




Arora V Programmable IO (GPIO) **User Guide**

UG304-1.1E, 05/25/2023

Copyright © 2023 Guangdong Gowin Semiconductor Corporation. All Rights Reserved.

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

Disclaimer

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

Revision History

Date	Version	Description
04/20/2023	1.0E	Initial version published.
05/25/2023	1.1E	Chapter 5 IP Generation added.

Contents

Contents	i
List of Figures	iv
List of Tables	vi
1 About This Guide	1
1.1 Purpose	1
1.2 Related Documents	1
1.3 Terminology and Abbreviations	1
1.4 Support and Feedback	2
2 GPIO Overview	3
3 Input/Output Buffer	5
3.1 GPIO Level Standard	5
3.2 GPIO Banking	5
3.3 Power Supply Requirements	5
3.3.1 LVCMOS Buffer Configuration	6
3.3.2 Differential Buffer Configuration	6
3.4 Emulated Differential Circuit Matching Network	6
3.4.1 Emulated LVDS	6
3.4.2 Emulated LVPECL	7
3.4.3 Emulated RSDS	7
3.4.4 Emulated BLVDS	7
3.5 GPIO Software Configuration	8
3.5.1 Location	8
3.5.2 Level Standard	8
3.5.3 Drive Strength	8
3.5.4 Pull Up/Pull Down	8
3.5.5 Reference Voltage	8
3.5.6 Hysteresis	9
3.5.7 Open Drain	9
3.5.8 Termination Resistors for Single-ended Signal	9
3.5.9 Termination Resistor for Differential Signal	9
3.5.10 PCI Clamp	9

3.5.11 Pull-up/Pull-down Strength	9
3.6 GPIO Primitive	9
3.6.1 IBUF	9
3.6.2 OBUF	10
3.6.3 TBUF	11
3.6.4 IOBUF	13
3.6.5 LVDS Input Buffer	14
3.6.6 LVDS Output Buffer	15
3.6.7 LVDS Tristate Buffer	17
3.6.8 LVDS Inout Buffer	19
3.6.9 MIPI_IBUF	20
3.6.10 MIPI_OBUF_A	22
3.6.11 I3C_IOBUF	24
3.6.12 IBUF_R	25
3.6.13 IOBUF_R	26
3.6.14 ELVDS_IOBUF_R	28
3.6.15 TLVDS_IBUF_ADC	29
4 Input/Output Logic	31
4.1 SDR Mode	32
4.1.1 DFFSE	32
4.1.2 DFFRE	34
4.1.3 DFFPE	35
4.1.4 DFFCE	37
4.1.5 DLCE	39
4.1.6 DLPE	40
4.2 DDR Mode Input Logic	42
4.2.1 IDDR	42
4.2.2 IDDR_C	44
4.2.3 IDES4	45
4.2.4 IDES8	48
4.2.5 IDES10	50
4.2.6 IVIDEO	53
4.2.7 IDES16	55
4.2.8 IDDR_MEM	59
4.2.9 IDES4_MEM	61
4.2.10 IDES8_MEM	64
4.2.11 IDES14	67
4.2.12 IDES32	70
4.2.13 OSIDES32	75

4.3 DDR Mode Output Logic	78
4.3.1 ODDR	78
4.3.2 ODDRC	81
4.3.3 OSER4	83
4.3.4 OSER8	87
4.3.5 OSER10	91
4.3.6 OVIDEO	93
4.3.7 OSER16	95
4.3.8 ODDR_MEM	98
4.3.9 OSER4_MEM	101
4.3.10 OSER8_MEM	105
4.4 Delay Module	110
4.4.1 IODELAY	110
5 IP Generation	113
5.1 IP Configuration	113
5.2 IP Generation Files	115

List of Figures

Figure 2-1 IOB Structure View	3
Figure 3-1 LVDS25E Matching Network	7
Figure 3-2 LVPECL Matching Network	7
Figure 3-3 RSDSE Matching Network	7
Figure 3-4 BLVDS Matching Network	8
Figure 3-5 IBUF Port Diagram	10
Figure 3-6 OBUF Port Diagram	11
Figure 3-7 TBUF Port Diagram	12
Figure 3-8 IOBUF Port Diagram	13
Figure 3-9 TLVDS_IBUF Port Diagram	14
Figure 3-10 TLVDS_OBUF/ELVDS_OBUF Port Diagram	15
Figure 3-11 TLVDS_TBUF/ELVDS_TBUF Port Diagram	17
Figure 3-12 TLVDS_IOBUF/ELVDS_IOBUF Port Diagram	19
Figure 3-13 MIPI_IBUF Port Diagram	20
Figure 3-14 MIPI_OBUF_A Port Diagram	23
Figure 3-15 I3C_IOBUF Port Diagram	24
Figure 3-16 IBUF_R Port Diagram	25
Figure 3-17 IOBUF_R Port Diagram	26
Figure 3-18 ELVDS_IOBUF_R Port Diagram	28
Figure 3-19 TLVDS_IBUF_ADC Port Diagram	29
Figure 4-1 I/O Logic View-Output	31
Figure 4-2 I/O Logic View-Input	32
Figure 4-3 DFFSE Port Diagram	32
Figure 4-4 DFFRE Port Diagram	34
Figure 4-5 DFFPE Port Diagram	36
Figure 4-6 DFFCE Port Diagram	37
Figure 4-7 DLCE Port Diagram	39
Figure 4-8 DLPE Port Diagram	41
Figure 4-9 IDDR Logic Diagram	42
Figure 4-10 IDDR Timing Diagram	43
Figure 4-11 IDDR Port Diagram	43
Figure 4-12 IDDR Port Diagram	44

Figure 4-13 CALIB Timing Diagram	46
Figure 4-14 IDES4 Port Diagram	46
Figure 4-15 IDES8 Port Diagram	48
Figure 4-16 IDES10 Port Diagram	51
Figure 4-17 IVIDEO Port Diagram	53
Figure 4-18 IDES16 Port Diagram	56
Figure 4-19 IDDR_MEM Port Diagram	59
Figure 4-20 IDES4_MEM Port Diagram	61
Figure 4-21 IDES8_MEM Port Diagram	64
Figure 4-22 IDES14 Port Diagram	67
Figure 4-23 IDES32 Port Diagram	70
Figure 4-24 OSIDES32 Port Diagram	75
Figure 4-25 ODDR Logic Diagram	78
Figure 4-26 ODDR Timing Diagram	79
Figure 4-27 ODDR Port Diagram	79
Figure 4-28 ODDRC Logic Diagram	81
Figure 4-29 ODDRC Port Diagram	81
Figure 4-30 OSER4 Logic Diagram	84
Figure 4-31 OSER4 Port Diagram	84
Figure 4-32 OSER8 Logic Diagram	87
Figure 4-33 OSER8 Port Diagram	87
Figure 4-34 OSER10 Port Diagram	91
Figure 4-35 OVIDEO Port Diagram	93
Figure 4-36 OSER16 Port Diagram	96
Figure 4-37 ODDR_MEM Logic Diagram	99
Figure 4-38 ODDR_MEM Port Diagram	99
Figure 4-39 OSER4_MEM Logic Diagram	102
Figure 4-40 OSER4_MEM Diagram	102
Figure 4-41 OSER8_MEM Logic Diagram	106
Figure 4-42 OSER8_MEM Port Diagram	106
Figure 4-43 IODELAY Port Diagram	110
Figure 5-1 DDR IP Customization	113

List of Tables

Table 1-1 Abbreviations and Terminology	1
Table 3-1 IBUF Port Description	10
Table 3-2 OBUF Port Description.....	11
Table 3-3 TBUF Port Description	12
Table 3-4 IOBUF Port Description.....	13
Table 3-5 TLVDS_IBUF Port Description	14
Table 3-6 TLVDS_OBUF/ELVDS_OBUF Port Description	15
Table 3-7 TLVDS_TBUF/ELVDS_TBUF Port Description.....	17
Table 3-8 TLVDS_IOBUF/ELVDS_IOBUF Port Description	19
Table 3-9 MIPI_IBUF Port Description	21
Table 3-10 MIPI_OBUF_A Device Supported	22
Table 3-11 MIPI_OBUF_A Port Description	23
Table 3-12 I3C_IOBUF Port Description	24
Table 3-13 IBUF_R Port Description.....	25
Table 3-14 IOBUF_R Port Description	26
Table 3-15 ELVDS_IOBUF_R Port Description	28
Table 3-16 TLVDS_IBUF_ADC Device Supported	29
Table 3-17 TLVDS_IBUF_ADC Port Description	30
Table 4-1 DFFSE Port Description.....	32
Table 4-2 DFFSE Parameter Description	33
Table 4-3 DFFRE Port Description.....	34
Table 4-4 DFFRE Parameter Description	34
Table 4-5 DFFPE Port Description.....	36
Table 4-6 DFFPE Parameter Description	36
Table 4-7 DFFCE Port Description.....	37
Table 4-8 DFFCE Parameter Description	38
Table 4-9 DLCE Port Description	39
Table 4-10 DLCE Parameter Description.....	39
Table 4-11 DLPE Port Description	41
Table 4-12 DLPE Parameter Description.....	41
Table 4-13 IDDR Port Description.....	43
Table 4-14 IDDRC Port Description	44

Table 4-15 IDES4 Port Description	46
Table 4-16 IDES8 Port Description	48
Table 4-17 IDES10 Port Description	51
Table 4-18 IVIDEO Port Description	54
Table 4-19 IDES16 Port Description	56
Table 4-20 IDDR_MEM Port Description	59
Table 4-21 IDES4_MEM Port Description.....	62
Table 4-22 IDES8_MEM Port Description.....	64
Table 4-23 IDES14 Port Description	67
Table 4-24 IDES32 Port Description	71
Table 4-25 OSIDES32 Port Description.....	76
Table 4-26 OSIDES32 Parameter Description.....	76
Table 4-27 ODDR Port Description	79
Table 4-28 ODDR Parameter Description.....	79
Table 4-29 ODDRC Port Description	82
Table 4-30 ODDRC Parameter Description	82
Table 4-31 OSER4 Port Description	84
Table 4-32 IDES4 Parameter Description.....	85
Table 4-33 OSER8 Port Description	88
Table 4-34 OSER8 Parameter Description	88
Table 4-35 OSER10 Port Description	91
Table 4-36 OVIDEO Port Description	94
Table 4-37 OSER16 Port Description	96
Table 4-38 ODDR_MEM Port Description	99
Table 4-39 ODDR_MEM Parameter Description	100
Table 4-40 OSER4_MEM Port Description	102
Table 4-41 OSER4_MEM Parameter Description.....	103
Table 4-42 OSER8_MEM Port Description.....	106
Table 4-43 OSER8_MEM Parameter Description.....	107
Table 4-44 IODELAY Port Description	111
Table 4-45 IODELAY Parameter Description	111

1 About This Guide

1.1 Purpose

Arora V 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 Arora V FPGA products.

Arora V GPIO architecture and Gowin Software usage are also provided to help you better understand 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:

- [DS981, GW5AT series of FPGA Products Data Sheet](#)
- [DS1103, GW5A series of FPGA Products Data Sheet](#)
- [DS1104, GW5AST series of FPGA Products Data Sheet](#)
- [SUG100, Gowin Software User Guide](#)

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
IOB	Input/Output Block
IO Buffer	Input/Output Buffer

Terminology and Abbreviations	Meaning
IO Logic	Input/Output Logic
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

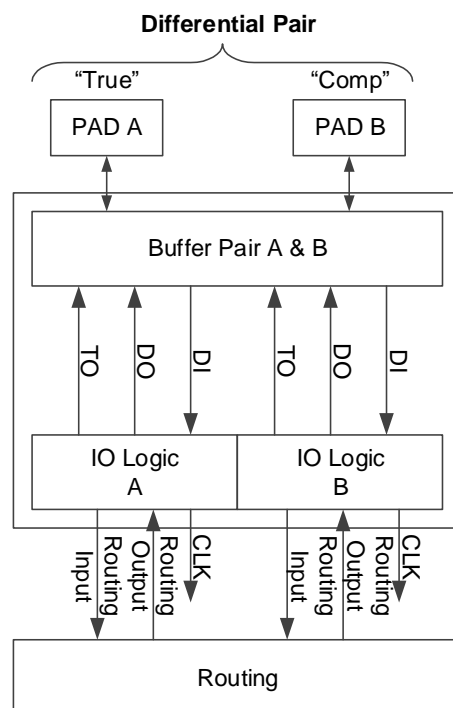
2 GPIO Overview

The GPIO in the GOWINSEMI Arora V FPGA products 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 Arora V series of 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 Arora V FPGA are:

- V_{CCIO} is supplied based on bank.
- Support LVCMOS, PCI, LVTTTL, SSTL, HSTL, LVDS, Mini_LVDS, RSDS, PPDS, BLVDS.
- Support MIPI level standard and MIPI I3C OpenDrain/PushPull conversion.
- Support hysteresis option for input signals
- Support drive strength option for output signals
- Support slew rate option for output signals
- Support individual bus keeper, pull-up/down resistor, and open drain output options
- Support hot socket
- I/O logic supports SDR mode and DDR mode, etc.

3 Input/Output Buffer

3.1 GPIO Level Standard

GOWINSEMI Arora V 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 Arora V 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 see [3.2 GPIO Banking](#).

For the pin voltage requirements for different level standards supported by GOWINSEMI Arora V FPGA products, please refer to the "I/O Level Standards" section in the data sheet.

3.2 GPIO Banking

The generic attributes of GPIO are:

- All banks support emulated LVDS differential output using external resistor network.
- 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 Arora V FPGA products can be powered and operated when V_{CC} and V_{CCIO} reach a certain threshold and power on reset signal (POR) is set. Before and during configuration, all GPIOs of the device are internally weak pull-up. After the configuration is complete, the I/O state is None, which can be configured via software. The state of CONFIG-related I/Os varies depending on the configuration mode. There are no power-on and power-off sequence requirements for core voltage and pin voltage for GOWINSEMI Arora V 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 I/O standards such as SSTL, HSTL, etc., each bank also provides an independent reference voltage (V_{REF}). Users can use the V_{REF} source (GW5AT-138 V_{REF} 0.6V/0.675V/0.75V/0.9V, GW5A-25 V_{REF} 0.6V/0.75V/0.9V/1.25V/1.5V) embedded in the IOB, and V_{CCIO} -based proportion voltage (33%, 42%, 50%, 58%), or an external V_{REF} , that is, use any I/O pin in the bank as an external V_{REF} input.

The GPIO in GOWINSEMI Arora V FPGA 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-keeper 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 Arora V 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 mode, the input hysteresis and bus-keeper will be disabled for the buffer.

All banks in Arora V devices supports on-chip programmable 100 Ohm input differential matched resistance.

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.

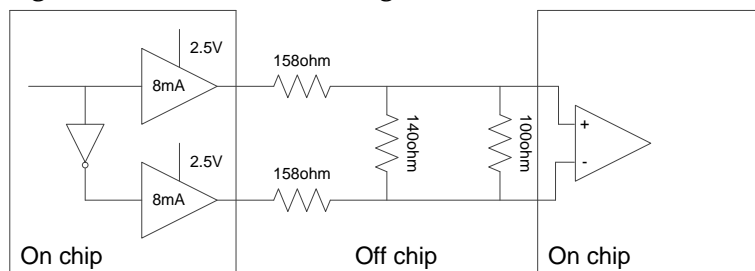
3.4 Emulated Differential Circuit Matching Network

3.4.1 Emulated LVDS

GOWINSEMI Arora V FPGA products can build compatible LVDS output standards via the complementary LVCMOS output and external

matching network. Figure 3-1 shows the external matching network.

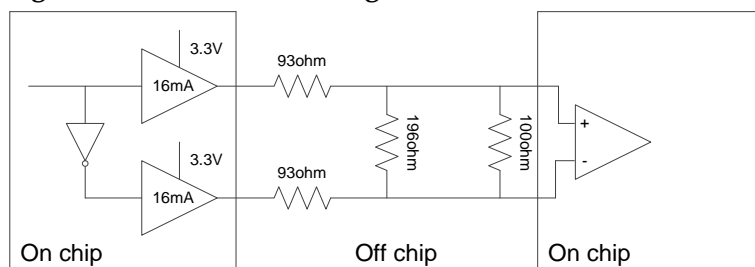
Figure 3-1 LVDS25E Matching Network



3.4.2 Emulated LVPECL

GOWINSEMI Arora V 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.

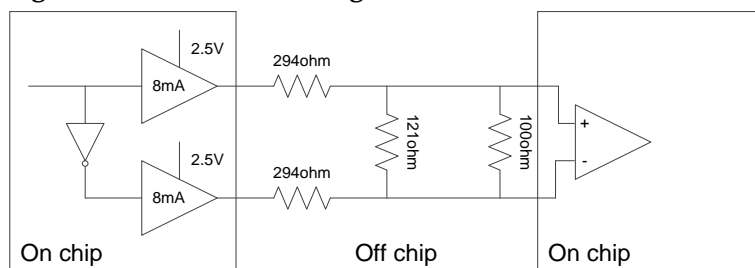
Figure 3-2 LVPECL Matching Network



3.4.3 Emulated RSDS

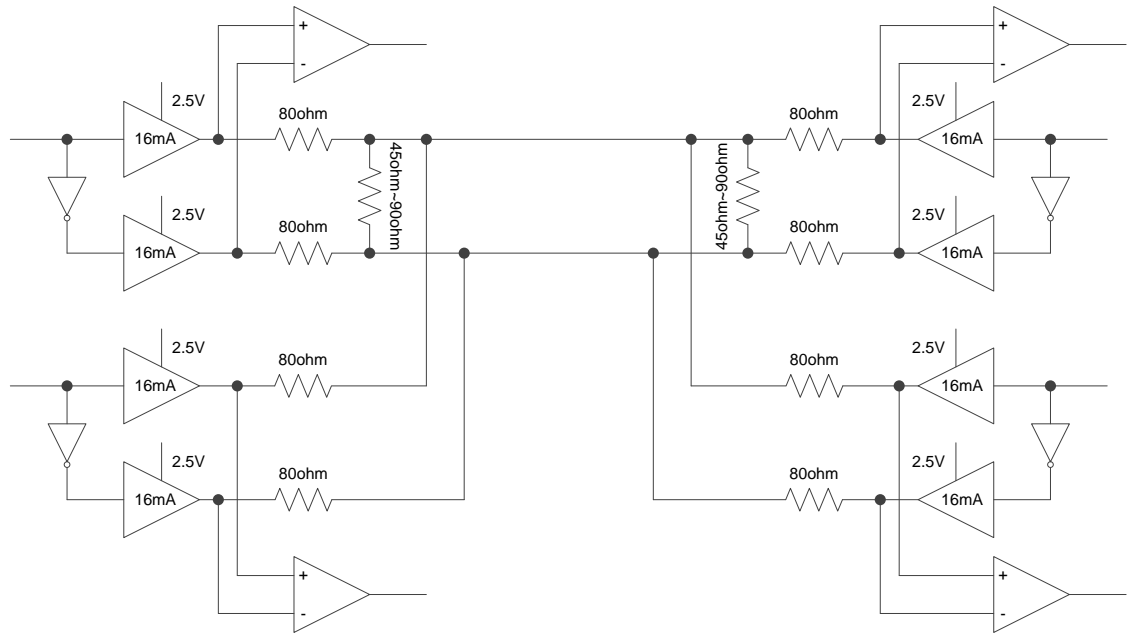
GOWINSEMI Arora V FPGA products can build compatible RSDS output standards via the complementary LVCMOS output and external matching network. Figure 3-3 shows the external matching network.

Figure 3-3 RSDSE Matching Network



3.4.4 Emulated BLVDS

GOWINSEMI Arora V 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=LVCMS18D;
```

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-keeper), and NONE (high impedance).

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

3.5.5 Reference Voltage

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 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 Resistor for Differential Signal

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

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

3.5.10 PCI Clamp

Support PCI Clamp diode ON and OFF. ON can limit the overcharge on the input and output pins.

```
IO_PORT "xxx" PCI_CLAMP=ON;
```

3.5.11 Pull-up/Pull-down Strength

Pull-up/pull-down strength setting provides different pull-up/pull-down strength on the pins, including MEDIUM, WEAK and STRONG options.

```
IO_PORT "xxx" PULL_STRENGTH=MEDIUM;
```

3.6 GPIO Primitive

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

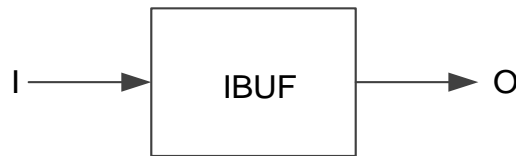
3.6.1 IBUF

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
  );
  
```

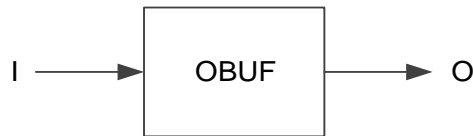
3.6.2 OBUF

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
  );
  
```

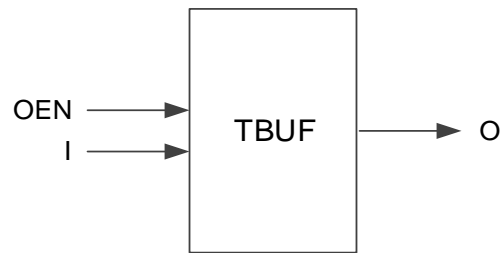
3.6.3 TBUF

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
  );
  
```

);

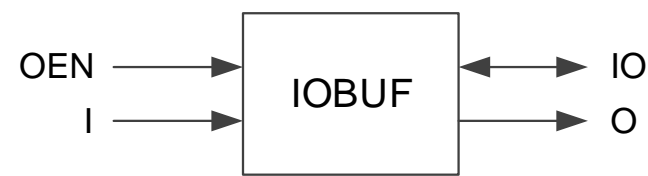
3.6.4 IOBUF

Primitive Introduction

Bidirectional buffer (IOBUF) is used as an input buffer when OEN is high and 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 includes TLVDS_IBUF.

True LVDS Input Buffer (TLVDS_IBUF).

Port Diagram

Figure 3-9 TLVDS_IBUF Port Diagram



Port Description

Table 3-5 TLVDS_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

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
    );

```

3.6.6 LVDS Output Buffer

Primitive Introduction

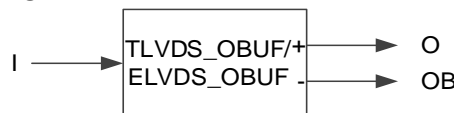
LVDS includes TLVDS_OBUF and ELVDS_OBUF.

True LVDS Output Buffer (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

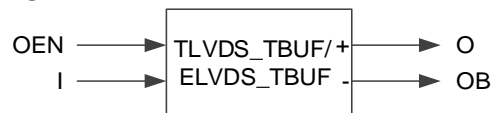
LVDS tristate buffer includes TLVDS_TBUF and ELVDS_TBUF.

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

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

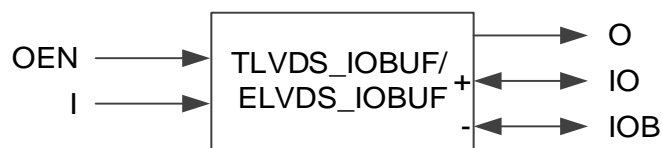
The LVDS inout buffer includes TLVDS_IOBUF and ELVDS_IOBUF.

True LVDS Bidirectional 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.

Emulated LVDS bi-directional Buffer (ELVDS_IOBUF) is used as emulated differential input buffer when OEN is high and used as emulated differential output buffer when OEN is low.

Port Diagram

Figure 3-12 TLVDS_IOBUF/ELVDS_IOBUF Port Diagram



Port Description

Table 3-8 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 input/output
IO	Inout	Differential A 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.

Functional Description

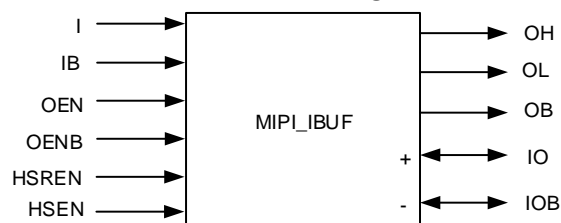
MIPI_IBUF supports LP and HS mode. IO and IOB are connected to pad.

LP mode supports bidirection; 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; HSREN controls the termination resistor. HSEH controls HS mode enable.

Port Diagram

Figure 3-13 MIPI_IBUF Port Diagram



Port Description

Table 3-9 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 resistor.
HSEN	Input	Enable HS mode
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	<ul style="list-style-type: none"> ● In LP mode, IO is output when OEN is low and input when OEN is high. ● In HS mode, IO is input.
IOB	Inout	<ul style="list-style-type: none"> ● In LP mode, IOB is output when OENB is low and IOB is input when OENB is high. ● 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),
    .HSEN(HSEN),
    .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;
        HSEN: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,
        HSEN=>HSEN,
        HSREN=>HSREN
    );

```

3.6.10 MIPI_OBUF_A

Primitive Introduction

MIPI Output Buffer with IL Signal (MIPI_OBUF_A) includes HS mode and LP modes.

When MODESEL is high, MIPI_OBUF_A is in the HS mode; when MODESEL is low, MIPI_OBUF_A is in the LP mode.

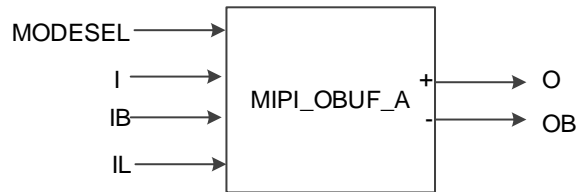
Device Supported

Table 3-10 MIPI_OBUF_A Device Supported

Product Family	Series	Device
Arora	GW5A	GW5A-25

Port Diagram

Figure 3-14 MIPI_OBUF_A Port Diagram



Port Description

Table 3-11 MIPI_OBUF_A Port Description

Port	I/O	Description
I	Input	In HS mode, I is the input signal of A.
IB	Input	In LP mode, IB is the input signal of B.
IL	Input	In LP mode, IL is the input signal of A.
MODESEL	Input	Mode selected, HS or LP.
O	Output	Data output signal of A, differential output of A in HS mode, single-ended output of A in LP mode.
OB	Output	Data output signal of B, differential output of B in HS mode, single-ended output of B 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
    );

```

3.6.11 I3C_IOBUF

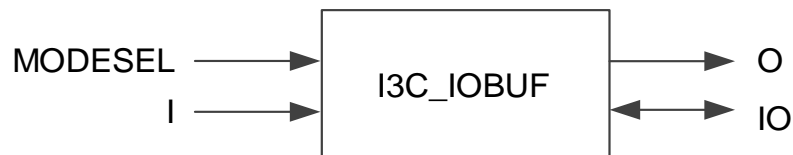
Primitive Introduction

I3C bidirectional Buffer (I3C_IOBUF) includes Normal mode and I3C mode.

I3C_IOBUF is used as a bidirectional buffer when MODESEL is high and used as a normal buffer when MODESEL is low.

Port Diagram

Figure 3-15 I3C_IOBUF Port Diagram



Port Description

Table 3-12 I3C_IOBUF Port Description

Ports	I/O	Description
I	Input	Data input signal
IO	Inout	Input and output signal, bidirectional.
MODESEL	Input	Mode selected 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
    );

```

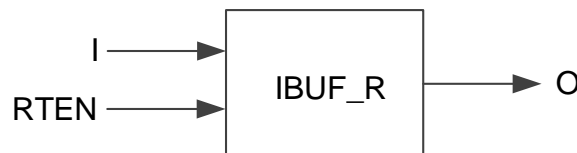
3.6.12 IBUF_R

Primitive Introduction

Input buffer with dynamic ODT (IBUF_R).

Port Diagram

Figure 3-16 IBUF_R Port Diagram



Port Description

Table 3-13 IBUF_R Port Description

Port	I/O	Description
I	Input	Data input signal
RTEN	Input	Dynamic enable ODT resistor
O	Output	Data output signal

Primitive Instantiation**Verilog Instantiation:**

```
IBUF_R uut(
    .O(O),
    .I(I),
    .RTEN(RTEN)
);
```

Vhdl Instantiation:

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

3.6.13 IOBUF_R**Primitive Introduction**

Bidirectional buffer with dynamic ODT (IOBUF_R).

Port Diagram

Figure 3-17 IOBUF_R Port Diagram

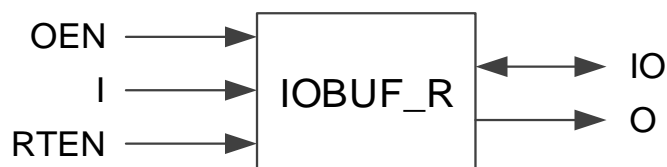
**Port Description**

Table 3-14 IOBUF_R Port Description

Port	I/O	Description
I	Input	Data input signal

Port	I/O	Description
OEN	Input	Output tristate enable signal
RTEN	Input	Dynamic enable ODT resistor
O	Output	Data output signal
IO	Inout	Input/output signal, bidirectional.

Primitive Instantiation

Verilog Instantiation:

```
IOBUF_R uut(
    .O(O),
    .IO(IO),
    .I(I),
    .OEN(OEN),
    .RTEN(RTEN)
);
```

Vhdl Instantiation:

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

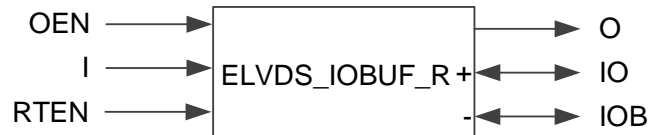
3.6.14 ELVDS_IOBUF_R

Primitive Introduction

Emulated LVDS bidirectional buffer with dynamic ODT (ELVDS_IOBUF_R).

Port Diagram

Figure 3-18 ELVDS_IOBUF_R Port Diagram



Port Description

Table 3-15 ELVDS_IOBUF_R Port Description

Port	I/O	Description
I	Input	Data input signal
OEN	Input	Output tristate enable signal
RTEN	Input	Dynamic enable ODT resistor
IO	Inout	A differential input/output, bidirectional.
IOB	Inout	B differential input/output, bidirectional.
O	Output	Data output signal

Primitive Instantiation

Verilog Instantiation:

```

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

Vhdl Instantiation:

```

COMPONENT ELVDS_IOBUF_R
PORT (
    O:OUT std_logic;
    IO:INOUT std_logic;
    IOB:INOUT std_logic;
  
```

```

        I:IN std_logic;
        OEN:IN std_logic;
        RTEN:IN std_logic;
    );
END COMPONENT;
uut:ELVDS_IOBUF_R
    PORT MAP(
        O=>O,
        IO=>IO,
        IOB=>IOB,
        I=> I,
        OEN=>OEN,
        RTEN=>RTEN
    );

```

3.6.15 TLVDS_IBUF_ADC

Primitive Introduction

True LVDS input buffer is used in conjunction with the ADC module to implement the ADC dynamic voltage source selection (TLVDS_IBUF_ADC).

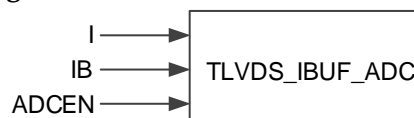
Device Supported

Table 3-16 TLVDS_IBUF_ADC Device Supported

Product Family	Series	Device
Arora	GW5AT	GW5AT-138
	GW5AST	GW5AST-138B
	GW5A	GW5A-25

Port Diagram

Figure 3-19 TLVDS_IBUF_ADC Port Diagram



Port Description

Table 3-17 TLVDS_IBUF_ADC Port Description

Port	I/O	Description
I	Input	Differential input signal of A
IB	Input	Differential input signal of B
ADCEN	Input	ADC dynamic enable signal

Primitive Instantiation

Verilog Instantiation:

```
TLVDS_IBUF_ADC uut(
    .ADCEN(ADCEN),
    .IB(IB),
    .I(I)
);
```

Vhdl Instantiation:

```
COMPONENT TLVDS_IBUF_ADC
PORT (
    ADCEN:IN std_logic;
    IB:IN std_logic;
    I:IN std_logic
);
END COMPONENT;
uut: TLVDS_IBUF_ADC
PORT MAP(
    ADCEN=>ADCEN,
    IB=>IB,
    I=>I
);
```


4 Input/Output Logic

I/O logic in GOWINSEMI Arora V 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, bidirectional signal and tristate output signal (output signal with tristate control).

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

Figure 4-1 I/O Logic View-Output

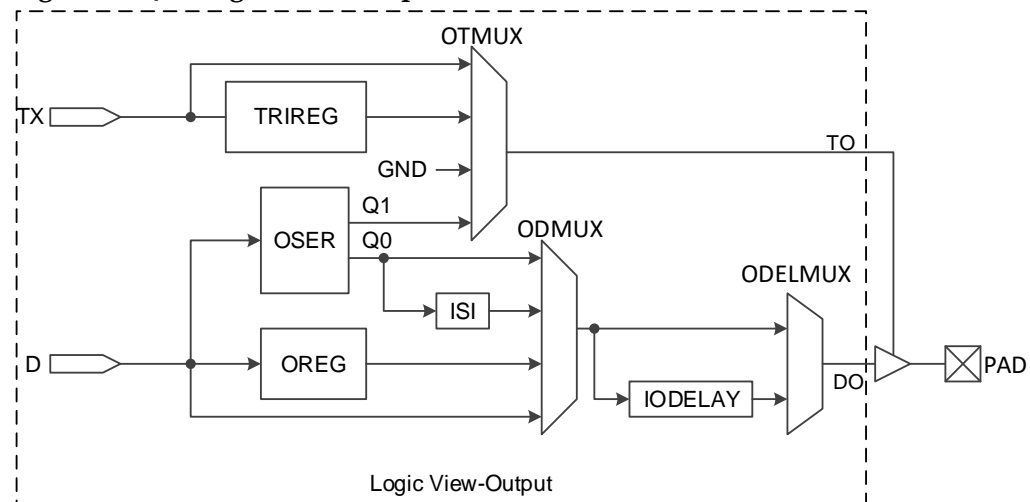
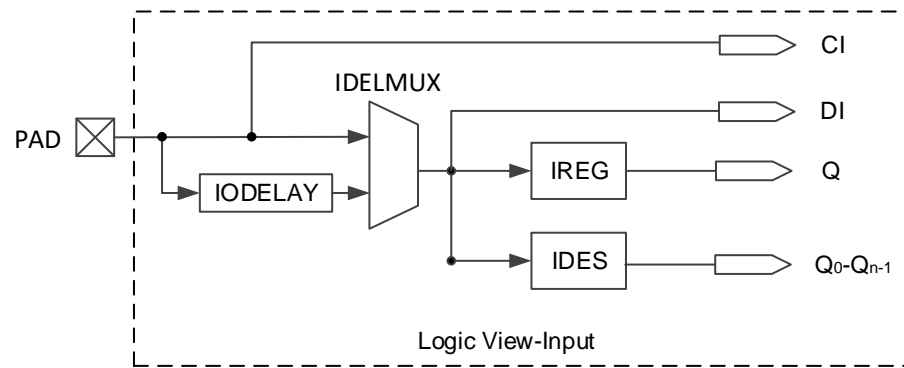


Figure 4-2 shows the input of the I/O logic in GOWINSEMI Arora V FPGA products.

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 lologic when the input D of the FF/LATCH is driven by a Buffer/IODELAY that does not drive other lologics, or when the output Q of the FF/LATCH only drives a Buffer/IODELAY and the Buffer is not a MIPI Buffer.

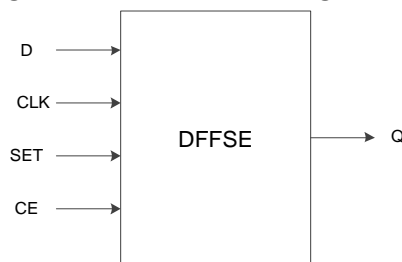
4.1.1 DFFSE

Primitive Introduction

D Flip-Flop with Clock Enable and Synchronous Set (DFFSE), triggered by the rising edge, is a D flip-flop with clock enable and synchronous set.

Port Diagram

Figure 4-3 DFFSE Port Diagram



Port Description

Table 4-1 DFFSE Port Description

Port	I/O	Description
D	Input	Data input signal
CLK	Input	Clock Input signal

Port	I/O	Description
SET	Input	Synchronous set signal, active-high.
CE	Input	Clock enable signal
Q	Output	Data output signal

Parameter Description

Table 4-2 DFFSE Parameter Description

Name	Value	Default	Description
INIT	1'b1	1'b1	DFFSE initial value

Primitive Instantiation

Verilog Instantiation:

```

DFFSE instName (
    .D(D),
    .CLK(CLK),
    .SET(SET),
    .CE(CE),
    .Q(Q)
);
defparam instName.INIT=1'b1;

```

Vhdl Instantiation:

```

COMPONENT DFFSE
    GENERIC (INIT:bit:= '1');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic;
        SET:IN std_logic;
        CE:IN std_logic
    );
END COMPONENT;

uut:DFFSE
    GENERIC MAP(INIT=>'1')
    PORT MAP (
        Q=>Q,

```

```

D=>D,
CLK=>CLK,
SET=>SET,
CE=>CE
);

```

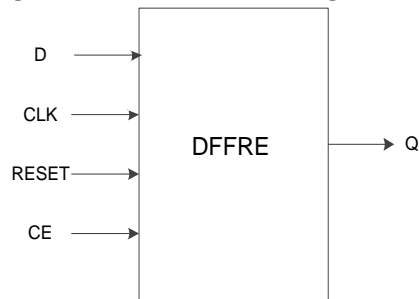
4.1.2 DFFRE

Primitive Introduction

D Flip-Flop with Clock Enable and Synchronous Reset (DFFRE), triggered by the rising edge, is a D flip-flop with clock enable and synchronous reset.

Port Diagram

Figure 4-4 DFFRE Port Diagram



Port Description

Table 4-3 DFFRE Port Description

Port	I/O	Description
D	Input	Data input signal
CLK	Input	Clock Input signal
RESET	Input	Synchronous reset signal, active-high.
CE	Input	Clock enable signal
Q	Output	Data output signal

Parameter Description

Table 4-4 DFFRE Parameter Description

Name	Value	Default	Description
INIT	1'b0	1'b0	DFFRE initial value

Primitive Instantiation

Verilog Instantiation:

```
DFFRE instName (
```

```

        .D(D),
        .CLK(CLK),
        .RESET(RESET),
        .CE(CE),
        .Q(Q)
    );
    defparam instName.INIT=1'b0;

```

Vhdl Instantiation:

```

COMPONENT DFFRE
    GENERIC (INIT:bit:= '0');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic;
        RESET:IN std_logic;
        CE:IN std_logic
    );
END COMPONENT;
uut:DFFRE
    GENERIC MAP(INIT=>'0')
    PORT MAP (
        Q=>Q,
        D=>D,
        CLK=>CLK,
        RESET=>RESET,
        CE=>CE
    );

```

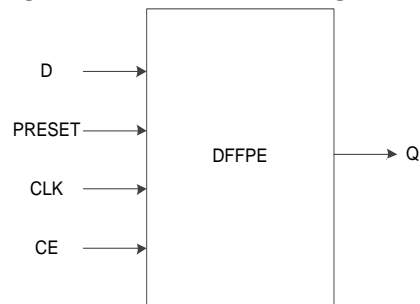
4.1.3 DFFPE

Primitive Introduction

D Flip-Flop with Clock Enable and Asynchronous Preset (DFFPE), triggered by the rising edge, is a D flip-flop with clock enable and asynchronous preset.

Port Diagram

Figure 4-5 DFFPE Port Diagram



Port Description

Table 4-5 DFFPE Port Description

Port	I/O	Description
D	Input	Data input signal
CLK	Input	Clock Input signal
RESET	Input	Asynchronous preset signal, active-high.
CE	Input	Clock enable signal
Q	Output	Data output signal

Parameter Description

Table 4-6 DFFPE Parameter Description

Name	Value	Default	Description
INIT	1'b1	1'b1	DFFPE initial value

Primitive Instantiation

Verilog Instantiation:

```

DFFPE instName (
    .D(D),
    .CLK(CLK),
    .PRESET(PRESET),
    .CE(CE),
    .Q(Q)
);
defparam instName.INIT=1'b1;
  
```

Vhdl Instantiation:

```

COMPONENT DFFPE
    GENERIC (INIT:bit:= '1');
  
```

```

        PORT(
            Q:OUT std_logic;
            D:IN std_logic;
            CLK:IN std_logic;
            PRESET:IN std_logic;
            CE:IN std_logic
        );
    END COMPONENT;
    uut:DFFPE
        GENERIC MAP(INIT=>'1')
        PORT MAP (
            Q=>Q,
            D=>D,
            CLK=>CLK,
            PRESET=>PRESET,
            CE=>CE
        );

```

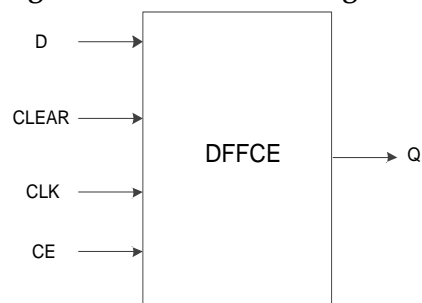
4.1.4 DFFCE

Primitive Introduction

D Flip-Flop with Clock Enable and Asynchronous Clear (DFFCE), triggered by the rising edge, is a D flip-flop with clock enable and asynchronous clear.

Port Diagram

Figure 4-6 DFFCE Port Diagram



Port Description

Table 4-7 DFFCE Port Description

Port	I/O	Description
D	Input	Data input signal
CLK	Input	Clock Input signal

Port	I/O	Description
CLEAR	Input	Asynchronous clear signal, active-high.
CE	Input	Clock enable signal
Q	Output	Data output signal

Parameter Description

Table 4-8 DFFCE Parameter Description

Name	Value	Default	Description
INIT	1'b0	1'b0	DFFCE initial value

Primitive Instantiation

Verilog Instantiation:

```

DFFCE instName (
    .D(D),
    .CLK(CLK),
    .CLEAR(CLEAR),
    .CE(CE),
    .Q(Q)
);
defparam instName.INIT=1'b0;

```

Vhdl Instantiation:

```

COMPONENT DFFCE
    GENERIC (INIT:bit:= '0');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic;
        CLEAR:IN std_logic;
        CE:IN std_logic
    );
END COMPONENT;

uut:DFFCE
    GENERIC MAP(INIT=>'0')
    PORT MAP (
        Q=>Q,

```



```

D=>D,
CLK=>CLK,
CLEAR=>CLEAR,
CE=>CE
);

```

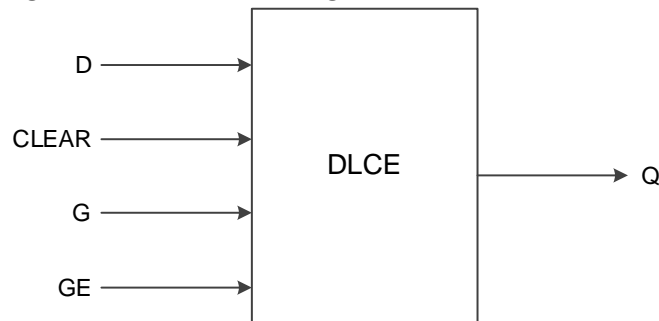
4.1.5 DLCE

Primitive Introduction

Data Latch with Asynchronous Clear and Latch Enable (DLCE) is a data latch with asynchronous clear and latch enable, and the control signal G is active high.

Port Diagram

Figure 4-7 DLCE Port Diagram



Port Description

Table 4-9 DLCE Port Description

Port	I/O	Description
D	Input	Data input signal
CLEAR	Input	Asynchronous clear signal, active-high.
G	Input	Data control signal, active-high.
GE	Input	Data control enable signal
Q	Output	Data output signal

Parameter Description

Table 4-10 DLCE Parameter Description

Name	Value	Default	Description
INIT	1'b0	1'b0	DLCE initial value

Primitive Instantiation

Verilog Instantiation:

```
DLCE instName (
```

```

        .D(D),
        .CLEAR(CLEAR),
        .G(G),
        .GE(GE),
        .Q(Q)
    );
    defparam instName.INIT=1'b0;

```

Vhdl Instantiation:

```

COMPONENT DLCE
    GENERIC (INIT:bit:= '0');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        G:IN std_logic;
        GE:IN std_logic;
        CLEAR:IN std_logic
    );
END COMPONENT;
uut:DLCE
    GENERIC MAP(INIT=>'0')
    PORT MAP (
        Q=>Q,
        D=>D,
        G=>G,
        GE=>GE,
        CLEAR=>CLEAR
    );

```

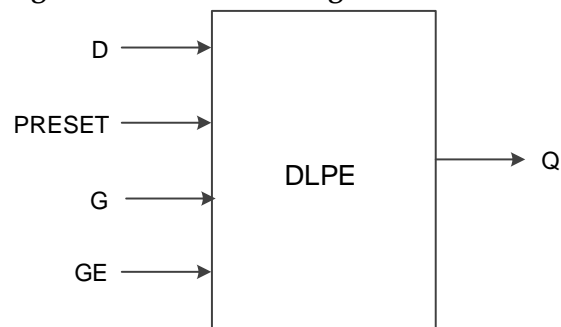
4.1.6 DLPE

Primitive Introduction

Data Latch with Asynchronous Preset and Latch Enable (DLPE) is a data latch with asynchronous preset and latch enable, and the control signal G is active high.

Port Diagram

Figure 4-8 DLPE Port Diagram



Port Description

Table 4-11 DLPE Port Description

Port	I/O	Description
D	Input	Data input signal
PRESET	Input	Asynchronous preset signal, active-high.
G	Input	Data control signal, active-high.
GE	Input	Data control enable signal
Q	Output	Data output signal

Parameter Description

Table 4-12 DLPE Parameter Description

Name	Value	Default	Description
INIT	1'b1	1'b1	DLPE initial value

Primitive Instantiation

Verilog Instantiation:

```

DLPE instName (
    .D(D),
    .PRESET(PRESET),
    .G(G),
    .GE(GE),
    .Q(Q)
);
defparam instName.INIT=1'b1;
  
```

Vhdl Instantiation:

```

COMPONENT DLPE
  
```

```

    GENERIC (INIT:bit:= '1');
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        G:IN std_logic;
        GE:IN std_logic;
        PRESET:IN std_logic

    );
END COMPONENT;
uut:DLPE
    GENERIC MAP(INIT=>'1')
    PORT MAP (
        Q=>Q,
        D=>D,
        G=>G,
        GE=>GE,
        PRESET =>PRESET
    );

```

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 4-9 and its timing diagram is as shown in Figure 4-10.

Figure 4-9 IDDR Logic Diagram

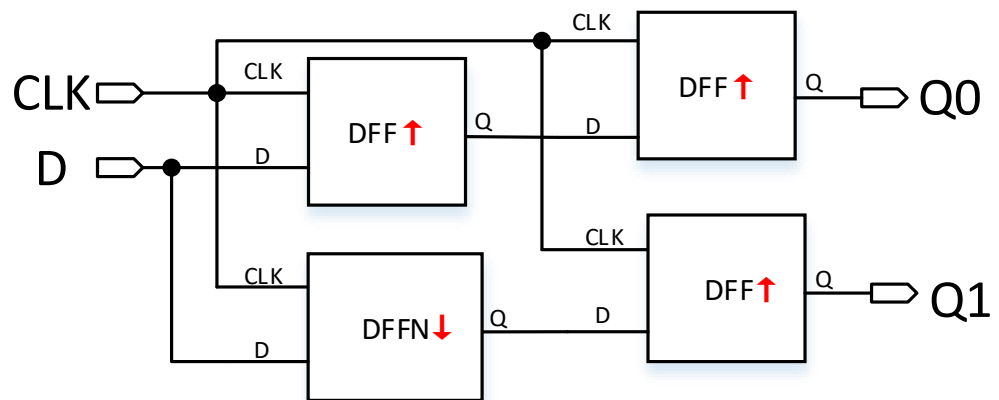
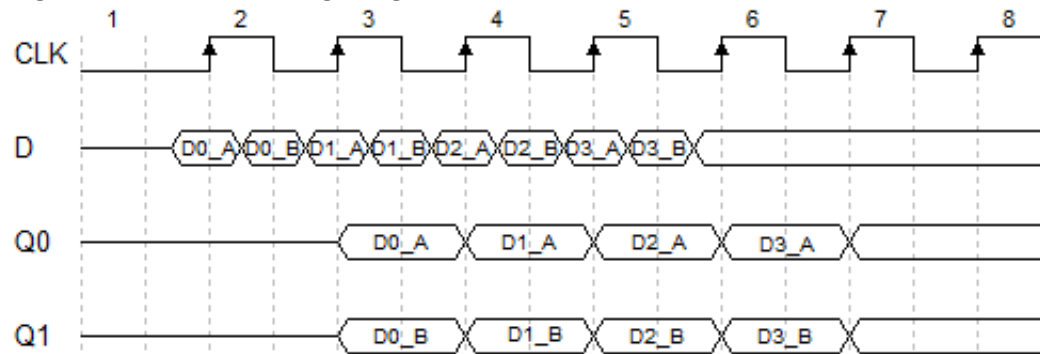


Figure 4-10 IDDR Timing Diagram



Port Diagram

Figure 4-11 IDDR Port Diagram



Port Description

Table 4-13 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

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.

Verilog Instantiation:

```
IDDR uut(
    .Q0(Q0),
    .Q1(Q1),
    .D(D),
    .CLK(CLK)
);
```

Vhdl Instantiation:

```

COMPONENT IDDR
    PORT(
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        D:IN std_logic;
        CLK:IN std_logic
    );
END COMPONENT;
uut:IDDR
    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 4-12 IDDRC Port Diagram****Port Description****Table 4-14 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

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.

Verilog Instantiation:

```
IDDRC uut(
    .Q0(Q0),
    .Q1(Q1),
    .D(D),
    .CLK(CLK),
    .CLEAR(CLEAR)
);
```

Vhdl Instantiation:

```
COMPONENT IDDRC
    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
    PORT MAP (
        Q0=>Q0,
        Q1=>Q1,
        D=>D,
        CLEAR=>CLEAR,
        CLK=>CLK
    );
```

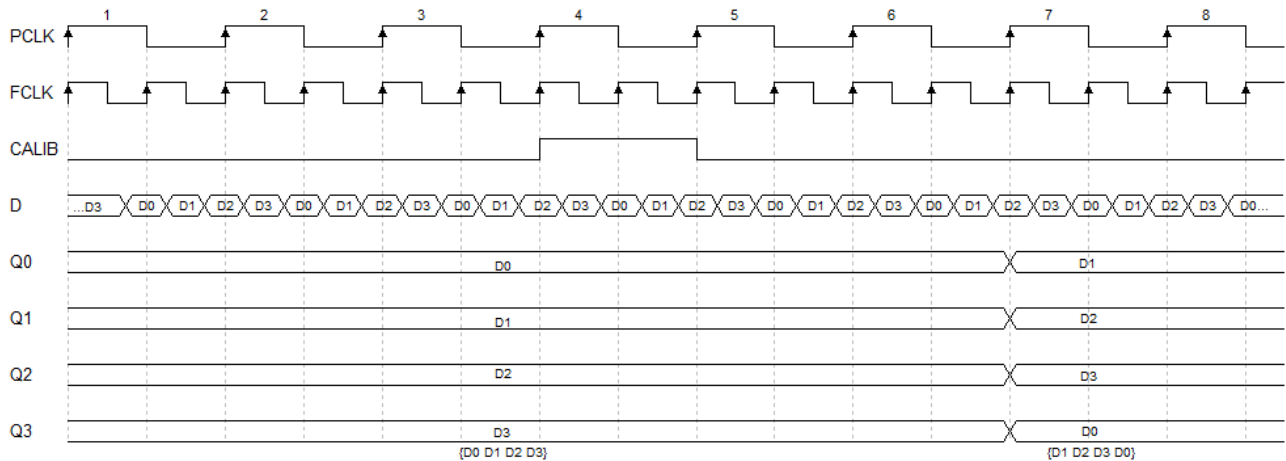
4.2.3 IDES4**Primitive Introduction**

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 4-13.

Figure 4-13 CALIB Timing Diagram



Note!

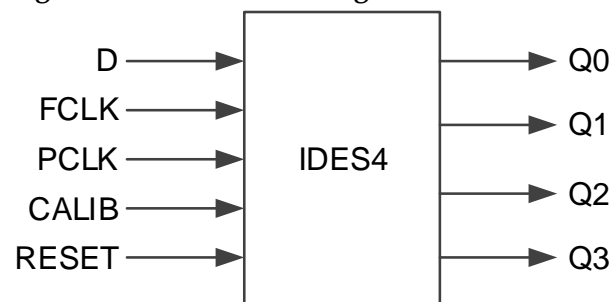
The pulse width and timing of the CALIB signal in the example are for reference only and can be adjusted as needed, the pulse width is equal to or greater than T_{PCLK} .

PCLK is usually obtained by FCLK frequency division:

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

Port Diagram

Figure 4-14 IDES4 Port Diagram



Port Description

Table 4-15 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.

Port	I/O	Description
RESET	Input	Asynchronous reset input signal, active-high.
Q3=Q0	Output	IDES4 data output signal

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.

Verilog Instantiation:

```
IDES4 uut(
    .Q0(Q0),
    .Q1(Q1),
    .Q2(Q2),
    .Q3(Q3),
    .D(D),
    .FCLK(FCLK),
    .PCLK(PCLK),
    .CALIB(CALIB),
    .RESET(RESET)
);
```

Vhdl Instantiation:

```
COMPONENT IDES4
    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
  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

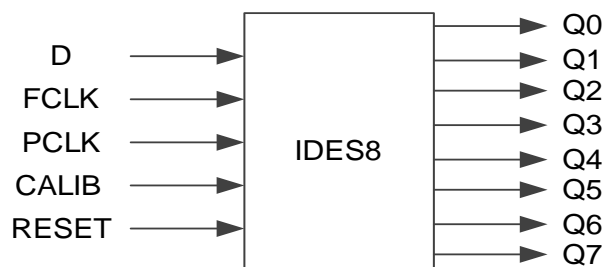
IDES8 mode realizes 1:8 serial parallel conversion and output data is provided to FPGA logic at the same clock edge. CALIB is supported to adjust the sequence of output data. The data is shifted by one bit per pulse. After four shifts, the data output will be the same as the data before the shift.

PCLK is usually obtained by FCLK frequency division:

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

Port Diagram

Figure 4-15 IDES8 Port Diagram



Port Description

Table 4-16 IDES8 Port Description

Port	I/O	Description
D	Input	IDES8 data input signal

Port	I/O	Description
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

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.

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)
);
```

Vhdl Instantiation:

```
COMPONENT IDES8
    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
    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

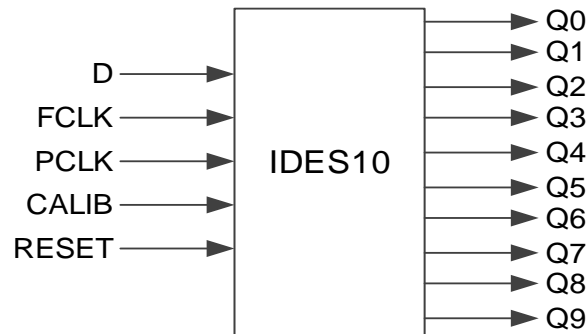
the shift.

PCLK is usually obtained by FCLK frequency division:

$$f_{PCLK} = 1/5 f_{FCLK}$$

Port Diagram

Figure 4-16 IDES10 Port Diagram



Port Description

Table 4-17 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

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.

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)
    );

```

Vhdl Instantiation:

```

COMPONENT IDES10
    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
    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

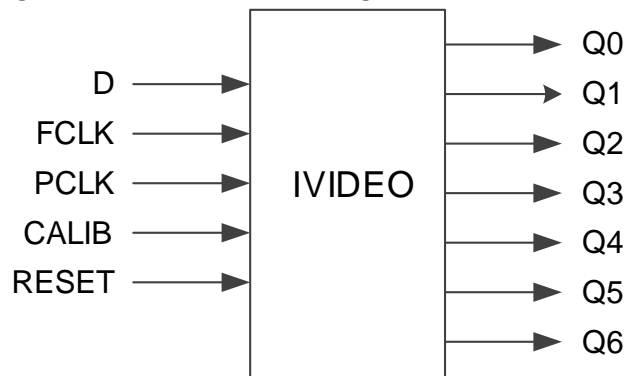
IVIDEO mode realizes 1:7 serial parallel conversion and output data is provided to FPGA logic at the same clock edge. CALIB is supported to adjust the sequence of output data. The data is shifted by two bits per pulse. After seven shifts, the data output will be the same as the data before the shift.

PCLK is usually obtained by FCLK frequency division:

$$f_{PCLK} = 1/3.5 f_{FCLK}$$

Port Diagram

Figure 4-17 IVIDEO Port Diagram



Port Description

Table 4-18 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

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.

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)
);
```

Vhdl Instantiation:

```
COMPONENT IVIDEO
    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
    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.

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

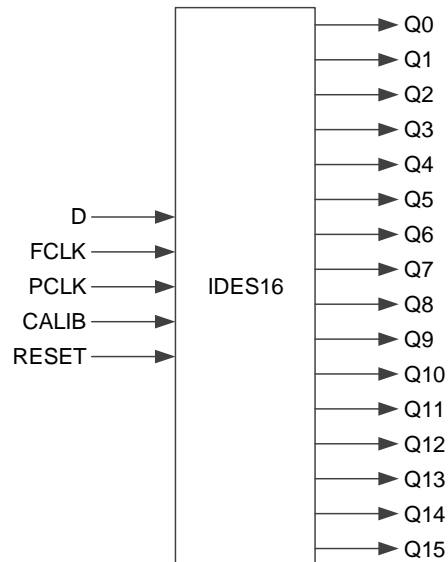
shift.

PCLK is usually obtained by FCLK frequency division:

$$f_{PCLK} = 1/8 f_{FCLK}$$

Port Diagram

Figure 4-18 IDES16 Port Diagram



Port Description

Table 4-19 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

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.

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)  
);
```

Vhdl Instantiation:

```
COMPONENT IDES16
```

```
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
```

```
    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.

Functional Description

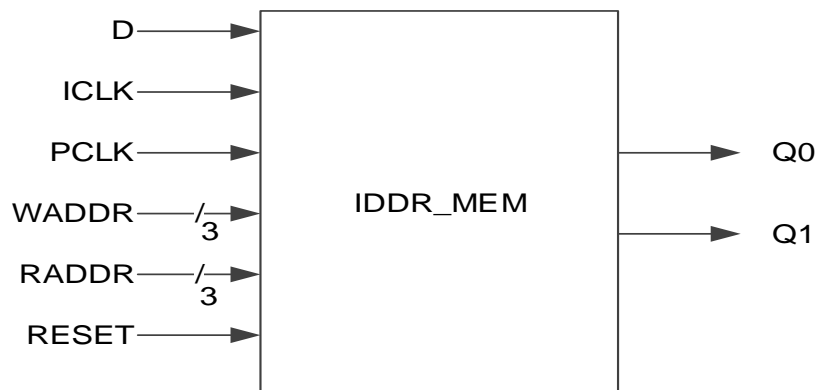
IDDR_MEM output data is provided to FPGA logic at the same clock edge. IDDR_MEM needs to be used with DQS. ICLK connects the DQSR90 of DQS output signals and sends data to IDDR_MEM according to the ICLK clock edge. WADDR [2: 0] connects the WPOINT output signal of DQS; RADDR [2: 0] connects the RPOINT 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 4-19 IDDR_MEM Port Diagram



Port Description

Table 4-20 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
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

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)
);
```

Vhdl Instantiation:

```
COMPONENT IDDR_MEM
    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
    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.

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.

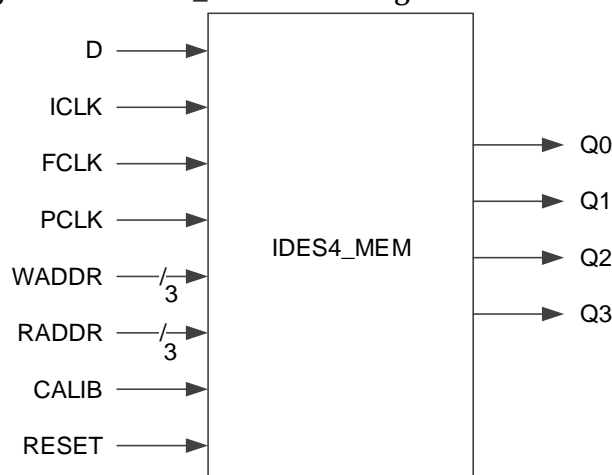
The frequency relation between PCLK, FCLK and ICLK is

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

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

Port Diagram

Figure 4-20 IDES4_MEM Port Diagram



Port Description

Table 4-21 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

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)
);

```

Vhdl Instantiation:


```
COMPONENT IDES4_MEM
  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
  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.

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.

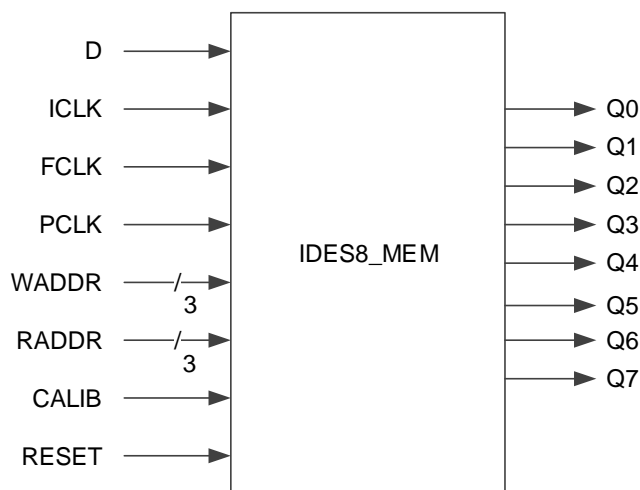
The frequency relation between PCLK, FCLK and ICLK is

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

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

Port Diagram

Figure 4-21 IDES8_MEM Port Diagram



Port Description

Table 4-22 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

Port	I/O	Description
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

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)
);

```

Vhdl Instantiation:

```

COMPONENT IDES8_MEM
    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
    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
    );

```

);

4.2.11 IDES14

Primitive Introduction

The 1 to 14 Deserializer (IDES14) is a deserializer of 1 bit serial input and 14 bits parallel output.

Functional Description

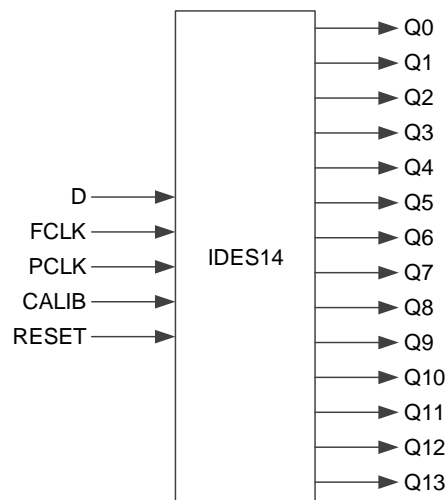
IDES14 mode realizes 1:14 serial parallel conversion and the output data is provided to FPGA logic at the same clock edge. CALIB is supported to adjust the sequence of output data. The data is shifted by one bit per pulse. After fourteen shifts, the data output will be the same as the data before the shift.

PCLK is usually obtained by FCLK frequency division:

$$f_{PCLK} = 1/7 f_{FCLK}$$

Port Diagram

Figure 4-22 IDES14 Port Diagram



Port Description

Table 4-23 IDES14 Port Description

Port	I/O	Description
D	Input	IDES14 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.
Q13~Q0	Output	IDES14 data output signal

Connection Rule

Data input D of IDES14 can be directly from IBUF or from the output DO of IODELAY module.

Primitive Instantiation

The primitive can be instantiated directly.

Verilog Instantiation:

```
IDES14 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),
    .D(D),
    .FCLK(FCLK),
    .PCLK(PCLK),
    .CALIB(CALIB),
    .RESET(RESET)
);
```

Vhdl Instantiation:

```
COMPONENT IDES14
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;
        D:IN std_logic;
        FCLK:IN std_logic;
        PCLK:IN std_logic;
        CALIB:IN std_logic;
        RESET:IN std_logic
    );
END COMPONENT;
uut:IDES14
    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,
        D=>D,
        FCLK=>FCLK,
        PCLK=>PCLK,
```

```

    CALIB=>CALIB,
    RESET=>RESET
);

```

4.2.12 IDES32

Primitive Introduction

The 1 to 32 Deserializer (IDES32) is a deserializer of 1 bit serial input and 32 bits parallel output.

Functional Description

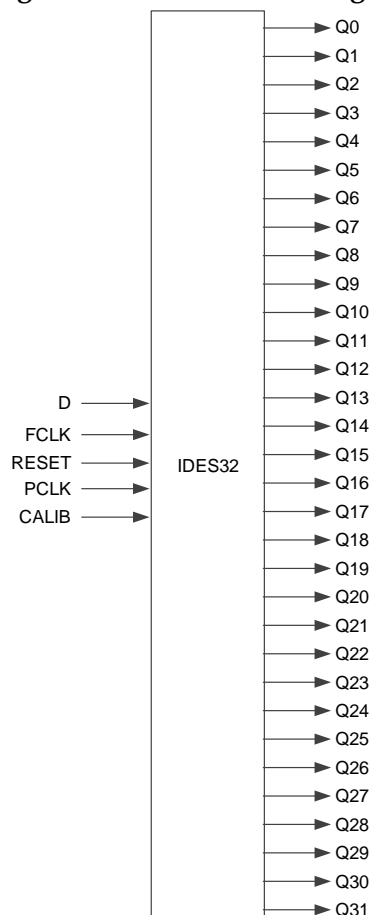
IDES32 mode realizes 1:32 serial parallel conversion and the output data is provided to FPGA logic at the same clock edge. CALIB is supported to adjust the sequence of output data. The data is shifted by one bit per pulse. After thirty-two shifts, the data output will be the same as the data before the shift.

PCLK is usually obtained by FCLK frequency division:

$$f_{PCLK} = 1/16 f_{FCLK}$$

Port Diagram

Figure 4-23 IDES32 Port Diagram



Port Description

Table 4-24 IDES32 Port Description

Port	I/O	Description
D	Input	IDES32 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.
Q31~Q0	Output	IDES32 data output signal

Connection Rule

Data input D of IDES32 can be directly from IBUF or from the output DO of IODELAY module.

Primitive Instantiation

The primitive can be instantiated directly.

Verilog Instantiation:

```

IDES32 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),
    .Q16(Q16),
    .Q17(Q17),

```

```

.Q18(Q18),
.Q19(Q19),
.Q20(Q20),
.Q21(Q21),
.Q22(Q22),
.Q23(Q23),
.Q24(Q24),
.Q25(Q25),
.Q26(Q26),
.Q27(Q27),
.Q28(Q28),
.Q29(Q29),
.Q30(Q30),
.Q31(Q31),
.D(D),
.FCLK(FCLK),
.PCLK(PCLK),
.CALIB(CALIB),
.RESET(RESET)
);

```

Vhdl Instantiation:

```
COMPONENT IDES32
```

```
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;
Q16:OUT std_logic;
Q17:OUT std_logic;
Q18:OUT std_logic;
Q19:OUT std_logic;
Q20:OUT std_logic;
Q21:OUT std_logic;
Q22:OUT std_logic;
Q23:OUT std_logic;
Q24:OUT std_logic;
Q25:OUT std_logic;
Q26:OUT std_logic;
Q27:OUT std_logic;
Q28:OUT std_logic;
Q29:OUT std_logic;
Q30:OUT std_logic;
Q31: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:IDES32
  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,  
Q16=>Q16,  
Q17=>Q17,  
Q18=>Q18,  
Q19=>Q19,  
Q20=>Q20,  
Q21=>Q21,  
Q22=>Q22,  
Q23=>Q23,  
Q24=>Q24,  
Q25=>Q25,  
Q26=>Q26,  
Q27=>Q27,  
Q28=>Q28,  
Q29=>Q29,  
Q30=>Q30,  
Q31=>Q31,  
D=>D,  
FCLK=>FCLK,  
PCLK=>PCLK,  
CALIB=>CALIB,  
RESET=>RESET  
);
```

4.2.13 OSIDES32

Primitive Introduction

OSIDES32 is a deserializer of 1 bit serial input and 32 bits parallel output to implement oversampling serial-to-parallel function.

Functional Description

Using two IOLs can achieve 1:32 oversampling serial to parallel, and asynchronous reset is supported; but the function of adjusting the output data sequence by CALIB is not supported.

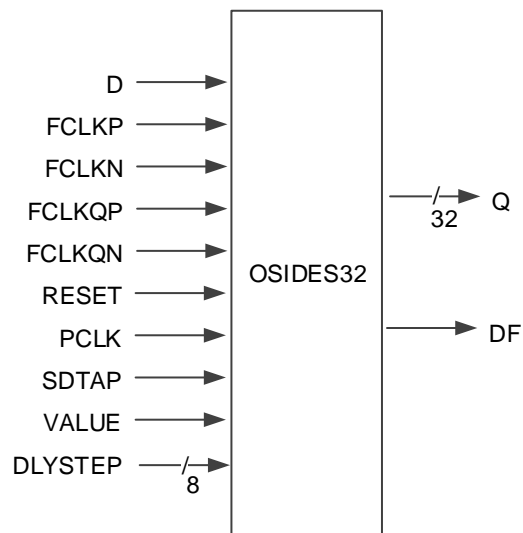
The major difference between oversampling and normal serial-to-parallel mode is that oversampling is performed 4 times in one fclk, while the normal serial-to-parallel is performed only twice in one fclk, that is, rising and falling edge samplings.

The frequency relationship between PCLK and FCLKP, FCLKN, FCLKQP, FCLKQN is

$$f_{PCLK} = 1/4f_{FCLKP} = 1/4f_{FCLKN} = 1/4f_{FCLKQP} = 1/4f_{FCLKQN}$$

Port Diagram

Figure 4-24 OSIDES32 Port Diagram



Port Description

Table 4-25 OSIDES32 Port Description

Port	I/O	Description
D	Input	OSIDES32 data input signal
FCLKP	Input	High speed clock input signal
FCLKN	Input	High speed clock input signal
FCLKQP	Input	High speed clock input signal
FCLKQN	Input	High speed clock input signal
PCLK	Input	Primary clock input signal
RESET	Input	Asynchronous reset input signal, active-high.
SDTAP	Input	Control loading static delay step 0: Load static delay 1: Dynamically adjust delay
VALUE	Input	Dynamically adjust delay value at VALUE falling edge; each pulse moves one delay step.
DLYSTEP[7:0]	Input	Dynamic delay value
Q[31:0]	Output	OSIDES32 data output signal
DF	Output	IODELAY output flag bit

Parameter Description

Table 4-26 OSIDES32 Parameter Description

Name	Value	Default	Description
C_STATIC_DLY	0~255	0	Static delay step control
DYN_DLY_EN	"FALSE"/"TRUE"	"FALSE"	Dynamic mode enable control
ADAPT_EN	"FALSE"/"TRUE"	"FALSE"	Adaptive mode enable control

Connection Rule

Data input D of OSIDES32 can be directly from IBUF.

Primitive Instantiation

The primitive can be instantiated directly.

Verilog Instantiation:

```
OSIDES32 uut(
    .Q(Q),
    .D(D),
    .FCLKP(FCLKP),
    .FCLKN(FCLKN),
    .FCLKQP(FCLKQP),
    .FCLKQN(FCLKQN),
```

```

        .PCLK(PCLK),
        .SDTAP(SDTAP),
        .VALUE(VALUE),
        .RESET(RESET),
        .DLYSTEP(DLYSTEP),
        .DF(DF)
    );
    defparam uut.C_STATIC_DLY=0;
    defparam uut.DYN_DLY_EN="FALSE";
    defparam uut.ADAPT_EN="FALSE";

```

Vhdl Instantiation:

```

    COMPONENT OSIDES32(
        C_STATIC_DLY:integer:=0;
        DYN_DLY_EN:string="FALSE";
        ADAPT_EN:string="FALSE"
    );
    PORT(
        Q:OUT std_logic;
        D:IN std_logic;
        FCLKP:IN std_logic;
        FCLKN:IN std_logic;
        FCLKQP:IN std_logic;
        FCLKQN:IN std_logic;
        PCLK:IN std_logic;
        SDTAP:IN std_logic;
        VALUE:IN std_logic;
        RESET:IN std_logic;
        DLYSTEP: IN std_logic;
        DF:OUT std_logic
    );
END COMPONENT;
uut:OSIDES32
    GENERIC MAP (C_STATIC_DLY=>0,
        DYN_DLY_EN=>"FALSE",
        ADAPT_EN=>"FALSE"

```

```

)
PORT MAP (
    Q=>Q,
    D=>D,
    FCLKP=>FCLKP,
    FCLKN=>FCLKN,
    FCLKQP=>FCLKQP,
    FCLKQN=>FCLKQN,
    PCLK=>PCLK,
    SDTAP=>SDTAP,
    VALUE=>VALUE,
    RESET=>RESET,
    DLYSTEP=>DLYSTEP,
    DF=>DF
);

```

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. Q0 is the double rate data output, Q1 is used for the OEN signal of IOBUF/TBUF connected by Q0. ODDR logic diagram is as shown in Figure 4-25 and its timing diagram is as shown in Figure 4-26.

Figure 4-25 ODDR Logic Diagram

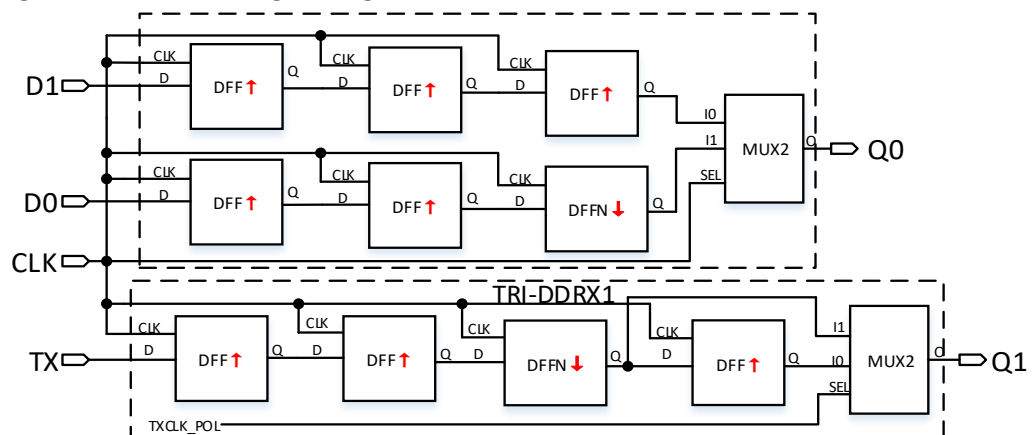
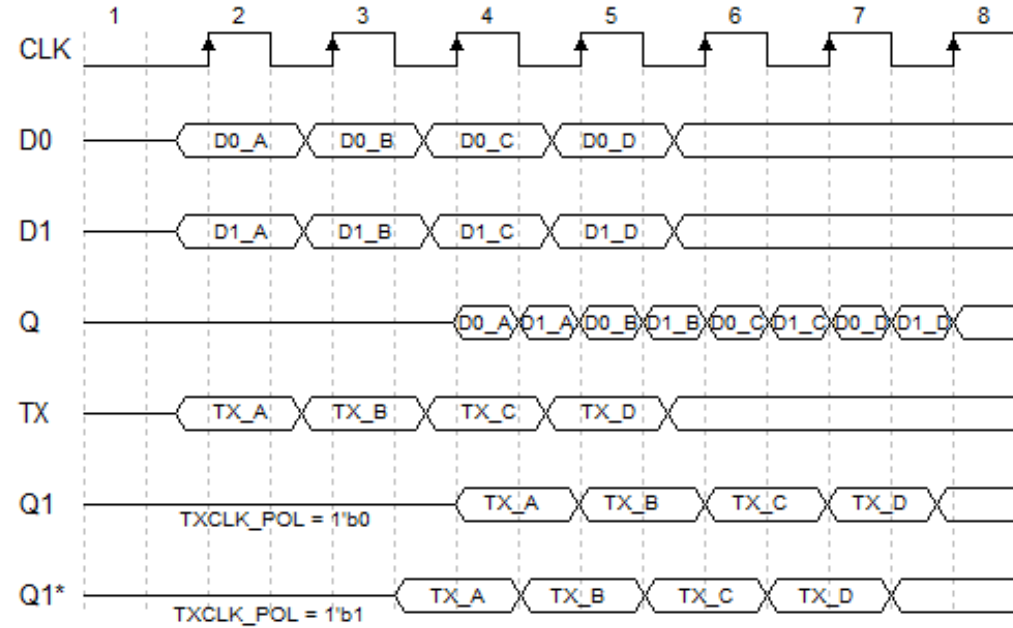


Figure 4-26 ODDR Timing Diagram



Port Diagram

Figure 4-27 ODDR Port Diagram



Port Description

Table 4-27 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 4-28 ODDR Parameter Description

Name	Value	Default	Description
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 output clock polarity control <ul style="list-style-type: none"> 1'b0: Q1 posedge output 1'b1: Q1 negedge output

Connection Rule

- Q0 can be directly connected to OBUF/IOBUF/TBUF, 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.

Verilog Instantiation:

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

Vhdl Instantiation:

```
COMPONENT ODDR
    GENERIC (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 (TXCLK_POL=>'0'
    )
    PORT MAP (
        Q0=>Q0,
        Q1=>Q1,
```

```

D0=>D0,
D1=>D1,
TX=>TX,
CLK=>CLK

```

```

);

```

4.3.2 ODDRC

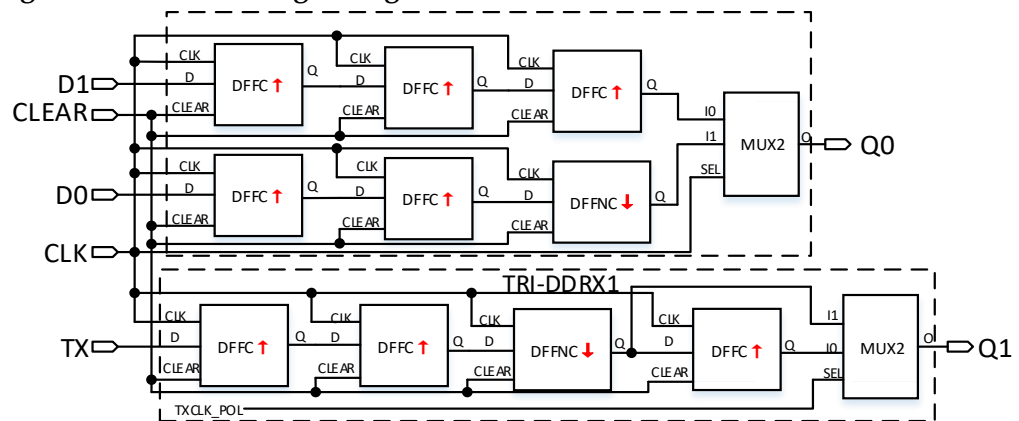
Primitive Introduction

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

Functional Description

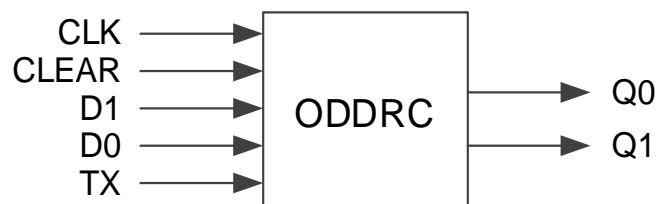
ODDRC mode is used for transferring double data rate signals from FPGA devices. Where Q0 is the double rate data output, Q1 is used for the OEN signal of IOBUF/TBUF connected to Q1. Its logic diagram is as shown in Figure 4-28.

Figure 4-28 ODDRC Logic Diagram



Port Diagram

Figure 4-29 ODDRC Port Diagram



Port Description

Table 4-29 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 clear input signal, active-high.
Q0	Output	ODDRC data output signal
Q1	Output	ODDRC 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 ODDRC Parameter Description

Name	Value	Default	Description
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 output clock polarity control <ul style="list-style-type: none"> 1'b0: Q1 posedge output 1'b1: Q1 negedge output

Connection Rule

- Q0 can directly connect OBUF/IOBUF/TBUF, 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.

Verilog Instantiation:

```

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

```

Vhdl Instantiation:

```

COMPONENT ODDRC
    GENERIC (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 (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

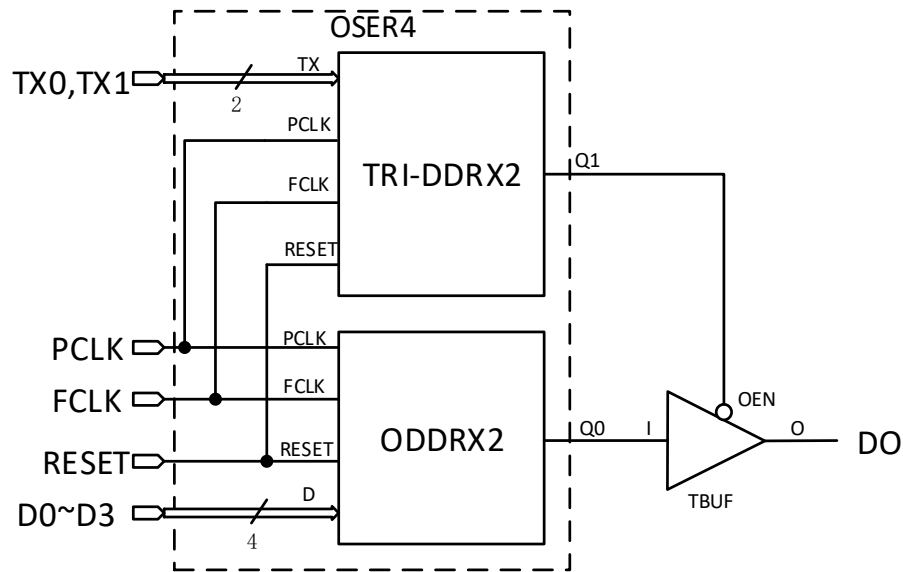
The 4 to 1 Serializer (OSER4) is a serializer of 4 bits parallel input and 1 bit serial output.

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 4-30.

Figure 4-30 OSER4 Logic Diagram

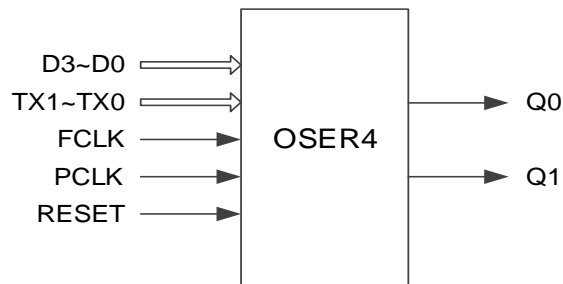


PCLK is usually obtained by FCLK frequency division:

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

Port Diagram

Figure 4-31 OSER4 Port Diagram



Port Description

Table 4-31 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 4-32 IDER4 Parameter Description

Name	Value	Default	Description
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 output clock polarity control <ul style="list-style-type: none"> 1'b0: data posedge output 1'b1: data negedge output
HWL	"false", "true"	"false"	OSER4 data d_up0/1 timing relationship control <ul style="list-style-type: none"> "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/IOBUF/TBUF, 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.

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

Vhdl Instantiation:

```
COMPONENT OSER4
```

```

        GENERIC (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 (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

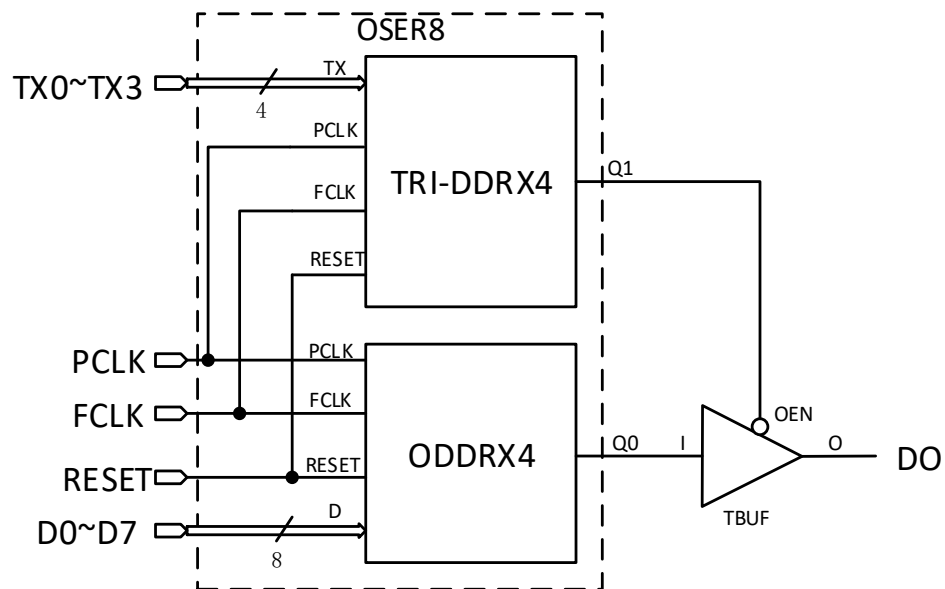
Primitive Introduction

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 4-32.

Figure 4-32 OSER8 Logic Diagram

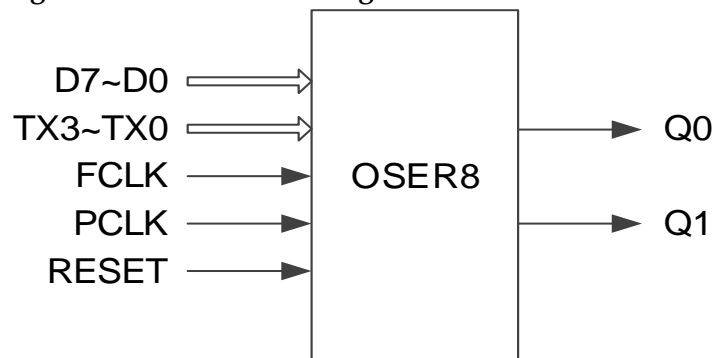


PCLK is usually obtained by FCLK frequency division:

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

Port Diagram

Figure 4-33 OSER8 Port Diagram



Port Description

Table 4-33 OSER8 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 4-34 OSER8 Parameter Description

Name	Value	Default	Description
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 output clock polarity control <ul style="list-style-type: none"> 1'b0: data posedge output 1'b1: data negedge output
HWL	"false", "true"	"false"	OSER8 data d_up0/1 timing relationship control <ul style="list-style-type: none"> "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/IOBUF/TBUF, 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.

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

```

Vhdl Instantiation:

```

COMPONENT OSER8
    GENERIC (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 (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

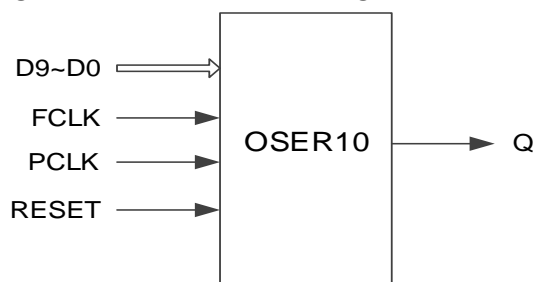
The 10 to 1 Serializer (OSER10) is a serializer of 10 bits parallel input and 1 bit serial output.

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 4-34 OSER10 Port Diagram



Port Description

Table 4-35 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

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.

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)
);

```

Vhdl Instantiation:

```

COMPONENT OSER10
    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
    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

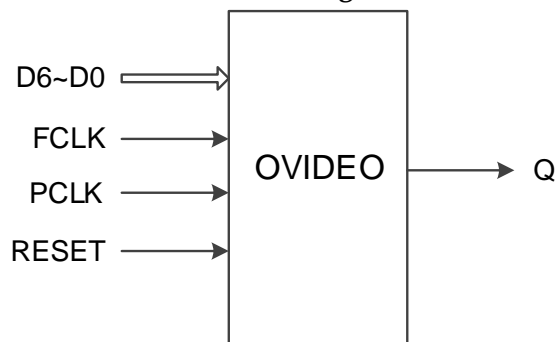
The 7 to 1 Serializer (OVIDEO) is a serializer of 7 bits parallel input and 1 bit serial output,

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 4-35 OVIDEO Port Diagram



Port Description

Table 4-36 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

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.

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)
);
```

Vhdl Instantiation:

```
COMPONENT OVIDEO
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
    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.

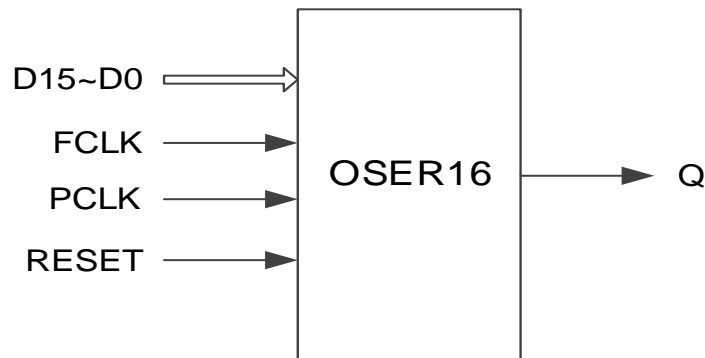
Functional Description

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 4-36 OSER16 Port Diagram



Port Description

Table 4-37 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	OSER16_MEM data output signal

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.

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)  
);
```

Vhdl Instantiation:

```
COMPONENT OSER16  
  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
    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

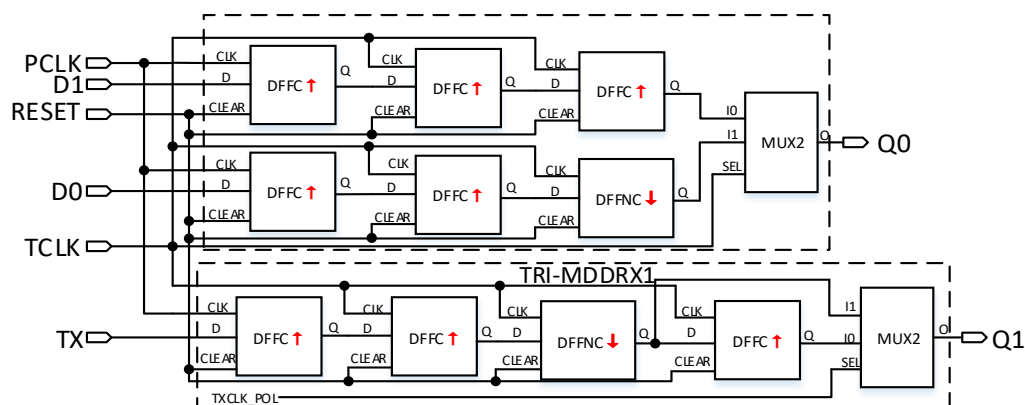
Primitive Introduction

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

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 4-37.

Figure 4-37 ODDR_MEM Logic Diagram

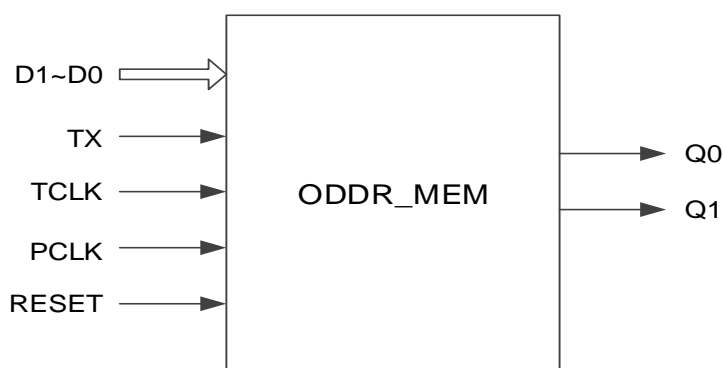


The frequency relation between PCLK and TCLK is $f_{PCLK} = f_{TCLK}$.

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

Port Diagram

Figure 4-38 ODDR_MEM Port Diagram



Port Description

Table 4-38 ODDR_MEM Port Description

Port	I/O	Description
D1~D0	Input	ODDR_MEM data input signal
TX	Input	Q1 generated by TRI-MDDR1
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 4-39 ODDR_MEM Parameter Description

Name	Value	Default	Description
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 output clock polarity control <ul style="list-style-type: none"> 1'b0: data posedge output 1'b1: data negedge output
TCLK_SOURCE	"DQSW", "DQSW270"	"DQSW"	TCLK source selection <ul style="list-style-type: none"> "DQSW" comes from DQSW0 in DQS module. "DQSW270" comes from DQSW270 from DQS module.

Connection Rule

- Q0 can directly connect to OBUF/IOBUF/TBUF, 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.TCLK_SOURCE ="DQSW";
defparam uut.TXCLK_POL=1'b0;

```

Vhdl Instantiation:

```

COMPONENT ODDR_MEM
    GENERIC (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 (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

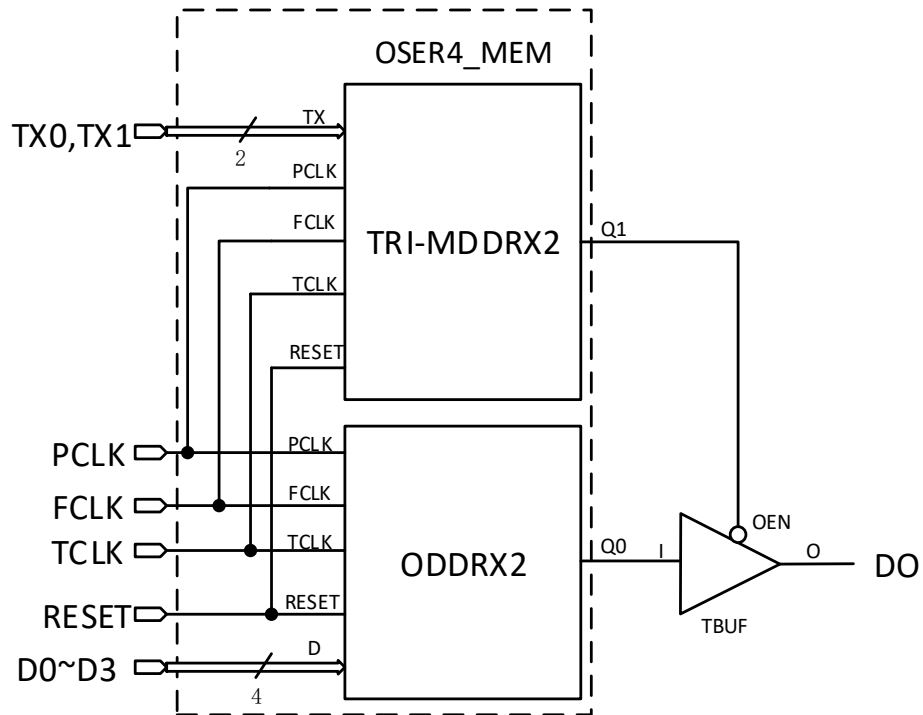
Primitive Introduction

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

Functional Description

OSER4_MEM realizes 4:1 parallel serial conversion. Unlike OSER4, OSER4_MEM needs to be used with DQS. 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 4-39.

Figure 4-39 OSER4_MEM Logic Diagram

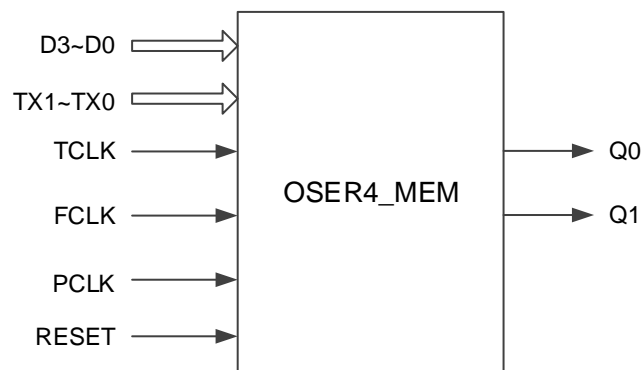


The frequency relation among PCLK, FCLK and TCLK is
 $f_{PCLK} = 1/2 f_{FCLK} = 1/2 f_{TCLK}$.

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

Port Diagram

Figure 4-40 OSER4_MEM Diagram



Port Description

Table 4-40 OSER4_MEM Port Description

Port	I/O	Description
D3~D0	Input	OSER4_MEM data input signal
TX1~TX0	Input	Q1 generated by TRI-MDDR2

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 4-41 OSER4_MEM Parameter Description

Name	Value	Default	Description
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 output clock polarity control <ul style="list-style-type: none"> 1'b0: data posedge output 1'b1: data negedge output
TCLK_SOURCE	"DQSW","DQSW270"	" DQSW "	TCLK source selection <ul style="list-style-type: none"> "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 <ul style="list-style-type: none"> "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/IOBUF/TBUF, 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.HWL="false";
defparam uut.TCLK_SOURCE="DQSW";
defparam uut.TXCLK_POL=1'b0;

```

Vhdl Instantiation:

```

COMPONENT OSER4_MEM
    GENERIC (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 (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

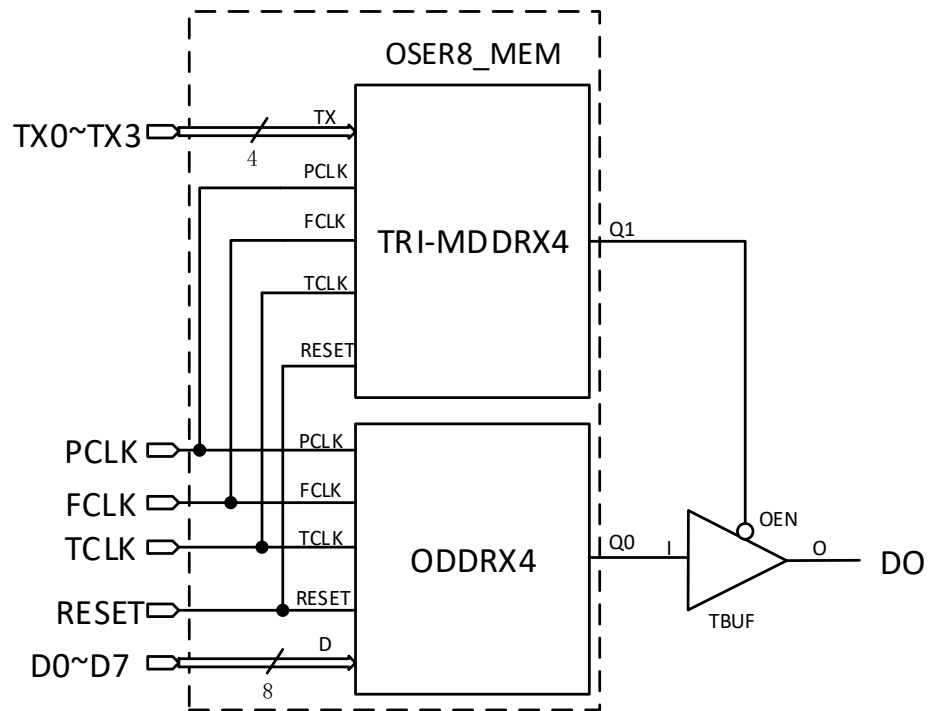
Primitive Introduction

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

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 4-41.

Figure 4-41 OSER8_MEM Logic Diagram

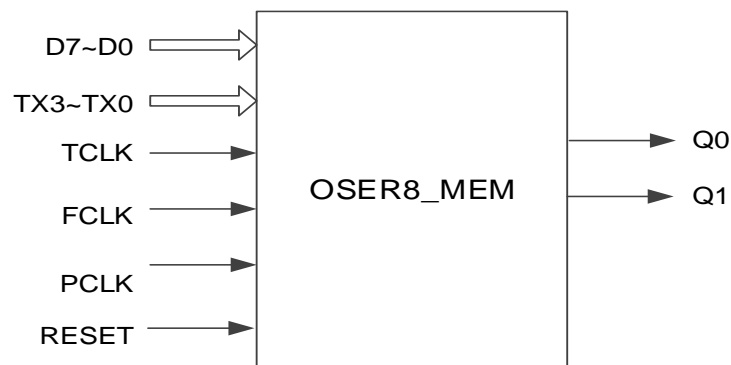


The frequency relation between PCLK, FCLK and TCLK is
 $f_{PCLK} = 1/4 f_{FCLK} = 1/4 f_{TCLK}$.

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

Port Diagram

Figure 4-42 OSER8_MEM Port Diagram



Port Description

Table 4-42 OSER8_MEM Port Description

Port Name	I/O	Description
D7~D0	Input	OSER8_MEM data input signal
TX3~TX0	Input	Q1 generated by TRI-MDDR4

Port Name	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	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 4-43 OSER8_MEM Parameter Description

Name	Value	Default	Description
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 output clock polarity control <ul style="list-style-type: none"> 1'b0: data posedge output 1'b1: data negedge output
TCLK_SOURCE	"DQSW","DQSW270"	" DQSW "	TCLK source selection <ul style="list-style-type: none"> "DQSW" comes from DQSW0 in DQS module. "DQSW270" comes from DQSW270 from DQS module.
HWL	"false", "true"	"false"	OSER8_MEM data d_up0/1 timing relationship control <ul style="list-style-type: none"> "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/IOBUF/TBUF, 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.HWL ="false";
defparam uut.TCLK_SOURCE ="DQSW";
defparam uut.TXCLK_POL=1'b0;

```

Vhdl Instantiation:

```

COMPONENT OSER8_MEM
    GENERIC (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 (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

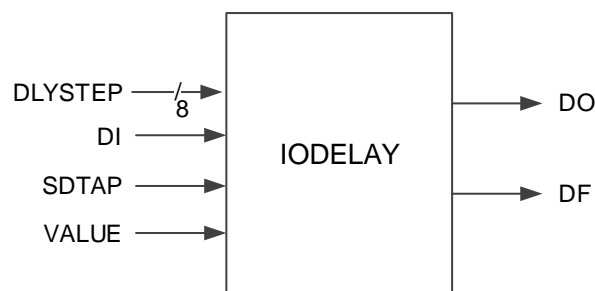
Input/Output delay (IODELAY) is a programmable delay unit in IO module.

Functional Description

Arora V devices support IODELAY module, providing a total of 256 (0~255) delays. The single-step delay time is about 12.5ps. IODELAY can be used for input or output of I/O and general logic. The IODELAY includes IDELAY and ODELAY, which act on input logic and output logic respectively. Delay code has three setting modes: static mode, dynamic mode, and adaptive mode.

Port Diagram

Figure 4-43 IODELAY Port Diagram



Port Description

Table 4-44 IODELAY Port Description

Port	I/O	Description
DI	Input	Data input signal
SDTAP	Input	Control loading static delay step <ul style="list-style-type: none"> ● 0: Load static delay ● 1: Dynamically adjust the delay
VALUE	Input	Dynamically adjust delay value at VALUE falling edge; each pulse moves one delay step.
DLYSTEP[7:0]	Input	Dynamic delay value
DO	Output	Data output signal
DF	Output	Output flag bit

Parameter Description

Table 4-45 IODELAY Parameter Description

Name	Value	Default	Description
C_STATIC_DLY	0~255	0	Static delay step control
DYN_DLY_EN	"FALSE"/"TRUE"	"FALSE"	Dynamic mode enable control
ADAPT_EN	"FALSE"/"TRUE"	"FALSE"	Adaptive mode enable control

Primitive Instantiation

Verilog Instantiation:

```

IODELAY iodelay_inst(
    .DO(dout),
    .DF(df),
    .DI(di),
    .SDTAP(sdtap),
    .VALUE(value),
    .DLYSTEP(dlystep)
);
defparam iodelay_inst.C_STATIC_DLY=0;
defparam iodelay_inst.DYN_DLY_EN="FALSE";
defparam iodelay_inst.ADAPT_EN="FALSE";

```

Vhdl Instantiation:

```

COMPONENT IODELAY
    GENERIC (C_STATIC_DLY:integer:=0;
             DYN_DLY_EN:string:="FALSE";

```

```
        ADAPT_EN:string:="FALSE"
    );
    PORT(
        DO:OUT std_logic;
        DF:OUT std_logic;
        DI:IN std_logic;
        SDTAP:IN std_logic;
        VALUE:IN std_logic;
        DLYSTEP:IN std_logic
    );
END COMPONENT;
uut:IODELAY
    GENERIC MAP (C_STATIC_DLY=>0,
                  DYN_DLY_EN=>"FALSE",
                  ADAPT_EN=>"FALSE"
    )
    PORT MAP (
        DO=>dout,
        DF=>df,
        DI=>di,
        SDTAP=>sdtap,
        VALUE=>value,
        DLYSTEP=>dlystep
    );
```

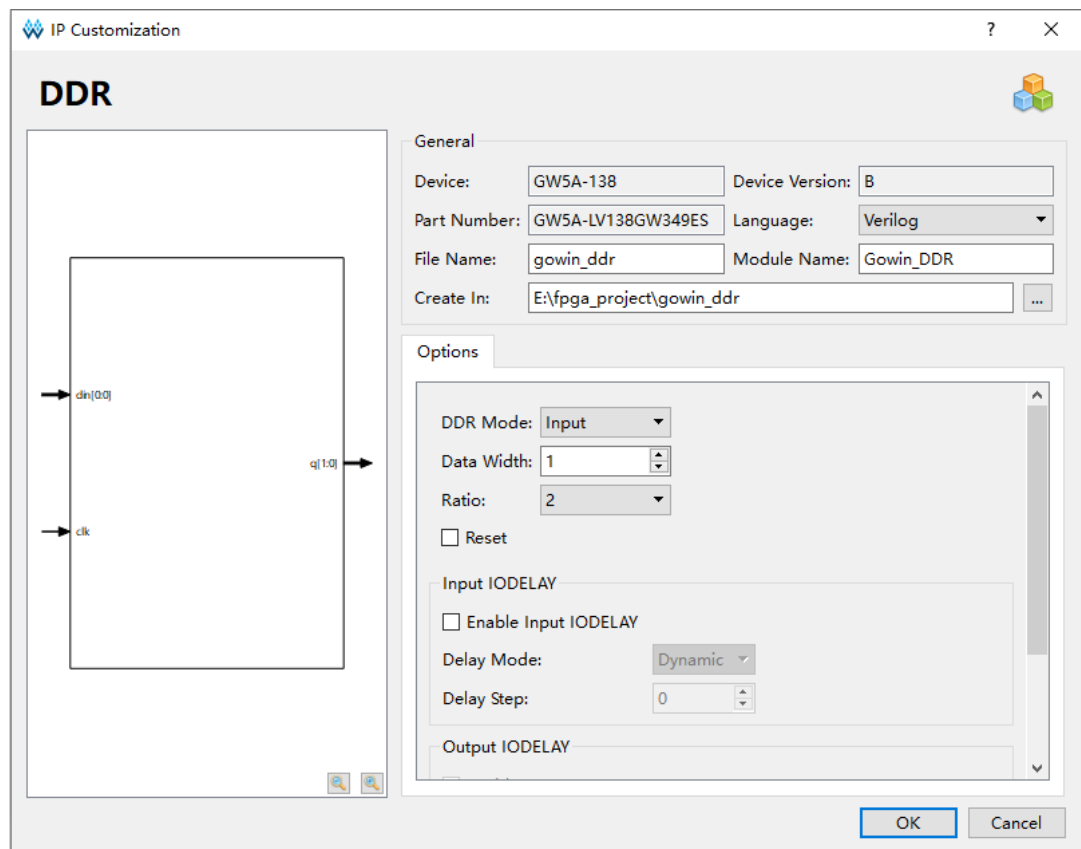
5 IP Generation

Gowin 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 DDR IP Customization



1.General: 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.
- 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 a target path.

2.Options: Options configuration box is used to customize the IP, as shown Figure 5-1.

- DDR Mode: Configures DDR mode, including input, output, tristate and Bidirectional.
- Data Width: Configures the data width of the DDR, and the range is 1~64.
- Ratio: DDR data conversion ratio, including 2, 4,7,8,10,14,16,32.
- Reset: When Ratio is 2, this option can be enabled or disabled, and IDDR or ODDR will be instantiated when enabled.
- Enable Input IODELAY: Configures whether DDR uses a input delay module.
 - "Delay Mode": Configures the delay mode. "Dynamic" means using IODELAY and adjusting the delay step dynamically; "Static" means using IODELAY and adjusting the delay step statically; "Adaptive" means using IODELAY and adjusting the delay step adaptively.
 - "Delay Step": Selects the number of steps in the static delay mode, ranging 0 to 255.
- Enable Output IODELAY: Configures whether DDR uses a output delay module.
 - "Delay Mode": Configures the delay mode. "Dynamic" means using IODELAY and adjusting the delay step dynamically; "Static" means using IODELAY and adjusting the delay step statically; "Adaptive" means using IODELAY and adjusting the delay step adaptively.
 - "Delay Step": Selects the number of steps in the static delay mode, ranging 0 to 255.

- 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.

