Machine Translated by Google

TangDynasty

Software Manual

Version number: 5.0.3

2021.03

Shanghai Anlu Information Technology Co., Ltd.

Machine Translated by Google

Anlu Company publishes this user manual document only for TD software based on Anlu FPGA/CPLD devices.

users, and other individuals shall not reproduce, distribute in any form without written consent of Anlu Company,

Republishing, downloading, displaying, any means and forms including but not limited to: electronic, mechanical, reproduction

printing, recording, etc. Anlu Corporation assumes no responsibility for anyone's use of this document. Anlu Company

The right to make changes to this document at any time without notice is reserved. Anlu Company for

The correction of errors in the documentation and the updating of the documentation after correction is not undertaken under any obligation. Anlu company in technology

No responsibility for technical support and information assistance that may be provided.

The copyright of this document belongs to Shanghai Anlu Information Technology Co., Ltd. All trademarks are

company assets.

Machine Translated by Google

content

Machine Translated by Google

Machine Translated by Google

Machine Translated by Google

Machine Translated by Google

Machine Translated by Google

Machine Translated by Google

1 Start the software

software requirements

Users need to install the following software in order to use this guide:

ÿ TD

Operating system requirements for TD to run under Linux:

ÿ Red Hat Enterprise 6.0 and above

Operating system requirements for TD to run under Windows:

ÿ Windows 7 Service Pack 1 and above

hardware requirements

The user's computer hardware requires the following configuration:

ÿ Processor: Above 2GHz

ÿ Memory: 8GB or more

ÿ Hard disk: more than 600M free space

Install and uninstall **TD**

To install TD, please double-click the msi file in the TD installation disk and follow the installation steps to complete the installation

During the installation process, if you are reminded to install vc_redist.exe, please install it according to the prompt, and it will continue after installation.

Install the TD software until the installation is complete. If you cannot successfully install vc_redist.exe, please update the Windows patch.

To uninstall TD, please click StartÿControl PanelÿSelect TDÿUninstall

Machine Translated by Google

Start the **TD** software

Operating system requirements for TD to run under Windows:

ÿ Windows7 and above

Start ÿ All Programs ÿ TD ÿ td.exe

Start TD under Linux:

Before starting TD, set the ld library path:

eg setenv LD_LIBRARY_PATH **$(td_install_path)/lib**

ÿ Interface mode: /td_install_dir/td-gui

ÿ Command mode: /td_install_dir/td

Start the TD CMD window under Windows:

Start ÿ All Programs ÿ td_commands_prompt

get help

Users can email support@anlogic.com for assistance with TD software and related tools.

Machine Translated by Google

2 Project Management

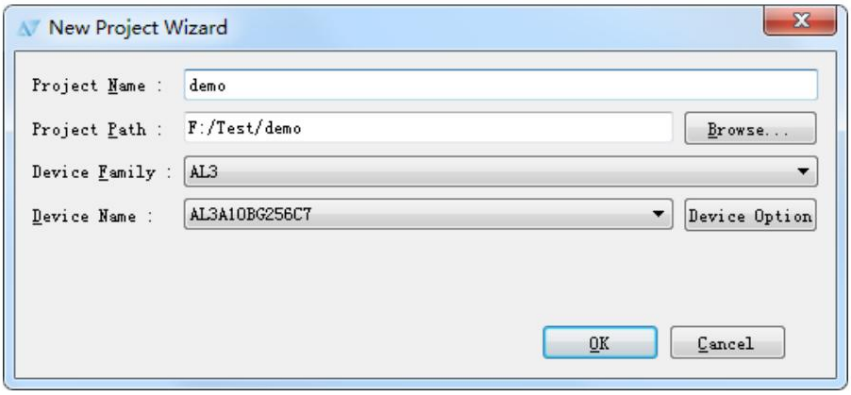
2.1 Create a new project

Create new project:

1. Select **Project ÿ New Project...** A new project dialog box will pop up

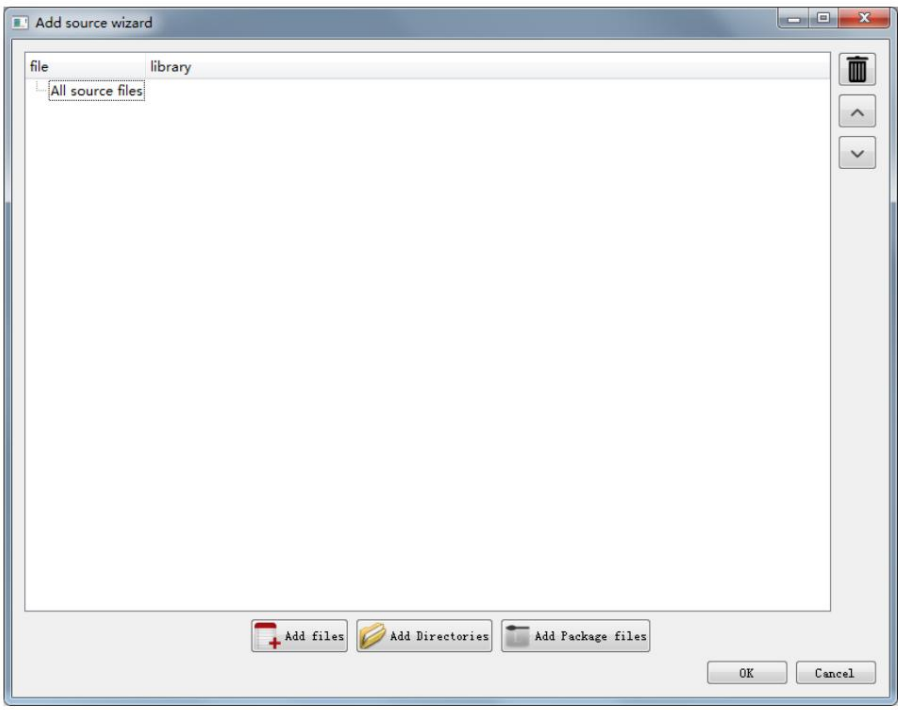
2. Specify the storage path of the created project and enter the project name

3. Select **Device Family** and **Device** Name, the defaults are AL3 and AL3A10BG256C7.



Add source files:

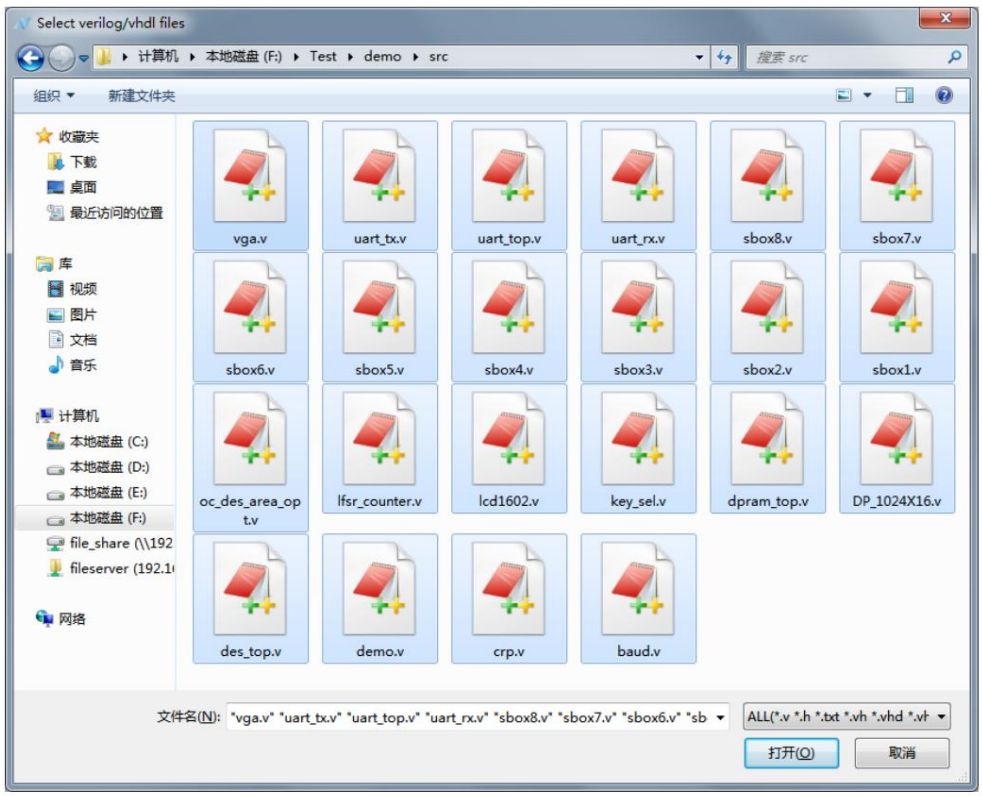
1. Select **Source ÿ Add Source...**

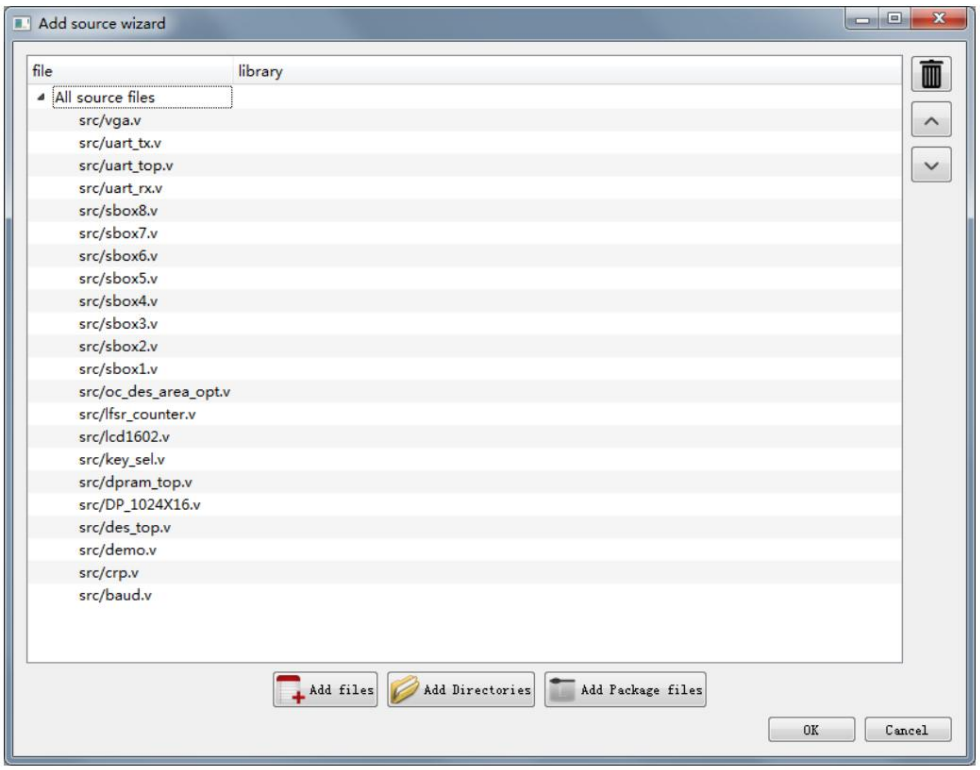


Machine Translated by Google

2. The Add Source Wizard interface supports adding files, folders and custom VHDL library files.

a. Click **Add files to enter the** Select verilog/vhdl files interface to select the HDL source files to be added file, click Open.

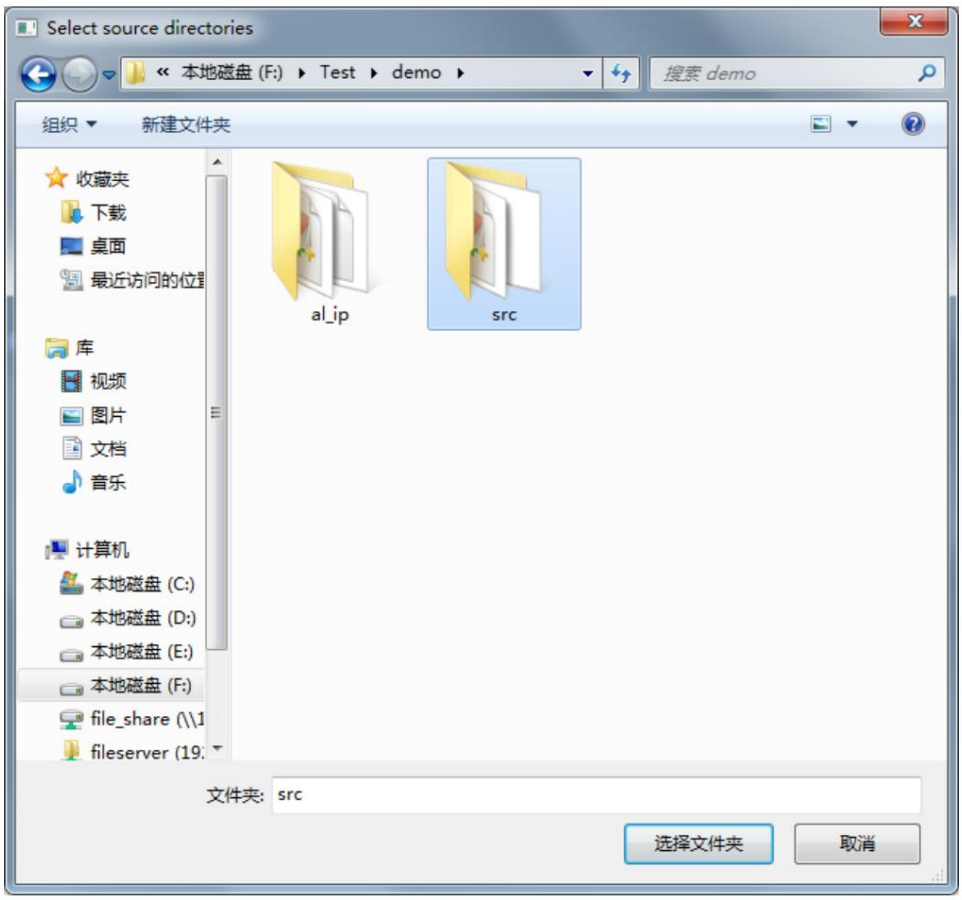


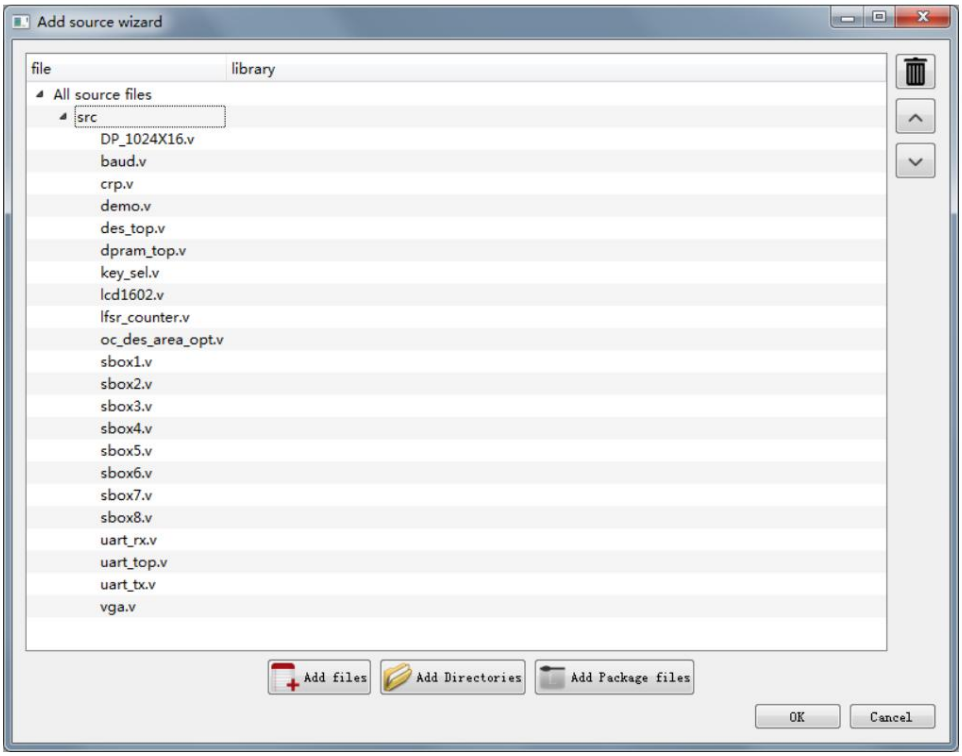


Machine Translated by Google

b. Click **Add Directories to enter the** Select source directories interface and select the files to be added.

Folder, clicking Select Folder will automatically add all the HDL source files contained in the folder.



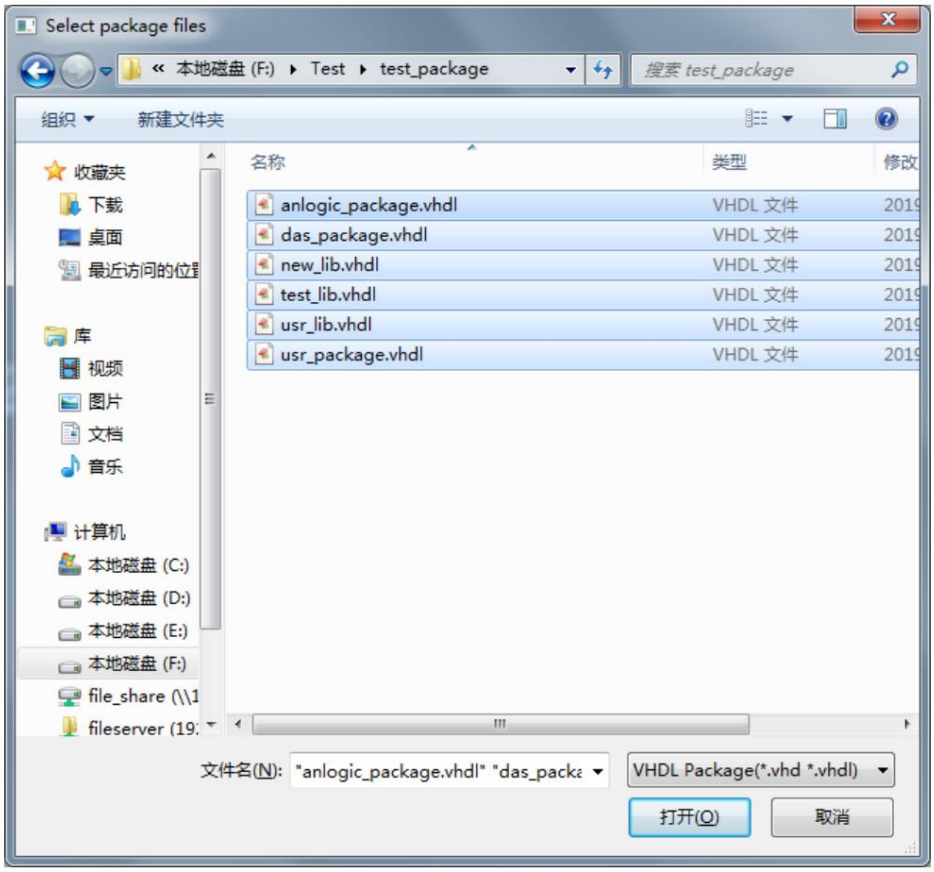


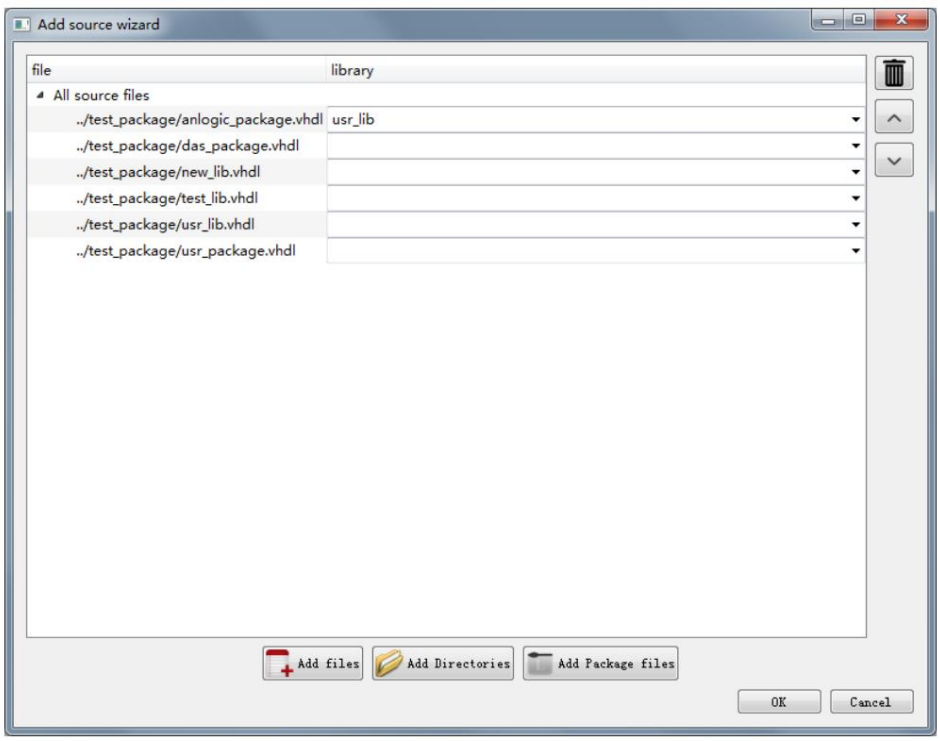
Machine Translated by Google

c. Click **Add Package files to enter the** Select package files interface to select the VHDL to be added

library file, click Open. Defined in the Add source wizard interface or selected from the drop-down menu

The library name to which the package belongs.





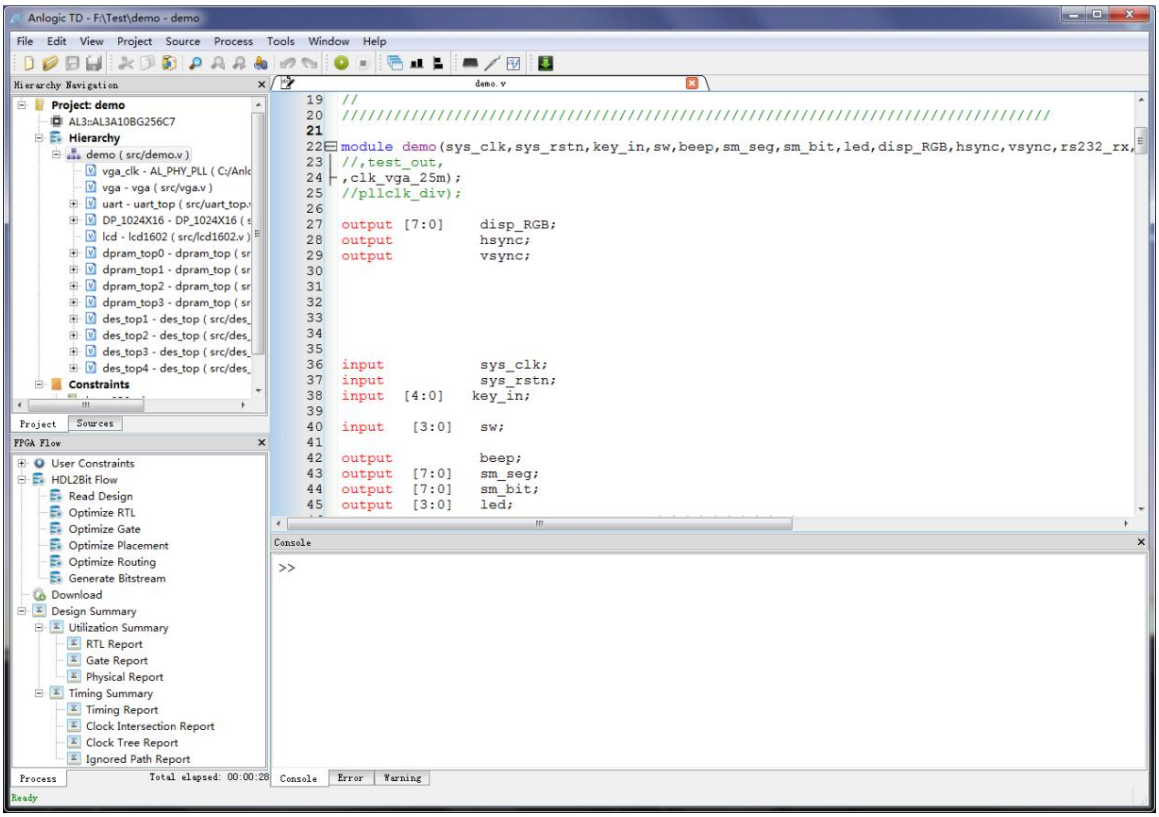
Machine Translated by Google

Click the three icons on the right to delete, move up, and move down the selected single or multiple files respectively;

Click OK to complete **Add Source…**

3. At this point, all the added source files are visible in Hierarchy and can be opened by double-clicking.

Package files can be viewed in the Sources section of the Hierarchy Navigation.



Create source files:

1. Select **Source ÿ New Source...**

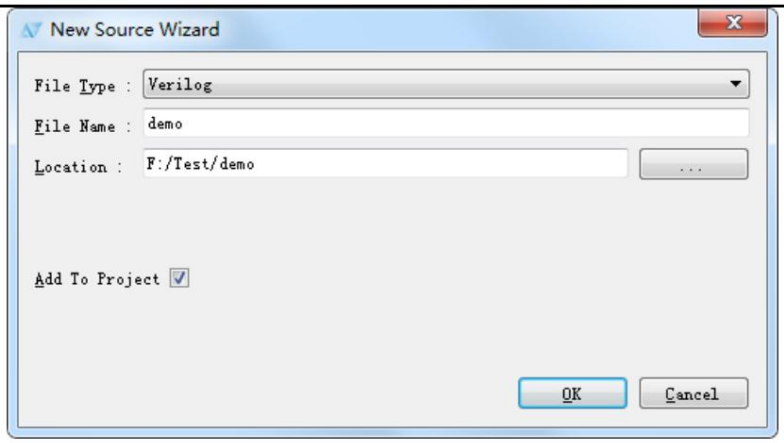
2. The source file type is Verilog by default.

3. Enter the **File Name.**

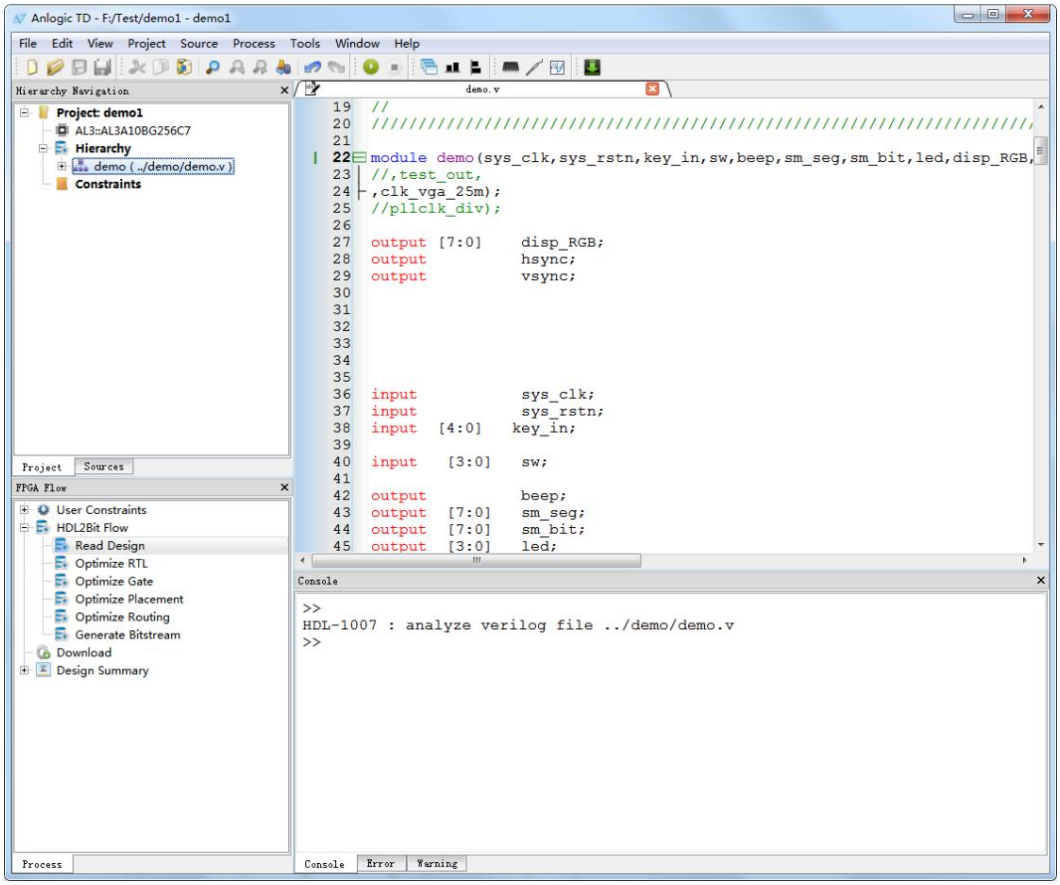
4. Make sure **Add To Project** is checked .

5. Click OK to complete the creation

Machine Translated by Google



6. Enter the file content and select **File ÿ Save** to save the file



Set the top-level module

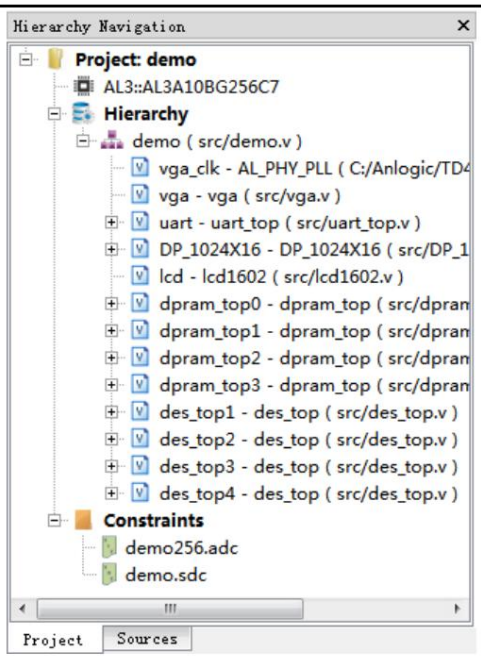
Manually setting the top-level module is optional. If no top-level module is set, the TD software will automatically analyze the module's

Hierarchy selects the topmost module.

In the Hierarchy Navigation window, right-click the row of the target module, select **Set As Top, and then**

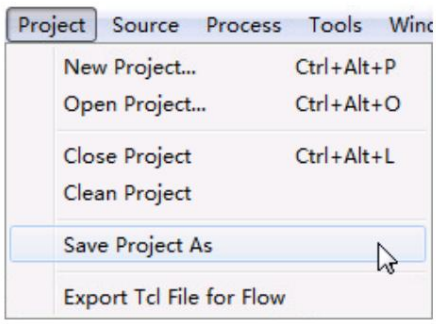
A purple top-level module marker will appear before the target module.

Machine Translated by Google



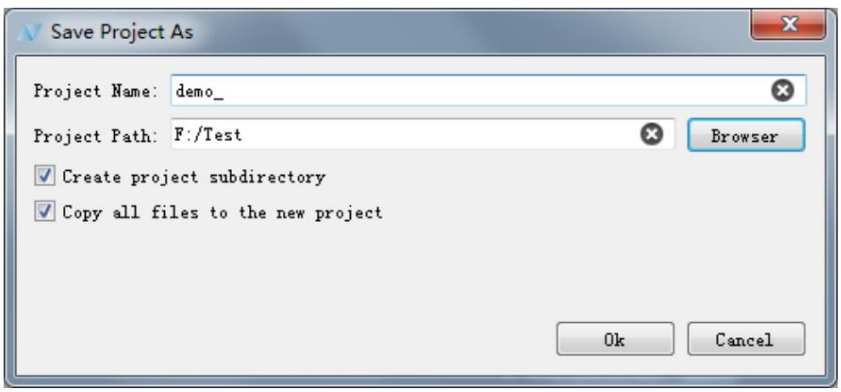
Save the project as:

1. Select **Project ÿ Save Project As**



2. Enter **Project Name** and select **Project Path,** Create project subdirectory and Copy all

files to the new project are checked by default



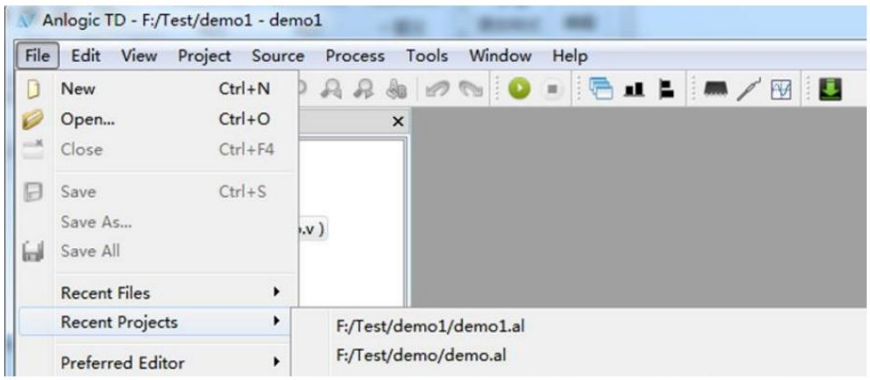
3. Click OK to save the project as, and view the corresponding project file in the selected path

Machine Translated by Google

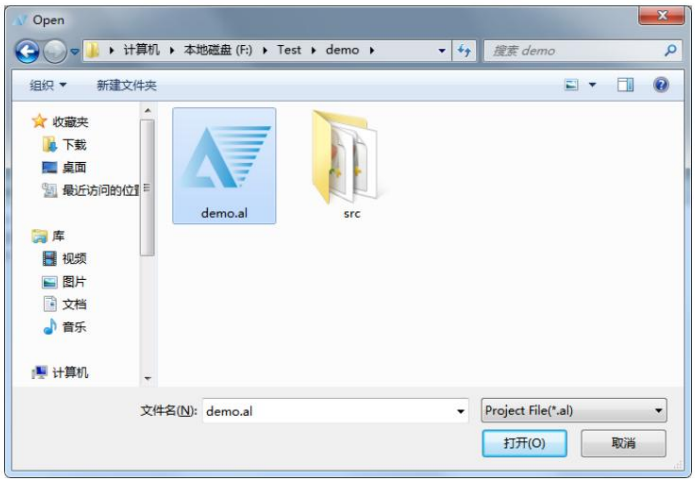
2.2 Open the project

TD will keep the opened projects and files for the user according to the order in which the projects are opened.

File ÿ Recent Projects and **File ÿ Recent Files** open previously opened projects or files.



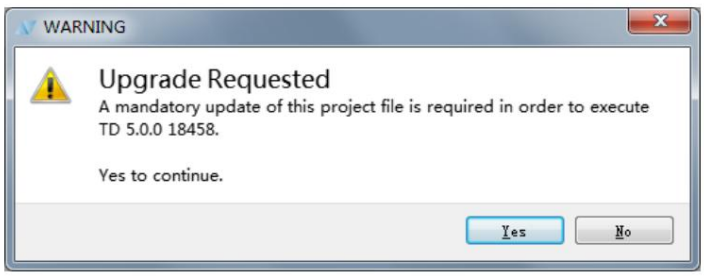
Users can also open an existing project by selecting the .al file via **Project ÿ Open Project .**



*If the project is an .al file generated with a version before TD5.0, it needs to be opened with TD5.0 and later versions.

The .al file must be upgraded, and once upgraded, the .al file cannot be opened with the version before TD5.0, otherwise it will be

An error occurred in the .al file that prevented the project from opening properly.



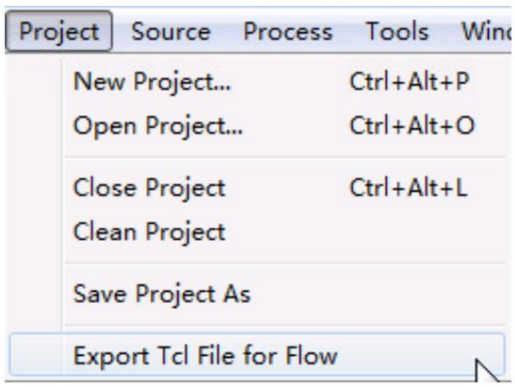
Machine Translated by Google

2.3 Export **tcl** script

The TD software supports running Flow using tcl scripts, which reduces user interface operations.

Click **Project ÿ Export Tcl File for** Flow, the prj_name.tcl file will be generated in the project directory,

This file records all commands for the last operation of Flow.



For example, the following operations are performed on the interface:

1. Open the project demo.al

2. Set the parameter Optimize RTL rtl_sim_model ON

3. Run HDL2Bit Flow

4. Export the tcl script demo.tcl

demo.tcl

```
1  import_device al3_10.db -package BGA256
2  open_project demo.al
3  elaborate -top demo
4  read_adc "demo256.adc"
5  optimize_rtl
6  write_verilog "simulation/demo_rtl_sim.v"
7  report_area -file "demo_rtl.area"
8  read_sdc "src/demo.sdc"
9  export_db "demo_rtl.db"
10 optimize_gate -maparea "demo_gate.area"
11 legalize_phy_inst
12 read_sdc "src/demo.sdc"
13 export_db "demo_gate.db"
14 place
15 route
16 report_area -io_info -file "demo_phy.area"
17 export_db "demo_pr.db"
18 start_timer
19 report_timing -mode FINAL -net_info -ep_num 3 -path_num 3 -file "demo_phy.timing"
20 bitgen -bit "demo.bit" -version 0X00 -g ucode:010011000000000000000000
```

Machine Translated by Google

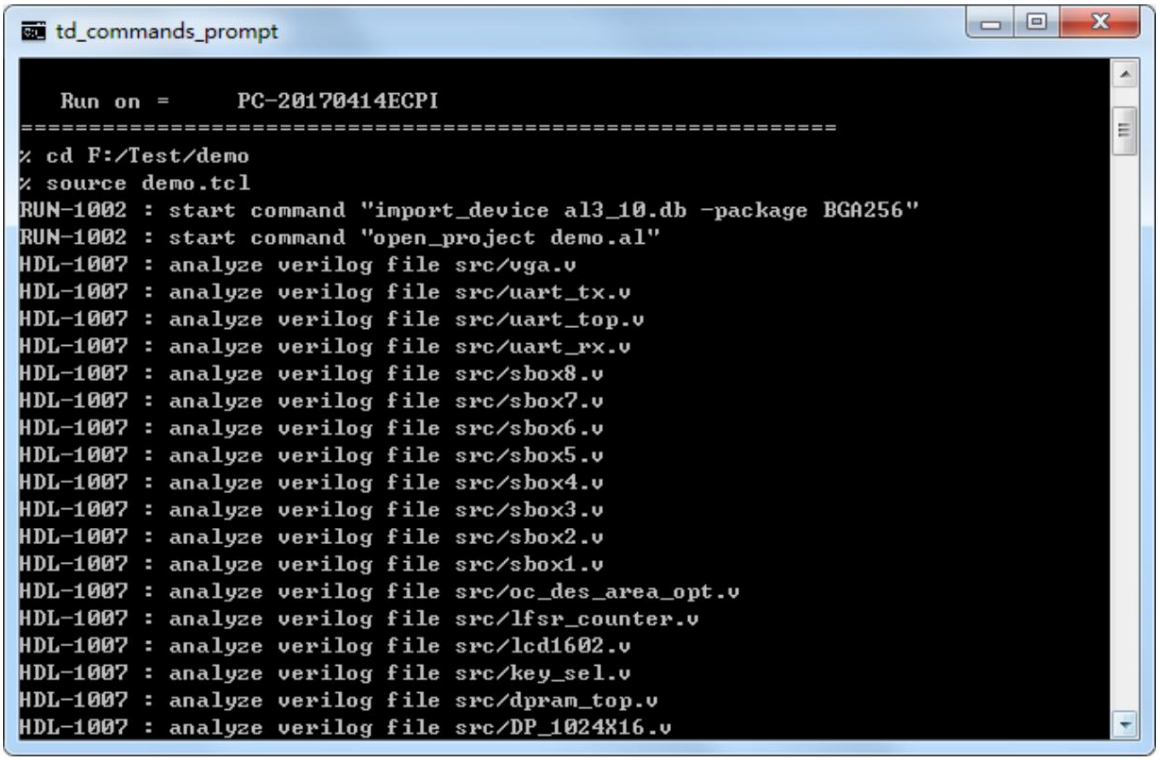
The steps to execute the tcl script are:

1. Launch the TD CMD window under Windows:

Start ÿ All Programs ÿ td_commands_prompt

2. Enter the directory where the project is located

3. Execute the command: source demo.tcl.

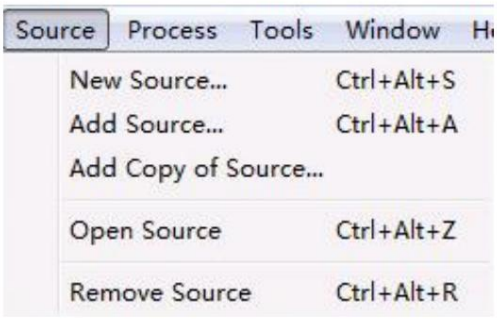


Machine Translated by Google

2.4 Source file management

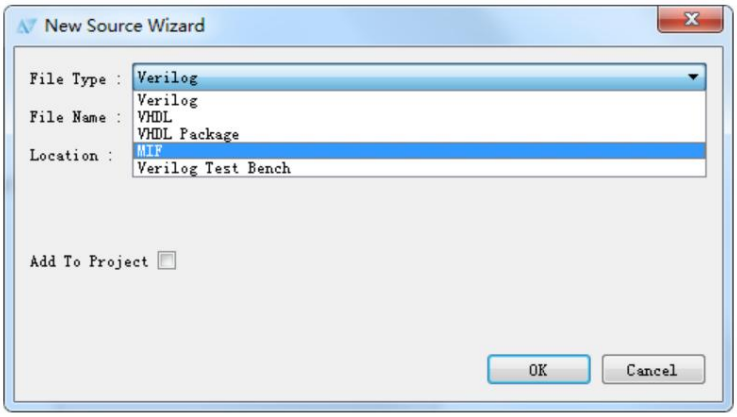
2.4.1. New file

1. Source ÿ New Source

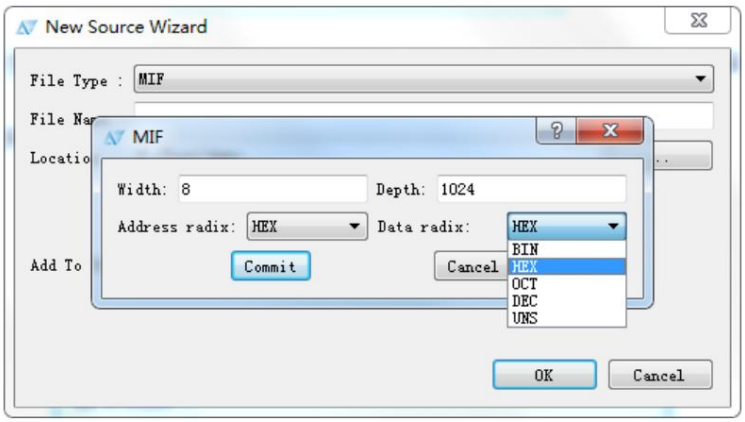


2. Select the type of generated file: Verilog, **VHDL, VHDL Package, MIF, Verilog Test Bench,**

Enter a file name, select a file path, and choose whether to add it to the project.



3. When the selected type is **MIF ,** the following configuration interface will appear:



Enter the width and depth of the MIF file, select the base of data and address, and the generated MIF file is as follows:

Machine Translated by Google

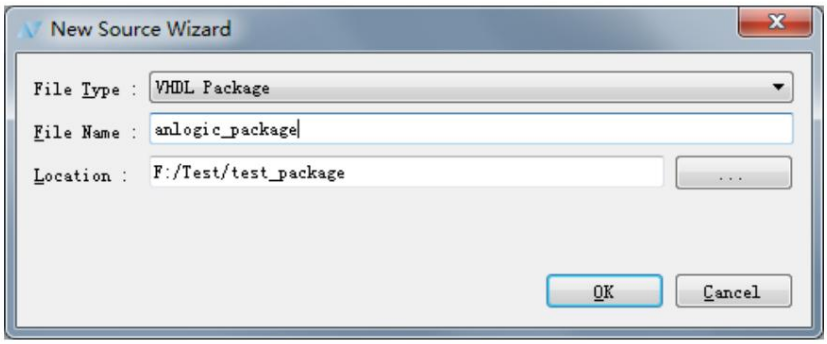

```
bram8x1024.mif
1  DEPTH        =   1024;
2  WIDTH        =   8;
3  ADDRESS_RADIX    =   HEX;
4  DATA_RADIX   =   HEX;
5  CONTENT      BEGIN
6  000      :00;
7  001      :01;
8  002      :02;
9  003      :03;
10 004      :04;
11 005      :05;
12 006      :06;
13 007      :07;
14 008      :08;
15 009      :09;
16 00a      :0a;
17 00b      :0b;
18 00c      :0c;
19 00d      :0d;
20 00e      :0e;
21 00f      :0f;
22 [010..3ff]   :00;
23 END;
```

2.4.2 Create **VHDL Package**

TD software supports users to use custom VHDL library files, and commonly used entities can be stored in the VHDL library and package. The specific operations are as follows:

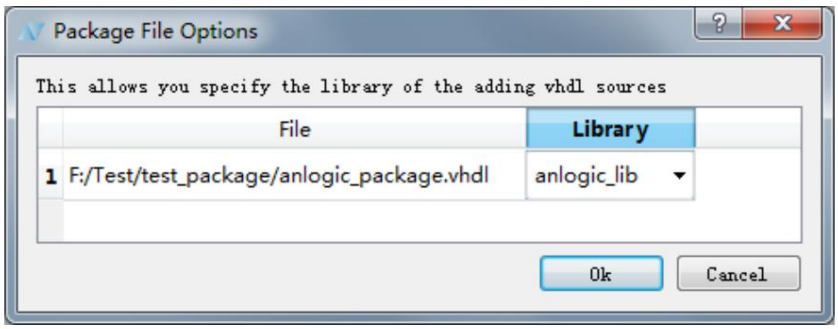
1. Source ÿ New Source

Select VHDL Package, fill in the File Name, and select the folder where the file is located.



After clicking OK, a dialog box as shown below will pop up to define the Library name to which the package belongs.

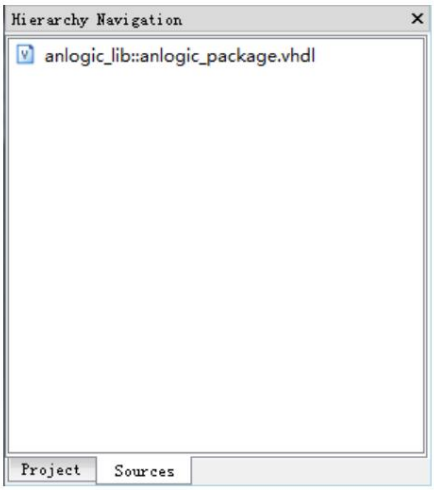
say. If multiple libraries are defined in the project, they can be selected through the drop-down menu.



Machine Translated by Google

Continue to click OK and a new anlogic_package.vhdl will be created and added by default to

under construction.



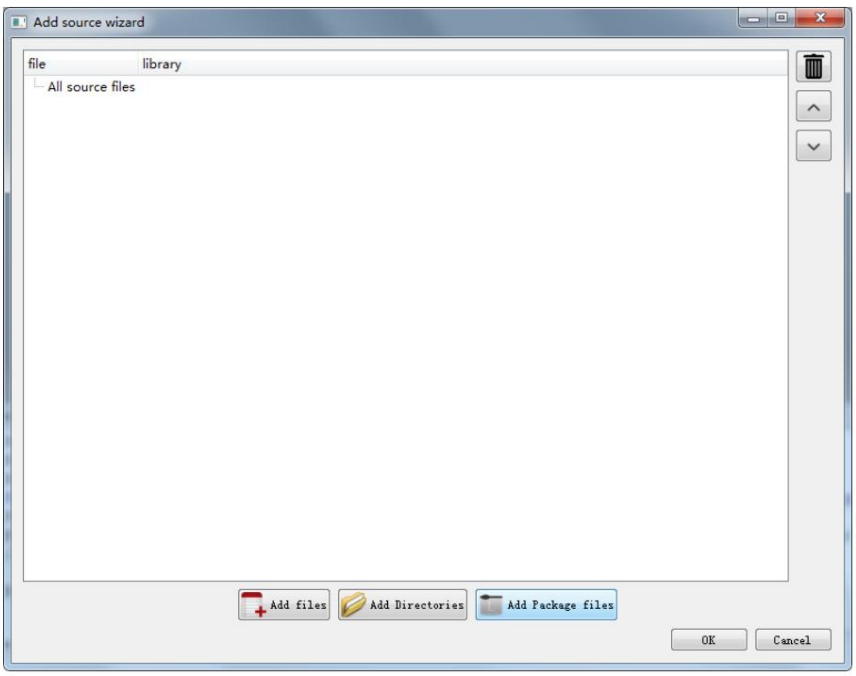
The Library can be used after the related entities are defined in anlogic_package.vhdl.

```
21  library IEEE;
22  use IEEE.std_logic_1164.all;
23  library anlogic_lib;
24  use anlogic_lib.anlogic_package.ALL;
25
```

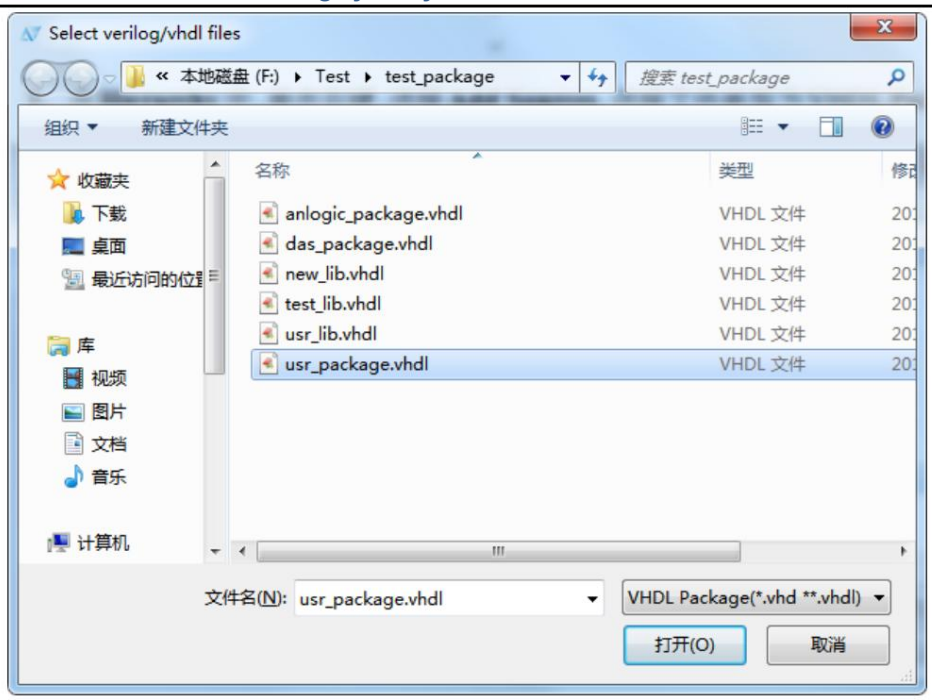
2. In **Hierarchy ,** right-click, select **Add Sources, under the** Add source wizard interface

Select Add Package files, select an existing vhdl file and open it to create a

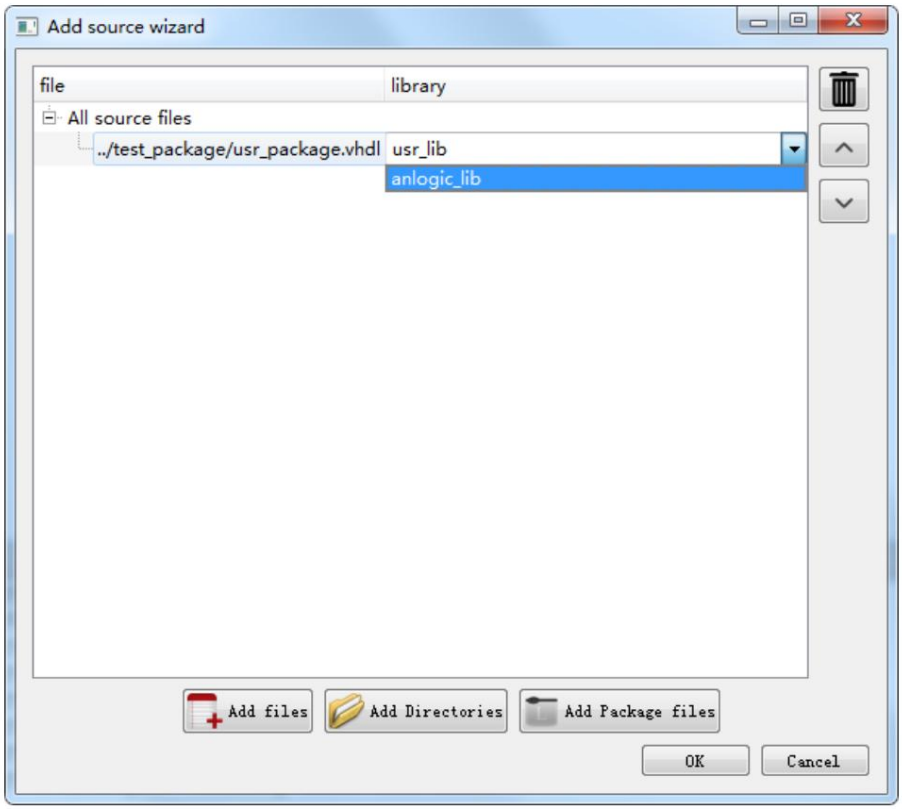
New or select an existing Library.



Machine Translated by Google



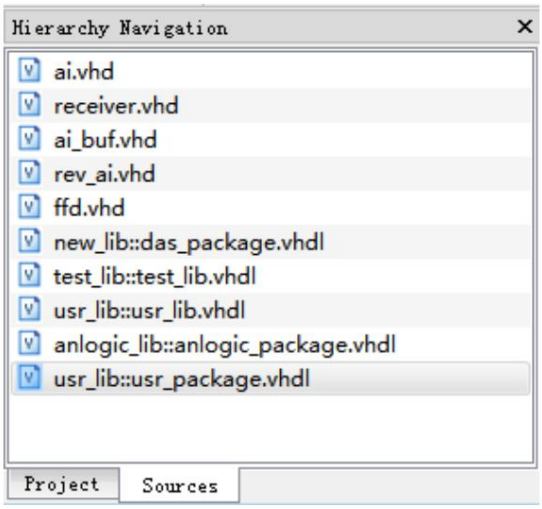
After clicking OK, the usr_package.vhdl file is added to the project by default, and the Library can be used.



After the addition is complete, you can view the content under each Library in the Sources column of the Hierarchy Navigation.

Included files.

Machine Translated by Google



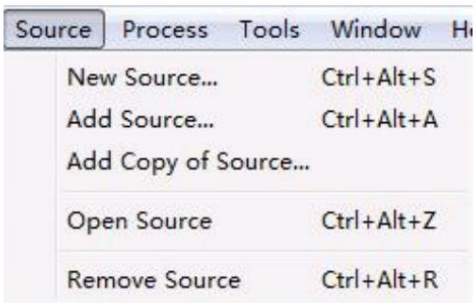
To remove a package file, in the Sources column of the Hierarchy Navigation, select the file, right-click, select Remove Souce.

Note: Once the package file is removed, to use it again, it needs to be re-assigned Library.

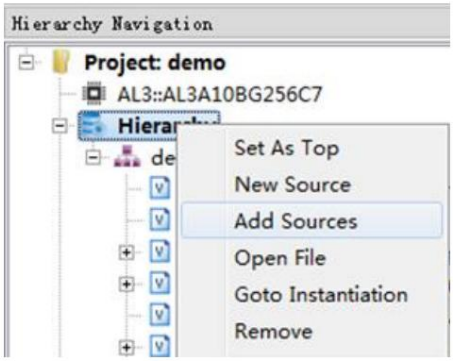
2.4.3 Adding, removing files and changing file attributes

There are three ways to add files:

1. **Source** ÿ **Add Source**

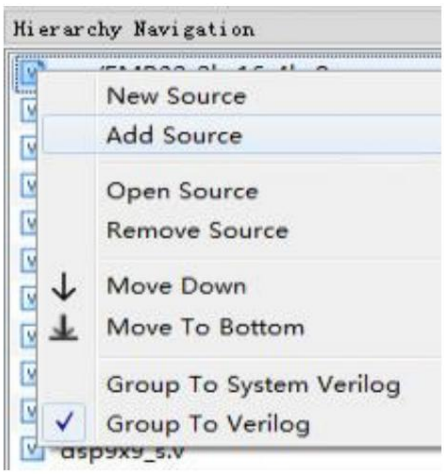


2. On the **Porject** interface of **Hierarchy Navigation ,** right-click within the **Hierarchy** range and select **Add Sources**



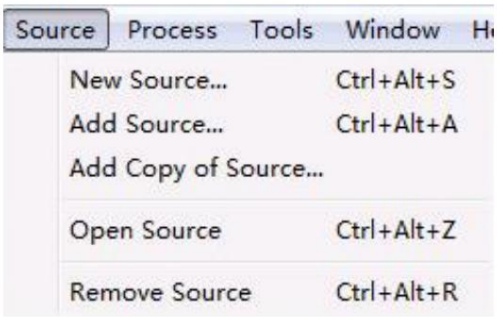
Machine Translated by Google

3. On the **Compile Order** interface of **Hierarchy Navigation ,** right-click and select **Add Source**

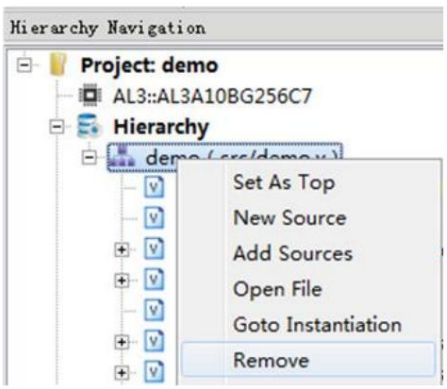


There are also three ways to remove files:

1. **Source** ÿ **Remove Source**

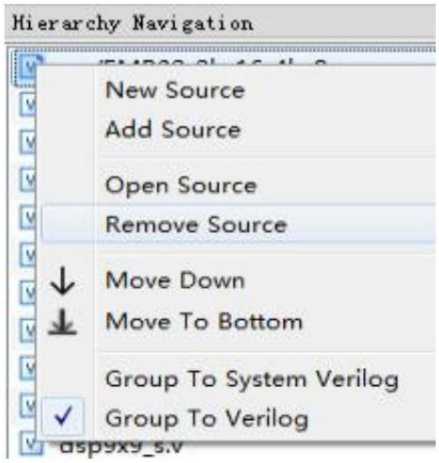


2. In the **Porject** interface of **Hierarchy Navigation ,** select a file in **Hierarchy** and click right key, select **Remove**



3. On the **Compile Order** interface of **Hierarchy Navigation ,** select a file and right-click, select **Remove Source**

Machine Translated by Google



File property changes with .v suffix:

TD now supports System Verilog, and file attributes can be selected or changed for files with a .v suffix.

1. Select file properties when adding files

.v files added by default file type Verilog(*.v, *.h, *.txt, *.vh) belong to Group To by default

Verilog; the .v file added by the file type System Verilog(*.sv, *.v) belongs to Group To System by default

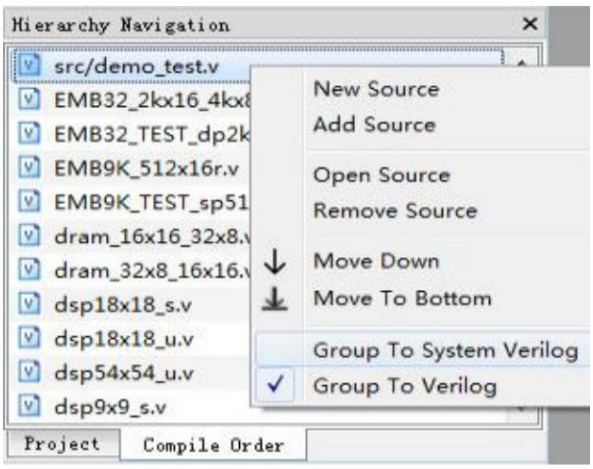
Verilog.

2. Change file properties for files already in the project

To change the properties of a .v file that has been added to the project , **you can**

On the **Order** interface, select the .v file whose properties need to be changed, right-click, and select Group To Verilog / Group

To System Verilog.



Machine Translated by Google

2.4.4 Editing files

TD Editor has many convenient functions for editing files. The specific operations can be performed through the **Edit** option in the menu bar.

line view.

Undo, Redo can undo and redo when editing;

Cut, Copy, Paste are the same as conventional cut, copy and paste functions;

Find search function, Find **Previous** to find the previous one, Find **Next to find the next one, and Replace to** replace

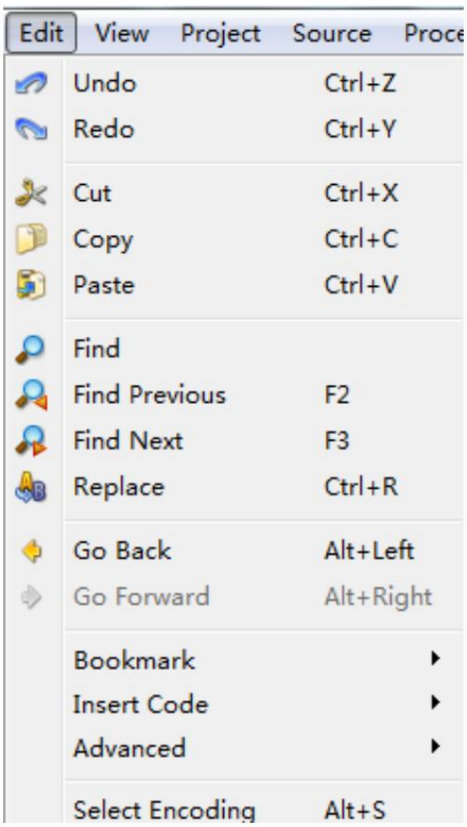
Function;

Go Back jumps back to the beginning of the current line, Go **Forward** jumps to the end of the current line;

Bookmark bookmark function;

Insert Code insert code function;

Select Encoding encodes characters.



Machine Translated by Google

The following mainly introduces the search and replace function, Bookmark **bookmark function, Insert Code** function and **Advanced**

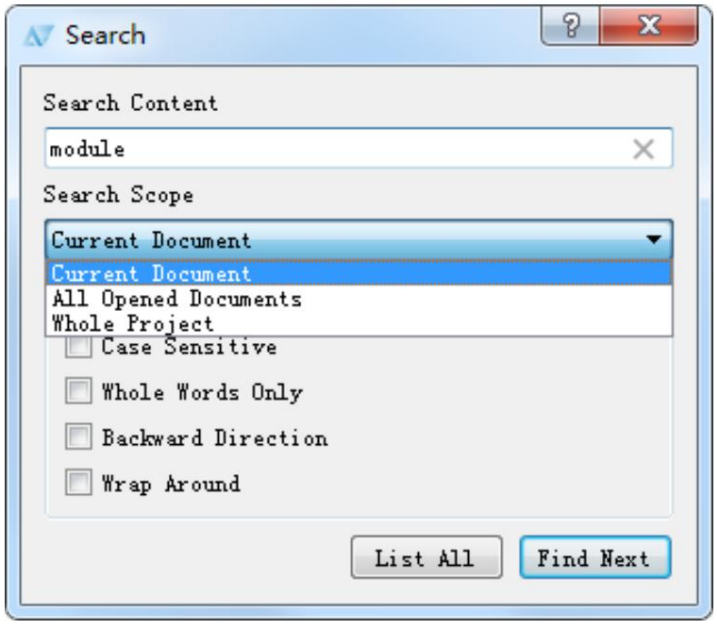
The functions involved in:

1. Find function

Enter the function through **Edit ÿ** Find, or the shortcut **Ctrl + F ,** and the following selection box will appear:

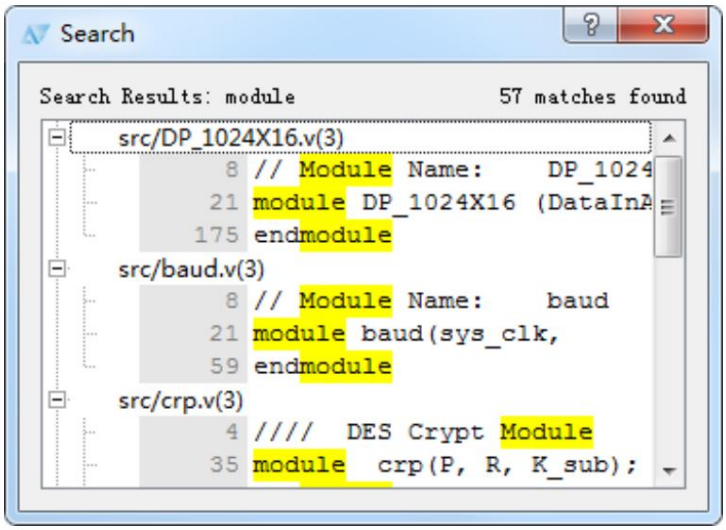
Enter the characters you want to find, select the scope of the search: the current document, all open documents or the entire project,

You can also choose the matching method according to your needs: case matching, whole word matching, up and down, and circular search.



When you click List All, all relevant characters found in the search range will be listed, and you can

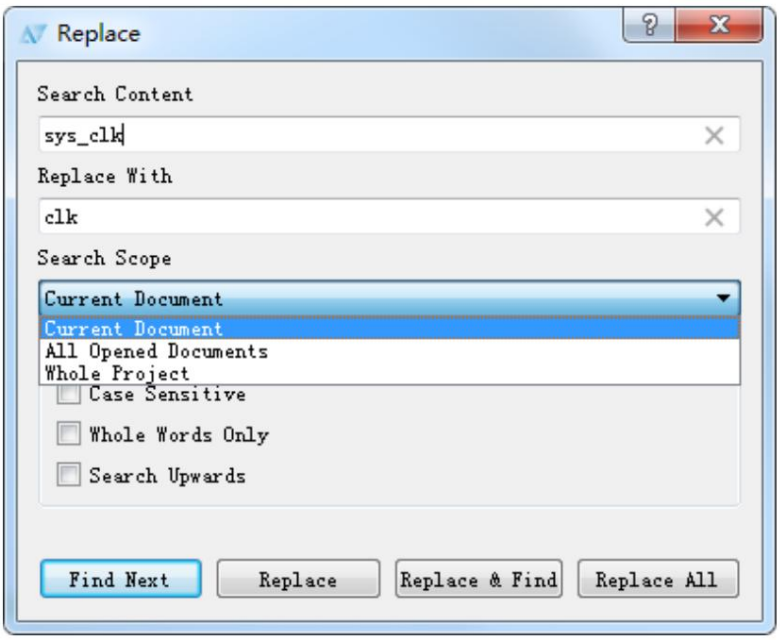
Jump to the location of the source file where the character is located by double-clicking.



Machine Translated by Google

2. Replacement function

Enter the function through **Edit ÿ** Find, or the shortcut **Ctrl + R ,** and the following selection box will appear:



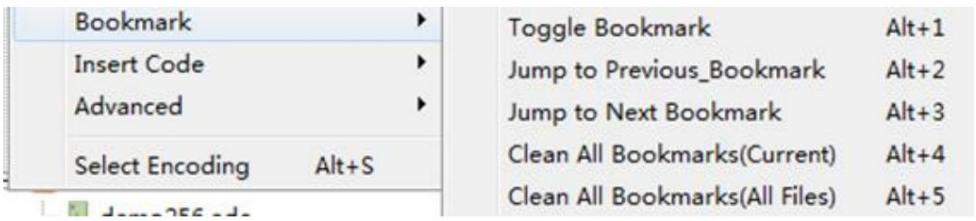
Enter the character you want to find, and enter the replacement content, you can also select the search range and matching method

If you select the search scope as "Whole Project" and click "Replace All", the entire

All sys_clk in the project are replaced with clk.

3. **Bookmark** bookmark function

Expand **Edit ÿ** Bookmark, you can see the following functions:



Toggle Bookmark adds a bookmark in front of the line where the cursor is located, if the line already has a bookmark, then

will cancel the bookmark;

Jump to Previous Bookmark jumps to the previous bookmark;

Jump to Next Bookmark jumps to the next bookmark;

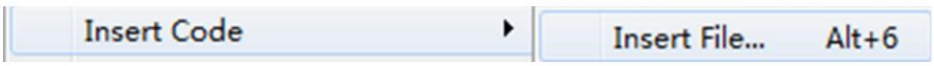
Machine Translated by Google

Clean All Bookmark(Current) clears all bookmarks of the current file;

Clean All Bookmark(All Files) clears all bookmarks for all files.

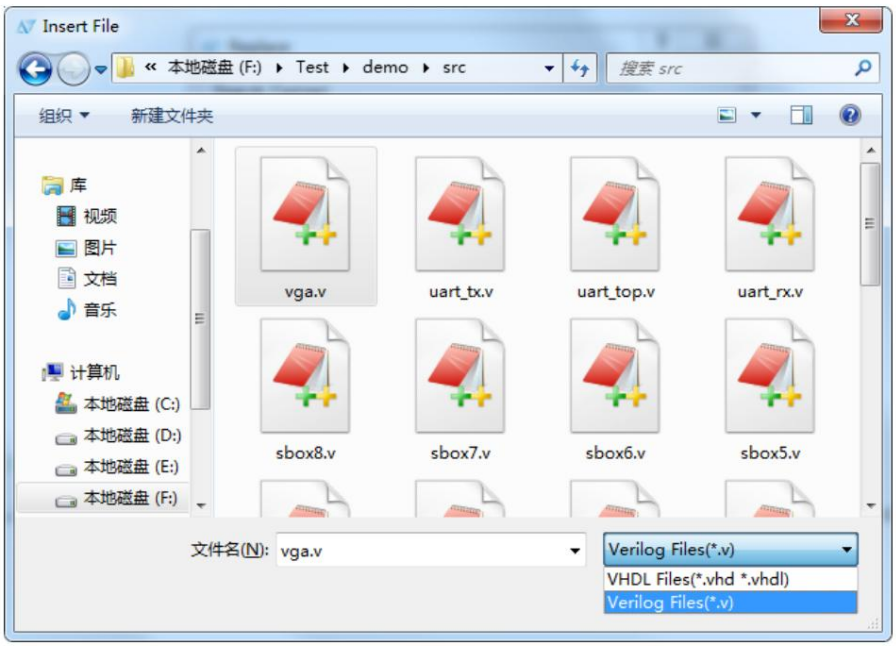
4. **Insert Code** function

Expand **Edit ÿ Insert** Code, you can see the following functions:



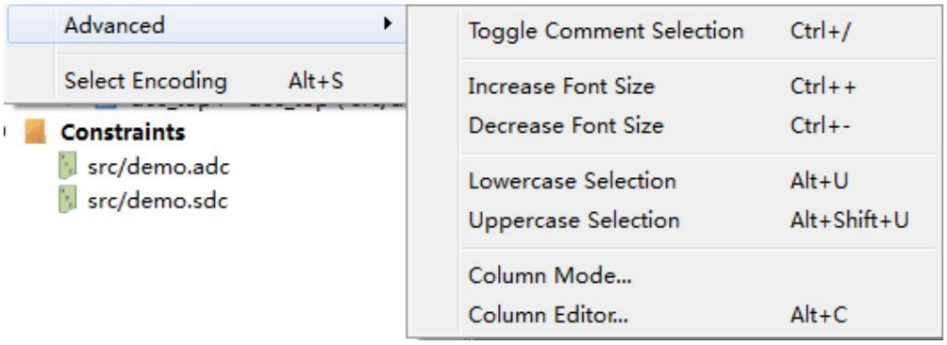
Insert File… Select a file (.v/.vhd/.vhdl) and insert all the code in that file into the current document

The cursor position.



5. **Advanced** features

Expand **Edit ÿ** Advanced, you can see the following functions:



Machine Translated by Google

Toggle Comment Selection Comment the selected code, if the selected code is already commented code, it will be uncommented;

Increase Font Size to enlarge the font;

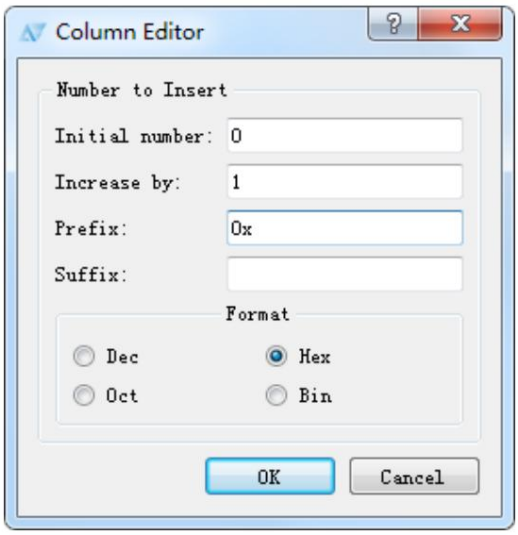
Decrease Font Size reduces the font size;

Lowercase Selection converts the selected characters to lowercase characters;

Uppercase Selection converts the selected characters to uppercase characters;

Column Mode... Column operation mode;

Column Editor… The column editor, shown below, can be incremented in column operation mode, and can be Select a prefix or suffix for the input data.



Machine Translated by Google

3 IP Generator

IP Builder is a graphical interactive design interface for creating IP cores. Users can select IP

Configure and automatically generate the corresponding IP blocks. Currently supported IP modules are divided into **Primitive IP** and **Soft IP.**

There are COMMON, PLL, DSP, RAM, FIFO, RAMFIFO, DRAM, SDRAM, ADC,

Divider, LVDS7_1, CORDIC, CRC, CPU, TEMAC. (ELF series devices only support DRAM

module)

3.1 COMMON module

The Common module contains some commonly used units: BUFG, IDDR, ODDR.

3.1.1 BUFG module

The global clock module can reduce the delay and skew of the global clock signal.

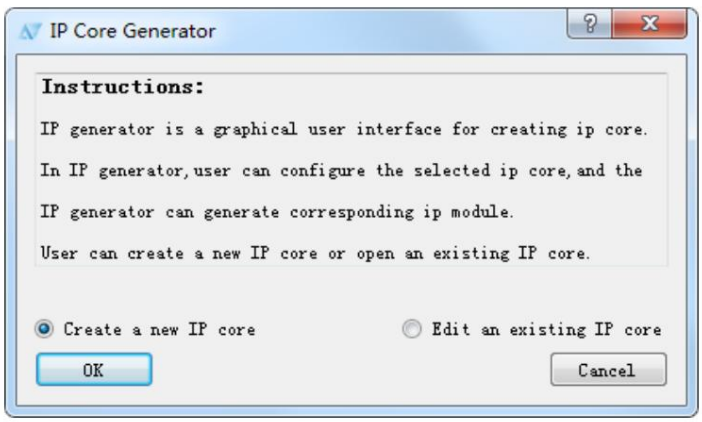
Note: The use conditions of the BUFG module are limited, and cannot be added after the output port of GCLK IO and PLL.

In most cases, the TD software will automatically add a BUFG module for the clock signal in a timely manner. recommended only in software

This module is manually instantiated without adding it.

1. Create a BUFG module

Select **Tools** ÿ **IP** Generator, select "Create **a new IP core"**



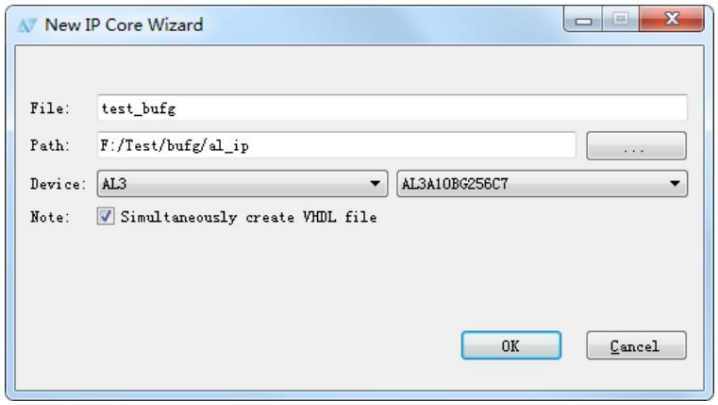
Machine Translated by Google

Enter the module name and select the storage path. Here, if the BUFG module is created on the basis of the existing project, save the

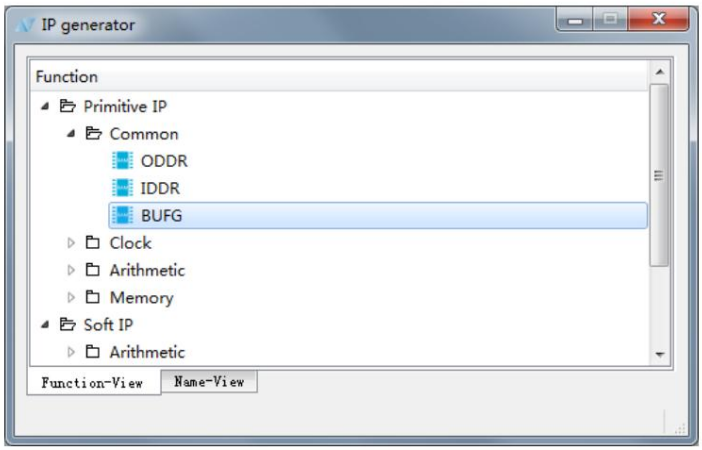
The storage path and device name will be consistent with the project. If a BUFG module is created without a project, the user needs to

Manually set the save path and device name. If **"Simultaneously create VHDL file"** is checked , TD will generate

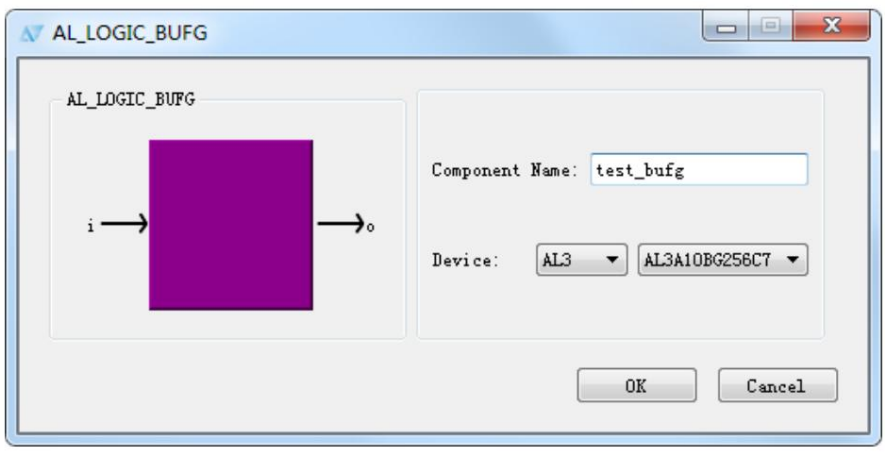
into the corresponding VHDL file.



In the Function window **Primitive IP ,** expand the **Common** module, double-click **BUFG** to open the configuration interface



Enter the module name, select the corresponding device, the default is the engineering device

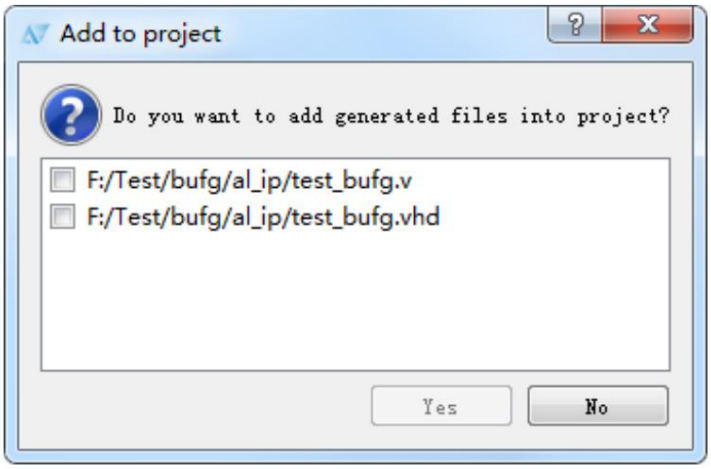


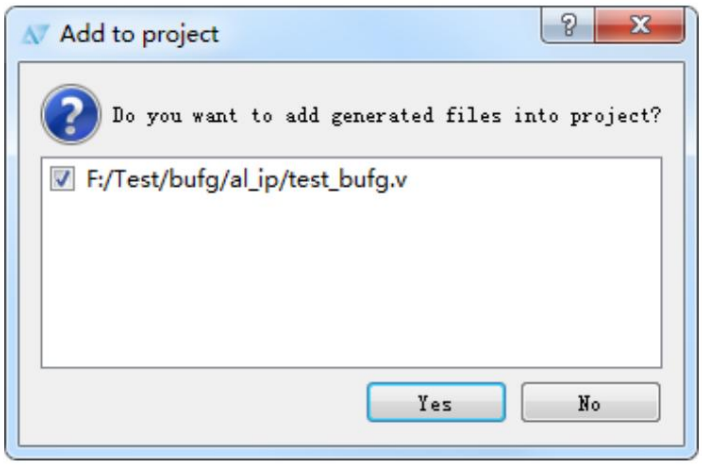
Machine Translated by Google

Click "OK" to complete the setting, and the generated file is as follows:

Continue to click "OK", and choose whether to add files to the project, check any one of the .v/.vhd files, it will automatically

to automatically hide the other file, or uncheck it to redisplay both files.





Machine Translated by Google

2. Instantiate the BUFG module

Take a new project as an example to introduce the process of instantiating the BUFG module. The user instantiates on the basis of the existing project

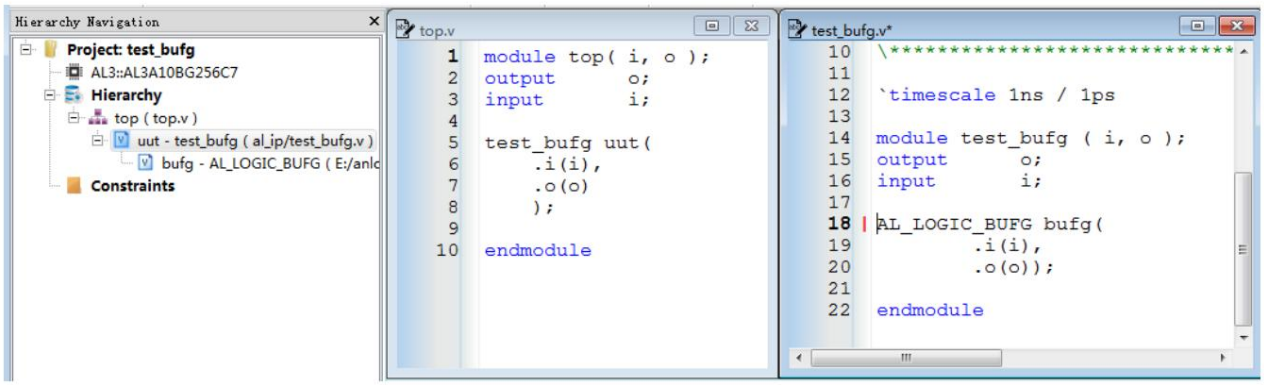
The process is the same.

Create a new project and add a top-level module to the project;

Add the test_bufg.v generated in the previous step to the project;

Call the test_bufg module in the top-level module, modify the inst name and port name, and click the save button, that is

Completed the instantiation of the BUFG module. Click **File ÿ Save to** save the file.



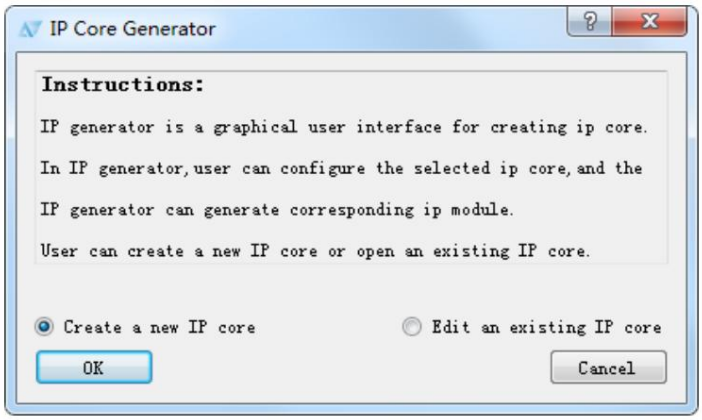
Machine Translated by Google

3.1.2 IDDR module

The input double-edge sampling module is a dedicated input register that can be used for double-edge sampling of the input signal.

1. Create the IDDR module

Select **Tools** ÿ **IP** Generator, select "Create **a new IP core"**

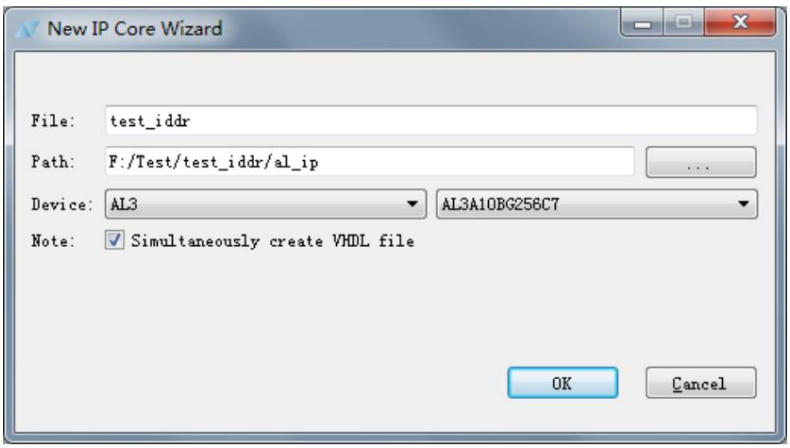


Enter the module name and select the storage path. Here, if the IDDR module is created on the basis of a project, save the

The storage path and device name will be consistent with the project. If an IDDR module is created without a project, the user needs to

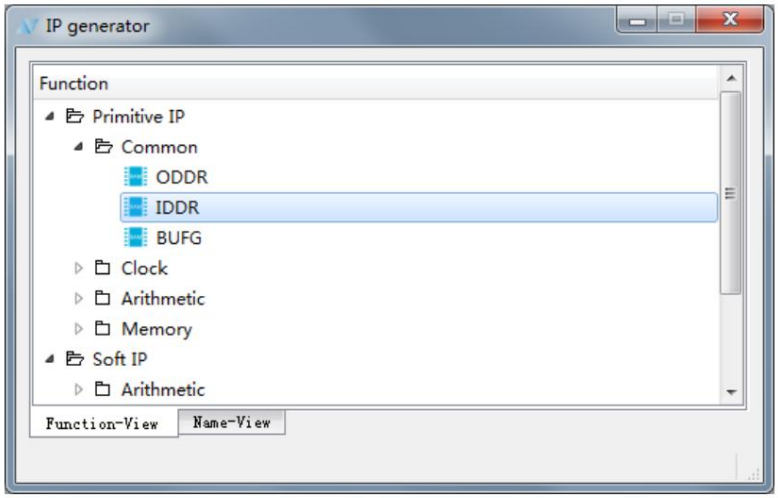
Manually set the save path and device name. If **"Simultaneously create VHDL file"** is checked , TD will generate

into the corresponding VHDL file.

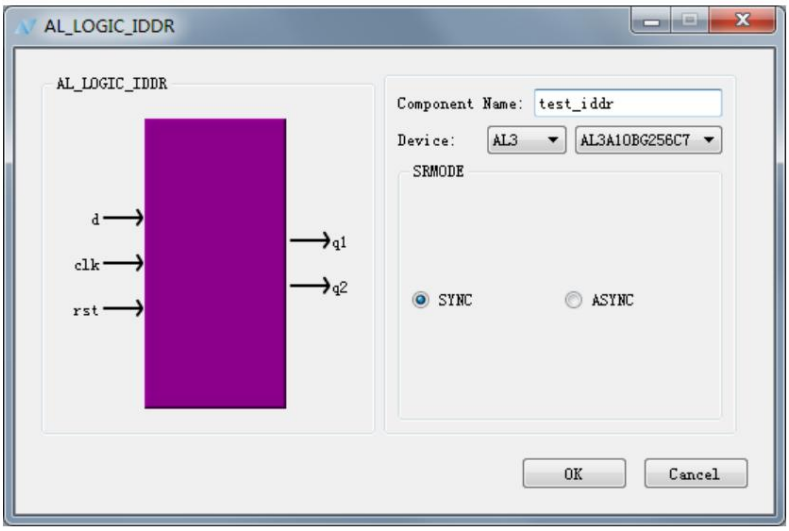


Machine Translated by Google

In the Function window **Primitive IP** expand the **Common** module, double-click **IDDR** to open the configuration interface



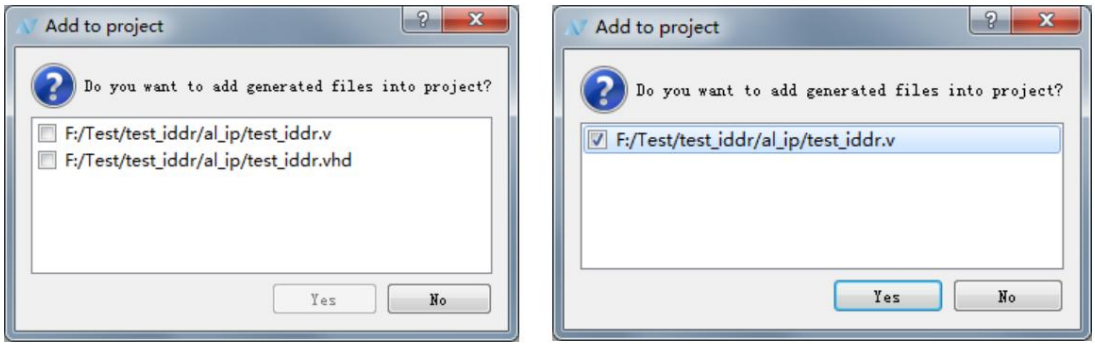
Enter the module name, select the corresponding device, the default is the engineering device



Click "OK" to complete the setting, and the generated file is as follows:

Machine Translated by Google

Continue to click "OK" and choose whether to add files to the project.



2. Instantiate the IDDR module

Take a new project as an example to introduce the process of instantiating the IDDR module. Users can also carry out on the basis of existing projects

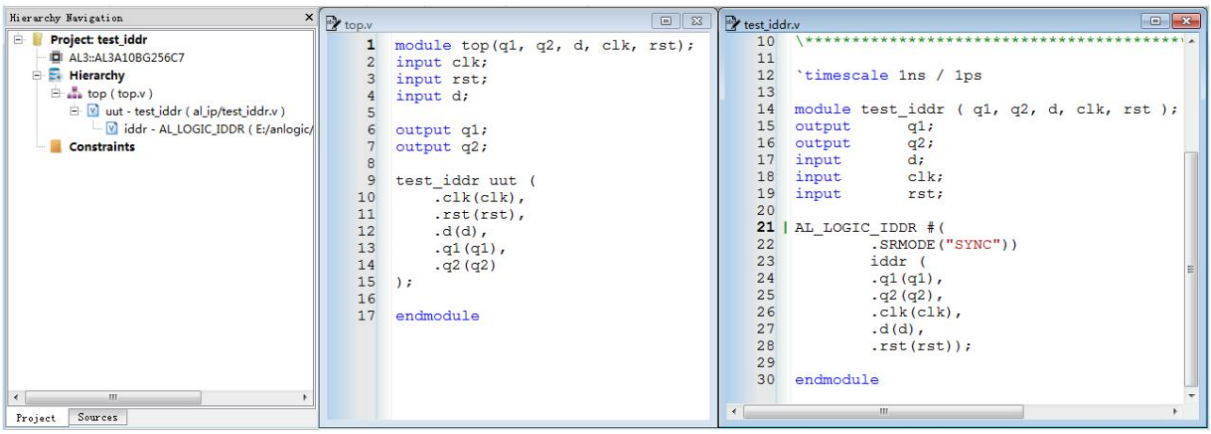
Instantiation, the instantiation process is the same.

Create a new project and add a top-level module to the project;

Add the test_iddr.v generated in the previous step to the project;

Call the test_iddr module in the top-level module, modify the inst name and port name, and click the save button, that is

The instantiation of the IDDR module is complete. Click **File ÿ Save to** save the file.



*For devices like AL3S10, IDDR can be used to sample both edges of the RGMII input signal

AL_LOGIC_IDDR IDDR_0

(.q1(rxd_r2g_tmp[3]), .q2(rxd_r2g_tmp[7]), .clk(rxc), .d(rxd[3]), .rst(~rst_n));

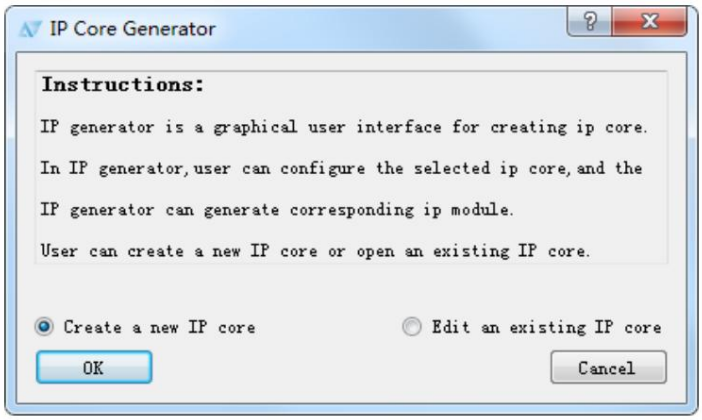
Machine Translated by Google

3.1.3 ODDR module

The output double-edge driving module can be used for double-edge driving of the output signal.

1. Create the ODDR module

Select **Tools** ÿ **IP** Generator, select "Create **a new IP core"**

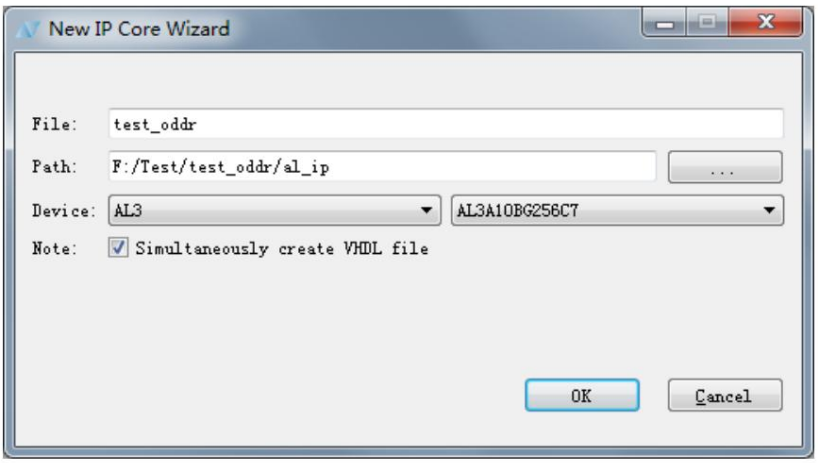


Enter the module name and select the storage path. Here, if the ODDR module is created on the basis of a project, save the

The storage path and device name will be consistent with the project. If an ODDR module is created without a project, the user needs to

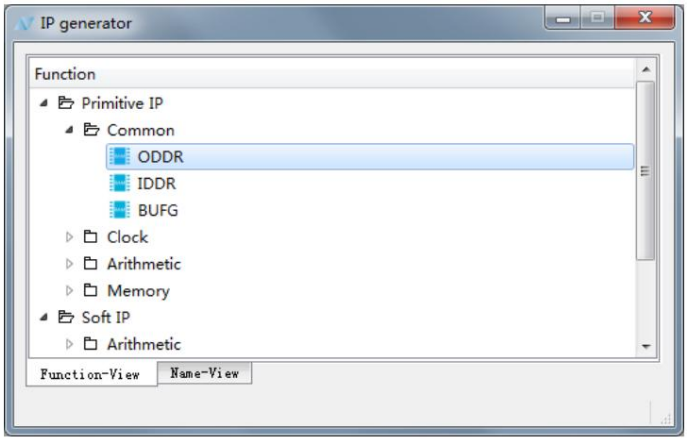
Manually set the save path and device name. If **"Simultaneously create VHDL file"** is checked , TD will generate

into the corresponding VHDL file.

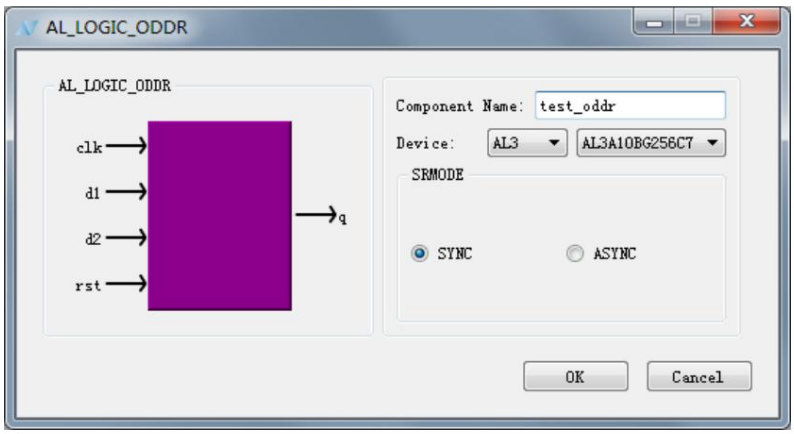


Machine Translated by Google

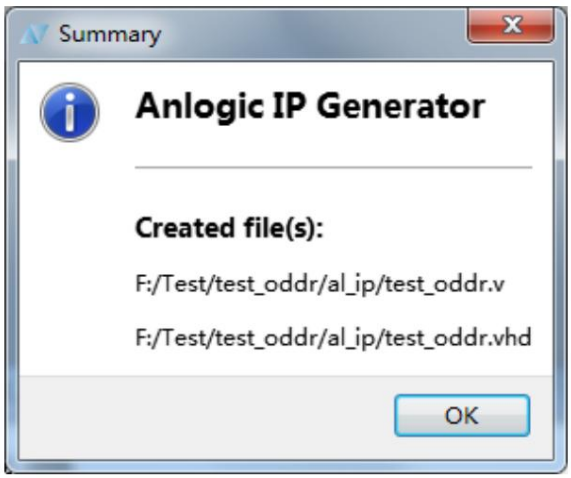
In the Function window **Primitive IP** , expand the **Common** module, double-click **ODDR** to open the configuration interface



Enter the module name, select the corresponding device, the default is the engineering device

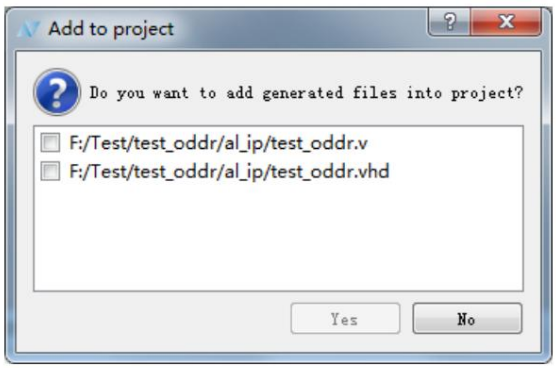


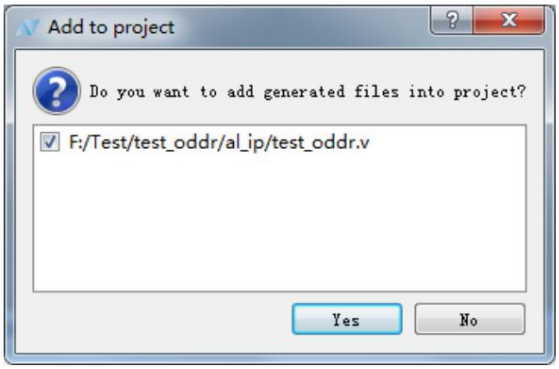
Click "OK" to complete the setting, and the generated file is as follows:



Continue to click "OK" and choose whether to add files to the project.

Machine Translated by Google





2. Instantiate the ODDR module

Take a new project as an example to introduce the process of instantiating the ODDR module. Users can also make progress on the basis of existing projects.

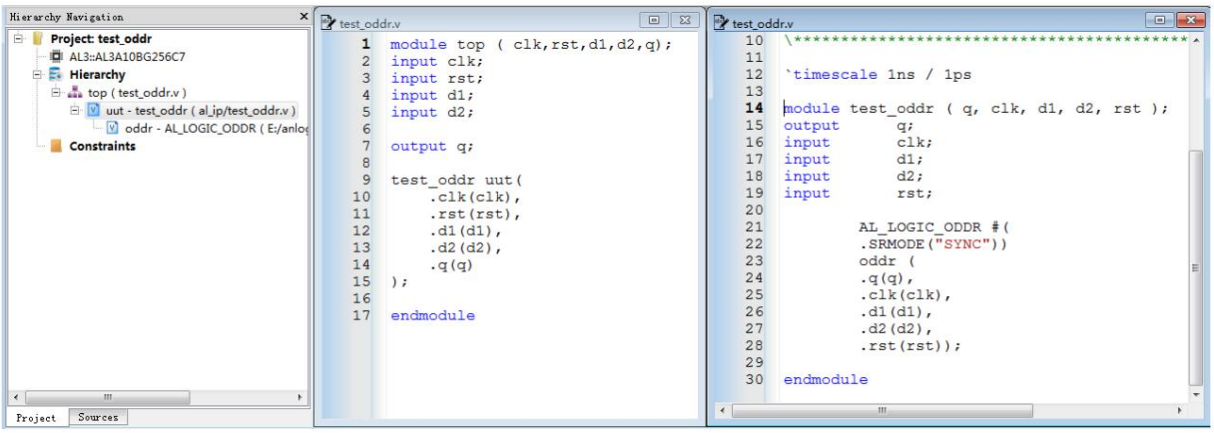
Lines are instantiated, and the instantiation process is consistent.

Create a new project and add a top-level module to the project;

Add the test_oddr.v generated in the previous step to the project;

Call the test_oddr module in the top-level module, modify the inst name and port name, and click the save button, that is

The instantiation of the ODDR module is complete. Click **File ý Save to** save the file.



*For devices such as AL3S10, ODDR is used for double edge driving of RGMII output signal

AL_LOGIC_ODDR ODDR_0

(.q(txd[0]), .clk(txc_tmp), .d1(txd_tmp[4]), .d2(txd_tmp[0]), .rst(RST_OUT0));

Machine Translated by Google

3.2 PLL module

This manual introduces the PLL module in the EAGLE series. EAGLE series FPGAs are embedded with up to 4 multi-function Phase-locked loop (PLL0~PLL3), which can realize high-performance clock management function. Each PLL can achieve clock division/multiplication, Input and feedback clock alignment, multi-phase clock output capabilities.

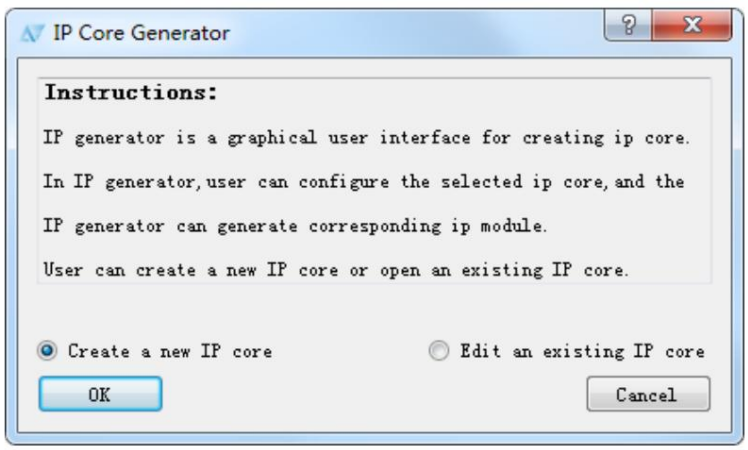
The PLL reference clock inputs are: clock network output, interconnect output, and internal oscillator output.

PLL feedback clock inputs are: clock network output, internal register clock node, interconnect output, PLL internal Feedback clock and phase shift clock C0~C4.

The PLL has a dedicated clock output pin for the output driver chip.

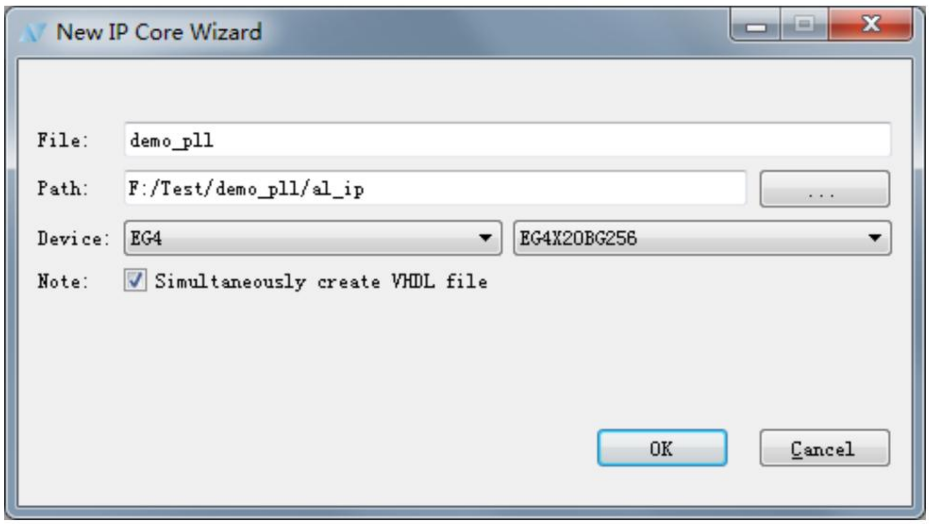
3.2.1 Create PLL module

1. Select **Tools** ÿ **IP** Generator, select "Create **a new IP core"**

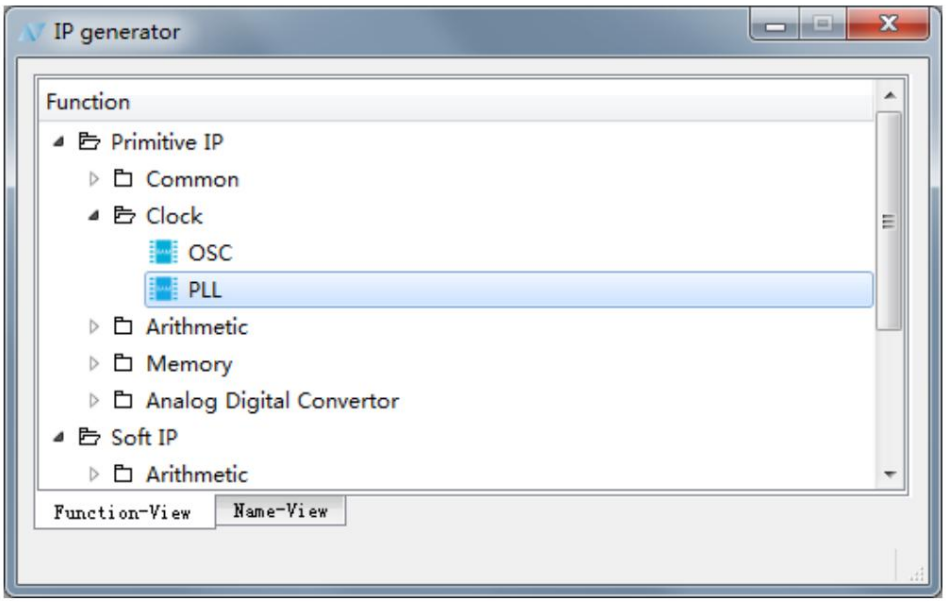


2. Enter the module name and select the storage path. Here, if the PLL module is created on the basis of a project, The storage path and device name will be consistent with the project. If you create a PLL block without a project, Users need to manually set the save path and device name. If **"Simultaneously create VHDL" is** checked **file",** TD will generate the corresponding VHDL file.

Machine Translated by Google



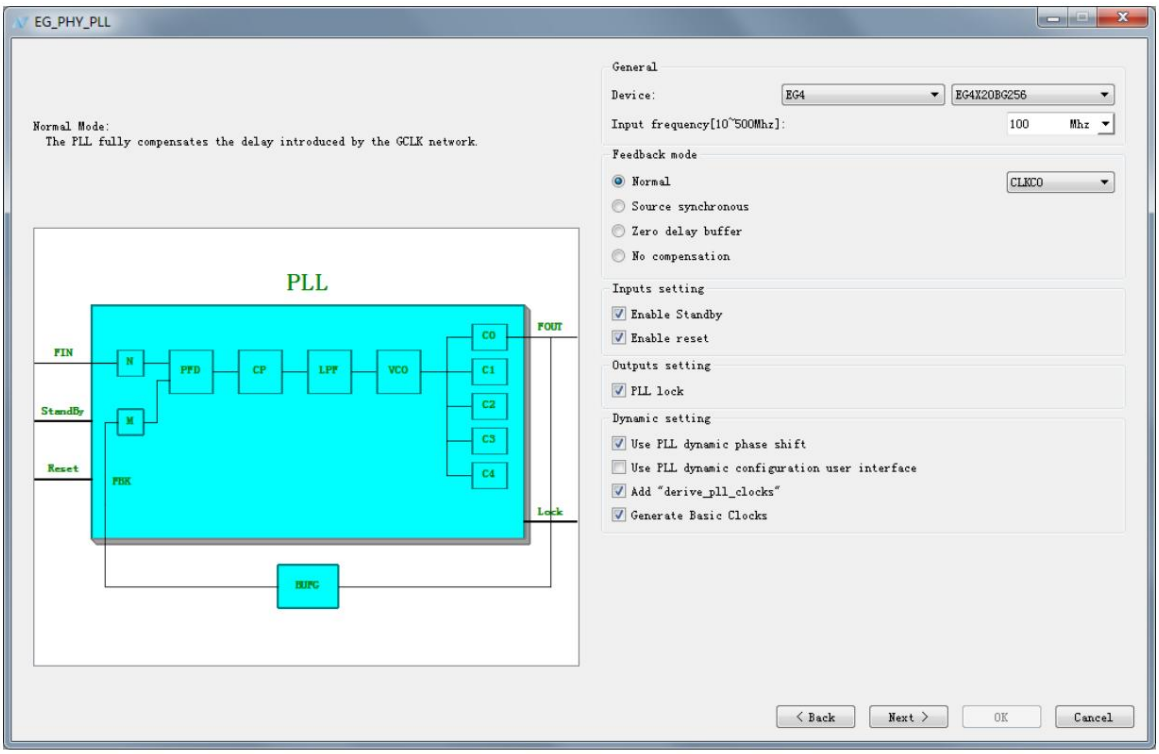
3. In the Function window **Primitive IP ,** expand **Clock** ÿ PLL, and double-click **PLL** to open the configuration interface.



Machine Translated by Google

4. Set the related parameters of the PLL

1) PLL mode setting



The PLL supports 4 feedback modes, each of which supports clock division/multiplication and phase shifting.

a) **Normal mode (Normal)**

In normal mode, the PLL will compensate the GCLK network delay to ensure that the internal register input clock phase and time

The clock pins are in phase.

b) **Source synchronous mode (Source-Synchronous)**

Source synchronous mode adjusts the clock phase to ensure data port to IOB input register through dynamic phase shift function

The delay is equal to the delay from the clock input port to the IOB register (data and clock input port modes are

under the same circumstances).

c) No Compensation Mode (No **Compensation)**

In no compensation mode, the PLL does not compensate for clock network delays, the PLL uses internal self-feedback, which

Improve the jitter characteristics of the PLL.

Machine Translated by Google

d) Zero **Delay Buffer**

In zero-delay buffer mode, the clock output pins are phase-aligned with the PLL reference clock input pins.

The PLL parameter characteristics are shown in the following table:

Parameter	Feature
Input clock frequency range	10~400MHz
Output clock frequency range	4~400MHz
VCO frequency range	300~1200MHz
Number of output ports	5 (the phase of each port can be selected independently)
Reference clock frequency division factor (M)	1~128
Feedback clock frequency division factor (N)	1~128
Output clock frequency division factor (C0~C4)	1~128
Phase shift resolution	45°
Output port selectable phase offset (degrees)	0,45,90,135,180,225,270,315
User dynamic phase shift control	Yes (+/- 45 degree phase shift per unit)
Lock status output	Lock
Dedicated clock output pin	support

When "Add derive_pll_clocks" is selected, all PLLs used will be automatically added when compiling the project

The clkc[x] port generates clock constraints, and the frequency and phase of the generated clock will strictly follow the internal parameters of the PLL.

number setting. And selecting "Generate Basic Clocks" will define the FIN frequency on the corresponding PLL refclk

The reference clock of the rate, otherwise it will automatically search for the refclk pin and the clock defined on the connected net.

will report an error.

Machine Translated by Google

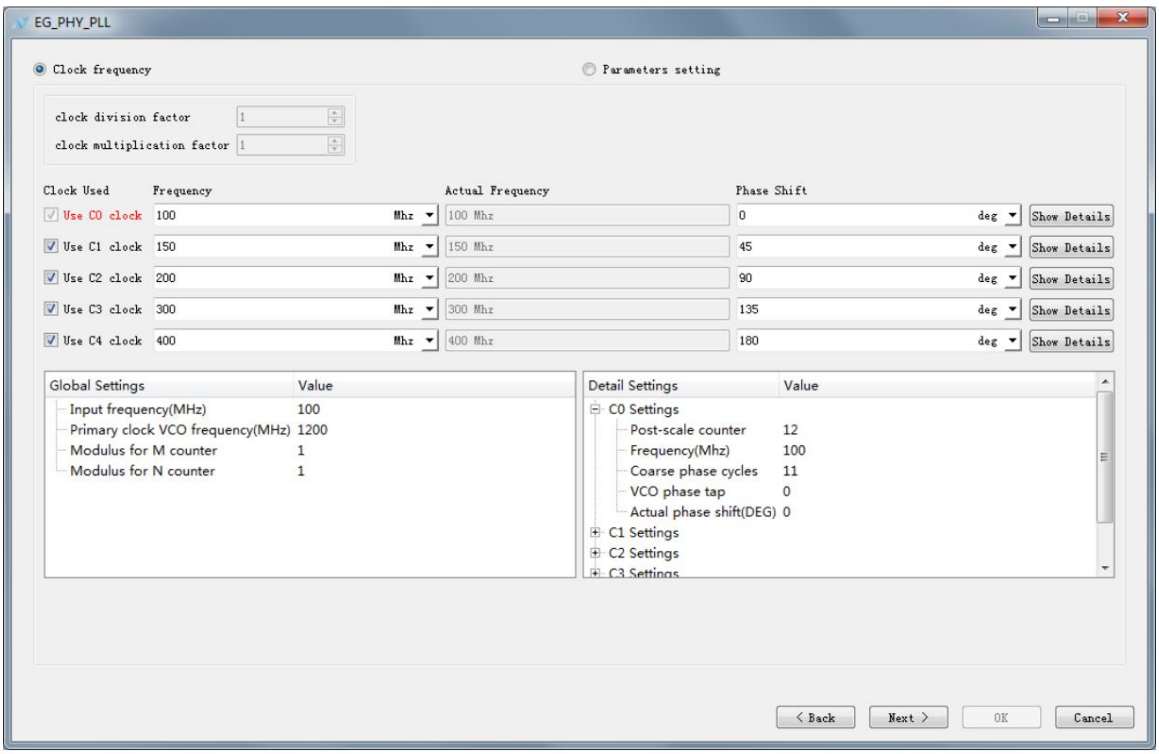
2) Setting of the output clock

Each PLL has 5 output clocks C0~C4, the number of output clocks can be selected and the output can be configured according to requirements

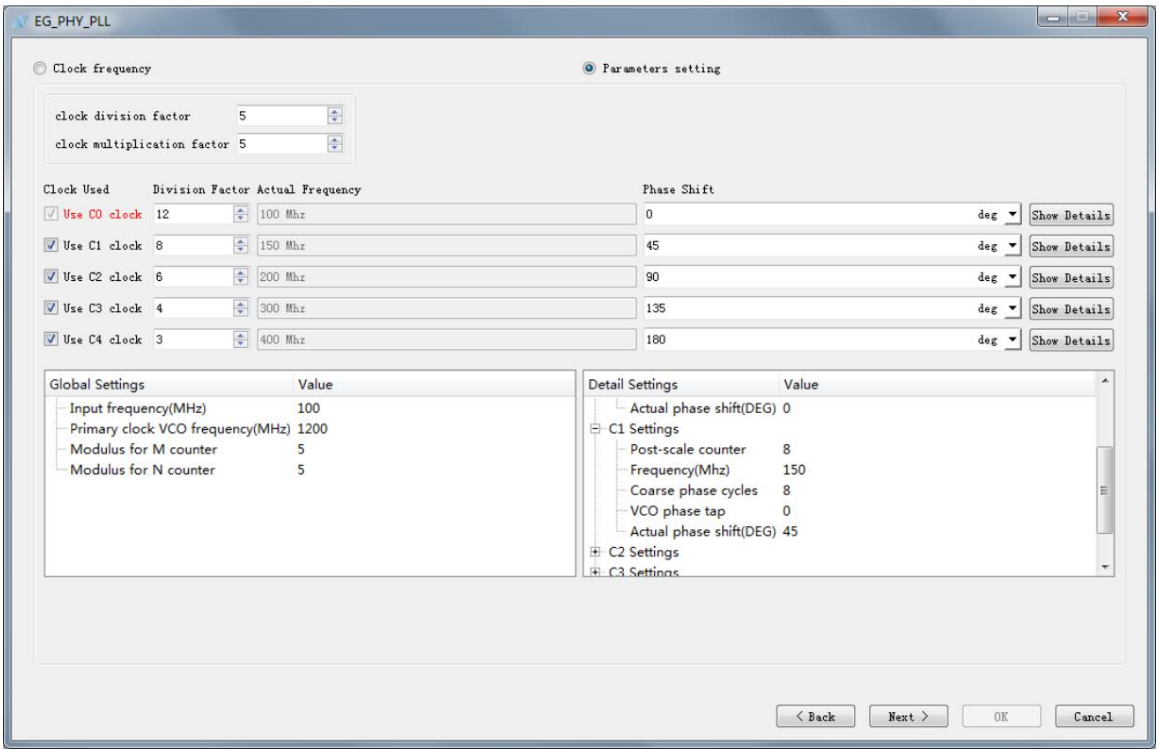
The frequency and phase offset of the clock. When setting the output frequency, you can directly set the output on the **Clock frequency** interface

Frequency, or according to the input frequency, the frequency division coefficient can be set in the **Parameters setting** interface. Click on "Show **Details"**

The values of various performance parameters of this output can be viewed in the lower right corner.

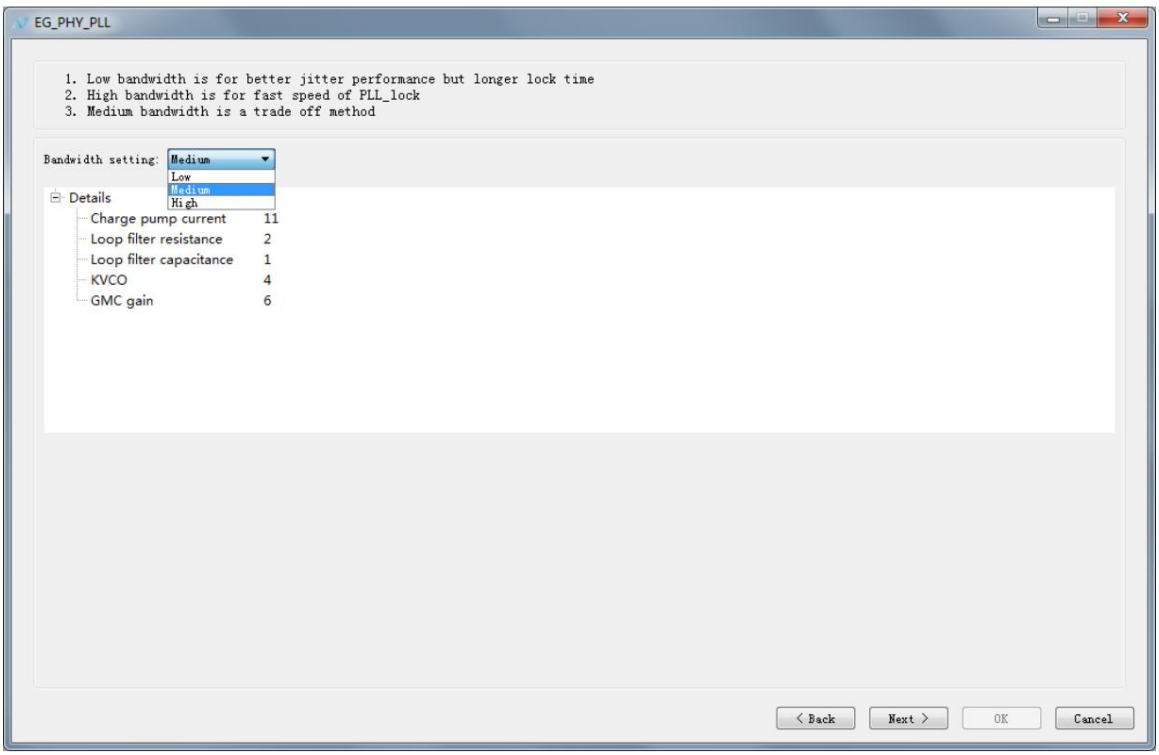


Machine Translated by Google



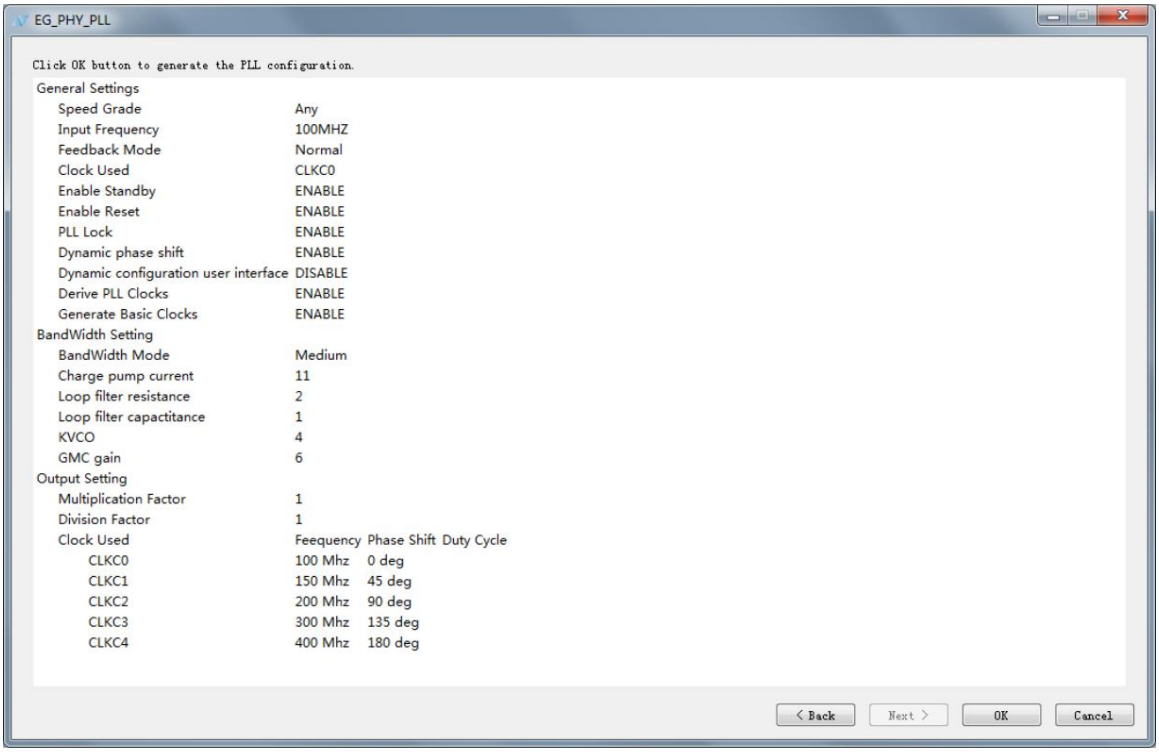
3) Bandwidth setting

The values of Bandwidth can be set to Low, Medium, and High respectively, and the default value is Medium. Click "Show Details" to view the value of each performance parameter of the PLL under this bandwidth.



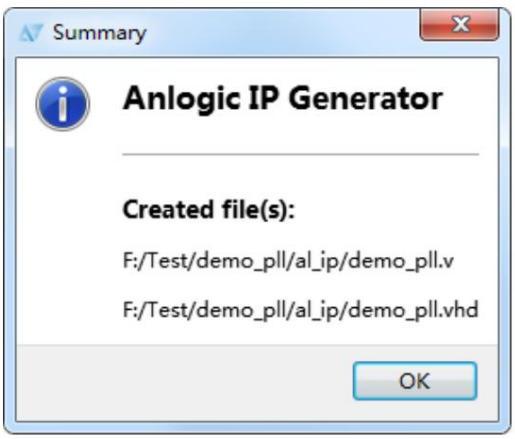
Machine Translated by Google

4) Finally, confirm whether the parameters are correct, and click “Finish” to complete the PLL configuration.



5. TD will give the path of the generated file, after clicking "OK", you can choose whether to generate the file according to the prompt

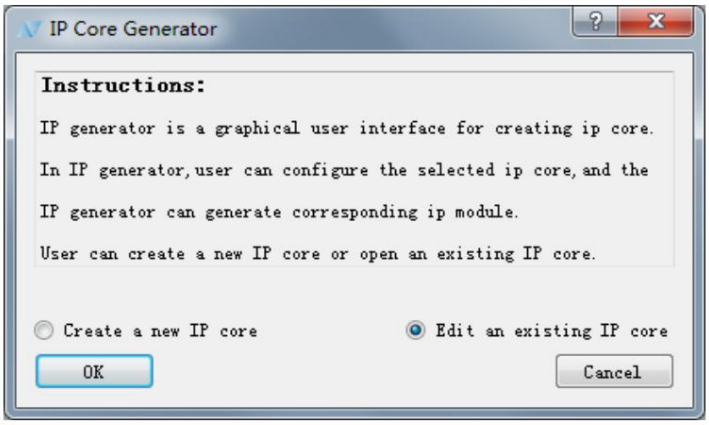
added to the project.

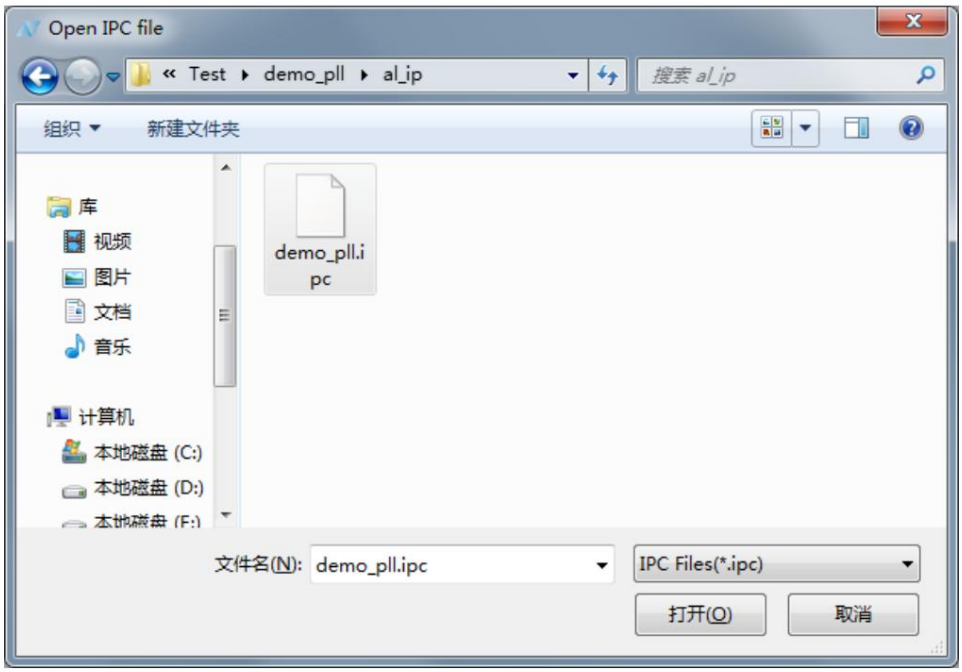


6. An existing IP core can be opened by selecting **Tools ÿ IP Generator and selecting “Edit an existing IP** core”.

existing IP.

Machine Translated by Google





Machine Translated by Google

3.2.2 Instantiate the **PLL** module

Take a new project as an example to introduce the process of instantiating the PLL module. Users can also carry out on the basis of existing projects

Instantiation, the instantiation process is the same.

1. Create a new project and add a top-level module to the project.

2. Add the demo_pll.v generated in the previous step to the project

3. Call the demo_pll module in the top-level module, and modify the inst name and port name, click the save button,

That completes the instantiation of the PLL block. Click **File ý Save to** save the file

top.v*

```
module top (reset,
            stdby,
            extlock,
            clk0_out,
            clk1_out,
            clk2_out,
            clk3_out);

      input reset;
      input stdby;
      output extlock;
      output clk0_out;
      output clk1_out;
      output clk2_out;
      output clk3_out;

      wire osc_clk;

      EG_PHY_OSC EG_PHY_OSC(
        .osc_dis(1'b0),
        .osc_clk(osc_clk)
      );

      demo_pll uut (
           .refclk(osc_clk),
           .reset(reset),
           .stdby(stdby),
           .extlock(extlock),
           .clk0_out(clk0_out),
           .clk1_out(clk1_out),
           .clk2_out(clk2_out),
           .clk3_out(clk3_out));

endmodule
```

demo_pll.v*

```
module demo_pll(refclk,reset,stdby,extlock,
          clk0_out,clk1_out,clk2_out,clk3_out);
      input refclk;
      input reset;
      input stdby;
      output extlock;
      output clk0_out;
      output clk1_out;
      output clk2_out;
      output clk3_out;

      wire clk0_buf;

      EG_LOGIC_BUFG bufg_feedback( .i(clk0_buf), .o(clk0_out) );

      EG_PHY_PLL #(.DYNCFG("DISABLE"),
            .FIN("100.000"),
            .FEEDBK_PATH("CLKC0_EXT"),
            .STDBY_ENABLE("ENABLE"),
            .PLLRST_ENA("ENABLE"),
            .SYNC_ENABLE("ENABLE"),
            .GMC_GAIN(6),
            .ICP_CURRENT(3),
            .KVCO(6),
            .LPF_CAPACITOR(3),
            .LPF_RESISTOR(2),
            .REFCLK_DIV(5),
            .FBCLK_DIV(5),
            .CLKC0_ENABLE("ENABLE"),
            .CLKC0_DIV(12),
            .CLKC0_CPHASE(12),
            .CLKC0_FPHASE(0),
            .CLKC1_ENABLE("ENABLE"),
            .CLKC1_DIV(6),
            .CLKC1_CPHASE(6),
            .CLKC1_FPHASE(6),
            .CLKC2_ENABLE("ENABLE"),
            .CLKC2_DIV(4),
            .CLKC2_CPHASE(5),
            .CLKC2_FPHASE(0),
            .CLKC3_ENABLE("ENABLE"),
            .CLKC3_DIV(3),
            .CLKC3_CPHASE(4),
            .CLKC3_FPHASE(1) )
      pll_inst (.refclk(refclk),
            .reset(reset),
            .stdby(stdby),
            .extlock(extlock),
            .psclk(1'b0),
            .psdown(1'b0),
            .psstep(1'b0),
            .psclksel(3'b000),
            .dclk(1'b0),
            .dcs(1'b0),
            .dwe(1'b0),
            .di(8'b00000000),
            .daddr(6'b000000),
            .fbclk(clk0_out),
            .clkc({open, clk3_out, clk2_out, clk1_out, clk0_buf}));

endmodule
```

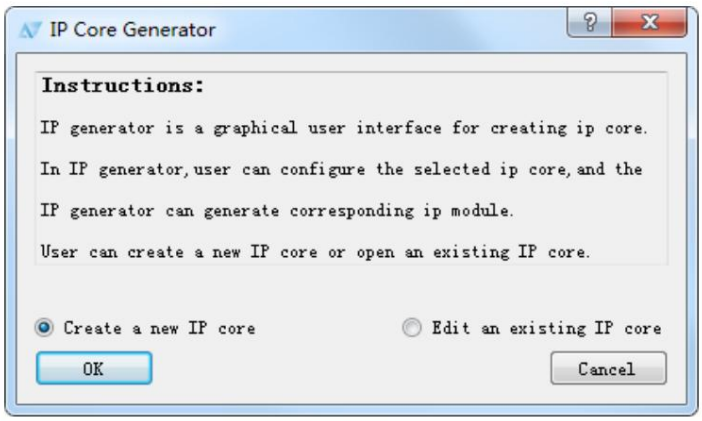
Machine Translated by Google

3.3 DSP module

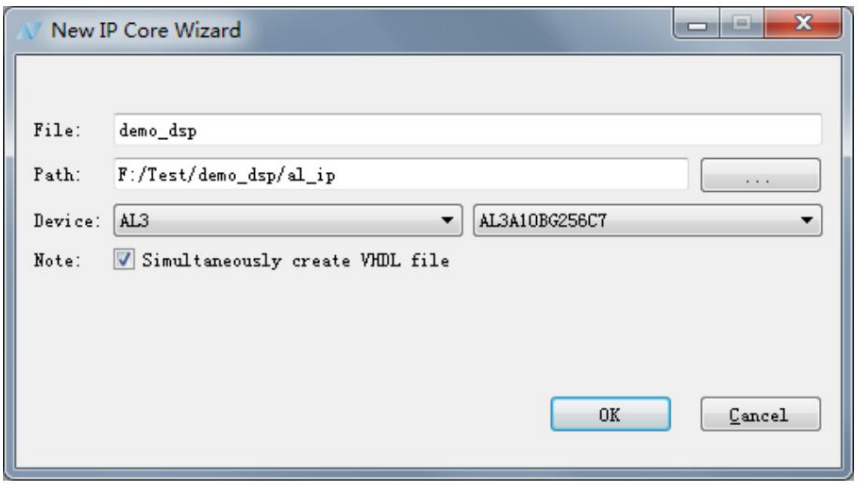
In the AL3 family of devices, the embedded multiplier can be configured as an 18x18 multiplier with input and output registers multipliers, or configured as two 9x9 multipliers.

3.3.1 Create DSP module

1. Select **Tools ÿ IP** Generator, select "Create **a new IP core"**

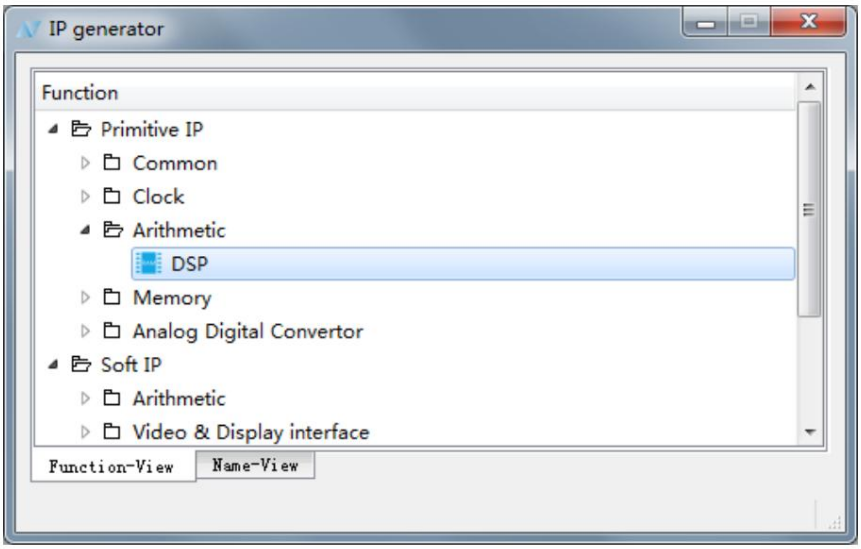


2. Enter the module name and select the storage path. Here, if the DSP module is created on the basis of a project, the storage The path and device name will be the same as the project. If a DSP module is created without a project, the user needs to Manually set the save path and device name. If **"Simultaneously create VHDL file" is** checked , TD will Generate the corresponding VHDL file.

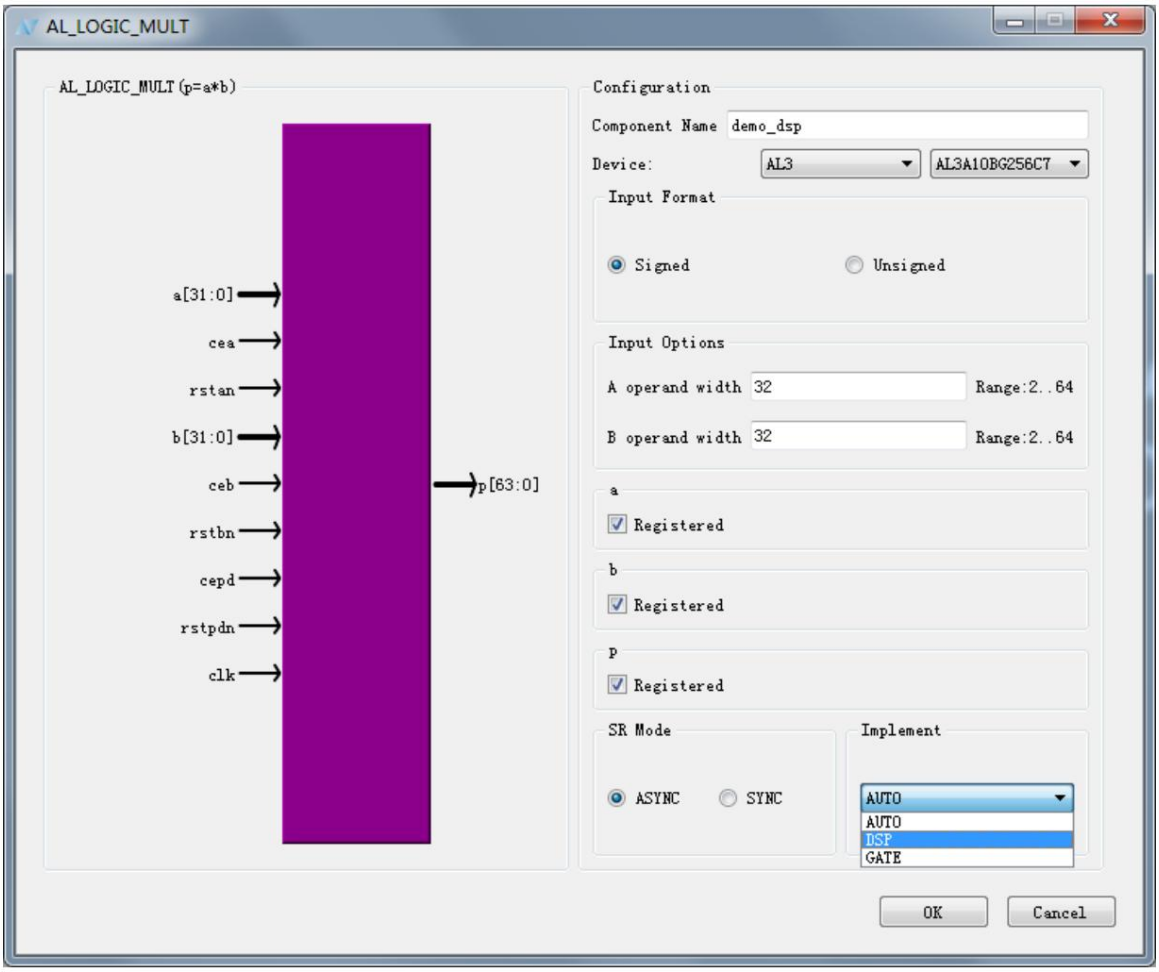


Machine Translated by Google

3. In the Function window **Primitive IP ,** expand **Arithmetic ÿ** DSP, and double-click **DSP** to open the configuration interface



4. Fill **in the "Component** Name" and set the corresponding parameters



Machine Translated by Google

In **IP Generator ,** users can customize the implementation of multiplication operation, and provide three parameters for users to choose.

Among them, DSP means that the hardware DSP is forced to implement the multiplication operation. If the DSP is not enough, an error will be reported, and the hardware DSP will

It is faster than using logic gates; **GATE** means that only logic gates are used to implement multiplication operations; AUTO means

Use DSP first, if DSP is not enough, use logic gate to realize multiplication operation. The default parameter is: AUTO.

The embedded multipliers of the AL3 family of devices are composed of the following units: input registers, multiplier cores, and input multipliers.

out the register.

ÿ Input register

Depending on the mode of operation of the multiplier, each multiplier input signal can be connected to an input register, or directly as

9bit or 18bit form connected to the internal multiplier. Each input of the multiplier can be individually set whether to use the input

into the register.

The following control signals are available for each input register in the embedded multiplier:

ÿ Clock (clk)

ÿ Clock Enable (cea / ceb)

ÿ Synchronous/asynchronous clear (rstan / rstbn : n means active low).

All input and output registers in the same embedded multiplier are driven by the same clock signal, clock enabled

Signal and asynchronous clear signal drivers can be configured independently.

ÿ Multiplier core

The multipliers of the embedded multiplier block support either 9×9 or 18×18 multipliers, and can implement any of these configuration widths.

other multipliers in between. Depending on the data width or operating mode of the multiplier, a single embedded multiplier can

to perform one or two multiplications.

The two operands of the multiplier can be declared as signed/unsigned with the signed/unsigned option to

This determines the type of multiplier.

Machine Translated by Google

ÿ Output register

Depending on the operating mode of the multiplier, the output registers can be used in 18-bit or 36-bit form for embedded

The output of the multiplier is registered. The following control signals are available for each output register in the embedded multiplier

device:

ÿ Clock (clk)

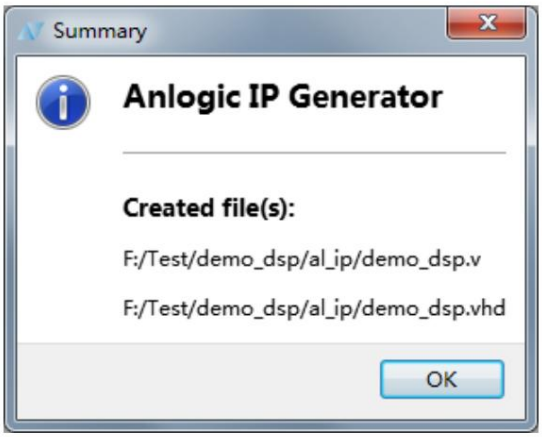
ÿ Clock Enable (cepd)

ÿ Synchronous/asynchronous clear (rstpdn : n means active low)

All input and output registers in the same embedded multiplier are driven by the same clock signal, clock enabled

Signal and asynchronous clear signal drivers can be configured independently.

5. Click "OK" to complete the setting of DSP, TD will give the path of the generated file.



An existing IP core can be opened by selecting Tools ÿ IP Generator and selecting "Edit an existing IP core"

IP.

Machine Translated by Google

3.3.2 Instantiate the **DSP** module

This manual takes a new project as an example to introduce the process of instantiating a DSP module. Users can also build on existing projects

The instantiation is performed on the above, and the instantiation process is the same.

1. Create a new project and add a top-level module to the project.

2. Add the demo_dsp.v generated in the previous step to the project

3. Call the demo_dsp module in the top-level module, and modify the inst name and port name, click the save button,

That is, the instantiation of the DSP module is completed.

```
1  module top ( p, a, b, cea, ceb, cepd,
2               clk, rstan, rstbn, rstpdn )
3
4       output [86:0] p;
5
6       input  [63:0] a;
7       input  [22:0] b;
8       input  cea;
9       input  ceb;
10      input  cepd;
11      input  clk;
12      input  rstan;
13      input  rstbn;
14      input  rstpdn;
15
16      demo_dsp uut(
17            .a(a),
18            .b(b),
19            .p(p),
20            .cea(cea),
21            .ceb(ceb),
22            .cepd(cepd),
23            .clk(clk),
24            .rstan(rstan),
25            .rstbn(rstbn),
26            .rstpdn(rstpdn));
27
28
29  endmodule
```

```
12
13  module demo_dsp ( p, a, b, cea, ceb, cepd,
14                    clk, rstan, rstbn, rstpdn );
15
16      output [86:0] p;
17
18      input  [63:0] a;
19      input  [22:0] b;
20      input  cea;
21      input  ceb;
22      input  cepd;
23      input  clk;
24      input  rstan;
25      input  rstbn;
26      input  rstpdn;
27
28      AL_LOGIC_MULT #( .INPUT_WIDTH_A(64),
29                       .INPUT_WIDTH_B(23),
30                       .OUTPUT_WIDTH(87),
31                       .INPUTFORMAT("SIGNED"),
32                       .INPUTREGA("ENABLE"),
33                       .INPUTREGB("ENABLE"),
34                       .OUTPUTREG("ENABLE"),
35                       .IMPLEMENT("AUTO"),
36                       .SRMODE("ASYNC"))
37                  inst(
38                       .a(a),
39                       .b(b),
40                       .p(p),
41                       .cea(cea),
42                       .ceb(ceb),
43                       .cepd(cepd),
44                       .clk(clk),
45                       .rstan(rstan),
46                       .rstbn(rstbn),
47                       .rstpdn(rstpdn));
48
49  endmodule
```

Console

```
Utilization Statistics
#le                    1329    out of   8640 15.38%
  #lut                 1329    out of   8640 15.38%
  #reg                    0    out of   8640  0.00%
#dsp                      3    out of      3    100%
#bram                     0    out of     48      0%
 #bram9k                  0
 #fifo9k                  0
```

Console | Error | Warning

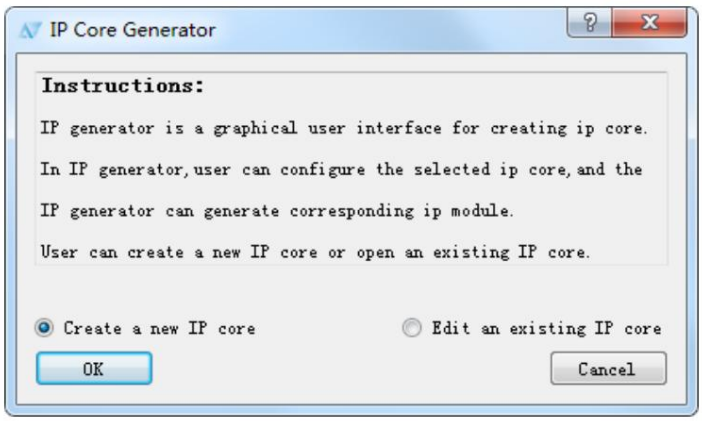
Machine Translated by Google

3.4 Divider module

The TD software implements a clock-driven divider.

3.4.1 Create Divider Module

1. Select **Tools ÿ IP** Generator and select "Create **a new IP core".**

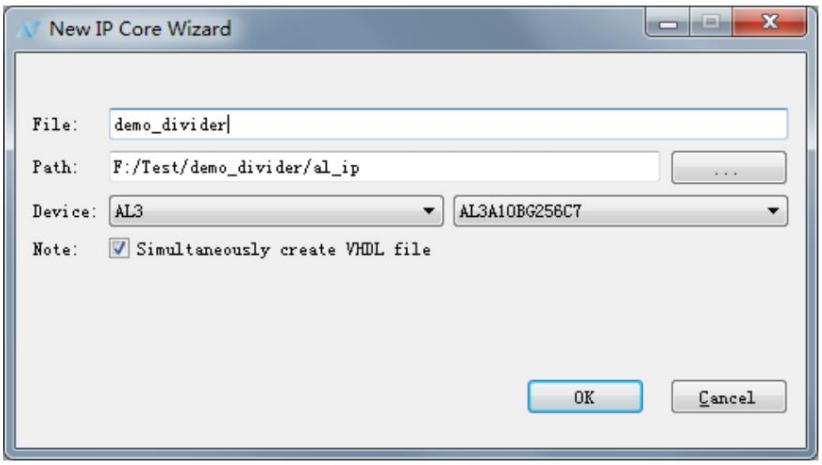


2. Enter the module name and select the storage path. Here, if the Divider model is created on the basis of a project

Blocks, storage paths and device names will be the same as the project. If created without a project

Divider module, the user needs to manually set the save path and device name. If **"Simultaneously**

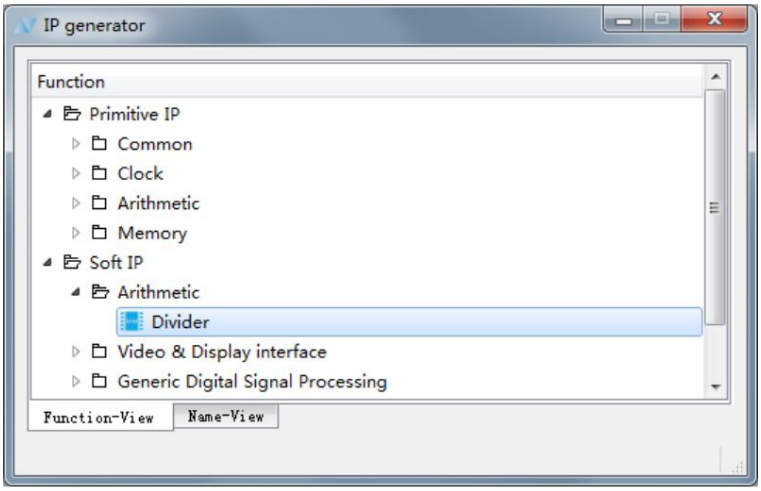
create VHDL file", TD will generate the corresponding VHDL file.



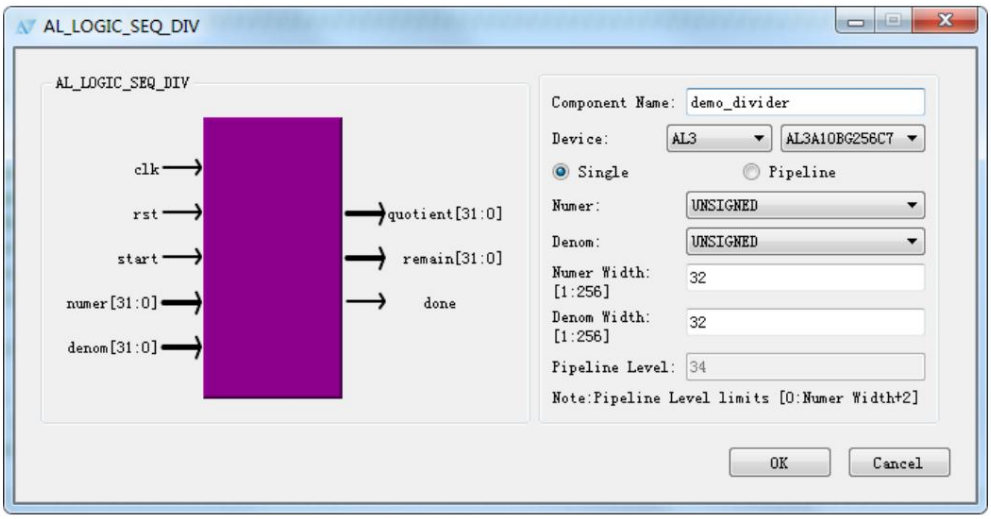
3. In the Function window **Soft IP ,** expand **Arithmetic ÿ Divider, double-click Divider** to open the configuration interface

noodle.

Machine Translated by Google



4. Fill in the **Component Name** and corresponding operands.



in,

1) Single and Pipeline correspond to two modes of serial divider and pipeline divider respectively;

2) Numer Width is the bit width of the dividend and also the bit width of the quotient;

3) Denom Width is the bit width of the divisor and also the bit width of the remainder;

4) The data type of Numer/Denom can be signed or unsigned;

5) Pipeline Level is the number of pipeline stages. In serial divider mode, the number of pipeline stages is fixed.

The bit width of the Pipeline pipeline divider can be modified;

6) clk is the clock used for driving calculation;

7) rst is the reset signal. When rst is 1, the internal registers and outputs (quotient, remain) will be set to 0,

Machine Translated by Google

And done will be set to 1;

8) start is the calculated start signal. When start is set to 1, the input data is buffered to the internal register

In the device; and when start changes from 1 to 0, the calculation process really starts;

9) numer is the dividend. Although start is 0, changing the value of numer will not affect the calculation (the original

The value has been buffered to the internal register on the rising edge of the clock when start is 1). However, in order to prevent the waveform

There is a misunderstanding when displaying, please set done to 1 (or wait for the number of clock cycles required for the calculation process)

After that, it will provide the input for the next calculation;

10) denom is the divisor. Note the same as number;

11) quotient is a quotient. Only when done is 1, the value of quotient is the final result calculated;

12) remain is the remainder. Only when done is 1, the value of remain is the final result of the calculation;

13) done is the signal that the calculation is completed. When done is set to 1, the calculation is complete. to ensure the output

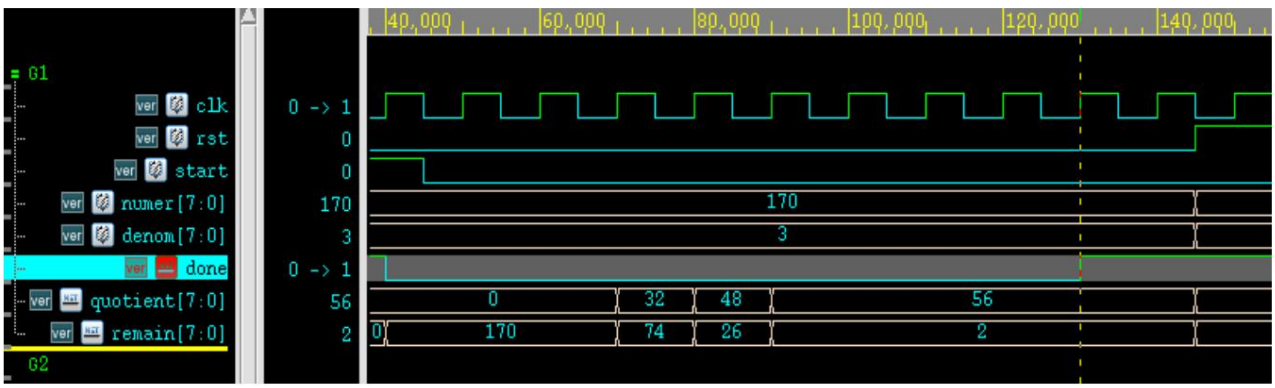
The correctness of the result requires that the output values quotient and remain are used only when the done signal is 1; the same

It also needs to provide the next set of input values after the done signal is 1.

The simulated waveforms are as follows:

numer = 170, denom = 3, when start changes from 1 to 0, start the calculation until done = 1, get

out quotient=56, remain=2.

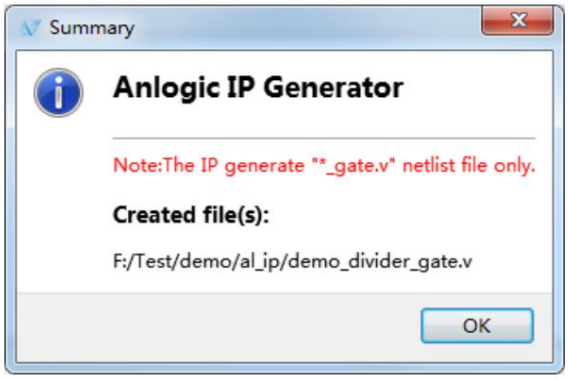


5. Click "OK" to complete the Divider settings, TD will generate the corresponding netlist file and give the generated file

Machine Translated by Google

path. Divider only supports Verilog, not VHDL for now. By selecting Tools ÿ IP Generator,

Select "Edit an existing IP core" to open an existing IP.



3.4.2 Instantiate the **Divider** module

This manual takes a new project as an example to introduce the process of instantiating the Divider module. Users can also use the base of existing projects

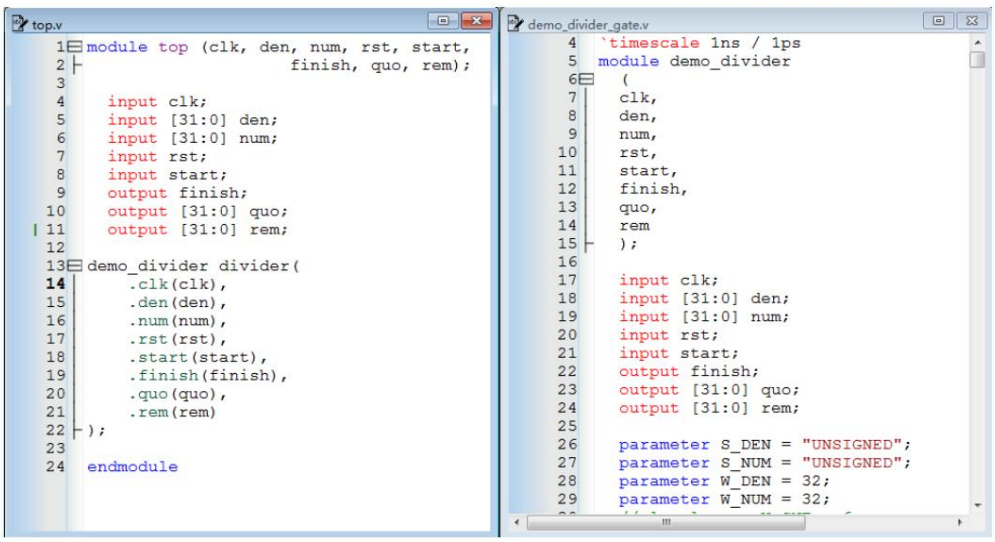
On the basis of instantiation, the instantiation process is consistent.

1. Create a new project and add a top-level module to the project.

2. Add the demo_divider_gate.v generated in the previous step to the project

3. Call the demo_divider module in the top-level module, modify the inst name and port name, and click Save

   button to complete the instantiation of the Divider module.



Machine Translated by Google

3.5 BRAM module

The AL3 family of devices supports Embedded Memory Blocks. Included in AL3-10

There are two types of EMBs: EMB9K and EMB32K.

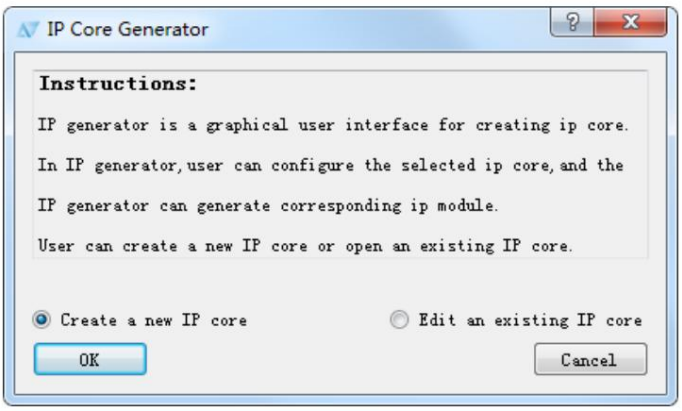
Each EMB9K block has a capacity of 9Kbits, and multiple EMB9K modules are arranged in a row, and are distributed in the programmable function blocks by row.

(Programmable Function Block, PFB). EMB32K has a capacity of 32Kbits per block, distributed in the IO space

in the gap.

3.5.1 Create BRAM module

4. Select **Tools ÿ IP** Generator, select "Create **a new IP core"**

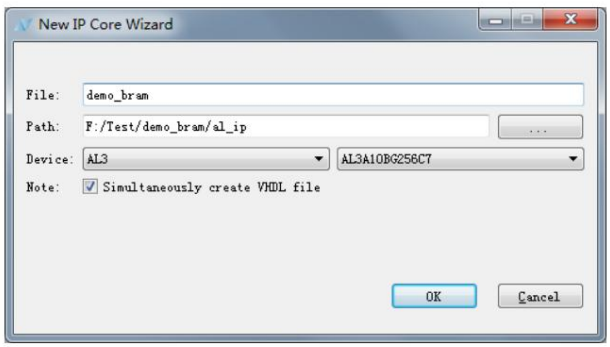


5. Enter the module name and select the storage path. Here, if a BRAM module is created on the basis of a project,

The storage path and device name will be consistent with the project. If you create a BRAM module without a project

block, the user needs to manually set the save path and device name. If **"Simultaneously create VHDL" is** checked

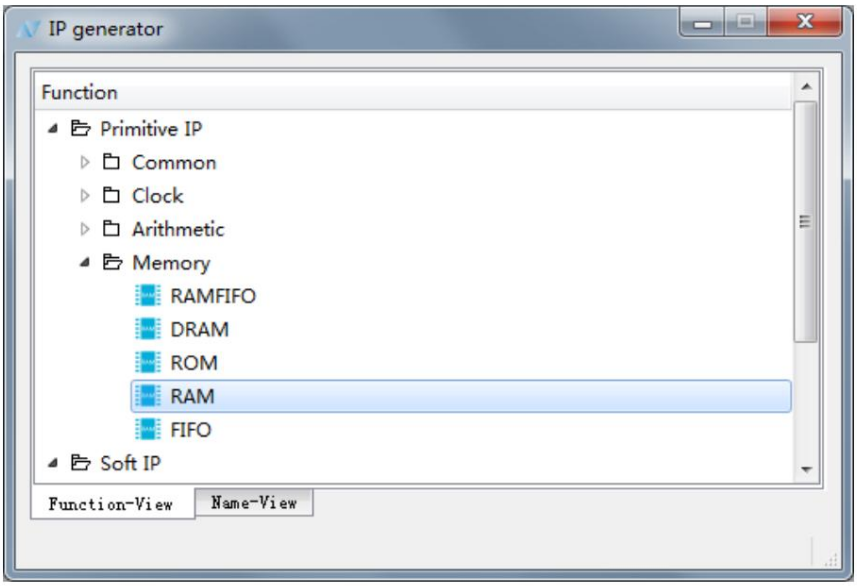
file", TD will generate the corresponding VHDL file.



Machine Translated by Google

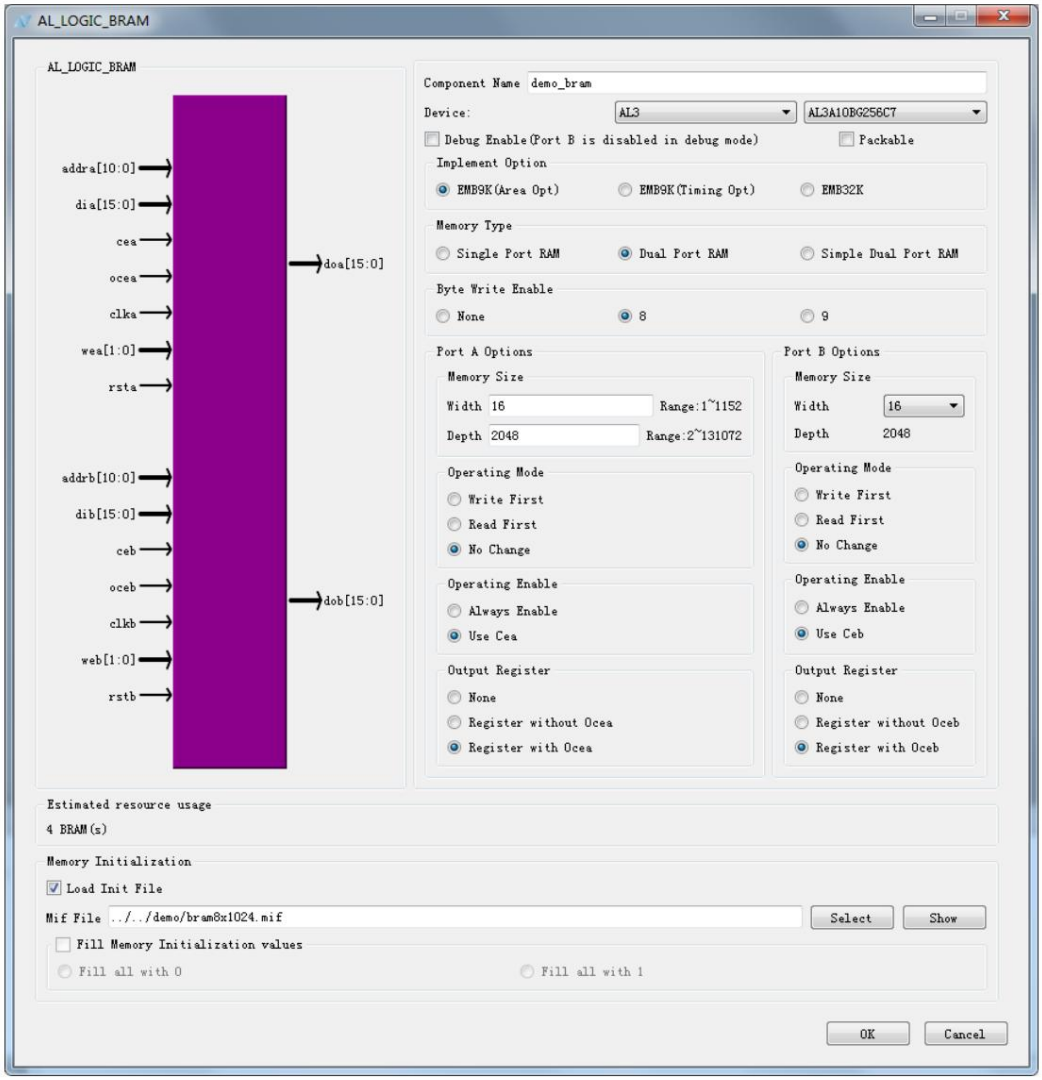
6. In the Function window **Primitive IP ,** expand **Memory ÿ** RAM, and double-click **RAM** to open the configuration interface

noodle



7. Fill **in "Component** Name" and set the corresponding parameters

Machine Translated by Google



This manual takes EMB9K as an example to introduce the use of BRAM modules of AL3 series devices.

EMB9K enables:

ÿ Single Port RAM

ÿ Dual Port RAM

ÿ Simple dual-port RAM (Simple Dual RAM, also known as pseudo dual-port)

The features supported by the EMB9K module are:

ÿ 9216 (9K) bits per block

ÿ A/B port clock is independent

ÿ A/B port data bit width can be configured separately, true dual port from x1 to x9, support x18 simple dual port (one write one

Machine Translated by Google

read)

ÿ 9 or 18-bit write operation with Byte Enable control

ÿ Selectable output latch (support 1-stage pipeline)

ÿ Support data initialization in RAM mode (EMB9K is

data initialization)

ÿ Supports multiple write operation modes. You can choose to write only (No Change), read first and then write (Read First), write first

After reading (Write First) three modes

ÿ Support Byte Enable function.

If the check box in front of "Debug Enable" is checked, the TD will default the EMB mode to Single Port RAM,

In this case, port B will be occupied, and the data of port A can be read back, which is convenient for users to use BramEditor

Debug. Among them, EMB9k is mainly for area optimization, and EMB9k (fast) is mainly for timing optimization.

Byte Enable means that when the input data port bit width of BRAM is multiple bytes, a set of data is used when reading data.

byte enable signal to control whether each byte is written or not. Select **Byte Write Enable** on the interface

The value is None or 8 or 9. When byte-write is None, it means that the byte enable function is not enabled; when

When byte-write is 8, the data width of port A and port B (if there is port B) must be an integer multiple of 8.

The value is used as the width of wea and web; when byte-write is 9, the data of port A and port B (if there is port B)

The width must be a multiple of 9, which is used as the width for wea and web. When the byte enable function is activated,

It is not recommended to use BRAM32K, because when the depth of BRAM is relatively small, a lot of memory will be wasted.

8. Add initialization file

The initialization file of TD supports users to describe in third-party mif (memory initialization file) format, or use

The verilog storage space is initialized in dat format to describe.

ÿ The mif format is described as follows:

Machine Translated by Google

The initialization file in mif format contains each initialization address and data, and must define the depth of the memory data.

degree and width. User can define data and address format as Binary BIN, Hexadecimal HEX, Octal

Decimal OCT, Unsigned Decimal UNS, etc. The value of the data must match the data format.

ÿ The dat format is described as follows:

Memory data can be stored in a file with a hexadecimal address, where the address is denoted by "@". rise

The start address is defined by the user, and the end address can be determined according to the depth of the memory data. in order to be able to

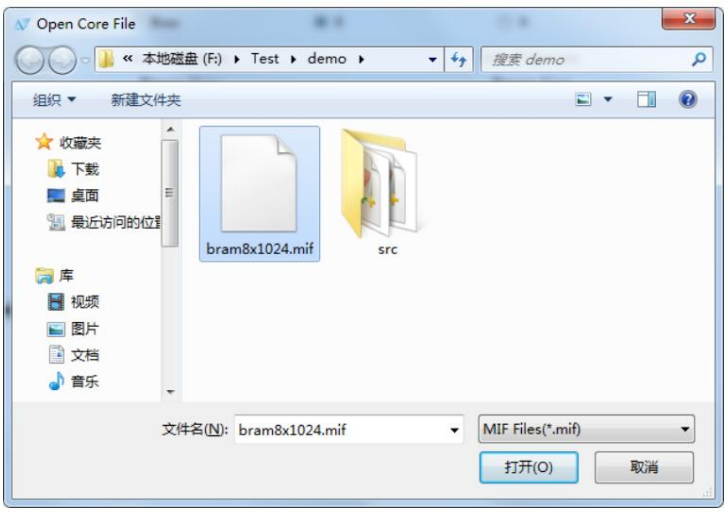
Clear correspondence between data and addresses, often adding a recognizable address flag to the file. If the initialization file is large

When , the address can also be omitted directly.

When adding an initialization file for the BRAM module, you can tick the check box in front of "Load **Init** File" and select the desired

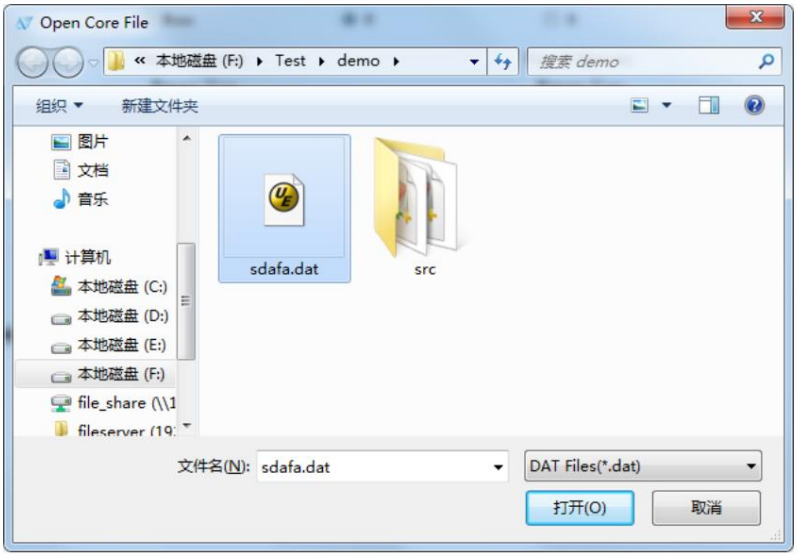
The file to be added, when the .mif format is selected in the drop-down box in the lower right corner, only the .mif file is provided in the folder for

User selection, as shown in the following figure.



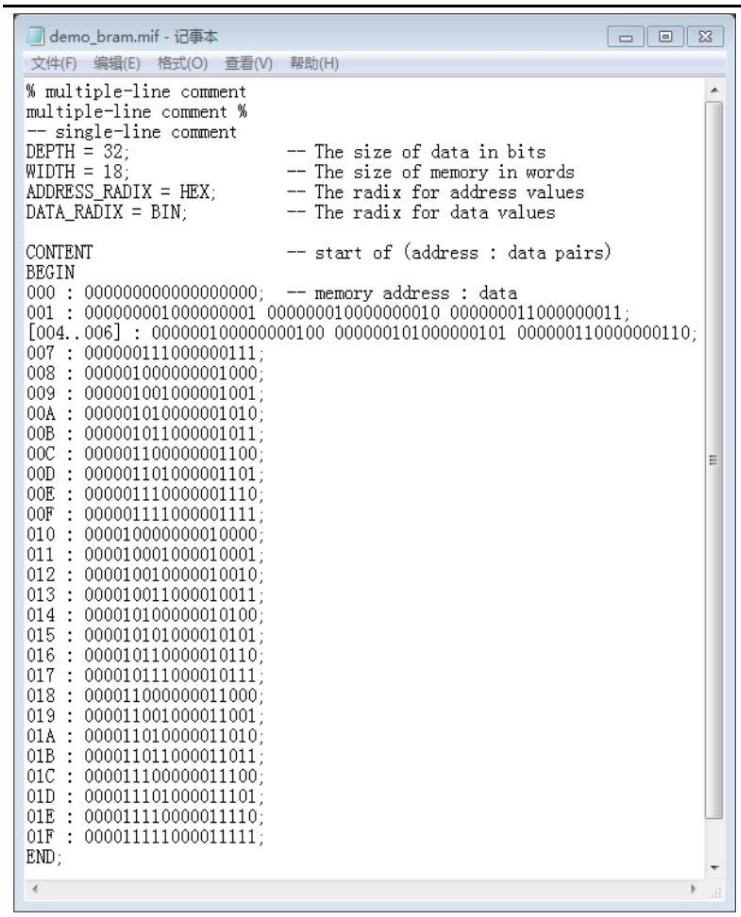
If the .dat format is selected, only .dat files are provided in the folder for users to choose, as shown in the following figure.

Machine Translated by Google

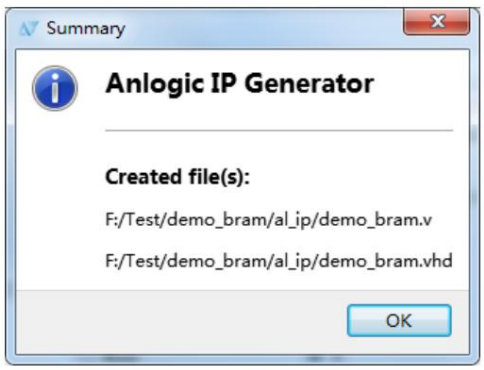


Click the button "Show" to see the content of the added file, here is the format of the mif file and the dat file formula example.

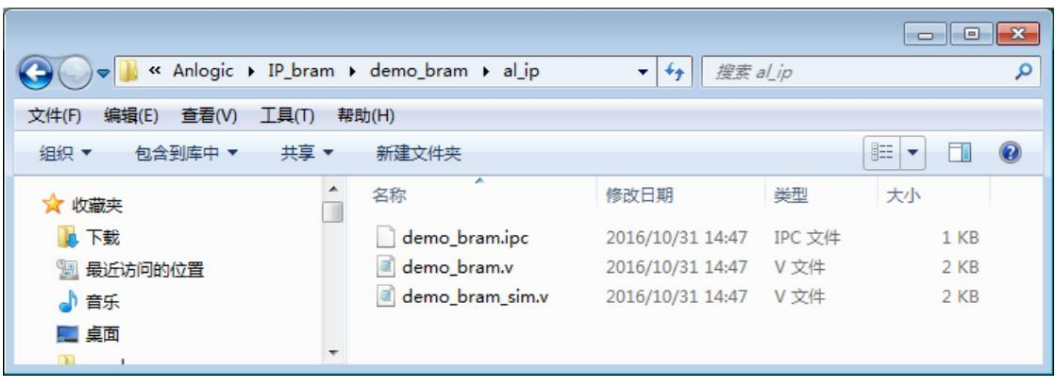
Machine Translated by Google



Click "OK" to complete the creation of the BRAM, and the path to the generated file given by TD is as follows:



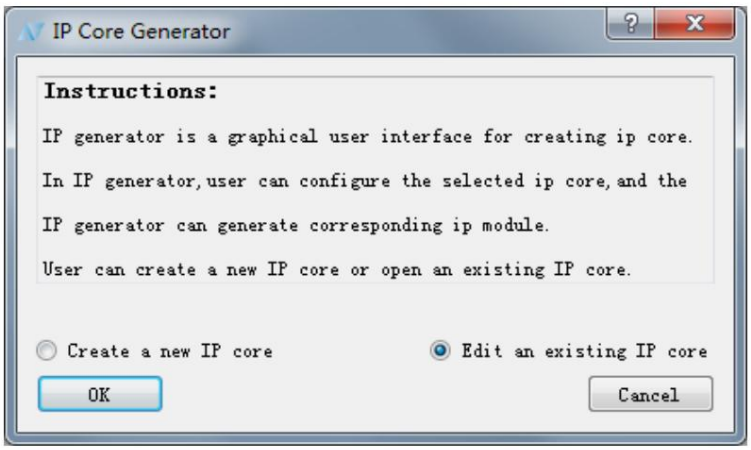
In the project directory, you can see the following files:



Machine Translated by Google

Among them, demo_bram.ipc is the project file generated by IP Generator, which can be opened as follows:

Select **Tools ÿ IP Generator and select "Edit an exist IP core".**



demo_bram.v Creates a file for the user to instantiate the BRAM module.

demo_bram_sim.v describes the division of BRAM for users to simulate and debug, but it cannot be added.

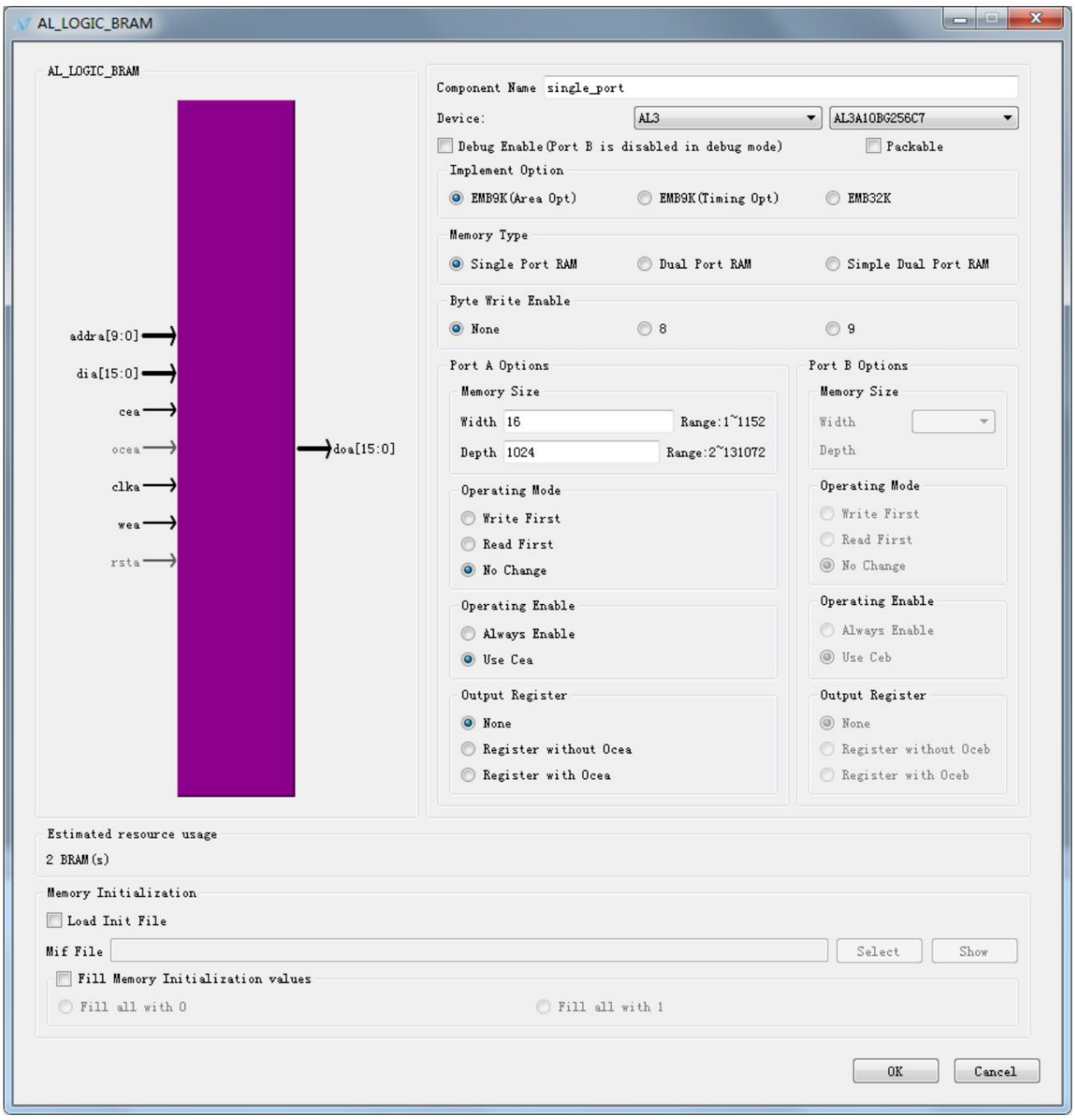
Add it as the source file of the project, otherwise it will conflict with the module in demo_bram.v.

Machine Translated by Google

The following introduces the interface settings of three different modes of BRAM and examples of generated files:

1. Single Port RAM

Single-port mode supports non-simultaneous read or write operations to the same address. The interface settings are as follows:



Machine Translated by Google

The files generated in stand-alone mode are as follows:

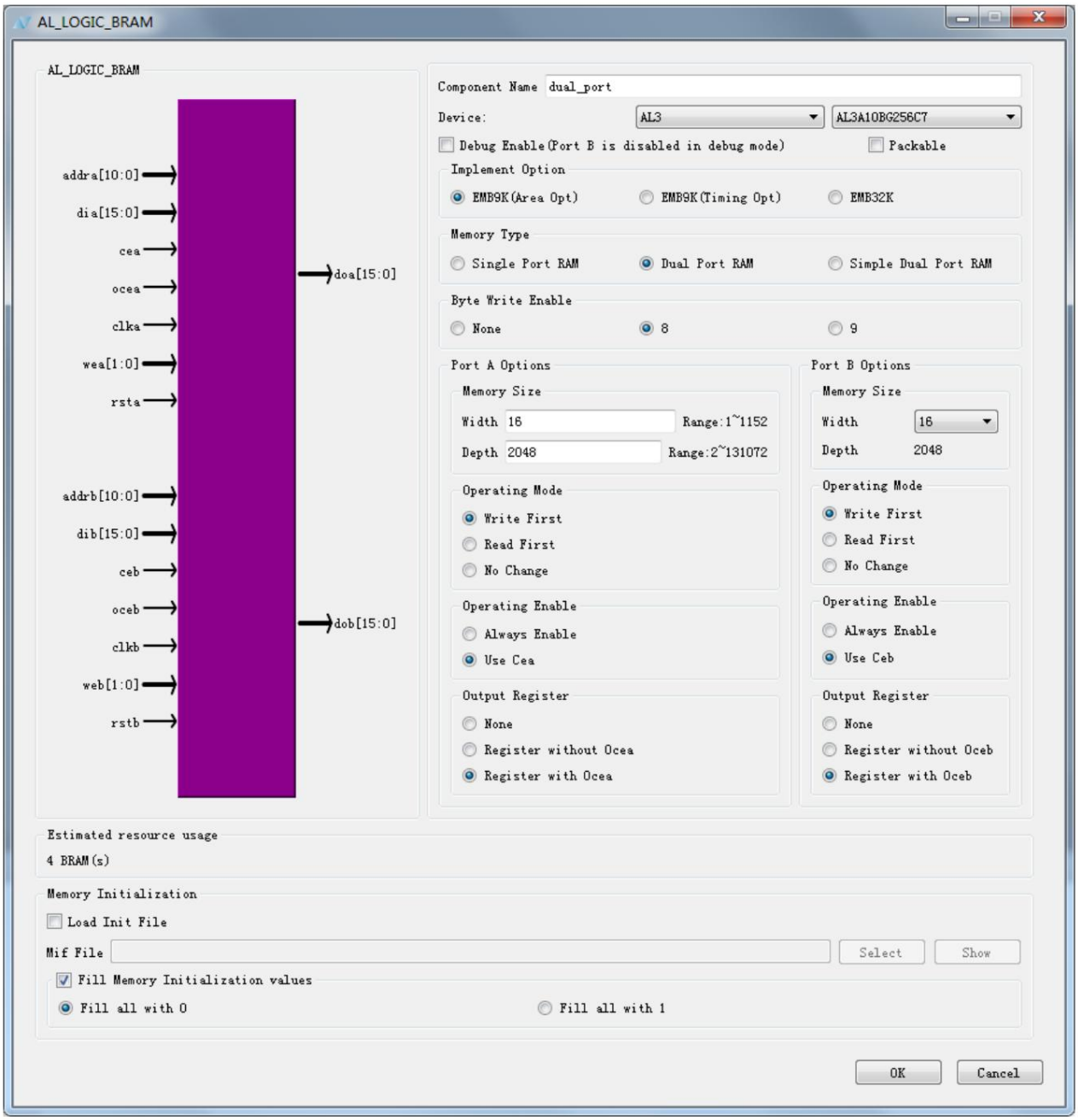
```
10  \*************************************************************/
11
12  `timescale 1ns / 1ps
13
14  module single_port ( doa, dia, addra, cea, clka, wea, rsta );
15
16      output [15:0] doa;
17
18      input  [15:0] dia;
19      input  [9:0] addra;
20      input  wea;
21      input  cea;
22      input  clka;
23      input  rsta;
24
25      AL_LOGIC_BRAM #( .DATA_WIDTH_A(16),
26                       .ADDR_WIDTH_A(10),
27                       .DATA_DEPTH_A(1024),
28                       .DATA_WIDTH_B(16),
29                       .ADDR_WIDTH_B(10),
30                       .DATA_DEPTH_B(1024),
31                       .MODE("SP"),
32                       .REGMODE_A("NOREG"),
33                       .WRITEMODE_A("NORMAL"),
34                       .RESETMODE("SYNC"),
35                       .IMPLEMENT("9K"),
36                       .DEBUGGABLE("NO"),
37                       .PACKABLE("NO"),
38                       .INIT_FILE("NONE"),
39                       .FILL_ALL("NONE"))
40                  inst(
41                       .dia(dia),
42                       .dib({16{1'b0}}),
43                       .addra(addra),
44                       .addrb({10{1'b0}}),
45                       .cea(cea),
46                       .ceb(1'b0),
47                       .ocea(1'b0),
48                       .oceb(1'b0),
49                       .clka(clka),
50                       .clkb(1'b0),
51                       .wea(wea),
52                       .web(1'b0),
53                       .bea(1'b0),
54                       .beb(1'b0),
55                       .rsta(rsta),
56                       .rstb(1'b0),
57                       .doa(doa),
58                       .dob());
59
60  endmodule
```

Machine Translated by Google

2. Dual Port RAM

Dual port mode supports all combinations of independent read and write operations for port A/B: two reads, two writes, one read and one write.

The interface settings are as follows:



Machine Translated by Google

The files generated in dual-port mode are as follows:

```
13
14  module dual_port (
15      doa, dia, addra, cea, clka, wea, rsta, ocea,
16      dob, dib, addrb, ceb, clkb, web, rstb, oceb
17  );
18      output [15:0] doa;
19      output [15:0] dob;
20      input  [15:0] dia;
21      input  [15:0] dib;
22      input  [10:0] addra;
23      input  [10:0] addrb;
24      input  [1:0] wea;
25      input  [1:0] web;
26      input  cea;
27      input  ceb;
28      input  clka;
29      input  clkb;
30      input  rsta;
31      input  rstb;
32      input  ocea;
33      input  oceb;
34
35      AL_LOGIC_BRAM #( .DATA_WIDTH_A(16),
36                  .DATA_WIDTH_B(16),
37                  .ADDR_WIDTH_A(11),
38                  .ADDR_WIDTH_B(11),
39                  .DATA_DEPTH_A(2048),
40                  .DATA_DEPTH_B(2048),
41                  .BYTE_ENABLE(8),
42                  .BYTE_A(2),
43                  .BYTE_B(2),
44                  .MODE("DP"),
45                  .REGMODE_A("OUTREG"),
46                  .REGMODE_B("OUTREG"),
47                  .WRITEMODE_A("WRITETHROUGH"),
48                  .WRITEMODE_B("WRITETHROUGH"),
49                  .RESETMODE("SYNC"),
50                  .IMPLEMENT("9K"),
51                  .INIT_FILE("NONE"),
52                  .FILL_ALL("0"))
53              inst(
54                  .dia(dia),
55                  .dib(dib),
56                  .addra(addra),
57                  .addrb(addrb),
58                  .cea(cea),
59                  .ceb(ceb),
60                  .ocea(ocea),
61                  .oceb(oceb),
62                  .clka(clka),
63                  .clkb(clkb),
64                  .wea(1'b0),
```

Machine Translated by Google

3. Simple Dual Port RAM

When an EMB9K is configured as 18-bit write or 18-bit read, the EMB9K does not support dual-port mode, but

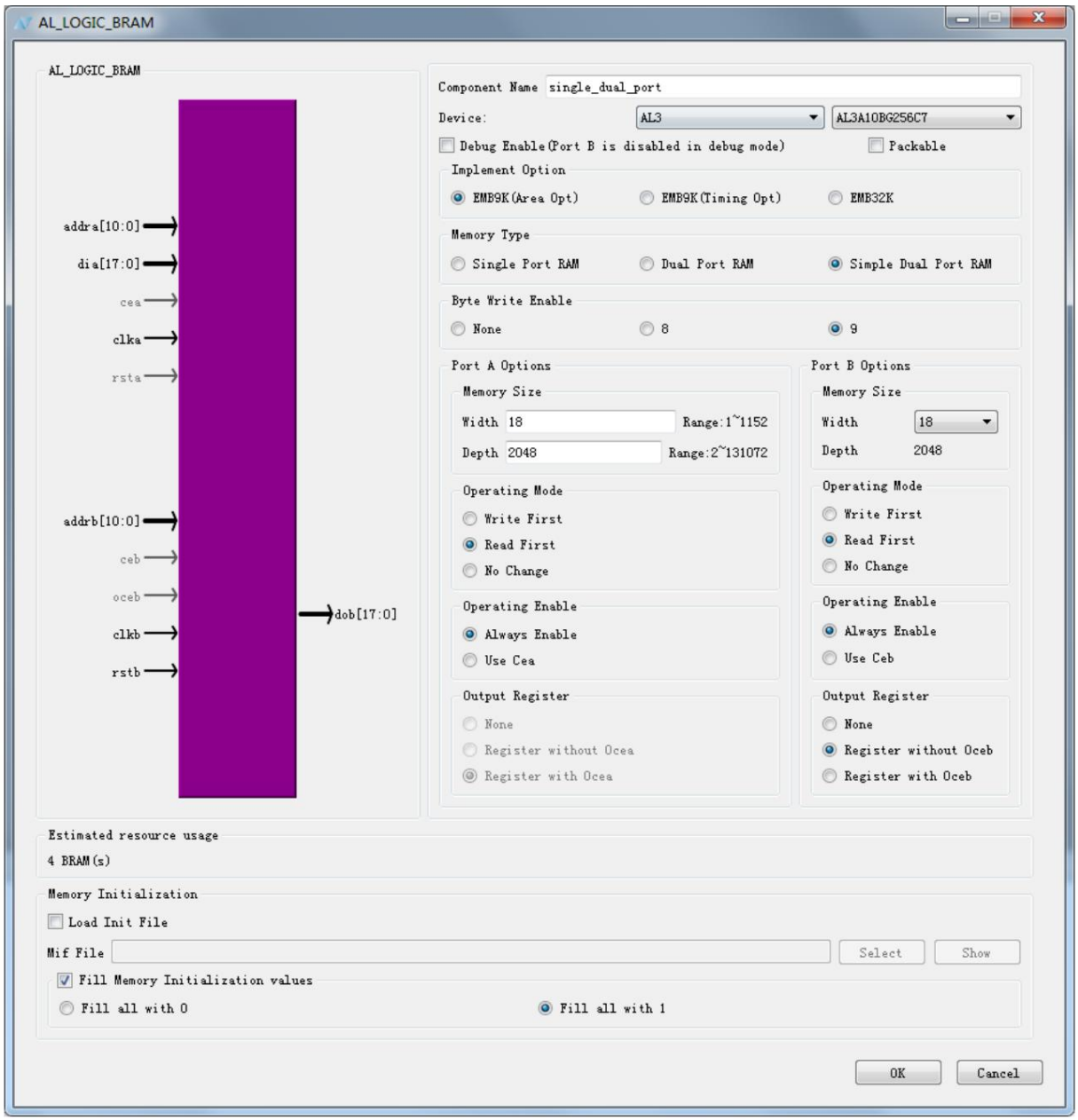
Supports single-port and simple dual-port modes. When EMB9K is in 18-bit mode, the A port control signal is written as

Control signal, B port control signal as readout control signal. When the EMB9K is configured for 18-bit writes,

dib[8:0] is input as high 9-bit data, dia[8:0] is input as low 9-bit data; when EMB9K is configured as 18

When the bit is read, dob[8:0] is output as the upper 9-bit data, and doa[8:0] is output as the lower 9-bit data.

The interface settings of simple dual-port mode are as follows:



Machine Translated by Google

The files generated in simple dual-port mode are as follows:

```
13
14  module single_dual_port (
15      dia, addra, clka, rsta, wea,
16      dob, addrb, clkb, rstb
17  );
18      output [17:0] dob;
19      input  [17:0] dia;
20      input  [10:0] addra;
21      input  [10:0] addrb;
22      input  [1:0] wea;
23      input  clka;
24      input  clkb;
25      input  rsta;
26      input  rstb;
27
28      AL_LOGIC_BRAM #( .DATA_WIDTH_A(18),
29                       .DATA_WIDTH_B(18),
30                       .ADDR_WIDTH_A(11),
31                       .ADDR_WIDTH_B(11),
32                       .DATA_DEPTH_A(2048),
33                       .DATA_DEPTH_B(2048),
34                       .BYTE_ENABLE(9),
35                       .BYTE_A(2),
36                       .BYTE_B(2),
37                       .MODE("PDPW"),
38                       .REGMODE_A("OUTREG"),
39                       .REGMODE_B("OUTREG"),
40                       .WRITEMODE_A("READBEFOREWRITE"),
41                       .WRITEMODE_B("READBEFOREWRITE"),
42                       .RESETMODE("SYNC"),
43                       .IMPLEMENT("9K"),
44                       .INIT_FILE("NONE"),
45                       .FILL_ALL("1"))
46                 inst(
47                       .dia(dia),
48                       .dib({18{1'b0}}),
49                       .addra(addra),
50                       .addrb(addrb),
51                       .cea(1'b1),
52                       .ceb(1'b1),
53                       .ocea(1'b0),
54                       .oceb(1'b1),
55                       .clka(clka),
56                       .clkb(clkb),
57                       .wea(1'b0),
58                       .bea(wea),
59                       .rsta(rsta),
60                       .rstb(rstb),
61                       .doa(),
62                       .dob(dob));
63  endmodule
```

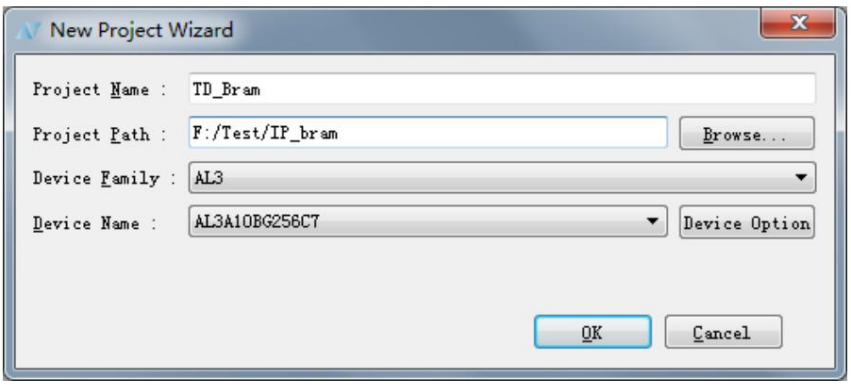
Machine Translated by Google

3.5.2 Instantiate the **BRAM** module

This manual takes a new project as an example to introduce the process of instantiating a BRAM module. Users can also use the base of existing projects

On the basis of instantiation, the instantiation process is consistent.

1. Create a new project and add a top-level module to the project.



2. Add the demo_bram.v generated in the previous step to the project

3. Call the demo_bram module in the top-level module, and modify the inst name and port name, click Save

button to complete the instantiation of the BRAM module.

```
1  module top( doa, dia, addra, cea, ocea, clka, wea, rsta );
2
3       output [1023:0] doa;
4
5
6       input  [1023:0] dia;
7       input  [10:0] addra;
8       input  cea;
9       input  ocea;
10      input  clka;
11      input  wea;
12      input  rsta;
13
14      demo_bram uut(
15            .dia(dia),
16            .addra(addra),
17            .cea(cea),
18            .clka(clka),
19            .wea(wea),
20            .rsta(rsta),
21            .doa(doa));
22
23 endmodule
```

Machine Translated by Google

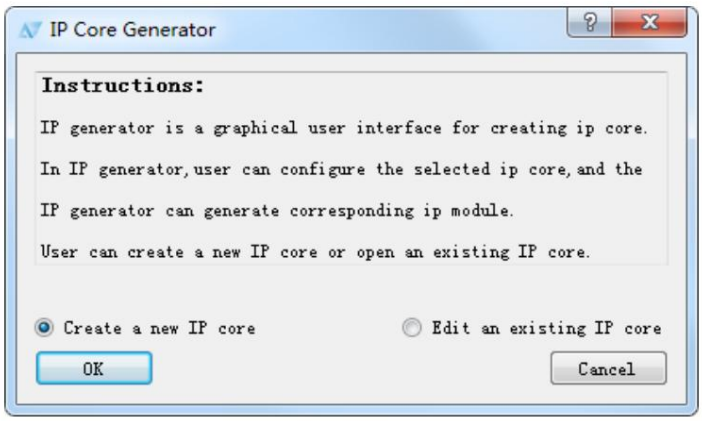
3.6 FIFO module

EMB9k internal hardware supports synchronous/asynchronous FIFO mode. EMB9K bit width setting and simple in FIFO mode

The dual-port RAM settings are the same, and can support up to 18bit width.

3.6.1 Create FIFO module

1. Select **Tools ÿ IP** Generator, select "Create **a new IP core"**

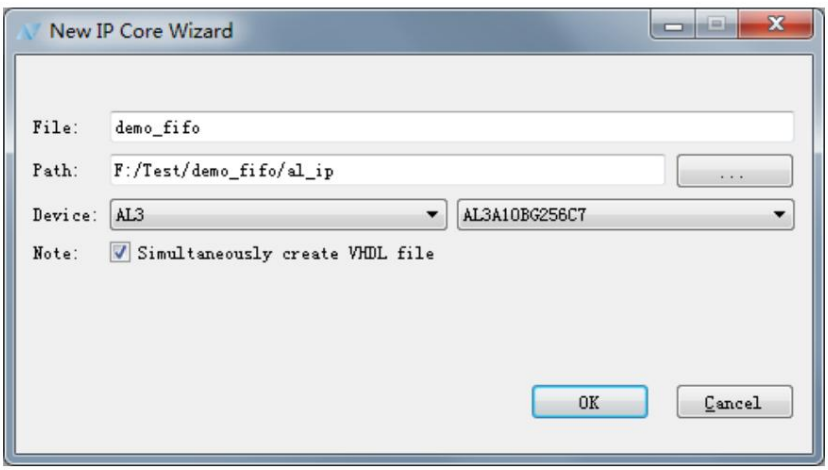


2. Enter the module name and select the storage path. Here, if the FIFO module is created on the basis of a project,

The storage path and device name will be consistent with the project. If you create a FIFO mode without a project

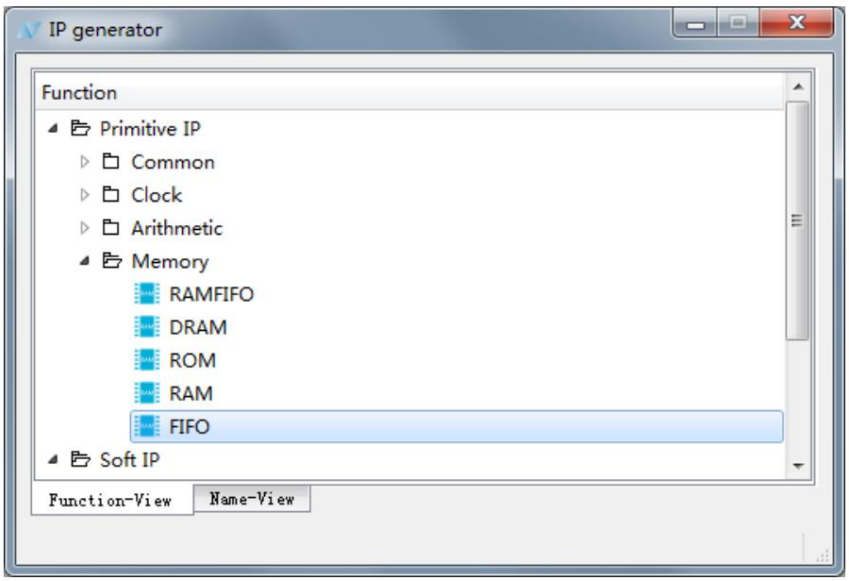
block, the user needs to manually set the save path and device name. If **"Simultaneously create VHDL" is** checked

file", TD will generate the corresponding VHDL file.

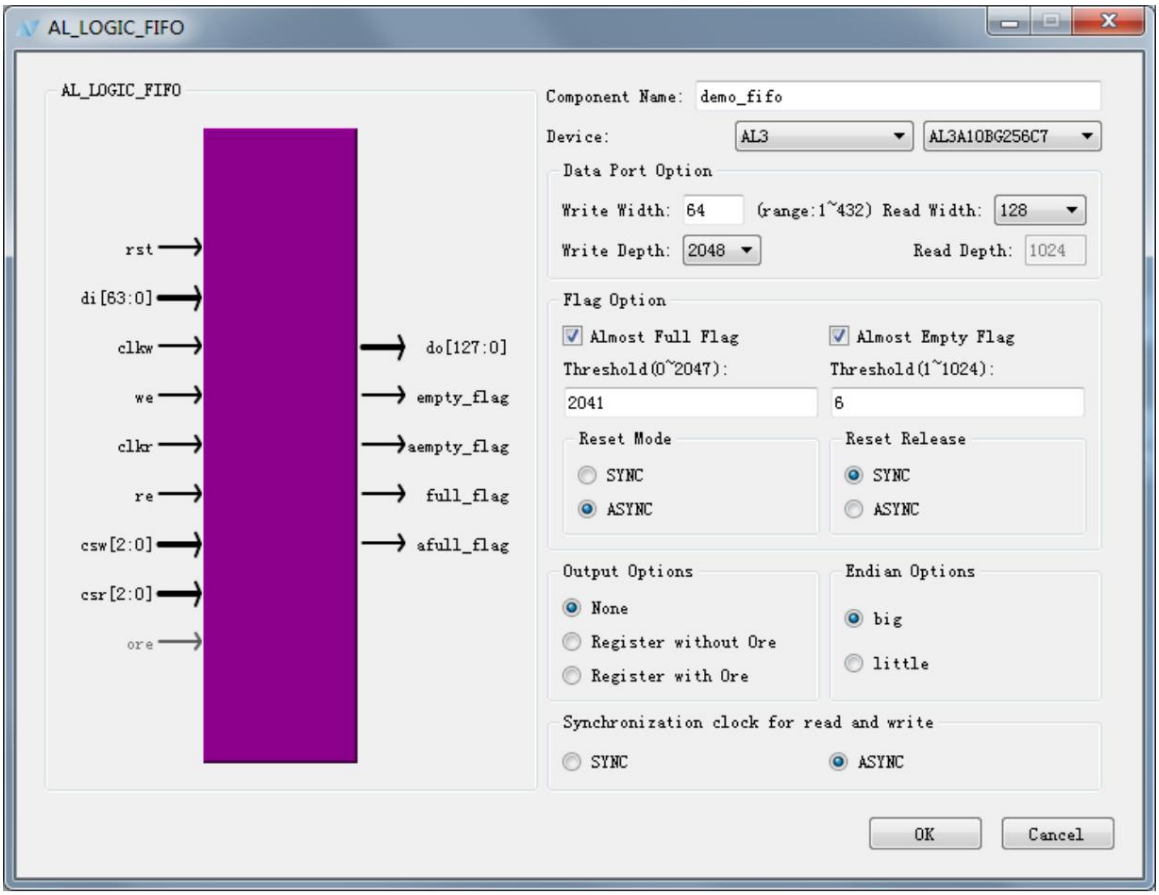


Machine Translated by Google

3. In the Function window **Primitive IP ,** expand **MemoryÿFIFO, and double-click FIFO** to open the configuration interface



4. Fill **in the "Component** Name" and set the corresponding parameters



Machine Translated by Google

Endian Options is the endian mode of the output data, which is embodied as:

Endian Options : little mode:

When the written data is: AA,BB,CC,DD, then read as BB_AA,DD_CC

When the written data is: BB_AA,DD_CC, then read out as AA,BB,CC,DD

Endian Options: In big mode:

When the written data is: AA,BB,CC,DD, then read out as AA_BB, CC_DD

When the written data is: AA_BB,CC_DD, then read as AA,BB,CC,DD

Description of empty full flag in FIFO mode:

FIFO flag name	Direction setting range description		
Empty_Flag (EF) output 0			FIFO read empty flag, synchronized with clkr
Aempty_Flag (AE) output 1 to FF-1			FIFO almost empty flag, synchronized with clkr, The relative read lead is determined by the AE_POINTER parameter
Full_Flag (FF) Input 1 to Max	FIFO full flag, synchronized with clkr,		FIFO full capacity is determined by the FULL_POINTER parameter
Afull_Flag (AF) Input 1 to FF-1			FIFO almost full flag, synchronized with clkw, The FIFO is almost full determined by the AF_POINTER parameter

ÿ Setting of empty full flag

In FIFO mode, the user can set the FIFO empty flag attributes through software: Empty flag (Empty_Flag),

Almost empty flag (Almost_Empty), full flag (Full_Flag), almost full flag (Almost_Full). when the internal count

When the counter reaches the flag value, the port corresponding to the corresponding empty and full pointer EF/AE/FF/AF will output a high level.

ÿ Explanation of the empty and full flag pointer

Empty pointer (EF) is fixed to 0 in FIFO mode , AE/AF/FF pointer is a 14-bit binary number

"0b0X_XXXX_XXXX_XXXX", the highest bit [13] is always 0, the valid bits [12:0], representing 8K depth 1bit

width. The validity of the lowest 4 bits[3:0] of the pointer depends on the read/write bit width.

Machine Translated by Google

ÿ Use csw/csr reverse configuration to implement simple FIFO

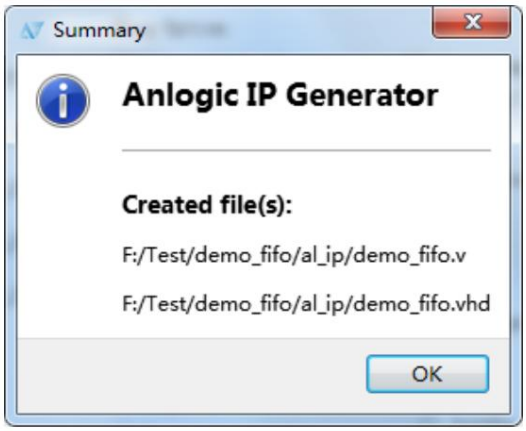
csw is the 3-bit chip select signal of the FIFO write port, which can be reversed; csr is the 3-bit chip select signal of the FIFO read port,

Can be reversed. In order to avoid pointer overflow when the FIFO is full or read, the full signal can be reversed through the interconnect resource.

It is connected to the csw end backward, and the null signal is connected to the csr end after the reverse direction. Reverse logic can take advantage of the reverse inside csw/csr

implemented with logic.

5. Click "OK" to complete the FIFO setting, the TD will give the path of the generated file.



You can also open and edit the existing demo_fifo.ipc through "Edit **an existing IP** core".

Machine Translated by Google

3.6.2 Instantiating the **FIFO** module

This manual takes a new project as an example to introduce the process of instantiating the FIFO module. Users can also build on existing projects

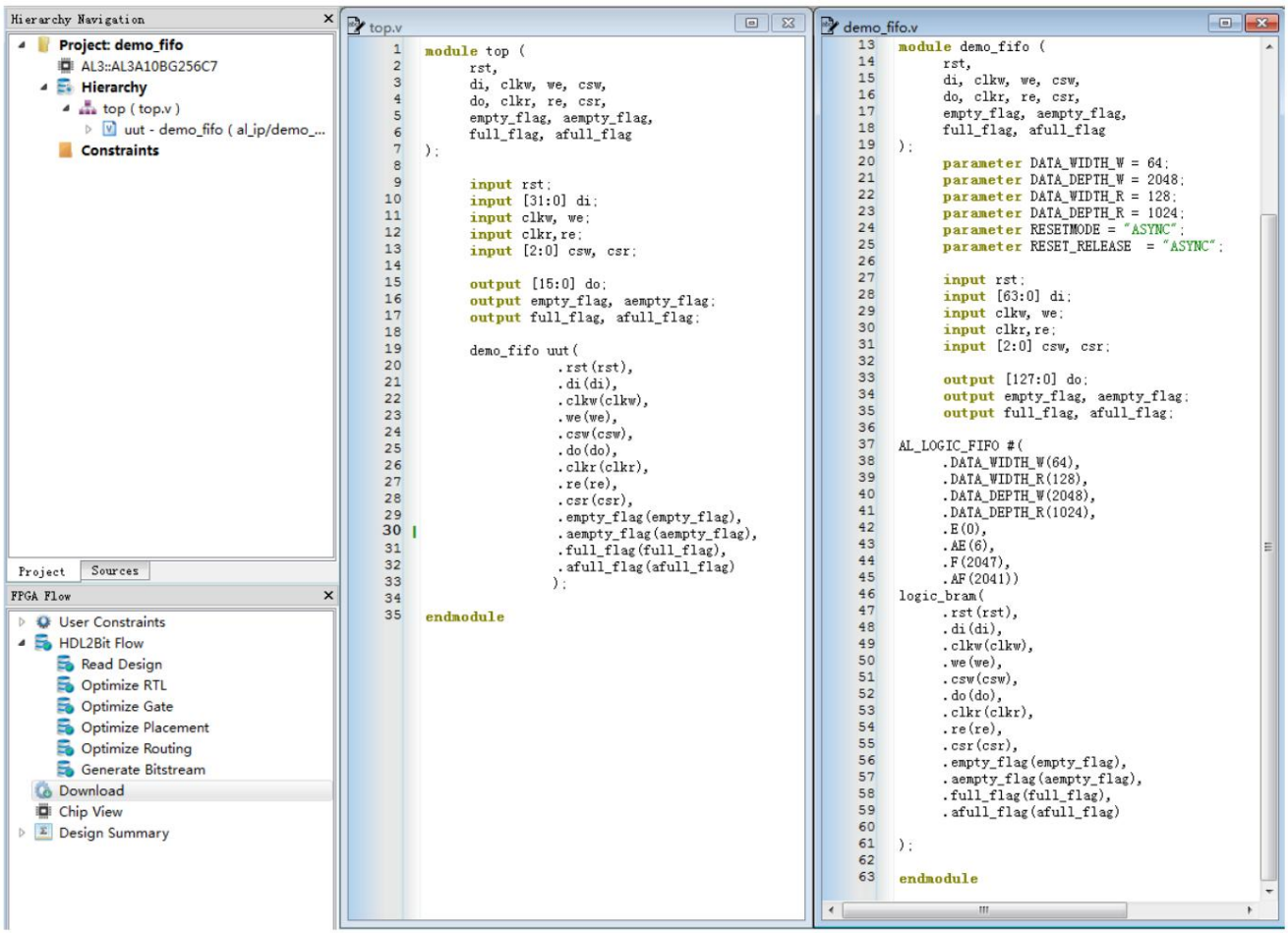
The instantiation is performed on the above, and the instantiation process is the same.

1. Create a new project and add a top-level module to the project.

2. Add the demo_fifo.v generated in the previous step to the project

3. Call the demo_fifo module in the top-level module, and modify the inst name and port name, click the save button, that is

The instantiation of the FIFO module is complete.



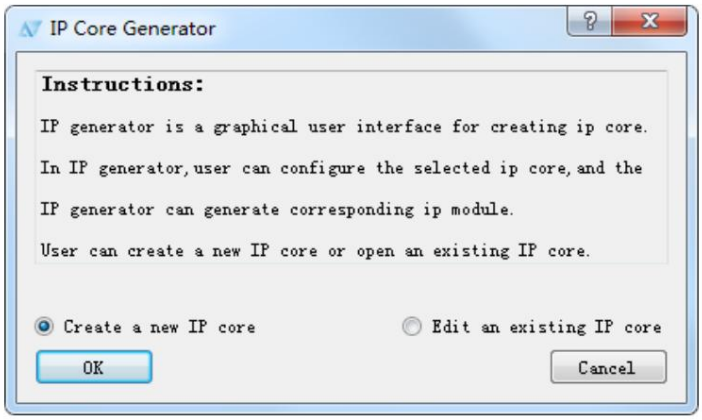
Machine Translated by Google

3.7 RAMFIFO module

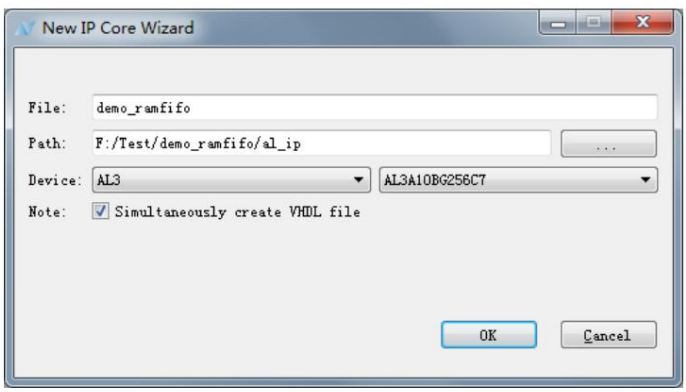
The RAMFIFO module is an asynchronous FIFO module based on on-chip RAM storage resources. When the module supports async Clock read and write, asynchronous reset function, empty/full signal indication function, read and write data number indication function, SHOWAHEAD function, with good timing characteristics and reasonable resource consumption.

3.7.1 Create RAMFIFO module

1. Select **Tools** ÿ **IP** Generator, select "Create **a new IP core"**



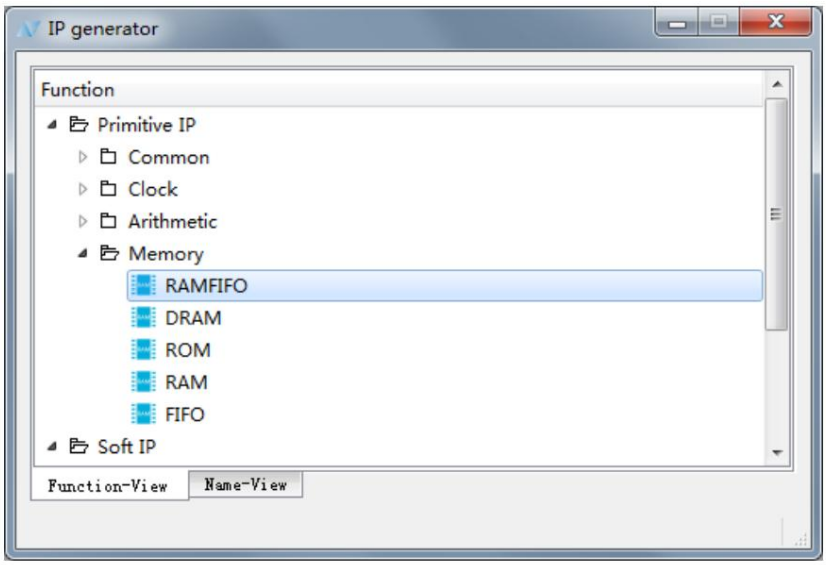
2. Enter the module name and select the storage path. Here, if the RAMFIFO module is created on the basis of a project Blocks, storage paths and device names will be the same as the project. If created without a project RAMFIFO module, the user needs to manually set the save path and device name. If **"Simultaneously create VHDL file",** TD will generate the corresponding VHDL file.



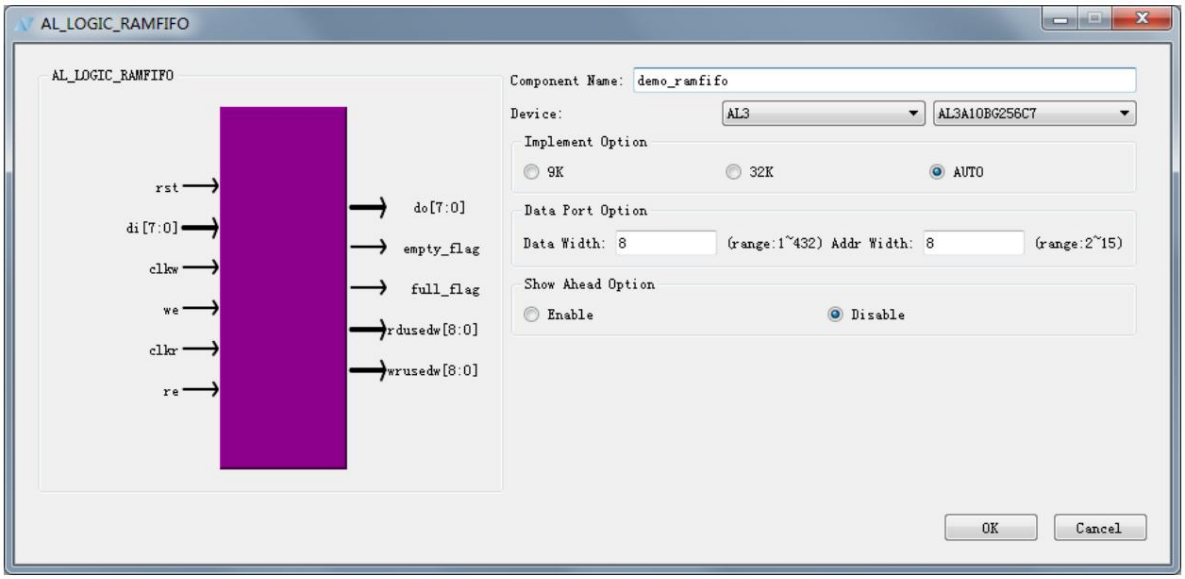
Machine Translated by Google

3. In the Function window **Primitive IP ,** expand **Memory** ÿ **RAMFIFO, double-click RAMFIFO** to open

Open configuration interface



4. Fill **in the "Component** Name" and set the corresponding parameters



The parameter definitions are shown in the following table:

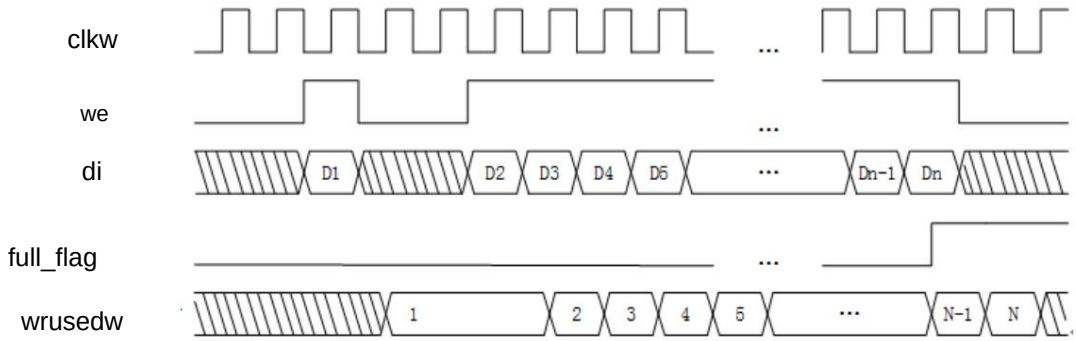
Parameter Description	
DATA_WIDTH Read and write	FIFO data bit width
ADDR_WIDTH Read and write	FIFO address width, FIFO depth is 2 ADDR_WIDTH times square
SHOWAHEAD	SHOWAHEAD mode is enabled, 1 means SHOWAHEAD mode, 0 means normal pass mode

Machine Translated by Google

The port definitions are shown in the following table:

port		illustrate
rst	Function Asynchronous reset signal active high for resetting the entire module	
di	The data input bit width is determined by the parameter DATA_WIDTH	
clkw	The write clock signal provides a synchronous clock for the write side	
we	Write the data to the FIFO when the write enable signal is high and the FIFO is not full	
clkr	The read clock signal provides a synchronous clock for the read side	
re	Read data from the FIFO when the read enable signal is high and the FIFO is not empty	
do	Data output	out The bit width is determined by the parameter DATA_WIDTH
empty_flag empty indicator signal full indicator		When high, it means the FIFO is empty, and the read enable signal will be ignored
readable data bit width is determined by	signal full flag	When high, it means the FIFO is full, and the write enable signal will be ignored
rdusedw	parameter ADDR_WIDTH	
wrusedw	The bit width of the written data is determined by the parameter ADDR_WIDTH	

Write timing diagram:

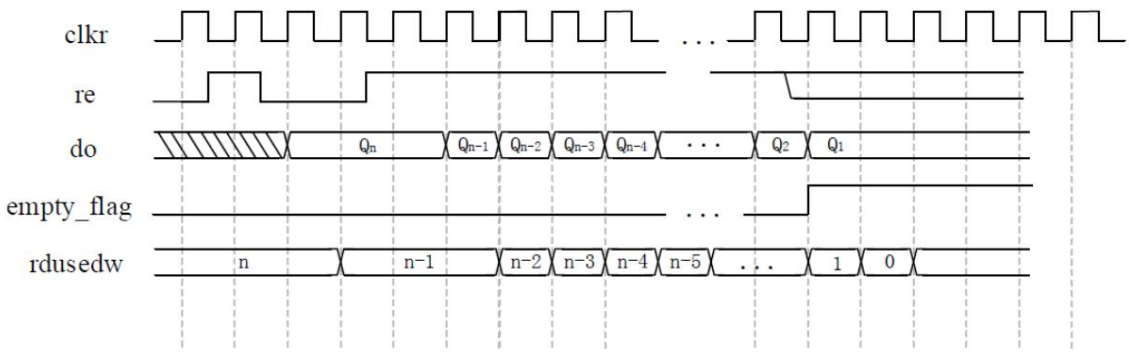


At the rising edge of the clock, if we is high and full_flag is low, write di at this time into the FIFO to complete a

The number of write data at the same time indicates that the signal wrusedw is incremented by 1 (wrusedw, as shown in the timing diagram, is delayed by one

clock cycle output), rdusedw is incremented by 1 after a number of clock delays (the specific delay is related to the read and write clock cycles).

Normal mode read timing diagram:



Machine Translated by Google

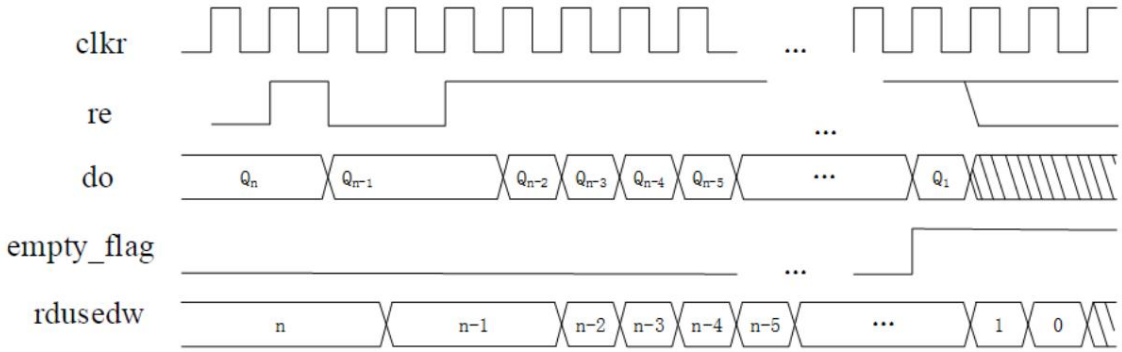
On the rising edge of the clock, if re is high and empty_flag is low, the FIFO is read out on the next clock cycle

When the data reaches do, a read operation is completed, and the indicator signal rdusedw of the number of read data is decremented by 1 at the same time.

As shown in the sequence diagram, the output is delayed by one clock cycle), and wrusedw is decremented by 1 after a number of clock delays (specific delay and read and write

clock cycles).

SHOWAHEAD mode read timing diagram:



On the rising edge of the clock, if re is high and empty_flag is low, the number of FIFOs is read out in the current clock cycle

According to the data to do, a read operation is completed, and the indicator signal rdusedw of the number of read data is decremented by 1 at the same time (rdusedw is the timing

As shown in the figure, the output is delayed by one clock cycle), and wrusedw is decremented by 1 after a number of clock delays (the specific delay and read/write time

clock cycle).

The only difference between the SHOWAHEAD mode read timing and the normal mode read timing is that the output data is

It has been prepared in do before, and the re signal is used as a response signal to inform the FIFO that the current data has been read and can be prepared.

next data.

5. Click "OK" to complete the setting of RAMFIFO, the TD will give the path of the generated file.

An existing IP core can be opened by selecting Tools ÿ IP Generator and selecting "Edit **an existing IP** core"

demo_ramfifo.ipc.

Machine Translated by Google

3.7.2 Instantiate the **RAMFIFO** module

This manual takes a new project as an example to introduce the process of instantiating the RAMFIFO module. Users can also

On the basis of instantiation, the instantiation process is consistent.

1. Create a new project and add a top-level module to the project.

2. Add the demo_ramfifo.v generated in the previous step to the project

3. Call the demo_ramfifo module in the top-level module, modify the inst name and port name, and click Save

   button to complete the instantiation of the RAMFIFO module.

top.v

```
module top (
    rst,
    di, clkr, re, clkw, we,
    do,
    empty_flag,full_flag,rdusedw,wrusedw

);

    input rst;
    input [7:0] di;
    input clkw, we;
    input clkr,re;

    output [7:0] do;
    output empty_flag;
    output full_flag;
    output [8:0] rdusedw;
    output [8:0] wrusedw;

demo_ramfifo ramfifo(
    .rst(rst),
    .di(di),
    .clkw(clkw),
    .we(we),
    .do(do),
    .clkr(clkr),
    .re(re),
    .empty_flag(empty_flag),
    .full_flag(full_flag),
    .rdusedw(rdusedw),
    .wrusedw(wrusedw)
);

endmodule
```

demo_ramfifo.v

```
module demo_ramfifo (
    rst,
    di, clkr, re, clkw, we,
    do,
    empty_flag,full_flag,rdusedw,wrusedw

);

    input rst;
    input [7:0] di;
    input clkw, we;
    input clkr,re;

    output [7:0] do;
    output empty_flag;
    output full_flag;
    output [8:0] rdusedw;
    output [8:0] wrusedw;

AL_LOGIC_RAMFIFO #(
    .DATA_WIDTH(8),
    .ADDR_WIDTH(8),
    .SHOWAHEAD(0),
    .IMPLEMENT("AUTO")
)
logic_ramfifo(
    .rst(rst),
    .di(di),
    .clkw(clkw),
    .we(we),
    .do(do),
    .clkr(clkr),
    .re(re),
    .empty_flag(empty_flag),
    .full_flag(full_flag),
    .rdusedw(rdusedw),
    .wrusedw(wrusedw)
);

endmodule
```

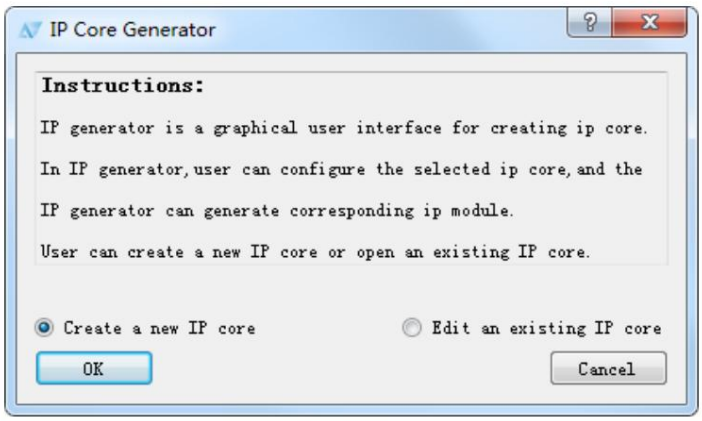
Machine Translated by Google

3.8 DRAM module

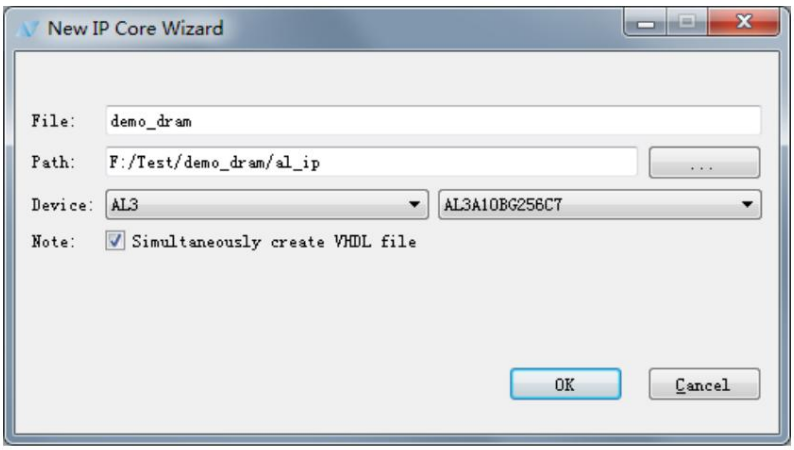
In AL3 series devices, each PLB contains 2 MSLICEs (4 LUTs), enabling 16x4 RAM block, each DRAM supports simple dual-port RAM.

3.8.1 Creating a DRAM Module

1. Select **Tools ÿ IP** Generator, select "Create **a new IP core"**



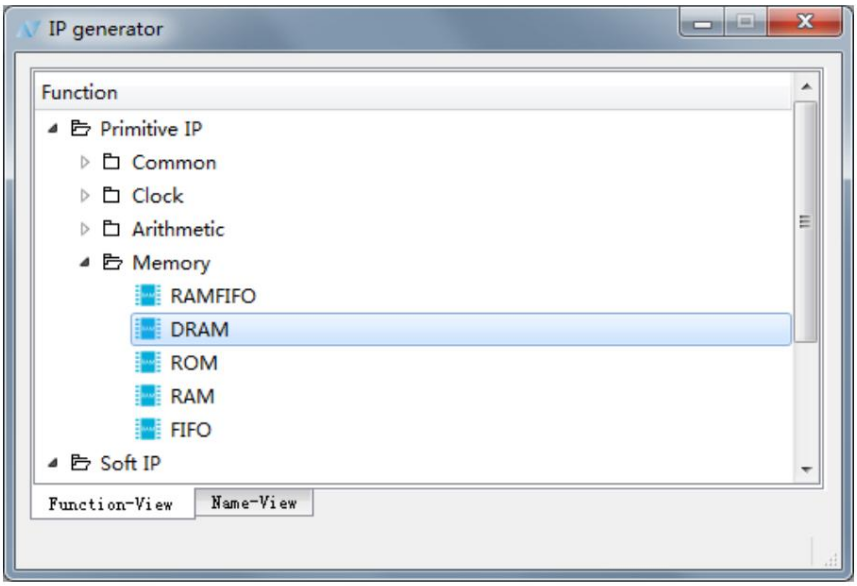
2. Enter the module name and select the storage path. Here, if a DRAM module is created on the basis of a project, The storage path and device name will be consistent with the project. If you create a DRAM module without engineering block, the user needs to manually set the save path and device name. If **"Simultaneously create VHDL" is** checked **file",** TD will generate the corresponding VHDL file.



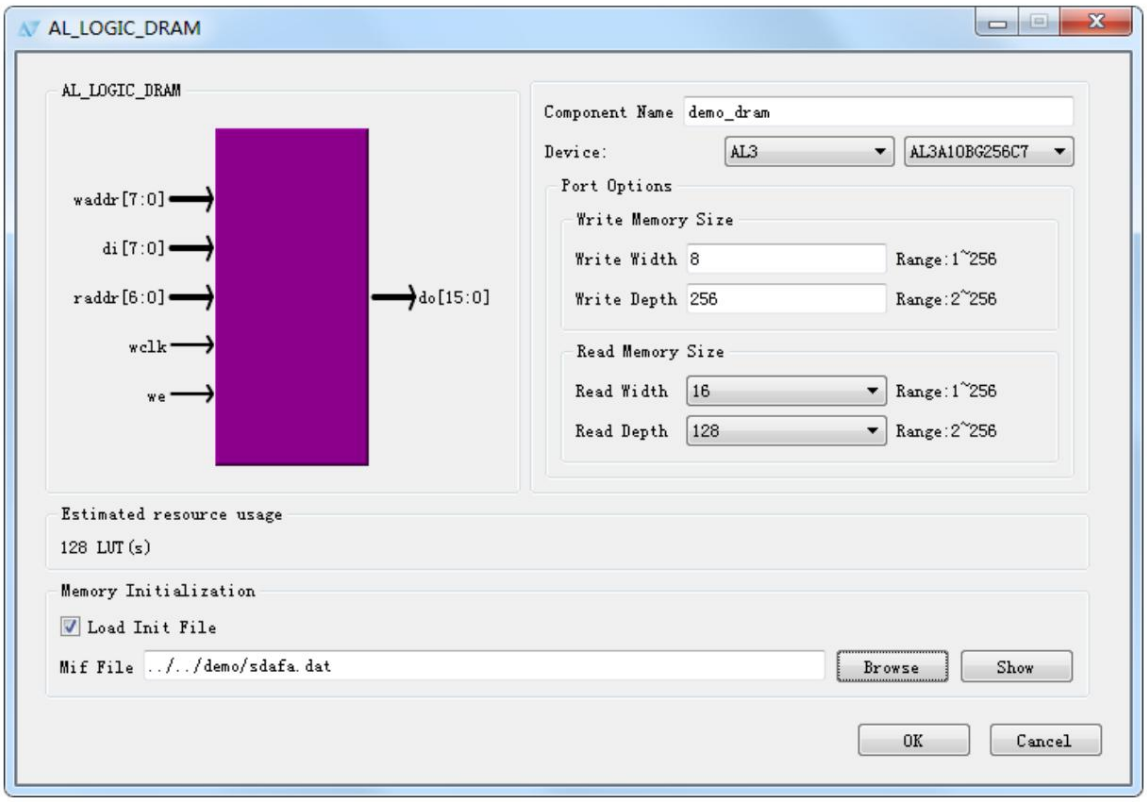
Machine Translated by Google

3. In the Function window **Primitive IP ,** expand **Memory** ÿ **DRAM, and double-click DRAM** to open the configuration

interface



4. Fill **in the "Component** Name" and set the corresponding parameters



Here, the format of Init File is the same as in BRAM.

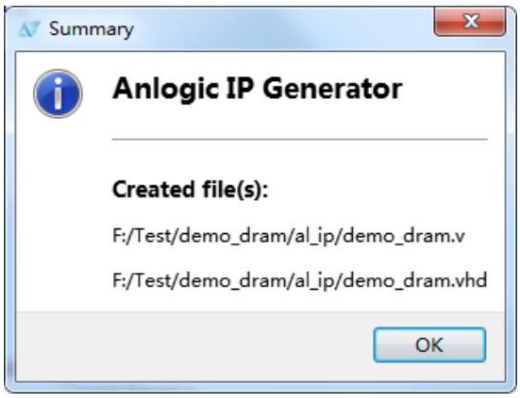
Machine Translated by Google

The port list and description are as follows:

Among them, the write port is synchronized by WCLK, and the read port works asynchronously.

port	Function	illustrate
WCLK	Write Operation Clock	Active on rising edge (programmable inversion)
WE	Write Enable Write	Embedded WCLK rising edge synchronization latch
WADDR[3:0]	Address Write Data	Embedded WCLK rising edge synchronization latch
DI[3:0]	Read Address Read	Embedded WCLK rising edge synchronization latch
RADDR[3:0]	Data	asynchronous
DO[3:0]		asynchronous

5. Click "OK" to complete the DRAM setting, the TD will give the path to generate the file.



You can also open and edit the existing demo_dram.ipc through "Edit **an existing IP** core".

Machine Translated by Google

3.8.2 Instantiating a **DRAM** Module

This manual introduces the process of instantiating a DRAM module by taking a new project as an example. Users can also use the base of existing projects

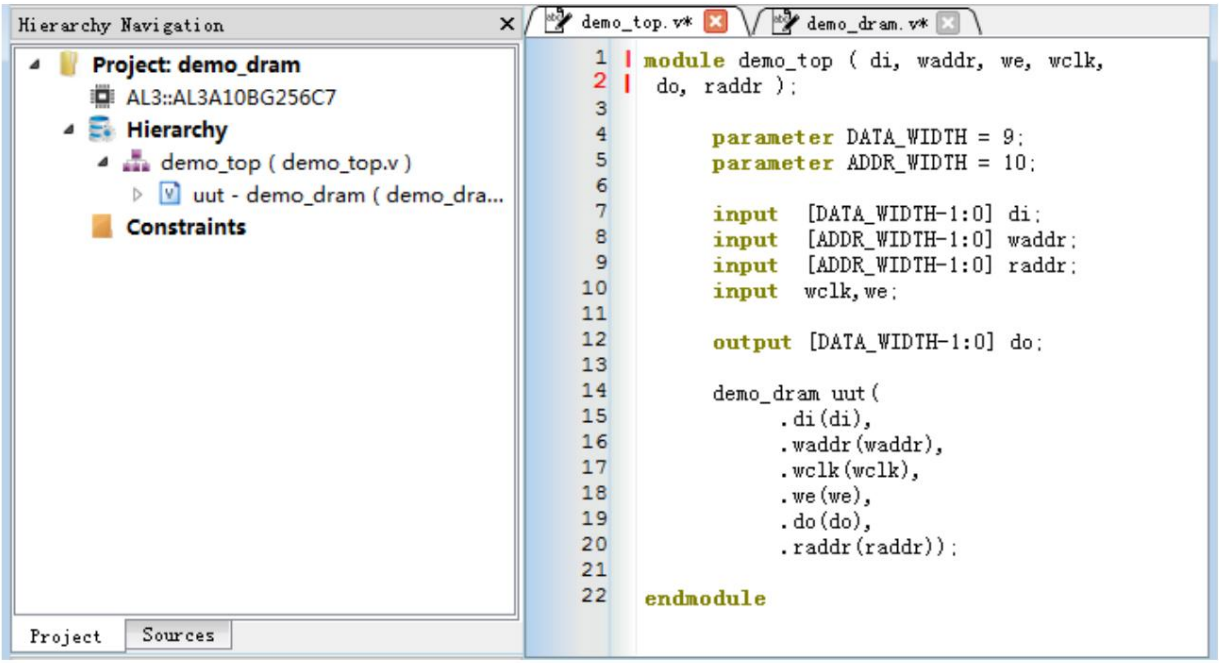
On the basis of instantiation, the instantiation process is consistent.

1. Create a new project and add a top-level module to the project.

2. Add the demo_dram.v generated in the previous step to the project

3. Call the demo_dram module in the top-level module, and modify the inst name and port name, click the save button,

That is, the instantiation of the DRAM module is completed.



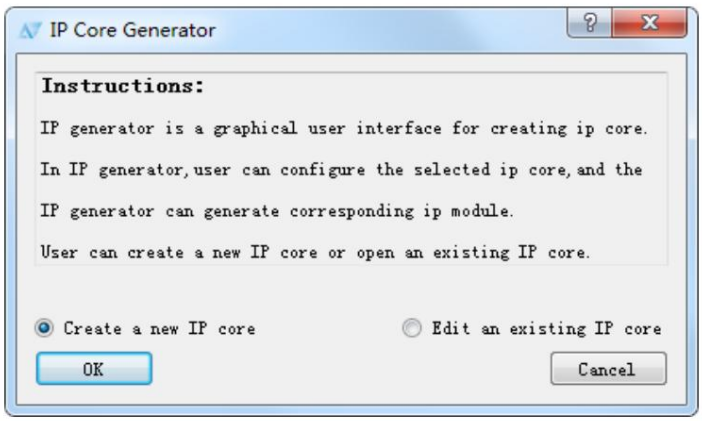
Machine Translated by Google

3.9 SDRAM module

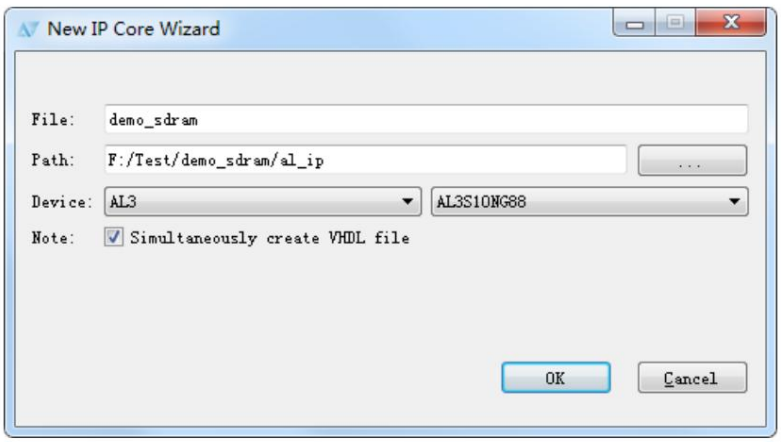
Some devices have an embedded SDRAM module, this section describes the creation, instantiation, and usage of the module related notes.

3.9.1 Create SDRAM module

1. Select **Tools ÿ IP** Generator, select "Create **a new IP core"**



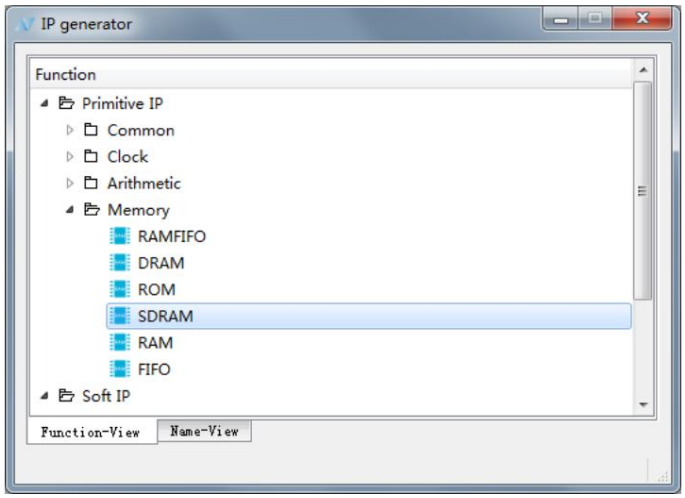
2. Enter the module name and select the storage path. Here, if the SDRAM module is created on the basis of a project Blocks, storage paths and device names will be the same as the project. If created without a project SDRAM module, the user needs to manually set the save path and device name. If **"Simultaneously create VHDL file",** TD will generate the corresponding VHDL file.

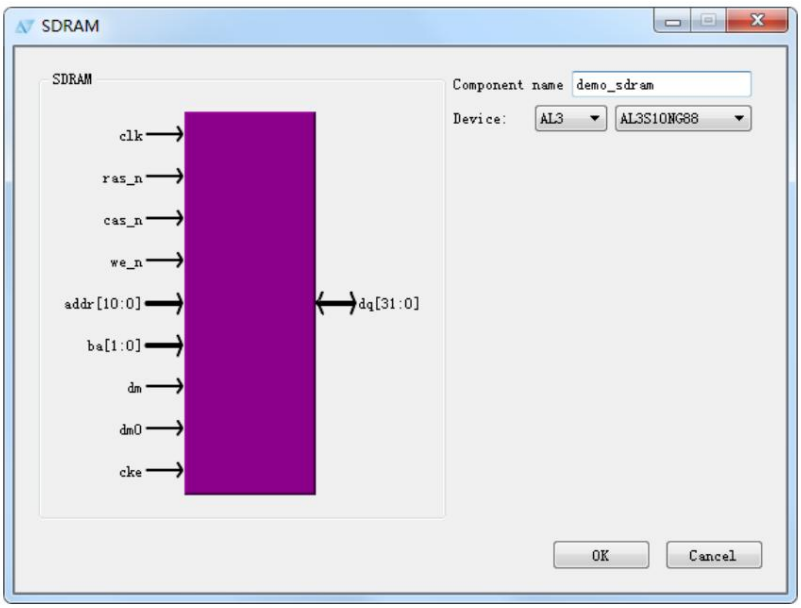


Machine Translated by Google

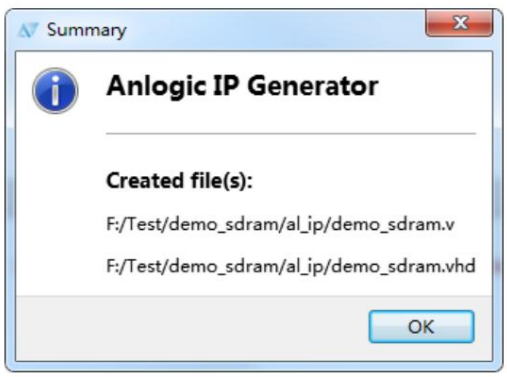
3. In the Function window **Primitive IP ,** expand **Memory ÿ SDRAM, and double-click SDRAM** to open the configuration.

interface





4. Click "OK" to complete the setting, and the generated file is as follows:



Machine Translated by Google

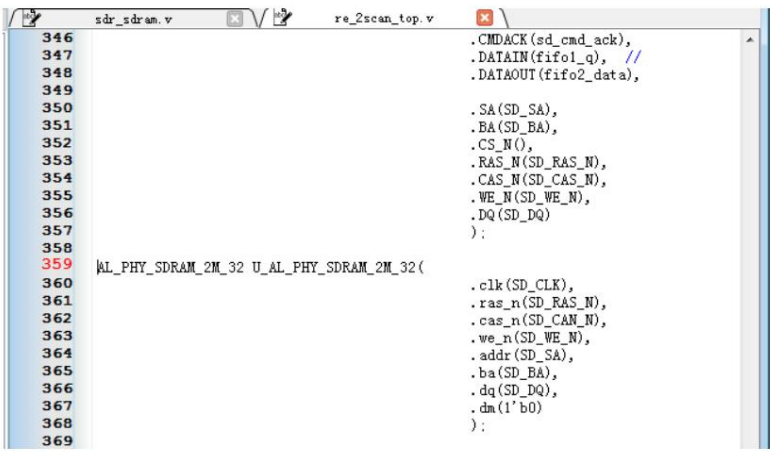
3.9.2 Instantiating the **SDRAM** Module

The following takes the AL3S10 device as an example to introduce the use of the SDRAM module. SDRAM and FPGA via software

Deep integration. So if you want to use SDRAM, you only need to instantiate the following IP block at the top level. of this IP

The prototype is as follows:

```
AL_PHY_SDRAM_2M_32 U_AL_PHY_SDRAM_2M_32(
        .clk(SD_CLK),                   // SDRAM clock                  1bit width
       .ras_n(SD_RAS_N),                // SDRAM row strobe 1bit width
        .cas_n(SD_CAS_N),               //SDRAM column strobe 1bit width
        .we_n(SD_WE_N),                 //SDRAM write enable 1bit width
        .addr(SD_SA),                   //SDRAM address                11bits bit width
        .ba(SD_BA),                     // SDRAM BANK address 2bits width
        .dq(SD_DQ),                     // SDRAM data                   32bits bit width
        .dm1(1'b0)                      // SDRAM data mask 1bit width
);
```



Machine Translated by Google

The pin assignments of the SDRAM are as follows:

SDRAM pin name	SDRAM Pin Description	pin connection
DQ0 ~ DQ31	Data Pin 0 ~ Data Pin 31 Address	connected to IP
SA0 ~ SA10	Pin 0 ~ Address Pin 10	connected to IP
BA0	BANK address pin 0	connected to IP
BA1	BANK address pin 1	connected to IP
WE_N	write enable row strobe	connected to IP
RAS_N	column strobe chip	connected to IP
CAS_N	clock chip select data	connected to IP
CLK	0-7 mask data 8-15	connected to IP
CS_N	mask data 16-23 mask	Internally pulled low
DM0	data 24-31 mask clock	Internally pulled low
DM1	enable	connected to IP
DM2		Internally pulled low
DM3		Internally pulled low
CKE		Internal pull up

When using SDRAM, please note the following:

1. DQ0~DQ31 are bidirectional data pins;

2. There is a 90-degree phase difference between the SDRAM operating clock CLK and the FPGA internal clock;

3. When assigning a value of 0 to DM1, 32-bit data is available; assigning a value of 1 to DM1, masking 8-15 bits of data can reduce power consumption.

consumption.

Machine Translated by Google

3.10 ADC module

The Eagle series has an 8-channel 12-bit 1MSPS ADC embedded in the chip's BANK8. ADC mode

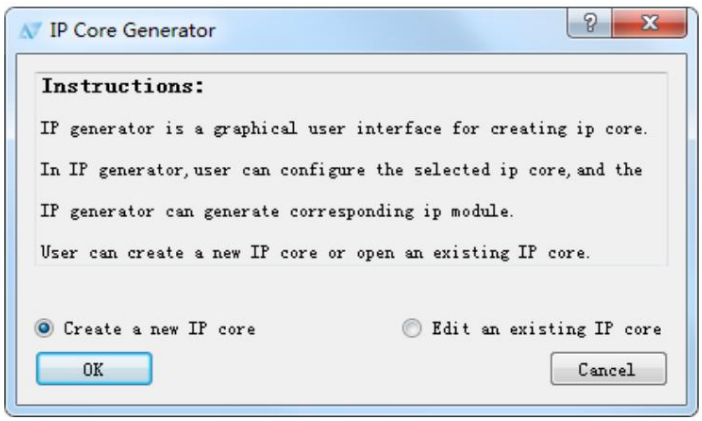
The block requires separate 3.3V analog operating voltage and analog ground as well as a separate VREF voltage input. 8 channels

The input and user IO are multiplexed, and can be used as ordinary IO when the user does not need to use the ADC module. When using ADC

, the VCCIO voltage of BANK8 should not be lower than the ADC analog supply voltage.

3.10.1 Create ADC module

1. Select **Tools ÿ IP** Generator, select "Create **a new IP core"**

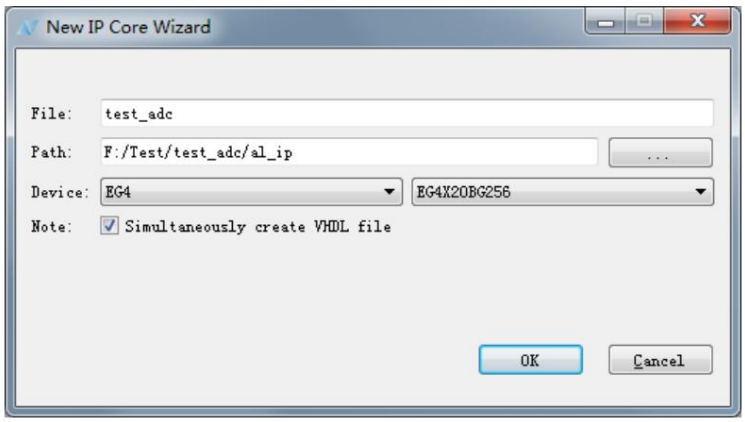


2. Enter the module name and select the storage path. Here, if the ADC module is created on the basis of a project, save the

   The storage path and device name will be consistent with the project. If you create an ADC module without a project, use

   Users need to manually set the save path and device name. If **"Simultaneously create VHDL file" is** checked ,

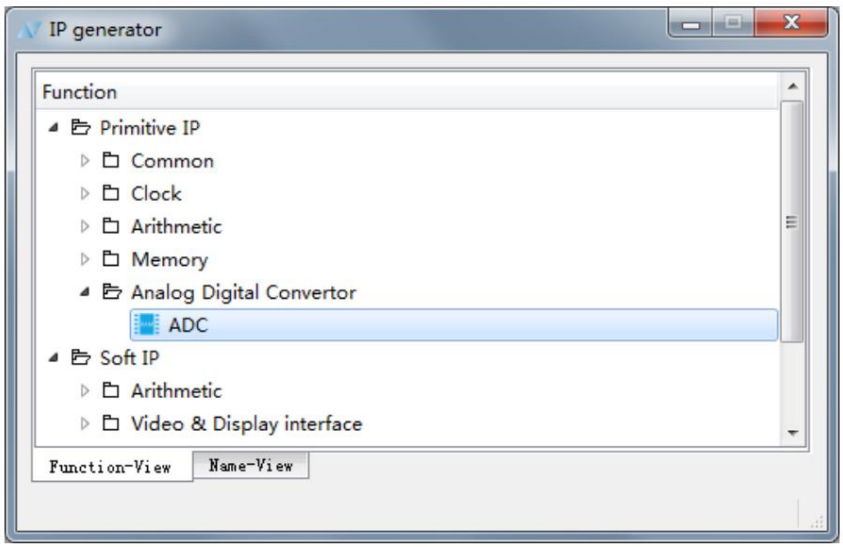
   TD will generate the corresponding VHDL file.



Machine Translated by Google

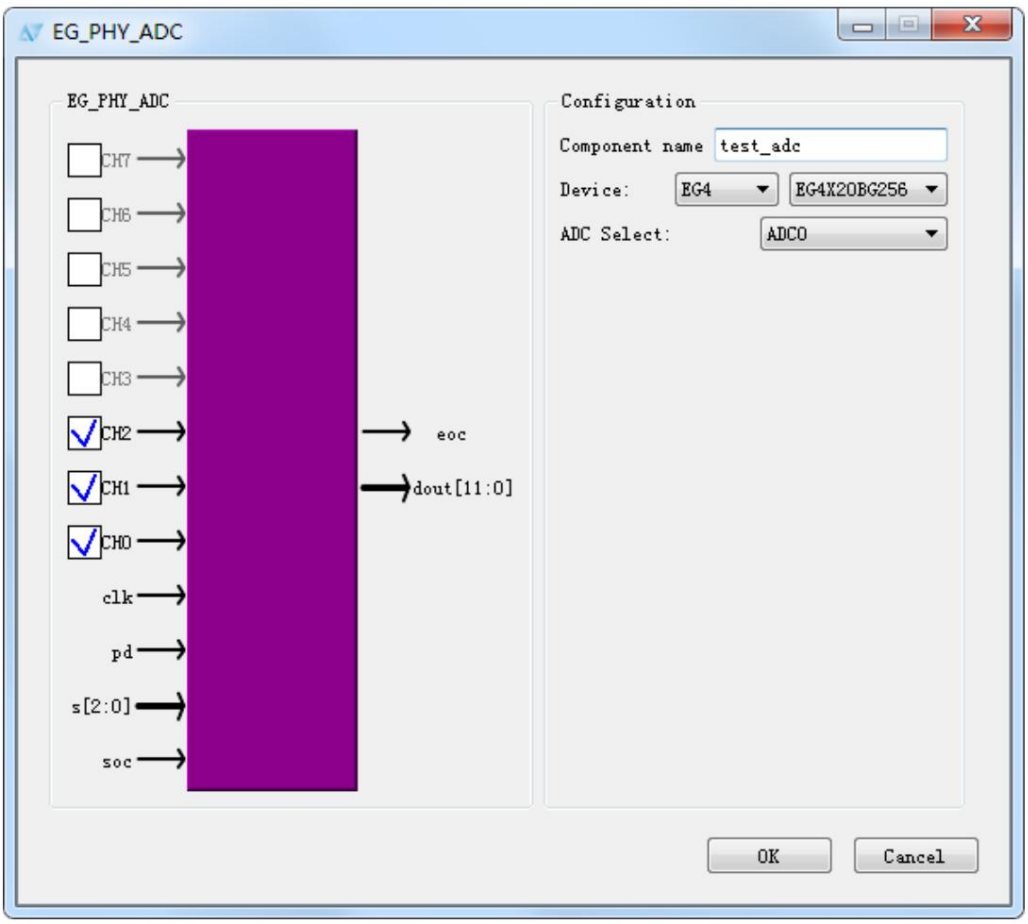
3. In the Function window **Primitive IP ,** expand **Analog Digital Convertor ÿ** ADC, and double-click **ADC**

Open the configuration interface.



4. **Fill in "Component** Name" and select the channel of ADC (for EF2 series, ADC0 and ADC1 are optional).

The channels that can currently be used depend on the package type of the current device.



Machine Translated by Google

The ADC module external/internal ports are described as follows:

Chip Port Name Port Type		illustrate
ADC_VDDD Power PAD		3.3V digital power input
ADC_VDDA External power supply PAD		3.3V analog power input
ADC_VSSA External power supply PAD		3.3V analog ground
ADC_VREF External PAD		Independent input, sampling reference analog potential input, input voltage range 2.0V~3.3V, not more than VDDA
ADC_CH<7:0> External PAD		8-channel sampling signal input, multiplexed with user IO
Internal Port Name Port Direction Input		illustrate
clk	Input Input	ADC clock
pd	(from FPGA)	ADC low-power power-down mode
s<2:0>	Input (from FPGA) Output (to	ADC channel selection signal input
soc	FPGA) dout<11:0> Output (to	ADC sampling enable signal input, active high
eoc	FPGA)	ADC conversion completed output, active high
		ADC conversion result of corresponding channel

*In ADC, ADC_CH<7:0> and ADC_VREF do not support hot swap. When there is a need for hot swapping, it is recommended to avoid ADC Multiplexed pins, please refer to the datasheet of the corresponding device for the specific ADC module performance and precautions.

5. Click "OK" to complete the setting, the generated file is as follows:

The existing test_adc.ipc can also be opened and edited by “Edit **an existing IP** core”.

Machine Translated by Google

3.10.2 Instantiate the **ADC** module

This manual takes a new project as an example to introduce the process of instantiating the ADC module. Users can also build on existing projects

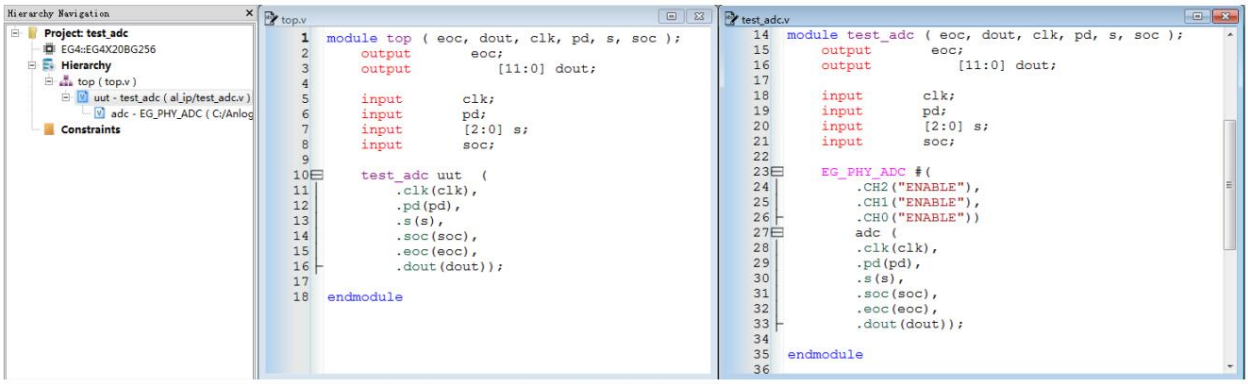
The instantiation is performed on the above, and the instantiation process is the same.

1. Create a new project and add a top-level module to the project.

2. Add the test_adc.v generated in the previous step to the project

3. Call the test_adc module in the top-level module, and modify the inst name and port name, click the save button,

That completes the instantiation of the ADC module.



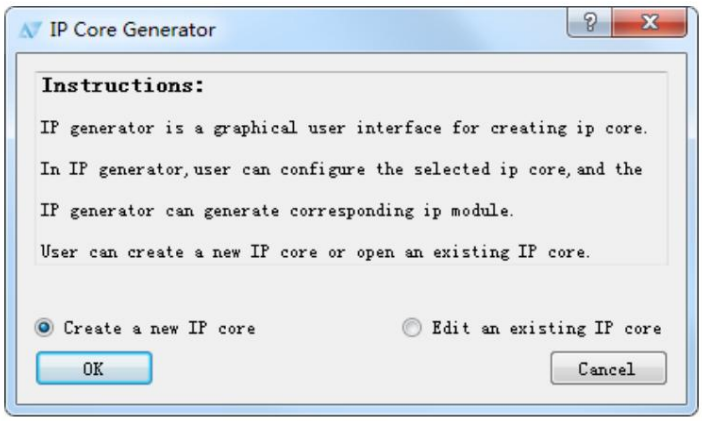
Machine Translated by Google

3.11 LVDS7_1 module

IP Generator supports the use of the LVDS7_1 module, which is the LVDS LCD driver interface module, which supports VESA Protocol standards and JEIDA protocol standards.

3.11.1 Create LVDS7_1 module

1. Select **Tools ÿ IP** Generator, select "Create **a new IP core"**

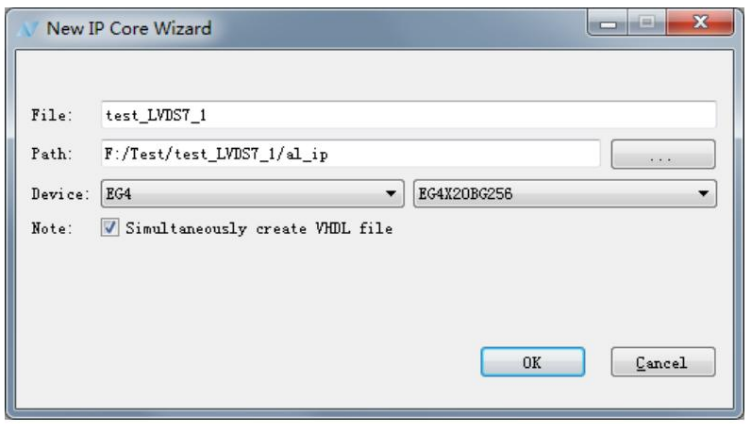


2. Enter the module name and select the storage path. Here, if the LVDS7_1 module is created on the basis of a project,

   The storage path and device name will be consistent with the project. If you create an LVDS7_1 module without a project

   block, the user needs to manually set the save path and device name. If **"Simultaneously create VHDL" is** checked

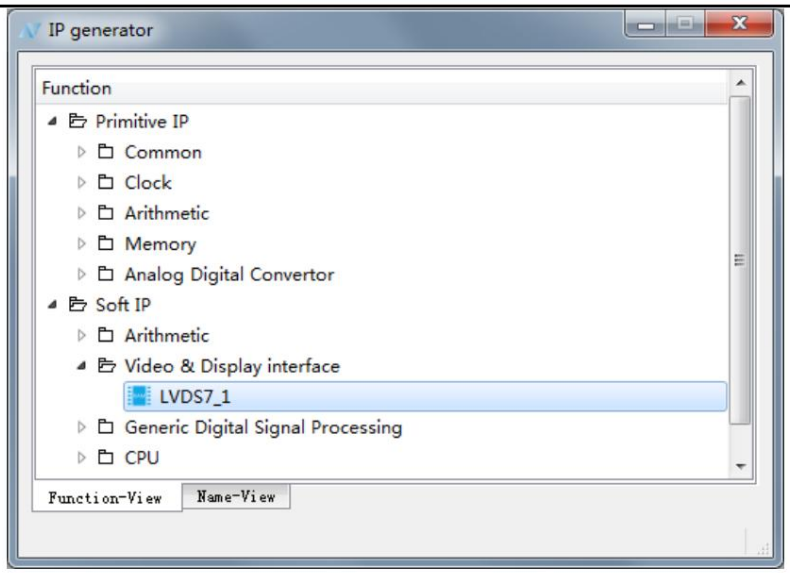
   file", TD will generate the corresponding VHDL file.



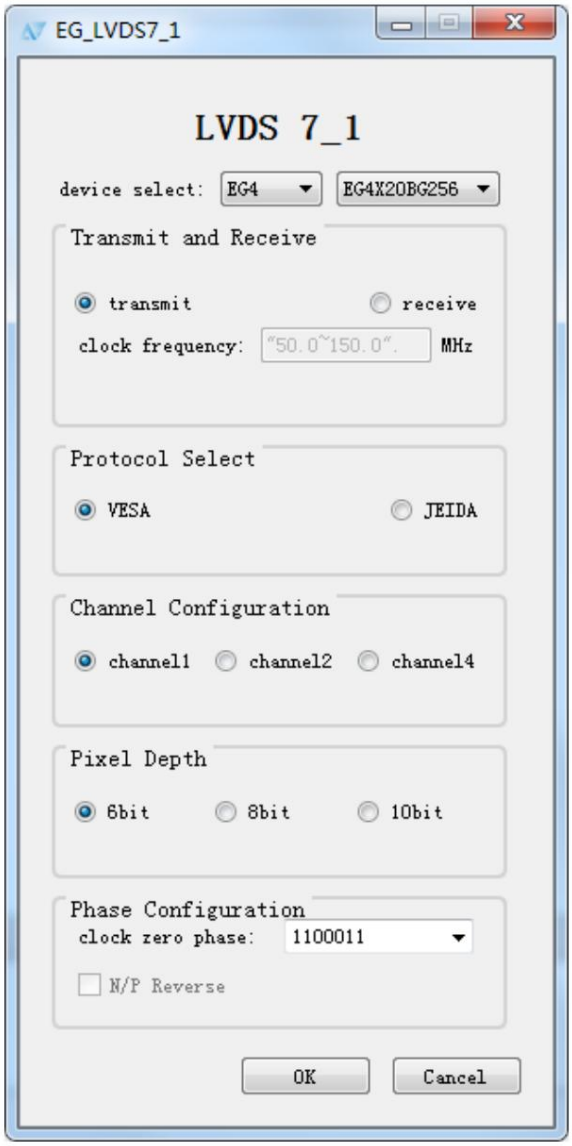
3. In the Function window **Soft IP ,** expand **Video & Display interface ÿ LVDS7_1, double-click**

   LVDS7_1 Open the configuration interface.

Machine Translated by Google



4. Set the parameters in sequence as required.



Machine Translated by Google

Clock Frequency:

Support input 50~150MHz, only optional when receive.

Transmit and Receive:

transmit: select the sending mode; receive: select the receiving mode.

Protocol Select:

LVDS7:1 has two standard protocols, VESA and JEIDA, choose one of the two protocols.

Channel Configuration:

1-4 channels are supported. channel1: use single-channel transmission; channel2: use dual-channel transmission;

channel4: use four-channel transmission.

Pixel Depth:

3 pixel depths are supported. 6bit: 6bit data mode; 8bit: 8bit data mode; 10bit: 10bit data

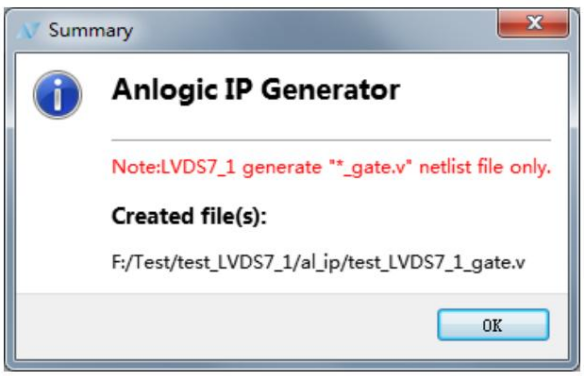
data mode;

Phase Configuration:

The default clock zero phase is 7`b1100011, and the phase can be entered manually.

N/P Reverse: When checked, the LVDS N/P pins are reversed, which is only optional for receive.

5. Click "OK" to complete the setting, and the generated netlist file is as follows:



LVDS7_1 only supports Verilog, not VHDL for now. Similarly, through the "Edit **an existing IP core" method**

to open and edit the existing test_LVDS7_1.ipc.

Machine Translated by Google

3.11.2 Instantiate the **LVDS7_1** module

This manual takes a new project as an example to introduce the process of instantiating the LVDS7_1 module.

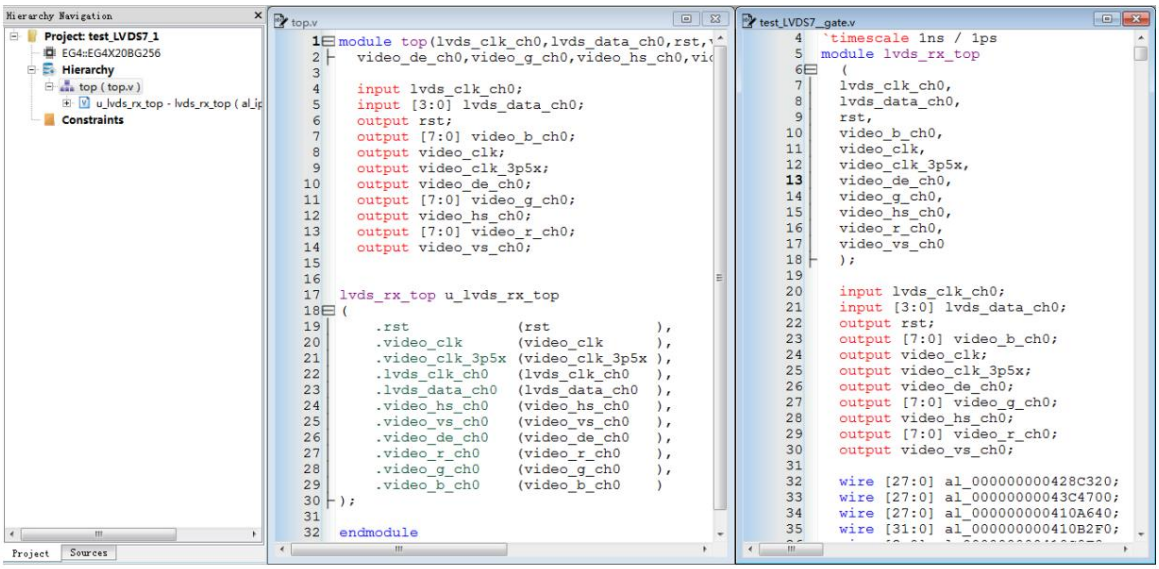
On the basis of instantiation, the instantiation process is consistent.

1. Create a new project and add a top-level module to the project.

2. Add the test_LVDS7_1_gate.v generated in the previous step to the project

3. Call the test_LVDS7_1 module in the top-level module, modify the inst name and port name, and click Save

   button to complete the instantiation of the LVDS7_1 module.



Machine Translated by Google

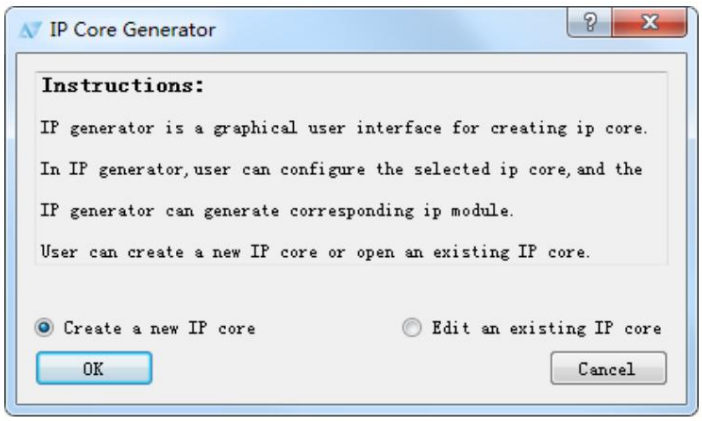
3.12 CPU module

IP Generator supports the use of CPU modules, that is, a 32-bit soft-core CPU embedded in the FPGA,

Flexible control functions can be realized.

3.12.1 Create CPU module

1. Select **Tools ÿ IP** Generator, select "Create **a new IP core"**

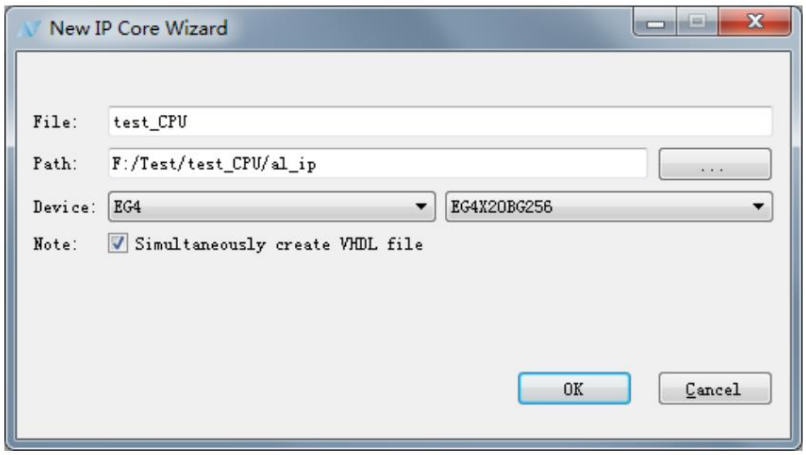


2. Enter the module name and select the storage path. Here, if the CPU module is created on the basis of the existing project, save the

The storage path and device name will be consistent with the project. To create a CPU module without a project, use

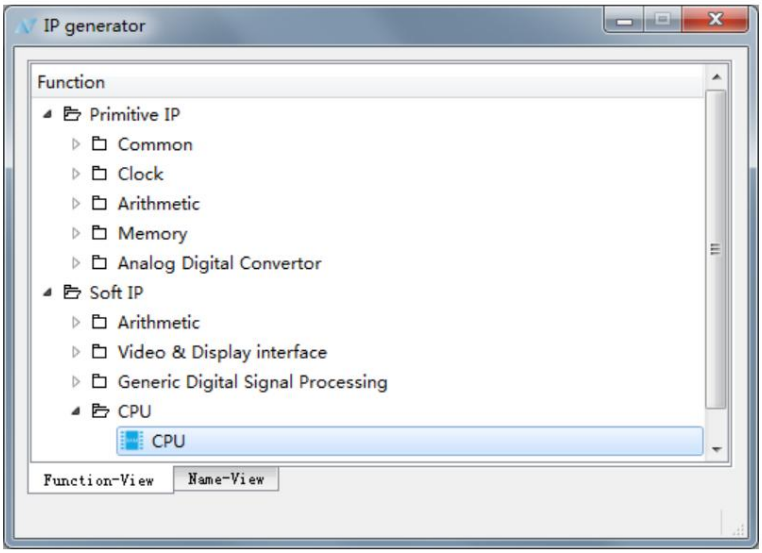
Users need to manually set the save path and device name. If **"Simultaneously create VHDL file" is** checked ,

TD will generate the corresponding VHDL file.

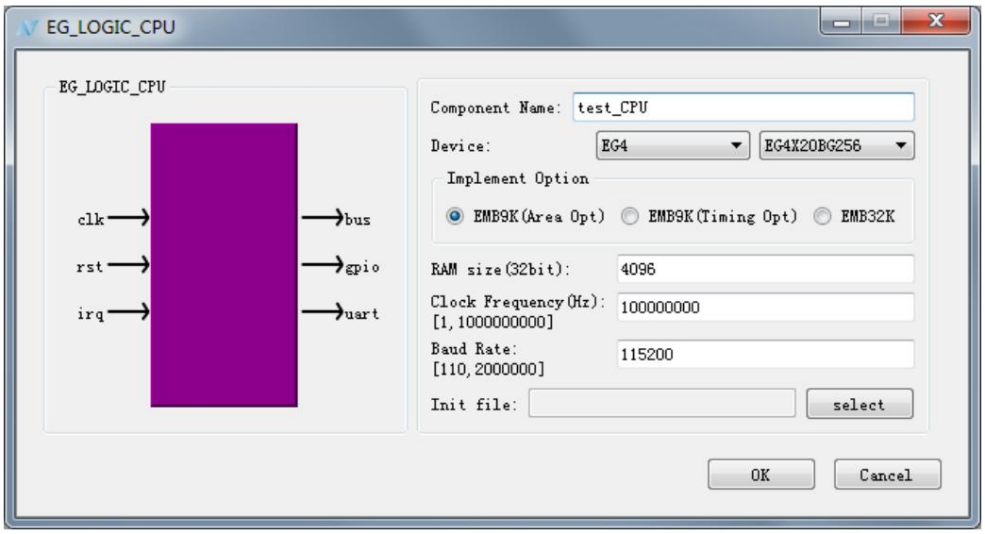


Machine Translated by Google

3. In the Function window **Soft IP ,** expand **CPU ÿ** CPU, and double-click the **CPU** to open the configuration interface.



4. Set the parameters in sequence as required.



Implement Option: RAM type selection, the optional range is consistent with the selected device.

RAM size (32bit): RAM size, ranging from 1 to the upper limit of the BRAM of the FPGA device.

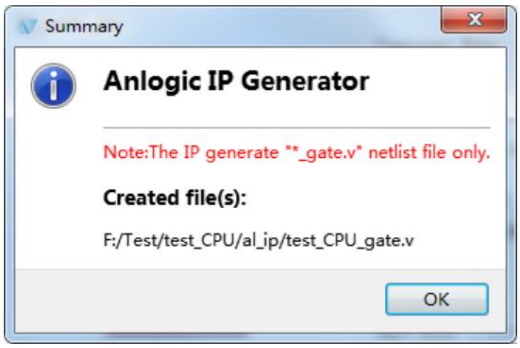
Clock Frequency(Hz): The clock frequency, the range is 1 ~ 1000000000.

Baud Rate: Serial port baud rate, the range is 110 ~ 2000000.

Init file: To add an initialization file, you need to select the corresponding mif/dat file.

Machine Translated by Google

5. Click "OK" to complete the setting, and the generated netlist file is as follows.



The CPU only supports Verilog, not VHDL for now. You can also use the "Edit **an existing IP core" method**

to open and edit the existing test_CPU_1.ipc.

3.12.2 Instantiate the **CPU** module

This manual takes a new project as an example to introduce the process of instantiating the CPU module.

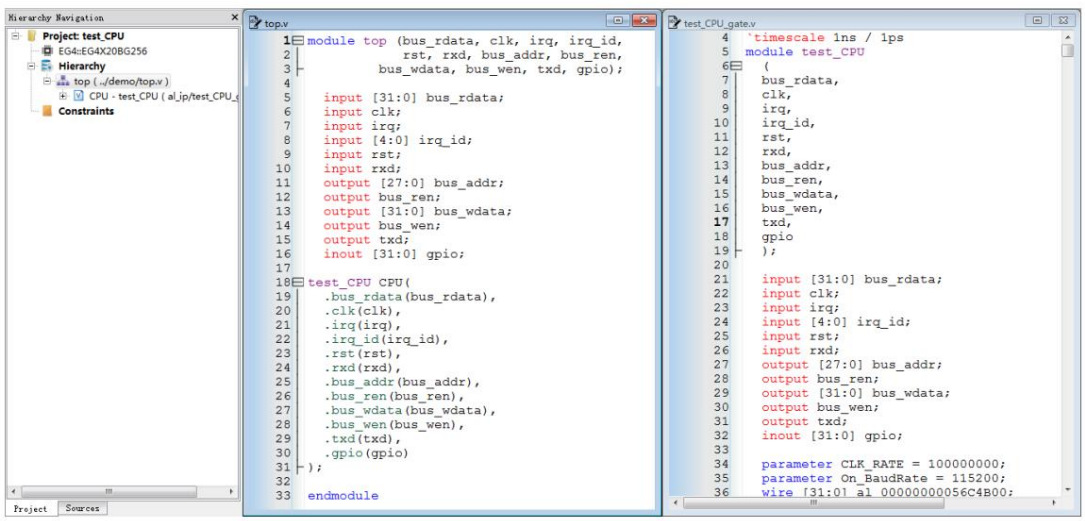
The instantiation is performed on the above, and the instantiation process is the same.

1. Create a new project and add a top-level module to the project.

2. Add the test_CPU_gate.v generated in the previous step to the project

3. Call the test_CPU module in the top-level module, and modify the inst name and port name, click the save button,

This completes the instantiation of the CPU module.



上海安路信息科技股份有限公司
SHANGHAI ANLOGIC INFOTECH CO., LTD.

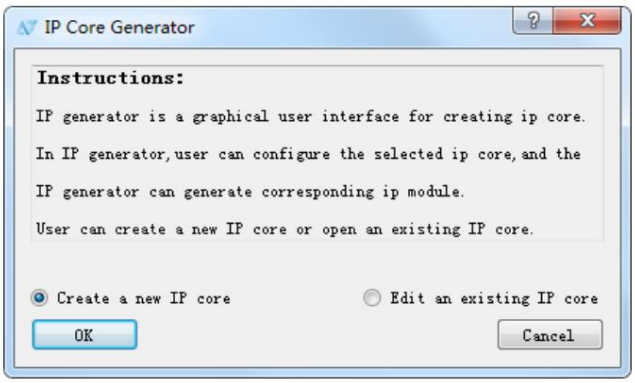
Machine Translated by Google

3.13 CRC module

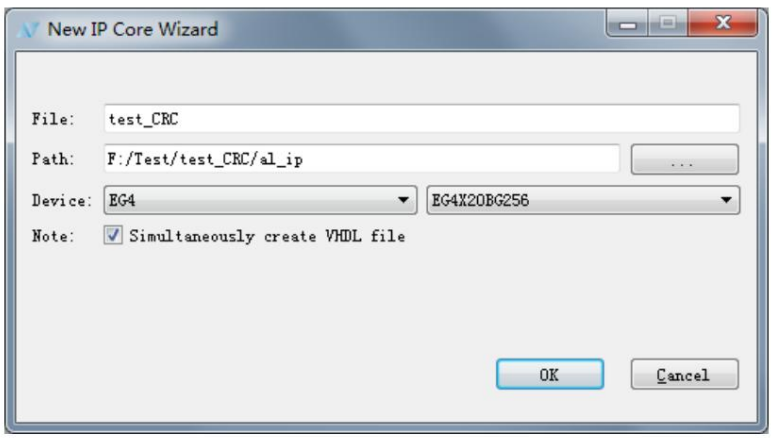
IP Generator supports the use of CRC modules. CRC uses the principle of division and remainder for error detection (Error Detecting). The basic idea of the CRC check is to append a binary sequence number to the frame to be sent (ie. check code), generate a new frame and send it to the receiver. At the receiving end, by judging whether the frame of the receiving end can be generated The same check code is used to verify the correctness of data transmission.

3.13.1 Create CRC module

1. Select **Tools ÿ IP** Generator, select "Create **a new IP core"**



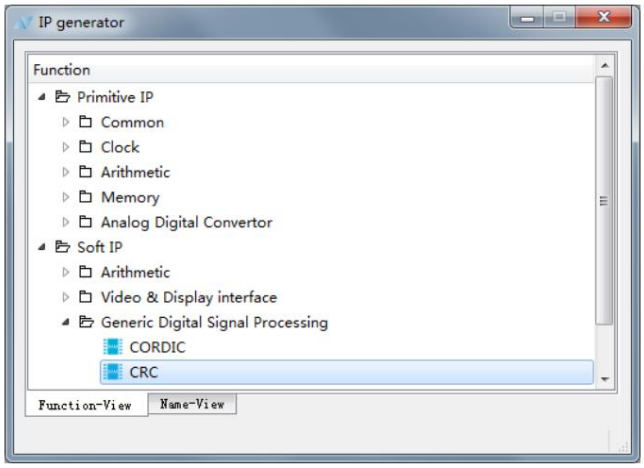
2. Enter the module name and select the storage path. Here, if the CRC module is created on the basis of a project, save the The storage path and device name will be consistent with the project. To create a CRC module without a project, use Users need to manually set the save path and device name. If **"Simultaneously create VHDL file" is** checked , TD will generate the corresponding VHDL file.



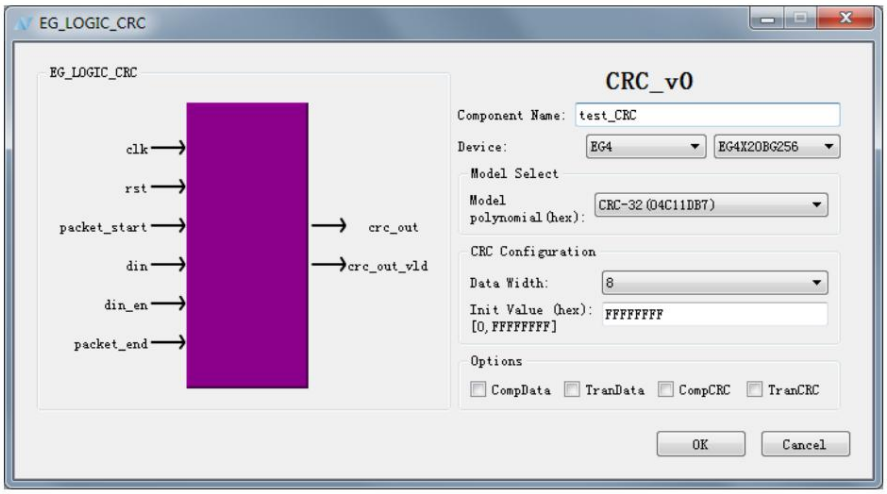
Machine Translated by Google

3. In the Function window **Soft IP ,** expand **Generic Digital Signal Processing ÿ** CRC, double-click

CRC opens the configuration interface.



4. Set the parameters in sequence as required.



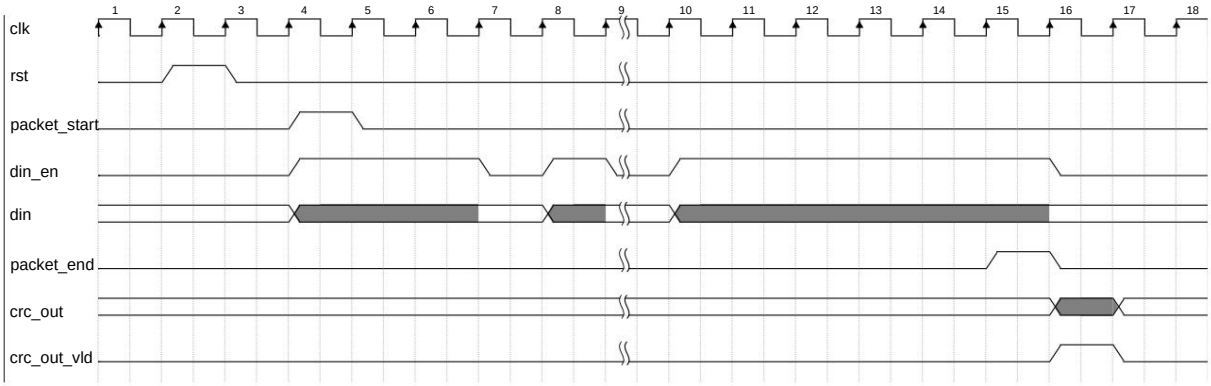
CRC IP Port Signal Description

interface		describe
clk		CRC IP clock signal
rst		reset signal
packet_start		Input data frame start signal
packet_end	direction input input input input	Input data frame end signal
din	input	input data (data_in)
din_en	input	Input data enable (data_enable)
crc_out	output	CRC output check code data
crc_out_vld	Output	CRC output data valid

description: The input operand is valid when the din_en signal is high, and the output check code result is in the crc_out_vld signal. Active when high.

Machine Translated by Google

CRC Timing Diagram



packet_start indicates the start signal of the data frame. The input data din is valid when the enable signal din_en is set high.

The final check result crc_out is output one cycle after the frame end signal packet_end, when crc_out_vld is set

Valid when high. The IP supports discontinuous patterns of data in incoming frames.

parameter configuration

Model Select: CRC mode selection

Model(polynomial): Select the CRC mode, the default is CRC-32 (4C11DB7) mode, in brackets

The polynomial in hexadecimal form for this mode.

CRC modes include CRC-5(05); CRC-8(07); CRC-10(233); CRC-16(8005); CRC-16(1021);

CRC-32 (4C11DB7) six modes.

CRC Configuration: CRC configuration

Data Width: Set the bit width of the input data. The data bit width is 8bits by default, as shown in the following table

shown:

Machine Translated by Google

CRC mode and supported data bit width table

mode (polynomial **hex)**	data bit width			
	1-bit	**8-bits**	**16-bits**	**32-bits**
CRC-5(05)	ÿ	ÿ		
CRC-8(07)	ÿ	ÿ		
CRC-10(233)	ÿ	ÿ		
CRC-16(8005)ÿ		ÿ	ÿ	
CRC-16(1021) ÿ		ÿ	ÿ	
CRC-32(04C11DB7) ÿ		ÿ	ÿ	ÿ

Initial Value: Set the initial value of the operation, in hexadecimal form.

Options: CRC options

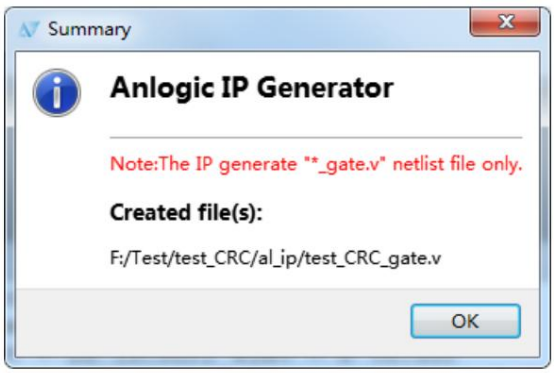
CompData: Select the complement of the input data.

TranData: Select the transpose of the input data.

CompCRC: Select the complement of the CRC data.

TranCRC: Select the transpose of the CRC data.

5. Click "OK" to complete the setting, and the generated netlist file is as follows:



CRC only supports Verilog, not VHDL for now. You can also use the "Edit **an existing IP** core" method to

Open and edit the existing test_CRC.ipc.

Machine Translated by Google

3.13.2 Instantiating the **CRC** module

This manual takes a new project as an example to introduce the process of instantiating the CRC module.

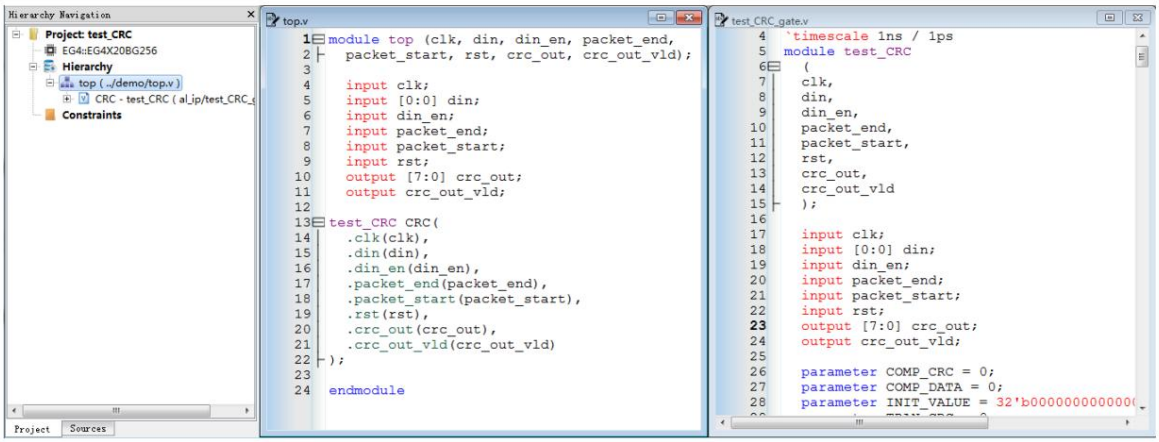
The instantiation is performed on the above, and the instantiation process is the same.

1. Create a new project and add a top-level module to the project.

2. Add the test_CRC_gate.v generated in the previous step to the project

3. Call the test_CRC module in the top-level module, and modify the inst name and port name, click the save button,

That completes the instantiation of the CRC module.



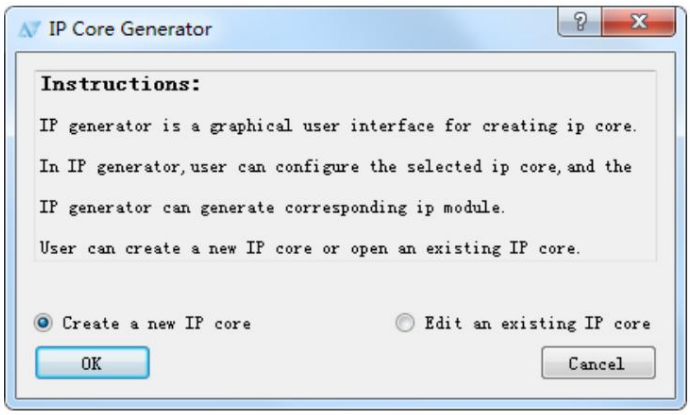
Machine Translated by Google

3.14 CORDIC module

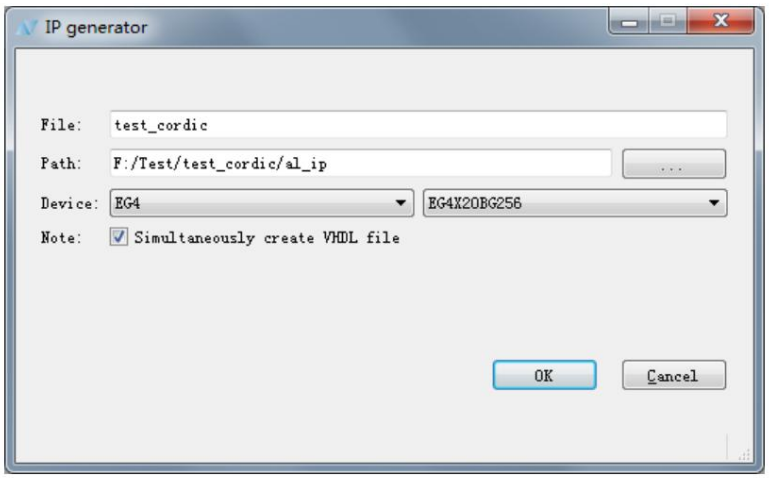
IP Generator supports the use of CORDIC blocks. CORDIC: Coordinate rotation digital calculation, is a mathematical calculation calculation method. Through coordinate rotation, complex trigonometric and hyperbolic functions can be decomposed into addition and subtraction. According to CORDIC The internal parameter configuration of CORDIC has a variety of application methods.

3.14.1 Create CORDIC module

1. Select **Tools ÿ IP** Generator, select "Create **a new IP core"**



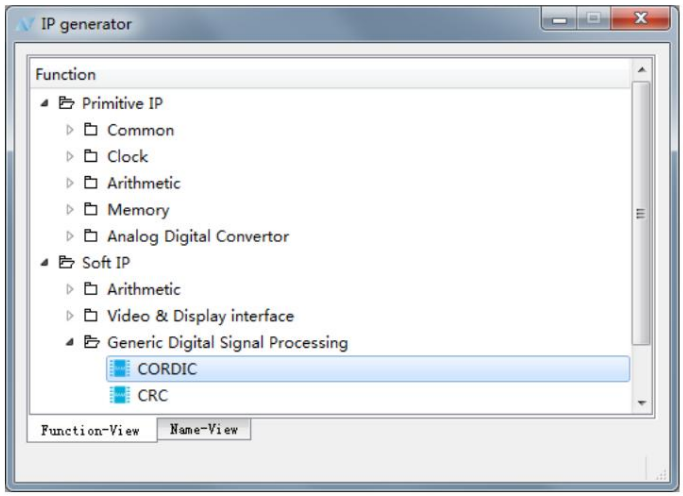
2. Enter the module name and select the storage path. Here, if the CORDIC module is created on the basis of a project, The storage path and device name will be consistent with the project. If you create a CORDIC model without a project block, the user needs to manually set the save path and device name. If **"Simultaneously create VHDL" is** checked **file",** TD will generate the corresponding VHDL file.



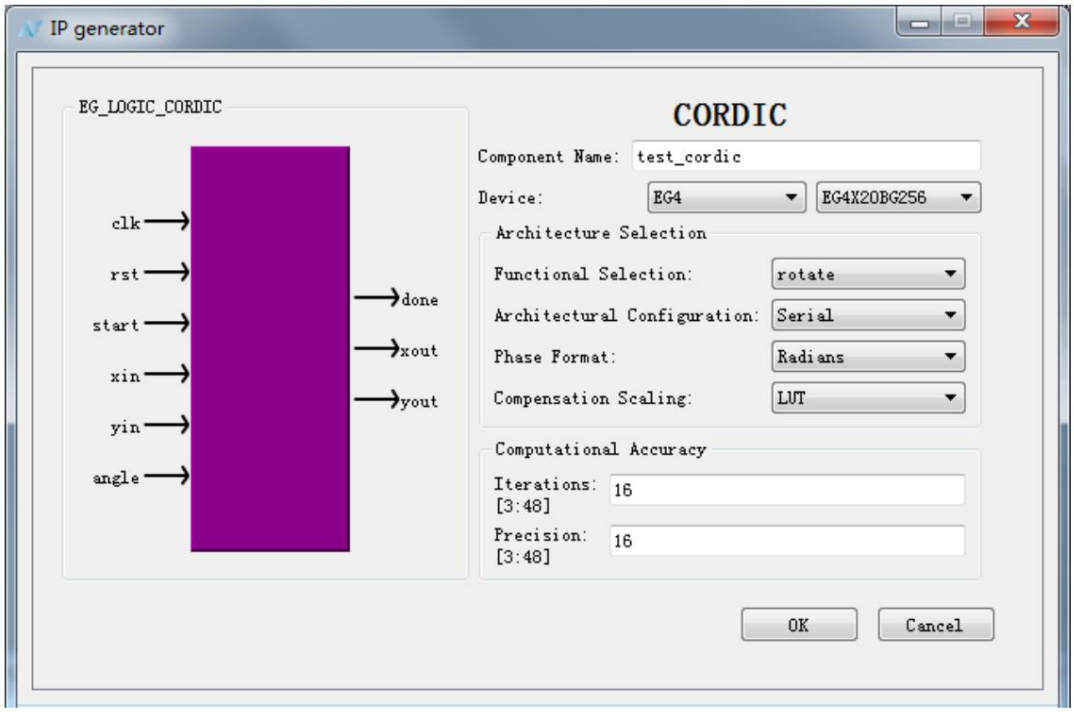
Machine Translated by Google

3. In the Function window **Soft IP ,** expand **Generic Digital Signal Processing ÿ CORDIC, double**

Click **CORDIC** to open the configuration interface.



4. Set the parameters in sequence as required.



parameter configuration

Architecture Selection:

Functional Selection:

Machine Translated by Google

ÿ rotate: coordinate rotation mode, used to calculate coordinate rotation

ÿ Orthogonal2polar: Orthogonal coordinate to polar coordinate, used for coordinate space conversion, complex number modulo and angle

ÿ polar2orthogonal: polar coordinates to orthogonal coordinates for coordinate space conversion

ÿ cos-sin: trigonometric function calculation, used to calculate sin and cos values

ÿ atan: inverse triangular tangent function calculation

ÿ cosh-sinh: Hyperbolic sine, hyperbolic cosine function calculation

ÿ atanh: Inverse hyperbolic tangent calculation

ÿ sqrt: used to calculate the root of a number

Architectural Configuration:

ÿ Serial: Serial architecture

ÿ Pipeline: pipeline architecture

Phase Format:

ÿ Radians: radian angle

ÿ Scaled Radians: Angle normalization

Compensation Scaling:

ÿ LUT: Display lookup table

ÿ DSP: Digital Signal Processor

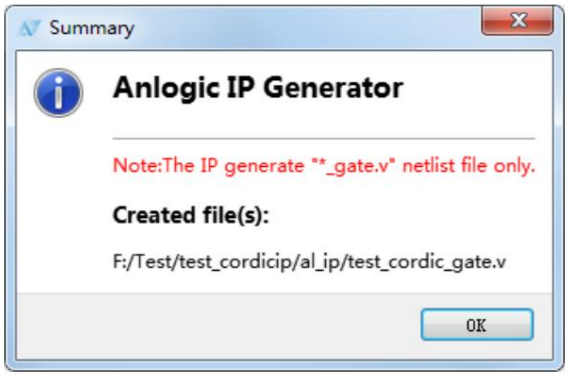
Computational Accuracy:

Itrations: The number of iterations

Precision: decimal precision

Machine Translated by Google

5. Click "OK" to complete the setting, and the generated netlist file is as follows:



CORDIC only supports Verilog, not VHDL for now. Similarly, through the "Edit **an existing IP core" method**

to open and edit an existing test_cordic.ipc.

3.14.2 Instantiating the **CORDIC** module

This manual takes a new project as an example to introduce the process of instantiating the CORDIC module.

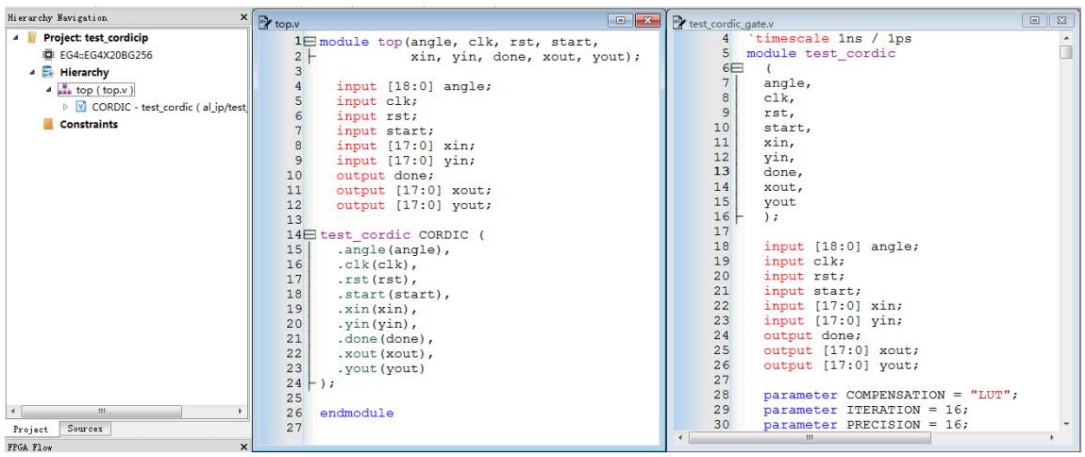
On the basis of instantiation, the instantiation process is consistent.

1. Create a new project and add a top-level module to the project.

2. Add the test_cordic_gate.v generated in the previous step to the project

3. Call the test_cordic module in the top-level module, and modify the inst name and port name, click the save button,

That is, the instantiation of the CORDIC module is completed.



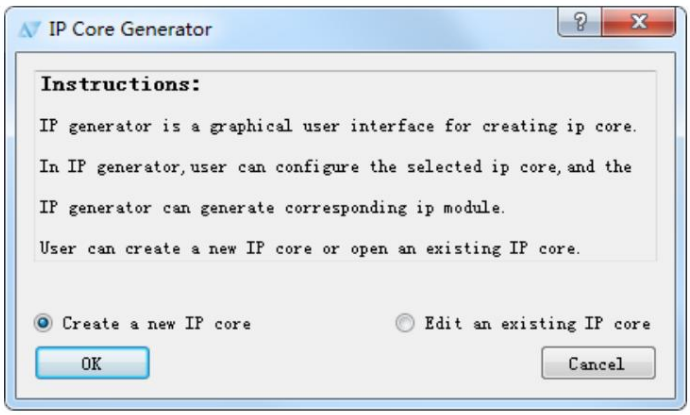
Machine Translated by Google

3.15 TEMAC module

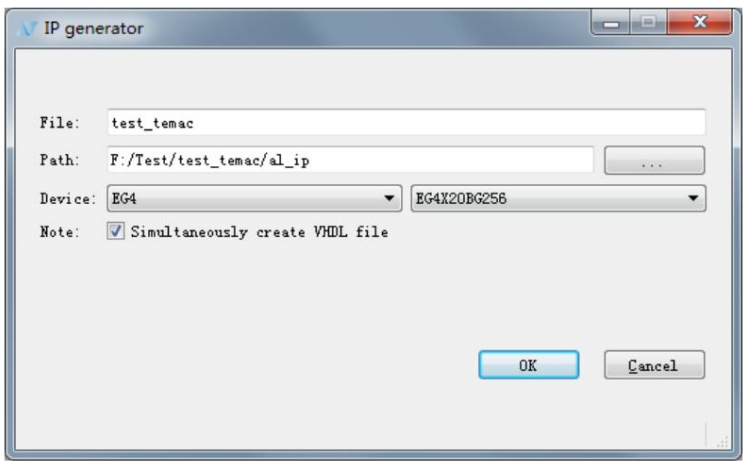
IP Generator supports the use of TEMAC blocks. TEMAC is the abbreviation of Tri-speed Ethernet MAC, which supports 10/100/1000Mbps speeds. According to the internal parameter configuration of the MAC core, TEMAC has a variety of application modes.

3.15.1 Create TEMAC module

1. Select **Tools ÿ IP** Generator, select "Create **a new IP core"**

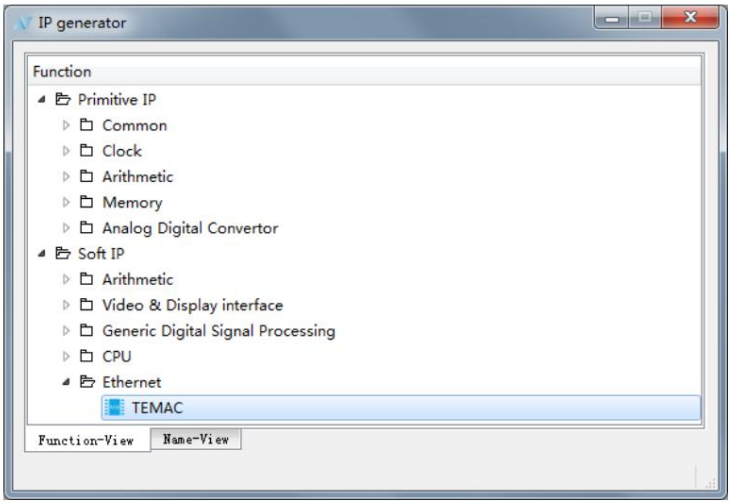


2. Enter the module name and select the storage path. Here, if the TEMAC module is created on the basis of a project, The storage path and device name will be consistent with the project. If you create a TEMAC model without a project block, the user needs to manually set the save path and device name. If **"Simultaneously create VHDL" is** checked **file",** TD will generate the corresponding VHDL file.

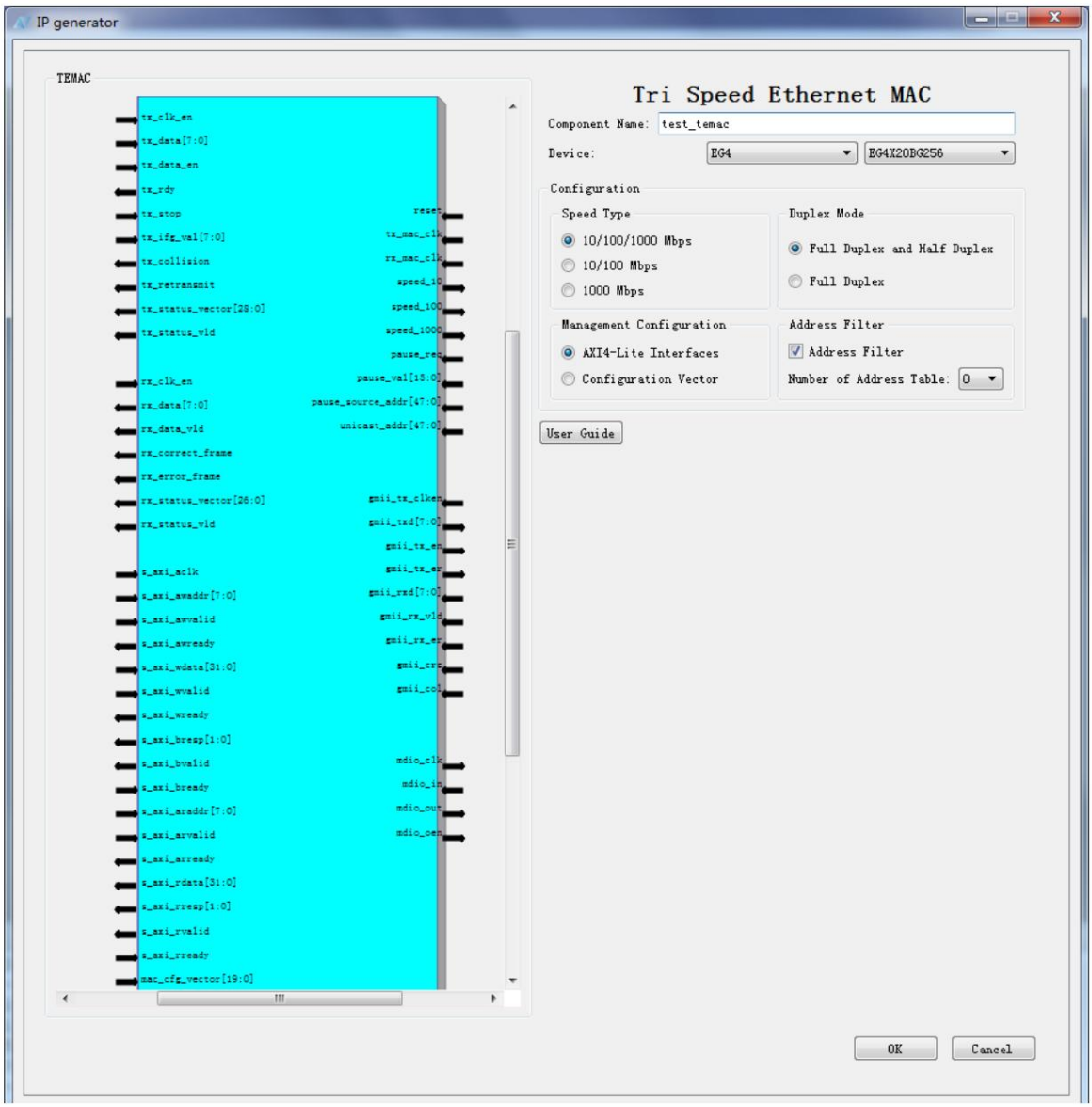


Machine Translated by Google

3. In the Function window **Soft IP ,** expand **Ethernet ÿ TEMAC, and double-click TEMAC** to open the configuration interface.



4. Set the parameters in sequence as required.



Machine Translated by Google

TEMAC IP Port Signal Description

Transmitter user side port

Signal	Direction description
tx_clk_en	input Transmitter clock enable signal. Combined with the tx_mac_clk signal it will be tri-speed (10/100/1000Mb/s) Ethernet support to generate corresponding speed clock
tx_data[7:0]	input This port provides the frame data to be sent
tx_data_en	input Control enable signal for sending frame data
tx_rdy	output handshake signal. Declared when the Mac core is ready to receive the current frame of data
tx_stop	input is declared by the user to force the MAC core to terminate the current frame transmission
tx_ifg_val[7:0]	input Configurable interframe gap adjustment value
tx_collision	output This signal is valid only in half-duplex mode, and the MAC core declares that the medium is pulsed. burst, any transmission in progress should be aborted
tx_retransmit	output This signal is valid only in half-duplex mode, when it is simultaneously with the tx_collision signal When it is high, it prompts the user that the terminated frame data should be re-supplied to the MAC core for retransmission
tx_status_vector[28:0]	output sends the status statistics of the previous frame
tx_status_vld	output is high at the end of frame transmission, indicating that tx_status_vector is valid

Receiver user side port

Signal	Direction description
rx_clk_en	input Receiver clock enable signal. Combined with the rx_mac_clk signal it will be a three-speed (10/100/1000Mb/s) Ethernet support to generate corresponding speed clock
rx_data[7:0]	output frame data received
rx_data_vld	output Control valid signal for receiving frame data
rx_correct_frame	output is declared at the end of frame reception, indicating that the frame was received correctly
rx_error_frame	output is declared at the end of frame reception, indicating that the frame was received in error
rx_status_vector[26:0]	output receives the status statistics of the previous frame
rx_status_vld	output is high at the end of frame reception, indicating that rx_status_vector is valid

Reset, Clock, Speed Indication Ports

Signal Direction Description		
reset	input Asynchronous reset of the entire MAC core, active high	
tx_mac_clk	input	Transmitter clock for the MAC core
rx_mac_clk	input	Receiver clock for the MAC core
speed_10	output indicates that the MAC core is running at 10 Mb/s	
speed_100	output indicates that the MAC core is running at 100 Mb/s	
speed_1000	output indicates that the MAC core is running at 1000 Mb/s	

Machine Translated by Google

Flow control, unicast address port

Signal Direction Description		
pause_req	input	sends a pause control frame request signal to insert a pause frame in the next frame
pause_val[15:0]	input	pause value, inserted into the parameter field of the send pause frame
pause_source_addr[47:0]	input	Pause frame MAC source address. This location can be written to through the AXI management configuration interface address register to overwrite it
unicast_addr[47:0]	input	Sets the default address of the MAC. Can be written to unicast ground through the management configuration interface address register to overwrite it

Manage configuration ports

Signal	Direction description	
s_axi_aclk	input	AXI4-Lite bus clock
s_axi_awaddr[7:0]	input	write address
s_axi_awvalid	input	write address is valid
s_axi_awready	output	write address ready
s_axi_wdata[31:0]	input	write data
s_axi_wvalid	input	write data is valid
s_axi_wready	output	write data ready
s_axi_bresp[1:0]	output	write response
s_axi_bvalid	output	write response valid
s_axi_bready	input	write response ready
s_axi_araddr[7:0]	input	read address
s_axi_arvalid	input	read address is valid
s_axi_arready	output	read address ready
s_axi_rdata[31:0]	output	read data
s_axi_rresp[1:0]	output	read response
s_axi_rvalid	output	read data is valid
s_axi_rready	input	read data ready

vector configuration port

Signal	Direction description	
mac_cfg_vector[19:0]	input	When the management interface AXI4-Lite is not used, the vector configuration bus executes the IP core base Configuration of this function

GMII/MII port

Signal	Direction description	
gmii_tx_clken	input	Clock enable signal for transmitter GMII / MII logic. Combine tx_mac_clk letter Number will generate corresponding speeds for tri-speed (10/100/1000 Mb/s) Ethernet support clock
gmii_txd[7:0]	The data sent by the output MAC.	When the IP core is running at 1 Gb/s, this bus 8 Bit data width valid; when IP core is running at 10 Mb/s or 100 Mb/s , the lower 4-bit data width of this bus is valid

Machine Translated by Google

gmii_tx_en	output	MAC enable control signal for sending data
gmii_tx_er	output	MAC error flag signal for sending data
gmii_rxd[7:0]	The input bit	Data received by the MAC. When the IP core is running at 1 Gb/s, this bus 8 data width is valid; when the IP core is running at 10 Mb/s or 100 Mb/s , the lower 4-bit data width of this bus is valid
gmii_rx_vld	input	A valid control signal for the MAC to receive data
gmii_rx_er	input	MAC receive data error flag signal
gmii_crs gmii_col	input	Carrier sense control signal input to the MAC
	input	Collision control signal input to MAC

MDIO port

Signal	Direction	description
mdio_clk	output	MDIO clock, generated using the AXI management interface configuration.
mdio_in	input	Input data signal used to communicate configuration and status to the PHY
mdio_out	output	Output data signal used to communicate configuration and status to the PHY.
mdio_oen	output	Tri-state control signal for MDIO signal. "0" means that the value mdio_out is declared to on the MDIO bus; "1" means that the value mdio_in is declared on the MDIO bus

parameter configuration

Speed Type:

ÿ 10/100/1000 Mbps: MAC core supports triple speed mode

ÿ 10/100 Mbps: MAC core supports 10/100 Mbps speed mode

ÿ 1000 Mbps: MAC core only supports 1000 Mbps speed mode

Duplex Mode:

ÿ Full Deuplex and Half Duplex: MAC core supports half duplex and full duplex mode

ÿ Full Deuplex: The MAC core only supports full duplex mode

Management Configuration:

ÿ AXI4-Lite Interfaces: The MAC core only supports AXI interface management configuration

ÿ Configuration Vector: MAC core only supports vector bus configuration

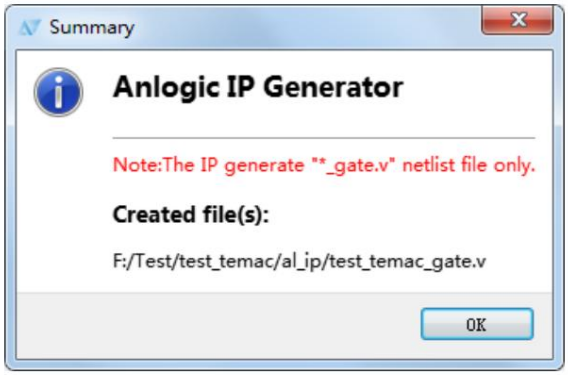
Address Filter: The MAC core supports address filtering

ÿ Number of Address Table: 0ÿnumberÿ16, the number of addresses in the address table supported by the MAC core

上海安路信息科技股份有限公司
SHANGHAI ANLOGIC INFOTECH CO., LTD.

Machine Translated by Google

5. Click "OK" to complete the setting, and the generated netlist file is as follows:



TEMAC only supports Verilog, not VHDL for now. Similarly, through the "Edit **an existing IP core" method**

to open and edit an existing test_temac.ipc.

3.15.2 Instantiating the **TEMAC** module

This manual takes a new project as an example to introduce the process of instantiating the TEMAC module.

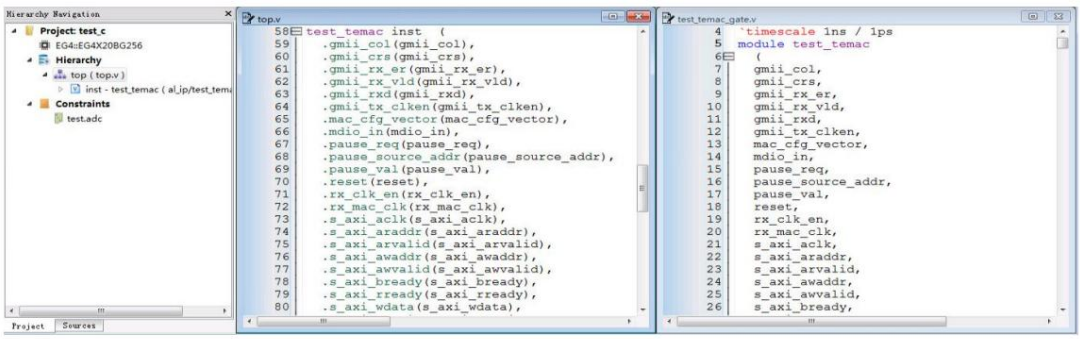
On the basis of instantiation, the instantiation process is consistent.

1. Create a new project and add a top-level module to the project.

2. Add the test_temac_gate.v generated in the previous step to the project

3. Call the test_temac module in the top-level module, and modify the inst name and port name, click the save button,

That completes the instantiation of the TEMAC module.



Machine Translated by Google

4 User constraints

User constraints include two parts: physical constraints and timing constraints. Physical constraints use Anlogic's custom ADC (Anlogic Design Constraint) format, which constrains the pin characteristics and internal unit characteristics of the FPGA chip.

Pin features include **Bank, Location, IOStandard, Drivestrength, PullType, SlewRate,**

More than 10 kinds of **DiffResistor, PCIClamp, PackReg, InDelay,** etc. The cell characteristic constraints support the user in the ADC

The unit Location is specified with set_inst_assignment in the file. Pin characteristic constraints can be set through the interface.

Please refer to Appendix 9.1 for a description of physical constraints. Industry-standard SDC (Synopsys Design Constraint) for timing constraints format, which constrains the external delay information and internal timing requirements of the FPGA chip, and is related to the delay in the design process

The optimization takes into account the constraints of SDC. Please refer to Appendix 9.2 for timing constraints description.

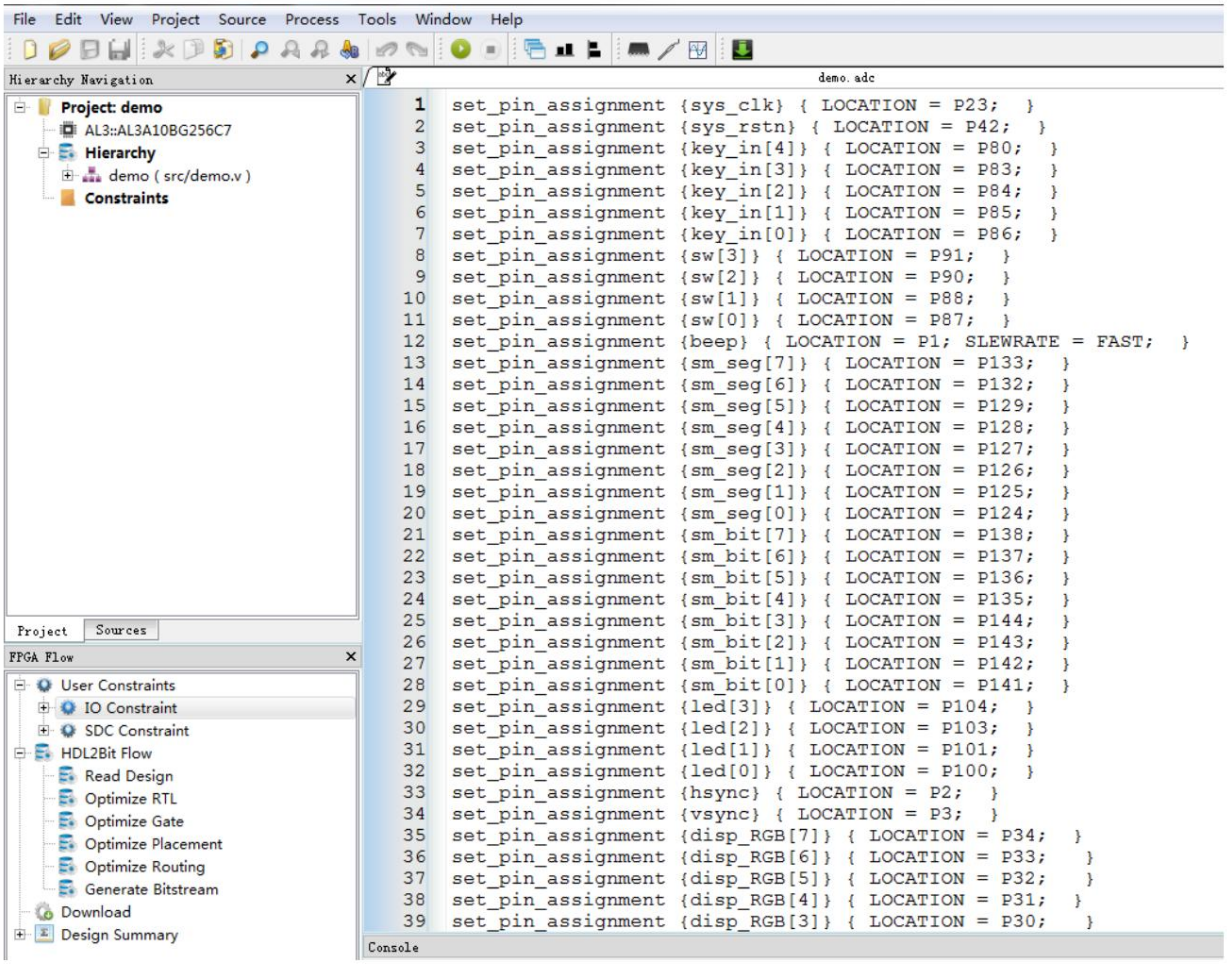
4.1 Physical constraints

4.1.1 Add IO constraints

New **ADC** file

1. Click the New button and enter the ADC command in the newly opened window

Machine Translated by Google

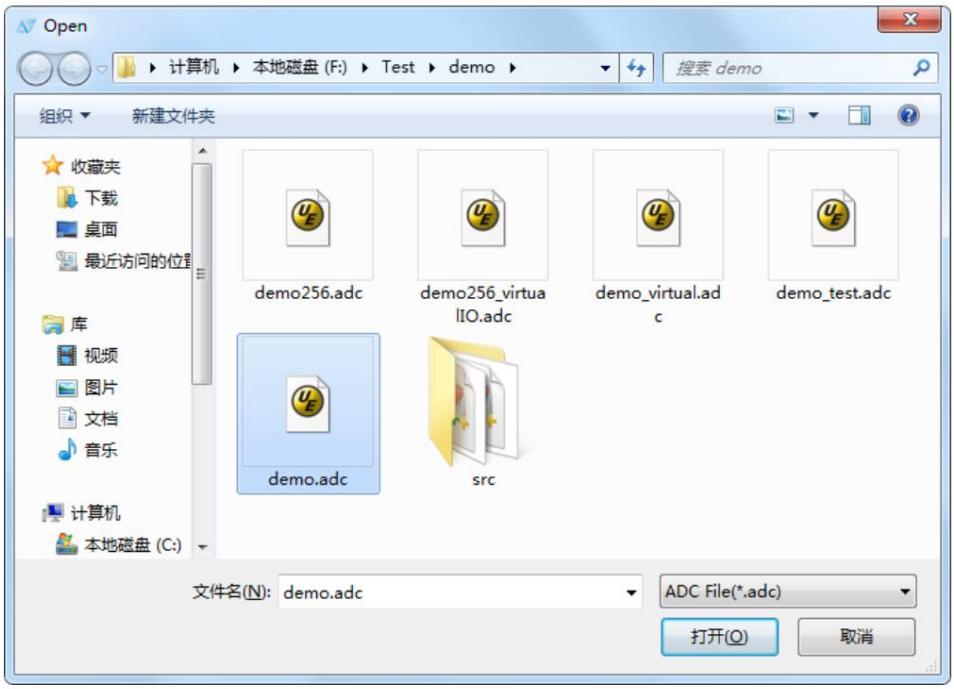


2. Click the Save button, set the file path to the project directory, the file name is demo.adc, and then click save.

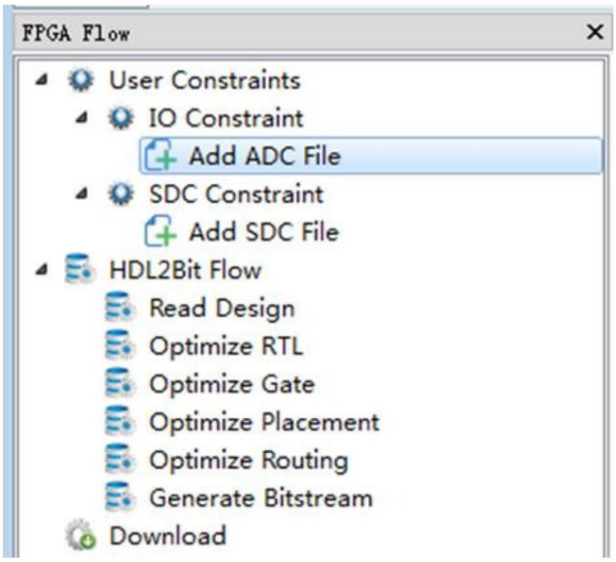
Machine Translated by Google

Add **ADC** file

1. Right-click **Constraints in Project in the Hierarchy Navigation panel and** select **Add ADC File,** select demo.adc, open.



Or by double-clicking **Add ADC File** under **User Constraints** in the FPGA Flow panel , select demo.adc, click to open.

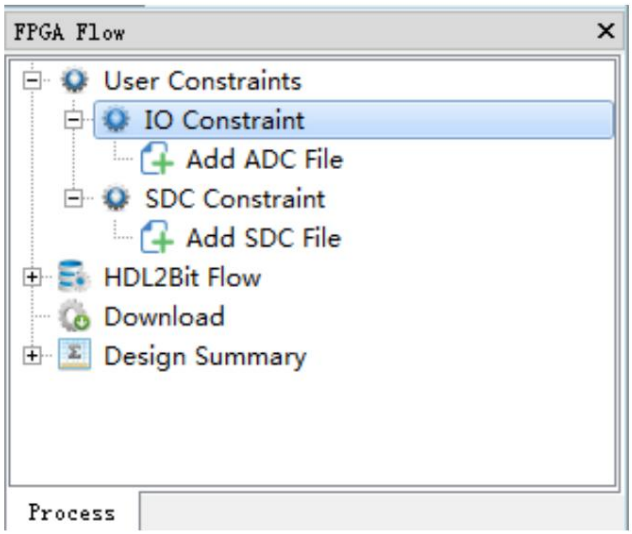


Machine Translated by Google

4.1.2 Interface setting **IO** constraints

There are two ways to open the interface for setting IO constraints:

1. In the **FPGA Flow** panel, expand **User Constraints and double-click IO Constraint;**



2. Expand **Tools** in the menu bar and double-click **IO Constraint.**

Set the Bank , **Location, PullType ,**

IOStandard and other parameters.

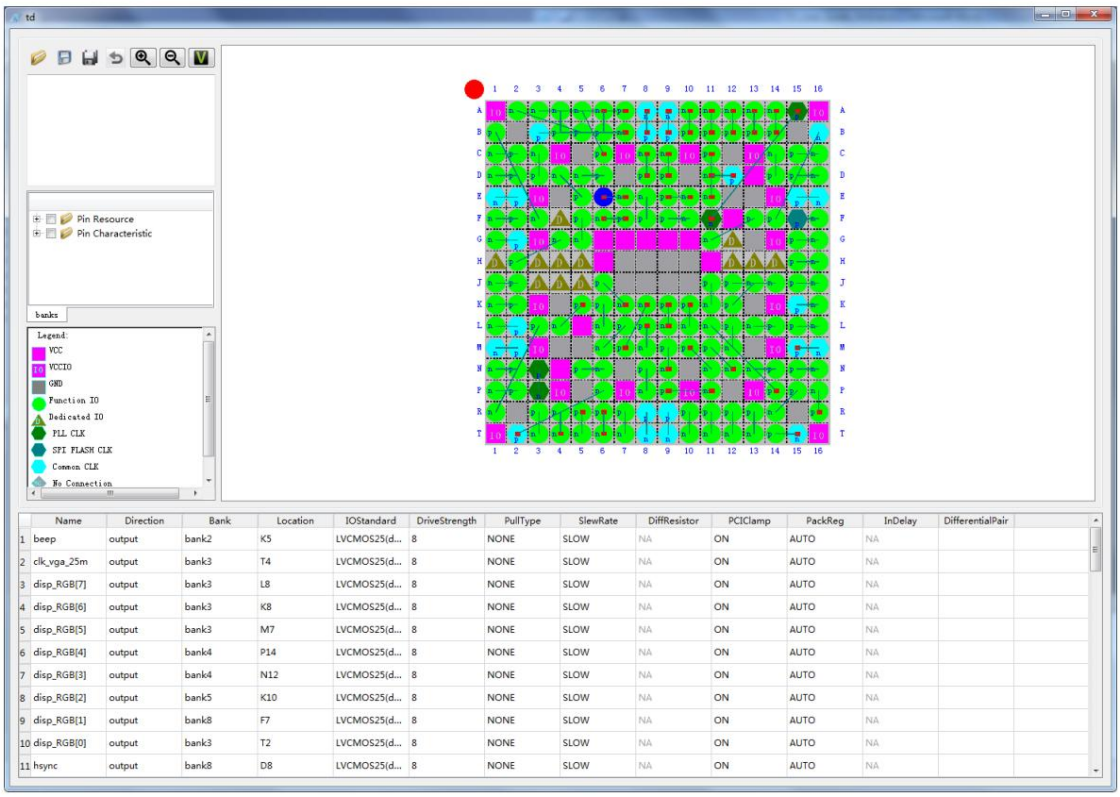
When the chip package is BGA256, the IO interface configuration properties are shown in the following table

Machine Translated by Google

Table 4-1 IO interface configuration properties

category	colour	shape	Function
	magenta	square	power supply
	grey	square	land
	green	round	Function IO
	blue	hexagon	Common CLK
	dark green	hexagon	PLL
	Navy blue	hexagon	Spi flash clock
	dark yellow	triangle	Dedicated IO
	light grey	diamond	No Connection
		short slash	differential pair

When the chip package is BGA256, the configuration interface of IO constraints is shown in the figure below



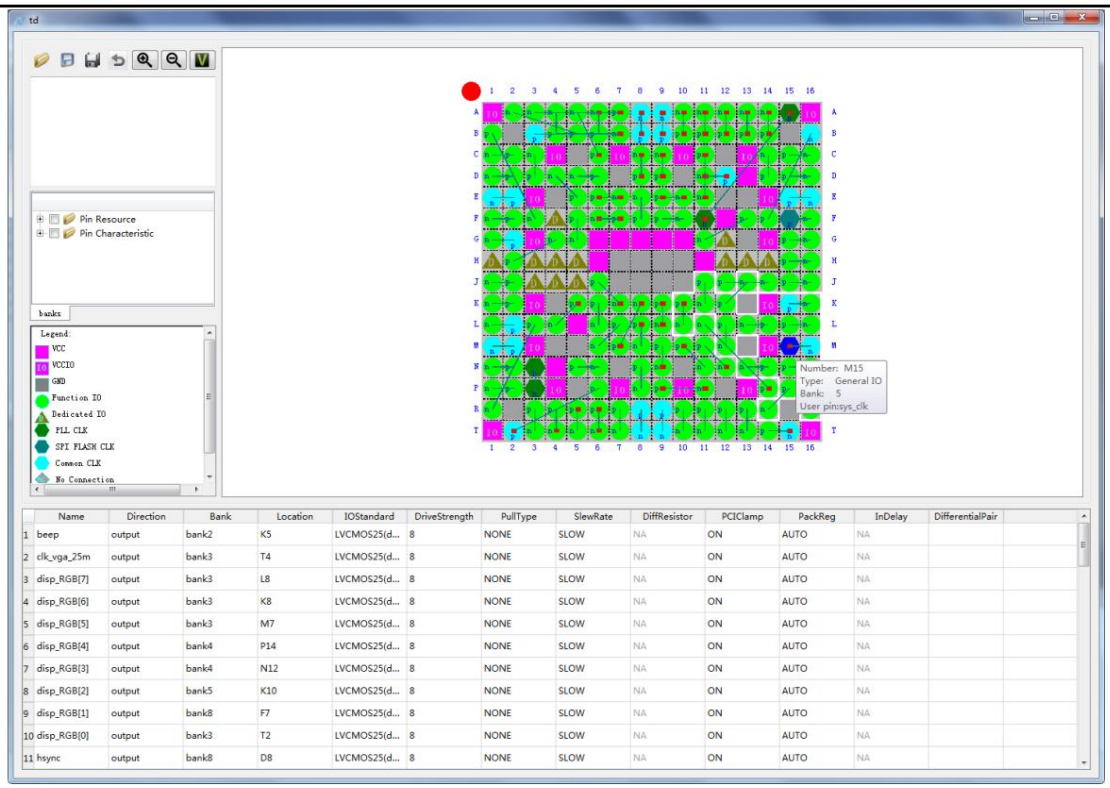
	Name	Direction	Bank	Location	IOStandard	DriveStrength	PullType	SlewRate	DiffResistor	PCIClamp	PackReg	InDelay	DifferentialPair
1	beep	output	bank2	K5	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
2	clk_vga_25m	output	bank3	T4	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
3	disp_RGB[7]	output	bank3	L8	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
4	disp_RGB[6]	output	bank3	K8	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
5	disp_RGB[5]	output	bank3	M7	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
6	disp_RGB[4]	output	bank4	P14	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
7	disp_RGB[3]	output	bank4	N12	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
8	disp_RGB[2]	output	bank5	K10	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
9	disp_RGB[1]	output	bank8	F7	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
10	disp_RGB[0]	output	bank3	T2	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
11	hsync	output	bank8	D8	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	

Small red dots in each shape indicate that the IO port is occupied. When the mouse stays on an IO port, there will be a floating

The window displays the information of the port, and has a white border to display the bank where it is located.

上海安路信息科技股份有限公司
SHANGHAI ANLOGIC INFOTECH CO., LTD.

Machine Translated by Google



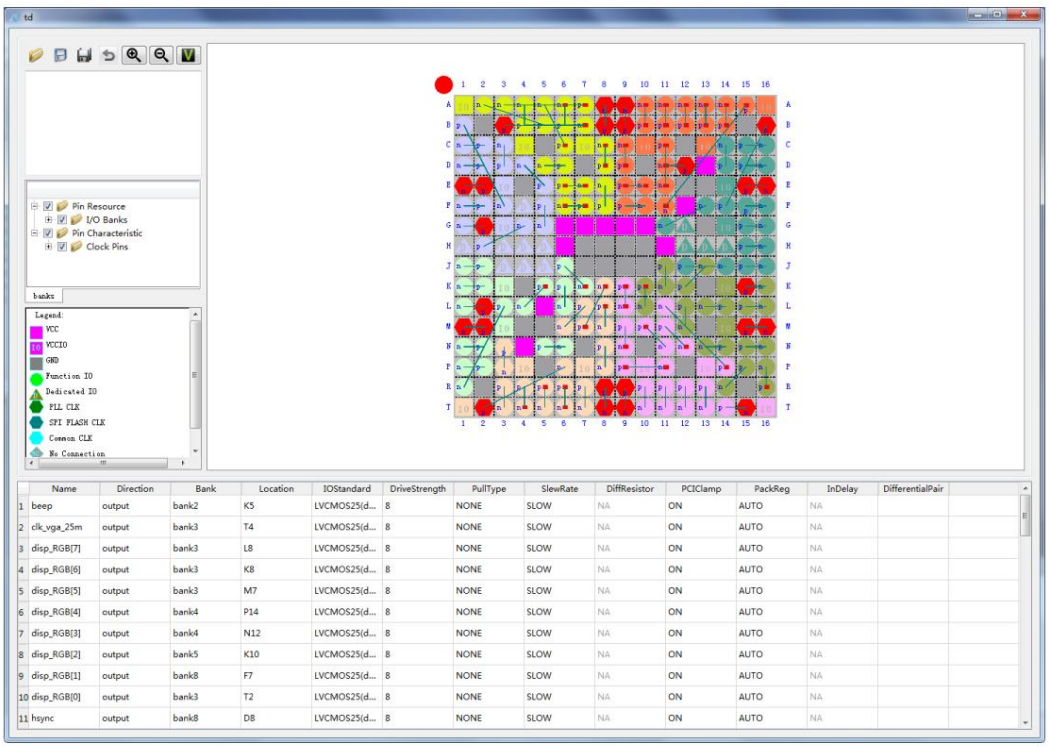
	Name	Direction	Bank	Location	IOStandard	DriveStrength	PullType	SlewRate	DiffResistor	PCIClamp	PackReg	InDelay	DifferentialPair
1	beep	output	bank2	K5	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
2	clk_vga_25m	output	bank3	T4	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
3	disp_RGB[7]	output	bank3	L8	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
4	disp_RGB[6]	output	bank3	K8	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
5	disp_RGB[5]	output	bank3	M7	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
6	disp_RGB[4]	output	bank4	P14	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
7	disp_RGB[3]	output	bank4	N12	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
8	disp_RGB[2]	output	bank5	K10	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
9	disp_RGB[1]	output	bank8	F7	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
10	disp_RGB[0]	output	bank3	T2	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
11	hsync	output	bank8	D8	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	

Users can also distinguish Banks through Pin Resource and Pin Characteristic on the left side of the interface.

The various colors displayed are shown below. Among them, red: clock, magenta: power, gray: ground, these three

This type of port does not belong to any Bank. The red origin in the upper left corner corresponds to the small concave point on the chip, indicating that the chip leads

The starting point of the foot.



	Name	Direction	Bank	Location	IOStandard	DriveStrength	PullType	SlewRate	DiffResistor	PCIClamp	PackReg	InDelay	DifferentialPair
1	beep	output	bank2	K5	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
2	clk_vga_25m	output	bank3	T4	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
3	disp_RGB[7]	output	bank3	L8	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
4	disp_RGB[6]	output	bank3	K8	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
5	disp_RGB[5]	output	bank3	M7	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
6	disp_RGB[4]	output	bank4	P14	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
7	disp_RGB[3]	output	bank4	N12	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
8	disp_RGB[2]	output	bank5	K10	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
9	disp_RGB[1]	output	bank8	F7	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
10	disp_RGB[0]	output	bank3	T2	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	
11	hsync	output	bank8	D8	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA	

Machine Translated by Google

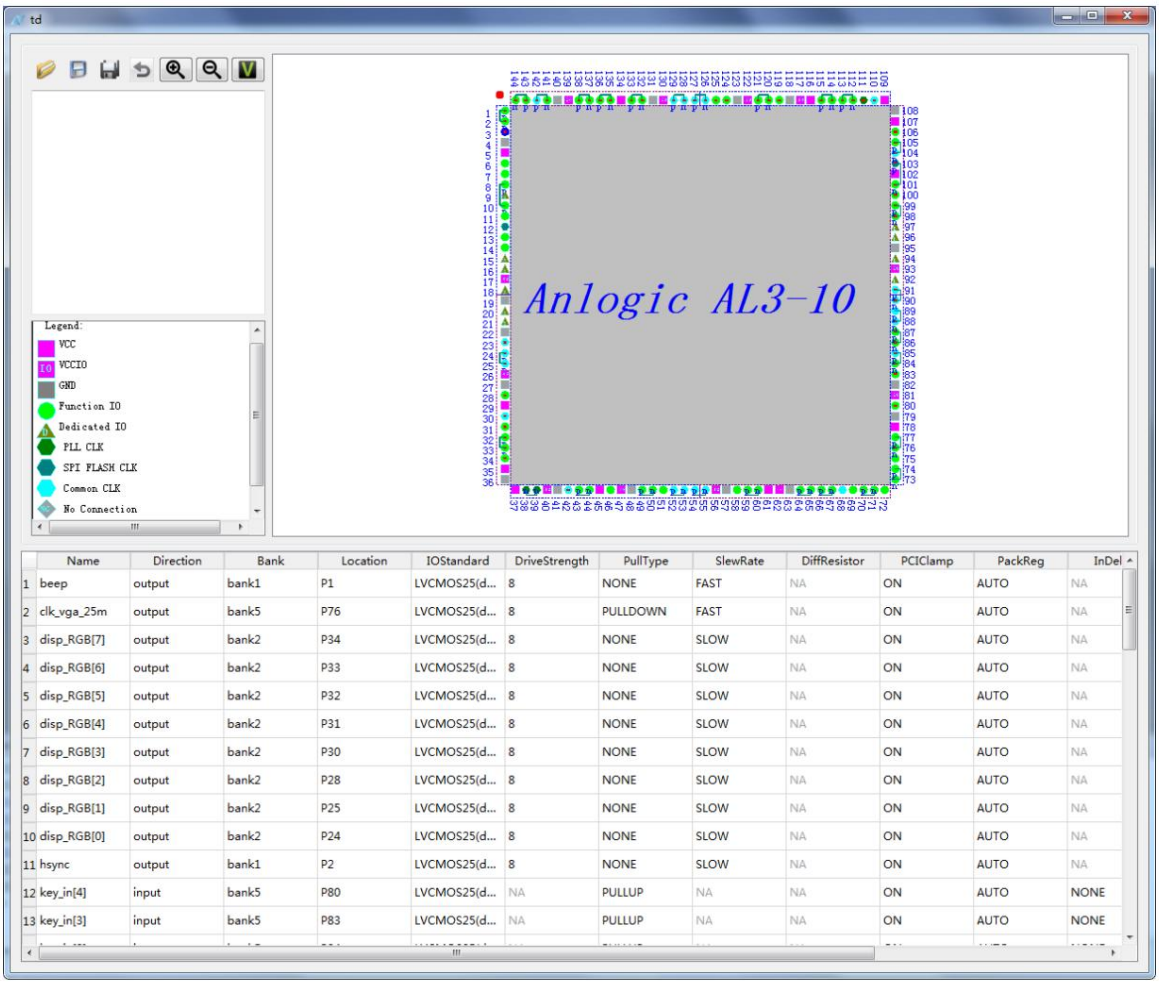
The short slashes represent differential pairs, such as B1 (positive) and F3 (negative) in the figure above. If you choose B1

If the IOStandard attribute is LVDS25, the TD will automatically assign the corresponding differential pair pin F3 (negative).

The IOStandard attribute is also LVDS25, and the differential signal corresponding to beep in the following figure is beep(n).

Name	Direction	Bank	Location	IOStandard	DriveStrength	PullType
vsync	output	bank1	P14	LVCMOS33	4	NONE
hsync	output	bank1	P3	PCI33	1.5	NONE
beep	output	bank1	P2	LVDS25	NA	NONE
beep(n)	output	bank1	P1	LVDS25	NA	NONE

When the chip package is LG144, the IO constraint configuration properties are shown in Table 4-1, and the configuration interface is as follows:



	Name	Direction	Bank	Location	IOStandard	DriveStrength	PullType	SlewRate	DiffResistor	PCIClamp	PackReg	InDel
1	beep	output	bank1	P1	LVCMOS25(d...	8	NONE	FAST	NA	ON	AUTO	NA
2	clk_vga_25m	output	bank5	P76	LVCMOS25(d...	8	PULLDOWN	FAST	NA	ON	AUTO	NA
3	disp_RGB[7]	output	bank2	P34	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA
4	disp_RGB[6]	output	bank2	P33	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA
5	disp_RGB[5]	output	bank2	P32	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA
6	disp_RGB[4]	output	bank2	P31	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA
7	disp_RGB[3]	output	bank2	P30	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA
8	disp_RGB[2]	output	bank2	P28	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA
9	disp_RGB[1]	output	bank2	P25	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA
10	disp_RGB[0]	output	bank2	P24	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA
11	hsync	output	bank1	P2	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA
12	key_in[4]	input	bank5	P80	LVCMOS25(d...	NA	PULLUP	NA	NA	ON	AUTO	NONE
13	key_in[3]	input	bank5	P83	LVCMOS25(d...	NA	PULLUP	NA	NA	ON	AUTO	NONE

BANK and LOCATION settings:

When the chip package is different, the number of pins and pin names are different, and the number of banks is also different. The Anlogic AL3-10 series

For example, the IO pins are divided into 8 Banks. When setting the pin position, you can specify the Bank first , and then specify the

上海安路信息科技股份有限公司
SHANGHAI ANLOGIC INFOTECH CO., LTD.

Machine Translated by Google

Location, or directly specify Location, TD will automatically match the corresponding Bank. If only **Bank** is selected , not

When specifying **Location ,** TD will select the Pin with the smallest pin number in the **Bank** as this pin by default .

Location . Location also supports direct input of the pin name, double-click at **Location** and input the desired pin

pin name, TD will also automatically match the corresponding **Bank.**

	Name	Direction	Bank	Location	IOStandard	DriveStrength	PullType	SlewRate
1	beep	output		P1	LVCMOS25(d...	8	NONE	SLOW
2	disp_RGB[0]	output	bank3	T2	LVCMOS25(d...	8	NONE	SLOW

When the specified Location is **VirtualIO ,** the Port is assigned to a non-physical IO and does not belong to any

Bank, when synthesizing wiring, will not allocate any IO resources to it, and all the level parameters of VirtualIO are not available.

set up. The interface supports one-click setting of the **VirtualIO** function. Click the button in the upper left corner, and the unassigned pins will be automatically assigned.

The Port is allocated as VirtualIO.

In addition, the interface supports the drag and drop function, click to select a Port Name, and drag it to the desired interface on the upper graphical interface.

The Location to assign to this Port. After the drag is completed, a red dot appears in the Location, and the location of the Port below

The information will also change accordingly.

	Name	Direction	Bank	Location	IOStandard	DriveStrength	PullType	SlewRate
1	beep	output		virtualIO	LVTTL33	8	NONE	Slow
2	hsync	output		virtualIO	PCI33	1.5	NONE	Slow

	Name	Direction	Bank	Location	IOStandard	DriveStrength	PullType	SlewRate	DiffResistor
1	beep	output	bank4	P54	LVTTL33	8	NONE	Slow	NA
2	hsync	output		P1	PCI33	1.5	NONE	Slow	NA
3	vsync	output		P12	LVCMOS33	4	NONE	Slow	NA
4	disp_RGB1...	output		P6	LVCMOS33	4	NA	NA	NA
5	disp_RGB1...	output		P7	LVCMOS33	4	NONE	Slow	NA
6	disp_RGB1...	output		P8	LVCMOS25(default)	8	NONE	Slow	NA
7	disp_RGB1...	output	bank1	P10	LVCMOS25(default)	8	NA	NA	NA
8	disp_RGB1...	output	bank1	P11	LVCMOS25(default)	8	NA	NA	NA

bank1
bank2
bank3
bank4
bank5
bank6
bank7
bank8

Machine Translated by Google

IOstandard settings:

IOStandard sets the level standard of the IO port. Each **bank** can be arbitrarily set to support the device's

Level standard, the level of different level standards in the same **Bank** should be consistent. TD provides LVCMOS, LVDS,

LVTTL33, PCI33 for users to choose. Among them, **LVCMOS** has 1.2v, 1.5V, 1.8V, 2.5V, 3.3V power

pressure can be selected. LVDS is a differential pair input and output. When the selected IO port is an input signal, only LVDS25 can be selected.

LVDS33, LVPECL33; when the selected IO port is an output signal, **LVDS25_E, LVDS33_E,**

LVPECL33_E. The default level standard is LVCMOS25 (default).

Name	Direction	Bank	Location	IOStandard	DriveStrength	PullType	SlewRate	DiffResistor	PCIClamp	PackReg	InDelay	DifferentialPair
beep	output	bank2	K5	LVCMOS33	8	NONE	SLOW	NA	ON	AUTO	NA	
clk_vga_25m	output	bank3	T4	LVCMOS25	8	NONE	SLOW	NA	ON	AUTO	NA	
disp_RGB[0]	output	bank3	T2	LVCMOS12	8	NONE	SLOW	NA	ON	AUTO	NA	
disp_RGB[1]	output	bank8	F7	LVCMOS15	8	NONE	SLOW	NA	ON	AUTO	NA	
disp_RGB[2]	output	bank5	K10	LVCMOS18	8	NONE	SLOW	NA	ON	AUTO	NA	
disp_RGB[3]	output	bank4	N12	LVCMOS33	8	NONE	SLOW	NA	ON	AUTO	NA	
disp_RGB[4]	output	bank4	P14	LVTTL33	8	NONE	SLOW	NA	ON	AUTO	NA	
disp_RGB[5]	output	bank3	M7	PCI33	8	NONE	SLOW	NA	ON	AUTO	NA	
disp_RGB[6]	output	bank3	K8	LVDS25	8	NONE	SLOW	NA	ON	AUTO	NA	
disp_RGB[7]	output	bank3	L8	LVDS25_E	8	NONE	SLOW	NA	ON	AUTO	NA	
hsync	output	bank8	D8	LVPECL33_E	8	NONE	SLOW	NA	ON	AUTO	NA	

Settings for **DriveStrength**

DriveStrength sets the drive capability of the IO port. Different level standards have different driving capabilities, such as

The drive capability of **LVCMOS25** is 4 , **8, 12, 16 in mA.** The smaller the value of **DriveStrength ,** the higher the drive energy.

The weaker the force, the larger the value of DriveStrength , which means the stronger the driving ability.

Name	Direction	Bank	Location	IOStandard	DriveStrength	PullType
hsync2	output	bank4	P72	LVCMOS25(default)	8	NONE
clk_vga_25m	output	bank5	P77	LVCMOS33	4	NONE
key_in[4]	input	bank5	P80	LVCMOS33	12	NONE
key_in[3]	input	bank5	P83	LVCMOS25(default)	16	NONE
key_in[2]	input	bank5	P84	LVCMOS25(default)	NA	NONE

PullType settings:

PullType sets the pull-up type of the IO **port. There are four options for PullType: PULLUP, PULLDOWN ,**

上海安路信息科技股份有限公司
SHANGHAI ANLOGIC INFOTECH CO., LTD.

Machine Translated by Google

NONE, KEEPER. Digital circuits have three states: high-level, low-level, and high-impedance states. When the input is invalid

When the signal is turned on, it can be placed in a state by means of a pull-up (PULLUP) resistor and a pull-down (PULLDOWN) resistor.

stable state. When (KEEPER) is selected, keep the level at the last valid value. When the IO port is set to **LVDS**

At this time, PullType can only be set to **None.**

	Name	Direction	Bank	Location	IOStandard	DriveStrength	**PullType**	SlewRate
1	beep	output	bank4	P54	LVTTL33	8	NONE	Slow
2	hsync	output	bank1	P1	PCI33	1.5	NONE PULLUP PULLDOWN KEEPER	Slow
3	vsync	output	bank1	P12	LVCMOS33	4		Slow
4	disp_RGB1...	output	bank1	P6	LVCMOS33	4	NA	NA
5	disp_RGB1...	output	bank1	P7	LVCMOS33	4	NONE	Slow
6	disp_RGB1...	output	bank1	P8	LVCMOS25(default)	8	NONE	Slow

The meaning and scope of other level parameters are shown in the following table:

Scope of application of parameters		Meaning	optional value
SlewRate single-ended	output	Output Slew Rate	Slow, Med, Fast
DiffResistor differential	input	Differential Termination Resistor 100ohm	100, None
PCIClamp Single-ended	input and output	PCI level standards require clamps	ON, OFF
PackReg input, output	pack registers into pads	Auto means according to the global set value ON, OFF has higher priority than global set value	Auto, ON, OFF
InDelay	enter	Adjust input delay	Actual delay depends on core Types of chips and IOBs
OutDelay output		Adjust output delay	Actual delay depends on core Types of chips and IOBs

Back Annotation function:

If the adc file does not exist in the project or is not completely set, the IO Constraint interface supports back **annotation**

mark function. Run the project directly to Optimize Placement on the FPGA Flow interface, open IO Constraint,

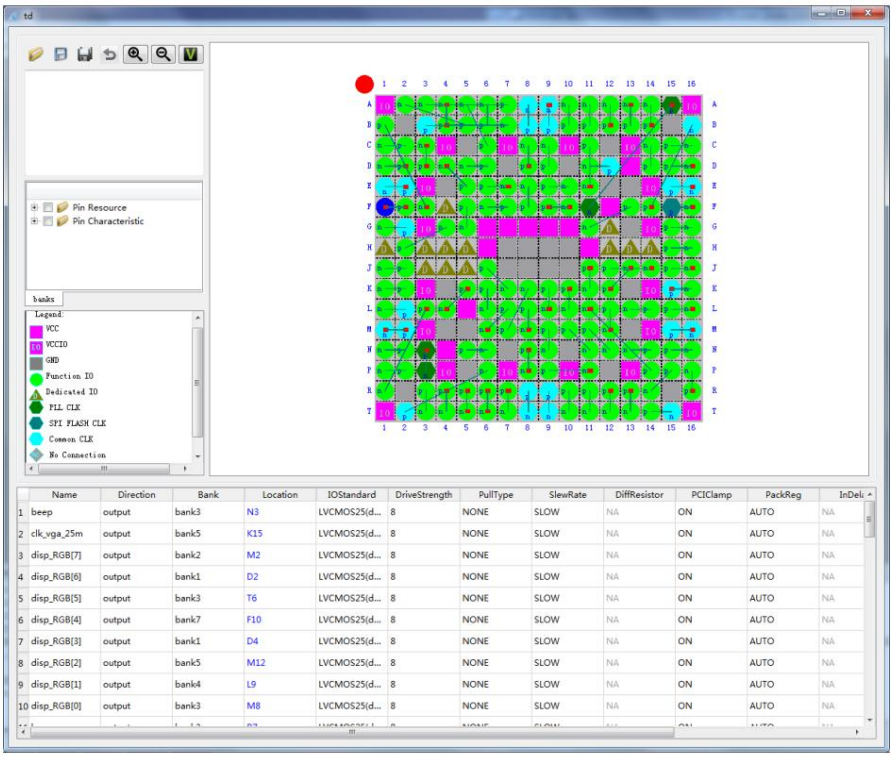
Click the back button in the upper left corner of the interface to automatically assign a Location to the Port without assigned pins, and the corresponding top of the interface will be assigned a Location.

The location of the interface will be displayed as a red dot, and the Location information at the bottom of the interface will be displayed in blue.

When all settings are completed, the user can click the save button in the upper left corner, enter the name of the file, click save,

Machine Translated by Google

At this time, the Location information display at the bottom of the interface turns black.



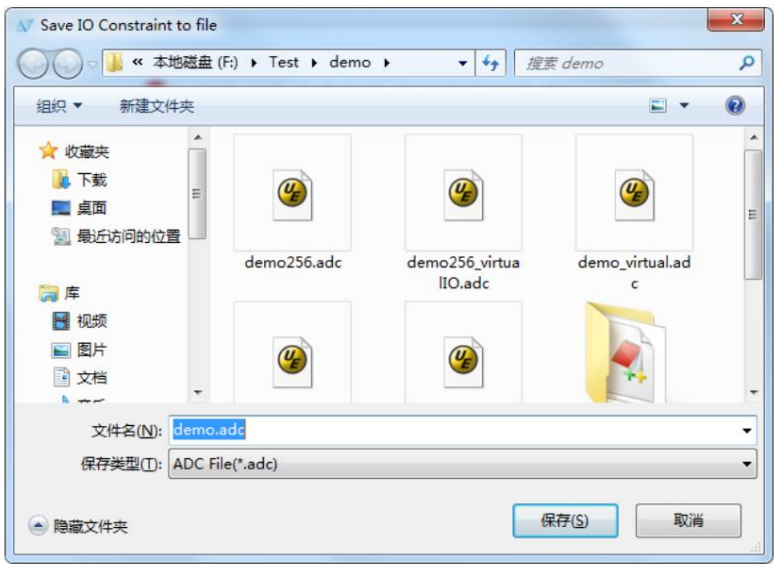
	Name	Direction	Bank	Location	IOStandard	DriveStrength	PullType	SlewRate	DiffResistor	PCIClamp	PackReg	InDel
1	beep	output	bank3	N3	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA
2	clk_vga_25m	output	bank5	K15	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA
3	disp_RGB[7]	output	bank2	M2	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA
4	disp_RGB[6]	output	bank1	D2	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA
5	disp_RGB[5]	output	bank3	T6	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA
6	disp_RGB[4]	output	bank7	F10	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA
7	disp_RGB[3]	output	bank1	D4	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA
8	disp_RGB[2]	output	bank5	M12	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA
9	disp_RGB[1]	output	bank4	L9	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA
10	disp_RGB[0]	output	bank3	M8	LVCMOS25(d...	8	NONE	SLOW	NA	ON	AUTO	NA

When all settings are completed, the user can click the save button in the upper left corner, enter a name for the file, and click save.

If the adc file has been added to the project, when the IO Constraint interface is opened, the information in the adc file will be read and changed.

After setting and saving, the content of adc will be updated directly. If the adc file is not added to the project, after saving, the

Then add the adc file to the project.



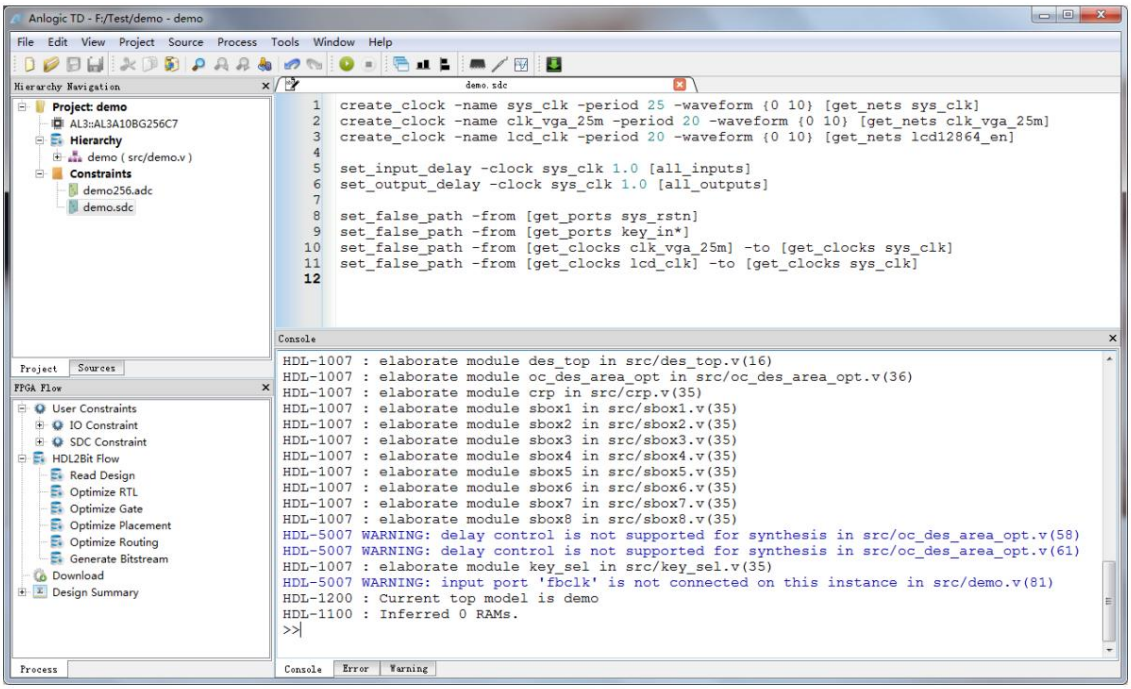
Machine Translated by Google

4.2 Timing Constraints

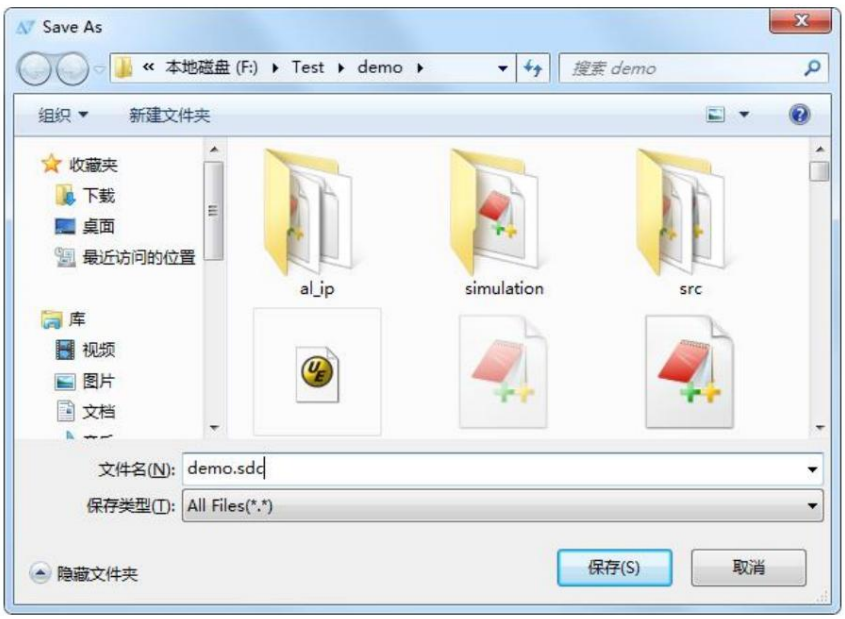
4.2.1 Add timing constraints

New **SDC** file

1. Click the New button and enter the SDC command in the newly opened window



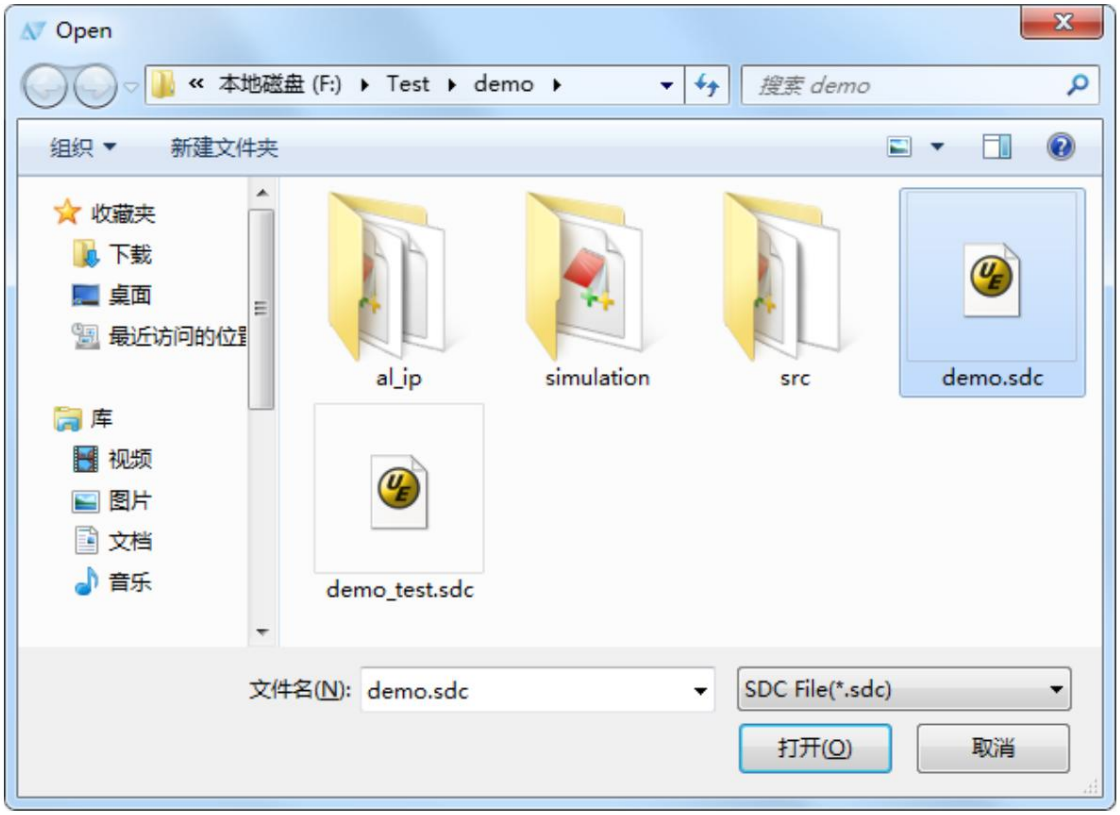
2. Click the Save button, set the file path under the project path, the file name can be set to demo.sdc, and then Click Save.



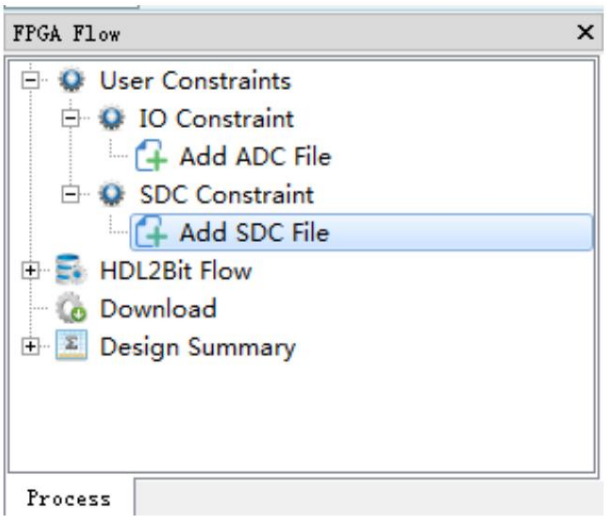
Machine Translated by Google

Add SDC file:

1. Right-click **IO Constraints in the Project in the Hierarchy Navigation panel and select Add SDC File,** select demo.sdc, click Open.



Or expand **User Constraints** in FPGA Flow , double-click **Add SDC File, and** select it as shown above demo.sdc Click to open.

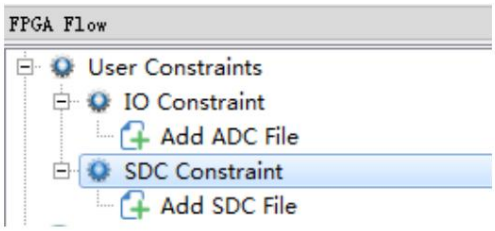


Machine Translated by Google

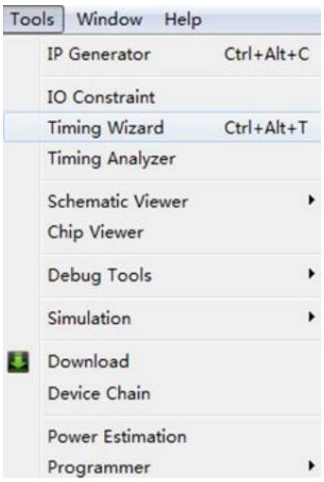
4.2.2 Interface setting timing constraints

There are two ways to open the interface for setting timing constraints:

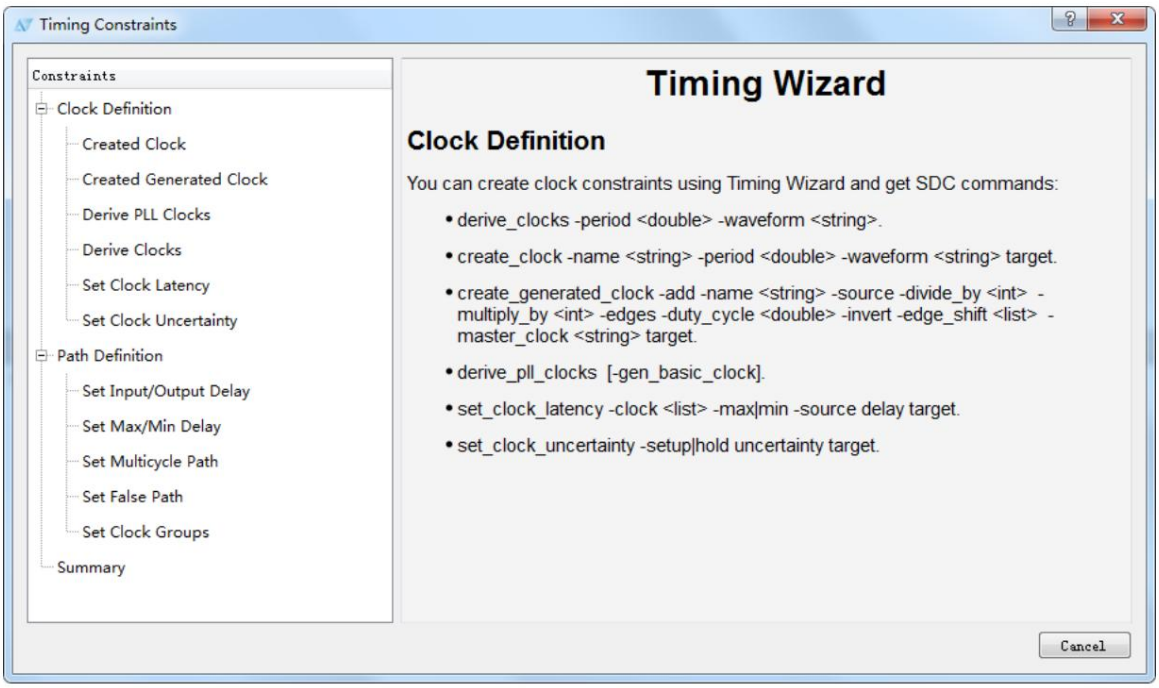
1. In the FPGA Flow panel, expand User Constraints and double-click SDC Constraint;



2. Expand Tools in the menu bar, double-click Timing Wizard, or use the shortcut Ctrl+Alt+T.



The main interface of Timing Wizard is shown in the following figure:



Machine Translated by Google

The Timing Wizard consists of three parts:

1. Clock Definition : This part mainly contains SDC commands to create or define clock constraints;

2. Path Definition: This part mainly contains SDC commands to create or define timing path constraints;

3. Summary : This section summarizes all the constraints of the created clock and timing paths.

The following describes the use of each command in detail:

1. Created Clocks

Format: create_clock -add -name <string> -period <double> -waveform <string> target

Definition: define a clock: target specifies the clock source, which can be nets, pins or ports, if

If target is empty, it means that a virtual clock is defined; -name specifies the clock name, if the item is empty, then

The clock name is the first item in the target list; -period is the period, this option must be specified, and the value must be greater than

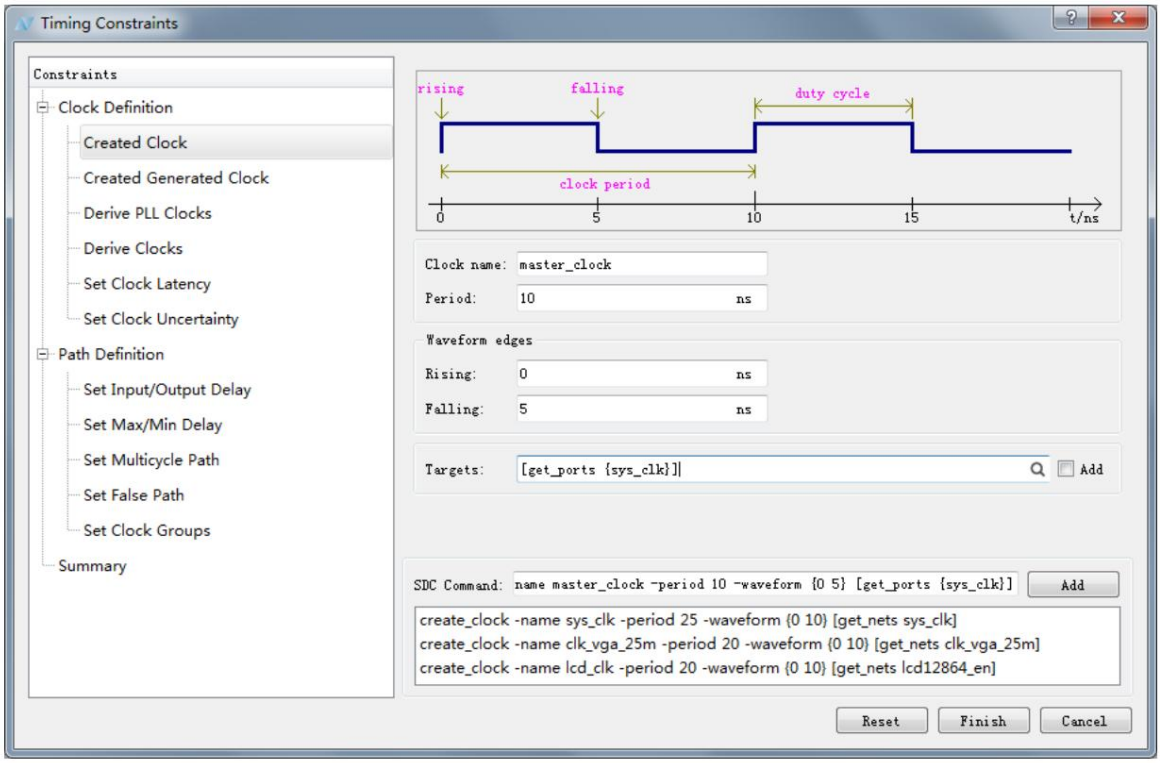
0; -waveform specifies the time point of the first rising edge and the first falling edge, temporarily only supports weekly

When the period contains two clock edges; -add indicates that when the clock has been defined on the target pin,

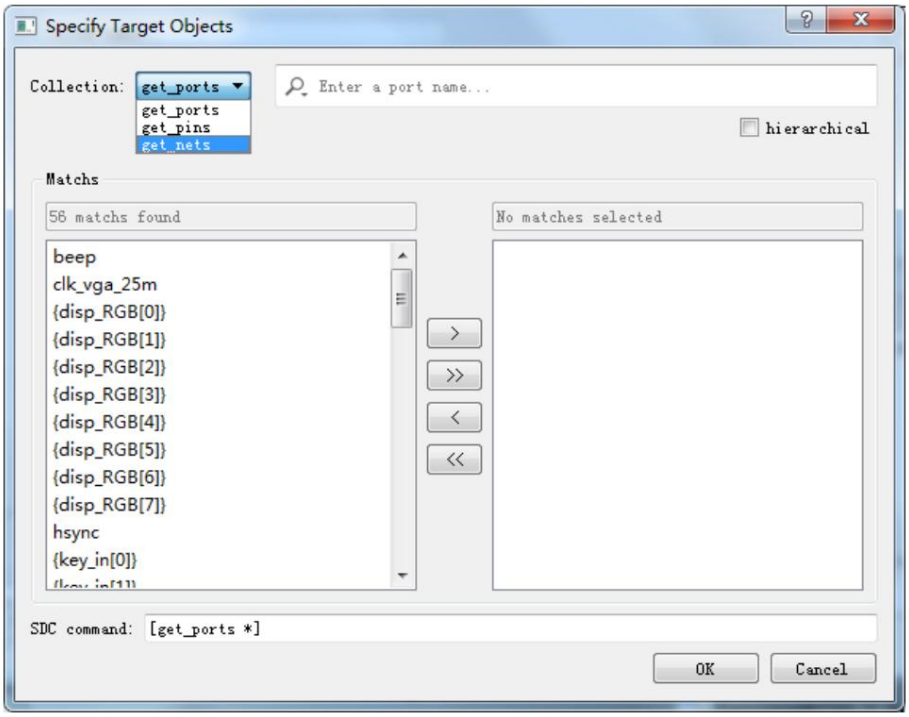
Add a new clock to this pin, otherwise overwrite the existing clock, mainly used in the case of clock multiplexer.

The parameter settings are shown in the following figure:

Machine Translated by Google



Click the Find button behind Targets to select different types of targets.



After the parameters are set, click Add to add the command to the box below, and you can continue to set other parameters

, otherwise the command will not be added to the Summary. If the sdc file has been added to the project, and

and the created_clock command already exists in the sdc file, the existing command will also be added to