


# Gowin プログラマブル汎用 IO(GPIO) ユーザーガイド

UG289-2.1.6J, 2024-11-22

## 著作権について(2024)

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高云**、Gowin、Arora、LittleBee、及び GOWINSEMI 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かに拘わらず) 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
2016/05/17	1.05J	初版。
2016/07/15	1.06J	図面を更新。
2016/08/02	1.07J	GW2A シリーズ FPGA 製品のサポートを追加。
2016/10/27	1.08J	GW2AR シリーズ FPGA 製品のサポートを追加。
2017/09/01	1.09J	GW1N-6/9 の特性と GW1NR の情報を更新。
2017/10/12	1.10J	IDES16/OSER16 の関連備考情報を追加。
2017/12/12	1.2J	<ul style="list-style-type: none"> <li>● IDDR/ODDR RESET 信号を削除。</li> <li>● LVDS の説明を更新。</li> <li>● memory 付きの入力/出力の説明を追加。</li> </ul>
2018/04/08	1.3J	第 7 章の図表を更新。
2020/05/14	1.4J	<ul style="list-style-type: none"> <li>● 「3.6 GPIO プリミティブ」を更新。</li> <li>● GW1N-6/GW1NR-6 デバイスの情報を削除。</li> </ul>
2020/08/27	1.5J	<ul style="list-style-type: none"> <li>● マニュアルの構造を最適化。</li> <li>● 第四章「入出力ロジック」と第五章「IP の呼び出し」を追加。</li> </ul>
2021/01/07	1.6J	IODELAYB モジュールの内容を更新。
2021/02/02	1.7J	<ul style="list-style-type: none"> <li>● MIPI_IBUF_HS,MIPI_IBUF_LP の説明を追加。</li> <li>● GW2AN-55C、GW1NR-2 のサポートを追加。</li> </ul>
2021/03/25	1.8J	<ul style="list-style-type: none"> <li>● GW1NZ-2 デバイスの情報を削除。</li> <li>● MIPI_OBUF、MIPI_OBUF_A をサポートするデバイスを更新。</li> </ul>
2021/06/21	1.9J	<ul style="list-style-type: none"> <li>● デバイス(GW1N-2B、GW1N-1P5、GW1N-1P5B、GW1NR-2B、GW2AN-18X、GW2AN-9X)のサポートを追加。</li> <li>● IP 呼び出しの図面を更新。</li> </ul>
2021/10/21	1.9.1J	GPIO 規格の説明を更新。
2021/11/23	1.9.2J	入力ロジックの図面を更新。
2022/01/24	2.0J	<ul style="list-style-type: none"> <li>● 入出力バッファの説明を更新。</li> <li>● コード例のフォーマットを微調整。</li> </ul>
2022/05/30	2.0.1J	終端抵抗の説明を更新。
2022/07/22	2.0.2J	OSER4 の説明を更新。
2022/08/10	2.0.3J	デバイスのバージョン情報を更新。
2022/11/04	2.1J	<ul style="list-style-type: none"> <li>● GW1NS-2、GW1NS-2C、GW1NSE-2C、GW1NSR-2、および GW1NSR-2C デバイスを削除。</li> <li>● 「3.6.14 ELVDS_IBUF_MIPI」を追加。</li> </ul>
2023/01/05	2.1.1J	<ul style="list-style-type: none"> <li>● IP 呼び出しの一部の図面を更新、Device Version オプションを追加。</li> <li>● 差動バッファの構成の情報を更新。</li> </ul>
2023/02/22	2.1.2J	Slew Rate の情報を削除。
2023/04/20	2.1.3J	<ul style="list-style-type: none"> <li>● 「3.3 電源供給の要件」における説明を更新。</li> <li>● 「表 3-10 MIPI_IBUF 対応デバイス」を更新。</li> </ul>

日付	バージョン	説明
2023/05/25	2.1.4J	<ul style="list-style-type: none"> <li>● 「図 5-1 DDR IP の構成ウィンドウ」を更新。</li> <li>● 「3.4.5 エミュレート MLVDS」を追加。</li> </ul>
2023/08/18	2.1.5J	<ul style="list-style-type: none"> <li>● GPIO のデフォルト状態の説明を最適化。</li> <li>● 「表 3-10 MIPI_IBUF 対応デバイス」を更新。</li> <li>● 「表 3-14 MIPI_OBUF_A 対応デバイス(追加)」を更新。</li> </ul>
2024/11/22	2.1.6J	プリミティブ IODELAY と IODELAYB の説明を更新。

# 目次

目次 .....	i
図一覧 .....	iv
表一覧 .....	vi
<b>1 本マニュアルについて .....</b>	<b>1</b>
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
<b>2 GPIO の概要 .....</b>	<b>3</b>
<b>3 入出力バッファ .....</b>	<b>5</b>
3.1 GPIO 規格 .....	5
3.2 GPIO のバンキング・スキーム .....	6
3.3 電源供給の要件 .....	6
3.3.1 LVCMOS バッファの構成 .....	6
3.3.2 差動バッファの構成 .....	7
3.4 エミュレート差動回路終端方式 .....	7
3.4.1 エミュレート LVDS .....	7
3.4.2 エミュレート LVPECL .....	8
3.4.3 エミュレート RSDS .....	8
3.4.4 エミュレート BLVDS .....	8
3.4.5 エミュレート MLVDS .....	9
3.5 GPIO の構成 .....	10
3.5.1 位置 .....	10
3.5.2 レベル規格 .....	10
3.5.3 ドライブ強度 .....	10
3.5.4 プルアップ/プルダウン .....	10
3.5.5 リファレンス電圧 .....	10
3.5.6 ヒステリシス .....	10

3.5.7 オープンドレイン .....	10
3.5.8 シングルエンド終端抵抗 .....	10
3.5.9 差動終端抵抗 .....	11
3.6 GPIO プリミティブ .....	11
3.6.1 IBUF .....	11
3.6.2 OBUF .....	12
3.6.3 TBUF .....	13
3.6.4 IOBUF .....	14
3.6.5 LVDS Input Buffer .....	15
3.6.6 LVDS Output Buffer .....	17
3.6.7 LVDS Tristate Buffer .....	19
3.6.8 LVDS Inout Buffer .....	21
3.6.9 MIPI_IBUF .....	23
3.6.10 MIPI_OBUF .....	26
3.6.11 MIPI_OBUF_A .....	27
3.6.12 I3C_IOBUF .....	29
3.6.13 MIPI_IBUF_HS/MIPI_IBUF_LP .....	31
3.6.14 ELVDS_IBUF_MIPI .....	33
<b>4 入出力ロジック .....</b>	<b>36</b>
4.1 SDR モード .....	37
4.2 DDR モードの入出力ロジック .....	37
4.2.1 IDDR .....	37
4.2.2 IDDRRC .....	41
4.2.3 IDES4 .....	42
4.2.4 IDES8 .....	46
4.2.5 IDES10 .....	49
4.2.6 IVIDEO .....	53
4.2.7 IDES16 .....	56
4.2.8 IDDR_MEM .....	60
4.2.9 IDES4_MEM .....	63
4.2.10 IDES8_MEM .....	67
4.3 DDR モードの出力ロジック .....	71
4.3.1 ODDR .....	71
4.3.2 ODDRC .....	74
4.3.3 OSER4 .....	77
4.3.4 OSER8 .....	81
4.3.5 OSER10 .....	85
4.3.6 OVIDEO .....	88
4.3.7 OSER16 .....	91

---

4.3.8 ODDR_MEM .....	95
4.3.9 OSER4_MEM .....	98
4.3.10 OSER8_MEM .....	103
4.4 遅延モジュール .....	108
4.4.1 IODELAY .....	108
4.4.2 IODELAYC .....	110
4.4.3 IODELAYB .....	113
4.5 サンプリングモジュール .....	117
<b>5 IP の呼び出し .....</b>	<b>120</b>
5.1 IP の構成 .....	120
5.2 生成されるファイル .....	122

## 図一覧

図 2-1 入出力ブロックの構造 .....	4
図 3-1 LVDS25E の外部終端 .....	8
図 3-2 LVPECL の外部終端 .....	8
図 3-3 RSDS の外部終端 .....	8
図 3-4 BLVDS の外部終端 .....	9
図 3-5 MLVDS の外部終端 .....	9
図 3-6 IBUF のポート図 .....	11
図 3-7 OBUF のポート図 .....	12
図 3-8 TBUF のポート図 .....	13
図 3-9 IOBUF のポート図 .....	14
図 3-10 TLVDS_IBUF/ELVDS_IBUF のポート図 .....	16
図 3-11 TLVDS_OBUF/ELVDS_OBUF のポート図 .....	18
図 3-12 TLVDS_TBUF/ELVDS_TBUF のポート図 .....	19
図 3-13 TLVDS_IOBUF/ELVDS_IOBUF のポート図 .....	22
図 3-14 MIPI_IBUF/MIPI_IBUF のポート図 .....	24
図 3-15 MIPI_OBUF のポート図 .....	26
図 3-16 MIPI_OBUF_A のポート図 .....	28
図 3-17 I3C_IOBUF のポート図 .....	30
図 3-18 MIPI_IBUF_HS/MIPI_IBUF_LP のポート図 .....	31
図 3-19 ELVDS_IBUF_MIPI のポート図 .....	34
図 4-1 入出力ロジックの出力部の説明図 .....	36
図 4-2 入出力ロジックの入力部の説明図 .....	37
図 4-3 IDDR のブロック図 .....	38
図 4-4 IDDR のタイミング図 .....	38
図 4-5 IDDR のポート図 .....	38
図 4-6 IDDR のポート図 .....	41
図 4-7 CALIB のタイミングの例 .....	43
図 4-8 IDES4 のポート図 .....	43
図 4-9 IDES8 のポート図 .....	46



図 4-10 IDES10 のポート図 .....	49
図 4-11 IVIDEO のポート図 .....	53
図 4-12 IDES16 のポート図 .....	57
図 4-13 IDDR_MEM のポート図 .....	61
図 4-14 IDES4_MEM のポート図 .....	64
図 4-15 IDES8_MEM のポート図 .....	68
図 4-16 ODDR のブロック図 .....	71
図 4-17 ODDR のタイミング図 .....	72
図 4-18 ODDR のポート図 .....	72
図 4-19 ODDRC のブロック図 .....	74
図 4-20 ODDRC のポート図 .....	75
図 4-21 OSER4 のブロック図 .....	77
図 4-22 OSER4 のポート図 .....	78
図 4-23 OSER8 のブロック図 .....	81
図 4-24 OSER8 のポート図 .....	82
図 4-25 OSER10 のポート図 .....	86
図 4-26 OVIDEO のポート図 .....	89
図 4-27 OSER16 のポート図 .....	92
図 4-28 ODDR_MEM のブロック図 .....	95
図 4-29 ODDR_MEM のポート図 .....	96
図 4-30 OSER4_MEM のブロック図 .....	99
図 4-31 OSER4_MEM のポート図 .....	100
図 4-32 OSER8_MEM のブロック図 .....	104
図 4-33 OSER8_MEM のポート図 .....	104
図 4-34 IODELAY のポート図 .....	108
図 4-35 IODELAYC のポート図 .....	111
図 4-36 IODELAYB の構造 .....	114
図 4-37 IODELAYB のポート図 .....	114
図 4-38 IEM のポート図 .....	117
図 5-1 DDR IP の構成ウィンドウ .....	120

# 表一覧

表 1-1 用語、略語.....	2
表 3-1 IBUF のポートの説明 .....	11
表 3-2 OBUF のポートの説明 .....	12
表 3-3 TBUF のポートの説明 .....	13
表 3-4 IOBUF のポートの説明 .....	14
表 3-5 TLVDS_IBUF/ELVDS_IBUF のポートの説明 .....	16
表 3-6 TLVDS_OBUF/ELVDS_OBUF のポートの説明 .....	18
表 3-7 TLVDS_TBUF/ELVDS_TBUF のポートの説明 .....	20
表 3-8 TLVDS_IOBUF 対応デバイス.....	21
表 3-9 TLVDS_IOBUF/ELVDS_IOBUF のポートの説明 .....	22
表 3-10 MIPI_IBUF 対応デバイス .....	23
表 3-11 MIPI_IBUF のポートの説明 .....	24
表 3-12 MIPI_OBUF 対応デバイス .....	26
表 3-13 MIPI_OBUF のポートの説明 .....	26
表 3-14 MIPI_OBUF_A 対応デバイス(追加) .....	28
表 3-15 MIPI_OBUF_A のポートの説明 .....	28
表 3-16 I3C_IOBUF 対応デバイス .....	29
表 3-17 I3C_IOBUF のポート図.....	30
表 3-18 MIPI_IBUF_HS/MIPI_IBUF_LP 対応デバイス .....	31
表 3-19 MIPI_IBUF_HS のポートの説明.....	31
表 3-20 MIPI_IBUF_LP のポートの説明 .....	31
表 3-21 MIPI_IBUF のポートの説明.....	34
表 4-1 IDDR のポートの説明 .....	38
表 4-2 IDDR のパラメータの説明 .....	39
表 4-3 IDDRC のポートの説明 .....	41
表 4-4 IDDRC のパラメータの説明.....	41
表 4-5 IDES4 のポートの説明.....	43
表 4-6 IDES4 のパラメータの説明.....	44
表 4-7 IDES8 のポートの説明.....	46

表 4-8 IDES8 のパラメータの説明.....	47
表 4-9 IDES10 のポートの説明.....	49
表 4-10 IDES10 のパラメータの説明.....	50
表 4-11 IVIDEO のポートの説明 .....	53
表 4-12 IVIDEO のパラメータの説明.....	54
表 4-13 IDES16 対応デバイス .....	56
表 4-14 IDES16 のポートの説明.....	57
表 4-15 IDES16 のパラメータの説明.....	57
表 4-16 IDDR_MEM 対応デバイス.....	60
表 4-17 IDDR_MEM のポートの説明 .....	61
表 4-18 IDDR_MEM のパラメータの説明.....	61
表 4-19 IDES4_MEM 対応デバイス .....	63
表 4-20 IDES4_MEM のポートの説明 .....	64
表 4-21 IDES4_MEM のパラメータの説明 .....	65
表 4-22 IDES8_MEM 対応デバイス .....	67
表 4-23 IDES8_MEM のポートの説明 .....	68
表 4-24 IDES8_MEM のパラメータの説明 .....	68
表 4-25 ODDR のポートの説明.....	72
表 4-26 ODDR のパラメータの説明 .....	72
表 4-27 ODDRC のポートの説明 .....	75
表 4-28 ODDRC のパラメータの説明.....	75
表 4-29 OSER4 のポートの説明 .....	78
表 4-30 OSER4 のパラメータの説明.....	79
表 4-31 OSER8 のポートの説明 .....	82
表 4-32 OSER8 のパラメータの説明.....	82
表 4-33 OSER10 のポートの説明 .....	86
表 4-34 OSER10 のパラメータの説明.....	86
表 4-35 OVIDEO のポートの説明 .....	89
表 4-36 OVIDEO のパラメータの説明 .....	89
表 4-37 OSER16 対応デバイス.....	91
表 4-38 OSER16 のポートの説明 .....	92
表 4-39 OSER16 のパラメータの説明.....	92
表 4-40 ODDR_MEM 対応デバイス .....	95
表 4-41 ODDR_MEM のポートの説明 .....	96
表 4-42 ODDR_MEM のパラメータの説明 .....	96
表 4-43 OSER4_MEM 対応デバイス .....	99
表 4-44 OSER4_MEM のポートの説明.....	100

表 4-45 OSER4_MEM のパラメータの説明 .....	100
表 4-46 OSER8_MEM 対応デバイス .....	103
表 4-47 OSER8_MEM のポートの説明.....	104
表 4-48 OSER8_MEM のパラメータの説明 .....	105
表 4-49 IODELAY のポートの説明 .....	109
表 4-50 IODELAY のパラメータの説明.....	109
表 4-51 IODELAYC 対応デバイス .....	110
表 4-52 IODELAYC のポートの説明 .....	111
表 4-53 IODELAYC のパラメータの説明 .....	111
表 4-54 IODELAYB 対応デバイス .....	113
表 4-55 IODELAYB のポートの説明 .....	114
表 4-56 IODELAYB のパラメータの説明 .....	115
表 4-57 IEM のポートの説明 .....	117
表 4-58 IEM のパラメータの説明.....	118

# 1 本マニュアルについて

## 1.1 マニュアルの内容

このマニュアルは、Gowin セミコンダクターFPGA 製品でサポートされる入出力バッファのレベル規格、バンキング・スキーム、および入出力ロジックの機能について説明します。また、ユーザーが GPIO を使いこなせるよう GPIO の構造と Gowin ソフトウェアの使用法も説明されています。

## 1.2 関連ドキュメント

GOWIN セミコンダクターのホームページ [www.gowinsemi.com/ja](http://www.gowinsemi.com/ja) 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](#))
- GW1NR シリーズ FPGA 製品データシート([DS117](#))
- GW1NS シリーズ FPGA 製品データシート([DS821](#))
- GW1NZ シリーズ FPGA 製品データシート([DS841](#))
- GW1NSR シリーズ FPGA 製品データシート([DS861](#))
- GW1NSE シリーズ安全 FPGA 製品データシート([DS871](#))
- GW1NSER シリーズ安全 FPGA 製品データシート([DS881](#))
- GW1NRF シリーズ Bluetooth FPGA 製品データシート([DS891](#))
- GW2A シリーズ FPGA 製品データシート([DS102](#))
- GW2AR シリーズ FPGA 製品データシート([DS226](#))
- GW2ANR シリーズ FPGA 製品データシート([DS961](#))
- GW2AN-18X & 9X FPGA 製品データシート([DS971](#))
- GW2AN-55 FPGA 製品データシート([DS976](#))

## 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
Bus Keeper	Bus Keeper	バスキーパ(バスホールド・ラッチ)
CFU	Configurable Function Unit	コンフィギュラブル機能ユニット
CRU	Configurable Routing Unit	コンフィギュラブル配線ユニット
DDR	Double Data Rate	ダブル・データ・レート
DES	Deserializer	デシリアライザ
ELDO	Emulated LVDS Output	エミュレートLVDS出力(電圧出力)
GPIO	Gowin Programmable Input/Output	Gowinプログラマブル汎用IO
IO Buffer	Input/Output Buffer	入出力バッファ
IO Logic	Input/Output Logic	入出力ロジック
IOB	Input/Output Block	入出力ブロック
Open Drain	Open Drain	オープンドレイン
SDR	Single Data Rate	シングル・データ・レート
SER	Serializer	シリアライザ
TLDO	True LVDS Output	True LVDS 出力(電流出力)

## 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](http://www.gowinsemi.com/ja)

E-mail : [support@gowinsemi.com](mailto:support@gowinsemi.com)

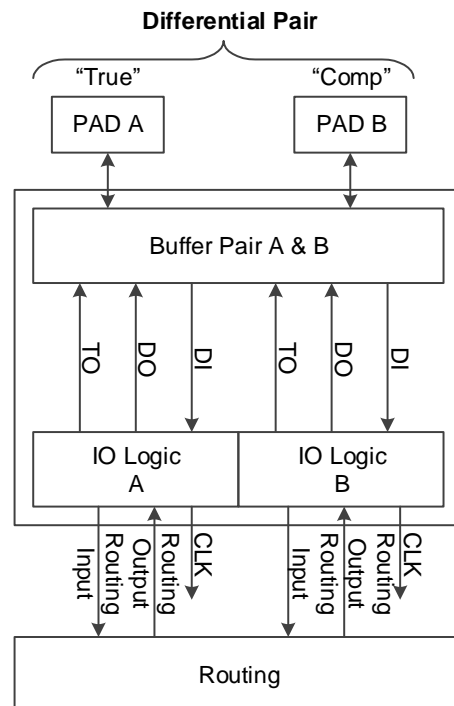
# 2 GPIO の概要

GOWIN セミコンダクターFPGA 製品の GPIO には、シングルエンドレベル規格から差動レベル規格まで、さまざまな外部バス、メモリ、ビデオアプリケーションなどとの接続を容易にするための業界のさまざまなレベル規格をサポートする柔軟性があります。

GOWIN セミコンダクターFPGA 製品の GPIO の基本要素は、入出力バッファ(IO Buffer)、入出力ロジック(IO Logic)、および対応するコンフィギュラブル配線ユニットなどで構成される出力ブロック(IOB)です。そのうちコンフィギュラブル配線ユニットは、コンフィギュラブル機能ユニット(CFU)内のコンフィギュラブル配線ユニット(CRU)と同様です。

図 2-1 示すように、各入出力ブロックには、**A** および **B** とマークされる 2 つの入出力ピンがあります。それらは 1 つの差動信号ペアを構成するか、シングルエンド信号として個別に使用することができます。入出力バッファは、主にさまざまなシングルエンドレベル規格および差動レベル規格のサポートに使用されます。入出力ロジックは、シリアル-パラレル変換、パラレル-シリアル変換、遅延制御、およびバイトアライメントなどの機能を統合し、主に高速データ伝送に使用されます。コンフィギュラブル配線ユニットは、入出力ブロックと他のオンチップリソースとの間の相互接続に使用されます。

図 2-1 入出力ブロックの構造



GOWIN セミコンダクターFPGA シリーズ製品の入出力ブロックの特徴：

- Bank 毎に供給される  $V_{CCIO}$
- LVCMOS、PCI、LVTTL、LVDS、SSTL、及び HSTL 等複数の規格をサポート
- 一部のデバイス<sup>[1]</sup>が MIPI 規格および MIPI I3C OpenDrain/PushPull 変換をサポート
- 入力信号のヒステリシス・オプションを提供
- 出力信号のドライブ強度オプションを提供
- 各ピンに独立したバスホールド、プルアップ/プルダウン抵抗、及びオープンドレイン出力オプションを提供
- ホットプラグをサポート
- 入出力ロジックは、シングル・データ・レート(SDR)、ダブル・データ・レート(DDR)など、複数のモードをサポート

注記：

[1] MIPI、I3C をサポートするデバイスについては、[3.6.9 MIPI IOBUF](#)、[3.6.10 MIPI OBUF](#)、および [3.6.12 I3C IOBUF](#) を参照してください。



# 3 入出力バッファ

## 3.1 GPIO 規格

GOWIN セミコンダクターFPGA 製品は、シングルエンド規格と差動規格をサポートしています。シングルエンド規格では、内部の **Bank** 電圧をリファレンス電圧として使用するか、任意のピンを外部リファレンス電圧入力ピンとして使用することができます。Gowin FPGA のすべてのバンクは差動入力をサポートしています。エミュレート LVDS 差動出力は外部終端抵抗および差動 LVCMOS バッファ出力により実装されます。さらに、True LVDS 差動出力および差動入力終端をサポートする特定のバンクがあります。詳細については、3.2GPIO のバンキング・スキームを参照してください。

GOWIN セミコンダクターFPGA 製品でサポートされている GPIO 規格については、対応するデータシートの「I/O 規格」セクションを参照してください。

- GW1N シリーズ FPGA 製品データシート([DS100](#))
- GW1NR シリーズ FPGA 製品データシート([DS117](#))
- GW1NS シリーズ FPGA 製品データシート([DS821](#))
- GW1NZ シリーズ FPGA 製品データシート([DS841](#))
- GW1NSR シリーズ FPGA 製品データシート([DS861](#))
- GW1NSE シリーズ安全 FPGA 製品データシート([DS871](#))
- GW1NSER シリーズ安全 FPGA 製品データシート([DS881](#))
- GW1NRF シリーズ Bluetooth FPGA 製品データシート([DS891](#))
- GW2A シリーズ FPGA 製品データシート([DS102](#))
- GW2AR シリーズ FPGA 製品データシート([DS226](#))
- GW2ANR シリーズ FPGA 製品データシート([DS961](#))
- GW2AN-18X & 9X FPGA 製品データシート([DS971](#))
- GW2AN-55 FPGA 製品データシート([DS976](#))

## 3.2 GPIO のバンキング・スキーム

GPIO の汎用属性：

- すべての **Bank** はエミュレート LVDS 差動出力をサポートしますが、外部抵抗ネットワークが必要です。
- すべての **Bank** は、プルアップ、プルダウン、およびバスホールド設定をサポートしています。
- 各 **Bank** は 1 つの **Bank** 電圧をサポートします。
- 各 **Bank** は、外部ピンまたは内部リファレンス電圧発生器からの 1 つのリファレンス電圧信号をサポートしています。

## 3.3 電源供給の要件

コア電圧(Vcc)と **Bank** 電圧(Vccio)が特定のしきい値に達すると、内部パワーオンリセット(PoR)が解放され、GOWIN セミコンダクターFPGA 製品のコアロジックがアクティブになります。コンフィギュレーション中、デバイスのすべての **GPIO** は内部の弱いプルアップ<sup>[1]</sup>でハイインピーダンスの状態であり、コンフィギュレーション後、I/O の状態はユーザーデザインおよび制約によって決定されます。コンフィギュレーション関連 I/O の状態は、コンフィギュレーション・モードによって異なります。GOWIN セミコンダクターFPGA 製品には、コア電圧と **Bank** 電圧のパワーアップ/パワーダウン・シーケンス要件はありません。

注記：

[1] GW2AN-18X/9X の場合は内部の弱いプルダウンです。

各バンクは 1 つのリファレンス電圧(VREF)入力をサポートします。**Bank** 内の任意のピンをリファレンス電圧入力ピンとして構成することができます。SSTL や HSTL などの規格をサポートするために、リファレンス電圧は **Bank** 電圧の半分に設定します。また、入力リファレンス電圧は、内部リファレンス電圧発生器によって生成することもできます。各バンクにはリファレンス電圧バスが 1 つしかないため、1 つのバンクの内部リファレンス電圧発生器と外部リファレンス電圧入力ピンを同時に有効にすることはできません。

GOWIN セミコンダクターFPGA 製品の **GPIO** バッファには、**A** および **B** とマークされる 2 つの入出力ピンがあります。ピン **A** は差動信号の T(True、正側)側に対応し、ピン **B** は差動信号の C(Comp、負側)側に対応します。

### 3.3.1 LVCMOS バッファの構成

すべての **GPIO** には、アプリケーションに応じて複数のモードに構成できる LVCMOS バッファが含まれています。各 LVCMOS バッファは、弱いプルアップ、弱いプルダウン、およびバスホールドに構成できます。弱いプルアップおよび弱いプルダウンは、ワイヤード **AND**、ワイヤード **OR** 等の論理制御に幅広く適用できます。バスホールドは最小の電力消費

で信号の前の状態をラッチし、バスホールドをオフにすると入力リーク電流が減少します。

すべての LVCMOS バッファは、プログラマブルなドライブ強度を持っています。各規格のドライブ強度オプションについては、対応するデータシートの「I/O 規格」セクションを参照してください。Gowin FPGA 製品のプログラマブルなドライブ強度は各設定の最低限のドライブ強度を保証するだけです。

ヒステリシスの設定は、主にノイズの多い場合のレベルの急激変動を防ぐために使用され、すべての LVCMOS バッファはヒステリシスの設定をサポートします。

差動ペアが 2 本のシングルエンドピンとして構成されている場合、ピン間の相対遅延が最小になり、信号の一貫性が最もよくなります。

### 3.3.2 差動バッファの構成

GPIO バッファが差動モードに構成された場合、入力ヒステリシスとバスホールド特性は無効になります。

次の GW1N シリーズ製品および GW2A シリーズ製品は、オンチップのプログラマブル 100 Ω 入力差動終端抵抗をサポートします。

- GW1N-4、GW1NR-4、GW1NRF-4B、GW1N-9、GW1NR-9、GW1N-1、GW1NR-1 の Bank0。
- GW1N-1S、GW1NS-4、GW1NS-4C、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW2A-18、GW2A-55、GW2AN-55、GW2ANR-18、GW2AR-18 の Bank0 と Bank1。
- GW1N-2、GW1NR-2、GW1N-1P5 の Bank2。
- GW2AN-18X、GW2AN-9X の Bank4 と Bank5。

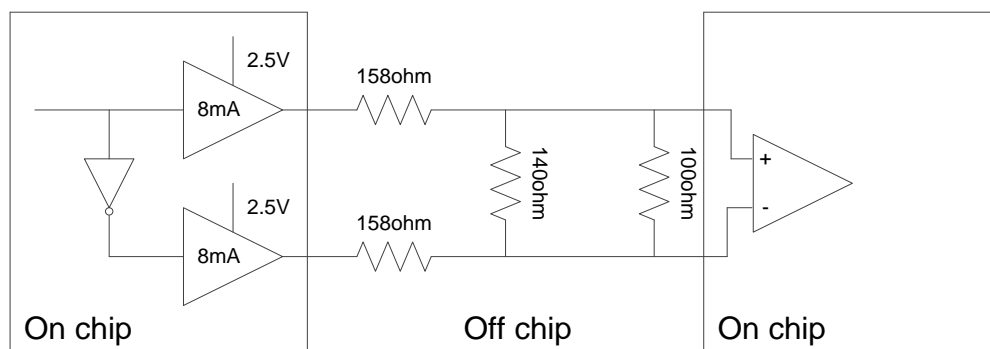
すべてのシングルエンド GPIO バッファペアは、LVPECL33E、MLVDS25E、BLVDS25E などのエミュレート LVDS 差動出力規格に構成できます。この場合、外部終端を追加する必要があります。

## 3.4 エミュレート差動回路終端方式

### 3.4.1 エミュレート LVDS

Gowin FPGA 製品では、コンプリメンタリな LVCMOS 出力と外部終端を利用して、エミュレート LVDS 出力規格を構築することができます。その外部終端方式は下図に示すとおりです。

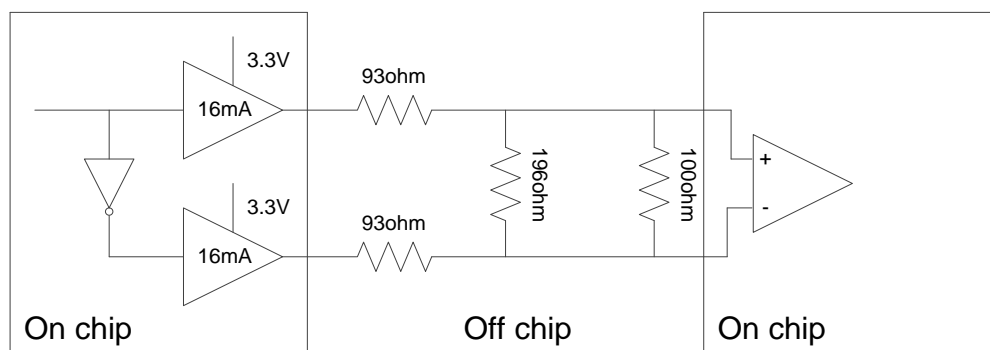
図 3-1 LVDS25E の外部終端



### 3.4.2 エミュレート LVPECL

Gowin FPGA 製品では、コンプリメンタリな LVCMOS 出力と外部終端を利用して、エミュレート LVPECL 出力規格を構築することができます。その外部終端方式は下図に示すとおりです。

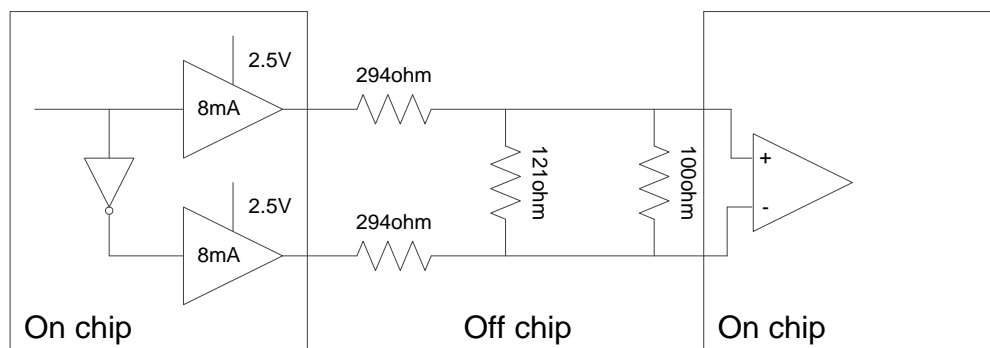
図 3-2 LVPECL の外部終端



### 3.4.3 エミュレート RSDS

Gowin FPGA 製品では、コンプリメンタリな LVCMOS 出力と外部終端を利用して、エミュレート RSDS 出力規格を構築することができます。その外部終端方式は下図に示すとおりです。

図 3-3 RSDS の外部終端

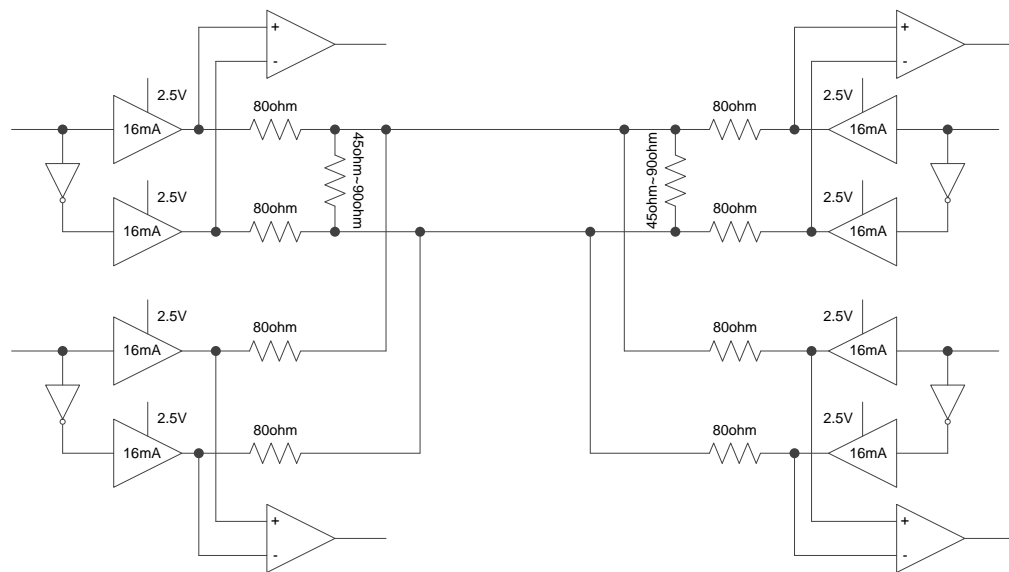


### 3.4.4 エミュレート BLVDS

Gowin FPGA 製品では、コンプリメンタリな LVCMOS 出力と外部終端を利用して、エミュレート BLVDS 出力規格を構築することができます。

す。その外部終端方式は下図に示すとおりです。

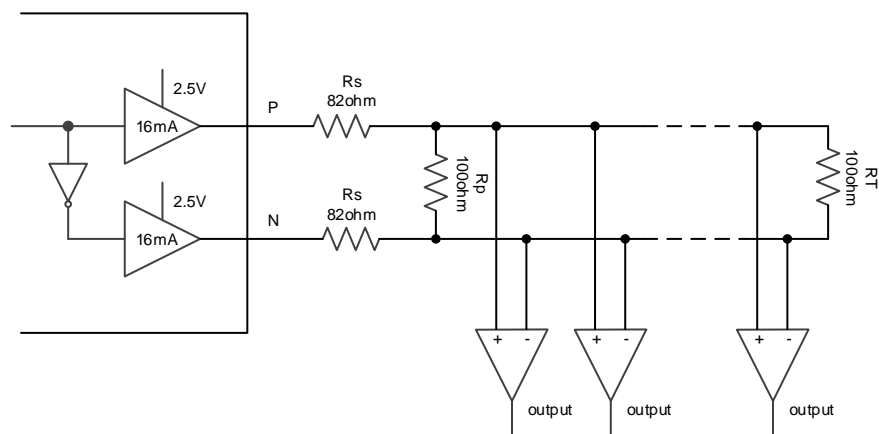
図 3-4 BLVDS の外部終端



### 3.4.5 エミュレート MLVDS

Gowin FPGA 製品では、コンプリメンタリな LVCMOS 出力と外部終端を利用して、エミュレート MLVDS 出力規格を構築することができます。その外部終端方式は下図に示すとおりです。

図 3-5 MLVDS の外部終端



## 3.5 GPIO の構成

Gowin ソフトウェアの Floorplanner を使用して GPIO の位置と属性を設定するか、CST ファイルをカスタマイズしてそれを実現できます。以下は、CST ファイルによりサポートされている物理制約について説明します。

### 3.5.1 位置

GPIO の物理位置をロックします。

```
IO_LOC "xxx" H4 exclusive;
```

### 3.5.2 レベル規格

GPIO の規格を設定します。

```
IO_PORT "xxx" IO_TYPE=LVCMOS18D;
```

### 3.5.3 ドライブ強度

出力ピンまたは双方向ピンのドライブ強度を設定します。

```
IO_PORT "xxx" DRIVE=12;
```

### 3.5.4 プルアップ/プルダウン

プルアップ/プルダウンを設定します。UP : プルアップ。DOWN : プルダウン。KEEPER : バスホールド。NONE : ハイインピーダンス。

```
IO_PORT "xxx" PULL_MODE=DOWN;
```

### 3.5.5 リファレンス電圧

GPIO のリファレンス電圧を設定します。リファレンス電圧は外部ピンまたは内部リファレンス電圧発生器から提供されます。

```
IO_PORT "xxx" VREF=VREF1_LOAD;
```

### 3.5.6 ヒステリシス

入力ピンまたは双方向ピンのためにヒステリシスを設定します。小さい順 : NONE->H2L->L2H->HIGH。

```
IO_PORT "xxx" HYSTERESIS=L2H;
```

### 3.5.7 オープンドレイン

出力ピンまたは双方向ピンのためにオープンドレインを開閉します。ON/OFF オプションが提供されます。

```
IO_PORT "xxx" OPEN_DRAIN=ON;
```

### 3.5.8 シングルエンド終端抵抗

シングルエンド信号のために終端抵抗を設定します。ON/OFF オプションが提供されます。

```
IO_PORT "xxx" SINGLE_RESISTOR=ON;
```

3.5.9 差動終端抵抗

差動信号のために終端抵抗を設定します。ON/OFF オプションが提供されます。

```
IO_PORT "xxx" Diff_RESISTOR=ON;
```

3.6 GPIO プリミティブ

IO Buffer は、機能によって通常のバッファ、エミュレート LVDS(ELVDS)バッファ、および True LVDS(TLVDS)バッファに分類できます。

3.6.1 IBUF

プリミティブの紹介

IBUF(Input Buffer)は入力バッファです。

ポート図

図 3-6 IBUF のポート図



ポートの説明

表 3-1 IBUF のポートの説明

ポート	I/O	説明
I	入力	データ入力信号
O	出力	データ出力信号

プリミティブのインスタンス化

Verilog でのインスタンス化 :

```
IBUF uut(  
    .O(O),  
    .I(I)  
);
```

VHDL でのインスタンス化 :

```
COMPONENT IBUF  
PORT (  
    O:OUT std_logic;
```

```

        I:IN std_logic
    );
END COMPONENT;
uut:IBUF
    PORT MAP(
        O=>O,
        I=>I
    );

```

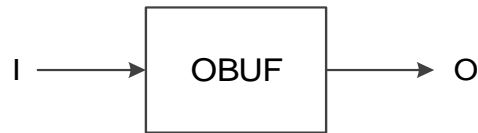
3.6.2 OBUF

プリミティブの紹介

OBUF(Output Buffer)は出力バッファです。

ポート図

図 3-7 OBUF のポート図



ポートの説明

表 3-2 OBUF のポートの説明

ポート	I/O	説明
I	入力	データ入力信号
O	出力	データ出力信号

プリミティブのインスタンス化

Verilog でのインスタンス化 :

```

OBUF uut(
    .O(O),
    .I(I)
);

```

VHDL でのインスタンス化 :

```

COMPONENT OBUF
    PORT (
        O:OUT std_logic;
        I:IN std_logic
    );
END COMPONENT;

```



```
uut:OBUF
  PORT MAP(
    O=>O,
    I=>I
  );
```

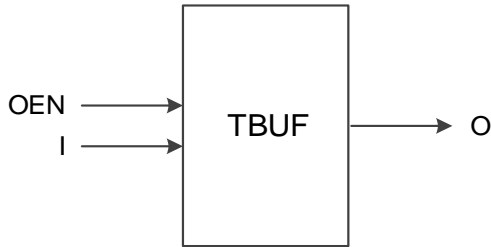
3.6.3 TBUF

プリミティブの紹介

TBUF(Output Buffer with Tristate Control)は、アクティブ Low のトライステートバッファです。

ポート図

図 3-8 TBUF のポート図



ポートの説明

表 3-3 TBUF のポートの説明

ポート	I/O	説明
I	入力	データ入力信号
OEN	入力	トライステート出力イネーブル
O	出力	データ出力信号

プリミティブのインスタンス化

Verilog でのインスタンス化 :

```
TBUF uut(
  .O(O),
  .I(I),
  .OEN(OEN)
);
```

VHDL でのインスタンス化 :

```
COMPONENT TBUF
  PORT (
```

```

        O:OUT std_logic;
        I:IN std_logic;
        OEN:IN std_logic

    );
END COMPONENT;
uut:TBUF
    PORT MAP(
        O=>O,
        I=>I,
        OEN=> OEN
    );

```

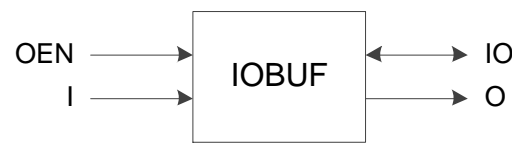
3.6.4 IOBUF

プリミティブの紹介

IOBUF(Bi-Directional Buffer)は双方向バッファです。OEN が High の場合は入力バッファとして使用され、Low の場合は出力バッファとして使用されます。

ポート図

図 3-9 IOBUF のポート図



ポートの説明

表 3-4 IOBUF のポートの説明

ポート	I/O	説明
I	入力	データ入力信号
OEN	入力	トリステート出力イネーブル
IO	双方向	入出力信号
O	出力	データ出力信号

プリミティブのインスタンス化

Verilog でのインスタンス化 :

```

IOBUF uut(
    .O(O),

```

```

        .IO(IO),
        .I(I),
        .OEN(OEN)
    );
VHDL でのインスタンス化 :
    COMPONENT IOBUF
    PORT (
        O:OUT std_logic;
        IO:INOUT std_logic;
        I:IN std_logic;
        OEN:IN std_logic
    );
    END COMPONENT;
    uut:IOBUF
    PORT MAP(
        O=>O,
        IO=>IO,
        I=>I,
        OEN=> OEN
    );

```

### 3.6.5 LVDS Input Buffer

#### プリミティブの紹介

LVDS 差動入力には、TLVDS\_IBUF と ELVDS\_IBUF の 2 種類があります。

TLVDS\_IBUF(True LVDS Input Buffer)は、真の差動入力バッファです。

注記：

GW1NZ-1、GW1N-1S デバイスは TLVDS\_IBUF をサポートしません。

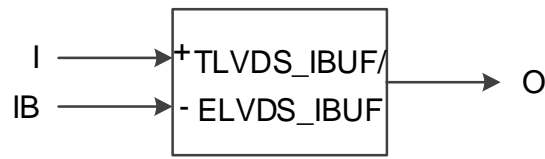
ELVDS\_IBUF(Emulated LVDS Input Buffer)は、エミュレート差動入力バッファです。

注記：

GW1NZ-1 デバイスは ELVDS\_IBUF をサポートしません。

ポート図

図 3-10 TLVDS\_IBUF/ELVDS\_IBUF のポート図



ポートの説明

表 3-5 TLVDS\_IBUF/ELVDS\_IBUF のポートの説明

ポート	I/O	説明
I	入力	差動入力A
IB	入力	差動入力B
O	出力	データ出力信号

プリミティブのインスタンス化

例 1

**Verilog** でのインスタンス化 :

```
TLVDS_IBUF uut(
    .O(O),
    .I(I),
    .IB(IB)
);
```

**VHDL** でのインスタンス化 :

```
COMPONENT TLVDS_IBUF
PORT (
    O:OUT std_logic;
    I:IN std_logic;
    IB:IN std_logic
);
END COMPONENT;
uut:TLVDS_IBUF
PORT MAP(
    O=>O,
    I=>I,
    IB=> IB
);
```

## 例 2

**Verilog** でのインスタンス化 :

```
ELVDS_IBUF uut(
    .O(O),
    .I(I),
    .IB(IB)
);
```

**VHDL** でのインスタンス化 :

```
COMPONENT ELVDS_IBUF
PORT (
    O:OUT std_logic;
    I:IN std_logic;
    IB:IN std_logic
);
END COMPONENT;
uut:ELVDS_IBUF
PORT MAP(
    O=>O,
    I=>I,
    IB=> IB
);
```

### 3.6.6 LVDS Ouput Buffer

#### プリミティブの紹介

LVDS 差動出力には、TLVDS\_OBUF と ELVDS\_OBUF の 2 種類があります。

TLVDS\_OBUF(True LVDS Output Buffer)は、真の差動出力バッファです。

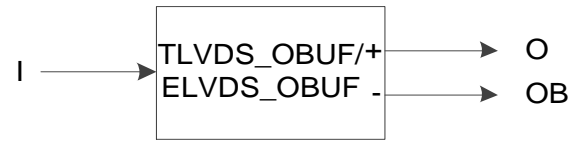
注記 :

GW1N-1、GW1NR-1、GW1NZ-1、GW1N-1S は TLVDS\_OBUF をサポートしません。

ELVDS\_OBUF(Emulated LVDS Output Buffer)は、エミュレート差動出力バッファです。

ポート図

図 3-11 TLVDS\_OBUF/ELVDS\_OBUF のポート図



ポートの説明

表 3-6 TLVDS\_OBUF/ELVDS\_OBUF のポートの説明

ポート	I/O	説明
I	入力	データ入力信号
OB	出力	差動出力B
O	出力	差動出力A

プリミティブのインスタンス化

例 1

**Verilog** でのインスタンス化 :

```
TLVDS_OBUF uut(
    .O(O),
    .OB(OB),
    .I(I)
);
```

**VHDL** でのインスタンス化 :

```
COMPONENT TLVDS_OBUF
PORT (
    O:OUT std_logic;
    OB:OUT std_logic;
    I:IN std_logic
);
END COMPONENT;
uut:TLVDS_OBUF
PORT MAP(
    O=>O,
    OB=>OB,
    I=> I
);
```

例 2

**Verilog** でのインスタンス化 :

```
ELVDS_OBUF uut(
    .O(O),
    .OB(OB),
    .I(I)
);
```

**VHDL** でのインスタンス化 :

```
COMPONENT ELVDS_OBUF
PORT (
    O:OUT std_logic;
    OB:OUT std_logic;
    I:IN std_logic
);
END COMPONENT;
uut:ELVDS_OBUF
PORT MAP(
    O=>O,
    OB=>OB,
    I=> I
);
```

### 3.6.7 LVDS Tristate Buffer

プリミティブの紹介

LVDS トライステート差動出力には、TLVDS\_TBUF と ELVDS\_TBUF の 2 種類があります。

TLVDS\_TBUF(True LVDS Tristate Buffer)は、真の差動トライステートバッファで、アクティブ Low です。

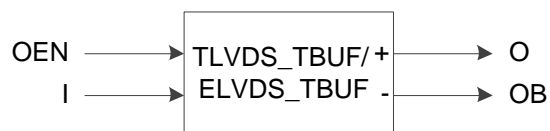
注記 :

GW1N-1、GW1NR-1、GW1NZ-1、GW1N-1S は TLVDS\_TBUF をサポートしません。

ELVDS\_TBUF(Emulated LVDS Tristate Buffer)は、エミュレート差動トライステートバッファで、アクティブ Low です。

ポート図

図 3-12 TLVDS\_TBUF/ELVDS\_TBUF のポート図



## ポートの説明

表 3-7 TLVDS\_TBUF/ELVDS\_TBUF のポートの説明

ポート	I/O	説明
I	入力	データ入力信号
OEN	入力	トライステート出力イネーブル
OB	出力	差動出力B
O	出力	差動出力A

## プリミティブのインスタンス化

## 例 1

**Verilog** でのインスタンス化 :

```
TLVDS_TBUF uut(
    .O(O),
    .OB(OB),
    .I(I),
    .OEN(OEN)
);
```

**VHDL** でのインスタンス化 :

```
COMPONENT TLVDS_TBUF
PORT (
    O:OUT std_logic;
    OB:OUT std_logic;
    I:IN std_logic;
    OEN:IN std_logic
);
END COMPONENT;
uut:TLVDS_TBUF
PORT MAP(
    O=>O,
    OB=>OB,
    I=> I,
    OEN=>OEN
);
```



## 例 2

**Verilog** でのインスタンス化 :

```

ELVDS_TBUF uut(
    .O(O),
    .OB(OB),
    .I(I),
    .OEN(OEN)
);

```

**VHDL** でのインスタンス化 :

```

COMPONENT ELVDS_TBUF
PORT (
    O:OUT std_logic;
    OB:OUT std_logic;
    I:IN std_logic;
    OEN:IN std_logic
);
END COMPONENT;
uut:ELVDS_TBUF
PORT MAP(
    O=>O,
    OB=>OB,
    I=> I,
    OEN=>OEN
);

```

### 3.6.8 LVDS Inout Buffer

#### プリミティブの紹介

LVDS 差動入出力には、TLVDS\_IOBUF と ELVDS\_IOBUF の 2 種類あります。

TLVDS\_IOBUF(True LVDS Bi-Directional Buffer)は、真の双方向バッファです。OEN が High の場合は、真の差動入力バッファとして使用され、OEN が Low の場合は、真の差動出力バッファとして使用されます。

#### サポートされるデバイス

表 3-8 TLVDS\_IOBUF 対応デバイス

ファミリー	シリーズ	デバイス
Arora	GW2A	GW2A-18, GW2A-18C, GW2A-55, GW2A-55C

ファミリー	シリーズ	デバイス
ファミリー	GW2AN	GW2AN-55C
	GW2AR	GW2AR-18, GW2AR-18C
	GW2ANR	GW2ANR-18C
LittleBee ファミリー	GW1N	GW1N-4, GW1N-4B, GW1N-4D
	GW1NR	GW1NR-4, GW1NR-4B, GW1NR-4D
	GW1NRF	GW1NRF-4B

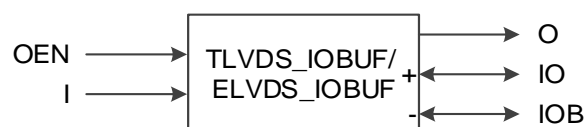
ELVDS\_IOBUF(Emulated LVDS Bi-Directional Buffer)は、エミュレート差動双方向バッファです。OEN が High の場合はエミュレート差動入力バッファとして使用され、OEN が Low の場合はエミュレート差動出力バッファとして使用されます。

注記：

GW1NZ-1 デバイスは ELVDS\_IOBUF をサポートしません。

ポート図

図 3-13 TLVDS\_IOBUF/ELVDS\_IOBUF のポート図



ポートの説明

表 3-9 TLVDS\_IOBUF/ELVDS\_IOBUF のポートの説明

ポート	I/O	説明
I	入力	データ入力信号
OEN	入力	トライステート出力イネーブル
O	出力	データ出力信号
IOB	双方向	差動入出力B
IO	双方向	差動入出力A

プリミティブのインスタンス化

**Verilog** でのインスタンス化：

```

ELVDS_IOBUF uut(
    .O(O),
    .IO(IO),
    .IOB(IOB),
    .I(I),

```

```

        .OEN(OEN)
    );
VHDL でのインスタンス化 :
    COMPONENT ELVDS_IOBUF
    PORT (
        O:OUT std_logic;
        IO:INOUT std_logic;
        IOB:INOUT std_logic;
        I:IN std_logic;
        OEN:IN std_logic
    );
    END COMPONENT;
    uut:ELVDS_IOBUF
    PORT MAP(
        O=>O,
        IO=>IO,
        IOB=>IOB,
        I=> I,
        OEN=>OEN
    );

```

### 3.6.9 MIPI\_IBUF

#### プリミティブの紹介

MIPI\_IBUF(MIPI Input Buffer )は、抵抗の動的構成をサポートする HS 入力モードと、LP 双方向モードとの 2 つの動作モードがあります。

#### サポートされるデバイス

表 3-10 MIPI\_IBUF 対応デバイス

ファミリー	シリーズ	デバイス
LittleBee ファミリー	GW1N	GW1N-9, GW1N-9C, GW1N-2, GW1N-2B, GW1N-2C, GW1N-1P5, GW1N-1P5B, GW1N-1P5C, GW1N-1S
	GW1NR	GW1NR-9, GW1NR-9C, GW1NR-2, GW1NR-2B, GW1NR-2C
	GW1NS	GW1NS-4, GW1NS-4C
	GW1NSER	GW1NSER-4C
	GW1NSR	GW1NSR-4, GW1NSR-4C
Arora	GW2AN	GW2AN-18X, GW2AN-9X

## 機能の説明

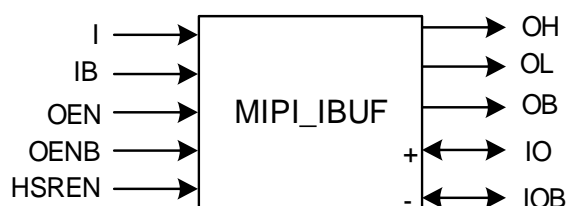
MIPI\_IBUF は、LP モードと HS モードをサポートします。IO および IOB は、pad に接続されます。

LP モード：双方向をサポートします。OEN が Low の場合、I は入力で、IO は出力です。OEN が High の場合、IO は入力で、OL は出力です。OENB が Low の場合、IB は入力で、IOB は出力です。OENB が High の場合、IOB は入力で、OB は出力です。

HS モード：IO と IOB は差動入力で、OH は出力です。HSREN は終端抵抗を制御します。

## ポート図

図 3-14 MIPI\_IBUF MIPI\_IBUF のポート図



## ポートの説明

表 3-11 MIPI\_IBUF のポートの説明

ポート	I/O	説明
I	入力	LPモードでは、OENがLowの場合、Iは入力です。
IB	入力	LPモードでは、OENBがLowの場合、IBは入力です。
HSREN	入力	HSモードにおける終端抵抗制御信号
OEN	入力	LPモードにおけるトライステート入出力の制御信号
OENB	入力	LPモードにおけるトライステート入出力の制御信号
OH	出力	HSモードにおけるデータ出力信号
OL	出力	LPモードでは、OENがHighの場合、OLは出力です。
OB	出力	LPモードでは、OENBがHighの場合、OBは出力です。
IO	双方向	<ul style="list-style-type: none"> <li>● LPモードでは、OEN が Low の場合、IO は出力で、OEN が High の場合、IO は入力です。</li> <li>● HSモードでは、IO は入力です。</li> </ul>
IOB	双方向	<ul style="list-style-type: none"> <li>● LPモードでは、OENB が Low の場合、IOB は出力で、OENB が High の場合、IOB は入力です。</li> <li>● HSモードでは、IOB は入力です。</li> </ul>

## 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_IBUF uut(  
    .OH(OH),  
    .OL(OL),  
    .OB(OB),  
    .IO(IO),  
    .IOB(IOB),  
    .I(I),  
    .IB(IB),  
    .OEN(OEN),  
    .OENB(OENB),  
    HSREN(HSREN)  
);
```

**VHDL** でのインスタンス化 :

```
COMPONENT MIPI_IBUF  
    PORT (  
        OH:OUT std_logic;  
        OL: OUT std_logic;  
        OB:OUT std_logic;  
        IO:INOUT std_logic;  
        IOB:INOUT std_logic;  
        I:IN std_logic;  
        IB:IN std_logic;  
        OEN:IN std_logic;  
        OENB:IN std_logic;  
        HSREN:IN std_logic  
    );  
END COMPONENT;  
uut: MIPI_IBUF  
    PORT MAP(  
        OH=>OH,  
        OL=>OL,  
        OB=>OB,  
        IO=>IO,  
        IOB=>IOB,  
        I=>I,
```

```

        IB=>IB,
        OEN=>OEN,
        OENB=>OENB,
        HSREN=>HSREN
    );

```

### 3.6.10 MIPI\_OBUF

#### プリミティブの紹介

MIPI\_OBUF には HS と LP の 2 つの動作モードがあります。

MIPI\_OBUF(MIPI Output Buffer)は、MIPI 出力バッファです。MODESEL が High の場合は、(HS)MIPI 高速出力バッファとして使用され、MODESEL が Low の場合は、(LP)MIPI 低消費電力出力バッファとして使用されます。

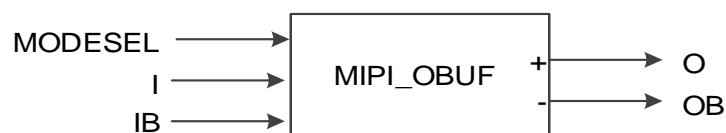
#### サポートされるデバイス

表 3-12 MIPI\_OBUF 対応デバイス

ファミリー	シリーズ	デバイス
LittleBee ファミリー	GW1N	GW1N-9, GW1N-9C
	GW1NR	GW1NR-9, GW1NR-9C
	GW1NS	GW1NS-4, GW1NS-4C
	GW1NSER	GW1NSER-4C
	GW1NSR	GW1NSR-4, GW1NSR-4C

#### ポート図

図 3-15 MIPI\_OBUF のポート図



#### ポートの説明

表 3-13 MIPI\_OBUF のポートの説明

ポート	I/O	説明
I	入力	HSモードまたはLPモードにおけるデータ入力A
IB	入力	LPモードにおけるデータ入力B
MODESEL	入力	モード選択(HSまたはLP)
O	出力	データ出力A。(HSモードでは差動出力A、LPモードではシングルエンド出力A)
OB	出力	データ出力B。(HSモードでは差動出力B、LPモードではシングルエンド出力B)

### 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_OBUF uut(
    .O(O),
    .OB(OB),
    .I(I),
    .IB(IB),
    .MODESEL(MODESEL)
);

```

#### VHDL でのインスタンス化 :

```

COMPONENT MIPI_OBUF
PORT (
    O:OUT std_logic;
    OB:OUT std_logic;
    I:IN std_logic;
    IB:IN std_logic;
    MODESEL:IN std_logic
);
END COMPONENT;

uut: MIPI_OBUF
PORT MAP(
    O=>O,
    OB=>OB,
    I=>I,
    IB=>IB,
    MDOESEL=>MODESEL
);

```

### 3.6.11 MIPI\_OBUF\_A

#### プリミティブの紹介

MIPI\_OBUF\_A には HS と LP の 2 種類の動作モードがあります。

MIPI\_OBUF\_A(MIPI Output Buffer with IL Signal)は、MIPI 出力バッファです。MODESEL が High の場合は(HS)MIPI 高速出力バッファとして使用され、MODESEL が Low の場合は(LP)MIPI 低消費電力出力バッファとして使用されます。MIPI\_OBUF と比較して、LP モードにおける入力 A として IL ポートが追加されています。

## サポートされるデバイス

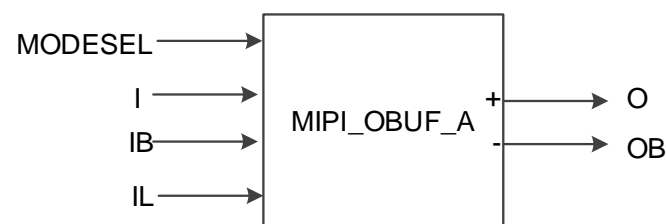
MIPI\_OBUF\_A 対応デバイスについては、表 3-12 および下表を参照してください。

表 3-14 MIPI\_OBUF\_A 対応デバイス(追加)

ファミリー	シリーズ	デバイス
LittleBee ファミリー	GW1N	GW1N-2, GW1N-2B, GW1N-2C, GW1N-1P5, GW1N-1P5B, GW1N-1P5C
	GW1NR	GW1NR-2, GW1NR-2B, GW1NR-2C

## ポート図

図 3-16 MIPI\_OBUF\_A のポート図



## ポートの説明

表 3-15 MIPI\_OBUF\_A のポートの説明

ポート	I/O	説明
I	入力	HSモードにおけるデータ入力A
IB	入力	LPモードにおけるデータ入力B
IL	入力	LPモードにおけるデータ入力A
MODESEL	入力	モード選択(HSまたはLP)
O	出力	データ出力A。(HSモードでは差動出力A、LPモードではシングルエンド出力A)
OB	出力	データ出力B。(HSモードでは差動出力B、LPモードではシングルエンド出力B)

## 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_OBUF_A uut(
    .O(O),
    .OB(OB),
    .I(I),
    .IB(IB),
    .IL(IL),

```



```

        .MODESEL(MODESEL)
    );
VHDL でのインスタンス化 :
    COMPONENT MIPI_OBUF_A
    PORT (
        O:OUT std_logic;
        OB:OUT std_logic;
        I:IN std_logic;
        IB:IN std_logic;
        IL: IN std_logic;
        MODESEL:IN std_logic
    );
    END COMPONENT;
    uut: MIPI_OBUF_A
    PORT MAP(
        O=>O,
        OB=>OB,
        I=>I,
        IB=>IB,
        IL=>IL,
        MDOESEL=>MODESEL
    );

```

### 3.6.12 I3C\_IOBUF

#### プリミティブの紹介

I3C\_IOBUF には、Normal と I3C の 2 つの動作モードがあります。

I3C\_IOBUF(I3C Bi-Directional Buffer)は、I3C 双方向バッファです。MODESEL が High の場合は I3C 双方向バッファとして使用され、MODESEL が Low の場合は通常の双方向バッファとして使用されます。

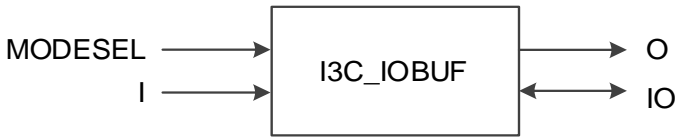
#### サポートされるデバイス

表 3-16 I3C\_IOBUF 対応デバイス

ファミリー	シリーズ	デバイス
LittleBee ファミリー	GW1N	GW1N-9, GW1N-9C
	GW1NR	GW1NR-9, GW1NR-9C
	GW1NS	GW1NS-4, GW1NS-4C
	GW1NSER	GW1NSER-4C
	GW1NSR	GW1NSR-4, GW1NSR-4C

ポート図

図 3-17 I3C\_IOBUF のポート図



ポートの説明

表 3-17 I3C\_IOBUF のポート図

ポート	I/O	説明
I	入力	データ入力信号
IO	双方向	入出力信号
MODESEL	入力	モード(NormalまたはI3C)選択信号
O	出力	データ出力信号

プリミティブのインスタンス化

**Verilog** でのインスタンス化 :

```
I3C_IOBUF uut(  
    .O(O),  
    .IO(IO),  
    .I(I),  
    .MODESEL(MODESEL)  
);
```

**VHDL** でのインスタンス化 :

```
COMPONENT I3C_IOBUF  
PORT (  
    O:OUT std_logic;  
    IO:INOUT std_logic;  
    I:IN std_logic;  
    MODESEL:IN std_logic  
);  
END COMPONENT;  
uut: I3C_IOBUF  
PORT MAP(  
    O=>O,
```

```
IO=>IO,  
I=>I,  
MDOESEL=>MODESEL  
);
```

3.6.13 MIPI\_IBUF\_HS/MIPI\_IBUF\_LP

プリミティブの紹介

MIPI\_IBUF\_HS は差動入力で HS モードを実装し、MIPI\_IBUF\_LP はシングルエンド入力で LP モードを実装します。

サポートされるデバイス

表 3-18 MIPI\_IBUF\_HS/MIPI\_IBUF\_LP 対応デバイス

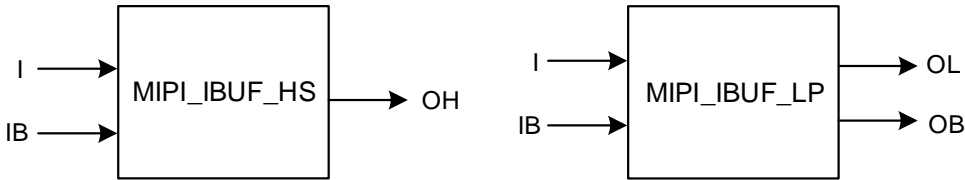
ファミリー	シリーズ	デバイス
LittleBeeファミリー	GW1NR	GW1NR-2

機能の説明

ユーザーは、MIPI\_IBUF\_HS と MIPI\_IBUF\_LP の組み合わせで HS モードと LP モードを実装でき、Floorplanner を使用してその位置を制約できます。MIPI\_IBUF\_HS の入力 I と MIPI\_IBUF\_LP の I は同じ信号に接続する必要があり、MIPI\_IBUF\_HS の入力 IB と MIPI\_IBUF\_LP の IB は同じ信号に接続する必要があります。

ポート図

図 3-18 MIPI\_IBUF\_HS/MIPI\_IBUF\_LP のポート図



ポートの説明

表 3-19 MIPI\_IBUF\_HS のポートの説明

ポート	I/O	説明
I	入力	HSモードにおける差動入力A
IB	入力	HSモードにおける差動入力B
OH	出力	HSモードにおけるデータ出力

表 3-20 MIPI\_IBUF\_LP のポートの説明

ポート	I/O	説明
I	入力	LPモードにおけるシングルエンド入力A

ポート	I/O	説明
IB	入力	LPモードにおけるシングルエンド入力B
OL	出力	LPモードにおける出力A
OB	出力	LPモードにおける出力B

#### 接続ルール

- MIPI\_IBUF\_HS の出力 OH は **lologic**(入出力ロジック)に接続できます。
- MIPI\_IBUF\_LP の出力 OL と OB は **lologic** に接続できません。

#### 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_IBUF_HS hs (
    .OH(OH),
    .I(I),
    .IB(IB)
);
MIPI_IBUF_LP lp (
    .OL(OL),
    .OB(OB),
    .I(I),
    .IB(IB)
);

```

##### VHDL でのインスタンス化 :

```

COMPONENT MIPI_IBUF_HS
PORT (
    OH:OUT std_logic;
    I:IN std_logic;
    IB:IN std_logic
);
END COMPONENT;
COMPONENT MIPI_IBUF_LP
PORT (
    OL: OUT std_logic;
    OB:OUT std_logic;

```

```

        I:IN std_logic;
        IB:IN std_logic

    );
END COMPONENT;
hs: MIPI_IBUF_HS
    PORT MAP(
        OH=>OH,
        I=>I,
        IB=>IB
    );
lp: MIPI_IBUF_LP
    PORT MAP(
        OL=>OL,
        OB=>OB,
        I=>I,
        IB=>IB
    );

```

### 3.6.14 ELVDS\_IBUF\_MIPI

#### プリミティブの紹介

ELVDS\_IBUF\_MIPI (Emulated LVDS Input MIPI Buffer) は、HS 入力モードと LP 入力モードの 2 つの動作モードを同時に有効にすることをサポートします。A ポートは HS モードをサポートし、B ポートは LP モードのみを実装します。

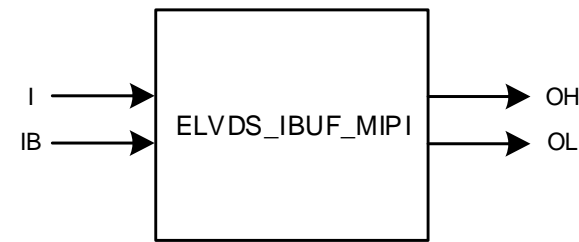
#### 機能の説明

ELVDS\_IBUF\_MIPI は LP モードと HS モードをサポートします。I と IB は pad に接続されます。

- LP モード : IB は入力で、OL は出力です。
- HS モード : I と IB は差動入力で、OH は出力です。

ポート図

図 3-19 ELVDS\_IBUF\_MIPi のポート図



ポートの説明

表 3-21 MIPi\_IBUF のポートの説明

ポート	I/O	説明
I	入力	HSモードでは、Iは入力です。
IB	入力	LPモードでは、IBは入力です。
OH	出力	HSモードにおけるデータ出力信号
OL	出力	LPモードにおけるデータ出力信号

プリミティブのインスタンス化

**Verilog** でのインスタンス化 :

```
ELVDS_IBUF_MIPi uut(  
    .OH(OH),  
    .OL(OL),  
    .I(I),  
    .IB(IB)  
);
```

**VHDL** でのインスタンス化 :

```
COMPONENT ELVDS_IBUF_MIPi  
PORT (  
    OH:OUT std_logic;  
    OL: OUT std_logic;  
    I:IN std_logic;  
    IB:IN std_logic  
);  
END COMPONENT;  
uut: ELVDS_IBUF_MIPi  
PORT MAP(  

```

```
OH=>OH,  
OL=>OL,  
I=>I,  
IB=>IB  
);
```

# 4 入出力ロジック

GOWIN セミコンダクターFPGA 製品の入出力ロジックは、SDR、DDR などの動作モードをサポートします。各動作モードでは、信号(または差動信号ペア)は、出力信号、入力信号、双方向信号、及びトライステート出力信号(トライステート制御付きの出力信号)に設定できます。

注記：

- GW1N-1、GW1NR-1、GW1NZ-1 デバイスの IOL6、IOR6 ピンは IO ロジックをサポートしません。
- GW1N-2、GW1NR-2、GW1N-1P5、GW1N-2B、GW1N-1P5B、および GW1NR-2B デバイスの IOT2、IOT3A ピンは IO ロジックをサポートしません。
- GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1N-4D、および GW1NR-4D デバイスの IOL10、IOR10 ピンは IO ロジックをサポートしません。

図 4-1 は GOWIN セミコンダクターFPGA 製品の入出力ロジックの出力を示します。

図 4-1 入出力ロジックの出力部の説明図

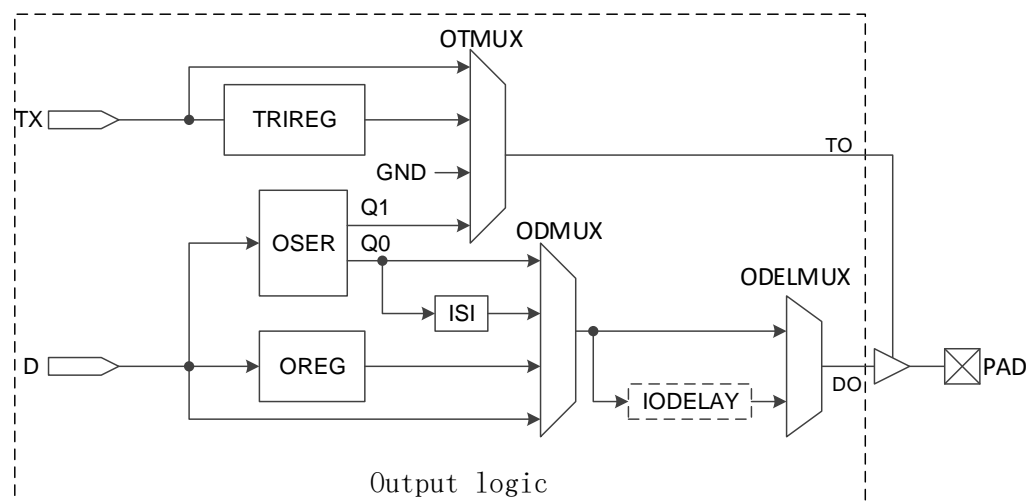
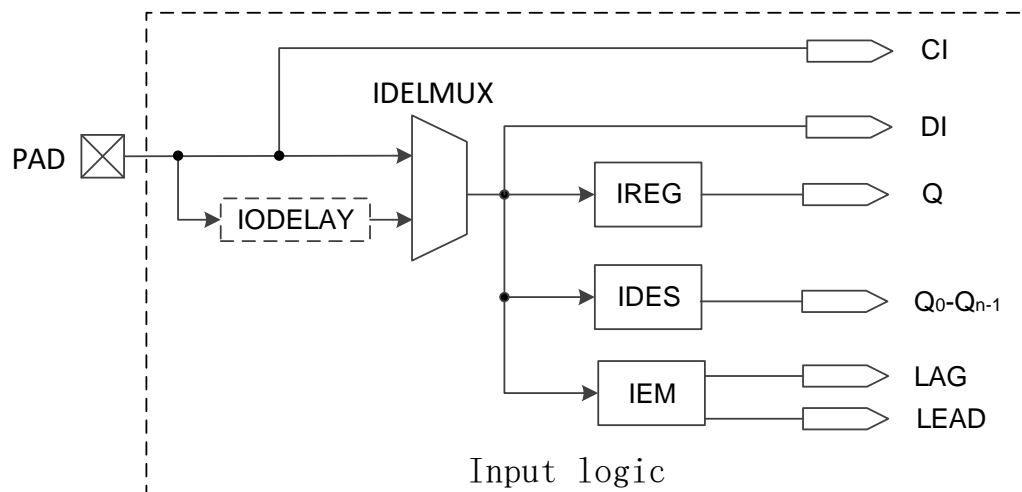


図 4-2 は GOWIN セミコンダクターFPGA 製品の入出力ロジックの入力を示します。



図 4-2 入出力ロジックの入力部の説明図



注記：

CI は GCLK 入力信号であり、ファブリックに接続できません。DI はファブリックに直接入力されます。

## 4.1 SDR モード

入出力ロジックは SDR モードをサポートし、入力レジスタ (IREG)、出力レジスタ (OREG)、およびトライステート・レジスタ (TRIREG) を提供します。その機能は CFU の FF/LATCH と同様です。FF/LATCH の入力 D が Buffer/IODELAY によって駆動され、かつ Buffer/IODELAY が他の IOLOGIC を駆動しない場合、または FF/LATCH の出力 Q が Buffer/IODELAY のみを駆動し、かつ Buffer が MIPI バッファでない場合、IOLOGIC として使用できます。

## 4.2 DDR モードの入力ロジック

### 4.2.1 IDDR

#### プリミティブの紹介

IDDR (Dual Data Rate Input) は、ダブル・データ・レートの入力を実現します。

#### 機能の説明

IDDR モードでは、出力データは同じクロックエッジで FPGA ロジックに提供されます。IDDR モードのブロック図は図 4-3、タイミング図は図 4-4 に示すとおりです。

図 4-3 IDDR のブロック図

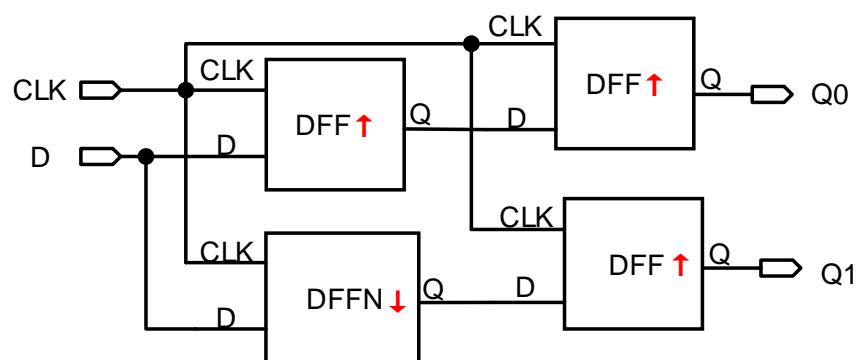
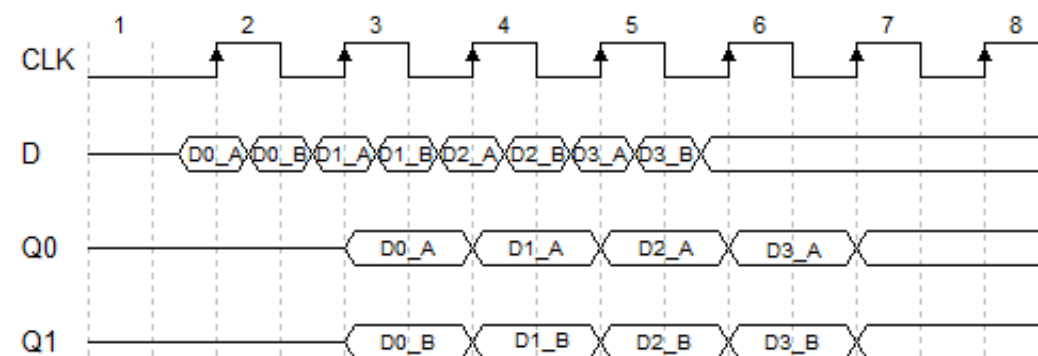


図 4-4 IDDR のタイミング図



ポート図

図 4-5 IDDR のポート図



ポートの説明

表 4-1 IDDR のポートの説明

ポート名	I/O	説明
D	入力	IDDRデータ入力信号
CLK	入力	クロック入力信号
Q0, Q1	出力	IDDRデータ出力信号

## パラメータの説明

表 4-2 IDDR のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
Q0_INIT	1'b0	1'b0	Q0出力の初期値
Q1_INIT	1'b0	1'b0	Q1出力の初期値

## 接続ルール

IDDR のデータ入力 D は、IBUF から直接取得するか、IODELAY モジュールを介してその出力 DO から取得することができます。

## プリミティブのインスタンス化

プリミティブを直接インスタンス化するか、IP Core Generator で生成できます。詳しくは、[5 IP の呼び出し](#)を参照してください。

### Verilog でのインスタンス化：

```
IDDR uut(
    .Q0(Q0),
    .Q1(Q1),
    .D(D),
    .CLK(CLK)
);
defparam uut.Q0_INIT = 1'b0;
defparam uut.Q1_INIT = 1'b0;
```

### VHDL でのインスタンス化：

```
COMPONENT IDDR
    GENERIC (Q0_INIT:bit:= '0';
             Q1_INIT:bit:= '0'
    );
    PORT(
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic
    );
END COMPONENT;
uut:IDDR
```

```
        GENERIC MAP (Q0_INIT=>'0',  
                      Q1_INIT=>'0'  
    )  
  
    PORT MAP (  
        Q0=>Q0,  
        Q1=>Q1,  
        D=>D,  
        CLK=>CLK  
    );
```

4.2.2 IDDRRC

プリミティブの紹介

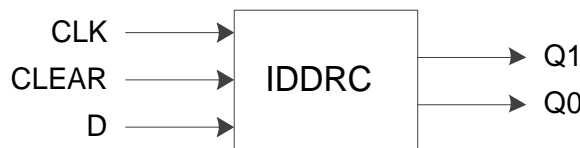
IDDRRC(Dual Data Rate Input with Asynchronous Clear)は、IDDDR に比べて、非同期リセット機能をさらに備えています。

機能の説明

IDDRRC モードでは、出力データは同じクロックエッジで FPGA ロジックに提供されます。

ポート図

図 4-6 IDDRRC のポート図



ポートの説明

表 4-3 IDDRRC のポートの説明

ポート名	I/O	説明
D	入力	IDDRRCデータ入力信号
CLK	入力	クロック入力信号
CLEAR	入力	非同期クリア入力、アクティブHigh
Q0, Q1	出力	IDDRRCデータ出力信号

パラメータの説明

表 4-4 IDDRRC のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
Q0_INIT	1'b0	1'b0	Q0出力の初期値
Q1_INIT	1'b0	1'b0	Q1出力の初期値

接続ルール

IDDRRC のデータ入力 D は、IBUF から直接取得するか、IODELAY モジュールを介してその出力 DO から取得することができます。

プリミティブのインスタンス化

プリミティブを直接インスタンス化するか、IP Core Generator で生成できます。詳しくは、[5 IP](#) の呼び出しを参照してください。

Verilog でのインスタンス化 :

```
IDDRRC uut(
```

```

        .Q0(Q0),
        .Q1(Q1),
        .D(D),
        .CLK(CLK),
        .CLEAR(CLEAR)
    );
    defparam uut.Q0_INIT = 1'b0;
    defparam uut.Q1_INIT = 1'b0;
VHDL でのインスタンス化 :
    COMPONENT IDDRRC
        GENERIC (Q0_INIT:bit:= '0';
                 Q1_INIT:bit:= '0'
        );
        PORT(
            Q0:OUT std_logic;
            Q1:OUT std_logic;
            D:IN std_logic;
            CLEAR:IN std_logic;
            CLK:IN std_logic
        );
    END COMPONENT;
    uut:IDDRRC
        GENERIC MAP (Q0_INIT=>'0',
                     Q1_INIT=>'0'
        )
        PORT MAP (
            Q0=>Q0,
            Q1=>Q1,
            D=>D,
            CLEAR=>CLEAR,
            CLK=>CLK
        );

```

### 4.2.3 IDDES4

#### プリミティブの紹介

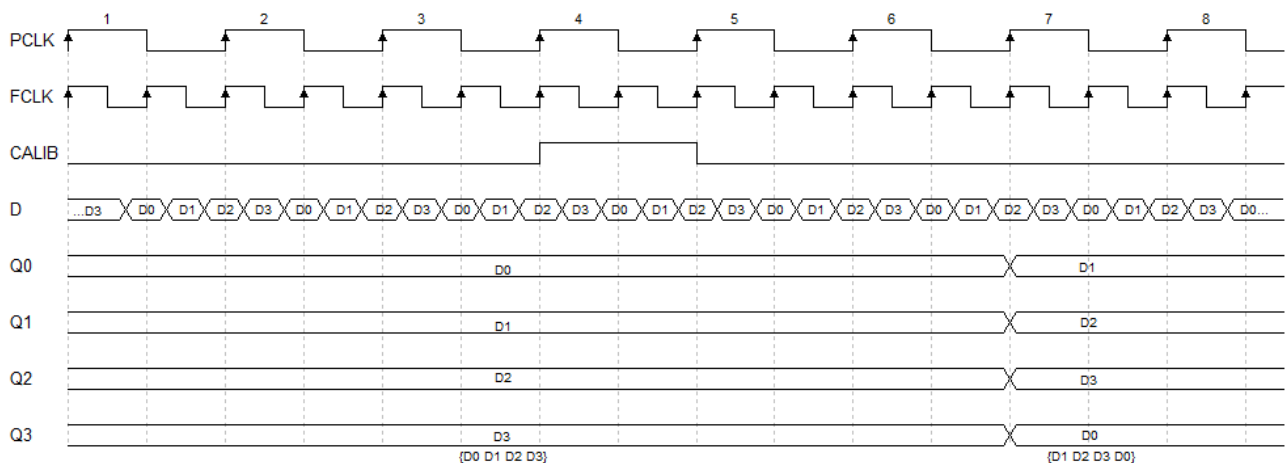
IDDES4(1 to 4 Deserializer)は、1 ビットのシリアル入力、4 ビットの

パラレル出力のデシリアライザです。

### 機能の説明

IDES4 モードでは、データは 1 : 4 デシリアライズされ、出力データは同じクロックエッジで **FPGA** ロジックに提供されます。**CALIB** によって出力データのシーケンスを調整することがサポートされます。データはパルスごとに 1 ビットシフトされ、4 回シフトされると、出力データはシフト前のデータと同じになります。**CALIB** のタイミングの例を図 4-7 に示します。

図 4-7 **CALIB** のタイミングの例



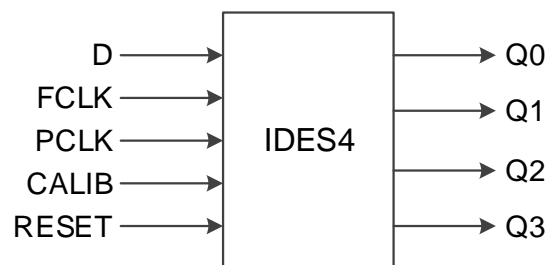
### 注記：

この例における **CALIB** 信号のパルス幅とタイミングは参照用であり、必要に応じて調整できます。しかし、そのパルス幅は  $T_{PCLK}$  以上である必要があります。

$PCLK$  は通常、 $FCLK$  の分周によって得られます： $f_{PCLK} = 1/2 f_{FCLK}$ 。

### ポート図

図 4-8 IDES4 のポート図



### ポートの説明

表 4-5 IDES4 のポートの説明

ポート名	I/O	説明
D	入力	IDES4データ入力信号
FCLK	入力	高速クロック入力信号
PCLK	入力	マスタークロック入力信号

ポート名	I/O	説明
CALIB	入力	出力データのシーケンスの調整用、アクティブHigh
RESET	入力	非同期リセット入力、アクティブHigh
Q3~Q0	出力	IDES4データ出力信号

### パラメータの説明

表 4-6 IDES4 のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
GSREN	"false", "true"	"false"	グローバルリセットGSRを有効にします
LSREN	"false", "true"	"true"	ローカルリセットRESETを有効にします

### 接続ルール

IDES4 のデータ入力 D は、IBUF から直接取得するか、IODELAY モジュールを介してその出力 DO から取得することができます。

### プリミティブのインスタンス化

プリミティブを直接インスタンス化するか、IP Core Generator で生成できます。詳しくは、[5 IP の呼び出し](#)を参照してください。

#### Verilog でのインスタンス化：

```

IDES4 uut(
    .Q0(Q0),
    .Q1(Q1),
    .Q2(Q2),
    .Q3(Q3),
    .D(D),
    .FCLK(FCLK),
    .PCLK(PCLK),
    .CALIB(CALIB),
    .RESET(RESET)
);
defparam uut.GSREN="false";
defparam uut.LSREN="true";

```

#### VHDL でのインスタンス化：



```
COMPONENT IDES4
    GENERIC (GSREN:string:="false";
             LSREN:string:="true"
    );
    PORT(
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        Q2:OUT std_logic;
        Q3:OUT std_logic;
        D:IN std_logic;
        FCLK:IN std_logic;
        PCLK:IN std_logic;
        CALIB:IN std_logic;
        RESET:IN std_logic
    );
END COMPONENT;

uut:IDES4
    GENERIC MAP (GSREN=>"false",
                 LSREN=>"true"
    )
    PORT MAP (
        Q0=>Q0,
        Q1=>Q1,
        Q2=>Q2,
        Q3=>Q3,
        D=>D,
        FCLK=>FCLK,
        PCLK=>PCLK,
        CALIB=>CALIB,
        RESET=>RESET
    );
```

## 4.2.4 IDES8

### プリミティブの紹介

IDES8(1 to 8 Deserializer)は、1 ビットのシリアル入力、8 ビットの平行出力のデシリアライザです。

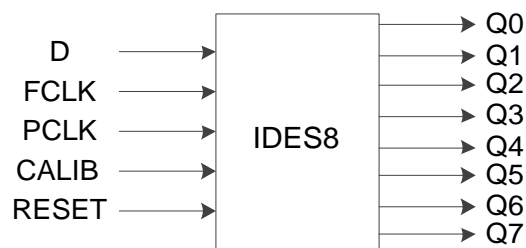
### 機能の説明

IDES8 モードでは、データは 1 : 8 デシリアライズされ、出力データは同じクロックエッジで FPGA ロジックに提供されます。CALIB によって出力シーケンスを調整することがサポートされます。データはパルスごとに 1 ビットシフトされ、8 回シフトされると、出力データはシフト前のデータと同じになります。

PCLK は通常、FCLK の分周によって得られます： $f_{PCLK} = 1/4 f_{FCLK}$ 。

### ポート図

図 4-9 IDES8 のポート図



### ポートの説明

表 4-7 IDES8 のポートの説明

ポート名	I/O	説明
D	入力	IDES8データ入力信号
FCLK	入力	高速クロック入力信号
PCLK	入力	マスタークロック入力信号
CALIB	入力	出力データのシーケンスの調整用、アクティブHigh
RESET	入力	非同期リセット入力、アクティブHigh
Q7~Q0	出力	IDES8データ出力信号

## パラメータの説明

表 4-8 IDES8 のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
GSREN	"false", "true"	"false"	グローバルリセットGSRを有効にします
LSREN	"false", "true"	"true"	ローカルリセットRESETを有効にします

## 接続ルール

IDES8 のデータ入力 D は、IBUF から直接取得するか、IODELAY モジュールを介してその出力 DO から取得することができます。

## プリミティブのインスタンス化

プリミティブを直接インスタンス化するか、IP Core Generator で生成できます。詳しくは、[5 IP の呼び出し](#)を参照してください。

### Verilog でのインスタンス化 :

```

IDES8 uut(
    .Q0(Q0),
    .Q1(Q1),
    .Q2(Q2),
    .Q3(Q3),
    .Q4(Q4),
    .Q5(Q5),
    .Q6(Q6),
    .Q7(Q7),
    .D(D),
    .FCLK(FCLK),
    .PCLK(PCLK),
    .CALIB(CALIB),
    .RESET(RESET)
);
defparam uut.GSREN="false";
defparam uut.LSREN ="true";

```

### VHDL でのインスタンス化 :

```

COMPONENT IDES8
    GENERIC (GSREN:string:="false";

```

```
LSREN:string:="true"
);
PORT(
  Q0:OUT std_logic;
  Q1:OUT std_logic;
  Q2:OUT std_logic;
  Q3:OUT std_logic;
  Q4:OUT std_logic;
  Q5:OUT std_logic;
  Q6:OUT std_logic;
  Q7:OUT std_logic;
  D:IN std_logic;
  FCLK:IN std_logic;
  PCLK:IN std_logic;
  CALIB:IN std_logic;
  RESET:IN std_logic
);
END COMPONENT;
uut:IDES8
  GENERIC MAP (GSREN=>"false",
               LSREN=>"true"
  )
  PORT MAP (
    Q0=>Q0,
    Q1=>Q1,
    Q2=>Q2,
    Q3=>Q3,
    Q4=>Q4,
    Q5=>Q5,
    Q6=>Q6,
    Q7=>Q7,
    D=>D,
    FCLK=>FCLK,
    PCLK=>PCLK,
    CALIB=>CALIB,
```

```

RESET=>RESET
);

```

## 4.2.5 IDES10

### プリミティブの紹介

IDES10(1 to 10 Deserializer)は、1 ビットのシリアル入力、10 ビットの平行出力のデシリアライザです。

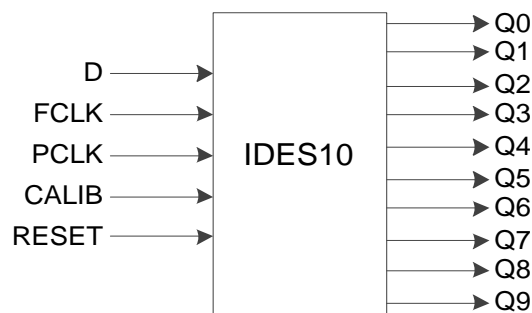
### 機能の説明

IDES10 モードでは、データは 1 : 10 デシリアライズされ、出力データは同じクロックエッジで FPGA ロジックに提供されます。CALIB によって出力データのシーケンスを調整することがサポートされます。データはパルスごとに 1 ビットシフトされ、10 回シフトされると、出力データはシフト前のデータと同じになります。

PCLK は通常、FCLK の分周によって得られます： $f_{PCLK} = 1/5 f_{FCLK}$ 。

### ポート図

図 4-10 IDES10 のポート図



### ポートの説明

表 4-9 IDES10 のポートの説明

ポート名	I/O	説明
D	入力	IDES10データ入力信号
FCLK	入力	高速クロック入力信号
PCLK	入力	マスタークロック入力信号
CALIB	入力	出力データのシーケンスの調整用、アクティブ High
RESET	入力	非同期リセット入力、アクティブ High
Q9~Q0	出力	IDES10データ出力信号

## パラメータの説明

表 4-10 IDDES10 のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
GSREN	"false", "true"	"false"	グローバルリセットGSRを有効にします
LSREN	"false", "true"	"true"	ローカルリセットRESETを有効にします

## 接続ルール

IDDES10 のデータ入力 D は、IBUF から直接取得するか、IODELAY モジュールを介してその出力 DO から取得することができます。

## プリミティブのインスタンス化

プリミティブを直接インスタンス化するか、IP Core Generator で生成できます。詳しくは、[5 IP の呼び出し](#)を参照してください。

### Verilog でのインスタンス化：

```
IDDES10 uut(
    .Q0(Q0),
    .Q1(Q1),
    .Q2(Q2),
    .Q3(Q3),
    .Q4(Q4),
    .Q5(Q5),
    .Q6(Q6),
    .Q7(Q7),
    .Q8(Q8),
    .Q9(Q9),
    .D(D),
    .FCLK(FCLK),
    .PCLK(PCLK),
    .CALIB(CALIB),
    .RESET(RESET)
);
defparam uut.GSREN="false";
defparam uut.LSREN ="true";
```

### VHDL でのインスタンス化：

```
COMPONENT IDDES10
```

```
        GENERIC (GSREN:string:="false";
                  LSREN:string:="true"
);
    PORT(
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        Q2:OUT std_logic;
        Q3:OUT std_logic;
        Q4:OUT std_logic;
        Q5:OUT std_logic;
        Q6:OUT std_logic;
        Q7:OUT std_logic;
        Q8:OUT std_logic;
        Q9:OUT std_logic;
        D:IN std_logic;
        FCLK:IN std_logic;
        PCLK:IN std_logic;
        CALIB:IN std_logic;
        RESET:IN std_logic
    );
END COMPONENT;
 uut:IDES10
    GENERIC MAP (GSREN=>"false",
                  LSREN=>"true"
    )
    PORT MAP (
        Q0=>Q0,
        Q1=>Q1,
        Q2=>Q2,
        Q3=>Q3,
        Q4=>Q4,
        Q5=>Q5,
        Q6=>Q6,
        Q7=>Q7,
        Q8=>Q8,
```

```
Q9=>Q9,  
D=>D,  
FCLK=>FCLK,  
PCLK=>PCLK,  
CALIB=>CALIB,  
RESET=>RESET  
);
```



## 4.2.6 IVIDEO

### プリミティブの紹介

IVIDEO(1 to 7 Deserializer)は、1 ビットのシリアル入力、7 ビットの平行出力のデシリアライザです。

### 機能の説明

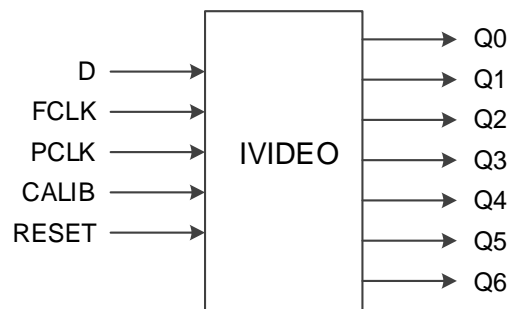
IVIDEO モードでは、データは 1 : 7 デシリアライズされ、出力データは同じクロックエッジで FPGA ロジックに提供されます。CALIB によって出力データのシーケンスを調整することがサポートされます。データはパルスごとに 2 ビットシフトされ、7 回シフトされると、出力データはシフト前のデータと同じになります。

PCLK は通常、FCLK の分周によって得られます：

$$f_{PCLK} = 1/3.5 f_{FCLK}。$$

### ポート図

図 4-11 IVIDEO のポート図



### ポートの説明

表 4-11 IVIDEO のポートの説明

ポート名	I/O	説明
D	入力	IVIDEOデータ入力信号
FCLK	入力	高速クロック入力信号
PCLK	入力	マスタークロック入力信号
CALIB	入力	出力データのシーケンスの調整用、アクティブ High
RESET	入力	非同期リセット入力、アクティブ High
Q6~Q0	出力	IVIDEOデータ出力

## パラメータの説明

表 4-12 IVIDEO のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
GSREN	"false", "true"	"false"	グローバルリセットGSRを有効にします
LSREN	"false", "true"	"true"	ローカルリセットRESETを有効にします

## 接続ルール

IVIDEO のデータ入力 D は、IBUF から直接取得するか、IODELAY モジュールを介してその出力 DO から取得することができます。

## プリミティブのインスタンス化

プリミティブを直接インスタンス化するか、IP Core Generator で生成できます。詳しくは、[5 IP の呼び出し](#)を参照してください。

### Verilog でのインスタンス化 :

```
IVIDEO uut(
    .Q0(Q0),
    .Q1(Q1),
    .Q2(Q2),
    .Q3(Q3),
    .Q4(Q4),
    .Q5(Q5),
    .Q6(Q6),
    .D(D),
    .FCLK(FCLK),
    .PCLK(PCLK),
    .CALIB(CALIB),
    .RESET(RESET)
);
defparam uut.GSREN="false";
defparam uut.LSREN="true";
```

### VHDL でのインスタンス化 :

```
COMPONENT IVIDEO
    GENERIC (GSREN:string:="false";
             LSREN:string:="true"
    );
```

```
        PORT(  
            Q0:OUT std_logic;  
            Q1:OUT std_logic;  
            Q2:OUT std_logic;  
            Q3:OUT std_logic;  
            Q4:OUT std_logic;  
            Q5:OUT std_logic;  
            Q6:OUT std_logic;  
            D:IN std_logic;  
            FCLK:IN std_logic;  
            PCLK:IN std_logic;  
            CALIB:IN std_logic;  
            RESET:IN std_logic  
        );  
    END COMPONENT;  
    uut:IVIDEO  
        GENERIC MAP (GSREN=>"false",  
                     LSREN=>"true"  
        )  
        PORT MAP (  
            Q0=>Q0,  
            Q1=>Q1,  
            Q2=>Q2,  
            Q3=>Q3,  
            Q4=>Q4,  
            Q5=>Q5,  
            Q6=>Q6,  
            D=>D,  
            FCLK=>FCLK,  
            PCLK=>PCLK,  
            CALIB=>CALIB,  
            RESET=>RESET  
        );
```

## 4.2.7 IDES16

### プリミティブの紹介

IDES16(1 to 16 Deserializer)は、1 ビットのシリアル入力、16 ビットの平行出力のデシリアライザです。

### サポートされるデバイス

表 4-13 IDES16 対応デバイス

ファミリー	シリーズ	デバイス
LittleBee ファミリー	GW1N	GW1N-1S, GW1N-9, GW1N-9C, GW1N-2, GW1N-1P5, GW1N-2B, GW1N-1P5B
	GW1NR	GW1NR-9, GW1NR-9C, GW1NR-2, GW1NR-2B
	GW1NS	GW1NS-4, GW1NS-4C
	GW1NSER	GW1NSER-4C
	GW1NSR	GW1NSR-4, GW1NSR-4C

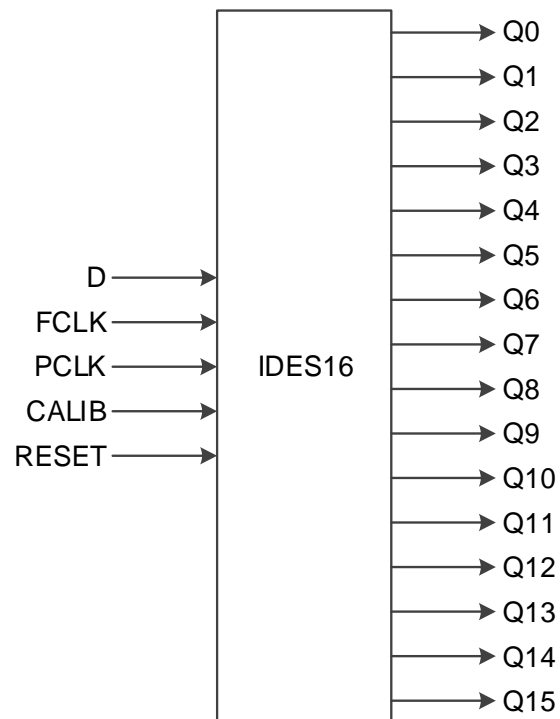
### 機能の説明

IDES16 モードでは、データは 1 : 16 デシリアライズされ、出力データは同じクロックエッジで FPGA ロジックに提供されます。CALIB によって出力データのシーケンスを調整することがサポートされます。データはパルスごとに 1 ビットシフトされ、16 回シフトされると、出力データはシフト前のデータと同じになります。

PCLK は通常、FCLK の分周によって得られます： $f_{PCLK} = 1/8 f_{FCLK}$ 。

## ポート図

図 4-12 IDES16 のポート図



## ポートの説明

表 4-14 IDES16 のポートの説明

ポート名	I/O	説明
D	入力	IDES16データ入力信号
FCLK	入力	高速クロック入力信号
PCLK	入力	マスタークロック入力信号
CALIB	入力	出力データのシーケンスの調整用、アクティブHigh
RESET	入力	非同期リセット入力、アクティブHigh
Q15~Q0	出力	IDES16データ出力信号

## パラメータの説明

表 4-15 IDES16 のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
GSREN	"false", "true"	"false"	グローバルリセットGSRを有効にします
LSREN	"false", "true"	"true"	ローカルリセットRESETを有効にします

### 接続ルール

IDES16 のデータ入力 D は、IBUF から直接取得するか、IODELAY モジュールを介してその出力 DO から取得することができます。

### プリミティブのインスタンス化

プリミティブを直接インスタンス化するか、IP Core Generator で生成できます。詳しくは、[5 IP の呼び出し](#)を参照してください。

#### Verilog でのインスタンス化 :

```
IDES16 uut(
    .Q0(Q0),
    .Q1(Q1),
    .Q2(Q2),
    .Q3(Q3),
    .Q4(Q4),
    .Q5(Q5),
    .Q6(Q6),
    .Q7(Q7),
    .Q8(Q8),
    .Q9(Q9),
    .Q10(Q10),
    .Q11(Q11),
    .Q12(Q12),
    .Q13(Q13),
    .Q14(Q14),
    .Q15(Q15),
    .D(D),
    .FCLK(FCLK),
    .PCLK(PCLK),
    .CALIB(CALIB),
    .RESET(RESET)
);
defparam uut.GSREN="false";
defparam uut.LSREN="true";
```

#### VHDL でのインスタンス化 :

```
COMPONENT IDES16
    GENERIC (GSREN:string:="false";
```

```
LSREN:string:="true"
);
PORT(
  Q0:OUT std_logic;
  Q1:OUT std_logic;
  Q2:OUT std_logic;
  Q3:OUT std_logic;
  Q4:OUT std_logic;
  Q5:OUT std_logic;
  Q6:OUT std_logic;
  Q7:OUT std_logic;
  Q8:OUT std_logic;
  Q9:OUT std_logic;
  Q10:OUT std_logic;
  Q11:OUT std_logic;
  Q12:OUT std_logic;
  Q13:OUT std_logic;
  Q14:OUT std_logic;
  Q15:OUT std_logic;
  D:IN std_logic;
  FCLK:IN std_logic;
  PCLK:IN std_logic;
  CALIB:IN std_logic;
  RESET:IN std_logic
);
END COMPONENT;
 uut:IDES16
    GENERIC MAP (GSREN=>"false",
                  LSREN=>"true"
    )
    PORT MAP (
      Q0=>Q0,
      Q1=>Q1,
      Q2=>Q2,
      Q3=>Q3,
```

```

Q4=>Q4,
Q5=>Q5,
Q6=>Q6,
Q7=>Q7,
Q8=>Q8,
Q9=>Q9,
Q10=>Q10,
Q11=>Q11,
Q12=>Q12,
Q13=>Q13,
Q14=>Q14,
Q15=>Q15,
D=>D,
FCLK=>FCLK,
PCLK=>PCLK,
CALIB=>CALIB,
RESET=>RESET
);

```

## 4.2.8 IDDR\_MEM

### プリミティブの紹介

IDDR\_MEM(Dual Data Rate Input with Memory)は、メモリ付きのダブル・データ・レートの実現します。

### サポートされるデバイス

表 4-16 IDDR\_MEM 対応デバイス

ファミリー	シリーズ	デバイス
Arora ファミリー	GW2A	GW2A-18, GW2A-18C, GW2A-55, GW2A-55C
	GW2AN	GW2AN-55C
	GW2AR	GW2AR-18, GW2AR-18C
	GW2ANR	GW2ANR-18C

### 機能の説明

IDDR\_MEM 出力データは同じクロックエッジで FPGA ロジックに提供されます。IDDR\_MEM には DQS が必要です。ICLK は DQS の出力信号 DQSR90 に接続され、データは ICLK のクロックエッジに従って IDDR\_MEM に入力されます。WADDR[2:0]は DQS の出力信号 WPOINT に接続されます。RADDR[2:0]は DQS の出力信号 RPOINT に接続されま



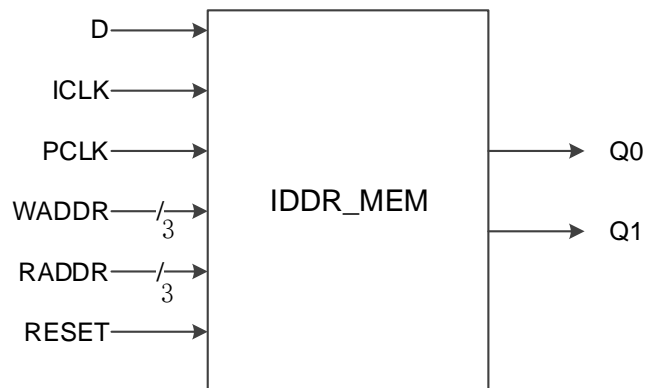
す。

PCLK と ICLK の周波数関係は： $f_{PCLK} = f_{ICLK}$ 。

PCLK と ICLK の間には一定の位相関係があり、その位相関係は DQS の DLLSTEP 値により決定できます。

ポート図

図 4-13 IDDR\_MEM のポート図



ポートの説明

表 4-17 IDDR\_MEM のポートの説明

ポート名	I/O	説明
D	入力	IDDR_MEMデータ入力信号
ICLK	入力	DQSモジュールのDQSR90からのクロック入力信号
PCLK	入力	マスタークロック入力信号
WADDR[2:0]	入力	DQSモジュールのWPOINTからの書き込みアドレス信号
RADDR[2:0]	入力	DQSモジュールのRPOINTからの読み出しアドレス信号
RESET	入力	非同期リセット入力、アクティブHigh
Q1~Q0	出力	IDDR_MEMデータ出力信号

パラメータの説明

表 4-18 IDDR\_MEM のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
GSREN	"false", "true"	"false"	グローバルリセットGSRを有効にします
LSREN	"false", "true"	"true"	ローカルリセットRESETを有効にします

### 接続ルール

- IDDR\_MEM のデータ入力 D は、IBUF から直接取得するか、IODELAY モジュールを介してその出力 DO から取得することができます。
- ICLK は、DQS モジュールの DQSR90 に接続する必要があります。
- WADDR[2:0]は、DQS モジュールの WPOINT に接続する必要があります。
- RADDR[2:0]は、DQS モジュールの RPOINT に接続する必要があります。

### プリミティブのインスタンス化

#### Verilog でのインスタンス化 :

```
IDDR_MEM iddr_mem_inst(
    .Q0(q0),
    .Q1(q1),
    .D(d),
    .ICLK (iclk),
    .PCLK(pclk),
    .WADDR(waddr[2:0]),
    .RADDR(raddr[2:0]),
    .RESET(reset)
);
```

```
defparam uut.GSREN="false";
```

```
defparam uut.LSREN ="true";
```

#### VHDL でのインスタンス化 :

```
COMPONENT IDDR_MEM
    GENERIC (GSREN:string:="false";
             LSREN:string:="true"
    );
    PORT(
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        D:IN std_logic;
        ICLK:IN std_logic;
        PCLK:IN std_logic;
        WADDR:IN std_logic_vector(2 downto 0);
```

```

        RADDR:IN std_logic_vector(2 downto 0);
        RESET:IN std_logic
    );
END COMPONENT;
uut:IDDR_MEM
    GENERIC MAP (GSREN=>"false",
                  LSREN=>"true"
    )
    PORT MAP (
        Q0=>q0,
        Q1=>q1,
        D=>d,
        ICLK=>iclk,
        PCLK=>pclk,
        WADDR=>waddr,
        RADDR=>raddr,
        RESET=>reset
    );
```

4.2.9 IDES4\_MEM

プリミティブの紹介

IDES4\_MEM(1 to 4 Deserializer with Memory) は、メモリ付きの 1:4 デシリアライザです。

サポートされるデバイス

表 4-19 IDES4\_MEM 対応デバイス

ファミリー	シリーズ	デバイス
Arora ファミリー	GW2A	GW2A-18, GW2A-18C, GW2A-55, GW2A-55C
	GW2AN	GW2AN-55C
	GW2AR	GW2AR-18, GW2AR-18C
	GW2ANR	GW2ANR-18C

機能の説明

IDES4\_MEM モードでは、データは 1:4 デシリアライズされ、出力データは同じクロックエッジで FPGA ロジックに提供されます。CALIB によって出力データのシーケンスを調整することがサポートされます。データはパルスごとに 1 ビットシフトされ、4 回シフトされると、出力データはシフト前のデータと同じになります。

IDES4\_MEM には DQS が必要です。ICLK は DQS の出力信号 DQSR90 に接続され、データは ICLK のクロックエッジに従って IDES4\_MEM に入力されます。WADDR[2:0] は DQS の出力信号 WPOINT に接続されます。RADDR[2:0] は DQS の出力信号 RPOINT に接続されます。

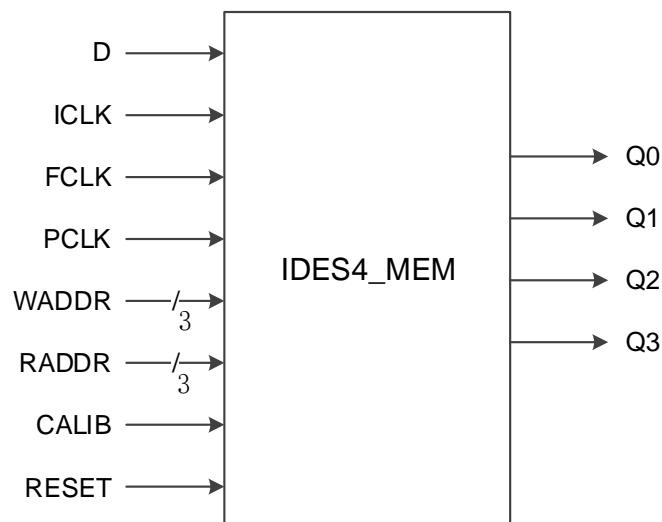
PCLK、FCLK、および ICLK の周波数関係は：

$$f_{PCLK} = 1/2 f_{FCLK} = 1/2 f_{ICLK}。$$

FCLK と ICLK の間には一定の位相関係があり、その位相関係は DQS の DLLSTEP 値により決定できます。

#### ポート図

図 4-14 IDES4\_MEM のポート図



#### ポートの説明

表 4-20 IDES4\_MEM のポートの説明

ポート名	I/O	説明
D	入力	IDES4_MEMデータ入力信号
ICLK	入力	DQSモジュールのDQSR90からのクロック入力信号
FCLK	入力	高速クロック入力信号
PCLK	入力	マスタークロック入力信号
WADDR[2:0]	入力	DQSモジュールのWPOINTからの書き込みアドレス信号
RADDR[2:0]	入力	DQSモジュールのRPOINTからの読み出しアドレス信号
CALIB	入力	出力データのシーケンスの調整用、アクティブ High
RESET	入力	非同期リセット入力、アクティブ High

ポート名	I/O	説明
Q3~Q0	出力	IDES4_MEMデータ出力信号

### パラメータの説明

表 4-21 IDES4\_MEM のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
GSREN	"false", "true"	"false"	グローバルリセットGSRを有効にします
LSREN	"false", "true"	"true"	ローカルリセットRESETを有効にします

### 接続ルール

- IDES4\_MEM のデータ入力 D は、IBUF から直接取得するか、IODELAY モジュールを介してその出力 DO から取得することができます。
- ICLK は、DQS モジュールの DQSR90 に接続する必要があります。
- WADDR[2:0]は、DQS モジュールの WPOINT に接続する必要があります。
- RADDR[2:0]は、DQS モジュールの RPOINT に接続する必要があります。

### プリミティブのインスタンス化

#### Verilog でのインスタンス化：

```

IDES4_MEM ides4_mem_inst(
    .Q0(q0),
    .Q1(q1),
    .Q2(q2),
    .Q3(q3),
    .D(d),
    .ICLK(iclk),
    .FCLK(fclk),
    .PCLK(pclk),
    .WADDR(waddr[2:0]),
    .RADDR(raddr[2:0]),
    .CALIB(calib),
    .RESET(reset)
);

```

```
defparam uut.GSREN="false";
```

```
defparam uut.LSREN="true";
```

**VHDL** でのインスタンス化 :

```
COMPONENT IDES4_MEM
```

```
    GENERIC (GSREN:string:="false";
             LSREN:string:="true"
```

```
);
```

```
PORT(
```

```
    Q0:OUT std_logic;
```

```
    Q1:OUT std_logic;
```

```
    Q2:OUT std_logic;
```

```
    Q3:OUT std_logic;
```

```
    D:IN std_logic;
```

```
    ICLK:IN std_logic;
```

```
    FCLK:IN std_logic;
```

```
    PCLK:IN std_logic;
```

```
    WADDR:IN std_logic_vector(2 downto 0);
```

```
    RADDR:IN std_logic_vector(2 downto 0);
```

```
    CALIB:IN std_logic;
```

```
    RESET:IN std_logic
```

```
);
```

```
END COMPONENT;
```

```
uut:IDES4_MEM
```

```
    GENERIC MAP (GSREN=>"false",
                 LSREN=>"true"
```

```
)
```

```
    PORT MAP (
```

```
        Q0=>q0,
```

```
        Q1=>q1,
```

```
        Q2=>q2,
```

```
        Q3=>q3,
```

```
        D=>d,
```

```
        ICLK=>iclk,
```

```
        FCLK=>fclk,
```

```
        PCLK=>pclk,
```

```

WADDR=>waddr,
RADDR=>raddr,
CALIB=>calib,
RESET=>reset
);

```

#### 4.2.10 IDES8\_MEM

##### プリミティブの紹介

IDES8\_MEM(1 to 8 Deserializer with Memory) は、メモリ付きの 1:8 デシリアライザです。

##### サポートされるデバイス

表 4-22 IDES8\_MEM 対応デバイス

ファミリー	シリーズ	デバイス
Arora ファミリー	GW2A	GW2A-18, GW2A-18C, GW2A-55, GW2A-55C
	GW2AN	GW2AN-55C
	GW2AR	GW2AR-18, GW2AR-18C
	GW2ANR	GW2ANR-18C

##### 機能の説明

IDES8\_MEM モードでは、データは 1:8 デシリアライズされ、出力データは同じクロックエッジで FPGA ロジックに提供されます。CALIB によって出力シーケンスを調整することがサポートされます。データはパルスごとに 1 ビットシフトされ、8 回シフトされると、出力データはシフト前のデータと同じになります。IDES8\_MEM には DQS が必要です。ICLK は DQS の出力信号 DQSR90 に接続され、データは ICLK のクロックエッジに従って IDES8\_MEM に入力されます。WADDR[2:0] は DQS の出力信号 WPOINT に接続されます。RADDR[2:0] は DQS の出力信号 RPOINT に接続されます。

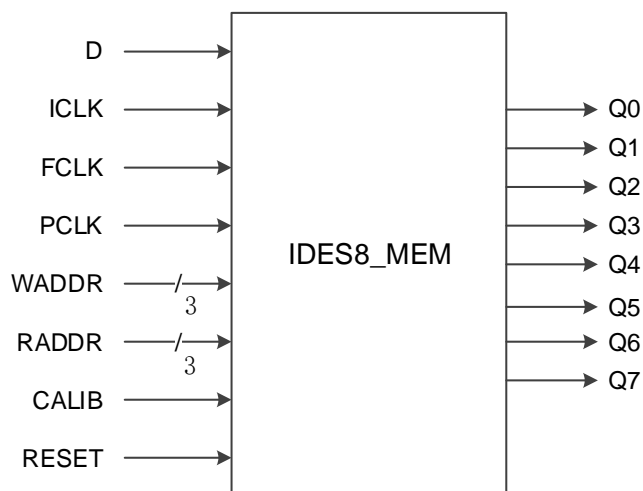
PCLK、FCLK、および ICLK の周波数関係は：

$$f_{PCLK} = 1/4 f_{FCLK} = 1/4 f_{ICLK}。$$

FCLK と ICLK の間には一定の位相関係があり、その位相関係は DQS の DLLSTEP 値により決定できます。

## ポート図

図 4-15 IDES8\_MEM のポート図



## ポートの説明

表 4-23 IDES8\_MEM のポートの説明

ポート名	I/O	説明
D	入力	IDES8_MEMデータ入力信号
ICLK	入力	DQSモジュールのDQSR90からのクロック入力信号
FCLK	入力	高速クロック入力信号
PCLK	入力	マスタークロック入力信号
WADDR[2:0]	入力	DQSモジュールのWPOINTからの書き込みアドレス
RADDR[2:0]	入力	DQSモジュールのRPOINTからの読み出しアドレス
CALIB	入力	出力データのシーケンスの調整用、アクティブHigh
RESET	入力	非同期リセット入力、アクティブHigh
Q7~Q0	出力	IDES8_MEMデータ出力信号

## パラメータの説明

表 4-24 IDES8\_MEM のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
GSREN	"false", "true"	"false"	グローバルリセットGSRを有効にします
LSREN	"false", "true"	"true"	ローカルリセットRESETを有効にします



**接続ルール**

- IDES8\_MEM のデータ入力 D は、IBUF から直接取得するか、IODELAY モジュールを介してその出力 DO から取得することができます。
- ICLK は、DQS モジュールの DQSR90 に接続する必要があります。
- WADDR[2:0]は、DQS モジュールの WPOINT に接続する必要があります。
- RADDR[2:0]は、DQS モジュールの RPOINT に接続する必要があります。

**プリミティブのインスタンス化****Verilog** でのインスタンス化 :

```

IDES8_MEM ides8_mem_inst(
    .Q0(q0),
    .Q1(q1),
    .Q2(q2),
    .Q3(q3),
    .Q4(q4),
    .Q5(q5),
    .Q6(q6),
    .Q7(q7),
    .D(d),
    .ICLK(iclk),
    .FCLK(fclk),
    .PCLK(pclk),
    .WADDR(waddr[2:0]),
    .RADDR(raddr[2:0]),
    .CALIB(calib),
    .RESET(reset)
);
defparam uut.GSREN="false";
defparam uut.LSREN ="true";

```

**VHDL** でのインスタンス化 :

```

COMPONENT IDES8_MEM
    GENERIC (GSREN:string:="false";

```

```
LSREN:string:="true"
);
PORT(
  Q0:OUT std_logic;
  Q1:OUT std_logic;
  Q2:OUT std_logic;
  Q3:OUT std_logic;
  Q4:OUT std_logic;
  Q5:OUT std_logic;
  Q6:OUT std_logic;
  Q7:OUT std_logic;
  D:IN std_logic;
  ICLK:IN std_logic;
  FCLK:IN std_logic;
  PCLK:IN std_logic;
  WADDR:IN std_logic_vector(2 downto 0);
  RADDR:IN std_logic_vector(2 downto 0);
  CALIB:IN std_logic;
  RESET:IN std_logic
);
END COMPONENT;
uut:IDES8_MEM
  GENERIC MAP (GSREN=>"false",
               LSREN=>"true"
  )
  PORT MAP (
    Q0=>q0,
    Q1=>q1,
    Q2=>q2,
    Q3=>q3,
    Q4=>q4,
    Q5=>q5,
    Q6=>q6,
    Q7=>q7,
    D=>d,
```

```

ICLK=>iclk,
FCLK=>fclk,
PCLK=>pclk,
WADDR=>waddr,
RADDR=>raddr,
CALIB=>calib,
RESET=>reset

```

```

);

```

## 4.3 DDR モードの出力ロジック

### 4.3.1 ODDR

#### プリミティブの紹介

ODDR(Dual Data Rate Output)は、ダブル・データ・レートの出力を実現します。

#### 機能の説明

ODDR モードは、FPGA からダブル・データ・レートの信号を転送するために使用されます。Q0 はダブル・データ・レートのデータ出力で、Q1 は Q0 に接続される IOBUF/TBUF の OEN 信号に使用されます。ODDR モードのブロック図は図 4-16、タイミング図は図 4-17 に示すとおりです。

図 4-16 ODDR のブロック図

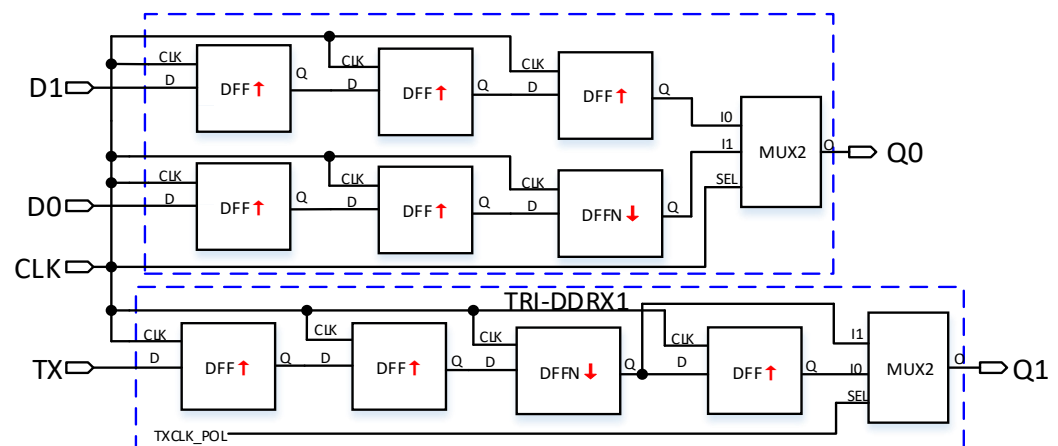
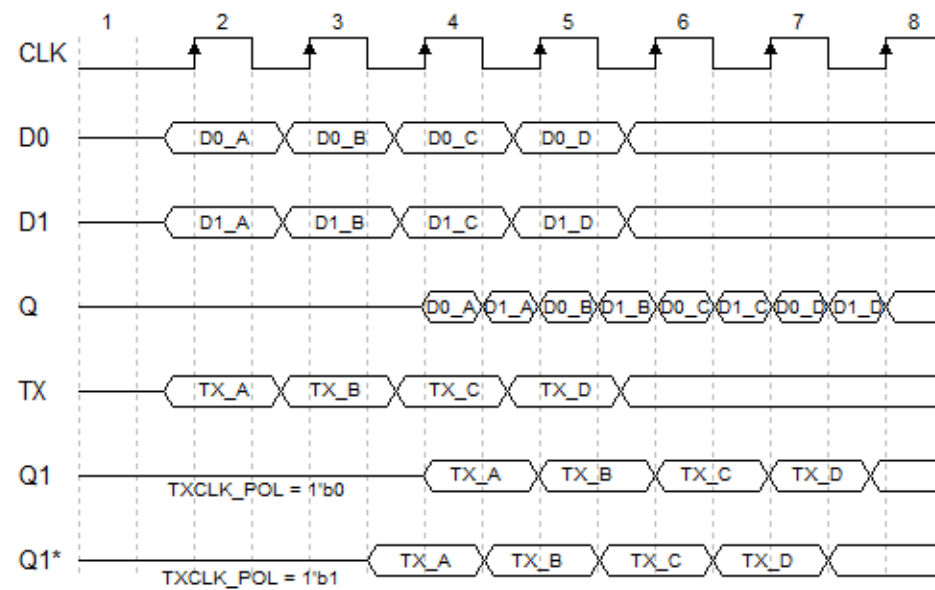


図 4-17 ODDR のタイミング図



## ポート図

図 4-18 ODDR のポート図



## ポートの説明

表 4-25 ODDR のポートの説明

ポート名	I/O	説明
D0, D1	入力	ODDRデータ入力信号
TX	入力	TRI-DDRX1を通じてQ1を生成
CLK	入力	クロック入力信号
Q0	出力	ODDRデータ出力信号
Q1	出力	ODDRトライステート・イネーブル信号。Q0に接続されるIOBUF/TBUFのOEN信号に接続するか、フローティングのままにします。

## パラメータの説明

表 4-26 ODDR のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
TXCLK_POL	1'b0, 1'b1	1'b0	Q1の出力クロックの極性の制御

パラメータ名	値の範囲	デフォルト値	説明
			<ul style="list-style-type: none"> <li>● 1'b0 : Q1 は立ち上がりエッジで出力されます</li> <li>● 1'b1 : Q1 は立ち下がりエッジで出力されます</li> </ul>
INIT	1'b0	1'b0	ODDR出力の初期値

### 接続ルール

- Q0 は、OBUF に直接接続するか、IODELAY モジュールを介してその入力ポート DI に接続します。
- Q1 は、Q0 に接続される IOBUF/TBUF の OEN 信号に接続するか、フローティングのままにする必要があります。

### プリミティブのインスタンス化

プリミティブを直接インスタンス化するか、IP Core Generator で生成できます。詳しくは、5\_IP の呼び出しを参照してください。

#### Verilog でのインスタンス化 :

```
ODDR uut(
    .Q0(Q0),
    .Q1(Q1),
    .D0(D0),
    .D1(D1),
    .TX(TX),
    .CLK(CLK)
);
defparam uut.INIT=1'b0;
defparam uut.TXCLK_POL=1'b0;
```

#### VHDL でのインスタンス化 :

```
COMPONENT ODDR
    GENERIC (CONSTANT INIT: std_logic='0';
             TXCLK_POL:bit='0'
    );
    PORT(
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        D0:IN std_logic;
```

```

        D1:IN std_logic;
        TX:IN std_logic;
        CLK:IN std_logic

    );
END COMPONENT;
uut:ODDR
    GENERIC MAP (INIT=>'0',
                  TXCLK_POL=>'0'
    )
    PORT MAP (
        Q0=>Q0,
        Q1=>Q1,
        D0=>D0,
        D1=>D1,
        TX=>TX,
        CLK=>CLK
    );

```

### 4.3.2 ODDRC

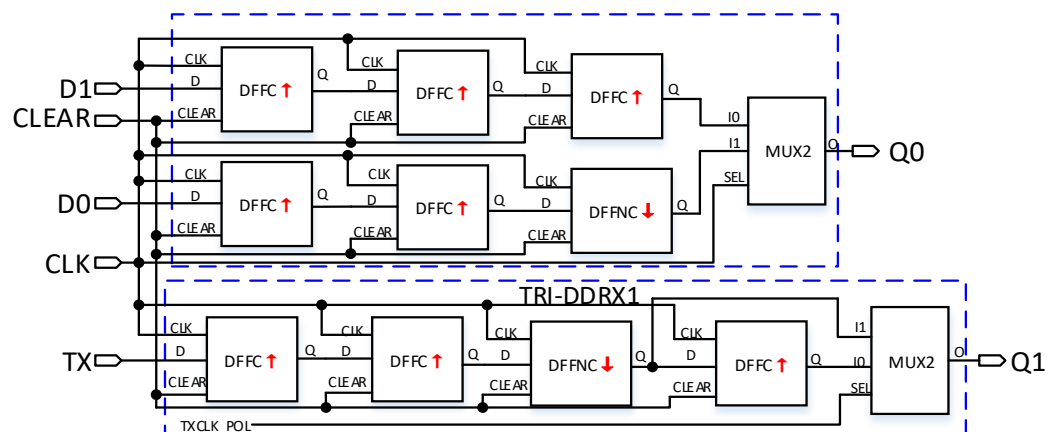
#### プリミティブの紹介

ODDRC(Dual Data Rate Output with Asynchronous Clear)は ODDR に比べて、非同期リセット機能をさらに備えています。

#### 機能の説明

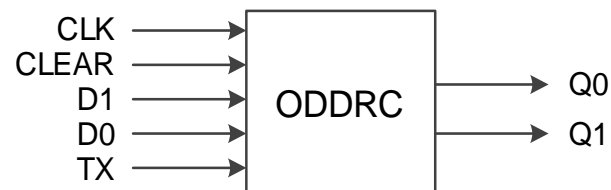
ODDRC モードは、FPGA からダブル・データ・レート of 信号を転送するために使用されます。Q0 はダブル・データ・レート of データ出力で、Q1 は Q0 に接続される IOBUF/TBUF の OEN 信号に使用されます。そのブロック図を図 4-19 に示します。

図 4-19 ODDRC のブロック図



## ポート図

図 4-20 ODDRC のポート図



## ポートの説明

表 4-27 ODDRC のポートの説明

ポート名	I/O	説明
D0, D1	入力	ODDRCデータ入力信号
TX	入力	TRI-DDRX1を通じてQ1を生成
CLK	入力	クロック入力信号
CLEAR	入力	非同期クリア入力、アクティブHigh
Q0	出力	ODDRCデータ出力
Q1	出力	ODDRCトライステート・イネーブル信号。 Q0に接続されるIOBUF/TBUFのOEN信号に接続するか、フローティングのままにします。

## パラメータの説明

表 4-28 ODDRC のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
TXCLK_POL	1'b0, 1'b1	1'b0	Q1の出力クロックの極性の制御 <ul style="list-style-type: none"> <li>1'b0 : Q1は立ち上がりエッジで出力されます</li> <li>1'b1 : Q1は立ち下がりエッジで出力されます</li> </ul>
INIT	1'b0	1'b0	ODDRC出力の初期値

## 接続ルール

- Q0 は、OBUF に直接接続するか、IODELAY モジュールを介してその入力ポート DI に接続します。
- Q1 は、Q0 に接続される IOBUF/TBUF の OEN 信号に接続するか、フローティングのままにする必要があります。

## プリミティブのインスタンス化

プリミティブを直接インスタンス化するか、IP Core Generator で生

成できます。詳しくは、[5 IP の呼び出し](#)を参照してください。

**Verilog** でのインスタンス化 :

```
ODDRC uut(  
    .Q0(Q0),  
    .Q1(Q1),  
    .D0(D0),  
    .D1(D1),  
    .TX(TX),  
    .CLK(CLK),  
    .CLEAR(CLEAR)  
);  
defparam uut.INIT=1'b0;  
defparam uut.TXCLK_POL=1'b0;
```

**VHDL** でのインスタンス化 :

```
COMPONENT ODDRC  
    GENERIC (CONSTANT INIT : std_logic := '0';  
             TXCLK_POL : bit := '0'  
    );  
    PORT(  
        Q0:OUT std_logic;  
        Q1:OUT std_logic;  
        D0:IN std_logic;  
        D1:IN std_logic;  
        TX:IN std_logic;  
        CLK:IN std_logic;  
        CLEAR:IN std_logic  
    );  
END COMPONENT;  
uut:ODDRC  
    GENERIC MAP (INIT=>'0',  
                 TXCLK_POL=>'0'  
    )  
    PORT MAP (  
        Q0=>Q0,  
        Q1=>Q1,
```



```

D0=>D0,
D1=>D1,
TX=>TX,
CLK=>CLK,
CLEAR=>CLEAR
);

```

### 4.3.3 OSER4

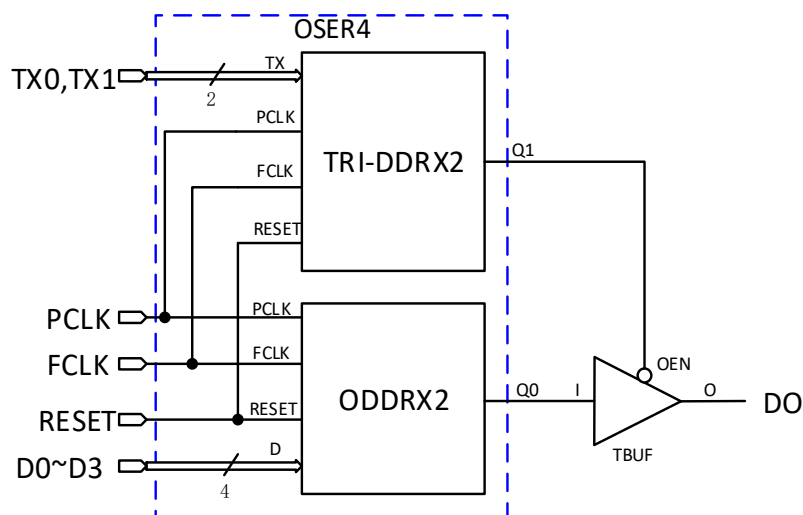
#### プリミティブの紹介

OSER4(4 to 1 Serializer)は、4 ビットの平行入力、1 ビットのシリアル出力のシリアライザです。

#### 機能の説明

OSER4 モードでは、データは 4 : 1 シリアライズされます。Q0 は、OSER4 データのシリアル出力で、Q1 は Q0 に接続される IOBUF/TBUF の OEN 信号に使用されます。TX0/TX1 は、IOBUF/TBUF の OEN 入力制御信号であり、データ D0~D3 と同期に DDR 経由で出力できます。TX0/TX1 は、TRI-DDRX2 を介して、IOBUF/TBUF の OEN 信号に接続される Q1 として出力されます。D0~D3 は、ODDRX2 を介して、IOBUF/TBUF に接続されるデータ入力 I として出力されます(出力順序は D0、D1、D2、D3)。そのブロック図を図 4-21 に示します。

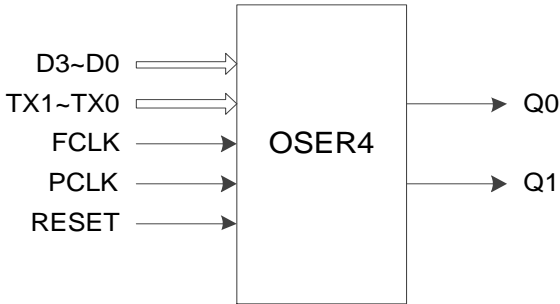
図 4-21 OSER4 のブロック図



PCLK は通常、FCLK の分周によって得られます： $f_{PCLK} = 1/2 f_{FCLK}$ 。

ポート図

図 4-22 OSER4 のポート図



ポートの説明

表 4-29 OSER4 のポートの説明

ポート名	I/O	説明
D3~D0	入力	OSER4データ入力信号
TX1~TX0	入力	TRI-DDRX2を通じてQ1を生成
FCLK	入力	高速クロック入力信号
PCLK	入力	マスタークロック入力信号
RESET	入力	非同期リセット入力、アクティブHigh
Q0	出力	OSER4データ出力信号
Q1	出力	OSER4トライステート・イネーブル信号。 Q0に接続されるIOBUF/TBUFのOEN信号に接続するか、フローティングのままにすることができます。

## パラメータの説明

表 4-30 OSER4 のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
GSREN	"false", "true"	"false"	グローバルリセットGSRを有効にします
LSREN	"false", "true"	"true"	ローカルリセットRESETを有効にします
TXCLK_POL	1'b0, 1'b1	1'b0	Q1の出力クロックの極性の制御 <ul style="list-style-type: none"> <li>● 1'b0 : データは立ち上がりエッジで出力されます。</li> <li>● 1'b1 : データは立ち下がりエッジで出力されます。</li> </ul>
HWL	"false", "true"	"false"	OSER4データd_up0/1のタイミング関係の制御 <ul style="list-style-type: none"> <li>● "false" : d_up1 は d_up0 より 1 サイクル先です。</li> <li>● "true" : d_up1 と d_up0 のタイミングは同じです。</li> </ul>

## 接続ルール

- Q0 は、OBUF に直接接続するか、IODELAY モジュールを介してその入力ポート DI に接続します。
- Q1 は、Q0 に接続される IOBUF/TBUF の OEN 信号に接続するか、フローティングのままにする必要があります。

## プリミティブのインスタンス化

プリミティブを直接インスタンス化するか、IP Core Generator で生成できます。詳しくは、[5 IP の呼び出し](#)を参照してください。

### Verilog でのインスタンス化 :

```
OSER4 uut(
    .Q0(Q0),
    .Q1(Q1),
    .D0(D0),
    .D1(D1),
    .D2(D2),
    .D3(D3),
    .TX0(TX0),
    .TX1(TX1),
    .PCLK(PCLK),
```

```

        .FCLK(FCLK),
        .RESET(RESET)
    );
    defparam uut.GSREN="false";
    defparam uut.LSREN ="true";
    defparam uut.HWL ="false";
    defparam uut.TXCLK_POL =1'b0;
VHDL でのインスタンス化 :
    COMPONENT OSER4
        GENERIC (GSREN:string:="false";
                  LSREN:string:="true";
                  HWL:string:="false";
                  TXCLK_POL:bit:='0'
        );
        PORT(
            Q0:OUT std_logic;
            Q1:OUT std_logic;
            D0:IN std_logic;
            D1:IN std_logic;
            D2:IN std_logic;
            D3:IN std_logic;
            TX0:IN std_logic;
            TX1:IN std_logic;
            FCLK:IN std_logic;
            PCLK:IN std_logic;
            RESET:IN std_logic
        );
    END COMPONENT;
    uut:OSER4
        GENERIC MAP (GSREN=>"false",
                     LSREN=>"true",
                     HWL=>"false",
                     TXCLK_POL=>'0'
        )
        PORT MAP (

```

```

Q0=>Q0,
Q1=>Q1,
D0=>D0,
D1=>D1,
D2=>D2,
D3=>D3,
TX0=>TX0,
TX1=>TX1,
FCLK=>FCLK,
PCLK=>PCLK,
RESET=>RESET
);

```

#### 4.3.4 OSER8

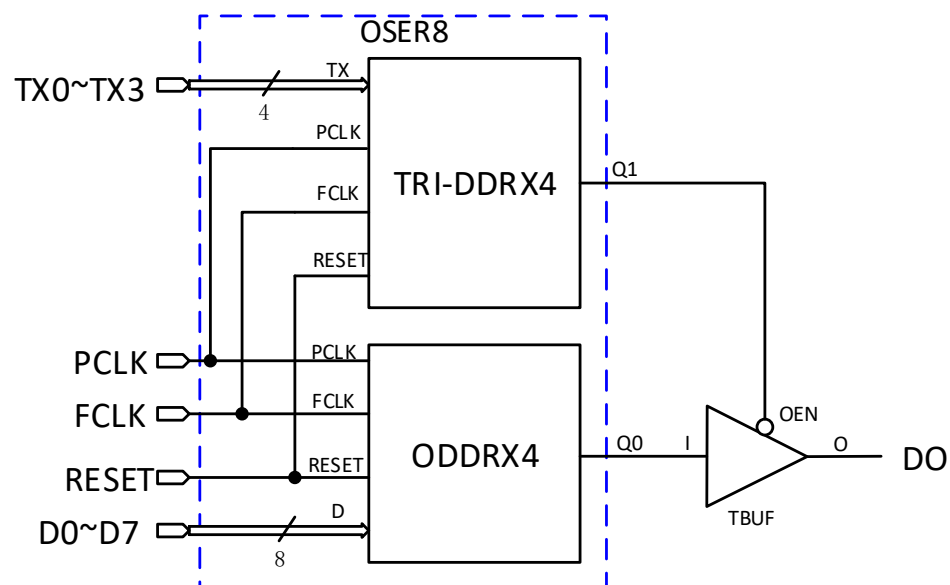
##### プリミティブの紹介

OSER8(8 to 1 Serializer)は、8 ビットの平行入力、1 ビットのシリアル出力のシリアライザです。

##### 機能の説明

OSER8 モードでは、データは 8:1 シリアライズされます。Q0 は OSER8 データのシリアル出力で、Q1 は Q0 に接続される IOBUF/TBUF の OEN 信号に使用されます。そのブロック図を図 4-23 に示します。

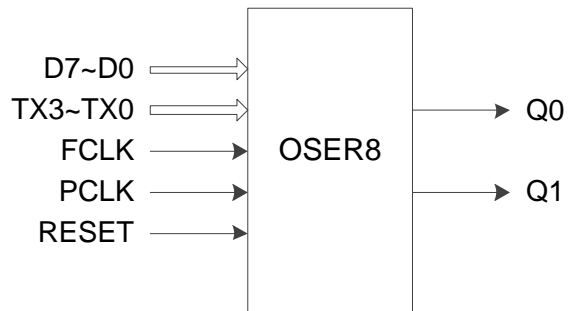
図 4-23 OSER8 のブロック図



PCLK は通常、FCLK の分周によって得られます： $f_{PCLK} = 1/4 f_{FCLK}$ 。

## ポート図

図 4-24 OSER8 のポート図



## ポートの説明

表 4-31 OSER8 のポートの説明

ポート名	I/O	説明
D7~D0	入力	OSER8データ入力信号
TX3~TX0	入力	TRI-DDRX4を通じてQ1を生成
FCLK	入力	高速クロック入力信号
PCLK	入力	マスタークロック入力信号
RESET	入力	非同期リセット入力、アクティブHigh
Q0	出力	OSER8データ出力信号
Q1	出力	OSER8トライステート・イネーブル信号。Q0に接続されるIOBUF/TBUFのOEN信号に接続するか、フローティングのままにすることができます。

## パラメータの説明

表 4-32 OSER8 のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
GSREN	"false", "true"	"false"	グローバルリセットGSRを有効にします
LSREN	"false", "true"	"true"	ローカルリセットRESETを有効にします
TXCLK_POL	1'b0, 1'b1	1'b0	Q1の出力クロックの極性の制御 <ul style="list-style-type: none"> <li>● 1'b0 : データは立ち上がりエッジで出力されます。</li> <li>● 1'b1 : データは立ち下がりエッジで出力されます。</li> </ul>
HWL	"false", "true"	"false"	OSER8データd_up0/1のタイミング関係の制御

パラメータ名	値の範囲	デフォルト値	説明
			<ul style="list-style-type: none"> <li>● "false" : d_up1 は d_up0 より 1 サイクル先です。</li> <li>● "true" : d_up1 と d_up0 のタイミングは同じです。</li> </ul>

#### 接続ルール

- Q0 は、OBUF に直接接続するか、IODELAY モジュールを介してその入力ポート DI に接続します。
- Q1 は、Q0 に接続される IOBUF/TBUF の OEN 信号に接続するか、フローティングのままにする必要があります。

#### プリミティブのインスタンス化

プリミティブを直接インスタンス化するか、IP Core Generator で生成できます。詳しくは、[5 IP の呼び出し](#)を参照してください。

#### Verilog でのインスタンス化：

```
OSER8 uut(
    .Q0(Q0),
    .Q1(Q1),
    .D0(D0),
    .D1(D1),
    .D2(D2),
    .D3(D3),
    .D4(D4),
    .D5(D5),
    .D6(D6),
    .D7(D7),
    .TX0(TX0),
    .TX1(TX1),
    .TX2(TX2),
    .TX3(TX3),
    .PCLK(PCLK),
    .FCLK(FCLK),
    .RESET(RESET)
);
defparam uut.GSREN="false";
```

```
defparam uut.LSREN ="true";  
defparam uut.HWL ="false";  
defparam uut.TXCLK_POL =1'b0;
```

**VHDL** でのインスタンス化 :

```
COMPONENT OSER8  
    GENERIC (GSREN:string:="false";  
             LSREN:string:="true";  
             HWL:string:="false";  
             TXCLK_POL:bit:='0'  
    );  
    PORT(  
        Q0:OUT std_logic;  
        Q1:OUT std_logic;  
        D0:IN std_logic;  
        D1:IN std_logic;  
        D2:IN std_logic;  
        D3:IN std_logic;  
        D4:IN std_logic;  
        D5:IN std_logic;  
        D6:IN std_logic;  
        D7:IN std_logic;  
        TX0:IN std_logic;  
        TX1:IN std_logic;  
        TX2:IN std_logic;  
        TX3:IN std_logic;  
        FCLK:IN std_logic;  
        PCLK:IN std_logic;  
        RESET:IN std_logic  
    );  
END COMPONENT;  
uut:OSER8  
    GENERIC MAP (GSREN=>"false",  
                 LSREN=>"true",  
                 HWL=>"false",  
                 TXCLK_POL=>'0'
```



```

)
    PORT MAP (
        Q0=>Q0,
        Q1=>Q1,
        D0=>D0,
        D1=>D1,
        D2=>D2,
        D3=>D3,
        D4=>D4,
        D5=>D5,
        D6=>D6,
        D7=>D7,
        TX0=>TX0,
        TX1=>TX1,
        TX2=>TX2,
        TX3=>TX3,
        FCLK=>FCLK,
        PCLK=>PCLK,
        RESET=>RESET
    );

```

### 4.3.5 OSER10

#### プリミティブの紹介

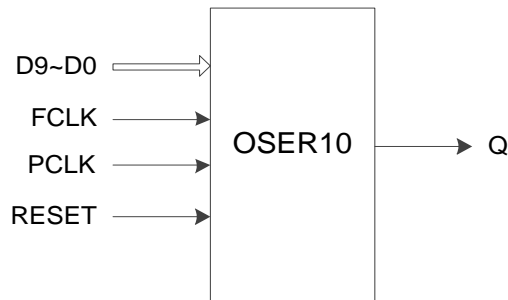
OSER10(10 to 1 Serializer)は、10 ビットのパラレル入力、1 ビットのシリアル出力のシリアライザです。

#### 機能の説明

OSER10 モードでは、データは 10:1 シリアライズされます。PCLK は通常、FCLK の分周によって得られます： $f_{PCLK} = 1/5 f_{FCLK}$ 。

## ポート図

図 4-25 OSER10 のポート図



## ポートの説明

表 4-33 OSER10 のポートの説明

ポート名	I/O	説明
D9~D0	入力	OSER10データ入力信号
FCLK	入力	高速クロック入力信号
PCLK	入力	マスタークロック入力信号
RESET	入力	非同期リセット入力、アクティブHigh
Q	出力	OSER10データ出力信号

## パラメータの説明

表 4-34 OSER10 のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
GSREN	"false", "true"	"false"	グローバルリセットGSRを有効にします
LSREN	"false", "true"	"true"	ローカルリセットRESETを有効にします

## 接続ルール

Q は、OBUF に直接接続するか、IODELAY モジュールを介してその入力ポート DI に接続します。

## プリミティブのインスタンス化

プリミティブを直接インスタンス化するか、IP Core Generator で生成できます。詳しくは、[5 IP の呼び出し](#)を参照してください。

## Verilog でのインスタンス化：

```

OSER10 uut(
    .Q(Q),
    .D0(D0),

```

```
.D1(D1),  
.D2(D2),  
.D3(D3),  
.D4(D4),  
.D5(D5),  
.D6(D6),  
.D7(D7),  
.D8(D8),  
.D9(D9),  
.PCLK(PCLK),  
.FCLK(FCLK),  
.RESET(RESET)  
);  
defparam uut.GSREN="false";  
defparam uut.LSREN="true";
```

**VHDL** でのインスタンス化 :

```
COMPONENT OSER10  
    GENERIC (GSREN:string:="false";  
             LSREN:string:="true"  
    );  
    PORT(  
        Q:OUT std_logic;  
        D0:IN std_logic;  
        D1:IN std_logic;  
        D2:IN std_logic;  
        D3:IN std_logic;  
        D4:IN std_logic;  
        D5:IN std_logic;  
        D6:IN std_logic;  
        D7:IN std_logic;  
        D8:IN std_logic;  
        D9:IN std_logic;  
        FCLK:IN std_logic;  
        PCLK:IN std_logic;  
        RESET:IN std_logic
```

```

    );
END COMPONENT;
uut:OSER10
    GENERIC MAP (GSREN=>"false",
                  LSREN=>"true"
    )
    PORT MAP (
        Q=>Q,
        D0=>D0,
        D1=>D1,
        D2=>D2,
        D3=>D3,
        D4=>D4,
        D5=>D5,
        D6=>D6,
        D7=>D7,
        D8=>D8,
        D9=>D9,
        FCLK=>FCLK,
        PCLK=>PCLK,
        RESET=>RESET
    );

```

### 4.3.6 OVIDEO

#### プリミティブの紹介

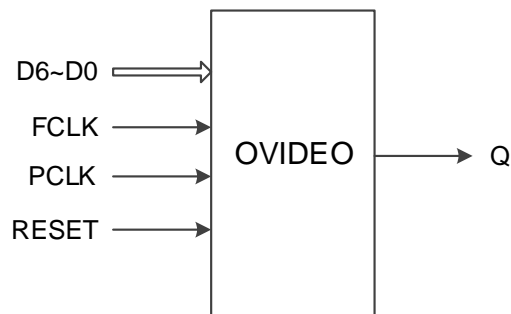
OVIDEO(7 to 1 Serializer)は、7 ビットの平行入力、1 ビットのシリアル出力のシリアライザです。

#### 機能の説明

OVIDEO モードでは、データは 7:1 シリアライズされます。PCLK は通常、FCLK の分周によって得られます： $f_{PCLK} = 1/3.5 f_{FCLK}$ 。

## ポート図

図 4-26 OVIDEO のポート図



## ポートの説明

表 4-35 OVIDEO のポートの説明

ポート名	I/O	説明
D6~D0	入力	OVIDEOデータ入力信号
FCLK	入力	高速クロック入力信号
PCLK	入力	マスタークロック入力信号
RESET	入力	非同期リセット入力、アクティブ High
Q	出力	OVIDEOデータ出力

## パラメータの説明

表 4-36 OVIDEO のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
GSREN	"false", "true"	"false"	グローバルリセットGSRを有効にします
LSREN	"false", "true"	"true"	ローカルリセットRESETを有効にします

## 接続ルール

Q は、OBUF に直接接続するか、IODELAY モジュールを介してその入力ポート DI に接続します。

## プリミティブのインスタンス化

プリミティブを直接インスタンス化するか、IP Core Generator で生成できます。詳しくは、[5 IP の呼び出し](#)を参照してください。

## Verilog でのインスタンス化：

```
OVIDEO uut(
    .Q(Q),
```

```
.D0(D0),  
.D1(D1),  
.D2(D2),  
.D3(D3),  
.D4(D4),  
.D5(D5),  
.D6(D6),  
.PCLK(PCLK),  
.FCLK(FCLK),  
.RESET(RESET)  
);  
defparam uut.GSREN="false";  
defparam uut.LSREN="true";
```

**VHDL** でのインスタンス化 :

```
COMPONENT OVIDEO  
    GENERIC (GSREN:string:="false";  
             LSREN:string:="true"  
    );  
    PORT(  
        Q:OUT std_logic;  
        D0:IN std_logic;  
        D1:IN std_logic;  
        D2:IN std_logic;  
        D3:IN std_logic;  
        D4:IN std_logic;  
        D5:IN std_logic;  
        D6:IN std_logic;  
        FCLK:IN std_logic;  
        PCLK:IN std_logic;  
        RESET:IN std_logic  
    );  
END COMPONENT;  
uut:OVIDEO  
    GENERIC MAP (GSREN=>"false",  
                 LSREN=>"true"
```

```
)  
    PORT MAP (  
        Q=>Q,  
        D0=>D0,  
        D1=>D1,  
        D2=>D2,  
        D3=>D3,  
        D4=>D4,  
        D5=>D5,  
        D6=>D6,  
        FCLK=>FCLK,  
        PCLK=>PCLK,  
        RESET=>RESET  
    );
```

4.3.7 OSER16

プリミティブの紹介

OSER16(16 to 1 Serializer)は、16 ビットのパラレル入力、1 ビットのシリアル出力のシリアライザです。

サポートされるデバイス

表 4-37 OSER16 対応デバイス

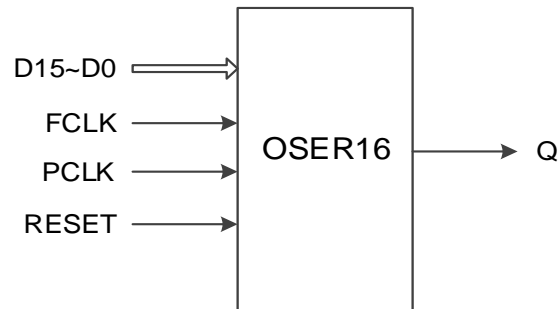
ファミリー	シリーズ	デバイス
LittleBee ファミリー	GW1N	GW1N-1S, GW1N-9, GW1N-9C, GW1N-2, GW1N-1P5, GW1N-2B, GW1N-1P5B
	GW1NR	GW1NR-9, GW1NR-9C, GW1NR-2, GW1NR-2B
	GW1NS	GW1NS-4, GW1NS-4C
	GW1NSER	GW1NSER-4C
	GW1NSR	GW1NSR-4, GW1NSR-4C

機能の説明

OSER16 モードでは、データは 16:1 シリアライズされます。PCLK は通常、FCLK の分周によって得られます： $f_{PCLK} = 1/8 f_{FCLK}$ 。

## ポート図

図 4-27 OSER16 のポート図



## ポートの説明

表 4-38 OSER16 のポートの説明

ポート名	I/O	説明
D15~D0	入力	OSER16データ入力信号
FCLK	入力	高速クロック入力信号
PCLK	入力	マスタークロック入力信号
RESET	入力	非同期リセット入力、アクティブHigh
Q	出力	OSER16データ出力信号

## パラメータの説明

表 4-39 OSER16 のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
GSREN	"false", "true"	"false"	グローバルリセットGSRを有効にします
LSREN	"false", "true"	"true"	ローカルリセットRESETを有効にします

## 接続ルール

Q は、OBUF に直接接続するか、IODELAY モジュールを介してその入力ポート DI に接続します。

## プリミティブのインスタンス化

プリミティブを直接インスタンス化するか、IP Core Generator で生成できます。詳しくは、[5 IP の呼び出し](#)を参照してください。

## Verilog でのインスタンス化：

```

OSER16 uut(
    .Q(Q),
    .D0(D0),

```



```

.D1(D1),
.D2(D2),
.D3(D3),
.D4(D4),
.D5(D5),
.D6(D6),
.D7(D7),
.D8(D8),
.D9(D9),
.D10(D10),
.D11(D11),
.D12(D12),
.D13(D13),
.D14(D14),
.D15(D15),
.PCLK(PCLK),
.FCLK(FCLK),
.RESET(RESET)
);
defparam uut.GSREN="false";
defparam uut.LSREN="true";

```

**VHDL** でのインスタンス化 :

```

COMPONENT OSER16
    GENERIC (GSREN:string:="false";
             LSREN:string:="true"
    );
    PORT(
        Q:OUT std_logic;
        D0:IN std_logic;
        D1:IN std_logic;
        D2:IN std_logic;
        D3:IN std_logic;
        D4:IN std_logic;
        D5:IN std_logic;
        D6:IN std_logic;

```

```
D7:IN std_logic;
D8:IN std_logic;
D9:IN std_logic;
D10:IN std_logic;
D11:IN std_logic;
D12:IN std_logic;
D13:IN std_logic;
D14:IN std_logic;
D15:IN std_logic;
FCLK:IN std_logic;
PCLK:IN std_logic;
RESET:IN std_logic
);
END COMPONENT;
uut:OSER16
    GENERIC MAP (GSREN=>"false",
                  LSREN=>"true"
    )
    PORT MAP (
        Q=>Q,
        D0=>D0,
        D1=>D1,
        D2=>D2,
        D3=>D3,
        D4=>D4,
        D5=>D5,
        D6=>D6,
        D7=>D7,
        D8=>D8,
        D9=>D9,
        D10=>D10,
        D11=>D11,
        D12=>D12,
        D13=>D13,
        D14=>D14,
```

```

D15=>D15,
FCLK=>FCLK,
PCLK=>PCLK,
RESET=>RESET
);

```

### 4.3.8 ODDR\_MEM

#### プリミティブの紹介

ODDR\_MEM(Dual Data Rate Output with Memory)は、メモリ付きのダブル・データ・レートの実現します。

#### サポートされるデバイス

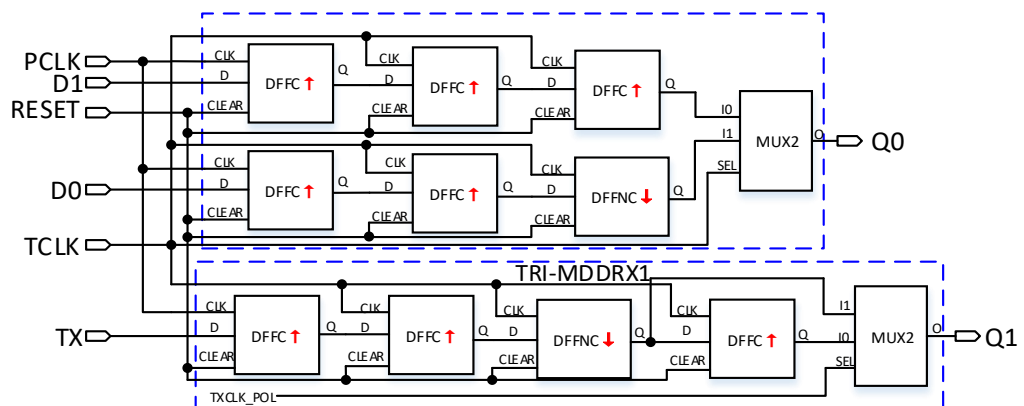
表 4-40 ODDR\_MEM 対応デバイス

ファミリー	シリーズ	デバイス
Arora	GW2A	GW2A-18, GW2A-18C, GW2A-55, GW2A-55C
	GW2AN	GW2AN-55C
	GW2AR	GW2AR-18, GW2AR-18C
	GW2ANR	GW2ANR-18C

#### 機能の説明

ODDR\_MEM モードは、FPGA からダブル・データ・レートの信号を転送するために使用されます。ODDR とは異なり、ODDR\_MEM には DQS が必要です。TCLK は DQS の出力信号 DQSW0 または DQSW270 に接続され、データは TCLK のクロックエッジに基づいて ODDR\_MEM から出力されます。ODDR\_MEM の Q0 は、ダブル・データ・レートのデータ出力であり、Q1 は、Q0 に接続される IOBUF/TBUF の OEN 信号に使用されます。そのブロック図を図 4-28 に示します。

図 4-28 ODDR\_MEM のブロック図

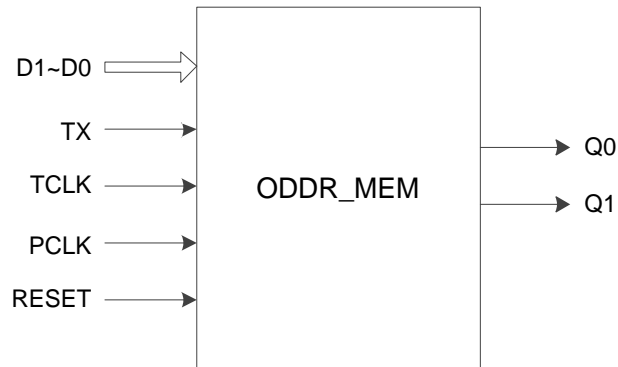


PCLK と TCLK の周波数関係は： $f_{PCLK} = f_{TCLK}$ 。

PCLK と TCLK の間には一定の位相関係があり、その位相関係は DQS の DLLSTEP 値および WSTEP 値により決定できます。

ポート図

図 4-29 ODDR\_MEM のポート図



ポートの説明

表 4-41 ODDR\_MEM のポートの説明

ポート名	I/O	説明
D1~D0	入力	ODDR_MEMデータ入力信号
TX	入力	TRI-MDDR1を通じてQ1を生成
TCLK	入力	DQSモジュールのDQSW0またはDQSW270からのクロック入力信号
PCLK	入力	マスタークロック入力信号
RESET	入力	非同期リセット入力、アクティブHigh
Q0	出力	ODDR_MEMデータ出力
Q1	出力	ODDR_MEMトライステート・イネーブル信号。 Q0に接続されるIOBUF/TBUFのOEN信号に接続するか、フローティングのままにします。

パラメータの説明

表 4-42 ODDR\_MEM のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
GSREN	"false", "true"	"false"	グローバルリセットGSRを有効にします
LSREN	"false", "true"	"true"	ローカルリセットRESETを有効にします
TXCLK_POL	1'b0, 1'b1	1'b0	Q1の出力クロックの極性の制御 ● 1'b0 : データは立ち上がりエッジで出力されます。

パラメータ名	値の範囲	デフォルト値	説明
			<ul style="list-style-type: none"> <li>● 1'b1 : データは立ち下がりエッジで出力されます。</li> </ul>
TCLK_SOURCE	"DQSW", "DQSW270"	"DQSW"	TCLKのソースの選択 <ul style="list-style-type: none"> <li>● "DQSW" : DQS モジュールの DQSW0 から。</li> <li>● DQSW270" : DQS モジュールの DQSW270 から。</li> </ul>

#### 接続ルール

- Q0 は、OBUF に直接接続するか、IODELAY モジュールを介してその入力ポート DI に接続します。
- Q1 は、Q0 に接続される IOBUF/TBUF の OEN 信号に接続するか、フローティングのままにする必要があります。
- TCLK は、DQS モジュールの DQSW0 または DQSW270 に接続し、対応するパラメータを構成する必要があります。

#### プリミティブのインスタンス化

##### Verilog でのインスタンス化 :

```

ODDR_MEM oddr_mem_inst(
    .Q0(q0),
    .Q1(q1),
    .D0(d0),
    .D1(d1),
    .TX(tx),
    .TCLK(tclk),
    .PCLK(pclk),
    .RESET(reset)
);
defparam uut.GSREN="false";
defparam uut.LSREN="true";
defparam uut.TCLK_SOURCE="DQSW";
defparam uut.TXCLK_POL=1'b0;

```

##### VHDL でのインスタンス化 :

```

COMPONENT ODDR_MEM
    GENERIC (GSREN:string:="false";
             LSREN:string:="true";
             TXCLK_POL:bit:='0';

```

```

        TCLK_SOURCE:string:="DQSW"
    );
    PORT(
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        D0:IN std_logic;
        D1:IN std_logic;
        TX:IN std_logic;
        TCLK:IN std_logic;
        PCLK:IN std_logic;
        RESET:IN std_logic
    );
END COMPONENT;
 uut:ODDR_MEM
    GENERIC MAP (GSREN=>"false",
                  LSREN=>"true",
                  TXCLK_POL=>'0',
                  TCLK_SOURCE=>"DQSW"
    )
    PORT MAP (
        Q0=>q0,
        Q1=>q1,
        D0=>d0,
        D1=>d1,
        TX=>tx,
        TCLK=>tclk,
        PCLK=>pclk,
        RESET=>reset
    );

```

### 4.3.9 OSER4\_MEM

#### プリミティブの紹介

OSER4\_MEM(4 to 1 Serializer with Memory)は、メモリ付きの 4:1 シリアライザです。

## サポートされるデバイス

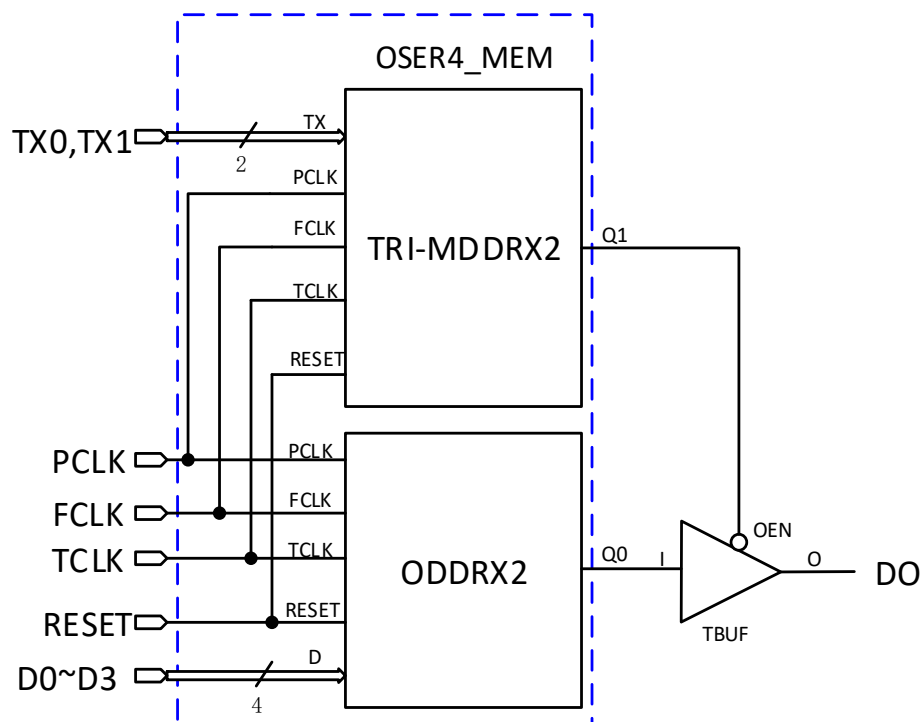
表 4-43 OSER4\_MEM 対応デバイス

ファミリー	シリーズ	デバイス
Arora ファミリー	GW2A	GW2A-18, GW2A-18C, GW2A-55, GW2A-55C
	GW2AN	GW2AN-55C
	GW2AR	GW2AR-18, GW2AR-18C
	GW2ANR	GW2ANR-18C

## 機能の説明

OSER4\_MEM モードでは、データは 4 : 1 シリアルライズされます。OSER4 とは異なり、OSER4\_MEM には DQS が必要です。TCLK は DQS の出力信号 DQSW0 または DQSW270 に接続され、データは TCLK のクロックエッジに基づいて OSER4\_MEM から出力されます。OSER4\_MEM の Q0 は、データのシリアル出力で、Q1 は、Q0 に接続される IOBUF/TBUF の OEN 信号に使用されます。そのブロック図を図 4-30 に示します。

図 4-30 OSER4\_MEM のブロック図



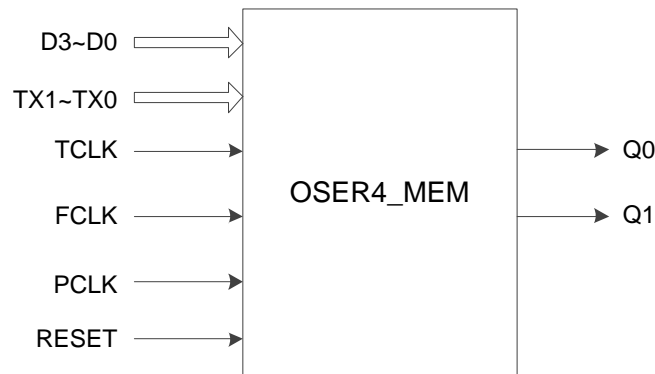
PCLK、FCLK、および TCLK の周波数関係は：

$$f_{PCLK} = 1/2 f_{FCLK} = 1/2 f_{TCLK}。$$

FCLK と TCLK の間には一定の位相関係があり、その位相関係は DQS の DLLSTEP 値および WSTEP 値により決定できます。

## ポート図

図 4-31 OSER4\_MEM のポート図



## ポートの説明

表 4-44 OSER4\_MEM のポートの説明

ポート名	I/O	説明
D3~D0	入力	OSER4_MEMデータ入力信号
TX1~TX0	入力	TRI-MDDR2を通じてQ1を生成
TCLK	入力	DQSモジュールのDQSW0またはDQSW270からのクロック入力信号
FCLK	入力	高速クロック入力信号
PCLK	入力	マスタークロック入力信号
RESET	入力	非同期リセット入力、アクティブHigh
Q0	出力	OSER4_MEMデータ出力信号
Q1	出力	OSER4_MEMトライステート・イネーブル信号。Q0に接続されるIOBUF/TBUFのOEN信号に接続するか、フローティングのままにすることができます。

## パラメータの説明

表 4-45 OSER4\_MEM のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
GSREN	"false", "true"	"false"	グローバルリセットGSRを有効にします
LSREN	"false", "true"	"true"	ローカルリセットRESETを有効にします
TXCLK_POL	1'b0, 1'b1	1'b0	Q1の出力クロックの極性の制御 ● 1'b0 : データは立ち上がりエッジで出力されます。



パラメータ名	値の範囲	デフォルト値	説明
			<ul style="list-style-type: none"> <li>● 1'b1 : データは立ち下がりエッジで出力されます。</li> </ul>
TCLK_SOURCE	"DQSW", "DQSW270"	" DQSW "	TCLKのソースの選択 <ul style="list-style-type: none"> <li>● "DQSW" : DQS モジュールの DQSW0 から。</li> <li>● "DQSW270": DQS モジュールの DQSW270 から</li> </ul>
HWL	"false", "true"	"false"	OSER4_MEMデータd_up0/1のタイミング関係の制御 <ul style="list-style-type: none"> <li>● "false" : d_up1 は d_up0 より 1 サイクル先です。</li> <li>● "true" : d_up1 と d_up0 のタイミングは同じです。</li> </ul>

#### 接続ルール

- Q0 は、OBUF に直接接続するか、IODELAY モジュールを介してその入力ポート DI に接続します。
- Q1 は、Q0 に接続される IOBUF/TBUF の OEN 信号に接続するか、フローティングのままにする必要があります。
- TCLK は、DQS モジュールの DQSW0 または DQSW270 に接続し、対応するパラメータを構成する必要があります。

#### プリミティブのインスタンス化

##### Verilog でのインスタンス化 :

```
OSER4_MEM oser4_mem_inst(
    .Q0(q0),
    .Q1(q1),
    .D0(d0),
    .D1(d1),
    .D2(d2),
    .D3(d3),
    .TX0(tx0),
    .TX1(tx1),
    .TCLK(tclk),
    .FCLK(fclk),
    .PCLK(pclk),
    .RESET(reset)
```

```
);  
defparam uut.GSREN="false";  
defparam uut.LSREN="true";  
defparam uut.HWL="false";  
defparam uut.TCLK_SOURCE="DQSW";  
defparam uut.TXCLK_POL=1'b0;
```

**VHDL** でのインスタンス化 :

```
COMPONENT OSER4_MEM  
    GENERIC (GSREN:string:="false";  
             LSREN:string:="true";  
             HWL:string:="false";  
             TXCLK_POL:bit:='0';  
             TCLK_SOURCE:string:="DQSW"  
    );  
    PORT(  
        Q0:OUT std_logic;  
        Q1:OUT std_logic;  
        D0:IN std_logic;  
        D1:IN std_logic;  
        D2:IN std_logic;  
        D3:IN std_logic;  
        TX0:IN std_logic;  
        TX1:IN std_logic;  
        TCLK:IN std_logic;  
        FCLK:IN std_logic;  
        PCLK:IN std_logic;  
        RESET:IN std_logic  
    );  
END COMPONENT;  
uut:OSER4_MEM  
    GENERIC MAP (GSREN=>"false",  
                 LSREN=>"true",  
                 HWL=>"false",  
                 TXCLK_POL=>'0',  
                 TCLK_SOURCE=>"DQSW"
```

```

)
PORT MAP (
Q0=>q0,
Q1=>q1,
D0=>d0,
D1=>d1,
D2=>d2,
D3=>d3,
TX0=>tx0,
TX1=>tx1,
TCLK=>tclk,
FCLK=>fclk,
PCLK=>pclk,
RESET=>reset
);
```

4.3.10 OSER8\_MEM

プリミティブの紹介

OSER8\_MEM(8 to 1 Serializer with Memory)は、メモリ付きの 8:1 シリアライザです。

サポートされるデバイス

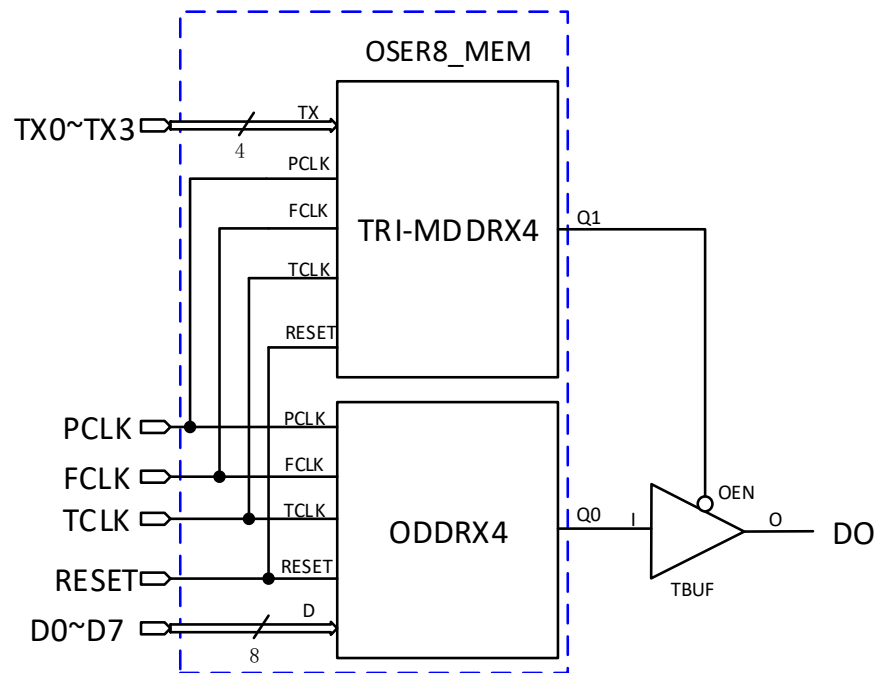
表 4-46 OSER8\_MEM 対応デバイス

ファミリー	シリーズ	デバイス
Arora ファミリー	GW2A	GW2A-18, GW2A-18C, GW2A-55, GW2A-55C
	GW2AN	GW2AN-55C
	GW2AR	GW2AR-18, GW2AR-18C
	GW2ANR	GW2ANR-18C

機能の説明

OSER8\_MEM モードでは、データは 8:1 シリアライズされます。OSER8 とは異なり、OSER8\_MEM には DQS が必要です。TCLK は DQS の出力信号 DQSW0 または DQSW270 に接続され、データは TCLK のクロックエッジに基づいて OSER8\_MEM から出力されます。OSER8\_MEM の Q0 は、データのシリアル出力で、Q1 は、Q0 に接続される IOBUF/TBUF の OEN 信号に使用されます。そのブロック図を図 4-32 に示します。

図 4-32 OSER8\_MEM のブロック図



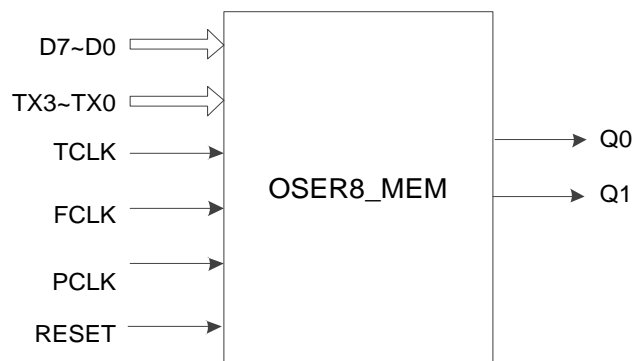
PCLK、FCLK、および TCLK の周波数関係は：

$$f_{PCLK} = 1/4 f_{FCLK} = 1/4 f_{TCLK}。$$

FCLK と TCLK の間には一定の位相関係があり、その位相関係は DQS の DLLSTEP 値および WSTEP 値により決定できます。

ポート図

図 4-33 OSER8\_MEM のポート図



ポートの説明

表 4-47 OSER8\_MEM のポートの説明

ポート名	I/O	説明
D7~D0	入力	OSER8_MEMデータ入力信号
TX3~TX0	入力	TRI-MDDR4を通じてQ1を生成
TCLK	入力	DQSモジュールのDQSW0またはDQSW270から

ポート名	I/O	説明
		のクロック入力信号
FCLK	入力	高速クロック入力信号
PCLK	入力	マスタークロック入力信号
RESET	入力	非同期リセット入力、アクティブHigh
Q0	出力	OSER8_MEMデータ出力信号
Q1	出力	OSER8_MEMトライステート・イネーブル信号。Q0に接続されるIOBUF/TBUFのOEN信号に接続するか、フローティングのままにすることができます。

### パラメータの説明

表 4-48 OSER8\_MEM のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
GSREN	"false", "true"	"false"	グローバルリセットGSRを有効にします
LSREN	"false", "true"	"true"	ローカルリセットRESETを有効にします
TXCLK_POL	1'b0, 1'b1	1'b0	Q1の出力クロックの極性の制御 <ul style="list-style-type: none"> <li>● 1'b0 : データは立ち上がりエッジで出力されます。</li> <li>● 1'b1 : データは立ち下がりエッジで出力されます。</li> </ul>
TCLK_SOURCE	"DQSW", "DQSW270"	"DQSW "	TCLKのソースの選択 <ul style="list-style-type: none"> <li>● "DQSW" : DQS モジュールの DQSW0 から。</li> <li>● DQSW270" : DQS モジュールの DQSW270 から。</li> </ul>
HWL	"false", "true"	"false"	OSER8_MEMデータd_up0/1のタイミング関係の制御 <ul style="list-style-type: none"> <li>● "false" : d_up1 は d_up0 より 1 サイクル先です。</li> <li>● "true" : d_up1 と d_up0 のタイミングは同じです。</li> </ul>

### 接続ルール

- Q0 は、OBUF に直接接続するか、IODELAY モジュールを介してその入力ポート DI に接続します。
- Q1 は、Q0 に接続される IOBUF/TBUF の OEN 信号に接続するか、フ

ローテイングのままにする必要があります。

- TCLK は、DQS モジュールの DQSW0 または DQSW270 に接続し、対応するパラメータを構成する必要があります。

#### プリミティブのインスタンス化

##### Verilog でのインスタンス化 :

```
OSER8_MEM oser8_mem_inst(
    .Q0(q0),
    .Q1(q1),
    .D0(d0),
    .D1(d1),
    .D2(d2),
    .D3(d3),
    .D4 (d4),
    .D5 (d5),
    .D6 (d6),
    .D7 (d7),
    .TX0 (tx0),
    .TX1 (tx1),
    .TX2 (tx2),
    .TX3 (tx3),
    .TCLK (tclk),
    .FCLK (fclk),
    .PCLK (pclk),
    .RESET(reset)
);
defparam uut.GSREN="false";
defparam uut.LSREN ="true";
defparam uut.HWL ="false";
defparam uut.TCLK_SOURCE ="DQSW";
defparam uut.TXCLK_POL=1'b0;
```

##### VHDL でのインスタンス化 :

```
COMPONENT OSER8_MEM
    GENERIC (GSREN:string:="false";
             LSREN:string:="true";
             HWL:string:="false";
```

```
        TXCLK_POL:bit:='0';
        TCLK_SOURCE:string:="DQSW"
    );
    PORT(
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        D0:IN std_logic;
        D1:IN std_logic;
        D2:IN std_logic;
        D3:IN std_logic;
        D4:IN std_logic;
        D5:IN std_logic;
        D6:IN std_logic;
        D7:IN std_logic;
        TX0:IN std_logic;
        TX1:IN std_logic;
        TX2:IN std_logic;
        TX3:IN std_logic;
        TCLK:IN std_logic;
        FCLK:IN std_logic;
        PCLK:IN std_logic;
        RESET:IN std_logic
    );
END COMPONENT;
uut:OSER8_MEM
    GENERIC MAP (GSREN=>"false",
                  LSREN=>"true",
                  HWL=>"false",
                  TXCLK_POL=>'0',
                  TCLK_SOURCE=>"DQSW"
    )
    PORT MAP (
        Q0=>q0,
        Q1=>q1,
        D0=>d0,
```

```

D1=>d1,
D2=>d2,
D3=>d3,
D4=>d4,
D5=>d5,
D6=>d6,
D7=>d7,
TX0=>tx0,
TX1=>tx1,
TX2=>tx2,
TX3=>tx3,
TCLK=>tclk,
FCLK=>fclk,
PCLK=>pclk,
RESET=>reset
);

```

## 4.4 遅延モジュール

### 4.4.1 IODELAY

#### プリミティブの紹介

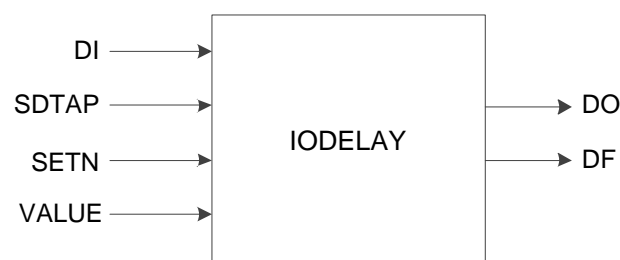
IODELAY(Input/Output delay)は、IO ブロックに含まれるプログラムブルな遅延モジュールです。

#### 機能の説明

各 IO には、合計 128(0~127)の遅延設定を提供する IODELAY モジュールが含まれています。GW1N シリーズ FPGA の場合、ステップごとの遅延時間は約 30ps であり、GW2A シリーズ FPGA の場合、ステップごとの遅延時間は約 18ps(18K デバイス)または 22ps(55K デバイス)です。IODELAY は、I/O ロジックの入力または出力に使用できますが、同時に使用することはできません。

#### ポート図

図 4-34 IODELAY のポート図





## ポートの説明

表 4-49 IODELAY のポートの説明

ポート名	I/O	説明
DI	入力	データ入力信号
SDTAP	入力	静的/動的遅延の選択 ● 0 : 静的遅延を使用します ● 1 : 遅延を動的に調整します
SETN	入力	遅延の動的調整の方向を設定します ● 0: 遅延を増やします ● 1: 遅延を減らします
VALUE	入力	VALUEの立ち下がリエッジで遅延が動的に調整されます。パルスごとに1遅延ステップ移動します
DO	出力	データ出力信号
DF	出力	動的遅延調整のunder-flowまたはover-flowを示すフラグ

## パラメータの説明

表 4-50 IODELAY のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
C_STATIC_DLY	0~127	0	静的遅延ステップ数の制御

## プリミティブのインスタンス化

Verilog でのインスタンス化 :

```

IODELAY iodelay_inst(
    .DO(dout),
    .DF(df),
    .DI(di),
    .SDTAP(sdtap),
    .SETN(setn),
    .VALUE(value)
);
defparam iodelay_inst.C_STATIC_DLY=0;

```

VHDL でのインスタンス化 :

```

COMPONENT IODELAY

```

```

        GENERIC (C_STATIC_DLY:integer:=0
    );
    PORT(
        DO:OUT std_logic;
        DF:OUT std_logic;
        DI:IN std_logic;
        SDTAP:IN std_logic;
        SETN:IN std_logic;
        VALUE:IN std_logic
    );
END COMPONENT;
 uut:IODELAY
    GENERIC MAP (C_STATIC_DLY=>0
    )
    PORT MAP (
        DO=>dout,
        DF=>df,
        DI=>di,
        SDTAP=>sdtap,
        SETN=>setn,
        VALUE=>value
    );

```

4.4.2 IODELAYC

プリミティブの紹介

IODELAYC(Input/Output delay)は、IO ブロックに含まれるプログラマブルな遅延モジュールです。

サポートされるデバイス

表 4-51 IODELAYC 対応デバイス

ファミリー	シリーズ	デバイス
LittleBee	GW1N	GW1N-9C
	GW1NR	GW1NR-9C

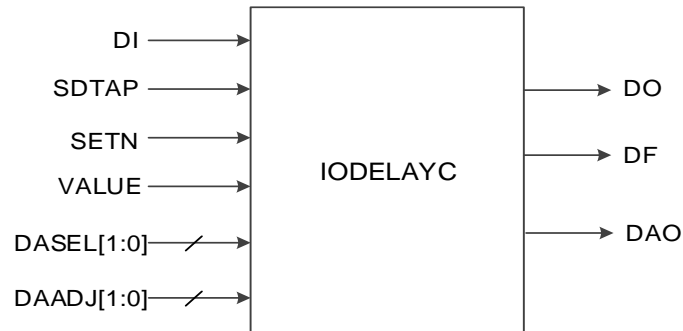
機能の説明

各 IO には、合計 128(0～127)の遅延設定を提供する IODELAYC モジュールが含まれています。IODELAY と比較して、より多くの遅延調整オ

プッシュが追加されています。IODELAYC は、I/O ロジックの入力にのみ使用できます。

### ポート図

図 4-35 IODELAYC のポート図



### ポートの説明

表 4-52 IODELAYC のポートの説明

ポート名	I/O	説明
DI	入力	データ入力信号
SDTAP	入力	静的/動的遅延の選択 ● 0 : 静的遅延を使用します ● 1 : 遅延を動的に調整します
SETN	入力	遅延の動的調整の方向を設定します ● 0: 遅延を増やします ● 1: 遅延を減らします
VALUE	入力	VALUEの立ち下がリエッジで遅延が動的に調整されます。パルスごとに1遅延ステップ移動します
DASEL[1:0]	入力	DAO遅延モードの動的制御
DAADJ[1:0]	入力	DOに対するDAOの遅延値の動的制御
DO	出力	データ出力信号
DAO	出力	データ遅延調整出力
DF	出力	動的遅延調整のunder-flowまたはover-flowを示すフラグ

### パラメータの説明

表 4-53 IODELAYC のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
C_STATIC_DLY	0~127	0	静的遅延ステップ数の制御

パラメータ名	値の範囲	デフォルト値	説明
DYN_DA_SEL	“true” / “false”	false	<ul style="list-style-type: none"> <li>● false : DA_SEL パラメータで DAO 遅延モードを静的に制御</li> <li>● true : DASEL 信号で DAO 遅延モードを動的に制御</li> </ul>
DA_SEL	2'b00~2'b11	2'b00	DAO遅延モードの静的制御

### プリミティブのインスタンス化

#### Verilog でのインスタンス化 :

```

IODELAYC iodelayc_inst(
    .DO(dout),
    .DAO(douta),
    .DF(df),
    .DI(di),
    .SDTAP(sdtap),
    .SETN(setn),
    .VALUE(value),
    .DASEL(dasel),
    .DAADJ(daadj)
);
defparam iodelayc_inst.C_STATIC_DLY=0;
defparam iodelayc_inst.DYN_DA_SEL="true";
defparam iodelayc_inst.DA_SEL=2'b01;

```

#### VHDL でのインスタンス化 :

```

COMPONENT IODELAYC
    GENERIC (C_STATIC_DLY:integer:=0;
             DYN_DA_SEL:string:="false";
             DA_SEL:bit_vector:="00"
    );
    PORT(
        DO:OUT std_logic;
        DAO:OUT std_logic;
        DF:OUT std_logic;
        DI:IN std_logic;

```

```
SDTAP:IN std_logic;
SETN:IN std_logic;
VALUE:IN std_logic;
DASEL : IN std_logic_vector(1 downto 0);
DAADJ : IN std_logic_vector(1 downto 0)
);
END COMPONENT;
uut:IODELAYC
    GENERIC MAP (C_STATIC_DLY=>0,
                  DYN_DA_SEL=>"true",
                  DA_SEL=>"01"
    )
    PORT MAP (
        DO=>dout,
        DAO=>dout,
        DF=>df,
        DI=>di,
        SDTAP=>sdtap,
        SETN=>setn,
        VALUE=>value,
        DASEL=>dasel,
        DAADJ=>daadj
    );
```

4.4.3 IODELAYB

プリミティブの紹介

IODELAYB(Input/Output delay)は、IO ブロックに含まれるプログラマブルな遅延モジュールです。

サポートされるデバイス

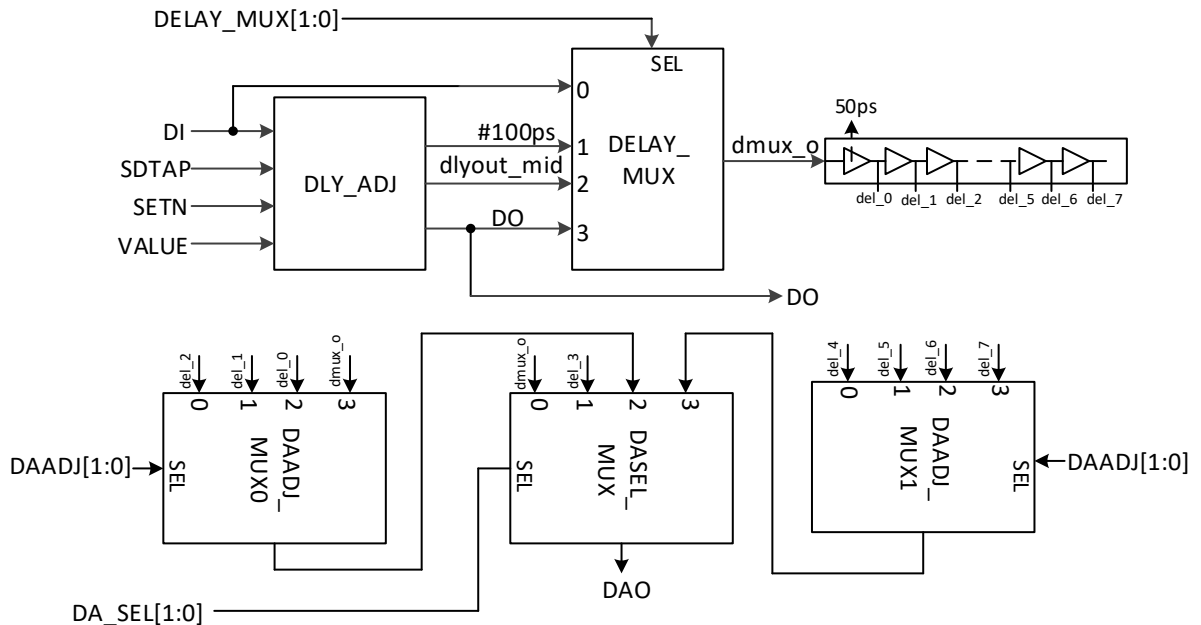
表 4-54 IODELAYB 対応デバイス

ファミリー	シリーズ	デバイス
LittleBeeファミリー	GW1N	GW1N-2, GW1N-1P5, GW1N-2B, GW1N-1P5B
	GW1NR	GW1NR-2, GW1NR-2B
	GW1NZ-2	GW1NZ-2B, GW1NZ-2C

## 機能の説明

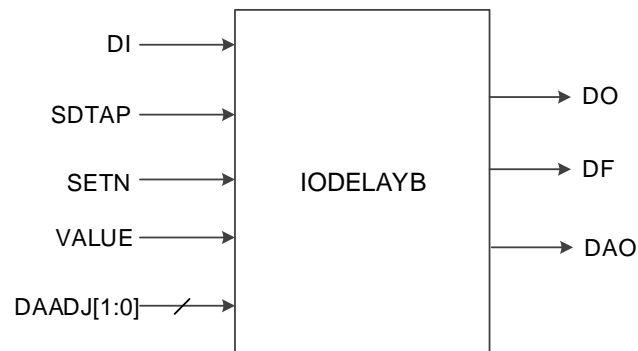
各 IO には、合計 128(0~127)の遅延設定を提供する IODELAYB モジュールが含まれています。IODELAY と比較して、より多くの遅延調整オプションが追加されています。そのブロック図を図 4-36 に示します。IODELAYB は、I/O ロジックの入力にのみ使用できます。

図 4-36 IODELAYB の構造



## ポート図

図 4-37 IODELAYB のポート図



## ポートの説明

表 4-55 IODELAYB のポートの説明

ポート名	I/O	説明
DI	入力	データ入力信号
SDTAP	入力	静的/動的遅延の選択 ● 0 : 静的遅延を使用します ● 1 : 遅延を動的に調整します
SETN	入力	遅延の動的調整の方向を設定します

ポート名	I/O	説明
		<ul style="list-style-type: none"> <li>● 0: 遅延を増やします</li> <li>● 1: 遅延を減らします</li> </ul>
VALUE	入力	VALUEの立ち下がりエッジで遅延が動的に調整されます。パルスごとに1遅延ステップ移動します。
DAADJ[1:0]	入力	DOに対するDAOの遅延値の動的制御
DO	出力	データ出力信号
DAO	出力	データ遅延調整出力
DF	出力	動的遅延調整のunder-flowまたはover-flowを示すフラグ。

### パラメータの説明

表 4-56 IODELAYB のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
C_STATIC_DLY	0~127	0	静的遅延ステップ数の制御
DELAY_MUX	2'b00~2'b11	2'b00	Delay MUXの選択 <ul style="list-style-type: none"> <li>● 2'b00:dmux_o=DI;</li> <li>● 2'b01:#100ps dmux_o=DI</li> <li>● 2'b10:dmux_o=dlyout_mid</li> <li>● 2'b11:dmux_o=DO</li> </ul>
DA_SEL	2'b00~2'b11	2'b00	DAO遅延モードの静的制御

注記：

IODELAYB を使用する場合、パラメータ DELAY\_MUX と DA\_SEL の関係は次のとおりです。

- DELAY\_MUX:2/3 -> DA\_SEL:0/1。つまり、DELAY\_MUX が 2 または 3 の場合、DA\_SEL は 0 または 1 になります。
- DELAY\_MUX:0/1 -> DA\_SEL:0/2/3。つまり、DELAY\_MUX が 0 または 1 の場合、DA\_SEL は 0、2、または 3 になります。

### 接続ルール

DO は IDDR/IDES に接続できず、DAO は IDDR/IDES のデータ入力にのみ接続できます。

### プリミティブのインスタンス化

**Verilog** でのインスタンス化：

```
IODELAYB iodelayb_inst(
    .DO(dout),
    .DAO(douta),
    .DF(df),
    .DI(di),
```

```

        .SDTAP(sdtap),
        .SETN(setn),
        .VALUE(value),
        .DAADJ(daadj)
    );
    defparam iodelayb_inst.C_STATIC_DLY=0;
    defparam iodelayb_inst.DELAY_MUX = 2'b00;
    defparam iodelayb_inst.DA_SEL=2'b00;

```

**VHDL** でのインスタンス化 :

```

COMPONENT IODELAYB
    GENERIC (C_STATIC_DLY:integer:=0;
             DELAY_MUX : bit_vector := "00";
             DA_SEL:bit_vector:= "00"
    );
    PORT(
        DO:OUT std_logic;
        DAO:OUT std_logic;
        DF:OUT std_logic;
        DI:IN std_logic;
        SDTAP:IN std_logic;
        SETN:IN std_logic;
        VALUE:IN std_logic;
        DAADJ : IN std_logic_vector(1 downto 0)
    );
END COMPONENT;

uut:IODELAYB
    GENERIC MAP (C_STATIC_DLY=>0,
                 DELAY_MUX =>"00",
                 DA_SEL=>"00"
    )
    PORT MAP (
        DO=>dout,
        DAO=>douta,
        DF=>df,
        DI=>di,

```



```
SDTAP=>sdtap,  
SETN=>setn,  
VALUE=>value,  
DAADJ=>daadj  
);
```

4.5 サンプリングモジュール

プリミティブの紹介

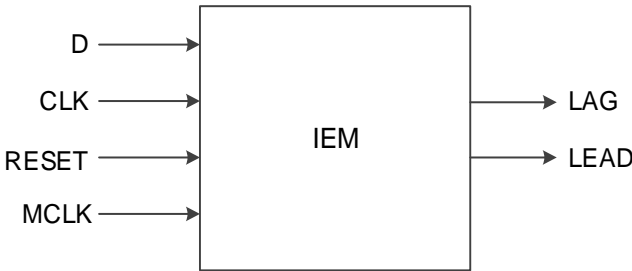
IEM(Input Edge Monitor)は、IO モジュールに含まれるサンプリングモジュールです。

機能の説明

IEM は、データエッジをサンプリングするために使用され、遅延モジュールと併用して動的サンプリングウィンドウを調整できます。DDR モードで使用されます。

ポート図

図 4-38 IEM のポート図



ポートの説明

表 4-57 IEM のポートの説明

ポート名	I/O	説明
D	入力	データ入力信号
CLK	入力	クロック入力信号
RESET	入力	非同期リセット入力、アクティブHigh
MCLK	入力	IEM検出クロック。ユーザーロジックから取得でき、出力フラグに使用されます。
LAG	出力	IEMエッジ比較LAG出力フラグ
LEAD	出力	IEMエッジ比較LEAD出力フラグ

## パラメータの説明

表 4-58 IEM のパラメータの説明

パラメータ名	値の範囲	デフォルト値	説明
WINSIZE	"SMALL", "MIDSMALL", "MIDLARGE", "LARGE"	"SMALL"	ウィンドウサイズの設定
GSREN	"false", "true"	"false"	グローバルリセットGSRを有効にします
LSREN	"false", "true"	"true"	ローカルリセットRESETを有効にします

## プリミティブのインスタンス化

### Verilog でのインスタンス化 :

```

IEM iem_inst(
    .LAG(lag),
    .LEAD(lead),
    .D(d),
    .CLK(clk),
    .MCLK(mclk),
    .RESET(reset)
);

defparam iodelay_inst.WINSIZE = "SMALL";
defparam iodelay_inst.GSREN = "false";
defparam iodelay_inst.LSREN = "true";

```

### VHDL でのインスタンス化 :

```

COMPONENT IEM
    GENERIC (WINSIZE:string:="SMALL";
             GSREN:string:="false";
             LSREN:string:="true"
    );
    PORT(
        LAG:OUT std_logic;
        LEAD:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic;
        MCLK:IN std_logic;

```

```
        RESET:IN std_logic
    );
END COMPONENT;
 uut:IEM
    GENERIC MAP (WINSIZE=>"SMALL",
                  GSREN=>"false",
                  LSREN=>"true"
    )
    PORT MAP (
        LAG=>lag,
        LEAD=>lead,
        D=>d,
        CLK=>clk,
        MCLK=>mclk,
        RESET=>reset
    );
```

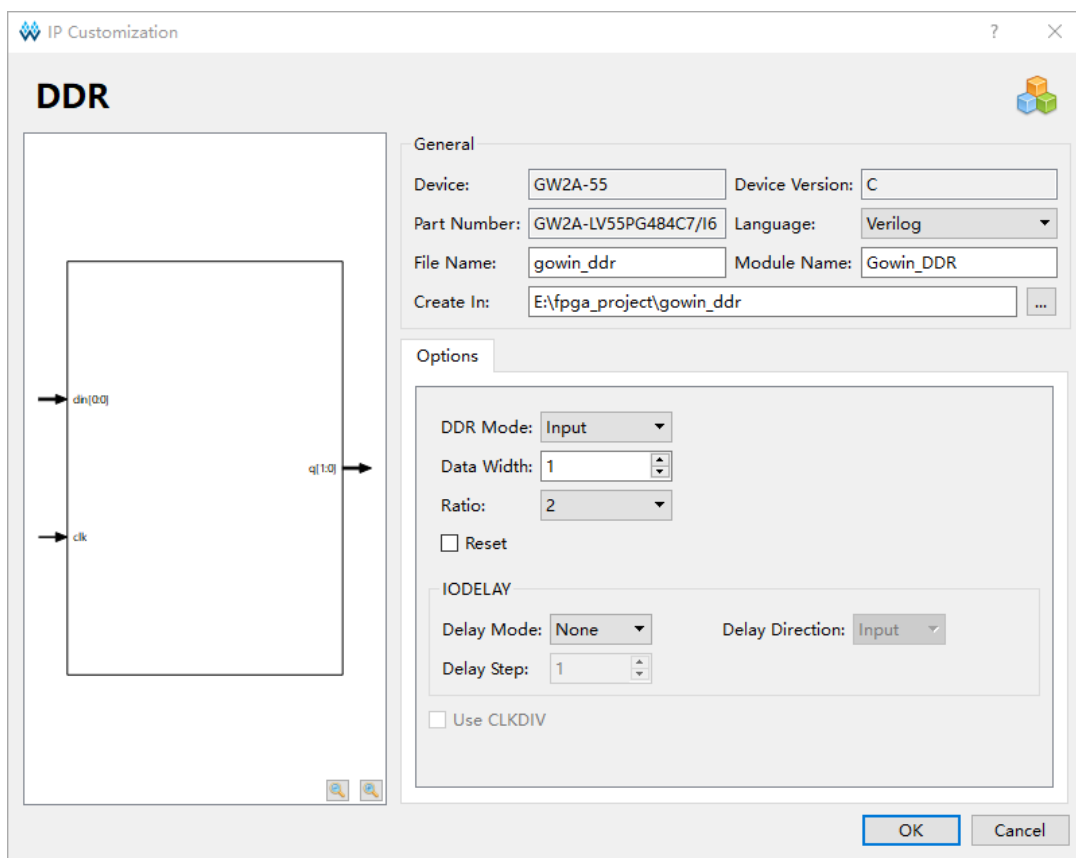
# 5 IP の呼び出し

現在、DDR のみがサポートされます。IP Core Generator の画面で DDR をクリックすると、右側に DDR の概要が表示されます。

## 5.1 IP の構成

IP Core Generator 画面で「DDR」をダブルクリックすると、「IP Customization」ウィンドウがポップアップします。このウィンドウには、「General」構成タブおよびポート図があります(図 5-1)。

図 5-1 DDR IP の構成ウィンドウ



1. General 構成タブ : General 構成タブは、IP ファイルの構成に使用さ

れます。

- **Device** : 対象デバイス。
- **Device Version** : デバイスのバージョン。
- **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. **Options 構成タブ** : **Options** 構成タブは IP のカスタマイズに使用されます(図 5-1)。

- **DDR Mode** : 「**Input**」(入力)、「**Output**」(出力)、「**Tristate**」(トライステート)、および「**Bidirectional**」(双方向)を含む 4 つの DDR モードがあります。
- **Data Width** : DDR のデータ幅(1~64)を構成します。
- **Ratio** : DDR データ変換の比率(2,4,7,8,10,16 を含む)を構成します。
- **Reset** : **Ratio** として 2 が選択されている場合、このオプションを有効にするか無効にするかを選択できます。有効にすると、**IDDRC** または **ODDRC** がインスタンス化されます。
- **IODELAY** : DDR に遅延モジュールを使用するかどうかを構成します。
  - **Delay Mode** : 遅延モードを構成します。「**None**」は **IODELAY** を使用しないことを意味し、「**Dynamic**」は **IODELAY** を使用して遅延ステップ数を動的に調整することを意味し、「**Static**」は **IODELAY** を使用して遅延ステップ数を静的に調整することを意味します。
  - **Delay Step** : 遅延を静的に調整するためのステップ数(1~128)を選択します。
  - **Delay Direction** : **Bidirectional** という **DDR Mode** で **IODELAY** を使用する場合は、**IODELAY** を入力側または出力側に接続す

るかを選択します。

- **Use CLKDIV** : 有効にすると、**CLKDIV** はインスタンス化され、クロック信号 **fclk** は周波数分割されます。**Ratio** が 2 の場合、チェックできません。

3. ポート図 : ポート図は、IP Core の構成結果を表示します(図 5-1)。

## 5.2 生成されるファイル

IP 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\_dds.v」は完全な verilog モジュールです。
- 「gowin\_dds\_tmp.v」は IP のテンプレートファイルです。
- 「gowin\_dds.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。

