



Gowin GW1N-2 Hardened MIPI D-PHY RX

ユーザーガイド

IPUG778-1.2J, 2025-09-26

著作権について(2025)

著作権に関する全ての権利は、**Guangdong Gowin Semiconductor Corporation** に留保されています。

 **GOWIN** 及び **Gowin** は、当社により、中国、米国特許商標庁、及びその他の国において登録されています。商標又はサービスマークとして特定されたその他全ての文字やロゴは、それぞれの権利者に帰属しています。何れの団体及び個人も、当社の書面による許可を得ず、本文書の内容の一部もしくは全部を、いかなる視聴覚的、電子的、機械的、複写、録音等の手段によりもしくは形式により、伝搬又は複製をしてはなりません。

免責事項

当社は、GOWINSEMI Terms and Conditions of Sale (GOWINSEMI取引条件) に規定されている内容を除き、(明示的か又は黙示的に拘わらず) いかなる保証もせず、また、知的財産権や材料の使用によりあなたのハードウェア、ソフトウェア、データ、又は財産が被った損害についても責任を負いません。当社は、事前の通知なく、いつでも本文書の内容を変更することができます。本文書を参照する何れの団体及び個人も、最新の文書やエラッタ (不具合情報) については、当社に問い合わせる必要があります。

バージョン履歴

日付	バージョン	説明
2021/06/01	1.0J	初版。
2021/07/07	1.01J	付録 A MIPI D-PHY データレート表を追加。
2022/02/15	1.1J	● 表 3-1 の注記を追加。 ● GUI から Select clock source for HS lane output data オプションを削除。
2025/09/26	1.2J	● 「表 3-1 MIPI_DPHY_RX のポートの説明」を最適化。 ● 「3.4 典型的なタイミング」を追加。

目次

目次	i
図一覧	ii
表一覧	iii
1 本マニュアルについて	1
1.1 マニュアルの内容	1
1.2 関連ドキュメント	1
1.3 用語、略語	2
1.4 テクニカル・サポートとフィードバック	2
2 概要	3
2.1 特徴	3
2.2 機能の説明	4
3 MIPI D-PHY RX プリミティブ	5
3.1 ポート図	5
3.2 ポートの説明	6
3.3 パラメータの説明	9
3.4 典型的なタイミング	9
3.5 プリミティブのインスタンス化	10
4 MIPI D-PHY RX の呼び出しと構成	18
4.1 MIPI D-PHY RX の構成	18
4.2 MIPI D-PHY RX の関連ファイル	19
付録 A MIPI D-PHY のデータレート表	21

図一覧

図 2-1 MIPI D-PHY RX の構造	4
図 3-1 MIPI_DPHY_RX のポート図	5
図 3-2 D-PHY HS バーストデータ転送	9
図 3-3 MIPI D-PHY RX の典型的なタイミング	10
図 4-1 MIPI D-PHY RX の構成画面	18

表一覧

表 1-1 用語、略語	2
表 3-1 MIPI_DPHY_RX のポートの説明.....	6
表 3-2 MIPI_DPHY_RX のパラメータの説明	9
表 A-1 MIPI D-PHY のデータレート	22

1 本マニュアルについて

1.1 マニュアルの内容

本マニュアルは、ユーザーが Gowin GW1N-2 Hardened MIPI D-PHY RX を使いこなせるよう、その機能、ポート、呼び出しと構成などについて説明しています。本マニュアルは、ユーザーが Gowin GW1N-2 Hardened MIPI D-PHY RX を使いこなせるように作成されています。

1.2 関連ドキュメント

1. GOWIN セミコンダクターのホームページから、以下の関連ドキュメントがダウンロード、参考できます：
 - GW1N シリーズ FPGA 製品データシート([DS100](#))
 - GW1N シリーズ FPGA 製品パッケージ及びピンアウト ユーザーガイド([UG103](#))
 - GW1N-2 デバイス Pinout([UG171](#))
2. MIPI Alliance Specification for D-PHY, Version 1.2 August 2014, www.mipi.org

1.3 用語、略語

本マニュアルで使用される用語、略語、及びその意味を表 1-1 に示します。

表 1-1 用語、略語

用語、略語	正式名称	意味
CSI	Camera Serial Interface	カメラ・シリアル・インターフェース
DSI	Display Serial Interface	ディスプレイ・シリアル・インターフェース
HS	High Speed	高速
I/O	Input/Output	入力/出力
LP	Low Power	低消費電力
MIPI	Mobile Industry Processor Interface	モバイル産業用プロセッサ・インターフェース
ODT	On-Die Termination	オンダイ終端抵抗

1.4 テクニカル・サポートとフィードバック

GOWIN セミコンダクターは、包括的な技術サポートをご提供しています。使用に関するご質問、ご意見については、直接弊社までお問い合わせください。

ホームページ : www.gowinsemi.com/ja

E-mail : support@gowinsemi.com

2 概要

GW1N-2 デバイスは、『MIPI Alliance Standard for D-PHY Specification』バージョン 1.2 をサポートするハード MIPI D-PHY RX を備えています。このハード MIPI D-PHY は、DSI インターフェースおよび CSI-2 インターフェースに適しています。

2.1 特徴

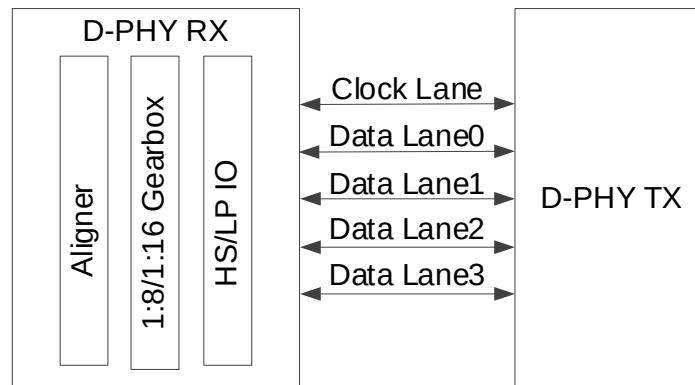
MIPI D-PHY RX の主な特徴は以下に示す通りです：

- 単方向の高速(HS、High-speed)モードをサポート。データ・レーンあたりの最大転送レートは 2Gbps。
- 最大 4 つのデータ・レーンと 1 つのクロック・レーンをサポート。
- 双方向の低消費電力(LP、Low-power)モードをサポート。データレートは 10Mbps。
- Word Alignment と Lane Alignment を含む高速同期をサポート。
- MIPI D-PHY RX 1:8 モードおよび 1:16 モードをサポート。
- MIPI DSI および MIPI CSI-2 リンク層をサポート。
- MIPI D-PHY RX ハードコア専用の MIPI Bank は Bank6。

2.2 機能の説明

MIPI D-PHY RX には、HS/LP IO、1:8/1:16 Gearbox、および Aligner が含まれています(図 2-1)。

図 2-1 MIPI D-PHY RX の構造



HS/LP I/O

ODT の動的切り替えがサポートされ、LP TX/RX と HS RX の動的切り替えがサポートされます。

1:8/1:16 Gearbox

8 ビットまたは 16 ビットのモードがサポートされます。
HS_8BIT_MODE パラメータにより設定可能です。

Aligner

Word align(ワードアライン)および Lane align(レーンアライン)がサポートされます。MIPI_DPHY_RX ポートの WALIGN_BY および LALIGN_EN を介して設定可能です。そのうち、Word align のキーは、ユーザー定義 (ALIGN_BYTE)をサポートし、MIPI_DPHY_RX の ONE_BYTE0_MATCH ポートにより 2 バイトまたは 3 バイトモードとして設定することをサポートします。

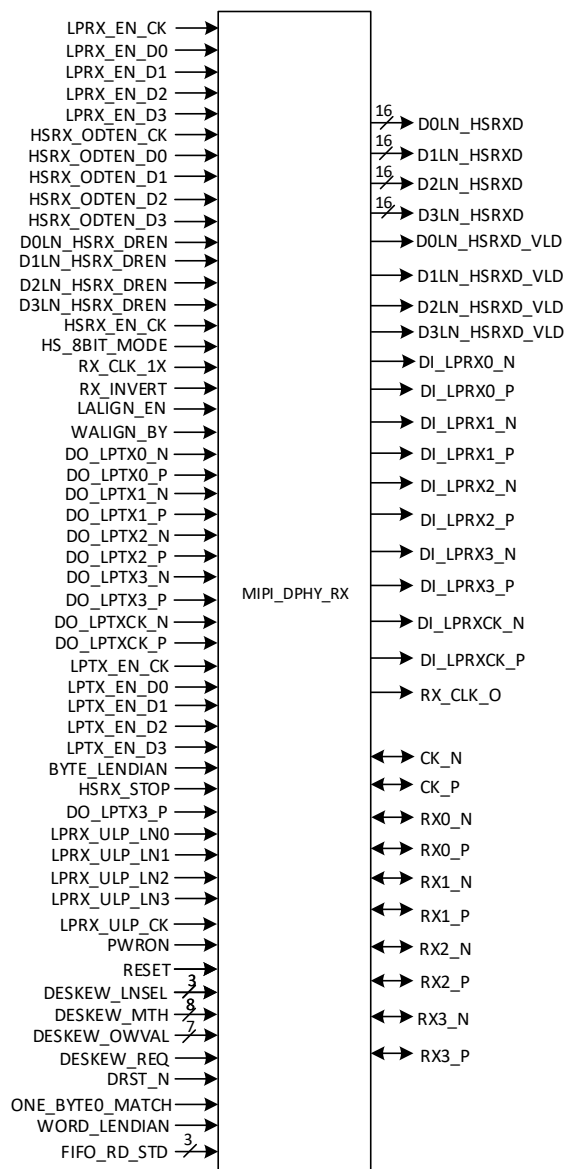
注記：

MIPI_DPHY_RX のポートの説明については、3 MIPI D-PHY RX プリミティブ > ポートの説明を参照してください。

3 MIPI D-PHY RX プリミティブ

3.1 ポート図

図 3-1 MIPI_DPHY_RX のポート図



3.2 ポートの説明

表 3-1 MIPI_DPHY_RX のポートの説明

ポート	I/O	説明
MIPI INTERFACE Signals		
CK_N ^[1]	入出力	CK Lane Complement Input
CK_P ^[1]	入出力	CK Lane True Input
RX<0~3>_N ^[1]	入出力	Data Lane <0~3> Complement Input
RX<0~3>_P ^[1]	入出力	Data Lane <0~3> True Input
CONFIG Signals		
BYTE_LENDIAN	入力	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]	入力	FIFO Read Threshold. Can only be 1 in 8-bits mode.
HS_8BIT_MODE	入力	Selection of Data Width to Fabric ● 1'b1: 8bit output mode ● 1'b0: 16bit output mode
LALIGN_EN	入力	Lane Aligner Enable ● 1'b1: lane aligner is enabled ● 1'b0: lane aligner is disabled
ONE_BYTE0_MATCH	入力	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	入力	Data Polarity Selection
WALIGN_BY	入力	Word aligner bypass ● 1'b1: bypass ● 1'b0: word aligner is enabled
WORD_LENDIAN	入力	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}; 1'b0: Big endian, first word in high byte of 16bit. Lane aligner is searching: {KEY, 8'hx};
RESET and CLOCK Signals		
DRST_N	入力	Digital reset, active low
HSRX_STOP	入力	HSRX Clock Stop Signal for synchronization

ポート	I/O	説明
PWRON	入力	Power-On Control: ● 1'b1: HSRX on ● 1'b0: HSRX off to standby in low power state
RESET	入力	Reset signal: 1'b1: reset all;
RX_CLK_1X	入力	1X clock from fabric, max 93.75MHz@1.5Gbps
RX_CLK_O	出力	RX output 1X clock, max 93.75MHz@1.5Gbps
HSRX Signals		
D<0~3>LN_HSRX_DREN	入力	Data Lane<0~3> HSRX driver enabled: ● 1'b1 driver enabled ● 1'b0 driver disabled
D<0~3>LN_HSRXD[7/15:0]	出力	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	出力	Data Lane<0~3> HS data output valid to fabric
HSRX_EN_CK	入力	CK Lane HSRX enabled
HSRX_ODTEN_CK	入力	CK Lane HSRX ODT enabled: ● 1'b1 ODT enabled ● 1'b0 ODT disabled
HSRX_ODTEN_D<0~3>	入力	Data Lane<0~3> HSRX ODT enabled: ● 1'b1 ODT enabled ● 1'b0 ODT disabled
LPRX Signals		
DI_LPRX<0~3>_N	出力	Data Lane <0~3> Complement Pad LPRX input
DI_LPRX<0~3>_P	出力	Data Lane <0~3> True Pad LPRX input
DI_LPRXCK_N	出力	CK Lane Complement Pad LPRX input
DI_LPRXCK_P	出力	CK Lane True Pad LPRX input
LPRX_EN_CK	入力	CK Lane LPRX enabled
LPRX_EN_D<0~3>	入力	Data Lane <0~3> LPRX enabled
LPTX Signals		
DO_LPTX<0~3>_N	入力	Data Lane <0~3> Complement Pad LPTX output
DO_LPTX<0~3>_P	入力	Data Lane <0~3> True Pad LPTX output
DO_LPTXCK_N	入力	CK Lane Complement Pad LPTX output
DO_LPTXCK_P	入力	CK Lane True Pad LPTX output
LPTX_EN_CK	入力	CK Lane LPTX enabled
LPTX_EN_D<0~3>	入力	Data Lane <0~3> LPTX enabled
Other Signals		
LPRX_ULP_LN0 ^[2]	入力	Data Lane0 LPRX ULP Control: 1'b0 disable
LPRX_ULP_LN1 ^[2]	入力	Data Lane1 LPRX ULP Control: 1'b0 disable
LPRX_ULP_LN2 ^[2]	入力	Data Lane2 LPRX ULP Control: 1'b0 disable
LPRX_ULP_LN3 ^[2]	入力	Data Lane3 LPRX ULP Control: 1'b0 disable
LPRX_ULP_CK ^[2]	入力	CK Lane LPRX ULP Control: 1'b0 disable

ポート	I/O	説明
DESKEW_LNSEL[2:0]	入力	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]	入力	counter threshold for searching one edge
DESKEW_OWVAL [6:0]	入力	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 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	入力	deskew function request to all data lanes

注記：

- ^[1] MIPI 専用ピンで、ユーザーによる制約は必要ありません。詳細については、『GW1N シリーズ FPGA 製品パッケージ及びピンアウト ユーザーガイド([UG103](#))』および『GW1N-2 デバイス Pinout([UG171](#))』を参照してください。
- ^[2] 通常は、ULP 機能を無効にする「0」にセットされます。

3.3 パラメータの説明

表 3-2 MIPI_DPHY_RX のパラメータの説明

パラメータ	デフォルト	説明
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 典型的なタイミング

図 3-2 D-PHY HS バーストデータ転送

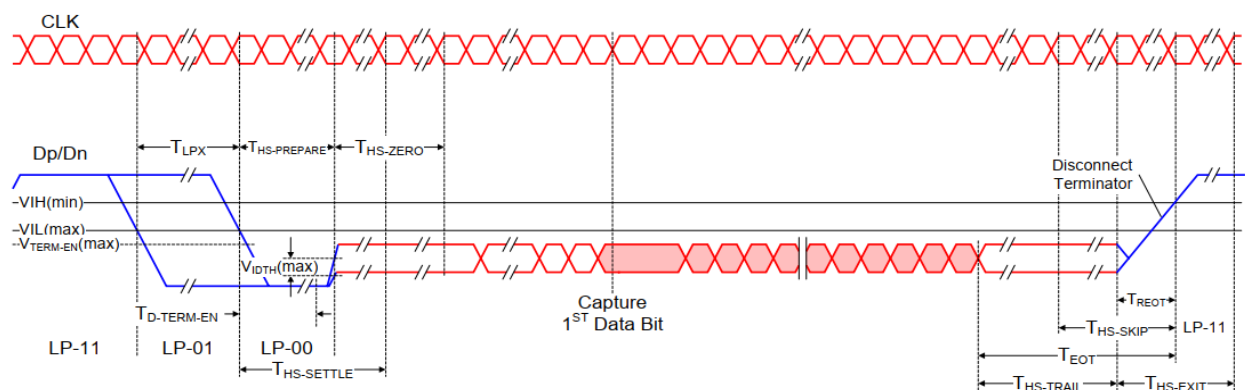


図 3-2 は MIPI Alliance Specification for D-PHY, Version 1.2 August 2014 (<http://www.mipi.org>)より引用したもので、パケットのバースト転送のシーケンスを示しています。

図 3-3 MIPI D-PHY RX の典型的なタイミング

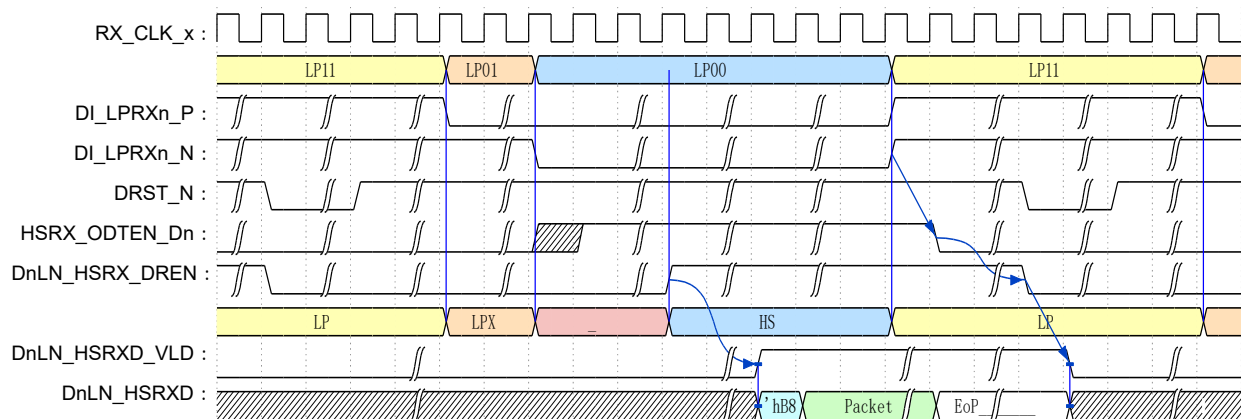


図 3-3 に、MIPI D-PHY RX の典型的な制御タイミングを示します。

- LP 状態が「11」から「01」、そして「00」に変化したことが検出されると、HSRX_ODTEN_Dn を「1」にセットして終端抵抗をオンにします。終端抵抗がオンにされる時点が図 3-2 の $T_{HS-Prepare}$ 期間内にあることを保証する必要があります。
- 遅延後、DnLN_HSRX_DREN を「1」にセットして HSRX driver を有効にし、高速データ受信を開始します。HSRX driver が有効にされる時点（つまり、DnLN_HSRX_DREN の立ち上がりエッジ）が図 3-2 の $T_{HS-ZERO}$ 期間内にあることを保証する必要があります。つまり、 $T_{HS-Prepare} < T_{HS-SETTLE} < (T_{HS-Prepare} + T_{HS-ZERO})$ という条件を満たす必要があります。
- LP 状態が「00」から「11」に戻ることが検出されると、HSRX_ODTEN_Dn を「0」にセットして終端抵抗をオフにします。
- 受信には一定の遅延があるため、HSRX_ODTEN_Dn を「0」にセットした後、HSRX driver を無効にするために DnLN_HSRX_DREN を「0」にセットするまで、一定時間待つ必要があります。この時間は送信側の $T_{HS-TRAIL}$ に依存し、実際の状況に応じて調整する必要があります。
- DRST_N を「0」にセットすると、ハードコア内の高速データバッファがクリアされ、バッファ内の残留データが次の受信に干渉するのを防ぎます。DnLN_HSRX_DREN が「0」で、LP 状態が「00」でない限り、DRST_N を介してリセットを実行できます。

3.5 プリミティブのインスタンス化

Verilog でのインスタンス化：

```
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 でのインスタンス化 :

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  
);
```

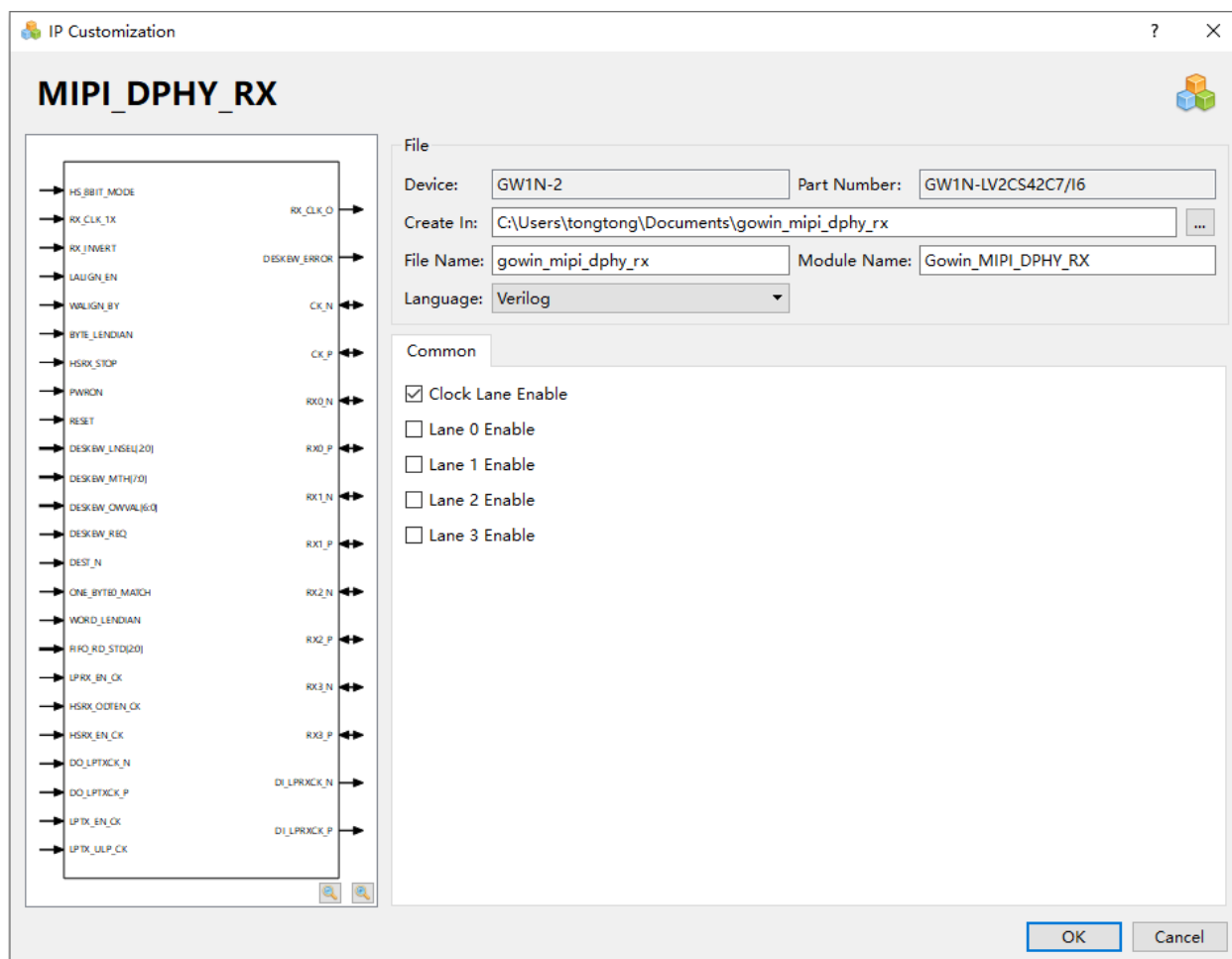
4 MIPI D-PHY RX の呼び出しと構成

Gowin ソフトウェア GUI のメニューバー > Tools > IP Core Generator を通じて IP を呼び出し・構成することができます。

4.1 MIPI D-PHY RX の構成

MIPI D-PHY RX の構成画面は図 4-1 に示すとおりです。

図 4-1 MIPI D-PHY RX の構成画面



1. File 構成タブ

File 構成タブは、生成される IP ファイルの構成に使用されます。

- **Device** : 対象デバイス。
- **Part Number** : 部品番号。
- **Language** : IP を実現するハードウェア記述言語。右側のドロップダウン・リストからターゲット言語(**Verilog** または **VHDL**)を選択します。
- **Module Name** : 生成される IP ファイルのモジュール名。右側のテキストボックスで編集できます。**Module Name** をプリミティブ名と同じにすることはできません。同じである場合、エラーが報告されます。
- **File Name** : 生成される IP ファイルのファイル名。右側のテキストボックスで再編集できます。
- **Create In** : 生成される IP ファイルのパス。右側のテキストボックスでパスを直接編集するか、テキストボックスの右側にある選択ボタンを使用してパスを選択できます。

2. Common

- **Clock Lane Enable** : clock lane を有効にします。
- **Lane 0 Enable** : lane 0 を有効にするかどうか選択します。
- **Lane 1 Enable** : lane 1 を有効にするかどうか選択します。
- **Lane 2 Enable** : lane 2 を有効にするかどうか選択します。
- **Lane 3 Enable** : lane 3 を有効にするかどうか選択します。

3. ポート図

ポート図は、現在 IP Core の構成結果を表示します(図 4-1)。

4. Help ボタン

「Help」をクリックすると、IP Core の構成情報ページが表示されます。Help ページには、IP Core の概要、Options の各構成の簡単な説明があります。

4.2 MIPI D-PHY RX の関連ファイル

MIPI D-PHY RX の構成が完了すると、「File Name」によって命名された 3 つのファイルが生成されます：

- 「gowin_mipi_dphy_rx.v」は完全な verilog モジュールです。
- 「gowin_mipi_dphy_rx_tmp.v」は IP のテンプレートファイルです。
- 「gowin_mipi_dphy_rx.ipc」は IP の構成ファイルです。

注記：

VHDL が設計の言語として選択されている場合、生成される最初の 2 つのファイル名のサフィックスは.vhd になります。

付録 **A** MIPI D-PHY のデータレート表

表 A-1 MIPI D-PHY のデータレート

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

