# Gowin Primitives

# User Guide

**SUG283-2.4E, 09/11/2020**

**Disclaimer**

## Revision History

| Date | Version | Description |
|---|---|---|
| 04/20/2017 | 1.0E | Initial version published. |
| 09/19/2017 | 1.1E | GW1NR-4, GW1N-6, GW1N-9, GW1NR-9 devices added;<br>ELVDS_IOBUF, TLVDS_IOBUF, BUFG, BUFS, OSC, IEM added;<br>DSP primitive updated;<br>Some ports of ODDR/ODDRC, IDDR_MEM, IDES4_MEM, IDES8_MEM, RAM16S1, RAM16S2、RAM16S4, RAM16SDP1, RAM16SDP2, RAM16SDP4, ROM16 updated;<br>Some Attribute of OSC, PLL and DLLDLY updated;<br>Some primitive instantiation updated;<br>MIPI_IBUF_HS, MIPI_IBUF_LP, MIPI_OBUF, IDES16 and OSER16 updated;<br>Some Attribute of CLKDIV updated. |
| 04/12/2018 | 1.2E | Vhdl primitives instantiation added. |
| 08/08/2018 | 1.3E | GW1N-2B, GW1N-4B, GW1NR-4B, GW1N-6ES, GW1N-9ES, GW1NR-9ES, GW1NS-2,GW1NS-2C devices added;<br>I3C_IOBUF, DHCEN added;<br>User Flash added;<br>EMPU added;<br>Primitive name updated. |
| 10/26/2018 | 1.4E | GW1NZ-1, GW1NSR-2C devices added;<br>OSCZ, FLASH96KZ added. |
| 11/15/2018 | 1.5E | GW1NSR-2 device added;<br>GW1N-6ES, GW1N-9ES, GW1NR-9ES devices removed. |
| 01/26/2019 | 1.6E | GW1NS-2 supported by 8 frequency division of CLKDIV added;<br>Removed GW1N-1 from the devices supported by TLVDS_TBUF/OBUF. |
| 02/25/2019 | 1.7E | Removed GW1N-1 from the devices supported by TLVDS_IOBUF. |
| 05/20/2019 | 1.8E | GW1N-1S device added;<br>MIPI_IBUF added;<br>OSCH added;<br>SPMI added;<br>I3C added;<br>Devices supported by OSC updated. |
| 10/20/2019 | 1.9E | IOB, BSRAM, CLOCK modules updated. |
| 11/28/2019 | 2.0E | GSR and INV modules added in Miscellaneous;<br>Devices supported updated;<br>FLASH64KZ added and FLASH96KZ removed. |
| 01/16/2020 | 2.1E | IODELAYA, rPLL, PLLVR, CLKDIV2 added;<br>DPB/DPX9B, SDPB/SDPX9B, rSDP/rSDPX9, rROM/rROMX9, pROM/pROMX9 added;<br>EMCU, BANDGAP, FLASH64K added;<br>IODELAY, PLL, CLKDIV, OSC, DQCE updated;<br>Placement rule of FF、LATCH added;<br>GW2A-55C added;<br>GW1N-6/GW1N-9/GW1NR-9 disabled DP/DPX9, DPB/DPX9B;<br>Notes of register added in IOLOGIC;<br>GW1NZ-1 disabled 1, 2, 4, 8 bit width of DP/DPB and 9 bit width of DPX9/DPX9. |

| Date | Version | Description |
| --- | --- | --- |
| 03/09/2020 | 2.2E | GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C disable DP/DPX9 and DPB/DPX9B;<br>OSCEN port description added in OSCF;<br>PLL/rPLL/PLLVR parameter description updated. |
| 06/08/2020 | 2.3E | GW1N-2, GW1N-2B and GW1N-6 removed;<br>GW1N-9C and GW1NR-9C added;<br>IODELAYC, DHCENC and DCC added;<br>Functional description of MIPI_IBUF added;<br>MIPI_IBUF_HS, MIPI_IBUF_LP and DLL removed;<br>Port diagram of LUT5 and MUX8 added;<br>VCC and GND added;<br>PLLVR, FLASH64K, BUFS, EMPU and CLKDIV2 updated;<br>DP/DPX9, ROM/ROMX9, SDP/SDPX9, rSDP/rSDPX9, rROM/rROMX9 and PLL removed. |
| 09/11/2020 | 2.4E | The description of IP invoking of ADC, BANDGAP, SPMI and I3C modules added. |

# Contents

# List of Figures

# List of Tables

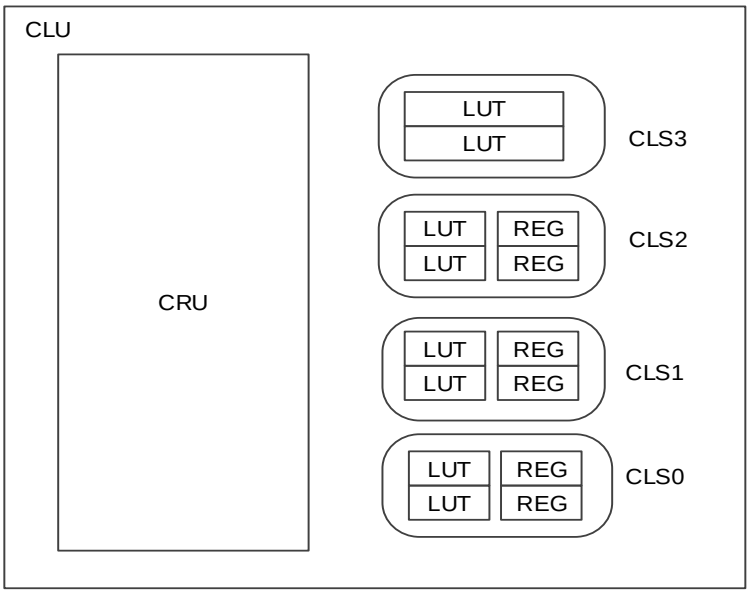
# 1 IOB

IOB includes input/output buffer (IO Buffer) and input/output logic (IO Logic). For IO Buffer and IO Logic primitives, see UG289, Gowin Programmable IO(GPIO) User Guide.

# 2 CLU

Configurable Logic Unit (CLU) is a basic block for FPGA products. One CFU includes four Configurable Logic Slices ( CLS ) and one Configurable Routing Unit ( CRU ), as shown in Figure 2-1. CLS can be configured as LUT, ALU and REG. CFU can implement the functions of MUX/LUT/ALU/FF/LATCH.

**Figure 2-1 CLU Port Diagram**




## 2.1 LUT

LUT includes LUT1, LUT2, LUT3 and LUT4, the differences between them are the bit width.

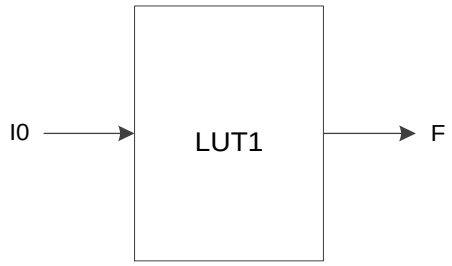
### 2.1.1 LUT1

**Primitive**

LUT1 is usually used as a buffer and an inverter. LUT1 is an 1-input lookup table. After initializing, you can look up the corresponding data according to the input address, then it outputs the data.

## Port Diagram

**Figure 2-2 LUT1 Port Diagram**




## Port Description

**Table 2-1 Port Description**

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

## Parameter

**Table 2-2 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 2'h0~2'h3 | 2'h0 | Initial value of LUT1 |

## Truth Table

**Table 2-3 Truth Table**

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

## Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
        );
```

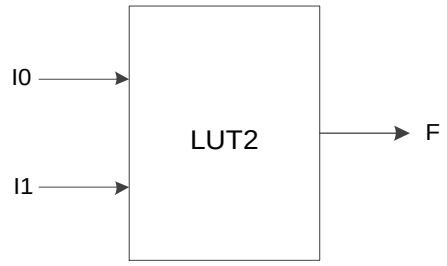
## 2.1.2 LUT2

### Primitive

LUT2 is a 2-input lookup table. After initializing, you can look up the corresponding data according to the input address, then it outputs the data.

### Port Diagram

**Figure 2-3 LUT2 Port Diagram**




### Port Description

**Table 2-4 Port Description**

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

### Parameter

**Table 2-5 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 4'h0~4'hf | 4'h0 | Initial value of LUT2 |

### Truth Table

**Table 2-6 Truth Table**

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

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
 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
     );
```

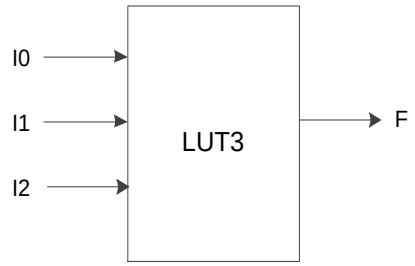
## 2.1.3 LUT3

### Primitive

LUT3 is a 3-input lookup table. After initializing, you can look up the corresponding data according to the input address, then it outputs the data.

## Port Diagram

**Figure 2-4 LUT3 Port Diagram**




## Port Description

**Table 2-7 Port Description**

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

## Parameter

**Table 2-8 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 8'h00~8'hff | 8'h00 | Initial value of LUT3 |

## Truth Table

**Table 2-9 Truth Table**

| 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] |

## Primitive Instantiation

### Verilog Instantiation:

LUT3 instName (

```
        .I0(I0),
        .I1(I1),
        .I2(I2),
        .F(F)
);
defparam instName.INIT=8'h10;
```

**Vhdl Instantiation:**

```
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
     );
```

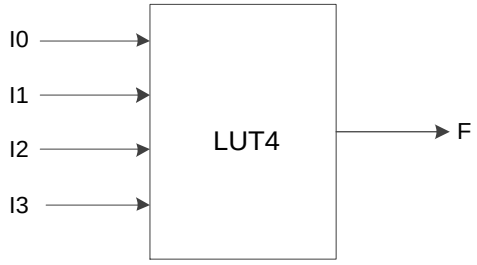
## 2.1.4 LUT4

### Primitive

LUT4 is a 4-input lookup table. After initializing, you can look up the corresponding data according to the input address, then it outputs the data.

### Port Diagram

**Figure 2-5 LUT4 Diagram**



## Port Description

**Table 2-10 Port Description**

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

## Parameter

**Table 2-11 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 16'h0000~16'hffff | 16'h0000 | Initial value of LUT4 |

## Truth Table

**Table 2-12 Truth Table**

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

## Primitive Instantiation

### Verilog Instantiation:

```
LUT4 instName (
    .I0(I0),
```

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

**Vhdl Instantiation:**

```
    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

**Primitive**

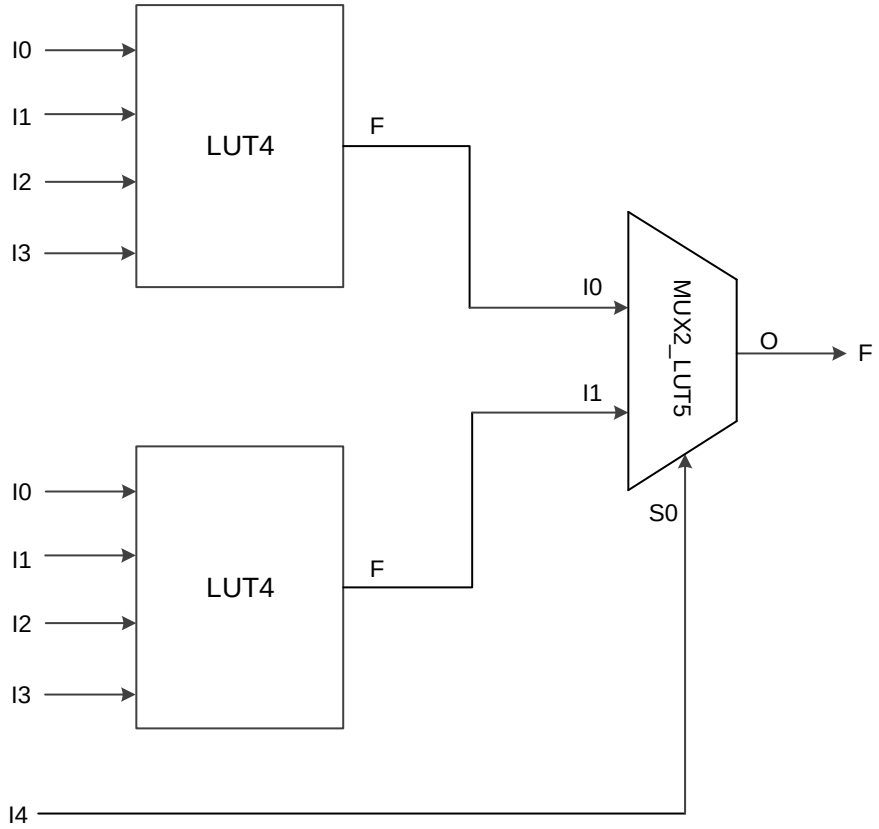
Wide LUT is used for constructing high-order LUT by LUT4 and MUX2. MUX2 series of Gowin FPGA supports MUX2_LUT5/ MUX2_LUT6/ MUX2_LUT7/ MUX2_LUT8.

The way of constructing high-order LUT is as follows: one LUT5 can be implemented by two LUT4s and one MUX2_LUT5; one LUT6 can be implemented by two LUT5s and one MUX2_LUT6; one LUT7 can be implemented by two LUT6s and one MUX2_LUT7; one LUT8 can be implemented by two LUT7s and MUX2_LUT8.

The following takes LUT5 as an example to introduce the use of Wide LUT.

## Port Diagram

**Figure 2-6 MUX2_LUT5 Port Diagram**




## Port Description

**Table 2-13 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| I0 | Input | Data input |
| I1 | Input | Data input |
| I2 | Input | Data input |
| I3 | Input | Data input |
| I4 | Input | Data input |
| F | Output | Data output |

## Parameter

**Table 2-14 Parameter**

| Parameter | Value | Default | Description |
|---|---|---|---|
| INIT | 32'h00000~32'hfffff | 32'h00000 | Initial value of LUT5 |

### Truth Table

**Table 2-15Truth Table**

| 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] |
| 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] |

### Primitive Instantiation

#### Verilog Instantiation:

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

**Vhdl Instantiation:**

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

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

# 2.2 MUX

MUX is a multiplexer. There are multiple inputs. It transmits one input to the output based on the channel-selection signal. Gowin MUX includes 2-to-1 multiplexer and 4-to-1 multiplexer.

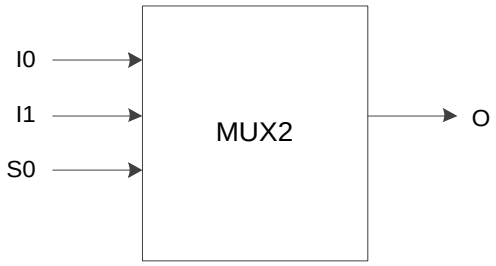
## 2.2.1 MUX2

**Primitive**

2-to-1 Multiplexer ( MUX2 ) selects one of the two inputs as the output based on the selection signal.

## Port Diagram

**Figure 2-7 MUX2 Port Diagram**




## Port Description

**Table 2-16 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| I0 | Input | Data Input |
| I1 | Input | Data Input |
| S0 | Input | Selection Signal |
| O | Output | Data Output |

## Truth Table

**Table 2-17 Truth Table**

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

## Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
    );
```

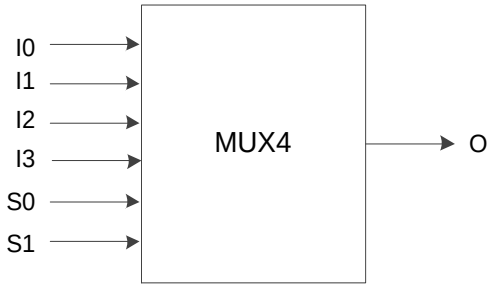
## 2.2.2 MUX4

### Primitive

4-to-1 Multiplexer ( MUX4 ) selects one of the four inputs as the output based on the selection signal.

### Port Diagram

**Figure 2-8 MUX4 Port Diagram**




### Port Description

**Table 2-18 Port Description**

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

### Truth Table

**Table 2-19 Truth Table**

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

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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

### Primitive

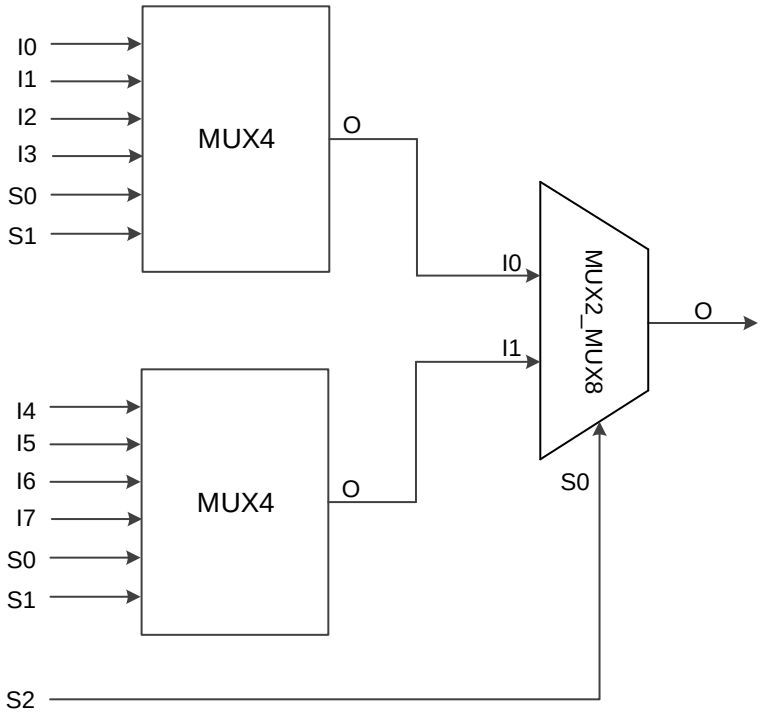
Wide MUX is used for constructing high-order MUX by MUX4 and MUX2. MUX2 series of Gowin FPGA supports MUX2_MUX8/ MUX2_MUX16/ MUX2_MUX32.

The way of constructing high-order MUX is as follows: One MUX8 can be implemented by two MUX4s and one MUX2_MUX8; one MUX16 can be implemented by two MUX8s and one MUX2_MUX16; one MUX32 can be implemented by two MUX16s and one MUX2_MUX32.

The following takes MUX2_MUX8 as an example to introduce the use of Wide MUX.

### Port Diagram

**Figure 2-9 MUX2_MUX8 Port Diagram**




### Port Description

**Table 2-20 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| I0 | Input | Data input |
| I1 | Input | Data input |
| I2 | Input | Data input |

| Port Name | I/O | Description |
|---|---|---|
| I3 | Input | Data input |
| I4 | Input | Data input |
| I5 | Input | Data input |
| I6 | Input | Data input |
| I7 | Input | Data input |
| S0 | Input | Selection signal |
| S1 | Input | Selection signal |
| S2 | Input | Selection signal |
| O | Output | Data output |

### Truth Table

**Table 2-21 Truth Table**

| 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 |

### Primitive Instantiation

**Verilog Instantiation:**

```
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 Instantiation:**

```
    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

### Primitive

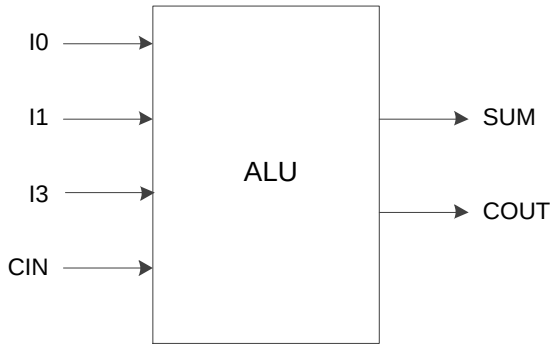
ALU is a 2-input arithmetic logic Unit and it can realize the functions of ADD/SUB/ADDSUB.

**Table 2-22 ALU Functions**

| Item | Description |
|---|---|
| ADD | ADD |
| SUB | SUB |
| ADDSUB | ADDSUB |
| CUP | CUP |
| CDN | CDN |
| CUPCDN | CUPCDN |
| GE | GE |
| NE | NE |
| LE | LE |
| MULT | MULT |

## Port Diagram

**Figure 2-10 ALU Port Diagram**




## Port Description

**Table 2-23 Port Description**

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

## Parameter

**Table 2-24 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| ALU_MODE | 0,1,2,3,4,5,6,7,8,9 | 0 | Select the function of arithmetic.<br>0:ADD;<br>1:SUB;<br>2:ADDSUB;<br>3:NE;<br>4:GE;<br>5:LE;<br>6:CUP;<br>7:CDN;<br>8:CUPCDN;<br>9:MULT |

## Primitive Instantiation

### Verilog Instantiation:

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

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

**Vhdl Instantiation:**

```
    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

Flip-flop is a basic component in the timing circuit. Timing logic in FPGA can be implemented through an FF. The commonly used FF includes DFF, DFFE, DFFS, DFFSE, etc. The differences between them are reset modes, triggering modes, etc.

**Table 2-25 Primitives Associated With FF**

| Primitive | Description |
|---|---|
| DFF | D flip-flop |
| DFFE | D flip-flop with clock enable |
| DFFS | D flip-flop with synchronous set |

| Primitive | Description |
|---|---|
| DFFSE | D flip-flop with clock enable and synchronous set |
| DFFR | D flip-flop with synchronous reset |
| DFFRE | D flip-flop with clock enable and synchronous reset |
| DFFP | D flip-flop with asynchronous preset |
| DFFPE | D flip-flop with clock enable and asynchronous preset |
| DFFC | D flip-flop with asynchronous clear |
| DFFCE | D flip-flop with clock enable and asynchronous clear |
| DFFN | Neg-edge D flip-flop |
| DFFNE | Neg-edge D flip-flop with clock enable |
| DFFNS | Neg-edge D flip-flop with synchronous set |
| DFFNSE | Neg-edge D flip-flop with clock enable and synchronous set |
| DFFNR | Neg-edge D flip-flop with synchronous reset |
| DFFNRE | Neg-edge D flip-flop with clock enable and synchronous reset |
| DFFNP | Neg-edge D flip-flop with asynchronous preset |
| DFFNPE | Neg-edge D flip-flop with clock enable and asynchronous preset |
| DFFNC | Neg-edge D flip-flop with asynchronous clear |
| DFFNCE | Neg-edge D flip-flop with clock enable and asynchronous clear |

## Placement Rule

**Table 2-26 Type of FF**

| No. | Type 1 | Type 2 |
|---|---|---|
| 1 | DFFS | DFFR |
| 2 | DFFSE | DFFRE |
| 3 | DFFP | DFFC |
| 4 | DFFPE | DFFCE |
| 5 | DFFNS | DFFNR |
| 6 | DFFNSE | DFFNRE |
| 7 | DFFNP | DFFNC |
| 8 | DFFNPE | DFFNCE |

1. DFF of the same type can be placed on two FFs in the same CLS. All input other than pin input must be in the same net;

2. DFF of two types but same No. can be palced on two FFs in the same CLS, as shown in Table 2-26. All input other than pin input must be in the same net;

3. DFF and ALU can be constrainted in the same or different locations of the same CLS;

4. DFF and LUT can be constrainted in the same or different locations of the same CLS;

**Note!**

The two nets via inverter can not be placed in the same CLS.

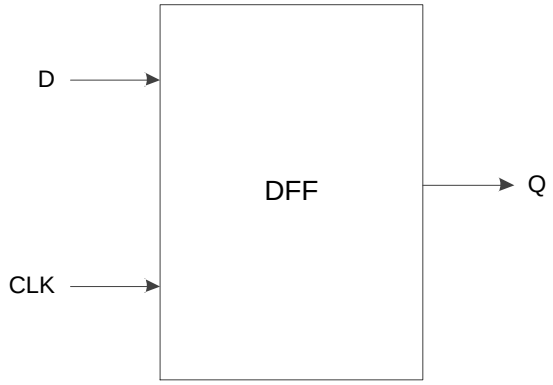
## 2.4.1 DFF

### Primitive

The D Flip-Flop ( DFF ), pos-edge triggered, is commonly used for signal sampling and processing .

### Port Diagram

**Figure 2-11 DFF Port Diagram**




### Port Description

**Table 2-27 Port Description**

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

### Parameter

**Table 2-28 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value for DFF |

### Primitive Instantiation

**Verilog Instantiation:**

```
DFF instName (
        .D(D),
        .CLK(CLK),
        .Q(Q)
```

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

**Vhdl Instantiation:**

```
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
     );
```

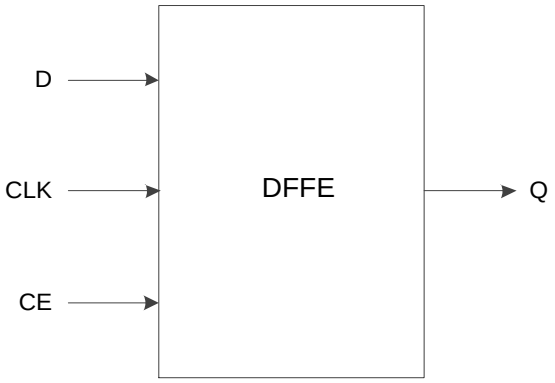
## 2.4.2 DFFE

### Primitive

DFFE, pos-edge triggered, is thw D Flip-Flop with clock enable.

### Port Diagram

**Figure 2-12 DFFE Port Diagram**




### Port Description

**Table 2-29 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| D | Input | Data Input |
| CLK | Input | Clock input |

| Port Name | I/O | Description |
|---|---|---|
| CE | Input | Clock Enable |
| Q | Output | Data Output |

### Parameter

**Table 2-30 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value for DFFE |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
    );
```

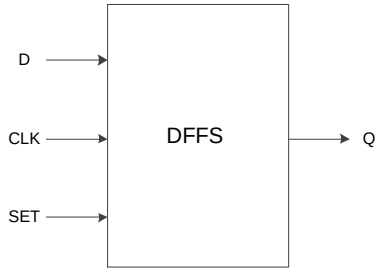
## 2.4.3 DFFS

### Primitive

DFFS, pos-edge triggered, is the D Flip-Flop with synchronous set .

### Port Diagram

**Figure 2-13 DFFS Port Diagram**




### Port Description

**Table 2-31 Port Description**

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

### Parameter

**Table 2-32 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value for DFFS |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
        );
```

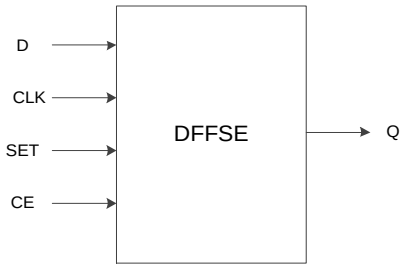
## 2.4.4 DFFSE

### Primitive

DFFSE, pos-edge triggered, is the D Flip-Flop with clock enable and synchronous set..

### Port Diagram

**Figure 2-14 DFFSE Port Diagram**




### Port Description

**Table 2-33 Port Description**

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

### Parameter

**Table 2-34 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value for DFFSE |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

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

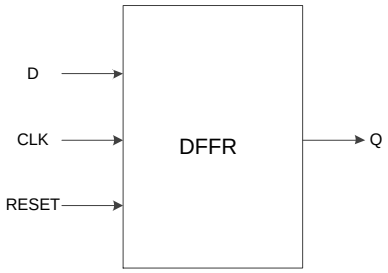
## 2.4.5 DFFR

### Primitive

DFFR, pos-edge triggered, is the D Flip-Flop with synchronous reset.

### Port Diagram

**Figure 2-15 DFFR Port Diagram**




### Port Description

**Table 2-35 Port Description**

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

### Parameter

**Table 2-36 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value for DFFR |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
    );
```

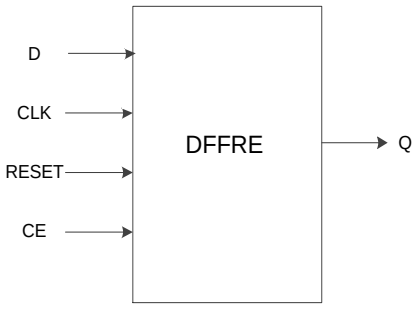
## 2.4.6 DFFRE

### Primitive

DFFRE, pos-edge triggered, is the D Flip-Flop with clock enable and synchronous reset.

### Port Diagram

**Figure 2-16 DFFRE Port Diagram**




### Port Description

**Table 2-37 Port Description**

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

### Parameter

**Table 2-38 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value for DFFRE |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

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

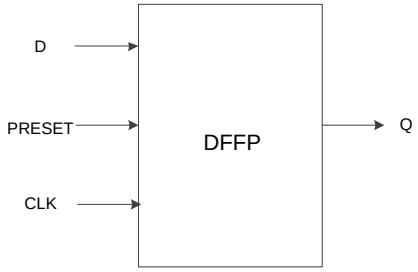
## 2.4.7 DFFP

### Primitive

DFFP, pos-edge triggered, is the D Flip-Flop with asynchronous preset.

### Port Diagram

**Figure 2-17 DFFP Port Diagram**




### Port Description

**Table 2-39 Port Description**

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

### Parameter

**Table 2-40 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value for DFFP |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
    );
```

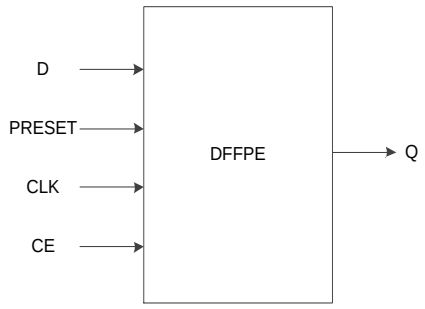
## 2.4.8 DFFPE

### Primitive

DFFPE, pos-edge triggered, is the D Flip-Flop with clock enable and asynchronous preset.

### Port Diagram

**Figure 2-18 DFFPE Port Diagram**




### Port Description

**Table 2-41 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| D | Input | Data Input |
| CLK | Input | Clock input |
| PRESET | Input | Asynchronous Preset Input |
| CE | Input | Clock Enable |

| Port Name | I/O | Description |
|---|---|---|
| Q | Output | Data Output |

### Parameter

**Table 2-42 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value for DFFPE |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

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

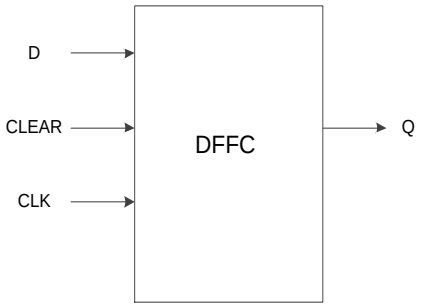
## 2.4.9 DFFC

### Primitive

DFFC, pos-edge triggered, is the D Flip-Flop with asynchronous clear.

### Port Diagram

**Figure 2-19 DFFC Blcok Diagram**




### Port Description

**Table 2-43 Port Description**

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

### Parameter

**Table 2-44 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1’b0,1’b1 | 1’b0 | Initial value for DFFC |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
    );
```

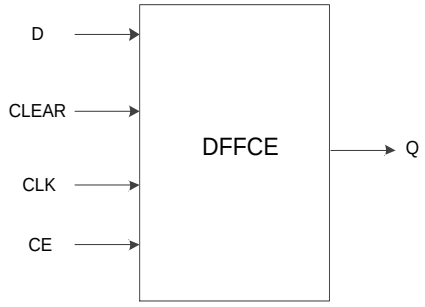
## 2.4.10 DFFCE

### Primitive

DFFCE, pos-edge triggered, is the D Flip-Flop with clock enable and asynchronous clear.

### Port Diagram

**Figure 2-20 DFFCE Port Diagram**




### Port Description

**Table 2-45 Port Description**

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

### Parameter

**Table 2-46 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value for DFFCE |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

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

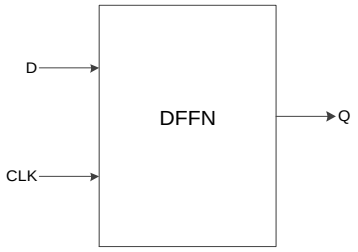
## 2.4.11 DFFN

### Primitive

DFFN is D Flip-Flop with neg-edge clock.

### Port Diagram

**Figure 2-21 DFFN Port Diagram**




### Port Description

**Table 2-47 Port Description**

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

### Parameter

**Table 2-48 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1’b0,1’b1 | 1’b0 | Initial value for DFFN |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
    );
```

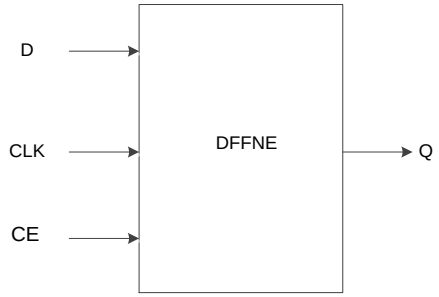
## 2.4.12 DFFNE

### Primitive

DFFNE, neg-edge triggered, is the D Flip-Flop with clock enable.

### Port Diagram

**Figure 2-22 DFFNE Port Diagram**




### Port Description

**Table 2-49 Port Description**

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

### Parameter

**Table 2-50 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value for DFFNE |

**Primitive Instantiation**

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
    );
```

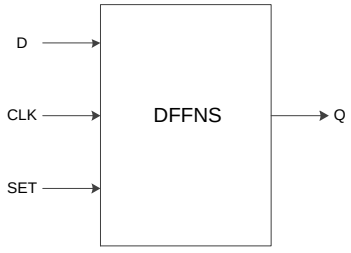
## 2.4.13 DFFNS

**Primitive**

DFFNS, neg-edge triggered, is the D Flip-Flop with synchronous set.

**Port Diagram**

**Figure 2-23 DFFNS Blcok Diagram**



### Port Description

**Table 2-51 Port Description**

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

### Parameter

**Table 2-52 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value for DFFNS |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
    );
```

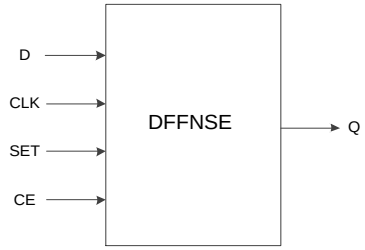
## 2.4.14 DFFNSE

### Primitive

DFFNSE, neg-edge triggered, is the D Flip-Flop with clock enable,and synchronous set.

### Port Diagram

**Figure 2-24 DFFNSE Port Diagram**




### Port Description

**Table 2-53 Port Description**

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

### Parameter

**Table 2-54 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value for DFFNSE |

### Primitive Instantiation

**Verilog Instantiation:**

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

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

**Vhdl Instantiation:**

```
    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
         );
```

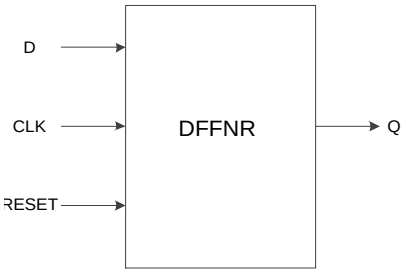
## 2.4.15 DFFNR

### Primitive

DFFNR, neg-edge triggered, is the D Flip-Flop with synchronous reset.

### Port Diagram

**Figure 2-25 DFFNR Port Diagram**



### Port Description

**Table 2-55 Port Description**

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

### Parameter

**Table 2-56 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value for DFFNR |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
    );
```

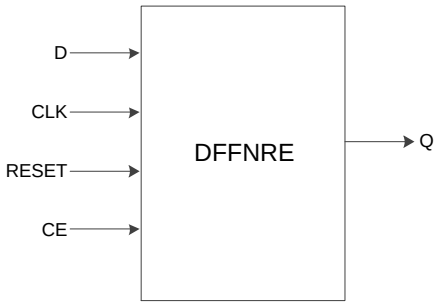
## 2.4.16 DFFNRE

### Primitive

DFFNRE, neg-edge triggered, is the D Flip-Flop with clock enable, and synchronous reset.

### Blcok Diagram

**Figure 2-26 DFFNRE Blcok Diagram**




### Port Description

**Table 2-57 Port Description**

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

### Parameter

**Table 2-58 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value for DFFNRE |

### Primitive Instantiation

**Verilog Instantiation:**

```
DFFNRE instName (
    .D(D),
    .CLK(CLK),
    .RESET(RESET),
    .CE(CE),
```

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

**Vhdl Instantiation:**

```
    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
        );
```

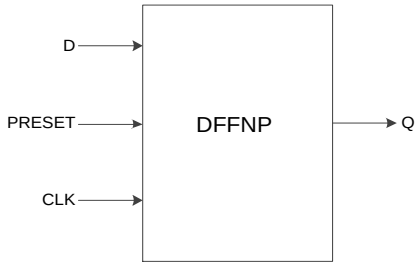
## 2.4.17 DFFNP

### Primitive

DFFNP, neg-edge triggered, is the D Flip-Flop with asynchronous preset.

### Port Diagram

**Figure 2-27 DFFNP Port Diagram**



### Port Description

**Table 2-59 Port Description**

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

### Parameter

**Table 2-60 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value for DFFNP |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
    );
```

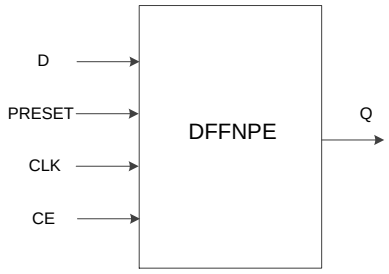
## 2.4.18 DFFNPE

### Primitive

DFFNPE, neg-edge triggered, is the D Flip-Flop with clock enable, and asynchronous preset.

### Port Diagram

**Figure 2-28 DFFNPE Port Diagram**




### Port Description

**Table 2-61 Port Description**

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

### Parameter

**Table 2-62 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value for DFFNPE |

### Primitive Instantiation

**Verilog Instantiation:**

```
DFFNPE instName (
    .D(D),
    .CLK(CLK),
    .PRESET(PRESET),
    .CE(CE),
    .Q(Q)
```

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

**Vhdl Instantiation:**

```
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
    );
```

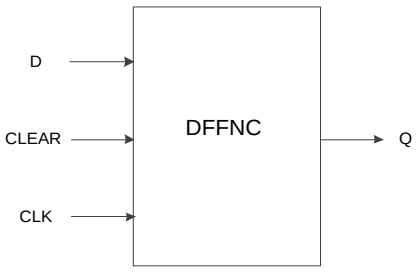
## 2.4.19 DFFNC

### Primitive

DFFNC, neg-edge triggered, is the D Flip-Flop with asynchronous clear.

### Port Diagram

**Figure 2-29 DFFNC Port Diagram**



### Port Description

**Table 2-63 Port Description**

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

### Parameter

**Table 2-64 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1’b0,1’b1 | 1’b0 | Initial value for DFFNC |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
    );
```

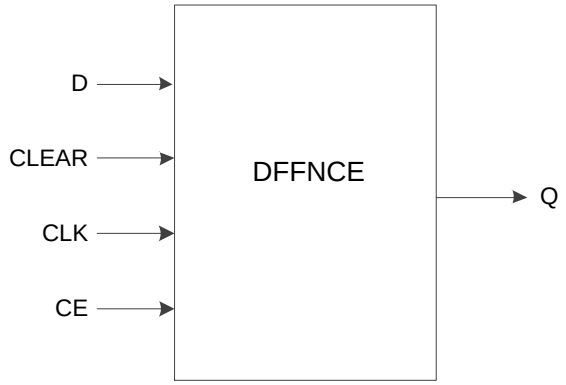
## 2.4.20 DFFNCE

### Primitive

DFFNCE, neg-edge triggered, is the D Flip-Flop with clock enable and asynchronous clear.

### Port Diagram

**Figure 2-30 DFFNCE Port Diagram**




### Port Description

**Table 2-65 Port Description**

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

### Parameter

**Table 2-66 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1’b0,1’b1 | 1’b0 | Initial value for DFFNCE |

### Primitive Instantiation

**Verilog Instantiation:**

```
DFFNCE instName (
    .D(D),
```

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

**Vhdl Instantiation:**

```
    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

LATCH is a memory cell circuit and its status can be changed by specified input level.

**Table 2-67 Primitives Related with LATCH**

| Primitive | Description |
|---|---|
| DL | Data Latch |
| DLE | Data latch with enable |
| DLC | Data latch with asynchronous clearing |
| DLCE | Data latch with enable and asynchronous clearing |
| DLP | Data latch with asynchronous preset |
| DLPE | Data latch with asynchronous preset and enable |
| DLN | Data latch, active-low |

| Primitive | Description |
|---|---|
| DLNE | Data latch with enable, active-low |
| DLNC | Data latch with asynchronous clearing, active-low |
| DLNCE | Data latch with enable and asynchronous clearing, active-low |
| DLNP | Data latch with asynchronous preset, active-low |
| DLNPE | Data latch with asynchronous preset and enable, active-low |

### Placement Rule

**Table 2-68 Type of LATCH**

| No. | Type 1 | Type 2 |
|---|---|---|
| 1 | DLC | DLP |
| 2 | DLCE | DLPE |
| 3 | DLNC | DLNP |
| 4 | DLNCE | DLNPE |

1. DL of the same type can be placed on two FFs in the same CLS. All input other than pin input must be in the same net;

   5. DL of two types but same No. can be palced on two FFs in the same CLS, as shown in Table 2-68. All input other than pin input must be in the same net;

   6. DL and ALU can be constrainted in the same or different locations of the same CLS;

   7. DL and LUT can be constrainted in the same or different locations of the same CLS;

**Note!**

The two nets via inverter can not be placed in the same CLS.

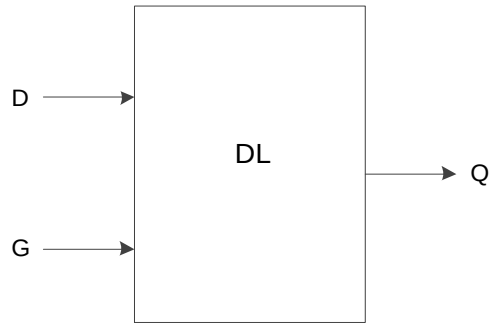
## 2.5.1 DL

### Primitive

The Data Latch ( DL ) is a kind of commonly used latch. The control signal G is active-high.

## Port Diagram

**Figure 2-31 DL Port Diagram**




## Port Description

**Table 2-69 Port Description**

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

## Parameter

**Table 2-70 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1’b0,1’b1 | 1’b0 | Initial value for initial DL |

## Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
    );
```

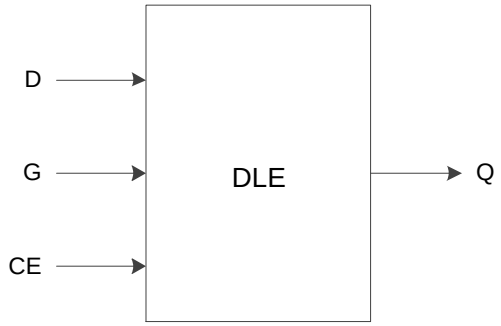
## 2.5.2 DLE

### Primitive

Data Latch with Latch Enable ( DLE ) is a latch with the function of enable control. The control signal G is active-high.

### Port Diagram

**Figure 2-32 DLE Port Diagram**




### Port Description

**Table 2-71 Port Description**

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

### Parameter

**Table 2-72 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value for initial DLE |

#### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
    );
```

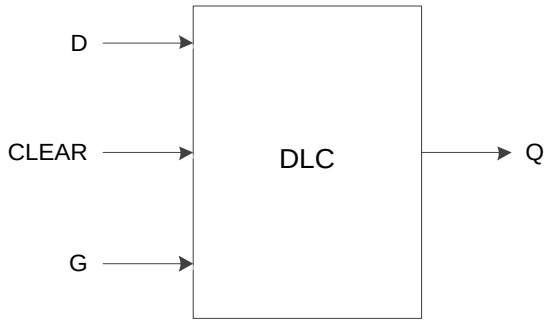
### 2.5.3 DLC

#### Primitive

Data Latch with Asynchronous Clear ( DLC ) is a latch with the function of clear. The control signal G is active-high.

## Port Diagram

**Figure 2-33 DLC Port Diagram**




## Port Description

**Table 2-73 Port Description**

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

## Parameter

**Table 2-74 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value for initial DLC |

## Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
    );
```

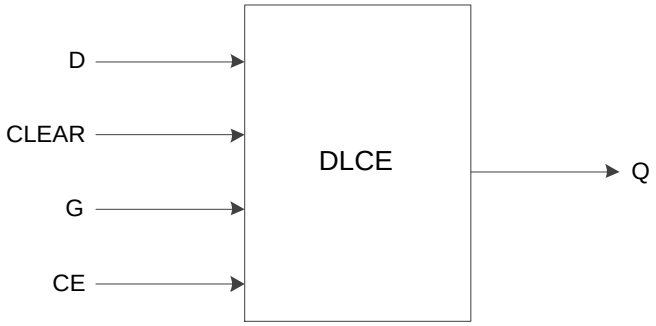
## 2.5.4 DLCE

### Primitive

Data Latch with Asynchronous Clear and Latch Enable ( DLCE ) is a latch with the function of enable control and clear. The control signal G is active-high.

### Port Diagram

**Figure 2-34 DLCE Port Diagram**




### Port Description

**Table 2-75 Port Description**

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

### Parameter

**Table 2-76 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value for initial DLCE |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

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

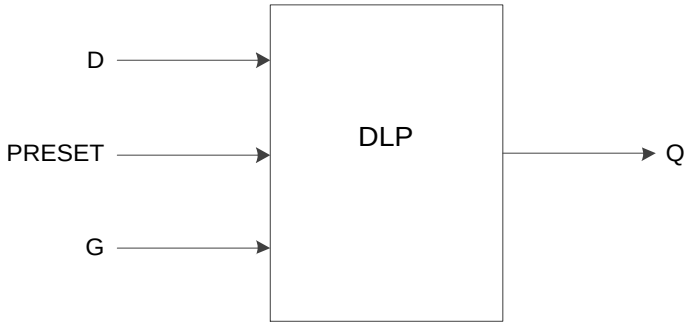
### Primitive

Data Latch with Asynchronous Preset ( DLP ) is a latch with preset

function. The control signal G is active-high.

## Port Diagram

**Figure 2-35 DLP Blcok Diagram**




## Port Description

**Table 2-77 Port Description**

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

## Parameter

**Table 2-78 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value for initial DLP |

## Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
    );
```

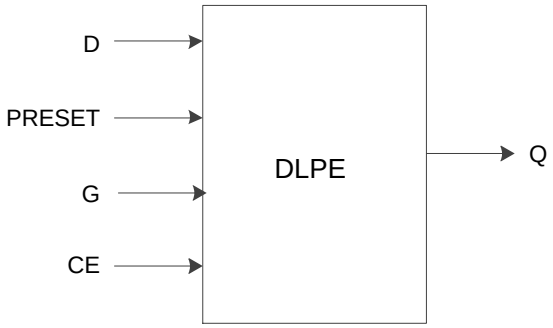
## 2.5.6 DLPE

### Primitive

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

### Port Diagram

**Figure 2-36 DLPE Port Diagram**




### Port Description

**Table 2-79 Port Description**

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

### Parameter

**Table 2-80 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value for initial DLPE |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
COMPONENT DLPE
    GENERIC (INIT:bit:='1');
    PORT(
      Q:OUT std_logic;
      D:IN std_logic;
      G:IN std_logic;
      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
     );
```

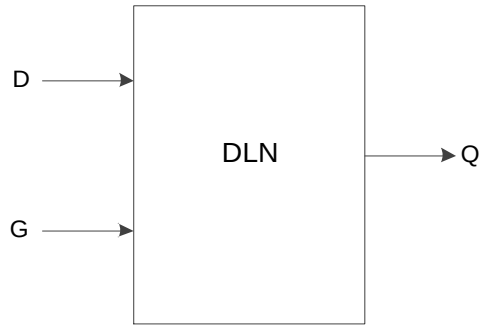
## 2.5.7 DLN

### Primitive

Data Latch with Inverted Gate ( DLN ) is a latch with the control signal active-low.

### Port Diagram

**Figure 2-37 DLNP Port Diagram**




### Port Description

**Table 2-81 Port Description**

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

### Parameter

**Table 2-82 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value for initial DLN |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
    );
```

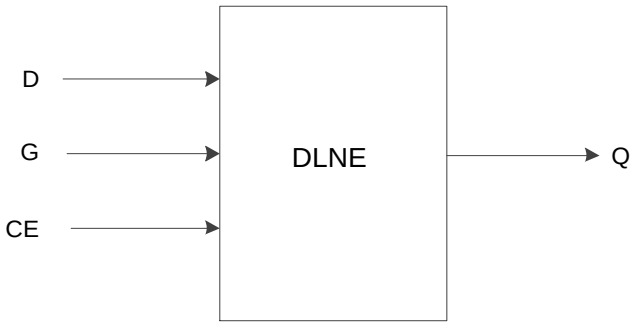
## 2.5.8 DLNE

### Primitive

Data Latch with Latch Enable and Inverted Gate ( DLNE ) is a latch with the function of enable control, and control signal G is active-low.

### Port Diagram

**Figure 2-38 DLNE Port Diagram**




### Port Description

**Table 2-83 Port Description**

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

### Parameter

**Table 2-84 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1’b0,1’b1 | 1’b0 | Initial value for initial DLNE |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
    );
```

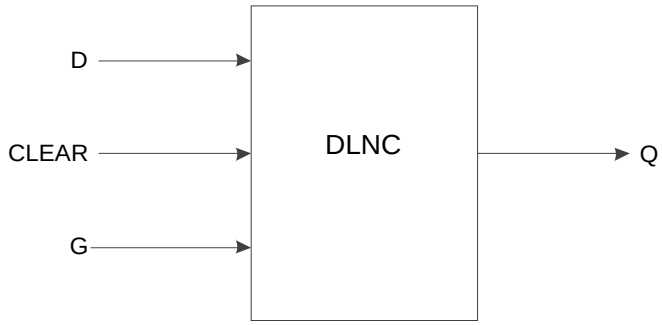
## 2.5.9 DLNC

### Primitive

Data Latch with Asynchronous Clear and Inverted Gate ( DLNC ) is a latch with the function of clear, and control signal G is active-low.

## Port Diagram

**Figure 2-39 DLNC Port Diagram**




## Port Description

**Table 2-85 Port Description**

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

## Parameter

**Table 2-86 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value for initial DLNC |

## Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
    );
```

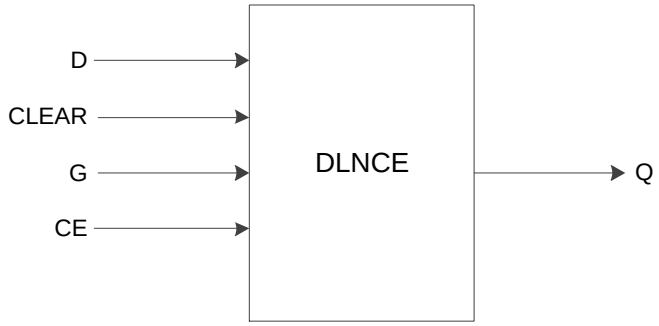
## 2.5.10 DLNCE

### Primitive

Data Latch with Asynchronous Clear, Latch Enable, and Inverted Gate ( DLNCE ) is a latch with the function of enable control and clear, and control signal G is active-low.

### Port Diagram

**Figure 2-40 DLNCE Port Diagram**




### Port Description

**Table 2-87 Port Description**

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

### Parameter

**Table 2-88 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value for initial DLNCE |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

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

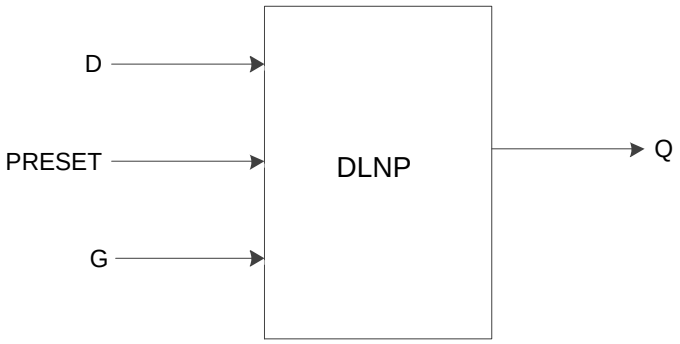
## 2.5.11 DLNP

### Primitive

Data Latch with Asynchronous Preset and Inverted Gate ( DLNP ) is a latch with the function of asynchronous preset, and control signal G is active-low.

### Port Diagram

**Figure 2-41 DLNP Port Diagram**




### Port Description

**Table 2-89 Port Description**

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

### Parameter

**Table 2-90 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value for initial DLNPE |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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
     );
```

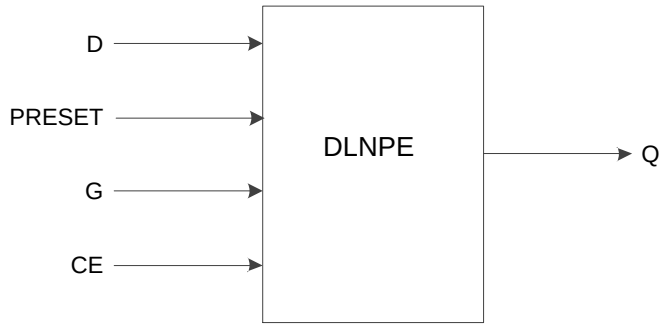
## 2.5.12 DLNPE

### Primitive

Data Latch with Asynchronous Preset,Latch Enable and Inverted Gate ( DLNPE ) is a latch with the function of enable control and preset, and control signal G is active-low.

### Port Diagram

**Figure 2-42 DLNPE Port Diagram**




### Port Description

**Table 2-91 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| D | Input | Data Input |

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

### Parameter

**Table 2-92 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value for initial DLNPE |

### Primitive Instantiation

**Verilog Instantiation:**

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

**Vhdl Instantiation:**

```
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

Gowin FPGA products provide abundant memory resources, including Block Static Random Access Memory (BSRAM) and Shadow Static Random Access Memory (SSRAM). For BSRAM and SSRAM primitives, see UG285, Gowin BSRAM & SSRAM User Guide.

# 4 DSP

Gowin FPGA products provide abundant DSP resources, which can meet the demand of high-performance digital signal processing. For DSP primitives, see UG287, Gowin DSP User Guide.

# 5 Clock

Gowin FPGA products provide dedicated global clock (GCLK, including PCLK and SCLK) directly connected to all resources of the devices. In addition to GCLK, PLL, HCLK and bidirectional data strobe circuit for DDR Memory (DQS) are also available. For clock primitives, see UG286, Gowin Clock User Guide.

# 6 User Flash

Gowin LittleBee® Family FPGA products provide User Flash. Different series of devices support different sizes of Flash, including FLASH96K, FLASH64K, FLASH64KZ, FLASH128K, FLASH256K and FLASH608K. For the Flash primitives, see UG295, Gowin Flash User Guide.

# 7 EMPU

## 7.1 MCU

### Primitive

ARM Cortex-M3 Microcontroller Unit (MCU) is a micro-processor based on the ARM Cortex-m3. The 32-bit AHB/APB bus is used. It supports the functions of two UARTs, two Timers and Watchdog. It also provides 16-bit GPIO, two UARTs, two JTAGs, two User Interrupt interfaces, AHB Flash read interface, AHB Sram read/write interface, two AHB bus extension interfaces, and one APB bus extension interface.

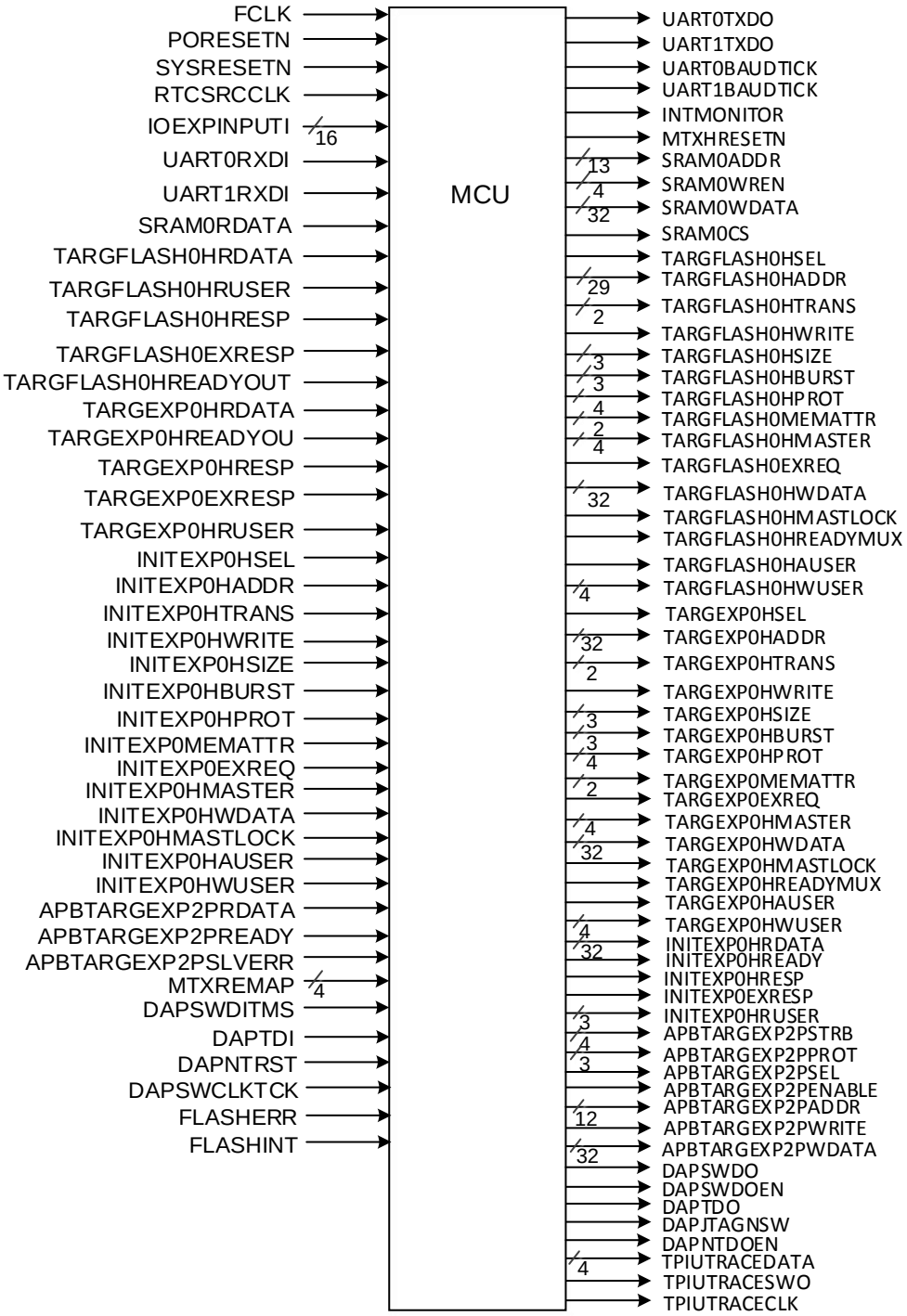
### Device Supported

**Table 7-1 Device Supported**

| Family | Series | Device |
|---|---|---|
| GW1N | GW1NS | GW1NS-2C |
| | GW1NSE | GW1NSE-2C |
| | GW1NSR | GW1NSR-2C |

## Port Diagram

**Figure 7-1 MCU Port Diagram**




## Port Description

**Table 7-2 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| FCLK | input | Free running clock |
| PORESETN | input | Power on reset |

| Port Name | I/O | Description |
|---|---|---|
| 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 |
| UART1TXDO | output | UART1TXDO |

| Port Name | I/O | Description |
|---|---|---|
| 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 |
| APBTARGEXP2PPROT[2:0] | output | APBTARGEXP2, PPROT |

| Port Name | I/O | Description |
|---|---|---|
| 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 |

### Primitive Instantiation

**Verilog Instantiation:**

```
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(targflash0mematttr[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(targexp0mematttr[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 Instantiation:**

```
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=>targexp0mematttr,
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

### Primitive

EMCU(ARM Cortex-M3 Microcontroller Unit) is a micro-processor based on the ARM Cortex-m3. The 32-bit AHB/APB bus is used. It supports the functions of two UARTs, two Timers and Watchdog. It also provides 16-bit GPIO, two UARTs and JTAGs, six User Interrupt interfaces, AHB Flash read interface, AHB Sram read/write interface, two AHB bus extension interfaces, and one APB bus extension interface.

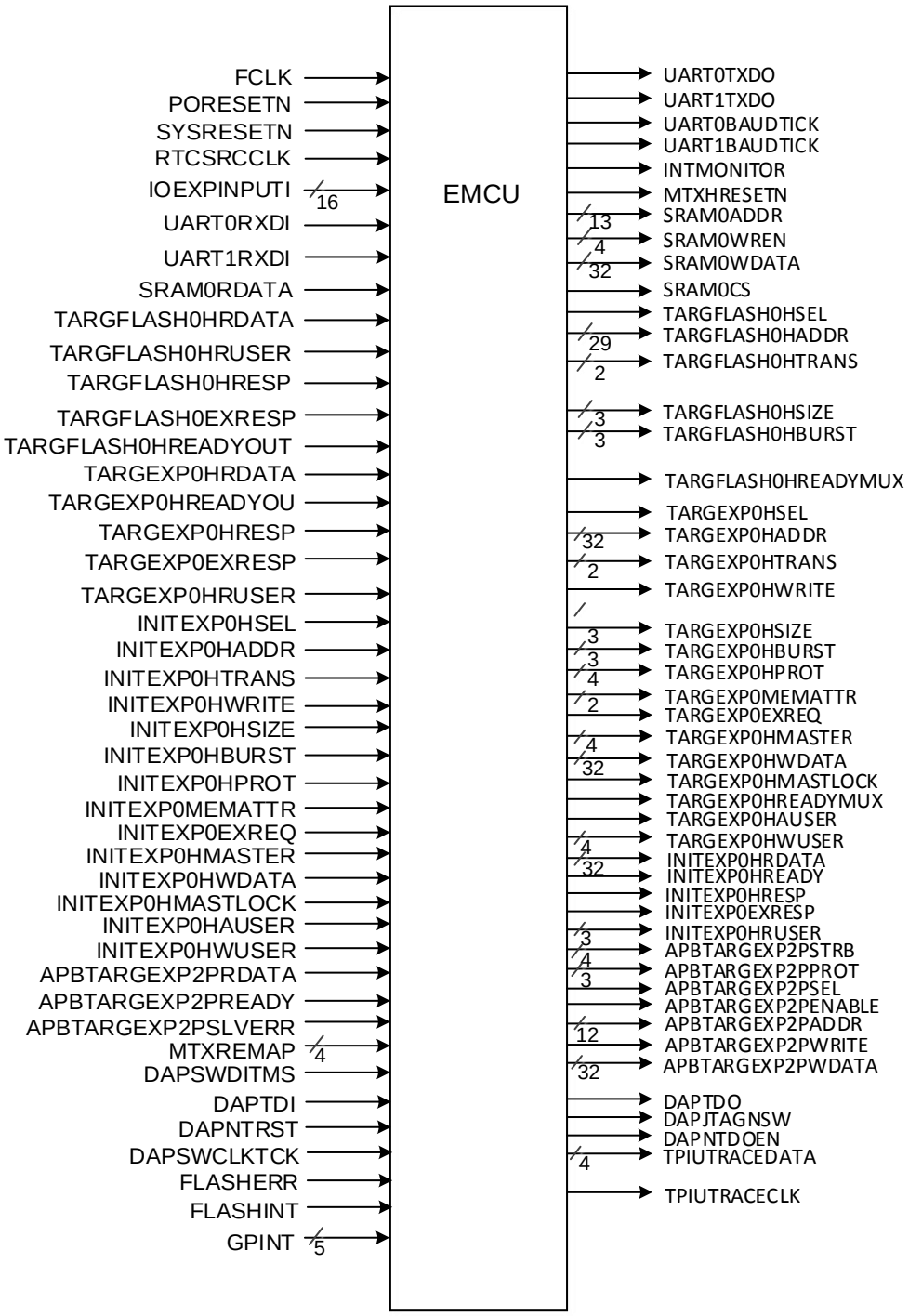
### Device Supported

**Table 7-3 Device Supported**

| Family | Series | Device |
|---|---|---|
| GW1N | GW1NS | GW1NS-4C |
| | GW1NSR | GW1NSR-4C |
| | GW1NSER | GW1NSER-4C |

## Port Diagram

**Figure 7-2 Port Diagram of EMCU**




## Port Description

**Table 7-4 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| FCLK | input | Free running clock |
| PORESETN | input | Power on reset |
| SYSRESETN | input | System reset |

| Port Name | I/O | Description |
|---|---|---|
| RTCSRCCLK | input | Used to generate RTC clock |
| IOEXPINPUTI[15:0] | input | IOEXPINPUTI |
| UART0RXDI | input | UART0RXDI |
| UART1RXDI | input | UART1RXDI |
| SRAM0RDATA[31:0] | input | SRAM Read data bus |
| TARGFLASH0HRDATA[31:0] | input | TARGFLASH0, HRDATA |
| TARGFLASH0HRUSER[2:0] | input | TARGFLASH0, HRUSER |
| TARGFLASH0HRESP | input | TARGFLASH0, HRESP |
| TARGFLASH0EXRESP | input | TARGFLASH0, EXRESP |
| TARGFLASH0HREADYOUT | input | TARGFLASH0, EXRESP |
| TARGEXP0HRDATA[31:0] | input | TARGEXP0, HRDATA |
| TARGEXP0HREADYOUT | input | TARGEXP0, HREADY |
| TARGEXP0HRESP | input | TARGEXP0, HRESP |
| TARGEXP0EXRESP | input | TARGEXP0, EXRESP |
| TARGEXP0HRUSER[2:0] | input | TARGEXP0, HRUSER |
| INITEXP0HSEL | input | INITEXP0, HSELx |
| INITEXP0HADDR[31:0] | input | INITEXP0, HADDR |
| INITEXP0HTRANS[1:0] | input | INITEXP0, HTRANS |
| INITEXP0HWRITE | input | INITEXP0, HWRITE |
| INITEXP0HSIZE[2:0] | input | INITEXP0, HSIZE |
| INITEXP0HBURST[2:0] | input | INITEXP0, HBURST |
| INITEXP0HPROT[3:0] | input | INITEXP0, HPROT |
| INITEXP0MEMATTR[1:0] | input | INITEXP0, MEMATTR |
| INITEXP0EXREQ | input | INITEXP0, EXREQ |
| INITEXP0HMASTER[3:0] | input | INITEXP0, HMASTER |
| INITEXP0HWDATA[31:0] | input | INITEXP0, HWDATA |
| INITEXP0HMASTLOCK | input | INITEXP0, HMASTLOCK |
| INITEXP0HAUSER | input | INITEXP0, HAUSER |
| INITEXP0HWUSER[3:0] | input | INITEXP0, HWUSER |
| APBTARGEXP2PRDATA[31:0] | input | APBTARGEXP2, PRDATA |
| APBTARGEXP2PREADY | input | APBTARGEXP2, PREADY |
| APBTARGEXP2PSLVERR | input | APBTARGEXP2, PSLVERR |
| MTXREMAP[3:0] | input | The MTXREMAP signals control the remapping of the boot memory range. |
| DAPSWDITMS | input | Debug TMS |
| DAPTDI | input | Debug TDI |
| DAPNTRST | input | Test reset |
| DAPSWCLKTCK | input | Test clock / SWCLK |
| FLASHERR | input | Output clock, used by the TPA to sample the other pins |
| FLASHINT | input | Output clock, used by the TPA to sample the other pins |
| GPINT | input | GPINT |
| IOEXPOUTPUTO[15:0] | output | IOEXPOUTPUTO |
| IOEXPOUTPUTENO[15:0] | output | IOEXPOUTPUTENO |
| UART0TXDO | output | UART0TXDO |
| UART1TXDO | output | UART1TXDO |

| Port Name | I/O | Description |
|---|---|---|
| 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 |

| Port Name | I/O | Description |
|---|---|---|
| TPIUTRACEDATA[3:0] | output | Output data |
| TPIUTRACECLK | output | Output clock, used by the TPA to sample the other pins |

### Primitive Instantiaton

**Verilog Instantiation:**

```
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(targexp0mematttr[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 Instantiation:**

```
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

## Primitive

The USB20_PHY is a complete mixed signal IP solution that can implement OTG connection from Soc to other special manufacture technology. USB20_PHY supports USB 2.0 480-Mbps protocol and data rate and it is backward compatible with USB 1.1 1.5-Mbps and 12-Mbps protocols and data rate.

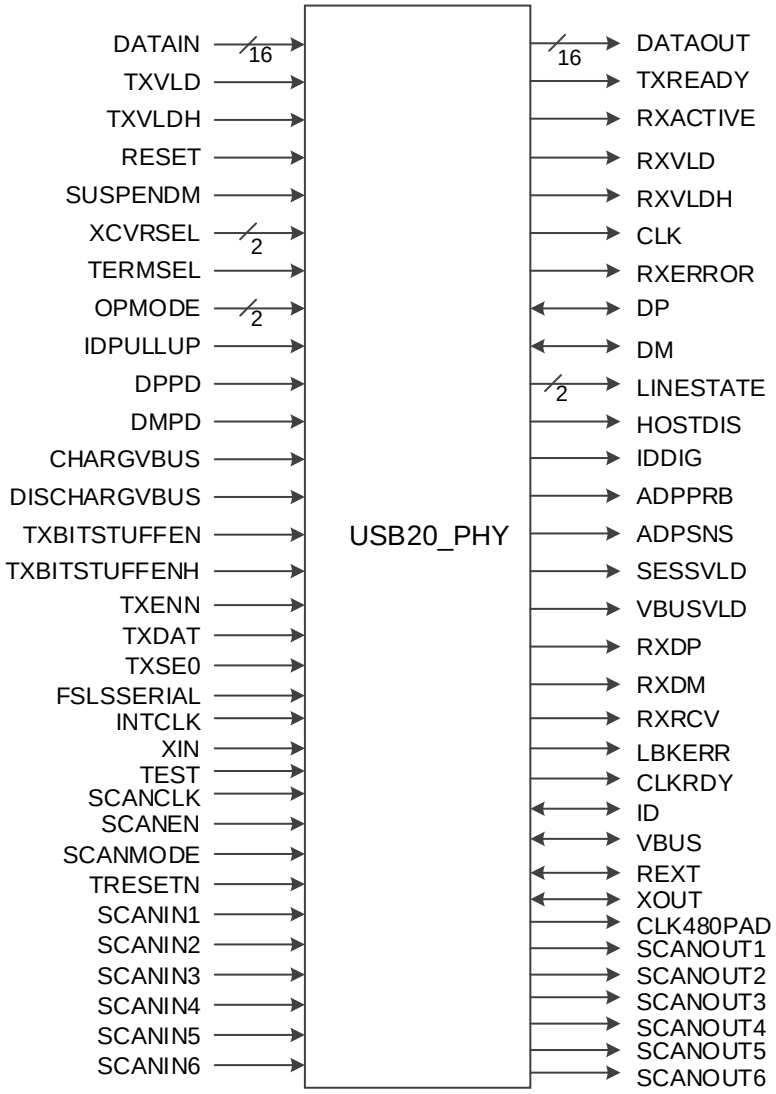
## Device Supported

**Table 7-5 Device Supported**

| Family | Series | Device |
|---|---|---|
| GW1N | GW1NS | GW1NS-2, GW1NS-2C |
| | GW1NSE | GW1NSE-2C |
| | GW1NSR | GW1NSR-2, GW1NSR-2C |

## Port Diagram

**Figure 7-3 USB20_PHY Port Diagram**




## Port Description

**Table 7-6 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| 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 |

| Port Name | I/O | Description |
|---|---|---|
| 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. |

| Port Name | I/O | Description |
|---|---|---|
| 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 |

## Parameter

**Table 7-7 Parameter**

| Name | Default | Description |
|---|---|---|
| DATABUS16_8 | 1’b0 | Selects between 8 and 16 bit data transfers. |
| ADP_PRBEN | 1’b0 | Enables/disables the ADP Probe comparator |
| TEST_MODE | 5’b0 | used for testing and debugging purpose |
| HSDRV1 | 1’b0 | High speed drive adjustment. Please connect to 0 for normal operation. |
| HSDRV0 | 1’b0 | High speed drive adjustment. Please connect to 0 for normal operation. |
| CLK_SEL | 1’b0 | Clock source selection signal. 0 to select external clock provided by the crystal connected on XIN, XOUT. 1 to select internal clock provided on INTCLK port |
| M | 4’b0 | M divider input data bit |
| N | 6'b101000 | N divider input data bit |
| 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) |

## Primitive Instantiation

### Verilog Instantiation:

```
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 Instantiation:
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,
0IDPULLUP=>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

## Primitive

It is an 8-channel, 12-bit, single port ADC with the features of low power, low leakage and high-dynamic.

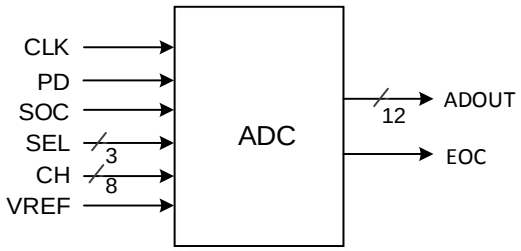
## Device Supported

**Table 7-8 Device Supported**

| Family | Series | Device |
|---|---|---|
| GW1N | GW1NS | GW1NS-2, GW1NS-2C |
| | GW1NSE | GW1NSE-2C |
| | GW1NSR | GW1NSR-2, GW1NSR-2C |

## Port Diagram

**Figure 7-4 ADC Port Diagram**




## Port Description

**Table 7-9 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| ADOUT[11:0] | Output | ad conversion results. |
| EOC | Output | end of conversion. |
| CLK | Input | main clock. |
| PD | Input | power down signal. |
| SOC | Input | start of conversion. |
| SEL[2:0] | Input | channel select signal. |
| CH[7:0] | Input | channel signal-ended analog voltage input. |
| VREF | Input | voltage reference |

## Primitive Instantiation

### Verilog Instantiation:

```
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 Instantiation:**

```
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
        );
```

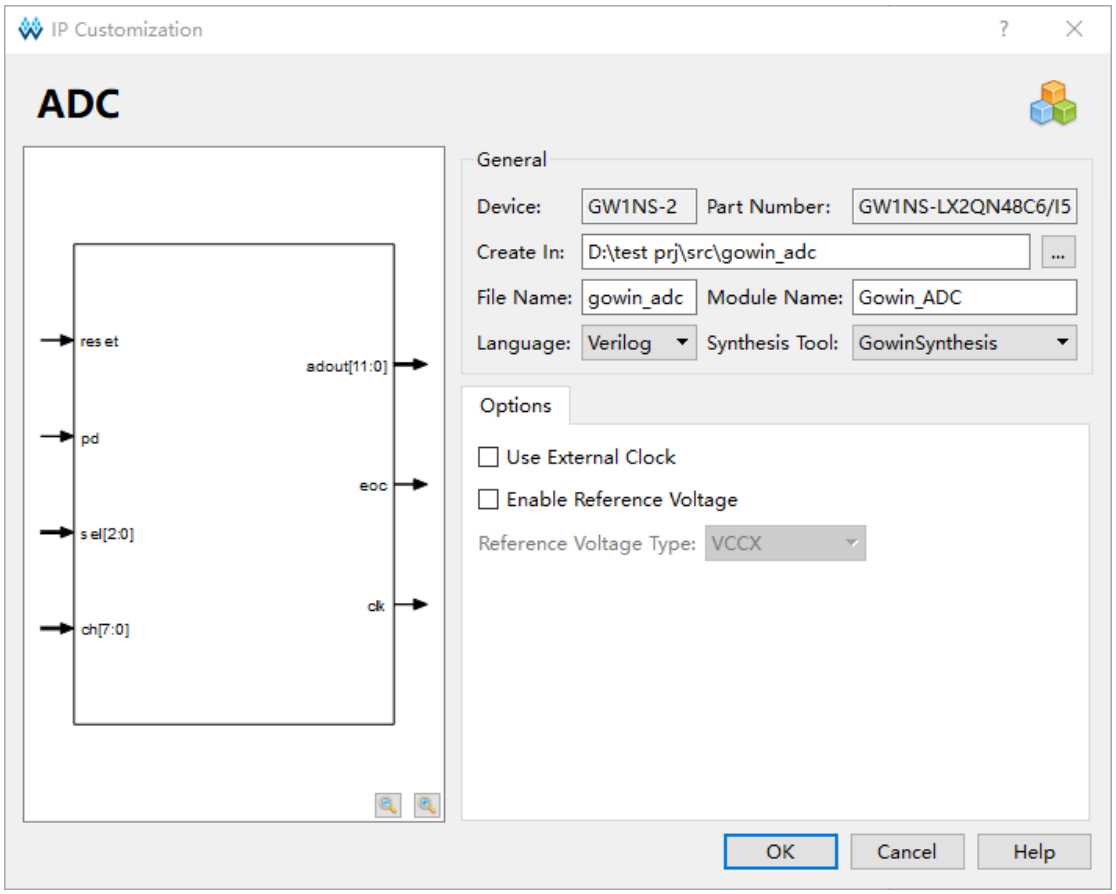
### Invoke IP

Click "ADC" on the IP Core Generator, and a brief introduction to the ADC will be displayed.

#### IP Configuration

Double-click the "ADC" to open the "IP Customization" window. This displays the "File", "Options", ports diagram, and the "Help", as shown in Figure 7-5.

**Figure 7-5 IP Customization of ADC**




1. File
   - Device: Selected device;
   - Part Number: Selected Part Number;
   - Create In: The target path. You can reedit in the textbox or select a path by clicking the button.
   - File Name: The file name. You can reedit in the textbox.
   - Module Name: The module name. You can reedit in the textbox. The module name can not be the same as the primitive name. If it is the same, an error prompt will pop up.
   - Language: Verilog and VHDL;
   - Synthesis Tool: GowinSynthesis and Synplify Pro.

   2. Options
   - Use External Clock: Configure external clock.
   - Enable Reference Voltage: Configure enable reference voltage, and the default is VCCX.
   - Reference Voltage Type: VCCX, 34/40(*VCCX), 31/40(*VCCX), 29/40(*VCCX), 27/40(*VCCX), 22/40(*VCCX), 20/40(*VCCX) and External.

   3. The ports diagram is based on the current IP Core configuration, as

shown in Figure 7-5;

4. Help
Click "Help" to open the IP Core configuration information. The Help page displays "Information" and "Options".

**Generated Files**

After configuration, it will generate three files that are named after the "File Name".

- "gowin_adc.v " file is a complete Verilog module to generate instance SPMI, and it is generated according to the IP configuration;
- "gowin_adc_tmp.v" is the instance template file;
- "gowin_adc.ipc" file is IP configuration file. The user can load the file to configure the IP.

**Note!**

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.

# 8 Miscellaneous

## 8.1 GSR

### Primitive

Global Reset/Set (GSR)

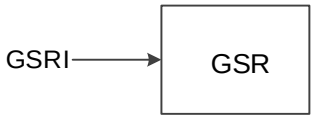
### Devices Supported

**Table 8-1 Device Supported**

| Family | Series | Device |
|---|---|---|
| GW2A | GW2A | GW2A-18, GW2A-18C, GW2A-55, GW2A-55C |
| | GW2AR | GW2AR-18, GW2AR-18C |
| | GW2ANR | GW2ANR-18C |
| GW1N | GW1N | GW1N-1, GW1N-1S, GW1N-4, GW1N-4B, GW1N-9, GW1N-9C |
| | GW1NR | GW1NR-4, GW1NR-4B, GW1NR-9, GW1NR-9C |
| | GW1NRF | GW1NRF-4B |
| | GW1NS | GW1NS-2, GW1NS-2C, GW1NS-4, GW1NS-4C |
| | GW1NSE | GW1NSE-2C |
| | GW1NSER | GW1NSER-4C |
| | GW1NSR | GW1NSR-2, GW1NSR-2C, GW1NSR-4, GW1NSR-4C |
| | GW1NZ | GW1NZ-1 |

### Port Diagram

**Figure 8-1 GSR Port Diagram**




### Port Description

**Table 8-2 Port Description**

| Name | I/O | Description |
|---|---|---|
| GSRI | Input | GSR input，active-low |

### Primitive Instantiation

**Verilog instantiation:**

```
GSR gsr_inst(
      .GSRI(GSRI)
);
```

**Vhdl instantiation:**

```
COMPONENT GSR
      PORT (
            GSRI:IN std_logic
      );
END COMPONENT;
gsr_inst:GSR
        PORT MAP(
          GSRI => GSRI
        );
```

# 8.2 INV

### Primitive

Inverter (INV)

### Devices Supported

**Table 8-3 Device Supported**

| Family | Series | Device |
|---|---|---|
| GW2A | GW2A | GW2A-18, GW2A-18C, GW2A-55, GW2A-55C |
| | GW2AR | GW2AR-18, GW2AR-18C |
| | GW2ANR | GW2ANR-18C |
| GW1N | GW1N | GW1N-1, GW1N-1S, GW1N-4, GW1N-4B, GW1N-9, GW1N-9C |
| | GW1NR | GW1NR-4, GW1NR-4B, GW1NR-9, GW1NR-9C |
| | GW1NRF | GW1NRF-4B |
| | GW1NS | GW1NS-2, GW1NS-2C, GW1NS-4, GW1NS-4C |
| | GW1NSE | GW1NSE-2C |
| | GW1NSER | GW1NSER-4C |
| | GW1NSR | GW1NSR-2, GW1NSR-2C, GW1NSR-4, GW1NSR-4C |
| | GW1NZ | GW1NZ-1 |

### Port Diagram

**Figure 8-2 INV Port Diagram**

### Port Description

**Table 8-4 Port Description**

| Name | I/O | Description |
|---|---|---|
| I | Input | INV data input |
| O | Output | INV data output |

### Primitive Instantiation

**Verilog instantiation:**

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

**Vhdl instantiation:**

```
COMPONENT INV
      PORT (
            O:OUTPUT std_logic;
            I:IN std_logic

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

## 8.3 VCC

### Primitive

VCC

### Devices Supported

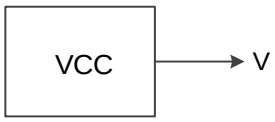
**Table 8-5 Device Supported**

| Family | Series | Device |
|---|---|---|
| GW2A | GW2A | GW2A-18, GW2A-18C, GW2A-55, GW2A-55C |
| | GW2AR | GW2AR-18, GW2AR-18C |
| | GW2ANR | GW2ANR-18C |
| GW1N | GW1N | GW1N-1, GW1N-1S, GW1N-4, GW1N-4B, GW1N-9, GW1N-9C |
| | GW1NR | GW1NR-4, GW1NR-4B, GW1NR-9, GW1NR-9C |
| | GW1NRF | GW1NRF-4B |

| Family | Series | Device |
|---|---|---|
| | GW1NS | GW1NS-2, GW1NS-2C, GW1NS-4, GW1NS-4C |
| | GW1NSE | GW1NSE-2C |
| | GW1NSER | GW1NSER-4C |
| | GW1NSR | GW1NSR-2, GW1NSR-2C, GW1NSR-4, GW1NSR-4C |
| | GW1NZ | GW1NZ-1 |

### Port Diagram

**Figure 8-3 VCC Port Diagram**




### Port Description

**Table 8-6 Port Description**

| Name | I/O | Description |
|---|---|---|
| V | Output | VCC output |

### Primitive Instantiation

**Verilog instantiation:**

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

**Vhdl instantiation:**

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

## 8.4 GND

### Primitive

GND

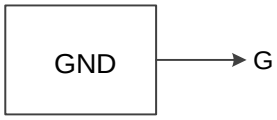
## Devices Supported

**Table 8-7 Device Supported**

| Family | Series | Device |
|---|---|---|
| GW2A | GW2A | GW2A-18, GW2A-18C, GW2A-55, GW2A-55C |
| | GW2AR | GW2AR-18, GW2AR-18C |
| | GW2ANR | GW2ANR-18C |
| GW1N | GW1N | GW1N-1, GW1N-1S, GW1N-4, GW1N-4B, GW1N-9, GW1N-9C |
| | GW1NR | GW1NR-4, GW1NR-4B, GW1NR-9, GW1NR-9C |
| | GW1NRF | GW1NRF-4B |
| | GW1NS | GW1NS-2, GW1NS-2C, GW1NS-4, GW1NS-4C |
| | GW1NSE | GW1NSE-2C |
| | GW1NSER | GW1NSER-4C |
| | GW1NSR | GW1NSR-2, GW1NSR-2C, GW1NSR-4, GW1NSR-4C |
| | GW1NZ | GW1NZ-1 |

## Port Diagram

**Figure 8-4 GND Port Diagram**




## Port Description

**Table 8-8 Port Description**

| Name | I/O | Description |
|---|---|---|
| G | Output | GND output |

## Primitive Instantiation

**Verilog instantiation:**

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

**Vhdl instantiation:**

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

```
);
```

# 8.5 BANDGAP

### Primitive

BANDGAP

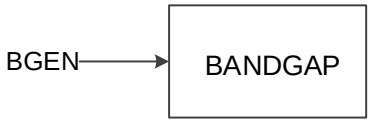
### Supported Devices

**Table 8-9 Device Supported**

| Family | Series | Device |
|---|---|---|
| GW1N | GW1NZ | GW1NZ-1 |

### Port Diagram

**Figure 8-5 BANDGAP Port Diagram**




### Port Description

**Table 8-10 Port Description**

| Name | I/O | Description |
|---|---|---|
| BGEN | Input | BANDGAP enable signal, active-high |

### Primitive Instantiation

**Verilog Instantiation:**

```
BANDGAP uut (
    .BGEN(bgen)
);
```

**Vhdl Instantiation:**

```
COMPONENT BANDGAP
    PORT (
        BGEN:IN std_logic
    );
END COMPONENT;
uut:BANDGAP
      PORT MAP(
     BGEN=> I
     );
```

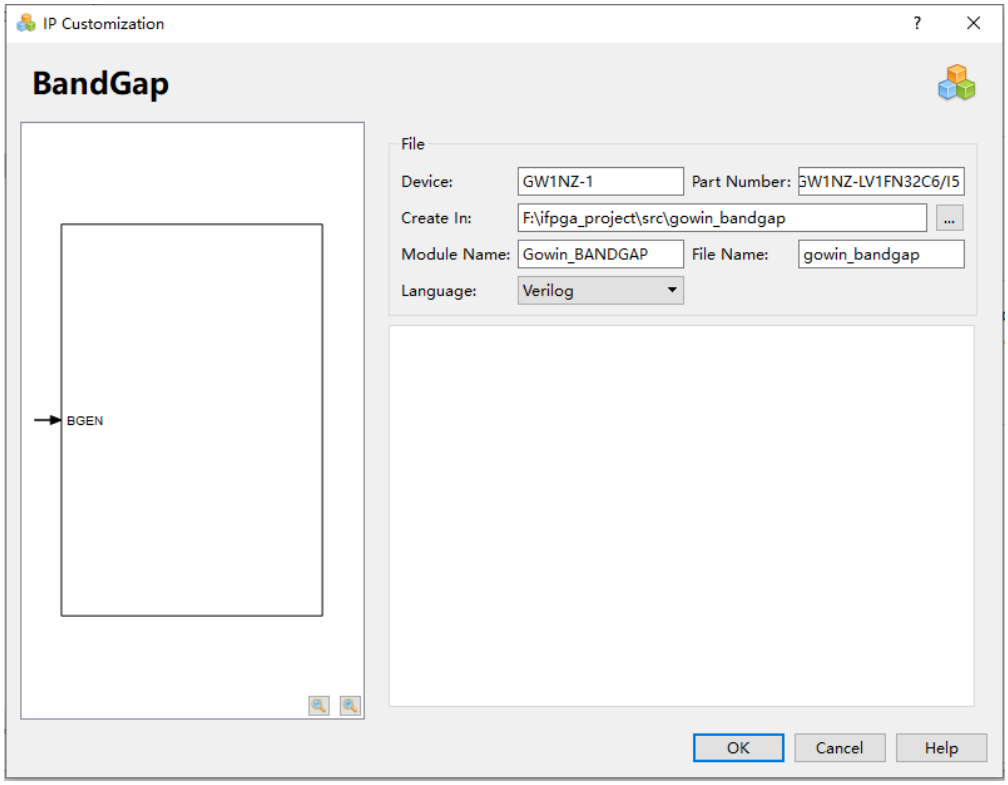
### Invoke IP

Click "BandGap" on the IP Core Generator, and a brief introduction to the BandGap will be displayed.

### IP Configuration

Double-click the "BandGap" to open the "IP Customization" window. This displays the "File", "Options", ports diagram, and the "Help", as shown in Figure 8-6.

**Figure 8-6 IP Customization of BandGap**




1. File
   - The File displays the basic information related to BandGap.
   - The BandGap file configuration is similar to that of SP. For the detailed configuration, please see 7.4 ADC > Invoke IP.
2. The ports diagram is based on the current IP Core configuration, as shown in Figure 8-6;
3. Help
   Click "Help" to open the IP Core configuration information. The Help page displays "Information".

### Generated Files

After configuration, it will generate three files that are named after the "File Name".

- "gowin_bandgap.v " file is a complete Verilog module to generate instance BandGap, and it is generated according to the IP configuration;
- "gowin_bandgap_tmp.v" is the instance template file;
- "gowin_bandgap.ipc" file is IP configuration file. The user can load the file to configure the IP.

**Note!**

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.

# 8.6 SPMI

## Primitive Instantiation

System Power Management Interface ( SPMI ) is a two-wire serial interface, which can be used to dynamically control the internal power supply of the on-chip system.

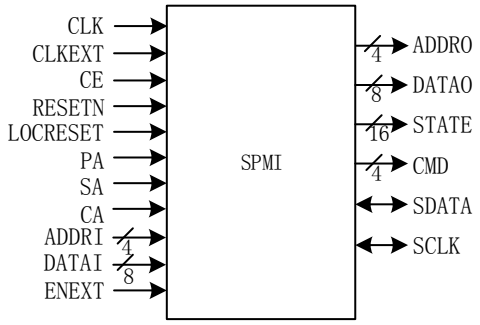
## Supported Devices

**Table 8-11 Device Supported**

| Family | Series | Device |
|---|---|---|
| GW1N | GW1NZ | GW1NZ-1 |

## Port Diagram

**Figure 8-7 SPMI Port Diagram**




## Port Description

**Table 8-12 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| CLK | input | Clock input |
| CLKEXT | input | External clock input |
| CE | input | Clock Enable |
| RESETN | input | Reset input |
| ENEXT | input | Enext input |
| LOCRESET | input | Local reset input |
| PA | input | Priority arbitration input |
| SA | input | Secondary arbitration input |
| CA | input | Connection arbitration input |
| ADDRI | input | Addr input |
| DATAI | input | Data input |
| ADDRO | output | Addr output |
| DATAO | output | datat output |
| STATE | output | state output |
| CMD | output | command output |

| Port Name | I/O | Description |
|---|---|---|
| SDATA | inout | SPMI Serial data |
| SCLK | inout | SPMI Serial Clock |

### Primitive Instantiation

**Verilog Instantiation:**

```
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 Instantiation:**

```
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
        );
```

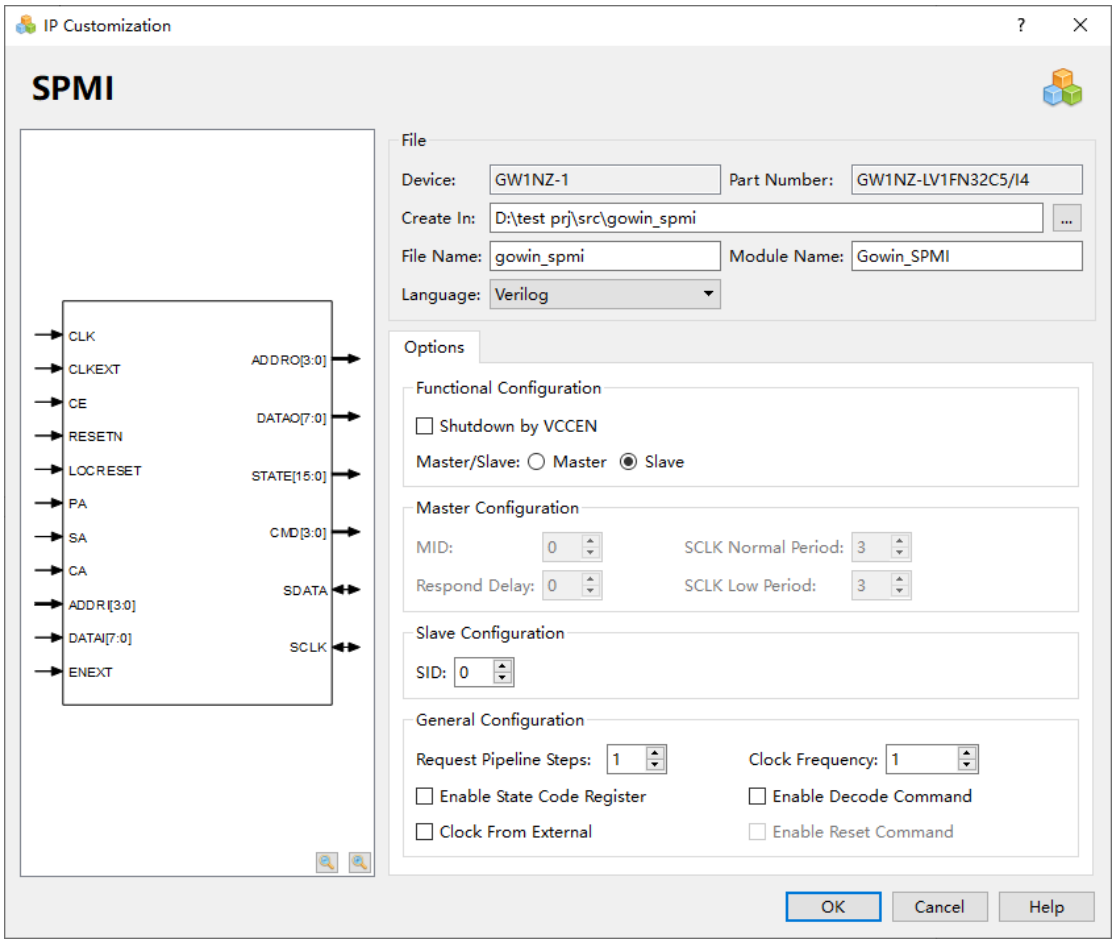
### Invoke IP

Click "SPMI" on the IP Core Generator, and a brief introduction to the SPMI will be displayed.

#### IP Configuration

Double-click the "SPMI" to open the "IP Customization" window. This displays the "File", "Options", ports diagram, and the "Help", as shown in Figure 8-8.

**Figure 8-8 IP Customization of SPMI**




5. File

- The File displays the basic information related to SPMI.
- The SPMI file configuration box is similar to that of SP. For the details, please see 7.4 ADC > Invoke IP.

6. Options

- The Options is used to configure SPMI by users, as shown in Figure 8-8.
- Functional Configuration：
  - Shutdown by VCCEN：Shutdown by external pin VCCEN If this option is checked, the communication function of SPMI will be disabled.
  - Master/Slave: Set SPMI as Master or Slave.
- Master Configuration:
  - MID: Master ID. The range is 0-3, and default value is 0.
  - Respond Delay: Set the response delay.
  - SCLK Normal Period: Set SCLK period in normal mode.
  - SCLK Normal Period: Set SCLK period in low mode.

- Slave Configuration:
  SID: Slave ID.
- General configuration:
  - Enable State Code Register: Enable or disable the state code register. If "Enable State Code Register" is checked, the output state code will pass a register.
  - Request Pipeline Steps: Set the sampling delay step of the request signal.
  - Enable Decode Command: Enable or disable decode. If "Enable Decode Command" is checked, SPMI will decode the reset, sleep, shutdown, and wakeup.
  - Enable Decode Command: Enable or disable reset.
  - Clock From External: Enable or disable the external clock.
  - Clock Frequency: System clock frequency.

7. The ports diagram is based on the current IP Core configuration, as shown in;

8. Help
Click "Help" to open the IP Core configuration information. The Help page displays "Information" and "Options".

#### Generated Files

After configuration, it will generate three files that are named after the "File Name".

- "gowin_spmi.v " file is a complete Verilog module to generate instance SPMI, and it is generated according to the IP configuration;
- "gowin_spmi_tmp.v" is the instance template file;
- "gowin_spmi.ipc " file is IP configuration file. The user can load the file to configure the IP.

**Note!**

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.

## 8.7 I3C

### Primitive

I3C (Improved Inter Integrated Circuit) is a two-wire bus with the key features of I2C and SPI, which can effectively reduce the physical ports of integrated circuit, support the advantages of low power, high data rate and other existing port protocols.

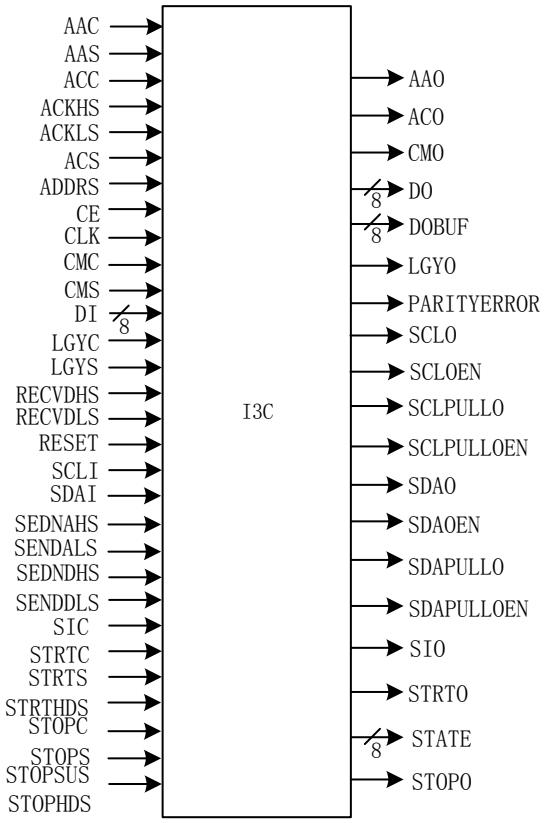
### Supported Devices

**Table 8-13 Device Supported**

| Family | Series | Device |
|---|---|---|
| GW1N | GW1NZ | GW1NZ-1 |

## Port Diagram

**Figure 8-9 I3C Port Diagram**




## Port Description

**Table 8-14 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| CE | input | Clock Enable |
| RESET | input | Reset input |
| CLK | input | Clock input |
| LGYS | input | The current communication object is the I2C setting signal |
| CMS | input | The device enters the Master's set signal |
| ACS | input | Select the setting signal when determining whether to continue. |
| AAS | input | Reply the ACK setting signal when a reply is required from the ACK/NACK |
| STOPS | input | Input the STOP command |
| STRTS | input | Input the START command. |
| LGYC | input | The current communication object is the I2C |
| CMC | input | The reset signal that the device is in master. |
| ACC | input | The reset signal that selects continue when selecting whether to continue |
| AAC | input | Reply the ACK reset signal when a reply is required from the ACK/NACK |
| SIC | input | Interrupt to identify the reset signal |
| STOPC | input | The reset signal is in STOP state |

| Port Name | I/O | Description |
|---|---|---|
| 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 |

### Primitive Instantiation

Verilog Instantiation:

```
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)
    );
```

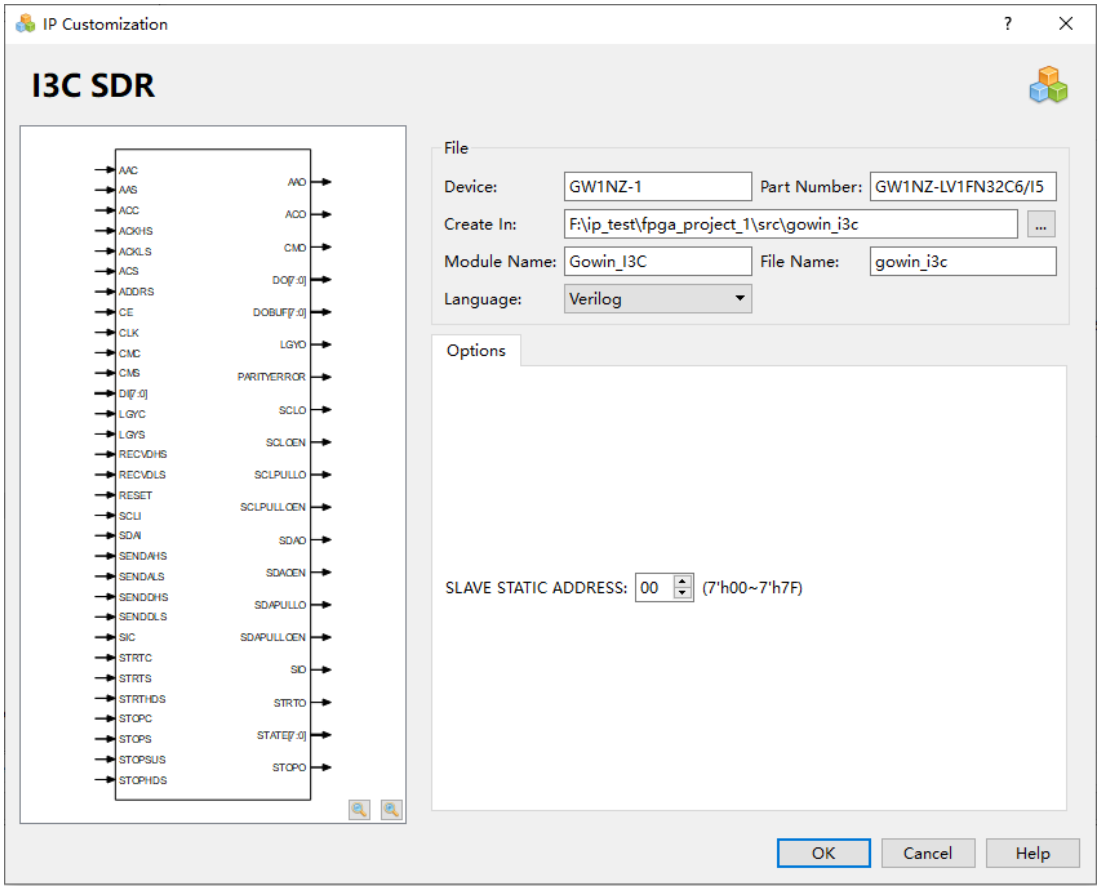
### Invoke IP

Click "I3C > I3C SDR" on the "IP Core Generator" page. A brief introduction to the I3C SDR will be displayed.

### Configure IP

Double-click "I3C SDR", and the "IP Customization" window pops up. This displays the "File", "Options", port diagram, and "Help", as shown in Figure 8-10.

**Figure 8-10 IP Customization of I3C**




1. File

   - The File displays the basic information related to the I3C.

- The I3C file configuration is similar to that of ADC. For the detailed configuration instructions, please see 7.4 ADC > Invoke IP.

2. Options

- The Options is used to configure I3C by users, as shown in Figure 8-10.
- SLAVE STATIC ADDRESS: Specify the static address of the Slave.

3. The ports diagram is based on the IP Core configuration, as shown in;

4. Help
   Click "Help" to open the IP Core configuration information. The Help page displays "Information" and "Options".

**Generated Files**

After configuration, it will generate three files that are named after the "File Name".

- "gowin_i3c.v " file is a complete Verilog module to generate instance I3C, and it is generated according to the IP configuration;
- "gowin_i3c_tmp.v" is the instance template file;
- "gowin_i3c.ipc " file is IP configuration file. You can load the file to configure the IP.

**Note!**

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.


GOWIN
PROGRAMMING FOR THE FUTURE