Gowin 原语指南

SUG283-2.1, 2020-01-16

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
2017/04/20	1.0	初始版本。
2017/09/19	1.1	● 增加支持器件系列 GW1NR-4、GW1N-6、GW1N-9、GW1NR-9； ● 增加 ELVDS_IOBUF、TLVDS_IOBUF、BUFG、BUFS,、OSC,、IEM； ● 更新 DSP 原语； ● 更新 ODDR/ODDRC、IDDR_MEM、IDES4_MEM、IDES8_MEM、RAM16S1, RAM16S2、RAM16S4、RAM16SDP1、RAM16SDP2、RAM16SDP4、ROM16 部分 port 名称； ● 更新 OSC、PLL、DLLDLY 部分 Attribute； ● 更新部分原语例化； ● 增加 MIPI_IBUF_HS，MIPI_IBUF_LP，MIPI_OBUF，IDES16，OSER16； ● 更新 CLKDIV 部分 Attribute。
2018/04/12	1.2	增加 vhdl 原语例化。
2018/08/08	1.3	● 增加支持器件系列 GW1N-2B、GW1N-4B、GW1NR-4B、GW1N-6ES、GW1N-9ES、GW1NR-9ES、GW1NS-2、GW1NS-2C； ● 增加 I3C_IOBUF、DHCEN； ● 增加 User Flash； ● 增加 EMPU； ● 更新原语名称。
2018/10/26	1.4	● 增加支持器件系列 GW1NZ-1、GW1NSR-2C； ● 增加 OSCZ、FLASH96KZ。
2018/11/15	1.5	● 增加支持器件系列 GW1NSR-2; ● 删除器件 GW1N-6ES、GW1N-9ES、GW1NR-9ES。
2019/01/26	1.6	● CLKDIV 的 8 分频新增支持 GW1NS-2 器件； ● 删除 TLVDS_TBUF/OBUF 支持器件中的 GW1N-1。
2019/02/25	1.7	删除 TLVDS_IOBUF 支持器件中的 GW1N-1。
2019/05/20	1.8	● 增加支持器件系列 GW1N-1S; ● 增加 MIPI_IBUF; ● 增加 OSCH; ● 增加 SPMI; ● 增加 I3C; ● 更新 OSC 的支持器件。
2019/10/20	1.9	更新 IOB、BSRAM、CLOCK 模块。
2019/11/28	2.0	● 增加 GSR、INV 等 Miscellaneous 模块; ● 更新支持器件信息; ● 增加 FLASH64KZ，删除 FLASH96KZ。
2020/01/16	2.1	● 增加 IODELAYA、rPLL、PLLVR、CLKDIV2； ● 增加 DPB/DPX9B、SDPB/SDPX9B、rSDP/rSDPX9、rROM/rROMX9、pROM/pROMX9； ● 增加 EMCU、BANDGAP、FLASH64K； ● 更新 IODELAY、PLL、CLKDIV、OSC、DQCE；

日期	版本	说明
		● 增加 FF、LATCH 放置规则； ● 增加支持器件 GW2A-55C； ● GW1N-6/GW1N-9/GW1NR-9 禁掉 DP/DPX9、DPB/DPX9B； ● IOLOGIC 增加 register 说明备注； ● GW1NZ-1 禁掉 DP/DPB 的 1,2,4,8 位宽，DPX9/DPX9 的 9 位宽。

目录

图目录

表目录

1 IOB

1.1 Buffer/LVDS

Buffer，缓冲器，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）。

1.1.1 IBUF

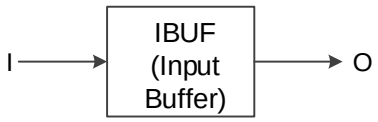
原语介绍

IBUF(Input Buffer)，输入缓冲器。

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

结构框图

图 1-1 IBUF 结构框图



Port 介绍

表 1-1 Port 介绍

Port Name	I/O	Description
I	Input	Data Input
O	Output	Data Output

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
        );
```

1.1.2 OBUF

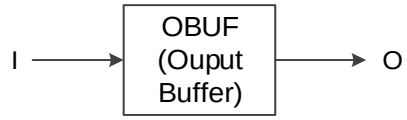
原语介绍

OBUF(Output Buffer)，输出缓冲器。

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

结构框图

图 1-2 OBUF 结构框图



Port 介绍

表 1-2 Port 介绍

Port Name	I/O	Description
I	Input	Data Input
O	Output	Data Output

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
        );
```

1.1.3 TBUF

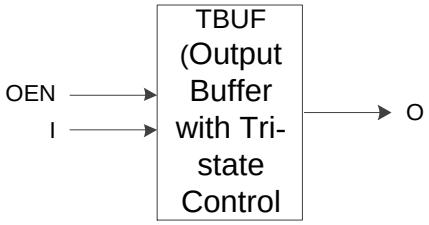
原语介绍

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

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

结构框图

图 1-3 TBUF 结构框图



Port 介绍

表 1-3 Port 介绍

Port Name	I/O	Description
I	Input	Data Input
OEN	Input	Output Enable
O	Output	Data Output

原语示例例化

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
            );
```

1.1.4 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。

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

结构框图

图 1-4 IOBUF 结构框图

PORT 介绍

表 1-4 Port 介绍

Port Name	I/O	Description
I	Input	Data Input
OEN	Input	Output Enable
IO	Inout	Inout Port
O	Output	Data Output

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
             );
```

1.1.5 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。

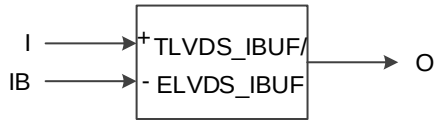
支持器件：GW1N-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

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

支持器件：GW1N-1、GW1N-1S、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

结构框图

图 1-5 TLVDS_IBUF/ELVDS_IBUF 结构框图



Port 介绍

表 1-5 Port 介绍

Port Name	I/O	Description
I	Input	Differential Input
IB	Input	Differential Input
O	Output	Data Output

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
        );
```

1.1.6 LVDS ouput buffer

原语介绍

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

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

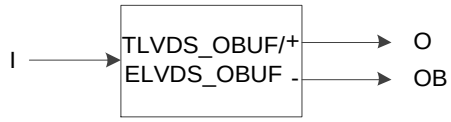
支持器件：GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

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

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

结构框图

图 1-6 TLVDS_OBUF/ELVDS_OBUF 结构框图



Port 介绍

表 1-6 Port 介绍

Port Name	I/O	Description
I	Input	Data Input
OB	Output	Differential Output
O	Output	Differential Output

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
        );
```

1.1.7 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。

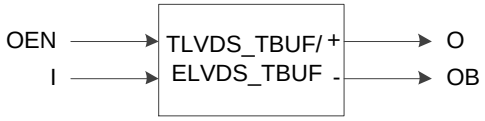
支持器件：GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

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

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

结构框图

图 1-7 TLVDS_TBUF/ELVDS_TBUF 结构框图



Port 介绍

表 1-7 Port 介绍

Port Name	I/O	Description
I	Input	Data Input
OEN	Input	Output Enable
OB	Output	Differential Output
O	Output	Differential Output

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
            );
```

1.1.8 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

支持器件： GW1N-2、GW1N-2B、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

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。

支持器件：GW1N-1、GW1N-1S、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

结构框图

图 1-8 TLVDS_IOBUF/ELVDS_IOBUF 结构框图

Port 介绍

表 1-8 Port 介绍

Port Name	I/O	Description
I	Input	Data Input

OEN	Input	Output Enable
O	Output	Data Output
IOB	Inout	Differential Inout
IO	Inout	Differential Inout

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
    );
```

1.1.9 MIPI_IBUF_HS

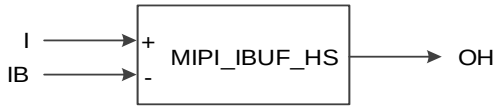
原语介绍

MIPI_IBUF_HS(MIPI High Speed Input Buffer)，MIPI 高速输入缓冲器。

支持器件：GW1N-1S、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9。

结构框图

图 1-9 MIPI_IBUF_HS 结构框图



Port 介绍

表 1-9 Port 介绍

Port Name	I/O	Description
I	Input	Differential Input
IB	Input	Differential Input
OH	Output	Data Output

原语例化

Verilog 例化：

```
MIPI_IBUF_HS uut(
    .OH(OH),
    .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;
uut: MIPI_IBUF_HS
        PORT MAP(
            OH=>OH,
            I=>I,
            IB=>IB
        );
```

1.1.10 MIPI_IBUF_LP

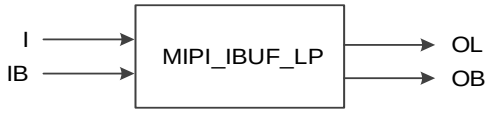
原语介绍

MIPI_IBUF_LP(MIPI Low Power Input Buffer)，MIPI 低功耗输入缓冲器。

支持器件：GW1N-1S、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9。

结构框图

图 1-10 MIPI_IBUF_LP 结构框图



Port 介绍

表 1-10 Port 介绍

Port Name	I/O	Description
I	Input	Data Input
IB	Input	Data Input
OL	Output	Data Output
OB	Output	Data Output

原语例化

Verilog 例化：

```
MIPI_IBUF_LP uut(
    .OL(OL),
    .OB(OB),
    .I(I),
    .IB(IB)
);
```

Vhdl 例化：

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

1.1.11 MIPI_IBUF

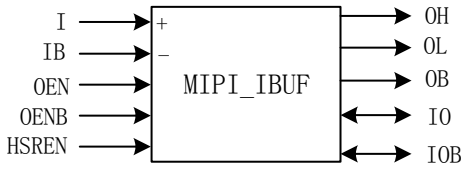
原语介绍

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。

支持器件：GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9。

结构框图

图 1-11 MIPI_IBUF 结构图



Port 介绍

表 1-11 Port 介绍

Port Name	I/O	Description
I	Input	Data Input
IB	Input	Data Input
HSREN	Input	Mode Selection, HS or LP
OEN	Input	Data Input
OENB	Input	Data Input
OH	Output	Data Output
OL	Output	Data Output
OB	Output	Data Output
IO	Output	Data Output
IOB	Output	Data Output

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 (
OEN, OENB,
            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
            );
```

1.1.12 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。

支持器件：GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9。

结构框图

图 1-12 MIPI_OBUF 结构框图

Port 介绍

表 1-12 Port 介绍

Port Name	I/O	Description
I	Input	Data Input
IB	Input	Data Input
MODESEL	Input	Mode Selection, HS or LP
O	Output	Data Output
OB	Output	Data Output

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
          );
```

1.1.13 I3C_IOBUF

原语介绍

I3C_IOBUF 有两种工作模式：Normal 模式和 I3C 模式。

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

支持器件：GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9。

结构框图

图 1-13 I3C_IOBUF 结构框图

Port 介绍

表 1-13 Port 介绍

Port Name	I/O	Description
I	Input	Data Input
IO	Inout	Inout Port
MODESEL	Input	Mode Selection, Normal or I3C
O	Output	Data Output

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
        );
```

1.2 IOLOGIC

IOLOGIC 中的 register 同 CLU 中的 FF/LATCH，请参考 2.4FF/2.5LATCH 章节。

1.2.1 IDDR

原语名称

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

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、

GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

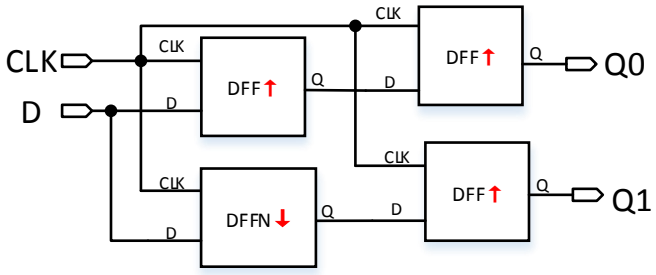
端口示意图

图 1-14 IDDR 端口示意图

功能描述

IDDR 模式，输出数据在同一时钟边沿提供给 FPGA 逻辑。其逻辑框图如图 1-15 所示。

图 1-15 IDDR 逻辑框图



端口介绍

表 1-14 端口介绍

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

参数介绍

表 1-15 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

连接合法性规则

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

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
            );
```

1.2.2 ODDR

原语名称

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

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

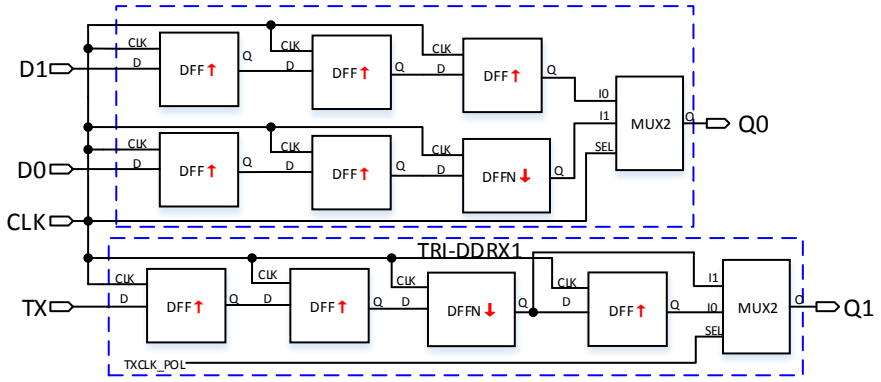
端口示意图

图 1-16 ODDR 端口示意图

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。其逻辑框图如图 1-17 所示。

图 1-17 ODDR 逻辑框图



端口介绍

表 1-16 端口介绍

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

参数介绍

表 1-17 参数介绍

参数名	取值范围	默认值	描述
TXCLK_POL	1’b0, 1’b1	1’b0	Q1 输出时钟极性控制 1'b0:Q1 上升沿输出; 1'b1:Q1 下降沿输出
INIT	1’b0	1’b0	ODDR 输出的初始取值

连接合法性规则

- 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

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:bit:='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
          );
```

1.2.3 IDDRC

原语名称

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。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 1-18 IDDRC 端口示意图

功能描述

IDDRC 模式，输出数据在同一时钟边沿提供给 FPGA 逻辑。

端口介绍

表 1-18 端口介绍

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

参数介绍

表 1-19 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

连接合法性规则

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

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
        );
```

1.2.4 ODDRC

原语名称

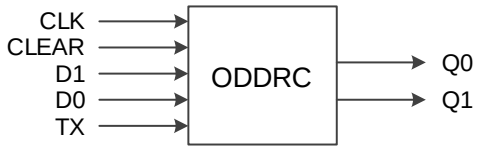
ODDRC(Dual Data Rate Output with Asynchronous Clear)与 ODDR 功能类似，实现双倍速率输出，同时具有异步复位功能。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

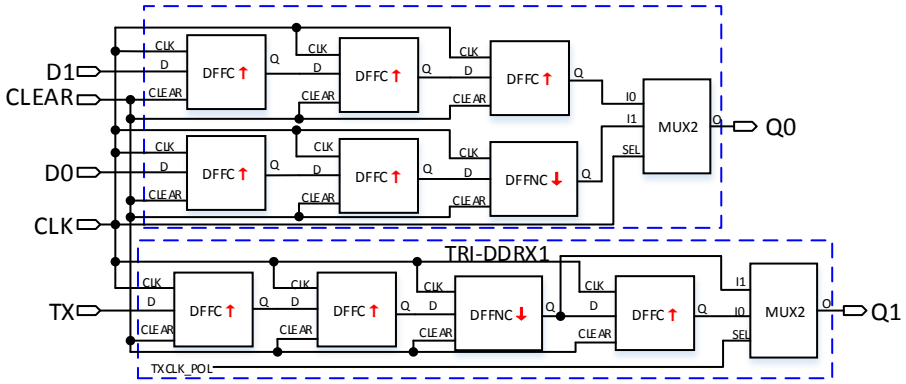
图 1-19 ODDRC 端口示意图



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 1-20 所示。

图 1-20 ODDRC 逻辑框图



端口介绍

表 1-20 端口介绍

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

参数介绍

表 1-21 参数介绍

参数名	取值范围	默认值	描述
TXCLK_POL	1’b0, 1’b1	1’b0	Q1 输出时钟极性控制 1'b0:Q1 上升沿输出; 1'b1:Q1 下降沿输出
INIT	1’b0	1’b0	ODDRC 输出的初始取值

连接合法性规则

- 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

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:bit:='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
        );
```

1.2.5 IDES4

原语名称

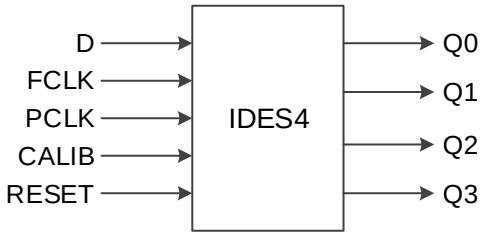
IDES4(1 to 4 Deserializer)为 1 位串行输入、4 位并行输出的解串器。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

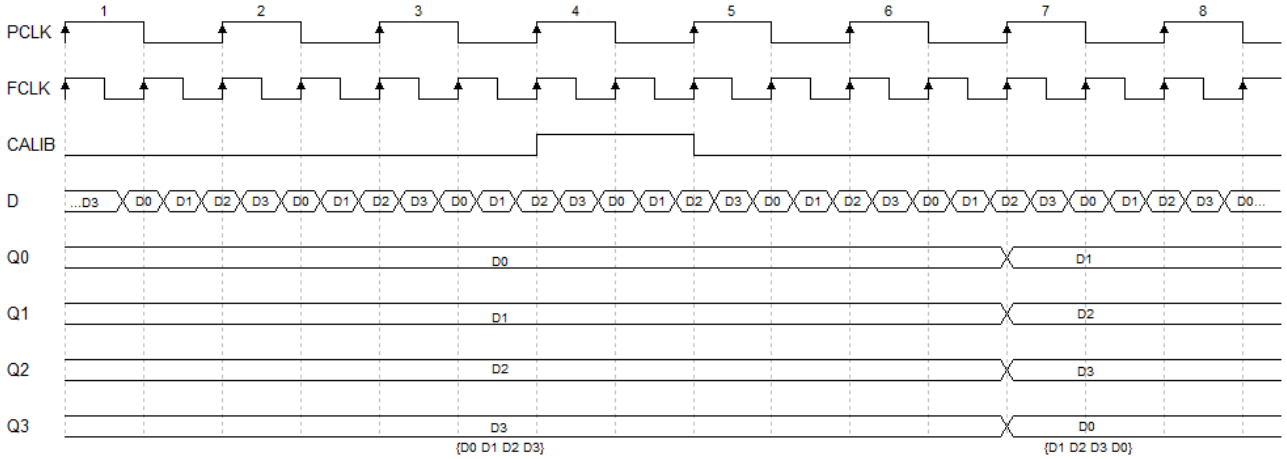
图 1-21 IDES4 端口示意图



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 1-22 所示。

图 1-22 CALIB 示例时序图



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

PCLK 通常由 FCLK 分频获得，

$f_{PCLK} = 1/2\, f_{FCLK}$ 。

端口介绍

表 1-22 端口介绍

端口名	I/O	描述
D	Input	IDES4 数据输入
FCLK	Input	高速时钟输入
PCLK	Input	主时钟输入
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，高电平有效
Q3~Q0	Output	IDES4 数据输出

参数介绍

表 1-23 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

连接合法性规则

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

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
    );
```

1.2.6 IDES8

原语名称

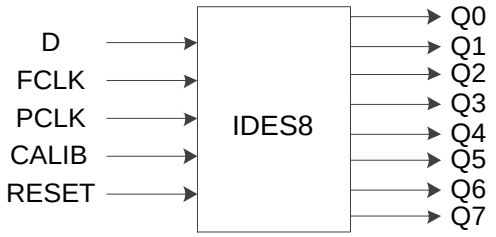
IDES8(1 to 8 Deserializer)为 1 位串行输入、8 位并行输出的解串器。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 1-23 IDES8 端口示意图



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，

$f_{PCLK} = 1/4\, f_{FCLK}$ 。

端口介绍

表 1-24 端口介绍

端口名	I/O	描述
D	Input	IDES8 数据输入
FCLK	Input	高速时钟输入
PCLK	Input	主时钟输入

端口名	I/O	描述
CALIB	Input	CALIB 信号输入，用于调整输出数据顺序，高电平有效
RESET	Input	异步复位输入，高电平有效
Q7~Q0	Output	IDES8 数据输出

参数介绍

表 1-25 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

连接合法性规则

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

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
        );
```

1.2.7 IDES10

原语名称

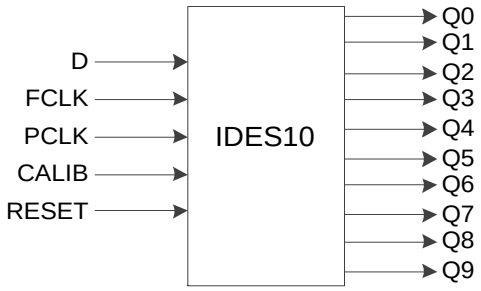
IDES10(1 to 10 Deserializer)为 1 位串行输入、10 位并行输出的解串器。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 1-24 IDES10 端口示意图



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次，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，

$f_{PCLK} = 1/5\, f_{FCLK}$。

端口介绍

表 1-26 端口介绍

端口名	I/O	描述
D	Input	IDES10 数据输入
FCLK	Input	高速时钟输入
PCLK	Input	主时钟输入
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，高电平有效
Q9~Q0	Output	IDES10 数据输出

参数介绍

表 1-27 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

连接合法性规则

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

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
    );
```

1.2.8 IVIDEO

原语名称

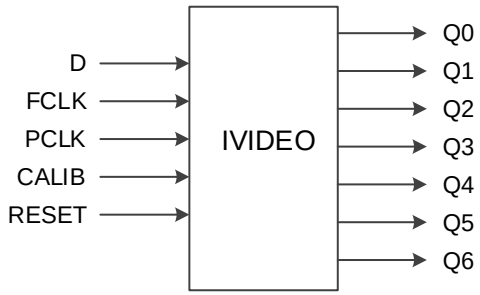
IVIDEO(1 to 7 Deserializer)为 1 位串行输入、7 位并行输出的解串器。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 1-25 IVIDEO 端口示意图



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 分频而来，

$f_{PCLK} = 1/3.5\, f_{FCLK}$ 。

端口介绍

表 1-28 端口介绍

端口名	I/O	描述
D	Input	IVIDEO 数据输入
FCLK	Input	高速时钟输入
PCLK	Input	主时钟输入
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，高电平有效
Q6~Q0	Output	IVIDEO 数据输出

参数介绍

表 1-29 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

连接合法性规则

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

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
        );
```

1.2.9 IDES16

原语名称

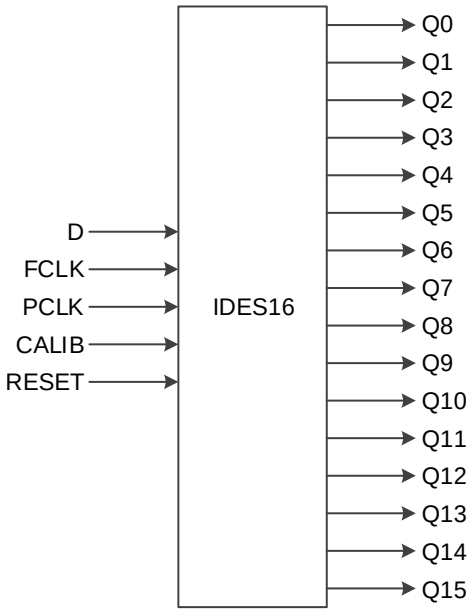
IDES16(1 to 16 Deserializer)为 1 位串行输入、16 位并行输出的解串器。

适用器件

支持器件：GW1N-1S、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9。

端口示意图

图 1-26 IDES16 端口示意图



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，

$f_{PCLK} = 1/8\, f_{FCLK}$。

端口介绍

表 1-30 端口介绍

端口名	I/O	描述
D	Input	IDES16 数据输入
FCLK	Input	高速时钟输入
PCLK	Input	主时钟输入
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，高电平有效
Q15~Q0	Output	IDES16 数据输出

参数介绍

表 1-31 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

连接合法性规则

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

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
        );
```

1.2.10 OSER4

原语名称

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

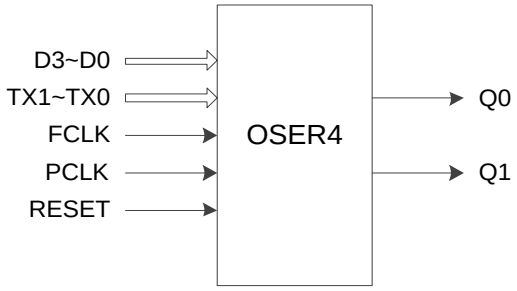
适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、

GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

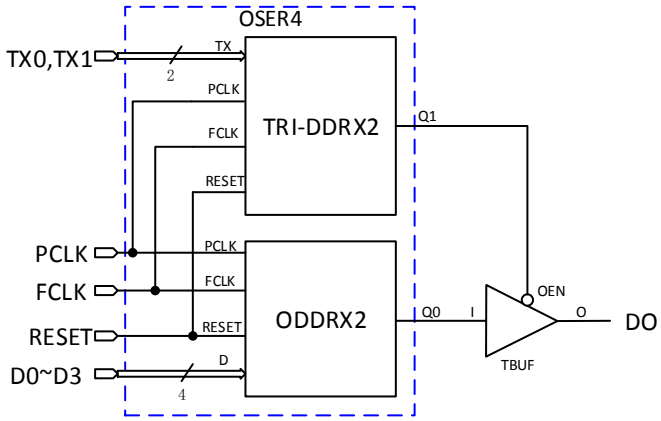
图 1-27 OSER4 端口示意图



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 1-28 所示。

图 1-28 OSER4 逻辑框图



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

端口介绍

表 1-32 端口介绍

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

参数介绍

表 1-33 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 1'b0:数据上升沿输出； 1'b1:数据下降沿输出
HWL	"false", "true"	"false"	OSER4 数据 d_up0/1 时序关系控制 "false": d_up1 比 d_up0 提前一个周期； "true": d_up1 和 d_up0 时序相同

连接合法性规则

- 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

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
        );
```

1.2.11 OSER8

原语名称

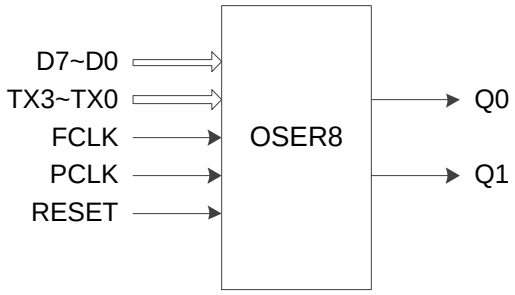
OSER8(8 to 1 Serializer)为 8 位并行输入、1 位串行输出的串化器。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

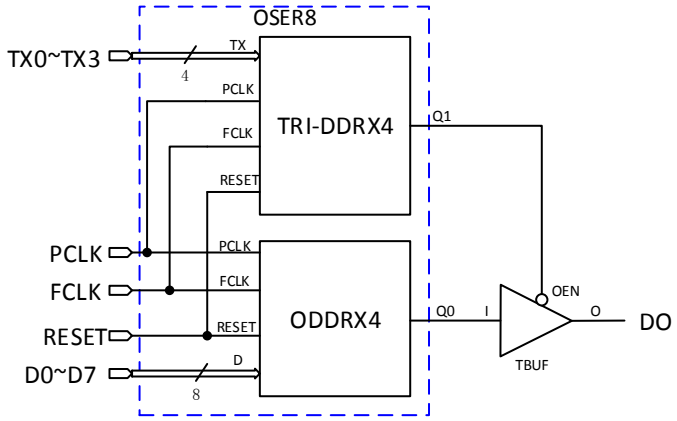
图 1-29 OSER8 端口示意图



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 1-30 所示。

图 1-30 OSER8 逻辑框图



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

端口介绍

表 1-34 端口介绍

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

参数介绍

表 1-35 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 1'b0:数据上升沿输出； 1'b1:数据下降沿输出
HWL	"false", "true"	"false"	OSER8 数据 d_up0/1 时序关系控制 "false": d_up1 比 d_up0 提前一个周期； "true": d_up1 和 d_up0 时序相同

连接合法性规则

- 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

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
                );
```

1.2.12 OSER10

原语名称

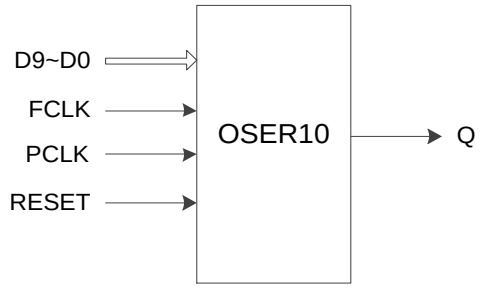
OSER10(10 to 1 Serializer)为 10 位并行输入、1 位串行输出的串化器。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 1-31 OSER10 端口示意图



功能描述

OSER10 模式，实现 10:1 并串转换。PCLK 通常由 FCLK 分频而得来，$f_{PCLK} = 1/5\, f_{FCLK}$ 。

端口介绍

表 1-36 端口介绍

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

参数介绍

表 1-37 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

连接合法性规则

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

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
        );
```

1.2.13 OVIDEO

原语名称

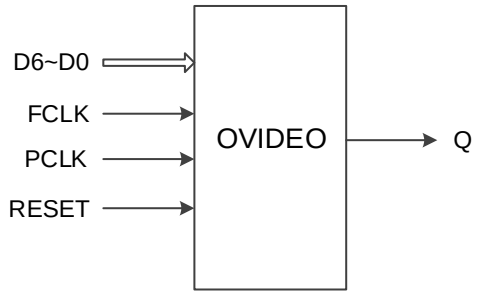
OVIDEO(7 to 1 Serializer)为 7 位并行输入、1 位串行输出的串化器。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 1-32 OVIDEO 端口示意图



功能描述

OVIDEO 模式，实现 7:1 并串转换。PCLK 通常由 FCLK 分频而得来，$f_{PCLK} = 1/3.5\, f_{FCLK}$。

端口介绍

表 1-38 端口介绍

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

参数介绍

表 1-39 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

连接合法性规则

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

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
        );
```

1.2.14 OSER16

原语名称

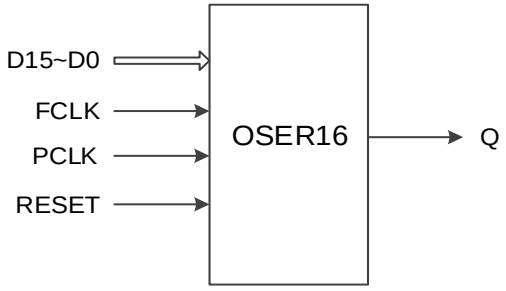
OSER16(16 to 1 Serializer)为 16 位并行输入、1 位串行输出的串化器。

适用器件

支持器件：GW1N-1S、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9。

端口示意图

图 1-33 OSER16 端口示意图



功能描述

OSER16 模式，实现 16:1 并串转换。PCLK 通常由 FCLK 分频而得来，$f_{PCLK} = 1/8\, f_{FCLK}$。

端口介绍

表 1-40 端口介绍

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

参数介绍

表 1-41 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

连接合法性规则

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

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
    );
```

1.2.15 IDDR_MEM

原语名称

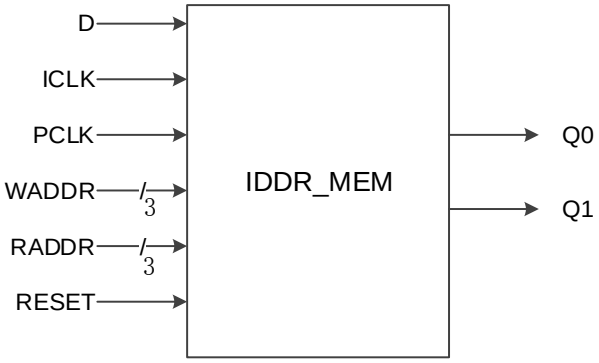
IDDR_MEM(Dual Data Rate Input with Memory)，实现带 memory 的双倍数据速率输入。

适用器件

支持器件：GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 1-34 IDDR_MEM 端口示意图



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。

端口介绍

表 1-42 端口介绍

端口名	I/O	描述
D	Input	IDDR_MEM 数据输入
ICLK	Input	时钟输入，来自 DQS 模块的 DQSR90
PCLK	Input	主时钟输入

端口名	I/O	描述
WADDR[2:0]	Input	写地址，来自 DQS 模块的 WPOINT
RADDR[2:0]	Input	读地址，来自 DQS 模块的 RPOINT
RESET	Input	异步复位输入，高电平有效
Q1~Q0	Output	IDDR_MEM 数据输出

参数介绍

表 1-43 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

连接合法性规则

- 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
        );
```

1.2.16 ODDR_MEM

原语名称

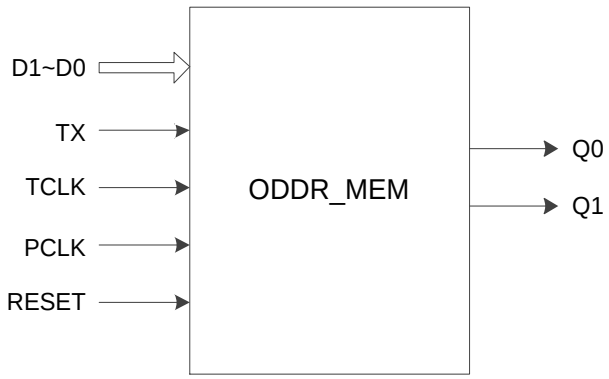
ODDR_MEM(Dual Data Rate Output with Memory)，实现带 memory 的双倍数据速率输出。

适用器件

支持器件：GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

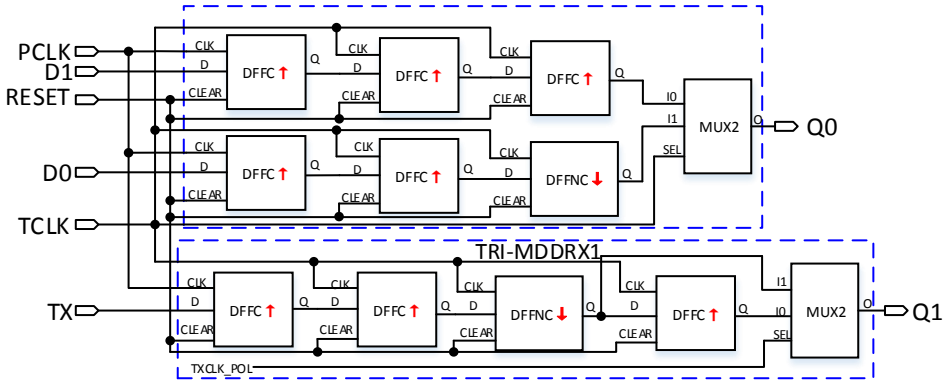
图 1-35 ODDR_MEM 端口示意图



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 1-36 所示。

图 1-36 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。

端口介绍

表 1-44 端口介绍

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

参数介绍

表 1-45 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 1'b0:数据上升沿输出； 1'b1:数据下降沿输出
TCLK_SOURCE	"DQSW","DQSW270"	" DQSW "	TCLK 来源选择 "DQSW": 来自 DQS 模块的 DQSW0； DQSW270": 来自 DQS 模块的 DQSW270

连接合法性规则

- 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
    );
```

1.2.17 IDES4_MEM

原语名称

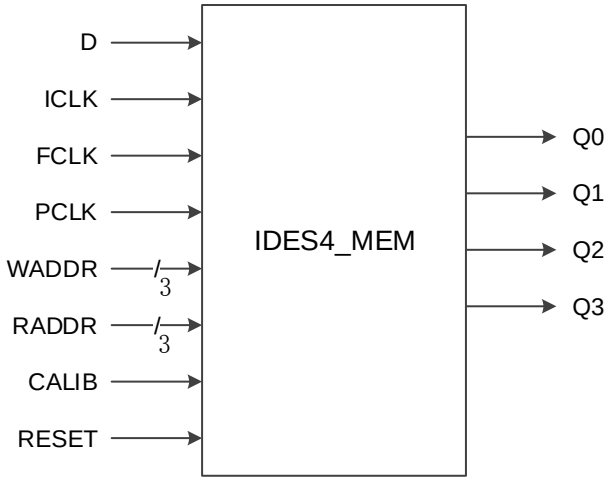
IDES4_MEM(4 to 1 Deserializer with Memory) 带存储功能的 1:4 串并转换器，可实现 1 位串行转 4 位并行。

适用器件

支持器件：GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 1-37 IDES4_MEM 端口示意图



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。

端口介绍

表 1-46 端口介绍

端口名	I/O	描述
D	Input	IDES4_MEM 数据输入
ICLK	Input	时钟输入，来自 DQS 模块的 DQSR90
FCLK	Input	高速时钟输入
PCLK	Input	主时钟输入
WADDR[2:0]	Input	写地址，来自 DQS 模块的 WPOINT
RADDR[2:0]	Input	读地址，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，高电平有效
Q3~Q0	Output	IDES4_MEM 数据输出

参数介绍

表 1-47 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

连接合法性规则

- 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
        );
```

1.2.18 OSER4_MEM

原语名称

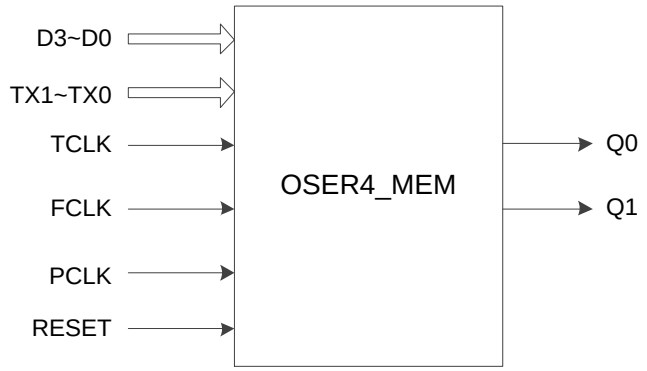
OSER4_MEM(4 to 1 Serializer with Memory) 带存储功能的 4:1 并串转换器，可实现 4 位并行转 1 位串行。

适用器件

支持器件：GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

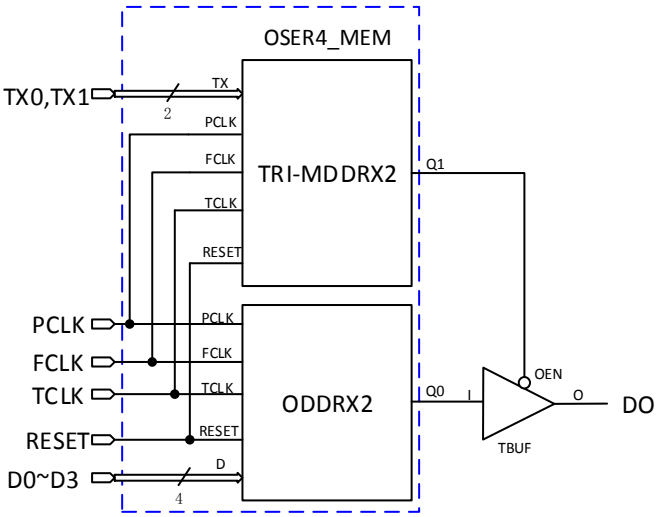
图 1-38 OSER4_MEM 端口示意图



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 1-39 所示。

图 1-39 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。

端口介绍

表 1-48 端口介绍

端口名	I/O	描述
D3~D0	Input	OSER4_MEM 数据输入
TX1~TX0	Input	通过 TRI-MDDRX2 产生 Q1

端口名	I/O	描述
TCLK	Input	时钟输入，来自 DQS 模块的 DQSW0 或 DQSW270
FCLK	Input	高速时钟输入
PCLK	Input	主时钟输入
RESET	Input	异步复位输入，高电平有效
Q0	Output	OSER4_MEM 数据输出
Q1	Output	OSER4_MEM 三态使能数据输出，可连接 Q0 所连的 IOBUF/TBUF 的 OEN 信号，或悬空

参数介绍

表 1-49 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 1'b0:数据上升沿输出； 1'b1:数据下降沿输出
TCLK_SOURCE	"DQSW","DQSW270"	" DQSW "	TCLK 来源选择 "DQSW": 来自 DQS 模块的 DQSW0； DQSW270": 来自 DQS 模块的 DQSW270
HWL	"false", "true"	"false"	OSER4_MEM 数据 d_up0/1 时序关系控制 "false": d_up1 比 d_up0 提前一个周期； "true": d_up1 和 d_up0 时序相同

连接合法性规则

- 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
    );
```

1.2.19 IDES8_MEM

原语名称

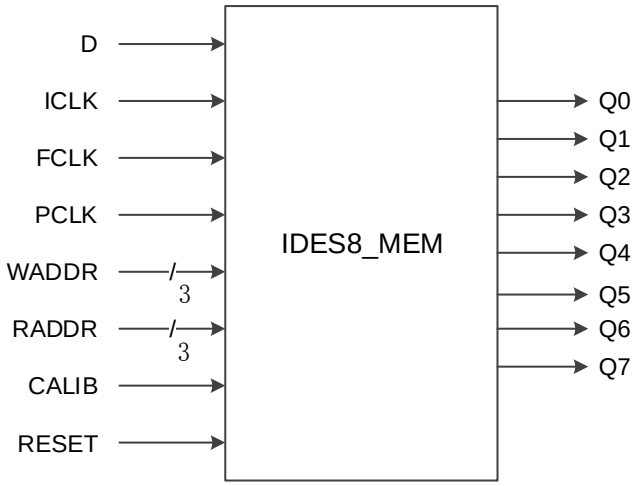
IDES8_MEM(8 to 1 Deserializer with Memory) 带存储功能的 1:8 串并转换器，可实现 1 位串行转 8 位并行。

适用器件

支持器件：GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 1-40 IDES8_MEM 端口示意图



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。

端口介绍

表 1-50 端口介绍

端口名	I/O	描述
D	Input	IDES8_MEM 数据输入

端口名	I/O	描述
ICLK	Input	时钟输入，来自 DQS 模块的 DQSR90
FCLK	Input	高速时钟输入
PCLK	Input	主时钟输入
WADDR[2:0]	Input	写地址，来自 DQS 模块的 WPOINT
RADDR[2:0]	Input	读地址，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，高电平有效
Q7~Q0	Output	IDES8_MEM 数据输出

参数介绍

表 1-51 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

连接合法性规则

- 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
        );
```

1.2.20 OSER8_MEM

原语名称

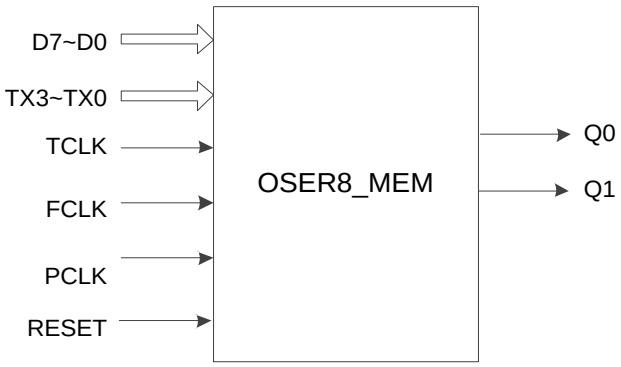
OSER8_MEM(8 to 1 Serializer with Memory) 带存储功能的 8:1 并串转换器，可实现 8 位并行转 1 位串行。

适用器件

支持器件：GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

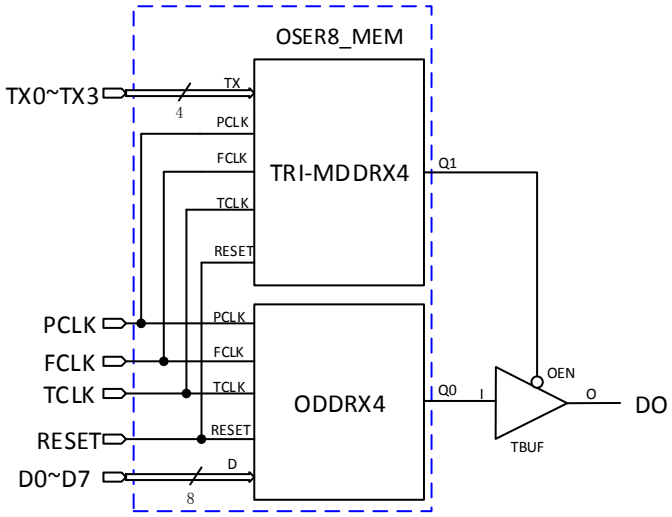
图 1-41 OSER8_MEM 端口示意图



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 1-42 所示。

图 1-42 OSER8_MEM 逻辑框图



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

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

端口介绍

表 1-52 端口介绍

端口名	I/O	描述
D7~D0	Input	OSER8_MEM 数据输入
TX3~TX0	Input	通过 TRI-MDDRX4 产生 Q1
TCLK	Input	时钟输入，来自 DQS 模块的 DQSW0 或 DQSW270
FCLK	Input	高速时钟输入
PCLK	Input	主时钟输入
RESET	Input	异步复位输入，高电平有效
Q0	Output	OSER8_MEM 数据输出
Q1	Output	OSER8_MEM 三态使能数据输出，可连接 Q0 所连的 IOBUF/TBUF 的 OEN 信号，或悬空

参数介绍

表 1-53 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 1'b0:数据上升沿输出； 1'b1:数据下降沿输出
TCLK_SOURCE	"DQSW","DQSW270"	" DQSW "	TCLK 来源选择 "DQSW": 来自 DQS 模块的 DQSW0； DQSW270": 来自 DQS 模块的 DQSW270
HWL	"false", "true"	"false"	OSER8_MEM 数据 d_up0/1 时序关系控制 "false": d_up1 比 d_up0 提前一个周期； "true": d_up1 和 d_up0 时序相同

连接合法性规则

- 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
        );
```

1.2.21 IODELAY

原语名称

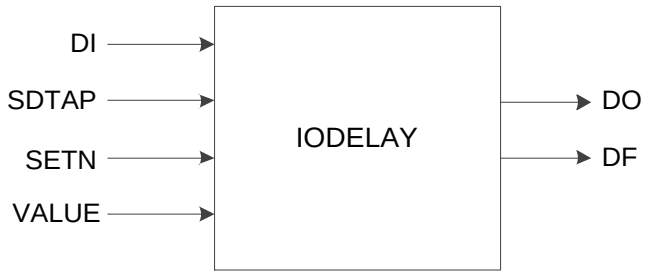
IODELAY(Input/Output delay)输入输出延时，是 IO 模块所包含的一个可编程延时单元。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 1-43 IODELAY 端口示意图



功能描述

每个 IO 都包含 IODELAY 模块，总共提供 128 步的延迟，GW1N 系列 FPGA 一步的延迟时间约为 30ps，GW2A 系列 FPGA 一步的延迟时间约为 18ps。IODELAY 可用于 I/O 逻辑的输入或输出，但不能同时作用。

端口介绍

表 1-54 端口介绍

端口名	I/O	描述
DI	Input	数据输入
SDTAP	Input	控制加载静态延时步长 0：加载静态延时 1：动态调整延时
SETN	Input	设置动态调整延时的方向 0: 增加延时； 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
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 1-55 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
        );
```

1.2.22 IODELAYA

原语名称

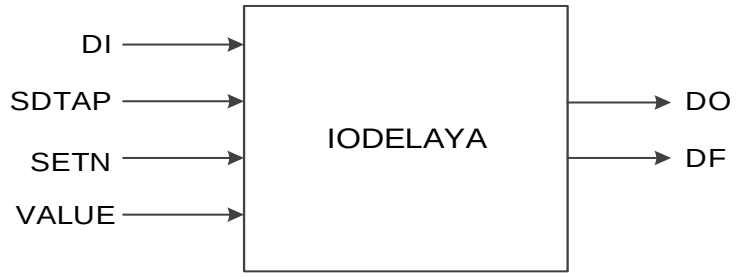
IODELAYA(Input/Output delay)输入输出延时，是 IO 模块所包含的一个可编程延时单元。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 1-44 IODELAYA 端口示意图



功能描述

IODELAYA 提供 128 步的延迟，GW1N 系列 FPGA 一步的延迟时间约为 30ps，GW2A 系列 FPGA 一步的延迟时间约为 18ps，其 delay code 在数据的下降沿变化，并且不会产生毛刺。IODELAYA 可用于 I/O 逻辑的输入或输出，但不能同时作用。

端口介绍

表 1-56 端口介绍

端口名	I/O	描述
DI	Input	数据输入
SDTAP	Input	控制加载静态延时步长 0：加载静态延时 1：动态调整延时
SETN	Input	设置动态调整延时的方向 0: 增加延时； 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
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 1-57 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：

```
IODELAYA iodelaya_inst(
    .DO(dout),
    .DF(df),
```

```
        .DI(di),
        .SDTAP(sdtap),
        .SETN(setn),
        .VALUE(value)
    );
    defparam iodelaya_inst.C_STATIC_DLY=0;
```

Vhdl 例化：

```
    COMPONENT IODELAYA
              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:IODELAYA
             GENERIC MAP (C_STATIC_DLY=>0
            )
             PORT MAP (
                  DO=>dout,
                  DF=>df,
                  DI=>di,
                  SDTAP=>sdtap,
                  SETN=>setn,
                  VALUE=>value
            );
```

1.2.23 IEM

原语名称

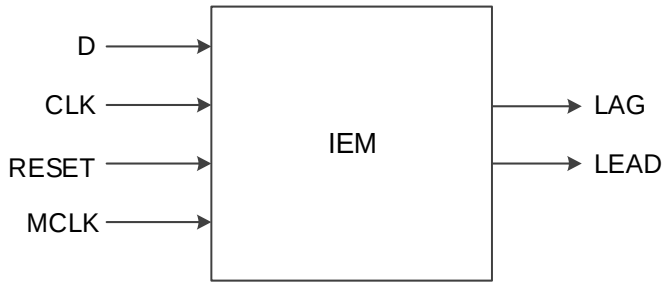
IEM(Input Edge Monitor)输入边沿监测，是 IO 模块所包含的一个取样模块。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 1-45 IEM 端口示意图



功能描述

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

端口介绍

表 1-58 端口介绍

端口名	I/O	描述
D	Input	数据输入
CLK	Input	时钟输入
RESET	Input	异步复位输入，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 1-59 参数介绍

参数名	取值范围	默认值	描述
WINSIZE	"SMALL","MIDSMALL", "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";
```

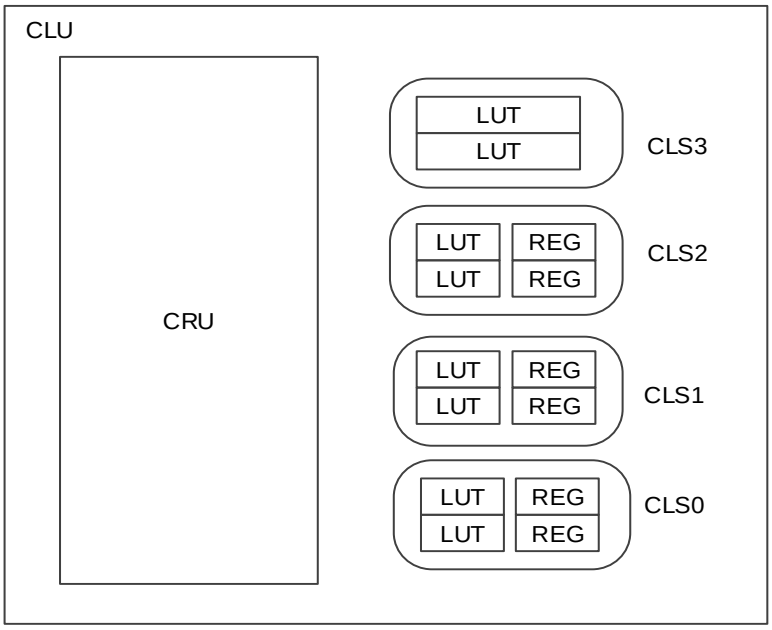
Vhdl 例化：

```
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
       );
```

2 CLU

可配置逻辑单元 CLU(Configurable Logic Unit)是构成 FPGA 产品的基本单元，每个 CLU 由四个可配置功能部分 CLS(Configurable Logic Section)和一个可配置绕线单元 CRU(Configurable Routing Unit)组成，CLU 的结构示意图如图 2-1 所示。其中可配置功能部分可配置查找表 LUT、2 输入算术逻辑单元 ALU 和寄存器 REG。CLU 模块可实现 MUX/LUT/ALU/FF/LATCH 等模块的功能。

图 2-1 CLU 结构示意图



2.1 LUT

输入查找表 LUT，常用的 LUT 结构有 LUT1、LUT2、LUT3、LUT4，其区别在于查找表输入位宽的不同。

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、

GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

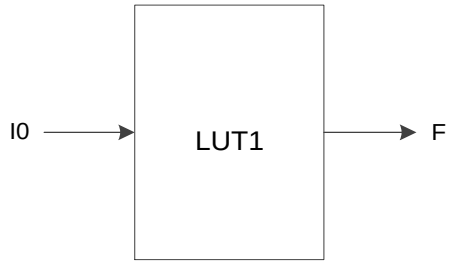
2.1.1 LUT1

原语介绍

LUT1(1-input Look-up Table)是其中最简单的一种，常用于实现缓冲器和反相器。LUT1 为 1 输入的查找表，通过 parameter 给 INIT 赋初值后，根据输入的地址查找对应的数据并输出结果。

结构框图

图 2-2 LUT1 结构框图



Port 介绍

表 2-1 Port 介绍

Port Name	I/O	Description
I0	Input	Data Input
F	Output	Data Output

Attribute 介绍

表 2-2 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	2'h0~2'h3	2'h0	Initial value for LUT1

真值表

表 2-3 真值表

Input(I0)	Output(F)
0	INIT[0]
1	INIT[1]

原语例化

Verilog 例化：

```
LUT1 instName (
        .I0(I0),
        .F(F)
);
defparam instName.INIT=2'h1;
```

Vhdl 例化：

```
COMPONENT LUT1
        GENERIC (INIT:bit_vector:=X"0");
        PORT(
                F:OUT std_logic;
                I0:IN std_logic
            );
END COMPONENT;
uut:LUT1
        GENERIC MAP(INIT=>X"0")
        PORT MAP (
            F=>F,
            I0=>I0
          );
```

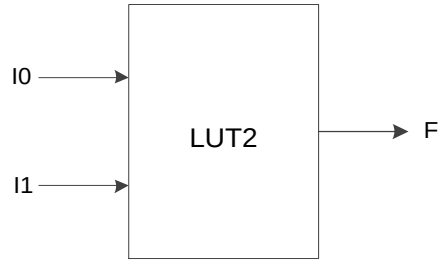
2.1.2 LUT2

原语介绍

LUT2(2-input Look-up Table)为 2 输入的查找表，通过 parameter 给 INIT 赋初值后，根据输入的地址查找对应的数据并输出结果。

结构框图

图 2-3 LUT2 结构框图



Port 介绍

表 2-4 Port 介绍

Port Name	I/O	Description
I0	Input	Data Input
I1	Input	Data Input
F	Output	Data Output

Attribute 介绍

表 2-5 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	4’h0~4’hf	4’h0	Initial value for LUT2

真值表

表 2-6 真值表

Input(I1)	Input(I0)	Output(F)
0	0	INIT[0]
0	1	INIT[1]
1	0	INIT[2]
1	1	INIT[3]

原语例化

Verilog 例化：

```
LUT2 instName (
        .I0(I0),
        .I1(I1),
        .F(F)
);
defparam instName.INIT=4'h1;
```

Vhdl 例化：

```
    COMPONENT LUT2
        GENERIC (INIT:bit_vector:=X"0");
        PORT(
                F:OUT std_logic;
                I0:IN std_logic;
                I1:IN std_logic
        );
END COMPONENT;
uut:LUT2
        GENERIC MAP(INIT=>X"0")
        PORT MAP (
            F=>F,
            I0=>I0,
            I1=>I1
        );
```

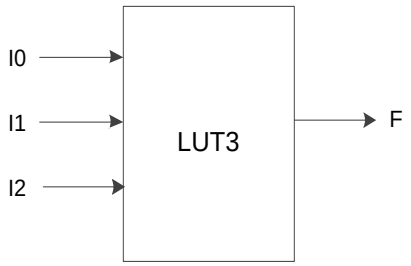
2.1.3 LUT3

原语介绍

LUT3(3-input Look-up Table)为 3 输入的查找表，通过 parameter 给 INIT 赋初值后，根据输入的地址查找对应的数据并输出结果。

结构框图

图 2-4 LUT3 结构框图



Port 介绍

表 2-7 Port 介绍

Port Name	I/O	Description
I0	Input	Data Input
I1	Input	Data Input
I2	Input	Data Input
F	Output	Data Output

Attribute 介绍

表 2-8 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	8'h00~8'hff	8'h00	Initial value for LUT3

真值表

表 2-9 真值表

Input(I2)	Input(I1)	Input(I0)	Output(F)
0	0	0	INIT[0]
0	0	1	INIT[1]
0	1	0	INIT[2]
0	1	1	INIT[3]
1	0	0	INIT[4]
1	0	1	INIT[5]
1	1	0	INIT[6]
1	1	1	INIT[7]

原语例化

Verilog 例化：

```
LUT3 instName (
        .I0(I0),
        .I1(I1),
```

```
        .I2(I2),
        .F(F)
);
defparam instName.INIT=8'h10;
```

Vhdl 例化

```
COMPONENT LUT3
        GENERIC (INIT:bit_vector:=X"00");
        PORT(
                F:OUT std_logic;
                I0:IN std_logic;
                I1:IN std_logic;
                I2:IN std_logic
        );
END COMPONENT;
uut:LUT3
        GENERIC MAP(INIT=>X"00")
        PORT MAP (
            F=>F,
            I0=>I0,
            I1=>I1,
            I2=>I2
        );
```

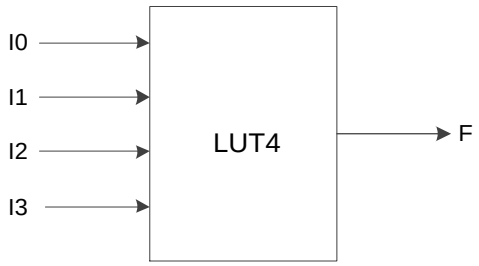
2.1.4 LUT4

原语介绍

LUT4(4-input Look-up Table)为 4 输入的查找表，通过 parameter 给 INIT 赋初值后，根据输入的地址查找对应的数据并输出结果。

结构框图

图 2-5 LUT4 结构框图



Port 介绍

表 2-10 Port 介绍

Port Name	I/O	Description
I0	Input	Data Input
I1	Input	Data Input
I2	Input	Data Input
I3	Input	Data Input
F	Output	Data Output

Attribute 介绍

表 2-11 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	16'h0000~16'hffff	16'h0000	Initial value for LUT4

真值表

表 2-12 真值表

Input(I3)	Input(I2)	Input(I1)	Input(I0)	Output(F)
0	0	0	0	INIT[0]
0	0	0	1	INIT[1]
0	0	1	0	INIT[2]
0	0	1	1	INIT[3]
0	1	0	0	INIT[4]
0	1	0	1	INIT[5]
0	1	1	0	INIT[6]
0	1	1	1	INIT[7]
1	0	0	0	INIT[8]
1	0	0	1	INIT[9]
1	0	1	0	INIT[10]
1	0	1	1	INIT[11]
1	1	0	0	INIT[12]
1	1	0	1	INIT[13]
1	1	1	0	INIT[14]
1	1	1	1	INIT[15]

原语例化

Verilog 例化：

```
LUT4 instName (
        .I0(I0),
        .I1(I1),
        .I2(I2),
        .I3(I3),
        .F(F)
);
defparam instName.INIT=16'h1011;
```

Vhdl 例化：

```
COMPONENT LUT4
        GENERIC (INIT:bit_vector:=X"0000");
        PORT(
                F:OUT std_logic;
                I0:IN std_logic;
                I1:IN std_logic;
```

```
                I2:IN std_logic;
                I3:IN std_logic
        );
END COMPONENT;
uut:LUT4
        GENERIC MAP(INIT=>X"0000")
        PORT MAP (
            F=>F,
            I0=>I0,
            I1=>I1,
            I2=>I2,
            I3=>I3
        );
```

2.1.5 Wide LUT

原语介绍

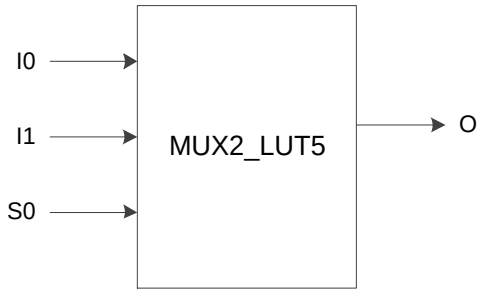
Wide LUT 是通过 LUT4 和 MUX2 构造高阶 LUT，高云 FPGA 目前支持的构造高阶 LUT 的 MUX2 有 MUX2_LUT5/ MUX2_LUT6/ MUX2_LUT7/ MUX2_LUT8。

高阶 LUT 的构造方式如下：两个 LUT4 和 MUX2_LUT5 可组合实现 LUT5，两个组合实现的 LUT5 和 MUX2_LUT6 可组合实现 LUT6，两个组合实现的 LUT6 和 MUX2_LUT7 可组合实现 LUT7，两个组合实现的 LUT7 和 MUX2_LUT8 可组合实现 LUT8。

以 MUX2_LUT5 为例介绍 Wide LUT 的使用。

结构框图

图 2-6 MUX2_LUT5 结构框图



Port 介绍

表 2-13 Port 介绍

Port Name	I/O	Description
I0	Input	Data Input
I1	Input	Data Input
S0	Input	Select Signal Input
O	Output	Data Output

真值表

表 2-14 真值表

Input(S0)	Output(O)
0	I0
1	I1

原语例化

Verilog 例化：

```
MUX2_LUT5 instName (
    .I0(f0),
    .I1(f1),
    .S0(i5),
    .O(o)
);
LUT4 lut_0 (
    .I0(i0),
    .I1(i1),
    .I2(i2),
    .I3(i3),
    .F(f0)
);
defparam lut_0.INIT=16'h184A;
LUT4 lut_1 (
    .I0(i0),
    .I1(i1),
    .I2(i2),
    .I3(i3),
    .F(f1)
);
defparam lut_1.INIT=16'h184A;
```

Vhdl 例化：

```
COMPONENT MUX2_LUT5
        PORT(
                O:OUT std_logic;
                I0:IN std_logic;
                I1:IN std_logic;
                S0:IN std_logic
        );
END COMPONENT;
COMPONENT LUT4
        PORT(
                F:OUT std_logic;
                I0:IN std_logic;
                I1:IN std_logic;
                I2:IN std_logic;
                I3:IN std_logic
        );
```

```
END COMPONENT;
uut0: MUX2_LUT5
        PORT MAP (
            O=>o,
            I0=>f0,
            I1=>f1,
            S0=>i5
        );
uut1:LUT4
        GENERIC MAP(INIT=>X"0000")
        PORT MAP (
            F=>f0,
            I0=>i0,
            I1=>i1,
            I2=>i2,
            I3=>i3
        );
uut2:LUT4
        GENERIC MAP(INIT=>X"0000")
        PORT MAP (
            F=>f1,
            I0=>i0,
            I1=>i1,
            I2=>i2,
            I3=>i3
        );
```

2.2 MUX

MUX 是多路复用器，拥有多路输入，通过通道选择信号确定其中一路数据传送到输出端。高云原语中有 2 选 1 和 4 选 1 两种多路复用器。

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

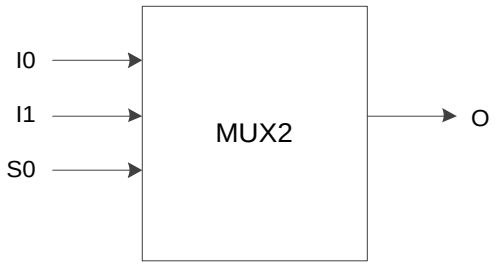
2.2.1 MUX2

原语介绍

MUX2(2-to-1 Multiplexer)是 2 选 1 的复用器，根据选择信号，从两个输入中选择其中一个作为输出。

结构框图

图 2-7 MUX2 结构框图



Port 介绍

表 2-15 Port 介绍

Port Name	I/O	Description
I0	Input	Data Input
I1	Input	Data Input
S0	Input	Select Signal Input
O	Output	Data Output

真值表

表 2-16 真值表

Input(S0)	Output(O)
0	I0
1	I1

原语例化

Verilog 例化：

```
MUX2 instName (
        .I0(I0),
        .I1(I1),
        .S0(S0),
        .O(O)
);
```

Vhdl 例化：

```
COMPONENT MUX2
        PORT(
                O:OUT std_logic;
                I0:IN std_logic;
                I1:IN std_logic;
                S0:IN std_logic
        );
END COMPONENT;
uut:MUX2
        PORT MAP (
```

```
        O=>O,
        I0=>I0,
        I1=>I1,
        S0=>S0
);
```

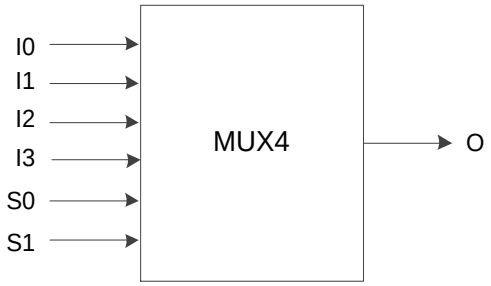
2.2.2 MUX4

原语介绍

MUX4(4-to-1 Multiplexer)是 4 选 1 的多路复用器，根据选择信号，从四个输入中选择其中一个作为输出。

结构框图

图 2-8 MUX4 结构框图



Port 介绍

表 2-17 Port 介绍

Port Name	I/O	Description
I0	Input	Data Input
I1	Input	Data Input
I2	Input	Data Input
I3	Input	Data Input
S0	Input	Select Signal Input
S1	Input	Select Signal Input
O	Output	Data Output

真值表

表 2-18 真值表

Input(S1)	Input(S0)	Output(O)
0	0	I0
0	1	I1
1	0	I2
1	1	I3

原语例化

Verilog 例化：

```
MUX4 instName (
    .I0(I0),
    .I1(I1),
    .I2(I2),
    .I3(I3),
    .S0(S0),
    .S1(S1),
    .O(O)
);
```

Vhdl 例化：

```
COMPONENT MUX4
        PORT(
                O:OUT std_logic;
                I0:IN std_logic;
                I1:IN std_logic;
                I2:IN std_logic;
                I3:IN std_logic;
                S0:IN std_logic;
                S1:IN std_logic
        );
END COMPONENT;
uut:MUX4
        PORT MAP (
            O=>O,
            I0=>I0,
            I1=>I1,
            I2=>I2,
            I3=>I3,
            S0=>S0,
            S1=>S1
        );
```

2.2.3 Wide MUX

原语介绍

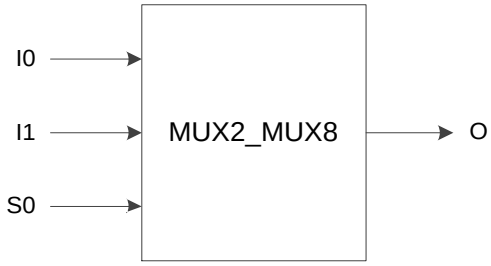
Wide MUX 是通过 MUX4 和 MUX2 构造高阶 MUX，高云 FPGA 目前支持的构造高阶 MUX 的 MUX2 有 MUX2_MUX8/ MUX2_MUX16/ MUX2_MUX32。

高阶 MUX 的构造方式如下：两个 MUX4 和 MUX2_MUX8 可组合实现 MUX8，两个组合实现的 MUX8 和 MUX2_MUX16 可组合实现 MUX16，两个组合实现的 MUX16 和 MUX2_MUX32 可组合实现 MUX32。

以 MUX2_MUX8 为例介绍 Wide MUX 的使用。

结构框图

图 2-9 MUX2_MUX8 结构框图



Port 介绍

表 2-19 Port 介绍

Port Name	I/O	Description
I0	Input	Data Input
I1	Input	Data Input
S0	Input	Select Signal Input
O	Output	Data Output

真值表

表 2-20 真值表

Input(S0)	Output(O)
0	I0
1	I1

原语例化

Verilog 例化：

```
MUX2_MUX8 instName (
    .I0(o0),
    .I1(o1),
    .S0(S2),
    .O(O)
);
MUX4 mux_0 (
    .I0(i0),
    .I1(i1),
    .I2(i2),
    .I3(i3),
    .S0(s0),
    .S1(s1),
    .O(o0)
);
MUX4 mux_1 (
    .I0(i4),
```

```
        .I1(i5),
        .I2(i6),
        .I3(i7),
        .S0(s0),
        .S1(s1),
        .O(o1)
    );
```

Vhdl 例化：

```
    COMPONENT MUX2_MUX8
            PORT(
                    O:OUT std_logic;
                    I0:IN std_logic;
                    I1:IN std_logic;
                    S0:IN std_logic
            );
    END COMPONENT；
    COMPONENT MUX4
            PORT(
                    O:OUT std_logic;
                    I0:IN std_logic;
                    I1:IN std_logic;
                    I2:IN std_logic;
                    I3:IN std_logic;
                    S0:IN std_logic;
                    S1:IN std_logic
            );
    END COMPONENT;
    uut1:MUX2_MUX8
            PORT MAP (
                O=>O,
                I0=>o0,
                I1=>o1,
                S0=>S2
            );
    uut2:MUX4
            PORT MAP (
                O=>o0,
                I0=>I0,
                I1=>I1,
                I2=>I2,
                I3=>I3,
                S0=>S0,
                S1=>S1
            );
    uut3:MUX4sss
            PORT MAP (
                O=>o1,
                I0=>I4,
                I1=>I5,
                I2=>I6,
```

```
        I3=>I7,
        S0=>S0,
        S1=>S1
    );
```

2.3 ALU

原语介绍

ALU(2-input Arithmetic Logic Unit)2 输入算术逻辑单元，实现了 ADD/SUB/ADDSUB 等功能。

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

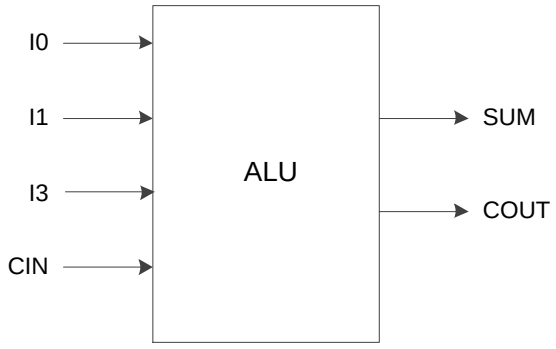
具体功能如表 2-21 所示。

表 2-21 ALU 功能

项目	描述
ADD	加法运算
SUB	减法运算
ADDSUB	加/减法运算
CUP	加计数器
CDN	减计数器
CUPCDN	加/减计数器
GE	大于比较器
NE	不等于比较器
LE	小于比较器
MULT	乘法器

结构框图

图 2-10 ALU 结构框图



Port 介绍

表 2-22 Port 介绍

Port Name	Input/Output	Description
I0	Input	Data Input
I1	Input	Data Input
I3	Input	Data Input
CIN	Input	Carry Input
COUT	Output	Carry Output
SUM	Output	Data Output

Attribute 介绍

表 2-23 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
ALU_MODE	0,1,2,3,4,5,6,7,8,9	0	Select the function of arithmetic. 0:ADD; 1:SUB; 2:ADDSUB; 3:NE; 4:GE; 5:LE; 6:CUP; 7:CDN; 8:CUPCDN; 9:MULT

原语例化

Verilog 例化：

```
ALU instName (
    .I0(I0),
    .I1(I1),
    .I3(I3),
    .CIN(CIN),
    .COUT(COUT),
    .SUM(SUM)
);
defparam instName.ALU_MODE=1;
```

Vhdl 例化：

```
COMPONENT ALU
    GENERIC (ALU_MODE:integer:=0);
        PORT(
            COUT:OUT std_logic;
            SUM:OUT std_logic;
            I0:IN std_logic;
            I1:IN std_logic;
            I3:IN std_logic;
```

```
                CIN:IN std_logic
            );
END COMPONENT;
uut:ALU
    GENERIC MAP(ALU_MODE=>1)
      PORT MAP (
          COUT=>COUT,
          SUM=>SUM,
          I0=>I0,
          I1=>I1,
          I3=>I3,
          CIN=>CIN
        );
```

2.4 FF

触发器是时序电路中常用的基本元件，FPGA 内部的时序逻辑都可通过 FF 结构实现，常用的 FF 有 DFF、DFFE、DFFS、DFFSE 等，其区别在于复位方式、触发方式等方面。

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。与 FF 相关的原语有 20 个，如表 2-24 所示。

表 2-24 与 FF 相关的原语

原语	描述
DFF	D 触发器
DFFE	带时钟使能 D 触发器
DFFS	带同步置位 D 触发器
DFFSE	带时钟使能、同步置位 D 触发器
DFFR	带同步复位 D 触发器
DFFRE	带时钟使能、同步复位 D 触发器
DFFP	带异步置位 D 触发器
DFFPE	带时钟使能、异步置位 D 触发器
DFFC	带异步复位 D 触发器
DFFCE	带时钟使能、异步复位 D 触发器
DFFN	下降沿 D 触发器
DFFNE	下降沿带时钟使能 D 触发器
DFFNS	下降沿带同步置位 D 触发器
DFFNSE	下降沿带时钟使能、同步置位 D 触发器
DFFNR	下降沿带同步复位 D 触发器
DFFNRE	下降沿带时钟使能、同步复位 D 触发器
DFFNP	下降沿带异步置位 D 触发器

原语	描述
DFFNPE	下降沿带时钟使能、异步置位 D 触发器
DFFNC	下降沿带异步复位 D 触发器
DFFNCE	下降沿带时钟使能、异步复位 D 触发器

放置规则

表 2-25 FF 类型

编号	类型 1	类型 2
1	DFFS	DFFR
2	DFFSE	DFFRE
3	DFFP	DFFC
4	DFFPE	DFFCE
5	DFFNS	DFFNR
6	DFFNSE	DFFNRE
7	DFFNP	DFFNC
8	DFFNPE	DFFNCE

1. 相同类型的 DFF，可以放置在同一个 CLS 的 2 个 FF 上，除数据输入 pin 外的其它输入必须共线；
2. 不同类型的 DFF，表 2-25 中同一编号的两种类型可以放置在同一个 CLS 的 2 个 FF 上，除数据输入 pin 外的其它输入必须共线；
3. 可以约束 DFF 和 ALU 在同一个 CLS 的相同或不同位置；
4. 可以约束 DFF 和 LUT 在同一个 CLS 的相同或不同位置。

注！

共线是指必须是同一条 net，经过反相器前后的两条 net 为不共线，不可放置在同一个 CLS。

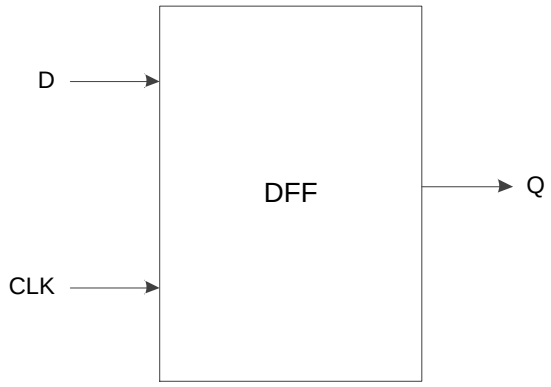
2.4.1 DFF

原语介绍

DFF(D Flip-Flop)是其中最简单常用的一种触发器，常用于信号采样和处理，是上升沿触发的 D 触发器。

结构框图

图 2-11 DFF 结构框图



Port 介绍

表 2-26 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
Q	Output	Data Output

Attribute 介绍

表 2-27 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFF

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DFF
        GENERIC (INIT:bit:='0');
        PORT(
                Q:OUT std_logic;
                D:IN std_logic;
                CLK:IN std_logic
);
END COMPONENT;
uut:DFF
```

```
GENERIC MAP(INIT=>'0')
PORT MAP (
    Q=>Q,
    D=>D,
    CLK=>CLK
);
```

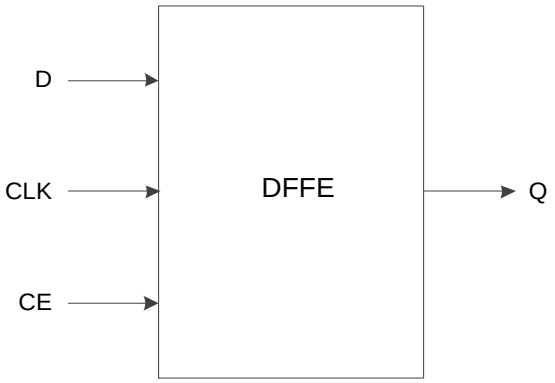
2.4.2 DFFE

原语介绍

DFFE(D Flip-Flop with Clock Enable)是上升沿触发的 D 触发器，具有时钟使能功能。

结构框图

图 2-12 DFFE 结构框图



Port 介绍

表 2-28 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
CE	Input	Clock Enable
Q	Output	Data Output

Attribute 介绍

表 2-29 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFFE

原语例化

Verilog 例化：

```
DFFE instName (
        .D(D),
        .CLK(CLK),
```

```
        .CE(CE),
        .Q(Q)
    );
    defparam instName.INIT=1'b0;
```

Vhdl 例化：

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

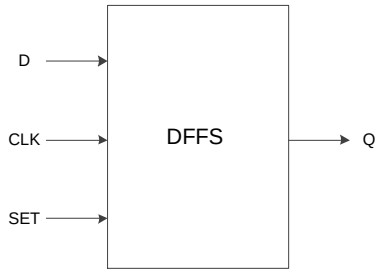
2.4.3 DFFS

原语介绍

DFFS(D Flip-Flop with Synchronous Set)是上升沿触发的 D 触发器，具有同步置位功能。

结构框图

图 2-13 DFFS 结构框图



Port 介绍

表 2-30 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
SET	Input	Synchronous Set Input
Q	Output	Data Output

Attribute 介绍

表 2-31 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for DFFS

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DFFS
        GENERIC (INIT:bit:='1');
        PORT(
                Q:OUT std_logic;
                D:IN std_logic;
                CLK:IN std_logic;
                SET:IN std_logic
        );
END COMPONENT;
uut:DFFS
        GENERIC MAP(INIT=>'1')
        PORT MAP (
            Q=>Q,
            D=>D,
            CLK=>CLK,
            SET=>SET
        );
```

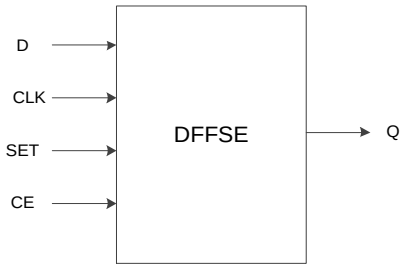
2.4.4 DFFSE

原语介绍

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

结构框图

图 2-14 DFFSE 结构框图



Port 介绍

表 2-32 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
SET	Input	Synchronous Set Input
CE	Input	Clock Enable
Q	Output	Data Output

Attribute 介绍

表 2-33 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1’b0,1’b1	1’b1	Initial value for DFFSE

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DFFSE
        GENERIC (INIT:bit:=’1’);
        PORT(
                Q:OUT std_logic;
                D:IN std_logic;
                CLK:IN std_logic;
                SET:IN std_logic;
```

```
                CE:IN std_logic
            );
END COMPONENT;
uut:DFFSE
        GENERIC MAP(INIT=>'1')
        PORT MAP (
            Q=>Q,
            D=>D,
            CLK=>CLK,
            SET=>SET,
            CE=>CE
        );
```

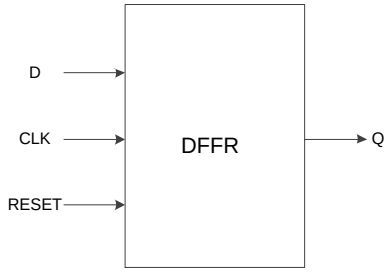
2.4.5 DFFR

原语介绍

DFFR(D Flip-Flop with Synchronous Reset)是上升沿触发的 D 触发器，具有同步复位功能。

结构框图

图 2-15 DFFR 结构框图



Port 介绍

表 2-34 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
RESET	Input	Synchronous Reset Input
Q	Output	Data Output

Attribute 介绍

表 2-35 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFFR

原语例化

Verilog 例化：

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

Vhdl 例化：

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

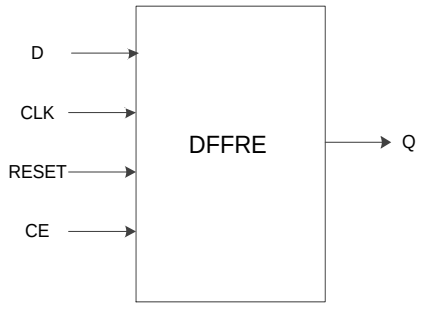
2.4.6 DFFRE

原语介绍

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

结构框图

图 2-16 DFFRE 结构框图



Port 介绍

表 2-36 Port 介绍

Port Name	I/O	Description
D	Input	Data Input

CLK	Input	Clock Input
RESET	Input	Synchronous Reset Input
CE	Input	Clock Enable
Q	Output	Data Output

Attribute 介绍

表 2-37 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFFRE

原语例化

Verilog 例化：

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

Vhdl 例化：

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

2.4.7 DFFP

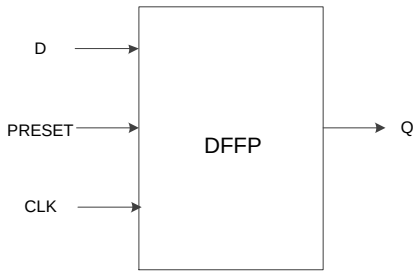
原语介绍

DFFP(D Flip-Flop with Asynchronous Preset)是上升沿触发的 D 触发器，

具有异步置位功能。

结构框图

图 2-17 DFFP 结构框图



Port 介绍

表 2-38 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
PRESET	Input	Asynchronous Preset Input
Q	Output	Data Output

Attribute 介绍

表 2-39 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for DFFP

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DFFP
        GENERIC (INIT:bit:='1');
        PORT(
            Q:OUT std_logic;
            D:IN std_logic;
            CLK:IN std_logic;
            PRESET:IN std_logic
        );
```

```
END COMPONENT;
uut:DFFP
        GENERIC MAP(INIT=>'1')
        PORT MAP (
            Q=>Q,
            D=>D,
            CLK=>CLK,
            PRESET=>PRESET
        );
```

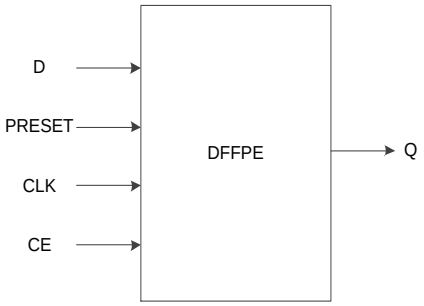
2.4.8 DFFPE

原语介绍

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

结构框图

图 2-18 DFFPE 结构框图



Port 介绍

表 2-40 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
PRESET	Input	Asynchronous Preset Input
CE	Input	Clock Enable
Q	Output	Data Output

Attribute 介绍

表 2-41 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for DFFPE

原语例化

Verilog 例化：

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

Vhdl 例化：

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

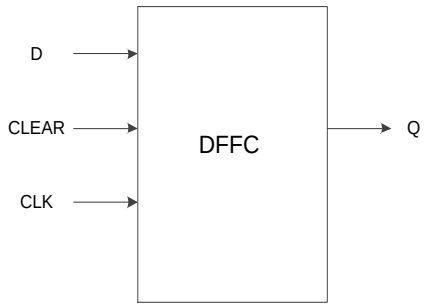
2.4.9 DFFC

原语介绍

DFFC(D Flip-Flop with Asynchronous Clear)是上升沿触发的 D 触发器，具有异步复位功能。

结构框图

图 2-19 DFFC 结构框图



Port 介绍

表 2-42 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
CLEAR	Input	Asynchronous Clear Input
Q	Output	Data Output

Attribute 介绍

表 2-43 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFFC

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DFFC
        GENERIC (INIT:bit:='0');
        PORT(
                Q:OUT std_logic;
                D:IN std_logic;
                CLK:IN std_logic;
                CLEAR:IN std_logic
        );
END COMPONENT;
uut:DFFC
        GENERIC MAP(INIT=>'0')
        PORT MAP (
            Q=>Q,
            D=>D,
            CLK=>CLK,
            CLEAR=>CLEAR
        );
```

2.4.10 DFFCE

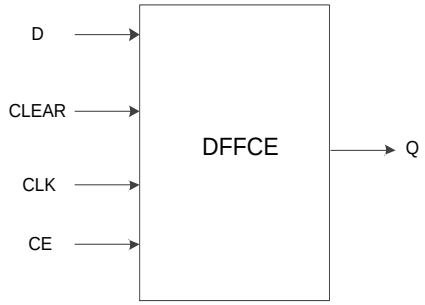
原语介绍

DFFCE(D Flip-Flop with Clock Enable and Asynchronous Clear)是上

升沿触发的 D 触发器，具有异步复位和时钟使能功能。

结构框图

图 2-20 DFFCE 结构框图



Port 介绍

表 2-44 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
CLEAR	Input	Asynchronous Clear Input
CE	Input	Clock Enable
Q	Output	Data Output

Attribute 介绍

表 2-45 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1’b0,1’b1	1’b0	Initial value for DFFCE

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DFFCE
        GENERIC (INIT:bit:=’0’);
        PORT(
            Q:OUT std_logic;
            D:IN std_logic;
```

```
                CLK:IN std_logic;
                CLEAR:IN std_logic;
                CE:IN std_logic
        );
END COMPONENT;
uut:DFFCE
        GENERIC MAP(INIT=>'0')
        PORT MAP (
            Q=>Q,
            D=>D,
            CLK=>CLK,
            CLEAR=>CLEAR,
            CE=>CE
        );
```

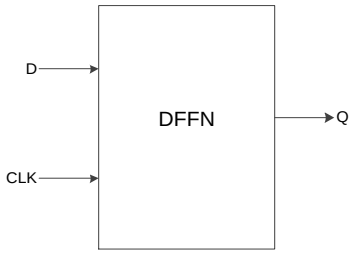
2.4.11 DFFN

原语介绍

DFFN(D Flip-Flop with Negative-Edge Clock)是下降沿触发的 D 触发器。

结构框图

图 2-21 DFFN 结构框图



Port 介绍

表 2-46 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
Q	Output	Data Output

Attribute 介绍

表 2-47 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFFN

原语例化

Verilog 例化：

```
DFFN instName (
```

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

Vhdl 例化：

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

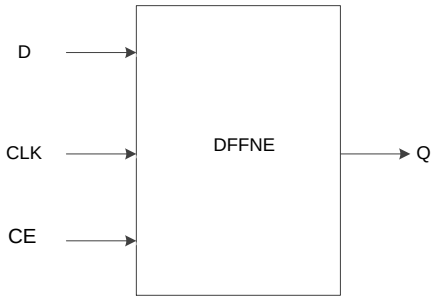
2.4.12 DFFNE

原语介绍

DFFNE(D Flip-Flop with Negative-Edge Clock and Clock Enable)是下降沿触发的 D 触发器，具有时钟使能功能。

结构框图

图 2-22 DFFNE 结构框图



Port 介绍

表 2-48 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
CE	Input	Clock Enable
Q	Output	Data Output

Attribute 介绍

表 2-49 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1’b0,1’b1	1’b0	Initial value for DFFNE

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DFFNE
        GENERIC (INIT:bit:=’0’);
        PORT(
            Q:OUT std_logic;
            D:IN std_logic;
            CLK:IN std_logic;
            CE:IN std_logic
        );
END COMPONENT;
uut:DFFNE
        GENERIC MAP(INIT=>’0’)
        PORT MAP (
            Q=>Q,
            D=>D,
            CLK=>CLK,
            CE=>CE
        );
```

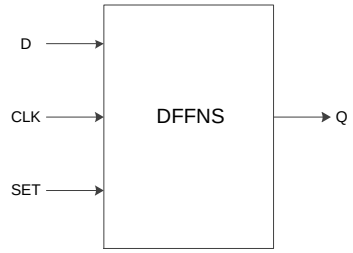
2.4.13 DFFNS

原语介绍

DFFNS(D Flip-Flop with Negative-Edge Clock and Synchronous Set)是下降沿触发的 D 触发器，具有同步置位功能。

结构框图

图 2-23 DFFNS 结构框图



Port 介绍

表 2-50 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
SET	Input	Synchronous Set Input
Q	Output	Data Output

Attribute 介绍

表 2-51 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1’b0,1’b1	1’b1	Initial value for DFFNS

原语例化

Verilog 例化：

```
DFFNS instName (
    .D(D),
    .CLK(CLK),
    .SET(SET),
    .Q(Q)
);
defparam instName.INIT=1’b1;
```

Vhdl 例化：

```
COMPONENT DFFNS
        GENERIC (INIT:bit:=’1’);
        PORT(
            Q:OUT std_logic;
            D:IN std_logic;
            CLK:IN std_logic;
            SET:IN std_logic
        );
END COMPONENT;
uut:DFFNS
        GENERIC MAP(INIT=>’1’)
```

```
    PORT MAP (
        Q=>Q,
        D=>D,
        CLK=>CLK,
        SET=>SET
    );
```

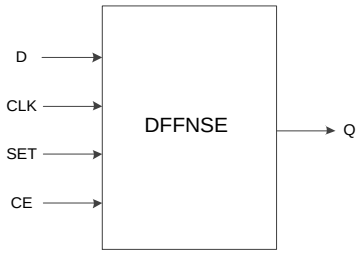
2.4.14 DFFNSE

原语介绍

DFFNSE(D Flip-Flop with Negative-Edge Clock,Clock Enable,and Synchronous Set)是下降沿触发的 D 触发器，具有同步置位和时钟使能功能。

结构框图

图 2-24 DFFNSE 结构框图



Port 介绍

表 2-52 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
SET	Input	Synchronous Set Input
CE	Input	Clock Enable
Q	Output	Data Output

Attribute 介绍

表 2-53 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1’b0,1’b1	1’b1	Initial value for DFFNSE

原语例化

Verilog 例化：

```
DFFNSE instName (
    .D(D),
    .CLK(CLK),
    .SET(SET),
    .CE(CE),
```

```
        .Q(Q)
    );
    defparam instName.INIT=1'b1;
```

Vhdl 例化：

```
    COMPONENT DFFNSE
            GENERIC (INIT:bit:='1');
            PORT(
                    Q:OUT std_logic;
                    D:IN std_logic;
                    CLK:IN std_logic;
                    SET:IN std_logic;
                    CE:IN std_logic
             );
    END COMPONENT;
    uut:DFFNSE
            GENERIC MAP(INIT=>'1')
            PORT MAP (
                Q=>Q,
                D=>D,
                CLK=>CLK,
                SET=>SET,
                CE=>CE
           );
```

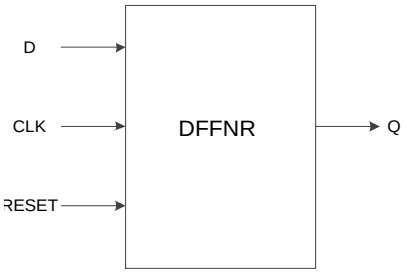
2.4.15 DFFNR

原语介绍

DFFNR(D Flip-Flop with Negative-Edge Clock and Synchronous Reset)是下降沿触发的 D 触发器，具有同步复位功能。

结构框图

图 2-25 DFFNR 结构框图



Port 介绍

表 2-54 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
RESET	Input	Synchronous Reset Input
Q	Output	Data Output

Attribute 介绍

表 2-55 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1’b0,1’b1	1’b0	Initial value for DFFNR

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DFFNR
        GENERIC (INIT:bit:=’0’);
        PORT(
                Q:OUT std_logic;
                D:IN std_logic;
                CLK:IN std_logic;
                RESET:IN std_logic
        );
END COMPONENT;
uut:DFFNR
        GENERIC MAP(INIT=>’0’)
        PORT MAP (
            Q=>Q,
            D=>D,
            CLK=>CLK,
            RESET=>RESET
        );
```

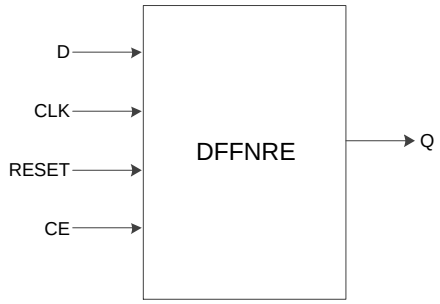
2.4.16 DFFNRE

原语介绍

DFFNRE(D Flip-Flop with Negative-Edge Clock,Clock Enable, and Synchronous Reset)是下降沿触发的 D 触发器，具有同步复位和时钟使能功能。

结构框图

图 2-26 DFFNRE 结构框图



Port 介绍

表 2-56 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
RESET	Input	Synchronous Reset Input
CE	Input	Clock Enable
Q	Output	Data Output

Attribute 介绍

表 2-57 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFFNRE

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DFFNRE
        GENERIC (INIT:bit:='0');
        PORT(
                Q:OUT std_logic;
                D:IN std_logic;
                CLK:IN std_logic;
                RESET:IN std_logic;
```

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

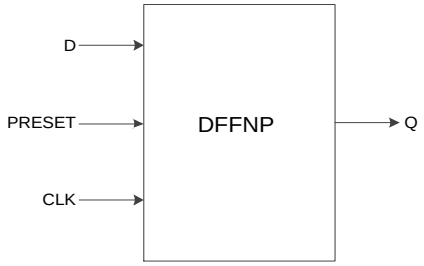
2.4.17 DFFNP

原语介绍

DFFNP(D Flip-Flop with Negative-Edge Clock and Asynchronous Preset)是下降沿触发的 D 触发器，具有异步置位功能。

结构框图

图 2-27 DFFNP 结构框图



Port 介绍

表 2-58 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
PRESET	Input	Asynchronous Preset Input
Q	Output	Data Output

Attribute 介绍

表 2-59 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for DFFNP

原语例化

Verilog 例化：

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

Vhdl 例化：

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

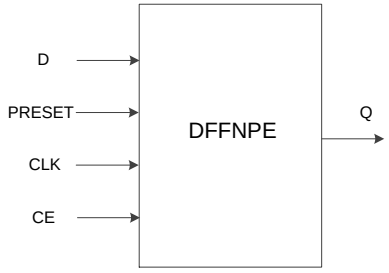
2.4.18 DFFNPE

原语介绍

DFFNPE(D Flip-Flop with Negative-Edge Clock,Clock Enable, and Asynchronous Preset)是下降沿触发的 D 触发器，具有异步置位和时钟使能功能。

结构框图

图 2-28 DFFNPE 结构框图



Port 介绍

表 2-60 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
PRESET	Input	Asynchronous Preset Input
CE	Input	Clock Enable
Q	Output	Data Output

Attribute 介绍

表 2-61 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1’b0,1’b1	1’b1	Initial value for DFFNPE

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DFFNPE
          GENERIC (INIT:bit:=’1’);
          PORT(
                Q:OUT std_logic;
                D:IN std_logic;
                CLK:IN std_logic;
                PRESET:IN std_logic;
                CE:IN std_logic
          );
END COMPONENT;
uut:DFFNPE
        GENERIC MAP(INIT=>’1’)
        PORT MAP (
            Q=>Q,
            D=>D,
            CLK=>CLK,
            PRESET=>PRESET,
            CE=>CE
        );
```

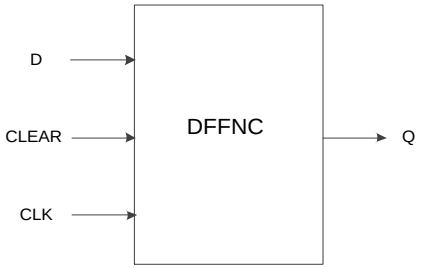
2.4.19 DFFNC

原语介绍

DFFNC(D Flip-Flop with Negative-Edge Clock and Asynchronous Clear)是下降沿触发的 D 触发器，具有异步复位功能。

结构框图

图 2-29 DFFNC 结构框图



Port 介绍

表 2-62 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
CLEAR	Input	Asynchronous Clear Input
Q	Output	Data Output

Attribute 介绍

表 2-63 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1’b0,1’b1	1’b0	Initial value for DFFNC

原语例化

Verilog 例化：

```
DFFNC instName (
        .D(D),
        .CLK(CLK),
        .CLEAR(CLEAR),
        .Q(Q)
);
defparam instName.INIT=1’b0;
```

Vhdl 例化：

```
COMPONENT DFFNC
        GENERIC (INIT:bit:=’0’);
        PORT(
                Q:OUT std_logic;
```

```
                D:IN std_logic;
                CLK:IN std_logic;
                CLEAR:IN std_logic
        );
END COMPONENT;
uut:DFFNC
        GENERIC MAP(INIT=>'0')
        PORT MAP (
            Q=>Q,
            D=>D,
            CLK=>CLK,
            CLEAR=>CLEAR
        );
```

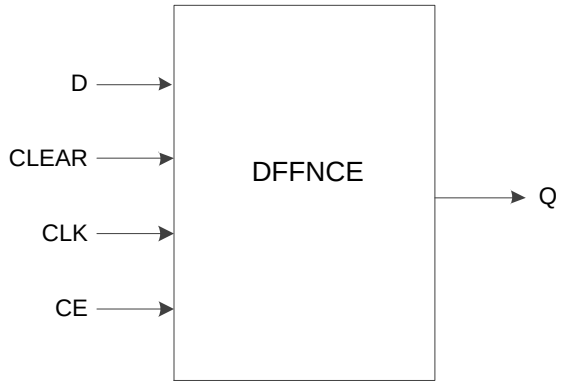
2.4.20 DFFNCE

原语介绍

DFFNCE(D Flip-Flop with Negative-Edge Clock,Clock Enable and Asynchronous Clear)是下降沿触发的 D 触发器，具有异步复位和时钟使能功能。

结构框图

图 2-30 DFFNCE 结构框图



Port 介绍

表 2-64 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLK	Input	Clock Input
CLEAR	Input	Asynchronous Clear Input
CE	Input	Clock Enable
Q	Output	Data Output

Attribute 介绍

表 2-65 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFFNCE

原语例化

Verilog 例化：

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

Vhdl 例化：

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

2.5 LATCH

锁存器是一种对电平触发的存储单元电路，其可在特定输入电平作用下改变状态。

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。与 LATCH 相关的原语有 12 个，如表 2-65 所示。

表 2-66 与 LATCH 相关的原语

原语	描述
DL	数据锁存器
DLE	带锁存使能的数据锁存器
DLC	带异步清零的数据锁存器
DLCE	带异步清零和锁存使能的数据锁存器
DLP	带异步预置位的数据锁存器
DLPE	带异步预置位和锁存使能的数据锁存器
DLN	低电平有效的数据锁存器
DLNE	带锁存使能的低电平有效的数据锁存器
DLNC	带异步清零的低电平有效的数据锁存器
DLNCE	带异步清零和锁存使能的低电平有效的数据锁存器
DLNP	带异步预置位的低电平有效的数据锁存器
DLNPE	带异步预置位和锁存使能的低电平有效的数据锁存器

放置规则

表 2-67 LATCH 类型

编号	类型 1	类型 2
1	DLC	DLP
2	DLCE	DLPE
3	DLNC	DLNP
4	DLNCE	DLNPE

1. 相同类型的 DL，可以放置在同一个 CLS 的 2 个 FF 上，除数据输入 pin 外的其它输入必须共线；
2. 不同类型的 DL，表 2-68 中同一编号的两种类型可以放置在同一个 CLS 的 2 个 FF 上，除数据输入 pin 外的其它输入必须共线；
3. 可以约束 DL 和 ALU 在同一个 CLS 的相同或不同位置；
4. 可以约束 DL 和 LUT 在同一个 CLS 的相同或不同位置。

注！

共线是指必须是同一条 net，经过反相器前后的两条 net 为不共线，不可放置在同一个 CLS。

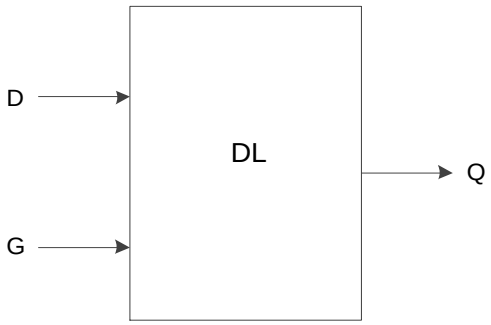
2.5.1 DL

原语介绍

DL(Data Latch)是其中最简单常用的一种锁存器，控制信号 G 高电平有效。

结构框图

图 2-31 DL 结构框图



Port 介绍

表 2-68 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
G	Input	Control Signal Input
Q	Output	Data Output

Attribute 介绍

表 2-69 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1’b0,1’b1	1’b0	Initial value for initial DL

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DL
        GENERIC (INIT:bit:=’0’);
        PORT(
                Q:OUT std_logic;
                D:IN std_logic;
                G:IN std_logic
        );
END COMPONENT;
uut:DL
        GENERIC MAP(INIT=>’0’)
        PORT MAP (
```

```
        Q=>Q,
        D=>D,
        G=>G
    );
```

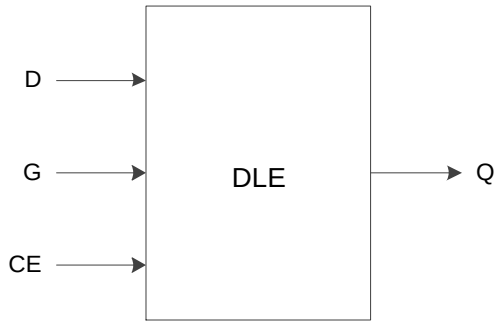
2.5.2 DLE

原语介绍

DLE(Data Latch with Latch Enable)是具有使能控制的一种锁存器，控制信号 G 高电平有效。

结构框图

图 2-32 DLE 结构框图



Port 介绍

表 2-70 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
G	Input	Control Signal Input
CE	Input	Clock Enable
Q	Output	Data Output

Attribute 介绍

表 2-71 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1’b0,1’b1	1’b0	Initial value for initial DLE

原语例化

Verilog 例化：

```
DLE instName (
        .D(D),
        .G(G),
        .CE(CE),
        .Q(Q)
);
```

```
defparam instName.INIT=1'b0;
```

Vhdl 例化：

```
COMPONENT DLE
        GENERIC (INIT:bit:='0');
        PORT(
                Q:OUT std_logic;
                D:IN std_logic;
                G:IN std_logic;
                CE:IN std_logic
        );
END COMPONENT;
uut:DLE
        GENERIC MAP(INIT=>'0')
        PORT MAP (
            Q=>Q,
            D=>D,
            G=>G,
            CE=>CE
        );
```

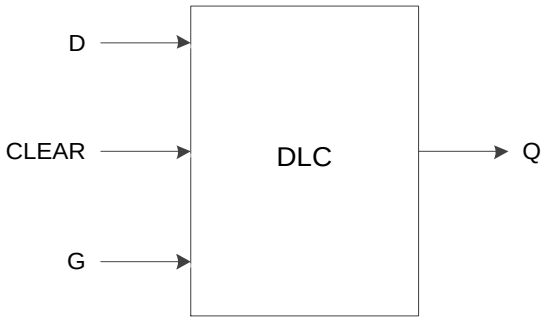
2.5.3 DLC

原语介绍

DLC(Data Latch with Asynchronous Clear)是具有复位功能的一种锁存器，控制信号 G 高电平有效。

结构框图

图 2-33 DLC 结构框图



Port 介绍

表 2-72 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLEAR	Input	Asynchronous Clear Input
G	Input	Control Signal Input
Q	Output	Data Output

Attribute 介绍

表 2-73 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1’b0,1’b1	1’b0	Initial value for initial DLC

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DLC
          GENERIC (INIT:bit:=’0’);
          PORT(
                  Q:OUT std_logic;
                  D:IN std_logic;
                  G:IN std_logic;
                  CLEAR:IN std_logic
          );
END COMPONENT;
uut:DLC
        GENERIC MAP(INIT=>’0’)
        PORT MAP (
             Q=>Q,
             D=>D,
             G=>G,
             CLEAR=>CLEAR
        );
```

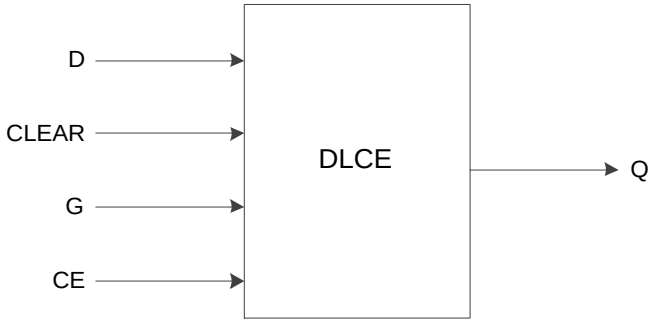
2.5.4 DLCE

原语介绍

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

结构框图

图 2-34 DLCE 结构框图



Port 介绍

表 2-74 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLEAR	Input	Asynchronous Clear Input
G	Input	Control Signal Input
CE	Input	Clock Enable
Q	Output	Data Output

Attribute 介绍

表 2-75 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1’b0,1’b1	1’b0	Initial value for initial DLCE

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DLCE
        GENERIC (INIT:bit:=’0’);
        PORT(
                Q:OUT std_logic;
                D:IN std_logic;
                G:IN std_logic;
```

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

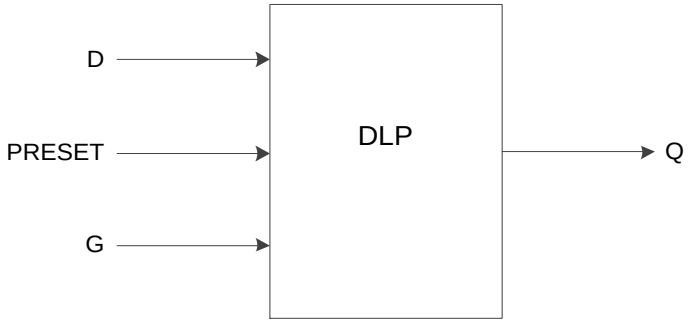
2.5.5 DLP

原语介绍

DLP(Data Latch with Asynchronous Preset)是具有置位功能的一种锁存器，控制信号 G 高电平有效。

结构框图

图 2-35 DLP 结构框图



Port 介绍

表 2-76 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
PRESET	Input	Asynchronous Preset Input
G	Input	Control Signal Input
Q	Output	Data Output

Attribute 介绍

表 2-77 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for initial DLP

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DLP
          GENERIC (INIT:bit:='1');
          PORT(
                  Q:OUT std_logic;
                  D:IN std_logic;
                  G:IN std_logic;
                  PRESET:IN std_logic
           );
END COMPONENT;
uut:DLP
         GENERIC MAP(INIT=>'1')
         PORT MAP (
              Q=>Q,
              D=>D,
              G=>G,
              PRESET => PRESET
        );
```

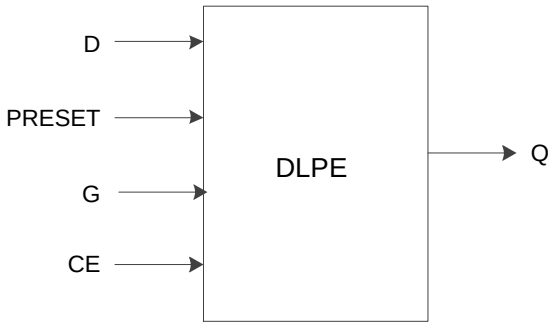
2.5.6 DLPE

原语介绍

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

结构框图

图 2-36 DLPE 结构框图



Port 介绍

表 2-78 Port 介绍

Port Name	I/O	Description
D	Input	Data Output
PRESET	Input	Asynchronous Preset Input
G	Input	Control Signal Input
CE	Input	Clock Enable
Q	Output	Data Output

Attribute 介绍

表 2-79 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1’b0,1’b1	1’b1	Initial value for initial DLPE

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DLPE
          GENERIC (INIT:bit:=’1’);
          PORT(
                Q:OUT std_logic;
                D:IN std_logic;
                G:IN std_logic;
                CE:IN std_logic;
                PRESET:IN std_logic
          );
END COMPONENT;
uut:DLPE
        GENERIC MAP(INIT=>’1’)
        PORT MAP (
             Q=>Q,
             D=>D,
             G=>G,
            CE=>CE
             PRESET =>PRESET
        );
```

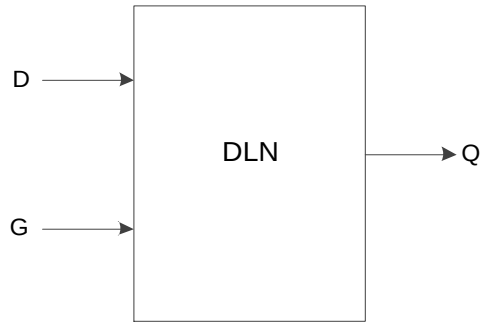
2.5.7 DLN

原语介绍

DLN(Data Latch with Inverted Gate)是控制信号低电平有效的锁存器。

结构框图

图 2-37 DLN 结构框图



Port 介绍

表 2-80 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
G	Input	Control Signal Input
Q	Output	Data Output

Attribute 介绍

表 2-81 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for initial DLN

原语例化

Verilog 例化：

```
DLN instName (
    .D(D),
    .G(G),
    .Q(Q)
);
defparam instName.INIT=1'b0;
```

Vhdl 例化：

```
COMPONENT DLN
        GENERIC (INIT:bit:='0');
        PORT(
            Q:OUT std_logic;
            D:IN std_logic;
            G:IN std_logic
```

```
        );
END COMPONENT;
uut:DLN
        GENERIC MAP(INIT=>'0')
        PORT MAP (
            Q=>Q,
            D=>D,
            G=>G
        );
```

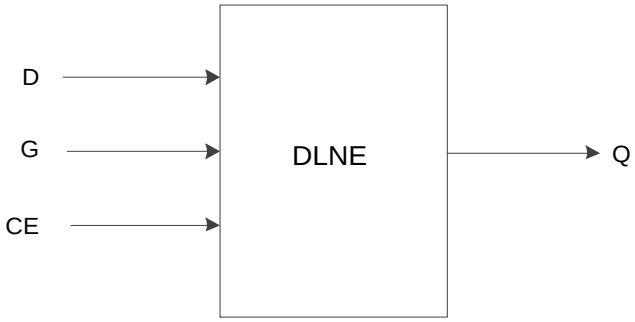
2.5.8 DLNE

原语介绍

DLNE(Data Latch with Latch Enable and Inverted Gate)是一种具有使能控制的锁存器，控制信号 G 低电平有效。

结构框图

图 2-38 DLNE 结构框图



Port 介绍

表 2-82 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
G	Input	Control Signal Input
CE	Input	Clock Enable
Q	Output	Data Output

Attribute 介绍

表 2-83 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for initial DLNE

原语例化

Verilog 例化：

```
DLNE instName (
    .D(D),
```

```
        .G(G),
        .CE(CE),
        .Q(Q)
    );
    defparam instName.INIT=1'b0;
```

Vhdl 例化：

```
    COMPONENT DLNE
            GENERIC (INIT:bit:='0');
            PORT(
                    Q:OUT std_logic;
                    D:IN std_logic;
                    G:IN std_logic;
                    CE:IN std_logic
             );
    END COMPONENT;
    uut:DLNE
            GENERIC MAP(INIT=>'0')
            PORT MAP (
                Q=>Q,
                D=>D,
                G=>G,
                CE => CE
           );
```

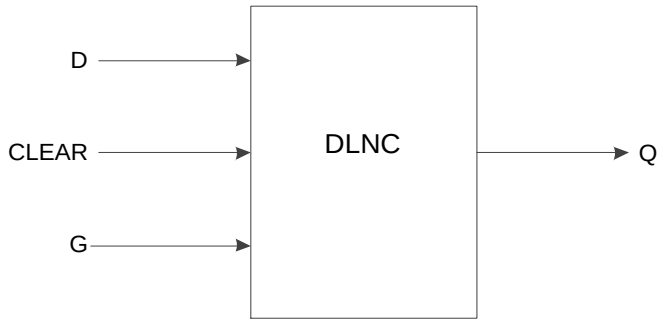
2.5.9 DLNC

原语介绍

DLNC(Data Latch with Asynchronous Clear and Inverted Gate)是一种具有复位功能的锁存器，控制信号 G 低电平有效。

结构框图

图 2-39 DLNC 结构框图



Port 介绍

表 2-84 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLEAR	Input	Asynchronous Clear Input
G	Input	Control Signal Input

Port Name	I/O	Description
Q	Output	Data Output

Attribute 介绍

表 2-85 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1’b0,1’b1	1’b0	Initial value for initial DLNC

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DLNC
          GENERIC (INIT:bit:=’0’);
          PORT(
                Q:OUT std_logic;
                D:IN std_logic;
                G:IN std_logic;
                CLEAR:IN std_logic
          );
END COMPONENT;
uut:DLNC
          GENERIC MAP(INIT=>’0’)
          PORT MAP (
               Q=>Q,
               D=>D,
               G=>G,
               CLEAR => CLEAR
          );
```

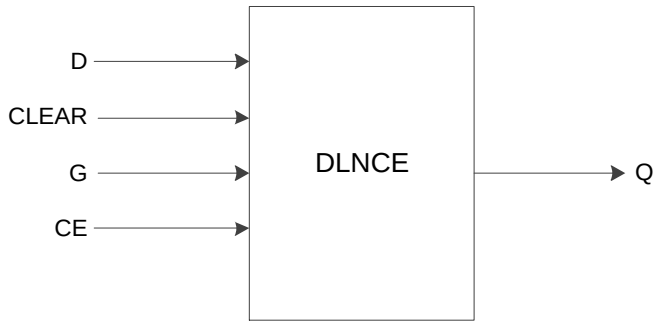
2.5.10 DLNCE

原语介绍

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

结构框图

图 2-40 DLNCE 结构框图



Port 介绍

表 2-86 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
CLEAR	Input	Asynchronous Clear Input
G	Input	Control Signal Input
CE	Input	Clock Enable
Q	Output	Data Output

Attribute 介绍

表 2-87 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1’b0,1’b1	1’b0	Initial value for initial DLNCE

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DLNCE
        GENERIC (INIT:bit:=’0’);
        PORT(
                Q:OUT std_logic;
                D:IN std_logic;
                G:IN std_logic;
```

```
                CE:IN std_logic;
                CLEAR:IN std_logic
        );
END COMPONENT;
uut:DLNCE
        GENERIC MAP(INIT=>'0'
        )
        PORT MAP (
            Q=>Q,
            D=>D,
            G=>G,
            CE=>CE,
            CLEAR=>CLEAR
        );
```

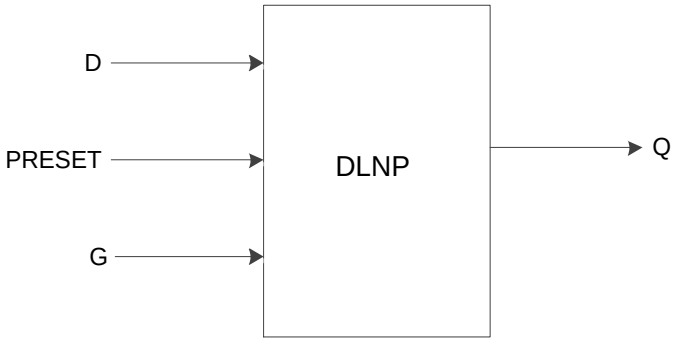
2.5.11 DLNP

原语介绍

DLNP(Data Latch with Asynchronous Clear and Inverted Gate)是具有置位功能的一种锁存器，控制信号 G 低电平有效。

结构框图

图 2-41 DLNP 结构框图



Port 介绍

表 2-88 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
PRESET	Input	Asynchronous Preset Input
G	Input	Control Signal Input
Q	Output	Data Output

Attribute 介绍

表 2-89 Attribute 介绍

Attribute Name	Allowed Values	Default	Description

INIT	1’b0,1’b1	1’b1	Initial value for initial DLNPE

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DLNP
          GENERIC (INIT:bit:=’1’);
          PORT(
                  Q:OUT std_logic;
                  D:IN std_logic;
                  G:IN std_logic;
                  PRESET:IN std_logic
          );
END COMPONENT;
uut:DLNP
          GENERIC MAP(INIT=>’1’)
          PORT MAP (
               Q=>Q,
               D=>D,
               G=>G,
               PRESET => PRESET
          );
```

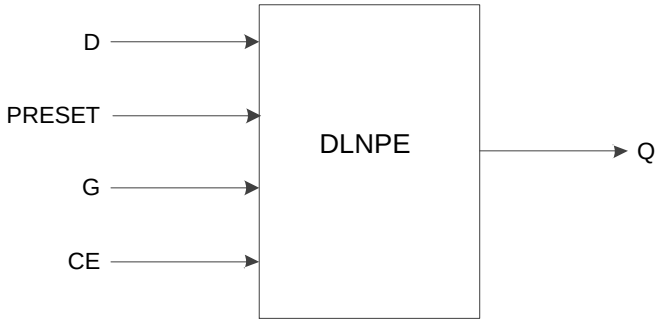
2.5.12 DLNPE

原语介绍

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

结构框图

图 2-42 DLNPE 结构框图



Port 介绍

表 2-90 Port 介绍

Port Name	I/O	Description
D	Input	Data Input
PRESET	Input	Asynchronous Preset Input
G	Input	Control Signal Input
CE	Input	Clock Enable
Q	Output	Data Output

Attribute 介绍

表 2-91 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for initial DLNPE

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DLNPE
          GENERIC (INIT:bit:='1');
          PORT(
                  Q:OUT std_logic;
                  D:IN std_logic;
                  G:IN std_logic;
                  CE:IN std_logic;
                  PRESET:IN std_logic
          );
END COMPONENT;
uut:DLNPE
          GENERIC MAP(INIT=>'1')
          PORT MAP (
               Q=>Q,
               D=>D,
               G=>G,
               CE=>CE,
               PRESET => PRESET
          );
```

3 CFU

CFU(Configurable Fuction Unit) 是可配置功能单元。与 CLU 不同的是，CFU 可配置为 SSRAM 模式。

3.1 SSRAM

SSRAM 是分布式静态随机存储器，可配置成单端口模式，半双端口模式和只读模式，如表 3-1 所示。

支持器件：GW1NZ-1、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

表 3-1 SSRAM

原语	描述
RAM16S1	地址深度 16，数据宽度为 1 的单端口 SSRAM
RAM16S2	地址深度 16，数据宽度为 2 的单端口 SSRAM
RAM16S4	地址深度 16，数据宽度为 4 的单端口 SSRAM
RAM16SDP1	地址深度 16，数据宽度为 1 的半双端口 SSRAM
RAM16SDP2	地址深度 16，数据宽度为 2 的半双端口 SSRAM
RAM16SDP4	地址深度 16，数据宽度为 4 的半双端口 SSRAM
ROM16	地址深度 16，数据宽度为 1 的只读 ROM

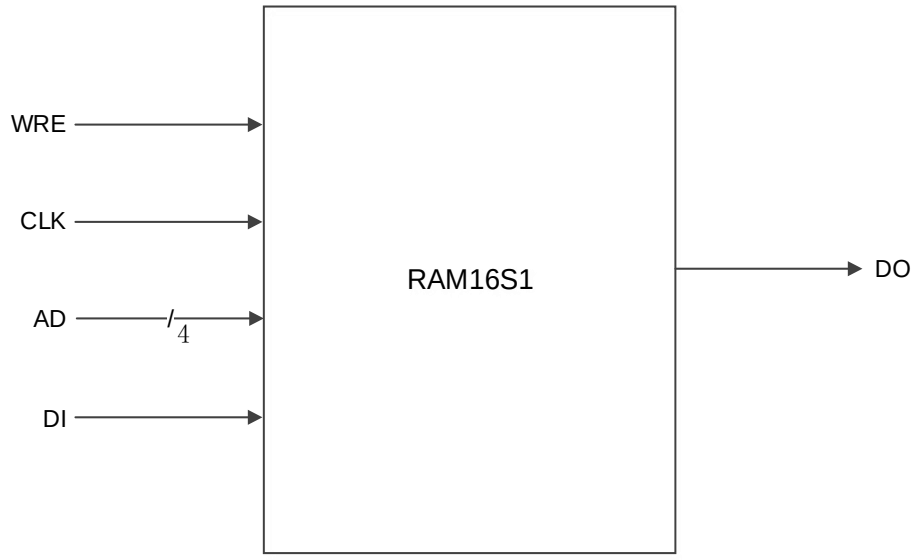
3.1.1 RAM16S1

原语介绍

RAM16S1(16-Deep by 1-Wide Single-port SSRAM)是地址深度为 16，数据位宽为 1 的单端口 SSRAM。

结构框图

图 3-1 RAM16S1 结构框图



Port 介绍

表 3-2 Port 介绍

Port Name	I/O	Description
DI	Input	Data Input
CLK	Input	Clock Input
WRE	Input	Write Enable Input
AD[3:0]	Input	Address Input
DO	Output	Data Output

Attribute 介绍

表 3-3 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT_0	16'h0000~16'hffff	16'h0000	Specifies Initial Contents of the RAM

原语例化

Verilog 例化：

```
RAM16S1 instName(
    .DI(DI),
    .WRE(WRE),
    .CLK(CLK),
    .AD(AD[3:0]),
    .DO(DOUT)
);
defparam instName.INIT_0=16'h1100;
```

Vhdl 例化：

```
COMPONENT RAM16S1
        GENERIC (INIT:bit_vector:=X"0000");
        PORT(
                DO:OUT std_logic;
                DI:IN std_logic;
                CLK:IN std_logic;
                WRE:IN std_logic;
                AD:IN std_logic_vector(3 downto 0)
        );
END COMPONENT;
uut:RAM16S1
        GENERIC MAP(INIT=>X"0000")
        PORT MAP (
            DO=>DOUT,
            DI=>DI,
            CLK=>CLK,
            WRE=>WRE,
            AD=>AD
        );
```

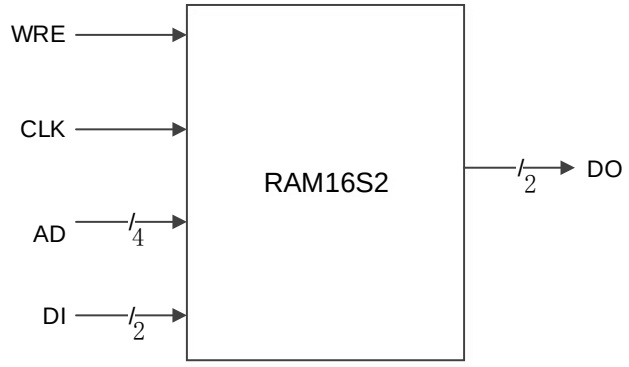
3.1.2 RAM16S2

原语介绍

RAM16S2(16-Deep by 2-Wide Single-port SSRAM)是地址深度为 16，数据位宽为 2 的单端口 SSRAM。

结构框图

图 3-2 RAM16S2 结构框图



Port 介绍

表 3-4 Port 介绍

Port Name	I/O	Description
DI[1:0]	Input	Data Input
CLK	Input	Clock Input
WRE	Input	Write Enable Input
AD[3:0]	Input	Address Input

Port Name	I/O	Description
DO[1:0]	Output	Data Output

Attribute 介绍

表 3-5 Attribute 介绍

Attribute Name	AllowedValues	Default	Description
INIT_0~ INIT_1	16'h0000~16'hffff	16'h0000	Specifies Initial Contents of the RAM

原语例化

Verilog 例化：

```
RAM16S2 instName(
      .DI(DI[1:0]),
      .WRE(WRE),
      .CLK(CLK),
      .AD(AD[3:0]),
      .DO(DOUT[1:0])
);
defparam instName.INIT_0=16'h0790;
defparam instName.INIT_1=16'h0f00;
```

Vhdl 例化：

```
COMPONENT RAM16S2
          GENERIC (INIT_0:bit_vector:=X"0000";
                       INIT_1:bit_vector:=X"0000"
          );
           PORT(
                    DO:OUT std_logic_vector(1 downto 0);
                    DI:IN std_logic_vector(1 downto 0);
                    CLK:IN std_logic;
                    WRE:IN std_logic;
                    AD:IN std_logic_vector(3 downto 0)
           );
END COMPONENT;
uut:RAM16S2
         GENERIC MAP(INIT_0=>X"0000",
                              INIT_1=>X"0000"
        )
         PORT MAP (
              DO=>DOUT,
              DI=>DI,
              CLK=>CLK,
              WRE=>WRE,
              AD=>AD
        );
```

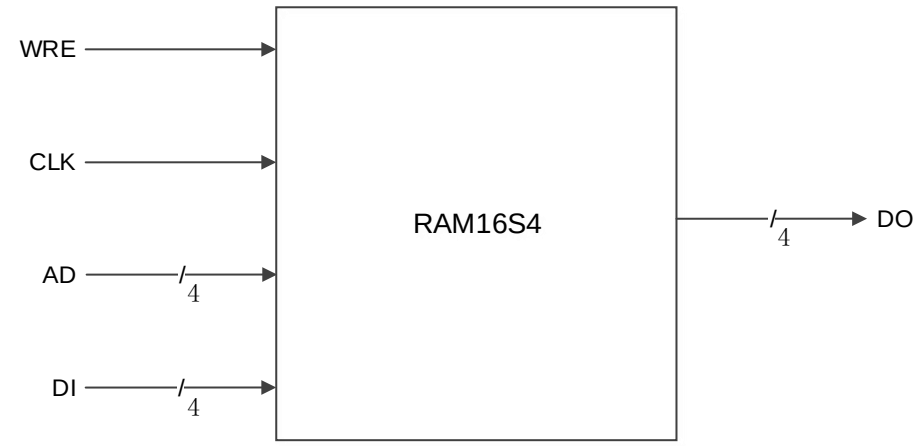
3.1.3 RAM16S4

原语介绍

RAM16S4(16-Deep by 4-Wide Single-port SSRAM)是地址深度为 16 ，数据位宽为 4 的单端口 SSRAM。

结构框图

图 3-3 RAM16S4 结构框图



Port 介绍

表 3-6 Port 介绍

Port Name	I/O	Description
DI[3:0]	Input	Data Input
CLK	Input	Clock Input
WRE	Input	Write Enable Input
AD[3:0]	Input	Address Input
DO[3:0]	Output	Data Output

Attribute 介绍

表 3-7 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT_0~ INIT_3	16’h0000~16’hffff	16’h0000	Specifies Initial Contents of the RAM

原语例化

Verilog 例化：

```
RAM16S4 instName(
    .DI(DI[3:0]),
    .WRE(WRE),
    .CLK(CLK),
    .AD(AD[3:0]),
    .DO(DOUT[3:0])
```

```
);
defparam instName.INIT_0=16'h0450;
defparam instName.INIT_1=16'h1ac3;
defparam instName.INIT_2=16'h1240;
defparam instName.INIT_3=16'h045c;
```

Vhdl 例化：

```
COMPONENT RAM16S4
        GENERIC (INIT_0:bit_vector:=X"0000";
                 INIT_1:bit_vector:=X"0000";
                 INIT_2:bit_vector:=X"0000";
                 INIT_3:bit_vector:=X"0000"
        );
         PORT(
                DO:OUT std_logic_vector(3 downto 0);
                DI:IN std_logic_vector(3 downto 0);
                CLK:IN std_logic;
                WRE:IN std_logic;
                AD:IN std_logic_vector(3 downto 0)
        );
END COMPONENT;
uut:RAM16S4
        GENERIC MAP(INIT_0=>X"0000",
                    INIT_1=>X"0000",
                    INIT_2=>X"0000",
                    INIT_3=>X"0000"
        )
         PORT MAP (
            DO=>DOUT,
            DI=>DI,
            CLK=>CLK,
            WRE=>WRE,
            AD=>AD
        );
```

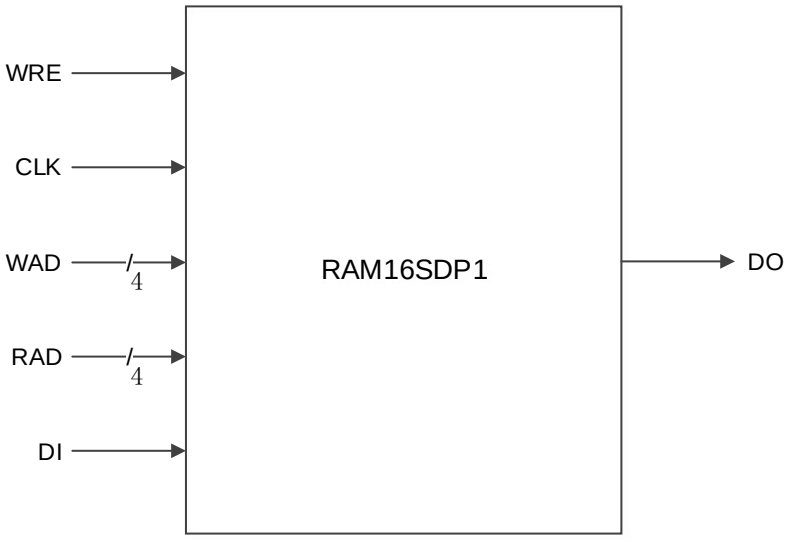
3.1.4 RAM16SDP1

原语介绍

RAM16SDP1(16-Deep by 1-Wide Semi Dual-port SSRAM)是地址深度为 16 ，数据位宽为 1 的半双端口 SSRAM。

结构框图

图 3-4 RAMSDP1 结构框图



Port 介绍

表 3-8 Port 介绍

Port Name	I/O	Description
DI	Input	Data Input
CLK	Input	Clock Input
WRE	Input	Write Enable Input
WAD[3:0]	Input	Write Address
RAD[3:0]	Input	Read Address
DO	Output	Data Output

Attribute 介绍

表 3-9 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT_0	16'h0000~16'hffff	16'h0000	Specifies Initial Contents of the RAM

原语例化

Verilog 例化：

```
RAM16SDP1 instName(
    .DI(DI),
    .WRE(WRE),
    .CLK(CLK),
    .WAD(WAD[3:0]),
    .RAD(RAD[3:0]),
    .DO(DOUT)
);
```

```
defparam instName.INIT_0=16'h0100;
```

Vhdl 例化：

```
COMPONENT RAM16SDP1
        GENERIC (INIT_0:bit_vector:=X"0000");
        PORT(
                DO:OUT std_logic;
                DI:IN std_logic;
                CLK:IN std_logic;
                WRE:IN std_logic;
                WAD:IN std_logic_vector(3 downto 0);
                RAD:IN std_logic_vector(3 downto 0)
        );
END COMPONENT;
uut:RAM16SDP1
        GENERIC MAP(INIT_0=>X"0000")
        PORT MAP (
            DO=>DOUT,
            DI=>DI,
            CLK=>CLK,
            WRE=>WRE,
            WAD=>WAD,
            RAD=>RAD
        );
```

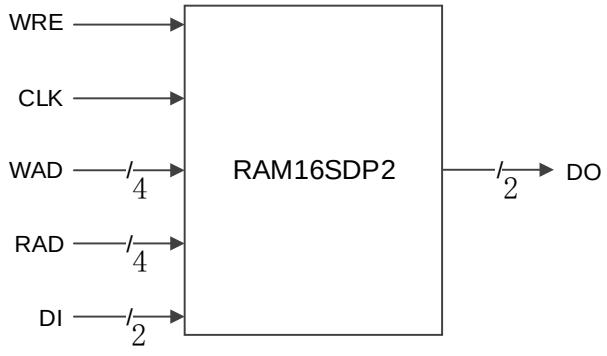
3.1.5 RAM16SDP2

原语介绍

RAM16SDP2(16-Deep by 2-Wide Semi Dual-port SSRAM)是地址深度为 16 ， 数据位宽为 2 的半双端口 SSRAM。

结构框图

图 3-5 RAM16SDP2 结构框图



Port 介绍

表 3-10 Port 介绍

Port Name	I/O	Description
DI[1:0]	Input	Data Input
CLK	Input	Clock Input

Port Name	I/O	Description
WRE	Input	Write Enable Input
WAD[3:0]	Input	Write Address
RAD[3:0]	Input	Read Address
DO[1:0]	Output	Data Output

Attribute 介绍

表 3-11 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT_0~ INIT_1	16'h0000~ 16'hffff	16'h0000	Specifies Initial Contents of the RAM

原语例化

Verilog 例化：

```
RAM16SDP2 instName(
      .DI(DI[1:0]),
      .WRE(WRE),
      .CLK(CLK),
      .WAD(WAD[3:0]),
      .RAD(RAD[3:0]),
      .DO(DOUT[1:0])
);
defparam instName.INIT_0=16'h5600;
defparam instName.INIT_1=16'h0af0;
```

Vhdl 例化：

```
COMPONENT RAM16SDP2
        GENERIC (INIT_0:bit_vector:=X"0000";
                 INIT_1:bit_vector:=X"0000"
        );
        PORT(
              DO:OUT std_logic_vector(1 downto 0);
              DI:IN std_logic_vector(1 downto 0);
              CLK:IN std_logic;
              WRE:IN std_logic;
              WAD:IN std_logic_vector(3 downto 0);
              RAD:IN std_logic_vector(3 downto 0)
        );
END COMPONENT;
uut:RAM16SDP2
        GENERIC MAP(INIT_0=>X"0000",
                    INIT_1=>X"0000"
       )
        PORT MAP (
            DO=>DOUT,
            DI=>DI,
```

```
        CLK=>CLK,
        WRE=>WRE,
        WAD=>WAD,
        RAD=>RAD
    );
```

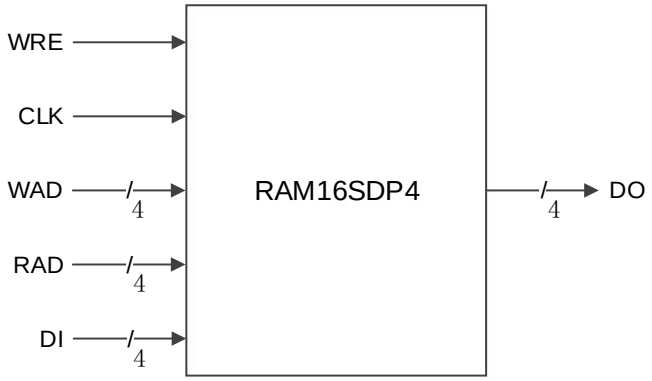
3.1.6 RAM16SDP4

原语介绍

RAM16SDP4(16-Deep by 4-Wide Semi Dual-port SSRAM)是地址深度为 16 ，数据位宽为 4 的半双端口 SSRAM。

结构框图

图 3-6 RAMSDP4 结构框图



Port 介绍

表 3-12 Port 介绍

Port Name	I/O	Description
DI[3:0]	Input	Data Input
CLK	Input	Clcok Input
WRE	Input	Write Enable Input
WAD[3:0]	Input	Write Address
RAD[3:0]	Input	Read Address
DO[3:0]	Output	Data Output

Attribute 介绍

表 3-13 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT_0~ INIT_3	16'h0000~16'hffff	16'h0000	Specifies Initial Contents of the RAM

原语例化

Verilog 例化：

```
RAM16SDP4 instName(
    .DI(DI[3:0]),
    .WRE(WRE),
    .CLK(CLK),
    .WAD(WAD[3:0]),
    .RAD(RAD[3:0]),
    .DO(DOUT[3:0])
);
defparam instName.INIT_0=16'h0340;
defparam instName.INIT_1=16'h9065;
defparam instName.INIT_2=16'hac12;
defparam instName.INIT_3=16'h034c;
```

Vhdl 例化：

```
COMPONENT RAM16SDP2
        GENERIC (INIT_0:bit_vector:=X"0000";
                 INIT_1:bit_vector:=X"0000";
                 INIT_2:bit_vector:=X"0000";
                 INIT_3:bit_vector:=X"0000";
        );
         PORT(
                DO:OUT std_logic_vector(3 downto 0);
                DI:IN std_logic_vector(3 downto 0);
                CLK:IN std_logic;
                WRE:IN std_logic;
                WAD:IN std_logic_vector(3 downto 0);
                RAD:IN std_logic_vector(3 downto 0)
        );
END COMPONENT;
uut:RAM16SDP2
        GENERIC MAP(INIT_0=>X"0000",
                    INIT_1=>X"0000",
                    INIT_2=>X"0000",
                    INIT_3=>X"0000"
       )
        PORT MAP (
            DO=>DOUT,
            DI=>DI,
            CLK=>CLK,
            WRE=>WRE,
            WAD=>WAD,
            RAD=>RAD
        );
```

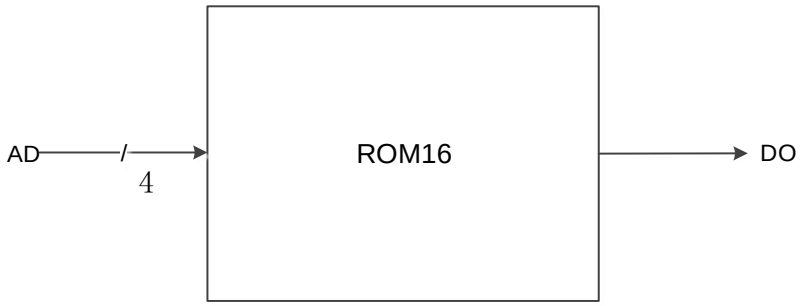
3.1.7 ROM16

原语介绍

ROM16 是地址深度为 16，数据位宽为 1 的只读存储器，存储器的内容通过 INIT 进行初始化。

结构框图

图 3-7 ROM16 结构框图



Port 介绍

表 3-14 Port 介绍

Port Name	I/O	Description
AD[3:0]	Input	Address Input
DO	Output	Data Output

Attribute 介绍

表 3-15 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
INIT_0	16'h0000~16'hffff	16'h0000	Specifies Initial Contents of the ROM

原语例化

Verilog 例化：

```
ROM16 instName (
    .AD(AD[3:0]),
    .DO(DOUT)
);
defparam instName.INIT_0=16'hfc00;
```

Vhdl 例化：

```
COMPONENT ROM16
        GENERIC (INIT:bit_vector:=X"0000");
        PORT(
            DO:OUT std_logic;
            AD:IN std_logic_vector(3 downto 0)
        );
END COMPONENT;
uut:ROM16
        GENERIC MAP(INIT=>X"0000")
        PORT MAP (
            DO=>DOUT,
            AD=>AD
        );
```

4 Block SRAM

Block SRAM—块状静态随机存储器，具有静态存取功能。根据配置模式，可分为单端口模式（SP/SPX9）、双端口模式（DP/DPX9）、半双端口模式（SDP/SDPX9）和只读模式（ROM/ROMX9）。

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

注！

- GW1N-1S、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9 不支持 DP/DPX9、DPB/DPX9B。
- GW1NZ-1 的 DP/DPB 只支持 1Kx16 模式，DPX9/DPX9B 只支持 1Kx18 模式。

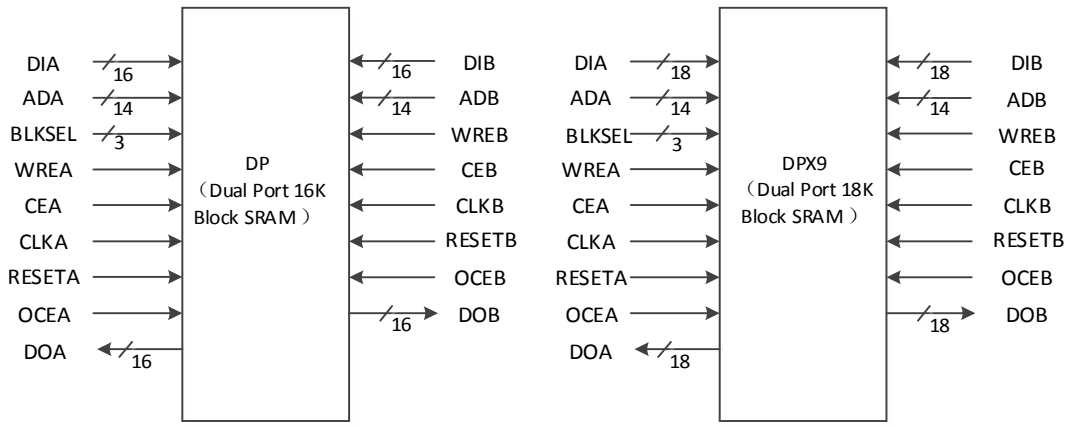
4.1 DP/DPX9

原语名称

DP/DPX9(True Dual Port 16K Block SRAM/True Dual Port 18K Block SRAM)，16K/18K 双端口 BSRAM。

端口示意图

图 4-1 DP/DPX9 端口示意图



功能描述

DP/DPX9 的存储空间分别为 16K bit/18K bit，其工作模式为双端口模式，端口 A 和端口 B 均可分别独立实现读/写操作，可支持 2 种读模式（bypass 模式和 pipeline 模式）和 3 种写模式（normal 模式、write-through 模式和 read-before-write 模式）。

DP 配置为 16bit、DPX9 配置为 18bit 时，可实现 BSRAM 的 byte enable 功能，即通过写入地址端口 AD 的低四位控制写入存储器的数据，高电平使能。ADA[0]控制 DIA[7:0]/DIA[8:0]是否写入存储器，ADA[1]控制 DIA[15:8]/DIA[17:9]是否写入存储器, ADB[0]控制 DIB[7:0]/DIB[8:0]是否写入存储器，ADB[1]控制 DIB[15:8]/DIB[17:9]是否写入存储器。

读模式

通过参数 READ_MODE0、READ_MODE1 来启用或禁用 A 端、B 端输出 pipeline 寄存器，使用输出 pipeline 寄存器时，读操作需要额外的延迟周期。

写模式

包括 normal 模式、write-through 模式和 read-before-write 模式，A 端、B 端写模式通过参数 WRITE_MODE0、WRITE_MODE1 来分别配置使用，不同模式对应的内部时序波形图如图 4-2 到图 4-7 所示。

图 4-2 DP/DPX9 Normal 写模式时序波形图（Bypass 读模式）

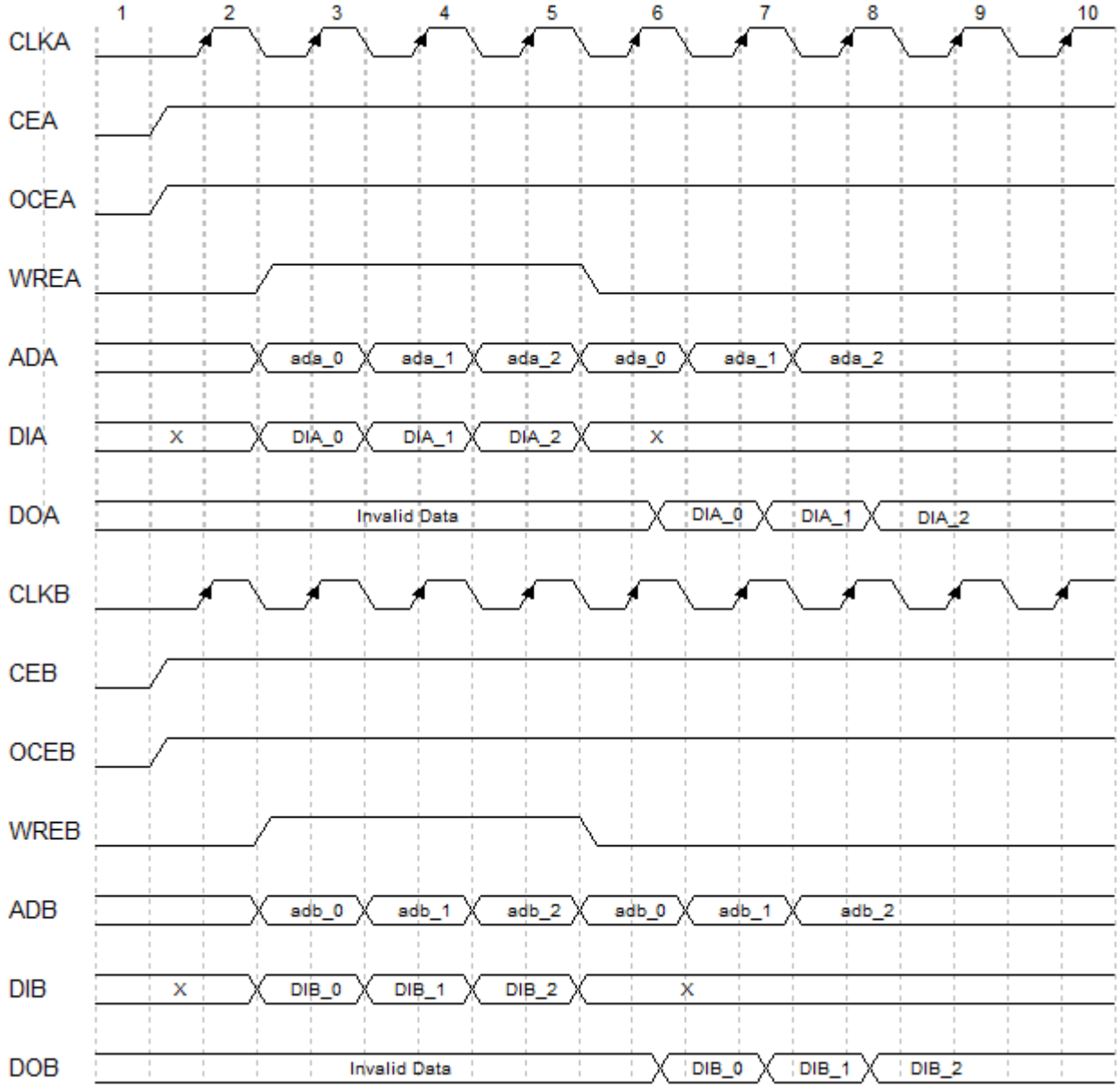


图 4-3 DP/DPX9 Normal 写模式时序波形图（Pipeline 读模式）

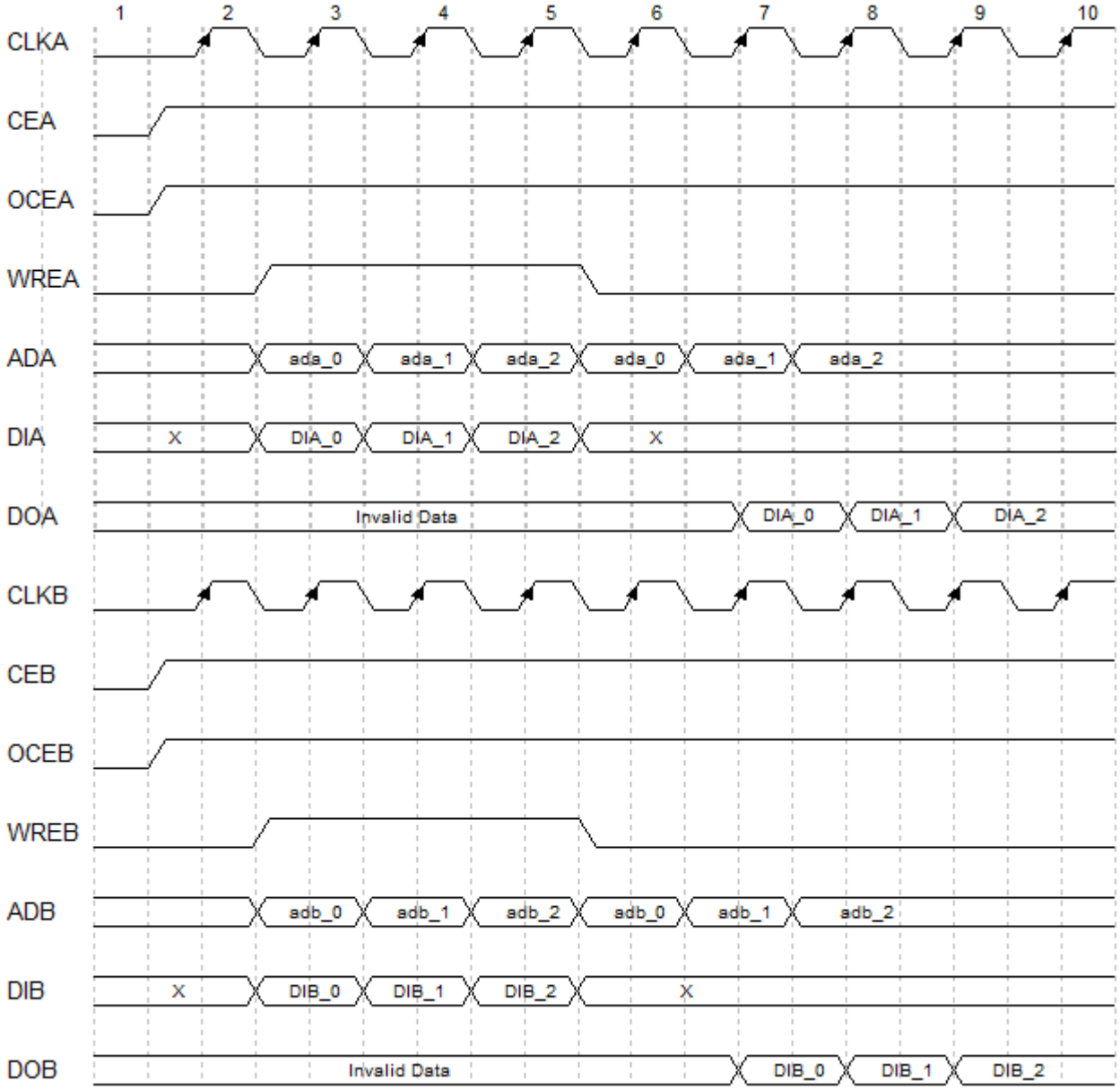


图 4-4 DP/DPX9 Write-through 写模式时序波形图（Bypass 读模式）

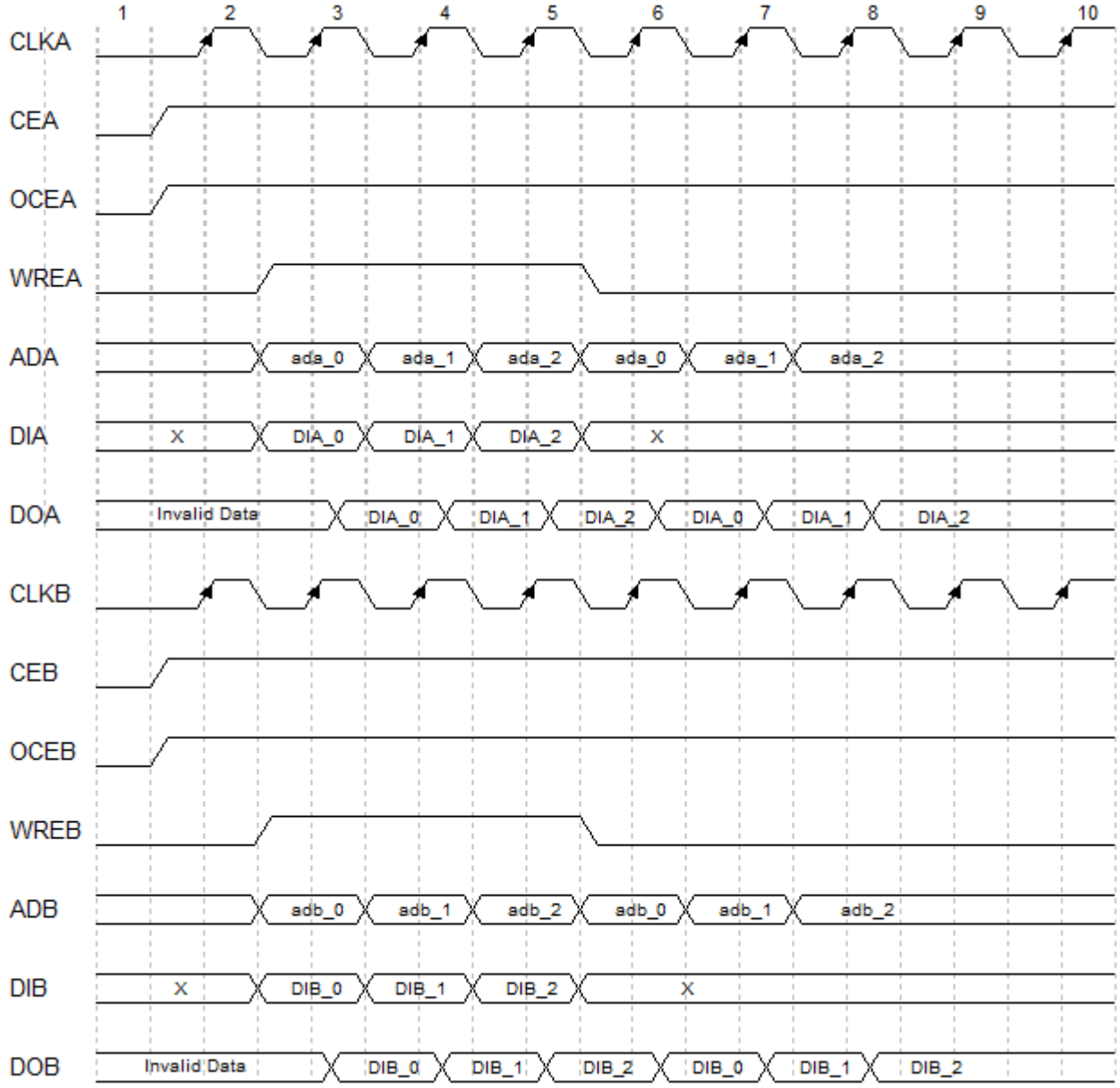


图 4-5 DP/DPX9 Write-through 写模式时序波形图（Pipeline 读模式）

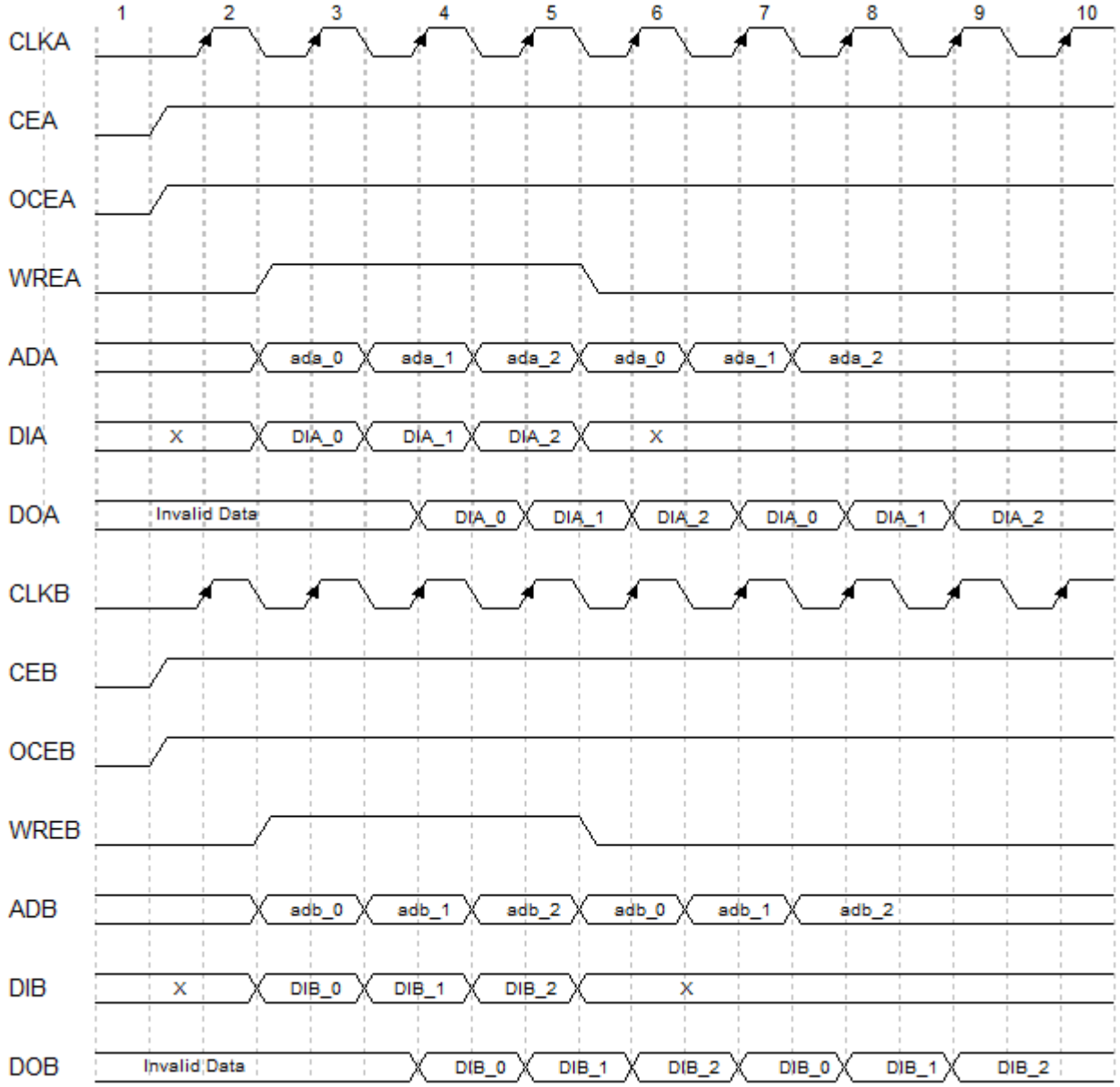


图 4-6 DP/DPX9 Read-before-write 写模式时序波形图（Bypass 读模式）

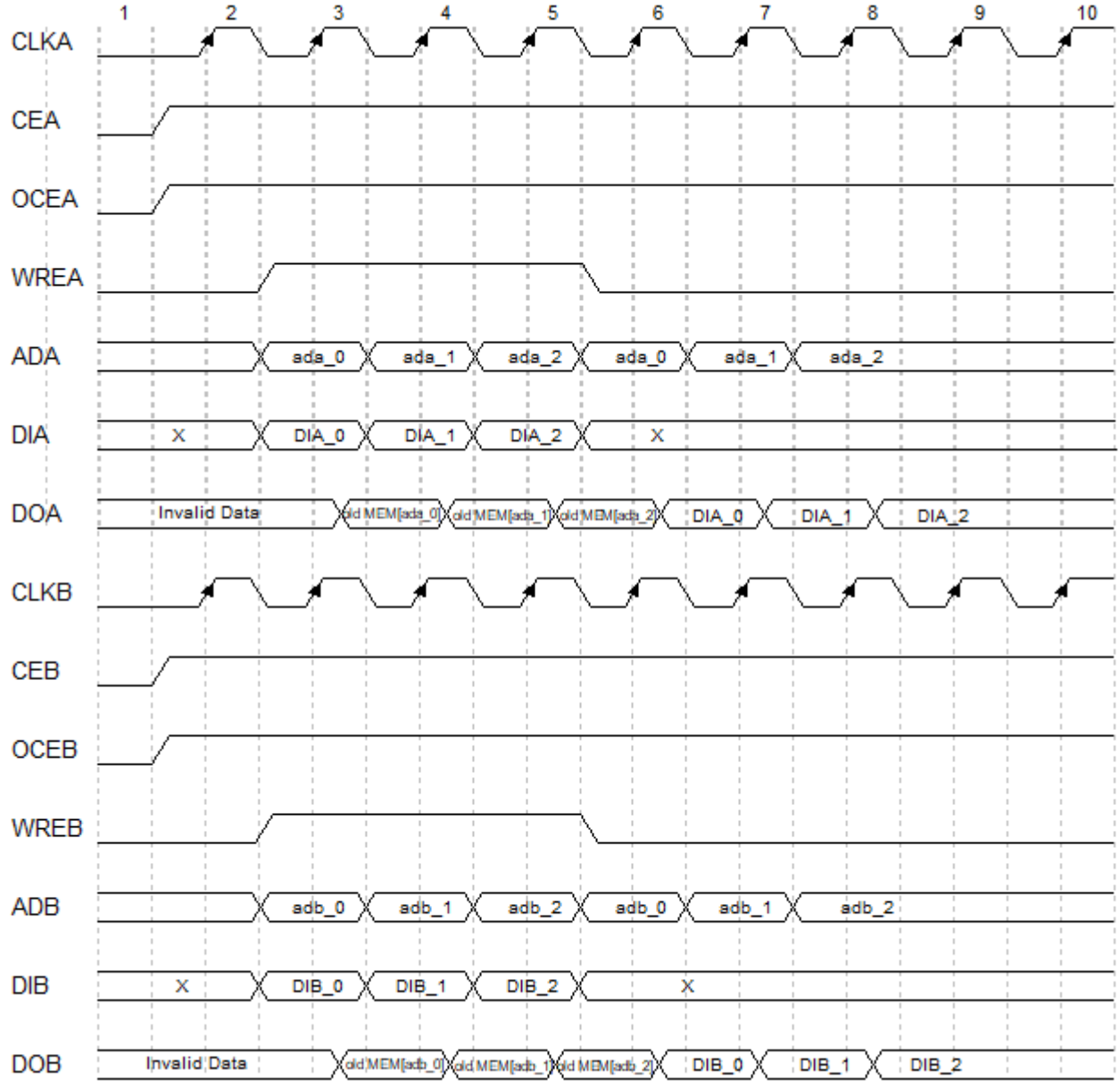
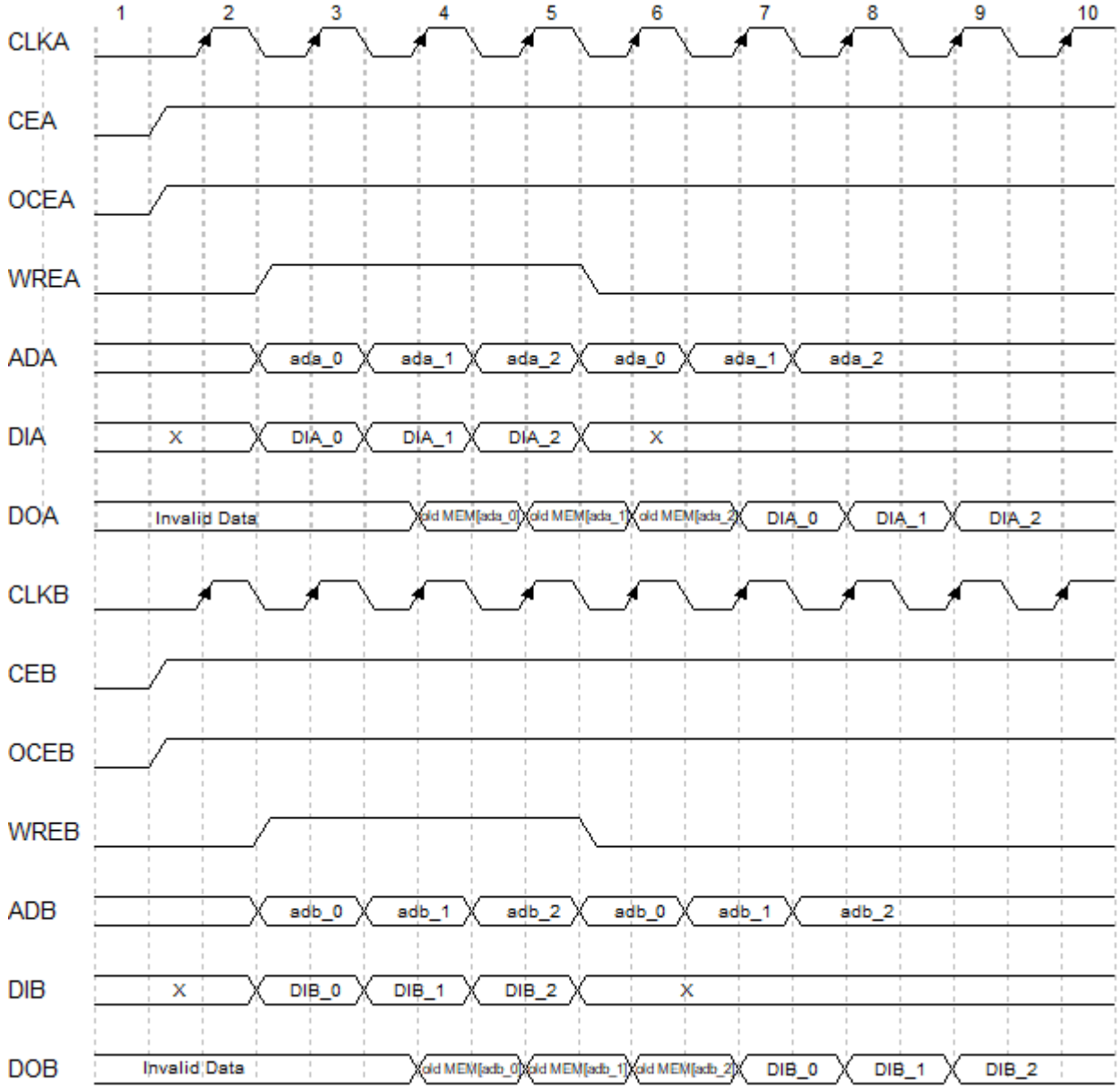


图 4-7 DP/DPX9 Read-before-write 写模式时序波形图（Pipeline 读模式）



端口介绍

表 4-1 端口介绍

端口名	I/O	描述
DOA[15:0]/DOA[17:0]	Output	A 端数据输出
DOB[15:0]/DOB[17:0]	Output	B 端数据输出
DIA[15:0]/DIA[17:0]	Input	A 端数据输入
DIB[15:0]/DIB[17:0]	Input	B 端数据输入
ADA[13:0]	Input	A 端地址输入
ADB[13:0]	Input	B 端地址输入
WREA	Input	A 端写使能输入 1：写入；

端口名	I/O	描述
		0：读出
WREB	Input	B 端写使能输入 1：写入； 0：读出
CEA	Input	A 端时钟使能信号，高电平有效
CEB	Input	B 端时钟使能信号，高电平有效
CLKA	Input	A 端时钟输入
CLKB	Input	B 端时钟输入
RESETA	Input	A 端复位输入，支持同步复位和异步复位，高电平有效
RESETB	Input	B 端复位输入，支持同步复位和异步复位，高电平有效
OCEA	Input	A 端输出时钟使能信号，用于 A 端 pipline 模式，对 bypass 模式无效
OCEB	Input	B 端输出时钟使能信号，用于 B 端 pipline 模式，对 bypass 模式无效
BLKSEL[2:0]	Input	BSRAM 块选择信号, 用于需要多个 BSRAM 存储单元级联实现容量扩展

参数介绍

表 4-2 参数介绍

参数名	参数类型	取值范围	默认值	描述
READ_MODE0	Integer	1’b0,1’b1	1’b0	A 端读模式配置 1’b0:bypass 模式 1’b1:pipeline 模式
READ_MODE1	Integer	1’b0,1’b1	1’b0	B 端读模式配置 1’b0:bypass 模式 1’b1:pipeline 模式
WRITE_MODE0	Integer	2’b00,2’b01,2’b10	2’b00	A 端写模式配置 2'b00: normal 模式 2'b01: write-through 模式 2'b10: read-before-write 模式
WRITE_MODE1	Integer	2’b00,2’b01,2’b10	2’b00	B 端写模式配置 2'b00: normal 模式 2'b01: write-through 模式 2'b10: read-before-write 模式
BIT_WIDTH_0	Integer	DP:1,2,4,8,16 DPX9:9,18	DP:16 DPX9:18	A 端数据宽度配置
BIT_WIDTH_1	Integer	DP:1,2,4,8,16 DPX9:9,18	DP:16 DP:18	B 端数据宽度配置
BLK_SEL	Integer	3’b000~3’b111	3’b000	BSRAM 块选择参数设置，与端口 BLKSEL 相等时该 BSRAM 被选中。使用 IP Core Generator 进行存储扩展时软件自动进行扩展处理。
RESET_MODE	String	SYNC,ASYNC	SYNC	复位模式配置

参数名	参数类型	取值范围	默认值	描述
				SYNC：同步复位 ASYNC：异步复位
INIT_RAM_00~ INIT_RAM_3F	Integer	DP:256'h0…0~256'h1…1 DPX9:288'h0…0~288'h1…1	DP:256'h0…0 DPX9:288'h0…0	用于设置 B-SRAM 存储单元的初始化数据

配置关系

表 4-3 数据宽度和地址深度配置关系

双端口模式	BSRAM 容量	数据宽度	地址深度
DP	16K	1	14
		2	13
		4	12
		8	11
		16	10
DPX9	18K	9	11
		18	10

原语例化

示例一

Verilog 例化：

```
DP bram_dp_0 (
        .DOA({doa[15:8],doa[7:0]}),
        .DOB({doa[15:8],dob[7:0]}),
        .CLKA(clka),
        .OCEA(ocea),
        .CEA(cea),
        .RESETA(reseta),
        .WREA(wrea),
        .CLKB(clkb),
        .OCEB(oceb),
        .CEB(ceb),
        .RESETB(resetb),
        .WREB(wreb),
        .BLKSEL({3'b000}),
        .ADA({ada[10:0],3'b000}),
        .DIA({{8{1'b0}},dia[7:0]})
        .ADB({adb[10:0],3'b000}),
        .DIB({{8{1'b0}},dib[7:0]})
    );
    defparam bram_dp_0.READ_MODE0 = 1'b0;
    defparam bram_dp_0.READ_MODE1 = 1'b0;
    defparam bram_dp_0.WRITE_MODE0 = 2'b00;
    defparam bram_dp_0.WRITE_MODE1 = 2'b00;
```

```
defparam bram_dp_0.BIT_WIDTH_0 = 8;
defparam bram_dp_0.BIT_WIDTH_1 = 8;
defparam bram_dp_0.BLK_SEL = 3'b000;
defparam bram_dp_0.RESET_MODE = "SYNC";
defparam bram_dp_0.INIT_RAM_00 =
256'h00A000000000000B00A000000000000B00A000000000000B00A00
0000000000B;
defparam bram_dp_0.INIT_RAM_3E =
256'h00A000000000000B00A000000000000B00A000000000000B00A00
0000000000B;
defparam bram_dp_0.INIT_RAM_3F =
256'h00A000000000000B00A000000000000B00A000000000000B00A00
0000000000B;
```

Vhdl 例化：

```
COMPONENT DP
        GENERIC (
                BIT_WIDTH_0:integer:=16;
                BIT_WIDTH_1:integer:=16;
                READ_MODE0:bit:='0';
                READ_MODE1:bit:='0';
                WRITE_MODE0:bit_vector:="00";
                WRITE_MODE1:bit_vector:="00";
                BLK_SEL:bit_vector:="000";
                RESET_MODE:string:="SYNC";
                INIT_RAM_00:bit_vector:=X"000000000000000
0000000000000000000000000000000000000000000000000";
                INIT_RAM_01:bit_vector:=X"000000000000000
0000000000000000000000000000000000000000000000000";
                INIT_RAM_3F:bit_vector:=X"000000000000000
0000000000000000000000000000000000000000000000000"
        );
        PORT (
                DOA,DOB:OUT std_logic_vector(15 downto 0):
=conv_std_logic_vector(0,16);
                CLKA,CLKB,CEA,CEB,OCEA,OCEB,RESETA,
RESETB,WREA,WREB:IN std_logic;
                ADA,ADB:IN std_logic_vector(13 downto 0);
                BLKSEL:IN std_logic_vector(2 downto 0);
                DIA,DIB:IN std_logic_vector(15 downto 0)
        );
END COMPONENT;
uut:DP
    GENERIC MAP(
                BIT_WIDTH_0=>16,
                BIT_WIDTH_1=>16,
                READ_MODE0=>'0',
                READ_MODE1=>'0',
                WRITE_MODE0=>"00",
                WRITE_MODE1=>"00",
                BLK_SEL=>"000",
```

```
                    RESET_MODE=>"SYNC",
    INIT_RAM_00=>X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_RAM_01=>X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_RAM_3F=>X"0000000000000000000000000000000000000000000000000000000000000000"
      )
      PORT  MAP(
          DOA=>doa,
          DOB=>dob,
          CLKA=>clka,
          CLKB=>clkb,
          CEA=>ceb,
          CEB=>ceb,
          OCEA=>ocea,
          OCEB=>oceb,
          RESETA=>reseta,
          RESETB=>resetb,
          WREA=>wrea,
          WREB=>wreb,
          ADA=>ada,
          ADB=>adb,
          BLKSEL=>blksel,
          DIA=>dia,
          DIB=>dib
      );
```

示例二

Verilog 例化：

```
DPX9 bram_dpx9_0 (
    .DOA(doa[17:0]),
    .DOB(dob[17:0]),
    .CLKA(clka),
    .OCEA(ocea),
    .CEA(cea),
    .RESETA(reseta),
    .WREA(wrea),
    .CLKB(clkb),
    .OCEB(oceb),
    .CEB(ceb),
    .RESETB(resetb),
    .WREB(wreb),
    .BLKSEL({3'b000}),
    .ADA({ada[9:0], 2'b00,byte_ena[1:0]}),
    .DIA(dia[17:0]),
    .ADB({adb[9:0], 2'b00,byte_enb[1:0]}),
    .DIB(dib[17:0])
);
defparam bram_dpx9_0.READ_MODE0 = 1'b1;
```

```
defparam bram_dpx9_0.READ_MODE1 = 1'b1;
defparam bram_dpx9_0.WRITE_MODE0 = 2'b01;
defparam bram_dpx9_0.WRITE_MODE1 = 2'b01;
defparam bram_dpx9_0.BIT_WIDTH_0 = 18;
defparam bram_dpx9_0.BIT_WIDTH_1 = 18;
defparam bram_dpx9_0.BLK_SEL = 3'b000;
defparam bram_dpx9_0.RESET_MODE = "SYNC";
defparam bram_dpx9_0.INIT_RAM_00 =
288'h000000000C000000000000D0000000000C000000000000D000
0000000C000000000000D0;
defparam bram_dpx9_0.INIT_RAM_01 =
288'h000000000C000000000000D0000000000C000000000000D000
0000000C000000000000D0;
defparam bram_dpx9_0.INIT_RAM_3F =
288'h000000000C000000000000D0000000000C000000000000D000
0000000C000000000000D0;
```

Vhdl 例化：

```
COMPONENT DPX9
        GENERIC (
                BIT_WIDTH_0:integer:=18;
                BIT_WIDTH_1:integer:=18;
                READ_MODE0:bit:='0';
                READ_MODE1:bit:='0';
                WRITE_MODE0:bit_vector:="00";
                WRITE_MODE1:bit_vector:="00";
                BLK_SEL:bit_vector:="000";
                RESET_MODE:string:="SYNC";
                INIT_RAM_00:bit_vector:=X"00000000000000
0000000000000000000000000000000000000000000000000000";
                INIT_RAM_01:bit_vector:=X"00000000000000
0000000000000000000000000000000000000000000000000000";
                INIT_RAM_3F:bit_vector:=X"00000000000000
0000000000000000000000000000000000000000000000000000"
        );
        PORT (
                DOA,DOB:OUT std_logic_vector(17 downto 0)
:=conv_std_logic_vector(0,18);
                CLKA,CLKB,CEA,CEB,OCEA,OCEB,RESETA,
RESETB,WREA,WREB:IN std_logic;
                ADA,ADB:IN std_logic_vector(13 downto 0);
                BLKSEL:IN std_logic_vector(2 downto 0);
                DIA:IN std_logic_vector(17 downto 0);
                DIB:IN std_logic_vector(17 downto 0)
        );
    END COMPONENT;
    uut:DPX9
        GENERIC MAP(
                BIT_WIDTH_0=>18,
                BIT_WIDTH_1=>18,
                READ_MODE0=>'0',
```

```
                                READ_MODE1=>'0',
                                WRITE_MODE0=>"00",
                                WRITE_MODE1=>"00",
                                BLK_SEL=>"000",
                                RESET_MODE=>"SYNC",
                                INIT_RAM_00=>X"0000000000000000000
0000000000000000000000000000000000000000000000000",
                                INIT_RAM_01=>X"0000000000000000000
0000000000000000000000000000000000000000000000000",
                                INIT_RAM_3F=>X"0000000000000000000
0000000000000000000000000000000000000000000000000"
                    )
                     PORT  MAP(
                          DOA=>doa,
                          DOB=>dob,
                          CLKA=>clka,
                          CLKB=>clkb,
                          CEA=>ceb,
                          CEB=>ceb,
                          OCEA=>ocea,
                          OCEB=>oceb,
                          RESETA=>reseta,
                          RESETB=>resetb,
                          WREA=>wrea,
                          WREB=>wreb,
                          ADA=>ada,
                          ADB=>adb,
                          BLKSEL=>blksel,
                          DIA=>dia,
                          DIB=>dib
               );
```

4.2 DPB/DPX9B

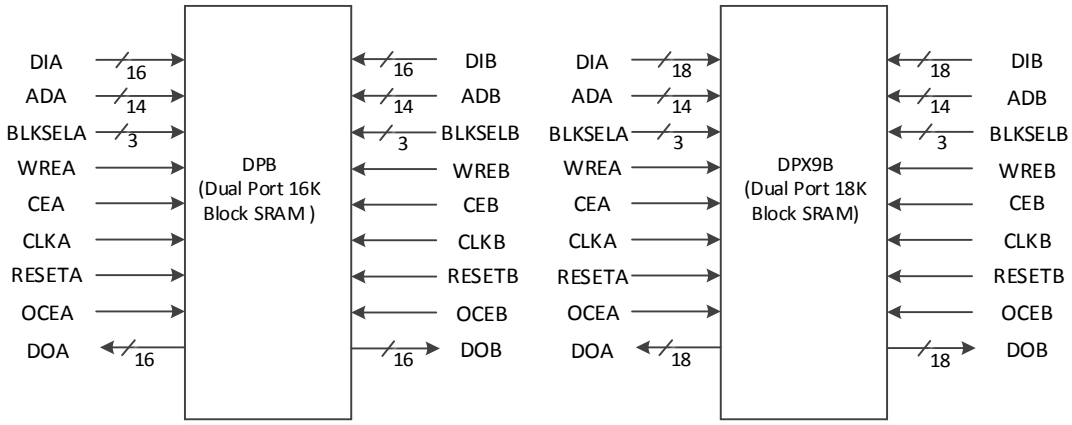
原语名称

DPB/DPX9B(True Dual Port 16K Block SRAM/True Dual Port 18K Block SRAM)，16K/18K 双端口 BSRAM。

DPB/DPX9B 是 DP/DPX9 的优化版本，A 端口和 B 端口分别独立支持块选择信号 BLKSELA 和 BLKSELB，推荐优先使用 DPB/DPX9B。

端口示意图

图 4-8 DPB/DPX9B 端口示意图



功能描述

DPB/DPX9B 的存储空间分别为 16K bit/18K bit，其工作模式为双端口模式，端口 A 和端口 B 均可分别独立实现读/写操作，可支持 2 种读模式（bypass 模式和 pipeline 模式）和 3 种写模式（normal 模式、write-through 模式和 read-before-write 模式）。

DPB 配置为 16bit、DPX9B 配置为 18bit 时，可实现 BSRAM 的 byte enable 功能，即通过写入地址端口 AD 的低四位控制写入存储器的数据，高电平使能。ADA[0]控制 DIA[7:0]/DIA[8:0]是否写入存储器，ADA[1]控制 DIA[15:8]/DIA[17:9]是否写入存储器，ADB[0]控制 DIB[7:0]/DIB[8:0]是否写入存储器，ADB[1]控制 DIB[15:8]/DIB[17:9]是否写入存储器。

读模式

通过参数 READ_MODE0、READ_MODE1 来启用或禁用 A 端、B 端输出 pipeline 寄存器，使用输出 pipeline 寄存器时，读操作需要额外的延迟周期。

写模式

包括 normal 模式、write-through 模式和 read-before-write 模式，A 端、B 端写模式通过参数 WRITE_MODE0、WRITE_MODE1 来分别配置使用，不同模式对应的内部时序波形图如图 4-9 到图 4-14 所示。

图 4-9 DPB/DPX9B Normal 写模式时序波形图（Bypass 读模式）

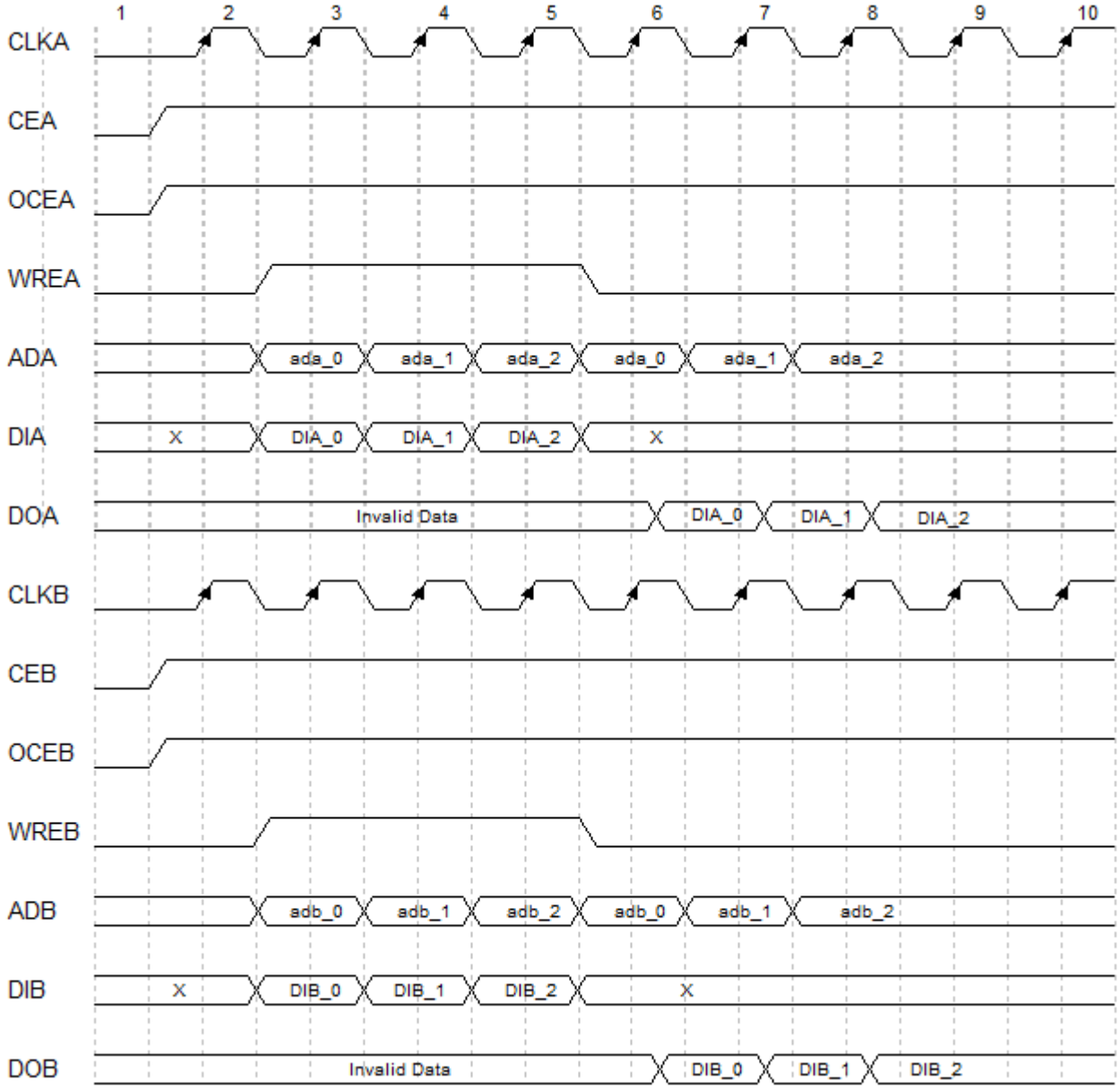


图 4-10 DPB/DPX9B Normal 写模式时序波形图（Pipeline 读模式）

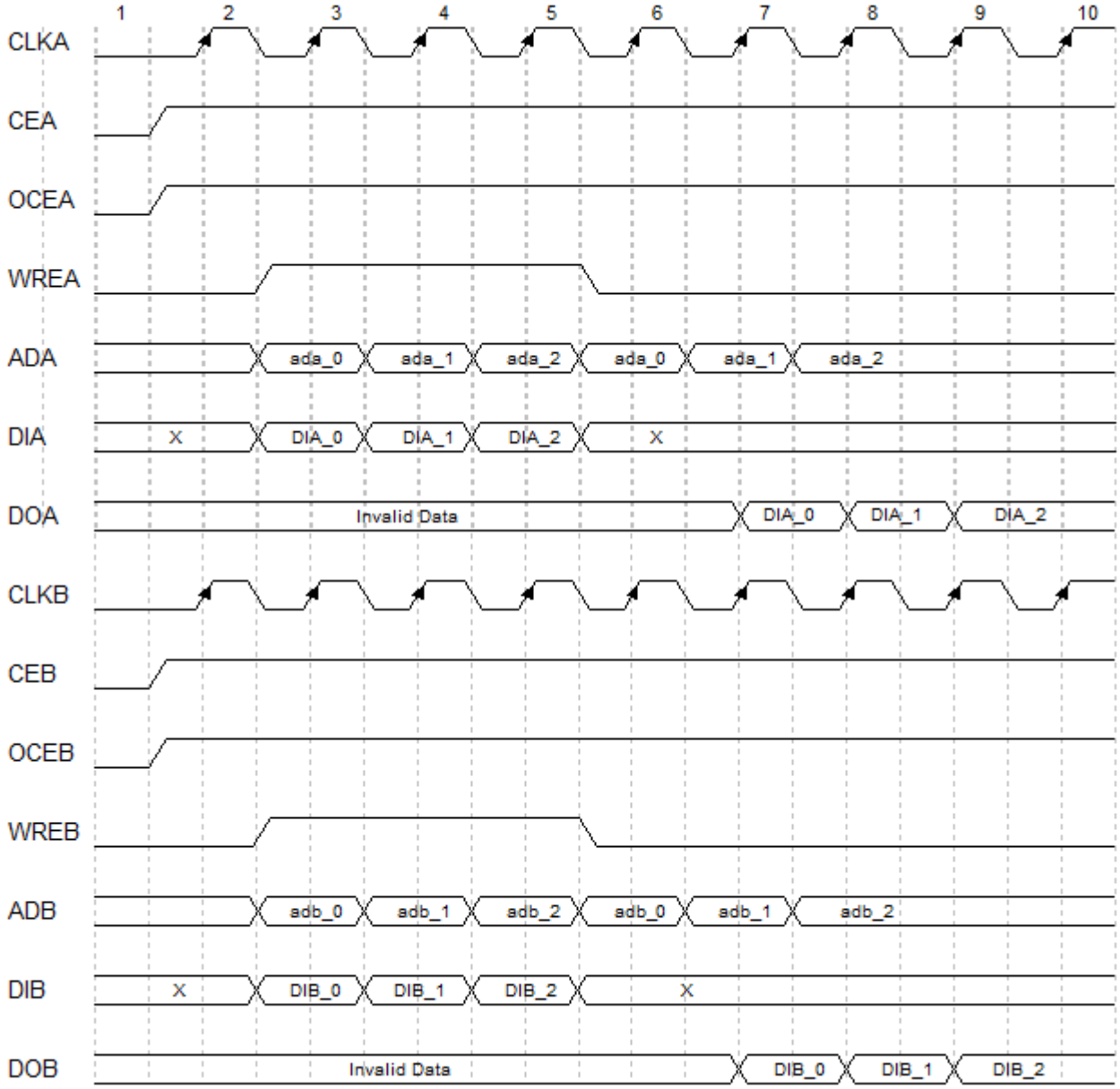


图 4-11 DPB/DPX9B Write-through 写模式时序波形图（Bypass 读模式）

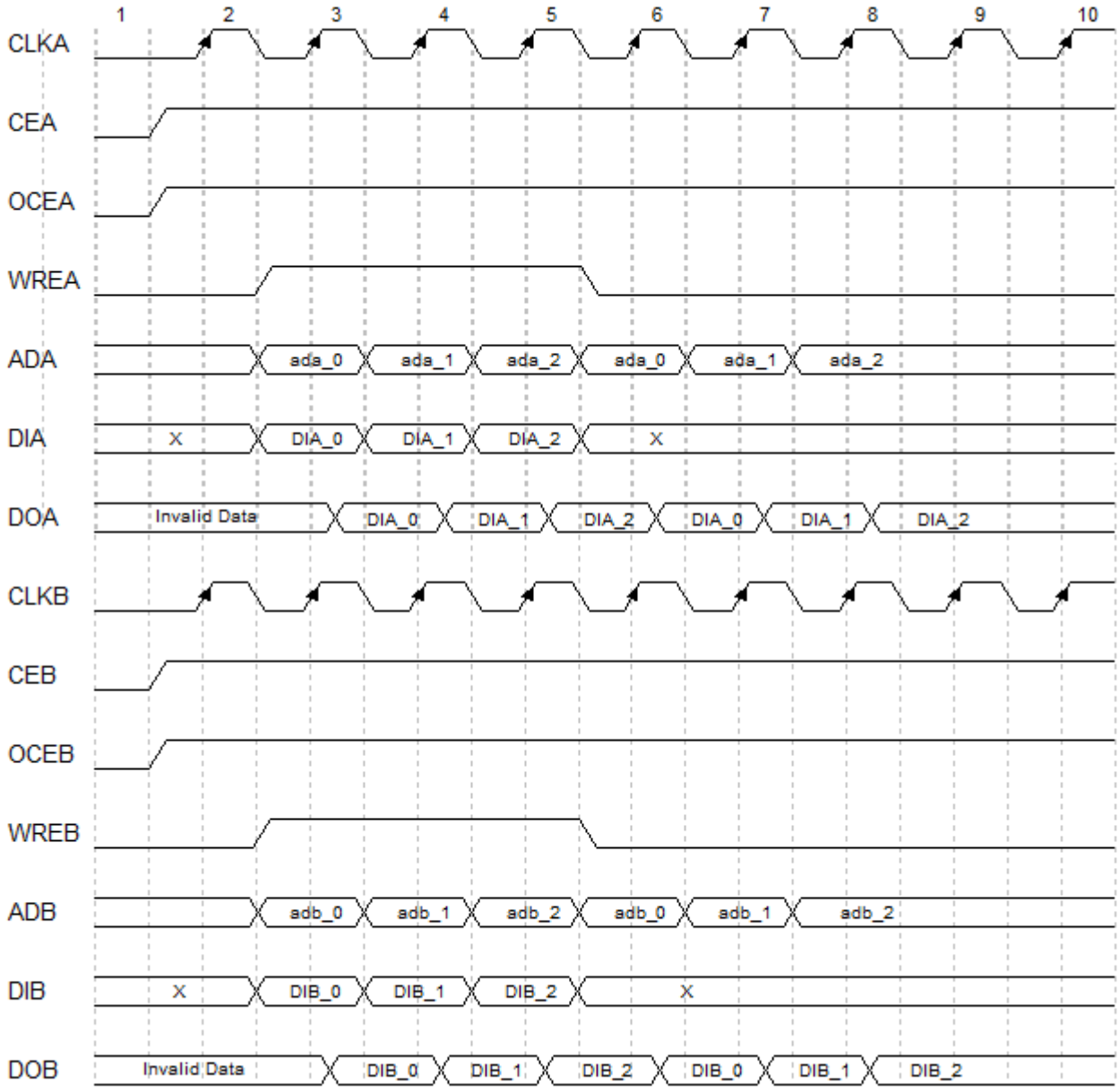


图 4-12 DPB/DPX9B Write-through 写模式时序波形图（Pipeline 读模式）

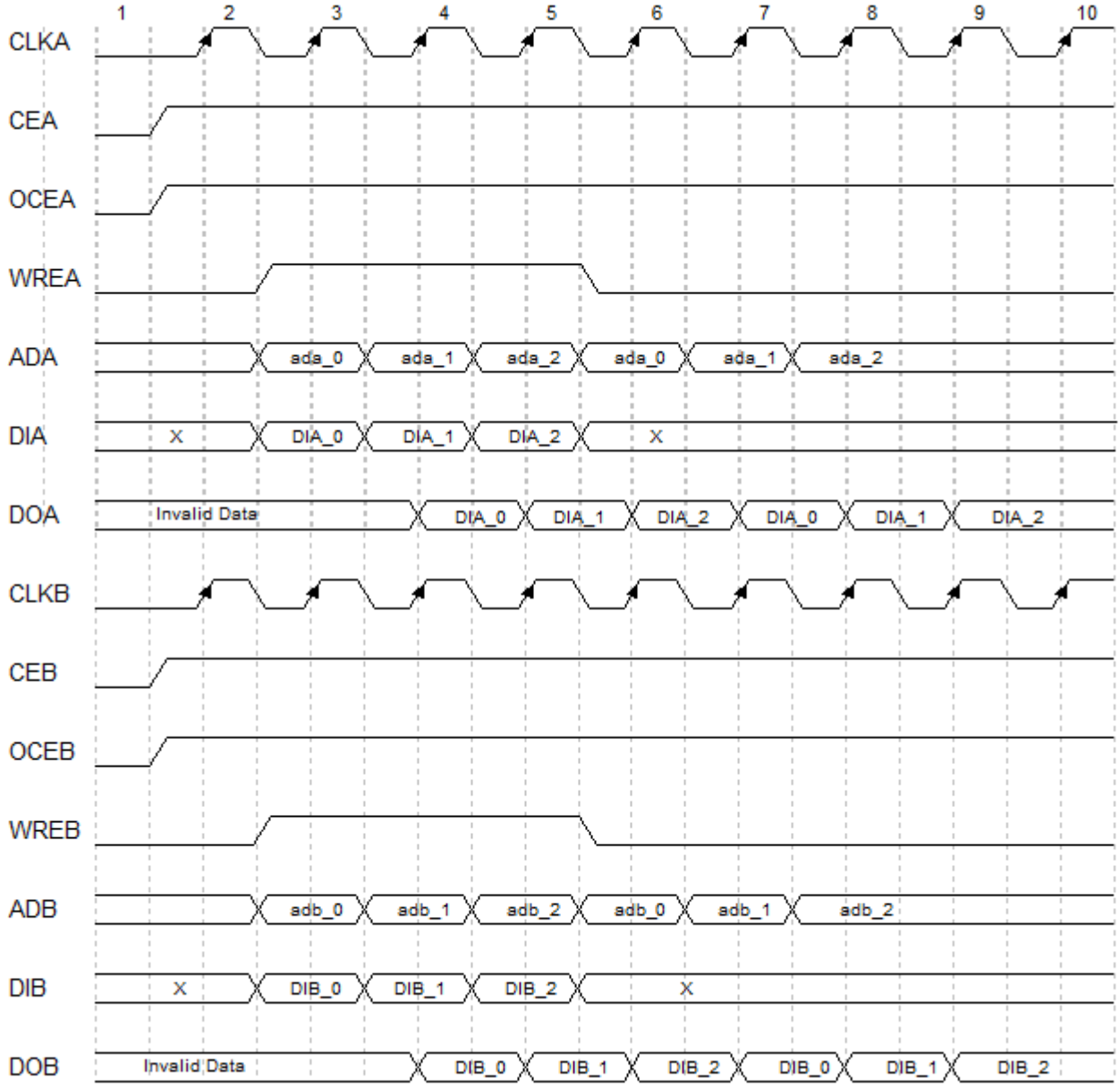


图 4-13 DPB/DPX9B Read-before-write 写模式时序波形图（Bypass 读模式）

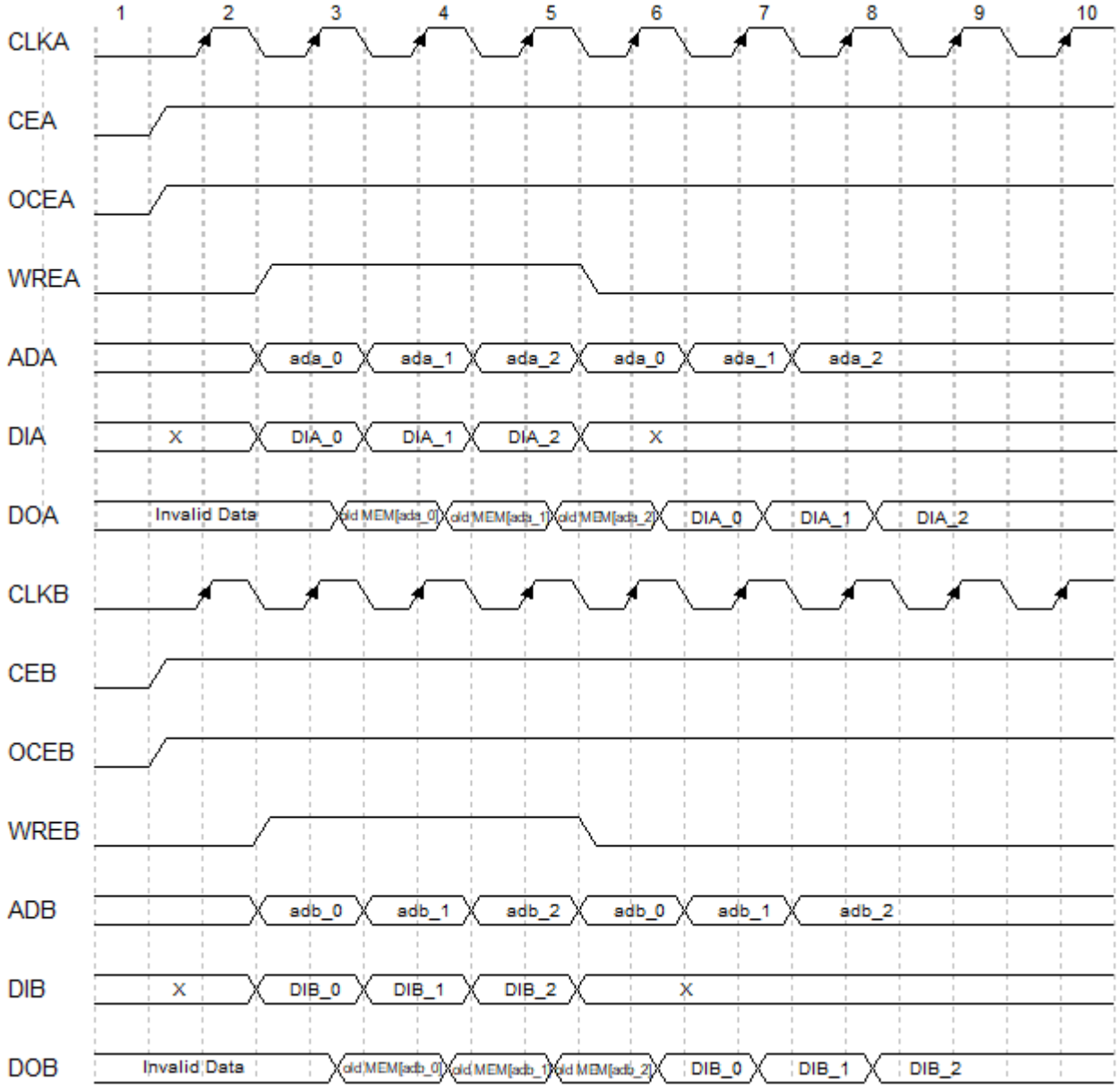
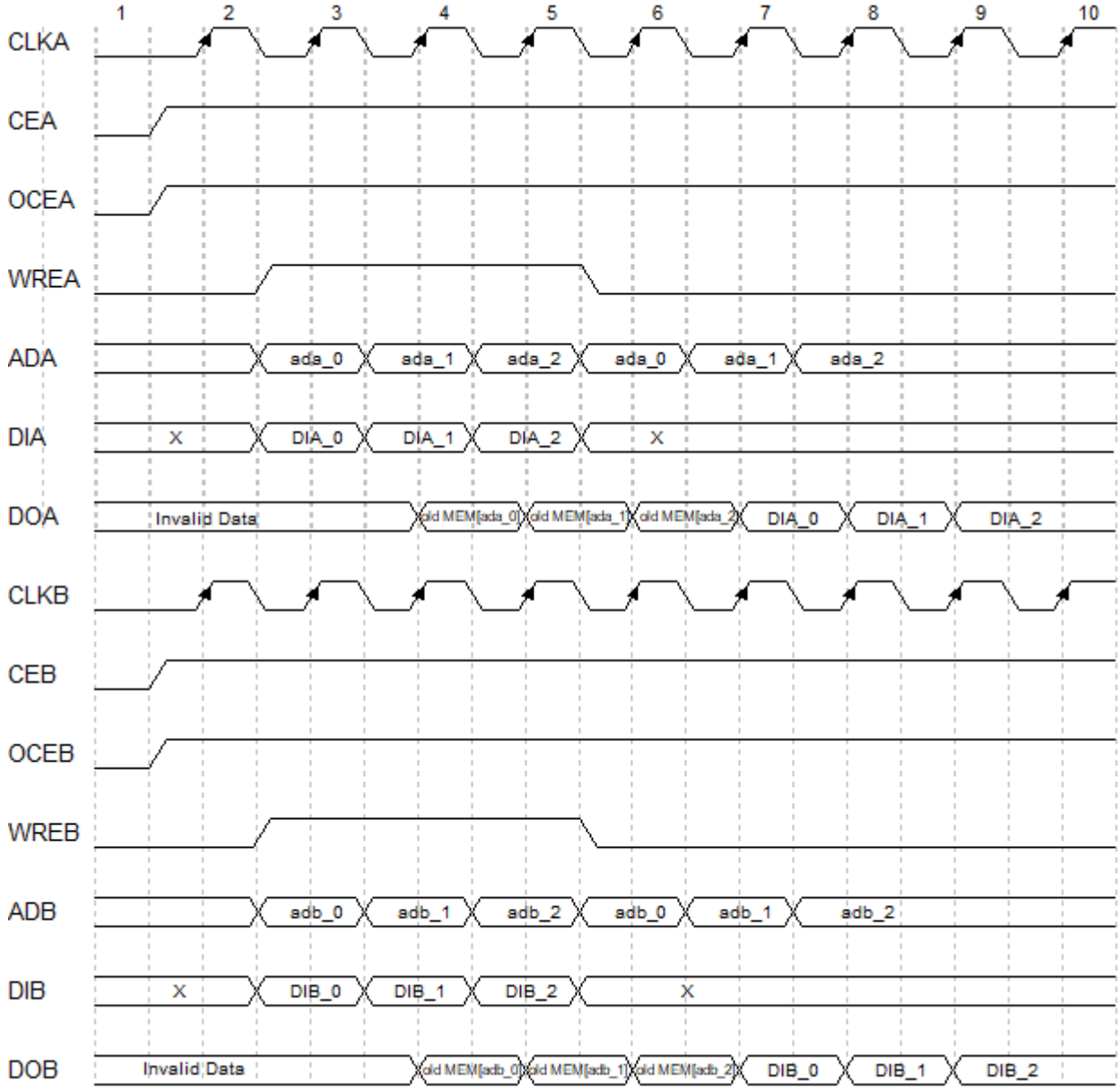


图 4-14 DPB/DPX9B Read-before-write 写模式时序波形图（Pipeline 读模式）



端口介绍

表 4-4 端口介绍

端口名	I/O	描述
DOA[15:0]/DOA[17:0]	Output	A 端数据输出
DOB[15:0]/DOB[17:0]	Output	B 端数据输出
DIA[15:0]/DIA[17:0]	Input	A 端数据输入
DIB[15:0]/DIB[17:0]	Input	B 端数据输入
ADA[13:0]	Input	A 端地址输入
ADB[13:0]	Input	B 端地址输入
WREA	Input	A 端写使能输入 1：写入；

端口名	I/O	描述
		0：读出
WREB	Input	B 端写使能输入 1：写入； 0：读出
CEA	Input	A 端时钟使能信号，高电平有效
CEB	Input	B 端时钟使能信号，高电平有效
CLKA	Input	A 端时钟输入
CLKB	Input	B 端时钟输入
RESETA	Input	A 端复位输入，支持同步复位和异步复位，高电平有效
RESETB	Input	B 端复位输入，支持同步复位和异步复位，高电平有效
OCEA	Input	A 端输出时钟使能信号，用于 A 端 pipline 模式，对 bypass 模式无效
OCEB	Input	B 端输出时钟使能信号，用于 B 端 pipline 模式，对 bypass 模式无效
BLKSELA[2:0]	Input	BSRAM A 端口块选择信号，用于需要多个 BSRAM 存储单元级联实现容量扩展
BLKSELB[2:0]	Input	BSRAM B 端口块选择信号，用于需要多个 BSRAM 存储单元级联实现容量扩展

参数介绍

表 4-5 参数介绍

参数名	参数类型	取值范围	默认值	描述
READ_MODE0	Integer	1’b0,1’b1	1’b0	A 端读模式配置 1’b0:bypass 模式 1’b1:pipeline 模式
READ_MODE1	Integer	1’b0,1’b1	1’b0	B 端读模式配置 1’b0:bypass 模式 1’b1:pipeline 模式
WRITE_MODE0	Integer	2’b00,2’b01,2’b10	2’b00	A 端写模式配置 2'b00: normal 模式 2'b01: write-through 模式 2'b10: read-before-write 模式
WRITE_MODE1	Integer	2’b00,2’b01,2’b10	2’b00	B 端写模式配置 2'b00: normal 模式 2'b01: write-through 模式 2'b10: read-before-write 模式
BIT_WIDTH_0	Integer	DPB:1,2,4,8,16 DPX9B:9,18	DPB:16 DPX9B:18	A 端数据宽度配置
BIT_WIDTH_1	Integer	DPB:1,2,4,8,16 DPX9B:9,18	DPB:16 DPB:18	B 端数据宽度配置
BLK_SEL_0	Integer	3’b000~3’b111	3’b000	BSRAM A 端口块选择参数设置，与端口 BLKSELA 相等时

参数名	参数类型	取值范围	默认值	描述
				该 BSRAM 被选中。使用 IP Core Generator 进行存储扩展时软件自动进行扩展处理。
BLK_SEL_1	Integer	3'b000~3'b111	3'b000	BSRAM B 端口块选择参数设置，与端口 BLKSELB 相等时该 BSRAM 被选中。使用 IP Core Generator 进行存储扩展时软件自动进行扩展处理。
RESET_MODE	String	SYNC,ASYNC	SYNC	复位模式配置 SYNC：同步复位 ASYNC：异步复位
INIT_RAM_00~ INIT_RAM_3F	Integer	DPB:256'h0…0~256'h1…1 DPX9B:288'h0…0~288'h1…1	DPB:256'h0…0 DPX9B:288'h0…0	用于设置 B-SRAM 存储单元的初始化数据

配置关系

表 4-6 数据宽度和地址深度配置关系

双端口模式	BSRAM 容量	数据宽度	地址深度
DPB	16K	1	14
		2	13
		4	12
		8	11
		16	10
DPX9B	18K	9	11
		18	10

原语例化

示例一

Verilog 例化：

```
DPB bram_dpb_0 (
    .DOA({doa[15:8],doa[7:0]}),
    .DOB({doa[15:8],dob[7:0]}),
    .CLKA(clka),
    .OCEA(ocea),
    .CEA(cea),
    .RESETA(reseta),
    .WREA(wrea),
    .CLKB(clkb),
    .OCEB(oceb),
    .CEB(ceb),
    .RESETB(resetb),
    .WREB(wreb),
    .BLKSELA({3'b000}),
```

```
        .BLKSELB({3'b000}),
        .ADA({ada[10:0],3'b000}),
        .DIA({{8{1'b0}},dia[7:0]})
        .ADB({adb[10:0],3'b000}),
        .DIB({{8{1'b0}},dib[7:0]})
    );
    defparam bram_dpb_0.READ_MODE0 = 1'b0;
    defparam bram_dpb_0.READ_MODE1 = 1'b0;
    defparam bram_dpb_0.WRITE_MODE0 = 2'b00;
    defparam bram_dpb_0.WRITE_MODE1 = 2'b00;
    defparam bram_dpb_0.BIT_WIDTH_0 = 8;
    defparam bram_dpb_0.BIT_WIDTH_1 = 8;
    defparam bram_dpb_0.BLK_SEL_0 = 3'b000;
    defparam bram_dpb_0.BLK_SEL_1 = 3'b000;
    defparam bram_dpb_0.RESET_MODE = "SYNC";
    defparam bram_dpb_0.INIT_RAM_00 =
256'h00A000000000000B00A000000000000B00A000000000000B00A00
0000000000B;
    defparam bram_dpb_0.INIT_RAM_3E =
256'h00A000000000000B00A000000000000B00A000000000000B00A00
0000000000B;
    defparam bram_dpb_0.INIT_RAM_3F =
256'h00A000000000000B00A000000000000B00A000000000000B00A00
0000000000B;
```

Vhdl 例化：

```
    COMPONENT DPB
        GENERIC (
            BIT_WIDTH_0:integer:=16;
            BIT_WIDTH_1:integer:=16;
            READ_MODE0:bit:='0';
            READ_MODE1:bit:='0';
            WRITE_MODE0:bit_vector:="00";
            WRITE_MODE1:bit_vector:="00";
            BLK_SEL_0:bit_vector:="000";
            BLK_SEL_1:bit_vector:="000";
            RESET_MODE:string:="SYNC";
            INIT_RAM_00:bit_vector:=X"000000000000000
0000000000000000000000000000000000000000000000000";
            INIT_RAM_01:bit_vector:=X"000000000000000
0000000000000000000000000000000000000000000000000";
            INIT_RAM_3F:bit_vector:=X"000000000000000
0000000000000000000000000000000000000000000000000"
        );
        PORT (
            DOA,DOB:OUT std_logic_vector(15 downto 0):
=conv_std_logic_vector(0,16);
            CLKA,CLKB,CEA,CEB,OCEA,OCEB,RESETA,
RESETB,WREA,WREB:IN std_logic;
            ADA,ADB:IN std_logic_vector(13 downto 0);
            BLKSELA:IN std_logic_vector(2 downto 0);
```

```
                    BLKSELB:IN std_logic_vector(2 downto 0);
                    DIA,DIB:IN std_logic_vector(15 downto 0)
            );
    END COMPONENT;
    uut:DPB
        GENERIC MAP(
                    BIT_WIDTH_0=>16,
                    BIT_WIDTH_1=>16,
                    READ_MODE0=>'0',
                    READ_MODE1=>'0',
                    WRITE_MODE0=>"00",
                    WRITE_MODE1=>"00",
                    BLK_SEL_0=>"000",
                    BLK_SEL_1=>"000",
                    RESET_MODE=>"SYNC",
    INIT_RAM_00=>X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_RAM_01=>X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_RAM_3F=>X"0000000000000000000000000000000000000000000000000000000000000000"
        )
        PORT MAP(
            DOA=>doa,
            DOB=>dob,
            CLKA=>clka,
            CLKB=>clkb,
            CEA=>ceb,
            CEB=>ceb,
            OCEA=>ocea,
            OCEB=>oceb,
            RESETA=>reseta,
            RESETB=>resetb,
            WREA=>wrea,
            WREB=>wreb,
            ADA=>ada,
            ADB=>adb,
            BLKSELA=>blksela,
            BLKSELB=>blkselb,
            DIA=>dia,
            DIB=>dib
        );
```

示例二

Verilog 例化：

```
DPX9B bram_dpx9b_0 (
    .DOA(doa[17:0]),
    .DOB(dob[17:0]),
    .CLKA(clka),
    .OCEA(ocea),
```

```
        .CEA(cea),
        .RESETA(reseta),
        .WREA(wrea),
        .CLKB(clkb),
        .OCEB(oceb),
        .CEB(ceb),
        .RESETB(resetb),
        .WREB(wreb),
        .BLKSELA({3'b000}),
        .BLKSELB({3'b000}),
        .ADA({ada[9:0], 2'b00,byte_ena[1:0]}),
        .DIA(dia[17:0]),
        .ADB({adb[9:0], 2'b00,byte_enb[1:0]}),
        .DIB(dib[17:0])
    );
    defparam bram_dpx9b_0.READ_MODE0 = 1'b1;
    defparam bram_dpx9b_0.READ_MODE1 = 1'b1;
    defparam bram_dpx9b_0.WRITE_MODE0 = 2'b01;
    defparam bram_dpx9b_0.WRITE_MODE1 = 2'b01;
    defparam bram_dpx9b_0.BIT_WIDTH_0 = 18;
    defparam bram_dpx9b_0.BIT_WIDTH_1 = 18;
    defparam bram_dpx9b_0.BLK_SEL_0 = 3'b000;
    defparam bram_dpx9b_0.BLK_SEL_1 = 3'b000;
    defparam bram_dpx9b_0.RESET_MODE = "SYNC";
    defparam bram_dpx9b_0.INIT_RAM_00 =
    288'h000000000C000000000000D0000000000C000000000000D000
    0000000C000000000000D0;
    defparam bram_dpx9b_0.INIT_RAM_01 =
    288'h000000000C000000000000D0000000000C000000000000D000
    0000000C000000000000D0;
    defparam bram_dpx9b_0.INIT_RAM_3F =
    288'h000000000C000000000000D0000000000C000000000000D000
    0000000C000000000000D0;
```

Vhdl 例化：

```
    COMPONENT  DPX9B
            GENERIC  (
                    BIT_WIDTH_0:integer:=18;
                    BIT_WIDTH_1:integer:=18;
                    READ_MODE0:bit:='0';
                    READ_MODE1:bit:='0';
                    WRITE_MODE0:bit_vector:="00";
                    WRITE_MODE1:bit_vector:="00";
                    BLK_SEL_0:bit_vector:="000";
                    BLK_SEL_1:bit_vector:="000";
                    RESET_MODE:string:="SYNC";
                    INIT_RAM_00:bit_vector:=X"00000000000000
000000000000000000000000000000000000000000000000";
                    INIT_RAM_01:bit_vector:=X"00000000000000
000000000000000000000000000000000000000000000000";
                    INIT_RAM_3F:bit_vector:=X"00000000000000
```

```
00000000000000000000000000000000000000000000000000"
                    );
                    PORT (
                                DOA,DOB:OUT std_logic_vector(17 downto 0)
:=conv_std_logic_vector(0,18);
                                CLKA,CLKB,CEA,CEB,OCEA,OCEB,RESETA,
RESETB,WREA,WREB:IN std_logic;
                                ADA,ADB:IN std_logic_vector(13 downto 0);
                                BLKSELA:IN std_logic_vector(2 downto 0);
                                BLKSELB:IN std_logic_vector(2 downto 0);
                                DIA:IN std_logic_vector(17 downto 0);
                                DIB:IN std_logic_vector(17 downto 0)
                     );
               END COMPONENT;
               uut:DPX9B
                   GENERIC MAP(
                                BIT_WIDTH_0=>18,
                                BIT_WIDTH_1=>18,
                                READ_MODE0=>'0',
                                READ_MODE1=>'0',
                                WRITE_MODE0=>"00",
                                WRITE_MODE1=>"00",
                                BLK_SEL_0=>"000",
                                BLK_SEL_1=>"000",
                                RESET_MODE=>"SYNC",
                                INIT_RAM_00=>X"0000000000000000000
00000000000000000000000000000000000000000000000",
                                INIT_RAM_01=>X"0000000000000000000
00000000000000000000000000000000000000000000000",
                                INIT_RAM_3F=>X"0000000000000000000
00000000000000000000000000000000000000000000000"
               )
                PORT MAP(
                    DOA=>doa,
                    DOB=>dob,
                    CLKA=>clka,
                    CLKB=>clkb,
                    CEA=>ceb,
                    CEB=>ceb,
                    OCEA=>ocea,
                    OCEB=>oceb,
                    RESETA=>reseta,
                    RESETB=>resetb,
                    WREA=>wrea,
                    WREB=>wreb,
                    ADA=>ada,
                    ADB=>adb,
                    BLKSELA=>blksela,
                    BLKSELB=>blkselb,
                    DIA=>dia,
```

```
                DIB=>dib
    );
```

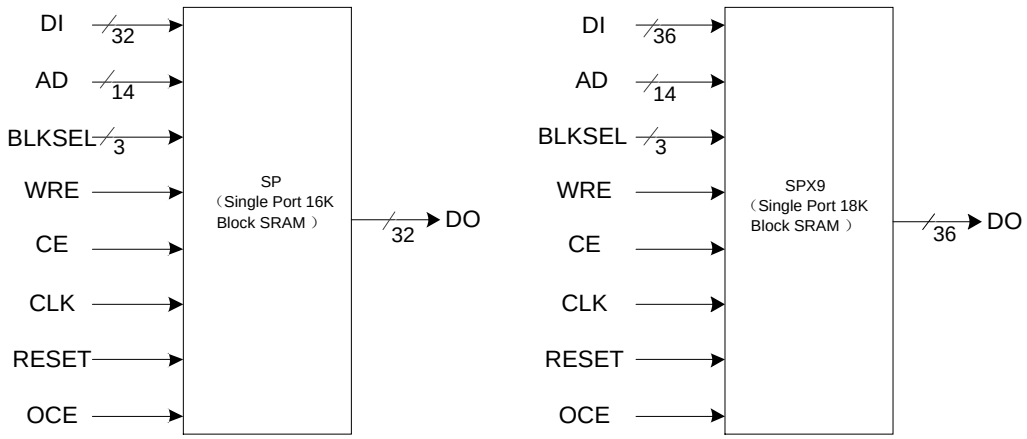
4.3 SP/SPX9

原语名称

SP/SPX9(Single Port 16K Block SRAM/Single Port 18K Block SRAM),16K/18K 单端口 BSRAM。

端口示意图

图 4-15 SP/SPX9 端口示意图



功能描述

SP/SPX9 存储空间为 16K bit/18K bit，其工作模式为单端口模式，由一个时钟控制单端口的读/写操作，可支持 2 种读模式（bypass 模式和 pipeline 模式）和 3 种写模式（normal 模式、write-through 模式和 read-before-write 模式）。

SP 配置为 16bit/32bit、SPX9 配置为 18bit/36bit 时，可实现 BSRAM 的 byte enable 功能，即通过写入地址端口 AD 的低四位控制写入存储器的数据，高电平使能。AD[0]控制 DI[7:0]/DI[8:0]是否写入存储器，AD[1]控制 DI[15:8]/DI[17:9]是否写入存储器，AD[2]控制 DI[23:16]/DI[26:18]是否写入存储器，AD[3]控制 DI[31:24]/DI[35:27]是否写入存储器。

读模式

通过参数 READ_MODE 来启用或禁用输出 pipeline 寄存器，使用输出 pipeline 寄存器时，读操作需要额外的延迟周期。

写模式

包括 normal 模式、write-through 模式和 read-before-write 模式，通过参数 WRITE_MODE 来配置使用。

SP/SPX9 不同模式对应的内部时序波形图可参考 DP/DPX9　A 端/B 端时序图 4-2 到图 4-7。

端口介绍

表 4-7 端口介绍

端口名	I/O	描述
DO[31:0]/DO[35:0]	Output	数据输出
DI[31:0]/DI[35:0]	Input	数据输入
AD[13:0]	Input	地址输入
WRE	Input	写使能输入 1：写入； 0：读出
CE	Input	时钟使能输入，高电平有效
CLK	Input	时钟输入
RESET	Input	复位输入，支持同步复位和异步复位，高电平有效
OCE	Input	输出时钟使能信号，用于 pipline 模式，对 bypass 模式无效
BLKSEL[2:0]	Input	BSRAM 块选择信号，用于需要多个 BSRAM 存储单元级联实现容量扩展

参数介绍

表 4-8 参数介绍

参数名	参数类型	取值范围	默认值	描述
READ_MODE	Integer	1’b0,1’b1	1’b0	读模式配置 1’b0:bypass 模式 1’b1:pipeline 模式
WRITE_MODE	Integer	2’b00,2’b01,2’b10	2’b00	写模式配置 2'b00: normal 模式 2'b01:write-through 模式; 2'b10: read-before-write 模式
BIT_WIDTH	Integer	SP:1,2,4,8,16,32 SPX9:9,18,36	SP:32 SPX9:36	数据宽度配置
BLK_SEL	Integer	3’b000~3’b111	3’b000	BSRAM 块选择参数设置，与端口 BLKSEL 相等时该 BSRAM 被选中。使用 IP Core Generator 进行存储扩展时软件自动进行扩展处理。
RESET_MODE	String	SYNC,ASYNC	SYNC	复位模式配置 SYNC：同步复位 ASYNC：异步复位
INIT_RAM_00~ INIT_RAM_3F	Integer	SP:256’h0…0~256’h1…1 SPX9:288’h0…0~288’h1…1	SP:256’h0…0 SPX9:288’h0…0	用于设置 B-SRAM 存储单元的初始化数据

配置关系

表 4-9 数据宽度和地址深度配置关系

单端口模式	BSRAM 容量	数据宽度	地址深度
SP	16K	1	14
		2	13
		4	12
		8	11
		16	10
		32	9
SPX9	18K	9	11
		18	10
		36	9

原语例化

示例一

Verilog 例化：

```
SP bram_sp_0 (
    .DO({dout[31:8], dout[7:0]}),
    .CLK(clk),
    .OCE(oce),
    .CE(ce),
    .RESET(reset),
    .WRE(wre),
    .BLKSEL({3’b000}),
    .AD({ad[10:0], 3’b000}),
    .DI({{24{1’b0}}, din[7:0]})
);
defparam bram_sp_0.READ_MODE = 1'b0;
defparam bram_sp_0.WRITE_MODE = 2'b00;
defparam bram_sp_0.BIT_WIDTH = 8;
defparam bram_sp_0.BLK_SEL = 3'b000;
defparam bram_sp_0.RESET_MODE = "SYNC";
defparam bram_sp_0.INIT_RAM_00 =
256'h00A000000000000B00A000000000000B00A000000000000B00
A000000000000B;
defparam bram_sp_0.INIT_RAM_01 =
256'h00A000000000000B00A000000000000B00A000000000000B00
A000000000000B;
defparam bram_sp_0.INIT_RAM_3F =
256'h00A000000000000B00A000000000000B00A000000000000B00
A000000000000B;
```

Vhdl 例化：

```
COMPONENT SP
        GENERIC(
                BIT_WIDTH:integer:=32;
```

```
                        READ_MODE:bit:='0';
                        WRITE_MODE:bit_vector:="01";
                        BLK_SEL:bit_vector:="000";
                        RESET_MODE:string:="SYNC";
                        INIT_RAM_00:bit_vector:=X"00A000000000000B
00A000000000000B00A000000000000B00A000000000000B ";
                        INIT_RAM_01:bit_vector:=X"00A000000000000B
00A000000000000B00A000000000000B00A000000000000B ";
                        INIT_RAM_3F:bit_vector:=X"00A000000000000B
00A000000000000B00A000000000000B00A000000000000B "
                    );
                    PORT(
                        DO:OUT std_logic_vector(31 downto 0):=conv_
std_logic_vector(0,32);
                        CLK,CE,OCE,RESET,WRE:IN std_logic;
                        AD:IN std_logic_vector(13 downto 0);
                        BLKSEL:IN std_logic_vector(2 downto 0);
                        DI:IN std_logic_vector(31 downto 0)
                    );
        END COMPONENT;
        uut:SP
            GENERIC MAP(
                        BIT_WIDTH=>32,
                        READ_MODE=>'0',
                        WRITE_MODE=>"01",
                        BLK_SEL=>"000",
                        RESET_MODE=>"SYNC",
                        INIT_RAM_00=>X"00A000000000000B00A00
0000000000B00A000000000000B00A000000000000B ",
                        INIT_RAM_01=>X"00A000000000000B00A00
0000000000B00A000000000000B00A000000000000B ",
                        INIT_RAM_02=>X"00A000000000000B00A00
0000000000B00A000000000000B00A000000000000B ",
                        INIT_RAM_3F=>X"00A000000000000B00A00
0000000000B00A000000000000B00A000000000000B "
                )
                PORT MAP (
                    DO=>dout,
                    CLK=>clk,
                    OCE=>oce,
                    CE=>ce,
                    RESET=>reset,
                    WRE=>wre,
                    BLKSEL=>blksel,
                    AD=>ad,
                    DI=>din
                );
```

示例二

Verilog 例化：

```
SPX9 bram_spx9_0 (
```

```
        .DO({dout[35:18],dout[17:0]}),
        .CLK(clk),
        .OCE(oce),
        .CE(ce),
        .RESET(reset),
        .WRE(wre),
        .BLKSEL({3'b000}),
        .AD({ad[9:0], 2'b00, byte_en[1:0]}),
        .DI({{18{1'b0}},din[17:0]})
    );
    defparam bram_spx9_0.READ_MODE = 1'b0;
    defparam bram_spx9_0.WRITE_MODE = 2'b00;
    defparam bram_spx9_0.BIT_WIDTH = 18;
    defparam bram_spx9_0.BLK_SEL = 3'b000;
    defparam bram_spx9_0.RESET_MODE = "SYNC";
    defparam bram_spx9_0.INIT_RAM_00 =
    288'h000000000C000000000000D0000050000C000000000000D000
    0000000C000000000000D0;
    defparam bram_spx9_0.INIT_RAM_01 =
    288'h000000000C000000000000D0000000000C000000003000D000
    0000000C000000000040D0;
    defparam bram_spx9_0.INIT_RAM_3F =
    288'h0000A0000C000000000000D0000000000C000000000000D001
    0000000C000000000000D0;
```

Vhdl 例化：

```
    COMPONENT SPX9
        GENERIC(
            BIT_WIDTH:integer:=9;
            READ_MODE:bit:='0';
            WRITE_MODE:bit_vector:="00";
            BLK_SEL : bit_vector:="000";
            RESET_MODE : string:="SYNC";
            INIT_RAM_00:bit_vector:=X"000000000C000000
000000D0000050000C000000000000D0000000000C000000000000D0";
            INIT_RAM_01:bit_vector:=X"000000000C000000
000000D0000000000C000000003000D0000000000C000000000040D0";
            INIT_RAM_3F:bit_vector:=X"0000A0000C000000
000000D0000000000C000000000000D0010000000C000000000000D0"
        );
        PORT(
            DO:OUT std_logic_vector(35 downto 0):=conv_
std_logic_vector(0,36);
            CLK,CE,OCE,RESET,WRE:IN std_logic;
            AD:IN std_logic_vector(13 downto 0);
            DI:IN std_logic_vector(35 downto 0);
            BLKSEL:std_logic_vector(2 downto 0)
        );
    END COMPONENT;
    uut:SPX9
        GENERIC MAP(
```

```
                BIT_WIDTH=>9,
                READ_MODE=>'0',
                WRITE_MODE=>"00",
                BLK_SEL=>"000",
                RESET_MODE=>"SYNC",
                INIT_RAM_00=>X"0000000000000000000
0000000000000000000000000000000000000000000000000000",
                INIT_RAM_01=>X"0000000000000000000
0000000000000000000000000000000000000000000000000000",
                INIT_RAM_3F=>X"0000000000000000000
0000000000000000000000000000000000000000000000000000"
        )
        PORT MAP(
            DO=>dout,
            CLK=>clk,
            OCE=>oce,
            CE=>ce,
            RESET=>reset,
            WRE=>wre,
            BLKSEL=>blksel,
            AD=>ad,
            DI=>din
        );
```

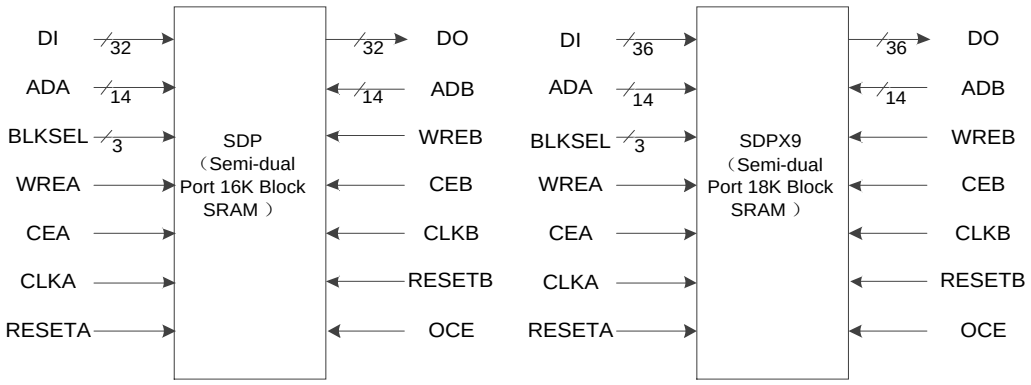
4.4 SDP/SDPX9

原语名称

SDP/SDPX9(Semi Dual Port 16K Block SRAM /Semi Dual Port 18K Block SRAM),16K/18K 半双端口 BSRAM。

端口示意图

图 4-16 SDP/SDPX9 端口示意图



功能描述

SDP/SDPX9 存储空间分别为 16K bit/18K bit，其工作模式为半双端口模式，端口 A 进行写操作，端口 B 进行读操作，可支持 2 种读模式（bypass 模式和 pipeline 模式）和 1 种写模式（normal 模式）。

SDP 配置为 16bit/32bit、SDPX9 配置为 18bit/36bit 时，可实现 BSRAM

的 byte enable 功能，即通过写入地址端口 AD 的低四位控制写入存储器的数据，高电平使能。ADA[0]控制 DI[7:0]/DI[8:0]是否写入存储器，ADA[1]控制 DI[15:8]/DI[17:9]是否写入存储器，ADA[2]控制 DI[23:16]/DI[26:18]是否写入存储器，ADA[3]控制 DI[31:24]/DI[35:27]是否写入存储器。

读模式

通过参数 READ_MODE 来启用或禁用输出 pipeline 寄存器，使用输出 pipeline 寄存器时，读操作需要额外的延迟周期。

写模式

SDP/SDPX9 端口 A 进行写操作，端口 B 进行读操作，支持 normal 模式。

不同模式对应的内部时序波形图如图 4-17 和图 4-18 所示。

图 4-17 SDP/SDPX9 Normal 写模式时序波形图（Bypass 读模式）

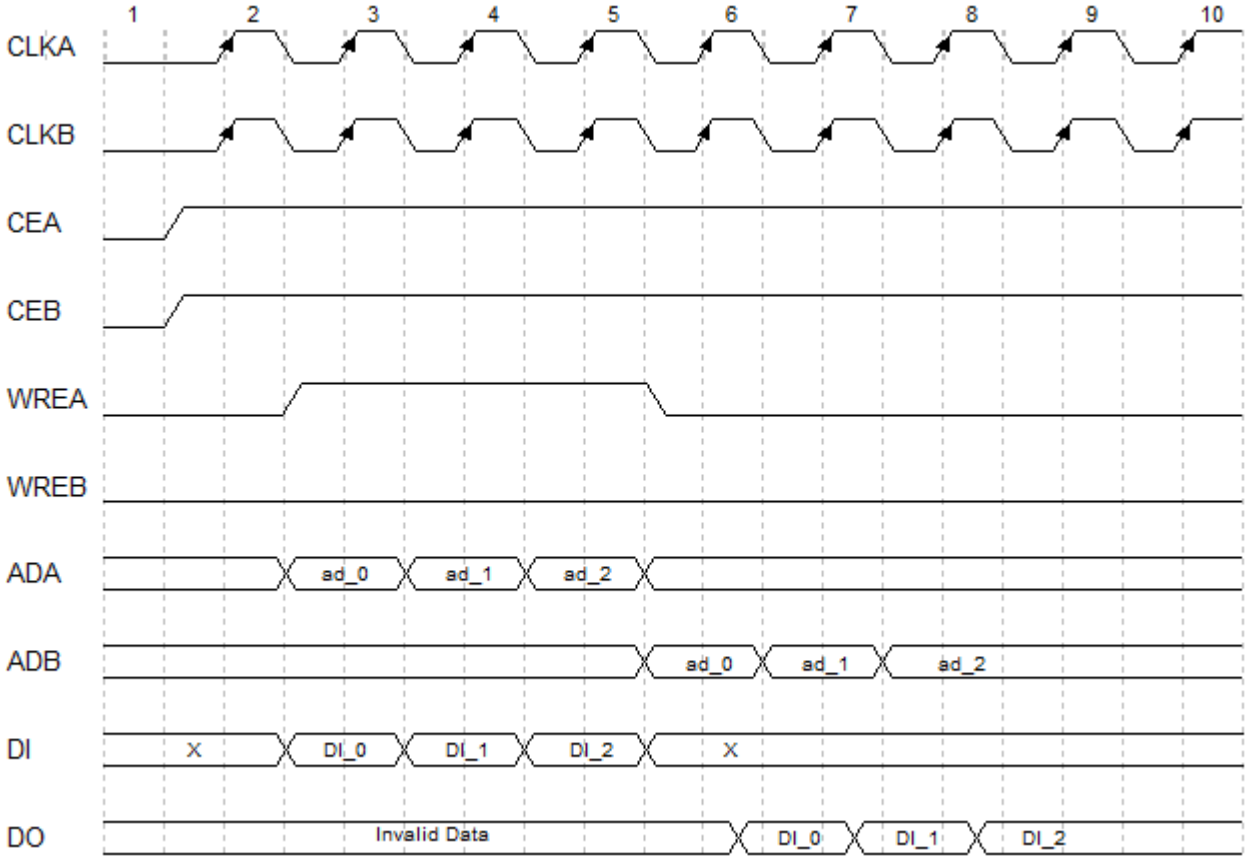
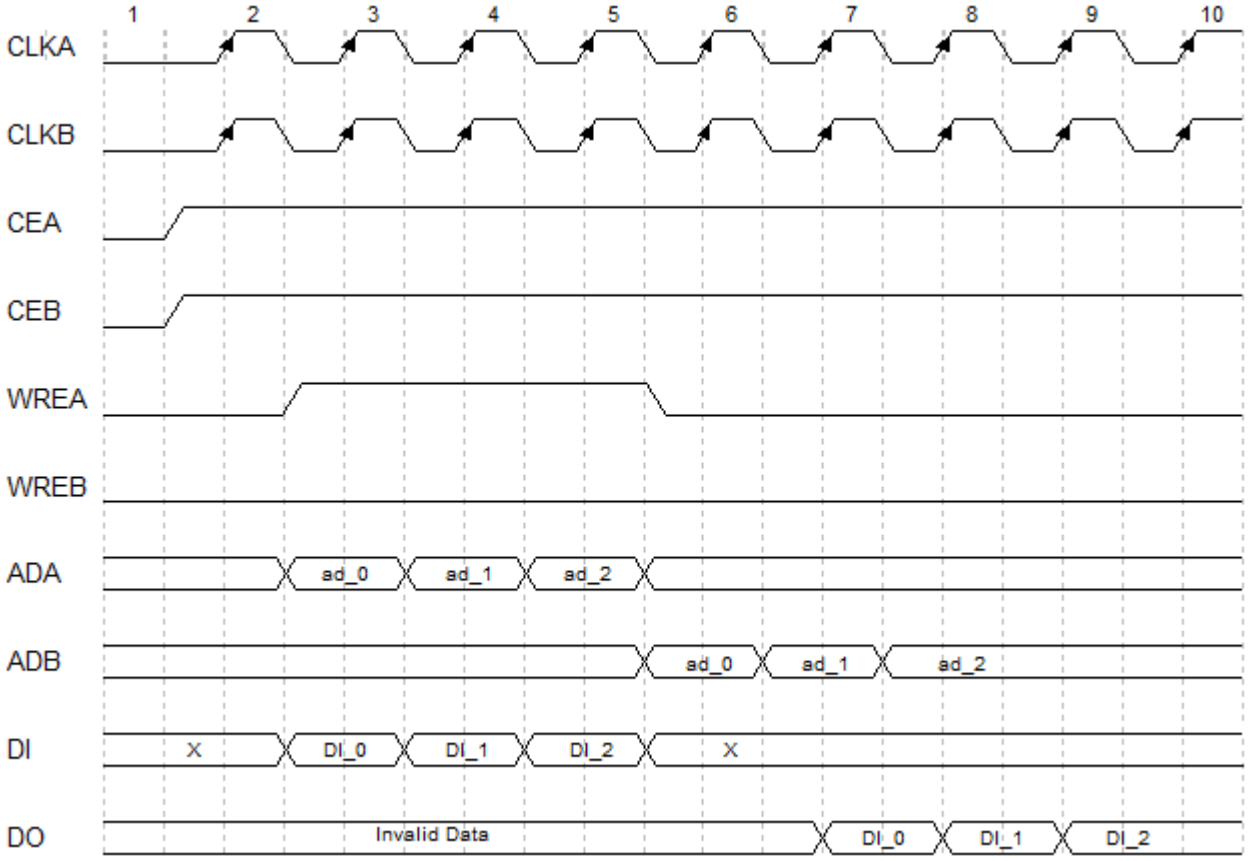


图 4-18 SDP/SDPX9 Normal 写模式时序波形图（Pipeline 读模式）



端口介绍

表 4-10 端口介绍

端口名	I/O	描述
DO[31:0]/DO[35:0]	Output	数据输出
DI[31:0]/DI[35:0]	Input	数据输入
ADA[13:0]	Input	A 端地址输入
ADB[13:0]	Input	B 端地址输入
WREA	Input	A 端写使能输入（一般配置为 1） 1：写入； 0：读出
WREB	Input	B 端写使能输入（一般配置为 0） 1：写入； 0：读出
CEA	Input	A 端时钟使能信号，高电平有效
CEB	Input	B 端时钟使能信号，高电平有效
CLKA	Input	A 端时钟输入
CLKB	Input	B 端时钟输入
RESETA	Input	A 端复位输入，支持同步复位和异步复位，高电平有效
RESETB	Input	B 端复位输入，支持同步复位和异步复

端口名	I/O	描述
		位，高电平有效
OCE	Input	输出时钟使能信号，用于 pipline 模式，对 bypass 模式无效
BLKSEL[2:0]	Input	BSRAM 块选择信号，用于需要多个 BSRAM 存储单元级联实现容量扩展

参数介绍

表 4-11 参数介绍

参数名	参数类型	取值范围	默认值	描述
READ_MODE	Integer	1’b0,1’b1	1’b0	读模式配置 • 1’b0:bypass 模式 • 1’b1:pipeline 模式
BIT_WIDTH_0	Integer	SDP:1,2,4,8,16,32 SDPX9:9,18,36	SDP:32 SDPX9:36	A 端数据宽度配置
BIT_WIDTH_1	Integer	SDP:1,2,4,8,16,32 SDPX9:9,18,36	SDP:32 SDPX9:36	B 端数据宽度配置
BLK_SEL	Integer	3’b000~3’b111	3’b000	BSRAM 块选择参数设置，与端口 BLKSEL 相等时该 BSRAM 被选中。使用 IP Core Generator 进行存储扩展时软件自动进行扩展处理。
RESET_MODE	String	SYNC,ASYNC	SYNC	复位模式配置 SYNC：同步复位 ASYNC：异步复位
INIT_RAM_00~ INIT_RAM_3F	Integer	SDP:256’h0…0~256’h1…1 SDPX9:288’h0…0~288’h1…1	SDP:256’h0…0 SDPX9:288’h0…0	用于设置 B-SRAM 存储单元的初始化数据

配置关系

表 4-12 数据宽度和地址深度配置关系

半双端口模式	BSRAM 容量	数据宽度	地址深度
SDP	16K	1	14
		2	13
		4	12
		8	11
		16	10
		32	9
SDPX9	18K	9	11
		18	10
		36	9

原语例化

示例一

Verilog 例化：

```
SDP bram_sdp_0 (
    .DO({dout[31:16],dout[15:0]}),
    .CLKA(clka),
    .CEA(cea),
    .RESETA(reseta),
    .WREA(wrea),
    .CLKB(clkb),
    .CEB(ceb),
    .RESETB(resetb),
    .WREB(wreb),
    .OCE(oce),
    .BLKSEL({3'b000}),
    .ADA({ada[9:0], 2'b00, byte_en[1:0]}),
    .DI({{16{1'b0}},din[15:0]}),
    .ADB({adb[9:0],4'b0000})
);
defparam bram_sdp_0.READ_MODE = 1'b1;
defparam bram_sdp_0.BIT_WIDTH_0 = 16;
defparam bram_sdp_0.BIT_WIDTH_1 = 16;
defparam bram_sdp_0.BLK_SEL = 3'b000;
defparam bram_sdp_0.RESET_MODE = "SYNC";
defparam bram_sdp_0.INIT_RAM_00 =
256'h00A000000000000B00A000000000000B00A000000000000B00
A000000000000B;
defparam bram_sdp_0.INIT_RAM_3F =
256'h00A000000000000B00A000000000000B00A000000000000B00
A000000000000B;
```

Vhdl 例化：

```
 COMPONENT  SDP
        GENERIC(
                BIT_WIDTH_0:integer:=16;
                BIT_WIDTH_1:integer:=16;
                READ_MODE:bit:='0';
                BLK_SEL:bit_vector:="000";
                RESET_MODE:string:="SYNC";
                INIT_RAM_00:bit_vector:=X"00A000000000000
B00A000000000000B00A000000000000B00A000000000000B";
                INIT_RAM_01:bit_vector:=X"00A000000000000
B00A000000000000B00A000000000000B00A000000000000B";
                INIT_RAM_3F:bit_vector:=X"00A000000000000
B00A000000000000B00A000000000000B00A000000000000B"
        );
        PORT(
                DO:OUT  std_logic_vector(31  downto  0):=conv_
std_logic_vector(0,32);
                CLKA,CLKB,CEA,CEB,OCE,RESETA,RESETB,
```

```
WREA,WREB:IN  std_logic;
                        ADA,ADB:IN  std_logic_vector(13  downto  0);
                        BLKSEL:IN  std_logic_vector(2  downto  0);
                        DI:IN  std_logic_vector(31  downto  0)
                    );
    END  COMPONENT;
    uut:SDP
        GENERIC  MAP(
                        BIT_WIDTH_0=>16,
                        BIT_WIDTH_1=>16,
                        READ_MODE=>'0',
                        BLK_SEL=>"000",
                        RESET_MODE=>"SYNC",
                        INIT_RAM_00=>X"00A000000000000B00A00
0000000000B00A000000000000B00A000000000000B",
                        INIT_RAM_01=>X"00A000000000000B00A00
0000000000B00A000000000000B00A000000000000B",
                        INIT_RAM_3F=>X"00A000000000000B00A00
0000000000B00A000000000000B00A000000000000B"
                        )
            PORT MAP(
                DO=>dout,
                CLKA=>clka,
                CEA=>cea,
                RESETA=>reseta,
                WREA=>wrea，
                CLKB=>clkb,
                CEB=>ceb,
                RESETB=>resetb,
                WREB=>wreb,
                OCE=>oce,
                BLKSEL=>blksel,
                ADA=>ada,
                DI=>din,
                ADB=>adb
            );
```

示例二

Verilog 例化：

```
SDPX9 bram_sdpx9_0 (
    .DO({dout[35:9],dout[8:0]}),
    .CLKA(clka),
    .CEA(cea),
    .RESETA(reseta),
    .WREA(wrea),
    .CLKB(clkb),
    .CEB(ceb),
    .RESETB(resetb),
    .WREB(wreb),
```

```
        .OCE(oce),
        .BLKSEL({3'b000}),
        .ADA({ada[10:0],3'b000}),
        .DI({{27{1'b0}},din[8:0]}),
        .ADB({adb[10:0],3'b000})
    );
    defparam bram_sdpx9_0.READ_MODE = 1'b0;
    defparam bram_sdpx9_0.BIT_WIDTH_0 = 9;
    defparam bram_sdpx9_0.BIT_WIDTH_1 = 9;
    defparam bram_sdpx9_0.BLK_SEL = 3'b000;
    defparam bram_sdpx9_0.RESET_MODE = "SYNC";
    defparam bram_sdpx9_0.INIT_RAM_00 =
    288'h000000000C000000000000D0000050000C000000000000D000
    0000000C000000000000D0;
    defparam bram_sdpx9_0.INIT_RAM_01 =
    288'h000000000C000000000000D0000000000C000000003000D000
    0000000C000000000040D0;
    defparam bram_sdpx9_0.INIT_RAM_3F =
    288'h0000A0000C000000000000D0000000000C000000000000D001
    0000000C000000000000D0;
```

Vhdl 例化：

```
    COMPONENT SDPX9
        GENERIC(
            BIT_WIDTH_0:integer:=18;
            BIT_WIDTH_1:integer:=18;
            READ_MODE:bit:='0';
            BLK_SEL:bit_vector:="000";
            RESET_MODE:string:="SYNC";
            INIT_RAM_00:bit_vector:=X"000000000C00000
0000000D0000050000C000000000000D0000000000C000000000000D0"
;
            INIT_RAM_01:bit_vector:=X"000000000C00000
0000000D0000000000C000000003000D0000000000C000000000040D0"
;
            INIT_RAM_3F:bit_vector:=X"0000A0000C00000
0000000D0000000000C000000000000D0010000000C000000000000D0"
        );
        PORT(
            DO:OUT std_logic_vector(35 downto 0):=conv
_std_logic_vector(0,36);
            CLKA,CLKB,CEA,CEB,OCE,RESETA,RESETB,
WREA,WREB:IN std_logic;
            ADA,ADB:IN std_logic_vector(13 downto 0);
            BLKSEL:IN std_logic_vector(2 downto 0);
            DI:IN std_logic_vector(35 downto 0)
        );
  END COMPONENT;
  uut:SDP
    GENERIC MAP(
            BIT_WIDTH_0=>18,
```

```
                BIT_WIDTH_1=>18,
                READ_MODE=>'0',
                BLK_SEL=>"000",
                RESET_MODE=>"SYNC",
                INIT_RAM_00=>X"000000000C000000000000D00
00050000C000000000000D0000000000C000000000000D0",
                INIT_RAM_01=>X"000000000C000000000000D00
00000000C000000003000D0000000000C000000000040D0",
                INIT_RAM_3F=>X"0000A0000C000000000000D00
00000000C000000000000D0010000000C000000000000D0"
    )
    PORT  MAP(
        DO=>dout,
        CLKA=>clka,
        CEA=>cea,
        RESETA=>reseta,
        WREA=>wrea，
        CLKB=>clkb,
        CEB=>ceb,
        RESETB=>resetb,
        WREB=>wreb,
        OCE=>oce,
        BLKSEL=>blksel,
        ADA=>ada,
        DI=>din,
        ADB=>adb
    );
```

4.5 SDPB/SDPX9B

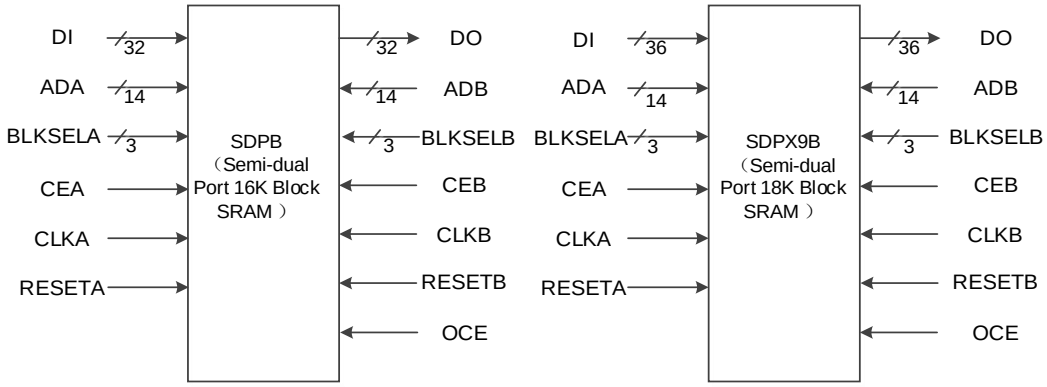
原语名称

SDPB/SDPX9B(Semi Dual Port 16K Block SRAM /Semi Dual Port 18K Block SRAM),16K/18K 半双端口 BSRAM。

SDPB/SDPX9B 是 SDP/SDPX9 的优化版本，A 端口和 B 端口分别独立支持块选择信号 BLKSELA 和 BLKSELB，同时删掉了写使能信号 WREA 和 WREB，推荐优先使用 SDPB/SDPX9B。

端口示意图

图 4-19 SDPB/SDPX9B 端口示意图



功能描述

SDPB/SDPX9B 存储空间分别为 16K bit/18K bit，其工作模式为半双端口模式，端口 A 进行写操作，端口 B 进行读操作，可支持 2 种读模式（bypass 模式和 pipeline 模式）和 1 种写模式（normal 模式）。

SDPB 配置为 16bit/32bit、SDPX9B 配置为 18bit/36bit 时，可实现 BSRAM 的 byte enable 功能，即通过写入地址端口 AD 的低四位控制写入存储器的数据，高电平使能。ADA[0]控制 DI[7:0]/DI[8:0]是否写入存储器，ADA[1]控制 DI[15:8]/DI[17:9]是否写入存储器，ADA[2]控制 DI[23:16]/DI[26:18]是否写入存储器，ADA[3]控制 DI[31:24]/DI[35:27]是否写入存储器。

读模式

通过参数 READ_MODE 来启用或禁用输出 pipeline 寄存器，使用输出 pipeline 寄存器时，读操作需要额外的延迟周期。

写模式

SDPB/SDPX9B 端口 A 进行写操作，端口 B 进行读操作，支持 normal 模式。

端口介绍

表 4-13 端口介绍

端口名	I/O	描述
DO[31:0]/DO[35:0]	Output	数据输出
DI[31:0]/DI[35:0]	Input	数据输入
ADA[13:0]	Input	A 端地址输入
ADB[13:0]	Input	B 端地址输入
CEA	Input	A 端时钟使能信号，高电平有效
CEB	Input	B 端时钟使能信号，高电平有效
CLKA	Input	A 端时钟输入
CLKB	Input	B 端时钟输入
RESETA	Input	A 端复位输入，支持同步复位和异步复位，高电平有效
RESETB	Input	B 端复位输入，支持同步复位和异步复位，高电平有效
OCE	Input	输出时钟使能信号，用于 pipline 模式，对 bypass 模式无效
BLKSELA[2:0]	Input	BSRAM A 端口块选择信号，用于需要多个 BSRAM 存储单元级联实现容量扩展
BLKSELB[2:0]	Input	BSRAM B 端口块选择信号，用于需要多个 BSRAM 存储单元级联实现容量扩展

参数介绍

表 4-14 参数介绍

参数名	参数类型	取值范围	默认值	描述
READ_MODE	Integer	1'b0,1'b1	1'b0	读模式配置 • 1'b0:bypass 模式 • 1'b1:pipeline 模式
BIT_WIDTH_0	Integer	SDPB:1,2,4,8,16,32 SDPX9B:9,18,36	SDPB:32 SDPX9B:36	A 端数据宽度配置
BIT_WIDTH_1	Integer	SDPB:1,2,4,8,16,32 SDPX9B:9,18,36	SDPB:32 SDPX9B:36	B 端数据宽度配置
BLK_SEL_0	Integer	3'b000~3'b111	3'b000	BSRAM A 端口块选择参数设置，与端口 BLKSEL 相等时该 BSRAM 被选中。使用 IP Core Generator 进行存储扩展时软件自动进行扩展处理。
BLK_SEL_1	Integer	3'b000~3'b111	3'b000	BSRAM B 端口块选择参数设置，与端口 BLKSEL 相等时该 BSRAM 被选中。使用 IP Core Generator 进行存储扩展时软件自动进行扩展处理。
RESET_MODE	String	SYNC,ASYNC	SYNC	复位模式配置 SYNC：同步复位 ASYNC：异步复位
INIT_RAM_00~ INIT_RAM_3F	Integer	SDPB:256'h0…0~256'h1…1 SDPX9B:288'h0…0~288'h1…1	SDPB:256'h0…0 SDPX9B:288'h0…0	用于设置 B-SRAM 存储单元的初始化数据

配置关系

表 4-15 数据宽度和地址深度配置关系

半双端口模式	BSRAM 容量	数据宽度	地址深度
SDPB	16K	1	14
		2	13
		4	12
		8	11
		16	10
		32	9
SDPX9B	18K	9	11
		18	10
		36	9

原语例化

示例一

Verilog 例化：

```
SDPB bram_sdpb_0 (
    .DO({dout[31:16],dout[15:0]}),
    .CLKA(clka),
    .CEA(cea),
    .RESETA(reseta),
    .CLKB(clkb),
    .CEB(ceb),
    .RESETB(resetb),
    .OCE(oce),
    .BLKSELA({3'b000}),
    .BLKSELB({3'b000}),
    .ADA({ada[9:0], 2'b00, byte_en[1:0]}),
    .DI({{16{1'b0}},din[15:0]}),
    .ADB({adb[9:0],4'b0000})
);
defparam bram_sdpb_0.READ_MODE = 1'b1;
defparam bram_sdpb_0.BIT_WIDTH_0 = 16;
defparam bram_sdpb_0.BIT_WIDTH_1 = 16;
defparam bram_sdpb_0.BLK_SEL_0 = 3'b000;
defparam bram_sdpb_0.BLK_SEL_1 = 3'b000;
defparam bram_sdpb_0.RESET_MODE = "SYNC";
defparam bram_sdpb_0.INIT_RAM_00 =
256'h00A000000000000B00A000000000000B00A000000000000B00
A000000000000B;
defparam bram_sdpb_0.INIT_RAM_3F =
256'h00A000000000000B00A000000000000B00A000000000000B00
A000000000000B;
```

Vhdl 例化：

```
 COMPONENT  SDPB
        GENERIC(
                BIT_WIDTH_0:integer:=16;
                BIT_WIDTH_1:integer:=16;
                READ_MODE:bit:='0';
                BLK_SEL_0:bit_vector:="000";
                BLK_SEL_1:bit_vector:="000";
                RESET_MODE:string:="SYNC";
                INIT_RAM_00:bit_vector:=X"00A000000000000
B00A000000000000B00A000000000000B00A000000000000B";
                INIT_RAM_01:bit_vector:=X"00A000000000000
B00A000000000000B00A000000000000B00A000000000000B";
                INIT_RAM_3F:bit_vector:=X"00A000000000000
B00A000000000000B00A000000000000B00A000000000000B"
        );
        PORT(
                DO:OUT  std_logic_vector(31  downto  0):=conv_
std_logic_vector(0,32);
                CLKA,CLKB,CEA,CEB:IN  std_logic;
                OCE,RESETA,RESETB:IN  std_logic;
                ADA,ADB:IN  std_logic_vector(13  downto  0);
```

```
                    BLKSELA:IN std_logic_vector(2 downto 0);
                    BLKSELB:IN std_logic_vector(2 downto 0);
                    DI:IN std_logic_vector(31 downto 0)
                );
    END COMPONENT;
    uut:SDPB
        GENERIC MAP(
                    BIT_WIDTH_0=>16,
                    BIT_WIDTH_1=>16,
                    READ_MODE=>'0',
                    BLK_SEL_0=>"000",
                    BLK_SEL_1=>"000",
                    RESET_MODE=>"SYNC",
                    INIT_RAM_00=>X"00A000000000000B00A00
0000000000B00A000000000000B00A000000000000B",
                    INIT_RAM_01=>X"00A000000000000B00A00
0000000000B00A000000000000B00A000000000000B",
                    INIT_RAM_3F=>X"00A000000000000B00A00
0000000000B00A000000000000B00A000000000000B"
                    )
            PORT MAP(
                DO=>dout,
                CLKA=>clka,
                CEA=>cea,
                RESETA=>reseta,
                CLKB=>clkb,
                CEB=>ceb,
                RESETB=>resetb,
                OCE=>oce,
                BLKSELA=>blksela,
                BLKSELB=>blkselb,
                ADA=>ada,
                DI=>din,
                ADB=>adb
            );
```

示例二

Verilog 例化：

```
SDPX9B bram_sdpx9b_0 (
    .DO({dout[35:9],dout[8:0]}),
    .CLKA(clka),
    .CEA(cea),
    .RESETA(reseta),
    .CLKB(clkb),
    .CEB(ceb),
    .RESETB(resetb),
    .OCE(oce),
    .BLKSELA({3'b000}),
    .BLKSELB({3'b000}),
    .ADA({ada[10:0],3'b000}),
```

```
        .DI({{27{1'b0}},din[8:0]}),
        .ADB({adb[10:0],3'b000})
    );
    defparam bram_sdpx9b_0.READ_MODE = 1'b0;
    defparam bram_sdpx9b_0.BIT_WIDTH_0 = 9;
    defparam bram_sdpx9b_0.BIT_WIDTH_1 = 9;
    defparam bram_sdpx9b_0.BLK_SEL_0 = 3'b000;
    defparam bram_sdpx9b_0.BLK_SEL_1 = 3'b000;
    defparam bram_sdpx9b_0.RESET_MODE = "SYNC";
    defparam bram_sdpx9b_0.INIT_RAM_00 =
    288'h000000000C000000000000D0000050000C000000000000D000
    0000000C000000000000D0;
    defparam bram_sdpx9b_0.INIT_RAM_01 =
    288'h000000000C000000000000D0000000000C000000003000D000
    0000000C000000000040D0;
    defparam bram_sdpx9b_0.INIT_RAM_3F =
    288'h0000A0000C000000000000D0000000000C000000000000D001
    0000000C000000000000D0;
```

Vhdl 例化：

```
    COMPONENT SDPX9B
        GENERIC(
            BIT_WIDTH_0:integer:=18;
            BIT_WIDTH_1:integer:=18;
            READ_MODE:bit:='0';
            BLK_SEL_0:bit_vector:="000";
            BLK_SEL_1:bit_vector:="000";
            RESET_MODE:string:="SYNC";
            INIT_RAM_00:bit_vector:=X"000000000C00000
0000000D0000050000C000000000000D0000000000C000000000000D0"
;
            INIT_RAM_01:bit_vector:=X"000000000C00000
0000000D0000000000C000000003000D0000000000C000000000040D0"
;
            INIT_RAM_3F:bit_vector:=X"0000A0000C00000
0000000D0000000000C000000000000D0010000000C000000000000D0"
        );
        PORT(
            DO:OUT std_logic_vector(35 downto 0):=conv
_std_logic_vector(0,36);
            CLKA,CLKB,CEA,CEB:IN std_logic;
            OCE,RESETA,RESETB:IN std_logic;
            ADA,ADB:IN std_logic_vector(13 downto 0);
            BLKSELA:IN std_logic_vector(2 downto 0);
            BLKSELB:IN std_logic_vector(2 downto 0);
            DI:IN std_logic_vector(35 downto 0)
        );
  END COMPONENT;
  uut:SDPB
    GENERIC MAP(
            BIT_WIDTH_0=>18,
```

```
                BIT_WIDTH_1=>18,
                READ_MODE=>'0',
                BLK_SEL_0=>"000",
                BLK_SEL_1=>"000",
                RESET_MODE=>"SYNC",
                INIT_RAM_00=>X"000000000C000000000000D00
00050000C000000000000D0000000000C000000000000D0",
                INIT_RAM_01=>X"000000000C000000000000D00
00000000C000000003000D0000000000C000000000040D0",
                INIT_RAM_3F=>X"0000A0000C000000000000D00
00000000C000000000000D0010000000C000000000000D0"
    )
    PORT  MAP(
        DO=>dout,
        CLKA=>clka,
        CEA=>cea,
        RESETA=>reseta,
        CLKB=>clkb,
        CEB=>ceb,
        RESETB=>resetb,
        OCE=>oce,
        BLKSELA=>blksela,
        BLKSELB=>blkselb,
        ADA=>ada,
        DI=>din,
        ADB=>adb
    );
```

4.6 rSDP/rSDPX9

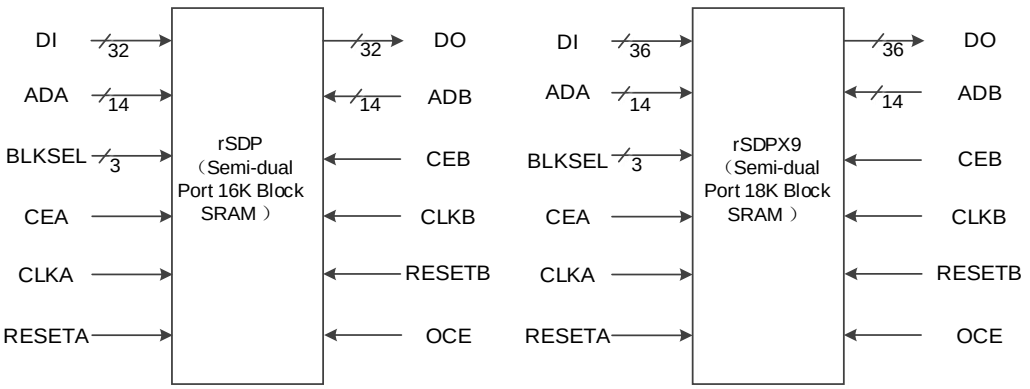
原语名称

rSDP/rSDPX9(Semi Dual Port 16K Block SRAM /Semi Dual Port 18K Block SRAM),16K/18K 半双端口 BSRAM。

rSDP/rSDPX9 是 SDP/SDPX9 的修订版，删掉了写使能信号 WREA 和 WREB。

端口示意图

图 4-20 rSDP/rSDPX9 端口示意图



功能描述

rSDP/rSDPX9 存储空间分别为 16K bit/18K bit，其工作模式为半双端口模式，端口 A 进行写操作，端口 B 进行读操作，可支持 2 种读模式（bypass 模式和 pipeline 模式）和 1 种写模式（normal 模式）。

rSDP 配置为 16bit/32bit、rSDPX9 配置为 18bit/36bit 时，可实现 BSRAM 的 byte enable 功能，即通过写入地址端口 AD 的低四位控制写入存储器的数据，高电平使能。ADA[0]控制 DI[7:0]/DI[8:0]是否写入存储器，ADA[1]控制 DI[15:8]/DI[17:9]是否写入存储器，ADA[2]控制 DI[23:16]/DI[26:18]是否写入存储器，ADA[3]控制 DI[31:24]/DI[35:27]是否写入存储器。

读模式

通过参数 READ_MODE 来启用或禁用输出 pipeline 寄存器，使用输出 pipeline 寄存器时，读操作需要额外的延迟周期。

写模式

rSDP/rSDPX9 端口 A 进行写操作，端口 B 进行读操作，支持 normal 模式。不同模式对应的内部时序波形图同 SDP/SDPX9。

端口介绍

表 4-16 端口介绍

端口名	I/O	描述
DO[31:0]/DO[35:0]	Output	数据输出
DI[31:0]/DI[35:0]	Input	数据输入
ADA[13:0]	Input	A 端地址输入
ADB[13:0]	Input	B 端地址输入
CEA	Input	A 端时钟使能信号，高电平有效
CEB	Input	B 端时钟使能信号，高电平有效
CLKA	Input	A 端时钟输入
CLKB	Input	B 端时钟输入
RESETA	Input	A 端复位输入，支持同步复位和异步复位，高电平有效
RESETB	Input	B 端复位输入，支持同步复位和异步复位，高电平有效
OCE	Input	输出时钟使能信号，用于 pipline 模式，对 bypass 模式无效
BLKSEL[2:0]	Input	BSRAM 块选择信号，用于需要多个 BSRAM 存储单元级联实现容量扩展

参数介绍

表 4-17 参数介绍

参数名	参数类型	取值范围	默认值	描述
READ_MODE	Integer	1’b0,1’b1	1’b0	读模式配置 • 1’b0:bypass 模式

参数名	参数类型	取值范围	默认值	描述
				• 1’b1:pipeline 模式
BIT_WIDTH_0	Integer	rSDP:1,2,4,8,16,32 rSDPX9:9,18,36	rSDP:32 rSDPX9:36	A 端数据宽度配置
BIT_WIDTH_1	Integer	rSDP:1,2,4,8,16,32 rSDPX9:9,18,36	rSDP:32 rSDPX9:36	B 端数据宽度配置
BLK_SEL	Integer	3’b000~3’b111	3’b000	BSRAM 块选择参数设置，与端口 BLKSEL 相等时该 BSRAM 被选中。使用 IP Core Generator 进行存储扩展时软件自动进行扩展处理。
RESET_MODE	String	SYNC,ASYNC	SYNC	复位模式配置 SYNC：同步复位 ASYNC：异步复位
INIT_RAM_00~ INIT_RAM_3F	Integer	rSDP:256’h0…0~256’h1…1 rSDPX9:288’h0…0~288’h1…1	rSDP:256’h0…0 rSDPX9:288’h0…0	用于设置 B-SRAM 存储单元的初始化数据

配置关系

表 4-18 数据宽度和地址深度配置关系

半双端口模式	BSRAM 容量	数据宽度	地址深度
rSDP	16K	1	14
		2	13
		4	12
		8	11
		16	10
		32	9
rSDPX9	18K	9	11
		18	10
		36	9

原语例化

示例一

Verilog 例化：

```
rSDP bram_rsdp_0 (
    .DO({dout[31:16],dout[15:0]}),
    .CLKA(clka),
    .CEA(cea),
    .RESETA(reseta),
    .CLKB(clkb),
    .CEB(ceb),
    .RESETB(resetb),
    .OCE(oce),
```

```
        .BLKSEL({3'b000}),
        .ADA({ada[9:0], 2'b00, byte_en[1:0]}),
        .DI({{16{1'b0}},din[15:0]}),
        .ADB({adb[9:0],4'b0000})
    );
    defparam bram_rsdp_0.READ_MODE = 1'b1;
    defparam bram_rsdp_0.BIT_WIDTH_0 = 16;
    defparam bram_rsdp_0.BIT_WIDTH_1 = 16;
    defparam bram_rsdp_0.BLK_SEL = 3'b000;
    defparam bram_rsdp_0.RESET_MODE = "SYNC";
    defparam bram_rsdp_0.INIT_RAM_00 =
    256'h00A000000000000B00A000000000000B00A000000000000B00
    A000000000000B;
    defparam bram_rsdp_0.INIT_RAM_3F =
    256'h00A000000000000B00A000000000000B00A000000000000B00
    A000000000000B;
```

Vhdl 例化：

```
    COMPONENT  rSDP
            GENERIC(
                    BIT_WIDTH_0:integer:=16;
                    BIT_WIDTH_1:integer:=16;
                    READ_MODE:bit:='0';
                    BLK_SEL:bit_vector:="000";
                    RESET_MODE:string:="SYNC";
                    INIT_RAM_00:bit_vector:=X"00A000000000000
B00A000000000000B00A000000000000B00A000000000000B";
                    INIT_RAM_01:bit_vector:=X"00A000000000000
B00A000000000000B00A000000000000B00A000000000000B";
                    INIT_RAM_3F:bit_vector:=X"00A000000000000
B00A000000000000B00A000000000000B00A000000000000B"
                );
                PORT(
                    DO:OUT  std_logic_vector(31  downto  0):=conv_
std_logic_vector(0,32);
                    CLKA,CLKB,CEA,CEB,OCE,RESETA,RESETB:I
N  std_logic;
                    ADA,ADB:IN  std_logic_vector(13  downto  0);
                    BLKSEL:IN  std_logic_vector(2  downto  0);
                    DI:IN  std_logic_vector(31  downto  0)
                );
    END  COMPONENT;
    uut:rSDP
        GENERIC  MAP(
                    BIT_WIDTH_0=>16,
                    BIT_WIDTH_1=>16,
                    READ_MODE=>'0',
                    BLK_SEL=>"000",
                    RESET_MODE=>"SYNC",
                    INIT_RAM_00=>X"00A000000000000B00A00
0000000000B00A000000000000B00A000000000000B",
```

```
                INIT_RAM_01=>X"00A000000000000B00A00
0000000000B00A000000000000B00A000000000000B",
                INIT_RAM_3F=>X"00A000000000000B00A00
0000000000B00A000000000000B00A000000000000B"
                )
        PORT MAP(
            DO=>dout,
            CLKA=>clka,
            CEA=>cea,
            RESETA=>reseta,
            CLKB=>clkb,
            CEB=>ceb,
            RESETB=>resetb,
            OCE=>oce,
            BLKSEL=>blksel,
            ADA=>ada,
            DI=>din,
            ADB=>adb
        );
```

示例二

Verilog 例化：

```
rSDPX9 bram_rsdpx9_0 (
    .DO({dout[35:9],dout[8:0]}),
    .CLKA(clka),
    .CEA(cea),
    .RESETA(reseta),
    .CLKB(clkb),
    .CEB(ceb),
    .RESETB(resetb),
    .OCE(oce),
    .BLKSEL({3'b000}),
    .ADA({ada[10:0],3'b000}),
    .DI({{27{1'b0}},din[8:0]}),
    .ADB({adb[10:0],3'b000})
);
defparam bram_rsdpx9_0.READ_MODE = 1'b0;
defparam bram_rsdpx9_0.BIT_WIDTH_0 = 9;
defparam bram_rsdpx9_0.BIT_WIDTH_1 = 9;
defparam bram_rsdpx9_0.BLK_SEL = 3'b000;
defparam bram_rsdpx9_0.RESET_MODE = "SYNC";
defparam bram_rsdpx9_0.INIT_RAM_00 =
288'h000000000C000000000000D0000050000C000000000000D000
0000000C000000000000D0;
defparam bram_rsdpx9_0.INIT_RAM_01 =
288'h000000000C000000000000D0000000000C000000003000D000
0000000C000000000040D0;
defparam bram_rsdpx9_0.INIT_RAM_3F =
288'h0000A0000C000000000000D0000000000C000000000000D001
0000000C000000000000D0;
```

Vhdl 例化：

```
    COMPONENT rSDPX9
            GENERIC(
                    BIT_WIDTH_0:integer:=18;
                    BIT_WIDTH_1:integer:=18;
                    READ_MODE:bit:='0';
                    BLK_SEL:bit_vector:="000";
                    RESET_MODE:string:="SYNC";
                    INIT_RAM_00:bit_vector:=X"000000000C00000
0000000D0000050000C000000000000D0000000000C000000000000D0"
;
                    INIT_RAM_01:bit_vector:=X"000000000C00000
0000000D0000000000C000000003000D0000000000C000000000040D0"
;
                    INIT_RAM_3F:bit_vector:=X"0000A0000C00000
0000000D0000000000C000000000000D0010000000C000000000000D0"
            );
            PORT(
                    DO:OUT std_logic_vector(35 downto 0):=conv
_std_logic_vector(0,36);
                    CLKA,CLKB,CEA,CEB,OCE,RESETA,RESETB:
IN std_logic;
                    ADA,ADB:IN std_logic_vector(13 downto 0);
                    BLKSEL:IN std_logic_vector(2 downto 0);
                    DI:IN std_logic_vector(35 downto 0)
            );
  END COMPONENT;
  uut:rSDP
      GENERIC MAP(
                    BIT_WIDTH_0=>18,
                    BIT_WIDTH_1=>18,
                    READ_MODE=>'0',
                    BLK_SEL=>"000",
                    RESET_MODE=>"SYNC",
                    INIT_RAM_00=>X"000000000C000000000000D00
00050000C000000000000D0000000000C000000000000D0",
                    INIT_RAM_01=>X"000000000C000000000000D00
00000000C000000003000D0000000000C000000000040D0",
                    INIT_RAM_3F=>X"0000A0000C000000000000D00
00000000C000000000000D0010000000C000000000000D0"
      )
      PORT MAP(
          DO=>dout,
          CLKA=>clka,
          CEA=>cea,
          RESETA=>reseta,
          CLKB=>clkb,
          CEB=>ceb,
          RESETB=>resetb,
          OCE=>oce,
```

```
        BLKSEL=>blksel,
        ADA=>ada,
        DI=>din,
        ADB=>adb
    );
```

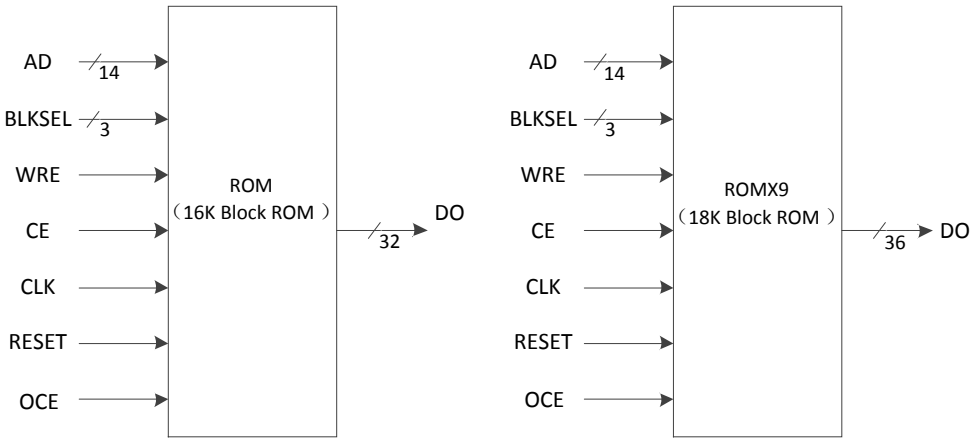
4.7 ROM/ROMX9

原语名称

ROM/ROMX9(16K/18K Block ROM),16K/18K 块状只读储存器。

端口示意图

图 4-21 ROM/ROMX9 端口示意图



功能描述

ROM/ROMX9 存储空间分别为 16K bit/18K bit，其工作模式为只读模式，可支持 2 种读模式（bypass 模式和 pipeline 模式）。

读模式

通过参数 READ_MODE 来启用或禁用输出 pipeline 寄存器，使用输出 pipeline 寄存器时，读操作需要额外的延迟周期。ROM/ROMX9 不同读模式对应的内部时序波形图可参考 DP/DPX9 时序图 4-2 到图 4-3（WRE=0）。

端口介绍

表 4-19 端口介绍

端口名	I/O	描述
DO[31:0]/DO[35:0]	Output	数据输出
AD[13:0]	Input	地址输入
WRE	Input	写使能输入（一般配置为 0）
CE	Input	时钟使能输入，高电平有效
CLK	Input	时钟输入
RESET	Input	复位输入，支持同步复位和异步复位，高电平有效
OCE	Input	输出时钟使能信号，用于 pipline

端口名	I/O	描述
		模式，对 bypass 模式无效
BLKSEL[2:0]	Input	BSRAM 块选择信号，用于需要多个 BSRAM 存储单元级联实现容量扩展

参数介绍

表 4-20 参数介绍

参数名	参数类型	取值范围	默认值	描述
READ_MODE	Integer	1’b0,1’b1	1’b0	读模式配置 1’b0:bypass 模式 1’b1:pipeline 模式
BIT_WIDTH	Integer	ROM:1,2,4,8,16,32 ROMX9:9,18,36	ROM:32 ROMX9:36	数据宽度配置
BLK_SEL	Integer	3’b000~3’b111	3’b000	BSRAM 块选择参数设置，与端口 BLKSEL 相等时该 BSRAM 被选中。使用 IP Core Generator 进行存储扩展时软件自动进行扩展处理。
RESET_MODE	String	SYNC,ASYNC	SYNC	复位模式配置 SYNC：同步复位 ASYNC：异步复位
INIT_RAM_00~ INIT_RAM_3F	Integer	ROM:256’h0…0~256’h1…1 ROMX9:288’h0…0~288’h1…1	ROM:256’h0…0 ROMX9:288’h0…0	用于设置 B-SRAM 存储单元的初始化数据

配置关系

表 4-21 配置关系

只读模式	BSRAM 容量	数据宽度	地址深度
ROM	16K	1	14
		2	13
		4	12
		8	11
		16	10
		32	9
ROMX9	18K	9	11
		18	10
		36	9

原语例化

示例一

Verilog 例化：

```
ROM bram_rom_0 (
    .DO({dout[31:8],dout[7:0]}),
    .CLK(clk),
    .OCE(oce),
    .CE(ce),
    .RESET(reset),
    .WRE(wre),
    .BLKSEL({3'b000}),
    .AD({ad[10:0],3'b000})
);
defparam bram_rom_0.READ_MODE = 1'b0;
defparam bram_rom_0.BIT_WIDTH = 8;
defparam bram_rom_0.BLK_SEL = 3'b000;
defparam bram_rom_0.RESET_MODE = "SYNC";
defparam bram_rom_0.INIT_RAM_00 =
256'h9C23645D0F78986FFC3E36E141541B95C19F2F7164085E631
A819860D8FF0000;
defparam bram_rom_0.INIT_RAM_01 =
256'h000000000000000000000000000000000000000000000000
000FFFFFFBDCF;
```

Vhdl 例化：

```
COMPONENT ROM
        GENERIC(
                BIT_WIDTH:integer:=1;
                READ_MODE:bit:='0';
                BLK_SEL:bit_vector:="000";
                RESET_MODE:string:="SYNC";
                INIT_RAM_00:bit_vector:=X"9C23645D0F78986FF
C3E36E141541B95C19F2F7164085E631A819860D8FF0000";
                INIT_RAM_01:bit_vector:=X"000000000000000000
0000000000000000000000000000000000FFFFFFBDCF"
        );
        PORT(
                DO:OUT std_logic_vector(31 downto 0):=conv_std
_logic_vector(0,32);
                CLK,CE,OCE,RESET,WRE:IN std_logic;
                BLKSEL:IN std_logic_vector(2 downto 0);
                AD:IN std_logic_vector(13 downto 0)
        );
END COMPONENT;
uut:ROM
    GENERIC MAP(
                BIT_WIDTH=>1,
                READ_MODE=>'0',
                BLK_SEL=>"000",
                RESET_MODE=>"SYNC",
                INIT_RAM_00=>X"9C23645D0F78986FFC3E36
E141541B95C19F2F7164085E631A819860D8FF0000",
                INIT_RAM_01=>X"000000000000000000000000
```

```
0000000000000000000000000000FFFFFFBDCF  "
        )
        PORT MAP(
                DO=>do,
                AD=>ad,
                CLK=>clk,
                CE=>ce,
                OCE=>oce,
                RESET=>reset,
                WRE=>wre,
                BLKSEL=>blksel
        );
```

示例二

Verilog 例化：

```
ROMX9 bram_romx9_0 (
    .DO({dout[35:9],dout[8:0]}),
    .CLK(clk),
    .OCE(oce),
    .CE(ce),
    .RESET(reset),
    .WRE(wre),
    .BLKSEL({3'b000}),
    .AD({ad[10:0],3'b000})
);
defparam bram_romx9_0.READ_MODE = 1'b0;
defparam bram_romx9_0.BIT_WIDTH = 9;
defparam bram_romx9_0.BLK_SEL = 3'b000;
defparam bram_romx9_0.RESET_MODE = "SYNC";
defparam bram_romx9_0.INIT_RAM_00 =
288'hCE08CC85D07DE1316FFE0F86DE1A09523795E0E7E5E71B2
020BC630D6053160EC7FC0000;
defparam bram_romx9_0.INIT_RAM_01 =
288'h000000000000000000000000000000000000000000000000000
0000000001FFFFFFF7ACF;
```

Vhdl 例化：

```
COMPONENT ROMX9
        GENERIC(
                BIT_WIDTH:integer:=9;
                READ_MODE:bit:='0';
                BLK_SEL:bit_vector:="000";
                RESET_MODE:string:="SYNC";
                INIT_RAM_00:bit_vector:=X"CE08CC85D07DE131
6FFE0F86DE1A09523795E0E7E5E71B2020BC630D6053160EC7FC000
0";
                INIT_RAM_01:bit_vector:=X"000000000000000000
00000000000000000000000000000000000000001FFFFFFF7ACF"
        );
        PORT(
```

```
                DO:OUT std_logic_vector(35 downto 0):=conv_std
_logic_vector(0,36);
                CLK,CE,OCE,RESET,WRE:IN std_logic;
                BLKSEL:IN std_logic_vector(2 downto 0);
                AD:IN std_logic_vector(13 downto 0)
        );
    END COMPONENT;
    Uut:ROMX9
        GENERIC MAP(
                BIT_WIDTH=>9,
                READ_MODE=>'0',
                BLK_SEL=>"000",
                RESET_MODE=>"SYNC",
                INIT_RAM_00=>X"CE08CC85D07DE1316F
FE0F86DE1A09523795E0E7E5E71B2020BC630D6053160EC7FC0000",
                INIT_RAM_01=>X"00000000000000000000
0000000000000000000000000000000000000001FFFFFFF7ACF"
        )
        PORT MAP(
            DO=>do,
            AD=>ad,
            CLK=>clk,
            CE=>ce,
            OCE=>oce,
            RESET=>reset,
            WRE=>wre,
            BLKSEL=>blksel
        );
```

4.8 rROM/rROMX9

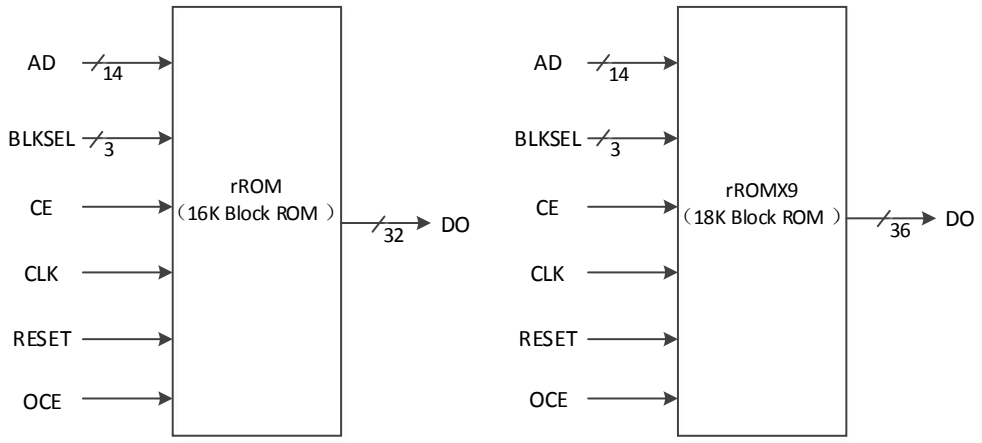
原语名称

rROM/rROMX9(16K/18K Block ROM),16K/18K 块状只读储存器。

rROM/rROMX9 是 ROM/ROMX9 的修订版，删掉了写使能信号 WRE。

端口示意图

图 4-22 rROM/rROMX9 端口示意图



功能描述

rROM/rROMX9 存储空间分别为 16K bit/18K bit，其工作模式为只读模式，可支持 2 种读模式（bypass 模式和 pipeline 模式）。

读模式

通过参数 READ_MODE 来启用或禁用输出 pipeline 寄存器，使用输出 pipeline 寄存器时，读操作需要额外的延迟周期。

端口介绍

表 4-22 端口介绍

端口名	I/O	描述
DO[31:0]/DO[35:0]	Output	数据输出
AD[13:0]	Input	地址输入
CE	Input	时钟使能输入，高电平有效
CLK	Input	时钟输入
RESET	Input	复位输入，支持同步复位和异步复位，高电平有效
OCE	Input	输出时钟使能信号，用于 pipline 模式，对 bypass 模式无效
BLKSEL[2:0]	Input	BSRAM 块选择信号，用于需要多个 BSRAM 存储单元级联实现容量扩展

参数介绍

表 4-23 参数介绍

参数名	参数类型	取值范围	默认值	描述
READ_MODE	Integer	1’b0,1’b1	1’b0	读模式配置 1’b0:bypass 模式 1’b1:pipeline 模式
BIT_WIDTH	Integer	rROM:1,2,4,8,16,32 rROMX9:9,18,36	rROM:32 rROMX9:36	数据宽度配置
BLK_SEL	Integer	3’b000~3’b111	3’b000	BSRAM 块选择参数设置，与端口 BLKSEL 相等时该 BSRAM 被选中。使用 IP Core Generator 进行存储扩展时软件自动进行扩展处理。
RESET_MODE	String	SYNC,ASYNC	SYNC	复位模式配置 SYNC：同步复位 ASYNC：异步复位
INIT_RAM_00~ INIT_RAM_3F	Integer	rROM:256’h0…0~256’h1…1 rROMX9:288’h0…0~288’h1…1	rROM:256’h0…0 rROMX9:288’h0…0	用于设置 B-SRAM 存储单元的初始化数据

配置关系

表 4-24 配置关系

只读模式	BSRAM 容量	数据宽度	地址深度
rROM	16K	1	14
		2	13
		4	12
		8	11
		16	10
		32	9
rROMX9	18K	9	11
		18	10
		36	9

原语例化

示例一

Verilog 例化：

```
rROM bram_rrom_0 (
    .DO({dout[31:8],dout[7:0]}),
    .CLK(clk),
    .OCE(oce),
    .CE(ce),
    .RESET(reset),
    .BLKSEL({3’b000}),
    .AD({ad[10:0],3’b000})
);
defparam bram_rrom_0.READ_MODE = 1'b0;
defparam bram_rrom_0.BIT_WIDTH = 8;
defparam bram_rrom_0.BLK_SEL = 3'b000;
defparam bram_rrom_0.RESET_MODE = "SYNC";
defparam bram_rrom_0.INIT_RAM_00 =
256'h9C23645D0F78986FFC3E36E141541B95C19F2F7164085E631
A819860D8FF0000;
defparam bram_rrom_0.INIT_RAM_01 =
256'h000000000000000000000000000000000000000000000000
000FFFFFFBDCF;
```

Vhdl 例化：

```
COMPONENT rROM
        GENERIC(
                BIT_WIDTH:integer:=1;
                READ_MODE:bit:='0';
                BLK_SEL:bit_vector:="000";
                RESET_MODE:string:="SYNC";
                INIT_RAM_00:bit_vector:=X"9C23645D0F78986FF
C3E36E141541B95C19F2F7164085E631A819860D8FF0000";
                INIT_RAM_01:bit_vector:=X"000000000000000000
```

```
0000000000000000000000000000000000FFFFFFBDCF"
            );
            PORT(
                    DO:OUT std_logic_vector(31 downto 0):=conv_std
_logic_vector(0,32);
                    CLK,CE,OCE,RESET:IN std_logic;
                    BLKSEL:IN std_logic_vector(2 downto 0);
                    AD:IN std_logic_vector(13 downto 0)
            );
    END COMPONENT;
    uut:rROM
        GENERIC MAP(
                    BIT_WIDTH=>1,
                    READ_MODE=>'0',
                    BLK_SEL=>"000",
                    RESET_MODE=>"SYNC",
                    INIT_RAM_00=>X"9C23645D0F78986FFC3E36
E141541B95C19F2F7164085E631A819860D8FF0000",
                    INIT_RAM_01=>X"0000000000000000000000000
0000000000000000000000000000FFFFFFBDCF  "
        )
        PORT MAP(
                DO=>do,
                AD=>ad,
                CLK=>clk,
                CE=>ce,
                OCE=>oce,
                RESET=>reset,
                BLKSEL=>blksel
        );
```

示例二

Verilog 例化：

```
rROMX9 bram_rromx9_0 (
    .DO({dout[35:9],dout[8:0]}),
    .CLK(clk),
    .OCE(oce),
    .CE(ce),
    .RESET(reset),
    .BLKSEL({3’b000}),
    .AD({ad[10:0],3’b000})
);
defparam bram_rromx9_0.READ_MODE = 1'b0;
defparam bram_rromx9_0.BIT_WIDTH = 9;
defparam bram_rromx9_0.BLK_SEL = 3'b000;
defparam bram_rromx9_0.RESET_MODE = "SYNC";
defparam bram_rromx9_0.INIT_RAM_00 =
288'hCE08CC85D07DE1316FFE0F86DE1A09523795E0E7E5E71B2
020BC630D6053160EC7FC0000;
defparam bram_rromx9_0.INIT_RAM_01 =
```

```
    288'h000000000000000000000000000000000000000000000000000
    0000000001FFFFFFF7ACF;
```

Vhdl 例化：

```
    COMPONENT rROMX9
        GENERIC(
            BIT_WIDTH:integer:=9;
            READ_MODE:bit:='0';
            BLK_SEL:bit_vector:="000";
            RESET_MODE:string:="SYNC";
            INIT_RAM_00:bit_vector:=X"CE08CC85D07DE131
6FFE0F86DE1A09523795E0E7E5E71B2020BC630D6053160EC7FC000
0";
            INIT_RAM_01:bit_vector:=X"000000000000000000
0000000000000000000000000000000000000001FFFFFFF7ACF"
        );
        PORT(
            DO:OUT std_logic_vector(35 downto 0):=conv_std
_logic_vector(0,36);
            CLK,CE,OCE,RESET:IN std_logic;
            BLKSEL:IN std_logic_vector(2 downto 0);
            AD:IN std_logic_vector(13 downto 0)
        );
    END COMPONENT;
    Uut:rROMX9
        GENERIC MAP(
                BIT_WIDTH=>9,
                READ_MODE=>'0',
                BLK_SEL=>"000",
                RESET_MODE=>"SYNC",
                INIT_RAM_00=>X"CE08CC85D07DE1316F
FE0F86DE1A09523795E0E7E5E71B2020BC630D6053160EC7FC0000",
                INIT_RAM_01=>X"000000000000000000
0000000000000000000000000000000000000001FFFFFFF7ACF"
        )
        PORT MAP(
            DO=>do,
            AD=>ad,
            CLK=>clk,
            CE=>ce,
            OCE=>oce,
            RESET=>reset,
            BLKSEL=>blksel
        );
```

4.9 pROM/pROMX9

原语名称

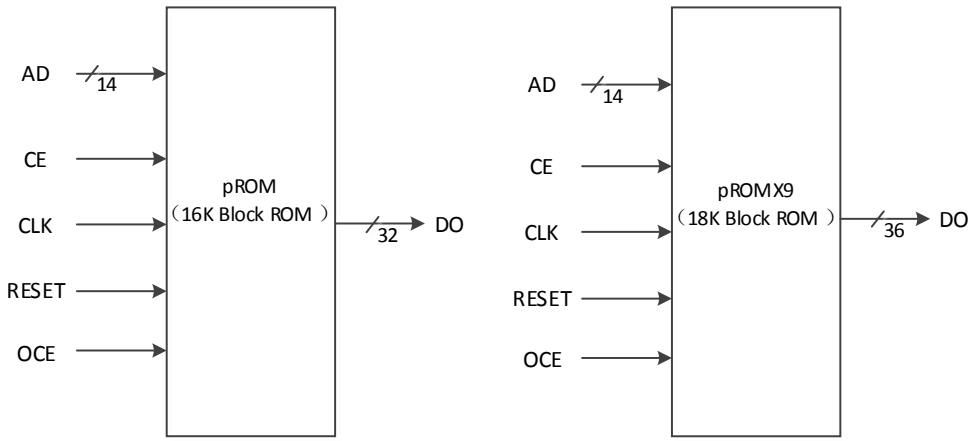
pROM/pROMX9(16K/18K Block ROM),16K/18K 块状只读储存器。

pROM/pROMX9 是 ROM/ROMX9 的优化版，删掉了写使能信号 WRE

和块选择信号 BLKSEL。

端口示意图

图 4-23 pROM/pROMX9 端口示意图



功能描述

pROM/pROMX9 存储空间分别为 16K bit/18K bit，其工作模式为只读模式，可支持 2 种读模式（bypass 模式和 pipeline 模式）。

读模式

通过参数 READ_MODE 来启用或禁用输出 pipeline 寄存器，使用输出 pipeline 寄存器时，读操作需要额外的延迟周期。

端口介绍

表 4-25 端口介绍

端口名	I/O	描述
DO[31:0]/DO[35:0]	Output	数据输出
AD[13:0]	Input	地址输入
CE	Input	时钟使能输入，高电平有效
CLK	Input	时钟输入
RESET	Input	复位输入，支持同步复位和异步复位，高电平有效
OCE	Input	输出时钟使能信号，用于 pipline 模式，对 bypass 模式无效

参数介绍

表 4-26 参数介绍

参数名	参数类型	取值范围	默认值	描述
READ_MODE	Integer	1’b0,1’b1	1’b0	读模式配置 1’b0:bypass 模式 1’b1:pipeline 模式
BIT_WIDTH	Integer	pROM:1,2,4,8,16,32 pROMX9:9,18,36	pROM:32 pROMX9:36	数据宽度配置

参数名	参数类型	取值范围	默认值	描述
RESET_MODE	String	SYNC,ASYNC	SYNC	复位模式配置 SYNC：同步复位 ASYNC：异步复位
INIT_RAM_00~ INIT_RAM_3F	Integer	pROM:256'h0…0~256'h1…1 pROMX9:288'h0…0~288'h1…1	pROM:256'h0…0 pROMX9:288'h0…0	用于设置 B-SRAM 存储单元的初始化数据

配置关系

表 4-27 配置关系

只读模式	BSRAM 容量	数据宽度	地址深度
pROM	16K	1	14
		2	13
		4	12
		8	11
		16	10
		32	9
pROMX9	18K	9	11
		18	10
		36	9

原语例化

示例一

Verilog 例化：

```
pROM bram_prom_0 (
    .DO({dout[31:8],dout[7:0]}),
    .CLK(clk),
    .OCE(oce),
    .CE(ce),
    .RESET(reset),
    .AD({ad[10:0],3'b000})
);
defparam bram_prom_0.READ_MODE = 1'b0;
defparam bram_prom_0.BIT_WIDTH = 8;
defparam bram_prom_0.RESET_MODE = "SYNC";
defparam bram_prom_0.INIT_RAM_00 =
256'h9C23645D0F78986FFC3E36E141541B95C19F2F7164085E631
A819860D8FF0000;
defparam bram_prom_0.INIT_RAM_01 =
256'h0000000000000000000000000000000000000000000000000
000FFFFFFBDCF;
```

Vhdl 例化：

```
COMPONENT pROM
    GENERIC(
```

```
                        BIT_WIDTH:integer:=1;
                        READ_MODE:bit:='0';
                        RESET_MODE:string:="SYNC";
                        INIT_RAM_00:bit_vector:=X"9C23645D0F78986FF
C3E36E141541B95C19F2F7164085E631A819860D8FF0000";
                        INIT_RAM_01:bit_vector:=X"000000000000000000
0000000000000000000000000000000000000FFFFFFBDCF"
            );
            PORT(
                        DO:OUT  std_logic_vector(31  downto  0):=conv_std
_logic_vector(0,32);
                        CLK,CE,OCE,RESET:IN  std_logic;
                        AD:IN  std_logic_vector(13  downto  0)
            );
    END COMPONENT;
    uut:pROM
        GENERIC MAP(
                        BIT_WIDTH=>1,
                        READ_MODE=>'0',
                        RESET_MODE=>"SYNC",
                        INIT_RAM_00=>X"9C23645D0F78986FFC3E36
E141541B95C19F2F7164085E631A819860D8FF0000",
                        INIT_RAM_01=>X"000000000000000000000000
0000000000000000000000000000000000000FFFFFFBDCF  "
        )
         PORT MAP(
                DO=>do,
                AD=>ad,
                CLK=>clk,
                CE=>ce,
                OCE=>oce,
                RESET=>reset
         );
```

示例二

Verilog 例化：

```
    pROMX9 bram_promx9_0 (
        .DO({dout[35:9],dout[8:0]}),
        .CLK(clk),
        .OCE(oce),
        .CE(ce),
        .RESET(reset),
        .AD({ad[10:0],3’b000})
    );
    defparam bram_promx9_0.READ_MODE = 1'b0;
    defparam bram_promx9_0.BIT_WIDTH = 9;
    defparam bram_promx9_0.RESET_MODE = "SYNC";
    defparam bram_promx9_0.INIT_RAM_00 =
    288'hCE08CC85D07DE1316FFE0F86DE1A09523795E0E7E5E71B2
    020BC630D6053160EC7FC0000;
```

```
defparam bram_promx9_0.INIT_RAM_01 =
288'h000000000000000000000000000000000000000000000000000000001FFFFFFF7ACF;
```

Vhdl 例化：

```
COMPONENT pROMX9
        GENERIC(
                BIT_WIDTH:integer:=9;
                READ_MODE:bit:='0';
                RESET_MODE:string:="SYNC";
                INIT_RAM_00:bit_vector:=X"CE08CC85D07DE1316FFE0F86DE1A09523795E0E7E5E71B2020BC630D6053160EC7FC0000";
                INIT_RAM_01:bit_vector:=X"000000000000000000000000000000000000000000000000000000001FFFFFFF7ACF"
            );
        PORT(
                DO:OUT std_logic_vector(35 downto 0):=conv_std_logic_vector(0,36);
                CLK,CE,OCE,RESET:IN std_logic;
                AD:IN std_logic_vector(13 downto 0)
            );
    END COMPONENT;
    Uut:pROMX9
        GENERIC MAP(
                        BIT_WIDTH=>9,
                        READ_MODE=>'0',
                        RESET_MODE=>"SYNC",
                        INIT_RAM_00=>X"CE08CC85D07DE1316FFE0F86DE1A09523795E0E7E5E71B2020BC630D6053160EC7FC0000",
                        INIT_RAM_01=>X"000000000000000000000000000000000000000000000000000000001FFFFFFF7ACF"
            )
        PORT MAP(
                DO=>do,
                AD=>ad,
                CLK=>clk,
                CE=>ce,
                OCE=>oce,
                RESET=>reset
            );
```

5 DSP

DSP(Digital Signal Processing）是数字信号处理，包含预加器（Pre-Adder），乘法器（MULT）和 54 位算术逻辑单元（ALU54D）。

支持器件：GW1N-2、GW1N-2B、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

5.1 Pre-adder

Pre-adder 是预加器，实现预加、预减和移位功能。Pre-adder 按照位宽分为两种，分别是 9 位位宽的 PADD9 和 18 位位宽的 PADD18。

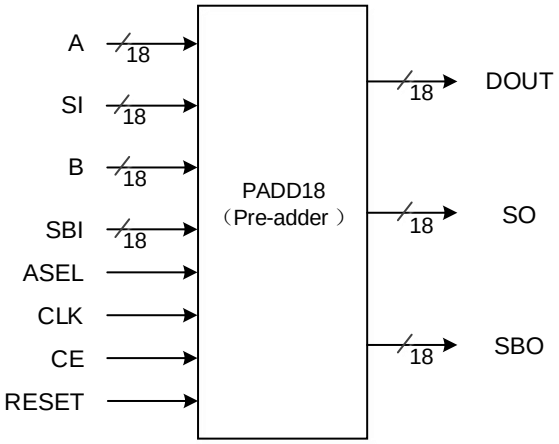
5.1.1 PADD18

原语介绍

PADD18（18-bit Pre-Adder）是 18 位预加器，实现了 18 位的预加、预减或移位功能。

结构框图

图 5-1 结构框图



Port 介绍

表 5-1 Port 介绍

Port Name	I/O	Description
A[17:0]	Input	18-bit Data Input A
B[17:0]	Input	18-bit Data Input B
SI[17:0]	Input	Shift Data Input A
SBI[17:0]	Input	Pre-adder Shift Input, backward direction
ASEL	Input	Source Selection, SI orA
CLK	Input	Clock Input
CE	Input	Clock Enable
RESET	Input	Reset Input
SO[17:0]	Output	Shift Data Output A
SBO[17:0]	Output	Pre-adder Shift Output, backward direction
DOUT[17:0]	Output	Data Output

Attribute 介绍

表 5-2 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
AREG	1'b0,1'b1	1'b0	Input A(A or SI)register can be bypassed 1'b0: bypass mode 1'b1: registered mode
BREG	1'b0,1'b1	1'b0	Input B(B or SBI) register can be bypassed 1'b0: bypass mode 1'b1: registered mode
ADD_SUB	1'b0,1'b1	1'b0	ADD/SUB Selection 1'b0: add 1'b1: sub
PADD_RESET_MODE	SYNC,ASYNC	SYNC	Reset mode config,synchronous or asynchronous
BSEL_MODE	1'b1,1'b0	1'b1	Input B Selection. 1'b1: select SBI 1'b0: select B
SOREG	1'b0,1'b1	1'b0	Shift output register at port SO can be bypassed 1'b0: bypass mode 1'b1: registered mode

原语例化

Verilog 例化：

```
PADD18 padd18_inst(
    .A(a[17:0]),
    .B(b[17:0]),
    .SO(so[17:0]),
    .SBO(sbo[17:0]),
    .DOUT(dout[17:0]),
```

```
        .SI(si[17:0]),
        .SBI(sbi[17:0]),
        .CE(ce),
        .CLK(clk),
        .RESET(reset),
        .ASEL(asel)
    );
    defparam padd18_inst.AREG = 1'b0;
    defparam padd18_inst.BREG = 1'b0;
    defparam padd18_inst.ADD_SUB = 1'b0;
    defparam padd18_inst.PADD_RESET_MODE = "SYNC";
    defparam padd18_inst.SOREG = 1'b0;
    defparam padd18_inst.BSEL_MODE = 1'b1;
```

Vhdl 例化：

```
    COMPONENT PADD18
            GENERIC (AREG:bit:='0';
                        BREG:bit:='0';
                        SOREG:bit:='0';
                        ADD_SUB:bit:='0';
                        PADD_RESET_MODE:string:="SYNC" ;
                        BSEL_MODE:bit:='1'
                );
                PORT(
                        A:IN std_logic_vector(17 downto 0);
                        B:IN std_logic_vector(17 downto 0);
                        ASEL:IN std_logic;
                        CE:IN std_logic;
                        CLK:IN std_logic;
                        RESET:IN std_logic;
                        SI:IN std_logic_vector(17 downto 0);
                        SBI:IN std_logic_vector(17 downto 0);
                        SO:OUT std_logic_vector(17 downto 0);
                        SBO:OUT std_logic_vector(17 downto 0);
                        DOUT:OUT std_logic_vector(17 downto 0)
                );
    END COMPONENT;
    uut:PADD18
            GENERIC MAP (AREG=>'0',
                                BREG=>'0',
                                SOREG=>'0',
                                ADD_SUB=>'0',
                                PADD_RESET_MODE=>"SYNC",
                                BSEL_MODE=>'1'
            )
            PORT MAP (
                A=>a,
                B=>b,
                ASEL=>asel,
                CE=>ce,
                CLK=>clk,
```

```
        RESET=>reset,
        SI=>si,
        SBI=>sbi,
        SO=>so,
        SBO=>sbo,
        DOUT=>dout
    );
```

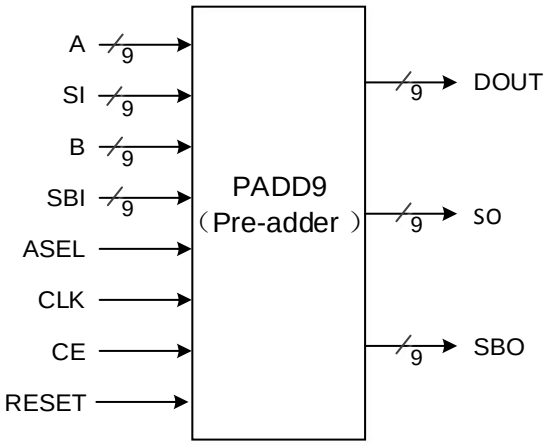
5.1.2 PADD9

原语介绍

PADD9（9-bit Pre-Adder）是 9 位预加器，实现了 9 位的预加、预减或移位功能。

结构框图

图 5-2 PADD9 结构框图



Port 介绍

表 5-3 Port 介绍

Port Name	I/O	Description
A[8:0]	Input	9-bit Data Input A
B[8:0]	Input	9-bit Data Input B
SI[8:0]	Input	Shift Data Input A
SBI[8:0]	Input	Pre-adder Shift Input, backward direction
ASEL	Input	Source Selection, SI or A
CLK	Input	Clock input
CE	Input	Clock Enable
RESET	Input	Reset Input
SO[8:0]	Output	Shift Data Output A
SBO[8:0]	Output	Pre-adder Shift Output, backward direction
DOUT[8:0]	Output	Data Output

Attribute 介绍

表 5-4 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
AREG	1’b0,1’b1	1’b0	Input A(A or SI) register can be bypassed 1’b0: bypass mode 1’b1: registered mode
BREG	1’b0,1’b1	1’b0	Input B(B or SBI) register can be bypassed 1’b0: bypass mode 1’b1: registered mode
ADD_SUB	1’b0,1’b1	1’b0	ADD/SUB Selection 1’b0: add 1’b1: sub
PADD_RESET_MODE	SYNC,ASYNC	SYNC	Reset mode config,synchronous or asynchronous
BSEL_MODE	1’b1,1’b0	1’b1	Input B Selection. 1'b1: select SBI 1'b0: select B
SOREG	1’b0,1’b1	1’b0	Shift output register at port SO can be bypassed 1’b0: bypass mode 1’b1: registered mode

原语例化

Verilog 例化：

```
PADD9 padd9_inst(
        .A(a[8:0]),
        .B(b[8:0]),
        .SO(so[8:0]),
        .SBO(sbo[8:0]),
        .DOUT(dout[8:0]),
        .SI(si[8:0]),
        .SBI(sbi[8:0]),
        .CE(ce),
        .CLK(clk),
        .RESET(reset),
        .ASEL(asel)
    );
    defparam padd9_inst.AREG = 1'b0;
    defparam padd9_inst.BREG = 1'b0;
    defparam padd9_inst.ADD_SUB = 1'b0;
    defparam padd9_inst.PADD_RESET_MODE = "SYNC";
    defparam padd9_inst.SOREG = 1'b0;
    defparam padd9_inst.BSEL_MODE = 1'b1;
```

Vhdl 例化：

```
    COMPONENT PADD9
            GENERIC (AREG:bit:='0';
                        BREG:bit:='0';
                        SOREG:bit:='0';
                        ADD_SUB:bit:='0';
```

```
                        PADD_RESET_MODE:string:="SYNC" ;
                        BSEL_MODE:bit:='1'
                );
                 PORT(
                        A:IN std_logic_vector(8 downto 0);
                        B:IN std_logic_vector(8 downto 0);
                        ASEL:IN std_logic;
                        CE:IN std_logic;
                        CLK:IN std_logic;
                        RESET:IN std_logic;
                        SI:IN std_logic_vector(8 downto 0);
                        SBI:IN std_logic_vector(8 downto 0);
                        SO:OUT std_logic_vector(8 downto 0);
                        SBO:OUT std_logic_vector(8 downto 0);
                        DOUT:OUT std_logic_vector(8 downto 0)
                );
END COMPONENT;
uut:PADD9
        GENERIC MAP (AREG=>'0',
                        BREG=>'0',
                        SOREG=>'0',
                        ADD_SUB=>'0',
                        PADD_RESET_MODE=>"SYNC",
                        BSEL_MODE=>'1'
        )
        PORT MAP (
            A=>a,
            B=>b,
            ASEL=>asel,
            CE=>ce,
            CLK=>clk,
            RESET=>reset,
            SI=>si,
            SBI=>sbi,
            SO=>so,
            SBO=>sbo,
            DOUT=>dout
        );
```

5.2 Multiplier

Multiplier 是 DSP 的乘法器单元，乘法器的乘数输入信号定义为 MDIA 和 MDIB，乘积输出信号定义为 MOUT，可实现乘法运算： $DOUT = A * B$ 。

Multiplier 根据数据位宽可配置成 9x9，18x18，36x36 等乘法器，分别对应原语 MULT9X9，MULT18X18，MULT36X36。

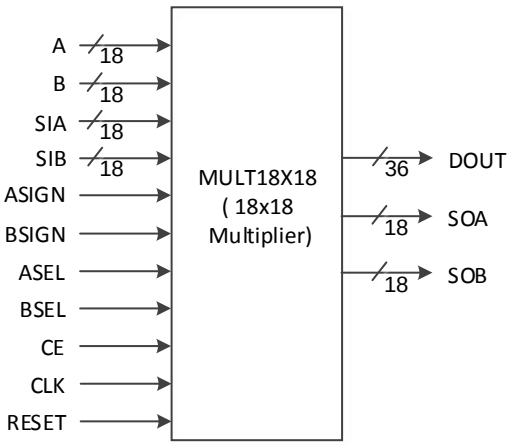
5.2.1 MULT18X18

原语介绍

MULT18X18（18x18 Multiplier）是 18x18 乘法器，实现了 18 位乘法运算。

结构框图

图 5-3 MULT18X18 结构框图



Port 介绍

表 5-5 Port 介绍

Port Name	I/O	Description
A[17:0]	Input	18-bit Data Input A
B[17:0]	Input	18-bit Data Input B
SIA[17:0]	Input	18-bit Shift Data Input A
SIB[17:0]	Input	18-bit Shift Data Input B
ASIGN	Input	Input A Sign Bit
BSIGN	Input	Input B Sign Bit
ASEL	Input	Source Selection, SIA or A
BSEL	Input	Source Selection, SIB or B
CLK	Input	Clock Input
CE	Input	Clock Enable
RESET	Input	Reset Input
DOUT[35:0]	Output	Multiplier Data Output
SOA[17:0]	Output	Multiplier Register Output A
SOB[17:0]	Output	Multiplier Register Output B

Attribute 介绍

表 5-6 Attribute 介绍

Attribute Name	Allowed Values	Default	Description

Attribute Name	Allowed Values	Default	Description
AREG	1’b0,1’b1	1’b0	Input A(SIA or A) register can be bypassed 1’b0:bypass mode 1’b1:registered mode
BREG	1’b0,1’b1	1’b0	Input B(SIB or B) register can be bypassed 1’b0:bypass mode ’b1:registered mode
OUT_REG	1’b0,1’b1	1’b0	Output register can be bypassed 1’b0:bypass mode 1’b1:registered mode
PIPE_REG	1’b0,1’b1	1’b0	Pipeline register can be bypassed 1’b0:bypass mode 1’b1:registered mode
ASIGN_REG	1’b0,1’b1	1’b0	ASIGN input register can be bypassed 1’b0:bypass mode 1’b1:registered mode
BSIGN_REG	1’b0,1’b1	1’b0	BSIGN input register can be bypassed 1’b0:bypass mode 1’b1:registered mode
SOA_REG	1’b0,1’b1	1’b0	SOA register can be bypassed 1’b0:bypass mode 1’b1:registered mode
MULT_RESET_MODE	SYNC,ASYNC	SYNC	Reset mode config, synchronous or asynchronous

原语例化

Verilog 例化：

```
MULT18X18 uut(
    .DOUT(dout[35:0]),
    .SOA(soa[17:0]),
    .SOB(sob[17:0]),
    .A(a[17:0]),
    .B(b[17:0]),
    .SIA(sia[17:0]),
    .SIB(sib[17:0]),
    .ASIGN(asign),
    .BSIGN(bsign),
    .ASEL(asel),
    .BSEL(bsel),
    .CE(ce),
    .CLK(clk),
    .RESET(reset)
  );
defparam uut.AREG=1'b1;
defparam uut.BREG=1'b1;
defparam uut.OUT_REG=1'b1;
defparam uut.PIPE_REG=1'b0;
defparam uut.ASIGN_REG=1'b0;
defparam uut.BSIGN_REG=1'b0;
defparam uut.SOA_REG=1'b0;
defparam uut.MULT_RESET_MODE="ASYNC";
```

Vhdl 例化：

```
COMPONENT MULT18X18
        GENERIC (AREG:bit:='0';
                 BREG:bit:='0';
                 OUT_REG:bit:='0';
                 PIPE_REG:bit:='0';
                 ASIGN_REG:bit:='0';
                 BSIGN_REG:bit:='0';
                 SOA_REG:bit:='0';
                 MULT_RESET_MODE:string:="SYNC"
        );
        PORT(
             A:IN std_logic_vector(17 downto 0);
             B:IN std_logic_vector(17 downto 0);
             SIA:IN std_logic_vector(17 downto 0);
             SIB:IN std_logic_vector(17 downto 0);
             ASIGN:IN std_logic;
             BSIGN:IN std_logic;
             ASEL:IN std_logic;
             BSEL:IN std_logic;
             CE:IN std_logic;
             CLK:IN std_logic;
             RESET:IN std_logic;
             SOA:OUT std_logic_vector(17 downto 0);
             SOB:OUT std_logic_vector(17 downto 0);
             DOUT:OUT std_logic_vector(35 downto 0)
        );
END COMPONENT;
uut:MULT18X18
        GENERIC MAP (AREG=>'1',
                     BREG=>'1',
                     OUT_REG=>'1',
                     PIPE_REG=>'0',
                     ASIGN_REG=>'0',
                     BSIGN_REG=>'0',
                     SOA_REG=>'0',
                     MULT_RESET_MODE=>"ASYNC"
        )
        PORT MAP (
            A=>a,
            B=>b,
            SIA=>sia,
            SIB=>sib,
            ASIGN=>asign,
            BSIGN=>bsign,
            ASEL=>asel,
            BSEL=>bsel,
            CE=>ce,
            CLK=>clk,
            RESET=>reset,
```

```
        SOA=>soa,
        SOB=>sob,
        DOUT=>dout
    );
```

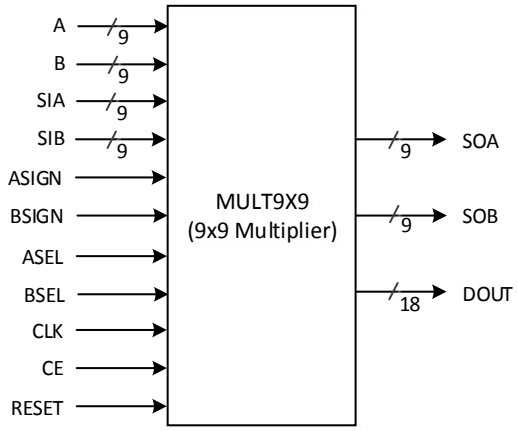
5.2.2 MULT9X9

原语介绍

MULT9X9（9x9 Multiplier）是 9x9 乘法器，实现了 9 位乘法运算。

结构框图

图 5-4 MULT9X9 结构框图



Port 介绍

表 5-7 Port 介绍

Port Name	I/O	Description
A[8:0]	Input	9-bit Data Input A
B[8:0]	Input	9-bit Data Input B
SIA[8:0]	Input	9-bit Shift Data Input A
SIB[8:0]	Input	9-bit Shift Data Input B
ASIGN	Input	Input A Sign bit
BSIGN	Input	Input B Sign bit
ASEL	Input	Source Selection, SIA or A
BSEL	Input	Source Selection, SIB or B
CLK	Input	Clock Input
CE	Input	Clock Enable
RESET	Input	Reset Input
DOUT[17:0]	Output	Multiplier Data Output
SOA[8:0]	Output	Multiplier Register Output A
SOB[8:0]	Output	Multiplier Register Output B

Attribute 介绍

表 5-8 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
AREG	1’b0,1’b1	1’b0	Input A(SIA or A) register can be bypassed 1’b0:bypass mode 1’b1:registered mode
BREG	1’b0,1’b1	1’b0	Input B(SIB or B) register can be bypassed 1’b0:bypass mode 1’b1:registered mode
OUT_REG	1’b0,1’b1	1’b0	Output register can be bypassed 1’b0:bypass mode 1’b1:registered mode
PIPE_REG	1’b0,1’b1	1’b0	Pipeline register can be bypassed 1’b0:bypass mode 1’b1:registered mode
ASIGN_REG	1’b0,1’b1	1’b0	ASIGN input register can be bypassed 1’b0:bypass mode 1’b1:registered mode
BSIGN_REG	1’b0,1’b1	1’b0	BSIGN input register can be bypassed 1’b0:bypass mode 1’b1:registered mode
SOA_REG	1’b0,1’b1	1’b0	SOA register can be bypassed 1’b0:bypass mode 1’b1:registered mode
MULT_RESET_MODE	SYNC，ASYNC	SYNC	Reset mode config, synchronous or asynchronous

原语例化

Verilog 例化：

```
MULT9X9 uut(
    .DOUT(dout[17:0]),
    .SOA(soa[8:0]),
    .SOB(sob[8:0]),
    .A(a[8:0]),
    .B(b[8:0]),
    .SIA(sia[8:0]),
    .SIB(sib[8:0]),
    .ASIGN(asign),
    .BSIGN(bsign),
    .ASEL(asel),
    .BSEL(bsel),
    .CE(ce),
    .CLK(clk),
    .RESET(reset)
  );
defparam uut.AREG=1'b1;
defparam uut.BREG=1'b1;
defparam uut.OUT_REG=1'b1;
defparam uut.PIPE_REG=1'b0;
defparam uut.ASIGN_REG=1'b0;
```

```
defparam uut.BSIGN_REG=1'b0;
defparam uut.SOA_REG=1'b0;
defparam uut.MULT_RESET_MODE="ASYNC";
```

Vhdl 例化：

```
COMPONENT MULT9X9
        GENERIC (AREG:bit:='0';
                 BREG:bit:='0';
                 OUT_REG:bit:='0';
                 PIPE_REG:bit:='0';
                 ASIGN_REG:bit:='0';
                 BSIGN_REG:bit:='0';
                 SOA_REG:bit:='0';
                 MULT_RESET_MODE:string:="SYNC"
        );
        PORT(
             A:IN std_logic_vector(8 downto 0);
             B:IN std_logic_vector(8 downto 0);
             SIA:IN std_logic_vector(8 downto 0);
             SIB:IN std_logic_vector(8 downto 0);
             ASIGN:IN std_logic;
             BSIGN:IN std_logic;
             ASEL:IN std_logic;
             BSEL:IN std_logic;
             CE:IN std_logic;
             CLK:IN std_logic;
             RESET:IN std_logic;
             SOA:OUT std_logic_vector(8 downto 0);
             SOB:OUT std_logic_vector(8 downto 0);
             DOUT:OUT std_logic_vector(17 downto 0)
        );
END COMPONENT;
uut:MULT9X9
        GENERIC MAP (AREG=>'1',
                     BREG=>'1',
                     OUT_REG=>'1',
                     PIPE_REG=>'0',
                     ASIGN_REG=>'0',
                     BSIGN_REG=>'0',
                     SOA_REG=>'0',
                     MULT_RESET_MODE=>"ASYNC"
        )
        PORT MAP (
            A=>a,
            B=>b,
            SIA=>sia,
            SIB=>sib,
            ASIGN=>asign,
            BSIGN=>bsign,
            ASEL=>asel,
            BSEL=>bsel,
```

```
        CE=>ce,
        CLK=>clk,
        RESET=>reset,
        SOA=>soa,
        SOB=>sob,
        DOUT=>dout
    );
```

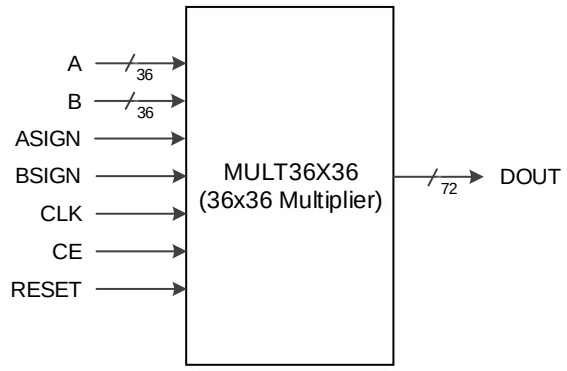
5.2.3 MULT36X36

原语介绍

MULT36X36（36x36 Multiplier）是 36x36 乘法器，实现了 36 位的乘法运算。

结构框图

图 5-5 MULT36X36 结构框图



Port 介绍

表 5-9 Port 介绍

Port Name	I/O	Description
A[35:0]	Input	36-bit Data Input A
B[35:0]	Input	36-bit Data Input B
ASIGN	Input	Input A Sign bit
BSIGN	Input	Input B Sign bit
CLK	Input	Clock Input
CE	Input	Clock Enable
RESET	Input	Reset Input
DOUT[71:0]	Output	Multiplier Data Output

Attribute 介绍

表 5-10 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
AREG	1’b0,1’b1	1’b0	Input A(A) register can be bypassed. 1’b0:bypass mode 1’b1:registered mode

Attribute Name	Allowed Values	Default	Description
BREG	1’b0,1’b1	1’b0	Input B(B) register can be bypassed. 1’b0:bypass mode 1’b1:registered mode
OUT0_REG	1’b0,1’b1	1’b0	Thefirst outputregister can be bypassed 1’b0:bypass mode 1’b1:registered mode
OUT1_REG	1’b0,1’b1	1’b0	The second output register can be bypassed 1’b0:bypass mode 1’b1:registered mode
PIPE_REG	1’b0,1’b1	1’b0	Pipeline register can be bypassed 1’b0:bypass mode 1’b1:registered mode
ASIGN_REG	1’b0,1’b1	1’b0	ASIGN input register can be bypassed 1’b0:bypass mode 1’b1:registered mode
BSIGN_REG	1’b0,1’b1	1’b0	BSIGN input register can be bypassed 1’b0:bypass mode 1’b1:registered mode
MULT_RESET_MODE	SYNC,ASYNC	SYNC	Reset mode config,synchronous or asynchronous

原语例化

Verilog 例化：

```
MULT36X36 uut(
    .DOUT(mout[71:0]),
    .A(mdia[35:0]),
    .B(mdib[35:0]),
    .ASIGN(asign),
    .BSIGN(bsign),
    .CE(ce),
    .CLK(clk),
    .RESET(reset)
  );
defparam uut.AREG=1'b0;
defparam uut.BREG=1'b0;
defparam uut.OUT0_REG=1'b0;
defparam uut.OUT1_REG=1'b1;
defparam uut.PIPE_REG=1'b0;
defparam uut.ASIGN_REG=1'b1;
defparam uut.BSIGN_REG=1'b1;
defparam uut.MULT_RESET_MODE="ASYNC";
```

Vhdl 例化：

```
COMPONENT MULT36X36
        GENERIC (AREG:bit:='0';
                 BREG:bit:='0';
                 OUT0_REG:bit:='0';
                 OUT1_REG:bit:='0';
                 PIPE_REG:bit:='0';
                 ASIGN_REG:bit:='0';
                 BSIGN_REG:bit:='0';
```

```
                        MULT_RESET_MODE:string:="SYNC"
        );
        PORT(
                A:IN std_logic_vector(35 downto 0);
                B:IN std_logic_vector(35 downto 0);
                ASIGN:IN std_logic;
                BSIGN:IN std_logic;
                CE:IN std_logic;
                CLK:IN std_logic;
                RESET:IN std_logic;
                DOUT:OUT std_logic_vector(71 downto 0)
        );
END COMPONENT;
uut:MULT36X36
        GENERIC MAP (AREG=>'0',
                        BREG=>'0',
                        OUT0_REG=>'0',
                        OUT1_REG=>'1',
                        PIPE_REG=>'0',
                        ASIGN_REG=>'1',
                        BSIGN_REG=>'1',
                        MULT_RESET_MODE=>"ASYNC"
        )
        PORT MAP (
            A=>mdia,
            B=>mdib,
            ASIGN=>asign,
            BSIGN=>bsign,
            CE=>ce,
            CLK=>clk,
            RESET=>reset,
            DOUT=>mout
        );
```

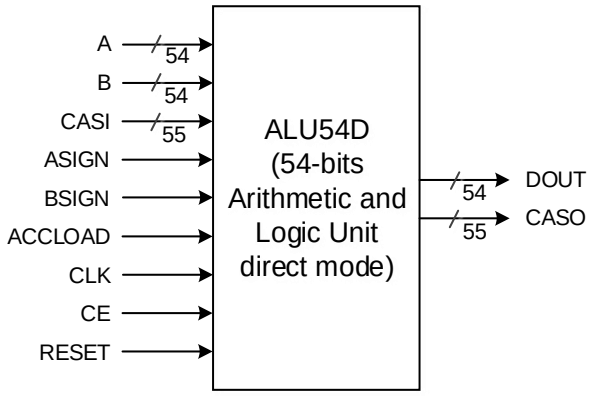
5.3 ALU54D

原语介绍

ALU54D（54-bit Arithmetic Logic Unit）是 54 位算术逻辑单元，实现 54 位的算术逻辑运算。

结构框图

图 5-6 ALU54D 结构框图



Port 介绍

表 5-11 Port 介绍

Port Name	I/O	Description
A[53:0]	Input	54-bit Data Input A
B[53:0]	Input	54-bit Data Input B
CASI[54:0]	Input	55-bit Data Carry Input
ASIGN	Input	Input A Sign Bit
BSIGN	Input	Input B Sign Bit
ACCLOAD	Input	Accumulator Reload Mode Selection
CLK	Input	Clock Input
CE	Input	Clock Enable
RESET	Input	Reset Input
DOUT[53:0]	Output	ALU54D Data Output
CASO[54:0]	Output	55-bit Data Carry Output

Attribute 介绍

表 5-12 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
AREG	1’b0,1’b1	1’b0	Input A(A) registers can be bypassed 1’b0:bypass mode 1’b1: registered mode
BREG	1’b0,1’b1	1’b0	Input B(B) registers can be bypassed 1’b0:bypass mode 1’b1: registered mode
ASIGN_REG	1’b0,1’b1	1’b0	ASIGN input register can be bypassed 1’b0:bypass mode 1’b1:registered mode
BSIGN_REG	1’b0,1’b1	1’b0	BSIGN input register can be bypassed 1’b0:bypass mode 1’b1:registered mode
ACCLOAD_REG	1’b0,1’b1	1’b0	Stage register of

Attribute Name	Allowed Values	Default	Description
			ACCLOAD can be bypassed 1’b0:bypass mode 1’b1:registered mode
OUT_REG	1’b0,1’b1	1’b0	The output registers can be bypassed. 1’b0:bypass mode 1’b1: registered mode
B_ADD_SUB	1’b0,1’b1	1’b0	B_OUT ADD/SUB Selection 1’b0: add 1’b1: sub
C_ADD_SUB	1’b0,1’b1	1’b0	C_OUT ADD/SUB Selection 1’b0: add 1’b1: sub
ALUMODE	0,1,2	0	ALU54 Operation Mode and Unit Input Selection 0:ACC/0 +/- B +/- A; 1:ACC/0 +/- B + CASI; 2:A +/- B + CASI;
ALU_RESET_MODE	SYNC,ASYNC	SYNC	Reset mode config, synchronous or asynchronous

原语例化

Verilog 例化：

```
ALU54D alu54_inst (
    .A(a[53:0]),
    .B(b[53:0]),
    .CASI(casi[54:0]),
    .ASIGN(asign),
    .BSIGN(bsign),
    .ACCLOAD(accload),
    .CE(ce),
    .CLK(clk),
    .RESET(reset),
    .DOUT(dout[53:0]),
    .CASO(caso[54:0])
);
defparam alu54_inst.AREG=1'b1;
defparam alu54_inst.BREG=1'b1;
defparam alu54_inst.ASIGN_REG=1'b0;
defparam alu54_inst.BSIGN_REG=1'b0;
defparam alu54_inst.ACCLOAD_REG=1'b1;
defparam alu54_inst.OUT_REG=1'b0;
defparam alu54_inst.B_ADD_SUB=1'b0;
defparam alu54_inst.C_ADD_SUB=1'b0;
defparam alu54_inst.ALUMODE=0;
defparam alu54_inst.ALU_RESET_MODE="SYNC";
```

Vhdl 例化：

```
COMPONENT ALU54D
        GENERIC (AREG:bit:='0';
                 BREG:bit:='0';
```

```
                        ASIGN_REG:bit:='0';
                        BSIGN_REG:bit:='0';
                        ACCLOAD_REG:bit:='0';
                        OUT_REG:bit:='0';
                        B_ADD_SUB:bit:='0';
                        C_ADD_SUB:bit:='0';
                        ALUD_MODE:integer:=0;
                        ALU_RESET_MODE:string:="SYNC"
                );
                PORT(
                        A:IN std_logic_vector(53 downto 0);
                        B:IN std_logic_vector(53 downto 0);
                        ASIGN:IN std_logic;
                        BSIGN:IN std_logic;
                        CE:IN std_logic;
                        CLK:IN std_logic;
                        RESET:IN std_logic;
                        ACCLOAD:IN std_logic;
                        CASI:IN std_logic_vector(54 downto 0);
                        CASO:OUT std_logic_vector(54 downto 0);
                        DOUT:OUT std_logic_vector(53 downto 0)
                );
END COMPONENT;
uut:ALU54D
        GENERIC MAP (AREG=>'1',
                        BREG=>'1',
                        ASIGN_REG=>'0',
                        BSIGN_REG=>'0',
                        ACCLOAD_REG=>'1',
                        OUT_REG=>'0',
                        B_ADD_SUB=>'0',
                        C_ADD_SUB=>'0',
                        ALUD_MODE=>0,
                        ALU_RESET_MODE=>"SYNC"
        )
        PORT MAP (
            A=>a,
            B=>b,
            ASIGN=>asign,
            BSIGN=>bsign,
            CE=>ce,
            CLK=>clk,
            RESET=>reset,
            ACCLOAD=>accload,
            CASI=>casi,
            CASO=>caso,
            DOUT=>dout
        );
```

5.4 MULTALU

MULTALU（Multiplier with ALU）是带 ALU 功能的乘法器，分为 36X18 位和 18X18 位，分别对应原语 MULTALU36X18 和 MULTALU18X18。

MULTALU36X18 有三种运算模式：

$$DOUT = A*B \pm C$$

$$DOUT = \sum(A*B)$$

$$DOUT = A*B + CASI$$

MULTALU18X18 有三种运算模式：

$$DOUT = \sum(A*B) \pm C$$

$$DOUT = \sum(A*B) + CASI$$

$$DOUT = A*B \pm D + CASI$$

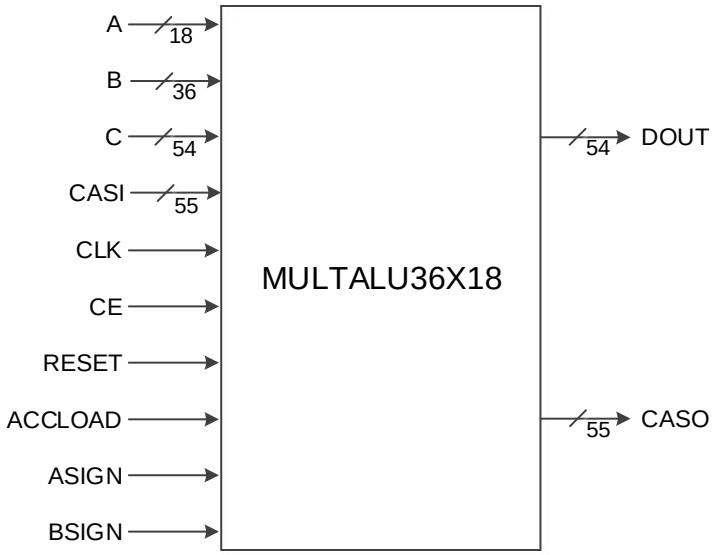
5.4.1 MULTALU36X18

原语介绍

MULTALU36X18（36x18 Multiplier with ALU）是带 ALU 功能的 36X18 乘法器。

结构框图

图 5-7 MULTALU36X18 结构框图



Port 介绍

表 5-13 Port 介绍

Port Name	I/O	Description
A[17:0]	Input	18-bit Data Input A
B[35:0]	Input	36-bit Data Input B
C[53:0]	Input	54-bit Reload Data Input
CASI[54:0]	Input	55-bit Data Carry Input

Port Name	I/O	Description
ASIGN	Input	Input A Sign Bit
BSIGN	Input	Input B Sign Bit
CLK	Input	Clock Input
CE	Input	Clock Enable
RESET	Input	Reset Input
ACCLOAD	Input	Accumulator Reload Mode Selection
DOUT[53:0]	Output	Data Output
CASO[54:0]	Output	55-bit Data Carry Output

Attribute 介绍

表 5-14 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
AREG	1’b0,1’b1	1’b0	Input A(A)register can be bypassed 1’b0:bypass mode 1’b1:registered mode
BREG	1’b0,1’b1	1’b0	Input B(B)register can be bypassed 1’b0:bypass mode 1’b1:registered mode
CREG	1’b0,1’b1	1’b0	Input C(C) register can be bypassed 1’b0:bypass mode 1’b1:registered mode
OUT_REG	1’b0,1’b1	1’b0	The output registers can be bypassed. 1’b0:bypass mode 1’b1: registered mode
PIPE_REG	1’b0,1’b1	1’b0	Pipeline register can be bypassed . 1’b0:bypass mode 1’b1:registered mode
ASIGN_REG	1’b0,1’b1	1’b0	ASIGN input register can be bypassed 1’b0:bypass mode 1’b1:registered mode
BSIGN_REG	1’b0,1’b1	1’b0	BSIGN input register can be bypassed. 1’b0:bypass mode 1’b1:registered mode
ACCLOAD_REG0	1’b0,1’b1	1’b0	The first stage register of ACCLOAD can be bypassed 1’b0:bypass mode 1’b1:registered mode
ACCLOAD_REG1	1’b0,1’b1	1’b0	The second stage register of ACCLOAD can be bypassed 1’b0:bypass mode 1’b1:registered mode
MULT_RESET_MODE	SYNC,ASYNC	SYNC	Reset mode config,synchronous or asynchronous
MULTALU36X18_MODE	0,1,2	0	MULTALU36X18 Operation Mode and Unit Input Selection 0:36x18 +/- C; 1:ACC/0 + 36x18; 2: 36x18 + CASI
C_ADD_SUB	1’b0,1’b1		C_OUT ADD/SUB Selection

Attribute Name	Allowed Values	Default	Description
		1’b0	1’b0: add 1’b1: sub

原语例化

Verilog 例化：

```
MULTALU36X18 multalu36x18_inst(
    .CASO(caso[54:0]),
    .DOUT(dout[53:0]),
    .ASIGN(asign),
    .BSIGN(bsign),
    .CE(ce),
    .CLK(clk),
    .RESET(reset),
    .CASI(casi[54:0]),
    .ACCLOAD(accload),
    .A(a[17:0]),
    .B(b[35:0]),
    .C(c[53:0])
);
 defparam multalu36x18_inst.AREG = 1'b1;
 defparam multalu36x18_inst.BREG = 1'b0;
 defparam multalu36x18_inst.CREG = 1'b0;
 defparam multalu36x18_inst.OUT_REG = 1'b1;
 defparam multalu36x18_inst.PIPE_REG = 1'b0;
 defparam multalu36x18_inst.ASIGN_REG = 1'b0;
 defparam multalu36x18_inst.BSIGN_REG = 1'b0;
 defparam multalu36x18_inst.ACCLOAD_REG0 = 1'b1;
 defparam multalu36x18_inst.ACCLOAD_REG1 = 1'b0;
 defparam multalu36x18_inst.SOA_REG = 1'b0;
 defparam multalu36x18_inst.MULT_RESET_MODE = "SYNC";
 defparam multalu36x18_inst.MULTALU36X18_MODE = 0;
 defparam multalu36x18_inst.C_ADD_SUB = 1'b0;
```

Vhdl 例化：

```
 COMPONENT MULTALU36X18
        GENERIC (AREG:bit:='0';
                 BREG:bit:='0';
                 CREG:bit:='0';
                 OUT_REG:bit:='0';
                 PIPE_REG:bit:='0';
                 ASIGN_REG:bit:='0';
                 BSIGN_REG:bit:='0';
                 ACCLOAD_REG0:bit:='0';
                 ACCLOAD_REG1:bit:='0';
                 SOA_REG:bit:='0';
                 MULTALU36X18_MODE:integer:=0;
                 C_ADD_SUB:bit:='0';
                 MULT_RESET_MODE:string:="SYNC"
```

```
        );
        PORT(
                A:IN std_logic_vector(17 downto 0);
                B:IN std_logic_vector(35 downto 0);
                C:IN std_logic_vector(53 downto 0);
                ASIGN:IN std_logic;
                BSIGN:IN std_logic;
                CE:IN std_logic;
                CLK:IN std_logic;
                RESET:IN std_logic;
                ACCLOAD:IN std_logic;
                CASI:IN std_logic_vector(54 downto 0);
                CASO:OUT std_logic_vector(54 downto 0);
                DOUT:OUT std_logic_vector(53 downto 0)
        );
END COMPONENT;
uut:MULTALU36X18
        GENERIC MAP (AREG=>'1',
                        BREG=>'0',
                        CREG=>'0',
                        OUT_REG=>'1',
                        PIPE_REG=>'0',
                        ASIGN_REG=>'0',
                        BSIGN_REG=>'0',
                        ACCLOAD_REG0=>'1',
                        ACCLOAD_REG1=>'0',
                        SOA_REG=>'0',
                        MULTALU36X18_MODE=>0,
                        C_ADD_SUB=>'0',
                        MULT_RESET_MODE=>"SYNC"
        )
        PORT MAP (
            A=>a,
            B=>b,
            C=>c,
            ASIGN=>asign,
            BSIGN=>bsign,
            CE=>ce,
            CLK=>clk,
            RESET=>reset,
            ACCLOAD=>accload,
            CASI=>casi,
            CASO=>caso,
            DOUT=>dout
    );
```

5.4.2 MULTALU18X18

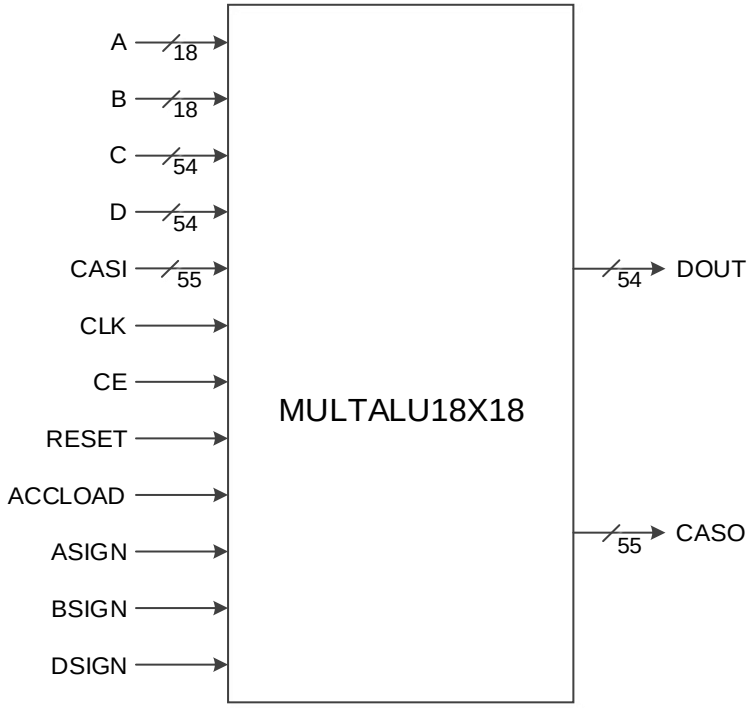
原语介绍

MULTALU18X18（18x18 Multiplier with ALU）是带 ALU 功能的 18x18

乘法器。

结构框图

图 5-8 MULTALU18X18 结构框图



Port 介绍

表 5-15 Port 介绍

Port Name	I/O	Description
A[17:0]	Input	18-bit Data Input A
B[17:0]	Input	18-bit Data Input B
C[53:0]	Input	54-bit Data Input C
D[53:0]	Input	54-bit Data Input D
CASI[54:0]	Input	55-bit Data Carry Input
ASIGN	Input	Input A Sign Bit
BSIGN	Input	Input B Sign Bit
DSIGN	Input	Input D Sign Bit
CLK	Input	Clock Input
CE	Input	Clock Enable
RESET	Input	Reset Input
ACCLOAD	Input	Accumulator Reload Mode selection
DOUT[53:0]	Output	Data Output
CASO[54:0]	Output	55-bit Data Carry Output

Attribute 介绍

表 5-16 Attribute 介绍

Attribute Name	Allowed Values	Default Value	Description
AREG	1’b0,1’b1	1’b0	Input A(A)register can be bypassed 1’b0:bypass mode 1’b1:registered mode
BREG	1’b0,1’b1	1’b0	Input B(B)register can be bypassed 1’b0:bypass mode 1’b1:registered mode
CREG	1’b0,1’b1	1’b0	Input C(C) register can be bypassed 1’b0:bypass mode 1’b1:registered mode
DREG	1’b0,1’b1	1’b0	Input D(D) register can be bypassed 1’b0:bypass mode 1’b1:registered mode
DSIGN_REG	1’b0,1’b1	1’b0	DSIGN input register can be bypassed 1’b0:bypass mode 1’b1:registered mode
ASIGN_REG	1’b0,1’b1	1’b0	ASIGN input register can be bypassed 1’b0:bypass mode 1’b1:registered mode
BSIGN_REG	1’b0,1’b1	1’b0	BSIGN input register can be bypassed. 1’b0:bypass mode 1’b1:registered mode
ACCLOAD_REG0	1’b0,1’b1	1’b0	The first stage register of ACCLOAD can be bypassed 1’b0:bypass mode 1’b1:registered mode
ACCLOAD_REG1	1’b0,1’b1	1’b0	The second stage register of ACCLOAD can be bypassed 1’b0:bypass mode 1’b1:registered mode
MULT_RESET_MODE	SYNC,ASYNC	SYNC	Reset mode config,synchronous or asynchronous
PIPE_REG	1’b0,1’b1	1’b0	Pipeline register can be bypassed . 1’b0:bypass mode 1’b1:registered mode
OUT_REG	1’b0,1’b1	1’b0	The output registers can be bypassed. 1’b0:bypass mode 1’b1: registered mode
B_ADD_SUB	1’b0,1’b1	1’b0	B_OUT ADD/SUB Selection 1’b0: add 1’b1: sub
C_ADD_SUB	1’b0,1’b1	1’b0	C_OUT ADD/SUB Selection 1’b0: add 1’b1: sub
MULTALU18X18_MODE	0,1,2	0	MULTALU36X18 Operation Mode and Unit Input Selection 0:ACC/0 +/- 18x18 +/- C; 1:ACC/0 +/- 18x18 + CASI; 2: 18x18 +/- D + CASI;

原语例化

Verilog 例化：

```
MULTALU18X18 multalu18x18_inst(
    .CASO(caso[54:0]),
    .DOUT(dout[53:0]),
    .ASIGN(asign),
    .BSIGN(bsign),
    .DSIGN(dsign),
    .CE(ce),
    .CLK(clk),
    .RESET(reset),
    .CASI(casi[54:0]),
    .ACCLOAD(accload),
    .A(a[17:0]),
    .B(b[17:0]),
    .C(c[53:0])
    .D(d[53:0])
);
 defparam multalu18x18_inst.AREG = 1'b1;
 defparam multalu18x18_inst.BREG = 1'b0;
 defparam multalu18x18_inst.CREG = 1'b0;
 defparam multalu18x18_inst.DREG = 1'b0;
 defparam multalu18x18_inst.OUT_REG = 1'b1;
 defparam multalu18x18_inst.PIPE_REG = 1'b0;
 defparam multalu18x18_inst.ASIGN_REG = 1'b0;
 defparam multalu18x18_inst.BSIGN_REG = 1'b0;
 defparam multalu18x18_inst.DSIGN_REG = 1'b0;
 defparam multalu18x18_inst.ACCLOAD_REG0 = 1'b1;
 defparam multalu18x18_inst.ACCLOAD_REG1 = 1'b0;
 defparam multalu18x18_inst.MULT_RESET_MODE = "SYNC";
 defparam multalu18x18_inst.MULTALU18X18_MODE = 0;
 defparam multalu18x18_inst.B_ADD_SUB = 1'b0;
 defparam multalu18x18_inst.C_ADD_SUB = 1'b0;
```

Vhdl 例化：

```
 COMPONENT MULTALU18X18
        GENERIC (AREG:bit:='0';
                 BREG:bit:='0';
                 CREG:bit:='0';
                 DREG:bit:='0';
                 OUT_REG:bit:='0';
                 PIPE_REG:bit:='0';
                 ASIGN_REG:bit:='0';
                 BSIGN_REG:bit:='0';
                 DSIGN_REG:bit:='0';
                 ACCLOAD_REG0:bit:='0';
                 ACCLOAD_REG1:bit:='0';
                 B_ADD_SUB:bit:='0';
                 C_ADD_SUB:bit:='0';
                 MULTALU18X18_MODE:integer:=0;
                 MULT_RESET_MODE:string:="SYNC"
        );
        PORT(
```

```
                    A:IN std_logic_vector(17 downto 0);
                    B:IN std_logic_vector(17 downto 0);
                    C:IN std_logic_vector(53 downto 0);
                    D:IN std_logic_vector(53 downto 0);
                    ASIGN:IN std_logic;
                    BSIGN:IN std_logic;
                    DSIGN:IN std_logic;
                    CE:IN std_logic;
                    CLK:IN std_logic;
                    RESET:IN std_logic;
                    ACCLOAD:IN std_logic;
                    CASI:IN std_logic_vector(54 downto 0);
                    CASO:OUT std_logic_vector(54 downto 0);
                    DOUT:OUT std_logic_vector(53 downto 0)
                );
END COMPONENT;
uut:MULTALU18X18
        GENERIC MAP (AREG=>'1',
                            BREG=>'0',
                            CREG=>'0',
                            DREG=>'0',
                            OUT_REG=>'1',
                            PIPE_REG=>'0',
                            ASIGN_REG=>'0',
                            BSIGN_REG=>'0',
                            DSIGN_REG=>'0',
                            ACCLOAD_REG0=>'1',
                            ACCLOAD_REG1=>'0',
                            B_ADD_SUB=>'0',
                            C_ADD_SUB=>'0',
                            MULTALU18X18_MODE=>0,
                            MULT_RESET_MODE=>"SYNC"
        )
        PORT MAP (
            A=>a,
            B=>b,
            C=>c,
            D=>d,
            ASIGN=>asign,
            BSIGN=>bsign,
            DSIGN=>dsign,
            CE=>ce,
            CLK=>clk,
            RESET=>reset,
            ACCLOAD=>accload,
            CASI=>casi,
            CASO=>caso,
            DOUT=>dout
    );
```

5.5 MULTADDALU

MULTADDALU（The Sum of Two Multipliers with ALU）是带 ALU 功能的乘加器，实现乘法求和后累加或 reload 运算，对应的原语为 MULTADDALU18X18。

三种运算模式如下：

$$DOUT = A0 * B0 \pm A1 * B1 \pm C$$

$$DOUT = \sum (A0 * B0 \pm A1 * B1)$$

$$DOUT = A0 * B0 \pm A1 * B1 + CASI$$

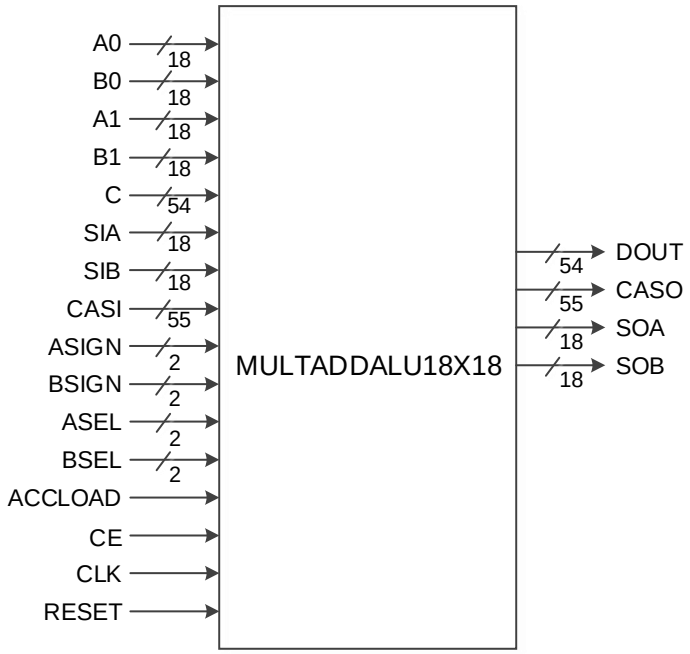
5.5.1 MULTADDALU18X18

原语介绍

MULTADDALU18X18（The Sum of Two 18x18 Multipliers with ALU）是带 ALU 功能的 18x18 乘加器，实现 18 位的乘法求和后累加或 reload 运算。

结构框图

图 5-9 MULTADDALU18X18 结构框图



Port 介绍

表 5-17 Port 介绍

Port Name	I/O	Description
A0[17:0]	Input	18-bit Data Input A0
B0[17:0]	Input	18-bit Data Input B0
A1[17:0]	Input	18-bit Data Input A1
B1[17:0]	Input	18-bit Data Input B1
C[53:0]	Input	54-bit Reload Data Input
SIA[17:0]	Input	18-bit Shift Data Input A

Port Name	I/O	Description
SIB[17:0]	Input	18-bit Shift Data Input B
CASI[54:0]	Input	55-bit Data Carry Input
ASIGN[1:0]	Input	InputA0,A1 Sign bit
BSIGN[1:0]	Input	Input B0,B1 Sign bit
ASEL[1:0]	Input	Input A0,A1 Source Selection
BSEL[1:0]	Input	Input B0,B1 Source Selection
CLK	Input	Clock Input
CE	Input	Clock Enable
RESET	Input	Reset Input
ACCLOAD	Input	Accumulator Reload Mode Selection
DOUT[53:0]	Output	Data Output
CASO[54:0]	Output	55-bit Data Carry Output
SOA[17:0]	Output	Multiplier Register Output A
SOB[17:0]	Output	Multiplier Register Output B

Attribute 介绍

表 5-18 Attribute 介绍

Attribute Name	Allowed Values	Default	Description
A0REG	1’b0,1’b1	1’b0	Input A0(A0 or SIA) register can be bypassed. 1’b0:bypass mode 1’b1:registered mode
A1REG	1’b0,1’b1	1’b0	Input A1(A1 or Register Output A0) register can be bypassed. 1’b0:bypass mode 1’b1:registered mode
B0REG	1’b0,1’b1	1’b0	Input B0(B0 or SIB) register can be bypassed. 1’b0:bypass mode 1’b1:registered mode
B1REG	1’b0,1’b1	1’b0	Input B1(B1 or Register Output B0) register can be bypassed. 1’b0:bypass mode 1’b1:registered mode
CREG	1’b0,1’b1	1’b0	Input C(C) register can be bypassed 1’b0:bypass mode 1’b1:registered mode
PIPE0_REG	1’b0,1’b1	1’b0	Multiplier0 Pipeline register can be bypassed. 1’b0:bypass mode 1’b1:registered mode
PIPE1_REG	1’b0,1’b1	1’b0	Multiplier1 Pipeline register can be bypassed. 1’b0:bypass mode 1’b1:registered mode
OUT_REG	1’b0,1’b1	1’b0	Output register can be bypassed 1’b0:bypass mode

Attribute Name	Allowed Values	Default	Description
			1’b1:registered mode
ASIGN0_REG	1’b0,1’b1	1’b0	ASIGN[0] input register can be bypassed. 1’b0:bypass mode 1’b1:registered mode
ASIGN1_REG	1’b0,1’b1	1’b0	ASIGN[1] input register can be bypassed. 1’b0:bypass mode 1’b1:registered mode
ACCLOAD_REG0	1’b0,1’b1	1’b0	The first stage register of ACCLOAD can be bypassed 1’b0:bypass mode 1’b1:registered mode
ACCLOAD_REG1	1’b0,1’b1	1’b0	The second stage register of ACCLOAD can be bypassed 1’b0:bypass mode 1’b1:registered mode
BSIGN0_REG	1’b0,1’b1	1’b0	BSIGN[0] input register can be bypassed. 1’b0:bypass mode 1’b1:registered mode
BSIGN1_REG	1’b0,1’b1	1’b0	BSIGN[1] input register can be bypassed. 1’b0:bypass mode 1’b1:registered mode
SOA_REG	1’b0,1’b1	1’b0	SOA register can be bypassed. 1’b0:bypassmode 1’b1:registered mode
B_ADD_SUB	1’b0,1’b1	1’b0	B_OUT ADD/SUB Selection 1’b0: add 1’b1: sub
C_ADD_SUB	1’b0,1’b1	1’b0	C_OUT ADD/SUB Selection 1’b0: add 1’b1: sub
MULTADDALU18X18_MODE	0,1,2	0	MULTADDALU18X18 Operation Mode and Unit Input Selection 0:18x18 +/- 18x18 +/- C; 1: ACC/0 + 18x18 +/- 18x18; 2:18x18 +/- 18x18 + CASI
MULT_RESET_MODE	SYNC，ASYNC	SYNC	Reset mode config, synchronous or asynchronous

原语例化

Verilog 例化：

```
MULTADDALU18X18 uut(
    .DOUT(dout[53:0]),
    .CASO(caso[54:0]),
    .SOA(soa[17:0]),
    .SOB(sob[17:0]),
    .A0(a0[17:0]),
    .B0(b0[17:0]),
    .A1(a1[17:0]),
    .B1(b1[17:0]),
    .C(c[53:0]),
    .SIA(sia[17:0]),
```

```
        .SIB(sib[17:0]),
        .CASI(casi[54:0]),
        .ACCLOAD(accload),
        .ASEL(asel[1:0]),
        .BSEL(bsel[1:0]),
        .ASIGN(asign[1:0]),
        .BSIGN(bsign[1:0]),
        .CLK(clk),
        .CE(ce),
        .RESET(reset)
    );
    defparam uut.A0REG = 1'b0;
    defparam uut.A1REG = 1'b0;
    defparam uut.B0REG = 1'b0;
    defparam uut.B1REG = 1'b0;
    defparam uut.CREG = 1'b0;
    defparam uut.PIPE0_REG = 1'b0;
    defparam uut.PIPE1_REG = 1'b0;
    defparam uut.OUT_REG = 1'b0;
    defparam uut.ASIGN0_REG = 1'b0;
    defparam uut.ASIGN1_REG = 1'b0;
    defparam uut.ACCLOAD_REG0 = 1'b0;
    defparam uut.ACCLOAD_REG1 = 1'b0;
    defparam uut.BSIGN0_REG = 1'b0;
    defparam uut.BSIGN1_REG = 1'b0;
    defparam uut.SOA_REG = 1'b0;
    defparam uut.B_ADD_SUB = 1'b0;
    defparam uut.C_ADD_SUB = 1'b0;
    defparam uut.MULTADDALU18X18_MODE = 0;
    defparam uut.MULT_RESET_MODE = "SYNC";
```

Vhdl 例化：

```
    COMPONENT MULTADDALU18X18
            GENERIC (A0REG:bit:='0';
                     B0REG:bit:='0';
                     A1REG:bit:='0';
                     B1REG:bit:='0';
                     CREG:bit:='0';
                     OUT_REG:bit:='0';
                     PIPE0_REG:bit:='0';
                     PIPE1_REG:bit:='0';
                     ASIGN0_REG:bit:='0';
                     BSIGN0_REG:bit:='0';
                     ASIGN1_REG:bit:='0';
                     BSIGN1_REG:bit:='0';
                     ACCLOAD_REG0:bit:='0';
                     ACCLOAD_REG1:bit:='0';
                     SOA_REG:bit:='0';
                     B_ADD_SUB:bit:='0';
                     C_ADD_SUB:bit:='0';
                     MULTADDALU18X18_MODE:integer:=0;
```

```
                        MULT_RESET_MODE:string:="SYNC"
                    );
                    PORT(
                        A0:IN std_logic_vector(17 downto 0);
                        A1:IN std_logic_vector(17 downto 0);
                        B0:IN std_logic_vector(17 downto 0);
                        B1:IN std_logic_vector(17 downto 0);
                        SIA:IN std_logic_vector(17 downto 0);
                        SIB:IN std_logic_vector(17 downto 0);
                        C:IN std_logic_vector(53 downto 0);
                        ASIGN:IN std_logic_vector(1 downto 0);
                        BSIGN:IN std_logic_vector(1 downto 0);
                        ASEL:IN std_logic_vector(1 downto 0);
                        BSEL:IN std_logic_vector(1 downto 0);
                        CE:IN std_logic;
                        CLK:IN std_logic;
                        RESET:IN std_logic;
                        ACCLOAD:IN std_logic;
                        CASI:IN std_logic_vector(54 downto 0);
                        SOA:OUT std_logic_vector(17 downto 0);
                        SOB:OUT std_logic_vector(17 downto 0);
                        CASO:OUT std_logic_vector(54 downto 0);
                        DOUT:OUT std_logic_vector(53 downto 0)
                    );
END COMPONENT;
uut:MULTADDALU18X18
                GENERIC MAP (A0REG=>'0',
                             B0REG=>'0',
                             A1REG=>'0',
                             B1REG=>'0',
                             CREG=>'0',
                             OUT_REG=>'0',
                             PIPE0_REG=>'0',
                             PIPE1_REG=>'0',
                             ASIGN0_REG=>'0',
                             BSIGN0_REG=>'0',
                             ASIGN1_REG=>'0',
                             BSIGN1_REG=>'0',
                             ACCLOAD_REG0=>'0',
                             ACCLOAD_REG1=>'0',
                             SOA_REG=>'0',
                             B_ADD_SUB=>'0',
                             C_ADD_SUB=>'0',
                             MULTADDALU18X18_MODE=>0,
                             MULT_RESET_MODE=>"SYNC"
                )
                PORT MAP (
                    A0=>a0,
                    A1=>a1,
                    B0=>b0,
```

```
        B1=>b1,
        SIA=>sia,
        SIB=>sib,
        C=>c,
        ASIGN=>asign,
        BSIGN=>bsign,
        ASEL=>asel,
        BSEL=>bsel,
        CE=>ce,
        CLK=>clk,
        RESET=>reset,
        ACCLOAD=>accload,
        CASI=>casi,
        SOA=>soa,
        SOB=>sob,
        CASO=>caso,
        DOUT=>dout
    );
```

6 Clock

6.1 PLL

原语名称

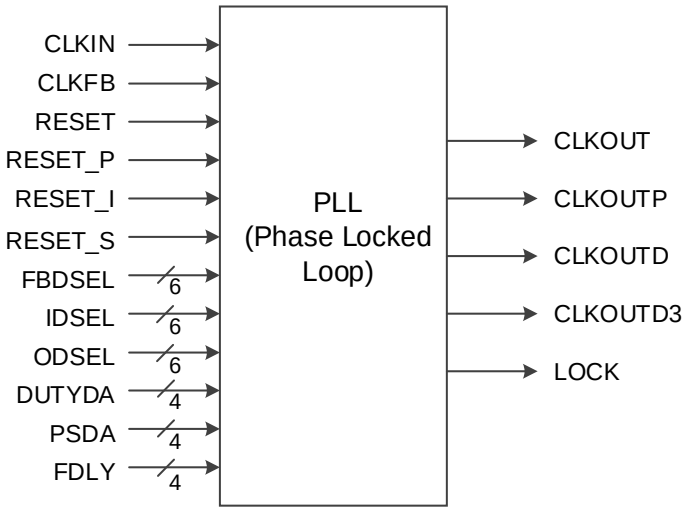
高云 FPGA 提供了 PLL（Phase_Locked Loop，锁相环），利用外部输入的参考时钟信号控制环路内部振荡信号的频率和相位。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 6-1 PLL 端口示意图



功能描述

PLL 可基于给定的输入时钟进行时钟相位调整、占空比调整、频率调整

（倍频和分频）等来产生不同相位和频率的输出时钟。

PLL 的性能如下：

表 6-1 PLL 性能

器件 频率（MHz）	GW1N 家族	GW1N-1S	GW1NS-2	GW1NZ-1	GW2A 家族
输入频率范围	3 ~ 450	3 ~ 450	3 ~ 450	3 ~ 400(LV) 3 ~ 200(ZV)	3 ~ 500
VCO 震荡频率范围	400~900	400~1200	400~1500	400 ~ 800(LV) 200 ~ 400(ZV)	500~1300
输出频率范围	3.125~450	3.125~600	3.125~750	3.125~400(LV) 1.5625~200(ZV)	3.125~500

PLL 可对输入时钟 CLKIN 进行频率调整（倍频和分频），计算公式如下：

1. $f_{\mathrm{CLKOUT}} = (f_{CLKIN} * \mathrm{FBDIV})/\mathrm{IDIV}$
2. $f_{VCO} = f_{CLKOUT} * ODIV$
3. $f_{CLKOUTD} = f_{CLKOUT}/SDIV$
4. $f_{PFD} = f_{CLKIN}/IDIV = f_{CLKOUT}/FBDIV$

注!

- f_{CLKIN} 为输入时钟 CLKIN 频率，f_{CLKOUT} 为 CLKOUT 和 CLKOUTP 时钟频率，$f_{CLKOUTD}$ 为 CLKOUTD 时钟频率，f_{PFD} 为 PFD 鉴相频率；
- IDIV、FBDIV、ODIV、SDIV 为不同分频器实际的分频系数，即可通过调整不同分频系数来得到期望频率的时钟信号。

端口介绍

表 6-2 端口介绍

端口名	I/O	描述
CLKIN	Input	参考时钟输入
CLKFB	Input	反馈时钟输入
RESET	Input	PLL 异步复位输入，高电平有效
RESET_P	Input	PLL 关断（Power Down）输入，高电平有效
RESET_I	Input	PLL IDIV 分频器异步复位输入，高电平有效
RESET_S	Input	PLL SDIV 分频器异步复位输入，高电平有效
FBDSEL[5:0]	Input	动态控制 FBDIV 取值，范围 1~64
IDSEL[5:0]	Input	动态控制 IDIV 取值，范围 1~64
ODSEL[5:0]	Input	动态控制 ODIV 取值， 2,4,8,16,32,48,64,80,96,112,128
DUTYDA[3:0]	Input	占空比动态调整
PSDA[3:0]	Input	相位动态调整
FDLY[3:0]	Input	精细延时动态调整
CLKOUT	Output	PLL 时钟输出
LOCK	Output	PLL 锁定指示，1 表示锁定，0 表示失锁
CLKOUTP	Output	PLL 带有相位和占空比调整的时钟输出
CLKOUTD	Output	PLL 经过 SDIV 的时钟输出，CLKOUT 或 CLKOUTP 经过 SDIV 分频器后的输出
CLKOUTD3	Output	PLL 经过 DIV3 的时钟输出，CLKOUT 或 CLKOUTP

端口名	I/O	描述
		经过 3 分频后的输出

参数介绍

表 6-3 参数介绍

参数名	取值范围	默认值	描述
FCLKIN	3~500	100	参考时钟频率
IDIV_SEL	0~63	0	IDIV 分频系数静态设置
DYN_IDIV_SEL	true,false	false	IDIV 分频系数静态控制参数或动态控制信号选择 false: 静态，即选择参数 IDIV_SEL true: 动态，即选择信号 IDSEL
FBDIV_SEL	0~63	0	FBDIV 分频系数静态设置
DYN_FBDIV_SEL	true,false	false	FBDIV 分频系数静态控制参数或动态控制信号选择 false: 静态，即选择参数 FBDIV_SEL true: 动态，即选择信号 FBDSEL
ODIV_SEL	2,4,8,16,32,48,64,80,96,112,128	8	ODIV 分频系数静态设置
DYN_ODIV_SEL	true,false	false	ODIV 分频系数静态控制参数或动态控制信号选择 false: 静态，即选择参数 ODIV_SEL true: 动态，即选择信号 ODSEL
PSDA_SEL	0000~1111	0000	相位静态调整
DUTYDA_SEL	0010~1110	1000	占空比静态调整
DYN_DA_EN	true,false	false	选择动态信号作为相位和占空比调整的控制 false: 静态控制 true: 动态控制
CLKOUT_FT_DIR	1'b1	1'b1	CLKOUT 微调方向设置 1'b1: add
CLKOUT_DLY_STEP	0,1,2,4	0	CLKOUT 微调系数设置 CLKOUT_DLY_STEP*delay(delay=50ps)
CLKOUTP_FT_DIR	1'b1	1'b1	CLKOUTP 微调方向设置 1'b1: add
CLKOUTP_DLY_STEP	0,1,2	0	CLKOUTP 微调系数设置 CLKOUTP_DLY_STEP*delay(delay=50ps)
DYN_SDIV_SEL	2~128（偶数）	2	SDIV 分频系数静态设置
CLKFB_SEL	internal,external	internal	CLKFB 来源选择 internal:来自内部 CLKOUT 反馈 external: 来自外部信号反馈
CLKOUTD_SRC	CLKOUT,CLKOUTP	CLKOUT	CLKOUTD 来源选择
CLKOUTD3_SRC	CLKOUT,CLKOUTP	CLKOUT	CLKOUTD3 来源选择
CLKOUT_BYPASS	true,false	false	旁路 PLL，CLKOUT 直接来自 CLKIN true: CLKIN 旁路 PLL 直接作用于 CLKOUT false: 正常模式
CLKOUTP_BYP	true,false	false	旁路 PLL，CLKOUTP 直接来自 CLKIN

参数名	取值范围	默认值	描述
ASS			true: CLKIN 旁路 PLL 直接作用于 CLKOUTP false: 正常模式
CLKOUTD_BYPASS	true,false	false	旁路 PLL，CLKOUTD 直接来自 CLKIN true: CLKIN 旁路 PLL 直接作用于 CLKOUTD false: 正常模式
DEVICE	GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。	GW1N-2	器件选择

原语例化

Verilog 例化：

```
PLL pll_inst(
        .CLKOUT(clkout),
        .LOCK(lock),
        .CLKOUTP(clkoutp),
        .CLKOUTD(clkoutd),
        .CLKOUTD3(clkoutd3),
        .RESET(reset),
        .RESET_P(reset_p),
        .RESET_I(reset_i),
        .RESET_S(reset_s),
        .CLKIN(clkin),
        .CLKFB(clkfb),
        .FBDSEL(fbdsel),
        .IDSEL(idsel),
        .ODSEL(odsel),
        .PSDA(psda),
        .DUTYDA(dutyda),
        .FDLY(fdly)
);
defparam pll_inst.FCLKIN = "50";
defparam pll_inst.DYN_IDIV_SEL = "false";
defparam pll_inst.IDIV_SEL = 0;
defparam pll_inst.DYN_FBDIV_SEL = "false";
defparam pll_inst.FBDIV_SEL = 1;
defparam pll_inst.ODIV_SEL = 8;
defparam pll_inst.PSDA_SEL = "0100";
defparam pll_inst.DYN_DA_EN = "false";
```

```
defparam pll_inst.DUTYDA_SEL = "1000";
defparam pll_inst.CLKOUT_FT_DIR = 1'b1;
defparam pll_inst.CLKOUTP_FT_DIR = 1'b1;
defparam pll_inst.CLKOUT_DLY_STEP = 0;
defparam pll_inst.CLKOUTP_DLY_STEP = 0;
defparam pll_inst.CLKFB_SEL ="external";
defparam pll_inst.CLKOUT_BYPASS = "false";
defparam pll_inst.CLKOUTP_BYPASS = "false";
defparam pll_inst.CLKOUTD_BYPASS = "false";
defparam pll_inst.DYN_SDIV_SEL = 2;
defparam pll_inst.CLKOUTD_SRC = "CLKOUT";
defparam pll_inst.CLKOUTD3_SRC = "CLKOUT";
defparam pll_inst.DEVICE = "GW1N-4";
```

Vhdl 例化：

```
COMPONENT PLL
    GENERIC(
            FCLKIN:STRING:= "100.0";
            DEVICE:STRING:= "GW2A-18";
            DYN_IDIV_SEL:STRING:="false";
            IDIV_SEL:integer:=0;
            DYN_FBDIV_SEL:STRING:="false";
            FBDIV_SEL:integer:=0;
            DYN_ODIV_SEL:STRING:="false";
            ODIV_SEL:integer:=8;
            PSDA_SEL:STRING:="0000";
            DYN_DA_EN:STRING:="false";
            DUTYDA_SEL:STRING:="1000";
            CLKOUT_FT_DIR:bit:='1';
            CLKOUTP_FT_DIR:bit:='1';
            CLKOUT_DLY_STEP:integer:=0;
            CLKOUTP_DLY_STEP:integer:=0;
            CLKOUTD3_SRC:STRING:="CLKOUT";
            CLKFB_SEL : STRING:="internal";
            CLKOUT_BYPASS:STRING:="false";
            CLKOUTP_BYPASS:STRING:="false";
            CLKOUTD_BYPASS:STRING:="false";
            CLKOUTD_SRC:STRING:="CLKOUT";
            DYN_SDIV_SEL:integer:=2
     );
        PORT(
            CLKIN:IN  std_logic;
            CLKFB:IN  std_logic;
            IDSEL:IN  std_logic_vector(5  downto  0);
```

```
                FBDSEL:IN std_logic_vector(5 downto 0);
                ODSEL:IN std_logic_vector(5 downto 0);
                RESET:IN std_logic;
                RESET_P:IN std_logic;
                RESET_I:IN std_logic;
                RESET_S:IN std_logic;
                PSDA,FDLY:IN std_logic_vector(3 downto 0);
                DUTYDA:IN std_logic_vector(3 downto 0);
                LOCK:OUT std_logic;
                CLKOUT:OUT std_logic;
                CLKOUTD:OUT std_logic;
                CLKOUTP:OUT std_logic;
                CLKOUTD3:OUT std_logic
            );
END COMPONENT;
uut:PLL
    GENERIC MAP(
                    FCLKIN =>"100.0",
                    DEVICE =>"GW2A-18",
                    DYN_IDIV_SEL=>"false",
                    IDIV_SEL=>0,
                    DYN_FBDIV_SEL=>"false",
                    FBDIV_SEL=>0,
                    DYN_ODIV_SEL=>"false",
                    ODIV_SEL=>8,
                    PSDA_SEL=>"0000",
                    DYN_DA_EN=>"false",
                    DUTYDA_SEL=>"1000",
                    CLKOUT_FT_DIR=>'1',
                    CLKOUTP_FT_DIR=>'1',
                    CLKOUT_DLY_STEP=>0,
                    CLKOUTP_DLY_STEP=>0,
                    CLKOUTD3_SRC=>"CLKOUT",
                    CLKFB_SEL=>"internal",
                    CLKOUT_BYPASS=>"false",
                    CLKOUTP_BYPASS=>"false",
                    CLKOUTD_BYPASS=>"false",
                    CLKOUTD_SRC=>"CLKOUT",
                    DYN_SDIV_SEL=>2
    )
```

```
        PORT  MAP(
            CLKIN=>clkin,
            CLKFB=>clkfb,
            IDSEL=>idsel,
            FBDSEL=>fbdsel,
            ODSEL=>odsel,
            RESET=>reset,
            RESET_P=>reset_p,
            RESET_I=>reset_i,
            RESET_S=>reset_s,
            PSDA=>psda,
            FDLY=>fdly,
            DUTYDA=>dutyda,
            LOCK=>lock,
            CLKOUT=>clkout,
            CLKOUTD=>clkoutd,
            CLKOUTP=>clkoutp ,
            CLKOUTD3=>clkoutd3
    );
```

6.2 rPLL

原语名称

高云 FPGA 提供了 rPLL（Phase_Locked Loop，锁相环），利用外部输入的参考时钟信号控制环路内部振荡信号的频率和相位。

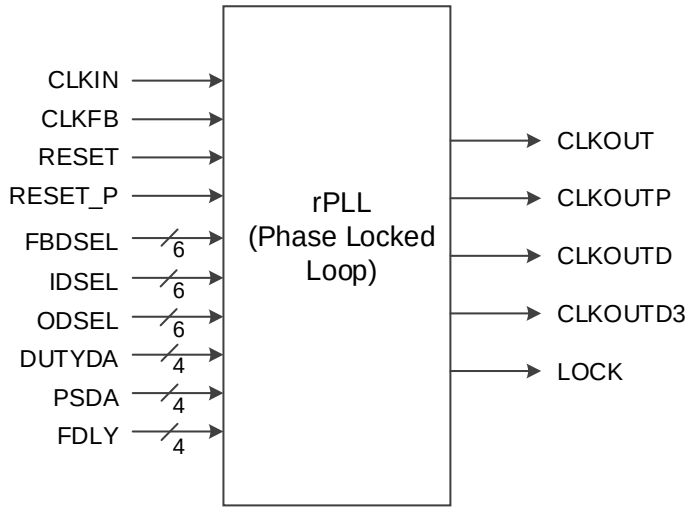
rPLL 是 PLL 的修订版，删掉了复位信号 RESET_I 和 RESET_S。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 6-2 rPLL 端口示意图



功能描述

rPLL 可基于给定的输入时钟进行时钟相位调整、占空比调整、频率调整（倍频和分频）等来产生不同相位和频率的输出时钟。

rPLL 的性能如下：

表 6-4 rPLL 性能

器件 / 频率（MHz）	GW1N 家族	GW1N-1S	GW1NS-2	GW1NZ-1	GW2A 家族
输入频率范围	3 ~ 450	3 ~ 450	3 ~ 450	3 ~ 400(LV) 3 ~ 200(ZV)	3 ~ 500
VCO 震荡频率范围	400~900	400~1200	400~1500	400 ~ 800(LV) 200 ~ 400(ZV)	500~1300
输出频率范围	3.125~450	3.125~600	3.125~750	3.125~400(LV) 1.5625~200(ZV)	3.125~500

rPLL 可对输入时钟 CLKIN 进行频率调整（倍频和分频），计算公式如下：

1. $f_{\mathrm{CLKOUT}} = (f_{CLKIN} * \mathrm{FBDIV})/\mathrm{IDIV}$
2. $f_{VCO} = f_{CLKOUT} * ODIV$
3. $f_{CLKOUTD} = f_{CLKOUT}/SDIV$
4. $f_{PFD} = f_{CLKIN}/IDIV = f_{CLKOUT}/FBDIV$

注：

- f_{CLKIN} 为输入时钟 CLKIN 频率，f_{CLKOUT} 为 CLKOUT 和 CLKOUTP 时钟频率，$f_{CLKOUTD}$ 为 CLKOUTD 时钟频率，f_{PFD} 为 PFD 鉴相频率；
- IDIV、FBDIV、ODIV、SDIV 为不同分频器实际的分频系数，即可通过调整不同分频系数来得到期望频率的时钟信号。

端口介绍

表 6-5 端口介绍

端口名	I/O	描述
CLKIN	Input	参考时钟输入
CLKFB	Input	反馈时钟输入
RESET	Input	rPLL 异步复位输入，高电平有效
RESET_P	Input	rPLL 关断（Power Down）输入，高电平有效
FBDSEL[5:0]	Input	动态控制 FBDIV 取值，范围 1~64
IDSEL[5:0]	Input	动态控制 IDIV 取值，范围 1~64
ODSEL[5:0]	Input	动态控制 ODIV 取值， 2,4,8,16,32,48,64,80,96,112,128
DUTYDA[3:0]	Input	占空比动态调整
PSDA[3:0]	Input	相位动态调整
FDLY[3:0]	Input	精细延时动态调整
CLKOUT	Output	rPLL 时钟输出
LOCK	Output	rPLL 锁定指示，1 表示锁定，0 表示失锁
CLKOUTP	Output	rPLL 带有相位和占空比调整的时钟输出
CLKOUTD	Output	rPLL 经过 SDIV 的时钟输出，CLKOUT 或 CLKOUTP 经过 SDIV 分频器后的输出
CLKOUTD3	Output	rPLL 经过 DIV3 的时钟输出，CLKOUT 或 CLKOUTP 经过 3 分频后的输出

参数介绍

表 6-6 参数介绍

参数名	取值范围	默认值	描述
FCLKIN	3~500	100	参考时钟频率
IDIV_SEL	0~63	0	IDIV 分频系数静态设置
DYN_IDIV_SEL	true,false	false	IDIV 分频系数静态控制参数或动态控制信号选择 false: 静态，即选择参数 IDIV_SEL true: 动态，即选择信号 IDSEL
FBDIV_SEL	0~63	0	FBDIV 分频系数静态设置
DYN_FBDIV_SEL	true,false	false	FBDIV 分频系数静态控制参数或动态控制信号选择 false: 静态，即选择参数 FBDIV_SEL true: 动态，即选择信号 FBDSEL
ODIV_SEL	2,4,8,16,32,48,64,80,96,112,128	8	ODIV 分频系数静态设置
DYN_ODIV_SEL	true,false	false	ODIV 分频系数静态控制参数或动态控制信号选择 false: 静态，即选择参数 ODIV_SEL true: 动态，即选择信号 ODSEL
PSDA_SEL	0000~1111	0000	相位静态调整

参数名	取值范围	默认值	描述
DUTYDA_SEL	0010~1110	1000	占空比静态调整
DYN_DA_EN	true,false	false	选择动态信号作为相位和占空比调整的控制 false: 静态控制 true: 动态控制
CLKOUT_FT_DIR	1'b1	1'b1	CLKOUT 微调方向设置 1'b1: add
CLKOUT_DLY_STEP	0,1,2,4	0	CLKOUT 微调系数设置 CLKOUT_DLY_STEP*delay(delay=50ps)
CLKOUTP_FT_DIR	1'b1	1'b1	CLKOUTP 微调方向设置 1'b1: add
CLKOUTP_DLY_STEP	0,1,2	0	CLKOUTP 微调系数设置 CLKOUTP_DLY_STEP*delay(delay=50ps)
DYN_SDIV_SEL	2~128（偶数）	2	SDIV 分频系数静态设置
CLKFB_SEL	internal,external	internal	CLKFB 来源选择 internal:来自内部 CLKOUT 反馈 external: 来自外部信号反馈
CLKOUTD_SRC	CLKOUT,CLKOUTP	CLKOUT	CLKOUTD 来源选择
CLKOUTD3_SRC	CLKOUT,CLKOUTP	CLKOUT	CLKOUTD3 来源选择
CLKOUT_BYPASS	true,false	false	旁路 rPLL，CLKOUT 直接来自 CLKIN true: CLKIN 旁路 rPLL 直接作用于 CLKOUT false: 正常模式
CLKOUTP_BYPASS	true,false	false	旁路 rPLL，CLKOUTP 直接来自 CLKIN true: CLKIN 旁路 rPLL 直接作用于 CLKOUTP false: 正常模式
CLKOUTD_BYPASS	true,false	false	旁路 rPLL，CLKOUTD 直接来自 CLKIN true: CLKIN 旁路 rPLL 直接作用于 CLKOUTD false: 正常模式
DEVICE	GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。	GW1N-2	器件选择

原语例化

Verilog 例化：

```
rPLL rpll_inst(
        .CLKOUT(clkout),
        .LOCK(lock),
        .CLKOUTP(clkoutp),
        .CLKOUTD(clkoutd),
        .CLKOUTD3(clkoutd3),
        .RESET(reset),
        .RESET_P(reset_p),
        .CLKIN(clkin),
        .CLKFB(clkfb),
        .FBDSEL(fbdsel),
        .IDSEL(idsel),
        .ODSEL(odsel),
        .PSDA(psda),
        .DUTYDA(dutyda),
        .FDLY(fdly)
);
defparam rpll_inst.FCLKIN = "50";
defparam rpll_inst.DYN_IDIV_SEL = "false";
defparam rpll_inst.IDIV_SEL = 0;
defparam rpll_inst.DYN_FBDIV_SEL = "false";
defparam rpll_inst.FBDIV_SEL = 1;
defparam rpll_inst.ODIV_SEL = 8;
defparam rpll_inst.PSDA_SEL = "0100";
defparam rpll_inst.DYN_DA_EN = "false";
defparam rpll_inst.DUTYDA_SEL = "1000";
defparam rpll_inst.CLKOUT_FT_DIR = 1'b1;
defparam rpll_inst.CLKOUTP_FT_DIR = 1'b1;
defparam rpll_inst.CLKOUT_DLY_STEP = 0;
defparam rpll_inst.CLKOUTP_DLY_STEP = 0;
defparam rpll_inst.CLKFB_SEL ="external";
defparam rpll_inst.CLKOUT_BYPASS = "false";
defparam rpll_inst.CLKOUTP_BYPASS = "false";
defparam rpll_inst.CLKOUTD_BYPASS = "false";
defparam rpll_inst.DYN_SDIV_SEL = 2;
defparam rpll_inst.CLKOUTD_SRC = "CLKOUT";
defparam rpll_inst.CLKOUTD3_SRC = "CLKOUT";
defparam rpll_inst.DEVICE = "GW1N-4";
```

Vhdl 例化：

```
COMPONENT rPLL
    GENERIC(
            FCLKIN:STRING:= "100.0";
            DEVICE:STRING:= "GW1N-2";
            DYN_IDIV_SEL:STRING:="false";
            IDIV_SEL:integer:=0;
            DYN_FBDIV_SEL:STRING:="false";
```

```
                FBDIV_SEL:integer:=0;
                DYN_ODIV_SEL:STRING:="false";
                ODIV_SEL:integer:=8;
                PSDA_SEL:STRING:="0000";
                DYN_DA_EN:STRING:="false";
                DUTYDA_SEL:STRING:="1000";
                CLKOUT_FT_DIR:bit:='1';
                CLKOUTP_FT_DIR:bit:='1';
                CLKOUT_DLY_STEP:integer:=0;
                CLKOUTP_DLY_STEP:integer:=0;
                CLKOUTD3_SRC:STRING:="CLKOUT";
                CLKFB_SEL : STRING:="internal";
                CLKOUT_BYPASS:STRING:="false";
                CLKOUTP_BYPASS:STRING:="false";
                CLKOUTD_BYPASS:STRING:="false";
                CLKOUTD_SRC:STRING:="CLKOUT";
                DYN_SDIV_SEL:integer:=2
        );
            PORT(
                CLKIN:IN std_logic;
                CLKFB:IN std_logic;
                IDSEL:IN std_logic_vector(5 downto 0);
                FBDSEL:IN std_logic_vector(5 downto 0);
                ODSEL:IN std_logic_vector(5 downto 0);
                RESET:IN std_logic;
                RESET_P:IN std_logic;
                PSDA,FDLY:IN std_logic_vector(3 downto 0);
                DUTYDA:IN std_logic_vector(3 downto 0);
                LOCK:OUT std_logic;
                CLKOUT:OUT std_logic;
                CLKOUTD:OUT std_logic;
                CLKOUTP:OUT std_logic;
                CLKOUTD3:OUT std_logic
            );
    END COMPONENT;
    uut:rPLL
        GENERIC MAP(
                    FCLKIN =>"100.0",
                    DEVICE =>"GW2A-18",
                    DYN_IDIV_SEL=>"false",
```

```
                IDIV_SEL=>0,
                DYN_FBDIV_SEL=>"false",
                FBDIV_SEL=>0,
                DYN_ODIV_SEL=>"false",
                ODIV_SEL=>8,
                PSDA_SEL=>"0000",
                DYN_DA_EN=>"false",
                DUTYDA_SEL=>"1000",
                CLKOUT_FT_DIR=>'1',
                CLKOUTP_FT_DIR=>'1',
                CLKOUT_DLY_STEP=>0,
                CLKOUTP_DLY_STEP=>0,
                CLKOUTD3_SRC=>"CLKOUT",
                CLKFB_SEL=>"internal",
                CLKOUT_BYPASS=>"false",
                CLKOUTP_BYPASS=>"false",
                CLKOUTD_BYPASS=>"false",
                CLKOUTD_SRC=>"CLKOUT",
                DYN_SDIV_SEL=>2
        )
        PORT MAP(
            CLKIN=>clkin,
            CLKFB=>clkfb,
            IDSEL=>idsel,
            FBDSEL=>fbdsel,
            ODSEL=>odsel,
            RESET=>reset,
            RESET_P=>reset_p,
            PSDA=>psda,
            FDLY=>fdly,
            DUTYDA=>dutyda,
            LOCK=>lock,
            CLKOUT=>clkout,
            CLKOUTD=>clkoutd,
            CLKOUTP=>clkoutp,
            CLKOUTD3=>clkoutd3
    );
```

6.3 PLLVR

原语名称

高云 FPGA 提供了 PLLVR（Phase_Locked Loop，锁相环），利用外部输入的参考时钟信号控制环路内部振荡信号的频率和相位。

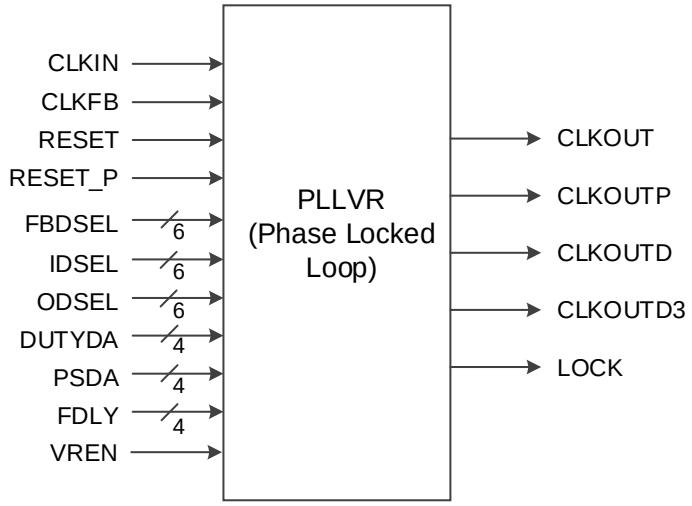
PLLVR 与 PLL 相比，删掉了复位信号 RESET_I 和 RESET_S，增加了电源调节信号 VREN。

适用器件

支持器件：GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C。

端口示意图

图 6-3 PLLVR 端口示意图



功能描述

PLLVR 可基于给定的输入时钟进行时钟相位调整、占空比调整、频率调整（倍频和分频）等来产生不同相位和频率的输出时钟。

PLLVR 的性能如下：

表 6-7 PLLVR 性能

器件 / 频率（MHz）	GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C
输入频率范围	3 ~ 500
VCO 震荡频率范围	500~1500
输出频率范围	3.906~750

PLLVR 可对输入时钟 CLKIN 进行频率调整（倍频和分频），计算公式如下：

1. $f_{\mathrm{CLKOUT}} = (f_{CLKIN} * \mathrm{FBDIV})/\mathrm{IDIV}$
2. $f_{VCO} = f_{CLKOUT} * ODIV$
3. $f_{CLKOUTD} = f_{CLKOUT}/SDIV$
4. $f_{PFD} = f_{CLKIN}/IDIV = f_{CLKOUT}/FBDIV$

注：

- f_{CLKIN} 为输入时钟 CLKIN 频率，f_{CLKOUT} 为 CLKOUT 和 CLKOUTP 时钟频率，$f_{CLKOUTD}$ 为 CLKOUTD 时钟频率，f_{PFD} 为 PFD 鉴相频率；
- IDIV、FBDIV、ODIV、SDIV 为不同分频器实际的分频系数，即可通过调整不同分频系数来得到期望频率的时钟信号。

端口介绍

表 6-8 端口介绍

端口名	I/O	描述
CLKIN	Input	参考时钟输入
CLKFB	Input	反馈时钟输入
RESET	Input	PLLVR 异步复位输入，高电平有效
RESET_P	Input	PLLVR 关断（Power Down）输入，高电平有效
FBDSEL[5:0]	Input	动态控制 FBDIV 取值，范围 1~64
IDSEL[5:0]	Input	动态控制 IDIV 取值，范围 1~64
ODSEL[5:0]	Input	动态控制 ODIV 取值， 2,4,8,16,32,48,64,80,96,112,128
DUTYDA[3:0]	Input	占空比动态调整
PSDA[3:0]	Input	相位动态调整
FDLY[3:0]	Input	精细延时动态调整
VREN	Input	PLLVR 电源调节使能信号，高电平有效
CLKOUT	Output	PLLVR 时钟输出
LOCK	Output	PLLVR 锁定指示，1 表示锁定，0 表示失锁
CLKOUTP	Output	PLLVR 带有相位和占空比调整的时钟输出
CLKOUTD	Output	PLLVR 经过 SDIV 的时钟输出，CLKOUT 或 CLKOUTP 经过 SDIV 分频器后的输出
CLKOUTD3	Output	PLLVR 经过 DIV3 的时钟输出，CLKOUT 或 CLKOUTP 经过 3 分频后的输出

参数介绍

表 6-9 参数介绍

参数名	取值范围	默认值	描述
FCLKIN	3~500	100	参考时钟频率
IDIV_SEL	0~63	0	IDIV 分频系数静态设置
DYN_IDIV_SEL	true,false	false	IDIV 分频系数静态控制参数或动态控制信号选择 false: 静态，即选择参数 IDIV_SEL true: 动态，即选择信号 IDSEL
FBDIV_SEL	0~63	0	FBDIV 分频系数静态设置
DYN_FBDIV_SEL	true,false	false	FBDIV 分频系数静态控制参数或动态控制信号选择 false: 静态，即选择参数 FBDIV_SEL true: 动态，即选择信号 FBDSEL

参数名	取值范围	默认值	描述
ODIV_SEL	2,4,8,16,32,48,64,80,96,112,128	8	ODIV 分频系数静态设置
DYN_ODIV_SEL	true,false	false	ODIV 分频系数静态控制参数或动态控制信号选择 false: 静态，即选择参数 ODIV_SEL true: 动态，即选择信号 ODSEL
PSDA_SEL	0000~1111	0000	相位静态调整
DUTYDA_SEL	0010~1110	1000	占空比静态调整
DYN_DA_EN	true,false	false	选择动态信号作为相位和占空比调整的控制 false: 静态控制 true: 动态控制
CLKOUT_FT_DIR	1'b1	1'b1	CLKOUT 微调方向设置 1'b1: add
CLKOUT_DLY_STEP	0,1,2,4	0	CLKOUT 微调系数设置 CLKOUT_DLY_STEP*delay(delay=50ps)
CLKOUTP_FT_DIR	1'b1	1'b1	CLKOUTP 微调方向设置 1'b1: add
CLKOUTP_DLY_STEP	0,1,2	0	CLKOUTP 微调系数设置 CLKOUTP_DLY_STEP*delay(delay=50ps)
DYN_SDIV_SEL	2~128（偶数）	2	SDIV 分频系数静态设置
CLKFB_SEL	internal,external	internal	CLKFB 来源选择 internal:来自内部 CLKOUT 反馈 external: 来自外部信号反馈
CLKOUTD_SRC	CLKOUT,CLKOUTP	CLKOUT	CLKOUTD 来源选择
CLKOUTD3_SRC	CLKOUT,CLKOUTP	CLKOUT	CLKOUTD3 来源选择
CLKOUT_BYPASS	true,false	false	旁路 PLLVR，CLKOUT 直接来自 CLKIN true: CLKIN 旁路 PLLVR 直接作用于 CLKOUT false: 正常模式
CLKOUTP_BYPASS	true,false	false	旁路 PLLVR，CLKOUTP 直接来自 CLKIN true: CLKIN 旁路 PLLVR 直接作用于 CLKOUTP false: 正常模式
CLKOUTD_BYPASS	true,false	false	旁路 PLLVR，CLKOUTD 直接来自 CLKIN true: CLKIN 旁路 PLLVR 直接作用于 CLKOUTD false: 正常模式
DEVICE	GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C。	GW1N-2	器件选择

原语例化

Verilog 例化：

```
PLLVR pllvr_inst(
    .CLKOUT(clkout),
    .LOCK(lock),
    .CLKOUTP(clkoutp),
```

```
        .CLKOUTD(clkoutd),
        .CLKOUTD3(clkoutd3),
        .VREN(vren),
        .RESET(reset),
        .RESET_P(reset_p),
        .CLKIN(clkin),
        .CLKFB(clkfb),
        .FBDSEL(fbdsel),
        .IDSEL(idsel),
        .ODSEL(odsel),
        .PSDA(psda),
        .DUTYDA(dutyda),
        .FDLY(fdly)
    );
    defparam pllvr_inst.FCLKIN = "50";
    defparam pllvr_inst.DYN_IDIV_SEL = "false";
    defparam pllvr_inst.IDIV_SEL = 0;
    defparam pllvr_inst.DYN_FBDIV_SEL = "false";
    defparam pllvr_inst.FBDIV_SEL = 1;
    defparam pllvr_inst.ODIV_SEL = 8;
    defparam pllvr_inst.PSDA_SEL = "0100";
    defparam pllvr_inst.DYN_DA_EN = "false";
    defparam pllvr_inst.DUTYDA_SEL = "1000";
    defparam pllvr_inst.CLKOUT_FT_DIR = 1'b1;
    defparam pllvr_inst.CLKOUTP_FT_DIR = 1'b1;
    defparam pllvr_inst.CLKOUT_DLY_STEP = 0;
    defparam pllvr_inst.CLKOUTP_DLY_STEP = 0;
    defparam pllvr_inst.CLKFB_SEL ="external";
    defparam pllvr_inst.CLKOUT_BYPASS = "false";
    defparam pllvr_inst.CLKOUTP_BYPASS = "false";
    defparam pllvr_inst.CLKOUTD_BYPASS = "false";
    defparam pllvr_inst.DYN_SDIV_SEL = 2;
    defparam pllvr_inst.CLKOUTD_SRC = "CLKOUT";
    defparam pllvr_inst.CLKOUTD3_SRC = "CLKOUT";
    defparam pllvr_inst.DEVICE = "GW1NS-4";
```

Vhdl 例化：

```
    COMPONENT PLLVR
        GENERIC(
                FCLKIN:STRING:= "100.0";
                DEVICE:STRING:= "GW1NS-4";
                DYN_IDIV_SEL:STRING:="false";
                IDIV_SEL:integer:=0;
                DYN_FBDIV_SEL:STRING:="false";
                FBDIV_SEL:integer:=0;
                DYN_ODIV_SEL:STRING:="false";
                ODIV_SEL:integer:=8;
                PSDA_SEL:STRING:="0000";
```

```
                DYN_DA_EN:STRING:="false";
                DUTYDA_SEL:STRING:="1000";
                CLKOUT_FT_DIR:bit:='1';
                CLKOUTP_FT_DIR:bit:='1';
                CLKOUT_DLY_STEP:integer:=0;
                CLKOUTP_DLY_STEP:integer:=0;
                CLKOUTD3_SRC:STRING:="CLKOUT";
                CLKFB_SEL : STRING:="internal";
                CLKOUT_BYPASS:STRING:="false";
                CLKOUTP_BYPASS:STRING:="false";
                CLKOUTD_BYPASS:STRING:="false";
                CLKOUTD_SRC:STRING:="CLKOUT";
                DYN_SDIV_SEL:integer:=2
        );
            PORT(
                CLKIN:IN std_logic;
                CLKFB:IN std_logic;
                IDSEL:IN std_logic_vector(5 downto 0);
                FBDSEL:IN std_logic_vector(5 downto 0);
                ODSEL:IN std_logic_vector(5 downto 0);
                VREN:IN std_logic;
                RESET:IN std_logic;
                RESET_P:IN std_logic;
                PSDA,FDLY:IN std_logic_vector(3 downto 0);
                DUTYDA:IN std_logic_vector(3 downto 0);
                LOCK:OUT std_logic;
                CLKOUT:OUT std_logic;
                CLKOUTD:OUT std_logic;
                CLKOUTP:OUT std_logic;
                CLKOUTD3:OUT std_logic
            );
    END COMPONENT;
    uut:PLLVR
        GENERIC MAP(
                    FCLKIN =>"100.0",
                    DEVICE =>"GW1NS-4",
                    DYN_IDIV_SEL=>"false",
                    IDIV_SEL=>0,
                    DYN_FBDIV_SEL=>"false",
                    FBDIV_SEL=>0,
```

```
                DYN_ODIV_SEL=>"false",
                ODIV_SEL=>8,
                PSDA_SEL=>"0000",
                DYN_DA_EN=>"false",
                DUTYDA_SEL=>"1000",
                CLKOUT_FT_DIR=>'1',
                CLKOUTP_FT_DIR=>'1',
                CLKOUT_DLY_STEP=>0,
                CLKOUTP_DLY_STEP=>0,
                CLKOUTD3_SRC=>"CLKOUT",
                CLKFB_SEL=>"internal",
                CLKOUT_BYPASS=>"false",
                CLKOUTP_BYPASS=>"false",
                CLKOUTD_BYPASS=>"false",
                CLKOUTD_SRC=>"CLKOUT",
                DYN_SDIV_SEL=>2
    )
     PORT  MAP(
        CLKIN=>clkin,
        CLKFB=>clkfb,
        IDSEL=>idsel,
        FBDSEL=>fbdsel,
        ODSEL=>odsel,
        VREN=>vren,
        RESET=>reset,
        RESET_P=>reset_p,
        PSDA=>psda,
        FDLY=>fdly,
        DUTYDA=>dutyda,
        LOCK=>lock,
        CLKOUT=>clkout,
        CLKOUTD=>clkoutd,
        CLKOUTP=>clkoutp,
        CLKOUTD3=>clkoutd3
);
```

6.4 DLL/DLLDLY

6.4.1 DLL

原语名称

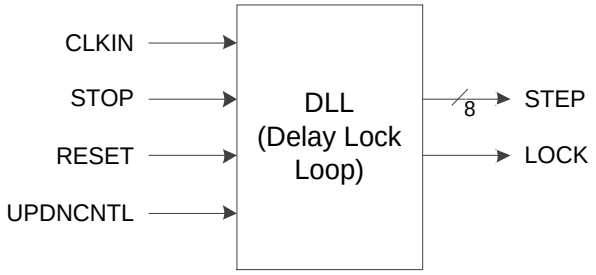
DLL （Delay-Locked Loop）是延时锁相环，主要用于精确产生时间延时。

适用器件

支持器件：GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 6-4 DLL 端口示意图



功能描述

DLL 可基于给定的输入时钟进行时钟相位调整产生不同相位的延时步长 STEP。计算过的输出 STEP 信号会送到相邻的 Bank 中，如 DQS、DLLDLY 模块，同时信号 STEP 也可通过布线送到用户逻辑中去。

端口介绍

表 6-10 端口介绍

端口名	I/O	描述
STEP[7:0]	Output	延时步长输出
LOCK	Output	锁定指示输出，1 表示锁定，0 表示失锁
CLKIN	Input	时钟输入
STOP	Input	停止输入时钟和内部震荡时钟
RESET	Input	异步复位输入，高电平有效
UPDNCNTL	Input	DLL 延时步长更新控制，低电平有效

参数介绍

表 6-11 参数介绍

参数名	参数类型	取值范围	默认值	描述
DLL_FORCE	Integer	0,1	0	DLL 强制延时步长、锁定控制 1:强制锁定，延时步长为 255（最大），用于较低的输入频率模式 0: 正常模式，通过 DLL 生成延时步长和锁定信号
CODESCAL	String	000,001,010,011, 100,101, 110, 111	000	DLL 相移配置（45°~135°） 000:101° 001:112° 010:123° 011:135° 100:79° 101:68° 110:57° 111:45°
SCAL_EN	String	true,false	true	DLL 启用相位偏移功能: true:启用，相位偏移根据参数 CODESCAL 设置 false:禁用，默认 90°相移
DIV_SEL	Integer	1’b0,1’b1	1’b0	DLL 锁定模式选择: 1'b0:正常锁定模式 1'b1:快速锁定模式

连接合法性规则

DLL 的输出 STEP 可连接至 DQS、DLLDLY 模块，同时也可通过布线送到用户逻辑中去。

原语例化

Verilog 例化：

```
DLL dll_inst (
        .STEP(step),
        .LOCK(lock),
        .CLKIN(clkin),
        .STOP(stop),
        .RESET(reset),
        .UPDNCNTL(1’b0)
);
defparam dll_inst.DLL_FORCE = 1;
defparam dll_inst.CODESCAL = "000";
defparam dll_inst.SCAL_EN = "true";
defparam dll_inst.DIV_SEL = 1'b0;
```

Vhdl 例化：

```
COMPONENT DLL
        GENERIC(
```

```
                    DLL_FORCE:integer:=0;
                    DIV_SEL:bit:='1';
                    CODESCAL:STRING:="000";
                    SCAL_EN:STRING:="true"
            );
                PORT(
                    CLKIN:IN std_logic;
                    STOP:IN std_logic;
                    RESET:IN std_logic;
                    UPDNCNTL:IN std_logic;
                    LOCK:OUT std_logic;
                    STEP:OUT std_logic_vector(7 downto 0)
                );
END COMPONENT;
uut:DLL
    GENERIC MAP(
                    DLL_FORCE=>0,
                    DIV_SEL=>'1',
                    CODESCAL=>"000",
                    SCAL_EN=>"true"
    )
     PORT MAP(
        CLKIN=>clkin,
        STOP=>stop,
        RESET=>reset,
        UPDNCNTL=>updncntl,
        LOCK=>lock,
        STEP=>step
        );
```

6.4.2 DLLDLY

原语名称

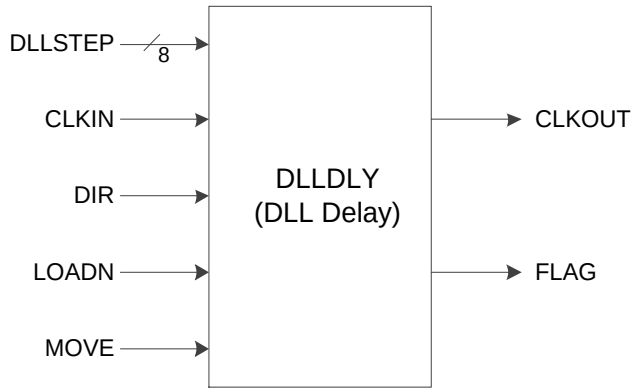
DLLDLY(DLL Delay)为时钟延时模块，依据 DLLSTEP 信号对输入时钟进行调整，得到该时钟的延时调整输出。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 6-5 DLLDLY 端口示意图



功能描述

DLLDLY 可与 DLL 配合使用，DLL 提供不同相位的延时 STEP，DLLDLY 根据 DLLSTEP 产生对应相位的延时，得到基于 CLKIN 的延时输出。

端口介绍

表 6-12 端口介绍

端口名	I/O	描述
CLKOUT	Output	时钟输出
FLAG	Output	输出标志，用以表示动态调整延时的 under-flow 或 over-flow
DLLSTEP[7:0]	Input	延时步长输入，来自 DLL 的输出 STEP
CLKIN	Input	时钟输入
DIR	Input	设置动态调整延时的方向 0: 增加延时； 1: 减少延时
LOADN	Input	控制加载延时步长 0: 加载延时步长 DLLSTEP； 1: 动态调整延时
MOVE	Input	MOVE 为下降沿时动态调整延时，每个脉冲移动一个延时步长

参数介绍

表 6-13 参数介绍

参数名	参数类型	取值范围	默认值	描述
DLL_INSEL	Integer	1’b0,1’b1	1’b0	DLLDLY 旁路模式选择 1'b0:旁路模式，即输出直接来自 CLKIN 1'b1: 正常模式，使用 DLLDLY 延时模块
DLY_SIGN	String	1’b0,1’b1	1’b0	设置调整延时的符号:

参数名	参数类型	取值范围	默认值	描述
				1'b0:'+' 1'b1: '-'
DLY_ADJ	Integer	0~255	0	延时调整设置: dly_sign=0 DLY_ADJ; dly_sign=1 -256+ DLY_ADJ

连接合法性规则

DLLDLY 的输入 DLLSTEP 来自 DLL 模块的 STEP，若器件没有 DLL 则可来自用户逻辑。

原语例化

Verilog 例化：

```
DLLDLY dlldly_0 (
    .CLKIN(clkin),
    .DLLSTEP(step[7:0]),
    .DIR(dir),
    .LOADN(loadn),
    .MOVE(move),
    .CLKOUT(clkout),
    .FLAG(flag)
);
defparam dlldly_0.DLL_INSEL=1'b1;
defparam dlldly_0.DLY_SIGN=1'b1;
defparam dlldly_0.DLY_ADJ=0;
```

Vhdl 例化：

```
COMPONENT DLLDLY
        GENERIC(
                DLL_INSEL:bit:='0';
                DLY_SIGN:bit:='0';
                LY_ADJ:integer:=0
         );
             PORT(
                DLLSTEP:IN std_logic_vector(7 downto 0);
                CLKIN:IN std_logic;
                DIR,LOADN,MOVE:IN std_logic;
                CLKOUT:OUT std_logic;
                FLAG:OUT std_logic
            );
END COMPONENT;
uut:DLLDLY
    GENERIC MAP(
                 DLL_INSEL=>'0',
                 DLY_SIGN=>'0',
                 LY_ADJ=>0
     )
    PORT MAP(
```

```
    DLLSTEP=>step,
    CLKIN=>clkin,
    DIR=>dir,
    LOADN=>loadn,
    MOVE=>move,
    CLKOUT=>clkout,
    FLAG=>flag
);
```

6.5 CLKDIV

原语名称

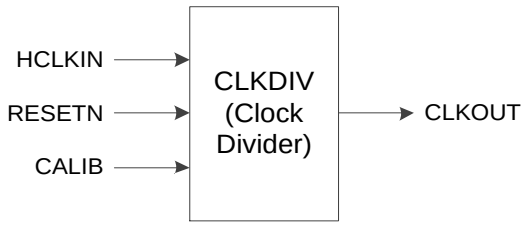
CLKDIV (Clock Divider)为时钟分频器，实现时钟频率调整。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 6-6 CLKDIV 端口示意图



功能描述

CLKDIV 为高速时钟分频模块，生成和输入时钟相位一致的分频时钟，用于 IO 逻辑。在 GW1N-1S、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1NS-4、GW1NS-4C、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9 下支持 2/3.5/4/5/8 分频，其他器件下支持 2/3.5/4/5 分频。

端口介绍

表 6-14 端口介绍

端口名	I/O	描述
HCLKIN	Input	时钟输入
RESETN	Input	异步复位输入，低电平有效
CALIB	Input	CALIB 输入，调整输出时钟
CLKOUT	Output	时钟输出

参数介绍

表 6-15 参数介绍

参数名	取值范围	默认值	描述
DIV_MODE	2, 3.5, 4, 5（8）	2	设置时钟分频系数
GSREN	false, true	false	启用全局复位 GSR

原语例化

Verilog 例化：

```
CLKDIV clkdiv_inst (
      .HCLKIN(hclkin),
      .RESETN(resetn),
      .CALIB(calib),
      .CLKOUT(clkout)
);
defparam clkdiv_inst.DIV_MODE=“3.5”;
defparam clkdiv_inst.GSREN=“false”;
```

Vhdl 例化：

```
COMPONENT CLKDIV
        GENERIC(
              DIV_MODE:STRING:="2";
              GSREN:STRING:="false"
              );
       PORT(
              HCLKIN:IN std_logic;
              RESETN:IN std_logic;
              CALIB:IN std_logic;
              CLKOUT:OUT std_logic
              );
END CONPONENT;
    uut:CLKDIV
        GENERIC MAP(
                    DIV_MODE=>"2",
                    GSREN=>"false"
                    )
            PORT MAP(
                    HCLKIN=>hclkin,
                    RESETN=>resetn,
                    CALIB=>calib,
                    CLKOUT=>clkout
);
```

6.6 CLKDIV2

原语名称

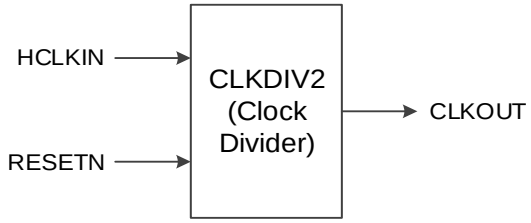
CLKDIV2 (Clock Divider)为时钟分频器，实现时钟的二分频调整。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 6-7 CLKDIV2 端口示意图



功能描述

CLKDIV2 为高速时钟分频模块，生成和输入时钟相位一致的 2 分频时钟。

端口介绍

表 6-16 端口介绍

端口名	I/O	描述
HCLKIN	Input	时钟输入
RESETN	Input	异步复位输入，低电平有效
CLKOUT	Output	时钟输出

参数介绍

表 6-17 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

原语例化

Verilog 例化：

```
CLKDIV2 clkdiv2_inst (
    .HCLKIN(hclkin),
    .RESETN(resetn),
    .CLKOUT(clkout)
);
defparam clkdiv2_inst.GSREN=“false”;
```

Vhdl 例化：

```
COMPONENT CLKDIV2
        GENERIC(
            GSREN:STRING:="false"
```

```
            );
        PORT(
            HCLKIN:IN std_logic;
            RESETN:IN std_logic;
            CLKOUT:OUT std_logic
            );
END CONPONENT;
    uut:CLKDIV2
        GENERIC MAP(
                GSREN=>"false"
                )
            PORT MAP(
                HCLKIN=>hclkin,
                RESETN=>resetn,
                CLKOUT=>clkout
);
```

6.7 DQCE

原语名称

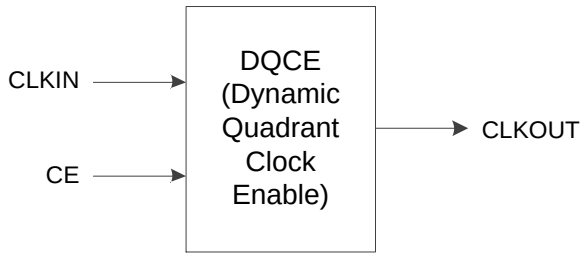
DQCE (Dynamic Quadrant Clock Enable) ，动态使能高云 FPGA 象限时钟。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 6-8 DQCE 端口示意图



功能描述

通过 DQCE 可动态打开/关闭 GCLK0~GCLK5。关闭 GCLK0~GCLK5 时钟，GCLK0~GCLK5 驱动的内部逻辑不再翻转，降低了器件的总体功耗。DQCE 正常工作，需要 CLKIN 信号至少有一次高电平到低电平的下降沿变化。

端口介绍

表 6-18 端口介绍

端口名	I/O	描述
CLKIN	Input	时钟输入
CE	Input	时钟使能输入，高电平有效
CLKOUT	Output	时钟输出

原语例化

Verilog 例化：

```
DQCE dqce_inst (
    .CLKIN(clkin),
    .CE(ce),
    .CLKOUT(clkout)
);
```

Vhdl 例化：

```
COMPONENT DQCE
    PORT(
        CLKOUT:OUT std_logic;
        CE:IN std_logic;
        CLKIN:IN std_logic
    );
END COMPONENT;
uut:DQCE
PORT MAP(
    CLKIN=>clkin,
    CLKOUT=>clkout,
    CE=>ce
);
```

6.8 DCS

原语名称

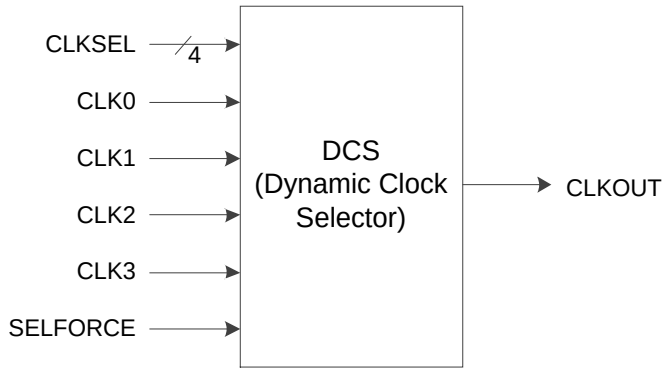
DCS (Dynamic Clock Select) 是动态时钟选择器，可动态选择象限时钟 GCLK6 和 GCLK7。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 6-9 DCS 端口示意图



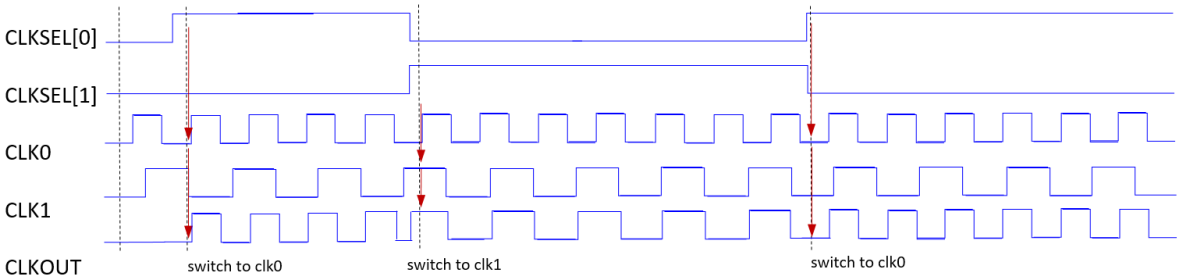
功能描述

每个象限的 GCLK6~GCLK7 由 DCS 控制，选择四个输入时钟中的一个作为全局时钟。内部逻辑可以通过 CRU 在四个时钟输入之间动态选择，输出不带毛刺的时钟。

DCS 存在两种时钟切换模式，分别是“Non-Glitchless”和“Glitchless”模式。

在 Non-Glitchless 模式下（输入 SELFORCE = ‘1’），DCS 的作用类似于常规多路复用器，仅通过 CLKSEL 信号切换时钟信号，允许输出上的毛刺，实际情况取决于切换的时间。Non-Glitchless 模式时序如图 6-10 所示，用 CLKSEL[3]~CLKSEL[0]分别对应选择 CLK3~CLK0，高电平有效，转换时序相同。

图 6-10 Non-Glitchless 模式时序图



在 Glitchless 无毛刺模式下（输入 SELFORCE = ‘0’），通过参数 DCS_MODE 设置模式，配置 CLKSEL 信号动态切换时钟信号，可以避免输出时钟上的毛刺。Glitchless 模式时序如图 6-11 到图 6-14 所示，用 CLKSEL[3]~CLKSEL[0]分别对应选择 CLK3~CLK0，转换时序相同。

图 6-11 DCS mode：RISING 时序图

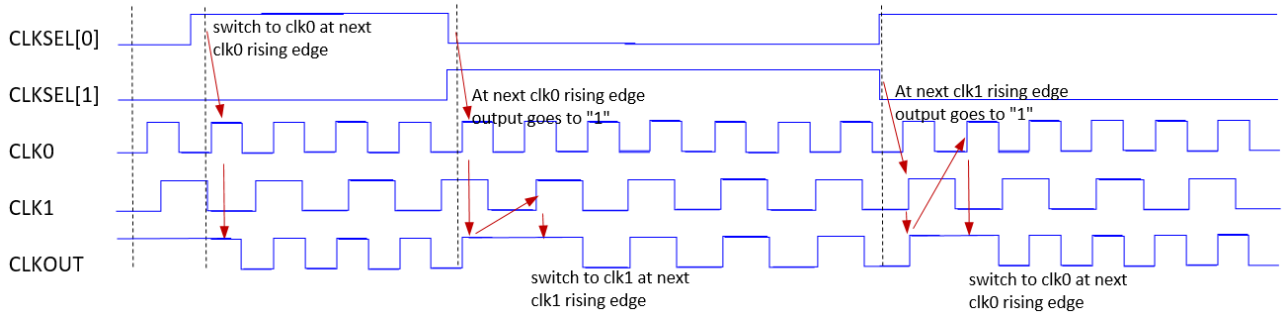


图 6-12 DCS mode：FALLING 时序图

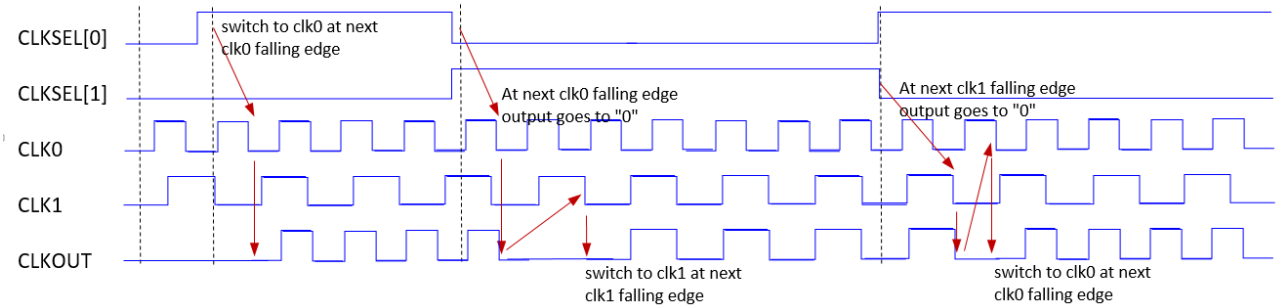


图 6-13 DCS mode：CLK0_GND 时序图

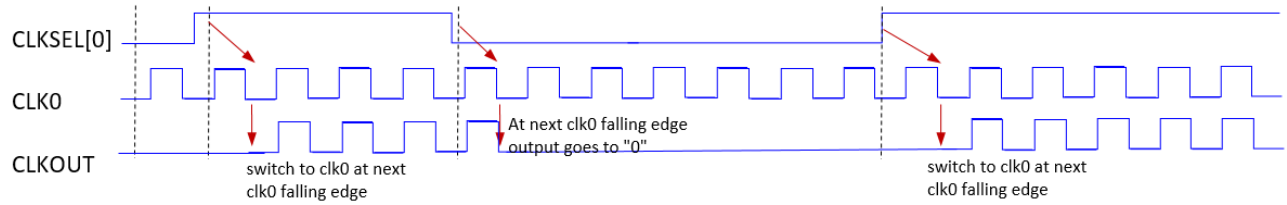
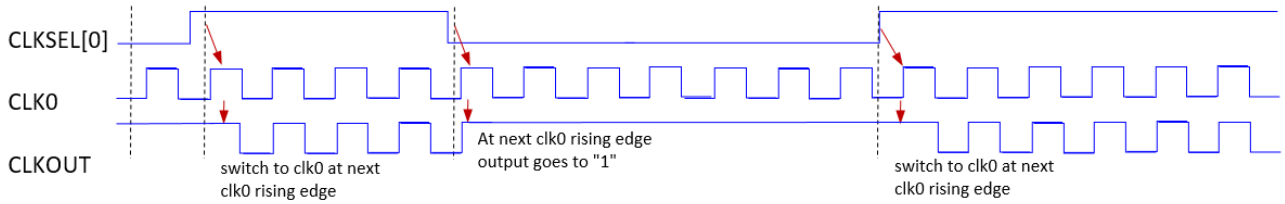


图 6-14 DCS mode：CLK0_VCC 时序图



端口介绍

表 6-19 端口介绍

端口名	I/O	描述
CLK0	Input	时钟输入 0
CLK1	Input	时钟输入 1
CLK2	Input	时钟输入 2
CLK3	Input	时钟输入 3
CLKSEL[3:0]	Input	时钟选择信号
SELFORCE	Input	强制模式选择 0：glitchless 模式 1：Non-glitchless 模式

端口名	I/O	描述
CLKOUT	Output	时钟输出

参数介绍

表 6-20 参数介绍

参数名	取值范围	默认值	描述
DCS_MODE	CLK0,CLK1,CLK2,CLK3, GND,VCC,RISING,FALLING, CLK0_GND,CLK1_GND, CLK2_GND,CLK3_GND, CLK0_VCC,CLK1_VCC, CLK2_VCC,CLK3_VCC	RISING	设置 DCS 模式

原语例化

Verilog 例化：

```
DCS dcs_inst (
      .CLK0(clk0),
      .CLK1(clk1),
      .CLK2(clk2),
      .CLK3(clk3),
      .CLKSEL(clksel[3:0]),
      .SELFORCE(selforce),
      .CLKOUT(clkout)
);
defparam dcs_inst.DCS_MODE=“RISING”;
```

Vhdl 例化：

```
COMPONENT DCS
        GENERIC(DCS_MODE:string:="RISING");
            PORT(
                    CLK0:IN std_logic;
                    CLK1:IN std_logic;
                    CLK2:IN std_logic;
                    CLK3:IN std_logic;
                    CLKSEL:IN std_logic_vector(3 downto 0);
                    SELFORCE:IN std_logic;
                    CLKOUT:OUT std_logic
                );
END COMPONENT;
uut:DCS
          GENERIC MAP(DCS_MODE=>"RISING")
          PORT MAP(
              CLK0=>clk0,
              CLK1=>clk1,
              CLK2=>clk2,
              CLK3=>clk3,
              CLKSEL=>clksel,
              SELFORCE=>selforce,
```

```
        CLKOUT=>clkout
    );
```

6.9 DQS

原语介绍

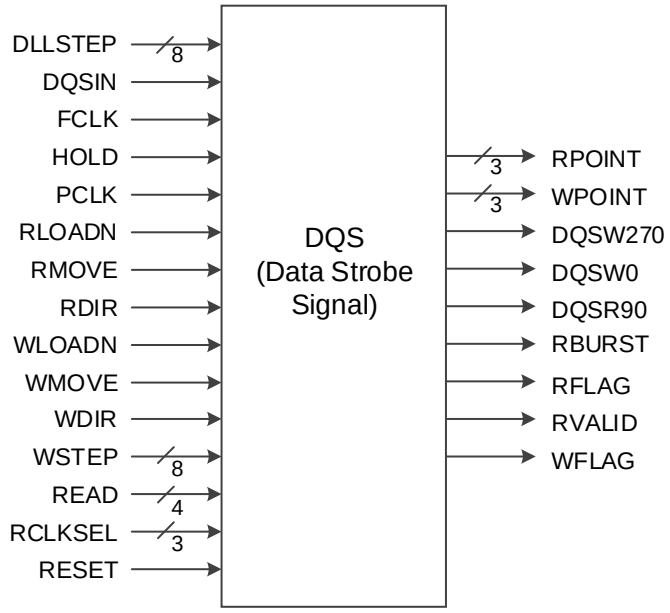
DQS（Bidirectional Data Strobe Circuit for DDR Memory）是 DDR 存储器接口双向数据选通脉冲电路。

适用器件

支持器件：GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 6-15 DQS 端口示意图



功能描述

DQS 是内存控制器 IP 的关键器件，主要用于调整 DQSIN 与 DQSR90、DQSW0 与 DQSW270 信号间的相位关系并完成写平衡、读校准。

端口介绍

表 6-21 端口介绍

端口名	I/O	描述
DLLSTEP[7:0]	input	DQS 延时步长控制输入，来自 DLL 模块
DQSIN	input	DQS 输入，来自 IO PAD
FCLK	input	快速时钟，可来自两个不同 FCLK 时钟树输出
HOLD	input	用于 DQS 写入，停止写入相关信号来同步输出时钟；用于 DQS 读取，来复位 FIFO 计数器
PCLK	input	主时钟，来自 PCLK 时钟树

端口名	I/O	描述
RDIR	input	调整 DDR 读取的延时方向 “0” 增加延时 “1” 减少延时
RLOADN	input	将 DDR 读取的最终延时步长复位至初始值，低电平有效
RMOVE	input	RMOVE 为下降沿时改变 DDR 读取的延时步长，每个脉冲改变一次
WDIR	input	调整 DDR 写入的延时方向 “0” 增加延时 “1” 减少延时
WLOADN	input	将 DDR 写入的最终延时步长复位至初始值，低电平有效
WMOVE	input	WMOVE 为下降沿时改变 DDR 写入的延时步长，每个脉冲改变一次
WSTEP[7:0]	input	用于 DDR 写均衡延时控制
READ[3:0]	input	READ 信号，用于 DDR 读模式
RCLKSEL[2:0]	input	选择读时钟源和极性控制
RESET	input	DQS 复位输入，高电平有效
RPOINT[2:0]	output	FIFO 控制读指针，作用于 IOLOGIC 的 RADDR，或通过绕线作用于用户逻辑
WPOINT[2:0]	output	FIFO 控制写指针，作用于 IOLOGIC 的 WADDR，或通过绕线作用于用户逻辑
DQSW0	output	PCLK/FCLK 0° 相移输出，可作用于 IOLOGIC 的 TCLK，或通过绕线作用于用户逻辑
DQSW270	output	PCLK/FCLK 270° 相移输出，可作用于 IOLOGIC 的 TCLK，或通过绕线作用于用户逻辑
DQSR90	output	DQSI 相移 90° 输出，可作用于 IOLOGIC 的 ICLK，或通过绕线作用于用户逻辑
RFLAG	output	READ 延时调整输出标志，用以表示读取延时调整 under-flow 或 over-flow
WFLAG	output	WRITE 延时调整输出标志，用以表示写入延时调整 under-flow 或 over-flow
RVALID	output	READ 模式数据有效标志
RBURST	output	READ 突发检测输出

参数介绍

表 6-22 参数介绍

参数名	取值范围	默认值	描述
FIFO_MODE_SEL	1’b0 , 1’b1	1’b0	FIFO 模式选择 1'b0: DDR memory 模式 1'b1: GDDR 模式
RD_PNTR	000,001,010,011,100,101,110,111	3'b000	FIFO 读指针设置
DQS_MODE	X1,X2_DDR2,X2_DDR3,X4,X2_DDR3_EXT	X1	DQS 模式选择
HWL	false,true	false	update0/1 时序关系控制 "false": update1 比 update0 提前一个

参数名	取值范围	默认值	描述
			周期； "true": update1 和 update0 时序相同
GSREN	false,true	false	启用全局复位 GSR

连接合法性规则

- DQS 的输入 DQSI 来自 IO PAD；
- DQS 的输入 DLLSTEP 来自 DLL 模块的输出 STEP；
- DQS 的输出 RPOINT 可连接至 IOLOGIC 的 RADDR，也可作用于用户逻辑；
- DQS 的输出 WPOINT 可连接至 IOLOGIC 的 WADDR，也可作用于用户逻辑；
- DQS 的输出 DQSR90 可连接至 IOLOGIC 的 ICLK，也可作用于用户逻辑；
- DQS 的输出 DQSW0/ DQSW270 可连接至 IOLOGIC 的 TCLK，也可作用于用户逻辑。

原语例化

Verilog 例化：

```
DQS uut (
        .DQSIN(dqs),
        .PCLK(pclk),
        .FCLK(fclk),
        .RESET(reset),
        .READ(read),
        .RCLKSEL(rsel),
        .DLLSTEP(step),
        .WSTEP(wstep),
        .RLOADN(1'b0),
        .RMOVE(1'b0),
        .RDIR(1'b0),
        .WLOADN(1'b0),
        .WMOVE(1'b0),
        .WDIR(1'b0),
        .HOLD(hold),
        .DQSR90(dqsr90),
        .DQSW0(dqsw0),
        .DQSW270(dqsw270),
        .RPOINT(rpoint),
        .WPOINT(wpoint),
        .RVALID(rvalid),
        .RBURST(rburst),
        .RFLAG(rflag),
        .WFLAG(wflag)
);
defparam uut.DQS_MODE = "X1";
defparam uut.FIFO_MODE_SEL = 1'b0;
```

```
defparam uut.RD_PNTR = 3'b001;
```

Vhdl 例化：

```
COMPONENT DQS
    GENERIC(
            FIFO_MODE_SEL:bit:='0';
            RD_PNTR : bit_vector:="000";
            DQS_MODE:string:="X1";
            HWL:string:="false";
            GSREN : string:="false"
     );
    PORT(
        DQSIN,PCLK,FCLK,RESET:IN std_logic;
        READ:IN std_logic_vector(3 downto 0);
        RCLKSEL:IN std_logic_vector(2 downto 0);
        DLLSTEP,WSTEP:IN std_logic_vector(7 downto 0);
        RLOADN,RMOVE,RDIR,HOLD:IN std_logic;
        WLOADN,WMOVE,WDIR:IN std_logic;
        DQSR90,DQSW0,DQSW270:OUT std_logic;
        RPOINT, WPOINT:OUT std_logic_vector(2 downto 0);
        RVALID,RBURST,RFLAG,WFLAG:OUT std_logic
     );
END COMPONENT;
uut:DQS
    GENERIC MAP(
                FIFO_MODE_SEL=>'0',
                RD_PNTR=>"000",
                DQS_MODE=>"X1",
                HWL=>"false",
                GSREN=>"false"
     )
      PORT MAP(
        DQSIN=>dqsin,
        PCLK=>pclk,
        FCLK=>fclk,
        RESET=>reset,
        READ=>read,
        RCLKSEL=>rclksel,
        DLLSTEP=>step,
        WSTEP=>wstep,
        RLOADN=>rloadn,
        RMOVE=>rmove,
        RDIR=>rdir,
        HOLD=>hold,
        WLOADN=>wloadn,
        WMOVE=>wmove,
        WDIR=>wdir,
        DQSR90=>dqsr90,
        DQSW0=>dqsw0,
        DQSW270=>dqsw270,
        RPOINT=>rpoint,
```

```
        WPOINT=>wpoint,
        RVALID=>rvalid,
        RBURST=>rburst,
        RFLAG=>rflag,
        WFLAG=>wflag
);
```

6.10 OSC

原语名称

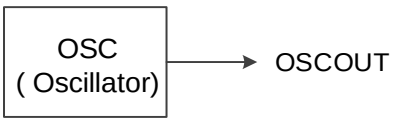
OSC(Oscillator)是片内晶振。

适用器件

支持器件：GW1N-2、GW1N-2B、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 6-16 OSC 端口示意图



功能描述

GOWIN FPGA 内嵌了一个可编程片内晶振，编程过程中为 MSPI 编程模式提供时钟源，还可以为用户设计提供时钟源，通过配置工作参数，可以获得多达 64 种时钟频率。

器件输出时钟频率可以通过如下公式计算得到：

$f_{CLKOUT} = f_{osc}/\mathrm{FREQ\_DIV}$；

其中f_{osc}为 OSC 振荡频率，GW1N-2、GW1N-2B、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B 为 210MHz，其它器件为 250MHz，除数 FREQ_DIV 为配置参数，范围为 2~128 的偶数。

端口介绍

表 6-23 端口介绍

端口名	I/O	描述
OSCOUT	output	OSC 输出时钟

参数介绍

表 6-24 参数介绍

参数名	取值范围	默认值	描述
FREQ_DIV	2~128(even)	100	OSC 分频系数设置
DEVICE	GW1N-2、GW1N-2B、GW1N-4、GW1N-4B、	GW1N-2(GW1N 系列)	器件选择

参数名	取值范围	默认值	描述
	GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。	GW2A-18(GW2A 系列)	

原语例化

Verilog 例化：

```
OSC uut(
    .OSCOUT(oscout)
    );
defparam uut.FREQ_DIV=100;
defparam uut.DEVICE="GW2A-18";
```

Vhdl 例化：

```
COMPONENT OSC
        GENERIC(
                    FREQ_DIV:integer:=100;
                    DEVICE:string:="GW2A-18"
         );
        PORT(OSCOUT:OUT STD_LOGIC);
END COMPONENT;
uut:OSC
      GENERIC MAP(
                    FREQ_DIV=>100,
                    DEVICE=>"GW2A-18"
       )
      PORT MAP(OSCOUT=>oscout);
```

6.11 OSCZ

原语名称

OSCZ(Oscillator)是带有动态关闭 OSC 功能的片内晶振。

适用器件

支持器件：GW1NZ-1、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C。

端口示意图

图 6-17 OSCZ 端口示意图

功能描述

GW1NZ 系列 FPGA 产品内嵌了一个可编程的片内晶振，时钟精度可达±5%，支持动态打开/关闭 OSC 功能。编程过程中为 MSPI 编程模式提供

时钟源，还可以为用户设计提供时钟源，通过配置工作参数，可以获得多达 64 种时钟频率。输出时钟频率可以通过如下公式计算得到：

$f_{CLKOUT} = 250MHz/\text{FREQ\_DIV}$；

其中除数 FREQ_DIV 为配置参数，范围为 2~128 的偶数。

端口介绍

表 6-25 端口介绍

端口名	I/O	描述
OSCEN	input	OSC 使能信号
OSCOUT	output	OSC 时钟输出

参数介绍

表 6-26 参数介绍

参数名	取值范围	默认值	描述
FREQ_DIV	2~128(even)	100	OSC 分频系数设置

原语例化

Verilog 例化：

```
OSCZ uut(
    .OSCOUT(oscout)，
    .OSCEN(oscen)
    );
defparam uut.FREQ_DIV=100;
```

Vhdl 例化：

```
COMPONENT OSCZ
        GENERIC(
                FREQ_DIV:integer:=100;
         );
        PORT(
                OSCOUT:OUT STD_LOGIC;
                OSCEN:IN std_logic
              );
END COMPONENT;
uut:OSCZ
    GENERIC MAP(
                FREQ_DIV=>100,
     )
    PORT MAP(
                OSCOUT=>oscout,
                OSCEN(oscen)
                );
```

6.12 OSCF

原语名称

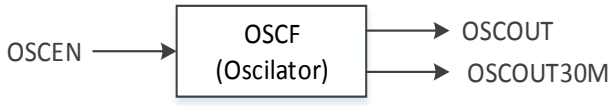
OSCF(Oscillator with CLKOUT30M and Dynamic OSC Enable)是带30M 输出时钟和动态使能的片内晶振。

适用器件

支持器件：GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C。

端口示意图

图 6-18 OSCF 端口示意图



功能描述

GW1NS 系列 FPGA 产品内嵌了一个可编程的片内晶振，时钟精度可达±5%，支持动态打开/关闭 OSC 功能。编程过程中为 MSPI 编程模式提供时钟源，还可以为用户设计提供时钟源，通过配置工作参数，可以获得多达 64 种时钟频率。输出时钟频率可以通过如下公式计算得到：

$f_{CLKOUT} = 240MHz/\text{FREQ\_DIV}$;

其中除数 FREQ_DIV 为配置参数，范围为 2~128 的偶数。

端口介绍

表 6-27 端口介绍

端口名	I/O	描述
OSCEN	input	OSC 使能信号
OSCOUT	output	OSC 时钟输出
OSCOUT30M	output	OSC 30M 时钟输出，需连接至 FLASH128K 的 PCLK

参数介绍

表 6-28 参数介绍

参数名	取值范围	默认值	描述
FREQ_DIV	2~128(even)	96	OSC 分频系数设置

连接合法性规则

OSC 的输出 OSCOUT30M 需连接至 FLASH128K 的 PCLK。

原语例化

Verilog 例化：

```
OSCF uut(
```

```
    .OSCOUT(oscout)，
    .OSCOUT30M(oscout30m),
    .OSCEN(oscen)
    );
  defparam uut.FREQ_DIV=96;
```

Vhdl 例化：

```
  COMPONENT OSCF
        GENERIC(
                FREQ_DIV:integer:=96;
         );
        PORT(
              OSCOUT:OUT std_logic;
              OSCOUT30M:OUT std_logic;
              OSCEN:IN std_logic
              );
  END COMPONENT;
  uut:OSCF
      GENERIC MAP(FREQ_DIV=>96)
      PORT MAP(
                OSCOUT=>oscout,
                OSCOUT30M=>oscout30m,
                OSCEN(oscen)
                );
```

6.13 OSCH

原语名称

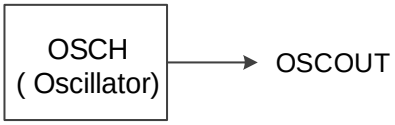
OSCH(Oscillator)是片内晶振。

适用器件

支持器件：GW1N-1、GW1N-1S。

端口示意图

图 6-19 OSCH 端口示意图



功能描述

OSCH 可编程片内晶振，编程过程中为 MSPI 编程模式提供时钟源，还可以为用户设计提供时钟源，通过配置工作参数，可以获得多达 64 种时钟频率。输出时钟频率可以通过如下公式计算得到：

$f_{CLKOUT} = 240MHz/\text{FREQ\_DIV}$；

其中除数 FREQ_DIV 为配置参数，范围为 2~128 的偶数。

Port 介绍

表 6-29 Port 介绍

端口名	I/O	描述
OSCOUT	output	OSC 时钟输出

参数介绍

表 6-30 参数介绍

参数名	取值范围	默认值	描述
FREQ_DIV	2~128(even)	100	OSC 分频系数设置

原语例化

Verilog 例化：

```
OSCH uut(
    .OSCOUT(oscout)
    );
defparam uut.FREQ_DIV=100;
```

Vhdl 例化：

```
COMPONENT OSCH
        GENERIC(
                FREQ_DIV:integer:=100;
         );
        PORT(OSCOUT:OUT STD_LOGIC);
END COMPONENT;
uut:OSCH
    GENERIC MAP(
                FREQ_DIV=>100,
     )
    PORT MAP(OSCOUT=>oscout);
```

6.14 DHCEN

原语名称

DHCEN（Dynamic HCLK Clock Eanble with Inverted Gate）可动态的打开/关闭 HCLK 高速时钟信号，低电平时导通。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 6-20 DHCEN 端口示意图

端口介绍

表 6-31 端口介绍

端口名	I/O	描述
CLKIN	input	时钟输入
CE	input	时钟使能输入，低电平有效
CLKOUT	output	时钟输出

原语例化

Verilog 例化：

```
DHCEN dhcen_inst (
    .CLKIN(clkin),
    .CE(ce),
    .CLKOUT(clkout)
);
```

Vhdl 例化：

```
COMPONENT DHCEN
    PORT(
        CLKOUT:OUT std_logic;
        CE:IN std_logic;
        CLKIN:IN std_logic
    );
END COMPONENT;
uut:DHCEN
PORT MAP(
    CLKIN=>clkin,
    CLKOUT=>clkout,
    CE=>ce
);
```

6.15 BUFG

原语名称

BUFG(Global Clock Buffer)是全局时钟缓冲器。

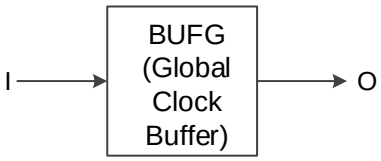
适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、

GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 6-21 BUFG 端口示意图



端口介绍

表 6-32 端口介绍

端口名	I/O	描述
O	output	时钟输出
I	input	时钟输入

原语例化

Verilog 例化：

```
BUFG uut(
    .O(o),
    .I(i)
    );
```

Vhdl 例化：

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

6.16 BUFS

原语名称

BUFS(Long Wire Clock Buffer)是长线时钟缓冲器。

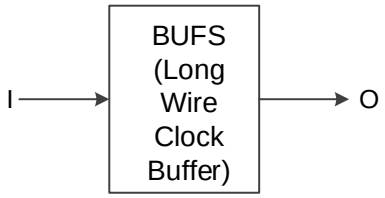
适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、

GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 6-22 BUFS 端口示意图



端口介绍

表 6-33 Port 介绍

端口名	I/O	描述
O	output	时钟输出
I	input	时钟输入

原语例化

Verilog 例化：

```
BUFS uut(
    .O(o),
    .I(i)
    );
```

Vhdl 例化：

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

7 User Flash

7.1 FLASH96K

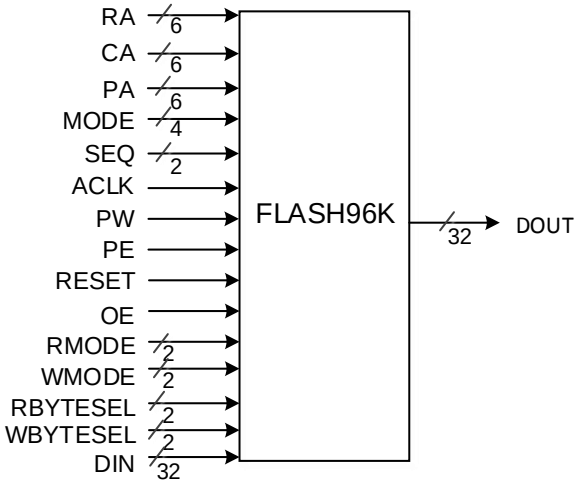
原语介绍

FLASH96K（96Kbit User Flash）的存储空间为 96K bit。寄存器的宽度和深度是固定的，不可对其进行配置。其宽度为 4 Byte（32 bit），地址深度为 3k，具有非易失性和断电保存功能，但不具有 BSRAM 的初始值功能。

支持器件：GW1N-1、GW1N-1S。

结构框图

图 7-1 FLASH96K 结构框图



Port 介绍

表 7-1 Port 介绍

Port Name	I/O	Description
DOUT[31:0]	Output	Data Output
DIN[31:0]	Input	Data Input
RA[5:0]	Input	Row Address
CA[5:0]	Input	Column Address

Port Name	I/O	Description
PA[5:0]	Input	Page latch Address
MODE[3:0]	Input	Operation mode select
SEQ[1:0]	Input	NV operation sequence control
ACLK	Input	Synchronous clock for read and write operation
PW	Input	Write page latch clock
RESET	Input	Macro reset
PE	Input	Pump enable
OE	Input	Output enable
RMODE[1:0]	Input	Read out bit width select
WMODE[1:0]	Input	Write in bit width select
RBYTESEL[1:0]	Input	Read data Byte address
WBYTESEL[1:0]	Input	Write data Byte address

原语例化

Verilog 例化：

```
FLASH96K flash96k_inst(
        .RA(ra[5:0]),
        .CA(ca[5:0]),
        .PA(pa[5:0]),
        .MODE(mode[3:0]),
        .SEQ(seq[1:0]),
        .ACLK(aclk),
        .PW(pw),
        .RESET(reset),
        .PE(pe),
        .OE(oe),
        .RMODE(rmode[1:0]),
        .WMODE(wmode[1:0]),
        .RBYTESEL(rbytesel[1:0]),
        .WBYTESEL(wbytesel[1:0]),
        .DIN(din[31:0]),
        .DOUT(dout[31:0])
);
```

Vhdl 例化：

```
COMPONENT FLASH96K
        PORT(
            RA:IN std_logic_vector(5 downto 0);
            CA:IN std_logic_vector(5 downto 0);
            PA:IN std_logic_vector(5 downto 0);
            MODE:IN std_logic_vector(3 downto 0);
            SEQ:IN std_logic_vector(1 downto 0);
            ACLK:IN std_logic;
            PW:IN std_logic;
            RESET:IN std_logic;
```

```
            PE:IN std_logic;
            OE:IN std_logic;
            RMODE:IN std_logic_vector(1 downto 0);
            WMODE:IN std_logic_vector(1 downto 0);
            RBYTESEL:IN std_logic_vector(1 downto 0);
            WBYTESEL:IN std_logic_vector(1 downto 0);
            DIN:IN std_logic_vector(31 downto 0);
            DOUT:OUT std_logic_vector(31 downto 0)
        );
END COMPONENT;
uut: FLASH96K
        PORT MAP (
            RA=>ra,
            CA=>ca,
            PA=>pa,
            MODE=>mode,
            SEQ=>seq,
            RESET=>reset,
            ACLK=>aclk,
            PW=>pw,
            PE=>pe,
            OE=>oe,
            RMODE=>rmode,
            WMODE=>wmode,
             RBYTESEL=>rbytesel,
            WBYTESEL=> wbytesel,
            DIN=>din,
            DOUT=>dout
    );
```

7.2 FLASH64KZ

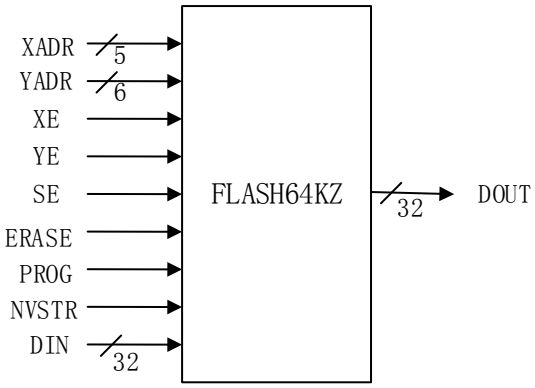
原语介绍

FLASH64KZ（64Kbit User Flash）的存储空间为 64K bit。寄存器的宽度和深度是固定的，不可对其进行配置。具有非易失性和断电保存功能，但不具有 BSRAM 的初始值功能。

支持器件：GW1NZ-1。

结构框图

图 7-2 FLASH64KZ 结构框图



Port 介绍

表 7-2 Port 介绍

Port Name	I/O	Description
DOUT[31:0]	Output	Data Output
DIN[31:0]	Input	Data Input
XADR[4:0]	Input	X address input
YADR[5:0]	Input	Y address input
XE	Input	X address enable
YE	Input	Y address enable
SE	Input	Sense amplifier enable
ERASE	Input	Defines erase cycle
PROG	Input	Defines program cycle
NVSTR	Input	Defines non-volatile store cycle

原语例化

Verilog 例化：

```
FLASH64KZ flash64kz_inst(
    .XADR(xadr[4:0]),
    .YADR(yadr[5:0]),
    .XE(xe),
    .YE(ye),
    .SE(se),
    .ERASE(erase),
    .PROG(prog),
    .NVSTR(nvstr),
    .DIN(din[31:0]),
    .DOUT(dout[31:0])
);
```

Vhdl 例化：

```
COMPONENT FLASH64KZ
```

```
        PORT(
            XADR:IN std_logic_vector(4 downto 0);
            YADR:IN std_logic_vector(5 downto 0);
            XE:IN std_logic;
            YE:IN std_logic;
            SE:IN std_logic;
            ERASE:IN std_logic;
            PROG:IN std_logic;
            NVSTR:IN std_logic;
            DIN:IN std_logic_vector(31 downto 0);
            DOUT:OUT std_logic_vector(31 downto 0)
        );
END COMPONENT;
uut: FLASH64KZ
        PORT MAP (
            XADR=>xadr,
            YADR=>yadr,
            XE=>xe,
            YE=>ye,
            SE=>se,
            ERASE=>erase,
            PROG=>prog,
            NVSTR=>nvstr,
            DIN=>din,
            DOUT=>dout
        );
```

7.3 FLASH64K

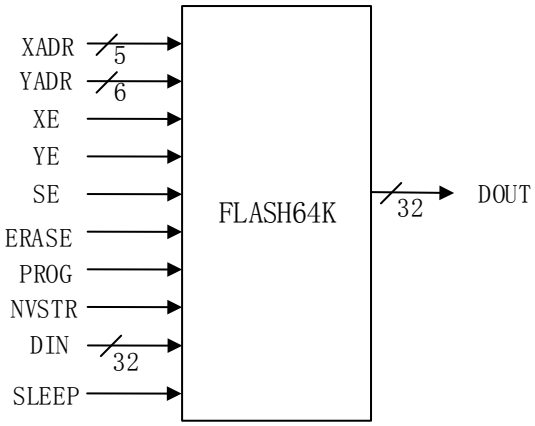
原语介绍

FLASH64K（64Kbit User Flash）的存储空间为 64K bit。寄存器的宽度和深度是固定的，不可对其进行配置。具有非易失性和断电保存功能，但不具有 BSRAM 的初始值功能。

支持器件：GW1NZ-ZV1FN32I2、GW1NZ-ZV1FN32I3、GW1NZ-ZV1CS16I2、GW1NZ-ZV1CS16I3。

结构框图

图 7-3 FLASH64K 结构框图



Port 介绍

表 7-3 Port 介绍

Port Name	I/O	Description
DOUT[31:0]	Output	Data Output
DIN[31:0]	Input	Data Input
XADR[4:0]	Input	X address input
YADR[5:0]	Input	Y address input
XE	Input	X address enable
YE	Input	Y address enable
SE	Input	Sense amplifier enable
ERASE	Input	Defines erase cycle
PROG	Input	Defines program cycle
NVSTR	Input	Defines non-volatile store cycle
SLEEP	Input	Sleep mode enable, active high

原语例化

Verilog 例化：

```
FLASH64K flash64k_inst(
    .XADR(xadr[4:0]),
    .YADR(yadr[5:0]),
    .XE(xe),
    .YE(ye),
    .SE(se),
    .ERASE(erase),
    .PROG(prog),
    .NVSTR(nvstr),
    .DIN(din[31:0]),
    .SLEEP(sleep),
    .DOUT(dout[31:0])
```

```
    );
```

Vhdl 例化：

```
COMPONENT FLASH64K
        PORT(
            XADR:IN std_logic_vector(4 downto 0);
            YADR:IN std_logic_vector(5 downto 0);
            XE:IN std_logic;
            YE:IN std_logic;
            SE:IN std_logic;
            ERASE:IN std_logic;
            PROG:IN std_logic;
            NVSTR:IN std_logic;
            DIN:IN std_logic_vector(31 downto 0);
            SLEEP:IN std_logic;
            DOUT:OUT std_logic_vector(31 downto 0)
        );
END COMPONENT;
uut: FLASH64K
        PORT MAP (
            XADR=>xadr,
            YADR=>yadr,
            XE=>xe,
            YE=>ye,
            SE=>se,
            ERASE=>erase,
            PROG=>prog,
            NVSTR=>nvstr,
            DIN=>din,
            SLEEP=>sleep,
            DOUT=>dout
);
```

7.4 FLASH128K

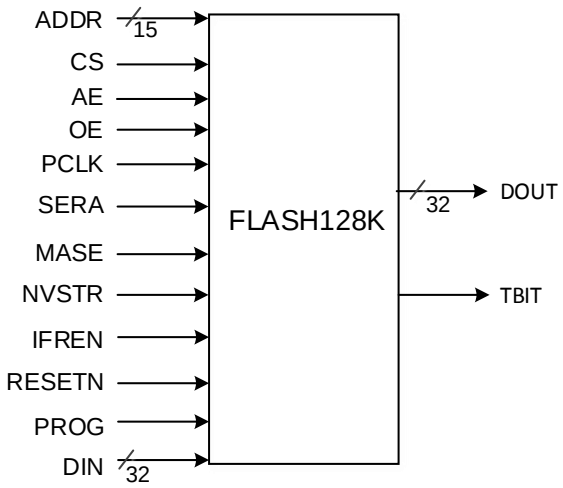
原语介绍

FLASH128K（128KByte Embedded Flash）的存储空间为 128K Byte。寄存器的宽度和深度是固定的，不可对其进行配置。具有非易失性和断电保存功能，但不具有 BSRAM 的初始值功能。

支持器件：GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C。

结构框图

图 7-4 FLASH128K 结构框图



Port 介绍

表 7-4 Port 介绍

Port Name	I/O	Description
DOUT[31:0]	Output	Data Output
TBIT	Output	Indicator of write or erase
DIN[31:0]	Input	Data Input
ADDR[14:0]	Input	Address Input
CS	Input	Chip enable
AE	Input	Address enable
OE	Input	Output enable
PCLK	Input	Clock input
PROG	Input	Defines program cycle
SERA	Input	Sector erase signal
MASE	Input	Chip erase signal
NVSTR	Input	Defines non-volatile store cycle
IFREN	Input	Flash IP information page Selection
RESETN	Input	Power On Reset Input

原语例化

Verilog 例化：

```
FLASH128K flash128k_inst(
    .ADDR(addr[14:0]),
    .CS(cs),
    .AE(ae),
    .OE(oe),
    .PCLK(pclk),
```

```
        .PROG(prog),
        .SERA(sera),
        .MASE(mase),
        .NVSTR(nvstr),
        .IFREN(ifren),
        .RESETN(resetn),
        .DIN(din[31:0]),
        .DOUT(dout[31:0]),
        .TBIT(tbit)
    );
```

Vhdl 例化：

```
    COMPONENT FLASH128K
            PORT(
                DIN:IN std_logic_vector(31 downto 0);
                ADDR:IN std_logic_vector(14 downto 0);
                CS:IN std_logic;
                AE:IN std_logic;
                OE:IN std_logic;
                PCLK:IN std_logic;
                PROG:IN std_logic;
                SERA:IN std_logic;
                MASE:IN std_logic;
                NVSTR:IN std_logic;
                IFREN:IN std_logic;
                RESETN:IN std_logic;
                DOUT:OUT std_logic_vector(31 downto 0);
                TBIT:OUT std_logic;
            );
    END COMPONENT;
    uut: FLASH128K
            PORT MAP (
                DIN=>din,
                ADDR=>addr,
                CS=>cs,
                AE=>ae,
                OE=>oe,
                PCLK=>pclk,
                PROG=>prog,
                SERA=>sera,
                MASE=>mase,
                NVSTR=>nvstr,
                IFREN=>ifren,
                RESETN=>resetn,
                DOUT=>dout,
                TBIT=>tbit
            );
```

7.5 FLASH256K

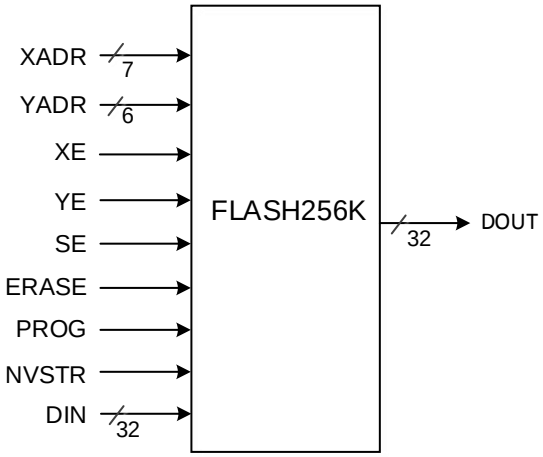
原语介绍

FLASH256K（256Kbit User Flash）的存储空间为 256K bit。寄存器的宽度和深度是固定的，不可对其进行配置。具有非易失性和断电保存功能，但不具有 BSRAM 的初始值功能。

支持器件： GW1N-2、GW1N-2B、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C。

结构框图

图 7-5 FLASH256K 结构框图



Port 介绍

表 7-5 Port 介绍

Port Name	I/O	Description
DOUT[31:0]	Output	Data Output
DIN[31:0]	Input	Data Input
XADR[6:0]	Input	X address input
YADR[5:0]	Input	Y address input
XE	Input	X address enable
YE	Input	Y address enable
SE	Input	Sense amplifier enable
PROG	Input	Defines program cycle
ERASE	Input	Defines erase cycle
NVSTR	Input	Defines non-volatile store cycle

原语例化

Verilog 例化：

```
FLASH256K flash256k_inst(
    .XADR(xadr[6:0]),
```

```
        .YADR(yadr[5:0]),
        .XE(xe),
        .YE(ye),
        .SE(se),
        .ERASE(erase),
        .PROG(prog),
        .NVSTR(nvstr),
        .DIN(din[31:0]),
        .DOUT(dout[31:0])
    );
```

Vhdl 例化：

```
    COMPONENT FLASH256K
            PORT(
                DIN:IN std_logic_vector(31 downto 0);
                XADR:IN std_logic_vector(6 downto 0);
                YADR:IN std_logic_vector(5 downto 0);
                XE:IN std_logic;
                YE:IN std_logic;
                SE:IN std_logic;
                ERASE:IN std_logic;
                PROG:IN std_logic;
                NVSTR:IN std_logic;
                DOUT:OUT std_logic_vector(31 downto 0)
        );
    END COMPONENT;
    uut: FLASH256K
            PORT MAP (
                DIN=>din,
                XADR=>xadr,
                YADR=>yadr,
                XE=>xe,
                YE=>ye,
                SE=>se,
                ERASE=>erase,
                PROG=>prog,
                NVSTR=>nvstr,
                DOUT=>dout
            );
```

7.6 FLASH608K

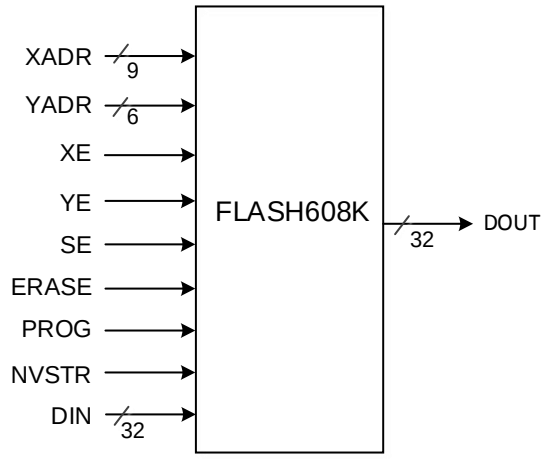
原语介绍

FLASH608K（608Kbit Users Flash）的存储空间为 608K bit。寄存器的宽度和深度是固定的，不可对其进行配置。具有非易失性和断电保存功能，但不具有 BSRAM 的初始值功能。

支持器件：GW1N-6、GW1N-9、GW1NR-9、。

结构框图

图 7-6 FLASH608K 结构框图



Port 介绍

表 7-6 Port 介绍

Port Name	I/O	Description
DOUT[31:0]	Output	Data Output
DIN[31:0]	Input	Data Input
XADR[8:0]	Input	X address input
YADR[5:0]	Input	Y address input
XE	Input	X address enable
YE	Input	Y address enable
SE	Input	Sense amplifier enable
PROG	Input	Defines program cycle
ERASE	Input	Defines erase cycle
NVSTR	Input	Defines non-volatile store cycle

原语例化

Verilog 例化：

```
FLASH608K flash608k_inst(
    .XADR(xadr[8:0]),
    .YADR(yadr[5:0]),
    .XE(xe),
    .YE(ye),
    .SE(se),
    .ERASE(erase),
    .PROG(prog),
    .NVSTR(nvstr),
    .DIN(din[31:0]),
    .DOUT(dout[31:0])
);
```

Vhdl 例化：

```
COMPONENT FLASH608K
        PORT(
            DIN:IN std_logic_vector(31 downto 0);
            XADR:IN std_logic_vector(8 downto 0);
            YADR:IN std_logic_vector(5 downto 0);
            XE:IN std_logic;
            YE:IN std_logic;
            SE:IN std_logic;
            ERASE:IN std_logic;
            PROG:IN std_logic;
            NVSTR:IN std_logic;
            DOUT:OUT std_logic_vector(31 downto 0)
    );
END COMPONENT;
uut: FLASH608K
        PORT MAP (
            DIN=>din,
            XADR=>xadr,
            YADR=>yadr,
            XE=>xe,
            YE=>ye,
            SE=>se,
            ERASE=>erase,
            PROG=>prog,
            NVSTR=>nvstr,
            DOUT=>dout
        );
```

8 EMPU

8.1 MCU

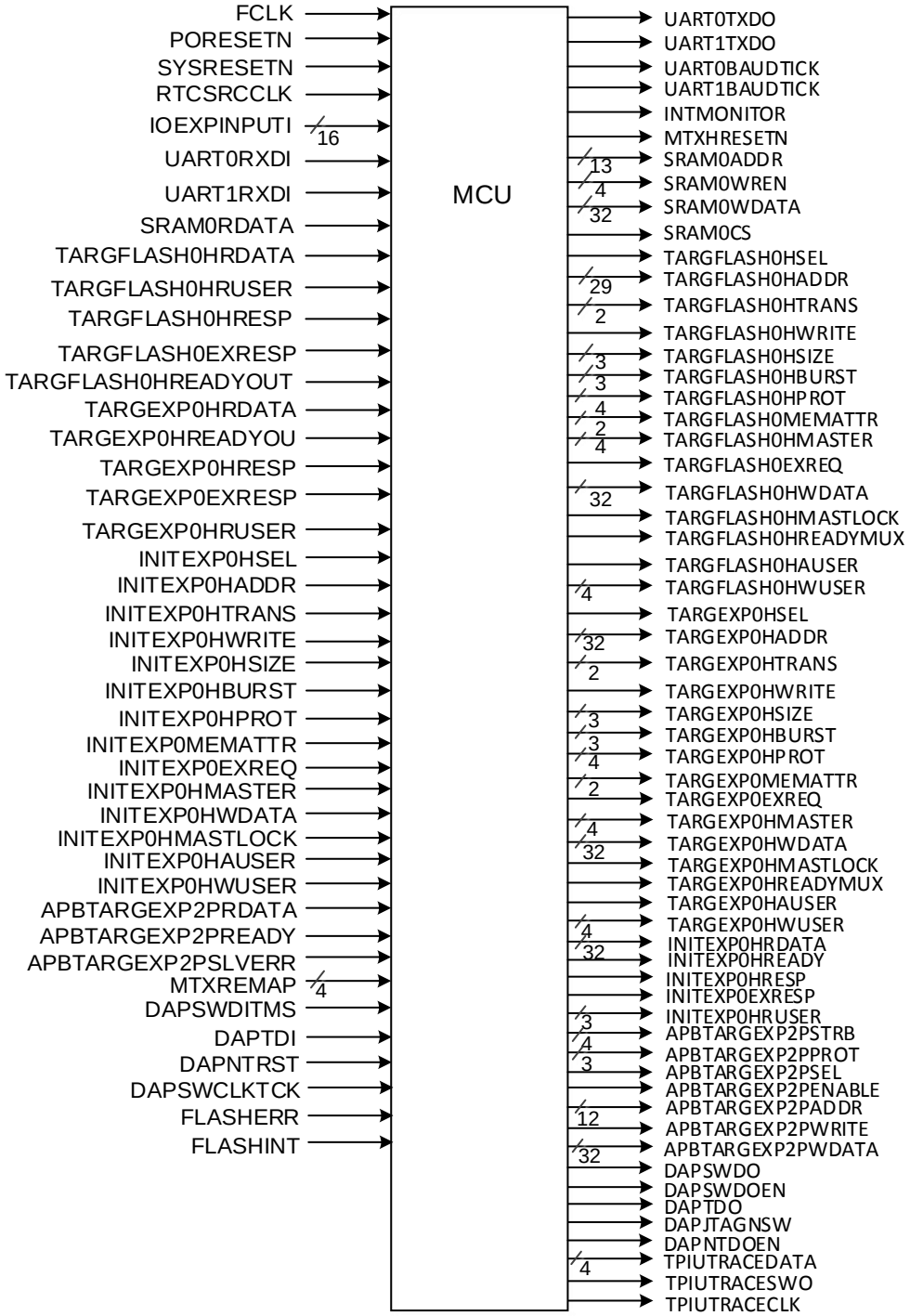
原语介绍

MCU(ARM Cortex-M3 Microcontroller Unit)是一款基于 ARM Cortex-M3 的微处理器。采用了 32 位 AHB/APB 的总线模式。其内部实现了 2 个 UART、2 个 Timer 和 Watchdog 的功能。并且对外提供 16 位 GPIO、2 个 UART、JTAG、2 个 User Interrupt 接口。以及 AHB Flash 读取接口、AHB Sram 读写接口。同时对外还提供了 2 个 AHB 总线扩展接口和 1 个 APB 总线扩展接口。

支持器件：GW1NS-2C、GW1NSR-2C、GW1NSE-2C。

结构框图

图 8-1 MCU 结构框图



Port 介绍

表 8-1 Port 介绍

Port Name	I/O	Description
FCLK	input	Free running clock
PORESETN	input	Power on reset
SYSRESETN	input	System reset

Port Name	I/O	Description
RTCSRCCLK	input	Used to generate RTC clock
IOEXPINPUTI[15:0]	input	IOEXPINPUTI
UART0RXDI	input	UART0RXDI
UART1RXDI	input	UART1RXDI
SRAM0RDATA[31:0]	input	SRAM Read data bus
TARGFLASH0HRDATA[31:0]	input	TARGFLASH0, HRDATA
TARGFLASH0HRUSER[2:0]	input	TARGFLASH0, HRUSER
TARGFLASH0HRESP	input	TARGFLASH0, HRESP
TARGFLASH0EXRESP	input	TARGFLASH0, EXRESP
TARGFLASH0HREADYOUT	input	TARGFLASH0, EXRESP
TARGEXP0HRDATA[31:0]	input	TARGEXP0, HRDATA
TARGEXP0HREADYOUT	input	TARGEXP0, HREADY
TARGEXP0HRESP	input	TARGEXP0, HRESP
TARGEXP0EXRESP	input	TARGEXP0, EXRESP
TARGEXP0HRUSER[2:0]	input	TARGEXP0, HRUSER
INITEXP0HSEL	input	INITEXP0, HSELx
INITEXP0HADDR[31:0]	input	INITEXP0, HADDR
INITEXP0HTRANS[1:0]	input	INITEXP0, HTRANS
INITEXP0HWRITE	input	INITEXP0, HWRITE
INITEXP0HSIZE[2:0]	input	INITEXP0, HSIZE
INITEXP0HBURST[2:0]	input	INITEXP0, HBURST
INITEXP0HPROT[3:0]	input	INITEXP0, HPROT
INITEXP0MEMATTR[1:0]	input	INITEXP0, MEMATTR
INITEXP0EXREQ	input	INITEXP0, EXREQ
INITEXP0HMASTER[3:0]	input	INITEXP0, HMASTER
INITEXP0HWDATA[31:0]	input	INITEXP0, HWDATA
INITEXP0HMASTLOCK	input	INITEXP0, HMASTLOCK
INITEXP0HAUSER	input	INITEXP0, HAUSER
INITEXP0HWUSER[3:0]	input	INITEXP0, HWUSER
APBTARGEXP2PRDATA[31:0]	input	APBTARGEXP2, PRDATA
APBTARGEXP2PREADY	input	APBTARGEXP2, PREADY
APBTARGEXP2PSLVERR	input	APBTARGEXP2, PSLVERR
MTXREMAP[3:0]	input	The MTXREMAP signals control the remapping of the boot memory range.
DAPSWDITMS	input	Debug TMS
DAPTDI	input	Debug TDI
DAPNTRST	input	Test reset
DAPSWCLKTCK	input	Test clock / SWCLK
FLASHERR	input	Output clock, used by the TPA to sample the other pins
FLASHINT	input	Output clock, used by the TPA to sample the other pins
IOEXPOUTPUTO[15:0]	output	IOEXPOUTPUTO
IOEXPOUTPUTENO[15:0]	output	IOEXPOUTPUTENO
UART0TXDO	output	UART0TXDO
UART1TXDO	output	UART1TXDO
UART0BAUDTICK	output	UART0BAUDTICK
UART1BAUDTICK	output	UART1BAUDTICK

Port Name	I/O	Description
INTMONITOR	output	INTMONITOR
MTXHRESETN	output	SRAM/Flash Chip reset
SRAM0ADDR[12:0]	output	SRAM address
SRAM0WREN[3:0]	output	SRAM Byte write enable
SRAM0WDATA[31:0]	output	SRAM Write data
SRAM0CS	output	SRAM Chip select
TARGFLASH0HSEL	output	TARGFLASH0, HSELx
TARGFLASH0HADDR[28:0]	output	TARGFLASH0, HADDR
TARGFLASH0HTRANS[1:0]	output	TARGFLASH0, HTRANS
TARGFLASH0HWRITE	output	TARGFLASH0, HWRITE
TARGFLASH0HSIZE[2:0]	output	TARGFLASH0, HSIZE
TARGFLASH0HBURST[2:0]	output	TARGFLASH0, HBURST
TARGFLASH0HPROT[3:0]	output	TARGFLASH0, HPROT
TARGFLASH0MEMATTR[1:0]	output	TARGFLASH0, MEMATTR
TARGFLASH0EXREQ	output	TARGFLASH0, EXREQ
TARGFLASH0HMASTER[3:0]	output	TARGFLASH0, HMASTER
TARGFLASH0HWDATA[31:0]	output	TARGFLASH0, HWDATA
TARGFLASH0HMASTLOCK	output	TARGFLASH0, HMASTLOCK
TARGFLASH0HREADYMUX	output	TARGFLASH0, HREADYOUT
TARGFLASH0HAUSER	output	TARGFLASH0, HAUSER
TARGFLASH0HWUSER[3:0]	output	TARGFLASH0, HWUSER
TARGEXP0HSEL	output	TARGEXP0, HSELx
TARGEXP0HADDR[31:0]	output	TARGEXP0, HADDR
TARGEXP0HTRANS[1:0]	output	TARGEXP0, HTRANS
TARGEXP0HWRITE	output	TARGEXP0, HWRITE
TARGEXP0HSIZE[2:0]	output	TARGEXP0, HSIZE
TARGEXP0HBURST[2:0]	output	TARGEXP0, HBURST
TARGEXP0HPROT[3:0]	output	TARGEXP0, HPROT
TARGEXP0MEMATTR[1:0]	output	TARGEXP0, MEMATTR
TARGEXP0EXREQ	output	TARGEXP0, EXREQ
TARGEXP0HMASTER[3:0]	output	TARGEXP0, HMASTER
TARGEXP0HWDATA[31:0]	output	TARGEXP0, HWDATA
TARGEXP0HMASTLOCK	output	TARGEXP0, HMASTLOCK
TARGEXP0HREADYMUX	output	TARGEXP0, HREADYOUT
TARGEXP0HAUSER	output	TARGEXP0, HAUSER
TARGEXP0HWUSER[3:0]	output	TARGEXP0, HWUSER
INITEXP0HRDATA[31:0]	output	INITEXP0, HRDATA
INITEXP0HREADY	output	INITEXP0, HREADY
INITEXP0HRESP	output	INITEXP0, HRESP
INITEXP0EXRESP	output	INITEXP0,EXRESP
INITEXP0HRUSER[2:0]	output	INITEXP0, HRUSER
APBTARGEXP2PSTRB[3:0]	output	APBTARGEXP2, PSTRB
APBTARGEXP2PPROT[2:0]	output	APBTARGEXP2, PPROT
APBTARGEXP2PSEL	output	APBTARGEXP2, PSELx
APBTARGEXP2PENABLE	output	APBTARGEXP2, PENABLE
APBTARGEXP2PADDR[11:0]	output	APBTARGEXP2, PADDR

Port Name	I/O	Description
APBTARGEXP2PWRITE	output	APBTARGEXP2, PWRITE
APBTARGEXP2PWDATA[31:0]	output	APBTARGEXP2, PWDATA
DAPSWDO	output	Serial Wire Data Out
DAPSWDOEN	output	Serial Wire Output Enable
DAPTDO	output	Debug TDO
DAPJTAGNSW	output	JTAG or Serial-Wire selection JTAG mode(1) or SW mode(0)
DAPNTDOEN	output	TDO output pad control signal
TPIUTRACEDATA[3:0]	output	Output data
TPIUTRACESWO	output	Serial Wire Viewer data
TPIUTRACECLK	output	Output clock, used by the TPA to sample the other pins

原语例化

Verilog 例化：

```
MCU u_sse050_top_syn (
  .FCLK(fclk),
  .PORESETN(poresetn),
  .SYSRESETN(sysresetn),
  .RTCSRCCLK(rtcsrcclk),
  .IOEXPINPUTI(ioexpinputi[15:0]),
  .IOEXPOUTPUTO(ioexpoutputo[15:0]),
  .IOEXPOUTPUTENO(ioexpoutputeno[15:0]),
  .UART0RXDI(uart0rxdi),
  .UART0TXDO(uart0txdo),
  .UART1RXDI(uart1rxdi),
  .UART1TXDO(uart1txdo),
  .SRAM0RDATA(sram0rdata[31:0]),
  .SRAM0ADDR(sram0addr[12:0]),
  .SRAM0WREN(sram0wren[3:0]),
  .SRAM0WDATA(sram0wdata[31:0]),
  .SRAM0CS(sram0cs),
  .MTXHRESETN(mtxhreset),
  .TARGFLASH0HSEL(targflash0hsel),
  .TARGFLASH0HADDR(targflash0haddr[28:0]),
  .TARGFLASH0HTRANS(targflash0htrans[1:0]),
  .TARGFLASH0HWRITE(targflash0hwrite),
  .TARGFLASH0HSIZE(targflash0hsize[2:0]),
  .TARGFLASH0HBURST(targflash0hburst[2:0]),
  .TARGFLASH0HPROT(targflash0hprot[3:0]),
  .TARGFLASH0MEMATTR(targflash0memattr[1:0]),
  .TARGFLASH0EXREQ(targflash0exreq),
  .TARGFLASH0HMASTER(targflash0hmaster[3:0]),
  .TARGFLASH0HWDATA(targflash0hwdata[31:0]),
  .TARGFLASH0HMASTLOCK(targflash0hmastlock),
  .TARGFLASH0HREADYMUX(targflash0hreadymux),
  .TARGFLASH0HAUSER(targflash0hauser),
```

```
.TARGFLASH0HWUSER(targflash0hwuser[3:0]),
.TARGFLASH0HRDATA(targflash0hrdata[31:0]),
.TARGFLASH0HRUSER(targflash0hruser[2:0]),
.TARGFLASH0HRESP(targflash0hresp),
.TARGFLASH0EXRESP(targflash0exresp),
.TARGFLASH0HREADYOUT(targflash0hreadyout),
.TARGEXP0HSEL(targexp0hsel),
.TARGEXP0HADDR(targexp0haddr[31:0]),
.TARGEXP0HTRANS(targexp0htrans[1:0]),
.TARGEXP0HWRITE(targexp0hwrite),
.TARGEXP0HSIZE(targexp0hsize[2:0]),
.TARGEXP0HBURST(targexp0hburst[2:0]),
.TARGEXP0HPROT(targexp0hprot[3:0]),
.TARGEXP0MEMATTR(targexp0memattr[1:0]),
.TARGEXP0EXREQ(targexp0exreq),
.TARGEXP0HMASTER(targexp0hmaster[3:0]),
.TARGEXP0HWDATA(targexp0hwdata[31:0]),
.TARGEXP0HMASTLOCK(targexp0hmastlock),
.TARGEXP0HREADYMUX(targexp0hreadymux),
.TARGEXP0HAUSER(targexp0hauser),
.TARGEXP0HWUSER(targexp0hwuser[3:0]),
.TARGEXP0HRDATA(targexp0hrdata[31:0]),
.TARGEXP0HREADYOUT(targexp0hreadyout),
.TARGEXP0HRESP(targexp0hresp),
.TARGEXP0EXRESP(targexp0exresp),
.TARGEXP0HRUSER(targexp0hruser[2:0]),
.INITEXP0HSEL(initexp0hsel),
.INITEXP0HADDR(initexp0haddr[31:0]),
.INITEXP0HTRANS(initexp0htrans[1:0]),
.INITEXP0HWRITE(initexp0hwrite),
.INITEXP0HSIZE(initexp0hsize[2:0]),
.INITEXP0HBURST(initexp0hburst[2:0]),
.INITEXP0HPROT(initexp0hprot[3:0]),
.INITEXP0MEMATTR(initexp0memattr[1:0]),
.INITEXP0EXREQ(initexp0exreq),
.INITEXP0HMASTER(initexp0hmaster[3:0]),
.INITEXP0HWDATA(initexp0hwdata[31:0]),
.INITEXP0HMASTLOCK(initexp0hmastlock),
.INITEXP0HAUSER(initexp0hauser),
.INITEXP0HWUSER(initexp0hwuser[3:0]),
.INITEXP0HRDATA(initexp0hrdata[31:0]),
.INITEXP0HREADY(initexp0hready),
.INITEXP0HRESP(initexp0hresp),
.INITEXP0EXRESP(initexp0exresp),
.INITEXP0HRUSER(initexp0hruser[2:0]),
.APBTARGEXP2PSEL(apbtargexp2psel),
.APBTARGEXP2PENABLE(apbtargexp2penable),
.APBTARGEXP2PADDR(apbtargexp2paddr[11:0]),
.APBTARGEXP2PWRITE(apbtargexp2pwrite),
.APBTARGEXP2PWDATA(apbtargexp2pwdata[31:0]),
```

```
  .APBTARGEXP2PRDATA(apbtargexp2prdata[31:0]),
  .APBTARGEXP2PREADY(apbtargexp2pready),
  .APBTARGEXP2PSLVERR(apbtargexp2pslverr),
  .APBTARGEXP2PSTRB(apbtargexp2pstrb[3:0]),
  .APBTARGEXP2PPROT(apbtargexp2pprot[2:0]),
  .MTXREMAP(mtxremap[3:0]),
  .DAPSWDITMS(dapswditms),
  .DAPSWDO(dapswdo),
  .DAPSWDOEN(dapswdoen),
  .DAPTDI(daptdi),
  .DAPTDO(daptdo),
  .DAPNTRST(dapntrst),
  .DAPSWCLKTCK(dapswclk_tck),
  .DAPNTDOEN(dapntdoen),
  .DAPJTAGNSW(dapjtagnsw),
  .TPIUTRACEDATA(tpiutracedata[3:0]),
  .TPIUTRACESWO(tpiutraceswo),
  .TPIUTRACECLK(tpiutraceclk),
  .FLASHERR(flasherr),
  .FLASHINT(flashint)
);
```

Vhdl 例化：

```
COMPONENT MCU
        PORT(
FCLK:IN std_logic;
PORESETN:IN std_logic;
SYSRESETN:IN std_logic;
RTCSRCCLK:IN std_logic;
UART0RXDI:IN std_logic;
UART1RXDI:IN std_logic;
CLK:IN std_logic;
RESET:IN std_logic;
IOEXPINPUTI:IN std_logic_vector(15 downto 0);
SRAM0RDATA:IN std_logic_vector(31 downto 0);
TARGFLASH0HRDATA:IN std_logic_vector(31 downto 0);
TARGFLASH0HRUSER:IN std_logic_vector(2 downto 0);
TARGFLASH0HRESP:IN std_logic;
TARGFLASH0EXRESP:IN std_logic;
TARGFLASH0HREADYOUT:IN std_logic;
TARGEXP0HRDATA: IN std_logic_vector(31 downto 0);
TARGEXP0HREADYOUT:IN std_logic;
TARGEXP0HRESP:IN std_logic;
TARGEXP0EXRESP:IN std_logic;
TARGEXP0HRUSER: IN std_logic_vector(2 downto 0);
INITEXP0HSEL:IN std_logic;
INITEXP0HADDR: IN std_logic_vector(31 downto 0);
INITEXP0HTRANS: IN std_logic_vector(1 downto 0);
INITEXP0HWRITE: IN std_logic;
INITEXP0HSIZE: IN std_logic_vector(2 downto 0);
INITEXP0HBURST: IN std_logic_vector(2 downto 0);
```

```
INITEXP0HPROT: IN std_logic_vector(3 downto 0);
INITEXP0MEMATTR: IN std_logic_vector(1 downto 0);
INITEXP0EXREQ: IN std_logic;
INITEXP0HMASTER: IN std_logic_vector(3 downto 0);
INITEXP0HWDATA: IN std_logic_vector(31 downto 0);
INITEXP0HMASTLOCK: IN std_logic;
INITEXP0HAUSER: IN std_logic;
INITEXP0HWUSER: IN std_logic_vector(3 downto 0);
APBTARGEXP2PRDATA: IN std_logic_vector(3 downto 0);
APBTARGEXP2PREADY: IN std_logic;
APBTARGEXP2PSLVERR: IN std_logic;
MTXREMAP: IN std_logic_vector(3 downto 0);
DAPSWDITMS: IN std_logic;
DAPTDI: IN std_logic;
DAPNTRST: IN std_logic;
DAPSWCLKTCK: IN std_logic;
FLASHERR: IN std_logic;
FLASHINT: IN std_logic;
IOEXPOUTPUTO:OUT std_logic_vector(15 downto 0);
IOEXPOUTPUTENO:OUT std_logic_vector(15 downto 0);
IOEXPINPUTI:OUT std_logic_vector(15 downto 0);
UART0TXDO: OUT std_logic;
UART1TXDO: OUT std_logic;
UART0BAUDTICK: OUT std_logic;
UART1BAUDTICK: OUT std_logic;
INTMONITOR: OUT std_logic;
MTXHRESETN: OUT std_logic;
SRAM0ADDR:OUT std_logic_vector(12 downto 0);
SRAM0WREN:OUT std_logic_vector(3 downto 0);
SRAM0WDATA:OUT std_logic_vector(31 downto 0);
SRAM0CS: OUT std_logic;
TARGFLASH0HSEL: OUT std_logic;
TARGFLASH0HWRITE: OUT std_logic;
TARGFLASH0EXREQ: OUT std_logic;
TARGFLASH0HMASTLOCK: OUT std_logic;
TARGFLASH0HREADYMUX: OUT std_logic;
TARGFLASH0HAUSER: OUT std_logic;
SRAM0RDATA:OUT std_logic_vector(31 downto 0);
TARGFLASH0HADDR:OUT std_logic_vector(28 downto 0);
TARGFLASH0HTRANS:OUT std_logic_vector(1 downto 0);
TARGFLASH0HSIZE:OUT std_logic_vector(2 downto 0);
TARGFLASH0HBURST:OUT std_logic_vector(2 downto 0);
TARGFLASH0HPROT:OUT std_logic_vector(3 downto 0);
TARGFLASH0MEMATTR:OUT std_logic_vector(1 downto 0);
TARGFLASH0HMASTER:OUT std_logic_vector(3 downto 0);
TARGFLASH0HWDATA:OUT std_logic_vector(31 downto 0);
TARGFLASH0HWUSER:OUT std_logic_vector(3 downto 0);
TARGFLASH0HRDATA:OUT std_logic_vector(31 downto 0);
TARGEXP0HADDR:OUT std_logic_vector(31 downto 0);
TARGEXP0HSEL: OUT std_logic;
```

```
        TARGEXP0HWRITE: OUT std_logic;
        TARGEXP0EXREQ: OUT std_logic;
        TARGEXP0HMASTLOCK: OUT std_logic;
        TARGEXP0HREADYMUX: OUT std_logic;
        TARGEXP0HAUSER: OUT std_logic;
        INITEXP0HREADY: OUT std_logic;
        INITEXP0HRESP: OUT std_logic;
        INITEXP0EXRESP: OUT std_logic;
        TARGEXP0HTRANS:OUT std_logic_vector(1 downto 0);
        TARGEXP0HSIZE:OUT std_logic_vector(2 downto 0);
        TARGEXP0HBURST:OUT std_logic_vector(2 downto 0);
        TARGEXP0HPROT:OUT std_logic_vector(3 downto 0);
        TARGEXP0MEMATTR:OUT std_logic_vector(1 downto 0);
        TARGEXP0HMASTER:OUT std_logic_vector(3 downto 0);
        TARGEXP0HWDATA:OUT std_logic_vector(31 downto 0);
        TARGEXP0HWUSER:OUT std_logic_vector(3 downto 0);
        INITEXP0HRDATA:OUT std_logic_vector(31 downto 0);
        INITEXP0HRUSER:OUT std_logic_vector(2 downto 0);
        APBTARGEXP2PSTRB:OUT std_logic_vector(3 downto 0);
        APBTARGEXP2PPROT:OUT std_logic_vector(2 downto 0);
        APBTARGEXP2PADDR:OUT std_logic_vector(11 downto 0);
        APBTARGEXP2PWDATA:OUT std_logic_vector(31 downto 0);
        TPIUTRACEDATA:OUT std_logic_vector(3 downto 0);
        APBTARGEXP2PSEL: OUT std_logic;
        APBTARGEXP2PENABLE: OUT std_logic;
        APBTARGEXP2PWRITE: OUT std_logic;
        DAPSWDO: OUT std_logic;
        DAPSWDOEN: OUT std_logic;
        DAPTDO: OUT std_logic;
        DAPJTAGNSW: OUT std_logic;
        DAPNTDOEN: OUT std_logic;
        TPIUTRACESWO: OUT std_logic;
        TPIUTRACECLK: OUT std_logic;
    );
        END COMPONENT;

        uut: MCU
            PORT MAP (
        FCLK=> fclk;
        PORESETN=> poresetn;
        SYSRESETN=> sysresetn;
        RTCSRCCLK=> rtcsrcclk;
        UART0RXDI=> uart0rxdi;
        UART1RXDI=> uart1rxdi;
        CLK=>clk,
        RESET=>reset,
        IOEXPINPUTI=>ioexpinputi,
        SRAM0RDATA=>sram0rdata,
        TARGFLASH0HRDATA=>targflash0hrdata,
        TARGFLASH0HRUSER=>targflash0hruser,
```

```
TARGFLASH0HRESP=>targflash0hresp,
TARGFLASH0EXRESP=>targflash0exresp,
TARGFLASH0HREADYOUT=>targflash0hreadyout,
TARGEXP0HRDATA=>targexp0hrdata,
TARGEXP0HREADYOUT=>targexp0hreadyout,
TARGEXP0HRESP=>targexp0hresp,
TARGEXP0EXRESP=>targexp0exresp,
TARGEXP0HRUSER=>targexp0hruser,
INITEXP0HSEL=>initexp0hsel,
INITEXP0HADDR=>initexp0haddr,
INITEXP0HTRANS=>initexp0htrans,
INITEXP0HWRITE=>initexp0hwrite,
INITEXP0HSIZE=>initexp0hsize,
INITEXP0HBURST=>initexp0hburst,
INITEXP0HPROT=>initexp0hprot,
INITEXP0MEMATTR=>initexp0memattr,
INITEXP0EXREQ=>initexp0exreq,
INITEXP0HMASTER=>initexp0hmaster,
INITEXP0HWDATA=>initexp0hwdata,
INITEXP0HMASTLOCK=>initexp0hmastlock,
INITEXP0HAUSER=>initexp0hauser,
INITEXP0HWUSER=>initexp0hwuser,
APBTARGEXP2PRDATA=>apbtargexp2prdata,
APBTARGEXP2PREADY=>apbtargexp2pready,
APBTARGEXP2PSLVERR=>apbtargexp2pslverr,
MTXREMAP=>mtxremap,
DAPSWDITMS=>dapswditms,
DAPTDI=>daptdi,
DAPNTRST=>dapntrst,
DAPSWCLKTCK=>dapswclktck,
FLASHERR=>flasherr,
FLASHINT=>flashint,
IOEXPOUTPUTO=>ioexpoutputo,
IOEXPOUTPUTENO=>ioexpoutputeno,
IOEXPINPUTI=>ioexpinputi,
UART0TXDO=>uart0txdo,
UART1TXDO=>uart1txdo,
UART0BAUDTICK=>uart0baudtick,
UART1BAUDTICK=>uart1baudtick,
INTMONITOR=>intmonitor,
MTXHRESETN=>mtxhresetn,
SRAM0ADDR=>sram0addr,
SRAM0WREN=>sram0wren,
SRAM0WDATA=>sram0wdata,
SRAM0CS=>sram0cs,
TARGFLASH0HSEL=>targflash0hsel,
TARGFLASH0HWRITE=>targflash0hwrite,
TARGFLASH0EXREQ=>targflash0exreq,
TARGFLASH0HMASTLOCK=>targflash0hmastlock,
TARGFLASH0HREADYMUX=>targflash0hreadymux,
```

```
TARGFLASH0HAUSER=>targflash0hauser,
SRAM0RDATA=>sram0rdata,
TARGFLASH0HADDR=>targflash0haddr,
TARGFLASH0HTRANS=>targflash0htrans,
TARGFLASH0HSIZE=>targflash0hsize,
TARGFLASH0HBURST=>targflash0hburst,
TARGFLASH0HPROT=>targflash0hprot,
TARGFLASH0MEMATTR=>targflash0memattr,
TARGFLASH0HMASTER=>targflash0hmaster,
TARGFLASH0HWDATA=>targflash0hwdata,
TARGFLASH0HWUSER=>targflash0hwuser,
TARGFLASH0HRDATA=>targflash0hrdata,
TARGEXP0HADDR=>targexp0haddr,
TARGEXP0HSEL=>targexp0hsel,
TARGEXP0HWRITE=>targexp0hwrite,
TARGEXP0EXREQ=>targexp0exreq,
TARGEXP0HMASTLOCK=>targexp0hmastlock,
TARGEXP0HREADYMUX=>targexp0hreadymux,
TARGEXP0HAUSER=>targexp0hauser,
INITEXP0HREADY=>initexp0hready,
INITEXP0HRESP=>initexp0hresp,
INITEXP0EXRESP=>initexp0exresp,
TARGEXP0HTRANS=>targexp0htrans,
TARGEXP0HSIZE=>targexp0hsize,
TARGEXP0HBURST=>targexp0hburst,
TARGEXP0HPROT=>targexp0hprot,
TARGEXP0MEMATTR=>targexp0memattr,
TARGEXP0HMASTER=>targexp0hmaster,
TARGEXP0HWDATA=>targexp0hwdata,
TARGEXP0HWUSER=>targexp0hwuser,
INITEXP0HRDATA=>initexp0hrdata,
INITEXP0HRUSER=>initexp0hruser,
APBTARGEXP2PSTRB=>apbtargexp2pstrb,
APBTARGEXP2PPROT=>apbtargexp2pprot,
APBTARGEXP2PADDR=>apbtargexp2paddr,
APBTARGEXP2PWDATA=>apbtargexp2pwdata,
TPIUTRACEDATA=>tpiutracedata,
APBTARGEXP2PSEL=>apbtargexp2psel,
APBTARGEXP2PENABLE=>apbtargexp2penable,
APBTARGEXP2PWRITE=>apbtargexp2pwrite,
DAPSWDO=>dapswdo,
DAPSWDOEN=>dapswdoen,
DAPTDO=>daptdo,
DAPJTAGNSW=>dapjtagnsw,
DAPNTDOEN=>dapntdoen,
TPIUTRACESWO=>tpiutraceswo,
TPIUTRACECLK=>tpiutraceclk );
```

8.2 USB20_PHY

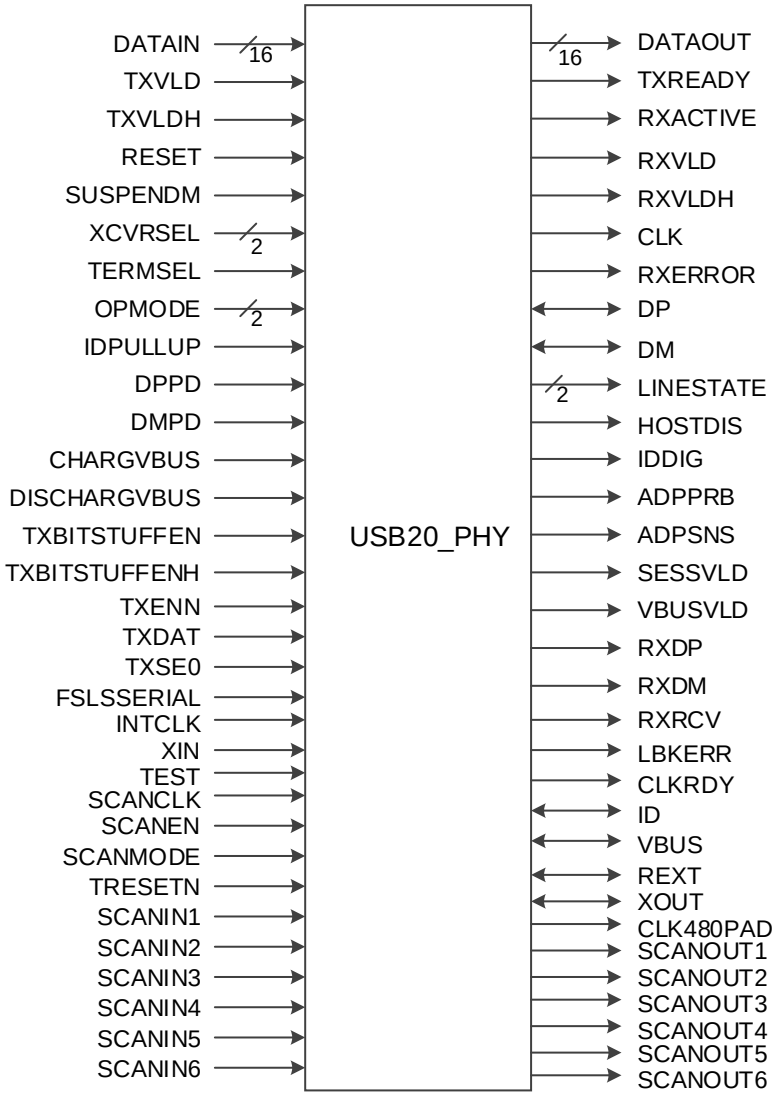
原语介绍

USB20_PHY 是完整的混合信号 IP 解决方案，实现从 Soc（System-on-Chip）到其他特殊制造工艺的 OTG 连接。USB20_PHY 支持 USB 2.0 480-Mbps 的协议和数据速率，并且后向兼容 USB 1.1 1.5-Mbps 和 12-Mbps 的协议和数据速率。

支持器件：GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C。

结构框图

图 8-2 USB20_PHY 结构框图



Port 介绍

表 8-2 Port 介绍

Port Name	I/O	Description
DATAIN[15:0]	input	16-bit parallel USB data input bus

Port Name	I/O	Description
TXVLD	input	Transmit Valid. Indicates that the DataIn bus is valid.
TXVLDH	input	Transmit Valid High.When DataBus16_8 = 1, this signal indicates that the DataIn[15:8] bus contains valid transmit data.
RESET	input	Reset. Reset all state machines in the UTM.
SUSPENDM	input	Suspend. 0:suspend, 1: normal
XCVRSEL[1:0]	input	Transceiver Select. This signal selects between the LS, FS and HS transceivers
TERMSEL	input	Termination Select. This signal selects between the FS and HS terminations
OPMODE[1:0]	input	Operational Mode. These signals select between various operational modes
IDPULLUP	input	Signal that enables the sampling of the analog Id line.
DPPD	input	This signal enables the 15k Ohm pull-down resistor on the DP line.
DMPD	input	0b : Pull-down resistor not connected to DM; 1b : Pull-down resistor connected to DM
CHARGVBUS	input	This signal enables charging Vbus
DISCHARGVBUS	input	The signal enables discharging Vbus.
TXBITSTUFFEN	input	Indicates if the data on the DataOut[7:0] lines needs to be bitstuffed or not.
TXBITSTUFFENH	input	Indicates if the data on the DataOut[15:8] lines needs to be bitstuffed or not.
TXENN	input	Active low enable signal. Only used when FsLsSerialMode is set to 1b
TXDAT	input	Differential data at D+/D- output. Only used when FsLsSerialMode is set to 1b
TXSE0	input	Force Single-Ended Zero. Only used when FsLsSerialMode is set to 1b
FSLSSERIAL	input	0b : FS and LS packets are sent using the parallel interface. 1b : FS and LS packets are sent using the serial interface.
INTCLK	input	Clock signals provided internally of the SoC
TEST	input	For IP TESTing purpose.Please leave it unconnected since there are already soft pull-down in the IP
SCANCLK	input	Clock signals for scan mode
SCANEN	input	Select to shift mode
SCANMODE	input	High effective signal to enter scan mode
TRESETN	input	Low effective RESET signal for scan mode
SCANIN1	input	Scan chain input
SCANIN2	input	Scan chain input
SCANIN3	input	Scan chain input
SCANIN4	input	Scan chain input
SCANIN5	input	Scan chain input
SCANIN6	input	Scan chain input
DP	inout	USB data pin Data+
DM	inout	USB data pin Data–
ID	inout	ID signal from the cable
VBUS	inout	Vbus signals connected with the cable
REXT	inout	12.7K High precision resistor
XIN	inout	Crystal in signals, supported range is 12MHZ~24MHZ
XOUT	inout	Crystal out signals
DATAOUT[15:0]	output	DataOut. 16-bit parallel USB data output bus.
TXREADY	output	Transmit Data Ready.

Port Name	I/O	Description
RXACTIVE	output	Receive Active. Indicates that the receive state machine has detected SYNC and is active.
RXVLD	output	Receive Data Valid. Indicates that the DataOut bus has valid data.
RXVLDH	output	Receive Data Valid High.
CLK	output	Clock. This output is used for clocking receive and transmit parallel data.
RXERROR	output	Receive Error.
LINESTATE[1:0]	output	Line State. These signals reflect the current state of the single ended receivers.
HOSTDIS	output	This signal is used for all types of peripherals connected to it.
IDDIG	output	Indicates whether the connected plug is a mini-A or mini-B.
ADPPRB	output	Indicates if the voltage on Vbus (0.6V < Vth < 0.75V).
ADPSNS	output	Indicates if the voltage on Vbus (0.2V < Vth < 0.55V).
SESSVLD	output	Indicates if the session for an A/B-peripheral is valid (0.8V < Vth < 2V).
VBUSVLD	output	Indicates if the voltage on Vbus is at a valid level for operation (4.4V < Vth < 4.75V)
RXDP	output	Single-ended receive data, positive terminal.This signal is only valid if FsLsSerialMode is set to 1b
RXDM	output	Single-ended receive data, negative terminal.This signal is only valid if FsLsSerialMode is set to 1b
RXRCV	output	Receive data.This signal is only valid if FsLsSerialMode is set to 1b
LBKERR	output	used for observation
CLKRDY	output	Observation/debug signal to show that the internal PLL has locked and is ready.
CLK480PAD	output	480MHZ clock output for observation
SCANOUT1	output	Scan chain output
SCANOUT2	output	Scan chain output
SCANOUT3	output	Scan chain output
SCANOUT4	output	Scan chain output
SCANOUT5	output	Scan chain output
SCANOUT6	output	Scan chain output

Attribute 介绍

表 8-3 Attribute 介绍

Attribute Name	Default	Description
DATABUS16_8	1’b0	Selects between 8 and 16 bit data transfers.
ADP_PRBEN	1’b0	Enables/disables the ADP Probe comparator
TEST_MODE	5’b0	used for testing and debugging purpose
HSDRV1	1’b0	High speed drive adjustment. Please connect to 0 for normal operation.
HSDRV0	1’b0	High speed drive adjustment. Please connect to 0 for normal operation.
CLK_SEL	1’b0	Clock source selection signal. 0 to select external clock provided by the crystal connected on XIN, XOUT. 1 to select internal clock provided on INTCLK port
M	4’b0	M divider input data bit
N	6'b101000	N divider input data bit

Attribute Name	Default	Description
C	2'b01	Control charge pump current input data bit, it supports from 30uA (00) to 60uA (11).
FOC_LOCK	1'b0	0: LOCK is generated by PLL lock detector. 1: LOCK is always high(always lock)

原语例化

Verilog 例化：

```
USB20_PHY usb20_phy_inst (
  .DATAOUT(dataout[15:0]),
  .TXREADY(txready),
  .RXACTIVE(rxactive),
  .RXVLD(rxvld),
  .RXVLDH(rxvldh),
  .CLK(clk),
  .RXERROR(rxerror),
  .DP(dp),
  .DM(dm),
  .LINESTATE(linestate[1:0]),
  .DATAIN(datain[15:0]),
  .TXVLD(txvld),
  .TXVLDH(txvldh),
  .RESET(reset),
  .SUSPENDM(suspendm),
  .XCVRSEL(xcvrsel[1:0]),
  .TERMSEL(termsel),
  .OPMODE(opmode[1:0]),
  .HOSTDIS(hostdis),
  .IDDIG(iddig),
  .ADPPRB(adpprb),
  .ADPSNS(adpsns),
  .SESSVLD(sessvld),
  .VBUSVLD(vbusvld),
  .RXDP(rxdp),
  .RXDM(rxdm),
  .RXRCV(rxrcv),
  .IDPULLUP(idpullup),
  .DPPD(dppd),
  .DMPD(dmpd),
  .CHARGVBUS(chargvbus),
  .DISCHARGVBUS(dischargvbus),
  .TXBITSTUFFEN(txbitstuffen),
  .TXBITSTUFFENH(txbitstuffenh),
  .TXENN(txenn),
  .TXDAT(txdat),
  .TXSE0(txse0),
  .FSLSSERIAL(fslsserial),
  .LBKERR(lbkerr),
  .CLKRDY(clkrdy),
```

```
    .INTCLK(intclk),
    .ID(id),
    .VBUS(vbus),
    .REXT(rext),
    .XIN(xin),
    .XOUT(xout),
    .CLK480PAD(clk480pad),
    .TEST(test),
    .SCANOUT1(scanout1),
    .SCANOUT2(scanout2),
    .SCANOUT3(scanout3),
    .SCANOUT4(scanout4),
    .SCANOUT5(scanout5),
    .SCANOUT6(scanout6),
    .SCANCLK(scanclk),
    .SCANEN(scanen),
    .SCANMODE(scanmode),
    .TRESETN(tresetn),
    .SCANIN1(scanin1),
    .SCANIN2(scanin2),
    .SCANIN3(scanin3),
    .SCANIN4(scanin4),
    .SCANIN5(scanin5),
    .SCANIN6(scanin6)
);
    defparam usb20_phy_inst.DATABUS16_8 = 1'b0;
    defparam usb20_phy_inst.ADP_PRBEN = 1'b0;
    defparam usb20_phy_inst.TEST_MODE = 5'b0;;
    defparam usb20_phy_inst.HSDRV1 = 1'b0;
    defparam usb20_phy_inst.HSDRV0 = 1'b0;
    defparam usb20_phy_inst.CLK_SEL = 1'b0;
    defparam usb20_phy_inst.M = 4'b0;
    defparam usb20_phy_inst.N = 6'b101000;
    defparam usb20_phy_inst.C = 2'b01;
    defparam usb20_phy_inst.FOC_LOCK = 1'b0;
```

Vhdl 例化：

```
    COMPONENT USB20_PHY
            GENERIC (
                        TEST_MODE:bit_vector:="00000";
                        DATABUS16_8:bit:='0';
                        ADP_PRBEN:bit:='0';
                        HSDRV1:bit:='0';
                        HSDRV0:bit:='0';
                        CLK_SEL:bit:='0';
                        M:bit_vector:="0000";
                        N:bit_vector:=" 101000";
                        C:bit_vector:="01";
                        FOC_LOCK:bit:='0';
                );
```

```
PORT(
        DATAIN:IN std_logic_vector(15 downto 0);
        TXVLD:IN std_logic;
        TXVLDH:IN std_logic;
        RESET:IN std_logic;
        SUSPENDM:IN std_logic;
        XCVRSEL:IN std_logic_vector(1 downto 0);
        TERMSEL:IN std_logic;
        OPMODE:IN std_logic_vector(1 downto 0);
        DATAOUT:OUT std_logic_vector(15 downto 0);
        TXREADY:OUT std_logic;
        RXACTIVE:OUT std_logic;
        RXVLD:OUT std_logic;
        RXVLDH:OUT std_logic;
        CLK:OUT std_logic;
        RXERROR:OUT std_logic;
        DP:INOUT std_logic;
        DM:INOUT std_logic;
        LINESTATE:OUT std_logic_vector(1 downto 0);
        IDPULLUP:IN std_logic;
        DPPD:IN std_logic;
        DMPD:IN std_logic;
        CHARGVBUS:IN std_logic;
        DISCHARGVBUS:IN std_logic;
        TXBITSTUFFEN:IN std_logic;
        TXBITSTUFFENH:IN std_logic;
        TXENN:IN std_logic;
        TXDAT:IN std_logic;
        TXSE0:IN std_logic;
        FSLSSERIAL:IN std_logic;
        HOSTDIS:OUT std_logic;
        IDDIG:OUT std_logic;
        ADPPRB:OUT std_logic;
        ADPSNS:OUT std_logic;
        SESSVLD:OUT std_logic;
        VBUSVLD:OUT std_logic;
        RXDP:OUT std_logic;
        RXDM:OUT std_logic;
        RXRCV:OUT std_logic;
        LBKERR:OUT std_logic;
        CLKRDY:OUT std_logic;
        INTCLK:IN std_logic;
        ID:INOUT std_logic;
        VBUS:INOUT std_logic;
        REXT:INOUT std_logic;
        XIN:IN std_logic;
        XOUT:INOUT std_logic;
        TEST:IN std_logic;
        CLK480PAD:OUT std_logic;
        SCANCLK:IN std_logic;
```

```
                SCANEN:IN std_logic;
                SCANMODE:IN std_logic;
                TRESETN:IN std_logic;
                SCANIN1:IN std_logic;
                SCANOUT1:OUT std_logic;
                SCANIN2:IN std_logic;
                SCANOUT2:OUT std_logic;
                SCANIN3:IN std_logic;
                SCANOUT3:OUT std_logic;
                SCANIN4:IN std_logic;
                SCANOUT4:OUT std_logic;
                SCANIN5:IN std_logic;
                SCANOUT5:OUT std_logic;
                SCANIN6:IN std_logic;
                SCANOUT6:OUT std_logic;
        );
END COMPONENT;
uut: USB20_PHY
        PORT MAP (
                DATAIN=>datain,
                TXVLD=>txvld,
                TXVLDH=>txvldh,
                RESET=>reset,
                SUSPENDM=>suspendm,
                XCVRSEL=>xcvrsel,
                TERMSEL=>termsel,
                OPMODE=>opmode,
                DATAOUT=>dataout,
                TXREADY=>txready,
                RXACTIVE=>rxactive,
                RXVLD=>rxvld,
                RXVLDH=>rxvldh,
                CLK=>clk,
                RXERROR=>rxerror,
                DP=>dp,
                DM=>dm,
                LINESTATE=>linestate,
                IDPULLUP=>idpullup,
                DPPD=>dppd,
                DMPD=>dmpd,
                CHARGVBUS=>chargvbus,
                DISCHARGVBUS=>dischargvbus,
                TXBITSTUFFEN=>txbitstuffen,
                TXBITSTUFFENH=>txbitstuffenh,
                TXENN=>txenn,
                TXDAT=>txdat,
                TXSE0=>txse0,
                FSLSSERIAL=>fslsserial,
                HOSTDIS=>hostdis,
                IDDIG=>iddig,
```

```
        ADPPRB=>adpprb,
        ADPSNS=>adpsns,
        SESSVLD=>sessvld,
        VBUSVLD=>vbusvld,
        RXDP=>rxdp,
        RXDM=>rxdm,
        RXRCV=>rxrcv,
        LBKERR=>lbkerr,
        CLKRDY=>clkrdy,
        INTCLK=>intclk,
        ID=>id,
        VBUS=>vbus,
        REXT=>rext,
        XIN=>xin,
        XOUT=>xout,
        TEST=>test,
        CLK480PAD=>clk480pad,
        SCANCLK=>scanclk,
        SCANEN=>scanen,
        SCANMODE=>scanmode,
        TRESETN=>tresetn,
        SCANIN1=>scanin1,
        SCANOUT1=>scanout1,
        SCANIN2=>scanin2,
        SCANOUT2=>scanout2,
        SCANIN3=>scanin3,
        SCANOUT3=>scanout3,
        SCANIN4=>scanin4,
        SCANOUT4=>scanout4,
        SCANIN5=>scanin5,
        SCANOUT5=>scanout5,
        SCANIN6=>scanin6,
        SCANOUT6=>scanout6
    );
```

8.3 ADC

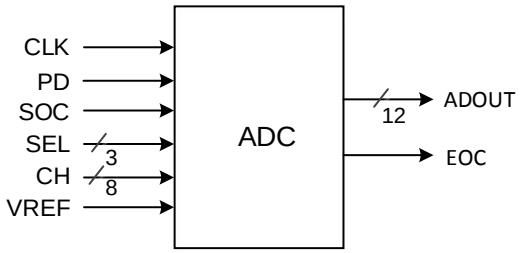
原语介绍

ADC（Analog-to-digital Converter）是一个 8 通道单端 12 位的模数转换器，具有低功耗、低漏电和高动态特性。

支持器件：GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C。

结构框图

图 8-3 ADC 结构框图



Port 介绍

表 8-4 Port 介绍

Port Name	I/O	Description
ADOUT[11:0]	Output	ad conversion results.
EOC	Output	end of conversion.
CLK	Input	main clock.
PD	Input	power down signal.
SOC	Input	start of conversion.
SEL[2:0]	Input	channel select signal.
CH[7:0]	Input	channel signal-ended analog voltage input.
VREF	Input	voltage reference

原语例化

Verilog 例化：

```
ADC adc_inst(
      .CLK(clk),
      .PD(pd),
      .SOC(soc),
      .SEL(sel[2:0]),
      .CH(ch[7:0]),
      .VREF(vref),
      .EOC(eoc),
      .ADOUT(adout[11:0])
);
```

Vhdl 例化：

```
COMPONENT ADC
        PORT(
            CLK=>IN std_logic;
            PD=>IN std_logic;
            SOC=>IN std_logic;
            SEL=>IN std_logic_vector(2 downto 0);
            CH=>IN std_logic_vector(7 downto 0);
            VREF=>IN std_logic;
            EOC=>OUT std_logic;
```

```
            ADOUT=>OUT std_logic_vector(11 downto 0)
        );
END COMPONENT;
uut=> ADC
        PORT MAP (
            CLK=>clk,
            PD=>pd,
            SOC=>soc,
            SEL=>sel,
            CH=>ch,
            VREF=>vref,
            EOC=>eoc,
            ADOUT=>adout
        );
```

8.4 EMCU

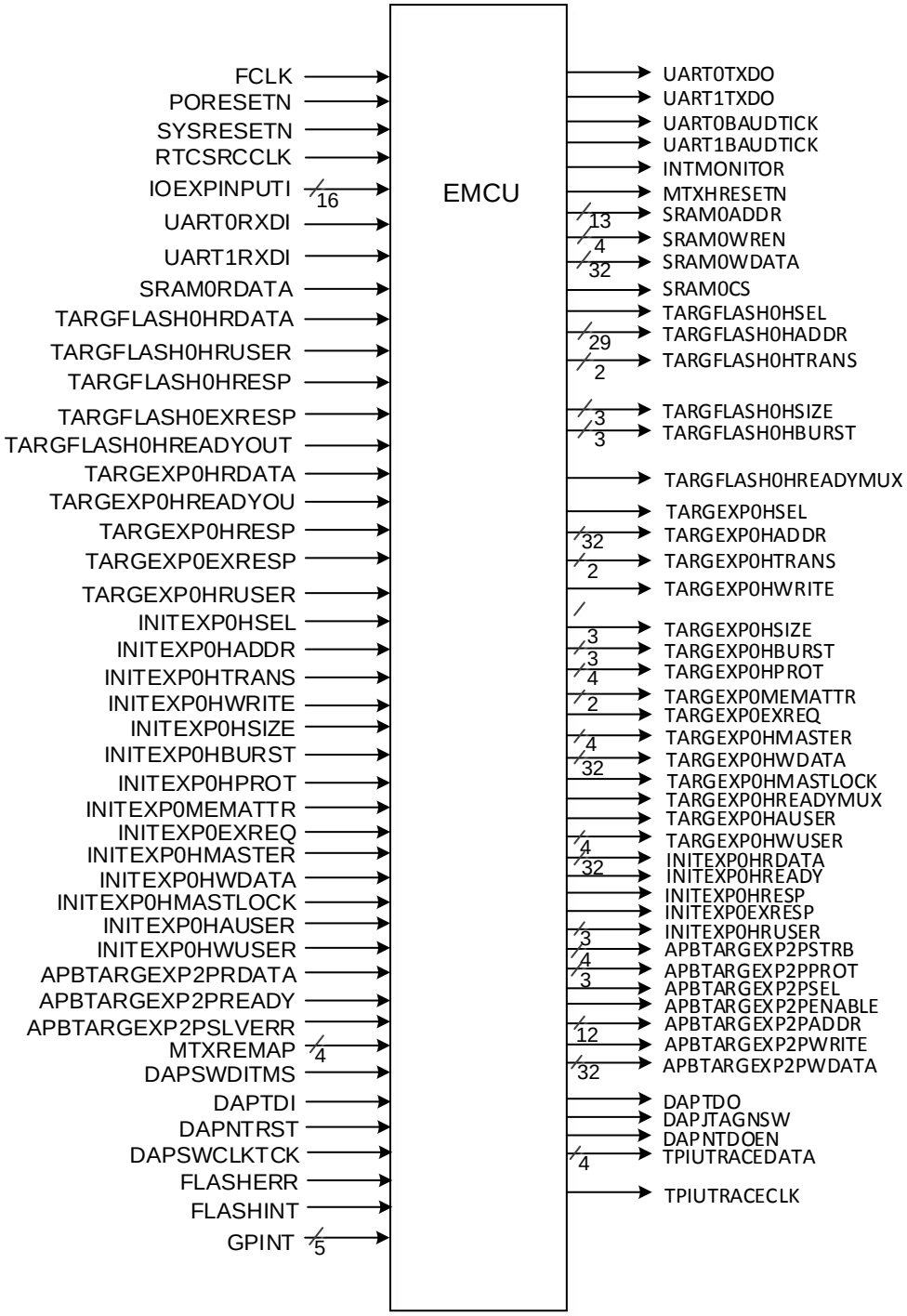
原语介绍

EMCU(ARM Cortex-M3 Microcontroller Unit)是一款基于 ARM Cortex-M3 的微处理器。采用了 32 位 AHB/APB 的总线模式。其内部实现了 2 个 UART、2 个 Timer 和 Watchdog 的功能。并且对外提供 16 位 GPIO、2 个 UART、JTAG、2 个 User Interrupt 接口。以及 AHB Flash 读取接口、AHB Sram 读写接口。同时对外还提供了 2 个 AHB 总线扩展接口和 1 个 APB 总线扩展接口。

支持器件：GW1NSR-4C、GW1NSER-4C。

结构框图

图 8-4 EMCU 结构框图



Port 介绍

表 8-5 Port 介绍

Port Name	I/O	Description
FCLK	input	Free running clock
PORESETN	input	Power on reset
SYSRESETN	input	System reset
RTCSRCCLK	input	Used to generate RTC clock

Port Name	I/O	Description
IOEXPINPUTI[15:0]	input	IOEXPINPUTI
UART0RXDI	input	UART0RXDI
UART1RXDI	input	UART1RXDI
SRAM0RDATA[31:0]	input	SRAM Read data bus
TARGFLASH0HRDATA[31:0]	input	TARGFLASH0, HRDATA
TARGFLASH0HRUSER[2:0]	input	TARGFLASH0, HRUSER
TARGFLASH0HRESP	input	TARGFLASH0, HRESP
TARGFLASH0EXRESP	input	TARGFLASH0, EXRESP
TARGFLASH0HREADYOUT	input	TARGFLASH0, EXRESP
TARGEXP0HRDATA[31:0]	input	TARGEXP0, HRDATA
TARGEXP0HREADYOUT	input	TARGEXP0, HREADY
TARGEXP0HRESP	input	TARGEXP0, HRESP
TARGEXP0EXRESP	input	TARGEXP0, EXRESP
TARGEXP0HRUSER[2:0]	input	TARGEXP0, HRUSER
INITEXP0HSEL	input	INITEXP0, HSELx
INITEXP0HADDR[31:0]	input	INITEXP0, HADDR
INITEXP0HTRANS[1:0]	input	INITEXP0, HTRANS
INITEXP0HWRITE	input	INITEXP0, HWRITE
INITEXP0HSIZE[2:0]	input	INITEXP0, HSIZE
INITEXP0HBURST[2:0]	input	INITEXP0, HBURST
INITEXP0HPROT[3:0]	input	INITEXP0, HPROT
INITEXP0MEMATTR[1:0]	input	INITEXP0, MEMATTR
INITEXP0EXREQ	input	INITEXP0, EXREQ
INITEXP0HMASTER[3:0]	input	INITEXP0, HMASTER
INITEXP0HWDATA[31:0]	input	INITEXP0, HWDATA
INITEXP0HMASTLOCK	input	INITEXP0, HMASTLOCK
INITEXP0HAUSER	input	INITEXP0, HAUSER
INITEXP0HWUSER[3:0]	input	INITEXP0, HWUSER
APBTARGEXP2PRDATA[31:0]	input	APBTARGEXP2, PRDATA
APBTARGEXP2PREADY	input	APBTARGEXP2, PREADY
APBTARGEXP2PSLVERR	input	APBTARGEXP2, PSLVERR
MTXREMAP[3:0]	input	The MTXREMAP signals control the remapping of the boot memory range.
DAPSWDITMS	input	Debug TMS
DAPTDI	input	Debug TDI
DAPNTRST	input	Test reset
DAPSWCLKTCK	input	Test clock / SWCLK
FLASHERR	input	Output clock, used by the TPA to sample the other pins
FLASHINT	input	Output clock, used by the TPA to sample the other pins
GPINT	input	GPINT
IOEXPOUTPUTO[15:0]	output	IOEXPOUTPUTO
IOEXPOUTPUTENO[15:0]	output	IOEXPOUTPUTENO
UART0TXDO	output	UART0TXDO
UART1TXDO	output	UART1TXDO
UART0BAUDTICK	output	UART0BAUDTICK
UART1BAUDTICK	output	UART1BAUDTICK

Port Name	I/O	Description
INTMONITOR	output	INTMONITOR
MTXHRESETN	output	SRAM/Flash Chip reset
SRAM0ADDR[12:0]	output	SRAM address
SRAM0WREN[3:0]	output	SRAM Byte write enable
SRAM0WDATA[31:0]	output	SRAM Write data
SRAM0CS	output	SRAM Chip select
TARGFLASH0HSEL	output	TARGFLASH0, HSELx
TARGFLASH0HADDR[28:0]	output	TARGFLASH0, HADDR
TARGFLASH0HTRANS[1:0]	output	TARGFLASH0, HTRANS
TARGFLASH0HSIZE[2:0]	output	TARGFLASH0, HSIZE
TARGFLASH0HBURST[2:0]	output	TARGFLASH0, HBURST
TARGFLASH0HREADYMUX	output	TARGFLASH0, HREADYOUT
TARGEXP0HSEL	output	TARGEXP0, HSELx
TARGEXP0HADDR[31:0]	output	TARGEXP0, HADDR
TARGEXP0HTRANS[1:0]	output	TARGEXP0, HTRANS
TARGEXP0HWRITE	output	TARGEXP0, HWRITE
TARGEXP0HSIZE[2:0]	output	TARGEXP0, HSIZE
TARGEXP0HBURST[2:0]	output	TARGEXP0, HBURST
TARGEXP0HPROT[3:0]	output	TARGEXP0, HPROT
TARGEXP0MEMATTR[1:0]	output	TARGEXP0, MEMATTR
TARGEXP0EXREQ	output	TARGEXP0, EXREQ
TARGEXP0HMASTER[3:0]	output	TARGEXP0, HMASTER
TARGEXP0HWDATA[31:0]	output	TARGEXP0, HWDATA
TARGEXP0HMASTLOCK	output	TARGEXP0, HMASTLOCK
TARGEXP0HREADYMUX	output	TARGEXP0, HREADYOUT
TARGEXP0HAUSER	output	TARGEXP0, HAUSER
TARGEXP0HWUSER[3:0]	output	TARGEXP0, HWUSER
INITEXP0HRDATA[31:0]	output	INITEXP0, HRDATA
INITEXP0HREADY	output	INITEXP0, HREADY
INITEXP0HRESP	output	INITEXP0, HRESP
INITEXP0EXRESP	output	INITEXP0,EXRESP
INITEXP0HRUSER[2:0]	output	INITEXP0, HRUSER
APBTARGEXP2PSTRB[3:0]	output	APBTARGEXP2, PSTRB
APBTARGEXP2PPROT[2:0]	output	APBTARGEXP2, PPROT
APBTARGEXP2PSEL	output	APBTARGEXP2, PSELx
APBTARGEXP2PENABLE	output	APBTARGEXP2, PENABLE
APBTARGEXP2PADDR[11:0]	output	APBTARGEXP2, PADDR
APBTARGEXP2PWRITE	output	APBTARGEXP2, PWRITE
APBTARGEXP2PWDATA[31:0]	output	APBTARGEXP2, PWDATA
DAPTDO	output	Debug TDO
DAPJTAGNSW	output	JTAG or Serial-Wire selection JTAG mode(1) or SW mode(0)
DAPNTDOEN	output	TDO output pad control signal
TPIUTRACEDATA[3:0]	output	Output data
TPIUTRACECLK	output	Output clock, used by the TPA to sample the other pins

原语例化

Verilog 例化：

```
MCU u_sse050_top_syn (
  .FCLK(fclk),
  .PORESETN(poresetn),
  .SYSRESETN(sysresetn),
  .RTCSRCCLK(rtcsrcclk),
  .IOEXPINPUTI(ioexpinputi[15:0]),
  .IOEXPOUTPUTO(ioexpoutputo[15:0]),
  .IOEXPOUTPUTENO(ioexpoutputeno[15:0]),
  .UART0RXDI(uart0rxdi),
  .UART0TXDO(uart0txdo),
  .UART1RXDI(uart1rxdi),
  .UART1TXDO(uart1txdo),
  .SRAM0RDATA(sram0rdata[31:0]),
  .SRAM0ADDR(sram0addr[12:0]),
  .SRAM0WREN(sram0wren[3:0]),
  .SRAM0WDATA(sram0wdata[31:0]),
  .SRAM0CS(sram0cs),
  .MTXHRESETN(mtxhreset),
  .TARGFLASH0HSEL(targflash0hsel),
  .TARGFLASH0HADDR(targflash0haddr[28:0]),
  .TARGFLASH0HTRANS(targflash0htrans[1:0]),
  .TARGFLASH0HSIZE(targflash0hsize[2:0]),
  .TARGFLASH0HBURST(targflash0hburst[2:0]),
  .TARGFLASH0HREADYMUX(targflash0hreadymux),
  .TARGFLASH0HRDATA(targflash0hrdata[31:0]),
  .TARGFLASH0HRUSER(targflash0hruser[2:0]),
  .TARGFLASH0HRESP(targflash0hresp),
  .TARGFLASH0EXRESP(targflash0exresp),
  .TARGFLASH0HREADYOUT(targflash0hreadyout),
  .TARGEXP0HSEL(targexp0hsel),
  .TARGEXP0HADDR(targexp0haddr[31:0]),
  .TARGEXP0HTRANS(targexp0htrans[1:0]),
  .TARGEXP0HWRITE(targexp0hwrite),
  .TARGEXP0HSIZE(targexp0hsize[2:0]),
  .TARGEXP0HBURST(targexp0hburst[2:0]),
  .TARGEXP0HPROT(targexp0hprot[3:0]),
  .TARGEXP0MEMATTR(targexp0memattr[1:0]),
  .TARGEXP0EXREQ(targexp0exreq),
  .TARGEXP0HMASTER(targexp0hmaster[3:0]),
  .TARGEXP0HWDATA(targexp0hwdata[31:0]),
  .TARGEXP0HMASTLOCK(targexp0hmastlock),
  .TARGEXP0HREADYMUX(targexp0hreadymux),
  .TARGEXP0HAUSER(targexp0hauser),
  .TARGEXP0HWUSER(targexp0hwuser[3:0]),
  .TARGEXP0HRDATA(targexp0hrdata[31:0]),
  .TARGEXP0HREADYOUT(targexp0hreadyout),
  .TARGEXP0HRESP(targexp0hresp),
```

```
  .TARGEXP0EXRESP(targexp0exresp),
  .TARGEXP0HRUSER(targexp0hruser[2:0]),
  .INITEXP0HSEL(initexp0hsel),
  .INITEXP0HADDR(initexp0haddr[31:0]),
  .INITEXP0HTRANS(initexp0htrans[1:0]),
  .INITEXP0HWRITE(initexp0hwrite),
  .INITEXP0HSIZE(initexp0hsize[2:0]),
  .INITEXP0HBURST(initexp0hburst[2:0]),
  .INITEXP0HPROT(initexp0hprot[3:0]),
  .INITEXP0MEMATTR(initexp0mematttr[1:0]),
  .INITEXP0EXREQ(initexp0exreq),
  .INITEXP0HMASTER(initexp0hmaster[3:0]),
  .INITEXP0HWDATA(initexp0hwdata[31:0]),
  .INITEXP0HMASTLOCK(initexp0hmastlock),
  .INITEXP0HAUSER(initexp0hauser),
  .INITEXP0HWUSER(initexp0hwuser[3:0]),
  .INITEXP0HRDATA(initexp0hrdata[31:0]),
  .INITEXP0HREADY(initexp0hready),
  .INITEXP0HRESP(initexp0hresp),
  .INITEXP0EXRESP(initexp0exresp),
  .INITEXP0HRUSER(initexp0hruser[2:0]),
  .APBTARGEXP2PSEL(apbtargexp2psel),
  .APBTARGEXP2PENABLE(apbtargexp2penable),
  .APBTARGEXP2PADDR(apbtargexp2paddr[11:0]),
  .APBTARGEXP2PWRITE(apbtargexp2pwrite),
  .APBTARGEXP2PWDATA(apbtargexp2pwdata[31:0]),
  .APBTARGEXP2PRDATA(apbtargexp2prdata[31:0]),
  .APBTARGEXP2PREADY(apbtargexp2pready),
  .APBTARGEXP2PSLVERR(apbtargexp2pslverr),
  .APBTARGEXP2PSTRB(apbtargexp2pstrb[3:0]),
  .APBTARGEXP2PPROT(apbtargexp2pprot[2:0]),
  .MTXREMAP(mtxremap[3:0]),
  .DAPSWDITMS(dapswditms),
  .DAPTDI(daptdi),
  .DAPTDO(daptdo),
  .DAPNTRST(dapntrst),
  .DAPSWCLKTCK(dapswclk_tck),
  .DAPNTDOEN(dapntdoen),
  .DAPJTAGNSW(dapjtagnsw),
  .TPIUTRACEDATA(tpiutracedata[3:0]),
  .TPIUTRACECLK(tpiutraceclk),
  .FLASHERR(flasherr),
   .GPINT(gpint),
  .FLASHINT(flashint)
);
```

Vhdl 例化：

```
COMPONENT MCU
        PORT(
FCLK:IN std_logic;
PORESETN:IN std_logic;
```

```
SYSRESETN:IN std_logic;
RTCSRCCLK:IN std_logic;
UART0RXDI:IN std_logic;
UART1RXDI:IN std_logic;
CLK:IN std_logic;
RESET:IN std_logic;
IOEXPINPUTI:IN std_logic_vector(15 downto 0);
SRAM0RDATA:IN std_logic_vector(31 downto 0);
TARGFLASH0HRDATA:IN std_logic_vector(31 downto 0);
TARGFLASH0HRUSER:IN std_logic_vector(2 downto 0);
TARGFLASH0HRESP:IN std_logic;
TARGFLASH0EXRESP:IN std_logic;
TARGFLASH0HREADYOUT:IN std_logic;
TARGEXP0HRDATA: IN std_logic_vector(31 downto 0);
TARGEXP0HREADYOUT:IN std_logic;
TARGEXP0HRESP:IN std_logic;
TARGEXP0EXRESP:IN std_logic;
TARGEXP0HRUSER: IN std_logic_vector(2 downto 0);
INITEXP0HSEL:IN std_logic;
INITEXP0HADDR: IN std_logic_vector(31 downto 0);
INITEXP0HTRANS: IN std_logic_vector(1 downto 0);
INITEXP0HWRITE: IN std_logic;
INITEXP0HSIZE: IN std_logic_vector(2 downto 0);
INITEXP0HBURST: IN std_logic_vector(2 downto 0);
INITEXP0HPROT: IN std_logic_vector(3 downto 0);
INITEXP0MEMATTR: IN std_logic_vector(1 downto 0);
INITEXP0EXREQ: IN std_logic;
INITEXP0HMASTER: IN std_logic_vector(3 downto 0);
INITEXP0HWDATA: IN std_logic_vector(31 downto 0);
INITEXP0HMASTLOCK: IN std_logic;
INITEXP0HAUSER: IN std_logic;
INITEXP0HWUSER: IN std_logic_vector(3 downto 0);
APBTARGEXP2PRDATA: IN std_logic_vector(3 downto 0);
APBTARGEXP2PREADY: IN std_logic;
APBTARGEXP2PSLVERR: IN std_logic;
MTXREMAP: IN std_logic_vector(3 downto 0);
DAPSWDITMS: IN std_logic;
DAPTDI: IN std_logic;
DAPNTRST: IN std_logic;
DAPSWCLKTCK: IN std_logic;
FLASHERR: IN std_logic;
FLASHINT: IN std_logic;
GPINT: IN std_logic;
IOEXPOUTPUTO:OUT std_logic_vector(15 downto 0);
IOEXPOUTPUTENO:OUT std_logic_vector(15 downto 0);
IOEXPINPUTI:OUT std_logic_vector(15 downto 0);
UART0TXDO: OUT std_logic;
UART1TXDO: OUT std_logic;
UART0BAUDTICK: OUT std_logic;
UART1BAUDTICK: OUT std_logic;
```

```
        INTMONITOR: OUT std_logic;
        MTXHRESETN: OUT std_logic;
        SRAM0ADDR:OUT std_logic_vector(12 downto 0);
        SRAM0WREN:OUT std_logic_vector(3 downto 0);
        SRAM0WDATA:OUT std_logic_vector(31 downto 0);
        SRAM0CS: OUT std_logic;
        TARGFLASH0HSEL: OUT std_logic;
        TARGFLASH0HREADYMUX: OUT std_logic;
        SRAM0RDATA:OUT std_logic_vector(31 downto 0);
        TARGFLASH0HADDR:OUT std_logic_vector(28 downto 0);
        TARGFLASH0HTRANS:OUT std_logic_vector(1 downto 0);
        TARGFLASH0HSIZE:OUT std_logic_vector(2 downto 0);
        TARGFLASH0HBURST:OUT std_logic_vector(2 downto 0);
        TARGFLASH0HRDATA:OUT std_logic_vector(31 downto 0);
        TARGEXP0HADDR:OUT std_logic_vector(31 downto 0);
        TARGEXP0HSEL: OUT std_logic;
        TARGEXP0HWRITE: OUT std_logic;
        TARGEXP0EXREQ: OUT std_logic;
        TARGEXP0HMASTLOCK: OUT std_logic;
        TARGEXP0HREADYMUX: OUT std_logic;
        TARGEXP0HAUSER: OUT std_logic;
        INITEXP0HREADY: OUT std_logic;
        INITEXP0HRESP: OUT std_logic;
        INITEXP0EXRESP: OUT std_logic;
        TARGEXP0HTRANS:OUT std_logic_vector(1 downto 0);
        TARGEXP0HSIZE:OUT std_logic_vector(2 downto 0);
        TARGEXP0HBURST:OUT std_logic_vector(2 downto 0);
        TARGEXP0HPROT:OUT std_logic_vector(3 downto 0);
        TARGEXP0MEMATTR:OUT std_logic_vector(1 downto 0);
        TARGEXP0HMASTER:OUT std_logic_vector(3 downto 0);
        TARGEXP0HWDATA:OUT std_logic_vector(31 downto 0);
        TARGEXP0HWUSER:OUT std_logic_vector(3 downto 0);
        INITEXP0HRDATA:OUT std_logic_vector(31 downto 0);
        INITEXP0HRUSER:OUT std_logic_vector(2 downto 0);
        APBTARGEXP2PSTRB:OUT std_logic_vector(3 downto 0);
        APBTARGEXP2PPROT:OUT std_logic_vector(2 downto 0);
        APBTARGEXP2PADDR:OUT std_logic_vector(11 downto 0);
        APBTARGEXP2PWDATA:OUT std_logic_vector(31 downto 0);
        TPIUTRACEDATA:OUT std_logic_vector(3 downto 0);
        APBTARGEXP2PSEL: OUT std_logic;
        APBTARGEXP2PENABLE: OUT std_logic;
        APBTARGEXP2PWRITE: OUT std_logic;
        DAPTDO: OUT std_logic;
        DAPJTAGNSW: OUT std_logic;
        DAPNTDOEN: OUT std_logic;
        TPIUTRACECLK: OUT std_logic;
    );
        END COMPONENT;

        uut: MCU
```

```
        PORT MAP (
FCLK=> fclk;
PORESETN=> poresetn;
SYSRESETN=> sysresetn;
RTCSRCCLK=> rtcsrcclk;
UART0RXDI=> uart0rxdi;
UART1RXDI=> uart1rxdi;
CLK=>clk,
RESET=>reset,
IOEXPINPUTI=>ioexpinputi,
SRAM0RDATA=>sram0rdata,
TARGFLASH0HRDATA=>targflash0hrdata,
TARGFLASH0HRUSER=>targflash0hruser,
TARGFLASH0HRESP=>targflash0hresp,
TARGFLASH0EXRESP=>targflash0exresp,
TARGFLASH0HREADYOUT=>targflash0hreadyout,
TARGEXP0HRDATA=>targexp0hrdata,
TARGEXP0HREADYOUT=>targexp0hreadyout,
TARGEXP0HRESP=>targexp0hresp,
TARGEXP0EXRESP=>targexp0exresp,
TARGEXP0HRUSER=>targexp0hruser,
INITEXP0HSEL=>initexp0hsel,
INITEXP0HADDR=>initexp0haddr,
INITEXP0HTRANS=>initexp0htrans,
INITEXP0HWRITE=>initexp0hwrite,
INITEXP0HSIZE=>initexp0hsize,
INITEXP0HBURST=>initexp0hburst,
INITEXP0HPROT=>initexp0hprot,
INITEXP0MEMATTR=>initexp0memattr,
INITEXP0EXREQ=>initexp0exreq,
INITEXP0HMASTER=>initexp0hmaster,
INITEXP0HWDATA=>initexp0hwdata,
INITEXP0HMASTLOCK=>initexp0hmastlock,
INITEXP0HAUSER=>initexp0hauser,
INITEXP0HWUSER=>initexp0hwuser,
APBTARGEXP2PRDATA=>apbtargexp2prdata,
APBTARGEXP2PREADY=>apbtargexp2pready,
APBTARGEXP2PSLVERR=>apbtargexp2pslverr,
MTXREMAP=>mtxremap,
DAPSWDITMS=>dapswditms,
DAPTDI=>daptdi,
DAPNTRST=>dapntrst,
DAPSWCLKTCK=>dapswclktck,
FLASHERR=>flasherr,
FLASHINT=>flashint,
GPINT=>gpint,
IOEXPOUTPUTO=>ioexpoutputo,
IOEXPOUTPUTENO=>ioexpoutputeno,
IOEXPINPUTI=>ioexpinputi,
UART0TXDO=>uart0txdo,
```

```
UART1TXDO=>uart1txdo,
UART0BAUDTICK=>uart0baudtick,
UART1BAUDTICK=>uart1baudtick,
INTMONITOR=>intmonitor,
MTXHRESETN=>mtxhresetn,
SRAM0ADDR=>sram0addr,
SRAM0WREN=>sram0wren,
SRAM0WDATA=>sram0wdata,
SRAM0CS=>sram0cs,
TARGFLASH0HSEL=>targflash0hsel,
TARGFLASH0HREADYMUX=>targflash0hreadymux,
SRAM0RDATA=>sram0rdata,
TARGFLASH0HADDR=>targflash0haddr,
TARGFLASH0HTRANS=>targflash0htrans,
TARGFLASH0HSIZE=>targflash0hsize,
TARGFLASH0HBURST=>targflash0hburst,
TARGFLASH0HRDATA=>targflash0hrdata,
TARGEXP0HADDR=>targexp0haddr,
TARGEXP0HSEL=>targexp0hsel,
TARGEXP0HWRITE=>targexp0hwrite,
TARGEXP0EXREQ=>targexp0exreq,
TARGEXP0HMASTLOCK=>targexp0hmastlock,
TARGEXP0HREADYMUX=>targexp0hreadymux,
TARGEXP0HAUSER=>targexp0hauser,
INITEXP0HREADY=>initexp0hready,
INITEXP0HRESP=>initexp0hresp,
INITEXP0EXRESP=>initexp0exresp,
TARGEXP0HTRANS=>targexp0htrans,
TARGEXP0HSIZE=>targexp0hsize,
TARGEXP0HBURST=>targexp0hburst,
TARGEXP0HPROT=>targexp0hprot,
TARGEXP0MEMATTR=>targexp0memattr,
TARGEXP0HMASTER=>targexp0hmaster,
TARGEXP0HWDATA=>targexp0hwdata,
TARGEXP0HWUSER=>targexp0hwuser,
INITEXP0HRDATA=>initexp0hrdata,
INITEXP0HRUSER=>initexp0hruser,
APBTARGEXP2PSTRB=>apbtargexp2pstrb,
APBTARGEXP2PPROT=>apbtargexp2pprot,
APBTARGEXP2PADDR=>apbtargexp2paddr,
APBTARGEXP2PWDATA=>apbtargexp2pwdata,
TPIUTRACEDATA=>tpiutracedata,
APBTARGEXP2PSEL=>apbtargexp2psel,
APBTARGEXP2PENABLE=>apbtargexp2penable,
APBTARGEXP2PWRITE=>apbtargexp2pwrite,
DAPTDO=>daptdo,
DAPJTAGNSW=>dapjtagnsw,
DAPNTDOEN=>dapntdoen,
TPIUTRACECLK=>tpiutraceclk );
```

9 SPMI 和 I3C

9.1 SPMI

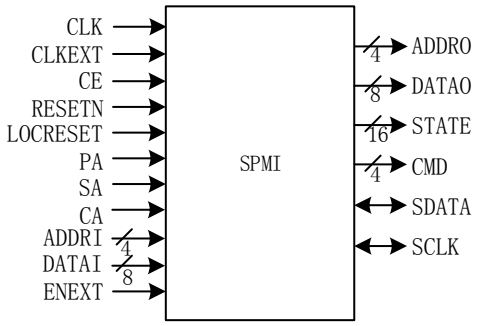
原语介绍

SPMI（System Power Management Interface）是一种双线串行接口，可用于动态控制片上系统内部电源的关断与开启。

支持器件：GW1NZ-1。

结构框图

图 9-1 SPMI 结构框图



Port 介绍

表 9-1 Port 介绍

Port Name	I/O	Description
CLK	input	Clock input
CLKEXT	input	External clock input
CE	input	Clock Enable
RESETN	input	Reset input
ENEXT	input	Enext input
LOCRESET	input	Local reset input
PA	input	Priority arbitration input
SA	input	Secondary arbitration input
CA	input	Connection arbitration input
ADDRI	input	Addr input
DATAI	input	Data input

Port Name	I/O	Description
ADDRO	output	Addr output
DATAO	output	datat output
STATE	output	state output
CMD	output	command output
SDATA	inout	SPMI Serial data
SCLK	inout	SPMI Serial Clock

原语例化

Verilog 例化：

```
SPMI uut (
        .ADDRO(addro),
        .DATAO(datao),
        .STATE(state),
        .CMD(cmd),
        .SDATA(sdata),
        .SCLK(sclk),
        .CLK(clk),
        .CE(ce),
        .RESETN(resetn),
        .LOCRESET(locreset),
        .PA(pa),
        .SA(sa),
        .CA(ca),
        .ADDRI(addri),
        .DATAI(datai),
        .CLKEXT(clkext),
        .ENEXT(enext)
);
```

Vhdl 例化：

```
COMPONENT SPMI
        PORT(
            CLK:IN std_logic;
            CLKEXT:IN std_logic;
            CE:IN std_logic;
            RESETN:IN std_logic;
            ENEXT:IN std_logic;
            LOCRESET:IN std_logic;
            PA:IN std_logic;
            SA:IN std_logic;
            CA:IN std_logic;
            ADDRI:IN std_logic_vector(3 downto 0);
            DATAI:IN std_logic_vector(7 downto 0);
            ADDRO:OUT std_logic_vector(3 downto 0);
            DATAO:OUT std_logic_vector(7 downto 0);
            STATE:OUT std_logic_vector(15 downto 0);
            CMD:OUT std_logic_vector(3 downto 0);
            SDATA:INOUT std_logic;
```

```
            SCLK:INOUT std_logic
        );
    END COMPONENT;
    uut: SPMI
        PORT MAP (
            CLK=>clk,
            CLKEXT=>clkext,
            CE=>ce,
            RESETN=>resetn,
            ENEXT=>enext,
            LOCRESET=>locreset,
            PA=>pa,
            SA=>sa,
            CA=>ca,
            ADDRI=>addri,
            DATAI=>datai,
            ADDRO=>addro,
            DATAO=>datao,
            STATE=>state,
            CMD=>cmd,
            SDATA=>sdata,
            SCLK=>sclk
        );
```

9.2 I3C

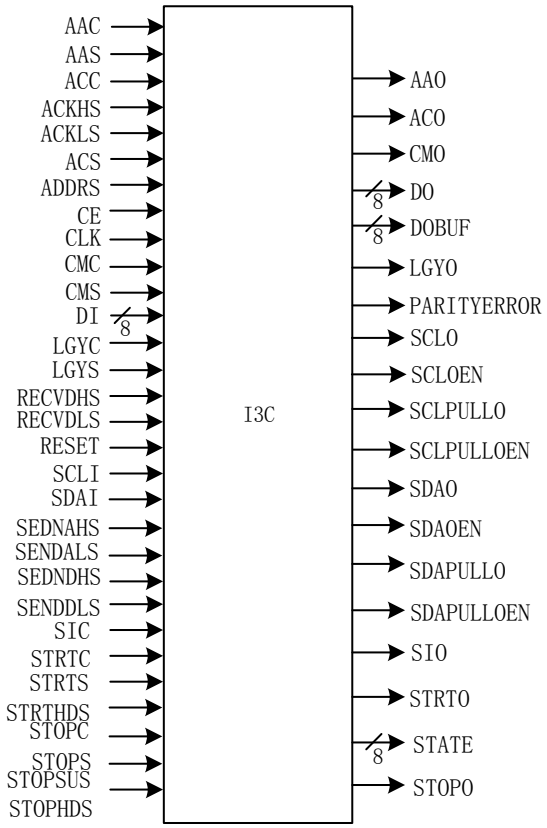
原语介绍

I3C (Improved Inter Integrated Circuit)是一种两线式总线，兼具了 I2C 和 SPI 的关键特性，能有效的减少集成电路芯片系统的物理端口、支持低功耗、高数据速率和其他已有端口协议的优点。

支持器件：GW1NZ-1。

结构框图

图 9-2 I3C 结构框图



Port 介绍

表 9-2 Port 介绍

Port Name	I/O	Description
CE	input	Clock Enable
RESET	input	Reset input
CLK	input	Clock input
LGYS	input	The current communication object is the I2C setting signal
CMS	input	The device enters the Master's set signal
ACS	input	Select the setting signal when determining whether to continue.
AAS	input	Reply the ACK setting signal when a reply is required from the ACK/NACK
STOPS	input	Input the STOP command
STRTS	input	Input the START command.
LGYC	input	The current communication object is the I2C
CMC	input	The reset signal that the device is in master.
ACC	input	The reset signal that selects continue when selecting whether to continue
AAC	input	Reply the ACK reset signal when a reply is required from the ACK/NACK
SIC	input	Interrupt to identify the reset signal
STOPC	input	The reset signal is in STOP state
STRTC	input	The reset signal is in START state

Port Name	I/O	Description
STRTHDS	input	Adjust the setting signal when generating START
SENDAHS	input	Adjust the setting signal of SCL at a high level when the address is sent.
SENDALS	input	Adjust the setting signal of SCL at a low level when the address is sent
ACKHS	input	Adjust the setting signal of SCL at a high level in ACK.
SENDDLS	input	Adjust the setting signal of SCL at a low level in ACK.
RECVDHS	input	Adjust the setting signal of SCL at a high level when the data are received
RECVDLS	input	Adjust the setting signal of SCL at a low level when the data are received
ADDRS	input	The slave address setting interface
DI	input	Data Input.
SDAI	input	I3C serial data input
SCLI	input	I3C serial clock input
LGYO	output	Output the current communication object as the I2C command.
CMO	output	Output the command of the device is in the Master mode.
ACO	output	Continue to output when selecting whether to continue
AAO	output	Reply ACK when you need to reply ACK/NACK
SIO	output	Interrupt to output the identity bit
STOPO	output	Output the STOP command
STRTO	output	Output the START command
PARITYERROR	output	Output check when receiving data
DOBUF	output	Data output after caching
DO	output	Data output directly
STATE	output	Output the internal state
SDAO	output	I3C serial data output
SCLO	output	I3C serial clock output
SDAOEN	output	I3C serial data oen output
SCLOEN	output	I3C serial clock oen output
SDAPULLO	output	Controllable pull-up of the I3C serial data
SCLPULLO	output	Controllable pull-up of the I3C serial clock
SDAPULLOEN	output	Controllable pull-up of the I3C serial data oen
SCLPULLOEN	output	Controllable pull-up of the I3C serial clock oen

原语例化

Verilog 例化：

```
I3C i3c_inst (
    .LGYO(lgyo),
    .CMO(cmo),
    .ACO(aco),
    .AAO(aao),
    .SIO(sio),
    .STOPO(stopo),
    .STRTO(strto),
    .PARITYERROR(parityerror),
```

```
        .DOBUF(dobuf),
        .DO(dout),
        .STATE(state),
        .SDAO(sdao),
        .SCLO(sclo),
        .SDAOEN(sdaoen),
        .SCLOEN(scloen),
        .SDAPULLO(sdapullo),
        .SCLPULLO(sclpullo),
        .SDAPULLOEN(sdapulloen),
        .SCLPULLOEN(sclpulloen),
        .LGYS(lgys),
        .CMS(cms),
        .ACS(acs),
        .AAS(aas),
        .STOPS(stops),
        .STRTS(strts),
        .LGYC(lgyc),
        .CMC(cmc),
        .ACC(acc),
        .AAC(aac),
        .SIC(sic),
        .STOPC(stopc),
        .STRTC(strtc),
        .STRTHDS(strthds),
        .SENDAHS(sendahs),
        .SENDALS(sendals),
        .ACKHS(ackhs),
        .ACKLS(ackls),
        .STOPSUS(stopsus),
        .STOPHDS(stophds),
        .SENDDHS(senddhs),
        .SENDDLS(senddls),
        .RECVDHS(recvdhs),
        .RECVDLS(recvdls),
        .ADDRS(addrs),
        .DI(di),
        .SDAI(sdai),
        .SCLI(scli),
        .CE(ce),
        .RESET(reset),
        .CLK(clk)
    );
```

10 Miscellaneous

10.1 GSR

原语名称

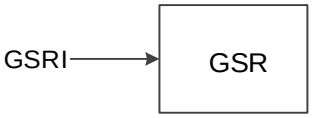
GSR(Global Reset/Set)， 全局复位/置位模块。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 10-1 GSR 端口示意图



功能描述

GSR 模块，可以实现全局复位/置位功能，低电平有效。默认一般连接高电平，若想动态控制，可连接外部信号，拉低实现寄存器等模块的复位。

原语定义

端口介绍

表 10-1 端口介绍

端口名	I/O	描述
GSRI	Input	GSR 输入，低电平有效

原语例化

Verilog 例化：

```
GSR gsr_inst(
      .GSRI(GSRI)
);
```

Vhdl 例化：

```
COMPONENT GSR
      PORT (
            GSRI:IN std_logic
      );
END COMPONENT;
gsr_inst:GSR
        PORT MAP(
            GSRI => GSRI
         );
```

10.2 INV

原语名称

INV(Inverter)， 取反模块。

适用器件

支持器件：GW1N-1、GW1N-1S、GW1NZ-1、GW1N-2、GW1N-2B、GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C、GW1N-4、GW1N-4B、GW1NR-4、GW1NR-4B、GW1NRF-4B、GW1NS-4、GW1NSR-4、GW1NSR-4C、GW1NSER-4C、GW1N-6、GW1N-9、GW1NR-9、GW2A-18、GW2AR-18、GW2A-55、GW2A-55C。

端口示意图

图 10-2 INV 端口示意图

功能描述

INV 模块，可以实现取反功能。

原语定义

端口介绍

表 10-2 端口介绍

端口名	I/O	描述
I	Input	INV 数据输入
O	Output	INV 数据输出

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT INV
      PORT (
            O:OUTPUT std_logic;
            I:IN std_logic

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

10.3 BANDGAP

原语名称

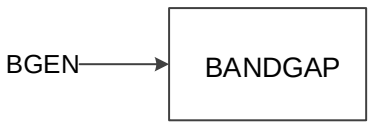
BANDGAP。

适用器件

支持器件：GW1NZ-1。

端口示意图

图 10-3 BANDGAP 端口示意图



功能描述

在 GW1NZ-1 器件中，BANDGAP 的功能是为芯片中的某些模块提供恒定的电压和电流，若关掉 BANDGAP，则 OSC、PLL、FLASH 等模块将不再工作，可以起到降低器件功耗的作用。

原语定义

端口介绍

表 10-3 端口介绍

端口名	I/O	描述
BGEN	Input	BANDGAP 使能信号，高电平有效。

原语例化

Verilog 例化：

```
BANDGAP uut (
      .BGEN(bgen)
);
```

Vhdl 例化：

```
COMPONENT BANDGAP
      PORT (
            BGEN:IN std_logic
      );
END COMPONENT;
uut:BANDGAP
        PORT MAP(
            BGEN=> I
        );
```


GOWIN高云
智慧逻辑 定制未来