



# Gowin 可编程通用管脚(GPIO) 用户指南

UG289-1.9,2021-06-21

版权所有 © 2021 广东高云半导体科技股份有限公司

**GOWIN高云**、Gowin、高云、小蜜蜂以及晨熙均为广东高云半导体科技股份有限公司注册商标，本手册中提到的其他任何商标，其所有权利属其拥有者所有。未经本公司书面许可，任何单位和个人都不得擅自摘抄、复制、翻译本文档内容的部分或全部，并不得以任何形式传播。

### **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，或以禁止发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  |
|------------|------|---|
| 2016/05/17 | 1.05 | 初始版本。   |
| 2016/07/15 | 1.06 | 标准化插图。  |
| 2016/08/02 | 1.07 | 适用 GW2A 系列 FPGA 产品。   |
| 2016/10/27 | 1.08 | 适用 GW2AR 系列 FPGA 产品。  |
| 2017/09/01 | 1.09 | 更新 GW1N-6/9 新特性及 GW1NR 相关内容。  |
| 2017/10/12 | 1.10 | 增加 IDE16/OSER16 相关备注信息。   |
| 2017/12/12 | 1.2  | <ul style="list-style-type: none"><li>● 去掉 IDDR/ODDR RESET 信号；</li><li>● 更新 LVDS 描述；</li><li>● 添加带 memory 的输入/输出描述。</li></ul>                           |
| 2018/04/08 | 1.3  | 更新第 7 章图表内容。  |
| 2020/05/14 | 1.4  | <ul style="list-style-type: none"><li>● 更新 3.6 GPIO 原语；</li><li>● 删除 GW1N-6/ GW1NR-6 器件信息。</li></ul>  |
| 2020/08/27 | 1.5  | <ul style="list-style-type: none"><li>● 修改章节结构；</li><li>● 增加第四章“输入输出逻辑”与第五章“IP 调用”。</li></ul>   |
| 2021/01/07 | 1.6  | 增加 IODELAYB 模块内容。   |
| 2021/02/02 | 1.7  | <ul style="list-style-type: none"><li>● 增加 MIPI_IBUF_HS,MIPI_IBUF_LP 描述；</li><li>● 增加 GW2AN-55C、GW1NR-2 新器件支持。</li></ul>                                |
| 2021/03/25 | 1.8  | <ul style="list-style-type: none"><li>● 删除 GW1NZ-2 器件信息；</li><li>● 更新 MIPI_OBUF、MIPI_OBUF_A 的适用器件。</li></ul>  |
| 2021/06/21 | 1.9  | <ul style="list-style-type: none"><li>● 增加器件 GW1N-2B、GW1N-1P5、GW1N-1P5B、GW1NR-2B、GW2AN-18X 和 GW2AN-9X 支持；</li><li>● 更新 IP 调用部分结构图，删除 help 内容。</li></ul> |

# 目录

|                             |           |
|-----------------------------|-----------|
| 目录 .....                    | <b>i</b>  |
| 图目录 .....                   | <b>iv</b> |
| 表目录 .....                   | <b>vi</b> |
| <b>1 关于本手册 .....</b>        | <b>1</b>  |
| 1.1 手册内容 .....              | 1         |
| 1.2 相关文档 .....              | 1         |
| 1.3 术语、缩略语 .....            | 2         |
| 1.4 技术支持与反馈 .....           | 2         |
| <b>2 GPIO 概述 .....</b>      | <b>3</b>  |
| <b>3 输入输出缓存 .....</b>       | <b>5</b>  |
| 3.1 GPIO 电平标准 .....         | 5         |
| 3.2 GPIO 分区策略 .....         | 8         |
| 3.2.1 GW1N 系列 FPGA 产品 ..... | 8         |
| 3.2.2 GW2A 系列 FPGA 产品 ..... | 9         |
| 3.3 供电要求 .....              | 10        |
| 3.3.1 LVCMOS 缓存配置 .....     | 10        |
| 3.3.2 差分缓存配置 .....          | 10        |
| 3.4 模拟差分电路匹配网络 .....        | 11        |
| 3.4.1 模拟 LVDS .....         | 11        |
| 3.4.2 模拟 LVPECL .....       | 11        |
| 3.4.3 模拟 RSDS .....         | 11        |
| 3.4.4 模拟 BLVDS .....        | 11        |
| 3.5 GPIO 软件配置 .....         | 12        |
| 3.5.1 位置 .....              | 12        |
| 3.5.2 电平标准 .....            | 12        |
| 3.5.3 驱动能力 .....            | 12        |
| 3.5.4 上下拉模式 .....           | 12        |
| 3.5.5 参考电压 .....            | 12        |

|                                       |           |
|---------------------------------------|-----------|
| 3.5.6 迟滞.....                         | 13        |
| 3.5.7 漏极开路.....                       | 13        |
| 3.5.8 转换速率.....                       | 13        |
| 3.5.9 单端匹配电阻.....                     | 13        |
| 3.5.10 差分匹配电阻.....                    | 13        |
| 3.6 GPIO 原语.....                      | 13        |
| 3.6.1 IBUF.....                       | 13        |
| 3.6.2 OBUF.....                       | 14        |
| 3.6.3 TBUF.....                       | 15        |
| 3.6.4 IOBUF.....                      | 17        |
| 3.6.5 LVDS Input Buffer.....          | 18        |
| 3.6.6 LVDS Ouput Buffer.....          | 20        |
| 3.6.7 LVDS Tristate Buffer.....       | 22        |
| 3.6.8 LVDS Inout Buffer.....          | 24        |
| 3.6.9 MIPI_IBUF.....                  | 25        |
| 3.6.10 MIPI_OBUF.....                 | 28        |
| 3.6.11 MIPI_OBUF_A.....               | 30        |
| 3.6.12 I3C_IOBUF.....                 | 31        |
| 3.6.13 MIPI_IBUF_HS/MIPI_IBUF_LP..... | 33        |
| <b>4 输入输出逻辑.....</b>                  | <b>36</b> |
| 4.1 SDR 模式.....                       | 37        |
| 4.2 DDR 模式输入逻辑.....                   | 37        |
| 4.2.1 IDDR.....                       | 37        |
| 4.2.2 IDDRC.....                      | 39        |
| 4.2.3 IDES4.....                      | 41        |
| 4.2.4 IDES8.....                      | 44        |
| 4.2.5 IDES10.....                     | 47        |
| 4.2.6 IVIDEO.....                     | 50        |
| 4.2.7 IDES16.....                     | 53        |
| 4.2.8 IDDR_MEM.....                   | 57        |
| 4.2.9 IDES4_MEM.....                  | 60        |
| 4.2.10 IDES8_MEM.....                 | 63        |
| 4.3 DDR 模式输出逻辑.....                   | 67        |
| 4.3.1 ODDR.....                       | 67        |
| 4.3.2 ODDRC.....                      | 70        |
| 4.3.3 OSER4.....                      | 73        |
| 4.3.4 OSER8.....                      | 76        |
| 4.3.5 OSER10.....                     | 80        |
| 4.3.6 OVIDEO.....                     | 83        |

---

|                       |            |
|-----------------------|------------|
| 4.3.7 OSER16.....     | 85         |
| 4.3.8 ODDR_MEM.....   | 89         |
| 4.3.9 OSER4_MEM.....  | 92         |
| 4.3.10 OSER8_MEM..... | 96         |
| 4.4 延时模块.....         | 101        |
| 4.4.1 IODELAY.....    | 101        |
| 4.4.2 IODELAYC.....   | 103        |
| 4.4.3 IODELAYB.....   | 106        |
| 4.5 取样模块.....         | 110        |
| <b>5 IP 调用.....</b>   | <b>113</b> |
| 5.1 IP 配置.....        | 113        |
| 5.2 IP 生成文件.....      | 115        |

# 图目录

|   |    |
|---|----|
| 图 2-1 输入输出模块结构示意图.....                      | 3  |
| 图 3-1 GW1N-4 分区分布示意图.....                   | 9  |
| 图 3-2 GW2A 系列 FPGA 产品分区分布示意图.....           | 9  |
| 图 3-3 LVDS25E 匹配网络.....                     | 11 |
| 图 3-4 LVPECL 匹配网络.....                      | 11 |
| 图 3-5 RSDSE 匹配网络.....                       | 11 |
| 图 3-6 BLVDS 匹配网络.....                       | 12 |
| 图 3-7 IBUF 端口示意图.....                       | 13 |
| 图 3-8 OBUF 端口示意图.....                       | 14 |
| 图 3-9 TBUF 端口示意图.....                       | 16 |
| 图 3-10 IOBUF 端口示意图.....                     | 17 |
| 图 3-11 TLVDS_IBUF/ELVDS_IBUF 端口示意图.....     | 18 |
| 图 3-12 TLVDS_OBUF/ELVDS_OBUF 端口示意图.....     | 20 |
| 图 3-13 TLVDS_TBUF/ELVDS_TBUF 端口示意图.....     | 22 |
| 图 3-14 TLVDS_IOBUF/ELVDS_IOBUF 端口示意图.....   | 24 |
| 图 3-15 MIPI_IBUF 端口示意图.....                 | 26 |
| 图 3-16 MIPI_OBUF 端口示意图.....                 | 28 |
| 图 3-17 MIPI_OBUF_A 端口示意图.....               | 30 |
| 图 3-18 I3C_IOBUF 端口示意图.....                 | 32 |
| 图 3-19 MIPI_IBUF_HS/MIPI_IBUF_LP 端口示意图..... | 33 |
| 图 4-1 输入输出逻辑输出示意图 – 输出部分.....               | 36 |
| 图 4-2 输入输出逻辑输入示意图 – 输入部分.....               | 37 |
| 图 4-3 IDDR 逻辑框图.....                        | 37 |
| 图 4-4 IDDR 时序图.....                         | 38 |
| 图 4-5 IDDR 端口示意图.....                       | 38 |
| 图 4-6 IDDR_C 端口示意图.....                     | 40 |
| 图 4-7 CALIB 示例时序图.....                      | 42 |
| 图 4-8 IDES4 端口示意图.....                      | 42 |
| 图 4-9 IDES8 端口示意图.....                      | 45 |

|   |     |
|---|-----|
| 图 4-10 IDER10 端口示意图 .....               | 48  |
| 图 4-11 IVIDEO 端口示意图 .....               | 51  |
| 图 4-12 IDER16 端口示意图 .....               | 54  |
| 图 4-13 IDDR_MEM 端口示意图 .....             | 58  |
| 图 4-14 IDER4_MEM 端口示意图 .....            | 61  |
| 图 4-15 IDER8_MEM 端口示意图 .....            | 64  |
| 图 4-16 ODDR 逻辑框图 .....                  | 67  |
| 图 4-17 ODDR 时序图 .....                   | 68  |
| 图 4-18 ODDR 端口示意图 .....                 | 68  |
| 图 4-19 ODDRC 逻辑框图 .....                 | 70  |
| 图 4-20 ODDRC 端口示意图 .....                | 71  |
| 图 4-21 OSER4 逻辑框图 .....                 | 73  |
| 图 4-22 OSER4 端口示意图 .....                | 73  |
| 图 4-23 OSER8 逻辑框图 .....                 | 76  |
| 图 4-24 OSER8 端口示意图 .....                | 77  |
| 图 4-25 OSER10 端口示意图 .....               | 80  |
| 图 4-26 OVIDEO 端口示意图 .....               | 83  |
| 图 4-27 OSER16 端口示意图 .....               | 86  |
| 图 4-28 ODDR_MEM 逻辑框图 .....              | 89  |
| 图 4-29 ODDR_MEM 端口示意图 .....             | 90  |
| 图 4-30 OSER4_MEM 逻辑框图 .....             | 93  |
| 图 4-31 OSER4_MEM 端口示意图 .....            | 93  |
| 图 4-32 OSER8_MEM 逻辑框图 .....             | 97  |
| 图 4-33 OSER8_MEM 端口示意图 .....            | 98  |
| 图 4-34 IODELAY 端口示意图 .....              | 102 |
| 图 4-35 IODELAYC 端口示意图 .....             | 104 |
| 图 4-36 IODELAYB 内部结构框图 .....            | 107 |
| 图 4-37 IODELAYB 端口示意图 .....             | 107 |
| 图 4-38 IEM 端口示意图 .....                  | 110 |
| 图 5-1 DDR 的 IP Customization 窗口结构 ..... | 113 |



# 表目录

|  |    |
|--|----|
| 表 1-1 术语、缩略语.....                          | 2  |
| 表 3-1 高云半导体 FPGA 产品支持的输出电平标准及部分可选配置.....   | 5  |
| 表 3-2 高云半导体 FPGA 产品支持的输入电平标准及部分可选配置.....   | 6  |
| 表 3-3 IBUF 端口介绍.....                       | 14 |
| 表 3-4 OBUF 端口介绍.....                       | 15 |
| 表 3-5 TBUF 端口介绍.....                       | 16 |
| 表 3-6 IOBUF 端口介绍.....                      | 17 |
| 表 3-7 TLVDS_IBUF/ELVDS_IBUF 端口介绍.....      | 18 |
| 表 3-8 TLVDS_OBUF/ELVDS_OBUF 端口介绍.....      | 20 |
| 表 3-9 TLVDS_TBUF/ELVDS_TBUF 端口介绍.....      | 22 |
| 表 3-10 TLVDS_IOBUF 适用器件.....               | 24 |
| 表 3-11 TLVDS_IOBUF/ELVDS_IOBUF 端口介绍.....   | 24 |
| 表 3-12 MIPI_IBUF 适用器件.....                 | 26 |
| 表 3-13 MIPI_IBUF 端口介绍.....                 | 26 |
| 表 3-14 MIPI_OBUF 适用器件.....                 | 28 |
| 表 3-15 MIPI_OBUF 端口介绍.....                 | 29 |
| 表 3-16 MIPI_OBUF_A 适用器件(附加).....           | 30 |
| 表 3-17 MIPI_OBUF_A 端口介绍.....               | 30 |
| 表 3-18 I3C_IOBUF 适用器件.....                 | 32 |
| 表 3-19 I3C_IOBUF 端口介绍.....                 | 32 |
| 表 3-20 MIPI_IBUF_HS/MIPI_IBUF_LP 适用器件..... | 33 |
| 表 3-21 MIPI_IBUF_HS 端口介绍.....              | 34 |
| 表 3-22 MIPI_IBUF_LP 端口介绍.....              | 34 |
| 表 4-1 IDDR 端口介绍.....                       | 38 |
| 表 4-2 IDDR 参数介绍.....                       | 38 |
| 表 4-3 IDDRC 端口介绍.....                      | 40 |
| 表 4-4 IDDRC 参数介绍.....                      | 40 |
| 表 4-5 IDES4 端口介绍.....                      | 42 |
| 表 4-6 IDES4 参数介绍.....                      | 43 |

|                             |    |
|-----------------------------|----|
| 表 4-7 IDES8 端口介绍.....       | 45 |
| 表 4-8 IDES8 参数介绍.....       | 45 |
| 表 4-9 IDES10 端口介绍.....      | 48 |
| 表 4-10 IDES10 参数介绍.....     | 48 |
| 表 4-11 IVIDEO 端口介绍 .....    | 51 |
| 表 4-12 IVIDEO 参数介绍 .....    | 51 |
| 表 4-13 IDES16 适用器件.....     | 53 |
| 表 4-14 IDES16 端口介绍.....     | 54 |
| 表 4-15 IDES16 参数介绍.....     | 54 |
| 表 4-16 IDDR_MEM 适用器件.....   | 57 |
| 表 4-17 IDDR_MEM 端口介绍.....   | 58 |
| 表 4-18 IDDR_MEM 参数介绍.....   | 58 |
| 表 4-19 IDES4_MEM 适用器件 ..... | 60 |
| 表 4-20 IDES4_MEM 端口介绍 ..... | 61 |
| 表 4-21 IDES4_MEM 参数介绍 ..... | 61 |
| 表 4-22 IDES8_MEM 适用器件 ..... | 63 |
| 表 4-23 IDES8_MEM 端口介绍 ..... | 64 |
| 表 4-24 IDES8_MEM 参数介绍 ..... | 65 |
| 表 4-25 ODDR 端口介绍 .....      | 68 |
| 表 4-26 ODDR 参数介绍 .....      | 68 |
| 表 4-27 ODDRC 端口介绍.....      | 71 |
| 表 4-28 ODDRC 参数介绍.....      | 71 |
| 表 4-29 OSER4 端口介绍.....      | 73 |
| 表 4-30 OSER4 参数介绍.....      | 74 |
| 表 4-31 OSER8 端口介绍.....      | 77 |
| 表 4-32 OSER8 参数介绍.....      | 77 |
| 表 4-33 OSER10 端口介绍.....     | 80 |
| 表 4-34 OSER10 参数介绍.....     | 81 |
| 表 4-35 OVIDEO 端口介绍 .....    | 83 |
| 表 4-36 OVIDEO 参数介绍 .....    | 83 |
| 表 4-37 OSER16 适用器件.....     | 85 |
| 表 4-38 OSER16 端口介绍.....     | 86 |
| 表 4-39 OSER16 参数介绍.....     | 86 |
| 表 4-40 ODDR_MEM 适用器件.....   | 89 |
| 表 4-41 ODDR_MEM 端口介绍.....   | 90 |
| 表 4-42 ODDR_MEM 参数介绍.....   | 90 |
| 表 4-43 OSER4_MEM 适用器件 ..... | 92 |

---

|                             |     |
|-----------------------------|-----|
| 表 4-44 OSER4_MEM 端口介绍 ..... | 94  |
| 表 4-45 OSER4_MEM 参数介绍 ..... | 94  |
| 表 4-46 OSER8_MEM 适用器件 ..... | 97  |
| 表 4-47 OSER8_MEM 端口介绍 ..... | 98  |
| 表 4-48 OSER8_MEM 参数介绍 ..... | 98  |
| 表 4-49 IODELAY 端口介绍 .....   | 102 |
| 表 4-50 IODELAY 参数介绍 .....   | 102 |
| 表 4-51 IODELAYC 适用器件 .....  | 104 |
| 表 4-52 IODELAYC 端口介绍 .....  | 104 |
| 表 4-53 IODELAYC 参数介绍 .....  | 105 |
| 表 4-54 IODELAYB 适用器件 .....  | 106 |
| 表 4-55 IODELAYB 端口介绍 .....  | 108 |
| 表 4-56 IODELAYB 参数介绍 .....  | 108 |
| 表 4-57 IEM 端口介绍 .....       | 110 |
| 表 4-58 IEM 参数介绍 .....       | 111 |

# 1 关于本手册

## 1.1 手册内容

Gowin 可编程通用管脚(GPIO)主要描述了高云半导体 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 GPIO 的架构和 Gowin 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](http://www.gowinsemi.com.cn) 可以下载、查看以下相关器件文档：

1. [DS100](#)，GW1N 系列 FPGA 产品数据手册
2. [DS117](#)，GW1NR 系列 FPGA 产品数据手册
3. [DS821](#)，GW1NS 系列 FPGA 产品数据手册
4. [DS841](#)，GW1NZ 系列 FPGA 产品数据手册
5. [DS861](#)，GW1NSR 系列 FPGA 产品数据手册
6. [DS871](#)，GW1NSE 系列安全 FPGA 产品数据手册
7. [DS881](#)，GW1NSER 系列安全 FPGA 产品数据手册
8. [DS891](#)，GW1NRF 系列蓝牙 FPGA 产品数据手册
9. [DS102](#)，GW2A 系列 FPGA 产品数据手册
10. [DS226](#)，GW2AR 系列 FPGA 产品数据手册
11. [DS961](#)，GW2ANR 系列 FPGA 产品数据手册
12. [DS971](#)，GW2AN 系列 FPGA 产品数据手册

## 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               |
|------------|---------------------------------|------------------|
| 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           |
| CFU        | Configurable Function Unit      | 可配置功能单元          |
| CRU        | Configurable Routing Unit       | 可编程布线单元          |
| Slew Rate  | Slew Rate                       | 转换速率             |
| Bus Keeper | Bus Keeper                      | 总线保持             |
| Open Drain | Open Drain                      | 漏极开路             |
| SDR        | Single Data Rate                | 单倍速率             |
| DDR        | Double Data Rate                | 双倍速率             |
| SER        | Serializer                      | 串行器              |
| DES        | Deserializer                    | 解串器              |
| TLDO       | True LVDS Output                | 真 LVDS 输出(电流输出)  |
| 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    |

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

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

Tel: +86 0755 8262 0391

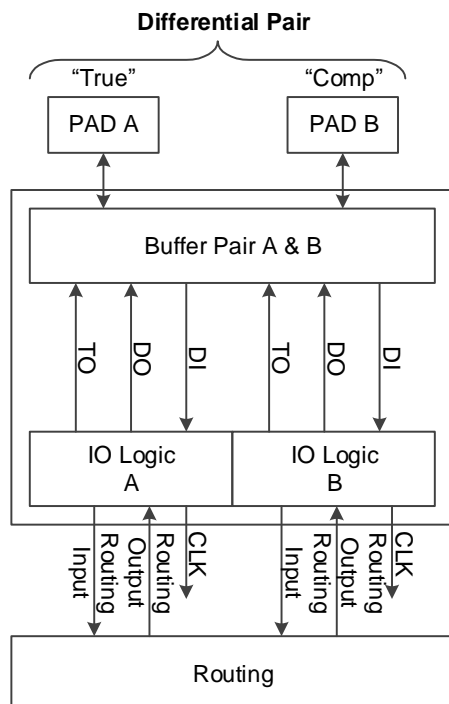
# 2 GPIO 概述

高云半导体 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。

高云半导体 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



高云半导体 FPGA 系列产品中输入输出模块的功能特点:

- 基于分区(Bank)的管脚供电( $V_{CC0}$ )机制
  - 支持 LVCMOS、PCI、LVTTTL、LVDS、SSTL 以及 HSTL 等多种电平标准
  - 部分器件<sup>[1]</sup>支持 MIPI 电平标准以及 MIPI I3C OpenDrain/PushPull 转换
  - 提供输入信号去迟滞选项
  - 提供输出信号驱动电流选项
  - 提供输出信号转换速率选项
  - 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
- 注!

[1]: 支持 MIPI、I3C 的具体器件可参考 3.6.9、3.6.10 和 3.6.12 适用器件部分。

# 3 输入输出缓存

## 3.1 GPIO 电平标准

高云半导体 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 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 FPGA 产品不同的电平标准对管脚电压的要求，如表 3-1 和表 3-2 所示。

**表 3-1 高云半导体 FPGA 产品支持的输出电平标准及部分可选配置**

| I/O 输出标准  | 单端/差分 | Bank V <sub>CC0</sub> (V) | 输出驱动能力(mA)   |
|-----------|-------|---------------------------|--------------|
| LVTTTL33  | 单端    | 3.3                       | 4,8,12,16,24 |
| LVCNMOS33 | 单端    | 3.3                       | 4,8,12,16,24 |
| LVCNMOS25 | 单端    | 2.5                       | 4,8,12,16    |
| LVCNMOS18 | 单端    | 1.8                       | 4,8,12       |
| LVCNMOS15 | 单端    | 1.5                       | 4,8          |
| LVCNMOS12 | 单端    | 1.2                       | 4,8          |
| SSTL25_I  | 单端    | 2.5                       | 8            |
| SSTL25_II | 单端    | 2.5                       | 8            |
| SSTL33_I  | 单端    | 3.3                       | 8            |
| SSTL33_II | 单端    | 3.3                       | 8            |
| SSTL18_I  | 单端    | 1.8                       | 8            |
| SSTL18_II | 单端    | 1.8                       | 8            |
| SSTL15    | 单端    | 1.5                       | 8            |
| HSTL18_I  | 单端    | 1.8                       | 8            |
| HSTL18_II | 单端    | 1.8                       | 8            |
| HSTL15_I  | 单端    | 1.5                       | 8            |



| I/O 输出标准   | 单端/差分 | Bank V <sub>CCO</sub> (V) | 输出驱动能力(mA)       |
|------------|-------|---------------------------|------------------|
| PCI33      | 单端    | 3.3                       | N/A              |
| LVPECL33E  | 差分    | 3.3                       | 16               |
| MLVDS25E   | 差分    | 2.5                       | 16               |
| BLVDS25E   | 差分    | 2.5                       | 16               |
| RSDS25E    | 差分    | 2.5                       | 8                |
| LVDS25E    | 差分    | 2.5                       | 8                |
| LVDS25     | 差分    | 2.5                       | 1.25,2.0,2.5,3.5 |
| RSDS       | 差分    | 2.5/3.3                   | 2                |
| MINILVDS   | 差分    | 2.5/3.3                   | 2                |
| PPLVDS     | 差分    | 2.5/3.3                   | 3.5              |
| SSTL15D    | 差分    | 1.5                       | 8                |
| SSTL25D_I  | 差分    | 2.5                       | 8                |
| SSTL25D_II | 差分    | 2.5                       | 8                |
| SSTL33D_I  | 差分    | 3.3                       | 8                |
| SSTL33D_II | 差分    | 3.3                       | 8                |
| SSTL18D_I  | 差分    | 1.8                       | 8                |
| SSTL18D_II | 差分    | 1.8                       | 8                |
| HSTL18D_I  | 差分    | 1.8                       | 8                |
| HSTL18D_II | 差分    | 1.8                       | 8                |
| HSTL15D_I  | 差分    | 1.5                       | 8                |
| LVC MOS12D | 差分    | 1.2                       | 8,4              |
| LVC MOS15D | 差分    | 1.5                       | 8,4              |
| LVC MOS18D | 差分    | 1.8                       | 8,12,4           |
| LVC MOS25D | 差分    | 2.5                       | 8,16,12,4        |
| LVC MOS33D | 差分    | 3.3                       | 8,24,16,12,4     |
| MIPI       | 差分    | 1.2                       | 3.5              |

表 3-2 高云半导体 FPGA 产品支持的输入电平标准及部分可选配置

| I/O 输入标准  | 单端/差分 | Bank V <sub>CCO</sub> (V) | 支持去迟滞选项 | 是否需要 V <sub>REF</sub> |
|-----------|-------|---------------------------|---------|-----------------------|
| LVTTTL33  | 单端    | 3.3                       | 是       | 否                     |
| LVC MOS33 | 单端    | 3.3                       | 是       | 否                     |
| LVC MOS25 | 单端    | 2.5                       | 是       | 否                     |
| LVC MOS18 | 单端    | 1.8                       | 是       | 否                     |
| LVC MOS15 | 单端    | 1.5                       | 是       | 否                     |
| LVC MOS12 | 单端    | 1.2                       | 是       | 否                     |
| SSTL15    | 单端    | 1.5                       | 否       | 是                     |
| SSTL25_I  | 单端    | 2.5                       | 否       | 是                     |

| I/O 输入标准      | 单端/差分 | Bank V <sub>CC0</sub> (V) | 支持去迟滞选项 | 是否需要 V <sub>REF</sub> |
|---------------|-------|---------------------------|---------|-----------------------|
| SSTL25_II     | 单端    | 2.5                       | 否       | 是                     |
| SSTL33_I      | 单端    | 3.3                       | 否       | 是                     |
| SSTL33_II     | 单端    | 3.3                       | 否       | 是                     |
| SSTL18_I      | 单端    | 1.8                       | 否       | 是                     |
| SSTL18_II     | 单端    | 1.8                       | 否       | 是                     |
| HSTL18_I      | 单端    | 1.8                       | 否       | 是                     |
| HSTL18_II     | 单端    | 1.8                       | 否       | 是                     |
| HSTL15_I      | 单端    | 1.5                       | 否       | 是                     |
| PCI33         | 单端    | 3.3                       | 是       | 否                     |
| LVC MOS33OD25 | 单端    | 2.5                       | 否       | 否                     |
| LVC MOS33OD18 | 单端    | 1.8                       | 否       | 否                     |
| LVC MOS33OD15 | 单端    | 1.5                       | 否       | 否                     |
| LVC MOS25OD18 | 单端    | 1.8                       | 否       | 否                     |
| LVC MOS25OD15 | 单端    | 1.5                       | 否       | 否                     |
| LVC MOS18OD15 | 单端    | 1.5                       | 否       | 否                     |
| LVC MOS15OD12 | 单端    | 1.2                       | 否       | 否                     |
| LVC MOS25UD33 | 单端    | 3.3                       | 否       | 否                     |
| LVC MOS18UD25 | 单端    | 2.5                       | 否       | 否                     |
| LVC MOS18UD33 | 单端    | 3.3                       | 否       | 否                     |
| LVC MOS15UD18 | 单端    | 1.8                       | 否       | 否                     |
| LVC MOS15UD25 | 单端    | 2.5                       | 否       | 否                     |
| LVC MOS15UD33 | 单端    | 3.3                       | 否       | 否                     |
| LVC MOS12UD15 | 单端    | 1.5                       | 否       | 否                     |
| LVC MOS12UD18 | 单端    | 1.8                       | 否       | 否                     |
| LVC MOS12UD25 | 单端    | 2.5                       | 否       | 否                     |
| LVC MOS12UD33 | 单端    | 3.3                       | 否       | 否                     |
| LVDS25        | 差分    | 2.5                       | 否       | 否                     |
| RS DS         | 差分    | 2.5/3.3                   | 否       | 否                     |
| MINILVDS      | 差分    | 2.5/3.3                   | 否       | 否                     |
| PPLVDS        | 差分    | 2.5/3.3                   | 否       | 否                     |
| LVDS25E       | 差分    | 2.5                       | 否       | 否                     |
| MLVDS25E      | 差分    | 2.5                       | 否       | 否                     |
| BLVDS25E      | 差分    | 2.5                       | 否       | 否                     |
| RS DS25E      | 差分    | 2.5                       | 否       | 否                     |
| LVPECL33E     | 差分    | 3.3                       | 否       | 否                     |
| SSTL15D       | 差分    | 1.5                       | 否       | 否                     |
| SSTL25D_I     | 差分    | 2.5                       | 否       | 否                     |
| SSTL25D_II    | 差分    | 2.5                       | 否       | 否                     |

| I/O 输入标准   | 单端/差分 | Bank V <sub>CC0</sub> (V) | 支持去迟滞选项 | 是否需要 V <sub>REF</sub> |
|------------|-------|---------------------------|---------|-----------------------|
| SSTL33D_I  | 差分    | 3.3                       | 否       | 否                     |
| SSTL33D_II | 差分    | 3.3                       | 否       | 否                     |
| SSTL18D_I  | 差分    | 1.8                       | 否       | 否                     |
| SSTL18D_II | 差分    | 1.8                       | 否       | 否                     |
| HSTL18D_I  | 差分    | 1.8                       | 否       | 否                     |
| HSTL18D_II | 差分    | 1.8                       | 否       | 否                     |
| HSTL15D_I  | 差分    | 1.5                       | 否       | 否                     |
| LVC MOS12D | 差分    | 1.2                       | 否       | 否                     |
| LVC MOS15D | 差分    | 1.5                       | 否       | 否                     |
| LVC MOS18D | 差分    | 1.8                       | 否       | 否                     |
| LVC MOS25D | 差分    | 2.5                       | 否       | 否                     |
| LVC MOS33D | 差分    | 3.3                       | 否       | 否                     |
| MIPI       | 差分    | 1.2                       | 否       | 否                     |

## 3.2 GPIO 分区策略

GPIO 的通用属性：

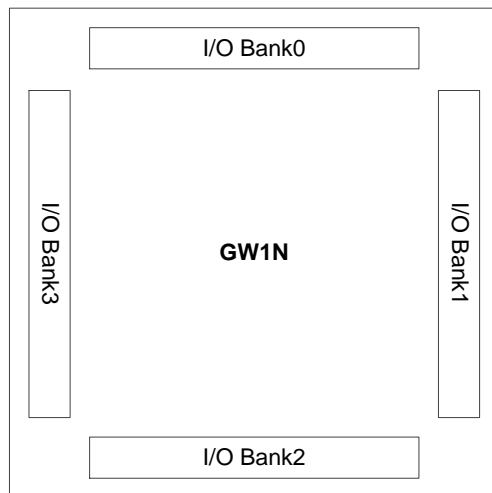
- 所有分区支持模拟 LVDS 差分输出，但需要使用外部电阻网络；
- 所有分区支持上拉、下拉以及总线保持设置；
- 每个分区支持一种管脚电压；
- 每个分区支持一个参考电压信号，无论它来自外部管脚或者来自内部参考电压生成器。

本手册以 GW1N 和 GW2A 系列器件为例来介绍高云 FPGA 产品的 GPIO 分区策略，其他系列器件的分区情况可参考对应产品数据手册。

### 3.2.1 GW1N 系列 FPGA 产品

GW1N 系列 FPGA 产品分区以 GW1N-4 器件为例介绍，如图 3-1 所示，管脚被划分为 4 个分区，每个分区由独立的管脚电源(V<sub>CC0</sub>)供电，管脚电源可以配置为 3.3V、2.5V、1.8V、1.5V 或 1.2V。GW1N 系列其他器件的分区情况可参考 [DS100](#)，GW1N 系列 FPGA 产品数据手册中“I/O 电平标准”部分。

图 3-1 GW1N-4 分区分布示意图

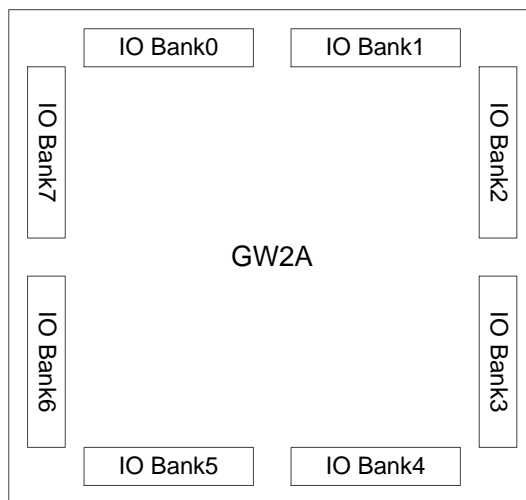


GW1N 系列 FPGA 产品在分区 1/2/3 支持真 LVDS 差分输出，分区 0 支持 100 欧姆输入差分匹配电阻。GW1N-9、GW1N-9C 器件的 top 分区支持 MIPI 输入，bottom 分区支持 MIPI 输出。

### 3.2.2 GW2A 系列 FPGA 产品

GW2A 系列 FPGA 产品的管脚被划分为 8 个分区，每个分区由独立的管脚电源(Vcco)供电，管脚电源可以配置为 3.3V、2.5V、1.8V、1.5V 或 1.2V。

图 3-2 GW2A 系列 FPGA 产品分区分布示意图



GW2A 系列 FPGA 产品所有分区都支持真 LVDS 差分输出，分区 0/1 支持 100 欧姆输入差分匹配电阻。

## 3.3 供电要求

核电压( $V_{CC}$ )和管脚电压( $V_{CC0}$ )达到特定阈值时, 内部的上电复位信号(POR)会置位, 高云半导体 FPGA 产品内核逻辑被激活。空白芯片默认的 GPIO 状态是三态输入弱上拉。高云半导体 FPGA 产品对内核电压和管脚电压无上下电顺序要求。

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 和 HSTL 等电平标准输入, 参考电压设置为管脚电压的一半。输入参考电压也可由内部参考电压生成器产生。由于每个分区只有一条参考电压总线, 一个分区内部参考电压生成器和外部参考电压输入管脚不能同时有效。

高云半导体 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可以参照表 3-1。高云半导体 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。

去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 缓存都支持去迟滞设置。

转换速率设置会在时钟上升沿和时钟下降沿同时生效, LVCMOS 缓存可以配置成低噪声模式(SLOW)和高速模式(FAST)。

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, 输入去迟滞和总线保持特性被禁用。

GW1N 器件在分区 0 支持片内可编程的 100 欧姆输入差分匹配电阻。GW1N 器件分区分布示意图如图 3-1 所示。

GW2A 器件在分区 0/1 支持片内可编程的 100 欧姆输入差分匹配电阻。GW2A 器件分区分布示意图如图 3-2 所示。

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。

GW1N 器件在分区 1/2/3 支持真 LVDS 差分输出驱动。

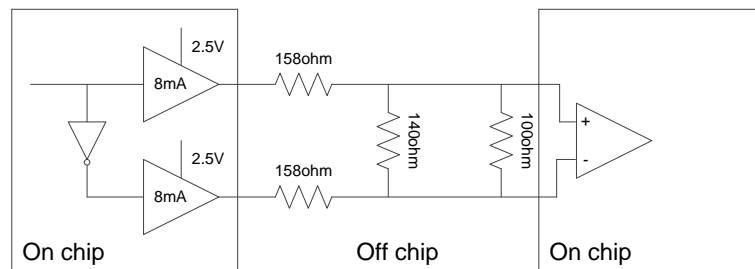
GW2A 器件在所有分区支持真 LVDS 差分输出驱动。

## 3.4 模拟差分电路匹配网络

### 3.4.1 模拟 LVDS

高云半导体 FPGA 产品通过互补的 LVC MOS 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 3-3 所示。

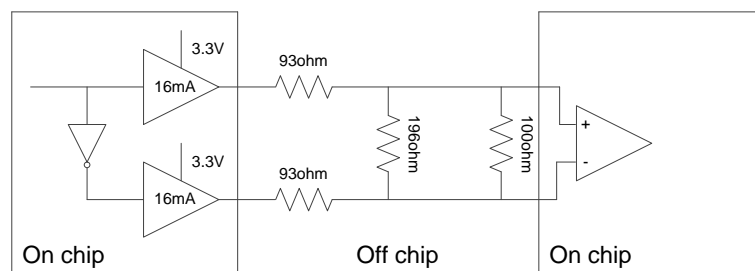
图 3-3 LVDS25E 匹配网络



### 3.4.2 模拟 LVPECL

高云半导体 FPGA 产品通过互补的 LVC MOS 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 3-4 所示。

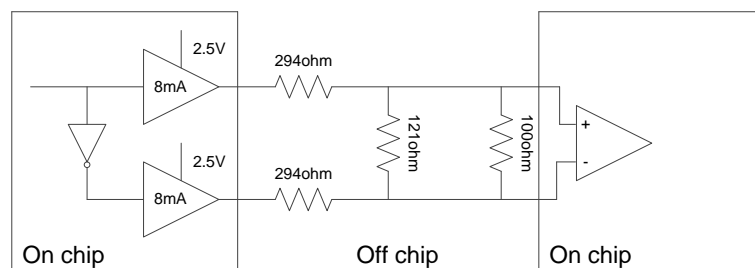
图 3-4 LVPECL 匹配网络



### 3.4.3 模拟 RS DS

高云半导体 FPGA 产品通过互补的 LVC MOS 输出加上外部匹配网络可以构建兼容 RS DS 输出标准，其外部匹配网络如图 3-5 所示。

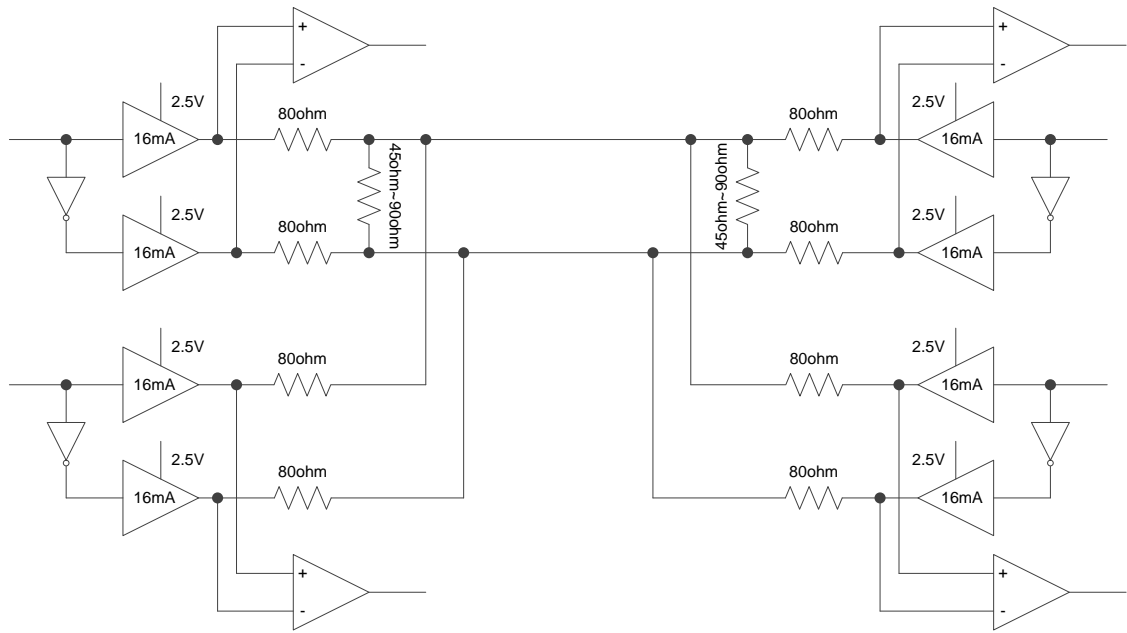
图 3-5 RSDSE 匹配网络



### 3.4.4 模拟 BLVDS

高云半导体 FPGA 产品通过互补的 LVC MOS 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 3-6 所示。

图 3-6 BLVDS 匹配网络



## 3.5 GPIO 软件配置

可通过 Gowin 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=LVC MOS18D;
```

### 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 转换速率

为输出管脚或双向管脚设置转换速率，SLOW：低噪声模式；FAST：高速模式。

```
IO_PORT "xxx" SLEW_RATE=SLOW;
```

### 3.5.9 单端匹配电阻

为单端信号设置终端匹配电阻，提供 OFF 和 100 欧选项。

```
IO_PORT "xxx" SINGLE_RESISTOR=100;
```

### 3.5.10 差分匹配电阻

为差分信号设置终端匹配电阻，提供 OFF 和 100 欧选项。

```
IO_PORT "xxx" Diff_RESISTOR=100;
```

## 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 3-7 IBUF 端口示意图





## 端口介绍

表 3-3 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 3-8 OBUF 端口示意图



## 端口介绍

表 3-4 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
    );
```

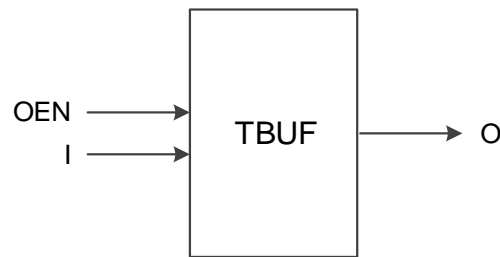
## 3.6.3 TBUF

### 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 3-9 TBUF 端口示意图



## 端口介绍

表 3-5 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
);
```

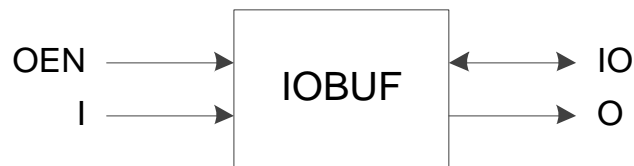
### 3.6.4 IOBUF

#### 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 3-10 IOBUF 端口示意图



#### 端口介绍

表 3-6 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 差分输入分为两种：TLVDS\_IBUF 和 ELVDS\_IBUF。

TLVDS\_IBUF(True LVDS Input Buffer)，真差分输入缓冲器。

注！

GW1NZ-1、GW1N-1S 器件不支持 TLVDS\_IBUF。

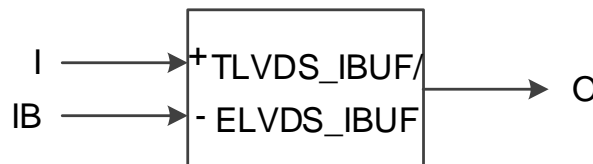
ELVDS\_IBUF(Emulated LVDS Input Buffer)，模拟差分输入缓冲器。

注！

GW1NZ-1 器件不支持 ELVDS\_IBUF。

#### 端口示意图

图 3-11 TLVDS\_IBUF/ELVDS\_IBUF 端口示意图



#### 端口介绍

表 3-7 TLVDS\_IBUF/ELVDS\_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

示例一

**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
    );

```

## 示例二

**Verilog 例化:**

```

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

```

**Vhdl 例化:**

```

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

```

```

uut:ELVDS_IBUF
    PORT MAP(
        O=>O,
        I=>I,
        IB=> IB
    );

```

### 3.6.6 LVDS Output 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。

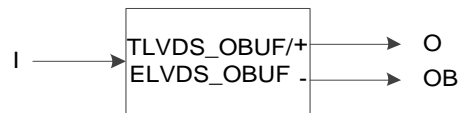
注！

GW1N-1、GW1NR-1、GW1NZ-1、GW1N-1S 器件不支持 TLVDS\_OBUF。

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

#### 端口示意图

图 3-12 TLVDS\_OBUF/ELVDS\_OBUF 端口示意图



#### 端口介绍

表 3-8 TLVDS\_OBUF/ELVDS\_OBUF 端口介绍

| 端口 | I/O    | 描述        |
|----|--------|-----------|
| I  | Input  | 数据输入信号    |
| 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_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。

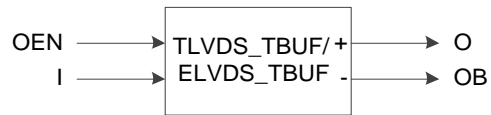
注！

GW1N-1、GW1NR-1、GW1NZ-1、GW1N-1S 器件不支持 TLVDS\_TBUF。

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 3-13 TLVDS\_TBUF/ELVDS\_TBUF 端口示意图



#### 端口介绍

表 3-9 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。

#### 适用器件

表 3-10 TLVDS\_IOBUF 适用器件

| 家族                         | 系列     | 器件                                   |
|----------------------------|--------|--------------------------------------|
| 晨熙®(Arora)<br>家族           | GW2A   | GW2A-18, GW2A-18C, GW2A-55, GW2A-55C |
|                            | GW2AN  | GW2AN-55C                            |
|                            | GW2AR  | GW2AR-18, GW2AR-18C                  |
|                            | GW2ANR | GW2ANR-18C                           |
| 小蜜蜂®<br>(LittleBee®)<br>家族 | GW1N   | GW1N-4, GW1N-4B, GW1N-4C             |
|                            | GW1NR  | GW1NR-4, GW1NR-4B, GW1NR-4C          |
|                            | GW1NRF | GW1NRF-4B                            |

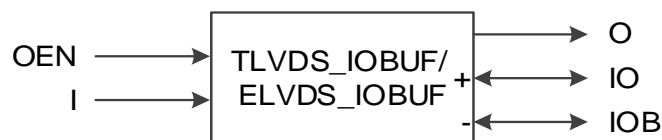
ELVDS\_IOBUF(Emulated LVDS Bi-Directional Buffer)，模拟差分双向缓冲器，当 OEN 为高电平时，作为模拟差分输入缓冲器；OEN 为低电平时，作为模拟差分输出缓冲器。

注！

GW1NZ-1 器件不支持 ELVDS\_IOBUF。

#### 端口示意图

图 3-14 TLVDS\_IOBUF/ELVDS\_IOBUF 端口示意图



#### 端口介绍

表 3-11 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。

## 适用器件

表 3-12 MIPI\_IBUF 适用器件

| 家族                         | 系列      | 器件  |
|----------------------------|---------|---|
| 小蜜蜂®<br>(LittleBee®)<br>家族 | GW1N    | GW1N-9, GW1N-9C, GW1N-2, GW1N-1P5, GW1N-2B, GW1N-1P5B |
|                            | GW1NR   | GW1NR-9, GW1NR-9C, GW1NR-2, GW1NR-2B                  |
|                            | GW1NS   | GW1NS-2, GW1NS-2C, GW1NS-4, GW1NS-4C                  |
|                            | GW1NSE  | GW1NSE-2C   |
|                            | GW1NSER | GW1NSER-4C  |
|                            | GW1NSR  | GW1NSR-2, GW1NSR-2C, GW1NSR-4, GW1NSR-4C              |
| 晨熙®<br>(Arora)             | GW2AN   | GW2AN-18X, GW2AN-9X                                   |

## 功能描述

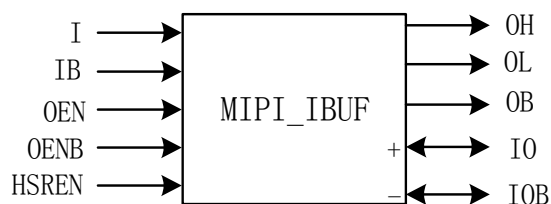
MIPI\_IBUF 支持 LP、HS 模式，IO，IOB 连接到 pad。

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。

## 端口示意图

图 3-15 MIPI\_IBUF 端口示意图



## 端口介绍

表 3-13 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             |
| 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。  |

| 端口  | I/O    | 描述  |
|-----|--------|---|
| OB  | Output | LP 模式下, OENB 高电平时 OB 为输出。   |
| IO  | Inout  | <ul style="list-style-type: none"> <li>● LP 模式下, OEN 低电平时 IO 为输出, OEN 高电平时 IO 为输入;</li> <li>● HS 模式下, IO 为输入。</li> </ul>      |
| IOB | Inout  | <ul style="list-style-type: none"> <li>● LP 模式下, OENB 低电平时 IOB 为输出, OENB 高电平时 IOB 为输入;</li> <li>● HS 模式下, IOB 为输入。</li> </ul> |

### 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),
    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;
        HSREN:IN std_logic
    );

```

```

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

```

### 3.6.10 MIPI\_OBUF

#### 原语介绍

MIPI\_OBUF 有两种工作模式：HS 模式和 LP 模式。

MIPI\_OBUF(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。

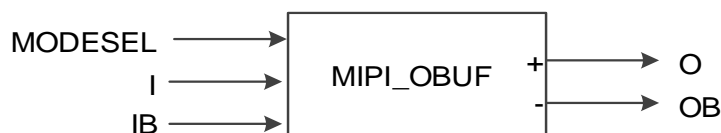
#### 适用器件

表 3-14 MIPI\_OBUF 适用器件

| 家族                         | 系列      | 器件                                       |
|----------------------------|---------|--|
| 小蜜蜂®<br>(LittleBee®)<br>家族 | GW1N    | GW1N-9, GW1N-9C                          |
|                            | GW1NR   | GW1NR-9, GW1NR-9C                        |
|                            | GW1NS   | GW1NS-2, GW1NS-2C, GW1NS-4, GW1NS-4C     |
|                            | GW1NSE  | GW1NSE-2C                                |
|                            | GW1NSER | GW1NSER-4C                               |
|                            | GW1NSR  | GW1NSR-2, GW1NSR-2C, GW1NSR-4, GW1NSR-4C |

#### 端口示意图

图 3-16 MIPI\_OBUF 端口示意图



## 端口介绍

表 3-15 MIPI\_OBUF 端口介绍

| 端口      | I/O    | 描述                                       |
|---------|--------|--|
| I       | Input  | A 端数据输入信号，可用于 HS 模式或 LP 模式。              |
| IB      | Input  | LP 模式下 B 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 uut(
    .O(O),
    .OB(OB),
    .I(I),
    .IB(IB),
    .MODESEL(MODESEL)
);

```

### Vhdl 例化:

```

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

```



MDOESEL=>MODESEL

);

### 3.6.11 MIPI\_OBUF\_A

#### 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 with IL Signal)，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。与 MIPI\_OBUF 的区别是增加了 IL 端口作为 LP 模式时 A 端输入。

#### 适用器件

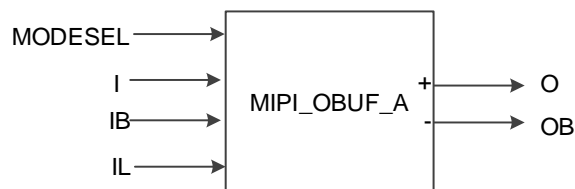
MIPI\_OBUF\_A 的适用器件除表 3-14 外，还适用于下表所列器件。

表 3-16 MIPI\_OBUF\_A 适用器件(附加)

| 家族                         | 系列    | 器件                                   |
|----------------------------|-------|--------------------------------------|
| 小蜜蜂®<br>(LittleBee®)<br>家族 | GW1N  | GW1N-2, GW1N-1P5, GW1N-2B, GW1N-1P5B |
|                            | GW1NR | GW1NR-2, GW1NR-2B                    |

#### 端口示意图

图 3-17 MIPI\_OBUF\_A 端口示意图



#### 端口介绍

表 3-17 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
    );

```

## 3.6.12 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。

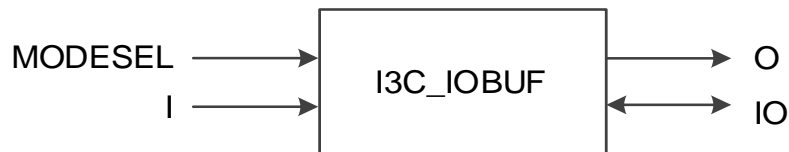
### 适用器件

表 3-18 I3C\_IOBUF 适用器件

| 家族                         | 系列      | 器件                                       |
|----------------------------|---------|--|
| 小蜜蜂®<br>(LittleBee®)<br>家族 | GW1N    | GW1N-9, GW1N-9C                          |
|                            | GW1NR   | GW1NR-9, GW1NR-9C                        |
|                            | GW1NS   | GW1NS-2, GW1NS-2C, GW1NS-4, GW1NS-4C     |
|                            | GW1NSE  | GW1NSE-2C                                |
|                            | GW1NSER | GW1NSER-4C                               |
|                            | GW1NSR  | GW1NSR-2, GW1NSR-2C, GW1NSR-4, GW1NSR-4C |

### 端口示意图

图 3-18 I3C\_IOBUF 端口示意图



### 端口介绍

表 3-19 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
    );

```

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

#### 原语介绍

MIPI\_IBUF\_HS 为差分输入实现 HS 模式，MIPI\_IBUF\_LP 通过单端输入实现 LP 模式。

#### 适用器件

表 3-20 MIPI\_IBUF\_HS/MIPI\_IBUF\_LP 适用器件

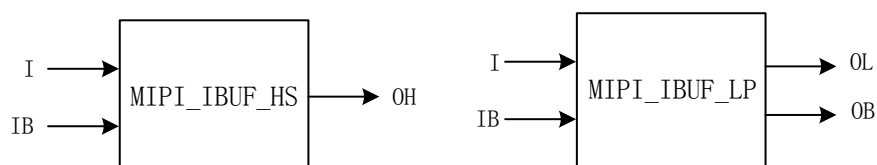
| 家族                         | 系列    | 器件      |
|----------------------------|-------|---------|
| 小蜜蜂®<br>(LittleBee®)<br>家族 | GW1NR | GW1NR-2 |

#### 功能描述

用户可使用 MIPI\_IBUF\_HS 和 MIPI\_IBUF\_LP 组合实现支持 HS、LP 模式，通过 Floorplanner 约束位置使用。MIPI\_IBUF\_HS 的输入 I 和 MIPI\_IBUF\_LP 的 I 需连接相同信号，MIPI\_IBUF\_HS 的输入 IB 和 MIPI\_IBUF\_LP 的 IB 需连接相同信号。

#### 端口示意图

图 3-19 MIPI\_IBUF\_HS/MIPI\_IBUF\_LP 端口示意图



## 端口介绍

表 3-21 MIPI\_IBUF\_HS 端口介绍

| 端口 | I/O    | 描述              |
|----|--------|-----------------|
| I  | Input  | HS 模式差分输入 A 端信号 |
| IB | Input  | HS 模式差分输入 B 端信号 |
| OH | Output | HS 模式数据输出信号     |

表 3-22 MIPI\_IBUF\_LP 端口介绍

| 端口 | I/O    | 描述              |
|----|--------|-----------------|
| I  | Input  | LP 模式 A 端单端输入信号 |
| IB | Input  | LP 模式 B 端单端输入信号 |
| OL | Output | LP 模式 A 端输出信号   |
| OB | Output | LP 模式 B 端输出信号   |

## 连接规则

- MIPI\_IBUF\_HS 的输出 OH 可以连接 Iologic（输入输出逻辑）；
- MIPI\_IBUF\_LP 的输出 OL 和 OB 不允许连接 Iologic。

## 原语例化

### Verilog 例化:

```

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

```

### Vhdl 例化:

```

COMPONENT MIPI_IBUF_HS
PORT (
    OH:OUT std_logic;

```

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

# 4 输入输出逻辑

高云半导体 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)。

注!

- GW1N-1、GW1NR-1、GW1NZ-1、GW1NS-2、GW1NS-2C、GW1NSR-2C、GW1NSR-2、GW1NSE-2C 器件 IOL6、IOR6 管脚不支持 IO 逻辑；
- GW1N-2、GW1NR-2、GW1N-1P5、GW1N-2B、GW1N-1P5B、GW1NR-2B 器件的 IOT2、IOT3A 管脚不支持 IO 逻辑；
- GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1N-4C、GW1NR-4C 器件的 IOL10、IOR10 管脚不支持 IO 逻辑。

图 4-1 为高云半导体 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

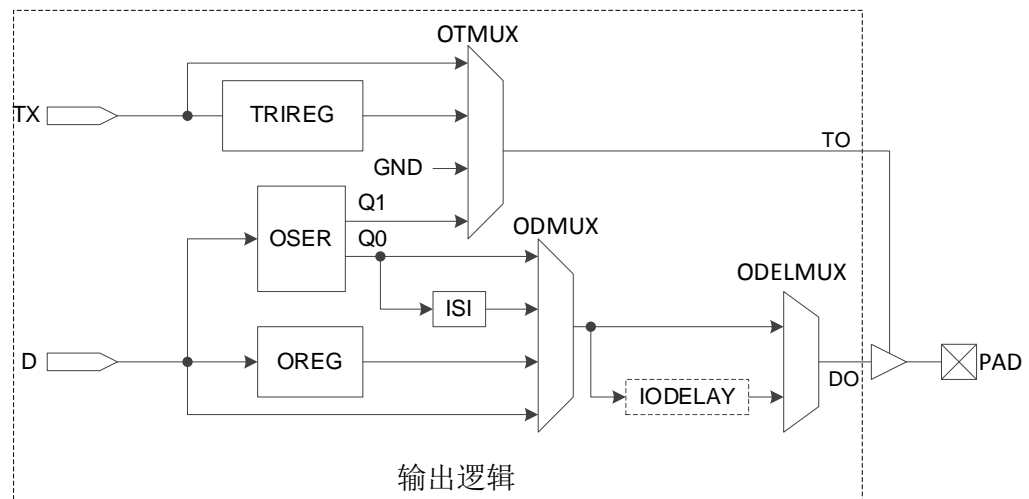
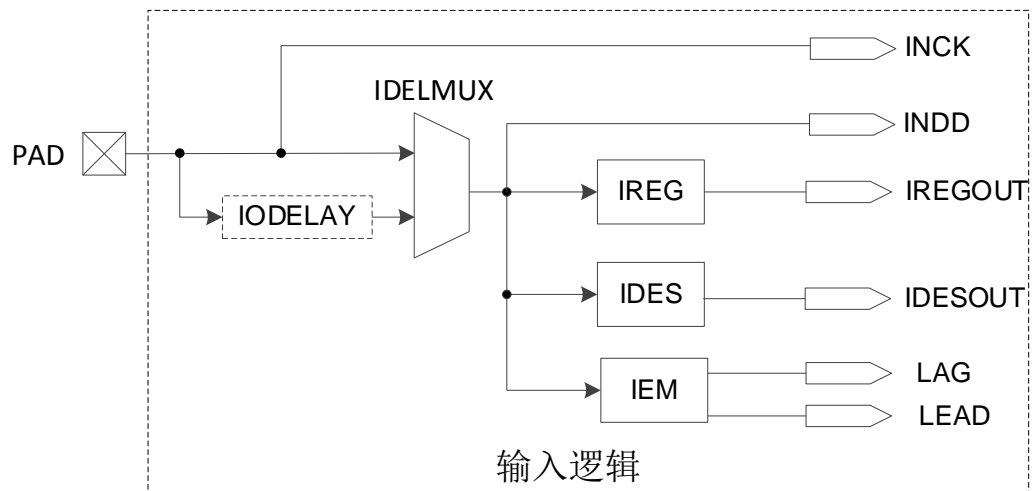


图 4-2 为高云半导体 FPGA 产品的输入输出逻辑的输入部分。

图 4-2 输入输出逻辑输入示意图 – 输入部分



## 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。

## 4.2 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 4-3 所示，时序图如图 4-4 所示。

图 4-3 IDDR 逻辑框图

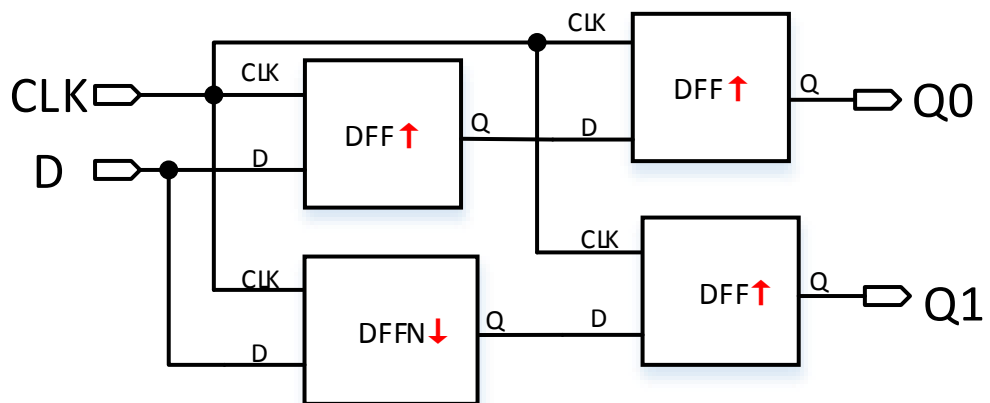
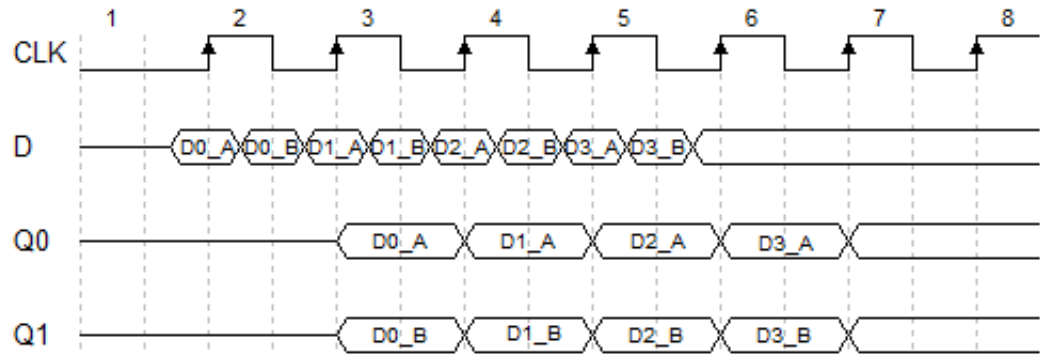




图 4-4 IDDR 时序图



### 端口示意图

图 4-5 IDDR 端口示意图



### 端口介绍

表 4-1 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 |

### 参数介绍

表 4-2 IDDR 参数介绍

| 参数名     | 取值范围 | 默认值  | 描述         |
|---------|------|------|------------|
| Q0_INIT | 1'b0 | 1'b0 | Q0 输出的初始取值 |
| Q1_INIT | 1'b0 | 1'b0 | Q1 输出的初始取值 |

### 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，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

#### Verilog 例化:

```

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

```

**Vhdl 例化:**

```

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

```

**4.2.2 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 4-6 IDDRC 端口示意图



## 端口介绍

表 4-3 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   |

## 参数介绍

表 4-4 IDDRC 参数介绍

| 参数名     | 取值范围 | 默认值  | 描述         |
|---------|------|------|------------|
| Q0_INIT | 1'b0 | 1'b0 | Q0 输出的初始取值 |
| Q1_INIT | 1'b0 | 1'b0 | Q1 输出的初始取值 |

## 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，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

### Verilog 例化:

```

IDDRC uut(
    .Q0(Q0),
    .Q1(Q1),
    .D(D),
    .CLK(CLK),

```

```

        .CLEAR(CLEAR)
    );
    defparam uut.Q0_INIT = 1'b0;
    defparam uut.Q1_INIT = 1'b0;
Vhdl 例化:
    COMPONENT IDDRC
        GENERIC (Q0_INIT:bit:='0';
                Q1_INIT:bit:='0'
        );
        PORT(
            Q0:OUT std_logic;
            Q1:OUT std_logic;
            D:IN std_logic;
            CLEAR:IN std_logic;
            CLK:IN std_logic
        );
    END COMPONENT;
    uut:IDDRC
        GENERIC MAP (Q0_INIT=>'0',
                    Q1_INIT=>'0'
        )
        PORT MAP (
            Q0=>Q0,
            Q1=>Q1,
            D=>D,
            CLEAR=>CLEAR,
            CLK=>CLK
        );

```

### 4.2.3 IDES4

#### 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 4-7 所示。



## 参数介绍

表 4-6 IDES4 参数介绍

| 参数名   | 取值范围            | 默认值     | 描述           |
|-------|-----------------|---------|--------------|
| GSREN | "false", "true" | "false" | 启用全局复位 GSR   |
| LSREN | "false", "true" | "true"  | 启用本地复位 RESET |

## 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，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

### Verilog 例化:

```

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

```

### Vhdl 例化:

```

COMPONENT IDES4
    GENERIC (GSREN:string:="false";
            LSREN:string:="true"
    );
    PORT(
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        Q2:OUT std_logic;

```

```

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

```

## 4.2.4 IDES8

### 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。

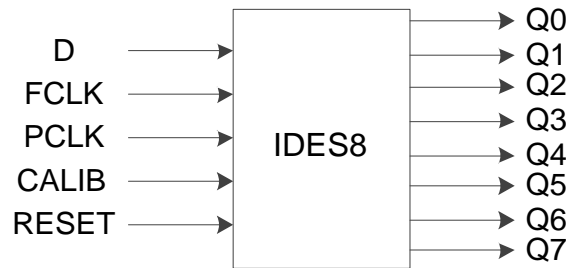
### 功能描述

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 4-9 IDES8 端口示意图



## 端口介绍

表 4-7 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                    |

## 参数介绍

表 4-8 IDES8 参数介绍

| 参数名   | 取值范围            | 默认值     | 描述           |
|-------|-----------------|---------|--------------|
| GSREN | "false", "true" | "false" | 启用全局复位 GSR   |
| LSREN | "false", "true" | "true"  | 启用本地复位 RESET |

## 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, 也可以通过 IP Core Generator 工具产生, 具体可参考 5 IP 调用。

### 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)
    );
    defparam uut.GSREN="false";
    defparam uut.LSREN="true";

```

**Vhdl 例化:**

```

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

```

```
        GENERIC MAP (GSREN=>"false",
                    LSREN=>"true"
        )
        PORT MAP (
            Q0=>Q0,
            Q1=>Q1,
            Q2=>Q2,
            Q3=>Q3,
            Q4=>Q4,
            Q5=>Q5,
            Q6=>Q6,
            Q7=>Q7,
            D=>D,
            FCLK=>FCLK,
            PCLK=>PCLK,
            CALIB=>CALIB,
            RESET=>RESET
        );
```

## 4.2.5 IDES10

### 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。

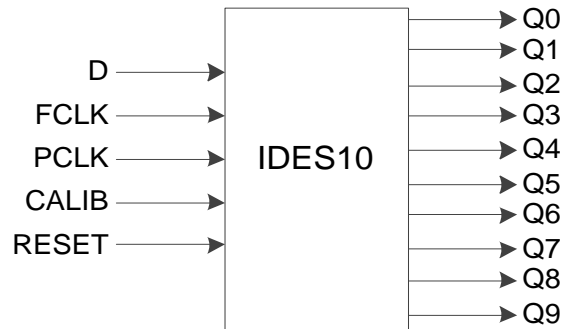
### 功能描述

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 4-10 IDES10 端口示意图



## 端口介绍

表 4-9 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              |

## 参数介绍

表 4-10 IDES10 参数介绍

| 参数名   | 取值范围            | 默认值     | 描述           |
|-------|-----------------|---------|--------------|
| GSREN | "false", "true" | "false" | 启用全局复位 GSR   |
| LSREN | "false", "true" | "true"  | 启用本地复位 RESET |

## 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，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

### 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)  
);  
defparam uut.GSREN="false";  
defparam uut.LSREN = "true";
```

**Vhdl 例化:**

```
COMPONENT IDES10  
    GENERIC (GSREN:string:="false";  
            LSREN:string:="true"  
    );  
    PORT(  
        Q0:OUT std_logic;  
        Q1:OUT std_logic;  
        Q2:OUT std_logic;  
        Q3:OUT std_logic;  
        Q4:OUT std_logic;  
        Q5:OUT std_logic;  
        Q6:OUT std_logic;  
        Q7:OUT std_logic;  
        Q8:OUT std_logic;  
        Q9:OUT std_logic;  
        D:IN std_logic;  
        FCLK:IN std_logic;
```

```

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

```

## 4.2.6 IVIDEO

### 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 4-11 IVIDEO 端口示意图

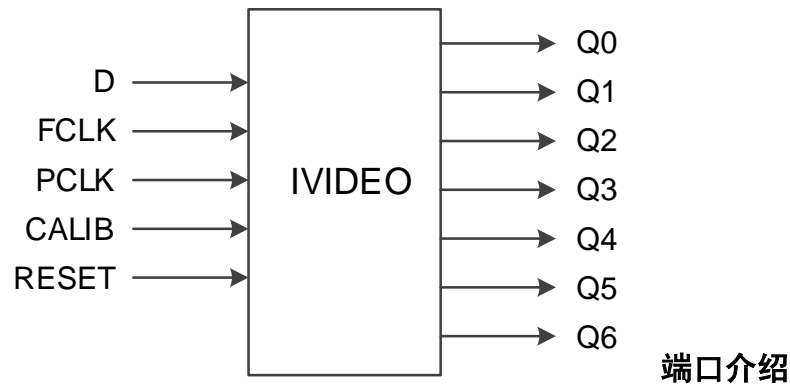


表 4-11 IVIDEO 端口介绍

| 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                    |
| 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              |

## 参数介绍

表 4-12 IVIDEO 参数介绍

| 参数名   | 取值范围            | 默认值     | 描述           |
|-------|-----------------|---------|--------------|
| GSREN | "false", "true" | "false" | 启用全局复位 GSR   |
| LSREN | "false", "true" | "true"  | 启用本地复位 RESET |

## 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，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

### 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)
    );
    defparam uut.GSREN="false";
    defparam uut.LSREN="true";

```

**Vhdl 例化:**

```

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

```

```

)
PORT MAP (
    Q0=>Q0,
    Q1=>Q1,
    Q2=>Q2,
    Q3=>Q3,
    Q4=>Q4,
    Q5=>Q5,
    Q6=>Q6,
    D=>D,
    FCLK=>FCLK,
    PCLK=>PCLK,
    CALIB=>CALIB,
    RESET=>RESET
);

```

## 4.2.7 IDES16

### 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。

### 适用器件

表 4-13 IDES16 适用器件

| 家族                         | 系列      | 器件   |
|----------------------------|---------|--|
| 小蜜蜂®<br>(LittleBee®)<br>家族 | GW1N    | GW1N-1S, GW1N-9, GW1N-9C, GW1N-2, GW1N-1P5, GW1N-2B, GW1N-1P5B |
|                            | GW1NR   | GW1NR-9, GW1NR-9C, GW1NR-2, GW1NR-2B                           |
|                            | GW1NS   | GW1NS-2, GW1NS-2C, GW1NS-4, GW1NS-4C                           |
|                            | GW1NSE  | GW1NSE-2C  |
|                            | GW1NSER | GW1NSER-4C   |
|                            | GW1NSR  | GW1NSR-2, GW1NSR-2C, GW1NSR-4, GW1NSR-4C                       |

### 功能描述

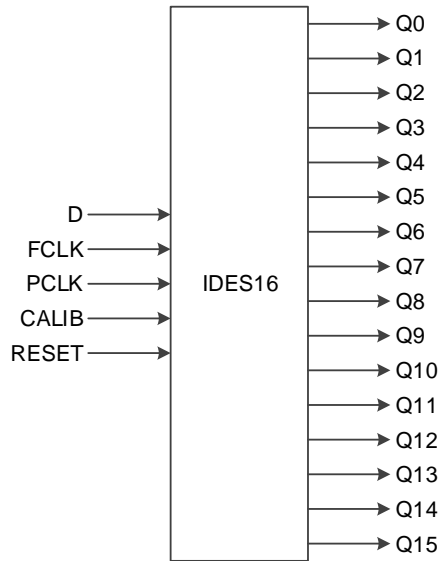
IDES16 模式，实现 1: 16 串并转换，输出数据在同一时钟边沿提供给 FPGA 逻辑。支持 CALIB 调整输出数据顺序，每个脉冲数据移位一位，移位十六次后，数据输出将与移位前的数据相同。

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



## 端口示意图

图 4-12 IDES16 端口示意图



## 端口介绍

表 4-14 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              |

## 参数介绍

表 4-15 IDES16 参数介绍

| 参数名   | 取值范围            | 默认值     | 描述           |
|-------|-----------------|---------|--------------|
| GSREN | "false", "true" | "false" | 启用全局复位 GSR   |
| LSREN | "false", "true" | "true"  | 启用本地复位 RESET |

## 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，也可以通过 IP Core Generator 工具产生，具体

可参考 5 IP 调用。

**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)  
);  
defparam uut.GSREN="false";  
defparam uut.LSREN="true";
```

**Vhdl 例化:**

```
COMPONENT IDES16  
    GENERIC (GSREN:string:="false";  
            LSREN:string:="true"  
    );  
    PORT(  
        Q0:OUT std_logic;  
        Q1:OUT std_logic;
```

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

```

Q9=>Q9,
Q10=>Q10,
Q11=>Q11,
Q12=>Q12,
Q13=>Q13,
Q14=>Q14,
Q15=>Q15,
D=>D,
FCLK=>FCLK,
PCLK=>PCLK,
CALIB=>CALIB,
RESET=>RESET

```

);

## 4.2.8 IDDR\_MEM

### 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。

### 适用器件

表 4-16 IDDR\_MEM 适用器件

| 家族                   | 系列     | 器件                                   |
|----------------------|--------|--------------------------------------|
| 晨熙®<br>(Arora)<br>家族 | GW2A   | GW2A-18, GW2A-18C, GW2A-55, GW2A-55C |
|                      | GW2AN  | GW2AN-55C                            |
|                      | GW2AR  | GW2AR-18, GW2AR-18C                  |
|                      | GW2ANR | GW2ANR-18C                           |

### 功能描述

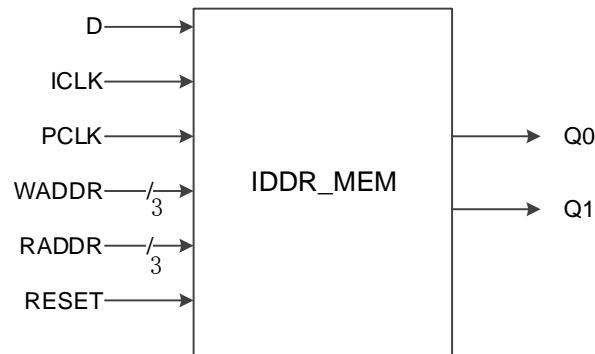
IDDR\_MEM 输出数据在同一时钟边沿提供给 FPGA 逻辑。IDDR\_MEM 需要配合 DQS 使用, 其中, ICLK 连接 DQS 的输出信号 DQSR90, 且根据 ICLK 的时钟沿将数据送入 IDDR\_MEM; WADDR[2:0]连接 DQS 的输出信号 WPOINT; RADDR[2:0]连接 DQS 的输出信号 RPOINT。

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 4-13 IDDR\_MEM 端口示意图



## 端口介绍

表 4-17 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           |

## 参数介绍

表 4-18 IDDR\_MEM 参数介绍

| 参数名   | 取值范围            | 默认值     | 描述           |
|-------|-----------------|---------|--------------|
| GSREN | "false", "true" | "false" | 启用全局复位 GSR   |
| LSREN | "false", "true" | "true"  | 启用本地复位 RESET |

## 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)
);
defparam uut.GSREN="false";
defparam uut.LSREN ="true";

```

**Vhdl 例化:**

```

COMPONENT IDDR_MEM
    GENERIC (GSREN:string:="false";
            LSREN:string:="true"
    );
    PORT(
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        D:IN std_logic;
        ICLK:IN std_logic;
        PCLK:IN std_logic;
        WADDR:IN std_logic_vector(2 downto 0);
        RADDR:IN std_logic_vector(2 downto 0);
        RESET:IN std_logic
    );
END COMPONENT;
uut:IDDR_MEM
    GENERIC MAP (GSREN=>"false",
                LSREN=>"true"
    )
    PORT MAP (
        Q0=>q0,
        Q1=>q1,

```

```

D=>d,
ICLK=>iclk,
PCLK=>pclk,
WADDR=>waddr,
RADDR=>raddr,
RESET=>reset

```

```
);
```

## 4.2.9 IDES4\_MEM

### 原语介绍

IDES4\_MEM(4 to 1 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。

### 适用器件

表 4-19 IDES4\_MEM 适用器件

| 家族                   | 系列     | 器件                                   |
|----------------------|--------|--------------------------------------|
| 晨熙®<br>(Arora)<br>家族 | GW2A   | GW2A-18, GW2A-18C, GW2A-55, GW2A-55C |
|                      | GW2AN  | GW2AN-55C                            |
|                      | GW2AR  | GW2AR-18, GW2AR-18C                  |
|                      | GW2ANR | GW2ANR-18C                           |

### 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。

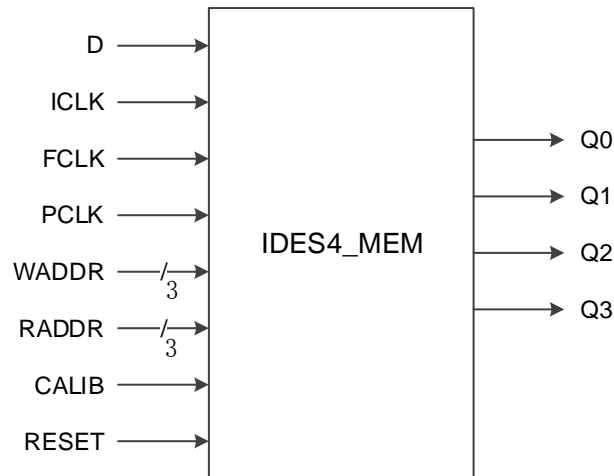
IDES4\_MEM 与 IDES4 不同，IDES4\_MEM 需要配合 DQS 使用，其中，ICLK 连接 DQS 的输出信号 DQSR90，且根据 ICLK 的时钟沿将数据送入 IDES4\_MEM；WADDR[2:0]连接 DQS 的输出信号 WPOINT；RADDR[2:0]连接 DQS 的输出信号 RPOINT。

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 4-14 IDES4\_MEM 端口示意图



## 端口介绍

表 4-20 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           |

## 参数介绍

表 4-21 IDES4\_MEM 参数介绍

| 参数名   | 取值范围            | 默认值     | 描述           |
|-------|-----------------|---------|--------------|
| GSREN | "false", "true" | "false" | 启用全局复位 GSR   |
| LSREN | "false", "true" | "true"  | 启用本地复位 RESET |

## 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)
);
defparam uut.GSREN="false";
defparam uut.LSREN="true";

```

#### Vhdl 例化:

```

COMPONENT IDES4_MEM
    GENERIC (GSREN:string:="false";
            LSREN:string:="true"
    );
    PORT(
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        Q2:OUT std_logic;
        Q3:OUT std_logic;
        D:IN std_logic;
        ICLK:IN std_logic;
        FCLK:IN std_logic;
        PCLK:IN std_logic;
        WADDR:IN std_logic_vector(2 downto 0);

```

```

        RADDR:IN std_logic_vector(2 downto 0);
        CALIB:IN std_logic;
        RESET:IN std_logic
    );
END COMPONENT;
 uut:IDES4_MEM
    GENERIC MAP (GSREN=>"false",
                 LSREN=>"true"
    )
    PORT MAP (
        Q0=>q0,
        Q1=>q1,
        Q2=>q2,
        Q3=>q3,
        D=>d,
        ICLK=>iclk,
        FCLK=>fclk,
        PCLK=>pclk,
        WADDR=>waddr,
        RADDR=>raddr,
        CALIB=>calib,
        RESET=>reset
    );

```

#### 4.2.10 IDES8\_MEM

##### 原语介绍

IDES8\_MEM(8 to 1 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。

##### 适用器件

表 4-22 IDES8\_MEM 适用器件

| 家族               | 系列     | 器件                                   |
|------------------|--------|--------------------------------------|
| 晨熙®(Arora)<br>家族 | GW2A   | GW2A-18, GW2A-18C, GW2A-55, GW2A-55C |
|                  | GW2AN  | GW2AN-55C                            |
|                  | GW2AR  | GW2AR-18, GW2AR-18C                  |
|                  | GW2ANR | GW2ANR-18C                           |

## 功能描述

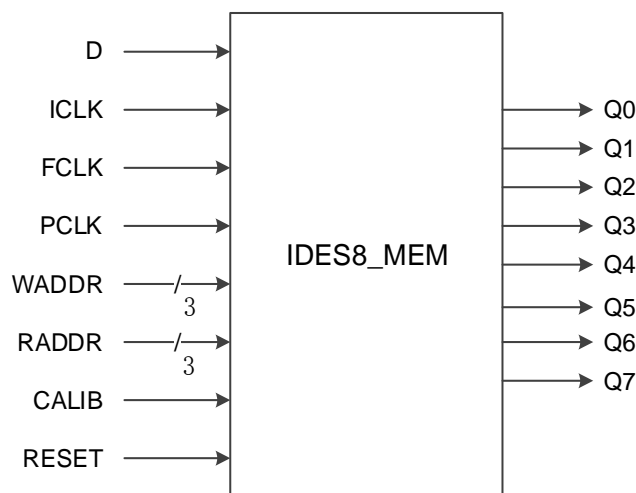
IDES8\_MEM 实现 1: 8 串并转换，输出数据在同一时钟边沿提供给 FPGA 逻辑。支持 CALIB 调整输出数据顺序，每个脉冲数据移位一位，移位八次后，数据输出将与移位前的数据相同。与 IDES8 不同，IDES8\_MEM 需要配合 DQS 使用，其中，ICLK 连接 DQS 的输出信号 DQSR90，且根据 ICLK 的时钟沿将数据送入 IDES8\_MEM；WADDR[2:0]连接 DQS 的输出信号 WPOINT；RADDR[2:0]连接 DQS 的输出信号 RPOINT。

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 4-15 IDES8\_MEM 端口示意图



## 端口介绍

表 4-23 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            |
| Q7~Q0      | Output | IDES8_MEM 数据输出信号          |

## 参数介绍

表 4-24 IDES8\_MEM 参数介绍

| 参数名   | 取值范围            | 默认值     | 描述           |
|-------|-----------------|---------|--------------|
| GSREN | "false", "true" | "false" | 启用全局复位 GSR   |
| LSREN | "false", "true" | "true"  | 启用本地复位 RESET |

## 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)
);
defparam uut.GSREN="false";
defparam uut.LSREN="true";

```

### Vhdl 例化:

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

Q7=>q7,  
 D=>d,  
 ICLK=>iclk,  
 FCLK=>fclk,  
 PCLK=>pclk,  
 WADDR=>waddr,  
 RADDR=>raddr,  
 CALIB=>calib,  
 RESET=>reset

);

### 4.3 DDR 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 4-16 所示, 时序图如图 4-17 所示。

图 4-16 ODDR 逻辑框图

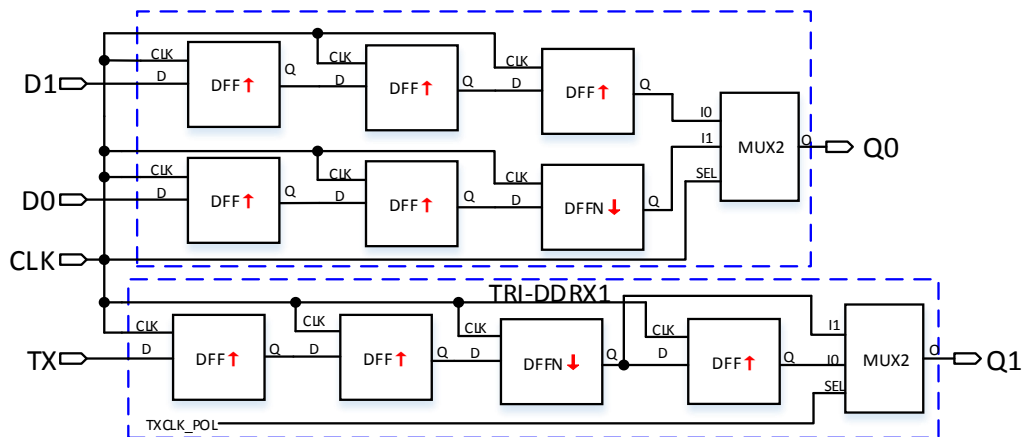
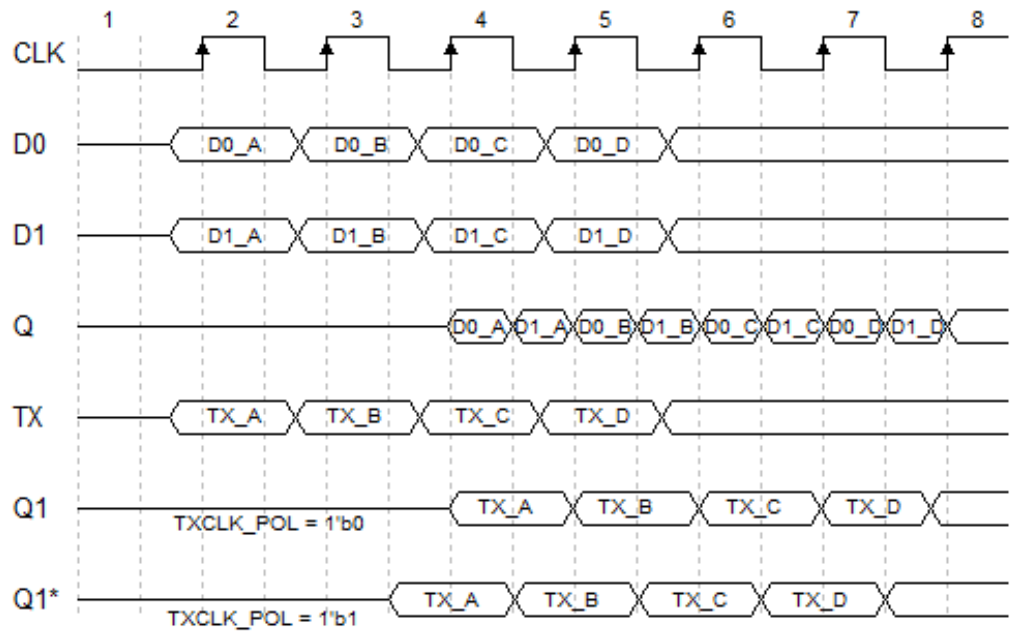


图 4-17 ODDR 时序图



端口示意图

图 4-18 ODDR 端口示意图



端口介绍

表 4-25 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 4-26 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<br>1'b0:Q1 上升沿输出;<br>1'b1:Q1 下降沿输出 |

| 参数名  | 取值范围 | 默认值  | 描述           |
|------|------|------|--------------|
| INIT | 1'b0 | 1'b0 | ODDR 输出的初始取值 |

### 连接规则

- Q0 可直接连接 OBUF，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，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

#### Verilog 例化:

```

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

```

#### Vhdl 例化:

```

COMPONENT ODDR
    GENERIC (CONSTANT INIT: std_logic='0';
             TXCLK_POL:bit='0'
    );
    PORT(
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        D0:IN std_logic;
        D1:IN std_logic;
        TX:IN std_logic;
        CLK:IN std_logic
    );
END COMPONENT;

```



```

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

```

## 4.3.2 ODDRC

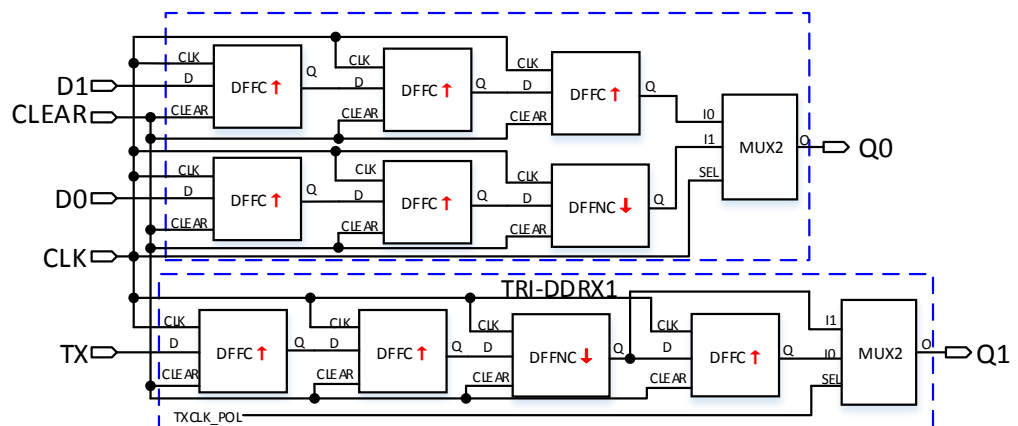
### 原语介绍

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

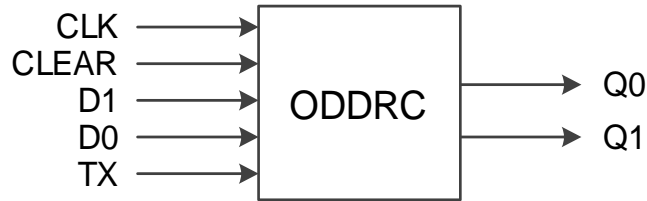
ODDRC 模式，用于从 FPGA 器件传输双倍数据速率信号。其中 Q0 为双倍速率数据输出，Q1 用于 Q0 所连的 IOBUF/TBUF 的 OEN 信号。其逻辑框图如图 4-19 所示。

图 4-19 ODDRC 逻辑框图



## 端口示意图

图 4-20 ODDRC 端口示意图



## 端口介绍

表 4-27 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 4-28 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<br>1'b0:Q1 上升沿输出;<br>1'b1:Q1 下降沿输出 |
| INIT      | 1'b0       | 1'b0 | ODDRC 输出的初始取值                                  |

## 连接规则

- Q0 可直接连接 OBUF，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，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

### Verilog 例化:

```
ODDRC uut(
    .Q0(Q0),
```

```

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

```

**Vhdl 例化:**

```

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

```

CLEAR=>CLEAR

);

### 4.3.3 OSER4

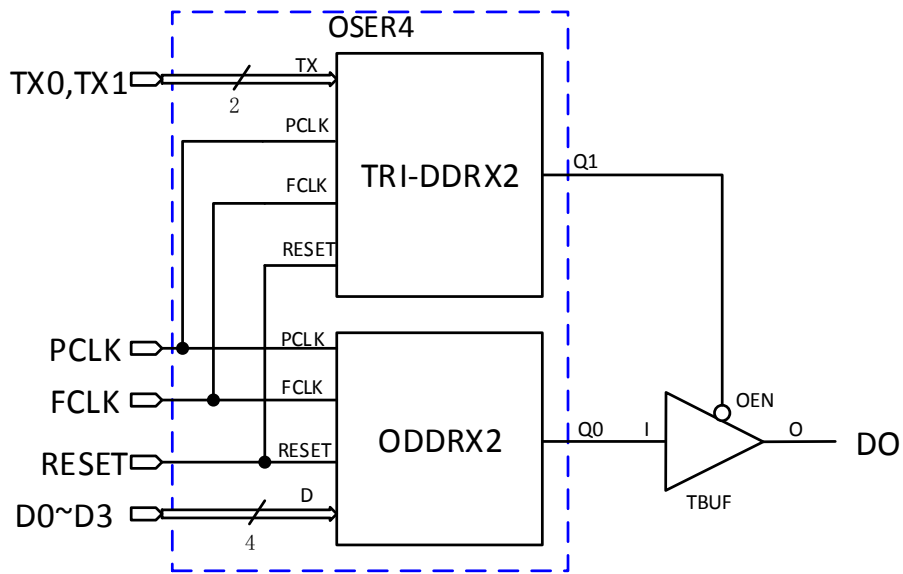
#### 原语介绍

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。逻辑框图如图 4-21 所示。

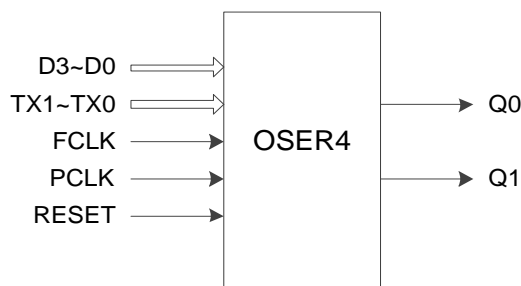
图 4-21 OSER4 逻辑框图



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

#### 端口示意图

图 4-22 OSER4 端口示意图



#### 端口介绍

表 4-29 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 |

| 端口名   | I/O    | 描述   |
|-------|--------|--|
| 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 4-30 OSER4 参数介绍

| 参数名       | 取值范围            | 默认值     | 描述  |
|-----------|-----------------|---------|---|
| GSREN     | "false", "true" | "false" | 启用全局复位 GSR  |
| LSREN     | "false", "true" | "true"  | 启用本地复位 RESET  |
| TXCLK_POL | 1'b0, 1'b1      | 1'b0    | Q1 输出时钟极性控制<br>1'b0:数据上升沿输出;<br>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<br>"false": d_up1 比 d_up0 提前一个周期;<br>"true": d_up1 和 d_up0 时序相同 |

## 连接规则

- Q0 可直接连接 OBUF，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，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

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

```

**Vhdl 例化:**

```

COMPONENT OSER4
    GENERIC (GSREN:string:="false";
             LSREN:string:="true";
             HWL:string:="false";
             TXCLK_POL:bit:='0'
    );
    PORT(
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        D0:IN std_logic;
        D1:IN std_logic;
        D2:IN std_logic;
        D3:IN std_logic;
        TX0:IN std_logic;
        TX1:IN std_logic;
        FCLK:IN std_logic;
        PCLK:IN std_logic;
        RESET:IN std_logic
    );
END COMPONENT;
uut:OSER4
    GENERIC MAP (GSREN=>"false",
                LSREN=>"true",
                HWL=>"false",
                TXCLK_POL=>'0'

```

```

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

```

### 4.3.4 OSER8

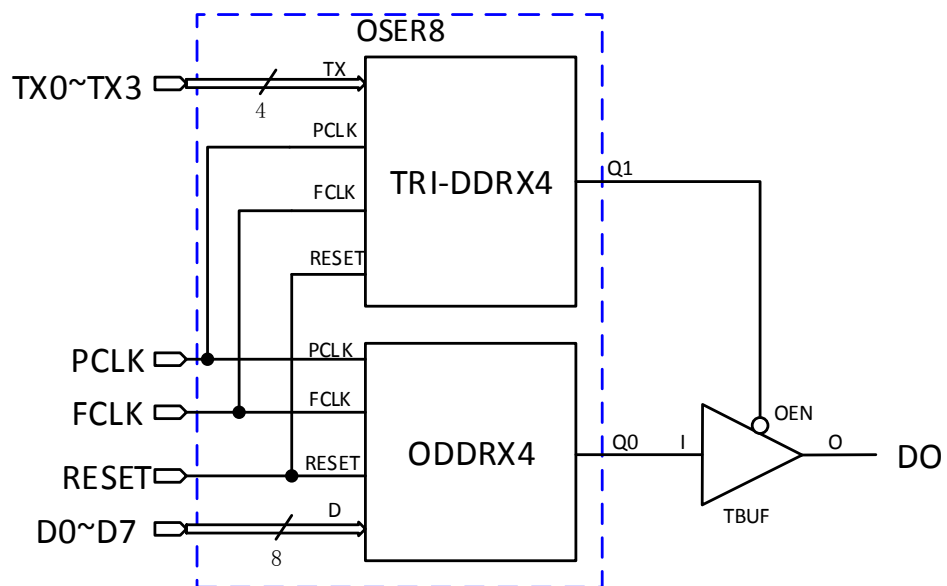
#### 原语介绍

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，Q1 用于 Q0 所连的 IOBUF/TBUF 的 OEN 信号。逻辑框图如图 4-23 所示。

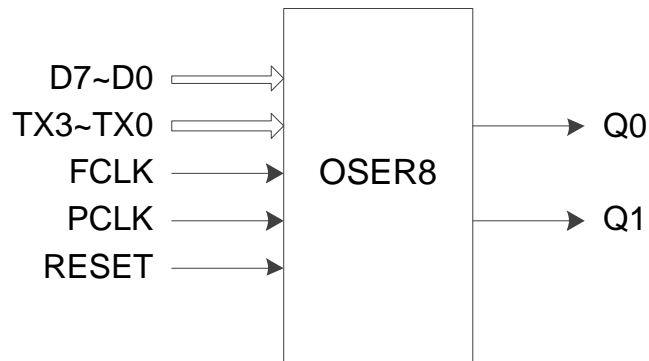
图 4-23 OSER8 逻辑框图



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

## 端口示意图

图 4-24 OSER8 端口示意图



## 端口介绍

表 4-31 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 4-32 OSER8 参数介绍

| 参数名       | 取值范围            | 默认值     | 描述   |
|-----------|-----------------|---------|--|
| GSREN     | "false", "true" | "false" | 启用全局复位 GSR   |
| LSREN     | "false", "true" | "true"  | 启用本地复位 RESET   |
| TXCLK_POL | 1'b0, 1'b1      | 1'b0    | Q1 输出时钟极性控制 <ul style="list-style-type: none"> <li>● 1'b0:数据上升沿输出;</li> <li>● 1'b1:数据下降沿输出</li> </ul>  |
| HWL       | "false", "true" | "false" | OSER8 数据 d_up0/1 时序关系控制 <ul style="list-style-type: none"> <li>● "false": d_up1 比 d_up0 提前一个周期;</li> <li>● "true": d_up1 和 d_up0 时序相同</li> </ul> |

## 连接规则

- Q0 可直接连接 OBUF，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，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

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

### Vhdl 例化:

```
COMPONENT OSER8
    GENERIC (GSREN:string:="false";
             LSREN:string:="true";
             HWL:string:="false";
             TXCLK_POL:bit:='0'
    );
    PORT(
```

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

```

D7=>D7,
TX0=>TX0,
TX1=>TX1,
TX2=>TX2,
TX3=>TX3,
FCLK=>FCLK,
PCLK=>PCLK,
RESET=>RESET

```

);

### 4.3.5 OSER10

#### 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。

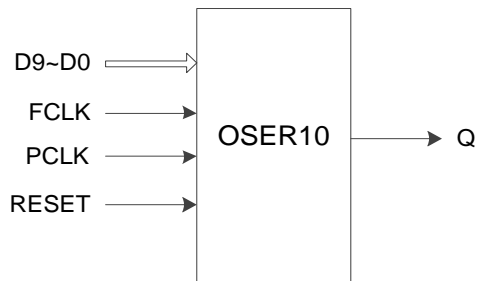
#### 功能描述

OSER10 模式，实现 10:1 并串转换。PCLK 通常由 FCLK 分频获得，

$$f_{PCLK} = 1/5 f_{FCLK}。$$

#### 端口示意图

图 4-25 OSER10 端口示意图



#### 端口介绍

表 4-33 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  |

## 参数介绍

表 4-34 OSER10 参数介绍

| 参数名   | 取值范围            | 默认值     | 描述           |
|-------|-----------------|---------|--------------|
| GSREN | "false", "true" | "false" | 启用全局复位 GSR   |
| LSREN | "false", "true" | "true"  | 启用本地复位 RESET |

## 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，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

### 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)
);
defparam uut.GSREN="false";
defparam uut.LSREN="true";
```

### Vhdl 例化:

```
COMPONENT OSER10
    GENERIC (GSREN:string:="false";
            LSREN:string:="true"
```

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

```

PCLK=>PCLK,
RESET=>RESET
);

```

### 4.3.6 OVIDEO

#### 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。

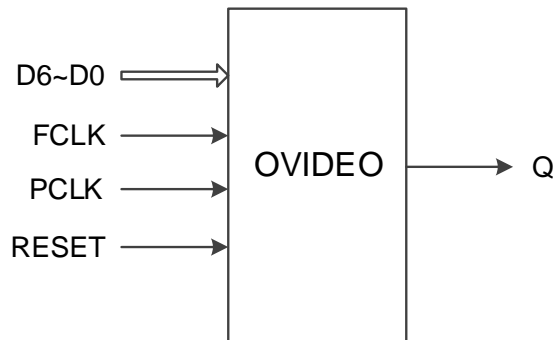
#### 功能描述

OVIDEO 模式，实现 7:1 并串转换。PCLK 通常由 FCLK 分频获得：

$$f_{PCLK} = 1/3.5 f_{FCLK}。$$

#### 端口示意图

图 4-26 OVIDEO 端口示意图



#### 端口介绍

表 4-35 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  |

#### 参数介绍

表 4-36 OVIDEO 参数介绍

| 参数名   | 取值范围            | 默认值     | 描述           |
|-------|-----------------|---------|--------------|
| GSREN | "false", "true" | "false" | 启用全局复位 GSR   |
| LSREN | "false", "true" | "true"  | 启用本地复位 RESET |

## 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，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

### 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)  
);  
defparam uut.GSREN="false";  
defparam uut.LSREN="true";
```

### Vhdl 例化:

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

```

        FCLK:IN std_logic;
        PCLK:IN std_logic;
        RESET:IN std_logic

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

```

### 4.3.7 OSER16

#### 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。

#### 适用器件

表 4-37 OSER16 适用器件

| 家族                         | 系列      | 器件   |
|----------------------------|---------|--|
| 小蜜蜂®<br>(LittleBee®)<br>家族 | GW1N    | GW1N-1S, GW1N-9, GW1N-9C, GW1N-2, GW1N-1P5, GW1N-2B, GW1N-1P5B |
|                            | GW1NR   | GW1NR-9, GW1NR-9C, GW1NR-2, GW1NR-2B                           |
|                            | GW1NS   | GW1NS-2, GW1NS-2C, GW1NS-4, GW1NS-4C                           |
|                            | GW1NSE  | GW1NSE-2C  |
|                            | GW1NSER | GW1NSER-4C   |
|                            | GW1NSR  | GW1NSR-2, GW1NSR-2C, GW1NSR-4, GW1NSR-4C                       |



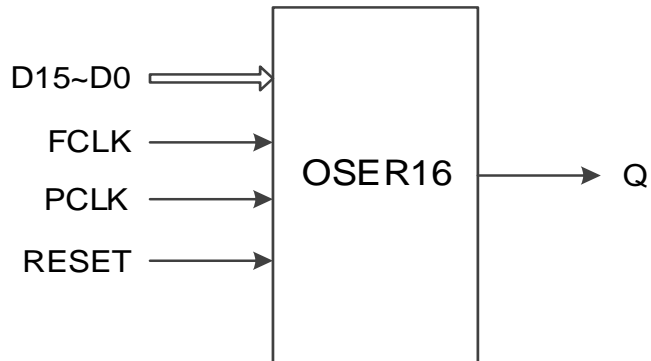
### 功能描述

OSER16 模式，实现 16:1 并串转换。PCLK 通常由 FCLK 分频获得：

$$f_{PCLK} = 1/8 f_{FCLK}。$$

### 端口示意图

图 4-27 OSER16 端口示意图



### 端口介绍

表 4-38 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  |

### 参数介绍

表 4-39 OSER16 参数介绍

| 参数名   | 取值范围            | 默认值     | 描述           |
|-------|-----------------|---------|--------------|
| GSREN | "false", "true" | "false" | 启用全局复位 GSR   |
| LSREN | "false", "true" | "true"  | 启用本地复位 RESET |

### 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，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

#### 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)  
);  
defparam uut.GSREN="false";  
defparam uut.LSREN="true";
```

**Vhdl 例化:**

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

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

D13=>D13,  
 D14=>D14,  
 D15=>D15,  
 FCLK=>FCLK,  
 PCLK=>PCLK,  
 RESET=>RESET

);

### 4.3.8 ODDR\_MEM

#### 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。

#### 适用器件

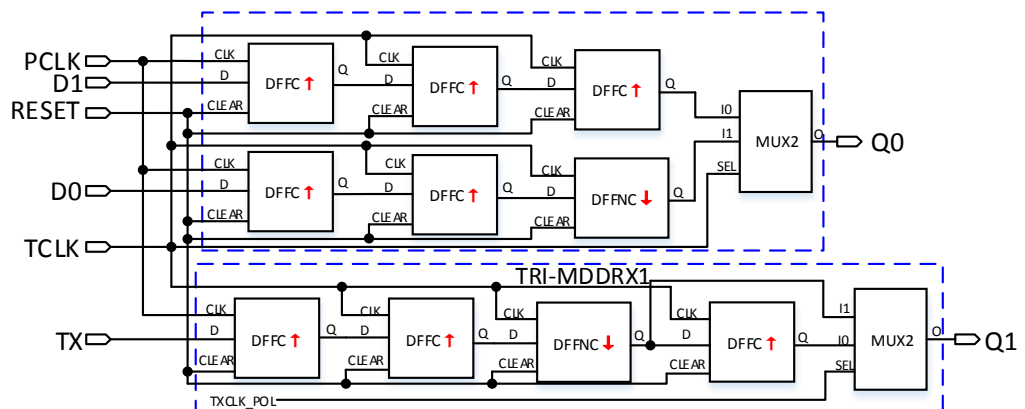
表 4-40 ODDR\_MEM 适用器件

| 家族          | 系列     | 器件                                   |
|-------------|--------|--------------------------------------|
| 晨熙® (Arora) | GW2A   | GW2A-18, GW2A-18C, GW2A-55, GW2A-55C |
|             | GW2AN  | GW2AN-55C                            |
|             | GW2AR  | GW2AR-18, GW2AR-18C                  |
|             | GW2ANR | GW2ANR-18C                           |

#### 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 4-28 所示。

图 4-28 ODDR\_MEM 逻辑框图

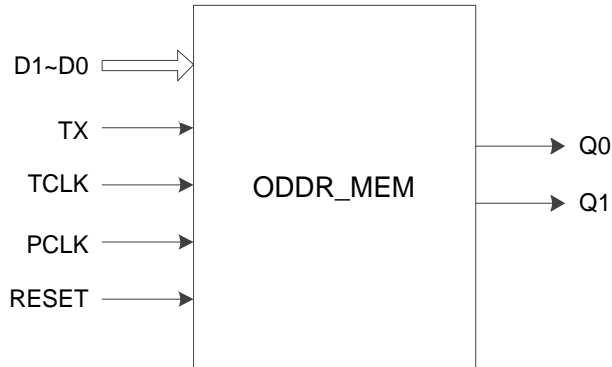


PCLK 和 TCLK 的频率关系为:  $f_{PCLK} = f_{TCLK}$ 。

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 4-29 ODDR\_MEM 端口示意图



### 端口介绍

表 4-41 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-MDDR1 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 4-42 ODDR\_MEM 参数介绍

| 参数名         | 取值范围              | 默认值     | 描述  |
|-------------|-------------------|---------|---|
| GSREN       | "false", "true"   | "false" | 启用全局复位 GSR  |
| LSREN       | "false", "true"   | "true"  | 启用本地复位 RESET  |
| TXCLK_POL   | 1'b0, 1'b1        | 1'b0    | Q1 输出时钟极性控制 <ul style="list-style-type: none"> <li>● 1'b0:数据上升沿输出;</li> <li>● 1'b1:数据下降沿输出</li> </ul>                   |
| TCLK_SOURCE | "DQSW", "DQSW270" | "DQSW"  | TCLK 来源选择 <ul style="list-style-type: none"> <li>● "DQSW": 来自 DQS 模块的 DQSW0;</li> <li>● "DQSW270": 来自 DQS 模块</li> </ul> |

| 参数名 | 取值范围 | 默认值 | 描述        |
|-----|------|-----|-----------|
|     |      |     | 的 DQSW270 |

### 连接规则

- Q0 可直接连接 OBUF，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.GSREN="false";
defparam uut.LSREN="true";
defparam uut.TCLK_SOURCE="DQSW";
defparam uut.TXCLK_POL=1'b0;

```

#### Vhdl 例化：

```

COMPONENT ODDR_MEM
    GENERIC (GSREN:string:="false";
             LSREN:string:="true";
             TXCLK_POL:bit:='0';
             TCLK_SOURCE:string:="DQSW"
    );
    PORT(
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        D0:IN std_logic;

```

```

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

```

### 4.3.9 OSER4\_MEM

#### 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。

#### 适用器件

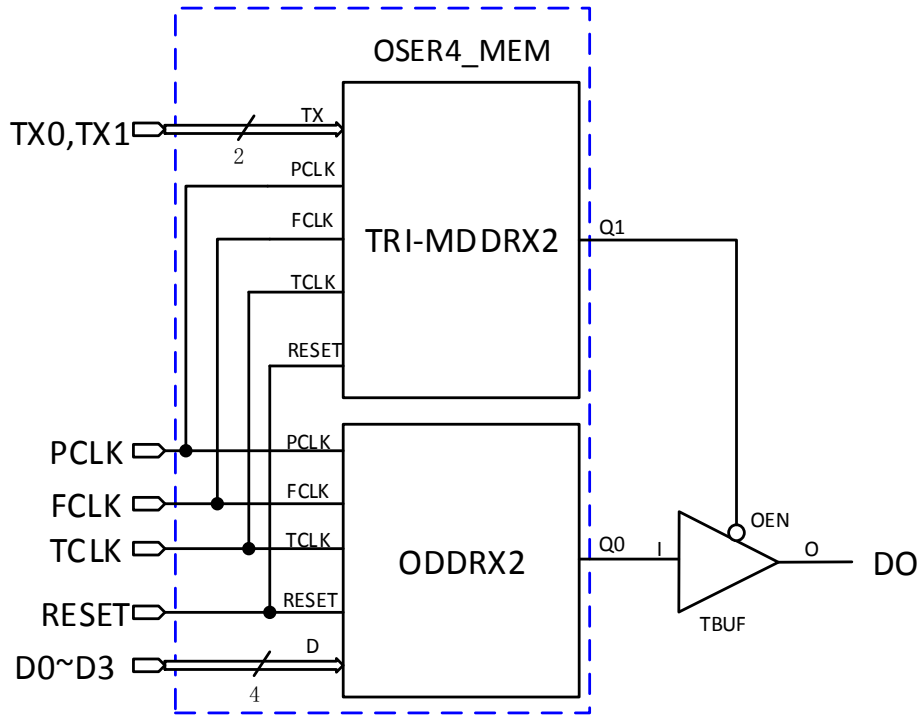
表 4-43 OSER4\_MEM 适用器件

| 家族                | 系列     | 器件                                   |
|-------------------|--------|--------------------------------------|
| 晨熙® (Arora)<br>家族 | GW2A   | GW2A-18, GW2A-18C, GW2A-55, GW2A-55C |
|                   | GW2AN  | GW2AN-55C                            |
|                   | GW2AR  | GW2AR-18, GW2AR-18C                  |
|                   | GW2ANR | GW2ANR-18C                           |

**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, Q1 用于 Q0 所连的 IOBUF/TBUF 的 OEN 信号。其逻辑框图如图 4-30 所示。

**图 4-30 OSER4\_MEM 逻辑框图**

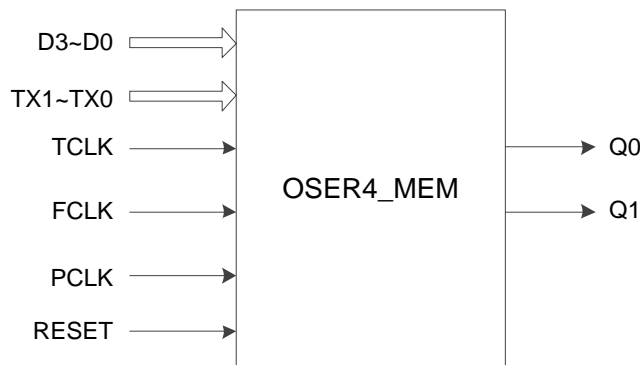


PCLK、FCLK 和 TCLK 的频率关系为： $f_{PCLK} = 1/2 f_{FCLK} = 1/2 f_{TCLK}$ 。

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 4-31 OSER4\_MEM 端口示意图**





## 端口介绍

表 4-44 OSER4\_MEM 端口介绍

| 端口名     | I/O    | 描述  |
|---------|--------|---|
| D3~D0   | Input  | OSER4_MEM 数据输入信号  |
| TX1~TX0 | Input  | 通过 TRI-MDDR2 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 4-45 OSER4\_MEM 参数介绍

| 参数名         | 取值范围             | 默认值      | 描述   |
|-------------|------------------|----------|--|
| GSREN       | "false", "true"  | "false"  | 启用全局复位 GSR   |
| LSREN       | "false", "true"  | "true"   | 启用本地复位 RESET   |
| TXCLK_POL   | 1'b0, 1'b1       | 1'b0     | Q1 输出时钟极性控制 <ul style="list-style-type: none"> <li>● 1'b0:数据上升沿输出;</li> <li>● 1'b1:数据下降沿输出</li> </ul>  |
| TCLK_SOURCE | "DQSW","DQSW270" | " DQSW " | TCLK 来源选择 <ul style="list-style-type: none"> <li>● "DQSW": 来自 DQS 模块的 DQSW0;</li> <li>● "DQSW270": 来自 DQS 模块的 DQSW270</li> </ul>                     |
| HWL         | "false", "true"  | "false"  | OSER4_MEM 数据 d_up0/1 时序关系控制 <ul style="list-style-type: none"> <li>● "false": d_up1 比 d_up0 提前一个周期;</li> <li>● "true": d_up1 和 d_up0 时序相同</li> </ul> |

## 连接规则

- Q0 可直接连接 OBUF，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.GSREN="false";
defparam uut.LSREN="true";
defparam uut.HWL="false";
defparam uut.TCLK_SOURCE="DQSW";
defparam uut.TXCLK_POL=1'b0;

```

**Vhdl 例化:**

```

COMPONENT OSER4_MEM
    GENERIC (GSREN:string:="false";
            LSREN:string:="true";
            HWL:string:="false";
            TXCLK_POL:bit:='0';
            TCLK_SOURCE:string:="DQSW"
    );
    PORT(
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        D0:IN std_logic;
        D1:IN std_logic;
        D2:IN std_logic;

```

```

        D3:IN std_logic;
        TX0:IN std_logic;
        TX1:IN std_logic;
        TCLK:IN std_logic;
        FCLK:IN std_logic;
        PCLK:IN std_logic;
        RESET:IN std_logic
    );
END COMPONENT;
 uut:OSER4_MEM
    GENERIC MAP (GSREN=>"false",
                 LSREN=>"true",
                 HWL=>"false",
                 TXCLK_POL=>'0',
                 TCLK_SOURCE=>"DQSW"
    )
    PORT MAP (
        Q0=>q0,
        Q1=>q1,
        D0=>d0,
        D1=>d1,
        D2=>d2,
        D3=>d3,
        TX0=>tx0,
        TX1=>tx1,
        TCLK=>tclk,
        FCLK=>fclk,
        PCLK=>pclk,
        RESET=>reset
    );

```

### 4.3.10 OSER8\_MEM

#### 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。

适用器件

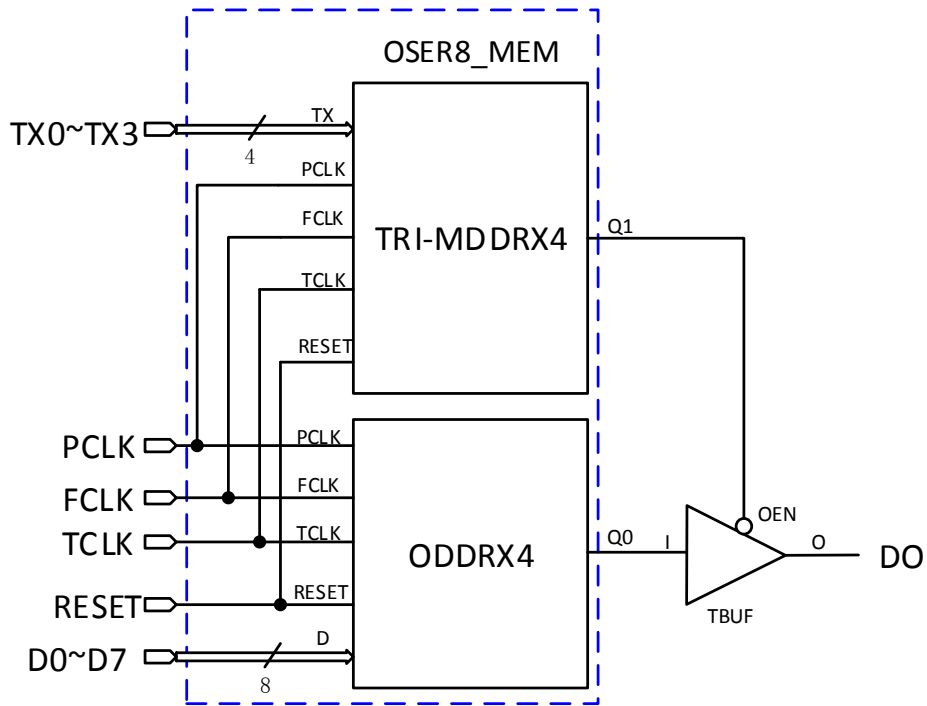
表 4-46 OSER8\_MEM 适用器件

| 家族             | 系列     | 器件                                   |
|----------------|--------|--------------------------------------|
| 晨熙® (Arora) 家族 | GW2A   | GW2A-18, GW2A-18C, GW2A-55, GW2A-55C |
|                | GW2AN  | GW2AN-55C                            |
|                | GW2AR  | GW2AR-18, GW2AR-18C                  |
|                | GW2ANR | GW2ANR-18C                           |

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, Q1 用于 Q0 所连的 IOBUF/TBUF 的 OEN 信号。其逻辑框图如图 4-32 所示。

图 4-32 OSER8\_MEM 逻辑框图

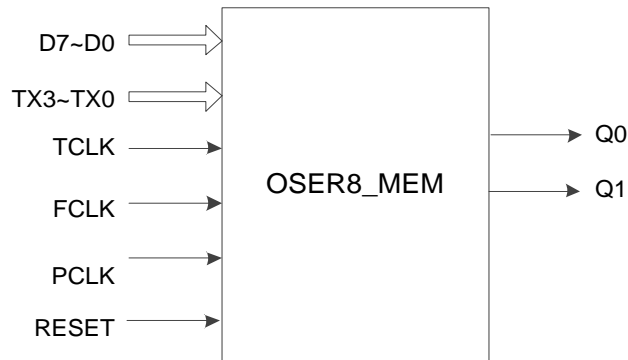


PCLK、FCLK 和 TCLK 的频率关系为： $f_{PCLK} = 1/4 f_{FCLK} = 1/4 f_{TCLK}$ 。

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 4-33 OSER8\_MEM 端口示意图



## 端口介绍

表 4-47 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-MDDR4 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 | 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 4-48 OSER8\_MEM 参数介绍

| 参数名         | 取值范围              | 默认值      | 描述  |
|-------------|-------------------|----------|---|
| GSREN       | "false", "true"   | "false"  | 启用全局复位 GSR  |
| LSREN       | "false", "true"   | "true"   | 启用本地复位 RESET  |
| TXCLK_POL   | 1'b0, 1'b1        | 1'b0     | Q1 输出时钟极性控制 <ul style="list-style-type: none"> <li>● 1'b0:数据上升沿输出;</li> <li>● 1'b1:数据下降沿输出</li> </ul>                           |
| TCLK_SOURCE | "DQSW", "DQSW270" | " DQSW " | TCLK 来源选择 <ul style="list-style-type: none"> <li>● "DQSW": 来自 DQS 模块的 DQSW0;</li> <li>● DQSW270": 来自 DQS 模块的 DQSW270</li> </ul> |

| 参数名 | 取值范围            | 默认值     | 描述   |
|-----|-----------------|---------|--|
| HWL | "false", "true" | "false" | OSER8_MEM 数据 d_up0/1 时序关系控制 <ul style="list-style-type: none"> <li>● "false": d_up1 比 d_up0 提前一个周期;</li> <li>● "true": d_up1 和 d_up0 时序相同</li> </ul> |

### 连接规则

- Q0 可直接连接 OBUF，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.GSREN="false";
defparam uut.LSREN ="true";

```

```

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

```

**Vhdl 例化:**

```

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

```

```
HWL=>"false",
TXCLK_POL=>'0',
TCLK_SOURCE=>"DQSW"
)
PORT MAP (
    Q0=>q0,
    Q1=>q1,
    D0=>d0,
    D1=>d1,
    D2=>d2,
    D3=>d3,
    D4=>d4,
    D5=>d5,
    D6=>d6,
    D7=>d7,
    TX0=>tx0,
    TX1=>tx1,
    TX2=>tx2,
    TX3=>tx3,
    TCLK=>tclk,
    FCLK=>fclk,
    PCLK=>pclk,
    RESET=>reset
);
```

## 4.4 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。

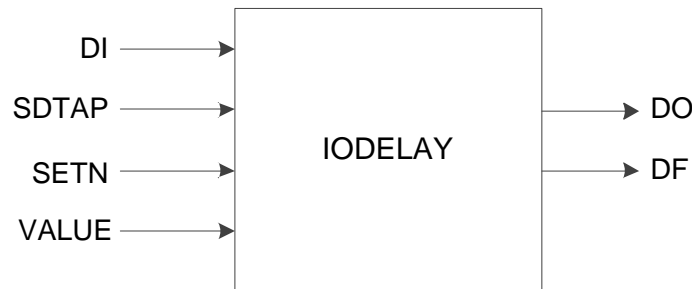
#### 功能描述

每个 IO 都包含 IODELAY 模块，总共提供 128（0~127）种延迟配置，GW1N 系列 FPGA 单步的延迟时间约为 30ps，GW2A 系列 FPGA 单步的延迟时间约为 18ps。IODELAY 可用于 I/O 逻辑的输入或输出，但不能同时作用。



## 端口示意图

图 4-34 IODELAY 端口示意图



## 端口介绍

表 4-49 IODELAY 端口介绍

| 端口名   | I/O    | 描述  |
|-------|--------|---|
| DI    | Input  | 数据输入信号                                    |
| SDTAP | Input  | 控制加载静态延时步长<br>● 0: 加载静态延时<br>● 1: 动态调整延时  |
| SETN  | Input  | 设置动态调整延时的方向<br>● 0: 增加延时;<br>● 1: 减少延时    |
| VALUE | Input  | 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          |
| DO    | Output | 数据输出信号                                    |
| DF    | Output | 输出标志位, 用以表示动态调整延时的 under-flow 或 over-flow |

## 参数介绍

表 4-50 IODELAY 参数介绍

| 参数名          | 取值范围  | 默认值 | 描述       |
|--------------|-------|-----|----------|
| C_STATIC_DLY | 0~127 | 0   | 静态延时步长控制 |

## 原语例化

### Verilog 例化:

```

    IODELAY iodelay_inst(
        .DO(dout),
        .DF(df),
        .DI(di),
  
```

```

        .SDTAP(sdtap),
        .SETN(setn),
        .VALUE(value)
    );
    defparam iodelay_inst.C_STATIC_DLY=0;

```

**Vhdl 例化:**

```

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

```

## 4.4.2 IODELAYC

### 原语介绍

IODELAYC(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。

## 适用器件

表 4-51 IODELAYC 适用器件

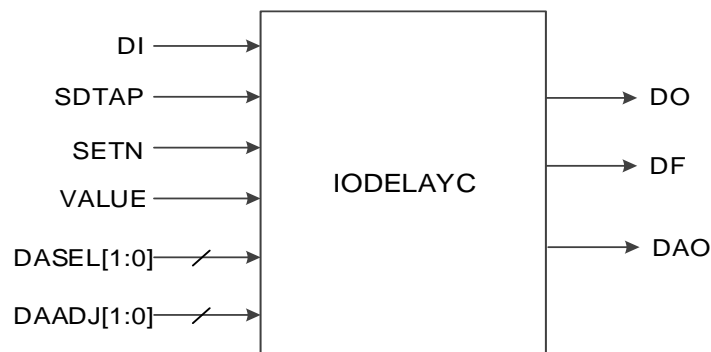
| 家族                | 系列    | 器件       |
|-------------------|-------|----------|
| 小蜜蜂® (LittleBee®) | GW1N  | GW1N-9C  |
|                   | GW1NR | GW1NR-9C |

## 功能描述

每个 IO 都包含 IODELAYC 模块，总共提供 128 (0~127) 种延迟配置，与 IODELAY 相比，增加了更多的延时调整。IODELAYC 仅用于 I/O 逻辑的输入，不可用于 I/O 逻辑输出。

## 端口示意图

图 4-35 IODELAYC 端口示意图



## 端口介绍

表 4-52 IODELAYC 端口介绍

| 端口名        | I/O    | 描述   |
|------------|--------|--|
| DI         | Input  | 数据输入信号   |
| SDTAP      | Input  | 控制加载静态延时步长<br><ul style="list-style-type: none"> <li>● 0: 加载静态延时</li> <li>● 1: 动态调整延时</li> </ul> |
| SETN       | Input  | 设置动态调整延时的方向<br><ul style="list-style-type: none"> <li>● 0: 增加延时;</li> <li>● 1: 减少延时</li> </ul>   |
| VALUE      | Input  | 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  |
| DASEL[1:0] | Input  | 动态控制 DAO 延时模式  |
| DAADJ[1:0] | Input  | 动态控制 DAO 相对 DO 的延时值  |
| DO         | Output | 数据输出信号   |
| DAO        | Output | 数据延时调整输出信号   |
| DF         | Output | 输出标志位，用以表示动态调整延时的 under-flow   |

| 端口名 | I/O | 描述          |
|-----|-----|-------------|
|     |     | 或 over-flow |

## 参数介绍

表 4-53 IODELAYC 参数介绍

| 参数名          | 取值范围           | 默认值   | 描述   |
|--------------|----------------|-------|--|
| C_STATIC_DLY | 0~127          | 0     | 静态延时步长控制   |
| DYN_DA_SEL   | "true"/"false" | false | <ul style="list-style-type: none"> <li>● false: 选择参数 DA_SEL 静态控制 DAO 延时模式</li> <li>● true: 选择信号 DASEL 动态控制 DAO 延时模式</li> </ul> |
| DA_SEL       | 2'b00~2'b11    | 2'b00 | 静态控制 DAO 延时模式  |

## 原语例化

### Verilog 例化:

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

### Vhdl 例化:

```
COMPONENT IODELAYC
    GENERIC (C_STATIC_DLY:integer:=0;
             DYN_DA_SEL:string:="false";
             DA_SEL:bit_vector:="00"
    );
    PORT(
```

```

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

```

### 4.4.3 IODELAYB

#### 原语介绍

IODELAYB(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。

#### 适用器件

表 4-54 IODELAYB 适用器件

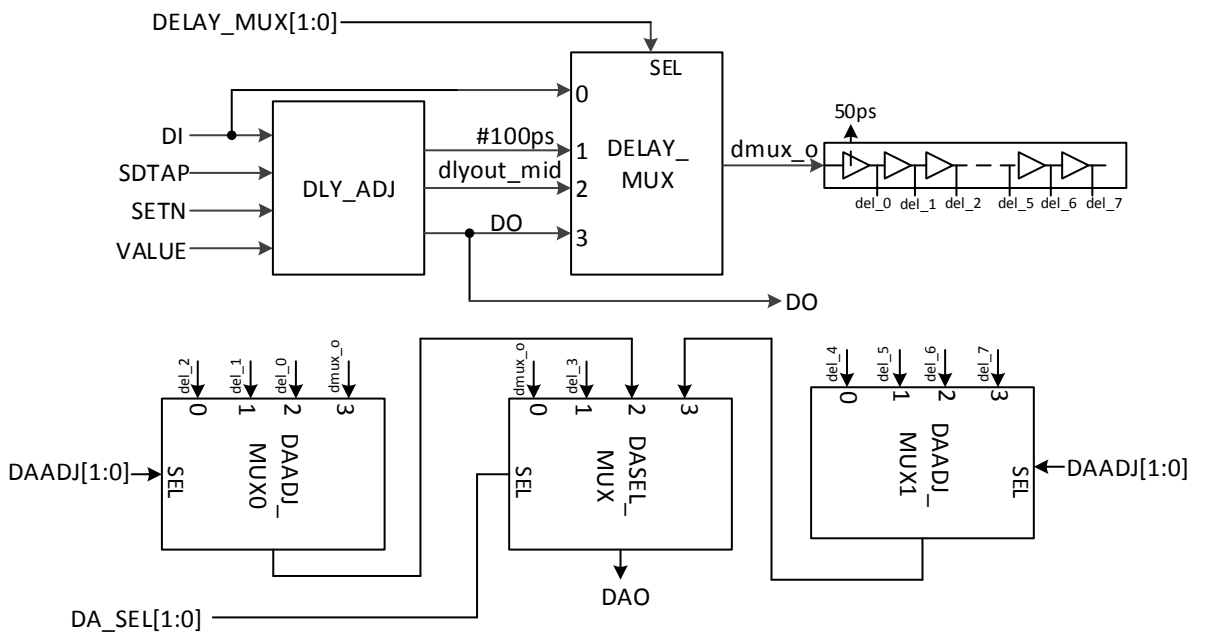
| 家族               | 系列   | 器件                                   |
|------------------|------|--------------------------------------|
| 小蜜蜂 <sup>®</sup> | GW1N | GW1N-2, GW1N-1P5, GW1N-2B, GW1N-1P5B |

| 家族                          | 系列    | 器件                |
|-----------------------------|-------|-------------------|
| (LittleBee <sup>®</sup> )家族 | GW1NR | GW1NR-2, GW1NR-2B |

### 功能描述

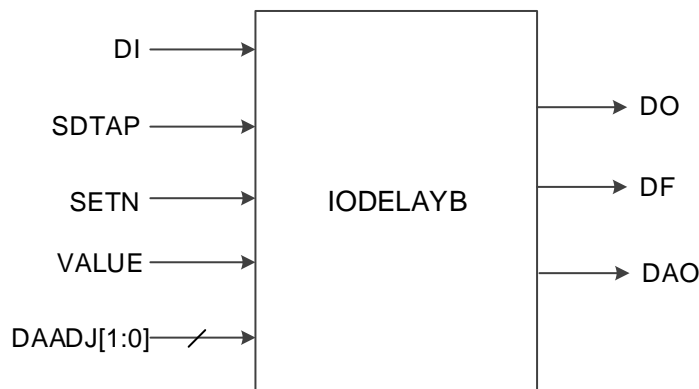
每个 IO 都包含 IODELAYB 模块，总共提供 128 (0~127) 种延迟配置，与 IODELAY 相比，增加了更多的延时调整，其内部结构框图如图 4-36 所示。IODELAYB 仅用于 I/O 逻辑的输入，不可用于 I/O 逻辑输出。

图 4-36 IODELAYB 内部结构框图



### 端口示意图

图 4-37 IODELAYB 端口示意图



## 端口介绍

表 4-55 IODELAYB 端口介绍

| 端口名        | I/O    | 描述   |
|------------|--------|--|
| DI         | Input  | 数据输入信号   |
| SDTAP      | Input  | 控制加载静态延时步长<br><ul style="list-style-type: none"> <li>● 0: 加载静态延时</li> <li>● 1: 动态调整延时</li> </ul> |
| SETN       | Input  | 设置动态调整延时的方向<br><ul style="list-style-type: none"> <li>● 0: 增加延时;</li> <li>● 1: 减少延时</li> </ul>   |
| VALUE      | Input  | 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。  |
| DAADJ[1:0] | Input  | 动态控制 DAO 相对 DO 的延时值  |
| DO         | Output | 数据输出信号   |
| DAO        | Output | 数据延时调整输出信号   |
| DF         | Output | 输出标志位, 用以表示动态调整延时的 under-flow 或 over-flow。   |

## 参数介绍

表 4-56 IODELAYB 参数介绍

| 参数名          | 取值范围        | 默认值   | 描述  |
|--------------|-------------|-------|---|
| C_STATIC_DLY | 0~127       | 0     | 静态延时步长控制  |
| DELAY_MUX    | 2'b00~2'b11 | 2'b00 | Delay MUX 选择<br><ul style="list-style-type: none"> <li>● 2'b00:dmux_o=DI;</li> <li>● 2'b01:#100ps dmux_o=DI;</li> <li>● 2'b10:dmux_o=dlyout_mid;</li> <li>● 2'b11:dmux_o=DO。</li> </ul> |
| DA_SEL       | 2'b00~2'b11 | 2'b00 | 静态控制 DAO 延时模式   |

注!

在使用 IODELAYB 时, 参数 DELAY\_MUX 和 DA\_SEL 的关联关系如下:

- DELAY\_MUX:2/3 -> DA\_SEL:0/1。即 DELAY\_MUX 为 2 或 3 时, DA\_SEL 可取 0 或 1;
- DELAY\_MUX:0/1 -> DA\_SEL:0/2/3。即 DELAY\_MUX 为 0 或 1 时, DA\_SEL 可取 0 或 2 或 3。

## 连接规则

DO 不能连接 IDDR/IDES, DAO 只能连接 IDDR/IDES 的数据输入。

## 原语例化

Verilog 例化:

```
IODELAYB iodelayb_inst(
```

```

        .DO(dout),
        .DAO(douta),
        .DF(df),
        .DI(di),
        .SDTAP(sdtap),
        .SETN(setn),
        .VALUE(value),
        .DAADJ(daadj)
    );
    defparam iodelayb_inst.C_STATIC_DLY=0;
    defparam iodelayb_inst.DELAY_MUX = 2'b00;
    defparam iodelayb_inst.DA_SEL=2'b00;

```

**Vhdl 例化:**

```

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

 uut:IODELAYB
    GENERIC MAP (C_STATIC_DLY=>0,
                 DELAY_MUX =>"00",
                 DA_SEL=>"00"
    )
    PORT MAP (

```



```

DO=>dout,
DAO=>douta,
DF=>df,
DI=>di,
SDTAP=>sdtap,
SETN=>setn,
VALUE=>value,
DAADJ=>daadj
);

```

## 4.5 取样模块

### 原语介绍

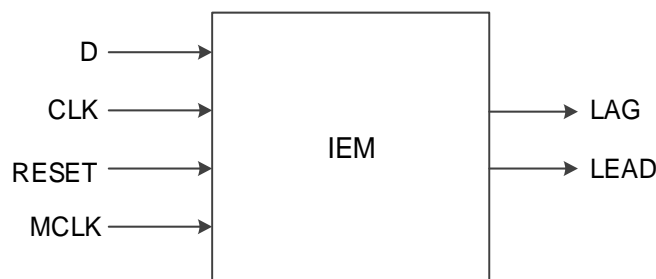
IEM(Input Edge Monitor)输入边沿监测，是 IO 模块所包含的一个取样模块。

### 功能描述

IEM 用来取样数据边沿，可与延迟模块一起使用来调节动态取样窗口，用于 DDR 模式。

### 端口示意图

图 4-38 IEM 端口示意图



### 端口介绍

表 4-57 IEM 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  | 异步复位输入信号，高电平有效           |
| MCLK  | Input  | IEM 检测时钟，可来自用户逻辑，作用于输出标志 |
| LAG   | Output | IEM 边沿比较 LAG 输出标志        |
| LEAD  | Output | IEM 边沿比较 LEAD 输出标志       |

## 参数介绍

表 4-58 IEM 参数介绍

| 参数名     | 取值范围  | 默认值     | 描述           |
|---------|---|---------|--------------|
| WINSIZE | "SMALL", "MIDSMALL",<br>"MIDLARGE", "LARGE" | "SMALL" | 窗口大小设置       |
| GSREN   | "false", "true"                             | "false" | 启用全局复位 GSR   |
| LSREN   | "false", "true"                             | "true"  | 启用本地复位 RESET |

## 原语例化

### Verilog 例化:

```

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

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

```

### Vhdl 例化:

```

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

```

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

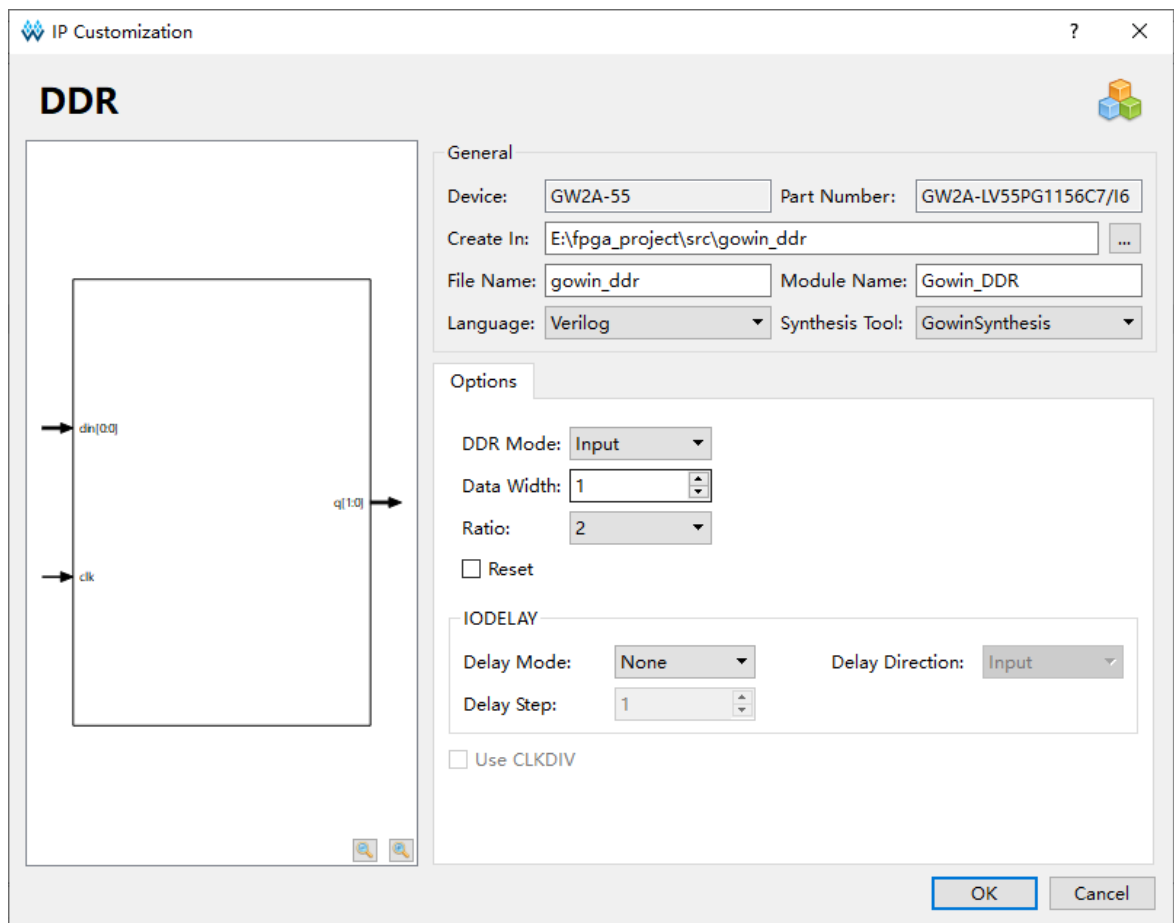
# 5 IP 调用

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“File”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. File 配置框

File 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;
- **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;
- **Synthesis Tool:** 配置选择综合工具;
- **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 2,4,7,8,10,16;
- **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;
- **IODELAY:** 配置 DDR 是否使用延时模块;
  - “Delay Mode”, 配置 Delay 模式, “None” 表示不使用 IODELAY, “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。
  - “Delay Step”, 选择静态调整延时的步数, 范围 1~128。
  - “Delay Direction”, DDR Mode 双向模式时, 若使用 IODELAY, 选择 IODELAY 连接输入端或输出端。
- **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\_dds.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\_dds\_tmp.v，为用户提供 IP 设计使用模板文件；
- IP 配置文件“gowin\_dds.ipc”，用户可加载该文件对 IP 进行配置。

**注！**

如配置中选择语言是 VHDL，则产生的前两个文件名后缀为.vhd。

