




# Arora V Configurable Function Unit (CFU)

## User Guide

UG303-1.0E, 04/20/2023

**Copyright © 2023 Guangdong Gowin Semiconductor Corporation. All Rights Reserved.**

**GOWIN**  is a trademark of Guangdong Gowin Semiconductor Corporation and is registered in China, the U.S. Patent and Trademark Office, and other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders. No part of this document may be reproduced or transmitted in any form or by any denotes, electronic, mechanical, photocopying, recording or otherwise, without the prior written consent of GOWINSEMI.

#### **Disclaimer**

GOWINSEMI assumes no liability and provides no warranty (either expressed or implied) and is not responsible for any damage incurred to your hardware, software, data, or property resulting from usage of the materials or intellectual property except as outlined in the GOWINSEMI Terms and Conditions of Sale. GOWINSEMI may make changes to this document at any time without prior notice. Anyone relying on this documentation should contact GOWINSEMI for the current documentation and errata.

**Revision History**

Date	Version	Description
04/20/2023	1.0E	Initial version published.

# Contents

<b>Contents .....</b>	<b>i</b>
<b>List of Figures .....</b>	<b>iii</b>
<b>List of Tables .....</b>	<b>iv</b>
<b>1 About This Guide .....</b>	<b>1</b>
1.1 Purpose .....	1
1.2 Related Documents .....	1
1.3 Abbreviations and Terminology .....	1
1.4 Support and Feedback .....	2
<b>2 Configurable Function Unit .....</b>	<b>3</b>
2.1 CLS .....	4
2.1.1 CLS .....	4
2.1.2 REG .....	4
2.2 CRU .....	5
<b>3 CFU Primitives .....</b>	<b>6</b>
3.1 LUT .....	6
3.1.1 LUT1 .....	6
3.1.2 LUT2 .....	7
3.1.3 LUT3 .....	9
3.1.4 LUT4 .....	11
3.1.5 Wide LUT .....	13
3.2 MUX .....	17
3.2.1 MUX2 .....	17
3.2.2 MUX4 .....	18
3.2.3 Wide MUX .....	20
3.3 ALU .....	24
3.4 FF .....	26
3.4.1 DFFSE .....	27
3.4.2 DFFRE .....	28
3.4.3 DFFPE .....	30
3.4.4 DFFCE .....	32
3.5 LATCH .....	34

3.5.1 DLCE .....	34
3.5.2 DLPE .....	36
3.6 SSRAM .....	38

# List of Figures

Figure 2-1 CFU Diagram.....	3
Figure 2-2 Register in CFU .....	4
Figure 3-1 LUT1 Port Diagram.....	6
Figure 3-2 LUT2 Port Diagram.....	8
Figure 3-3 LUT3 Port Diagram.....	9
Figure 3-4 LUT4 Diagram .....	11
Figure 3-5 LUT5 Port Diagram.....	14
Figure 3-6 MUX2 Port Diagram .....	17
Figure 3-7 MUX4 Port Diagram .....	18
Figure 3-8 MUX8 Port Diagram .....	21
Figure 3-9 ALU Port Diagram.....	24
Figure 3-10 DFFSE Port Diagram.....	27
Figure 3-11 DFFRE Port Diagram.....	29
Figure 3-12 DFFPE Port Diagram.....	30
Figure 3-13 DFFCE Port Diagram .....	32
Figure 3-14 DLCE Port Diagram.....	35
Figure 3-15 DLPE Port Diagram .....	36

# List of Tables

Table 1-1 Abbreviations and Terminology .....	1
Table 2-1 Register Description in CFU .....	4
Table 3-1 Port Description.....	6
Table 3-2 Parameter .....	6
Table 3-3 Truth Table.....	7
Table 3-4 Port Description.....	8
Table 3-5 Parameter .....	8
Table 3-6 Truth Table.....	8
Table 3-7 Port Description.....	9
Table 3-8 Parameter .....	10
Table 3-9 Truth Table.....	10
Table 3-10 Port Description.....	11
Table 3-11 Parameter.....	11
Table 3-12 Truth Table.....	12
Table 3-13 Port Description.....	14
Table 3-14 Parameter .....	14
Table 3-15 Truth Table .....	15
Table 3-16 Port Description.....	17
Table 3-17 Truth Table.....	17
Table 3-18 Port Description.....	18
Table 3-19 Truth Table.....	19
Table 3-20 Port Description.....	21
Table 3-21 Truth Table.....	21
Table 3-22 ALU Functions .....	24
Table 3-23 Port Description.....	24
Table 3-24 Parameter .....	25
Table 3-25 Primitives Associated With FF .....	26
Table 3-26 FF Type .....	26
Table 3-27 Port Description.....	27
Table 3-28 Parameter .....	27
Table 3-29 Port Description.....	29
Table 3-30 Parameter .....	29

Table 3-31 Port Description.....	30
Table 3-32 Parameter .....	31
Table 3-33 Port Description.....	32
Table 3-34 Parameter .....	32
Table 3-35 Primitives Associated with LATCH .....	34
Table 3-36 LATCH Type .....	34
Table 3-37 Port Description.....	35
Table 3-38 Parameter .....	35
Table 3-39 Port Description.....	37
Table 3-40 Parameter .....	37



# 1 About This Guide

## 1.1 Purpose

Arora V Configurable Function Unit (CFU) User Guide describes the structure, operating modes, and primitives of CFU of Arora V products.

## 1.2 Related Documents

The latest user guides are available on the GOWINSEMI® Website. You can see related documents at [www.gowinsemi.com](http://www.gowinsemi.com):

- [DS981, GW5AT series of FPGA Products Data Sheet](#)
- [DS1103, GW5A series of FPGA Products Data Sheet](#)
- [DS1104, GW5AST series of FPGA Products Data Sheet](#)
- [SUG100, Gowin Software User Guide](#)
- [UG300, Arora V 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
ALU	Arithmetic Logic Unit
BSRAM	Block Static Random Access Memory
CFU	Configurable Function Unit
CLS	Configurable Logic Section
CRU	Configurable Routing Unit
DFF	D Flip Flop
DL	Data Latch
LUT	Look-up Table
MUX2	Multiplexer 2:1
REG	Register
SSRAM	Shadow Static Random Access Memory

Abbreviations and Terminology	Full Name
ALU	Arithmetic Logic Unit

## 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.

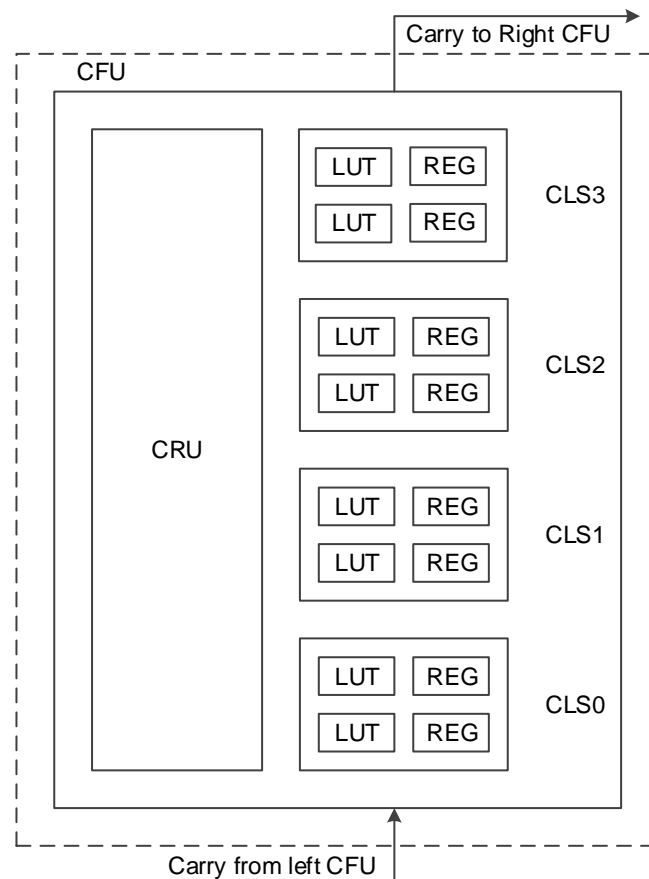
Website: [www.gowinsemi.com](http://www.gowinsemi.com)

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

# 2 Configurable Function Unit

The configurable function unit is the basic unit for FPGA core of GOWINSEMI. As shown in Figure 2-1, each unit consists of four configurable logic sections (CLS) and its configurable routing unit (CRU), and each of the four configurable logic sections contains two 4-input LUTs and two registers. The configurable logic sections in the CFU can be configured as basic logic, ALU, SRAM, and ROM depending on the applications.

**Figure 2-1 CFU Diagram**



**Note!**

GW5AT devices support REG of CLS3; CLK/CE/SR signals of CLS3/CLS2 are from 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. CLS can implement high-order LUT.

- 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 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~CLS3), as shown in Figure 2-2 below.

Figure 2-2 Register in CFU

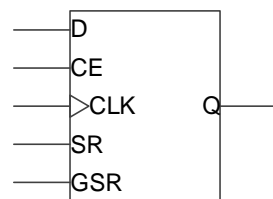


Table 2-1 Register Description in CFU

Signal	I/O	Description
D	I	Data input <sup>[1]</sup>
CE	I	CLK enable, can be configured as active-high or active-low <sup>[2]</sup> .
CLK	I	Clock, can be configured as rising edge or falling edge triggering <sup>[2]</sup> .
SR	I	Local Set/Reset, can be configured as <sup>[2]</sup> : <ul style="list-style-type: none"> <li>● Synchronized reset</li> </ul>

Signal	I/O	Description
		<ul style="list-style-type: none"> <li>● Synchronized set</li> <li>● Asynchronous reset</li> <li>● Asynchronous set</li> <li>● N/A</li> </ul>
GSR <sup>[3], [4]</sup>	I	Global Set/Reset, can be configured as <sup>[4]</sup> : <ul style="list-style-type: none"> <li>● Asynchronous reset</li> <li>● Asynchronous set</li> <li>● N/A</li> </ul>
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; therefore, the register can still be used alone when LUTs are in use.
- [2] Except for the situation that CLS2/CLS3 are in the same net, CE/CLK/SR in CFU are independent.
- [3] In Gowin FPGA products, GSR has its own dedicated network.
- [4] When both SR and GSR are active, 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 inter-CFU, intra-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.

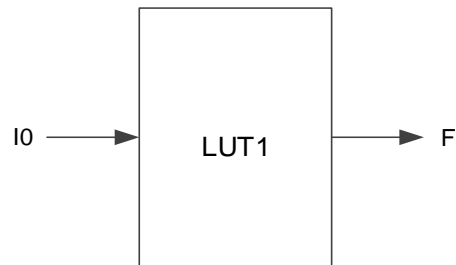
### 3.1.1 LUT1

**Primitive**

LUT1 is usually used as a buffer and an inverter. LUT1 is a 1-input look-up table. After initialization, 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

Name	I/O	Description
I0	Input	Data input signal
F	Output	Data output signal

**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
    );
```

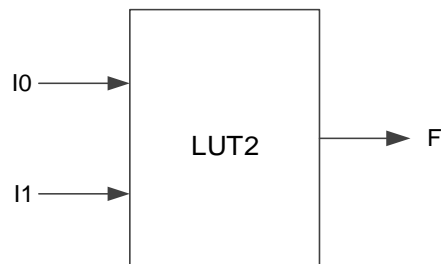
## 3.1.2 LUT2

### 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

Name	I/O	Description
I0	Input	Data input signal
I1	Input	Data input signal
F	Output	Data output signal

## 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:**

```

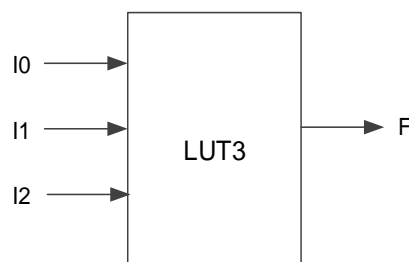
COMPONENT LUT2
  GENERIC (INIT:bit_vector:=X"0");
  PORT(
    F:OUT std_logic;
    I0:IN std_logic;
    I1:IN std_logic
  );
END COMPONENT;

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 signal
I1	Input	Data input signal
I2	Input	Data input signal
F	Output	Data output signal

## 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
  );

```

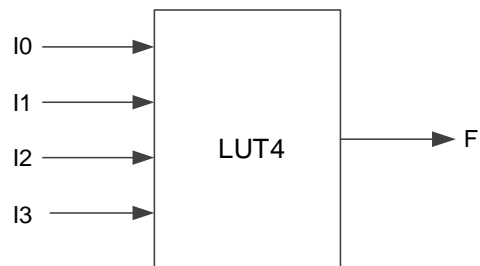
### 3.1.4 LUT4

#### 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

Name	I/O	Description
I0	Input	Data input signal
I1	Input	Data input signal
I2	Input	Data input signal
I3	Input	Data input signal
F	Output	Data output signal

#### 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

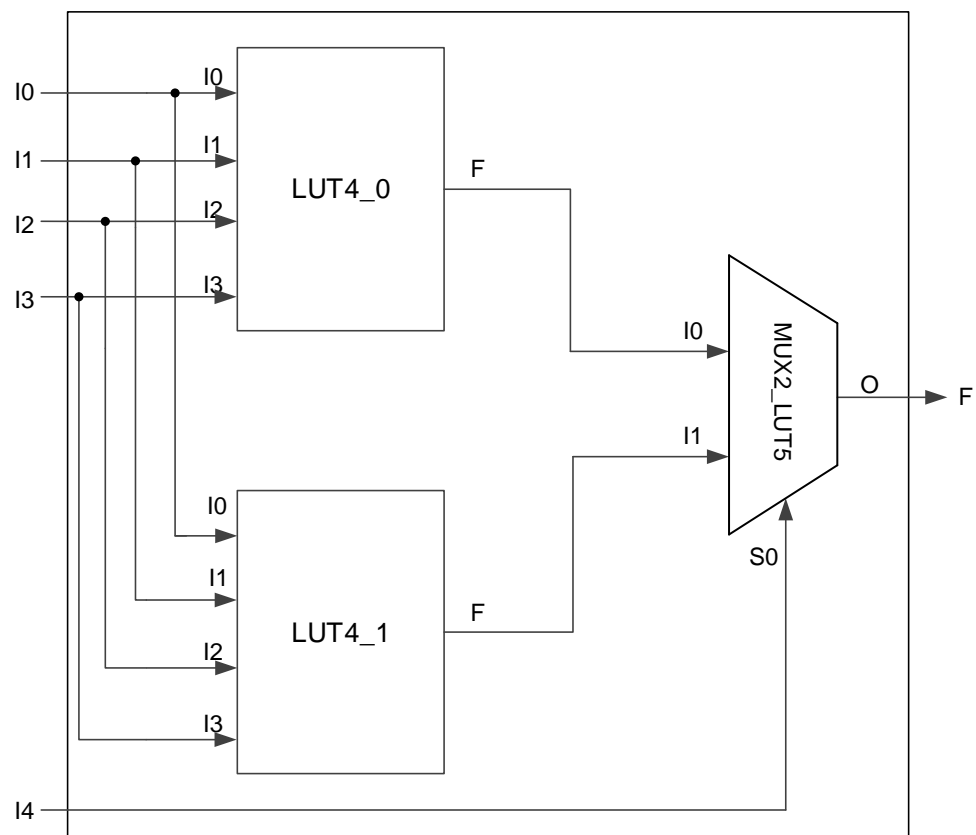
Wide LUT is used for forming high-order input number of LUT by LUT4 and MUX2. Gowin MUX2 supports the high-order formation of MUX2\_LUT5/ MUX2\_LUT6/ MUX2\_LUT7/ MUX2\_LUT8.

The way of the high-order 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

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

## Parameter

Table 3-14 Parameter

Parameter	Value	Default	Description
INIT	32'h00000~32'hffff	32'h00000	Initial value of LUT5

## Truth Table

Table 3-15 Truth 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.

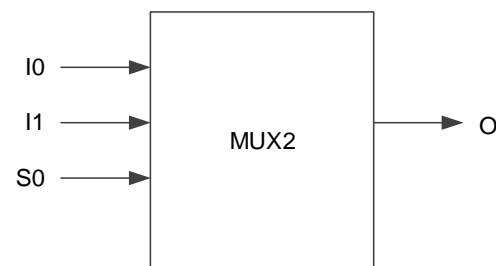
### 3.2.1 MUX2

#### 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

Name	I/O	Description
I0	Input	Data snput
I1	Input	Data Input
S0	Input	Data selected signal
O	Output	Data output signal

#### 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
    );

```

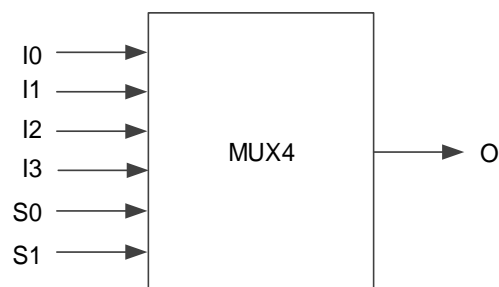
### 3.2.2 MUX4

#### 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

Name	I/O	Description
I0	Input	Data input signal

Name	I/O	Description
I1	Input	Data input signal
I2	Input	Data input signal
I3	Input	Data input signal
S0	Input	Data selected signal
S1	Input	Data selected signal
O	Output	Data output signal

### 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

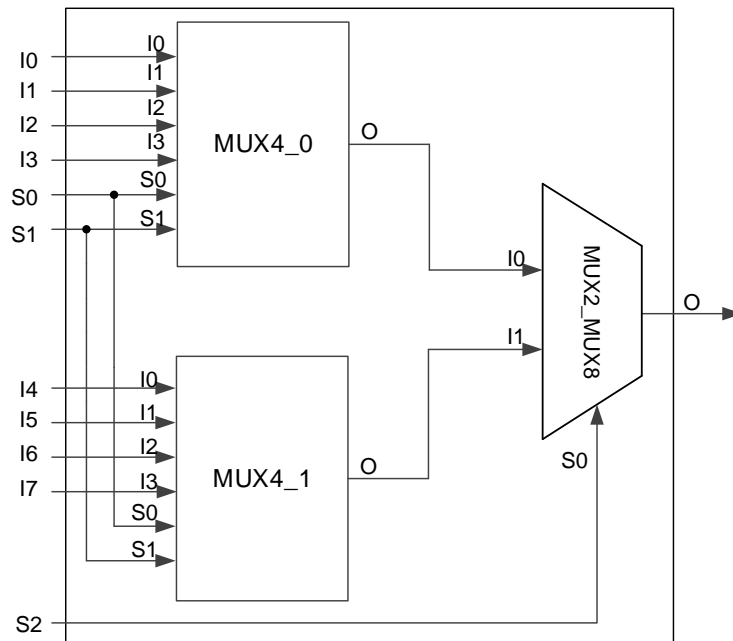
Wide MUX is used for forming high-order input number of MUX by MUX4 and MUX2. Gowin MUX2 support the high-order formation of MUX2\_MUX8/ MUX2\_MUX16/ MUX2\_MUX32.

The way of this 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

Name	I/O	Description
I0	Input	Data input signal
I1	Input	Data input signal
I2	Input	Data input signal
I3	Input	Data input signal
I4	Input	Data input signal
I5	Input	Data input signal
I6	Input	Data input signal
I7	Input	Data input signal
S0	Input	Data selected signal
S1	Input	Data selected signal
S2	Input	Data selected signal
O	Output	Data output signal

## Truth Table

Table 3-21 Truth Table

Input(S2)	Input(S1)	Input(S0)	Output(O)
0	0	0	I0
0	0	1	I1

Input(S2)	Input(S1)	Input(S0)	Output(O)
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

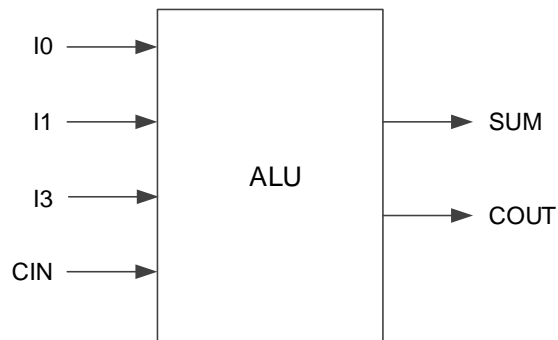
ALU is a 2-input arithmetic logic unit and it can realize the functions of ADD/SUB/ADDSUB, as shown in Table 3-22.

Table 3-22 ALU Functions

Item	Description
ADD	Adder
SUB	Subtractor
ADDSUB	Adder or subtractor selecting from I3: <ul style="list-style-type: none"> <li>● 1: Adder</li> <li>● 0: Subtractor</li> </ul>
CUP	Up counter
CDN	Down counter
CUPCDN	Up counter or down counter selecting from I3: <ul style="list-style-type: none"> <li>● 1: Up counter</li> <li>● 0: Down counter</li> </ul>
GE	Greater than or equal to comparator
NE	Not equal to comparator
LE	Less than or equal to comparator

### Port Diagram

Figure 3-9 ALU Port Diagram



### Note!

The CIN of GW5AT devices can come from the COUT of the previous ALU, also can be from logic or constants.

### Port Description

Table 3-23 Port Description

Name	Input/Output	Description
I0	Input	Data input signal
I1	Input	Data input signal
I3	Input	Data selected signal, used to select ADDSUB or CUPCDN.



Name	Input/Output	Description
CIN	Input	Data carry input signal
COUT	Output	Data carry output signal
SUM	Output	Data output signal

### 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. <ul style="list-style-type: none"> <li>● 0: ADD</li> <li>● 1: SUB</li> <li>● 2: ADDSUB</li> <li>● 3: NE</li> <li>● 4: GE</li> <li>● 5: LE</li> <li>● 6: CUP</li> <li>● 7: CDN</li> <li>● 8: CUPCDN</li> </ul>

### 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 DFFSE, DFFRE, DFFPE, DFFCE. The difference between them is the reset mode.

There are four primitives associated with FF, as shown in Table 3-25.

**Table 3-25 Primitives Associated With FF**

Primitive	Description
DFFSE	D flip-flop with clock enable and synchronous set
DFFRE	D flip-flop with clock enable and synchronous reset
DFFPE	D flip-flop with clock enable and asynchronous preset
DFFCE	D flip-flop with clock enable and asynchronous clear

### Placement Rule

**Table 3-26 FF Type**

No.	Type 1	Type 2
1	DFFSE	DFFRE
2	DFFPE	DFFCE

- 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 placed 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 constrained in the same or different locations of the same CLS.
- DFF and LUT can be constrained in the same or different locations of the same CLS.

**Note!**

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

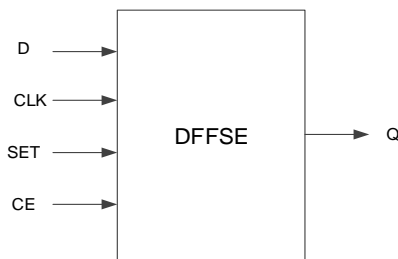
### 3.4.1 DFFSE

#### Primitive

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

#### Port Diagram

Figure 3-10 DFFSE Port Diagram



#### Port Description

Table 3-27 Port Description

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

#### Parameter

Table 3-28 Parameter

Name	Value	Default	Description
INIT	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
    );
END COMPONENT;
uut:DFFSE
    GENERIC MAP(INIT=>'1')
    PORT MAP (
        Q=>Q,
        D=>D,
        CLK=>CLK,
        SET=>SET,
        CE=>CE
    );

```

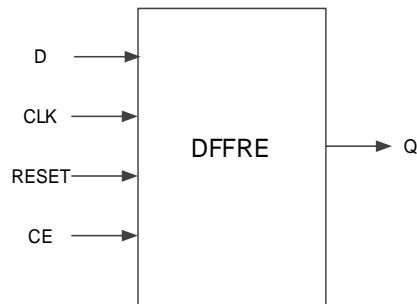
### **3.4.2 DFFRE**

#### **Primitive**

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

## Port Diagram

Figure 3-11 DFFRE Port Diagram



## Port Description

Table 3-29 Port Description

Name	I/O	Description
D	Input	Data input signal
CLK	Input	Clock input signal
RESET	Input	Synchronous reset signal, active-high.
CE	Input	Clock enable signal
Q	Output	Data output signal

## Parameter

Table 3-30 Parameter

Name	Value	Default	Description
INIT	1'b0	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
    );

```

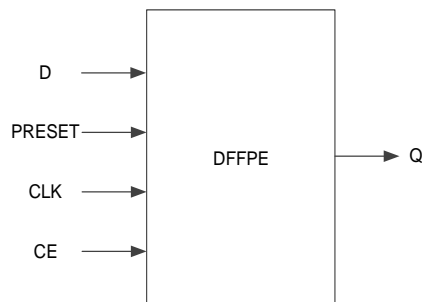
### 3.4.3 DFFPE

#### Primitive

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

#### Port Diagram

Figure 3-12 DFFPE Port Diagram



#### Port Description

Table 3-31 Port Description

Name	I/O	Description
D	Input	Data input signal
CLK	Input	Clock input signal

Name	I/O	Description
PRESET	Input	Asynchronous preset signal, active-high.
CE	Input	Clock enable signal
Q	Output	Data output signal

### Parameter

Table 3-32 Parameter

Name	Value	Default	Description
INIT	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
    );

```

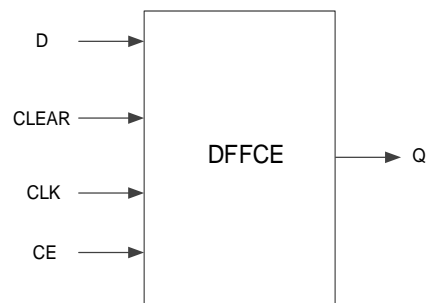
### 3.4.4 DFFCE

#### Primitive

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

#### Port Diagram

Figure 3-13 DFFCE Port Diagram



#### Port Description

Table 3-33 Port Description

Name	I/O	Description
D	Input	Data input signal
CLK	Input	Clock input signal
CLEAR	Input	Asynchronous clear signal, active-high.
CE	Input	Clock enable signal
Q	Output	Data output signal

#### Parameter

Table 3-34 Parameter

Name	Value	Default	Description
INIT	1'b0	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
    );

```

## 3.5 LATCH

LATCH is a memory cell circuit and its status can be changed by specified input level. There are two primitives associated with latch, as shown in Table 3-35.

**Table 3-35 Primitives Associated with LATCH**

Primitive	Description
DLCE	Data latch with enable and asynchronous clear
DLPE	Data latch with asynchronous preset and enable

### Placement Rule

**Table 3-36 LATCH Type**

No.	Type 1	Type 2
1	DLCE	DLPE

- 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 placed on two FFs in the same CLS, as shown in Table 3-36. All input other than pin input must be in the same net.
- DL and ALU can be constrained in the same or different locations of the same CLS.
- DL and LUT can be constrained in the same or different locations of the same CLS.

#### **Note!**

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

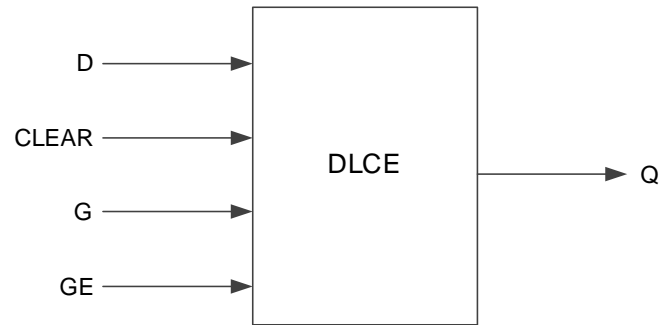
### 3.5.1 DLCE

#### **Primitive**

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

Port Diagram

Figure 3-14 DLCE Port Diagram



Port Description

Table 3-37 Port Description

Name	I/O	Description
D	Input	Data input signal
CLEAR	Input	Asynchronous clear signal, active-high.
G	Input	Data control signal, active-high.
GE	Input	Level enable signal
Q	Output	Data output signal

Parameter

Table 3-38 Parameter

Name	Value	Default	Description
INIT	1'b0	1'b0	Initial value of DLCE

Primitive Instantiation

Verilog Instantiation:

```
DLCE instName (  
    .D(D),  
    .CLEAR(CLEAR),  
    .G(G),  
    .GE(GE),  
    .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;
        GE:IN std_logic;
        CLEAR:IN std_logic
    );
END COMPONENT;

 uut:DLCE
    GENERIC MAP(INIT=>'0')
    PORT MAP (
        Q=>Q,
        D=>D,
        G=>G,
        GE=>GE,
        CLEAR=>CLEAR
    );

```

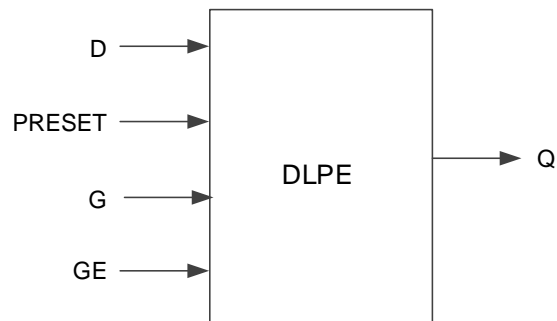
### 3.5.2 DLPE

#### 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 3-15 DLPE Port Diagram



## Port Description

**Table 3-39 Port Description**

Name	I/O	Description
D	Input	Data output signal
PRESET	Input	Asynchronous preset signal, active-high.
G	Input	Data control signal, active-high.
CE	Input	Level enable signal
Q	Output	Data output signal

## Parameter

**Table 3-40 Parameter**

Name	Value	Default	Description
INIT	1'b1	1'b1	Initial value of DLPE

## Primitive Instantiation

### Verilog Instantiation:

```
DLPE instName (
    .D(D),
    .PRESET(PRESET),
    .G(G),
    .GE(GE),
    .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;
        GE:IN std_logic;
        PRESET:IN std_logic
    );
END COMPONENT;
uut:DLPE
```

```
GENERIC MAP(INIT=>'1')  
PORT MAP (  
    Q=>Q,  
    D=>D,  
    G=>G,  
    GE=>GE  
    PRESET =>PRESET  
);
```

## 3.6 SSRAM

For the SSRAM primitives, you can see [UG300, Arora V BSRAM & SSRAM User Guide](#).

