Gowin 原语用户指南

SUG283-2.4, 2020-09-11

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； ● 增加 FF、LATCH 放置规则； ● 增加支持器件 GW2A-55C；

日期	版本	说明
		● 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。
2020/03/09	2.2	● GW1NS-2、GW1NS-2C、GW1NSR-2、GW1NSR-2C、GW1NSE-2C 禁掉 DP/DPX9、DPB/DPX9B； ● OSCF 补充 OSCEN 端口说明； ● 更新 PLL/rPLL/PLLVR 参数说明。
2020/06/08	2.3	● 删除器件 GW1N-2、GW1N-2B、GW1N-6； ● 增加器件 GW1N-9C、GW1NR-9C； ● 增加 IODELAYC、DHCENC、DCC； ● 增加 MIPI_IBUF 功能描述。 ● 删除 MIPI_IBUF_HS、MIPI_IBUF_LP、DLL； ● 增加 LUT5、MUX8 端口示意图； ● 增加 VCC、GND； ● 更新 PLLVR、FLASH64K、BUFS、EMPU、CLKDIV2 原语介绍； ● 删除 DP/DPX9、ROM/ROMX9、SDP/SDPX9、rSDP/rSDPX9、rROM/rROMX9、PLL； ● 调整 Iologic 结构顺序，统一端口示意图标题名称。
2020/09/11	2.4	添加 ADC, BANDGAP, SPMI, I3C 模块 IP 调用说明。

目录

图目录

表目录

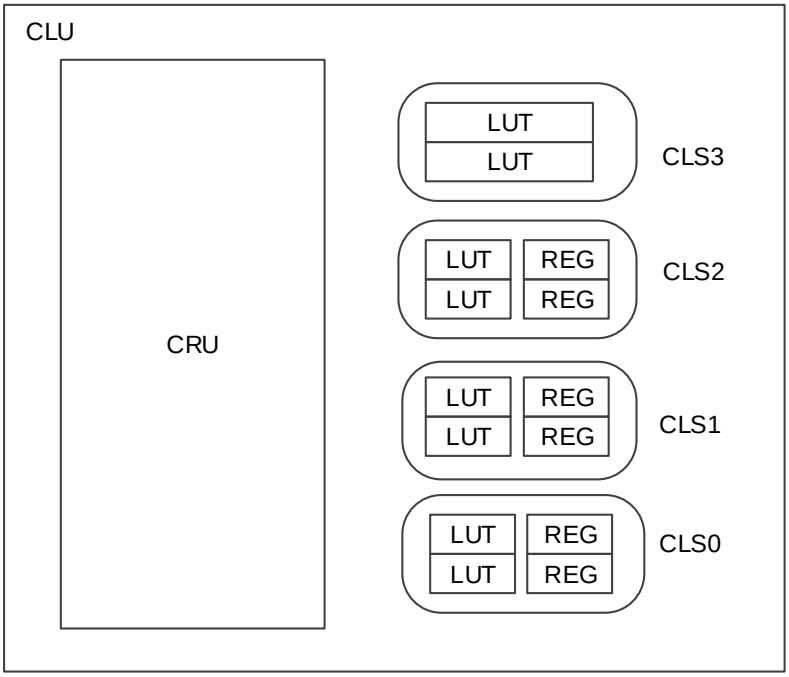
1 IOB

IOB 主要包括输入输出缓存(IO Buffer)、输入输出逻辑(IO Logic)，其中 IO Buffer 及 IO Logic 原语可参考 UG289，Gowin 可编程通用管脚(GPIO)用户指南。

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 由四个可配置逻辑片 CLS(Configurable Logic Slice)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。

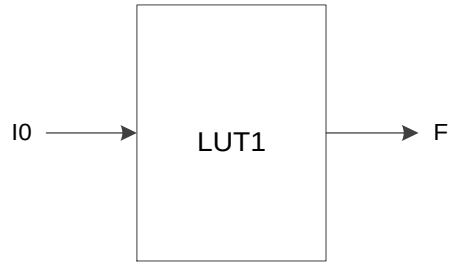
2.1.1 LUT1

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。

端口示意图

图 2-2 LUT1 端口示意图



端口介绍

表 2-1 LUT1 端口介绍

端口	I/O	描述
I0	Input	数据输入
F	Output	数据输出

参数介绍

表 2-2 LUT1 参数介绍

参数	范围	默认	描述
INIT	2'h0~2'h3	2'h0	LUT1 初始值

真值表

表 2-3 LUT1 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
       );
```

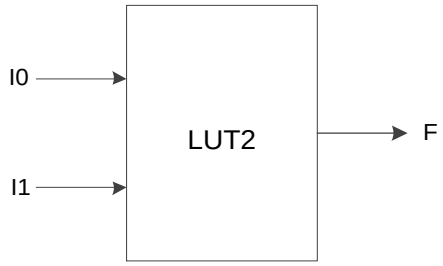
2.1.2 LUT2

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。

端口示意图

图 2-3 LUT2 端口示意图



端口介绍

表 2-4 LUT2 端口介绍

端口	I/O	描述
I0	Input	数据输入
I1	Input	数据输入
F	Output	数据输出

参数介绍

表 2-5 LUT2 参数介绍

参数	范围	默认	描述
INIT	4'h0~4'hf	4'h0	LUT2 初始值

真值表

表 2-6 LUT2 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
       );
```

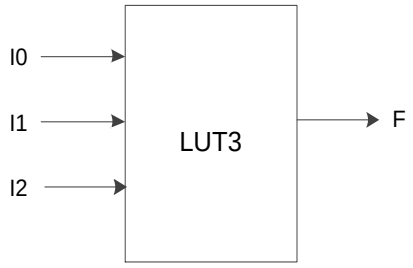
2.1.3 LUT3

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。

端口示意图

图 2-4 LUT3 端口示意图



端口介绍

表 2-7 LUT3 端口介绍

端口	I/O	描述
I0	Input	数据输入
I1	Input	数据输入
I2	Input	数据输入
F	Output	数据输出

参数介绍

表 2-8 LUT3 参数介绍

参数	范围	默认	描述
INIT	8'h00~8'hff	8'h00	LUT3 初始值

真值表

表 2-9 LUT3 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
       );
```

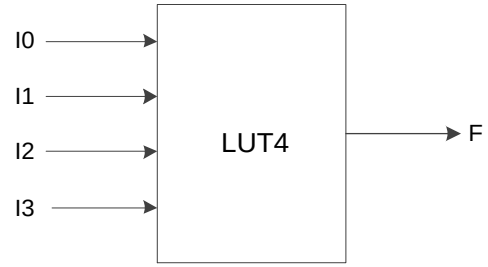
2.1.4 LUT4

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。

端口示意图

图 2-5 LUT4 端口示意图



端口介绍

表 2-10 LUT4 端口介绍

端口	I/O	描述
I0	Input	数据输入
I1	Input	数据输入
I2	Input	数据输入
I3	Input	数据输入
F	Output	数据输出

参数介绍

表 2-11 LUT4 参数介绍

参数	范围	默认	描述
INIT	16'h0000~16'hffff	16'h0000	LUT4 初始值

真值表

表 2-12 LUT4 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]

Input(I3)	Input(I2)	Input(I1)	Input(I0)	Output(F)
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

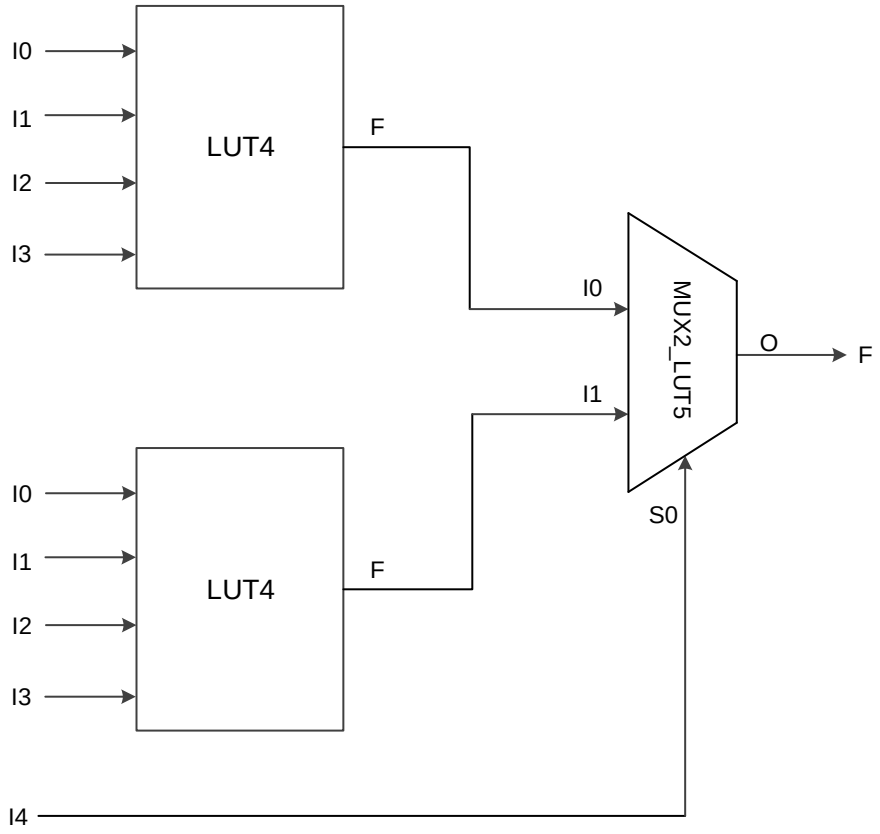
Wide LUT 是通过 LUT4 和 MUX2 构造高阶 LUT，高云 FPGA 目前支持的构造高阶 LUT 的 MUX2 有 MUX2_LUT5/ MUX2_LUT6/ MUX2_LUT7/ MUX2_LUT8。

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 LUT5 为例介绍 Wide LUT 的使用。

端口示意图

图 2-6 LUT5 端口示意图



端口介绍

表 2-13 LUT5 端口介绍

端口名	I/O	描述
I0	Input	数据输入
I1	Input	数据输入
I2	Input	数据输入

端口名	I/O	描述
I3	Input	数据输入
I4	Input	数据输入
F	Output	数据输出

参数介绍

表 2-14 LUT5 参数介绍

参数	范围	默认	描述
INIT	32'h00000~32'hfffff	32'h00000	LUT5 初始值

真值表

表 2-15 LUT5 真值表

Input(I4)	Input(I3)	Input(I2)	Input(I1)	Input(I0)	Output(F)
0	0	0	0	0	INIT[0]
0	0	0	0	1	INIT[1]
0	0	0	1	0	INIT[2]
0	0	0	1	1	INIT[3]
0	0	1	0	0	INIT[4]
0	0	1	0	1	INIT[5]
0	0	1	1	0	INIT[6]
0	0	1	1	1	INIT[7]
0	1	0	0	0	INIT[8]
0	1	0	0	1	INIT[9]
0	1	0	1	0	INIT[10]
0	1	0	1	1	INIT[11]
0	1	1	0	0	INIT[12]
0	1	1	0	1	INIT[13]
0	1	1	1	0	INIT[14]
0	1	1	1	1	INIT[15]
1	0	0	0	0	INIT[16]
1	0	0	0	1	INIT[17]
1	0	0	1	0	INIT[18]
1	0	0	1	1	INIT[19]
1	0	1	0	0	INIT[20]
1	0	1	0	1	INIT[21]
1	0	1	1	0	INIT[22]
1	0	1	1	1	INIT[23]
1	1	0	0	0	INIT[24]

Input(I4)	Input(I3)	Input(I2)	Input(I1)	Input(I0)	Output(F)
1	1	0	0	1	INIT[25]
1	1	0	1	0	INIT[26]
1	1	0	1	1	INIT[27]
1	1	1	0	0	INIT[28]
1	1	1	0	1	INIT[29]
1	1	1	1	0	INIT[30]
1	1	1	1	1	INIT[31]

原语例化

Verilog 例化：

```
MUX2_LUT5 instName (
    .I0(f0),
    .I1(f1),
    .S0(i4),
    .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=>i4
        );
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。

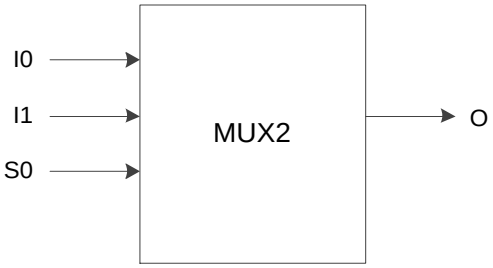
2.2.1 MUX2

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。

端口示意图

图 2-7 MUX2 端口示意图



端口介绍

表 2-16 MUX2 端口介绍

端口	I/O	描述
I0	Input	数据输入
I1	Input	数据输入
S0	Input	选择信号
O	Output	数据输出

真值表

表 2-17 MUX2 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
        );
```

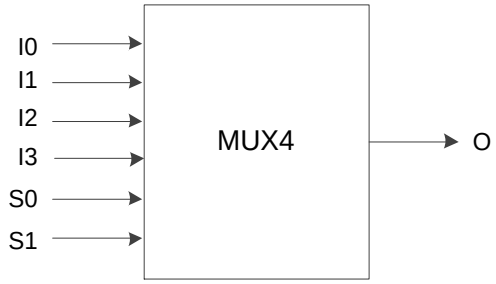
2.2.2 MUX4

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。

端口示意图

图 2-8 MUX4 端口示意图



端口介绍

表 2-18 MUX4 端口介绍

端口	I/O	描述

I0	Input	数据输入
I1	Input	数据输入
I2	Input	数据输入
I3	Input	数据输入
S0	Input	选择信号
S1	Input	选择信号
O	Output	数据输出

真值表

表 2-19 MUX4 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

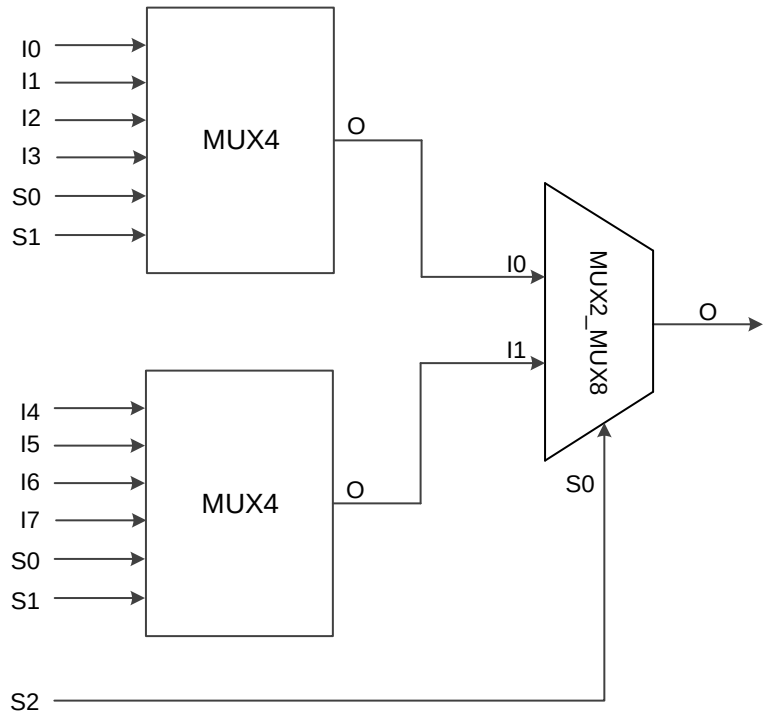
Wide MUX 是通过 MUX4 和 MUX2 构造高阶 MUX，高云 FPGA 目前支持的构造高阶 MUX 的 MUX2 有 MUX2_MUX8/ MUX2_MUX16/ MUX2_MUX32。

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 MUX8 为例介绍 Wide MUX 的使用。

端口示意图

图 2-9 MUX8 端口示意图



端口介绍

表 2-20 MUX8 端口介绍

端口	输入/输出	描述
I0	Input	数据输入
I1	Input	数据输入
I2	Input	数据输入
I3	Input	数据输入
I4	Input	数据输入
I5	Input	数据输入
I6	Input	数据输入
I7	Input	数据输入
S0	Input	选择信号
S1	Input	选择信号
S2	Input	选择信号
O	Output	数据输出

真值表

表 2-21 MUX8 真值表

Input(S2)	Input(S1)	Input(S0)	Output(O)
0	0	0	I0
0	0	1	I1
0	1	0	I2
0	1	1	I3
1	0	0	I4
1	0	1	I5
1	1	0	I6
1	1	1	I7

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。

具体功能如表 2-22 所示。

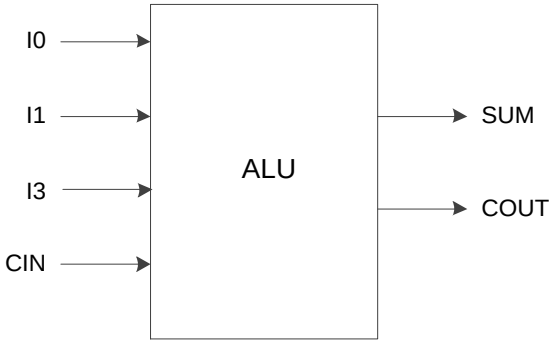
表 2-22 ALU 功能

项目	描述
ADD	加法运算
SUB	减法运算

项目	描述
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

端口示意图

图 2-10 ALU 端口示意图



端口介绍

表 2-23 ALU 端口介绍

端口	Input/Output	描述
I0	Input	数据输入
I1	Input	数据输入
I3	Input	数据输入
CIN	Input	进位输入
COUT	Output	进位输出
SUM	Output	数据输出

参数介绍

表 2-24 ALU 参数介绍

参数	范围	默认	描述

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。

与 FF 相关的原语有 20 个，如表 2-25 所示。

表 2-25 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
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 2-26 FF 类型

编号	类型 1	类型 2
1	DFFS	DFFR

编号	类型 1	类型 2
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。

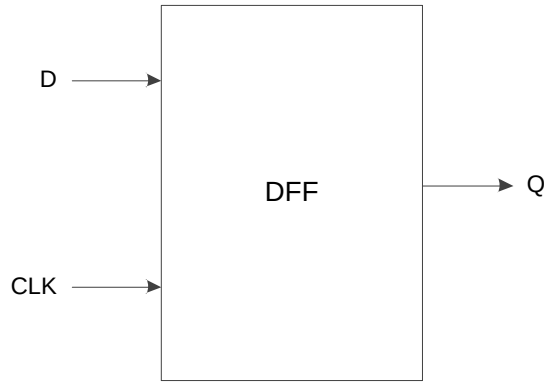
2.4.1 DFF

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。

端口示意图

图 2-11 DFF 端口示意图



端口介绍

表 2-27 DFF 端口介绍

端口	I/O	描述
D	Input	数据输入

CLK	Input	时钟输入
Q	Output	数据输出

参数介绍

表 2-28 DFF 参数介绍

参数	范围	默认	描述
INIT	1’b0,1’b1	1’b0	DFF 初始值

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
       );
```

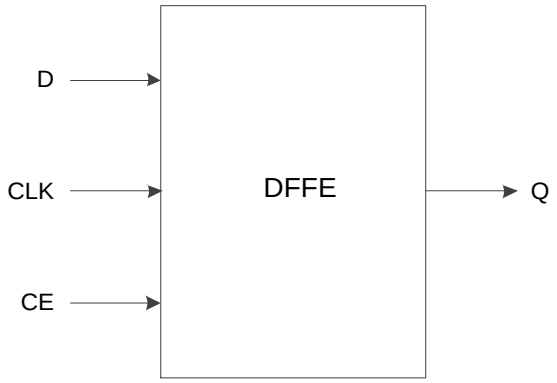
2.4.2 DFFE

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。

端口示意图

图 2-12 DFFE 端口示意图



端口介绍

表 2-29 DFFE 端口介绍

端口	I/O	描述
D	Input	数据输入
CLK	Input	时钟输入
CE	Input	时钟使能
Q	Output	数据输出

参数介绍

表 2-30 DFFE 参数介绍

参数	范围	默认	描述
INIT	1’b0,1’b1	1’b0	DFFE 初始值

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
     );
```

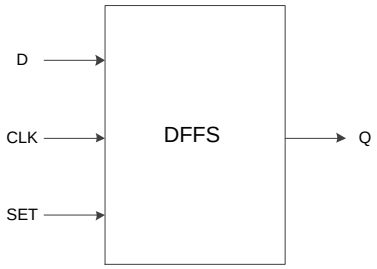
2.4.3 DFFS

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。

端口示意图

图 2-13 DFFS 端口示意图



端口介绍

表 2-31 DFFS 端口介绍

端口	I/O	描述
D	Input	数据输入
CLK	Input	时钟输入
SET	Input	同步置位
Q	Output	数据输出

参数介绍

表 2-32 DFFS 参数介绍

参数	范围	默认	描述
INIT	1'b0,1'b1	1'b1	DFFS 初始值

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
     );
```

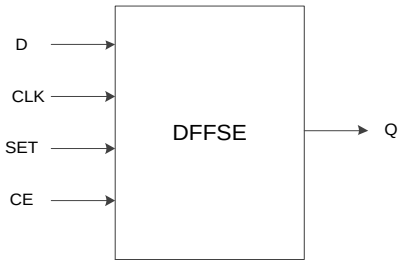
2.4.4 DFFSE

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。

端口示意图

图 2-14 DFFSE 端口示意图



端口介绍

表 2-33 DFFSE 端口介绍

端口	I/O	描述
D	Input	数据输入
CLK	Input	时钟输入
SET	Input	同步置位
CE	Input	时钟使能
Q	Output	数据输出

参数介绍

表 2-34 DFFSE 参数介绍

参数	范围	默认	描述
INIT	1’b0,1’b1	1’b1	DFFSE 初始值

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DFFSE
    GENERIC (INIT:bit:=’1’);
    PORT(
```

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

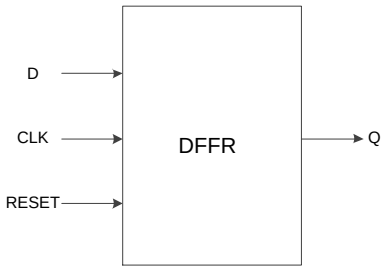
2.4.5 DFFR

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。

端口示意图

图 2-15 DFFR 端口示意图



端口介绍

表 2-35 DFFR 端口介绍

端口	I/O	描述
D	Input	数据输入
CLK	Input	时钟输入
RESET	Input	同步复位
Q	Output	数据输出

参数介绍

表 2-36 DFFR 参数介绍

参数	范围	默认	描述
INIT	1'b0,1'b1	1'b0	DFFR 初始值

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
       );
```

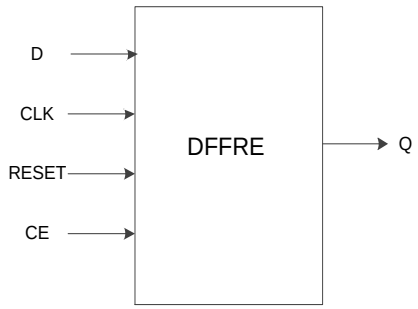
2.4.6 DFFRE

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。

端口示意图

图 2-16 DFFRE 端口示意图



端口介绍

表 2-37 DFFRE 端口介绍

端口	I/O	描述
D	Input	数据输入
CLK	Input	时钟输入
RESET	Input	同步复位
CE	Input	时钟使能
Q	Output	数据输出

参数介绍

表 2-38 DFFRE 参数介绍

参数	范围	默认	描述
INIT	1’b0,1’b1	1’b0	DFFRE 初始值

原语例化

Verilog 例化：

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

Vhdl 例化：

```
COMPONENT DFFRE
        GENERIC (INIT:bit:=’0’);
```

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

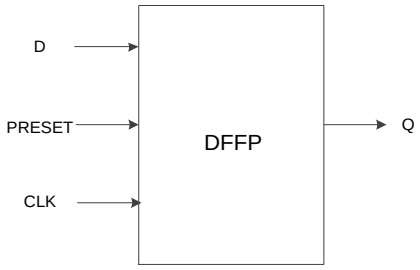
2.4.7 DFFP

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。

端口示意图

图 2-17 DFFP 端口示意图



端口介绍

表 2-39 DFFP 端口介绍

端口	I/O	描述
D	Input	数据输入
CLK	Input	时钟输入
PRESET	Input	异步 Preset 输入
Q	Output	数据输出

参数介绍

表 2-40 DFFP 参数介绍

参数	范围	默认	描述
INIT	1'b0,1'b1	1'b1	DFFP 初始值

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
    );
```

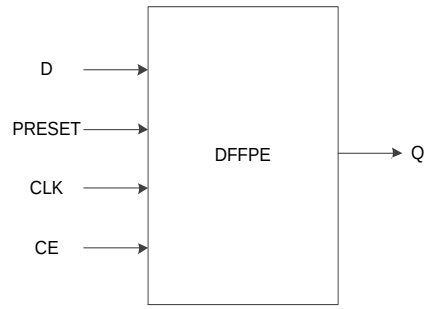
2.4.8 DFFPE

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。

端口示意图

图 2-18 DFFPE 端口示意图



端口介绍

表 2-41 DFFPE 端口介绍

端口	I/O	描述
D	Input	数据输入
CLK	Input	时钟输入
PRESET	Input	异步 Preset 输入
CE	Input	时钟使能
Q	Output	数据输出

参数介绍

表 2-42 DFFPE 参数介绍

参数	范围	默认	描述
INIT	1'b0,1'b1	1'b1	DFFPE 初始值

原语例化

Verilog 例化：

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

Vhdl 例化：

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

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

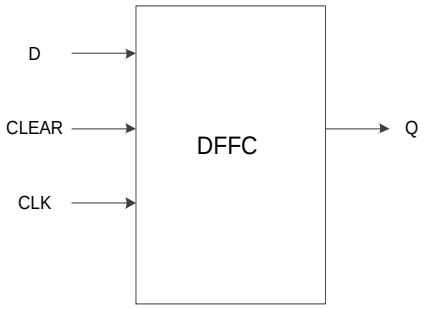
2.4.9 DFFC

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。

端口示意图

图 2-19 DFFC 端口示意图



端口介绍

表 2-43 DFFC 端口介绍

端口	I/O	描述
D	Input	数据输入
CLK	Input	时钟输入
CLEAR	Input	异步清零

端口	I/O	描述
Q	Output	数据输出

参数介绍

表 2-44 DFFC 参数介绍

参数	范围	默认	描述
INIT	1’b0,1’b1	1’b0	DFFC 初始值

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
       );
```

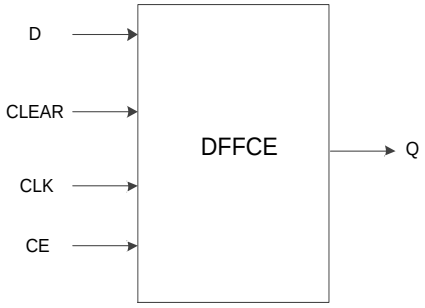
2.4.10 DFFCE

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。

端口示意图

图 2-20 DFFCE 端口示意图



端口介绍

表 2-45 DFFCE 端口介绍

端口	I/O	描述
D	Input	数据输入
CLK	Input	时钟输入
CLEAR	Input	异步清零
CE	Input	时钟使能
Q	Output	数据输出

参数介绍

表 2-46 DFFCE 参数介绍

参数	范围	默认	描述
INIT	1'b0,1'b1	1'b0	DFFCE 初始值

原语例化

Verilog 例化：

```
DFFCE instName (
    .D(D),
    .CLK(CLK),
    .CLEAR(CLEAR),
    .CE(CE),
    .Q(Q)
```

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

Vhdl 例化：

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

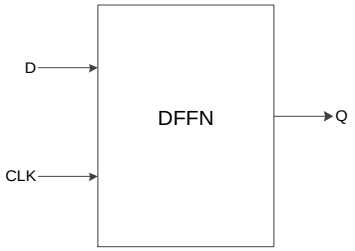
2.4.11 DFFN

原语介绍

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

端口示意图

图 2-21 DFFN 端口示意图



端口介绍

表 2-47 DFFN 端口介绍

端口	I/O	描述
D	Input	数据输入
CLK	Input	时钟输入
Q	Output	数据输出

参数介绍

表 2-48 DFFN 参数介绍

参数	范围	默认	描述
INIT	1'b0,1'b1	1'b0	DFFN 初始值

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
       );
```

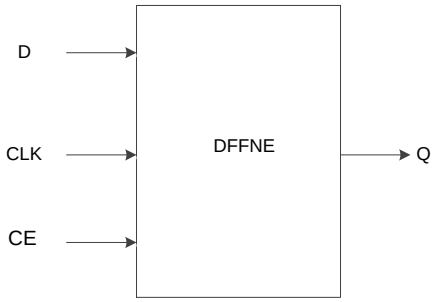
2.4.12 DFFNE

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。

端口示意图

图 2-22 DFFNE 端口示意图



端口介绍

表 2-49 DFFNE 端口介绍

端口	I/O	描述
D	Input	数据输入
CLK	Input	时钟输入
CE	Input	时钟使能
Q	Output	数据输出

参数介绍

表 2-50 DFFNE 参数介绍

参数	范围	默认	描述
INIT	1’b0,1’b1	1’b0	DFFNE 初始值

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
       );
```

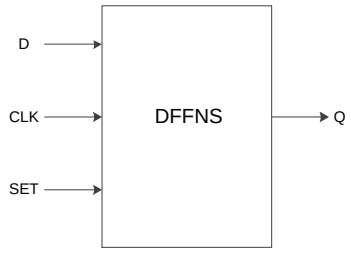
2.4.13 DFFNS

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。

端口示意图

图 2-23 DFFNS 端口示意图



端口介绍

表 2-51 DFFNS 端口介绍

端口	I/O	描述
D	Input	数据输入
CLK	Input	时钟输入
SET	Input	同步置位

端口	I/O	描述
Q	Output	数据输出

参数介绍

表 2-52 DFFNS 参数介绍

参数	范围	默认	描述
INIT	1'b0,1'b1	1'b1	DFFNS 初始值

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
    );
```

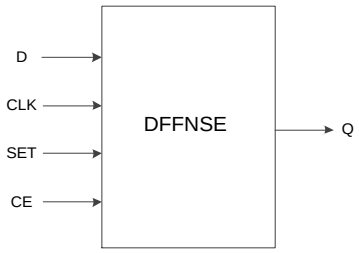
2.4.14 DFFNSE

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。

端口示意图

图 2-24 DFFNSE 端口示意图



端口介绍

表 2-53 DFFNSE 端口介绍

端口	I/O	描述
D	Input	数据输入
CLK	Input	时钟输入
SET	Input	同步置位
CE	Input	时钟使能
Q	Output	数据输出

参数介绍

表 2-54 DFFNSE 参数介绍

参数	范围	默认	描述
INIT	1’b0,1’b1	1’b1	DFFNSE 初始值

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
       );
```

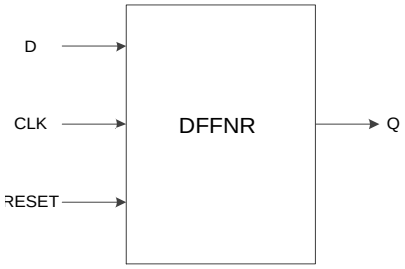
2.4.15 DFFNR

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。

端口示意图

图 2-25 DFFNR 端口示意图



端口介绍

表 2-55 DFFNR 端口介绍

端口	I/O	描述
D	Input	数据输入

端口	I/O	描述
CLK	Input	时钟输入
RESET	Input	同步复位
Q	Output	数据输出

参数介绍

表 2-56 DFFNR 参数介绍

参数	范围	默认	描述
INIT	1'b0,1'b1	1'b0	DFFNR 初始值

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
    );
```

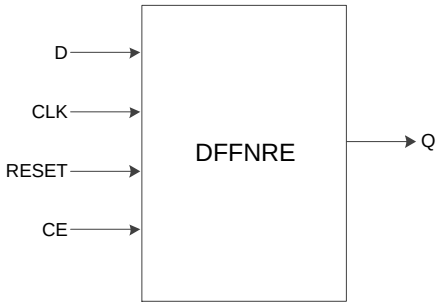
2.4.16 DFFNRE

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。

端口示意图

图 2-26 DFFNRE 端口示意图



端口介绍

表 2-57 DFFNRE 端口介绍

端口	I/O	描述
D	Input	数据输入
CLK	Input	时钟输入
RESET	Input	同步复位
CE	Input	时钟使能
Q	Output	数据输出

参数介绍

表 2-58 DFFNRE 参数介绍

参数	范围	默认	描述
INIT	1’b0,1’b1	1’b0	DFFNRE 初始值

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
       );
```

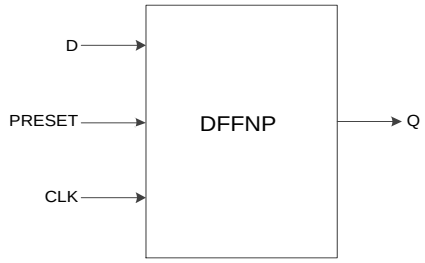
2.4.17 DFFNP

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。

端口示意图

图 2-27 DFFNP 端口示意图



端口介绍

表 2-59 DFFNP 端口介绍

端口	I/O	描述
D	Input	数据输入
CLK	Input	时钟输入
PRESET	Input	异步置位
Q	Output	数据输出

参数介绍

表 2-60 DFFNP 参数介绍

参数	范围	默认	描述
INIT	1'b0,1'b1	1'b1	DFFNP 初始值

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
    );
```

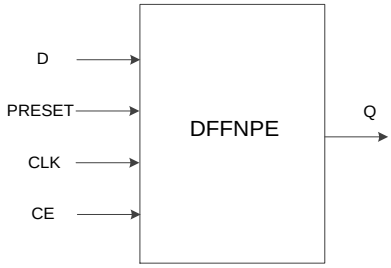
2.4.18 DFFNPE

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。

端口示意图

图 2-28 DFFNPE 端口示意图



端口介绍

表 2-61 DFFNPE 端口介绍

端口	I/O	描述
D	Input	数据输入
CLK	Input	时钟输入
PRESET	Input	异步置位
CE	Input	时钟使能
Q	Output	数据输出

参数介绍

表 2-62 DFFNPE 参数介绍

参数	范围	默认	描述
INIT	1’b0,1’b1	1’b1	DFFNPE 初始值

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
       );
```

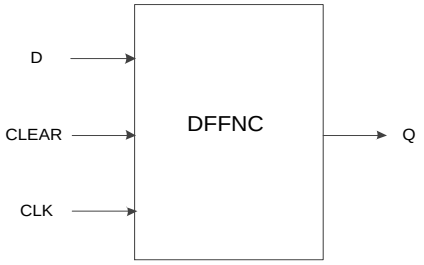
2.4.19 DFFNC

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。

端口示意图

图 2-29 DFFNC 端口示意图



端口介绍

表 2-63 DFFNC 端口介绍

端口	I/O	描述
D	Input	数据输入
CLK	Input	时钟输入
CLEAR	Input	异步复位
Q	Output	数据输出

参数介绍

表 2-64 DFFNC 参数介绍

参数	范围	默认	描述
INIT	1’b0,1’b1	1’b0	DFFNC 初始值

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
    );
```

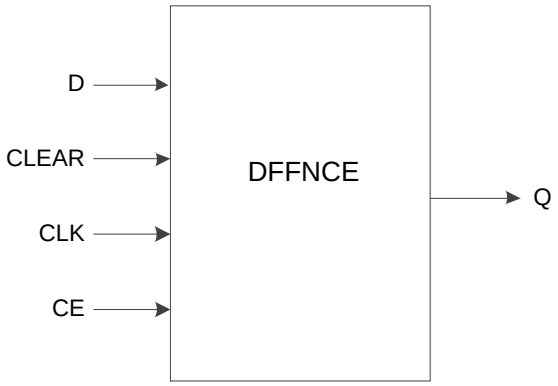
2.4.20 DFFNCE

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。

端口示意图

图 2-30 DFFNCE 端口示意图



端口介绍

表 2-65 DFFNCE 端口介绍

端口	I/O	描述
D	Input	数据输入
CLK	Input	时钟输入
CLEAR	Input	异步复位
CE	Input	时钟使能
Q	Output	数据输出

参数介绍

表 2-66 DFFNCE 参数介绍

参数	范围	默认	描述
INIT	1’b0,1’b1	1’b0	DFFNCE 初始值

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。

与 LATCH 相关的原语有 12 个，如表 2-67 所示。

表 2-67 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

原语	描述
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 2-68 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。

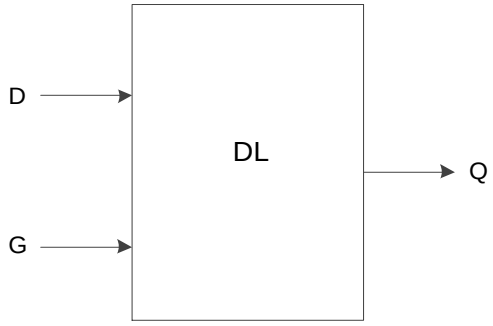
2.5.1 DL

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。

端口示意图

图 2-31 DL 端口示意图



端口介绍

表 2-69 DL 端口介绍

端口	I/O	描述
D	Input	数据输入
G	Input	控制信号
Q	Output	数据输出

参数介绍

表 2-70 DL 参数介绍

参数	范围	默认	描述
INIT	1’b0,1’b1	1’b0	DL 初始值

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
    );
```

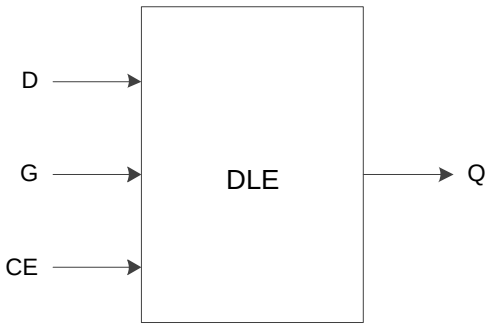
2.5.2 DLE

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。

端口示意图

图 2-32 DLE 端口示意图



端口介绍

表 2-71 DLE 端口介绍

端口	I/O	描述
D	Input	数据输入
G	Input	控制信号
CE	Input	时钟使能
Q	Output	数据输出

参数介绍

表 2-72 DLE 参数介绍

参数	范围	默认	描述
INIT	1'b0,1'b1	1'b0	DLE 初始值

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
       );
```

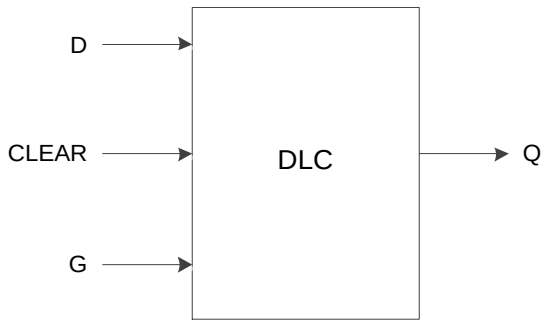
2.5.3 DLC

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。

端口示意图

图 2-33 DLC 端口示意图



端口介绍

表 2-73 DLC 端口介绍

端口	I/O	描述
D	Input	数据输入
CLEAR	Input	异步复位
G	Input	控制信号
Q	Output	数据输出

参数介绍

表 2-74 DLC 参数介绍

参数	范围	默认	描述
INIT	1’b0,1’b1	1’b0	DLC 初始值

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
    );
```

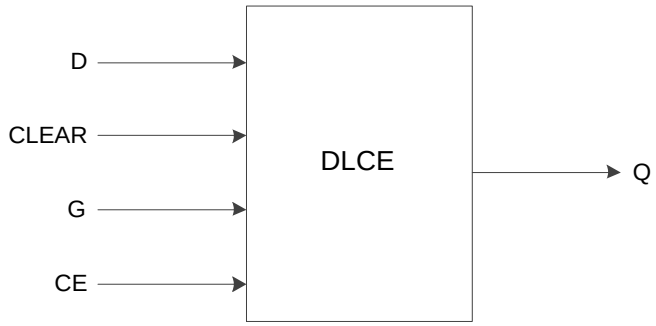
2.5.4 DLCE

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。

端口示意图

图 2-34 DLCE 端口示意图



端口介绍

表 2-75 DLCE 端口介绍

端口	I/O	描述
D	Input	数据输入
CLEAR	Input	异步复位
G	Input	控制信号
CE	Input	时钟使能
Q	Output	数据输出

参数介绍

表 2-76 DLCE 参数介绍

参数	范围	默认	描述
INIT	1’b0,1’b1	1’b0	DLCE 初始值

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
      );
```

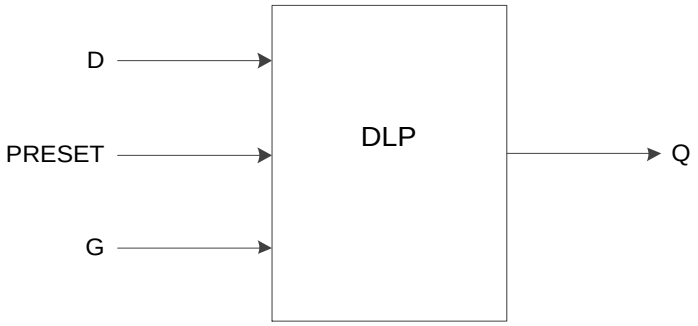
2.5.5 DLP

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。

端口示意图

图 2-35 DLP 端口示意图



端口介绍

表 2-77 DLP 端口介绍

端口	I/O	描述
D	Input	数据输入
PRESET	Input	异步置位
G	Input	控制信号
Q	Output	数据输出

参数介绍

表 2-78 DLP 参数介绍

参数	范围	默认	描述
INIT	1’b0,1’b1	1’b1	DLP 初始值

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
       );
```

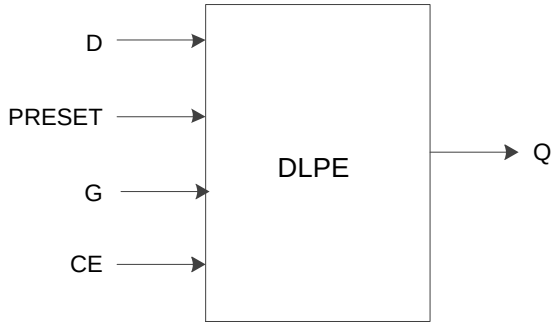
2.5.6 DLPE

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。

端口示意图

图 2-36 DLPE 端口示意图



端口介绍

表 2-79 DLPE 端口介绍

端口	I/O	描述
D	Input	数据输出
PRESET	Input	异步置位

端口	I/O	描述
G	Input	控制信号
CE	Input	时钟使能
Q	Output	数据输出

参数介绍

表 2-80 DLPE 参数介绍

参数	范围	默认	描述
INIT	1'b0,1'b1	1'b1	DLPE 初始值

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
    );
```

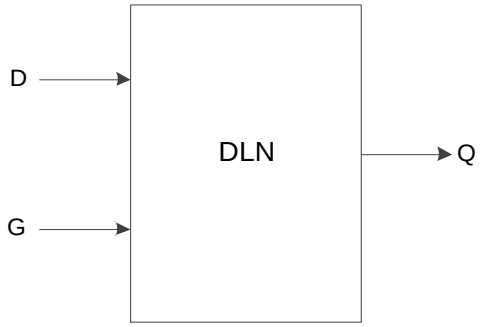
2.5.7 DLN

原语介绍

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

端口示意图

图 2-37 DLN 端口示意图



端口介绍

表 2-81 DLN 端口介绍

端口	I/O	描述
D	Input	数据输入
G	Input	控制信号
Q	Output	数据输出

参数介绍

表 2-82 DLN 参数介绍

参数	范围	默认	描述
INIT	1’b0,1’b1	1’b0	DLN 初始值

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
      );
```

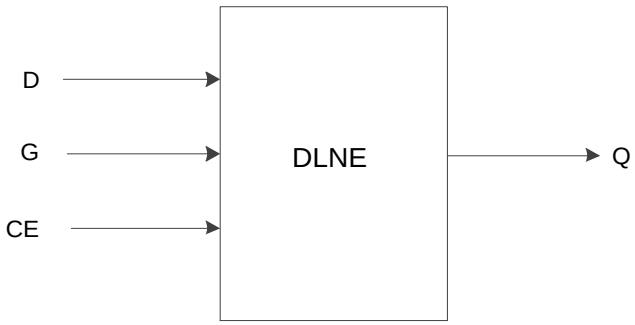
2.5.8 DLNE

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。

端口示意图

图 2-38 DLNE 端口示意图



端口介绍

表 2-83 DLNE 端口介绍

端口	I/O	描述
D	Input	数据输入
G	Input	控制信号
CE	Input	时钟使能
Q	Output	数据输出

参数介绍

表 2-84 DLNE 参数介绍

参数	范围	默认	描述
INIT	1'b0,1'b1	1'b0	DLNE 初始值

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
       );
```

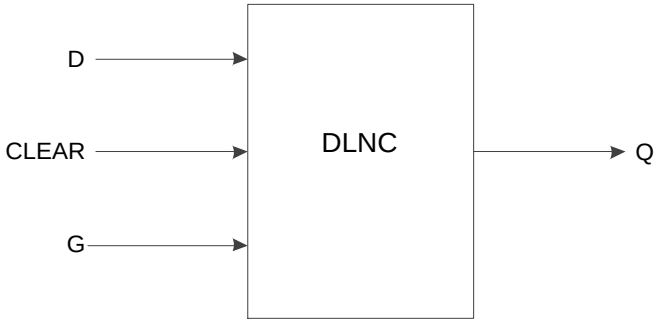
2.5.9 DLNC

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。

端口示意图

图 2-39 DLNC 端口示意图



端口介绍

表 2-85 DLNC 端口介绍

端口	I/O	描述
D	Input	数据输入
CLEAR	Input	异步复位
G	Input	控制信号
Q	Output	数据输出

参数介绍

表 2-86 DLNC 参数介绍

参数	范围	默认	描述
INIT	1’b0,1’b1	1’b0	DLNC 初始值

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
        );
```

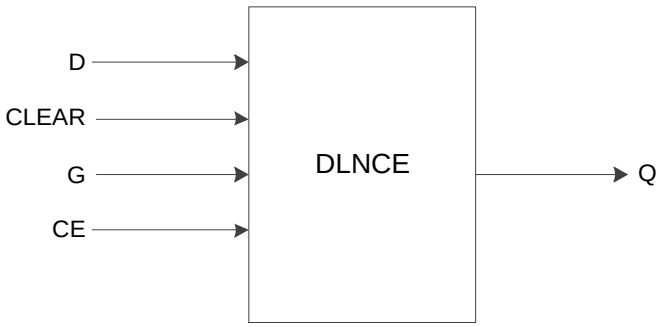
2.5.10 DLNCE

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。

端口示意图

图 2-40 DLNCE 端口示意图



端口介绍

表 2-87 DLNCE 端口介绍

端口	I/O	描述
D	Input	数据输入
CLEAR	Input	异步复位
G	Input	控制信号
CE	Input	时钟使能
Q	Output	数据输出

参数介绍

表 2-88 DLNCE 参数介绍

参数	范围	默认	描述
INIT	1'b0,1'b1	1'b0	DLNCE 初始值

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
       );
```

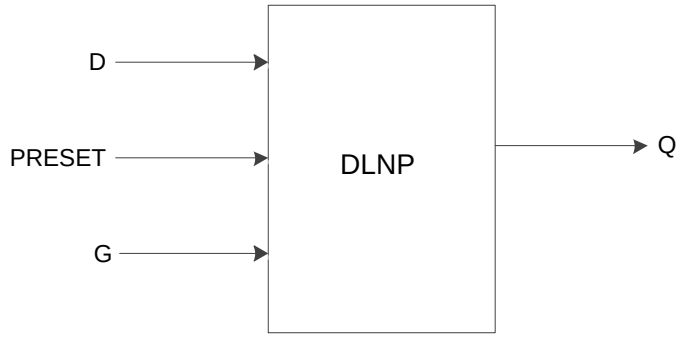
2.5.11 DLNP

原语介绍

DLNP(Data Latch with Asynchronous Preset 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。

端口示意图

图 2-41 DLNP 端口示意图



端口介绍

表 2-89 DLNP 端口介绍

端口	I/O	描述
D	Input	数据输入
PRESET	Input	异步置位
G	Input	控制信号
Q	Output	数据输出

参数介绍

表 2-90 DLNP 参数介绍

参数	范围	默认	描述
INIT	1'b0,1'b1	1'b1	DLNPE 初始值

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
      );
```

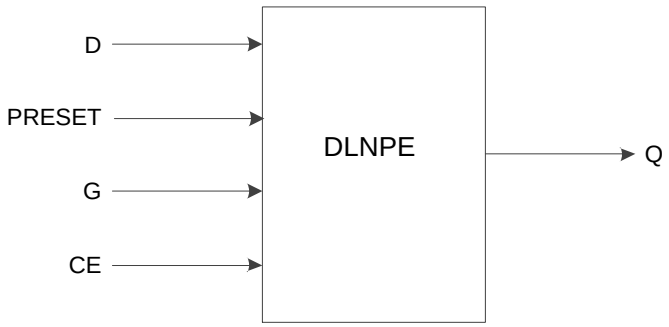
2.5.12 DLNPE

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。

端口示意图

图 2-42 DLNPE 端口示意图



端口介绍

表 2-91 DLNPE 端口介绍

端口	I/O	描述
D	Input	数据输入

端口	I/O	描述
PRESET	Input	异步置位
G	Input	控制信号
CE	Input	时钟使能
Q	Output	数据输出

参数介绍

表 2-92 DLNPE 参数介绍

参数	范围	默认	描述
INIT	1'b0,1'b1	1'b1	DLNPE 初始值

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 Memory

高云半导体 FPGA 产品提供了丰富的存储器资源，包括块状静态随机存储器（BSRAM）和附加静态随机存储器（SSRAM）。BSRAM/SSRAM 原语可参考 UG285，Gowin 存储器(BSRAM & SSRAM)用户指南。

4 DSP

高云半导体 FPGA 产品具有丰富的 DSP 资源，可满足用户对高性能数字信号的处理需求。DSP 原语可参考 UG287，Gowin 数字信号处理器(DSP)用户指南。

5 Clock

高云半导体 FPGA 产品提供了专用全局时钟网络(GCLK，包括 PCLK 和 SCLK)，直接连接到器件的所有资源。除了 GCLK 资源，还提供了锁相环（PLL）、高速时钟 HCLK 和 DDR 存储器接口数据脉冲时钟 DQS 等时钟资源。CLOCK 原语可参考 UG286，Gowin 时钟资源(Clock)用户指南。

6 User Flash

Gowin 小蜜蜂®家族 FPGA 产品提供用户闪存资源（User Flash）。不同系列器件支持不同容量大小的 Flash，包括 FLASH96K、FLASH64K、FLASH64KZ、FLASH128K、FLASH256K 和 FLASH608K。FLASH 原语可参考 UG295,Gowin 闪存资源(User Flash)用户指南。

7 EMPU

7.1 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。

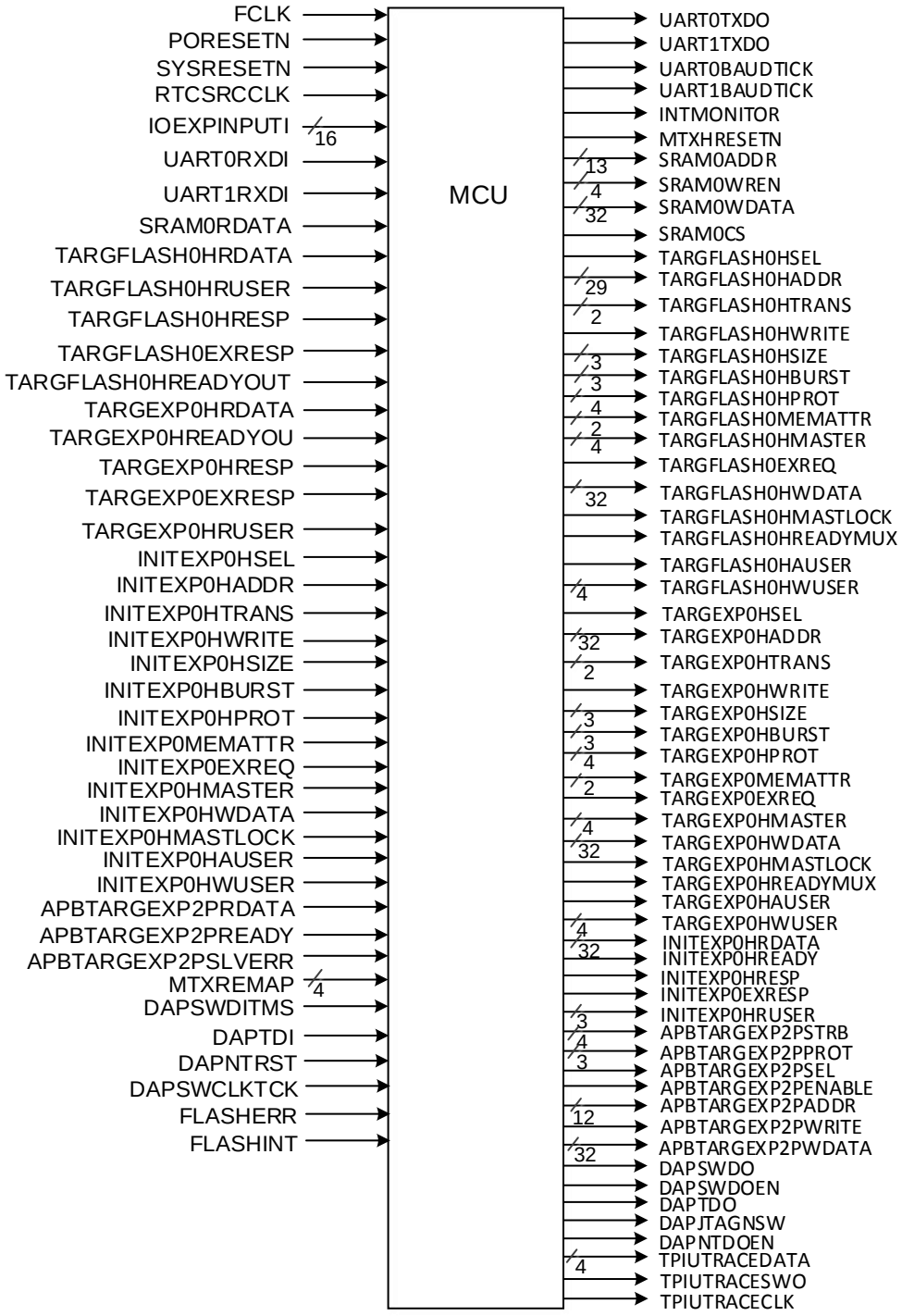
适用器件

表 7-1 MCU 适用器件

家族	系列	器件
GW1N	GW1NS	GW1NS-2C
	GW1NSE	GW1NSE-2C
	GW1NSR	GW1NSR-2C

端口示意图

图 7-1 MCU 端口示意图



端口介绍

表 7-2 MCU 端口介绍

端口	I/O	描述
FCLK	input	Free running clock

端口	I/O	描述
PORESETN	input	Power on reset
SYSRESETN	input	System reset
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

端口	I/O	描述
UART1TXDO	output	UART1TXDO
UART0BAUDTICK	output	UART0BAUDTICK
UART1BAUDTICK	output	UART1BAUDTICK
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

端口	I/O	描述
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
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 );
```

7.2 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、6 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。EMCU 增强了中断处理能力，改善了 FLASH 接口，提高了 MCU 运行主频。

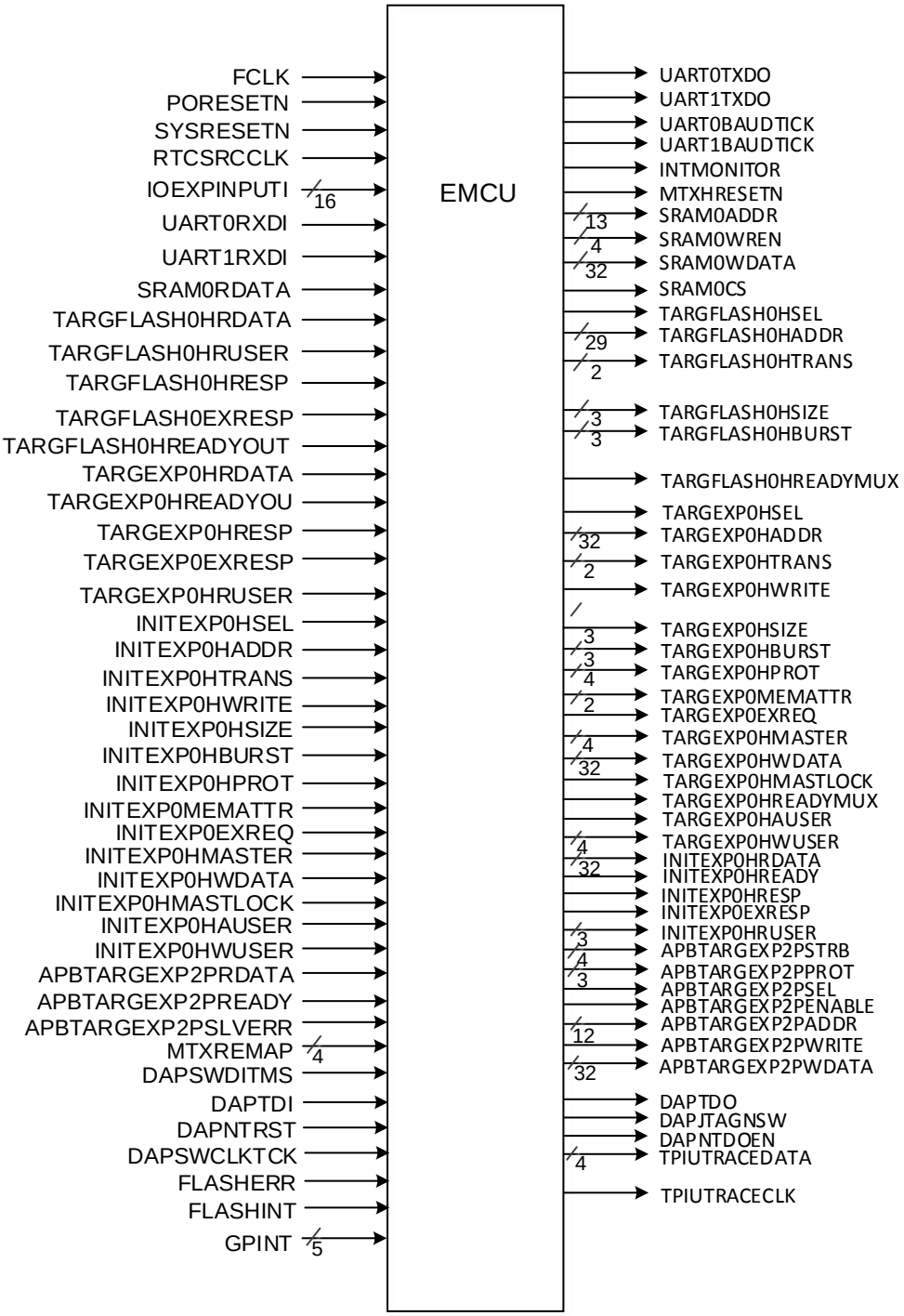
适用器件

表 7-3 EMCU 适用器件

家族	系列	器件
GW1N	GW1NS	GW1NS-4C
	GW1NSR	GW1NSR-4C
	GW1NSER	GW1NSER-4C

端口示意图

图 7-2 EMCU 端口示意图



端口介绍

表 7-4 EMCU 端口介绍

端口	I/O	描述
FCLK	input	Free running clock
PORESETN	input	Power on reset
SYSRESETN	input	System reset

端口	I/O	描述
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
GPINT	input	GPINT
IOEXPOUTPUTO[15:0]	output	IOEXPOUTPUTO
IOEXPOUTPUTENO[15:0]	output	IOEXPOUTPUTENO
UART0TXDO	output	UART0TXDO
UART1TXDO	output	UART1TXDO

端口	I/O	描述
UART0BAUDTICK	output	UART0BAUDTICK
UART1BAUDTICK	output	UART1BAUDTICK
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

端口	I/O	描述
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 );
```

7.3 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。

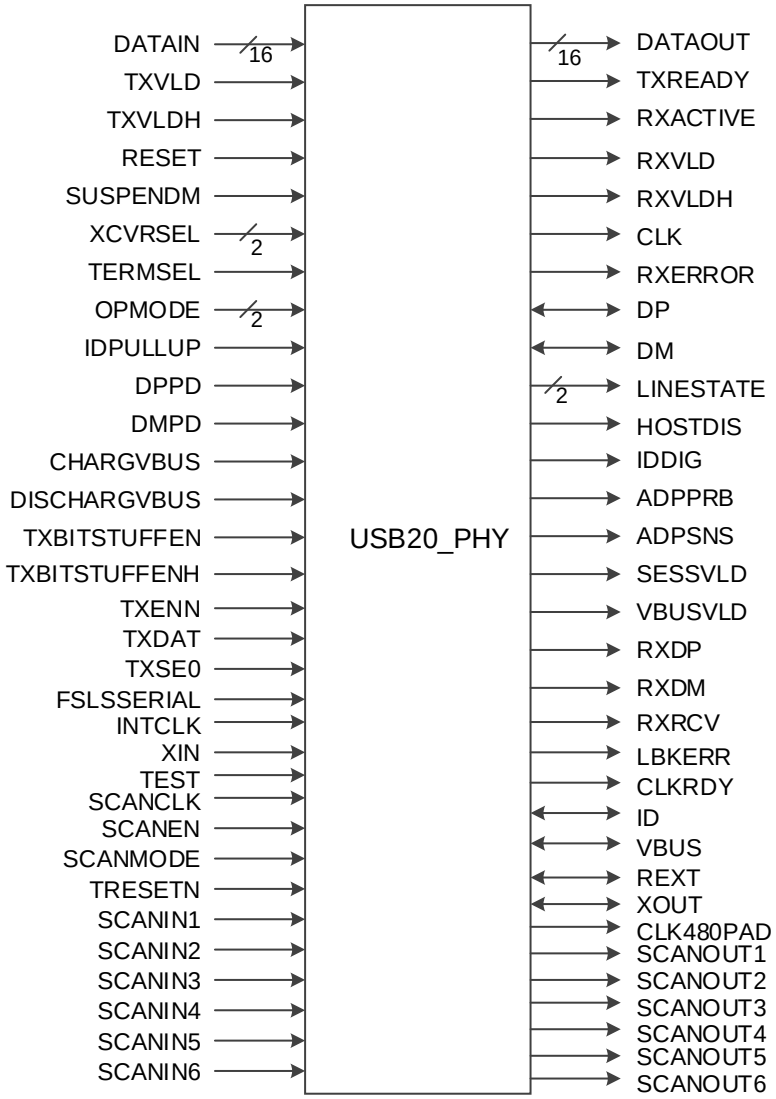
适用器件

表 7-5 USB20_PHY 适用器件

家族	系列	器件
GW1N	GW1NS	GW1NS-2, GW1NS-2C
	GW1NSE	GW1NSE-2C
	GW1NSR	GW1NSR-2, GW1NSR-2C

端口示意图

图 7-3 USB20_PHY 端口示意图



端口介绍

表 7-6 USB20_PHY 端口介绍

端口	I/O	描述
DATAIN[15:0]	input	16-bit parallel USB data input bus
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

端口	I/O	描述
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.
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.

端口	I/O	描述
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

参数介绍

表 7-7 USB20_PHY 参数介绍

参数	默认	描述
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
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
    );
```

7.4 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。

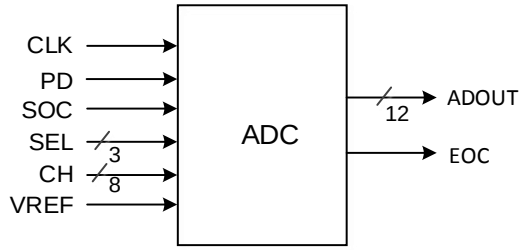
适用器件

表 7-8 ADC 适用器件

家族	系列	器件
GW1N	GW1NS	GW1NS-2, GW1NS-2C
	GW1NSE	GW1NSE-2C
	GW1NSR	GW1NSR-2, GW1NSR-2C

端口示意图

图 7-4 ADC 端口示意图



端口介绍

表 7-9 ADC 端口介绍

端口	I/O	描述
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
        );
```

IP 调用

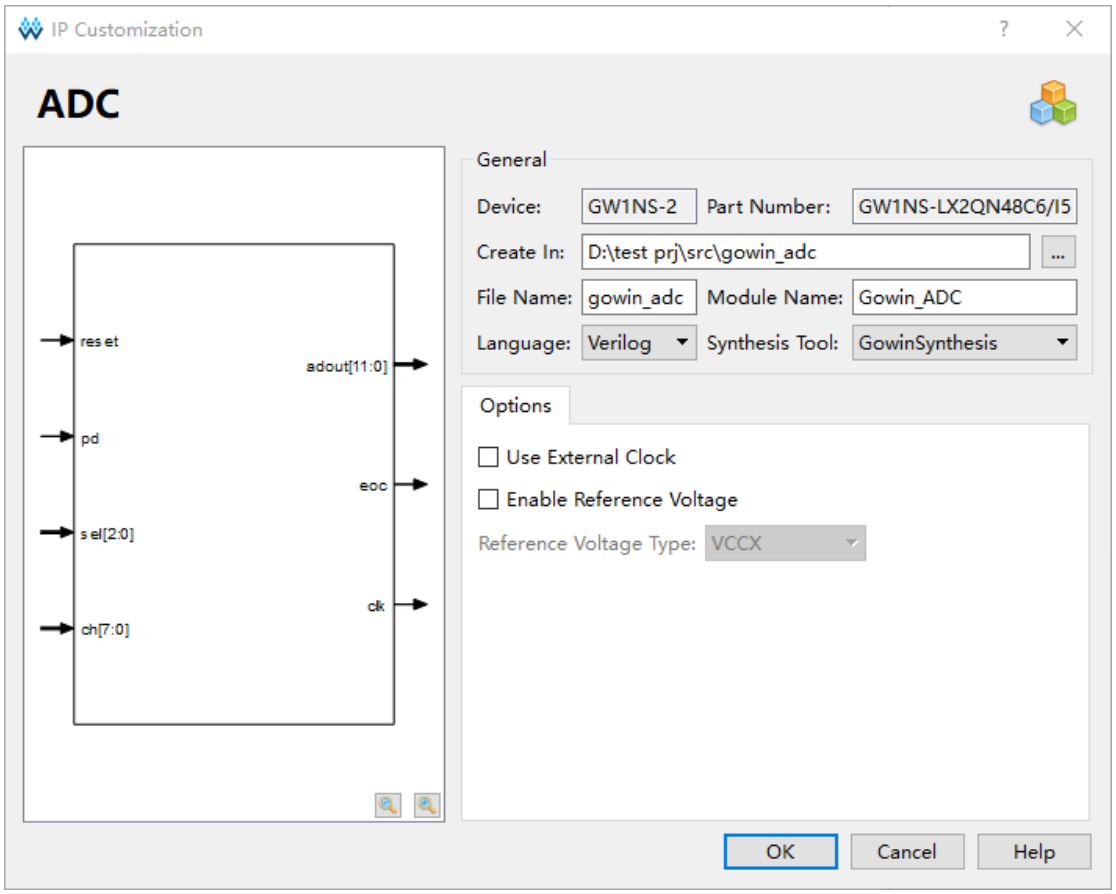
在 IP Core Generator 界面中单击“ADC”，界面右侧会显示 ADC 的相关信息概要。

IP 配置

在 IP Core Generator 界面中，双击“ADC”，弹出 ADC 的“IP Customization”窗口，该窗口包括“File”配置框、端口显示框图和“Help”

按钮，如图 7-5 所示。

图 7-5 ADC 的 IP Customization 窗口结构



1. File 配置框

File 配置框用于配置产生的 IP 设计文件的相关信息。

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

2. Options 配置框

Options 配置框用于用户自定义配置 IP，Options 配置框如图 7-5 所示。

- Use External Clock：配置 ADC 是否使用外部时钟；
- Enable Reference Voltage：配置 ADC 是否使能参考电压。若使能，则通过选项 Reference Voltage Type 选择参考电压类型；若不使能，则默认参考电压为 VCCX。
- Reference Voltage Type：配置参考电压的类型，包括 VCCX、34/40(*VCCX)、31/40(*VCCX)、29/40(*VCCX)、27/40(*VCCX)、22/40(*VCCX)、20/40(*VCCX)和 External。

3. 端口显示框图

端口显示框图显示 IP Core 的配置结果示例框图，如图 7-5 所示。

4. Help 按钮

单击“Help”，显示 IP Core 的配置信息的页面。Help 页面包括 IP Core 的概要介绍，以及 Options 各项配置的简要说明。

IP 生成文件

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

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

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。

8 其它

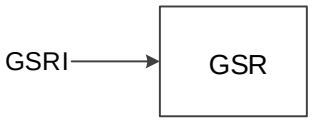
8.1 GSR

原语介绍

GSR(Global Set/Reset)， 全局置位/复位模块，可以实现全局置位/复位功能，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。

端口示意图

图 8-1 GSR 端口示意图



端口介绍

表 8-1 GSR 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
            );
```

8.2 INV

原语介绍

INV(Inverter)，取反模块。

端口示意图

图 8-2 INV 端口示意图

端口介绍

表 8-2 INV 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:OUT std_logic;
                I:IN std_logic

        );
END COMPONENT;
uut:INV
```

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

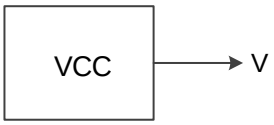
8.3 VCC

原语介绍

逻辑高电平发生器。

端口示意图

图 8-3 VCC 端口示意图



端口介绍

表 8-3 VCC 端口介绍

端口名	I/O	描述
V	Output	VCC 输出

原语例化

Verilog 例化：

```
VCC uut (
      .V(V)
);
```

Vhdl 例化：

```
COMPONENT VCC
      PORT (
            V:OUT std_logic
      );
END COMPONENT;
uut:VCC
        PORT MAP(
          V => V
        );
```

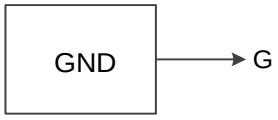
8.4 GND

原语介绍

逻辑低电平发生器。

端口示意图

图 8-4 GND 端口示意图



端口介绍

表 8-4 GND 端口介绍

端口名	I/O	描述
G	Output	GND 输出

原语例化

Verilog 例化：

```
GND uut (
    .G(G)
);
```

Vhdl 例化：

```
COMPONENT GND
    PORT (
        G:OUT std_logic
    );
END COMPONENT;
uut:GND
      PORT MAP(
        G => G
);
```

8.5 BANDGAP

原语介绍

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。

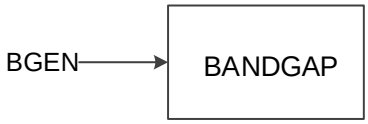
适用器件

表 8-5 BANDGAP 适用器件

家族	系列	器件
GW1N	GW1NZ	GW1NZ-1

端口示意图

图 8-5 BANDGAP 端口示意图



端口介绍

表 8-6 BANDGAP 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
        );
```

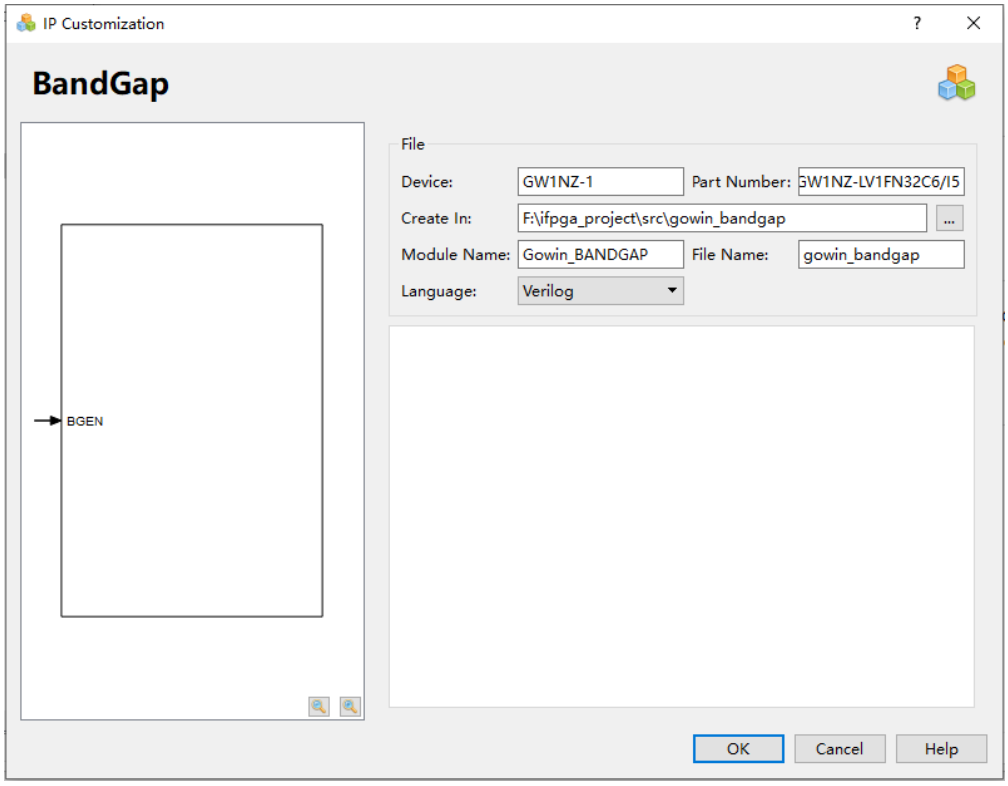
IP 调用

在 IP Core Generator 界面中，单击“BandGap”，界面右侧会显示 BandGap 的相关信息概要。

IP 配置

在 IP Core Generator 界面中，双击 BandGap，弹出 BandGap 的“IP Customization”窗口。该窗口包括“File”配置框、“Options”配置框、端口显示框图和“Help”按钮，如图 8-6 所示。

图 8-6 BandGap 的 IP Customization 窗口结构



1. File 配置框

   - File 配置框用于配置产生的 IP 设计文件的相关信息。
   - BandGap 的 File 配置框的使用和 ADC 模块类似，具体请参考 7.4 ADC > IP 调用部分的 File 配置框。

2. 端口显示框图

   端口显示框图显示当前 IP Core 的配置结果示例框图，如图 8-6 所示。

3. Help 按钮

   单击“Help”，显示 IP Core 的配置信息的页面。Help 页面包括当前 IP Core 的概要介绍。

IP 生成文件

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

- IP 设计文件“gowin_bandgap.v”为完整的 verilog 模块，根据用户的 IP 配置，产生实例化的 BandGap；
- IP 设计使用模板文件 gowin_bandgap_tmp.v，为用户提供 IP 设计使用模板文件；
- IP 配置文件：“gowin_bandgap.ipc”，用户可加载该文件对 IP 进行配置。

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。

8.6 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。

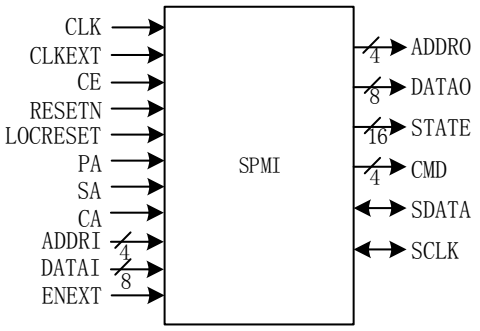
适用器件

表 8-7 SPMI 适用器件

家族	系列	器件
GW1N	GW1NZ	GW1NZ-1

端口示意图

图 8-7 SPMI 端口示意图



端口介绍

表 8-8 SPMI 端口介绍

端口	I/O	描述
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
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
        );
```

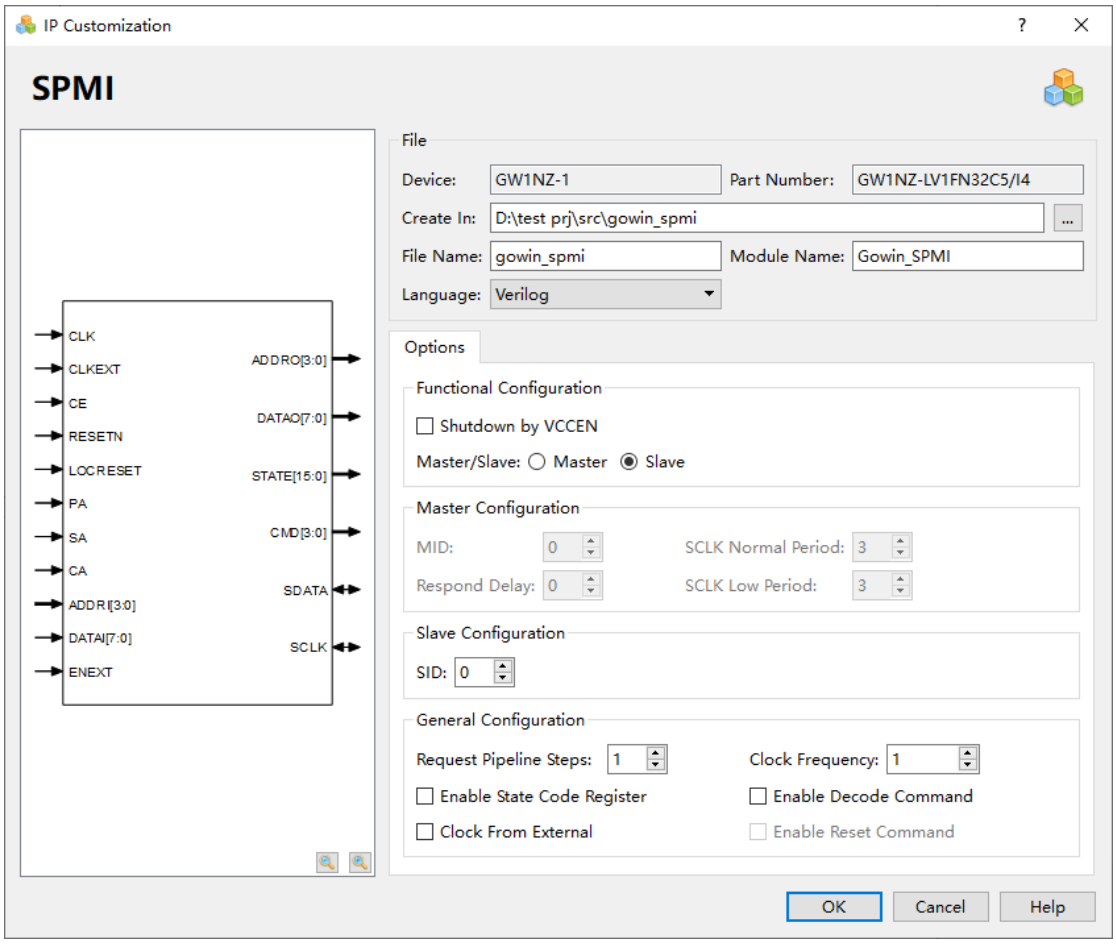
IP 调用

在 IP Core Generator 界面中，单击“SPMI”，界面右侧会显示 SPMI 的相关信息概要。

IP 配置

在 IP Core Generator 界面中，双击 SPMI，弹出 SPMI 的“IP Customization”窗口。该窗口包括“File”配置框、“Options”配置框、端口显示框图和“Help”按钮，如图 8-8 所示。

图 8-8 SPMI 的 IP Customization 窗口结构



1. File 配置框
   - File 配置框用于配置产生的 IP 设计文件的相关信息。
   - SPMI 的 File 配置框的使用和 ADC 模块类似，具体请参考 7.4 ADC > IP 调用部分的 File 配置框。
2. Options 配置框
   - Options 配置框用于用户自定义配置 IP，Options 配置框如图 8-8 所示。
   - Functional Configuration：
     - Shutdown by VCCEN： 通过外部引脚 VCCEN 关闭。如果选择此选项，则 SPMI 的通信功能将不可用。
     - Master/Slave：将 SPMI 设置为主机或从机。
   - Master Configuration：
     - MID：主机的 ID，设置范围为 0-3，默认值为 0。
     - Respond Delay：设置响应延迟时间。

- SCLK Normal Period：Normal 模式下 sclk 的周期长度。
- SCLK Low Period：睡眠模式下 sclk 的周期长度。

- Slave Configuration：

  SID：设置 SPMI 从机的 ID。

- General configuration：
  - Enable State Code Register：启用或禁用寄存器。例如，如果选择“启用状态代码寄存器”选项，则输出 STATE 数据将通过一个寄存器。
  - Request Pipeline Steps：设置请求信号采样时间的延迟步长。
  - Enable Decode Command：启用或禁用解码。如果选择启用解码命令，SPMI 将解码复位，睡眠，关闭和唤醒命令。
  - Enable Reset Command：启用或禁用重置命令。
  - Clock From External：启用或禁用外部时钟。
  - Clock Frequency：系统时钟频率。

3. 端口显示框

   端口显示框图显示当前 IP Core 的配置结果示例框图，如图 8-8 所示。

4. Help 按钮

   单击“Help”，显示 IP Core 的配置信息的页面。Help 页面包括当前 IP Core 的概要介绍，以及 Options 各项配置的简要说明。

IP 生成文件

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

- IP 设计文件“gowin_spmi.v”为完整的 verilog 模块，根据用户的 IP 配置，产生实例化的 SPMI；
- IP 设计使用模板文件 gowin_spmi_tmp.v，为用户提供 IP 设计使用模板文件；
- IP 配置文件：“gowin_spmi.ipc”，用户可加载该文件对 IP 进行配置。

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。

8.7 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。

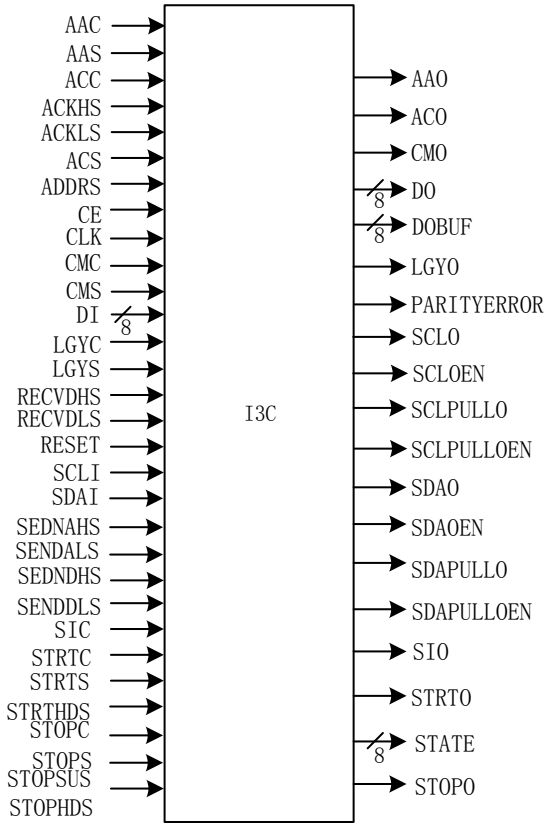
适用器件

表 8-9 I3C 适用器件

家族	系列	器件
GW1N	GW1NZ	GW1NZ-1

端口示意图

图 8-9 I3C 端口示意图



端口介绍

表 8-10 I3C 端口介绍

端口	I/O	描述
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

端口	I/O	描述
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
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)
    );
```

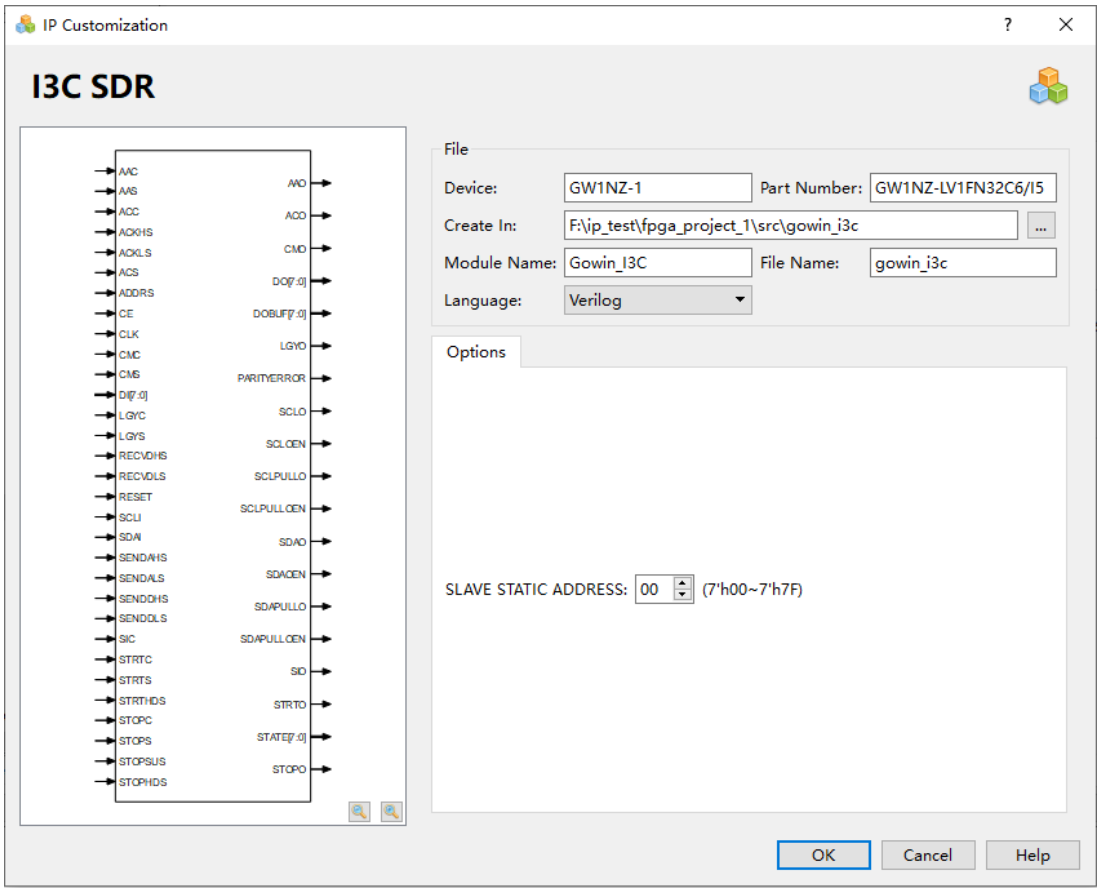
IP 调用

在 IP Core Generator 界面中单击 I3C 下的 I3C SDR，界面右侧会显示 I3C SDR 的相关信息概要。

IP 配置

在 IP Core Generator 界面中，双击“I3C SDR”，弹出 I3C 的“IP Customization”窗口，该窗口包括“File”配置框、“Options”配置框、端口显示框图以及“Help”按钮，如图 8-10 所示。

图 8-10 I3C 的 IP Customization 窗口结构



1. File 配置框

   - File 配置框用于配置产生的 IP 设计文件的相关信息。
   - I3C 的 File 配置框的使用和 ADC 模块的类似，请参考 7.4 ADC > IP 调用部分的 File 配置框。

2. Options 配置框

   - Options 配置框用于用户自定义配置 IP，Options 配置框如图 8-10 所示。
   - SLAVE STATIC ADDRESS - 指定从机的静态地址。

3. 端口显示框图

   端口显示框图显示 IP Core 的配置结果示例框图，如图 8-10 所示。

4. Help 按钮

   单击“Help”，显示 IP Core 的配置信息的页面。Help 页面包括 IP Core 的概要介绍，以及 Options 各项配置的简要说明。

IP 生成文件

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

- IP 设计文件“gowin_i3c.v”为完整的 verilog 模块，根据用户的 IP 配置，产生实例化的 I3C；
- IP 设计使用模板文件 gowin_i3c_tmp.v，为用户提供 IP 设计使用模板文件；
- IP 配置文件：“gowin_i3c.ipc”，用户可加载该文件对 IP 进行配置。

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。


GOWIN高云
智慧逻辑 定制未来