Gowin BSRAM & SSRAM

User Guide

UG285-1.3.6E, 05/25/2023

Disclaimer

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

Revision History

Date	Version	Description
05/31/2018	1.09E	Initial version published.
08/17/2020	1.2E	The chapter structure modified and content optimized.
06/21/2021	1.3E	Some figures updated and "Help" information removed of chapter 6 IP Generation.
10/12/2021	1.3.1E	The description of RESET updated.
07/22/2022	1.3.2E	The disable mode and related devices of BSRAM updated.
08/10/2022	1.3.3E	The device information of SSRAM primitive updated.
11/01/2022	1.3.4E	GW1NS-2 removed.
01/05/2023	1.3.5E	The configuration box "File" modified to "General" and "Device Version" option added on the IP interface.
05/25/2023	1.3.6E	● Read-before-write in dual port mode is not supported. ● A note added in the functional description of dual port mode and semi dual port mode respectively.

Contents

List of Figures

List of Tables

1 About This Guide

1.1 Purpose

This manual describes the features, operating modes, primitives and IP generation of Gowin BSRAM & SSRAM to help you better understand Gowin products.

1.2 Related Documents

The latest user guides are available on GOWINSEMI Website. You can find the related documents at www.gowinsemi.com:

- DS100, GW1N series of Products Data Sheet
- DS117, GW1NR series of FPGA Products Data Sheet
- DS102, GW2A series of FPGA Products Data Sheet
- DS226, GW2AR series of FPGA Products Data Sheet
- SUG100, Gowin Software User Guide

1.3 Abbreviations and Terminology

The abbreviations and terminology used in this manual are shown in Table 1-1.

Table 1-1 Abbreviations and Terminology

Abbreviations and Terminology	Name
BSRAM	Block Static Random Access Memory
CFU	Configurable Function Unit
CST	Constraints
DP	True Dual Port 16K Block SRAM
ROM	Read-only Memory
SDP	Semi Dual Port 16K Block SRAM
SP	Single Port 16K Block SRAM
SSRAM	Shadow Static Random Access Memory

1.4 Support and Feedback

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

Website: www.gowinsemi.com

E-mail: support@gowinsemi.com

2 Overview

GOWINSEMI FPGA products provide abundant memories, including BSRAM and SSRAM.

Each BSRAM can be configured with up to 18 Kbits. Data width and address depth can also be configured. Each BSRAM has two ports: Port A and Port B. Either port can be a read or write port. They have independent clocks, addresses, data, and control signals, and they share the same storage memory.

Configurable Function Unit (CFU) is a basic unit that composes Gowin FPGA products. It can be configured as SSRAM, including 16 x 4 bits SRAM or ROM16.

2.1 Features

- Each BSRAM can store up to 18Kbits of data.
- Supports up to 380MHz (230MHz in read-before-write mode).
- Supports Single Port mode (SP).
- Supports Dual Port mode (DP).
- Supports Semi-dual Port mode (SDP).
- Supports Read-only-mode (ROM).
- Supports up to 36 bits data width.
- Dual Port and Semi-dual Port support independent clocks and independent data width.
- Supports pipeline mode and bypass mode.
- Supports normal mode, write-through mode, and read-before-write mode.

2.2 Configuration Mode

Each BSRAM can be configured as 16 Kbits or18 Kbits. Data width and address depth configuration is as shown in Table 2-1.

Table 2-1 Configuration Mode

Size	Single Port Mode	Dual Port Mode	Semi-dual Port Mode	Read Only Mode
16Kbits	16K x 1	16K x 1	16K x 1	16K x 1
	8K x 2	8K x 2	8K x 2	8K x 2
	4K x 4	4K x 4	4K x 4	4K x 4
	2K x 8	2K x 8	2K x 8	2K x 8
	1K x 16	1K x 16	1K x 16	1K x 16
	512 x 32	-	512 x 32	512 x 32
18Kbits	2K x 9	2K x 9	2K x 9	2K x 9
	1K x 18	1K x 18	1K x 18	1K x 18
	512 x 36	-	512 x 36	512 x 36

Each BSRAM has 14-bit address line, that is AD[13:0], and the maximum address depth is 16,384. Different data widths use different address lines, as shown in Table 2-2.

Table 2-2 BSRAM Data Width and Address Width

Size	Mode	Data Width	Address Depth	Address Width
16Kbits	16K x 1	[0:0]	16,384	[13:0]
	8K x 2	[1:0]	8,192	[13:1]
	4K x 4	[3:0]	4,096	[13:2]
	2K x 8	[7:0]	2,048	[13:3]
	1K x 16	[15:0]	1,024	[13:4]
	512 x 32	[31:0]	512	[13:5]
18Kbits	2K x 9	[8:0]	2,048	[13:3]
	1K x 18	[17:0]	1,024	[13:4]
	512 x 36	[35:0]	512	[13:5]

Dual Port and Semi-dual Port support independent read/write clocks and independent read/write data width. In Dual Port mode, the data widths supported by Port A and Port B are as shown in Table 2-3. In Semi-dual Port mode, the data widths supported by Port A and Port B are as shown in Table 2-4.

Table 2-3 Data Width in Dual Port Mode

Size	Port B	Port A						
		16K x 1	8K x 2	4K x 4	2K x 8	1K x 16	2K x 9	1K x 18
16Kbits	16K x 1	Yes	Yes	Yes	Yes	Yes	N/A	N/A
	8K x 2	Yes	Yes	Yes	Yes	Yes	N/A	N/A
	4K x 4	Yes	Yes	Yes	Yes	Yes	N/A	N/A
	2K x 8	Yes	Yes	Yes	Yes	Yes	N/A	N/A
	1K x 16	Yes	Yes	Yes	Yes	Yes	N/A	N/A
18Kbits	2K x 9	N/A	N/A	N/A	N/A	N/A	Yes	Yes
	1K x 18	N/A	N/A	N/A	N/A	N/A	Yes	Yes

Table 2-4 Data Width in Semi Dual Port Mode

Size	Port B	Port A								
		16Kx 1	8K x 2	4Kx4	2Kx8	1K x 16	512x32	2K x 9	1Kx18	512x36
16Kbits	16K x 1	Yes	Yes	Yes	Yes	Yes	Yes	N/A	N/A	N/A
	8K x 2	Yes	Yes	Yes	Yes	Yes	Yes	N/A	N/A	N/A
	4K x 4	Yes	Yes	Yes	Yes	Yes	Yes	N/A	N/A	N/A
	2K x 8	Yes	Yes	Yes	Yes	Yes	Yes	N/A	N/A	N/A
	1K x 16	Yes	Yes	Yes	Yes	Yes	Yes	N/A	N/A	N/A
	512 x 32	Yes	Yes	Yes	Yes	Yes	Yes	N/A	N/A	N/A
18Kbits	2K x 9	N/A	N/A	N/A	N/A	N/A	N/A	Yes	Yes	Yes
	1K x 18	N/A	N/A	N/A	N/A	N/A	N/A	Yes	Yes	Yes

3 BSRAM Primitives

BSRAM includes single port mode (SP/SPX9), dual port mode (DPB/DPX9B), semi-dual mode (SDPB/SDPX9B), and read-only mode (pROM/pROMX9).

Note!

- GW1N-9/GW1N-1S/GW1NR-9/GW1NS-4 series does not support dual port mode.
- For GW1N-9/GW1NR-9/GW1NS-4 series, 32/36-bit SP/SPX9 is divided into two SP/SPX9s, which occupy two BSRAMs.
- GW1NZ-1/GW1NZ-1C does not support dual port mode with 1/2/4/8/9 bit widths.
- GW1N-4D/GW1NR-4D/GW2AN-18X/GW2AN-9X does not support read-before-write in dual port mode with1/2/4/8/9 bits width.

3.1 Dual Port Mode

Primitive

DPB/DPX9B (True Dual Port 16K BSRAM/True Dual Port 18K BSRAM)

Functional Description

DPB/DPX9B works in dual port mode with its memory space of 16K bit/18K bit. Port A and port B support read/write operations respectively[1]. DPB/DPX9B supports 2 read modes (bypass mode and pipeline mode) and 2 write modes (normal mode and write-through mode).

Note!

[1] Performing read and write operations to the same address at the same time is not recommended.

- Read Mode

The output pipeline register at A port and B port are configured by READ_MODE0 and READ_MODE1. When the output pipeline register is used, the read operation needs extra delay cycle.

- Write Mode

The write mode at A and B is configured by WRITE_MODE0 and WRITE_MODE1. The modes include normal mode and write-through mode. The corresponding timing diagram of different modes is shown from Figure 3-1 to Figure 3-4.

Figure 3-1 Timing Diagram of DPB/DPX9B Normal（Bypass）

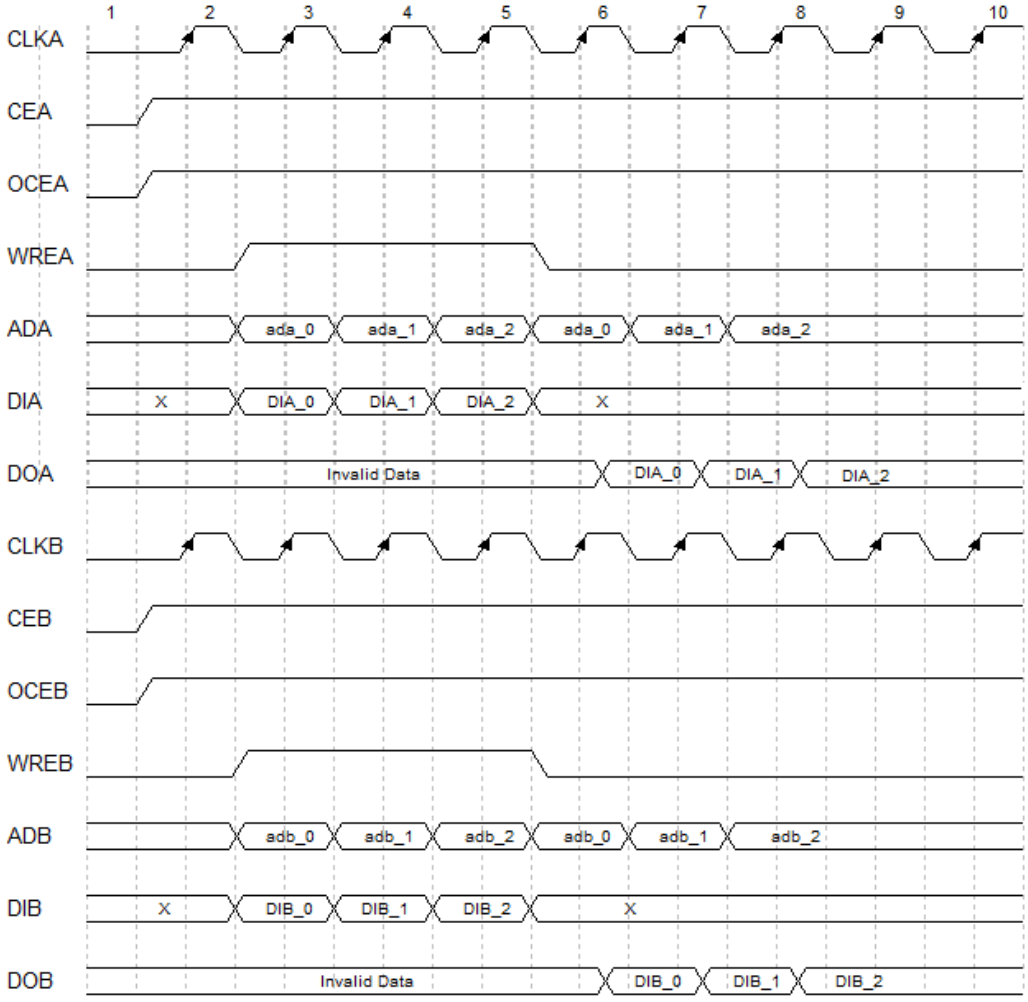


Figure 3-2 Timing Diagram of DPB/DPX9B Normal（Pipeline）

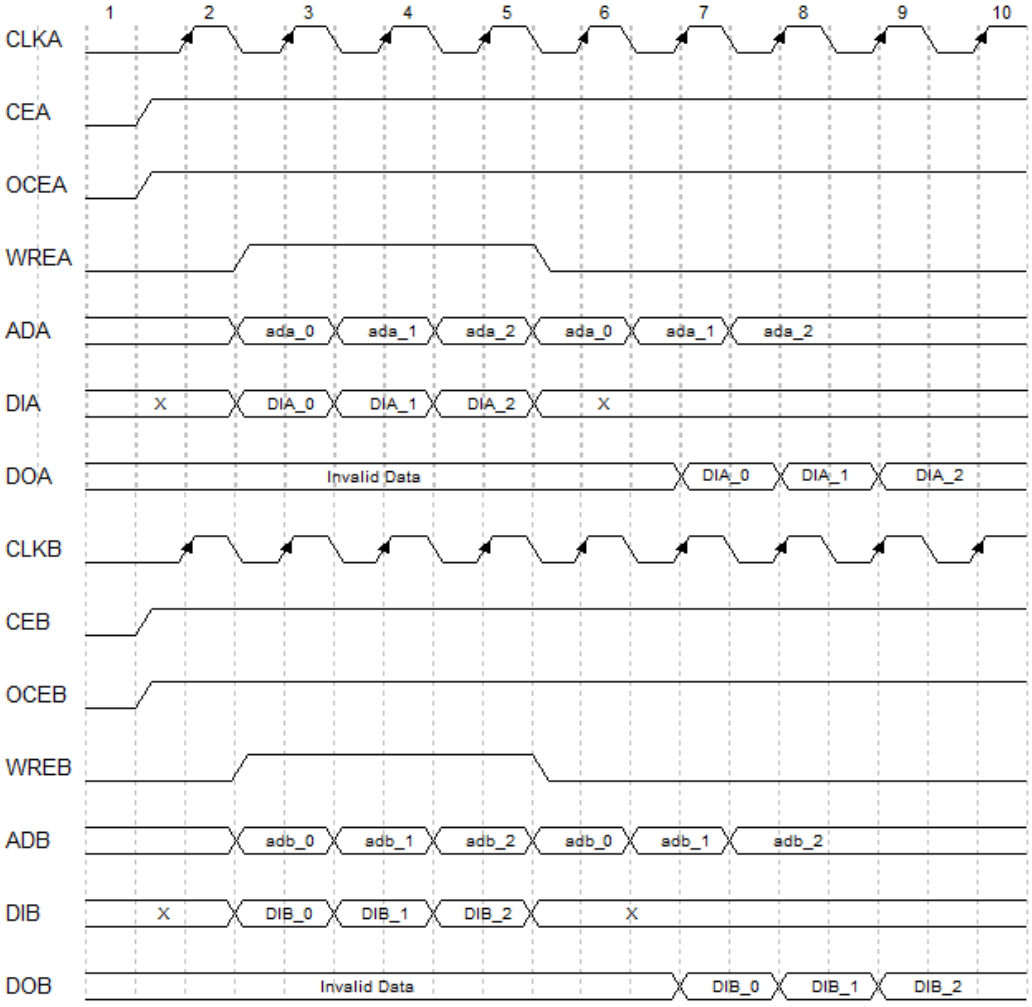


Figure 3-3 Timing Diagram of DPB/DPX9B Write-through（Bypass）

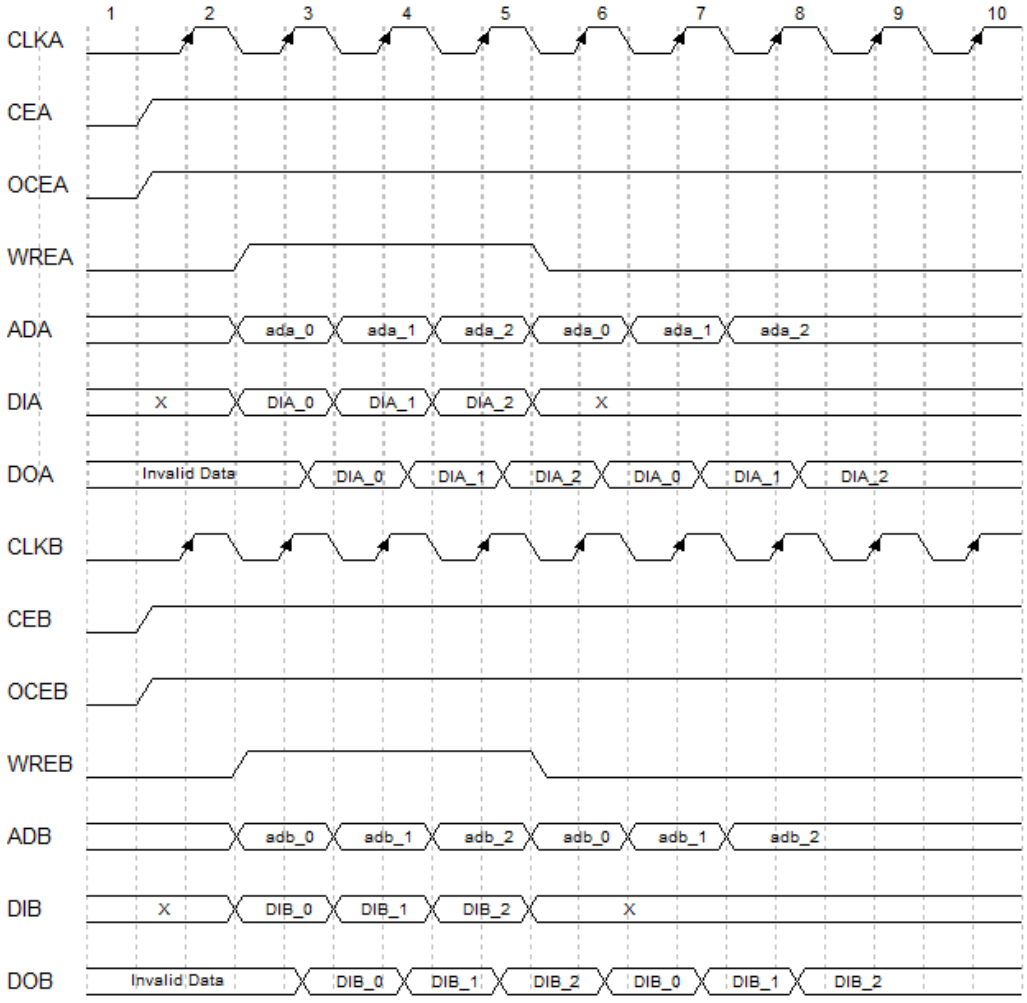
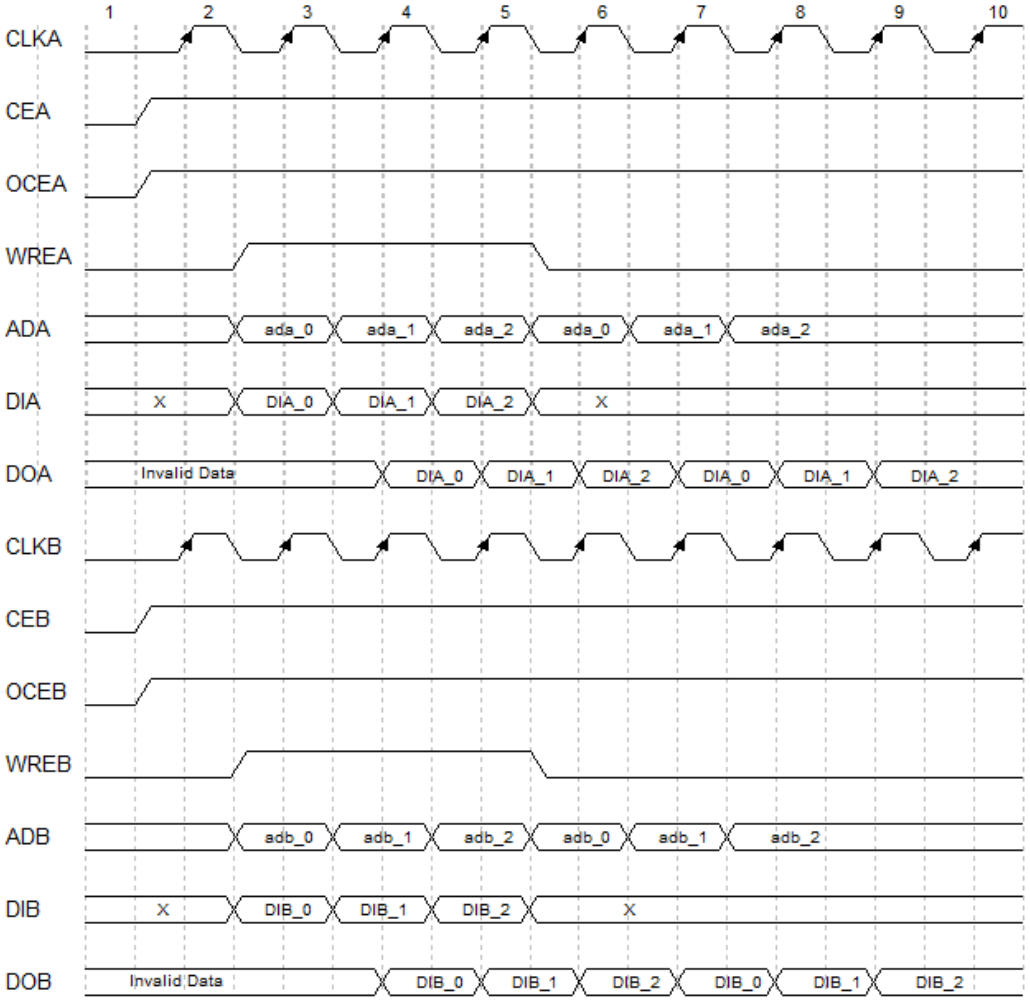


Figure 3-4 Timing Diagram of DPB/DPX9B Write-through（Pipeline）



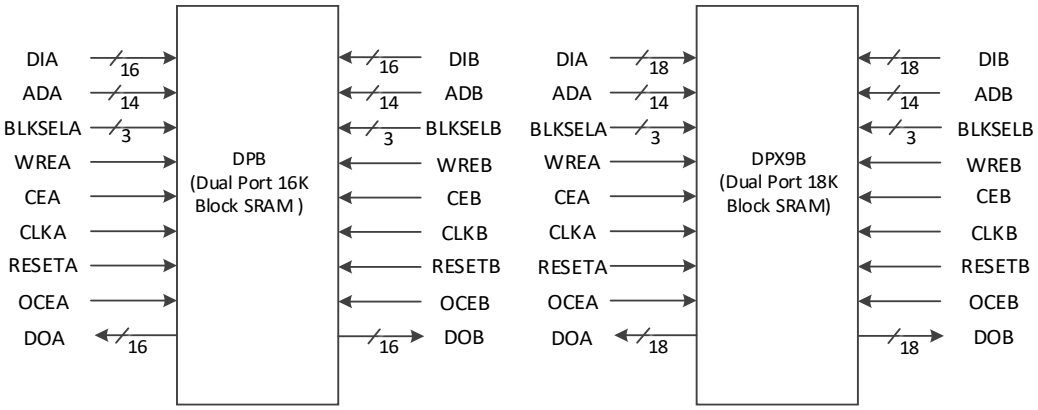
Configuration Relationship

Table 3-1 Configuration Relationship of Data Width and Address Depth

Dual Port Mode	BSRAM Capacity	Data Width	Address Depth
DPB	16K	1	14
		2	13
		4	12
		8	11
		16	10
DPX9B	18K	9	11
		18	10

Port Diagram

Figure 3-5 DPB/DPX9B Diagram



Port Description

Table 3-2 Port Description

Name	I/O	Description
DOA[15:0]/DOA[17:0]	Output	A data output
DOB[15:0]/DOB[17:0]	Output	B data output
DIA[15:0]/DIA[17:0]	Input	A data input
DIB[15:0]/DIB[17:0]	Input	B data input
ADA[13:0]	Input	A address input
ADB[13:0]	Input	B address input
WREA	Input	A write enable input 1: write 0: read
WREB	Input	B write enable input 1: write 0: read
CEA	Input	A clock enable signal, active-high
CEB	Input	B clock enable signal, active-high
CLKA	Input	A clock input
CLKB	Input	B clock input
RESETA	Input	A reset input, synchronous reset and asynchronous reset supported, active-high. It is the RESETA reset register, rather than the value of reset register.
RESETB	Input	B reset input, synchronous reset and asynchronous reset supported, active-high. It is the RESETB reset register, rather than the value of reset register.
OCEA	Input	A output clock enable signal used in Pipeline, invalid in Bypass
OCEB	Input	B output clock enable signal used in Pipeline, invalid in Bypass

Name	I/O	Description
BLKSELA[2:0]	Input	BSRAM A block selection signal for multiple BSRAM memory units cascading to realize capacity expansion
BLKSELB[2:0]	Input	BSRAM B block selection signal for multiple BSRAM memory units cascading to realize capacity expansion

Parameter

Table 3-3 Parameter Description

Name	Type	Range	Default	Description
READ_MODE0	Integer	1'b0,1'b1	1'b0	A read mode configuration 1'b0:bypass 1'b1:pipeline
READ_MODE1	Integer	1'b0,1'b1	1'b0	B read mode configuration 1'b0:bypass 1'b1:pipeline
WRITE_MODE0	Integer	2'b00,2'b01	2'b00	A write mode configuration 2'b00: normal 2'b01: write-through
WRITE_MODE1	Integer	2'b00,2'b01	2'b00	B write mode configuraion 2'b00: normal 2'b01: write-through
BIT_WIDTH_0	Integer	DPB:1,2,4,8,16 DPX9B:9,18	DPB:16 DPX9B:18	A data width configuration
BIT_WIDTH_1	Integer	DPB:1,2,4,8,16 DPX9B:9,18	DPB:16 DPB:18	B data width configuration
BLK_SEL_0	Integer	3'b000~3'b111	3'b000	When BSRAM A block selection parameter is equal to BLKSELA, the BSRAM is selected. The software will handle expansion automatically when IP Core Generator is used to expand storage capacity.
BLK_SEL_1	Integer	3'b000~3'b111	3'b000	When BSRAM B block selection

Name	Type	Range	Default	Description
				parameter is equal to BLKSELB, the BSRAM is selected. The software will handle expansion automatically when IP Core Generator is used to expand storage capacity.
RESET_MODE	String	SYNC,ASYNC	SYNC	Reset mode configuration SYNC: synchronous reset ASYNC: asynchronous reset
INIT_RAM_00~ INIT_RAM_3F	Integer	DPB:256'h0…0~256'h1…1 DPX9B:288'h0…0~288'h1…1	DPB:256'h0…0 DPX9B:288'h0…0	Used to set BSRAM initialization data

Primitive Instantiation

You can instantiate the primitives directly or generate them through IP Core Generator. For details, see chapter 6 IP Generation. Take DPB as an example to introduce the instantiation.

Verilog Instantiation：

```
DPB bram_dpb_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),
```

```
        .BLKSELA({3'b000}),
        .BLKSELB({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_dpb_0.READ_MODE0 = 1'b0;
    defparam bram_dpb_0.READ_MODE1 = 1'b0;
    defparam bram_dpb_0.WRITE_MODE0 = 2'b00;
    defparam bram_dpb_0.WRITE_MODE1 = 2'b00;
    defparam bram_dpb_0.BIT_WIDTH_0 = 8;
    defparam bram_dpb_0.BIT_WIDTH_1 = 8;
    defparam bram_dpb_0.BLK_SEL_0 = 3'b000;
    defparam bram_dpb_0.BLK_SEL_1 = 3'b000;
    defparam bram_dpb_0.RESET_MODE = "SYNC";
    defparam bram_dpb_0.INIT_RAM_00 =
256'h00A000000000000B00A000000000000B00A000000000000B00A00
0000000000B;
    defparam bram_dpb_0.INIT_RAM_3E =
256'h00A000000000000B00A000000000000B00A000000000000B00A00
0000000000B;
    defparam bram_dpb_0.INIT_RAM_3F =
256'h00A000000000000B00A000000000000B00A000000000000B00A00
0000000000B;
```

Vhdl Instantiation：

```
  COMPONENT  DPB
        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_0:bit_vector:="000";
                BLK_SEL_1:bit_vector:="000";
                RESET_MODE:string:="SYNC";
```

```
                INIT_RAM_00:bit_vector:=X"000000000000000
0000000000000000000000000000000000000000000000000";
                INIT_RAM_01:bit_vector:=X"000000000000000
0000000000000000000000000000000000000000000000000";
                INIT_RAM_3F:bit_vector:=X"000000000000000
0000000000000000000000000000000000000000000000000"
        );
        PORT (
                DOA,DOB:OUT std_logic_vector(15 downto 0):
=conv_std_logic_vector(0,16);
                CLKA,CLKB,CEA,CEB,OCEA,OCEB,RESETA,
RESETB,WREA,WREB:IN std_logic;
                ADA,ADB:IN std_logic_vector(13 downto 0);
                BLKSELA:IN std_logic_vector(2 downto 0);
                BLKSELB:IN std_logic_vector(2 downto 0);
                DIA,DIB:IN std_logic_vector(15 downto 0)
        );
    END COMPONENT;
    uut:DPB
        GENERIC MAP(
                BIT_WIDTH_0=>16,
                BIT_WIDTH_1=>16,
                READ_MODE0=>'0',
                READ_MODE1=>'0',
                WRITE_MODE0=>"00",
                WRITE_MODE1=>"00",
                BLK_SEL_0=>"000",
                BLK_SEL_1=>"000",
                RESET_MODE=>"SYNC",
    INIT_RAM_00=>X"000000000000000000000000000000000000000000
0000000000000000000000",
    INIT_RAM_01=>X"000000000000000000000000000000000000000000
0000000000000000000000",
    INIT_RAM_3F=>X"000000000000000000000000000000000000000000
0000000000000000000000"
        )
        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,
        BLKSELA=>blksela,
        BLKSELB=>blkselb,
        DIA=>dia,
        DIB=>dib
    );
```

3.2 Single Port Mode

Primitive

SP/SPX9 (Single Port 16K BSRAM/Single Port 18K BSRAM)

Functional Description

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

- Read mode

The output pipeline register is configured by READ_MODE. When the output pipeline register is used, the read operation needs extra delay cycle.

- Write mode

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

For the timing diagram of single port BSRAM different read/write modes, see the dual port BSRAM A/B ports timing diagram from Figure 3-1 to Figure 3-4, Figure 3-6 and Figure 3-7.

Figure 3-6 Timing Diagram of SP/SPX9 Read-before-write (Bypass)

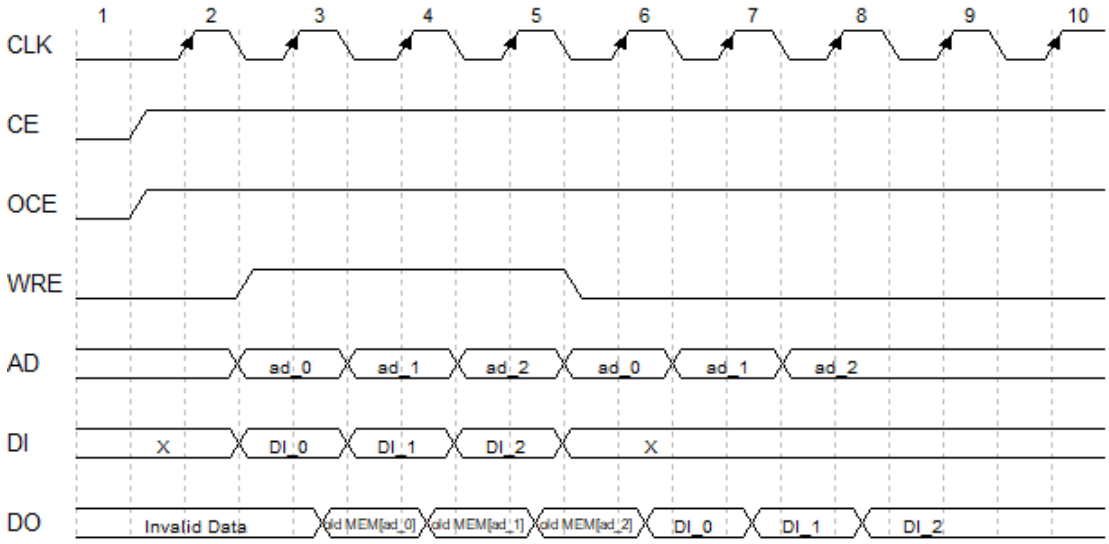
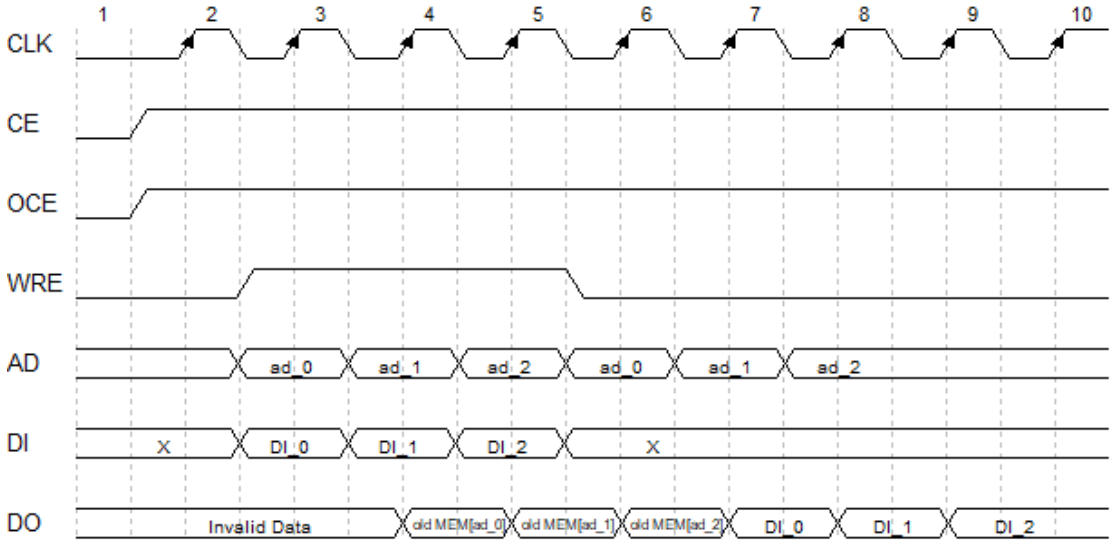


Figure 3-7 Timing Diagram of SP/SPX9 Read-before-write (Pipeline)



Configuration Relationship

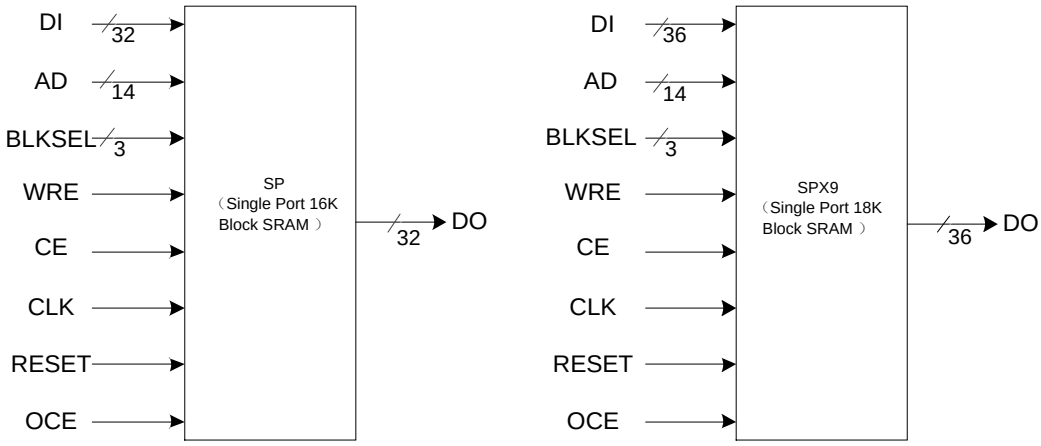
Table 3-4 Configuration Relationship of Data Width and Address Depth

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

Single Port Mode	BSRAM Capacity	Data width	Address Depth
		36	9

Port Diagram

Figure 3-8 SP/SPX9 Port Diagram



Port Description

Table 3-5 Port Description

Port Name	I/O	Description
DO[31:0]/DO[35:0]	Output	Data output
DI[31:0]/DI[35:0]	Input	Data input
AD[13:0]	Input	Address input
WRE	Input	Write enable input 1: write 0: read
CE	Input	Clock enable input, active-high.
CLK	Input	Clock input
RESET	Input	Reset input, synchronous reset and asynchronous reset supported, active-high
OCE	Input	Output clock enable signal used in Pipeline, invalid in Bypass
BLKSEL[2:0]	Input	BSRAM block selection signal for multiple BSRAM memory units cascading to realize capacity expansion

Parameter

Table 3-6 Parameter Description

Name	Type	Value	Default Value	Description
READ_MODE	Integer	1'b0,1'b1	1'b0	Read mode configuration 1'b0:bypass 1'b1:pipeline

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

Primitive Instantiation

You can instantiate the primitives directly or generate them through IP Core Generator. For details, see chapter 6 IP Generation. Take SP as an example to introduce the instantiation.

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
        );
```

3.3 Semi Dual Port Mode

Primitive

SDPB/SDPX9B (Semi Dual Port 16K BSRAM /Semi Dual Port 18K BSRAM).

Functional Description

SDPB/SDPX9B works in semi-dual port mode with its memory space of 16K bit/18K bit. Port A support write operation and port B support read operation[1]. SDPB/SDPX9B supports two read modes (bypass mode and pipeline mode) and one write mode (normal mode).

Note!

[1] Performing read and write operations to the same address at the same time is not recommended.

- Read Mode

You can configure the output pipeline register by READ_MODE. When the output pipeline register is used, the read operation needs extra delay cycle.

- Write Mode

Port A supports write operation and port B support read operation.

The timing diagram of semi-dual port different read modes is shown in Figure 3-9 and Figure 3-10.

Figure 3-9 Timing Diagram of Semi Dual Port BSRAM Normal（Bypass）

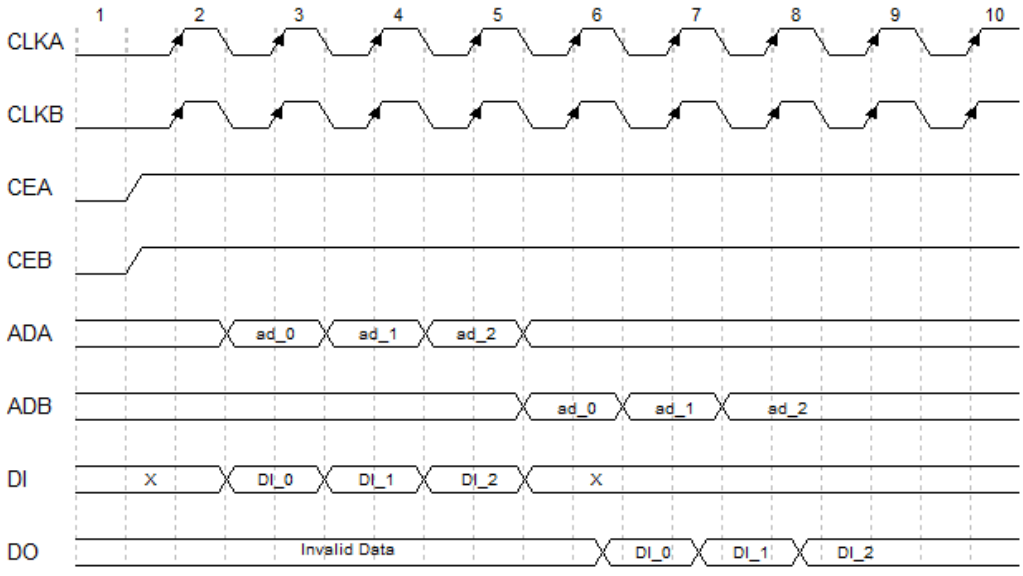
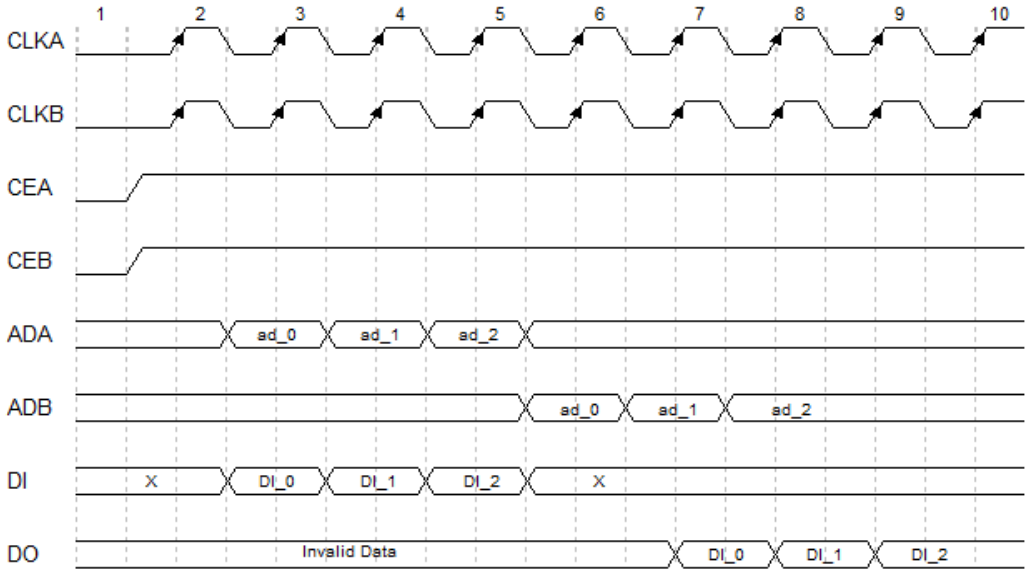


Figure 3-10 Timing Diagram of Semi Dual Port BSRAM Normal（Pipeline）



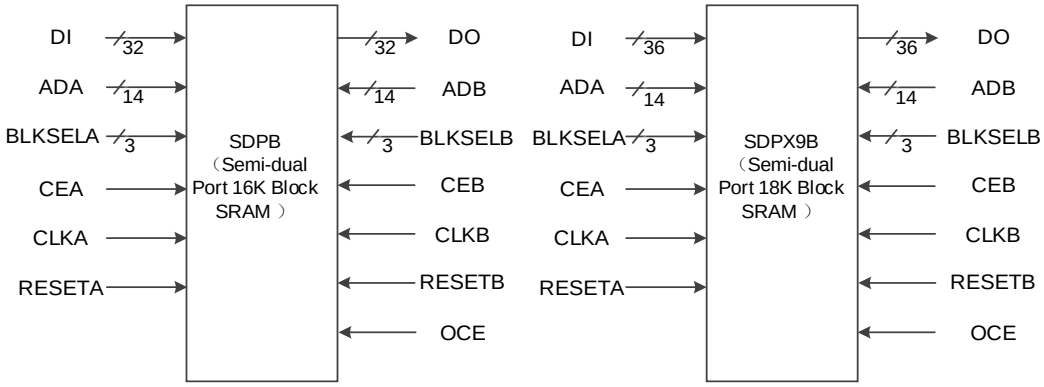
Configuration Relationship

Table 3-7 Configuration Relationship of Data Width and Address Depth

Semi-dual Port Mode	BSRAM Capacity	Data Width	Address Depth
SDPB	16K	1	14
		2	13
		4	12
		8	11
		16	10
		32	9
SDPX9B	18K	9	11
		18	10
		36	9

Port Diagram

Figure 3-11 SDPB/SDPX9B Diagram



Port Description

Table 3-8 Port Description

Name	I/O	Description
DO[31:0]/DO[35:0]	Output	Data output
DI[31:0]/DI[35:0]	Input	Data input
ADA[13:0]	Input	A address input
ADB[13:0]	Input	B address input
CEA	Input	A clock enable signal, active-high
CEB	Input	B clock enable signal, active-high
CLKA	Input	A clock input
CLKB	Input	B clock input
RESETA	Input	A reset input, synchronous reset and asynchronous reset supported, active-high.
RESETB	Input	B reset input, synchronous reset and asynchronous reset supported, active-high. It is the RESETB reset register, rather than the value of reset register.
OCE	Input	Output clock enable signal used in Pipeline, invalid in Bypass
BLKSELA[2:0]	Input	BSRAM A port block selection signal for multiple BSRAM memory units cascading to realize capacity expansion
BLKSELB[2:0]	Input	BSRAM B port block selection signal for multiple BSRAM memory units cascading to realize capacity expansion

Parameter

Table 3-9 Parameter Description

Name	Type	Range	Default	Description
READ_MODE	Integer	1'b0,1'b1	1'b0	Read mode configuartion 1'b0:bypass 1'b1:pipeline
BIT_WIDTH_0	Integer	SDPB:1,2,4,8,16,32 SDPX9B:9,18,36	SDPB:32 SDPX9B:36	A data width configuration
BIT_WIDTH_1	Integer	SDPB:1,2,4,8,16,32 SDPX9B:9,18,36	SDPB:32 SDPX9B:36	B data width configuration
BLK_SEL_0	Integer	3'b000~3'b111	3'b000	When BSRAM A block selection parameter is equal to BLKSEL, the BSRAM is selected. The software will handle

Name	Type	Range	Default	Description
				expansion automatically when IP Core Generator is used to expand storage capacity.
BLK_SEL_1	Integer	3'b000~3'b111	3'b000	When BSRAM B block selection parameter is equal to BLKSEL, the BSRAM is selected. The software will handle expansion automatically when IP Core Generator is used to expand storage capacity.
RESET_MODE	String	SYNC,ASYNC	SYNC	Reset mode configuration SYNC: synchronous reset ASYNC: asynchronous reset
INIT_RAM_00~ INIT_RAM_3F	Integer	SDPB:256'h0…0 ~256'h1…1 SDPX9B:288'h0 …0~288'h1…1	SDPB: 256'h0 …0 SDPX 9B:28 8'h0… 0	Used to set BSRAM initialization data

Primitive Instantiation

You can instantiate the primitives directly or generate them through IP Core Generator. For details, see chapter 6 IP Generation. Take SDPB as an example to introduce the instantiation.

Verilog Instantiation：

```
SDPB bram_sdpb_0 (
    .DO({dout[31:16],dout[15:0]}),
    .CLKA(clka),
    .CEA(cea),
    .RESETA(reseta),
    .CLKB(clkb),
    .CEB(ceb),
    .RESETB(resetb),
    .OCE(oce),
    .BLKSELA({3'b000}),
    .BLKSELB({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_sdpb_0.READ_MODE = 1'b1;
    defparam bram_sdpb_0.BIT_WIDTH_0 = 16;
    defparam bram_sdpb_0.BIT_WIDTH_1 = 16;
    defparam bram_sdpb_0.BLK_SEL_0 = 3'b000;
    defparam bram_sdpb_0.BLK_SEL_1 = 3'b000;
    defparam bram_sdpb_0.RESET_MODE = "SYNC";
    defparam bram_sdpb_0.INIT_RAM_00 =
    256'h00A000000000000B00A000000000000B00A000000000000B00
    A000000000000B;
    defparam bram_sdpb_0.INIT_RAM_3F =
    256'h00A000000000000B00A000000000000B00A000000000000B00
    A000000000000B;
```

Vhdl Instantiation：

```
 COMPONENT  SDPB
        GENERIC(
                BIT_WIDTH_0:integer:=16;
                BIT_WIDTH_1:integer:=16;
                READ_MODE:bit:='0';
                BLK_SEL_0:bit_vector:="000";
                BLK_SEL_1: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:IN  std_logic;
                OCE,RESETA,RESETB:IN  std_logic;
                ADA,ADB:IN  std_logic_vector(13  downto  0);
```

```
                    BLKSELA:IN std_logic_vector(2 downto 0);
                    BLKSELB:IN std_logic_vector(2 downto 0);
                    DI:IN std_logic_vector(31 downto 0)
                );
    END COMPONENT;
    uut:SDPB
        GENERIC MAP(
                    BIT_WIDTH_0=>16,
                    BIT_WIDTH_1=>16,
                    READ_MODE=>'0',
                    BLK_SEL_0=>"000",
                    BLK_SEL_1=>"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,
                CLKB=>clkb,
                CEB=>ceb,
                RESETB=>resetb,
                OCE=>oce,
                BLKSELA=>blksela,
                BLKSELB=>blkselb,
                ADA=>ada,
                DI=>din,
                ADB=>adb
            );
```

3.4 Read-only Mode

Primitive

pROM/pROMX9 (16K Block ROM /18K Block ROM)

Functional Description

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

Enable or disable output pipeline register by READ_MODE. When using the output pipeline register, the read requires an additional delay.

The timing diagram of ROM different read modes is as shown in Figure 3-12 and Figure 3-13.

Figure 3-12 Timing Diagrm of ROM（Bypass）

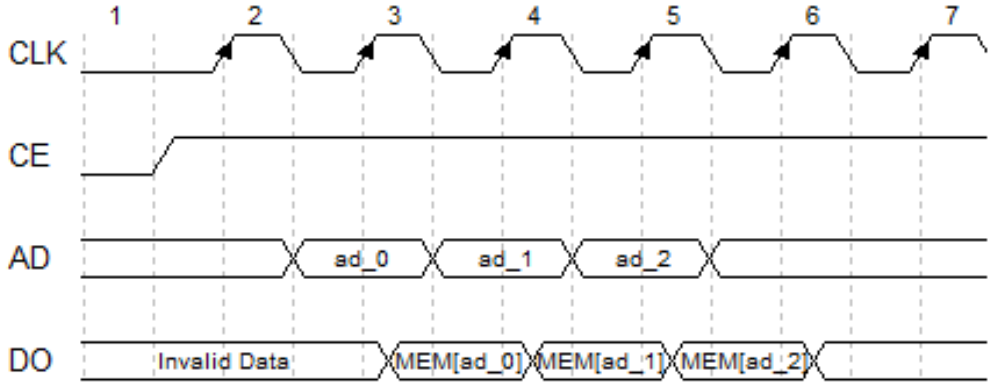
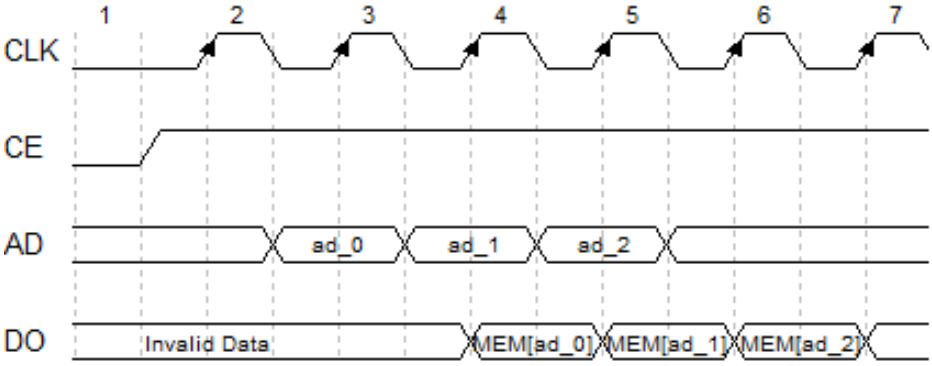


Figure 3-13 Timing Diagram of ROM（Pipeline）



Configuration Relationship

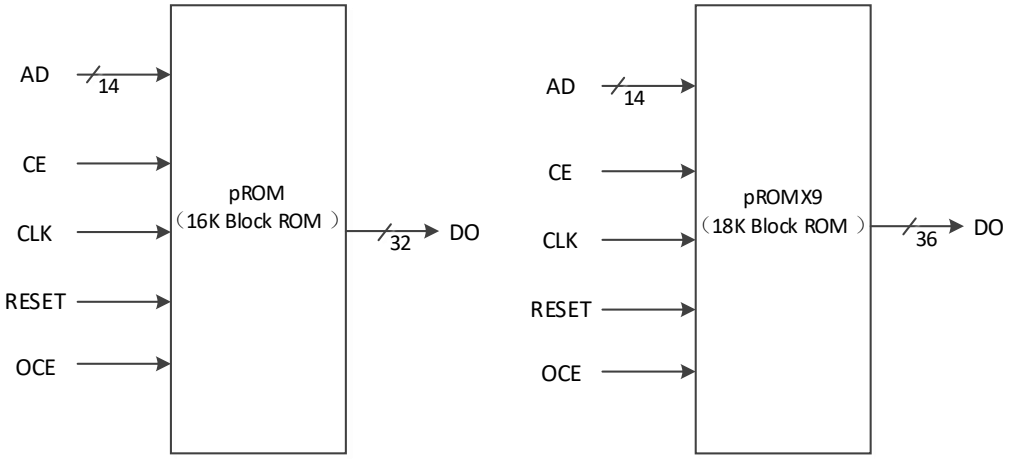
Table 3-10 Configuration Relationship of Data Width and Address Depth

Read-only Mode	BSRAM Capacity	Data Width	Address Depth
pROM	16K	1	14
		2	13
		4	12

Read-only Mode	BSRAM Capacity	Data Width	Address Depth
		8	11
		16	10
		32	9
pROMX9	18K	9	11
		18	10
		36	9

Port Diagram

Figure 3-14 pROM/pROMX9 Diagram



Port Description

Table 3-11 Port Description

Name	I/O	Description
DO[31:0]/DO[35:0]	Output	Data output
AD[13:0]	Input	Address input
CE	Input	Clock enable input, active-high.
CLK	Input	Clock input
RESET	Input	Reset input, supporting synchronous and asynchronous reset, active-high. It is the RESET reset register, rather than the value of reset register.
OCE	Input	Output clock enable signal used for Pipeline, invalid in Bypass.

Parameter

Table 3-12 Parameter Description

Name	Type	Range	Default	Description
READ_MODE	Integer	1'b0,1'b1	1'b0	Read mode configuration 1'b0:bypass

Name	Type	Range	Default	Description
				1'b1:pipeline
BIT_WIDTH	Integer	pROM:1,2,4,8,16,32 pROMX9:9,18,36	pROM:32 pROMX9:36	Data width configuration
RESET_MODE	String	SYNC,ASYNC	SYNC	Reset mode configuration SYNC: synchronous reset ASYNC: asynchronous reset
INIT_RAM_00~ INIT_RAM_3F	Integer	pROM:256'h0…0~256'h1…1 pROMX9:288'h0…0~288'h1…1	pROM:256'h0…0 pROMX9:288'h0…0	Used to set BSRAM initialization data

Primitive Instantiation

You can instantiate the primitives directly or generate them through IP Core Generator. For details, see chapter 6 IP Generation. Take pROM as an example to introduce the instantiation.

Verilog Instantiation：

```
pROM bram_prom_0 (
    .DO({dout[31:8],dout[7:0]}),
    .CLK(clk),
    .OCE(oce),
    .CE(ce),
    .RESET(reset),
    .AD({ad[10:0],3'b000})
);
defparam bram_prom_0.READ_MODE = 1'b0;
defparam bram_prom_0.BIT_WIDTH = 8;
defparam bram_prom_0.RESET_MODE = "SYNC";
defparam bram_prom_0.INIT_RAM_00 =
256'h9C23645D0F78986FFC3E36E141541B95C19F2F7164085E631A819860D8FF0000;
defparam bram_prom_0.INIT_RAM_01 =
256'h0000000000000000000000000000000000000000000000000000000FFFFFFBDCF;
```

Vhdl Instantiation：

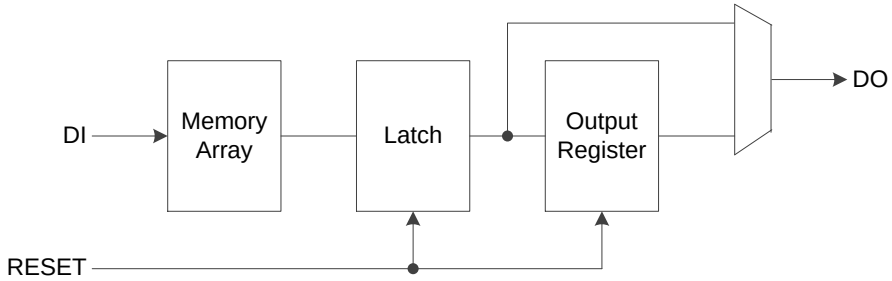
```
COMPONENT pROM
```

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

4 BSRAM Output Reset

RESET signal acts on the output module and the module outputs reset data 0. The diagram is shown in Figure 4-1.

Figure 4-1 Reset Diagram



When the RESET signal is high-level, the output port outputs 0.

RESET supports synchronous and asynchronous reset, which is set by RESET_MODE when you call the primitives library. When you use IP Core Generator, the reset mode can be selected through GUI. For the details, see chapter 6 IP Generation.

RESET signal resets the latch and the output register. When RESET signal is valid, the port outputs 0 whether you use the register output mode or the bypass output mode.

Note!

RESET signal must be set to 0 (invalid) during write operation.

The timing diagrams of synchronous and asynchronous reset modes are as shown from Figure 4-2 to Figure 4-5. DO_RAM represents the data in the storage array, and DO represents the data of the output port.

The register output mode is as follows.

- When synchronous reset is valid, DO is reset to 0 at CLK pos-edge.
- When asynchronous reset is valid, DO is reset to 0 without waiting for CLK pos-edge.
- When reset is invalid but OCE signal is valid, DO outputs DO_RAM.
- When reset and OCE signal are both invalid, DO retains the last output.

The bypass output mode is as follows:

- When synchronous reset is valid, DO is reset to 0 at CLK pos-edge.
- When asynchronous reset is valid, DO is reset to 0 without waiting for CLK pos-edge.
- When reset is invalid, DO outputs DO_RAM whether OCE signal is valid.

Figure 4-2 Timing Diagram of Synchronous Reset（Pipeline）

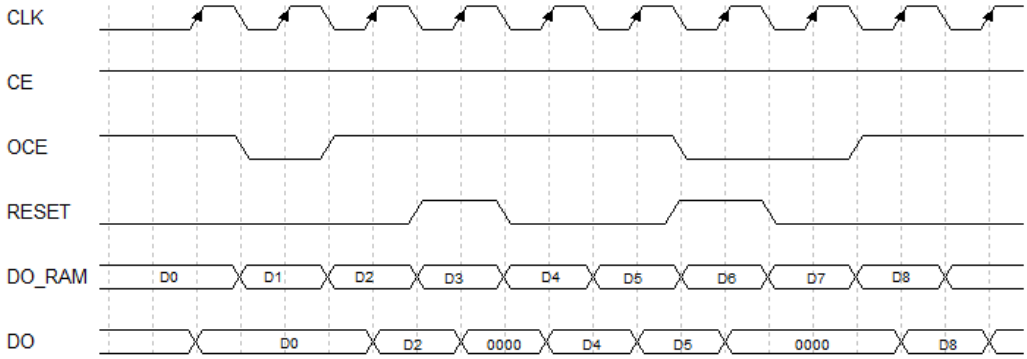


Figure 4-3 Timing Diagram of Synchronous Reset（Bypass）

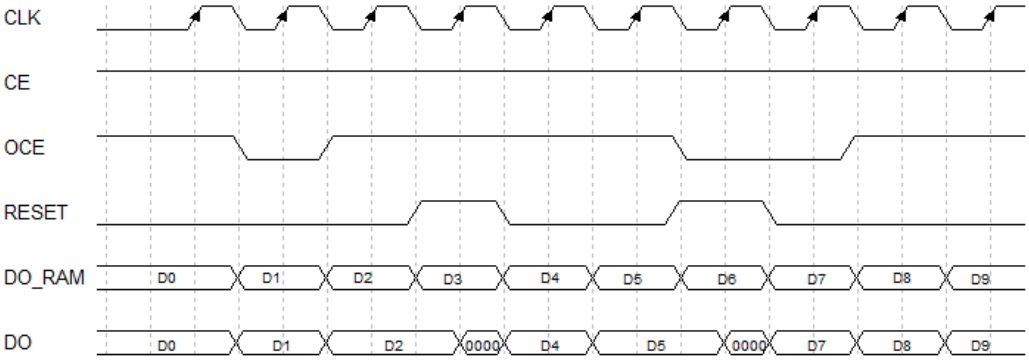


Figure 4-4 Timing Diagram of Asynchronous Reset（Pipeline）

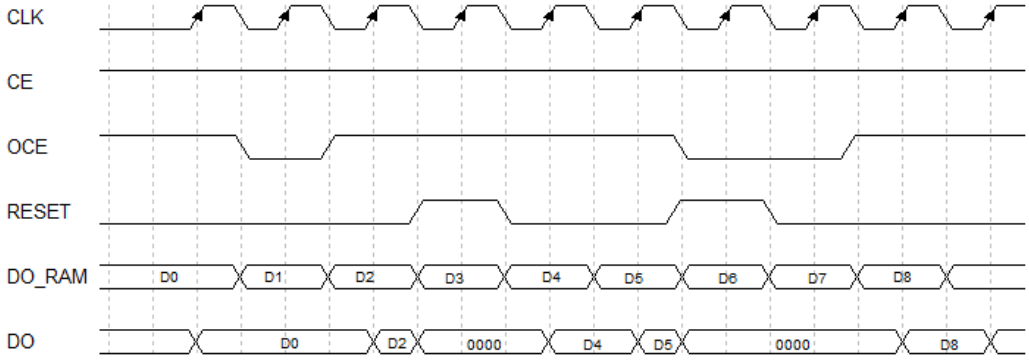
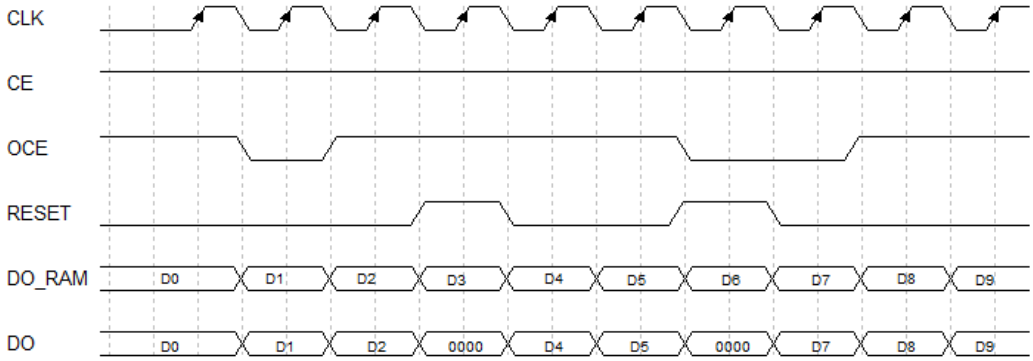


Figure 4-5 Timing Diagram of Asynchronous Reset（Bypass）



5 SSRAM Primitives

The Shadow Memory can be configured as single port mode, semi-dual port mode and read-only mode, as shown in Table 5-1.

Table 5-1 Shadow Memory Mode

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

Note!

GW1N-1, GW1N-1S, GW1N-4, GW1N-4B, GW1NR-1, GW1NR-4, GW1NR-4B, GW1NRF-4B, GW1NS-4, GW1NS-4C, GW1NSER-4C, GW1NSR-4, GW1NSR-4C, GW1N-4D and GW1NR-4D devices do not support SSRAM.

5.1 RAM16S1

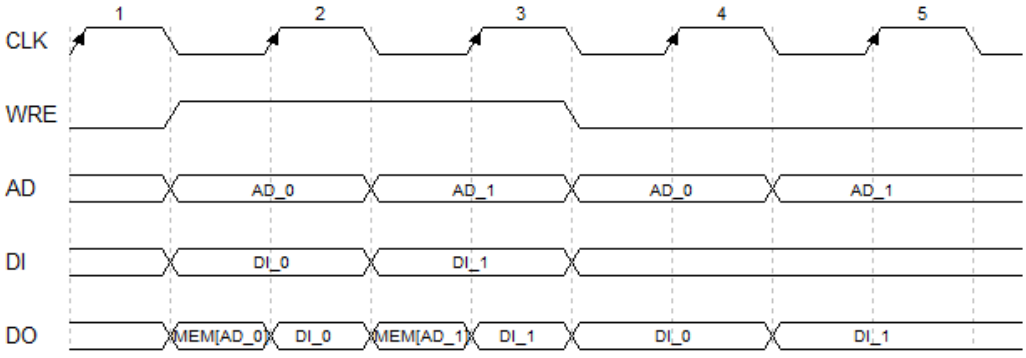
Primitive

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

Functional Description

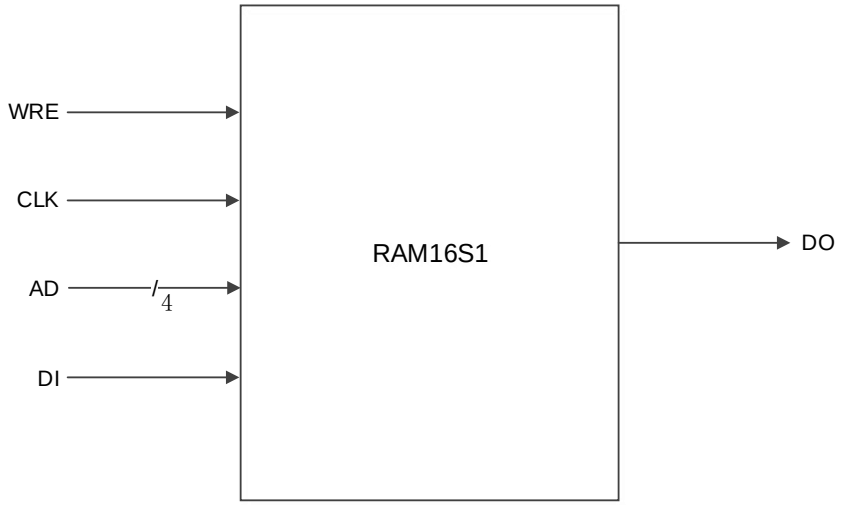
RAM16S1 is a single-port SSRAM with bit width of 1. The read and write addresses are the same. When WRE is high, you can perform write operation. At this time, the data is loaded to the address corresponding to memory at the pos-edge of CLK. The address determines the read data in RAM in read operation. That is, SSRAM is implemented by the LUT of CFU, synchronous write and asynchronous read. However, if required, a register associated with each LUT can be used for synchronous read. The timing diagram is shown in Figure 5-1.

Figure 5-1 Timing Diagram of RAM16S1



Port Diagram

Figure 5-2 RAM16S1Blcok Diagram



Port Description

Table 5-2 Port Description

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

Parameter

Table 5-3 Parameter Description

Name	Value	Default	Description
INIT_0	16'h0000~16'hffff	16'h0000	Initial value of the RAM16S1

Primitive Instantiation

You can instantiate the primitives directly or generate them through IP Core Generator. For details, see chapter 6 IP Generation.

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
    );
```

5.2 RAM16S2

Primitive

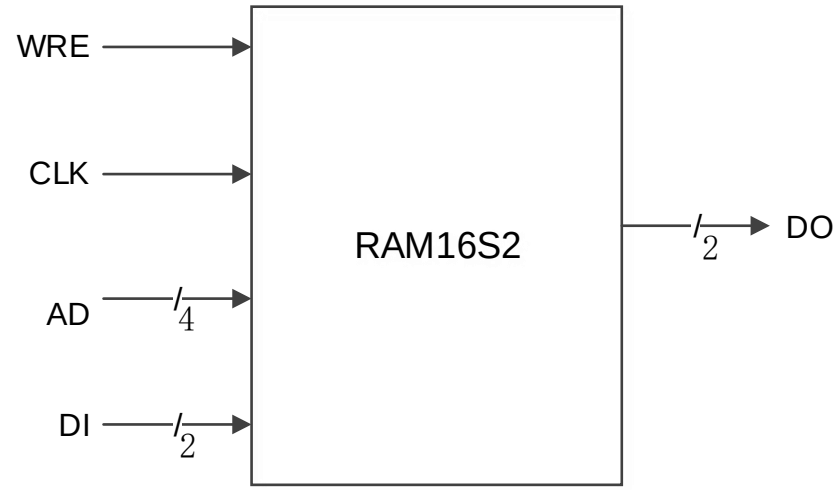
16-Deep by 2-Wide Single-port SSRAM (RAM16S2) is a single-port SSRAM with address depth of 16 and bit width of 2.

Functional Description

RAM16S2 is a single-port SSRAM with bit width of 2. The read and write addresses are the same. When WRE is high, you can perform write operation. At this time, the data is loaded to the address corresponding to memory at the pos-edge of CLK. The address determines the read data in RAM in read operation. That is, SSRAM is implemented by the LUT of CFU, synchronous write and asynchronous read. However, if required, a register associated with each LUT can be used for synchronous read. The timing diagram is shown in Figure 5-1.

Port Diagram

Figure 5-3 RAM16S2 Port Diagram



Port Description

Table 5-4 Port Description

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

Parameter

Table 5-5 Parameter Description

Name	AllowedValues	Default	Description
INIT_0~ INIT_1	16'h0000~16'hffff	16'h0000	Initial value of the RAM16S2

Primitive Instantiation

You can instantiate the primitives directly or generate them through IP Core Generator. For details, see chapter 6 IP Generation.

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
    );
```

5.2.1 RAM16S4

Primitive

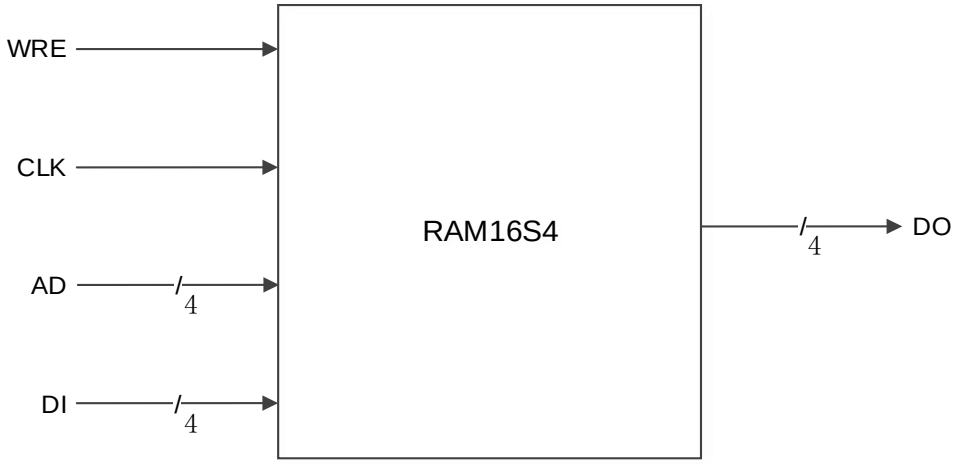
16-Deep by 4-Wide Single-port SSRAM (RAM16S4) is a single-port SSRAM with address depth of 16 and bit width of 4.

Functional Description

RAM16S4 is a single-port SSRAM with bit width of 4. The read and write addresses are the same. When WRE is high, you can perform write operation. At this time, the data is loaded to the address corresponding to memory at the pos-edge of CLK. The address determines the read data in RAM in read operation. That is, SSRAM is implemented by the LUT of CFU, synchronous write and asynchronous read. However, if required, a register associated with each LUT can be used for synchronous read. The timing diagram is shown in Figure 5-1.

Diagram

Figure 5-4 RAM16S4 Diagram



Port Description

Table 5-6 Port Description

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

Parameter

Table 5-7 Parameter Description

Name	Value	Default	Description
INIT_0~ INIT_3	16'h0000~16'hffff	16'h0000	Initial value of the RAM16S4

Primitive Instantiation

You can instantiate the primitives directly or generate them through IP Core Generator. For details, see chapter 6 IP Generation.

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
    );
```

5.3 RAM16SDP1

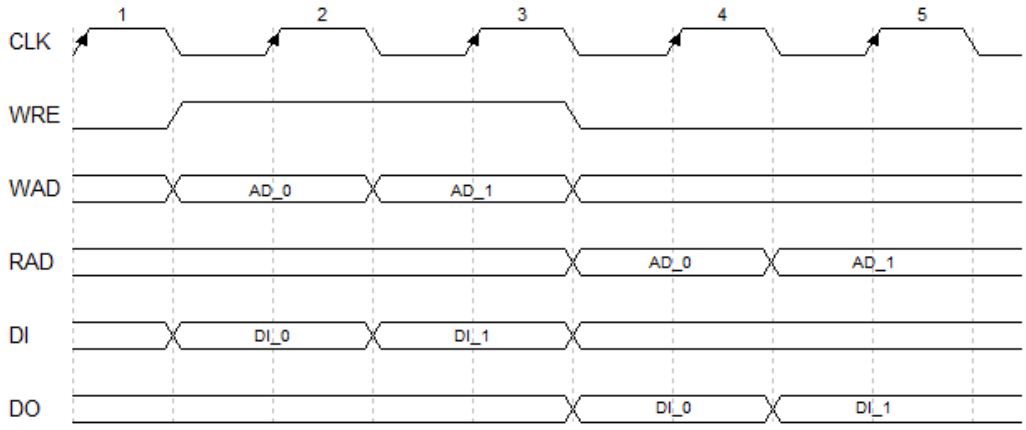
Primitive

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

Functional Description

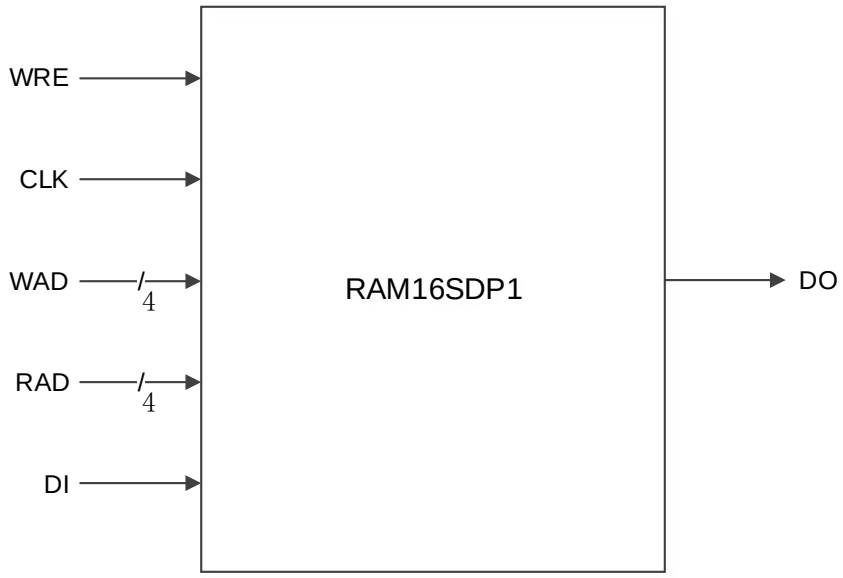
RAM16SDP1 is a semi-dual-port SSRAM with bit width of 1. The read address is RAD and the write address is WAD, and they are asynchronous. When WRE is high, you can perform write operation. At this time, the data is loaded to the address corresponding to memory at the pos-edge of CLK. The address determines the read data in RAM in read operation. The timing diagram is shown in Figure 5-5.

Figure 5-5 Timing Diagram of RAM16SDP1



Port Diagram

Figure 5-6 RAMSDP1 Port Diagram



Port Description

Table 5-8 Port Description

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

Parameter

Table 5-9 Parameter Description

Name	Value	Default	Description
INIT_0	16'h0000~16'hffff	16'h0000	Initial value of the RAM16SDP1

Primitive Instantiation

You can instantiate the primitives directly or generate them through IP Core Generator. For details, see chapter 6 IP Generation.

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
        );
```

5.4 RAM16SDP2

Primitive

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

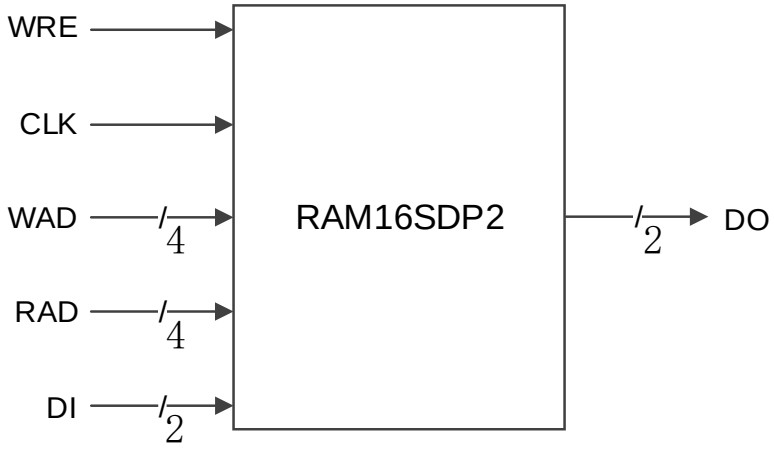
Functional Description

RAM16SDP2 is a semi-dual-port SSRAM with bit width of 2. The read address is RAD and the write address is WAD, and they are asynchronous.

When WRE is high, you can perform write operation. At this time, the data is loaded to the address corresponding to memory at the pos-edge of CLK. The address determines the read data in RAM in read operation. The timing diagram is shown in Figure 5-5.

Port Diagram

Figure 5-7 RAM16SDP2 Port Diagram



Port Description

Table 5-10 Port Description

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

Parameter

Table 5-11 Parameter Description

Name	Value	Default	Description
INIT_0~ INIT_1	16'h0000~16'hffff	16'h0000	Initial value of the RAM16SDP2

Primitive Instantiation

You can instantiate the primitives directly or generate them through IP Core Generator. For details, see chapter 6 IP Generation.

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
         );
```

5.5 RAM16SDP4

Primitive

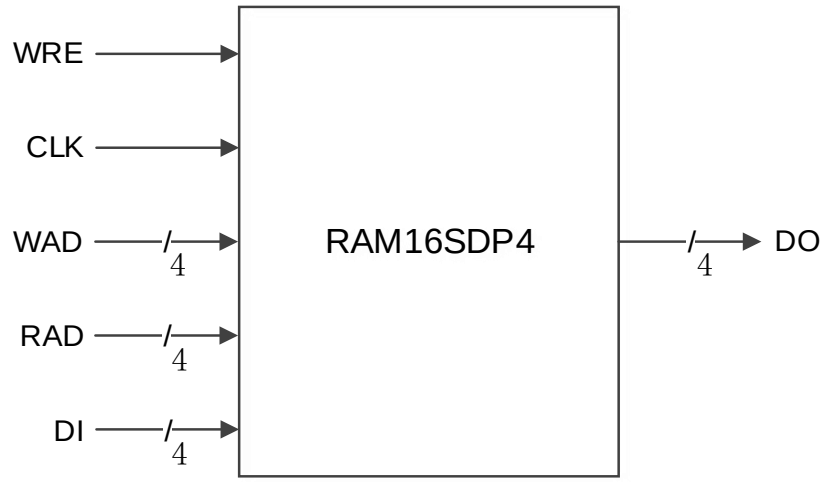
16-Deep by 4-Wide Semi Dual-port SSRAM (RAM16SDP4) is a semi-dual-port SSRAM with address depth of 16 and bit width of 4.

Functional Description

RAM16SDP4 is a semi-dual-port SSRAM with bit width of 4. The read address is RAD and the write address is WAD, and they are asynchronous. When WRE is high, you can perform write operation. At this time, the data is loaded to the address corresponding to memory at the pos-edge of CLK. The address determines the read data in RAM in read operation. The timing diagram is shown in Figure 5-5.

Diagram

Figure 5-8 RAMSDP4 Diagram



Port Description

Table 5-12 Port Description

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

Parameter

Table 5-13 Parameter Description

Name	Value	Default	Description
INIT_0~ INIT_3	16'h0000~16'hffff	16'h0000	Initial value of of the RAM16SDP4

Primitive Instantiation

You can instantiate the primitives directly or generate them through IP Core Generator. For details, see chapter 6 IP Generation.

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
);
```

5.6 ROM16

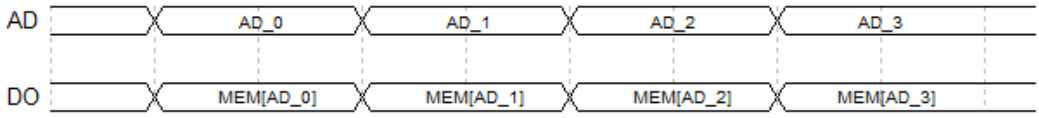
Primitive

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

Functional Description

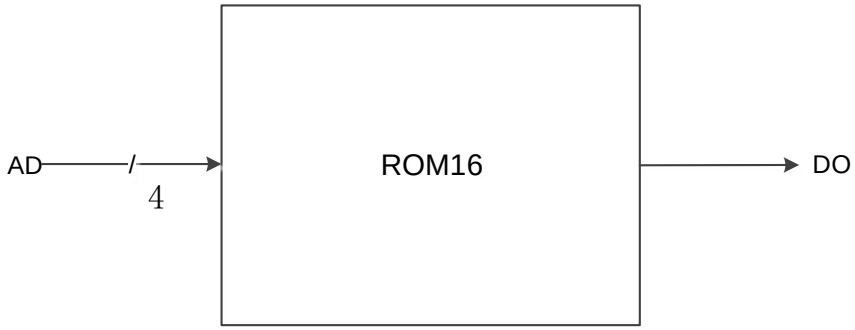
ROM16 is a read-only memory with a data bit width of 1, and the address determines the output data in ROM. The timing diagram is shown in Figure 5-9.

Figure 5-9 Timing Diagram of ROM16



Port Diagram

Figure 5-10 ROM16 Port Diagram



Port Description

Table 5-14 Port Description

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

Parameter

Table 5-15 Parameter Description

Name	Value	Default	Description
INIT_0	16'h0000~16'hffff	16'h0000	Initial value of the ROM16

Primitive Instantiation

You can instantiate the primitives directly or generate them through IP Core Generator. For details, see chapter 6 IP Generation.

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
    );
```

6 IP Generation

You can configure data width, address depth, write mode and read mode of IP Core in Gowin software, and Gowin software generates the corresponding IP module. In addition, there are two ways to implement BSRAM and SSRAM. You can configure ports and parameters to generate the required IP module by invoking Gowin software library files. You can also select synthesis tool to automatically generate BSRAM and SSRAM modes.

In IP Core Generator, the BSRAM module can implement single-port mode, semi-dual-port mode, dual-port mode and read-only mode. The SSRAM module can implement single-port mode, semi-dual-port mode and read-only mode. The following takes BSRAM dual-port mode and SSRAM single-port mode to introduce IP invoking.

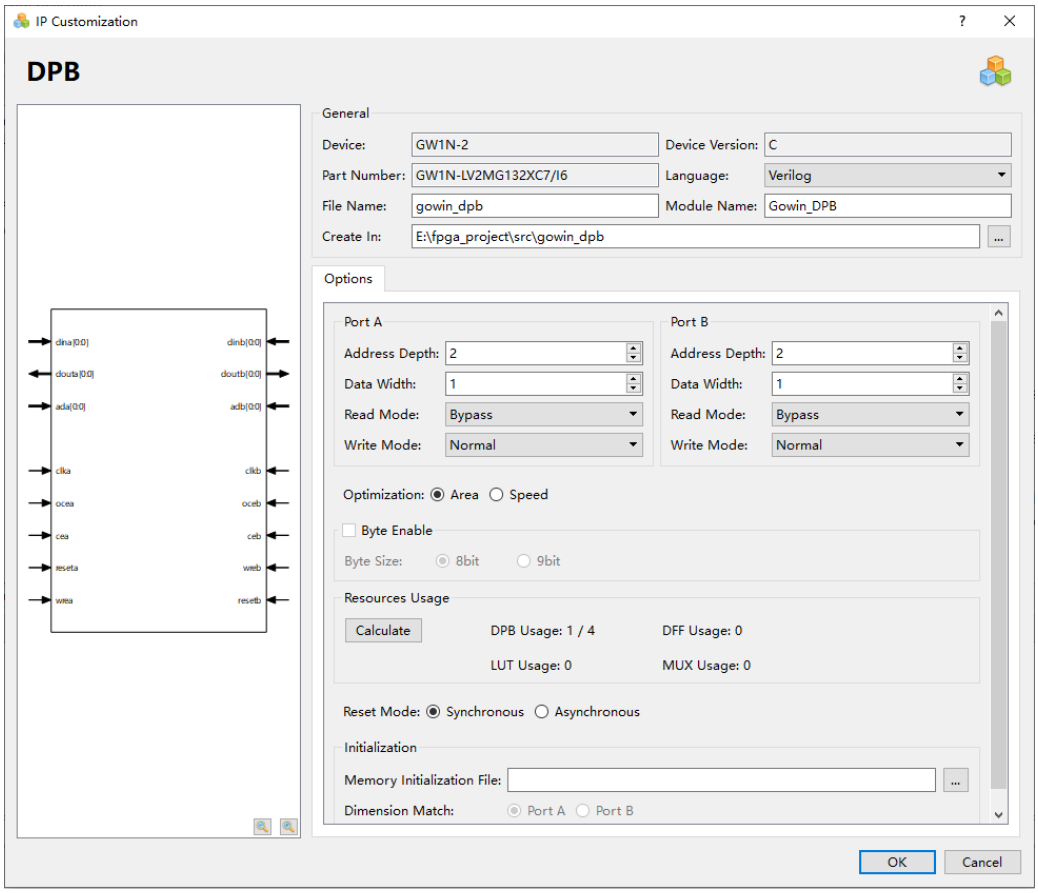
6.1 BSRAM Dual-port Mode

DP is the dual port mode of BSRAM, which can be implemented by DPB and DPX9B primitives. Click "DPB" on the IP Core Generator interface. A brief introduction to the DPB will be displayed on the right of the screen.

IP Configuration

Double click the "DPB" to open the "IP Customization" window. This includes the "General" configuration, "Options" configuration, and ports diagram, as shown in Figure 6-1.

Figure 6-1 IP Customization of DPB



1. The General configuration box includes the basic information related to IP deisgn file, as shown in Figure 6-1.
   - Device: Select a device
   - Device Version: Select a device version
   - Part Number: Select a part number
   - Language: Verilog and VHDL
   - Module Name: The module name of the generated IP Core files. Enter the module name in the text box on the right side. Module name cannot be the same as the primitive Name. If it is the same, an error will be reported;
   - File Name: The name of the generated IP Core files. Enter the file name in the text box on the right side;
   - Create In: The path in which the generated IP files will be stored; Enter the target path in the box on the right side or select the target path by clicking the option button.
2. The Options configuration box is as shown in Figure 6-1.
   - Width & Depth: Configure address depth and data width. If the configuration cannot be implemented by one module, multiple modules will be used to implement the configuration;

- Resource Usage: Calculate and display the resource usage of Block Ram, DFF, LUT, and MUX;
- Read/Write Mode: Configure Read/Write modes. DPB supports the following modes;
  - Two Read modes: Bypass and Pipeline;
  - Two Write modes: Normal and Write-Through
- Reset Mode: synchronous or asynchronous;
- Initialization: Allows you to select a memory initialization file for the module. INIT value is written in the initialization file in Binary, Hex or Hex with address. The Memory Initialization File can be written or generated by the menus "File > New > Memory File". For details, please refer to *SUG100, Gowin Software User Guide*.

Note!

- The address depth, data width and read/write mode of the Port A and Port B of the DPB can be configured independently.
- Port A and Port B of DPB are read and written to the same memory, so Address Depth*Data Width of Port A and Port B must be the same.
- The data width in the initialization file (Memory initialization File) should match the port data width selected by Dimension Match.
- If the address depth and data width of DP Port A and Port B are different, an error message will pop up.
- If the data width is different, the Init value of the generated DPB instantiation is 0 by default, and an error message will be displayed: Error (MG2105): Initial values' width is unequal to user's width.

3. Ports Diagram
   - The ports configuration diagram displays the IP Core configuration. Input/Output bit-width updates in real time based on the "Options" configuration, as shown in Figure 6-2.
   - "Address Depth" of Port A and Port B affects the bit-width of address; "Data Width" affects the bit-width of input and output.

IP Generation Files

After configuration, click "OK" to generate three files that are named after the "File Name":

- " gowin_dpb.v " file is a complete Verilog module to generate instance RAM16S, and it is generated according to the IP configuration;
- "gowin_dpb_tmp.v" is the instance template file;
- "gowin_dpb.ipc " file is IP configuration file. You can load the file to configure the IP.

Note!

If VHDL is selected as the hardware description language, the first two files will be named with an .vhd suffix. Taking verilog for instance, the following sections introduce the generated files.

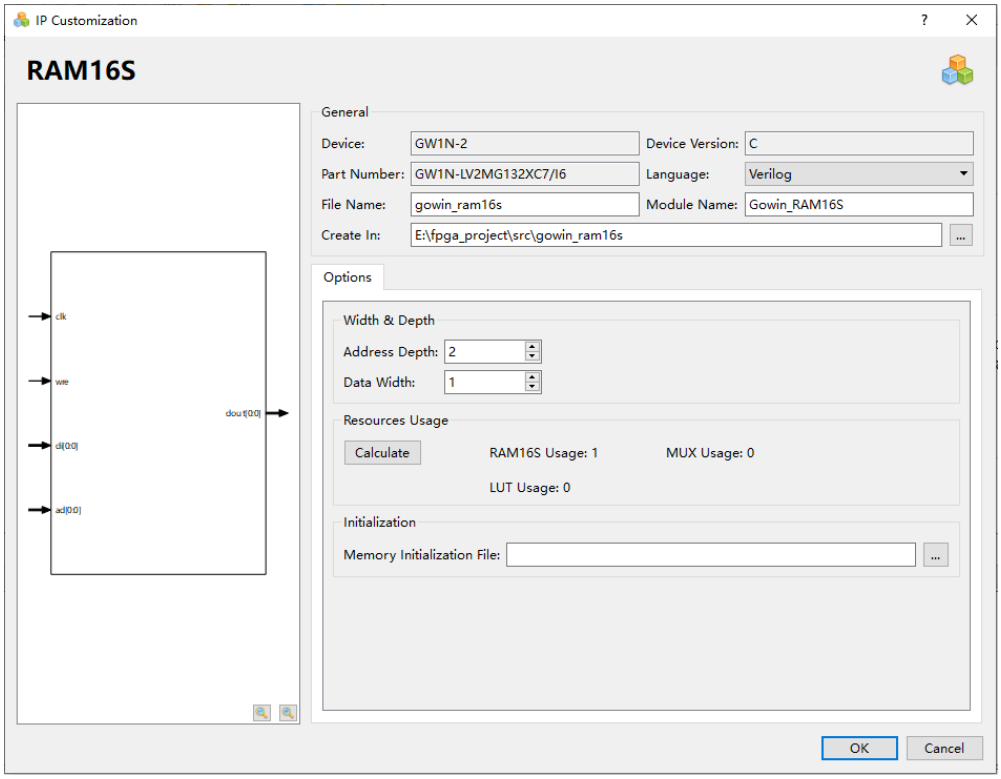
6.2 BSRAM Single-port Mode

RAM16S is SSRAM single-port mode, which can be implemented by RAM16S1, RAM16S2 and RAM16S4 primitives. Click "RAM16S" on the IP Core Generator, and a brief introduction to the RAM16S will be displayed.

IP Configuration

Double-click "RAM16", and the "IP Customization" window pops up. This includes the "File", "Options", and ports diagram, as shown in Figure 6-2.

Figure 6-2 IP Customization of RAM16S



1. The General configuration box is used to configure IP design file. The File configuration box of the RAM16S is similar to the BSRAM dual-port mode. For details, refer to the General configuration box of 6.1 BSRAM Dual-port Mode.
2. The Options configuration box is as shown in Figure 6-2. The Options configuration box of the RAM16S is similar to the BSRAM dual-port mode. For details, refer to the Options configuration box of 6.1 BSRAM Dual-port Mode.
3. Ports Diagram
   - The ports diagram is based on the IP Core configuration. The input/output bit-width updates in real time based on the "Options" configuration.
   - "Address Depth" affects the bit-width of Address; "Data Width" affects the bit-width of input and output.

IP Generation Files

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

- " gowin_ram16s.v " file is a complete Verilog module to generate instance RAM16S, and it is generated according to the IP configuration;
- "gowin_ram16s_tmp.v" is the instance template file;
- "gowin_ram16s.ipc" file is IP configuration file. You can load the file to configure the IP.

Note!

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

7 Initialization File

In BSRAM and SSRAM modes, each bit of memory can be initialized to 0 or 1. The initial value is written in binary, hex, or hex with address in the initialization file.

7.1 Bin File

The Bin file is a text file consisting of 0 and 1. The row number represents the address depth, and the column number represents the data width.

#File_format=Bin

#Address_depth=16

#Data_width=32

00001100000100000000100100010000

10000000010010000100000001000000

01000000100000001000000010000000

00100000100001001100000011000000

7.2 Hex File

The Hex file is similar to the Bin file, and it consists of the hexadecimal number 0~F. The row number represents the address depth and the binary bit of each row represents the data width.

#File_format=Hex

#Address_depth=8

#Data_width=16

3A40

A28E

0B52

1C49

D602

0801

03E6

4C18

7.3 Hex File with Address

The Hex file with address records the addresses and data recorded in the file. The addresses and data are composed of 0~F. The address and data are separated by a colon. The file only records the written addresses and data. The unrecorded addresses default to 0.

#File_format=AddrHex

#Address_depth=256

#Data_width=16

9:FFFF

23:00E0

2a:001F

30:1E00


GOWIN
PROGRAMMING FOR THE FUTURE