Gowin Primitive

User Guide

SUG283-2.0E, 2019-11-28

Disclaimer

Revision History

Date	Version	Description
04/20/2017	1.0E	Initial version published.
09/19/2017	1.1E	● GW1NR-4, GW1N-6, GW1N-9, GW1NR-9 devices added； ● ELVDS_IOBUF, TLVDS_IOBUF, BUFG, BUFS, OSC, IEM added； ● DSP primitive updated; ● Some ports of ODDR/ODDRC, IDDR_MEM, IDES4_MEM, IDES8_MEM, RAM16S1, RAM16S2、RAM16S4, RAM16SDP1, RAM16SDP2, RAM16SDP4, ROM16 updated; ● Some Attribute of OSC, PLL and DLLDLY updated; ● Some primitive instantiation updated; ● MIPI_IBUF_HS, MIPI_IBUF_LP, MIPI_OBUF, IDES16 and OSER16 updated; ● Some Attribute of CLKDIV updated.
04/12/2018	1.2E	Vhdl primitives instantiation added.
08/08/2018	1.3E	● GW1N-2B, GW1N-4B, GW1NR-4B, GW1N-6ES, GW1N-9ES, GW1NR-9ES, GW1NS-2,GW1NS-2C devices added; ● I3C_IOBUF, DHCEN added; ● User Flash added; ● EMPU added; ● Primitive name updated.
10/26/2018	1.4E	● GW1NZ-1, GW1NSR-2C devices added; ● OSCZ, FLASH96KZ added.
11/15/2018	1.5E	● GW1NSR-2 device added; ● GW1N-6ES, GW1N-9ES, GW1NR-9ES devices removed.
01/26/2019	1.6E	● GW1NS-2 supported by 8 frequency division of CLKDIV added; ● Removed GW1N-1 from the devices supported by TLVDS_TBUF/OBUF.
02/25/2019	1.7E	Removed GW1N-1 from the devices supported by TLVDS_IOBUF.
05/20/2019	1.8E	● GW1N-1S device added; ● MIPI_IBUF added; ● OSCH added; ● SPMI added; ● I3C added; ● Devices supported by OSC updated.
10/20/2019	1.9E	IOB, BSRAM, CLOCK modules updated.
11/28/2019	2.0E	● GSR and INV modules added in Miscellaneous; ● Devices supported updated; ● FLASH64KZ added and FLASH96KZ removed.

Contents

List of Figures

List of Tables

1 IOB

1.1 Buffer/LVDS

Buffer includes normal buffer, emulated LVDS, and true LVDS.

1.1.1 IBUF

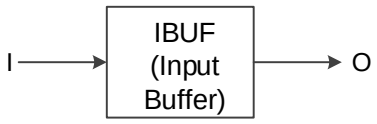
Primitive Introduction

Input Buffer (IBUF)

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Block Diagram

Figure 1-1 IBUF Block Diagram



Port Description

Table 1-1 Port Description

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

```
COMPONENT IBUF
    PORT (
        O:OUT std_logic;
        I:IN std_logic
    );
END COMPONENT;
uut:IBUF
      PORT MAP(
        O=>O,
        I=>I
      );
```

1.1.2 OBUF

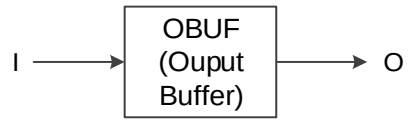
Primitive Introduction

Output Buffer (OBUF)

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Block Diagram

Figure 1-2 OBUF Block Diagram



Port Description

Table 1-2 Port Description

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

```
COMPONENT OBUF
    PORT (
        O:OUT std_logic;
        I:IN std_logic
    );
```

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

1.1.3 TBUF

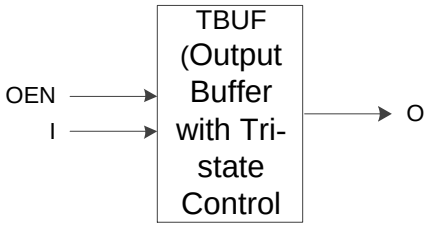
Primitive Introduction

Output Buffer with Tri-state Control (TBUF), active-low.

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Block Diagram

Figure 1-3 TBUF Block Diagram



Port Description

Table 1-3 Port Description

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

Primitive Introduction

Verilog Instantiation:

```
TBUF uut(
    .O(O),
    .I(I),
    .OEN(OEN)
);
```

Vhdl Instantiation:

```
COMPONENT TBUF
      PORT (
            O:OUT std_logic;
            I:IN std_logic;
              OEN:IN std_logic
```

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

1.1.4 IOBUF

Primitive Introduction

Bi-Directional Buffer (IOBUF) is used as an input buffer when OEN is high and is used as an output buffer when ONE is low.

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Block Diagram

Figure 1-4 IOBUF Block Diagram

Port Description

Table 1-4 Port Description

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

Primitive Instantiation

Verilog Instantiation:

```
    IOBUF uut(
        .O(O),
        .IO(IO),
        .I(I),
        .OEN(OEN)
    );
```

Vhdl Instantiation:

```
    COMPONENT IOBUF
          PORT (
         O:OUT std_logic;
```

```
    IO:INOUT std_logic;
            I:IN std_logic;
              OEN:IN std_logic
      );
END COMPONENT;
uut:IOBUF
        PORT MAP(
          O=>O,
       IO=>IO,
          I=>I,
             OEN=> OEN
         );
```

1.1.5 LVDS input buffer

Primitive Introduction

The LVDS includes TLVDS_IBUF and ELVDS_IBUF.

True LVDS Input Buffer (TLVDS_IBUF).

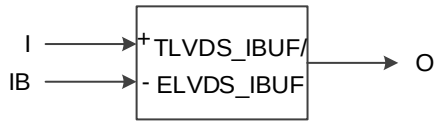
Devices supported: GW1N-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Emulated LVDS Input Buffer (ELVDS_IBUF).

Devices supported: GW1N-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Blockc Diagram

Figure 1-5 TLVDS_IBUF/ELVDS_IBUF Block Diagram



Port Description

Table 1-5 Port Description

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

Primitive Instantiation

Example One

Verilog Instantiation:

```
TLVDS_IBUF uut(
```

```
    .O(O),
        .I(I),
        .IB(IB)
);
```

Vhdl Instantiation:

```
COMPONENT TLVDS_IBUF
    PORT (
        O:OUT std_logic;
        I:IN std_logic;
        IB:IN std_logic
    );
END COMPONENT;
uut:TLVDS_IBUF
        PORT MAP(
        O=>O,
        I=>I,
        IB=> IB
        );
```

Example Two

Verilog Instantiation:

```
ELVDS_IBUF uut(
    .O(O),
        .I(I),
        .IB(IB)
);
```

Vhdl Instantiation:

```
COMPONENT ELVDS_IBUF
    PORT (
        O:OUT std_logic;
        I:IN std_logic;
        IB:IN std_logic
    );
END COMPONENT;
uut:ELVDS_IBUF
        PORT MAP(
        O=>O,
        I=>I,
        IB=> IB
        );
```

1.1.6 LVDS output buffer

Primitive Introduction

LVDS includes TLVDS_OBUF and ELVDS_OBUF.

True LVDS Output Buffer (TLVDS_OBUF) .

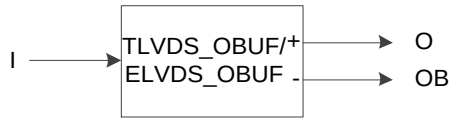
Devices supported: GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Emulated LVDS Output Buffer (ELVDS_OBUF).

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Block Diagram

Figure 1-6 TLVDS_OBUF/ELVDS_OBUF Blcok Diagram



Port Description

Table 1-6 Port Description

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

Primitive Instantiation

Example One

Verilog Instantiation:

```
TLVDS_IBUF uut(
  .O(O),
  .OB(OB),
  .I(I)
);
```

Vhdl Instantiation:

```
COMPONENT TLVDS_OBUF
    PORT (
        O:OUT std_logic;
          OB:OUT std_logic;
          I:IN std_logic
    );
END COMPONENT;
uut:TLVDS_OBUF
      PORT MAP(
        O=>O,
          OB=>OB,
          I=> I
      );
```

Example Two

Verilog Instantiation:

```
ELVDS_OBUF uut(
  .O(O),
  .OB(OB),
```

```
    .I(I)
);
```

Vhdl Instantiation:

```
COMPONENT ELVDS_OBUF
    PORT (
        O:OUT std_logic;
        OB:OUT std_logic;
        I:IN std_logic
    );
END COMPONENT;
uut:ELVDS_OBUF
        PORT MAP(
        O=>O,
        OB=>OB,
        I=> I
        );
```

1.1.7 LVDS tristate buffer

Primitive Introduction

LVDS tristate buffer includes TLVDS_TBUF and ELVDS_TBUF.

True LVDS Tristate Buffer (TLVDS_TBUF) is active-low.

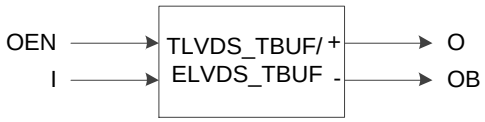
Devies supported: GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Emulated LVDS Tristate Buffer (ELVDS_TBUF), active-low

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Block Diagram

Figure 1-7 TLVDS_TBUF/ELVDS_TBUF Blcok Diagram



Port Description

Table 1-7 Port Description

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

Primitive Instantiation

Example One

Verilog Instantiation:

```
TLVDS_TBUF uut(
 .O(O),
 .OB(OB),
 .I(I),
 .OEN(OEN)
);
```

Vhdl Instantiation:

```
COMPONENT TLVDS_TBUF
    PORT (
        O:OUT std_logic;
          OB:OUT std_logic;
          I:IN std_logic;
          OEN:IN std_logic
    );
END COMPONENT;
uut:TLVDS_TBUF
      PORT MAP(
       O=>O,
          OB=>OB,
          I=> I,
          OEN=>OEN
      );
```

Example Two

Verilog Instantiation:

```
ELVDS_TBUF uut(
 .O(O),
 .OB(OB),
 .I(I),
 .OEN(OEN)
);
```

Vhdl Instantiation:

```
COMPONENT ELVDS_TBUF
    PORT (
        O:OUT std_logic;
          OB:OUT std_logic;
          I:IN std_logic;
          OEN:IN std_logic
    );
END COMPONENT;
uut:ELVDS_TBUF
      PORT MAP(
       O=>O,
          OB=>OB,
          I=> I,
          OEN=>OEN
      );
```

1.1.8 LVDS inout buffer

Primitive Introduction

The LVDS inout buffer includes TLVDS_IOBUF and ELVDS_IOBUF.

True LVDS Bi-Directional Buffer (TLVDS_IOBUF) is used as true differential input buffer when ONE is high and used as true differential output buffer when ONE is low.

Devies supported: GW1N-2, GW1N-2B, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW2A-18, GW2AR-18, GW2A-55.

ELVDS_IOBUF is used as emulated differential input buffer when OEN is high and used as emulated differential output buffer when OEN is low.

Devices supported: GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Block Diagram

Figure 1-8 TLVDS_IOBUF/ELVDS_IOBUF Block Diagram

Port Description

Table 1-8 Port Description

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

Primitive Instantiation

Verilog Instantiation:

```
ELVDS_IOBUF uut(
  .O(O),
  .IO(IO),
  .IOB(IOB),
  .I(I),
  .OEN(OEN)
);
```

Vhdl Instantiation:

```
COMPONENT ELVDS_IOBUF
    PORT (
        O:OUT std_logic;
          IO:INOUT std_logic;
```

```
                IOB:INOUT std_logic;
                I:IN std_logic;
                OEN:IN std_logic
        );
    END COMPONENT;
    uut:ELVDS_IOBUF
            PORT MAP(
             O=>O,
                IO=>IO,
                IOB=>IOB,
                I=> I,
                OEN=>OEN
          );
```

1.1.9 MIPI_IBUF_HS

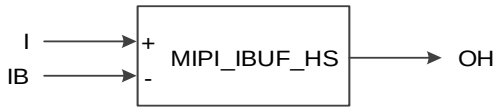
Primitive Introduction

MIPI High Speed Input Buffer (MIPI_IBUF_HS)

Devices supported: GW1N-1S, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9.

Block Diagram

Figure 1-9 MIPI_IBUF_HS Block Diagram



Port Description

Table 1-9 Port Description

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

Primitive Instantiation

Verilog Instantiation:

```
MIPI_IBUF_HS uut(
  .OH(OH),
  .I(I),
  .IB(IB)
);
```

Vhdl Instantiation:

```
COMPONENT MIPI_IBUF_HS
      PORT (
            OH:OUT std_logic;
              I:IN std_logic;
```

```
                IB:IN std_logic
        );
    END COMPONENT;
    uut:   MIPI_IBUF_HS
            PORT MAP(
             OH=>OH,
                I=>I,
                IB=>IB
            );
```

1.1.10 MIPI_IBUF_LP

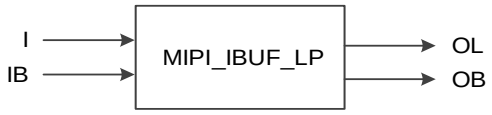
Primitive Introduction

MIPI Low Power Input Buffer (MIPI_IBUF_LP)

Devices supported: GW1N-1S, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9.

Block Diagram

Figure 1-10 MIPI_IBUF_LP Block Diagram



Port Description

Table 1-10 Port Description

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

```
COMPONENT MIPI_IBUF_LP
      PORT (
      OL:OUT std_logic;
      OB:OUT std_logic;
                I:IN std_logic;
                IB:IN std_logic
```

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

1.1.11 MIPI_IBUF

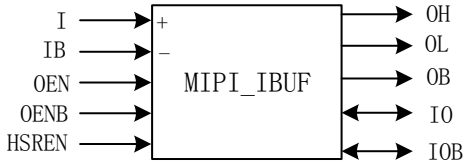
Primitive Introduction

MIPI Input Buffer (MIPI_IBUF) includes HS input mode and LP bi-direction mode, and HS mode supports dynamic resistance configuration.

Devices supported: GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9.

Block Diagram

Figure 1-11 MIPI_IBUF Block Diagram



Port Description

Table 1-11 Port Description

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

Primitive Instantiation

Verilog Instantiation:

```
MIPI_IBUF uut(
```

```
        .OH(OH),
        .OL(OL),
        .OB(OB),
        .IO(IO),
        .IOB(IOB),
        .I(I),
        .IB(IB),
        .OEN(OEN),
        .OENB(OENB),
        HSREN(HSREN)
    );
```

Vhdl Instantiation:

```
    COMPONENT MIPI_IBUF
          PORT (
    OEN, OENB,
        OH:OUT std_logic;
        OL:   OUT std_logic;
        OB:OUT std_logic;
        IO:INOUT std_logic;
                  IOB:INOUT std_logic;
                 I:IN std_logic;
        IB:IN std_logic;
                  OEN:IN std_logic;
        OENB:IN std_logic;
        HSREN:IN std_logic
          );
    END COMPONENT;
    uut:   MIPI_IBUF
              PORT MAP(
                  OH=>OH,
                  OL=>OL,
                  OB=>OB,
                  IO=>IO,
                  IOB=>IOB,
                  I=>I,
                  IB=>IB,
                  OEN=>OEN,
                  OENB=>OENB,
                  HSREN=>HSREN
                );
```

1.1.12 MIPI_OBUF

Primitive Introduction

MIPI Output Buffer (MIPI_OBUF) includes HS mode and LP mode.

MIPI_OBUF is used as (HS) MIPI output buffer when MODESEL is high and used as (LP) MIPI output buffer when MODESEL is low.

Devices supported: GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9.

Block Diagram

Figure 1-12 MIPI_OBUF Block Diagram

Port Description

Table 1-12 Port Description

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

Primitive Instantiation

Verilog Instantiation:

```
MIPI_OBUF uut(
 .O(O),
 .OB(OB),
 .I(I),
 .IB(IB),
 .MODESEL(MODESEL)
);
```

Vhdl Instantiation:

```
COMPONENT MIPI_OBUF
      PORT (
      O:OUT std_logic;
      OB:OUT std_logic;
                I:IN std_logic;
                IB:IN std_logic;
      MODESEL:IN std_logic
      );
END COMPONENT;
uut:   MIPI_OBUF
          PORT MAP(
           O=>O,
    OB=>OB,
              I=>I,
              IB=>IB,
    MDOESEL=>MODESEL
         );
```

1.1.13 I3C_IOBUF

Primitive Introduction

I3C Bi-Directional Buffer (I3C_IOBUF) includes Normal mode and I3C mode.

I3C_IOBUF is used as bi-directional buffer when MODESEL is high and used as normal buffer when MODESEL is low.

Devices supported: GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9.

Block Diagram

Figure 1-13 I3C_IOBUF Block Diagram

Port Description

Table 1-13 Port Description

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

Primitive Instantiation

Verilog Instantiation:

```
I3C_IOBUF uut(
  .O(O),
  .IO(IO),
  .I(I),
  .MODESEL(MODESEL)
);
```

Vhdl Instantiation:

```
COMPONENT I3C_IOBUF
      PORT (
      O:OUT std_logic;
      IO:INOUT std_logic;
                I:IN std_logic;
      MODESEL:IN std_logic
      );
END COMPONENT;
uut:   I3C_IOBUF
          PORT MAP(
           O=>O,
    IO=>IO,
```

```
        I=>I,
MDOESEL=>MODESEL
    );
```

1.2 IOLOGIC

1.2.1 IDDR

Primitive Name

Input Double Data Rate (IDDR)

Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

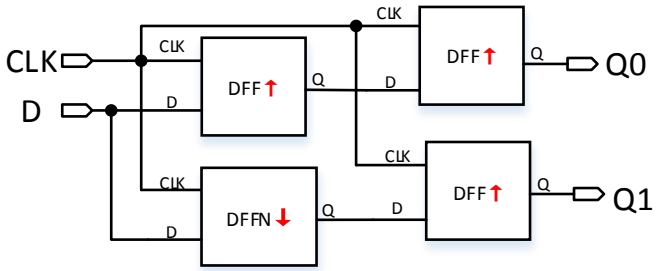
Port Diagram

Figure 1-14 IDDR Port Diagram

Functional Description

Output data is provided to FPGA logic at the same clock edge in IDDR mode. Its logic diagram is as shown in Figure 1-15.

Figure 1-15 IDDR Logic Diagram



Port Description

Table 1-14 Port Description

Port Name	I/O	Description
D	Input	IDDR data input
CLK	Input	Clock input
Q0,Q1	Output	IDDR data output

Parameter

Table 1-15 Parameter Description

Name	Value	Default Value	Description
Q0_INIT	1’b0	1’b0	Initail value of Q0 output
Q1_INIT	1’b0	1’b0	Initail value of Q1 output

Connection Validity Rule

Data input D of IDDR can be directly from IBUF or from DO in IODELAY module.

Primitive Instantiation

Verilog Instantiation:

```
IDDR uut(
        .Q0(Q0),
        .Q1(Q1),
        .D(D),
        .CLK(CLK)
);
defparam uut.Q0_INIT = 1'b0;
defparam uut.Q1_INIT = 1'b0;
```

Vhdl Instantiation:

```
COMPONENT IDDR
    GENERIC (Q0_INIT:bit:='0';
                        Q1_INIT:bit:='0'
          );
    PORT(
      Q0:OUT std_logic;
      Q1:OUT std_logic;
      D:IN std_logic;
        CLK:IN std_logic
        );
END COMPONENT;
uut:IDDR
  GENERIC MAP (Q0_INIT=>'0',
                              Q1_INIT=>'0'
        )
  PORT MAP (
              Q0=>Q0,
              Q1=>Q1,
              D=>D,
              CLK=>CLK
  );
```

1.2.2 ODDR

Primitive Name

Dual Data Rate Output (ODDR)

Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

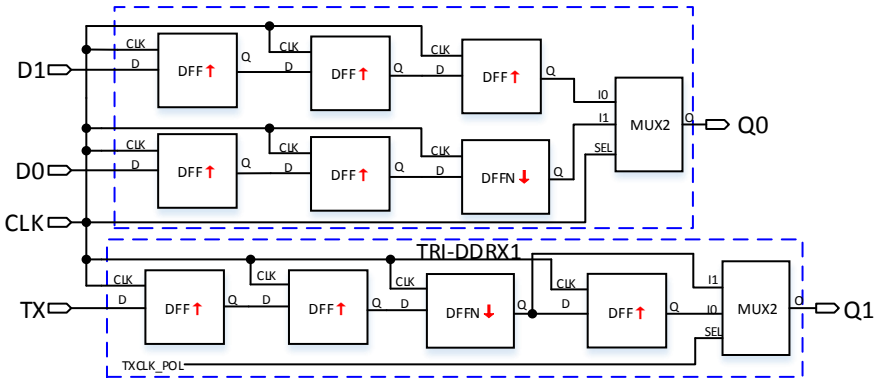
Port Diagram

Figure 1-16 ODDR Port Diagram

Functional Description

ODDR mode is used for transmitting double data rate signals from FPGA devices. Where Q0 is the double rate data output, Q1 is used for the OEN signal of IOBUF/TBUF connected by Q1. Its logic diagram is as shown in Figure 1-17.

Figure 1-17 ODDR Logic Diagram



Port Description

Table 1-16 Port Description

Port Name	I/O	Description
D0,D1	Input	ODDR data input
TX	Input	Q1 generated by TRI-DDRX1
CLK	Input	Clock input
Q0	Output	ODDR data output
Q1	Output	ODDR tri-state enable data output can be connected to the IOBUF/TBUF OEN signal connected by Q0, or suspend

Parameter

Table 1-17 Parameter Description

Name	Value	Default Value	Description
TXCLK_POL	1’b0, 1’b1	1’b0	Q1 output clock polarity control 1'b0:Q1 posedge output; 1'b1:Q1 negedge output
INIT	1’b0	1’b0	Initail value of ODDR output

Connection Validity Rule

- Q0 can directly connect OBUF, or connect input port DI in IODELAY module;
- Q1 shall connect the OEN signal of IOBUF/TBUF connected by Q0, or suspend.

Primitive Instantiation

Verilog Instantiation:

```
ODDR uut(
    .Q0(Q0),
    .Q1(Q1),
    .D0(D0),
    .D1(D1),
 .TX(TX),
    .CLK(CLK)
);
defparam uut.INIT=1’b0;
defparam uut.TXCLK_POL=1’b0;
```

Vhdl Instantiation:

```
COMPONENT ODDR
    GENERIC (CONSTANT INIT:bit:='0';
                      TXCLK_POL:bit:='0'
          );
    PORT(
      Q0:OUT std_logic;
      Q1:OUT std_logic;
      D0:IN std_logic;
                D1:IN std_logic;
                TX:IN std_logic;
      CLK:IN std_logic
   );
END COMPONENT;
uut:ODDR
   GENERIC MAP (INIT=>'0',
                          TXCLK_POL=>'0'
        )
   PORT MAP (
            Q0=>Q0,
            Q1=>Q1,
```

```
        D0=>D0,
        D1=>D1,
        TX=>TX,
        CLK=>CLK
);
```

1.2.3 IDDRC

Primitive Name

Dual Data Rate Input with Asynchronous Clear (IDDRC) is similar to IDDR to realize double data rate input and can be reset asynchronously.

Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

Figure 1-18 IDDRC Port Diagram

Functional Description

Output data is provided to FPGA logic at the same clock edge in IDDRC mode.

Port Description

Table 1-18 Port Description

Port Name	I/O	Description
D	Input	IDDRC data input
CLK	Input	Clock input
CLEAR	Input	Asynchronous reset input, active-high
Q0,Q1	Output	IDDRC data output

Parameter

Table 1-19 Parameter Description

Name	Value	Default Value	Description
Q0_INIT	1'b0	1'b0	Initail value of Q0 output
Q1_INIT	1'b0	1'b0	Initail value of Q1 output

Connection Validity Rule

Data input D of IDDRC can be directly from IBUF or from DO in

IODELAY module.

Primitive Instantiation

Verilog Instantiation:

```
IDDRC uut(
        .Q0(Q0),
        .Q1(Q1),
        .D(D),
        .CLK(CLK),
        .CLEAR(CLEAR)
);
defparam uut.Q0_INIT = 1'b0;
defparam uut.Q1_INIT = 1’b0;
```

Vhdl Instantiation:

```
COMPONENT IDDRC
    GENERIC (Q0_INIT:bit:='0';
                        Q1_INIT:bit:='0'
          );
    PORT(
      Q0:OUT std_logic;
      Q1:OUT std_logic;
      D:IN std_logic;
                CLEAR:IN std_logic;
      CLK:IN std_logic
   );
END COMPONENT;
uut:IDDRC
   GENERIC MAP (Q0_INIT=>'0',
                            Q1_INIT=>'0'
        )
   PORT MAP (
            Q0=>Q0,
            Q1=>Q1,
            D=>D,
            CLEAR=>CLEAR,
            CLK=>CLK
     );
```

1.2.4 ODDRC

Primitive Name

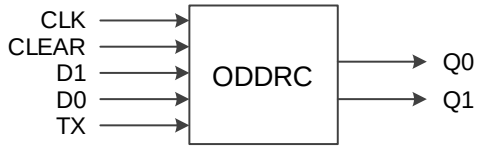
Dual Data Rate Output with Asynchronous Clear (ODDRC) is similar to ODDR to realize double data rate and can be reset asynchronously.

Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

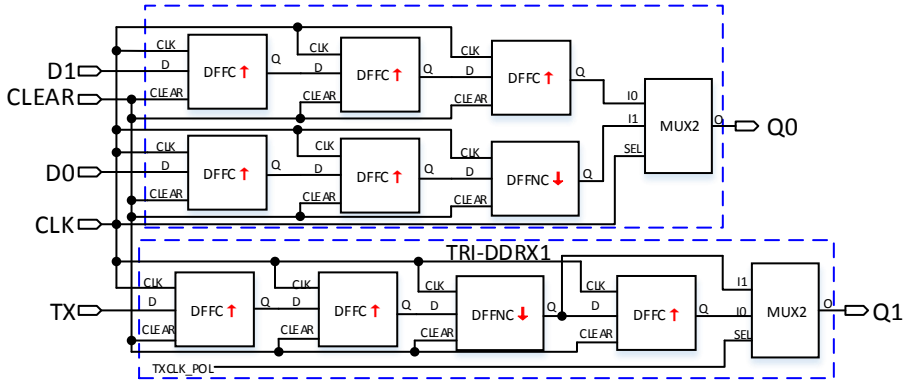
Figure 1-19 ODDRC Port Diagram



Functional Description

ODDRC mode is used for transmitting double data rate signals from FPGA devices. Where Q0 is the double rate data output, Q1 is used for the OEN signal of IOBUF/TBUF connected by Q1. Its logic diagram is as shown in Figure 1-20.

Figure 1-20 ODDRC Logic Diagram



Port Description

Table 1-20 Port Description

Port Name	I/O	Description
D0,D1	Input	ODDRC data input
TX	Input	Input Q1 generated by TRI-DDRX1
CLK	Input	Clock input
CLEAR	Input	Asynchronous reset input, active-high
Q0	Output	ODDRC data output
Q1	Output	ODDR tri-state enable data output can be connected to the IOBUF/TBUF OEN signal connected by Q0, or suspend

Parameter

Table 1-21 Parameter Description

Name	Value	Default Value	Description
TXCLK_POL	1’b0, 1’b1	1’b0	Q1 output clock polarity control 1'b0:Q1 posedge output; 1'b1:Q1 negedge output
INIT	1’b0	1’b0	Initail value of ODDRC output

Connection Validity Rule

- Q0 can directly connect OBUF, or connect input port DI in IODELAY module;
- Q1 shall connect the OEN signal of IOBUF/TBUF connected by Q0, or suspend.

Primitive Instantiation

Verilog Instantiation:

```
ODDRC uut(
 .Q0(Q0),
 .Q1(Q1),
 .D0(D0),
 .D1(D1),
 .TX(TX),
 .CLK(CLK),
 .CLEAR(CLEAR)
);
defparam uut.INIT=1’b0;
defparam uut.TXCLK_POL=1’b0;
```

Vhdl Instantiation:

```
COMPONENT ODDRC
    GENERIC (CONSTANT INIT:bit:='0';
                      TXCLK_POL:bit:='0'
        );
    PORT(
     Q0:OUT std_logic;
     Q1:OUT std_logic;
     D0:IN std_logic;
              D1:IN std_logic;
              TX:IN std_logic;
              CLK:IN std_logic;
     CLEAR:IN std_logic
   );
END COMPONENT;
uut:ODDRC
  GENERIC MAP (INIT=>'0',
                          TXCLK_POL=>'0'
      )
  PORT MAP (
            Q0=>Q0,
            Q1=>Q1,
            D0=>D0,
            D1=>D1,
            TX=>TX,
            CLK=>CLK,
            CLEAR=>CLEAR
    );
```

1.2.5 IDES4

Primitive Name

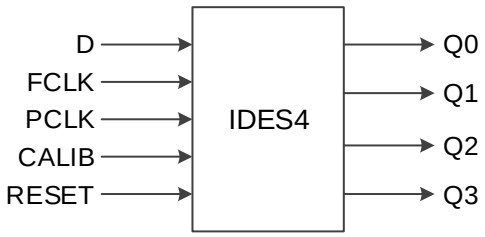
The 1 to 4 Deserializer (IDES4) is a deserializer of 1 bit serial input and 4 bits parallel output.

Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

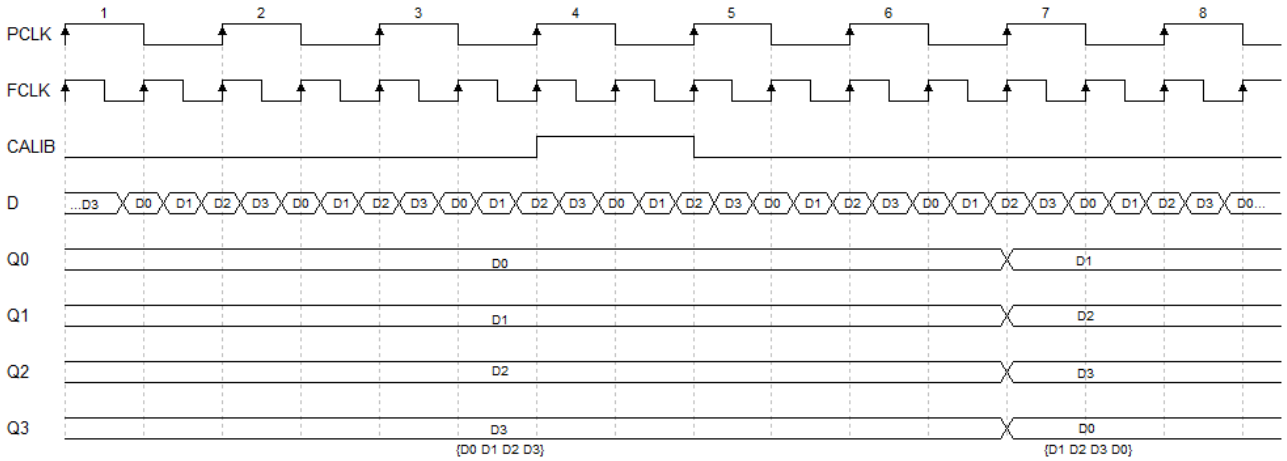
Figure 1-21 IDES4 Port Diagram



Functional Description

IDES4 mode realizes 1:4 serial parallel conversion and output data is provided to FPGA logic at the same clock edge. CALIB is supported to adjust the order of output data. Each pulse data is shifted by one bit. After four shifts, the data output will be the same as the data before the shift. Timing diagram of CALIB is as shown in Figure 1-22.

Figure 1-22 Timing Diagram of CALIB



It should be noted that the pulse width and timing of CALIB signals in the example are for reference only and can be adjusted as needed, with the pulse width greater than or equal to T_{PCLK} .

PCLK is usually obtained by FCLK frequency division.

$$f_{PCLK} = 1/2\, f_{FCLK}$$.

Port Description

Table 1-22 Port Description

Port Name	I/O	Description
D	Input	IDES4 data input
FCLK	Input	High speed clock input
PCLK	Input	Master clock input
CALIB:	Input	CALIB signal, used to adjust the order of output data, active-high
RESET	Input	Asynchronous reset input, active-high
Q3=Q0	Output	IDES4 data output

Parameter

Table 1-23 Parameter Description

Name	Value	Default Value	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Validity Rule

Data input D of IDES4 can be directly from IBUF or from DO in IODELAY module.

Primitive Instantiation

Verilog Instantiation:

```
IDES4 uut(
      .Q0(Q0),
      .Q1(Q1),
      .Q2(Q2),
      .Q3(Q3),
      .D(D),
      .FCLK(FCLK),
      .PCLK(PCLK),
      .CALIB(CALIB),
      .RESET(RESET)
);
defparam uut.GSREN="false";
defparam uut.LSREN ="true";
```

Vhdl Instantiation:

```
COMPONENT IDES4
    GENERIC (GSREN:string:="false";
                        LSREN:string:="true"
          );
    PORT(
      Q0:OUT std_logic;
      Q1:OUT std_logic;
      Q2:OUT std_logic;
```

```
                Q3:OUT std_logic;
    D:IN std_logic;
                FCLK:IN std_logic;
                PCLK:IN std_logic;
                CALIB:IN std_logic;
    RESET:IN std_logic
  );
END COMPONENT;
uut:IDES4
  GENERIC MAP (GSREN=>"false",
                          LSREN=>"true"
      )
  PORT MAP (
            Q0=>Q0,
            Q1=>Q1,
            Q2=>Q2,
            Q3=>Q3,
            D=>D,
            FCLK=>FCLK,
            PCLK=>PCLK,
            CALIB=>CALIB,
            RESET=>RESET
  );
```

1.2.6 IDES8

Primitive Name

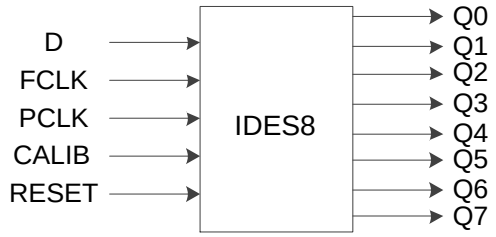
The 1 to 8 Deserializer (IDES8) is a deserializer of 1 bit serial input and 8 bits parallel output.

Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

Figure 1-23 IDES8 Port Diagram



Functional Description

IDES8 mode realizes 1:8 serial parallel conversion and output data is provided to FPGA logic at the same clock edge. CALIB is supported to

adjust the order of output data. Each pulse data is shifted by one bit. After four shifts, the data output will be the same as the data before the shift.

PCLK is usually obtained by FCLK frequency division, .

$f_{PCLK} = 1/4\, f_{FCLK}$.

Port Description

Table 1-24 Port Description

Port Name	I/O	Description
D	Input	IDES8 data input
FCLK	Input	High speed clock input
PCLK	Input	Master clock input
CALIB	Input	CALIB signal input, used to adjust the order of output data, active-high
RESET	Input	Asynchronous reset input, active-high
Q7=Q0	Output	IDES8 data output

Parameter

Table 1-25 Parameter Description

Name	Value	Default Value	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Validity Rule

Data input D of IDES8 can be directly from IBUF or from DO in IODELAY module.

Primitive Instantiation

Verilog Instantiation:

```
IDES8 uut(
        .Q0(Q0),
        .Q1(Q1),
        .Q2(Q2),
        .Q3(Q3),
        .Q4(Q4),
        .Q5(Q5),
        .Q6(Q6),
        .Q7(Q7),
        .D(D),
        .FCLK(FCLK),
        .PCLK(PCLK),
        .CALIB(CALIB),
        .RESET(RESET)
);
defparam uut.GSREN="false";
defparam uut.LSREN ="true";
```

Vhdl Instantiation:

```
COMPONENT IDES8
    GENERIC (GSREN:string:="false";
                        LSREN:string:="true"
        );
    PORT(
      Q0:OUT std_logic;
      Q1:OUT std_logic;
      Q2:OUT std_logic;
                Q3:OUT std_logic;
                Q4:OUT std_logic;
                Q5:OUT std_logic;
                Q6:OUT std_logic;
                Q7:OUT std_logic;
      D:IN std_logic;
                FCLK:IN std_logic;
                PCLK:IN std_logic;
                CALIB:IN std_logic;
      RESET:IN std_logic
    );
END COMPONENT;
uut:IDES8
    GENERIC MAP (GSREN=>"false",
                            LSREN=>"true"
        )
    PORT MAP (
            Q0=>Q0,
            Q1=>Q1,
            Q2=>Q2,
            Q3=>Q3,
            Q4=>Q4,
            Q5=>Q5,
            Q6=>Q6,
            Q7=>Q7,
            D=>D,
            FCLK=>FCLK,
            PCLK=>PCLK,
            CALIB=>CALIB,
            RESET=>RESET
      );
```

1.2.7 IDES10

Primitive Name

The 1 to 10 Deserializer (IDES10) is a deserializer of 1 bit serial input and 10 bits parallel output.

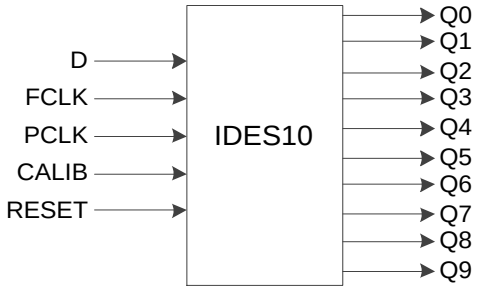
Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C,

GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

Figure 1-24 IDES10 Port Diagram



Functional Description

IDES10 mode realizes 1:10 serial parallel conversion and output data is provided to FPGA logic at the same clock edge. CALIB is supported to adjust the order of output data. Each pulse data is shifted by one bit. After four shifts, the data output will be the same as the data before the shift.

PCLK is usually obtained by FCLK frequency division, .

$f_{PCLK} = 1/5\, f_{FCLK}$.

Port Description

Table 1-26 Port Description

Port Name	I/O	Description
D	Input	IDES10 data input
FCLK	Input	High speed clock input
PCLK	Input	Master clock input
CALIB	Input	CALIB signal, used to adjust the order of output data, active-high
RESET	Input	Asynchronous reset input, active-high
Q9=Q0	Output	IDES10 data output

Parameter

Table 1-27 Parameter Description

Name	Value	Default Value	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Validity Rule

Data input D of IDES10 can be directly from IBUF or from DO in IODELAY module.

Primitive Instantiation

Verilog Instantiation:

```
IDES10 uut(
        .Q0(Q0),
        .Q1(Q1),
        .Q2(Q2),
        .Q3(Q3),
        .Q4(Q4),
        .Q5(Q5),
        .Q6(Q6),
        .Q7(Q7),
        .Q8(Q8),
        .Q9(Q9),
        .D(D),
        .FCLK(FCLK),
        .PCLK(PCLK),
        .CALIB(CALIB),
        .RESET(RESET)
    );
    defparam uut.GSREN="false";
    defparam uut.LSREN ="true";
```

Vhdl Instantiation:

```
COMPONENT IDES10
    GENERIC (GSREN:string:="false";
                        LSREN:string:="true"
        );
    PORT(
      Q0:OUT std_logic;
      Q1:OUT std_logic;
      Q2:OUT std_logic;
                Q3:OUT std_logic;
                Q4:OUT std_logic;
                Q5:OUT std_logic;
                Q6:OUT std_logic;
                Q7:OUT std_logic;
                Q8:OUT std_logic;
                Q9:OUT std_logic;
      D:IN std_logic;
                FCLK:IN std_logic;
                PCLK:IN std_logic;
                CALIB:IN std_logic;
      RESET:IN std_logic
    );
END COMPONENT;
uut:IDES10
    GENERIC MAP (GSREN=>"false",
                            LSREN=>"true"
        )
    PORT MAP (
```

```
        Q0=>Q0,
        Q1=>Q1,
        Q2=>Q2,
        Q3=>Q3,
        Q4=>Q4,
        Q5=>Q5,
        Q6=>Q6,
        Q7=>Q7,
        Q8=>Q8,
        Q9=>Q9,
        D=>D,
        FCLK=>FCLK,
        PCLK=>PCLK,
        CALIB=>CALIB,
        RESET=>RESET
    );
```

1.2.8 IVIDEO

Primitive Name

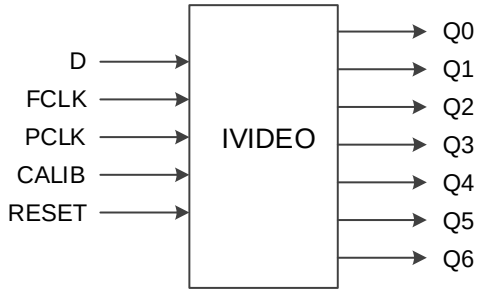
The 1 to 7 Deserializer (IVIDEO) is a deserializer of 1 bit serial input and 7 bits parallel output.

Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

Figure 1-25 IVIDEO Port Diagram



Functional Description

IVIDEO mode realizes 1:7 serial parallel conversion and output data is provided to FPGA logic at the same clock edge. CALIB is supported to adjust the order of output data. Each pulse data is shifted by one bit. After four shifts, the data output will be the same as the data before the shift.

PCLK is usually obtained by FCLK frequency division.

$$f_{PCLK} = 1/3.5\, f_{FCLK}$$.

Port Description

Table 1-28 Port Description

Port Name	I/O	Description
D	Input	IVIDEO data input
FCLK	Input	High speed clock input
PCLK	Input	Master clock input
CALIB	Input	CALIB signal, used to adjust the order of output data, active-high
RESET	Input	Asynchronous reset input, active-high
Q6=Q0	Output	IVIDEO data output

Parameter

Table 1-29 Parameter Description

Name	Value	Default Value	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Validity Rule

Data input D of IVIDEO can be directly from IBUF or from DO in IODELAY module.

Primitive Instantiation

Verilog Instantiation:

```
IVIDEO uut(
        .Q0(Q0),
        .Q1(Q1),
        .Q2(Q2),
        .Q3(Q3),
        .Q4(Q4),
        .Q5(Q5),
        .Q6(Q6),
        .D(D),
        .FCLK(FCLK),
        .PCLK(PCLK),
        .CALIB(CALIB),
        .RESET(RESET)
    );
    defparam uut.GSREN="false";
    defparam uut.LSREN ="true";
```

Vhdl Instantiation:

```
    COMPONENT IVIDEO
        GENERIC (GSREN:string:="false";
                                LSREN:string:="true"
            );
        PORT(
```

```
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        Q2:OUT std_logic;
                    Q3:OUT std_logic;
                    Q4:OUT std_logic;
                    Q5:OUT std_logic;
                    Q6:OUT std_logic;
                     D:IN std_logic;
                    FCLK:IN std_logic;
                    PCLK:IN std_logic;
                    CALIB:IN std_logic;
        RESET:IN std_logic
      );
END COMPONENT;
uut:IVIDEO
  GENERIC MAP (GSREN=>"false",
                              LSREN=>"true"
       )
  PORT MAP (
            Q0=>Q0,
            Q1=>Q1,
            Q2=>Q2,
            Q3=>Q3,
            Q4=>Q4,
            Q5=>Q5,
            Q6=>Q6,
            D=>D,
            FCLK=>FCLK,
            PCLK=>PCLK,
            CALIB=>CALIB,
            RESET=>RESET
       );
```

1.2.9 IDES16

Primitive Name

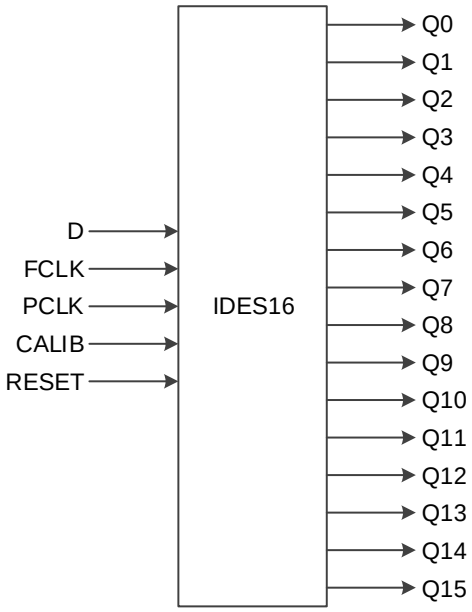
The 1 to 16 Deserializer (IDES16) is a deserializer of 1 bit serial input and 16 bits parallel output.

Devices Supported

Devices supported: GW1N-1S, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9.

Port Diagram

Figure 1-26 IDES16 Port Diagram



Functional Description

IDES16 mode realizes 1:10 serial parallel conversion and output data is provided to FPGA logic at the same clock edge. CALIB is supported to adjust the order of output data. Each pulse data is shifted by one bit. After four shifts, the data output will be the same as the data before the shift.

PCLK is usually obtained by FCLK frequency division.

$f_{PCLK} = 1/8\, f_{FCLK}$.

Port Description

Table 1-30 Port Description

Port Name	I/O	Description
D	Input	IDES16 data input
FCLK	Input	High speed clock input
PCLK	Input	Master clock input
CALIB	Input	CALIB signal, used to adjust the order of output data, active-high
RESET	Input	Asynchronous reset input, active-high
Q15=Q0	Output	IDES16 data output

Parameter

Table 1-31 Parameter Description

Name	Value	Default Value	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Validity Rule

Data input D of IDES16 can be directly from IBUF or from DO in IODELAY module.

Primitive Instantiation

Verilog Instantiation:

```
IDES16 uut(
      .Q0(Q0),
      .Q1(Q1),
      .Q2(Q2),
      .Q3(Q3),
      .Q4(Q4),
      .Q5(Q5),
      .Q6(Q6),
      .Q7(Q7),
      .Q8(Q8),
      .Q9(Q9),
      .Q10(Q10),
      .Q11(Q11),
      .Q12(Q12),
      .Q13(Q13),
      .Q14(Q14),
      .Q15(Q15),
      .D(D),
      .FCLK(FCLK),
      .PCLK(PCLK),
      .CALIB(CALIB),
      .RESET(RESET)
);
defparam uut.GSREN="false";
defparam uut.LSREN ="true";
```

Vhdl Instantiation:

```
COMPONENT IDES16
    GENERIC (GSREN:string:="false";
                        LSREN:string:="true"
        );
    PORT(
      Q0:OUT std_logic;
      Q1:OUT std_logic;
      Q2:OUT std_logic;
                Q3:OUT std_logic;
                Q4:OUT std_logic;
                Q5:OUT std_logic;
                Q6:OUT std_logic;
                Q7:OUT std_logic;
                Q8:OUT std_logic;
                Q9:OUT std_logic;
                Q10:OUT std_logic;
                Q11:OUT std_logic;
```

```
                Q12:OUT std_logic;
                Q13:OUT std_logic;
                Q14:OUT std_logic;
                Q15:OUT std_logic;
    D:IN std_logic;
                FCLK:IN std_logic;
                PCLK:IN std_logic;
                CALIB:IN std_logic;
    RESET:IN std_logic
  );
END COMPONENT;
uut:IDES16
  GENERIC MAP (GSREN=>"false",
                          LSREN=>"true"
      )
  PORT MAP (
            Q0=>Q0,
            Q1=>Q1,
            Q2=>Q2,
            Q3=>Q3,
            Q4=>Q4,
            Q5=>Q5,
            Q6=>Q6,
            Q7=>Q7,
            Q8=>Q8,
            Q9=>Q9,
            Q10=>Q10,
            Q11=>Q11,
            Q12=>Q12,
            Q13=>Q13,
            Q14=>Q14,
            Q15=>Q15,
            D=>D,
            FCLK=>FCLK,
            PCLK=>PCLK,
            CALIB=>CALIB,
            RESET=>RESET
   );
```

1.2.10 OSER4

Primitive Name

The 4 to 1 Serializer (OSER4) is a serializer of 4 bits parallel input and 1 bit parallel output.

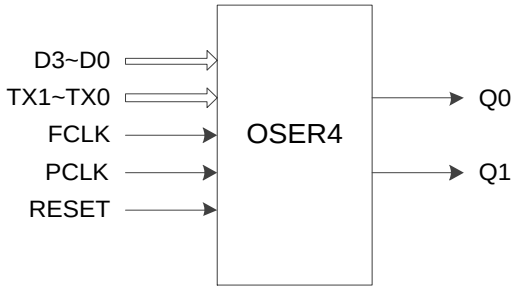
Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9,

GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

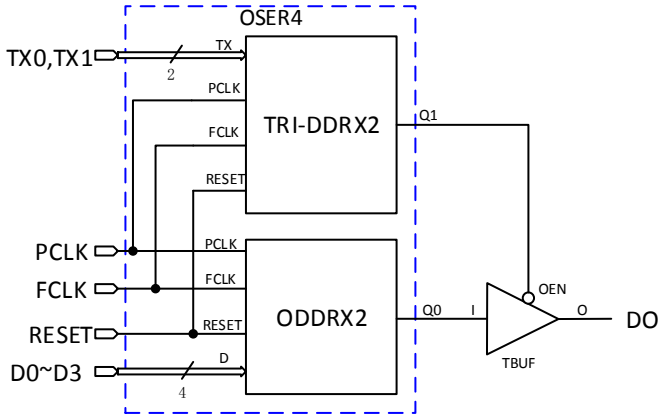
Figure 1-27 OSER4 Port Diagram



Functional Description

OSER4 mode realize 4:1 parallel to serial conversion. Where Q0 is the serial output, Q1 is used for the OEN signal of IOBUF/TBUF connected by Q0. Its logic diagram is as shown in Figure 1-28.

Figure 1-28 OSER4 Logic Diagram



PCLK is usually obtained by FCLK frequency division, $f_{PCLK} = 1/2\, f_{FCLK}$.

Port Description

Table 1-32 Port Description

Port Name	I/O	Description
D3=D0	Input	OSER4 data input
TX1=TX0	Input	Q1 generated by TRI-DDRX2
FCLK	Input	High speed clock input
PCLK	Input	Master clock input
RESET	Input	Asynchronous reset input, active-high
Q0	Output	OSER4 data output
Q1	Output	OSER4 tri-state enable data output can be connected to the IOBUF/TBUF OEN signal connected by Q0, or suspend

Parameter

Table 1-33 Parameter Description

Name	Value	Default Value	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset
TXCLK_POL	1’b0, 1’b1	1’b0	Q1 output clock polarity control 1'b0: data posedge output 1'b1: data negedge output
HWL	"false", "true"	"false"	OSER4 data d_up0/1 timing relationship control "False ": d_up1 is one cycle ahead of d_up0; "True ": d_up1 and d_up0 have the same timing

Connection Validity Rule

- Q0 can directly connect OBUF, or connect input port DI in IODELAY module;
- Q1 shall connect the OEN signal of IOBUF/TBUF connected by Q0, or suspend.

Primitive Instantiation

Verilog Instantiation:

```
OSER4 uut(
        .Q0(Q0),
        .Q1(Q1),
        .D0(D0),
        .D1(D1),
        .D2(D2),
        .D3(D3),
        .TX0(TX0),
        .TX1(TX1),
        .PCLK(PCLK),
        .FCLK(FCLK),
        .RESET(RESET)
    );
    defparam uut.GSREN="false";
    defparam uut.LSREN ="true";
    defparam uut.HWL ="false";
    defparam uut.TXCLK_POL =1'b0;
```

Vhdl Instantiation:

```
    COMPONENT OSER4
        GENERIC (GSREN:string:="false";
                                LSREN:string:="true";
                                HWL:string:="false";
                                TXCLK_POL:bit:='0'
                );
        PORT(
```

```
        Q0:OUT std_logic;
        Q1:OUT std_logic;
        D0:IN std_logic;
                    D1:IN std_logic;
                    D2:IN std_logic;
                    D3:IN std_logic;
                    TX0:IN std_logic;
                    TX1:IN std_logic;
                    FCLK:IN std_logic;
                    PCLK:IN std_logic;
          RESET:IN std_logic
      );
    END COMPONENT;
    uut:OSER4
      GENERIC MAP (GSREN=>"false",
                                LSREN=>"true",
                                HWL=>"false",
                                TXCLK_POL=>'0'
           )
      PORT MAP (
                Q0=>Q0,
                Q1=>Q1,
                D0=>D0,
                D1=>D1,
                D2=>D2,
                D3=>D3,
                TX0=>TX0,
                TX1=>TX1,
                FCLK=>FCLK,
                PCLK=>PCLK,
                RESET=>RESET
        );
```

1.2.11 OSER8

Primitive Name

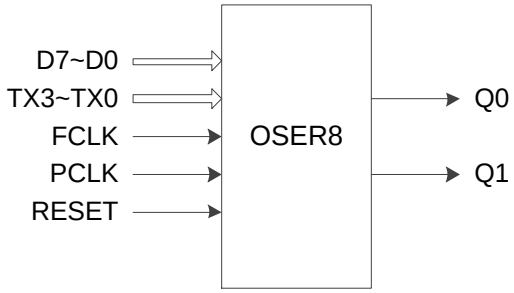
The 8 to 1 Serializer (OSER8) is a serializer of 8 bits parallel input and 1 bit parallel output.

Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

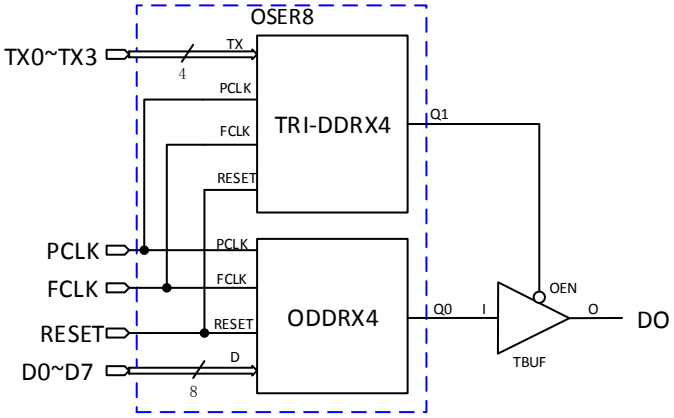
Figure 1-29 OSER8 Port Diagram



Functional Description

OSER8 mode realize 8:1 parallel to serial conversion. Where Q0 is the serial output, Q1 is used for the OEN signal of IOBUF/TBUF connected by Q0. Its logic diagram is as shown in Figure 1-30.

Figure 1-30 OSER8 Logic Diagram



PCLK is usually obtained by FCLK frequency division, $f_{PCLK} = 1/4\, f_{FCLK}$.

Port Description

Table 1-34 Port Description

Port Name	I/O	Description
D7=D0	Input	OSER8 data input
TX3~TX0	Input	Q1 generated by TRI-DDRX4
FCLK	Input	High speed clock input
PCLK	Input	Master clock input
RESET	Input	Asynchronous reset input, active-high
Q0	Output	OSER8 data output
Q1	Output	OSER8 tri-state enable data output can be connected to the IOBUF/TBUF OEN signal connected by Q0, or suspend

Parameter

Table 1-35 Parameter Description

Name	Value	Default Value	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset
TXCLK_POL	1’b0, 1’b1	1’b0	Q1 output clock polarity control 1'b0: data posedge output 1'b1: data negedge output
HWL	"false", "true"	"false"	OSER8 data d_up0/1 timing relationship control "False ": d_up1 is one cycle ahead of d_up0; "True ": d_up1 and d_up0 have the same timing

Connection Validity Rule

- Q0 can directly connect OBUF, or connect input port DI in IODELAY module;
- Q1 shall connect the OEN signal of IOBUF/TBUF connected by Q0, or suspend.

Primitive Instantiation

Verilog Instantiation:

```
OSER8 uut(
        .Q0(Q0),
        .Q1(Q1),
        .D0(D0),
        .D1(D1),
        .D2(D2),
        .D3(D3),
        .D4(D4),
        .D5(D5),
        .D6(D6),
        .D7(D7),
        .TX0(TX0),
        .TX1(TX1),
        .TX2(TX2),
        .TX3(TX3),
        .PCLK(PCLK),
        .FCLK(FCLK),
        .RESET(RESET)
);
defparam uut.GSREN="false";
defparam uut.LSREN ="true";
defparam uut.HWL ="false";
defparam uut.TXCLK_POL =1'b0;
```

Vhdl Instantiation:

```
COMPONENT OSER8
    GENERIC (GSREN:string:="false";
                        LSREN:string:="true";
                        HWL:string:="false";
                        TXCLK_POL:bit:='0'
          );
    PORT(
      Q0:OUT std_logic;
      Q1:OUT std_logic;
      D0:IN std_logic;
                D1:IN std_logic;
                D2:IN std_logic;
                D3:IN std_logic;
                D4:IN std_logic;
                D5:IN std_logic;
                D6:IN std_logic;
                D7:IN std_logic;
                TX0:IN std_logic;
                TX1:IN std_logic;
                TX2:IN std_logic;
                TX3:IN std_logic;
                FCLK:IN std_logic;
                PCLK:IN std_logic;
        RESET:IN std_logic
  );
END COMPONENT;
uut:OSER8
   GENERIC MAP (GSREN=>"false",
                           LSREN=>"true",
                           HWL=>"false",
                           TXCLK_POL=>'0'
         )
   PORT MAP (
              Q0=>Q0,
              Q1=>Q1,
              D0=>D0,
              D1=>D1,
              D2=>D2,
              D3=>D3,
              D4=>D4,
              D5=>D5,
              D6=>D6,
              D7=>D7,
              TX0=>TX0,
              TX1=>TX1,
              TX2=>TX2,
              TX3=>TX3,
              FCLK=>FCLK,
              PCLK=>PCLK,
              RESET=>RESET
```

);

1.2.12 OSER10

Primitive Name

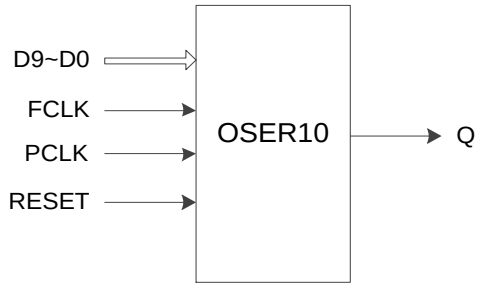
The 10 to 1 Serializer (OSER10) is a serializer of 10 bits parallel input and 1 bit parallel output.

Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

Figure 1-31 OSER10 Port Diagram



Functional Description

OSER10 mode realize 10:1 parallel to serial conversion. PCLK is usually obtained by FCLK frequency division, $f_{PCLK} = 1/5\, f_{FCLK}$.

Port Description

Table 1-36 Port Description

Port Name	I/O	Description
D9~D0	Input	OSER10 data input
FCLK	Input	High speed clock input
PCLK	Input	Master clock input
RESET	Input	Asynchronous reset input, active-high
Q	Output	OSER10 data output

Parameter

Table 1-37 Parameter Description

Name	Value	Default Value	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Validity Rule

Q can directly connect OBUF, or connect input port DI in IODELAY module;

Primitive Instantiation

Verilog Instantiation:

```
OSER10 uut(
      .Q(Q),
      .D0(D0),
      .D1(D1),
      .D2(D2),
      .D3(D3),
      .D4(D4),
      .D5(D5),
      .D6(D6),
      .D7(D7),
      .D8(D8),
      .D9(D9),
      .PCLK(PCLK),
      .FCLK(FCLK),
      .RESET(RESET)
);
defparam uut.GSREN="false";
defparam uut.LSREN ="true";
```

Vhdl Instantiation:

```
COMPONENT OSER10
    GENERIC (GSREN:string:="false";
                        LSREN:string:="true"
        );
    PORT(
     Q:OUT std_logic;
     D0:IN std_logic;
                D1:IN std_logic;
                D2:IN std_logic;
                D3:IN std_logic;
                D4:IN std_logic;
                D5:IN std_logic;
                D6:IN std_logic;
                D7:IN std_logic;
                D8:IN std_logic;
                D9:IN std_logic;
                FCLK:IN std_logic;
                PCLK:IN std_logic;
       RESET:IN std_logic
    );
END COMPONENT;
uut:OSER10
  GENERIC MAP (GSREN=>"false",
                        LSREN=>"true"
```

```
    )
PORT MAP (
        Q=>Q,
        D0=>D0,
        D1=>D1,
        D2=>D2,
        D3=>D3,
        D4=>D4,
        D5=>D5,
        D6=>D6,
        D7=>D7,
        D8=>D8,
        D9=>D9,
        FCLK=>FCLK,
        PCLK=>PCLK,
        RESET=>RESET
);
```

1.2.13 OVIDEO

Primitive Name

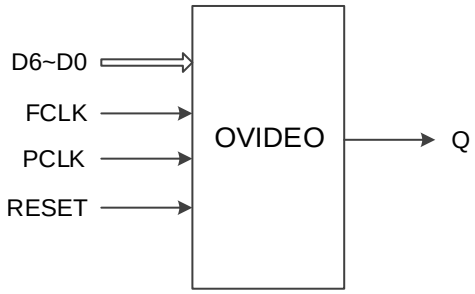
The 7 to 1 Serializer (OVIDEO) is a serializer of 7 bits parallel input and 1 bit parallel output,

Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

Figure 1-32 OVIDEO Port Diagram



Functional Description

OVIDEO mode realizes 7:1 parallel to serial conversion. PCLK is usually obtained by FCLK frequency division, $f_{PCLK} = 1/3.5\, f_{FCLK}$.

Port Description

Table 1-38 Port Description

Port Name	I/O	Description
D6=D0	Input	OVIDEO data input
FCLK	Input	High speed clock input
PCLK	Input	Master clock input
RESET	Input	Asynchronous reset input, active-high
Q	Output	OVIDEO data output

Parameter

Table 1-39 Parameter Description

Name	Value	Default Value	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Validity Rule

Q can directly connect OBUF, or connect input port DI in IODELAY module;

Primitive Instantiation

Verilog Instantiation:

```
OVIDEO uut(
      .Q(Q),
      .D0(D0),
      .D1(D1),
      .D2(D2),
      .D3(D3),
      .D4(D4),
      .D5(D5),
      .D6(D6),
      .PCLK(PCLK),
      .FCLK(FCLK),
      .RESET(RESET)
);
defparam uut.GSREN="false";
defparam uut.LSREN ="true";
```

Vhdl Instantiation:

```
COMPONENT OVIDEO
    GENERIC (GSREN:string:="false";
                    LSREN:string:="true"
        );
    PORT(
      Q:OUT std_logic;
      D0:IN std_logic;
```

```
                    D1:IN std_logic;
                    D2:IN std_logic;
                    D3:IN std_logic;
                    D4:IN std_logic;
                    D5:IN std_logic;
                    D6:IN std_logic;
                    FCLK:IN std_logic;
                    PCLK:IN std_logic;
        RESET:IN std_logic
    );
END COMPONENT;
uut:OVIDEO
    GENERIC MAP (GSREN=>"false",
                            LSREN=>"true"
        )
    PORT MAP (
            Q=>Q,
            D0=>D0,
            D1=>D1,
            D2=>D2,
            D3=>D3,
            D4=>D4,
            D5=>D5,
            D6=>D6,
            FCLK=>FCLK,
            PCLK=>PCLK,
            RESET=>RESET
    );
```

1.2.14 OSER16

Primitive Name

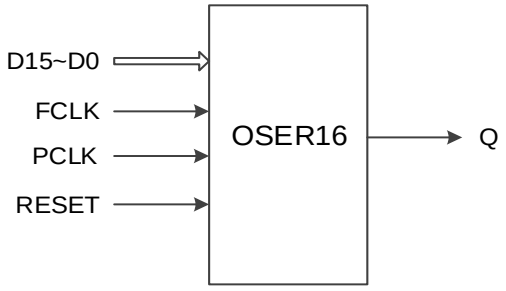
The 16 to 1 Serializer (OSER16) is a serializer of 16 bits parallel input and 1 bit parallel output.

Devices Supported

Devices supported: GW1N-1S, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9.

Port Diagram

Figure 1-33 OSER16 Port Diagram



Functional Description

OSER16 mode realizes 16:1 parallel to serial conversion. PCLK is usually obtained by FCLK frequency division, $f_{PCLK} = 1/8\, f_{FCLK}$.

Port Description

Table 1-40 Port Description

Port Name	I/O	Description
D15=D0	Input	OSER16 data input
FCLK	Input	High speed clock input
PCLK	Input	Master clock input
RESET	Input	Asynchronous reset input, active-high
Q	Output	OSER16 data output

Parameter

Table 1-41 Parameter Description

Name	Value	Default Value	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Validity Rule

Q can directly connect OBUF, or connect input port DI in IODELAY module;

Primitive Instantiation

Verilog Instantiation:

```
OSER16 uut(
    .Q(Q),
    .D0(D0),
    .D1(D1),
    .D2(D2),
    .D3(D3),
    .D4(D4),
    .D5(D5),
    .D6(D6),
    .D7(D7),
    .D8(D8),
    .D9(D9),
    .D10(D10),
    .D11(D11),
    .D12(D12),
    .D13(D13),
    .D14(D14),
    .D15(D15),
    .PCLK(PCLK),
    .FCLK(FCLK),
```

```
        .RESET(RESET)
    );
    defparam uut.GSREN="false";
    defparam uut.LSREN ="true";
```

Vhdl Instantiation:

```
    COMPONENT OSER16
        GENERIC (GSREN:string:="false";
                        LSREN:string:="true"
            );
        PORT(
          Q:OUT std_logic;
          D0:IN std_logic;
                    D1:IN std_logic;
                    D2:IN std_logic;
                    D3:IN std_logic;
                    D4:IN std_logic;
                    D5:IN std_logic;
                    D6:IN std_logic;
                    D7:IN std_logic;
                    D8:IN std_logic;
                    D9:IN std_logic;
                    D10:IN std_logic;
                    D11:IN std_logic;
                    D12:IN std_logic;
                    D13:IN std_logic;
                    D14:IN std_logic;
                    D15:IN std_logic;
                    FCLK:IN std_logic;
                    PCLK:IN std_logic;
          RESET:IN std_logic
        );
    END COMPONENT;
    uut:OSER16
      GENERIC MAP (GSREN=>"false",
                        LSREN=>"true"
          )
      PORT MAP (
                Q=>Q,
                D0=>D0,
                D1=>D1,
                D2=>D2,
                D3=>D3,
                D4=>D4,
                D5=>D5,
                D6=>D6,
                D7=>D7,
                D8=>D8,
                D9=>D9,
                D10=>D10,
                D11=>D11,
```

```
        D12=>D12,
        D13=>D13,
        D14=>D14,
        D15=>D15,
        FCLK=>FCLK,
        PCLK=>PCLK,
        RESET=>RESET
    );
```

1.2.15 IDDR_MEM

Primitive Name

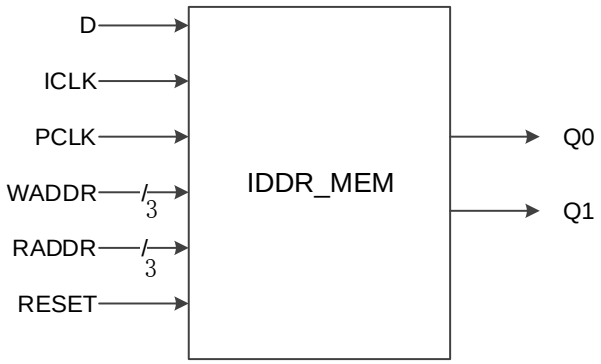
The Input Double Data Rate with Memory (IDDR_MEM) realizes double data rate input with memory.

Devices Supported

Devices supported: GW2A-18, GW2AR-18, and GW2A-55.

Port Diagram

Figure 1-34 IDDR_MEM Port Diagram



Functional Description

IDDR_MEM output data is provided to FPGA logic at the same clock edge. IDDR_MEM needs to be used with DQS. ICLK connects the DQSR90 of DQS output signals and sends data to IDDR_MEM according to the ICLK clock edge. WADDR [2: 0] connects the WPOINT output signal of DQS; RADDR [2: 0] connects the RPOINT output signal of DQS.

The frequency relation between PCLK and ICLK is $f_{PCLK} = f_{ICLK}$.

You can determine the phase relationship between PCLK and ICLK according to DLLSTEP value of DQS.

Port Description

Table 1-42 Port Description

Port Name	I/O	Description
D	Input	IDDR_MEM data input
ICLK	Input	Clock input from DQSR90 in DQS module
PCLK	Input	Master clock input

Port Name	I/O	Description
WADDR[2:0]	Input	Write address from WPOINT in DQS module
RADDR[2:0]	Input	Read address from RPOINT in DQS module
RESET	Input	Asynchronous reset input, active-high
Q1~Q0	Output	IDDR_MEM data input

Parameter

Table 1-43 Parameter Description

Name	Value	Default Value	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Validity Rule

- Data input D of IDDR_MEM can be directly from IBUF or from DO in IODELAY module.
- ICLK needs DQSR90 from DQS module;
- WADDR[2:0] needs WPOINT from DQS module
- RADDR[2:0] needs WPOINT from DQS module;

Primitive Instantiation

Verilog Instantiation:

```
IDDR_MEM iddr_mem_inst(
        .Q0(q0),
        .Q1(q1),
        .D(d),
        .ICLK (iclk),
        .PCLK(pclk),
        .WADDR(waddr[2:0]),
        .RADDR(raddr[2:0]),
        .RESET(reset)
);
defparam uut.GSREN="false";
defparam uut.LSREN ="true";
```

Vhdl Instantiation:

```
COMPONENT IDDR_MEM
    GENERIC (GSREN:string:="false";
                        LSREN:string:="true"
        );
    PORT(
      Q0:OUT std_logic;
                Q1:OUT std_logic;
      D:IN std_logic;
                ICLK:IN std_logic;
                PCLK:IN std_logic;
                WADDR:IN std_logic_vector(2 downto 0);
                RADDR:IN std_logic_vector(2 downto 0);
```

```
        RESET:IN std_logic
    );
END COMPONENT;
uut:IDDR_MEM
    GENERIC MAP (GSREN=>"false",
                          LSREN=>"true"
        )
    PORT MAP (
            Q0=>q0,
            Q1=>q1,
            D=>d,
            ICLK=>iclk,
            PCLK=>pclk,
            WADDR=>waddr,
            RADDR=>raddr,
            RESET=>reset
        );
```

1.2.16 ODDR_MEM

Primitive Name

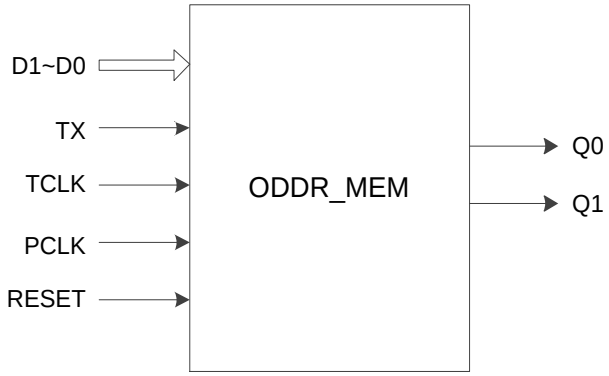
The Dual Data Rate Output with Memory (ODDR_MEM) realizes double data rate output with memory.

Devices Supported

Devices supported: GW2A-18, GW2AR-18, and GW2A-55.

Port Diagram

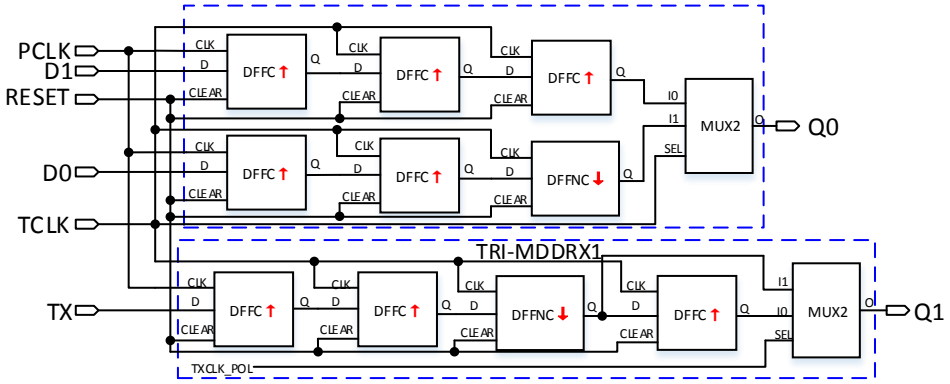
Figure 1-35 ODDR_MEM Port Diagram



Functional Description

ODDR_MEM mode is used for transmitting double data rate signals from FPGA devices. Unlike ODDR, the output double data rate with memory (ODDR_MEM) needs to be used with DQS. TCLK connects the DQSW0 or DQSW270 of DQS output signal, and outputs data from ODDR_MEM according to the TCLK clock edge. Where Q0 is the double rate data output, Q1 is used for the OEN signal of IOBUF/TBUF connected by Q0. Its logic diagram is as shown in Figure 1-36.

Figure 1-36 ODDR_MEM Logic Diagram



The frequency relation between PCLK and TCLK is $f_{PCLK} = f_{TCLK}$.

You can determine the phase relationship between PCLK and ICLK according to DLLSTEP and WSTEP value of DQS.

Port Description

Table 1-44 Port Description

Port Name	I/O	Description
D1=D0	Input	ODDR_MEM data input
TX	Input	Q1 generated by TRI-MDDRX1
TCLK	Input	Clock input from DQSW0 or DQSW270 in DQS module
PCLK	Input	Master clock input
RESET	Input	Asynchronous reset input, active-high
Q0	Output	ODDR_MEM data output
Q1	Output	ODDR_MEM tri-state enable data output can be connected to the IOBUF/TBUF OEN signal connected by Q0, or suspend

Parameter

Table 1-45 Parameter Description

Name	Value	Default Value	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset
TXCLK_POL	1’b0, 1’b1	1’b0	Q1 output clock polarity control 1'b0: data posedge output 1'b1: data negedge output
TCLK_SOURCE	"DQSW","DQSW270"	" DQSW "	TCLK source selection "DQSW" comes from DQSW0 in DQS module "DQSW270" comes from DQSW270 from

Name	Value	Default Value	Description
			DQS module

Connection Validity Rule

- Q0 can directly connect OBUF, or connect input port DI in IODELAY module;
- Q1 shall connect the OEN signal of IOBUF/TBUF connected by Q0, or suspend.
- TCLK needs DQSW0 or DQSW270 from DQS module and configure the corresponding parameters.

Primitive Instantiation

Verilog Instantiation:

```
ODDR_MEM oddr_mem_inst(
        .Q0(q0),
        .Q1(q1),
        .D0(d0),
        .D1(d1),
        .TX(tx),
        .TCLK(tclk),
        .PCLK(pclk),
        .RESET(reset)
);
defparam uut.GSREN="false";
defparam uut.LSREN ="true";
defparam uut.TCLK_SOURCE ="DQSW";
defparam uut.TXCLK_POL=1'b0;
```

Vhdl Instantiation:

```
COMPONENT ODDR_MEM
    GENERIC (GSREN:string:="false";
                        LSREN:string:="true";
                        TXCLK_POL:bit:='0';
                        TCLK_SOURCE:string:="DQSW"
          );
    PORT(
      Q0:OUT std_logic;
                Q1:OUT std_logic;
      D0:IN std_logic;
                D1:IN std_logic;
                TX:IN std_logic;
                TCLK:IN std_logic;
                PCLK:IN std_logic;
                RESET:IN std_logic
   );
END COMPONENT;
uut:ODDR_MEM
   GENERIC MAP (GSREN=>"false",
                        LSREN=>"true",
```

```
                                TXCLK_POL=>'0',
                                TCLK_SOURCE=>"DQSW"
        )
    PORT MAP (
            Q0=>q0,
            Q1=>q1,
            D0=>d0,
            D1=>d1,
            TX=>tx,
            TCLK=>tclk,
            PCLK=>pclk,
            RESET=>reset
     );
```

1.2.17 IDES4_MEM

Primitive Name

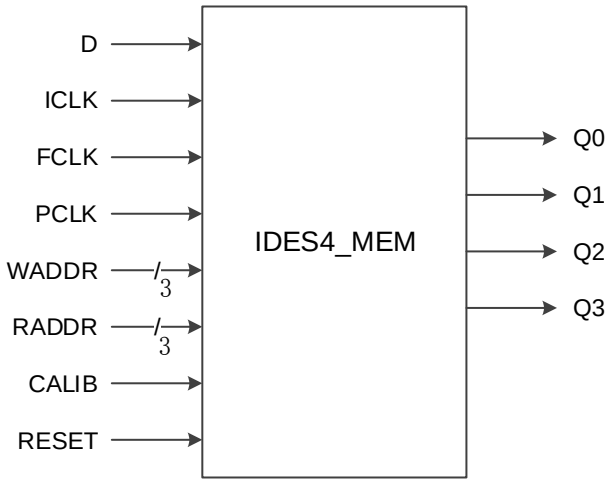
4 to 1 Deserializer with Memory (IDES4_MEM) realizes 1:4 serial convertion with memory.

Devices Supported

Devices supported: GW2A-18, GW2AR-18, and GW2A-55.

Port Diagram

Figure 1-37 IDES4_MEM Port Diagram



Functional Description

IDES4_MEM realizes 1:4 serial parallel conversion and output data is provided to FPGA logic at the same clock edge. CALIB is supported to adjust the order of output data. Each pulse data is shifted by one bit. After four shifts, the data output will be the same as the data before the shift.

The ICLK connects the DQSR90 of DQS output signal and sends data to IDES4_MEM according to the ICLK clock edge. WADDR [2: 0] connects the output signal WPOINT of DQS; RADDR [2: 0] connects the output signal RPOINT of DQS.

The frequency relation between PCLK, FCLK and ICLK is

$f_{PCLK} = 1/2\ f_{FCLK} = 1/2\ f_{ICLK}$.

You can determine the phase realationship between PCLK and ICLK according to DLLSTEP value of DQS.

Port Description

Table 1-46 Port Description

Port Name	I/O	Description
D	Input	IDES4_MEM data input
ICLK	Input	Clock input from DQSR90 in DQS module
FCLK	Input	High speed clock input
PCLK	Input	Master clock input
WADDR[2:0]	Input	Write address from WPOINT in DQS module
RADDR[2:0]	Input	Read address from RPOINT in DQS module
CALIB	Input	CALIB signal, used to adjust the order of output data, active-high
RESET	Input	Asynchronous reset input, active-high
Q3~Q0	Output	IDES4_MEM data output

Parameter

Table 1-47 Parameter Description

Name	Value	Default Value	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Validity Rule

- Data input D of IDES4_MEM can be directly from IBUF or from DO in IODELAY module.
- ICLK needs DQSR90 from DQS module;
- WADDR[2:0] needs WPOINT from DQS module
- RADDR[2:0] needs RPOINT from DQS module;

Primitive Instantiation

Verilog Instantiation:

```
IDES4_MEM ides4_mem_inst(
        .Q0(q0),
        .Q1(q1),
        .Q2(q2),
        .Q3(q3),
        .D(d),
        .ICLK(iclk),
        .FCLK(fclk),
        .PCLK(pclk),
        .WADDR(waddr[2:0]),
        .RADDR(raddr[2:0]),
```

```
        .CALIB(calib),
        .RESET(reset)
    );
    defparam uut.GSREN="false";
    defparam uut.LSREN ="true";
```

Vhdl Instantiation:

```
    COMPONENT IDES4_MEM
        GENERIC (GSREN:string:="false";
                        LSREN:string:="true"
            );
        PORT(
          Q0:OUT std_logic;
                    Q1:OUT std_logic;
                    Q2:OUT std_logic;
                    Q3:OUT std_logic;
          D:IN std_logic;
                    ICLK:IN std_logic;
                    FCLK:IN std_logic;
                    PCLK:IN std_logic;
                    WADDR:IN std_logic_vector(2 downto 0);
                    RADDR:IN std_logic_vector(2 downto 0);
                    CALIB:IN std_logic;
           RESET:IN std_logic
      );
    END COMPONENT;
    uut:IDES4_MEM
      GENERIC MAP (GSREN=>"false",
                            LSREN=>"true"
          )
      PORT MAP (
                Q0=>q0,
                Q1=>q1,
                Q2=>q2,
                Q3=>q3,
                D=>d,
                ICLK=>iclk,
                FCLK=>fclk,
                PCLK=>pclk,
                WADDR=>waddr,
                RADDR=>raddr,
                CALIB=>calib,
                RESET=>reset
        );
```

1.2.18 OSER4_MEM

Primitive Name

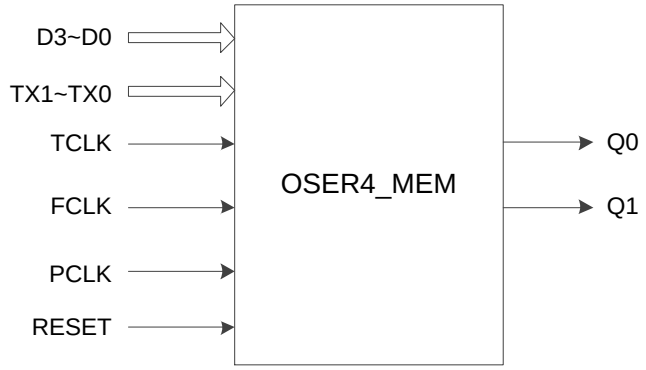
4 to 1 Serializer with Memory (OSER4_MEM) realizes 4:1 parallel serial conversion with memory.

Devices Supported

Devices supported: GW2A-18, GW2AR-18, and GW2A-55.

Port Diagram

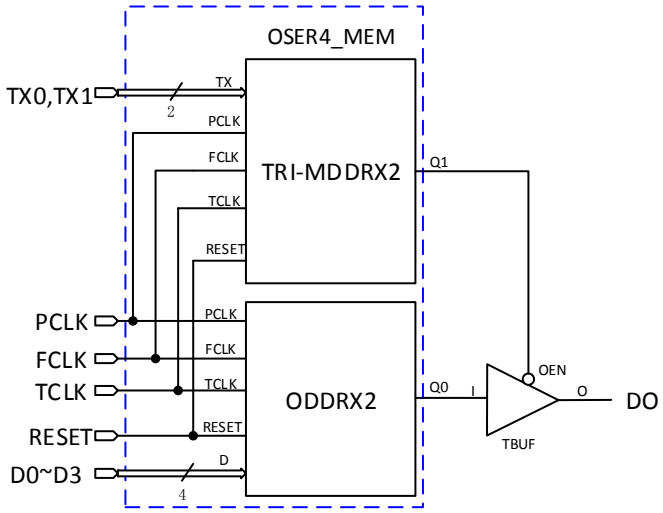
Figure 1-38 OSER4_MEM Port Diagram



Functional Description

OSER4_MEM realizes 4:1 parallel serial conversion. The TCLK connects the output signal DQSW0 or DQSW270 of DQS, and outputs data from the OSER4_MEM according to the TCLK clock edge. Where Q0 is the serial output, Q1 is used for the OEN signal of IOBUF/TBUF connected by Q0. Its logic diagram is as shown in Figure 1-39.

Figure 1-39 OSER4_MEM Logic Diagram



The frequency relation between PCLK, FCLK and TCLK is $f_{PCLK} = 1/2\ f_{FCLK} = 1/2\ f_{TCLK}$.

You can determine the phase realationship between PCLK and ICLK according to the DLLSTEP and WSTEP values of DQS.

Port Description

Table 1-48 Port Description

Port Name	I/O	Description
D3~D0	Input	OSER4_MEM data input
TX1~TX0	Input	Q1 generated by TRI-MDDRX2
TCLK	Input	Clock input from DQSW0 or DQSW270 in DQS module
FCLK	Input	High speed clock input
PCLK	Input	Master clock input
RESET	Input	Asynchronous reset input, active-high
Q0	Output	OSER4_MEM data output
Q1	Output	OSER4_MEM tri-state enable data output can be connected to the IOBUF/TBUF OEN signal connected by Q0, or suspend

Parameter

Table 1-49 Parameter Description

Name	Value	Default Value	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset
TXCLK_POL	1’b0, 1’b1	1’b0	Q1 output clock polarity control 1'b0: data posedge output 1'b1: data negedge output
TCLK_SOURCE	"DQSW","DQSW270"	" DQSW "	TCLK source selection "DQSW" comes from DQSW0 in DQS module "DQSW270" comes from DQSW270 from DQS module
HWL	"false", "true"	"false"	OSER4_MEM data d_up0/1 timing relationship control "False ": d_up1 is one cycle ahead of d_up0; "True ": d_up1 and d_up0 have the same timing

Connection Validity Rule

- Q0 can directly connect OBUF, or connect input port DI in IODELAY module;
- Q1 shall connect the OEN signal of IOBUF/TBUF connected by Q0, or suspend.
- TCLK needs DQSW0 or DQSW270 from DQS module and configure the corresponding parameters.

Primitive Instantiation

Verilog Instantiation:

```
OSER4_MEM oser4_mem_inst(
        .Q0(q0),
        .Q1(q1),
        .D0(d0),
        .D1(d1),
        .D2(d2),
        .D3(d3),
        .TX0(tx0),
        .TX1(tx1),
        .TCLK(tclk),
        .FCLK(fclk),
        .PCLK(pclk),
        .RESET(reset)
);
defparam uut.GSREN="false";
defparam uut.LSREN ="true";
defparam uut.HWL ="false";
defparam uut.TCLK_SOURCE ="DQSW";
defparam uut.TXCLK_POL=1'b0;
```

Vhdl Instantiation:

```
COMPONENT OSER4_MEM
    GENERIC (GSREN:string:="false";
                        LSREN:string:="true";
                        HWL:string:="false";
                        TXCLK_POL:bit:='0';
                        TCLK_SOURCE:string:="DQSW"
          );
    PORT(
      Q0:OUT std_logic;
                Q1:OUT std_logic;
      D0:IN std_logic;
                D1:IN std_logic;
                D2:IN std_logic;
                D3:IN std_logic;
                TX0:IN std_logic;
                TX1:IN std_logic;
                TCLK:IN std_logic;
                FCLK:IN std_logic;
                PCLK:IN std_logic;
                RESET:IN std_logic
   );
END COMPONENT;
uut:OSER4_MEM
   GENERIC MAP (GSREN=>"false",
                        LSREN=>"true",
                        HWL=>"false",
                        TXCLK_POL=>'0',
                        TCLK_SOURCE=>"DQSW"
         )
```

```
PORT MAP (
        Q0=>q0,
        Q1=>q1,
        D0=>d0,
        D1=>d1,
        D2=>d2,
        D3=>d3,
        TX0=>tx0,
        TX1=>tx1,
        TCLK=>tclk,
        FCLK=>fclk,
        PCLK=>pclk,
        RESET=>reset
    );
```

1.2.19 IDES8_MEM

Primitive Name

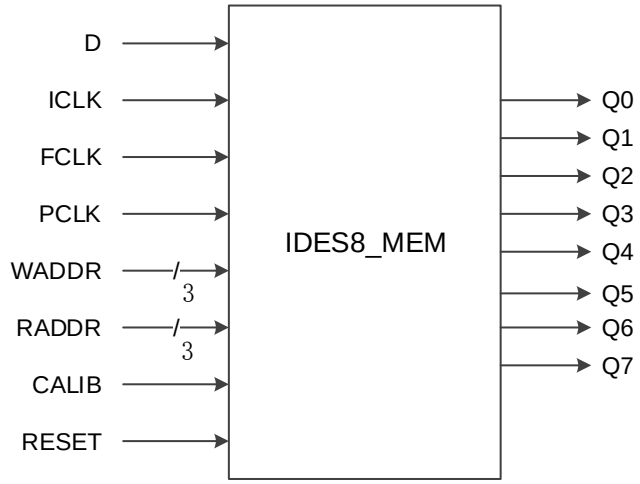
8 to 1 Deserializer with Memory (IDES8_MEM) realizes 8:1 parallel serial conversion with memory.

Devices Supported

Devices supported: GW2A-18, GW2AR-18, and GW2A-55.

Port Diagram

Figure 1-40 IDES8_MEM Port Diagram



Functional Description

IDES8_MEM realizes 1:8 serial parallel conversion and output data is provided to FPGA logic at the same clock edge. CALIB is supported to adjust the order of output data. Each pulse data is shifted by one bit. After four shifts, the data output will be the same as the data before the shift. The ICLK connects the output signal DQSR90 of DQS and sends data to IDES8_MEM according to the ICLK clock edge. WADDR[2:0] connects the output signal WPOINT of DQS; RADDR[2:0] connects Output signal

RPOINT of DQS.

The frequency relation between PCLK, FCLK and ICLK is $f_{PCLK} = 1/4\, f_{FCLK} = 1/4\, f_{ICLK}$.

You can determine the phase realationship between PCLK and ICLK according to DLLSTEP value of DQS.

Port Description

Table 1-50 Port Description

Port Name	I/O	Description
D	Input	IDES8_MEM data input
ICLK	Input	Clock input from DQSR90 in DQS module
FCLK	Input	High speed clock input
PCLK	Input	Master clock input
WADDR[2:0]	Input	Write address from WPOINT in DQS module
RADDR[2:0]	Input	Read address from RPOINT in DQS module
CALIB	Input	CALIB signal, used to adjust the order of output data, active-high
RESET	Input	Asynchronous reset input, active-high
Q7~Q0	Output	IDES8_MEM data output

Parameter

Table 1-51 Parameter Description

Name	Value	Default Value	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Connection Validity Rule

- Data input D of IDES8_MEM can be directly from IBUF or from DO in IODELAY module.
- ICLK needs DQSR90 from DQS module;
- WADDR[2:0] needs WPOINT from DQS module
- RADDR[2:0] needs RPOINT from DQS module;

Primitive Instantiation

Verilog Instantiation:

```
IDES8_MEM ides8_mem_inst(
    .Q0(q0),
    .Q1(q1),
    .Q2(q2),
    .Q3(q3),
    .Q4(q4),
    .Q5(q5),
    .Q6(q6),
    .Q7(q7),
```

```
        .D(d),
        .ICLK(iclk),
        .FCLK(fclk),
        .PCLK(pclk),
        .WADDR(waddr[2:0]),
        .RADDR(raddr[2:0]),
        .CALIB(calib),
        .RESET(reset)
    );
    defparam uut.GSREN="false";
    defparam uut.LSREN ="true";
```

Vhdl Instantiation:

```
    COMPONENT IDES8_MEM
        GENERIC (GSREN:string:="false";
                        LSREN:string:="true"
            );
        PORT(
          Q0:OUT std_logic;
                Q1:OUT std_logic;
                Q2:OUT std_logic;
                Q3:OUT std_logic;
                Q4:OUT std_logic;
                Q5:OUT std_logic;
                Q6:OUT std_logic;
                Q7:OUT std_logic;
          D:IN std_logic;
                ICLK:IN std_logic;
                FCLK:IN std_logic;
                PCLK:IN std_logic;
                WADDR:IN std_logic_vector(2 downto 0);
                RADDR:IN std_logic_vector(2 downto 0);
                CALIB:IN std_logic;
           RESET:IN std_logic
      );
    END COMPONENT;
    uut:IDES8_MEM
      GENERIC MAP (GSREN=>"false",
                        LSREN=>"true"
          )
      PORT MAP (
                Q0=>q0,
                Q1=>q1,
                Q2=>q2,
                Q3=>q3,
                Q4=>q4,
                Q5=>q5,
                Q6=>q6,
                Q7=>q7,
                D=>d,
                ICLK=>iclk,
```

```
        FCLK=>fclk,
        PCLK=>pclk,
        WADDR=>waddr,
        RADDR=>raddr,
        CALIB=>calib,
        RESET=>reset
    );
```

1.2.20 OSER8_MEM

Primitive Name

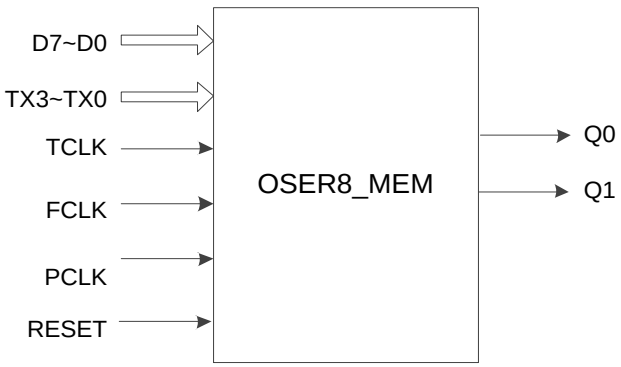
8 to 1 Serializer with Memory (OSER8_MEM) realizes 8:1 parallel serial conversion with memory.

Devices Supported

supports the GW2A-18, GW2AR-18, and GW2A-55 devices.

Port Diagram

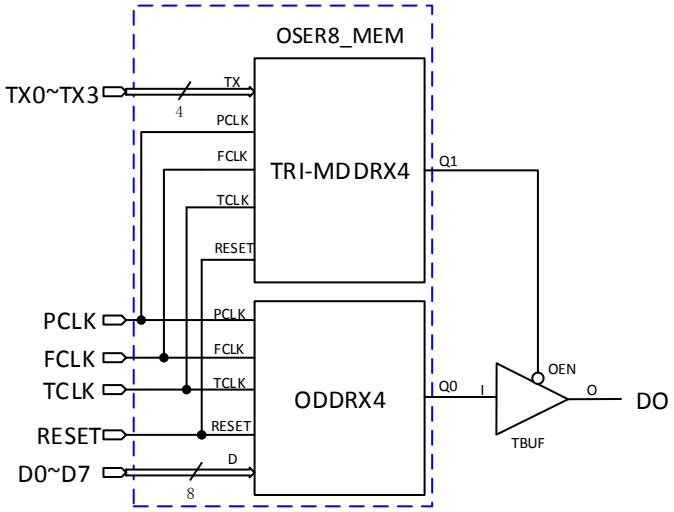
Figure1-41 OSER8_MEM Diagram



Functional Description

OSER8_MEM mode realizes 8:1 parallel serial conversion. The TCLK connects the output signal DQSW0 or DQSW270 of DQS, and outputs data from the OSER8_MEM according to the TCLK clock edge. Where Q0 is the serial output, Q1 is used for the OEN signal of IOBUF/TBUF connected by Q0. Its logic diagram is as shown in Figure 1-42.

Figure 1-42 OSER8_MEM Logic Diagram



The frequency relation between PCLK, FCLK and TCLK is $f_{PCLK} = 1/4\, f_{FCLK} = 1/4\, f_{TCLK}$.

You can determine the phase realationship between PCLK and ICLK according to DLLSTEP and WSTEP values of DQS.

Port Description

Table 1-52 Port Description

Port Name	I/O	Description
D7~D0	Input	OSER8_MEM data input
TX3~TX0	Input	Q1 generated by TRI-MDDRX4
TCLK	Input	Clock input from DQSW0 or DQSW270 in DQS module
FCLK	Input	High speed clock input
PCLK	Input	Master clock input
RESET	Input	Asynchronous reset input, active-high
Q0	Output	OSER8_MEM data output
Q1	Output	OSER8_MEM tri-state enable data output can be connected to the IOBUF/TBUF OEN signal connected by Q0, or suspend

Parameter

Table 1-53 Parameter Description

Name	Value	Default Value	Description
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset
TXCLK_POL	1’b0, 1’b1	1’b0	Q1 output clock polarity control 1'b0: data posedge output 1'b1: data negedge output
TCLK_SOURCE	"DQSW","DQSW270"	" DQSW "	TCLK source selection "DQSW" comes from DQSW0

Name	Value	Default Value	Description
			in DQS module "DQSW270" comes from DQSW270 from DQS module
HWL	"false", "true"	"false"	OSER8_MEM data d_up0/1 timing relationship control "False ": d_up1 is one cycle ahead of d_up0; "True ": d_up1 and d_up0 have the same timing

Connection Validity Rule

- Q0 can directly connect OBUF, or connect input port DI in IODELAY module;
- Q1 shall connect the OEN signal of IOBUF/TBUF connected by Q0, or suspend.
- TCLK needs DQSW0 or DQSW270 from DQS module and configure the corresponding parameters.

Primitive Instantiation

Verilog Instantiation:

```
OSER8_MEM oser8_mem_inst(
        .Q0(q0),
        .Q1(q1),
        .D0(d0),
        .D1(d1),
        .D2(d2),
        .D3(d3),
        .D4 (d4),
        .D5 (d5),
        .D6 (d6),
        .D7 (d7),
        .TX0 (tx0),
        .TX1 (tx1),
        .TX2 (tx2),
        .TX3 (tx3),
        .TCLK (tclk),
        .FCLK (fclk),
        .PCLK (pclk),
        .RESET(reset)
);
defparam uut.GSREN="false";
defparam uut.LSREN ="true";
defparam uut.HWL ="false";
defparam uut.TCLK_SOURCE ="DQSW";
defparam uut.TXCLK_POL=1'b0;
```

Vhdl Instantiation:

```
COMPONENT OSER8_MEM
    GENERIC (GSREN:string:="false";
                LSREN:string:="true";
```

```
                        HWL:string:="false";
                        TXCLK_POL:bit:='0';
                        TCLK_SOURCE:string:="DQSW"
        );
    PORT(
      Q0:OUT std_logic;
            Q1:OUT std_logic;
      D0:IN std_logic;
            D1:IN std_logic;
            D2:IN std_logic;
            D3:IN std_logic;
            D4:IN std_logic;
            D5:IN std_logic;
            D6:IN std_logic;
            D7:IN std_logic;
            TX0:IN std_logic;
            TX1:IN std_logic;
            TX2:IN std_logic;
            TX3:IN std_logic;
            TCLK:IN std_logic;
            FCLK:IN std_logic;
            PCLK:IN std_logic;
            RESET:IN std_logic
    );
END COMPONENT;
uut:OSER8_MEM
    GENERIC MAP (GSREN=>"false",
                        LSREN=>"true",
                        HWL=>"false",
                        TXCLK_POL=>'0',
                        TCLK_SOURCE=>"DQSW"
        )
    PORT MAP (
            Q0=>q0,
            Q1=>q1,
            D0=>d0,
            D1=>d1,
            D2=>d2,
            D3=>d3,
            D4=>d4,
            D5=>d5,
            D6=>d6,
            D7=>d7,
            TX0=>tx0,
            TX1=>tx1,
            TX2=>tx2,
            TX3=>tx3,
            TCLK=>tclk,
            FCLK=>fclk,
            PCLK=>pclk,
```

```
        RESET=>reset
    );
```

1.2.21 IODELAY

Primitive Name

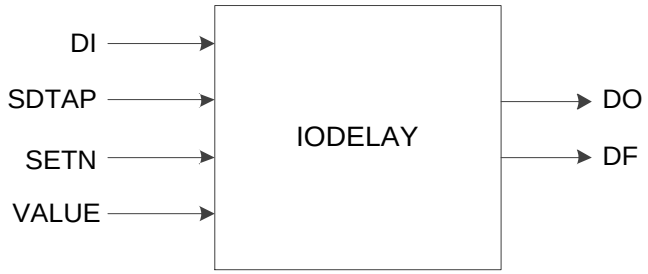
Input/Output delay (IODELAY) is a programmable delay unit in IO module.

Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

Figure 1-43 IODELAY Diagram



Functional Description

Each I/O has an IODELAY module. It can provide is about 128 steps x 25ps = 3200ps. IODELAY can be used for input or output of I/O logic, but not for both.

Port Description

Table 1-54 Port Description

Port Name	I/O	Description
DI	Input	Data input
SDTAP	Input	Control the loading static delay step length 0: loading static delay 1: adjust the dynamical delay
SETN	Input	Set the direction of dynamical delay adjustment 0: Increase delay 1: Decrease delay
VALUE	Input	VALUE is the delay value of negedge dynamical adjustment, and each pulse moves one delay step length
DO	Output	Data output
DF	Output	An output flag that represents under-flow or over-flow in dynamical delay adjustment

Parameter

Table 1-55 Parameter Description

Name	Value	Default Value	Description
C_STATIC_DLY	0~127	0	Control the static delay step length

Primitive Instantiation

Verilog Instantiation:

```
IODELAY iodelay_inst(
    .DO(dout),
    .DF(df),
    .DI(di),
    .SDTAP(sdtap),
    .SETN(setn),
    .VALUE(value)
);
defparam iodelay_inst.C_STATIC_DLY=0;
```

Vhdl Instantiation:

```
COMPONENT IODELAY
    GENERIC (C_STATIC_DLY:integer:=0
        );
    PORT(
      DO:OUT std_logic;
                DF:OUT std_logic;
      DI:IN std_logic;
                SDTAP:IN std_logic;
                SETN:IN std_logic;
                VALUE:IN std_logic
        );
END COMPONENT;
uut:IODELAY
  GENERIC MAP (C_STATIC_DLY=>0
      )
  PORT MAP (
            DO=>dout,
            DF=>df,
            DI=>di,
            SDTAP=>sdtap,
            SETN=>setn,
            VALUE=>value
    );
```

1.2.22 IEM

Primitive Name

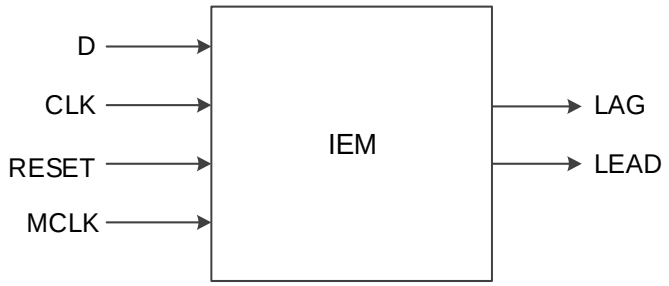
Input Edge Monitor (IEM) is a sampling module in IO module.

Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

Figure 1-44 IEM Port Diagram



Functional Description

IEM is used for sampling data edge, which can be used together with IODELAY module to adjust the dynamic sampling window for DDR mode.

Port Description

Table 1-56 Port Description

Port Name	I/O	Description
D	Input	Data input
CLK	Input	Clock input
RESET	Input	Asynchronous reset input, active-high
MCLK	Input	The IEM detection clock comes from user logic, used for output flag
LAG	Output	The output flag of IEM edge compared with LAG
LEAD	Output	The output flag of IEM edge compared with LEAD

Parameter

Table 1-57 Parameter Description

Name	Value	Default Value	Description
WINSIZE	"SMALL","MIDSMALL", "MIDLARGE","LARGE"	"SMALL"	Window size setting
GSREN	"false", "true"	"false"	Enable global reset
LSREN	"false", "true"	"true"	Enable local reset

Primitive Instantiation

Verilog Instantiation:

```
IEM iem_inst(
```

```
        .LAG(lag),
        .LEAD(lead),
        .D(d),
        .CLK(clk),
        .MCLK(mclk),
        .RESET(reset)
        );
    defparam iodelay_inst.WINSIZE = "SMALL";;
    defparam iodelay_inst.GSREN = "false";
    defparam iodelay_inst.LSREN = "true";
```

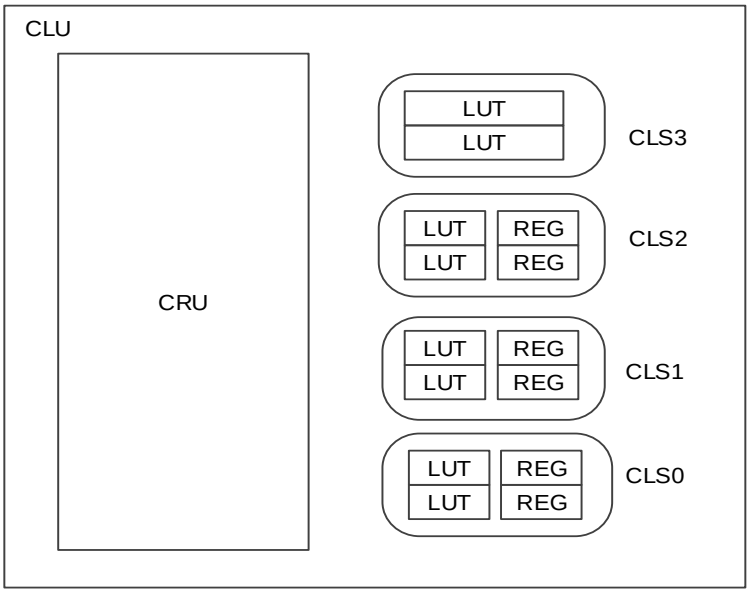
Vhdl Instantiation:

```
    COMPONENT IEM
        GENERIC (WINSIZE:string:="SMALL";
                            GSREN:string:="false";
                            LSREN:string:="true"
              );
        PORT(
          LAG:OUT std_logic;
                    LEAD:OUT std_logic;
                     D:IN std_logic;
                    CLK:IN std_logic;
                    MCLK:IN std_logic;
                    RESET:IN std_logic
      );
    END COMPONENT;
    uut:IEM
      GENERIC MAP (WINSIZE=>"SMALL",
                                GSREN=>"false",
                                LSREN=>"true"
            )
      PORT MAP (
                LAG=>lag,
                LEAD=>lead,
                D=>d,
                CLK=>clk,
                MCLK=>mclk,
                RESET=>reset
        );
```

2 CLU

Configurable Function Unit (CFU) is a basic block for FPGA products. One CFU includes four Configurable Logic Sections (CLU) and one Configurable Routing Unit (CRU), as shown in Figure 2-1. CLU can configure lookup LUT, ALU and REG. CFU can achieve the function of MUX/LUT/ALU/FF/LATCH.

Figure 2-1 CLU Block Diagram



2.1 LUT

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

Device supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

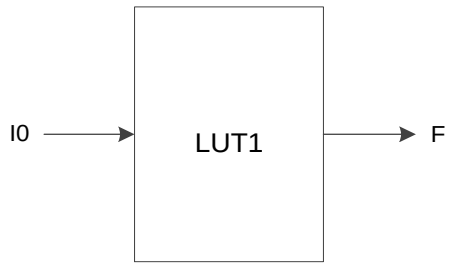
2.1.1 LUT1

Primitive Introduction

LUT1 is a simple type usually used for a buffer and an inverter. LUT1 is an one input lookup table. After initializing INIT by parameter, you can look up the corresponding data and output the result according to the input address.

Block Diagram

Figure 2-2 LUT1 Block Diagram



Port Description

Table 2-1 Port Description

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

Attribute Description

Table 2-2 Attribute Introduction

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

Truth Table

Table 2-3 MUX2_MUX8 Truth Table

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

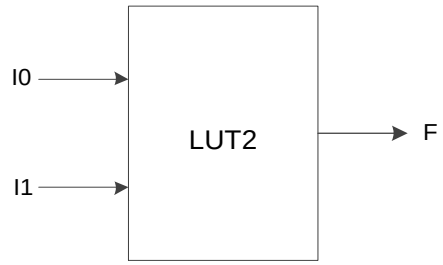
2.1.2 LUT2

Primitive Introduction

LUT2 is a two-input lookup table. After initializing INIT by parameter, you can look up the corresponding data and output the result according to the input address.

Block Diagram

Figure 2-3 LUT2 Block Diagram



Port Description

Table 2-4 Port Description

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

Attribute Description

Table 2-5 Attribute Introduction

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

Truth Table

Table 2-6 MUX2_MUX8 Truth Table

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

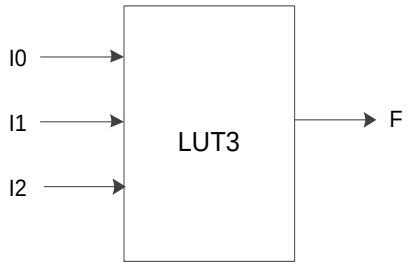
2.1.3 LUT3

Primitive Introduction

LUT3 is a three-input lookup table. After initializing INIT by parameter, you can look up the corresponding data and output the result according to the input address.

Block Diagram

Figure 2-4 LUT3 Block Diagram



Port Description

Table 2-7 Port Description

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

Attribute Description

Table 2-8 Attribute Introduction

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

Truth Table

Table 2-9 MUX2_MUX8 Truth Table

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

Primitive Instantiation

Verilog Instantiation:

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

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

Vhdl Instantiation:

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

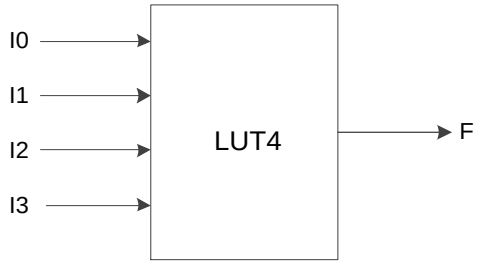
2.1.4 LUT4

Primitive Introduction

The LUT4 is a 4-input lookup table. After initializing INIT by parameter, you can look up the corresponding data and output result according to the input address.

Block Diagram

Figure 2-5 LUT4 Diagram



Port Description

Table 2-10 Port Description

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

Port Name	I/O	Description
F	Output	Data Output

Attribute Description

Table 2-11 Attribute Introduction

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

Truth Table

Table 2-12 MUX2_MUX8 Truth Table

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

```
COMPONENT LUT4
    GENERIC (INIT:bit_vector:=X"0000");
```

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

2.1.5 Wide LUT

Primitive Introduction

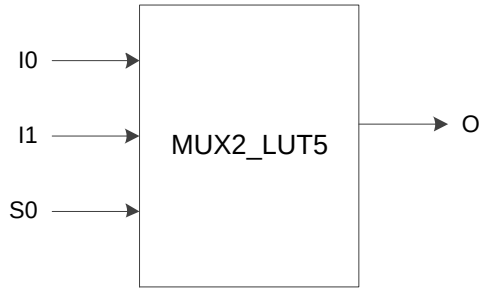
The Wide LUT is used for constructing high-order LUT using LUT4 and MUX2. MUX2 series of Gowin FPGA High-Order LUT support MUX2_LUT5/ MUX2_LUT6/ MUX2_LUT7/ MUX2_LUT8.

Modes of constructing high-order LUT are as follows: one LUT5 can be combined by two LUT4 and MUX2_LUT5; one LUT6 can be combined by two LUT5 and MUX2_LUT6; one LUT7 can be combined by two LUT6 and MUX2_LUT7;S LUT8 can be combined by two LUT7 and MUX2_LUT8.

The following mainly takes MUX2_LUT5 as an example to introduce the use of Wide LUT.

Block Diagram

Figure 2-6 MUX2_LUT5 Block Diagram



Port Description

Table 2-13 Port Description

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

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

Truth Table

Table 2-14 MUX2_MUX8 Truth Table

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

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

2.2 MUX

The MUX is a multiplexer. It transmits one data to the output based on the channel-selection signal. The Gowin MUX contains 2-to-1 multiplexer and 4-to-1 multiplexer.

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

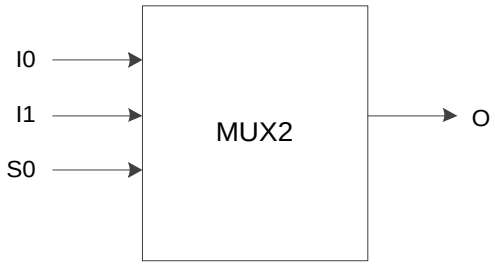
2.2.1 MUX2

Primitive Introduction

2-to-1 Multiplexer (MUX2) chooses one of the two inputs as an output based on the selected signal.

Blcok Diagram

Figure 2-7 MUX2 Block Diagram



Port Description

Table 2-15 Port Description

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

Truth Table

Table 2-16 MUX2_MUX8 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
         );
COMPONENT;
uut:MUX2
  PORT MAP (
            O=>O,
```

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

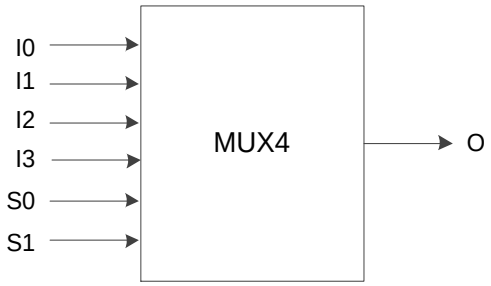
2.2.2 MUX4

Primitive Introduction

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

Block Diagram

Figure 2-8 MUX4 Block Diagram



Port Description

Table 2-17 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	Select Signal Input
S1	Input	Select Signal Input
O	Output	Data Output

Truth Table

Table 2-18 MUX2_MUX8 Truth Table

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

2.2.3 Wide MUX

Primitive Introduction

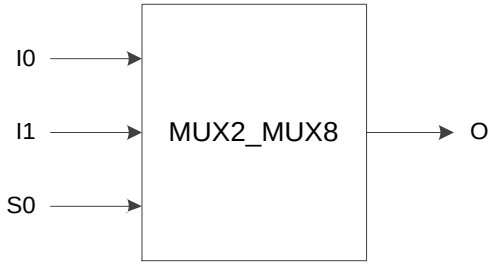
The Wide MUX is used for constructing high-order MUX using MUX4 and MUX2. The MUX2 series of Gowin FPGA High-Order MUX support MUX2_MUX8/ MUX2_MUX16/ MUX2_MUX32.

Modes of constructing high-order MUX are as follows: One MUX8 can be combined by two MUX4 and MUX2_MUX8; one MUX16 can be combined by two MUX8 and MUX2_MUX16; one MUX32 can be combined by two MUX16 and MUX2_MUX32.

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

Block Diagram

Figure 2-9 MUX2_MUX8 Block Diagram



Port Description

Table 2-19 Port Description

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

Truth Table

Table 2-20 MUX2_MUX8 Truth Table

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

Primitive Instantiation

Verilog Instantiation:

```
MUX2_MUX8 instName (
        .I0(o0),
        .I1(o1),
        .S0(S2),
        .O(O)
);
MUX4 mux_0 (
        .I0(i0),
        .I1(i1),
        .I2(i2),
        .I3(i3),
        .S0(s0),
        .S1(s1),
        .O(o0)
);
MUX4 mux_1 (
        .I0(i4),
        .I1(i5),
```

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

Vhdl Instantiation:

```
    COMPONENT MUX2_MUX8
        PORT(
          O:OUT std_logic;
            I0:IN std_logic;
                    I1:IN std_logic;
                    S0:IN std_logic
             );
    END COMPONENT;
    COMPONENT MUX4
             PORT(
          O:OUT std_logic;
            I0:IN std_logic;
                    I1:IN std_logic;
                    I2:IN std_logic;
                    I3:IN std_logic;
                    S0:IN std_logic;
                    S1:IN std_logic
            );
    END COMPONENT;
    uut1:MUX2_MUX8
      PORT MAP (
                O=>O,
                I0=>o0,
                I1=>o1,
                S0=>S2
        );
    uut2:MUX4
            PORT MAP (
                O=>o0,
                I0=>I0,
                I1=>I1,
                I2=>I2,
                I3=>I3,
                S0=>S0,
                S1=>S1
        );
    uut3:MUX4sss
            PORT MAP (
                O=>o1,
                I0=>I4,
                I1=>I5,
                I2=>I6,
                I3=>I7,
```

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

2.3 ALU

Primitive Introduction

The Arithmetic Logic Unit (ALU) realizes the functions of ADD/SUB/ADDSUB.

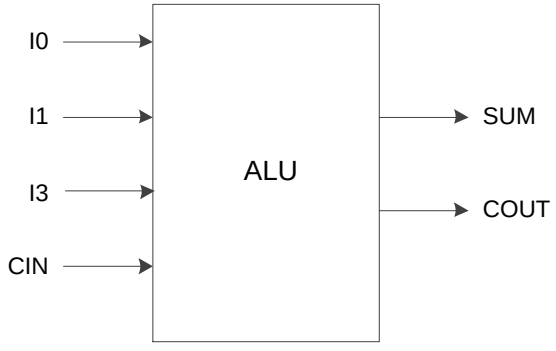
Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55. Specific functions are listed in Table 2-21.

Table 2-21 ALU Functions

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

Block Diagram

Figure 2-10 ALU Block Diagram



Port Description

Table 2-22 Port Description

Port Name	Input/Output	Description
I0	Input	Data Input

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

Attribute Description

Table 2-23 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

```
    GENERIC MAP(ALU_MODE=>1)
PORT MAP (
        COUT=>COUT,
        SUM=>SUM,
        I0=>I0,
        I1=>I1,
        I3=>I3,
        CIN=>CIN
  );
```

2.4 FF

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

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55. There are 20 primitives associated with FF, as shown in Table 2-24.

Table 2-24 Primitives Associated With FF

Primitive	Description
DFF	Double-edge flip-flops
DFFE	DFFE
DFFS	DFFS
DFFSE	DFFSE
DFFR	DFFR
DFFRE	DFFRE
DFFP	DFFP
DFFPE	DFFPE
DFFC	DFFC
DFFCE	DFFCE
DFFN	DFFN
DFFNE	DFFNE
DFFNS	DFFNS
DFFNSE	DFFNSE
DFFNR	DFFNR
DFFNRE	DFFNRE
DFFNP	DFFNP
DFFNPE	DFFNPE
DFFNC	DFFNC
DFFNCE	DFFNCE

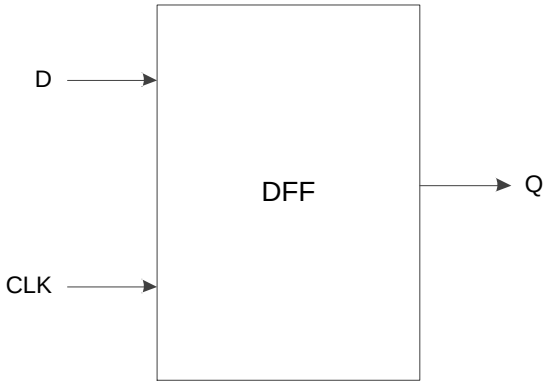
2.4.1 DFF

Primitive Introduction

The D Flip-Flop (DFF) is commonly used for signal sampling and processing for posedge.

Block Diagram

Figure 2-11 DFF Block Diagram



Port Description

Table 2-25 Port Description

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

Attribute Description

Table 2-26 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
INIT	1’b0,1’b1	1’b0	Initial value for DFF

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

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

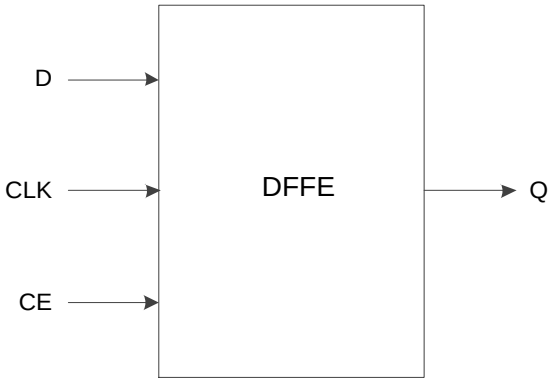
2.4.2 DFFE

Primitive Introduction

The D Flip-Flop with Clock Enable (DFFE) is with the function of clock enable for posedge.

Block Diagram

Figure 2-12 DFFE Block Diagram



Port Description

Table 2-27 Port Description

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

Attribute Description

Table 2-28 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

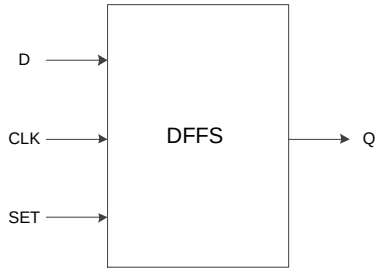
2.4.3 DFFS

Primitive Introduction

The D Flip-Flop with Synchronous Set (DFFS) is with the function of synchronous setting for posedge.

Block Diagram

Figure 2-13 DFFS Block Diagram



Port Description

Table 2-29 Port Description

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

Attribute Description

Table 2-30 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

2.4.4 DFFSE

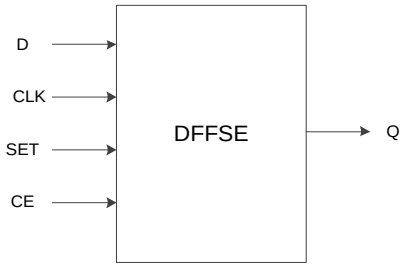
Primitive Introduction

The D Flip-Flop with Clock Enable and Synchronous Set (DFFSE) is

with the functions of synchronous setting and clock enable for posedge.

Block Diagram

Figure 2-14 DFFSE Block Diagram



Port Description

Table 2-31 Port Description

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

Attribute Description

Table 2-32 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for DFFSE

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

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

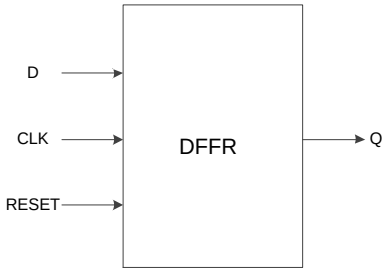
2.4.5 DFFR

Primitive Introduction

The D Flip-Flop with Synchronous Reset (DFFR) is with the function of synchronous resetting for posedge.

Block Diagram

Figure 2-15 DFFR Block Diagram



Port Description

Table 2-33 Port Description

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

Attribute Description

Table 2-34 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

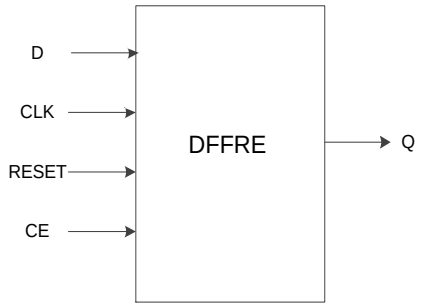
2.4.6 DFFRE

Primitive Introduction

The D Flip-Flop with Clock Enable and Synchronous Reset (DFFRE) is with the functions of synchronous setting and clock enable for posedge.

Block Diagram

Figure 2-16 DFFRE Block Diagram



Port Description

Table 2-35 Port Description

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

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

Attribute Description

Table 2-36 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

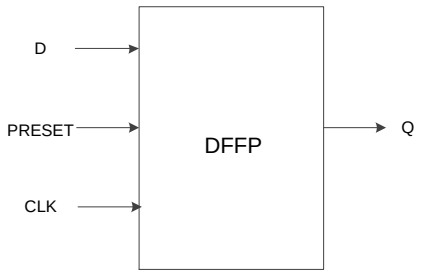
2.4.7 DFFP

Primitive Introduction

The D Flip-Flop with Asynchronous Preset (DFFP) is with the function of synchronous setting for posedge.

Block Diagram

Figure 2-17 DFFP Block Diagram



Port Description

Table 2-37 Port Description

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

Attribute Description

Table 2-38 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

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

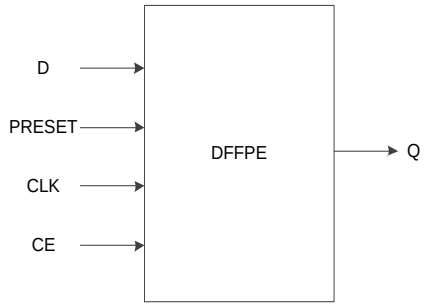
2.4.8 DFFPE

Primitive Introduction

The D Flip-Flop with Clock Enable and Asynchronous Preset (DFFPE) is with the functions of synchronous setting and clock enable for posedge.

Block Diagram

Figure 2-18 DFFPE Block Diagram



Port Description

Table 2-39 Port Description

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

Attribute Description

Table 2-40 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

```
DFFPE instName (
        .D(D),
        .CLK(CLK),
        .PRESET(PRESET),
```

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

Vhdl Instantiation:

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

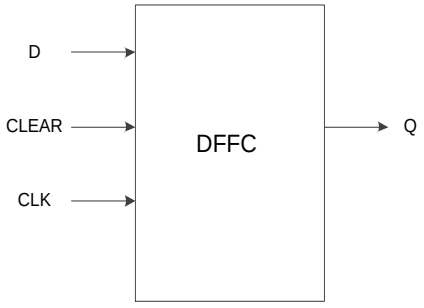
2.4.9 DFFC

Primitive Introduction

The D Flip-Flop with Asynchronous Clear (DFFC) is with the function of synchronous setting for posedge.

Block Diagram

Figure 2-19 DFFC Blcok Diagram



Port Description

Table 2-41 Port Description

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

Port Name	I/O	Description
Q	Output	Data Output

Attribute Description

Table 2-42 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

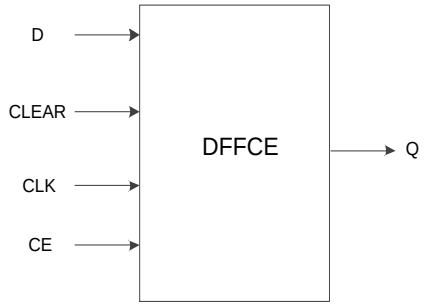
2.4.10 DFFCE

Primitive Introduction

The D Flip-Flop with Clock Enable and Asynchronous Clear (DFFCE) is with the functions of synchronous setting and clock enable for posedge.

Block Diagram

Figure 2-20 DFFCE Block Diagram



Port Description

Table 2-43 Port Description

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

Attribute Description

Table 2-44 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

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

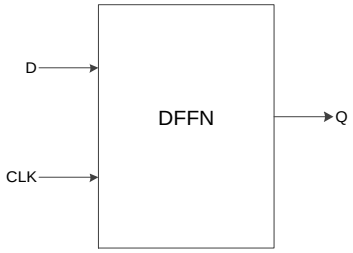
2.4.11 DFFN

Primitive Introduction

D Flip-Flop with Negative-Edge Clock (DFFN) is triggered by negedge.

Block Diagram

Figure 2-21 DFFN Block Diagram



Port Description

Table 2-45 Port Description

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

Attribute Description

Table 2-46 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

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

Vhdl Instantiation:

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

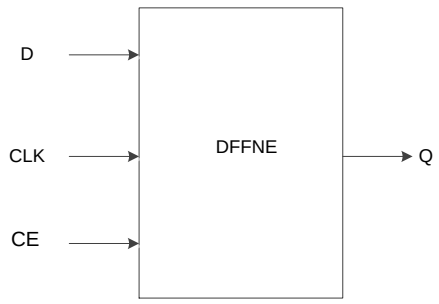
2.4.12 DFFNE

Primitive Introduction

The D Flip-Flop with Negative-Edge Clock and Clock Enable (DFFNE) is with the function of clock enable for negedge.

Block Diagram

Figure 2-22 DFFNE Block Diagram



Port Description

Table 2-47 Port Description

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

Attribute Description

Table 2-48 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFFNE

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

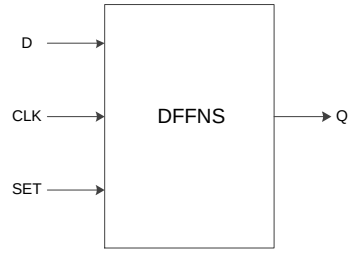
2.4.13 DFFNS

Primitive Introduction

The D Flip-Flop with Negative-Edge Clock and Synchronous Set (DFFNS) is with the function of synchronous setting for negedge.

Block Diagram

Figure 2-23 DFFNS Blcok Diagram



Port Description

Table 2-49 Port Description

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

Attribute Description

Table 2-50 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for DFFNS

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

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

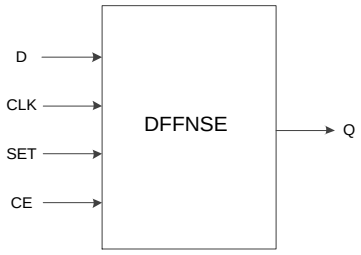
2.4.14 DFFNSE

Primitive Introduction

The D Flip-Flop with Negative-Edge Clock,Clock Enable,and Synchronous Set (DFFNSE) is with the function of synchronous setting and clock enable for negedge.

Block Diagram

Figure 2-24 DFFNSE Block Diagram



Port Description

Table 2-51 Port Description

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

Attribute Description

Table 2-52 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for DFFNSE

Primitive Instantiation

Verilog Instantiation:

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

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

Vhdl Instantiation:

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

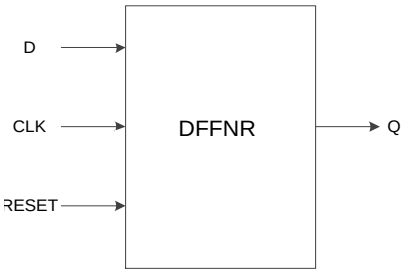
2.4.15 DFFNR

Primitive Introduction

The D Flip-Flop with Negative-Edge Clock and Synchronous Reset (DFFNR) is with the function of synchronous setting for negedge.

Block Diagram

Figure 2-25 DFFNR Block Diagram



Port Description

Table 2-53 Port Description

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

Attribute Description

Table 2-54 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for DFFNR

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

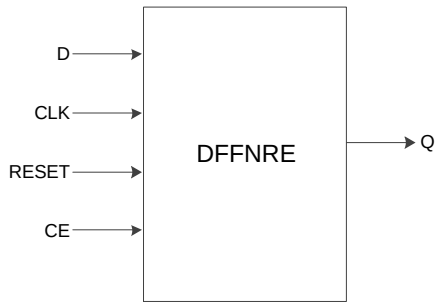
2.4.16 DFFNRE

Primitive Introduction

The D Flip-Flop with Negative-Edge Clock,Clock Enable, and Synchronous Reset (DFFNRE) is with the function of synchronous setting and clock enable for negedge.

Blcok Diagram

Figure 2-26 DFFNRE Blcok Diagram



Port Description

Table 2-55 Port Description

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

Attribute Description

Table 2-56 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

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

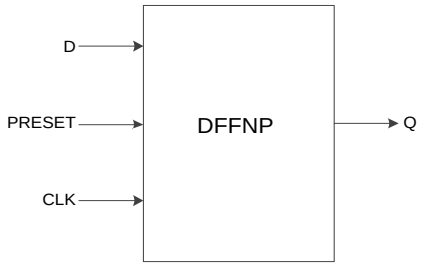
2.4.17 DFFNP

Primitive Introduction

The D Flip-Flop with Negative-Edge Clock and Asynchronous Preset (DFFNP) is with the function of asynchronous setting for negedge.

Block Diagram

Figure 2-27 DFFNP Block Diagram



Port Description

Table 2-57 Port Description

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

Attribute Description

Table 2-58 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

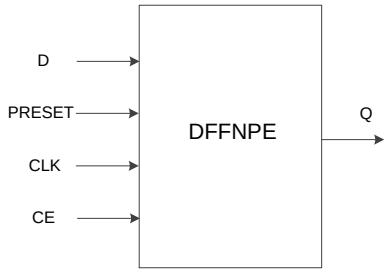
2.4.18 DFFNPE

Primitive Introduction

The D Flip-Flop with Negative-Edge Clock,Clock Enable, and Asynchronous Preset (DFFNPE) is with the function of asynchronous setting and clock enable for negedge.

Block Diagram

Figure 2-28 DFFNPE Block Diagram



Port Description

Table 2-59 Port Description

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

Attribute Description

Table 2-60 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for DFFNPE

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

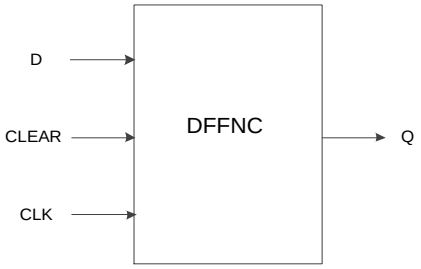
2.4.19 DFFNC

Primitive Introduction

The D Flip-Flop with Negative-Edge Clock and Asynchronous Clear (DFFNC) is with the function of asynchronous setting for negedge.

Block Diagram

Figure 2-29 DFFNC Block Diagram



Port Description

Table 2-61 Port Description

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

Attribute Description

Table 2-62 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

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

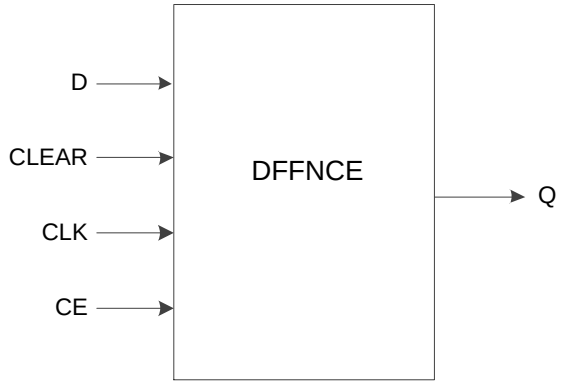
2.4.20 DFFNCE

Primitive Introduction

The D Flip-Flop with Negative-Edge Clock,Clock Enable and Asynchronous Clear (DFFNCE) is with the function of asynchronous setting and clock enable for negedge.

Block Diagram

Figure 2-30 DFFNCE Block Diagram



Port Description

Table 2-63 Port Description

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

Attribute Description

Table 2-64 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

2.5 LATCH

LATCH is a kind of memory cell circuit and its status can be changed under the specified input pulse.

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55. There are 12 primitives related to LATCH,as shown in Table 2-65.

Table 2-65 Primitives Related with LATCH

Primitive	Description
DL	Data Latch
DLE	DLE
DLC	DLC
DLCE	DLCE
DLP	DLP
DLPE	DLPE
DLN	DLN
DLNE	DLNE
DLNC	DLNC
DLNCE	DLNCE
DLNP	DLNP
DLNPE	DLNPE

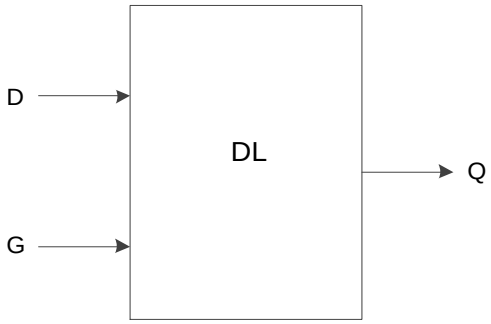
2.5.1 DL

Primitive Introduction

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

Block Diagram

Figure 2-31 DL Block Diagram



Port Description

Table 2-66 Port Description

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

Attribute Description

Table 2-67 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

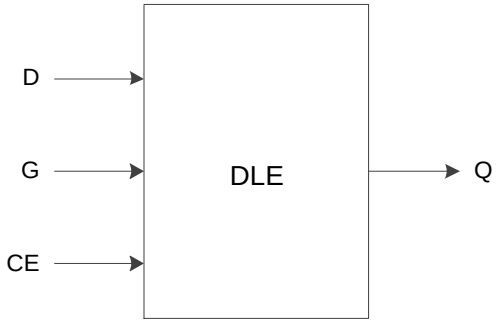
2.5.2 DLE

Primitive Introduction

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

Block Diagram

Figure 2-32 DLE Block Diagram



Port Description

Table 2-68 Port Description

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

Attribute Description

Table 2-69 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for initial DLE

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

2.5.3 DLC

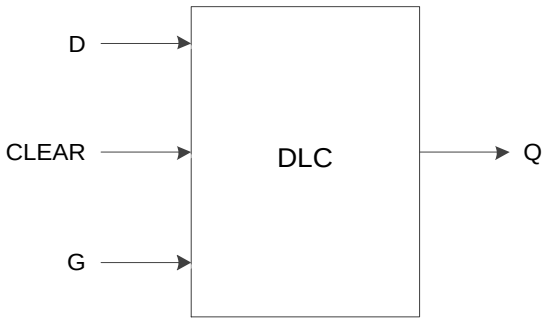
Primitive Introduction

Data Latch with Asynchronous Clear (DLC) is a latch with the function

of reset . The control signal G is active-high.

Block Diagram

Figure 2-33 DLC Block Diagram



Port Description

Table 2-70 Port Description

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

Attribute Description

Table 2-71 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for initial DLC

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

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

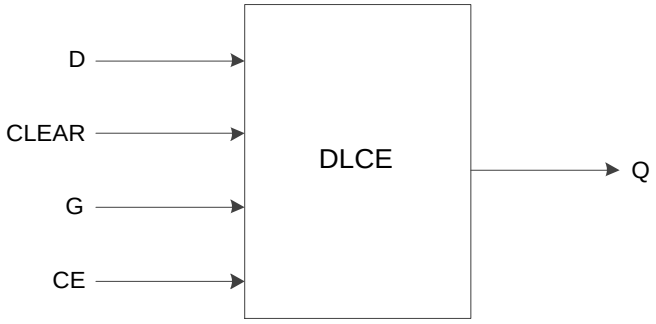
2.5.4 DLCE

Primitive Introduction

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

Block Diagram

Figure 2-34 DLCE Block Diagram



Port Description

Table 2-72 Port Description

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

Attribute Description

Table 2-73 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for initial DLCE

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

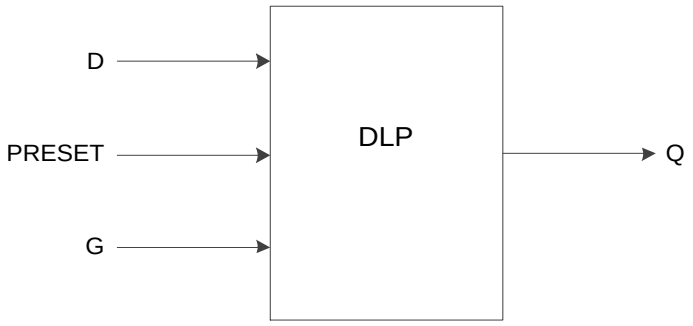
2.5.5 DLP

Primitive Introduction

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

Block Diagram

Figure 2-35 DLP Blcok Diagram



Port Description

Table 2-74 Port Description

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

Attribute Description

Table 2-75 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

2.5.6 DLPE

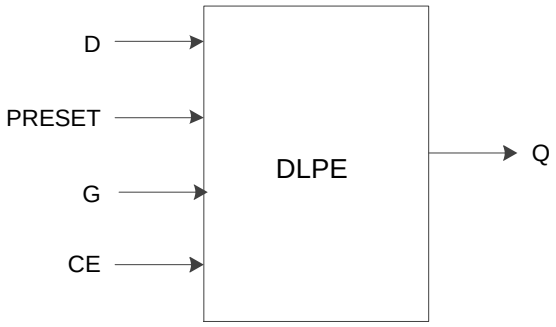
Primitive Introduction

Data Latch with Asynchronous Preset and Latch Enable (DLPE) is a

latch with the functions of enable control and set, and control signal G is active-high.

Block Diagram

Figure 2-36 DLPE Block Diagram



Port Description

Table 2-76 Port Description

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

Attribute Description

Table 2-77 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
INIT	1'b0,1'b1	1'b1	Initial value for initial DLPE

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

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

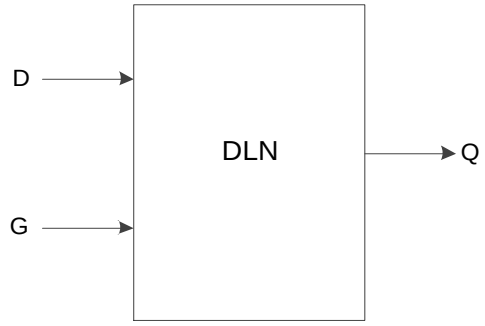
2.5.7 DLN

Primitive Introduction

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

Block Diagram

Figure 2-37 DLNP Block Diagram



Port Description

Table 2-78 Port Description

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

Attribute Description

Table 2-79 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

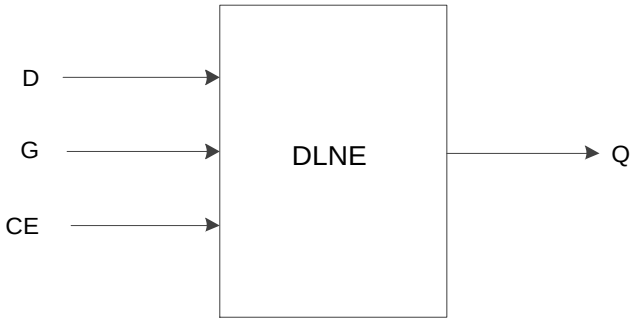
2.5.8 DLNE

Primitive Introduction

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.

Block Diagram

Figure 2-38 DLNE Block Diagram



Port Description

Table 2-80 Port Description

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

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

Attribute Description

Table 2-81 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
INIT	1’b0,1’b1	1’b0	Initial value for initial DLNE

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

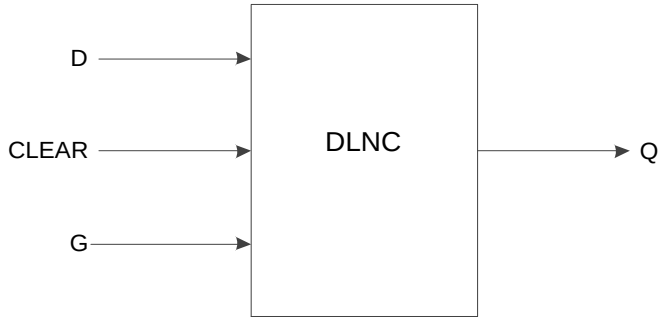
2.5.9 DLNC

Primitive Introduction

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

Block Diagram

Figure 2-39 DLNC Block Diagram



Port Description

Table 2-82 Port Description

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

Attribute Description

Table 2-83 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for initial DLNC

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

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

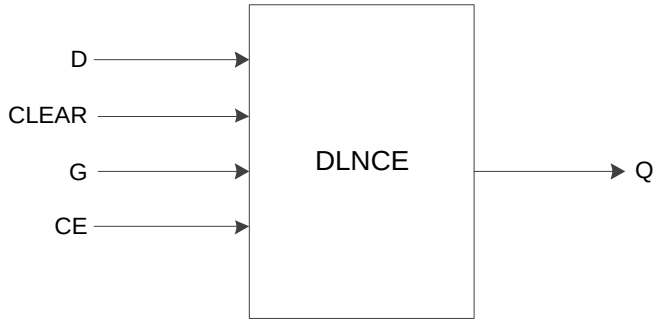
2.5.10 DLNCE

Primitive Introduction

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

Block Diagram

Figure 2-40 DLNCE Block Diagram



Port Description

Table 2-84 Port Description

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

Attribute Description

Table 2-85 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
INIT	1'b0,1'b1	1'b0	Initial value for initial DLNCE

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

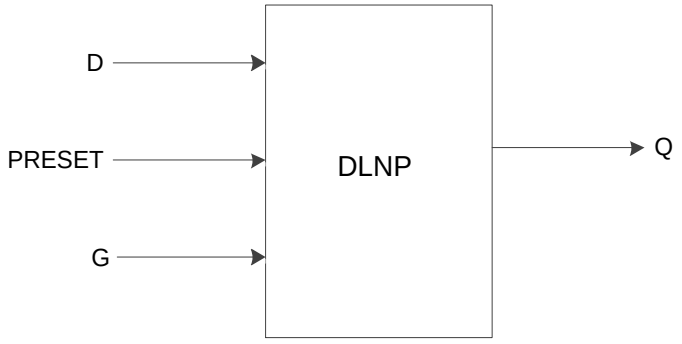
2.5.11 DLNP

Primitive Introduction

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

Block Diagram

Figure 2-41 DLNP Block Diagram



Port Description

Table 2-86 Port Description

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

Attribute Description

Table 2-87 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

2.5.12 DLNPE

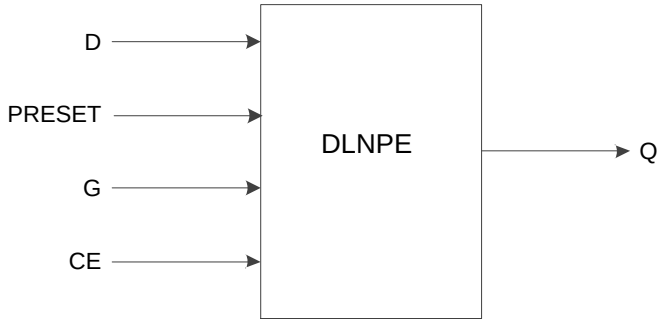
Primitive Introduction

Data Latch with Asynchronous Preset,Latch Enable and Inverted Gate

(DLNPE) is a latch with the functions of enable control and set, and control signal G is active-low.

Block Diagram

Figure 2-42 DLNPE Block Diagram



Port Description

Table 2-88 Port Description

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

Attribute Description

Table 2-89 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
INIT	1’b0,1’b1	1’b1	Initial value for initial DLNPE

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

```
COMPONENT DLNPE
    GENERIC (INIT:bit:=’1’);
    PORT(
      Q:OUT std_logic;
```

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

3 CFU

CFU is short for Configurable Fuction Unit. Unlike CLU, CFU can be configured as SSRAM mode.

3.1 SSRAM

The SSRAM can be configured as single port mode, semi-dual port mode and read-only mode, as shown in Table 3-1.

Devices supported: GW1NZ-1, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Table 3-1 SSRAM

Primitive	Description
RAM16S1	Single port SSRAM with address depth 16 and data width 1
RAM16S2	Single port SSRAM with address depth 16 and data width 2
RAM16S4	Single port SSRAM with address depth 16 and data width 4
RAM16SDP1	Semi-dual port SSRAM with address depth 16 and data width 1
RAM16SDP2	Semi-dual port SSRAM with address depth 16 and data width 2
RAM16SDP4	Semi-dual port SSRAM with address depth 16 and data width 4
ROM16	Read-only ROM with address depth 16 and data width 1

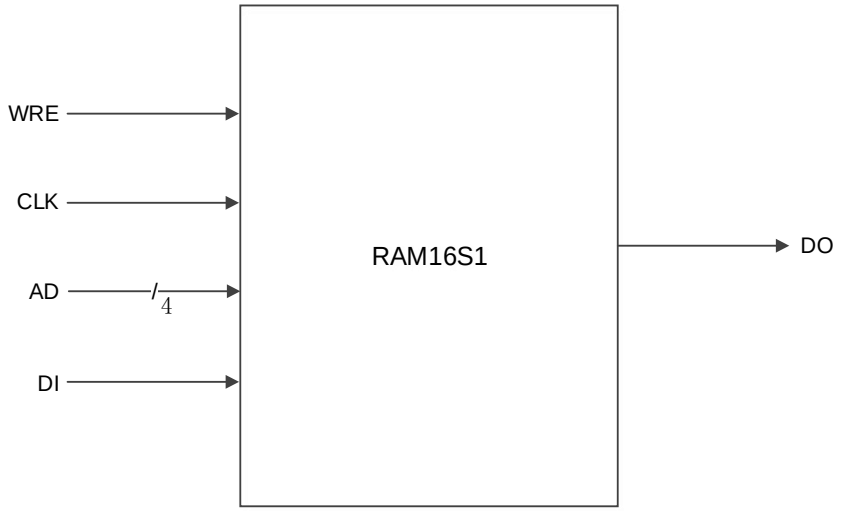
3.1.1 RAM16S1

Primitive Introduction

16-Deep by 1-Wide Single-port SSRAM (RAM16S1) is a single-port SSRAM with address depth of 16 and bit width of 1.

Block Diagram

Figure 3-1 RAM16S1Blcok Diagram



Port Description

Table 3-2 Port Description

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

Attribute Description

Table 3-3 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

```
COMPONENT RAM16S1
```

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

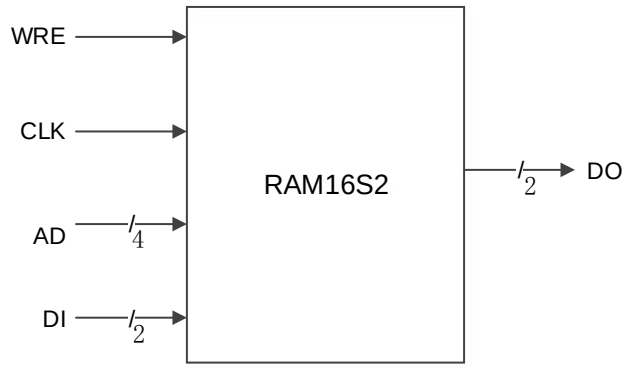
3.1.2 RAM16S2

Primitive Introduction

16-Deep by 2-Wide Single-port SSRAM (RAM16S2) is a single-port SSRAM with address depth of 16 and bit width of 2.

Block Diagram

Figure 3-2 RAM16S2 Block Diagram



Port Description

Table 3-4 Port Description

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

Attribute Description

Table 3-5 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

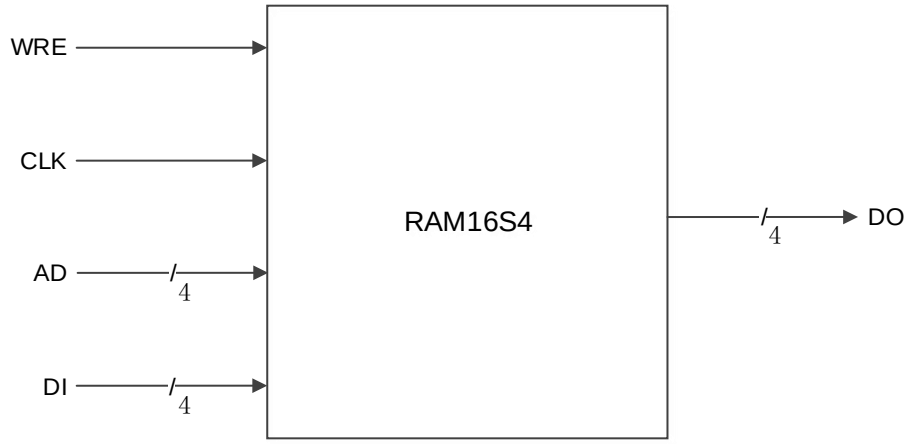
3.1.3 RAM16S4

Primitive Introduction

16-Deep by 4-Wide Single-port SSRAM (RAM16S4) is a single-port SSRAM with address depth of 16 and bit width of 4.

Diagram

Figure 3-3 RAM16S4 Diagram



Port Description

Table 3-6 Port Description

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

Attribute Description

Table 3-7 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

```
RAM16S4 instName(
    .DI(DI[3:0]),
    .WRE(WRE),
    .CLK(CLK),
    .AD(AD[3:0]),
    .DO(DOUT[3:0])
);
defparam instName.INIT_0=16'h0450;
defparam instName.INIT_1=16'h1ac3;
defparam instName.INIT_2=16'h1240;
defparam instName.INIT_3=16'h045c;
```

Vhdl Instantiation:

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

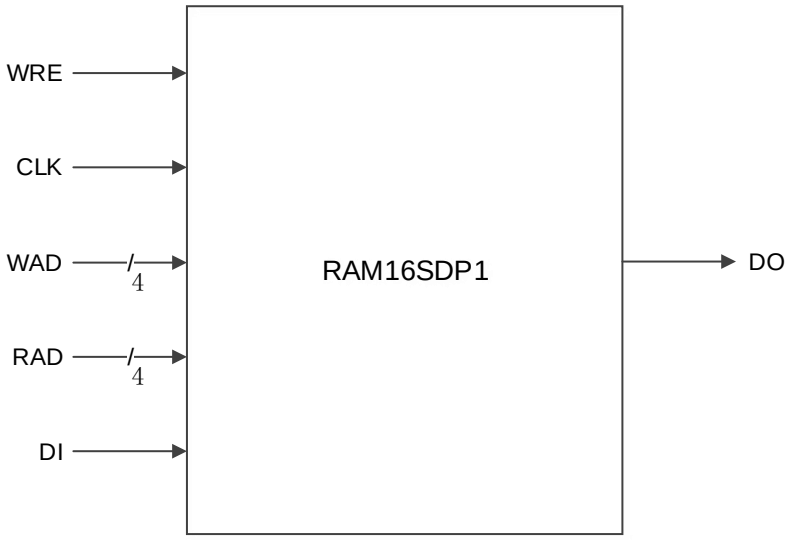
3.1.4 RAM16SDP1

Primitive Introduction

16-Deep by 1-Wide Semi Dual-port SSRAM (RAM16SDP1) is a semi-dual-port SSRAM with address depth of 16 and bit width of 1.

Block Diagram

Figure 3-4 RAMSDP1 Block Diagram



Port Description

Table 3-8 Port Description

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

Attribute Description

Table 3-9 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

```
RAM16SDP1 instName(
      .DI(DI),
      .WRE(WRE),
      .CLK(CLK),
      .WAD(WAD[3:0]),
      .RAD(RAD[3:0]),
      .DO(DOUT)
);
defparam instName.INIT_0=16'h0100;
```

Vhdl Instantiation:

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

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

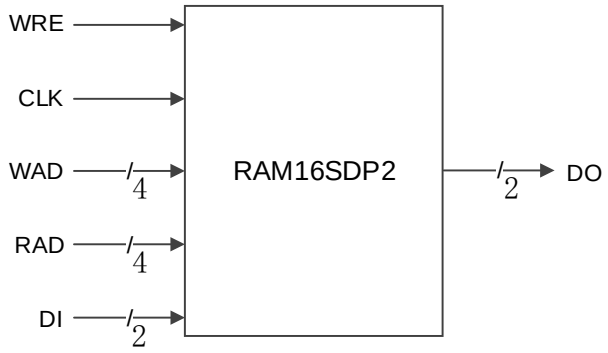
3.1.5 RAM16SDP2

Primitive Introduction

16-Deep by 2-Wide Semi Dual-port SSRAM (RAM16SDP2) is a semi-dual-port SSRAM with address depth of 16 and bit width of 2.

Block Diagram

Figure 3-5 RAM16SDP2 Block Diagram



Port Description

Table 3-10 Port Description

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

Attribute Description

Table 3-11 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

```
RAM16SDP2 instName(
    .DI(DI[1:0]),
    .WRE(WRE),
```

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

Vhdl Instantiation:

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

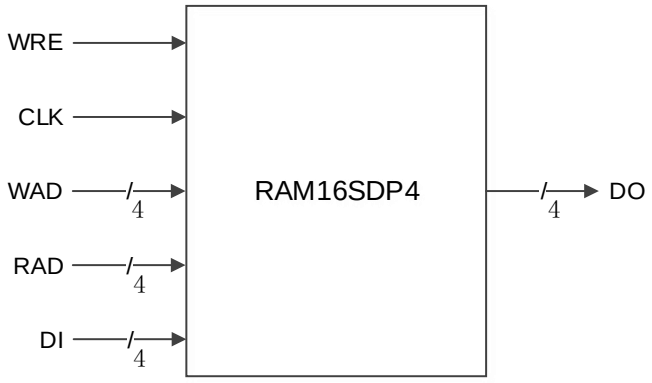
3.1.6 RAM16SDP4

Primitive Introduction

16-Deep by 4-Wide Semi Dual-port SSRAM (RAM16SDP4) is a semi-dual-port SSRAM with address depth of 16 and bit width of 4.

Diagram

Figure3-6 RAMSDP4 Diagram



Port Description

Table 3-12 Port Description

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

Attribute Description

Table 3-13 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

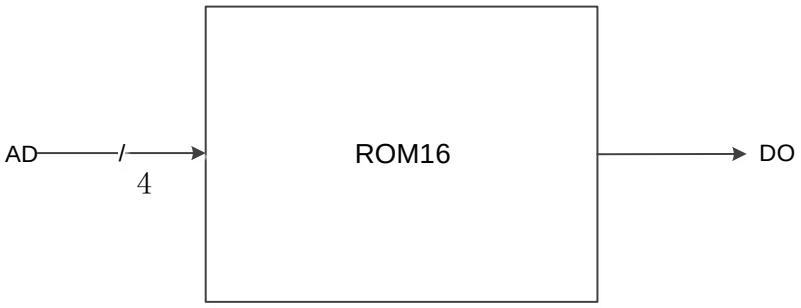
3.1.7 ROM16

Primitive Introduction

ROM16 is a read-only memory with address depth 16 and data width1. The memory is initialized using INIT.

Block Diagram

Figure 3-7 ROM16 Block Diagram



Port Description

Table 3-14 Port Description

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

Attribute Description

Table 3-15 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

4 Block SRAM

Block SRAM is a block static random access memory. According to the configuration mode, Block SRAM includes single port mode (SP/SPX9), dual port mode (DP/DPX9), semi-dual mode (SDP/SDPX9), and read-only mode (ROM/ROMX9).

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Note!

GW1N-1S, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C do not support DP/DPX9.

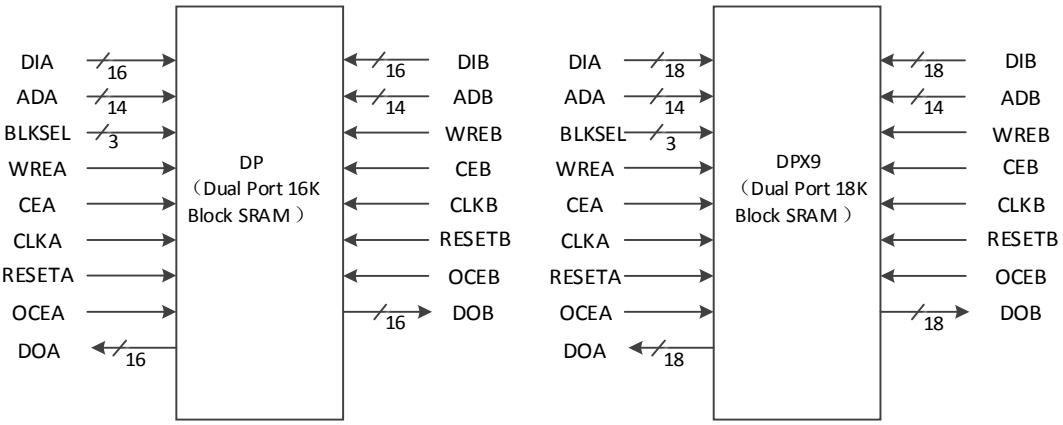
4.1 DP/DPX9

Primitive Name

DP/DPX9 (True Dual Port 16K Block SRAM/True Dual Port 18K Block SRAM).

Port Diagram

Figure 4-1 DP/DPX9 Port Diagram



Functional Description

DP/DPX9 works in dual port mode with its memory space 16K bit/18K bit. The read/write operation of the single port is controlled both at port A and port B. DP/DPX9 supports 2 read modes (bypass mode and pipeline mode) and 3 write modes (normal mode, write-through mode and read-before-write mode).

If DP is configured as 16bit and DPX9 is configured as 18bit, the byte enable function of BSRAM can be realized, i.e., the data written to memory by the lower four bit of AD, active-high. ADA[0] controls whether DIA [7:0] / DIA [8:0] writes to memory, ADA [1] controls whether DIA[15:8]/DIA [17:9] writes to memory, ADB[0] controls whether DIB [7:0] / DIB [8:0] writes to memory, and ADB [1] controls whether DIB [15:8] / DIB [17:9] writes to memory

Read mode

READ_MODE0 and READ_MODE1 enable or disable the output pipeline register at A port and B port. When the output pipeline register is used, the read operation needs extra delay cycle.

Write mode

WRITE_MODE0 and WRITE_MODE1 configure write mode at A and B including normal mode, write-through mode and read-before-write mode. The corresponding internal timing diagram of different modes is shown in the Figure 4-2 and Figure 4-7.

Figure 4-2 Timing Diagram of DP/DPX9 Normal（Bypass Read Mode）

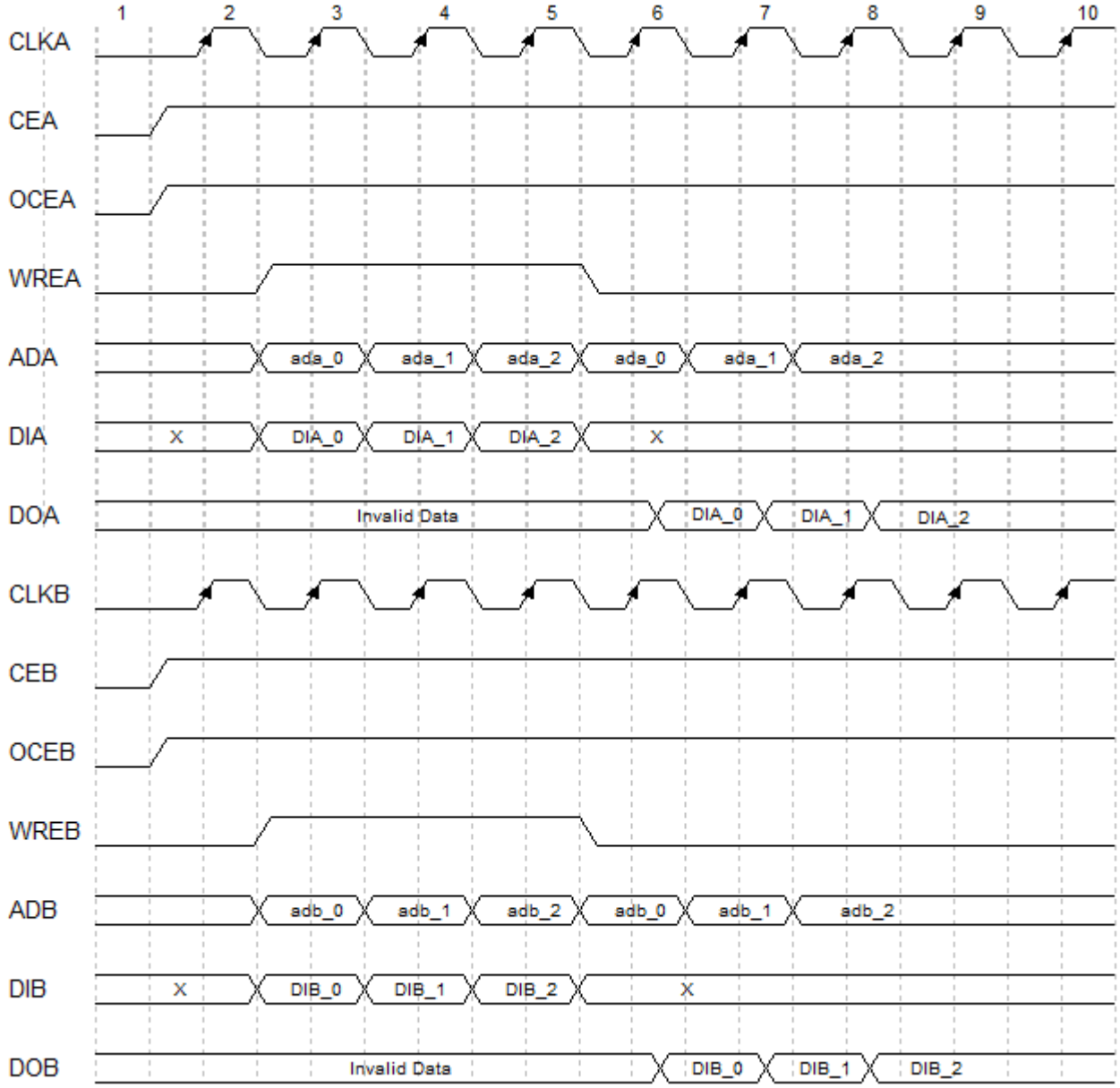


Figure 4-3 Timing Diagram of DP/DPX9 Normal（Pipeline Read Mode）

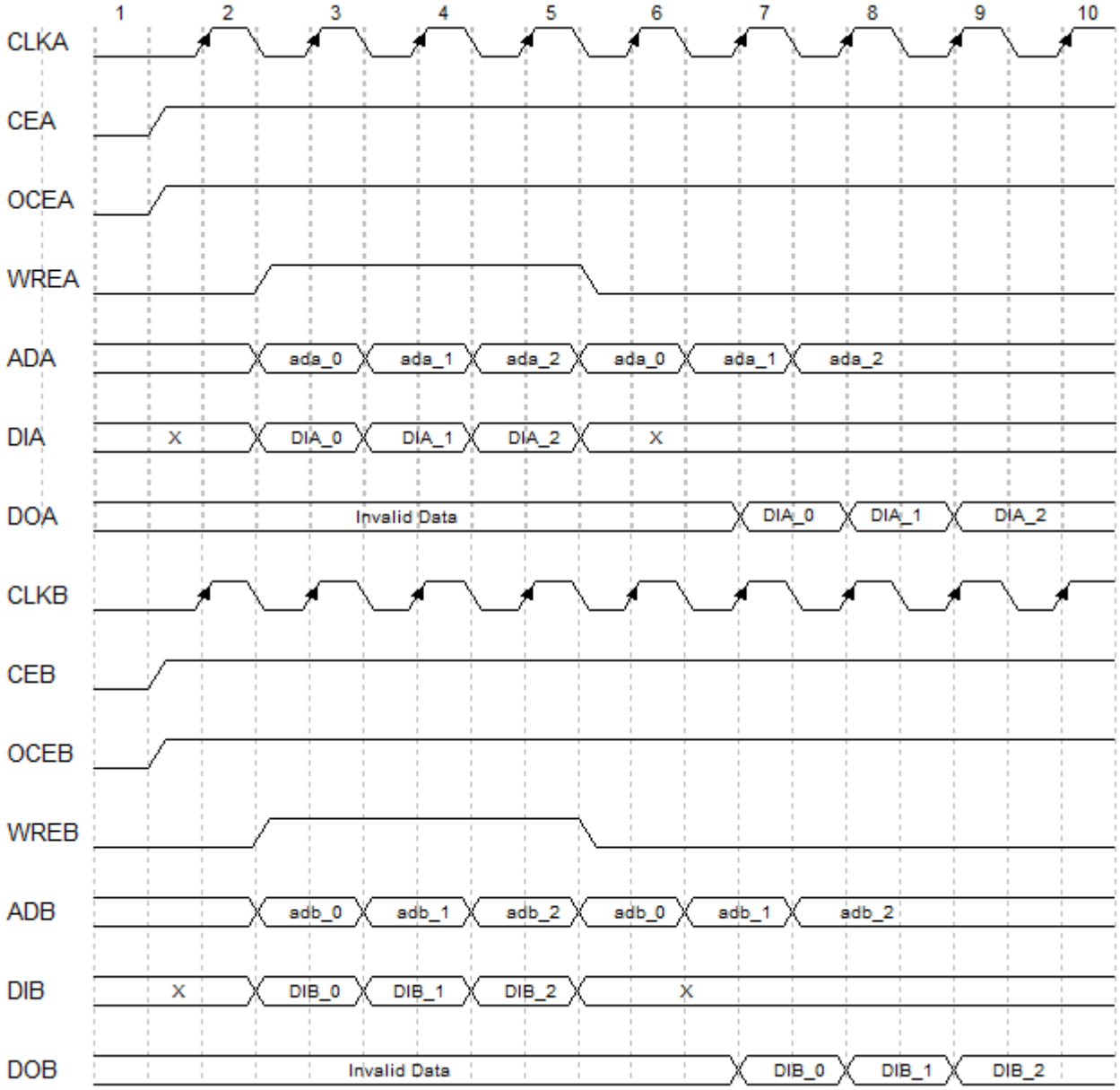


Figure 4-4 Timing Diagram of DP/DPX9 Write-Through（Bypass Read Mode）

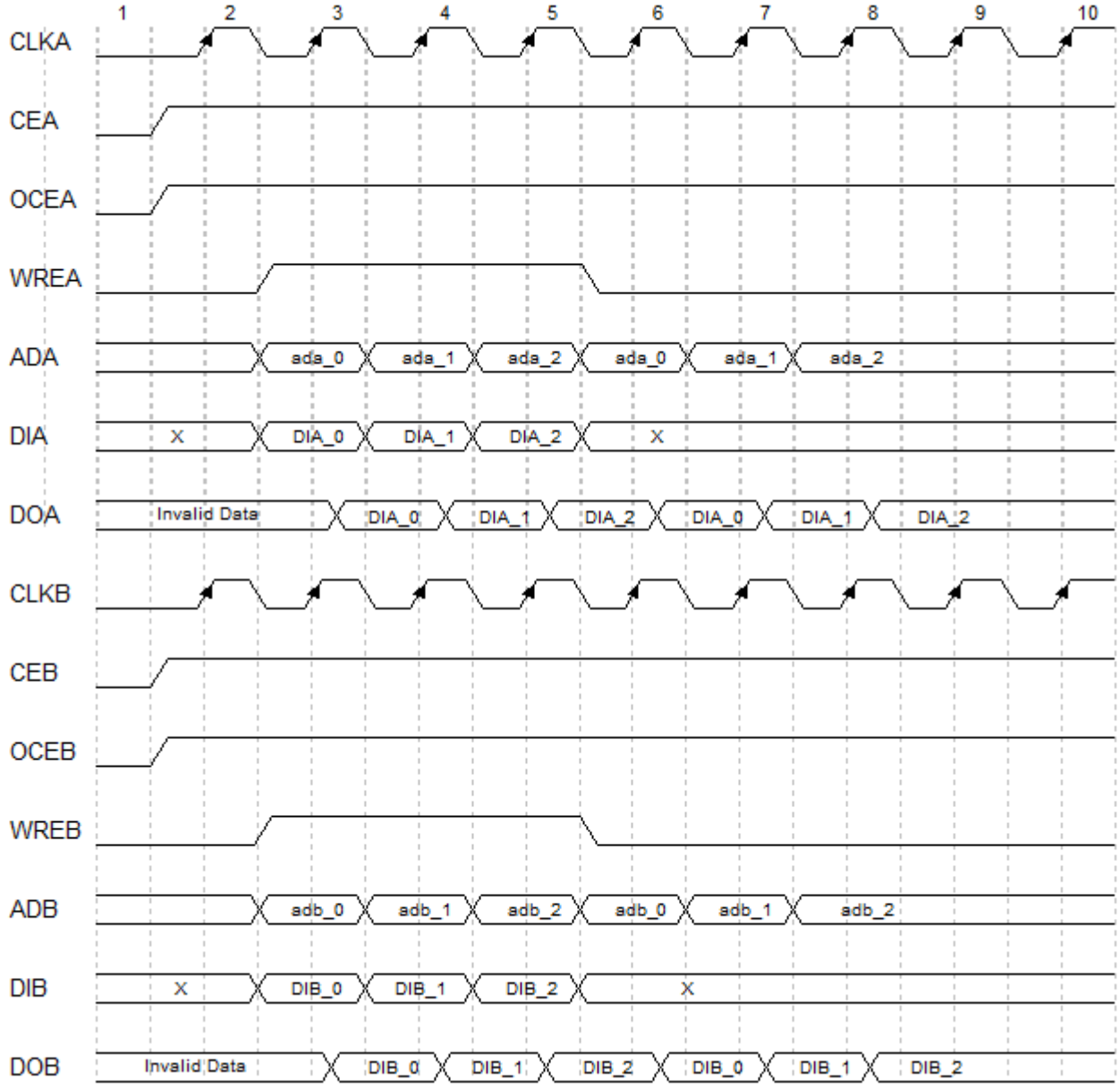


Figure 4-5 Timing Diagram of DP/DPX9 Write-Through（Pipeline Read Mode）

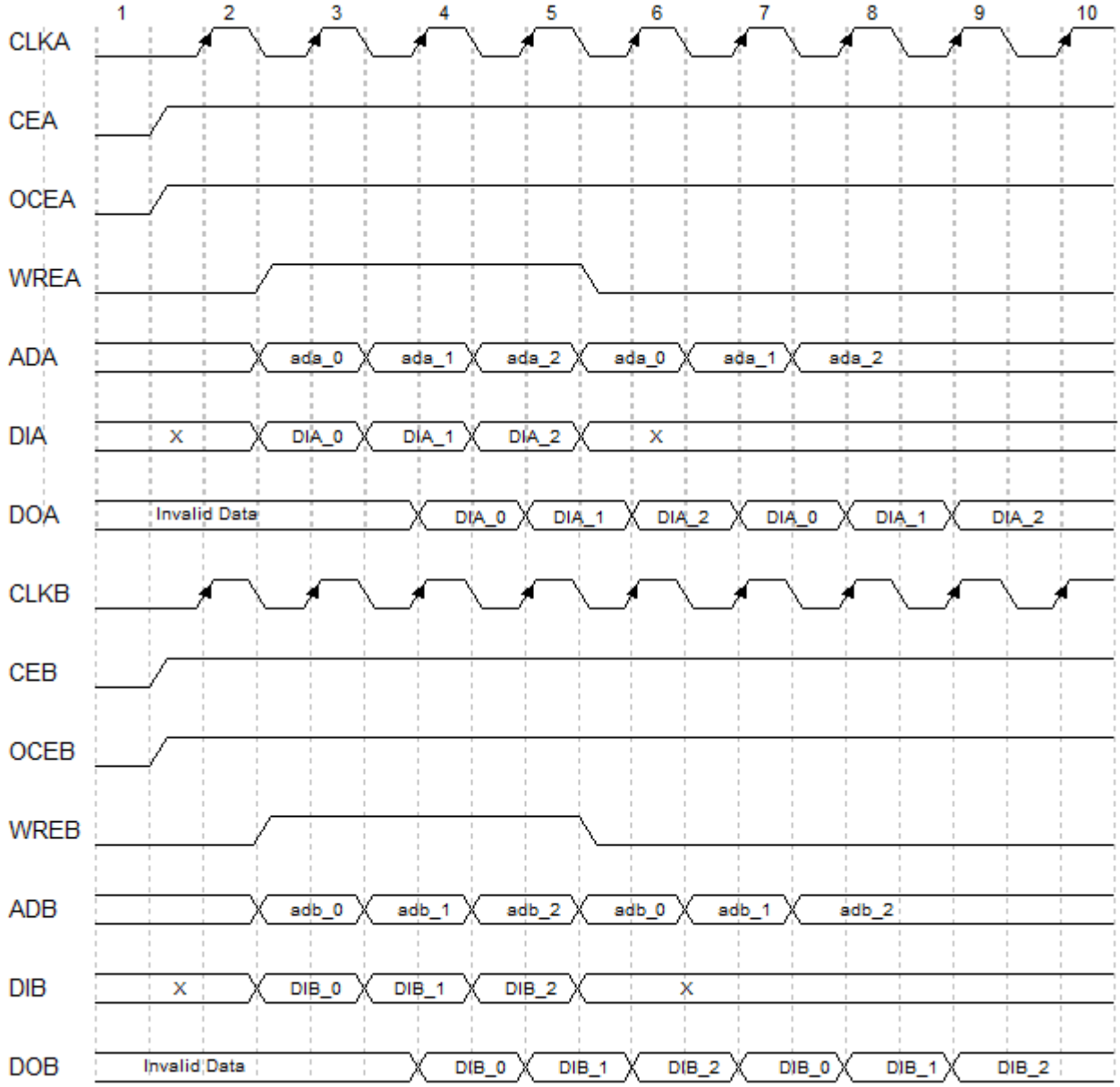


Figure 4-6 Timing Diagram of DP/DPX9 Read-Before-Write（Bypass Read Mode）

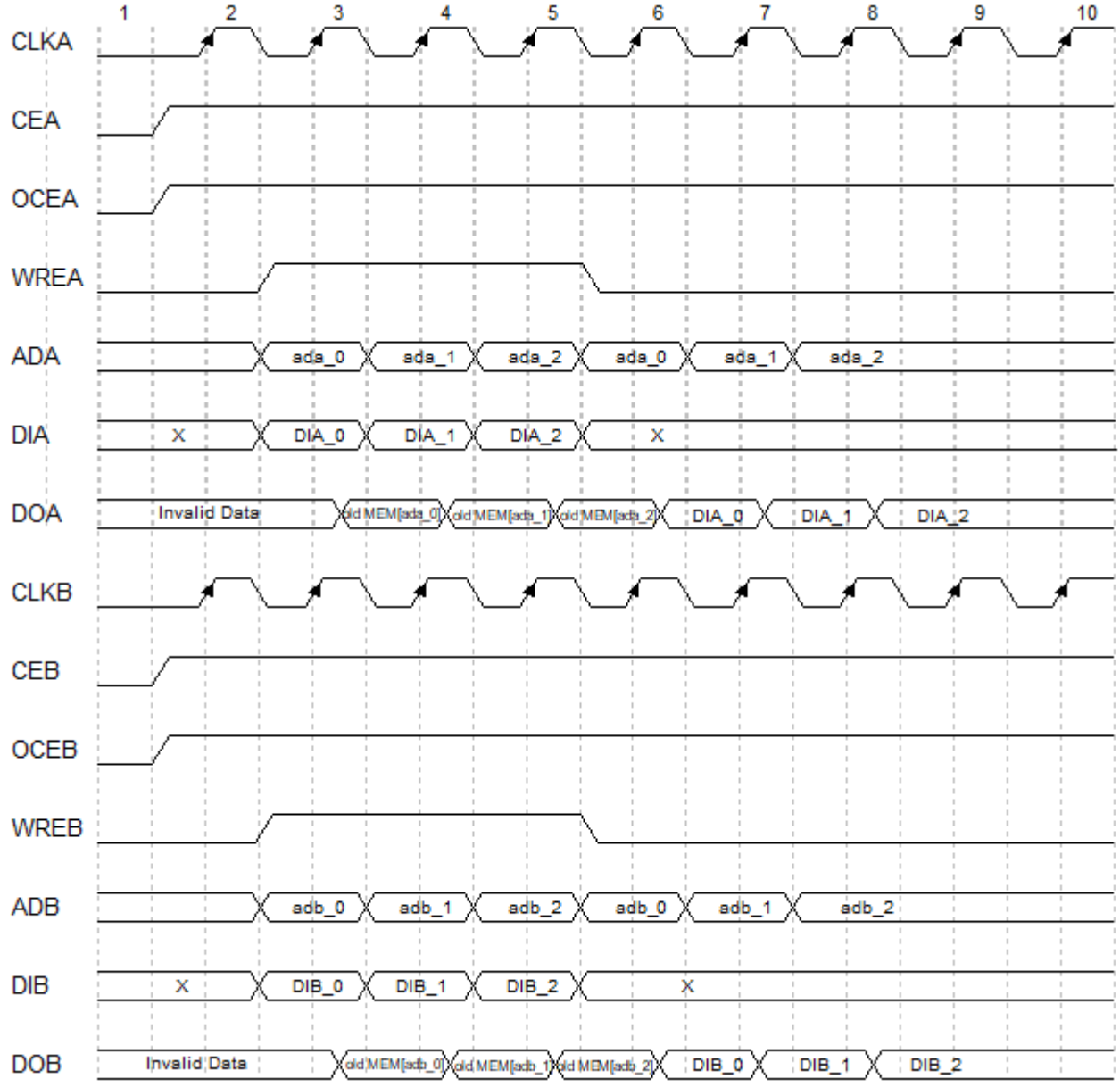
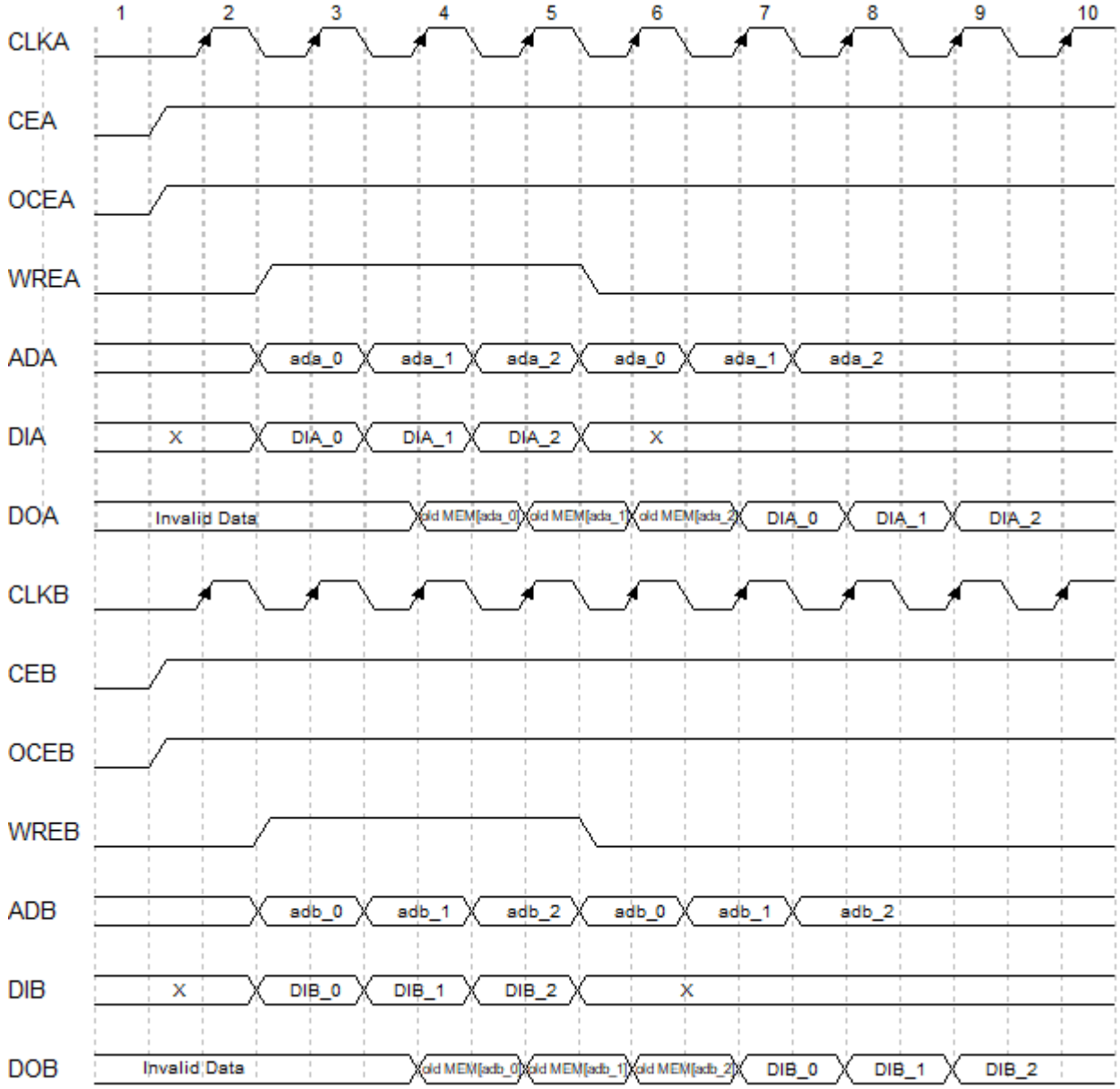


Figure 4-7 Timing Diagram of DP/DPX9 Read-Before-Write（Pipeline Read Mode）



Port Description

Table 4-1 Port Description

Port Name	I/O	Description
DOA[15:0]/DOA[17:0]	Output	A data output
DOB[15:0]/DOB[17:0]	Output	B data output
DIA[15:0]/DIA[17:0]	Input	A data input
DIB[15:0]/DIB[17:0]	Input	B data input
ADA[13:0]	Input	A address input
ADB[13:0]	Input	B address input
WREA	Input	A write enable input 1: write 0: read

Port Name	I/O	Description
WREB	Input	B write enable input 1: write 0: read
CEA	Input	A clock enable signal, active-high.
CEB	Input	B clock enable signal, active-high.
CLKA	Input	A clock input
CLKB	Input	B clock input
RESETA	Input	A reset input, synchronous reset and asynchronous reset supported, active-high
RESETB	Input	B reset input, synchronous reset and asynchronous reset supported, active-high
OCEA	Input	A output clock enable signal used in Pipeline, invalid in Bypass
OCEB	Input	B output clock enable signal used in Pipeline, invalid in Bypass
BLKSEL[2:0]	Input	BSRAM block selection signal for multiple BSRAM memory units cascading to realize capacity expansion

Parameter

Table 4-2 Parameter Description

Name	Type	Value	Default Value	Description
READ_MODE0	Integer	1'b0,1'b1	1'b0	A read mode configuration 1'b0:bypass 1'b1:pipeline
READ_MODE1	Integer	1'b0,1'b1	1'b0	B read mode configuration 1'b0:bypass 1'b1:pipeline
WRITE_MODE0	Integer	2'b00,2'b01,2'b10	2'b00	A write mode configuration 2'b00: Normal 2'b01: Write-through 2'b10: Read-before-write
WRITE_MODE1	Integer	2'b00,2'b01,2'b10	2'b00	B write mode configuration 2'b00: Normal 2'b01: Write-through 2'b10: Read-before-write
BIT_WIDTH_0	Integer	DP:1,2,4,8,16 DPX9:9,18	DP:16 DPX9:18	A data width configuration
BIT_WIDTH_1	Integer	DP:1,2,4,8,16 DPX9:9,18	DP:16 DP:18	B data width configuration
BLK_SEL	Integer	3'b000~3'b111	3'b000	BSRAM block selection parameter is equal to BLKSEL, and the BSRAM is selected. The software will handle expansion automatically when IP Core Generator is used to expand storage capacity.

Name	Type	Value	Default Value	Description
RESET_MODE	String	SYNC,ASYNC	SYNC	Reset mode configuratiom SYNC: synchronized reset ASYNC: asynchronous reset
INIT_RAM_00~ INIT_RAM_3F	Integer	DP:256'h0…0~256'h1…1 DPX9:288'h0…0~288'h1…1	DP:256'h0…0 DPX9:288'h0…0	Used to set up B-SRAM memory unit initialization data

Configuration Relationship

Table 4-3 Data Width and Address Depth Configuration Relationship

Dual Port Mode	BSRAM Capacity	Data width	Address Depth
DP	16K	1	14
		2	13
		4	12
		8	11
		16	10
DPX9	18K	9	11
		18	10

Primitive Instantiation

Example One

Verilog Instantiation:

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

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

Vhdl Instantiation:

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

```
                                RESET_MODE=>"SYNC",
    INIT_RAM_00=>X"0000000000000000000000000000000000000000
00000000000000000000000000",
    INIT_RAM_01=>X"0000000000000000000000000000000000000000
00000000000000000000000000",
    INIT_RAM_3F=>X"0000000000000000000000000000000000000000
00000000000000000000000000"
      )
      PORT  MAP(
          DOA=>doa,
          DOB=>dob,
          CLKA=>clka,
          CLKB=>clkb,
          CEA=>ceb,
          CEB=>ceb,
          OCEA=>ocea,
          OCEB=>oceb,
          RESETA=>reseta,
          RESETB=>resetb,
          WREA=>wrea,
          WREB=>wreb,
          ADA=>ada,
          ADB=>adb,
          BLKSEL=>blksel,
          DIA=>dia,
          DIB=>dib
      );
    Example Two
  Verilog Instantiation:
    DPX9 bram_dpx9_0 (
        .DOA(doa[17:0]),
        .DOB(dob[17:0]),
        .CLKA(clka),
        .OCEA(ocea),
        .CEA(cea),
        .RESETA(reseta),
        .WREA(wrea),
        .CLKB(clkb),
        .OCEB(oceb),
        .CEB(ceb),
        .RESETB(resetb),
        .WREB(wreb),
        .BLKSEL({3'b000}),
        .ADA({ada[9:0], 2'b00,byte_ena[1:0]}),
        .DIA(dia[17:0]),
        .ADB({adb[9:0], 2'b00,byte_enb[1:0]}),
        .DIB(dib[17:0])
    );
    defparam bram_dpx9_0.READ_MODE0 = 1'b1;
    defparam bram_dpx9_0.READ_MODE1 = 1'b1;
```

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

Vhdl Instantiation:

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

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

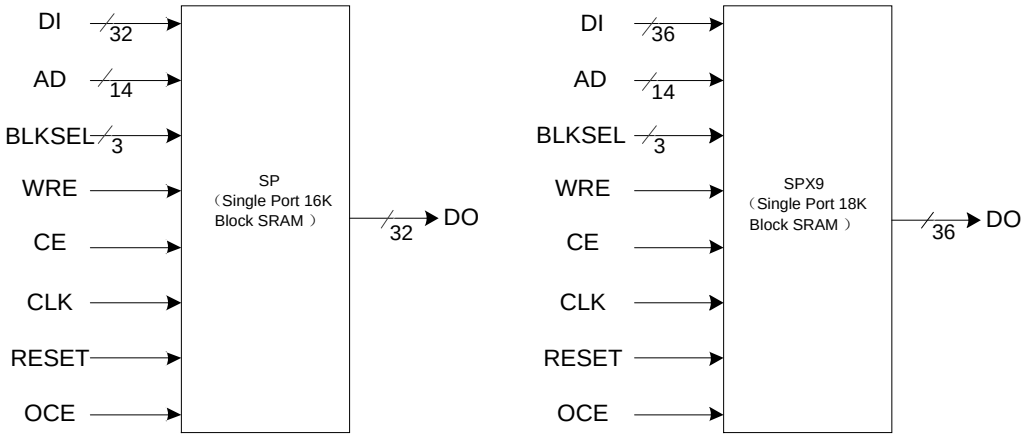
4.2 SP/SPX9

Primitive Name

SP/SPX9 (Single Port 16K Block SRAM/Single Port 18K Block SRAM)

Port Diagram

Figure 4-8 SP/SPX9 Port Diagram



Functional Description

SP/SPX9 works in single port mode with a memory capacity of 16K bit/18K bit. The read/write operation of the single port is controlled by a clock. SP/SPX9 supports two read modes (bypass mode and pipeline mode) and three write modes (normal mode, write-through mode and read-before-write mode).

If SP is configured as 16bit/32bit and SPX9 is configured as 18bit/36bit, the byte enable function of BSRAM can be realized; i.e., the data written to memory by the lower four bit of AD, active-high. AD[0] controls whether DI[7:0]/DI[8:0] writes to memory, AD[1] controls DI[15:8]/DI[17:9] writes to memory, AD[2] controls DI[23:16]/DI[26:18] writes to memory, AD[3] controls DI[31:24]/DI[35:27] writes to memory.

Read mode

READ_MODE enable or disable the output pipeline register. When the output pipeline register is used, the read operation needs extra delay cycle.

Write mode

Normal mode, write-through mode, and read-before-write mode are configured WRITE_MODE.

The internal timing diagram corresponding to different SP/SPX9 modes can be referred to DP/DPX9 A/B timing diagram from Figure 4-2 to Figure 4-7.

Port Description

Table 4-4 Port Description

Port Name	I/O	Description
DO[31:0]/DO[35:0]	Output	Data output
DI[31:0]/DI[35:0]	Input	Data input
AD[13:0]	Input	Address input
WRE	Input	Write enable input

Port Name	I/O	Description
		1: write 0: read
CE	Input	Clock enable input, active-high.
CLK	Input	Clock input
RESET	Input	Reset input, synchronous reset and asynchronous reset supported, active-high
OCE	Input	Output clock enable signal used in Pipeline, invalid in Bypass
BLKSEL[2:0]	Input	BSRAM block selection signal for multiple BSRAM memory units cascading to realize capacity expansion

Parameter

Table 4-5 Parameter Description

Name	Type	Value	Default Value	Description
READ_MODE	Integer	1'b0,1'b1	1'b0	Read mode configuration 1'b0:bypass 1'b1:pipeline
WRITE_MODE	Integer	2'b00,2'b01,2'b10	2'b00	Write mode configuration 2'b00: Normal 2'b01:write-through 2'b10: Read-before-write
BIT_WIDTH	Integer	SP:1,2,4,8,16,32 SPX9:9,18,36	SP:32 SPX9:36	Data width configuration
BLK_SEL	Integer	3'b000~3'b111	3'b000	BSRAM block selection parameter is equal to BLKSEL, and the BSRAM is selected. The software will handle expansion automatically when IP Core Generator is used to expand storage capacity.
RESET_MODE	String	SYNC,ASYNC	SYNC	Reset mode configuratiom SYNC: synchronized reset ASYNC: asynchronous reset
INIT_RAM_00~ INIT_RAM_3F	Integer	SP:256'h0…0~256'h1…1 SPX9:288'h0…0~288'h1…1	SP:256'h0…0 SPX9:288'h0…0	Used to set up B-SRAM memory unit initialization data

Configuration Relationship

Table 4-6 Data Width and Address Depth Configuration Relationship

Single Port Mode	BSRAM Capacity	Data width	Address Depth
SP	16K	1	14
		2	13
		4	12

Single Port Mode	BSRAM Capacity	Data width	Address Depth
		8	11
		16	10
		32	9
SPX9	18K	9	11
		18	10
		36	9

Primitive Instantiation

Example One

Verilog Instantiation:

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

Vhdl Instantiation:

```
COMPONENT SP
        GENERIC(
                BIT_WIDTH:integer:=32;
                READ_MODE:bit:='0';
                WRITE_MODE:bit_vector:="01";
                BLK_SEL:bit_vector:="000";
                RESET_MODE:string:="SYNC";
                INIT_RAM_00:bit_vector:=X"00A000000000000B
00A000000000000B00A000000000000B00A000000000000B ";
                INIT_RAM_01:bit_vector:=X"00A000000000000B
```

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

Example Two

Verilog Instantiation:

```
SPX9 bram_spx9_0 (
    .DO({dout[35:18],dout[17:0]}),
    .CLK(clk),
    .OCE(oce),
    .CE(ce),
    .RESET(reset),
    .WRE(wre),
    .BLKSEL({3'b000}),
```

```
        .AD({ad[9:0], 2'b00, byte_en[1:0]}),
        .DI({{18{1'b0}},din[17:0]})
    );
    defparam bram_spx9_0.READ_MODE = 1'b0;
    defparam bram_spx9_0.WRITE_MODE = 2'b00;
    defparam bram_spx9_0.BIT_WIDTH = 18;
    defparam bram_spx9_0.BLK_SEL = 3'b000;
    defparam bram_spx9_0.RESET_MODE = "SYNC";
    defparam bram_spx9_0.INIT_RAM_00 =
    288'h000000000C000000000000D0000050000C000000000000D000
    0000000C000000000000D0;
    defparam bram_spx9_0.INIT_RAM_01 =
    288'h000000000C000000000000D0000000000C000000003000D000
    0000000C000000000040D0;
    defparam bram_spx9_0.INIT_RAM_3F =
    288'h0000A0000C000000000000D0000000000C000000000000D001
    0000000C000000000000D0;
```

Vhdl Instantiation:

```
    COMPONENT SPX9
        GENERIC(
            BIT_WIDTH:integer:=9;
            READ_MODE:bit:='0';
            WRITE_MODE:bit_vector:="00";
            BLK_SEL : bit_vector:="000";
            RESET_MODE : string:="SYNC";
            INIT_RAM_00:bit_vector:=X"000000000C000000
000000D0000050000C000000000000D0000000000C000000000000D0";
            INIT_RAM_01:bit_vector:=X"000000000C000000
000000D0000000000C000000003000D0000000000C000000000040D0";
            INIT_RAM_3F:bit_vector:=X"0000A0000C000000
000000D0000000000C000000000000D0010000000C000000000000D0"
        );
        PORT(
            DO:OUT std_logic_vector(35 downto 0):=conv_
std_logic_vector(0,36);
            CLK,CE,OCE,RESET,WRE:IN std_logic;
            AD:IN std_logic_vector(13 downto 0);
            DI:IN std_logic_vector(35 downto 0);
            BLKSEL:std_logic_vector(2 downto 0)
        );
    END COMPONENT;
    uut:SPX9
        GENERIC MAP(
            BIT_WIDTH=>9,
            READ_MODE=>'0',
            WRITE_MODE=>"00",
            BLK_SEL=>"000",
            RESET_MODE=>"SYNC",
            INIT_RAM_00=>X"0000000000000000000
0000000000000000000000000000000000000000000000000000",
```

```
                    INIT_RAM_01=>X"0000000000000000000
0000000000000000000000000000000000000000000000000000",
                    INIT_RAM_3F=>X"0000000000000000000
0000000000000000000000000000000000000000000000000000"
        )
        PORT MAP(
            DO=>dout,
            CLK=>clk,
            OCE=>oce,
            CE=>ce,
            RESET=>reset,
            WRE=>wre,
            BLKSEL=>blksel,
            AD=>ad,
            DI=>din
        );
```

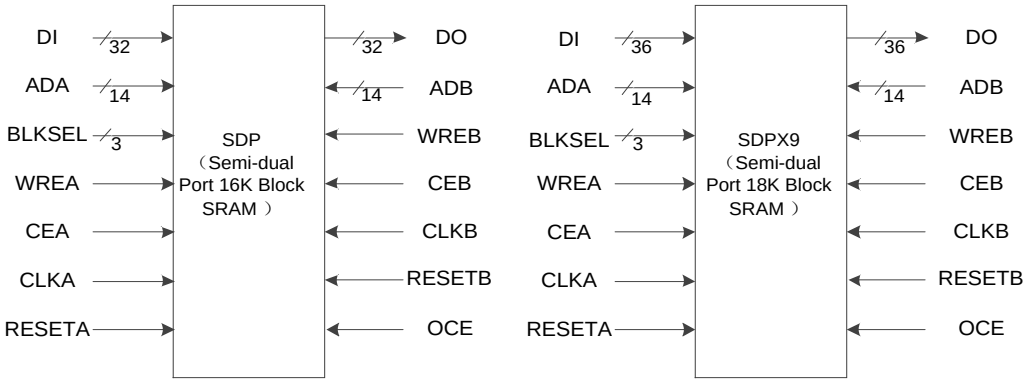
4.3 SDP/SDPX9

Primitive Name

SDP/SDPX9 (Semi Dual Port 16K Block SRAM /Semi Dual Port 18K Block SRAM)

Port Diagram

Figure 4-9 SDP/SDPX9 Port Diagram



Functional Description

SDP/SDPX9 works in semi-dual-port mode with a memory capacity of 16K bit/18K bit. Write operation is at A port and read operation is at B port. SDP/SDPX9 supports two read modes (bypass mode and pipeline mode) and one write modes (normal mode).

If SDP is configured as 16bit/32bit and SDPX9 is configured as 18bit/36bit, the byte enable function of BSRAM can be realized; i.e., the data written to memory by the lower four bit of AD, active-high. AD[0] controls whether DI[7:0]/DI[8:0] writes to memory, AD[1] controls DI[15:8]/DI[17:9] writes to memory, AD[2] controls DI[23:16]/DI[26:18] writes to memory, AD[3] controls DI[31:24]/DI[35:27] writes to memory.

Read mode

READ_MODE enable or disable the output pipeline register. When the output pipeline register is used, the read operation needs extra delay cycle.

Write mode

SDP/SDPX9 port A is for write operation, and port B is for read operation, supporting normal mode.

The corresponding internal timing diagram of different modes are shown in Figure 4-10 and Figure 4-11.

Figure 4-10 Timing Diagram of SDP/SDPX9 Normal（Bypass Read Mode）

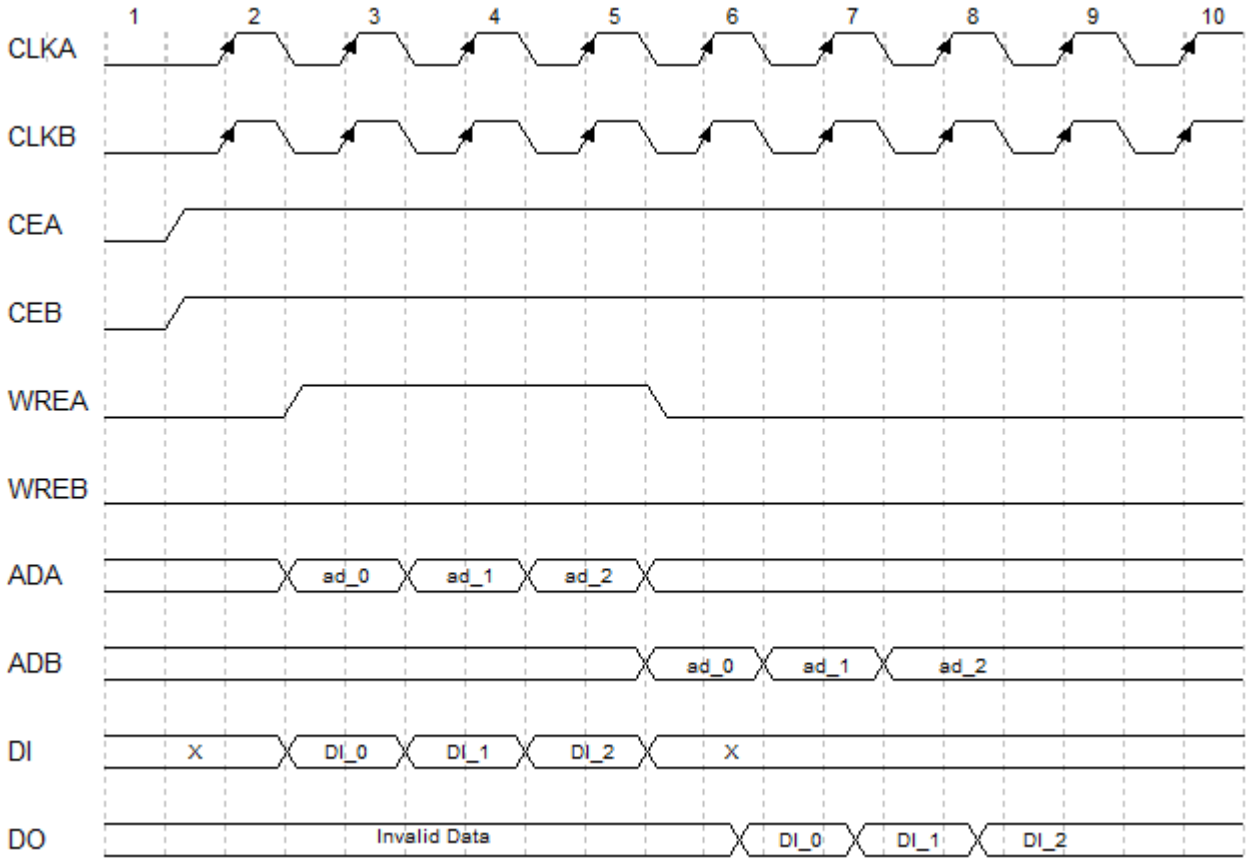
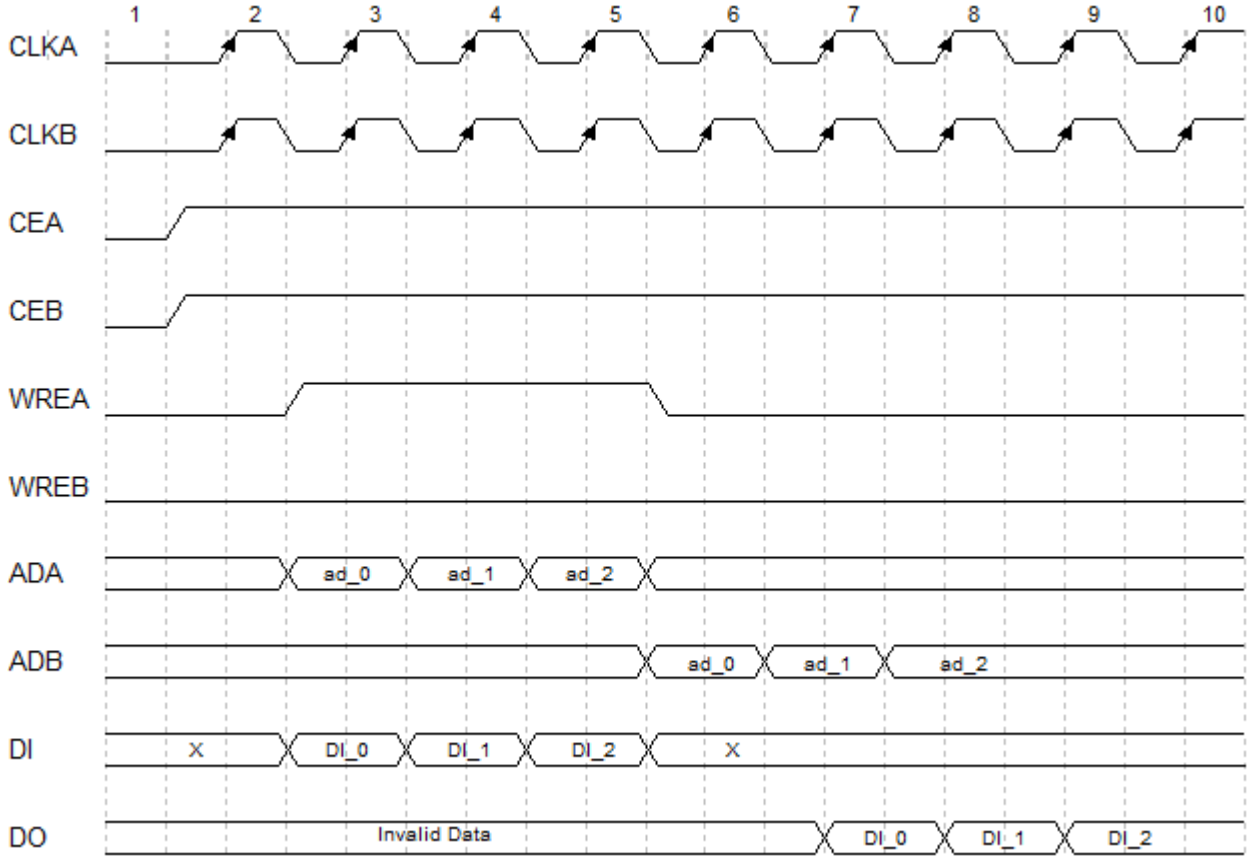


Figure 4-11 Timing Diagram of SDP/SDPX9 Normal（Pipeline Read Mode）



Port Description

Table 4-7 Port Description

Port Name	I/O	Description
DO[31:0]/DO[35:0]	Output	Data output
DI[31:0]/DI[35:0]	Input	Data input
ADA[13:0]	Input	A address input
ADB[13:0]	Input	B address input
WREA	Input	A write enable input (generally configured as 1) 1: write 0: read
WREB	Input	B write enable input (generally configured as 0) 1: write 0: read
CEA	Input	A clock enable signal, active-high.
CEB	Input	B clock enable signal, active-high.
CLKA	Input	A clock input
CLKB	Input	B clock input
RESETA	Input	A reset input, synchronous reset and asynchronous reset supported, active-high

Port Name	I/O	Description
RESETB	Input	B reset input, synchronous reset and asynchronous reset supported, active-high
OCE	Input	Output clock enable signal used in Pipeline, invalid in Bypass
BLKSEL[2:0]	Input	BSRAM block selection signal for multiple BSRAM memory units cascading to realize capacity expansion

Parameter

Table 4-8 Parameter Description

Name	Type	Value	Default Value	Description
READ_MODE	Integer	1'b0,1'b1	1'b0	Read mode configuration • 1'b0:bypass • 1'b1:pipeline
BIT_WIDTH_0	Integer	SDP:1,2,4,8,16,32 SDPX9:9,18,36	SDP:32 SDPX9:36	A data width configuration
BIT_WIDTH_1	Integer	SDP:1,2,4,8,16,32 SDPX9:9,18,36	SDP:32 SDPX9:36	B data width configuration
BLK_SEL	Integer	3'b000~3'b111	3'b000	BSRAM block selection parameter is equal to BLKSEL, and the BSRAM is selected. The software will handle expansion automatically when IP Core Generator is used to expand storage capacity.
RESET_MODE	String	SYNC,ASYNC	SYNC	Reset mode configuratiom SYNC: synchronized reset ASYNC: asynchronous reset
INIT_RAM_00~ INIT_RAM_3F	Integer	SDP:256'h0…0~256'h1…1 SDPX9:288'h0…0~288'h1…1	SDP:256'h0…0 SDPX9:288'h0…0	Used to set up B-SRAM memory unit initialization data

Configuration Relationship

Table 4-9 Data Width and Address Depth Configuration Relationship

Semi-dual port mode	BSRAM Capacity	Data width	Address Depth
SDP	16K	1	14
		2	13
		4	12
		8	11
		16	10
		32	9
SDPX9	18K	9	11

Semi-dual port mode	BSRAM Capacity	Data width	Address Depth
		18	10
		36	9

Primitive Instantiation

Example One

Verilog Instantiation:

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

Vhdl Instantiation:

```
 COMPONENT  SDP
          GENERIC(
                    BIT_WIDTH_0:integer:=16;
                    BIT_WIDTH_1:integer:=16;
                    READ_MODE:bit:='0';
                    BLK_SEL:bit_vector:="000";
                    RESET_MODE:string:="SYNC";
                    INIT_RAM_00:bit_vector:=X"00A000000000000
B00A000000000000B00A000000000000B00A000000000000B";
                    INIT_RAM_01:bit_vector:=X"00A000000000000
B00A000000000000B00A000000000000B00A000000000000B";
                    INIT_RAM_3F:bit_vector:=X"00A000000000000
B00A000000000000B00A000000000000B00A000000000000B"
```

```
                    );
                    PORT(
                            DO:OUT std_logic_vector(31 downto 0):=conv_
std_logic_vector(0,32);
                            CLKA,CLKB,CEA,CEB,OCE,RESETA,RESETB,
WREA,WREB:IN std_logic;
                            ADA,ADB:IN std_logic_vector(13 downto 0);
                            BLKSEL:IN std_logic_vector(2 downto 0);
                            DI:IN std_logic_vector(31 downto 0)
                     );
        END COMPONENT;
        uut:SDP
            GENERIC MAP(
                            BIT_WIDTH_0=>16,
                            BIT_WIDTH_1=>16,
                            READ_MODE=>'0',
                            BLK_SEL=>"000",
                            RESET_MODE=>"SYNC",
                            INIT_RAM_00=>X"00A000000000000B00A00
0000000000B00A000000000000B00A000000000000B",
                            INIT_RAM_01=>X"00A000000000000B00A00
0000000000B00A000000000000B00A000000000000B",
                            INIT_RAM_3F=>X"00A000000000000B00A00
0000000000B00A000000000000B00A000000000000B"
                            )
                PORT MAP(
                    DO=>dout,
                    CLKA=>clka,
                    CEA=>cea,
                    RESETA=>reseta,
                    WREA=>wrea，
                    CLKB=>clkb,
                    CEB=>ceb,
                    RESETB=>resetb,
                    WREB=>wreb,
                    OCE=>oce,
                    BLKSEL=>blksel,
                    ADA=>ada,
                    DI=>din,
                    ADB=>adb
                );
```

Example Two

Verilog Instantiation:

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

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

Vhdl Instantiation:

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

```
uut:SDP
    GENERIC  MAP(
                    BIT_WIDTH_0=>18,
                    BIT_WIDTH_1=>18,
                    READ_MODE=>'0',
                    BLK_SEL=>"000",
                    RESET_MODE=>"SYNC",
                    INIT_RAM_00=>X"000000000C000000000000D00
00050000C000000000000D0000000000C000000000000D0",
                    INIT_RAM_01=>X"000000000C000000000000D00
00000000C000000003000D0000000000C000000000040D0",
                    INIT_RAM_3F=>X"0000A0000C000000000000D00
00000000C000000000000D0010000000C000000000000D0"
    )
    PORT  MAP(
        DO=>dout,
        CLKA=>clka,
        CEA=>cea,
        RESETA=>reseta,
        WREA=>wrea，
        CLKB=>clkb,
        CEB=>ceb,
        RESETB=>resetb,
        WREB=>wreb,
        OCE=>oce,
        BLKSEL=>blksel,
        ADA=>ada,
        DI=>din,
        ADB=>adb
    );
```

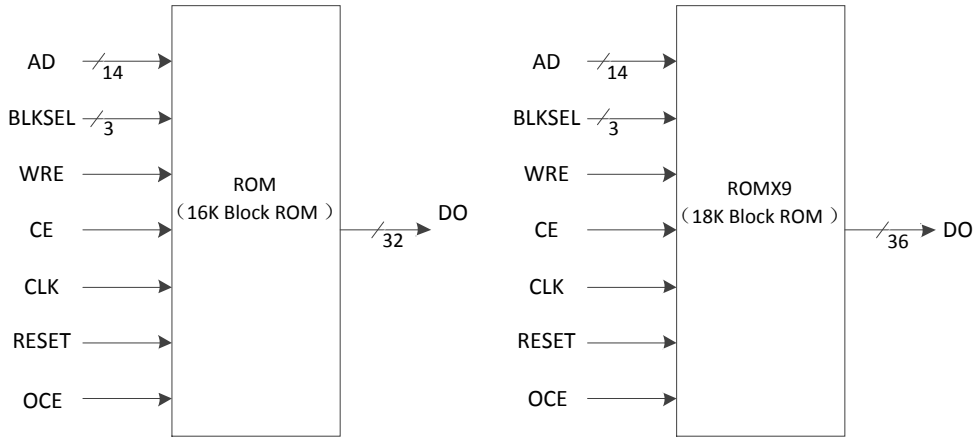
4.4 ROM/ROMX9

Primitive Name

ROM/ROMX9(16K/18K Block ROM) is 16K/18K block read-only memory.

Port Diagram

Figure 4-12 ROM/ROMX9 Port Diagram



Functional Description

ROM/ROMX9 works in read only mode with the memory capacity of 16K bit/18K bit, which support 2 modes (bypass mode and pipeline mode).

Read mode

READ_MODE enable or disable the output pipeline register. When the output pipeline register is used, the read operation needs extra delay cycle. For internal timing diagram corresponding to different ROM/ROMX9 reading modes, you can refer to DP/DPX9 timing diagram Figure 4-2 and Figure 4-3. (WRE=0).

Port Description

Table 4-10 Port Description

Port Name	I/O	Description
DO[31:0]/DO[35:0]	Output	Data output
AD[13:0]	Input	Address input
WRE	Input	Write enable input (generally configured as 0)
CE	Input	Clock enable input, active-high.
CLK	Input	Clock input
RESET	Input	Reset input, synchronous reset and asynchronous reset supported, active-high
OCE	Input	Output clock enable signal used in Pipeline, invalid in Bypass
BLKSEL[2:0]	Input	BSRAM block selection signal for multiple BSRAM memory units cascading to realize capacity expansion

Parameter

Table 4-11 Parameter Description

Name	Type	Value	Default Value	Description
READ_MODE	Integer	1'b0,1'b1	1'b0	Read mode configuration 1'b0:bypass 1'b1:pipeline
BIT_WIDTH	Integer	ROM:1,2,4,8,16,32 ROMX9:9,18,36	ROM:32 ROMX9:36	Data width configuration
BLK_SEL	Integer	3'b000~3'b111	3'b000	BSRAM block selection parameter is equal to BLKSEL, and the BSRAM is selected. The software will handle expansion automatically when IP Core Generator is used to expand storage capacity.
RESET_MODE	String	SYNC,ASYNC	SYNC	Reset mode configuratiom SYNC: synchronized reset ASYNC: asynchronous reset
INIT_RAM_00~ INIT_RAM_3F	Integer	ROM:256'h0…0~256'h1…1 ROMX9:288'h0…0~288'h1…1	ROM:256'h0…0 ROMX9:288'h0…0	Used to set up B-SRAM memory unit initialization data

Configuration Relationship

Table 4-12 Configuration Relationship

Read Only	BSRAM Capacity	Data width	Address Depth
ROM	16K	1	14
		2	13
		4	12
		8	11
		16	10
		32	9
ROMX9	18K	9	11
		18	10
		36	9

Primitive Instantiation

Example One

Verilog Instantiation:

```
ROM bram_rom_0 (
    .DO({dout[31:8],dout[7:0]}),
```

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

Vhdl Instantiation:

```
    COMPONENT ROM
        GENERIC(
            BIT_WIDTH:integer:=1;
            READ_MODE:bit:='0';
            BLK_SEL:bit_vector:="000";
            RESET_MODE:string:="SYNC";
            INIT_RAM_00:bit_vector:=X"9C23645D0F78986FF
C3E36E141541B95C19F2F7164085E631A819860D8FF0000";
            INIT_RAM_01:bit_vector:=X"000000000000000000
0000000000000000000000000000000000FFFFFFBDCF"
        );
        PORT(
            DO:OUT std_logic_vector(31 downto 0):=conv_std
_logic_vector(0,32);
            CLK,CE,OCE,RESET,WRE:IN std_logic;
            BLKSEL:IN std_logic_vector(2 downto 0);
            AD:IN std_logic_vector(13 downto 0)
        );
    END COMPONENT;
    uut:ROM
        GENERIC MAP(
            BIT_WIDTH=>1,
            READ_MODE=>'0',
            BLK_SEL=>"000",
            RESET_MODE=>"SYNC",
            INIT_RAM_00=>X"9C23645D0F78986FFC3E36
E141541B95C19F2F7164085E631A819860D8FF0000",
            INIT_RAM_01=>X"000000000000000000000000
0000000000000000000000000000000FFFFFFBDCF "
        )
        PORT MAP(
```

```
                DO=>do,
                AD=>ad,
                CLK=>clk,
                CE=>ce,
                OCE=>oce,
                RESET=>reset,
                WRE=>wre,
                BLKSEL=>blksel
        );
```

Example Two

Verilog Instantiation:

```
ROMX9 bram_romx9_0 (
    .DO({dout[35:9],dout[8:0]}),
    .CLK(clk),
    .OCE(oce),
    .CE(ce),
    .RESET(reset),
    .WRE(wre),
    .BLKSEL({3'b000}),
    .AD({ad[10:0],3'b000})
);
defparam bram_romx9_0.READ_MODE = 1'b0;
defparam bram_romx9_0.BIT_WIDTH = 9;
defparam bram_romx9_0.BLK_SEL = 3'b000;
defparam bram_romx9_0.RESET_MODE = "SYNC";
defparam bram_romx9_0.INIT_RAM_00 =
288'hCE08CC85D07DE1316FFE0F86DE1A09523795E0E7E5E71B2
020BC630D6053160EC7FC0000;
defparam bram_romx9_0.INIT_RAM_01 =
288'h000000000000000000000000000000000000000000000000000
0000001FFFFFFF7ACF;
```

Vhdl Instantiation:

```
COMPONENT ROMX9
        GENERIC(
                BIT_WIDTH:integer:=9;
                READ_MODE:bit:='0';
                BLK_SEL:bit_vector:="000";
                RESET_MODE:string:="SYNC";
                INIT_RAM_00:bit_vector:=X"CE08CC85D07DE131
6FFE0F86DE1A09523795E0E7E5E71B2020BC630D6053160EC7FC000
0";
                INIT_RAM_01:bit_vector:=X"000000000000000000
0000000000000000000000000000000000000001FFFFFFF7ACF"
        );
        PORT(
                DO:OUT  std_logic_vector(35  downto  0):=conv_std
_logic_vector(0,36);
                CLK,CE,OCE,RESET,WRE:IN  std_logic;
                BLKSEL:IN  std_logic_vector(2  downto  0);
```

```
                        AD:IN  std_logic_vector(13  downto  0)
            );
    END COMPONENT;
    Uut:ROMX9
        GENERIC MAP(
                        BIT_WIDTH=>9,
                        READ_MODE=>'0',
                        BLK_SEL=>"000",
                        RESET_MODE=>"SYNC",
                        INIT_RAM_00=>X"CE08CC85D07DE1316F
FE0F86DE1A09523795E0E7E5E71B2020BC630D6053160EC7FC0000",
                        INIT_RAM_01=>X"0000000000000000000
0000000000000000000000000000000000000001FFFFFFF7ACF"
        )
        PORT MAP(
            DO=>do,
            AD=>ad,
            CLK=>clk,
            CE=>ce,
            OCE=>oce,
            RESET=>reset,
            WRE=>wre,
            BLKSEL=>blksel
        );
```

5 DSP

The digital signal processor (DSP) includes a Pre-Adder, a MULT, and an ALU54D.

Devices supported: GW1N-2, GW1N-2B, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

5.1 Pre-adder

The pre-adder performs the functions of pre-adding, pre-subtracting, and shifting. According to the bit width, a pre-adder includes 9-bit wide PADD9 and 18-bit wide PADD18.

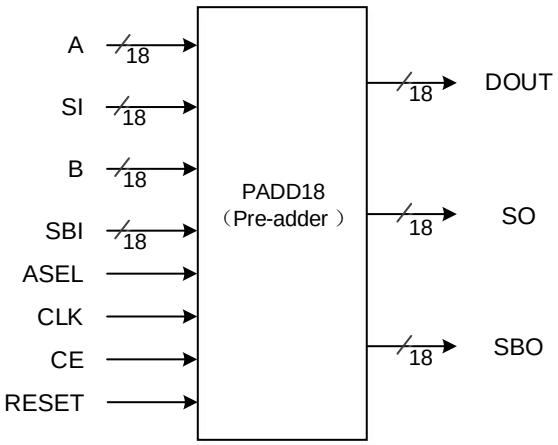
5.1.1 PADD18

Primitive Introduction

The 18-bit pre-adder (PADD18) is an 18-bit pre-adder that performs the function of pre-addition, pre-subtraction, or shift.

Block Diagram

Figure 5-1 PADD18 Blcok Diagram



Port Description

Table 5-1 Port Description

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

Attribute Description

Table 5-2 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

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

Vhdl Instantiation:

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

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

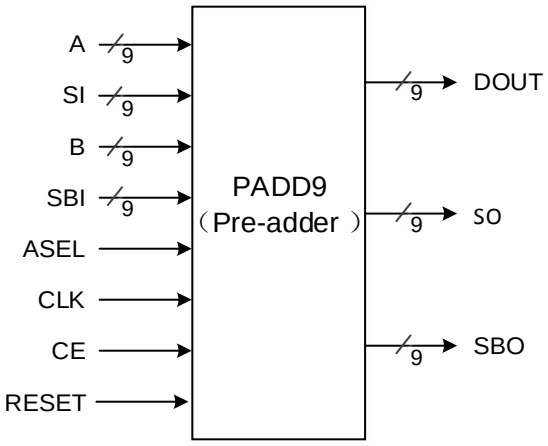
5.1.2 PADD9

Primitive Introduction

9-bit pre-adder (PADD9) is a 9-bit pre-adder that performs the function of pre-addition, pre-subtraction or shift.

Block Diagram

Figure 5-2 PADD9 Blcok Diagram



Port Description

Table 5-3 Port Description

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

Attribute Description

Table 5-4 Attribute Introduction

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

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

5.2 Multiplier

Multiplier is a DSP multiplier. Its input signals are MDIA and MDIB, and output signal is MOUT. Multiplication: $DOUT = A * B$.

Based on bit width, the multiplier can be configured as 9x9, 18x18 and 36x36 multipliers, which corresponds to MULT9X9, MULT18X18, and MULT36X36 primitives.

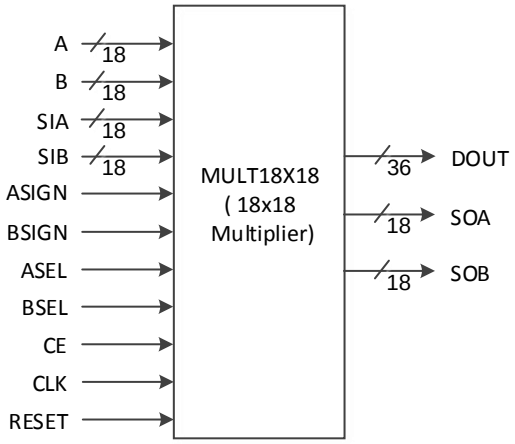
5.2.1 MULT18X18

Primitive Introduction

MULT18X18 supports 18-bit multiplication.

Block Diagram

Figure 5-3 MULT18X18 Block Diagram



Port Description

Table 5-5 Port Description

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

Attribute Description

Table 5-6 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
AREG	1'b0,1'b1	1'b0	Input A(SIA or A) register can be bypassed 1'b0:bypass mode 1'b1:registered mode
BREG	1'b0,1'b1	1'b0	Input B(SIB or B) register can be bypassed 1'b0:bypass mode 'b1:registered mode
OUT_REG	1'b0,1'b1	1'b0	Output register can be bypassed 1'b0:bypass mode

Attribute Name	Permitted Values	Default	Description
			1’b1:registered mode
PIPE_REG	1’b0,1’b1	1’b0	Pipeline register can be bypassed 1’b0:bypass mode 1’b1:registered mode
ASIGN_REG	1’b0,1’b1	1’b0	ASIGN input register can be bypassed 1’b0:bypass mode 1’b1:registered mode
BSIGN_REG	1’b0,1’b1	1’b0	BSIGN input register can be bypassed 1’b0:bypass mode 1’b1:registered mode
SOA_REG	1’b0,1’b1	1’b0	SOA register can be bypassed 1’b0:bypass mode 1’b1:registered mode
MULT_RESET_MODE	SYNC,ASYNC	SYNC	Reset mode config, synchronous or asynchronous

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

```
COMPONENT MULT18X18
    GENERIC (AREG:bit:='0';
                    BREG:bit:='0';
                    OUT_REG:bit:='0';
                    PIPE_REG:bit:='0';
                    ASIGN_REG:bit:='0';
```

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

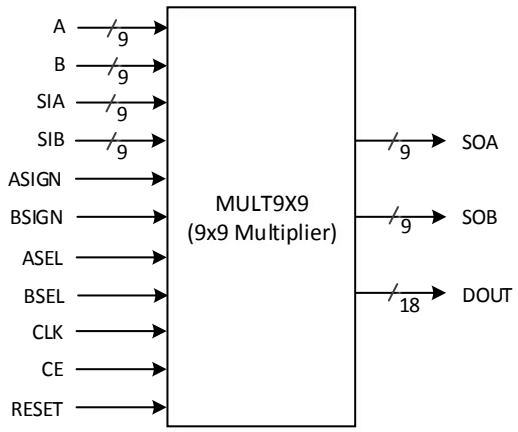
5.2.2 MULT9X9

Primitive Introduction

MULT9X9 supports 9-bit multiplication.

Block Diagram

Figure 5-4 MULT9X9 Block Diagram



Port Description

Table 5-7 Port Description

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

Attribute Description

Table 5-8 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
AREG	1'b0,1'b1	1'b0	Input A(SIA or A) register can be bypassed 1'b0:bypass mode 1'b1:registered mode
BREG	1'b0,1'b1	1'b0	Input B(SIB or B) register can be bypassed

Attribute Name	Permitted Values	Default	Description
			1’b0:bypass mode 1’b1:registered mode
OUT_REG	1’b0,1’b1	1’b0	Output register can be bypassed 1’b0:bypass mode 1’b1:registered mode
PIPE_REG	1’b0,1’b1	1’b0	Pipeline register can be bypassed 1’b0:bypass mode 1’b1:registered mode
ASIGN_REG	1’b0,1’b1	1’b0	ASIGN input register can be bypassed 1’b0:bypass mode 1’b1:registered mode
BSIGN_REG	1’b0,1’b1	1’b0	BSIGN input register can be bypassed 1’b0:bypass mode 1’b1:registered mode
SOA_REG	1’b0,1’b1	1’b0	SOA register can be bypassed 1’b0:bypass mode 1’b1:registered mode
MULT_RESET_MODE	SYNC，ASYNC	SYNC	Reset mode config, synchronous or asynchronous

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

```
COMPONENT MULT9X9
    GENERIC (AREG:bit:='0';
                    BREG:bit:='0';
```

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

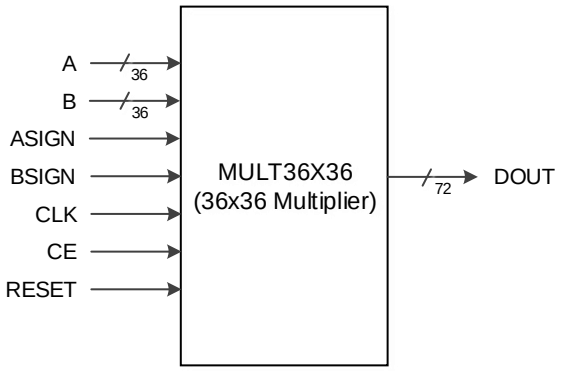
5.2.3 MULT36X36

Primitive Introduction

MULT36X36 supports 36-bit multiplication.

Block Diagram

Figure 5-5 MULT36X36 Block Diagram



Port Description

Table 5-9 Port Description

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

Attribute Description

Table 5-10 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
AREG	1’b0,1’b1	1’b0	Input A(A) register can be bypassed. 1’b0:bypass mode 1’b1:registered mode
BREG	1’b0,1’b1	1’b0	Input B(B) register can be bypassed. 1’b0:bypass mode 1’b1:registered mode
OUT0_REG	1’b0,1’b1	1’b0	Thefirst outputregister can be bypassed 1’b0:bypass mode 1’b1:registered mode
OUT1_REG	1’b0,1’b1	1’b0	The second output register can be bypassed 1’b0:bypass mode 1’b1:registered mode
PIPE_REG	1’b0,1’b1	1’b0	Pipeline register can be bypassed

Attribute Name	Permitted Values	Default	Description
			1’b0:bypass mode 1’b1:registered mode
ASIGN_REG	1’b0,1’b1	1’b0	ASIGN input register can be bypassed 1’b0:bypass mode 1’b1:registered mode
BSIGN_REG	1’b0,1’b1	1’b0	BSIGN input register can be bypassed 1’b0:bypass mode 1’b1:registered mode
MULT_RESET_MODE	SYNC,ASYNC	SYNC	Reset mode config,synchronous or asynchronous

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

```
COMPONENT MULT36X36
    GENERIC (AREG:bit:='0';
                    BREG:bit:='0';
                    OUT0_REG:bit:='0';
                    OUT1_REG:bit:='0';
                    PIPE_REG:bit:='0';
                    ASIGN_REG:bit:='0';
                    BSIGN_REG:bit:='0';
                    MULT_RESET_MODE:string:="SYNC"
          );
    PORT(
     A:IN std_logic_vector(35 downto 0);
            B:IN std_logic_vector(35 downto 0);
            ASIGN:IN std_logic;
            BSIGN:IN std_logic;
            CE:IN std_logic;
            CLK:IN std_logic;
```

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

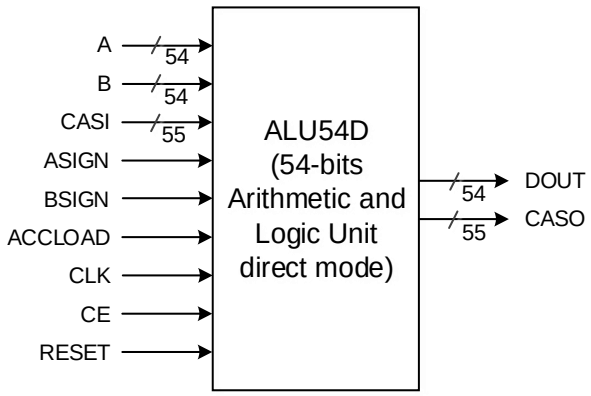
5.3 ALU54D

Primitive Introduction

54-bit Arithmetic Logic Unit (ALU54D) is a 54-bit arithmetic logic unit.

Block Diagram

Figure 5-6 ALU54D Blcok Diagram



Port Description

Table 5-11 Port Description

Port Name	I/O	Description
A[53:0]	Input	54-bit Data Input A

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

Attribute Description

Table 5-12 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
AREG	1'b0,1'b1	1'b0	Input A(A) registers can be bypassed 1'b0:bypass mode 1'b1: registered mode
BREG	1'b0,1'b1	1'b0	Input B(B) registers can be bypassed 1'b0:bypass mode 1'b1: registered mode
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN input register can be bypassed 1'b0:bypass mode 1'b1:registered mode
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN input register can be bypassed 1'b0:bypass mode 1'b1:registered mode
ACCLOAD_REG	1'b0,1'b1	1'b0	Stage register of ACCLOAD can be bypassed 1'b0:bypass mode 1'b1:registered mode
OUT_REG	1'b0,1'b1	1'b0	The output registers can be bypassed. 1'b0:bypass mode 1'b1: registered mode
B_ADD_SUB	1'b0,1'b1	1'b0	B_OUT ADD/SUB Selection 1'b0: add 1'b1: sub
C_ADD_SUB	1'b0,1'b1	1'b0	C_OUT ADD/SUB Selection 1'b0: add 1'b1: sub
ALUMODE	0,1,2	0	ALU54 Operation Mode and Unit Input Selection 0:ACC/0 +/- B +/- A; 1:ACC/0 +/- B + CASI; 2:A +/- B + CASI;
ALU_RESET_MODE	SYNC,ASYNC	SYNC	Reset mode config, synchronous or asynchronous

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

```
COMPONENT ALU54D
    GENERIC (AREG:bit:='0';
                        BREG:bit:='0';
                        ASIGN_REG:bit:='0';
                        BSIGN_REG:bit:='0';
                        ACCLOAD_REG:bit:='0';
                        OUT_REG:bit:='0';
                        B_ADD_SUB:bit:='0';
                        C_ADD_SUB:bit:='0';
                        ALUD_MODE:integer:=0;
                        ALU_RESET_MODE:string:="SYNC"
          );
    PORT(
      A:IN std_logic_vector(53 downto 0);
                B:IN std_logic_vector(53 downto 0);
                ASIGN:IN std_logic;
                BSIGN:IN std_logic;
                CE:IN std_logic;
                CLK:IN std_logic;
                RESET:IN std_logic;
                ACCLOAD:IN std_logic;
                CASI:IN std_logic_vector(54 downto 0);
                CASO:OUT std_logic_vector(54 downto 0);
```

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

5.4 MULTALU

Multiplier with ALU (MULTALU) is a multiplier with ALU function, including 36X18 bits and 18X18 bits, corresponding to the original MULTALU36X18 and MULTALU18X18 respectively.

MULTALU36X18 supports three arithmetic modes:

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

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

$$DOUT = A*B + CASI$$

MULTALU18X18 supports three arithmetic modes:

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

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

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

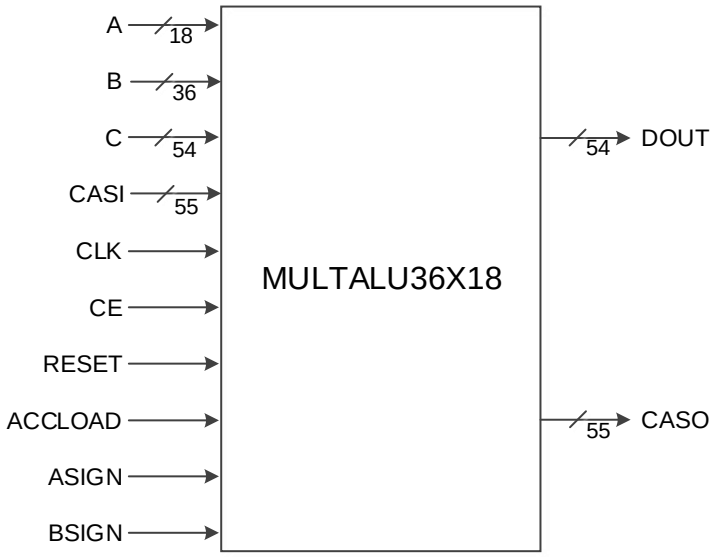
5.4.1 MULTALU36X18

Primitive Introduction

36x18 Multiplier with ALU (MULTALU36X18) is a 36x18 multiplier with ALU function.

Block Diagram

Figure 5-7 MULTALU36X18 Block Diagram



Port Description

Table 5-13 Port Description

Port Name	I/O	Description
A[17:0]	Input	18-bit Data Input A
B[35:0]	Input	36-bit Data Input B
C[53:0]	Input	54-bit Reload Data Input
CASI[54:0]	Input	55-bit Data Carry Input
ASIGN	Input	Input A Sign Bit
BSIGN	Input	Input B Sign Bit
CLK	Input	Clock input
CE	Input	Clock Enable
RESET	Input	Reset Input
ACCLOAD	Input	Accumulator Reload Mode Selection
DOUT[53:0]	Output	Data Output
CASO[54:0]	Output	55-bit Data Carry Output

Attribute Description

Table 5-14 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
AREG	1'b0,1'b1	1'b0	Input A(A)register can be bypassed 1'b0:bypass mode 1'b1:registered mode
BREG	1'b0,1'b1	1'b0	Input B(B)register can be bypassed 1'b0:bypass mode 1'b1:registered mode
CREG	1'b0,1'b1	1'b0	Input C(C) register can be bypassed

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

Primitive Instantiation

Verilog Instantiation:

```
MULTALU36X18 multalu36x18_inst(
  .CASO(caso[54:0]),
  .DOUT(dout[53:0]),
  .ASIGN(asign),
  .BSIGN(bsign),
  .CE(ce),
  .CLK(clk),
  .RESET(reset),
  .CASI(casi[54:0]),
  .ACCLOAD(accload),
  .A(a[17:0]),
  .B(b[35:0]),
  .C(c[53:0])
);
 defparam multalu36x18_inst.AREG = 1'b1;
 defparam multalu36x18_inst.BREG = 1'b0;
```

```
defparam multalu36x18_inst.CREG = 1'b0;
defparam multalu36x18_inst.OUT_REG = 1'b1;
defparam multalu36x18_inst.PIPE_REG = 1'b0;
defparam multalu36x18_inst.ASIGN_REG = 1'b0;
defparam multalu36x18_inst.BSIGN_REG = 1'b0;
defparam multalu36x18_inst.ACCLOAD_REG0 = 1'b1;
defparam multalu36x18_inst.ACCLOAD_REG1 = 1'b0;
defparam multalu36x18_inst.SOA_REG = 1'b0;
defparam multalu36x18_inst.MULT_RESET_MODE = "SYNC";
defparam multalu36x18_inst.MULTALU36X18_MODE = 0;
defparam multalu36x18_inst.C_ADD_SUB = 1'b0;
```

Vhdl Instantiation:

```
COMPONENT MULTALU36X18
  GENERIC (AREG:bit:='0';
                  BREG:bit:='0';
                  CREG:bit:='0';
                  OUT_REG:bit:='0';
                  PIPE_REG:bit:='0';
                  ASIGN_REG:bit:='0';
                  BSIGN_REG:bit:='0';
                  ACCLOAD_REG0:bit:='0';
                  ACCLOAD_REG1:bit:='0';
                  SOA_REG:bit:='0';
                  MULTALU36X18_MODE:integer:=0;
                  C_ADD_SUB:bit:='0';
                  MULT_RESET_MODE:string:="SYNC"
      );
  PORT(
    A:IN std_logic_vector(17 downto 0);
          B:IN std_logic_vector(35 downto 0);
          C:IN std_logic_vector(53 downto 0);
          ASIGN:IN std_logic;
          BSIGN:IN std_logic;
          CE:IN std_logic;
          CLK:IN std_logic;
          RESET:IN std_logic;
          ACCLOAD:IN std_logic;
          CASI:IN std_logic_vector(54 downto 0);
          CASO:OUT std_logic_vector(54 downto 0);
           DOUT:OUT std_logic_vector(53 downto 0)
      );
END COMPONENT;
uut:MULTALU36X18
  GENERIC MAP (AREG=>'1',
                    BREG=>'0',
                    CREG=>'0',
                    OUT_REG=>'1',
                    PIPE_REG=>'0',
                    ASIGN_REG=>'0',
                    BSIGN_REG=>'0',
```

```
                                ACCLOAD_REG0=>'1',
                                ACCLOAD_REG1=>'0',
                                SOA_REG=>'0',
                                MULTALU36X18_MODE=>0,
                                C_ADD_SUB=>'0',
                                MULT_RESET_MODE=>"SYNC"
            )
        PORT MAP (
                A=>a,
                B=>b,
                C=>c,
                ASIGN=>asign,
                BSIGN=>bsign,
                CE=>ce,
                CLK=>clk,
                RESET=>reset,
                ACCLOAD=>accload,
                CASI=>casi,
                CASO=>caso,
                DOUT=>dout
    );
```

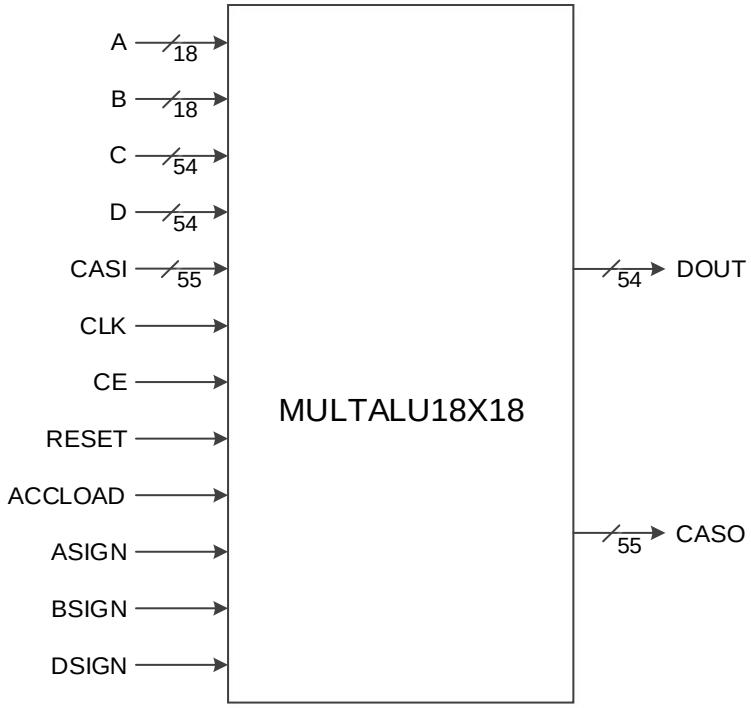
5.4.2 MULTALU18X18

Primitive Introduction

18x18 Multiplier with ALU (MULTALU18X18) is a 36x18 multiplier with ALU function.

Block Diagram

Figure 5-8 MULTALU18X18 Block Diagram



Port Description

Table 5-15 Port Description

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

Attribute Description

Table 5-16 Attribute Introduction

Attribute Name	Permitted Values	Default Value	Description
AREG	1'b0,1'b1	1'b0	Input A(A)register can be bypassed 1'b0:bypass mode 1'b1:registered mode
BREG	1'b0,1'b1	1'b0	Input B(B)register can be bypassed 1'b0:bypass mode 1'b1:registered mode
CREG	1'b0,1'b1	1'b0	Input C(C) register can be bypassed 1'b0:bypass mode 1'b1:registered mode
DREG	1'b0,1'b1	1'b0	Input D(D) register can be bypassed 1'b0:bypass mode 1'b1:registered mode
DSIGN_REG	1'b0,1'b1	1'b0	DSIGN input register can be bypassed 1'b0:bypass mode 1'b1:registered mode
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN input register can be bypassed 1'b0:bypass mode 1'b1:registered mode
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN input register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
ACCLOAD_REG0	1'b0,1'b1	1'b0	The first stage register of ACCLOAD can be bypassed 1'b0:bypass mode 1'b1:registered mode

Attribute Name	Permitted Values	Default Value	Description
ACCLOAD_REG1	1’b0,1’b1	1’b0	The second stage register of ACCLOAD can be bypassed 1’b0:bypass mode 1’b1:registered mode
MULT_RESET_MODE	SYNC,ASYNC	SYNC	Reset mode config,synchronous or asynchronous
PIPE_REG	1’b0,1’b1	1’b0	Pipeline register can be bypassed . 1’b0:bypass mode 1’b1:registered mode
OUT_REG	1’b0,1’b1	1’b0	The output registers can be bypassed. 1’b0:bypass mode 1’b1: registered mode
B_ADD_SUB	1’b0,1’b1	1’b0	B_OUT ADD/SUB Selection 1’b0: add 1’b1: sub
C_ADD_SUB	1’b0,1’b1	1’b0	C_OUT ADD/SUB Selection 1’b0: add 1’b1: sub
MULTALU18X18_MODE	0,1,2	0	MULTALU36X18 Operation Mode and Unit Input Selection 0:ACC/0 +/- 18x18 +/- C; 1:ACC/0 +/- 18x18 + CASI; 2: 18x18 +/- D + CASI;

Primitive Instantiation

Verilog Instantiation:

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

```
defparam multalu18x18_inst.DSIGN_REG = 1'b0;
defparam multalu18x18_inst.ACCLOAD_REG0 = 1'b1;
defparam multalu18x18_inst.ACCLOAD_REG1 = 1'b0;
defparam multalu18x18_inst.MULT_RESET_MODE = "SYNC";
defparam multalu18x18_inst.MULTALU18X18_MODE = 0;
defparam multalu18x18_inst.B_ADD_SUB = 1'b0;
defparam multalu18x18_inst.C_ADD_SUB = 1'b0;
```

Vhdl Instantiation:

```
COMPONENT MULTALU18X18
  GENERIC (AREG:bit:='0';
                BREG:bit:='0';
                CREG:bit:='0';
                DREG:bit:='0';
                OUT_REG:bit:='0';
                PIPE_REG:bit:='0';
                ASIGN_REG:bit:='0';
                BSIGN_REG:bit:='0';
                DSIGN_REG:bit:='0';
                ACCLOAD_REG0:bit:='0';
                ACCLOAD_REG1:bit:='0';
                B_ADD_SUB:bit:='0';
                C_ADD_SUB:bit:='0';
                MULTALU18X18_MODE:integer:=0;
                MULT_RESET_MODE:string:="SYNC"
      );
  PORT(
   A:IN std_logic_vector(17 downto 0);
          B:IN std_logic_vector(17 downto 0);
          C:IN std_logic_vector(53 downto 0);
          D:IN std_logic_vector(53 downto 0);
          ASIGN:IN std_logic;
          BSIGN:IN std_logic;
          DSIGN:IN std_logic;
          CE:IN std_logic;
          CLK:IN std_logic;
          RESET:IN std_logic;
          ACCLOAD:IN std_logic;
          CASI:IN std_logic_vector(54 downto 0);
          CASO:OUT std_logic_vector(54 downto 0);
           DOUT:OUT std_logic_vector(53 downto 0)
      );
END COMPONENT;
uut:MULTALU18X18
  GENERIC MAP (AREG=>'1',
                BREG=>'0',
                CREG=>'0',
                DREG=>'0',
                OUT_REG=>'1',
                PIPE_REG=>'0',
                ASIGN_REG=>'0',
```

```
                                    BSIGN_REG=>'0',
                                    DSIGN_REG=>'0',
                                    ACCLOAD_REG0=>'1',
                                    ACCLOAD_REG1=>'0',
                                    B_ADD_SUB=>'0',
                                    C_ADD_SUB=>'0',
                                    MULTALU18X18_MODE=>0,
                                   MULT_RESET_MODE=>"SYNC"
        )
    PORT MAP (
            A=>a,
            B=>b,
            C=>c,
            D=>d,
            ASIGN=>asign,
            BSIGN=>bsign,
            DSIGN=>dsign,
            CE=>ce,
            CLK=>clk,
            RESET=>reset,
            ACCLOAD=>accload,
            CASI=>casi,
            CASO=>caso,
            DOUT=>dout
);
```

5.5 MULTADDALU

The Sum of Two Multipliers with ALU (MULTADDALU) is a MAC with the function of ALU, and The corresponding primitive is MULTADDALU18X18.

The three operation modes are as follows:

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

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

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

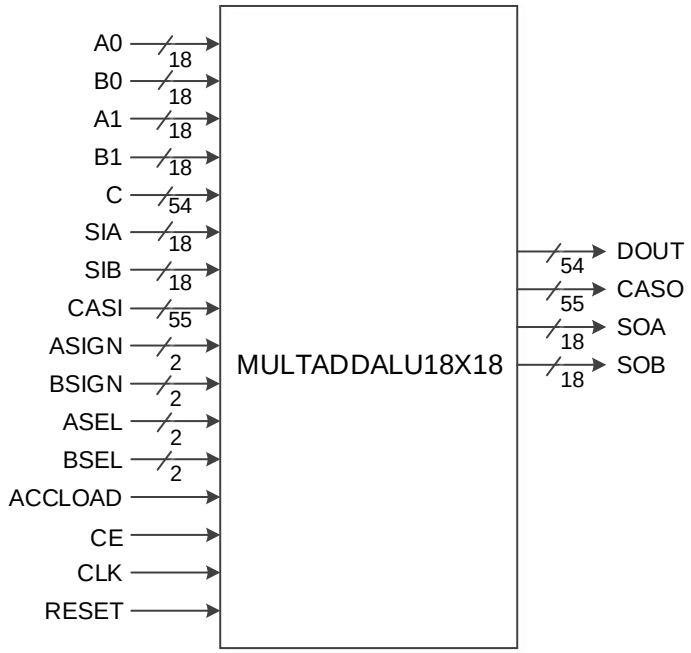
5.5.1 MULTADDALU18X18

Primitive Introduction

The Sum of Two 18x18 Multipliers with ALU (MULTADDALU18X18) is an 18x18 MAC with The function of ALU, which can be used to accumulate the sum of multiplication or reload.

Block Diagram

Figure 5-9 MULTADDALU18X18 Block Diagram



Port Description

Table 5-17 Port Description

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

Port Name	I/O	Description
SOB[17:0]	Output	Multiplier Register Output B

Attribute Description

Table 5-18 Attribute Introduction

Attribute Name	Permitted Values	Default	Description
A0REG	1'b0,1'b1	1'b0	Input A0(A0 or SIA) register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
A1REG	1'b0,1'b1	1'b0	Input A1(A1 or Register Output A0) register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
B0REG	1'b0,1'b1	1'b0	Input B0(B0 or SIB) register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
B1REG	1'b0,1'b1	1'b0	Input B1(B1 or Register Output B0) register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
CREG	1'b0,1'b1	1'b0	Input C(C) register can be bypassed 1'b0:bypass mode 1'b1:registered mode
PIPE0_REG	1'b0,1'b1	1'b0	Multiplier0 Pipeline register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
PIPE1_REG	1'b0,1'b1	1'b0	Multiplier1 Pipeline register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
OUT_REG	1'b0,1'b1	1'b0	Output register can be bypassed 1'b0:bypass mode 1'b1:registered mode
ASIGN0_REG	1'b0,1'b1	1'b0	ASIGN[0] input register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
ASIGN1_REG	1'b0,1'b1	1'b0	ASIGN[1] input register can be bypassed. 1'b0:bypass mode 1'b1:registered mode
ACCLOAD_REG0	1'b0,1'b1	1'b0	The first stage register of ACCLOAD can be bypassed 1'b0:bypass mode 1'b1:registered mode
ACCLOAD_REG1	1'b0,1'b1	1'b0	The second stage register of ACCLOAD can be bypassed 1'b0:bypass mode 1'b1:registered mode
BSIGN0_REG	1'b0,1'b1	1'b0	BSIGN[0] input register can be bypassed. 1'b0:bypass mode 1'b1:registered mode

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

Primitive Instantiation

Verilog Instantiation:

```
MULTADDALU18X18 uut(
    .DOUT(dout[53:0]),
    .CASO(caso[54:0]),
    .SOA(soa[17:0]),
    .SOB(sob[17:0]),
    .A0(a0[17:0]),
    .B0(b0[17:0]),
    .A1(a1[17:0]),
    .B1(b1[17:0]),
    .C(c[53:0]),
    .SIA(sia[17:0]),
    .SIB(sib[17:0]),
    .CASI(casi[54:0]),
    .ACCLOAD(accload),
    .ASEL(asel[1:0]),
    .BSEL(bsel[1:0]),
    .ASIGN(asign[1:0]),
    .BSIGN(bsign[1:0]),
    .CLK(clk),
    .CE(ce),
    .RESET(reset)
);
defparam uut.A0REG = 1'b0;
defparam uut.A1REG = 1'b0;
defparam uut.B0REG = 1'b0;
defparam uut.B1REG = 1'b0;
defparam uut.CREG = 1'b0;
defparam uut.PIPE0_REG = 1'b0;
defparam uut.PIPE1_REG = 1'b0;
```

```
defparam uut.OUT_REG = 1'b0;
defparam uut.ASIGN0_REG = 1'b0;
defparam uut.ASIGN1_REG = 1'b0;
defparam uut.ACCLOAD_REG0 = 1'b0;
defparam uut.ACCLOAD_REG1 = 1'b0;
defparam uut.BSIGN0_REG = 1'b0;
defparam uut.BSIGN1_REG = 1'b0;
defparam uut.SOA_REG = 1'b0;
defparam uut.B_ADD_SUB = 1'b0;
defparam uut.C_ADD_SUB = 1'b0;
defparam uut.MULTADDALU18X18_MODE = 0;
defparam uut.MULT_RESET_MODE = "SYNC";
```

Vhdl Instantiation:

```
COMPONENT MULTADDALU18X18
    GENERIC (A0REG:bit:='0';
                    B0REG:bit:='0';
                    A1REG:bit:='0';
                    B1REG:bit:='0';
                    CREG:bit:='0';
                    OUT_REG:bit:='0';
                    PIPE0_REG:bit:='0';
                    PIPE1_REG:bit:='0';
                    ASIGN0_REG:bit:='0';
                    BSIGN0_REG:bit:='0';
                    ASIGN1_REG:bit:='0';
                    BSIGN1_REG:bit:='0';
                    ACCLOAD_REG0:bit:='0';
                    ACCLOAD_REG1:bit:='0';
                    SOA_REG:bit:='0';
                    B_ADD_SUB:bit:='0';
                    C_ADD_SUB:bit:='0';
                    MULTADDALU18X18_MODE:integer:=0;
                    MULT_RESET_MODE:string:="SYNC"
        );
    PORT(
     A0:IN std_logic_vector(17 downto 0);
                A1:IN std_logic_vector(17 downto 0);
                B0:IN std_logic_vector(17 downto 0);
                B1:IN std_logic_vector(17 downto 0);
                SIA:IN std_logic_vector(17 downto 0);
                SIB:IN std_logic_vector(17 downto 0);
                C:IN std_logic_vector(53 downto 0);
                ASIGN:IN std_logic_vector(1 downto 0);
                BSIGN:IN std_logic_vector(1 downto 0);
                ASEL:IN std_logic_vector(1 downto 0);
                BSEL:IN std_logic_vector(1 downto 0);
                CE:IN std_logic;
                CLK:IN std_logic;
                RESET:IN std_logic;
                ACCLOAD:IN std_logic;
```

```
                    CASI:IN std_logic_vector(54 downto 0);
                    SOA:OUT std_logic_vector(17 downto 0);
                    SOB:OUT std_logic_vector(17 downto 0);
                    CASO:OUT std_logic_vector(54 downto 0);
                     DOUT:OUT std_logic_vector(53 downto 0)
        );
END COMPONENT;
uut:MULTADDALU18X18
  GENERIC MAP (A0REG=>'0',
                          B0REG=>'0',
                          A1REG=>'0',
                          B1REG=>'0',
                          CREG=>'0',
                          OUT_REG=>'0',
                          PIPE0_REG=>'0',
                          PIPE1_REG=>'0',
                          ASIGN0_REG=>'0',
                          BSIGN0_REG=>'0',
                          ASIGN1_REG=>'0',
                          BSIGN1_REG=>'0',
                          ACCLOAD_REG0=>'0',
                          ACCLOAD_REG1=>'0',
                          SOA_REG=>'0',
                          B_ADD_SUB=>'0',
                          C_ADD_SUB=>'0',
                          MULTADDALU18X18_MODE=>0,
                         MULT_RESET_MODE=>"SYNC"
        )
  PORT MAP (
            A0=>a0,
            A1=>a1,
            B0=>b0,
            B1=>b1,
            SIA=>sia,
            SIB=>sib,
            C=>c,
            ASIGN=>asign,
            BSIGN=>bsign,
            ASEL=>asel,
            BSEL=>bsel,
            CE=>ce,
            CLK=>clk,
            RESET=>reset,
            ACCLOAD=>accload,
            CASI=>casi,
            SOA=>soa,
            SOB=>sob,
            CASO=>caso,
            DOUT=>dout
    );
```

6 Clock

6.1 PLL

Primitive Name

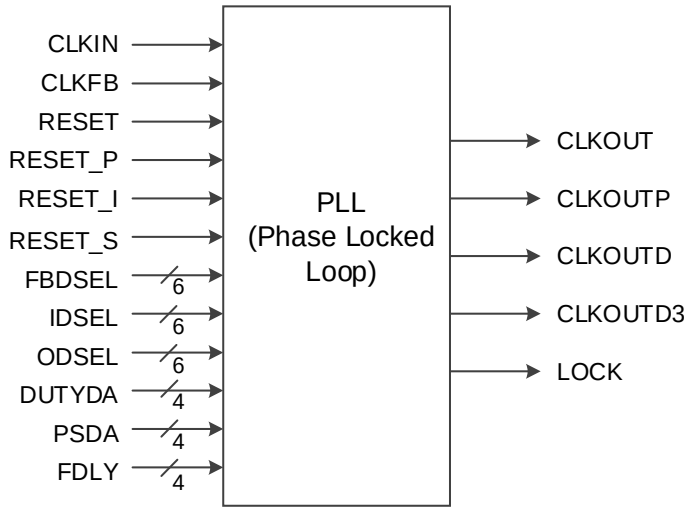
Gowin FPGA provides PLL. The frequency and phase of the internal oscillator signal is controlled by the external input reference clock.

Devices Supported

Devices Supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

Figure 6-1 PLL Port Diagram



Functional Description

Based on the input clock, PLL adjusts clock phase, duty cycle, frequency (multiplication and division) to output clocks with different phases and frequencies.

The main features of PLL are outlined in Table 6-1.

Table 6-1 PLL Features

Device Frequency（MHz）	GW1N Family	GW1N-1S	GW1NS-2	GW1NZ-1	GW2A Family
Input frequency	3 ~ 450	3 ~ 450	3 ~ 450	3 ~ 400(LV) 3 ~ 200(ZV)	3 ~ 500
VCO vibration frequency	400~900	400~1200	400~1500	400 ~ 800(LV) 200 ~ 400(ZV)	500~1300
Output frequency	3.125~450	3.125~600	3.125~750	3.125~400(LV) 1.5625~200(ZV)	3.125~500

PLL can adjust the frequency of the input clock CLKIN (multiply and division). The formulas are as follows:

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

Note！

- f_{CLKIN} is the input clock CLKIN frequency; f_{CLKOUT} is the CLKOUT and CLKOUTP clock frequency; $f_{CLKOUTD}$ is the CLKOUTD clock frequency, and f_{PFD} is the PFD phase discrimination frequency.
- IDIV, FBDIV, ODIV and SDIV are the actual frequency division coefficients of different frequency dividers, which can be adjusted to get the desired frequency clock signal.

Port Description

Table 6-2 Port Description

Port Name	I/O	Description
CLKIN	Input	Reference clock input
CLKFB	Input	Feedback clock input
RESET	Input	PLL asynchronous reset input, active-high
RESET_P	Input	PLL power down input, active-hiigh
RESET_I	Input	PLL IDIV asynchronous reset input, active-high
RESET_S	Input	PLL SDIV asynchronous reset input, active-high
FBDSEL[5:0]	Input	Dynamic control FBDIV value:1~64
IDSEL[5:0]	Input	Dynamic control IDIV value: 1~64
ODSEL[5:0]	Input	Dynamic control ODIV value: 64,80~96,112,128
DUTYDA[3:0]	Input	Duty cycle dynamic adjustment
PSDA[3:0]	Input	Phase dynamic adjustment
FDLY[3:0]	Input	Fine delay dynamic adjustment
CLKOUT	Output	PLL Clock output
LOCK	Output	PLL lock status: 1 locked, 0 unlocked
CLKOUTP	Output	PLL clock output with phase and duty cycle adjustment
CLKOUTD	Output	PLL after SDIV clock output, CLKOUT or CLKOUTP after SDIV frequency divider output
CLKOUTD3	Output	PLL after DIV3 clock output, CLKOUT or CLKOUTP after 3 frequency dividers output

Parameter

Table 6-3 Parameter Description

Name	Value	Default Value	Description
FCLKIN	3~500	100	Reference clock frequency
IDIV_SEL	0~63	0	IDIV frequency division coefficient static setting
DYN_IDIV_SEL	true,false	false	IDIV frequency division coefficient static control parameter or dynamic control signal selection false: Static, that is, select the parameter IDIV_SEL true: Dynamic, namely select signal IDSEL
FBDIV_SEL	0~63	0	FBDIV frequency division coefficient static setting
DYN_FBDIV_SEL	true,false	false	FBDIV frequency division coefficient static control parameter or dynamic control signal selection false: Static, that is, select the parameter FBDIV_SEL true: Dynamic, namely select signal FBDSEL
ODIV_SEL	2,4,8,16,32,48,64,80,96,112,128	8	ODIV frequency division coefficient static setting
DYN_ODIV_SEL	true,false	false	ODIV frequency division coefficient static control parameter or dynamic control signal selection false: Static, that is, select the parameter ODIV_SEL true: Dynamic, namely select signal ODSEL
PSDA_SEL	0000~1111	0000	Phase static adjustment
DUTYDA_SEL	0010~1110	1000	Duty cycle static adjustment
DYN_DA_EN	true,false	false	The dynamic signal is selected as the control of phase and duty cycle adjustment false: Static control true: Dynamic control
CLKOUT_FT_DIR	1'b1	1'b1	CLKOUT fine adjustment direction setting 1'b1: add
CLKOUT_DLY_STEP	0,1,2,4	0	CLKOUT fine adjustment coefficient setting CLKOUT_DLY_STEP*delay(delay=50ps)
CLKOUTP_FT_DIR	1'b1	1'b1	CLKOUTP fine adjustment direction setting 1'b1: add
CLKOUTP_DLY_STEP	0,1,2	0	CLKOUTP fine adjustment coefficient setting CLKOUTP_DLY_STEP*delay(delay=50ps)
DYN_SDIV_SEL	2~128 (Even)	2	SDIV frequency division coefficient static setting
CLKFB_SEL	internal,external	internal	CLKFB source selection internal: CLKOUT internal feedback external: Feedback from external signals
CLKOUTD_SRC	CLKOUT,CLKOUTP	CLKOUT	CLKOUTD source selection
CLKOUTD3_SRC	CLKOUT,CLKOUTP	CLKOUT	CLKOUTD3 source selection
CLKOUT_BYPA	true,false	false	Bypass PLL, CLKOUT comes directly from

Name	Value	Default Value	Description
SS			CLKIN true: CLKIN bypass PLL acts directly on CLKOUT false: Normal
CLKOUTP_BYPASS	true,false	false	Bypass PLL, CLKOUTP comes directly from CLKIN true: CLKIN bypass PLL acts directly on CLKOUTP false: Normal
CLKOUTD_BYPASS	true,false	false	Bypass PLL, CLKOUTD comes directly from CLKIN true: CLKIN bypass PLL acts directly on CLKOUTD false: Normal
DEVICE	GW1N-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, and GW2A-55 devices.	GW1N-2	Devices selection

Primitive Instantiation

Verilog Instantiation:

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

```
defparam pll_inst.IDIV_SEL = 0;
defparam pll_inst.DYN_FBDIV_SEL = "false";
defparam pll_inst.FBDIV_SEL = 1;
defparam pll_inst.ODIV_SEL = 8;
defparam pll_inst.PSDA_SEL = "0100";
defparam pll_inst.DYN_DA_EN = "false";
defparam pll_inst.DUTYDA_SEL = "1000";
defparam pll_inst.CLKOUT_FT_DIR = 1'b1;
defparam pll_inst.CLKOUTP_FT_DIR = 1'b1;
defparam pll_inst.CLKOUT_DLY_STEP = 0;
defparam pll_inst.CLKOUTP_DLY_STEP = 0;
defparam pll_inst.CLKFB_SEL ="external";
defparam pll_inst.CLKOUT_BYPASS = "false";
defparam pll_inst.CLKOUTP_BYPASS = "false";
defparam pll_inst.CLKOUTD_BYPASS = "false";
defparam pll_inst.DYN_SDIV_SEL = 2;
defparam pll_inst.CLKOUTD_SRC = "CLKOUT";
defparam pll_inst.CLKOUTD3_SRC = "CLKOUT";
defparam pll_inst.DEVICE = "GW1N-4";
```

Vhdl Instantiation:

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

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

```
                CLKOUTP_BYPASS=>"false",
                CLKOUTD_BYPASS=>"false",
                CLKOUTD_SRC=>"CLKOUT",
                DYN_SDIV_SEL=>2
        )
         PORT  MAP(
            CLKIN=>clkin,
            CLKFB=>clkfb,
            IDSEL=>idsel,
            FBDSEL=>fbdsel,
            ODSEL=>odsel,
            RESET=>reset,
            RESET_P=>reset_p,
            RESET_I=>reset_i,
            RESET_S=>reset_s,
            PSDA=>psda,
            FDLY=>fdly,
            DUTYDA=>dutyda,
            LOCK=>lock,
            CLKOUT=>clkout,
            CLKOUTD=>clkoutd,
            CLKOUTP=>clkoutp ,
            CLKOUTD3=>clkoutd3
    );
```

6.2 DLL/DLLDLY

6.2.1 DLL

Primitive Name

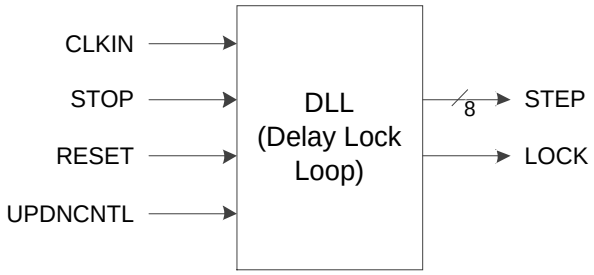
Delay lock loop (DLL) is mainly used for accurate generation of time delay.

Devices Supported

Devices supported: GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

Figure6-2 DLL Port Diagram



Functional Description

DLL can adjust the clock phase based on input clock to generate different phase delay step. The calculated output STEP signal will be sent to the adjacent Bank, such as DQS and DLLDLY module. Meanwhile, the signal STEP can also be sent to the user logic through routing.

Port Description

Table 6-4 Port Description

Port Name	I/O	Description
STEP[7:0]	Output	Delay step output
LOCK	Output	Lock status: 1 locked, 0 unlocked
CLKIN	Input	Clock input
STOP	Input	Disable input clock and internal oscillator
RESET	Input	Asynchronous reset input, active-high
UPDNCNTL	Input	DLL delay step update control, active-low

Parameter

Table 6-5 Parameter Description

Name	Type	Value	Default Value	Description
DLL_FORCE	Integer	0,1	0	DLL mandatory delay step, lock control 1: mandatory lock, delay step is 255 (maximum), used for lower input frequency mode 0: In normal mode, the DLL generates delay step and lock signal
CODESCAL	String	000,001,010,011,100,101, 110, 111	000	DLL phase shift configuration (45°~135°) 000:101° 001:112° 010:123° 011:135°

Name	Type	Value	Default Value	Description
				100:79° 101:68° 110:57° 111:45°
SCAL_EN	String	true,false	true	DLL enables phase shift: True: enabled, phase shift setting according to CODESCAL False: disabled, default 90° phase shift
DIV_SEL	Integer	1’b0,1’b1	1’b0	DLL lock mode selection 1'b0: normal lock mode 1'b1: quick lock mode

Connection Validity Rule

The DLL output step can be connected with DQS and DLLDLY modules. Meanwhile, the signal STEP can also be sent to the user logic through routing.

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

```
 COMPONENT DLL
        GENERIC(
                DLL_FORCE:integer:=0;
                DIV_SEL:bit:='1';
                CODESCAL:STRING:="000";
                SCAL_EN:STRING:="true"
        );
            PORT(
                CLKIN:IN std_logic;
                STOP:IN std_logic;
                RESET:IN std_logic;
                UPDNCNTL:IN std_logic;
                LOCK:OUT std_logic;
                STEP:OUT std_logic_vector(7 downto 0)
             );
```

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

6.2.2 DLLDLY

Primitive Name

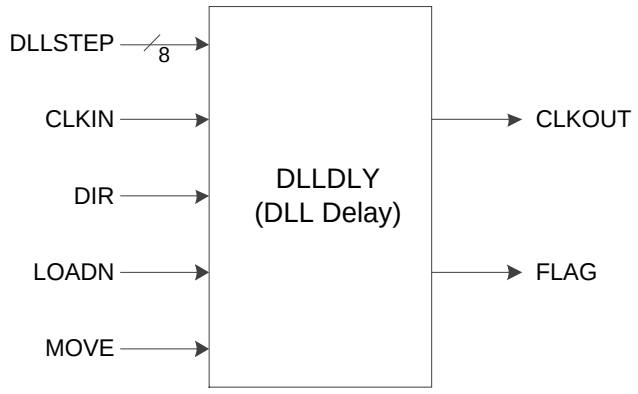
DLL Delay (DLLDLY) adjusts the input clock according to DLLSTEP signal and outputs the delay of the clock.

Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

Figure 6-3 DLLDLY Port Diagram



Functional Description

DLLDLY can be used with DLL, DLL provides different phase delay STEP, DLLDLY generates corresponding phase delay according to DLLSTEP and obtains the delay output based on CLKIN.

Port Description

Table 6-6 Port Description

Port Name	I/O	Description
CLKOUT	Output	Clock output
FLAG	Output	An output flag that represents under-flow or over-flow in dynamical delay adjustment
DLLSTEP[7:0]	Input	Delay STEP input, output STEP from DLL
CLKIN	Input	Clock input
DIR	Input	Set the direction of dynamical delay adjustment 0: Increase delay 1: Decrease delay
LOADN	Input	Control the loading delay step 0: Loading delay step DLLSTEP 1: adjust the dynamical delay
MOVE	Input	MOVE is the delay value of negedge dynamical adjustment, and each pulse moves one delay step

Parameter

Table 6-7 Parameter Description

Name	Type	Value	Default Value	Description
DLL_INSEL	Integer	1’b0,1’b1	1’b0	DLLDLY bypass mode selection 1'b0: bypass mode, the output comes directly from the CLKIN 1'b1: Normal mode, using DLLDLY delay module
DLY_SIGN	String	1’b0,1’b1	1’b0	Set the sign of delay adjustment 1'b0:'+' 1'b1: '-'
DLY_ADJ	Integer	0~255	0	Delay adjustment setting dly_sign=0 DLY_ADJ; dly_sign=1 -256+ DLY_ADJ

Connection Validity Rule

DLLSTEP of DLLDLY comes from STEP in DLL module , if the device does not have DLL , it can be from user logic.

Primitive Instantiation

Verilog Instantiation:

```
DLLDLY dlldly_0 (
  .CLKIN(clkin),
  .DLLSTEP(step[7:0]),
```

```
    .DIR(dir),
    .LOADN(loadn),
    .MOVE(move),
    .CLKOUT(clkout),
      .FLAG(flag)
  );
  defparam dlldly_0.DLL_INSEL=1'b1;
  defparam dlldly_0.DLY_SIGN=1'b1;
  defparam dlldly_0.DLY_ADJ=0;
```

Vhdl Instantiation:

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

6.3 CLKDIV

Primitive Name

The clock divider (CLKDIV) realizes clock frequency adjustment

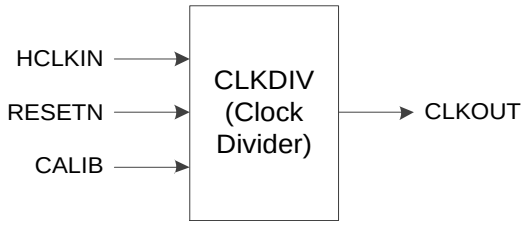
Devices Supported

Devices Supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9,

GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

Figure 6-4 CLKDIV Port Diagram



Functional Description

CLKDIV generates and inputs frequency division clock with same clock phase, which is used in the IO logic mode. It supports 2/3.5/4/5/8 frequency division in GW1N-6, GW1N-9, and GW1NS-2 and 2/3.5/4/5 frequency division of other devices.

Port Description

Table 6-8 Port Description

Port Name	I/O	Description
HCLKIN	Input	Clock input
RESETN	Input	Asynchronous reset input, active-low
CALIB	Input	CALIB input, adjust output clock
CLKOUT	Output	Clock output

Parameter

Table 6-9 Parameter Description

Name	Value	Default Value	Description
DIV_MODE	2, 3.5, 4, 5, 8 (Only GW1N-6, GW1N-9 and GW1NS-2 support 8)	2	Set the clock frequency division coefficient
GSREN	false, true	false	Enable global reset

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

6.4 DQCE

Primitive Name

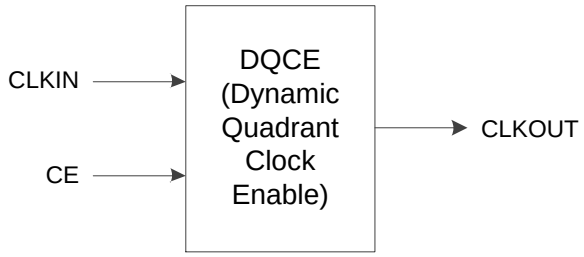
Dynamic Quadrant Clock Enable (DQCE) dynamically enables Gowin quadrant clock.

Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

Figure 6-5 DQCE Port Diagram



Functional Description

GCLK0~GCLK5 can be dynamically opened/closed through DQCE. When GCLK0~GCLK5 in the quadrant is off, all the logic driven by it will not toggle; therefore, lower power can be achieved.

Port Description

Table 6-10 Port Description

Port Name	I/O	Description
CLKIN	Input	Clock input
CE	Input	Clock enable input, active-high.
CLKOUT	Output	Clock output

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

6.5 DCS

Primitive Name

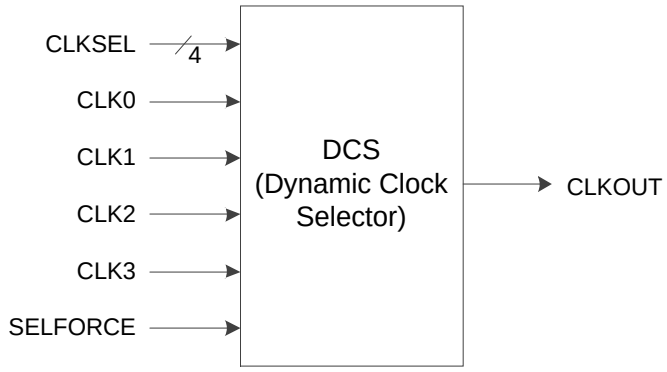
Dynamic clock select (DCS) selects quadrant clock GCLK6 and GCLK7 dynamically.

Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

Figure 6-6 DCS Port Diagram



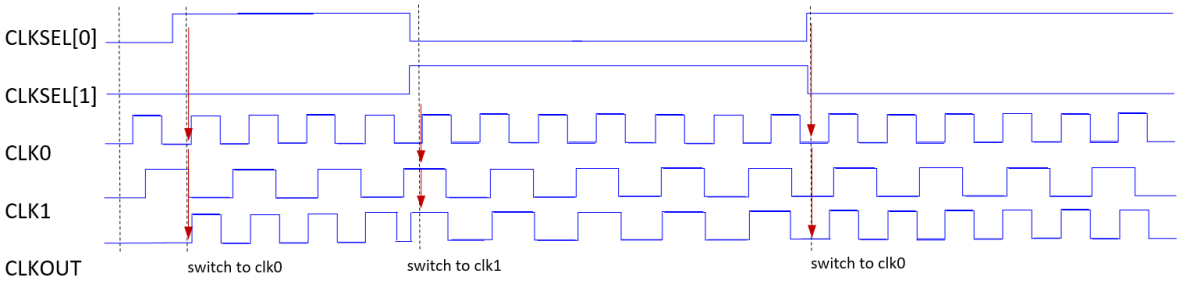
Functional Description

GCLK6~GCLK7 of each quadrant is controlled by the dynamic clock selector (DCS); select one of the four clocks as the global clock. Switch dynamically between CLK0~CLK3 by CRU, and output a glitch-free clock.

DCS has two clock switching modes, namely "non-glitchless" and "Glitchless".

In non-glitchless mode (input SELFORCE = '1'), DCS acts like a multiplexer, switching clock signals only through CLKSEL signals, allowing glitch on the output, depending on the time of switching. The timing of non-glitchless mode is shown in Figure 6-7. CLKSEL[3]~CLKSEL[0] are corresponding to CLK3~CLK0 with the same conversion timing, active-high.

Figure 6-7 Timing Diagram of Non-Glitchless



In Glitchless mode (input SELFORCE = '0'), DCS_MODE configures the CLKSEL signal to avoid glitch on the output clock. The timing of Glitchless mode is shown in Figure 6-8Figure 6-11. CLKSEL[3]~CLKSEL[0] are corresponding to CLK3~CLK0 with the same conversion timing.

Figure 6-8 RISING Timing Diagram in DCS Mode

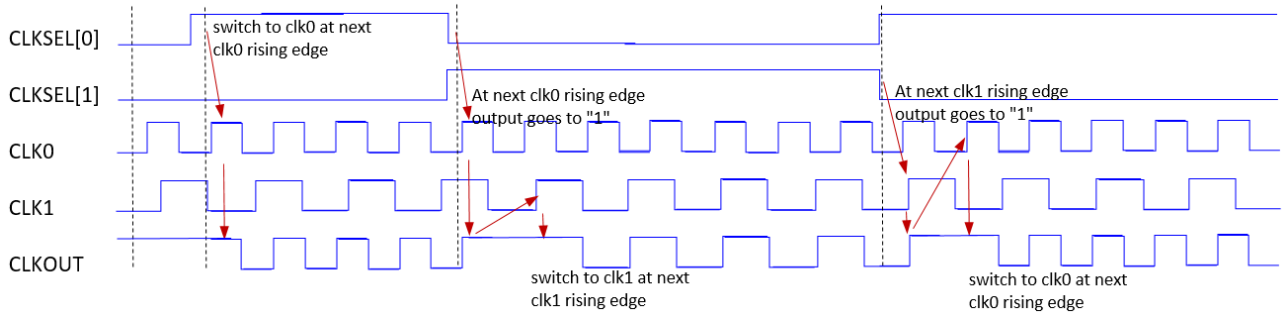


Figure 6-9 FALLING Timing Diagram in DCS Mode

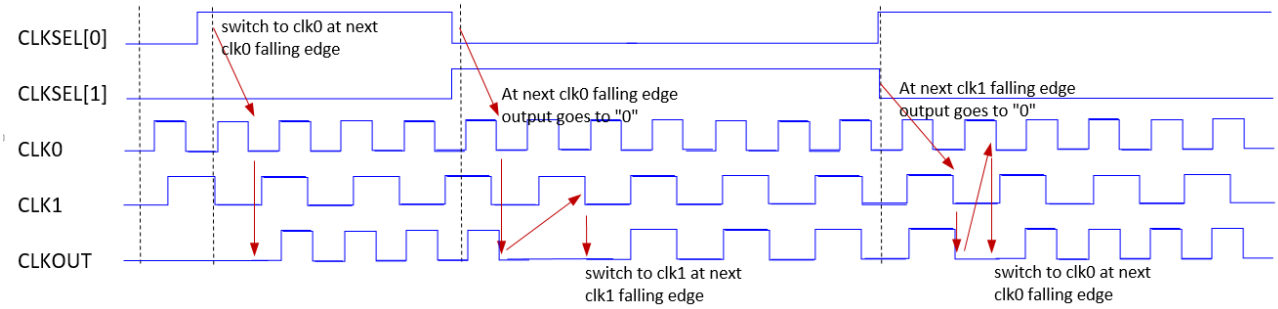


Figure 6-10 CLK0_GND Timing Diagram in DCS Mode

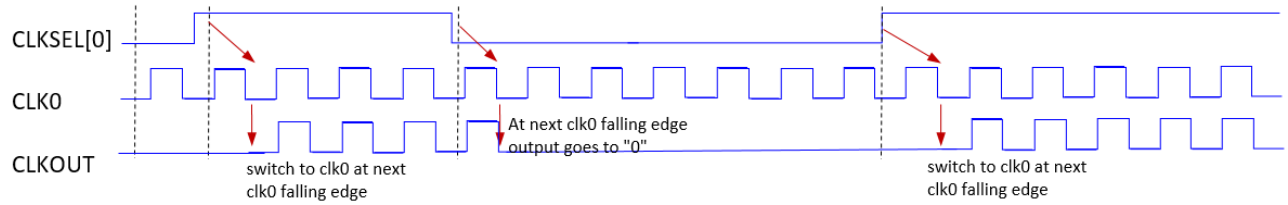
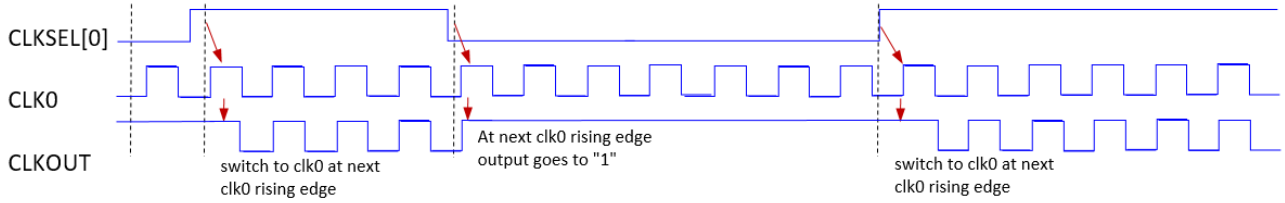


Figure 6-11 CLK0_VCC Timing Diagram in DCS Mode



Port Description

Table 6-11 Port Description

Port Name	I/O	Description
CLK0	Input	Clock input 0
CLK1	Input	Clock input 1
CLK2	Input	Clock input 2
CLK3	Input	Clock input 3
CLKSEL[3:0]	Input	Clock selection signal
SELFORCE	Input	Mandatory mode selection 0：glitchless mode

Port Name	I/O	Description
		1：Non-glitchless mode
CLKOUT	Output	Clock output

Parameter

Table 6-12 Parameter Description

Name	Value	Default Value	Description
DCS_MODE	CLK0,CLK1,CLK2,CLK3, GND,VCC,RISING,FALLIN G, CLK0_GND,CLK1_GND, CLK2_GND,CLK3_GND, CLK0_VCC,CLK1_VCC, CLK2_VCC,CLK3_VCC	RISING	Set DCS mode

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

```
        SELFORCE=>selforce,
        CLKOUT=>clkout
);
```

6.6 DQS

Primitive Introduction

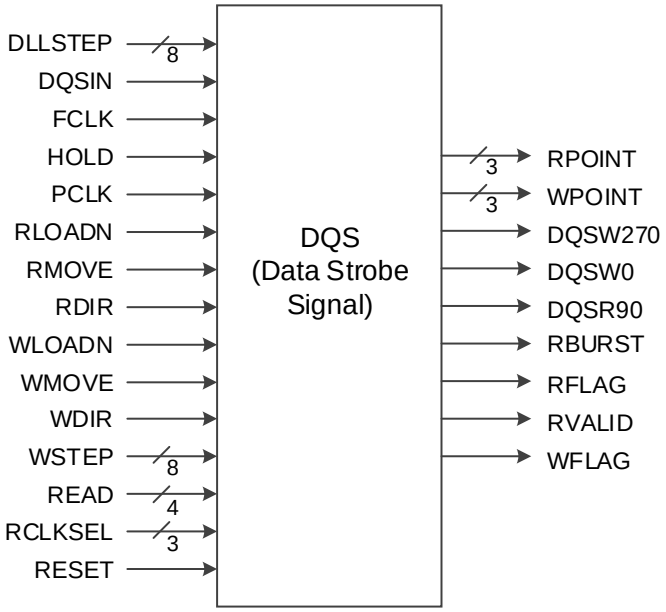
Bidirectional Data Strobe Circuit for DDR Memory (DQS) is a bidirectional data strobe pulse circuit of DDR Memory interface.

Devices Supported

Devices supported: GW2A-18, GW2AR-18, and GW2A-55 devices.

Port Diagram

Figure 6-12 DQS Port Diagram



Functional Description

DQS is a key component of IP, which is mainly used to adjust the phase relationship between DQSIN and DQSR90, DQSW0 and DQSW270, and to complete writing balance and reading calibration.

Port Description

Table 6-13 Port Description

Port Name	I/O	Description
DLLSTEP[7:0]	input	DQS delay step control input, from DLL module
DQSIN	input	DQS input from IO PAD
FCLK	input	Fast clock come from two different FCLK clock trees output
HOLD	input	For DQS write, stop writing related signals to synchronize the output clock; For DQS read, reset FIFO counter

Port Name	I/O	Description
PCLK	input	Master clock, from the PCLK clock tree
RDIR	input	Adjust the delay direction of DDR read "0" increases the delay "1" decreases the delay
RLOADN	input	Reset the final delay step read by DDR to the initial value, active-low
RMOVE	input	RMOVE is the delay step read by DDR in negedge, and each pulse is changed once
WDIR	input	Adjust the delay direction of DDR write "0" increases the delay "1" decreases the delay
WLOADN	input	Reset the final delay step write by DDR to the initial value, active-low
WMOVE	input	RMOVE is the delay step wrote by DDR in negedge, and each pulse is changed once
WSTEP[7:0]	input	DDR write equalization delay control
READ[3:0]	input	READ signal for DDR read mode
RCLKSEL[2:0]	input	Select read clock source and polarity control
RESET	input	DQS reset input, active-high
RPOINT[2:0]	output	FIFO controls the read pointer and works on RADDR in IOLOGIC or on user logic via routing
WPOINT[2:0]	output	FIFO controls the write pointer and works on WADDR in IOLOGIC or on user logic via routing
DQSW0	output	PCLK/FCLK 0° phase shift output works on TCLK in IOLOGIC or on user logic via routing
DQSW270	output	PCLK/FCLK 270° phase shift output works on TCLK in IOLOGIC or on user logic via routing
DQSR90	output	DQSI 90° phase shift output works on TCLK in IOLOGIC or on user logic via routing
RFLAG	output	An output flag that represents reading under-flow or over-flow in READ delay adjustment
WFLAG	output	An output flag that represents writing under-flow or over-flow in WRITE delay adjustment
RVALID	output	READ mode data valid flag
RBURST	output	READ burst detection output

Parameter

Table 6-14 Parameter Description

Name	Value	Default Value	Description
FIFO_MODE_SEL	1’b0 , 1’b1	1’b0	FIFO mode selection 1'b0: DDR memory mode 1'b1: GDDR mode
RD_PNTR	000,001,010,011,100,101,110,111	3'b000	FIFO read pointer setting
DQS_MODE	X1,X2_DDR2,X2_DDR3,X4,X2_DDR3_EXT	X1	DQS mode selection
HWL	false,true	false	update0/1 timing relationship control "False ": update1 is one cycle ahead of update0; "True ": the timing of update1 and

Name	Value	Default Value	Description
			update0 are the same
GSREN	false,true	false	Enable global reset

Connection Validity Rule

- DQSI of DQS comes from IO PAD;
- DLLSTEP input of DQS comes from STEP output in DLL module;
- The RPOINT output of DQS can be connected to the RADDR of IOLOGIC and can also work on user logic.
- The WPOINT output of DQS can be connected to the WADDR of IOLOGIC and can also work on user logic.
- DQSR90 output of DQS can be connected to the ICLK of IOLOGIC and can also work on user logic
- DQSW0/ DQSW270 output of DQS can be connected to the TCLK of IOLOGIC and can also work on user logic.

Primitive Instantiation

Verilog Instantiation:

```
DQS uut (
        .DQSIN(dqs),
        .PCLK(pclk),
        .FCLK(fclk),
        .RESET(reset),
        .READ(read),
        .RCLKSEL(rsel),
        .DLLSTEP(step),
        .WSTEP(wstep),
        .RLOADN(1'b0),
        .RMOVE(1'b0),
        .RDIR(1'b0),
        .WLOADN(1'b0),
        .WMOVE(1'b0),
        .WDIR(1'b0),
        .HOLD(hold),
        .DQSR90(dqsr90),
        .DQSW0(dqsw0),
        .DQSW270(dqsw270),
        .RPOINT(rpoint),
        .WPOINT(wpoint),
        .RVALID(rvalid),
        .RBURST(rburst),
        .RFLAG(rflag),
        .WFLAG(wflag)
);
defparam uut.DQS_MODE = "X1";
defparam uut.FIFO_MODE_SEL = 1'b0;
defparam uut.RD_PNTR = 3'b001;
```

Vhdl Instantiation:

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

```
        RBURST=>rburst,
        RFLAG=>rflag,
        WFLAG=>wflag
);
```

6.7 OSC

Primitive Name

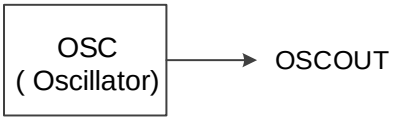
The OSC is the on-chip oscillator.

Devices Supported

Devices supported: GW1N-2, GW1N-2B, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

Figure 6-13 OSC Port Diagram



Functional Description

GOWIN FPGA is embedded with a programmable on-chip oscillator, which provides a clock source for MSPI programming mode.The on-chip oscillator also provides a clock resource for user designs. Up to 64 clock frequencies can be obtained by setting the parameters.

The following formula is employed to get the output clock frequency for GW1N-6/9 and GW2A-18/55 devices:

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

The following formula is employed to get the output clock frequency for GW1N-2/2B/4/4B devices:

$f_{CLKOUT} = 210MHz/\mathrm{FREQ\_DIV};$

"FREQ_DIV" is the configuration parameter and has a range of 2~128. It supports even numbers only.

Port Description

Table 6-15 Port Description

Port Name	I/O	Description
OSCOUT	output	OSC output Clock

Parameter

Table 6-16 Parameter Description

Name	Value	Default Value	Description
FREQ_DIV	2~128(even)	100	OSC frequency division coefficient setting

Name	Value	Default Value	Description
DEVICE	GW1N-1, GW1N-2, GW1N-2B, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55	GW1N-2 (GW1N series) GW2A-18 (GW2A series)	Devices selection

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

6.8 OSCZ

Primitive Name

OSCZ(Oscillator) is an on-chip oscillator with the function to dynamically turnoff OSC.

Devices Supported

Devices supported: GW1NZ-1, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C.

Port Diagram

Figure 6-14 OSCZ Port Diagram

Functional Description

The GW1NZ series FPGA products are embedded with a

programmable on-chip oscillator, clock accuracy up to ±5%, and support dynamic open/close OSC. The on-chip oscillator provides a clock resource for MSPI programming and user designs. Up to 64 clock frequencies can be obtained by setting the parameters. The following formula is employed to get the output clock frequency:

$f_{CLKOUT} = 250MHz/\text{FREQ\_DIV}$; "FREQ_DIV" is the configuration parameter and has a range of 2~128. It supports even numbers only.

Port Description

Table 6-17 Port Description

Port Name	I/O	Description
OSCEN	input	OSC Enable Signal
OSCOUT	output	OSC clock output

Parameter

Table 6-18 Parameter Description

Name	Value	Default Value	Description
FREQ_DIV	2~128(even)	100	OSC frequency division coefficient setting

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

6.9 OSCF

Primitive Name

Oscillator with CLKOUT30M and Dynamic OSC Enable (OSCF) is a 30M output clock and dynamic enable of on-chip oscillator.

Devices Supported

Devices supported: GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C.

Port Diagram

Figure 6-15 OSCF Port Diagram

Functional Description

The GW1NS series FPGA products are embedded with a programmable on-chip oscillator, clock accuracy up to ±5%, and support dynamic open/close OSC. The on-chip oscillator provides a clock resource for MSPI programming and user designs. Up to 64 clock frequencies can be obtained by setting the parameters. The following formula is employed to get the output clock frequency:

$f_{CLKOUT} = 240MHz/\mathrm{FREQ\_DIV}$;

"FREQ_DIV" is the configuration parameter and has a range of 2~128. It supports even numbers only.

Port Description

Table 6-19 Port Description

Port Name	I/O	Description
OSCEN	input	OSC enable signal
OSCOUT	output	OSC clock output
OSCOUT30M	output	OSC 30M clock output, connecting to PCLK of FLASH128K

Parameter

Table 6-20 Parameter Description

Name	Value	Default Value	Description
FREQ_DIV	2~128(even)	96	OSC frequency division coefficient setting

Connection Validity Rule

OSCOUT30M of OSC needs to be connected to PCLK of FLASH128K.

Primitive Instantiation

Verilog Instantiation:

```
OSCF uut(
    .OSCOUT(oscout)，
    .OSCOUT30M(oscout30m),
    .OSCEN(oscen)
    );
defparam uut.FREQ_DIV=96;
```

Vhdl Instantiation:

```
COMPONENT OSCF
        GENERIC(
                FREQ_DIV:integer:=96;
         );
        PORT(
                OSCOUT:OUT std_logic;
                OSCOUT30M:OUT std_logic;
                OSCEN:IN std_logic
                );
END COMPONENT;
uut:OSCF
      GENERIC MAP(FREQ_DIV=>96)
      PORT MAP(
                OSCOUT=>oscout,
                OSCOUT30M=>oscout30m,
                OSCEN(oscen)
                );
```

6.10 OSCH

Primitive Name

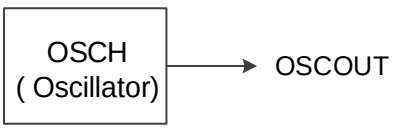
The OSCH is the on-chip oscillator.

Devices Supported

Devices supported: GW1N-1 and GW1N-1S devices.

Port Diagram

Figure 6-16 OSCH Port Diagram



Functional Description

The on-chip oscillator provides a clock resource for MSPI programming and user designs. Up to 64 clock frequencies can be obtained by setting the parameters. The following formula is employed to get the output clock frequency:

$f_{CLKOUT} = 240MHz/\mathrm{FREQ\_DIV}$; "FREQ_DIV" is the configuration parameter and has a range of 2~128. It supports even numbers only.

Port Description

Table 6-21 Port Description

Port Name	I/O	Description
OSCOUT	output	OSC clock output

Parameter

Table 6-22 Parameter Description

Name	Value	Default Value	Description
FREQ_DIV	2~128(even)	100	OSC frequency division coefficient setting

Primitive Instantiation

Verilog Instantiation:

```
OSCH uut(
    .OSCOUT(oscout)
    );
defparam uut.FREQ_DIV=100;
```

Vhdl Instantiation:

```
COMPONENT OSCH
        GENERIC(
                FREQ_DIV:integer:=100;
         );
        PORT(OSCOUT:OUT STD_LOGIC);
END COMPONENT;
uut:OSCH
      GENERIC MAP(
                FREQ_DIV=>100,
       )
      PORT MAP(OSCOUT=>oscout);
```

6.11 DHCEN

Primitive Name

Dynamic HCLK Clock Eanble with Inverted Gate (DHCEN) can dynamically open/close HCLK and breakover in low level.

Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

Figure 6-17 DHCEN Port Diagram

Port Description

Table 6-23 Port Description

Port Name	I/O	Description
CLKIN	input	Clock input
CE	input	Clock enable input, active-low.
CLKOUT	output	Clock output

Primitive Instantiation

Verilog Instantiation:

```
DHCEN dhcen_inst (
    .CLKIN(clkin),
    .CE(ce),
    .CLKOUT(clkout)
);
```

Vhdl Instantiation:

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

6.12 BUFG

Primitive Name

Global Clock Buffer (BUFG)

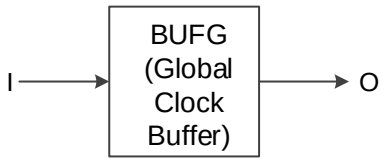
Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B,

GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

Figure 6-18 BUFG Port Diagram



Port Description

Table 6-24 Port Description

Port Name	I/O	Description
O	output	Clock output
I	input	Clock input

Primitive Instantiation

Verilog Instantiation:

```
BUFG uut(
    .O(o),
    .I(i)
    );
```

Vhdl Instantiation:

```
COMPONENT BUFG
        PORT(
                O:OUT std_logic;
                I:IN std_logic
        );
END COMPONENT;
uut:BUFG
        PORT MAP(
            O=>o,
            I=>i
    );
```

6.13 BUFS

Primitive Name

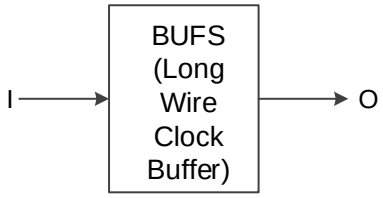
The Long Wire Clock Buffer (BUFS)

Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

Figure 6-19 BUFS Diagram



Port Description

Table 6-25 Port Description

Port Name	I/O	Description
O	output	Clock output
I	input	Clock input

Primitive Instantiation

Verilog Instantiation:

```
BUFS uut(
    .O(o),
    .I(i)
    );
```

Vhdl Instantiation:

```
COMPONENT BUFS
        PORT(
                O:OUT std_logic;
                I:IN std_logic
            );
END COMPONENT;
uut:BUFS
      PORT MAP(
          O=>o,
          I=>i
        );
```

7 User Flash

7.1 FLASH96K

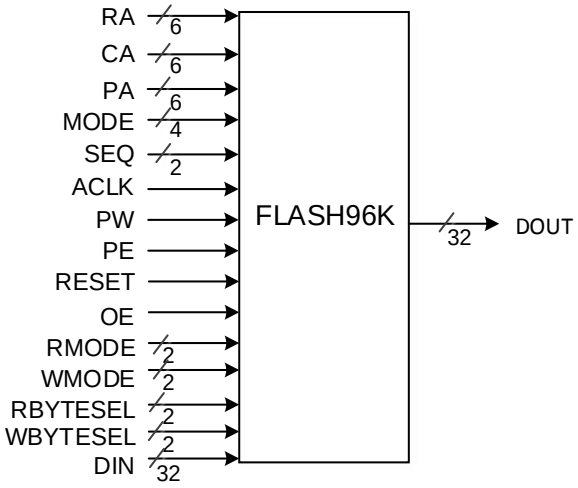
Primitive Introduction

96Kbit User Flash (FLASH96K) has a memory space of 96Kbit. The width and depth of the register are constant and cannot be configured. Its width is 4 bytes (32 bits) and the address depth is 3k. It has non-volatile and power-off saving functions, but without initial value of BSRAM function.

Devices supported: GW1N-1 and GW1N-1S.

Block Diagram

Figure 7-1 FLASH96K Blcok Diagram



Port Description

Table 7-1 Port Description

Port Name	I/O	Description
DOUT[31:0]	Output	Data Output
DIN[31:0]	Input	Data Input
RA[5:0]	Input	Row Address
CA[5:0]	Input	Column Address

Port Name	I/O	Description
PA[5:0]	Input	Page latch Address
MODE[3:0]	Input	Operation mode select
SEQ[1:0]	Input	NV operation sequence control
ACLK	Input	Synchronous clock for read and write operation
PW	Input	Write page latch clock
RESET	Input	Macro reset
PE	Input	Pump enable
OE	Input	Output enable
RMODE[1:0]	Input	Read out bit width select
WMODE[1:0]	Input	Write in bit width select
RBYTESEL[1:0]	Input	Read data Byte address
WBYTESEL[1:0]	Input	Write data Byte address

Primitive Instantiation

Verilog Instantiation:

```
FLASH96K flash96k_inst(
        .RA(ra[5:0]),
        .CA(ca[5:0]),
        .PA(pa[5:0]),
        .MODE(mode[3:0]),
        .SEQ(seq[1:0]),
        .ACLK(aclk),
        .PW(pw),
        .RESET(reset),
        .PE(pe),
        .OE(oe),
        .RMODE(rmode[1:0]),
        .WMODE(wmode[1:0]),
        .RBYTESEL(rbytesel[1:0]),
        .WBYTESEL(wbytesel[1:0]),
        .DIN(din[31:0]),
        .DOUT(dout[31:0])
);
```

Vhdl Instantiation:

```
COMPONENT FLASH96K
    PORT(
        RA:IN std_logic_vector(5 downto 0);
            CA:IN std_logic_vector(5 downto 0);
            PA:IN std_logic_vector(5 downto 0);
            MODE:IN std_logic_vector(3 downto 0);
            SEQ:IN std_logic_vector(1 downto 0);
            ACLK:IN std_logic;
            PW:IN std_logic;
            RESET:IN std_logic;
```

```
                    PE:IN std_logic;
                    OE:IN std_logic;
                    RMODE:IN std_logic_vector(1 downto 0);
                    WMODE:IN std_logic_vector(1 downto 0);
                    RBYTESEL:IN std_logic_vector(1 downto 0);
                    WBYTESEL:IN std_logic_vector(1 downto 0);
                    DIN:IN std_logic_vector(31 downto 0);
                    DOUT:OUT std_logic_vector(31 downto 0)
                );
        END COMPONENT;
        uut:   FLASH96K
          PORT MAP (
                    RA=>ra,
                    CA=>ca,
                    PA=>pa,
                    MODE=>mode,
                    SEQ=>seq,
                    RESET=>reset,
                    ACLK=>aclk,
                    PW=>pw,
                    PE=>pe,
                    OE=>oe,
                    RMODE=>rmode,
                    WMODE=>wmode,
    RBYTESEL=>rbytesel,
WBYTESEL=> wbytesel,
DIN=>din,
DOUT=>dout
                );
```

7.2 FLASH64KZ

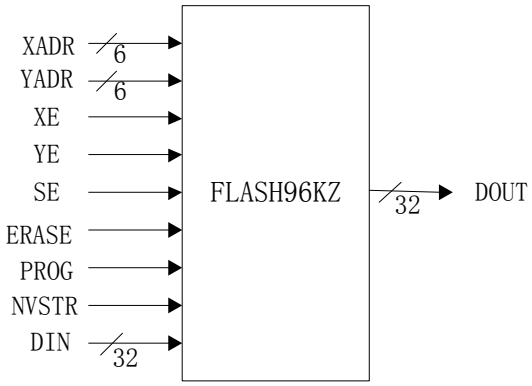
Primitive Introduction

64Kbit User Flash (FLASH64KZ) has a memory space of 64Kbit. The width and depth of the register are constant and cannot be configured. It has non-volatile and power-off saving functions, but without the initial value of BSRAM function.

Devices supported: GW1NZ-1 device.

Block Diagram

Figure 7-2 FLASH64KZ Block Diagram



Port Description

Table 7-2 Port Description

Port Name	I/O	Description
DOUT[31:0]	Output	Data Output
DIN[31:0]	Input	Data Input
XADR[4:0]	Input	X address input
YADR[5:0]	Input	Y address input
XE	Input	X address enable
YE	Input	Y address enable
SE	Input	Sense amplifier enable
ERASE	Input	Defines erase cycle
PROG	Input	Defines program cycle
NVSTR	Input	Defines non-volatile store cycle

Primitive Instantiation

Verilog Instantiation:

```
FLASH64KZ flash64kz_inst(
        .XADR(xadr[4:0]),
        .YADR(yadr[5:0]),
        .XE(xe),
        .YE(ye),
        .SE(se),
        .ERASE(erase),
        .PROG(prog),
        .NVSTR(nvstr),
        .DIN(din[31:0]),
        .DOUT(dout[31:0])
);
```

Vhdl Instantiation:

```
COMPONENT FLASH64KZ
    PORT(
```

```
        XADR:IN std_logic_vector(4 downto 0);
        YADR:IN std_logic_vector(5 downto 0);
                XE:IN std_logic;
                YE:IN std_logic;
                SE:IN std_logic;
                ERASE:IN std_logic;
                PROG:IN std_logic;
                NVSTR:IN std_logic;
                DIN:IN std_logic_vector(31 downto 0);
                DOUT:OUT std_logic_vector(31 downto 0)
            );
    END COMPONENT;
    uut:   FLASH64KZ
      PORT MAP (
                XADR=>xadr,
                YADR=>yadr,
                XE=>xe,
                YE=>ye,
                SE=>se,
                ERASE=>erase,
                PROG=>prog,
                NVSTR=>nvstr,
               DIN=>din,
DOUT=>dout
            );
```

7.3 FLASH128K

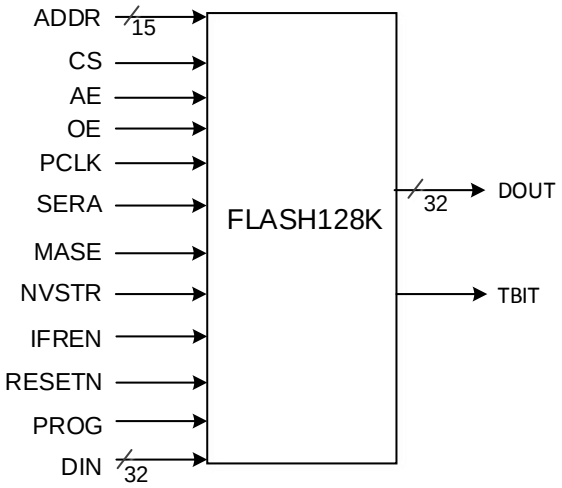
Primitive Introduction

128KByte User Flash (FLASH128K) has a memory space of 128Kbit. The width and depth of the register are constant and cannot be configured. It has non-volatile and power-off saving functions, but without the initial value of BSRAM function.

Devices supported: GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C.

Block Diagram

Figure 7-3 FLASH128K Block Diagram



Port Description

Table 7-3 Port Description

Port Name	I/O	Description
DOUT[31:0]	Output	Data Output
TBIT	Output	Indicator of write or erase
DIN[31:0]	Input	Data Input
ADDR[14:0]	Input	Address Input
CS	Input	Chip enable
AE	Input	Address enable
OE	Input	Output enable
PCLK	Input	Clock input
PROG	Input	Defines program cycle
SERA	Input	Sector erase signal
MASE	Input	Chip erase signal
NVSTR	Input	Defines non-volatile store cycle
IFREN	Input	Flash IP information page Selection
RESETN	Input	Power On Reset Input

Primitive Instantiation

Verilog Instantiation:

```
FLASH128K flash128k_inst(
    .ADDR(addr[14:0]),
    .CS(cs),
    .AE(ae),
    .OE(oe),
    .PCLK(pclk),
```

```
        .PROG(prog),
        .SERA(sera),
        .MASE(mase),
        .NVSTR(nvstr),
        .IFREN(ifren),
        .RESETN(resetn),
        .DIN(din[31:0]),
        .DOUT(dout[31:0]),
        .TBIT(tbit)
    );
```

Vhdl Instantiation:

```
    COMPONENT FLASH128K
        PORT(
        DIN:IN std_logic_vector(31 downto 0);
        ADDR:IN std_logic_vector(14 downto 0);
                CS:IN std_logic;
                AE:IN std_logic;
                OE:IN std_logic;
                PCLK:IN std_logic;
                PROG:IN std_logic;
                SERA:IN std_logic;
                MASE:IN std_logic;
                NVSTR:IN std_logic;
                IFREN:IN std_logic;
                RESETN:IN std_logic;
                DOUT:OUT std_logic_vector(31 downto 0);
                TBIT:OUT std_logic;
            );
    END COMPONENT;
    uut:   FLASH128K
      PORT MAP (
        DIN=>din,
                ADDR=>addr,
                CS=>cs,
                AE=>ae,
                OE=>oe,
                PCLK=>pclk,
                PROG=>prog,
                SERA=>sera,
                MASE=>mase,
                NVSTR=>nvstr,
                IFREN=>ifren,
                RESETN=>resetn,
DOUT=>dout,
TBIT=>tbit
        );
```

7.4 FLASH256K

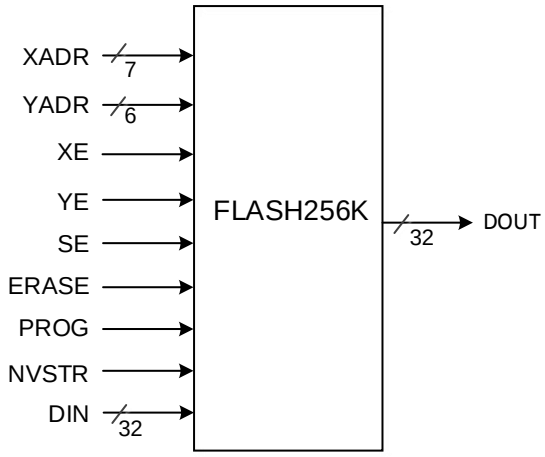
Primitive Introduction

256Kbit User Flash (FLASH256K) has a memory space of 256Kbit. The width and depth of the register are constant and cannot be configured. It has non-volatile and power-off saving functions, but without the initial value of BSRAM function.

Devices supported: GW1N-2, GW1N-2B, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C.

Block Diagram

Figure 7-4 FLASH256K Block Diagram



Port Description

Table 7-4 Port Description

Port Name	I/O	Description
DOUT[31:0]	Output	Data Output
DIN[31:0]	Input	Data Input
XADR[6:0]	Input	X address input
YADR[5:0]	Input	Y address input
XE	Input	X address enable
YE	Input	Y address enable
SE	Input	Sense amplifier enable
PROG	Input	Defines program cycle
ERASE	Input	Defines erase cycle
NVSTR	Input	Defines non-volatile store cycle

Primitive Instantiation

Verilog Instantiation:

```
FLASH256K flash256k_inst(
    .XADR(xadr[6:0]),
```

```
        .YADR(yadr[5:0]),
        .XE(xe),
        .YE(ye),
        .SE(se),
        .ERASE(erase),
        .PROG(prog),
        .NVSTR(nvstr),
        .DIN(din[31:0]),
        .DOUT(dout[31:0])
    );
```

Vhdl Instantiation:

```
    COMPONENT FLASH256K
        PORT(
        DIN:IN std_logic_vector(31 downto 0);
        XADR:IN std_logic_vector(6 downto 0);
                YADR:IN std_logic_vector(5 downto 0);
                XE:IN std_logic;
                YE:IN std_logic;
                SE:IN std_logic;
                ERASE:IN std_logic;
                PROG:IN std_logic;
                NVSTR:IN std_logic;
                DOUT:OUT std_logic_vector(31 downto 0)
         );
    END COMPONENT;
    uut:   FLASH256K
      PORT MAP (
        DIN=>din,
                XADR=>xadr,
                YADR=>yadr,
                XE=>xe,
                YE=>ye,
                SE=>se,
        ERASE=>erase,
                PROG=>prog,
                NVSTR=>nvstr,
DOUT=>dout
                );
```

7.5 FLASH608K

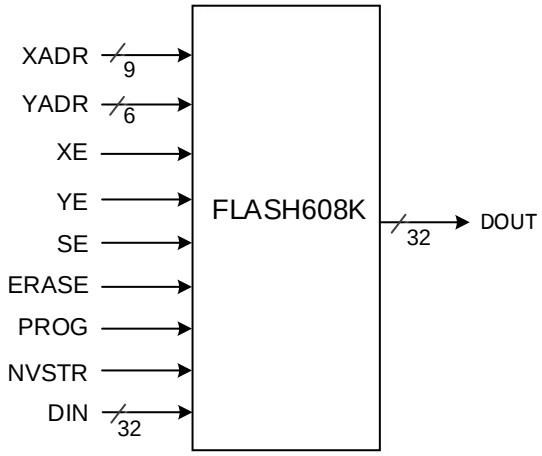
Primitive Introduction

608Kbit User Flash (FLASH608K) has a memory space of 608Kbit. The width and depth of the register are constant and cannot be configured. It has non-volatile and power-off saving functions, but without the initial value of BSRAM function.

supports the GW1N-6, GW1N-9, and GW1NR-9 devices.

Block Diagram

Figure 7-5 FLASH608K Block Diagram



Port Description

Table 7-5 Port Description

Port Name	I/O	Description
DOUT[31:0]	Output	Data Output
DIN[31:0]	Input	Data Input
XADR[8:0]	Input	X address input
YADR[5:0]	Input	Y address input
XE	Input	X address enable
YE	Input	Y address enable
SE	Input	Sense amplifier enable
PROG	Input	Defines program cycle
ERASE	Input	Defines erase cycle
NVSTR	Input	Defines non-volatile store cycle

Primitive Instantiation

Verilog Instantiation:

```
FLASH608K flash608k_inst(
    .XADR(xadr[8:0]),
    .YADR(yadr[5:0]),
    .XE(xe),
    .YE(ye),
    .SE(se),
    .ERASE(erase),
    .PROG(prog),
    .NVSTR(nvstr),
    .DIN(din[31:0]),
    .DOUT(dout[31:0])
);
```

Vhdl Instantiation:

```
COMPONENT FLASH608K
    PORT(
    DIN:IN std_logic_vector(31 downto 0);
    XADR:IN std_logic_vector(8 downto 0);
            YADR:IN std_logic_vector(5 downto 0);
            XE:IN std_logic;
            YE:IN std_logic;
            SE:IN std_logic;
            ERASE:IN std_logic;
            PROG:IN std_logic;
            NVSTR:IN std_logic;
            DOUT:OUT std_logic_vector(31 downto 0)
     );
END COMPONENT;
uut:   FLASH608K
   PORT MAP (
    DIN=>din,
            XADR=>xadr,
            YADR=>yadr,
            XE=>xe,
            YE=>ye,
            SE=>se,
    ERASE=>erase,
            PROG=>prog,
            NVSTR=>nvstr,
DOUT=>dout
    );
```

8 EMPU

8.1 MCU

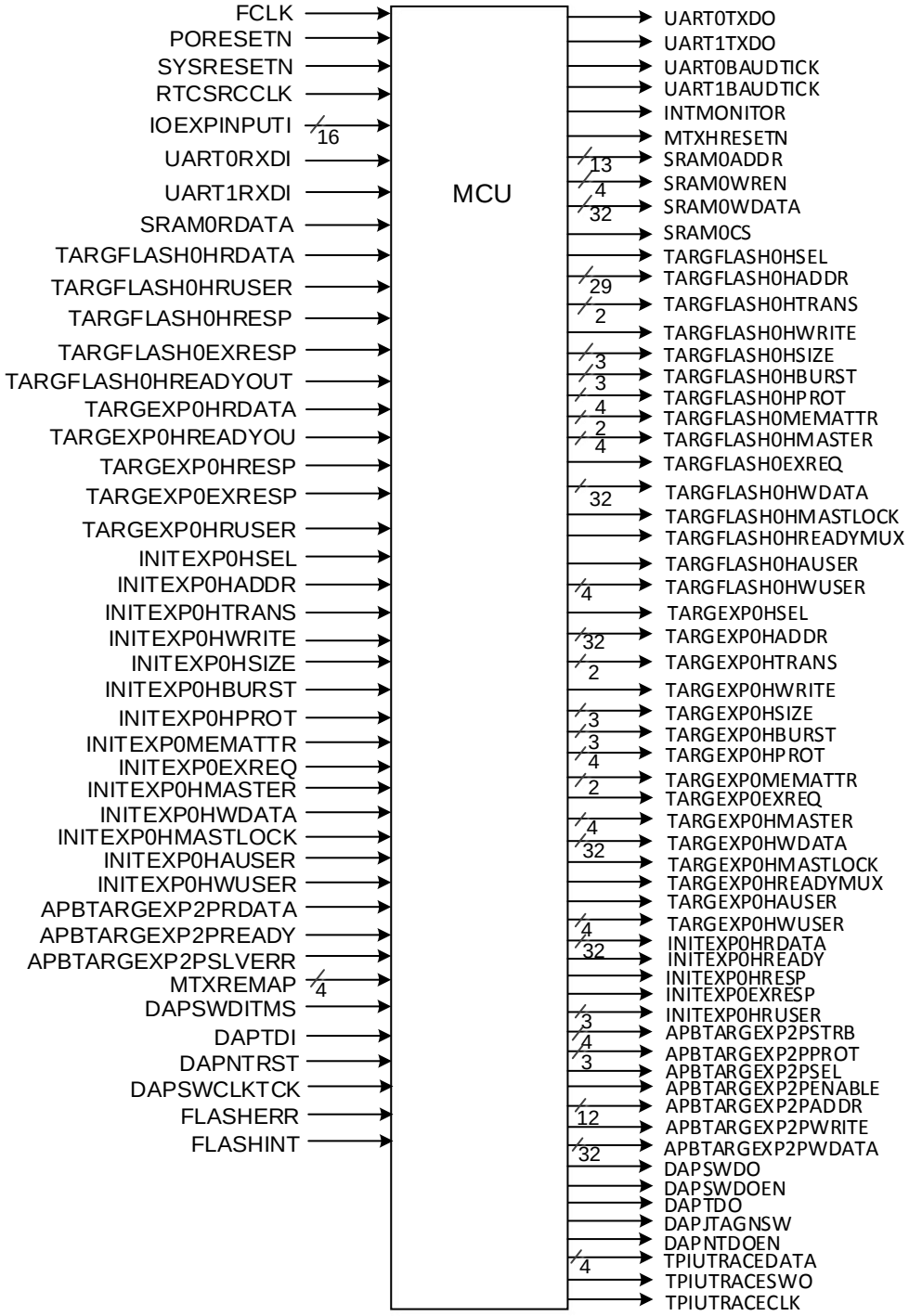
Primitive Introduction

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

Devices supported: GW1NS-2C, GW1NSR-2C, GW1NSE-2C.

Block Diagram

Figure 8-1 MCU Block Diagram



Port Description

Table 8-1 Port Description

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

Port Name	I/O	Description
SYSRESETN	input	System reset
RTCSRCCLK	input	Used to generate RTC clock
IOEXPINPUTI[15:0]	input	IOEXPINPUTI
UART0RXDI	input	UART0RXDI
UART1RXDI	input	UART1RXDI
SRAM0RDATA[31:0]	input	SRAM Read data bus
TARGFLASH0HRDATA[31:0]	input	TARGFLASH0, HRDATA
TARGFLASH0HRUSER[2:0]	input	TARGFLASH0, HRUSER
TARGFLASH0HRESP	input	TARGFLASH0, HRESP
TARGFLASH0EXRESP	input	TARGFLASH0, EXRESP
TARGFLASH0HREADYOUT	input	TARGFLASH0, EXRESP
TARGEXP0HRDATA[31:0]	input	TARGEXP0, HRDATA
TARGEXP0HREADYOUT	input	TARGEXP0, HREADY
TARGEXP0HRESP	input	TARGEXP0, HRESP
TARGEXP0EXRESP	input	TARGEXP0, EXRESP
TARGEXP0HRUSER[2:0]	input	TARGEXP0, HRUSER
INITEXP0HSEL	input	INITEXP0, HSELx
INITEXP0HADDR[31:0]	input	INITEXP0, HADDR
INITEXP0HTRANS[1:0]	input	INITEXP0, HTRANS
INITEXP0HWRITE	input	INITEXP0, HWRITE
INITEXP0HSIZE[2:0]	input	INITEXP0, HSIZE
INITEXP0HBURST[2:0]	input	INITEXP0, HBURST
INITEXP0HPROT[3:0]	input	INITEXP0, HPROT
INITEXP0MEMATTR[1:0]	input	INITEXP0, MEMATTR
INITEXP0EXREQ	input	INITEXP0, EXREQ
INITEXP0HMASTER[3:0]	input	INITEXP0, HMASTER
INITEXP0HWDATA[31:0]	input	INITEXP0, HWDATA
INITEXP0HMASTLOCK	input	INITEXP0, HMASTLOCK
INITEXP0HAUSER	input	INITEXP0, HAUSER
INITEXP0HWUSER[3:0]	input	INITEXP0, HWUSER
APBTARGEXP2PRDATA[31:0]	input	APBTARGEXP2, PRDATA
APBTARGEXP2PREADY	input	APBTARGEXP2, PREADY
APBTARGEXP2PSLVERR	input	APBTARGEXP2, PSLVERR
MTXREMAP[3:0]	input	The MTXREMAP signals control the remapping of the boot memory range.
DAPSWDITMS	input	Debug TMS
DAPTDI	input	Debug TDI
DAPNTRST	input	Test reset
DAPSWCLKTCK	input	Test clock / SWCLK
FLASHERR	input	Output clock, used by the TPA to sample the other pins
FLASHINT	input	Output clock, used by the TPA to sample the other pins
IOEXPOUTPUTO[15:0]	output	IOEXPOUTPUTO
IOEXPOUTPUTENO[15:0]	output	IOEXPOUTPUTENO
UART0TXDO	output	UART0TXDO
UART1TXDO	output	UART1TXDO
UART0BAUDTICK	output	UART0BAUDTICK

Port Name	I/O	Description
UART1BAUDTICK	output	UART1BAUDTICK
INTMONITOR	output	INTMONITOR
MTXHRESETN	output	SRAM/Flash Chip reset
SRAM0ADDR[12:0]	output	SRAM address
SRAM0WREN[3:0]	output	SRAM Byte write enable
SRAM0WDATA[31:0]	output	SRAM Write data
SRAM0CS	output	SRAM Chip select
TARGFLASH0HSEL	output	TARGFLASH0, HSELx
TARGFLASH0HADDR[28:0]	output	TARGFLASH0, HADDR
TARGFLASH0HTRANS[1:0]	output	TARGFLASH0, HTRANS
TARGFLASH0HWRITE	output	TARGFLASH0, HWRITE
TARGFLASH0HSIZE[2:0]	output	TARGFLASH0, HSIZE
TARGFLASH0HBURST[2:0]	output	TARGFLASH0, HBURST
TARGFLASH0HPROT[3:0]	output	TARGFLASH0, HPROT
TARGFLASH0MEMATTR[1:0]	output	TARGFLASH0, MEMATTR
TARGFLASH0EXREQ	output	TARGFLASH0, EXREQ
TARGFLASH0HMASTER[3:0]	output	TARGFLASH0, HMASTER
TARGFLASH0HWDATA[31:0]	output	TARGFLASH0, HWDATA
TARGFLASH0HMASTLOCK	output	TARGFLASH0, HMASTLOCK
TARGFLASH0HREADYMUX	output	TARGFLASH0, HREADYOUT
TARGFLASH0HAUSER	output	TARGFLASH0, HAUSER
TARGFLASH0HWUSER[3:0]	output	TARGFLASH0, HWUSER
TARGEXP0HSEL	output	TARGEXP0, HSELx
TARGEXP0HADDR[31:0]	output	TARGEXP0, HADDR
TARGEXP0HTRANS[1:0]	output	TARGEXP0, HTRANS
TARGEXP0HWRITE	output	TARGEXP0, HWRITE
TARGEXP0HSIZE[2:0]	output	TARGEXP0, HSIZE
TARGEXP0HBURST[2:0]	output	TARGEXP0, HBURST
TARGEXP0HPROT[3:0]	output	TARGEXP0, HPROT
TARGEXP0MEMATTR[1:0]	output	TARGEXP0, MEMATTR
TARGEXP0EXREQ	output	TARGEXP0, EXREQ
TARGEXP0HMASTER[3:0]	output	TARGEXP0, HMASTER
TARGEXP0HWDATA[31:0]	output	TARGEXP0, HWDATA
TARGEXP0HMASTLOCK	output	TARGEXP0, HMASTLOCK
TARGEXP0HREADYMUX	output	TARGEXP0, HREADYOUT
TARGEXP0HAUSER	output	TARGEXP0, HAUSER
TARGEXP0HWUSER[3:0]	output	TARGEXP0, HWUSER
INITEXP0HRDATA[31:0]	output	INITEXP0, HRDATA
INITEXP0HREADY	output	INITEXP0, HREADY
INITEXP0HRESP	output	INITEXP0, HRESP
INITEXP0EXRESP	output	INITEXP0,EXRESP
INITEXP0HRUSER[2:0]	output	INITEXP0, HRUSER
APBTARGEXP2PSTRB[3:0]	output	APBTARGEXP2, PSTRB
APBTARGEXP2PPROT[2:0]	output	APBTARGEXP2, PPROT
APBTARGEXP2PSEL	output	APBTARGEXP2, PSELx
APBTARGEXP2PENABLE	output	APBTARGEXP2, PENABLE

Port Name	I/O	Description
APBTARGEXP2PADDR[11:0]	output	APBTARGEXP2, PADDR
APBTARGEXP2PWRITE	output	APBTARGEXP2, PWRITE
APBTARGEXP2PWDATA[31:0]	output	APBTARGEXP2, PWDATA
DAPSWDO	output	Serial Wire Data Out
DAPSWDOEN	output	Serial Wire Output Enable
DAPTDO	output	Debug TDO
DAPJTAGNSW	output	JTAG or Serial-Wire selection JTAG mode(1) or SW mode(0)
DAPNTDOEN	output	TDO output pad control signal
TPIUTRACEDATA[3:0]	output	Output data
TPIUTRACESWO	output	Serial Wire Viewer data
TPIUTRACECLK	output	Output clock, used by the TPA to sample the other pins

Primitive Instantiation

Verilog Instantiation:

```
MCU u_sse050_top_syn (
        .FCLK(fclk),
        .PORESETN(poresetn),
        .SYSRESETN(sysresetn),
        .RTCSRCCLK(rtcsrcclk),
        .IOEXPINPUTI(ioexpinputi[15:0]),
        .IOEXPOUTPUTO(ioexpoutputo[15:0]),
        .IOEXPOUTPUTENO(ioexpoutputeno[15:0]),
        .UART0RXDI(uart0rxdi),
        .UART0TXDO(uart0txdo),
        .UART1RXDI(uart1rxdi),
        .UART1TXDO(uart1txdo),
        .SRAM0RDATA(sram0rdata[31:0]),
        .SRAM0ADDR(sram0addr[12:0]),
    .SRAM0WREN(sram0wren[3:0]),
    .SRAM0WDATA(sram0wdata[31:0]),
    .SRAM0CS(sram0cs),
    .MTXHRESETN(mtxhreset),
    .TARGFLASH0HSEL(targflash0hsel),
  .TARGFLASH0HADDR(targflash0haddr[28:0]),
  .TARGFLASH0HTRANS(targflash0htrans[1:0]),
  .TARGFLASH0HWRITE(targflash0hwrite),
  .TARGFLASH0HSIZE(targflash0hsize[2:0]),
  .TARGFLASH0HBURST(targflash0hburst[2:0]),
  .TARGFLASH0HPROT(targflash0hprot[3:0]),
  .TARGFLASH0MEMATTR(targflash0mematttr[1:0]),
  .TARGFLASH0EXREQ(targflash0exreq),
  .TARGFLASH0HMASTER(targflash0hmaster[3:0]),
  .TARGFLASH0HWDATA(targflash0hwdata[31:0]),
  .TARGFLASH0HMASTLOCK(targflash0hmastlock),
  .TARGFLASH0HREADYMUX(targflash0hreadymux),
  .TARGFLASH0HAUSER(targflash0hauser),
```

```
.TARGFLASH0HWUSER(targflash0hwuser[3:0]),
.TARGFLASH0HRDATA(targflash0hrdata[31:0]),
.TARGFLASH0HRUSER(targflash0hruser[2:0]),
.TARGFLASH0HRESP(targflash0hresp),
.TARGFLASH0EXRESP(targflash0exresp),
.TARGFLASH0HREADYOUT(targflash0hreadyout),
.TARGEXP0HSEL(targexp0hsel),
.TARGEXP0HADDR(targexp0haddr[31:0]),
.TARGEXP0HTRANS(targexp0htrans[1:0]),
.TARGEXP0HWRITE(targexp0hwrite),
.TARGEXP0HSIZE(targexp0hsize[2:0]),
.TARGEXP0HBURST(targexp0hburst[2:0]),
.TARGEXP0HPROT(targexp0hprot[3:0]),
.TARGEXP0MEMATTR(targexp0mematttr[1:0]),
.TARGEXP0EXREQ(targexp0exreq),
.TARGEXP0HMASTER(targexp0hmaster[3:0]),
.TARGEXP0HWDATA(targexp0hwdata[31:0]),
.TARGEXP0HMASTLOCK(targexp0hmastlock),
.TARGEXP0HREADYMUX(targexp0hreadymux),
.TARGEXP0HAUSER(targexp0hauser),
.TARGEXP0HWUSER(targexp0hwuser[3:0]),
.TARGEXP0HRDATA(targexp0hrdata[31:0]),
.TARGEXP0HREADYOUT(targexp0hreadyout),
.TARGEXP0HRESP(targexp0hresp),
.TARGEXP0EXRESP(targexp0exresp),
.TARGEXP0HRUSER(targexp0hruser[2:0]),
.INITEXP0HSEL(initexp0hsel),
.INITEXP0HADDR(initexp0haddr[31:0]),
.INITEXP0HTRANS(initexp0htrans[1:0]),
.INITEXP0HWRITE(initexp0hwrite),
.INITEXP0HSIZE(initexp0hsize[2:0]),
.INITEXP0HBURST(initexp0hburst[2:0]),
.INITEXP0HPROT(initexp0hprot[3:0]),
.INITEXP0MEMATTR(initexp0mematttr[1:0]),
.INITEXP0EXREQ(initexp0exreq),
.INITEXP0HMASTER(initexp0hmaster[3:0]),
.INITEXP0HWDATA(initexp0hwdata[31:0]),
.INITEXP0HMASTLOCK(initexp0hmastlock),
.INITEXP0HAUSER(initexp0hauser),
.INITEXP0HWUSER(initexp0hwuser[3:0]),
.INITEXP0HRDATA(initexp0hrdata[31:0]),
.INITEXP0HREADY(initexp0hready),
.INITEXP0HRESP(initexp0hresp),
.INITEXP0EXRESP(initexp0exresp),
.INITEXP0HRUSER(initexp0hruser[2:0]),
.APBTARGEXP2PSEL(apbtargexp2psel),
.APBTARGEXP2PENABLE(apbtargexp2penable),
.APBTARGEXP2PADDR(apbtargexp2paddr[11:0]),
.APBTARGEXP2PWRITE(apbtargexp2pwrite),
.APBTARGEXP2PWDATA(apbtargexp2pwdata[31:0]),
```

```
  .APBTARGEXP2PRDATA(apbtargexp2prdata[31:0]),
  .APBTARGEXP2PREADY(apbtargexp2pready),
  .APBTARGEXP2PSLVERR(apbtargexp2pslverr),
  .APBTARGEXP2PSTRB(apbtargexp2pstrb[3:0]),
  .APBTARGEXP2PPROT(apbtargexp2pprot[2:0]),
  .MTXREMAP(mtxremap[3:0]),
  .DAPSWDITMS(dapswditms),
  .DAPSWDO(dapswdo),
  .DAPSWDOEN(dapswdoen),
  .DAPTDI(daptdi),
  .DAPTDO(daptdo),
  .DAPNTRST(dapntrst),
  .DAPSWCLKTCK(dapswclk_tck),
  .DAPNTDOEN(dapntdoen),
  .DAPJTAGNSW(dapjtagnsw),
  .TPIUTRACEDATA(tpiutracedata[3:0]),
  .TPIUTRACESWO(tpiutraceswo),
  .TPIUTRACECLK(tpiutraceclk),
  .FLASHERR(flasherr),
  .FLASHINT(flashint)
);
```

Vhdl Instantiation:

```
COMPONENT MCU
    PORT(
FCLK:IN std_logic;
PORESETN:IN std_logic;
SYSRESETN:IN std_logic;
RTCSRCCLK:IN std_logic;
UART0RXDI:IN std_logic;
UART1RXDI:IN std_logic;
CLK:IN std_logic;
RESET:IN std_logic;
IOEXPINPUTI:IN std_logic_vector(15 downto 0);
SRAM0RDATA:IN std_logic_vector(31 downto 0);
TARGFLASH0HRDATA:IN std_logic_vector(31 downto 0);
TARGFLASH0HRUSER:IN std_logic_vector(2 downto 0);
TARGFLASH0HRESP:IN std_logic;
TARGFLASH0EXRESP:IN std_logic;
TARGFLASH0HREADYOUT:IN std_logic;
TARGEXP0HRDATA:   IN std_logic_vector(31 downto 0);
TARGEXP0HREADYOUT:IN std_logic;
TARGEXP0HRESP:IN std_logic;
TARGEXP0EXRESP:IN std_logic;
TARGEXP0HRUSER:   IN std_logic_vector(2 downto 0);
INITEXP0HSEL:IN std_logic;
INITEXP0HADDR:   IN std_logic_vector(31 downto 0);
INITEXP0HTRANS:   IN std_logic_vector(1 downto 0);
INITEXP0HWRITE:   IN std_logic;
INITEXP0HSIZE:   IN std_logic_vector(2 downto 0);
INITEXP0HBURST:   IN std_logic_vector(2 downto 0);
```

```
INITEXP0HPROT:    IN std_logic_vector(3 downto 0);
INITEXP0MEMATTR:    IN std_logic_vector(1 downto 0);
INITEXP0EXREQ:    IN std_logic;
INITEXP0HMASTER:    IN std_logic_vector(3 downto 0);
INITEXP0HWDATA:    IN std_logic_vector(31 downto 0);
INITEXP0HMASTLOCK:    IN std_logic;
INITEXP0HAUSER:    IN std_logic;
INITEXP0HWUSER:    IN std_logic_vector(3 downto 0);
APBTARGEXP2PRDATA:    IN std_logic_vector(3 downto 0);
APBTARGEXP2PREADY:    IN std_logic;
APBTARGEXP2PSLVERR:    IN std_logic;
MTXREMAP:    IN std_logic_vector(3 downto 0);
DAPSWDITMS:    IN std_logic;
DAPTDI:    IN std_logic;
DAPNTRST:    IN std_logic;
DAPSWCLKTCK:    IN std_logic;
FLASHERR:    IN std_logic;
FLASHINT:    IN std_logic;
IOEXPOUTPUTO:OUT std_logic_vector(15 downto 0);
IOEXPOUTPUTENO:OUT std_logic_vector(15 downto 0);
IOEXPINPUTI:OUT std_logic_vector(15 downto 0);
UART0TXDO:    OUT std_logic;
UART1TXDO:    OUT std_logic;
UART0BAUDTICK:    OUT std_logic;
UART1BAUDTICK:    OUT std_logic;
INTMONITOR:    OUT std_logic;
MTXHRESETN:    OUT std_logic;
SRAM0ADDR:OUT std_logic_vector(12 downto 0);
SRAM0WREN:OUT std_logic_vector(3 downto 0);
SRAM0WDATA:OUT std_logic_vector(31 downto 0);
SRAM0CS:    OUT std_logic;
TARGFLASH0HSEL:    OUT std_logic;
TARGFLASH0HWRITE:    OUT std_logic;
TARGFLASH0EXREQ:    OUT std_logic;
TARGFLASH0HMASTLOCK:    OUT std_logic;
TARGFLASH0HREADYMUX:    OUT std_logic;
TARGFLASH0HAUSER:    OUT std_logic;
SRAM0RDATA:OUT std_logic_vector(31 downto 0);
TARGFLASH0HADDR:OUT std_logic_vector(28 downto 0);
TARGFLASH0HTRANS:OUT std_logic_vector(1 downto 0);
TARGFLASH0HSIZE:OUT std_logic_vector(2 downto 0);
TARGFLASH0HBURST:OUT std_logic_vector(2 downto 0);
TARGFLASH0HPROT:OUT std_logic_vector(3 downto 0);
TARGFLASH0MEMATTR:OUT std_logic_vector(1 downto 0);
TARGFLASH0HMASTER:OUT std_logic_vector(3 downto 0);
TARGFLASH0HWDATA:OUT std_logic_vector(31 downto 0);
TARGFLASH0HWUSER:OUT std_logic_vector(3 downto 0);
TARGFLASH0HRDATA:OUT std_logic_vector(31 downto 0);
TARGEXP0HADDR:OUT std_logic_vector(31 downto 0);
TARGEXP0HSEL:    OUT std_logic;
```

```
        TARGEXP0HWRITE:    OUT std_logic;
        TARGEXP0EXREQ:    OUT std_logic;
        TARGEXP0HMASTLOCK:    OUT std_logic;
        TARGEXP0HREADYMUX:    OUT std_logic;
        TARGEXP0HAUSER:    OUT std_logic;
        INITEXP0HREADY:    OUT std_logic;
        INITEXP0HRESP:    OUT std_logic;
        INITEXP0EXRESP:    OUT std_logic;
        TARGEXP0HTRANS:OUT std_logic_vector(1 downto 0);
        TARGEXP0HSIZE:OUT std_logic_vector(2 downto 0);
        TARGEXP0HBURST:OUT std_logic_vector(2 downto 0);
        TARGEXP0HPROT:OUT std_logic_vector(3 downto 0);
        TARGEXP0MEMATTR:OUT std_logic_vector(1 downto 0);
        TARGEXP0HMASTER:OUT std_logic_vector(3 downto 0);
        TARGEXP0HWDATA:OUT std_logic_vector(31 downto 0);
        TARGEXP0HWUSER:OUT std_logic_vector(3 downto 0);
        INITEXP0HRDATA:OUT std_logic_vector(31 downto 0);
        INITEXP0HRUSER:OUT std_logic_vector(2 downto 0);
        APBTARGEXP2PSTRB:OUT std_logic_vector(3 downto 0);
        APBTARGEXP2PPROT:OUT std_logic_vector(2 downto 0);
        APBTARGEXP2PADDR:OUT std_logic_vector(11 downto 0);
        APBTARGEXP2PWDATA:OUT std_logic_vector(31 downto 0);
        TPIUTRACEDATA:OUT std_logic_vector(3 downto 0);
        APBTARGEXP2PSEL:    OUT std_logic;
        APBTARGEXP2PENABLE:    OUT std_logic;
        APBTARGEXP2PWRITE:    OUT std_logic;
        DAPSWDO:    OUT std_logic;
        DAPSWDOEN:    OUT std_logic;
        DAPTDO:    OUT std_logic;
        DAPJTAGNSW:    OUT std_logic;
        DAPNTDOEN:    OUT std_logic;
        TPIUTRACESWO:    OUT std_logic;
        TPIUTRACECLK:    OUT std_logic;
    );
        END COMPONENT;

        uut:    MCU
          PORT MAP (
        FCLK=> fclk;
        PORESETN=> poresetn;
        SYSRESETN=> sysresetn;
        RTCSRCCLK=> rtcsrcclk;
        UART0RXDI=> uart0rxdi;
        UART1RXDI=> uart1rxdi;
        CLK=>clk,
        RESET=>reset,
        IOEXPINPUTI=>ioexpinputi,
        SRAM0RDATA=>sram0rdata,
        TARGFLASH0HRDATA=>targflash0hrdata,
        TARGFLASH0HRUSER=>targflash0hruser,
```

TARGFLASH0HRESP=>targflash0hresp,
TARGFLASH0EXRESP=>targflash0exresp,
TARGFLASH0HREADYOUT=>targflash0hreadyout,
TARGEXP0HRDATA=>targexp0hrdata,
TARGEXP0HREADYOUT=>targexp0hreadyout,
TARGEXP0HRESP=>targexp0hresp,
TARGEXP0EXRESP=>targexp0exresp,
TARGEXP0HRUSER=>targexp0hruser,
INITEXP0HSEL=>initexp0hsel,
INITEXP0HADDR=>initexp0haddr,
INITEXP0HTRANS=>initexp0htrans,
INITEXP0HWRITE=>initexp0hwrite,
INITEXP0HSIZE=>initexp0hsize,
INITEXP0HBURST=>initexp0hburst,
INITEXP0HPROT=>initexp0hprot,
INITEXP0MEMATTR=>initexp0memattr,
INITEXP0EXREQ=>initexp0exreq,
INITEXP0HMASTER=>initexp0hmaster,
INITEXP0HWDATA=>initexp0hwdata,
INITEXP0HMASTLOCK=>initexp0hmastlock,
INITEXP0HAUSER=>initexp0hauser,
INITEXP0HWUSER=>initexp0hwuser,
APBTARGEXP2PRDATA=>apbtargexp2prdata,
APBTARGEXP2PREADY=>apbtargexp2pready,
APBTARGEXP2PSLVERR=>apbtargexp2pslverr,
MTXREMAP=>mtxremap,
DAPSWDITMS=>dapswditms,
DAPTDI=>daptdi,
DAPNTRST=>dapntrst,
DAPSWCLKTCK=>dapswclktck,
FLASHERR=>flasherr,
FLASHINT=>flashint,
IOEXPOUTPUTO=>ioexpoutputo,
IOEXPOUTPUTENO=>ioexpoutputeno,
IOEXPINPUTI=>ioexpinputi,
UART0TXDO=>uart0txdo,
UART1TXDO=>uart1txdo,
UART0BAUDTICK=>uart0baudtick,
UART1BAUDTICK=>uart1baudtick,
INTMONITOR=>intmonitor,
MTXHRESETN=>mtxhresetn,
SRAM0ADDR=>sram0addr,
SRAM0WREN=>sram0wren,
SRAM0WDATA=>sram0wdata,
SRAM0CS=>sram0cs,
TARGFLASH0HSEL=>targflash0hsel,
TARGFLASH0HWRITE=>targflash0hwrite,
TARGFLASH0EXREQ=>targflash0exreq,
TARGFLASH0HMASTLOCK=>targflash0hmastlock,
TARGFLASH0HREADYMUX=>targflash0hreadymux,

```
TARGFLASH0HAUSER=>targflash0hauser,
SRAM0RDATA=>sram0rdata,
TARGFLASH0HADDR=>targflash0haddr,
TARGFLASH0HTRANS=>targflash0htrans,
TARGFLASH0HSIZE=>targflash0hsize,
TARGFLASH0HBURST=>targflash0hburst,
TARGFLASH0HPROT=>targflash0hprot,
TARGFLASH0MEMATTR=>targflash0memattr,
TARGFLASH0HMASTER=>targflash0hmaster,
TARGFLASH0HWDATA=>targflash0hwdata,
TARGFLASH0HWUSER=>targflash0hwuser,
TARGFLASH0HRDATA=>targflash0hrdata,
TARGEXP0HADDR=>targexp0haddr,
TARGEXP0HSEL=>targexp0hsel,
TARGEXP0HWRITE=>targexp0hwrite,
TARGEXP0EXREQ=>targexp0exreq,
TARGEXP0HMASTLOCK=>targexp0hmastlock,
TARGEXP0HREADYMUX=>targexp0hreadymux,
TARGEXP0HAUSER=>targexp0hauser,
INITEXP0HREADY=>initexp0hready,
INITEXP0HRESP=>initexp0hresp,
INITEXP0EXRESP=>initexp0exresp,
TARGEXP0HTRANS=>targexp0htrans,
TARGEXP0HSIZE=>targexp0hsize,
TARGEXP0HBURST=>targexp0hburst,
TARGEXP0HPROT=>targexp0hprot,
TARGEXP0MEMATTR=>targexp0memattr,
TARGEXP0HMASTER=>targexp0hmaster,
TARGEXP0HWDATA=>targexp0hwdata,
TARGEXP0HWUSER=>targexp0hwuser,
INITEXP0HRDATA=>initexp0hrdata,
INITEXP0HRUSER=>initexp0hruser,
APBTARGEXP2PSTRB=>apbtargexp2pstrb,
APBTARGEXP2PPROT=>apbtargexp2pprot,
APBTARGEXP2PADDR=>apbtargexp2paddr,
APBTARGEXP2PWDATA=>apbtargexp2pwdata,
TPIUTRACEDATA=>tpiutracedata,
APBTARGEXP2PSEL=>apbtargexp2psel,
APBTARGEXP2PENABLE=>apbtargexp2penable,
APBTARGEXP2PWRITE=>apbtargexp2pwrite,
DAPSWDO=>dapswdo,
DAPSWDOEN=>dapswdoen,
DAPTDO=>daptdo,
DAPJTAGNSW=>dapjtagnsw,
DAPNTDOEN=>dapntdoen,
TPIUTRACESWO=>tpiutraceswo,
TPIUTRACECLK=>tpiutraceclk );
```

8.2 USB20_PHY

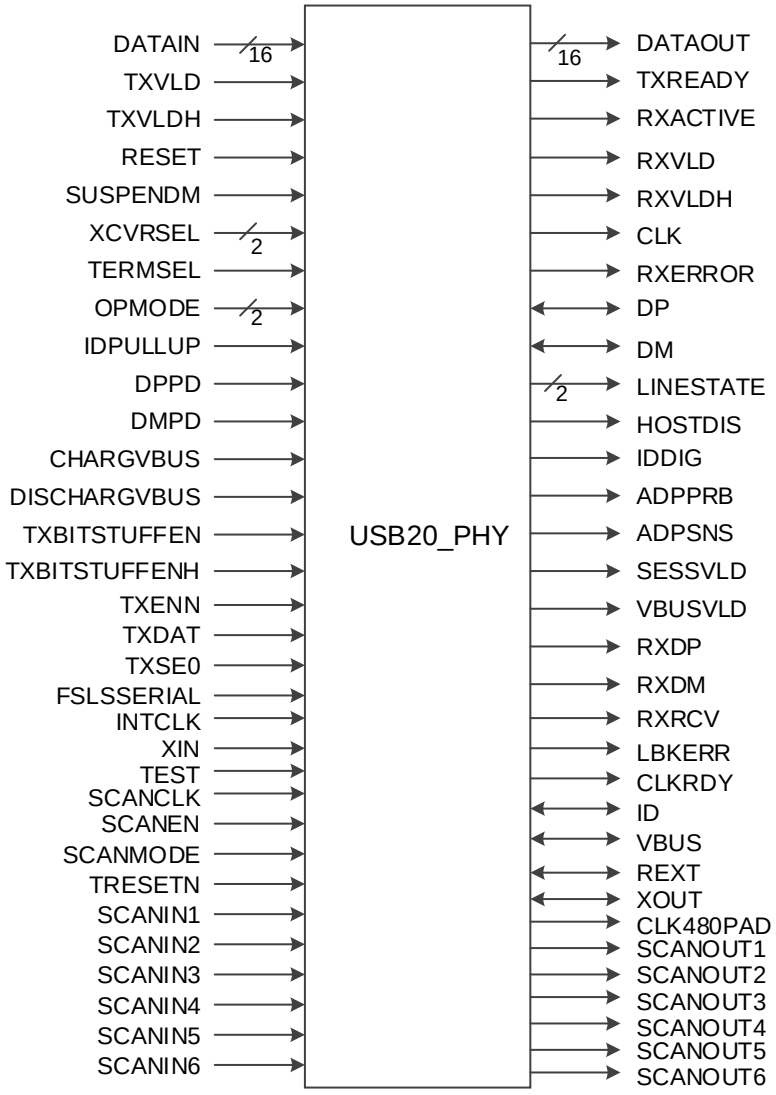
Primitive Introduction

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

Devices supported: GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C.

Block Diagram

Figure 8-2 USB20_PHY Block Diagram



Port Description

Table 8-2 Port Description

Port Name	I/O	Description
DATAIN[15:0]	input	16-bit parallel USB data input bus
TXVLD	input	Transmit Valid. Indicates that the DataIn bus is valid.
TXVLDH	input	Transmit Valid High.When DataBus16_8 = 1, this signal indicates that the DataIn[15:8] bus contains valid transmit data.
RESET	input	Reset. Reset all state machines in the UTM.
SUSPENDM	input	Suspend. 0:suspend, 1: normal
XCVRSEL[1:0]	input	Transceiver Select. This signal selects between the LS, FS and HS transceivers
TERMSEL	input	Termination Select. This signal selects between the FS and HS terminations
OPMODE[1:0]	input	Operational Mode. These signals select between various operational modes
IDPULLUP	input	Signal that enables the sampling of the analog Id line.
DPPD	input	This signal enables the 15k Ohm pull-down resistor on the DP line.
DMPD	input	0b : Pull-down resistor not connected to DM; 1b : Pull-down resistor connected to DM
CHARGVBUS	input	This signal enables charging Vbus
DISCHARGVBUS	input	The signal enables discharging Vbus.
TXBITSTUFFEN	input	Indicates if the data on the DataOut[7:0] lines needs to be bitstuffed or not.
TXBITSTUFFENH	input	Indicates if the data on the DataOut[15:8] lines needs to be bitstuffed or not.
TXENN	input	Active low enable signal. Only used when FsLsSerialMode is set to 1b
TXDAT	input	Differential data at D+/D- output. Only used when FsLsSerialMode is set to 1b
TXSE0	input	Force Single-Ended Zero. Only used when FsLsSerialMode is set to 1b
FSLSSERIAL	input	0b : FS and LS packets are sent using the parallel interface. 1b : FS and LS packets are sent using the serial interface.
INTCLK	input	Clock signals provided internally of the SoC
TEST	input	For IP TESTing purpose.Please leave it unconnected since there are already soft pull-down in the IP
SCANCLK	input	Clock signals for scan mode
SCANEN	input	Select to shift mode
SCANMODE	input	High effective signal to enter scan mode
TRESETN	input	Low effective RESET signal for scan mode
SCANIN1	input	Scan chain input
SCANIN2	input	Scan chain input
SCANIN3	input	Scan chain input
SCANIN4	input	Scan chain input
SCANIN5	input	Scan chain input
SCANIN6	input	Scan chain input
DP	inout	USB data pin Data+
DM	inout	USB data pin Data–
ID	inout	ID signal from the cable
VBUS	inout	Vbus signals connected with the cable
REXT	inout	12.7K High precision resistor

Port Name	I/O	Description
XIN	inout	Crystal in signals, supported range is 12MHZ~24MHZ
XOUT	inout	Crystal out signals
DATAOUT[15:0]	output	DataOut. 16-bit parallel USB data output bus.
TXREADY	output	Transmit Data Ready.
RXACTIVE	output	Receive Active. Indicates that the receive state machine has detected SYNC and is active.
RXVLD	output	Receive Data Valid. Indicates that the DataOut bus has valid data.
RXVLDH	output	Receive Data Valid High.
CLK	output	Clock. This output is used for clocking receive and transmit parallel data.
RXERROR	output	Receive Error.
LINESTATE[1:0]	output	Line State. These signals reflect the current state of the single ended receivers.
HOSTDIS	output	This signal is used for all types of peripherals connected to it.
IDDIG	output	Indicates whether the connected plug is a mini-A or mini-B.
ADPPRB	output	Indicates if the voltage on Vbus (0.6V < Vth < 0.75V).
ADPSNS	output	Indicates if the voltage on Vbus (0.2V < Vth < 0.55V).
SESSVLD	output	Indicates if the session for an A/B-peripheral is valid (0.8V < Vth < 2V).
VBUSVLD	output	Indicates if the voltage on Vbus is at a valid level for operation (4.4V < Vth < 4.75V)
RXDP	output	Single-ended receive data, positive terminal.This signal is only valid if FsLsSerialMode is set to 1b
RXDM	output	Single-ended receive data, negative terminal.This signal is only valid if FsLsSerialMode is set to 1b
RXRCV	output	Receive data.This signal is only valid if FsLsSerialMode is set to 1b
LBKERR	output	used for observation
CLKRDY	output	Observation/debug signal to show that the internal PLL has locked and is ready.
CLK480PAD	output	480MHZ clock output for observation
SCANOUT1	output	Scan chain output
SCANOUT2	output	Scan chain output
SCANOUT3	output	Scan chain output
SCANOUT4	output	Scan chain output
SCANOUT5	output	Scan chain output
SCANOUT6	output	Scan chain output

Attribute Description

Table 8-3 Attribute Introduction

Attribute Name	Default	Description
DATABUS16_8	1’b0	Selects between 8 and 16 bit data transfers.
ADP_PRBEN	1’b0	Enables/disables the ADP Probe comparator
TEST_MODE	5’b0	used for testing and debugging purpose
HSDRV1	1’b0	High speed drive adjustment. Please connect to 0 for normal operation.
HSDRV0	1’b0	High speed drive adjustment. Please connect to 0 for normal operation.
CLK_SEL	1’b0	Clock source selection signal. 0 to select external clock provided by the crystal connected on XIN,

Attribute Name	Default	Description
		XOUT. 1 to select internal clock provided on INTCLK port
M	4’b0	M divider input data bit
N	6'b101000	N divider input data bit
C	2'b01	Control charge pump current input data bit, it supports from 30uA (00) to 60uA (11).
FOC_LOCK	1'b0	0: LOCK is generated by PLL lock detector. 1: LOCK is always high(always lock)

Primitive Instantiation

Verilog Instantiation:

```
USB20_PHY usb20_phy_inst (
.DATAOUT(dataout[15:0]),
.TXREADY(txready),
.RXACTIVE(rxactive),
.RXVLD(rxvld),
.RXVLDH(rxvldh),
.CLK(clk),
.RXERROR(rxerror),
.DP(dp),
.DM(dm),
.LINESTATE(linestate[1:0]),
.DATAIN(datain[15:0]),
.TXVLD(txvld),
.TXVLDH(txvldh),
.RESET(reset),
.SUSPENDM(suspendm),
.XCVRSEL(xcvrsel[1:0]),
.TERMSEL(termsel),
.OPMODE(opmode[1:0]),
.HOSTDIS(hostdis),
.IDDIG(iddig),
.ADPPRB(adpprb),
.ADPSNS(adpsns),
.SESSVLD(sessvld),
.VBUSVLD(vbusvld),
.RXDP(rxdp),
.RXDM(rxdm),
.RXRCV(rxrcv),
.IDPULLUP(idpullup),
.DPPD(dppd),
.DMPD(dmpd),
.CHARGVBUS(chargvbus),
.DISCHARGVBUS(dischargvbus),
.TXBITSTUFFEN(txbitstuffen),
.TXBITSTUFFENH(txbitstuffenh),
.TXENN(txenn),
.TXDAT(txdat),
```

```
 .TXSE0(txse0),
 .FSLSSERIAL(fslsserial),
 .LBKERR(lbkerr),
 .CLKRDY(clkrdy),
 .INTCLK(intclk),
 .ID(id),
 .VBUS(vbus),
 .REXT(rext),
 .XIN(xin),
 .XOUT(xout),
 .CLK480PAD(clk480pad),
 .TEST(test),
 .SCANOUT1(scanout1),
 .SCANOUT2(scanout2),
 .SCANOUT3(scanout3),
 .SCANOUT4(scanout4),
 .SCANOUT5(scanout5),
 .SCANOUT6(scanout6),
 .SCANCLK(scanclk),
 .SCANEN(scanen),
 .SCANMODE(scanmode),
 .TRESETN(tresetn),
 .SCANIN1(scanin1),
 .SCANIN2(scanin2),
 .SCANIN3(scanin3),
 .SCANIN4(scanin4),
 .SCANIN5(scanin5),
 .SCANIN6(scanin6)
);
defparam usb20_phy_inst.DATABUS16_8 = 1'b0;
defparam usb20_phy_inst.ADP_PRBEN = 1'b0;
defparam usb20_phy_inst.TEST_MODE = 5'b0;;
defparam usb20_phy_inst.HSDRV1 = 1'b0;
defparam usb20_phy_inst.HSDRV0 = 1'b0;
defparam usb20_phy_inst.CLK_SEL = 1'b0;
defparam usb20_phy_inst.M = 4'b0;
defparam usb20_phy_inst.N = 6'b101000;
defparam usb20_phy_inst.C = 2'b01;
defparam usb20_phy_inst.FOC_LOCK = 1'b0;

Vhdl Instantiation:
COMPONENT USB20_PHY
GENERIC (
TEST_MODE:bit_vector:="00000";
                        DATABUS16_8:bit:='0';
                        ADP_PRBEN:bit:='0';
                        HSDRV1:bit:='0';
                        HSDRV0:bit:='0';
        CLK_SEL:bit:='0';
        M:bit_vector:="0000";
```

```
        N:bit_vector:=" 101000";
        C:bit_vector:="01";
        FOC_LOCK:bit:='0';
);
    PORT(
      DATAIN:IN std_logic_vector(15 downto 0);
TXVLD:IN std_logic;
TXVLDH:IN std_logic;
RESET:IN std_logic;
SUSPENDM:IN std_logic;
XCVRSEL:IN std_logic_vector(1 downto 0);
TERMSEL:IN std_logic;
OPMODE:IN std_logic_vector(1 downto 0);
DATAOUT:OUT std_logic_vector(15 downto 0);
TXREADY:OUT std_logic;
RXACTIVE:OUT std_logic;
RXVLD:OUT std_logic;
RXVLDH:OUT std_logic;
CLK:OUT std_logic;
RXERROR:OUT std_logic;
DP:INOUT std_logic;
DM:INOUT std_logic;
LINESTATE:OUT std_logic_vector(1 downto 0);
IDPULLUP:IN std_logic;
DPPD:IN std_logic;
DMPD:IN std_logic;
CHARGVBUS:IN std_logic;
DISCHARGVBUS:IN std_logic;
TXBITSTUFFEN:IN std_logic;
TXBITSTUFFENH:IN std_logic;
TXENN:IN std_logic;
TXDAT:IN std_logic;
TXSE0:IN std_logic;
FSLSSERIAL:IN std_logic;
HOSTDIS:OUT std_logic;
IDDIG:OUT std_logic;
ADPPRB:OUT std_logic;
ADPSNS:OUT std_logic;
SESSVLD:OUT std_logic;
VBUSVLD:OUT std_logic;
RXDP:OUT std_logic;
RXDM:OUT std_logic;
RXRCV:OUT std_logic;
LBKERR:OUT std_logic;
CLKRDY:OUT std_logic;
INTCLK:IN std_logic;
ID:INOUT std_logic;
VBUS:INOUT std_logic;
REXT:INOUT std_logic;
XIN:IN std_logic;
```

```
XOUT:INOUT std_logic;
TEST:IN std_logic;
CLK480PAD:OUT std_logic;
SCANCLK:IN std_logic;
SCANEN:IN std_logic;
SCANMODE:IN std_logic;
TRESETN:IN std_logic;
SCANIN1:IN std_logic;
SCANOUT1:OUT std_logic;
SCANIN2:IN std_logic;
SCANOUT2:OUT std_logic;
SCANIN3:IN std_logic;
SCANOUT3:OUT std_logic;
SCANIN4:IN std_logic;
SCANOUT4:OUT std_logic;
SCANIN5:IN std_logic;
SCANOUT5:OUT std_logic;
SCANIN6:IN std_logic;
SCANOUT6:OUT std_logic;
        );
END COMPONENT;
uut:   USB20_PHY
  PORT MAP (
DATAIN=>datain,
TXVLD=>txvld,
TXVLDH=>txvldh,
RESET=>reset,
SUSPENDM=>suspendm,
XCVRSEL=>xcvrsel,
TERMSEL=>termsel,
OPMODE=>opmode,
DATAOUT=>dataout,
TXREADY=>txready,
RXACTIVE=>rxactive,
RXVLD=>rxvld,
RXVLDH=>rxvldh,
CLK=>clk,
RXERROR=>rxerror,
DP=>dp,
DM=>dm,
LINESTATE=>linestate,
0IDPULLUP=>idpullup,
DPPD=>dppd,
DMPD=>dmpd,
CHARGVBUS=>chargvbus,
DISCHARGVBUS=>dischargvbus,
TXBITSTUFFEN=>txbitstuffen,
TXBITSTUFFENH=>txbitstuffenh,
TXENN=>txenn,
TXDAT=>txdat,
```

```
TXSE0=>txse0,
FSLSSERIAL=>fslsserial,
HOSTDIS=>hostdis,
IDDIG=>iddig,
ADPPRB=>adpprb,
ADPSNS=>adpsns,
SESSVLD=>sessvld,
VBUSVLD=>vbusvld,
RXDP=>rxdp,
RXDM=>rxdm,
RXRCV=>rxrcv,
LBKERR=>lbkerr,
CLKRDY=>clkrdy,
INTCLK=>intclk,
ID=>id,
VBUS=>vbus,
REXT=>rext,
XIN=>xin,
XOUT=>xout,
TEST=>test,
CLK480PAD=>clk480pad,
SCANCLK=>scanclk,
SCANEN=>scanen,
SCANMODE=>scanmode,
TRESETN=>tresetn,
SCANIN1=>scanin1,
SCANOUT1=>scanout1,
SCANIN2=>scanin2,
SCANOUT2=>scanout2,
SCANIN3=>scanin3,
SCANOUT3=>scanout3,
SCANIN4=>scanin4,
SCANOUT4=>scanout4,
SCANIN5=>scanin5,
SCANOUT5=>scanout5,
SCANIN6=>scanin6,
SCANOUT6=>scanout6
);
```

8.3 ADC

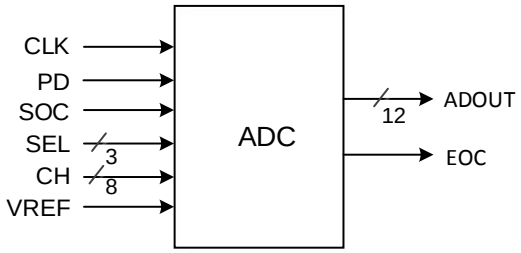
Primitive Introduction

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

Devices supported: GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C.

Block Diagram

Figure 8-3 ADC Block Diagram



Port Description

Table 8-4 Port Description

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

Primitive Instantiation

Verilog Instantiation:

```
ADC adc_inst(
      .CLK(clk),
      .PD(pd),
      .SOC(soc),
      .SEL(sel[2:0]),
      .CH(ch[7:0]),
      .VREF(vref),
      .EOC(eoc),
      .ADOUT(adout[11:0])
);
```

Vhdl Instantiation:

```
COMPONENT ADC
    PORT(
    CLK=>IN std_logic;
            PD=>IN std_logic;
            SOC=>IN std_logic;
            SEL=>IN std_logic_vector(2 downto 0);
            CH=>IN std_logic_vector(7 downto 0);
            VREF=>IN std_logic;
            EOC=>OUT std_logic;
            ADOUT=>OUT std_logic_vector(11 downto 0)
```

```
            );
        END COMPONENT;
        uut=> ADC
          PORT MAP (
                CLK=>clk,
                PD=>pd,
                SOC=>soc,
                SEL=>sel,
                CH=>ch,
                VREF=>vref,
EOC=>eoc,
ADOUT=>adout
            );
```

9 SPMI and I3C

9.1 SPMI

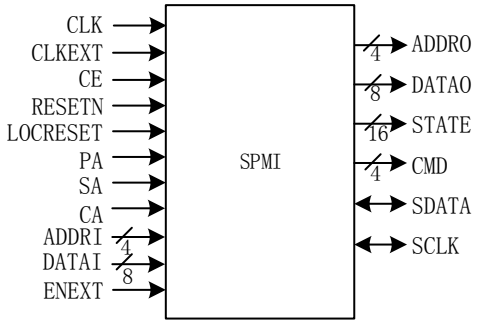
Primitive Introduction

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

Devices supported: GW1NZ-1 device.

Block Diagram

Figure 9-1 SPMI Block Diagram



Port Description

Table 9-1 Port Description

Port Name	I/O	Description
CLK	input	Clock input
CLKEXT	input	External clock input
CE	input	Clock Enable
RESETN	input	Reset input
ENEXT	input	Enext input
LOCRESET	input	Local reset input
PA	input	Priority arbitration input
SA	input	Secondary arbitration input
CA	input	Connection arbitration input
ADDRI	input	Addr input
DATAI	input	Data input

Port Name	I/O	Description
ADDRO	output	Addr output
DATAO	output	datat output
STATE	output	state output
CMD	output	command output
SDATA	inout	SPMI Serial data
SCLK	inout	SPMI Serial Clock

Primitive Instantiation

Verilog Instantiation:

```
SPMI uut (
        .ADDRO(addro),
        .DATAO(datao),
        .STATE(state),
        .CMD(cmd),
        .SDATA(sdata),
        .SCLK(sclk),
        .CLK(clk),
        .CE(ce),
        .RESETN(resetn),
        .LOCRESET(locreset),
        .PA(pa),
        .SA(sa),
        .CA(ca),
        .ADDRI(addri),
        .DATAI(datai),
        .CLKEXT(clkext),
        .ENEXT(enext)
);
```

Vhdl Instantiation:

```
COMPONENT SPMI
    PORT(
    CLK:IN std_logic;
                CLKEXT:IN std_logic;
                CE:IN std_logic;
                RESETN:IN std_logic;
                ENEXT:IN std_logic;
                LOCRESET:IN std_logic;
                PA:IN std_logic;
                SA:IN std_logic;
                CA:IN std_logic;
                ADDRI:IN std_logic_vector(3 downto 0);
                DATAI:IN std_logic_vector(7 downto 0);
                ADDRO:OUT std_logic_vector(3 downto 0);
                DATAO:OUT std_logic_vector(7 downto 0);
                STATE:OUT std_logic_vector(15 downto 0);
                CMD:OUT std_logic_vector(3 downto 0);
                SDATA:INOUT std_logic;
```

```
                SCLK:INOUT std_logic
    );
END COMPONENT;
uut:   SPMI
  PORT MAP (
   CLK=>clk,
                CLKEXT=>clkext,
                CE=>ce,
                RESETN=>resetn,
                ENEXT=>enext,
                LOCRESET=>locreset,
                PA=>pa,
                SA=>sa,
                CA=>ca,
                ADDRI=>addri,
                DATAI=>datai,
                ADDRO=>addro,
                DATAO=>datao,
                STATE=>state,
                CMD=>cmd,
                SDATA=>sdata,
                SCLK=>sclk
   );
```

9.2 I3C

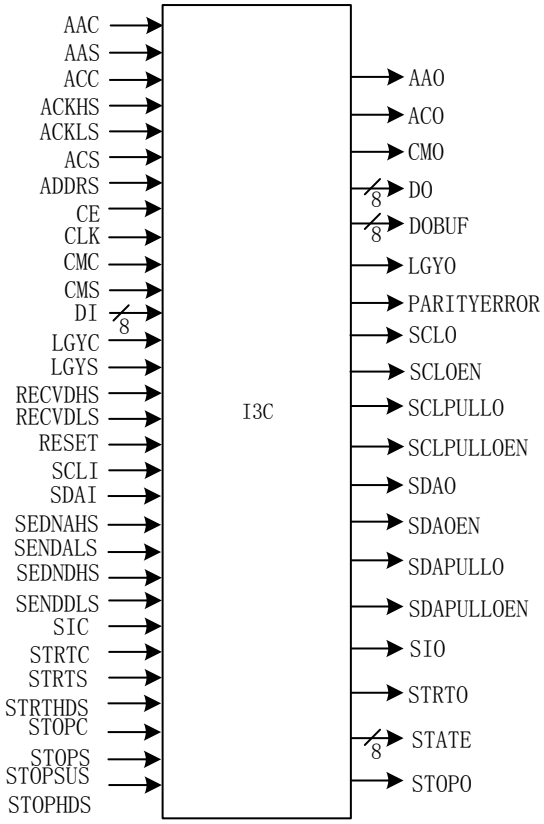
Primitive Introduction

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

Devices supported: GW1NZ-1.

Block Diagram

Figure 9-2 I3C Block Diagram



Port Description

Table 9-2 Port Description

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

Port Name	I/O	Description
STRTHDS	input	Adjust the setting signal when generating START
SENDAHS	input	Adjust the setting signal of SCL at a high level when the address is sent.
SENDALS	input	Adjust the setting signal of SCL at a low level when the address is sent
ACKHS	input	Adjust the setting signal of SCL at a high level in ACK.
SENDDLS	input	Adjust the setting signal of SCL at a low level in ACK.
RECVDHS	input	Adjust the setting signal of SCL at a high level when the data are received
RECVDLS	input	Adjust the setting signal of SCL at a low level when the data are received
ADDRS	input	The slave address setting interface
DI	input	Data Input.
SDAI	input	I3C serial data input
SCLI	input	I3C serial clock input
LGYO	output	Output the current communication object as the I2C command.
CMO	output	Output the command of the device is in the Master mode.
ACO	output	Continue to output when selecting whether to continue
AAO	output	Reply ACK when you need to reply ACK/NACK
SIO	output	Interrupt to output the identity bit
STOPO	output	Output the STOP command
STRTO	output	Output the START command
PARITYERROR	output	Output check when receiving data
DOBUF	output	Data output after caching
DO	output	Data output directly
STATE	output	Output the internal state
SDAO	output	I3C serial data output
SCLO	output	I3C serial clock output
SDAOEN	output	I3C serial data oen output
SCLOEN	output	I3C serial clock oen output
SDAPULLO	output	Controllable pull-up of the I3C serial data
SCLPULLO	output	Controllable pull-up of the I3C serial clock
SDAPULLOEN	output	Controllable pull-up of the I3C serial data oen
SCLPULLOEN	output	Controllable pull-up of the I3C serial clock oen

Primitive Instantiation

Verilog Instantiation:

```
I3C i3c_inst (
    .LGYO(lgyo),
    .CMO(cmo),
    .ACO(aco),
    .AAO(aao),
    .SIO(sio),
    .STOPO(stopo),
    .STRTO(strto),
    .PARITYERROR(parityerror),
```

```
        .DOBUF(dobuf),
        .DO(dout),
        .STATE(state),
        .SDAO(sdao),
        .SCLO(sclo),
        .SDAOEN(sdaoen),
        .SCLOEN(scloen),
        .SDAPULLO(sdapullo),
        .SCLPULLO(sclpullo),
        .SDAPULLOEN(sdapulloen),
        .SCLPULLOEN(sclpulloen),
        .LGYS(lgys),
        .CMS(cms),
        .ACS(acs),
        .AAS(aas),
        .STOPS(stops),
        .STRTS(strts),
        .LGYC(lgyc),
        .CMC(cmc),
        .ACC(acc),
        .AAC(aac),
        .SIC(sic),
        .STOPC(stopc),
        .STRTC(strtc),
        .STRTHDS(strthds),
        .SENDAHS(sendahs),
        .SENDALS(sendals),
        .ACKHS(ackhs),
        .ACKLS(ackls),
        .STOPSUS(stopsus),
        .STOPHDS(stophds),
        .SENDDHS(senddhs),
        .SENDDLS(senddls),
        .RECVDHS(recvdhs),
        .RECVDLS(recvdls),
        .ADDRS(addrs),
        .DI(di),
        .SDAI(sdai),
        .SCLI(scli),
        .CE(ce),
        .RESET(reset),
        .CLK(clk)
    );
```

10 Miscellaneous

10.1 GSR

Primitive Name

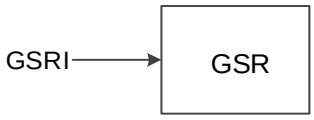
Global Reset/Set (GSR)

Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

Figure 10-1 GSR Port Diagram



Functional Description

GSR module can implement global reset/set function, active-low. The default is high level. External signals can be connected to be pulled down to realize the reset of register module and other modules.

Primitive Definition

Port Description

Table 10-1 Port Description

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

Primitive Instantiation

Verilog instantiation:

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

```
);
```

Vhdl instantiation:

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

10.2 INV

Primitive Name

Inverter (INV)

Devices Supported

Devices supported: GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1NSE-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NSR-4, GW1NSR-4C, GW1NSER-4C, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, GW2A-55.

Port Diagram

Figure 10-2 INV Port Diagram

Functional Description

INV module implements the inverter function.

Primitive Definition

Port Description

Table 10-2 Port Description

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

Primitive Instantiation

Verilog instantiation:

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

Vhdl instantiation:

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

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


GOWIN
PROGRAMMING FOR THE FUTURE