# Gowin Configurable Function Unit (CFU)

# User Manual

**UG288-1.09E, 12/17/2020**

## Disclaimer

## Revision History

| Date | Version | Description |
|---|---|---|
| 06/12/2018 | 1.08E | Initial version. |
| 12/17/2020 | 1.09E | Introduction to CFU updated. |

# Contents

# List of Figures

# List of Tables

# 1 About This Guide

## 1.1 Purpose

Gowin Configurable Function Unit (CFU) User Guide describes the structure, operation modes, and primitives of CFU.

## 1.2 Related Documents

The latest user guides are available on the Gowin Website. Refer to the related documents at www.gowinsemi.com:

- DS102, GW2A series of FPGA Products Data Sheet
- DS100, GW1N series of FPGA Products Data Sheet
- DS226, GW2AR series of FPGA Products Data Sheet
- UG285, Gowin BSRAM & SSRAM User Guide

## 1.3 Abbreviations and Terminology

Table 1-1 shows the abbreviations and terminology used in this guide.

**Table 1-1 Abbreviations and Terminology**

| Abbreviations and Terminology | Full Name |
|---|---|
| CFU | Configurable Function Unit |
| CLU | Configurable Logic Unit |
| LUT | Look-up Table |
| CRU | Configurable Routing Unit |
| SSRAM | Shadow Static Random Access Memory |
| BSRAM | Block Static Random Access Memory |
| ROM | Read Only Memory |
| CLS | Configurable Logic Section |
| REG | Register |
| MUX2 | Multiplexer 2:1 |
| ALU | Arithmetic Logic Unit |
| DFF | D Flip Flop |
| DL | Data Latch |

# 1.4 Support and Feedback

Gowin Semiconductor provides customers with comprehensive technical support. If you have any questions, comments, or suggestions, please feel free to contact us directly using any of the methods listed below.

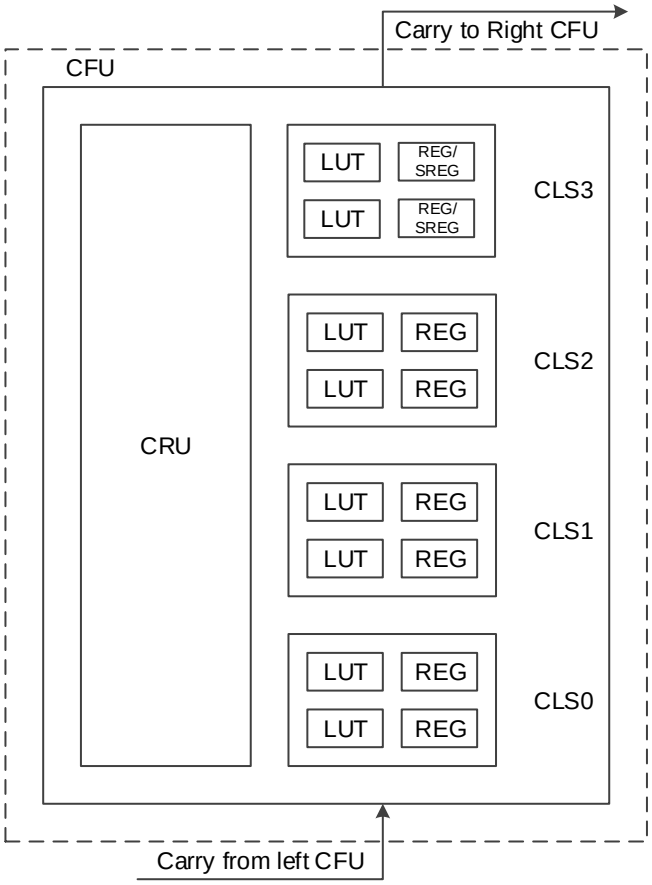
Website: www.gowinsemi.com

E-mail: support@gowinsemi.com

# 2 Configurable Function Unit

The configurable function unit and the configurable logic unit are two basic units for FPGA core of GOWINSEMI. As shown in Figure 2-1 each unit consists of four configurable logic sections and its configurable routing unit. Each of the three configurable logic sections contains two 4-input LUTs and two registers, and the other one only contians two 4-input LUTs. Configurable logical sections in CLU cannot be configured as SRAM, but as basic logic, ALU, and ROM. The configurable logic sections in the CFU can be configured as basic logic, ALU, SRAM, and ROM depending on the applications. This manual takes CFU as an example to introduce CFU and CLU.

**Figure 2-1 CFU Diagram**



**Note!**

- SREG needs special patch supporting. Please contact Gowin technical support or local office for this patch.
- At present, only GW1N-2 and GW1NZ-2 devices support REG of CLS3. And CLK, CE, and SR of CLS3 and CLS2 are driven by the same source.

# 2.1 CLS

## 2.1.1 CLS

The CLS supports three operation modes: basic logic mode, ALU mode, and memory mode.

- Basic Logic Mode

Each LUT can be configured as one four input LUT. A higher input number of LUT can be formed by combining LUT4 together.

  - One CLS can form one five input LUT (LUT5).
  - Two CLSs can form one six input LUT6 (LUT6).
  - Four CLSs can form one seven input LUT7 (LUT7).
  - Eight CLSs (two CLUs) can form one eight input LUT (LUT8).

- ALU Mode

When combined with carry chain logic, the LUT can be configured as the ALU mode to implement the following functions.

  - Adder and subtractor
  - Up/down counter
  - Comparator, including greater-than, less-than, and not-equal-to
  - MULT

- Memory mode

In this mode, a 16 x 4 SRAM or ROM16 can be formed by using one CFU.

## 2.1.2 REG

There are two registers in the configurable logic section (CLS0~CLS2), as shown in Figure 2-2 below.

**Figure 2-2 Register in CFU**

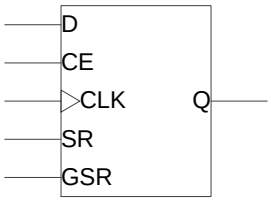



**Table 2-1 Register Description in CFU**

| Signal | I/O | Description |
|---|---|---|
| D | I | Data input [1] |

| Signal | I/O | Description |
|---|---|---|
| CE | I | CLK enable, can be high or low effective [2] |
| CLK | I | Clock, can be rising edge or falling edge trigging [2] |
| SR | I | Set/Reset, can be configured as [2]:<br>● Synchronized reset<br>● Synchronized set<br>● Asynchronous reset<br>● Asynchronous set<br>● Non |
| GSR[3,4] | I | Global Set/Reset, can be configured as[4]:<br>● Asynchronous reset<br>● Asynchronous set<br>● Non |
| Q | O | Register output |

**Note!**

- [1] The source of the signal D can be the output of a LUT, or the input of the CRU; as such, the register can be used alone when LUTs are in use.
- [2] CE, CLK, and SR in CFU are independent.
- [3] In Gowin FPGA products, GSR has its own dedicated network.
- [4] When both SR and GSR are effective, GSR has higher priority.

## 2.2 CRU

The main functions of the CRU are as follows:

- Input selection: Select input signals for the CFU.
- Configurable routing: Connect the input and output of the CFUs, including inside the CFU, CFU to CFU, and CFU to other functional blocks in FPGA.

# 3 CFU Primitives

## 3.1 LUT

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

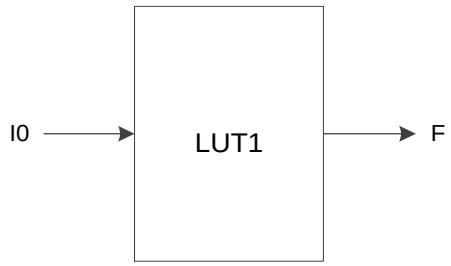
### 3.1.1 LUT1

#### Primitive

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

#### Port Diagram

**Figure 3-1 LUT1 Port Diagram**




#### Port Description

**Table 3-1 Port Description**

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

#### Parameter

**Table 3-2 Parameter**

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

#### Truth Table

**Table 3-3 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
     );
```

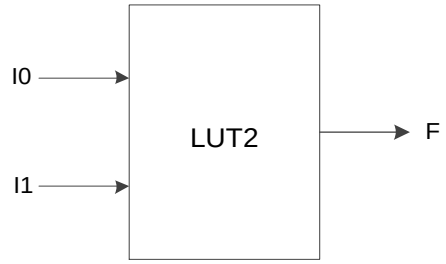
### 3.1.2 LUT2

#### Primitive

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

## Port Diagram

**Figure 3-2 LUT2 Port Diagram**




## Port Description

**Table 3-4 Port Description**

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

## Parameter

**Table 3-5 Parameter**

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

## Truth Table

**Table 3-6 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
     );
```

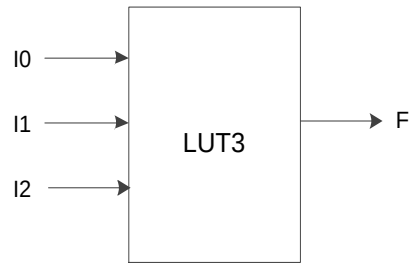
### 3.1.3 LUT3

#### Primitive

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

#### Port Diagram

**Figure 3-3 LUT3 Port Diagram**




#### Port Description

**Table 3-7 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 3-8 Parameter**

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

### Truth Table

**Table 3-9 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
    );
```

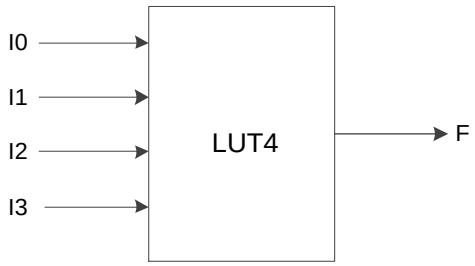
## 3.1.4 LUT4

### Primitive

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

### Port Diagram

**Figure 3-4 LUT4 Diagram**




### Port Description

**Table 3-10 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 3-11 Parameter**

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

### Truth Table

**Table 3-12 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
    );
```

### 3.1.5 Wide LUT

**Primitive**

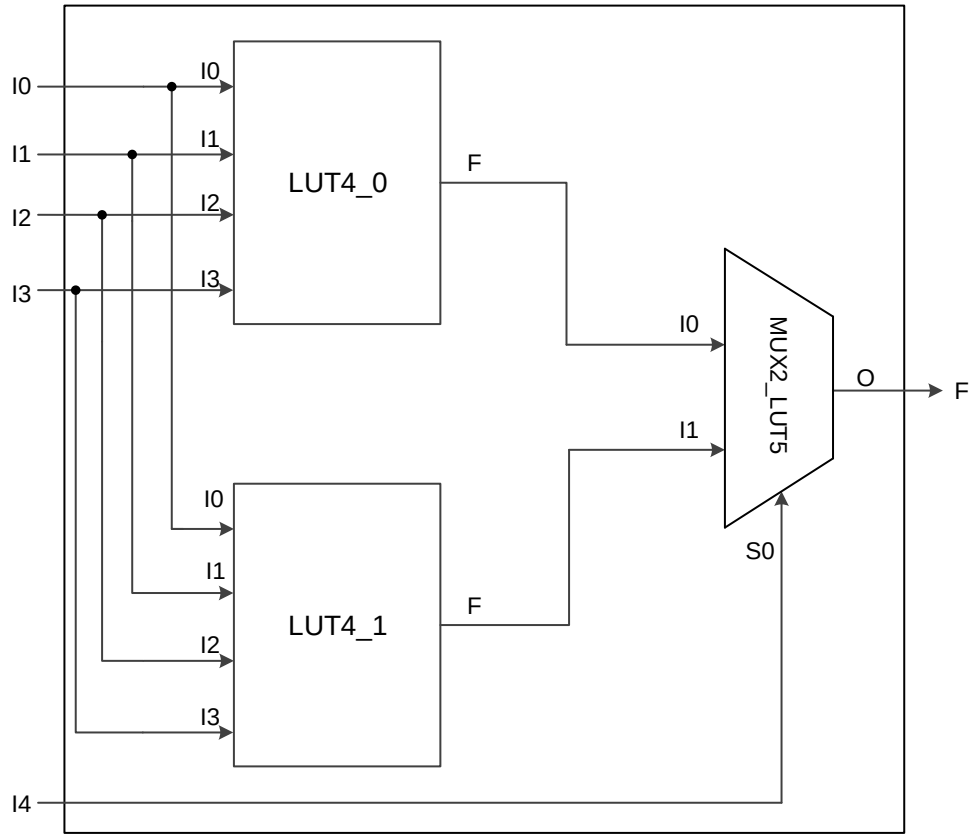
Wide LUT is used for forming higher input number of LUT by LUT4 and MUX2. MUX2_LUT5/ MUX2_LUT6/ MUX2_LUT7/ MUX2_LUT8 of Gowin MUX2 support this higher formation.

The way of the higher formation 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 3-5 LUT5 Port Diagram**




## Port Description

**Table 3-13 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 3-14 Parameter**

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

### Truth Table

**Table 3-15Truth 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:

LUT5 instName (

```
    .I0(i0),
    .I1(i1),
    .I2(i2),
    .I3(i3),
    .I4(i4),
    .F(f0)
);
defparam instName.INIT=32'h00000000;
```

**Vhdl Instantiation:**

```
COMPONENT LUT5
    PORT(
        F:OUT std_logic;
        I0:IN std_logic;
        I1:IN std_logic;
        I2:IN std_logic;
        I3:IN std_logic;
        I4:IN std_logic
    );
END COMPONENT;
uut:LUT5
    GENERIC MAP(INIT=>X"00000000")
    PORT MAP (
        F=>f0,
        I0=>i0,
        I1=>i1,
        I2=>i2,
        I3=>i3,
        I4=>i4
    );
```

# 3.2 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.

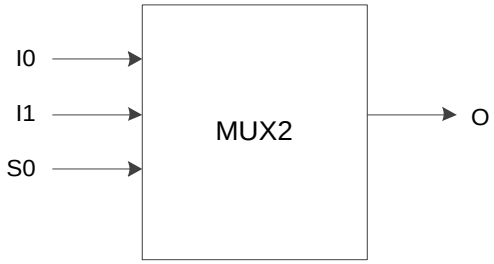
## 3.2.1 MUX2

### Primitive

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

### Port Diagram

**Figure 3-6 MUX2 Port Diagram**




### Port Description

**Table 3-16 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 3-17 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
       );
```

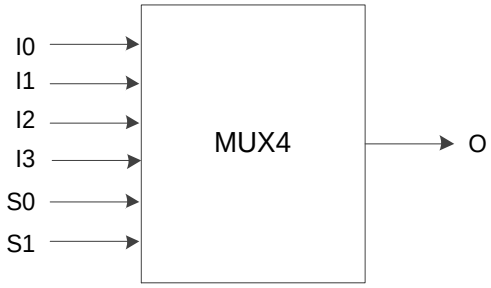
### 3.2.2 MUX4

**Primitive**

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

**Port Diagram**

**Figure 3-7 MUX4 Port Diagram**




**Port Description**

**Table 3-18 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 |

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

### Truth Table

**Table 3-19 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
);
```

### 3.2.3 Wide MUX

#### Primitive

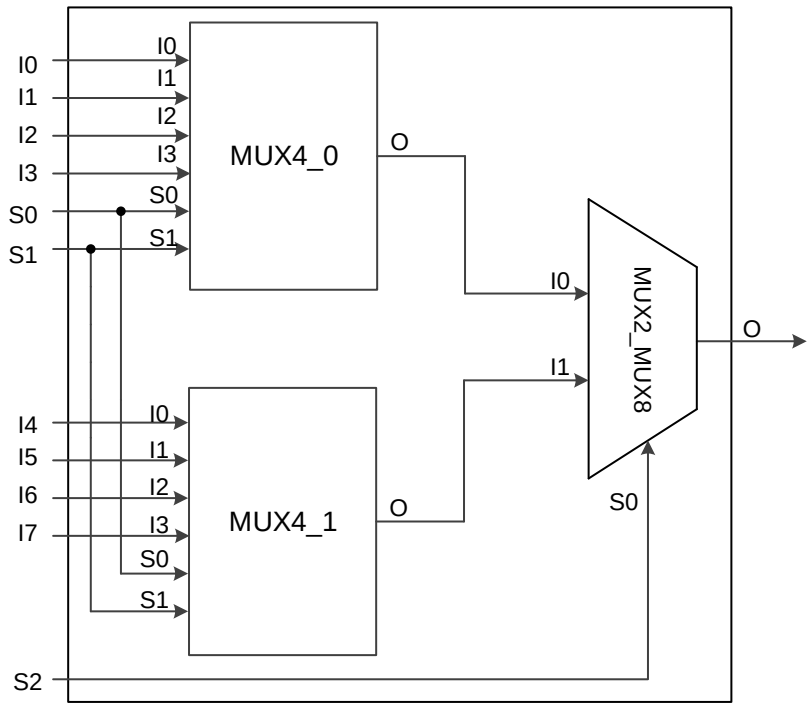
Wide MUX is used for forming higher input number of MUX by MUX4 and MUX2. MUX2_MUX8/ MUX2_MUX16/ MUX2_MUX32 of Gowin MUX2 support this formation.

The way of the formation 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 MUX8 as an example to introduce the use of Wide MUX.

#### Port Diagram

**Figure 3-8 MUX8 Port Diagram**



### Port Description

**Table 3-20 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 |
| 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 3-21 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:**

```
MUX8 instName (
    .I0(i0),
    .I1(i1),
    .I2(i2),
    .I3(i3),
    .I4(i4),
    .I5(i5),
```

```
        .I6(i6),
        .I7(i7),
        .S0(s0),
        .S1(s1),
        .S2(s2),
        .O(o0)
);
```

**Vhdl Instantiation:**

```
COMPONENT MUX8
        PORT(
            O:OUT std_logic;
              I0:IN std_logic;
                 I1:IN std_logic;
                 I2:IN std_logic;
                 I3:IN std_logic;
                 I4:IN std_logic;
                 I5:IN std_logic;
                 I6:IN std_logic;
                 I7:IN std_logic;
                S0:IN std_logic;
                S1:IN std_logic;
                S2:IN std_logic
        );
  END COMPONENT;
  uut:MUX8
        PORT MAP (
            O=>o0,
            I0=>I0,
            I1=>I1,
            I2=>I2,
            I3=>I3,
            I4=>I4,
            I5=>I5,
            I6=>I6,
            I7=>I7,
```

```
        S0=>S0,
        S1=>S1,
        S2=>S2
    );
```

# 3.3 ALU

### Primitive

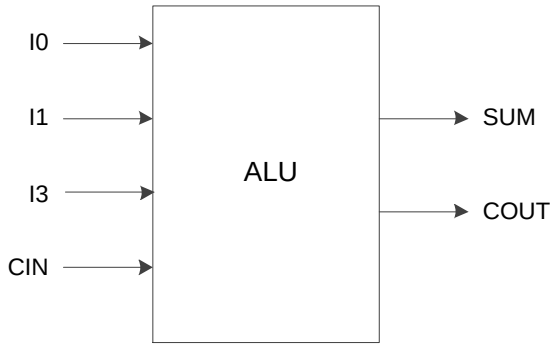
ALU is a 2-input arithmetic logic unit and it can realize the functions of ADD/SUB/ADDSUB, as shown in Table 3-22.

**Table 3-22 ALU Functions**

| Item | Description |
|---|---|
| ADD | Adder |
| SUB | Subtractor |
| ADDSUB | Adder or substractor selecting from I3: 1: Adder, 0: Subtractor. |
| CUP | Up counter |
| CDN | Down counter |
| CUPCDN | Up counter or down counter selecting from I3: 1: Up counter, 0: Down counter |
| GE | Greater than comparator |
| NE | Not equal to comparator |
| LE | Less than comparator |
| MULT | Multiplier |

### Port Diagram

**Figure 3-9 ALU Port Diagram**




### Port Description

**Table 3-23 Port Description**

| Port Name | Input/Output | Description |
|---|---|---|
| I0 | Input | Data Input |
| I1 | Input | Data Input |

| Port Name | Input/Output | Description |
|---|---|---|
| I3 | Input | Data selection, used to select ADDSUB or CUPCDN. |
| CIN | Input | Data Carry Input |
| COUT | Output | Data Carry Output |
| SUM | Output | Data Output |

### Parameter

**Table 3-24 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
    );
```

# 3.4 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 3-25 Primitives Associated With FF**

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

| Primitive | Description |
|---|---|
| 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 3-26 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 |

- 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;
- DFF of two types but same No. can be palced on two FFs in the same CLS, as shown in Table 3-26. All input other than pin input must be in the same net;
- DFF and ALU can be constrainted in the same or different locations of the same CLS;
- 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.

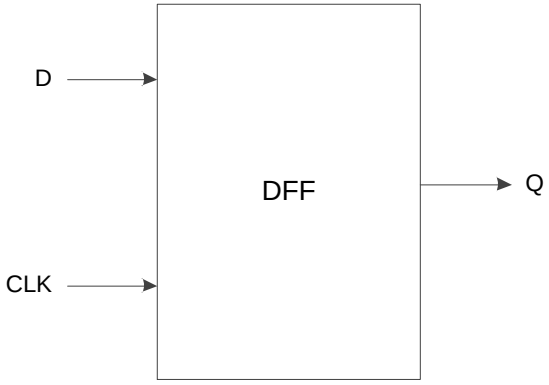
## 3.4.1 DFF

### Primitive

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

## Port Diagram

**Figure 3-10 DFF Port Diagram**




## Port Description

**Table 3-27 Port Description**

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

## Parameter

**Table 3-28 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value of 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
        );
```

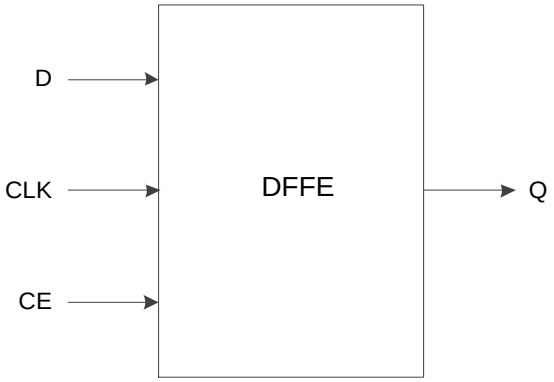
## 3.4.2 DFFE

### Primitive

D Flip-Flop with clock enable (DFFE) is pos-edge triggered.

### Port Diagram

**Figure 3-11 DFFE Port Diagram**




### Port Description

**Table 3-29 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 3-30 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value of 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
     );
```

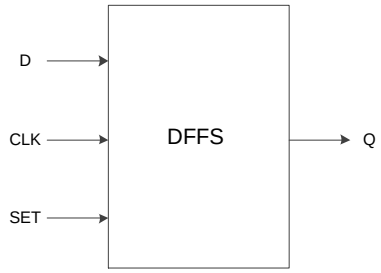
## 3.4.3 DFFS

### Primitive

D Flip-Flop with synchronous set (DFFS) is pos-edge triggered.

## Port Diagram

**Figure 3-12 DFFS Port Diagram**




## Port Description

**Table 3-31 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| D | Input | Data Input |
| CLK | Input | Clock input |
| SET | Input | Synchronous set, active-high. |
| Q | Output | Data Output |

## Parameter

**Table 3-32 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value of 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
    );
```

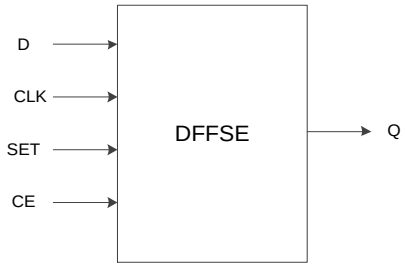
## 3.4.4 DFFSE

### Primitive

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

### Port Diagram

**Figure 3-13 DFFSE Port Diagram**




### Port Description

**Table 3-33 Port Description**

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

### Parameter

**Table 3-34 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value of 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
        );
COMPONENT;
uut:DFFSE
    GENERIC MAP(INIT=>'1')
    PORT MAP (
          Q=>Q,
          D=>D,
          CLK=>CLK,
          SET=>SET,
          CE=>CE
     );
```

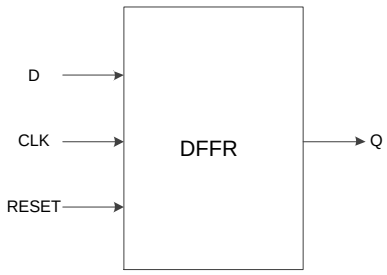
## 3.4.5 DFFR

### Primitive

D Flip-Flop with synchronous reset (DFFR) is pos-edge triggered.

### Port Diagram

**Figure 3-14 DFFR Port Diagram**




### Port Description

**Table 3-35 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| D | Input | Data Input |
| CLK | Input | Clock input |
| RESET | Input | Synchronous reset, active-high |
| Q | Output | Data Output |

### Parameter

**Table 3-36 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value of 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
    );
```

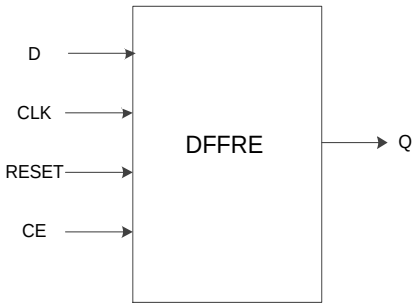
## 3.4.6 DFFRE

### Primitive

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

### Port Diagram

**Figure 3-15 DFFRE Port Diagram**




### Port Description

**Table 3-37 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| D | Input | Data Input |
| CLK | Input | Clock input |
| RESET | Input | Synchronous reset, active-high |
| CE | Input | Clock Enable |

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

### Parameter

**Table 3-38 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value of 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
    );
```

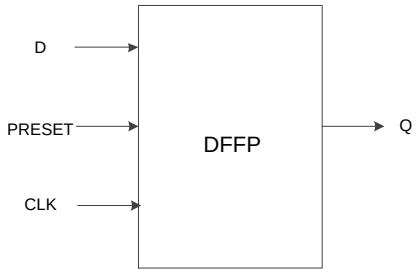
## 3.4.7 DFFP

### Primitive

D Flip-Flop with asynchronous preset (DFFP) is pos-edge triggered.

### Port Diagram

**Figure 3-16 DFFP Port Diagram**




### Port Description

**Table 3-39 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| D | Input | Data Input |
| CLK | Input | Clock input |
| PRESET | Input | Asynchronous preset, active-high |
| Q | Output | Data Output |

### Parameter

**Table 3-40 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value of 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
       );
```

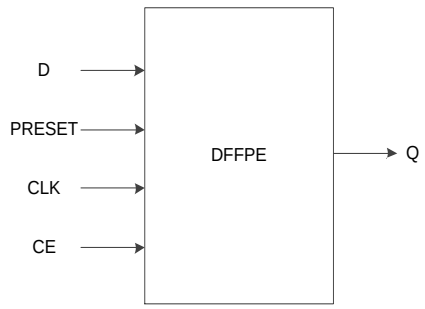
## 3.4.8 DFFPE

### Primitive

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

### Port Diagram

**Figure 3-17 DFFPE Port Diagram**



### Port Description

**Table 3-41 Port Description**

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

### Parameter

**Table 3-42 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value of 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
);
```

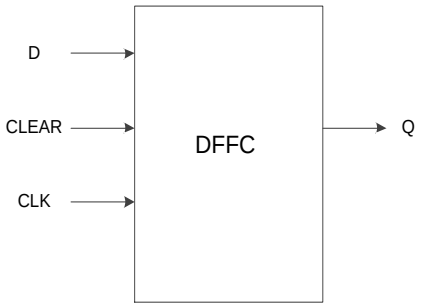
## 3.4.9 DFFC

### Primitive

D Flip-Flop with asynchronous clear (DFFC) is pos-edge triggered.

### Port Diagram

**Figure 3-18 DFFC Blcok Diagram**




### Port Description

**Table 3-43 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| D | Input | Data Input |
| CLK | Input | Clock input |
| CLEAR | Input | Asynchronous clear, active-high |
| Q | Output | Data Output |

### Parameter

**Table 3-44 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value of 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
     );
```

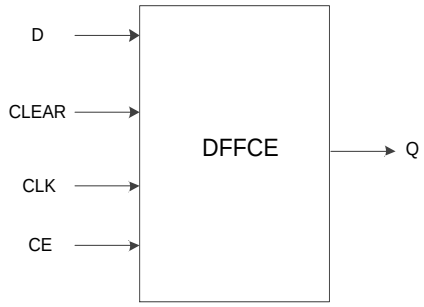
## 3.4.10 DFFCE

**Primitive**

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

## Port Diagram

**Figure 3-19 DFFCE Port Diagram**




## Port Description

**Table 3-45 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| D | Input | Data Input |
| CLK | Input | Clock input |
| CLEAR | Input | Asynchronous clear, active-high. |
| CE | Input | Clock Enable |
| Q | Output | Data Output |

## Parameter

**Table 3-46 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1’b0,1’b1 | 1’b0 | Initial value of 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
    );
```

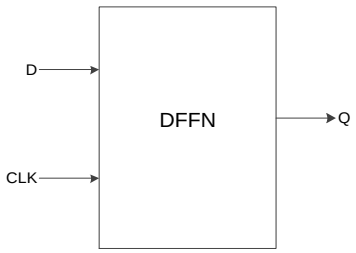
## 3.4.11 DFFN

### Primitive

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

### Port Diagram

**Figure 3-20 DFFN Port Diagram**




### Port Description

**Table 3-47 Port Description**

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

### Parameter

**Table 3-48 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value of 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
       );
```

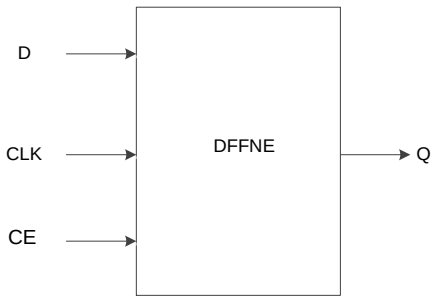
## 3.4.12 DFFNE

### Primitive

D Flip-Flop with clock enable (DFFNE) is neg-edge triggered.

## Port Diagram

**Figure 3-21 DFFNE Port Diagram**




## Port Description

**Table 3-49 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 3-50 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value of 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
     );
```

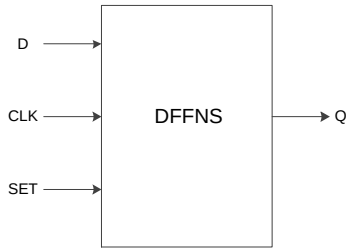
## 3.4.13 DFFNS

### Primitive

D Flip-Flop with synchronous set (DFFNS) is neg-edge triggered.

### Port Diagram

**Figure 3-22 DFFNS Blcok Diagram**




### Port Description

**Table 3-51 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| D | Input | Data Input |
| CLK | Input | Clock input |
| SET | Input | Synchronous set, active-high. |
| Q | Output | Data Output |

### Parameter

**Table 3-52 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value of 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
    );
```

## 3.4.14 DFFNSE

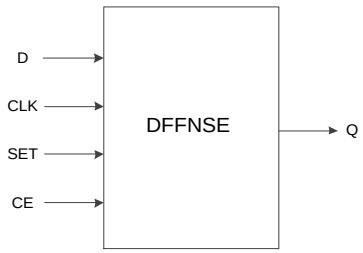
### Primitive

D Flip-Flop with clock enable and synchronous set (DFFNSE) is

neg-edge triggered.

### Port Diagram

**Figure 3-23 DFFNSE Port Diagram**




### Port Description

**Table 3-53 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| D | Input | Data Input |
| CLK | Input | Clock input |
| SET | Input | Synchronous set, active-high. |
| CE | Input | Clock Enable |
| Q | Output | Data Output |

### Parameter

**Table 3-54 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value of 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
    );
```

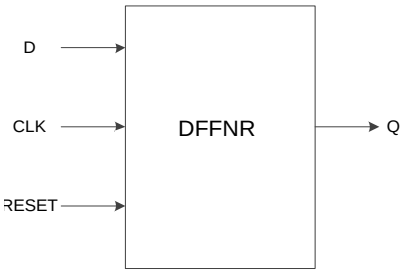
## 3.4.15 DFFNR

### Primitive

D Flip-Flop with synchronous reset (DFFNR) is neg-edge triggered.

### Port Diagram

**Figure 3-24 DFFNR Port Diagram**




### Port Description

**Table 3-55 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| D | Input | Data Input |
| CLK | Input | Clock input |
| RESET | Input | Synchronous reset, active-high. |
| Q | Output | Data Output |

### Parameter

**Table 3-56 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value of 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
    );
```

## 3.4.16 DFFNRE

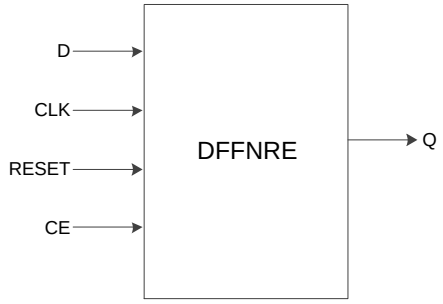
### Primitive

D Flip-Flop with clock enable and synchronous reset (DFFNRE) is

neg-edge triggered.

### Blcok Diagram

**Figure 3-25 DFFNRE Blcok Diagram**




### Port Description

**Table 3-57 Port Description**

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

### Parameter

**Table 3-58 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1’b0,1’b1 | 1’b0 | Initial value of 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
    );
```

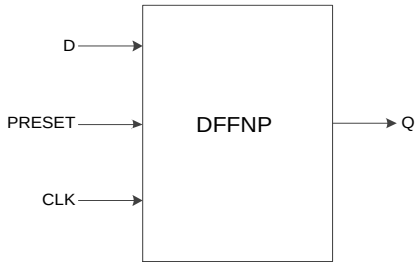
## 3.4.17 DFFNP

### Primitive

D Flip-Flop with asynchronous preset (DFFNP) is neg-edge triggered.

### Port Diagram

**Figure 3-26 DFFNP Port Diagram**




### Port Description

**Table 3-59 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| D | Input | Data Input |
| CLK | Input | Clock input |
| PRESET | Input | Asynchronous preset, active-high |

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

### Parameter

**Table 3-60 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value of 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
      );
```

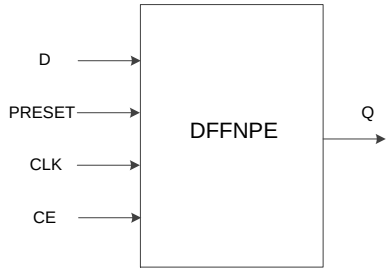
## 3.4.18 DFFNPE

### Primitive

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

### Port Diagram

**Figure 3-27 DFFNPE Port Diagram**




### Port Description

**Table 3-61 Port Description**

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

### Parameter

**Table 3-62 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value of 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
       );
```

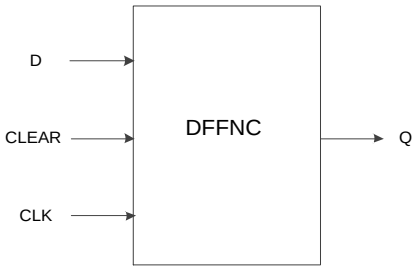
## 3.4.19 DFFNC

### Primitive

D Flip-Flop with asynchronous clear (DFFNC) is neg-edge triggered.

### Port Diagram

**Figure 3-28 DFFNC Port Diagram**



### Port Description

**Table 3-63 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| D | Input | Data Input |
| CLK | Input | Clock input |
| CLEAR | Input | Asynchronous clear, active-high. |
| Q | Output | Data Output |

### Parameter

**Table 3-64 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value of 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
    );
```

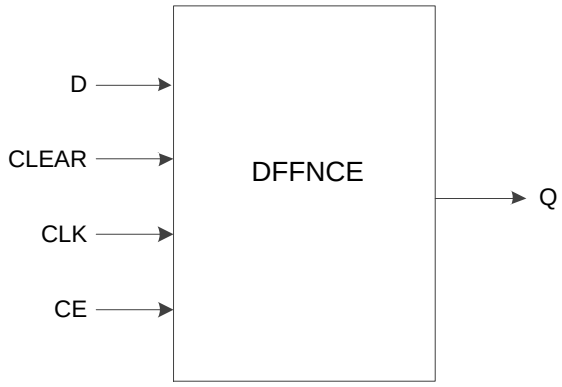
## 3.4.20 DFFNCE

### Primitive

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

### Port Diagram

**Figure 3-29 DFFNCE Port Diagram**




### Port Description

**Table 3-65 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| D | Input | Data Input |
| CLK | Input | Clock input |
| CLEAR | Input | Asynchronous clear, active-high. |
| CE | Input | Clock Enable |
| Q | Output | Data Output |

### Parameter

**Table 3-66 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1’b0,1’b1 | 1’b0 | Initial value of 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
       );
```

# 3.5 LATCH

LATCH is a memory cell circuit and its status can be changed by specified input level. There are twelve primitives related with latch, as shown in Table 3-67.

**Table 3-67 Primitives Related with LATCH**

| Primitive | Description |
|---|---|
| DL | Data Latch |
| DLE | Data latch with enable |
| DLC | Data latch with asynchronous clear |

| Primitive | Description |
|---|---|
| DLCE | Data latch with enable and asynchronous clear |
| DLP | Data latch with asynchronous preset |
| DLPE | Data latch with asynchronous preset and enable |
| DLN | Data latch, active-low |
| 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 3-68 Type of LATCH**

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

- 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;
- DL of two types but same No. can be palced on two FFs in the same CLS, as shown in Table 3-68. All input other than pin input must be in the same net;
- DL and ALU can be constrainted in the same or different locations of the same CLS;
- 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.

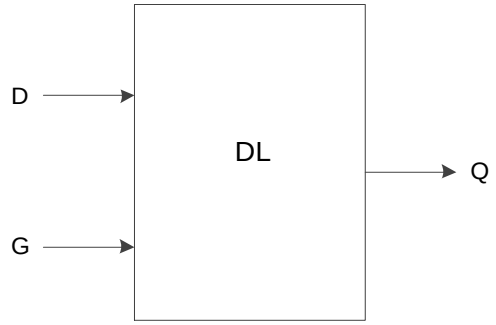
## 3.5.1 DL

### Primitive

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

## Port Diagram

**Figure 3-30 DL Port Diagram**




## Port Description

**Table 3-69 Port Description**

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

## Parameter

**Table 3-70 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value of 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
    );
```

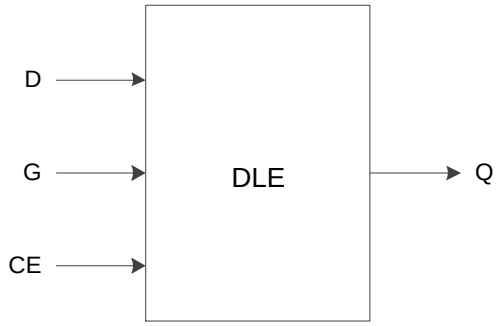
## 3.5.2 DLE

### 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 3-31 DLE Port Diagram**




### Port Description

**Table 3-71 Port Description**

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

### Parameter

**Table 3-72 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value of 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
      );
```

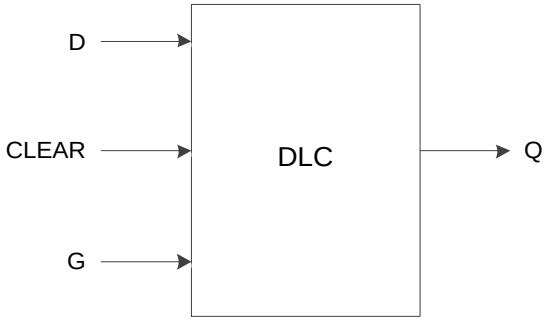
### 3.5.3 DLC

**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 3-32 DLC Port Diagram**




## Port Description

**Table 3-73 Port Description**

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

## Parameter

**Table 3-74 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1’b0,1’b1 | 1’b0 | Initial value of 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
    );
```

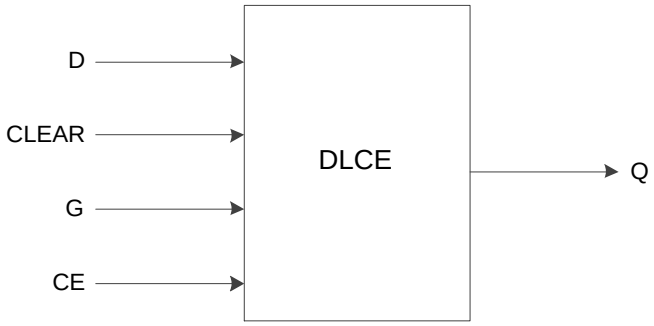
## 3.5.4 DLCE

### Primitive

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

### Port Diagram

**Figure 3-33 DLCE Port Diagram**




### Port Description

**Table 3-75 Port Description**

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

### Parameter

**Table 3-76 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value of 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
      );
```

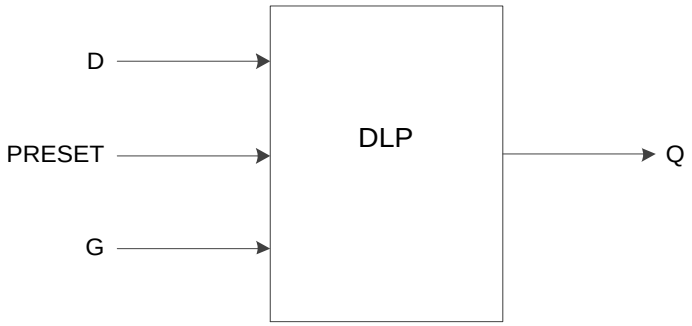
## 3.5.5 DLP

### Primitive

Data Latch with Asynchronous Preset ( DLP ) is a latch with preset function. The control signal G is active-high.

### Port Diagram

**Figure 3-34 DLP Blcok Diagram**




### Port Description

**Table 3-77 Port Description**

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

### Parameter

**Table 3-78 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1’b0,1’b1 | 1’b1 | Initial value of 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
      );
```

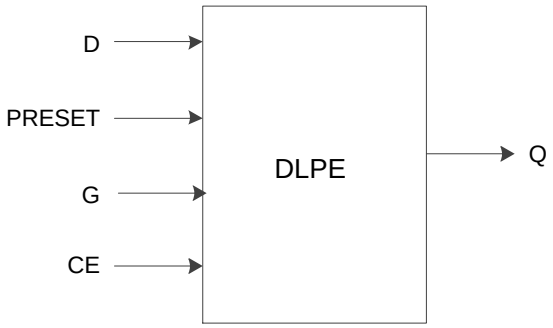
## 3.5.6 DLPE

### Primitive

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

### Port Diagram

**Figure 3-35 DLPE Port Diagram**



### Port Description

**Table 3-79 Port Description**

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

### Parameter

**Table 3-80 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value of 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
);
```

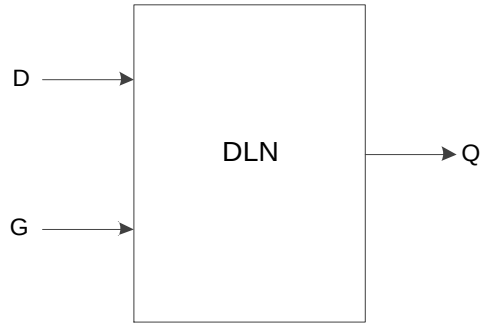
## 3.5.7 DLN

### Primitive

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

### Port Diagram

**Figure 3-36 DLNP Port Diagram**




### Port Description

**Table 3-81 Port Description**

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

### Parameter

**Table 3-82 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value of 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
        );
```

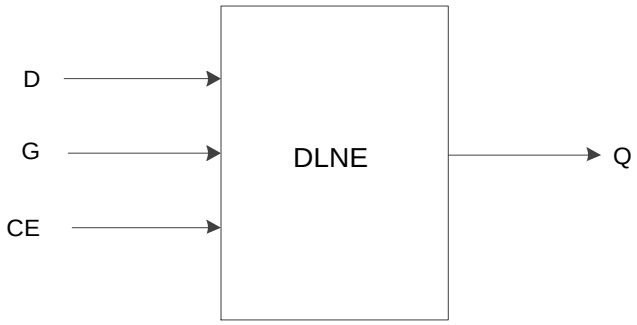
## 3.5.8 DLNE

### 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 3-37 DLNE Port Diagram**



### Port Description

**Table 3-83 Port Description**

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

### Parameter

**Table 3-84 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value of 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
    );
```

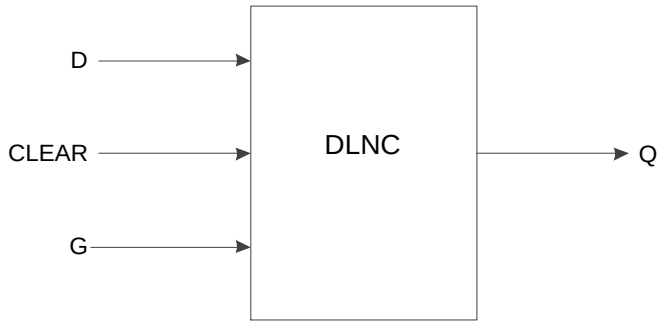
## 3.5.9 DLNC

### 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 3-38 DLNC Port Diagram**




### Port Description

**Table 3-85 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| D | Input | Data Input |
| CLEAR | Input | Asynchronous Clear, active-high. |
| G | Input | Control Signal, active-low. |
| Q | Output | Data Output |

### Parameter

**Table 3-86 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value of 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
        );
```

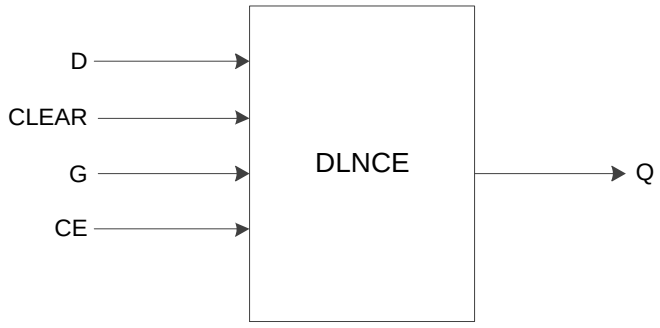
## 3.5.10 DLNCE

### 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 3-39 DLNCE Port Diagram**



### Port Description

**Table 3-87 Port Description**

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

### Parameter

**Table 3-88 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b0 | Initial value of 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
        );
END COMPONENT;
uut:DLNCE
        GENERIC MAP(INIT=>'0'
```

```
    )
PORT MAP (
        Q=>Q,
        D=>D,
        G=>G,
        CE=>CE,
        CLEAR=>CLEAR
);
```

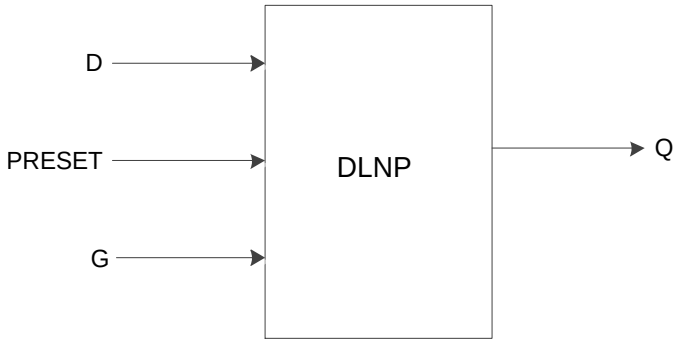
## 3.5.11 DLNP

### 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 3-40 DLNP Port Diagram**




### Port Description

**Table 3-89 Port Description**

| Port Name | I/O | Description |
|---|---|---|
| D | Input | Data Input |
| PRESET | Input | Asynchronous Preset, active-high. |
| G | Input | Control Signal, active-low. |
| Q | Output | Data Output |

### Parameter

**Table 3-90 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1’b0,1’b1 | 1’b1 | Initial value of 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
       );
```

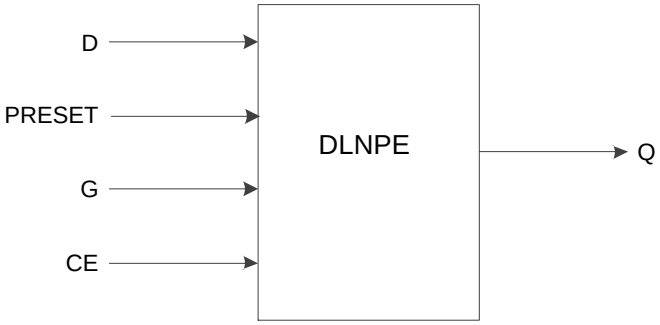
## 3.5.12 DLNPE

**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 3-41 DLNPE Port Diagram**




## Port Description

**Table 3-91 Port Description**

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

## Parameter

**Table 3-92 Parameter**

| Name | Value | Default | Description |
|---|---|---|---|
| INIT | 1'b0,1'b1 | 1'b1 | Initial value of 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
    );
```

# 3.6 SSRAM

For the SSRAM primitives, you can see UG285, Gowin BSRAM & SSRAM User Guide.


GOWIN
PROGRAMMING FOR THE FUTURE