Gowin Programmable IO (GPIO)

User Guide

UG289-2.1.2E, 02/22/2023

Revision History

Date	Version	Description
05/17/2016	1.05E	Initial version published.
07/15/2016	1.06E	The graphics standardized.
08/02/2016	1.07E	GW2A series of FPGA Products Data Sheet supported.
10/27/2016	1.08E	GW2A series of FPGA Products Data Sheet supported.
09/01/2017	1.09E	Features of GW1N-6/9 and GW1NR updated.
10/12/2017	1.10E	IDES16/OSER16 related notes added.
12/12/2017	1.2E	● IDDR/ODDR RESET signal removed. ● LVDS description updated. ● Input/output description with memory added.
04/08/2018	1.3E	The chart in Chapter 7 updated.
05/14/2020	1.4E	● GPIO Primitive updated. ● GW1N-6, GW1NR-6 devices deleted.
08/27/2020	1.5E	● The chapter structure modified. ● Chapter 4Input/Output Logic and Chapter 5IP Generation added.
01/07/2021	1.6E	IODELAYB added.
02/02/2021	1.7E	● The description of MIPI_IBUF_HS and MIPI_IBUF_LP added. ● GW2AN-55C and GW1NR-2 devices added.
03/25/2021	1.8E	● GW1NZ-2 removed. ● Devices supported of MIPI_OBUF and MIPI_OBUF_A updated.
06/21/2021	1.9E	● GW1N-2B, GW1N-1P5, GW1N-1P5B, GW1NR-2B, GW2AN-18X and GW2AN-9X added. ● "Help" information removed on IP configuration GUI.
10/21/2021	1.9.1E	The description of GPIO level standard updated.
11/23/2021	1.9.2E	Input logic view optimized.
01/24/2022	2.0E	● The description of input/output buffer improved. ● Sample code formatting adjusted.
05/30/2022	2.0.1E	The description of termination matching resistors for single-ended/differential singals updated.
07/22/2022	2.0.2E	The functional description of OSER4 updated.
08/10/2022	2.0.3E	The device information of LVDS Inout Buffer and Input/Output Logic updated.
11/04/2022	2.1E	● GW1NS-2, GW1NS-2C, GW1NSE-2C, GW1NSR-2, GW1NSR-2C removed. ● Chapter 3.6.14 ELVDS_IBUF_MIPI added.
01/05/2023	2.1.1E	● The configuration box "File" modified to "General" and "Device Version" option added on the IP interface. ● The description of differential buffer configuration updated.
02/22/2023	2.1.2E	The description of slew rate removed.

Contents

List of Figures

List of Tables

1 About This Guide

1.1 Purpose

Gowin Programmable IO (GPIO) User Guide provides descriptions of the level standard, banking of the input/output buffer, and input/output logic functions supported by GOWINSEMI FPGA products. Gowin GPIO architecture and Gowin Software usage are also provided to help you better understand the GPIO functions and rules.

1.2 Related Documents

The latest user guides are available on the GOWINSEMI Website. You can find the related documents at www.gowinsemi.com:

- DS100, GW1N series of FPGA Products Data Sheet
- DS117, GW1NR series of FPGA Products Data Sheet
- DS821, GW1NS series of FPGA Products Data Sheet
- DS841, GW1NZ series of FPGA Products Data Sheet
- DS861, GW1NSR series of FPGA Products Data Sheet
- DS871, GW1NSE series of SecureFPGA Products Data Sheet
- DS881, GW1NSER series of SecureFPGA Products Data Sheet
- DS891, GW1NRF series of Bluetooth FPGA Products Data Sheet
- DS102, GW2A series of FPGA Products Data Sheet
- DS226, GW2AR series of FPGA Products Data Sheet
- DS961, GW2ANR series of FPGA Products Data Sheet
- DS971, GW2AN-18X & 9X Data Sheet
- DS976, GW2AN-55 Data Sheet

1.3 Terminology and Abbreviations

Table 1-1 shows the abbreviations and terminology used in this manual.

Table 1-1 Abbreviations and Terminology

Terminology and Abbreviations	Meaning
Bus Keeper	Bus Keeper
CFU	Configurable Function Unit
CRU	Configurable Routing Unit
DDR	Double Data Rate
DES	Deserializer
ELDO	Emulated LVDS Output
GPIO	Gowin Programmable Input/Output
I/O Buffer	Input/Output Buffer
I/O Logic	Input/Output Logic
IOB	Input/Output Block
Open Drain	Open Drain
SDR	Single Data Rate
SER	Serializer
TLDO	True LVDS Output

1.4 Support and Feedback

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

Website: www.gowinsemi.com

E-mail: support@gowinsemi.com

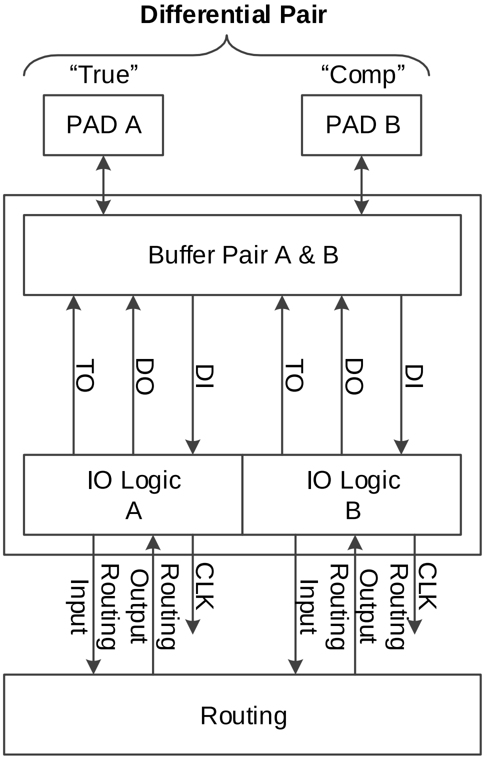
2 GPIO Overview

Gowin GPIO meets a variety of I/O standards and supports both single-ended and differential level standards, providing an easy connection with external buses, storage devices, video applications, and other standards.

The basic blocks of the GPIO in the GOWINSEMI FPGA products are IOB, including I/O buffer, I/O logic, and the relevant programmable routing unit. The programmable routing unit is similar to the CRU in CFU.

As shown in Figure 2-1, each IOB contains two pins (A and B). They can be used as a differential pair or as a single-end input/output. The I/O buffer supports both single-ended and differential standards. The I/O logic supports deserializer, serializer, delay control, and byte alignment, and is suitable for high-speed data transmission. The programmable routing unit is used to inter-connect I/O blocks with other on-chip resources.

Figure 2-1 IOB Structure View



The features of the input/output blocks in Gowin FPGA series are:

- VCCO is supplied based on bank.
- Supports LVCMOS, PCI, LVTTL, LVDS, SSTL, and HSTL.
- Some devices [1] support MIPI level standard and MIPI I3C OpenDrain/PushPull conversion.
- Supports input hysteresis option
- Supports output drive strength option
- Supports individual bus keeper, pull-up/down resistor, and open drain output options
- Supports hot socket
- I/O logic supports SDR mode and DDR mode, etc.

Note!

[1]: For devices that support MIPI and I3C, please refer to the devices supported in 3.6.9 MIPI_IBUF, 3.6.10 MIPI_OBUFand 3.6.12 I3C_IOBUF.

3 Input/Output Buffer

3.1 GPIO Level Standard

GOWINSEMI FPGA products support both single-ended and differential standards. The single-ended standard can use built-in IO voltage as a reference voltage or any I/O voltage as an external reference voltage input. All banks in GOWINSEMI FPGA products support differential input. Emulated LVDS differential output is implemented by using external resistors and differential LVCMOS buffer output. For banks supporting true LVDS differential output and differential input matching, please refer to 3.2 GPIO Banking for more details.

For the pin voltage requirements for different level standards supported by GOWINSEMI FPGA products, please refer to the "I/O Level Standards" section in the corresponding data sheets:

- DS100, GW1N series of FPGA Products Data Sheet
- DS117, GW1NR series of FPGA Products Data Sheet
- DS821, GW1NS series of FPGA Products Data Sheet
- DS841, GW1NZ series of FPGA Products Data Sheet
- DS861, GW1NSR series of FPGA Products Data Sheet
- DS871, GW1NSE series of SecureFPGA Products Data Sheet
- DS881, GW1NSER series of SecureFPGA Products Data Sheet
- DS891, GW1NRF series of Bluetooth FPGA Products Data Sheet
- DS102, GW2A series of FPGA Products Data Sheet
- DS226, GW2AR series of FPGA Products Data Sheet
- DS961, GW2ANR series of FPGA Products Data Sheet
- DS971, GW2AN-18X & 9X Data Sheet
- DS976, GW2AN-55 Data Sheet

3.2 GPIO Banking

The generic attributes of GPIO are:

- All banks support emulated LVDS differential output using external resistance.
- All banks support pull up, pull down, and bus-keeper settings.
- Each bank supports one kind of pin voltage.
- Each bank supports one reference voltage signal, whether it is from an external pin or from the internal reference voltage generator.

3.3 Power Supply Requirements

GOWINSEMI FPGA products can be powered and operated when V_{CC} and V_{CCO} reach a certain threshold and POR is set. By default, the GPIO is tristate input weak pull-up for blank chips. There are no power-on and power-off sequence requirements for core voltage and pin voltage for GOWINSEMI FPGA products.

Each bank supports one reference voltage input (V_{REF}). Any I/O in one Bank can be configured as an input reference voltage. To support SSTL and HSTL, the reference voltage can be set as half of the I/O voltage. The input reference voltage can also be generated by the internal reference voltage generator. The internal reference voltage generator and the external reference voltage cannot be effective at the same time because each bank has only one reference voltage.

The GOWINSEMI FPGA GPIO includes two input/output pins, marked as A and B respectively. Pin A corresponds to the T (True) of the differential pair, and Pin B corresponds to the C (Comp) of the differential pair.

3.3.1 LVCMOS Buffer Configuration

All GPIOs contain LVCMOS buffers. These LVCMOS buffers can be configured in a variety of modes to support different applications. Each LVCMOS buffer can be configured as weak pull-up, weak pull-down, and bus-keeper. The pull-up and pull-down offer a fixed characteristic, which is useful when creating wired logic such as wired ORs. The bus-keeper latches the signal in the last driven state, holding it at a valid level with minimal power consumption. Input leakage can be reduced by turning off the bus-hold circuit.

All LVCMOS buffers have programmable drive strength. Please refer to the corresponding data sheets for the detailed drive strength of different IO standards. The drive strength of GOWINSEMI FPGA products is guaranteed with minimum drive strength for each drive setting.

The hysteresis setting is used to prevent quick successive changes of levels in a noisy environment. All LVCMOS buffers support the hysteresis setting.

When a differential pair is configured as two single-ended pins, the relative delay between the two pins is maintained at a minimum, and the signal consistency is the best.

3.3.2 Differential Buffer Configuration

When a GPIO buffer is configured as a differential receiver, the input hysteresis and bus-hold will be disabled for the buffer.

GW1N and GW2A devices which support on-chip programmable 100 Ohm input differential matched resistance are listed as below:

- Bank 0 of GW1N-4, GW1NR-4, GW1NRF-4B, GW1N-9, GW1NR-9, GW1N-1, GW1NR-1.
- Bank 0 and bank 1 of GW1N-1S, GW1NS-4, GW1NS-4C, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW2A-18, GW2A-55, GW2AN-55, GW2ANR-18, GW2AR-18.
- Bank 2 of GW1N-2, GW1NR-2, GW1N-1P5.
- Bank 4 and bank 5 of GW2AN-18X, GW2AN-9X.

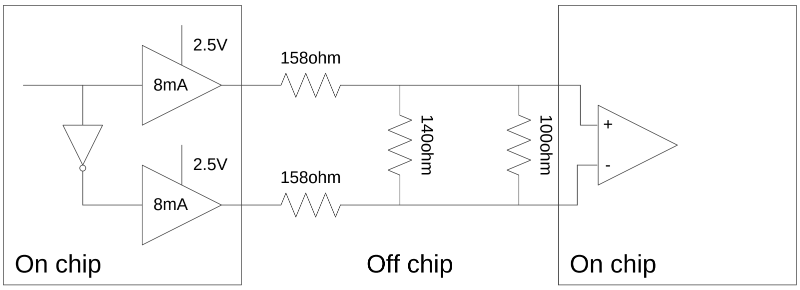
All the single-ended GPIO buffer pairs can be configured to support emulated LVDS differential output standards, such as LVPECL33E, MLVDS25E, BLVDS25E, etc. An off-chip impedance matching network is also required.

3.4 Emulated Differential Circuit Matching Network

3.4.1 Emulated LVDS

GOWINSEMI FPGA products can build compatible LVCMOS output standards via the complementary LVCMOS output and external matching network. Figure 3-1 shows the external matching network.

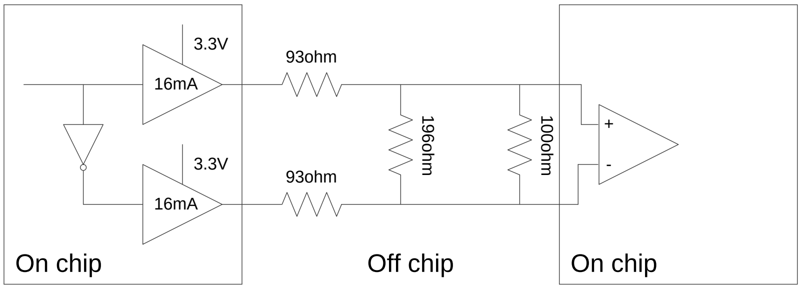
Figure 3-1 LVDS25E Matching Network



3.4.2 Emulated LVPECL

GOWINSEMI FPGA products can build compatible LVPECL output standards via the complementary LVCMOS output and external matching network. Figure 3-2 shows the external matching network.

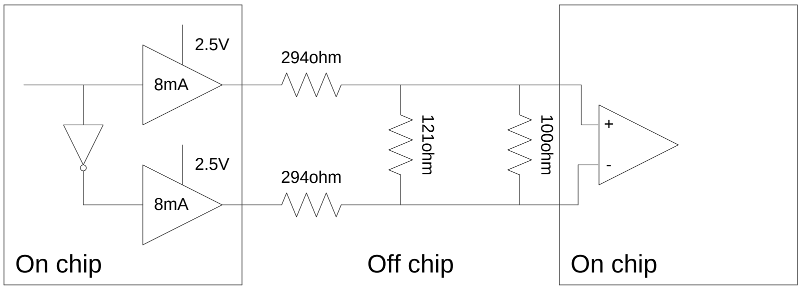
Figure 3-2 LVPECL Matching Network



3.4.3 Emulated RSDS

GOWINSEMI FPGA products can build compatible RSDS output standards via the complementary LVCMOS output and external matching network. Figure 3-3 shows the external matching network.

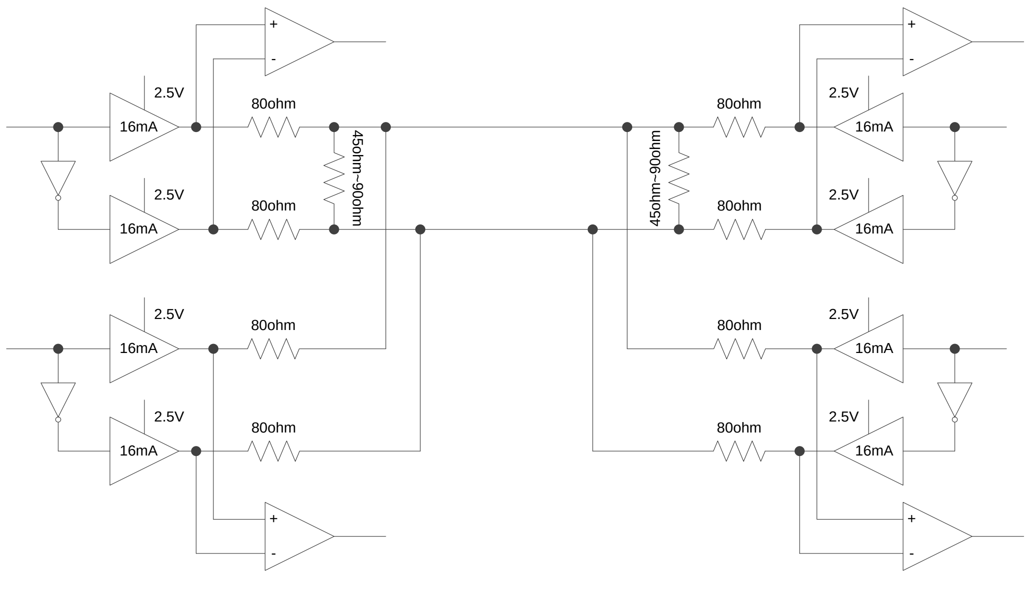
Figure 3-3 RSDS Matching Network



3.4.4 Emulated BLVDS

GOWINSEMI FPGA products can build compatible BLVDS output standards via the complementary LVCMOS output and external matching network. Figure 3-4 shows the external matching network.

Figure 3-4 BLVDS Matching Network



3.5 GPIO Software Configuration

You can set GPIO location, attributes, etc. through Floorplanner in Gowin Software, or you can customize the CST file to achieve this. The following is a detailed description of the physical constraints supported by CST files.

3.5.1 Location

Lock the physical location of GPIO:

IO_LOC "xxx" H4 exclusive;

3.5.2 Level Standard

Set the level standard for GPIO:

IO_PORT "xxx" IO_TYPE=LVCMOS18D;

3.5.3 Drive Strength

Set the drive strength of output pins or IO pins:

IO_PORT "xxx" DRIVE=12;

3.5.4 Pull Up/Pull Down

Set pull up/down modes, such as UP (pull-up), DOWN (pull down), KEEPER (bus-hold), and NONE (high impedance).

IO_PORT "xxx" PULL_MODE=DOWN;

3.5.5 Voltage Reference

Set reference voltage for GPIO. The reference voltage can be from external pins or internal reference voltage generator.

IO_PORT "xxx" VREF=VREF1_LOAD;

3.5.6 Hysteresis

Set the hysteresis value for input pins or bidirectional IO pins. The value is NONE, H2L, L2H, HIGH from small to large in sequence.

IO_PORT "xxx" HYSTERESIS=L2H;

3.5.7 Open Drain

Open Drain is available for both output and bidirectional IO pins. The values are ON and OFF.

IO_PORT "xxx" OPEN_DRAIN=ON;

3.5.8 Termination Matching Resistors for Single-ended Signal

Set termination matching resistors for single-ended signals. The values are OFF and ON.

IO_PORT "xxx" SINGLE_RESISTOR=ON;

3.5.9 Termination Matching Resistor for Differential Signal

Set termination matching resistors for differential signals. The values are OFF and ON.

IO_PORT "xxx" Diff_RESISTOR=ON;

3.6 GPIO Primitive

IO Buffer with buffer function includes normal buffer, emulated LVDS, and true LVDS.

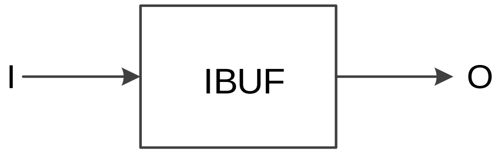
3.6.1 IBUF

Primitive Introduction

Input Buffer (IBUF)

Port Diagram

Figure 3-5 IBUF Port Diagram



Port Description

Table 3-1 IBUF Port Description

Port	I/O	Description
I	Input	Data input signal
O	Output	Data output signal

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

```
COMPONENT IBUF
        PORT (
                O:OUT std_logic;
                I:IN std_logic
        );
END COMPONENT;
uut:IBUF
```

```
        PORT MAP (
            O=>O,
            I=>I
        );
```

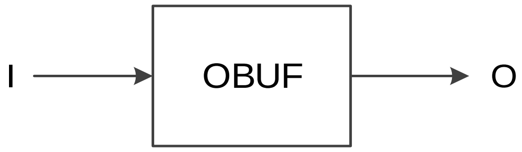
3.6.2 OBUF

Primitive Introduction

Output Buffer (OBUF).

Port Diagram

Figure 3-6 OBUF Port Diagram



Port Description

Table 3-2 OBUF Port Description

Port	I/O	Description
I	Input	Data input signal
O	Output	Data output signal

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

```
I=>I
);
```

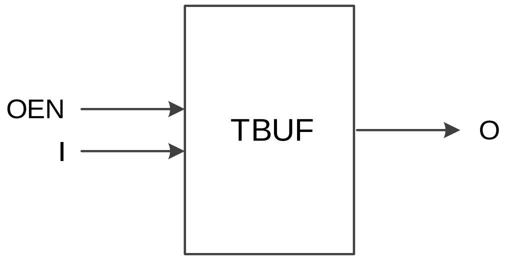
3.6.3 TBUF

Primitive Introduction

Output Buffer with Tristate Control (TBUF), active-low.

Port Diagram

Figure 3-7 TBUF Port Diagram



Port Description

Table 3-3 TBUF Port Description

Port	I/O	Description
I	Input	Data input signal
OEN	Input	Output tristate enable signal
O	Output	Data output signal

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

```
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
          );
```

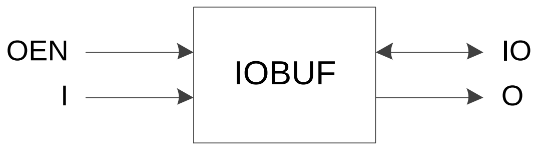
3.6.4 IOBUF

Primitive Introduction

Bi-Directional Buffer (IOBUF) is used as an input buffer when OEN is high and is used as an output buffer when ONE is low.

Port Diagram

Figure 3-8 IOBUF Port Diagram



Port Description

Table 3-4 IOBUF Port Description

Port	I/O	Description
I	Input	Data input signal
OEN	Input	Output tristate enable signal
IO	Inout	Input and output signals, bidirectional.
O	Output	Data output signal

Primitive Instantiation

Verilog Instantiation:

```
IOBUF uut(
        .O(O),
        .IO(IO),
        .I(I),
        .OEN(OEN)
);
```

Vhdl Instantiation:

```
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

Primitive Introduction

LVDS input buffer includes TLVDS_IBUF and ELVDS_IBUF.

True LVDS Input Buffer (TLVDS_IBUF).

Note!

GW1NZ-1 and GW1N-1S do not support TLVDS_IBUF.

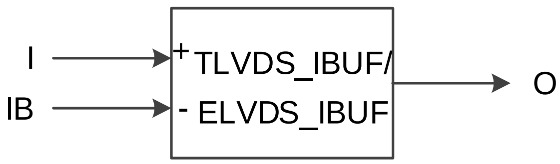
Emulated LVDS Input Buffer (ELVDS_IBUF).

Note!

GW1NZ-1 does not support ELVDS_IBUF.

Port Diagram

Figure 3-9 TLVDS_IBUF/ELVDS_IBUF Port Diagram



Port Description

Table 3-5 TLVDS_IBUF/ELVDS_IBUF Port Description

Port	I/O	Description
I	Input	Differential input A port signal
IB	Input	Differential input B port signal
O	Output	Data output signal

Primitive Instantiation

Example One

Verilog Instantiation:

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

Vhdl Instantiation:

```
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
    );
```

Example Two

Verilog Instantiation:

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

Vhdl Instantiation:

```
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

Primitive Introduction

LVDS output buffer includes TLVDS_OBUF and ELVDS_OBUF.

True LVDS Output Buffer (TLVDS_OBUF).

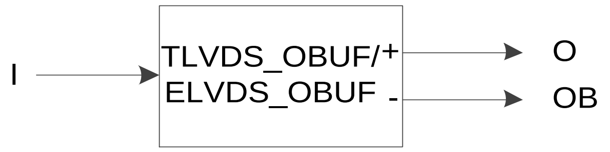
Note!

GW1N-1, GW1NR-1, GW1NZ-1 and GW1N-1S do not support TLVDS_OBUF.

Emulated LVDS Output Buffer (ELVDS_OBUF).

Port Diagram

Figure 3-10 TLVDS_OBUF/ELVDS_OBUF Port Diagram



Port Description

Table 3-6 TLVDS_OBUF/ELVDS_OBUF Port Description

Port	I/O	Description
I	Input	Data input signal
OB	Output	Differential output B signal
O	Output	Differential output A Signal

Primitive Instantiation

Example One

Verilog Instantiation:

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

Vhdl Instantiation:

```
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
      );
```

Example Two

Verilog Instantiation:

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

Vhdl Instantiation:

```
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

Primitive Introduction

LVDS tristate buffer includes TLVDS_TBUF and ELVDS_TBUF.

True LVDS Tristate Buffer (TLVDS_TBUF), active-low.

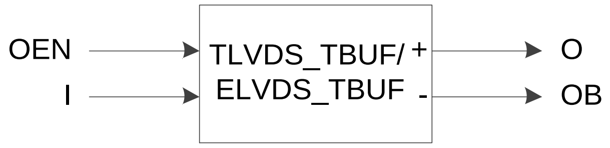
Note!

GW1N-1, GW1NR-1, GW1NZ-1 and GW1N-1S do not support TLVDS_TBUF.

Emulated LVDS Tristate Buffer (ELVDS_TBUF), active-low

Port Diagram

Figure 3-11 TLVDS_TBUF/ELVDS_TBUF Port Diagram



Port Description

Table 3-7 TLVDS_TBUF/ELVDS_TBUF Port Description

Port	I/O	Description
I	Input	Data input signal
OEN	Input	Output tristate enable signal
OB	Output	Differential B port output signal
O	Output	Differential A port output signal

Primitive Instantiation

Example One

Verilog Instantiation:

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

Vhdl Instantiation:

```
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
            );
```

Example Two

Verilog Instantiation:

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

Vhdl Instantiation:

```
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

Primitive Introduction

The LVDS inout buffer includes TLVDS_IOBUF and ELVDS_IOBUF.

True LVDS Bi-Directional Buffer (TLVDS_IOBUF) is used as true differential input buffer when OEN is high and used as true differential output buffer when OEN is low.

Devices Supported

Table 3-8 TLVDS_IOBUF Devices Supported

Product Family	Series	Device
Arora	GW2A	GW2A-18, GW2A-18C, GW2A-55, GW2A-55C
	GW2AN	GW2AN-55C
	GW2AR	GW2AR-18,GW2AR-18C
	GW2ANR	GW2ANR-18C
LittleBee®	GW1N	GW1N-4, GW1N-4B, GW1N-4D
	GW1NR	GW1NR-4, GW1NR-4B, GW1NR-4D
	GW1NRF	GW1NRF-4B

ELVDS_IOBUF is used as emulated differential input buffer when OEN is high and used as emulated differential output buffer when OEN is low.

Note!

The GW1NZ-1 does not support ELVDS_IOBUF.

Port Diagram

Figure 3-12 TLVDS_IOBUF/ELVDS_IOBUF Port Diagram

Port Description

Table 3-9 TLVDS_IOBUF/ELVDS_IOBUF Port Description

Port	I/O	Description
I	Input	Data input signal
OEN	Input	Output tristate enable signal
O	Output	Data output signal
IOB	Inout	Differential B port input/output
IO	Inout	Differential A port input/output

Primitive Instantiation

Verilog Instantiation:

```
ELVDS_IOBUF uut(
    .O(O),
    .IO(IO),
    .IOB(IOB),
    .I(I),
    .OEN(OEN)
);
```

Vhdl Instantiation:

```
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

Primitive Introduction

MIPI Input Buffer (MIPI_IBUF) includes HS input mode and LP bi-direction mode, and HS mode supports dynamic resistance configuration.

Devices Supported

Table 3-10 MIPI_IBUF Devices Supported

Product Family	Series	Device
LittleBee®	GW1N	GW1N-9, GW1N-9C, GW1N-2, GW1N-1P5, GW1N-2B, GW1N-1P5B

Product Family	Series	Device
	GW1NR	GW1NR-9, GW1NR-9C, GW1NR-2, GW1NR-2B
	GW1NS	GW1NS-4, GW1NS-4C
	GW1NSER	GW1NSER-4C
	GW1NSR	GW1NSR-4, GW1NSR-4C
Arora	GW2AN	GW2AN-18X, GW2AN-9X

Functional Description

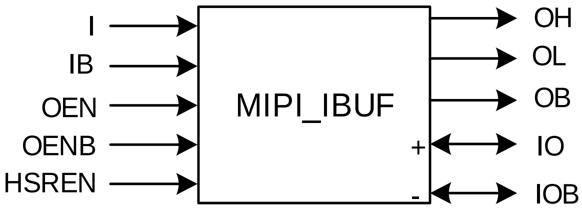
MIPI_IBUF supports LP and HS mode. IO and IOB are connected to pad.

LP mode: Supports bi-directional. When OEN is low, I is input and IO is output; when OEN is high, IO is input and OL is output; when OENB is low, IB is input and IOB is output; when OENB is high, IOB is input and OB is output.

HS mode: IO and IOB are the differential inputs. OH is the output, then HSREN controls the termination resistor.

Port Diagram

Figure 3-13 MIPI_IBUF Port Diagram



Port Description

Table 3-11 MIPI_IBUF Port Description

Port	I/O	Description
I	Input	In LP mode, I is the input when OEN is low.
IB	Input	In LP mode, IB is the input when OENB is low.
HSREN	Input	In HS mode, controls termination resistance.
OEN	Input	In LP mode, inputs/outputs tristate control signal.
OENB	Input	In LP mode, inputs/outputs tristate control signal.
OH	Output	In HS mode, data output signal.
OL	Output	In LP mode, OL is the output when OEN is high.
OB	Output	In LP mode, OB is the output when OENB is high.
IO	Inout	● In LP mode, IO is output when OEN is low and input when OEN is high. ● In HS mode, IO is input.
IOB	Inout	● In LP mode, IOB is output when OENB is low and IOB is input when OENB is high.

Port	I/O	Description
		● In HS mode, IOB is input.

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

```
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

Primitive Introduction

MIPI Output Buffer (MIPI_OBUF) includes HS mode and LP mode.

MIPI_OBUF is used as (HS) MIPI output buffer when MODESEL is high and used as (LP) MIPI output buffer when MODESEL is low.

Devices Supported

Table 3-12 MIPI_OBUF Devices Supported

Product Family	Series	Device
LittleBee®	GW1N	GW1N-9,GW1N-9C
	GW1NR	GW1NR-9,GW1NR-9C
	GW1NS	GW1NS-2, GW1NS-2C, GW1NS-4, GW1NS-4C
	GW1NZ	GW1NZ-2
	GW1NSE	GW1NSE-2C
	GW1NSER	GW1NSER-4C
	GW1NSR	GW1NSR-2, GW1NSR-2C, GW1NSR-4, GW1NSR-4C

Port Diagram

Figure 3-14 MIPI_OBUF Port Diagram

Port Description

Table 3-13 MIPI_OBUF Port Description

Port	I/O	Description
I	Input	A-ended data input signal for HS mode or LP mode
IB	Input	B-ended data input signal in LP mode
MODESEL	Input	Mode selection signal, HS or LP mode.

Port	I/O	Description
O	Output	A-ended data output signal, A differential output in HS mode, A single-ended output in LP mode.
OB	Output	B-ended data output signal, B differential output in HS mode, B single-ended output in LP mode.

Primitive Instantiation

Verilog Instantiation:

```
MIPI_OBUF uut(
    .O(O),
    .OB(OB),
    .I(I),
    .IB(IB),
    .MODESEL(MODESEL)
);
```

Vhdl Instantiation:

```
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

Primitive Introduction

MIPI_OBUF_A includes HS mode and LP mode.

MIPI Output Buffer with IL Signal (MIPI_OBUF_A) is used as (HS) MIPI output buffer when MODESEL is high and used as (LP) MIPI output buffer when MODESEL is low. The difference with MIPI_OBUF is the addition of the IL port as an A input in LP mode.

Devices Supported

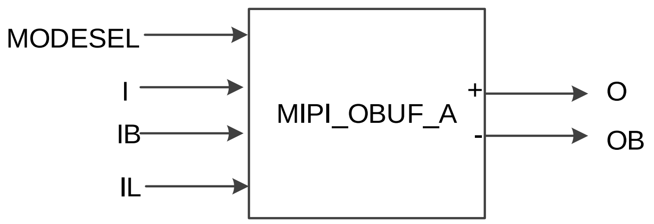
For the devices supported by MIPI_OBUF_A, apart from the devices listed in Table 3-12, you can also see the table below.

Table 3-14 MIPI_OBUF_A Devices Supported (Added)

Product Famliy	Series	Device
LittleBee®	GW1N	GW1N-2, GW1N-1P5, GW1N-2B, GW1N-1P5B
	GW1NR	GW1NR-2, GW1NR-2B

Port Diagram

Figure 3-15 MIPI_OBUF_A Port Diagram



Port Description

Table 3-15 MIPI_OBUF_A Port Description

Port	I/O	Description
I	Input	A-ended data input signal in HS mode
IB	Input	B-ended data input signal in LP mode
IL	Input	A-ended data input signal in LP mode
MODESEL	Input	Mode selection signal, HS or LP mode
O	Output	A-ended data output signal, A differential output in HS mode, A single-ended output in LP mode.
OB	Output	B-ended data output signal, B differential output in HS mode, B single-ended output in LP mode.

Primitive Instantiation

Verilog Instantiation:

```
MIPI_OBUF_A uut(
    .O(O),
    .OB(OB),
    .I(I),
    .IB(IB),
    .IL(IL),
    .MODESEL(MODESEL)
);
```

Vhdl Instantiation:

```
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

Primitive Introduction

I3C Bi-Directional Buffer (I3C_IOBUF) includes Normal mode and I3C mode.

I3C_IOBUF is used as a bi-directional buffer when MODESEL is high and used as a normal buffer when MODESEL is low.

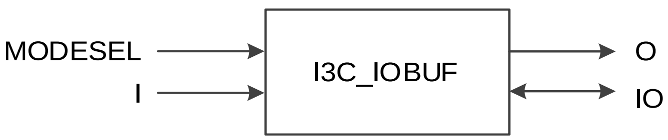
Devices Supported

Table 3-16 I3C_IOBUF Devices Supported

Product Family	Series	Device
LittleBee®	GW1N	GW1N-9,GW1N-9C
	GW1NR	GW1NR-9,GW1NR-9C
	GW1NS	GW1NS-2, GW1NS-2C, GW1NS-4, GW1NS-4C
	GW1NSE	GW1NSE-2C
	GW1NSER	GW1NSER-4C
	GW1NSR	GW1NSR-2, GW1NSR-2C, GW1NSR-4, GW1NSR-4C

Port Diagram

Figure 3-16 I3C_IOBUF Port Diagram



Port Description

Table 3-17 I3C_IOBUF Port Description

Ports	I/O	Description
I	Input	Data input signal
IO	Inout	Input and output signal, bidirectional.
MODESEL	Input	Mode selection signal, Normal mode or I3C mode
O	Output	Data output signal

Primitive Instantiation

Verilog Instantiation:

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

```
        .MODESEL(MODESEL)
    );
```

Vhdl Instantiation:

```
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

Primitive Introduction

MIPI_IBUF_HS implements HS mode for differential input and MIPI_IBUF_LP implements LP mode via single-ended input.

Devices Supported

Table 3-18 MIPI_IBUF_HS/MIPI_IBUF_LP Devices Supported

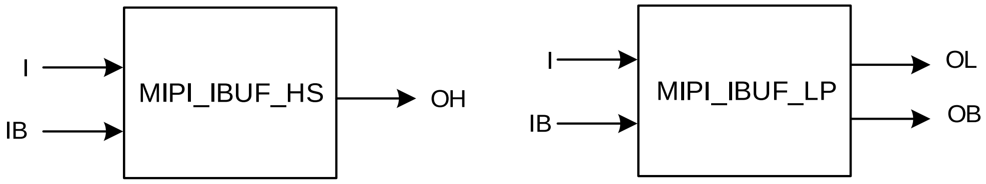
Family	Series	Device
LittleBee®	GW1NR	GW1NR-2

Functional Description

You can use the combination of MIPI_IBUF_HS and MIPI_IBUF_LP to support HS and LP modes via Floorplanner. Input I of MIPI_IBUF_HS and I of MIPI_IBUF_LP should be connected to the same signal, and input IB of MIPI_IBUF_HS and IB of MIPI_IBUF_LP should be connected to the same signal.

Port Diagram

Figure 3-17 MIPI_IBUF_HS/MIPI_IBUF_LP Port Diagram



Port Description

Table 3-19 MIPI_IBUF_HS Port Description

Port	I/O	Description
I	Input	In HS mode, differential input A-ended signal.
IB	Input	In HS mode, differential input B-ended signal.
OH	Output	In HS mode, data output signal.

Table 3-20 MIPI_IBUF_LP Port Description

Port	I/O	Description
I	Input	In LP mode, A single-ended input signal.
IB	Input	In LP mode, B single-ended input signal.
OL	Output	In LP mode, A-ended output singal.
OB	Output	In LP mode, B-ended output singal.

Connection Rule

- The output OH of MIPI_IBUF_HS can be connected to Iologic.
- The output OL and OB of MIPI_IBUF_LP are not allowed to be connected to Iologic.

Primitive Instantiation

Verilog Instantiation:

```
MIPI_IBUF_HS hs (
    .OH(OH),
    .I(I),
    .IB(IB)
);
MIPI_IBUF_LP lp (
    .OL(OL),
    .OB(OB),
    .I(I),
    .IB(IB)
```

```
);
```

Vhdl Instantiation:

```
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

Primitive Introduction

ELVDS_IBUF_MIPI (Emulated LVDS Input MIPI Buffer) implements HS mode and LP mode at the same time. Port A supports HS mode while port B implements LP mode only.

Functional Description

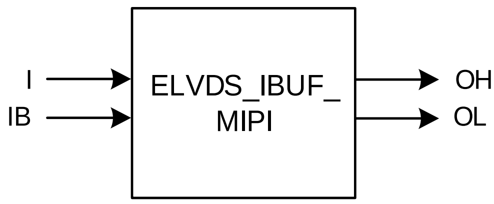
ELVDS_IBUF_MIPI supports LP and HS modes. Port I and port IB are connected to the pad.

LP mode: Port IB is input; port OL is output.

HS mode: Port I and Port IB are differential input; port OH is output.

Port Diagram

Figure 3-18 ELVDS_IBUF_MIPI Port Diagram



Port Description

Table 3-21 MIPI_IBUF Port Description

Port	I/O	Description
I	Input	I is put in HS mode
IB	Input	IB is input in LP mode
OH	Output	Data output signal in HS mode
OL	Output	Data output signal in LP mode

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

```
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 Input/Output Logic

I/O logic in GOWINSEMI FPGA products supports SDR and DDR modes, etc. In each mode, pin control (or pin differential signal pairs) can be configured as output signal, input signal, bi-directional signal and tristate output signal (output signal with tristate control).

Note!

- IOL6 and IOR6 pins of devices of GW1N-1, GW1NR-1, and GW1NZ-1 do not support IO logic.
- IOT2 and IOT3A pins of GW1N-2, GW1NR-2, GW1N-1P5, GW1N-2B, GW1N-1P5B, GW1NR-2B devices do not support IO logic.
- IOL10 and IOR10 pins of the devices of GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1N-4D and GW1NR-4D do not support IO logic.

Figure 4-1 shows the output of the I/O logic in GOWINSEMI FPGA products.

Figure 4-1 I/O Logic View-Output

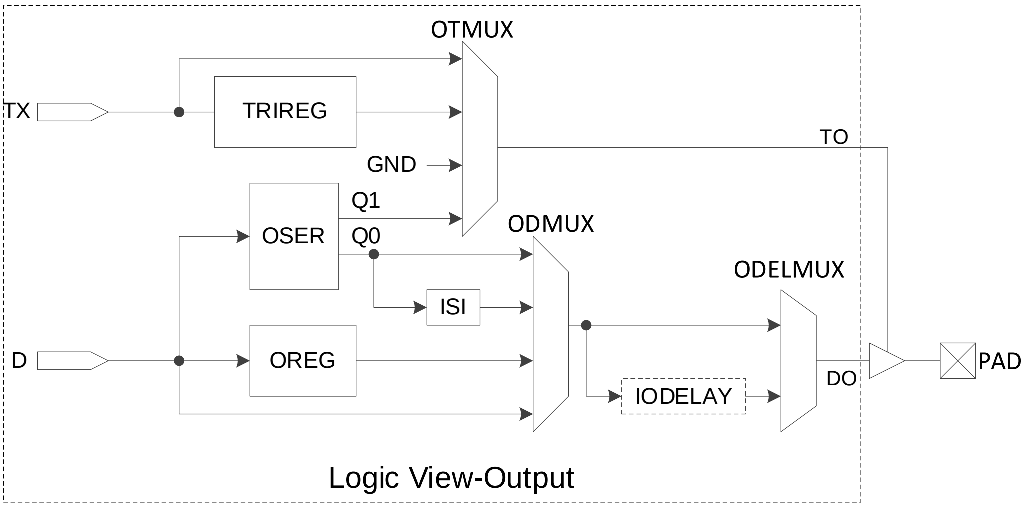
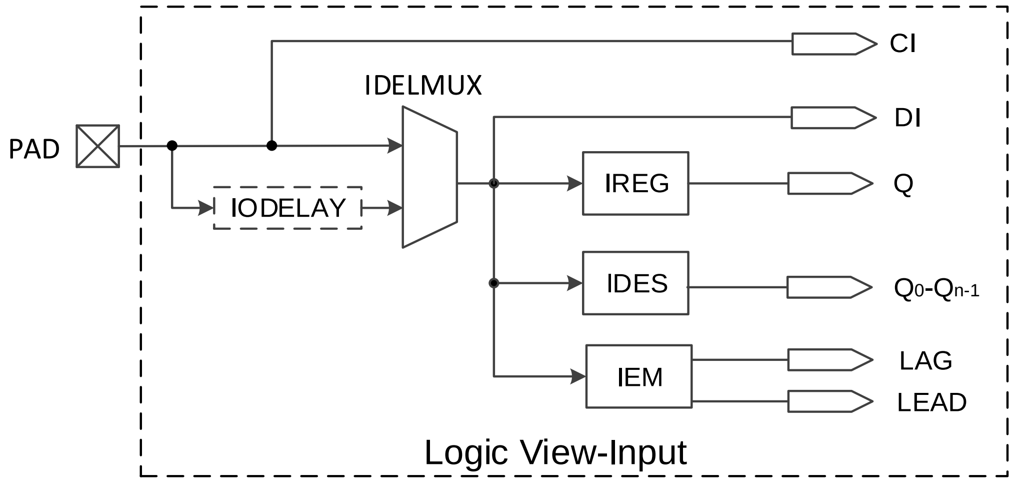


Figure 4-2 shows the input of the I/O logic in GOWINSEMI FPGA products.

Figure 4-2 I/O Logic View-Input



Note !

CI is the GCLK input signal and cannot connect to Fabric; DI is directly entered into Fabric.

4.1 SDR Mode

The input/output logic supports SDR mode and provides input register (IREG), output register (OREG) and tristate control register (TRIREG), the functions of which are the same as FF/LATCH in CFU. The FF/LATCH can be used as Iologic when the input D of the FF/LATCH is driven by a Buffer/IODELAY that does not drive other Iologics, or when the output Q of the FF/LATCH only drives a Buffer/IODELAY and the Buffer is not a MIPI Buffer.

4.2 DDR Mode Input Logic

4.2.1 IDDR

Primitive Introduction

Input Double Data Rate (IDDR)

Functional Description

Output data is provided to FPGA logic at the same clock edge in IDDR mode. IDDR logic diagram is as shown in Figure 4-3 and its timing diagram is as shown in Figure 4-4.

Figure 4-3 IDDR Logic Diagram

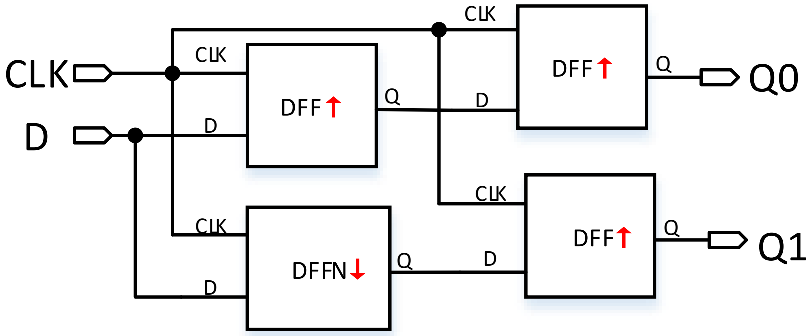
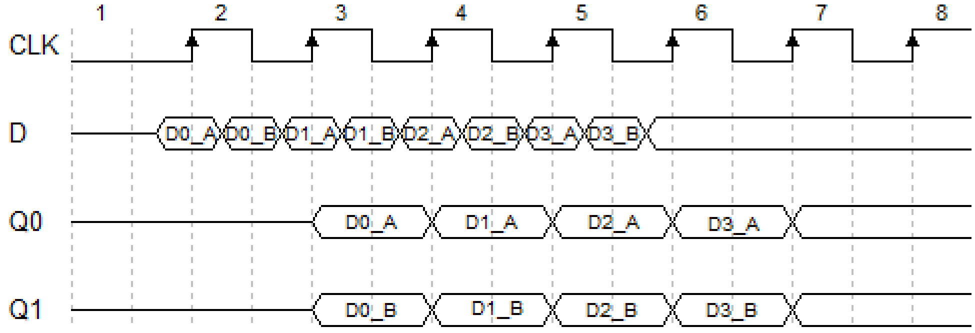
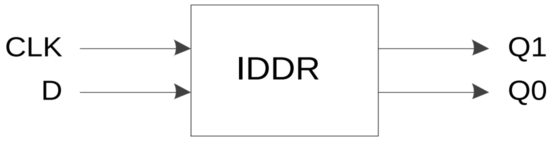


Figure 4-4 IDDR Timing Diagram



Port Diagram

Figure 4-5 IDDR Port Diagram



Port Description

Table 4-1 IDDR Port Description

Port	I/O	Description
D	Input	IDDR data input signal
CLK	Input	Clock input signal
Q0,Q1	Output	IDDR data output signal

Parameter Description

Table 4-2 IDDR Parameter Description

Name	Value	Default	Description
Q0_INIT	1'b0	1'b0	Initial value of Q0 output
Q1_INIT	1'b0	1'b0	Initial value of Q1 output

Connection Rule

Input D of IDDR can be directly from IBUF or from the output DO of IODELAY module.

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5 IP Generation.

Verilog Instantiation:

```
IDDR uut(
    .Q0(Q0),
```

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

Vhdl Instantiation:

```
    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 IDDRC

Primitive Introduction

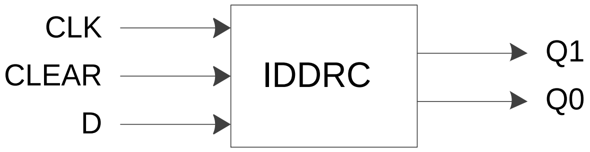
Dual Data Rate Input with Asynchronous Clear (IDDRC) is similar to IDDR to realize double data rate input and can be reset asynchronously.

Functional Description

Output data is provided to FPGA logic at the same clock edge in IDDRC mode.

Port Diagram

Figure 4-6 IDDRC Port Diagram



Port Description

Table 4-3 IDDRC Port Description

Port	I/O	Description
D	Input	IDDRC data input signal
CLK	Input	Clock input signal
CLEAR	Input	Asynchronous reset input signal, active-high.
Q0,Q1	Output	IDDRC data output signal

Parameter Description

Table 4-4 IDDRC Parameter Description

Name	Value	Default	Description
Q0_INIT	1'b0	1'b0	The initial value of Q0 output
Q1_INIT	1'b0	1'b0	The initial value of Q1 output

Connection Rule

Data input D of IDDRC can be directly from IBUF or from the output DO of IODELAY module.

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5 IP Generation.

Verilog Instantiation:

```
IDDRC uut(
        .Q0(Q0),
```

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

Vhdl Instantiation:

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

4.2.3 IDES4

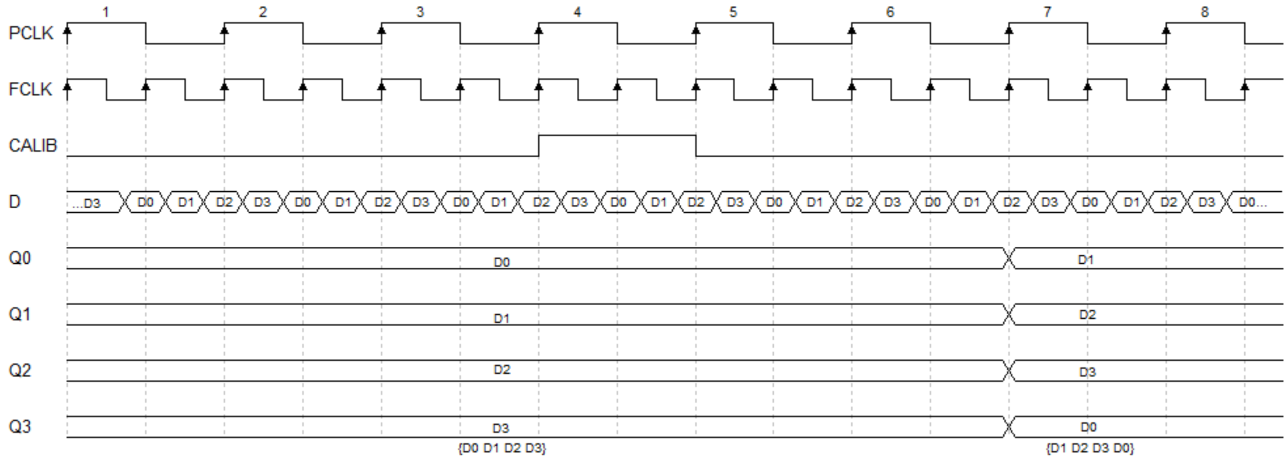
Primitive Introduction

The 1 to 4 Deserializer (IDES4) is a deserializer of 1 bit serial input and 4 bits parallel output.

Functional Description

IDES4 mode realizes 1:4 serial parallel conversion and output data is provided to FPGA logic at the same clock edge. CALIB is supported to adjust the sequence of output data. The data is shifted by one bit per pulse. After four shifts, the data output will be the same as the data before the shift. CALIB Timing diagram is as shown in Figure 4-7.

Figure 4-7 CALIB Timing Diagram



Note!

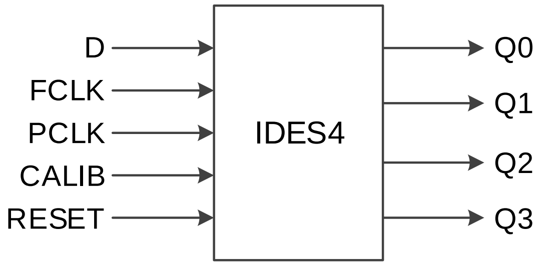
The pulse width and timing of the CALIB signal in the example are for reference only and can be adjusted as needed, the pulse width is equal to or greater than T_{PCLK}.

PCLK is usually obtained by FCLK frequency division：

$$f_{PCLK} = 1/2\, f_{FCLK}.$$

Port Diagram

Figure 4-8 IDES4 Port Diagram



Port Description

Table 4-5 IDES4 Port Description

Port	I/O	Description
D	Input	IDES4 data input signal
FCLK	Input	High speed clock Input signal
PCLK	Input	Primary clock input signal
CALIB	Input	CALIB signal, used to adjust the sequence of output data, active-high.
RESET	Input	Asynchronous reset input signal, active-high.
Q3=Q0	Output	IDES8 data output signal

Parameter Description

Table 4-6 IDES4 Parameter Description

Name	Value	Default	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Rule

Data input D of IDES4 can be directly from IBUF or from the output DO of IODELAY module.

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5 IP Generation.

Verilog Instantiation:

```
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 Instantiation:

```
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

Primitive Introduction

The 1 to 8 Deserializer (IDES8) is a deserializer of 1 bit serial input and 8 bits parallel output.

Functional Description

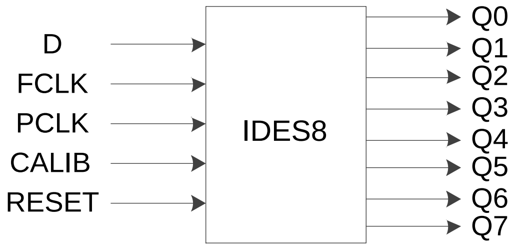
IDES8 mode realizes 1:8 serial parallel conversion and output data is provided to FPGA logic at the same clock edge. CALIB is supported to adjust the sequence of output data. The data is shifted by one bit per pulse. After four shifts, the data output will be the same as the data before the shift.

PCLK is usually obtained by FCLK frequency division：

$f_{PCLK} = 1/4\, f_{FCLK}$.

Port Diagram

Figure 4-9 IDES8 Port Diagram



Port Description

Table 4-7 IDES8 Port Description

Port	I/O	Description
D	Input	IDES8 data input signal
FCLK	Input	High speed clock input signal
PCLK	Input	Primary clock input signal
CALIB	Input	CALIB input signal, used to adjust the sequence of output data, active-high.
RESET	Input	Asynchronous reset input signal, active-high.
Q7=Q0	Output	IDES8 data output signal

Parameter Description

Table 4-8 IDES8 Parameter Description

Name	Value	Default	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Rule

Data input D of IDES8 can be directly from IBUF or from DO in IODELAY module.

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5 IP Generation.

Verilog Instantiation:

```
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 Instantiation:

```
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

Primitive Introduction

The 1 to 10 Deserializer (IDES10) is a deserializer of 1 bit serial input and 10 bits parallel output.

Functional Description

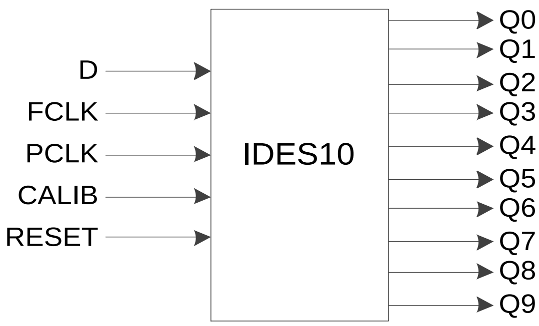
IDES10 mode realizes 1:10 serial parallel conversion and output data is provided to FPGA logic at the same clock edge. CALIB is supported to adjust the sequence of output data. The data is shifted by one bit per pulse. After ten shifts, the data output will be the same as the data before the shift.

PCLK is usually obtained by FCLK frequency division：

$f_{PCLK} = 1/5\, f_{FCLK}$.

Port Diagram

Figure 4-10 IDES10 Port Diagram



Port Description

Table 4-9 IDES10 Port Description

Port	I/O	Description
D	Input	IDES10 data input signal
FCLK	Input	High speed clock input signal
PCLK	Input	Primary clock input signal
CALIB	Input	CALIB signal, used to adjust the sequence of output data, active-high.
RESET	Input	Asynchronous reset input signal, active-high.
Q9=Q0	Output	IDES10 data output signal

Parameter Description

Table 4-10 IDES10 Parameter Description

Name	Value	Default	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Rule

Data input D of IDES10 can be directly from IBUF or from DO in IODELAY module.

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5 IP Generation.

Verilog Instantiation:

```
IDES10 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 Instantiation:

```
COMPONENT IDES10
        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

Primitive Introduction

The 1 to 7 Deserializer (IVIDEO) is a deserializer of 1 bit serial input and 7 bits parallel output.

Functional Description

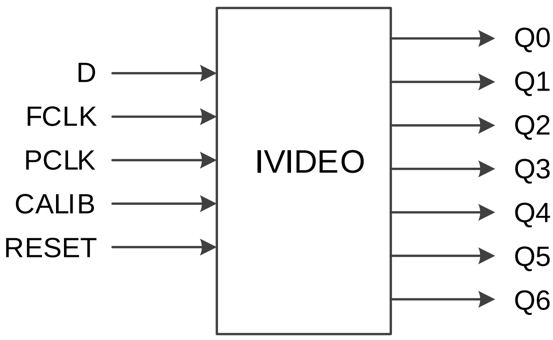
IVIDEO mode realizes 1:7 serial parallel conversion and output data is provided to FPGA logic at the same clock edge. CALIB is supported to adjust the sequence of output data. The data is shifted by two bits per pulse. After seven shifts, the data output will be the same as the data before the shift.

PCLK is usually obtained by FCLK frequency division：

$$f_{PCLK} = 1/3.5\, f_{FCLK}$$.

Port Diagram

Figure 4-11 IVIDEO Port Diagram



Port Description

Table 4-11 IVIDEO Port Description

Port	I/O	Description
D	Input	IVIDEO data input signal
FCLK	Input	High speed clock input signal
PCLK	Input	Primary clock input signal
CALIB	Input	CALIB signal, used to adjust the sequence of output data, active-high.
RESET	Input	Asynchronous reset input signal, active-high.
Q6~Q0	Output	IVIDEO data output signal

Parameter Description

Table 4-12 IVIDEO Parameter Description

Name	Value	Default	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Rule

Data input D of IVIDEO can be directly from IBUF or from DO in IODELAY module.

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5 IP Generation.

Verilog Instantiation:

```
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 Instantiation:

```
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

Primitive Introduction

The 1 to 16 Deserializer (IDES16) is a deserializer of 1 bit serial input and 16 bits parallel output.

Devices Supported

Table 4-13 IDES16 Devices Supported

Product Family	Series	Device
LittleBee®	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

Functional Description

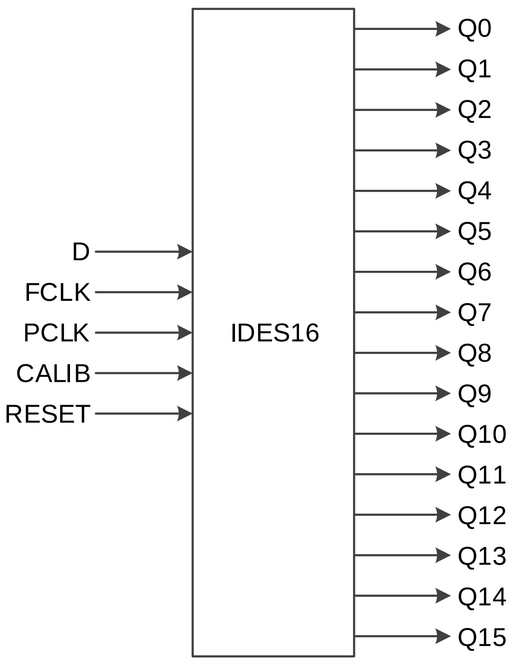
IDES16 mode realizes 1:16 serial parallel conversion and output data is provided to FPGA logic at the same clock edge. CALIB is supported to adjust the sequence of output data. Each pulse data is shifted by one bit. After sixteen shifts, the data output will be the same as the data before the shift.

PCLK is usually obtained by FCLK frequency division：

$f_{PCLK} = 1/8\, f_{FCLK}$.

Port Diagram

Figure 4-12 IDES16 Port Diagram



Port Description

Table 4-14 IDES16 Port Description

Port	I/O	Description
D	Input	IDES16 data input signal
FCLK	Input	High speed clock input signal
PCLK	Input	Primary clock input signal
CALIB	Input	CALIB signal, used to adjust the sequence of output data, active-high.
RESET	Input	Asynchronous reset input signal, active-high.
Q15~Q0	Output	IDES16 data output signal

Parameter Description

Table 4-15 IDES10 Parameter Description

Name	Value	Default	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Rule

Data input D of IDES16 can be directly from IBUF or from DO in IODELAY module.

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5 IP Generation.

Verilog Instantiation:

```
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 Instantiation:

```
    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

Primitive Introduction

The Input Double Data Rate with Memory (IDDR_MEM) realizes double data rate input with memory.

Devices Supported

Table 4-16 IDDR_MEM Devices Supported

Product Family	Series	Device
Arora	GW2A	GW2A-18, GW2A-18C, GW2A-55, GW2A-55C
	GW2AN	GW2AN-55C
	GW2AR	GW2AR-18,GW2AR-18C
	GW2ANR	GW2ANR-18C

Functional Description

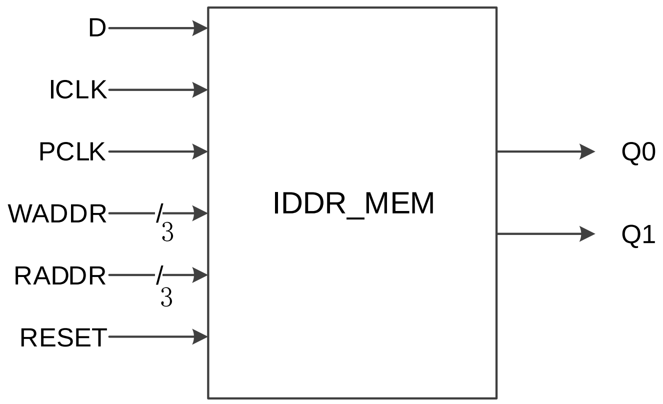
IDDR_MEM output data is provided to FPGA logic at the same clock edge. IDDR_MEM needs to be used with DQS. ICLK connects the DQSR90 of DQS output signals and sends data to IDDR_MEM according to the ICLK clock edge. WADDR [2: 0] connects the WPOINT output signal of DQS; RADDR [2: 0] connects the RPOINT output signal of DQS.

The frequency relation between PCLK and ICLK is $f_{PCLK} = f_{ICLK}$.

You can determine the phase relationship between PCLK and ICLK according to the DLLSTEP value of DQS.

Port Diagram

Figure 4-13 IDDR_MEM Port Diagram



Port Description

Table 4-17 IDDR_MEM Port Description

Port	I/O	Description
D	Input	IDDR_MEM data input signal
ICLK	Input	Clock input signal from DQSR90 in DQS module

Port	I/O	Description
PCLK	Input	Primary clock input signal
WADDR[2:0]	Input	Write address signal from WPOINT in DQS module
RADDR[2:0]	Input	Read address signal from RPOINT in DQS module
RESET	Input	Asynchronous reset input signal, active-high.
Q1~Q0	Output	IDDR_MEM data output signal

Parameter Description

Table 4-18 IDDR_MEM Parameter Description

Name	Value	Default	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Rule

- Data input D of IDDR_MEM can be directly from IBUF or from DO in IODELAY module.
- ICLK needs DQSR90 from a DQS module.
- WADDR[2:0] needs WPOINT from DQS module
- RADDR[2:0] needs RPOINT from DQS module;

Primitive Instantiation

Verilog Instantiation:

```
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 Instantiation:

```
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

Primitive Introduction

The 1 to 4 Deserializer with Memory (IDES4_MEM) realizes 1:4 serial-parallel with memory.

Devices Supported

Table 4-19 IDES4_MEM Devices Supported

Product Family	Series	Device
Arora	GW2A	GW2A-18, GW2A-18C, GW2A-55, GW2A-55C
	GW2AN	GW2AN-55C
	GW2AR	GW2AR-18,GW2AR-18C
	GW2ANR	GW2ANR-18C

Functional Description

IDES4_MEM realizes 1:4 serial parallel conversion and the output data is provided to FPGA logic at the same clock edge. CALIB is supported to adjust the sequence of output data. Each pulse data is shifted by one bit. After four shifts, the data output will be the same as the data before the shift.

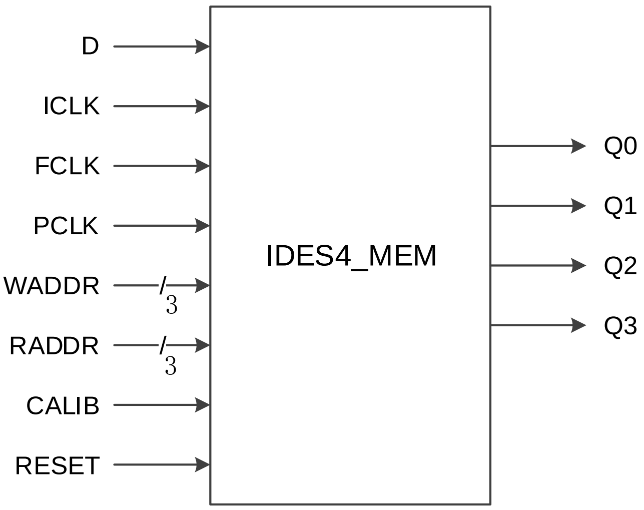
The ICLK connects the output signal DQSR90 of DQS and sends data to IDES4_MEM according to the ICLK clock edge. WADDR [2: 0] connects the output signal WPOINT of DQS; RADDR [2: 0] connects the output signal RPOINT of DQS.

The frequency relation between PCLK, FCLK and ICLK is $f_{PCLK} = 1/2\, f_{FCLK} = 1/2\, f_{ICLK}$.

You can determine the phase relationship between PCLK and ICLK according to the DLLSTEP value of DQS.

Port Diagram

Figure 4-14 IDES4_MEM Port Diagram



Port Description

Table 4-20 IDES4_MEM Port Description

Port	I/O	Description
D	Input	IDES4_MEM data input signal
ICLK	Input	Clock input signal from DQSR90 in DQS module
FCLK	Input	High speed clock input signal
PCLK	Input	Primary clock input signal
WADDR[2:0]	Input	Write address signal from WPOINT in DQS module
RADDR[2:0]	Input	Read address signal from RPOINT in DQS module
CALIB	Input	CALIB signal, used to adjust the sequence of output data, active-high.
RESET	Input	Asynchronous reset input signal, active-high.
Q3~Q0	Output	IDES4_MEM data output signal

Parameter Description

Table 4-21 IDES4_MEM Parameter Description

Name	Value	Default	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Rule

- Data input D of IDES4_MEM can be directly from IBUF or from DO in IODELAY module.
- ICLK needs DQSR90 from a DQS module.
- WADDR[2:0] needs WPOINT from DQS module
- RADDR[2:0] needs RPOINT from DQS module;

Primitive Instantiation

Verilog Instantiation:

```
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 Instantiation:

```
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

Primitive Introduction

The 1 to 8 Deserializer with Memory (IDES8_MEM) realizes 1:8 serial parallel with memory.

Devices Supported

Table 4-22 IDES8_MEM Devices Supported

Product Family	Series	Device
Arora	GW2A	GW2A-18, GW2A-18C, GW2A-55, GW2A-55C
	GW2AN	GW2AN-55C
	GW2AR	GW2AR-18,GW2AR-18C
	GW2ANR	GW2ANR-18C

Functional Description

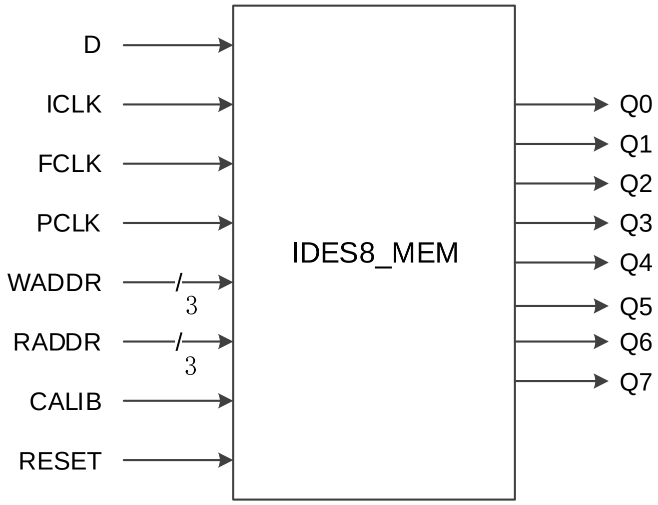
IDES8_MEM realizes 1:8 serial parallel conversion and output data is provided to FPGA logic at the same clock edge. CALIB is supported to adjust the sequence of output data. The data is shifted by one bit per pulse. After eight shifts, the data output will be the same as the data before the shift. The ICLK connects the output signal DQSR90 of DQS and sends data to IDES8_MEM according to the ICLK clock edge. WADDR[2:0] connects the output signal WPOINT of DQS; RADDR[2:0] connects output signal RPOINT of DQS.

The frequency relation between PCLK, FCLK and ICLK is $f_{PCLK} = 1/4\, f_{FCLK} = 1/4\, f_{ICLK}$.

You can determine the phase relationship between PCLK and ICLK according to the DLLSTEP value of DQS.

Port Diagram

Figure 4-15 IDES8_MEM Port Diagram



Port Description

Table 4-23 IDES8_MEM Port Description

Port	I/O	Description
D	Input	IDES8_MEM data input signal
ICLK	Input	Clock input signal from DQSR90 in DQS module
FCLK	Input	High speed clock Input signal
PCLK	Input	Primary clock input signal
WADDR[2:0]	Input	Write address signal from WPOINT in DQS module
RADDR[2:0]	Input	Read address signal from RPOINT in DQS module
CALIB	Input	CALIB signal, used to adjust the sequence of output data, active-high.
RESET	Input	Asynchronous reset input signal, active-high.
Q7~Q0	Output	IDES8_MEM data output signal

Parameter Description

Table 4-24 IDES8_MEM Parameters Description

Name	Value	Default	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Rule

- Data input D of IDES8_MEM can be directly from IBUF or from DO in IODELAY module.
- ICLK needs DQSR90 from a DQS module.
- WADDR[2:0] needs WPOINT from DQS module

- RADDR[2:0] needs RPOINT from DQS module.

Primitive Instantiation

Verilog Instantiation:

```
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 Instantiation:

```
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 Mode Output Logic

4.3.1 ODDR

Primitive Introduction

Dual Data Rate Output (ODDR)

Functional Description

ODDR mode is used for transferring double data rate signals from FPGA devices. Where Q0 is the double rate data output, Q1 is used for the OEN signal of IOBUF/TBUF connected by Q1. ODDR logic diagram is as shown in Figure 4-16 and its timing diagram is as shown in Figure 4-17.

Figure 4-16 ODDR Logic Diagram

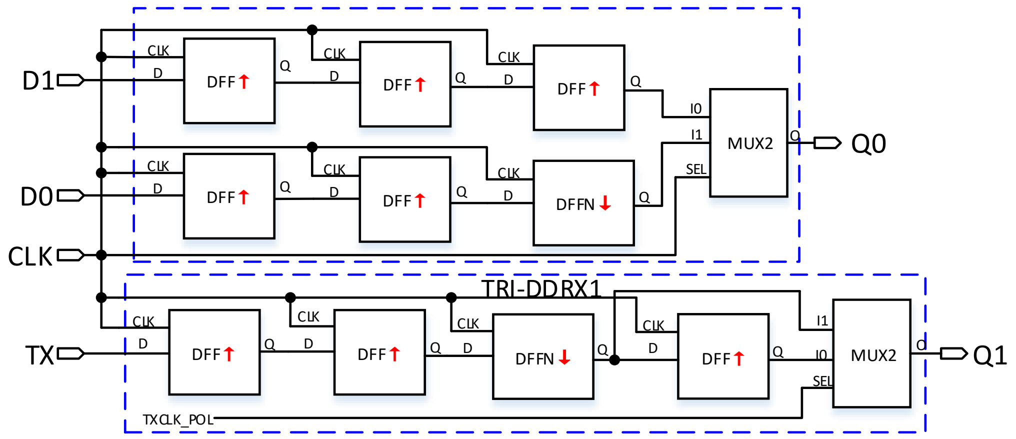
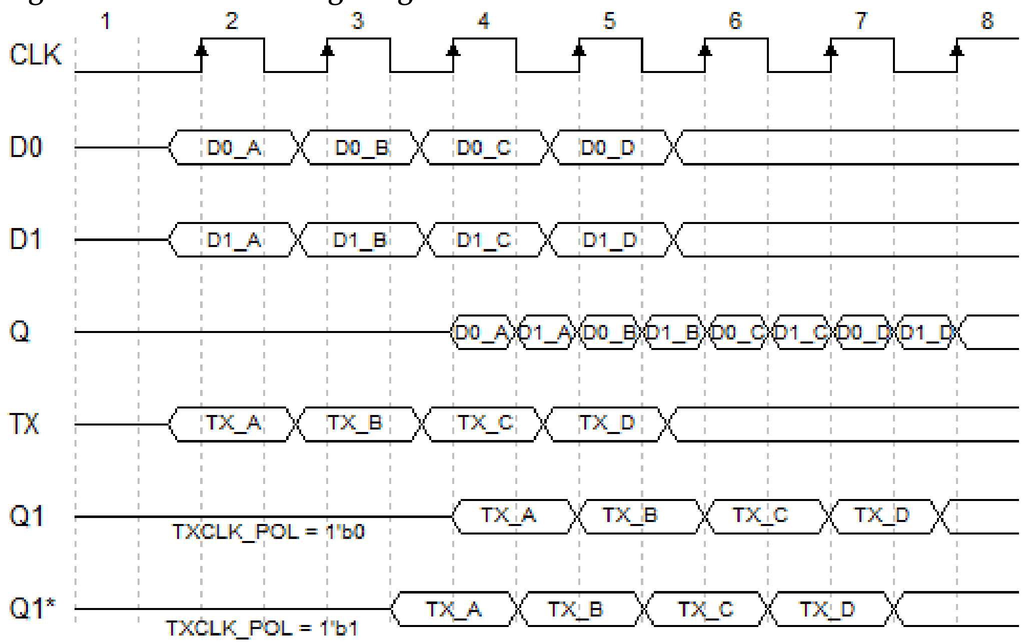
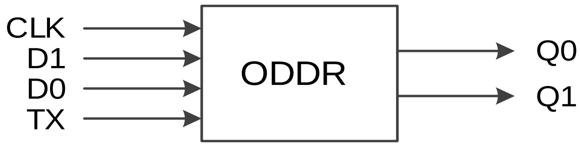


Figure 4-17 ODDR Timing Diagram



Port Diagram

Figure 4-18 ODDR Port Diagram



Port Description

Table 4-25 ODDR Port Description

Port	I/O	Description
D0,D1	Input	ODDR data input signal
TX	Input	Q1 generated by TRI-DDRX1
CLK	Input	Clock input signal
Q0	Output	ODDR data output signal
Q1	Output	ODDR tristate enable control data output can be connected to the IOBUF/TBUF OEN signal connected to Q0, or left floating.

Parameter Description

Table 4-26 ODDR Parameter Description

Name	Value	Default	Description
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 output clock polarity control ● 1'b0:Q1 posedge output; ● 1'b1:Q1 negedge output
INIT	1'b0	1'b0	Initial value of ODDR output

Connection Rule

- Q0 can be directly connected to OBUF, or connected to input port DI through IODELAY module;
- Q1 shall be connected to the OEN signal of IOBUF/TBUF connected to Q0, or left floating.

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5 IP Generation.

Verilog Instantiation:

```
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 Instantiation:

```
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

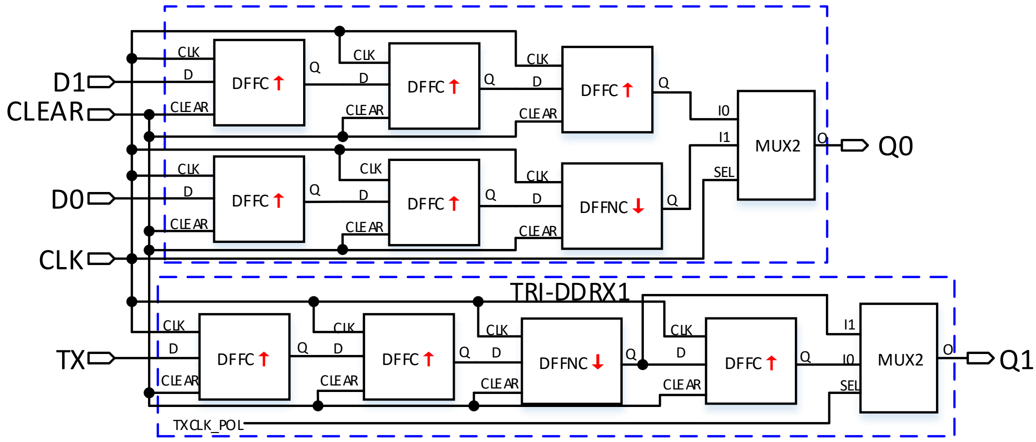
Primitive Introduction

Dual Data Rate Output with Asynchronous Clear (ODDRC) is similar to ODDR to realize double data rate and can be reset asynchronously.

Functional Description

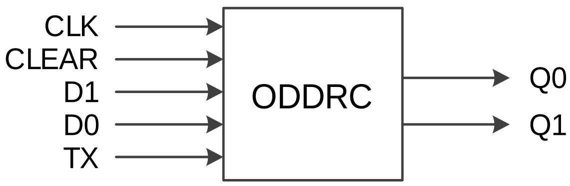
ODDRC mode is used for transferring double data rate signals from FPGA devices. Where Q0 is the double rate data output, Q1 is used for the OEN signal of IOBUF/TBUF connected to Q1. Its logic diagram is as shown in Figure 4-19.

Figure 4-19 ODDRC Logic Diagram



Port Diagram

Figure 4-20 ODDRC Port Diagram



Port Description

Table 4-27 ODDRC Port Description

Port	I/O	Description
D0, D1	Input	ODDRC data Input Signal
TX	Input	Input Q1 generated by TRI-DDRX1
CLK	Input	Clock input signal
CLEAR	Input	Asynchronous reset input signal, active-high.
Q0	Output	ODDRC data output signal
Q1	Output	ODDR tristate enable control data output can be connected to the IOBUF/TBUF OEN signal connected to Q0, or left floating.

Parameter Description

Table 4-28 ODDRC Parameter Description

Name	Value	Default	Description
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 output clock polarity control ● 1'b0:Q1 posedge output; ● 1'b1:Q1 negedge output
INIT	1'b0	1'b0	Initial value of ODDRC output

Connection Rule

- Q0 can directly connect OBUF, or connect input port DI in IODELAY module;
- Q1 shall connect the OEN signal of IOBUF/TBUF connected by Q0, or left floating.

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5 IP Generation.

Verilog Instantiation:

```
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 Instantiation:

```
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

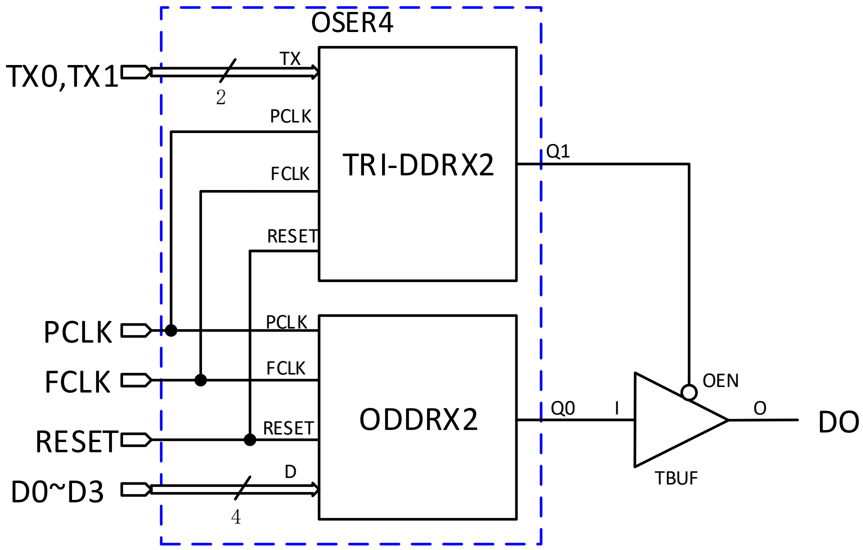
Primitive Introduction

The 4 to 1 Serializer (OSER4) is a serializer of 4 bits parallel input and 1 bit serial output.

Functional Description

OSER4 mode realizes 4:1 parallel to serial conversion, and Q0 is the serial output, Q1 is used for the OEN signal of IOBUF/TBUF connected to Q0. TX0/TX1 is the OEN input control signal of IOBUF/TBUF, and TX0/TX1 can be synchronized with the data D0~D3 through DDR. TX0/TX1 through TRI-DDRX2 is output as Q1 connected to the OEN signal of IOBUF/TBUF; D0~D3 through ODDRX2 is output as Q0 connected to the data input I of IOBUF/TBUF, and the sequence is D0, D1, D2, D3 in order. Its logic diagram is as shown in Figure 4-21.

Figure 4-21 OSER4 Logic Diagram

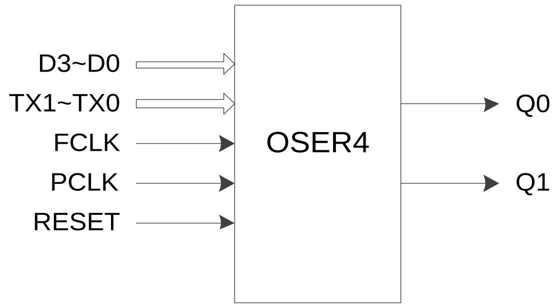


PCLK is usually obtained by FCLK frequency division:

$f_{PCLK} = 1/2\, f_{FCLK}$.

Port Diagram

Figure 4-22 OSER4 Port Diagram



Port Description

Table 4-29 OSER4 Port Description

Port	I/O	Description
D3~D0	Input	OSER4 data input signal
TX1~TX0	Input	Q1 generated by TRI-DDRX2
FCLK	Input	High speed clock input signal
PCLK	Input	Primary clock input signal
RESET	Input	Asynchronous reset input signal, active-high.
Q0	Output	OSER4 data output signal
Q1	Output	OSER4 tristate enable control data output can be connected to the IOBUF/TBUF OEN signal connected to Q0, or left floating.

Parameter Description

Table 4-30 IDES8_MEM Parameter Description

Name	Value	Default	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 output clock polarity control ● 1'b0: data posedge output ● 1'b1: data negedge output
HWL	"false", "true"	"false"	OSER4 data d_up0/1 timing relationship control ● "False ": d_up1 is one cycle ahead of d_up0. ● "True ": d_up1 and d_up0 have the same timing.

Connection Rule

- Q0 can directly connect OBUF, or connect input port DI in IODELAY module;
- Q1 shall connect the OEN signal of IOBUF/TBUF connected to Q0, or left floating.

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5IP Generation.

Verilog Instantiation:

```
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 Instantiation:

```
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

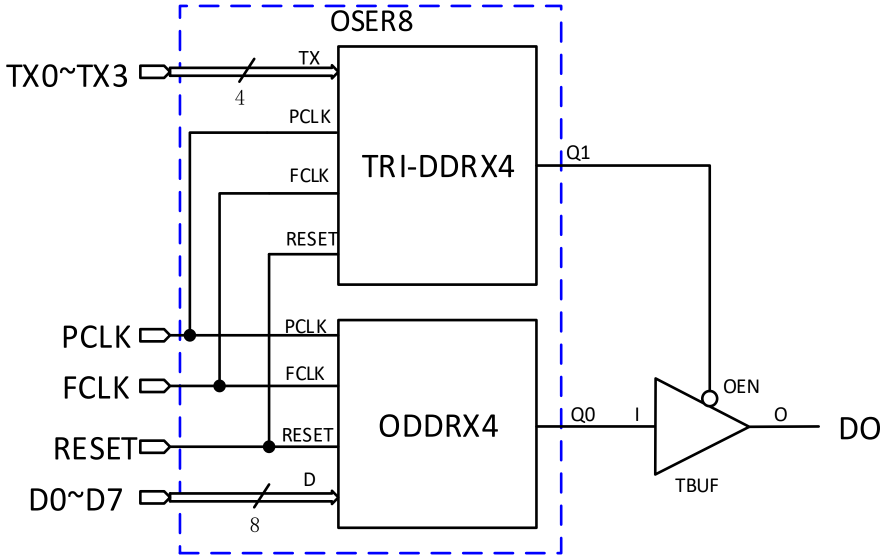
Primitive Introduction

The 8 to 1 Serializer (OSER8) is a serializer of 8 bits parallel input and 1 bit serial output.

Functional Description

OSER8 mode realizes 8:1 parallel to serial. Where Q0 is the serial output, Q1 is used for the OEN signal of IOBUF/TBUF connected to Q0. Its logic diagram is as shown in Figure 4-23.

Figure 4-23 OSER8 Logic Diagram

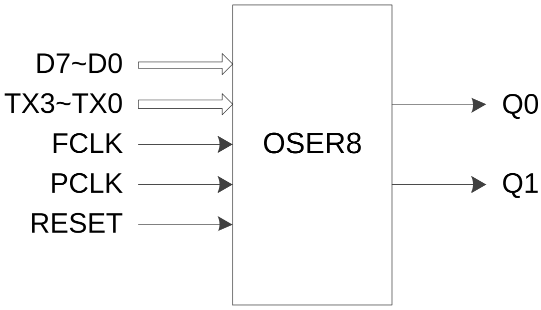


PCLK is usually obtained by FCLK frequency division：

$f_{PCLK} = 1/4\, f_{FCLK}$.

Port Diagram

Figure 4-24 OSER8 Port Diagram



Port Description

Table 4-31 OSER4 Port Description

Port	I/O	Description
D7~D0	Input	OSER8 data input signal
TX3~TX0	Input	Q1 generated by TRI-DDRX4
FCLK	Input	High speed clock input signal
PCLK	Input	Primary clock input signal
RESET	Input	Asynchronous reset input signal, active-high.
Q0	Output	OSER8 data output signal
Q1	Output	OSER8 tristate enable control data output can be connected to the IOBUF/TBUF OEN signal connected to Q0, or left floating.

Parameter Description

Table 4-32 IDES8_MEM Parameter Description

Name	Value	Default	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 output clock polarity control ● 1'b0: data posedge output ● 1'b1: data negedge output
HWL	"false", "true"	"false"	OSER8 data d_up0/1 timing relationship control ● "false ": d_up1 is one cycle ahead of d_up0. ● "true ": d_up1 and d_up0 have the same timing.

Connection Rule

- Q0 can directly connect OBUF, or connect input port DI in IODELAY module.

- Q1 shall connect the OEN signal of IOBUF/TBUF connected to Q0, or left floating.

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5 IP Generation.

Verilog Instantiation:

```
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 Instantiation:

```
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

Primitive Introduction

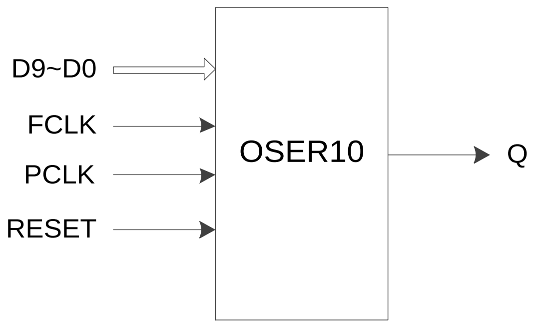
The 10 to 1 Serializer (OSER10) is a serializer of 10 bits parallel input and 1 bit serial output.

Functional Description

OSER10 mode realizes 10:1 parallel to serial conversion. PCLK is usually obtained by FCLK frequency division, $f_{PCLK} = 1/5\, f_{FCLK}$.

Port Diagram

Figure 4-25 OSER10 Port Diagram



Port Description

Table 4-33 OSER10 Port Description

Port	I/O	Description
D9~D0	Input	OSER10 data input signal
FCLK	Input	High speed clock input signal
PCLK	Input	Primary clock input signal
RESET	Input	Asynchronous reset input signal, active-high.
Q	Output	OSER10 data output signal

Parameter Description

Table 4-34 OSER10 Parameter Description

Name	Value	Default	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Rule

Q can directly connect to OBUF, or connect to input port DI in IODELAY module.

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5 IP Generation.

Verilog Instantiation:

```
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 Instantiation:

```
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

Primitive Introduction

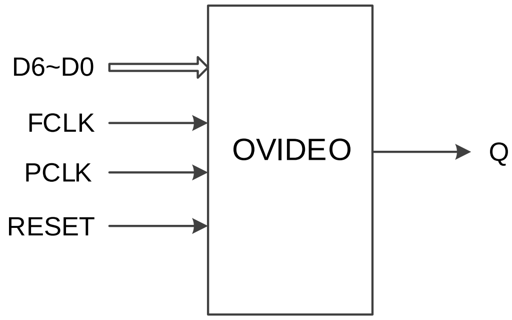
The 7 to 1 Serializer (OVIDEO) is a serializer of 7 bits parallel input and 1 bit serial output,

Functional Description

OVIDEO mode realizes 7:1 parallel to serial conversion. PCLK is usually obtained by FCLK frequency division: $f_{PCLK} = 1/3.5\, f_{FCLK}$.

Port Diagram

Figure 4-26 OVIDEO Port Diagram



Port Description

Table 4-35 OVIDEO Port Description

Port	I/O	Description
D6~D0	Input	OVIDEO data input signal
FCLK	Input	High speed clock input signal
PCLK	Input	Primary clock input signal
RESET	Input	Asynchronous reset input signal, active-high.
Q	Output	OVIDEO data output signal

Parameter Description

Table 4-36 OVIDEO Parameter Description

Name	Value	Default	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Rule

Q can directly connect to OBUF, or connect to input port DI in IODELAY module;

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5 IP Generation.

Verilog Instantiation:

```
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 Instantiation:

```
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

Primitive Introduction

The 16 to 1 Serializer (OSER16) is a serializer of 16 bits parallel input and 1 bit serial output.

Devices Supported

Table 4-37 OSER16 Devices Supported

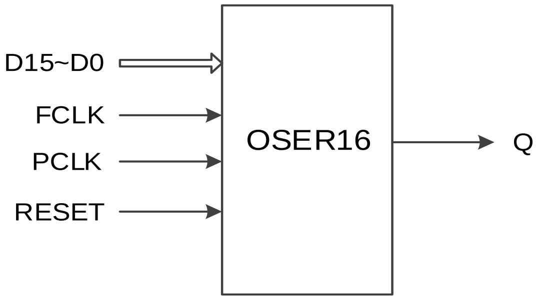
Product Family	Series	Device
LittleBee®	GW1N	GW1N-1S, GW1N-9, GW1N-9C, GW1N-2, GW1N-1P5, GW1N-2B, GW1N-1P5B
	GW1NR	GW1NR-9, GW1NR-9C, GW1NR-2, GW1NR-2B
	GW1NS	GW1NS-4, GW1NS-4C
	GW1NSER	GW1NSER-4C
	GW1NSR	GW1NSR-4, GW1NSR-4C

Functional Description

OSER16 mode realizes 16:1 parallel to serial conversion. PCLK is usually obtained by FCLK frequency division: $f_{PCLK} = 1/8\, f_{FCLK}$.

Port Diagram

Figure 4-27 OSER16 Port Diagram



Port Description

Table 4-38 OSER16 Port Description

Port	I/O	Description
D15~D0	Input	OSER16 data input signal
FCLK	Input	High speed clock input signal
PCLK	Input	Primary clock input signal
RESET	Input	Asynchronous reset input signal, active-high.
Q	Output	IDES8_MEM data output signal

Parameter Description

Table 4-39 OSER16 Parameter Description

Name	Value	Default	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Rule

Q can directly connect to OBUF, or connect to input port DI in IODELAY module;

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5 IP Generation.

Verilog Instantiation:

```
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 Instantiation:

```
  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

Primitive Introduction

Dual Data Rate Output with Memory (ODDR_MEM) realizes double data rate output with memory.

Devices Supported

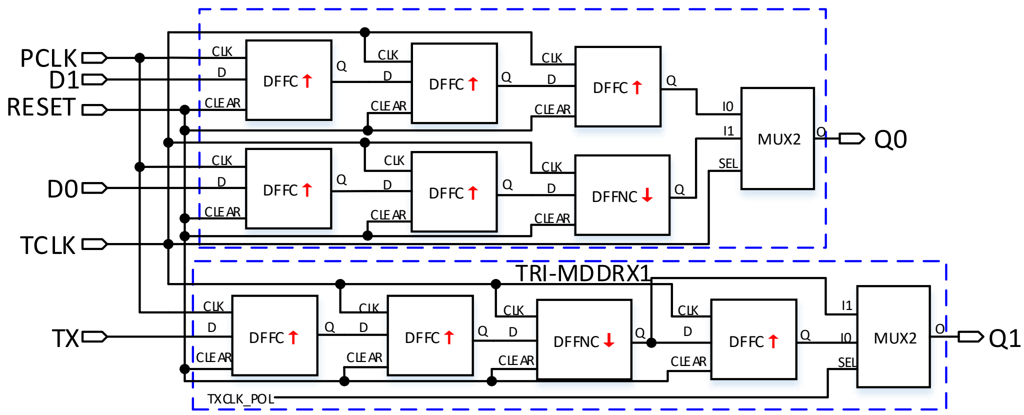
Table 4-40 ODDR_MEM Devices Supported

Product Family	Series	Device
Arora	GW2A	GW2A-18, GW2A-18C, GW2A-55, GW2A-55C
	GW2AN	GW2AN-55C
	GW2AR	GW2AR-18,GW2AR-18C
	GW2ANR	GW2ANR-18C

Functional Description

ODDR_MEM mode is used for transferring double data rate signals from FPGA devices. Unlike ODDR, the output double data rate with memory (ODDR_MEM) needs to be used with DQS. TCLK connects to the DQSW0 or DQSW270 of DQS output signal, and outputs data from ODDR_MEM according to the TCLK clock edge. Where Q0 is the double rate data output, Q1 is used for the OEN signal of IOBUF/TBUF connected to Q0. Its logic diagram is as shown in Figure 4-28.

Figure 4-28 ODDR_MEM Logic Diagram

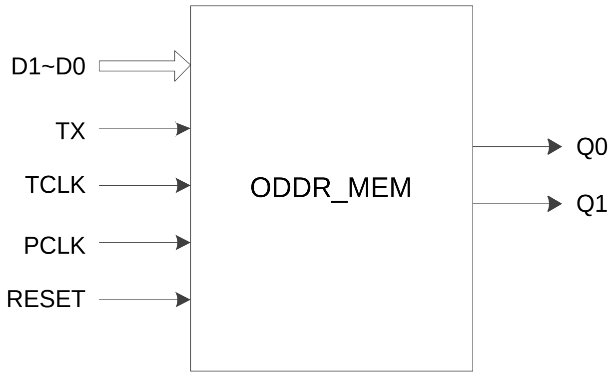


The frequency relation between PCLK and TCLK is $f_{PCLK} = f_{TCLK}$.

You can determine the phase relationship between PCLK and TCLK according to DLLSTEP and WSTEP value of DQS.

Port Diagram

Figure 4-29 ODDR_MEM Port Diagram



Port Description

Table 4-41 ODDR_MEM Port Description

Port	I/O	Description
D1~D0	Input	ODDR_MEM data input signal
TX	Input	Q1 generated by TRI-MDDRX1
TCLK	Input	Clock input signal from DQSW0 or DQSW270 in DQS module
PCLK	Input	Primary clock input signal
RESET	Input	Asynchronous reset input signal, active-high.
Q0	Output	ODDR_MEM data output signal
Q1	Output	ODDR_MEM tristate enable control data output can be connected to the IOBUF/TBUF OEN signal connected to Q0, or left floating.

Parameter Description

Table 4-42 ODDR_MEM Parameter Description

Name	Value	Default	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 output clock polarity control ● 1'b0: data posedge output ● 1'b1: data negedge output
TCLK_SOURCE	"DQSW", "DQSW270"	"DQSW"	TCLK source selection ● "DQSW" comes from DQSW0 in DQS module. ● "DQSW270" comes from DQSW270 from DQS module.

Connection Rule

- Q0 can directly connect to OBUF, or connect to input port DI in IODELAY module;
- Q1 shall connect to the OEN signal of IOBUF/TBUF connected to Q0, or left floating.
- TCLK needs DQSW0 or DQSW270 from DQS module and you need to configure the corresponding parameters.

Primitive Instantiation

Verilog Instantiation:

```
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 Instantiation:

```
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

Primitive Introduction

The 4 to 1 Serializer with Memory (OSER4_MEM) realizes 4:1 parallel serial conversion with memory.

Devices Supported

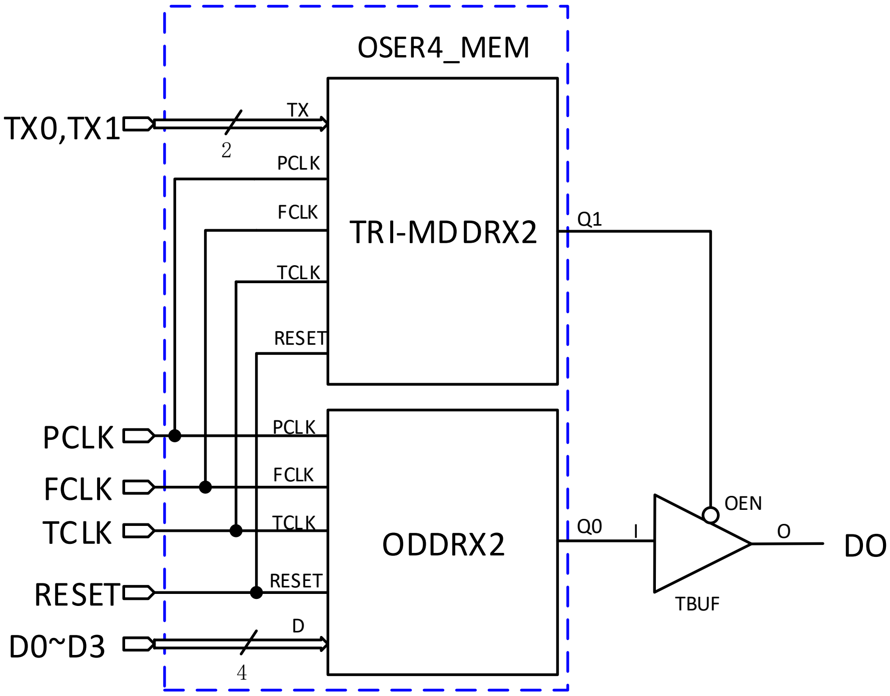
Table 4-43 OSER4_MEM Devices Supported

Product Family	Series	Device
Arora	GW2A	GW2A-18, GW2A-18C, GW2A-55, GW2A-55C
	GW2AN	GW2AN-55C
	GW2AR	GW2AR-18,GW2AR-18C
	GW2ANR	GW2ANR-18C

Functional Description

OSER4_MEM realizes 4:1 parallel serial conversion. The TCLK connects to the output signal DQSW0 or DQSW270 of DQS, and outputs data from the OSER4_MEM according to the TCLK clock edge, and Q0 is the serial output, Q1 is used for the OEN signal of IOBUF/TBUF connected to Q0. Its logic diagram is as shown in Figure 4-30.

Figure 4-30 OSER4_MEM Logic Diagram

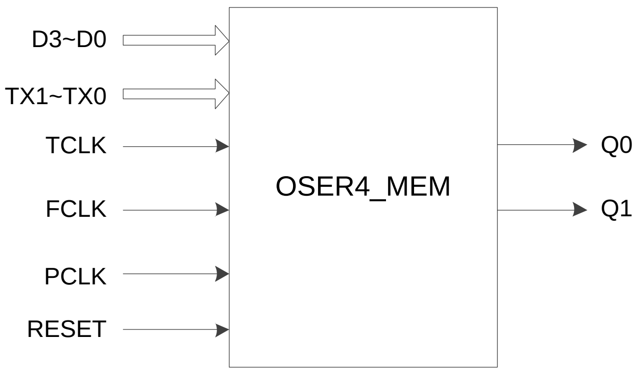


The frequency relation among PCLK, FCLK and TCLK is $f_{PCLK} = 1/2\, f_{FCLK} = 1/2\, f_{TCLK}$.

You can determine the phase realationship between FCLK and TCLK according to the DLLSTEP and WSTEP values of DQS.

Port Diagram

Figure 4-31 OSER4_MEM Diagram



Port Description

Table 4-44 OSER4_MEM Port Description

Port	I/O	Description
D3~D0	Input	OSER4_MEM data input signal
TX1~TX0	Input	Q1 generated by TRI-MDDRX2

Port	I/O	Description
TCLK	Input	Clock input signal from DQSW0 or DQSW270 in DQS module
FCLK	Input	High speed clock input signal
PCLK	Input	Primary clock input signal
RESET	Input	Asynchronous reset input signal, active-high.
Q0	Output	OSER4_MEM data output signal
Q1	Output	OSER4_MEM tristate enable control data output can be connected to the IOBUF/TBUF OEN signal connected to Q0, or left floating.

Parameter Description

Table 4-45 OSER4_MEM Parameter Description

Name	Value	Default	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 output clock polarity control ● 1'b0: data posedge output ● 1'b1: data negedge output
TCLK_SOURCE	"DQSW", "DQSW270"	" DQSW "	TCLK source selection ● "DQSW" comes from DQSW0 in DQS module. ● "DQSW270" comes from DQSW270 from DQS module.
HWL	"false", "true"	"false"	OSER4_MEM data d_up0/1 timing relationship control ● "False ": d_up1 is one cycle ahead of d_up0. ● "True ": d_up1 and d_up0 have the same timing.

Connection Rule

- Q0 can directly connect to OBUF, or connect to input port DI in IODELAY module.
- Q1 shall connect to the OEN signal of IOBUF/TBUF connected to Q0, or suspend.
- TCLK needs DQSW0 or DQSW270 from DQS module and you need to configure the corresponding parameters.

Primitive Instantiation

Verilog Instantiation:

```
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 Instantiation:

```
    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

Primitive Introduction

The 8 to 1 Serializer with Memory (OSER8_MEM) realizes 8:1 parallel serial with memory.

Devices Supported

Table 4-46 OSER8_MEM Devices Supported

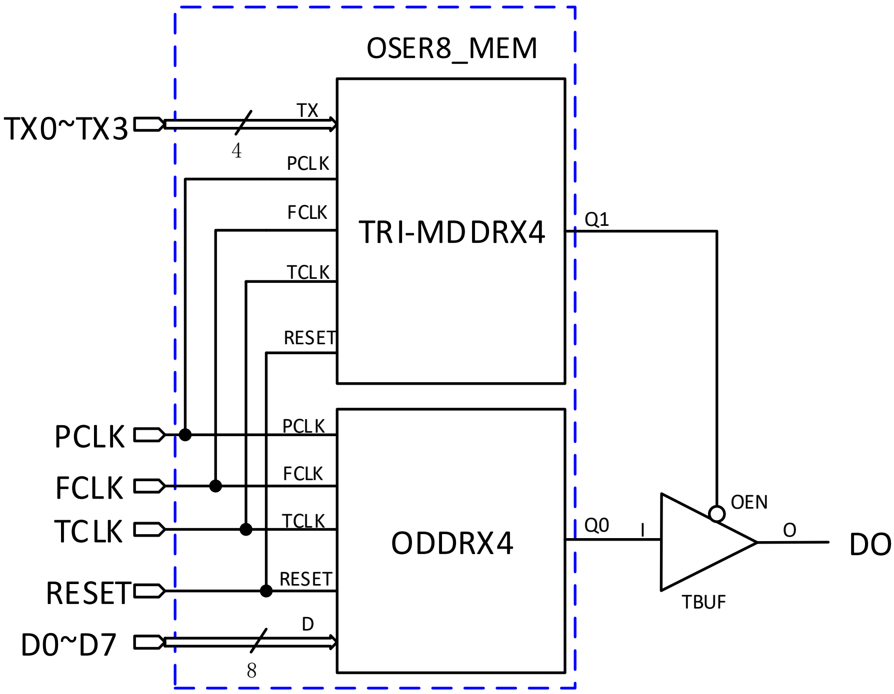
Product Family	Series	Device
Arora	GW2A	GW2A-18, GW2A-18C, GW2A-55, GW2A-55C
	GW2AN	GW2AN-55C

Product Family	Series	Device
	GW2AR	GW2AR-18,GW2AR-18C
	GW2ANR	GW2ANR-18C

Functional Description

OSER8_MEM mode realizes 8:1 parallel serial conversion. The TCLK connects the output signal DQSW0 or DQSW270 of DQS, and outputs data from the OSER8_MEM according to the TCLK clock edge, and Q0 is the serial output, Q1 is used for the OEN signal of IOBUF/TBUF connected to Q0. Its logic diagram is as shown in Figure 4-32.

Figure 4-32 OSER8_MEM Logic Diagram

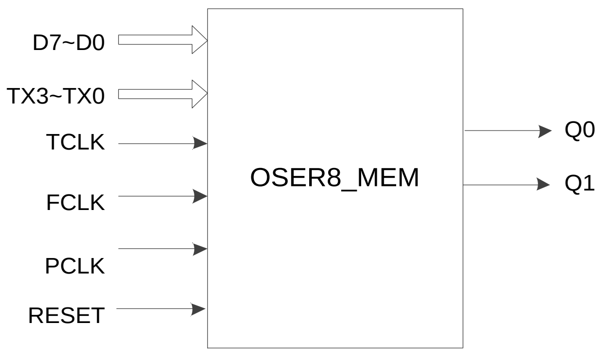


The frequency relation between PCLK, FCLK and TCLK is $f_{PCLK} = 1/4\ f_{FCLK} = 1/4\ f_{TCLK}$.

You can determine the phase realationship between FCLK and TCLK according to DLLSTEP and WSTEP values of DQS.

Port Diagram

Figure 4-33 OSER8_MEM Port Diagram



Port Description

Table 4-47 OSER4_MEM Port Description

Port Name	I/O	Description
D7~D0	Input	OSER8_MEM data input signal
TX3~TX0	Input	Q1 generated by TRI-MDDRX4
TCLK	Input	Clock input signal from DQSW0 or DQSW270 in DQS module
FCLK	Input	High speed clock input signal
PCLK	Input	Primary clock input signal
RESET	Input	Asynchronous reset input signal, active-high.
Q0	Output	OSER8_MEM data output signal
Q1	Output	OSER8_MEM tristate enable control data output can be connected to the IOBUF/TBUF OEN signal connected to Q0, or left floating.

Parameter Description

Table 4-48 OSER4_MEM Parameter Description

Name	Value	Default	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable Local Reset
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 output clock polarity control ● 1'b0: data posedge output ● 1'b1: data negedge output
TCLK_SOURCE	"DQSW", "DQSW270"	" DQSW "	TCLK source selection ● "DQSW" comes from DQSW0 in DQS module. ● "DQSW270" comes from DQSW270 from DQS module.

Name	Value	Default	Description
HWL	"false", "true"	"false"	OSER8_MEM data d_up0/1 timing relationship control ● "false ": d_up1 is one cycle ahead of d_up0. ● "true ": d_up1 and d_up0 have the same timing.

Connection Rule

- Q0 can directly connect to OBUF, or connect to input port DI in IODELAY module.
- Q1 shall connect to the OEN signal of IOBUF/TBUF connected to Q0, or left floating.
- TCLK needs DQSW0 or DQSW270 from DQS module and you need to configure the corresponding parameters.

Primitive Instantiation

Verilog Instantiation:

```
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 Instantiation:

```
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 Delay Module

4.4.1 IODELAY

Primitive Introduction

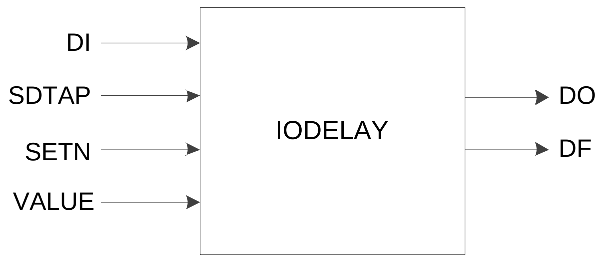
Input/Output delay (IODELAY) is a programmable delay unit in IO module.

Functional Description

Each IO contains an IODELAY module, providing a total of 128 (0~127) delays. The GW1N series of FPGA has a single-step delay time of about 30ps. And the GW2A series of FPGA has a single-step delay time of about 18ps. IODELAY can be used for input or output of I/O logic, but not for both at the same time.

Port Diagram

Figure 4-34 IODELAY Port Diagram



Port Description

Table 4-49 IODELAY Port Description

Port	I/O	Description
DI	Input	Data input signal
SDTAP	Input	Controls loading static delay step ● 0: loads static delay ● 1: adjusts the dynamic delay
SETN	Input	Sets the direction of dynamic delay adjustment ● 0: Increases delay ● 1: Decreases delay
VALUE	Input	VALUE is the delay value of negedge dynamic adjustment, and it moves one delay step per pulse.
DO	Output	Data output signal
DF	Output	An output flag that represents under-flow or over-flow in dynamic delay adjustment

Parameter Description

Table 4-50 IODELAY Parameter Description

Name	Value	Default	Description
C_STATIC_DLY	0~127	0	Controls the static delay step

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

```
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

Primitive Introduction

Input/Output delay (IODELAY) is a programmable delay unit in IO module.

Devices Supported

Table 4-51 IODELAYC Devices Supported

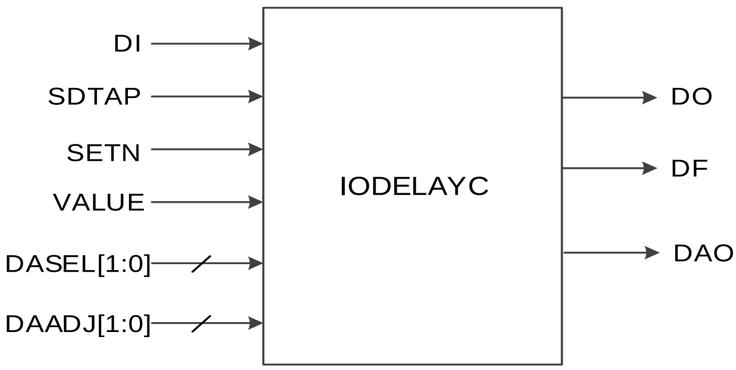
Product Family	Series	Device
LittleBee®	GW1N	GW1N-9C
	GW1NR	GW1NR-9C

Functional Description

Each IO contains the IODELAYC module, which provides a total of 128 (0 to 127) delays, adding more delay adjustments compared to IODELAY. IODELAYC can only be used for I/O logic input, not for I/O logic output.

Port Diagram

Figure 4-35 IODELAYC Port Diagram



Port Description

Table 4-52 IODELAYC Port Description

Port	I/O	Description
DI	Input	Data input signal
SDTAP	Input	Controls loading static delay step ● 0: loads static delay ● 1: adjusts the dynamical delay

Port	I/O	Description
SETN	Input	Sets the direction of dynamical delay adjustment ● 0: Increases delay ● 1: Decreases delay
VALUE	Input	VALUE is the delay value of negedge dynamical adjustment, and each pulse moves one delay step.
DASEL[1:0]	Input	Dynamically controls DAO delay mode
DAADJ[1:0]	Input	Dynamically controls the delay value of the DAO relative to the DO.
DO	Output	Data output signal
DAO	Output	Output signal of data delay adjustment
DF	Output	An output flag that represents under-flow or over-flow in dynamical delay adjustment.

Parameter Description

Table 4-53 IODELAYC Parameter Description

Name	Value	Default	Description
C_STATIC_DLY	0~127	0	Controls the static delay step
DYN_DA_SEL	"True"/ "false"	false	● false: selects parameter DA_SEL to statically control DAO delay mode. ● true: selects the signal DASEL to dynamically control DAO delay mode.
DA_SEL	2'b00~2'b11	2'b00	Static control of DAO delay mode

Primitive Instantiation

Verilog Instantiation:

```
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 Instantiation:

```
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

Primitive Introduction

Input/Output delay (IODELAYB) is a programmable delay unit in IO module.

Devices Supported

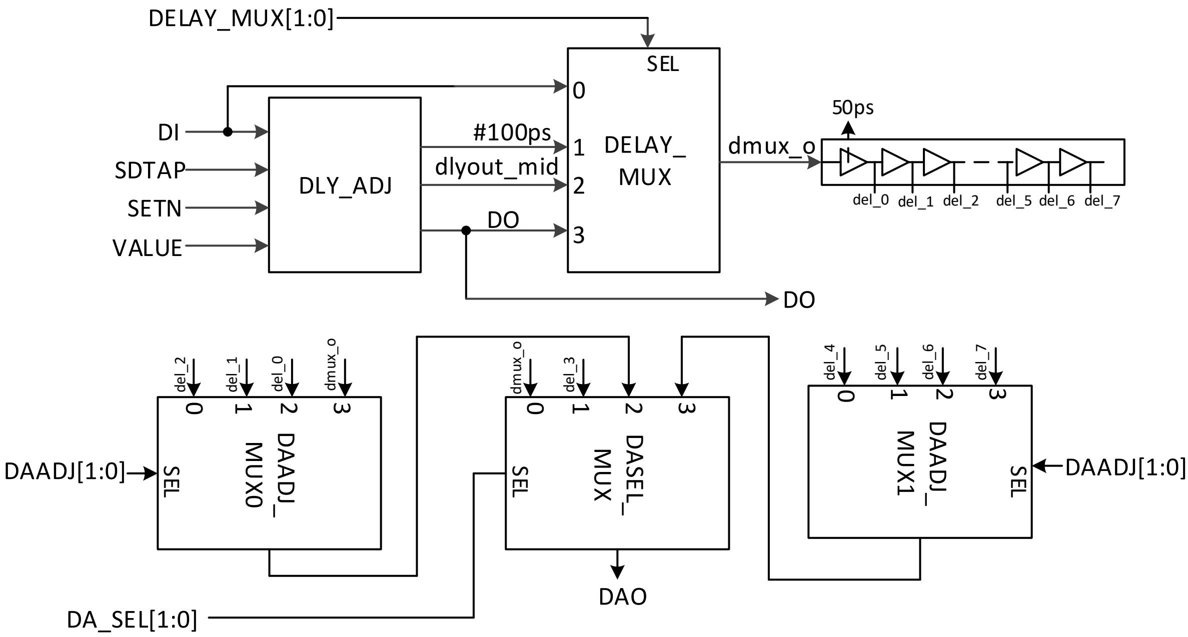
Table 4-54 Devices Supported

Family	Series	Device
LittleBee®	GW1N	GW1N-2, GW1N-1P5, GW1N-2B, GW1N-1P5B
	GW1NR	GW1NR-2, GW1NR-2B

Functional Description

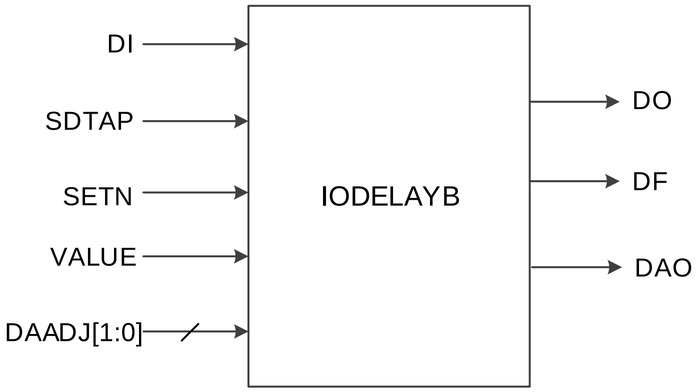
Each IO contains the IODELAYB module, which provides a total of 128 (0 to 127) delay. IODELAYB adds more delay adjustments compared to IODELAY. IODELAYB can only be used for I/O logic input, not for I/O logic output, and the diagram is as shown in Figure 4-36.

Figure 4-36 IODELAYB Diagram



Port Diagram

Figure 4-37 IODELAYB Port Diagram



Port Description

Table 4-55 IODELAYB Port Description

Port Name	I/O	Description
DI	Input	Data input signal
SDTAP	Input	Controls loading static delay step ● 0: loads static delay ● 1: adjusts the dynamical delay
SETN	Input	Sets the direction of dynamical delay adjustment ● 0: Increases delay ● 1: Decreases delay
VALUE	Input	VALUE is the delay value of negedge dynamical adjustment, and each pulse moves one delay step.
DAADJ[1:0]	Input	Dynamically controls the delay value of the DAO relative to the DO
DO	Output	Data output signal
DAO	Output	Output signal of data delay adjustment
DF	Output	An output flag that represents under-flow or over-flow in dynamical delay adjustment.

Parameter Description

Table 4-56 IODELAYB Parameter Description

Name	Value	Default	Description
C_STATIC_DLY	0~127	0	Controls Static delay step
DELAY_MUX	2'b00~2'b11	2'b00	Delay MUX options: ● 2'b00:dmux_o=DI ● 2'b01:#100ps dmux_o=DI ● 2'b10:dmux_o=dlyout_mid ● 2'b11:dmux_o=DO
DA_SEL	2'b00~2'b11	2'b00	DAO delay mode in static control

Note!

When IODELAYB used, the connection of parameters DELAY_MUX and DA_SEL are as follows:

- DELAY_MUX:2/3 -> DA_SEL:0/1. When DELAY_MUX is 2 or 3, DA_SEL can be 0 or 1.
- DELAY_MUX:0/1 -> DA_SEL:0/2/3. When DELAY_MUX is 0 or 1, DA_SELcan be 0, 2 or 3.

Connection Rule

DO can not connect to IDDR/IDES, and DAO can only connect to data input of IDDR/IDES.

Primitive Instantiation

Verilog Instantiation:

```
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 Instantiation：

```
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 IEM

Primitive Introduction

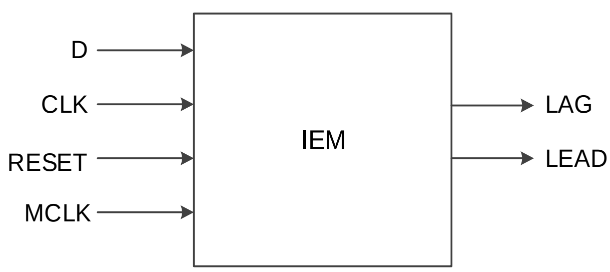
Input Edge Monitor (IEM) is a sampling module in IO module.

Functional Description

IEM is used for sampling data edge, which can be used together with delay module to adjust the dynamic sampling window for DDR mode.

Port Diagram

Figure 4-38 IEM Port Diagram



Port Description

Table 4-57 IEM Port Description

Port	I/O	Description
D	Input	Data input signal
CLK	Input	Clock input signal
RESET	Input	Asynchronous reset input signal, active-high.
MCLK	Input	IEM detecting clock, from user logic, acts on output flag.
LAG	Output	The lag flag after IEM edge comparison
LEAD	Output	The lead flag after IEM edge comparison

Parameter Description

Table 4-58 IEM Parameter Description

Name	Value	Default	Description
WINSIZE	"SMALL","MIDSMALL", "MIDLARGE","LARGE"	"SMALL"	Window size setting
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Primitive Instantiation

Verilog Instantiation:

```
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 Instantiation:

```
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
       );
```

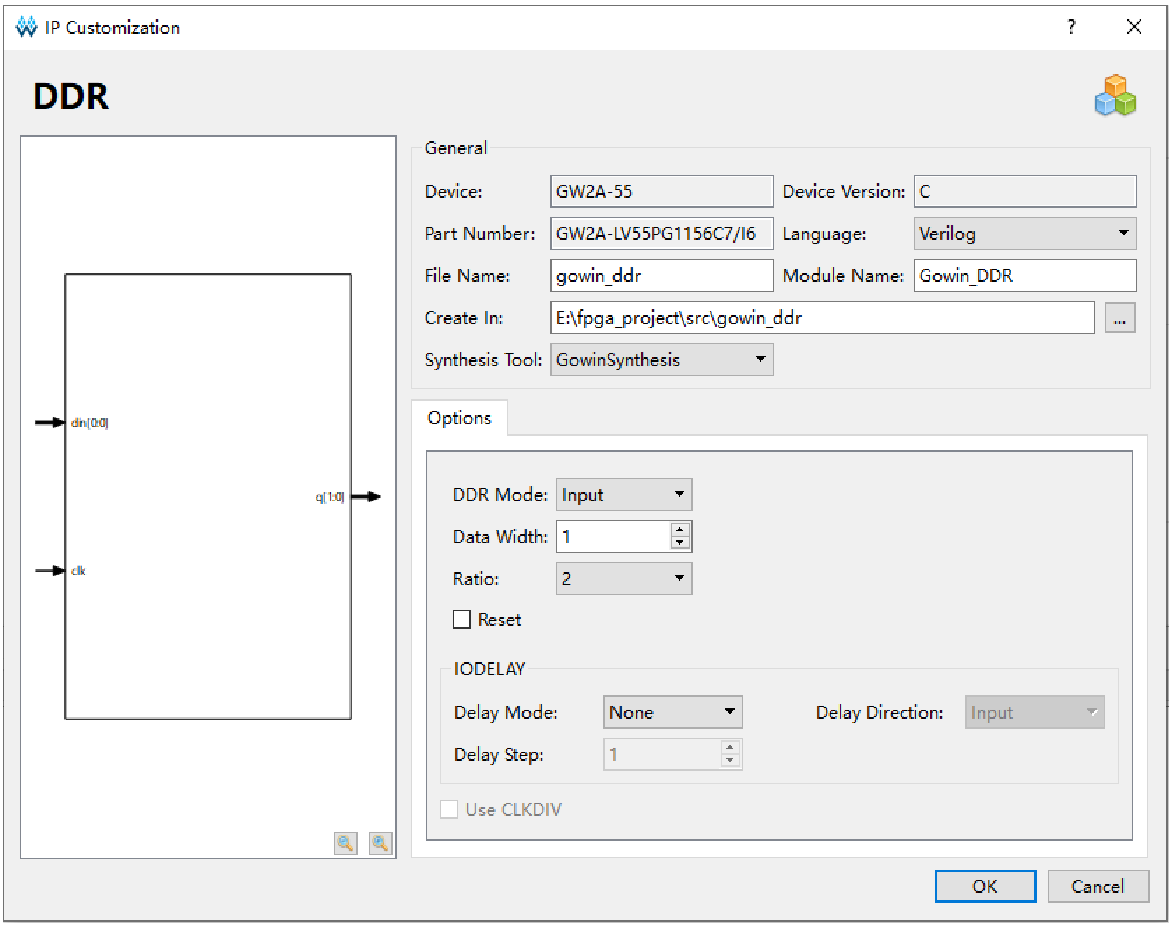
5 IP Generation

The software only supports DDR at present. Click DDR in the IP Core Generator interface, and a summary of DDR will be displayed on the right side of the interface.

5.1 IP Configuration

Double-click "DDR", and the "IP Customization" window pops up. This includes the "General", "Options", and port diagram, as shown in Figure 5-1.

Figure 5-1 IP Customization of DDR



1. General: The General configuration box is used to configure the generated IP design file.
   - Device: Displays the configured Device.
   - Device Version: Displays the configured device version.
   - Part Number: Displays the configured Part Number.
   - Language: Hardware description language used to generate the IP design files. Click the drop-down list to select the language, including Verilog and VHDL.
   - Synthesis Tool: Selects synthesis tools.
   - Module Name: The module name of the generated IP design files. Enter the module name in the text box. Module name cannot be the same as the primitive name. If it is the same, an error will be reported.
   - File Name: The name of the generated IP design files. Enter the file name in the text box.
   - Create In: The path in which the generated IP files will be stored. Enter the target path in the box or select the target path by clicking the option.
2. Options: The Options configuration box is used to customize the IP, as shown Figure 5-1.
   - DDR Mode: Configures DDR mode, including input, output, tristate and bi-directional state;
   - Data Width: Configures the data width of the DDR. The range is 1~64;
   - Ratio: DDR data conversion ratio, including 2,4,7,8,10,16;
   - Reset: When Ratio is 2, this option can be enabled or disabled, and IDDRC or ODDRC will be instantiated when enabled;
   - IODELAY: Configures whether DDR uses a delay module;
     - "Delay Mode": Configures the delay mode. "None" means no IODELAY; "Dynamic" means using IODELAY and adjusting the delay step dynamically; "Static" means using IODELAY and adjusting the delay step statically.
     - "Delay Step": Selects the number of steps to statically adjust the delay, ranging 1 to 128.
     - "Delay Direction": In bidirectional mode, When DDR Mode is "bidirectional", if IODELAY is used, select whether IODELAY is connected to input or output.
   - Use CLKDIV: CLKDIV will be instantiated and the frequency of fclk will be divided when CLKDIV is enabled. When Ratio is 2, it cannot be checked.
3. Port Diagram: The port diagram displays a sample diagram of IP Core configuration, as shown in Figure 5-1.

5.2 IP Generation Files

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

- The IP design file "gowin_ddr.v" is a complete verilog module, which generates DDR modules with corresponding functions according to the configuration.
- "Gowin_ddr_tmp.v" is the template file;
- "gowin_padd.ipc" file is IP configuration file. You can load the file to configure the IP.

Note!

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