Gowin IP Core Generator

User Guide

SUG284-1.9E, 11/28/2019

Disclaimer

Revision History

Date	Version	Description
3/7/2017	1.0E	Initial version published.
1/30/2018	1.2E	● GW1NR-4, GW1N-6, GW1N-9 and GW1NR-9 supported; ● BSRAM updated; ● DSP updated; ● PLL updated; ● User Flash updated.
8/25/2018	1.3E	● GW1N-2B, GW1N-4B, GW1N-6ES, GW1N-9ES, GW1NR-4B, GW1NR-9ES, GW1NS-2,GW1NS-2C supported; ● IP DDR3 and DDR3 PHY supported; ● PLL updated; ● OSC updated; ● User Flash updated; ● Interface display optimized; ● IP CORDIC, Complex Multiplier and DIVIDER added;
10/26/2018	1.4E	● GW1NZ-1 and GW1NSR-2C supported. ● I3Cand SPMI hardcore added.
11/15/2018	1.5E	● GW1NSR-2 supported; ● GW1N-6ES, GW1NS-9ES and GW1NR-9ES removed;
2019/02/12	1.6E	● IP RiscV N25 and CAN added; ● PSRAM, DDRx and MIPI updated.
201/9/02/25	1.7E	● IP Basic FIR Filter, FD Adaptive Filter, Integer Multiply Divider, NLMS Adaptive Filter, XCORR and Triple Speed Ethernet MAC added; ● Interface display optimized (Add to Current Project option removed).
5/17/2019	1.8E	● GW1N-1S supported; ● IP PSRAM Memory Interface 2CH, Advanced FIR Filter, Gowin_EMPU_M1, HyperRAM Memory Interface added; ● Shadow Memory of Hard Module, including RAM16S, RAM16SDP and ROM16 added; ● IP MIPI, DDR, DDR2, DDR3 and GOWIN_EMPU updated.
2/019/11/28	1.9E	● GW1NS-4, GW1NRF-4B, GW1NSE-2C, GW1NSER-4C, GW1NSR-4, GW1NSR-4C supported; ● Users can select Synplify Pro or GowinSynthesis as the synthesis tool; ● Classification of Soft IP Core adjusted; ● BandGap, rPLL, PLLVR, DPB, DPBX9, SDPB, SDPBX9, rSDP, rSDPX9, rROM, rROMX9, pROM, pROMX9 added in Hard Module; ● Boot comments of the preferred primitives added.

Contents

List of Figures

List of Tables

1 About This Guide

1.1 Purpose

This guide describes the usage of IP Core Generator in Gowin YunYuan software to help users realize complex designs in a more convenient way. Gowin Yunyuan software supports both Linux and Windows operating systems. The software screen shots and the supported products listed in this guide are based on the version Windows 1.9.3Beta. As the software is subject to change without notice, some information may not remain relevant and may need to be adjusted according to the software that is in use.

1.2 Supported Products

The information presented in this guide applies to the following products:

- GW1N series of FPGA products: GW1N-1, GW1N-2, GW1N-2B, GW1N-4, GW1N-4B, GW1N-6, GW1N-9, GW1N-1S
- GW1NR series of FPGA products: GW1NR-4, GW1NR-4B, GW1NR-9
- GW1NS series of FPGA products: GW1NS-2, GW1NS-2C, GW1NS-4
- GW2A series of FPGA products: GW2A-55 and GW2A-18;
- GW2AR series of FPGA products: GW2AR-18.
- GW1NZ series of FPGA products: GW1NZ-1
- GW1NSR series of FPGA products: GW1NSR-2, GW1NSR-2C, GW1NSR-4, and GW1NSR-4C
- GW1NRF series of Bluetooth FPGA products: GW1NRF-4B
- GW1NSE series of SecureFPGA products: GW1NSE-2C
- GW1NSER series of SecureFPGA products: GW1NSER-4C

1.3 Related Documents

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

1. DS100, GW1N series of FPGA Products Data Sheet
2. DS117, GW1NR series of FPGA Products Data Sheet
3. DS821, GW1NR series of FPGA Products Data Sheet
4. DS102, GW2A series of FPGA Products Data Sheet
5. DS226, GW2AR series of FPGA Products Data Sheet
6. DS841, GW1NR series of FPGA Products Data Sheet

7. DS861, GW1NR series of FPGA Products Data Sheet
8. DS891, GW1NR series of FPGA Products Data Sheet
9. DS871, GW1NR series of FPGA Products Data Sheet
10. DS881, GW1NR series of FPGA Products Data Sheet

1.4 Terminology and Abbreviations

Table 1-1 shows the abbreviations and terminology that are employed in this guide.

Table 1-1 Terminology and Abbreviations

Terminology and Abbreviations	Meaning
FPGA	Field Programmable Gate Array
IDE	Integrated Development Environment
IP Core	Intellectual Property Core
DP/DPX9	Dual Port
DPB/DPBX9	Dual Port
SP/SPX9	Single Port
SDP/SDPX9	Semi Dual Port
SDPB/SDPBX9	Semi Dual Port
rSDP/rSDPX9	Semi Dual Port
ROM/ROMX9	Read Only Memory
rROM/rROMX9	Read Only Memory
pROM/pROMX9	Read Only Memory
PADD	PADD
MULT	MULT
PLL	PLL
rPLL	PLL
PLLVR	PLL
DLL	DLL
OSC	On Chip Oscillator
SPMI	System Power Management Interface

1.5 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 by the following ways.

Website: www.gowinsemi.com
E-mail:support@gowinsemi.com
Tel: +86 755 8262 0391

2 Introduction

The IP Core Generator in Gowin YunYuan software is used to generate instantiation components and IPs that users can call to implement required functions. They help users realize complex designs in a more convenient way. The IP Core Generator includes the Hard Module associated with primitives and the Soft IP Core associated with the reference designs.

3 Usage

Select "Tools > IP Core Generator" in the menu bar or tool bar "  " to open the IP Core Generator interface, as shown in Figure 3-1.

The interface includes two parts:

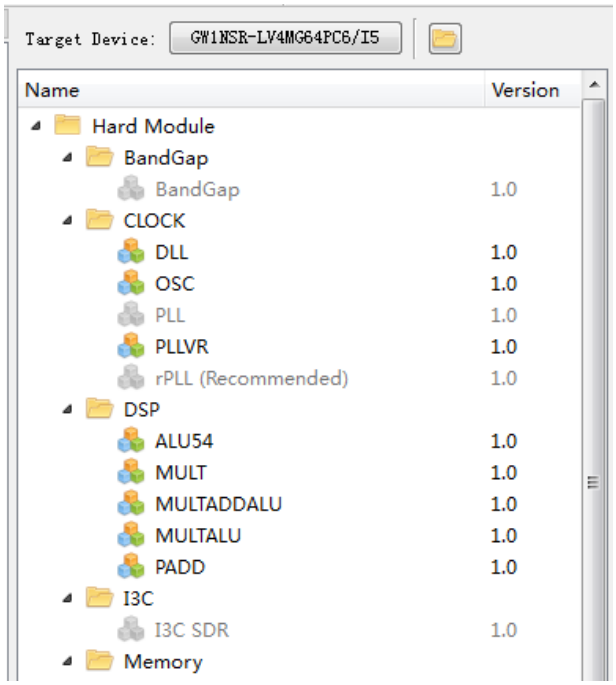
- The Hard Module associated with primitives;
- The Soft IP Core associated with the reference designs.

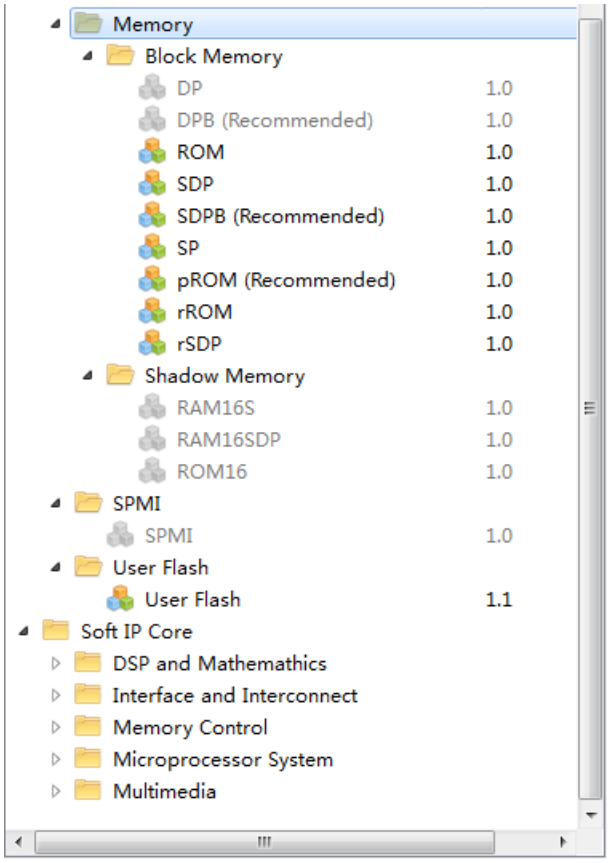
The Hard Module includes BandGap, CLOCK, DSP, I3C, Memory, SPMI and User Flash, etc.

Soft IP Core includes DSP and Mathematics, Interface and Interconnect, Memory Control, Microprocessor System, Multimedia, etc.

This guide introduces the usage of Hard Module.

Figure 3-1 IP Core Generator Interface



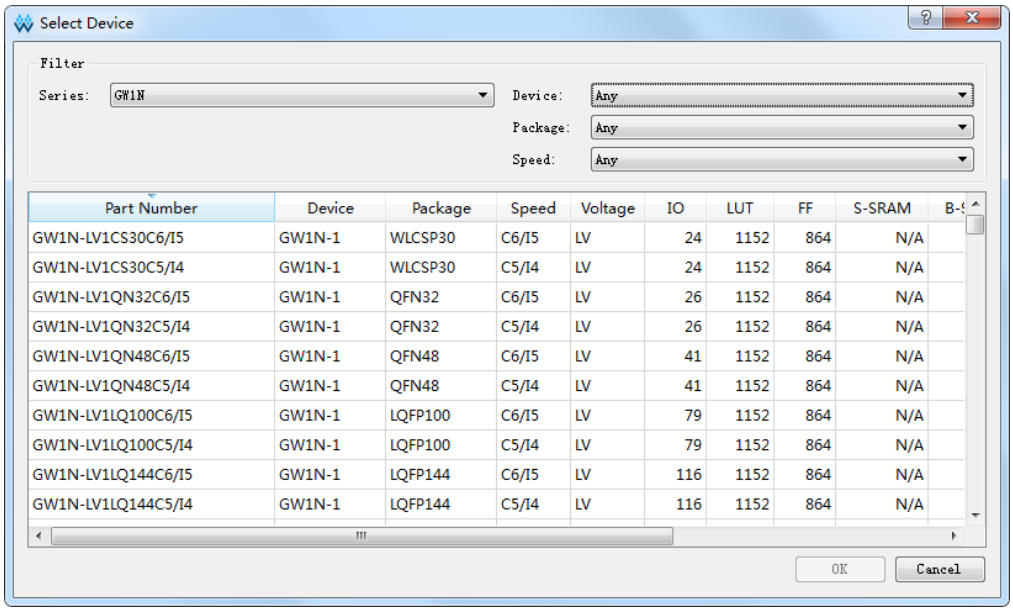


There are two icons at the top of the interface. One is for the target device, and "  " is used to open the IP core configuration files.

Target Device is used to configure device. Click the right display box and "Select Device" pops up, as shown in Figure 3-2.

The device information can be modified through this window. The modified part will be displayed in the box on the right side of the Target Device. Double-click on a highlighted IP to open the IP Customization dialog box. The modified part will also appear in the "Target Device" display box in the File configuration window of the IP Customization dialog box.

Figure 3-2 Select Device



After Device is selected, the IP Core Generator will automatically determine whether a specific module is supported or not based on the device.

- If supported, the module name is highlighted. Double-clicking to open the "IP Customization" configuration window. As shown in Figure 3-3, users can configure IP through the IP Customization configuration window. After the configuration is completed, click "OK" to generate IP. The configuration interface of each IP will be introduced in each section of chapter 3.
- The IP Cores or modules that are displayed in grey are not supported. As shown in Figure 3-1, GW1NSR-4 does not support SPMI.

"  " This icon can be used to open the configured IP core files. These can be edited according to the user requirements. Click the icon to open the "Select IP Config File" dialog box, and then select the IP Core Config file ".ipc". The Customize IP window opens for reconfiguration where the file path cannot be changed, as shown in Figure 3-4.

Figure 3-3 IP Customization

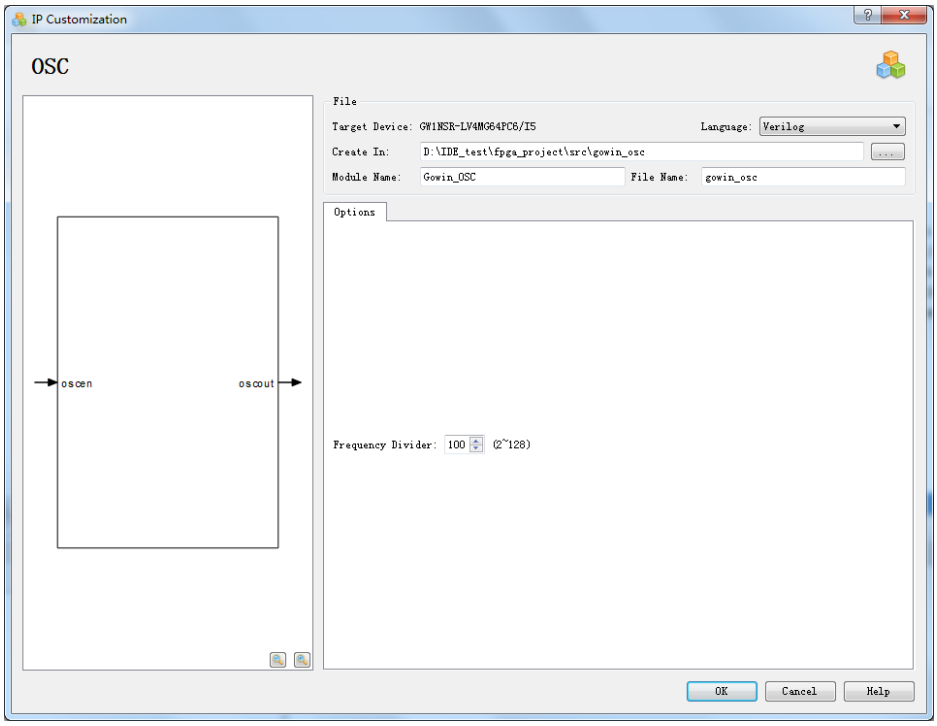
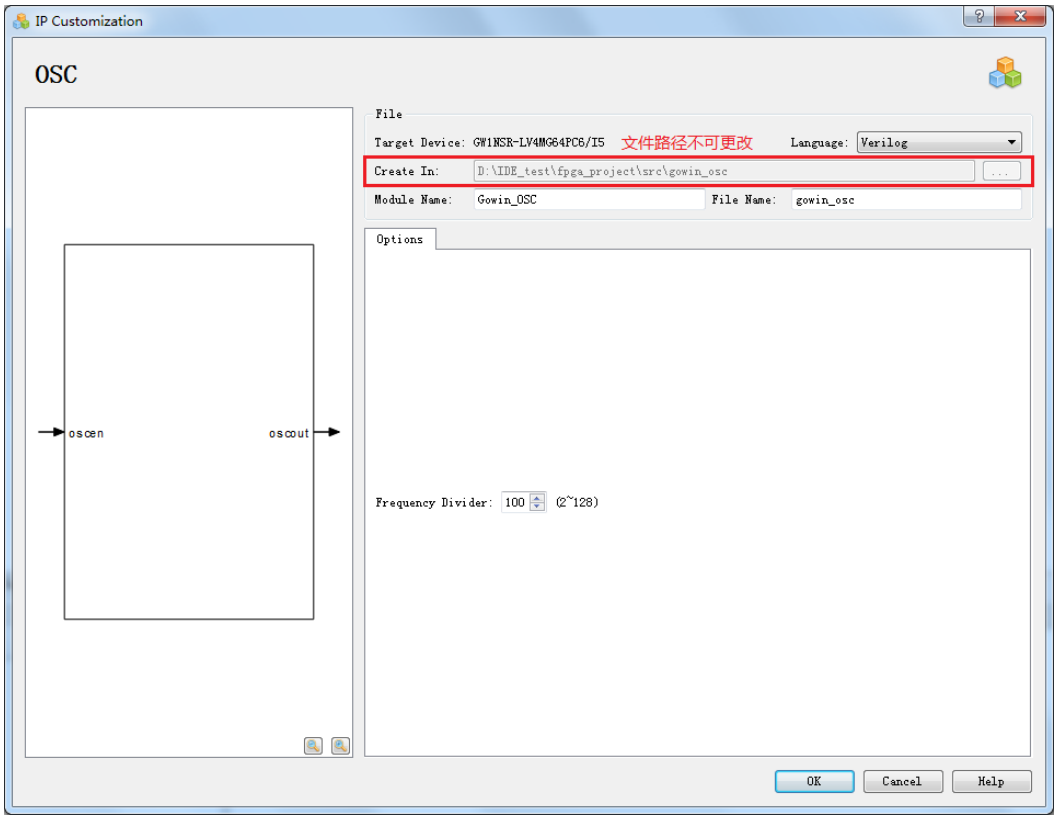


Figure 3-4 IP Customization of IPC File



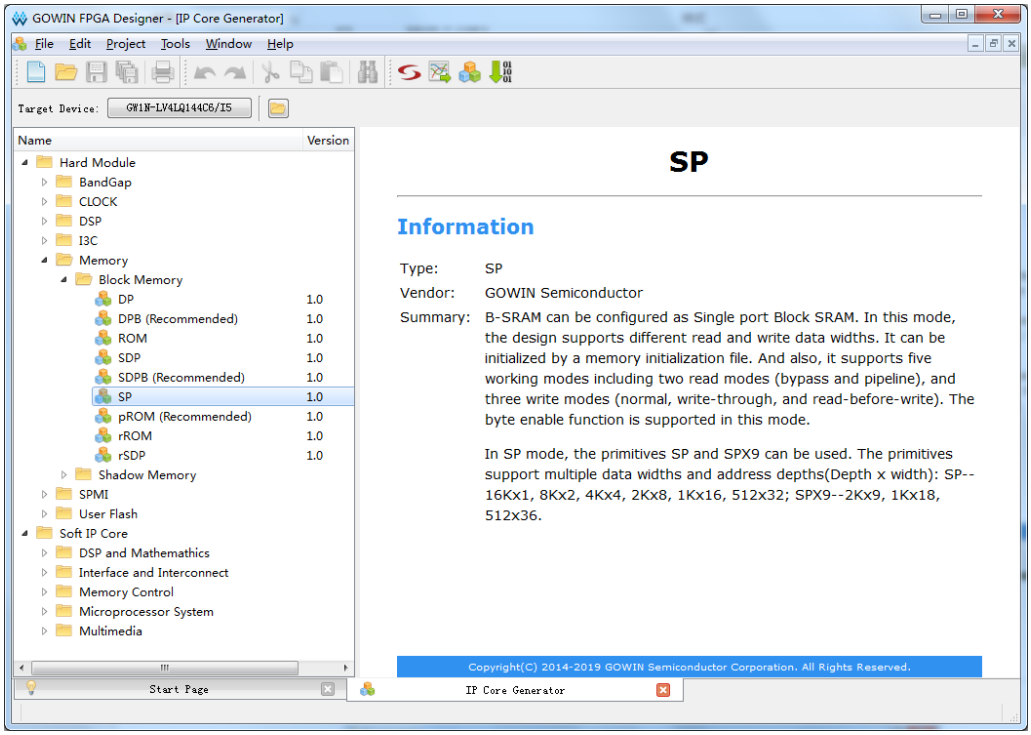
3.1 Block Memory

Currently, Block Memory (BSRAM) can realize Single Port (SP), Semi-dual Port (SDP), Semi-dual Port (rSDP), Semi-dual Port (SDPB), Dual Port (DP), Dual Port (DPB), Read Only Memory (rROM), Read Only Memory (pROM) and Read Only Memory (ROM).

3.1.1 SP

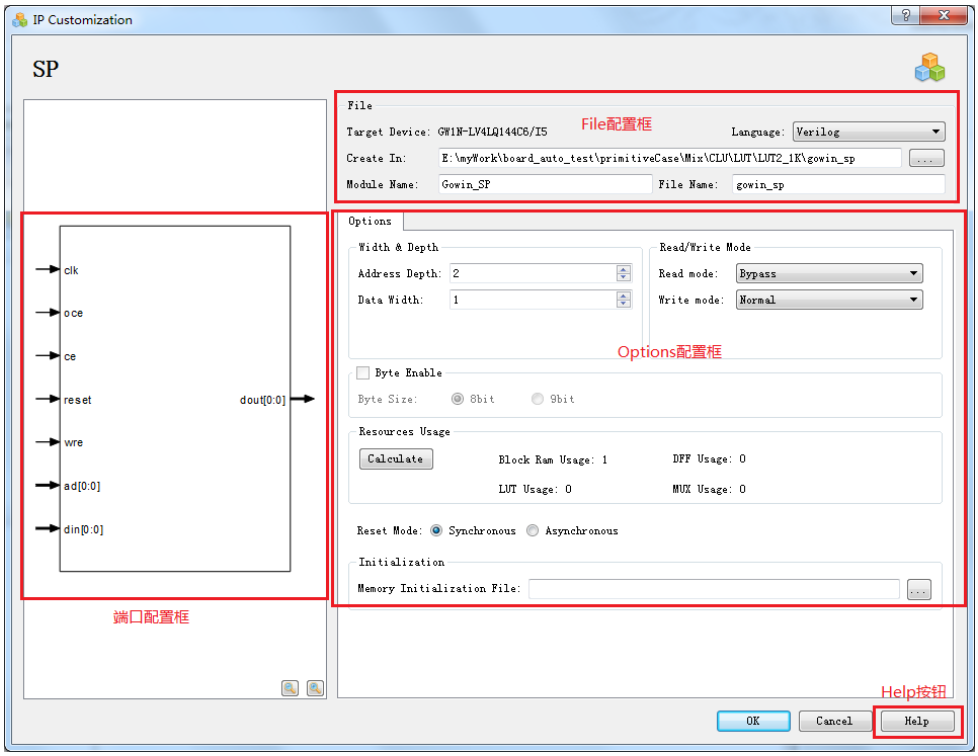
SP is a single port block memory that can be implemented by SP and SPX9. The maximum capacity of BSRAM varies according to the type of chip. Click "SP" in IP Core Generator interface. A brief introduction of SP shows on the right, as shown in Figure 3-5.

Figure 3-5 SP Summary Information



Double-clicking on "SP", and the "IP Customization" window pops up, as shown in Figure 3-6. This includes the "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button.

Figure 3-6 IP Customization of SP



1. File Configuration
   - The file configuration includes the basic information related to the SP instantiation file;
   - Target Device: Displays the information of configured device;
   - Language: Hardware description language used to generate the IP Core files. Click the right drop-down list to select the target language, including Verilog and VHDL, as shown in Figure 3-7；
   - 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. "Module Name is the same as primitive Name!" as shown in Figure 3-8；
   - 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.

Figure 3-7 Drop-down List of Language

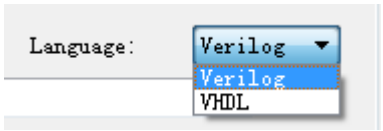
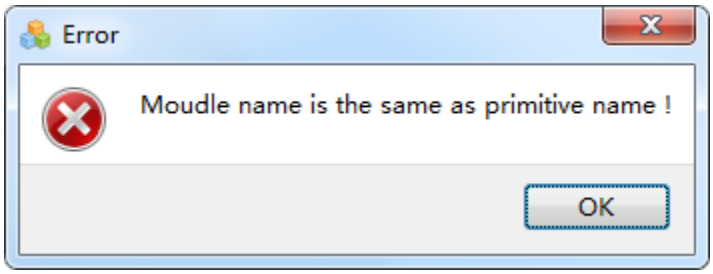


Figure 3-8 Error Message



2. Options Configuration

The Options configuration box is used to configure single-port block memory by users, as shown in Figure 3-6.

- Width & Depth: Configure SP address depth and data width. If the configuration cannot be implemented by one module, multiple modules will be used to implement the current configuration;
- Byte Enable is used to configure the use of Byte Enable. Check if the data width is equal to or greater than 9, and byte size can be 8 bits or 9 bits when using it;
- Resource Usage: Calculate and display the resource usage of Block Ram, DFF, LUT, and MUX for the current configuration;
- Read/Write Mode: Configures Read/Write mode；
- SP supports the following modes；
- Two Read modes: Bypass and Pipeline;
- Three Write modes: Normal, Write-Through, Read-before-Write；
- Reset Mode: Configure the reset mode of SP;
- Reset Mode can be synchronous or asynchronous；
- Initialization: Configure the INIT value of SP；
- Memory Initialization File: Allows you to select a memory initialization file for the module. INIT value is written in the Initialization File in Binary or Hex formats.

Note!

The Memory Initialization File can be written or generated by the menus "File->New->Memory File". For detailed instructions on how to generate the memory file and the associated file format, please refer to *Gowin Yunyuan Software User Guide*。

3. Ports Configuration Diagram
   - The ports configuration diagram displays the current IP Core configuration result. The Input/Output bit-width updates in real time based on the "Options" configuration, as shown in Figure 3-6.
   - "Address Depth" determines the bit-width of ad; "Data Width" determines the bit-width of din and dout.
4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-9.

Figure 3-9 Help

SP

Information

Type:	SP
Vendor:	GOWIN Semiconductor
Summary:	B-SRAM can be configured as Single port Block SRAM. In this mode, the design supports different read and write data widths. It can be initialized by a memory initialization file. And also, it supports five working modes including two read modes (bypass and pipeline), and three write modes (normal, write-through, and read-before-write). The byte enable function is supported in this mode. In SP mode, the primitives SP and SPX9 can be used. The primitives support multiple data widths and address depths(Depth x width): SP--16Kx1, 8Kx2, 4Kx4, 2Kx8, 1Kx16, 512x32; SPX9--2Kx9, 1Kx18, 512x36.

Options

Option	Description
Width & Depth	**Address Depth** - Set the size of the address depth.
	Data Width - Set the size of the data width.
Read/Write Mode	**Read Mode** - Set whether the read mode is bypass mode or pipeline mode.
	Write Mode - Set the write mode as normal mode, write-through mode or read-before-write mode.
Byte Enable	**Byte Enable** - Set whether to use byte enable function or not.
	Byte Size - Set whether the byte size is 8bit or 9bit if the byte enable selected.
	Note: Assume that the data width is represented by Width. (1) If Width<=8, byte enable function is invalid; (2) If Width=9, only 8 bit is valid; (3) If Width>9, both 8 bit and 9 bit are valid.
Resource Usage	**Calculate** - Calculate the resource usage in the design and display results below.
	Block Ram Usage - Display the number of Block Ram used.
	DFF Usage - Display the number of DFF used.
	LUT Usage - Display the number of LUT used.
	MUX Usage - Display the number of MUX used.
Reset Mode	**Reset Mode** - Set whether the reset mode is synchronous mode or asynchronous mode.
Initialization	**Memory Initialization File** - Set the memory initialization file (.mi) path.

The Help page contains a general description of the IP Core, and a brief introduction to the "Options".

IP Generation Files

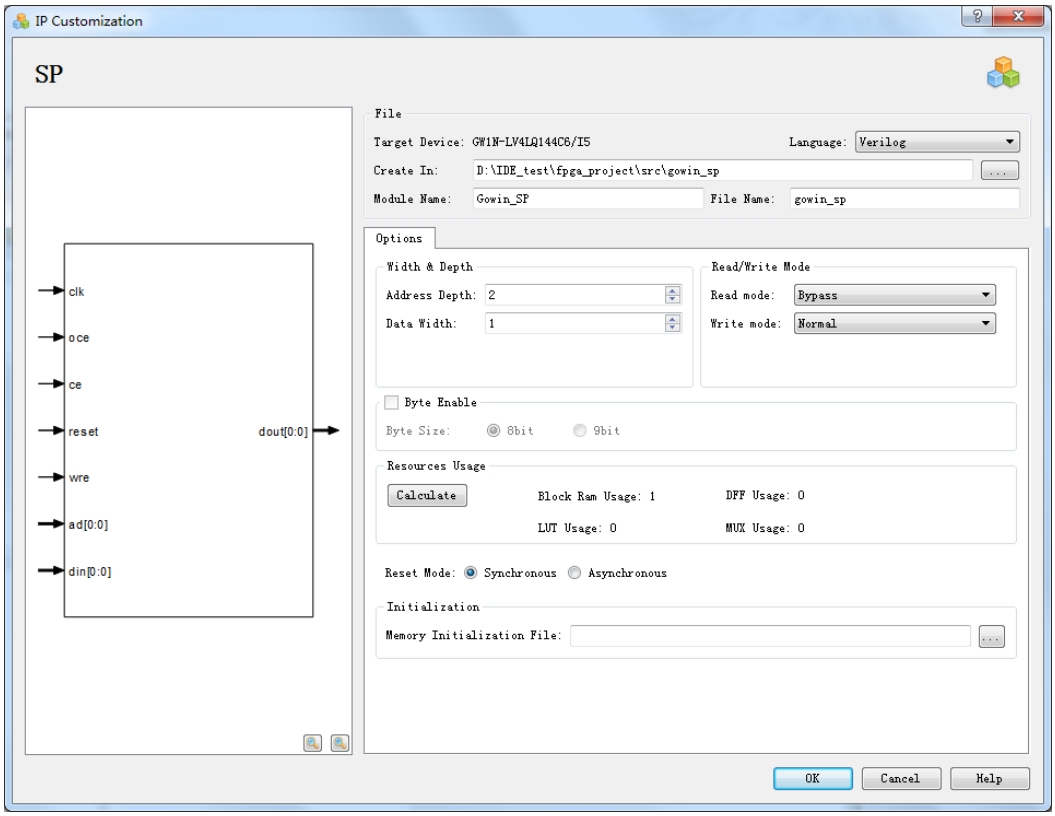
As shown in Figure 3-10, after customizing the IP, click "OK" to generate three files based on the "File Name".:

- Instantiate Gowin Primitive SP design file "gowin_sp.v";
- The instantiation template file for the IP design file "gowin_sp_tmp.v";
- The configuration files for the Primitive SP instantiation "gowin_sp.ipc".

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.

Figure 3-10 Configured IP Customization



SP Design File Instantiation

SP Design File Instantiation is a complete Verilog module. SP instantiation is generated according to the SP configuration that is displayed in the "IP Customization" window, as shown in Figure 3-11.

Figure 3-11 SP Design File Instantiation

```
module Gowin_SP (dout, clk, oce, ce, reset, wre, ad, din);

output [0:0] dout;
input clk;
input oce;
input ce;
input reset;
input wre;
input [0:0] ad;
input [0:0] din;

wire gw_gnd;

assign gw_gnd = 1'b0;

SP bram_sp_0 (
    .DO(dout[0]),
    .CLK(clk),
    .OCE(oce),
    .CE(ce),
    .RESET(reset),
    .WRE(wre),
    .BLKSEL({gw_gnd,gw_gnd,gw_gnd}),
    .AD({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,ad[0]}),
    .DI(din[0])
);

defparam bram_sp_0.READ_MODE = 1'b0;
defparam bram_sp_0.WRITE_MODE = 2'b00;
defparam bram_sp_0.BIT_WIDTH = 1;
defparam bram_sp_0.BLK_SEL = 3'b000;
defparam bram_sp_0.RESET_MODE = "SYNC";

endmodule //Gowin_SP
```

Instantiation Template File for the IP Design File

For the pratical use, the IP Core Generator generates the template file while generating the SP design file instantiation, as shown in, Figure 3--12.

Figure 3--12 Instantiation Template File for the IP Design File

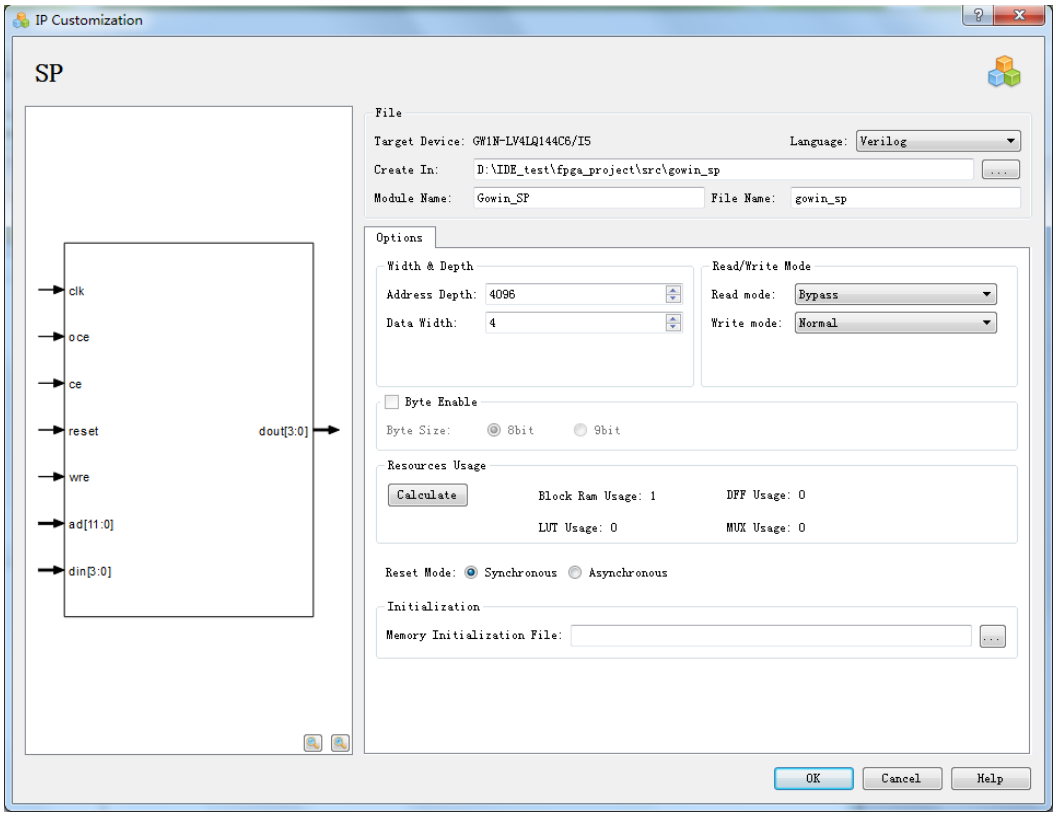
```
Gowin_SP your_instance_name(
    .dout(dout_o), //output [0:0] dout
    .clk(clk_i), //input clk
    .oce(oce_i), //input oce
    .ce(ce_i), //input ce
    .reset(reset_i), //input reset
    .wre(wre_i), //input wre
    .ad(ad_i), //input [0:0] ad
    .din(din_i) //input [0:0] din
);
```

SP Generation Example

If users need to generate a specific SP IP as follows: 4096 x 4, Bypass read mode, Normal write mode and Synchronous reset. Take the GW1N-LV4LQ144C6/I5 device for instance, the configuration interface is as shown in Figure 3-13. Usrers can configure initialization file in "Initialization" window, and then click "OK" to generate the customized SP IP design files.

The directory where the SP IP design file is generated is the path set in "Create In".

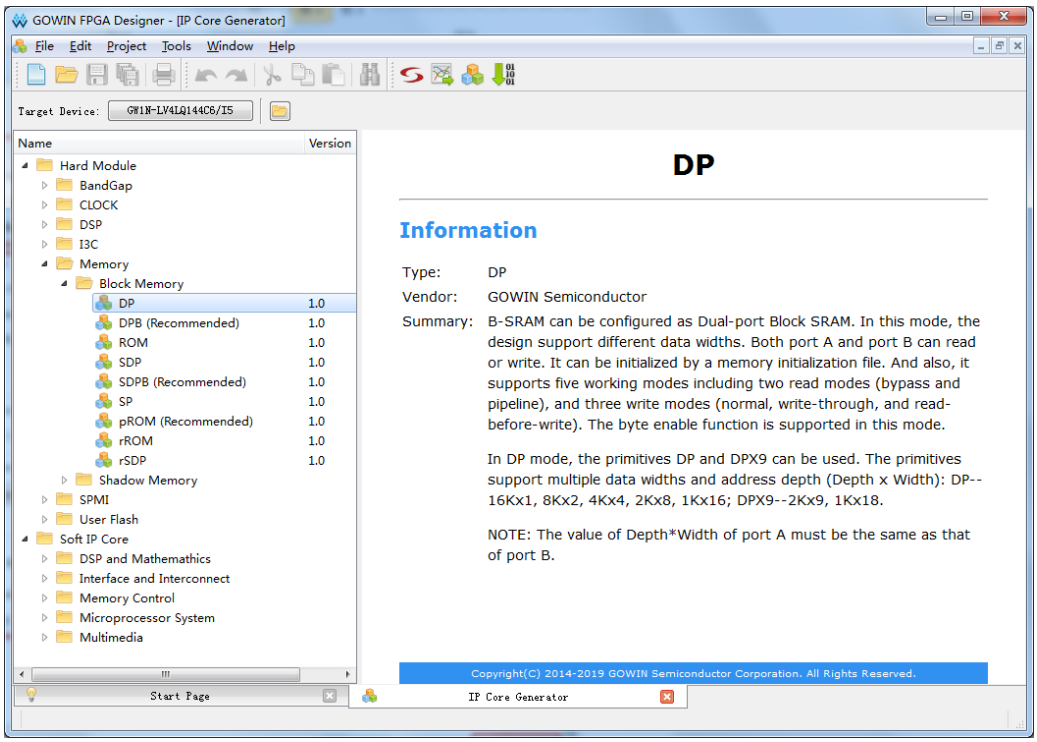
Figure 3-13 SP IP Customization Configuration



3.1.2 DP

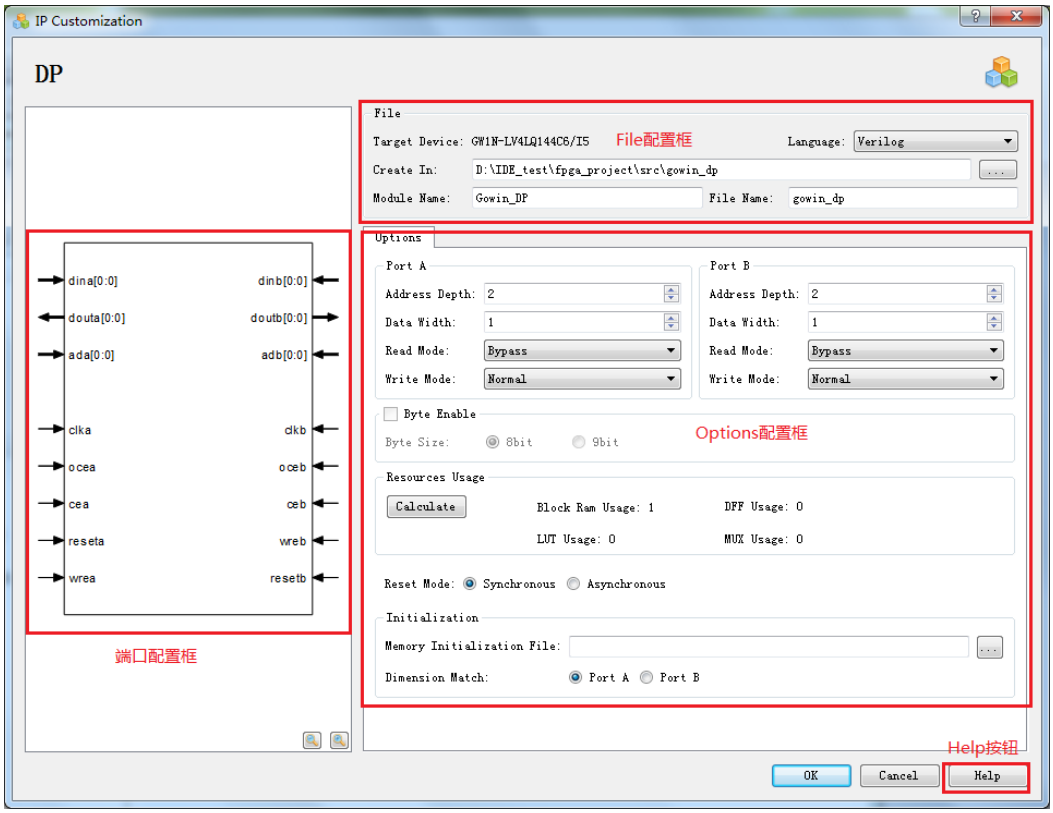
DP is the dual port block memory, which can be implemented by DP and DPX9. The maximum capacity of BSRAM varies according to the type of chip. Click "DP" on the IP Core Generator interface. A brief introduction to the DP will be displayed on the right of the screen, as shown in Figure 3-14.

Figure 3-14 DP Summary Information



Double click the "DP" to open the "IP Customization" window. This includes the "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button, as shown in Figure 3-15.

Figure 3-15 IP Customization of DP



1. File Configuration

The File configuration box includes the basic information related to the DP instantiation file, as shown in Figure 3-15.

The DP file configuration is similar to that of SP. For the detailed configuration, please refer to 3.1Block Memory>3.1.1SP > File Configuration.

2. Options Configuration

The Options configuration box is used to configure dual-port block memory by users, as shown in Figure 3-15.

DP Options configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1 Block Memory>3.1.1SP > Options Configuration.

Pay attention to the follows before configuring the DP:

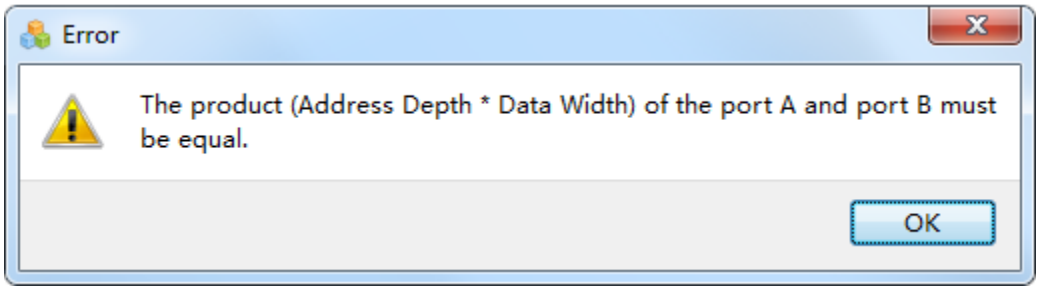
- The address depth, data width, and read/write mode of DP Port A and Port B can be configured independently.
- The address depth and data width of DP Port A and Port B must be equal because Port A and Port B read from or write to the same memory.
- The data width in the Memory initialization File should be consistent with the data width of the port specified in the "Dimension Match".

  Note!

  - If the address depth and data width of DP Port A and Port B are different, an error message will be displayed, as shown in Figure 3-16.
  - If the data width is different, the Init value of the generated DP instantiation is 0 by default, and an error message will be displayed:

    Error (MG2105): Initial values' width is unequal to user's width.

Figure 3-16 Error Message



3. Ports Configuration Diagram
   - The ports configuration diagram displays the current IP Core configuration result. Input/Output bit-width updates in real time based on the "Options" configuration, as shown in Figure 3-15.
   - "Address Depth" of Port A and Port B affects the bit-width of ada and adb; "Data Width" affects the bit-width of dia/doa and dib/dob.
4. Help

   Click "Help" to open the IP Core configuration information, as shown in Figure 3-17.

Figure 3-17 Help

DP

Information

Type:	DP
Vendor:	GOWIN Semiconductor
Summary:	B-SRAM can be configured as Dual-port Block SRAM. In this mode, the design support different data widths. Both port A and port B can read or write. It can be initialized by a memory initialization file. And also, it supports five working modes including two read modes (bypass and pipeline), and three write modes (normal, write-through, and read-before-write). The byte enable function is supported in this mode. In DP mode, the primitives DP and DPX9 can be used. The primitives support multiple data widths and address depth (Depth x Width): DP--16Kx1, 8Kx2, 4Kx4, 2Kx8, 1Kx16; DPX9--2Kx9, 1Kx18. NOTE: The value of Depth*Width of port A must be the same as that of port B.

Options

Option	Description
Port A	**Address Depth** - Set the size of the address depth.
	Data Width - Set the size of the Data width.
	Read Mode - Set whether the read mode is bypass mode or pipeline mode.
	Write Mode - Set the write mode as normal mode, write-through mode or read-before-write mode.
Port B	**Address Depth** - Set the size of the address depth.
	Data Width - Set the size of the Data width.
	Read Mode - Set whether the read mode is bypass mode or pipeline mode.
	Write Mode - Set the write mode as normal mode, write-through mode or read-before-write mode.
Byte Enable	**Byte Enable** - Set whether to use byte enable function or not.
	Byte Size - Set whether the byte size is 8bit or 9bit if the byte enable checkbox selected.
	Note: Assume that the data width is represented by Width. (1) If Width<=8, byte enable function is invalid; (2) If Width=9, only 8 bit is valid; (3) If Width>9, both 8 bit and 9 bit are valid.
Resource Usage	**Calculate** - Calculate the resource usage in the design and display results below.
	Block Ram Usage - Display the number of Block Ram used.
	DFF Usage - Display the number of DFF used.
	LUT Usage - Display the number of LUT used.
	MUX Usage - Display the number of MUX used.
Reset Mode	**Reset Mode** - Set whether the reset mode is synchronous mode or asynchronous mode.
Initialization	**Memory Initialization File** - Set the memory initialization file (.mi) path.
	Dimension Match - Set which port's dimensions the memory initialization file should conform to.

The "Help" page contains a general description of the IP Core, and a brief introduction to the "Options".

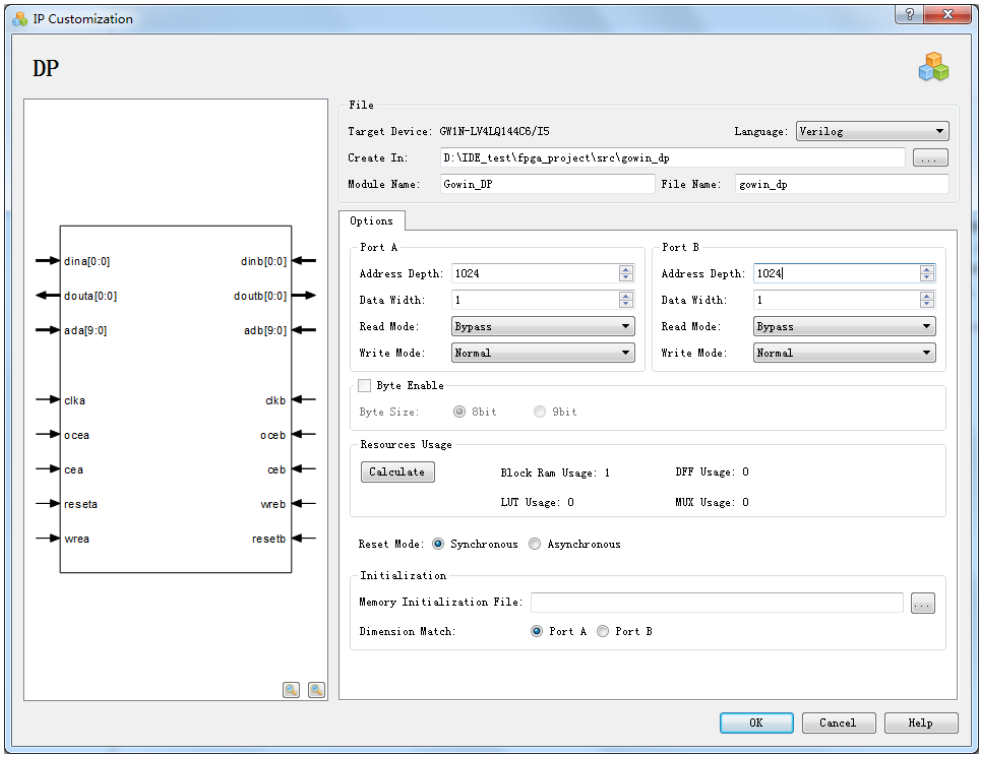
IP Generation Files

As shown in Figure 3-18, after customizing the DP IP, click "OK" to generate three files that are named after the "File Name":

- Instantiate Gowin Primitive DP design file "gowin_dp.v";
- The instantiation template file for the IP design file "gowin_dp_tmp.v";
- The configuration files for the Primitive DP instantiation "gowin_dp.ipc".

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.

Figure 3-18 Configured IP Customization



DP Design File Instantiation

DP instantiation design file is a complete Verilog module. DP instantiation is generated according to the DP configuration that is displayed in the "IP Customization" window, as shown in Figure 3-19.

Figure 3-19 DP Design File Instantiation

```
module Gowin_DP (douta, doutb, clka, ocea, cea, reseta, wrea, clkb, oceb, ceb, resetb, wreb, ada, dina, adb, dinb);

output [0:0] douta;
output [0:0] doutb;
input clka;
input ocea;
input cea;
input reseta;
input wrea;
input clkb;
input oceb;
input ceb;
input resetb;
input wreb;
input [0:0] ada;
input [0:0] dina;
input [0:0] adb;
input [0:0] dinb;

wire gw_gnd;

assign gw_gnd = 1'b0;

DP bram_dp_0 (
    .DOA(douta[0]),
    .DOB(doutb[0]),
    .CLKA(clka),
    .OCEA(ocea),
    .CEA(cea),
    .RESETA(reseta),
    .WREA(wrea),
    .CLKB(clkb),
    .OCEB(oceb),
    .CEB(ceb),
    .RESETB(resetb),
    .WREB(wreb),
    .BLKSEL({gw_gnd,gw_gnd,gw_gnd}),
    .ADA({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,ada[0]}),
    .DIA(dina[0]),
    .ADB({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,adb[0]}),
    .DIB(dinb[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 = 1;
defparam bram_dp_0.BIT_WIDTH_1 = 1;
defparam bram_dp_0.BLK_SEL = 3'b000;
defparam bram_dp_0.RESET_MODE = "SYNC";

endmodule //Gowin_DP
```

Instantiation Template File for the IP Design File

For the practical use, the IP Core Generator generates the template file while generating the DP design file instantiation, as shown in Figure 3--20.

Figure 3--20 Instantiation Template File for the IP Design File

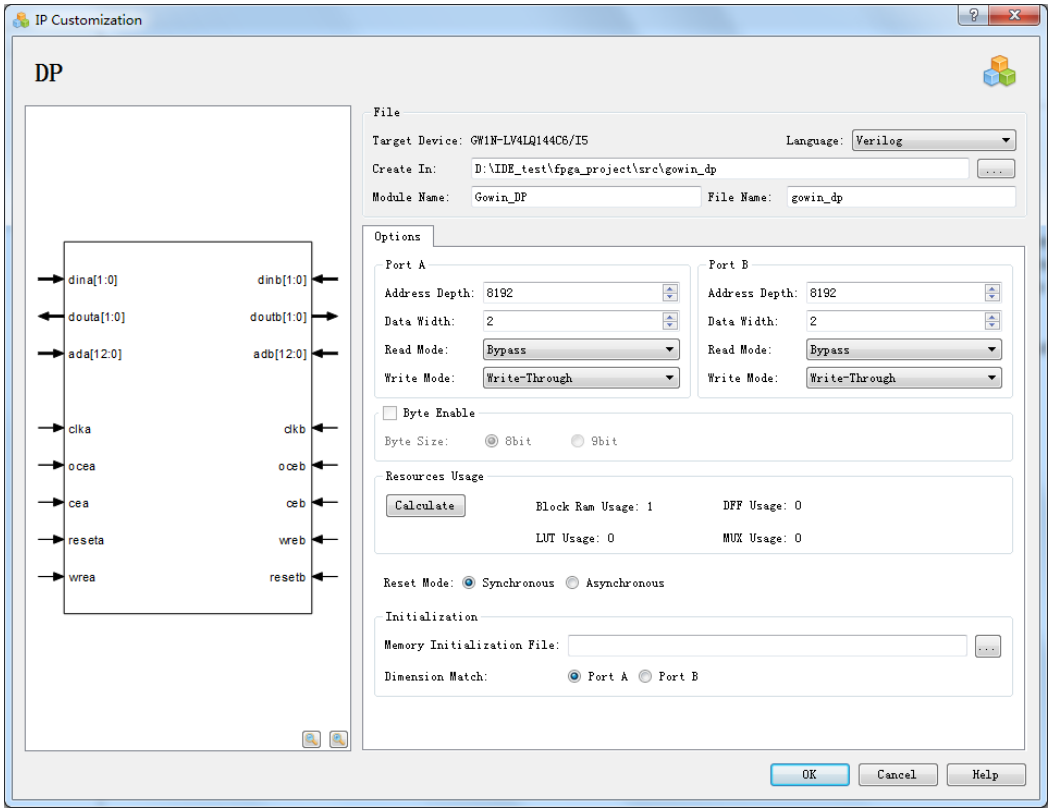
```
Gowin_DP your_instance_name(
     .douta(douta_o), //output [0:0] douta
     .doutb(doutb_o), //output [0:0] doutb
     .clka(clka_i), //input clka
     .ocea(ocea_i), //input ocea
     .cea(cea_i), //input cea
     .reseta(reseta_i), //input reseta
     .wrea(wrea_i), //input wrea
     .clkb(clkb_i), //input clkb
     .oceb(oceb_i), //input oceb
     .ceb(ceb_i), //input ceb
     .resetb(resetb_i), //input resetb
     .wreb(wreb_i), //input wreb
     .ada(ada_i), //input [9:0] ada
     .dina(dina_i), //input [0:0] dina
     .adb(adb_i), //input [9:0] adb
     .dinb(dinb_i) //input [0:0] dinb
);
```

DP Generation Example

If the users need to generate a specific DP IP as follows: 8192 x 2, Bypass read mode, write-through write mode and synchronous reset. Take the GW1N-LV4LQ144C6/I5 device for instance, the configuration interface is as shown in Figure 3-21. Select a memory initialization file for the module as required, and then click "OK" to generate the customized DP IP design files.

The directory where the DP IP design file is generated is the path set in "Create In".

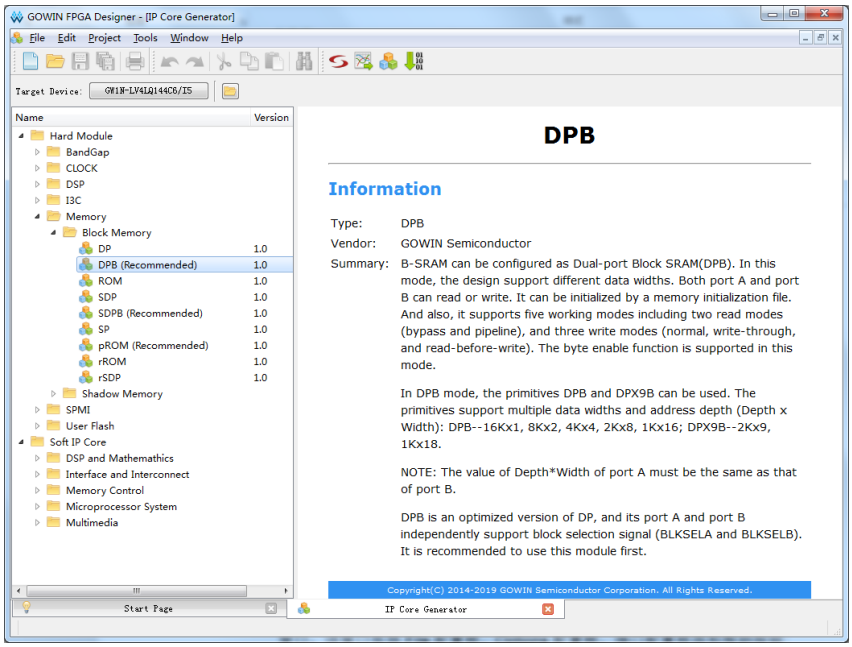
Figure 3-21 DP IP Customization Configuration



3.1.3 DPB

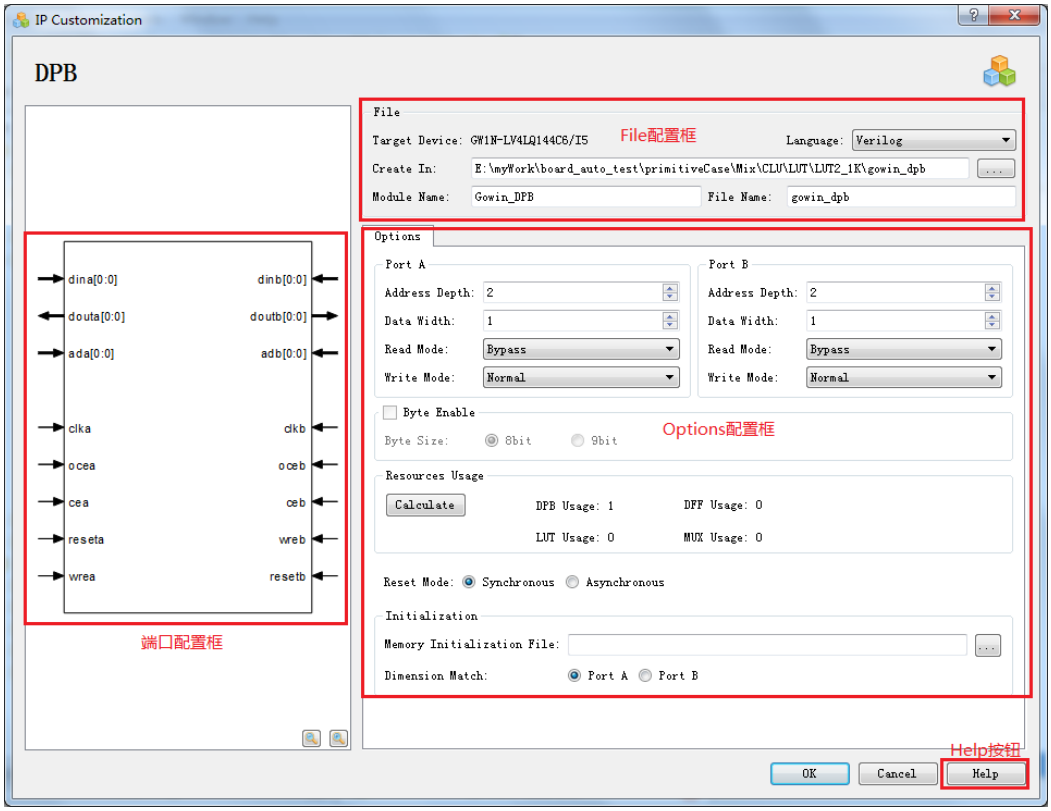
DPB is the dual-port block memory, which can be implemented by DPB and DPBX9. DPB is an optimized version of DP. Port A and port B support chip selection signals independently (BLKSELA and BLKSELB). Click "DPB" on the IP Core Generator interface. A brief introduction to the DPB will be displayed on the right of the screen, as shown in Figure 3-22.

Figure 3-22 DP Summary Information



Double click on the "DPB" to open the "IP Customization" window. This includes the "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button, as shown in Figure 3-23.

Figure 3-23 IP Customization of DPB



1. File Configuration

The file configuration includes the basic information related to the DPB instantiation file, as shown in Figure 3-23.

The DPB file configuration is similar to that of SP. For the detailed configuration, please refer to 3.1Block Memory >3.1.1SP > File Configuration.

2. Options Configuration

The Options configuration box is used to configure dual-port block memory by users, as shown in Figure 3-23.

DPB Options configuration is similar to that of SP. For the detailed configuration information, please refer to 3.1 Block Memory≥3.1.1SP > Options Configuration.

The notice of configuring DPB is similar to that of DP, please refer to Options configuration box in 3.1Block Memory > 3.1.2 DP.

3. Ports Configuration Diagram

- The ports configuration diagram displays the current IP Core configuration result. Input/Output bit-width updates in real time based on the "Options" configuration, as shown in Figure 3-23.
- "Address Depth" of Port A and Port B affects the bit-width of ada and adb; "Data Width" affects the bit-width of dia/doa and dib/dob.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-24.

Figure 3-24 Help

DPB

Information

Type:	DPB
Vendor:	GOWIN Semiconductor
Summary:	B-SRAM can be configured as Dual-port Block SRAM(DPB). In this mode, the design support different data widths. Both port A and port B can read or write. It can be initialized by a memory initialization file. And also, it supports five working modes including two read modes (bypass and pipeline), and three write modes (normal, write-through, and read-before-write). The byte enable function is supported in this mode. In DPB mode, the primitives DPB and DPX9B can be used. The primitives support multiple data widths and address depth (Depth x Width): DPB--16Kx1, 8Kx2, 4Kx4, 2Kx8, 1Kx16; DPX9B--2Kx9, 1Kx18. NOTE: The value of Depth*Width of port A must be the same as that of port B. DPB is an optimized version of DP, and its port A and port B independently support block selection signal (BLKSELA and BLKSELB). It is recommended to use this module first.

Options

Option	Description
Port A	**Address Depth** - Set the size of the address depth.
	Data Width - Set the size of the Data width.
	Read Mode - Set whether the read mode is bypass mode or pipeline mode.
	Write Mode - Set the write mode as normal mode, write-through mode or read-before-write mode.
Port B	**Address Depth** - Set the size of the address depth.
	Data Width - Set the size of the Data width.
	Read Mode - Set whether the read mode is bypass mode or pipeline mode.
	Write Mode - Set the write mode as normal mode, write-through mode or read-before-write mode.
Byte Enable	**Byte Enable** - Set whether to use byte enable function or not.
	Byte Size - Set whether the byte size is 8bit or 9bit if the byte enable checkbox selected.
	Note:(1) If Width<=8, byte enable function is invalid; (2) If Width=9, only 8 bit is valid; (3) If Width>9, both 8 bit and 9 bit are valid.
Resource Usage	**Calculate** - Calculate the resource usage in the design and display results below.
	DPB Usage - Display the number of DPB used.
	DFF Usage - Display the number of DFF used.
	LUT Usage - Display the number of LUT used.

The "Help" page contains a general description of the IP Core, and a brief introduction to the "Options".

IP Generation Files

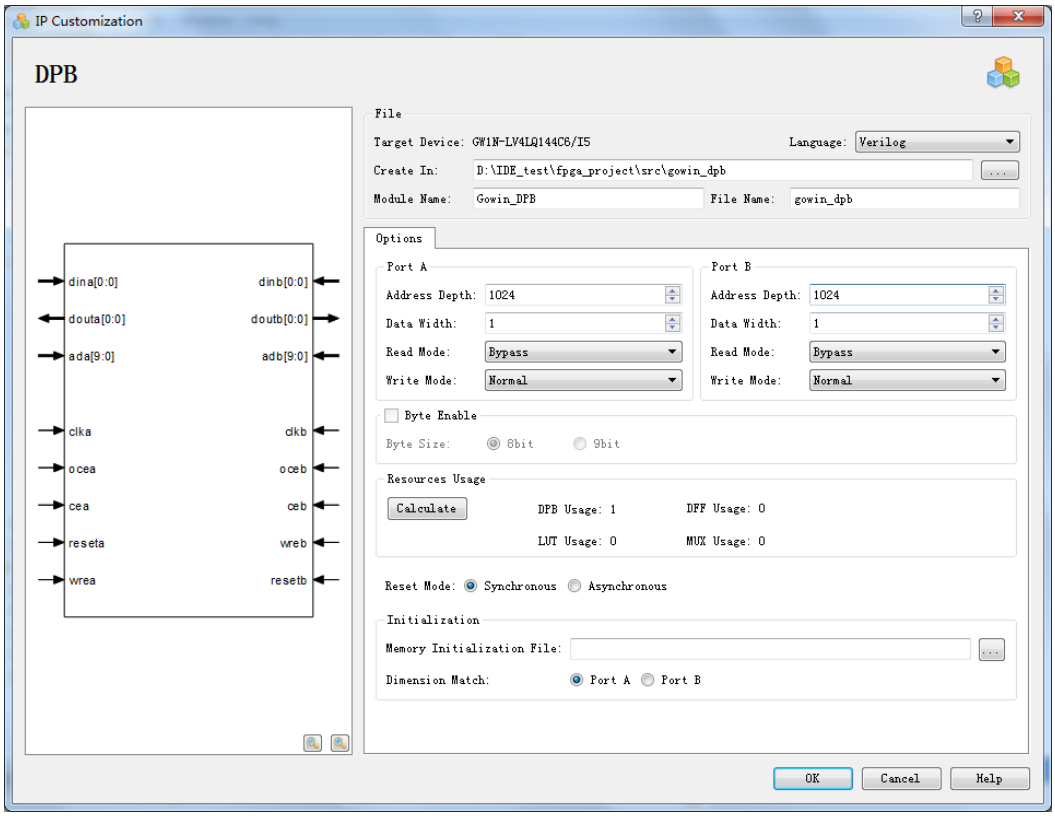
As shown in Figure 3-25, after customizing the DPB IP, click "OK" to

generate three files that are named after the "File Name":

- Instantiate Gowin Primitive DPB design file "gowin_dpb.v";
- The instantiation template file for the IP design file "gowin_dpb_tmp.v";
- The configuration files for the Primitive DPB instantiation "gowin_dpb.ipc".

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.

Figure 3-25 Configured IP Customization



DP Design File Instantiation

DPB design file instantiation is a complete Verilog module. DPB instantiation is generated according to the DPB configuration that is displayed in the "IP Customization" window, as shown in Figure 3-26.

Figure 3-26 DPB Design File Instantiation

```
module Gowin_DPB (douta, doutb, clka, ocea, cea, reseta, wrea, clkb, oceb, ceb, resetb, wreb, ada, dina, adb, dinb);

output [0:0] douta;
output [0:0] doutb;
input clka;
input ocea;
input cea;
input reseta;
input wrea;
input clkb;
input oceb;
input ceb;
input resetb;
input wreb;
input [9:0] ada;
input [0:0] dina;
input [9:0] adb;
input [0:0] dinb;

wire gw_gnd;

assign gw_gnd = 1'b0;

DPB dpb_inst_0 (
    .DOA(douta[0]),
    .DOB(doutb[0]),
    .CLKA(clka),
    .OCEA(ocea),
    .CEA(cea),
    .RESETA(reseta),
    .WREA(wrea),
    .CLKB(clkb),
    .OCEB(oceb),
    .CEB(ceb),
    .RESETB(resetb),
    .WREB(wreb),
    .BLKSELA({gw_gnd,gw_gnd,gw_gnd}),
    .BLKSELB({gw_gnd,gw_gnd,gw_gnd}),
    .ADA({gw_gnd,gw_gnd,gw_gnd,gw_gnd,ada[9:0]}),
    .DIA(dina[0]),
    .ADB({gw_gnd,gw_gnd,gw_gnd,gw_gnd,adb[9:0]}),
    .DIB(dinb[0])
);

defparam dpb_inst_0.READ_MODE0 = 1'b0;
defparam dpb_inst_0.READ_MODE1 = 1'b0;
defparam dpb_inst_0.WRITE_MODE0 = 2'b00;
defparam dpb_inst_0.WRITE_MODE1 = 2'b00;
defparam dpb_inst_0.BIT_WIDTH_0 = 1;
defparam dpb_inst_0.BIT_WIDTH_1 = 1;
defparam dpb_inst_0.BLK_SEL_0 = 3'b000;
defparam dpb_inst_0.BLK_SEL_1 = 3'b000;
defparam dpb_inst_0.RESET_MODE = "SYNC";

endmodule //Gowin_DPB
```

Instantiation Template File for the IP Design File

For the practical use, the IP Core Generator generates the template file while generating the DPB design file instantiation, as shown in Figure 3-27.

Figure 3-27 Instantiation Template File for the IP Design File

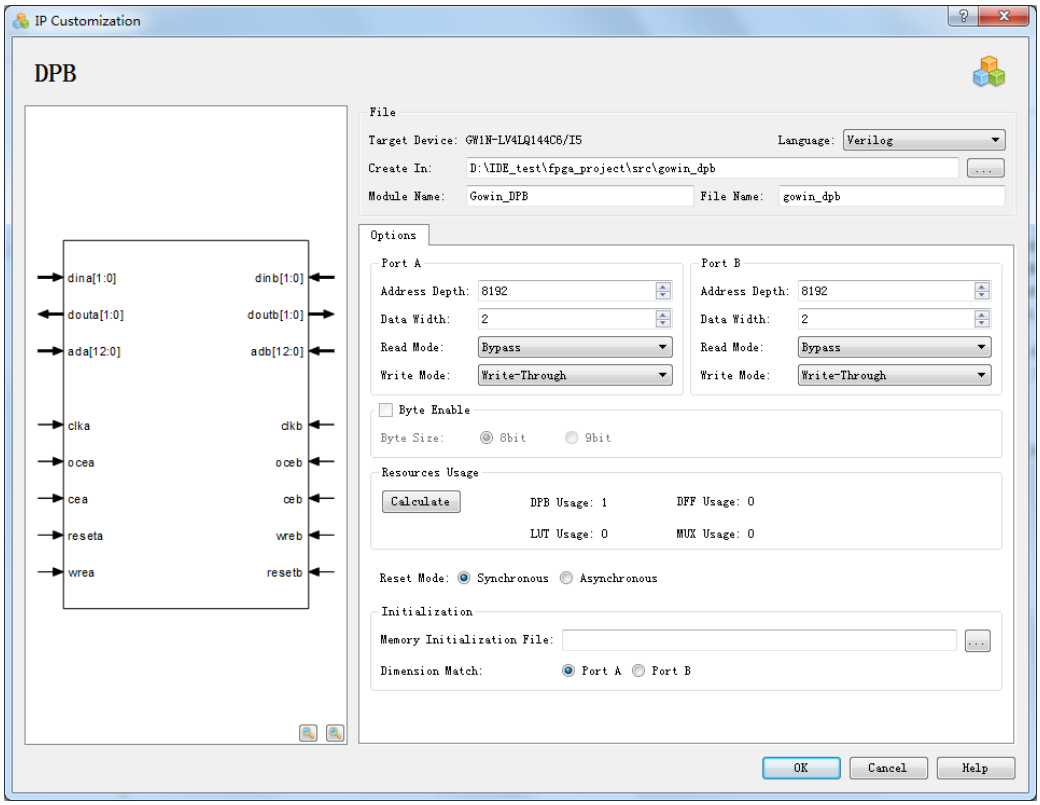
```
Gowin_DPB your_instance_name(
    .douta(douta_o), //output [0:0] douta
    .doutb(doutb_o), //output [0:0] doutb
    .clka(clka_i), //input clka
    .ocea(ocea_i), //input ocea
    .cea(cea_i), //input cea
    .reseta(reseta_i), //input reseta
    .wrea(wrea_i), //input wrea
    .clkb(clkb_i), //input clkb
    .oceb(oceb_i), //input oceb
    .ceb(ceb_i), //input ceb
    .resetb(resetb_i), //input resetb
    .wreb(wreb_i), //input wreb
    .ada(ada_i), //input [9:0] ada
    .dina(dina_i), //input [0:0] dina
    .adb(adb_i), //input [9:0] adb
    .dinb(dinb_i) //input [0:0] dinb
);
```

DPB Generation Example

If users need to generate a specific DPB IP as follows: 8192 x 2, Bypass read mode, write-through write mode and synchronous reset. Take the GW1N-LV4LQ144C6/I5 device for instance, the configuration interface is as shown in Figure 3-28. Select a memory initialization file for the module as required, and then click "OK" to generate the customized DPB IP design files.

The directory where the DPB IP design file is generated is the path set in "Create In".

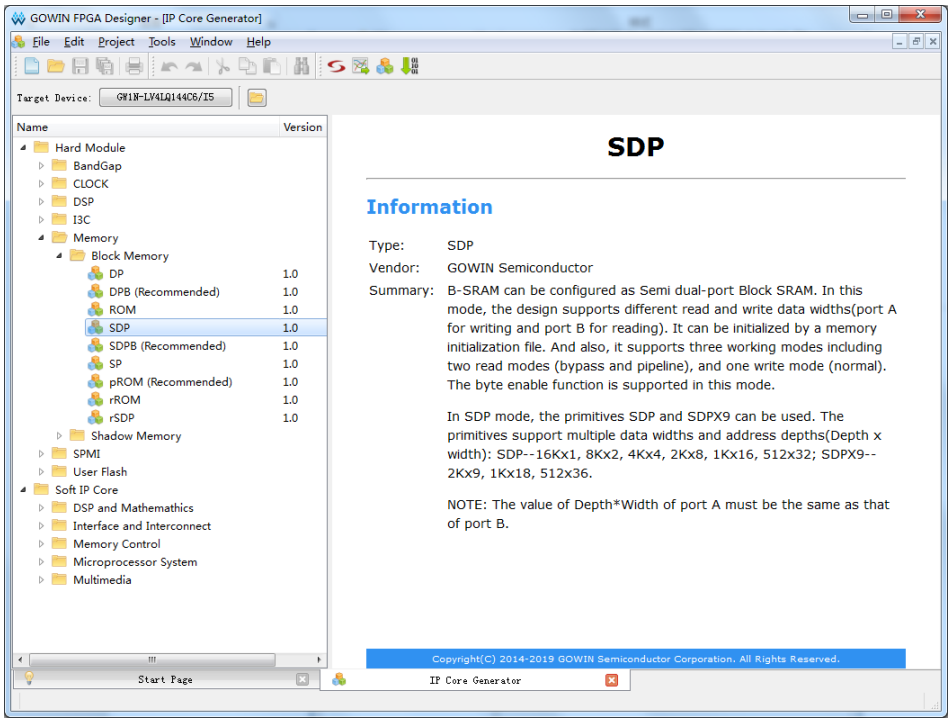
Figure 3-28 DPB IP Customization Configuration



3.1.4 SDP

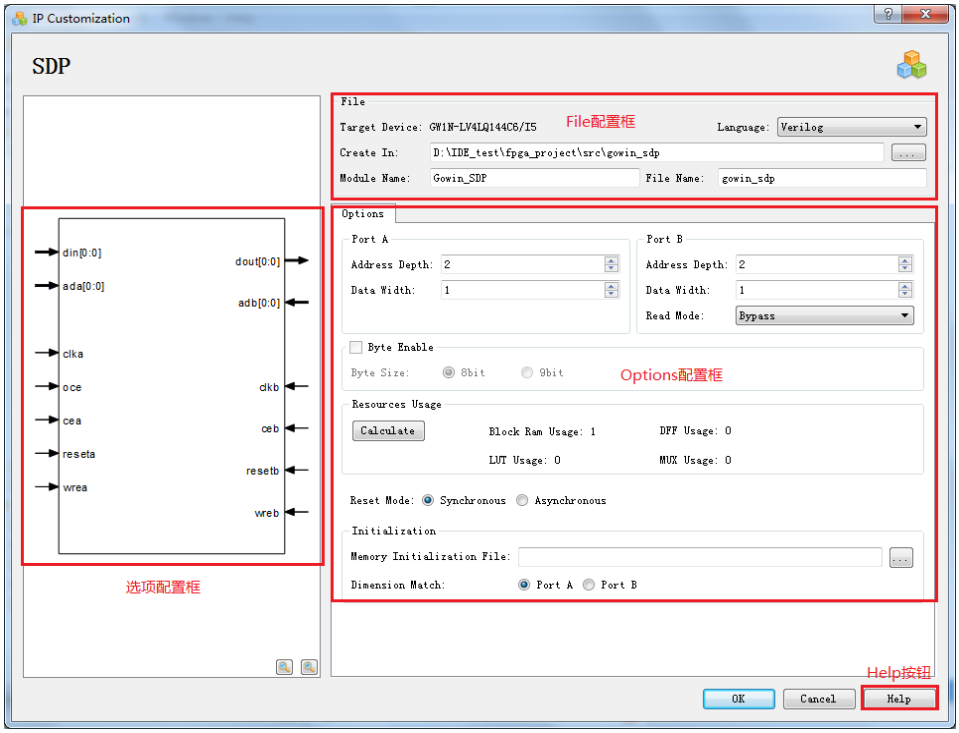
SDP is a Semi-dual Port Block Memory that can be implemented by SDP and SDPX9. The maximum capacity of BSRAM varies according to the type of chip. Click "SDP" on the IP Core Generator interface. A brief introduction to the SDP will be displayed on the right of the screen, as shown in Figure 3-29.

Figure 3-29 SDP Summary Information



Double click on the "SDP" to open the "IP Customization" window. This includes the "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button, as shown in Figure 3-30.

Figure 3-30 IP Customization of SDP



1. File Configuration

File configuration includes the basic information related to the SDP instantiation file, as shown in Figure 3-30.

The SDP file configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP> File Configuration.

2. Options Configuration

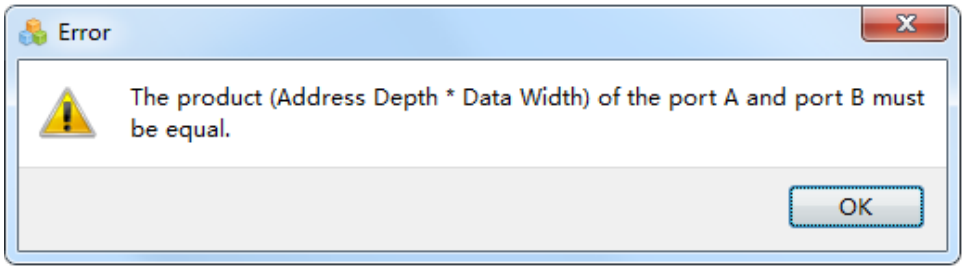
The Options configuration box is used to configure semi-dual port block memory by users, as shown in Figure 3-30.

SDP options configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP> Options Configuration.

Note!

- SDP only supports Port A write operation and Port B read operation; Read Mode configuration box can configure Port B Read Mode to be Bypass or Pipeline;
- The address depth and data width of SDP Port A and Port B can be configured independently.
- The address depth and data width of SDP Port A and Port B must be equal because Port A and Port B read from or write to the same memory. If not, Error message as shown in Figure 3-31 will pop up.
- The date width in Memory initialization File should be consistent with the data width of the port selected by "Dimension Match". If not, the Init. value of generated SDP instantiation is 0 by default, and Error message as follows will pop up:
  Error (MG2105): Initial value's width is unequal to user's width option.

Figure 3-31 SDP Configuration Error



3. Ports Configuration Diagram
   - The ports configuration diagram displays the current IP Core configuration result. Input/Output bit-width updates in real time based on the "Options" configuration, as shown in Figure 3-30.
   - Port A "Address Depth" affects the bit-width of ada, and Port A "Data Width" affects the bit-width of din; Port B "Address Depth" affects the bit-width of adb, and Port B "Data Width" affects the bit-width of dout.
4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-32.

Figure 3-32 Help

SDP

Information

Type:	SDP
Vendor:	GOWIN Semiconductor
Summary:	B-SRAM can be configured as Semi dual-port Block SRAM. In this mode, the design supports different read and write data widths(port A for writing and port B for reading). It can be initialized by a memory initialization file. And also, it supports three working modes including two read modes (bypass and pipeline), and one write mode (normal). The byte enable function is supported in this mode. In SDP mode, the primitives SDP and SDPX9 can be used. The primitives support multiple data widths and address depths(Depth x width): SDP--16Kx1, 8Kx2, 4Kx4, 2Kx8, 1Kx16, 512x32; SDPX9--2Kx9, 1Kx18, 512x36. NOTE: The value of Depth*Width of port A must be the same as that of port B.

Options

Option	Description
Port A	**Address Depth** - Set the size of the address depth.
	Data Width - Set the size of the Data width.
Port B	**Address Depth** - Set the size of the address depth.
	Data Width - Set the size of the Data width.
	Read Mode - Set whether the read mode is bypass mode or pipeline mode.
Byte Enable	**Byte Enable** - Set whether to use byte enable function or not.
	Byte Size - Set whether the byte size is 8bit or 9bit if the byte enable checkbox selected.
	Note: Assume that the data width is represented by Width. (1) If Width<=8, byte enable function is invalid; (2) If Width=9, only 8 bit is valid; (3) If Width>9, both 8 bit and 9 bit are valid.
Resource Usage	**Calculate** - Calculate the resource usage in the design and display results below.
	Block Ram Usage - Display the number of Block Ram used.
	DFF Usage - Display the number of DFF used.
	LUT Usage - Display the number of LUT used.
	MUX Usage - Display the number of MUX used.
Reset Mode	**Reset Mode** - Set whether the reset mode is synchronous mode or asynchronous mode.
Initialization	**Memory Initialization File** - Set the memory initialization file (.mi) path.
	Dimension Match - Set which port's dimensions the memory initialization file should conform to.

The "Help" page contains a general description of the IP Core, and a brief introduction to the "Options".

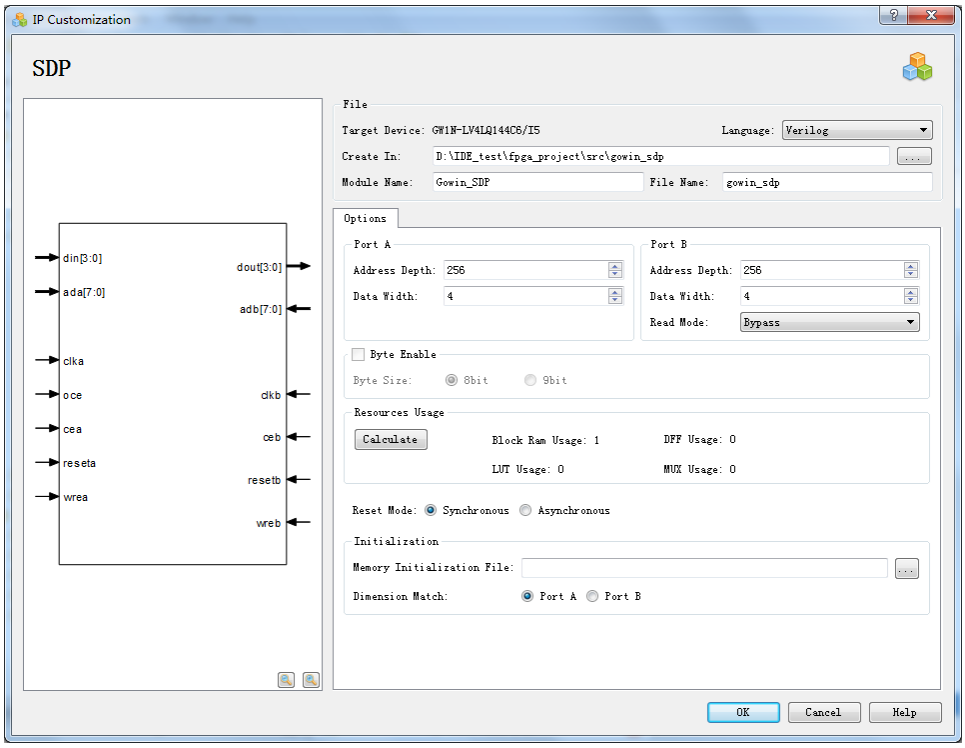
IP Generation Files

As shown in Figure 3-33, after customizing the SDP IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate Gowin Primitive SDP design file "gowin_sdp.v";
- The instantiation template file for the IP design file "gowin_sdp_tmp.v";
- The configuration files for the Primitive SPB instantiation "gowin_sdp.ipc".

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.

Figure 3-33 Configured IP Customization



SDP Design File Instantiation

SDP design file instantiation is a complete Verilog module. SDP instantiation is generated according to the SDP configuration that is displayed in the "IP Customization" window, as shown in Figure 3-34.

Note!

Din/Dout data width of the generated SDP instantiation is consistent with that of the SDP configured in the "IP Customization" window.

Figure 3-34 SDP Design File Instantiation

```
module Gowin_SDP (dout, clka, cea, reseta, wrea, clkb, ceb, resetb, wreb, oce, ada, din, adb);

output [3:0] dout;
input clka;
input cea;
input reseta;
input wrea;
input clkb;
input ceb;
input resetb;
input wreb;
input oce;
input [7:0] ada;
input [3:0] din;
input [7:0] adb;

wire gw_gnd;

assign gw_gnd = 1'b0;

SDP bram_sdp_0 (
    .DO(dout[3:0]),
    .CLKA(clka),
    .CEA(cea),
    .RESETA(reseta),
    .WREA(wrea),
    .CLKB(clkb),
    .CEB(ceb),
    .RESETB(resetb),
    .WREB(wreb),
    .OCE(oce),
    .BLKSEL({gw_gnd,gw_gnd,gw_gnd}),
    .ADA({gw_gnd,gw_gnd,gw_gnd,gw_gnd,ada[7:0],gw_gnd,gw_gnd}),
    .DI(din[3:0]),
    .ADB({gw_gnd,gw_gnd,gw_gnd,gw_gnd,adb[7:0],gw_gnd,gw_gnd})
);

defparam bram_sdp_0.READ_MODE = 1'b0;
defparam bram_sdp_0.BIT_WIDTH_0 = 4;
defparam bram_sdp_0.BIT_WIDTH_1 = 4;
defparam bram_sdp_0.BLK_SEL = 3'b000;
defparam bram_sdp_0.RESET_MODE = "SYNC";

endmodule //Gowin_SDP
```

Instantiation Template File for the IP Design File

For the practical use, the IP Core Generator generates the template file while generating the SPB design file instantiation, as shown in Figure 3-35.

Figure 3-35 Instantiation Template File for the IP Design File

```
Gowin_SDP your_instance_name(
    .dout(dout_o), //output [3:0] dout
    .clka(clka_i), //input clka
    .cea(cea_i), //input cea
    .reseta(reseta_i), //input reseta
    .wrea(wrea_i), //input wrea
    .clkb(clkb_i), //input clkb
    .ceb(ceb_i), //input ceb
    .resetb(resetb_i), //input resetb
    .wreb(wreb_i), //input wreb
    .oce(oce_i), //input oce
    .ada(ada_i), //input [7:0] ada
    .din(din_i), //input [3:0] din
    .adb(adb_i) //input [7:0] adb
);
```

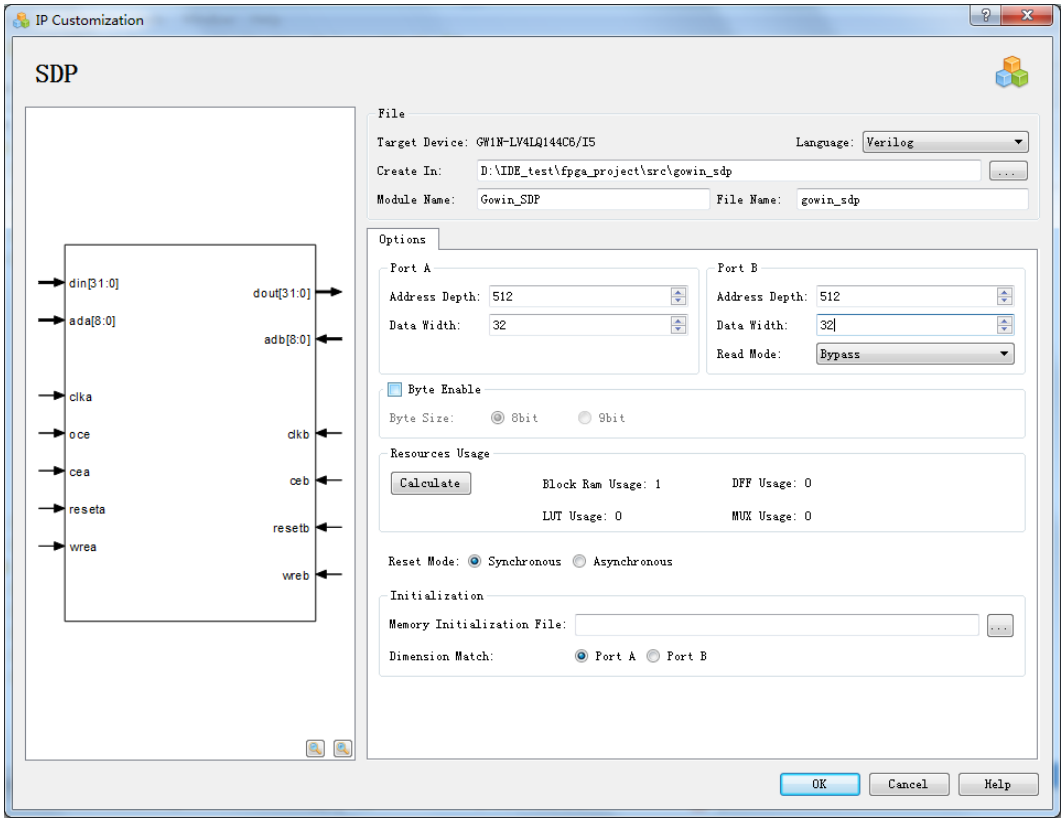
SDP Generation Example

If users need to generate a specific SDP IP as follows: 512 x 32,

Bypass read mode and synchronous reset.Take the GW1N-LV4LQ144C6/I5 device for instance, the configuration page is as shown in Figure 3-36. Select a memory initialization file for the module as required, and then click "OK" to generate the customized SDP IP design files.

The directory where the SPB IP design file is generated is the path set in "Create In".

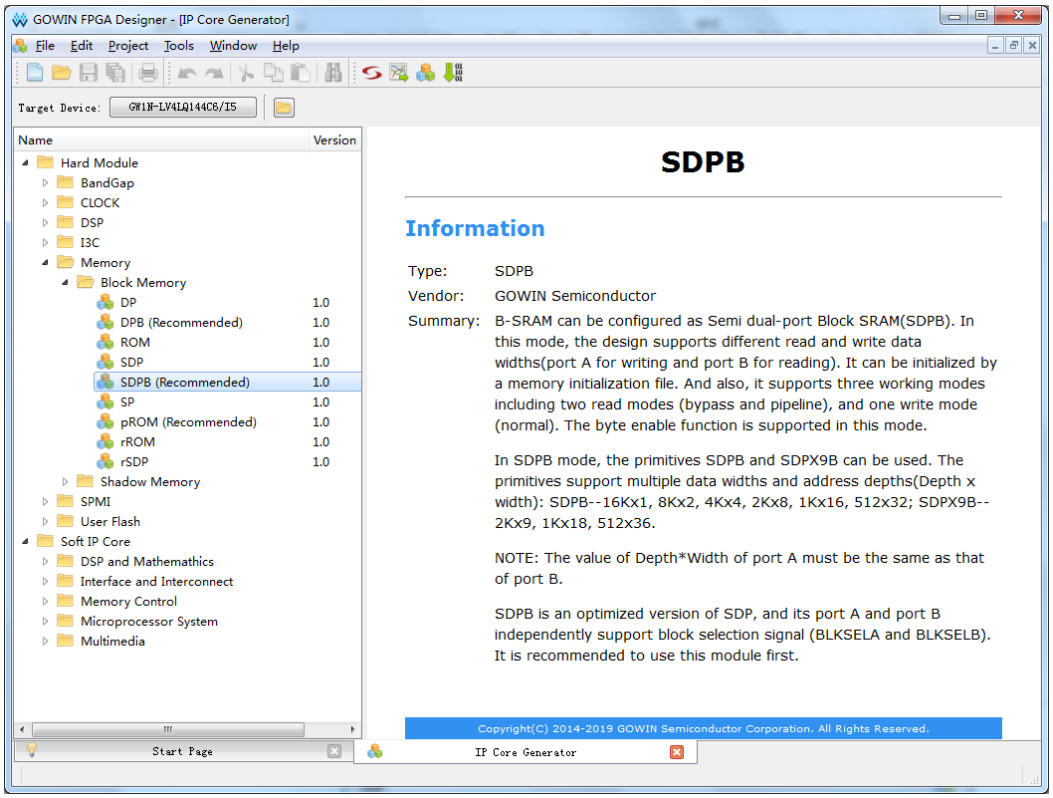
Figure 3-36 IP Customization of SDP Configuration



3.1.5 SDPB

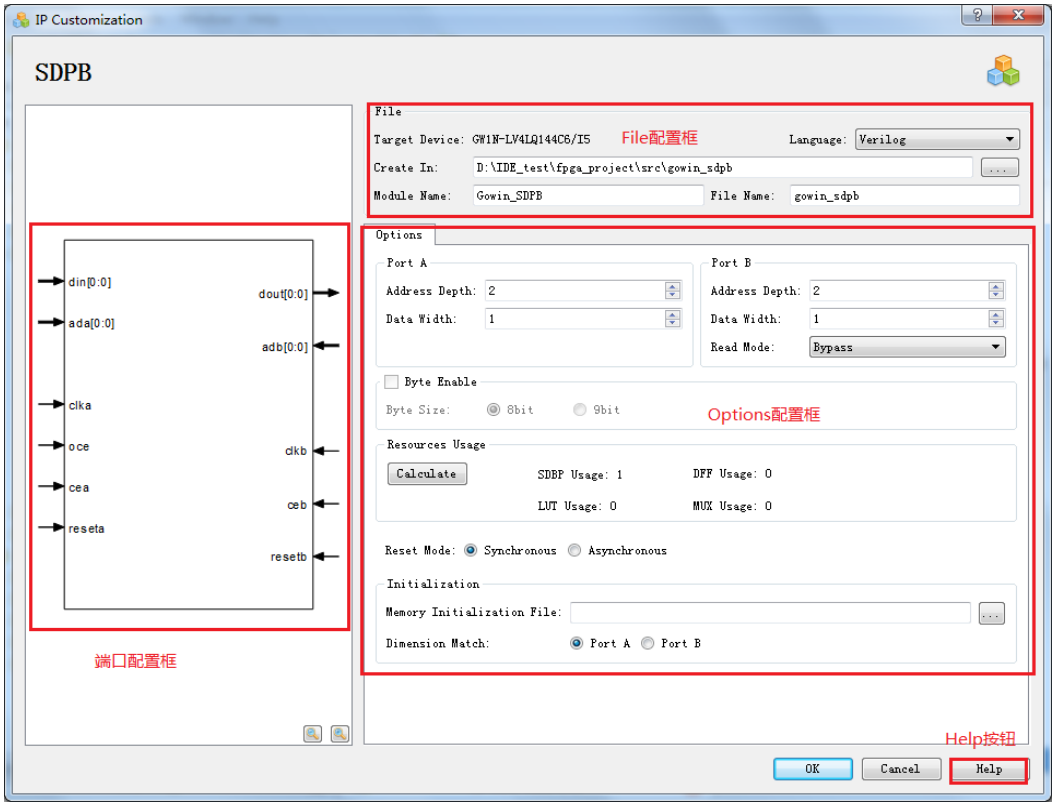
DPB is the semi-dual port block memory, which can be implemented by SDPB and SDPBX9. SDPB is an optimized version of SDP. Port A and port B support chip selection signals independently (BLKSELA and BLKSELB). Click "SDPB" on the IP Core Generator interface. A brief introduction to the SDPB will be displayed on the right of the screen, as shown in Figure 3-37.

Figure 3-37 SDPB Summary Information



Double click on the "SDPB" to open the "IP Customization" window. This includes the "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button, as shown in Figure 3-38.

Figure 3-38 IP Customization of SDPB



1. File Configuration

The file configuration includes the basic information related to the SDPB instantiation file, as shown in Figure 3-38.

The SDPB file configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP> File Configuration.

2. Options Configuration

The Options configuration box is used to configure semi-dual port block memory by users, as shown in Figure 3-38.

SDPB options configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP> Options Configuration.

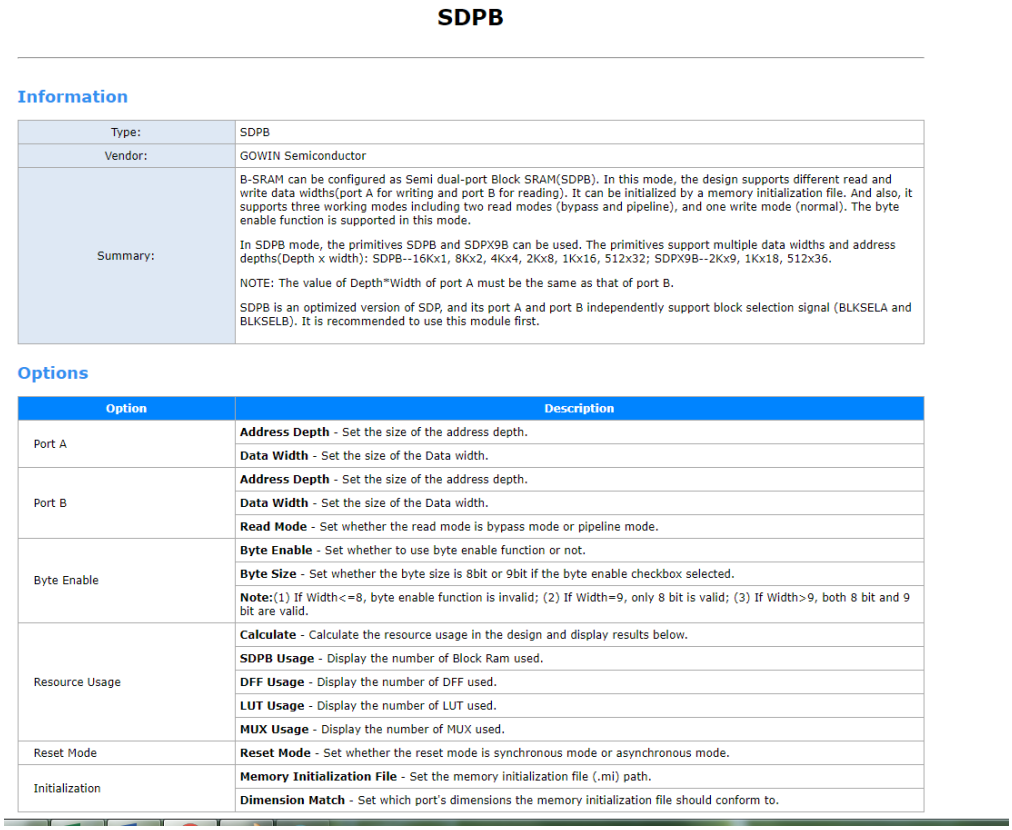
Note!

The notice of configuring SDPB is similar to that of SP, please refer to the Options configuration box in 3.1Block Memory > 3.1.4 SDP.

3. Ports Configuration Diagram
   - The ports configuration diagram displays the current IP Core configuration result. Input/Output bit-width updates in real time based on the "Options" configuration, as shown in Figure 3-38.
   - Port A "Address Depth" affects the bit-width of ada, and Port A "Data Width" affects the bit-width of din; Port B "Address Depth" affects the bit-width of adb, and Port B "Data Width" affects the bit-width of dout.
4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-39.

Figure 3-39 Help



SDPB

Information

Type:	SDPB
Vendor:	GOWIN Semiconductor
Summary:	B-SRAM can be configured as Semi dual-port Block SRAM(SDPB). In this mode, the design supports different read and write data widths(port A for writing and port B for reading). It can be initialized by a memory initialization file. And also, it supports three working modes including two read modes (bypass and pipeline), and one write mode (normal). The byte enable function is supported in this mode. In SDPB mode, the primitives SDPB and SDPX9B can be used. The primitives support multiple data widths and address depths(Depth x width): SDPB--16Kx1, 8Kx2, 4Kx4, 2Kx8, 1Kx16, 512x32; SDPX9B--2Kx9, 1Kx18, 512x36. NOTE: The value of Depth*Width of port A must be the same as that of port B. SDPB is an optimized version of SDP, and its port A and port B independently support block selection signal (BLKSELA and BLKSELB). It is recommended to use this module first.

Options

Option	Description
Port A	**Address Depth** - Set the size of the address depth.
	Data Width - Set the size of the Data width.
Port B	**Address Depth** - Set the size of the address depth.
	Data Width - Set the size of the Data width.
	Read Mode - Set whether the read mode is bypass mode or pipeline mode.
Byte Enable	**Byte Enable** - Set whether to use byte enable function or not.
	Byte Size - Set whether the byte size is 8bit or 9bit if the byte enable checkbox selected.
	Note:(1) If Width<=8, byte enable function is invalid; (2) If Width=9, only 8 bit is valid; (3) If Width>9, both 8 bit and 9 bit are valid.
Resource Usage	**Calculate** - Calculate the resource usage in the design and display results below.
	SDPB Usage - Display the number of Block Ram used.
	DFF Usage - Display the number of DFF used.
	LUT Usage - Display the number of LUT used.
	MUX Usage - Display the number of MUX used.
Reset Mode	**Reset Mode** - Set whether the reset mode is synchronous mode or asynchronous mode.
Initialization	**Memory Initialization File** - Set the memory initialization file (.mi) path.
	Dimension Match - Set which port's dimensions the memory initialization file should conform to.

The "Help" page contains a general description of the IP Core, and a brief introduction to the "Options".

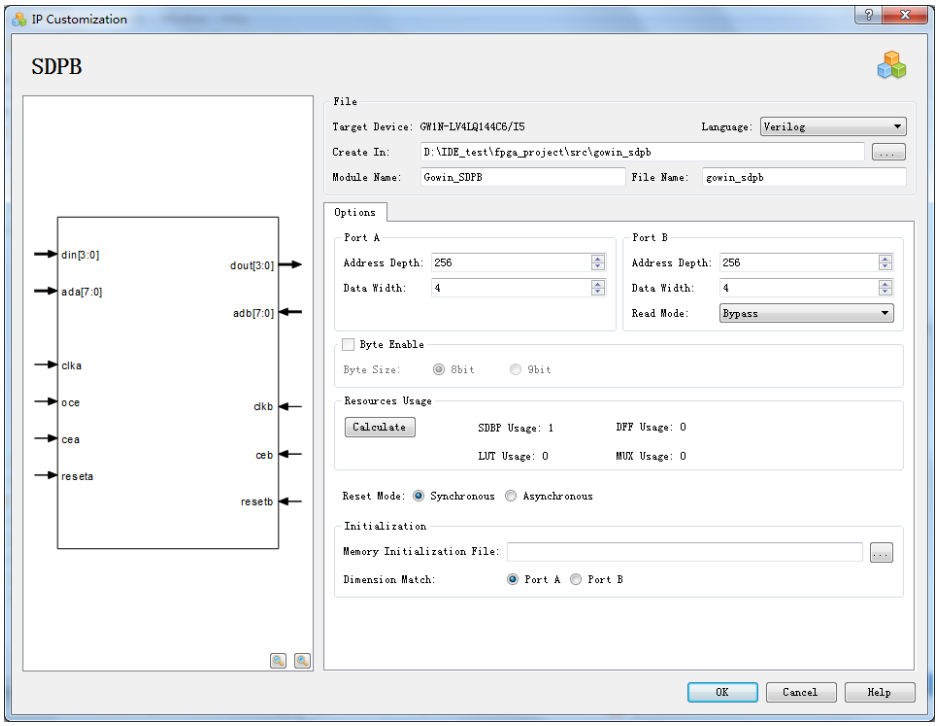
IP Generation Files

As shown in Figure 3-40, after customizing the SDPB IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate Gowin Primitive SDPB design file "gowin_sdpb.v";
- The instantiation template file for the IP design file "gowin_sdpb_tmp.v";
- The configuration files for the Primitive SPB instantiation "gowin_sdpb.ipc".

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.

Figure 3-40 Configured IP Customization



SDPB Design File Instantiation

SDPB design file instantiation is a complete Verilog module. SDP instantiation is generated according to the SDP configuration that is displayed in the "IP Customization" window, as shown in Figure 3-41.

Note!

Din/Dout data width of the generated SDPB instantiation is consistent with that of the SDPB configured in the "IP Customization" window.

Figure 3-41 SDPB Design File Instantiation

```
module Gowin_SDPB (dout, clka, cea, reseta, clkb, ceb, resetb, oce, ada, din, adb);

output [3:0] dout;
input clka;
input cea;
input reseta;
input clkb;
input ceb;
input resetb;
input oce;
input [7:0] ada;
input [3:0] din;
input [7:0] adb;

wire gw_gnd;

assign gw_gnd = 1'b0;

SDPB sdpb_inst_0 (
    .DO(dout[3:0]),
    .CLKA(clka),
    .CEA(cea),
    .RESETA(reseta),
    .CLKB(clkb),
    .CEB(ceb),
    .RESETB(resetb),
    .OCE(oce),
    .BLKSELA({gw_gnd,gw_gnd,gw_gnd}),
    .BLKSELB({gw_gnd,gw_gnd,gw_gnd}),
    .ADA({gw_gnd,gw_gnd,gw_gnd,gw_gnd,ada[7:0],gw_gnd,gw_gnd}),
    .DI(din[3:0]),
    .ADB({gw_gnd,gw_gnd,gw_gnd,gw_gnd,adb[7:0],gw_gnd,gw_gnd})
);

defparam sdpb_inst_0.READ_MODE = 1'b0;
defparam sdpb_inst_0.BIT_WIDTH_0 = 4;
defparam sdpb_inst_0.BIT_WIDTH_1 = 4;
defparam sdpb_inst_0.BLK_SEL_0 = 3'b000;
defparam sdpb_inst_0.BLK_SEL_1 = 3'b000;
defparam sdpb_inst_0.RESET_MODE = "SYNC";

endmodule //Gowin_SDPB
```

Instantiation Template File for the IP Design File

For the practical use, the IP Core Generator generates the template file while generating the SDPB design file instantiation, as shown in Figure 3-42.

Figure 3-42 Instantiation Template File for the IP Design File

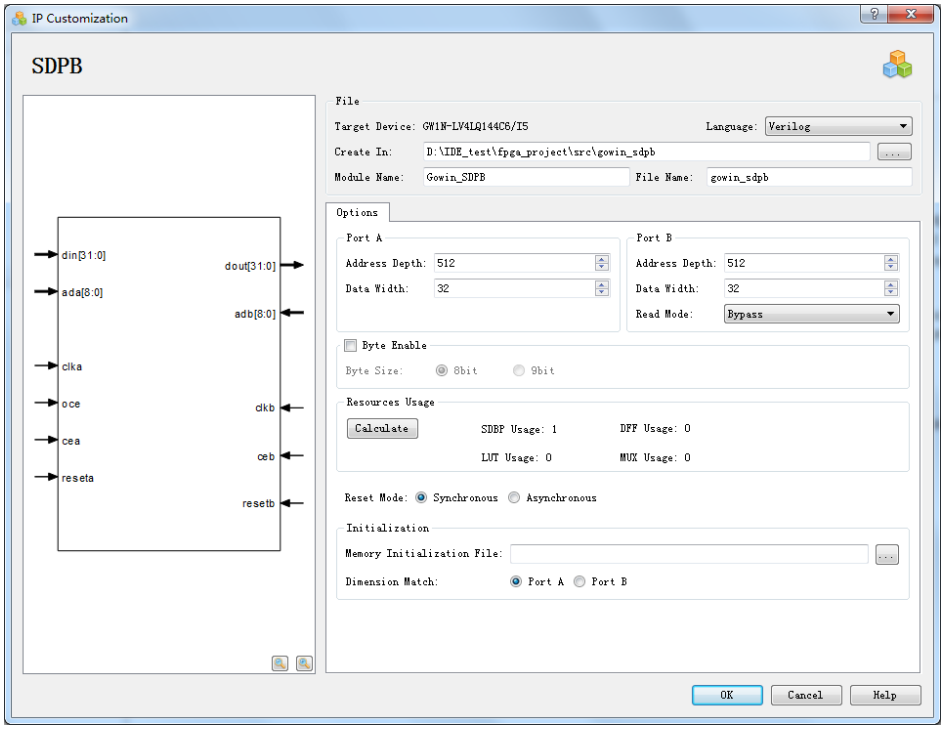
```
Gowin_SDPB your_instance_name(
    .dout(dout_o), //output [3:0] dout
    .clka(clka_i), //input clka
    .cea(cea_i), //input cea
    .reseta(reseta_i), //input reseta
    .clkb(clkb_i), //input clkb
    .ceb(ceb_i), //input ceb
    .resetb(resetb_i), //input resetb
    .oce(oce_i), //input oce
    .ada(ada_i), //input [7:0] ada
    .din(din_i), //input [3:0] din
    .adb(adb_i) //input [7:0] adb
);
```

SDPB Generation Example

If users need to generate a specific SDPB IP as follows: 512 x 32, Bypass read mode and synchronous reset.Take the GW1N-LV4LQ144C6/I5 device for instance, the configuration page is as shown in Figure 3-43. Select a memory initialization file for the module as required, and then click "OK" to generate the customized SDPB IP design files.

The directory where the SDPB IP design file is generated is the path set in "Create In".

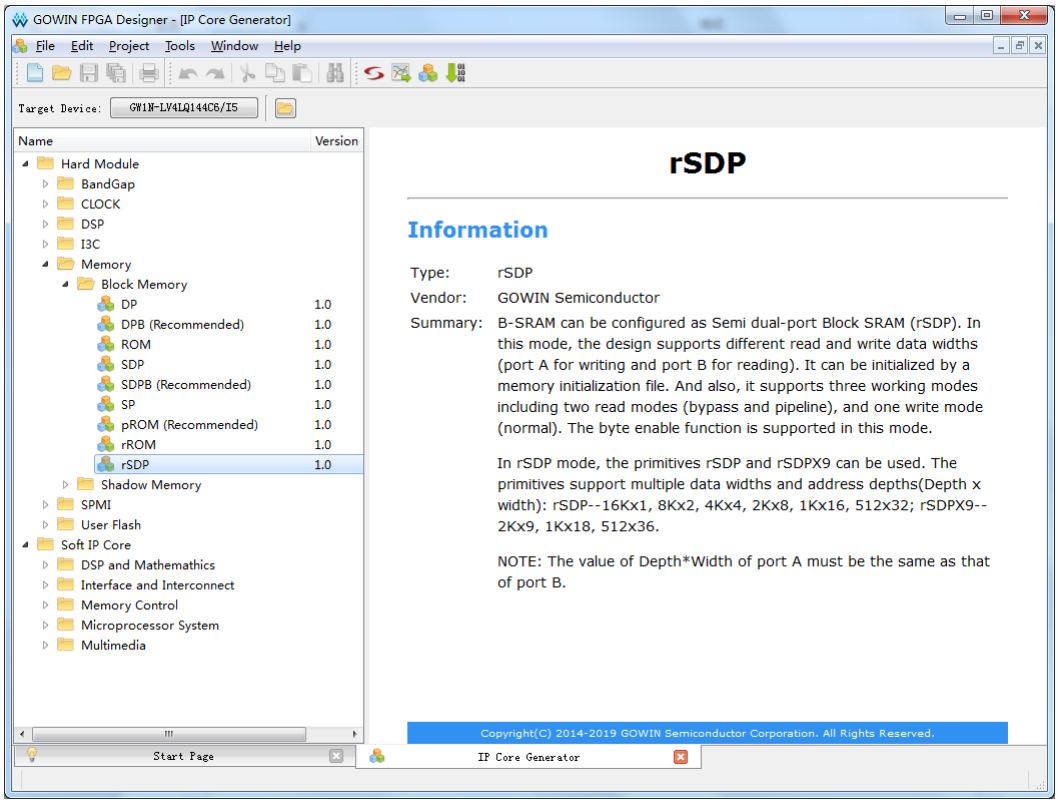
Figure 3-43 IP Customization of SDPB Configuration



3.1.6 rSDP

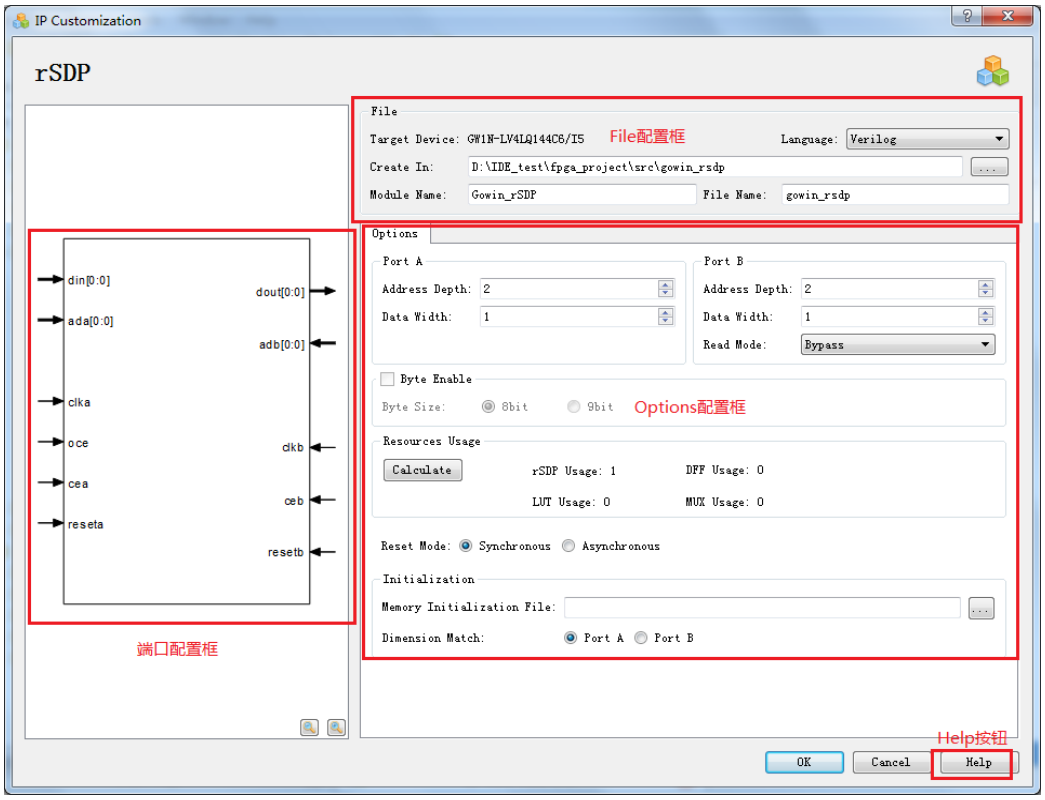
rSDP is the semi-dual port block memory, which can be implemented by rSDP and rSDPX9. Compared with SDP, rSDP removes WRE signal. Click "rSDP" on the IP Core Generator interface. A brief introduction to the rSDP will be displayed on the right of the screen, as shown in Figure 3-44.

Figure 3-44 rSDP Summary Information



Double click on the "rSDP" to open the "IP Customization" window. This includes the "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button, as shown in Figure 3-45.

Figure 3-45 IP Customization of rSDP



1. File Configuration

The file configuration includes the basic information related to the rSDP instantiation file, as shown in Figure 3-45.

The rSDP file configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP > File Configuration.

2. Options Configuration

The Options configuration box is used to configure semi-dual port block memory by users, as shown in Figure 3-45.

rSDP options configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP > Options Configuration.

Note!

The notice of configuring rSDP is similar to that of SP, please refer to the Options configuration box in 3.1Block Memory > 3.1.4 SDP.

3. Ports Configuration Diagram

- The ports configuration diagram displays the current IP Core configuration result. Input/Output bit-width updates in real time based on the "Options" configuration, as shown in Figure 3-45.
- Port A "Address Depth" affects the bit-width of ada, and Port A "Data Width" affects the bit-width of din; Port B "Address Depth" affects the bit-width of adb, and Port B "Data Width" affects the bit-width of dout.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-46.

Figure 3-46 Help

rSDP

Information

Type:	rSDP
Vendor:	GOWIN Semiconductor
Summary:	B-SRAM can be configured as Semi dual-port Block SRAM (rSDP). In this mode, the design supports different read and write data widths (port A for writing and port B for reading). It can be initialized by a memory initialization file. And also, it supports three working modes including two read modes (bypass and pipeline), and one write mode (normal). The byte enable function is supported in this mode. In rSDP mode, the primitives rSDP and rSDPX9 can be used. The primitives support multiple data widths and address depths(Depth x width): rSDP--16Kx1, 8Kx2, 4Kx4, 2Kx8, 1Kx16, 512x32; rSDPX9--2Kx9, 1Kx18, 512x36. NOTE: The value of Depth*Width of port A must be the same as that of port B.

Options

Option	Description
Port A	**Address Depth** - Set the size of the address depth.
	Data Width - Set the size of the Data width.
Port B	**Address Depth** - Set the size of the address depth.
	Data Width - Set the size of the Data width.
	Read Mode - Set whether the read mode is bypass mode or pipeline mode.
Byte Enable	**Byte Enable** - Set whether to use byte enable function or not.
	Byte Size - Set whether the byte size is 8bit or 9bit if the byte enable checkbox selected.
	Note: Assume that the data width is represented by Width. (1) If Width<=8, byte enable function is invalid; (2) If Width=9, only 8 bit is valid; (3) If Width>9, both 8 bit and 9 bit are valid.
Resource Usage	**Calculate** - Calculate the resource usage in the design and display results below.
	rSDP Usage - Display the number of rSDP used.
	DFF Usage - Display the number of DFF used.
	LUT Usage - Display the number of LUT used.
	MUX Usage - Display the number of MUX used.
Reset Mode	**Reset Mode** - Set whether the reset mode is synchronous mode or asynchronous mode.
Initialization	**Memory Initialization File** - Set the memory initialization file (.mi) path.
	Dimension Match - Set which port's dimensions the memory initialization file should conform to.

The "Help" page contains a general description of the IP Core, and a brief introduction to the "Options".

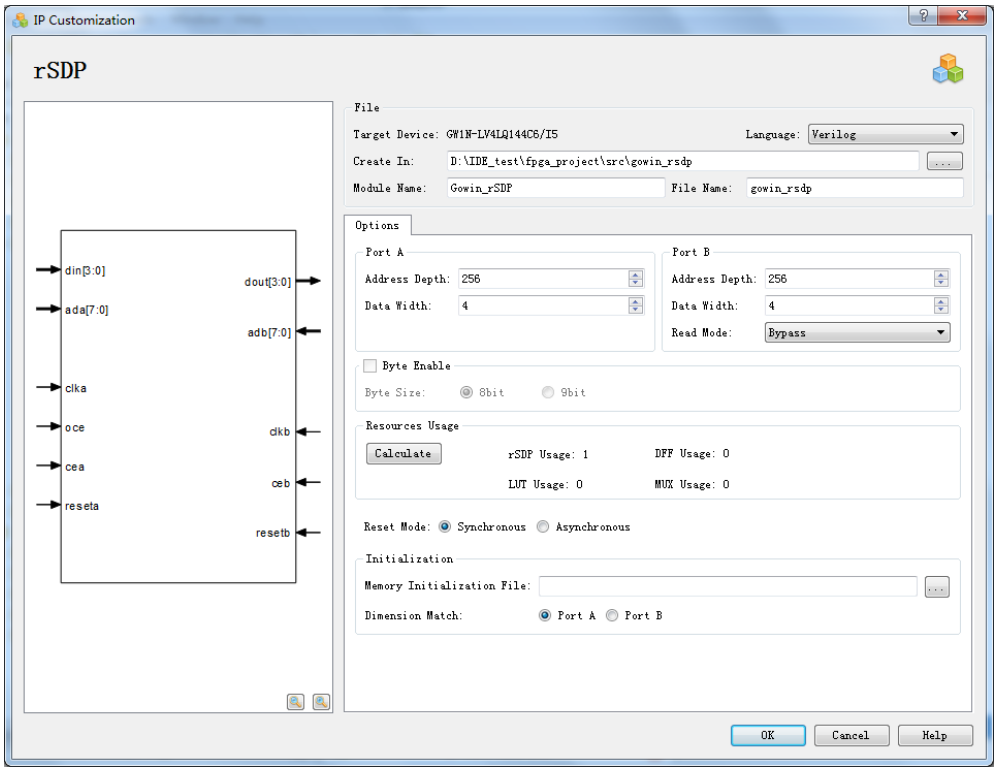
IP Generation Files

As shown in Figure 3-47, after customizing the rSDP IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate Gowin Primitive rSDP design file "gowin_rsdp.v";
- The instantiation template file for the IP design file "gowin_rsdp_tmp.v";
- The configuration files for the Primitive SPB instantiation "gowin_rsdp.ipc".

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.

Figure 3-47 Configured IP Customization



rSDP Design File Instantiation

rSDP design file instantiation is a complete Verilog module. rSDP instantiation is generated according to the SDP configuration that is displayed in the "IP Customization" window, as shown in Figure 3-48.

Note!

Din/Dout data width of the generated rSDP instantiation is consistent with that of the rSDP configured in the "IP Customization" window.

Figure 3-48 rSDP Design File Instantiation

```
module Gowin_rSDP (dout, clka, cea, reseta, clkb, ceb, resetb, oce, ada, din, adb);

output [3:0] dout;
input clka;
input cea;
input reseta;
input clkb;
input ceb;
input resetb;
input oce;
input [7:0] ada;
input [3:0] din;
input [7:0] adb;

wire gw_gnd;

assign gw_gnd = 1'b0;

rSDP rsdp_inst_0 (
    .DO(dout[3:0]),
    .CLKA(clka),
    .CEA(cea),
    .RESETA(reseta),
    .CLKB(clkb),
    .CEB(ceb),
    .RESETB(resetb),
    .OCE(oce),
    .BLKSEL({gw_gnd,gw_gnd,gw_gnd}),
    .ADA({gw_gnd,gw_gnd,gw_gnd,gw_gnd,ada[7:0],gw_gnd,gw_gnd}),
    .DI(din[3:0]),
    .ADB({gw_gnd,gw_gnd,gw_gnd,gw_gnd,adb[7:0],gw_gnd,gw_gnd})
);

defparam rsdp_inst_0.READ_MODE = 1'b0;
defparam rsdp_inst_0.BIT_WIDTH_0 = 4;
defparam rsdp_inst_0.BIT_WIDTH_1 = 4;
defparam rsdp_inst_0.BLK_SEL = 3'b000;
defparam rsdp_inst_0.RESET_MODE = "SYNC";

endmodule //Gowin_rSDP
```

Instantiation Template File for the IP Design File

For the practical use, the IP Core Generator generates the template file while generating the rSDP design file instantiation, as shown in Figure 3-49.

Figure 3-49 Instantiation Template File for the IP Design File

```
Gowin_rSDP your_instance_name(
    .dout(dout_o), //output [3:0] dout
    .clka(clka_i), //input clka
    .cea(cea_i), //input cea
    .reseta(reseta_i), //input reseta
    .clkb(clkb_i), //input clkb
    .ceb(ceb_i), //input ceb
    .resetb(resetb_i), //input resetb
    .oce(oce_i), //input oce
    .ada(ada_i), //input [7:0] ada
    .din(din_i), //input [3:0] din
    .adb(adb_i) //input [7:0] adb
);
```

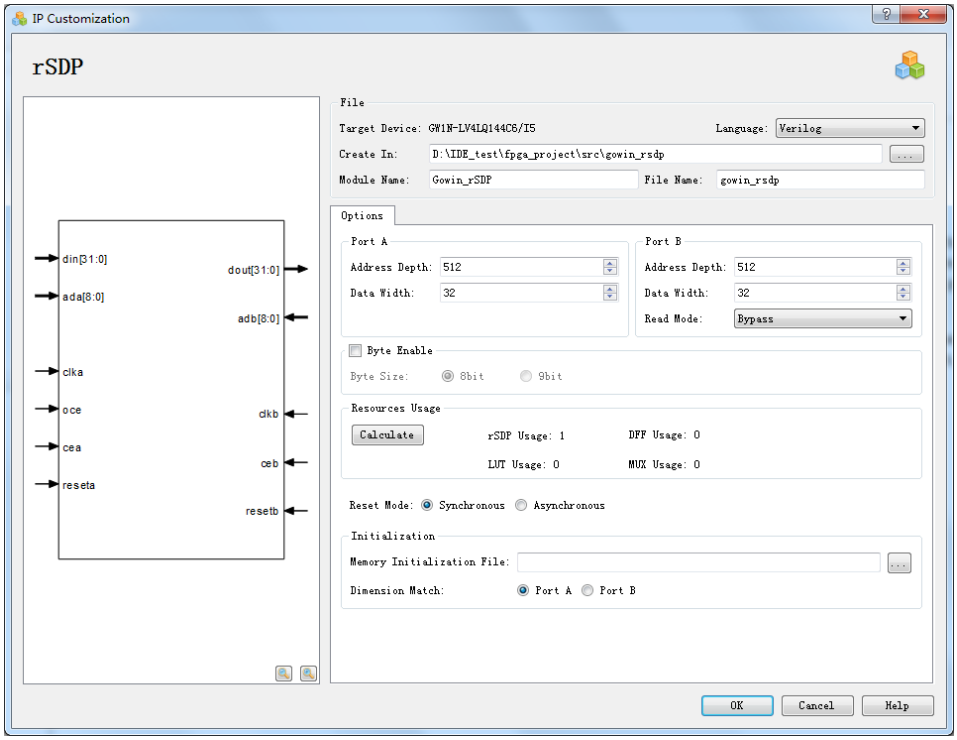
rSDP Generation Example

If users need to generate a specific rSDP IP as follows: 512 x 32,

Bypass read mode and synchronous reset.Take the GW1N-LV4LQ144C6/I5 device for instance, the configuration page is as shown in Figure 3-50. Select a memory initialization file for the module as required, and then click "OK" to generate the customized rSDP IP design files.

The directory where the rSDP IP design file is generated is the path set in "Create In".

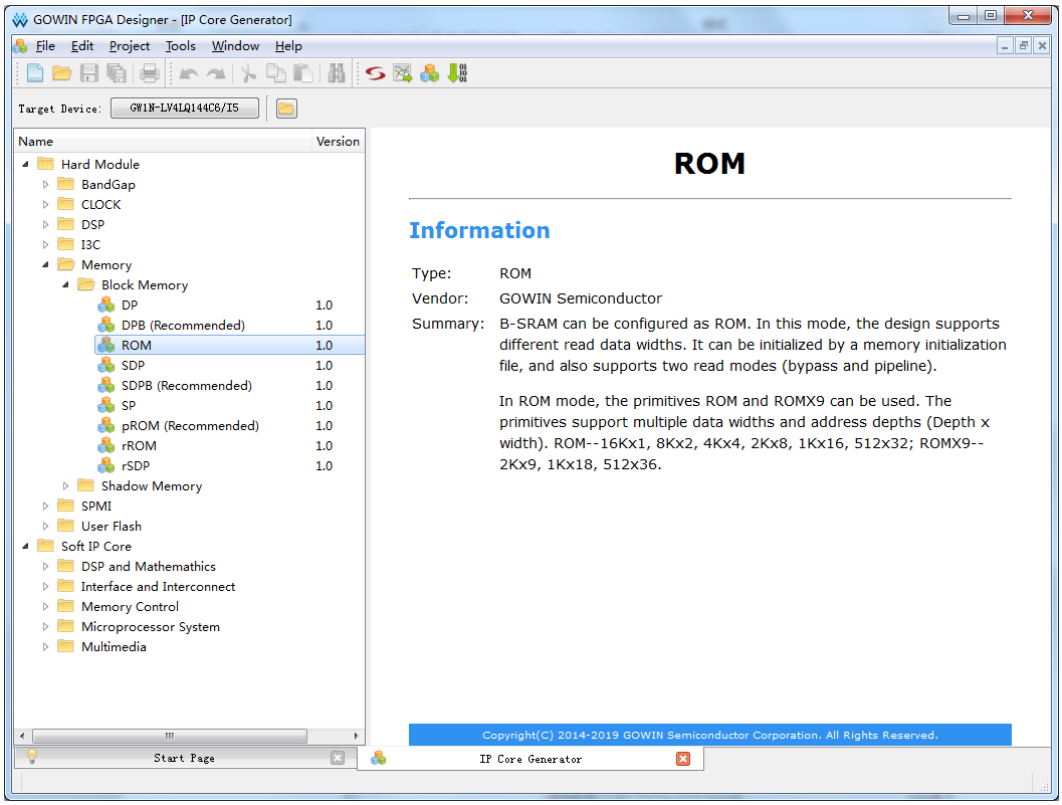
Figure 3-50 IP Customization of rSDP Configuration



3.1.7 ROM

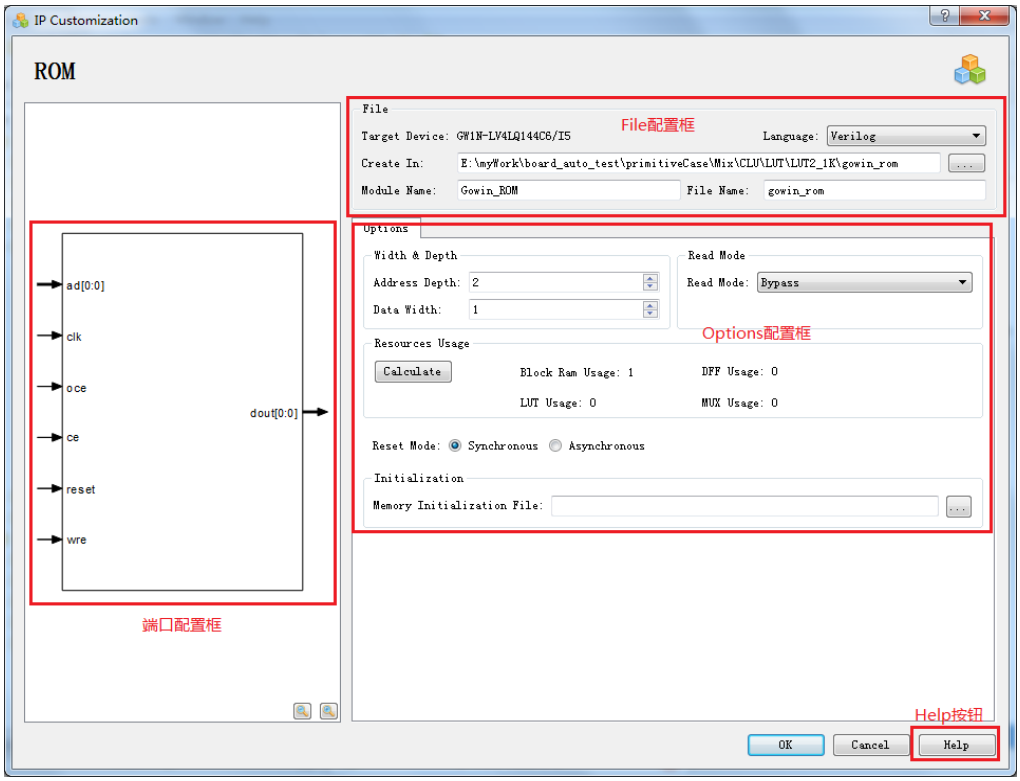
ROM is the read only block memory, which can be implemented by ROM and ROMX9. The maximum capacity of BSRAM varies according to the type of chip Click the "ROM" on the IP Core Generator page. A brief introduction to the ROM will be displayed on the right of the screen, as shown in Figure 3-51.

Figure 3-51 ROM Summary Information



Double click the "ROM" to open the "IP Customization" window. This includes "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button, as shown in Figure 3-52.

Figure 3-52 IP Customization of ROM



1. File Configuration

File configuration includes the basic information related to the ROM instantiation file, as shown in Figure 3-52.

The ROM file configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP> File Configuration.

2. Options Configuration

The Options configuration box is used to configure read only memory by users, as shown in Figure 3-52.

ROM Options configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP> Options Configuration.

Note!

- ROM only supports read operation; Read mode can be Bypass or Pipeline;
- The date width in Memory initialization File should be consistent with the data width configured. If not, the Init value of generated ROM instantiation is 0 by default, and Error message as follows will pop up:
  Error (MG2105): Initial values' width is unequal to user's width option.

3. Ports Configuration Diagram

The ports configuration diagram displays the current IP Core configuration result. Input/Output bit-width updates in real time based on the "Options" configuration, as shown in Figure 3-52.

"Address Depth" affects the bit-width of ad; "Data Width" affects the bit-width of dout.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-53.

Figure 3-53 Help

ROM

Information

Type:	ROM
Vendor:	GOWIN Semiconductor
Summary:	B-SRAM can be configured as ROM. In this mode, the design supports different read data widths. It can be initialized by a memory initialization file, and also supports two read modes (bypass and pipeline). In ROM mode, the primitives ROM and ROMX9 can be used. The primitives support multiple data widths and address depths (Depth x width). ROM--16Kx1, 8Kx2, 4Kx4, 2Kx8, 1Kx16, 512x32; ROMX9--2Kx9, 1Kx18, 512x36.

Options

Option	Description
Width & Depth	**Address Depth** - Set the size of the address depth.
	Data Width - Set the size of the data width.
Read Mode	**Read Mode** - Set whether the read mode is bypass mode or pipeline mode.
Resources Usage	**Calculate** - Calculate the resource usage in the design and display results below.
	Block Ram Usage - Display the number of Block Ram used.
	DFF Usage - Display the number of DFF used.
	LUT Usage - Display the number of LUT used.
	MUX Usage - Display the number of MUX used.
Reset Mode	**Reset Mode** - Set whether the reset mode is synchronous or asynchronous.
Initialization	**Memory Initialization File** - Set the memory initialization file (.mi) path.

The "Help" page contains a general description of the IP Core, and a brief introduction to the "Options".

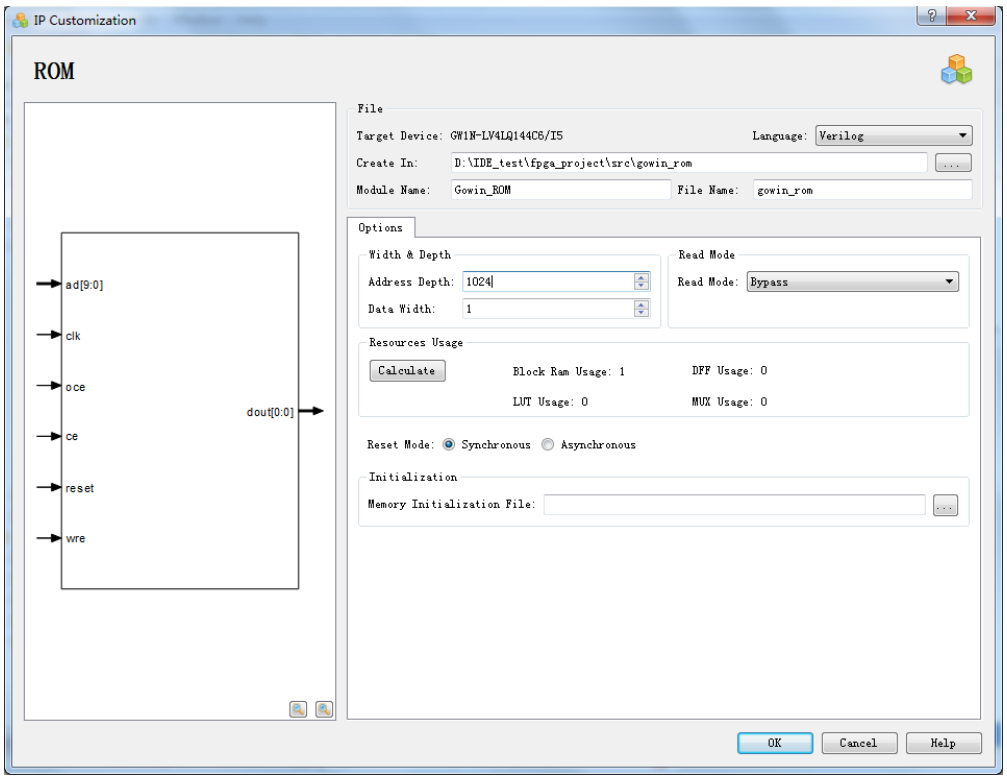
IP Generation Files

As shown in Figure 3-54, after customizing the ROM IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate Gowin Primitive ROM design file "gowin_rom.v";
- The instantiation template file for the IP design file "gowin_rom_tmp.v";
- The configuration files for the Primitive ROM instantiation "gowin_rom.ipc".

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.

Figure 3-54 Configured IP Customization



ROM Design File Instantiation

ROM design file instantiation is a complete Verilog module. ROM instantiation is generated according to the ROM configuration that is displayed in the "IP Customization" window, as shown in Figure 3-55.

Note!

Din/Dout data width of the generated ROM instantiation is consistent with that of the ROM configured in the "IP Customization" window.

Figure 3-55 ROM Design File Instantiation

```
module Gowin_ROM (dout, clk, oce, ce, reset, wre, ad);

output [0:0] dout;
input clk;
input oce;
input ce;
input reset;
input wre;
input [9:0] ad;

wire gw_gnd;

assign gw_gnd = 1'b0;

ROM bram_rom_0 (
    .DO(dout[0]),
    .CLK(clk),
    .OCE(oce),
    .CE(ce),
    .RESET(reset),
    .WRE(wre),
    .BLKSEL({gw_gnd,gw_gnd,gw_gnd}),
    .AD({gw_gnd,gw_gnd,gw_gnd,gw_gnd,ad[9:0]})
);

defparam bram_rom_0.READ_MODE = 1'b0;
defparam bram_rom_0.BIT_WIDTH = 1;
defparam bram_rom_0.BLK_SEL = 3'b000;
defparam bram_rom_0.RESET_MODE = "SYNC";

endmodule //Gowin_ROM
```

Instantiation template file for the IP design file

For practical use, the IP Core Generator generates the template file while generating ROM design file instantiation, as shown in Figure 3-56.

Figure 3-56 Instantiation Template File for the IP Design File

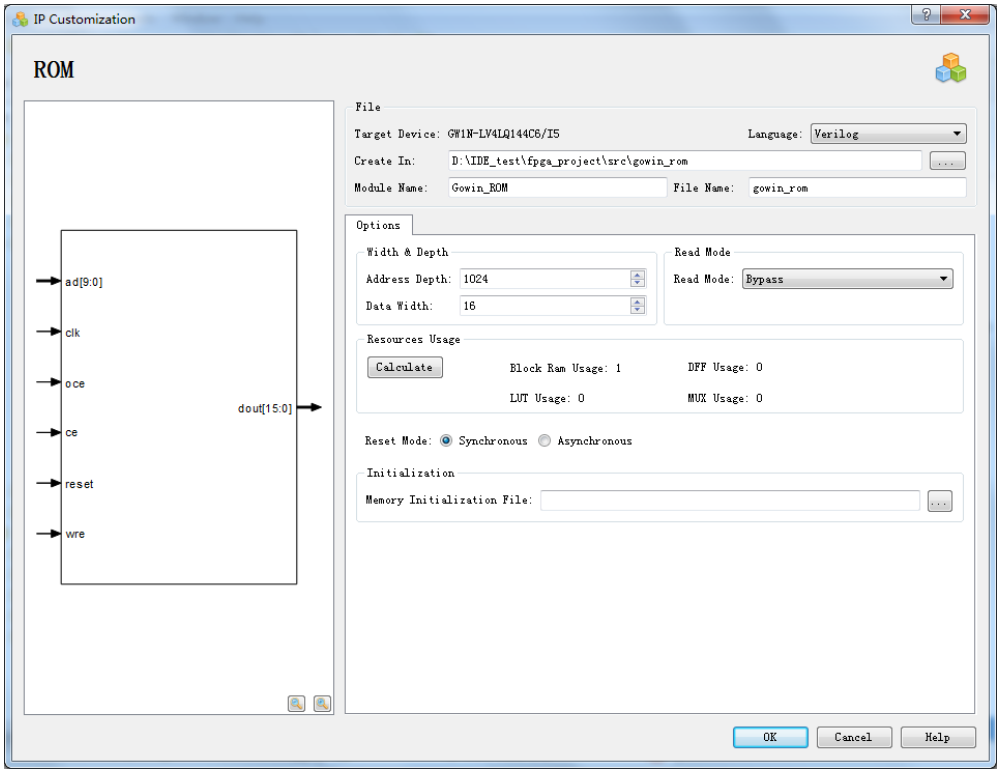
```
Gowin_ROM your_instance_name(
    .dout(dout_o), //output [0:0] dout
    .clk(clk_i), //input clk
    .oce(oce_i), //input oce
    .ce(ce_i), //input ce
    .reset(reset_i), //input reset
    .wre(wre_i), //input wre
    .ad(ad_i) //input [9:0] ad
);
```

ROM Generation Example

If users need to generate a specific ROM IP as follows: 1024 x 16, Bypass read mode and synchronous reset. Take the GW1N-LV4LQFP144C6/I5 device for instance, the configuration page is as shown in Figure 3-57. Select a memory initialization file for the module as required, and then click "OK" to generate the customized ROM IP design files.

The directory where the ROM IP design file is generated is the path set in "Create In".

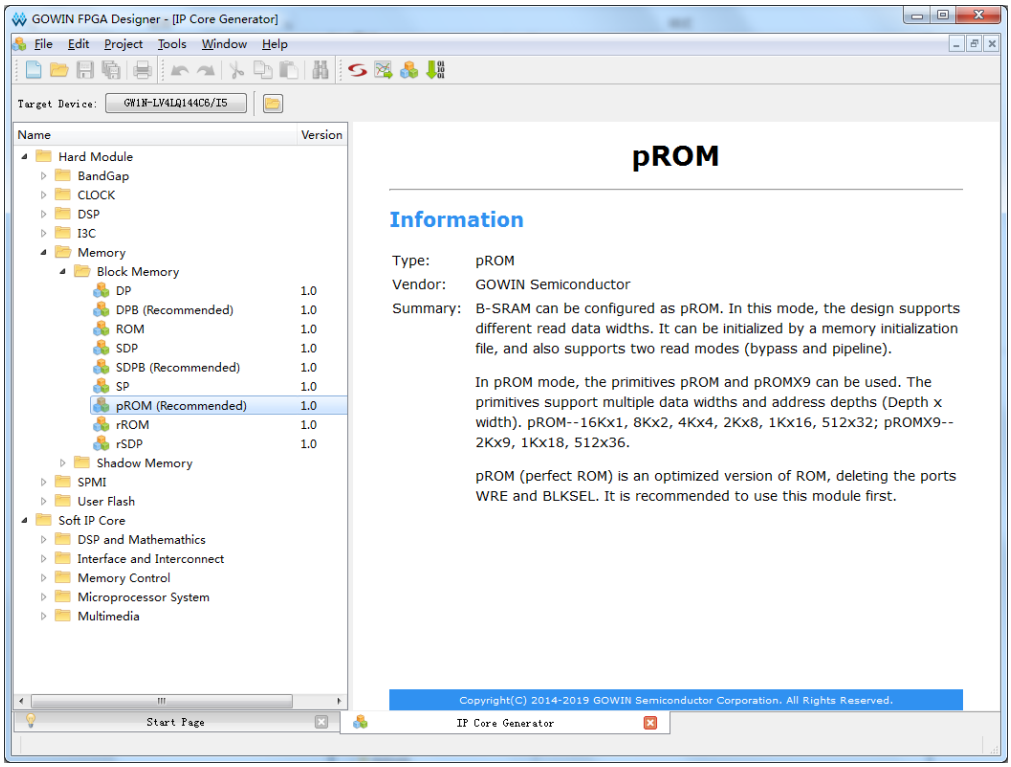
Figure 3-57 IP Customization of ROM Configuration



3.1.8 pROM

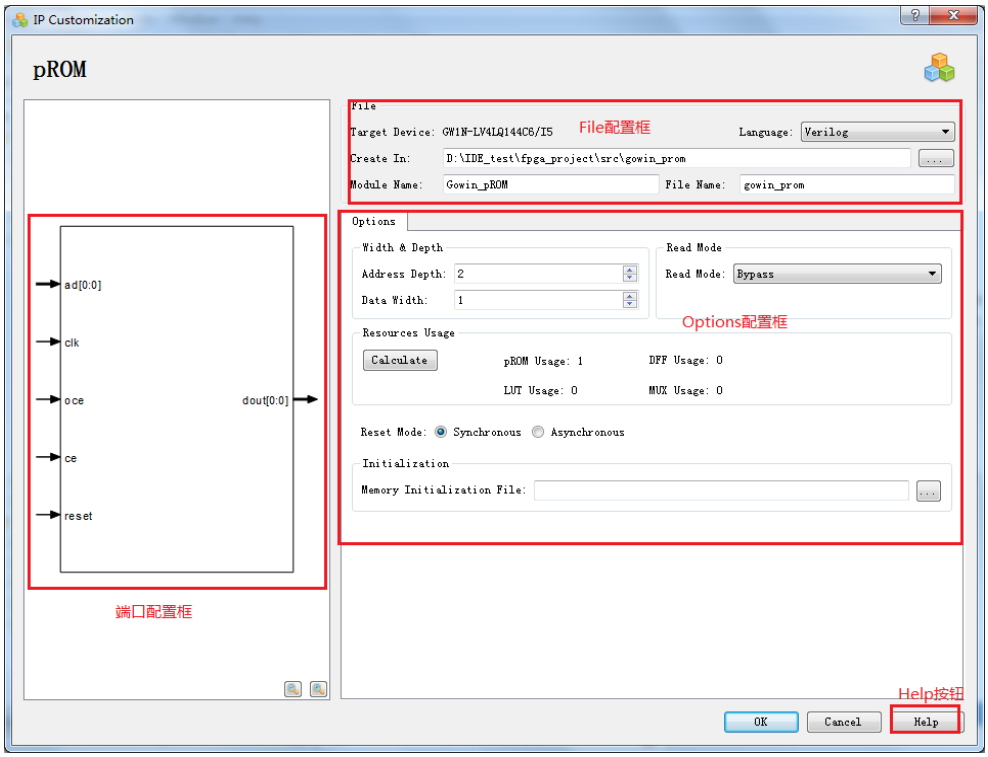
PROM is the read only memory, which can be implemented by pROM and pROMX9. PROM is the optimized version of ROM, and the ports of WRE and BLKSEL are removed. It is recommended that the user first use pROM. Click "pROM" on the IP Core Generator interface. A brief introduction to the pROM will be displayed on the right of the screen, as shown in Figure 3-58.

Figure 3-58 pROM Summary Information



Double click the "pROM" to open the "IP Customization" window This includes "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button, as shown in Figure 3-59.

Figure 3-59 IP Customization of pROM



1. File Configuration

File configuration includes the basic information related to the pROM

instantiation file, as shown in Figure 3-59.

The pROM file configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP> File Configuration.

2. Options Configuration

The Options configuration box is used to configure read only memory by users, as shown in Figure 3-59.

pROM Options configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP > Options Configuration.

Note!

- pROM only supports read operation; Read mode can be Bypass or Pipeline;
- The date width in Memory initialization File should be consistent with the data width configured. If not, the Init value of generated pROM instantiation is 0 by default, and Error message as follows will pop up:
  Error (MG2105): Initial values' width is unequal to user's width option.

3. Ports Configuration Diagram

The ports configuration diagram displays the current IP Core configuration result. Input/Output bit-width updates in real time based on the "Options" configuration, as shown in Figure 3-59.

"Address Depth" affects the bit-width of ad; "Data Width" affects the bit-width of dout.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-60.

Figure 3-60 Help

pROM

Information

Type:	pROM
Vendor:	GOWIN Semiconductor
Summary:	B-SRAM can be configured as pROM. In this mode, the design supports different read data widths. It can be initialized by a memory initialization file, and also supports two read modes (bypass and pipeline). In pROM mode, the primitives pROM and pROMX9 can be used. The primitives support multiple data widths and address depths (Depth x width). pROM--16Kx1, 8Kx2, 4Kx4, 2Kx8, 1Kx16, 512x32; pROMX9--2Kx9, 1Kx18, 512x36. pROM (perfect ROM) is an optimized version of ROM, deleting the ports WRE and BLKSEL. It is recommended to use this module first.

Options

Option	Description
Width & Depth	**Address Depth** - Set the size of the address depth.
	Data Width - Set the size of the data width.
Read Mode	**Read Mode** - Set whether the read mode is bypass mode or pipeline mode.
Resources Usage	**Calculate** - Calculate the resource usage in the design and display results below.
	pROM Usage - Display the number of Block Ram used.
	DFF Usage - Display the number of DFF used.
	LUT Usage - Display the number of LUT used.
	MUX Usage - Display the number of MUX used.
Reset Mode	**Reset Mode** - Set whether the reset mode is synchronous or asynchronous.
Initialization	**Memory Initialization File** - Set the memory initialization file (.mi) path.

The "Help" page contains a general description of the IP Core, and a brief introduction to the "Options".

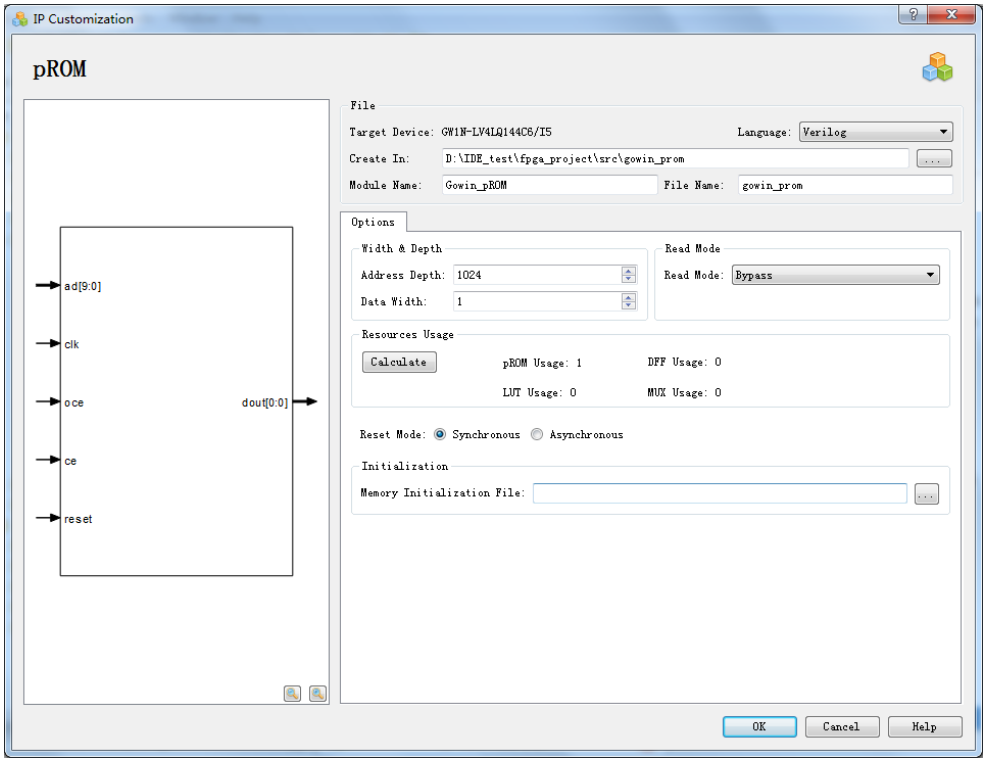
IP Generation Files

As shown in Figure 3-61, after customizing the pROM IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate Gowin Primitive pROM design file "gowin_prom.v";
- The instantiation template file for the IP design file "gowin_prom_tmp.v";
- The configuration files for the Primitive pROM instantiation "gowin_prom.ipc".

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.

Figure 3-61 Configured IP Customization



pROM Design File Instantiation

pROM design file instantiation is a complete Verilog module. ROM instantiation is generated according to the pROM configuration that is displayed in the "IP Customization" window, as shown in Figure 3-62.

Note!

Din/Dout data width of the generated pROM instantiation is consistent with that of the pROM configured in the "IP Customization" window.

Figure 3-62 pROM Design File Instantiation

```
module Gowin_pROM (dout, clk, oce, ce, reset, ad);

output [0:0] dout;
input clk;
input oce;
input ce;
input reset;
input [9:0] ad;

wire gw_gnd;

assign gw_gnd = 1'b0;

pROM prom_inst_0 (
    .DO(dout[0]),
    .CLK(clk),
    .OCE(oce),
    .CE(ce),
    .RESET(reset),
    .AD({gw_gnd,gw_gnd,gw_gnd,gw_gnd,ad[9:0]})
);

defparam prom_inst_0.READ_MODE = 1'b0;
defparam prom_inst_0.BIT_WIDTH = 1;
defparam prom_inst_0.RESET_MODE = "SYNC";

endmodule //Gowin_pROM
```

Instantiation template file for the IP design file

For practical use, the IP Core Generator generates the template file while generating pROM design file instantiation, as shown in Figure 3-63.

Figure 3-63 Instantiation Template File for the IP Design File

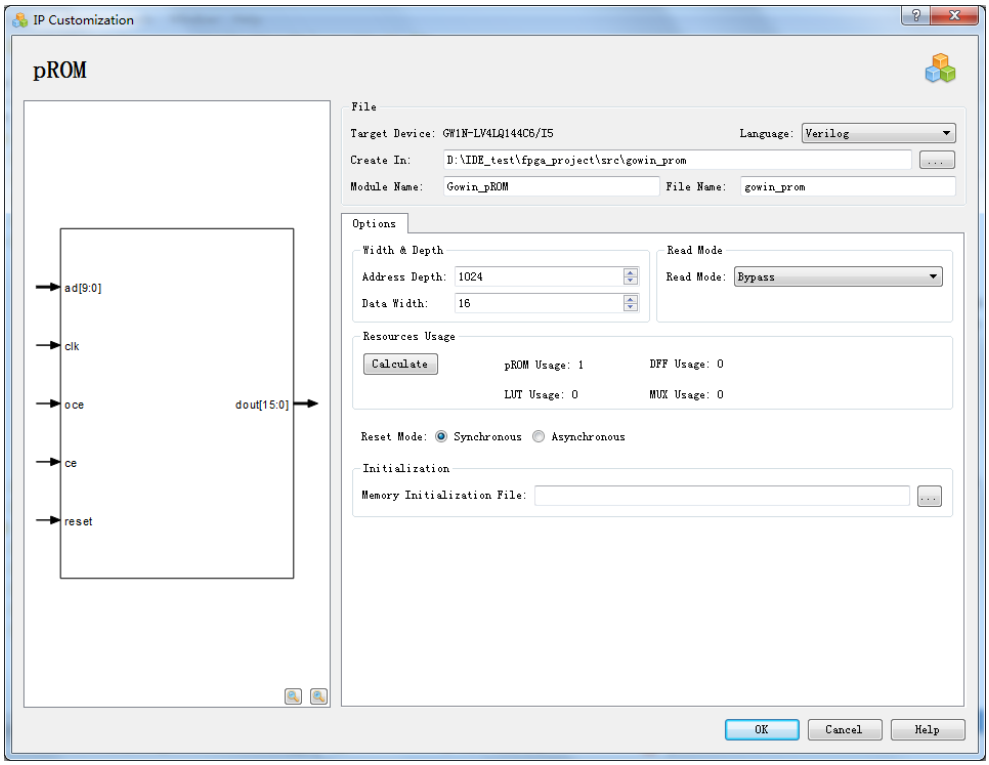
```
Gowin_pROM your_instance_name(
    .dout(dout_o), //output [0:0] dout
    .clk(clk_i), //input clk
    .oce(oce_i), //input oce
    .ce(ce_i), //input ce
    .reset(reset_i), //input reset
    .ad(ad_i) //input [9:0] ad
);
```

pROM Generation Example

If users need to generate a specific pROM IP as follows: 1024 x 16, Bypass read mode and synchronous reset. Take the GW1N-LV4LQFP144C6/I5 device for instance, the configuration page is as shown in Figure 3-64. Select a memory initialization file for the module as required, and then click "OK" to generate the customized ROM IP design files.

The directory where the pROM IP design file is generated is the path set in "Create In".

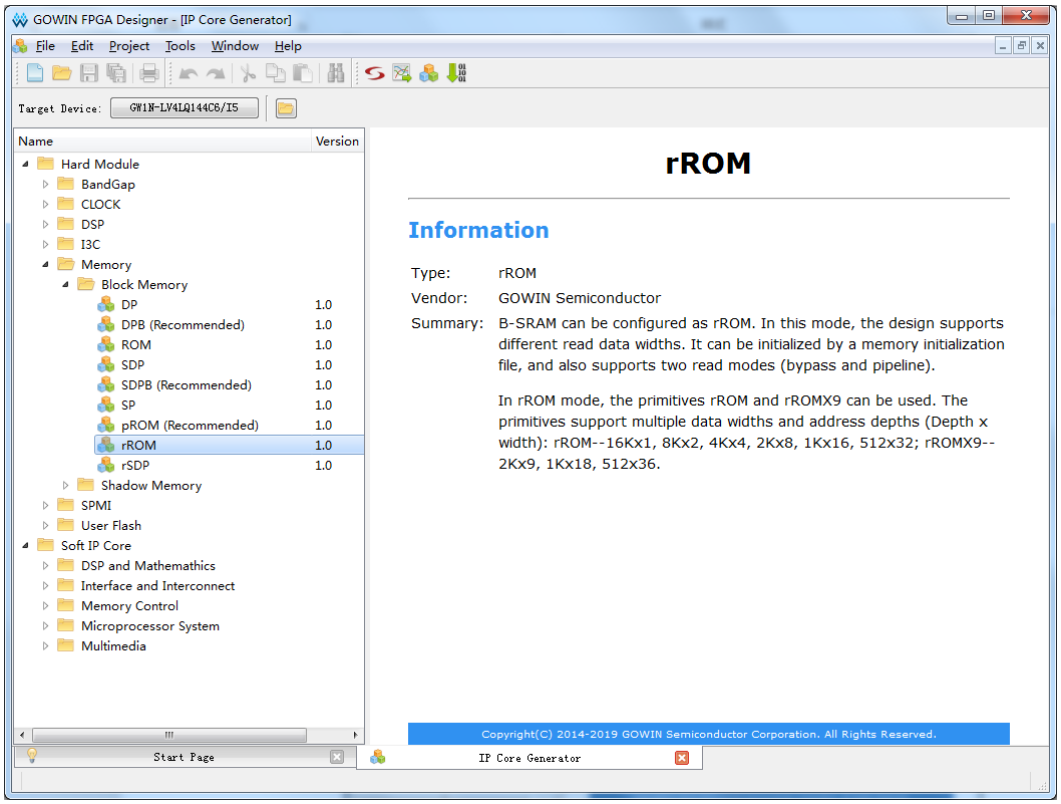
Figure 3-64 IP Customization of pROM Configuration



3.1.9 rROM

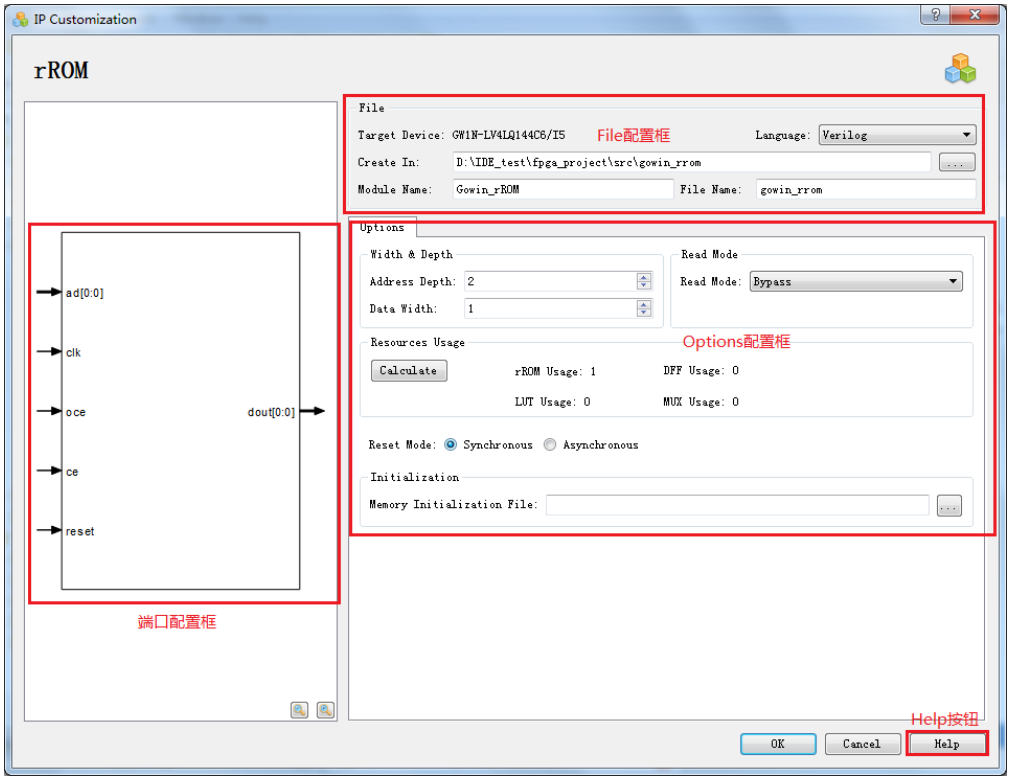
rROM is the read only memory, which can be implemented by rROM and rROMX9. Compared with ROM, rROM removes WRE port. Click "rROM" on the IP Core Generator interface. A brief introduction to the rROM will be displayed on the right of the screen, as shown in Figure 3-65.

Figure 3-65 rROM Summary Information



Double click the "rROM" to open the "IP Customization" window This includes "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button, as shown in Figure 3-66.

Figure 3-66 IP Customization of rROM



1. File Configuration

File configuration includes the basic information related to the rROM instantiation file, as shown in Figure 3-66.

The rROM file configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP> File Configuration.

2. Options Configuration

The Options configuration box is used to configure read only memory by users, as shown in Figure 3-66.

rROM Options configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP > Options Configuration.

Note!

- rROM only supports read operation; Read mode can be Bypass or Pipeline;
- The date width in Memory initialization File should be consistent with the data width configured. If not, the Init value of generated rROM instantiation is 0 by default, and Error message as follows will pop up:
  Error (MG2105): Initial values' width is unequal to user's width option.

3. Ports Configuration

The ports configuration diagram displays the current IP Core configuration result. Input/Output bit-width updates in real time based on the "Options" configuration, as shown in Figure 3-66.

"Address Depth" affects the bit-width of ad; "Data Width" affects the bit-width of dout.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-67.

Figure 3-67 Help

rROM

Information

Type:	rROM
Vendor:	GOWIN Semiconductor
Summary:	B-SRAM can be configured as rROM. In this mode, the design supports different read data widths. It can be initialized by a memory initialization file, and also supports two read modes (bypass and pipeline). In rROM mode, the primitives rROM and rROMX9 can be used. The primitives support multiple data widths and address depths (Depth x width): rROM--16Kx1, 8Kx2, 4Kx4, 2Kx8, 1Kx16, 512x32; rROMX9--2Kx9, 1Kx18, 512x36.

Options

Option	Description
Width & Depth	**Address Depth** - Set the size of the address depth.
	Data Width - Set the size of the data width.
Read Mode	**Read Mode** - Set whether the read mode is bypass mode or pipeline mode.
Resources Usage	**Calculate** - Calculate the resource usage in the design and display results below.
	rROM Usage - Display the number of rROM used.
	DFF Usage - Display the number of DFF used.
	LUT Usage - Display the number of LUT used.
	MUX Usage - Display the number of MUX used.
Reset Mode	**Reset Mode** - Set whether the reset mode is synchronous or asynchronous.
Initialization	**Memory Initialization File** - Set the memory initialization file (.mi) path.

The "Help" page contains a general description of the IP Core, and a brief introduction to the "Options".

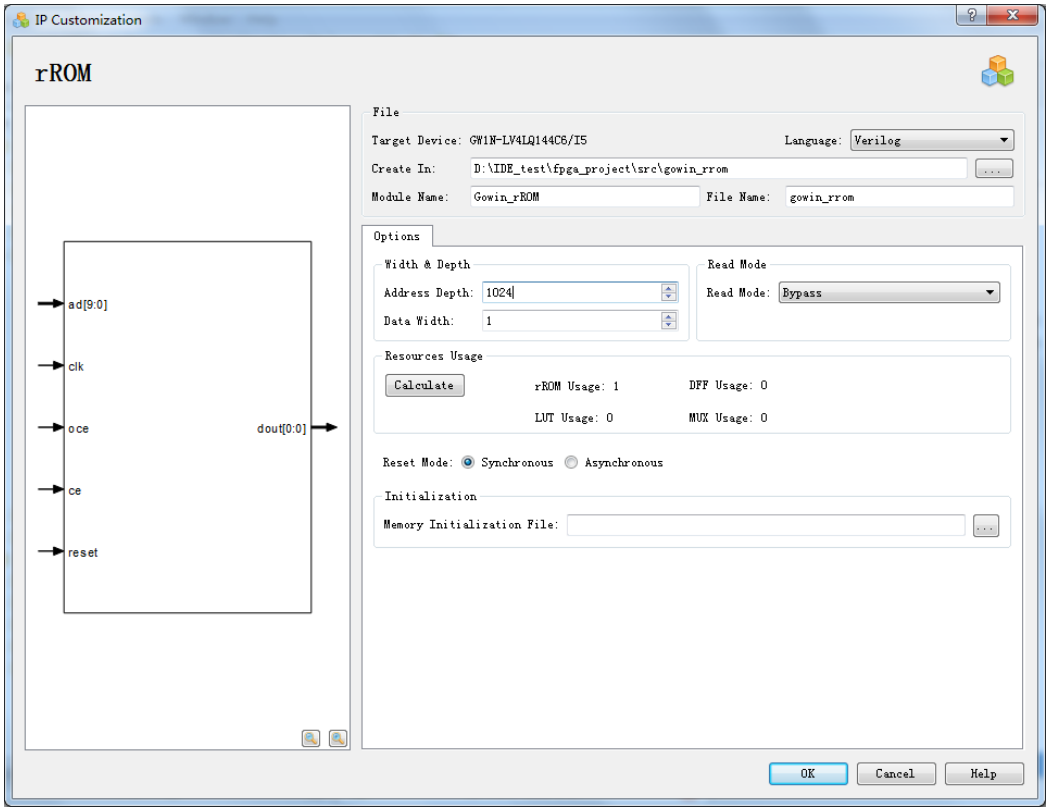
IP Generation Files

As shown in Figure 3-68, after customizing the rROM IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate Gowin Primitive rROM design file "gowin_rrom.v";
- The instantiation template file for the IP design file "gowin_rrom_tmp.v";
- The configuration files for the Primitive rROM instantiation "gowin_rrom.ipc".

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.

Figure 3-68 Configured IP Customization



rROM Design File Instantiation

rROM design file instantiation is a complete Verilog module. rROM instantiation is generated according to the pROM configuration that is displayed in the "IP Customization" window, as shown in Figure 3-69.

Note!

Din/Dout data width of the generated rROM instantiation is consistent with that of the rROM configured in the "IP Customization" window.

Figure 3-69 rROM Design File Instantiation

```
module Gowin_rROM (dout, clk, oce, ce, reset, ad);

output [0:0] dout;
input clk;
input oce;
input ce;
input reset;
input [9:0] ad;

wire gw_gnd;

assign gw_gnd = 1'b0;

rROM rrom_inst_0 (
    .DO(dout[0]),
    .CLK(clk),
    .OCE(oce),
    .CE(ce),
    .RESET(reset),
    .BLKSEL({gw_gnd,gw_gnd,gw_gnd}),
    .AD({gw_gnd,gw_gnd,gw_gnd,gw_gnd,ad[9:0]})
);

defparam rrom_inst_0.READ_MODE = 1'b0;
defparam rrom_inst_0.BIT_WIDTH = 1;
defparam rrom_inst_0.BLK_SEL = 3'b000;
defparam rrom_inst_0.RESET_MODE = "SYNC";

endmodule //Gowin_rROM
```

Instantiation template file for the IP design file

For practical use, the IP Core Generator generates the template file while generating rROM design file instantiation, as shown in Figure 3-70.

Figure 3-70 Instantiation Template File for the IP Design File

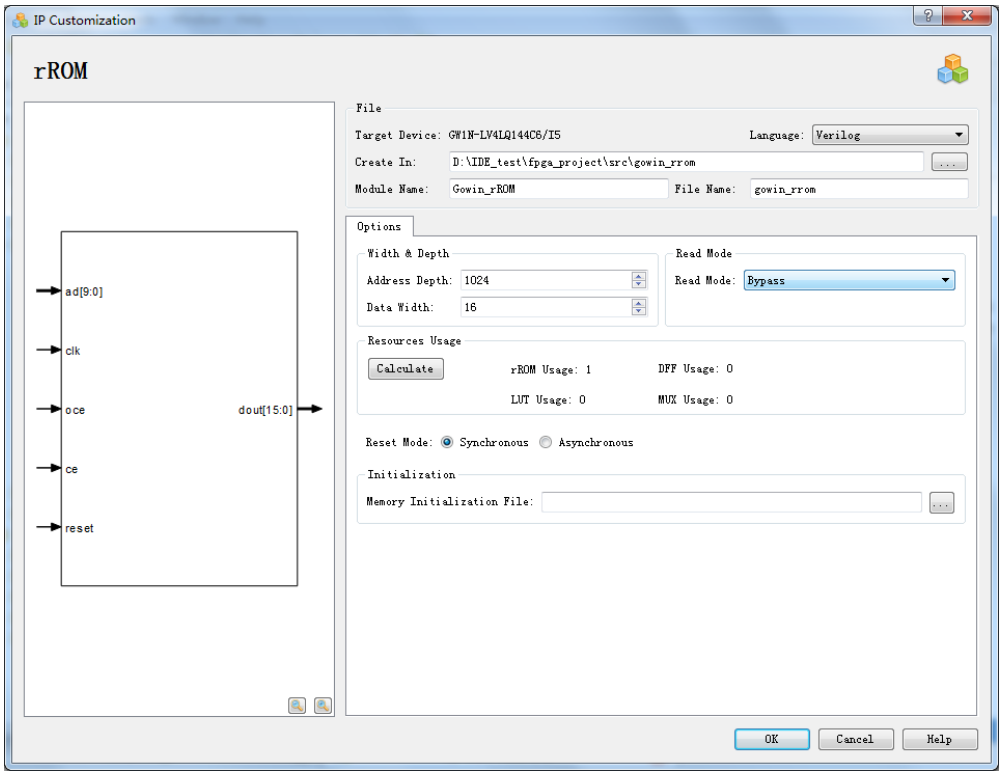
```
Gowin_rROM your_instance_name(
    .dout(dout_o), //output [0:0] dout
    .clk(clk_i), //input clk
    .oce(oce_i), //input oce
    .ce(ce_i), //input ce
    .reset(reset_i), //input reset
    .ad(ad_i) //input [9:0] ad
);
```

rROM Generation Example

If users need to generate a specific rROM IP as follows: 1024 x 16, Bypass read mode and synchronous reset. Take the GW1N-LV4LQFP144C6/I5 device for instance, the configuration page is as shown in Figure 3-71. Select a memory initialization file for the module as required, and then click "OK" to generate the customized ROM IP design files.

The directory where the rROM IP design file is generated is the path set in "Create In".

Figure 3-71 IP Customization of rROM Configuration



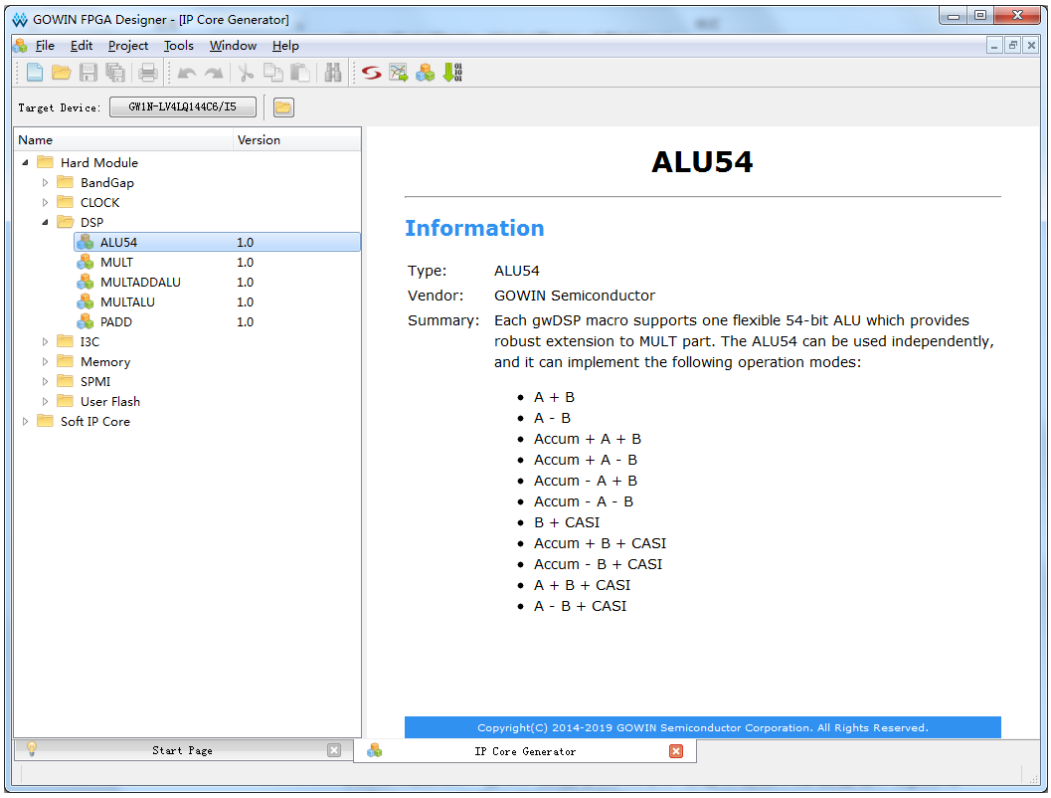
3.2 DSP

Currently, DSP module supports five types of Gowin devices generation: ALU54, MULT, MULTADDALU, MULTALU and PADD.

3.2.1 ALU54

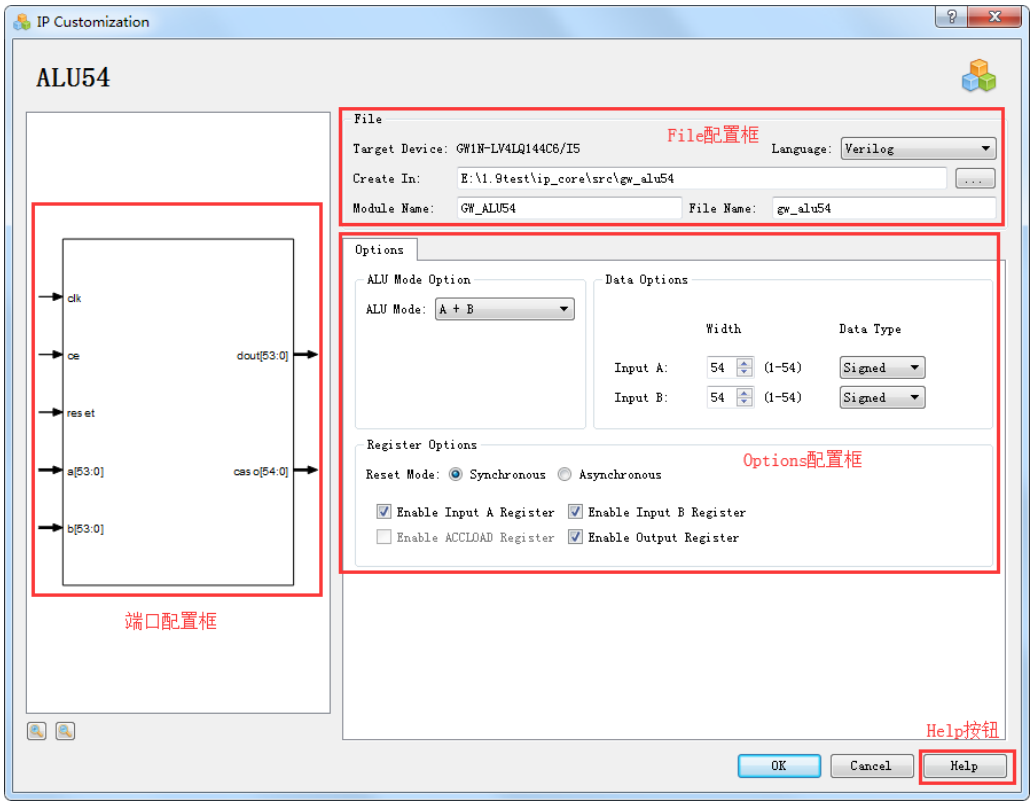
ALU54 can be used to implement 54-bit arithmetic and logical operations. Click "ALU54 " on the IP Core Generator page. A brief introduction to the ALU54 will be displayed on the right of the screen, as shown in Figure 3-72.

Figure 3-72 ALU54 Summary Information



On the IP Core Generator interface, double-clicking on "ALU54 " to open the "IP Customization" window, as shown in Figure 3-73. This includes "File" configuration, "Options" configuration, the port configuration diagram, and the "Help" button.

Figure 3-73 IP Customization of ALU54



1. File Configuration

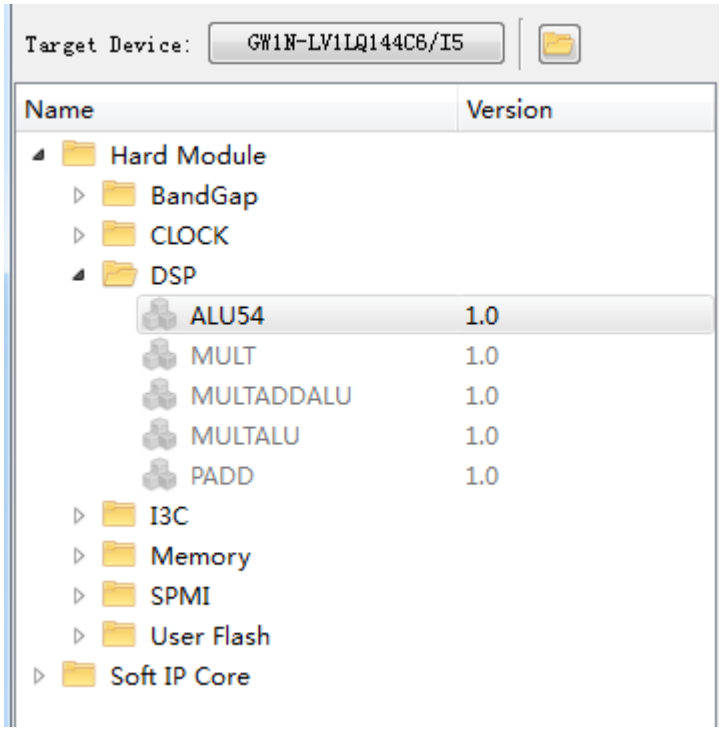
The file configuration includes the basic information related to the ALU54 instantiation file, as shown in Figure 3-73.

The ALU54 file configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP > File Configuration.

Note!

If GW1N-1, which does not support DSP, is selected, "Options" configuration will be grayed-out and unavailable, as shown in Figure 3-74.

Figure 3-74 DSP Displaying in Grey



2. Options Configuration

Options configuration includes configuration information related to the ALU54 instantiation file, as shown in Figure 3-73.

- ALU Mode Option: Allows users to select the operation modes The MULTADDALU can be configured in the following operation modes:
  - A + B;
  - A – B;
  - Accum + A + B;
  - Accum + A – B;
  - Accum - A + B;
  - Accum - A – B;
  - B + CASI;
  - Accum + B + CASI;
  - Accum - B + CASI;
  - A + B + CASI;
  - A - B + CASI;
- Data Options: Allows users to configure data options.

Configure ALU54 input data width. The data width of input port A/B can be configured as 1-54 bit;

Output width adjusts automatically according to the input width;

Data Type: Can be configured as signed or unsigned.

- Register Options: Allows users to configure registers operation mode.
  - Reset Mode: Configure whether the reset mode is synchronous or asynchronous;
  - Enable Input A Register: Allows users to enable or disable Input A register;

- Enable Input B Register: Allows users to enable or disable Input B register;
- Enable ACCLOAD Register: Allows users to enable or disable ACCLOAD register;
- Enable Output Register: allows users to enable or disable Output register.

3. Ports Configuration Diagram

The ports configuration diagram displays the current IP Core configuration result. Input/Output bit-width updates in real time based on the "Options" configuration, as shown in Figure 3-73.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-75.

Figure 3-75 Help

ALU54

Information

Type:	ALU54
Vendor:	GOWIN Semiconductor
Summary:	Each gwDSP macro supports one flexible 54-bit ALU which provides robust extension to MULT part. The ALU54 can be used independently, and it can implement the following operation modes: • A + B • A - B • Accum + A + B • Accum + A - B • Accum - A + B • Accum - A - B • B + CASI • Accum + B + CASI • Accum - B + CASI • A + B + CASI • A - B + CASI

Options

Option	Description
ALU54 Mode Option	**ALU54 Mode** - Set one of the ALU54 operation modes.
Data Options	**Input A Width** - Set the size of the first item in the ALU54.
	Input B Width - Set the size of the second item in the ALU54.
	Data Type - Set the data format of the inputs as signed or unsigned.
Register Options	**Reset Mode** - Set whether the reset mode is synchronous or asynchronous.
	Enable ... Register - Enable or disable registers. For example, if you choose Enable Input A Register, the input data will go through one register.

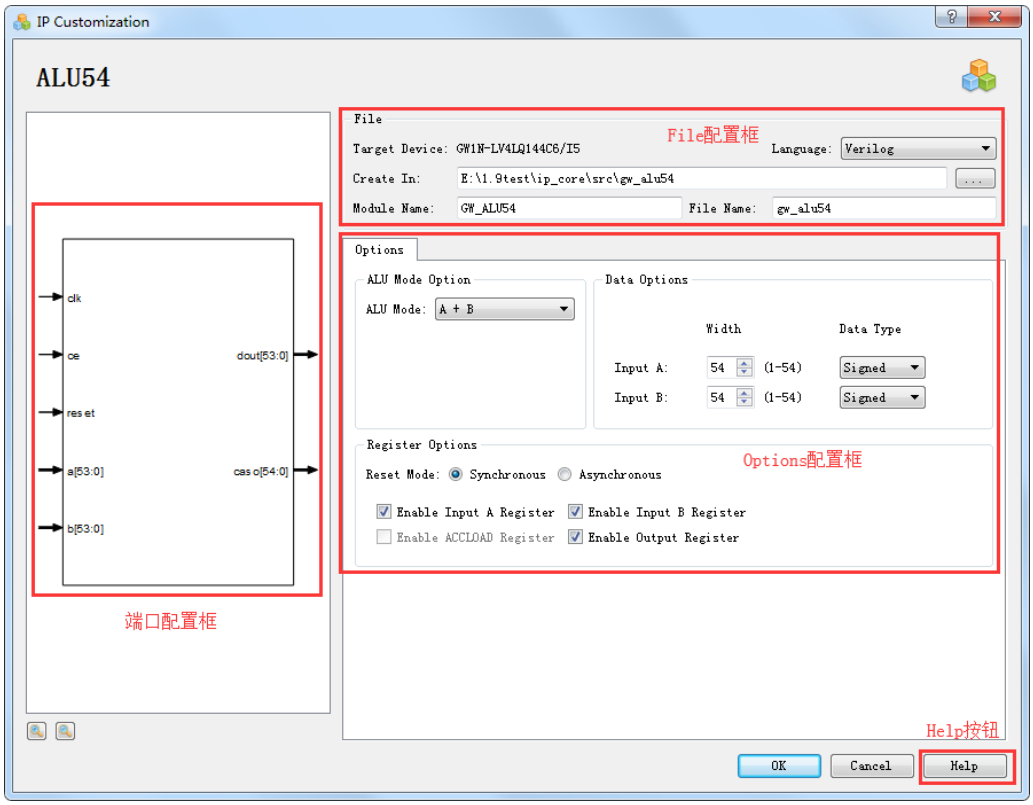
IP Generation Files

As shown in Figure 3-76, after customizing the IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate Gowin Primitive ALU54 design file "gw_alu54.v";
- The instantiation template file for the IP design file "gw_alu54_tmp.v";
- The configuration file for the Gowin Primitive ALU54 instantiation "gw_alu54.ipc".

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.

Figure 3-76 Configured IP Customization



ALU54 Design File Instantiation

ALU54 design file instantiation is a complete Verilog module. ALU54 instantiation is generated according to the ALU54 configuration specified in the "IP Customization" window, as shown in Figure 3-77.

Figure 3-77 ALU54 Design File Instantiation

```
module GW_ALU54 (dout, caso, a, b, ce, clk, reset);

output [53:0] dout;
output [54:0] caso;
input [53:0] a;
input [53:0] b;
input ce;
input clk;
input reset;

wire gw_vcc;
wire gw_gnd;

assign gw_vcc = 1'b1;
assign gw_gnd = 1'b0;

ALU54D alu54d_inst (
    .DOUT(dout),
    .CASO(caso),
    .A(a),
    .B(b),
    .ASIGN(gw_vcc),
    .BSIGN(gw_vcc),
    .CASI({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,
    .ACCLOAD(gw_gnd),
    .CE(ce),
    .CLK(clk),
    .RESET(reset)
);

defparam alu54d_inst.AREG = 1'b1;
defparam alu54d_inst.BREG = 1'b1;
defparam alu54d_inst.ASIGN_REG = 1'b0;
defparam alu54d_inst.BSIGN_REG = 1'b0;
defparam alu54d_inst.ACCLOAD_REG = 1'b0;
defparam alu54d_inst.OUT_REG = 1'b1;
defparam alu54d_inst.B_ADD_SUB = 1'b0;
defparam alu54d_inst.C_ADD_SUB = 1'b0;
defparam alu54d_inst.ALUD_MODE = 0;
defparam alu54d_inst.ALU_RESET_MODE = "SYNC";

endmodule //GW_ALU54
```

Instantiation Template File for the IP Design File

For practical use, the IP Core Generator generates the template file while generating the ALU54 design file instantiation, as shown in Figure 3-78.

Figure 3-78 Instantiation Template File for the IP Design File

```
GW_ALU54 your_instance_name(
    .dout(dout_o), //output [53:0] dout
    .caso(caso_o), //output [54:0] caso
    .a(a_i), //input [53:0] a
    .b(b_i), //input [53:0] b
    .ce(ce_i), //input ce
    .clk(clk_i), //input clk
    .reset(reset_i) //input reset
);
```

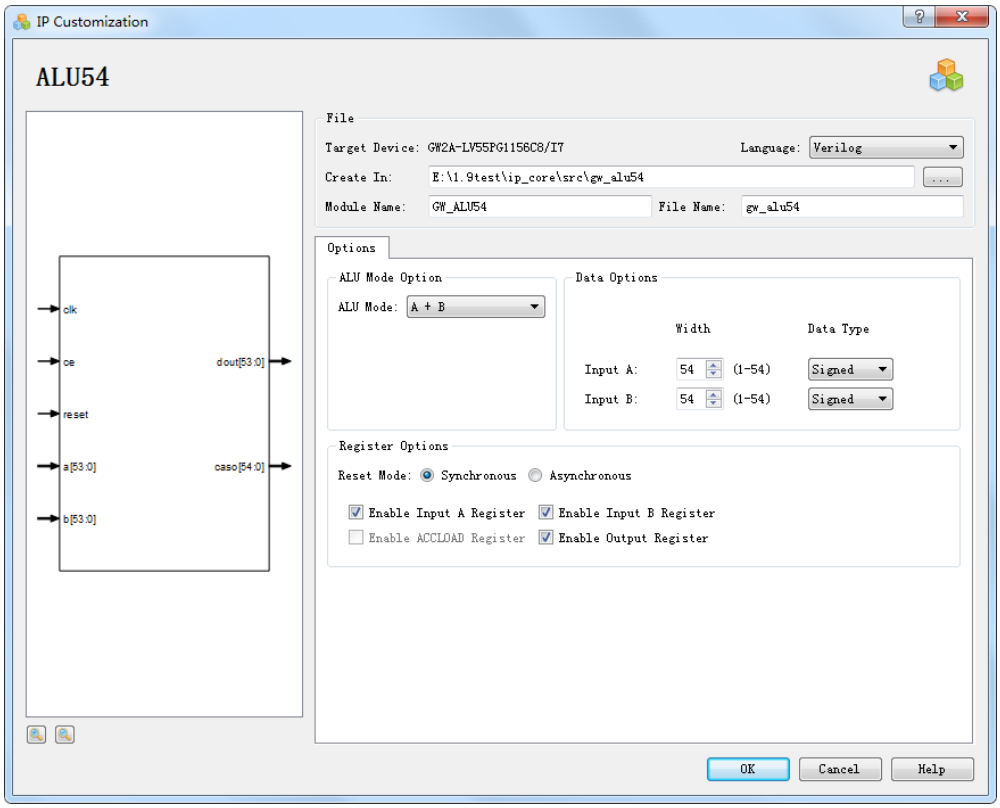
ALU54 Generation Example

If users need to generate a specific ALU54 IP with register as follows: addition of two 54-bit and synchronous. Take the GW2A-LV55PG1156C8/I7 device for instance, the configuration interface is as shown in Figure 3-79.

Click "OK" to generate the customized ALU54 IP design files.

The directory where the ALU54 IP design file is generated is the path set in "Create In".

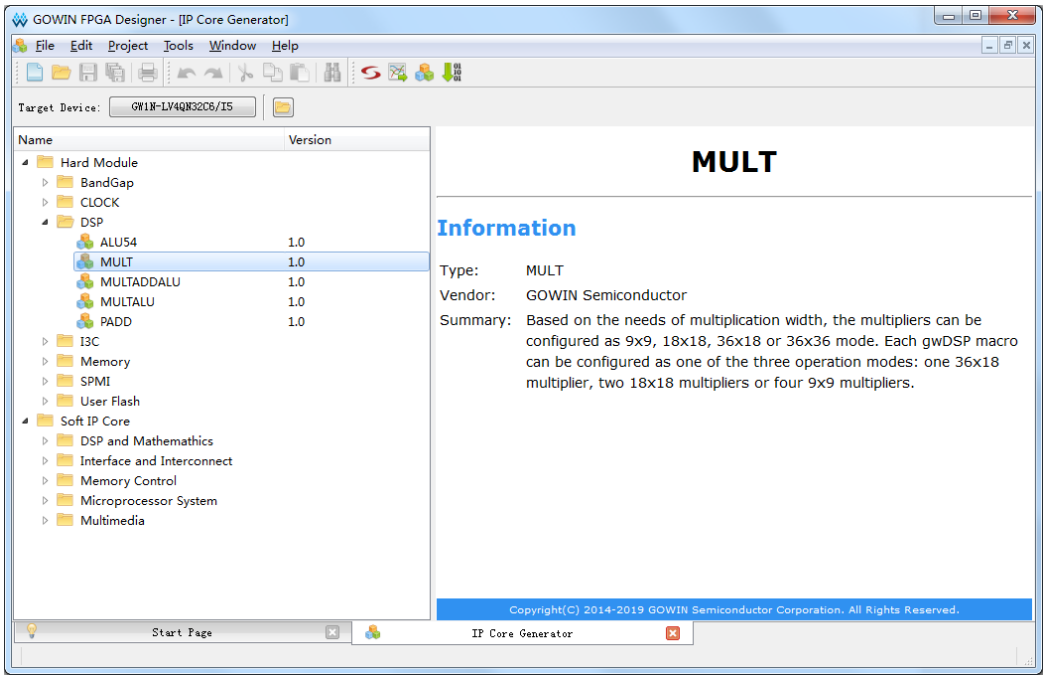
Figure 3-79 ALU54 IP Customization Configuration



3.2.2 MULT

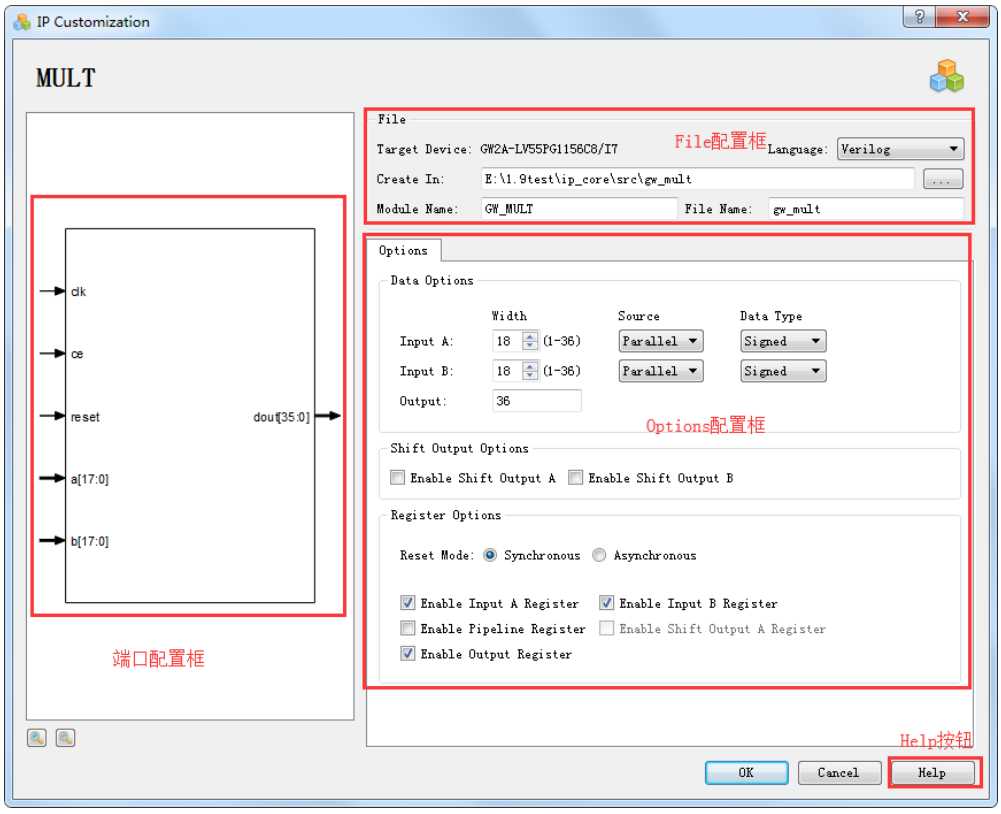
MULT can be configured as a multiplier. Click "MULT" on the IP Core Generator interface. A brief introduction to the MULT will be displayed on the right of the screen, as shown in Figure 3-80.

Figure 3-80 MULT Summary Information



Double click "MULT" to open the "IP Customization" window, as shown in Figure 3-81. This includes "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button.

Figure 3-81 IP Customization of ROM



1. File Configuration

The file configuration includes the basic information related to the

MULT instantiation file, as shown in Figure 3-81.

The MULT file configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP> File Configuration.

2. Options Configuration

Options configuration includes configuration information related to the MULT instantiation file, as shown in Figure 3-81.

- Data Options: Allows users to configure data options.
  - The maximum data width of the input ports (Input A Width/ Input B Width) is 36;
  - Output width adjusts automatically according to input width and generates MULT9X9, MULT18X18, or MULT36X36 according to the width during the instantiation.
  - Input A/B can be set as Parallel, Shift, or Dynamic.
  - The data type can be configured as Unsigned or Signed.
- Shift Output Options: Allows users to select whether to enable shift out. This option can be set when both Input A Width and Input B Width are less than or equal to 18.

Note!

If Either Input A Width or Input B Width is greater than 18, the Shift Output Options will be greyed out and cannot be configured.

- Register Options: The function and operation of the register options are the same as that of ALU54. Please refer to the Options Configuration section in 3.2.1ALU54 for further details.

3. Ports Configuration Diagram

The ports configuration diagram displays the current IP Core configuration result. Input/Output bit-width updates in real time based on the "Options" configuration, as shown in Figure 3-81.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-82. The "Help" page contains a general description of the IP Core, and a brief introduction to the "Options".

Figure 3-82 Help

MULT

Information

Type:	MULT
Vendor:	GOWIN Semiconductor
Summary:	Based on the needs of multiplication width, the multipliers can be configured as 9x9, 18x18, 36x18 or 36x36 mode. Each gwDSP macro can be configured as one of the three operation modes: one 36x18 multiplier, two 18x18 multipliers or four 9x9 multipliers.

Options

Option	Description
Data Options	**Input A Width** - Set the size of the first item in the multiplication.
	Input B Width - Set the size of the second item in the multiplication.
	Output Width - Size of the output. The output size is the sum of the input A and input B bit sizes.
	Source - Set the source of the input A/B as Parallel or Shift.
	Data Type - Set the data format of the inputs as signed or unsigned.
Shift Output Options	**Enable Shift Output A** - Enable or disable the shift out port A of the multiplication.
	Enable Shift Output B – Enable or disable the shift out port B of the multiplication.
	Note: If either of the A and B inputs is greater than 18 bits, the input and output shift options are not available.
Register Options	**Reset Mode** - Set whether the reset mode is synchronous or asynchronous.
	Enable ... Register - Enable or disable registers. For example, if you choose Enable Input A Register, the input data will go through one register.

IP Generation Files

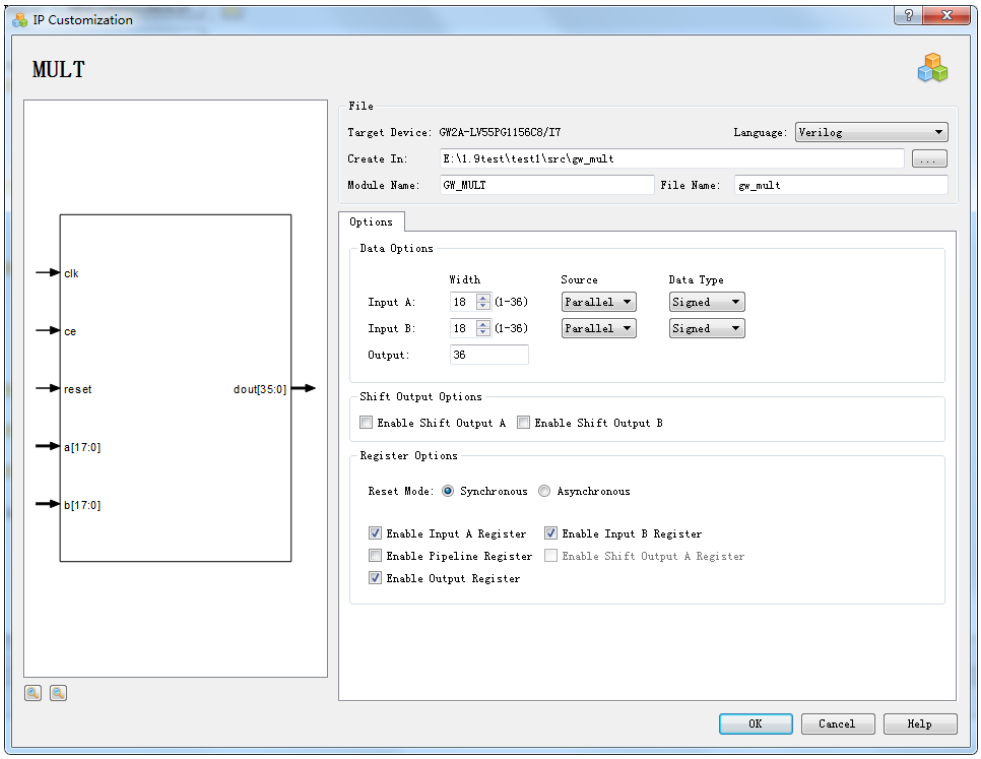
As shown in Figure 3-83, after customizing the IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate Gowin Primitive MULT design file "gw_mult.v";
- The instantiation template file for the IP design file "gw_mult_tmp.v";
- The configuration file for the Gowin Primitive MULT instantiation "gw_mult.ipc".

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

The "IP Customization" window is as shown in Figure 3-83.

Figure 3-83 Configured IP Customization



Design File for the Gowin Primitive MULT Instantiation

MULT design file instantiation is a complete Verilog module. MULT instantiation is generated according to the MULT configuration that is displayed in the "IP Customization" window, as shown in Figure 3-84.

Figure 3-84 MULT Design File Instantiation

```
module GW_MULT (dout, a, b, ce, clk, reset);

output [35:0] dout;
input [17:0] a;
input [17:0] b;
input ce;
input clk;
input reset;

wire [17:0] soa_w;
wire [17:0] sob_w;
wire gw_vcc;
wire gw_gnd;

assign gw_vcc = 1'b1;
assign gw_gnd = 1'b0;

MULT18X18 mult18x18_inst (
    .DOUT(dout),
    .SOA(soa_w),
    .SOB(sob_w),
    .A(a),
    .B(b),
    .ASIGN(gw_vcc),
    .BSIGN(gw_vcc),
    .SIA({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd}),
    .SIB({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd}),
    .CE(ce),
    .CLK(clk),
    .RESET(reset),
    .ASEL(gw_gnd),
    .BSEL(gw_gnd)
);

defparam mult18x18_inst.AREG = 1'b1;
defparam mult18x18_inst.BREG = 1'b1;
defparam mult18x18_inst.OUT_REG = 1'b1;
defparam mult18x18_inst.PIPE_REG = 1'b0;
defparam mult18x18_inst.ASIGN_REG = 1'b0;
defparam mult18x18_inst.BSIGN_REG = 1'b0;
defparam mult18x18_inst.SOA_REG = 1'b0;
defparam mult18x18_inst.MULT_RESET_MODE = "SYNC";

endmodule //GW_MULT
```

Instantiation Template File for the IP Design File

For practical use, the IP Core Generator generates the template file while generating the MULT design file instantiation, as shown in Figure 3-85.

Figure 3-85 Instantiation Template File for the IP Design File

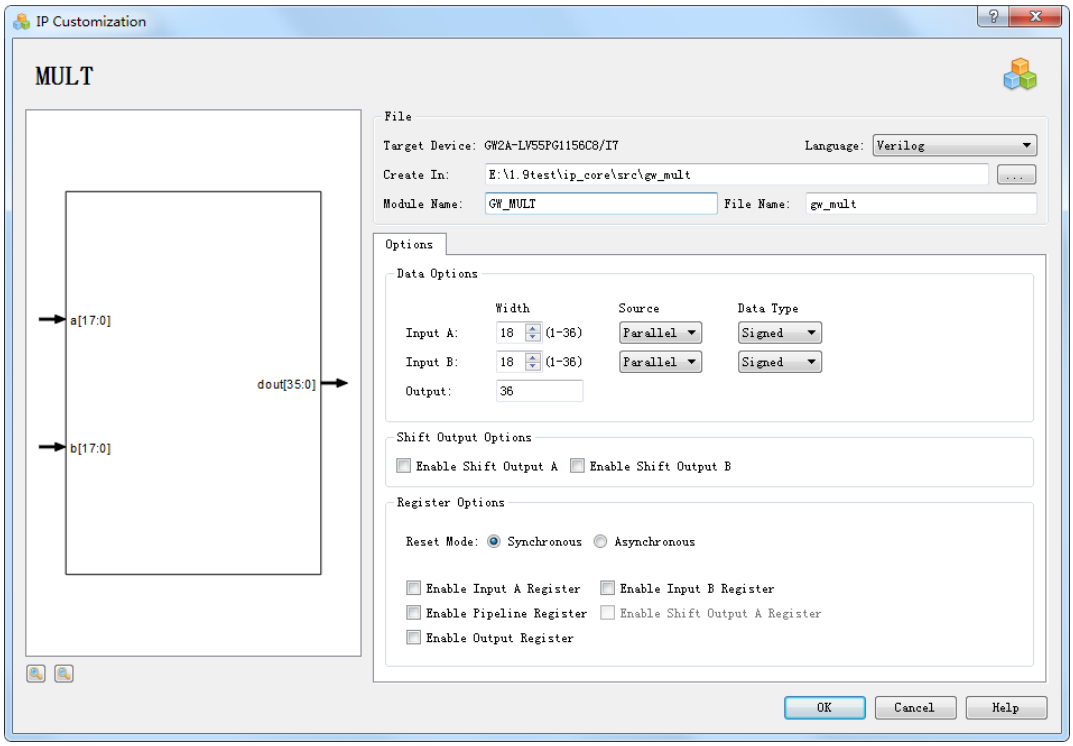
```
GW_MULT your_instance_name(
    .dout(dout_o), //output [35:0] dout
    .a(a_i), //input [17:0] a
    .b(b_i), //input [17:0] b
    .ce(ce_i), //input ce
    .clk(clk_i), //input clk
    .reset(reset_i) //input reset
);
```

MULT Generation Example

If users need to generate a specific MULT IP as follows: 9-bit signed multiplier, Bypass mode. Take the GW2A-LV55PG1156C8/I7 device for instance, the configuration interface is as shown in Figure 3-86. Click "OK" to generate the customized MULT IP design files.

The directory where the MULT IP design file is generated is the path set in "Create In".

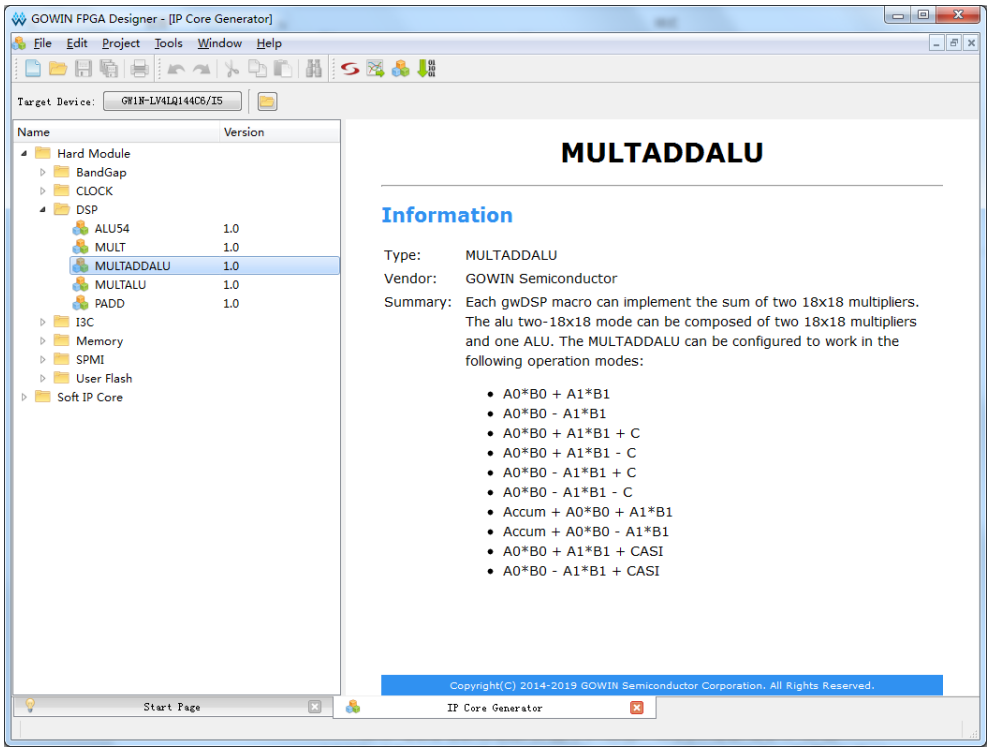
Figure 3-86 MULT IP Customization Configuration



3.2.3 MULTADDALU

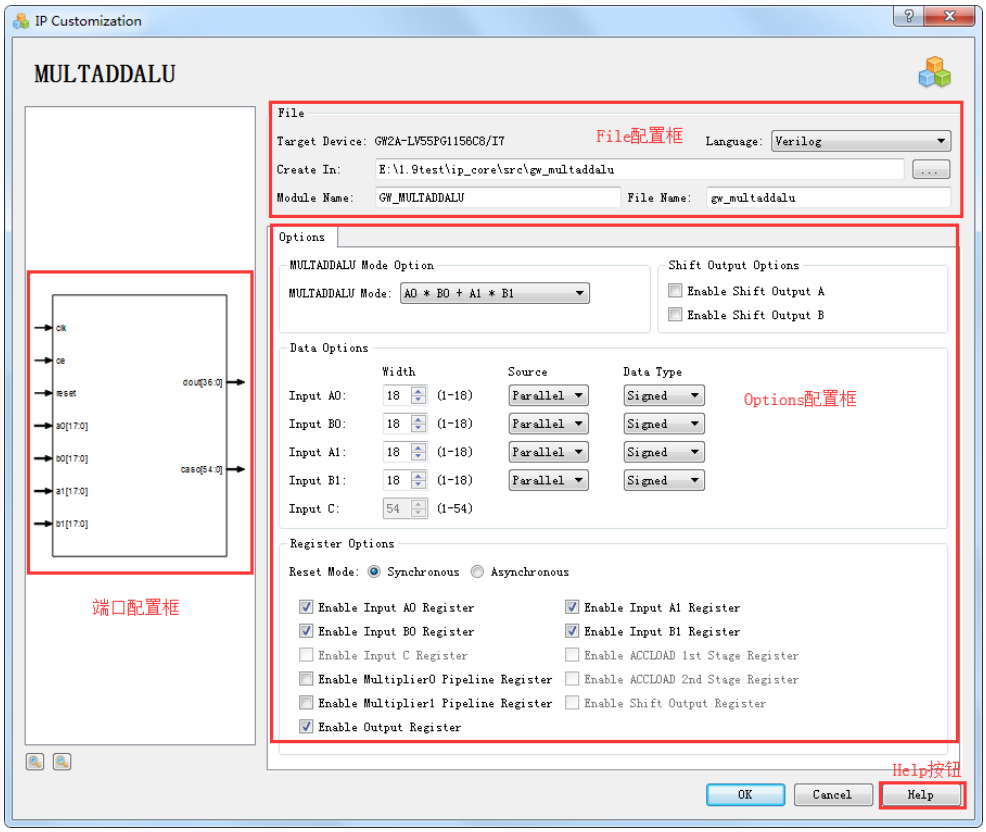
Each DSP macro can implement the sum of two 18x18 multipliers. Click "MULTADDALU" in the IP Core Generator interface. A brief introduction to the MULTADDALU will be displayed on the right of the screen, as shown in Figure 3-87.

Figure 3-87 MULTADDALU Summary Information



Double click the "MULTADDALU" to open the "IP Customization" window. This includes the "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button, as shown in Figure 3-88.

Figure 3-88 IP Customization of MULTADDALU



1. File Configuration

The file configuration includes the basic information related to the MULTADDALU instantiation file, as shown in Figure 3-88.

The MULTADDALU file configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP > File Configuration.

2. Options Configuration

Options configuration mainly includes configuration information related to the MULTADDALU instantiation file, as shown in Figure 3-88.

- MULTADDALU Mode Option: Allows users to select the operation modes. The MULTADDALU can be configured to work in the following operation modes:
  - A0*B0 + A1*B1
  - A0*B0 - A1*B1
  - A0*B0 + A1*B1 + C
  - A0*B0 + A1*B1 - C
  - A0*B0 - A1*B1 + C
  - A0*B0 - A1*B1 - C
  - Accum + A0*B0 + A1*B1
  - Accum + A0*B0 - A1*B1
  - A0*B0 + A1*B1 + CASI
  - A0*B0 - A1*B1 + CASI；
- The configuration of MULTADDALU Data Options and Register Options is similar to that of MULT. For the detailed configuration instructions, please refer to and 3.2.2MULT.

3. Ports Configuration Diagram

The ports configuration diagram displays the current IP Core configuration result. The Input/Output bit-width updates in real time based on "Data Options" and "Register Options" configuration, as shown in Figure 3-88.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-89.

Figure 3-89 Help

MULTADDALU

Information

Type:	MULTADDALU
Vendor:	GOWIN Semiconductor
Summary:	Each gwDSP macro can implement the sum of two 18x18 multipliers. The alu two-18x18 mode can be composed of two 18x18 multipliers and one ALU. The MULTADDALU can be configured to work in the following operation modes: • A0*B0 + A1*B1 • A0*B0 - A1*B1 • A0*B0 + A1*B1 + C • A0*B0 + A1*B1 - C • A0*B0 - A1*B1 + C • A0*B0 - A1*B1 - C • Accum + A0*B0 + A1*B1 • Accum + A0*B0 - A1*B1 • A0*B0 + A1*B1 + CASI • A0*B0 - A1*B1 + CASI

Options

Option	Description
MULTADDALU Mode Option	**MULTADDALU Mode** - Set one of the MULTADDALU operation modes.
Shift Output Options	**Enable Shift Output A** - Enable or disable the shift out port A of the DSP.
	Enable Shift Output B - Enable or disable the shift out port B of the DSP.
Data Options	**Input A0 Width** - Set the size of the first item in the first multiplication.
	Input B0 Width - Set the size of the second item in the first multiplication.
	Input A1 Width - Set the size of the first item in the second multiplication.
	Input B1 width - Set the size of the second item in the second multiplication.
	Input C width - Set the size of input C.
	Source - Set the source of the input A0/B0/A1/B1 as Parallel or Shift.
	Data Type - Set the data format of the input A0/B0/A1/B1 as signed or unsigned.
Register Options	**Reset Mode** - Set whether the reset mode is synchronous or asynchronous.
	Enable ... Register - Enable or disable registers. For example, if you choose Enable Input A0 Register, the input data will go through one register.

The Help page contains the IP Core general description, and a brief introduction to the "Data Options" and "Register Options".

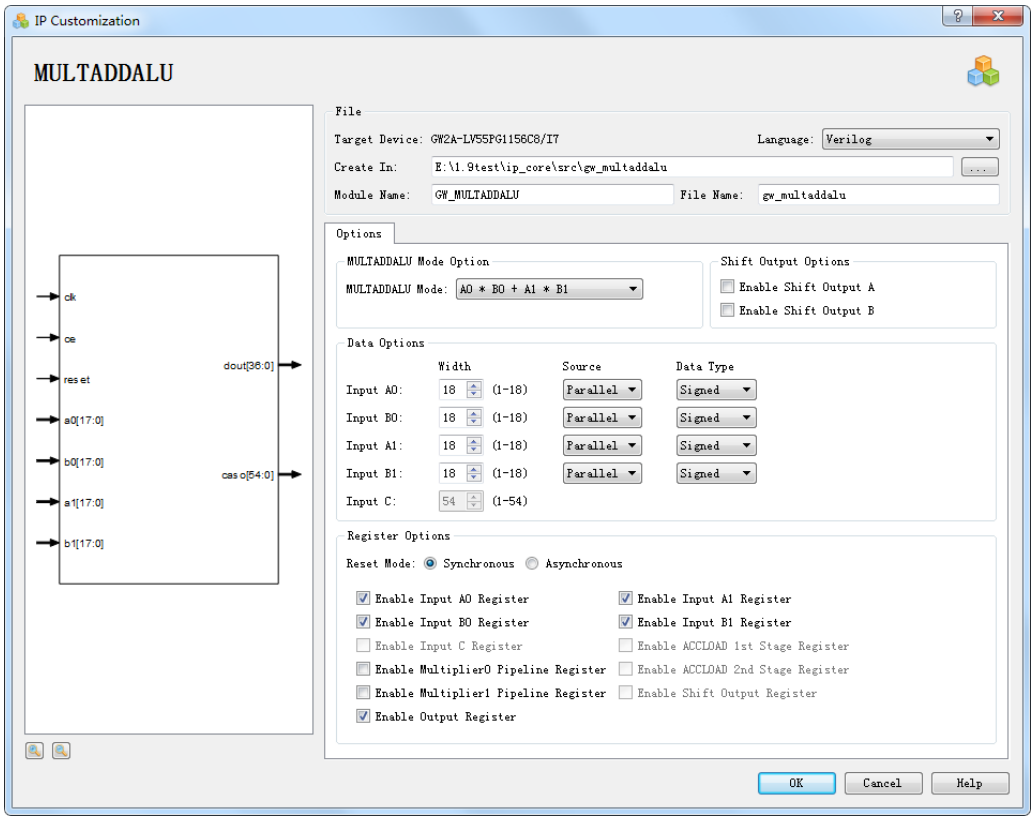
IP Generation Files

As shown in Figure 3-90, after customizing the IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate Gowin Primitive MULTADDALU design file "gw_multaddalu.v";
- The Instantiation template file for the IP design file "gw_ multaddalu _tmp.v";
- The configuration files for the Gowin Primitive SP instantiation "gw_multaddalu.ipc".

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

Figure 3-90 Configured IP Customization



Design File for the Gowin Primitive MULTADDALU Instantiation

MULTADDALU design file instantiation is a complete Verilog module. MULTADDALU instantiation is generated according to the MULTADDALU configuration that is displayed in the "IP Customization" window, as shown in Figure 3-91.

Figure 3-91 MULTADDSUM Design File Instantiation

```
module GW_MULTADDALU (dout, caso, a0, b0, a1, b1, ce, clk, reset);

output [36:0] dout;
output [54:0] caso;
input [17:0] a0;
input [17:0] b0;
input [17:0] a1;
input [17:0] b1;
input ce;
input clk;
input reset;

wire [16:0] dout_w;
wire [17:0] soa_w;
wire [17:0] sob_w;
wire gw_vcc;
wire gw_gnd;

assign gw_vcc = 1'b1;
assign gw_gnd = 1'b0;

MULTADDALU18X18 multaddalu18x18_inst (
    .DOUT({dout_w[16:0],dout[36:0]}),
    .CASO(caso),
    .SOA(soa_w),
    .SOB(sob_w),
    .C({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_
    .A0(a0),
    .B0(b0),
    .A1(a1),
    .B1(b1),
    .ASIGN({gw_vcc,gw_vcc}),
    .BSIGN({gw_vcc,gw_vcc}),
    .CASI({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,
    .ACCLOAD(gw_gnd),
    .SIA({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd}),
    .SIB({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd}),
    .CE(ce),
    .CLK(clk),
    .RESET(reset),
    .ASEL({gw_gnd,gw_gnd}),
    .BSEL({gw_gnd,gw_gnd})
);

defparam multaddalu18x18_inst.A0REG = 1'b1;
defparam multaddalu18x18_inst.B0REG = 1'b1;
defparam multaddalu18x18_inst.A1REG = 1'b1;
defparam multaddalu18x18_inst.B1REG = 1'b1;
defparam multaddalu18x18_inst.CREG = 1'b0;
defparam multaddalu18x18_inst.PIPE0_REG = 1'b0;
defparam multaddalu18x18_inst.PIPE1_REG = 1'b0;
defparam multaddalu18x18_inst.OUT_REG = 1'b1;
defparam multaddalu18x18_inst.ASIGN0_REG = 1'b0;
defparam multaddalu18x18_inst.ASIGN1_REG = 1'b0;
defparam multaddalu18x18_inst.ACCLOAD_REG0 = 1'b0;
defparam multaddalu18x18_inst.ACCLOAD_REG1 = 1'b0;
defparam multaddalu18x18_inst.BSIGN0_REG = 1'b0;
defparam multaddalu18x18_inst.BSIGN1_REG = 1'b0;
defparam multaddalu18x18_inst.SOA_REG = 1'b0;
defparam multaddalu18x18_inst.B_ADD_SUB = 1'b0;
defparam multaddalu18x18_inst.C_ADD_SUB = 1'b0;
defparam multaddalu18x18_inst.MULTADDALU18X18_MODE = 1;
defparam multaddalu18x18_inst.MULT_RESET_MODE = "SYNC";

endmodule //GW_MULTADDALU
```

Instantiation Template File for the IP Design File

For practical use, the IP Core Generator generates the template file while generating MULTADDALU design file instantiation, as shown in Figure 3-92.

Figure 3-92 Instantiation Template File for the IP Design File

```
GW_MULTADDALU your_instance_name(
    .dout(dout_o), //output [36:0] dout
    .caso(caso_o), //output [54:0] caso
    .a0(a0_i), //input [17:0] a0
    .b0(b0_i), //input [17:0] b0
    .a1(a1_i), //input [17:0] a1
    .b1(b1_i), //input [17:0] b1
    .ce(ce_i), //input ce
    .clk(clk_i), //input clk
    .reset(reset_i) //input reset
);
```

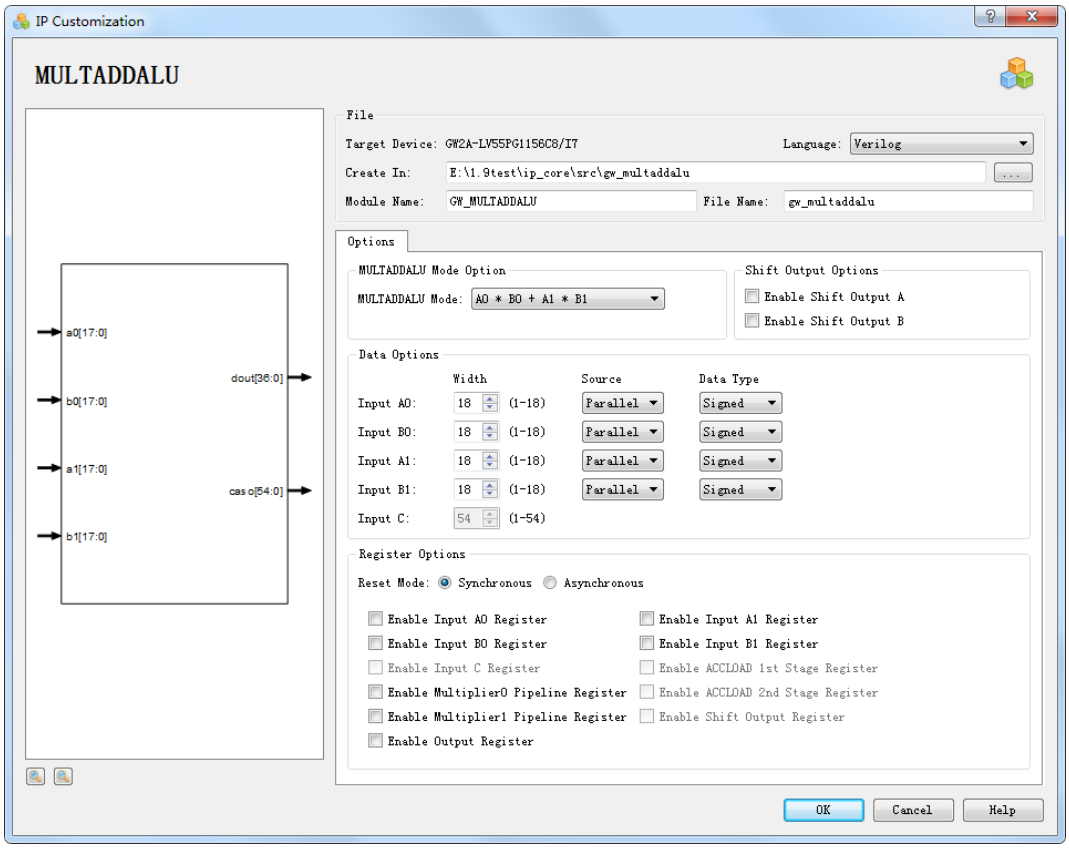
MULTADDALU Generation Example

If users need to generate a specific MULTADDALU IP as follows: Two sums of 18-bit signed multipliers, synchronous reset and Bypass mode. Take the GW2A-LV55PG1156C8/I7 device for instance, the configuration interface is as shown in and Figure 3-93. Click "OK" to generate the

customized MULTADDALU IP design files.

The directory where the MULTADDALU IP design file is generated is the path set in "Create In".

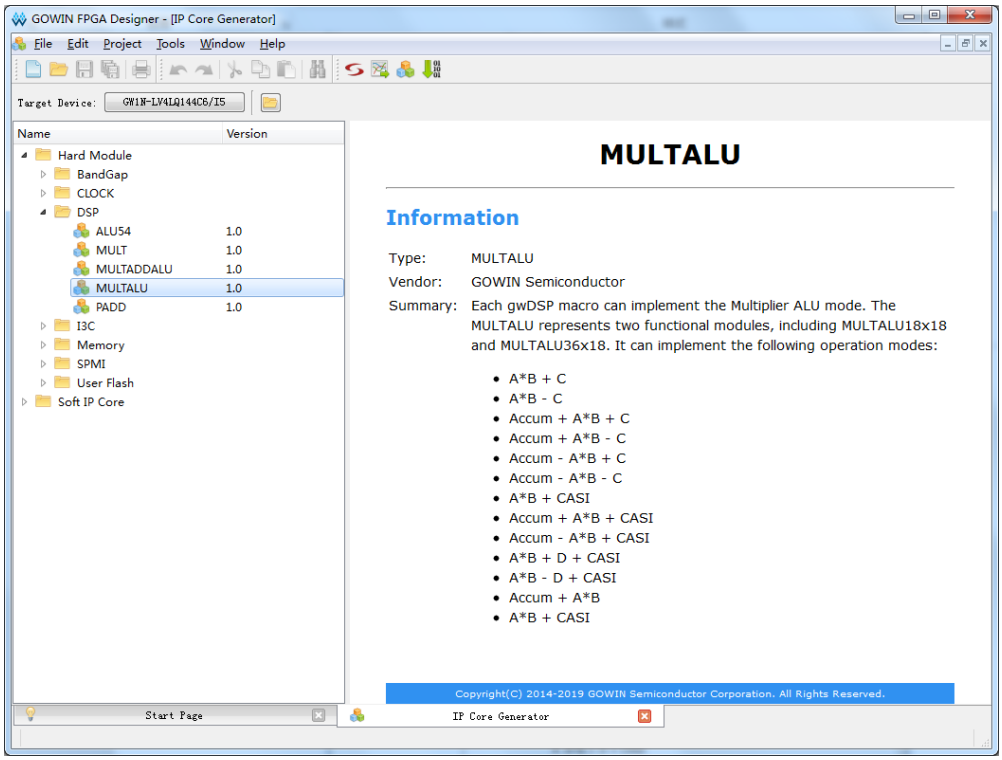
Figure 3-93 MULTADDALU IP Customization Configuration



3.2.4 MULTALU

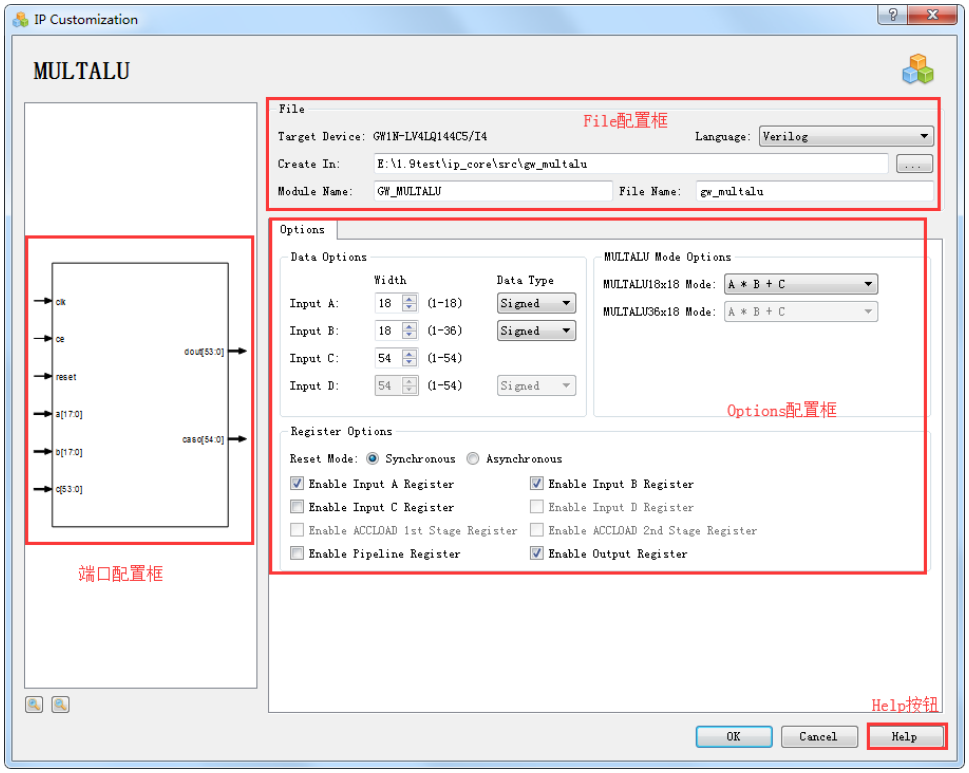
MULTALU can implement the Multiplier ALU mode. Click "MULTALU" on the IP Core Generator interface. A brief introduction to the MULTALU will be displayed on the right of the screen, as shown in Figure 3-94.

Figure 3-94 MULTALU Summary Infirmation



Double click the "MULTALU" to open the "IP Customization" window. This includes the "File" configuration, "Options" configuration, port configuration diagram, and th "Help" button, as shown in Figure 3-95.

Figure 3-95 IP Customization of ROM



1. File Configuration

The file configuration includes the basic information related to the MULTALU instantiation file, as shown in Figure 3-95.

The MULTALU file configuration is similar to that of SP. For the detailed configuration, please refer to 3.1Block Memory > 3.1.1SP> File Configuration.

2. Options Configuration

Options configuration includes configuration information related to the MULTALU instantiation file, as shown in Figure 3-95.

- MULTALU Mode Option:

MULTALU can generate two modules according to the input port width: MULTALU36X18 or MULTALU18X18. When the Input A width and Input B width are less than or equal to 18, the MULTALU36X18 mode will be greyed-out, and MULTALU18X18 mode can be configured as:

  - A*B + C
  - A*B - C
  - Accum + A*B + C
  - Accum + A*B - C
  - Accum - A*B + C
  - Accum - A*B - C
  - A*B + CASI
  - Accum + A*B + CASI
  - Accum - A*B + CASI
  - A*B + D + CASI
  - A*B - D + CASI

- When Input B width is greater than 18, the MULTALU18X18 mode will be greyed out, and the MULTALU36X18 mode can be configured as:
  - A*B + C
  - A*B - C
  - Accum + A*B
  - A*B + CASI
- The configuration of the MULTALU Data Options and Register Options is similar to that of MULT. For the detailed configuration instructions, please refer to 3.2.2MULT.

3. Ports Configuration Diagram

The ports configuration diagram displays the current IP Core configuration result. Input/Output bit-width updates in real time based on the "Options" configuration, as shown in Figure 3-95.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-96.

Figure 3-96 Help

MULTALU

Information

Type:	MULTALU
Vendor:	GOWIN Semiconductor
Summary:	Each gwDSP macro can implement the Multiplier ALU mode. The MULTALU represents two functional modules, including MULTALU18x18 and MULTALU36x18. It can implement the following operation modes: • A*B + C • A*B - C • Accum + A*B + C • Accum + A*B - C • Accum - A*B + C • Accum - A*B - C • A*B + CASI • Accum + A*B + CASI • Accum - A*B + CASI • A*B + D + CASI • A*B - D + CASI • Accum + A*B • A*B + CASI

Options

Option	Description
Data Options	**Input A Width** - Set the size of the first item in the multiplication.
	Input B Width - Set the size of the second item in the multiplication.
	Input C Width - Set the size of input C.
	Input D width - Set the size of input D.
	Data Type - Set the data format of the input A/B/D as signed or unsigned.
MULTALU Mode Options	**MULTALU18x18 Mode** - Set one of the MULTALU18X18 operation modes, the option is available only when widthB <= 18.
	MULTALU36x18 Mode - Set one of the MULTALU36X18 operation modes, the option is available only when widthB > 18.
Register Options	**Reset Mode** - Set whether the reset mode is synchronous or asynchronous.
	Enable ... Register - Enable or disable registers. For example, if you choose Enable Input A Register, the input data will go through one register.

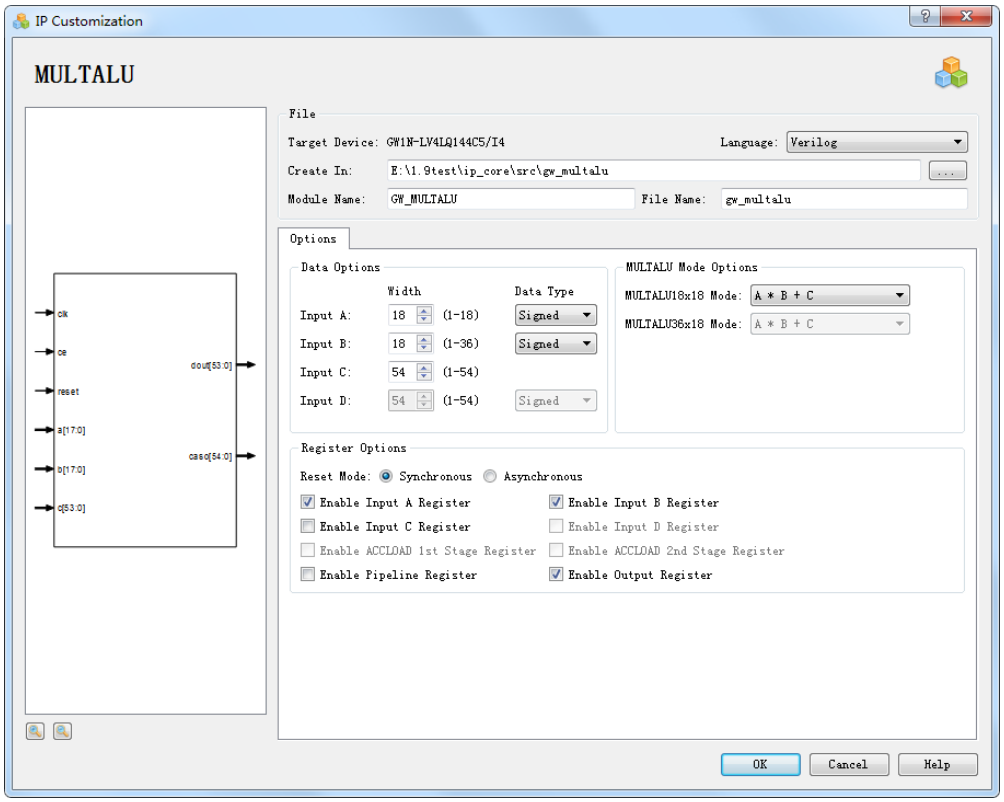
IP Generation Files

As shown in Figure 3-97, after customizing the IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate Gowin Primitive MULTALU design file "gw_multalu.v";
- The instantiation template file for the IP design file "gw_ multalu _tmp.v";
- The configuration file for the Gowin Primitive MULTALU instantiation "gw_multalu.ipc".

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

Figure 3-97 Configured IP Customization



Design File for the Gowin Primitive MULTALU Instantiation

MULTALU design file instantiation is a complete Verilog module. MULTALU instantiation is generated according to the MULTALU configuration that is displayed in the "IP Customization" window, as shown in Figure 3-98.

Figure 3-98 MULTALU Design File Instantiation

```
module GW_MULTALU (dout, caso, a, b, c, ce, clk, reset);

output [53:0] dout;
output [54:0] caso;
input [17:0] a;
input [17:0] b;
input [53:0] c;
input ce;
input clk;
input reset;

wire gw_vcc;
wire gw_gnd;

assign gw_vcc = 1'b1;
assign gw_gnd = 1'b0;

MULTALU18X18 multalu18x18_inst (
    .DOUT(dout),
    .CASO(caso),
    .A(a),
    .B(b),
    .C(c),
    .D({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,
    .ASIGN(gw_vcc),
    .BSIGN(gw_vcc),
    .DSIGN(gw_vcc),
    .CASI({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_g
    .ACCLOAD(gw_gnd),
    .CE(ce),
    .CLK(clk),
    .RESET(reset)
);

defparam multalu18x18_inst.AREG = 1'b1;
defparam multalu18x18_inst.BREG = 1'b1;
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'b0;
defparam multalu18x18_inst.ACCLOAD_REG1 = 1'b0;
defparam multalu18x18_inst.B_ADD_SUB = 1'b0;
defparam multalu18x18_inst.C_ADD_SUB = 1'b0;
defparam multalu18x18_inst.MULTALU18X18_MODE = 0;
defparam multalu18x18_inst.MULT_RESET_MODE = "SYNC";

endmodule //GW_MULTALU
```

Instantiation Template File for the IP Design File

For practical use, the IP Core Generator generates the template file while generating MULTALU design file instantiation, as shown in Figure 3-99.

Figure 3-99 Instantiation Template File for the IP Design File

```
GW_MULTALU your_instance_name(
    .dout(dout_o), //output [53:0] dout
    .caso(caso_o), //output [54:0] caso
    .a(a_i), //input [17:0] a
    .b(b_i), //input [17:0] b
    .c(c_i), //input [53:0] c
    .ce(ce_i), //input ce
    .clk(clk_i), //input clk
    .reset(reset_i) //input reset
);
```

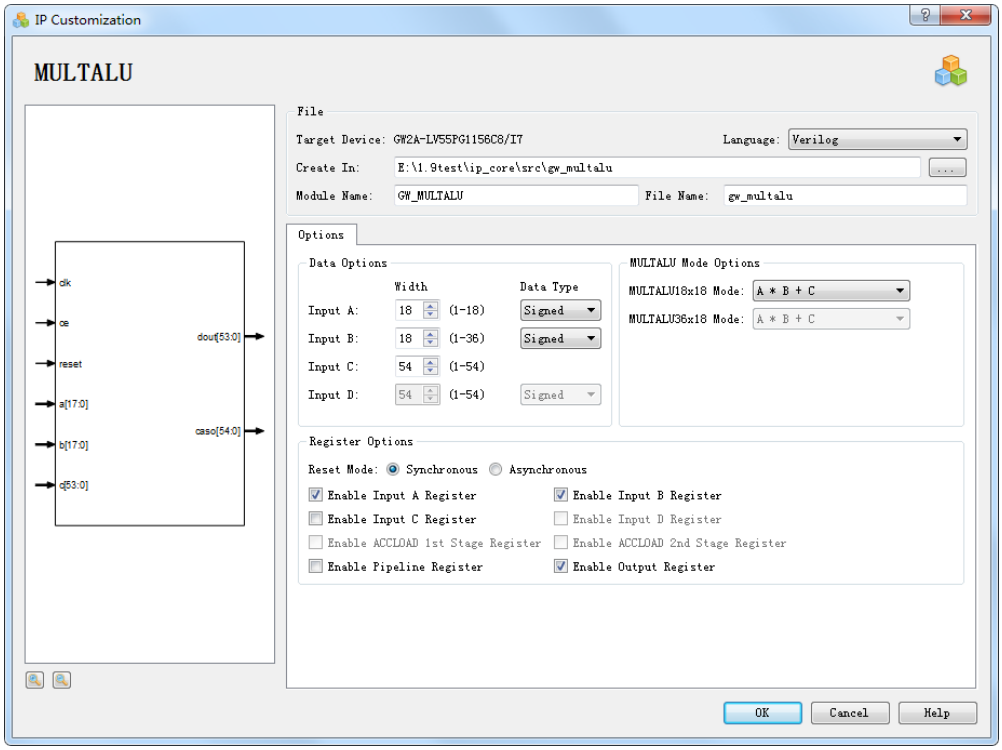
MULTALU Generation Example

If users need to generate a specific MULTALU IP as follows: 18-bit

signed multipliers sum, Register mode and Synchronous. Take the GW2A-LV55PG1156C8/I7 device for instance, the configuration interface is as shown in Figure 3-100. Click "OK" to generate the customized MULTALU IP design files.

The directory where the MULTALU IP design file is generated is the path set in "Create In".

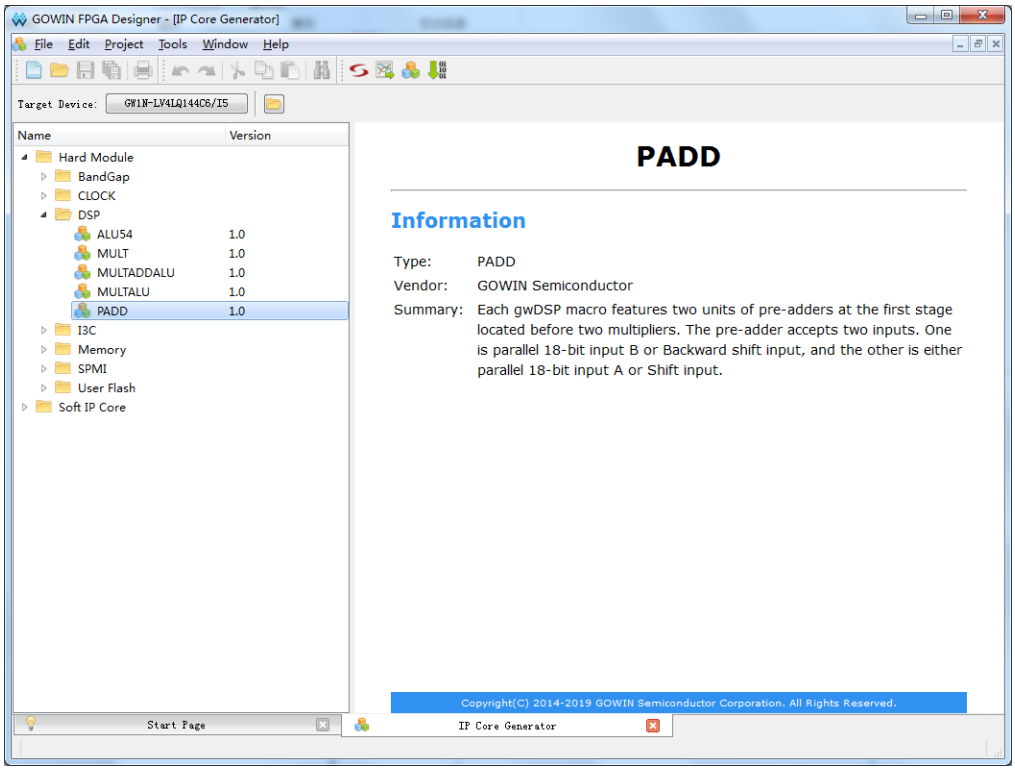
Figure 3-100 MULTALU IP Customization Configuration



3.2.5 PADD

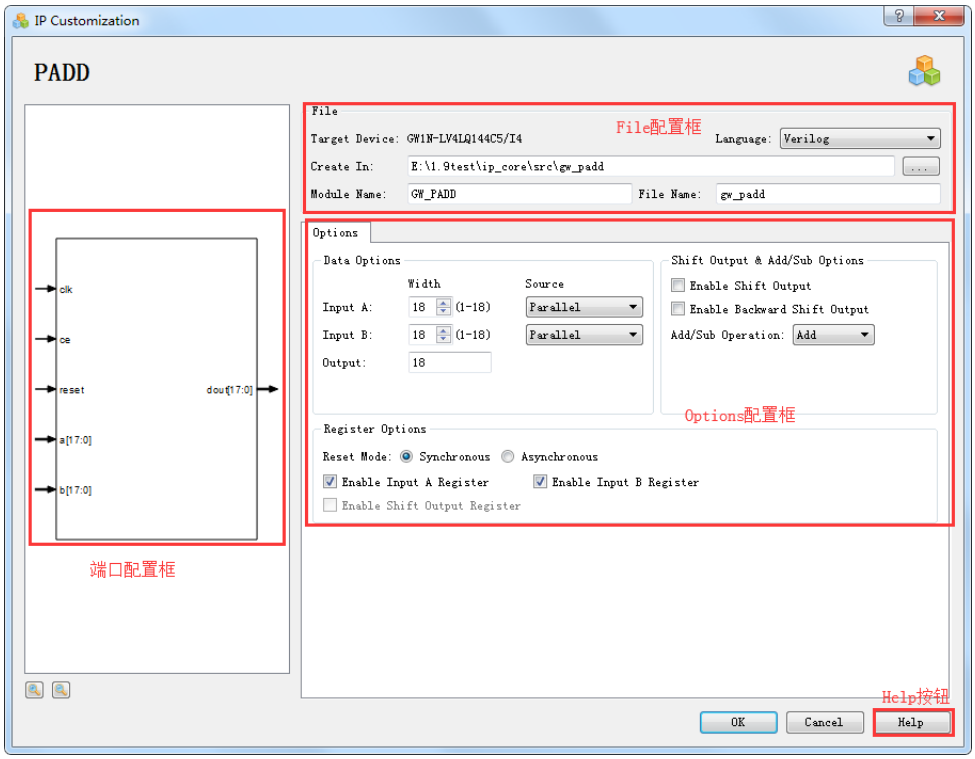
PADD can be configured as a pre-adder, pre-subtracter, or shifter. Click the "PADD" in the IP Core Generator interface. A brief introduction to the PADD will be displayed on the right of the screen, as shown in Figure 3-101.

Figure 3-101 PADD Summary Information



Double click the "PADD" to open the "IP Customization" window. This includes the "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button, as shown in Figure 3-102.

Figure 3-102 IP Customization of PADD



1. File Configuration

The file configuration mainly includes the basic information related to the PADD instantiation file, as shown in Figure 3-102.

The PADD file configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP> File Configuration.

2. Options Configuration

Options configuration includes configuration information related to the PADD instantiation file, as shown in Figure 3-102.

- Data Options: Allows users to configure data options.
  - The maximum data width of the input ports (Input A Width/ Input B Width) is 18;
  - The output width automatically adjusts according to the input width, and the width determines whether PADD9 or PADD18 are generated during instantiation.
  - Input A Source: Users can select Parallel A or Shift;
  - Input B Source: Users can select Parallel or Backward Shift.
- Shift Output and Add/Sub Options: Allows users to enable or disable Shift Output, Backward Shift Output, and add/sub operation.
  - Check "Enable Shift Output" to enable shift output;
  - Check "Enable Backward Shift Output" to enable backward shift output;
  - Configure "Add/Sub Operation" to select if the Adder is in add, subtract, or dynamic mode, or PADD performs add/sub operation by ports signal.
- Register Options: Allows users to configure registers operation mode.
  - Reset Mode: Sets whether the reset mode is synchronous or asynchronous;
  - Enable Input A Register: Allows users to enable or disable Input A register;
  - Enable Input B Register: Allows users to enable or disable Input B register;
  - Enable Output Register: Allows users to enable or disable Output register.

3. Ports Configuration Diagram

The ports configuration diagram displays current IP Core configuration result. The Input/Output bit-width updates in real time based on the "Options" configuration, as shown in Figure 3-102.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-103. The Help page contains a general description of the IP Core, and a brief introduction to the "Options".

Figure 3-103 Help

PADD

Information

Type:	PADD
Vendor:	GOWIN Semiconductor
Summary:	Each gwDSP macro features two units of pre-adders at the first stage located before two multipliers. The pre-adder accepts two inputs. One is parallel 18-bit input B or Backward shift input, and the other is either parallel 18-bit input A or Shift input.

Options

Option	Description
Data Options	**Input A Width** - Set the size of the first item in the Pre-adder.
	Input B Width - Set the size of the second item in the Pre-adder.
	Output Width - Size of the output.
	Input A Source - Set the source of the input A as Parallel or Shift.
	Input B Source - Set the source of the input B as Parallel or Backward Shift.
Shift Output & Add/Sub Options	**Enable Shift Output** - Enable or disable the shift out port of the Pre-adder.
	Enable Backward Shift Output - Enable or disable the backward shift out port of the Pre-adder.
	Add/Sub Operation - Set whether the mode is in add or subtract mode.
Register Options	**Reset Mode** - Set whether the reset mode is synchronous or asynchronous.
	Enable ... Register - Enable or disable registers. For example, if you choose Enable Input A Register, the input data will go through one register.

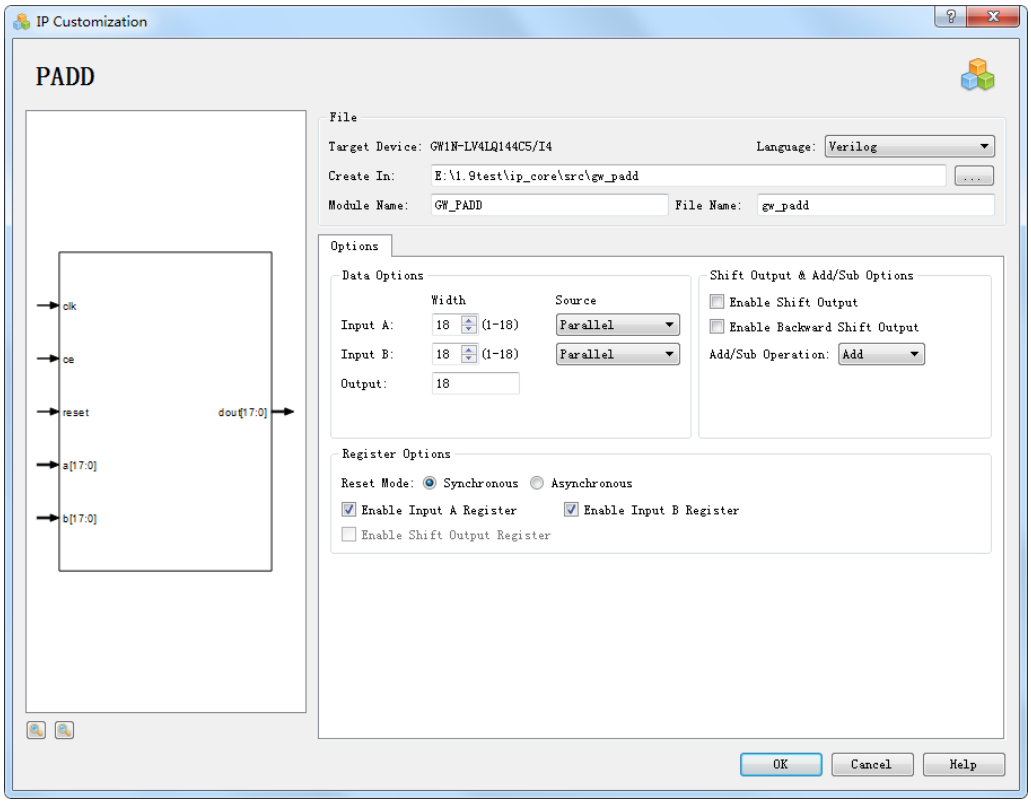
IP Generation Files

As shown in Figure 3-104, after customizing the IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate Gowin Primitive PADD instantiation "gw_padd.v";
- The instantiation template file for the IP design file "gw_padd_tmp.v";
- The configuration file for the Gowin Primitive PADD instantiation "gw_padd.ipc".

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

Figure 3-104 Configured IP Customization



Design File for the Gowin Primitive PADD Instantiation

PADD design file instantiation is a complete Verilog module. PADD instantiation is generated according to the PADD configuration that is displayed in the "IP Customization" window, as shown in Figure 3-105.

Figure 3-105 PADD Design File Instantiation

```
module GW_PADD (dout, a, b, ce, clk, reset);

output [17:0] dout;
input [17:0] a;
input [17:0] b;
input ce;
input clk;
input reset;

wire [17:0] so_w;
wire [17:0] sbo_w;
wire gw_gnd;

assign gw_gnd = 1'b0;

PADD18 padd18_inst (
    .DOUT(dout),
    .SO(so_w),
    .SBO(sbo_w),
    .A(a),
    .B(b),
    .SI({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,
         gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd}),
    .SBI({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,
         gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd}),
    .CE(ce),
    .CLK(clk),
    .RESET(reset),
    .ASEL(gw_gnd)
);

defparam padd18_inst.AREG = 1'b1;
defparam padd18_inst.BREG = 1'b1;
defparam padd18_inst.ADD_SUB = 1'b0;
defparam padd18_inst.PADD_RESET_MODE = "SYNC";
defparam padd18_inst.BSEL_MODE = 1'b0;
defparam padd18_inst.SOREG = 1'b0;

endmodule //GW_PADD
```

Instantiation Template File for the IP Design File

For efficiency purposes, the IP Core Generator generates the template file while generating PADD design file instantiation, as shown in Figure 3-106.

Figure 3-106 Instantiation Template File for the IP Design File

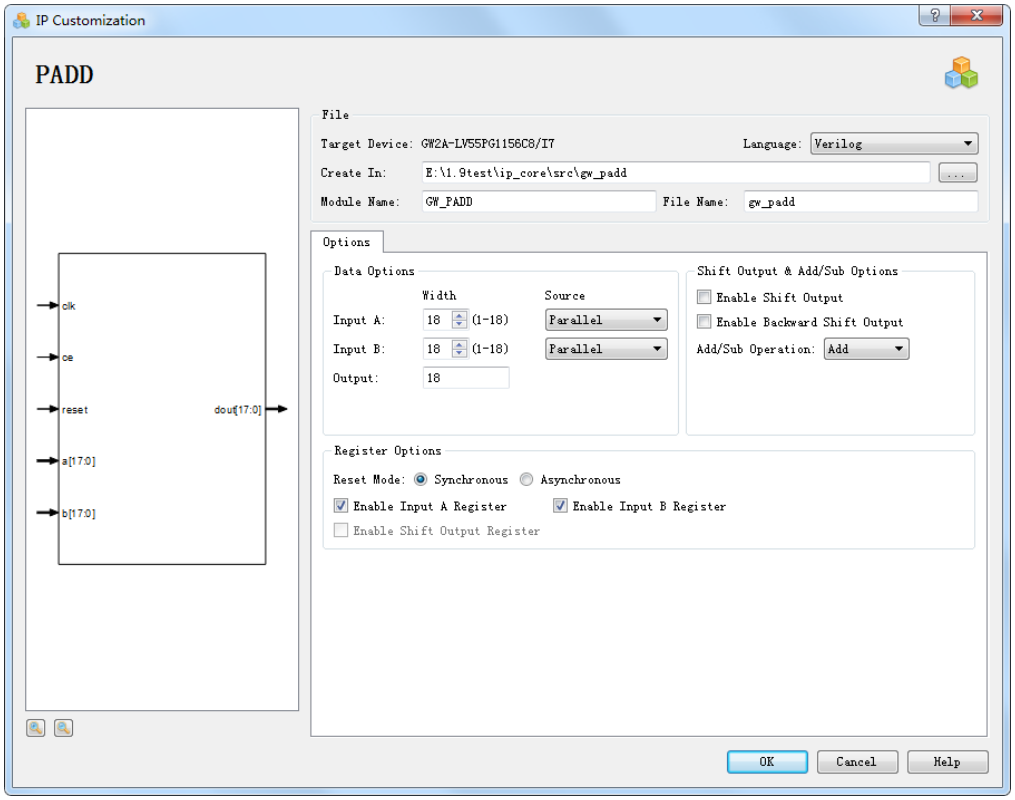
```
GW_PADD your_instance_name(
    .dout(dout_o), //output [17:0] dout
    .a(a_i), //input [17:0] a
    .b(b_i), //input [17:0] b
    .ce(ce_i), //input ce
    .clk(clk_i), //input clk
    .reset(reset_i) //input reset
);
```

PADD Generation Example

If users need to generate a specific PADD IP as follows: Input Width 18, Add operation and Synchronous.Take the GW2A-LV55PG1156C8/I7device for instance, the configuration page is as shown in Figure 3-107. Click "OK" to generate the customized PADD IP design files.

The directory where the PADD IP design file is generated is the path set in "Create In".

Figure 3-107 PADD IP Customization Configuration



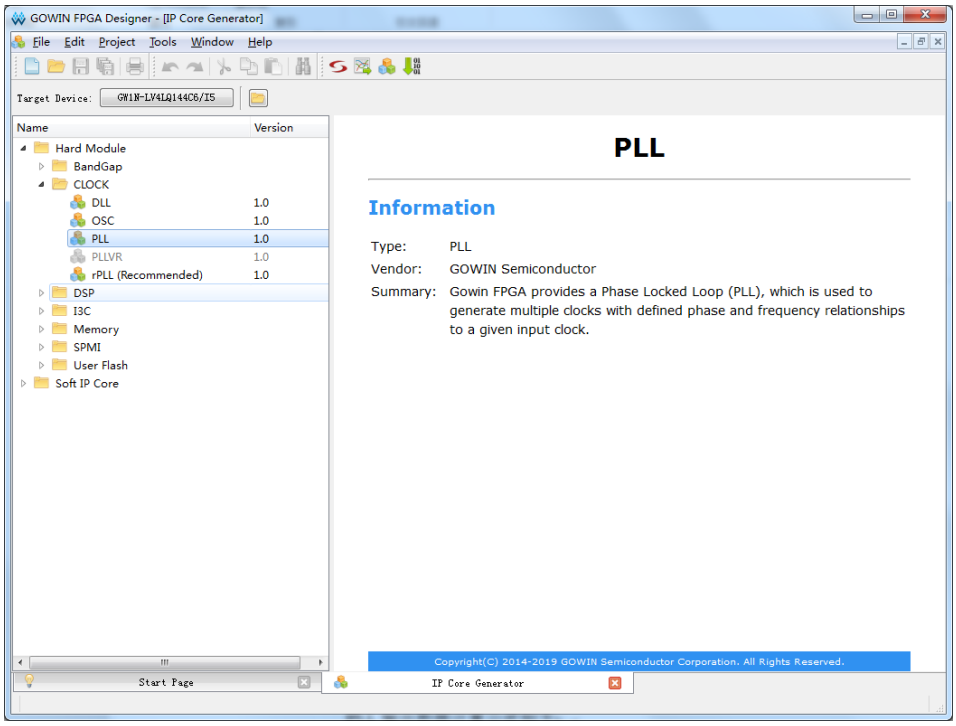
3.3 CLOCK

The CLOCK module currently supports five types of Gowin devices generation: PLL, rPLL, PLLVR, DLL, OSC.

3.3.1 PLL

Based on the "clkin" input, PLL adjusts the clock phase, duty cycle, and frequency (multiplication and division) to output different phases and frequencies. Click "PLL" on the IP Core Generator interface. A brief introduction to the PLL will be displayed on the right of the screen, as shown in Figure 3-108.

Figure 3-108 PLL Summary Information



The formulas for PLL output calculation are as follows:

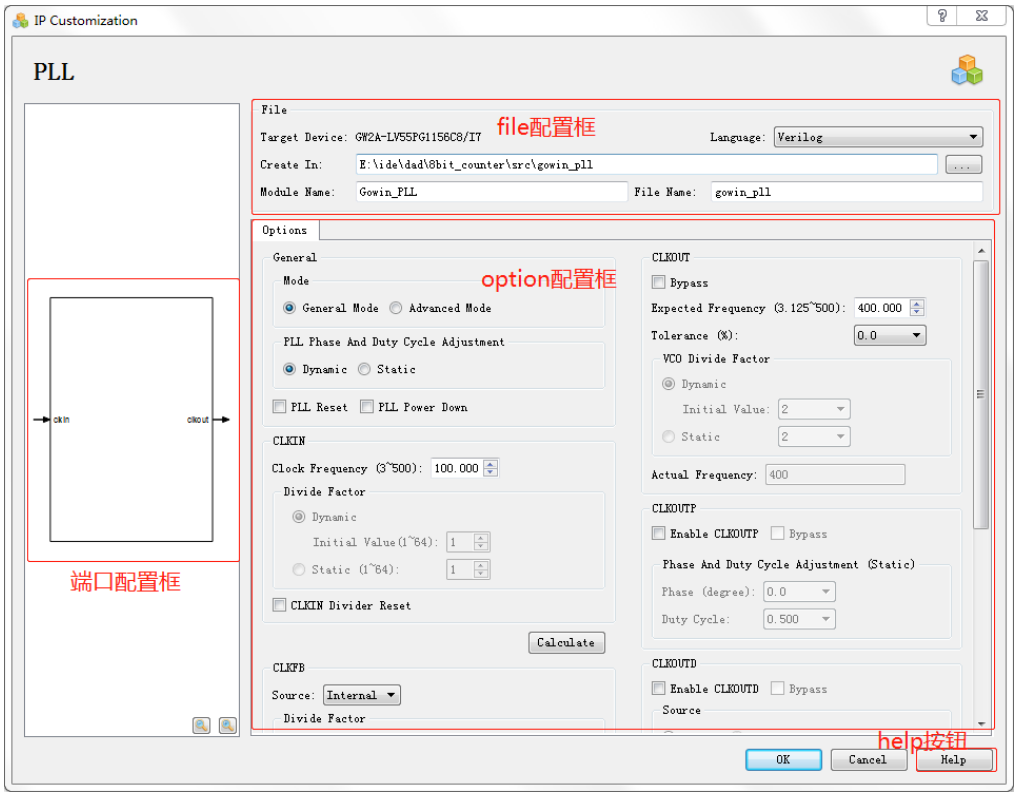
1. $f_{CLKOUT} = (f_{CLKIN}*FDIV)/IDIV$
2. $f_{CLKOUTD} = f_{CLKOUT}/SDIV$
3. $f_{VCO} = f_{CLKOUT}*ODIV$

Note!

- f_{CLKIN}: The frequency of input clock CLKIN;
- f_{CLKOUT}: The frequency of output clock CLKOUT;
- $f_{CLKOUTD}$: The frequency of output clock CLKOUTD, and CLKOUTD is the clock "CLKOUT" after division.
- f_{VCO}: VCO oscillation frequency.

Double click the "PLL" to open the "IP Customization" window. This includes the "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button, as shown in Figure 3-109.

Figure 3-109 IP Customization of PLL



1. File Configuration

The file configuration includes the basic information related to the PLL instantiation file, as shown in Figure 3-109.

The PLL file configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP> File Configuration.

2. Options Configuration

Options configuration includes configuration information related to the PLL instantiation file, as shown in Figure 3-109.

- General: Allows users to select "General Mode" or "Advanced Mode", select "Static Mode" or "Dynamic Mode" for PLL phase and Duty Cycle adjustment, and enable or disable PLL Reset.
  - Mode: Set the IP Core configuration mode as "General Mode" or "Advanced Mode".
  - PLL Phase And Duty Cycle Adjustment: Allows users to select "Static" Mode or "Dynamic" Mode.
  - "PLL Reset": configure PLL Reset enable.
  - "PLL Power Down": Configure the reset_p port to put the PLL in power-saving mode.
- CLKIN: Allows users to set input clock frequency, divide factor, and IDESEL Reset.
  - Clock Frequency: Specify the frequency in MHz. The frequency range is determined by the device selected.
  - Divide Factor: Allows users to set the Divide Factor as "Dynamic" or "Static" in Advanced mode. In Static mode, Divide Factor value can be set as a specific value, which ranges from 1 to 64. If the output frequency of CLKOUT is not

in the range required by the corresponding device, an error message will be displayed when the user clicks "Calculate" or "OK" as shown in Figure 3-110; If the frequency of CLKIN/IDIV is not in the range required by the corresponding device, an error message will be displayed when the user clicks "Calculate" or "OK" as shown in Figure 3-111.

- "CLKIN Divider Reset": Configure the CLKIN Divider Reset port.

- CLKFB: Allows users to set the source of the PLL feedback and divide factor.
  - Source: Specify the source of feedback as Internal or External;
  - Divide Factor: Allows users to set the Divide Factor as "Dynamic" or "Static" in advanced mode. In static mode, Divide Factor value can be set as a specific value, which ranges from 1 to 64. If the configuration is illegal, an error message will be displayed when the user clicks on "Calculate" or "OK", as shown in Figure 3-110
- Enable LOCK: Allows users to select whether to enable LOCK.
- CLKOUT: Allows users to specify the expected frequency tolerance fields of PLL clkout and VCO parameters..
  - Bypass: Allows users to enable/disable clkout bypass;
  - Expected Frequency: Set the output clock frequency in general mode. The frequency range is determined by the device selected.
  - Tolerance(%): Set a tolerance for the CLKOUT expected frequency and actual frequency calculated.
  - VCO Divide Factor: Allows users to set Divide Factor as "Dynamic" or "Static" in advanced mode. In static mode, the Divide Factor value can be set as a specific value, and the range is 2/4/8/16/32/48/64/80/96/112/128. If the configuration is illegal, an error message will be displayed when the user clicks on "Calculate" or "OK", as shown in Figure 3-110.
  - Actual Frequency: Clicking the "Calculate" button displays the actual frequency that the PLL can produce.
- CLKOUTP: Allows users to set Duty Cycle Fine Tuning (Dynamic) ,Phase And Duty Cycle Adjustment (Static) and enable/disable CLKOUTP.
  - Enable CLKOUTP: Enable/disable CLKOUTP;
  - Bypass: Allows users to enable/disable CLKOUTP bypass;
  - Phase And Duty Cycle Adjustment (Static): Configure (Phase [degree]) and (Duty Cycle [*1/16]) in static mode;
- CLKOUTD: Allows users to specify the source, expected frequency, and divide factor of the clock divider, and enable/disable CLKOUTD Reset.
  - Enable CLKOUTD: Usedto enable/disable CLKOUTD;
  - Bypass: Allows users to enable/disable CLKOUTD bypass;
  - Source: Select the source of CLKOUTD as "CLKOUT" or "CLKOUTP";
  - Expected Frequency: Set the output clock frequency in General mode. The frequency range is determined by the

device selected.
  - Tolerance(%): Set a tolerance for the CLKOUTD expected frequency and actual frequency calculated.
  - Divide Factor (2~128): Select the divide factor from the drop-down list in advanced mode. Only even numbers between 2 and 128 can be selected. If an odd number is set, error message will be displayed when the user clicks on "OK", as shown in; Figure 3-112
  - Actual Frequency: Clicking the "Calculate" button displays the actual frequency that the PLL can produce.
- CLKOUTD3: Allows users to set the source of CLKOUTD3.
  - Enable CLKOUTD3: Used to enable/disable CLKOUTD3; Selecting this option will produce a clkoutd3 port in the generated module. It is equal to clkout/3.
  - Source: Select the source of CLKOUTD3 as "CLKOUT" or "CLKOUTP";
- "CLKOUTD/CLKOUTD3 Divider Reset" can be checked when CLKOUTD or CLKOUTD3 enable so as to configure CLKOUTD/CLKOUTD3 Divider reset port.
- Calculate: This tool calculates the Divide Factor settings based on the input/output frequency in general mode. If the actual frequency calculated is different to the expected frequency, an "Error" window will pop up and the illegal value will be marked in red.
  - Figure 3-113 is the error message that is displayed when the actual frequency and expected frequency of CLKOUT are different;
  - Figure 3-114 is the error message that is displayed when the actual frequency and expected frequency of CLKOUTD are different.
  - In advanced mode, the tool calculates the output frequencies based on divide factors.
  - If the calculated results are illegal, an "error" message will pop up, and the illegal value will be marked in red, as shown in Figure 3-115.
  - Otherwise, a success message prompt will be displayed as shown in Figure 3-116.

Figure 3-110 Error - Illegal Configuration of CLKIN/CLKFB Divide Factor

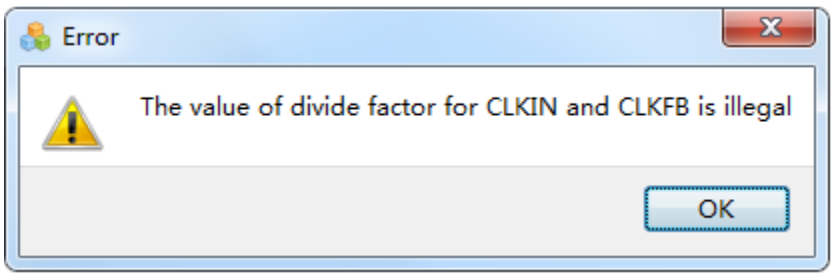


Figure 3-111 Error - Illegal Configuration of CLKIN Divide Factor

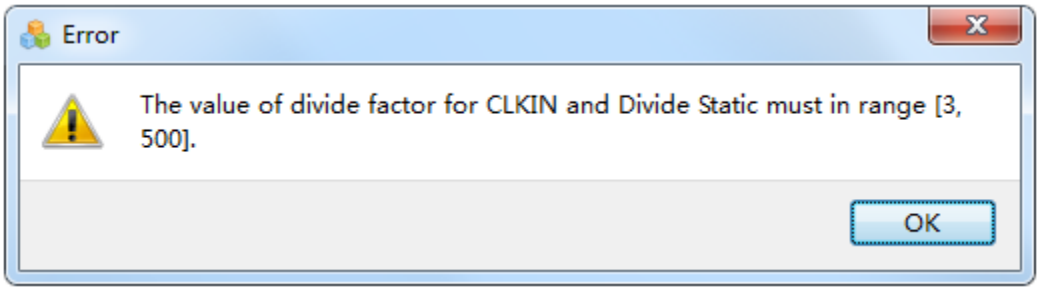


Figure 3-112 Error - Illegal Configuration of CLKOUTD Divide Factor

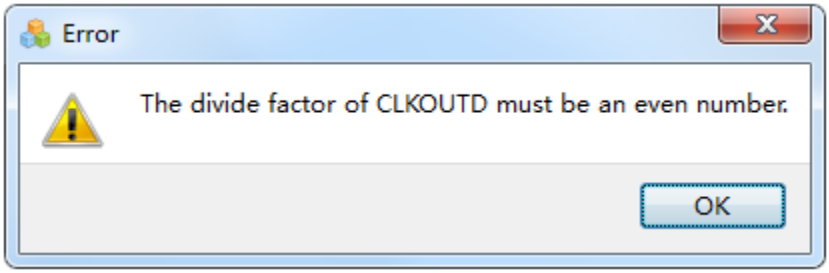


Figure 3-113 Error - Unequal Frequency of CLKOUT

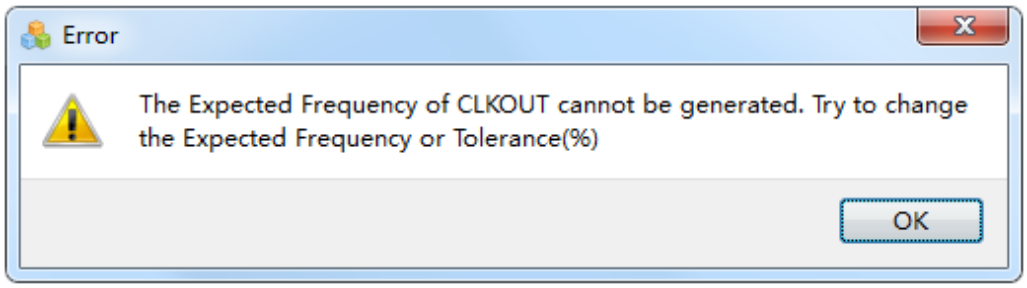


Figure 3-114 Error - Unequal Frequency of CLKOUTD

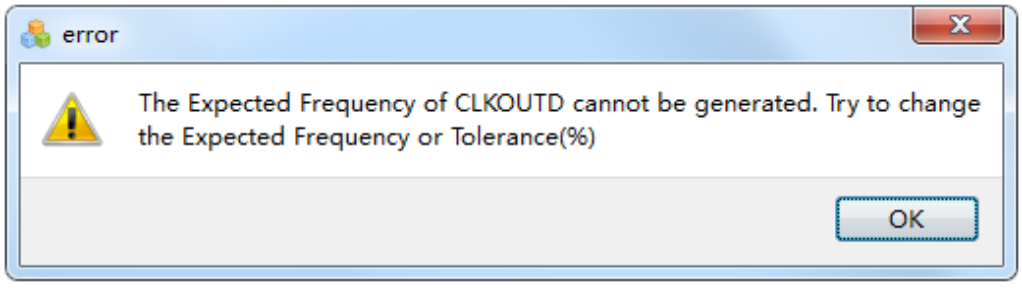


Figure 3-115 Error - Illegal Configuration of VCO

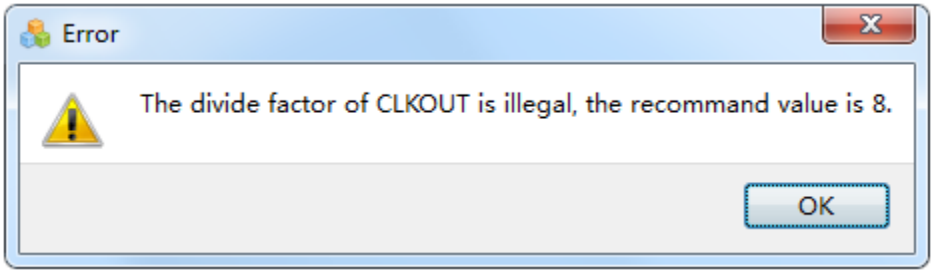
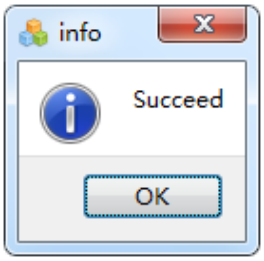


Figure 3-116 Info - Succeed



3. Ports Configuration Diagram

The ports configuration diagram displays the current IP Core configuration result. The number of Input/Output ports updates in real time based on the options configuration, as shown in Figure 3-109.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-117. Help page contains the IP Core general description, and brief introduction of "Options".

Figure 3-117 Help

PLL

Information

Type:	PLL
Vendor:	GOWIN Semiconductor
Summary:	GW FPGA provides a Phase Locked Loop (PLL), which is used to generate multiple clocks with defined phase and frequency relationships to a given input clock.

Options

Option	Description
General	**General Mode** - In this mode, entering input clock frequency and expected frequencies, software will automatically calculate divide factors.
	Advanced Mode - This mode is for advanced users. Allows you to enter input clock frequency and divide factors to achieve expected output frequency.
	PLL Phase And Duty Cycle Adjustment - Allows you to select Static Mode or Dynamic Mode.
	PLL Reset - Provides a reset pin to reset the PLL.
	PLL Power Down - Provides a reset_p port to power down the PLL.
CLKIN	CLKIN is the input reference clock for the PLL.
	Clock Frequency - Specify its frequency in MHz.
	Divide Factor - If in Advanced mode, also choose a divide factor which is from Dynamic or Static mode to achieve the expected output frequency. Static mode means select a static value from the drop-down list as divide factor, while Dynamic mode means that choose the value of port idsel as dynamic divide factor. When the Dynamic mode is selected, the user needs to set an initial value.
	CLKIN Divider Reset - Provides a reset_i port to reset the input clock divider.
CLKFB	**Source** - Specify the source of feedback.
	Divide Factor - In Advanced mode, the divide factor in the feedback path can be selected from port fbdsel or from the drop-down list. In General mode, the divide factor is shown when the "Calculate" button is clicked. When the Dynamic mode is selected, the user needs to set an initial value.
LOCK	**Enable LOCK** - Selecting this option will produce the lock port in the generated module.
CLKOUT	**Bypass** - The bypass option means clkout = clkin, it connects the output to the input, bypassing the PLL circuit. Bypassing CLKOUT disables the CLKOUT expected frequency and tolerance fields. If both CLKOUT and CLKOUTP are in Bypass, then everything is disabled except the CLKIN frequency option.
	The following options are not available when CLKOUT is in Bypass mode.
	VCO Divide Factor - In General mode, VCO Divide Factors cannot be selected. Clicking the "Calculate" button displays the actual values. In Advanced mode, Dynamic or Static mode can be selected. When the Dynamic mode is selected, the user needs to set an initial value.
	Expected Frequency - In General mode, set the output clock frequency.
	Tolerance - Set a tolerance for the clkout frequency, as a percentage of requested frequency. Since the divide factors can only take a certain set of values, not all frequencies can be generated. The tolerance value is used to guide the tool to select a suitable divide factor.
	Actual Frequency - Clicking the "Calculate" button displays the actual frequency that the PLL can produce.

CLKOUTP	**Enable CLKOUTP** – Selecting this option will produce the clkoutp port in the generated module. Clkoutp has the same frequency as clkout and it has the specified phase relation with clkout.
	Bypass – This option will connect the output to the input, bypassing the PLL circuit. Bypassing clkoutp disables its frequency, tolerance, and phase shift fields.
	PLL Phase And Duty Cycle Adjustment(Static)
	Phase - You can select phase shift in 22.5-degree increments from 0 to 360.
	Duty Cycle - Allows duty cycle selection in 1/16 increments.
CLKOUTD	**Enable CLKOUTD** - Selecting this option will produces clkoutd port in the generated module. In general mode, select the expected frequency. In advanced mode, select the divide factor for the expected frequency.
	Bypass - The bypass option means clkoutd = clkin, it connects the output to the input, bypassing the PLL circuit.
	Source - If clkoutd is enabled, you can select CLKOUT or CLKOUTP as the source of clkoutd. If CLKOUTP is not enabled, CLKOUT is used as the source.
	Expected Frequency - In General mode, set the CLKOUTD Frequency. This option is disabled in advanced mode.
	Divide Factor - In General mode, the divide factor cannot be selected. Clicking the "Calculate" button the tool will automatically displays the actual values. In Advanced mode, select the CLKOUTD divide factor from the drop-down list.
	Tolerance - Set a tolerance for the CLKOUTD frequency, as a percentage of requested frequency. Since the divide factors can only take a certain set of values, not all frequencies can be generated. The tolerance value is used to guide the tool to select a suitable divide factor.
	Actual Frequency - Clicking the "Calculate" button displays the actual frequency that the PLL can produce.
CLKOUTD3	**Enable CLKOUTD3** - Selecting this option will produces clkoutd3 port in the generated module, and it is equal to clkout/3.
	Source - If CLKOUTD3 is enabled, you can select CLKOUT or CLKOUTP as the source of CLKOUTP. If CLKOUTP is not enabled, CLKOUT is used as the source.
CLKOUTD/CLKOUTD3 Divider Reset	Select it if CLKOUTD or CLKOUTD3 divider reset is required as an input port. When CLKOUTD or CLKOUTD3 is enabled and this option is selected, an input port reset_s is generated.
Calculate	General Mode: The tool calculates Divide Factor settings based on input/output frequency. Advanced Mode: The tool calculates output frequencies based on divide factors.

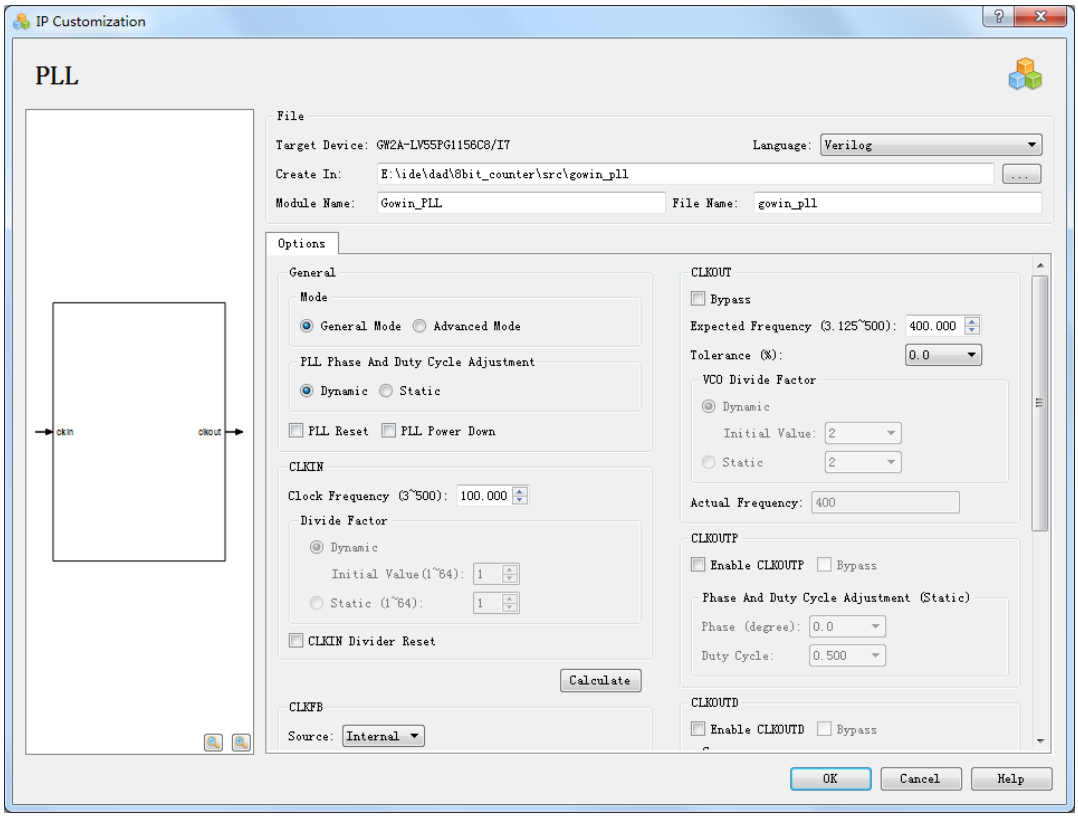
IP Generation Files

As shown in Figure 3-118, after customizing the IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate Gowin Primitive PLL design file "gowin_pll.v";
- The instantiation template file for the IP design file "gowin_pll_tmp.v";
- The configuration files for the Primitive PLL instantiation "gowin_pll.ipc".

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

Figure 3-118 Configured IP Customization



Design File for the Gowin Primitive PLL Instantiation

PLL design file instantiation is a complete Verilog module. PLL instantiation is generated according to the PLL configuration that is displayed in the "IP Customization" window, as shown in Figure 3-119.

Figure 3-119 PLL Design File Instantiation

```
module Gowin_PLL (clkout, clkin);
output clkout;
input clkin;
wire lock_o;
wire clkoutp_o;
wire clkoutd_o;
wire clkoutd3_o;
wire gw_gnd;
assign gw_gnd = 1'b0;

PLL pll_inst (
    .CLKOUT(clkout),
    .LOCK(lock_o),
    .CLKOUTP(clkoutp_o),
    .CLKOUTD(clkoutd_o),
    .CLKOUTD3(clkoutd3_o),
    .RESET(gw_gnd),
    .RESET_P(gw_gnd),
    .RESET_I(gw_gnd),
    .RESET_S(gw_gnd),
    .CLKIN(clkin),
    .CLKFB(gw_gnd),
    .FBDSEL({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd}),
    .IDSEL({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd}),
    .ODSEL({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd}),
    .PSDA({gw_gnd,gw_gnd,gw_gnd,gw_gnd}),
    .DUTYDA({gw_gnd,gw_gnd,gw_gnd,gw_gnd}),
    .FDLY({gw_gnd,gw_gnd,gw_gnd,gw_gnd})
);

defparam pll_inst.FCLKIN = "100";
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 = 3;
defparam pll_inst.DYN_ODIV_SEL = "false";
defparam pll_inst.ODIV_SEL = 2;
defparam pll_inst.PSDA_SEL = "0000";
defparam pll_inst.DYN_DA_EN = "true";
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 = "internal";
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 = "GW2A-55";

endmodule //Gowin_PLL
```

Instantiation Template File for the IP Design File

For the practical use, the IP Core Generator generates the template file while generating the PLL design file instantiation, as shown in Figure 3-120.

Figure 3-120 Instantiation Template File for the IP Design File

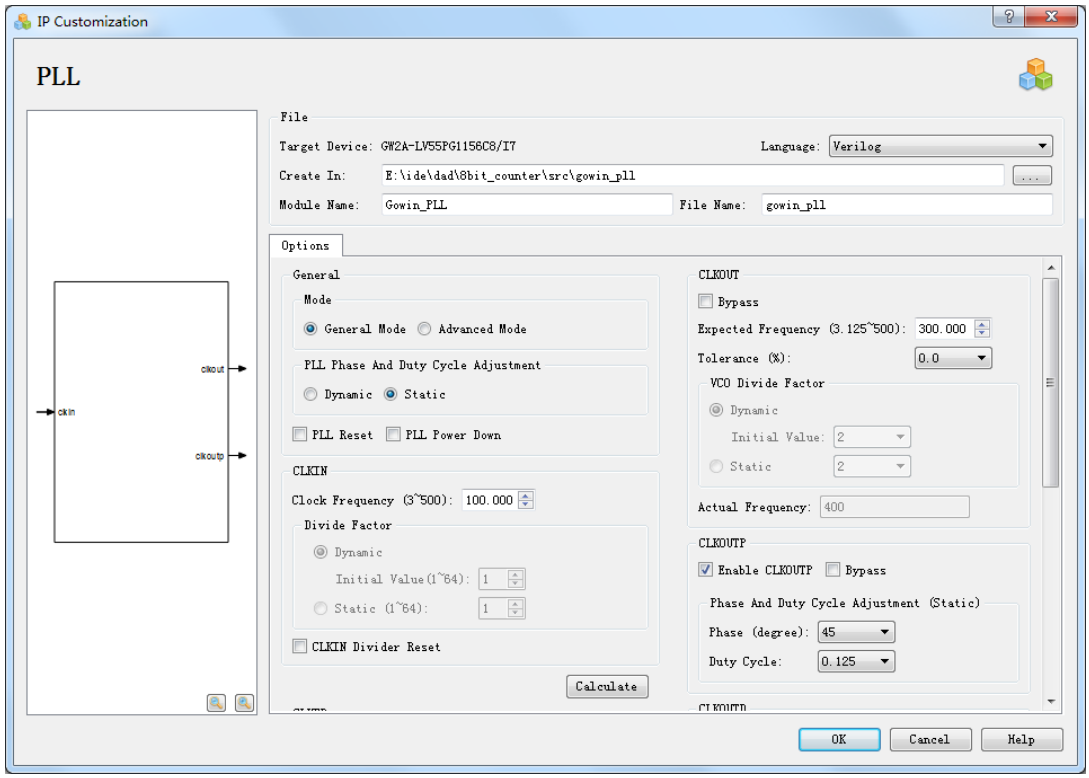
```
Gowin_PLL your_instance_name(
    .clkout(clkout_o), //output clkout
    .clkin(clkin_i) //input clkin
);
```

PLL Generation Example

Generate a specific PLL IP as follows: Input clock: 100MHz; Output clock: 300MHz; Enable CLKOUTP, and the phase adjustment degree is 45°; Enable CLKOUTD, and the expected output frequency is 150MHz. Take the GW2A-LV55PG1156C8/I7 device and general mode for instance, the configuration Iinterface is as shown in Figure 3-121. Click "OK" to generate the customized PLL IP design files.

The directory where the PLL IP design file is generated is the path set in "Create In".

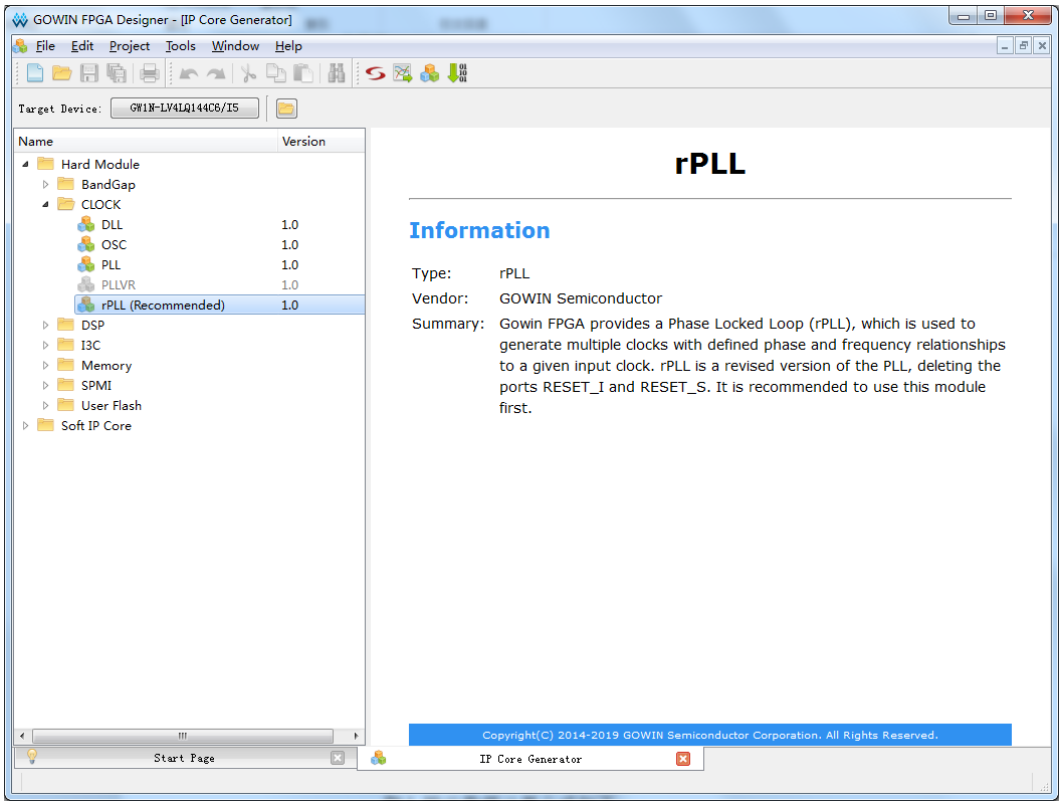
Figure 3-121 PLL IP Customization Configuration



3.3.2 rPLL

rPLL removes RESET_I and RESET_S ports based on PLL, and it is recommended to use rPLL first. Click "rPLL" on the IP Core Generator interface. A brief introduction to the rPLL will be displayed on the right of the screen, as shown in Figure 3-122.

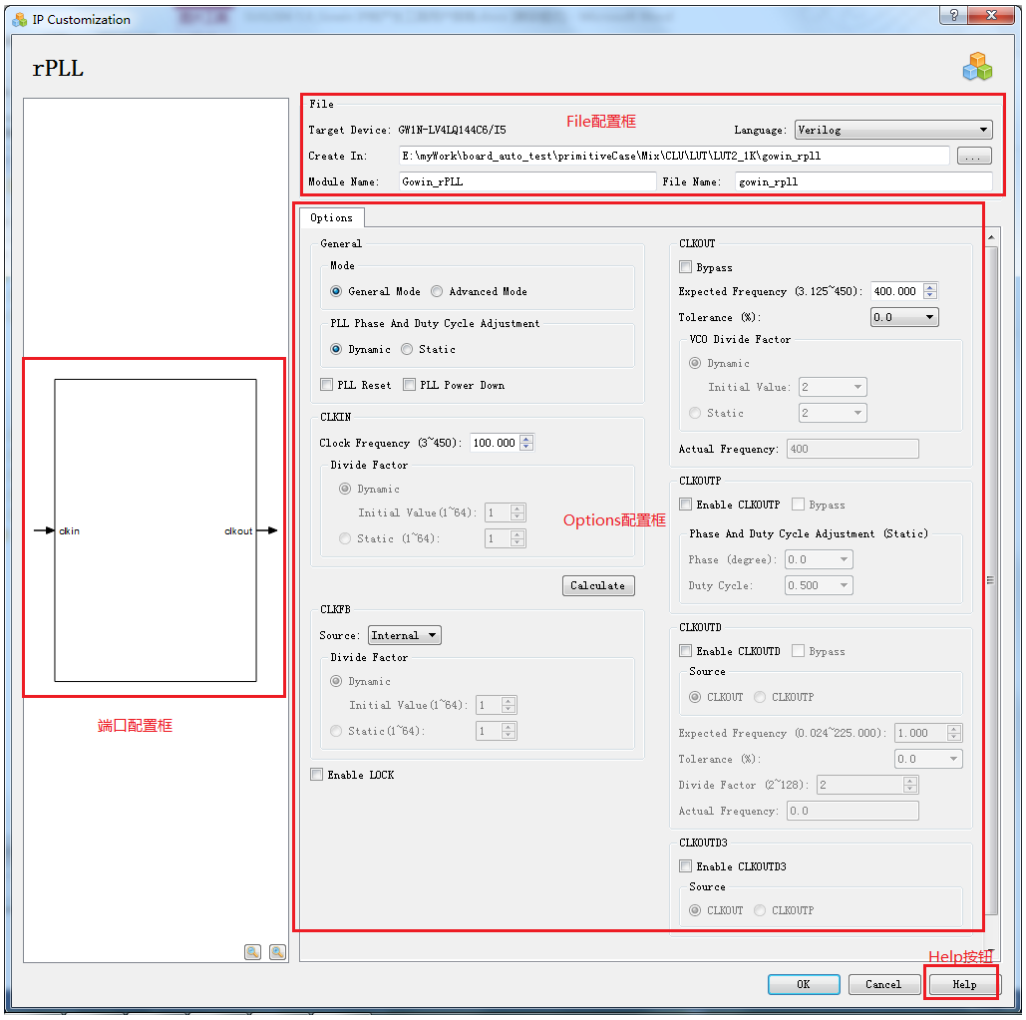
Figure 3-122 rPLL Summary Information



rPLL output data calculation formula is consistent with PLL, please refer to 3.3 CLOC > 3.3.1PLL.

Double click the "rPLL" to open the "IP Customization" window. This includes the "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button, as shown in Figure 3-123.

Figure 3-123 IP Customization of rPLL



1. File Configuration

The file configuration includes the basic information related to the rPLL instantiation file, as shown in Figure 3-123.

The rPLL file configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP > File Configuration.

2. Options Configuration

Options configuration includes configuration information related to the rPLL instantiation file, as shown in Figure 3-123.

The rPLL configuration is similar to that of PLL module. Please refer to the Options configuration box in 3.3 CLOCK > 3.3.1PLL. Where CLKIN removes "CLKIN Divider Reset" option, and CLKOUTD3 removes "CLKOUTD/CLKOUTD3 Divider Reset" option.

3. Ports Configuration

The ports configuration diagram displays the current IP Core configuration result. The number of Input/Output ports updates in real time based on the options configuration, as shown in Figure 3-123.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-124. Help page contains the IP Core general description, and brief

introduction of "Options".

Figure 3-124 Help

rPLL

Information

Type:	rPLL
Vendor:	GOWIN Semiconductor
Summary:	Gowin FPGA provides a Phase Locked Loop (rPLL), which is used to generate multiple clocks with defined phase and frequency relationships to a given input clock. rPLL is a revised version of the PLL, deleting the ports RESET_I and RESET_S. It is recommended to use this module first.

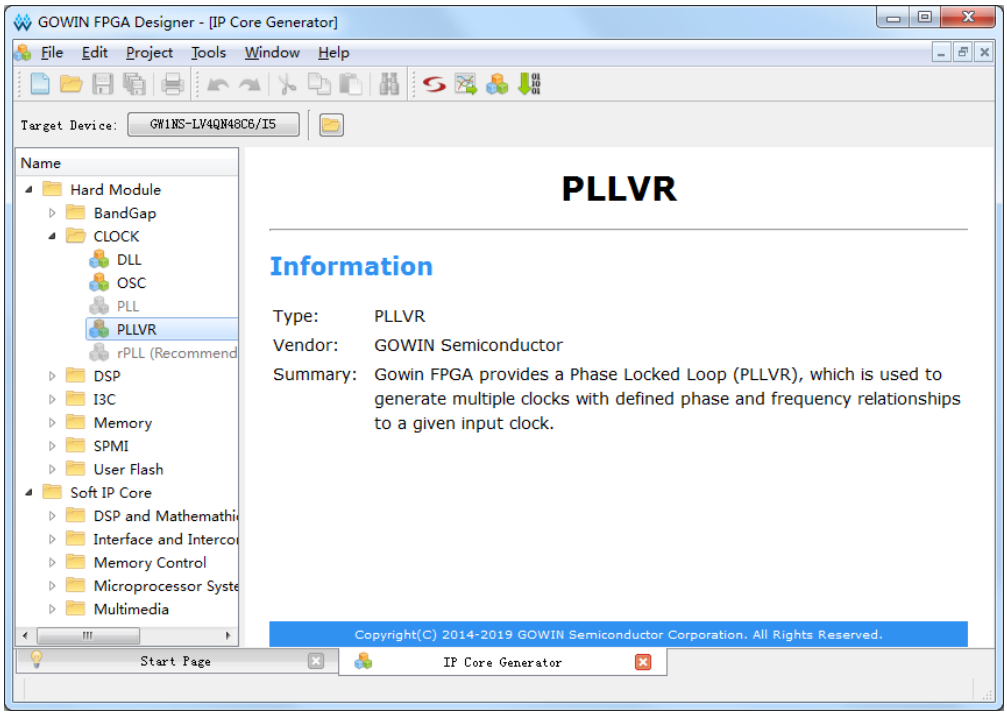
Options

Option	Description
General	**General Mode** - In this mode, entering input clock frequency and expected frequencies, software will automatically calculate divide factors.
	Advanced Mode - This mode is for advanced users. Allows you to enter input clock frequency and divide factors to achieve expected output frequency.
	PLL Phase And Duty Cycle Adjustment - Allows you to select Static Mode or Dynamic Mode.
	PLL Reset - Provides a reset pin to reset the PLL.
	PLL Power Down - Provides a reset_p port to power down the PLL.
CLKIN	CLKIN is the input reference clock for the PLL.
	Clock Frequency - Specify its frequency in MHz.
	Divide Factor - If in Advanced mode, also choose a divide factor which is from Dynamic or Static mode to achieve the expected output frequency. Static mode means select a static value from the drop-down list as divide factor, while Dynamic mode means that choose the value of port idsel as dynamic divide factor. When the Dynamic mode is selected, the user needs to set an initial value.
CLKFB	**Source** - Specify the source of feedback.
	Divide Factor - In Advanced mode, the divide factor in the feedback path can be selected from port fbdsel or from the drop-down list. In General mode, the divide factor is shown when the "Calculate" button is clicked. When the Dynamic mode is selected, the user needs to set an initial value.
LOCK	**Enable LOCK** - Selecting this option will produce the lock port in the generated module.
CLKOUT	**Bypass** - The bypass option means clkout = clkin, it connects the output to the input, bypassing the PLL circuit. Bypassing CLKOUT disables the CLKOUT expected frequency and tolerance fields. If both CLKOUT and CLKOUTP are in Bypass, then everything is disabled except the CLKIN frequency option.
	The following options are not available when CLKOUT is in Bypass mode.
	VCO Divide Factor - In General mode, VCO Divide Factors cannot be selected. Clicking the "Calculate" button displays the actual values. In Advanced mode, Dynamic or Static mode can be selected. When the Dynamic mode is selected, the user needs to set an initial value.
	Expected Frequency - In General mode, set the output clock frequency.
	Tolerance - Set a tolerance for the clkout frequency, as a percentage of requested frequency. Since the divide factors can only take a certain set of values, not all frequencies can be generated. The tolerance value is used to guide the tool to select a suitable divide factor.
	Actual Frequency - Clicking the "Calculate" button displays the actual frequency that the PLL can produce.

3.3.3 PLLVR

PLLVR removes RESET_I and RESET_S ports based on PLL, and VREN port is added. Click "PLLVR" on the IP Core Generator interface. A brief introduction to the PLLVR will be displayed on the right of the screen, as shown in Figure 3-125.

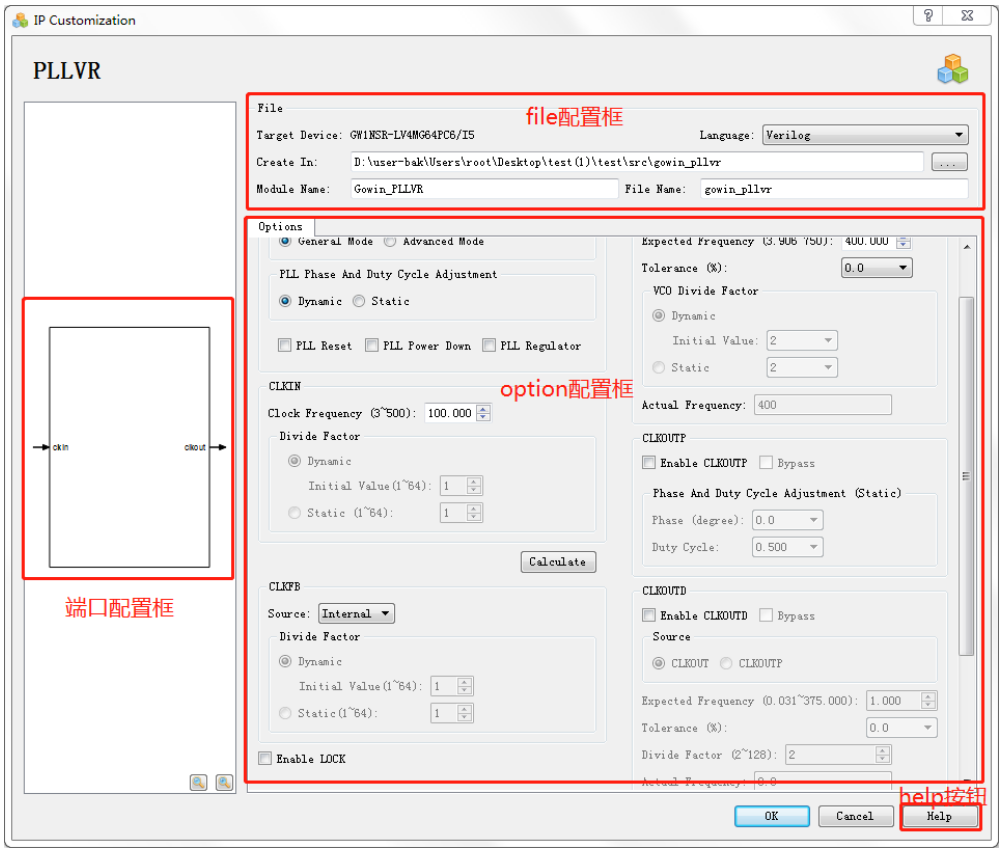
Figure 3-125 PLLVR Summary Information



PLLVR output data calculation formula is consistent with PLL, please refer to 3.3 CLOCK > 3.3.1PLL.

Double click the "PLLVR" to open the "IP Customization" window. This includes the "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button, as shown in Figure 3-126.

Figure 3-126 IP Customization of PLLVR



1. File Configuration

The file configuration includes the basic information related to the PLLVR instantiation file, as shown in Figure 3-126.

The PLLVR file configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP > File Configuration.

2. Options Configuration

Options configuration includes configuration information related to the PLLVR instantiation file, as shown in Figure 3-126.

The PLLVR configuration is similar to that of PLL module. Please refer to the Options configuration box in 3.3 CLOCK > 3.3.1PLL. Where CLKIN removes "CLKIN Divider Reset" option, and CLKOUTD3 removes "CLKOUTD/CLKOUTD3 Divider Reset" option, and "PLL Regulater" option is added.

3. Ports Configuration

The ports configuration diagram displays the current IP Core configuration result. The number of Input/Output ports updates in real time based on the options configuration, as shown in Figure 3-126.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-127. Help page contains the IP Core general description, and brief introduction of "Options".

Figure 3-127 Help

PLLVR

Information

Type:	PLLVR
Vendor:	GOWIN Semiconductor
Summary:	Gowin FPGA provides a Phase Locked Loop (PLLVR), which is used to generate multiple clocks with defined phase and frequency relationships to a given input clock.

Options

Option	Description
General	**General Mode** - In this mode, entering input clock frequency and expected frequencies, software will automatically calculate divide factors.
	Advanced Mode - This mode is for advanced users. Allows you to enter input clock frequency and divide factors to achieve expected output frequency.
	PLL Phase And Duty Cycle Adjustment - Allows you to select Static Mode or Dynamic Mode.
	PLL Reset - Provides a reset pin to reset the PLL.
	PLL Power Down - Provides a reset_p port to power down the PLL.
	PLL Regulator - Provides a vren port to PLL dynamic power regulation.
CLKIN	CLKIN is the input reference clock for the PLL.
	Clock Frequency - Specify its frequency in MHz.
	Divide Factor - If in Advanced mode, also choose a divide factor which is from Dynamic or Static mode to achieve the expected output frequency. Static mode means select a static value from the drop-down list as divide factor, while Dynamic mode means that choose the value of port idsel as dynamic divide factor. When the Dynamic mode is selected, the user needs to set an initial value.
CLKFB	**Source** - Specify the source of feedback.
	Divide Factor - In Advanced mode, the divide factor in the feedback path can be selected from port fbdsel or from the drop-down list. In General mode, the divide factor is shown when the "Calculate" button is clicked. When the Dynamic mode is selected, the user needs to set an initial value.
LOCK	**Enable LOCK** - Selecting this option will produce the lock port in the generated module.

CLKOUT	**Bypass** - The bypass option means clkout = clkin, it connects the output to the input, bypassing the PLL circuit. Bypassing CLKOUT disables the CLKOUT expected frequency and tolerance fields. If both CLKOUT and CLKOUTP are in Bypass, then everything is disabled except the CLKIN frequency option.
	The following options are not available when CLKOUT is in Bypass mode.
	VCO Divide Factor - In General mode, VCO Divide Factors cannot be selected. Clicking the "Calculate" button displays the actual values. In Advanced mode, Dynamic or Static mode can be selected. When the Dynamic mode is selected, the user needs to set an initial value.
	Expected Frequency - In General mode, set the output clock frequency.
	Tolerance - Set a tolerance for the clkout frequency, as a percentage of requested frequency. Since the divide factors can only take a certain set of values, not all frequencies can be generated. The tolerance value is used to guide the tool to select a suitable divide factor.
	Actual Frequency - Clicking the "Calculate" button displays the actual frequency that the PLL can produce.
CLKOUTP	**Enable CLKOUTP** – Selecting this option will produce the clkoutp port in the generated module. Clkoutp has the same frequency as clkout and it has the specified phase relation with clkout.
	Bypass – This option will connect the output to the input, bypassing the PLL circuit. Bypassing clkoutp disables its frequency, tolerance, and phase shift fields.
	PLL Phase And Duty Cycle Adjustment(Static)
	Phase - You can select phase shift in 22.5-degree increments from 0 to 360.
	Duty Cycle - Allows duty cycle selection in 1/16 increments.
CLKOUTD	**Enable CLKOUTD** - Selecting this option will produces clkoutd port in the generated module. In general mode, select the expected frequency. In advanced mode, select the divide factor for the expected frequency.
	Bypass - The bypass option means clkoutd = clkin, it connects the output to the input, bypassing the PLL circuit.
	Source - If clkoutd is enabled, you can select CLKOUT or CLKOUTP as the source of clkoutd. If CLKOUTP is not enabled, CLKOUT is used as the source.
	Expected Frequency - In General mode, set the CLKOUTD Frequency. This option is disabled in advanced mode.
	Divide Factor - In General mode, the divide factor cannot be selected. Clicking the "Calculate" button the tool will automatically displays the actual values. In Advanced mode, select the CLKOUTD divide factor from the drop-down list.
	Tolerance - Set a tolerance for the CLKOUTD frequency, as a percentage of requested frequency. Since the divide factors can only take a certain set of values, not all frequencies can be generated. The tolerance value is used to guide the tool to select a suitable divide factor.
	Actual Frequency - Clicking the "Calculate" button displays the actual frequency that the PLL can produce.
CLKOUTD3	**Enable CLKOUTD3** - Selecting this option will produces clkoutd3 port in the generated module, and it is equal to clkout/3.
	Source - If CLKOUTD3 is enabled, you can select CLKOUT or CLKOUTP as the source of CLKOUTP. If CLKOUTP is not enabled, CLKOUT is used as the source.
Calculate	General Mode: The tool calculates Divide Factor settings based on input/output frequency. Advanced Mode: The tool calculates output frequencies based on divide factors.

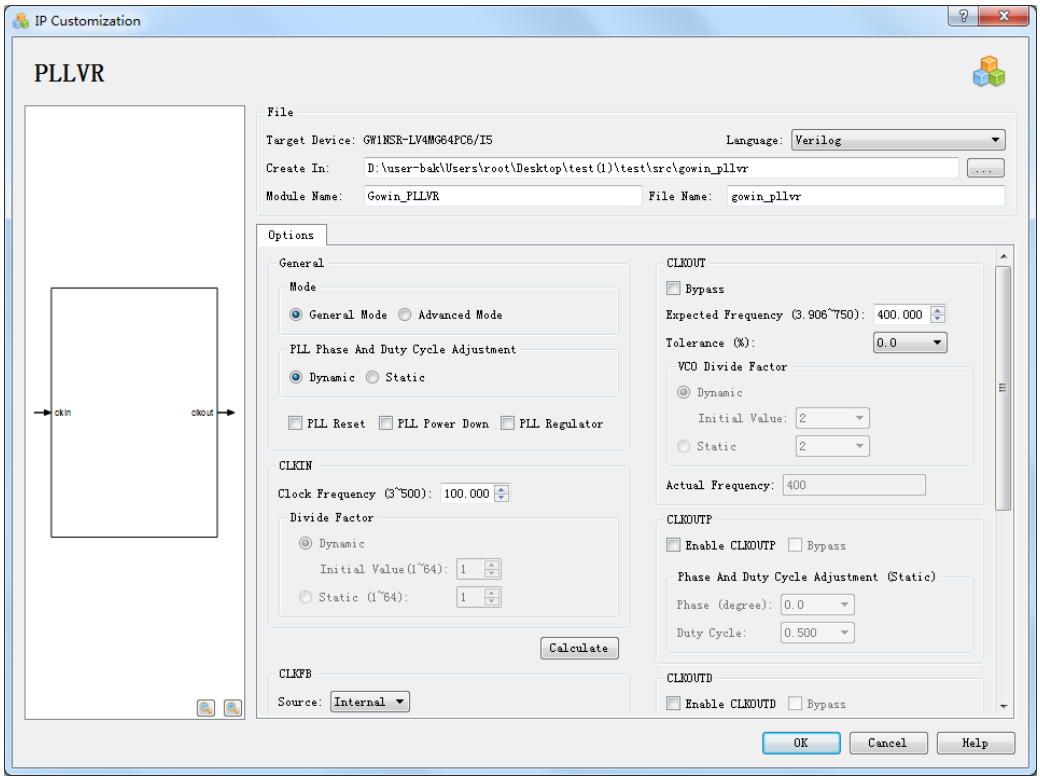
IP Generation Files

As shown in Figure 3-128, after customizing the IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate Gowin Primitive PLLVR design file "gowin_pllvr.v";
- The instantiation template file for the IP design file "gowin_pllvr_tmp.v";
- The configuration files for the Primitive PLLVR instantiation "gowin_pllvr.ipc".

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

Figure 3-128 Configured IP Customization



PLLVR Design File Instantiation

PLLVR design file instantiation is a complete Verilog module. PLL instantiation is generated according to the PLLVR configuration that is displayed in the "IP Customization" window, as shown in Figure 3-129.

Figure 3-129 PLLVR Design File Instantiation

```
module Gowin_PLLVR (clkout, clkin);
output clkout;
input clkin;
wire lock_o;
wire clkoutp_o;
wire clkoutd_o;
wire clkoutd3_o;
wire gw_vcc;
wire gw_gnd;
assign gw_vcc = 1'b1;
assign gw_gnd = 1'b0;
PLLVR pllvr_inst (
    .CLKOUT(clkout),
    .LOCK(lock_o),
    .CLKOUTP(clkoutp_o),
    .CLKOUTD(clkoutd_o),
    .CLKOUTD3(clkoutd3_o),
    .RESET(gw_gnd),
    .RESET_P(gw_gnd),
    .CLKIN(clkin),
    .CLKFB(gw_gnd),
    .FBDSEL({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd}),
    .IDSEL({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd}),
    .ODSEL({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd}),
    .PSDA({gw_gnd,gw_gnd,gw_gnd,gw_gnd}),
    .DUTYDA({gw_gnd,gw_gnd,gw_gnd,gw_gnd}),
    .FDLY({gw_gnd,gw_gnd,gw_gnd,gw_gnd}),
    .VREN(gw_vcc)
);
defparam pllvr_inst.FCLKIN = "100";
defparam pllvr_inst.DYN_IDIV_SEL = "false";
defparam pllvr_inst.IDIV_SEL = 0;
defparam pllvr_inst.DYN_FBDIV_SEL = "false";
defparam pllvr_inst.FBDIV_SEL = 3;
defparam pllvr_inst.DYN_ODIV_SEL = "false";
defparam pllvr_inst.ODIV_SEL = 2;
defparam pllvr_inst.PSDA_SEL = "0000";
defparam pllvr_inst.DYN_DA_EN = "true";
defparam pllvr_inst.DUTYDA_SEL = "1000";
defparam pllvr_inst.CLKOUT_FT_DIR = 1'b1;
defparam pllvr_inst.CLKOUTP_FT_DIR = 1'b1;
defparam pllvr_inst.CLKOUT_DLY_STEP = 0;
defparam pllvr_inst.CLKOUTP_DLY_STEP = 0;
defparam pllvr_inst.CLKFB_SEL = "internal";
defparam pllvr_inst.CLKOUT_BYPASS = "false";
defparam pllvr_inst.CLKOUTP_BYPASS = "false";
defparam pllvr_inst.CLKOUTD_BYPASS = "false";
defparam pllvr_inst.DYN_SDIV_SEL = 2;
defparam pllvr_inst.CLKOUTD_SRC = "CLKOUT";
defparam pllvr_inst.CLKOUTD3_SRC = "CLKOUT";
defparam pllvr_inst.DEVICE = "GW1NSR-4";
endmodule //Gowin_PLLVR
```

Instantiation Template File for the IP Design File

For the practical use, the IP Core Generator generates the template file while generating the PLLVR design file instantiation, as shown in Figure 3-130.

Figure 3-130 Instantiation Template File for the IP Design File

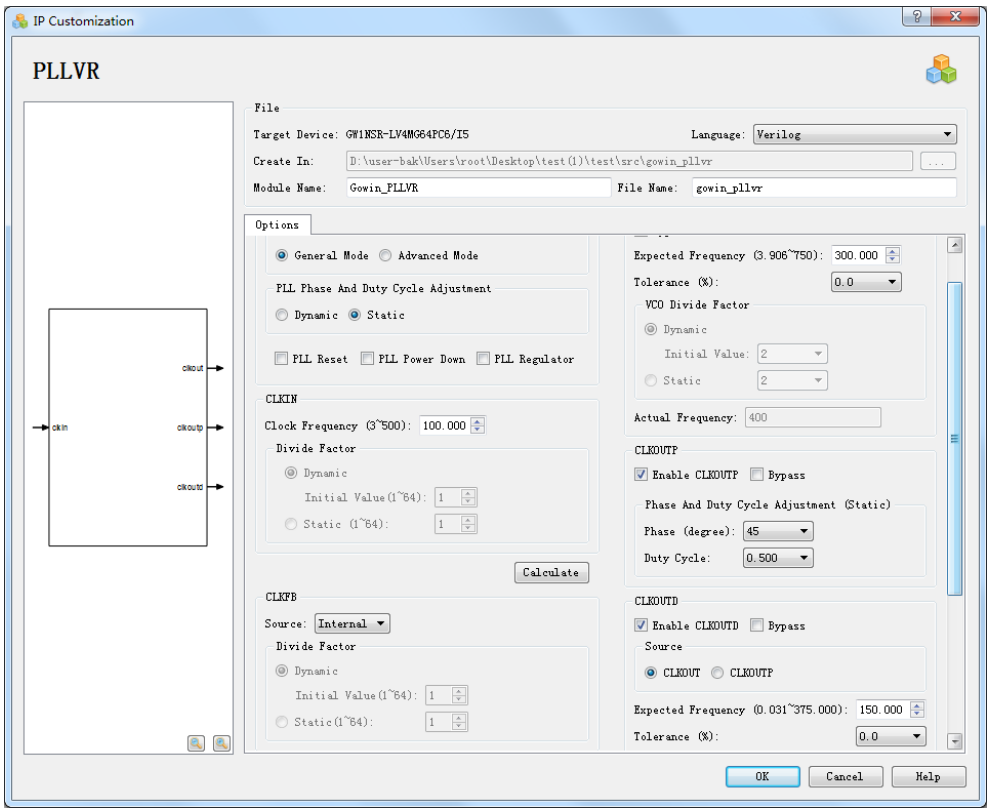
```
//--------Copy here to design--------

    Gowin_PLLVR your_instance_name(
        .clkout(clkout_o), //output clkout
        .clkin(clkin_i) //input clkin
    );

//--------Copy end-------------------
```

PLLVR Generation Example

Generate a specific PLLVR IP as follows: Input clock: 100MHz; Output clock: 300MHz; Enable CLKOUTP, and the phase adjustment degree is 45°; Enable CLKOUTD, and the expected output frequency is 150MHz. Take the GW1NSR-LV4MG64PC6/I5 device and general mode for instance, the configuration linterface is as shown in Figure 3-131. Click "OK" to generate the customized PLLVR IP design files.

The directory where the PLLVR IP design file is generated is the path set in "Create In".

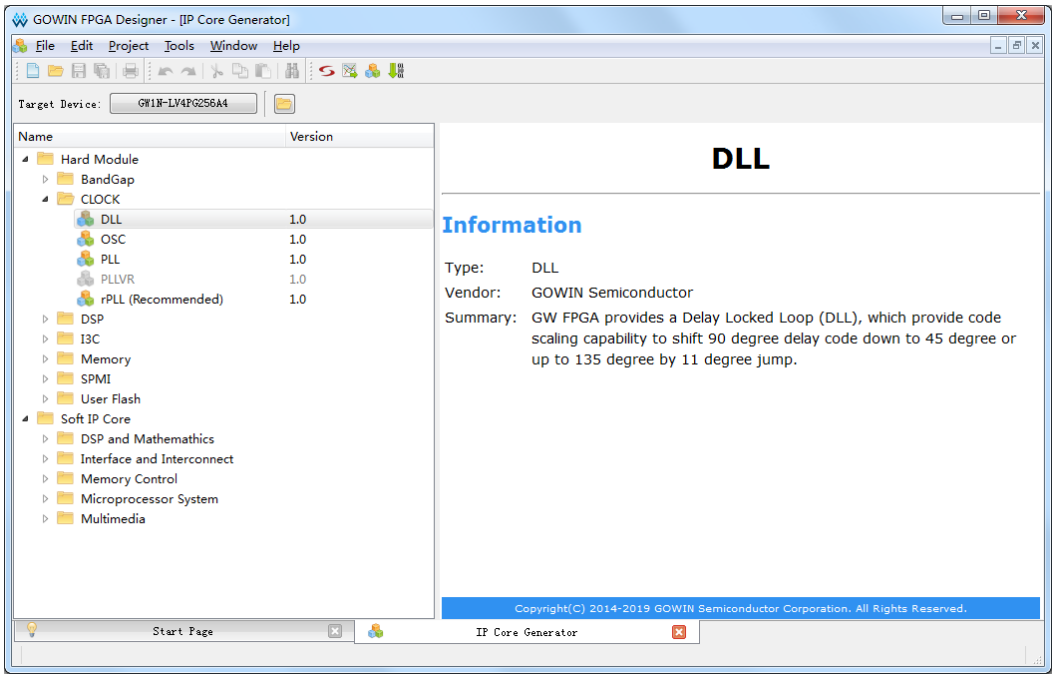
Figure 3-131 PLLVR IP Customization Configuration



3.3.4 DLL

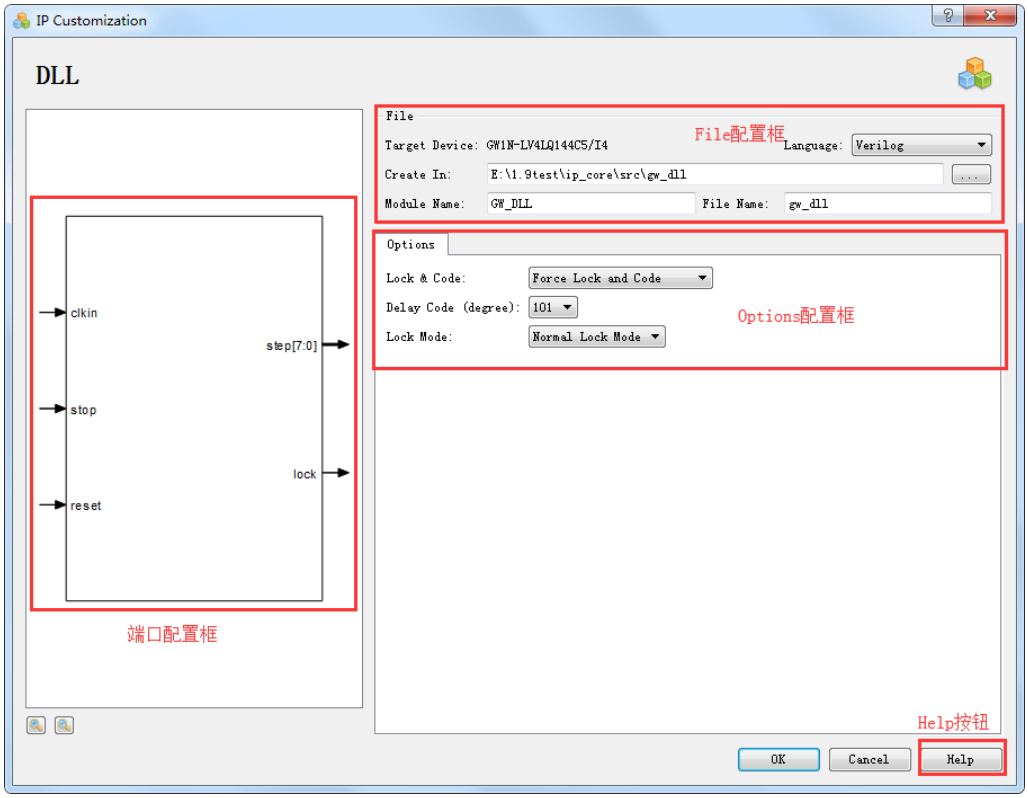
Delay Lock Loop (DLL) is used to ensure accurate time delay by the equal and precise division of the input signal cycle. Click "DLL" on the IP Core Generator interface. A brief introduction to the DLL will be displayed on the right of the screen, as shown in Figure 3-132.

Figure 3-132 DLL Summary



Double click the "DLL" to open the "IP Customization" window. This displays the "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button, as shown in Figure 3-133.

Figure 3-133 IP Customization of DLL



1. File Configuration

The file configuration includes the basic information related to the DLL instantiation file, as shown in Figure 3-133.

The DLL file configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP> File Configuration.

2. Options Configuration

The options configuration mainly includes the configuration information related to the DLL instantiation file, as shown in Figure 3-133.

- Lock and Code: Select LOCK&STEP output mode.
  - Force Lock and Code: In this mode, the LOCK value is 1, and the STEP value is forcibly set to 255.
  - Generated From DLL Loop: This mode ensures that the LOCK and STEP values are all generated by the DLL.
- Delay Code (degree): Specify a delay code (degree) for the DLL. The options are 101°, 112°, 123° , 135°, 79°, 68°, 57°, 45°, and 90°.
  - Lock Mode: Allows users to select Normal Lock Mode or Fast Lock Mode.
  - Normal Lock Mode: The DLL parameter DIV_SEL is set to 1'b0;
  - Fast Lock Mode: The DLL parameter DIV_SEL is set to 1'b1.

3. Ports Configuration Diagram

The ports configuration diagram displays the current IP Core configuration result, as shown in Figure 3-133.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-134.

Figure 3-134 Help

DLL

Information

Type:	DLL
Vendor:	GOWIN Semiconductor
Summary:	GW FPGA provides a Delay Locked Loop (DLL), which provide code scaling capability to shift 90 degree delay code down to 45 degree or up to 135 degree by 11 degree jump.

Options

Option	Description
Lock & Code	**Force Lock and Code** - In this mode, the STEP value is forcibly set to 255.
	Generated From DLL Loop - This mode ensures STEP value is generated by the DLL Loop.
Delay Code(degree)	Specify a delay code (degree) for DLL.
Lock Mode	Allows you to select **Normal Lock Mode** or **Fast Lock Mode**.

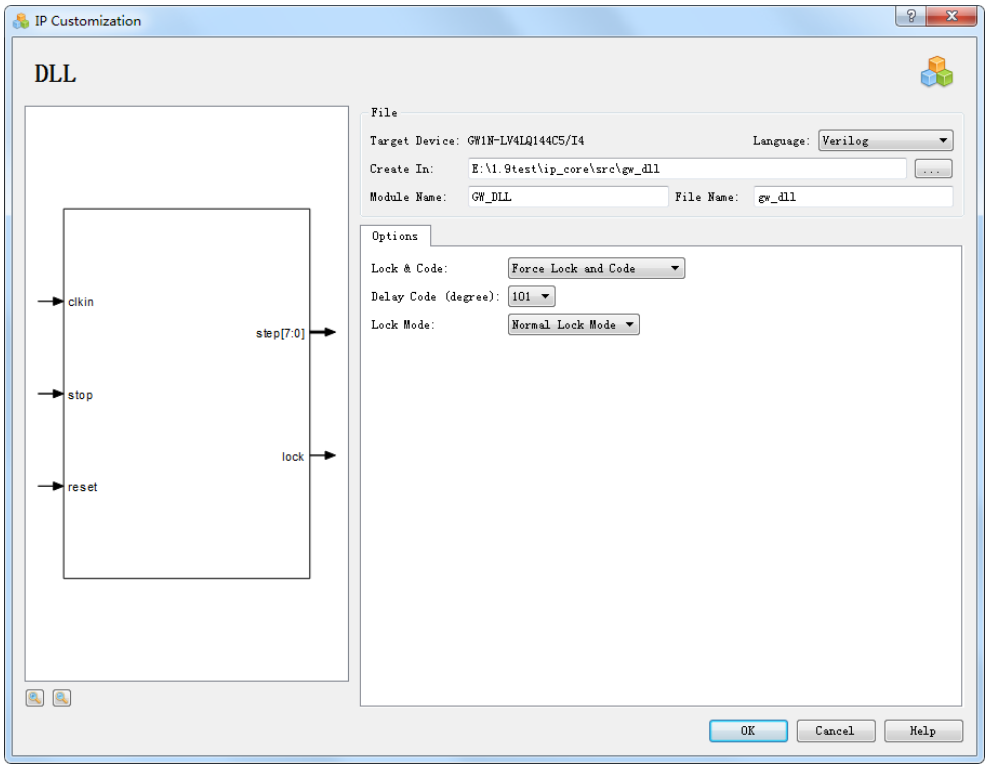
IP Generation Files

As shown in Figure 3-135, after customizing the IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate Gowin Primitive DLL instantiation "gw_dll.v";
- The instantiation template file for the IP design file "gw_dll_tmp.v";
- The configuration file for the Gowin Primitive DLL instantiation "gw_dll.ipc".

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.

Figure 3-135 Configured IP Customization



DLL Design File Instantiation

DLL design file instantiation is a complete Verilog module. DLL instantiation is generated according to the DLL configuration that is displayed in the "IP Customization" window, as shown in Figure 3-136.

Figure 3-136 DLL Design File Instantiation

```
module GW_DLL (step, lock, clkin, stop, reset);

output [7:0] step;
output lock;
input clkin;
input stop;
input reset;

wire gw_gnd;

assign gw_gnd = 1'b0;

DLL dll_inst (
    .STEP(step),
    .LOCK(lock),
    .CLKIN(clkin),
    .STOP(stop),
    .RESET(reset),
    .UPDNCNTL(gw_gnd)
);

defparam dll_inst.DLL_FORCE = 1;
defparam dll_inst.CODESCAL = "000";
defparam dll_inst.SCAL_EN = "true";
defparam dll_inst.DIV_SEL = 1'b0;

endmodule //GW_DLL
```

Instantiation Template File for the IP Design File

For the practical use, the IP Core Generator generates the template file while generating DLL design file instantiation, as shown in Figure 3-137.

Figure 3-137 Instantiation Template File for the IP Design File

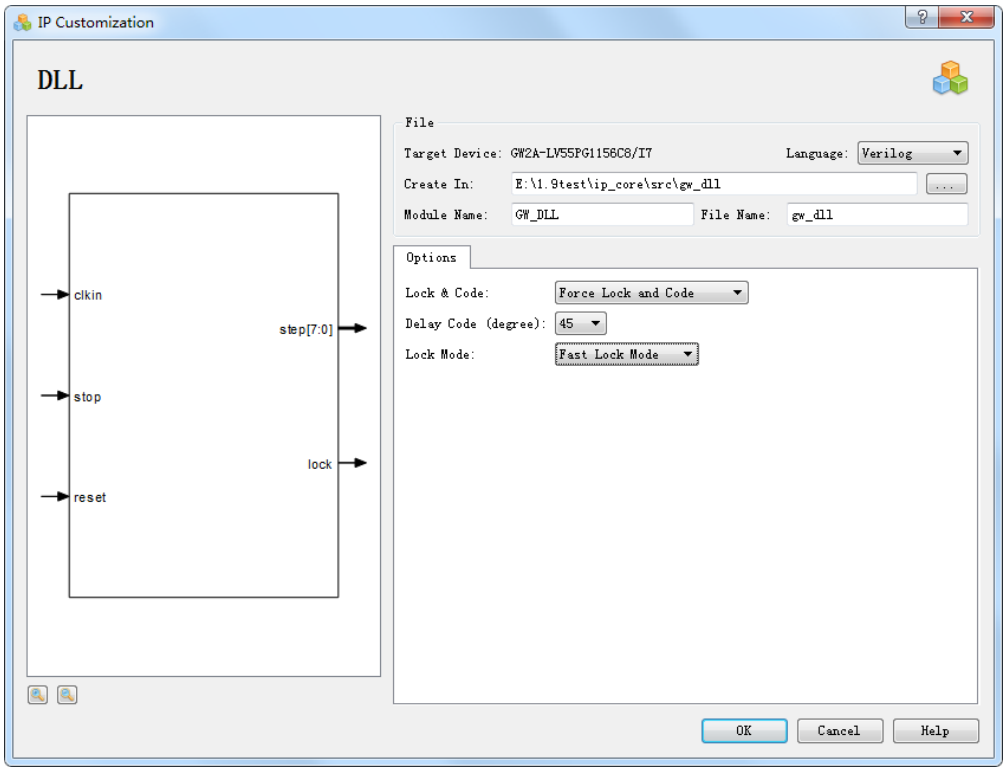
```
GW_DLL your_instance_name(
    .step(step_o), //output [7:0] step
    .lock(lock_o), //output lock
    .clkin(clkin_i), //input clkin
    .stop(stop_i), //input stop
    .reset(reset_i) //input reset
);
```

DLL Generation Example

Generate a specific DLL IP as follows: Delay Code is 45°; Fast lock mode. Take the GW2A-LV55PG1156C8/I7 device for instance, the configuration interface is as shown in Figure 3-138. Click "OK" to generate the customized DLL IP design files.

The directory where the DLL IP design file is generated is the path set in "Create In".

Figure 3-138 DLL IP Customization Configuration

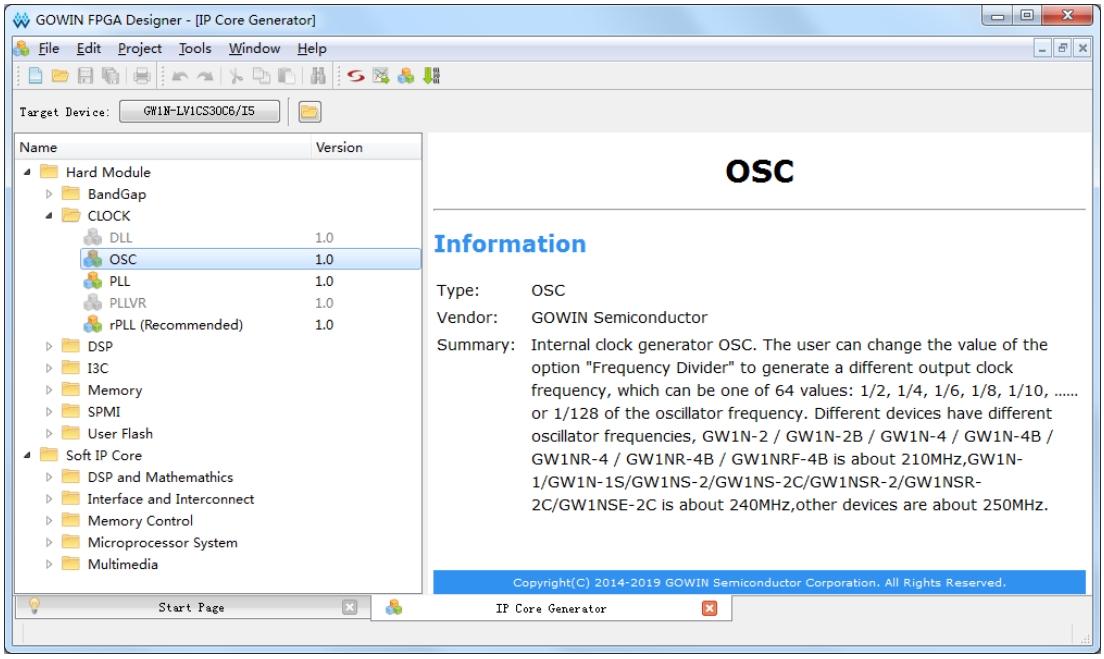


3.3.5 OSC

OSC is the internal crystal oscillator. It has a maximum frequency of 125MHz. Click "OSC" on the IP Core Generator page. A brief introduction to the OSC will be displayed on the right of the screen, as shown in Figure

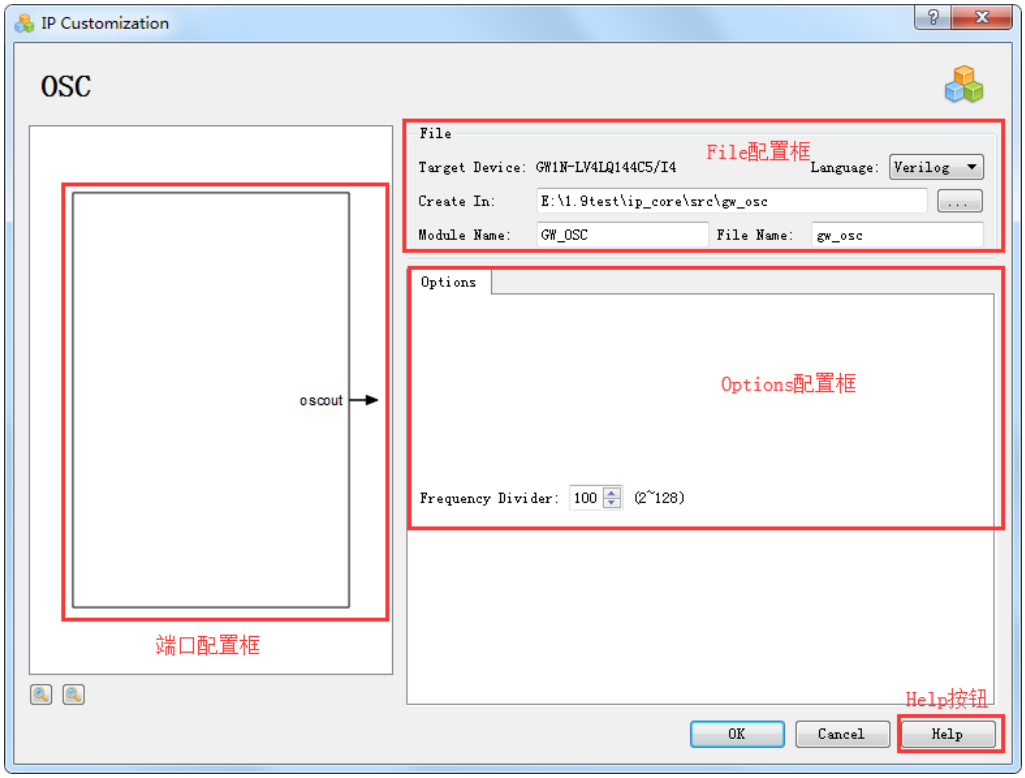
3-139.

Figure 3-139 OSC Summary Information



Double click "OSC", and the "IP Customization" window pops up, as shown in Figure 3-140. This displays the "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button.

Figure 3-140 IP Customization of OSC



1. File Configuration

The file configuration includes the basic information related to the OSC instantiation file, as shown in Figure 3-140.

The OSC file configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP> File Configuration.

2. Options Configuration

The options configuration includes the configuration information related to the OSC instantiation file, as shown in Figure 3-140.

Frequency Divider: Allows users to select any even number between 2 and 128.

3. Ports Configuration Diagram

Ports configuration diagram displays the current IP Core configuration result, as shown in Figure 3-140.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-141. Help page contains the IP Core general description, and a brief introduction to the "Options".

Figure 3-141 Help

OSC

Information

Type:	OSC
Vendor:	GOWIN Semiconductor
Summary:	Internal clock generator OSC. The user can change the value of the option "Frequency Divider" to generate a different output clock frequency, which can be one of 64 values: 1/2, 1/4, 1/6, 1/8, 1/10, ... or 1/128 of the oscillator frequency. Different devices have different oscillator frequencies, GW1N-2 / GW1N-2B / GW1N-4 / GW1N-4B / GW1NR-4 / GW1NR-4B / GW1NRF-4B is about 210MHz,GW1N-1/GW1N-1S/GW1NS-2/GW1NS-2C/GW1NSR-2/GW1NSR-2C/GW1NSE-2C is about 240MHz, other devices are about 250MHz.

Options

Option	Description
Frequency Divider	Allows you to select any even number between 2 ~ 128.

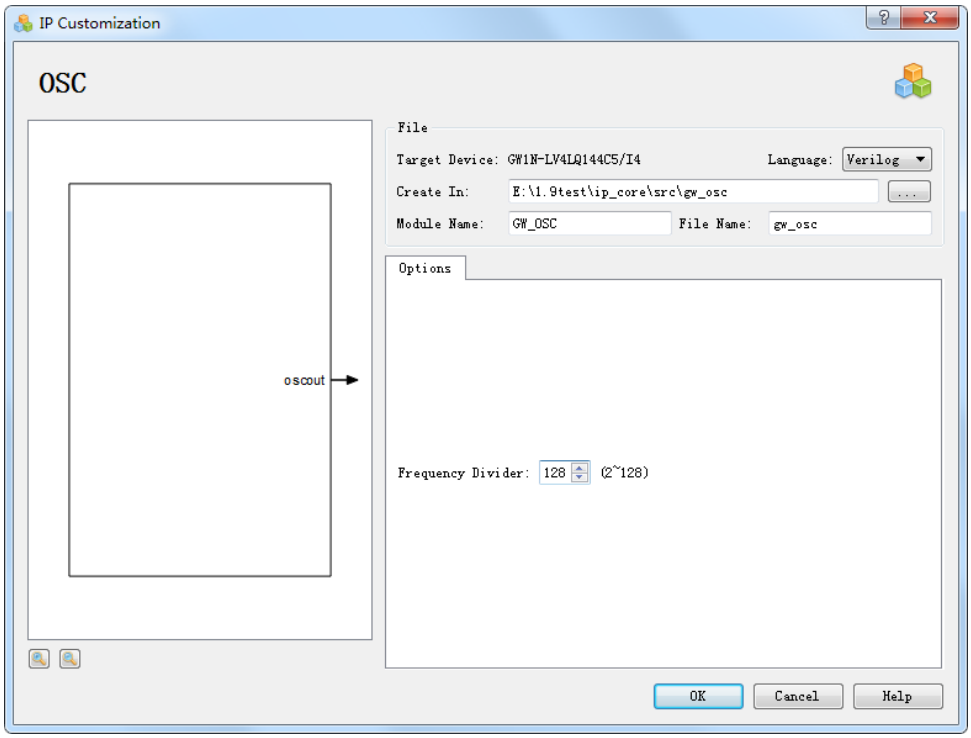
IP Generation Files

As shown in Figure 3-142, after customizing the IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate Gowin Primitive OSC instantiation "gw_osc.v";
- The instantiation template file for the IP design file "gw_osc_tmp.v";
- The configuration file fo r the Gowin Primitive OSC instantiation "gw_osc.ipc".

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

Figure 3-142 Configured IP Customization



OSC Design File Instantiation

OSC design file instantiation is a complete Verilog module. OSC instantiation is generated according to the OSC configuration that is displayed in the "IP Customization" window, as shown in Figure 3-143.

Figure 3-143 OSC Design File Instantiation

```
module GW_OSC (oscout);

output oscout;

OSC osc_inst (
    .OSCOUT(oscout)
);

defparam osc_inst.FREQ_DIV = 128;
defparam osc_inst.DEVICE = "GW2A-55";

endmodule //GW_OSC
```

Instantiation Template File for the IP Design File

For practical use, the IP Core Generator generates the template file while generating OSC design file instantiation, as shown in Figure 3-144.

Figure 3-144 Instantiation Template File for the IP Design File

```
GW_OSC your_instance_name(
    .oscout(oscout_o) //output oscout
);
```

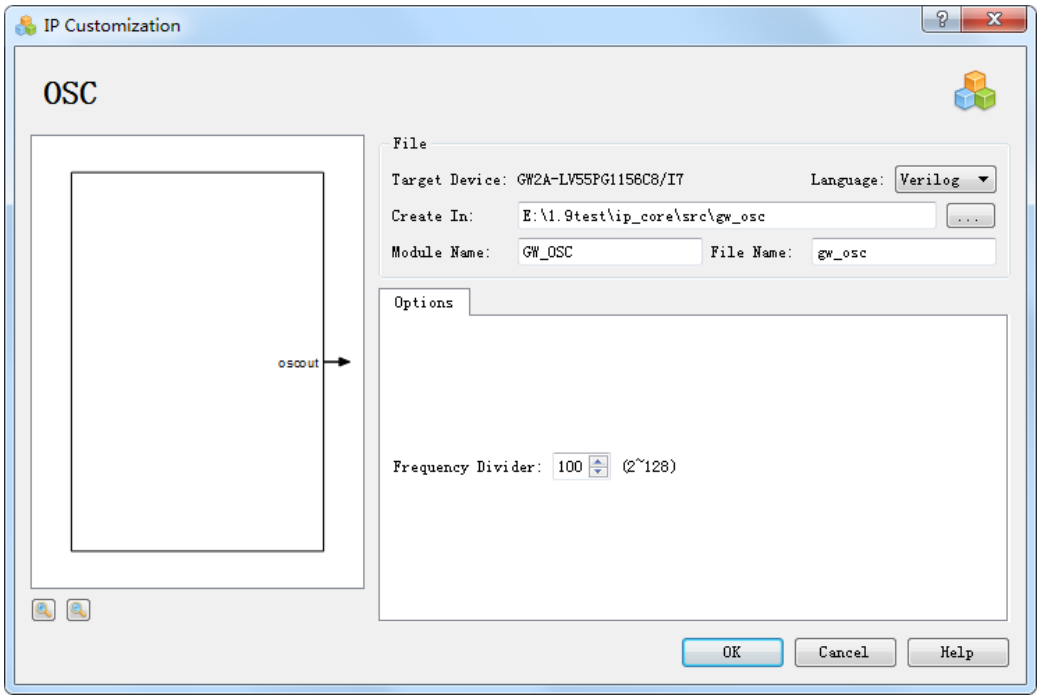
OSC Generation Example

Users can generate a specific OSC IP with 2.5MHz clock frequency.

Take the GW2A-LV55PG1156C8/I7 device for instance, the configuration interface is as shown in Figure 3-145. Click "OK" to generate the customized OSC IP design files.

The directory where the OSC IP design file is generated is the path set in "Create In".

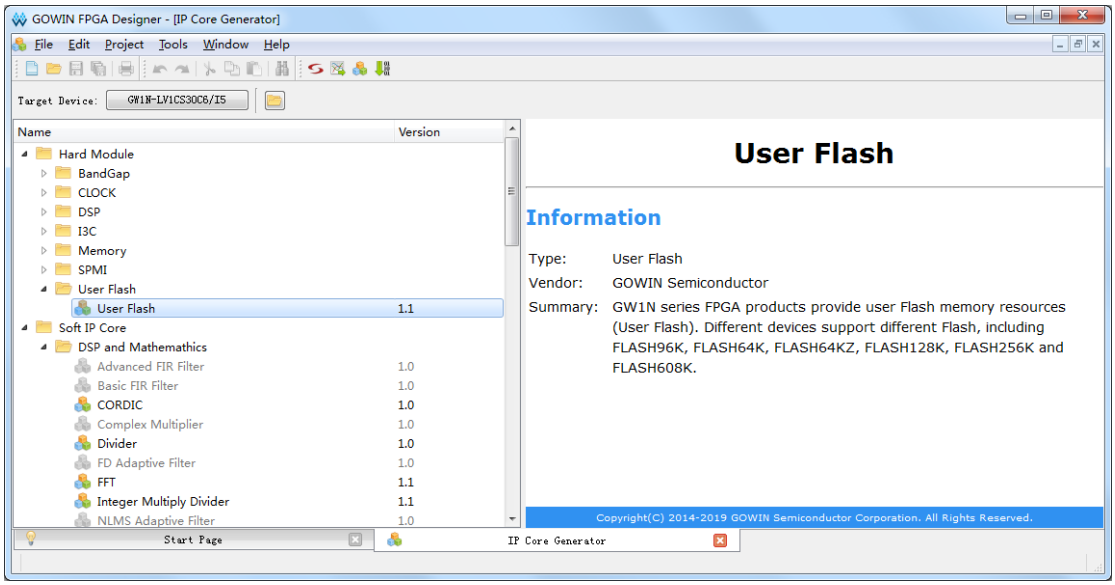
Figure 3-145 OSC IP Customization Setting



3.4 User Flash

User Flash is the user's flash memory. Click "User Flash" on the IP Core Generator page. A brief introduction to the User Flash will be displayed on the right of the screen, as shown in Figure 3-146.

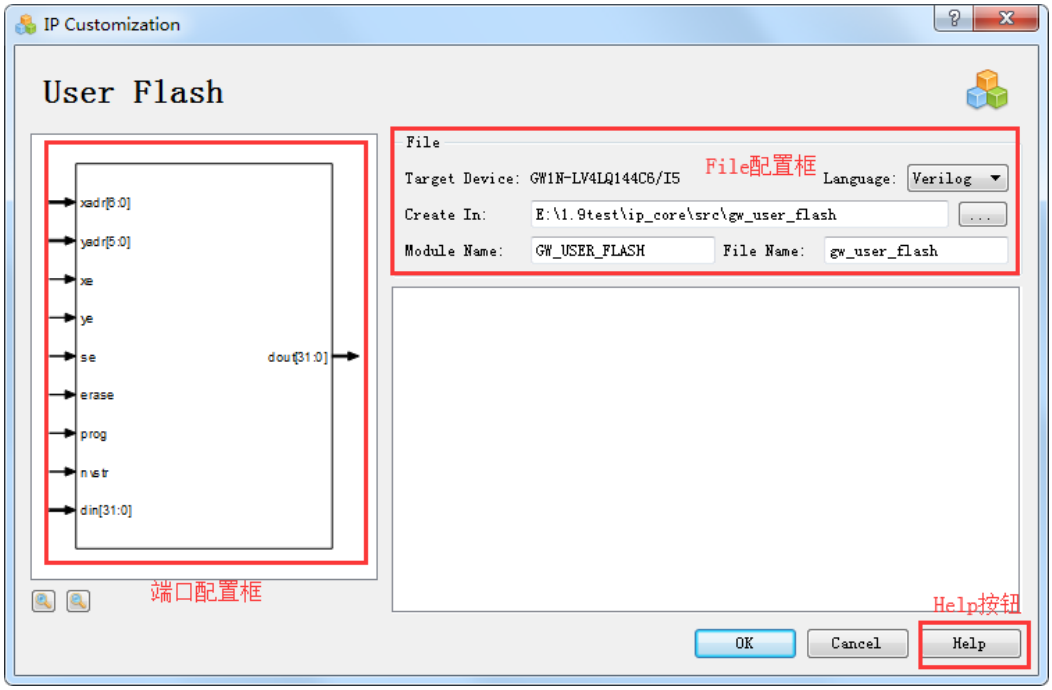
Figure 3-146 User Flash Summary Information



Double-clicking on "User Flash", and the "IP Customization" window

opens as shown in Figure 3-147. This displays the "File" configuration and port configuration diagram.

Figure 3-147 IP Customization of User Flash



1. File Configuration

The file configuration includes the basic information related to the User Flash instantiation file, as shown in Figure 3-147.

The User Flash file configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP> File Configuration.

Note!

- At present, devices that support FLASH96K include: GW1N-1/GW1N-1S
- Devices that support FLASH64KZ include: GW1NZ-1
- Devices that support FLASH64KZ include: GW1NZ-1
- Devices that support FLASH128K include: GW1NS-2/ GW1NS-2C /GW1NSR-2/GW1NSR-2C/GW1NSE-2C
- Devices that support FLASH256K include: GW1N-2/ GW1N-2B/GW1N-4/ GW1N-4B/ GW1NR-4/ GW1NR-4B/GW1NS-4/GW1NS-4C/GW1NSR-4/GW1NSR-4C /GW1NSER-4C
- Devices that support FLASH608K include: GW1N-6/ GW1N-9/ GW1NR-9
- If the Target Device selects any Device other than the above, the User Flash is grayed and cannot generate the corresponding IP.

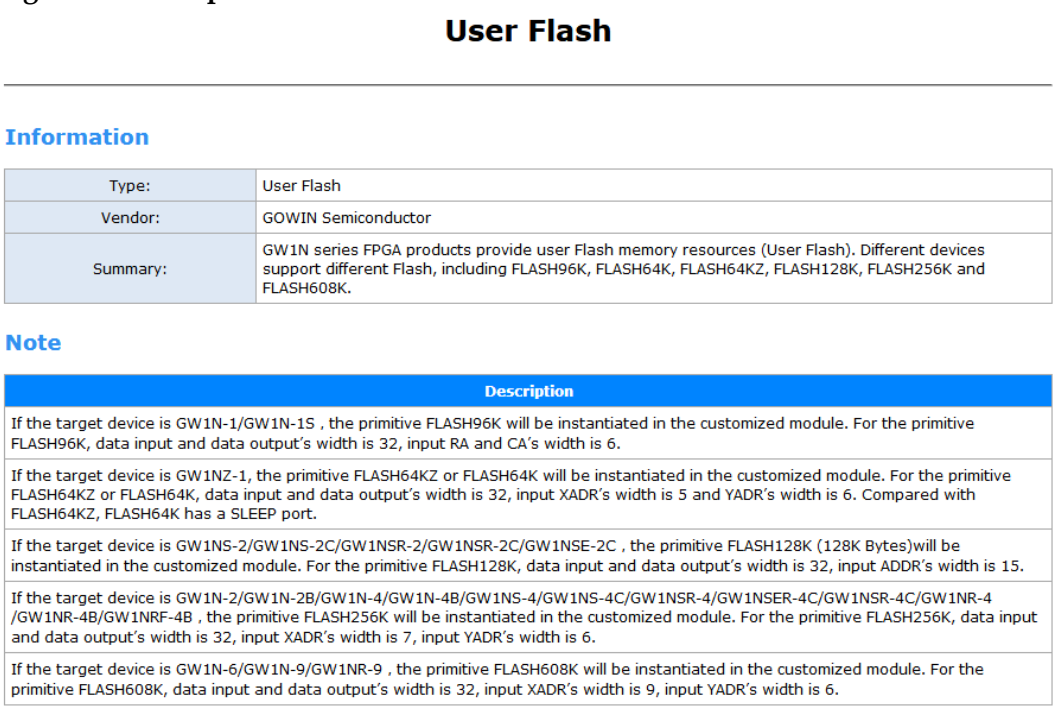
2. Ports Configuration Diagram

The ports configuration diagram displays the current IP Core configuration result, and User Flash input bit-width updates in real time based on the target device, as shown in Figure 3-147.

3. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-148. Help page contains the IP Core general description, and a brief introduction to the "Options".

Figure 3-148 Help



User Flash

Information

Type:	User Flash
Vendor:	GOWIN Semiconductor
Summary:	GW1N series FPGA products provide user Flash memory resources (User Flash). Different devices support different Flash, including FLASH96K, FLASH64K, FLASH64KZ, FLASH128K, FLASH256K and FLASH608K.

Note

Description
If the target device is GW1N-1/GW1N-1S , the primitive FLASH96K will be instantiated in the customized module. For the primitive FLASH96K, data input and data output's width is 32, input RA and CA's width is 6.
If the target device is GW1NZ-1, the primitive FLASH64KZ or FLASH64K will be instantiated in the customized module. For the primitive FLASH64KZ or FLASH64K, data input and data output's width is 32, input XADR's width is 5 and YADR's width is 6. Compared with FLASH64KZ, FLASH64K has a SLEEP port.
If the target device is GW1NS-2/GW1NS-2C/GW1NSR-2/GW1NSR-2C/GW1NSE-2C , the primitive FLASH128K (128K Bytes)will be instantiated in the customized module. For the primitive FLASH128K, data input and data output's width is 32, input ADDR's width is 15.
If the target device is GW1N-2/GW1N-2B/GW1N-4/GW1N-4B/GW1NS-4/GW1NS-4C/GW1NSR-4/GW1NSER-4C/GW1NSR-4C/GW1NR-4 /GW1NR-4B/GW1NRF-4B , the primitive FLASH256K will be instantiated in the customized module. For the primitive FLASH256K, data input and data output's width is 32, input XADR's width is 7, input YADR's width is 6.
If the target device is GW1N-6/GW1N-9/GW1NR-9 , the primitive FLASH608K will be instantiated in the customized module. For the primitive FLASH608K, data input and data output's width is 32, input XADR's width is 9, input YADR's width is 6.

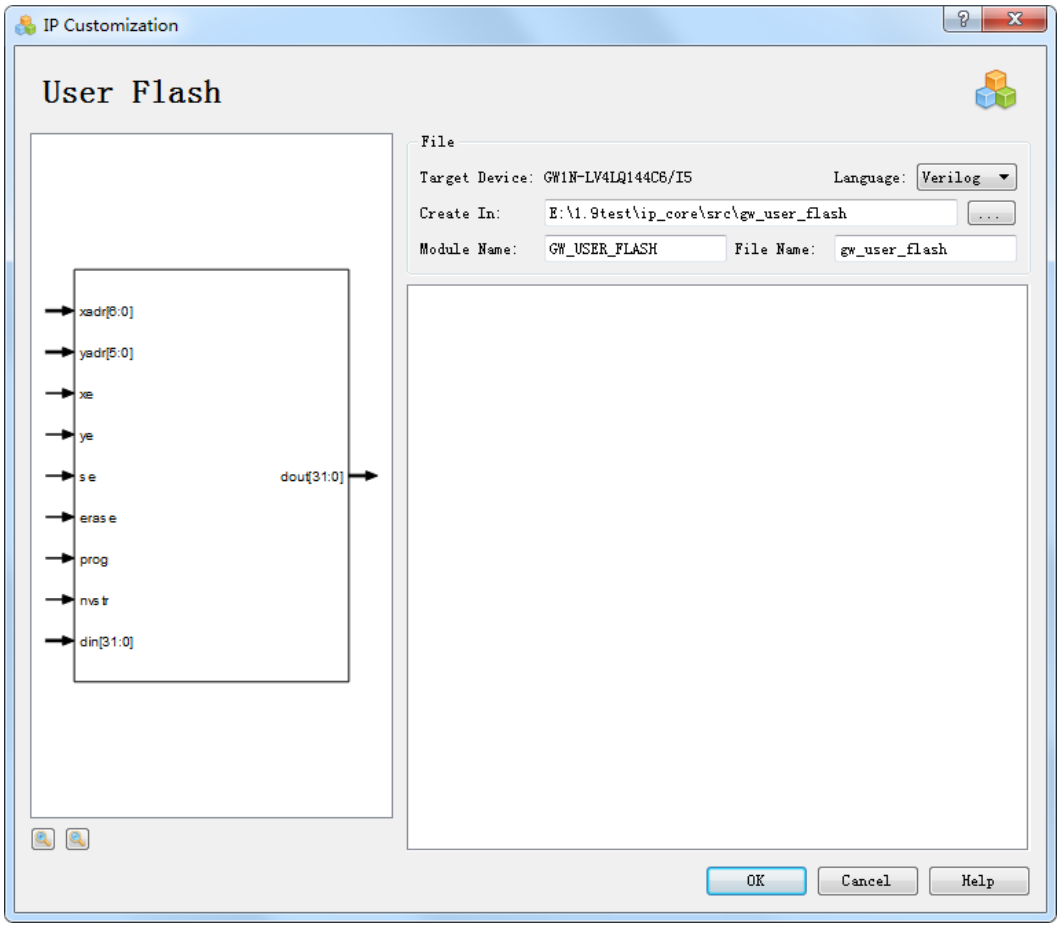
IP Generation Files

As shown in Figure 3-149, after customizing the IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate Gowin Primitive User Flash instantiation "gw_ user_flash.v";
- The instantiation template file for the IP design file "gw_user_lash _tmp.v";
- The configuration file for the Gowin Primitive User Flash instantiation "gw_user_flash.ipc".

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

Figure 3-149 Configured IP Customization



User Flash Design File Instantiation

User Flash design file instantiation is a complete Verilog module. User Flash instantiation is generated according to the User Flash configuration that is displayed in the "IP Customization" window, as shown in Figure 3-150. The generated GW1N-4 design files instantiation is the primitive FLASH256K.

Figure 3-150 User Flash Design File Instantiation

```
module GW_USER_FLASH (dout, xe, ye, se, prog, erase, nvstr, xadr, yadr, din);

output [31:0] dout;
input xe;
input ye;
input se;
input prog;
input erase;
input nvstr;
input [6:0] xadr;
input [5:0] yadr;
input [31:0] din;

FLASH256K flash_inst (
    .DOUT(dout),
    .XE(xe),
    .YE(ye),
    .SE(se),
    .PROG(prog),
    .ERASE(erase),
    .NVSTR(nvstr),
    .XADR(xadr),
    .YADR(yadr),
    .DIN(din)
);

endmodule //GW_USER_FLASH
```

Instantiation Template File for the IP Design File

For practical use, the IP Core Generator generates the template file while generating the user flash design file instantiation, as shown in Figure 3-151.

Figure 3-151 Instantiation Template File for the IP Design File

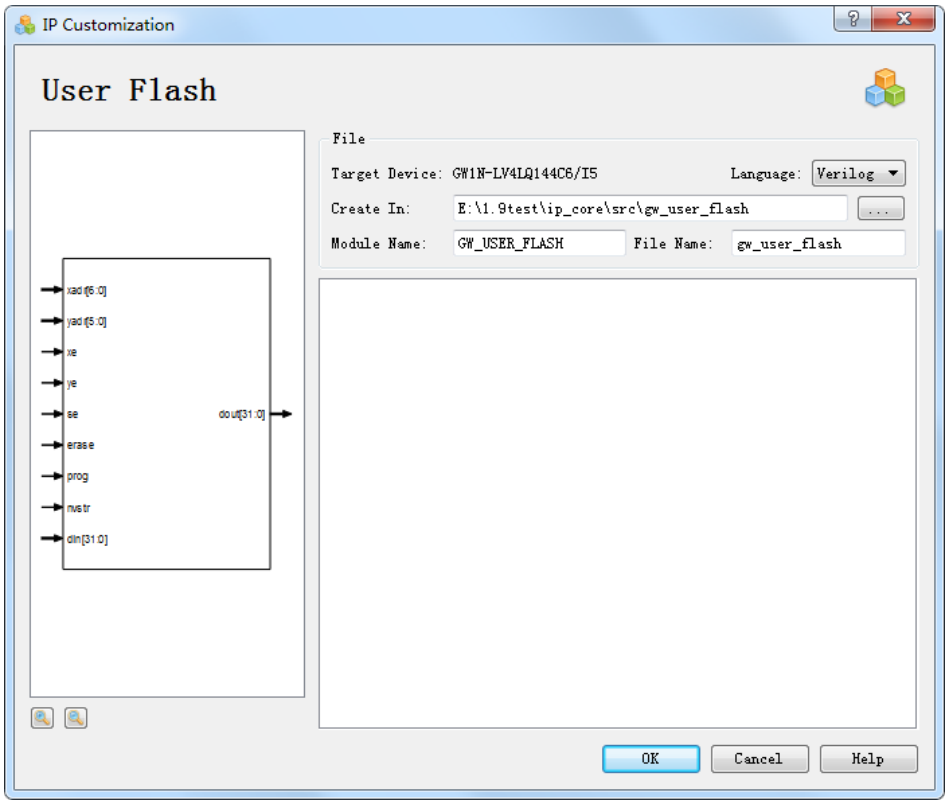
```
GW_USER_FLASH your_instance_name(
    .dout(dout_o), //output [31:0] dout
    .xe(xe_i), //input xe
    .ye(ye_i), //input ye
    .se(se_i), //input se
    .prog(prog_i), //input prog
    .erase(erase_i), //input erase
    .nvstr(nvstr_i), //input nvstr
    .xadr(xadr_i), //input [6:0] xadr
    .yadr(yadr_i), //input [5:0] yadr
    .din(din_i) //input [31:0] din
);
```

User Flash Generation Example

Take FLASH256K generation supported by GW1N-4 for instance, select the GW1N-LV4LQFP144C6/I5 device, as shown in Figure 3-152. Then click "OK" to generate the customized User Flash IP design files.

The directory where the User Flash IP design file is generated is the path for "Create In" In the configuration interface.

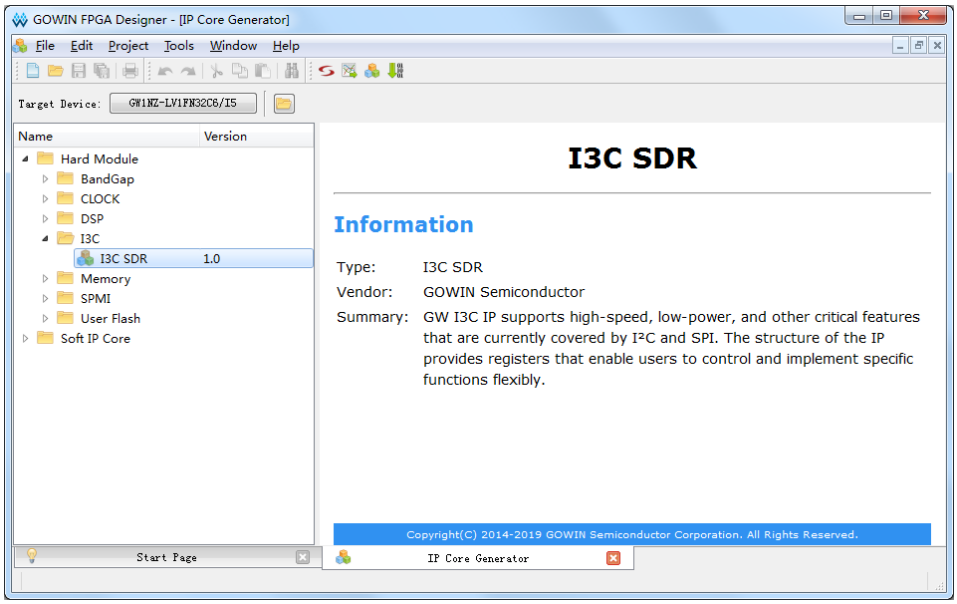
Figure 3-152 IP Customization of User Flash Configuration



3.5 I3C

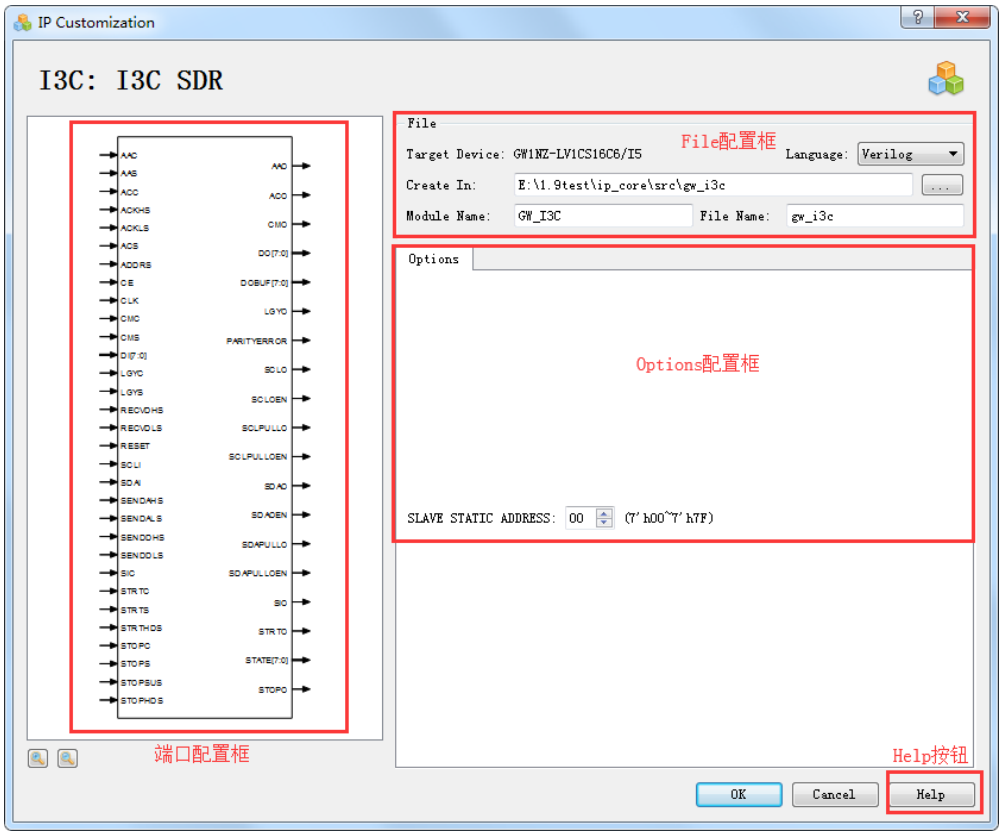
I3C IP offers the features of high-speed and low power and be compatible with the other key features of I2C and SPI. The I3C IP provides registers for users to control and realize specific functions. Click "I3C > I3C SDR" on the "IP Core Generator" page. A brief introduction to the I3C SDR will be displayed on the right of the screen, as shown in Figure 3-153.

Figure 3-153 I3C SDR Summary Information



Double-clicking on "I3C > SDR", and the "IP Customization" window will open as shown in Figure 3-154. This displays the File configuration, Options configuration and port configuration diagram.

Figure 3-154 IP Customization of I3C



1. File Configuration

The file configuration includes the basic information related to the I3C instantiation file, as shown in Figure 3-154.

The I3C file configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP> File Configuration.

Note!

Only GW1NZ-1 supports I3C at present. If you select the other devices as the target device, the I3C will be grey, and no corresponding device can be generated.

2. Ports Configuration Diagram

Ports configuration diagram displays the current IP Core configuration result, as shown in Figure 3-154.

3. Options Configuration

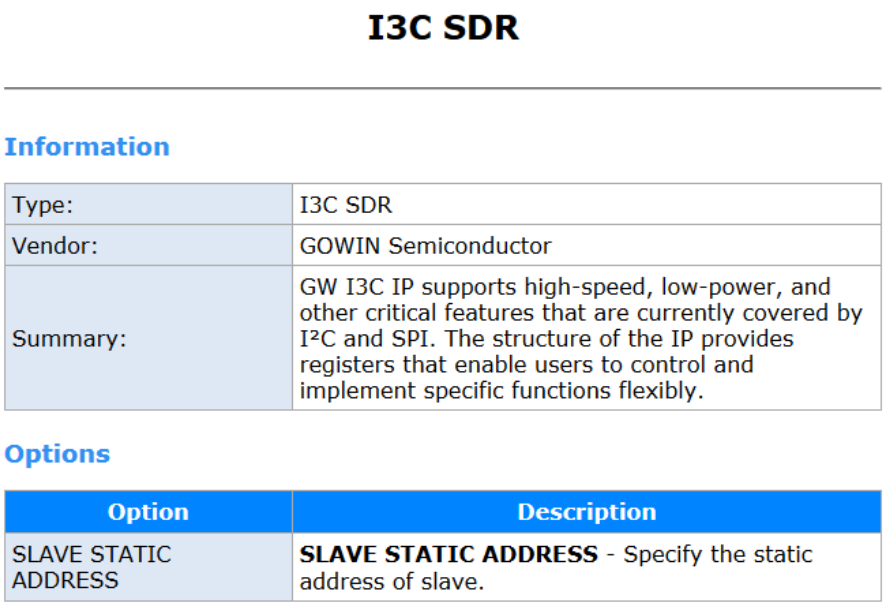
Options configuration includes the configuration information of I3C instantiation file, as shown in Figure 3-154.

SLAVE STATIC ADDRESS - Specify the static address of the Slave.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-155. Help page contains the IP Core general description, and a brief introduction to the "Options".

Figure 3-155 Help



I3C SDR

Information

Type:	I3C SDR
Vendor:	GOWIN Semiconductor
Summary:	GW I3C IP supports high-speed, low-power, and other critical features that are currently covered by I²C and SPI. The structure of the IP provides registers that enable users to control and implement specific functions flexibly.

Options

Option	Description
SLAVE STATIC ADDRESS	**SLAVE STATIC ADDRESS** - Specify the static address of slave.

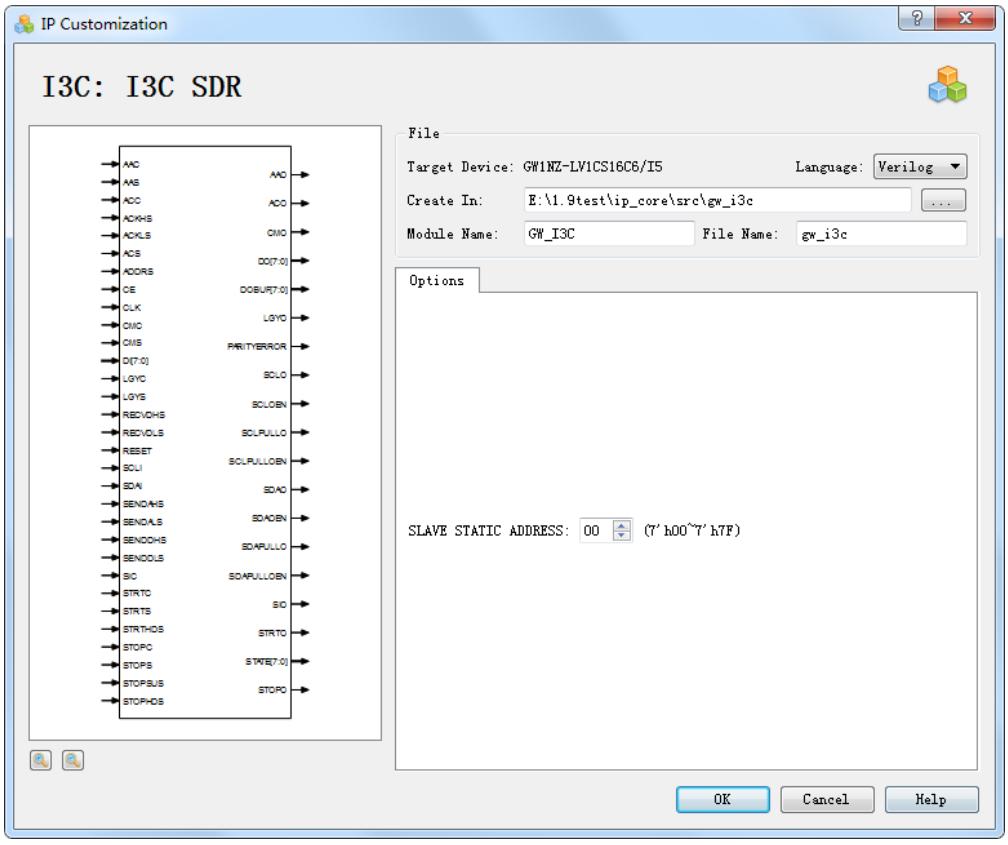
IP Generation Files

As shown in Figure 3-156, after customizing the IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate the Gowin Primitive I3C instantiation "gw_i3c.v";
- The instantiation template file for the IP design file "gw_i3c _tmp.v";
- The configuration file for the Gowin Primitive I3C instantiation "gw_ i3c.ipc".

Taking verilog for instance, the following sections introduce the generated files.

Figure 3-156 Configured IP Customization



I3C Design File Instantiation

The design file for the Gowin Primitive I3C instantiation is a complete Verilog module. I3C instantiation is generated according to the I3C configuration that is displayed in the "IP Customization" window, as shown in Figure 3-157. The generated GW1NZ-1 design files instantiation is the primitive I3C hardcore.

Figure 3-157 I3C Design File Instantiation

```
module GW_I3C (lgyo, cmo, aco, aao, sio, stopo, strto, parityerror, dobuf, dout,
               state, sdao, sclo, sdaoen, scloen, sdapullo, sclpullo, sdapulloen,
               sclpulloen, lgys, cms, acs, aas, stops, strts, lgyc, cmc, acc, aac,
               sic, stopc, strtc, strthds, sendahs, sendals, ackhs, ackls, stopsus,
               stophds, senddhs, senddls, recvdhs, recvdls, addrs, di, sdai, scli,
               ce, reset, clk);

output lgyo;
output cmo;
output aco;
output aao;
output sio;
output stopo;
output strto;
output parityerror;
output [7:0] dobuf;
output [7:0] dout;
output [7:0] state;
output sdao;
output sclo;
output sdaoen;
output scloen;
output sdapullo;
output sclpullo;
output sdapulloen;
output sclpulloen;
input lgys;
input cms;
input acs;
input aas;
input stops;
input strts;
input lgyc;
input cmc;
input acc;
input aac;
input sic;
input stopc;
input strtc;
input strthds;
input sendahs;
input sendals;
input ackhs;
input ackls;
input stopsus;
input stophds;
input senddhs;
input senddls;
input recvdhs;
input recvdls;
input addrs;
input [7:0] di;
input sdai;
input scli;
input ce;
input reset;
input clk;
```

```
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)
);

defparam i3c_inst.ADDRESS = 7'b0000000;

endmodule //GW_I3C

module I3C (
    AAC,        //assert ACK clear
    AAO,        //assert ACK output
    AAS,        //assert ACK set
    ACC,        //assert continuity clear
    ACKHS,      //ACK high period divider
    ACKLS,      //ACK low period divider
    ACO,        //assert continuity output
    ACS,        //assert continuity set
    ADDRS,      //set dynamic address
```

```
    CE,          //clock enable
    CLK,         //clock input
    CMC,         //current master set
    CMO,         //current master output
    CMS,         //current master set
    DI,          //data input
    DO,          //unbuffered data output
    DOBUF,       //buffered data output
    LGYC,        //legacy mode clear
    LGYO,        //legacy mode output
    LGYS,        //enter legacy mode set
    PARITYERROR,//indicater of parit bit error
    RECVDHS,     //set receiving data high period divider
    RECVDLS,     //set receiving data low period divider
    RESET,       //asyn.reset, active high
    SCLI,        //scl input
    SCLO,        //scl output
    SCLOEN,      //scl output enable, active low
    SCLPULLO,    //scl pull-up output
    SCLPULLOEN, //scl pull-up output enable, active low
    SDAI,        //sda input
    SDAO,        //sda output
    SDAOEN,      //sda output enable, active low
    SDAPULLO,    //sda pull-up output
    SDAPULLOEN, //sda pull-up output enable, active low
    SENDAHS,     //set sending address high period divider
    SENDALS,     //set sending address low period divider
    SENDDHS,     //set sending data high period divider
    SENDDLS,     //set sending data low period divider
    SIC,         //system interrupt clear
    SIO,         //system interrupt output
    STRTC,       //start celar
    STRTO,       //start output
    STRTS,       //start set
    STATE,       //state output
    STRTHDS,     //set start hold time
    STOPC,       //stop clear
    STOPO,       //stop output
    STOPS,       //stop set
    STOPSUS,     //set stop setup time
    STOPHDS      //set stop hold time
);

parameter ADDRESS = 7'b0;

input  LGYS, CMS, ACS, AAS, STOPS, STRTS;
output LGYO, CMO, ACO, AAO, SIO, STOPO, STRTO;
input  LGYC, CMC, ACC, AAC, SIC, STOPC, STRTC;
input  STRTHDS, SENDAHS, SENDALS, ACKHS;
input  ACKLS, STOPSUS, STOPHDS, SENDDHS;
input  SENDDLS, RECVDHS, RECVDLS, ADDRS;
output PARITYERROR;
input  [7:0] DI;
output [7:0] DOBUF;
output [7:0] DO;
output [7:0] STATE;
input  SDAI, SCLI;
output SDAO, SCLO;
output SDAOEN, SCLOEN;
output SDAPULLO, SCLPULLO;
output SDAPULLOEN, SCLPULLOEN;
input  CE, RESET, CLK;

endmodule
```

Instantiation Template File for the IP Design File

For practical use, the IP Core Generator generates the template file while generating I3C design file instantiation, as shown in Figure 3-158.

Figure 3-158 Instantiation Template File for the IP Design File

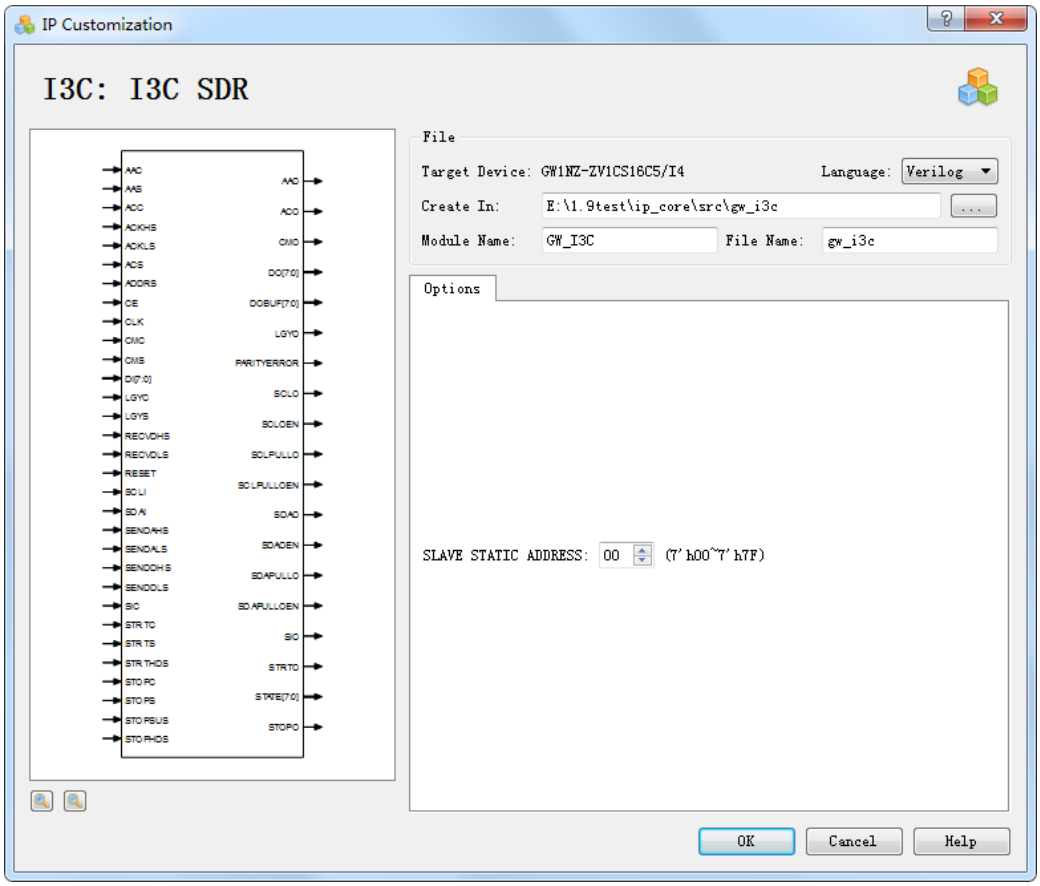
```
GW_I3C your_instance_name(
    .lgyo(lgyo_o), //output lgyo
    .cmo(cmo_o), //output cmo
    .aco(aco_o), //output aco
    .aao(aao_o), //output aao
    .sio(sio_o), //output sio
    .stopo(stopo_o), //output stopo
    .strto(strto_o), //output strto
    .parityerror(parityerror_o), //output parityerror
    .dobuf(dobuf_o), //output [7:0] dobuf
    .dout(dout_o), //output [7:0] dout
    .state(state_o), //output [7:0] state
    .sdao(sdao_o), //output sdao
    .sclo(sclo_o), //output sclo
    .sdaoen(sdaoen_o), //output sdaoen
    .scloen(scloen_o), //output scloen
    .sdapullo(sdapullo_o), //output sdapullo
    .sclpullo(sclpullo_o), //output sclpullo
    .sdapulloen(sdapulloen_o), //output sdapulloen
    .sclpulloen(sclpulloen_o), //output sclpulloen
    .lgys(lgys_i), //input lgys
    .cms(cms_i), //input cms
    .acs(acs_i), //input acs
    .aas(aas_i), //input aas
    .stops(stops_i), //input stops
    .strts(strts_i), //input strts
    .lgyc(lgyc_i), //input lgyc
    .cmc(cmc_i), //input cmc
    .acc(acc_i), //input acc
    .aac(aac_i), //input aac
    .sic(sic_i), //input sic
    .stopc(stopc_i), //input stopc
    .strtc(strtc_i), //input strtc
    .strthds(strthds_i), //input strthds
    .sendahs(sendahs_i), //input sendahs
    .sendals(sendals_i), //input sendals
    .ackhs(ackhs_i), //input ackhs
    .ackls(ackls_i), //input ackls
    .stopsus(stopsus_i), //input stopsus
    .stophds(stophds_i), //input stophds
    .senddhs(senddhs_i), //input senddhs
    .senddls(senddls_i), //input senddls
    .recvdhs(recvdhs_i), //input recvdhs
    .recvdls(recvdls_i), //input recvdls
    .addrs(addrs_i), //input addrs
    .di(di_i), //input [7:0] di
    .sdai(sdai_i), //input sdai
    .scli(scli_i), //input scli
    .ce(ce_i), //input ce
    .reset(reset_i), //input reset
    .clk(clk_i) //input clk
);
```

I3C Generation Example

As shown in Figure 3-159, take I3C generation supported by GW1NZ-1 for instance, select the GW1NZ-LV1CS16C5/I4 device and configure the "Options" in the "IP Customization" window. Then click "OK" to generate the customized I3C IP design files.

The directory where the I3C IP design file is generated is the path set in "Create In".

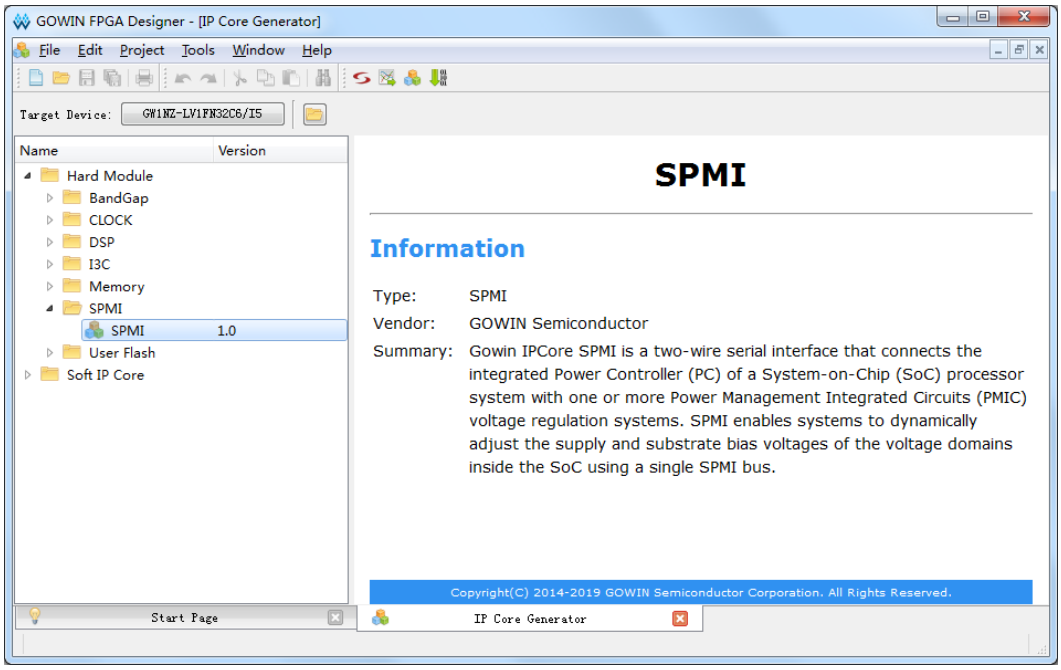
Figure 3-159 I3C IP Customization Configuration



3.6 SPMI

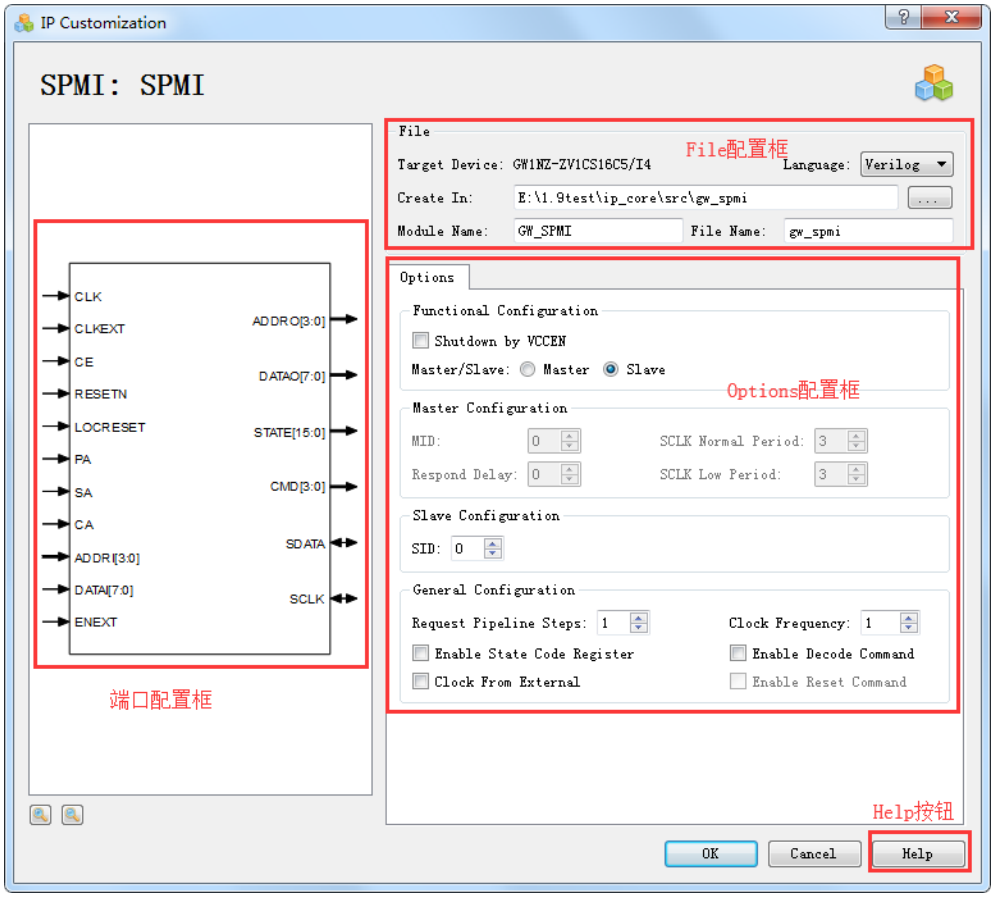
SPMI is a two-wire serial interface and it can make Power Controller (PC) integrated in System-on-Chip (SoC) connect with one or more voltage regulation systems of Power Management Integrated Circuits (PMIC) . SPMI enables systems to dynamically adjust the supply and substrate bias voltages of the voltage domains inside the SoC using a single SPMI bus. Click "SPMI" on the IP Core Generator interface. A brief introduction to the SPMI will be displayed on the right of the screen, as shown in Figure 3--160.

Figure 3--160 SPMI Summary Information



Double click the "SPMI" to open the "IP Customization" window. This includes the "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button, as shown in Figure 3-161.

Figure 3-161 IP Customization of SPMI



1. File Configuration

The file configuration includes the basic information related to the SPMI instantiation file, as shown in Figure 3-161.

The SPMI file configuration is similar to that of SP. For the detailed configuration, please refer to 3.1Block Memory>3.1.1SP > File Configuration.

Note!

Only GW1NZ-1 supports SPMI at present. If you select the other devices as the target device, the SPMI will be grey, and no corresponding device can be generated.

2. Options Configuration

Options configuration includes the configuration information of SPMI instantiation file, as shown in Figure 3-161.

- Functional Configuration：
  - Shutdown by VCCEN: Shutdown by external pin VCCEN If this option is checked, the communication function of SPMI will be disabled.
  - Master/Slave: Set SPMI as Master or Slave.
- Master Configuration:
  - MID: Master ID. The range is 0-3, and default value is 0.
  - Respond Delay: Set the response delay time.
  - SCLK Normal Period: Set SCLK period in normal mode.
  - SCLK Normal Period: Set SCLK period in lowl mode.
- Slave Configuration:
  - SID: Set the ID of the SPMI Slave.
- General configuration:
  - Enable State Code Register: Enable or disable the state code register. If "Enable State Code Register" is checked, the output state code will pass a register.
  - Request Pipeline Steps: Set the sampling time delay step of the request signal.
  - Enable Decode Command: Enable or disable decode. If "Enable Decode Command" is checked, SPMI will decode the reset, sleep, shutdown, and wakeup commands.
  - Enable Decode Command: Enable or disable reset command.
  - Clock From External: Enable or disable the external clock.
  - Clock Frequency: System clock frequency.

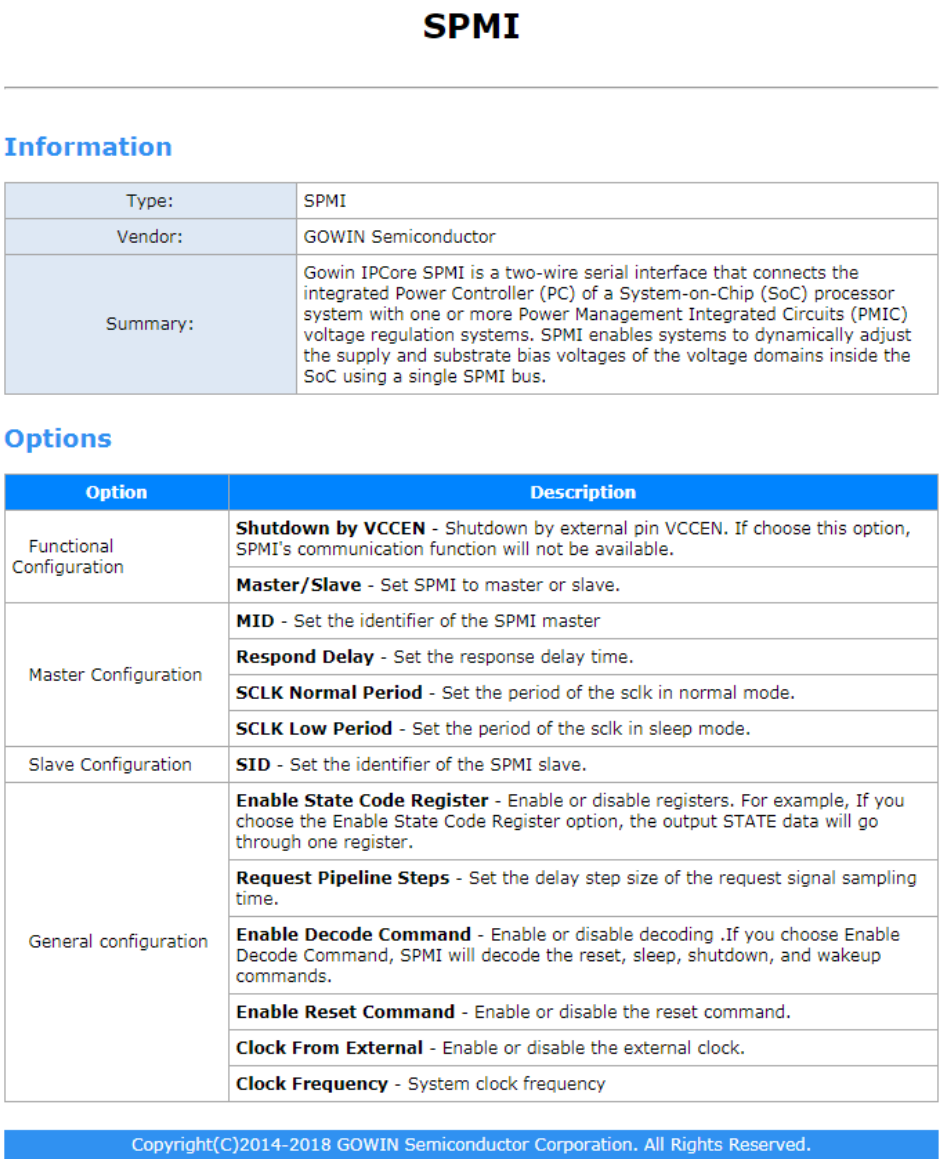
3. Ports Configuration Diagram

Ports configuration diagram displays the current IP Core configuration result, as shown in Figure 3-161.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-162.

Figure 3-162 Help



SPMI

Information

Type:	SPMI
Vendor:	GOWIN Semiconductor
Summary:	Gowin IPCore SPMI is a two-wire serial interface that connects the integrated Power Controller (PC) of a System-on-Chip (SoC) processor system with one or more Power Management Integrated Circuits (PMIC) voltage regulation systems. SPMI enables systems to dynamically adjust the supply and substrate bias voltages of the voltage domains inside the SoC using a single SPMI bus.

Options

Option	Description
Functional Configuration	**Shutdown by VCCEN** - Shutdown by external pin VCCEN. If choose this option, SPMI's communication function will not be available.
	Master/Slave - Set SPMI to master or slave.
Master Configuration	**MID** - Set the identifier of the SPMI master
	Respond Delay - Set the response delay time.
	SCLK Normal Period - Set the period of the sclk in normal mode.
	SCLK Low Period - Set the period of the sclk in sleep mode.
Slave Configuration	**SID** - Set the identifier of the SPMI slave.
General configuration	**Enable State Code Register** - Enable or disable registers. For example, If you choose the Enable State Code Register option, the output STATE data will go through one register.
	Request Pipeline Steps - Set the delay step size of the request signal sampling time.
	Enable Decode Command - Enable or disable decoding .If you choose Enable Decode Command, SPMI will decode the reset, sleep, shutdown, and wakeup commands.
	Enable Reset Command - Enable or disable the reset command.
	Clock From External - Enable or disable the external clock.
	Clock Frequency - System clock frequency

The "Help" page contains a general description of the IP Core, and a brief introduction to the "Options".

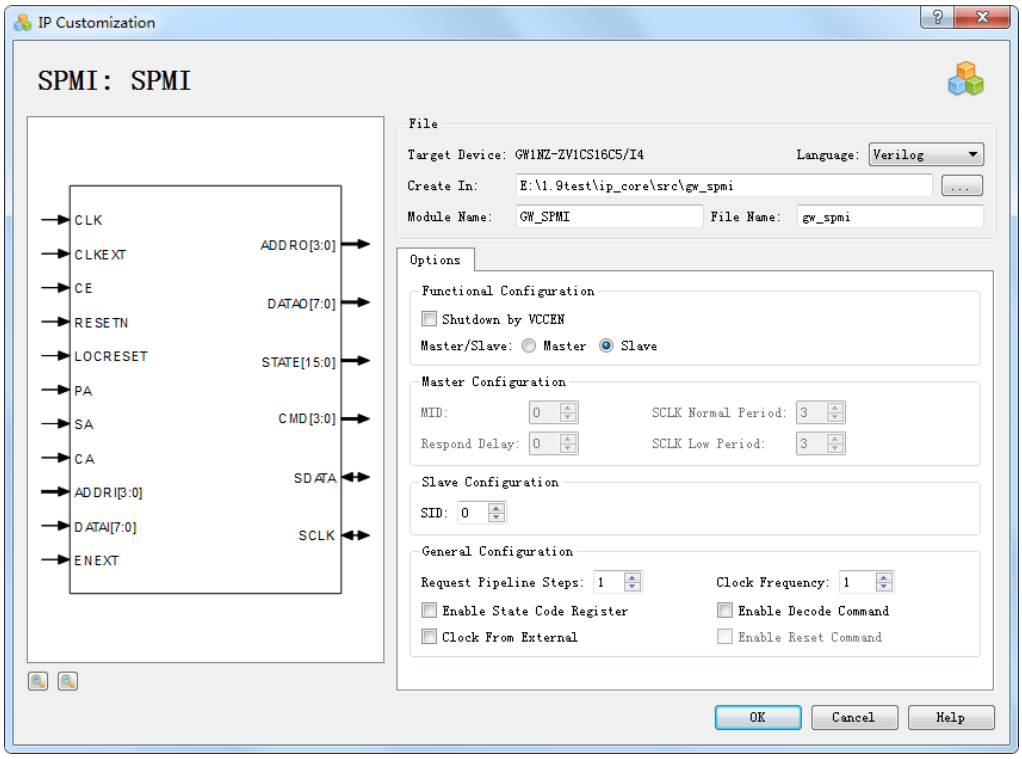
IP Generation Files

As shown in Figure 3-163, after customizing the IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate the Gowin primitive SPMI instantiation "gw_spmi.v";
- The instantiation template file for the IP design file "gw_spmi_tmp.v";
- The configuration files for the Gowin Primitive SPMI instantiation "gw_sp.ipc".

Taking verilog for instance, the following sections introduce the generated files.

Figure 3-163 Configured IP Customization



SPMI Design File Instantiation

SPMI design file instantiation is a complete Verilog module. SPMI instantiation is generated according to the SPMI configuration that is displayed in the "IP Customization" window, as shown in Figure 3-164.

Figure 3-164 SPMI Design File Instantiation

```
module GW_SPMI (addro, datao, state, cmd, sdata, sclk, clk, ce, resetn,
                locreset, pa, sa, ca, addri, datai, clkext, enext);

output [3:0] addro;
output [7:0] datao;
output [15:0] state;
output [3:0] cmd;
inout sdata;
inout sclk;
input clk;
input ce;
input resetn;
input locreset;
input pa;
input sa;
input ca;
input [3:0] addri;
input [7:0] datai;
input clkext;
input enext;

SPMI spmi_inst (
    .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)
);

defparam spmi_inst.FUNCTION_CTRL = 7'b0000100;
defparam spmi_inst.MSID_CLKSEL = 7'b0000000;
defparam spmi_inst.RESPOND_DELAY = 4'b0000;
defparam spmi_inst.SCLK_NORMAL_PERIOD = 7'b0000011;
defparam spmi_inst.SCLK_LOW_PERIOD = 7'b0000011;
defparam spmi_inst.CLK_FREQ = 7'b0000000;
defparam spmi_inst.SHUTDOWN_BY_ENABLE = 1'b0;

endmodule //GW_SPMI

module SPMI (CLK, CLKEXT, CE, RESETN, ENEXT, LOCRESET, PA, SA, CA, ADDRI,
             DATAI, ADDRO, DATAO, STATE, CMD, SDATA, SCLK)
/* synthesis syn_black_box black_box_pad_pin="SDATA,SCLK" syn_noprune = 1*/;

parameter FUNCTION_CTRL = 7'b0;
parameter MSID_CLKSEL = 7'b0;
parameter RESPOND_DELAY = 4'b0;
parameter SCLK_NORMAL_PERIOD = 7'b0;
parameter SCLK_LOW_PERIOD = 7'b0;
parameter CLK_FREQ = 7'b0;
parameter SHUTDOWN_BY_ENABLE = 1'b0;

input  CLKEXT, ENEXT;
inout  SDATA, SCLK;
input  CLK, CE, RESETN, LOCRESET;
input  PA, SA, CA;
input  [3:0] ADDRI;
input  [7:0] DATAI;
output [3:0] ADDRO;
output [7:0] DATAO;
output [15:0] STATE;
output [3:0] CMD;

endmodule
```

Instantiation Template File for the IP Design File

For the practical use, the IP Core Generator generates the template file while generating the SPMI design file instantiation, as shown in Figure 3-165.

Figure 3-165 Instantiation Template File for the IP Design File

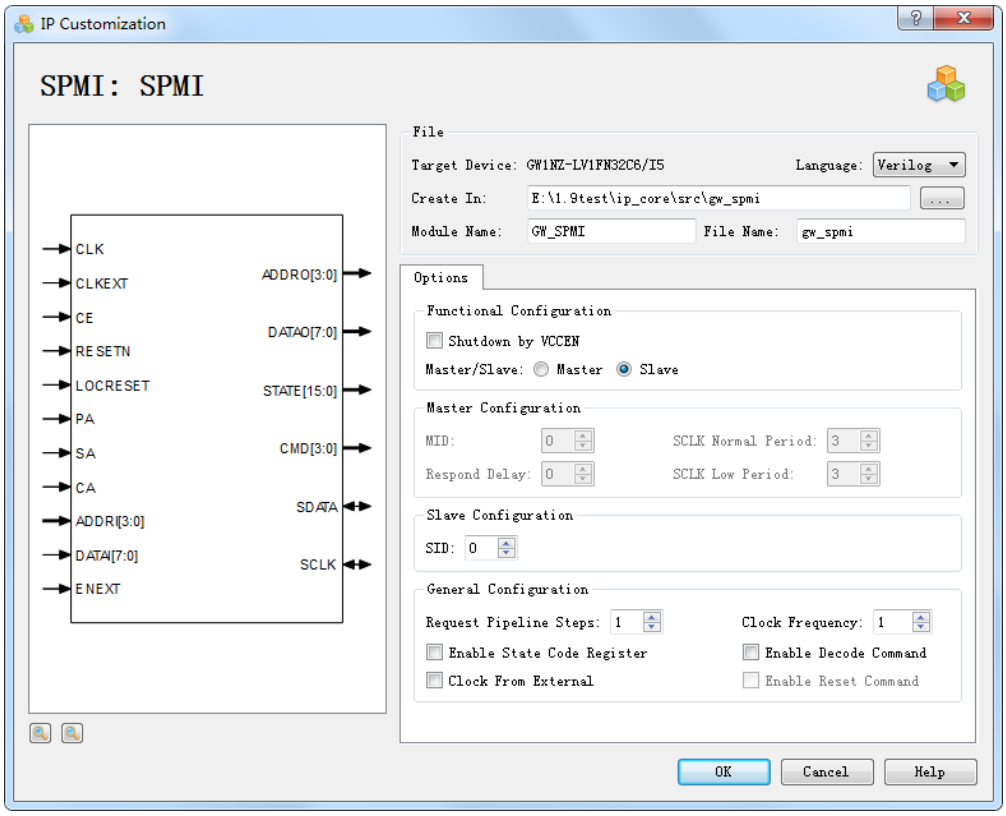
```
GW_SPMI your_instance_name(
    .addro(addro_o), //output [3:0] addro
    .datao(datao_o), //output [7:0] datao
    .state(state_o), //output [15:0] state
    .cmd(cmd_o), //output [3:0] cmd
    .sdata(sdata_io), //inout sdata
    .sclk(sclk_io), //inout sclk
    .clk(clk_i), //input clk
    .ce(ce_i), //input ce
    .resetn(resetn_i), //input resetn
    .locreset(locreset_i), //input locreset
    .pa(pa_i), //input pa
    .sa(sa_i), //input sa
    .ca(ca_i), //input ca
    .addri(addri_i), //input [3:0] addri
    .datai(datai_i), //input [7:0] datai
    .clkext(clkext_i), //input clkext
    .enext(enext_i) //input enext
);
```

SPMI Generation Example

As shown in Figure 3-166, take SPMI generation supported by GW1NZ-1 for instance, select the GW1NZ-LV1FN32C6/I5 device and configure the "Options" in the "IP Customization" window. Then click "OK" to generate the customized SPMI IP design files.

The directory where the SPMI IP design file is generated is the path set in "Create In".

Figure 3-166 SPMI IP Customization Setting



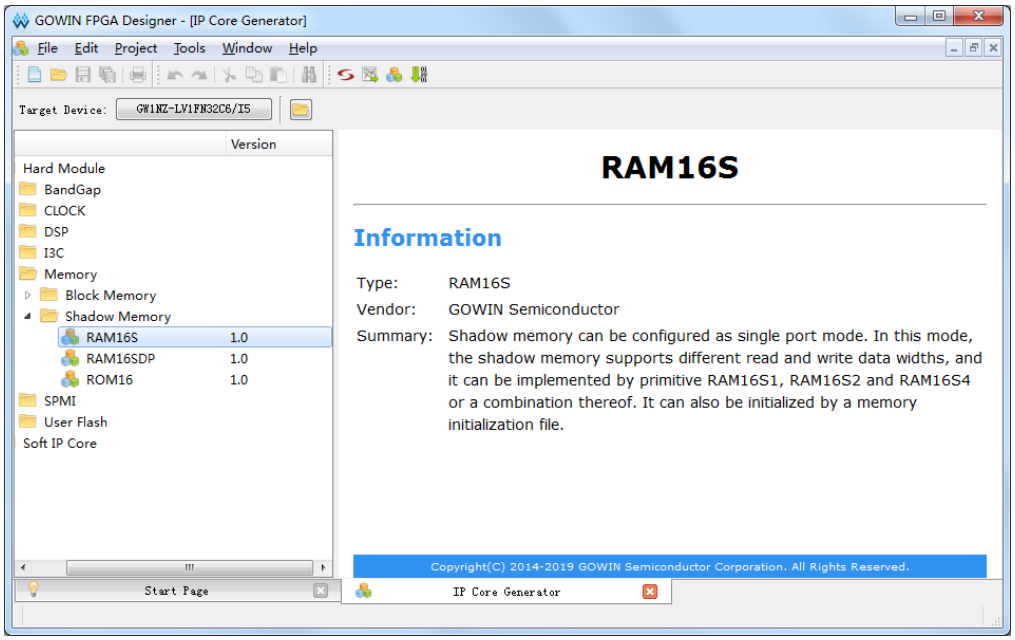
3.7 Shadow Memory

Currently, the Shadow Memory (SSRAM) module implements RAM16S (single-port mode), RAM16SDP (semi-dual-port mode), and ROM16 (read-only mode).

3.7.1 RAM16S

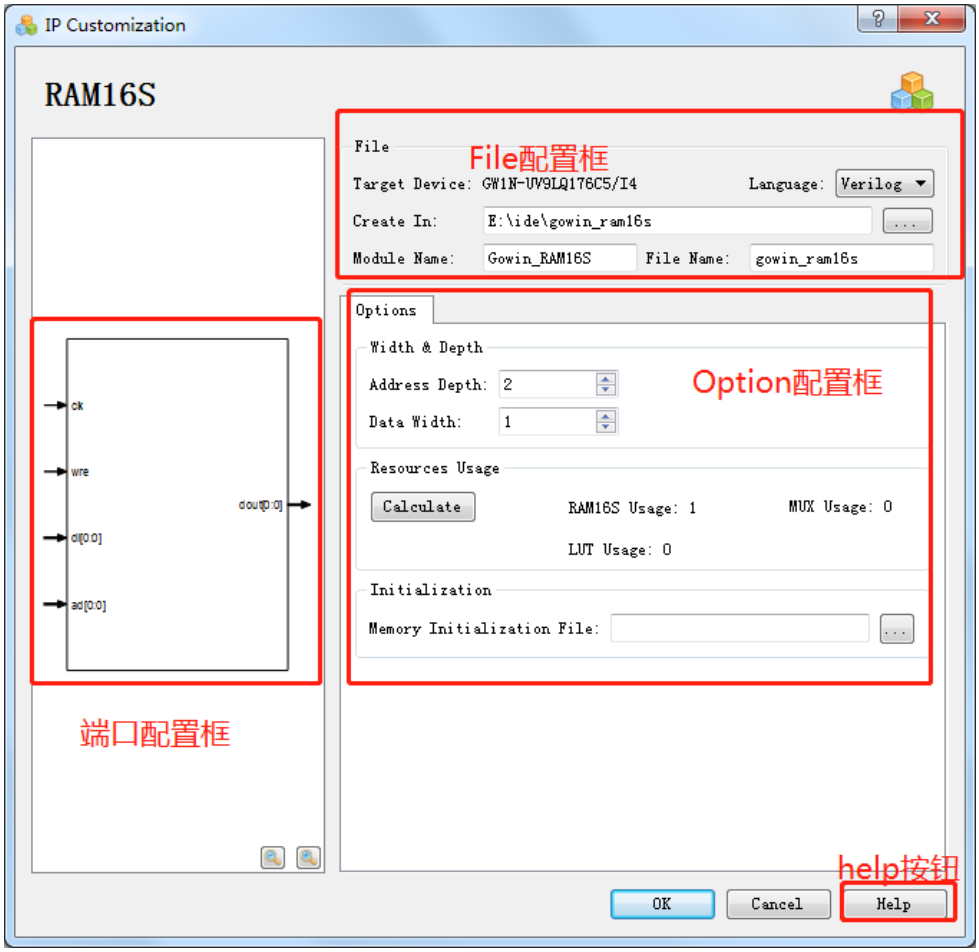
RAM16S is a single port shadow memory that can be implemented by RAM16S1, RAM16S2, RAM16S4. The maximum capacity of SSRAM varies according to the type of chip. Click "RAM16S" on the IP Core Generator page. A brief introduction to the RAM16S will be displayed on the right of the screen, as shown in Figure 3-167.

Figure 3-167 RAM16S Summary Information



Double click the "RAM16S" to open the "IP Customization" window. This includes the "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button, as shown in Figure 3-168.

Figure 3-168 IP Customization of RAM16S



1. File Configuration

The file configuration includes the basic information related to the RAM16S instantiation file, as shown in Figure 3-168.

The RAM16S file configuration is similar to that of SP. For the detailed configuration, please refer to 3.1Block Memory>3.1.1SP > File Configuration.

2. Options Configuration

The Options configuration box is used to configure sinle port shadow memory by users, as shown in Figure 3-168.

RAM16S Options configuration is similar to that of SP. For the detailed configuration information, please refer to 3.1 Block Memory>3.1.1SP > Options Configuration.

3. Ports Configuration Diagram

- The ports configuration diagram displays the current IP Core configuration result. Input/Output bit-width updates in real time based on the "Options" configuration, as shown in Figure 3-168.
- "Address Depth" affects the bit-width of ad; "Data Width" affects the bit-width of din and dout.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-169.

Figure 3-169 Help

RAM16S

Information

Type:	RAM16S
Vendor:	GOWIN Semiconductor
Summary:	Shadow memory can be configured as single port mode. In this mode, the shadow memory supports different read and write data widths, and it can be implemented by primitive RAM16S1, RAM16S2 and RAM16S4 or a combination thereof. It can also be initialized by a memory initialization file.

Options

Option	Description
Width & Depth	**Address Depth** - Set the size of the address depth.
	Data Width - Set the size of the data width.
Resources Usage	**Calculate** - Calculate the resource usage in the design and display results below.
	RAM16S Usage - Display the number of RAM16S used.
	LUT Usage - Display the number of LUT used.
	MUX Usage - Display the number of MUX used.
Initialization	**Memory Initialization File** - Set the memory initialization file (.mi) path.

The "Help" page contains a general description of the IP Core, and a brief introduction to the "Options".

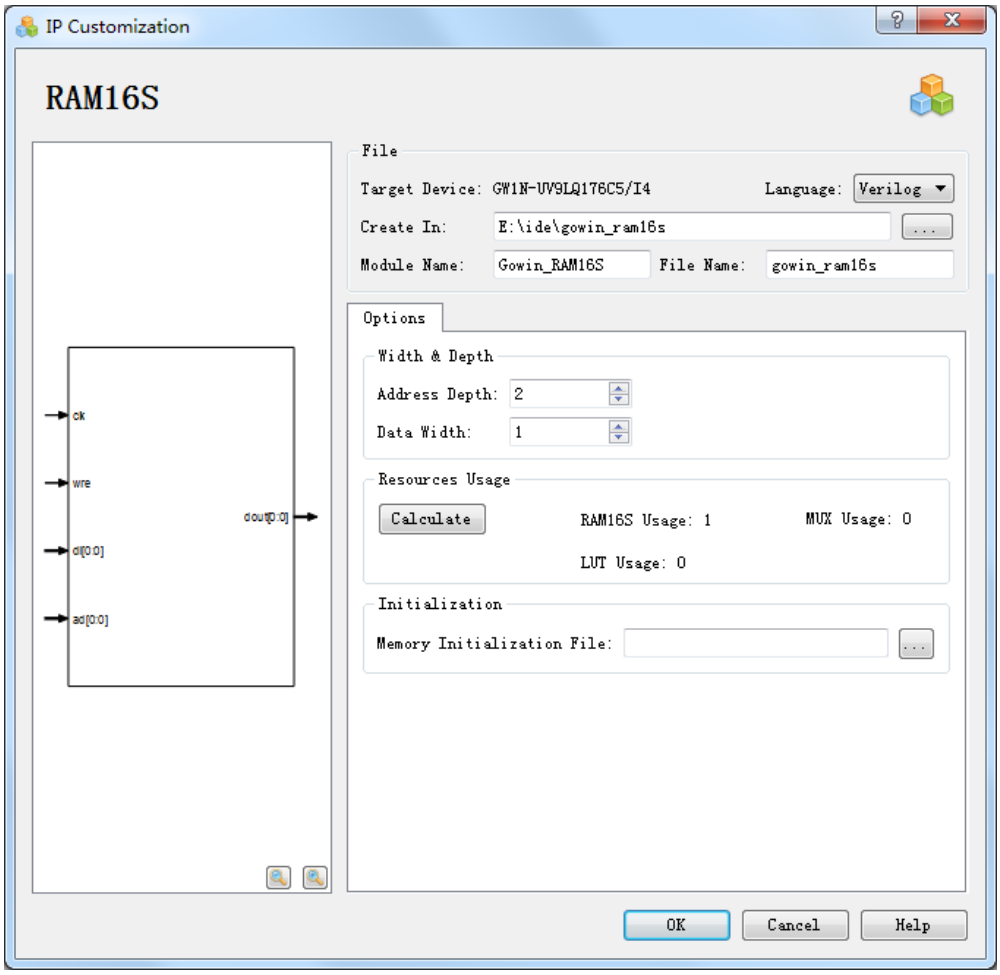
IP Generation Files

As shown in Figure 3-170, after customizing the IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate Gowin Primitive RAM16S design file "gowin_ram16s.v";
- The instantiation template file for the IP design file "gowin_ram16s_tmp.v";
- The configuration file for the Gowin Primitive RAM16S instantiation "gowin_ram16s.ipc".

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

Figure 3-170 Configured IP Customization



RAM16S Design File Instantiation

The design file for the Gowin Primitive RAM16S instantiation is a complete Verilog module. RAM16S instantiation is generated according to the RAM16S configuration that is displayed in the "IP Customization" window, as shown in Figure 3-171.

Figure 3-171 RAM16S Design File Instantiation

```
module Gowin_RAM16S (dout, di, ad, wre, clk);

output [0:0] dout;
input [0:0] di;
input [0:0] ad;
input wre;
input clk;

wire gw_gnd;

assign gw_gnd = 1'b0;

RAM16S1 ssram_spx1_0 (
    .DO(dout[0]),
    .DI(di[0]),
    .AD({gw_gnd,gw_gnd,gw_gnd,ad[0]}),
    .WRE(wre),
    .CLK(clk)
);

defparam ssram_spx1_0.INIT_0 = 16'h0000;

endmodule //Gowin_RAM16S
```

Instantiation Template File for the IP Design File

For practical use, the IP Core Generator generates the template file while generating RAM16S design file instantiation, as shown in Figure 3-172.

Figure 3-172 Instantiation Template File for the IP Design File

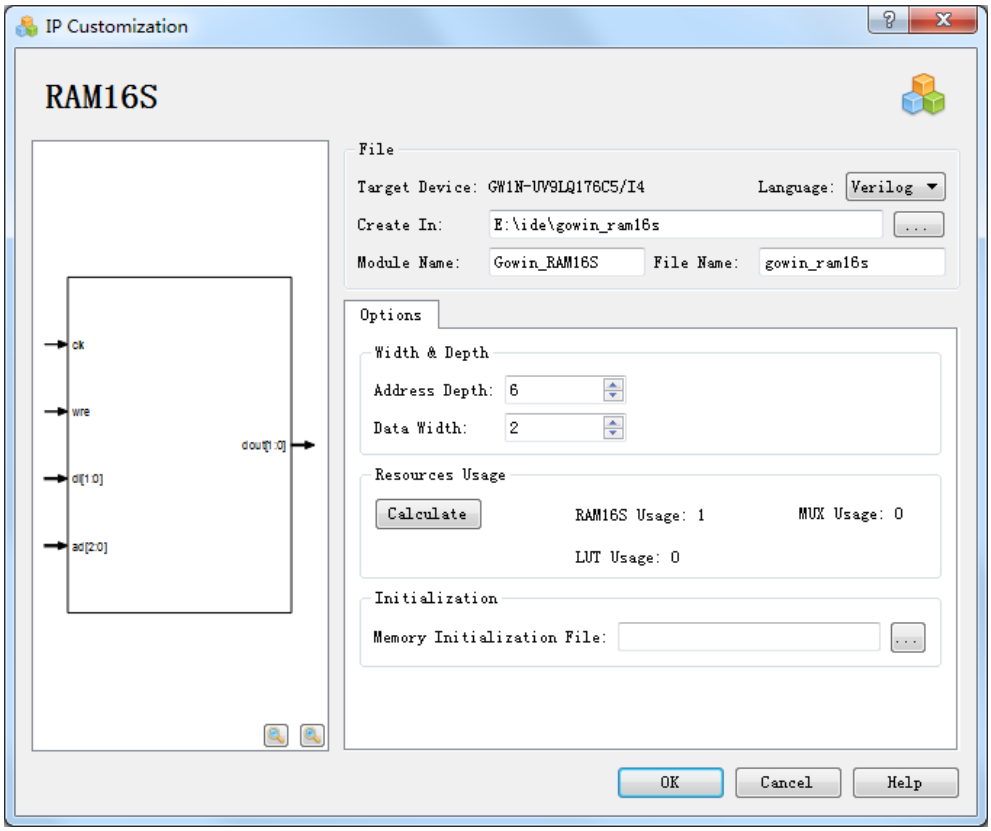
```
    Gowin_RAM16S your_instance_name(
        .dout(dout_o), //output [0:0] dout
        .di(di_i), //input [0:0] di
        .ad(ad_i), //input [0:0] ad
        .wre(wre_i), //input wre
        .clk(clk_i) //input clk
    );
```

RAM16S Generation Example

Generate a specific RAM16S IP as follows: address width and data width: 6 x 2; Take the GW1N-UV9LQ176C6/I5 device for instance, the configuration interface is as shown in Figure 3-173. Select a memory initialization file for the module as required, and then click "OK" to generate the customized RAM16S IP design files.

The directory where the RAM16S IP design file is generated is the path set in "Create In".

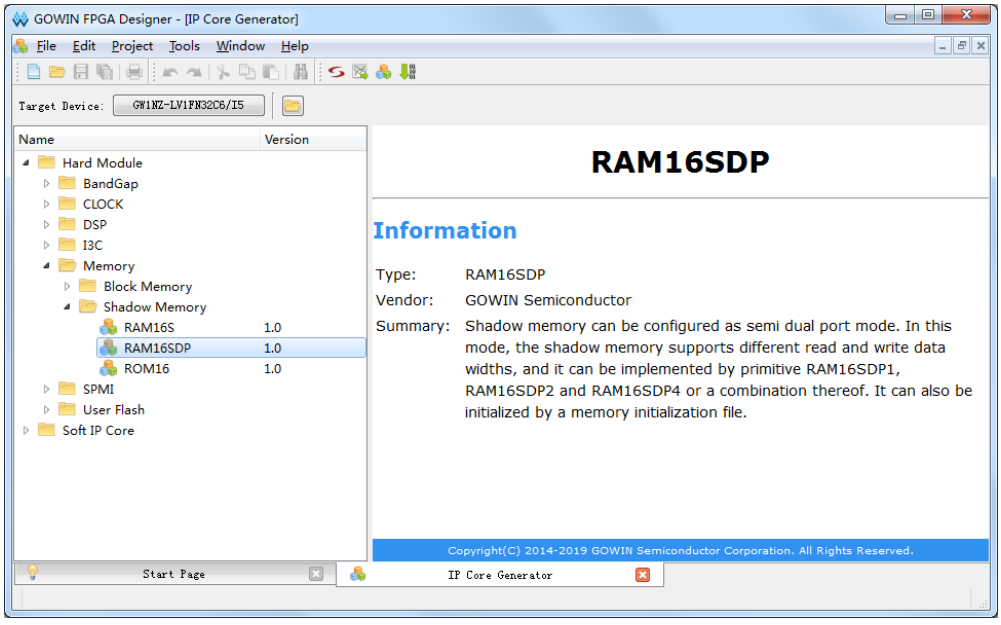
Figure 3-173 RAM16S IP Customization Configuration



3.7.2 RAM16SDP

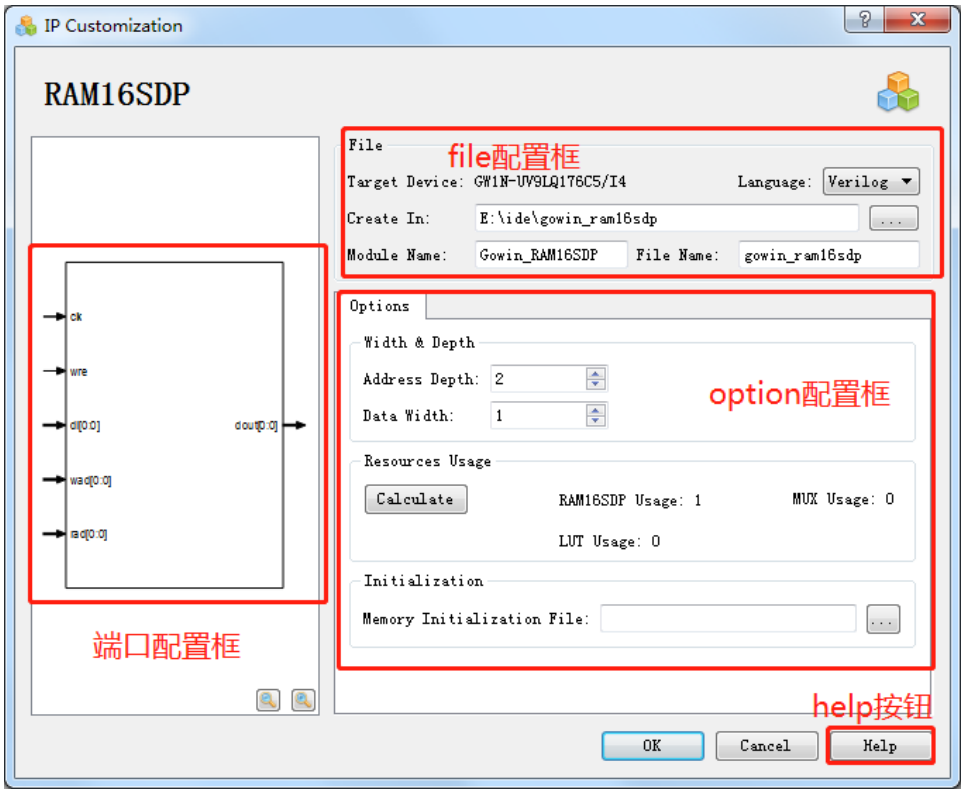
RAM16S is a single port shadow memory that can be implemented by RAM16SDP1, RAM16SDP2 and RAM16SDP4. The maximum capacity of SSRAM varies according to the type of chip. Click "RAM16SDP" on the IP Core Generator page. A brief introduction to the RAM16SDP will be displayed on the right of the screen, as shown in Figure 3-174.

Figure 3-174 RAM16SDP Summary Information



Double click the "RAM16SDP" to open the "IP Customization" window. This includes the "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button, as shown in Figure 3-175.

Figure 3-175 IP Customization of RAM16SDP



1. File Configuration

The file configuration includes the basic information related to the RAM16SDP instantiation file, as shown in Figure 3-175.

The RAM16SDP file configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP> File Configuration.

2. Options Configuration

The Options configuration box is used to configure semi-dual port block memory by users, as shown in Figure 3-175.

RAM16SDP options configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP> Options Configuration.

3. Ports Configuration Diagram

- The ports configuration diagram displays the current IP Core configuration result. Input/Output bit-width updates in real time based on the "Options" configuration, as shown in Figure 3-175.
- "Address Depth" affects the bit-width of wad and rad; "Data Width" affects the bit-width of di and dout.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-176.

Figure 3-176 Help

RAM16SDP

Information

Type:	RAM16SDP
Vendor:	GOWIN Semiconductor
Summary:	Shadow memory can be configured as semi dual port mode. In this mode, the shadow memory supports different read and write data widths, and it can be implemented by primitive RAM16SDP1, RAM16SDP2 and RAM16SDP4 or a combination thereof. It can also be initialized by a memory initialization file.

Options

Option	Description
Width & Depth	**Address Depth** - Set the size of the address depth.
	Data Width - Set the size of the data width.
Resources Usage	**Calculate** - Calculate the resource usage in the design and display results below.
	RAM16SDP Usage - Display the number of RAM16SDP used.
	LUT Usage - Display the number of LUT used.
	MUX Usage - Display the number of MUX used.
Initialization	**Memory Initialization File** - Set the memory initialization file (.mi) path.

The "Help" page contains a general description of the IP Core, and a brief introduction to the "Options".

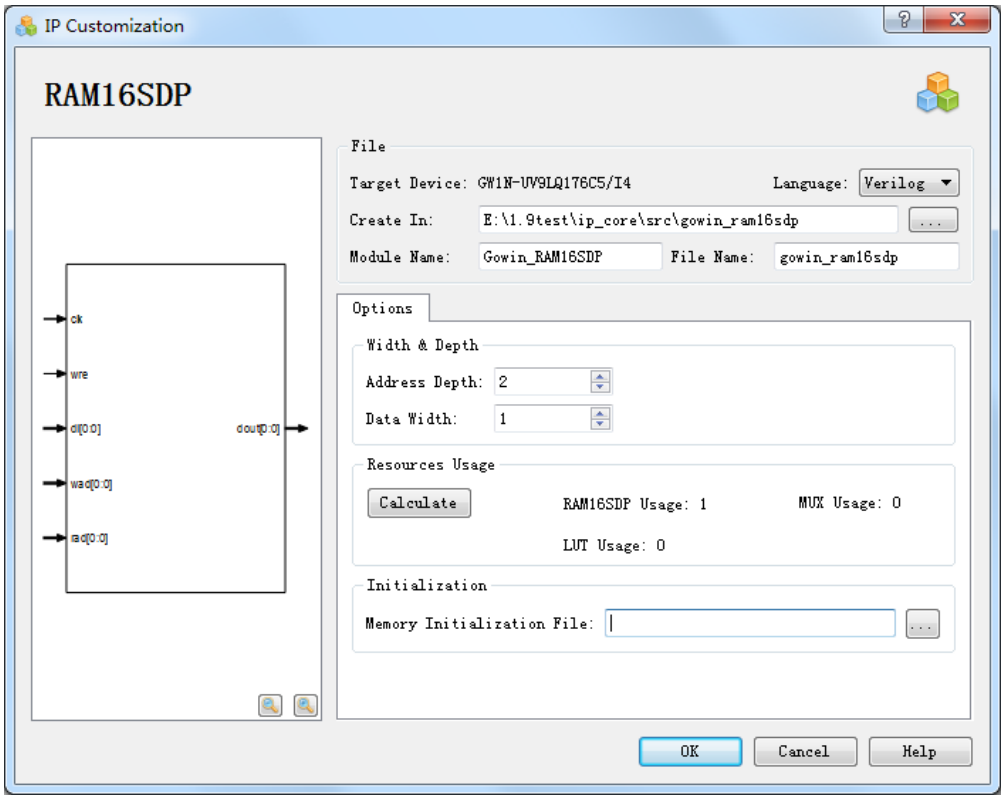
IP Generation Files

As shown in Figure 3-177, after customizing the IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate Gowin Primitive RAM16SDP design file "gowin_ram16sdp.v";
- The instantiation template file for the IP design file "gowin_ram16sdp_tmp.v";
- The configuration file for the Gowin Primitive RAM16SDP instantiation "gowin_ram16sdp.ipc".

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.

Figure 3-177 Configured IP Customization



RAM16SDP Design File Instantiation

The design file for the Gowin Primitive RAM16SDP instantiation is a complete Verilog module. RAM16SDP instantiation is generated according to the RAM16SDP configuration that is displayed in the "IP Customization" window, as shown in Figure 3-178.

Note!

Di/Dout data width of the generated RAM16SDP instantiation is consistent with that of the RAM16SDP configured in the "IP Customization" window.

Figure 3-178 RAM16SDP Design File Instantiation

```
module Gowin_RAM16SDP (dout, di, wad, rad, wre, clk);

output [0:0] dout;
input [0:0] di;
input [0:0] wad;
input [0:0] rad;
input wre;
input clk;

wire gw_gnd;

assign gw_gnd = 1'b0;

RAM16SDP1 ssram_sdpx1_0 (
    .DO(dout[0]),
    .DI(di[0]),
    .WAD({gw_gnd,gw_gnd,gw_gnd,wad[0]}),
    .RAD({gw_gnd,gw_gnd,gw_gnd,rad[0]}),
    .WRE(wre),
    .CLK(clk)
);

defparam ssram_sdpx1_0.INIT_0 = 16'h0000;

endmodule //Gowin_RAM16SDP
```

Instantiation Template File for the IP Design File

For the practical use, the IP Core Generator generates the template file while generating the RAM16SDP design file instantiation, as shown in Figure 3-179.

Figure 3-179 Instantiation Template File for the IP Design File

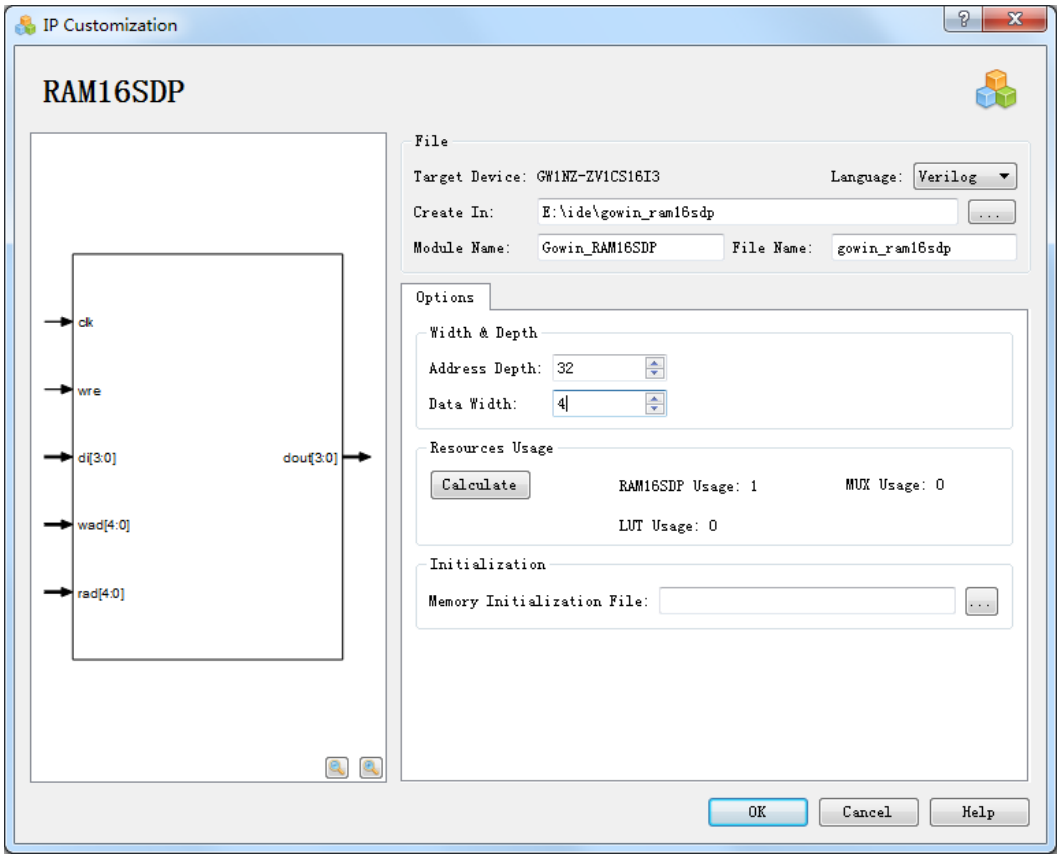
```
Gowin_RAM16SDP your_instance_name(
    .dout(dout_o), //output [0:0] dout
    .di(di_i), //input [0:0] di
    .wad(wad_i), //input [0:0] wad
    .rad(rad_i), //input [0:0] rad
    .wre(wre_i), //input wre
    .clk(clk_i) //input clk
);
```

RAM16SDP Generation Example

Generate a specific RAM16SDP IP as follows: address width and data width: 32 x 4; Take the GW1N-UV9LQ176C6/I5 device for instance, the configuration interface is as shown in Figure 3-180. Select a memory initialization file for the module as required, and then click "OK" to generate the customized RAM16SDP IP design files.

The directory where theRAM16SDP IP design file is generated is the path set in "Create In".

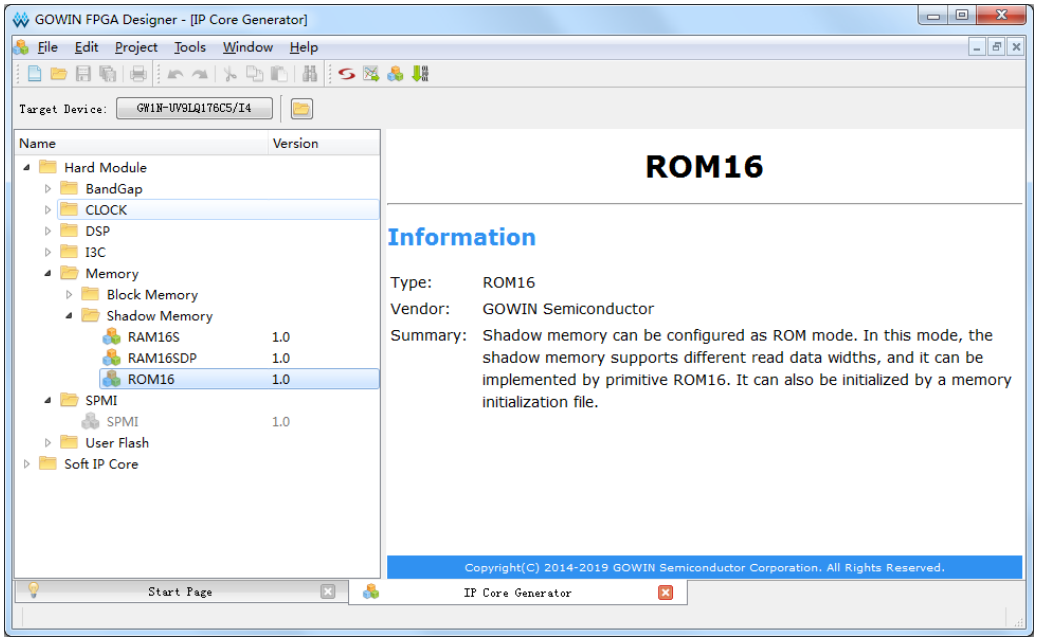
Figure 3-180 RAM16SDP IP Customization Configuration



3.7.3 ROM16

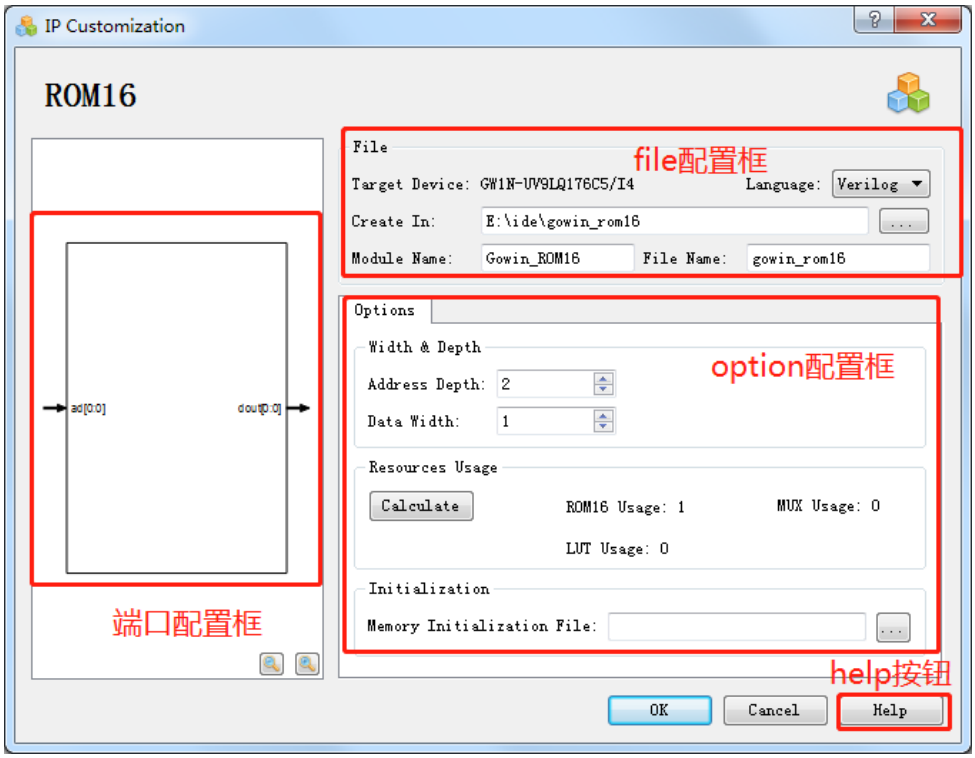
ROM16 is a read only memory. The maximum capacity of SSRAM varies according to the type of chip. Click "ROM16" on the IP Core Generator page. A brief introduction to the ROM16 will be displayed on the right of the screen, as shown in Figure 3-181.

Figure 3-181 ROM16 Summary Information



Double-click the "ROM16" to open the "IP Customization" window. This includes "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button, as shown in Figure 3-182.

Figure 3-182 IP Customization of ROM16



1. File Configuration

File configuration includes the basic information related to the ROM16 instantiation file, as shown in Figure 3-182.

The ROM16 file configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP> File Configuration.

2. Options Configuration

The Options configuration box is used to configure read only memory by users, as shown in Figure 3-182.

ROM16 Options configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP> Options Configuration.

3. Ports Configuration Diagram

The ports configuration diagram displays the current IP Core configuration result. Input/Output bit-width updates in real time based on the "Options" configuration, as shown in Figure 3-182.

"Address Depth" affects the bit-width of ad; "Data Width" affects the bit-width of dout.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-183.

Figure 3-183 Help

ROM16

Information

Type:	ROM16
Vendor:	GOWIN Semiconductor
Summary:	Shadow memory can be configured as ROM mode. In this mode, the shadow memory supports different read data widths, and it can be implemented by primitive ROM16. It can also be initialized by a memory initialization file.

Options

Option	Description
Width & Depth	**Address Depth** - Set the size of the address depth.
	Data Width - Set the size of the data width.
Resources Usage	**Calculate** - Calculate the resource usage in the design and display results below.
	Block Ram Usage - Display the number of ROM16 used.
	LUT Usage - Display the number of LUT used.
	MUX Usage - Display the number of MUX used.
Initialization	**Memory Initialization File** - Set the memory initialization file (.mi) path.

The "Help" page contains a general description of the IP Core, and a brief introduction to the "Options".

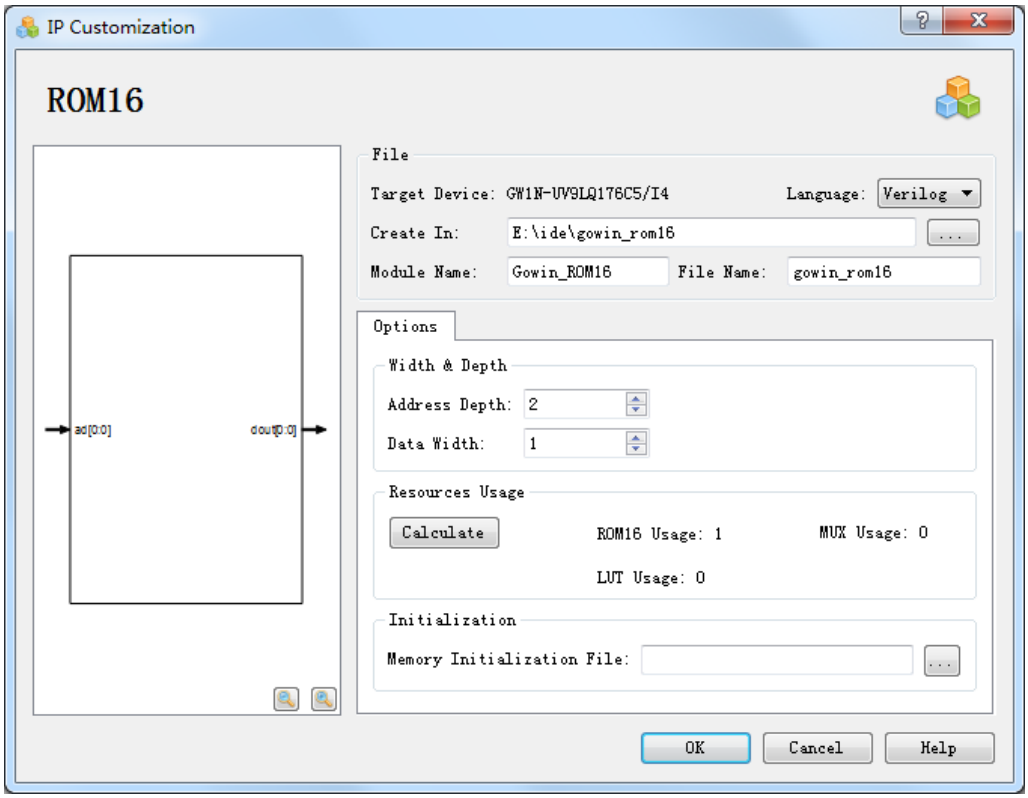
IP Generation Files

As shown in Figure 3-184, after customizing the IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate Gowin Primitive ROM16 design file "gowin_rom16.v";
- The instantiation template file for the IP design file "gowin_rom16_tmp.v";
- The configuration file for the Gowin Primitive ROM16 instantiation "gowin_rom16.ipc".

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

Figure 3-184 Configured IP Customization



ROM Design File Instantiation

The design file for the Gowin Primitive ROM16 instantiation is a complete Verilog module. ROM16 instantiation is generated according to the ROM16 configuration that is displayed in the "IP Customization" window, as shown in Figure 3-185.

Note!

Dout data width of the generated ROM16 instantiation is consistent with that of the ROM16 configured in the "IP Customization" window.

Figure 3-185 ROM16 Design File Instantiation

```
module Gowin_ROM16 (dout, ad);

output [3:0] dout;
input [3:0] ad;

ROM16 ssram_rom_0 (
    .DO(dout[0]),
    .AD(ad[3:0])
);

defparam ssram_rom_0.INIT_0 = 16'h0000;

ROM16 ssram_rom_1 (
    .DO(dout[1]),
    .AD(ad[3:0])
);

defparam ssram_rom_1.INIT_0 = 16'h0000;

ROM16 ssram_rom_2 (
    .DO(dout[2]),
    .AD(ad[3:0])
);

defparam ssram_rom_2.INIT_0 = 16'h0000;

ROM16 ssram_rom_3 (
    .DO(dout[3]),
    .AD(ad[3:0])
);

defparam ssram_rom_3.INIT_0 = 16'h0000;

endmodule //Gowin_ROM16
```

Instantiation template file for the IP design file

For practical use, the IP Core Generator generates the template file while generating ROM16 design file instantiation, as shown in Figure 3-186.

Figure 3-186 Instantiation Template File for the IP Design File

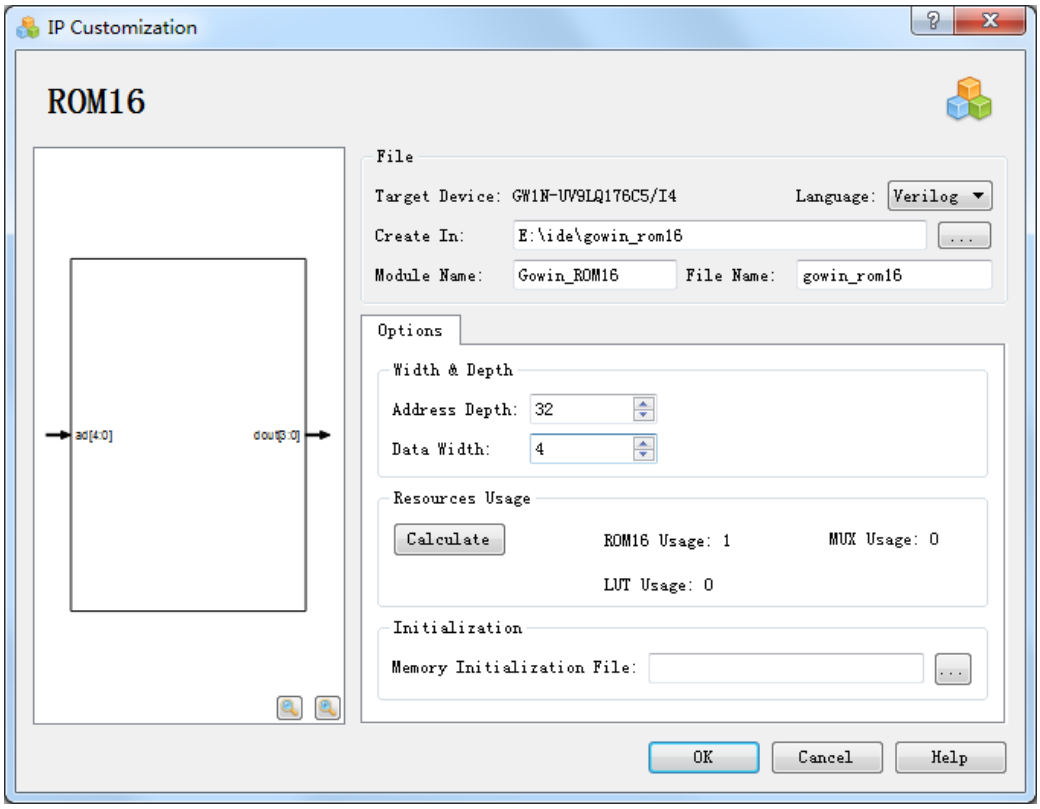
```
Gowin_ROM16 your_instance_name(
    .dout(dout_o), //output [3:0] dout
    .ad(ad_i) //input [3:0] ad
);
```

ROM16 Generation Example

Generate a specific ROM16 IP as follows:
address width and data width: 32 x 4;
Take the GW1N-UV9LQ176C6/I5 device for instance, the configuration interface is as shown in Figure 3-187. Select a memory initialization file for the module as required, and then click "OK" to generate the customized ROM16 IP design files.

The directory where the ROM16 IP design file is generated is the path set in "Create In".

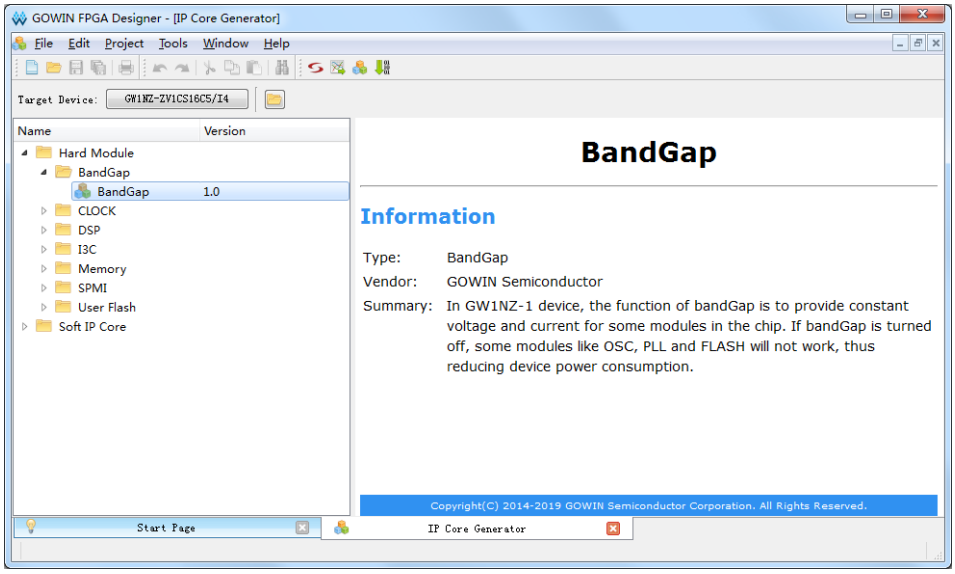
Figure 3-187 ROM16 IP Customization Configuration



3.8 BandGap

BandGap provides a constant voltage and current for certain modules in the chip. If BandGap is closed, certain modules such as OSC, PLL, and FLASH will not work, reducing device power consumption. Click "BandGap" on the IP Core Generator interface. A brief introduction to the BandGap will be displayed on the right of the screen, as shown in Figure 3-188.

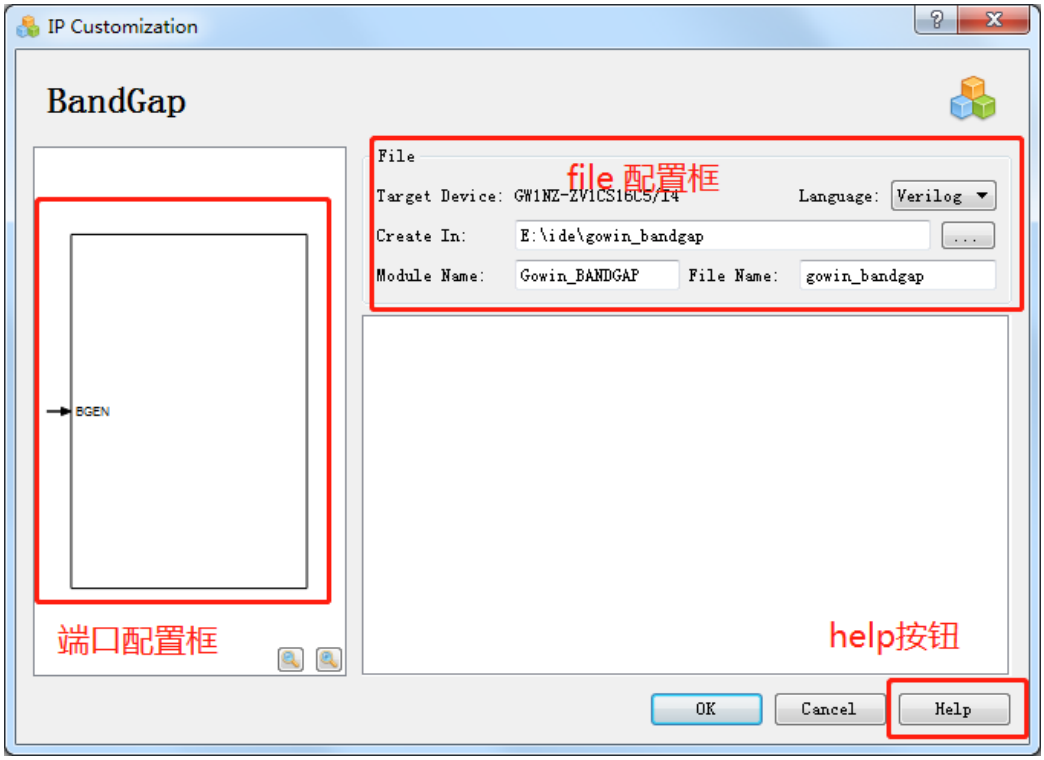
Figure 3-188 Bandgap Summary Information



Double click the "BandGap" to open the "IP Customization" window.

This includes the "File" configuration, "Options" configuration, port configuration diagram, and the "Help" button, as shown in Figure 3-189.

Figure 3-189 IP Customization of BandGap



1. File Configuration

The file configuration includes the basic information related to the BandGap instantiation file, as shown in Figure 3-189.

The BandGap file configuration is similar to that of SP. For the detailed configuration, please refer to 3.1Block Memory>3.1.1SP > File Configuration.

Note!

Only GW1NZ-1 supports BandGap at present. If you select the other devices as the target device, the BandGap will be grey, and no corresponding IP can be generated.

2. Ports Configuration Diagram

Ports configuration diagram displays the current IP Core configuration result, as shown in Figure 3-189.

3. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-190.

Figure 3-190 Help

BandGap

Information

Type:	BandGap
Vendor:	GOWIN Semiconductor
Summary:	In GW1NZ-1 device, the function of bandGap is to provide constant voltage and current for some modules in the chip. If bandGap is turned off, some modules like OSC, PLL and FLASH will not work, thus reducing device power consumption.

The "Help" page contains a general description of the IP Core.

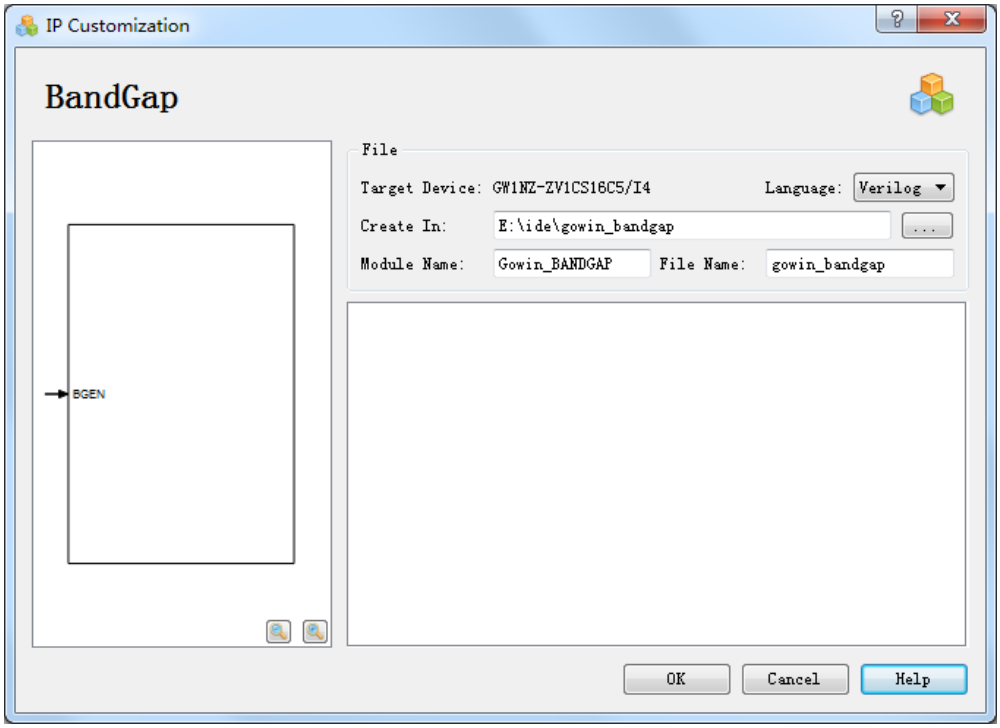
IP Generation Files

As shown in Figure 3-191, after customizing the IP, click "OK" to generate three files that are named after the "File Name" specified in the File configuration:

- Instantiate Gowin Primitive BandGap design file "gowin_bandgap.v";
- The instantiation template file for the IP design file "gowin_ bandgap _tmp.v";
- The configuration files for the Primitive BandGap instantiation "gowin_ bandgap.ipc".

Taking verilog for instance, the following sections introduce the generated files.

Figure 3-191 Configured IP Customization



BandGap Design File Instantiation

The design file for the Gowin Primitive BandGap instantiation is a complete Verilog module. BandGap instantiation is generated according to the BandGap configuration that is displayed in the "IP Customization" window, as shown in Figure 3-192.

Figure 3-192 BandGap Design File Instantiation

```
module Gowin_BANDGAP (bgen);

input bgen;

BANDGAP bandGap_inst (
    .BGEN(bgen)
);

endmodule //Gowin_BANDGAP

module BANDGAP ( BGEN )/*synthesis syn_black_box black_box_pad_pin = "BGEN" syn_noprune = 1*/;

input BGEN;

endmodule
```

Instantiation Template File for the IP Design File

For the practical use, the IP Core Generator generates the template file while generating the BandGap design file instantiation, as shown in Figure 3-193.

Figure 3-193 Instantiation Template File for the IP Design File

```
Gowin_BANDGAP your_instance_name(
    .bgen(bgen_i) //input bgen
);
```


GOWIN
PROGRAMMING FOR THE FUTURE