Gowin FPGA Primitive

User Guide

SUG283-1.8E, 05/20/2019

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, and IEM added; ● DSP primitive updated; ● Port names of ODDR/ODDRC, IDDR_MEM, IDES4_MEM, IDES8_MEM, RAM16S1, RAM16S2, RAM16S4, RAM16SDP1, RAM16SDP2, RAM16SDP4, and ROM16 updated; ● Attributes of OSC, PLL, and DLLDLY updated; ● Primitive instantiation updated; ● MIPI_IBUF_HS, MIPI_IBUF_LP, MIPI_OBUF, IDES16, and OSER16 added; ● Attribute of CLKDIV updated.
04/12/2018	1.2E	Primitive instantiation vhdl added.
09/13/2018	1.3E	● GW1N-2B, GW1N-4B, GW1NR-4B, GW1N-6ES, GW1N-9ES, GW1NR-9ES, GW1NS-2, GW1NS-2C devices added; ● I3C_IOBUF and DHCEN added; ● User Flash added; ● EMPU added; ● Primitive names description updated.
10/26/2018	1.4E	● GW1NZ-1 and GW1NSR-2C devices added; ● OSCZ and FLASH96KZ added.
11/15/2018	1.5E	GW1NSR-2 device added, GW1N-6ES, GW1N-9ES and GW1NR-9ES devices removed.
02/13/2019	1.6E	● CLKDIV: GW1NS-2 supports 8 frequency division; ● TLVDS_TBUF/OBUF does not support GW1N-1.
03/05/2019	1.7E	TLVDS_IOBUF does not support GW1N-1.
05/20/2019	1.8E	● GW1N-1S added; ● MIPI_IBUF added; ● OSCH added; ● SPMI added; ● I3C added; ● Devices supported by OSC updated.

Contents

List of Figures

List of Tables

1 IOB

1.1 Buffer/LVDS

Buffer includes normal buffer, ELVDS, and TLVDS.

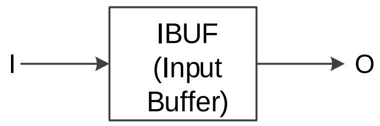
1.1.1 IBUF

Primitive Introduction

The input buffer (IBUF) supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-1 IBUF View



Port Description

Table 1-1 IBUF 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
         );
```

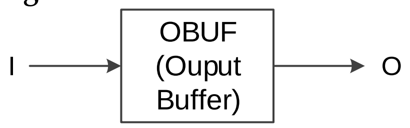
1.1.2 OBUF

Primitive Introduction

The output buffer (OBUF) supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-2 OBUF View



Port Description

Table1-2 OBUF 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
           );
```

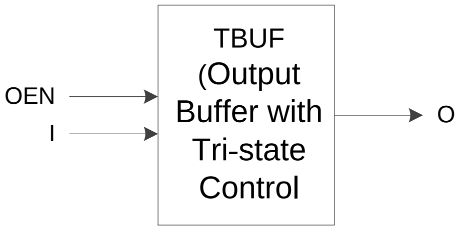
1.1.3 TBUF

Primitive Introduction

The Output Buffer with Tri-state Control (TBUF), low level enable, supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-3 TBUF View



Port Description

Table1-3 TBUF 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

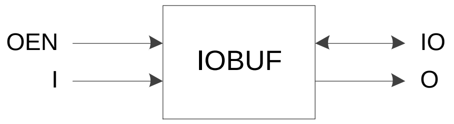
Primitive Introduction

When OEN is high, the Bi-Directional Buffer (IOBUF) is used as an input buffer; when OEN is low, the IOBUF is used as an output buffer.

The IOBUF supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-4 IOBUF View



Port Description

Table1-4 IOBUF 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

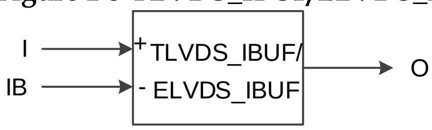
The LVDS differential input includes True LVDS Input Buffer (TLVDS_IBUF) and Emulated LVDS Input Buffer (ELVDS_IBUF).

The TLVDS_IBUF input buffer supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

The ELVDS_IBUF input buffer supports the GW1N-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW1N-1S, GW2A-18, GW2AR-18, and GW2A-55.

Architecture Overview

Figure 1-5 TLVDS_IBUF/ELVDS_IBUF View



Port Description

Table1-5 TLVDS_IBUF/ELVDS_IBUF 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

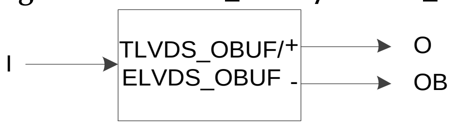
The LVDS differential output includes True LVDS Output Buffer (TLVDS_OBUF) and Emulated LVDS Output Buffer (ELVDS_OBUF).

The TLVDS_OBUF supports the GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

The ELVDS_OBUF supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-6 TLVDS_OBUF/ELVDS_OBUF View



Port Description

Table1-6 TLVDS_OBUF/ELVDS_OBUF 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

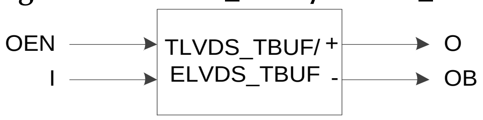
The LVDS tristate buffer includes True LVDS Tristate Buffer (TLVDS_TBUF) and Emulated LVDS Tristate Buffer (ELVDS_TBUF).

The TLVDS_TBUF, low level enable, supports the GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

The ELVDS_TBUF, low level enable, supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-7 TLVDS_TBUF/ELVDS_TBUF View



Port Description

Table1-7 TLVDS_TBUF/ELVDS_TBUF 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 True LVDS Bi-Directional Buffer (TLVDS_OBUF) and Emulated LVDS Bi-Directional Buffer (ELVDS_IBUF).

When OEN is high, the TLVDS_IOBUF is used as an true differential input buffer; when OEN is low, the IOBUF is used as an true differential output buffer.

The TLVDS_IOBUF inout buffer supports the GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55.

When OEN is high, the ELVDS_IOBUF is used as an emulated differential input buffer; when OEN is low, the OEN is used as an emulated differential output buffer.

The ELVDS_IOBUF inout buffer supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-8 TLVDS_IOBUF/ELVDS_IOBUF View

Port Description

Table1-8 TLVDS_IOBUF/ELVDS_IOBUF 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
        );
```

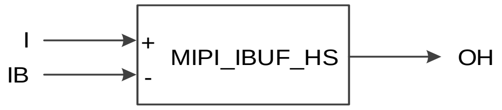
1.1.9 MIPI_IBUF_HS

Primitive Introduction

The MIPI High Speed Input Buffer (MIPI_IBUF_HS) supports GW1NS-2, GW1NS-2C, GW1N-6, GW1N-9, GW1NR-9, GW1NSR-2, and GW1NSR-2C devices.

Architecture Overview

Figure 1-9 MIPI_IBUF_HS View



Port Description

Table1-9 MIPI_IBUF_HS 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
        );
```

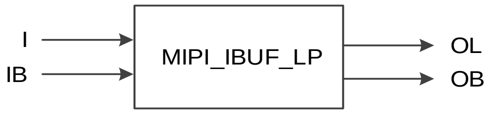
1.1.10 MIPI_IBUF_LP

Primitive Introduction

The MIPI Low Power Input Buffer (MIPI_IBUF_LP) supports GW1NS-2, GW1NS-2C, GW1N-6, GW1N-9, GW1NR-9, GW1NSR-2 and GW1NSR-2C devices.

Architecture Overview

Figure 1-10 MIPI_IBUF_LP View



Port Description

Table1-10 MIPI_IBUF_LP 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

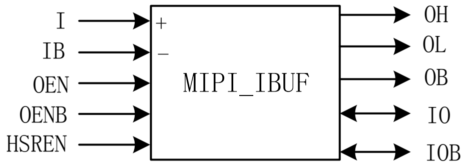
Primitive Introduction

MIPI_IBUF(MIPI Input Buffer) supports two working modes: HS input mode and LP bidirectional mode, among which HS mode supports dynamic resistance configuration.

MIPI_IBUF(MIPI Input Buffer) supports GW1N-1S, GW1NS-2, GW1NS-2C, GW1N-6, GW1N-9, GW1NR-9, GW1NR-6, GW1NSR-2 and GW1NSR-2C.

Architecture Overview

Figure 1-11 MIPI_IBUF View



Port Description

Table 1-11 MIPI_IBUF 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

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

When MODESEL is high, the MIPI_OBUF is used as a mobile industry processor high speed output buffer; when MODESEL is low, the MIPI_OBUF is used as a mobile industry processor low power output buffer.

The MIPI_OBUF supports GW1NS-2, GW1NS-2C, GW1N-6, GW1N-9, GW1NR-9, GW1NSR-2, and GW1NSR-2C devices.

Architecture Overview

Figure 1-12 MIPI_OBUF View

Port Description

Table1-12 MIPI_OBUF 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

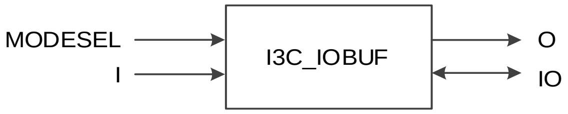
The I3C Bi-Directional Buffer (I3C_IOBUF) includes Normal mode and I3C mode.

When MODESEL is high, the I3C_IOBUF is used as an I3C bi-directional buffer; when MODESEL is low, the I3C_IOBUF is used as a normal bi-directional buffer.

The I3C_IOBUF supports GW1NS-2, GW1NS-2C, GW1N-6, GW1N-9, GW1NR-9, GW1NSR-2 and GW1NSR-2C devices.

Architecture Overview

Figure 1-13 I3C_IOBUF View



Port Description

Table1-13 I3C_IOBUF 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 Introduction

The Dual Data Rate Input (IDDR) supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1,

GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-14 IDDR View

Port Description

Table1-14 IDDR Port Description

Port Name	I/O	Description
D	Input	Data input, from port O of input buffer or port DO of IODELAY
CLK	Input	Clock input
Q0	Output	Data output
Q1	Output	Data output

Q0_INIT and Q1_INIT of IDDR means Register initialization of IDDR.

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 Introduction

The Dual Data Rate Output (ODDR) supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-15 ODDR View

Port Description

Table1-15 ODDR Port Description

Port Name	I/O	Description
D0	Input	Data input
D1	Input	Data input
TX	Input	Data input
CLK	Input	Clock input
Q0	Output	Data Output, to port I of output buffer or port DI of IODELAY
Q1	Output	Tristate enable output, to port OEN of tristate/inout buffer(Q0 connected)or dangling

INIT of ODDR means Register initialization of ODDR.

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

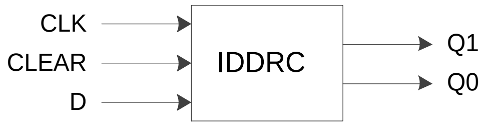
Primitive Introduction

Similar to IDDR, the Dual Data Rate Input with Asynchronous Clear (IDDRC) inputs double data rate and can asynchronously reset.

The IDDRC supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-16 IDDRC View



Port Description

Table1-16 IDDRC Port Description

Port Name	I/O	Description
D	Input	Data input (from port O of input buffer or port DO of IODELAY)
CLK	Input	Clock input
CLEAR	Input	Asynchronous Clear Input
Q0	Output	Data output
Q1	Output	Data output

Q0_INIT and Q1_INIT of IDDR denotes Register initialization of IDDRC.

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

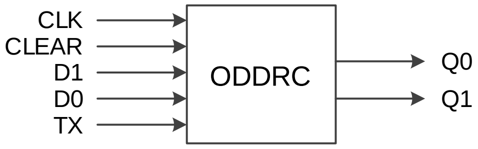
Primitive Introduction

Similar to ODDR, the Dual Data Rate Output with Asynchronous Clear (ODDRC) outputs double data rate and can asynchronously reset.

The ODDRC supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-17 ODDRC View



Port Description

Table1-17 ODDRC Port Description

Port Name	I/O	Description
D0	Input	Data input
D1	Input	Data input
TX	Input	Data input
CLK	Input	Clock input
CLEAR	Input	Asynchronous Clear Input
Q0	Output	Data Output, to port I of output buffer or port DI of IODELAY
Q1	Output	Tristate enable output (To port OEN of tristate/inout buffer [Q0 connected]or dangling)

INIT of ODDRC denotes register initialization of ODDRC.

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

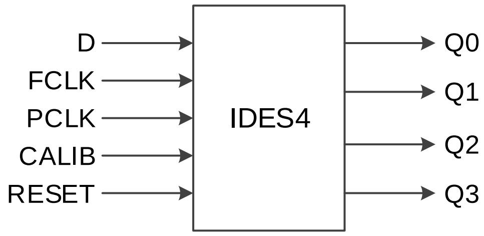
Primitive Introduction

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

It supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-18 IDES4 View



Port Description

Table1-18 IDES4 Port Description

Port Name	I/O	Description
D	Input	Data input (from port O of input buffer or port DO of IODELAY)
FCLK	Input	Fast clock input
PCLK	Input	Primary clock input
CALIB:	Input	Calib signal, adjust output
RESET	Input	Asynchronous reset input
Q3~Q0	Output	Data output

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

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

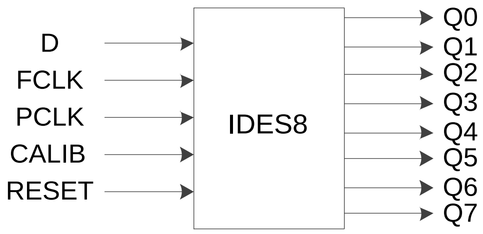
Primitive Introduction

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

It supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-19 IDES8 View



Port Description

Table1-19 IDES8 Port Description

Port Name	I/O	Description
D	Input	Data input (from port O of input buffer or port DO of IODELAY)
FCLK	Input	Fast clock input
PCLK	Input	Primary clock input
CALIB:	Input	Calib Signal Input
RESET	Input	Asynchronous reset input
Q7~Q0	Output	Data output

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

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

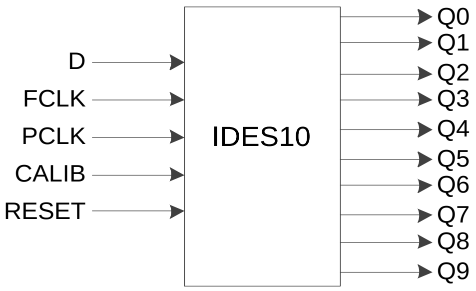
Primitive Introduction

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

It supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-20 IDES10 View



Port Description

Table1-20 IDES10 Port Description

Port Name	I/O	Description
D	Input	Data input (from port O of input buffer or port DO of IODELAY)
FCLK	Input	Fast clock input
PCLK	Input	Primary clock input
CALIB:	Input	Calib signal
RESET	Input	Asynchronous reset input
Q9~Q0	Output	Data output

PCLK is usually obtained by FCLK frequency division, $f_{PCLK} = 1/5\, f_{FCLK}$.

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

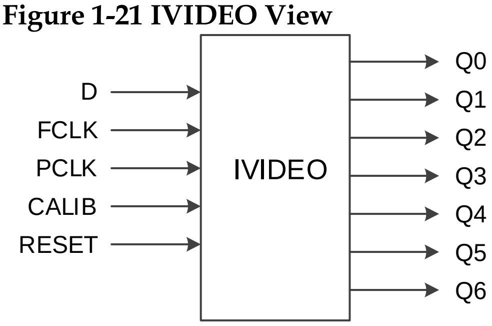
Primitive Introduction

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

It supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-21 IVIDEO View



Port Description

Table1-21 IVIDEO Port Description

Port Name	I/O	Description
D	Input	Data input (from port O of input buffer or port DO of IODELAY)
FCLK	Input	Fast clock input
PCLK	Input	Primary clock input
CALIB:	Input	Calib Signal Input
RESET	Input	Asynchronous reset input
Q6~Q0	Output	Data output

PCLK is usually obtained by FCLK frequency division, $f_{PCLK} = 1/3.5\, f_{FCLK}$.

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

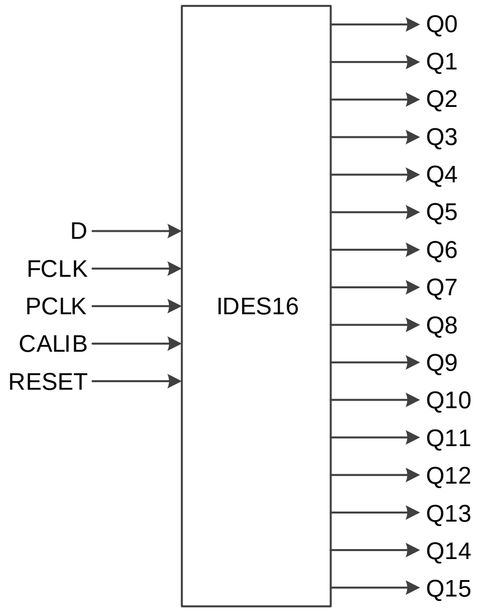
Primitive Introduction

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

The IDES16 supports GW1N-1S, GW1NS-2, GW1NS-2C, GW1N-6, GW1N-9, GW1NR-9, GW1NSR-2, and GW1NSR-2C.

Architecture Overview

Figure 1-22 IDES16 View



Port Description

Table1-22 IDES16 Port Description

Port Name	I/O	Description
D	Input	Data input (from port O of input buffer or port DO of IODELAY)
FCLK	Input	Fast clock input
PCLK	Input	Primary clock input
CALIB:	Input	Calib signal
RESET	Input	Asynchronous reset input
Q15~Q0	Output	Data output

PCLK is usually obtained by FCLK frequency division, $f_{PCLK} = 1/8\, f_{FCLK}$.

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

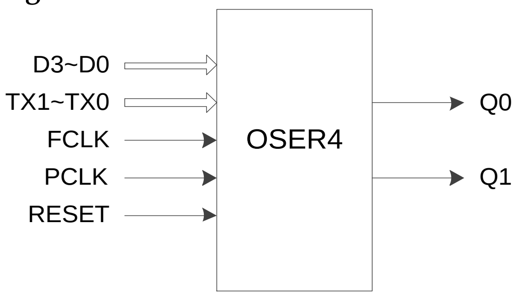
Primitive Introduction

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

It supports the GW1N-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-23 OSER4 View



Port Description

Table1-23 OSER4 Port Description

Port Name	I/O	Description
D3~D0	Input	Data input
TX1~TX0	Input	Tristate Data input
FCLK	Input	Fast clock input
PCLK	Input	Primary clock input
RESET	Input	Asynchronous reset input
Q0	Output	Data Output, to port I of output buffer or port DI of IODELAY
Q1	Output	Tristate enable output (To port OEN of tristate/inout buffer [Q0 connected]or dangling)

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

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

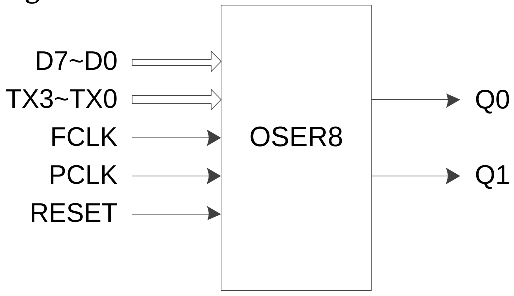
Primitive Introduction

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

It supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-24 OSER8 View



Port Description

Table1-24 OSER8 Port Description

Port Name	I/O	Description
D7~D0	Input	Data input
TX3~TX0	Input	Data input
FCLK	Input	Fast clock input
PCLK	Input	Primary clock input
RESET	Input	Asynchronous reset input
Q0	Output	Data Output, to port I of output buffer or port DI of IODELAY
Q1	Output	Tristate enable output (To port OEN of tristate/inout buffer [Q0 connected]or dangling)

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

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

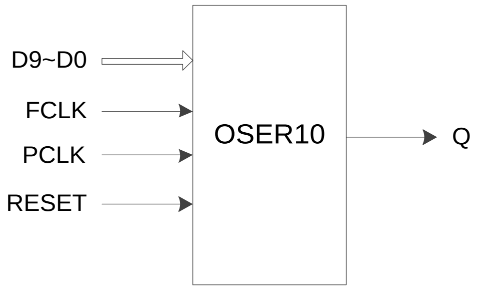
Primitive Introduction

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

It supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-25 OSER10 View



Port Description

Table1-25 OSER10 Port Description

Port Name	I/O	Description
D9~D0	Input	Data input
FCLK	Input	Fast clock input
PCLK	Input	Primary clock input
RESET	Input	Asynchronous reset input
Q	Output	Data Output, to port I of output buffer or port DI of IODELAY

PCLK is usually obtained by FCLK frequency division, $f_{PCLK} = 1/5\, f_{FCLK}$.

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

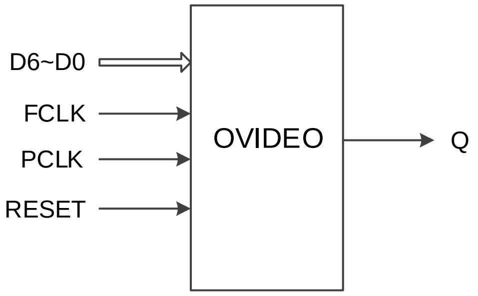
Primitive Introduction

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

It supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-26 OVIDEO View



Port Description

Table1-26 OVIDEO Port Description

Port Name	I/O	Description
D6~D0	Input	Data input
FCLK	Input	Fast clock input
PCLK	Input	Primary clock input
RESET	Input	Asynchronous reset input
Q	Output	Data Output, to port I of output buffer or port DI of IODELAY

PCLK is usually obtained by FCLK frequency division, $f_{PCLK} = 1/3.5\, f_{FCLK}$.

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

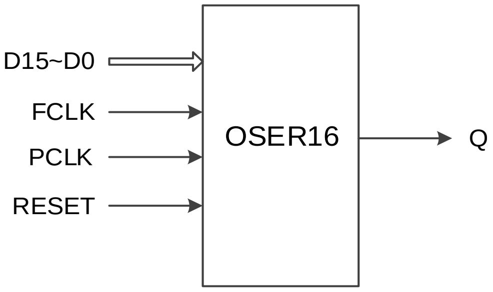
Primitive Introduction

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

The IDES16 supports GW1N-1S, GW1NS-2, GW1NS-2C, GW1N-6, GW1N-9, GW1NR-9, GW1NSR-2, and GW1NSR-2C.

Architecture Overview

Figure 1-27 OSER16 View



Port Description

Table1-27 OSER16 Port Description

Port Name	I/O	Description
D15~D0	Input	Data input
FCLK	Input	Fast clock input
PCLK	Input	Primary clock input
RESET	Input	Asynchronous reset input
Q	Output	Data Output, to port I of output buffer or port DI of IODELAY

PCLK is usually obtained by FCLK frequency division, $f_{PCLK} = 1/8\, f_{FCLK}$.

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

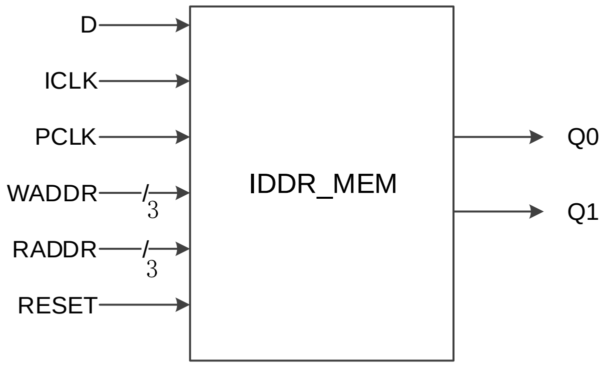
Primitive Introduction

The Dual Data Rate Input with Memory (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.

IDDR_MEM supports GW2A-18, GW2AR-18 and GW2A-55.

Architecture Overview

Figure 1-28 IDDR_MEM View



Port Description

Table1-28 IDDR_MEM Port Description

Port Name	I/O	Description
D	Input	Data input (from port O of input buffer or port DO of IODELAY)
ICLK	Input	Clock Input , from DQSR90 port of DQS
PCLK	Input	Primary clock input
WADDR[2:0]	Input	Write Address, from port WPOINT of DQS
RADDR[2:0]	Input	Read Address, from port RPOINT of DQS
RESET	Input	Asynchronous reset input
Q1~Q0	Output	Data output

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.

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

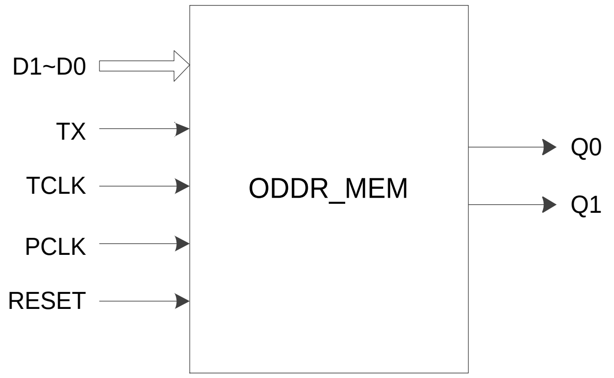
Primitive Introduction

Unlike ODDR, the Dual Data Rate Output 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.

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

Architecture Overview

Figure 1-29 ODDR_MEM View



Port Description

Table1-29 ODDR_MEM Port Description

Port Name	I/O	Description
D1~D0	Input	Data input
TX	Input	Data input
TCLK	Input	Clock Input, from port DQSW0 or DQSW270 of DQS
PCLK	Input	Primary clock input
RESET	Input	Asynchronous reset input
Q0	Output	Data Output, to port I of output buffer or port DI of IODELAY
Q1	Output	Tristate enable output (To port OEN of tristate/inout buffer [Q0 connected]or dangling)

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.

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 Introduction

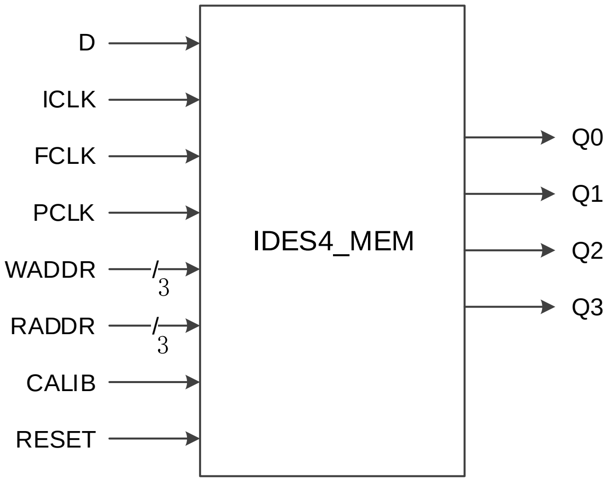
Unlike IDES4, the 4 to 1 Deserializer with Memory (IDES4_MEM) needs to be used with the DQS. The ICLK connects the DQSR90 of DQS output signal and sends data to IDES4_MEM according to the ICLK clock edge. WADDR [2: 0] connects the output signal WPOINT of DQS; RADDR

[2: 0] connects the output signal RPOINT of DQS.

The IDES4_MEM supports the GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-30 IDES4_MEM View



Port Description

Table1-30 IDES4_MEM Port Description

Port Name	I/O	Description
D	Input	Data input (from port O of input buffer or port DO of IODELAY)
ICLK	Input	Clock Input , from DQSR90 port of DQS
FCLK	Input	Fast clock input
PCLK	Input	Primary clock input
WADDR[2:0]	Input	Write Address, from port WPOINT of DQS
RADDR[2:0]	Input	Read Address, from port RPOINT of DQS
CALIB:	Input	Calib Signal Input
RESET	Input	Asynchronous reset input
Q3~Q0	Output	Data output

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.

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

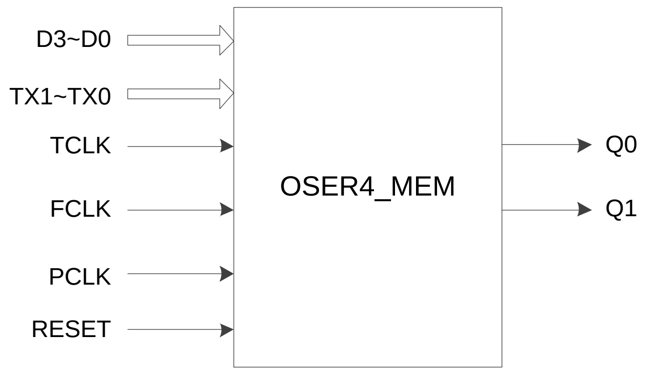
Primitive Introduction

Unlike OSER4, the 4 to 1 Serializer with Memory (OSER4_MEM) needs to be used with the DQS. The TCLK connects the output signal DQSW0 or DQSW270 of DQS, and outputs data from the OSER4_MEM according to the TCLK clock edge.

The OSER4_MEM supports the GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-31 OSER4_MEM View



Port Description

Table1-31 OSER4_MEM Port Description

Port Name	I/O	Description
D3~D0	Input	Data input
TX1~TX0	Input	Data input
TCLK	Input	Clock Input, from port DQSW0 or DQSW270 of DQS
FCLK	Input	Fast clock input
PCLK	Input	Primary clock input
RESET	Input	Asynchronous reset input
Q0	Output	Data Output, to port I of output buffer or port DI of IODELAY
Q1	Output	Tristate enable output (To port OEN of tristate/inout buffer [Q0 connected]or dangling)

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.

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

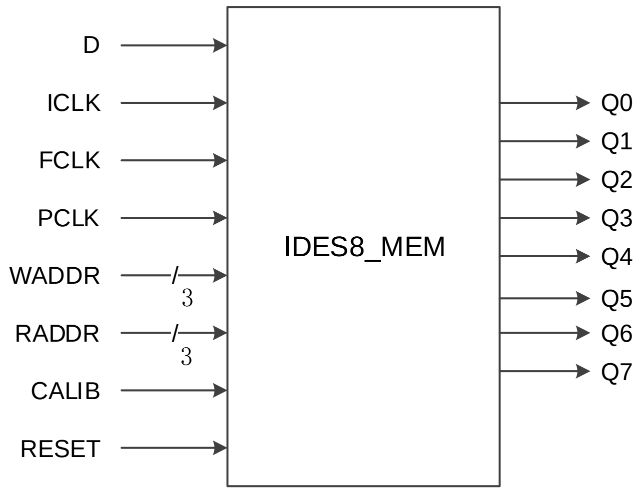
Primitive Introduction

Unlike IDES8, the 8 to 1 Deserializer with Memory (IDES8_MEM) needs to be used with DQS. 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 IDES8_MEM supports the GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-32 IDES8_MEM View



Port Description

Table1-32 IDES8_MEM Port Description

Port Name	I/O	Description
D	Input	Data input (from port O of input buffer or port DO of IODELAY)
ICLK	Input	Clock Input , from DQSR90 port of DQS
FCLK	Input	Fast clock input
PCLK	Input	Primary clock input
WADDR[2:0]	Input	Write Address, from port WPOINT of DQS
RADDR[2:0]	Input	Read Address, from port RPOINT of DQS
CALIB:	Input	Calib Signal Input
RESET	Input	Asynchronous reset input
Q7~Q0	Output	Data output

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.

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

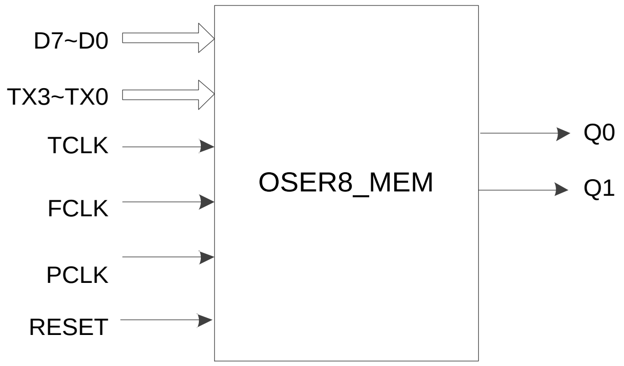
Primitive Introduction

Unlike OSER8, the 8 to 1 Serializer with Memory (OSER8_MEM) needs to be used with DQS. The TCLK connects the output signal DQSW0 or DQSW270 of DQS, and outputs data from the OSER8_MEM according to the TCLK clock edge.

The OSER8_MEM supports the GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-33 OSER8_MEM View



Port Description

Table1-33 OSER8_MEM Port Description

Port Name	I/O	Description
D7~D0	Input	Data input
TX3~TX0	Input	Data input
TCLK	Input	Clock Input, from port DQSW0 or DQSW270 of DQS
FCLK	Input	Fast clock input
PCLK	Input	Primary clock input
RESET	Input	Asynchronous reset input
Q0	Output	Data output
Q1	Output	Data output

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.

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

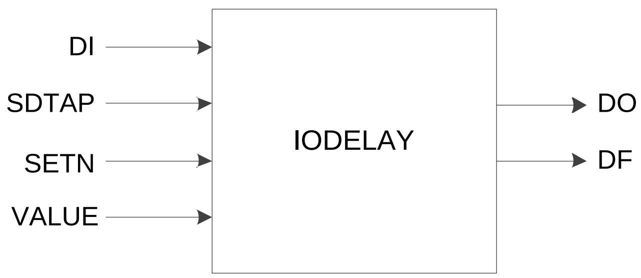
Primitive Introduction

The Input/Output delay (IODELAY) is a programmable absolute delay unit included in IO.

It supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-34 IODELAY View



Port Description

Table1-34 IODELAY Port Description

Port Name	I/O	Description
DI	Input	Data Input
SDTAP	Input	Control Delay Code's Download
SETN	Input	Direction can be Selected to Decide Delay, Increase or Decrease
VALUE	Input	Adjust Delay Value
DO	Output	Data Output
DF	Output	Margin test output flag for DELAY to indicate the under-flow or over-flow

Attribute Description

Table1-35 IODELAY Attribute Description

Attribute Name	Permitted Values	Default	Description
C_STATIC_DLY	0~127	0	Delay Control

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

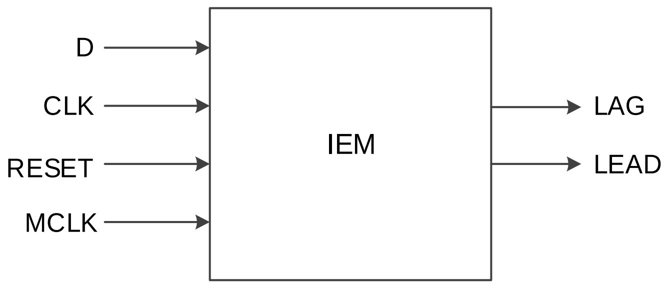
Primitive Introduction

The Input Edge Monitor (IEM) is a sampling module included in IO, which is used for sampling data edge and is available for DDR mode.

It supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 1-35 IEM Architecture Overview



Port Description

Table1-36 IEM Port Description

Port Name	I/O	Description
D	Input	Data input
CLK	Input	Control Delay Code's Download
RESET	Input	Direction can be Selected to Decide Delay, Increase or Decrease
MCLK	Input	Adjust Delay Value
LAG	Output	Data output
LEAD	Output	Margin test output flag for DELAY to indicate the under-flow or over-flow

Attribute Description

Table1-37 IEM Attribute Description

Attribute Name	Permitted Values	Default	Description
WINSIZE	SMALL,MIDSMALL, MIDLARGE, LARGE	SMALL	Delay Control
GSREN	false, true	false	Global reset
LSREN	false, true	true	Enable 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:

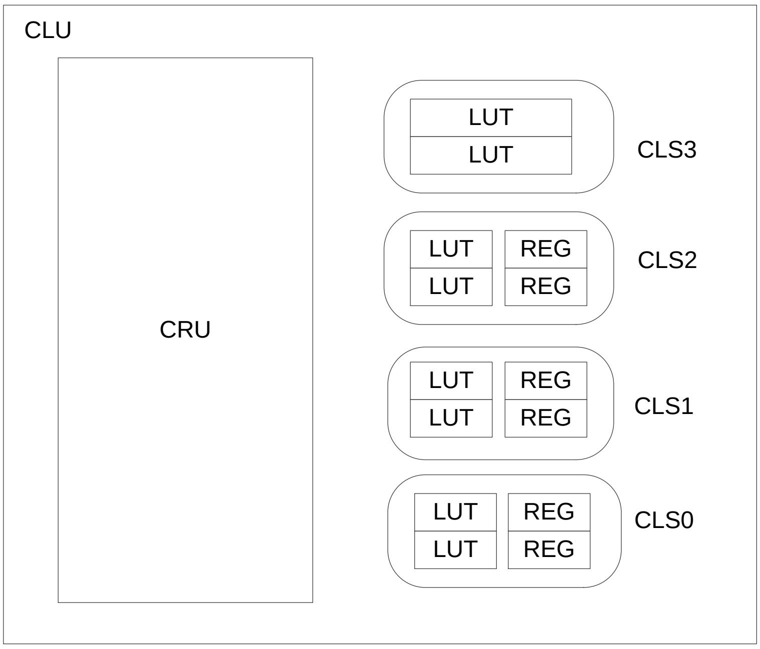
```
COMPONENT IEM
          GENERIC (WINSIZE:string:="SMALL";
                         GSREN:string:="false";
                         LSREN:string:="true"
         );
          PORT(
                 LAG:OUT std_logic;
                 LEAD:OUT std_logic;
                 D:IN std_logic;
                 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

The Configurable Logic Unit (CLU) is the base cell for FPGA. In each CLU, there are four Configurable Logic Slices (CLS) and one Configurable Routing Unit (CRU). Figure 2-1 shows the CLU structure. The CLS can be used to configure the LUT/LUT2 input arithmetical logic units (ALU) and the registers (REG). The CLU can achieve the function of MUX/LUT/ALU/FF/LATCH, etc.

Figure 2-1 CLU Structure



2.1 LUT

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

It supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

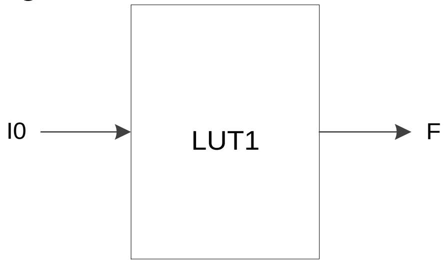
2.1.1 LUT1

Primitive Introduction

The 1-input Look-up Table (LUT1) is a simple type usually used for a buffer and an inverter. LUT1 is an one input lookup table. After initializing INIT using the appropriate parameter, you can look up the corresponding data and output the result according to the input address.

Architecture Overview

Figure 2-2 LUT1 View



Port Description

Table2-1 LUT1 Port Description

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

Attribute Description

Table2-2 LUT1 Attribute Description

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

Truth Table

Table2-3 Truth Table

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

```
COMPONENT LUT1
```

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

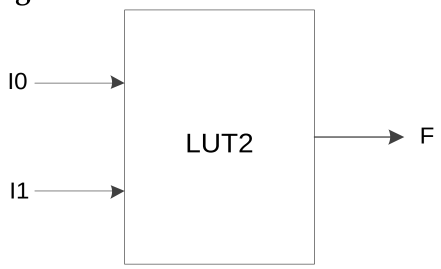
2.1.2 LUT2

Primitive Introduction

The 2-input Look-up Table (LUT2) is a two-input lookup table. After initializing INIT using the appropriate parameter, you can look up the corresponding data and output the result according to the input address.

Architecture Overview

Figure 2-3 LUT2 View



Port Description

Table2-4 LUT2 Port Description

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

Attribute Description

Table2-5 LUT2 Attribute Description

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

Truth Table

Table2-6 Truth Table

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

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

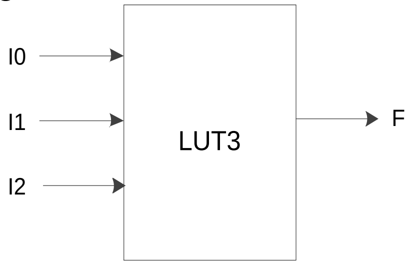
2.1.3 LUT3

Primitive Introduction

The 3-input Look-up Table (LUT3) is a three-input lookup table. After initializing INIT using the appropriate parameter, you can look up the corresponding data and output the result according to the input address.

Architecture Overview

Figure 2-4 LUT3 View



Port Description

Table2-7 LUT3 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

Table2-8 LUT3 Attribute Description

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

Truth Table

Table2-9 Truth Table

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

Primitive Instantiation

Verilog Instantiation:

```
LUT3 instName (
        .I0(I0),
        .I1(I1),
        .I2(I2),
```

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

Vhdl Instantiation:

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

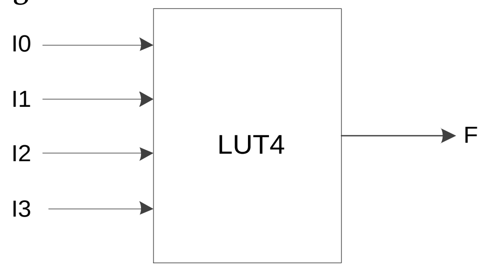
2.1.4 LUT4

Primitive Introduction

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

Architecture Overview

Figure 2-5 LUT4 View



Port Description

Table2-10 LUT4 Port Description

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

Attribute Description

Table2-11 LUT4 Attribute Description

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

Truth Table

Table2-12 Truth Table

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

Primitive Instantiation

Verilog Instantiation:

```
LUT4 instName (
        .I0(I0),
        .I1(I1),
        .I2(I2),
```

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

Vhdl Instantiation:

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

2.1.5 Wide LUT

Primitive Introduction

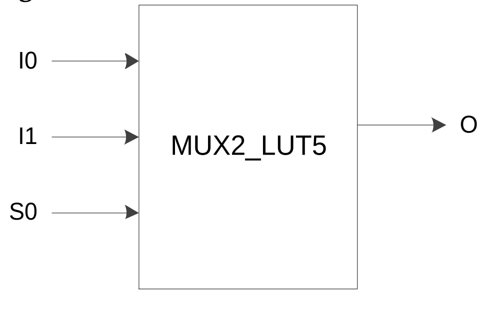
The Wide LUT is used for constructing high-order LUT using LUT4 and MUX2. MUX2 series of Gowin FPGA High-Order LUT support MUX2_LUT5/ MUX2_LUT6/ MUX2_LUT7/ MUX2_LUT8.

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.

Architecture Overview

Figure 2-6 MUX2_LUT5 View



Port Description

Table2-13 MUX2_LUT5 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

Table2-14 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 with multiple input. It transmits one data to the output according to the channel-select signal. The Gowin MUX contains 2-to-1 multiplexer and 4-to-1 multiplexer.

It supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

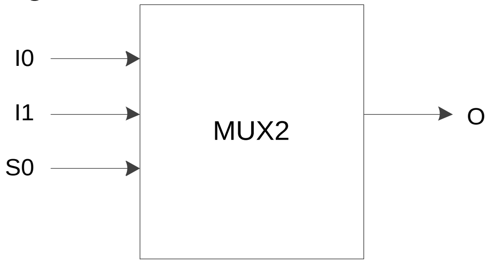
2.2.1 MUX2

Primitive Introduction

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

Architecture Overview

Figure 2-7 MUX2 View



Port Description

Table2-15 MUX2 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

Table2-16 Truth Table

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

Primitive Instantiation

Verilog Instantiation:

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

Vhdl Instantiation:

```
COMPONENT MUX2
        PORT(
                O:OUT std_logic;
                I0:IN std_logic;
                I1:IN std_logic;
                S0:IN std_logic
        );
```

```
END COMPONENT;
uut:MUX2
      PORT MAP (
          O=>O,
          I0=>I0,
          I1=>I1,
          S0=>S0
     );
```

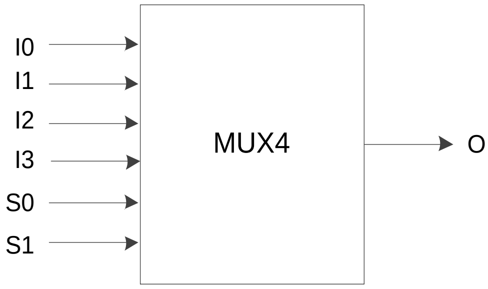
2.2.2 MUX4

Primitive Introduction

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

Architecture Overview

Figure 2-8 MUX4 View



Port Description

Table2-17 MUX4 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

Table2-18 MUX4 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

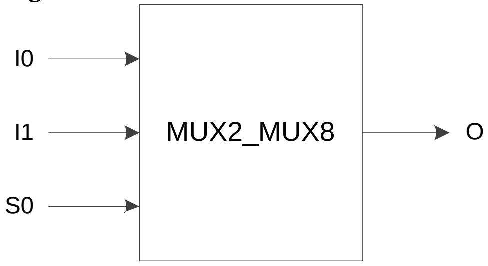
The Wide MUX is used for constructing high-order MUX using MUX4 and MUX2. The MUX2 series of Gowin FPGA High-Order MUX support MUX2_MUX8/ MUX2_MUX16/ MUX2_MUX32.

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.

Architecture Overview

Figure 2-9 MUX2_MUX8 View



Port Description

Table2-19 MUX2_MUX8 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

Table2-20 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 2-input Arithmetic Logic Unit (ALU) has the functions of ADD/SUB/ADDSUB.

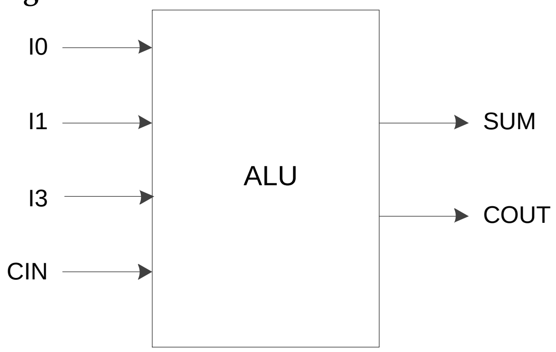
The ALU supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices. Specific functions are listed in Table2-21.

Table2-21 ALU Functions

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

Architecture Overview

Figure 2-10 ALU View



Port Description

Table2-22 ALU Port Description

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

Attribute Description

Table2-23 ALU Attribute Description

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.

The FF supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices. There are 20 primitives associated with FF, as shown in Table2-24.

Table2-24 Primitives Associated With FF

Primitive	Description
DFF	D Flip-Flop
DFFE	D Flip-Flop with Clock Enable
DFFS	D Flip-Flop with Synchronous Set
DFFSE	D Flip-Flop with Clock Enable and Synchronous Set
DFFR	D Flip-Flop with Synchronous Reset
DFFRE	D Flip-Flop with Clock Enable and Synchronous Reset
DFFP	D Flip-Flop with Asynchronous Preset
DFFPE	D Flip-Flop with Clock Enable and Asynchronous Preset
DFFC	D Flip-Flop with Asynchronous Clear
DFFCE	D Flip-Flop with Clock Enable and Asynchronous Clear
DFFN	D Flip-Flop with Negative-Edge Clock
DFFNE	D Flip-Flop with Negative-Edge Clock and Clock Enable
DFFNS	D Flip-Flop with Negative-Edge Clock and Synchronous Set
DFFNSE	D Flip-Flop with Negative-Edge Clock,Clock Enable,and Synchronous Set
DFFNR	D Flip-Flop with Negative-Edge Clock and Synchronous Reset
DFFNRE	D Flip-Flop with Negative-Edge Clock,Clock Enable, and Synchronous Reset
DFFNP	D Flip-Flop with Negative-Edge Clock and Asynchronous Preset
DFFNPE	D Flip-Flop with Negative-Edge Clock,Clock Enable, and Asynchronous Preset
DFFNC	D Flip-Flop with Negative-Edge Clock and Asynchronous Clear
DFFNCE	D Flip-Flop with Negative-Edge Clock,Clock Enable and Asynchronous Clear

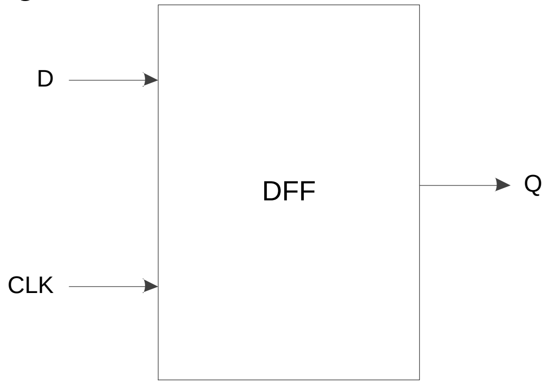
2.4.1 DFF

Primitive Introduction

The D Flip-Flop (DFF) is a D flip-flop flipped by rising edge, which is commonly used for signal sampling and processing.

Architecture Overview

Figure 2-11 DFF View



Port Description

Table2-25 DFF Port Description

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

Attribute Description

Table2-26 DFF Attribute Description

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
          );
```

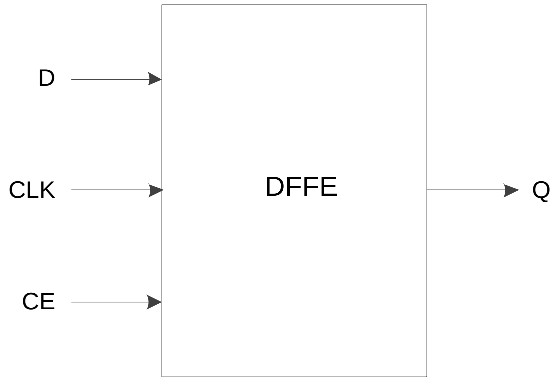
2.4.2 DFFE

Primitive Introduction

The D Flip-Flop with Clock Enable (DFFE) is a D flip-flop flipped by rising edge, with the function of clock enable.

Architecture Overview

Figure 2-12 DFFE View



Port Description

Table2-27 DFFE 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

Table2-28 DFFE Attribute Description

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
       );
```

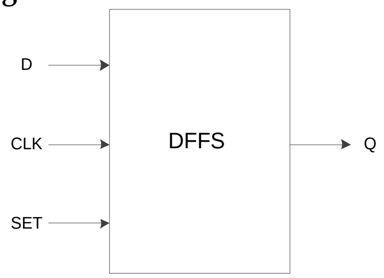
2.4.3 DFFS

Primitive Introduction

The D Flip-Flop with Synchronous Set (DFFS) is a D flip-flop flipped by rising edge, with the function of synchronous setting.

Architecture Overview

Figure 2-13 DFFS View



Port Description

Table2-29 DFFS 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

Table2-30 DFFE Attribute Description

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
        );
```

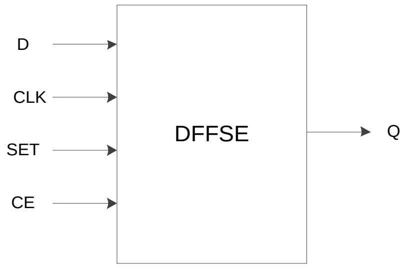
2.4.4 DFFSE

Primitive Introduction

The D Flip-Flop with Clock Enable and Synchronous Set (DFFSE) is a D flip-flop flipped by rising edge, with the functions of synchronous setting and clock enable.

Architecture Overview

Figure 2-14 DFFSE View



Port Description

Table2-31 DFFSE 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

Table2-32 DFFSE Attribute Description

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
        );
```

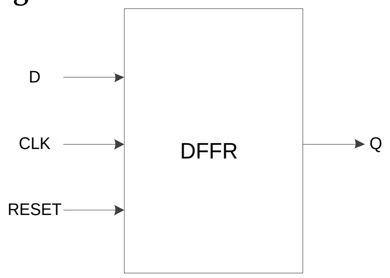
2.4.5 DFFR

Primitive Introduction

The D Flip-Flop with Synchronous Reset (DFFR) is a D flip-flop flipped by rising edge, with the function of synchronous resetting.

Architecture Overview

Figure 2-15 DFFR View



Port Description

Table2-33 DFFR 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

Table2-34 DFFR Attribute Description

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
        );
```

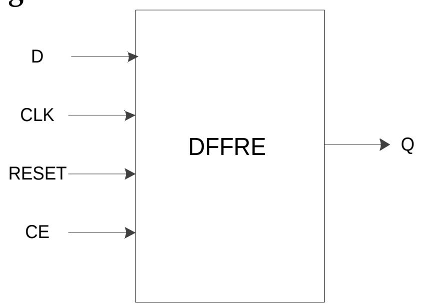
2.4.6 DFFRE

Primitive Introduction

The D Flip-Flop with Clock Enable and Synchronous Reset (DFFRE) is a D flip-flop flipped by rising edge, with the functions of synchronous setting and clock enable.

Architecture Overview

Figure 2-16 DFFRE View



Port Description

Table2-35 DFFRE 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

Table2-36 DFFRE Attribute Description

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
        );
```

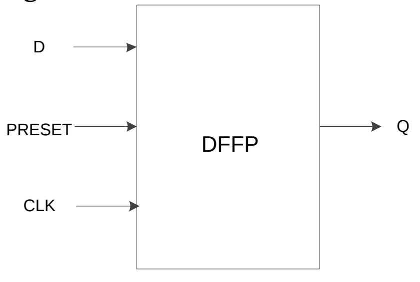
2.4.7 DFFP

Primitive Introduction

The D Flip-Flop with Asynchronous Preset (DFFP)is a D flip-flop flipped by rising edge, with the function of synchronous setting.

Architecture Overview

Figure 2-17 DFFP View



Port Description

Table2-37 DFFP 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

Table2-38 DFFP Attribute Description

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
        );
```

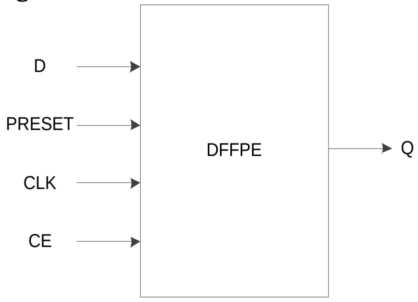
2.4.8 DFFPE

Primitive Introduction

The D Flip-Flop with Clock Enable and Asynchronous Preset (DFFPE) is a rising edge D trigger with the functions of synchronous setting and clock enable.

Architecture Overview

Figure 2-18 DFFPE View



Port Description

Table2-39 DFFPE 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

Table2-40 DFFPE Attribute Description

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
                );
```

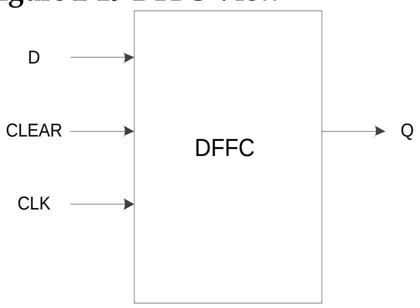
2.4.9 DFFC

Primitive Introduction

The D Flip-Flop with Asynchronous Clear (DFFC) is a D flip-flop flipped by rising edge, with the function of synchronous setting.

Architecture Overview

Figure 2-19 DFFC View



Port Description

Table2-41 DFFC 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

Table2-42 DFFC Attribute Description

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
          );
```

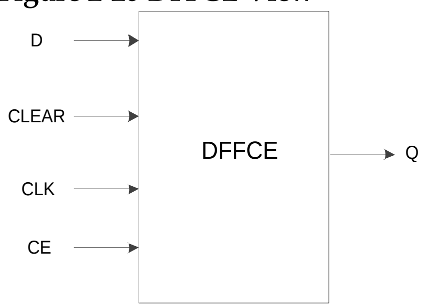
2.4.10 DFFCE

Primitive Introduction

The D Flip-Flop with Clock Enable and Asynchronous Clear (DFFCE) is a D flip-flop flipped by rising edge, with the functions of synchronous setting and clock enable.

Architecture Overview

Figure 2-20 DFFCE View



Port Description

Table2-43 DFFCE 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

Table2-44 DFFCE Attribute Description

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
        );
```

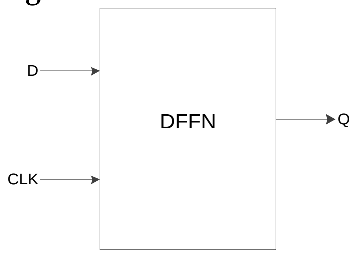
2.4.11 DFFN

Primitive Introduction

The D Flip-Flop with Negative-Edge Clock (DFFN) is a D flip-flop flipped by falling edge.

Architecture Overview

Figure 2-21 DFFN View



Port Description

Table2-45 DFFN Port Description

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

Attribute Description

Table2-46 DFFN Attribute Description

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
);
```

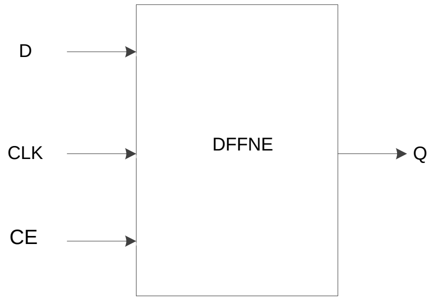
2.4.12 DFFNE

Primitive Introduction

The DFFNE is a D flip-flop flipped by falling edge, with the function of clock enable.

Architecture Overview

Figure 2-22 DFFNE View



Port Description

Table2-47 DFFNE 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

Table2-48 DFFNE Attribute Description

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
        );
```

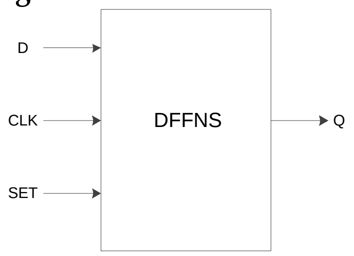
2.4.13 DFFNS

Primitive Introduction

The D Flip-Flop with Negative-Edge Clock and Synchronous Set (DFFNS) is a D flip-flop flipped by falling edge, with the function of synchronous setting.

Architecture Overview

Figure 2-23 DFFNS View



Port Description

Table2-49 DFFNS 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

Table2-50 DFFNS Attribute Description

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
          );
```

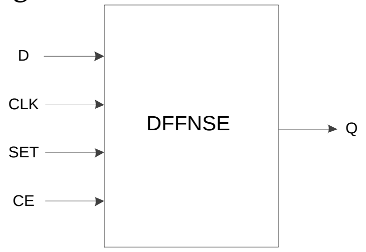
2.4.14 DFFNSE

Primitive Introduction

The D Flip-Flop with Negative-Edge Clock,Clock Enable,and Synchronous Set (DFFNSE) is a D flip-flop flipped by falling edge, with the functions of synchronous setting and clock enable.

Architecture Overview

Figure 2-24 DFFNSE View



Port Description

Table2-51 DFFNSE 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

Table2-52 DFFNSE Attribute Description

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
        );
```

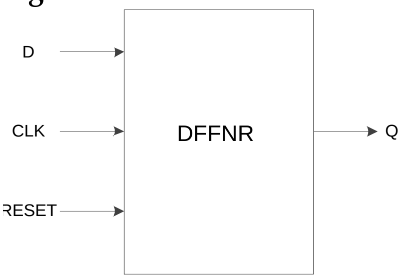
2.4.15 DFFNR

Primitive Introduction

The D Flip-Flop with Negative-Edge Clock and Synchronous Reset (DFFNR) is a D trigger of falling edge with the function of synchronous resetting.

Architecture Overview

Figure 2-25 DFFNR View



Port Description

Table2-53 DFFNR 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

Table2-54 DFFNR Attribute Description

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
          );
```

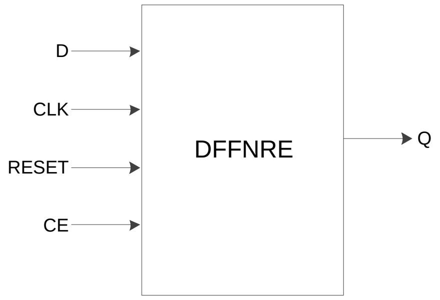
2.4.16 DFFNRE

Primitive Introduction

The D Flip-Flop with Negative-Edge Clock,Clock Enable, and Synchronous Reset (DFFNRE) is a D flip-flop flipped by falling edge, with the functions of synchronous resetting and clock enable.

Architecture Overview

Figure 2-26 DFFNRE View



Port Description

Table2-55 DFFNRE 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

Table2-56 DFFNRE Attribute Description

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
         );
```

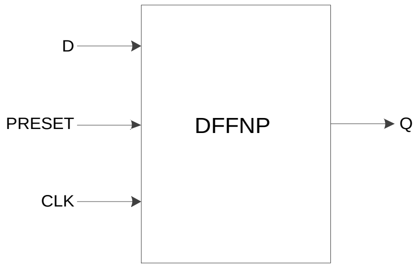
2.4.17 DFFNP

Primitive Introduction

The D Flip-Flop with Negative-Edge Clock and Asynchronous Preset (DFFNP) is a D trigger of falling edge with the function of asynchronous setting.

Architecture Overview

Figure 2-27 DFFNP View



Port Description

Table2-57 DFFNP 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

Table2-58 DFFNP Attribute Description

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
          );
```

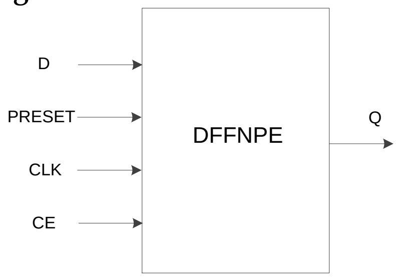
2.4.18 DFFNPE

Primitive Introduction

The D Flip-Flop with Negative-Edge Clock,Clock Enable, and Asynchronous Preset (DFFNPE) is a D trigger of falling edge with the functions of asynchronous setting and clock enable.

Architecture Overview

Figure 2-28 DFFNPE View



Port Description

Table2-59 DFFNPE 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

Table2-60 DFFNPE Attribute Description

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
         );
```

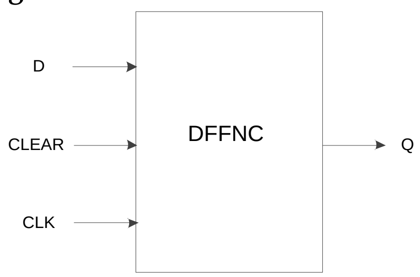
2.4.19 DFFNC

Primitive Introduction

The D Flip-Flop with Negative-Edge Clock and Asynchronous Clear (DFFNC) is a D flip-flop flipped by falling edge, with the function of asynchronous resetting.

Architecture Overview

Figure 2-29 DFFNC View



Port Description

Table2-61 DFFNC 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

Table2-62 DFFNC Attribute Description

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
          );
```

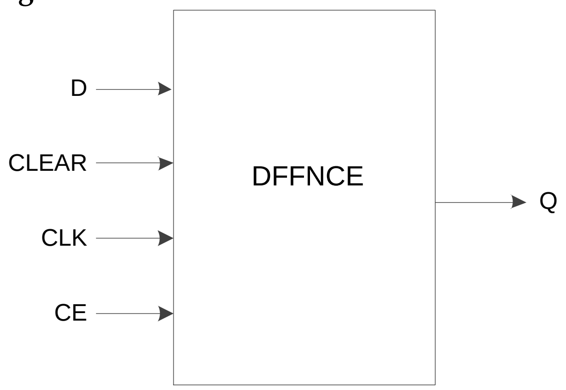
2.4.20 DFFNCE

Primitive Introduction

The D Flip-Flop with Negative-Edge Clock,Clock Enable and Asynchronous Clear (DFFNCE) is a D flip-flop flipped by falling edge, with the functions of asynchronous resetting and clock enable.

Architecture Overview

Figure 2-30 DFFNCE View



Port Description

Table2-63 DFFNCE 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

Table2-64 DFFNCE Attribute Description

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.

LATCH supports the GW1N-1, GW1N-1S, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices. There are 12 primitives associated with FF, as shown in Table2-24.

Table2-65 Primitives Associated With LATCH

Primitive	Description
DL	Data Latch
DLE	Data Latch with Latch Enable
DLC	Data Latch with Asynchronous Clear
DLCE	Data Latch with Asynchronous Clear and Latch Enable
DLP	Data Latch with Asynchronous Preset
DLPE	Data Latch with Asynchronous Preset and Latch Enable
DLN	Data Latch with Inverted Gate
DLNE	Data Latch with Latch Enable and Inverted Gate
DLNC	Data Latch with Asynchronous Clear and Inverted Gate
DLNCE	Data Latch with Asynchronous Clear, Latch Enable, and Inverted Gate
DLNP	Data Latch with Asynchronous Clear and Inverted Gate
DLNPE	Data Latch with Asynchronous Preset,Latch Enable and Inverted Gate

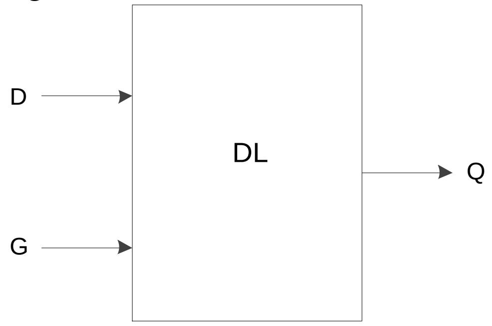
2.5.1 DL

Primitive Introduction

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

Architecture Overview

Figure 2-31 DL View



Port Description

Table2-66 DL Port Description

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

Attribute Description

Table2-67 DL Attribute Description

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
        );
```

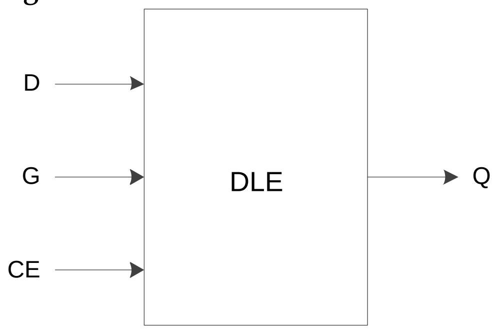
2.5.2 DLE

Primitive Introduction

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

Architecture Overview

Figure 2-32 DLE View



Port Description

Table2-68 DLE 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

Table2-69 DLE Attribute Description

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
      );
```

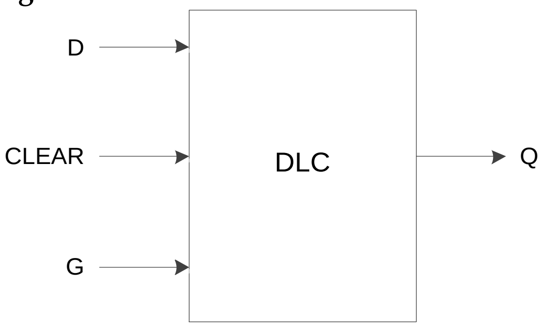
2.5.3 DLC

Primitive Introduction

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

Architecture Overview

Figure 2-33 DLC View



Port Description

Table2-70 DLC 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

Table2-71 DLC Attribute Description

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
        );
```

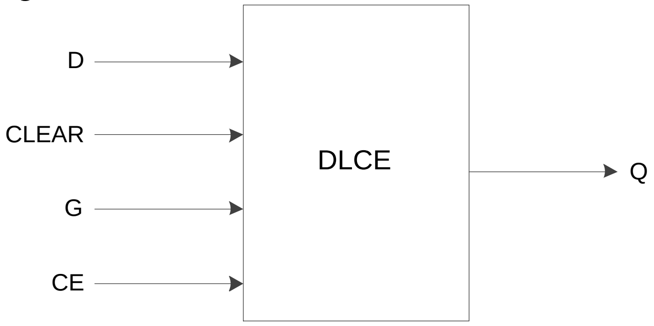
2.5.4 DLCE

Primitive Introduction

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

Architecture Overview

Figure 2-34 DLCE View



Port Description

Table2-72 DLCE 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

Table2-73 DLCE Attribute Description

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
     );
```

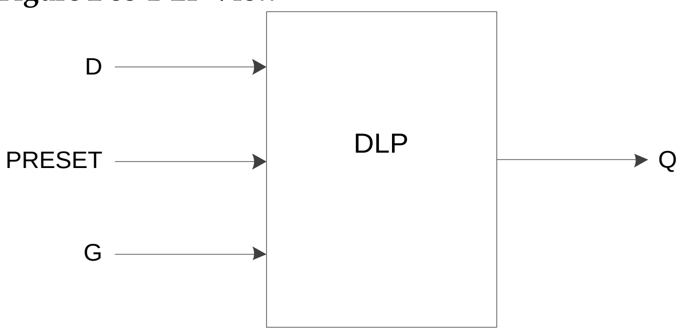
2.5.5 DLP

Primitive Introduction

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

Architecture Overview

Figure 2-35 DLP View



Port Description

Table2-74 DLP 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

Table2-75 DLP Attribute Description

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
          );
```

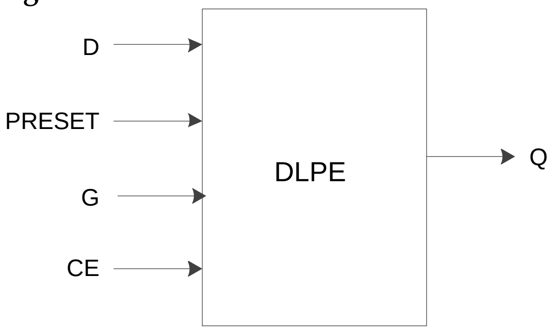
2.5.6 DLPE

Primitive Introduction

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

Architecture Overview

Figure 2-36 DLPE View



Port Description

Table2-76 DLPE 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

Table2-77 DLPE Attribute Description

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
          );
```

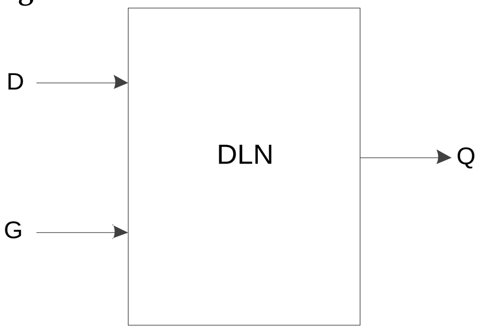
2.5.7 DLN

Primitive Introduction

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

Architecture Overview

Figure 2-37 DLN View



Port Description

Table2-78 DLN Port Description

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

Attribute Description

Table2-79 DLN Attribute Description

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
     );
```

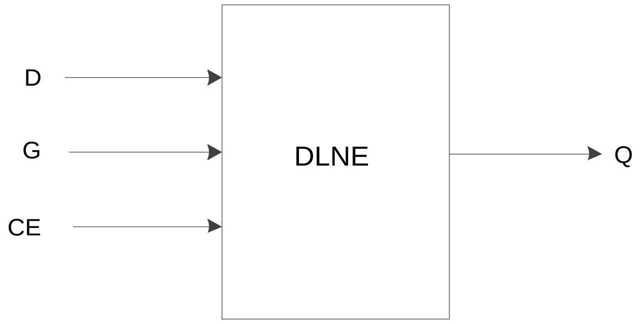
2.5.8 DLNE

Primitive Introduction

The DLNE is a latch with the function of enable control, and control signal G is low-active.

Architecture Overview

Figure 2-38 DLNE View



Port Description

Table2-80 DLNE 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

Table2-81 DLNE Attribute Description

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
      );
```

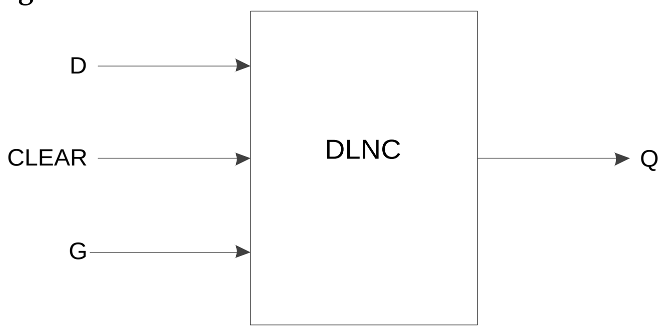
2.5.9 DLNC

Primitive Introduction

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

Architecture Overview

Figure 2-39 DLNC View



Port Description

Table2-82 DLNC 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

Table2-83 DLNC Attribute Description

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
             );
```

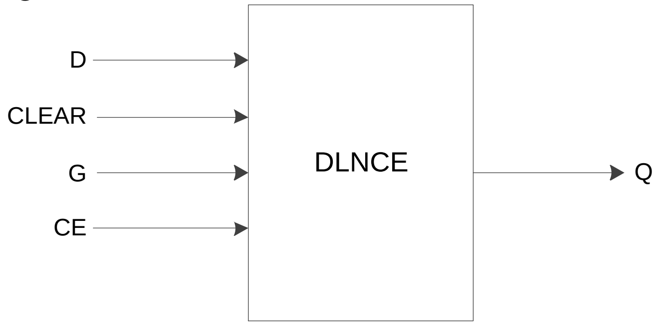
2.5.10 DLNCE

Primitive Introduction

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

Architecture Overview

Figure 2-40 DLNCE View



Port Description

Table2-84 DLNCE 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

Table2-85 DLNCE Attribute Description

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
            );
```

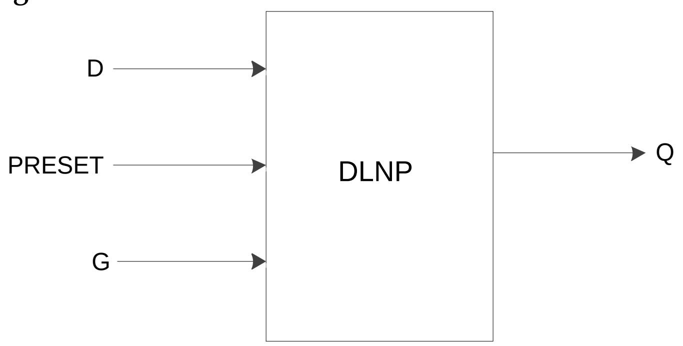
2.5.11 DLNP

Primitive Introduction

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

Architecture Overview

Figure 2-41 DLNP View



Port Description

Table2-86 DLNP 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

Table2-87 DLNP Attribute Description

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
        );
```

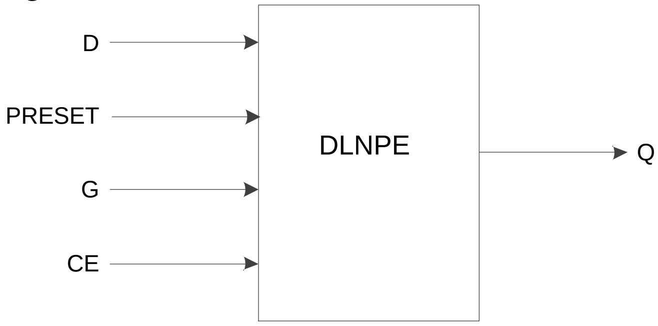
2.5.12 DLNPE

Primitive Introduction

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

Architecture Overview

Figure 2-42 DLNPE View



Port Description

Table2-88 DLNPE 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

Table2-89 DLNPE Attribute Description

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

Different with CLU, the Configurable Fuction Unit (CFU) can be configured as SSRAM mode.

3.1 SSRAM

The SSRAM is a static shadow random memory that can be configured as a single Port mode, a semi-dual mode and a read-only mode, as shown in Table3-1.

The SSRAM supports the GW1NS-2, GW1NS-2C, GW1N-6, GW1N-9, GW1NR-9, GW1NZ-1, GW1NSR-2, GW1NSR-2C, GW2A-18, GW2AR-18, and GW2A-55 devices.

Table3-1 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 SSRAM with address depth 16 and data width 1
RAM16SDP2	Semi-dual SSRAM with address depth 16 and data width 2
RAM16SDP4	Semi-dual SSRAM with address depth 16 and data width 4
ROM16	Read-only ROM with address depth 16 and data width 1

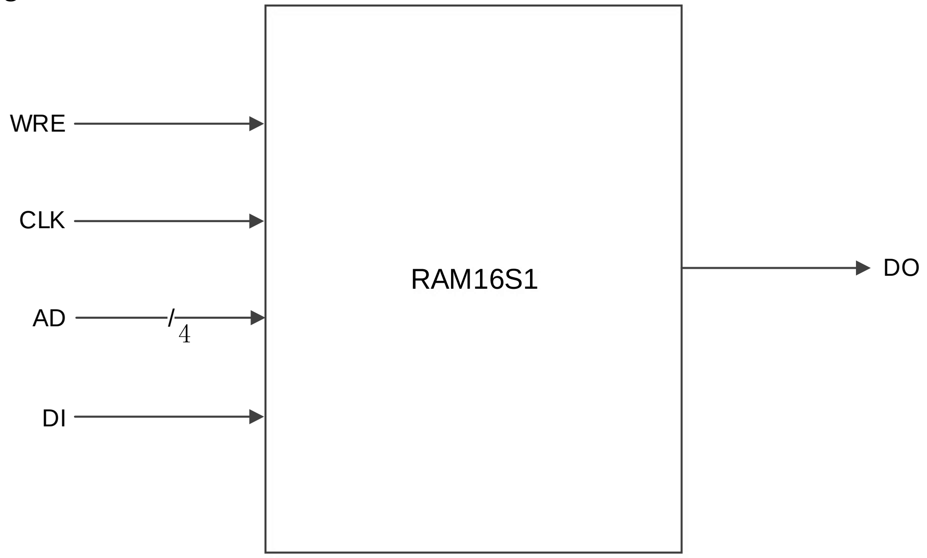
3.1.1 RAM16S1

Primitive Introduction

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

Architecture Overview

Figure 3-1 RAM16S1 View



Port Description

Table3-2 RAM16S1 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

Table3-3 RAM16S1 Attribute Description

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
           );
```

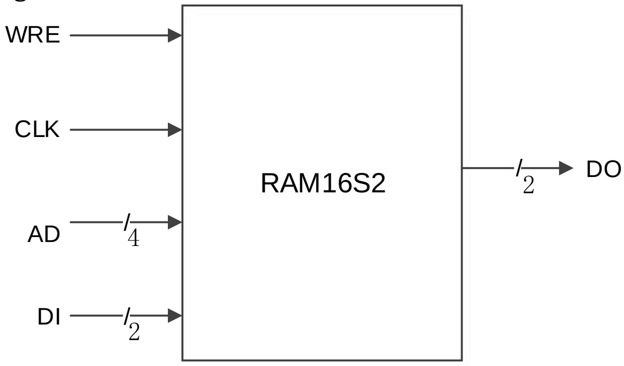
3.1.2 RAM16S2

Primitive Introduction

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

Architecture Overview

Figure 3-2 RAM16S2 View



Port Description

Table3-4 RAM16S2 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

Table3-5 RAM16S2 Attribute Description

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
            );
```

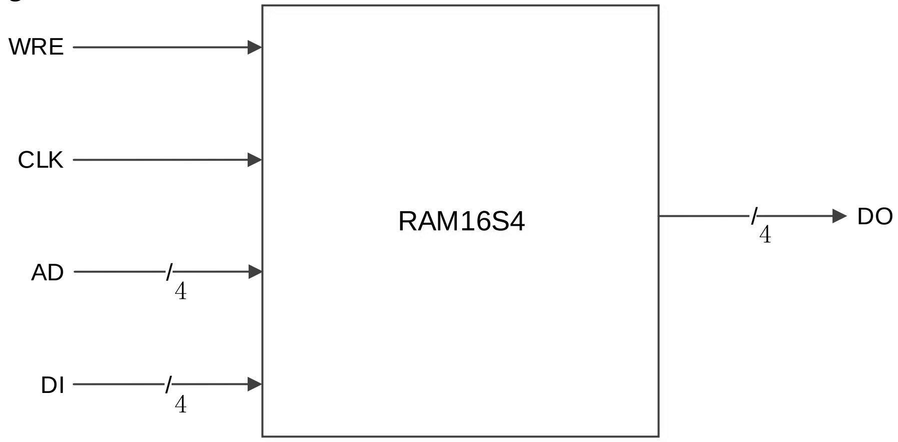
3.1.3 RAM16S4

Primitive Introduction

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

Architecture Overview

Figure 3-3 RAM16S4 View



Port Description

Table3-6 RAM16S4 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

Table3-7 RAM16S4 Attribute Description

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
       );
```

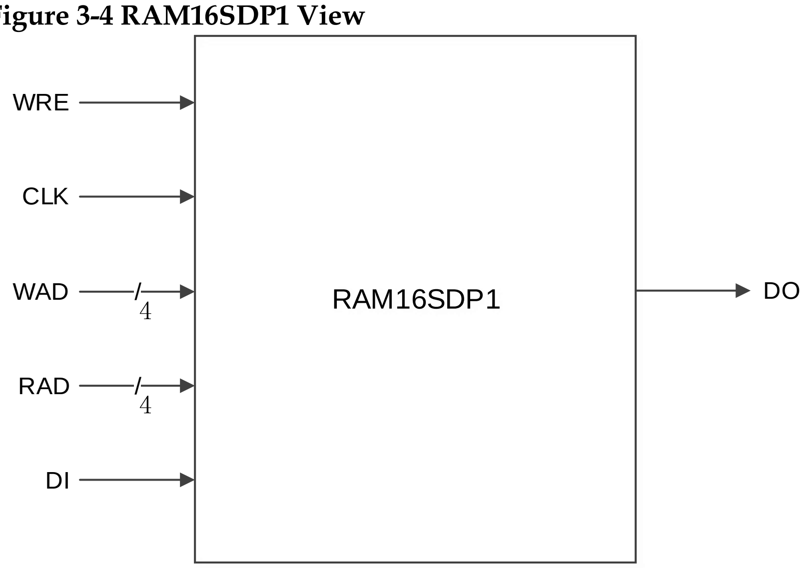
3.1.4 RAM16SDP1

Primitive Introduction

The 16-Deep by 1-Wide Semi Dual-port SSRAM (RAM16SDP1) is a semi-dual SSRAM with address depth 16 and data width1.

Architecture Overview

Figure 3-4 RAM16SDP1 View



Port Description

Table3-8 RAM16SDP1 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

Table3-9 RAM16SDP1 Attribute Description

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
    );
```

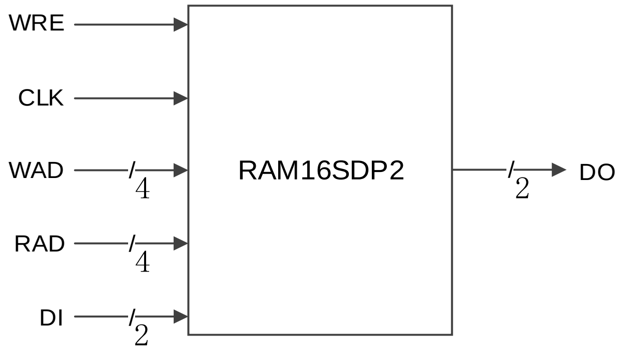
3.1.5 RAM16SDP2

Primitive Introduction

The 16-Deep by 2-Wide Semi Dual-port SSRAM (RAM16SDP2) is a semi-dual SSRAM with address depth16 and data width 2.

Architecture Overview

Figure 3-5 RAM16SDP2 View



Port Description

Table3-10 RAM16SDP2 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

Table3-11 RAM16SDP2 Attribute Description

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
       );
```

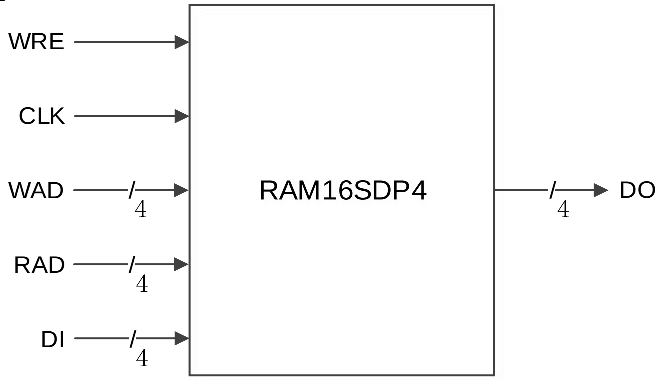
3.1.6 RAM16SDP4

Primitive Introduction

The 16-Deep by 4-Wide Semi Dual-port SSRAM (RAM16SDP4) is a semi-dual SSRAM with address depth16 and data width 4.

Architecture Overview

Figure 3-6 RAM16SDP4 View



Port Description

Table3-12 RAM16SDP4 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

Table3-13 RAM16SDP4 Attribute Description

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
      );
```

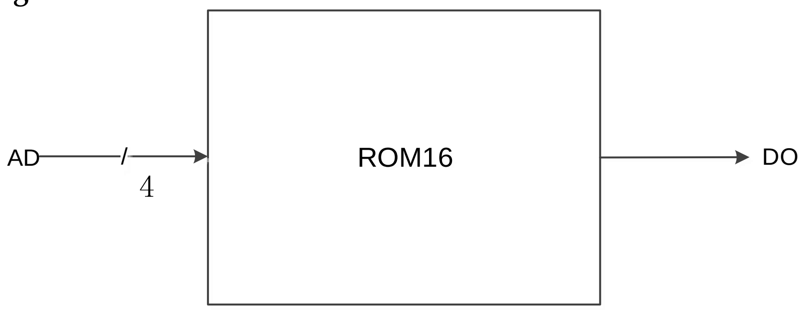
3.1.7 ROM16

Primitive Introduction

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

Architecture Overview

Figure 3-7 ROM16 View



Port Description

Table3-14 ROM16 Port Description

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

Attribute Description

Table3-15 ROM16 Attribute Description

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 register that performs the function of static accessing. 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).

Block SRAM supPorts the GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1N-2B, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, and GW2A-55 devices.

4.1 SP/SPX9

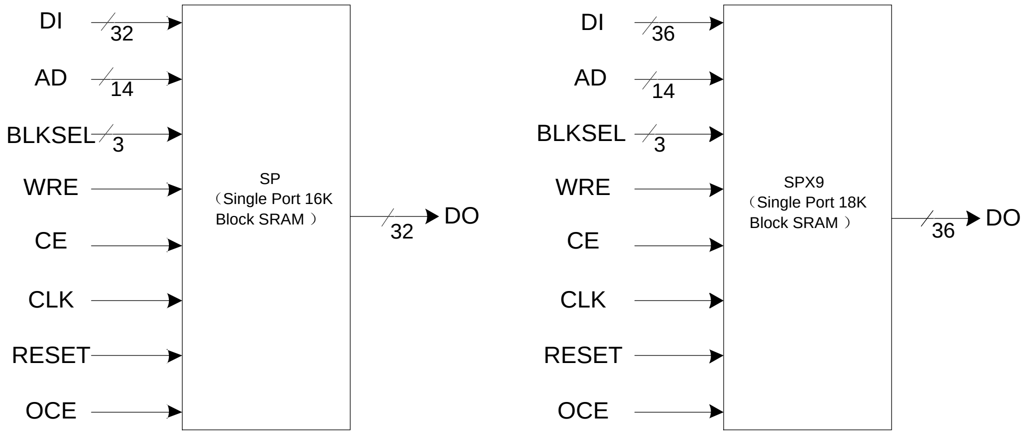
Primitive Introduction

The Single Port 16K Block SRAM/Single Port 18K Block SRAM (SP/SPX9) works in single Port mode with a memory space 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 are controlled by the low four bit of AD, high enable. AD[0] controls whether to write DI[7:0]/DI[8:0] to memory; AD[1] controls whether to write DI[15:8]/DI[17:9] to memory; AD[2] controls whether to write DI[23:16]/DI[26:18] to memory; AD[3] controls whether to write DI[31:24]/DI[35:27] to memory.

Architecture Overview

Figure 4-1 SP/SPX9 View



Port Description

Table4-1 SP/SPX9 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
CE	Input	Clock Enable
CLK	Input	Clock input
RESET	Input	Reset Input
OCE	Input	Output Clock Enable
BLKSEL[2:0]	Input	Block RAM Selection Input

Attribute Description

Table4-2 SP/SPX9 Attribute Description

Attribute Name	Type	Permitted Values	Default	Description
READ_MODE	Integer	1'b0,1'b1	1'b0	Output pipeline register can be bypassed ● 1'b0:bypass mode ● 1'b1:pipeline mode
WRITE_MODE	Integer	2'b00,2'b01,2'b10	2'b00	Write mode can be selected. ● 2'b00: normal mode ● 2'b01:write-through mode; ● 2'b10: read-before-write mode
BIT_WIDTH	Integer	SP:1,2,4,8,16,32 SPX9:9,18,36	SP:32 SPX9:36	Data width can be configured
BLK_SEL	Integer	3'b000~3'b111	3'b000	Block SRAM Selection
RESET_MODE	String	SYNC,ASYNC	SYNC	Reset mode config, synchronous or asynchronous
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	Initial value for initial BSRAM

Configuration Relationships

Table4-3 Configuration Relationships

Single Port Mode	BSRAM Capacity	Data Width	Address Depth
SP	16K	1	14
		2	13
		4	12
		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"00000000000000000000
0000000000000000000000000000000000000000000000000000",
                                   INIT_RAM_01=>X"00000000000000000000
0000000000000000000000000000000000000000000000000000",
                                   INIT_RAM_3F=>X"00000000000000000000
0000000000000000000000000000000000000000000000000000"
                 )
                 PORT MAP(
                       DO=>dout,
                       CLK=>clk,
                       OCE=>oce,
                       CE=>ce,
                       RESET=>reset,
                       WRE=>wre,
                       BLKSEL=>blksel,
                       AD=>ad,
                       DI=>din
                 );
```

4.2 SDP/SDPX9

Primitive Introduction

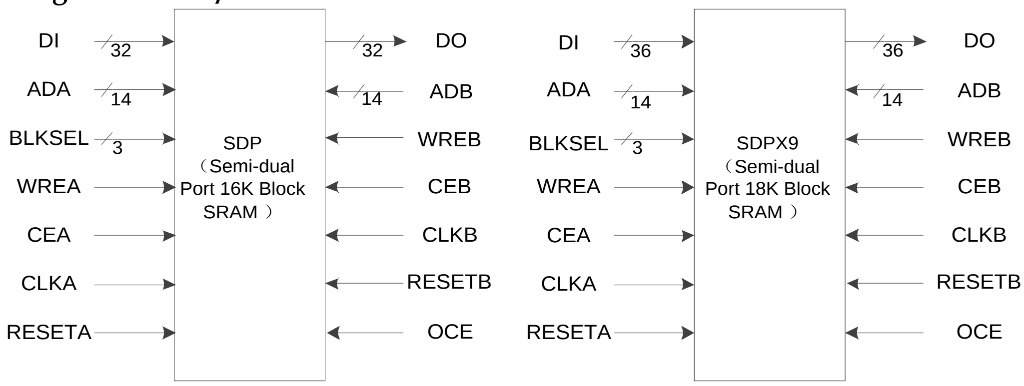
The Semi Dual Port 16K Block SRAM /Semi Dual Port 18K Block

SRAM (SDP/SDPX9) works in semi-dual Port mode with a memory space of 16K bit/18K bit. Write takes place at Port A and read at Port B. 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 are controlled by the low four bit of AD, high-level enable. ADA[0] controls whether to write DI[7:0]/DI[8:0] to memory; ADA[1] controls whether to write DI[15:8]/DI[17:9] to memory; ADA[2] controls whether to write DI[23:16]/DI[26:18] to memory; ADA[3] controls whether to write DI[31:24]/DI[35:27] to memory.

Architecture Overview

Figure 4-2 SDP/SDPX9 View



Port Description

Table4-4 SP/SPX9 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	Port A Address
ADB[13:0]	Input	Port B Address
WREA	Input	Port A Write Enable
WREB	Input	PortB Write Enable
CEA	Input	Port A Clock Enable
CEB	Input	Port B Clock Enable
CLKA	Input	Port A Clock Input
CLKB	Input	Port B Clock Input
RESETA	Input	Port A Reset Input
RESETB	Input	PortB Reset Input
OCE	Input	Output Clock Enable
BLKSEL[2:0]	Input	Block SRAM Selection Input

Attribute Description

Table4-5 SP/SPX9 Attribute Description

Attribute Name	Type	Permitted Values	Default	Description
READ_MODE	Integer	1’b0,1’b1	1’b0	Output pipeline register can be bypassed ● 1’b0:bypass mode ● 1’b1:pipeline mode
BIT_WIDTH_0	Integer	SDP:1,2,4,8,16,32 SDPX9:9,18,36	SDP:32 SDPX9:36	Port A’s data width can be configured
BIT_WIDTH_1	Integer	SDP:1,2,4,8,16,32 SDPX9:9,18,36	SDP:32 SDPX9:36	Port B’s data width can be configured
BLK_SEL	Integer	3’b000~3’b111	3’b000	Block SRAM Selection
RESET_MODE	String	SYNC,ASYNC	SYNC	Reset mode config, synchronous orasynchronous
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	Initial value for initial BSRAM

Configuration Relationships

Table4-6 Configuration Relationships

Single 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
		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
    );
```

4.3 DP/DPX9

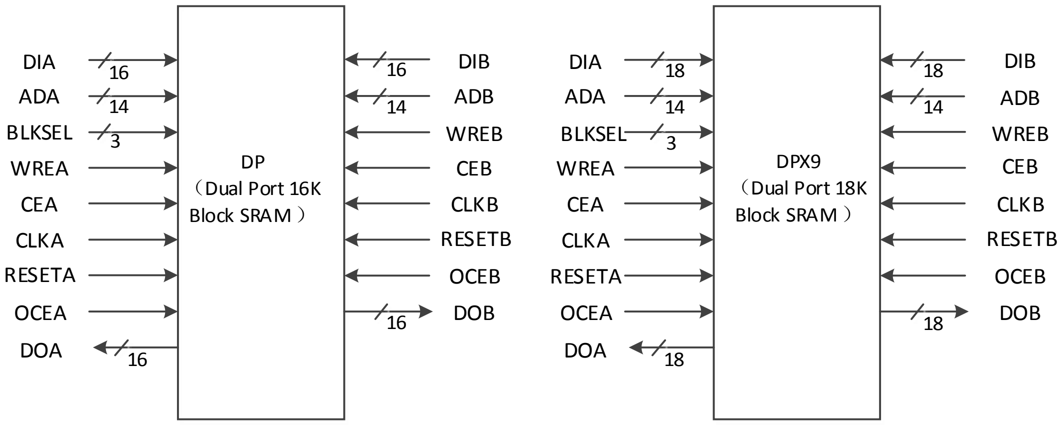
Primitive Introduction

The True Dual Port 16K Block SRAM/True Dual Port 18K Block SRAM (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/32bit and DPX9 is configured as 18bit/36bit, the byte enable function of BSRAM can be realized, i.e., the data written to memory are controlled by the low four bit of AD, high level enable. ADA[0] controls whether to write DIA[7:0]/DIA[8:0] to memory; ADA[1] controls whether to write DIA[15:8]/DIA[17:9] to memory; ADB[0] controls whether to write DIB[7:0]/DIB[8:0] to memory; ADB[1] controls whether to write DIB[15:8]/DIB[17:9] to memory.

Architecture Overview

Figure4-3 DP/DPX9 View



Port Description

Table4-7 DP/DPX9 Port Description

Port Name	I/O	Description
DOA[15:0]/DOA[17:0]	Output	Port A Data Output
DOB[15:0]/DOB[17:0]	Output	Port B Data Output
DIA[15:0]/DIA[17:0]	Input	Port A Data Input
DIB[15:0]/DIB[17:0]	Input	Port B Data Input
ADA[13:0]	Input	Port A Address
ADB[13:0]	Input	Port B Address
WREA	Input	Port A Write Enable
WREB	Input	Port B Write Enable
CEA	Input	Port A Clock Enable
CEB	Input	Port B Clock Enable
CLKA	Input	Port A Clock Input
CLKB	Input	Port B Clock Input
RESETA	Input	Port A Reset Input
RESETB	Input	Port B Reset Input
OCEA	Input	Port A Output Clock Enable
OCEB	Input	Port B Output Clock Enable
BLKSEL[2:0]	Input	Block RAM Selection

Attribute Description

Table4-8 DP/DPX9 Attribute Description

Attribute Name	Type	Permitted Values	Default	Description
READ_MODE0	Integer	1’b0,1’b1	1’b0	Port A‘s output pipeline register can be bypassed ● 1’b0:bypass mode ● 1’b1:pipeline mode
READ_MODE1	Integer	1’b0,1’b1	1’b0	Port B‘s output pipeline register can be bypassed ● 1’b0:bypass mode ● 1’b1:pipeline mode
WRITE_MODE0	Integer	2’b00,2’b01,2’b10	2’b00	Port A’s write mode can be selected ● 2'b00: normal mode ● 2'b01: write-through mode ● 2'b10: read-before-write mode
WRITE_MODE1	Integer	2’b00,2’b01,2’b10	2’b00	Port B’s write mode can be selected ● 2'b00: normal mode ● 2'b01: write-through mode ● 2'b10: read-before-write mode
BIT_WIDTH_0	Integer	DP:1,2,4,8,16,32 DPX9:9,18,36	DP:32 DPX9:36	Port A’s data width can be configured
BIT_WIDTH_1	Integer	DP:1,2,4,8,16,32 DPX9:9,18,36	DP:32 DP:36	Port B’s data width can be configured
BLK_SEL	Integer	3’b000~3’b111	3’b000	Block RAM Selection
RESET_MODE	String	SYNC,ASYNC	SYNC	Reset mode config, synchronous or asynchronous
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	Initial value for initial BSRAM

Configuration Relationships

Table4-9 Configuration Relationships

Single 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
                 00000000000000000000000000000000000000000000000000";
                                  INIT_RAM_01:bit_vector:=X"000000000000000
                 00000000000000000000000000000000000000000000000000";
                                  INIT_RAM_3F:bit_vector:=X"000000000000000
                 00000000000000000000000000000000000000000000000000"
                          );
                          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"000000000000000000000000
000000000000000000000000000000000000000000",
                                  INIT_RAM_01=>X"000000000000000000000000
000000000000000000000000000000000000000000",
                                  INIT_RAM_3F=>X"000000000000000000000000
000000000000000000000000000000000000000000"
                      )
                      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
000000000000000000000000000000000000000000000000000";
                    INIT_RAM_01:bit_vector:=X"00000000000000
000000000000000000000000000000000000000000000000000";
                    INIT_RAM_3F:bit_vector:=X"00000000000000
000000000000000000000000000000000000000000000000000"
          );
          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
000000000000000000000000000000000000000000000",
                      INIT_RAM_01=>X"0000000000000000000
000000000000000000000000000000000000000000000",
                      INIT_RAM_3F=>X"0000000000000000000
000000000000000000000000000000000000000000000"
          )
          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
        );
```

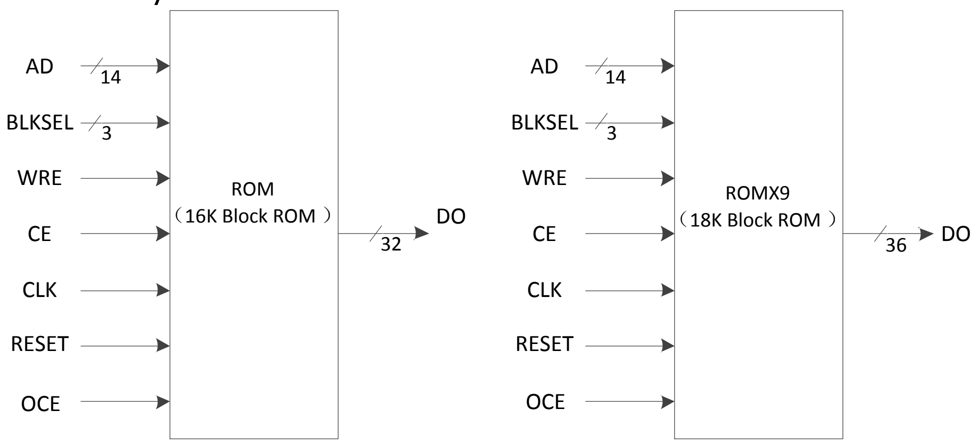
4.4 ROM/ROMX9

Primitive Introduction

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

Architecture Overview

Figure4-4 ROM/ROMX9 View



Port Description

Table4-10 ROM/ROMX9 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
CE	Input	Clock Enable
CLK	Input	Clock input
RESET	Input	Reset Input
OCE	Input	Output Clock Enable
BLKSEL[2:0]	Input	Block RAM Select

Attribute Description

Table4-11 ROM/ROMX9 Attribute Description

Attribute Name	Type	Permitted Values	Default	Description
READ_MODE	Integer	1’b0,1’b1	1’b0	Output pipeline register can be bypassed ● 1’b0:bypass mode ● 1’b1:pipeline mode
BIT_WIDTH	Integer	ROM:1,2,4,8,16,32 ROMX9:9,18,36	ROM:32 ROMX9:36	Data width can be configured
BLK_SEL	Integer	3’b000~3’b111	3’b000	Block RAM Selection
RESET_MODE	String	SYNC,ASYNC	SYNC	Reset mode config, synchronous or asynchronous
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	Initial value for initial BSRAM

Configuration Relationships

Table4-12 Configuration Relationships

Single Port Mode	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'h000000000000000000000000000000000000000000000000000
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
000000000000000000000000000000000000FFFFFFBDCF"
        );
        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
000000000000000000000000000000FFFFFFBDCF  "
     )
     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'h000000000000000000000000000000000000000000000000
0000000001FFFFFFF7ACF;
```

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
00000000000000000000000000000000000000001FFFFFFF7ACF"
         );
         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"00000000000000000000
000000000000000000000000000000000000001FFFFFFF7ACF"
        )
        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, which supports the GW1N-1, GW1N-2, GW1N-4, GW1NR-4, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, and GW2A-55 devices.

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.

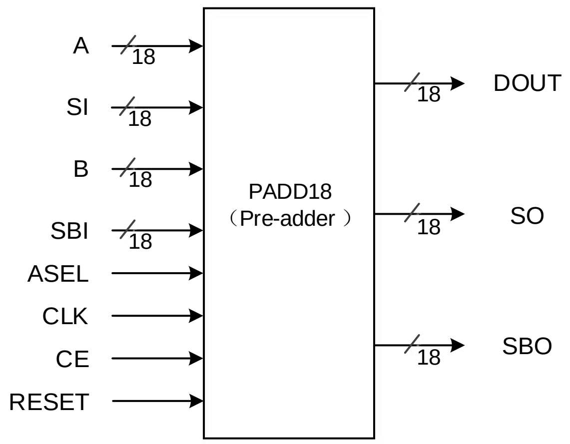
5.1.1 PADD18

Primitive Introduction

The 18-bit Pre-Adder (PADD18) performs the functions of pre-adding, pre-subtracting, and shifting.

Architecture Overview

Figure5-1 PADD18 Architecture Overview



Port Description

Table5-1 PADD18 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

Table5-2 PADD18 Attribute Description

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
);
```

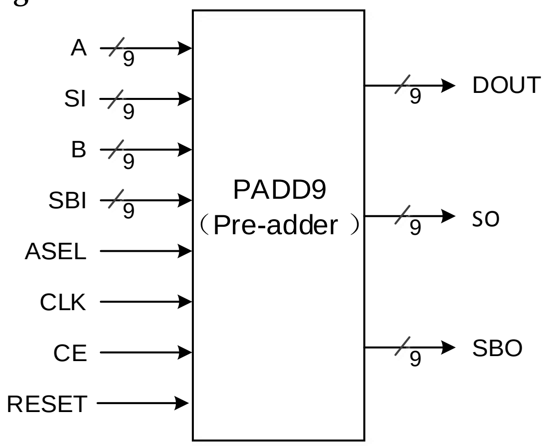
5.1.2 PADD9

Primitive Introduction

The 9-bit Pre-Adder (PADD9) can be configured as pre-adder, pre-subtracter, or shifter.

Architecture Overview

Figure5-2 PADD9 View



Port Description

Table5-3 PADD9 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

Table5-4 PADD9 Attribute Description

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.

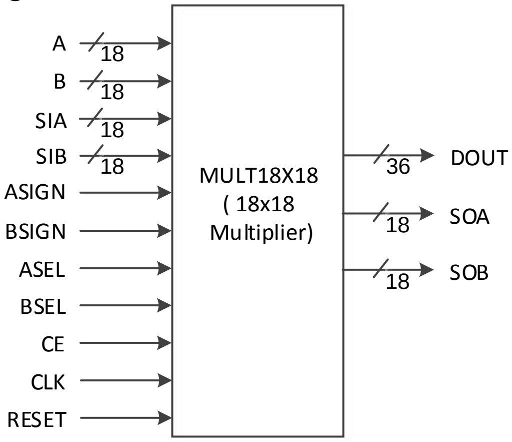
5.2.1 MULT18X18

Primitive Introduction

The 18x18 Multiplier (MULT18X18) supports 18-bit multiplication.

Architecture Overview

Figure5-3 MULT18X18 View



Port Description

Table5-5 MULT18X18 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

Table5-6 MULT18X18 Attribute Description

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 ● 1’b1:registeredmode
PIPE_REG	1’b0,1’b1	1’b0	Pipeline register can be bypassed ● 1’b0:bypass mode ● 1’b1:registeredmode
ASIGN_REG	1’b0,1’b1	1’b0	ASIGN input register can be bypassed ● 1’b0:bypass mode ● 1’b1:registeredmode
BSIGN_REG	1’b0,1’b1	1’b0	BSIGN input register can be bypassed ● 1’b0:bypass mode ● 1’b1:registeredmode
SOA_REG	1’b0,1’b1	1’b0	SOA register can bebypassed ● 1’b0:bypassmode ● 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
        );
```

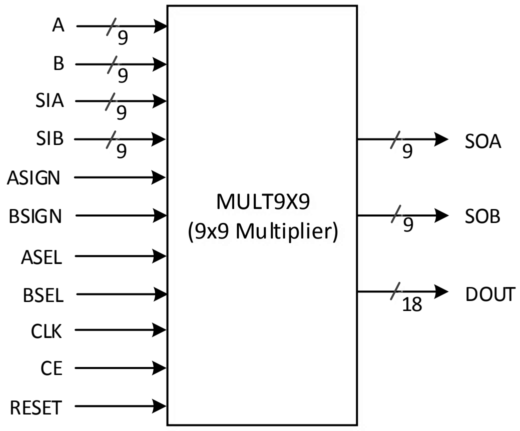
5.2.2 MULT9X9

Primitive Introduction

The 9x9 Multiplier (MULT9X9) supports 9-bit multiplication.

Architecture Overview

Figure5-4 MULT9X9 View



Port Description

Table5-7 MULT9X9 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

Table5-8 MULT9X9 Attribute Description

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 ● 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:bypassmode ● 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
        );
```

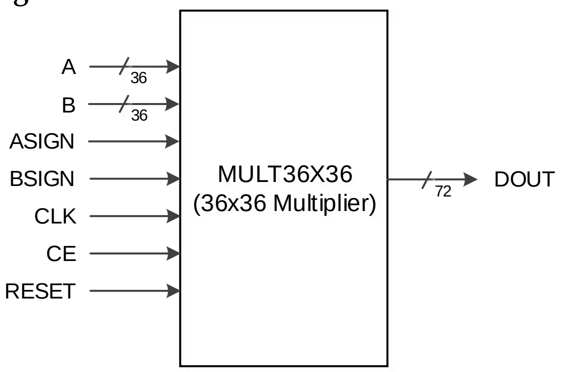
5.2.3 MULT36X36

Primitive Introduction

The 36x36 Multiplier (MULT36X36) supports 36-bit multiplication.

Architecture Overview

Figure5-5 MULT36X36 View



Port Description

Table5-9 MULT36X36 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

Table5-10 MULT36X36 Attribute Description

Attribute Name	Permitted Values	Default	Description
AREG	1’b0,1’b1	1’b0	Input A(A) register can bebypassed. ● 1’b0:bypass mode ● 1’b1:registered mode
BREG	1’b0,1’b1	1’b0	Input B(B) register can bebypassed. ● 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 ● 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
        );
```

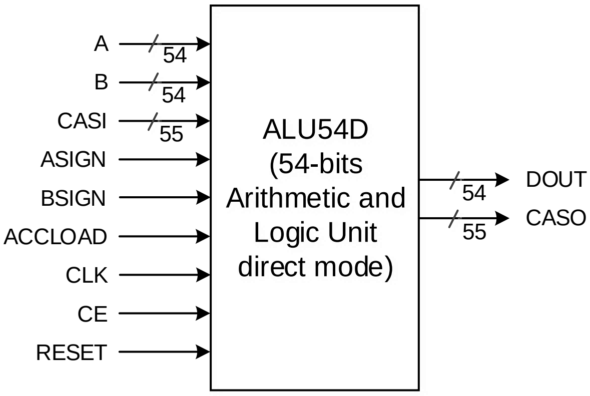
5.3 ALU54D

Primitive Introduction

The 54-bit Arithmetic Logic Unit (ALU54D) supports 54-bit arithmetic and logical operations.

Architecture Overview

Figure5-6 ALU54D Architecture



Port Description

Table5-11 ALU54D Port Description

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

Table5-12 ALU54D Attribute Description

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

The Multiplier with ALU (MULTALU) contains 36 x 18 MULTALU and 18 x 18 MULTALU, which corresponds to the primitives of MULTALU36X18 and MULTALU18X18 .

MULTALU36X18 supports three arithmetic modes:

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

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

$$DOUT = A*B + CASI$$

MULTALU18X18 supports three arithmetic modes:

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

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

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

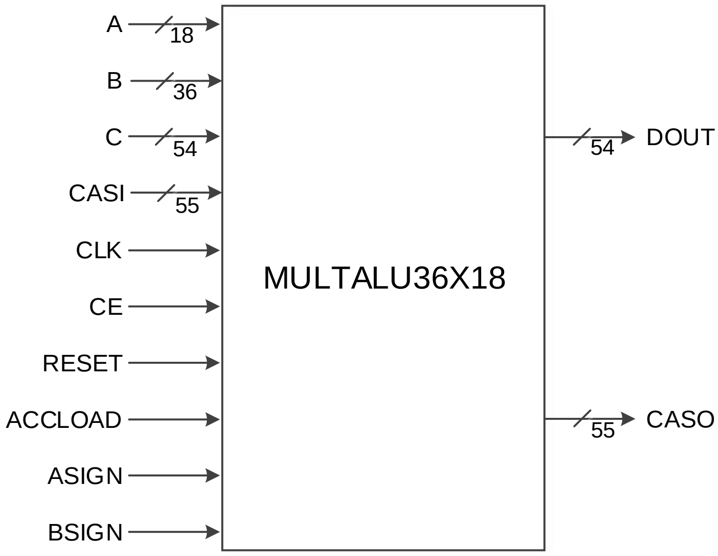
5.4.1 MULTALU36X18

Primitive Introduction

The 36x18 Multiplier with ALU (MULTALU36X18) is a 36X18 bit Multiple Accumulator.

Architecture Overview

Figure5-7 MULTALU36X18 View



Port Description

Table5-13 MULTALU36X18 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

Table5-14 MULTALU Attribute Description

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 ● 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
       );
```

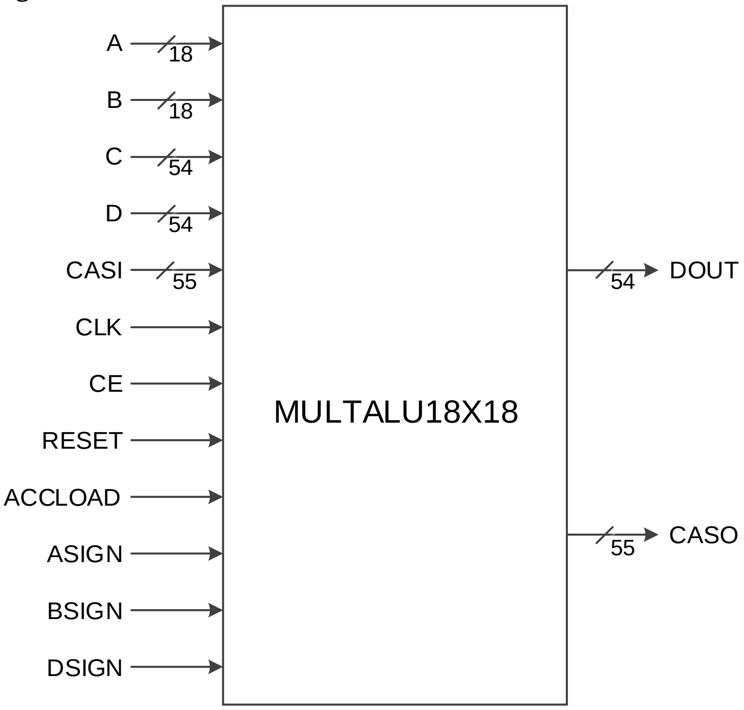
5.4.2 MULTALU18X18

Primitive Introduction

The 18x18 Multiplier with ALU (MULTALU18X18) is an 18-bit multiplicative accumulator.

Architecture Overview

Figure5-8 MULTALU18X18 View



Port Description

Table5-15 MULTALU18X18 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

Table5-16 MULTALU18X18 Attribute Description

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
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;

Attribute Name	Permitted Values	Default Value	Description
			● 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) mode can perform the MULTADD accumulation or reloading operations. The corresponding primitive is MULTADDALU18X18.

The three operation modes are as follows:

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

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

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

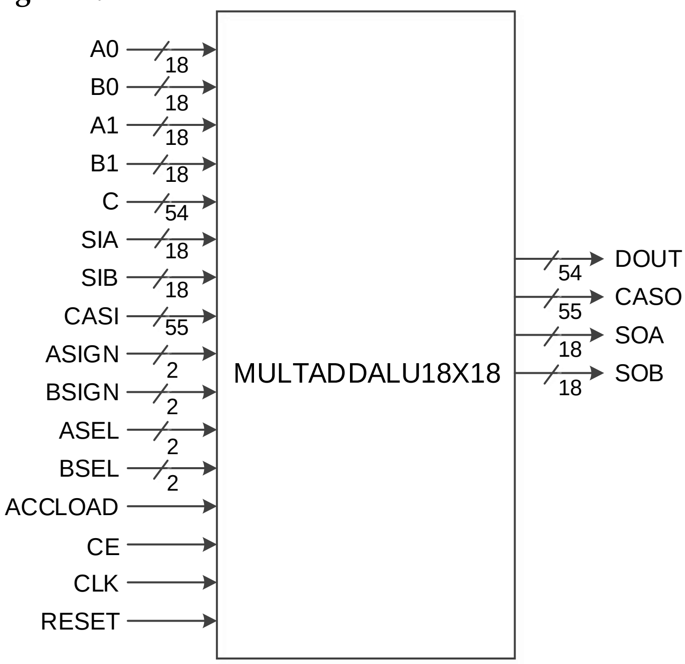
5.5.1 MULTADDALU18X18

Primitive Introduction

The Sum of Two 18x18 Multipliers with ALU (MULTADDALU18X18) supports 18-bit MULTADD accumulation or reloading.

Architecture Overview

Figure5-9 MULTADDALU18X18 View



Port Description

Table5-17 MULTADDALU18X18 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
PADDSI0[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
SOB[17:0]	Output	Multiplier Register Output B

Attribute Description

Table5-18 MULTADDALU18X18 Attribute Description

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
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

Attribute Name	Permitted Values	Default	Description
			● 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 Introduction

Gowin FPGA provides Phase Locked Loop (PLL). Based on the input clock, PLL adjusts the clock phase, duty cycle, frequency (multiplication and division) to the output different phases and frequencies.

The main features of PLL are outlined in:

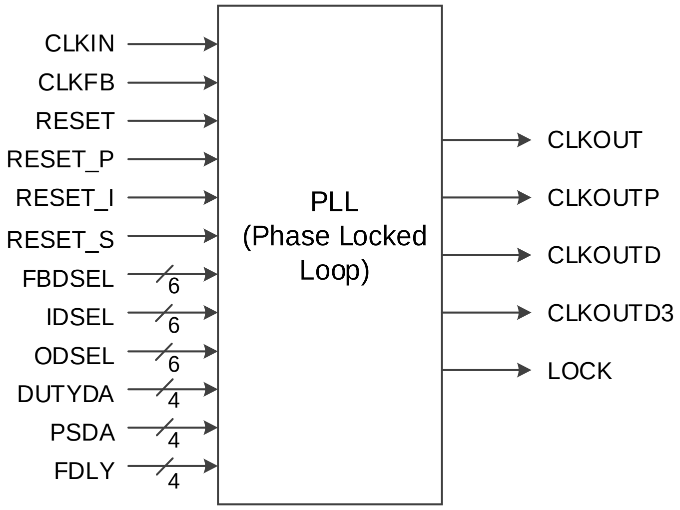
Table6-1 PLL Features

	GW1N Family	GW2A Family
Input frequency	3 MHz ~ 450MHz	3MHz ~ 500MHz
Input frequency / input frequency division multiple	3 MHz ~ 450MHz	3 MHz ~ 500MHz
VCO vibration frequency	400MHz ~ 900MHz	500MHz ~ 1300MHz
Output frequency	3.125MHz ~ 450MHz	3.125MHz ~ 500MHz

PLL supports GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure6-1 PLL View



Port Description

Table6-2 PLL Port Description

Port Name	I/O	Description
CLKIN	Input	Reference Clock Input
CLKFB	Input	Clock Feedback Input
RESET	Input	Reset Input
RESET_P	Input	Power Down Reset Input
RESET_I	Input	Reference Divider IDIV Reset Input
RESET_S	Input	Divider SDIV Reset Input
FBDSEL[5:0]	Input	Clock Frequency Multiplication Factor Dynamic Adjust
IDSEL[5:0]	Input	Clock Frequency Division Factor Dynamic Adjust
ODSEL[5:0]	Input	ODIV Frequency Division Factor Dynamic Adjust
DUTYDA[3:0]	Input	Duty Cycle Dynamic Adjust
PSDA[3:0]	Input	Phase Shift Dynamic Adjust
FDLY[3:0]	Input	Dynamic Fine Delay Adjust
CLKOUT	Output	Clock Output, No Phase Shift
LOCK	Output	PLL Lock Signal
CLKOUTP	Output	Clock Output With Phase Shift and Duty Cycle
CLKOUTD	Output	Clock Output (From CLKOUT and CLKOUTP), Divided by SDIV Divider
CLKOUTD3	Output	Clock Output (From CLKOUT and CLKOUTP), Divided by 3 Divider

Attribute Description

Table6-3 PLL Attribute Description

Attribute Name	Permitted Values	Default	Description
FCLKIN	3~500	100	Reference Clock Frequency
IDIV_SEL	0~63	0	Clock Frequency Division Factor Static Adjust
DYN_IDIV_SEL	true,false	false	Clock Frequency Division Factor Dynamic/ Static Select ● false: Static ● true: Dynamic
FBDIV_SEL	0~63	0	Clock Frequency Multiplication Factor Static Adjust
DYN_FBDIV_SEL	true,false	false	Clock Frequency Multiplication Factor Dynamic/ Static Select ● false: Static ● true: Dynamic
ODIV_SEL	2,4,8,16,32,48,64,80,96,112,128	8	ODIV Frequency Division Factor Static Adjust
DYN_ODIV_SEL	true,false	false	ODIV Frequency Division Factor Dynamic/ Static Select ● false: Static true: Dynamic
PSDA_SEL	0000~1111	0000	Phase Shift Static Adjust
DUTYDA_SEL	0010~1110	1000	Duty Cycle Static Adjust
DYN_DA_EN	true,false	false	● Phase Shift and Duty Cycle Dynamic/ Static Select ● false: Static ● true: Dynamic
CLKOUT_FT_DIR	1'b0,1'b1	1'b1	CLKOUT Fine-tuning Direction Adjust 1'b0: sub 1'b1: add
CLKOUT_DLY_STEP	0,1,2,4	0	CLKOUT Fine-tuning Factor CLKOUT_DLY_STEP*delay(delay=50ps)
CLKOUTP_FT_DIR	1'b0,1'b1	1'b1	CLKOUTP Fine-tuning Direction Adjust 1'b0: sub 1'b1: add
CLKOUTP_DLY_STEP	0,1,2	0	CLKOUTP Fine-tuning Factor CLKOUTP_DLY_STEP*delay(delay=50ps)
DYN_SDIV_SEL	2~128	2	SDIV Frequency Division Factor
CLKFB_SEL	internal,external	internal	CLKFB Source Select ● internal:Internal Feedback CLKOUT ● external: Internal Feedback Signal
CLKOUTD_SRC	CLKOUT,CLKOUTP	CLKOUT	CLKOUTD source Select
CLKOUTD3_SRC	CLKOUT,CLKOUTP	CLKOUT	CLKOUTD3 source Select
CLKOUT_BYPASS	true,false	false	● CLKIN Bypass PLL and Directly Drive the CLKOUT ● true: CLKIN Bypass to CLKOUT ● false: Normal Operation
CLKOUTP_BYPASS	true,false	false	● CLKIN Bypass PLL and Directly Drive the CLKOUTP ● true: CLKIN Bypass to CLKOUTP ● false: Normal Operation

Attribute Name	Permitted Values	Default	Description
CLKOUTD_BYPASS	true,false	false	● CLKIN Bypass PLL and Directly Drive the CLKOUTD ● true: CLKIN Bypass to CLKOUTD ● false: Normal Operation
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	● Device Select

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

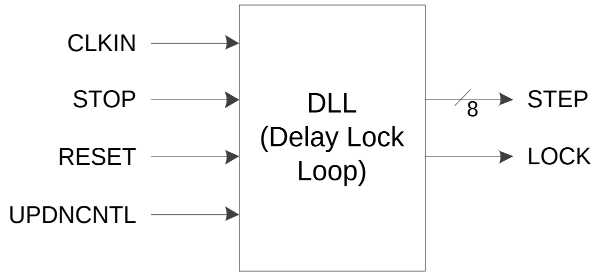
Primitive Introduction

The Delay-Locked Loop (DLL) is mainly used for clock reference delay, , which supports

The GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure6-2 DLL View



Port Description

Table6-4 DLL Port Description

Port Name	I/O	Description
STEP[7:0]	Output	Step Code Output
LOCK	Output	Lock Output
CLKIN	Input	Clock input
STOP	Input	Force DLL to stop working
RESET	Input	Reset Input
UPDNCNTL	Input	Control the Step Code Update

Attribute Description

Table6-5 DLL Attribute Description

Attribute Name	Type	Permitted Values	Default	Description
DLL_FORCE	Integer	0,1	0	Step and lock output mode can be selected ● 1:force lock and code ● 0:code/lock generated from DLL loop
CODESCAL	String	000,001,010,011,100,101, 110, 111	000	Phase offset config: ● 000:101° ● 001:112° ● 010:123° ● 011:135° ● 100:79° ● 101:68° ● 110:57° ● 111:45°
SCAL_EN	String	true,false	true	Output step mode can be selected: ● true: Phase offset correspond to parameter CODESCAL ● false:90°Phase offset
DIV_SEL	Integer	1’b0,1’b1	1’b0	Output lock mode can be selected: ● 1'b0:normal lock mode ● 1'b1:fast lock mode

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

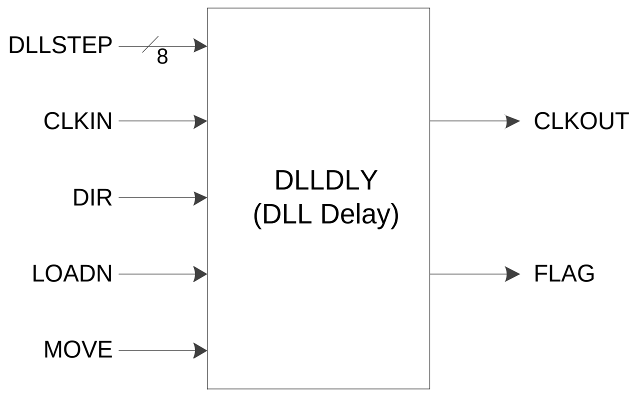
Primitive Introduction

The DLL Delay (DLLDLY) adjusts the input clock according to DLLSTEP signal and outputs the time delay of the clock. The DLLSTEP signal can come from STEP output of DLL.

DLLDLY supports the GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure6-3 DLLDLY View



Port Description

Table6-6 DLLDLY Port Description

Port Name	I/O	Description
CLKOUT	Output	Clock Output
FLAG	Output	Overflow Flag
DLLSTEP[7:0]	Input	STEP Input
CLKIN	Input	Clock input
DIR	Input	Direction can be Selected to Decide Delay, Increase or Decrease
LOADN	Input	Control Delay Code's Download
MOVE	Input	Adjust Delay Value

Attribute Description

Table6-7 DLLDLY Attribute Description

Attribute Name	Type	Permitted Values	Default	Description
DLL_INSEL	Integer	1'b0,1'b1	1'b0	DLLDLY can be bypassed ● 1'b0:bypass mode ● 1'b1: use dll_delay cell mode
DLY_SIGN	String	1'b0,1'b1	1'b0	Set symbol of delay adjustment: ● 1'b0:'+' ● 1'b1: '-'
DLY_ADJ	Integer	0~255	0	Delay adjustment: 1) dly_sign=0 dly_adj; 2) dly_sign=1 -256+dly_adj ●

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

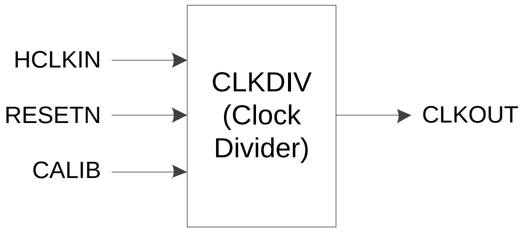
Primitive Introduction

The clock divider (CLKDIV) provides clock dynamic adjustment. GW1N-6, GW1N-9, and GW1NS-2 supports 2/3.5/4/5/8 frequency division. The other devices support 2/3.5/4/5 frequency division.

The CLKDIV supports GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure6-4 CLKDIV View



Port Description

Table6-8 CLKDIV Port Description

Port Name	I/O	Description
HCLKIN	Input	Clock input
RESETN	Input	Reset Input
CALIB:	Input	Calib Signal, adjust output clock
CLKOUT	Output	Clock Output

Attribute Description

Table6-9 CLKDIV Attribute Description

Attribute Name	Permitted Values	Default	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 parameter
GSREN	false, true	false	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

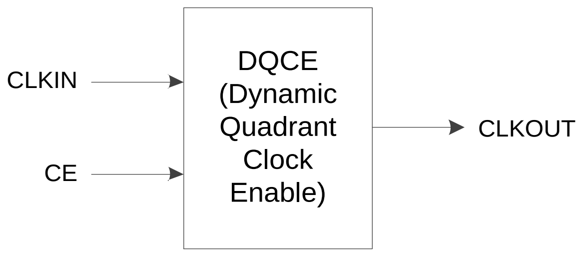
Primitive Introduction

GCLK0~GCLK5 can be dynamically turned on or off by dynamic quadrant clock enable (DQCE).

DQCE supports the GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure6-5 DQCE View



Port Description

Table6-10 DQCE Port Description

Port Name	I/O	Description
CLKIN	Input	Clock input
CE	Input	Clock Enable
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

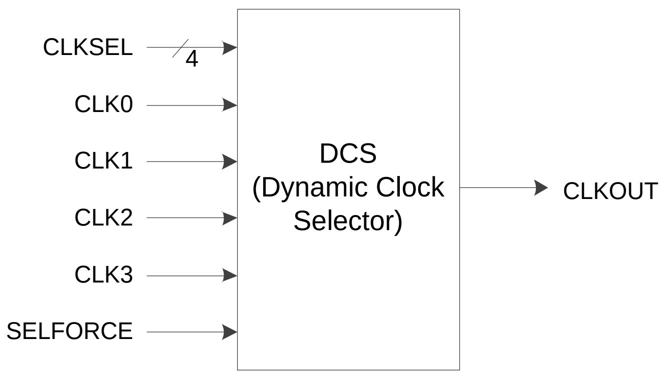
Primitive Introduction

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

DQCE supports the GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure6-6 DCS View



Port Description

Table6-11 DCS Port Description

Port Name	I/O	Description
CLK0	Input	Clock0 Input
CLK1	Input	Clock1 Input
CLK2	Input	Clock2 Input
CLK3	Input	Clock3 Input
CLKSEL[3:0]	Input	Clock Select Signal
SELFORCE	Input	Select Force Signal
CLKOUT	Output	Clock Output

Attribute Description

Table6-12 DCS Attribute Description

Attribute Name	Permitted Values	Default	Description
DCS_MODE	CLK0,CLK1,CLK2,CLK3, GND,VCC,RISING,FALLING, CLK0_GND,CLK1_GND, CLK2_GND,CLK3_GND, CLK0_VCC,CLK1_VCC, CLK2_VCC,CLK3_VCC	RISING	Set the clock selection 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

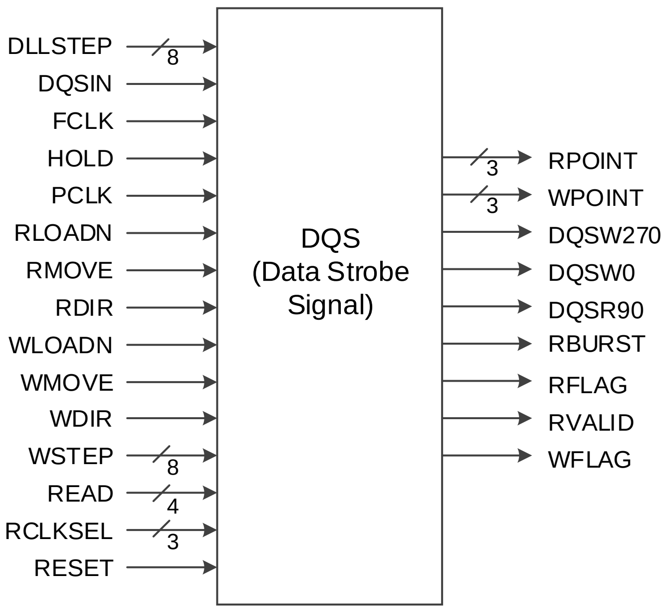
Primitive Introduction

The Bidirectional Data Strobe Circuit for DDR Memory (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.

OSER8_MEM supports the GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure6-7 DQS View



Port Description

Table6-13 DQS Port Description

Port Name	I/O	Description
DLLSTEP[7:0]	input	DQS delay control from DLL
DQSIN	input	DQS signal from PIO
FCLK	input	from 4 different ECLK tree output

Port Name	I/O	Description
HOLD	input	Stop DQSW0、DQSW270 and clear RPOINT、WPOINT
PCLK	input	from PCLK tree
RDIR	input	“0” to increase the code “1” to decrease the code for DDR read
RLOADN	input	asynchronous reset the final delay code to factory default value for DDR read
RMOVE	input	Move pulse’s rising edge will change the code according to direction value for DDR read
WDIR	input	“0” to increase the code “1” to decrease the code for DDR write
WLOADN	input	Asynchronous reset the final delay code to factory default value for DDR write
WMOVE	input	Move pulse’s rising edge will change the code according to direction value for DDR write
WSTEP[7:0]	input	DDR write leveling control also used for CDR mode
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 control
RPOINT[2:0]	output	FIFO control READ pointer (3-bits) to FIFO in PIC
WPOINT[2:0]	output	FIFO control WRITE pointer (3-bits) to FIFO in PIC
DQSW0	output	SCLK/ECLK phase shifted or delayed by 0 degree output
DQSW270	output	SCLK/ECLK phase shifted or delayed by 270 degree output
DQSR90	output	DQSI phase shifted or delayed by 90 degree output
RFLAG	output	Margin test output flag for READ to indicate the under-flow or over-flow
WFLAG	output	Margin test output flag for WRITE to indicate the under-flow or over-flow
RVALID	output	Data Valid Flag for READ mode
RBURST	output	READ burst detect output

Attribute Description

Table6-14 DQS Attribute Description

Attribute Name	Permitted Values	Default	Description
FIFO_MODE_SEL	1’b0 , 1’b1	1’b0	FIFO mode config 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	MDDR select
HWL	false,true	false	Updata0/1 time relation control
GSREN	false,true	false	Global reset can be set

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
);
```

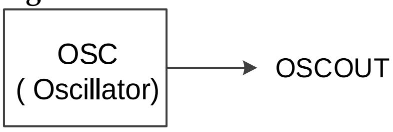
6.7 OSC

Primitive Introduction

The oscillator (OSC) supports the GW1N-2, GW1N-2B, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure 6-8 OSC View



Port Description

Table6-15 OSC Port Description

Port Name	I/O	Description
OSCOUT	output	OSC Clock Output

Attribute Description

Table6-16 OSC Attribute Description

Attribute Name	Permitted Values	Default	Description
FREQ_DIV	2~128(even)	100	Frequency Division Factor
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, and GW2A-55 devices.	GW1N-1 (GW1N series) GW2A-18 (GW2A series)	Device Select

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 Introduction

The OSCZ (Oscillator) is an on-chip crystal oscillator with dynamically shutting down the OSC, which supports the low-power function of the model.

The OSCZ supports the GW1NZ-1 device.

Architecture Overview

Figure 6-9 OSCZ View

Port Description

Table6-17 Port Port Description

Port Name	I/O	Description
OSCEN	input	OSC Enable
OSCOUT	output	OSC Clock Output

Attribute Description

Table6-18 Attribute Description

Attribute Name	Allowed Values	Default	Description
FREQ_DIV	2~128(even)	100	Frequency Division Factor

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)
                  );
```

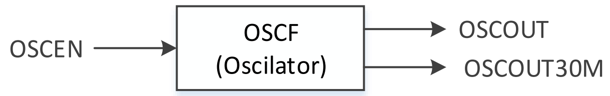
6.9 OSCF

Primitive Introduction

The Oscillator with CLKOUT30M and Dynamic OSC Enable (OSCF) supports GW1NS-2, GW1NS-2C, and GW1NSR-2C.

Architecture Overview

Figure 6-10 OSCF View



Port Description

Table6-19 OSCF Port Description

Port Name	I/O	Description
OSCEN	input	OSC Enable
OSCOUT	output	OSC Clock Output
OSCOUT30M	output	OSC Clock Output For Flash128K

Attribute Description

Table6-20 OSCF Attribute Description

Attribute Name	Permitted Values	Default	Description
FREQ_DIV	2~128(even)	96	Frequency Division Factor

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)
);
```

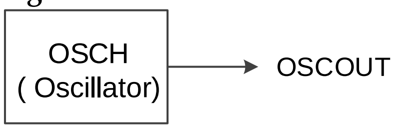
6.10 OSCH

Primitive Introduction

OSCH(Oscillator) is an on-chip crystal oscillator.
The OSCH supports GW1N-1 and GW1N-1S.

Architecture Overview

Figure 6-11 OSCH View



Port Description

Table 6-21 OSCH Port Description

Port Name	I/O	Description
OSCOUT	output	OSC Clock Output

Attribute Description

Table 6-22 OSCH Attribute Description

Port Name	I/O	Description
OSCOUT	output	OSC Clock Output

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 Introduction

The Dynamic HCLK Clock Eanble with Inverted Gate (DHCEN) can be used for HCLK stop and connecting when low level, users can dynamically turn on / off the high-speed clock signal.

The DHCEN supports the GW1N-1, GW1NZ-1, GW1N-2, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1N-2B, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure6-12 DHCEN View

Port Description

Table6-23 DHCEN Port Description

Port Name	I/O	Description
CLKIN	input	Clock input
CE	input	Clock Enable
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
    );
```

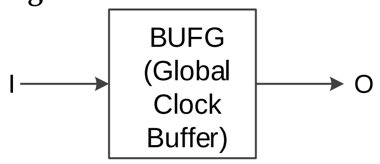
6.12 BUFG

Primitive Introduction

The BUFG (Global Clock Buffer) supports the GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure6-13 BUFG View



Port Description

Table6-24 BUFG 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
    );
```

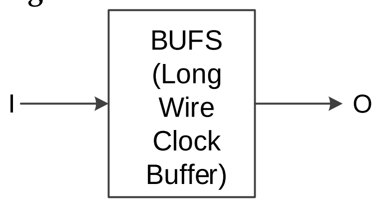
6.13 BUFS

Primitive Introduction

The Long Wire Clock Buffer (BUFS) supports the GW1N-1, GW1N-1S, GW1NZ-1, GW1N-2, GW1N-2B, GW1NS-2, GW1NS-2C, GW1NSR-2, GW1NSR-2C, GW1N-4, GW1N-4B, GW1NR-4, GW1NR-4B, GW1N-6, GW1N-9, GW1NR-9, GW2A-18, GW2AR-18, and GW2A-55 devices.

Architecture Overview

Figure6-14 BUFS View



Port Description

Table6-25 BUFS 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

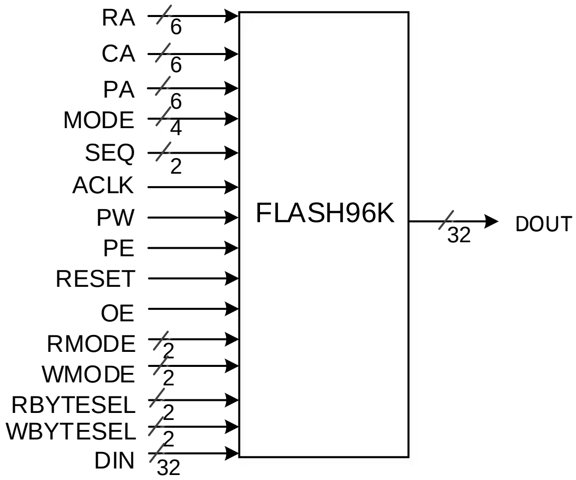
Primitive Introduction

The 96Kbit User Flash (FLASH96K) memory is 96k bits. 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 memory functions, but the initial value function of BSRAM is not included.

The FLASH96K supports the GW1N-1 and GW1N-1S device.

Architecture Overview

Figure 7-1 FLASH96K View



Port Description

Table7-1 FLASH96K 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
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 FLASH96KZ

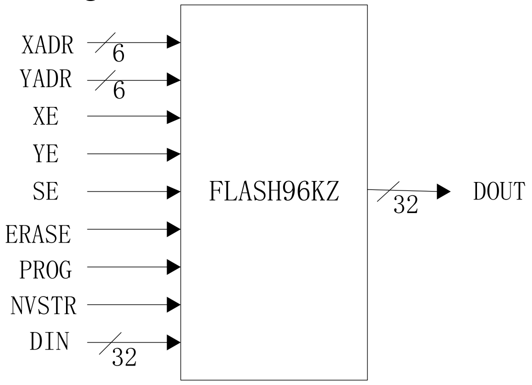
Primitive Introduction

The FLASH96KZ (96Kbit User Flash) memory is 96k bits. The width and depth of the register are constant and cannot be configured. It has non-volatile and power-off memory functions, but the initial value function of BSRAM is not included.

The FLASH96KZ supports the GW1NZ-1 device.

Architecture Overview

Figure 7-2 FLASH96KZ View



Port Description

Table7-2 Port Description

Port Name	I/O	Description
DOUT[31:0]	Output	Data Output
DIN[31:0]	Input	Data Input
XADR[5:0]	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:

```
FLASH96KZ flash96kz_inst(
        .XADR(xadr[5:0]),
        .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 FLASH96KZ
        PORT(
            XADR:IN std_logic_vector(5 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: FLASH96KZ
            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

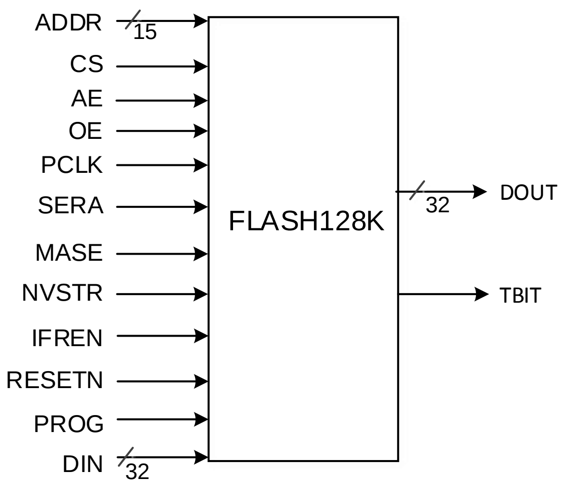
Primitive Introduction

The 128KByte Embedded Flash (FLASH128K) memory is 128K bits. The width and depth of the register are constant and cannot be configured. It has non-volatile and power-off memory functions, but the initial value function of BSRAM is not included.

The FLASH128K supports the GW1NS-2, GW1NS-2C, GW1NSR-2 and GW1NSR-2C devices.

Architecture Overview

Figure7-3 FLASH128K View



Port Description

Table7-3 FLASH128K 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

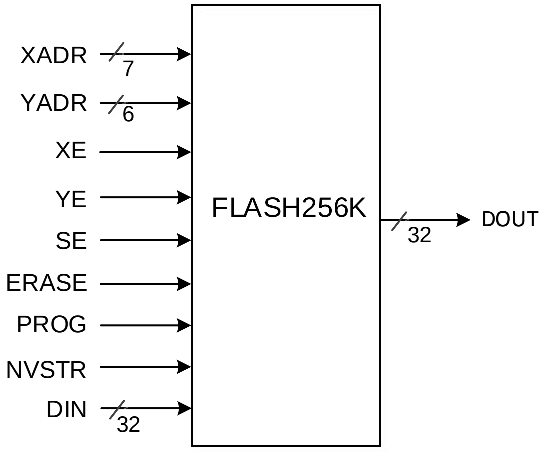
Primitive Introduction

The 256Kbit User Flash (FLASH256K) memory is 256K bits. The width and depth of the register are constant and cannot be configured. It has non-volatile and power-off memory functions, but the initial value function of BSRAM is not included.

The FLASH256K supports the devices of GW1N-2, GW1N-2B, GW1N-4, GW1N-4B, GW1NR-4 and GW1NR-4B.

Architecture Overview

Figure7-4 FLASH256K View



Port Description

Table7-4 FLASH256K 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

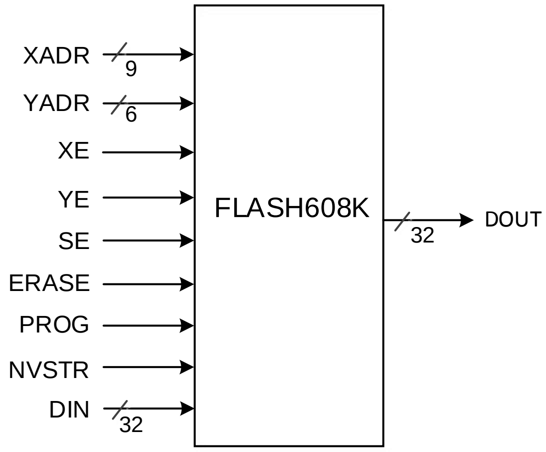
Primitive Introduction

The 608Kbit User Flash FLASH608K memory is 608K bits. The width and depth of the register are constant and cannot be configured. It has non-volatile and power-off memory functions, but the initial value function of BSRAM is not included.

The FLASH608K supports the GW1N-6, GW1N-9, and GW1NR-9 devices.

Architecture Overview

Figure7-5 FLASH608K View



Port Description

Table7-5 FLASH608K 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

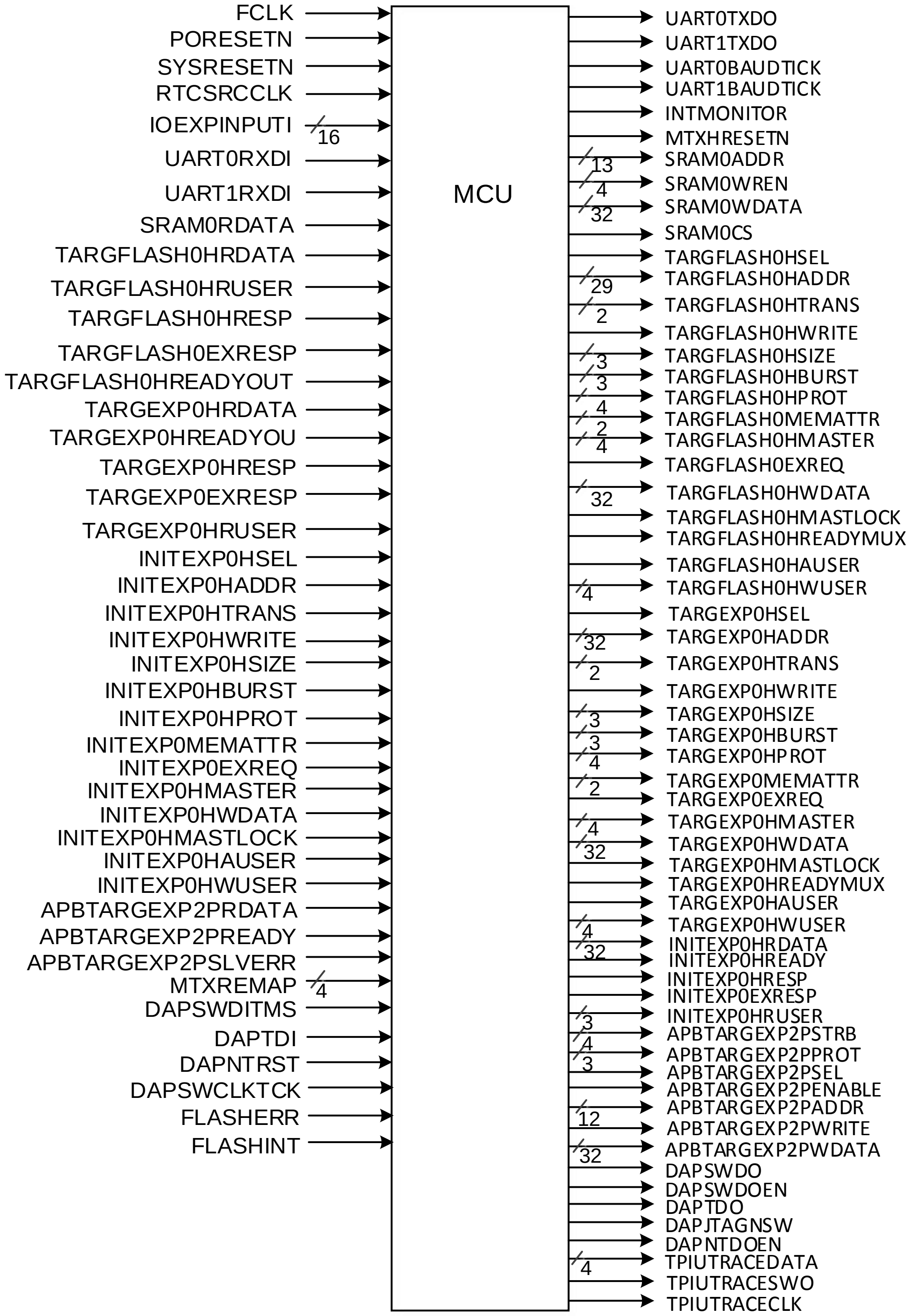
Primitive Introduction

The ARM Cortex-M3 Microcontroller Unit (MCU) is an embedded 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.

The MCU supports the GW1NS-2C and GW1NSR-2C device.

Architecture Overview

Figure8-1 MCU View



Port Description

Table8-1 MCU Port Description

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

Port Name	I/O	Description
IOEXPOUTPUTENO[15:0]	output	IOEXPOUTPUTENO
UART0TXDO	output	UART0TXDO
UART1TXDO	output	UART1TXDO
UART0BAUDTICK	output	UART0BAUDTICK
UART1BAUDTICK	output	UART1BAUDTICK
INTMONITOR	output	INTMONITOR
MTXHRESETN	output	SRAM/Flash Chip reset
SRAM0ADDR[12:0]	output	SRAM address
SRAM0WREN[3:0]	output	SRAM Byte write enable
SRAM0WDATA[31:0]	output	SRAM Write data
SRAM0CS	output	SRAM Chip select
TARGFLASH0HSEL	output	TARGFLASH0, HSELx
TARGFLASH0HADDR[28:0]	output	TARGFLASH0, HADDR
TARGFLASH0HTRANS[1:0]	output	TARGFLASH0, HTRANS
TARGFLASH0HWRITE	output	TARGFLASH0, HWRITE
TARGFLASH0HSIZE[2:0]	output	TARGFLASH0, HSIZE
TARGFLASH0HBURST[2:0]	output	TARGFLASH0, HBURST
TARGFLASH0HPROT[3:0]	output	TARGFLASH0, HPROT
TARGFLASH0MEMATTR[1:0]	output	TARGFLASH0, MEMATTR
TARGFLASH0EXREQ	output	TARGFLASH0, EXREQ
TARGFLASH0HMASTER[3:0]	output	TARGFLASH0, HMASTER
TARGFLASH0HWDATA[31:0]	output	TARGFLASH0, HWDATA
TARGFLASH0HMASTLOCK	output	TARGFLASH0, HMASTLOCK
TARGFLASH0HREADYMUX	output	TARGFLASH0, HREADYOUT
TARGFLASH0HAUSER	output	TARGFLASH0, HAUSER
TARGFLASH0HWUSER[3:0]	output	TARGFLASH0, HWUSER
TARGEXP0HSEL	output	TARGEXP0, HSELx
TARGEXP0HADDR[31:0]	output	TARGEXP0, HADDR
TARGEXP0HTRANS[1:0]	output	TARGEXP0, HTRANS
TARGEXP0HWRITE	output	TARGEXP0, HWRITE
TARGEXP0HSIZE[2:0]	output	TARGEXP0, HSIZE
TARGEXP0HBURST[2:0]	output	TARGEXP0, HBURST
TARGEXP0HPROT[3:0]	output	TARGEXP0, HPROT
TARGEXP0MEMATTR[1:0]	output	TARGEXP0, MEMATTR
TARGEXP0EXREQ	output	TARGEXP0, EXREQ
TARGEXP0HMASTER[3:0]	output	TARGEXP0, HMASTER
TARGEXP0HWDATA[31:0]	output	TARGEXP0, HWDATA
TARGEXP0HMASTLOCK	output	TARGEXP0, HMASTLOCK
TARGEXP0HREADYMUX	output	TARGEXP0, HREADYOUT
TARGEXP0HAUSER	output	TARGEXP0, HAUSER
TARGEXP0HWUSER[3:0]	output	TARGEXP0, HWUSER
INITEXP0HRDATA[31:0]	output	INITEXP0, HRDATA
INITEXP0HREADY	output	INITEXP0, HREADY
INITEXP0HRESP	output	INITEXP0, HRESP
INITEXP0EXRESP	output	INITEXP0,EXRESP
INITEXP0HRUSER[2:0]	output	INITEXP0, HRUSER

Port Name	I/O	Description
APBTARGEXP2PSTRB[3:0]	output	APBTARGEXP2, PSTRB
APBTARGEXP2PPROT[2:0]	output	APBTARGEXP2, PPROT
APBTARGEXP2PSEL	output	APBTARGEXP2, PSELx
APBTARGEXP2PENABLE	output	APBTARGEXP2, PENABLE
APBTARGEXP2PADDR[11:0]	output	APBTARGEXP2, PADDR
APBTARGEXP2PWRITE	output	APBTARGEXP2, PWRITE
APBTARGEXP2PWDATA[31:0]	output	APBTARGEXP2, PWDATA
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. A system might not connect all the bits of
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(targflash0memattr[1:0]),
   .TARGFLASH0EXREQ(targflash0exreq),
```

```
.TARGFLASH0HMASTER(targflash0hmaster[3:0]),
.TARGFLASH0HWDATA(targflash0hwdata[31:0]),
.TARGFLASH0HMASTLOCK(targflash0hmastlock),
.TARGFLASH0HREADYMUX(targflash0hreadymux),
.TARGFLASH0HAUSER(targflash0hauser),
.TARGFLASH0HWUSER(targflash0hwuser[3:0]),
.TARGFLASH0HRDATA(targflash0hrdata[31:0]),
.TARGFLASH0HRUSER(targflash0hruser[2:0]),
.TARGFLASH0HRESP(targflash0hresp),
.TARGFLASH0EXRESP(targflash0exresp),
.TARGFLASH0HREADYOUT(targflash0hreadyout),
.TARGEXP0HSEL(targexp0hsel),
.TARGEXP0HADDR(targexp0haddr[31:0]),
.TARGEXP0HTRANS(targexp0htrans[1:0]),
.TARGEXP0HWRITE(targexp0hwrite),
.TARGEXP0HSIZE(targexp0hsize[2:0]),
.TARGEXP0HBURST(targexp0hburst[2:0]),
.TARGEXP0HPROT(targexp0hprot[3:0]),
.TARGEXP0MEMATTR(targexp0memattr[1:0]),
.TARGEXP0EXREQ(targexp0exreq),
.TARGEXP0HMASTER(targexp0hmaster[3:0]),
.TARGEXP0HWDATA(targexp0hwdata[31:0]),
.TARGEXP0HMASTLOCK(targexp0hmastlock),
.TARGEXP0HREADYMUX(targexp0hreadymux),
.TARGEXP0HAUSER(targexp0hauser),
.TARGEXP0HWUSER(targexp0hwuser[3:0]),
.TARGEXP0HRDATA(targexp0hrdata[31:0]),
.TARGEXP0HREADYOUT(targexp0hreadyout),
.TARGEXP0HRESP(targexp0hresp),
.TARGEXP0EXRESP(targexp0exresp),
.TARGEXP0HRUSER(targexp0hruser[2:0]),
.INITEXP0HSEL(initexp0hsel),
.INITEXP0HADDR(initexp0haddr[31:0]),
.INITEXP0HTRANS(initexp0htrans[1:0]),
.INITEXP0HWRITE(initexp0hwrite),
.INITEXP0HSIZE(initexp0hsize[2:0]),
.INITEXP0HBURST(initexp0hburst[2:0]),
.INITEXP0HPROT(initexp0hprot[3:0]),
.INITEXP0MEMATTR(initexp0memattr[1:0]),
.INITEXP0EXREQ(initexp0exreq),
.INITEXP0HMASTER(initexp0hmaster[3:0]),
.INITEXP0HWDATA(initexp0hwdata[31:0]),
.INITEXP0HMASTLOCK(initexp0hmastlock),
.INITEXP0HAUSER(initexp0hauser),
.INITEXP0HWUSER(initexp0hwuser[3:0]),
.INITEXP0HRDATA(initexp0hrdata[31:0]),
.INITEXP0HREADY(initexp0hready),
.INITEXP0HRESP(initexp0hresp),
.INITEXP0EXRESP(initexp0exresp),
.INITEXP0HRUSER(initexp0hruser[2:0]),
```

```
  .APBTARGEXP2PSEL(apbtargexp2psel),
  .APBTARGEXP2PENABLE(apbtargexp2penable),
  .APBTARGEXP2PADDR(apbtargexp2paddr[11:0]),
  .APBTARGEXP2PWRITE(apbtargexp2pwrite),
  .APBTARGEXP2PWDATA(apbtargexp2pwdata[31:0]),
  .APBTARGEXP2PRDATA(apbtargexp2prdata[31:0]),
  .APBTARGEXP2PREADY(apbtargexp2pready),
  .APBTARGEXP2PSLVERR(apbtargexp2pslverr),
  .APBTARGEXP2PSTRB(apbtargexp2pstrb[3:0]),
  .APBTARGEXP2PPROT(apbtargexp2pprot[2:0]),
  .MTXREMAP(mtxremap[3:0]),
  .DAPSWDITMS(dapswditms),
  .DAPSWDO(dapswdo),
  .DAPSWDOEN(dapswdoen),
  .DAPTDI(daptdi),
  .DAPTDO(daptdo),
  .DAPNTRST(dapntrst),
  .DAPSWCLKTCK(dapswclk_tck),
  .DAPNTDOEN(dapntdoen),
  .DAPJTAGNSW(dapjtagnsw),
  .TPIUTRACEDATA(tpiutracedata[3:0]),
  .TPIUTRACESWO(tpiutraceswo),
  .TPIUTRACECLK(tpiutraceclk),
  .FLASHERR(flasherr),
  .FLASHINT(flashint)
);
```

Vhdl 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 OTG connect from the Soc (system-on-chip) to other special manufacturing processes. The USB20_PHY supports the protocol and data rate of the USB 2.0 480-mbps, and backwards compatible protocols and data rates of the USB 1.1 1.5-Mbps and 12-Mbps.

The USB20_PHY supports the GW1NS-2, GW1NS-2C, GW1NSR-2, and GW1NSR-2C devices.

Architecture Overview

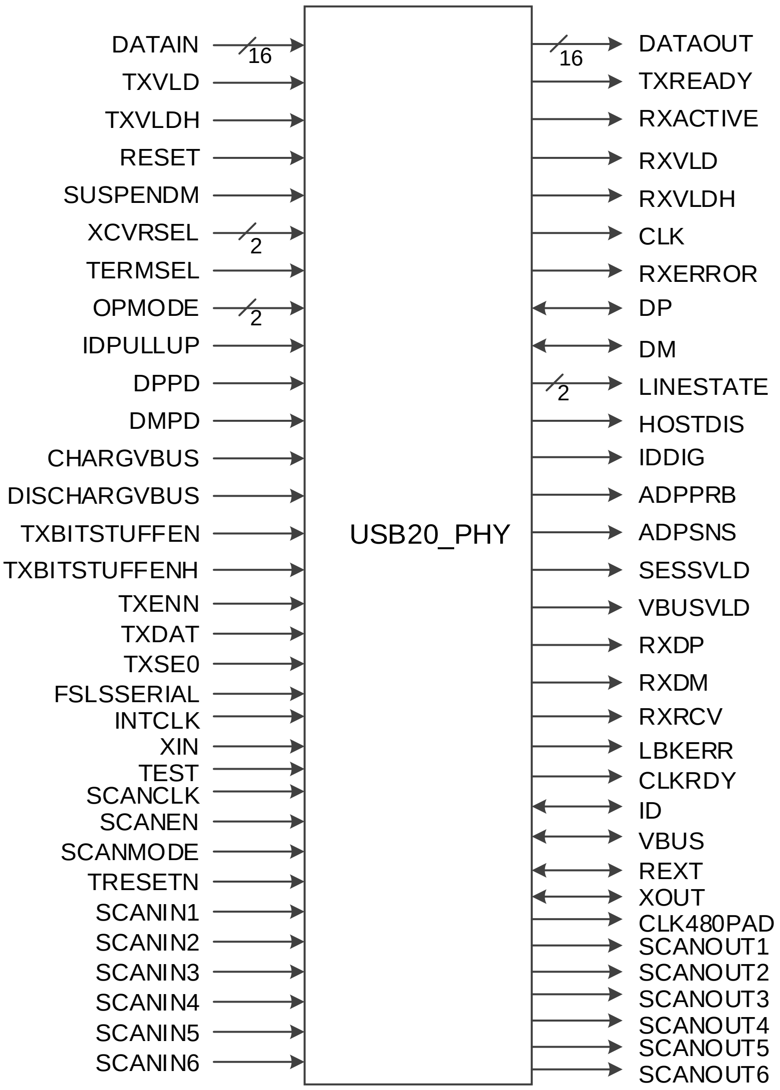


Figure8-2 USB20_PHY View

Port Description

Table8-2 USB20_PHY Port Description

Port Name	I/O	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
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.

Port Name	I/O	Description
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

Table8-3 MULTALU18X18 Attribute Description

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, XOUT. 1 to select internal clock provided on INTCLK port
M	4'b0	M divider input data bits
N	6'b101000	N divider input data bits
C	2'b01	Control charge pump current input data bits, 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,
            IDPULLUP=>idpullup,
            DPPD=>dppd,
            DMPD=>dmpd,
            CHARGVBUS=>chargvbus,
            DISCHARGVBUS=>dischargvbus,
            TXBITSTUFFEN=>txbitstuffen,
            TXBITSTUFFENH=>txbitstuffenh,
            TXENN=>txenn,
            TXDAT=>txdat,
            TXSE0=>txse0,
            FSLSSERIAL=>fslsserial,
            HOSTDIS=>hostdis,
            IDDIG=>iddig,
            ADPPRB=>adpprb,
            ADPSNS=>adpsns,
            SESSVLD=>sessvld,
            VBUSVLD=>vbusvld,
            RXDP=>rxdp,
            RXDM=>rxdm,
            RXRCV=>rxrcv,
            LBKERR=>lbkerr,
            CLKRDY=>clkrdy,
            INTCLK=>intclk,
            ID=>id,
            VBUS=>vbus,
            REXT=>rext,
            XIN=>xin,
            XOUT=>xout,
            TEST=>test,
            CLK480PAD=>clk480pad,
            SCANCLK=>scanclk,
            SCANEN=>scanen,
            SCANMODE=>scanmode,
            TRESETN=>tresetn,
            SCANIN1=>scanin1,
            SCANOUT1=>scanout1,
            SCANIN2=>scanin2,
            SCANOUT2=>scanout2,
            SCANIN3=>scanin3,
            SCANOUT3=>scanout3,
            SCANIN4=>scanin4,
            SCANOUT4=>scanout4,
            SCANIN5=>scanin5,
            SCANOUT5=>scanout5,
            SCANIN6=>scanin6,
            SCANOUT6=>scanout6
        );
```

8.3 ADC

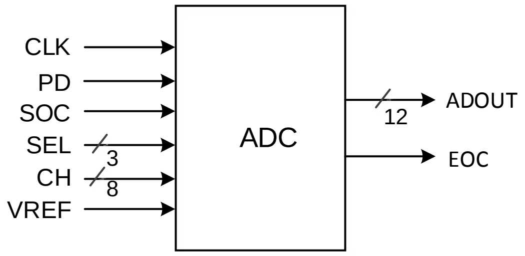
Primitive Introduction

The Analog-to-digital Converter (ADC) is an 8-channel single-end 12-bit AD converter, with the features of low power consumption, low leakage and high-dynamic.

The ADC supports the GW1NS-2, GW1NS-2C, GW1NSR-2 and GW1NSR-2C devices.

Architecture Overview

Figure8-3 ADC View



Port Description

Table8-4 ADC 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

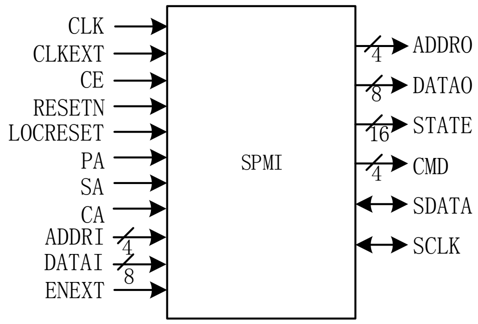
Primitive Introduction

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

SPMI (System Power Managemnent Interface) supports GW1NZ-1device.

Architecture Overview

Figure 9-1 SPMI View



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
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

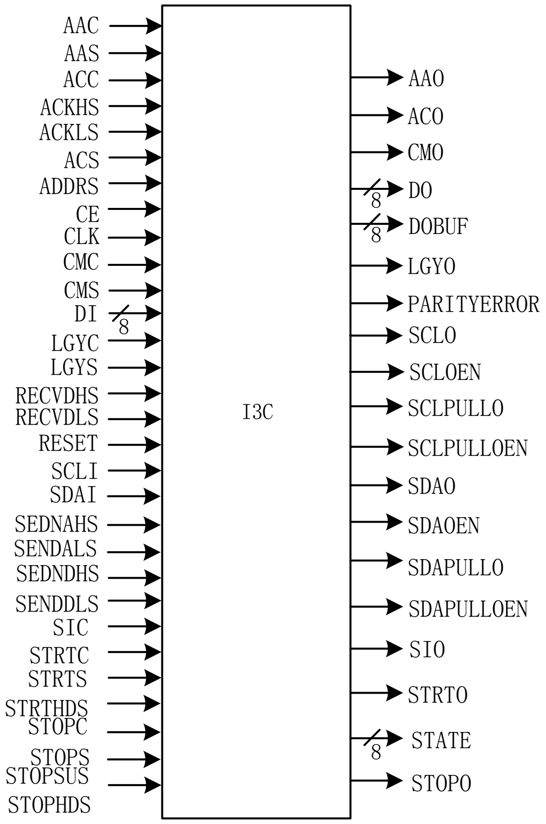
Primitive Introduction

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

I3C (Improved Inter Integrated Circuit) supports GW1NZ-1 device.

Architecture Overview

Figure 9-2 I3C View



Port Description

Table 9-2 I3C 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
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

Port Name	I/O	Description
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)
    );
```

GOWIN
PROGRAMMING FOR FUTURE