Arora V 可编程通用管脚(GPIO)

用户指南

UG304-1.3.2, 2025-02-26

免责声明

本文档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止反言或其它方式授予任何知识产权许可。除高云半导体在其产品的销售条款和条件中声明的责任之外，高云半导体概不承担任何法律或非法律责任。高云半导体对高云半导体产品的销售和／或使用不作任何明示或暗示的担保，包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知识产权的侵权责任等，均不作担保。高云半导体对文档中包含的文字、图片及其它内容的准确性和完整性不承担任何法律或非法律责任，高云半导体保留修改文档中任何内容的权利，恕不另行通知。高云半导体不承诺对这些文档进行适时的更新。

版本信息

日期	版本	说明
2023/04/20	1.0	初始版本。
2023/05/25	1.1	新增第 5 章 IP 调用。
2023/12/29	1.2	● OSIDES32 中新增 FCLK 相位关系描述。 ● 更新表 4-46 IODELAY 端口介绍。
2024/06/28	1.3	● 新增原语：OSIDES64、MIPI_CPHY_IBUF 和 MIPI_CPHY_OBUF。 ● 删除原语：IBUF_R 及 IOBUF_R。 ● 更新 IDDR 时序。
2024/12/31	1.3.1	● 优化 3.3 供电要求。 ● 更新 4.4.1 IODELAY。
2025/02/26	1.3.2	● 优化 3.3.2 差分缓存配置。

目录

图目录

表目录

1 关于本手册

1.1 手册内容

Arora V 可编程通用管脚(GPIO)主要描述了高云®半导体 Arora V FPGA 产品支持的输入输出缓存的电平标准、分区策略和输入输出逻辑的功能，同时阐述了 Arora V GPIO 的架构和高云半导体云源®软件用法以便客户对 GPIO 功能和规则有更深入的理解。

1.2 相关文档

通过登录高云半导体网站 www.gowinsemi.com.cn 可以下载、查看以下相关文档：

- DS981，GW5AT 系列 FPGA 产品数据手册
- DS1103，GW5A 系列 FPGA 产品数据手册
- DS1104，GW5AST 系列 FPGA 产品数据手册
- DS1105，GW5AS 系列 FPGA 产品数据手册
- DS1108, GW5AR 系列 FPGA 产品数据手册
- DS1118，GW5ART 系列 FPGA 产品数据手册
- SUG100，Gowin 云源软件用户指南

1.3 术语、缩略语

表 1-1 列出了本手册中出现的相关术语、缩略语及相关释义。

表 1-1 术语、缩略语

术语、缩略语	全称	含义
Bus Keeper	Bus Keeper	总线保持
CFU	Configurable Function Unit	可配置功能单元
CRU	Configurable Routing Unit	可编程布线单元
DDR	Double Data Rate	双倍速率
DES	Deserializer	解串器

术语、缩略语	全称	含义
ELDO	Emulated LVDS Output	模拟 LVDS 输出(电压输出)
GPIO	Gowin Programmable Input/Output	Gowin 可编程通用管脚
IOB	Input/Output Block	输入输出模块
IO Buffer	Input/Output Buffer	输入输出缓存
IO Logic	Input/Output Logic	输入输出逻辑
Open Drain	Open Drain	漏极开路
SDR	Single Data Rate	单倍速率
SER	Serializer	串行器
TLDO	True LVDS Output	真 LVDS 输出(电流输出)

1.4 技术支持与反馈

高云半导体提供全方位技术支持，在使用过程中如有任何疑问或建议，可直接与公司联系：

网址：www.gowinsemi.com.cn

E-mail：support@gowinsemi.com

Tel：+86 0755 8262 0391

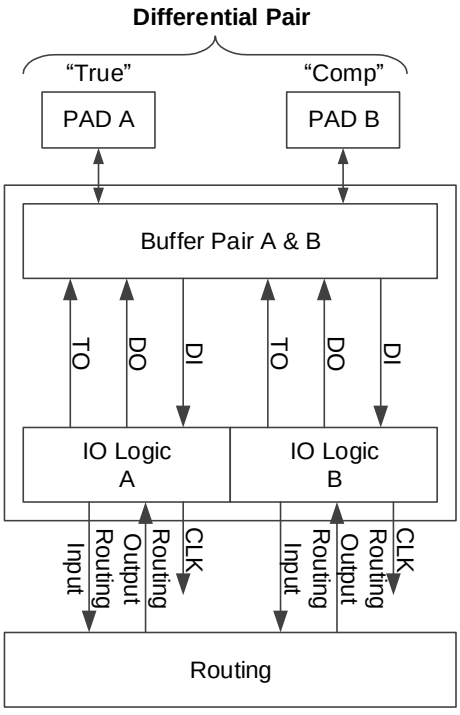
2 GPIO 概述

Arora V FPGA 产品的 GPIO 灵活适配多种业界通用管脚电平标准，从单端电平标准到差分电平标准的支持方便用户对接不同的外部总线、存储器设备、视频应用以及其他标准协议。

Arora V FPGA 产品 GPIO 的基本单元是输入输出模块(IOB)，主要包括输入输出缓存(IO Buffer)、输入输出逻辑(IO Logic)以及相应的可编程布线资源单元三个部分。其中可编程布线资源单元与可配置功能单元(CFU)中的可编程布线单元(CRU)类似。

如图 2-1 所示，每个输入输出模块包括两个输入输出管脚，分别标记为 A 和 B，它们可以配置成一组差分信号对，也可以作为单端信号分别使用。输入输出缓存主要用于支持各种单端电平标准和差分电平标准，输入输出逻辑集成了串并转换、并串转换、延时控制以及字节对齐等功能，主要用于高速数据传输场合。可编程布线资源单元用于输入输出模块和其他片内资源之间的互联。

图 2-1 输入输出模块结构示意图



Arora V FPGA 产品中输入输出模块的功能特点：

- 基于分区(Bank)的管脚供电(V_{CCIO})机制
- 支持 LVCMOS、PCI、LVTTL、SSTL、HSTL、LVDS、Mini_LVDS、RSDS、PPDS、BLVDS 等多种电平标准
- 支持 MIPI 电平标准以及 MIPI I3C OpenDrain/PushPull 转换
- 提供输入信号迟滞选项
- 提供输出信号驱动电流选项
- 提供输出信号 Slew Rate 选项
- 对每个管脚提供独立的总线保持、上拉/下拉电阻及漏极开路输出选项
- 支持热插拔
- 输入输出逻辑支持单倍速率（SDR）模式以及双倍速率（DDR）等多种模式

3 输入输出缓存

3.1 GPIO 电平标准

高云半导体 Arora V FPGA 产品同时支持单端电平标准和差分电平标准。单端电平标准可以采用内置的管脚电压作为参考电压，也可以使用任意一个管脚作为外部参考电压输入。高云半导体 Arora V FPGA 产品所有分区都支持差分输入。模拟 LVDS 差分输出使用外部电阻匹配和差分 LVCMOS 缓存输出实现。特定分区支持真 LVDS 差分输出和差分输入匹配，详细信息请参考 3.2 GPIO 分区策略。

高云半导体 Arora V FPGA 产品不同的电平标准对管脚电压的要求请参考对应数据手册中“I/O 电平标准”相关描述。

3.2 GPIO 分区策略

GPIO 的通用属性：

- 所有分区支持模拟 LVDS 差分输出，但需要使用外部电阻网络；
- 所有分区支持上拉、下拉以及总线保持设置；
- 每个分区支持一种管脚电压；
- 每个分区支持一个参考电压信号，无论它来自外部管脚或者来自内部参考电压生成器。

3.3 供电要求

核电压（V_{CC}）、辅助电压 V_{CCX} 及管脚电压（V_{CCIO}）达到特定阈值[1]时，内部的上电复位信号（POR）会置位，高云半导体 Arora V FPGA 产品内核逻辑被激活。配置过程中，Arora V 138K / 75K/ 60K / 25K FPGA 器件 GPIO 状态由配置管脚“PUDC_B[2]”控制，Arora V 15K FPGA 器件所有 GPIO 默认弱下拉，配置完成后 I/O 状态默认为 None，可通过软件配置，Config 相关 I/O 的状态根据配置模式的不同有所区别。高云半导体 Arora V FPGA 产品对内核电压和管脚电压无上下电顺序要求。

注！

- [1]POR 相关电压值请参考相关数据手册中 3.1.5 POR 特性。

- [2] PUDC_B 相关介绍请参考相关器件编程配置手册。

每个分区支持一个参考电压输入（V_{REF}）。一个分区内的任何管脚可以配置为输入参考电压。为了支持 SSTL，HSTL 等 I/O 输入标准，每个分区还提供一个独立的参考电压（V_{REF}），用户可以选择使用 IOB 内置的 V_{REF} 源，以及基于 V_{CCIO} 的比例电压，也可选择外部的 V_{REF} 输入（使用分区中任意一个 I/O 管脚作为外部 V_{REF} 输入）。

高云半导体 Arora V FPGA 产品的 GPIO 缓存包含两个输入输出管脚，分别标记为 A 和 B。管脚 A 对应于差分信号的 T（True）端，而管脚 B 对应于差分信号的 C（Comp）端。

3.3.1 LVCMOS 缓存配置

所有 GPIO 都包含 LVCMOS 缓存，LVCMOS 缓存可根据不同应用场合配置成多种模式。每个 LVCMOS 缓存可以设置成弱上拉、弱下拉以及总线保持。弱上拉和弱下拉提供了一种固定特征，可以广泛应用于线与、线或等逻辑控制。总线保持以最小功耗锁存信号的上一个状态，关闭总线保持可以降低输入漏电流。

所有 LVCMOS 缓存具有可编程的驱动能力，各种电平标准对应的驱动能力选项请参考对应数据手册中“I/O 电平标准”相关描述。高云半导体 Arora V FPGA 产品可编程的驱动能力仅保证相应设置最小的驱动能力。

迟滞设置主要用于在噪声环境下防止一系列电平的快速跳转，所有 LVCMOS 缓存都支持迟滞设置。

当一个差分对配置成两个单端管脚使用时，管脚间的相对延时最小，信号的一致性最好。

3.3.2 差分缓存配置

当 GPIO 缓存配置成差分模式时，输入迟滞和总线保持特性被禁用。

Arora V 15K 及 60K 器件所有分区支持片内可编程的 100 欧姆输入差分匹配电阻。

Arora V 138K、75K 及 25K 器件 Top 分区及 Bottom 分区支持片内可编程的 100 欧姆输入差分匹配电阻。

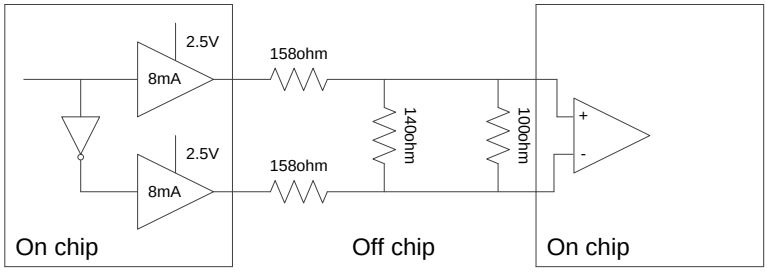
所有单端 GPIO 缓存对都可以配置成模拟 LVDS 差分输出电平标准，比如 LVPECL33E，MLVDS25E，BLVDS25E 等。同时芯片外部需要添加电阻匹配网络。

3.4 模拟差分电路匹配网络

3.4.1 模拟 LVDS

高云半导体 Arora V FPGA 产品通过互补的 LVCMOS 输出加上外部匹配网络可以构建兼容 LVDS 输出标准，其外部匹配网络如图 3-1 所示。

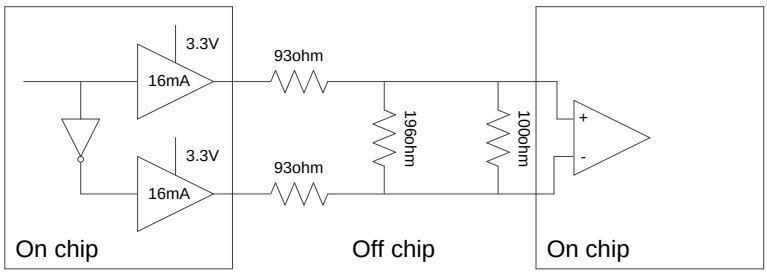
图 3-1 LVDS25E 匹配网络



3.4.2 模拟 LVPECL

高云半导体 Arora V FPGA 产品通过互补的 LVCMOS 输出加上外部匹配网络可以构建兼容 LVPECL 输出标准，其外部匹配网络如图 3-2 所示。

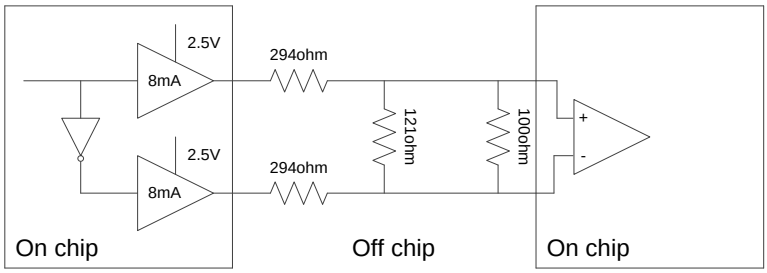
图 3-2 LVPECL 匹配网络



3.4.3 模拟 RSDS

高云半导体 Arora V FPGA 产品通过互补的 LVCMOS 输出加上外部匹配网络可以构建兼容 RSDS 输出标准，其外部匹配网络如图 3-3 所示。

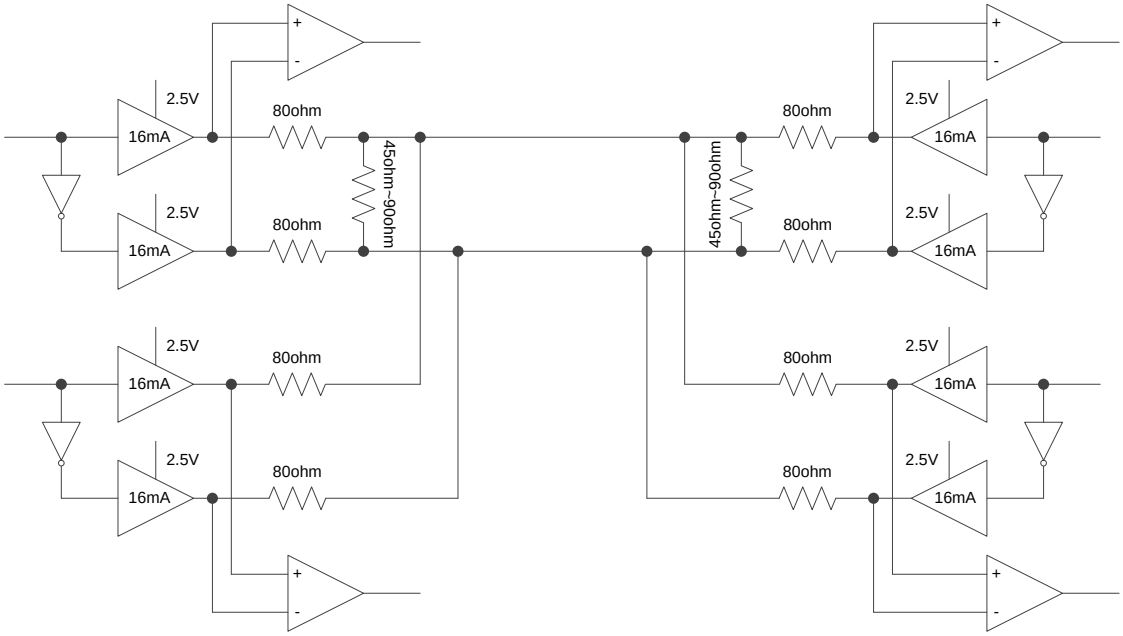
图 3-3 RSDS 匹配网络



3.4.4 模拟 BLVDS

高云半导体 Arora V FPGA 产品通过互补的 LVCMOS 输出加上外部匹配网络可以构建兼容 BLVDS 输出标准，其外部匹配网络如图 3-4 所示。

图 3-4 BLVDS 匹配网络



3.5 GPIO 软件配置

可通过云源软件的 Floorplanner 对 GPIO 位置、属性等进行设置，也可以自定义 CST 文件来实现。下面对 CST 文件支持的物理约束作详细介绍。

3.5.1 位置

对 GPIO 进行物理位置锁定。

IO_LOC "xxx" H4 exclusive;

3.5.2 电平标准

为 GPIO 设置电平标准。

IO_PORT "xxx" IO_TYPE=LVCMOS18D;

3.5.3 驱动能力

为输出管脚或双向管脚设置驱动能力。

IO_PORT "xxx" DRIVE=12;

3.5.4 上下拉模式

设置上下拉模式，其中 UP：上拉；DOWN：下拉；KEEPER：总线保持；NONE：高阻。

IO_PORT "xxx" PULL_MODE=DOWN;

3.5.5 参考电压

为 GPIO 设置参考电压，既可以来自外部管脚也可以来自内部参考电压生成器。

IO_PORT "xxx" VREF=VREF1_LOAD;

3.5.6 迟滞

为输入管脚或双向管脚设置迟滞量，从小到大依次是 NONE->H2L->L2H->HIGH。

IO_PORT "xxx" HYSTERESIS=L2H;

3.5.7 漏极开路

为输出管脚或双向管脚打开或关闭漏极开路，提供 ON/OFF 选项。

IO_PORT "xxx" OPEN_DRAIN=ON;

3.5.8 单端匹配电阻

为单端信号设置终端匹配电阻，提供 OFF 和 ON 选项。

IO_PORT "xxx" SINGLE_RESISTOR=ON;

3.5.9 差分匹配电阻

为差分信号设置终端匹配电阻，提供 OFF 和 ON 选项。

IO_PORT "xxx" DIFF_RESISTOR=ON;

3.5.10 PCI Clamp

支持 PCI 钳位二极管的开或关，钳位二极管的打开可以限制输入输出引脚上出现的过冲，提供 OFF 和 ON 选项。

IO_PORT "xxx" PCI_CLAMP=ON;

3.5.11 上拉/下拉强度

支持上拉/下拉强度的设置，以提供引脚上不同的上拉/下拉能力，提供 MEDIUM、WEAK 和 STRONG 选项。

IO_PORT "xxx" PULL_STRENGTH=MEDIUM;

3.6 GPIO 原语

IO Buffer，具有缓存功能。根据不同功能，可分为普通 buffer、模拟 LVDS（ELVDS）和真 LVDS（TLVDS）。

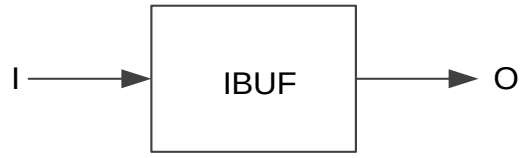
3.6.1 IBUF

原语介绍

IBUF(Input Buffer)，输入缓冲器。

端口示意图

图 3-5 IBUF 端口示意图



端口介绍

表 3-1 IBUF 端口介绍

端口	I/O	描述
I	Input	数据输入信号
O	Output	数据输出信号

原语例化

Verilog 例化：

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

Vhdl 例化：

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

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

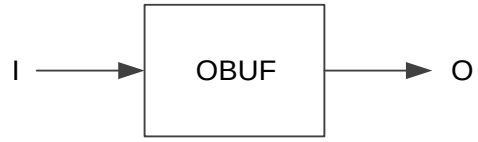
3.6.2 OBUF

原语介绍

OBUF(Output Buffer)，输出缓冲器。

端口示意图

图 3-6 OBUF 端口示意图



端口介绍

表 3-2 OBUF 端口介绍

端口	I/O	描述
I	Input	数据输入信号
O	Output	数据输出信号

原语例化

Verilog 例化：

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

Vhdl 例化：

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

```
);
```

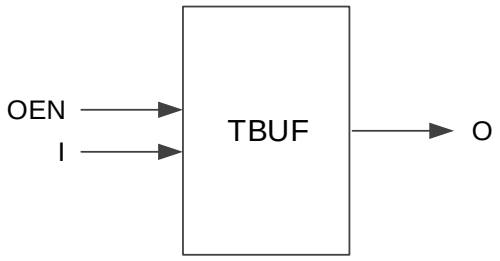
3.6.3 TBUF

原语介绍

TBUF(Output Buffer with Tristate Control)，三态缓冲器，低电平使能。

端口示意图

图 3-7 TBUF 端口示意图



端口介绍

表 3-3 TBUF 端口介绍

端口	I/O	描述
I	Input	数据输入信号
OEN	Input	输出三态使能信号
O	Output	数据输出信号

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT TBUF
    PORT (
        O:OUT std_logic;
        I:IN std_logic;
        OEN:IN std_logic
    );
```

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

3.6.4 IOBUF

原语介绍

IOBUF (Bi-Directional Buffer)，双向缓冲器。当 OEN 为高电平时，作为输入缓冲器；OEN 为低电平时，作为输出缓冲器。

端口示意图

图 3-8 IOBUF 端口示意图

端口介绍

表 3-4 IOBUF 端口介绍

端口	I/O	描述
I	Input	数据输入信号
OEN	Input	输出三态使能信号
IO	Inout	输入输出信号，双向。
O	Output	数据输出信号

原语例化

Verilog 例化：

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

Vhdl 例化：

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

3.6.5 LVDS Input Buffer

原语介绍

LVDS 差分输入：TLVDS_IBUF。

TLVDS_IBUF(True LVDS Input Buffer)，真差分输入缓冲器。

端口示意图

图 3-9 TLVDS_IBUF 端口示意图

端口介绍

表 3-5 TLVDS_IBUF 端口介绍

端口	I/O	描述
I	Input	差分输入 A 端信号
IB	Input	差分输入 B 端信号
O	Output	数据输出信号

原语例化

Verilog 例化：

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

Vhdl 例化：

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

3.6.6 LVDS Ouput Buffer

原语介绍

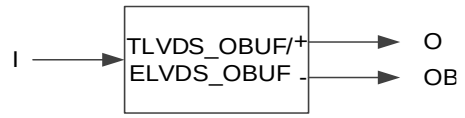
LVDS 差分输出分为两种：TLVDS_OBUF 和 ELVDS_OBUF。

TLVDS_OBUF(True LVDS Output Buffer)，真差分输出缓冲器。

ELVDS_OBUF(Emulated LVDS Output Buffer)，模拟差分输出缓冲器。

端口示意图

图 3-10 TLVDS_OBUF/ELVDS_OBUF 端口示意图



端口介绍

表 3-6 TLVDS_OBUF/ELVDS_OBUF 端口介绍

端口	I/O	描述
I	Input	数据输入信号

端口	I/O	描述
OB	Output	B 端差分输出信号
O	Output	A 端差分输出信号

原语例化

示例一

Verilog 例化：

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

Vhdl 例化：

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

示例二

Verilog 例化：

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

Vhdl 例化：

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

3.6.7 LVDS Tristate Buffer

原语介绍

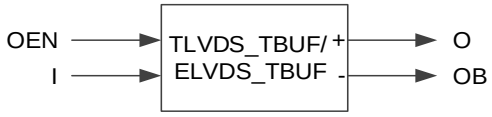
LVDS 三态差分输出分为两种：TLVDS_TBUF 和 ELVDS_TBUF。

TLVDS_TBUF(True LVDS Tristate Buffer)，真差分三态缓冲器，低电平使能。

ELVDS_TBUF(Emulated LVDS Tristate Buffer)，模拟差分三态缓冲器，低电平使能。

端口示意图

图 3-11 TLVDS_TBUF/ELVDS_TBUF 端口示意图



端口介绍

表 3-7 TLVDS_TBUF/ELVDS_TBUF 端口介绍

端口	I/O	描述
I	Input	数据输入信号
OEN	Input	输出三态使能信号
OB	Output	B 端差分输出信号
O	Output	A 端差分输出信号

原语例化

示例一

Verilog 例化：

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

Vhdl 例化：

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

示例二

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT ELVDS_TBUF
  PORT (
      O:OUT std_logic;
      OB:OUT std_logic;
```

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

3.6.8 LVDS Inout Buffer

原语介绍

LVDS 差分输入输出分为两种：TLVDS_IOBUF 和 ELVDS_IOBUF。

TLVDS_IOBUF(True LVDS Bi-Directional Buffer)，真差分双向缓冲器，当 OEN 为高电平时，作为真差分输入缓冲器；OEN 为低电平时，作为真差分输出缓冲器。

ELVDS_IOBUF(Emulated LVDS Bi-Directional Buffer)，模拟差分双向缓冲器，当 OEN 为高电平时，作为模拟差分输入缓冲器；OEN 为低电平时，作为模拟差分输出缓冲器。

端口示意图

图 3-12 TLVDS_IOBUF/ELVDS_IOBUF 端口示意图

端口介绍

表 3-8 TLVDS_IOBUF/ELVDS_IOBUF 端口介绍

端口	I/O	描述
I	Input	数据输入信号
OEN	Input	输出三态使能信号
O	Output	数据输出信号
IOB	Inout	B 端差分输入输出
IO	Inout	A 端差分输入输出

原语例化

Verilog 例化：

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

Vhdl 例化：

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

3.6.9 MIPI_IBUF

原语介绍

MIPI_IBUF(MIPI Input Buffer)有两种工作模式：HS 输入模式和 LP 双向模式，其中 HS 模式支持动态电阻配置。

功能描述

MIPI_IBUF 支持 LP 、HS 模式，IO，IOB 连接到 pad。

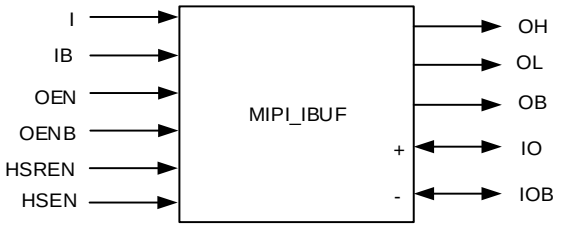
LP 模式：支持双向，OEN 低电平时，I 为输入 IO 为输出；OEN 高电平时，IO 为输入 OL 为输出；OENB 低电平时，IB 为输入 IOB 为输出；

OENB 高电平时，IOB 为输入 OB 为输出。

HS 模式：IO、IOB 为差分输入，OH 为输出，此时 HSREN 控制终端电阻，HSEN 控制 HS 模式使能。

端口示意图

图 3-13 MIPI_IBUF 端口示意图



端口介绍

表 3-9 MIPI_IBUF 端口介绍

端口	I/O	描述
I	Input	LP 模式下，OEN 低电平时 I 为输入。
IB	Input	LP 模式下，OENB 低电平时 IB 为输入。
HSREN	Input	HS 模式下控制终端电阻
HSEN	Input	使能 HS 模式
OEN	Input	LP 模式下输入输出三态控制信号
OENB	Input	LP 模式下输入输出三态控制信号
OH	Output	HS 模式下数据输出信号
OL	Output	LP 模式下，OEN 高电平时 OL 为输出。
OB	Output	LP 模式下，OENB 高电平时 OB 为输出。
IO	Inout	● LP 模式下，OEN 低电平时 IO 为输出，OEN 高电平时 IO 为输入； ● HS 模式下，IO 为输入。
IOB	Inout	● LP 模式下，OENB 低电平时 IOB 为输出，OENB 高电平时 IOB 为输入； ● HS 模式下，IOB 为输入。

原语例化

Verilog 例化：

```
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 例化：

```
COMPONENT MIPI_IBUF
  PORT (
        OH:OUT std_logic;
        OL: OUT std_logic;
        OB:OUT std_logic;
        IO:INOUT std_logic;
        IOB:INOUT std_logic;
     I:IN std_logic;
        IB:IN std_logic;
    OEN:IN std_logic;
        OENB:IN std_logic;
        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

原语介绍

MIPI_OBUF_A 有两种工作模式：HS 模式和 LP 模式。

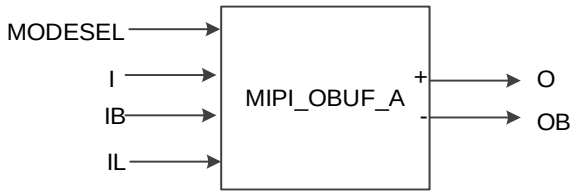
MIPI_OBUF_A(MIPI Output Buffer)，MIPI 输出缓冲器，当 MODESEL 为高电平时，作为(HS)MIPI 高速输出缓冲器；当 MODESEL 为低电平时，作为(LP)MIPI 低功耗输出缓冲器。

注！

GW5A-138B、GW5AS-138B、GW5AST-138B、GW5AT-138B、GW5AT-75B 器件不支持 MIPI_OBUF_A。

端口示意图

图 3-14 MIPI_OBUF_A 端口示意图



端口介绍

表 3-10 MIPI_OBUF_A 端口介绍

端口	I/O	描述
I	Input	HS 模式下 A 端数据输入信号
IB	Input	LP 模式下 B 端数据输入信号
IL	Input	LP 模式下 A 端数据输入信号
MODESEL	Input	模式选择信号，HS 或 LP 模式。
O	Output	A 端数据输出信号，HS 模式下为 A 差分输出，LP 模式下为 A 单端输出。
OB	Output	B 端数据输出信号，HS 模式下为 B 差分输出，LP 模式下为 B 单端输出。

原语例化

Verilog 例化：

```
MIPI_OBUF_A uut(
    .O(O),
```

```
    .OB(OB),
    .I(I),
    .IB(IB),
    .IL(IL),
    .MODESEL(MODESEL)
);
```

Vhdl 例化：

```
COMPONENT MIPI_OBUF_A
  PORT (
        O:OUT std_logic;
        OB:OUT std_logic;
        I:IN std_logic;
        IB:IN std_logic;
        IL: IN std_logic;
        MODESEL:IN std_logic
  );
END COMPONENT;
uut: MIPI_OBUF_A
    PORT MAP(
      O=>O,
      OB=>OB,
      I=>I,
      IB=>IB,
      IL=>IL,
      MDOESEL=>MODESEL
    );
```

3.6.11 MIPI_CPHY_IBUF

原语介绍

MIPI_CPHY_IBUF(MIPI CPHY Input Buffer)有两种工作模式：HS 输入模式和 LP 双向模式。

功能描述

MIPI_CPHY_IBUF 支持 LP 、HS 模式，由三对 IO 组成，IO0，IOB0，IO1，IOB1，IO2，IOB2 连接到 pad。

LP 模式：支持双向，OEN 低电平时，I0/I1/I2 为输入 IO0/ IO1/ IO2 为

输出；OEN 高电平时，IO0/ IO1/ IO2 为输入 OL0/ OL1/ OL2 为输出；OENB 低电平时，IB0/ IB1/ IB2 为输入 IOB0/ IOB1/ IOB2 为输出；OENB 高电平时，IOB0/ IOB1/ IOB2 为输入 OB0/ OB1/ OB2 为输出。

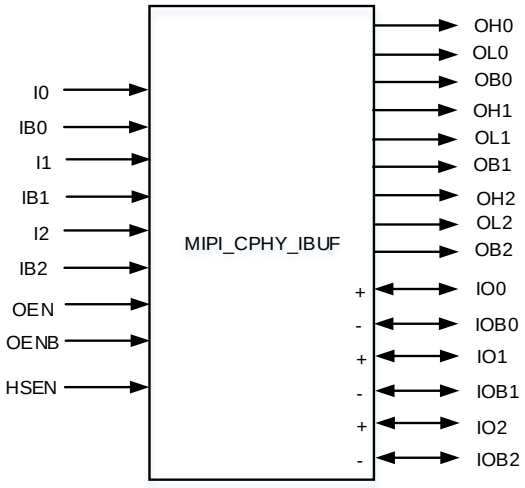
HS 模式：IO0、IOB0，IO1、IOB1，IO2、IOB2，为差分输入，OH0，OH1，OH2 为输出，此时 HSEN 控制 HS 模式使能。

注！

GW5A-138B、GW5AS-138B、GW5AST-138B、GW5AT-138B、GW5AT-75B 器件不支持 MIPI_CPHY_IBUF。

端口示意图

图 3-15 MIPI_CPHY_IBUF 端口示意图



端口介绍

表 3-11 MIPI_CPHY_IBUF 端口介绍

端口	I/O	描述
I0, I1, I2	Input	LP 模式下，OEN 低电平时 I0, I1, I2 为输入。
IB0, IB1, IB2	Input	LP 模式下，OENB 低电平时 IB0, IB1, IB2 为输入。
HSEN	Input	使能 HS 模式
OEN	Input	LP 模式下输入输出三态控制信号
OENB	Input	LP 模式下输入输出三态控制信号
OH0, OH1, OH2	Output	HS 模式下数据输出信号
OL0, OL1, OL2	Output	LP 模式下，OEN 高电平时 OL0, OL1, OL2 为输出。
OB0, OB1, OB2	Output	LP 模式下，OENB 高电平时 OB0, OB1, OB2 为输出。
IO0, IO1, IO2	Inout	● LP 模式下，OEN 低电平时 IO0, IO1, IO2 为输出，OEN 高电平时 IO0, IO1, IO2 为输入； ● HS 模式下，IO0, IO1, IO2 为输入。

端口	I/O	描述
IOB0, IOB1, IOB2	Inout	● LP 模式下，OENB 低电平时 IOB0, IOB1, IOB2 为输出，OENB 高电平时 IOB0, IOB1, IOB2 为输入； ● HS 模式下，IOB0, IOB1, IOB2 为输入。

原语例化

Verilog 例化：

```
MIPI_CPHY_IBUF uut(
    .OH0(OH0),
    .OL0(OL0),
    .OB0(OB0),
    .OH1(OH1),
    .OL1(OL1),
    .OB1(OB1),
    .OH2(OH2),
    .OL2(OL2),
    .OB2(OB2),
    .IO0(IO0),
    .IOB0(IOB0),
    .IO1(IO1),
    .IOB1(IOB1),
    .IO2(IO2),
    .IOB2(IOB2),
    .I0(I0),
    .IB0(IB0),
    .I1(I1),
    .IB1(IB1),
    .I2(I2),
    .IB2(IB2),
    .OEN(OEN),
    .OENB(OENB),
    .HSEN(HSEN)
);
```

Vhdl 例化：

```
COMPONENT MIPI_CPHY_IBUF
```

```
    PORT (
            OH0, OH1, OH2:OUT std_logic;
            OL0, OL1, OL2: OUT std_logic;
            OB0, OB1, OB2:OUT std_logic;
            IO0, IO1, IO2:INOUT std_logic;
            IOB0, IOB1, IOB2:INOUT std_logic;
            I0, I1, I2:IN std_logic;
            IB0, IB1, IB2:IN std_logic;
            OEN:IN std_logic;
            OENB:IN std_logic;
            HSEN:IN std_logic
    );
END COMPONENT;
uut: MIPI_CPHY_IBUF
     PORT MAP(
            OH0=>OH0,
            OL0=>OL0,
            OB0=>OB0,
            IO0=>IO0,
            IOB0=>IOB0,
            I0=>I0,
            IB0=>IB0,
            OH1=>OH1,
            OL1=>OL1,
            OB1=>OB1,
            IO1=>IO1,
            IOB1=>IOB1,
            I1=>I1,
            IB1=>IB1,
            OH2=>OH2,
            OL2=>OL2,
            OB2=>OB2,
            IO2=>IO2,
            IOB2=>IOB2,
            I2=>I2,
```

```
        IB2=>IB2,
        OEN=>OEN,
        OENB=>OENB,
        HSEN=>HSEN
    );
```

3.6.12 MIPI_CPHY_OBUF

原语介绍

MIPI_CPHY_OBUF 有两种工作模式：HS 模式和 LP 模式。

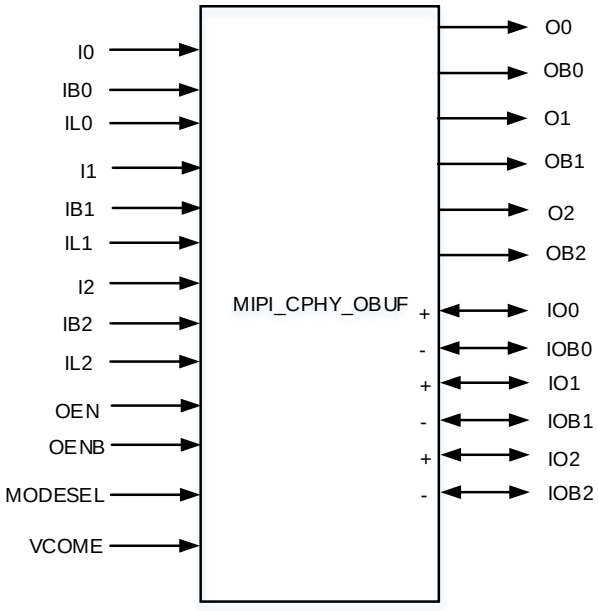
MIPI_CPHY_OBUF (MIPI CPHY Output Buffer)，由三对 IO 组成。当 MODESEL 为高电平时，作为(HS)MIPI CPHY 高速输出缓冲器；当 MODESEL 为低电平时，作为(LP)MIPI CPHY 低功耗双向缓冲器。

注！

GW5A-138B、GW5AS-138B、GW5AST-138B、GW5AT-138B、GW5AT-75B 器件不支持 MIPI_CPHY_OBUF。

端口示意图

图 3-16 MIPI_CPHY_OBUF 端口示意图



端口介绍

表 3-12 MIPI_CPHY_OBUF 端口介绍

端口	I/O	描述
I0, I1, I2	Input	HS 模式下 A 端数据输入信号
IB0, IB1, IB2	Input	LP 模式下 B 端数据输入信号

端口	I/O	描述
IL0, IL1, IL2	Input	LP 模式下 A 端数据输入信号
MODESEL	Input	模式选择信号，HS 或 LP 模式
VCOME	Input	共模电压使能信号，高电平有效
O0, O1, O2	Output	LP 模式下 A 单端输出
OB0, OB1, OB2	Output	LP 模式下 B 单端输出
IO0, IO1, IO2	Inout	● LP 模式下，OEN 低电平时 IO0, IO1, IO2 为输出，OEN 高电平时 IO0, IO1, IO2 为输入； ● HS 模式下，IO0, IO1, IO2 为差分 A 端输出
IOB0, IOB1, IOB2	Inout	● LP 模式下，OENB 低电平时 IOB0, IOB1, IOB2 为输出，OENB 高电平时 IOB0, IOB1, IOB2 为输入； ● HS 模式下，IOB0, IOB1, IOB2 为差分 B 端输出

原语例化

Verilog 例化：

```
MIPI_OBUF_A uut(
    .O0(O0),
    .OB0(OB0),
    .O1(O1),
    .OB1(OB1),
    .O2(O2),
    .OB2(OB2),
    .I0(I0),
    .IB0(IB0),
    .IL0(IL0),
    .I1(I1),
    .IB1(IB1),
    .IL1(IL1),
    .I2(I2),
    .IB2(IB2),
    .IL2(IL2),
    .IO0(IO0),
    .IOB0(IOB0),
```

```
    .IO1(IO1),
    .IOB1(IOB1),
    .IO2(IO2),
    .IOB2(IOB2),
    .OEN(OEN),
    .OENB(OENB),
    .VCOME(VCOME),
    .MODESEL(MODESEL)
);
```

Vhdl 例化：

```
COMPONENT MIPI_OBUF_A
  PORT (
        O0,O1,O2:OUT std_logic;
        OB0,OB1,OB2:OUT std_logic;
        I0,I1,I2:IN std_logic;
        IB0,IB1,IB2:IN std_logic;
        IL0,IL1,IL2: IN std_logic;
        OEN,OENB: IN std_logic;
        VCOME : IN std_logic;
        MODESEL:IN std_logic
  );
END COMPONENT;
uut: MIPI_OBUF_A
    PORT MAP(
      O0=>O0,
      OB0=>OB0,
      O1=>O1,
      OB1=>OB1,
      O2=>O2,
      OB2=>OB2,
      I0=>I0,
      IB0=>IB0,
      IL0=>IL0,
      I1=>I1,
      IB1=>IB1,
```

```
    IL1=>IL1,
    I2=>I2,
    IB2=>IB2,
    IL2=>IL2,
    OEN=>OEN,
    OENB=>OENB,
    VCOME=> VCOME,
    MDOESEL=>MODESEL
);
```

3.6.13 I3C_IOBUF

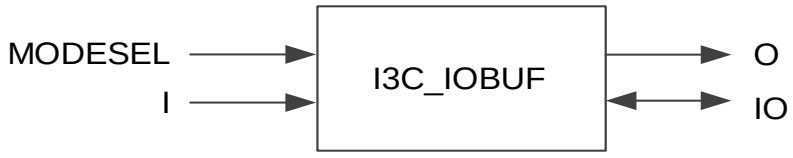
原语介绍

I3C_IOBUF 有两种工作模式：Normal 模式和 I3C 模式。

I3C_IOBUF(I3C Bi-Directional Buffer)，I3C 双向缓冲器，当 MODESEL 为高电平时，作为 I3C 双向缓冲器；当 MODESEL 为低电平时，作为普通双向缓冲器。

端口示意图

图 3-17 I3C_IOBUF 端口示意图



端口介绍

表 3-13 I3C_IOBUF 端口介绍

端口	I/O	描述
I	Input	数据输入信号
IO	Inout	输入输出信号，双向
MODESEL	Input	模式选择信号，Normal 模式或 I3C 模式。
O	Output	数据输出信号

原语例化

Verilog 例化：

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

```
    .I(I),
    .MODESEL(MODESEL)
);
```

Vhdl 例化：

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

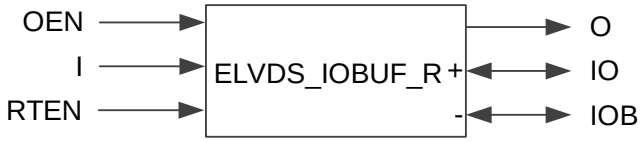
3.6.14 ELVDS_IOBUF_R

原语介绍

ELVDS_IOBUF_R(Emulated LVDS Bi-Directional Buffer with dynamic ODT)，模拟差分动态 ODT 双向缓冲器。

端口示意图

图 3-18 ELVDS_IOBUF_R 端口示意图



端口介绍

表 3-14 ELVDS_IOBUF_R 端口介绍

端口	I/O	描述
I	Input	数据输入信号
OEN	Input	输出三态使能信号
RTEN	Input	动态使能 ODT 电阻
IO	Inout	A 端差分输入输出，双向。
IOB	Inout	B 端差分输入输出，双向。
O	Output	数据输出信号

原语例化

Verilog 例化：

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

Vhdl 例化：

```
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

原语介绍

TLVDS_IBUF_ADC ，真差分输入缓冲器，和 ADC 模块配合使用，主要实现 ADC 的动态电压源选择功能。

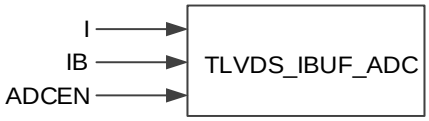
适用器件

表 3-15 TLVDS_IBUF_ADC 适用器件

家族	系列	器件
晨熙®	GW5AT	B 版 GW5AT-138
	GW5AST	B 版 GW5AST-138
	GW5A	GW5A-25

端口示意图

图 3-19 TLVDS_IBUF_ADC 端口示意图



端口介绍

表 3-16 TLVDS_IBUF_ADC 端口介绍

端口	I/O	描述
I	Input	差分输入 A 端信号
IB	Input	差分输入 B 端信号
ADCEN	Input	ADC 动态使能信号

原语例化

Verilog 例化：

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

```
);
```

Vhdl 例化：

```
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 输入输出逻辑

高云半导体 Arora V FPGA 产品的输入输出逻辑支持 SDR、DDR 等工作模式。每一种工作模式下，管脚控制(或管脚差分信号对)又可以配置成输出信号、输入信号、双向信号及三态输出信号(带三态控制的输出信号)。

图 4-1 为高云半导体 Arora V FPGA 产品输入输出逻辑的输出部分。

图 4-1 输入输出逻辑输出示意图 – 输出部分

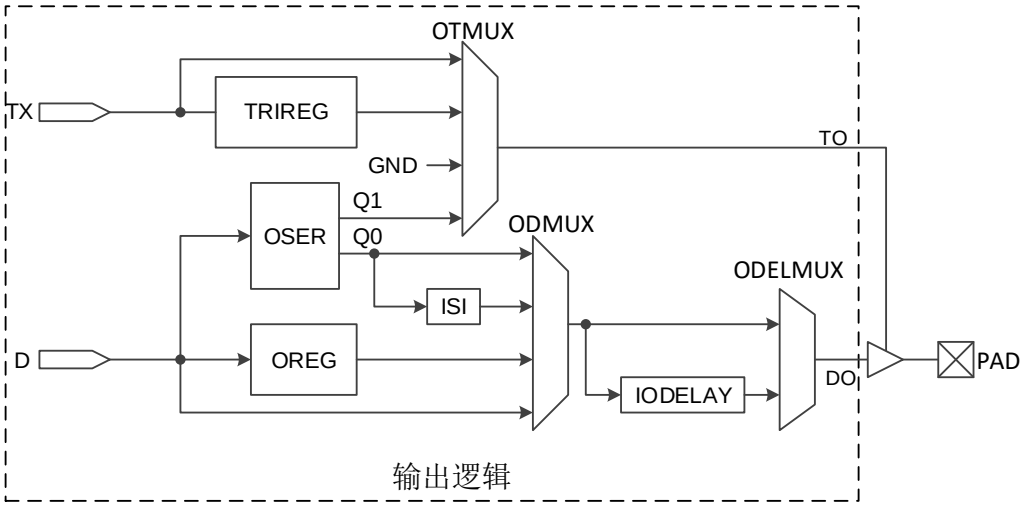
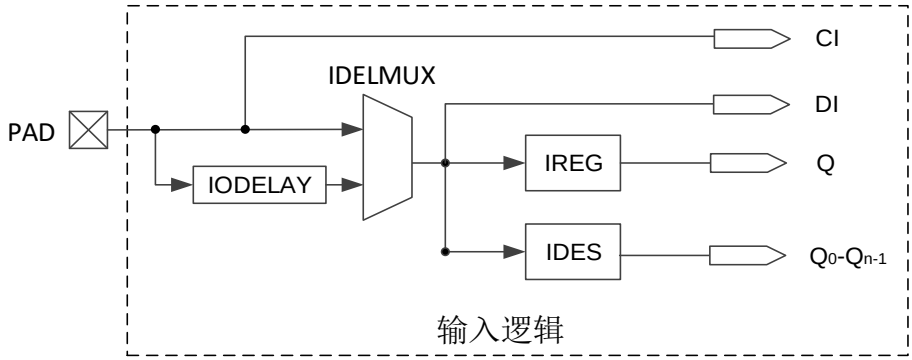


图 4-2 为高云半导体 Arora V FPGA 产品的输入输出逻辑的输入部分。

图 4-2 输入输出逻辑输入示意图 – 输入部分



注！

CI 为 GCLK 输入信号，不能连接到 Fabric；DI 直接输入到 Fabric。

4.1 SDR 模式

输入输出逻辑支持 SDR 模式，提供输入寄存器（IREG）、输出寄存器（OREG）和三态控制寄存器（TRIREG），其功能同 CFU 中的 FF/LATCH。当 FF/LATCH 的输入 D 被 Buffer/IODELAY 驱动，且该 Buffer/IODELAY 不驱动其他 Iologic 时，或当 FF/LATCH 的输出 Q 唯一驱动 Buffer/IODELAY，且该 Buffer 不是 MIPI Buffer 时，可以作为 IOLOGIC 使用。

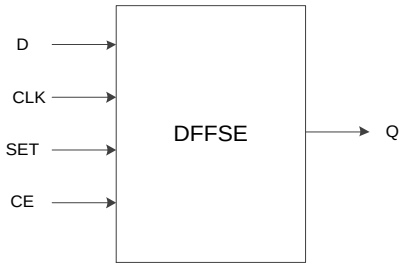
4.1.1 DFFSE

原语介绍

DFFSE(D Flip-Flop with Clock Enable and Synchronous Set)是上升沿触发的 D 触发器，具有同步置位和时钟使能功能。

端口示意图

图 4-3 DFFSE 端口示意图



端口介绍

表 4-1 DFFSE 端口介绍

端口名	I/O	描述
D	Input	数据输入信号
CLK	Input	时钟输入信号
SET	Input	同步置位信号，高电平有效
CE	Input	时钟使能信号
Q	Output	数据输出信号

参数介绍

表 4-2 DFFSE 参数介绍

参数名	取值范围	默认值	描述
INIT	1’b1	1’b1	DFFSE 初始值

原语例化

Verilog 例化：

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

Vhdl 例化：

```
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
      );
```

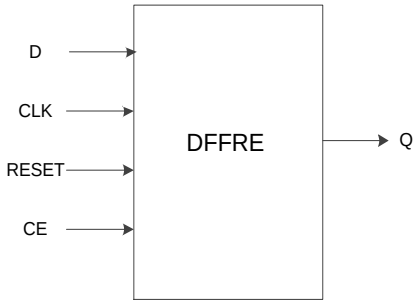
4.1.2 DFFRE

原语介绍

DFFRE(D Flip-Flop with Clock Enable and Synchronous Reset)是上升沿触发的 D 触发器，具有同步复位和时钟使能功能。

端口示意图

图 4-4 DFFRE 端口示意图



端口介绍

表 4-3 DFFRE 端口介绍

端口名	I/O	描述
D	Input	数据输入信号
CLK	Input	时钟输入信号
RESET	Input	同步复位信号，高电平有效
CE	Input	时钟使能信号
Q	Output	数据输出信号

参数介绍

表 4-4 DFFRE 参数介绍

参数名	取值范围	默认值	描述
INIT	1’b0	1’b0	DFFRE 初始值

原语例化

Verilog 例化：

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

Vhdl 例化：

```
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
        );
```

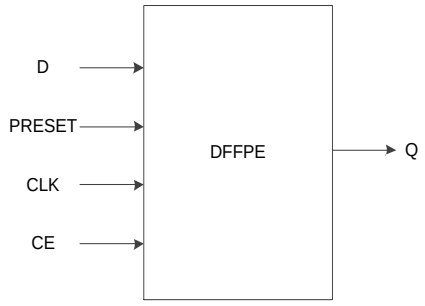
4.1.3 DFFPE

原语介绍

DFFPE(D Flip-Flop with Clock Enable and Asynchronous Preset)是上升沿触发的 D 触发器，具有异步置位和时钟使能功能。

端口示意图

图 4-5 DFFPE 端口示意图



端口介绍

表 4-5 DFFPE 端口介绍

端口名	I/O	描述
D	Input	数据输入信号

端口名	I/O	描述
CLK	Input	时钟输入信号
PRESET	Input	异步置位信号，高电平有效
CE	Input	时钟使能信号
Q	Output	数据输出信号

参数介绍

表 4-6 DFFPE 参数介绍

参数名	取值范围	默认值	描述
INIT	1’b1	1’b1	DFFPE 初始值

原语例化

Verilog 例化：

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

Vhdl 例化：

```
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
);
```

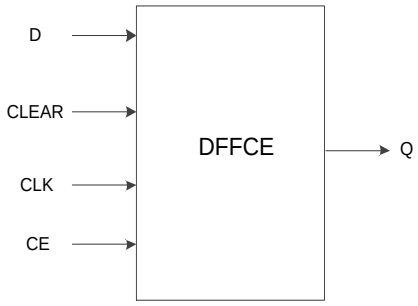
4.1.4 DFFCE

原语介绍

DFFCE(D Flip-Flop with Clock Enable and Asynchronous Clear)是上升沿触发的 D 触发器，具有异步复位和时钟使能功能。

端口示意图

图 4-6 DFFCE 端口示意图



端口介绍

表 4-7 DFFCE 端口介绍

端口名	I/O	描述
D	Input	数据输入信号
CLK	Input	时钟输入信号
CLEAR	Input	异步清零信号，高电平有效
CE	Input	时钟使能信号
Q	Output	数据输出信号

参数介绍

表 4-8 DFFCE 参数介绍

参数名	取值范围	默认值	描述
INIT	1’b0	1’b0	DFFCE 初始值

原语例化

Verilog 例化：

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

Vhdl 例化：

```
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
      );
```

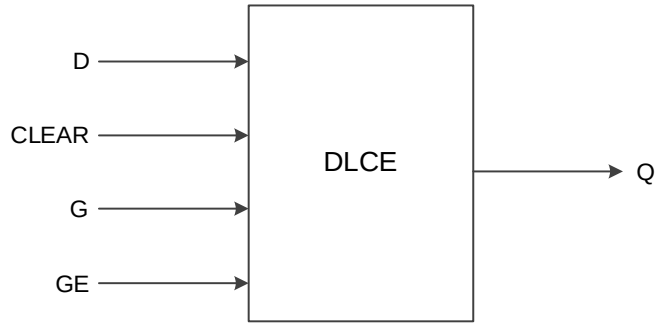
4.1.5 DLCE

原语介绍

DLCE(Data Latch with Asynchronous Clear and Latch Enable)是具有使能控制和复位功能的一种锁存器，控制信号 G 高电平有效。

端口示意图

图 4-7 DLCE 端口示意图



端口介绍

表 4-9 DLCE 端口介绍

端口名	I/O	描述
D	Input	数据输入信号
CLEAR	Input	异步清零信号，高电平有效
G	Input	数据控制信号，高电平有效
GE	Input	数据控制使能信号
Q	Output	数据输出信号

参数介绍

表 4-10 DLCE 参数介绍

参数名	取值范围	默认值	描述
INIT	1’b0	1’b0	DLCE 初始值

原语例化

Verilog 例化：

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

Vhdl 例化：

```
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
      );
```

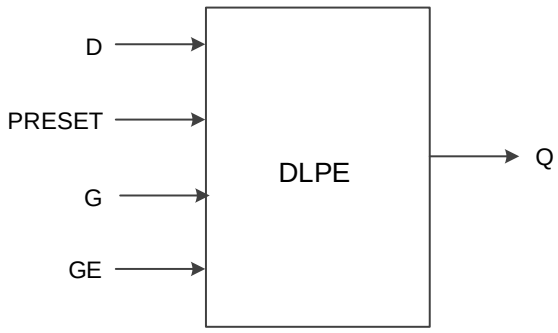
4.1.6 DLPE

原语介绍

DLPE(Data Latch with Asynchronous Preset and Latch Enable)是具有使能控制和置位功能的一种锁存器，控制信号 G 高电平有效。

端口示意图

图 4-8 DLPE 端口示意图



端口介绍

表 4-11 DLPE 端口介绍

端口名	I/O	描述
D	Input	数据输入信号
PRESET	Input	异步置位信号，高电平有效
G	Input	数据控制信号，高电平有效
GE	Input	数据控制使能信号
Q	Output	数据输出信号

参数介绍

表 4-12 DLPE 参数介绍

参数名	取值范围	默认值	描述
INIT	1’b1	1’b1	DLPE 初始值

原语例化

Verilog 例化：

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

Vhdl 例化：

```
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 模式输入逻辑

4.2.1 IDDR

原语介绍

IDDR(Dual Data Rate Input)，实现双倍数据速率输入。

功能描述

IDDR 模式，输出数据在同一时钟边沿提供给 FPGA 逻辑。IDDR 逻辑框图如图 4-9 所示，时序图如图 4-10 所示。

图 4-9 IDDR 逻辑框图

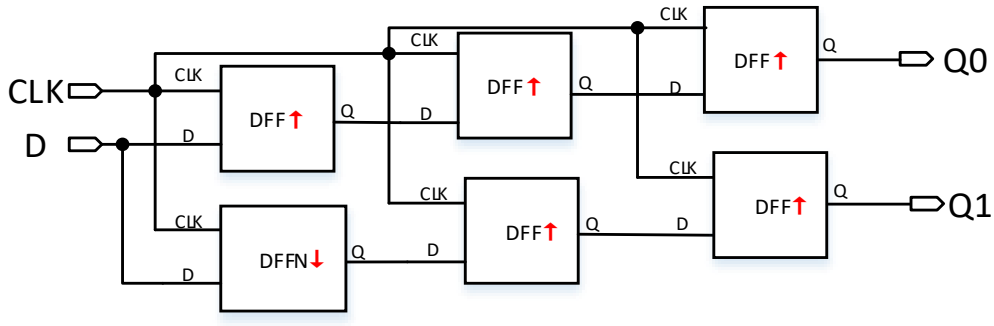
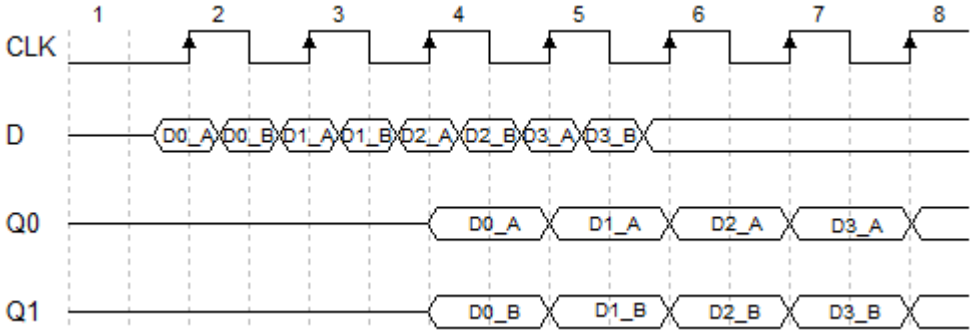


图 4-10 IDDR 时序图



端口示意图

图 4-11 IDDR 端口示意图

端口介绍

表 4-13 IDDR 端口介绍

端口名	I/O	描述
D	Input	IDDR 数据输入信号
CLK	Input	时钟输入信号
Q0, Q1	Output	IDDR 数据输出信号

连接规则

IDDR 的数据输入 D 可直接来自 IBUF，或经过 IODELAY 模块来自其输出 DO。

原语例化

可以直接实例化原语。

Verilog 例化：

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

Vhdl 例化：

```
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

原语介绍

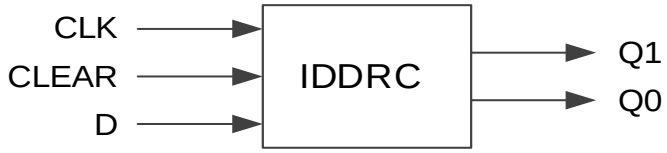
IDDRC(Dual Data Rate Input with Asynchronous Clear)与 IDDR 功能类似，实现双倍速率输入，同时具有异步复位功能。

功能描述

IDDRC 模式，输出数据在同一时钟边沿提供给 FPGA 逻辑。

端口示意图

图 4-12 IDDRC 端口示意图



端口介绍

表 4-14 IDDRC 端口介绍

端口名	I/O	描述
D	Input	IDDRC 数据输入信号
CLK	Input	时钟输入信号
CLEAR	Input	异步清零输入信号，高电平有效
Q0, Q1	Output	IDDRC 数据输出信号

连接规则

IDDRC 的数据输入 D 可直接来自 IBUF，或经过 IODELAY 模块来自其输出 DO。

原语例化

可以直接实例化原语。

Verilog 例化：

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

```
        .Q1(Q1),
        .D(D),
        .CLK(CLK),
        .CLEAR(CLEAR)
    );
```

Vhdl 例化：

```
    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

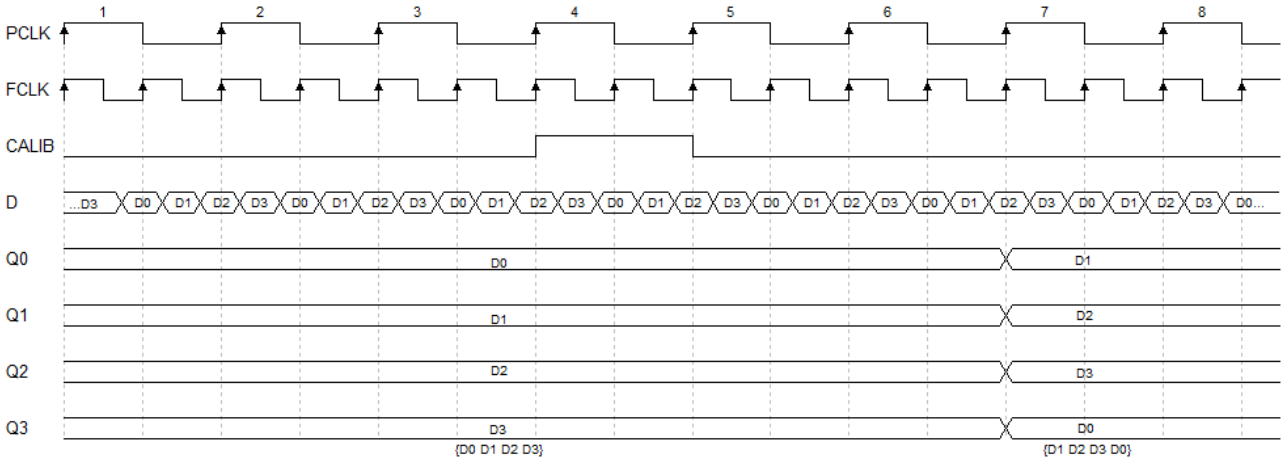
原语介绍

IDES4(1 to 4 Deserializer)为 1 位串行输入 4 位并行输出的解串器。

功能描述

IDES4 模式，实现 1：4 串并转换，输出数据在同一时钟边沿提供给 FPGA 逻辑。支持 CALIB 调整输出数据顺序，每个脉冲数据移位一位，移位四次后，数据输出将与移位前的数据相同。CALIB 示例时序图如图 4-13 所示。

图 4-13 CALIB 示例时序图



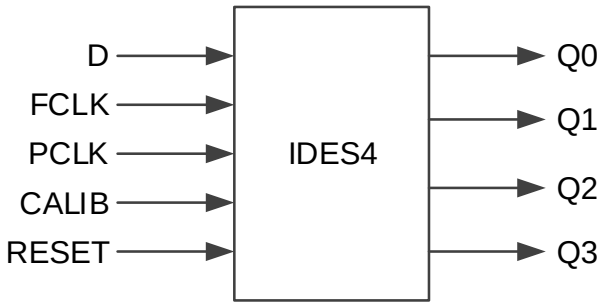
注！

示例中 CALIB 信号的脉冲宽度和时序仅供参考，可根据需要调整，其脉冲宽度大于等于 T_{PCLK} 即可。

PCLK 通常由 FCLK 分频获得：$f_{PCLK} = 1/2\, f_{FCLK}$ 。

端口示意图

图 4-14 IDES4 端口示意图



端口介绍

表 4-15 IDES4 端口介绍

端口名	I/O	描述
D	Input	IDES4 数据输入信号
FCLK	Input	高速时钟输入信号
PCLK	Input	主时钟输入信号
CALIB	Input	CALIB 信号，用于调整输出数据顺序，高电平有效。
RESET	Input	异步复位输入信号，高电平有效。
Q3~Q0	Output	IDES4 数据输出信号

连接规则

IDES4 的数据输入 D 可直接来自 IBUF，或经过 IODELAY 模块来自其输出 DO。

原语例化

可以直接实例化原语。

Verilog 例化：

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

Vhdl 例化：

```
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

原语介绍

IDES8(1 to 8 Deserializer)为 1 位串行输入 8 位并行输出的解串器。

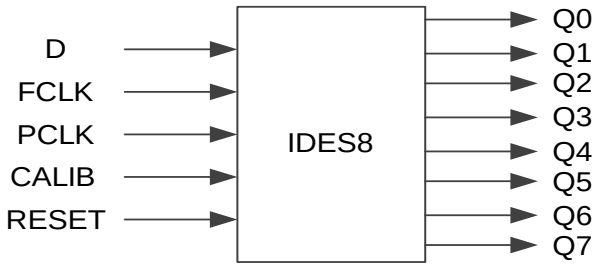
功能描述

IDES8 模式，实现 1：8 串并转换，输出数据在同一时钟边沿提供给 FPGA 逻辑。支持 CALIB 调整输出数据顺序，每个脉冲数据移位一位，移位八次后，数据输出将与移位前的数据相同。

PCLK 通常由 FCLK 分频获得：$f_{PCLK} = 1/4\, f_{FCLK}$ 。

端口示意图

图 4-15 IDES8 端口示意图



端口介绍

表 4-16 IDES8 端口介绍

端口名	I/O	描述
D	Input	IDES8 数据输入信号
FCLK	Input	高速时钟输入信号
PCLK	Input	主时钟输入信号
CALIB	Input	CALIB 信号输入信号，用于调整输出数据顺序，高电平有效
RESET	Input	异步复位输入信号，高电平有效
Q7~Q0	Output	IDES8 数据输出信号

连接规则

IDES8 的数据输入 D 可直接来自 IBUF，或经过 IODELAY 模块来自其输出 DO。

原语例化

可以直接实例化原语。

Verilog 例化：

```
IDES8 uut(
    .Q0(Q0),
    .Q1(Q1),
    .Q2(Q2),
     .Q3(Q3),
    .Q4(Q4),
    .Q5(Q5),
    .Q6(Q6),
    .Q7(Q7),
    .D(D),
    .FCLK(FCLK),
    .PCLK(PCLK),
    .CALIB(CALIB),
    .RESET(RESET)
);
```

Vhdl 例化：

```
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

原语介绍

IDES10(1 to 10 Deserializer)为 1 位串行输入 10 位并行输出的解串器。

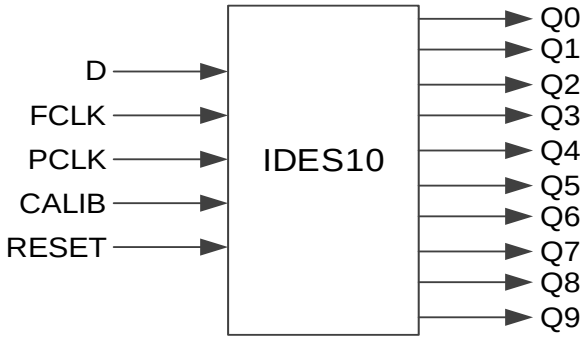
功能描述

IDES10 模式，实现 1：10 串并转换，输出数据在同一时钟边沿提供给 FPGA 逻辑。支持 CALIB 调整输出数据顺序，每个脉冲数据移位一位，移位十次后，数据输出将与移位前的数据相同。

PCLK 通常由 FCLK 分频获得：$f_{PCLK} = 1/5\, f_{FCLK}$。

端口示意图

图 4-16 IDES10 端口示意图



端口介绍

表 4-17 IDES10 端口介绍

端口名	I/O	描述
D	Input	IDES10 数据输入信号
FCLK	Input	高速时钟输入信号
PCLK	Input	主时钟输入信号
CALIB	Input	CALIB 信号，用于调整输出数据顺序，高电平有效。
RESET	Input	异步复位输入信号，高电平有效。
Q9~Q0	Output	IDES10 数据输出信号

连接规则

IDES10 的数据输入 D 可直接来自 IBUF，或经过 IODELAY 模块来自其输出 DO。

原语例化

可以直接实例化原语。

Verilog 例化：

```
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 例化：

```
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

原语介绍

IVIDEO(1 to 7 Deserializer)为 1 位串行输入 7 位并行输出的解串器。

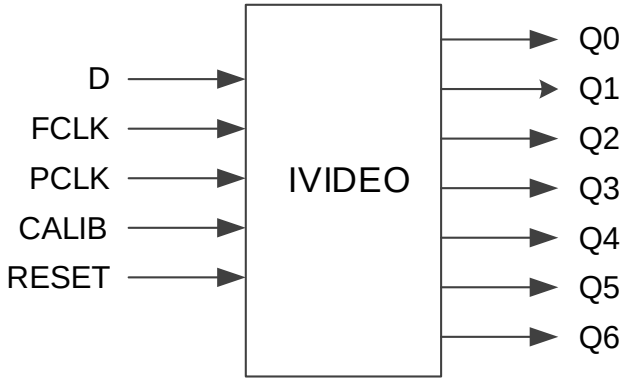
功能描述

IVIDEO 模式，实现 1：7 串并转换，输出数据在同一时钟边沿提供给 FPGA 逻辑。支持 CALIB 调整输出数据顺序，每个脉冲数据移位 2 位，移位七次后，数据输出将与移位前的数据相同。

PCLK 通常由 FCLK 分频获得：$f_{PCLK} = 1/3.5\, f_{FCLK}$ 。

端口示意图

图 4-17 IVIDEO 端口示意图



端口介绍

表 4-18 IVIDEO 端口介绍

端口名	I/O	描述
D	Input	IVIDEO 数据输入信号
FCLK	Input	高速时钟输入信号
PCLK	Input	主时钟输入信号

端口名	I/O	描述
CALIB	Input	CALIB 信号，用于调整输出数据顺序，高电平有效。
RESET	Input	异步复位输入信号，高电平有效。
Q6~Q0	Output	IVIDEO 数据输出信号

连接规则

IVIDEO 的数据输入 D 可直接来自 IBUF，或经过 IODELAY 模块来自其输出 DO。

原语例化

可以直接实例化原语。

Verilog 例化：

```
IVIDEO uut(
    .Q0(Q0),
    .Q1(Q1),
    .Q2(Q2),
     .Q3(Q3),
    .Q4(Q4),
    .Q5(Q5),
    .Q6(Q6),
    .D(D),
    .FCLK(FCLK),
    .PCLK(PCLK),
    .CALIB(CALIB),
    .RESET(RESET)
);
```

Vhdl 例化：

```
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

原语介绍

IDES16(1 to 16 Deserializer)为 1 位串行输入 16 位并行输出的解串器。

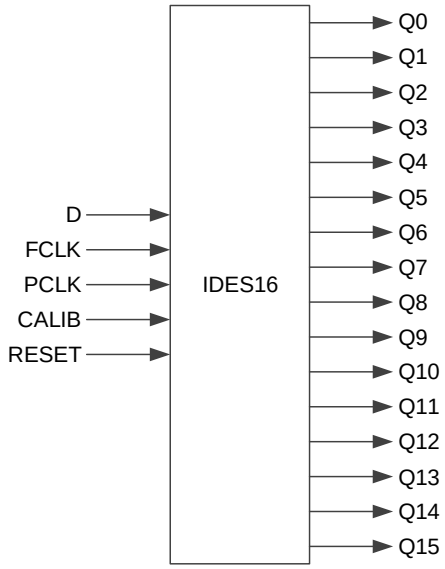
功能描述

IDES16 模式，实现 1：16 串并转换，输出数据在同一时钟边沿提供给 FPGA 逻辑。支持 CALIB 调整输出数据顺序，每个脉冲数据移位一位，移位十六次后，数据输出将与移位前的数据相同。

PCLK 通常由 FCLK 分频获得：$f_{PCLK} = 1/8\, f_{FCLK}$ 。

端口示意图

图 4-18 IDES16 端口示意图



端口介绍

表 4-19 IDES16 端口介绍

端口名	I/O	描述
D	Input	IDES16 数据输入信号
FCLK	Input	高速时钟输入信号
PCLK	Input	主时钟输入信号
CALIB	Input	CALIB 信号，用于调整输出数据顺序，高电平有效。
RESET	Input	异步复位输入信号，高电平有效。
Q15~Q0	Output	IDES16 数据输出信号

连接规则

IDES16 的数据输入 D 可直接来自 IBUF，或经过 IODELAY 模块来自其输出 DO。

原语例化

可以直接实例化原语。

Verilog 例化：

```
IDES16 uut(
    .Q0(Q0),
    .Q1(Q1),
    .Q2(Q2),
```

```
        .Q3(Q3),
        .Q4(Q4),
        .Q5(Q5),
        .Q6(Q6),
        .Q7(Q7),
        .Q8(Q8),
        .Q9(Q9),
        .Q10(Q10),
        .Q11(Q11),
        .Q12(Q12),
        .Q13(Q13),
        .Q14(Q14),
        .Q15(Q15),
       .D(D),
       .FCLK(FCLK),
       .PCLK(PCLK),
       .CALIB(CALIB),
       .RESET(RESET)
    );
```

Vhdl 例化：

```
    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

原语介绍

IDDR_MEM(Dual Data Rate Input with Memory)，实现带 memory 的双倍数据速率输入。

功能描述

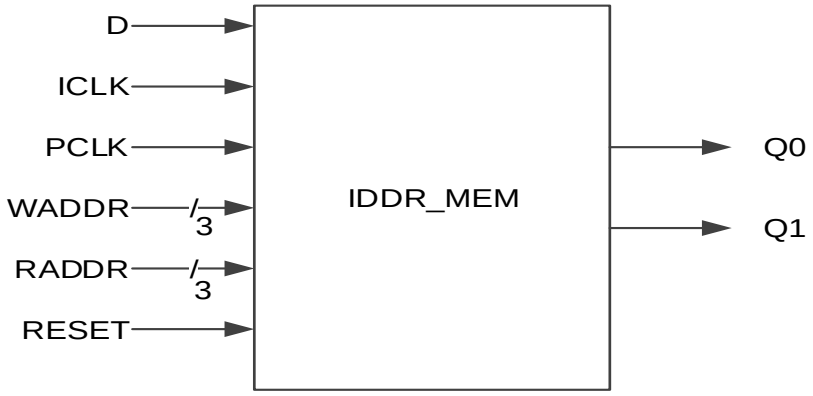
IDDR_MEM 输出数据在同一时钟边沿提供给 FPGA 逻辑。IDDR_MEM 需要配合 DQS 使用，其中，ICLK 连接 DQS 的输出信号 DQSR90，且根据 ICLK 的时钟沿将数据送入 IDDR_MEM；WADDR[2:0]连接 DQS 的输出信号 WPOINT；RADDR[2:0]连接 DQS 的输出信号 RPOINT。

PCLK 和 ICLK 的频率关系为：$f_{PCLK} = f_{ICLK}$。

PCLK 和 ICLK 之间存在一定的相位关系，可根据 DQS 的 DLLSTEP 值确定相位关系。

端口示意图

图 4-19 IDDR_MEM 端口示意图



端口介绍

表 4-20 IDDR_MEM 端口介绍

端口名	I/O	描述
D	Input	IDDR_MEM 数据输入信号
ICLK	Input	时钟输入信号，来自 DQS 模块的 DQSR90。
PCLK	Input	主时钟输入信号
WADDR[2:0]	Input	写地址信号，来自 DQS 模块的 WPOINT。
RADDR[2:0]	Input	读地址信号，来自 DQS 模块的 RPOINT。
RESET	Input	异步复位输入信号，高电平有效。
Q1~Q0	Output	IDDR_MEM 数据输出信号

连接规则

- IDDR_MEM 的数据输入 D 可直接来自 IBUF，或经过 IODELAY 模块来自其输出 DO；
- ICLK 需来自 DQS 模块的 DQSR90；
- WADDR[2:0]需来自 DQS 模块的 WPOINT；
- RADDR[2:0]需来自 DQS 模块的 RPOINT。

原语例化

Verilog 例化：

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

Vhdl 例化：

```
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

原语介绍

IDES4_MEM(1 to 4 Deserializer with Memory) 带存储功能的 1:4 串并转换器，可实现 1 位串行转 4 位并行。

功能描述

IDES4_MEM 实现 1: 4 串并转换，输出数据在同一时钟边沿提供给 FPGA 逻辑。支持 CALIB 调整输出数据顺序，每个脉冲数据移位一位，移位四次后，数据输出将与移位前的数据相同。

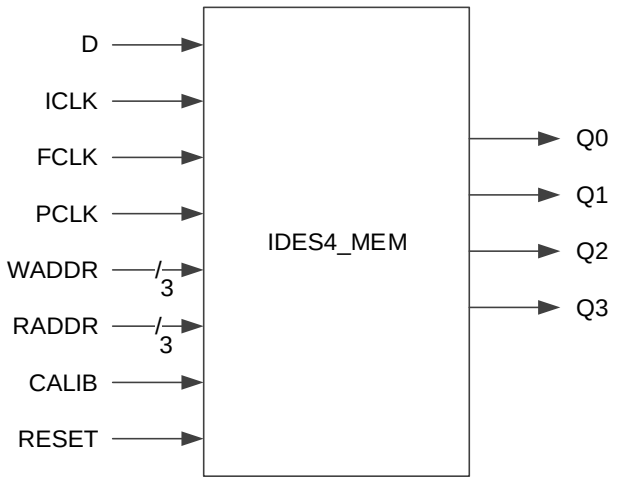
IDES4_MEM 与 IDES4 不同，IDES4_MEM 需要配合 DQS 使用，其中，ICLK 连接 DQS 的输出信号 DQSR90，且根据 ICLK 的时钟沿将数据送入 IDES4_MEM；WADDR[2:0]连接 DQS 的输出信号 WPOINT；RADDR[2:0]连接 DQS 的输出信号 RPOINT。

PCLK、FCLK 和 ICLK 的频率关系为：$f_{PCLK} = 1/2\ f_{FCLK} = 1/2\ f_{ICLK}$ 。

FCLK 和 ICLK 之间存在一定的相位关系，可根据 DQS 的 DLLSTEP 值确定相位关系。

端口示意图

图 4-20 IDES4_MEM 端口示意图



端口介绍

表 4-21 IDES4_MEM 端口介绍

端口名	I/O	描述
D	Input	IDES4_MEM 数据输入信号
ICLK	Input	时钟输入信号，来自 DQS 模块的 DQSR90。
FCLK	Input	高速时钟输入信号
PCLK	Input	主时钟输入信号
WADDR[2:0]	Input	写地址信号，来自 DQS 模块的 WPOINT。
RADDR[2:0]	Input	读地址信号，来自 DQS 模块的 RPOINT。
CALIB	Input	CALIB 信号，用于调整输出数据顺序，高电平有效。
RESET	Input	异步复位输入信号，高电平有效。
Q3~Q0	Output	IDES4_MEM 数据输出信号

连接规则

- IDES4_MEM 的数据输入 D 可直接来自 IBUF，或经过 IODELAY 模块来自其输出 DO；
- ICLK 需来自 DQS 模块的 DQSR90；
- WADDR[2:0]需来自 DQS 模块的 WPOINT；
- RADDR[2:0]需来自 DQS 模块的 RPOINT。

原语例化

Verilog 例化：

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

Vhdl 例化：

```
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

原语介绍

IDES8_MEM (1 to 8 Deserializer with Memory) 带存储功能的 1:8 串并

转换器，可实现 1 位串行转 8 位并行。

功能描述

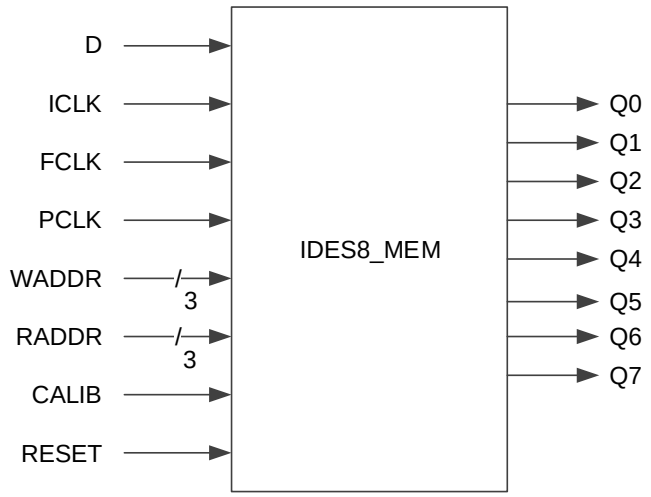
IDES8_MEM 实现 1: 8 串并转换，输出数据在同一时钟边沿提供给 FPGA 逻辑。支持 CALIB 调整输出数据顺序，每个脉冲数据移位一位，移位八次后，数据输出将与移位前的数据相同。与 IDES8 不同，IDES8_MEM 需要配合 DQS 使用，其中，ICLK 连接 DQS 的输出信号 DQSR90，且根据 ICLK 的时钟沿将数据送入 IDES8_MEM；WADDR[2:0] 连接 DQS 的输出信号 WPOINT；RADDR[2:0]连接 DQS 的输出信号 RPOINT。

PCLK、FCLK 和 ICLK 的频率关系为：$f_{PCLK} = 1/4\, f_{FCLK} = 1/4\, f_{ICLK}$ 。

FCLK 和 ICLK 之间存在一定的相位关系，可根据 DQS 的 DLLSTEP 值确定相位关系。

端口示意图

图 4-21 IDES8_MEM 端口示意图



端口介绍

表 4-22 IDES8_MEM 端口介绍

端口名	I/O	描述
D	Input	IDES8_MEM 数据输入信号
ICLK	Input	时钟输入信号，来自 DQS 模块的 DQSR90。
FCLK	Input	高速时钟输入信号
PCLK	Input	主时钟输入信号
WADDR[2:0]	Input	写地址信号，来自 DQS 模块的 WPOINT。
RADDR[2:0]	Input	读地址信号，来自 DQS 模块的 RPOINT。
CALIB	Input	CALIB 信号，用于调整输出数据顺序，高电平有效。
RESET	Input	异步复位输入信号，高电平有效。

端口名	I/O	描述
Q7~Q0	Output	IDES8_MEM 数据输出信号

连接规则

- IDES8_MEM 的数据输入 D 可直接来自 IBUF，或经过 IODELAY 模块来自其输出 DO；
- ICLK 需来自 DQS 模块的 DQSR90；
- WADDR[2:0]需来自 DQS 模块的 WPOINT；
- RADDR[2:0]需来自 DQS 模块的 RPOINT。

原语例化

Verilog 例化：

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

Vhdl 例化：

```
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

原语介绍

IDES14(1 to 14 Deserializer)为 1 位串行输入 14 位并行输出的解串器。

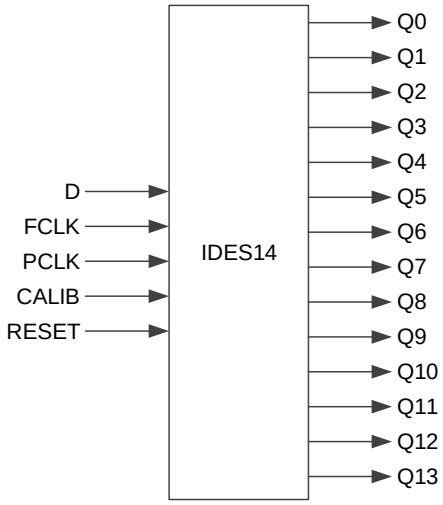
功能描述

IDES14 模式，实现 1：14 串并转换，输出数据在同一时钟边沿提供给 FPGA 逻辑。支持 CALIB 调整输出数据顺序，每个脉冲数据移位一位，移位十四次后，数据输出将与移位前的数据相同。

PCLK 通常由 FCLK 分频获得：$f_{PCLK} = 1/7\ f_{FCLK}$。

端口示意图

图 4-22 IDES14 端口示意图



端口介绍

表 4-23 IDES14 端口介绍

端口名	I/O	描述
D	Input	IDES14 数据输入信号
FCLK	Input	高速时钟输入信号
PCLK	Input	主时钟输入信号
CALIB	Input	CALIB 信号，用于调整输出数据顺序，高电平有效
RESET	Input	异步复位输入信号，高电平有效
Q13~Q0	Output	IDES14 数据输出信号

连接规则

IDES14 的数据输入 D 可直接来自 IBUF，或经过 IODELAY 模块来自其输出 DO。

原语例化

可以直接实例化原语。

Verilog 例化：

```
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 例化：

```
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

原语介绍

IDES32(1 to 32 Deserializer)为 1 位串行输入 32 位并行输出的解串器。

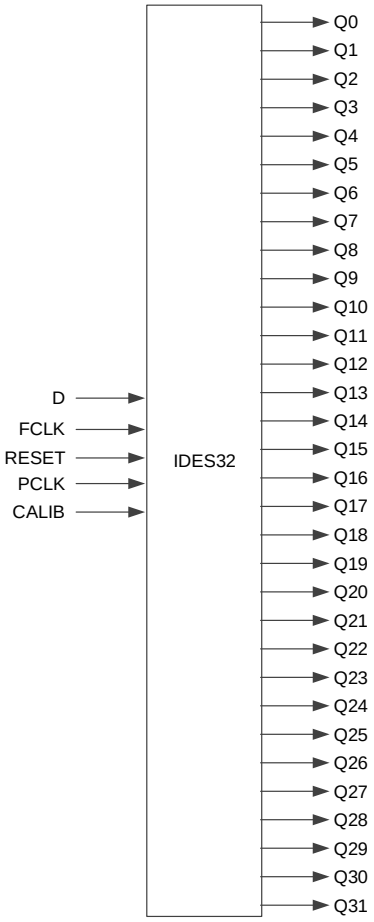
功能描述

IDES32 模式，实现 1：32 串并转换，输出数据在同一时钟边沿提供给 FPGA 逻辑。支持 CALIB 调整输出数据顺序，每个脉冲数据移位一位，移位三十二次后，数据输出将与移位前的数据相同。

PCLK 通常由 FCLK 分频获得：$f_{PCLK} = 1/16\ f_{FCLK}$。

端口示意图

图 4-23 IDES32 端口示意图



端口介绍

表 4-24 IDES32 端口介绍

端口名	I/O	描述
D	Input	IDES32 数据输入信号
FCLK	Input	高速时钟输入信号

端口名	I/O	描述
PCLK	Input	主时钟输入信号
CALIB	Input	CALIB 信号，用于调整输出数据顺序，高电平有效。
RESET	Input	异步复位输入信号，高电平有效。
Q31~Q0	Output	IDES32 数据输出信号

连接规则

IDES32 的数据输入 D 可直接来自 IBUF，或经过 IODELAY 模块来自其输出 DO。

原语例化

可以直接实例化原语。

Verilog 例化：

```
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 例化：

```
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

原语介绍

OSIDES32 实现 1 位串行输入 32 位并行输出的过采样串转并功能。

功能描述

使用两个 IOL 可以实现 1：32 的过采样串转并，支持异步复位功能。但是不支持 CALIB 调整输出数据顺序功能。

过采样与普通串转并模式的最大区别在于：过采样在同一个 fclk 周期进行 4 次采样，普通串转并在一个 fclk 周期只进行 fclk 上升沿和下降沿两次采样。

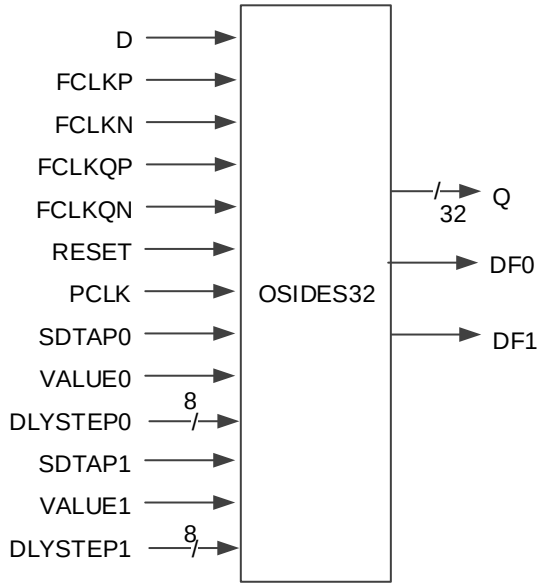
PCLK 和 FCLKP、FCLKN、FCLKQP、FCLKQN 的频率关系为：

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

四个 FCLK 相位关系为 P 0°、QP 90°、 N 180°、 QN 270°。

端口示意图

图 4-24 OSIDES32 端口示意图



端口介绍

表 4-25 OSIDES32 端口介绍

端口名	I/O	描述
D	Input	OSIDES32 数据输入信号
FCLKP	Input	高速时钟输入信号
FCLKN	Input	高速时钟输入信号
FCLKQP	Input	高速时钟输入信号
FCLKQN	Input	高速时钟输入信号
PCLK	Input	主时钟输入信号
RESET	Input	异步复位输入信号，高电平有效
SDTAP0	Input	IODELAY_0：控制加载静态延时步长 0：加载静态延时

端口名	I/O	描述
		1：动态调整延时
VALUE0	Input	IODELAY_0：VALUE 为下降沿时动态调整延时值，每个脉冲移动一个延时步长
DLYSTEP0[7:0]	Input	IODELAY_0：动态延时值
SDTAP1	Input	IODELAY_1：控制加载静态延时步长 0：加载静态延时 1：动态调整延时
VALUE1	Input	IODELAY_1：VALUE 为下降沿时动态调整延时值，每个脉冲移动一个延时步长
DLYSTEP1[7:0]	Input	IODELAY_1：动态延时值
Q[31:0]	Output	OSIDES32 数据输出信号
DF0	Output	IODELAY_0 输出标志位
DF1	Output	IODELAY_1 输出标志位

参数介绍

表 4-26 OSIDES32 参数介绍

参数名	取值范围	默认值	描述
C_STATIC_DLY_0	0~255	0	IODELAY_0 静态延时步长控制
DYN_DLY_EN_0	“FALSE”/”TRUE”	“FALSE”	IODELAY_0 动态模式使能控制
ADAPT_EN_0	“FALSE”/”TRUE”	“FALSE”	IODELAY_0 自适应模式使能控制
C_STATIC_DLY_1	0~255	0	IODELAY_1 静态延时步长控制
DYN_DLY_EN_1	“FALSE”/”TRUE”	“FALSE”	IODELAY_1 动态模式使能控制
ADAPT_EN_1	“FALSE”/”TRUE”	“FALSE”	IODELAY_1 自适应模式使能控制

连接规则

OSIDES32 的数据输入 D 直接来自 IBUF。

原语例化

可以直接实例化原语。

Verilog 例化：

```
OSIDES32 uut(
    .Q(Q),
    .D(D),
```

```
    .FCLKP(FCLKP),
    .FCLKN(FCLKN),
    .FCLKQP(FCLKQP),
    .FCLKQN(FCLKQN),
    .PCLK(PCLK),
    .RESET(RESET),
    .SDTAP0(SDTAP0),
    .VALUE0(VALUE0),
    .DLYSTEP0(DLYSTEP0),
    .SDTAP1(SDTAP1),
    .VALUE1(VALUE1),
    .DLYSTEP1(DLYSTEP1),
    .DF0(DF0),
    .DF1(DF1)
);
defparam uut.C_STATIC_DLY_0=0;
defparam uut.DYN_DLY_EN_0=”FALSE”;
defparam uut.ADAPT_EN_0=”FALSE”;
defparam uut.C_STATIC_DLY_1=0;
defparam uut.DYN_DLY_EN_1=”FALSE”;
defparam uut.ADAPT_EN_1=”FALSE”;
```

Vhdl 例化：

```
COMPONENT OSIDES32(
                    C_STATIC_DLY_0:integer:=0;
                    DYN_DLY_EN_0:string:=”FALSE”;
                    ADAPT_EN_0:string:=”FALSE”;
                    C_STATIC_DLY_1:integer:=0;
                    DYN_DLY_EN_1:string:=”FALSE”;
                    ADAPT_EN_1:string:=”FALSE”
);
    PORT(
        Q:OUT std_logic_vector(31 downto 0);
        D:IN std_logic;
        FCLKP:IN std_logic;
```

```
        FCLKN:IN std_logic;
        FCLKQP:IN std_logic;
        FCLKQN:IN std_logic;
        PCLK:IN std_logic;
        SDTAP0, SDTAP1:IN std_logic;
        VALUE0, VALUE1:IN std_logic;
        RESET:IN std_logic;
        DLYSTEP0, DLYSTEP1: IN std_logic_vector (7 downto 0);
        DF0, DF1:OUT std_logic
        );
END COMPONENT;
uut:OSIDES32
    GENERIC MAP (C_STATIC_DLY_0=>0,
                 DYN_DLY_EN_0=>”FALSE”,
                 ADAPT_EN_0=>”FALSE”,
                 C_STATIC_DLY_1=>0,
                 DYN_DLY_EN_1=>”FALSE”,
                 ADAPT_EN_1=>”FALSE”
        )
    PORT MAP (
        Q=>Q,
        D=>D,
        FCLKP=>FCLKP,
        FCLKN=>FCLKN,
        FCLKQP=>FCLKQP,
        FCLKQN=>FCLKQN,
        PCLK=>PCLK,
        SDTAP0=>SDTAP0,
        VALUE0=>VALUE0,
        SDTAP1=>SDTAP1,
        VALUE1=>VALUE1,
        RESET=>RESET,
        DLYSTEP0=>DLYSTEP0,
        DLYSTEP1=>DLYSTEP1,
        DF0=>DF0,
```

```
    DF1=>DF1
);
```

4.2.14 OSIDES64

原语介绍

OSIDES64 实现 1 位串行输入 64 位并行输出的过采样串转并功能。

功能描述

使用两个 IOL 可以实现 1：64 的过采样串转并，支持异步复位功能。但是不支持 CALIB 调整输出数据顺序功能。

过采样与普通串转并模式的最大区别在于：过采样在同一个 fclk 周期进行 4 次采样，普通串转并在一个 fclk 周期只进行 fclk 上升沿和下降沿两次采样。

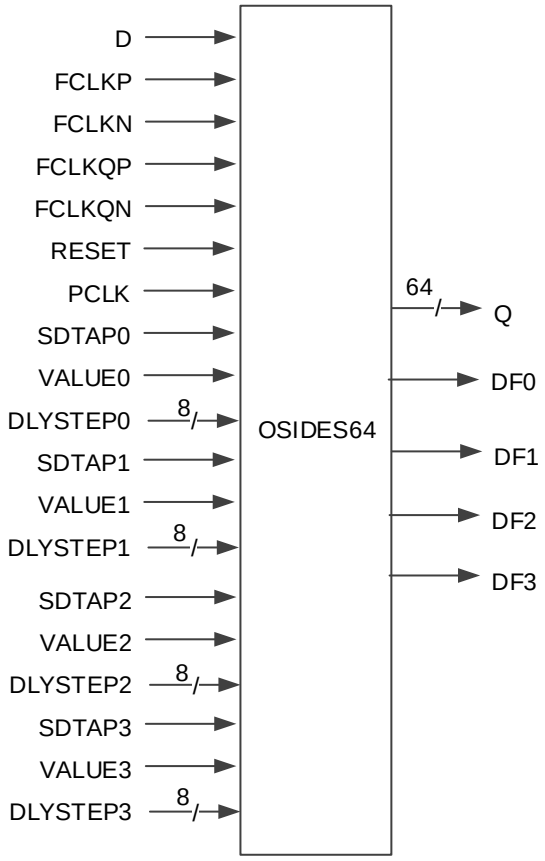
PCLK 和 FCLKP、FCLKN、FCLKQP、FCLKQN 的频率关系为：

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

四个 FCLK 相位关系为 P 0°、QP 90°、 N 180°、 QN 270°。

端口示意图

图 4-25 OSIDES64 端口示意图



端口介绍

表 4-27 OSIDES64 端口介绍

端口名	I/O	描述
D	Input	OSIDES64 数据输入信号
FCLKP	Input	高速时钟输入信号
FCLKN	Input	高速时钟输入信号
FCLKQP	Input	高速时钟输入信号
FCLKQN	Input	高速时钟输入信号
PCLK	Input	主时钟输入信号
RESET	Input	异步复位输入信号，高电平有效
SDTAP0	Input	IODELAY_0：控制加载静态延时步长 0：加载静态延时 1：动态调整延时
VALUE0	Input	IODELAY_0：VALUE 为下降沿时动态调整延时值，每个脉冲移动一个延时步长
DLYSTEP0[7:0]	Input	IODELAY_0：动态延时值
SDTAP1	Input	IODELAY_1：控制加载静态延时步长 0：加载静态延时 1：动态调整延时
VALUE1	Input	IODELAY_1：VALUE 为下降沿时动态调整延时值，每个脉冲移动一个延时步长
DLYSTEP1[7:0]	Input	IODELAY_1：动态延时值
SDTAP0	Input	IODELAY_2：控制加载静态延时步长 0：加载静态延时 1：动态调整延时
VALUE0	Input	IODELAY_2：VALUE 为下降沿时动态调整延时值，每个脉冲移动一个延时步长
DLYSTEP0[7:0]	Input	IODELAY_2：动态延时值
SDTAP1	Input	IODELAY_3：控制加载静态延时步长 0：加载静态延时 1：动态调整延时
VALUE1	Input	IODELAY_3：VALUE 为下降沿时动态调整延时值，每个脉冲移动一个延时步长
DLYSTEP1[7:0]	Input	IODELAY_3：动态延时值
Q[31:0]	Output	OSIDES64 数据输出信号
DF0	Output	IODELAY_0 输出标志位
DF1	Output	IODELAY_1 输出标志位
DF2	Output	IODELAY_2 输出标志位
DF3	Output	IODELAY_3 输出标志位

参数介绍

表 4-28 OSIDES64 参数介绍

参数名	取值范围	默认值	描述
C_STATIC_DLY_0	0~255	0	IODELAY_0 静态延时步长控制
DYN_DLY_EN_0	“FALSE”/”TRUE”	“FALSE”	IODELAY_0 动态模式使能控制
ADAPT_EN_0	“FALSE”/”TRUE”	“FALSE”	IODELAY_0 自适应模式使能控制
C_STATIC_DLY_1	0~255	0	IODELAY_1 静态延时步长控制
DYN_DLY_EN_1	“FALSE”/”TRUE”	“FALSE”	IODELAY_1 动态模式使能控制
ADAPT_EN_1	“FALSE”/”TRUE”	“FALSE”	IODELAY_1 自适应模式使能控制
C_STATIC_DLY_2	0~255	0	IODELAY_2 静态延时步长控制
DYN_DLY_EN_2	“FALSE”/”TRUE”	“FALSE”	IODELAY2 动态模式使能控制
ADAPT_EN_2	“FALSE”/”TRUE”	“FALSE”	IODELAY_2 自适应模式使能控制
C_STATIC_DLY_3	0~255	0	IODELAY_3 静态延时步长控制
DYN_DLY_EN_3	“FALSE”/”TRUE”	“FALSE”	IODELAY_3 动态模式使能控制
ADAPT_EN_3	“FALSE”/”TRUE”	“FALSE”	IODELAY_3 自适应模式使能控制

连接规则

OSIDES64 的数据输入 D 直接来自 IBUF。

原语例化

可以直接实例化原语。

Verilog 例化：

```
OSIDES64 uut(
    .Q(Q),
    .D(D),
    .FCLKP(FCLKP),
    .FCLKN(FCLKN),
    .FCLKQP(FCLKQP),
```

```
    .FCLKQN(FCLKQN),
    .PCLK(PCLK),
    .RESET(RESET),
    .SDTAP0(SDTAP0),
    .VALUE0(VALUE0),
    .DLYSTEP0(DLYSTEP0),
    .SDTAP1(SDTAP1),
    .VALUE1(VALUE1),
    .DLYSTEP1(DLYSTEP1),
    .SDTAP2(SDTAP2),
    .VALUE2(VALUE2),
    .DLYSTEP2(DLYSTEP2),
    .SDTAP3(SDTAP3),
    .VALUE3(VALUE3),
    .DLYSTEP3(DLYSTEP3),
    .DF0(DF0),
    .DF1(DF1),
    .DF2(DF2),
    .DF3(DF3)
);
defparam uut.C_STATIC_DLY_0=0;
defparam uut.DYN_DLY_EN_0=”FALSE”;
defparam uut.ADAPT_EN_0=”FALSE”;
defparam uut.C_STATIC_DLY_1=0;
defparam uut.DYN_DLY_EN_1=”FALSE”;
defparam uut.ADAPT_EN_1=”FALSE”;
defparam uut.C_STATIC_DLY_2=0;
defparam uut.DYN_DLY_EN_2=”FALSE”;
defparam uut.ADAPT_EN_2=”FALSE”;
defparam uut.C_STATIC_DLY_3=0;
defparam uut.DYN_DLY_EN_3=”FALSE”;
defparam uut.ADAPT_EN_3=”FALSE”;
```

Vhdl 例化：

```
COMPONENT OSIDES64(
```

```
                    C_STATIC_DLY_0:integer:=0;
                    DYN_DLY_EN_0:string:=”FALSE”;
                    ADAPT_EN_0:string:=”FALSE”;
                    C_STATIC_DLY_1:integer:=0;
                    DYN_DLY_EN_1:string:=”FALSE”;
                    ADAPT_EN_1:string:=”FALSE”;
                    C_STATIC_DLY_2:integer:=0;
                    DYN_DLY_EN_2:string:=”FALSE”;
                    ADAPT_EN_2:string:=”FALSE”;
                    C_STATIC_DLY_3:integer:=0;
                    DYN_DLY_EN_3:string:=”FALSE”;
                    ADAPT_EN_3:string:=”FALSE”
    );
        PORT(
            Q:OUT std_logic_vector(31 downto 0);
            D:IN std_logic;
            FCLKP:IN std_logic;
            FCLKN:IN std_logic;
            FCLKQP:IN std_logic;
            FCLKQN:IN std_logic;
            PCLK:IN std_logic;
            SDTAP0, SDTAP1, SDTAP2, SDTAP3:IN std_logic;
            VALUE0, VALUE1, VALUE2, VALUE3:IN std_logic;
            RESET:IN std_logic;
            DLYSTEP0, DLYSTEP1, DLYSTEP2, DLYSTEP3: IN
std_logic_vector (7 downto 0);
            DF0, DF1:OUT std_logic
            );
    END COMPONENT;
    uut:OSIDES64
        GENERIC MAP (C_STATIC_DLY_0=>0,
                    DYN_DLY_EN_0=>”FALSE”,
                    ADAPT_EN_0=>”FALSE”,
                    C_STATIC_DLY_1=>0,
                    DYN_DLY_EN_1=>”FALSE”,
```

```
                ADAPT_EN_1=>”FALSE”,
                C_STATIC_DLY_2=>0,
                DYN_DLY_EN_2=>”FALSE”,
                ADAPT_EN_2=>”FALSE”,
                C_STATIC_DLY_3=>0,
                DYN_DLY_EN_3=>”FALSE”,
                ADAPT_EN_3=>”FALSE”
    )
PORT MAP (
    Q=>Q,
    D=>D,
    FCLKP=>FCLKP,
    FCLKN=>FCLKN,
    FCLKQP=>FCLKQP,
    FCLKQN=>FCLKQN,
    PCLK=>PCLK,
    SDTAP0=>SDTAP0,
    VALUE0=>VALUE0,
    SDTAP1=>SDTAP1,
    VALUE1=>VALUE1,
    SDTAP2=>SDTAP2,
    VALUE2=>VALUE2,
    SDTAP3=>SDTAP3,
    VALUE3=>VALUE3,
    RESET=>RESET,
    DLYSTEP0=>DLYSTEP0,
    DLYSTEP1=>DLYSTEP1,
    DLYSTEP2=>DLYSTEP2,
    DLYSTEP3=>DLYSTEP3,
    DF0=>DF0,
    DF1=>DF1,
    DF2=>DF2,
    DF3=>DF3
    );
```

4.3 DDR 模式输出逻辑

4.3.1 ODDR

原语介绍

ODDR(Dual Data Rate Output)，实现双倍数据速率输出。

功能描述

ODDR 模式，用于从 FPGA 器件传输双倍数据速率信号。其中 Q0 为双倍速率数据输出，Q1 用于 Q0 所连的 IOBUF/TBUF 的 OEN 信号。ODDR 逻辑框图如图 4-26 所示，时序图如图 4-27 所示。

图 4-26 ODDR 逻辑框图

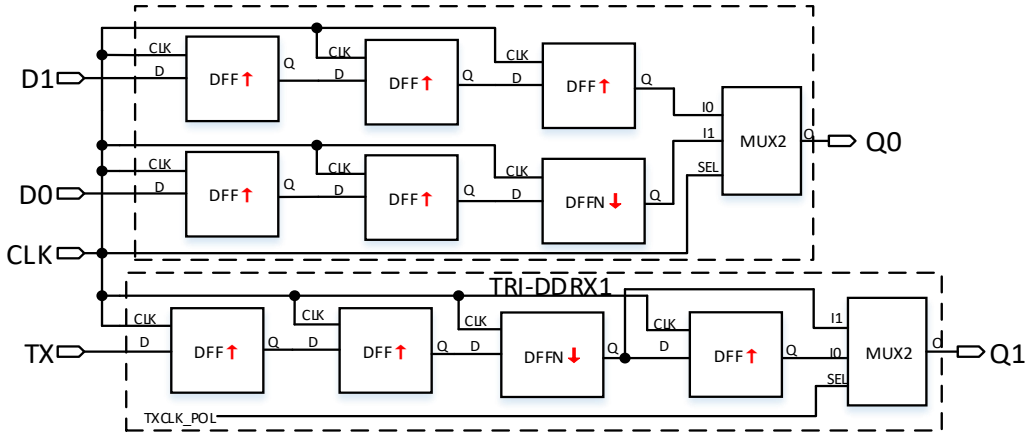
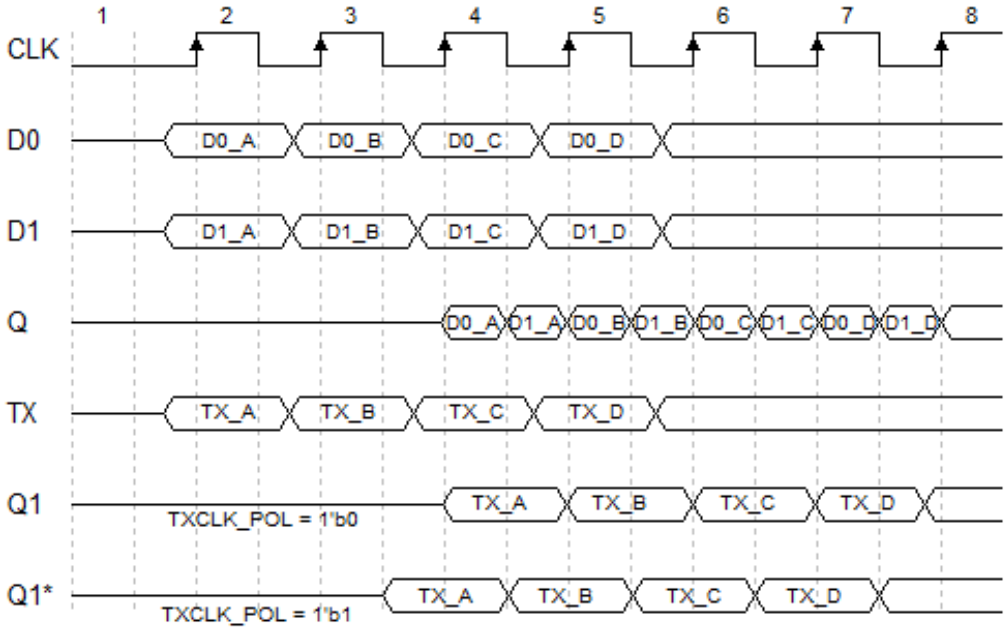


图 4-27 ODDR 时序图



端口示意图

图 4-28 ODDR 端口示意图

端口介绍

表 4-29 ODDR 端口介绍

端口名	I/O	描述
D0, D1	Input	ODDR 数据输入信号
TX	Input	通过 TRI-DDRX1 产生 Q1
CLK	Input	时钟输入信号
Q0	Output	ODDR 数据输出信号
Q1	Output	ODDR 三态使能控制输出信号，可连接 Q0 所连的 IOBUF/TBUF 的 OEN 信号，或悬空。

参数介绍

表 4-30 ODDR 参数介绍

参数名	取值范围	默认值	描述
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 输出时钟极性控制 ● 1'b0：Q1 上升沿输出 ● 1'b1：Q1 下降沿输出

连接规则

- Q0 可直接连接 OBUF/IOBUF/TBUF，或经过 IODELAY 模块连接其输入端口 DI；
- Q1 需连接 Q0 所连的 IOBUF/TBUF 的 OEN 信号，或悬空。

原语例化

可以直接实例化原语。

Verilog 例化：

```
ODDR uut(
    .Q0(Q0),
    .Q1(Q1),
    .D0(D0),
    .D1(D1),
```

```
    .TX(TX),
    .CLK(CLK)
);
defparam uut.TXCLK_POL=1'b0;
```

Vhdl 例化：

```
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

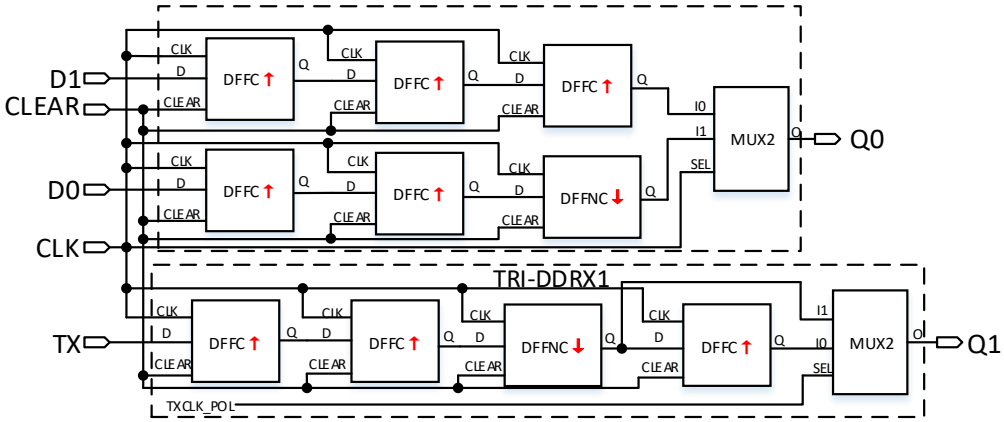
原语介绍

ODDRC(Dual Data Rate Output with Asynchronous Clear)与 ODDR 功能类似，实现双倍速率输出，同时具有异步复位功能。

功能描述

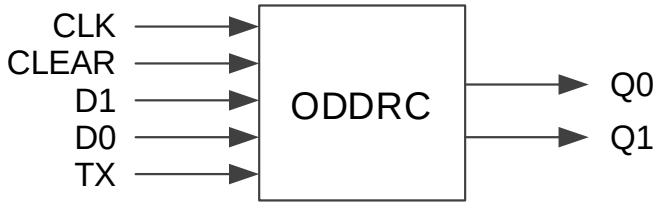
ODDRC 模式，用于从 FPGA 器件传输双倍数据速率信号。其中 Q0 为双倍速率数据输出，Q1 用于 Q0 所连的 IOBUF/TBUF 的 OEN 信号。其逻辑框图如图 4-29 所示。

图 4-29 ODDRC 逻辑框图



端口示意图

图 4-30 ODDRC 端口示意图



端口介绍

表 4-31 ODDRC 端口介绍

端口名	I/O	描述
D0, D1	Input	ODDRC 数据输入信号
TX	Input	通过 TRI-DDRX1 产生输出 Q1
CLK	Input	时钟输入信号
CLEAR	Input	异步清零输入信号，高电平有效
Q0	Output	ODDRC 数据输出信号
Q1	Output	ODDRC 三态使能控制输出信号，可连接 Q0 所连的 IOBUF/TBUF 的 OEN 信号，或悬空。

参数介绍

表 4-32 ODDRC 参数介绍

参数名	取值范围	默认值	描述
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 输出时钟极性控制 ● 1'b0：Q1 上升沿输出 ● 1'b1：Q1 下降沿输出

连接规则

- Q0 可直接连接 OBUF/IOBUF/TBUF，或经过 IODELAY 模块连接其输入端口 DI；
- Q1 需连接 Q0 所连的 IOBUF/TBUF 的 OEN 信号，或悬空。

原语例化

可以直接实例化原语。

Verilog 例化：

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

Vhdl 例化：

```
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

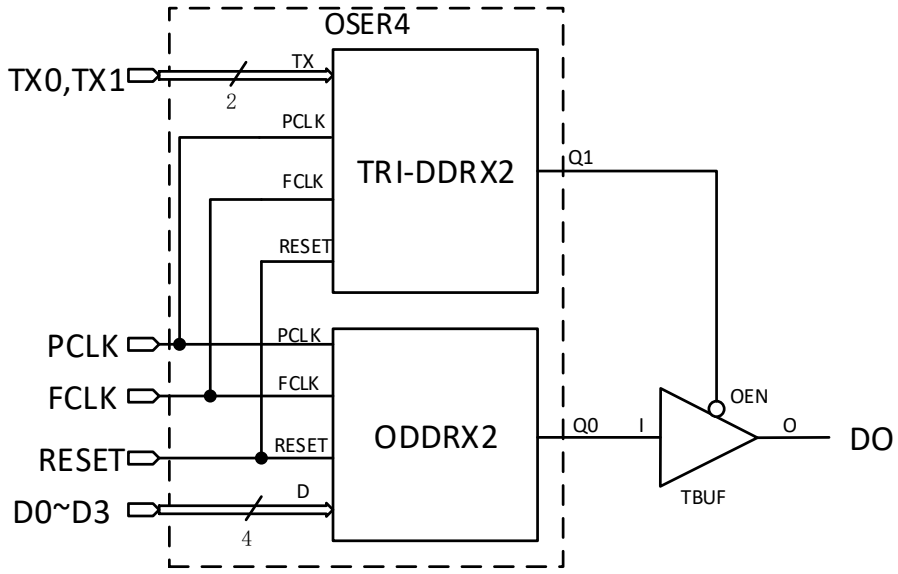
原语介绍

OSER4 (4 to 1 Serializer)为 4 位并行输入 1 位串行输出的串化器。

功能描述

OSER4 模式，实现 4：1 并串转换。其中 Q0 为 OSER4 数据串行输出，Q1 用于 Q0 所连的 IOBUF/TBUF 的 OEN 信号。TX0/TX1 为 IOBUF/TBUF 的 OEN 输入控制信号，可以同步和数据 D0~D3 一起经过 DDR。TX0/TX1 经过 TRI-DDRX2 输出为 Q1 连接 IOBUF/TBUF 的 OEN 信号，D0~D3 经过 ODDRX2 输出为 Q0 连接 IOBUF/TBUF 的数据输入 I，输出顺序依次为 D0，D1，D2，D3。逻辑框图如图 4-31 所示。

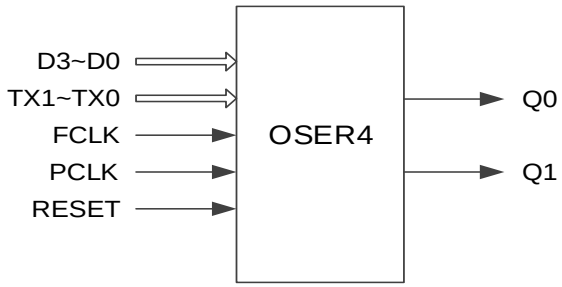
图 4-31 OSER4 逻辑框图



PCLK 通常由 FCLK 分频而获得：$f_{PCLK} = 1/2\, f_{FCLK}$。

端口示意图

图 4-32 OSER4 端口示意图



端口介绍

表 4-33 OSER4 端口介绍

端口名	I/O	描述
D3~D0	Input	OSER4 数据输入信号
TX1~TX0	Input	通过 TRI-DDRX2 产生 Q1
FCLK	Input	高速时钟输入信号
PCLK	Input	主时钟输入信号
RESET	Input	异步复位输入信号，高电平有效。
Q0	Output	OSER4 数据输出信号
Q1	Output	OSER4 三态使能控制输出信号，可连接 Q0 所连的 IOBUF/TBUF 的 OEN 信号，或悬空。

参数介绍

表 4-34 OSER4 参数介绍

参数名	取值范围	默认值	描述
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 输出时钟极性控制 ● 1'b0：数据上升沿输出 ● 1'b1：数据下降沿输出
HWL	"false", "true"	"false"	OSER4 数据 d_up0/1 时序关系控制 ● "false"：d_up1 比 d_up0 提前一个周期 ● "true"：d_up1 和 d_up0 时序相同

连接规则

- Q0 可直接连接 OBUF/IOBUF/TBUF，或经过 IODELAY 模块连接其输入端口 DI；
- Q1 需连接 Q0 所连的 IOBUF/TBUF 的 OEN 信号，或悬空。

原语例化

可以直接实例化原语。

Verilog 例化：

```
OSER4 uut(
    .Q0(Q0),
    .Q1(Q1),
    .D0(D0),
    .D1(D1),
    .D2(D2),
    .D3(D3),
    .TX0(TX0),
    .TX1(TX1),
    .PCLK(PCLK),
    .FCLK(FCLK),
    .RESET(RESET)
);
defparam uut.HWL ="false";
defparam uut.TXCLK_POL =1'b0;
```

Vhdl 例化：

```
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

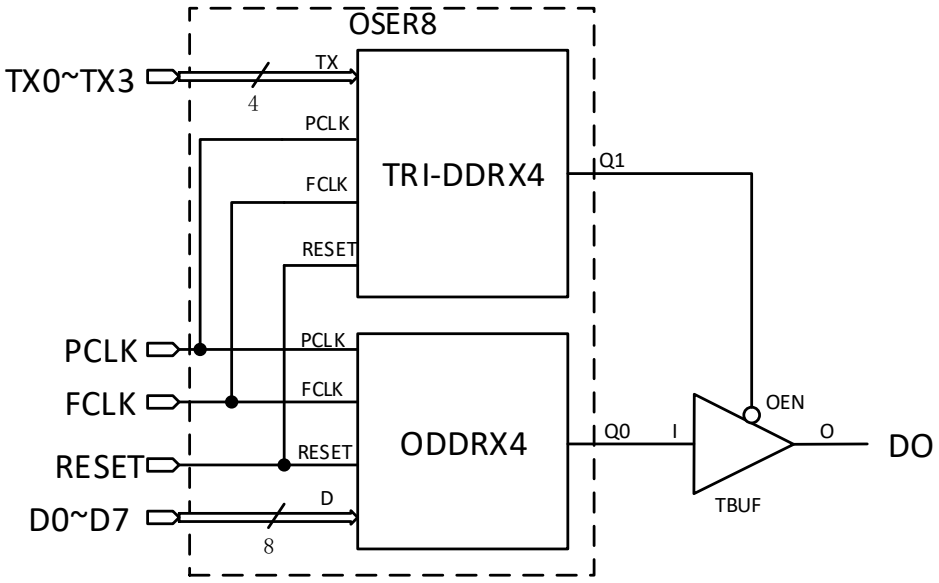
原语介绍

OSER8(8 to 1 Serializer)为 8 位并行输入 1 位串行输出的串化器。

功能描述

OSER8 模式，实现 8:1 并串转换。其中 Q0 为 OSER8 数据串行输出，输出顺序依次为 D0，D1，D2，D3，D4，D5，D6，D7。Q1 用于 Q0 所连的 IOBUF/TBUF 的 OEN 信号。逻辑框图如图 4-33 所示。

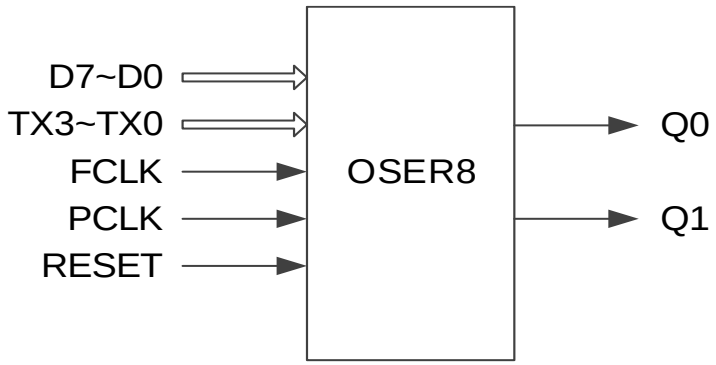
图 4-33 OSER8 逻辑框图



PCLK 通常由 FCLK 分频而得来：$f_{PCLK} = 1/4\, f_{FCLK}$。

端口示意图

图 4-34 OSER8 端口示意图



端口介绍

表 4-35 OSER8 端口介绍

端口名	I/O	描述
D7~D0	Input	OSER8 数据输入信号
TX3~TX0	Input	通过 TRI-DDRX4 产生 Q1
FCLK	Input	高速时钟输入信号
PCLK	Input	主时钟输入信号
RESET	Input	异步复位输入信号，高电平有效
Q0	Output	OSER8 数据输出信号
Q1	Output	OSER8 三态使能控制输出信号，可连接 Q0 所连的 IOBUF/TBUF 的 OEN 信号，或悬空。

参数介绍

表 4-36 OSER8 参数介绍

参数名	取值范围	默认值	描述
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 输出时钟极性控制 ● 1'b0：数据上升沿输出 ● 1'b1：数据下降沿输出
HWL	"false", "true"	"false"	OSER8 数据 d_up0/1 时序关系控制 ● "false": d_up1 比 d_up0 提前一个周期； ● "true": d_up1 和 d_up0 时序相同

连接规则

- Q0 可直接连接 OBUF/IOBUF/TBUF，或经过 IODELAY 模块连接其输入端口 DI；
- Q1 需连接 Q0 所连的 IOBUF/TBUF 的 OEN 信号，或悬空。

原语例化

可以直接实例化原语。

Verilog 例化：

```
OSER8 uut(
    .Q0(Q0),
    .Q1(Q1),
    .D0(D0),
    .D1(D1),
```

```
    .D2(D2),
    .D3(D3),
    .D4(D4),
    .D5(D5),
    .D6(D6),
    .D7(D7),
    .TX0(TX0),
    .TX1(TX1),
    .TX2(TX2),
    .TX3(TX3),
    .PCLK(PCLK),
    .FCLK(FCLK),
    .RESET(RESET)
  );
  defparam uut.HWL ="false";
  defparam uut.TXCLK_POL =1'b0;
```

Vhdl 例化：

```
  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

原语介绍

OSER10(10 to 1 Serializer)为 10 位并行输入 1 位串行输出的串化器。

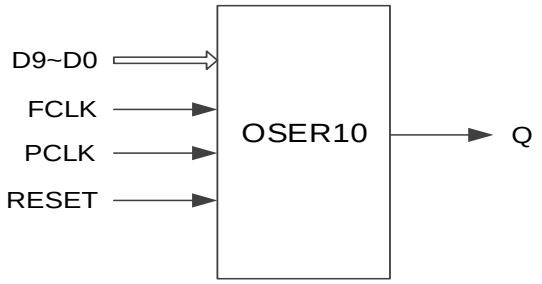
功能描述

OSER10 模式，实现 10:1 并串转换。OSER10 输出顺序依次为 D0，D1，D2，D3，D4，D5，D6，D7，D8，D9。

PCLK 通常由 FCLK 分频获得，$f_{PCLK} = 1/5\, f_{FCLK}$ 。

端口示意图

图 4-35 OSER10 端口示意图



端口介绍

表 4-37 OSER10 端口介绍

端口名	I/O	描述
D9~D0	Input	OSER10 数据输入信号
FCLK	Input	高速时钟输入信号
PCLK	Input	主时钟输入信号
RESET	Input	异步复位输入信号，高电平有效。
Q	Output	OSER10 数据输出信号

连接规则

Q 可直接连接 OBUF，或经过 IODELAY 模块连接其输入端口 DI。

原语例化

可以直接实例化原语。

Verilog 例化：

```
OSER10 uut(
    .Q(Q),
    .D0(D0),
    .D1(D1),
    .D2(D2),
     .D3(D3),
    .D4(D4),
    .D5(D5),
```

```
    .D6(D6),
    .D7(D7),
    .D8(D8),
    .D9(D9),
    .PCLK(PCLK),
    .FCLK(FCLK),
    .RESET(RESET)
);
```

Vhdl 例化：

```
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

原语介绍

OVIDEO(7 to 1 Serializer)为 7 位并行输入 1 位串行输出的串化器。

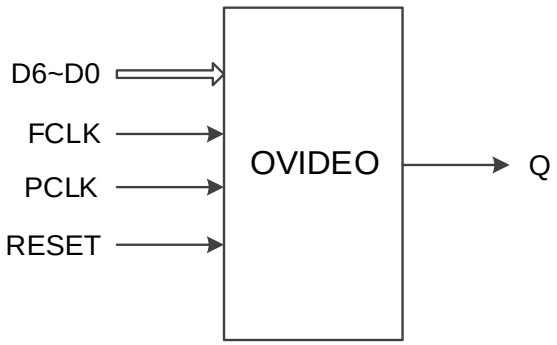
功能描述

OVIDEO 模式，实现 7:1 并串转换。OVIDEO 输出顺序依次为 D0，D1，D2，D3，D4，D5，D6。

PCLK 通常由 FCLK 分频获得：$f_{PCLK} = 1/3.5\, f_{FCLK}$。

端口示意图

图 4-36 OVIDEO 端口示意图



端口介绍

表 4-38 OVIDEO 端口介绍

端口名	I/O	描述
D6~D0	Input	OVIDEO 数据输入信号
FCLK	Input	高速时钟输入信号
PCLK	Input	主时钟输入信号
RESET	Input	异步复位输入信号，高电平有效。
Q	Output	OVIDEO 数据输出信号

连接规则

Q 可直接连接 OBUF，或经过 IODELAY 模块连接其输入端口 DI。

原语例化

可以直接实例化原语。

Verilog 例化：

```
OVIDEO uut(
    .Q(Q),
    .D0(D0),
    .D1(D1),
    .D2(D2),
     .D3(D3),
    .D4(D4),
    .D5(D5),
    .D6(D6),
    .PCLK(PCLK),
    .FCLK(FCLK),
    .RESET(RESET)
);
```

Vhdl 例化：

```
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

原语介绍

OSER16(16 to 1 Serializer)为 16 位并行输入 1 位串行输出的串化器。

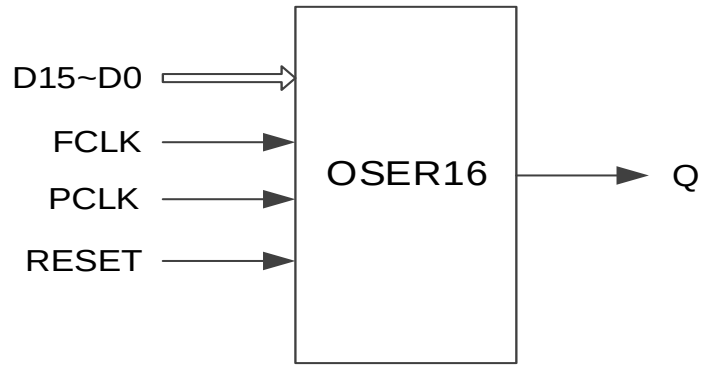
功能描述

OSER16 模式，实现 16:1 并串转换。OSER16 输出顺序依次为 D0，D1，D2，D3，D4，D5，D6，D7，D8，D9，D10，D11，D12，D13，D14，D15。

PCLK 通常由 FCLK 分频获得：$f_{PCLK} = 1/8\, f_{FCLK}$。

端口示意图

图 4-37 OSER16 端口示意图



端口介绍

表 4-39 OSER16 端口介绍

端口名	I/O	描述
D15~D0	Input	OSER16 数据输入信号
FCLK	Input	高速时钟输入信号
PCLK	Input	主时钟输入信号
RESET	Input	异步复位输入信号，高电平有效。
Q	Output	OSER16 数据输出信号

连接规则

Q 可直接连接 OBUF，或经过 IODELAY 模块连接其输入端口 DI。

原语例化

可以直接实例化原语。

Verilog 例化：

```
OSER16 uut(
    .Q(Q),
    .D0(D0),
    .D1(D1),
    .D2(D2),
    .D3(D3),
    .D4(D4),
    .D5(D5),
    .D6(D6),
    .D7(D7),
    .D8(D8),
    .D9(D9),
    .D10(D10),
    .D11(D11),
    .D12(D12),
    .D13(D13),
    .D14(D14),
    .D15(D15),
    .PCLK(PCLK),
    .FCLK(FCLK),
    .RESET(RESET)
```

```
);
```

Vhdl 例化：

```
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

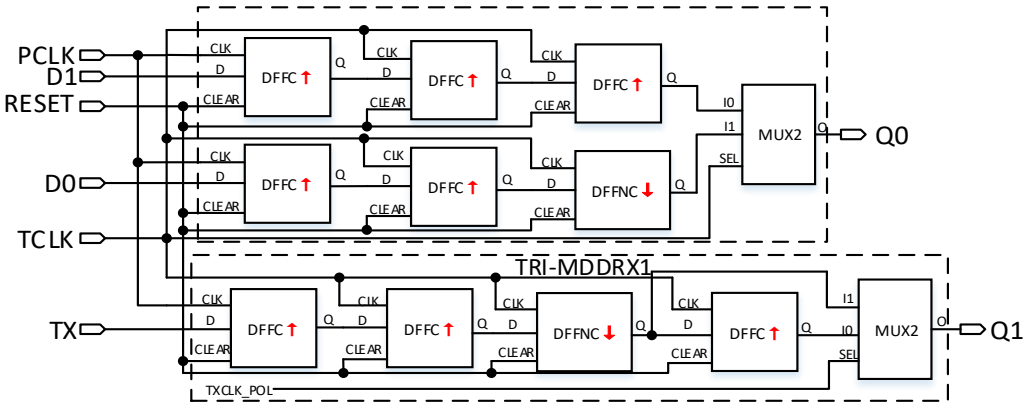
原语介绍

ODDR_MEM(Dual Data Rate Output with Memory)，实现带 memory 的双倍数据速率输出。

功能描述

ODDR_MEM 模式，从 FPGA 器件传输双倍数据速率信号。与 ODDR 不同，ODDR_MEM 需要配合 DQS 使用，TCLK 连接 DQS 的输出信号 DQSW0 或 DQSW270，且根据 TCLK 的时钟沿将数据从 ODDR_MEM 输出。ODDR_MEM 的 Q0 为双倍速率数据输出，Q1 用于 Q0 所连的 IOBUF/TBUF 的 OEN 信号。其逻辑框图如图 4-38 所示。

图 4-38 ODDR_MEM 逻辑框图

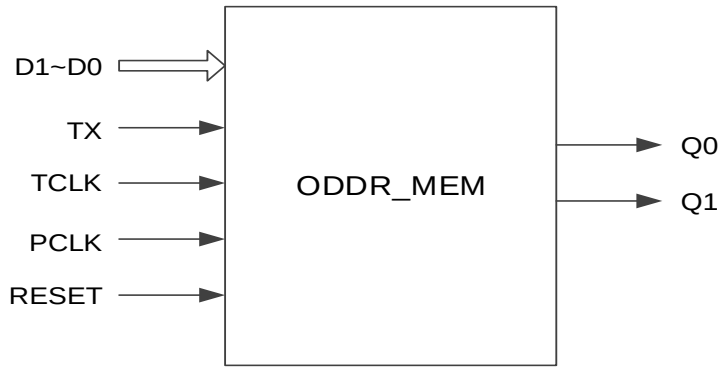


PCLK 和 TCLK 的频率关系为：$f_{PCLK} = f_{TCLK}$ 。

PCLK 和 TCLK 之间存在一定的相位关系，可根据 DQS 的 DLLSTEP 值和 WSTEP 值确定该相位关系。

端口示意图

图 4-39 ODDR_MEM 端口示意图



端口介绍

表 4-40 ODDR_MEM 端口介绍

端口名	I/O	描述
D1~D0	Input	ODDR_MEM 数据输入信号
TX	Input	通过 TRI-MDDRX1 产生 Q1
TCLK	Input	时钟输入信号，来自 DQS 模块的 DQSW0 或 DQSW270
PCLK	Input	主时钟输入信号
RESET	Input	异步复位输入信号，高电平有效。
Q0	Output	ODDR_MEM 数据输出信号
Q1	Output	ODDR_MEM 三态使能控制输出信号，可连接 Q0 所连的 IOBUF/TBUF 的 OEN 信号，或悬空。

参数介绍

表 4-41 ODDR_MEM 参数介绍

参数名	取值范围	默认值	描述
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 输出时钟极性控制 ● 1'b0：数据上升沿输出； ● 1'b1：数据下降沿输出
TCLK_SOURCE	"DQSW", "DQSW270"	"DQSW"	TCLK 来源选择 ● "DQSW": 来自 DQS 模块的 DQSW0 ● DQSW270": 来自 DQS 模块的 DQSW270

连接规则

- Q0 可直接连接 OBUF/IOBUF/TBUF，或经过 IODELAY 模块连接其输入端口 DI；
- Q1 需连接 Q0 所连的 IOBUF/TBUF 的 OEN 信号，或悬空；
- TCLK 需来自 DQS 模块的 DQSW0 或 DQSW270，并配置对应的参数。

原语例化

Verilog 例化：

```
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 例化：

```
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

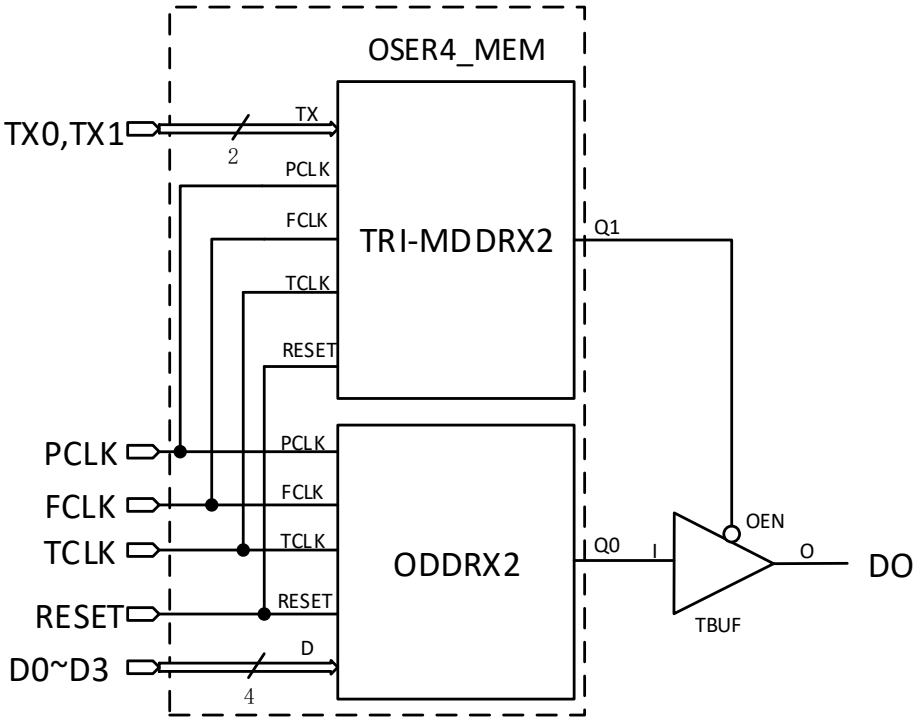
原语介绍

OSER4_MEM(4 to 1 Serializer with Memory) 带存储功能的 4:1 并串转换器，可实现 4 位并行转 1 位串行。

功能描述

OSER4_MEM 模式，实现 4：1 并串转换。与 OSER4 不同，OSER4_MEM 需要配合 DQS 使用，TCLK 连接 DQS 的输出信号 DQSW0 或 DQSW270，且根据 TCLK 的时钟沿将数据从 OSER4_MEM 输出。OSER4_MEM 的 Q0 为数据串行输出，输出顺序依次为 D0，D1，D2，D3。Q1 用于 Q0 所连的 IOBUF/TBUF 的 OEN 信号。其逻辑框图如图 4-40 所示。

图 4-40 OSER4_MEM 逻辑框图

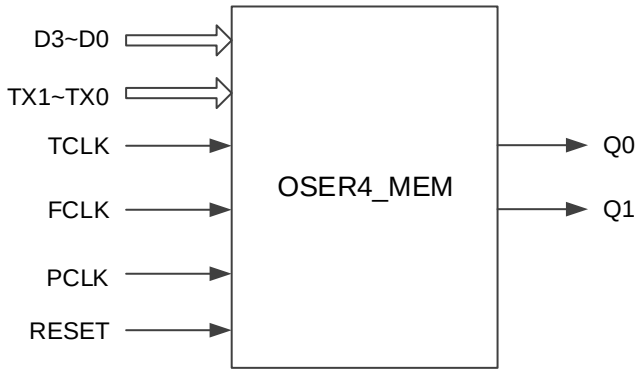


PCLK、FCLK 和 TCLK 的频率关系为：$f_{PCLK}=1/2\,f_{FCLK}=1/2\,f_{TCLK}$。

FCLK 和 TCLK 之间存在一定的相位关系，可根据 DQS 的 DLLSTEP 值和 WSTEP 值确定该相位关系。

端口示意图

图 4-41 OSER4_MEM 端口示意图



端口介绍

表 4-42 OSER4_MEM 端口介绍

端口名	I/O	描述
D3~D0	Input	OSER4_MEM 数据输入信号

端口名	I/O	描述
TX1~TX0	Input	通过 TRI-MDDRX2 产生 Q1
TCLK	Input	时钟输入信号，来自 DQS 模块的 DQSW0 或 DQSW270。
FCLK	Input	高速时钟输入信号
PCLK	Input	主时钟输入信号
RESET	Input	异步复位输入信号，高电平有效。
Q0	Output	OSER4_MEM 数据输出信号
Q1	Output	OSER4_MEM 三态使能控制输出信号，可连接 Q0 所连的 IOBUF/TBUF 的 OEN 信号，或悬空。

参数介绍

表 4-43 OSER4_MEM 参数介绍

参数名	取值范围	默认值	描述
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 输出时钟极性控制 ● 1'b0:数据上升沿输出； ● 1'b1:数据下降沿输出
TCLK_SOURCE	"DQSW","DQSW270"	" DQSW "	TCLK 来源选择 ● "DQSW": 来自 DQS 模块的 DQSW0； ● "DQSW270": 来自 DQS 模块的 DQSW270
HWL	"false", "true"	"false"	OSER4_MEM 数据 d_up0/1 时序关系控制 ● "false": d_up1 比 d_up0 提前一个周期； ● "true": d_up1 和 d_up0 时序相同

连接规则

- Q0 可直接连接 OBUF/IOBUF/TBUF，或经过 IODELAY 模块连接其输入端口 DI；
- Q1 需连接 Q0 所连的 IOBUF/TBUF 的 OEN 信号，或悬空；
- TCLK 需来自 DQS 模块的 DQSW0 或 DQSW270，并配置对应的参数。

原语例化

Verilog 例化：

```
OSER4_MEM oser4_mem_inst(
        .Q0(q0),
        .Q1(q1),
        .D0(d0),
        .D1(d1),
        .D2(d2),
        .D3(d3),
        .TX0(tx0),
        .TX1(tx1),
        .TCLK(tclk),
        .FCLK(fclk),
        .PCLK(pclk),
        .RESET(reset)
    );
    defparam uut.HWL ="false";
    defparam uut.TCLK_SOURCE ="DQSW";
    defparam uut.TXCLK_POL=1'b0;
```

Vhdl 例化：

```
    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

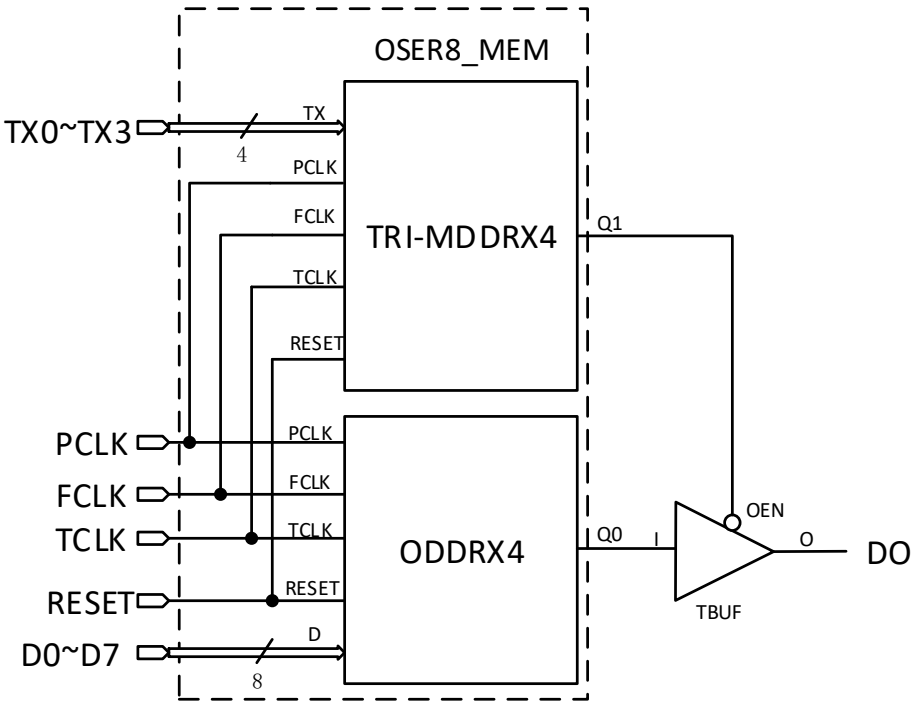
原语介绍

OSER8_MEM(8 to 1 Serializer with Memory) 带存储功能的 8:1 并串转换器，可实现 8 位并行转 1 位串行。

功能描述

OSER8_MEM 模式，实现 8:1 并串转换。与 OSER8 不同，OSER8_MEM 需要配合 DQS 使用，TCLK 连接 DQS 的输出信号 DQSW0 或 DQSW270，且根据 TCLK 的时钟沿将数据从 OSER8_MEM 输出。OSER8_MEM 的 Q0 为数据串行输出，输出顺序依次为 D0，D1，D2，D3，D4，D5，D6，D7。Q1 用于 Q0 所连的 IOBUF/TBUF 的 OEN 信号。其逻辑框图如图 4-42 所示。

图 4-42 OSER8_MEM 逻辑框图

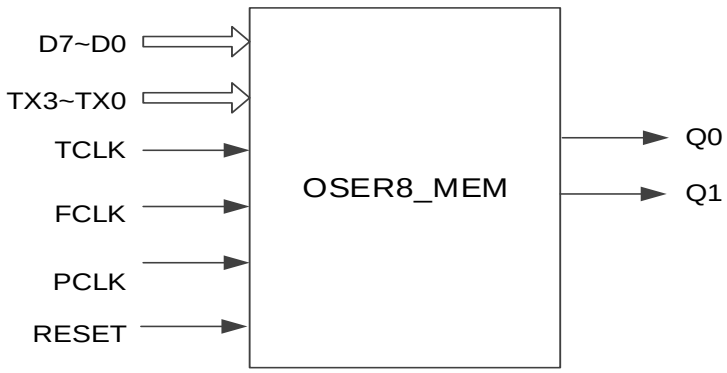


PCLK、FCLK 和 TCLK 的频率关系为：$f_{PCLK} = 1/4\, f_{FCLK} = 1/4\, f_{TCLK}$ 。

FCLK 和 TCLK 之间存在一定的相位关系，可根据 DQS 的 DLLSTEP 值和 WSTEP 值确定相位关系。

端口示意图

图 4-43 OSER8_MEM 端口示意图



端口介绍

表 4-44 OSER8_MEM 端口介绍

端口名	I/O	描述
D7~D0	Input	OSER8_MEM 数据输入信号
TX3~TX0	Input	通过 TRI-MDDRX4 产生 Q1

端口名	I/O	描述
TCLK	Input	时钟输入信号，来自 DQS 模块的 DQSW0 或 DQSW270。
FCLK	Input	高速时钟输入信号
PCLK	Input	主时钟输入信号
RESET	Input	异步复位输入信号，高电平有效。
Q0	Output	OSER8_MEM 数据输出信号
Q1	Output	OSER8_MEM 三态使能控制输出信号，可连接 Q0 所连的 IOBUF/TBUF 的 OEN 信号，或悬空。

参数介绍

表 4-45 OSER8_MEM 参数介绍

参数名	取值范围	默认值	描述
TXCLK_POL	1'b0, 1'b1	1'b0	Q1 输出时钟极性控制 ● 1'b0：数据上升沿输出 ● 1'b1：数据下降沿输出
TCLK_SOURCE	"DQSW","DQSW270"	" DQSW "	TCLK 来源选择 ● "DQSW": 来自 DQS 模块的 DQSW0； ● DQSW270": 来自 DQS 模块的 DQSW270
HWL	"false", "true"	"false"	OSER8_MEM 数据 d_up0/1 时序关系控制 ● "false": d_up1 比 d_up0 提前一个周期； ● "true": d_up1 和 d_up0 时序相同

连接规则

- Q0 可直接连接 OBUF/IOBUF/TBUF，或经过 IODELAY 模块连接其输入端口 DI；
- Q1 需连接 Q0 所连的 IOBUF/TBUF 的 OEN 信号，或悬空；
- TCLK 需来自 DQS 模块的 DQSW0 或 DQSW270，并配置对应的参数。

原语例化

Verilog 例化：

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

Vhdl 例化：

```
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 延时模块

4.4.1 IODELAY

原语介绍

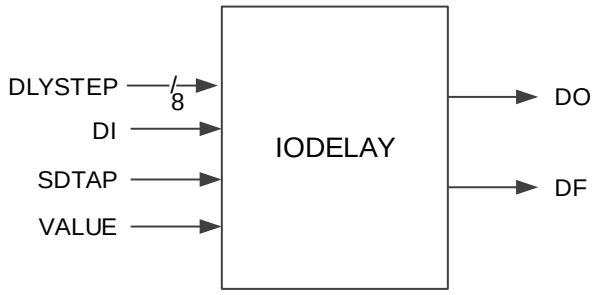
IODELAY(Input/Output delay)输入输出延时，是 IO 模块所包含的一个可编程延时单元。

功能描述

Arora V 器件支持 IODELAY 模块，总共提供 256（1~256）种延迟配置，单步的延迟时间约为 12.5ps。IODELAY 可用于 IO 逻辑和普通逻辑的输入输出延时。延时 delay code 有三种模式进行设置：静态模式、动态模式、自适应模式。

端口示意图

图 4-44 IODELAY 端口示意图



端口介绍

表 4-46 IODELAY 端口介绍

端口名	I/O	描述
DI	Input	数据输入信号
SDTAP	Input	控制加载延时步长 ● 0：加载静态延时/动态延时 ● 1： 自适应调整延时
VALUE	Input	● VALUE 为上升沿时动态调整延时值，每个脉冲移动一个延时步长

端口名	I/O	描述
DLYSTEP[7:0]	Input	动态延时值
DO	Output	数据输出信号
DF	Output	输出标志位

参数介绍

表 4-47 IODELAY 参数介绍

参数名	取值范围	默认值	描述
C_STATIC_DLY	1~256	1	静态延时步长控制
DYN_DLY_EN	“FALSE”/”TRUE”	“FALSE”	动态模式使能控制
ADAPT_EN	“FALSE”/”TRUE”	“FALSE”	自适应模式使能控制

原语例化

Verilog 例化：

```
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 例化：

```
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
     );
```

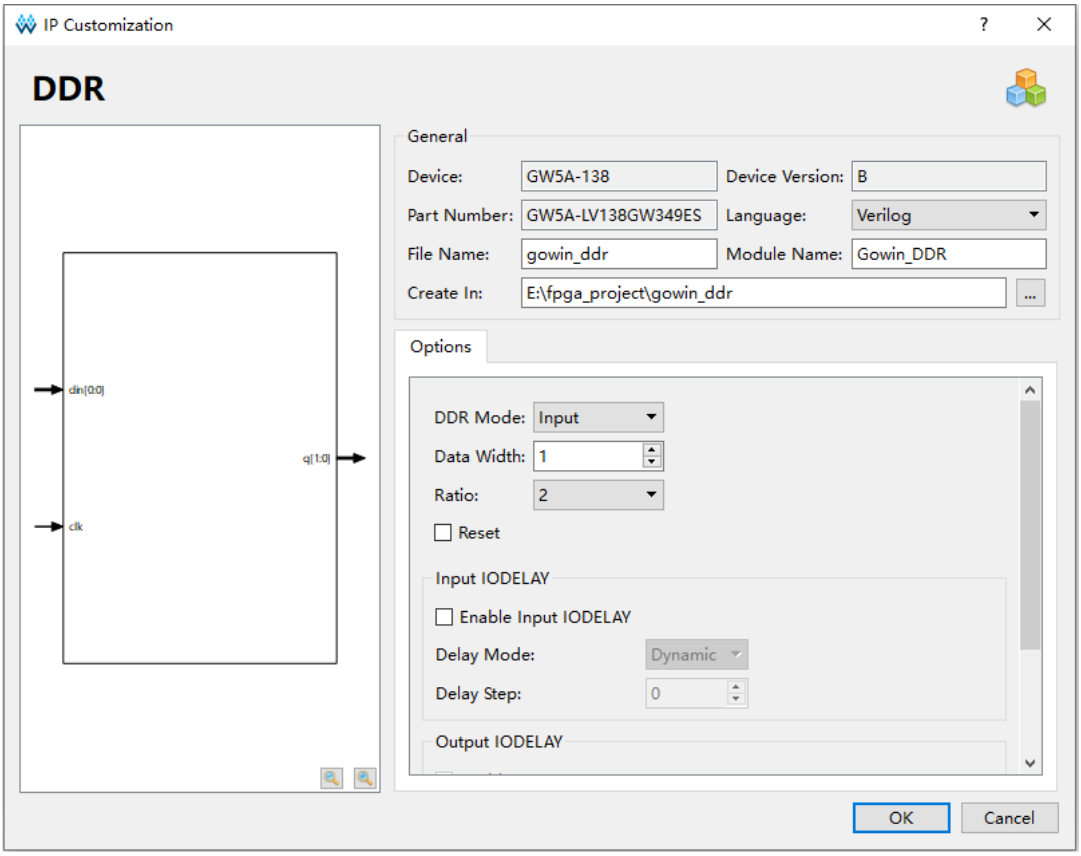
5 IP 调用

当前软件只支持 DDR，在 IP Core Generator 界面中单击 DDR，界面右侧会显示 DDR 的相关信息概要。

5.1 IP 配置

在 IP Core Generator 界面中，双击“DDR”，弹出 DDR 的“IP Customization”窗口，该窗口包括“General”、“Options”配置框和端口显示框图，如图 5-1 所示。

图 5-1 DDR 的 IP Customization 窗口结构



1. General 配置框：General 配置框用于配置产生的 IP 设计文件的相关信

息。

- Device：显示已配置的 Device 信息
- Device Version：显示已配置的 Device Version 信息
- Part Number：显示已配置的 Part Number 信息
- Language：配置产生的 IP 设计文件的硬件描述语言。选择右侧下拉列表框，选择目标语言，支持 Verilog 和 VHDL
- Module Name：配置产生的 IP 设计文件的 module name。在右侧文本框可重新编辑模块名称。Module Name 不能与原语名称相同，若相同，则报出 Error 提示
- File Name：配置产生的 IP 设计文件的文件名。在右侧文本框可重新编辑文件名称
- Create In：配置产生的 IP 设计文件的目标路径。可在右侧文本框中重新编辑目标路径，也可通过文本框右侧选择按钮选择目标路径

2. Options 配置框：Options 配置框用于用户自定义配置 IP，Options 配置框如图 5-1 所示。
   - DDR Mode：配置 DDR 模式，包括输入“Input”、输出“Output”、三态“Tristate”和双向“Bidirectional”，可在右侧选择四种模式
   - Data Width：配置 DDR 的数据宽度，支持的范围是 1~64
   - Ratio：配置 DDR 数据转换的比值，包括 2,4,7,8,10,14,16,32
   - Reset：Ratio 选择 2 时，可选择使能或不使能此选项，使能时将实例化 IDDRC 或 ODDRC
   - Enable Input IODELAY：配置 DDR 是否使用输入延时模块
     - Delay Mode，配置 Delay 模式，“Dynamic”表示使用 IODELAY 并动态调整延时步数，“Static”表示使用 IODELAY 并静态调整延时步数，“Adaptive”表示使用 IODELAY 并自适应调整延时步数
     - Delay Step，选择静态调整延时的步数，范围 0~255
   - Enable Output IODELAY：配置 DDR 是否使用输出延时模块
     - Delay Mode：配置 Delay 模式，“Dynamic”表示使用 IODELAY 并动态调整延时步数，“Static”表示使用 IODELAY 并静态调整延时步数，“Adaptive”表示使用 IODELAY 并自适应调整延时步数
     - Delay Step：选择静态调整延时的步数，范围 0~255
   - Use CLKDIV：使能时将实例化 CLKDIV，对时钟信号 fclk 进行分频，Ratio 为 2 时不能勾选
3. 端口显示框图：端口显示框图显示 IP Core 的配置结果示例框图，如图 5-1 所示。

5.2 IP 生成文件

IP 窗口配置完成后，产生以配置文件“File Name”命名的三个文件，以默认配置为例进行介绍：

- IP 设计文件“gowin_ddr.v”为完整的 verilog 模块，根据用户的 IP 配置，产生对应功能的 DDR 模块
- IP 设计使用模板文件 gowin_ddr_tmp.v，为用户提供 IP 设计使用模板文件
- IP 配置文件“gowin_ddr.ipc”，用户可加载该文件对 IP 进行配置

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。


GOWIN高云
智慧逻辑 定制未来