Gowin IP Core Generator

User Guide

SUG284-1.5E, 11/23/2018

Disclaimer

GOWINSEMI®, LittleBee®, Arora™, and the GOWINSEMI logos are trademarks of GOWINSEMI and are registered in China, the U.S. Patent and Trademark Office, and other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders, as described at www.gowinsemi.com.cn. GOWINSEMI assumes no liability and provides no warranty (either expressed or implied) and is not responsible for any damage incurred to your hardware, software, data, or property resulting from usage of the materials or intellectual property except as outlined in the GOWINSEMI Terms and Conditions of Sale. All information in this document should be treated as preliminary. GOWINSEMI may make changes to this document at any time without prior notice. Anyone relying on this documentation should contact GOWINSEMI for the current documentation and errata.

Revision History

Date	Version	Description
09/05/2018	1.3E	Initial version published.
11/02/2018	1.4E	● GW1NZ-1, GW1NSR-2C supported; ● The hardcore of I3C, SPMI added.
11/23/2018	1.5E	● GW1NSR-2 supported; ● GW1N-6ES, GW1N-9ES, and GW1NR-9ES deleted.

Contents

List of Figures

List of Tables

1 About This Guide

1.1 Purpose

This manual provides an overview of how to use the IP Core Generator that forms part of the Gowin Yunyuan software. This generator is designed to help users create complex designs with a more convenient way. Gowin Yunyuan software supports both Linux and Windows operating systems. The software screenshots and the supported products listed in this guide are based on the version Windows 1.8.1 Beta. 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:

1. GW1N series of FPGA products: GW1N-1, GW1N-2, GW1N-2B, GW1N-4, GW1N-4B, GW1N-6, GW1N-9
2. GW1NR series of FPGA products: GW1NR-4, GW1NR-4B, GW1NR-9
3. GW1NS series of FPGA products: GW1NS-2, GW1NS-2C;
4. GW2A series of FPGA products: GW2A-55, GW2A-18;
5. GW2AR series of FPGA products: GW2AR-18;
6. GW1NZ series of FPGA products: GW1NZ-1;
7. GW1NSR series of FPGA products: GW1NSR-2C, GW1NSR-2.

1.3 Related Documents

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

1. GW1N series of Products Data Sheet
2. GW1NR series of FPGA Products Data Sheet
3. GW1NS series of FPGA Products Data Sheet
4. GW2A series of FPGA Products Data Sheet
5. GW2AR series of FPGA Products Data Sheet
6. GW1NZ series of Products Data Sheet
7. GW1NSR series of FPGA Products Data Sheet

1.4 Abbreviations and Terminology

Table 1-1 shows the abbreviations and terminology that is employed in this manual.

Table 1-1 Abbreviations and Terminology

Abbreviations and Terminology	Name
FPGA	Field Programmable Gate Array
IDE	Integrated Development Environment
IP core	Intellectual Property Core
DP/DPX9	Dual Port
SP/SPX9	Single Port
SDP/SDPX9	Semi Dual Port
ROM/ROMX9	Read Only Memory
PADD	Pre-adder
MULT	Multiplier
PLL	Phase-locked Loop
DLL	Delay-locked Loop
OSC	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 using the information provided below.

Website: www.gowinsemi.com

E-mail: support@gowinsemi.com

2 Introduction

The IP Core Generator that is available in the Gowin software is predominantly used to generate instantiation components and IPs that users can call to implement the required functions. They provide users with a convenient way to create complex designs. The IP Core Generator includes the Hard modules associated with primitives and the Soft IP Cores associated with the reference designs.

3 IP Core Generation

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

This page includes two parts:

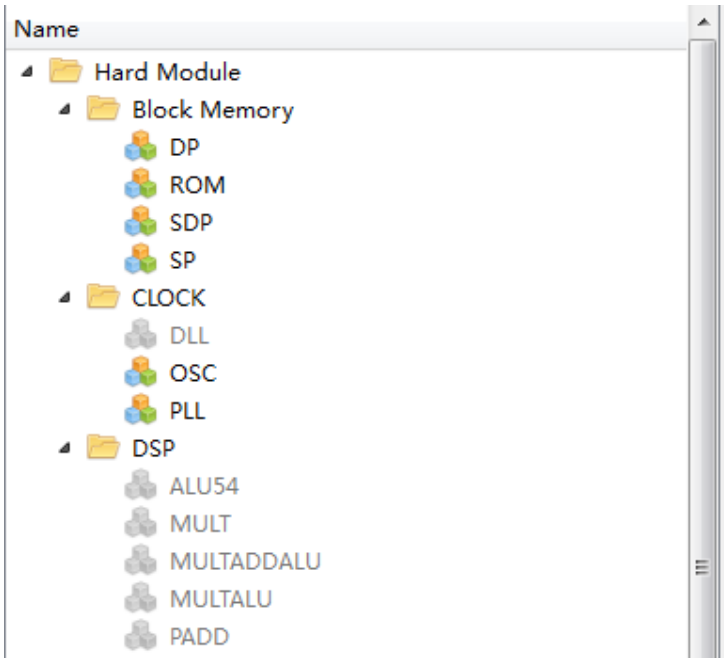
- The modules associated with primitives;
- The IP Cores associated with the reference designs.

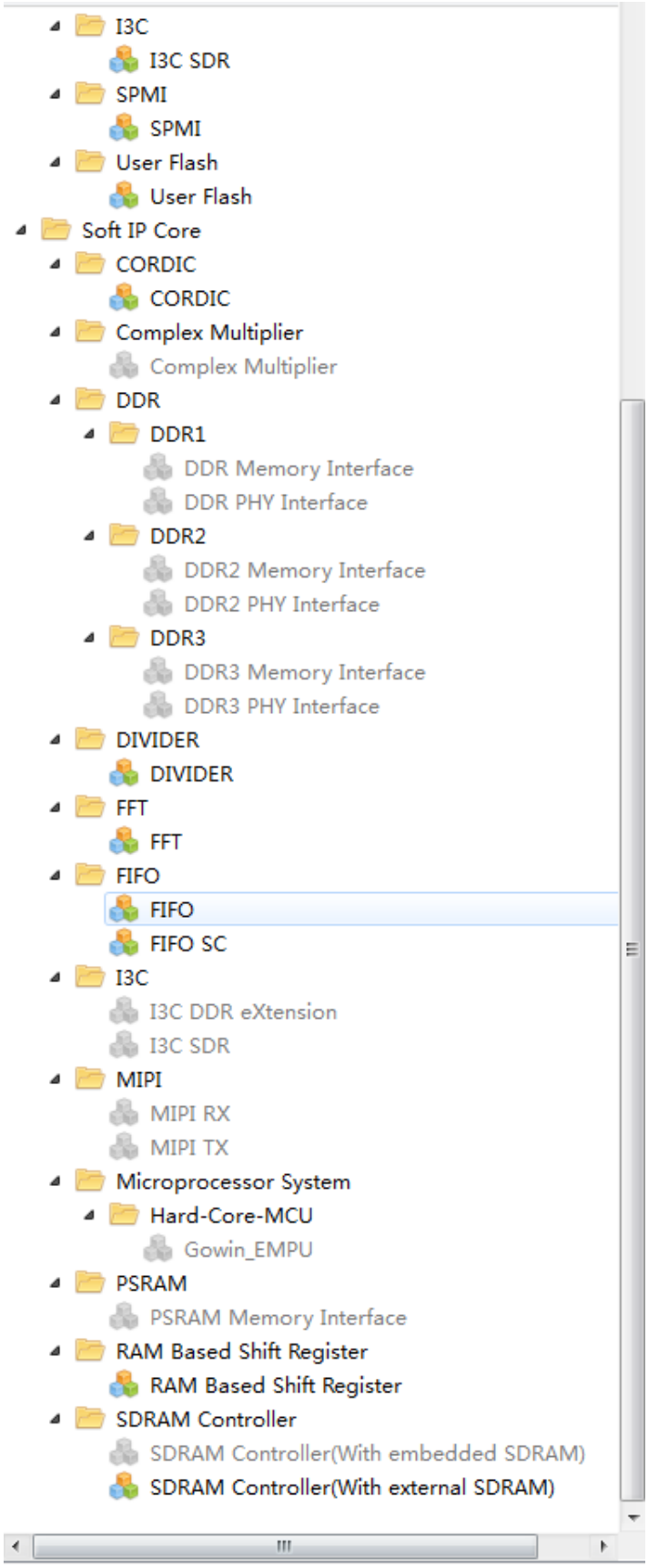
The Hard modules include Block Memory, CLOCK, DSP, I3C, SPMI, and User Flash, etc;

The Soft IP Cores include CORDIC, Complex Multiplier, DDR, DIVIDER, FFT, FIFO, I3C, MIPI, PSRAM, RAM Based Shift Register, SDRAM Controller, etc.

This manual mainly provides an overview of how to use the Hard modules.

Figure 3-1 IP Core Generator Page





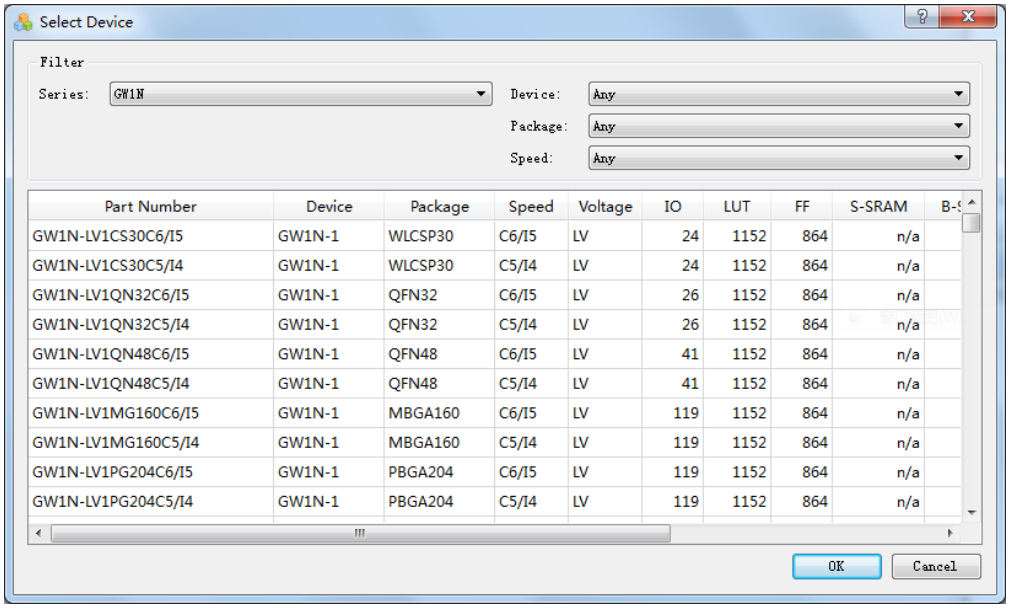
There are two icons at the top of the core generator page. One is for the target device, and  is used to open the IP core configuration files.

- "Target Device": Device to which the IP is to be targeted.

  Click the right display box "Target Device" to select the device that needs to be configured, as shown in Figure 3-2.

  The name of the device selected will appear in the "Target Device" display box (IP Customization> File).

Figure 3-2 Select Device



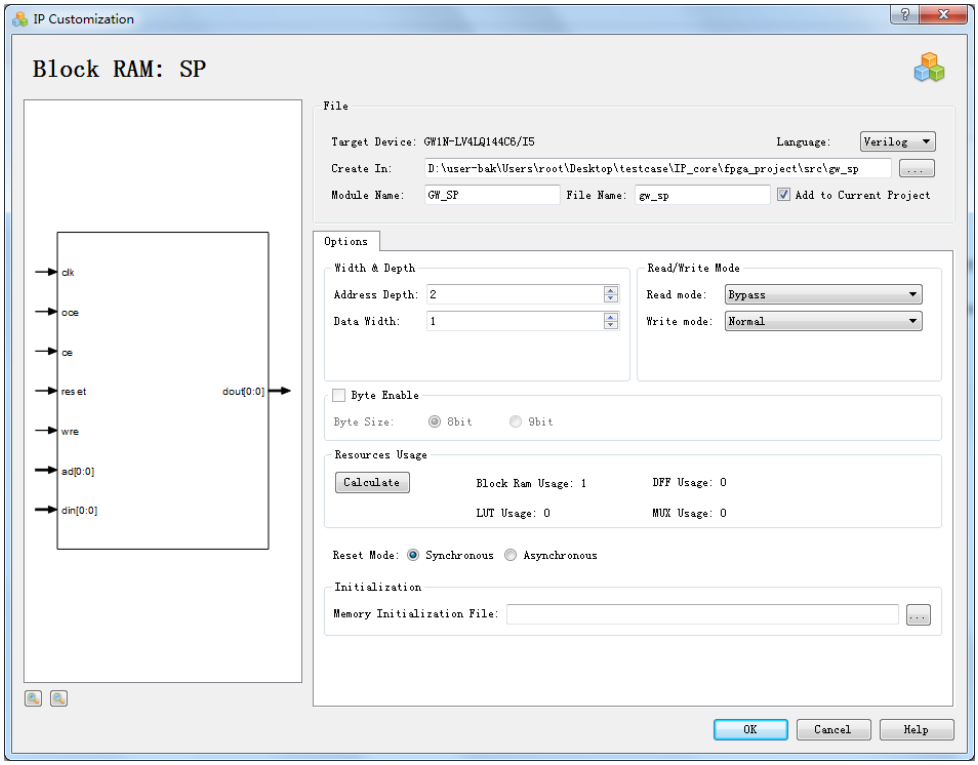
Part Number	Device	Package	Speed	Voltage	IO	LUT	FF	S-SRAM
GW1N-LV1CS30C6/I5	GW1N-1	WLCSP30	C6/I5	LV	24	1152	864	n/a
GW1N-LV1CS30C5/I4	GW1N-1	WLCSP30	C5/I4	LV	24	1152	864	n/a
GW1N-LV1QN32C6/I5	GW1N-1	QFN32	C6/I5	LV	26	1152	864	n/a
GW1N-LV1QN32C5/I4	GW1N-1	QFN32	C5/I4	LV	26	1152	864	n/a
GW1N-LV1QN48C6/I5	GW1N-1	QFN48	C6/I5	LV	41	1152	864	n/a
GW1N-LV1QN48C5/I4	GW1N-1	QFN48	C5/I4	LV	41	1152	864	n/a
GW1N-LV1MG160C6/I5	GW1N-1	MBGA160	C6/I5	LV	119	1152	864	n/a
GW1N-LV1MG160C5/I4	GW1N-1	MBGA160	C5/I4	LV	119	1152	864	n/a
GW1N-LV1PG204C6/I5	GW1N-1	PBGA204	C6/I5	LV	119	1152	864	n/a
GW1N-LV1PG204C5/I4	GW1N-1	PBGA204	C5/I4	LV	119	1152	864	n/a

After the user has selected the device, the IP Core Generator will list the the modules and IP Cores that are supported.

- The IP Cores or modules that are displayed in black are supported. Double click on the name of the item to open the configuration window;
- The IP Cores or modules that are displayed in grey are not supported. As shown in Figure 3-1, GW1N-4 does not support the DDR memory interface.

"  " 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 “IP Customization” window will open for reconfiguration, as shown in Figure 3-3.

Figure 3-3 IP Customization



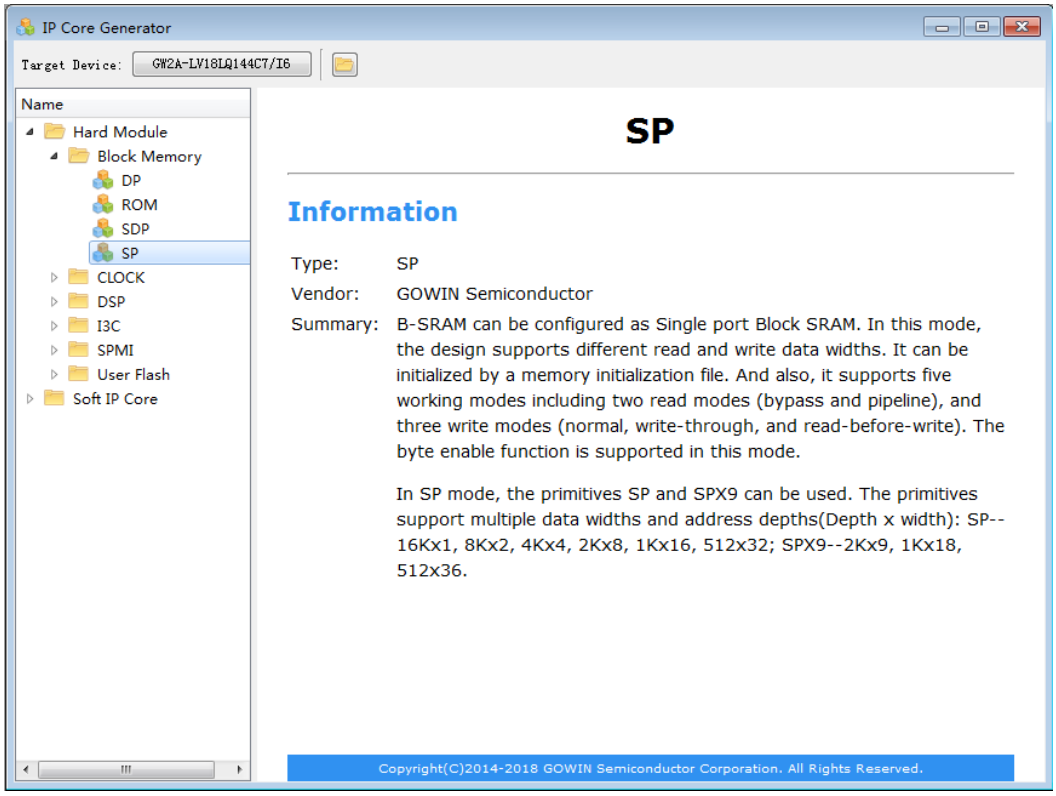
3.1 Block Memory

Currently, Block Memory (BSRAM) can be used to generate the following modules: Single Port (SP), Semi-dual Port (SDP), Dual Port (DP), and Read Only Memory (ROM).

3.1.1 SP

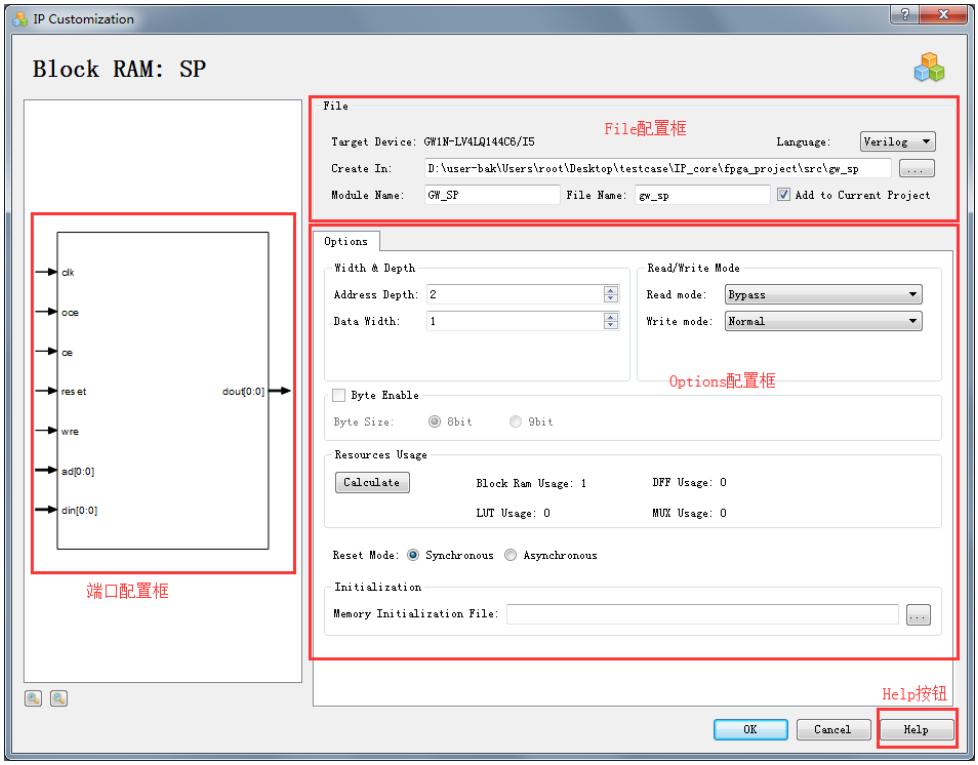
SP is a Single Port Block Memory that can be implemented by SP and SPX9. The maximum capacity depends on the chip type. Click "SP" on the IP Core Generator page. A brief introduction to the SP will be displayed on the right-hand side of the screen, , as shown in Figure3-4.

Figure3-4 SP Summary



Double click on “SP”, and the “IP Customization” window will open, as shown in Figure 3-5. This window includes file configuration, options configuration, port configuration diagram, and the “Help” button.

Figure 3-5 SP-IP Customization



1. File Configuration

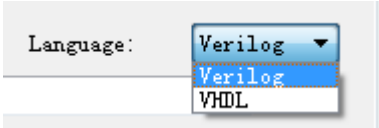
The file configuration mainly includes the basic information related to the SP instantiation file.

- Target Device: Display the configured device info.;
- Language: Hardware description language used to generate the IP Core files. Click the right drop-down list box to select the target language, including Verilog and VHDL, as shown in Figure 3-6.
- Module Name: Name given to the generated IP Core module. Enter the module name in the text box on the right side.
- File Name: Name given to the generated IP Core. Enter the file name in the text box that is displayed on the right side.
- Create In: Path to the directory in which the generated IP files will be stored. Enter the target directory in the box on the right side or select the target directory using the option button that appears next to the text button.
- Add to Current Project: Decide whether to save the IP Core file to the current project.
  - If the check box is not selected, the IP Core file will only be saved to the target directory.
  - If the check box is selected, the IP Core file will be both saved to the target directory and displayed in the design window of the currently opened project.

Note!

- "Add to Current Project" is checked by default;
- IP Core generates an RTL file, so if the current project is a Post-Synthesis Project, “Add to Current Project” is grey and non-configurable.

Figure 3-6 Language Drop-down List Box



2. Options Configuration
   Options configuration mainly includes configuration information related to the MULTALU instantiation file, as shown in Figure 3-5.

- Width & Depth: Set SP Address Depth and Data Width. If the setting cannot be implemented by one module, multiple modules will be used to implement the current configuration.;
- Byte Enable: used to configure the use of Byte Enable. It can be selected when the Data Width is equal to or greater than 9, and byte size can be 8 bits or 9 bits;
- 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 (.mi) 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 3-5.
- "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 3-7.

Figure 3-7 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 bit width data read and 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 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 With<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.
	File Format - Set whether the format of the memory initialization file content is Binary or Hex.

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

5. IP Generation Files

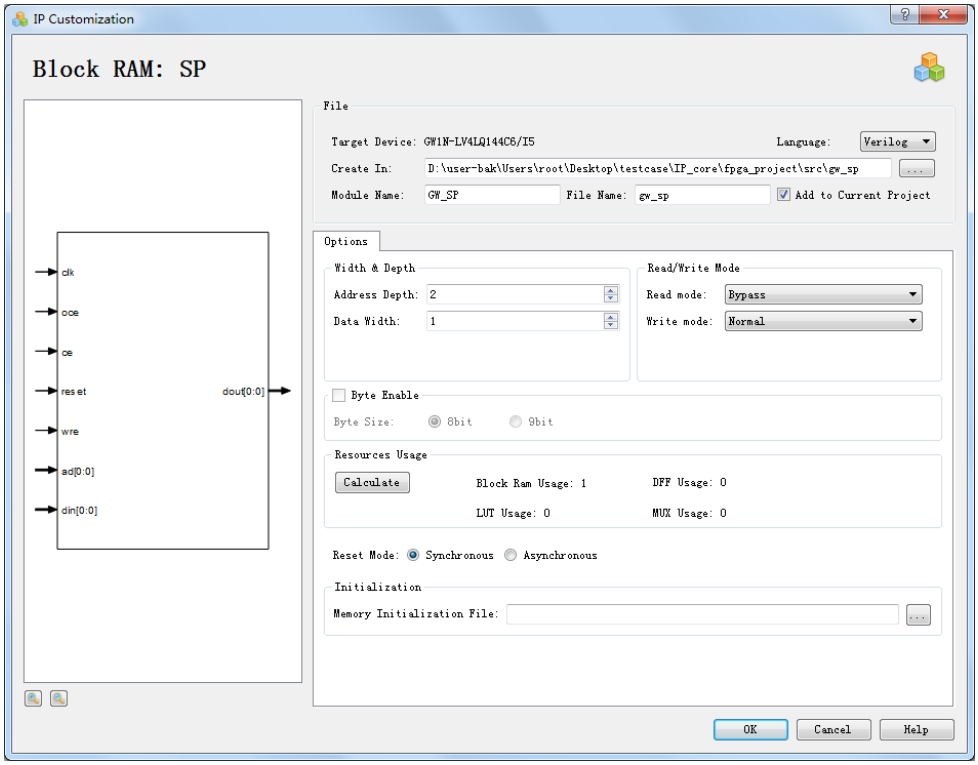
As shown in Figure 3-8, after customizing the IP, click “OK” to generate three files based on the "File Name" specified in the File configuration:

- Design file for the Gowin primitive SP instantiation "gw_sp.v";
- The instantiation template file for the IP design file "gw_sp_tmp.v";
- The configuration files for the Gowin Primitive SP instantiation "gw_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 3-8 IP Customization



6. SP Design File Instantiation

The design file for the Gowin primitive SP instantiation is a complete Verilog module. SP instantiation is generated according to the SP configuration provided in the "IP Customization" window, as shown in Figure 3-9.

Figure 3-9 SP Design File Instantiation

```
module GW_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 //GW_SP
```

7. The Instantiation Template File for the IP Design File

For efficiency purposes, the IP Core Generator generates the template file while generating the SP design file instantiation, as shown in Figure 3-10.

Figure 3-10 Instantiation Template File for the IP Design File

```
GW_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
);
```

8. SP Generation Example

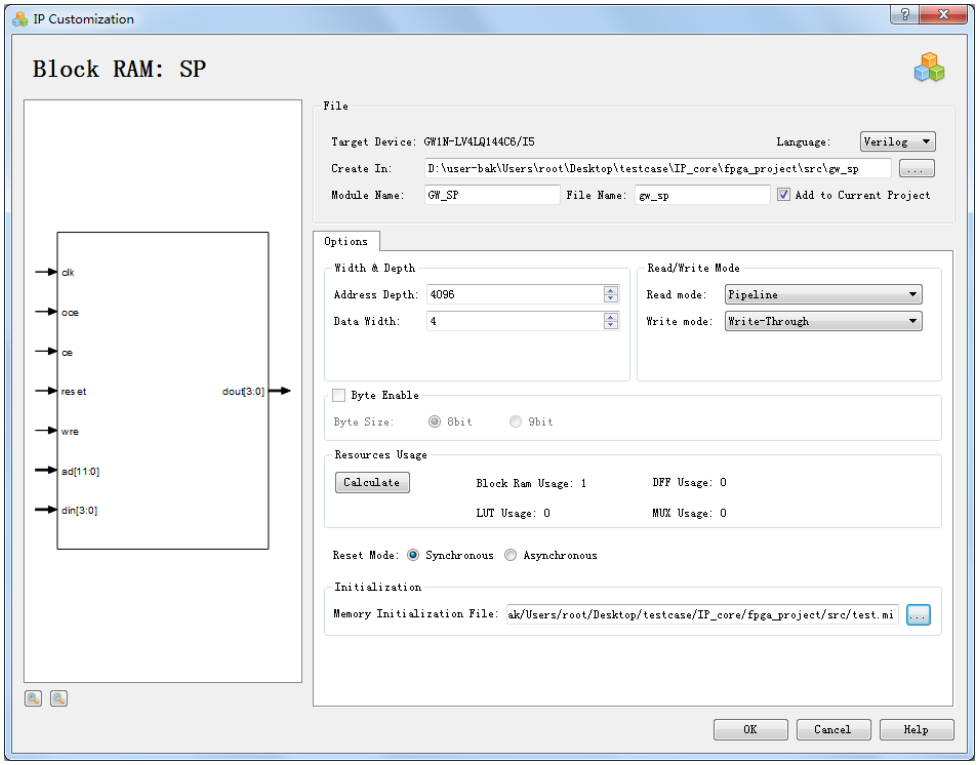
Generate a specific SP IP as follows:

- Width and Depth: 4096 x 4;
- Pipeline read mode;
- Write-through mode;
- Synchronous.

Take the GW1N-4-LQFP144 device for instance; the configuration page is as shown in Figure 3-11. Select a memory initialization file (.mi) for the module as required, and then click "OK" to generate the customized SP IP design files.

The generated IP files are stored in the directory set in the "Create in" text box. If “Add to Current Project” is checked, the generated IP design files will be automatically added to the project.

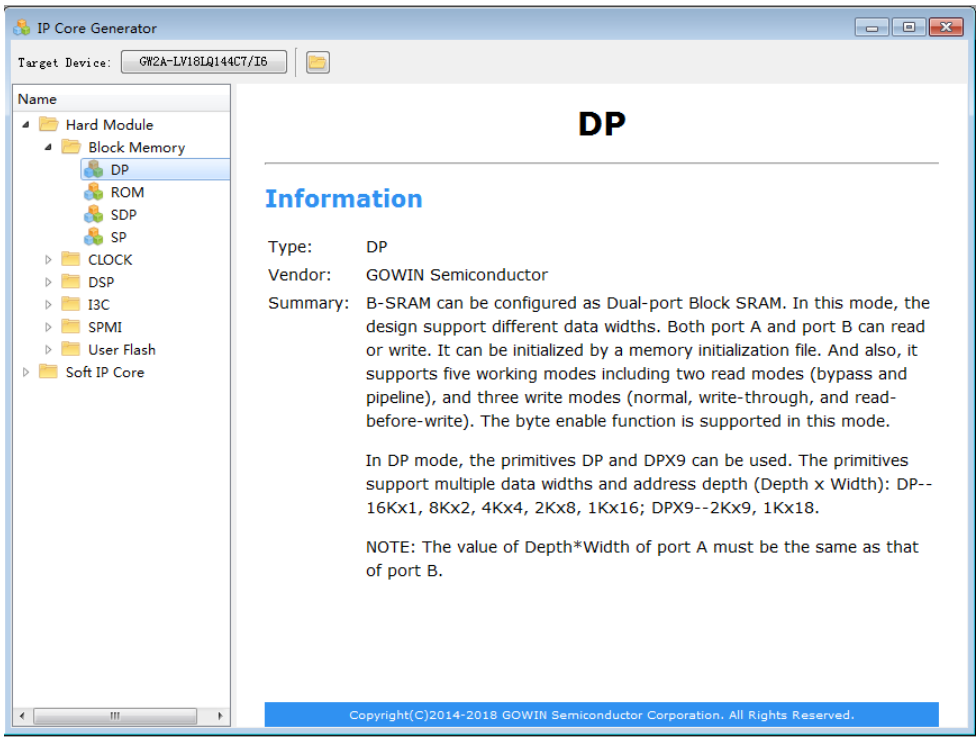
Figure 3-11 SP - IP Customization



3.1.2 DP

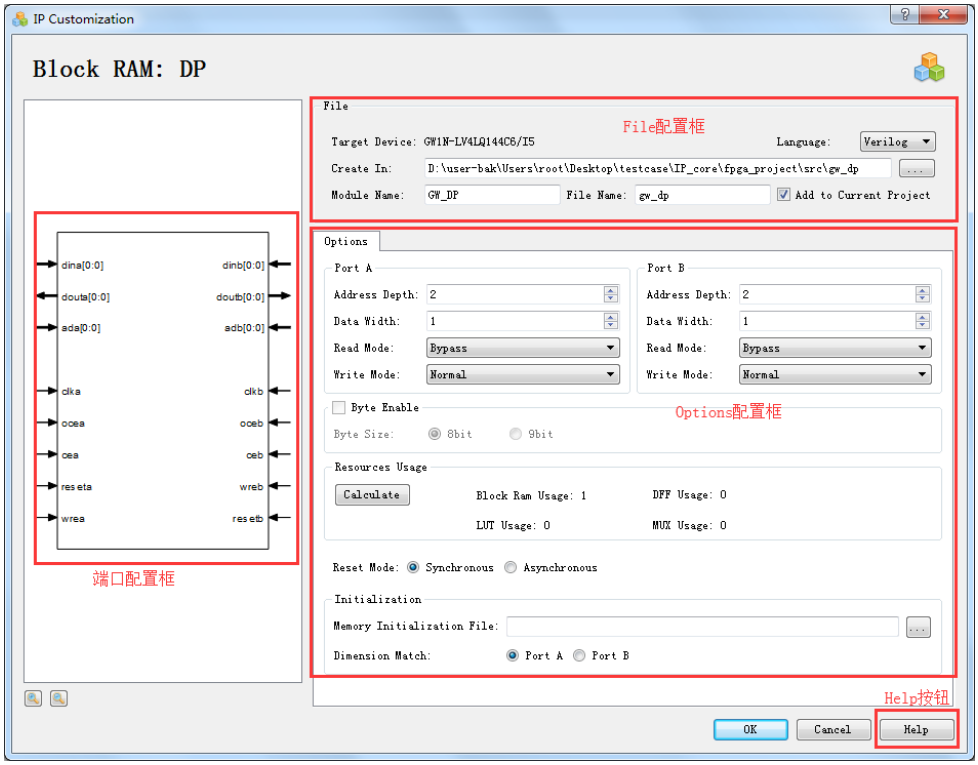
DP is the Dual Port Block Memory, which can be implemented by DP and DPX9. The maximum capacity depends on the chip type. Click "DP" on the IP Core Generator page. A brief introduction to the DP will be displayed on the right of the screen, as shown in Figure 3-12.

Figure 3-12 DP Summary Information



Double-click on the "DP" to open the "IP Customization" window. This displays the file configuration, options configuration, port configuration diagram, and the “Help” button, as shown in Figure 3-13.

Figure 3-13 DP – IP Customization



1. File Configuration

The File configuration window mainly includes the basic information related to the DP instantiation file, as shown in Figure 3-13.

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

Options configuration mainly includes the configuration information related to the DP instantiation file, as shown in Figure 3-13.

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 following before configuring the DP:

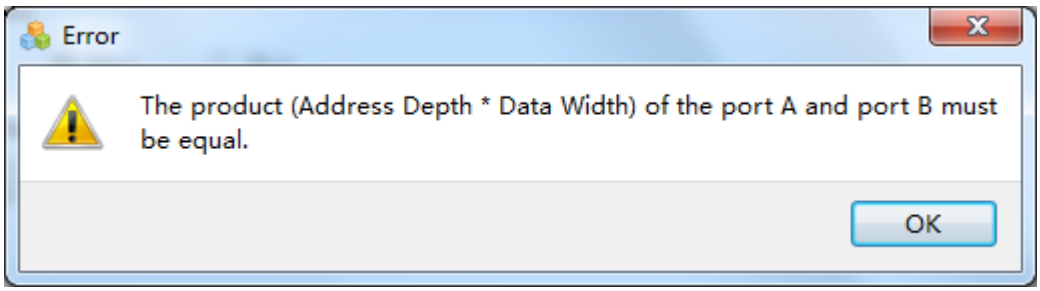
- The address depth, data width, and read/write mode of DP Port A and Port B can be configured independently.
- The address depth and data width of DP Port A and Port B must be equal because Port A and Port B read from or write to the same memory.
- The data width in the Memory initialization File should 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 3-14.

- 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 3-14 DP 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 3-13.
- "Address Depth" determines the bit-width of ada and adb; "Data Width" determines the bit-width of dia/doa and dib/dob.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-15.

Figure 3-15 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 bit width data. 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 With<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.
	File Format - Set whether the format of the memory initialization file content is Binary or Hex.

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

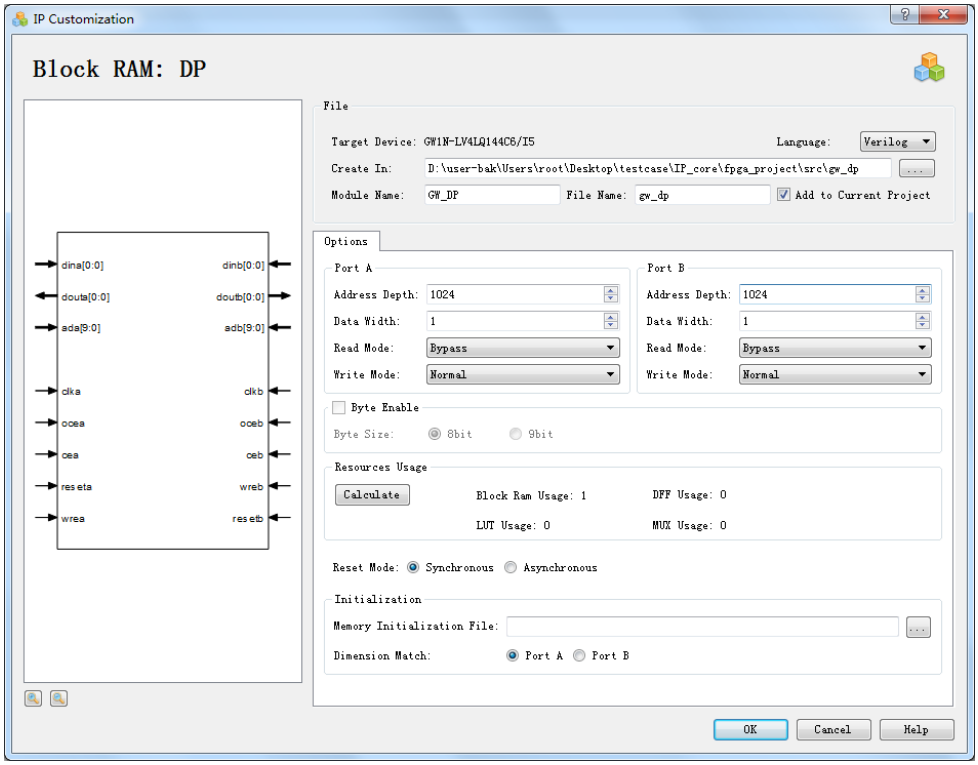
IP Generation Files

As shown in Figure 3-16, after customizing the IP, click “OK” to generate three files that are named according to the "File Name" specified in the file configuration:

- The design file for the Gowin Primitive DP instantiation "gw_dp.v";
- The instantiation template file for the IP design file "gw_dp_tmp.v";
- The configuration files for the Gowin Primitive DP instantiation "gw_dp.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 3-16 DP - IP Customization



DP Design File Instantiation

The design file of Gowin Primitive DP instantiation is a complete verilog module. DP instantiation is generated according to the DP configuration specified in the "IP Customization" window, as shown in Figure 3-17.

Figure 3-17 DP Design File Instantiation

```
module GW_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 [9:0] ada;
input [0:0] dina;
input [9:0] 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,ada[9:0]}),
    .DIA(dina[0]),
    .ADB({gw_gnd,gw_gnd,gw_gnd,gw_gnd,adb[9:0]}),
    .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 //GW_DP
```

Instantiation Template File for the IP Design File

For efficiency purposes, the IP Core Generator generates the template file while generating the DP design file instantiation, as shown in Figure 3-18.

Figure 3-18 Instantiation Template File for the IP Design File

```
GW_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

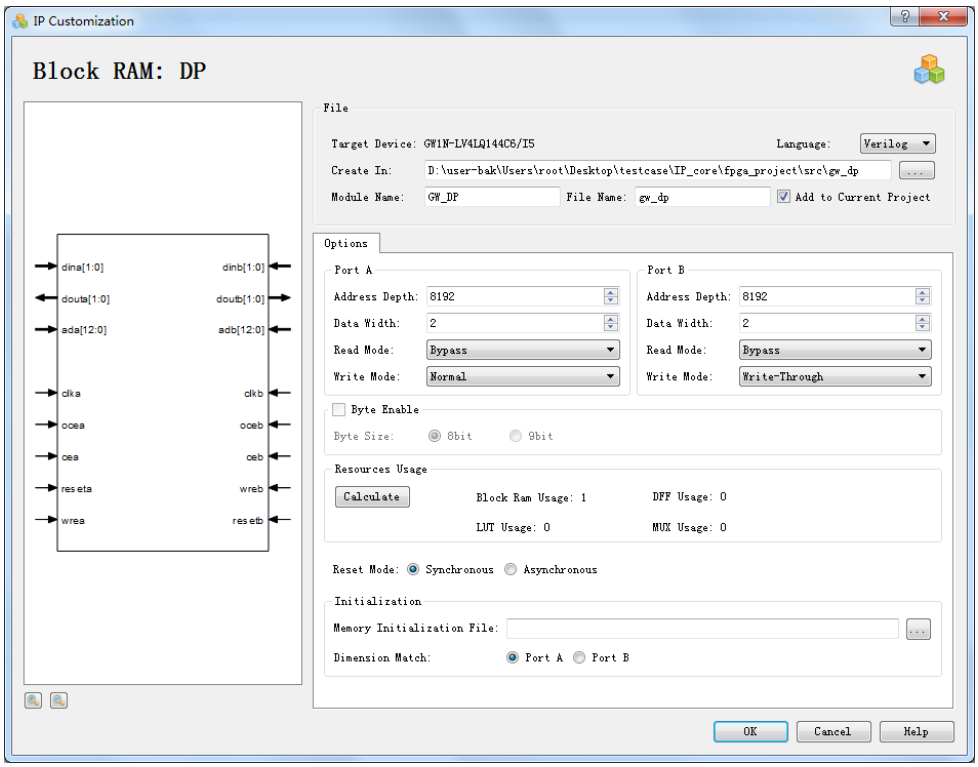
Generate a specific DP IP as follows:

- Width and Depth: 8192 x 2;
- Bypass read mode and write-through write mode;
- Synchronous.

Take the GW1N-4-LQFP144 device for instance; the configuration page is as shown in Figure 3-19. Select a memory initialization file (.mi) for the module as required, and then click "OK" to generate the customized DP IP design files.

The generated IP files are stored in the directory specified in the directory set in the "Create in" textbox. If “Add to Current Project” is checked, the generated IP design files will be automatically added to the project.

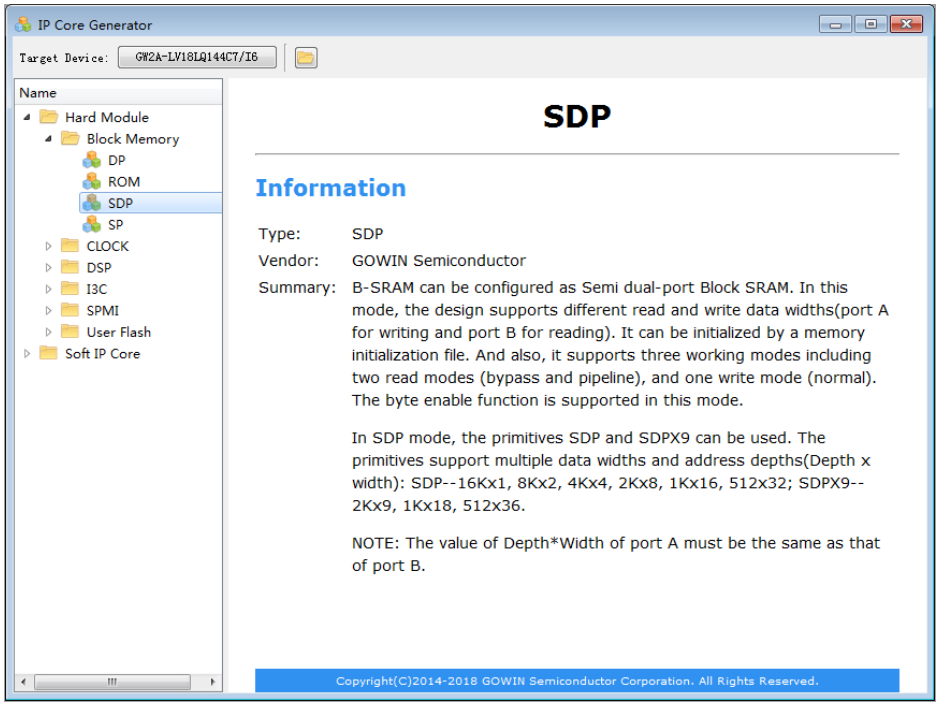
Figure 3-19 DP – IP Customization



3.1.3 SDP

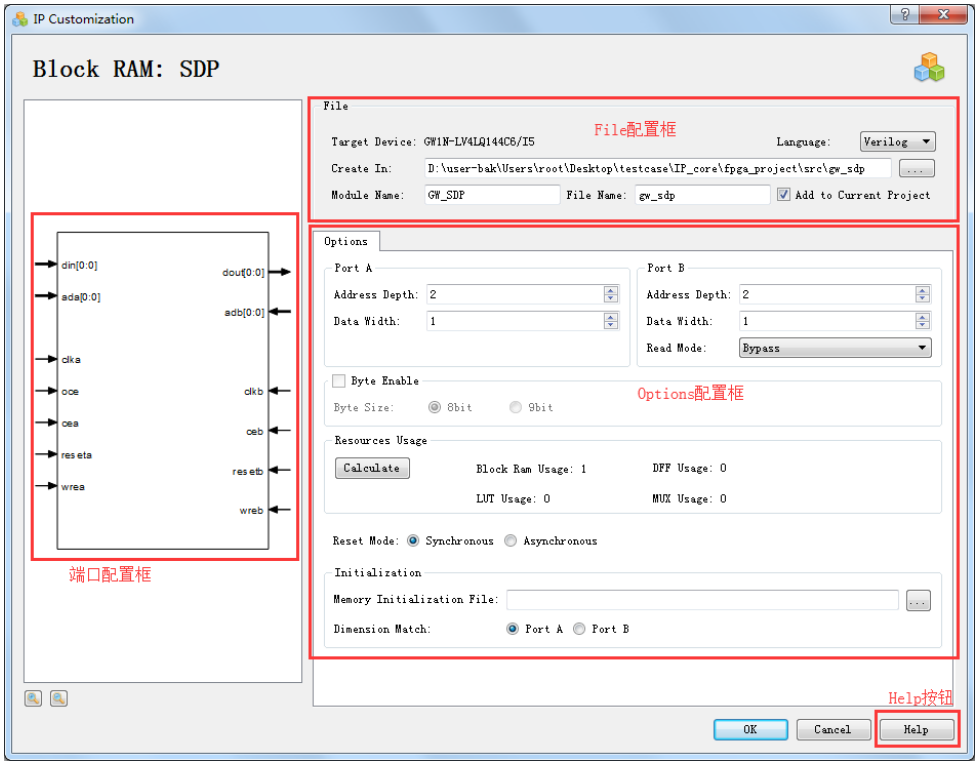
SDP is a Semi-dual Port Block Memory that can be implemented by SDP and SDPX9. The maximum capacity depends on the chip type. Click "SDP" on the IP Core Generator page. A brief introduction to the SDP will be displayed on the right of the screen, as shown in Figure 3-20.

Figure 3-20 SDP Summary Information



Double click on the "SDP" to open the "IP Customization" window, This displays the file configuration, options configuration, port configuration diagram, and the “Help” button, as shown in Figure 3-21.

Figure 3-21 SDP – IP Customization



1. File Configuration

File configuration mainly includes the basic information related to the SDP instantiation file, as shown in Figure 3-21.

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

Options configuration mainly includes the configuration information of SDP instantiation file, as shown in Figure 3-21.

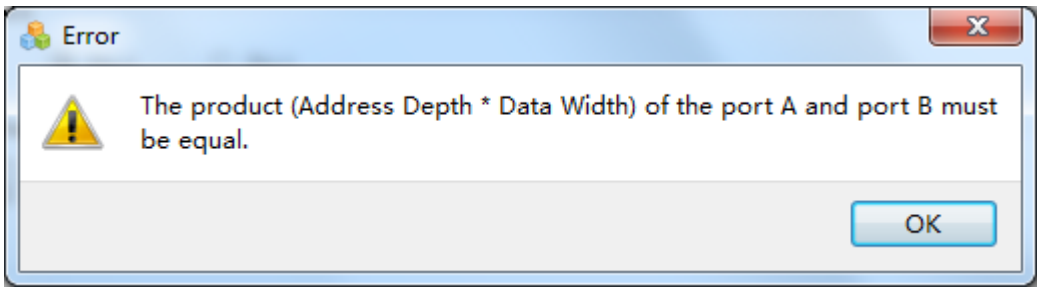
SDP options configuration is similar to that of SP. For the detailed configuration instructions, please refer to 3.1Block Memory > 3.1.1SP> Options Configuration.

Pay attention to the following before configuring the SDP:

- SDP only supports Port A write operation and Port B read operation; Port B Read Mode can 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 3-22 will pop up.
- The date width specified in the Memory initialization File should be consistent with the data width of the port selected by the "Dimension Match". If not, the Init value of the generated SDP instantiation is 0 by

default, and the following error message will be displayed:
Error (MG2105) : Initial values' width is unequal to user's width.

Figure 3-22 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 3-21.
- Port A "Address Depth" determines the bit-width of ada, and Port A "Data Width" determines the bit-width of din; Port B "Address Depth" determines the bit-width of adb, and Port B "Data Width" determines the bit-width of dout.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-23.

Figure 3-23 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 bit width data read and write (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 With<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.
	File Format - Set whether the format of the memory initialization file content is Binary or Hex.

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

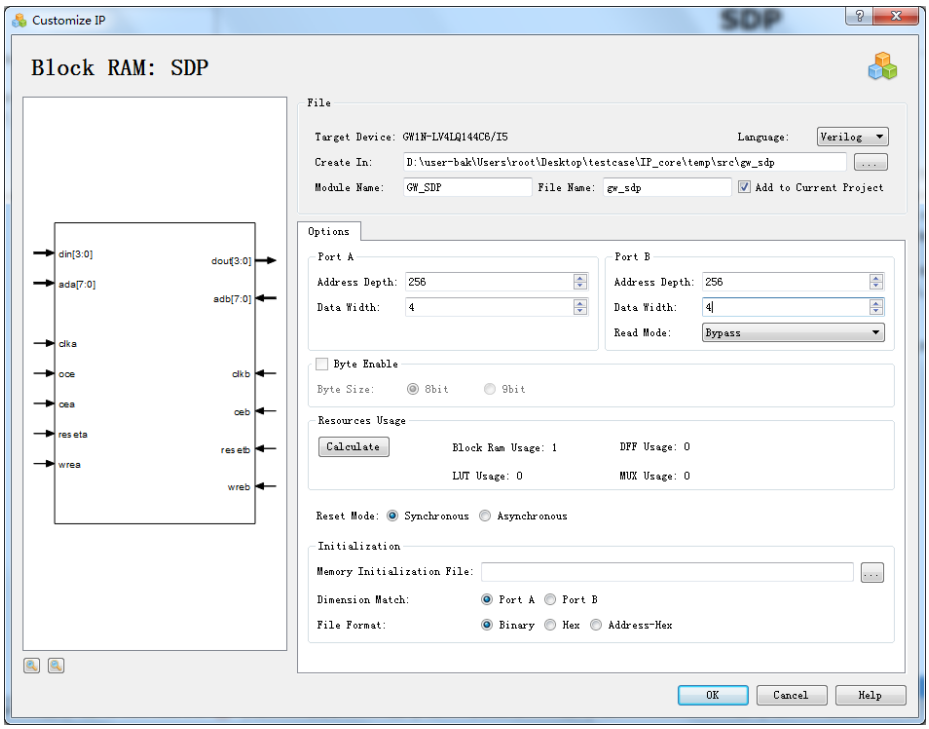
IP Generation Files

As shown in Figure 3-24, after customizing the IP, click “OK” to generate three files that are named according to "File Name" specified in the file configuration:

- The design file for the Gowin Primitive SDP instantiation "gw_sdp.v";
- The instantiation template file for the IP design file "gw_sdp_tmp.v";
- The configuration file for the Gowin Primitive SDP instantiation "gw_dp.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 3-24 IP Customization



SDP Design File Instantiation

The design file for the Gowin Primitive SDP instantiation is a complete verilog module. SDP instantiation is generated according to the SDP configuration specified in the "IP Customization" window, as shown in Figure 3-25.

Note!

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

Figure 3-25 SDP Design File Instantiation

```
module GW_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 //GW_SDP
```

Instantiation Template File for the IP Design File

For efficiency purposes, the IP Core Generator generates the template file while generating SDP design file instantiation, as shown in Figure 3-26.

Figure 3-26 Instantiation Template File for the IP Design File

```
GW_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

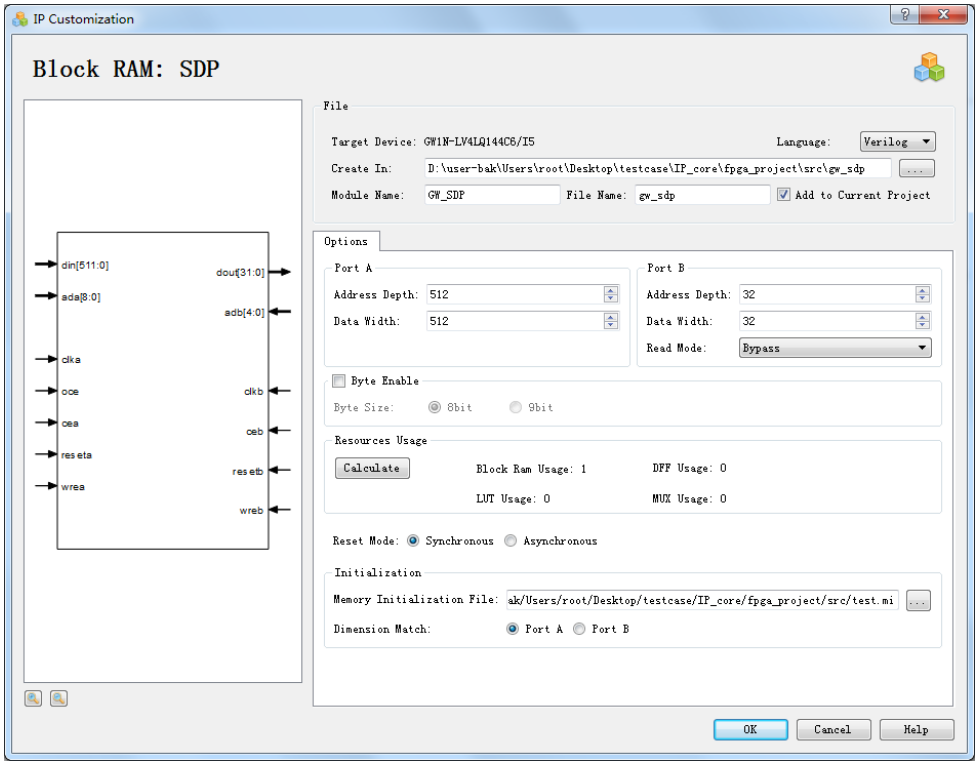
Generate a specific SDP IP as follows:

- Width and Depth: 512 x 32;
- Bypass read mode;
- Synchronous.

Take the GW1N-4-LQFP144 device for instance; the configuration page is as shown in Figure3-27. Select a memory initialization file (.mi) for the module as required, and then click "OK" to generate the customized SDP IP design files.

The generated IP files are stored in the directory specified in the directory set in the "Create in" textbox. If “Add to Current Project” is checked, the generated IP design files will be automatically added to the project.

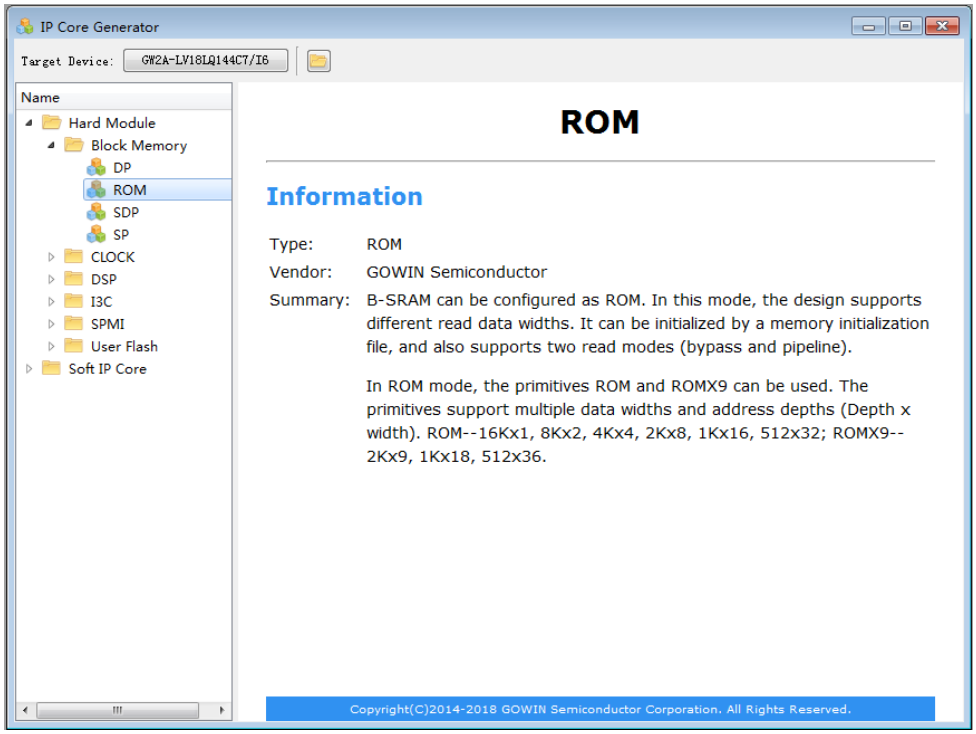
Figure3-27 SDP – IP Customization



3.1.4 ROM

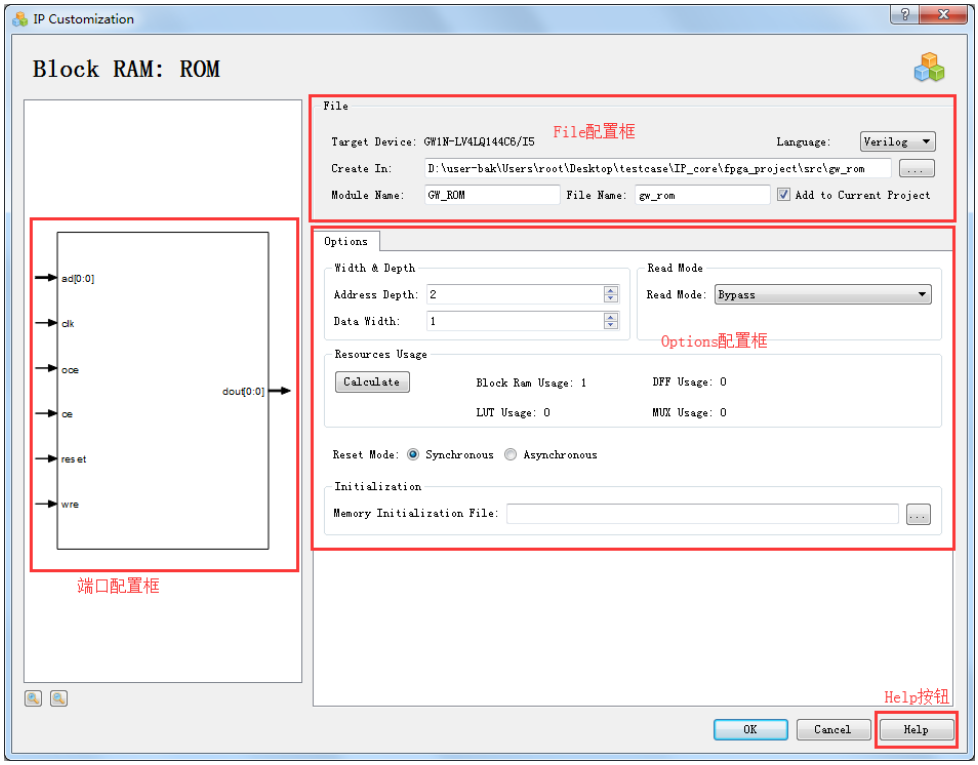
ROM is the Read Only Memory, which can be implemented by ROM and ROMX9. The maximum capacity depends on the chip type. Click the "ROM" on the IP Core Generator page. A brief introduction to the ROM will be displayed on the right of the screen, as shown in Figure 3-28.

Figure 3-28 ROM Information



On the IP Core Generator page, double click the "ROM" to open the "IP Customization" window. This displays File configuration, Options configuration, port configuration diagram, and the “Help” button, as shown in Figure 3-29.

Figure 3-29 ROM – IP Customization



1. File Configuration

File configuration mainly includes the basic information related to the ROM instantiation file, as shown in Figure 3-29.

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

Options configuration mainly includes the configuration information related to the ROM instantiation file, as shown in Figure 3-29.

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 specified in the Memory initialization File should be consistent with the data width configured. If not, the Init value of the generated ROM instantiation is 0 by default, and the following error message will be displayed:
- Error (MG2105): Initial values' width is unequal to user's width.

3. Ports Configuration Diagram

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 3-29.

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

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-30.

Figure 3-30 Help

ROM

Information

Type:	ROM
Vendor:	GOWIN Semiconductor
Summary:	B-SRAM can be configured as ROM. In this mode, the design supports different bit width data readings. 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.
	File Format - Set whether the format of the memory initialization file content is Binary or Hex.

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

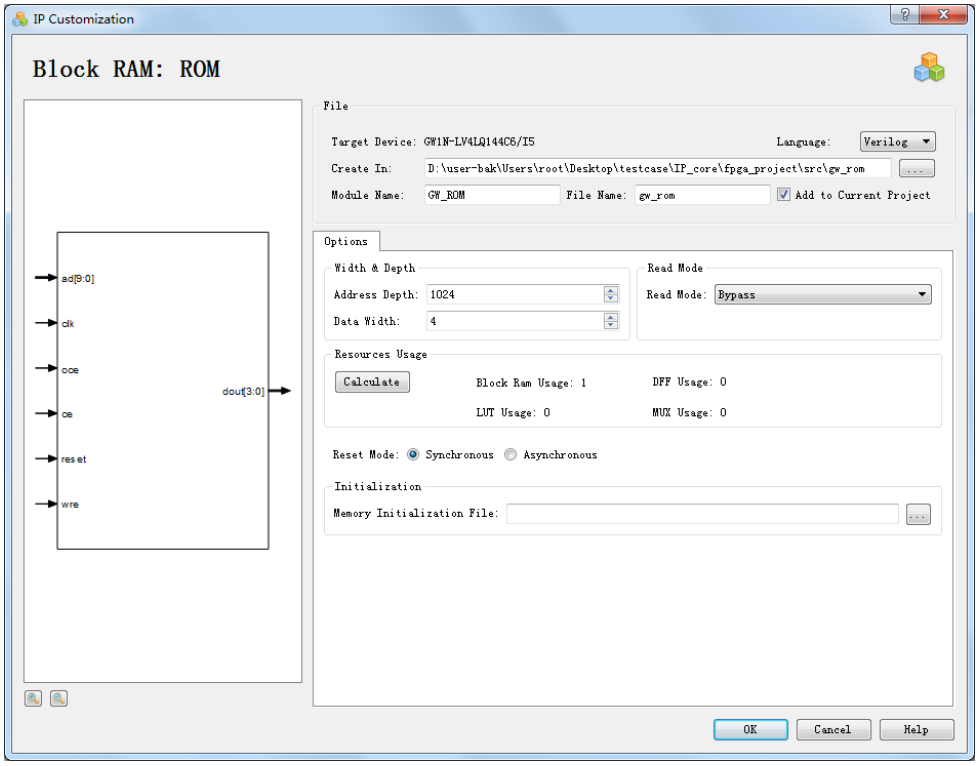
IP Generation Files

As shown in Figure 3-31, after customizing the IP, click “OK” to generate three files that are named according to the "File Name" specified in the File configuration:

- The design file for the Gowin Primitive ROM instantiation "gw_rom.v";
- The instantiation template file for the IP design file "gw_rom_tmp.v";
- The configuration files of Gowin Primitive ROM instantiation "gw_rom.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 3-31 ROM - IP Customization



Design File for the Gowin Primitive ROM Instantiation

The design file for the Gowin Primitive ROM instantiation is a complete Verilog module. ROM instantiation is generated according to the ROM configuration in "IP Customization" window, as shown in Figure 3-32.

Note!

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

Figure 3-32 Design File of Gowin Primitive ROM Instantiation

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

output [3:0] 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[3:0]),
    .CLK(clk),
    .OCE(oce),
    .CE(ce),
    .RESET(reset),
    .WRE(wre),
    .BLKSEL({gw_gnd,gw_gnd,gw_gnd}),
    .AD({gw_gnd,gw_gnd,ad[9:0],gw_gnd,gw_gnd})
);

defparam bram_rom_0.READ_MODE = 1'b0;
defparam bram_rom_0.BIT_WIDTH = 4;
defparam bram_rom_0.BLK_SEL = 3'b000;
defparam bram_rom_0.RESET_MODE = "SYNC";

endmodule //GW_ROM
```

Instantiation template file for the IP design file

For efficiency purposes, the IP Core Generator generates the template file while generating ROM design file instantiation, as shown in Figure 3-33.

Figure 3-33 Instantiation template file for the IP design file

```
GW_ROM your_instance_name(
    .dout(dout_o), //output [3:0] 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

Generate a specific ROM IP as follows:

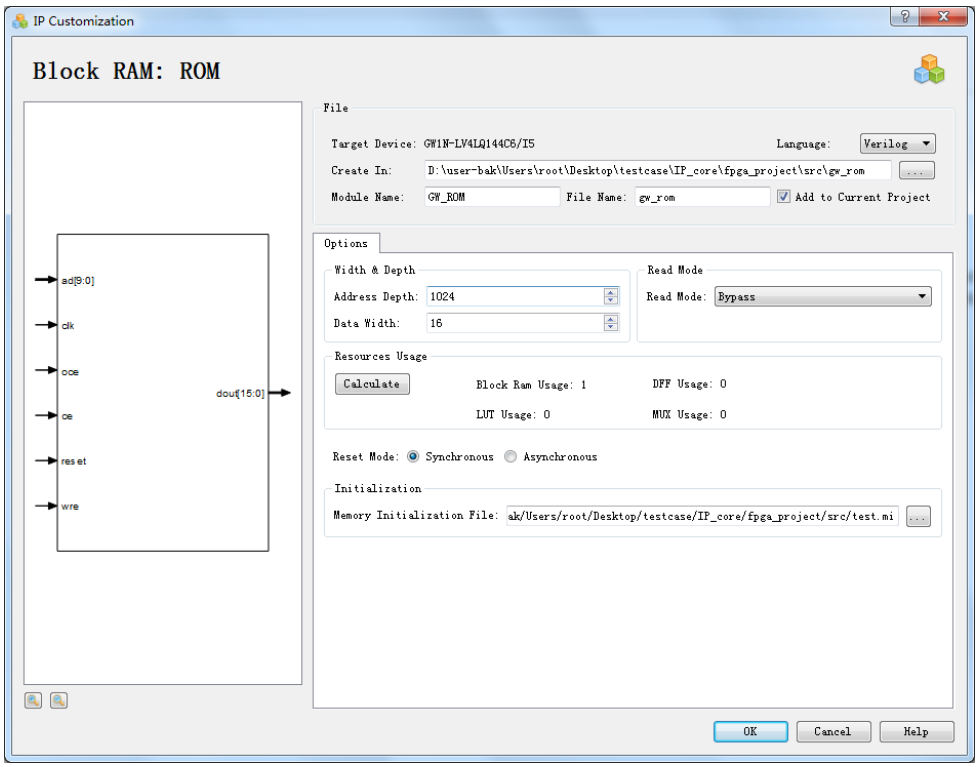
- Width and Depth: 1024 x 16;
- Bypass read mode;
- Synchronous.

Take the GW1N-4-LQFP144 device for instance; the configuration

page is as shown in Figure 3-34. Select a memory initialization file (.mi) for the module as required, and then click "OK" to generate the customized ROM IP design files.

The generated IP files are stored in the directory specified in the directory set in the "Create in" textbox. If “Add to Current Project” is checked, the generated IP design files will be automatically added to the project.

Figure 3-34 ROM –IP Customization Example



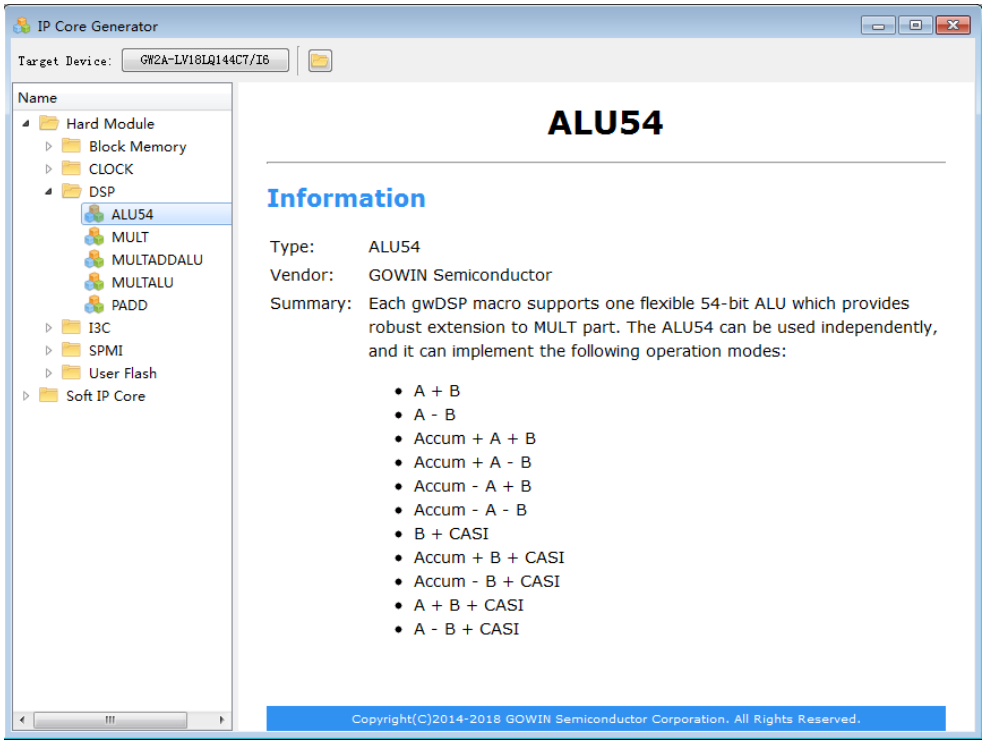
3.2 DSP

Currently, DSP module supports five Gowin devices generation: PADD, MAC, MULTADD, and MULTADDSUM.

3.2.1 ALU54

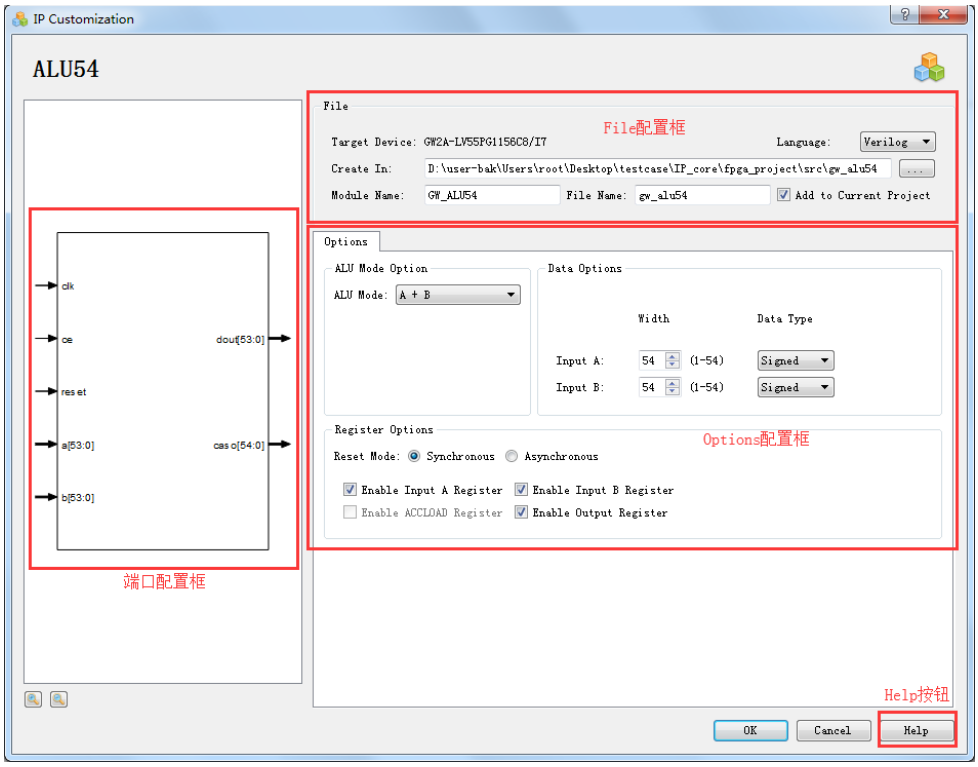
ALU54 can be used to implement 54-bit arithmetic and logical operations. Click "ALU54” on the IP Core Generator page. A brief introduction to the ALU54 will be displayed on the right of the screen, as shown in Figure 3-35.

Figure 3-35 ALU54 Summary



On the IP Core Generator page, double-click "ALU54" to open the "IP Customization" window, as shown in Figure 3-36. This displays file configuration, options configuration, the port configuration diagram, and the "Help" button.

Figure 3-36 ALU54 - IP Customization



1. File Configuration

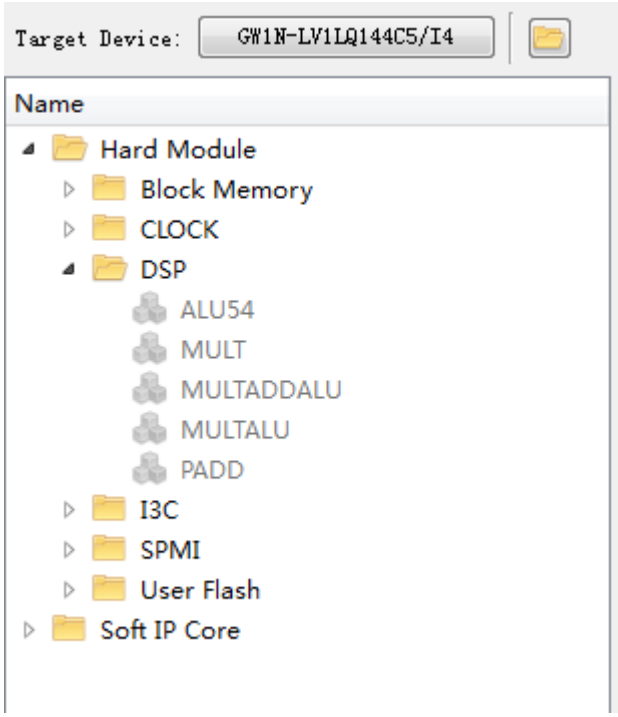
The file configuration mainly includes the basic information related to the ALU54 instantiation file, as shown in Figure 3-36.

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 greyed-out and un-available, as shown in Figure 3-37.

Figure 3-37 DSP Displaying in Grey



2. Options Configuration

Options configuration mainly includes configuration information related to the ALU54 instantiation file, as shown in Figure 3-36.

- ALU Mode Option: Allows users to select the operation modes The ALU can be configured to work 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 set 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 set as signed or unsigned.

- Register Options: Allows users to set registers working 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 ACCLOAD Register: Allows users to enable or disable ACCLOAD register;
  - Enable Output Register: Allows you to enable or disable Output register.

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 3-36.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-38.

Figure 3-38 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

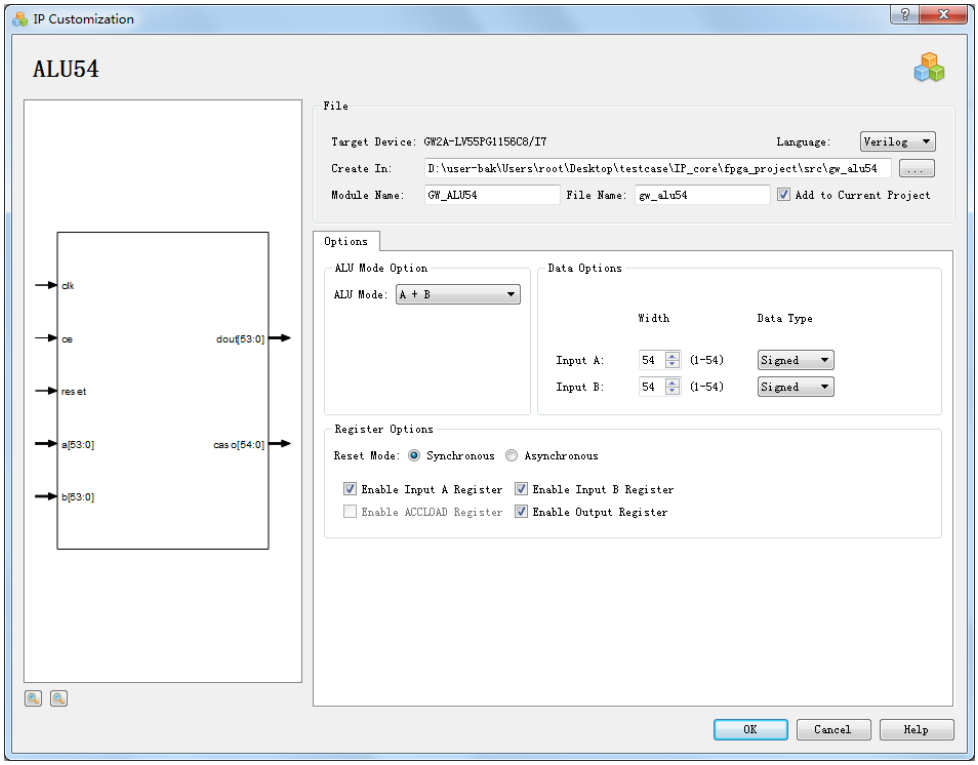
As shown in Figure 3-39, after customizing the IP, click “OK” to generate three files that are named according to the "File Name" specified in the file configuration:

- Design file for the Gowin Primitive ALU54 instantiation "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 3-39 IP Customization



The Design file for the Gowin Primitive ALU54 Instantiation

The design file for the Gowin Primitive ALU54 instantiation is a complete Verilog module. ALU54 instantiation is generated according to the OSC configuration specified in the "IP Customization" window, as shown in Figure 3-40.

Figure 3-40 The Design file for the Gowin Primitive ALU54 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 efficiency purposes, the IP Core Generator generates the template file while generating the ALU54 design file instantiation, as shown in Figure 3-41.

Figure 3-41 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

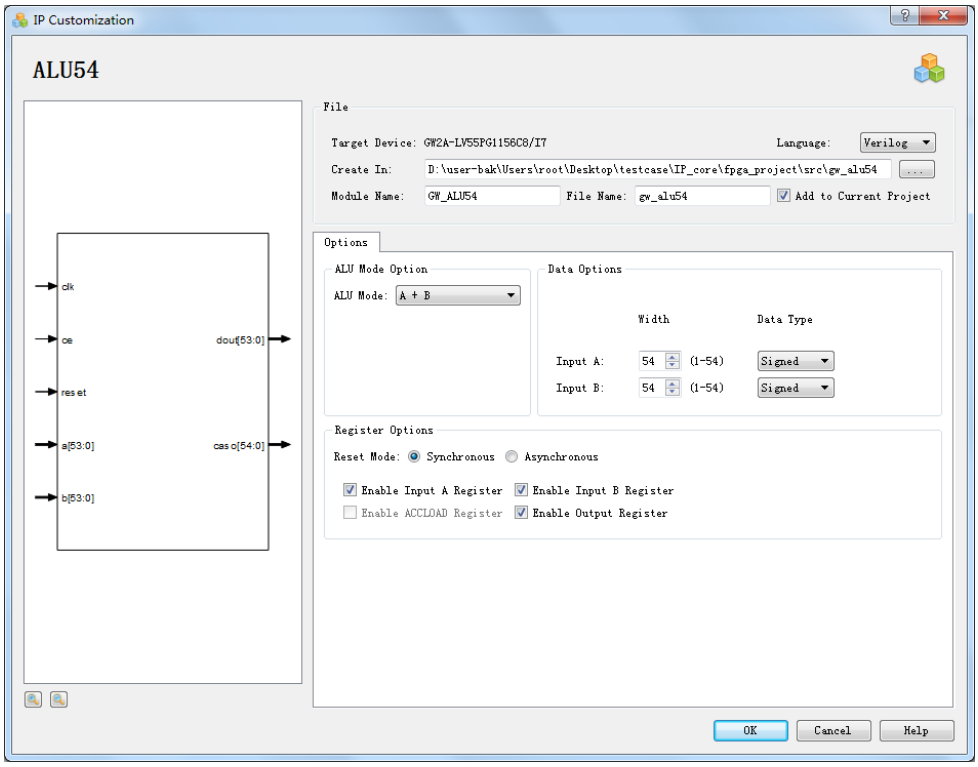
Generate a specific ALU54 IP with register as follows:

- Add operation: Addition of two 54-bit;
- Synchronous.

Take the GW2A-55-PBGA1156 device for instance, the configuration page is as shown in Figure 3-42. Click "OK" to generate the customized ALU54 IP design files.

The generated ALU54 IP files are stored in the directory specified in the "Create in" text box. If “Add to Current Project” is checked, the generated IP design files will be automatically added to the project.

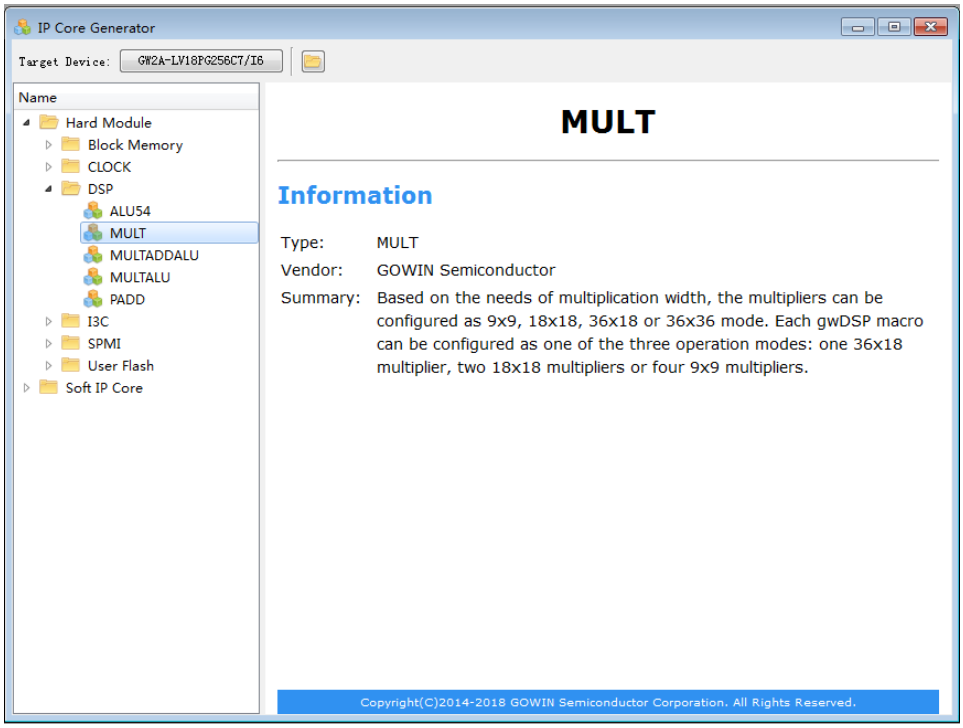
Figure 3-42 ALU54 – IP Customization



3.2.2 MULT

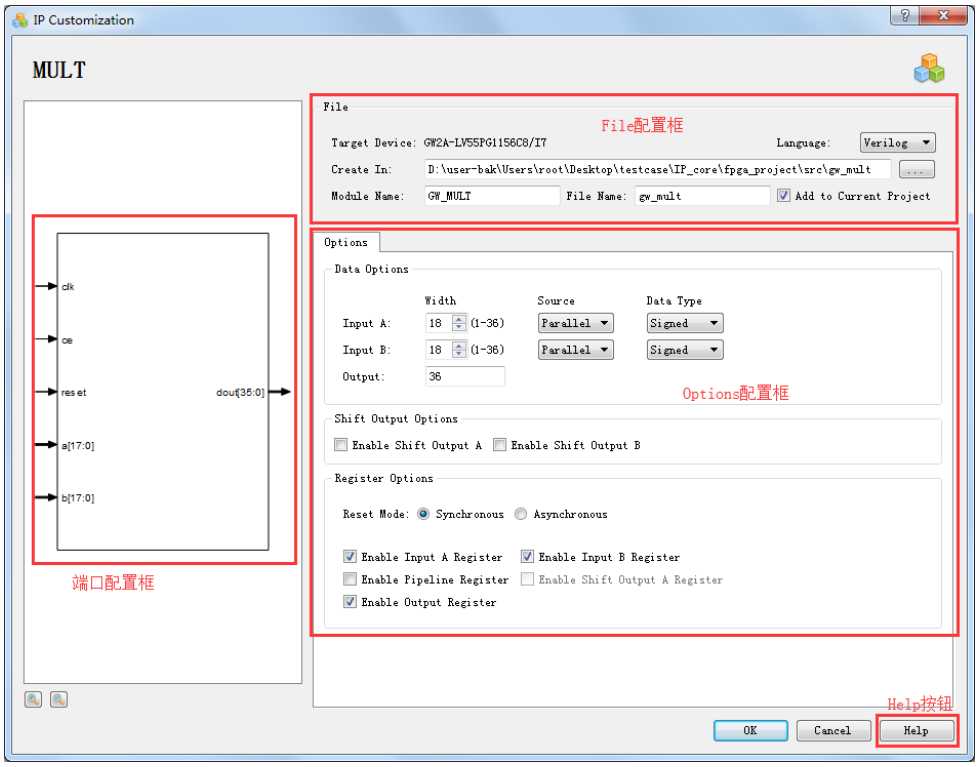
MULT can be configured as a multiplier. Click "MULT" on the IP Core Generator page. A brief introduction to the MULT will be displayed on the right of the screen, as shown in Figure 3-43.

Figure 3-43 MULT Summary



Double-click "MULT" to open the “IP Customization” window, as shown in Figure 3-44. This displays the File configuration, Options configuration, port configuration diagram, and the “Help” button.

Figure 3-44 MULT – IP Customization



1. File Configuration

The file configuration mainly includes the basic information related to

the MULT Instantiation file, as shown in Figure 3-44.

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 mainly includes configuration information related to the MULT instantiation file, as shown in Figure 3-44.

- Data Options: Allows users to set data options.
  - The maximum data width of 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 set 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. The Input/Output bit-width updates in real time based on the "Options" configuration, as shown in Figure 3-44.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-45. The Help page contains a general description of the IP Core, and a brief introduction to the "Options".

Figure 3-45 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

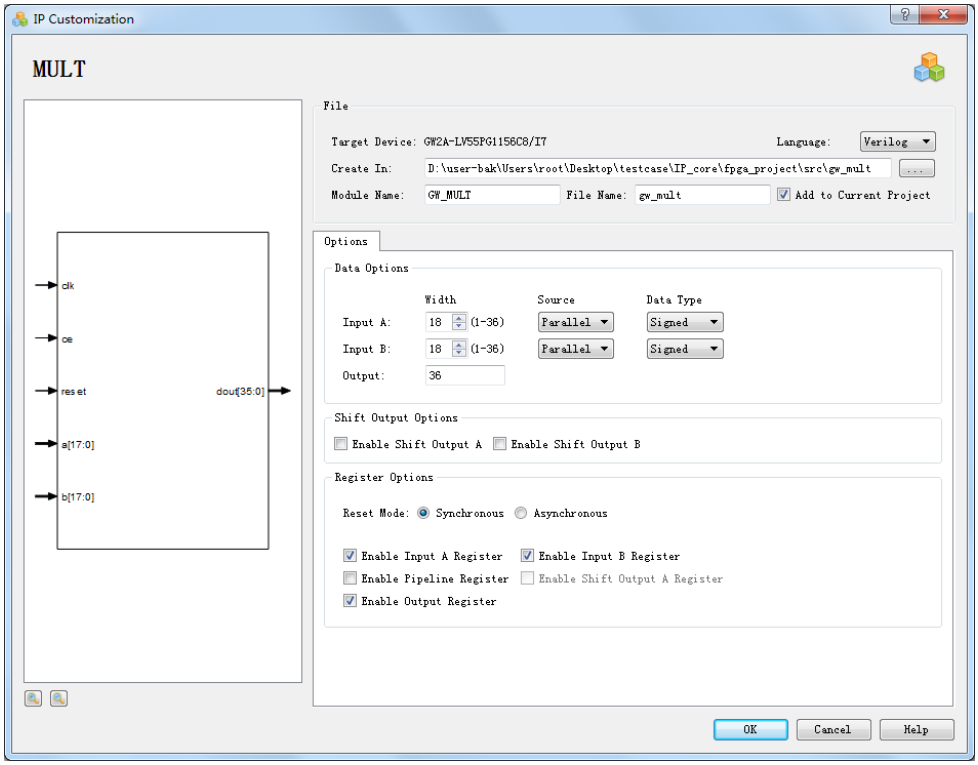
As shown in Figure 3-46, after customizing the IP, click “OK” to generate three files that are named according to the "File Name" specified in the File configuration:

- The design file for the Gowin Primitive MULT instantiation "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 an .vhd suffix. Taking verilog for instance, the following sections introduce the generated files.

The “IP Customization” window is as shown in Figure 3-46.

Figure 3-46 IP Customization



Design File for the Gowin Primitive MULT Instantiation

The design file for the Gowin Primitive MULT instantiation is a complete Verilog module. MULT instantiation is generated according to the MULT configuration specified in the “IP Customization” window, as shown in Figure 3-47.

Figure 3-47 Design File of Gowin Primitive MULT 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 efficiency purposes, the IP Core Generator generates the template file while generating the PADD design file instantiation, as shown in Figure 3-48.

Figure 3-48 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

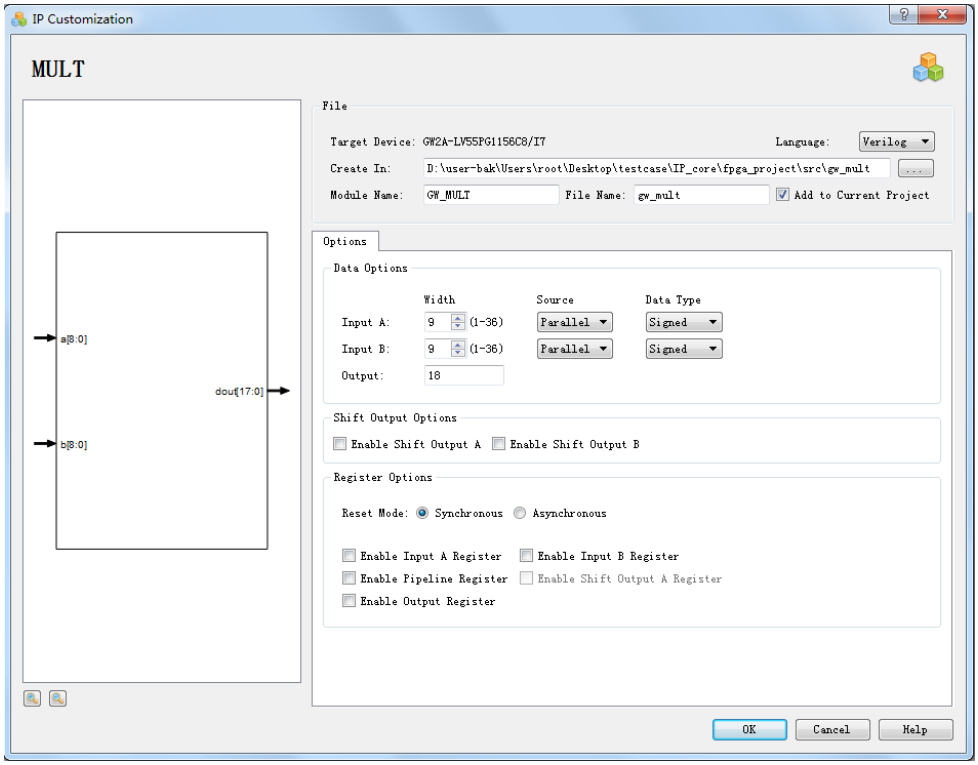
Generate a specific MULT IP as follows:

- 9-bit signed multiplier;
- Bypass mode.

Take the GW2A-55-PBGA1156 device for instance; the configuration page is as shown in Figure 3-49. Click "OK" to generate the customized MULT IP design files.

The generated IP files are stored in the directory specified in the directory set in the "Create in" textbox. If “Add to Current Project” is checked, the generated IP design files will be automatically added to the project.

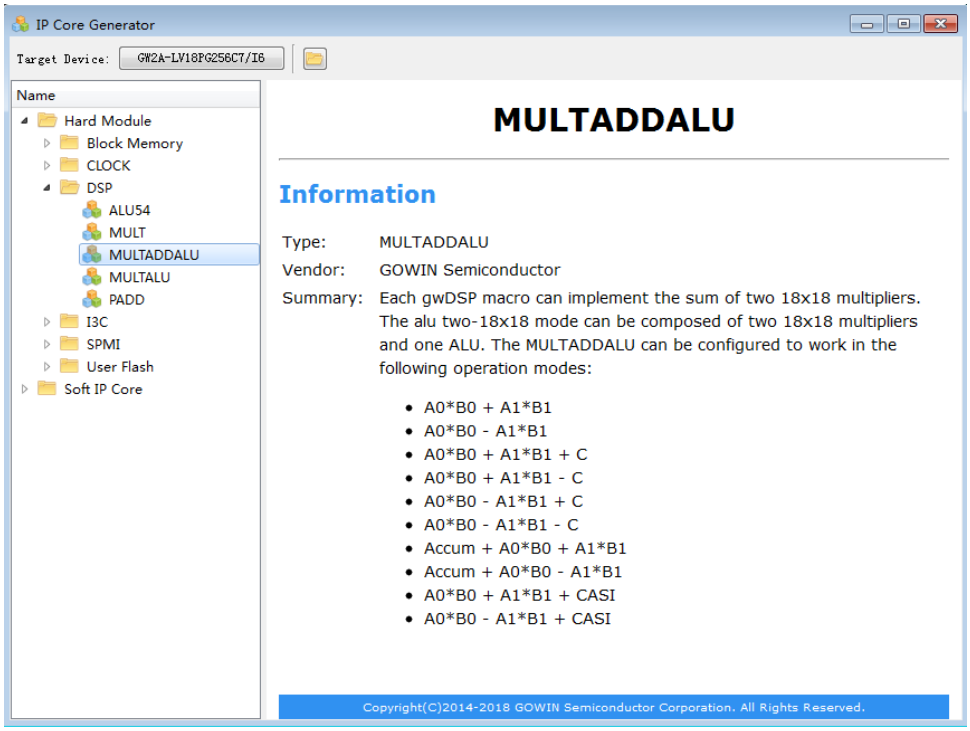
Figure 3-49 MULT – IP Customization



3.2.3 MULTADDALU

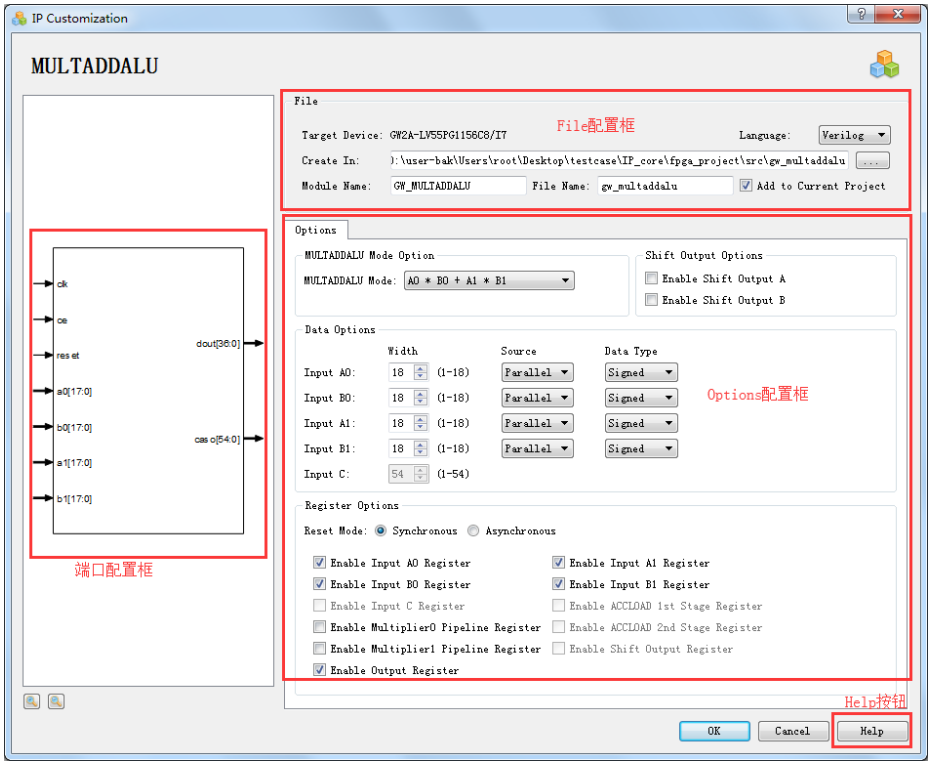
Each DSP macro can implement the sum of two 18x18 multipliers. Click "MULTADDALU" in the IP Core Generator page. A brief introduction to the MULTADDALU will be displayed on the right of the screen, as shown in Figure 3-50.

Figure 3-50 MULTADDALU Summary



Double-click "MULTADDALU" to open the “IP Customization” window. This displays the file configuration, options configuration, port configuration diagram, and the “Help” button, as shown in Figure 3-51.

Figure 3-51 MULT – IP Customization



1. File Configuration

The file configuration mainly includes the basic information related to the MULTADDALU instantiation file, as shown in Figure 3-51.

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 3-51.

- 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 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 3-51.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-52.

Figure 3-52 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".

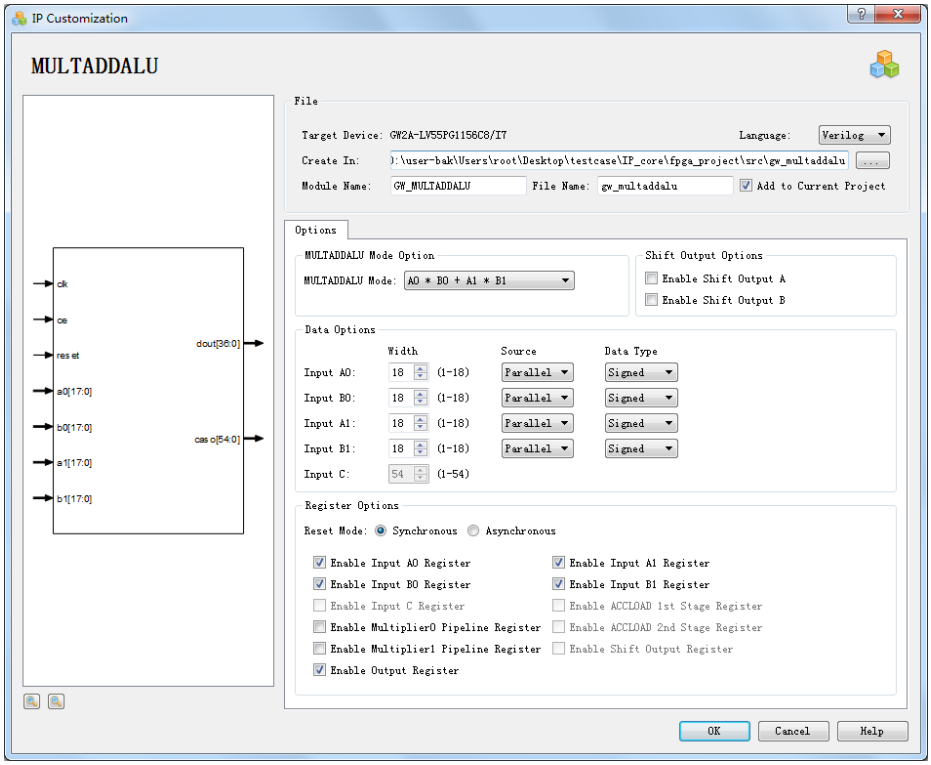
IP Generation Files

As shown in Figure 3-53, after customizing the IP, click “OK” to generate three files that are named according to the "File Name" specified in the File configuration:

- The design file for the Gowin Primitive MULTADDALU instantiation "gw_multaddalu.v";
- The Instantiation template file for the IP design file "gw_ multaddalu _tmp.v";
- The Configuration files for the Gowin Primitive MULTADDALU instantiation "gw_multaddalu.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 3-53 IP Customization



Design File for the Gowin Primitive MULTADDALU Instantiation

The design file for the Gowin Primitive MULTADDALU instantiation is a complete Verilog module. MULTADDALU instantiation is generated according to the MULTADDALU configuration specified in the “IP Customization” window, as shown in Figure 3-54.

Figure 3-54 Design File for the Gowin Primitive MULTADDSUM 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_
    .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,
    .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 efficiency purposes, the IP Core Generator generates the template file while generating MULTADDALU design file instantiation, as shown in Figure 3-55.

Figure 3-55 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

Generate a specific MULTADDALU IP as follows:

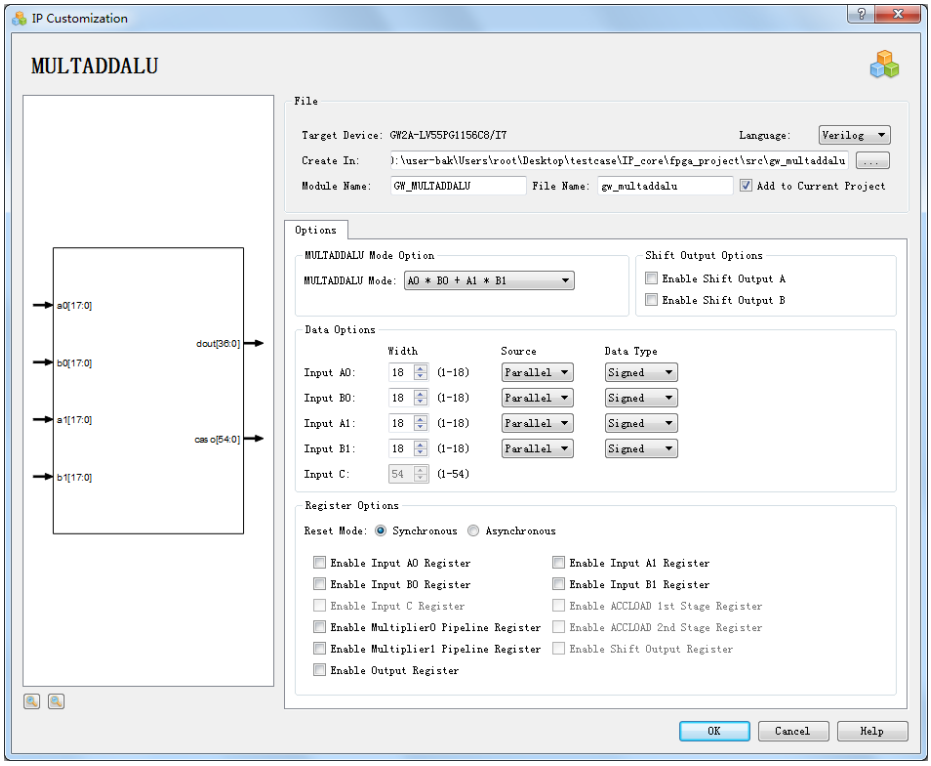
- Two sums of 18-bit signed multipliers;
- Synchronous reset;

- Bypass mode.

Take the GW2A-55-PBGA1156 device for instance, the configuration page is as shown in Figure 3-56. Click "OK" to generate the customized MULTADDALU IP design files.

The generated IP files are stored in the directory specified in the directory set in the "Create in" textbox. If “Add to Current Project” is checked, the generated IP design files will be automatically added to the project.

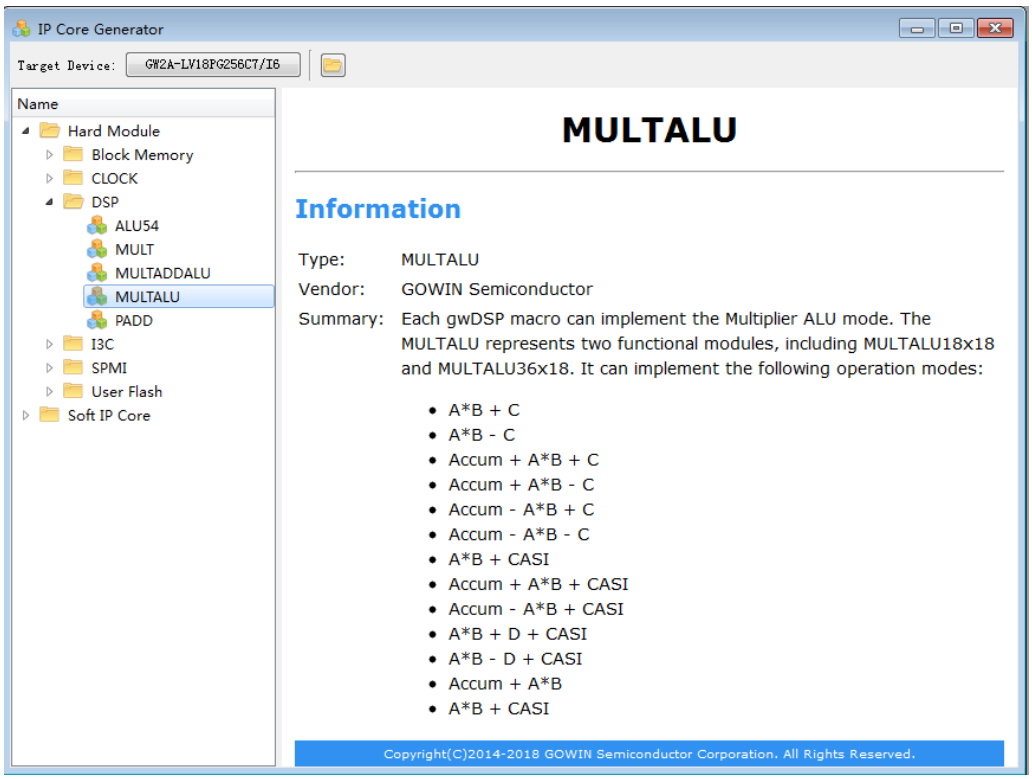
Figure 3-56 MULTADDALU - IP Customization



3.2.4 MULTALU

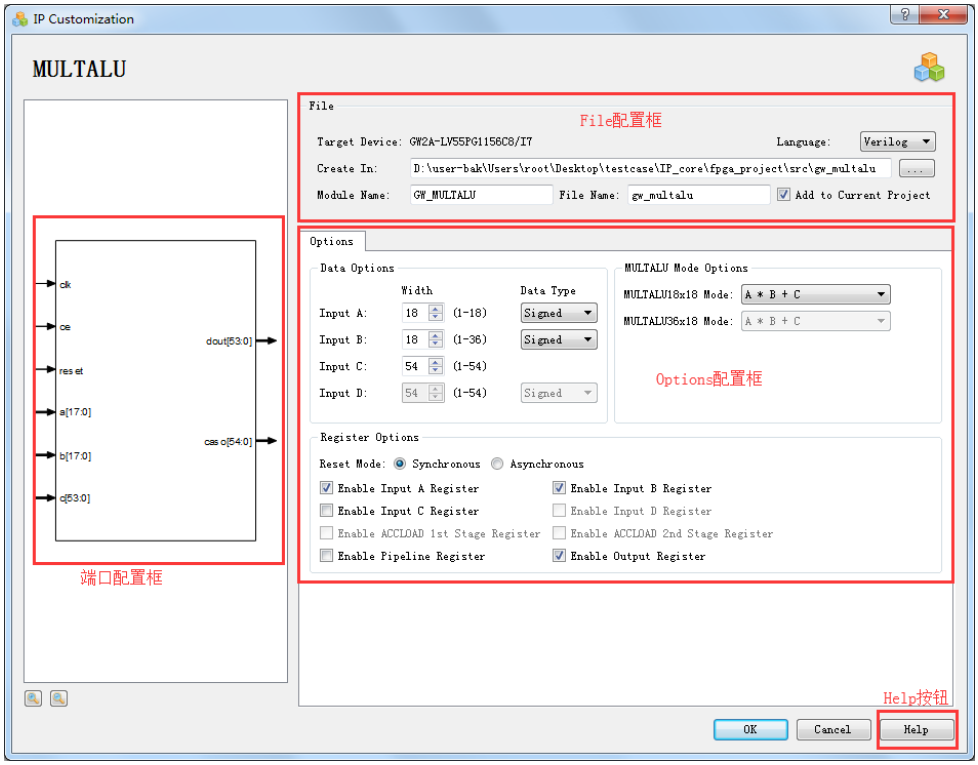
MULTALU can implement the Multiplier ALU mode. Click "MULTALU" on the IP Core Generator page. A brief introduction to the MULTALU will be displayed on the right of the screen, as shown in Figure 3-57.

Figure 3-57 MULTALU Summary



Double-click "MULTALU" to open the “IP Customization” window. This displays the file configuration, options configuration, port configuration diagram, and the “Help” button, as shown in Figure 3-58.

Figure 3-58 MULTALU – IP Customization



1. File Configuration

The File configuration mainly includes the basic information related to the MULTALU instantiation file, as shown in Figure 3-58.

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 mainly includes configuration information related to the MULTALU instantiation file, as shown in Figure 3-58.

- 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 is 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 3-58.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-59.

Figure 3-59 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.

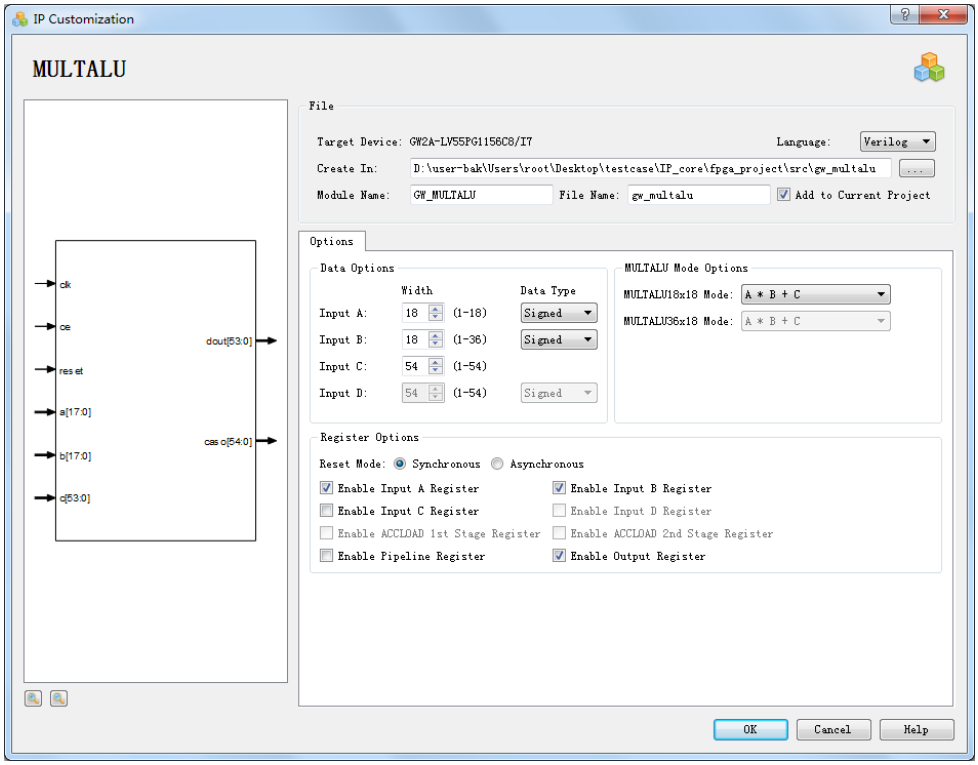
IP Generation Files

As shown in Figure 3-60, after customizing the IP, click “OK” to generate three files that are named according to the "File Name" specified in the File configuration:

- The design file for the Gowin Primitive MULTALU instantiation "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 an .vhd suffix. Taking verilog for instance, the following sections introduce the generated files.

Figure 3-60 IP Customization



Design File for the Gowin Primitive MULTALU Instantiation

The design file for the Gowin Primitive MULTALU instantiation is a complete Verilog module. MULTALU instantiation is generated according to the MULTALU configuration in the “IP Customization” window, as shown in Figure 3-61.

Figure 3-61 Design File for the Gowin Primitive MULTADD 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,
    .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_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 efficiency purposes, the IP Core Generator generates the template file while generating MULTALU design file instantiation, as shown in Figure 3-62.

Figure 3-62 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

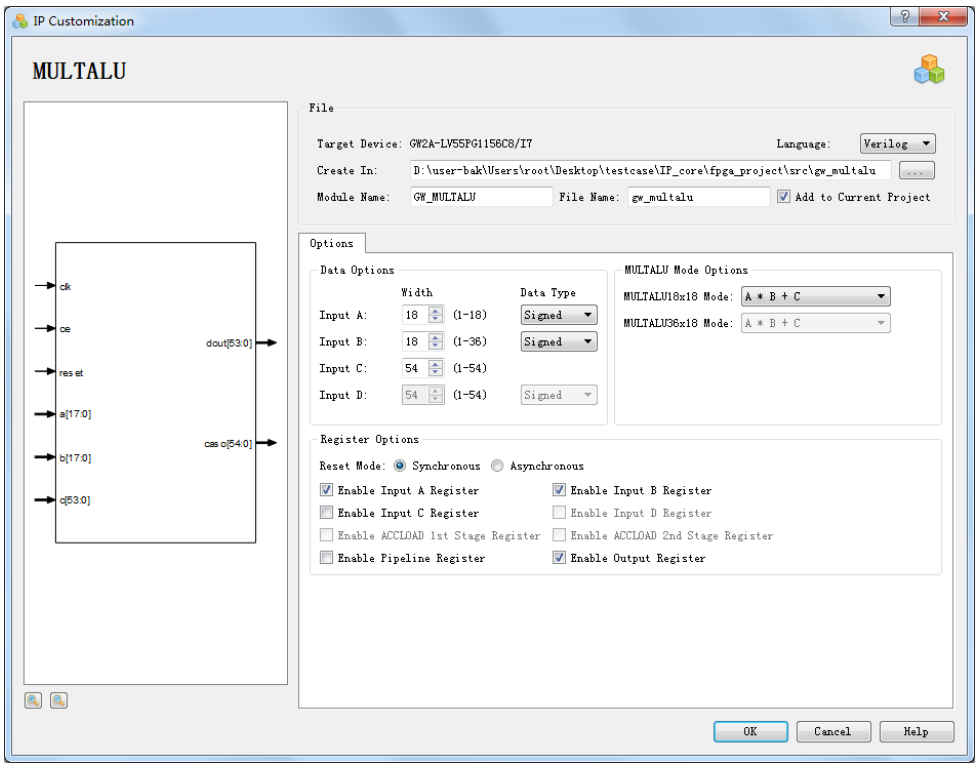
Generate a specific MULTALU IP as follows:

- 18-bit signed multipliers add;
- Register mode;
- Synchronous.

Take the GW2A-55-PBGA1156 device for instance; the configuration page is as shown in Figure 3-63. Click "OK" to generate the customized MULTADD IP design files.

The generated IP files are stored in the directory specified in the directory set in the "Create in" textbox. If “Add to Current Project” is checked, the generated IP design files will be automatically added to the project.

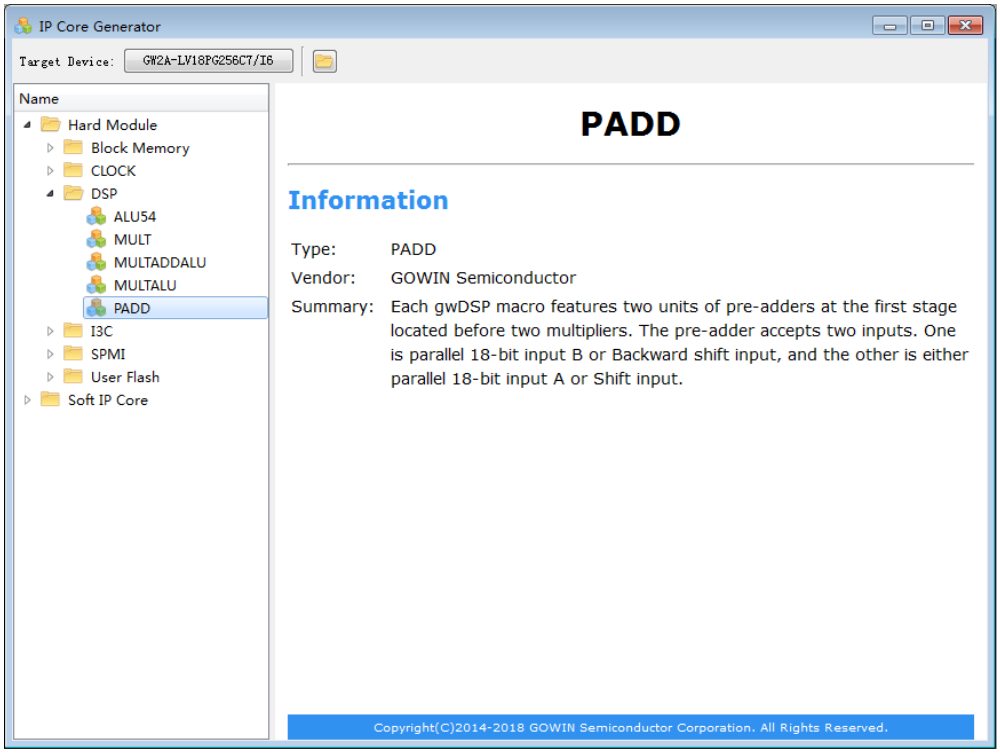
Figure 3-63 MULTALU - IP Customization



3.2.5 PADD

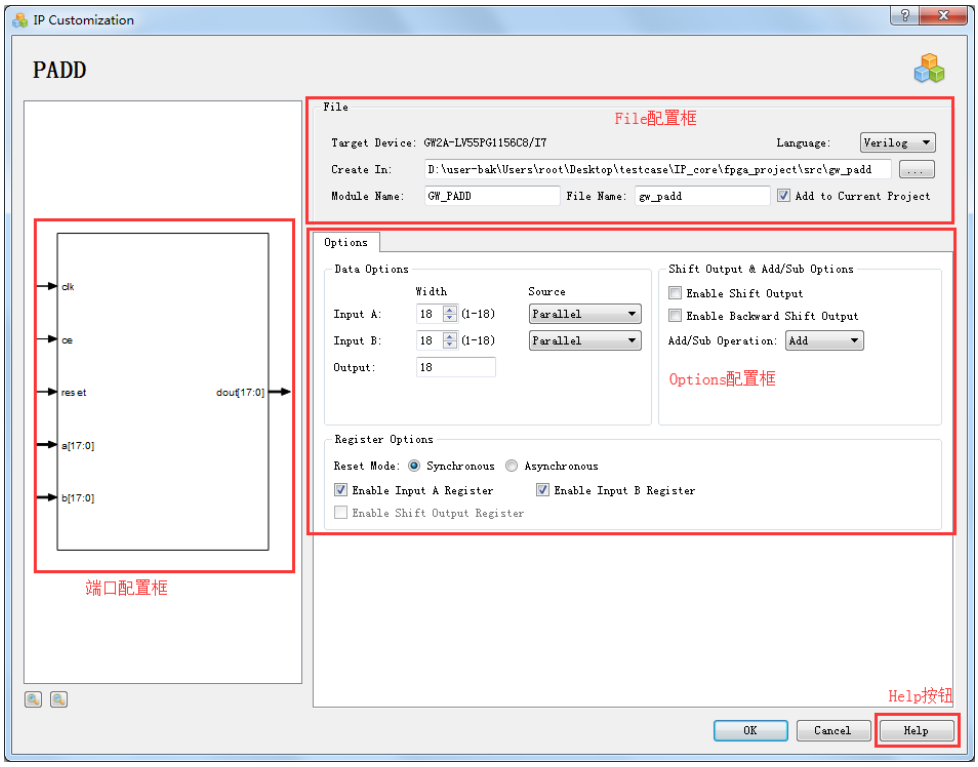
PADD can be configured as a pre-adder, pre-subtracter, or shifter. Click the "PADD" in the IP Core Generator page. A brief introduction to the PADD will be displayed on the right of the screen, as shown in Figure 3-64.

Figure 3-64 PADD Summary



Double-click on "PADD" to open the “IP Customization” window. This displays the file configuration, options configuration, port configuration diagram, and the “Help” button, as shown in Figure 3-65.

Figure 3-65 PADD – IP Customization



1. File Configuration

The file configuration mainly includes the basic information related to the PADD instantiation file, as shown in Figure 3-65.

The PADD 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 mainly includes configuration information related to the PADD instantiation file, as shown in Figure 3-65.

- Data Options: Allows users to set 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 per ports signal.
- Register Options: Allows users to set registers working 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 3-65.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-66. The Help page contains a general description of the IP Core, and a brief introduction to the "Options".

Figure 3-66 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.

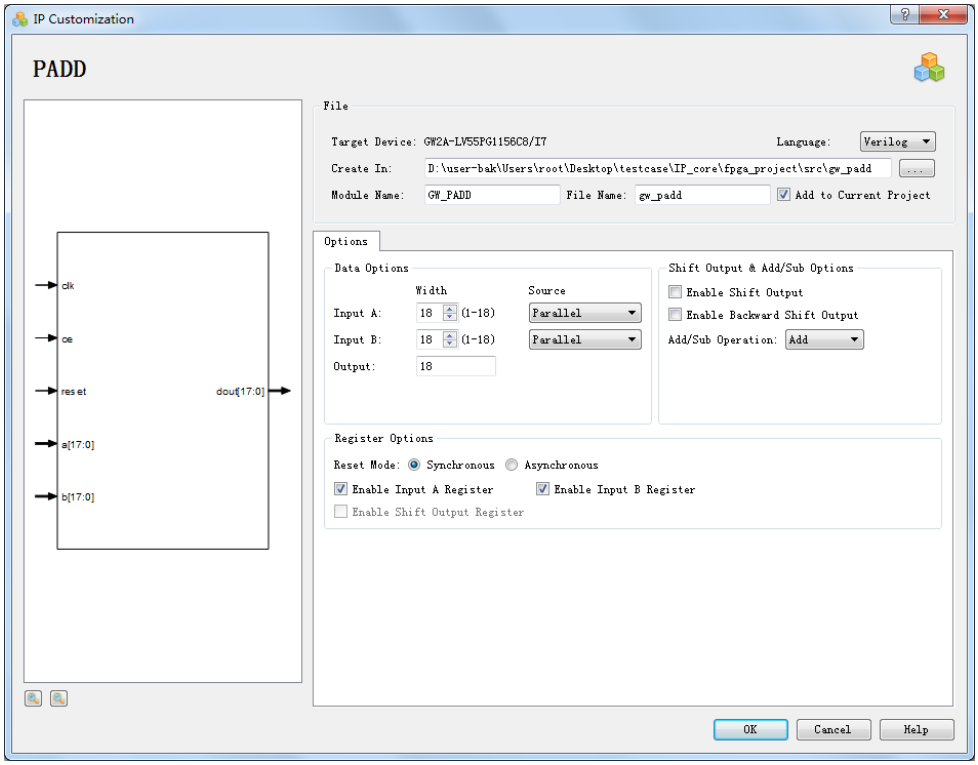
IP Generation Files

As shown in Figure 3-67, after customizing the IP, click “OK” to generate three files that are named according to the "File Name" specified in the File configuration:

- The design file for the 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 3-67 IP Customization



Design File for the Gowin Primitive PADD Instantiation

The design file for the Gowin Primitive PADD instantiation is a complete Verilog module. PADD instantiation is generated according to the PADD configuration provided in the"IP Customization" window, as shown in Figure 3-68.

Figure 3-68 Design File for the Gowin Primitive PADD 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_gn
    .SBI({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_(
    .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 3-69.

Figure 3-69 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

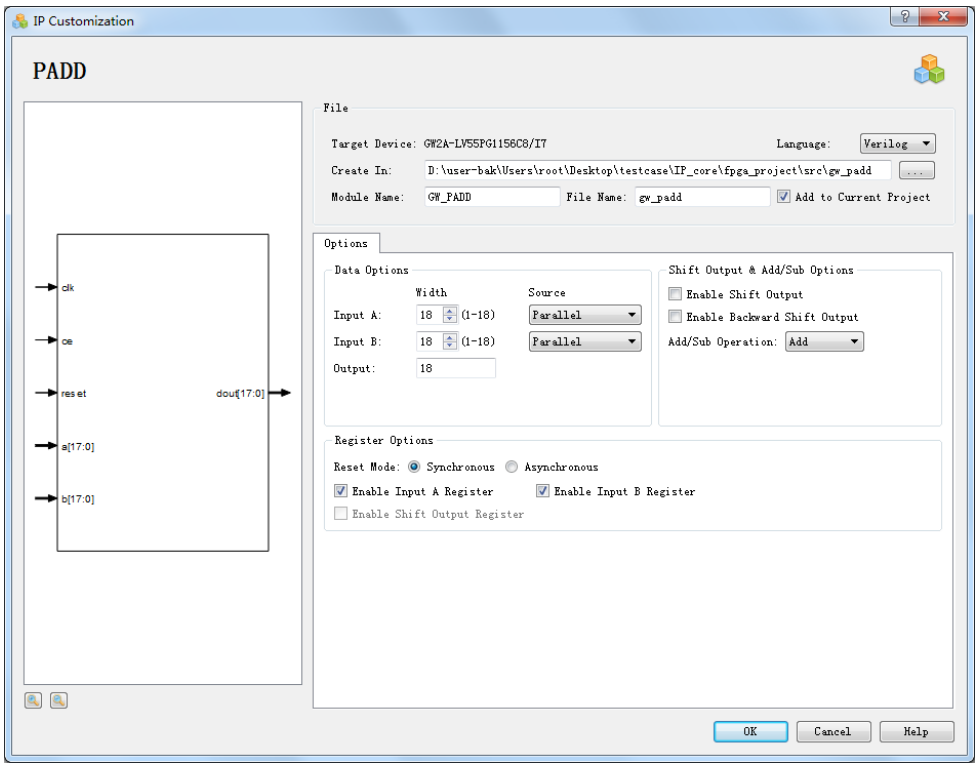
Generate a specific PADD IP as follows:

- Input Width:18;
- Add operation;
- Synchronous.

Take theGW2A-55-PBGA1156 device for instance, the configuration page is as shown in Figure 3-70. Click "OK" to generate the customized PADD IP design files.

The generated IP files are stored in the directory specified in the directory set in the "Create in" textbox. If “Add to Current Project” is checked, the generated IP design files will be automatically added to the project.

Figure 3-70 PADD – IP Customization



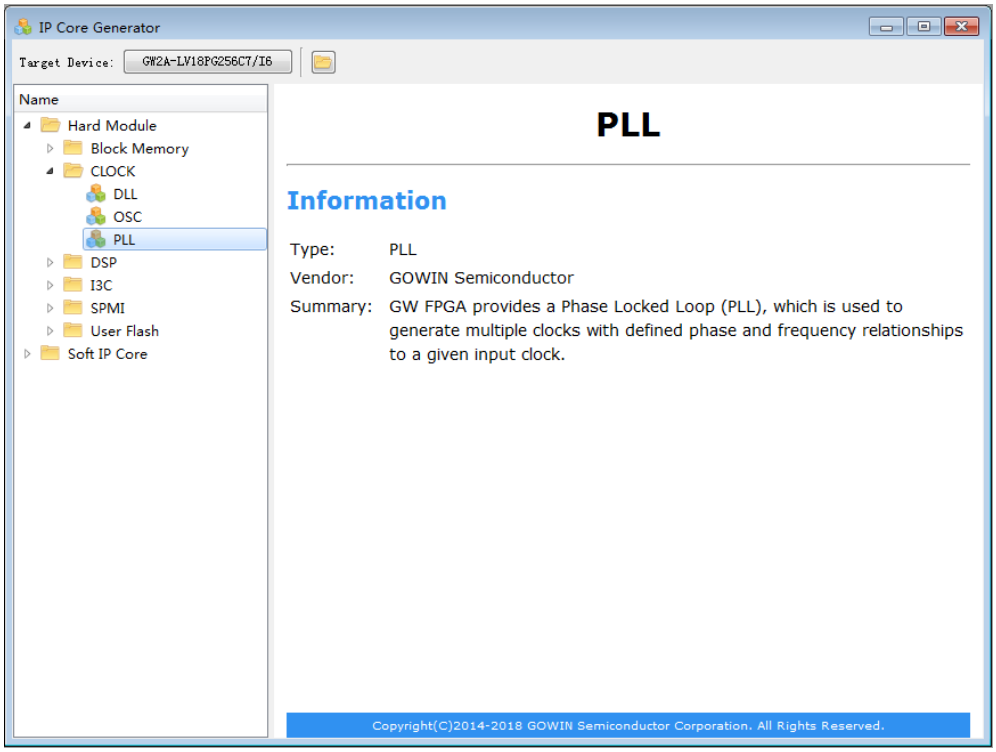
3.3 CLOCK

The CLOCK module currently supports three Gowin devices generation: PLL, DLL, and 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 page. A brief introduction to the PLL will be displayed on the right of the screen, as shown in Figure 3-71.

Figure 3-71 PLL Summary



The formulas for PLL output calculation are as follows:

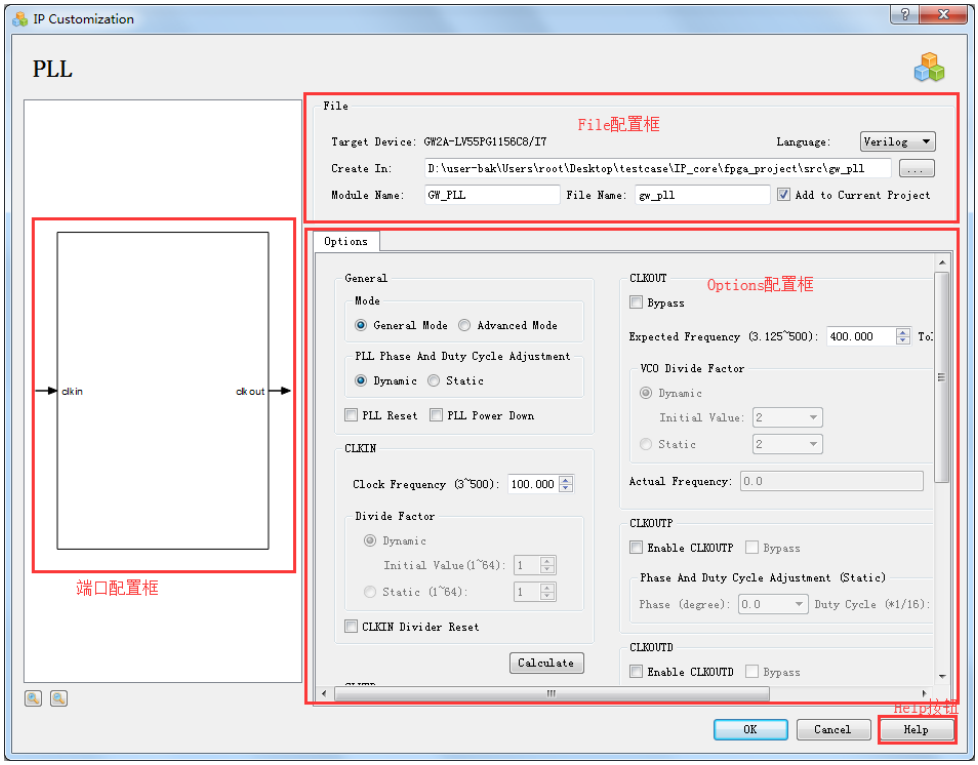
1. $f_{CLKOUT} = (f_{CLKIN}*FDIV)/IDIV$
2. $f_{CLKOUTD} = f_{CLKOUT}/SDIV$
3. $f_{VCO} = f_{CLKOUT}*ODIV$

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 on the "PLL" to open the “IP Customization” window. This displays the file configuration, options configuration, port configuration diagram, and the “Help” button, as shown in Figure 3-72.

Figure 3-72 PLL – IP Customization



1. File Configuration

The file configuration mainly includes the basic information related to the PLL instantiation file, as shown in Figure 3-72.

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 mainly includes configuration information related to the PLL instantiation file, as shown in Figure 3-72.

- 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: Used to 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: Allows users to enable or disable PLL Reset.
  - PLL Power Down: Allows users to configure the reset_p port to enable PLL in power down mode.
- CLKIN: Allows you 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 configuration is illegal, an error message will be displayed when

the user clicks on "Calculate" or "OK" as shown in Figure 3-73.
  - CLKIN Divider Reset: Allows users to configure the CLKIN Divider Reset.
- 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 3-73
- 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 3-73.
  - 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: Used to enable/disable CLKOUTP;
  - Bypass: Allows users to enable/disable CLKOUTP bypass;
  - Phase And Duty Cycle Adjustment (Static): Set (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: Used to 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 3-74;
  - 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 is enabled; used to configure the CLKOUTD/ CLKOUTD3 Divider Reset.
- 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 3-75 is the error message that is displayed when the actual frequency and expected frequency of CLKOUT are different;
  - Figure 3-76 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 3-77.
  - Otherwise, a success message prompt will be displayed as shown in Figure 3-78.

Figure 3-73 Error - Illegal Configuration of Divide Factor

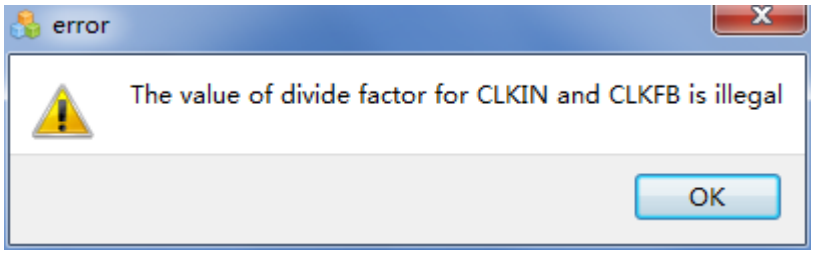


Figure 3-74 Error - Illegal Configuration of CLKOUTD

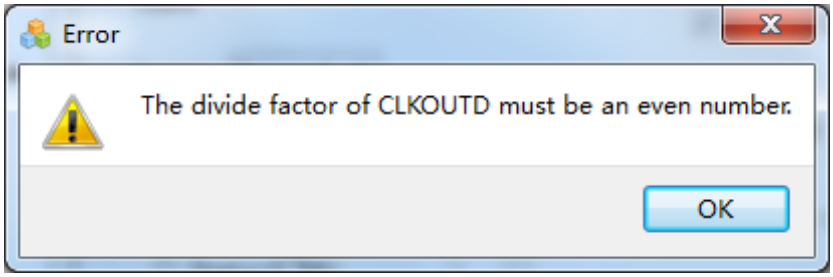


Figure 3-75 Error - Unequal Frequency of CLKOUT

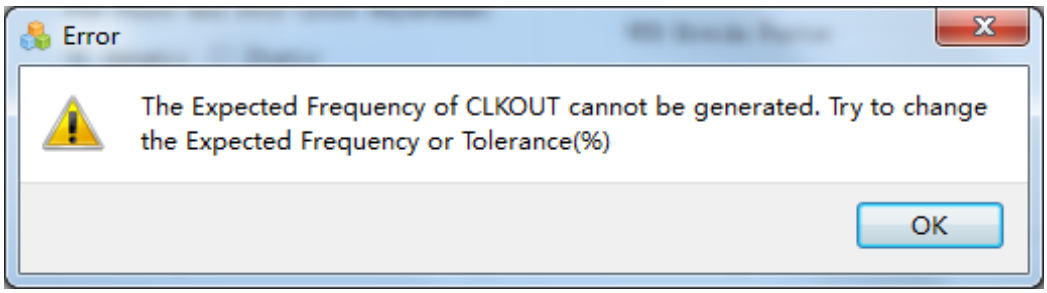


Figure 3-76 Error - Unequal Frequency of CLKOUT

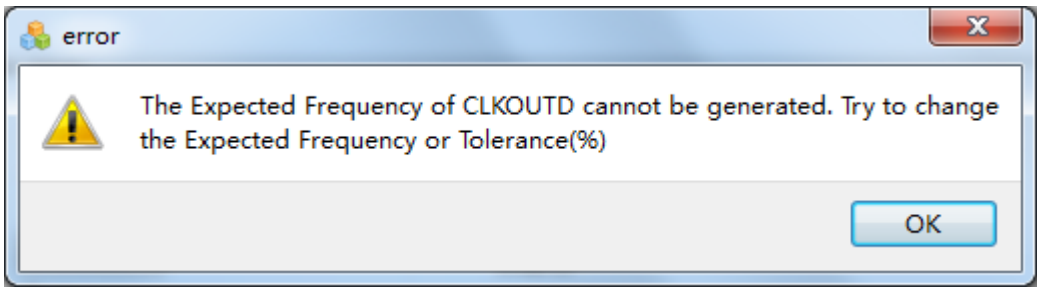


Figure 3-77 Error - Illegal Configuration of VCO

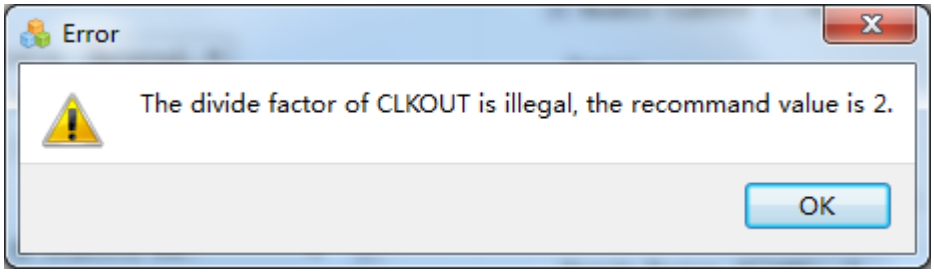
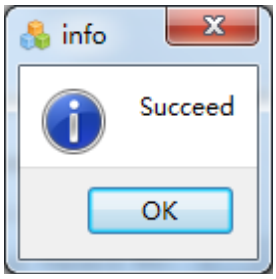


Figure 3-78 Info - Succeed



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 3-72.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-79. Help page contains the IP Core general description, and brief introduction of "Options".

Figure 3-79 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 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.
	Source - Specify the source of feedback.

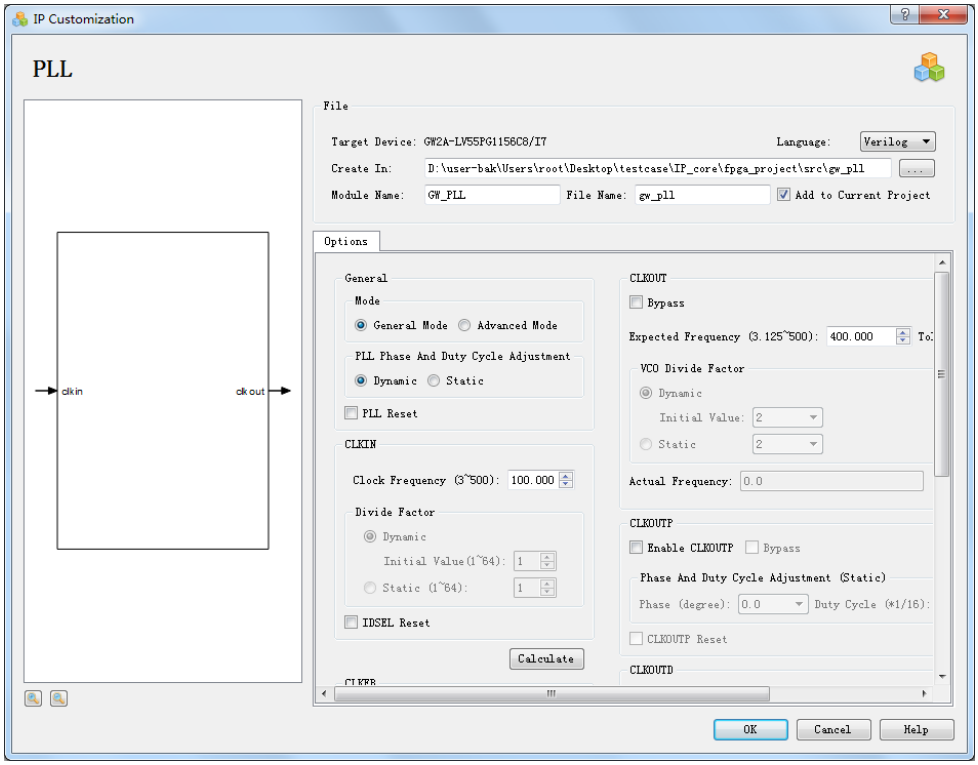
IP Generation Files

As shown in Figure 3-80, after customizing the IP, click “OK” to generate three files that are named according to the "File Name" specified in the File configuration:

- The design file of the Gowin Primitive PLL instantiation "gw_pll.v";
- The instantiation template file for the IP design file "gw_pll_tmp.v";
- The configuration files for the Gowin Primitive PLL instantiation "gw_pll.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 3-80 IP Customization



Design File for the Gowin Primitive PLL Instantiation

The design file for the Gowin Primitive PLL instantiation is a complete Verilog module. PLL instantiation is generated according to the MULT configuration displayed in the “IP Customization” window, as shown in Figure 3-81.

Figure 3-81 Design File for the Gowin Primitive PLL Instantiation

```
module GW_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 //GW_PLL
```

Instantiation Template File for the IP Design File

For efficiency purposes, the IP Core Generator generates the template file while generating PLL design file instantiation, as shown in Figure 3-82.

Figure 3-82 Instantiation Template File for the IP Design File

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

PLL Generation Example

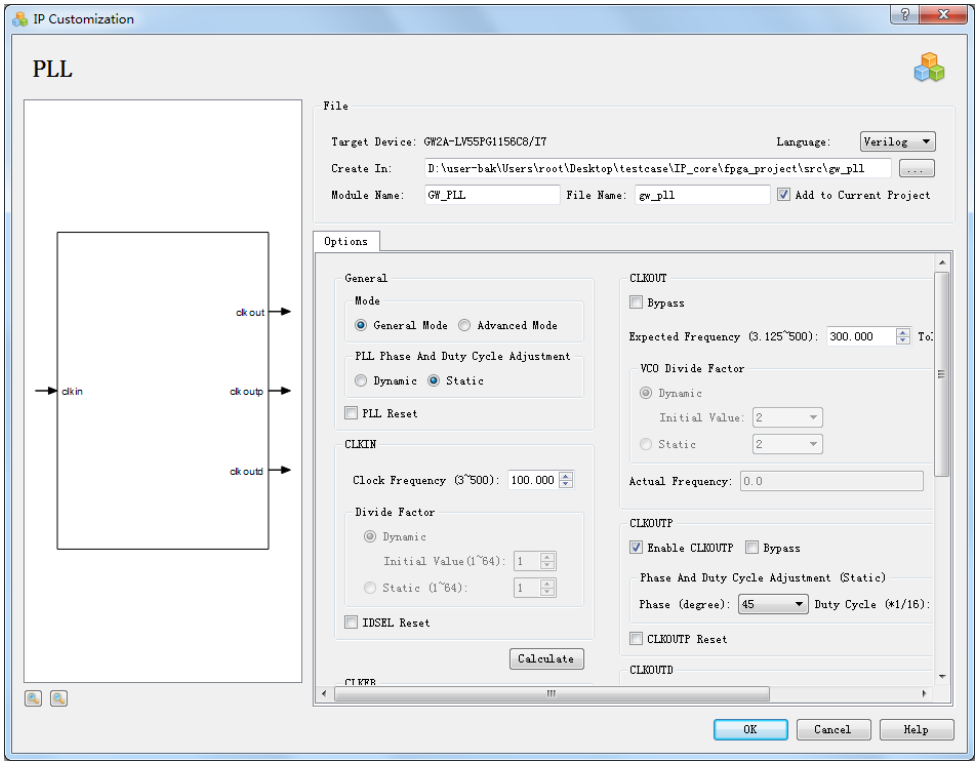
Generate a specific PLL IP as follows:

- 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 GW1N-4-PBGA256 device and general mode for instance; the configuration page is as shown in Figure 3-83. Click "OK" to generate the customized PLL IP design files.

The generated IP files are stored in the directory specified in the directory set in the "Create in" textbox. If “Add to Current Project” is checked, the generated IP design files will be automatically added to the project.

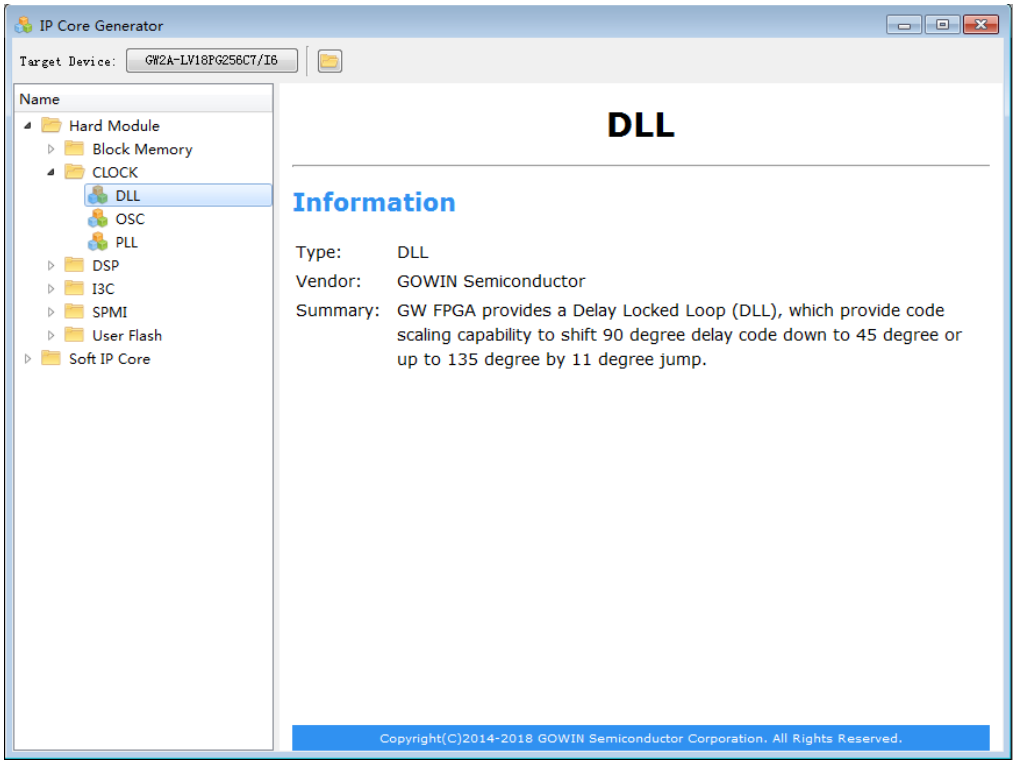
Figure 3-83 PLL – IP Customization



3.3.2 DLL

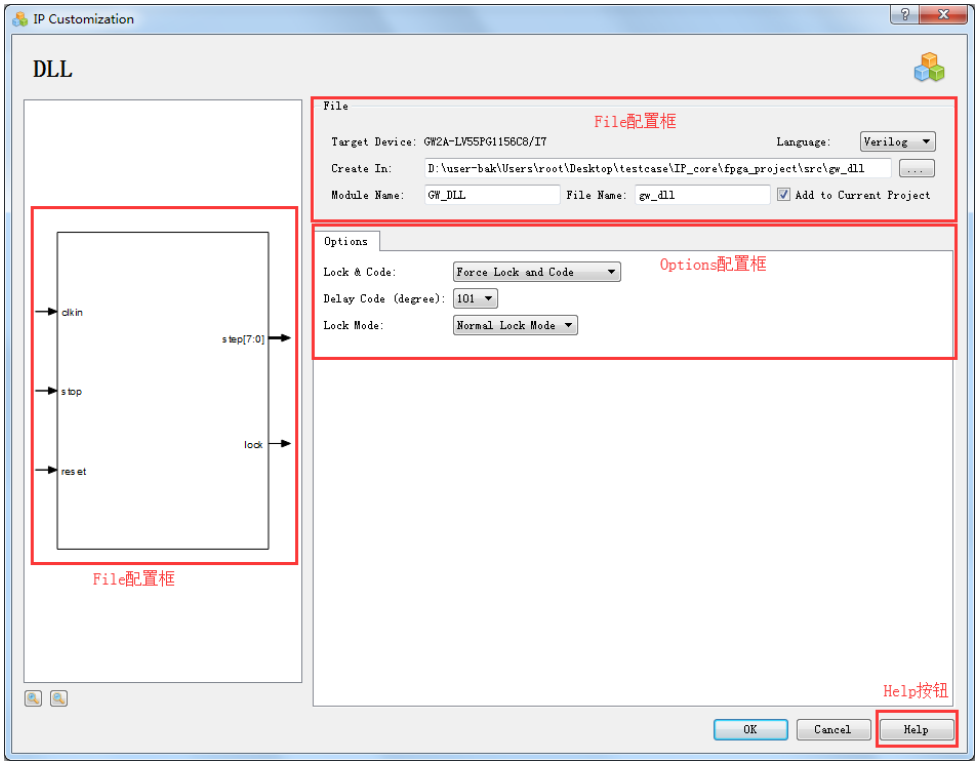
Delay Lock Loop (DLL) is mainly used to ensure accurate time delay by the equal and precise division of the input signal cycle. Click "DLL" on the IP Core Generator page. A brief introduction to the DLL will be displayed on the right of the screen, as shown in Figure 3-84.

Figure 3-84 DLL Summary



Double-click "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 3-85.

Figure 3-85 DLL – IP Customization



1. File Configuration

The file configuration mainly includes the basic information related to the DLL instantiation file, as shown in Figure 3-85.

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 3-85.

- 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 3-85.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-86.

Figure 3-86 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

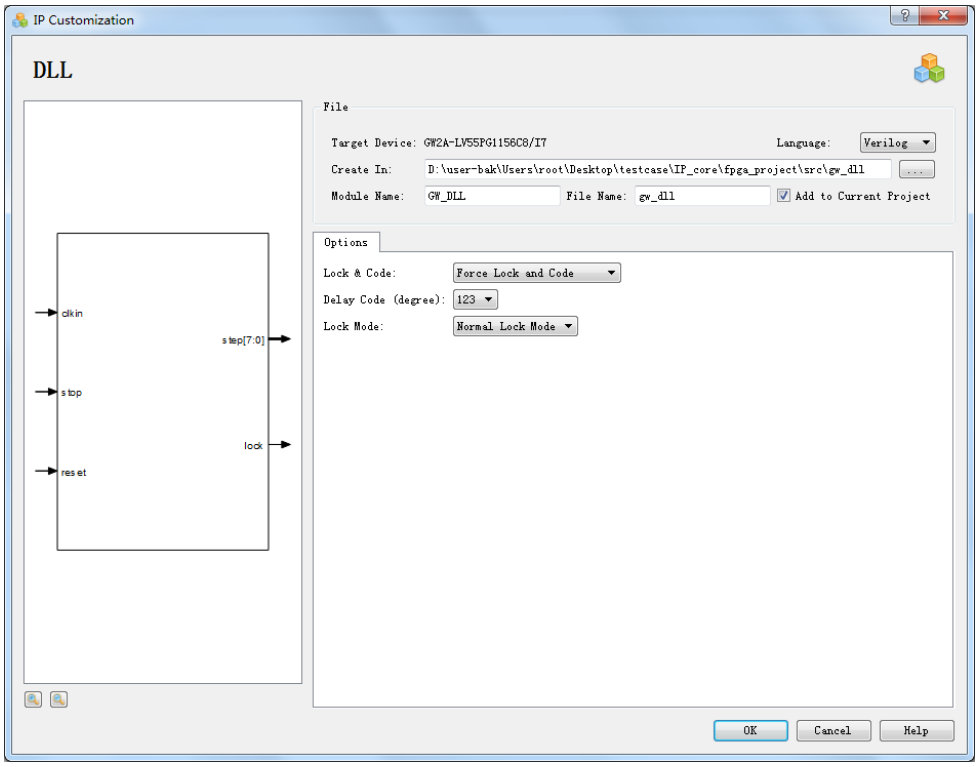
As shown in Figure 3-87, after customizing the IP, click “OK” to generate three files that are named according to the "File Name" specified in the File configuration:

- The design file for the 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 3-87 IP Customization



Design File for the Gowin Primitive DLL Instantiation

The design file for the Gowin Primitive DLL 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 3-88.

Figure 3-88 Design File of Gowin Primitive DLL 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 = "010";
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 efficiency purposes, the IP Core Generator generates the template file while generating DLL design file instantiation, as shown in Figure 3-89.

Figure 3-89 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

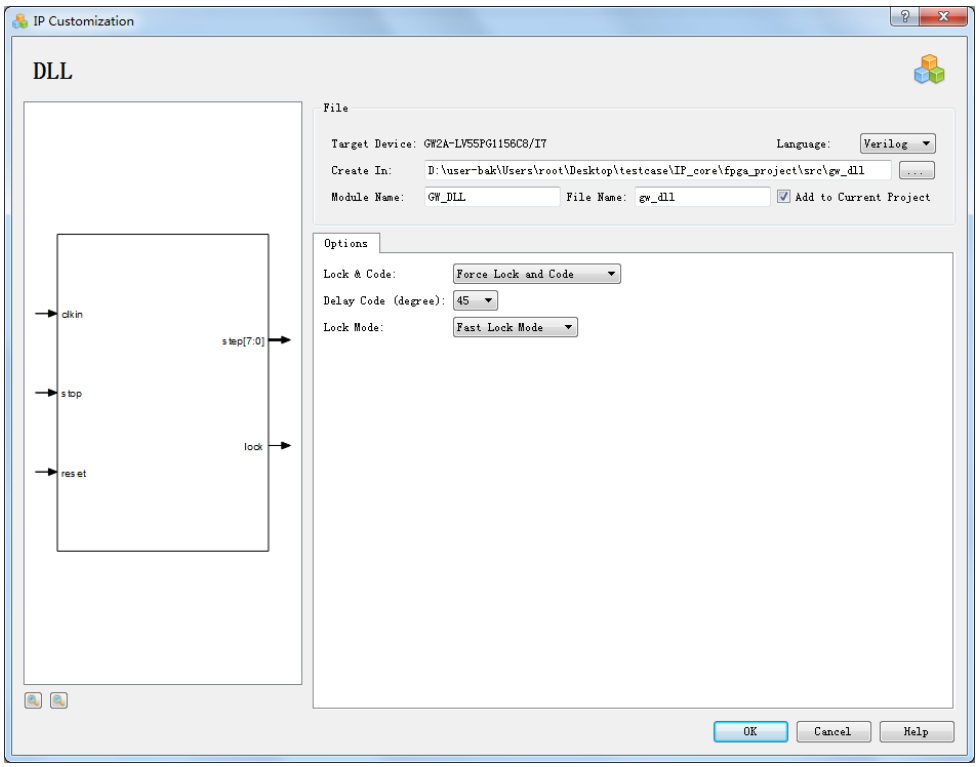
Generate a specific DLL IP as follows:

- Delay Code is 45°;
- Fast lock mode.

Take the GW1N-4-LQFP144 device for instance; the configuration page is as shown in Figure 3-90. Click "OK" to generate the customized DLL IP design files.

The generated IP files are stored in the directory specified in the directory set in the "Create in" textbox. If “Add to Current Project” is checked, the generated IP design files will be automatically added to the project.

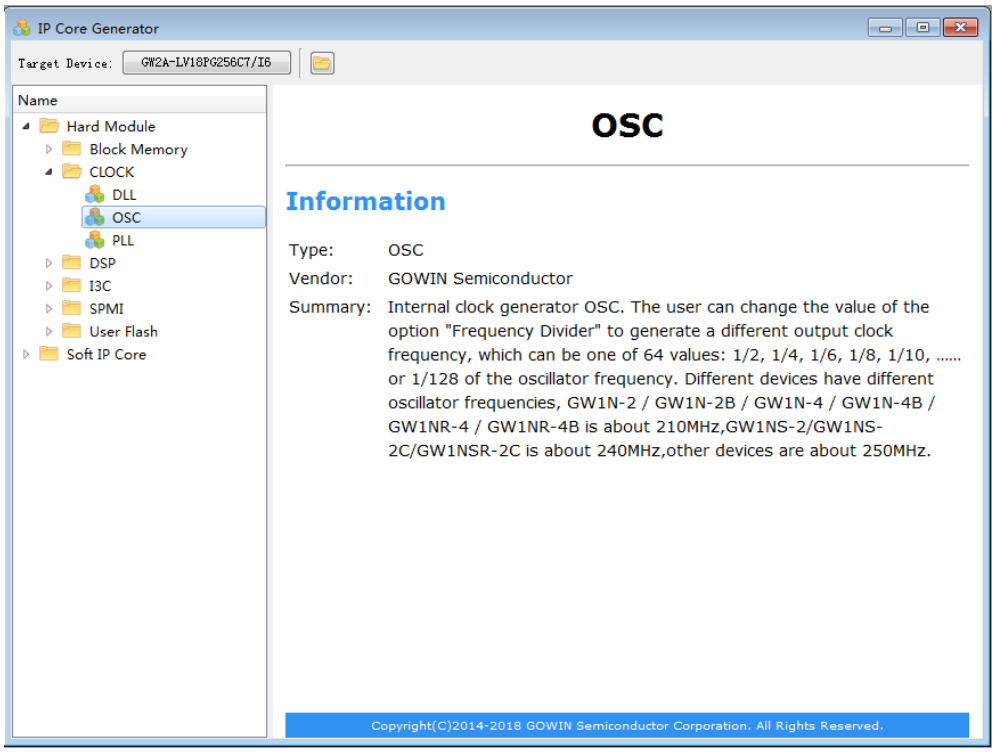
Figure 3-90 DLL – IP Customization



3.3.3 OSC

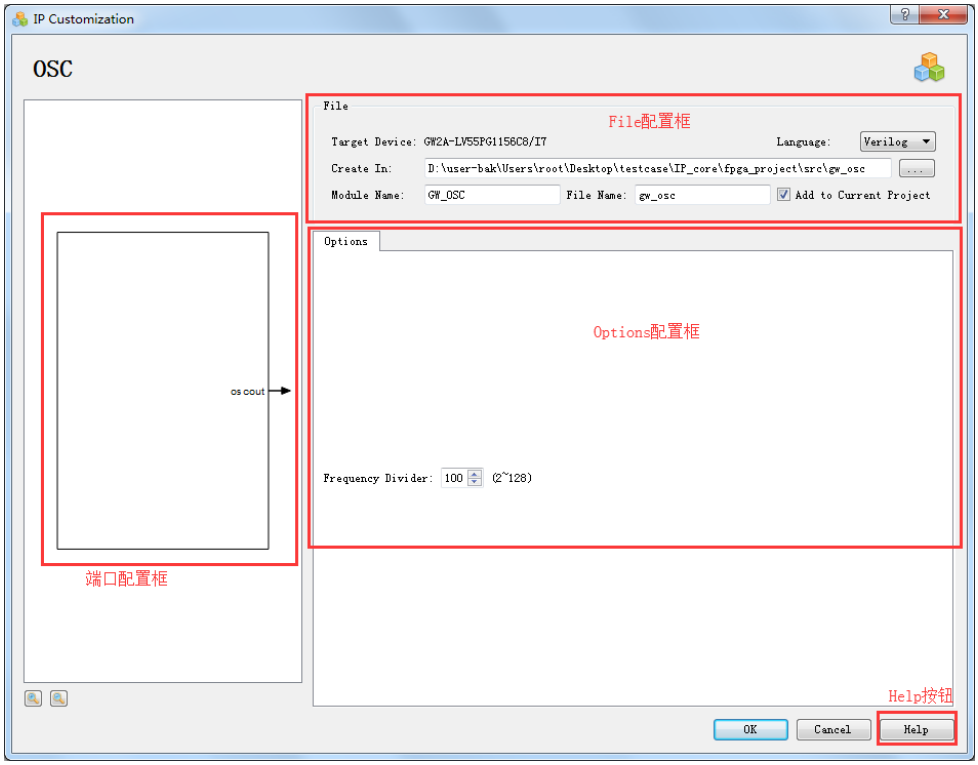
OSC is the internal clock oscillator. It has a maximum frequency of 250MHz. Click "OSC" on the IP Core Generator page. A brief introduction to the OSC will be displayed on the right of the screen, as shown in Figure 3-91.

Figure 3-91 OSC Summary



Double-click “OSC”, and the “IP Customization” window will open, as shown in Figure 3-92. This displays the File configuration, Options configuration, port configuration diagram, and the “Help” button.

Figure 3-92 OSC – IP Customization



1. File Configuration

The file configuration mainly includes the basic information related to the OSC instantiation file, as shown in Figure 3-92.

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

Options configuration mainly includes configuration information related to the OSC instantiation file, as shown in Figure 3-92.

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 3-92.

4. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-93. Help page contains the IP Core general description, and a brief introduction to the "Options".

Figure 3-93 Help

OSC

Information

Type:	OSC
Vendor:	GOWIN Semiconductor
Summary:	Internal clock generator OSC. User can change parameter FREQ_DIV to generate different output clock frequency, which can be one of 64 values: 1/2, 1/4, 1/6, 1/8, 1/10, 1/12, 1/14 ,....., or 1/128 of the oscillator frequency (about 250 MHz).

Options

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

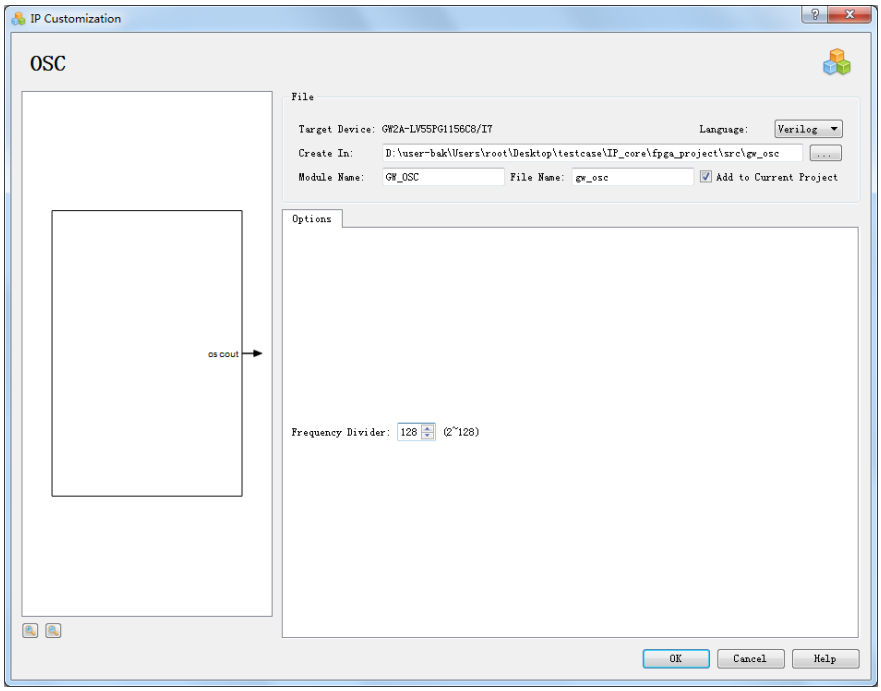
IP Generation Files

As shown in Figure 3-94, after customizing the IP, click “OK” to generate three files that are named according to the "File Name" specified in the File configuration:

- The design file for the 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 3-94 IP Customization



Design File for the Gowin Primitive OSC Instantiation

The design file for the Gowin Primitive OSC instantiation is a complete

Verilog module. OSC instantiation is generated according to the OSC configuration in "IP Customization" window, as shown in Figure 3-95.

Figure 3-95 Design File for the Gowin Primitive OSC 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 efficiency purposes, the IP Core Generator generates the template file while generating OSC design file instantiation, as shown in Figure 3-96.

Figure 3-96 Instantiation Template File for the IP Design File

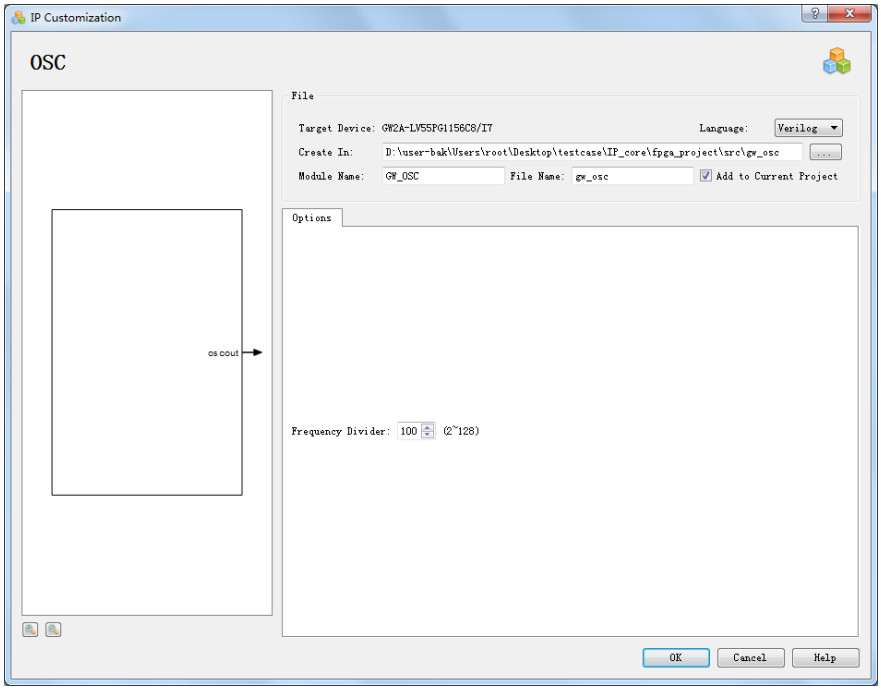
```
GW_OSC your_instance_name(
    .oscout(oscout_o) //output oscout
);
```

OSC Generation Example

Users can generate a specific OSC IP with 2.5MHz clock frequency. Take the GW1N-4-LQFP144 device for instance; the configuration page is as shown in Figure 3-97. Click "OK" to generate the customized OSC IP design files.

The generated IP files are stored in the directory specified in the directory set in the "Create in" textbox. If “Add to Current Project” is checked, the generated IP design files will be automatically added to the project.

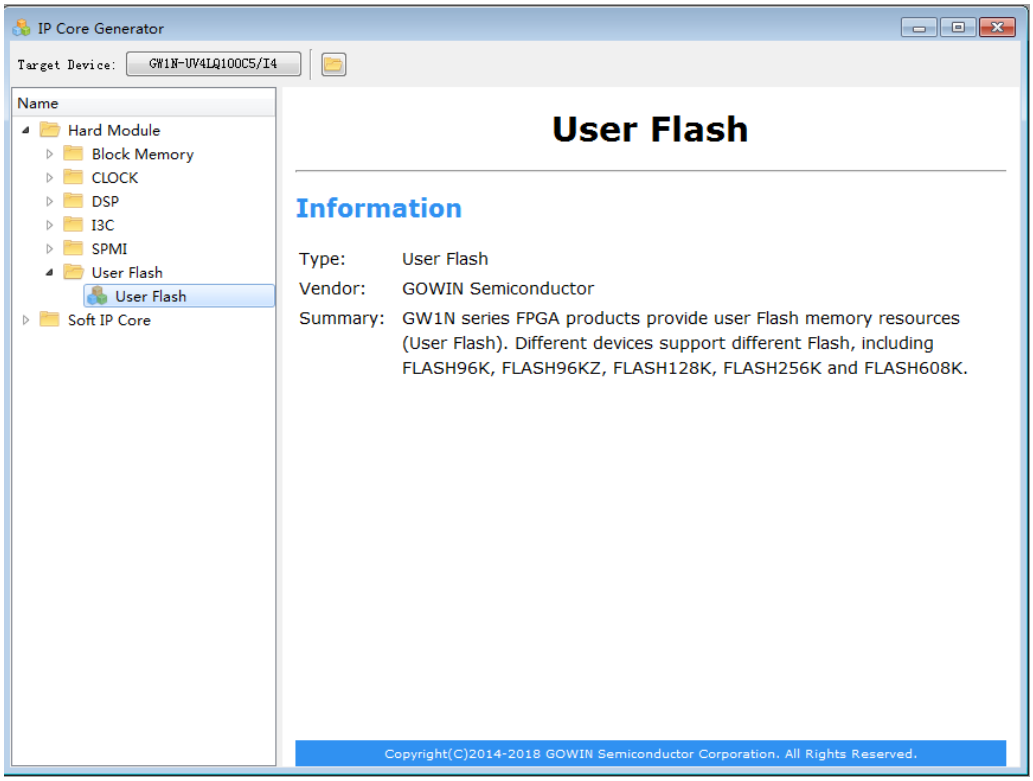
Figure 3-97 OSC – IP Customization



3.4 User Flash

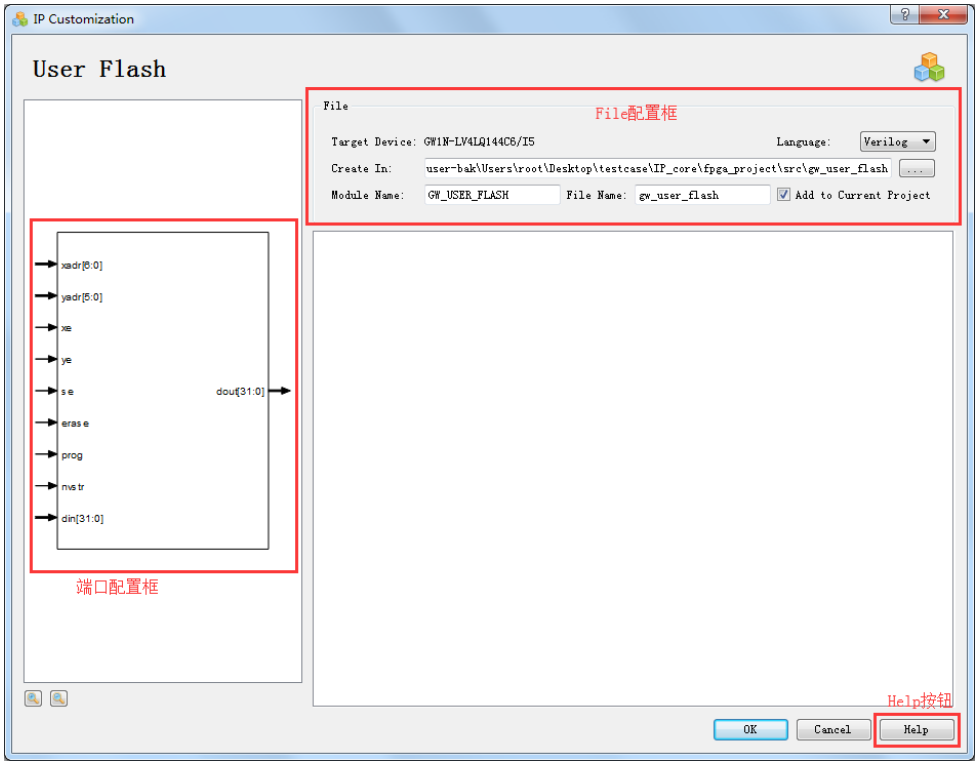
Click "User Flash" on the IP Core Generator page. A brief introduction to the User Flash will be displayed on the right of the screen, as shown in Figure 3-98.

Figure 3-98 User Flash Overview



Double-click “User Flash”, and the “IP Customization” window will open as shown in Figure 3-99. This displays the File configuration, Options configuration, port configuration diagram, and the “Help” button.

Figure 3-99 User Flash – IP Customization



1. File Configuration

The file configuration mainly includes the basic information related to the User Flash instantiation file, as shown in Figure 3-99.

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!

Currently, GW1N-1/GW1N-2/ GW1N-2B/GW1N-4/ GW1N-4B/ GW1N-6/ GW1N-6ES/ GW1N-9/ GW1N-9ES/GW1NR-4/ GW1NR-4B/ GW1NR-9/ GW1NR-9ES/GW1NS-2/ GW1NS-2C/ GW1NZ-1/GW1NSR-2C support user flash. If you select any other devices, the "OK" button in the “IP Customization” window will be grayed out, and no IP can be generated.

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 3-99.

3. Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-100. Help page contains the IP Core general description, and a brief introduction to the "Options".

Figure 3-100 Help

User Flash

Information

Type:	User Flash
Vendor:	GOWIN Semiconductor
Summary:	GW1N series FPGA products provide user Flash memory resources (User Flash), the characteristics are as follows: • 10,000 write life cycles • More than 10 years of data retention (+85 ℃) • Page erase capability: 2,048 bytes • Fast page erase / word programming operation • Clock frequency: 40MHz • Word programming time: ≤ 16μs • Page erase time: ≤120ms • Current - Read Current / Duration: 2.19mA / 25ns (VCC) & 0.5mA / 25ns (VCCX) - Program / Erase operation: 12 / 12mA (MAX)

Note

Description
• If the target device is GW1N-1 , the primitive FLASH96K will be instantiated in the customized module. For the primitive FLASH96K, data input and data output's width is 32, input XADR and YADR's width is 6. • If the target device is GW1N-2/GW1N-4/GW1NR-4 , 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.

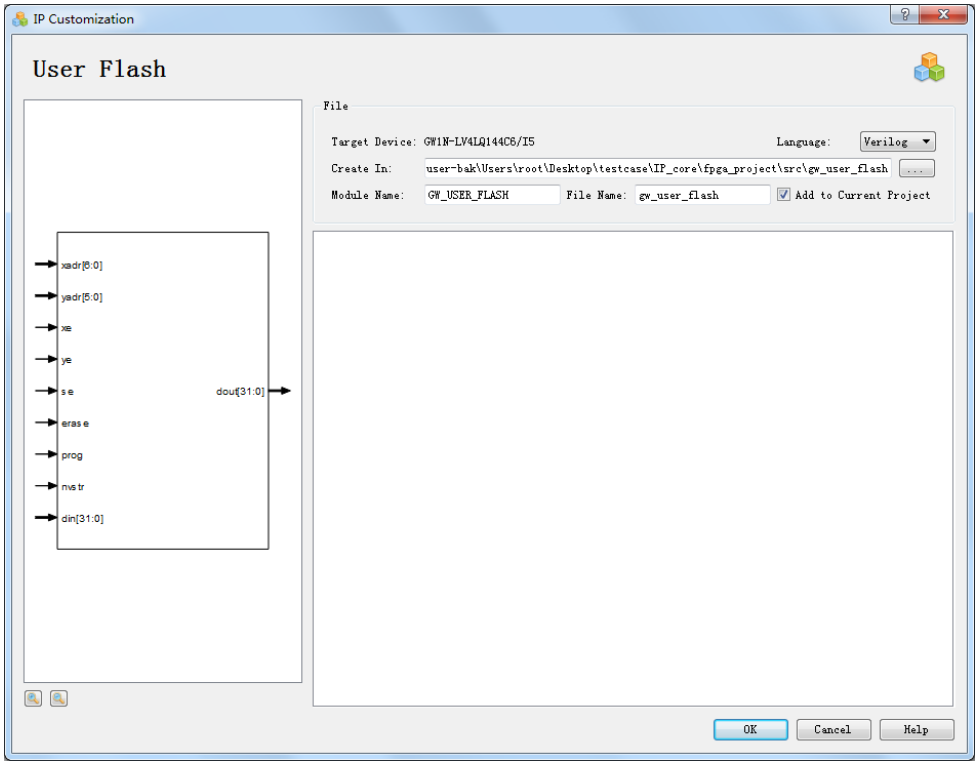
IP Generation Files

As shown in Figure 3-101, after customizing the IP, click “OK” to generate three files that are named according to the “File Name" specified in the file configuration:

- The design file for the 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 an .vhd suffix. Taking verilog for instance, the following sections introduce the generated files.

Figure 3-101 IP Customization



Design file for the Gowin Primitive User Flash instantiation

The design file for the Gowin Primitive User Flash 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 3-102. The generated GW1N-4 design files instantiation is the primitive FLASH256K.

Figure 3-102 Design file of Gowin Primitive User Flash 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 efficiency purposes, the IP Core Generator generates the template file while generating the user flash design file instantiation, as shown in Figure 3-103.

Figure 3-103 Instantiation Template File for the IP Design File

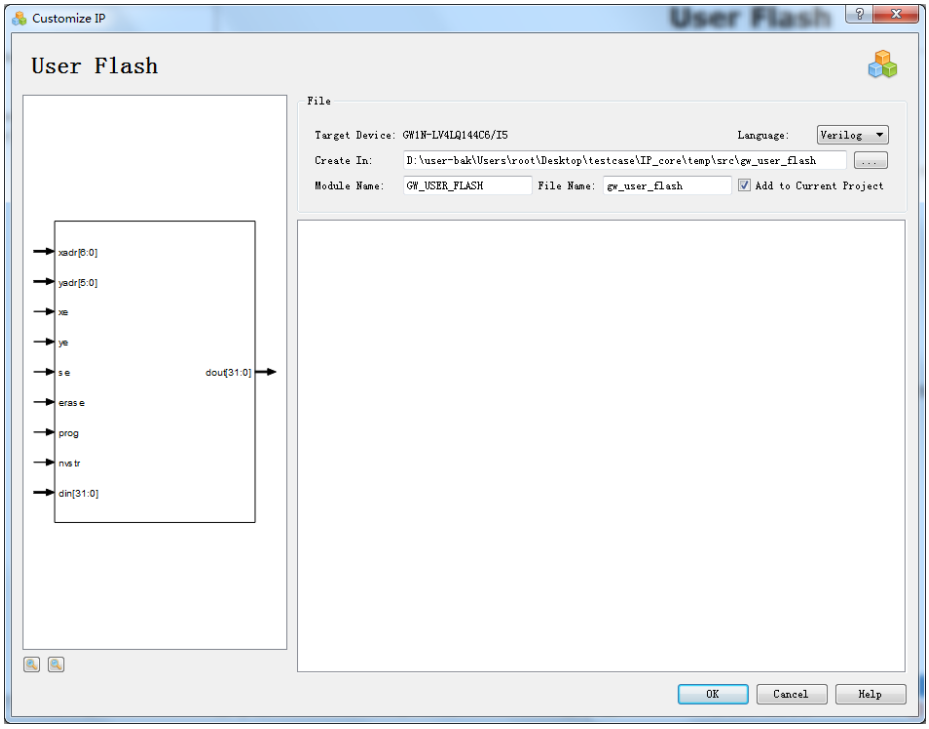
```
GW_USER_FLASH your_instance_name(
    .dout(dout_o), //output [31:0] dout
    .xe(xe_i), //input xe
    .ye(ye_i), //input ye
    .se(se_i), //input se
    .prog(prog_i), //input prog
    .erase(erase_i), //input erase
    .nvstr(nvstr_i), //input nvstr
    .xadr(xadr_i), //input [6:0] xadr
    .yadr(yadr_i),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

User Flash Generation Example

As shown in Figure 3-104, take FLASH256K generation supported by GW1N-4 for instance, select the GW1N-4 device, and select the package as required in the “IP Customization” window. Then click "OK" to generate the customized User Flash IP design files.

The generated IP files are stored in the directory set in "Create in". If “Add to Current Project” is checked, the generated IP design files will be automatically added to the project.

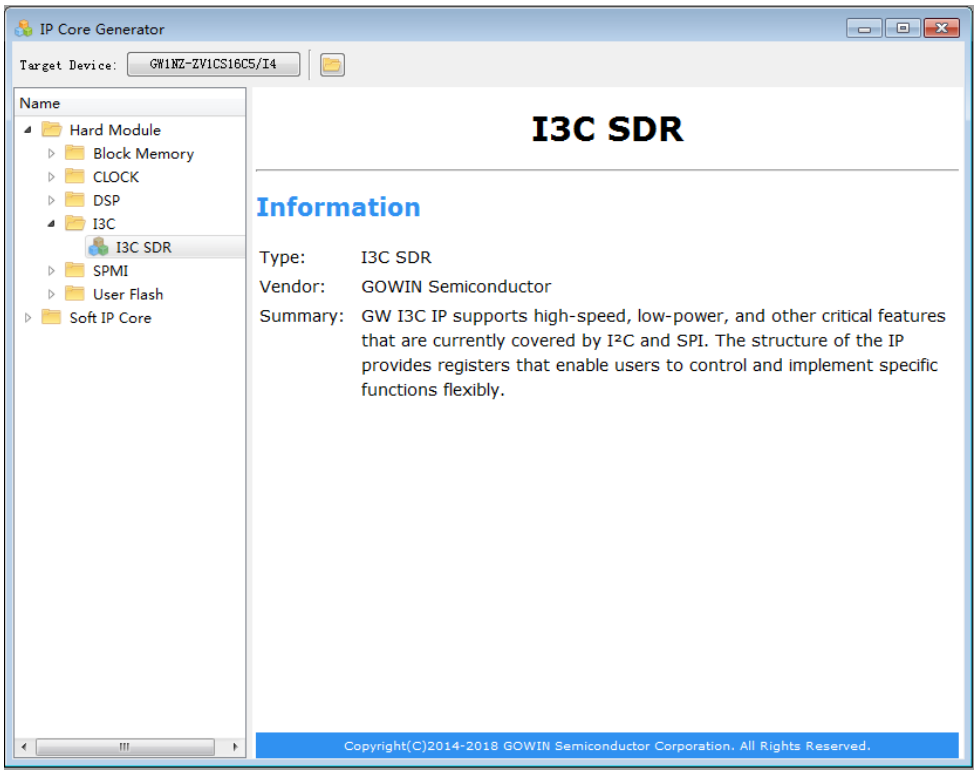
Figure 3-104 User Flash – IP Customization



3.5 I3C

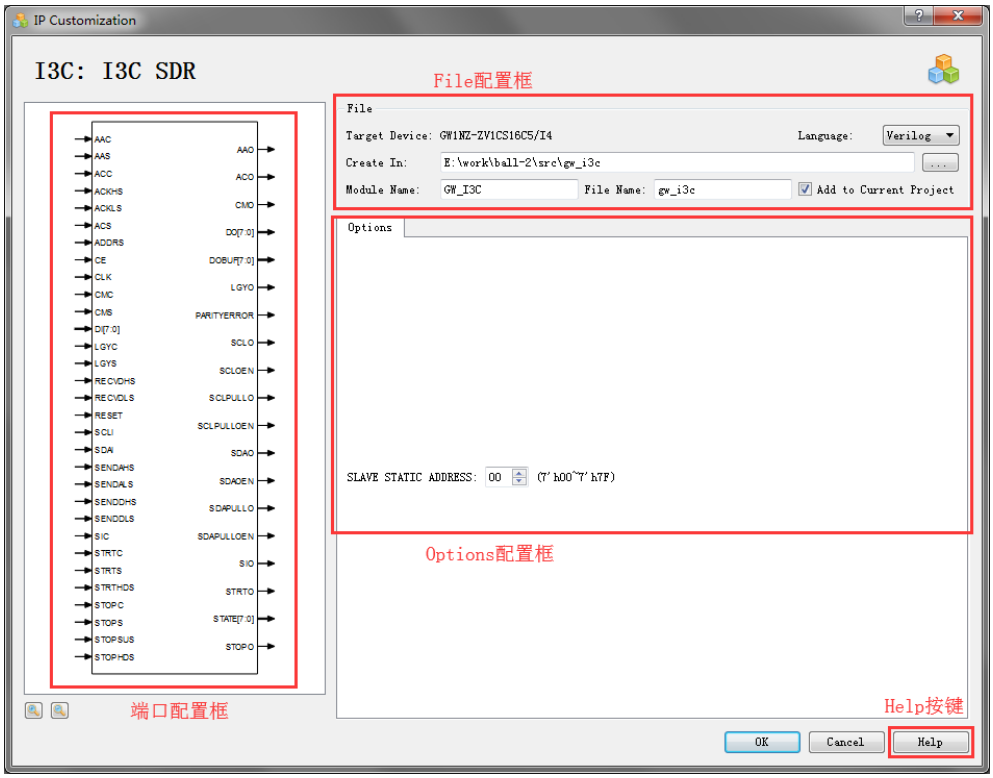
I3C IP offers the features of high-speed and low power and be compatible with the other key features of I2C and SPI. The I3C IP provides registers for users to control and realize specific functions. Click "I3C > I3C SDR" on the "IP Core Generator" page. A brief introduction to the I3C will be displayed on the right of the screen, as shown in Figure 3-105.

Figure 3-105 I3C SDR Information



Double-click “I3C SDR”, and the “IP Customization” window will open as shown in Figure 3-106. This displays the File configuration, Options configuration, port configuration diagram, and the “Help” button.

Figure 3-106 I3C Customization



1. File Configuration

The file configuration mainly includes the basic information related to the I3C instantiation file, as shown in Figure 3-106.

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 3-106.

3. Options Configuration

Options configuration mainly includes the configuration information of I3C instantiation file, as shown in Figure 3-106.

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

4. Hlep

Click "Help" to open IP Core configuration information, as shown in Figure 3-107. Help page contains the IP Core general description, and a brief introduction to the "Options".

Figure 3-107 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.

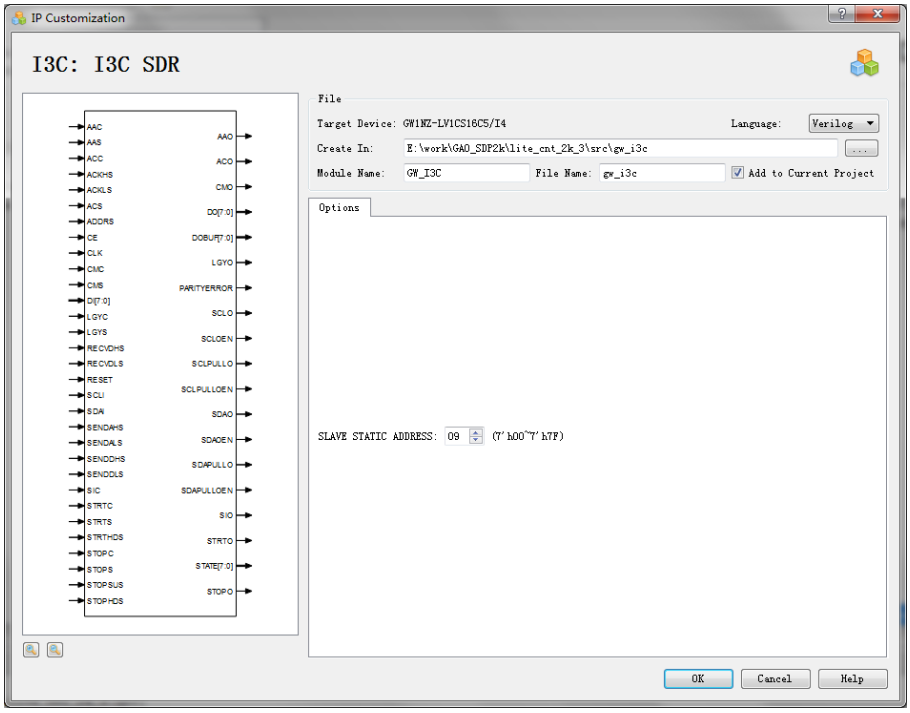
IP Generation Files

As shown in Figure 3-108, after customizing the IP, click “OK” to generate three files that are named according to the "File Name" specified in the File configuration:

- The design file for 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_ user_flash.ipc”.

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

Figure 3-108 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 "Customize IP" window, as shown in Figure 3-109. The generated GW1NZ-1 design files instantiation is the primitive I3C hardcore.

Figure 3-109 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 efficiency purposes, the IP Core Generator generates the template

file while generating I3C design file instantiation, as shown in Figure 3-11010.

Figure 3-110 Instantiation template file for the IP design file

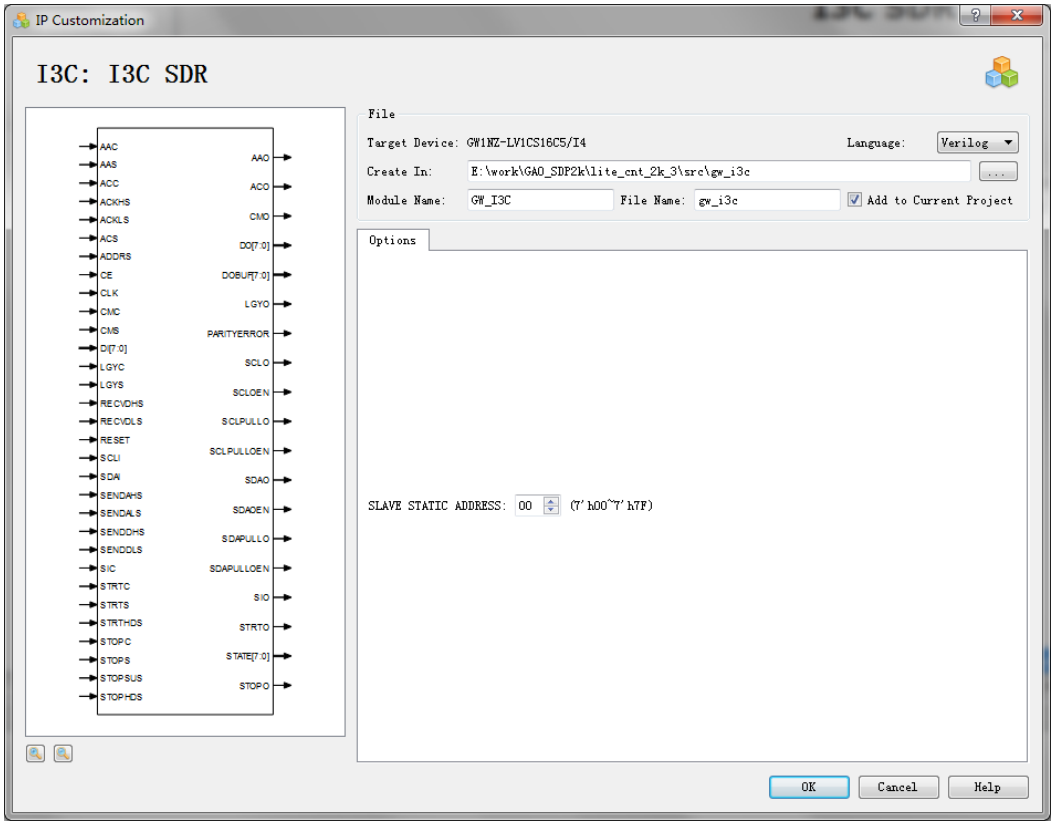
```
GW_I3C your_instance_name(
    .lgyo(lgyo_o), //output lgyo
    .cmo(cmo_o), //output cmo
    .aco(aco_o), //output aco
    .aao(aao_o), //output aao
    .sio(sio_o), //output sio
    .stopo(stopo_o), //output stopo
    .strto(strto_o), //output strto
    .parityerror(parityerror_o), //output parityerror
    .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 3-111, take I3C generation supported by GW1NZ-1 for instance, select the GW1NZ-LV1CS16C5/I4 device and select the package as required and configure the "Options" in the "IP Customization" window. Then click "OK" to generate the customized I3C IP design files.

The generated I3C IP files are stored in the directory specified in the directory set in the "Create in" textbox. If “Add to Current Project” is checked, the generated IP design files will be automatically added to the project.

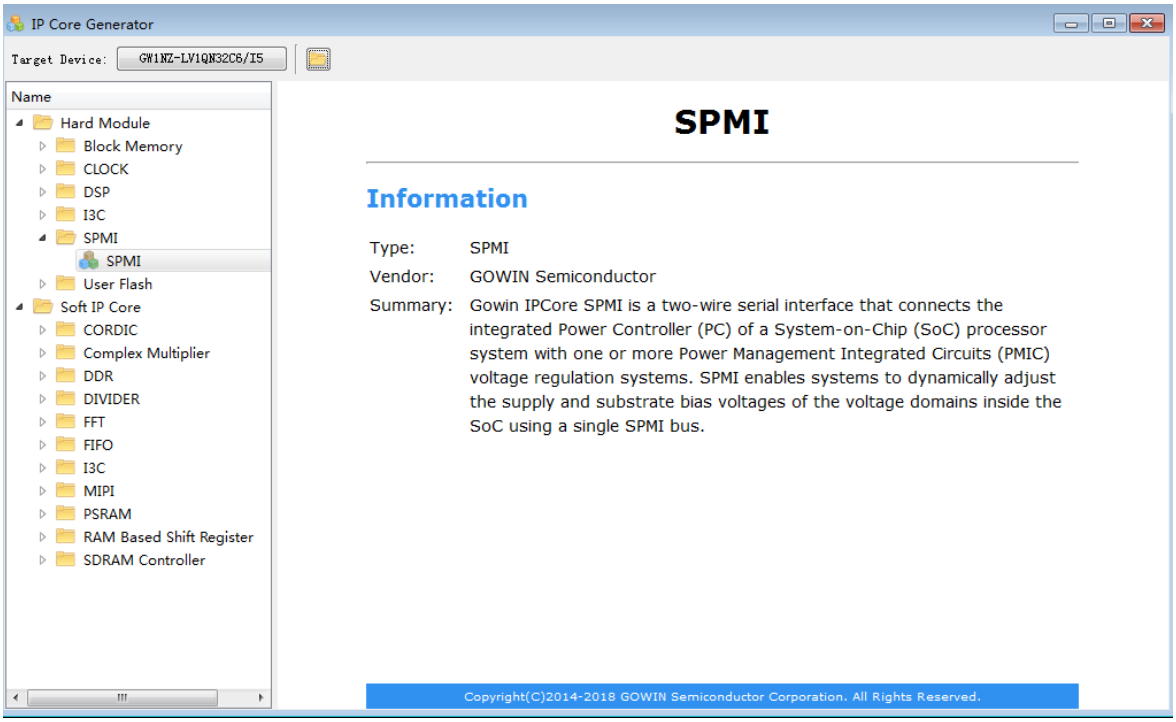
Figure 3-111 I3C IP Customization



3.6 SPMI

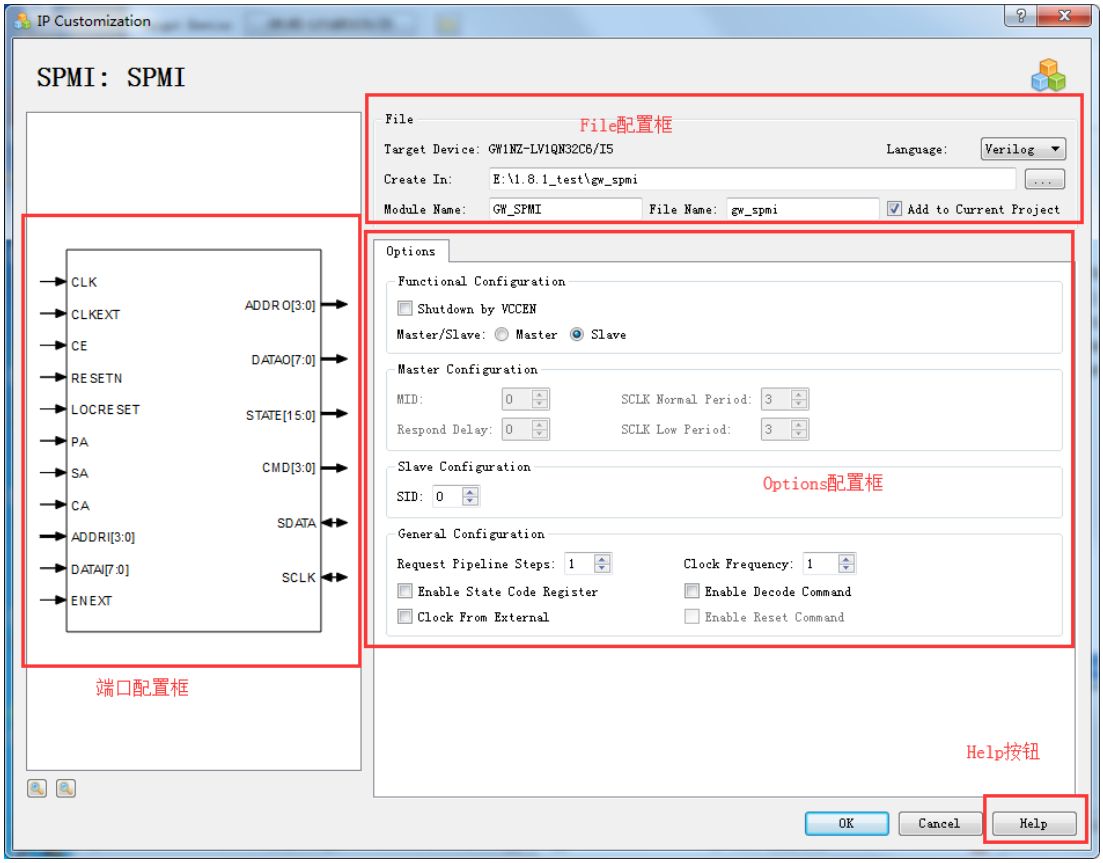
SPMI is a two-wire serial interface that connects the integrated Power Controller (PC) of a System-on-Chip (SoC) processor system with one or more Power Management Integrated Circuits (PMIC) Voltage regulation systems. SPMI enables systems to dynamically adjust the supply and substrate bias voltages of the voltage domains inside the SoC using a single SPMI bus. Click "SPMI" on the IP Core Generator page. A brief introduction to the SPMI will be displayed on the right of the screen, as shown in Figure 3-112.

Figure 3-112 SPMI Information



Double-click on the "SPMI" to open the "IP Customization" window. This displays the file configuration, options configuration, port configuration diagram, and the “Help” button, as shown in Figure 3-113.

Figure 3-113 SPMI Customization



File Configuration

The file configuration mainly includes the basic information related to the SPMI instantiation file, as shown in Figure 3-113.

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.

Options Configuration

Options configuration mainly includes the configuration information of SPMI instantiation file, as shown in Figure 3-113.

Functional Configuration:

- Shutdown by VCCEN**:** Shutdown by the "VCCEN" external pin. If this option is checked, the communication function of SPMI will be disabled.
- Master/Slave: Set SPMI as Master or Slave.
- Master Configuration:
- MID: Set the identifier of the master server of SPMI.
- Respond Delay: Set the response delay time.
- SCLK Normal Period: Set SCLK period in normal mode.
- SCLK Low Period: Set the SCLK period as sleep mode.
- Slave Configuration:
- SID: Set the identifier 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.

Ports Configuration

Ports configuration diagram displays the current IP Core configuration result, as shown in Figure 3-113.

Help

Click "Help" to open the IP Core configuration information, as shown in Figure 3-114.

Figure 3-114 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".

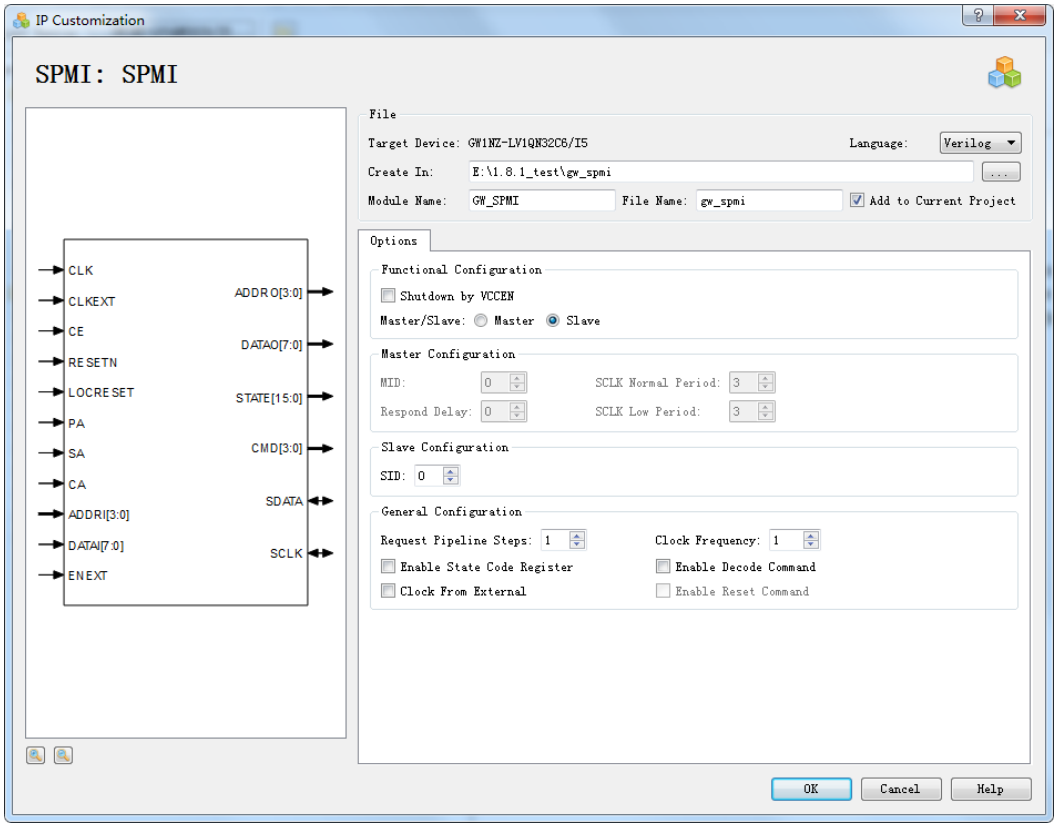
IP Generation Files

As shown in Figure 3-115, after customizing the IP, click “OK” to generate three files that are named according to the "File Name" specified in the File configuration:

- Design file for 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 3-115 SPMI Customization



SPMI Design File Instantiation

The design file for the Gowin Primitive SPMI 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 3-116.

Figure 3-116 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 efficiency purposes, the IP Core Generator generates the template file while generating SPMI design file instantiation, as shown in Figure 3-117.

Figure 3-117 Instantiation Template File for the IP Design File

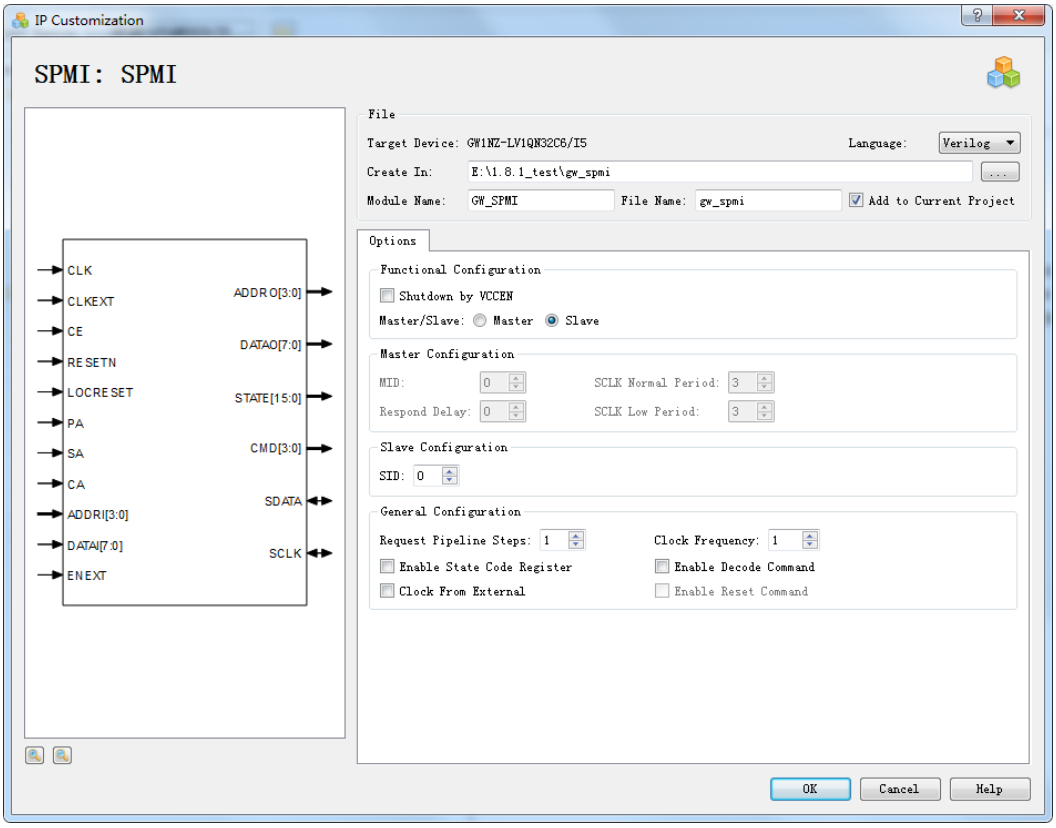
```
GW_SPMI your_instance_name(
    .addro(addro_o), //output [3:0] addro
    .datao(datao_o), //output [7:0] datao
    .state(state_o), //output [15:0] state
    .cmd(cmd_o), //output [3:0] cmd
    .sdata(sdata_io), //inout sdata
    .sclk(sclk_io), //inout sclk
    .clk(clk_i), //input clk
    .ce(ce_i), //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 3-118, take SPMI generation supported by GW1NZ-1 for instance, select the GW1NZ-LV1QN32C6/I5 device and select the package as required and configure the "Options" in the "IP Customization" window. Then click "OK" to generate the customized SPMI IP design files.

The generated IP files are stored in the directory specified in the directory set in the "Create in" textbox. If “Add to Current Project” is checked, the generated IP design files will be automatically added to the project.

Figure 3-118 SPMI IP Customization



GOWIN

PROGRAMMING FOR FUTURE