-Die Termination

1.4 Support and Feedback

Gowin Semiconductor provides customers with comprehensive technical support. If you have any questions, comments, or suggestions, please feel free to contact us directly by the following ways.

Website: www.gowinsemi.com

E-mail: support@gowinsemi.com

2 Overview

GW1N-2 provides a standalone MIPI D-PHY RX supporting MIPI Alliance Standard for D-PHY Specification V1.2. The dedicated D-PHY core supports MIPI DSI and CSI-2 mobile video interfaces for cameras and displays.

2.1 Features

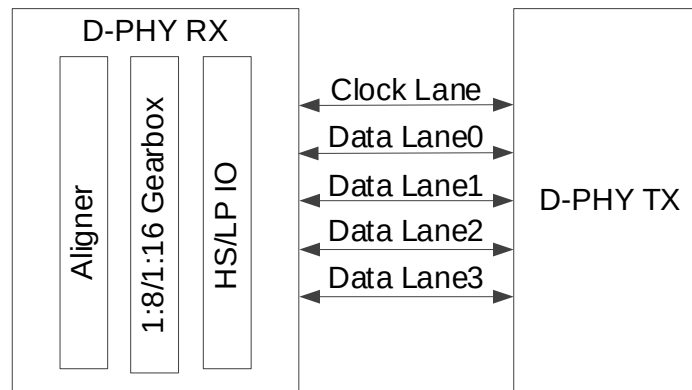
The main features of MIPI D-PHY RX are as follows:

- High Speed RX at up to 2 Gbps per lane.
- 1, 2 or 4 data lane and 1 clock lane support per PHY.
- Bidirectional Low-power (LP) mode at up to 10Mbps per lane.
- High-speed synchronization, including word alignment and lane alignment.
- 1:8 and 1:16 deserialization modes to FPGA fabric's user interface.
- Supports MIPI DSI and MIPI CSI-2 link layers.
- Available on the dedicated MIPI Bank (Bank6).

2.2 Functional Description

MIPI D-PHY RX includes three parts: HS/LP IO, 1:8/1:16 Gearbox, and Aligner, as shown in Figure 2-1.

Figure 2-1 MIPI D-PHY RX Diagram



HS/ LP I/O

Supports dynamic ODT, and dynamic LP TX/RX and HS RX.

1:8/1:16 Gearbox

You can configure 8-bit or 16-bit width by HS_8BIT_MODE.

Aligner

You can configure Word align and Lane align via WALIGN_BY and LALIGN_EN ports. The key of Word align can be user-defined (ALIGN_BYTE), supporting 2 bytes and 3 bytes mode, which is configured via ONE_BYTE0_MATCH port.

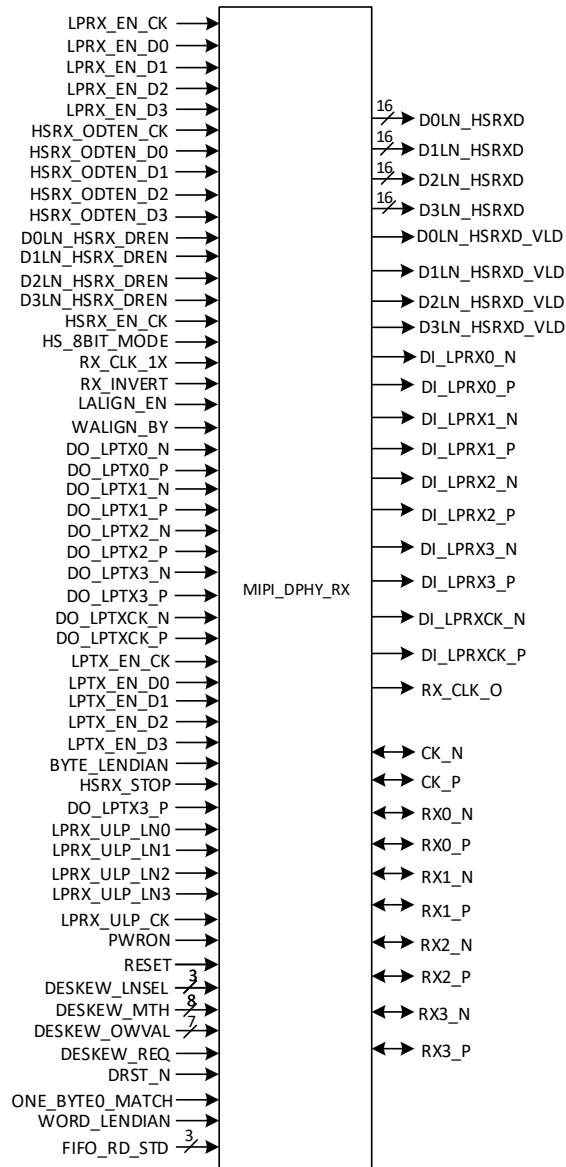
Note !

For the MIPI_DPHY_RX port description, you can see 3 MIPI D-PHY RX Primitive > Port Description.

3 MIPI D-PHY RX Primitives

3.1 Port Diagram

Figure 3-1 MIPI_DPHY_RX Port Diagram



3.2 Port Description

Table 3-1 MIPI_DPHY_RX Port Description

Port	I/O	Description
MIPI INTERFACE Signals		
CK_N ^[1]	inout	CK Lane Complement Input
CK_P ^[1]	inout	CK Lane True Input
RX<0~3>_N ^[1]	inout	Data Lane <0~3> Complement Input
RX<0~3>_P ^[1]	inout	Data Lane <0~3> True Input
CONFIG Signals		
BYTE_LENDIAN	input	Bit Data Little/Big Endian of 8bit 1'b1: Little endian, the bit entered first is in the low position of byte 1'b0: Big endian, the bit entered first is in the high position of byte
FIFO_RD_STD [2:0]	input	FIFO Read Threshold. Can only be 1 in 8-bits mode.
HS_8BIT_MODE	input	Selection of Data Width to Fabric 1'b1: 8bit output mode 1'b0: 16bit output mode
LALIGN_EN	input	Lane Aligner Enable 1'b1: lane aligner is enabled 1'b0: lane aligner is disabled
ONE_BYTE0_MATCH	input	Byte Count Match in Word Aligner 1'b1: three bytes, one more byte of 0 match before KEY, word aligner is searching {KEY,8'b0,8'b0}~{KEY[7],{KEY[6:0],1'b0},8'b0} (Little endian) 1'b0: two bytes, at most one byte of 0 match before KEY, word aligner is searching {KEY,8'b0}~{KEY[7],{KEY[6:0],1'b0}}(Little endian)
RX_INVERT	input	Data Polarity Selection
WALIGN_BY	input	Word aligner bypass 1'b1: bypass 1'b0: word aligner is enabled
WORD_LENDIAN	input	Data little/big endian of dual word(16bit, 8bit/word). Not used in 8bit data output mode. 1'b1: Little endian, first word in low byte of 16bit. Lane aligner is searching:{8'hx,KEY};

Port	I/O	Description
		1'b0: Big endian, first word in high byte of 16bit. Lane aligner is searching:{KEY,8'hex};
RESET and CLOCK Signals		
DRST_N	input	Digital reset, active low
HSRX_STOP	input	HSRX Clock Stop Signal for synchronization
PWRON	input	Power-On Control: 1'b1: HSRX on 1'b0: HSRX off to standby in low power state
RESET	input	Reset signal: 1'b1: reset all;
RX_CLK_1X	input	1X clock from fabric, max 93.75MHz@1.5Gbps
RX_CLK_O	output	RX output 1X clock, max 93.75MHz@1.5Gbps
HSRX Signals		
D<0~3>LN_HSRX_DREN	input	Data Lane<0~3> HSRX driver enabled: 1'b1 driver enabled 1'b0 driver disabled
D<0~3>LN_HSRXD[7/15:0]	output	Data Lane <0~3> HS data output to fabric 8bit output mode: Data Width=8 16bit output mode: Data Width=16
D<0~3>LN_HSRXD_VLD	output	Data Lane<0~3> HS data output valid to fabric
HSRX_EN_CK	input	CK Lane HSRX enabled
HSRX_ODTEN_CK	input	CK Lane HSRX ODT enabled: 1'b1 ODT enabled 1'b0 ODT disabled
HSRX_ODTEN_D<0~3>	input	Data Lane<0~3> HSRX ODT enabled: 1'b1 ODT enabled 1'b0 ODT disabled
LPRX Signals		
DI_LPRX<0~3>_N	output	Data Lane <0~3> Complement Pad LPRX input
DI_LPRX<0~3>_P	output	Data Lane <0~3> True Pad LPRX input
DI_LPRXCK_N	output	CK Lane Complement Pad LPRX input
DI_LPRXCK_P	output	CK Lane True Pad LPRX input
LPRX_EN_CK	input	CK Lane LPRX enabled
LPRX_EN_D<0~3>	input	Data Lane <0~3> LPRX enabled
LPTX Signals		
DO_LPTX<0~3>_N	input	Data Lane <0~3> Complement Pad LPTX output

Port	I/O	Description
DO_LPTX<0~3>_P	input	Data Lane <0~3> True Pad LPTX output
DO_LPTXCK_N	input	CK Lane Complement Pad LPTX output
DO_LPTXCK_P	input	CK Lane True Pad LPTX output
LPTX_EN_CK	input	CK Lane LPTX enabled
LPTX_EN_D<0~3>	input	Data Lane <0~3> LPTX enabled
Other Signals		
LPRX_ULP_LN0 ^[2]	input	Data Lane0 LPRX ULP Control: 1'b0 disable
LPRX_ULP_LN1 ^[2]	input	Data Lane1 LPRX ULP Control: 1'b0 disable
LPRX_ULP_LN2 ^[2]	input	Data Lane2 LPRX ULP Control: 1'b0 disable
LPRX_ULP_LN3 ^[2]	input	Data Lane3 LPRX ULP Control: 1'b0 disable
LPRX_ULP_CK ^[2]	input	CK Lane LPRX ULP Control: 1'b0 disable
DESKEW_LNSEL[2:0]	input	selection of lane to config delay overwrite value and make lane in deskew delay overwrite mode. 3'h0: NULL; select deskew error report output 3'h1: data lane0 3'h2: data lane1 3'h3: data lane2 3'h4: data lane3 3'h5: clock lane 3'h6: select lane aligner FIFO write full or read empty error report output 3'h7: select 4 data lane FIFO write full error report output NOTE! If any lane's deskew delay overwrite mode is ever active, a reset is a must before using this lane's deskew function normally.
DESKEW_MTH [7:0]	input	counter threshold for searching one edge
DESKEW_OWVAL [6:0]	input	IN deskew delay overwrite mode : set delay overwrite value of lane: 0~123. NOTE! This value is only registered into module when the lane is selected to config delay overwrite value. IN normal mode: [6]: deskew_rst_bypass mode 0: deskew's rstn is the same as drst_n 1: deskew's rstn is not controlled by drst_n [5]: deskew_fast_mode

Port	I/O	Description
		0: deskew is in initial mode 1: deskew is in fast mode [4]: deskew_en_low_delay 0: low delay is not enabled 1: low delay is enabled [3:0]: deskew_fast_loop_time
DESKEW_REQ	input	deskew function request to all data lanes

Note!

- ^[1] Dedicated to MIPI, no user constraints. For the detailed pins information, you can refer to [UG103, GW1N series of FPGA Products Package and Pinout User Guide](#), [UG171, GW1N-2 Pinout](#).
- ^[2] Usually it is set to '0' to disable the ULP function.

3.3 Parameter Description

Table 3-2 MIPI_DPHY_RX Parameter Description

Parameter	Default Value	Description
ALIGN_BYTE	8'b10111000	KEY for word aligner and lane aligner
MIPI_LANE0_EN	1'b0	1'b1, Lane0 on
MIPI_LANE1_EN	1'b0	1'b1, Lane1 on
MIPI_LANE2_EN	1'b0	1'b1, Lane2 on
MIPI_LANE3_EN	1'b0	1'b1, Lane3 on
MIPI_CK_EN	1'b1	CK Lane Select: 1'b1 CK Lane on
SYNC_CLK_SEL	1'b0	Select clock source for HS lane output data: ● 0: select fabric input clock rx_clk_1x ● 1: select output clock RX_CLK_O NOTE! Usually selects 0.

3.4 Typical Timing

Figure 3-2 D-PHY HS Burst Data Transmission

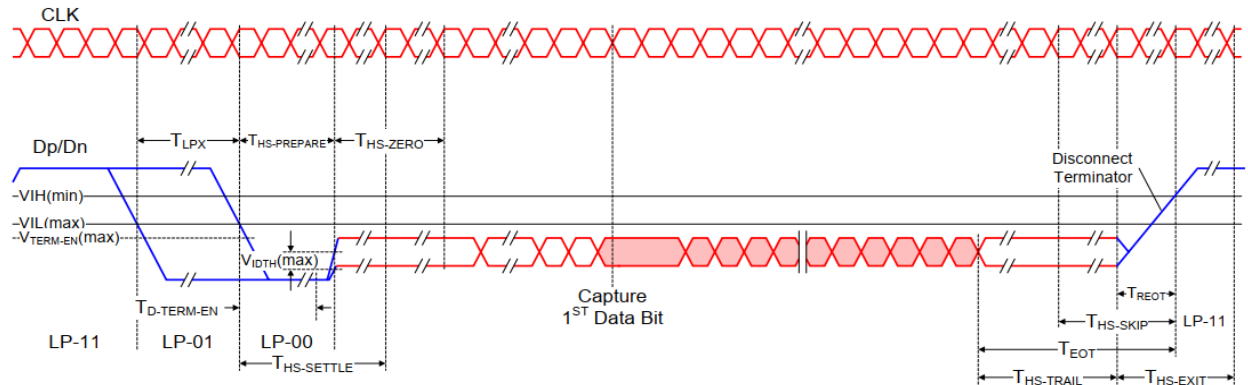


Figure 3-2, referenced from the MIPI Alliance Specification for D-PHY, Version 1.2 August 2014 (<http://www.mipi.org>), shows the sequence of events during an HS burst data transmission.

Figure 3-3 MIPI D-PHY RX Typical Timing

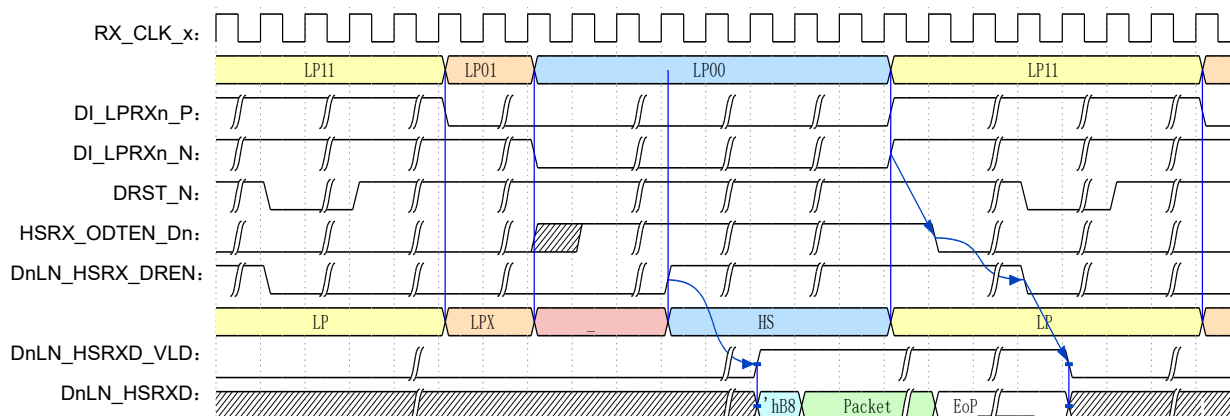


Figure 3-3 shows the typical MIPI D-PHY RX control timing.

- After detecting that the LP state changes from '11' to '01' and then to '00', set HSRX_ODTEN_Dn to '1' to enable the termination resistor. Ensure that the termination is turned on within the $T_{HS-Prepare}$ interval shown in Figure 3-2.
- After a delay, set DnLN_HSRX_DREN to '1' to enable the HSRX driver and thus begin high-speed data reception. Ensure that the HSRX driver is enabled (i.e., the rising edge of DnLN_HSRX_DREN) within the $T_{HS-ZERO}$ interval shown in Figure 3-2, satisfying the following conditions: $T_{HS-Prepare} < T_{HS-SETTLE} < (T_{HS-Prepare} + T_{HS-ZERO})$.
- After detecting that the LP state changes from '00' to '11', set HSRX_ODTEN_Dn to '0' to disable the termination resistor.
- Due to the delay in data reception, in some cases, after setting HSRX_ODTEN_Dn to '0', it is necessary to wait a certain period of time before setting DnLN_HSRX_DREN to '0' to disable the HSRX driver. The exact duration depends on the $T_{HS-TRAIL}$ duration of the transmitter and needs to be adjusted based on actual conditions.
- Setting DRST_N to '0' clears the high-speed data buffer inside the hard core, preventing any residual data in the buffer from interfering with the next reception. A reset can be performed via DRST_N as long as DnLN_HSRX_DREN is '0' and the LP state is not '00'.

3.5 Primitive Instantiation

Verilog Instantiation:

```
MIPI_DPHY_RX mipi_dphy_rx_inst (
    .D0LN_HSRXD(d0ln_hsrxd),
    .D1LN_HSRXD(d1ln_hsrxd),
    .D2LN_HSRXD(d2ln_hsrxd),
    .D3LN_HSRXD(d3ln_hsrxd),
```

```

.D0LN_HSRXD_VLD(d0ln_hsrxd_vld),
.D1LN_HSRXD_VLD(d1ln_hsrxd_vld),
.D2LN_HSRXD_VLD(d2ln_hsrxd_vld),
.D3LN_HSRXD_VLD(d3ln_hsrxd_vld),
.DI_LPRX0_N(di_lprx0_n),
.DI_LPRX0_P(di_lprx0_p),
.DI_LPRX1_N(di_lprx1_n),
.DI_LPRX1_P(di_lprx1_p),
.DI_LPRX2_N(di_lprx2_n),
.DI_LPRX2_P(di_lprx2_p),
.DI_LPRX3_N(di_lprx3_n),
.DI_LPRX3_P(di_lprx3_p),
.DI_LPRXCK_N(di_lprxck_n),
.DI_LPRXCK_P(di_lprxck_p),
.RX_CLK_O(rx_clk_o),
.CK_N(ck_n),
.CK_P(ck_p),
.RX0_N(rx0_n),
.RX0_P(rx0_p),
.RX1_N(rx1_n),
.RX1_P(rx1_p),
.RX2_N(rx2_n),
.RX2_P(rx2_p),
.RX3_N(rx3_n),
.RX3_P(rx3_p),
.LPRX_EN_CK(lprx_en_ck),
.LPRX_EN_D0(lprx_en_d0),
.LPRX_EN_D1(lprx_en_d1),
.LPRX_EN_D2(lprx_en_d2),
.LPRX_EN_D3(lprx_en_d3),
.HSRX_ODTEN_CK(hsrx_odten_ck),
.HSRX_ODTEN_D0(hsrx_odten_d0),
.HSRX_ODTEN_D1(hsrx_odten_d1),
.HSRX_ODTEN_D2(hsrx_odten_d2),
.HSRX_ODTEN_D3(hsrx_odten_d3),
.D0LN_HSRX_DREN(d0ln_hsrx_dren),
.D1LN_HSRX_DREN(d1ln_hsrx_dren),
.D2LN_HSRX_DREN(d2ln_hsrx_dren),
.D3LN_HSRX_DREN(d3ln_hsrx_dren),
.HSRX_EN_CK(hsrx_en_ck),
.HS_8BIT_MODE(hs_8bit_mode),
.RX_CLK_1X(rx_clk_1x),
.RX_INVERT(rx_invert),
.LALIGN_EN(lalign_en),

```

```

        .WALIGN_BY(walign_by),
        .DO_LPTX0_N(do_lptx0_n),
        .DO_LPTX0_P(do_lptx0_p),
        .DO_LPTX1_N(do_lptx1_n),
        .DO_LPTX1_P(do_lptx1_p),
        .DO_LPTX2_N(do_lptx2_n),
        .DO_LPTX2_P(do_lptx2_p),
        .DO_LPTX3_N(do_lptx3_n),
        .DO_LPTX3_P(do_lptx3_p),
        .DO_LPTXCK_N(do_lptxck_n),
        .DO_LPTXCK_P(do_lptxck_p),
        .LPTX_EN_CK(lptx_en_ck),
        .LPTX_EN_D0(lptx_en_d0),
        .LPTX_EN_D1(lptx_en_d1),
        .LPTX_EN_D2(lptx_en_d2),
        .LPTX_EN_D3(lptx_en_d3),
        .BYTE_LENDIAN(byte_lendian),
        .HSRX_STOP(hsrx_stop),
        .LPRX_ULP_LN0(lprx_ulp_ln0),
        .LPRX_ULP_LN1(lprx_ulp_ln1),
        .LPRX_ULP_LN2(lprx_ulp_ln2),
        .LPRX_ULP_LN3(lprx_ulp_ln3),
        .LPRX_ULP_CK(lprx_ulp_ck),
        .PWRON(pwron),
        .RESET(reset),
        .DESKEW_LNSEL(deskew_ltsel),
        .DESKEW_MTH(deskew_mth),
        .DESKEW_OWVAL(deskew_owval),
        .DESKEW_REQ(deskew_req),
        .DRST_N(drst_n),
        .ONE_BYTE0_MATCH(one_byte0_match),
        .WORD_LENDIAN(word_lendian),
        .FIFO_RD_STD(fifo_rd_std)
    );

    defparam mipi_dphy_rx_inst.MIPI_LANE0_EN = 1'b1;
    defparam mipi_dphy_rx_inst.MIPI_LANE1_EN = 1'b1;
    defparam mipi_dphy_rx_inst.MIPI_LANE2_EN = 1'b1;
    defparam mipi_dphy_rx_inst.MIPI_LANE3_EN = 1'b1;
    defparam mipi_dphy_rx_inst.MIPI_CK_EN = 1'b1;
    defparam mipi_dphy_rx_inst.SYNC_CLK_SEL = 1'b0;
    defparam mipi_dphy_rx_inst.ALIGN_BYTE = 8'b10111000;

```

Vhdl Instantiation:

```

COMPONENT MIPI_DPHY_RX
  GENERIC (
    MIPI_LANE0_EN: in bit := '0';
    MIPI_LANE1_EN: in bit := '0';
    MIPI_LANE2_EN: in bit := '0';
    MIPI_LANE3_EN: in bit := '0';
    MIPI_CK_EN: in bit := '1';
    SYNC_CLK_SEL: in bit := '1';
    ALIGN_BYTE: in bit_vector := "10111000"
  );
  PORT(
    D0LN_HSRXD: out std_logic_vector(15 downto 0);
    D1LN_HSRXD: out std_logic_vector(15 downto 0);
    D2LN_HSRXD: out std_logic_vector(15 downto 0);
    D3LN_HSRXD: out std_logic_vector(15 downto 0);
    D0LN_HSRXD_VLD: out std_logic;
    D1LN_HSRXD_VLD: out std_logic;
    D2LN_HSRXD_VLD: out std_logic;
    D3LN_HSRXD_VLD: out std_logic;
    DI_LPRX0_N: out std_logic;
    DI_LPRX0_P: out std_logic;
    DI_LPRX1_N: out std_logic;
    DI_LPRX1_P: out std_logic;
    DI_LPRX2_N: out std_logic;
    DI_LPRX2_P: out std_logic;
    DI_LPRX3_N: out std_logic;
    DI_LPRX3_P: out std_logic;
    DI_LPRXCK_N: out std_logic;
    DI_LPRXCK_P: out std_logic;
    RX_CLK_O: out std_logic;
    CK_N: inout std_logic;
    CK_P: inout std_logic;
    RX0_N: inout std_logic;
    RX0_P: inout std_logic;
    RX1_N: inout std_logic;
    RX1_P: inout std_logic;
    RX2_N: inout std_logic;
    RX2_P: inout std_logic;
    RX3_N: inout std_logic;
    RX3_P: inout std_logic;
    LPRX_EN_CK: in std_logic;
    LPRX_EN_D0: in std_logic;
    LPRX_EN_D1: in std_logic;
  );

```

```
LPRX_EN_D2: in std_logic;
LPRX_EN_D3: in std_logic;
HSRX_ODTEN_CK: in std_logic;
HSRX_ODTEN_D0: in std_logic;
HSRX_ODTEN_D1: in std_logic;
HSRX_ODTEN_D2: in std_logic;
HSRX_ODTEN_D3: in std_logic;
D0LN_HSRX_DREN: in std_logic;
D1LN_HSRX_DREN: in std_logic;
D2LN_HSRX_DREN: in std_logic;
D3LN_HSRX_DREN: in std_logic;
HSRX_EN_CK: in std_logic;
HS_8BIT_MODE: in std_logic;
RX_CLK_1X: in std_logic;
RX_INVERT: in std_logic;
LALIGN_EN: in std_logic;
WALIGN_BY: in std_logic;
DO_LPTX0_N: in std_logic;
DO_LPTX0_P: in std_logic;
DO_LPTX1_N: in std_logic;
DO_LPTX1_P: in std_logic;
DO_LPTX2_N: in std_logic;
DO_LPTX2_P: in std_logic;
DO_LPTX3_N: in std_logic;
DO_LPTX3_P: in std_logic;
DO_LPTXCK_N: in std_logic;
DO_LPTXCK_P: in std_logic;
LPTX_EN_CK: in std_logic;
LPTX_EN_D0: in std_logic;
LPTX_EN_D1: in std_logic;
LPTX_EN_D2: in std_logic;
LPTX_EN_D3: in std_logic;
BYTE_LENDIAN: in std_logic;
HSRX_STOP: in std_logic;
LPRX_ULP_LN0: in std_logic;
LPRX_ULP_LN1: in std_logic;
LPRX_ULP_LN2: in std_logic;
LPRX_ULP_LN3: in std_logic;
LPRX_ULP_CK: in std_logic;
PWRON: in std_logic;
RESET: in std_logic;
DESKEW_LNSEL: in std_logic_vector(2 downto 0);
DESKEW_MTH: in std_logic_vector(7 downto 0);
DESKEW_OWVAL: in std_logic_vector(6 downto 0);
```

```

        DESKEW_REQ: in std_logic;
        DRST_N: in std_logic;
        ONE_BYTE0_MATCH: in std_logic;
        WORD_LENDIAN: in std_logic;
        FIFO_RD_STD: in std_logic_vector(2 downto 0)
    );
END COMPONENT;
mipi_dphy_rx_inst: MIPI_DPHY_RX
    GENERIC MAP(
        MIPI_LANE0_EN => '1',
        MIPI_LANE1_EN => '1',
        MIPI_LANE2_EN => '1',
        MIPI_LANE3_EN => '1',
        MIPI_CK_EN => '1',
        SYNC_CLK_SEL => '0',
        ALIGN_BYTE => "10111000"
    )
    PORT MAP (
        D0LN_HSRXD => d0ln_hsrxd,
        D1LN_HSRXD => d1ln_hsrxd,
        D2LN_HSRXD => d2ln_hsrxd,
        D3LN_HSRXD => d3ln_hsrxd,
        D0LN_HSRXD_VLD => d0ln_hsrxd_vld,
        D1LN_HSRXD_VLD => d1ln_hsrxd_vld,
        D2LN_HSRXD_VLD => d2ln_hsrxd_vld,
        D3LN_HSRXD_VLD => d3ln_hsrxd_vld,
        DI_LPRX0_N => di_lprx0_n,
        DI_LPRX0_P => di_lprx0_p,
        DI_LPRX1_N => di_lprx1_n,
        DI_LPRX1_P => di_lprx1_p,
        DI_LPRX2_N => di_lprx2_n,
        DI_LPRX2_P => di_lprx2_p,
        DI_LPRX3_N => di_lprx3_n,
        DI_LPRX3_P => di_lprx3_p,
        DI_LPRXCK_N => di_lprxck_n,
        DI_LPRXCK_P => di_lprxck_p,
        RX_CLK_O => rx_clk_o,
        CK_N => ck_n,
        CK_P => ck_p,
        RX0_N => rx0_n,
        RX0_P => rx0_p,
        RX1_N => rx1_n,
        RX1_P => rx1_p,
        RX2_N => rx2_n,

```

```
RX2_P => rx2_p,  
RX3_N => rx3_n,  
RX3_P => rx3_p,  
LPRX_EN_CK => lprx_en_ck,  
LPRX_EN_D0 => lprx_en_d0,  
LPRX_EN_D1 => lprx_en_d1,  
LPRX_EN_D2 => lprx_en_d2,  
LPRX_EN_D3 => lprx_en_d3,  
HSRX_ODTEN_CK => hsrx_odten_ck,  
HSRX_ODTEN_D0 => hsrx_odten_d0,  
HSRX_ODTEN_D1 => hsrx_odten_d1,  
HSRX_ODTEN_D2 => hsrx_odten_d2,  
HSRX_ODTEN_D3 => hsrx_odten_d3,  
D0LN_HSRX_DREN => d0ln_hsrx_dren,  
D1LN_HSRX_DREN => d1ln_hsrx_dren,  
D2LN_HSRX_DREN => d2ln_hsrx_dren,  
D3LN_HSRX_DREN => d3ln_hsrx_dren,  
HSRX_EN_CK => hsrx_en_ck,  
HS_8BIT_MODE => hs_8bit_mode,  
RX_CLK_1X => rx_clk_1x,  
RX_INVERT => rx_invert,  
LALIGN_EN => lalign_en,  
WALIGN_BY => walign_by,  
DO_LPTX0_N => do_lptx0_n,  
DO_LPTX0_P => do_lptx0_p,  
DO_LPTX1_N => do_lptx1_n,  
DO_LPTX1_P => do_lptx1_p,  
DO_LPTX2_N => do_lptx2_n,  
DO_LPTX2_P => do_lptx2_p,  
DO_LPTX3_N => do_lptx3_n,  
DO_LPTX3_P => do_lptx3_p,  
DO_LPTXCK_N => do_lptxck_n,  
DO_LPTXCK_P => do_lptxck_p,  
LPTX_EN_CK => lptx_en_ck,  
LPTX_EN_D0 => lptx_en_d0,  
LPTX_EN_D1 => lptx_en_d1,  
LPTX_EN_D2 => lptx_en_d2,  
LPTX_EN_D3 => lptx_en_d3,  
BYTE_LENDIAN => byte_lendian,  
HSRX_STOP => hsrx_stop,  
LPRX_ULP_LN0 => lprx_ulp_ln0,  
LPRX_ULP_LN1 => lprx_ulp_ln1,  
LPRX_ULP_LN2 => lprx_ulp_ln2,  
LPRX_ULP_LN3 => lprx_ulp_ln3,
```

```
LPRX_ULP_CK => lprx_ulp_ck,  
PWRON => pwron,  
RESET => reset,  
DESKEW_LNSEL => deskew_ltsel,  
DESKEW_MTH => deskew_mth,  
DESKEW_OWVAL => deskew_owval,  
DESKEW_REQ => deskew_req,  
DRST_N => drst_n,  
ONE_BYTE0_MATCH => one_byte0_match,  
WORD_LENDIAN => word_lendian,  
FIFO_RD_STD => fifo_rd_std  
);
```

4 MIPI D-PHY RX Configuration and Generation

Start "IP Core Generator" from the "Tools" menu in the Gowin software and then complete the call and configure MIPI D-PHY RX.

4.1 MIPI D-PHY RX Configuration

The configuration options for MIPI D-PHY RX are shown in Figure 4-1.

Figure 4-1 MIPI D-PHY RX Configuration

MIPI_DPHY_RX

File

Device: GW1N-2 Part Number: GW1N-LV2CS42C7/I6

Create In: C:\Users\tongtong\Documents\gowin_mipi_dphy_rx

File Name: gowin_mipi_dphy_rx Module Name: Gowin_MIPI_DPHY_RX

Language: Verilog

Common

☒ Clock Lane Enable

☐ Lane 0 Enable

☐ Lane 1 Enable

☐ Lane 2 Enable

☐ Lane 3 Enable

OK Cancel

1. File Configuration Box

The File Configuration box is used to configure the information about the generated IP design file.

- **Device:** Displays information about the configured Device.
- **Part Number:** Display the configured Part Number.
- **Language:** Hardware description language used to generate the IP design files. Click the drop-down list to select the language, including Verilog and VHDL .
- **Module Name:** The module name of the generated IP design files. Enter the module name in the text box. Module name cannot be the same as the primitive name. If it is the same, an error will be reported.
- **File Name:** The name of the generated IP design files. Enter the file name in the text box.
- **Create In:** The path in which the generated IP files will be stored. Enter the target path in the box or select the target path by clicking the option.

2. Common

- Clock Lane Enable: Disable or enable clock lane.
- Lane 0 Enable: Disable or enable lane 0.
- Lane 1 Enable: Disable or enable lane 1.
- Lane 2 Enable: Disable or enable lane 2.
- Lane 3 Enable: Disable or enable lane 3.

3. Port Diagram

The port diagram displays a sample diagram of the IP Core configuration, as shown in Figure 4-1.

4. Help

Click "Help" to open the IP Core configuration information. The Help page includes an overview of the IP Core and a brief description of the Options.

4.2 MIPI D-PHY RX Generation File

After configuration, it will generate three files that are named after the "File Name".

- "gowin_mipi_dphy_rx.v" file is a complete Verilog module to generate an instantiated MIPI_DPHY_RX according to the IP configuration.
- "gowin_mipi_dphy_rx_tmp.v" is the template file.
- "gowin_mipi_dphy_rx.ipc" file is IP configuration file. You can load the file to configure the IP.

Note!

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.

Appendix **A** MIPI D-PHY Data Rates

Table A-1 MIPI D-PHY Data Rates

Resolution	Frame Rate (HZ)	Bits Per Pixel (Bits)	Total Data Rate (Mbps)	Lane Number	Per Lane Bit Rate (Mbps)	Recommended Gearing Ratio (1:N)	Per Lane Fabric Clock (MHz)
FHD 1920x1080p (2200x1125)	60	8	1188	2	594.0	8	74.25
		10	1485	2	742.5	8	92.81
		16	2376	2	1188.0	16	74.25
		18	2673	4	668.3	8	83.53
		24	3564	4	891.0	8	111.38
	120	8	2376	2	1188.0	16	74.25
		10	2970	4	742.5	8	92.81
		16	4752	4	1188.0	16	74.25
		18	5346	8	668.3	16	41.77
		24	7128	8	891.0	8	111.38
UHD 3840x2160p (4400x2250)	30	8	2376	4	594.0	8	74.25
		10	2970	4	742.5	8	92.81
		16	4752	4	1188.0	16	74.25
		18	5346	8	668.3	16	41.77
		24	7128	8	891.0	8	111.38
	60	8	4752	4	1188.0	16	74.25
		10	5940	8	742.5	16	46.41
		16	9504	8	1188.0	16	74.25
		18	10692	8	1336.5	16	83.53

