Tang Dynasty（TD）
Software manual

Version: 4.2

上海安路信息科技有限公司

2018.07

目录

1 Boot software

Software requirements

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

- TD 4.2

Operating system requirements for TD running under Linux:

- Red Hat Enterprise 6.0 and above

Operating system requirements for TD running under Windows:

- Windows 7 sp1 And above

Hardware requirements

The user's computer hardware requires the following configuration:

- 处理器：1GHz 以上
- 内存：500M 以上
- 硬盘：100M 以上剩余空间

Installation and uninstallation

安装 TD,请双击 TD 安装盘中的 msi 文件，然后遵照安装步骤完成安装

在安装的过程中，若提醒安装 vc_redist.exe，请根据提示进行安装，安装后会继续安装 TD 软件，直至安装完成。若无法成功安装 vc_redist.exe，请更新 Windows 补丁。

卸载 TD,请点击 开始→控制面板→选择 TD→卸载

Start TD software

在 Windows 下 TD 运行的操作系统要求：

- Windows7 及以上版本

Start → All Programs → TD → td.exe

在 Linux 下启动 TD：

- 界面模式：/td_install_dir/td -gui
- 命令模式：/td_install_dir/td

在 Windows 下启动 TD CMD 窗口：

Start → All Programs → td_commands_prompt

获得帮助

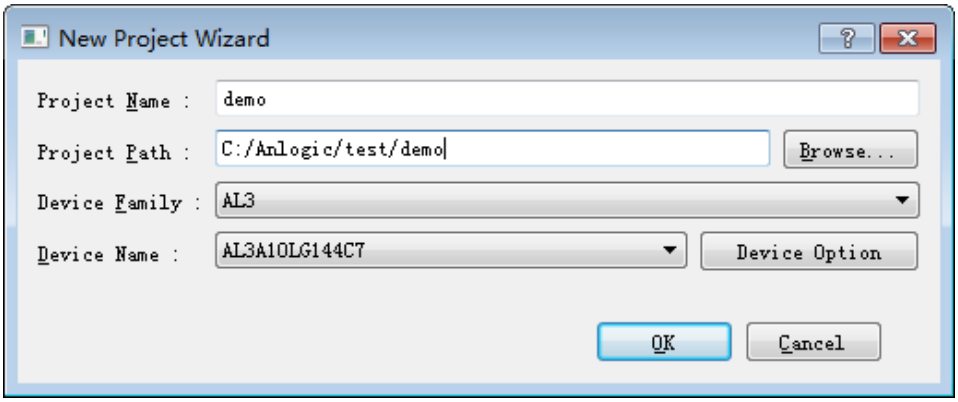
用户可发邮件至 support@anlogic.com 获得关于 TD 软件和相关工具的帮助。

2 Project management

2.1 Create a new project

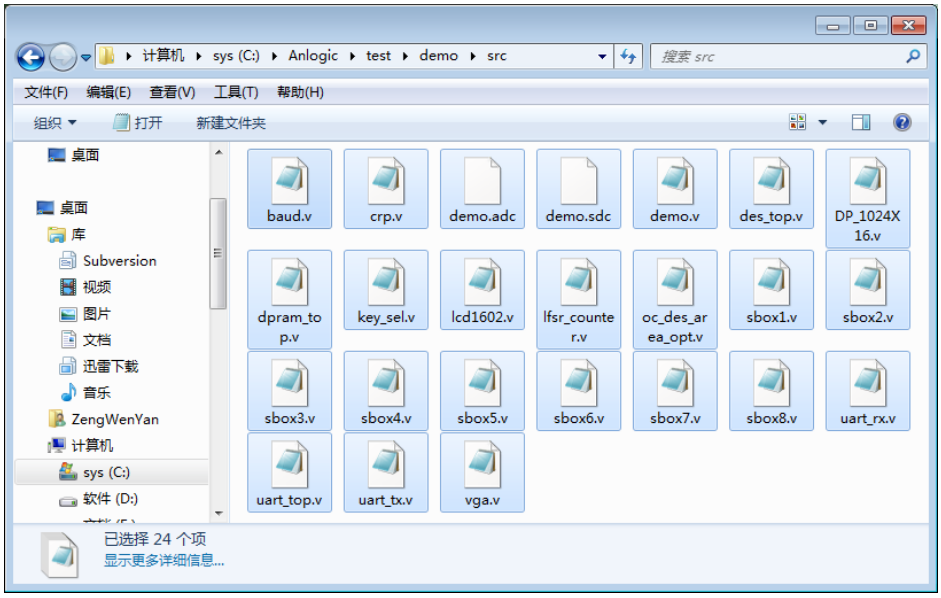
Create a new project: ：

1. 选择 **Project → New Project...** The new project dialog will pop up.
2. Specify the storage path of the created project and enter the project name
3. 选择 **Device Family** 和 **Device Name**，默认为 AL3A10LG144C7。

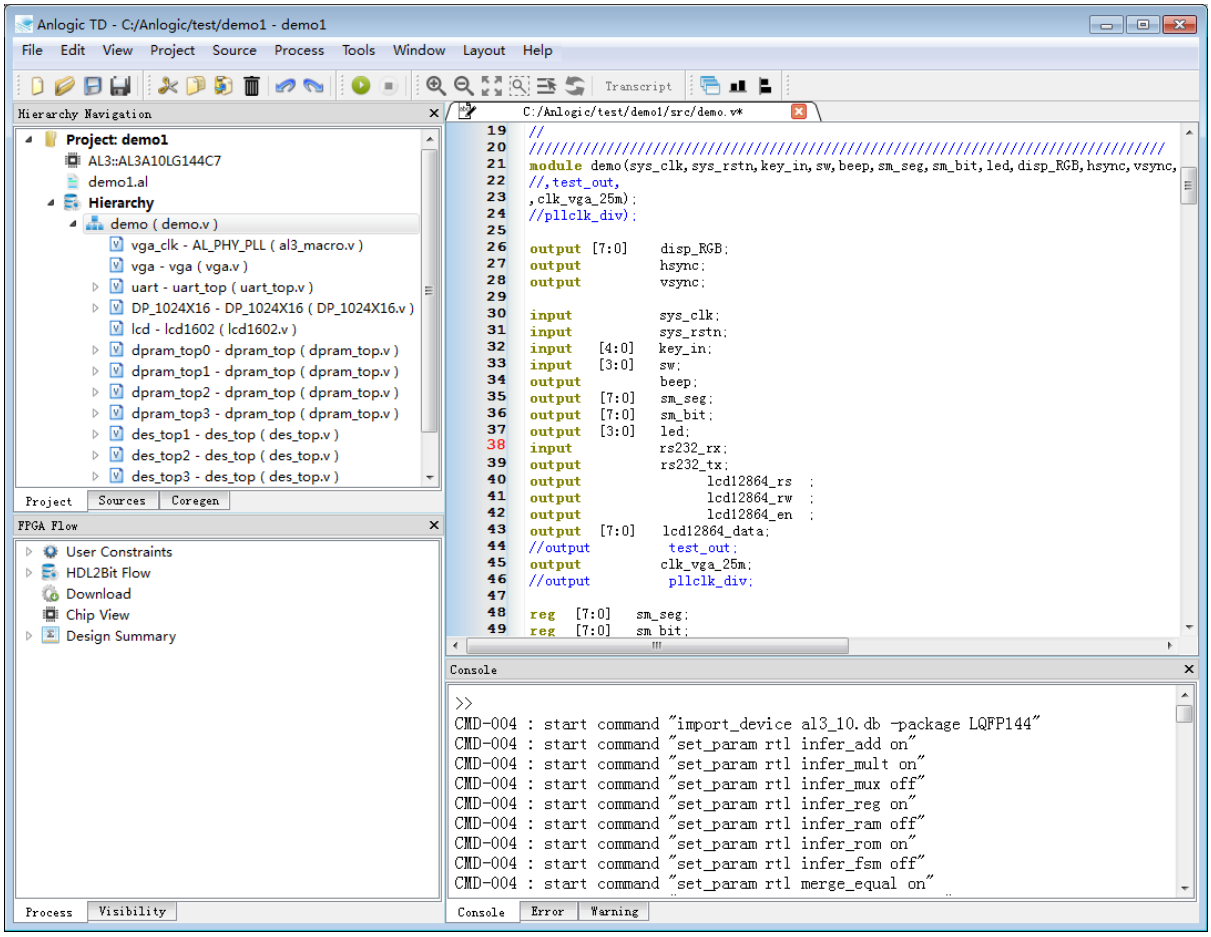


Add source file:

1. select **Source → Add Source...**
2. Select the HDL source file you want to add and click Open.

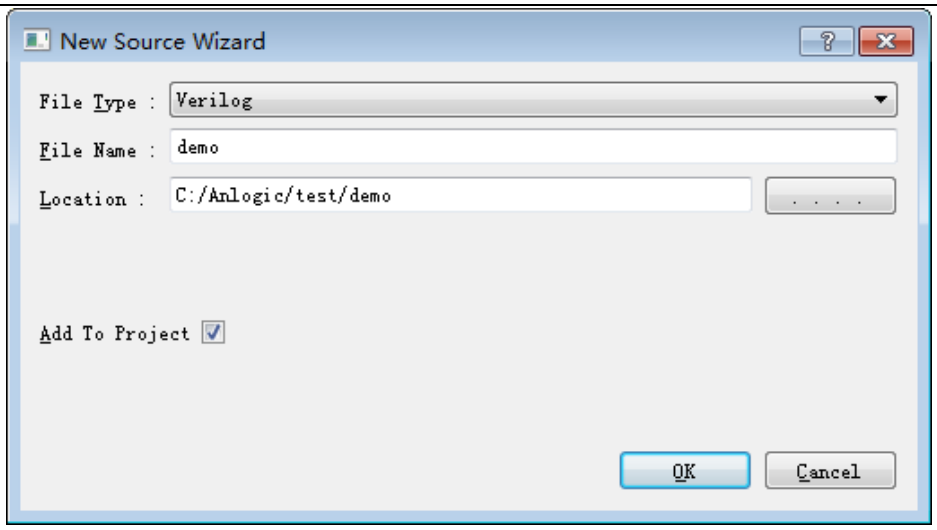


3. At this point, all the source files added can be seen in Hierarchy and the source file can be opened by double-clicking.

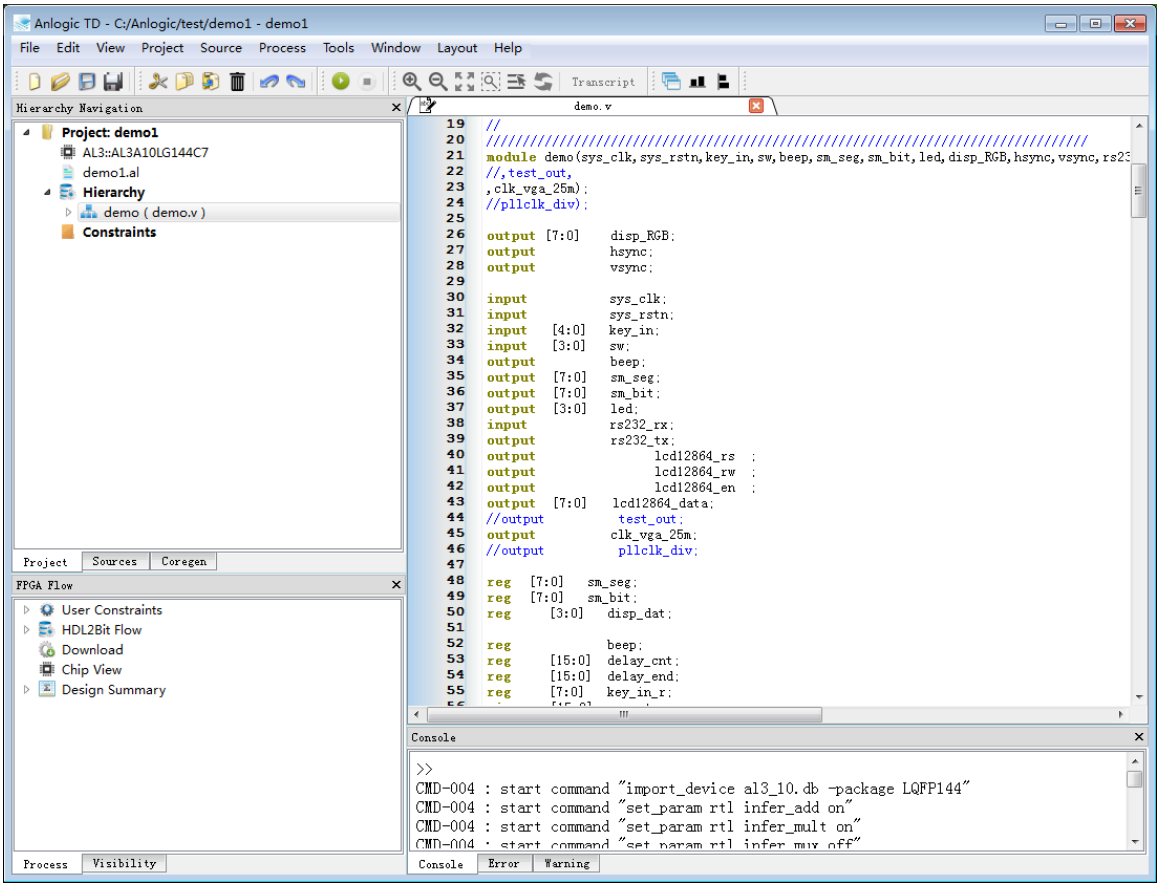


Create a source file:

1. Select Source → New Source...
2. The source file type defaults to Verilog.
3. Enter **File Name**.
4. Make sure **Add To Project** is checked.
5. Click **OK** to finish creating



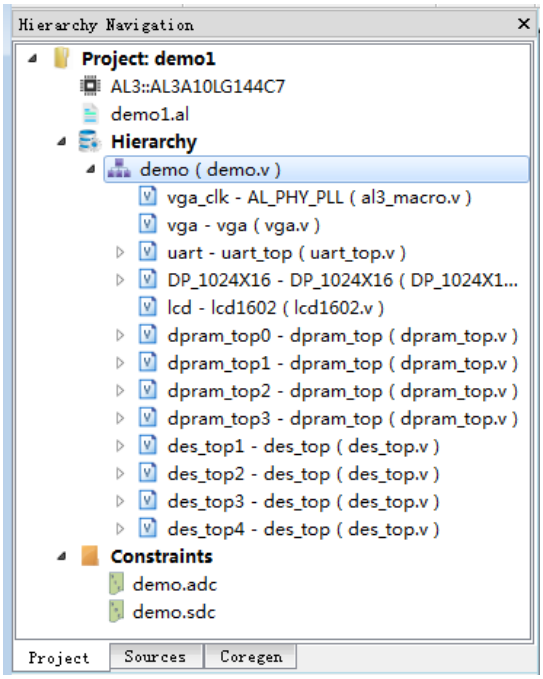
6. Enter the file contents and select **File → Save** to save the file.



Set the top level module

Manually setting the top level module is optional. If the top-level module is not set, the TD software will automatically analyze the module's hierarchy to select the top-level module.

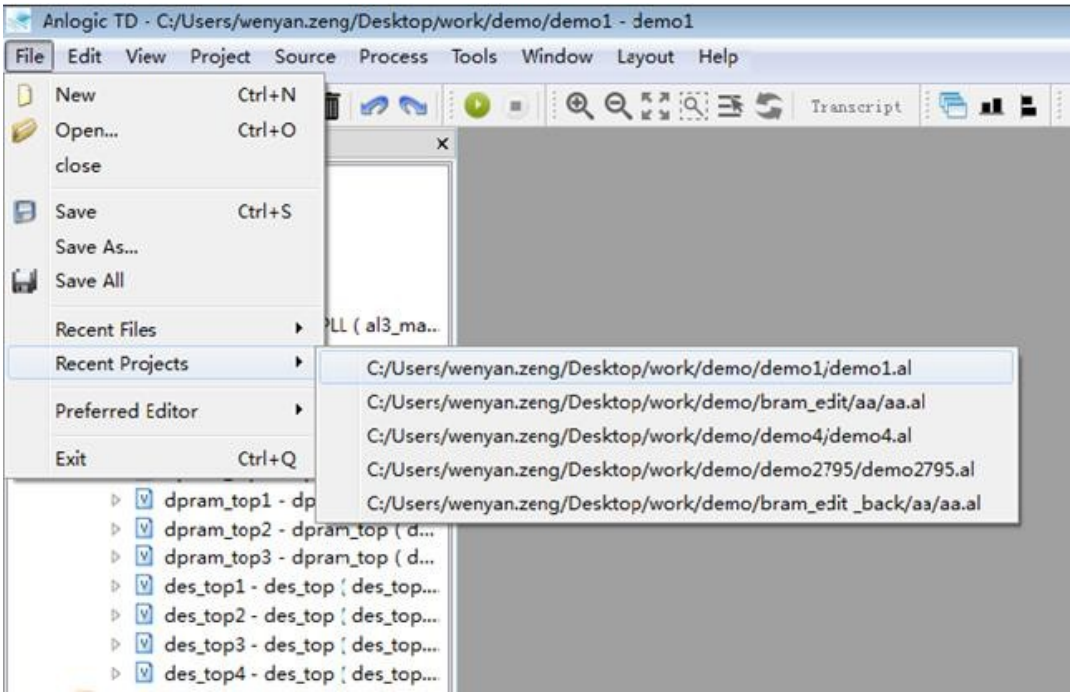
In the Hierarchy Navigation window, right-click on the row of the target module and select Set As Top. A purple top-level module tag will appear in front of the target module.



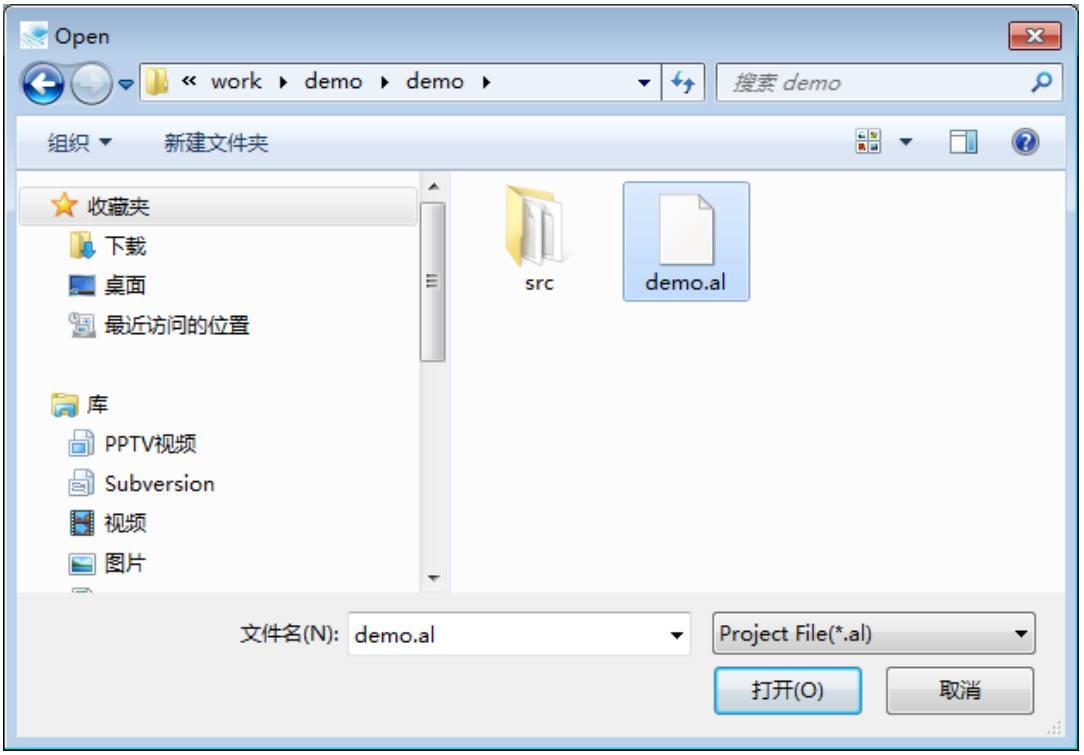
2.2 Open project

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

File ⇒ **Recent Projects** 和 **File** ⇒ **Recent Files** Open a project or file that you have opened.



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



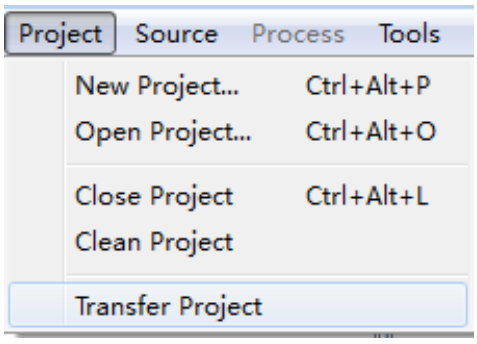
2.3 Conversion project

Users can import projects created by third-party tools (ISE, Quartus II, Diamond) into the TD software. During the conversion process, only the corresponding Source file, IO Constraint file, Timing Constraint file are converted.

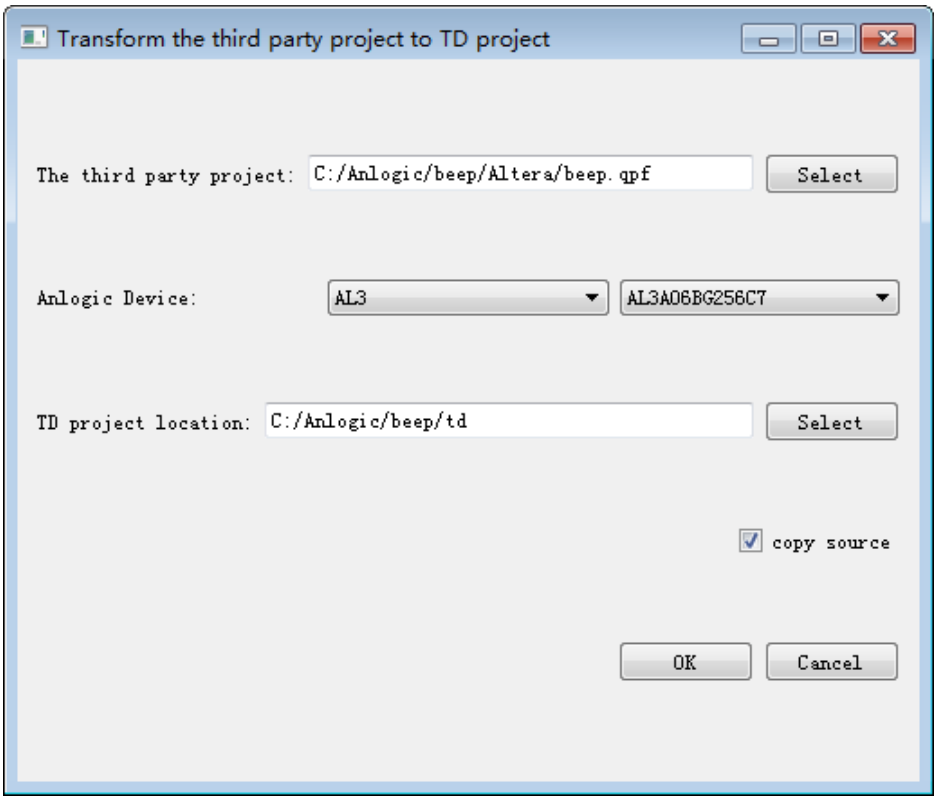
The IO Constraint file is only converted if the third-party device is compatible with the Anlogic device in the pin definition.

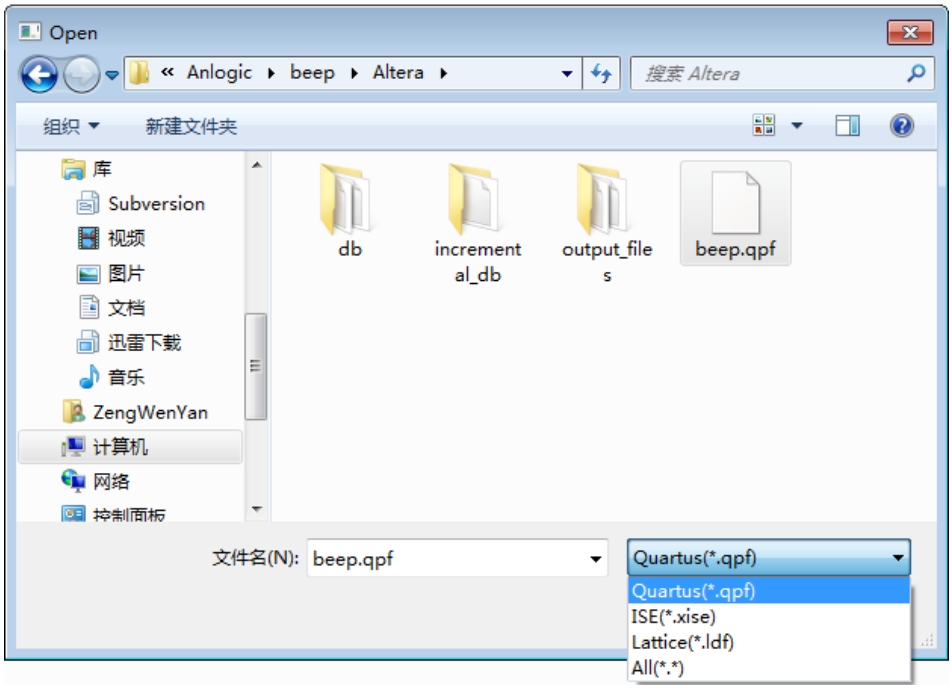
This is an example of a **Quartus II** project converted to a **TD** project.

1. **Project ⇒ Transfer Project**



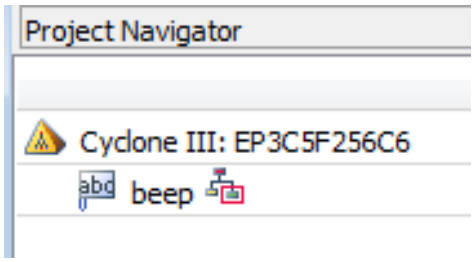
2. Select the project to be converted and the converted project directory, and select “copy source” to copy the source files from the third-party project to the target directory.

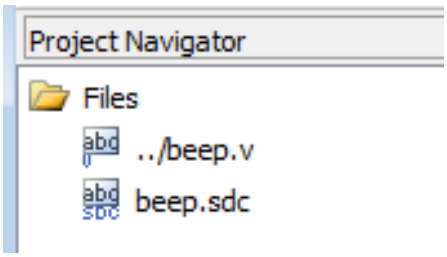




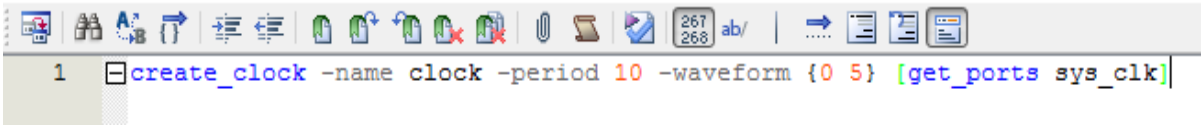
3. TD will open the converted project by default. The following is the comparison between the two projects before and after the conversion.

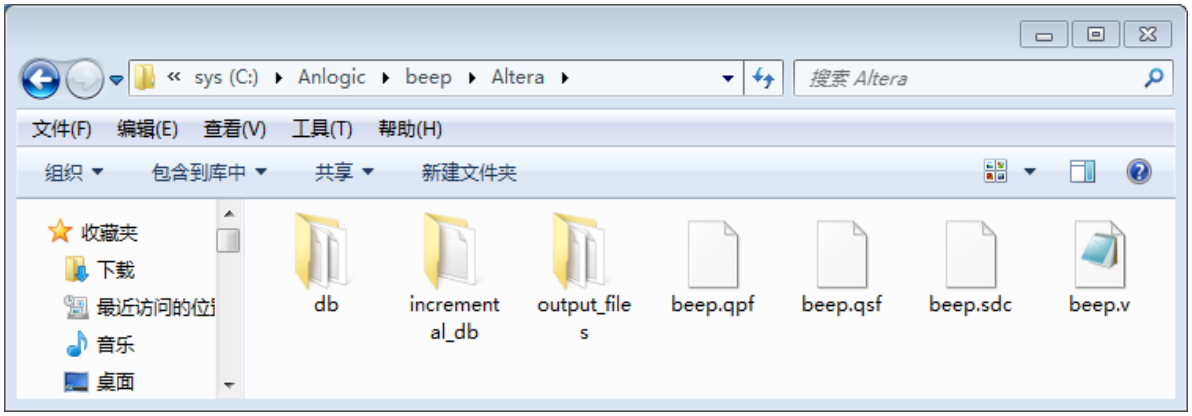
   Quartus Engineering: ：



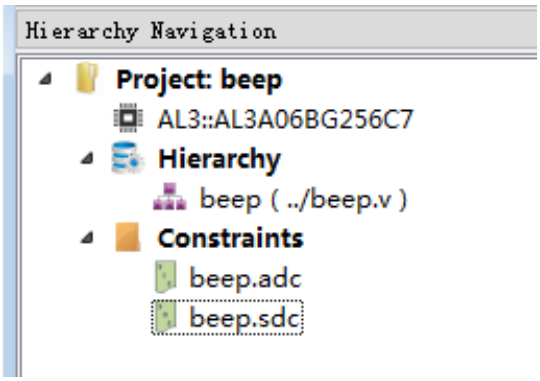


Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate
beep	Output	PIN_A7	8	B8_N0	PIN_A7	2.5 V (default)		8mA (default)	2 (default)
sys_clk	Input	PIN_E1	1	B1_N0	PIN_E1	2.5 V (default)		8mA (default)	
sys_rstn	Input	PIN_T5	3	B3_N0	PIN_T5	2.5 V (default)		8mA (default)	





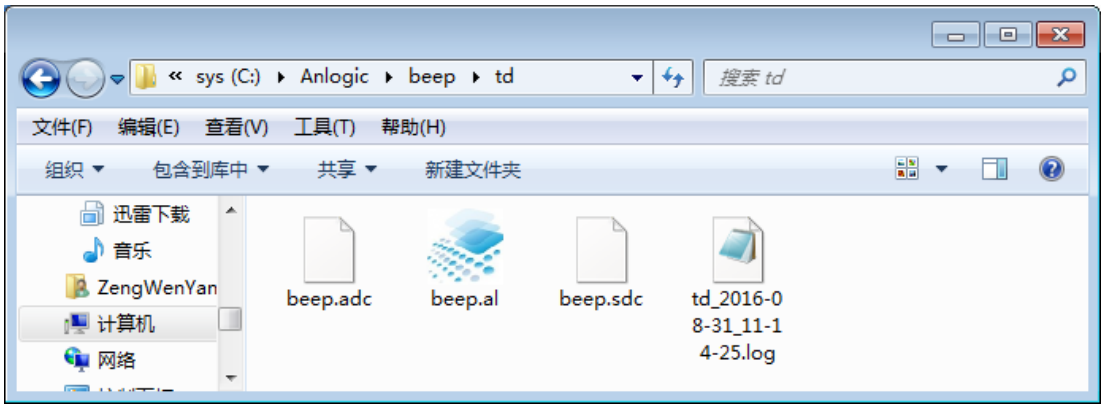
TD project:



	Name	Direction	Bank	Location	PullType	IOStandard	SlewRate	DriveStrength	VREF	DiffResistor
1	beep	output	bank8	A7	NONE	LVCMOS25	MED	8	NONE	NONE
2	sys_clk	input	bank1	E1	PULLUP	LVCMOS25	MED	8	NONE	NONE
3	sys_rstn	input	bank3	T5	PULLUP	LVCMOS25	MED	8	NONE	NONE

beep. sdc

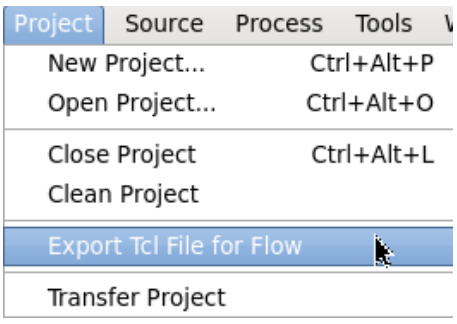
```
1  create_clock -name clock -period 10 -waveform {0 5} [get_ports sys_clk]
```



2.4 Export tcl script

TD software supports running Flow with tcl scripts to reduce user interface operations.

Clicking **Project -> Export Tcl File for Flow** will generate a **prj_name.tcl** file in the project directory that records all the commands for the last Flow operation.



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

1. Open the project demo.al
2. Setting parameters Optimize RTL rtl_sim_model ON

3. Run HDL2Bit Flow

4. Export tcl script demo.tcl

```
demo.tcl*
1  import_device al3_10.db -package LQFP144
2  open_project C:/Users/wenyan.zeng/Desktop/work/al3/demo/demo.al
3  elaborate -top demo
4  set_param rtl rtl_sim_model on
5  optimize_rtl
6  write_verilog "C:/Users/wenyan.zeng/Desktop/work/al3/demo/demo_rtl_sim.v"
7  report_area -file C:/Users/wenyan.zeng/Desktop/work/al3/demo/demo_rtl.area
8  read_sdc "src/demo.sdc"
9  read_adc "src/demo.adc"
10 export_db "C:/Users/wenyan.zeng/Desktop/work/al3/demo/demo_rtl.db"
11 set_param gate map_sim_model on
12 set_param gate gate_sim_model on
13 map_macro
14 map
15 write_verilog "C:/Users/wenyan.zeng/Desktop/work/al3/demo/demo_map_sim.v"
16 pack
17 write_verilog "C:/Users/wenyan.zeng/Desktop/work/al3/demo/demo_gate_sim.v"
18 report_area -file C:/Users/wenyan.zeng/Desktop/work/al3/demo/demo_gate.area
19 export_db "C:/Users/wenyan.zeng/Desktop/work/al3/demo/demo_gate.db"
20 set_param place effort high
21 start_timer
22 place
23 report_area -io_info -file C:/Users/wenyan.zeng/Desktop/work/al3/demo/demo_phy.area
24 set_param route opt_timing high
25 set_param route effort high
26 set_param route fix_hold on
27 route
28 export_db "C:/Users/wenyan.zeng/Desktop/work/al3/demo/demo_pr.db"
29 start_timer
30 report_timing -mode FINAL -net_info -ep_num 7 -path_num 8 -file "C:/Users/wenyan.zeng/Desktop/work/al3/demo/demo_phy.timing" -sdf C:/Users/wenyan.z
31 write_verilog -sdf "C:/Users/wenyan.zeng/Desktop/work/al3/demo/demo.sdf" "C:/Users/wenyan.zeng/Desktop/work/al3/demo/demo_phy_sim.v"
32 bitgen -bit "C:/Users/wenyan.zeng/Desktop/work/al3/demo/demo.bit" -version 0X0000 -svf "C:/Users/wenyan.zeng/Desktop/work/al3/demo/demo.svf" -svf_c
```

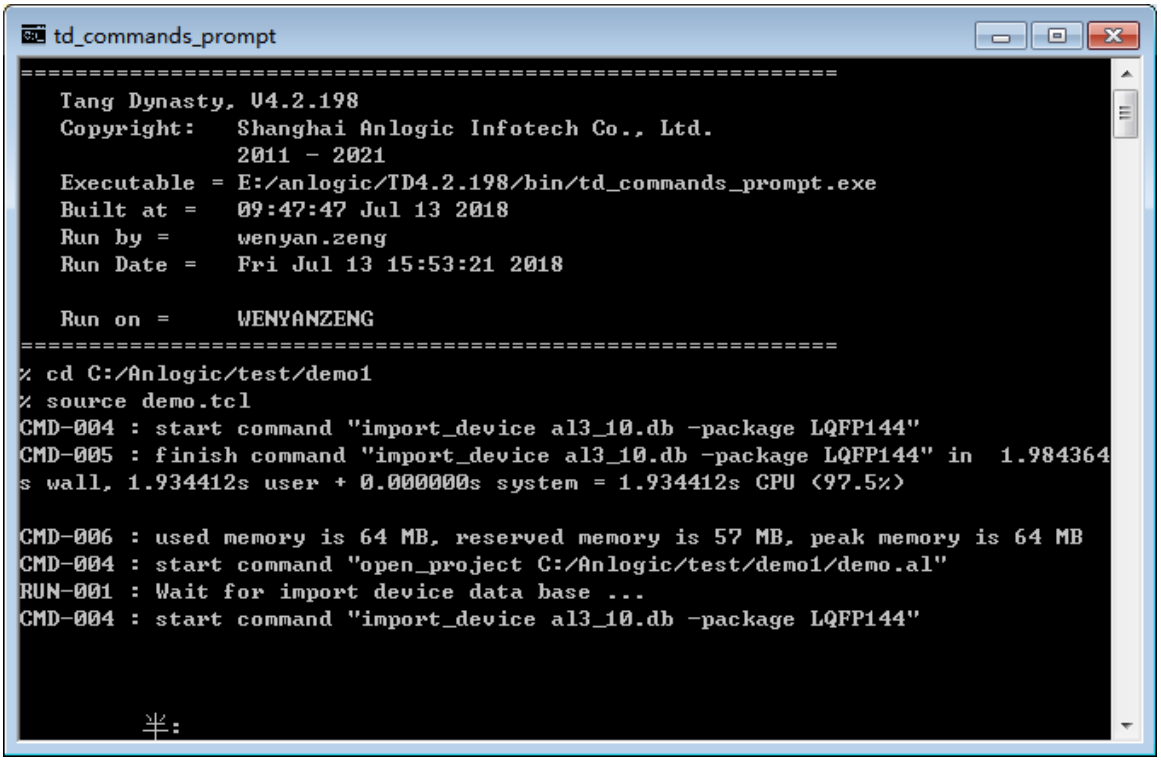
The steps to execute the tcl script are:

1. Launch the TD CMD window under Windows:

Start → All Programs → td_commands_prompt

2. Enter the directory where the project is located

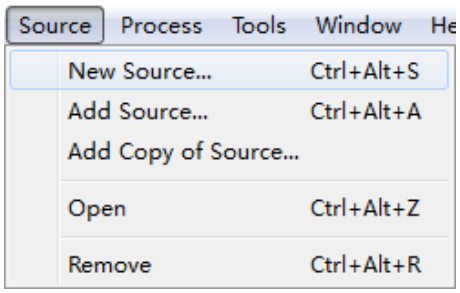
3. Execute the command: source demo.tcl.



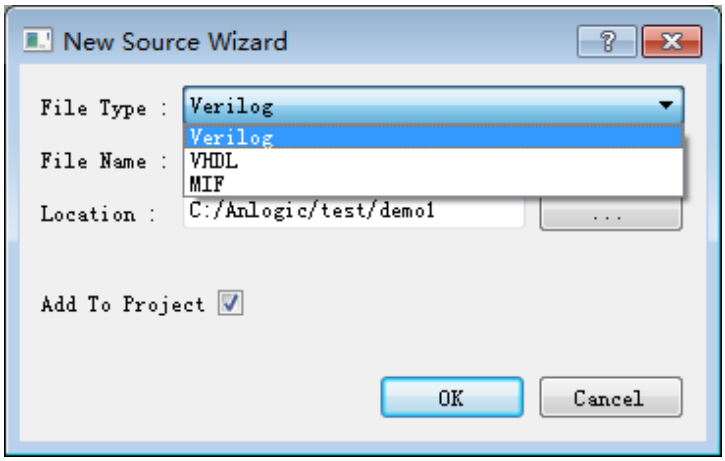
2.5 Source file management

2.5.1. create a new file

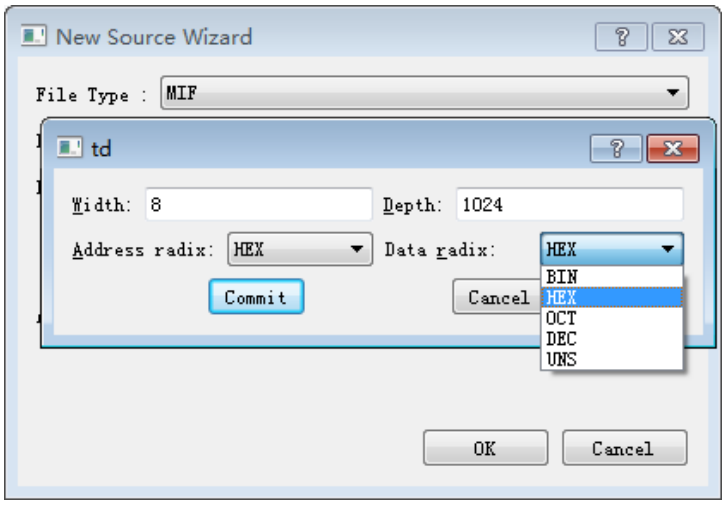
1. **Source ⇒ New Source**



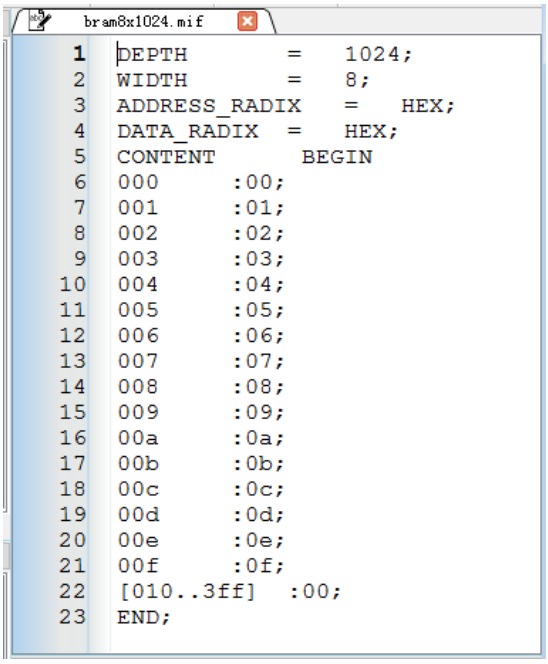
2. Select the type of generated file: **Verilog**, **VHDL**, **MIF**, enter the file name, select the file path, and choose whether to add to the project.



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



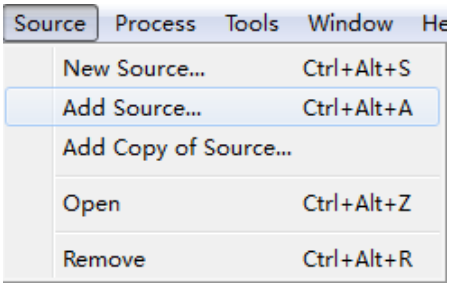
Enter the width and depth of the MIF file, select the cardinality of the data and address, and generate the MIF file as follows:



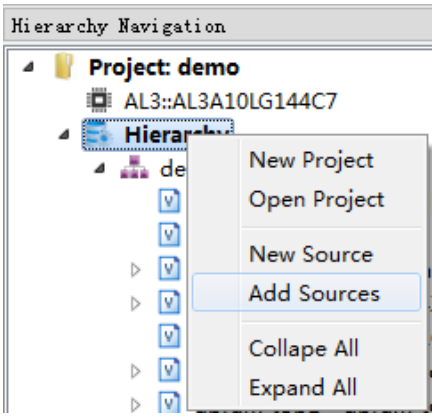
2.5.2 Add and remove files

There are two ways to add a file:

1. **Source ⇒ Add Source**

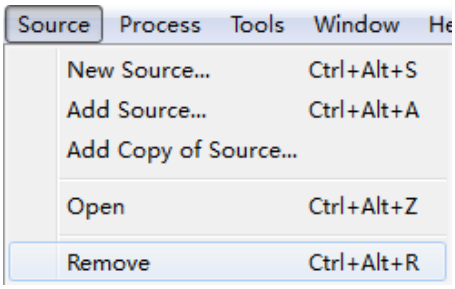


2. In **Hierarchy**, right click and select **Add Sources**

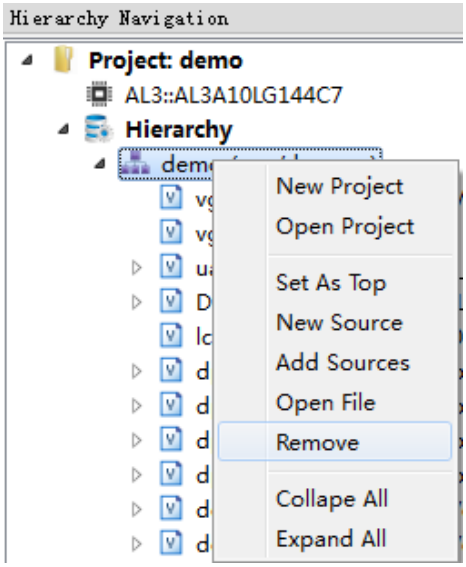


There are also two ways to remove a file:

1. **Source ⇒ Remove**



2. In **Hierarchy**, select a file and right click and **select Remove**



2.5.3 Edit file

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

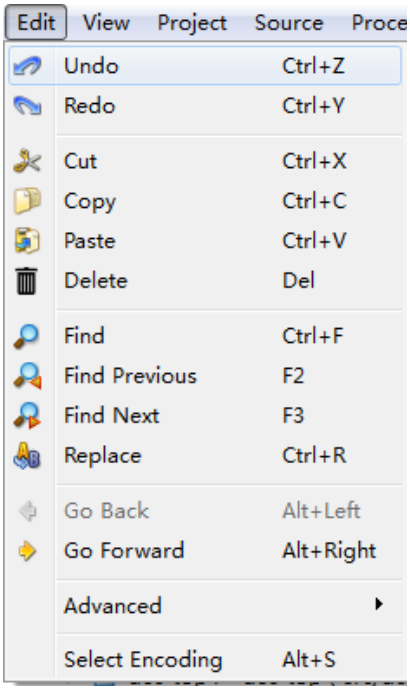
Undo，**Redo** Can be undone and redone at the time of editing;

Cut，**Copy**，**Paste**，**Delete** Same as regular cut, copy, paste, and delete functions;

Find function, Find Previous, Find Next, Replace function;

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

Select Encoding encodes characters

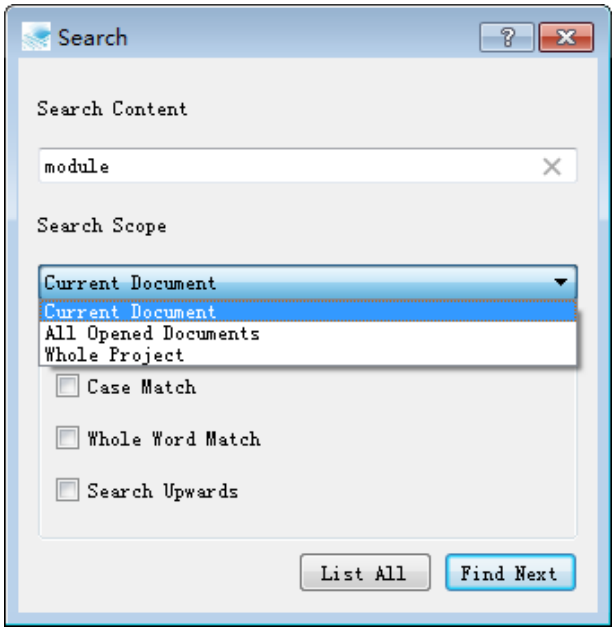


The following focuses on finding replacement features and the features involved in Advanced:

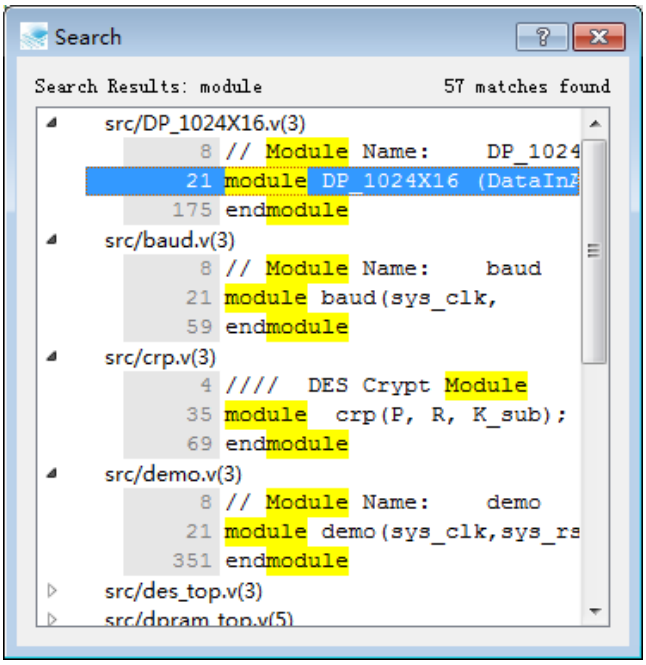
1. Search function

   Enter the function via **Edit → Find**, or the shortcut **Ctrl + F**. The following selection box will appear:

   Enter the characters you want to find and 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, or up and down search.

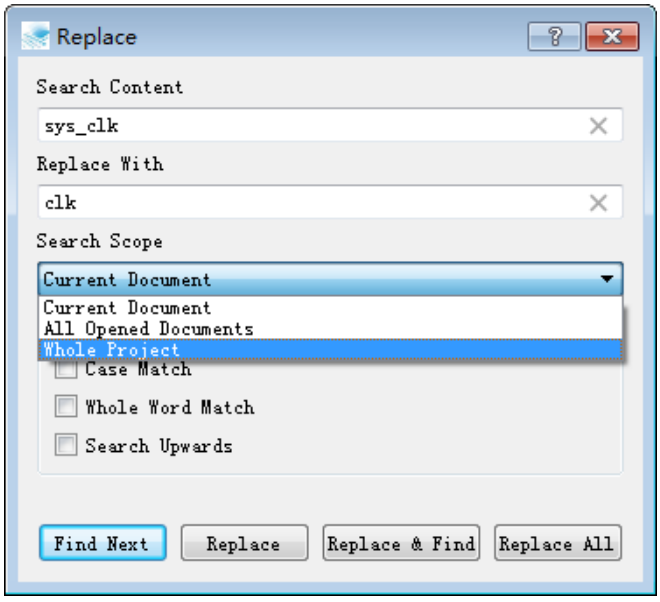


When you click List All, all relevant characters found in the search range will be listed, and you can double-click to jump to the location of the source file where the character is located.



2. Replacement function

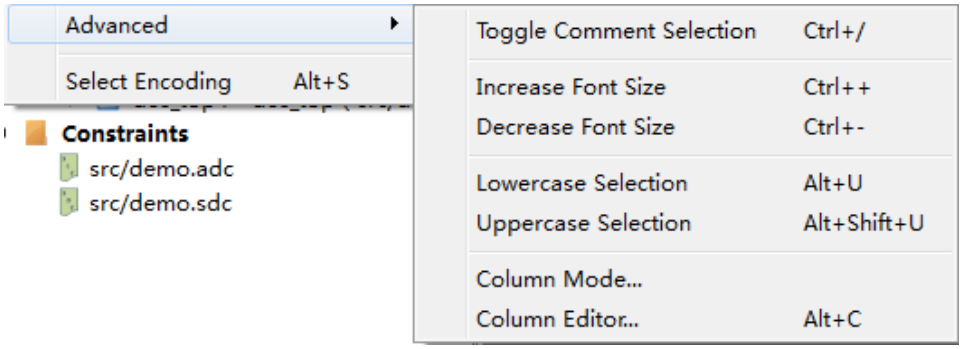
Enter the function via **Edit → Find**, or the shortcut **Ctrl + R**. The following selection box will appear:



Enter the characters you want to find and enter the replacement content. You can also select the search range and matching method. For example, if you select the search scope as “**Whole Project**” and click “**Replace All**”, all the sys_clk in the whole project will be Replace with clk.

3. **Advanced** Features

展开 **Edit** ⇒ **Advanced**, You can see the following features:



Toggle Comment Select 对选中的代码进行注释，如果选中的为已经注释的代码，则会解除注释；

Increase Font Size 放大字体；

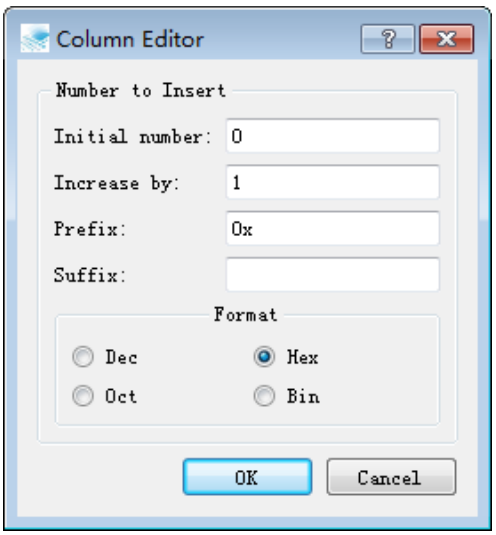
Decrease Font Size 缩小字体；

Lowercase Selection 转换选中的字符为小写字符；

Uppercase Selection 转换选中的字符为大写字符；

Column Mode…列操作模式；

Column Editor…列编辑器，如下所示，可在列操作模式下，进行递增，并可选择输入数据的前缀或后缀。



3 IP Builder

The IP Generator is a graphical interactive design interface for creating IP cores. Users can Configure the selected IP in the IP Generator and automatically generate the corresponding IP module. Currently supported IP modules are COMMON, PLL, DSP, RAM, FIFO, DRAM, SDRAM, MCU, ADC. (The ELF series devices only support DRAM modules.)

3.1 COMMON Module

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

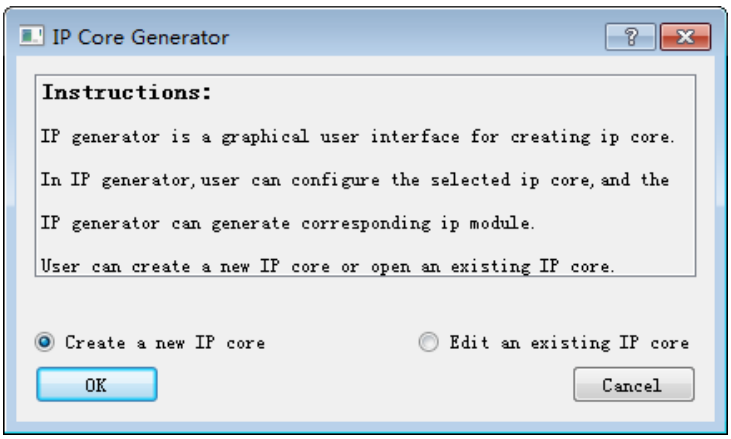
3.1.1 BUFG Module

The global clock module reduces the delay and offset of the global clock signal.

Note: The usage conditions of the BUFG module are limited. It cannot be added after the GCLK IO and the output port of the PLL. In most cases, the TD software will automatically add the BUFG module to the clock signal in a timely manner. It is recommended that the module be instantiated only if the software is not added.

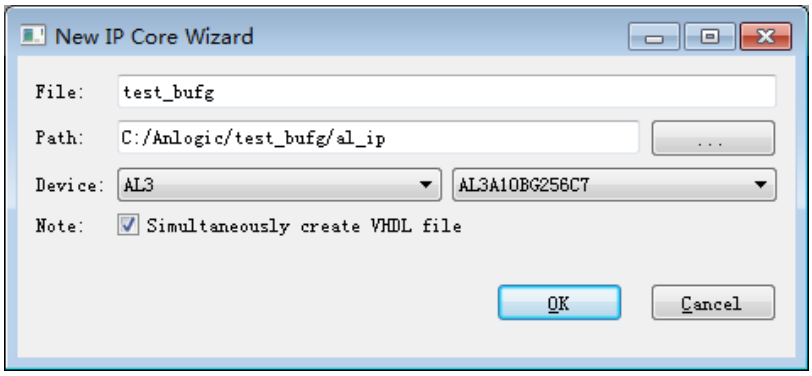
1. Create a BUFG module

选择 **Tools → IP Generator**，选择“ **Create a new IP core**”

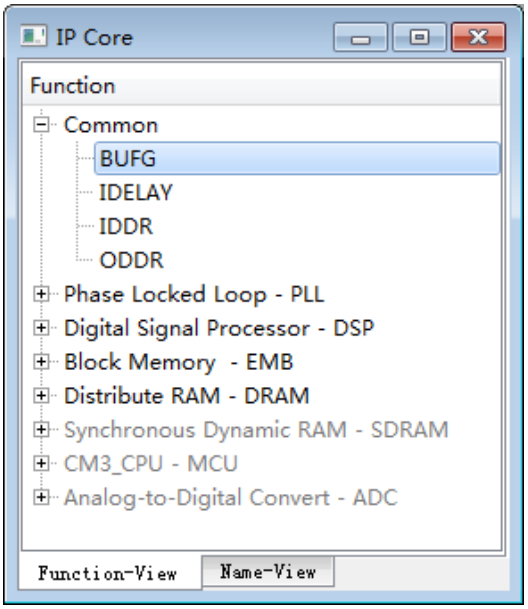


Enter the module name and select the storage path. Here, if the BUFG module is created on an engineering basis, the storage path and device name will be consistent with the project. If you create a BUFG module without engineering, you will need to manually set the save path and device name.

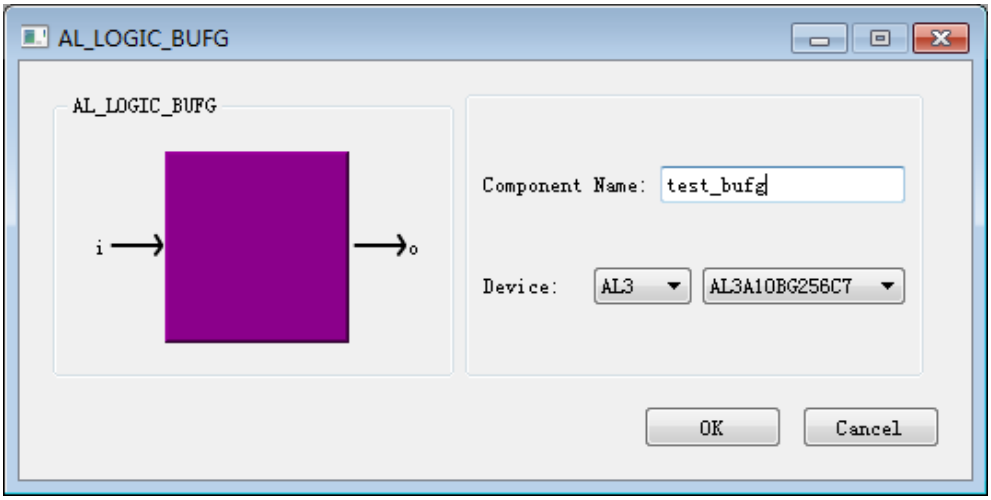
If you select “Simultaneously create VHDL file”, the TD will generate the corresponding VHDL file.



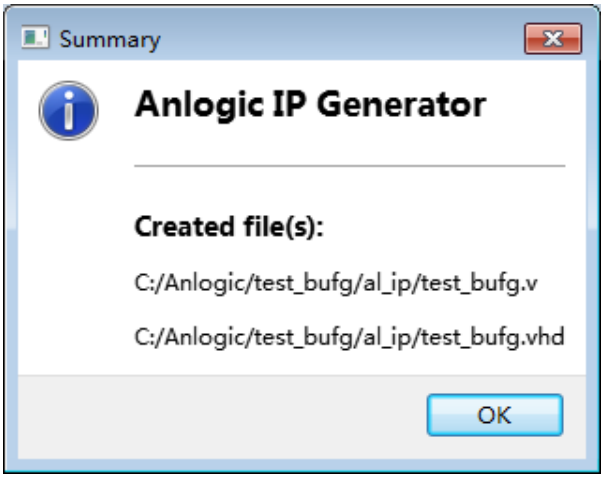
Expand the Common module in the Function window, double-click BUFG to open the configuration interface.



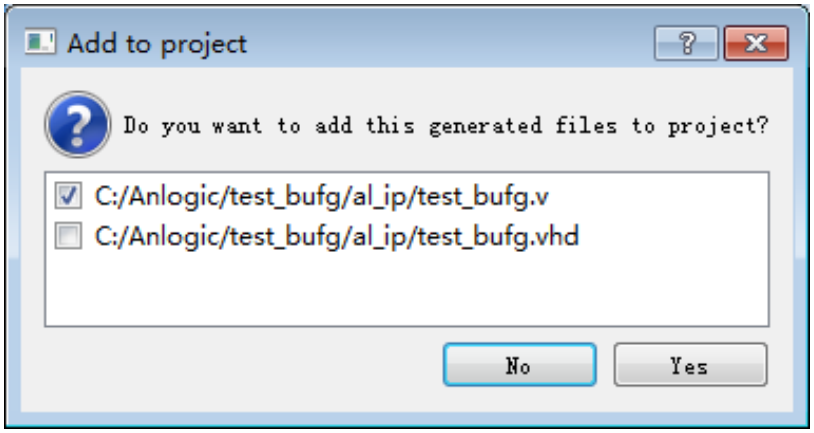
Enter the module name, select the corresponding device, the default is the engineering device



Click "OK" to complete the setup and generate the file as follows:



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



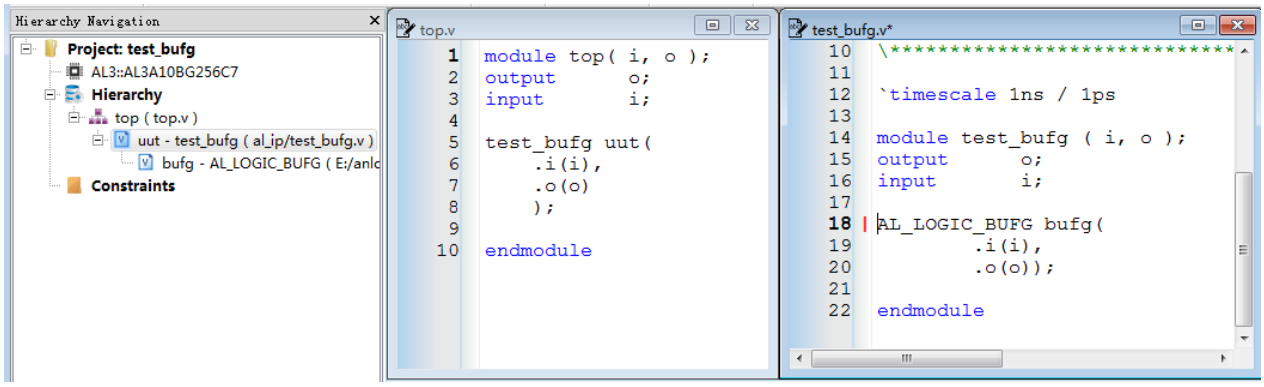
2. Instantiate the BUFG module

Take the new project as an example to introduce the process of instantiating the BUFG module. The process of instantiating the user based on the existing project is consistent.

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 to complete the instantiation of the **BUFG** module. Click **File → Save** to save the file.

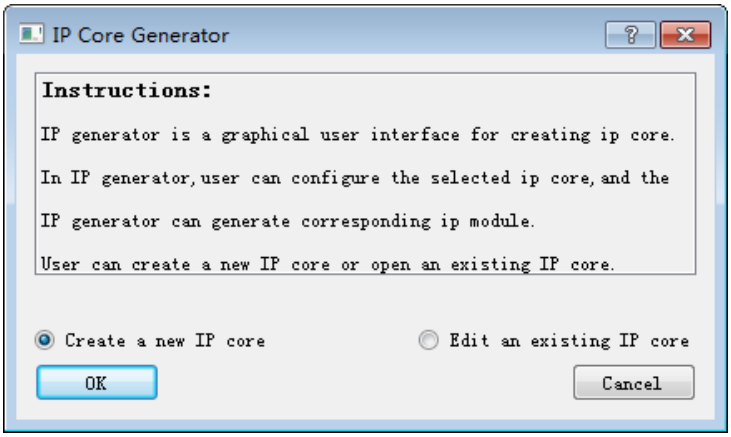


3.1.2 IDDR Module

The input dual edge sampling block is a dedicated input register that can be used to sample the dual edges of the input signal.

1. Create an IDDR module

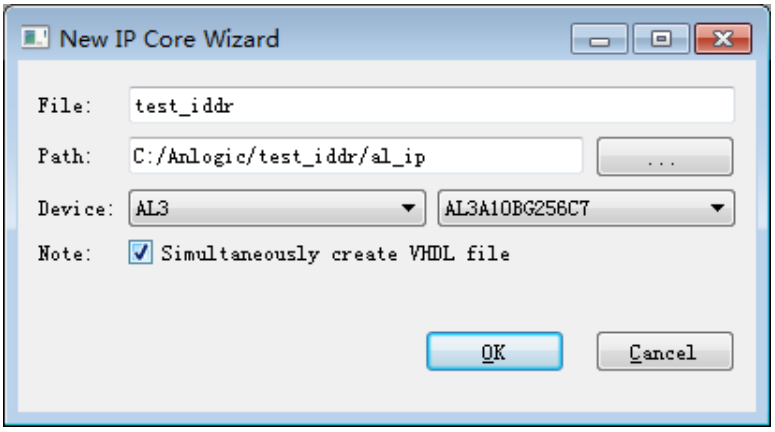
选择 **Tools → IP Generator**，选择“ **Create a new IP core**”



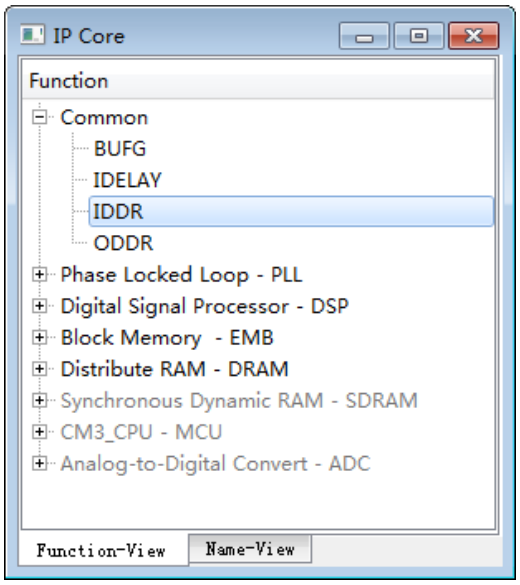
Enter the module name and select the storage path. Here, if an IDDR module is created on an engineering basis, the storage path and device name will be consistent with the project.

If the IDDR module is created without engineering, the user must manually set the save path and device name.

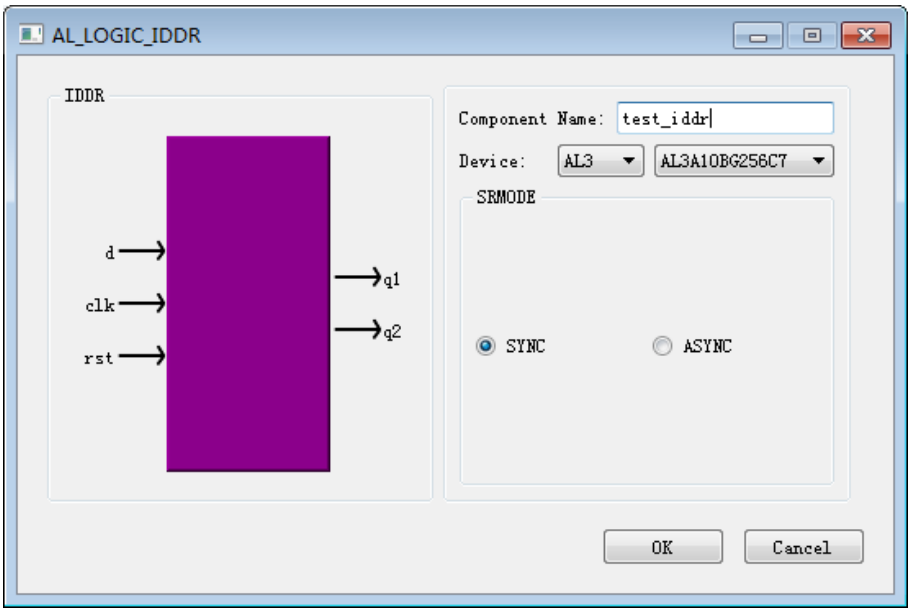
If you select “Simultaneously create VHDL file”, the TD will generate the corresponding VHDL file.



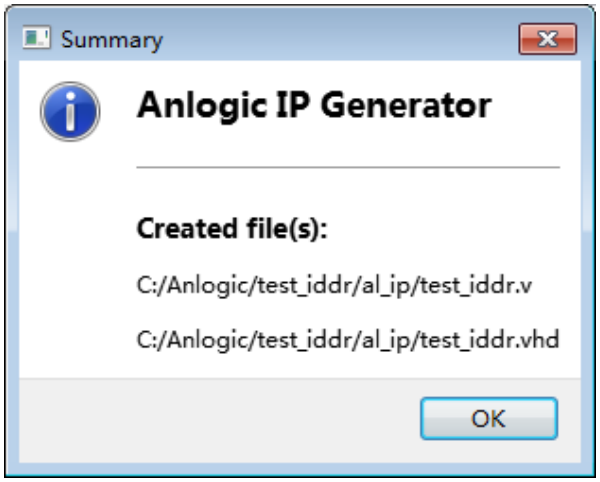
Expand the Common module in the Function window and double-click IDDR to open the configuration interface.



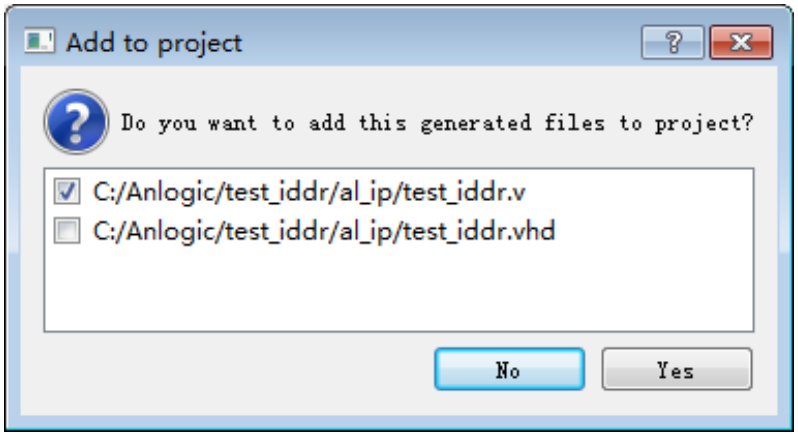
Enter the module name, select the corresponding device, the default is the engineering device



Click "OK" to complete the setup and generate the file as follows:



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



2. Instantiate the IDDR module

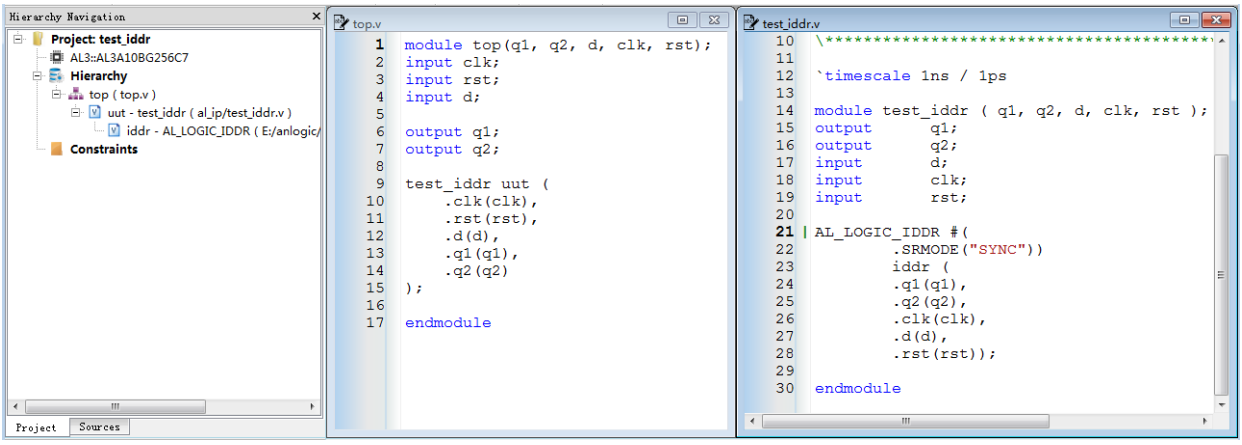
Take the new project as an example to introduce the process of instantiating the IDDR module. Users can also instantiate on the basis of existing projects, and the instantiation process is consistent.

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 to complete the instantiation of the IDDR module.

Click File → Save to save the file.

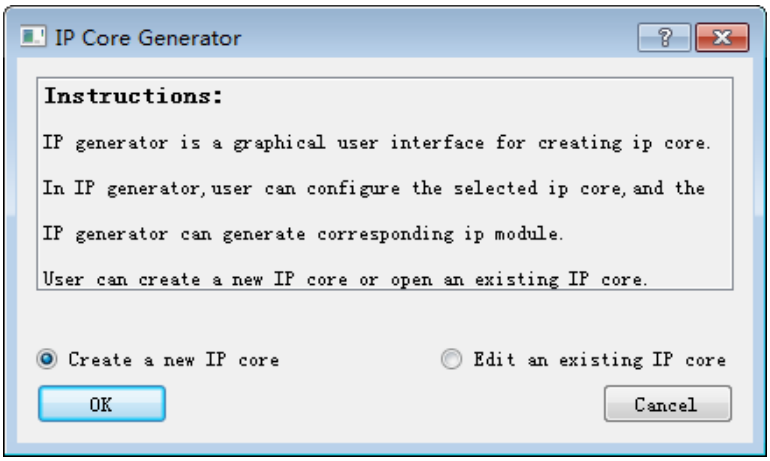


3.1.3 ODDR Module

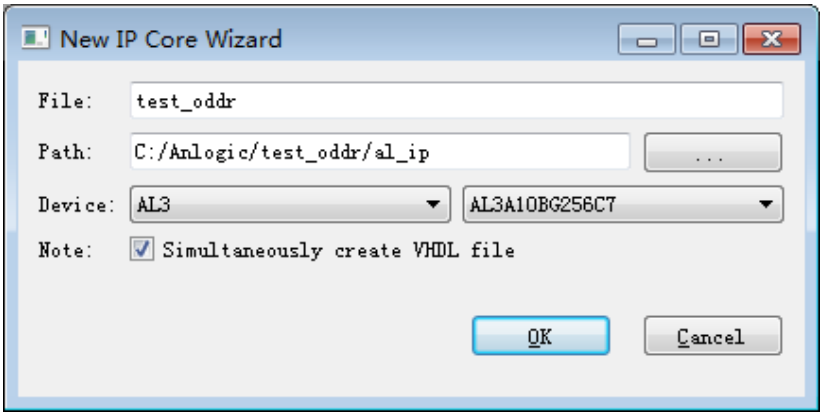
Output dual edge drive module for dual edge drive of the output signal.

1. Create an ODDR module

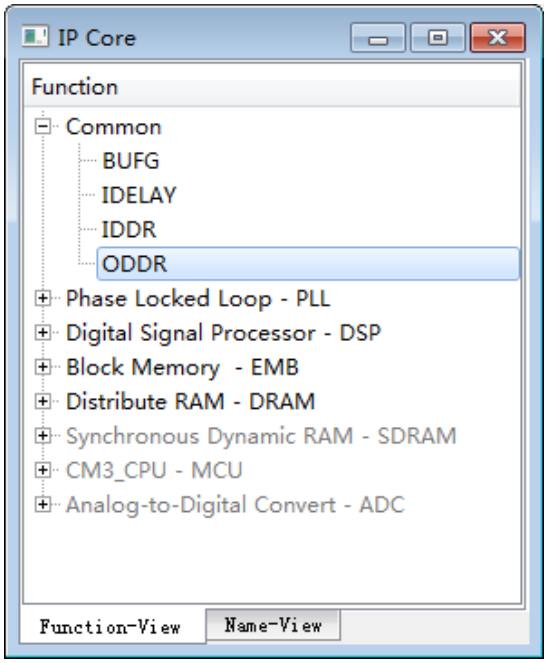
选择 Tools → IP Generator，选择“ Create a new IP core”



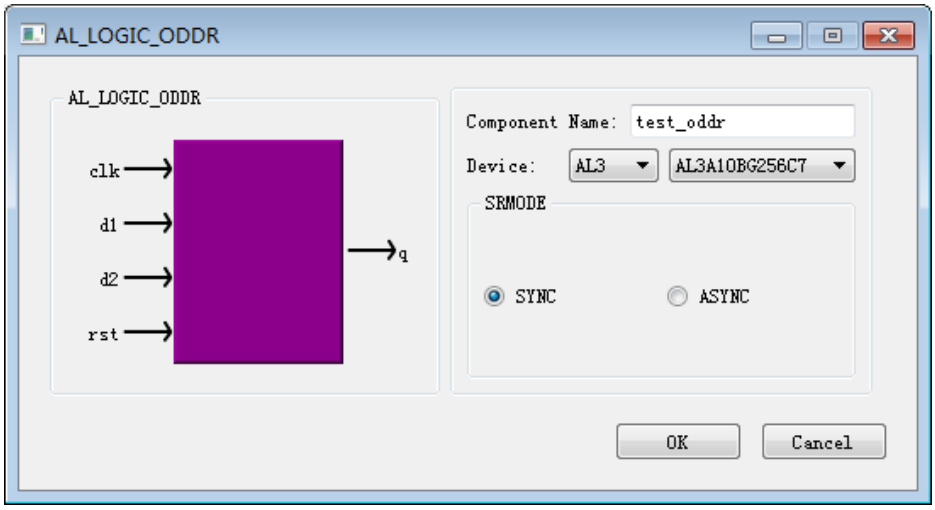
Enter the module name and select the storage path. Here, if an ODDR module is created on an engineering basis, the storage path and device name will be consistent with the project.
If the ODDR module is created without engineering, the user must manually set the save path and device name. If "Simultaneously create VHDL file" is checked, the TD will generate the corresponding VHDL file.



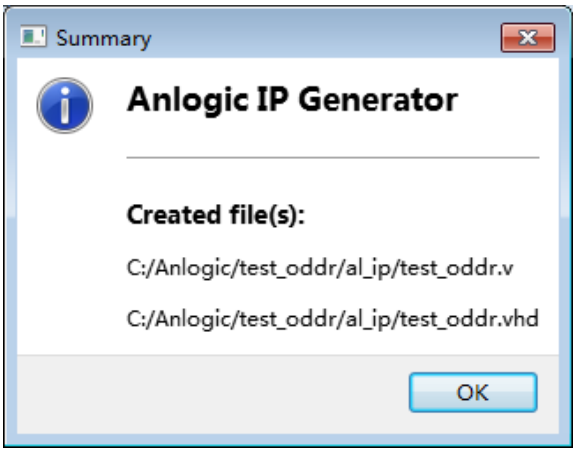
Expand the Common module in the Function window and double-click ODDR to open the configuration interface.



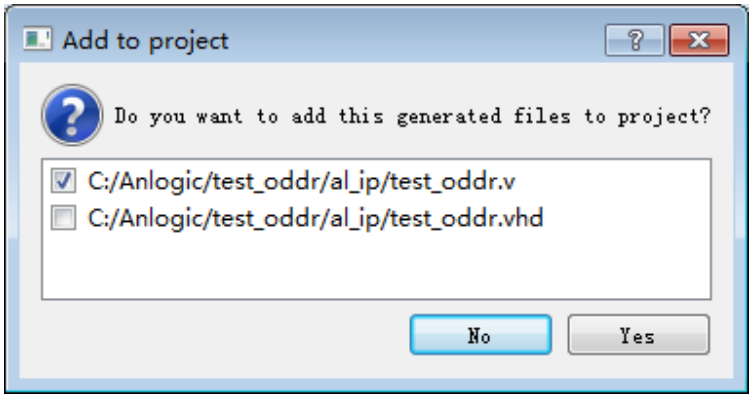
Enter the module name, select the corresponding device, the default is the engineering device



Click "OK" to complete the setup and generate the file as follows:



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



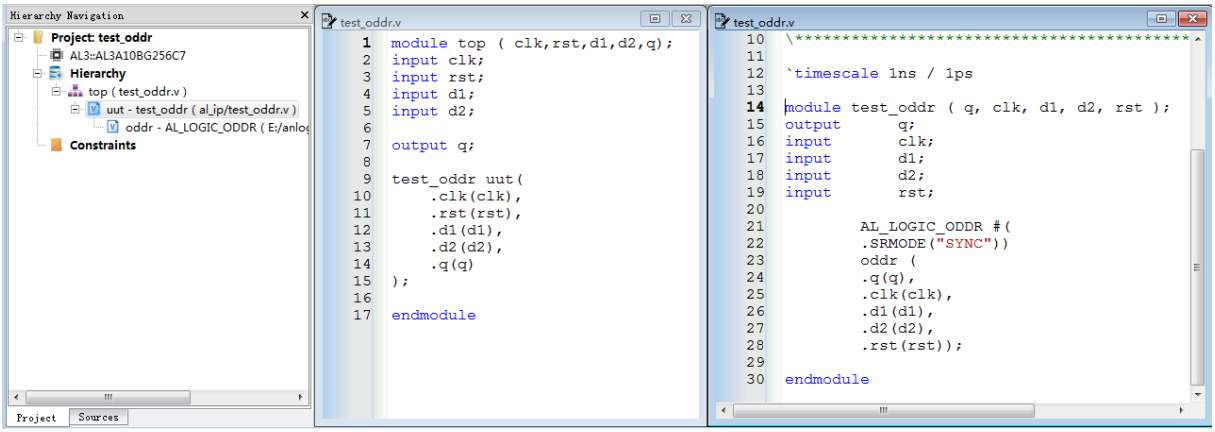
2. Instantiate the ODDR module

Take the new project as an example to introduce the process of instantiating the ODDR module. Users can also instantiate on the basis of existing projects, 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 to complete the instantiation of the ODDR module. Click File → Save to save the file.



3.2 PLL Module

This manual describes the PLL module in the EAGLE series. EAGLE series FPGAs have up to four multi-function phase-locked loops (PLL0~PLL3) for high-performance clock management. Each PLL implements clock division/multiplier, input and feedback clock alignment, and multi-phase clock output.

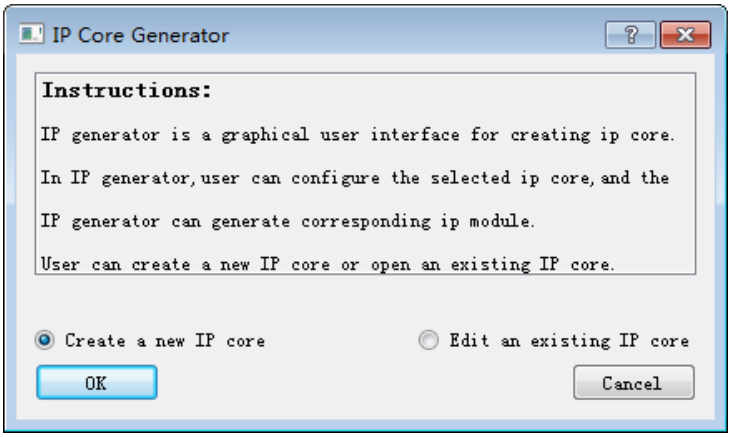
The PLL reference clock inputs are: clock network output, interconnect output, and internal oscillator output.

The 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.

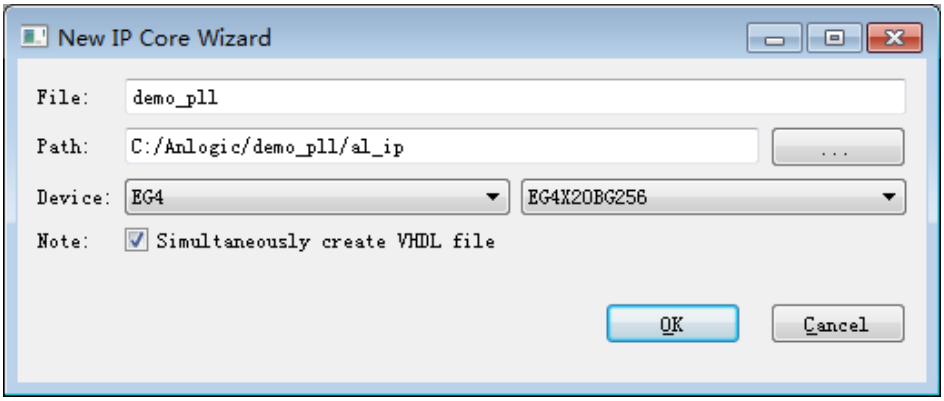
3.2.1 Create a PLL module

1. 选择 **Tools → IP Generator**，选择“ **Create a new IP core**”

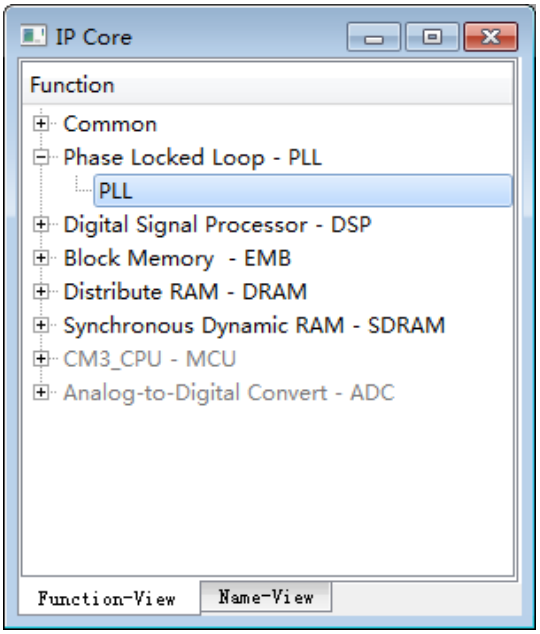


2. Enter the module name and select the storage path. Here, if the PLL module is created on an engineering basis, the storage path and device name will be consistent with the project. If the PLL module is created without engineering, the user must manually set the save path and device name.

   If "Simultaneously create VHDL file" is checked, the TD will generate the corresponding VHDL file.

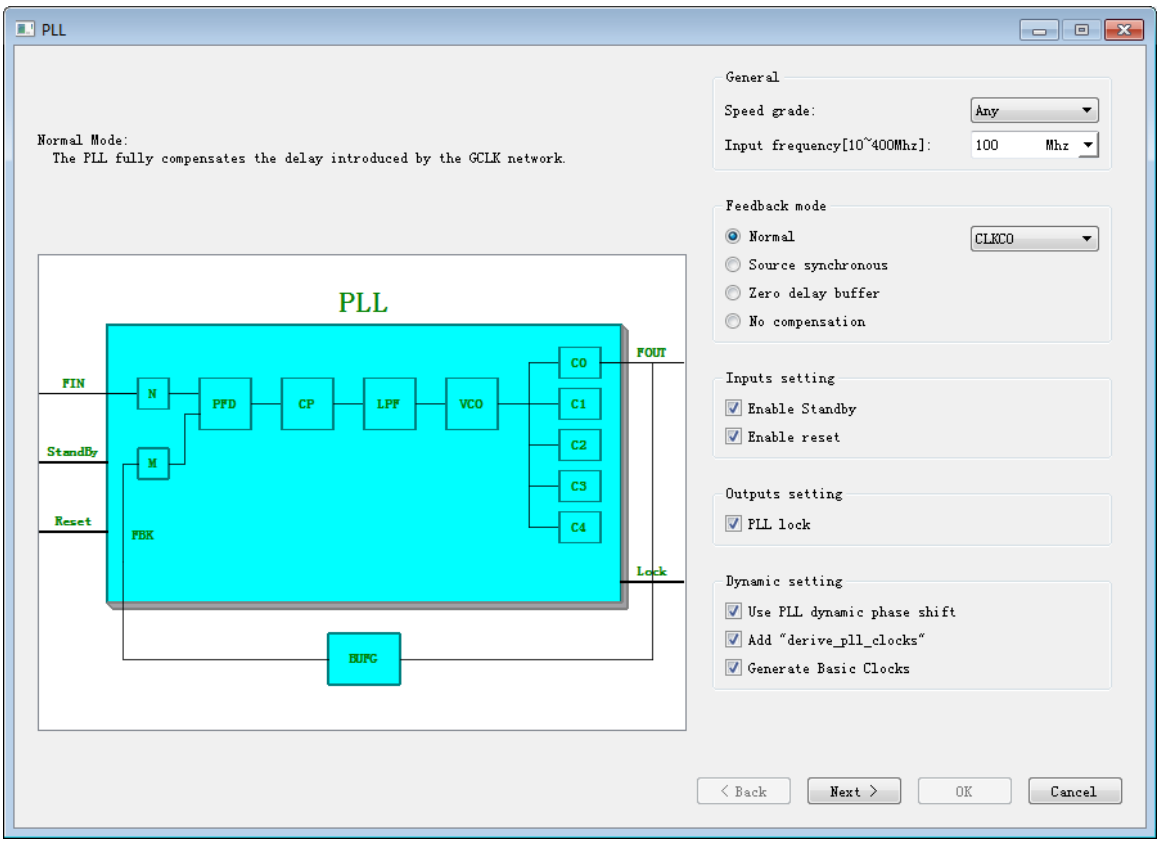


3. Expand Phase Locked Loop in the Function window and double-click PLL to open the configuration interface.



4. Set the relevant parameters of the PLL

1) PLL mode setting



The PLL supports four feedback modes, each of which supports clock division/multiplier and phase shift.

a) Normal mode (Normal)

In normal mode, the PLL compensates for the GCLK network delay, ensuring that the internal register input clock phase and clock pin phase are the same.

b) **Source Synchronous Mode (Source-Synchronous)**

Source Synchronous Mode The dynamic phase shift function adjusts the clock phase to ensure that the delay from the data port to the IOB input register is equal to the delay from the clock input port to the IOB register (data and clock input port modes are the same).

c) **No compensation mode (No Compensation)**

In uncompensated mode, the PLL does not compensate for clock network delay, and

the PLL uses internal self-feedback, which improves the jitter characteristics of the PLL.

d) **Zero delay buffer mode (Zero Delay Buffer)**

Zero delay buffer mode, clock output pin phase and PLL reference clock input pin phase alignment.

The PLL parameter characteristics are shown in the following table:

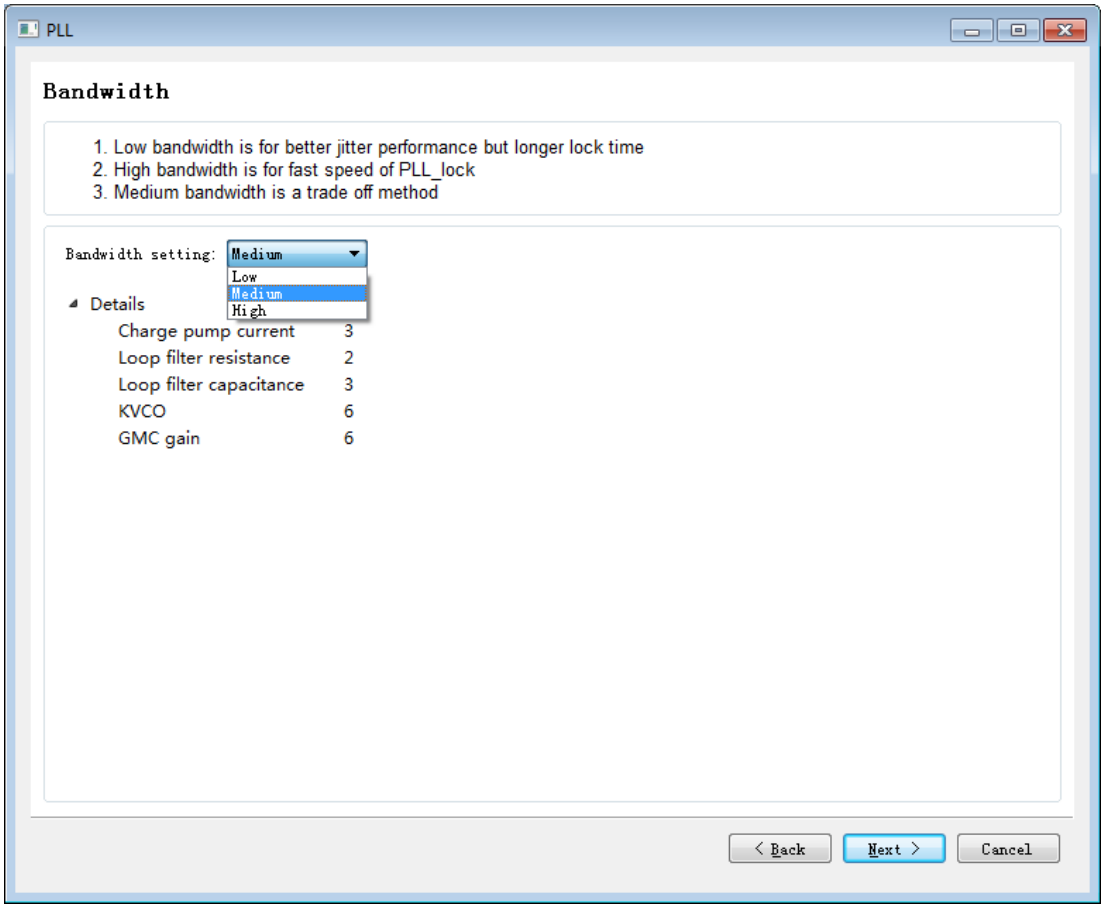
Parameter	**Feature**
Input clock frequency range	10~400 MHz
Output clock frequency range	4~400 MHz
VCO frequency range	300~1200 MHz
Number of output ports	5 (each port is independently independent)
Reference clock division factor (M)	1~128
Feedback clock division factor (N)	1~128
Output clock division factor (C0~C4)	1~128
Phase shift resolution	45°
Output port selectable phase offset	0,45,90,135,180,225,170,315 (degrees)
User dynamic phase shift control	Support (+/- 45 degree phase shift per unit)
Locked state output	Lock
Dedicated clock output pin	stand by

When "Add derive_pll_clocks" is selected, the clock constraint is automatically generated on all used PLL clkc[x] ports when compiling the project. The frequency and phase of the generated clock will be set strictly according to the parameters inside the PLL.

Selecting "Generate Basic Clocks" will define the reference clock of the FIN frequency on the corresponding PLL refclk, otherwise it will automatically search for the refclk pin and the clock defined on the connected net, and report an error if it is not found.

2) Bandwidth settings

The values of Bandwidth can be set to Low, Medium, and High respectively. The default value is Medium. Click on "Show Details" to see the values of the PLL performance parameters for this bandwidth.

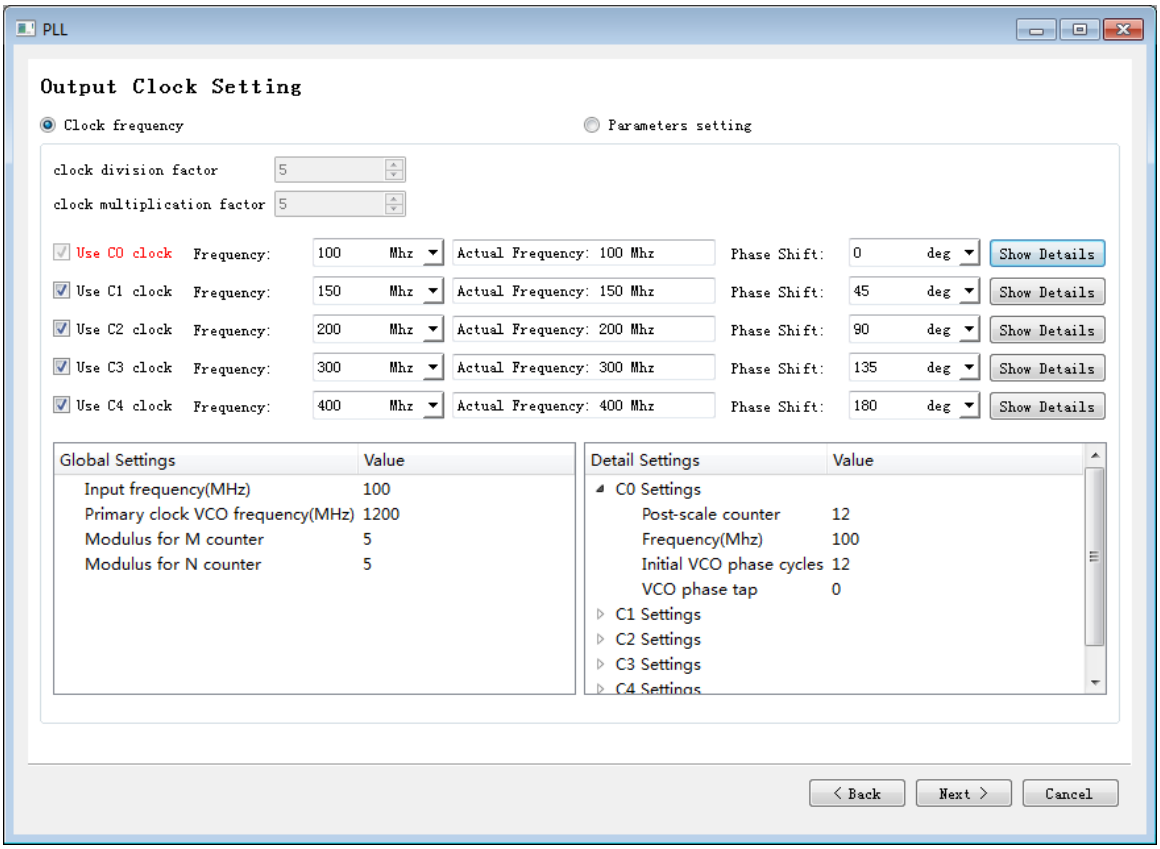


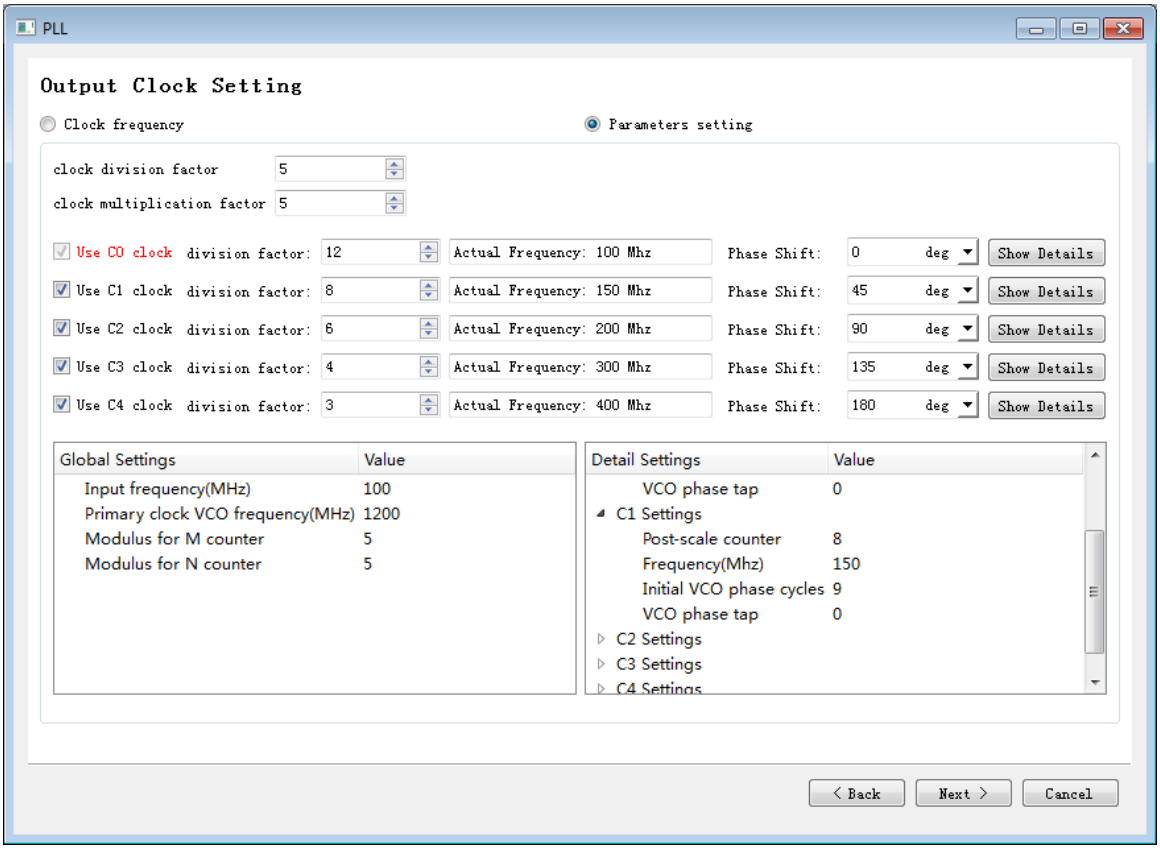
3) Output clock setting

Each PLL has five output clocks C0~C4, which can be selected according to the number of output clocks and configure the frequency and phase offset of the output clock.

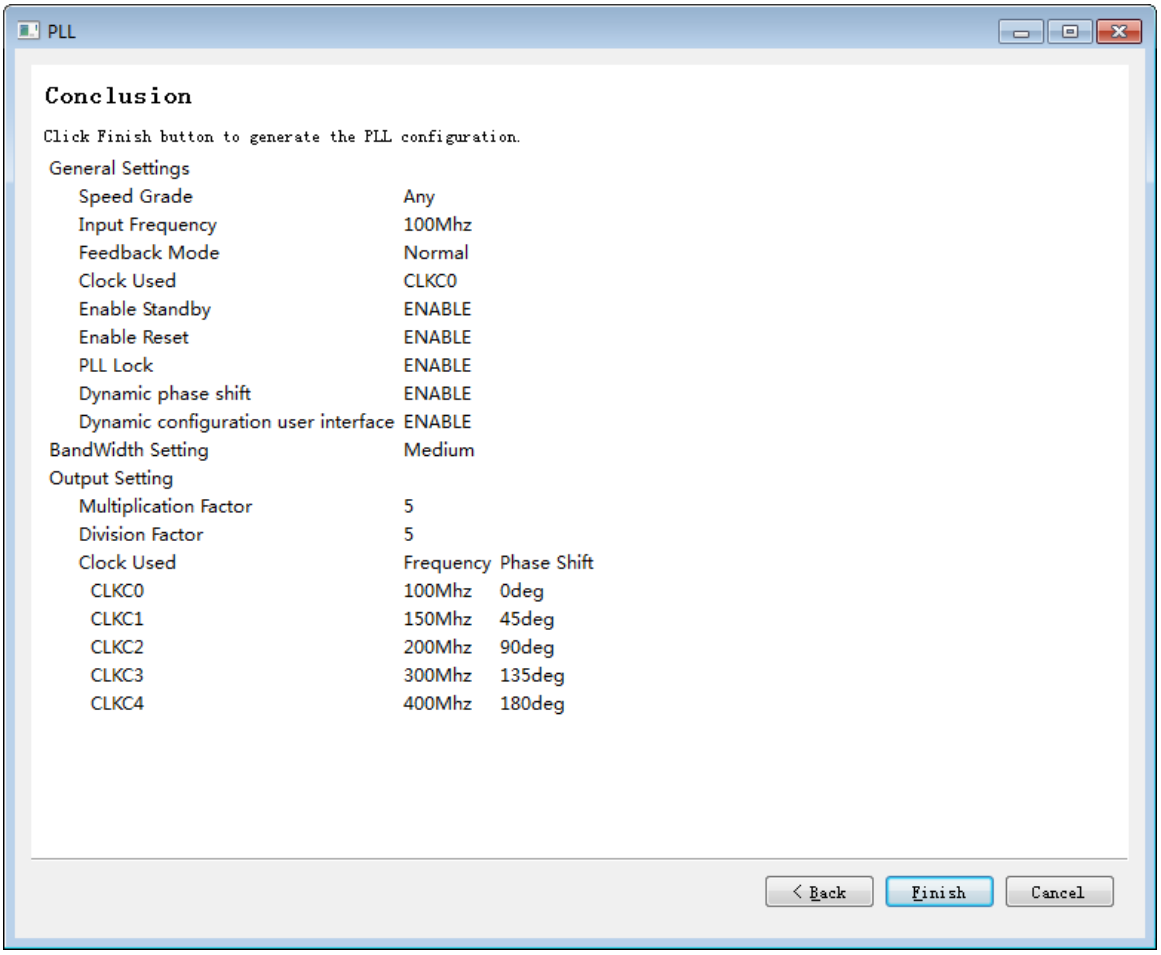
When setting the output frequency, you can set the output frequency directly on the Clock frequency interface, or set the division factor on the Parameters setting interface according to the input frequency.

Click "Show Details" to view the performance parameter values of the output in the lower right corner.

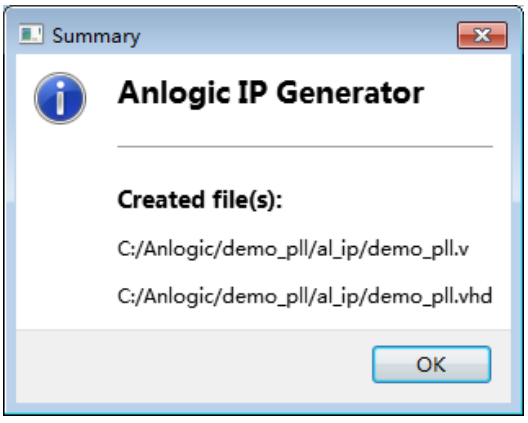

PLL
Output Clock Setting
Clock frequency
Parameters setting
clock division factor 5
clock multiplication factor 5
Use C0 clock Frequency: 100 Mhz Actual Frequency: 100 Mhz Phase Shift: 0 deg Show Details
Use C1 clock Frequency: 150 Mhz Actual Frequency: 150 Mhz Phase Shift: 45 deg Show Details
Use C2 clock Frequency: 200 Mhz Actual Frequency: 200 Mhz Phase Shift: 90 deg Show Details
Use C3 clock Frequency: 300 Mhz Actual Frequency: 300 Mhz Phase Shift: 135 deg Show Details
Use C4 clock Frequency: 400 Mhz Actual Frequency: 400 Mhz Phase Shift: 180 deg Show Details
Global Settings Value
Input frequency(MHz) 100
Primary clock VCO frequency(MHz) 1200
Modulus for M counter 5
Modulus for N counter 5
Detail Settings Value
C0 Settings
Post-scale counter 12
Frequency(Mhz) 100
Initial VCO phase cycles 12
VCO phase tap 0
C1 Settings
C2 Settings
C3 Settings
C4 Settings
< Back
Next >
Cancel


PLL
Output Clock Setting
Clock frequency
Parameters setting
clock division factor 5
clock multiplication factor 5
Use C0 clock division factor: 12 Actual Frequency: 100 Mhz Phase Shift: 0 deg Show Details
Use C1 clock division factor: 8 Actual Frequency: 150 Mhz Phase Shift: 45 deg Show Details
Use C2 clock division factor: 6 Actual Frequency: 200 Mhz Phase Shift: 90 deg Show Details
Use C3 clock division factor: 4 Actual Frequency: 300 Mhz Phase Shift: 135 deg Show Details
Use C4 clock division factor: 3 Actual Frequency: 400 Mhz Phase Shift: 180 deg Show Details
Global Settings Value
Input frequency(MHz) 100
Primary clock VCO frequency(MHz) 1200
Modulus for M counter 5
Modulus for N counter 5
Detail Settings Value
VCO phase tap 0
C1 Settings
Post-scale counter 8
Frequency(Mhz) 150
Initial VCO phase cycles 9
VCO phase tap 0
C2 Settings
C3 Settings
C4 Settings
< Back
Next >
Cancel

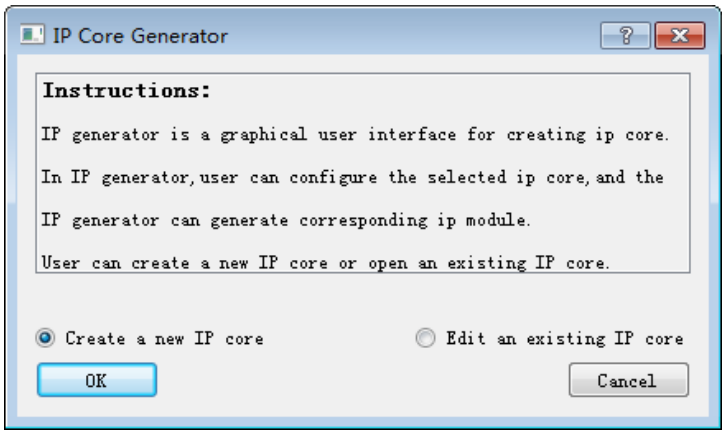
4) Finally, confirm that the parameters are correct. Click “Finish” to complete the configuration of the PLL.

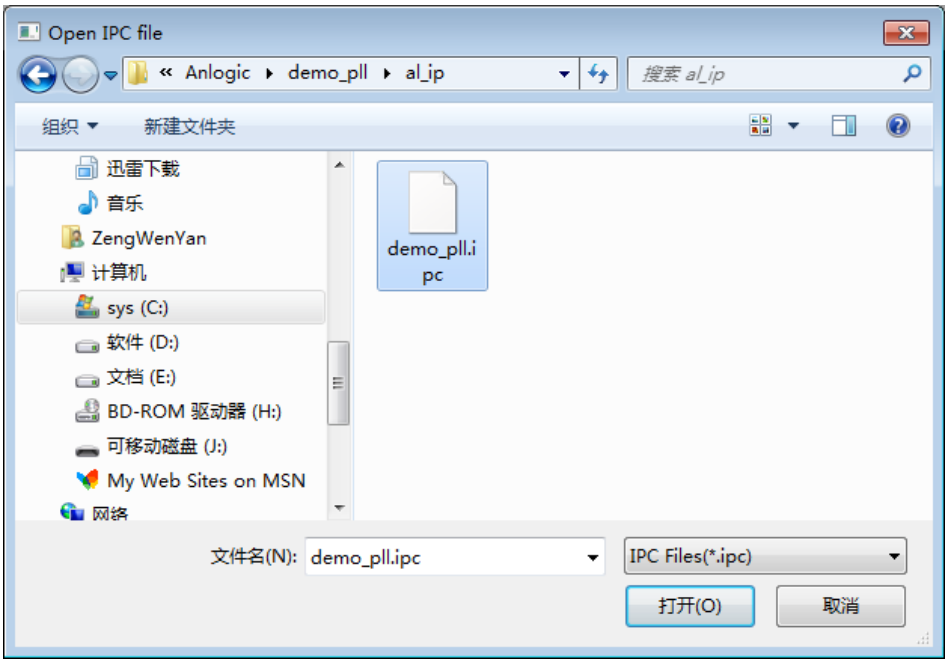


5. The TD will give the path to the generated file. After clicking “OK”, you can choose whether to add the generated file to the project according to the prompt.



6. You can open an existing IP by selecting Tools → IP Generator and selecting “Edit an existing IP core”.


IP Core Generator
Instructions:
IP generator is a graphical user interface for creating ip core.
In IP generator, user can configure the selected ip core, and the
IP generator can generate corresponding ip module.
User can create a new IP core or open an existing IP core.
Create a new IP core
Edit an existing IP core
OK
Cancel


Open IPC file
« Anlogic ▸ demo_pll ▸ al_ip
搜索 al_ip
组织
新建文件夹
迅雷下载
音乐
ZengWenYan
计算机
sys (C:)
软件 (D:)
文档 (E:)
BD-ROM 驱动器 (H:)
可移动磁盘 (J:)
My Web Sites on MSN
网络
demo_pll.ipc
文件名(N): demo_pll.ipc
IPC Files(*.ipc)
打开(O)
取消

3.2.2 Instantiate the PLL module

Take the new project as an example to introduce the process of instantiating the PLL module. Users can also instantiate on the basis of existing projects, and the instantiation process is consistent.

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, modify the inst name and port name, and click the Save button to complete the instantiation of the PLL module. Click File → 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*

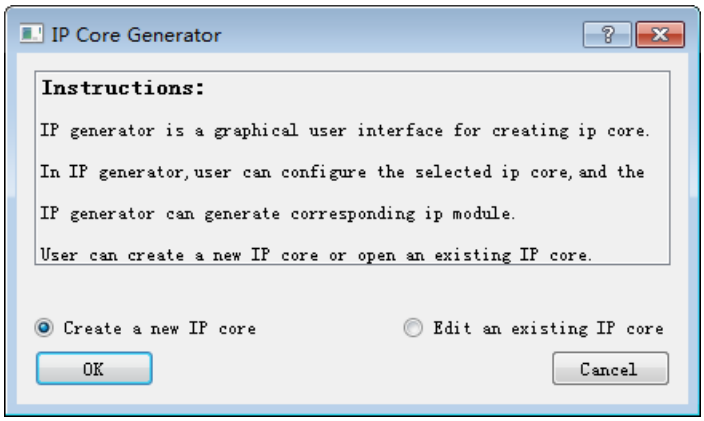
```
module demo_pll(refclk,reset,stdby,extlock,
            clk0_out,clk1_out,clk2_out,clk3_out);
    input refclk;
    input reset;
    input stdby;
    output extlock;
    output clk0_out;
    output clk1_out;
    output clk2_out;
    output clk3_out;

    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
```

3.3 DSP Module

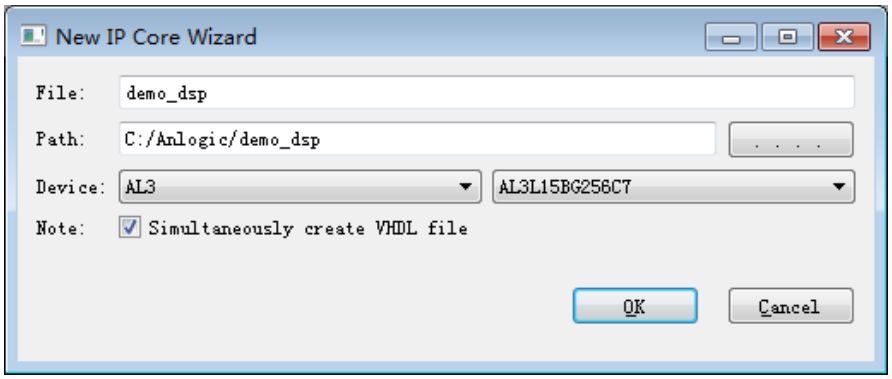
In the AL3 family of devices, the embedded multiplier can be configured as an 18×18 multiplier with input and output registers or as two 9×9 multipliers.

3.3.1 Create a DSP module

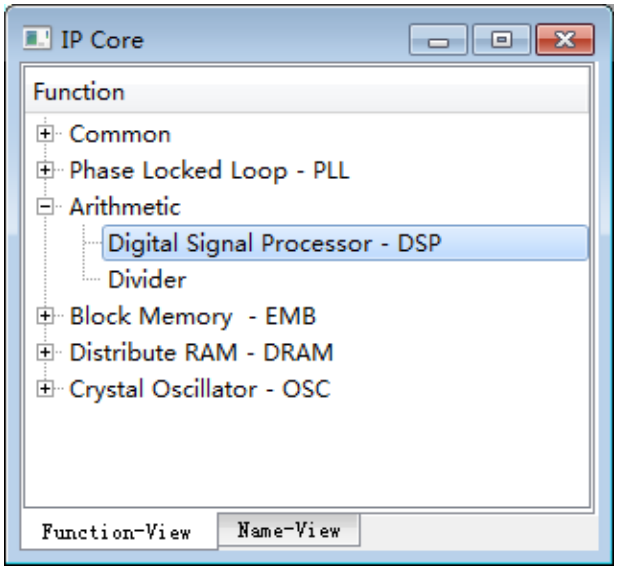
1. 选择 **Tools ⇒ IP Generator**，选择“ **Create a new IP core**”



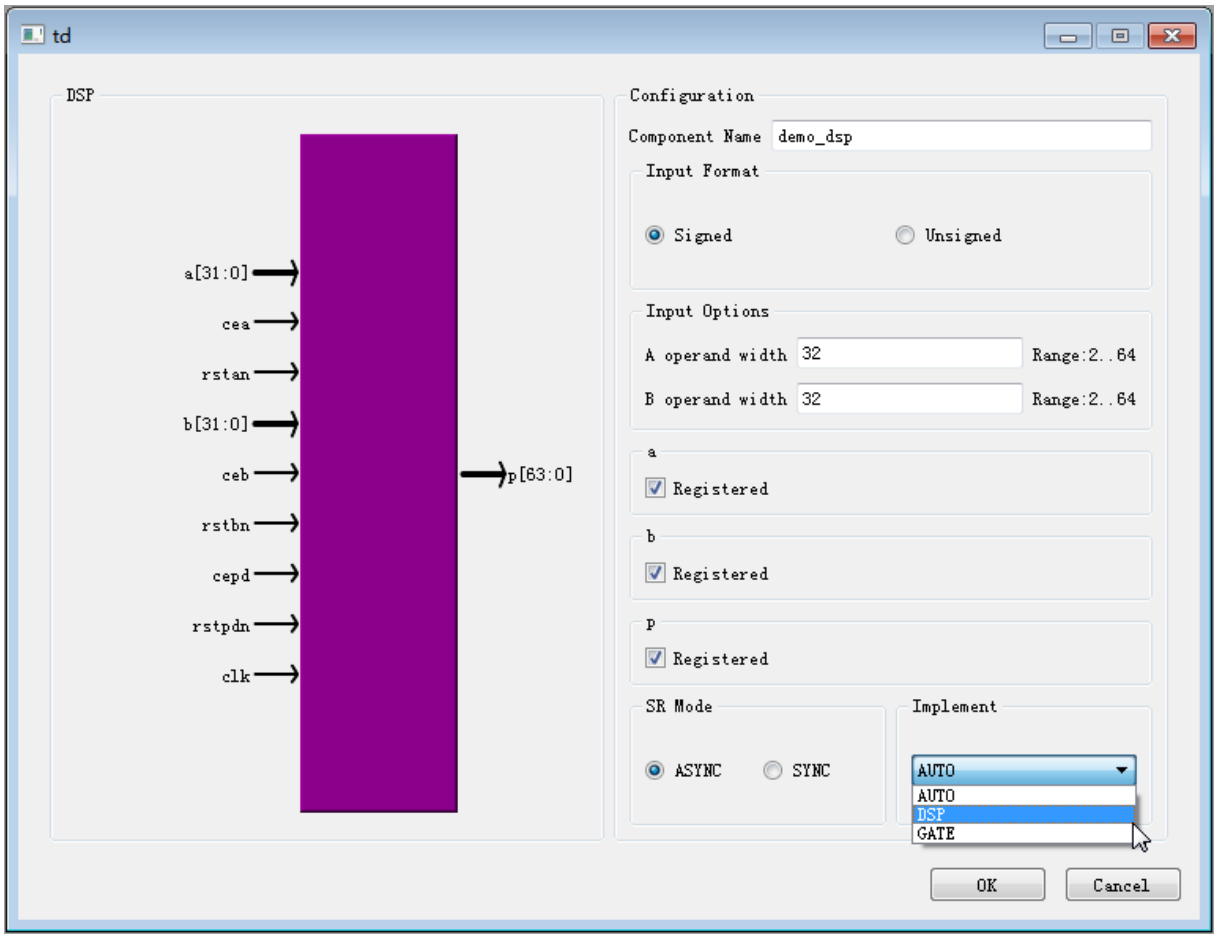
2. Enter the module name and select the storage path. Here, if you create a DSP module on an engineering basis, the storage path and device name will be consistent with the project. If you create a DSP module without engineering, you need to manually set the save path and device name. If "Simultaneously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



3. Expand Arithmetic→Digital Signal Processor in the Function window, double-click DSP to open the configuration interface.



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



Users in the IP Generator can customize the implementation of multiplication and provide three parameters for the user to choose.

Among them, the DSP indicates that the hardware DSP is forced to implement the multiplication operation. If the DSP is not enough, the hardware DSP realizes faster than the logic gate; the GATE indicates that only the logic gate is used to implement the multiplication operation; the AUTO indicates that the DSP is used preferentially, if the DSP Not enough, use logic gates to achieve multiplication. The default parameter is: AUTO.

The embedded multipliers of the AL3 family of devices consist of the following cells: input registers, multiplier cores, and output registers.

- Input register

  Depending on the mode of operation of the multiplier, each multiplier input signal can be connected to the input register or directly to the internal multiplier in 9bit or 18bit form. You can set whether each input of the multiplier uses the input register separately.

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

- Clock (clk)
- Clock enable (cea / ceb)
- 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, and the clock enable signal and the asynchronous clear signal driver can be independently configured.

- Multiplier core

  The multiplier of the embedded multiplier module supports both 9x9 or 18x18 multipliers and other multipliers between these configuration bit widths. Depending on the data width or mode of operation of the multiplier, a single embedded multiplier can perform one or two multiplication operations simultaneously.

The two operands of the multiplier can be declared as signed/unsigned numbers by the signed / unsigned option to determine the type of multiplier.

- Output register

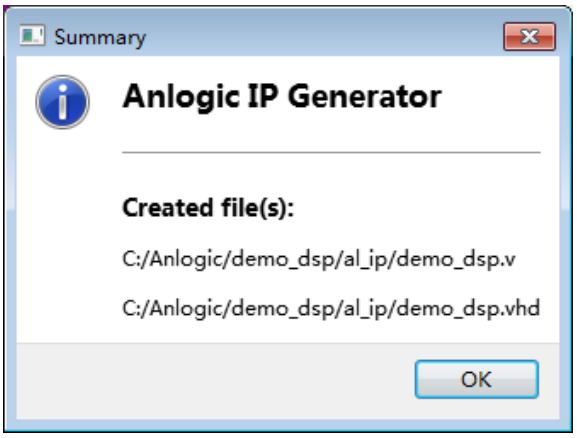
  Depending on the mode of operation of the multiplier, the output of the embedded multiplier can be registered using an output register in either 18-bit or 36-bit form.

  The following control signals are available for each of the output registers in the embedded multiplier:

  - Clock (clk)
  - Clock enable (cepd)
  - Sync/Asynchronous Clear (rstpdn : n means active low)

All input and output registers in the same embedded multiplier are driven by the same clock signal, and the clock enable signal and the asynchronous clear signal driver can be independently configured.

5. Click "OK" to complete the DSP setup and TD will give the path to the generated file.

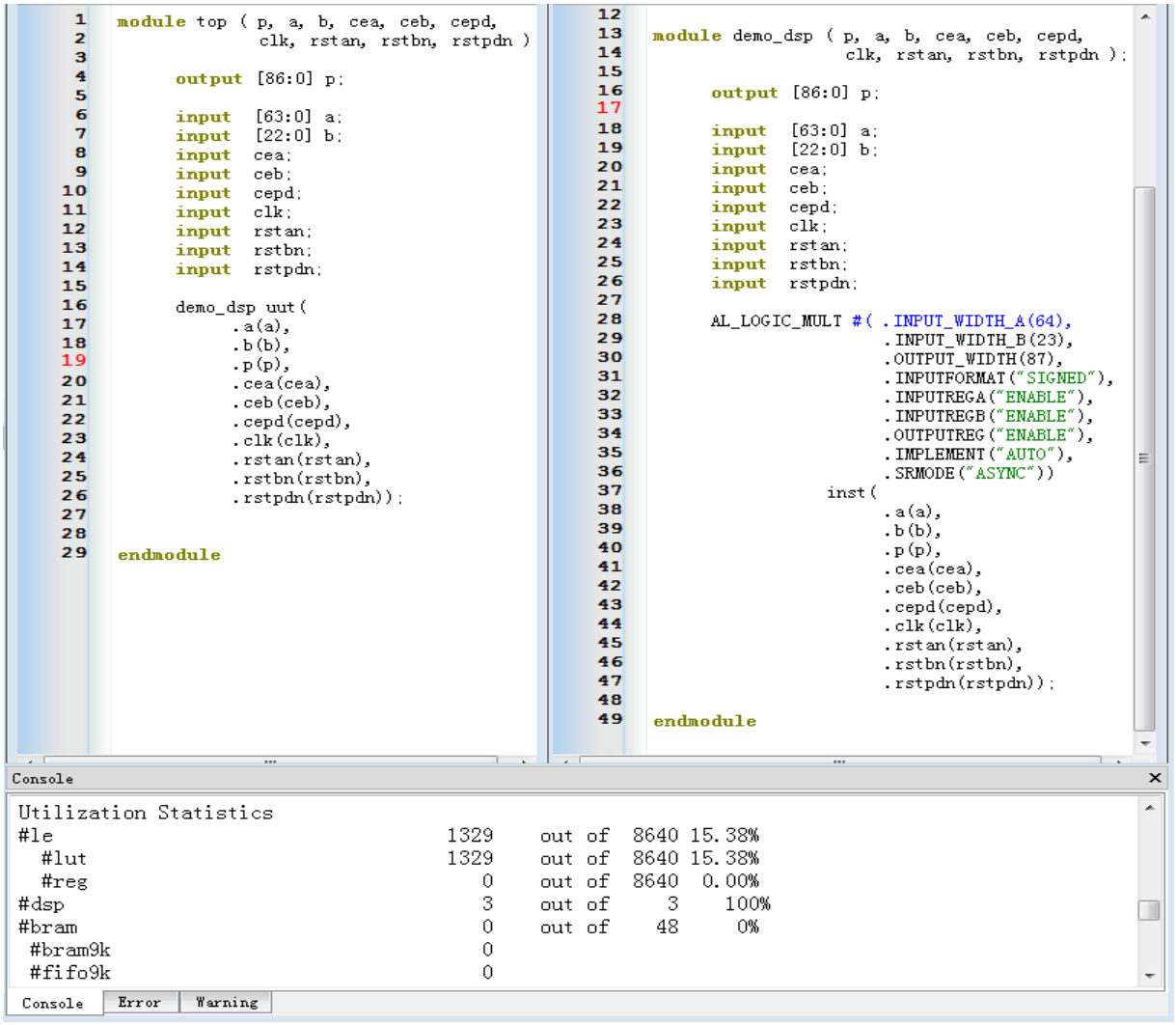


You can open an existing IP by selecting Tools → IP Generator and selecting “Edit an existing IP core”.

3.3.2 Instantiate DSP module

This manual uses the new project as an example to introduce the process of instantiating the DSP module. Users can also instantiate on the basis of existing projects, and the instantiation process is consistent.

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, modify the inst name and port name, and click the Save button to complete the instantiation of the DSP module.

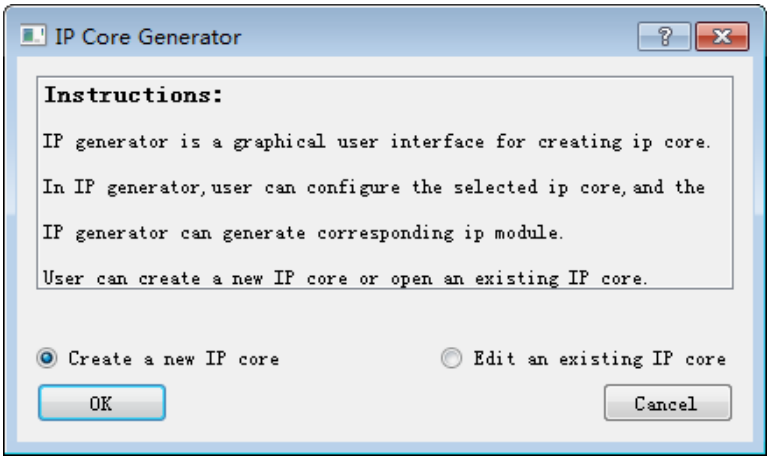


3.4 Divider Module

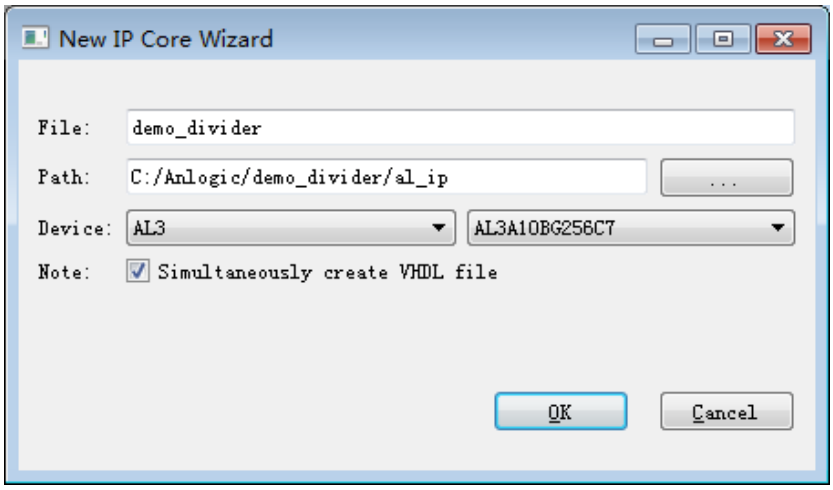
The TD software implements a clock-driven divider.

3.4.1 Create a Divider module

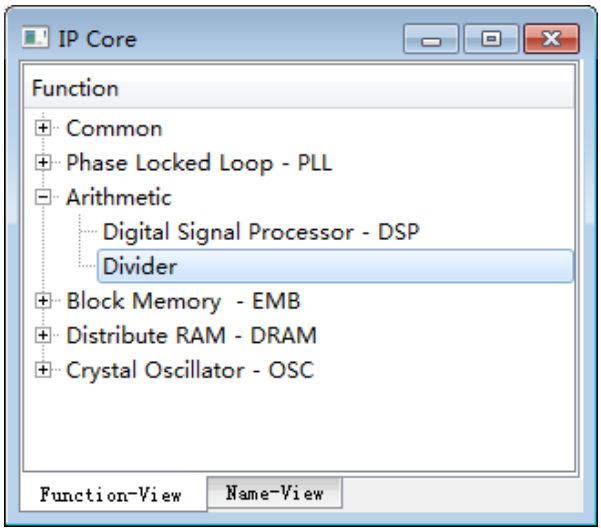
1. 选择 **Tools** ⇒ **IP Generator**，选择“ **Create a new IP core**”。



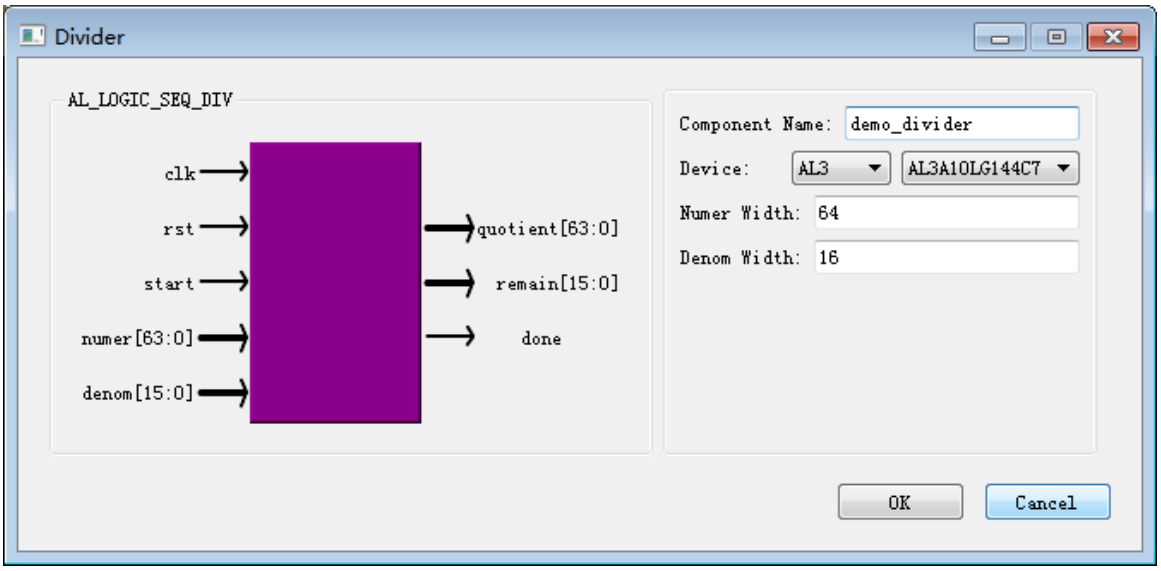
2. Enter the module name and select the storage path. Here, if you create a Divider module on an engineering basis, the storage path and device name will be consistent with the project. If you create a Divider module without engineering, you will need to manually set the save path and device name. If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



3. Expand Arithmetic→Divider in the Function window and double-click Divider to open the configuration interface.



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



among them,

1) Numer Width is the bit width of the dividend and also serves as the bit width of the quotient;

2) Denom Width is the bit width of the divisor and also serves as the bit width of the remainder;

3) Clk is the clock used for the drive calculation;

4) rst is the reset signal. When rst is 1, the internal register and output (quotient, remain) will be set to 0, and done will be set to 1;

5) Start is the calculated start signal. When start is set to 1, the input data is buffered into the internal register;

when start is changed from 1 to 0, the calculation process actually begins;

6) numer is the dividend. Although start is 0, changing the value of numer does not affect the calculation (the original value has been buffered to the internal register on the rising edge of clock with start 1). However, in order to prevent misunderstandings during waveform display, the done calculation will be provided after done is 1 (or wait for the number of clock cycles required by the calculation process);

7) denom is the divisor. Note the same as numer;

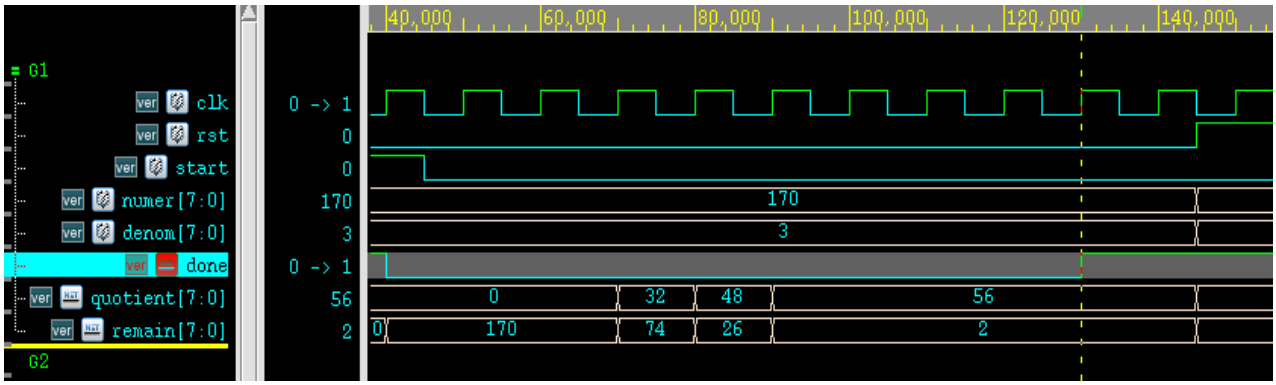
8) quotient for business. Only when done is 1, the value of quotient is the final result calculated;

9) remain is the remainder. Only when done is 1, the value of the remaining is the final result calculated;

10) done is the signal to complete the calculation. When done is set to 1, the calculation is complete. In order to ensure the correctness of the output, you need to use the output values quotient and remain when the done signal is 1, and also provide the next set of input values after the done signal is 1.

The simulation waveform is as follows:

numer = 170 , denom = 3 , When start changes from 1 to 0, the calculation starts until done = 1 and quotient = 56 and remaining = 2 are obtained.



5. Click "OK" to complete the Divider settings, and the TD will give you the path to the generated file.

You can open an existing IP by selecting Tools → IP Generator and selecting “Edit an existing IP core”.

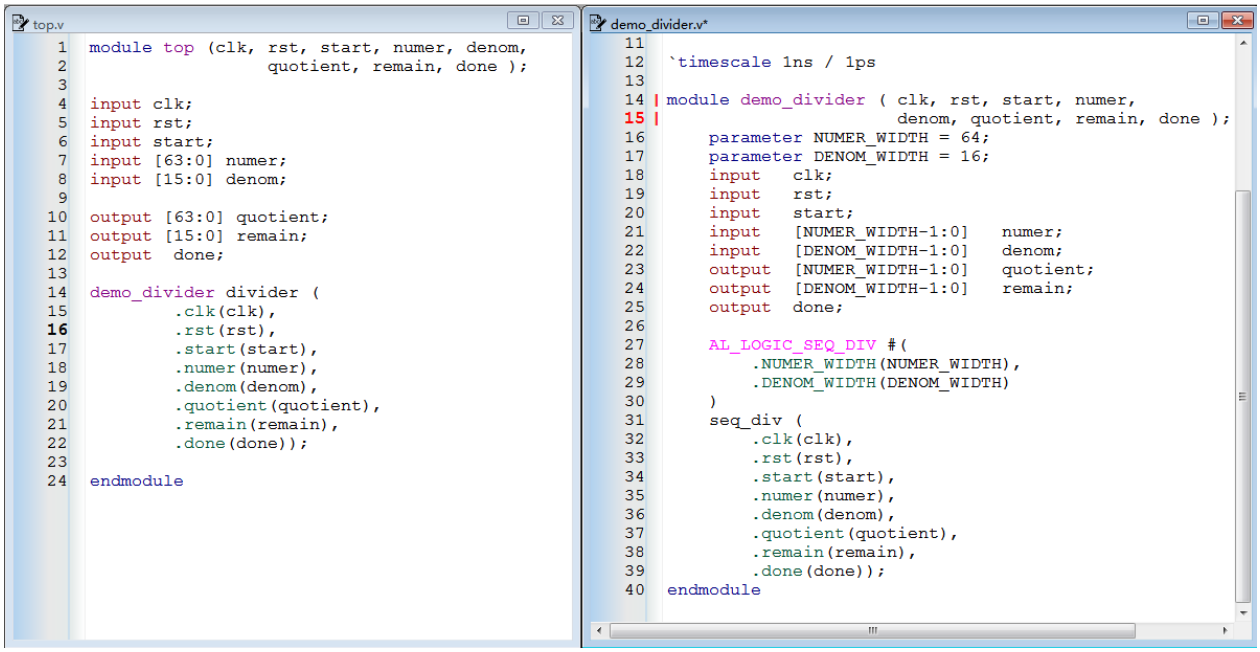
3.4.2 Instantiate the Divider module

This manual uses the new project as an example to introduce the process of instantiating the Divider module. Users can also instantiate on the basis of existing projects, and the instantiation process is consistent.

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

2. Add the demo_divider.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 the Save button to complete the instantiation of the Divider module.



top.v

```
1  module top (clk, rst, start, numer, denom,
2                  quotient, remain, done );
3
4  input clk;
5  input rst;
6  input start;
7  input [63:0] numer;
8  input [15:0] denom;
9
10 output [63:0] quotient;
11 output [15:0] remain;
12 output  done;
13
14 demo_divider divider (
15         .clk(clk),
16         .rst(rst),
17         .start(start),
18         .numer(numer),
19         .denom(denom),
20         .quotient(quotient),
21         .remain(remain),
22         .done(done));
23
24 endmodule
```

demo_divider.v*

```
11
12 `timescale 1ns / 1ps
13
14 module demo_divider ( clk, rst, start, numer,
15                       denom, quotient, remain, done );
16     parameter NUMER_WIDTH = 64;
17     parameter DENOM_WIDTH = 16;
18     input   clk;
19     input   rst;
20     input   start;
21     input   [NUMER_WIDTH-1:0]   numer;
22     input   [DENOM_WIDTH-1:0]   denom;
23     output  [NUMER_WIDTH-1:0]   quotient;
24     output  [DENOM_WIDTH-1:0]   remain;
25     output  done;
26
27     AL_LOGIC_SEQ_DIV #(
28         .NUMER_WIDTH(NUMER_WIDTH),
29         .DENOM_WIDTH(DENOM_WIDTH)
30     )
31     seq_div (
32         .clk(clk),
33         .rst(rst),
34         .start(start),
35         .numer(numer),
36         .denom(denom),
37         .quotient(quotient),
38         .remain(remain),
39         .done(done));
40 endmodule
```

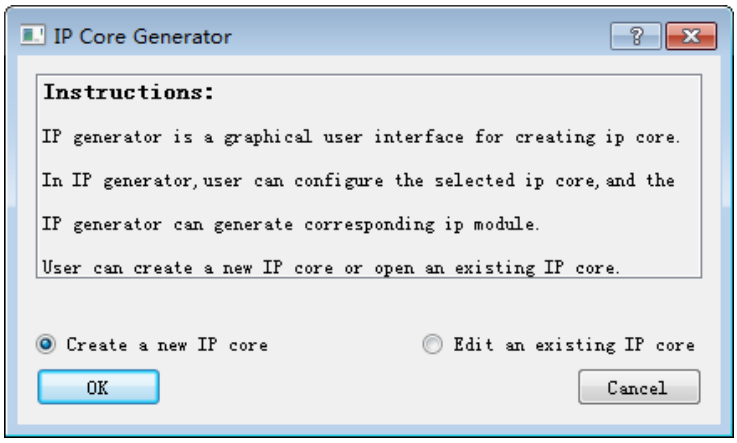
3.5 BRAM Module

The AL3 series devices support the Embedded Memory Block. Two types of EMB are included in the AL3-10: EMB9K and EMB32K.

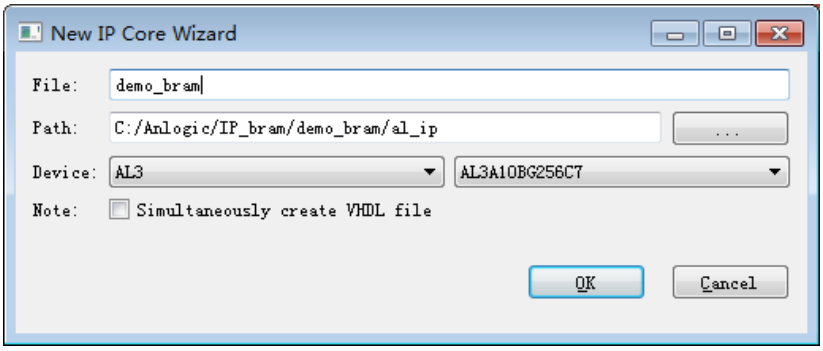
EMB9K Each block has a capacity of 9Kbits, and multiple EMB9K modules are arranged in a row and distributed in columns in an array of Programmable Function Blocks (PFBs). The EMB32K has a capacity of 32Kbits per block and is distributed in the IO gap.

3.5.1 Create a BRAM module

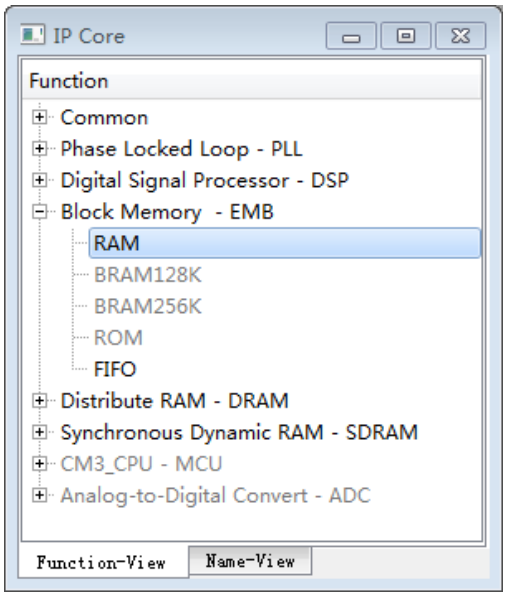
3. 选择 **Tools ⇒ IP Generator**，选择“ **Create a new IP core**”



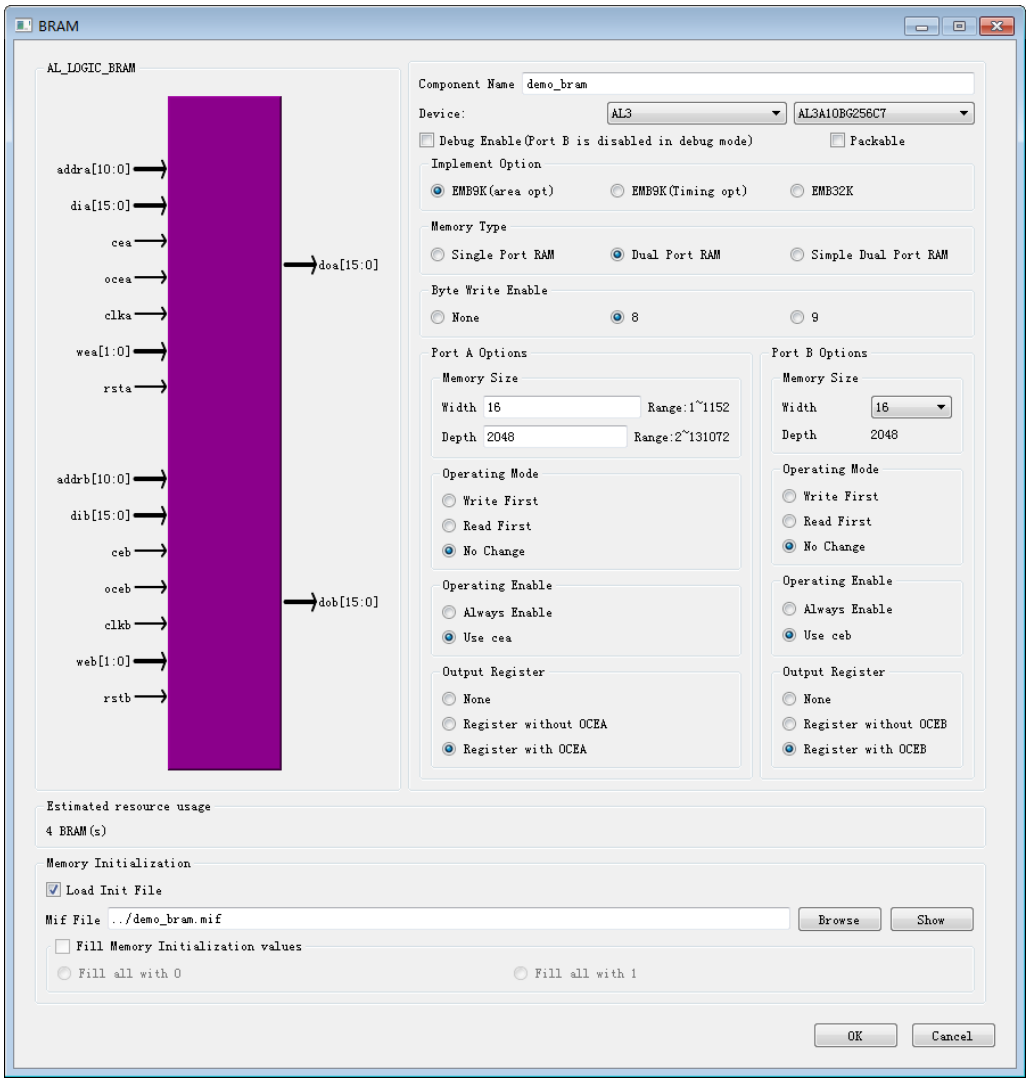
4. Enter the module name and select the storage path. Here, if a BRAM module is created on an engineering basis, the storage path and device name will be consistent with the project. If you create a BRAM module without engineering, you will need to manually set the save path and device name. If "Simultaneously create VHDL file" is checked, the TD will generate the corresponding VHDL file.



5. Expand Block Memory in the Function window and double-click RAM to open the configuration interface.



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



This manual uses EMB9K as an example to introduce the use of the AL3 series device BRAM module.

EMB9K can achieve:

- Single port RAM (Single Port RAM)
- Dual port RAM (Dual Port RAM)
- Simple dual RAM (also known as pseudo dual port)

Features supported by the EMB9K module are:

- 9216 （9K）bits / each block
- A/B Port clock independent
- A/B port data width can be separately configured, true dual port from x1 to x9, support x18 simple dual port (write one read)
- 9 or 18-bit write operation with Byte Enable control
- Output latch selectable (supports 1-stage pipeline)
- Support data initialization in RAM mode (data initialization of EMB9K during initialization through the initialization file)
- Support multiple write modes. You can choose No Change, Read First, Write First.
- Support for Byte Enable function.

If the check box in front of “Debug Enable” is checked, TD will default to the EMB mode as 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 based on area optimization, and EMB9k (fast) is mainly based on timing optimization.

Byte Enable refers to the input data of BRAM. When the bit width is multiple bytes, a byte enable signal is used to control whether each byte is written or not when reading data.

The value of Byte Write Enable can be selected as None or 8 or 9 on the interface. When byte-write is None, it means that the byte enable function is not enabled. When byte-write is 8, the data width of port A and port B (if port B) must be an integer multiple of 8, and the value of multiple is used.

The width of wea and web; when byte-write is 9, the data width of port A and port B (if port B) must be a multiple of 9, and the multiple is used as the width of wea and web. When the byte enable function is enabled, BRAM32K is not recommended because the BRAM is too small and wastes a lot of memory.

7. Add initialization file

The TD's initialization file supports the user's description in a third-party mif (memory initialization file) format, or the verilog storage space initialization dat format.

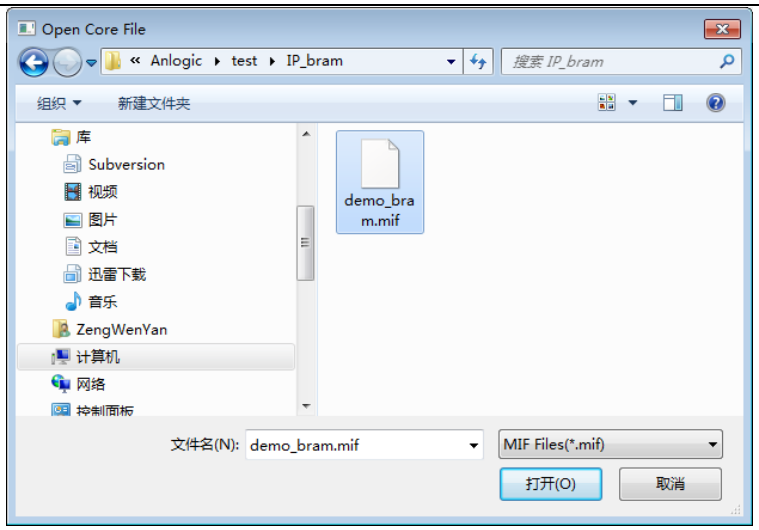
- The mif format is described as follows:

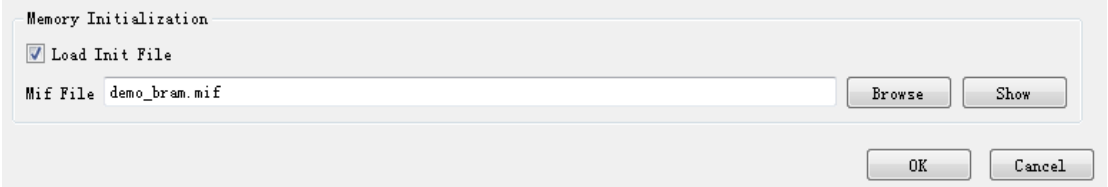
  The initialization file in mif format contains each initialization address and data, and the depth and width of the memory data must be defined. Users can define data and address formats as binary BIN, hexadecimal HEX, octal OCT, unsigned decimal UNS, and more. The value of the data must match the data format.

- The dat format is described as follows:

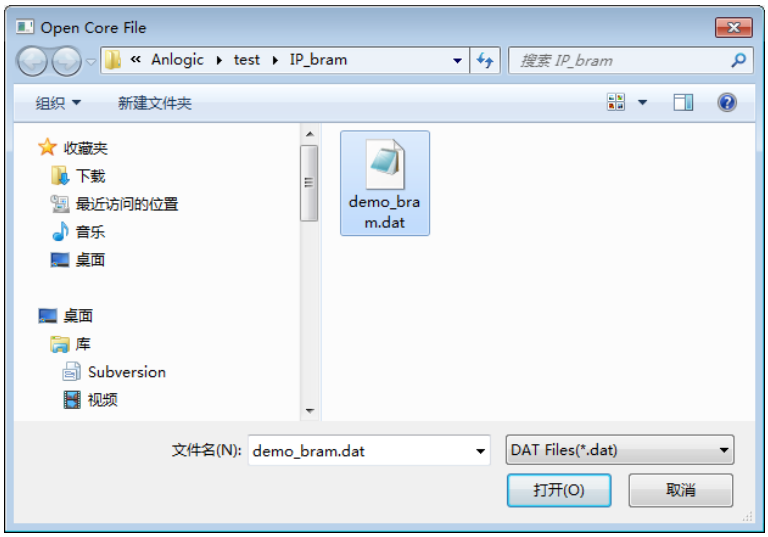
  The memory data can be stored in a file in hexadecimal address, where the address is indicated by "@". The starting address is defined by the user, and the ending address can be determined accordingly according to the depth of the memory data. In order to be able to clearly match the data to the address, an identifiable address flag is usually added to the file. If the initialization file is large, the address can be omitted directly.

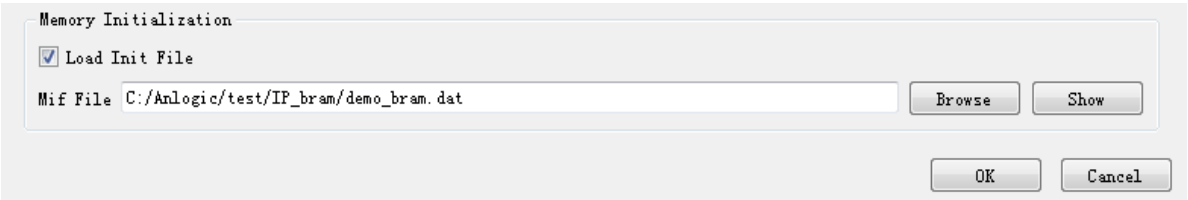
When adding an initialization file to the BRAM module, you can check the checkbox in front of “Load Init File” and select the file to be added. When the .mif format is selected in the drop-down box in the lower right corner, it is only available in the folder. The mif file is available for users to choose from, as shown in the following figure.



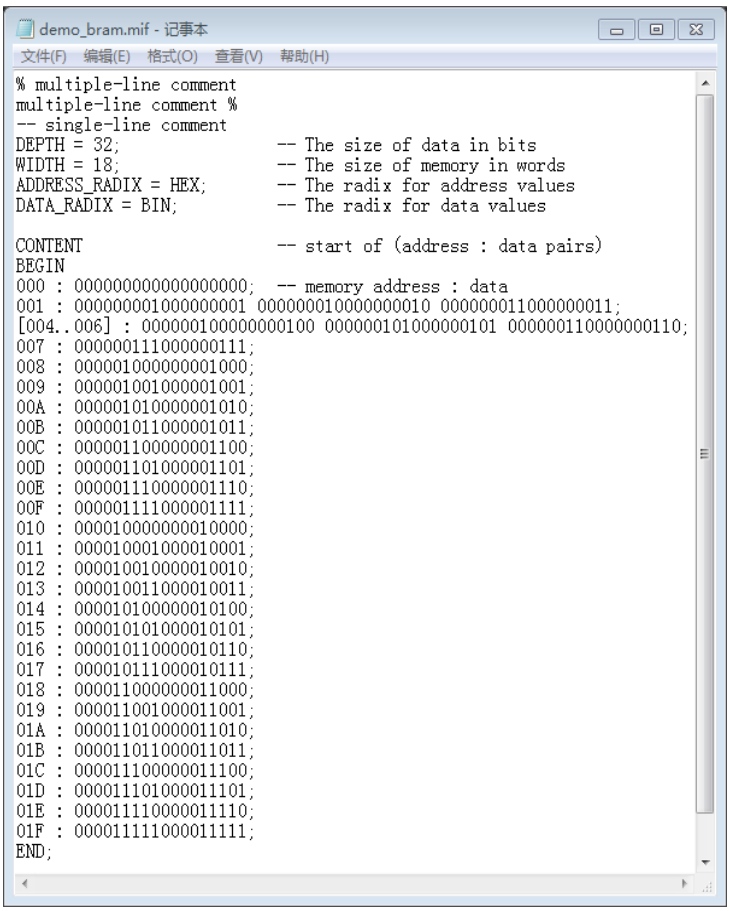


If the .dat format is selected, only the .dat file is provided in the folder for the user to select, as shown in the following figure.

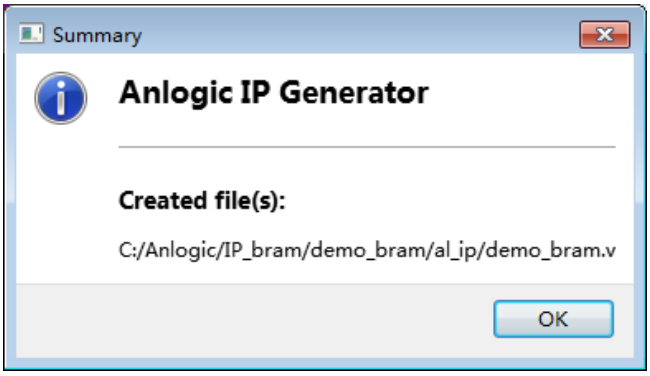




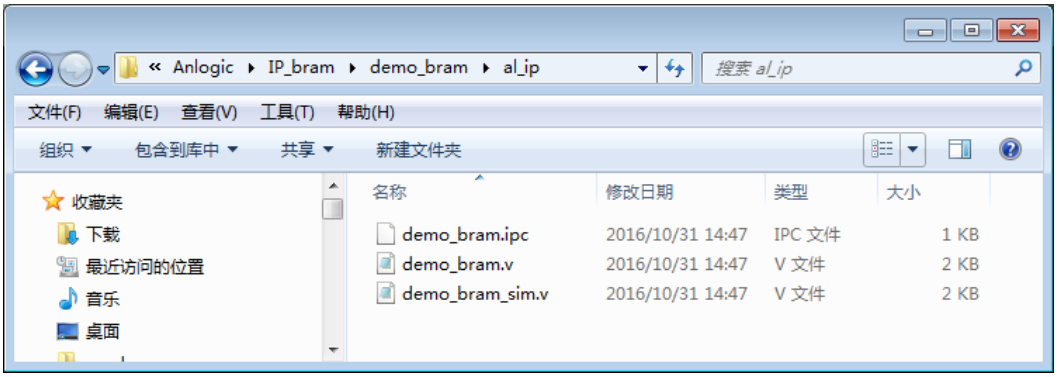
Click the button "Show" to see the contents of the added file. Here is an example of the format of the mif file and the dat file.



Click "OK" to complete the creation of BRAM. The path to the generated file by TD is as follows:

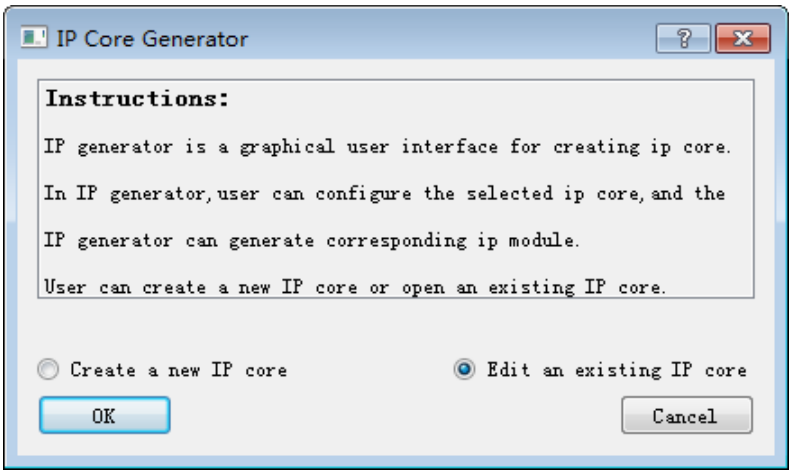


The following files can be seen in the project directory:



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

选择 **Tools ⇒ IP Generator**，选择“ **Edit an exist IP core**”。



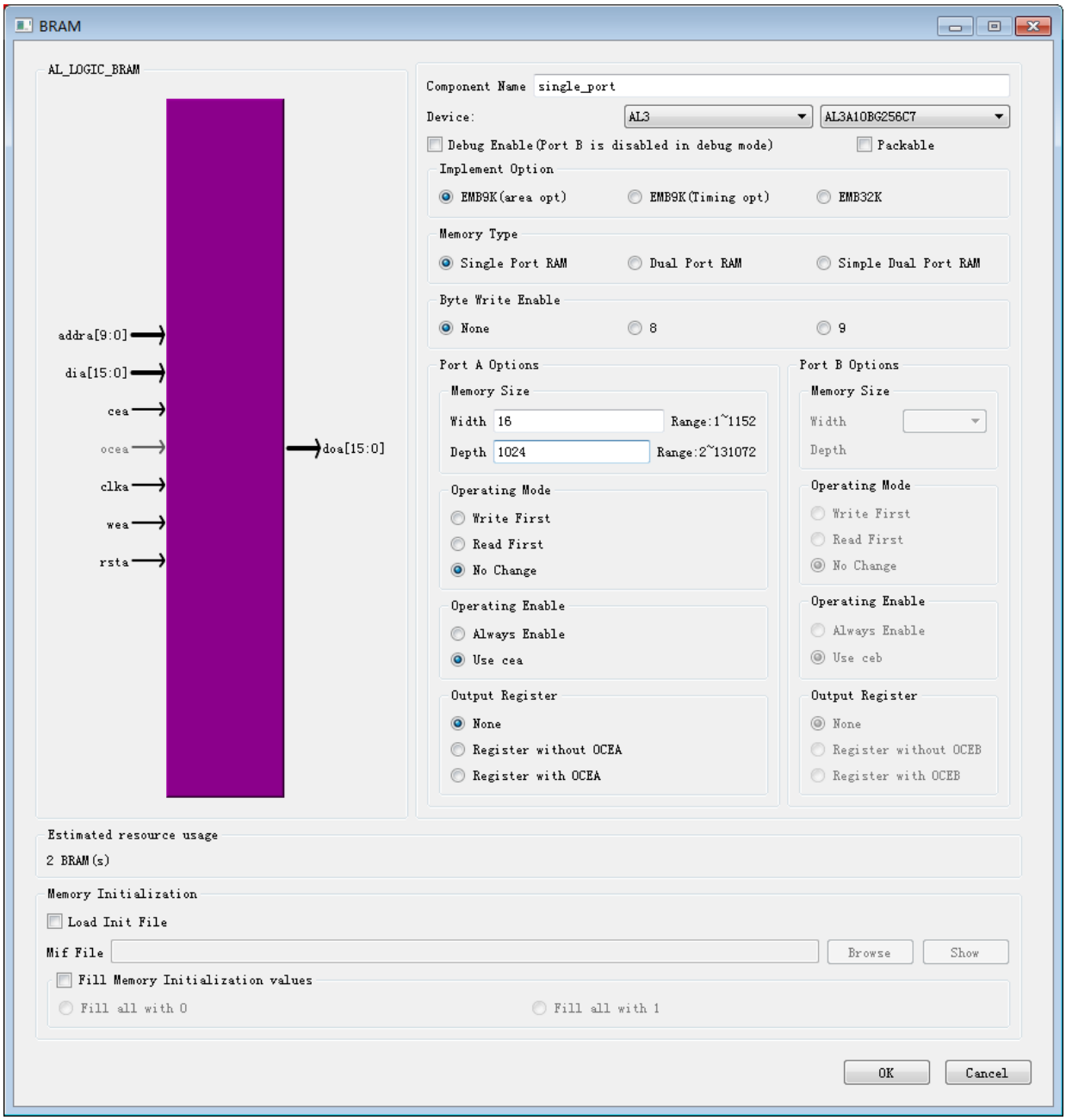
Demo_bram.v creates a file for the BRAM module to instantiate for the user.

Demo_bram_sim.v describes how BRAM is divided for user emulation and Debug, but it cannot be added as a source file for the project, otherwise it will conflict with the module in demo_bram.v.

The following are examples of interface settings and generated files for three different modes of BRAM:

1. Single port RAM (Single Port RAM)

Single-port mode supports read or write operations to the same address that are not simultaneous. The interface settings are as follows:

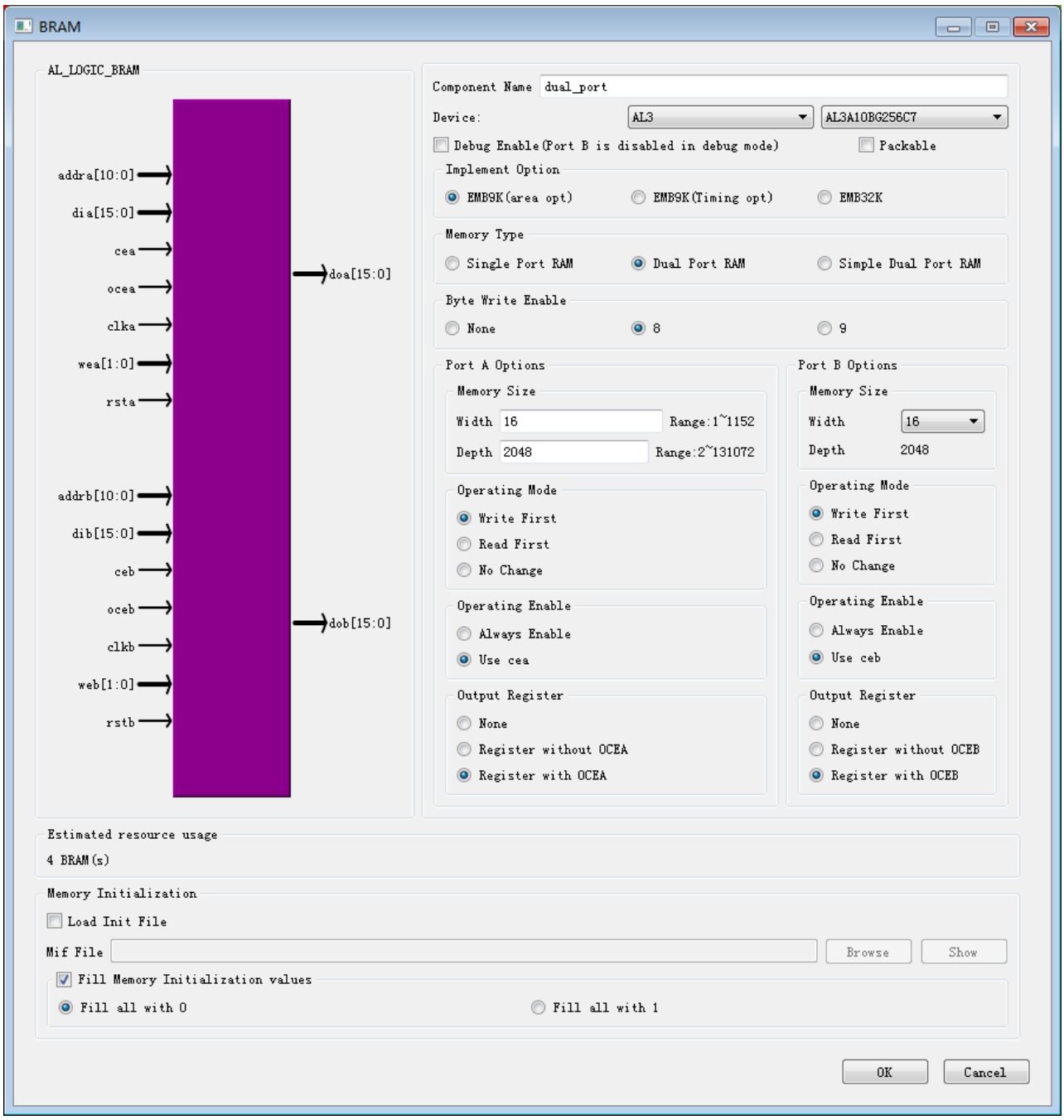


The files generated in single-port mode are as follows:

```
\*************************************************************/

`timescale 1ns / 1ps

module single_port ( doa, dia, addra, cea, clka, wea, rsta );

    output [15:0] doa;

    input  [15:0] dia;
    input  [9:0] addra;
    input  wea;
    input  cea;
    input  clka;
    input  rsta;

    AL_LOGIC_BRAM #( .DATA_WIDTH_A(16),
                .ADDR_WIDTH_A(10),
                .DATA_DEPTH_A(1024),
                .DATA_WIDTH_B(16),
                .ADDR_WIDTH_B(10),
                .DATA_DEPTH_B(1024),
                .MODE("SP"),
                .REGMODE_A("NOREG"),
                .WRITEMODE_A("NORMAL"),
                .RESETMODE("SYNC"),
                .IMPLEMENT("9K"),
                .DEBUGGABLE("NO"),
                .PACKABLE("NO"),
                .INIT_FILE("NONE"),
                .FILL_ALL("NONE"))
            inst(
                .dia(dia),
                .dib({16{1'b0}}),
                .addra(addra),
                .addrb({10{1'b0}}),
                .cea(cea),
                .ceb(1'b0),
                .ocea(1'b0),
                .oceb(1'b0),
                .clka(clka),
                .clkb(1'b0),
                .wea(wea),
                .web(1'b0),
                .bea(1'b0),
                .beb(1'b0),
                .rsta(rsta),
                .rstb(1'b0),
                .doa(doa),
                .dob());

endmodule
```

2. Dual port mode (Dual Port RAM)

Dual port mode supports all independent read and write operation combinations of port A/port B: two reads, two writes, one read and one write. The interface settings are as follows:



The files generated in dual port mode are as follows:

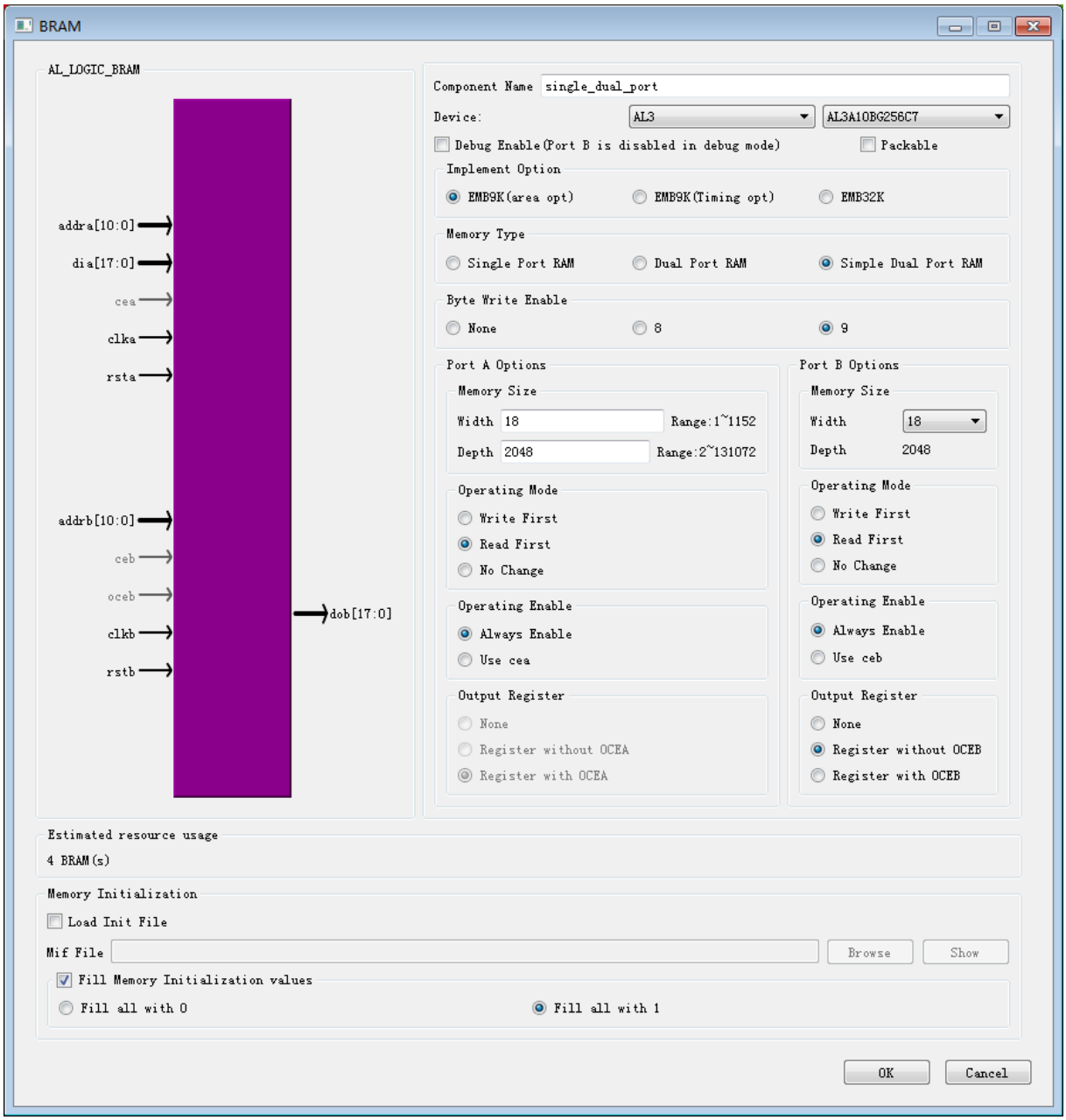
```
13
14  module dual_port (
15      doa, dia, addra, cea, clka, wea, rsta, ocea,
16      dob, dib, addrb, ceb, clkb, web, rstb, oceb
17  );
18      output [15:0] doa;
19      output [15:0] dob;
20      input  [15:0] dia;
21      input  [15:0] dib;
22      input  [10:0] addra;
23      input  [10:0] 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),
```

3. Simple Dual Port RAM (Simple Dual Port RAM)

When an EMB9K is configured for 18-bit write or 18-bit readout, the EMB9K does not support dual-port mode, but supports both single-port and simple dual-port modes. When the EMB9K is in the 18-bit mode, the A port control signal is used as the write control signal, and the B port control signal is used as the read control signal.

When EMB9K is configured for 18-bit write, dib[8:0] is used as the upper 9-bit data input, dia[8:0] is the lower 9-bit data input; when EMB9K is configured for 18-bit read, dob[8: 0] As the high 9-bit data output, doa[8:0] is output as the lower 9-bit data.

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



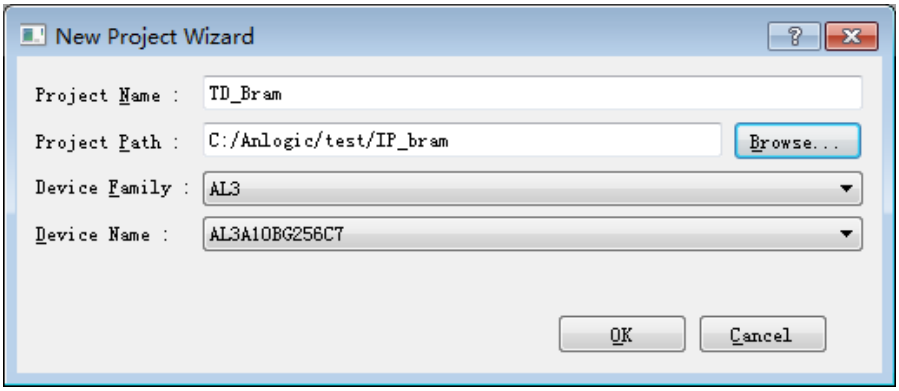
The files generated in simple dual-port mode are as follows:

```
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
```

3.5.2 Instantiate the BRAM module

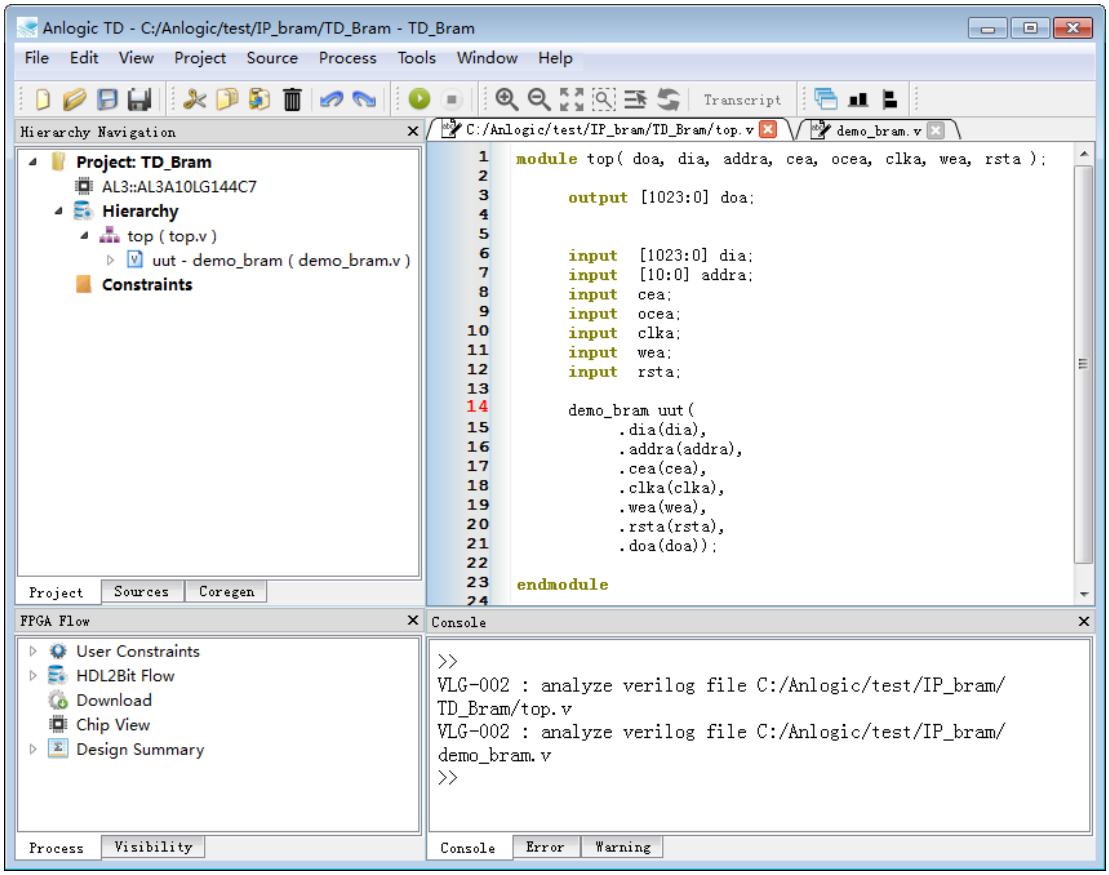
This manual introduces the process of instantiating a BRAM module with a new project as an example. Users can also instantiate on the basis of existing projects, and the instantiation process is consistent.

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, modify the inst name and port name, and click the Save button to complete the instantiation of the BRAM module.

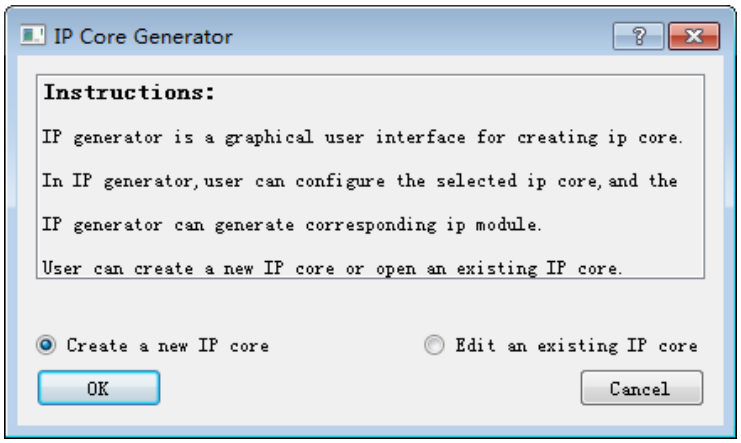


3.6 FIFO Module

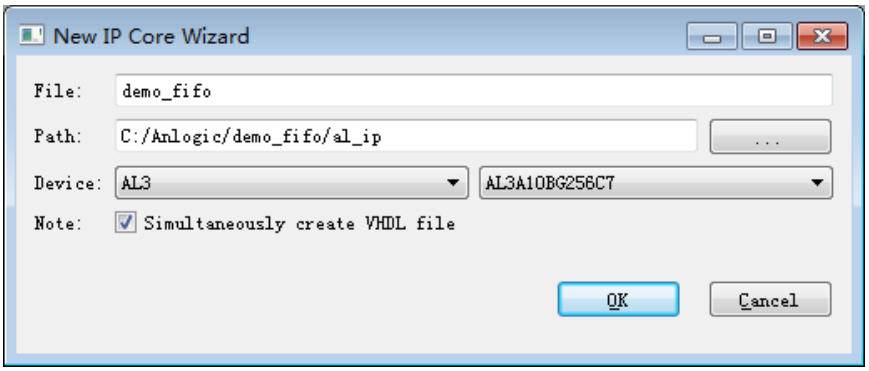
The EMB9k internal hardware supports synchronous/asynchronous FIFO mode. The EMB9K bit width setting in FIFO mode is the same as the simple dual port RAM setting, supporting up to 18 bit width.

3.6.1 Create a FIFO module

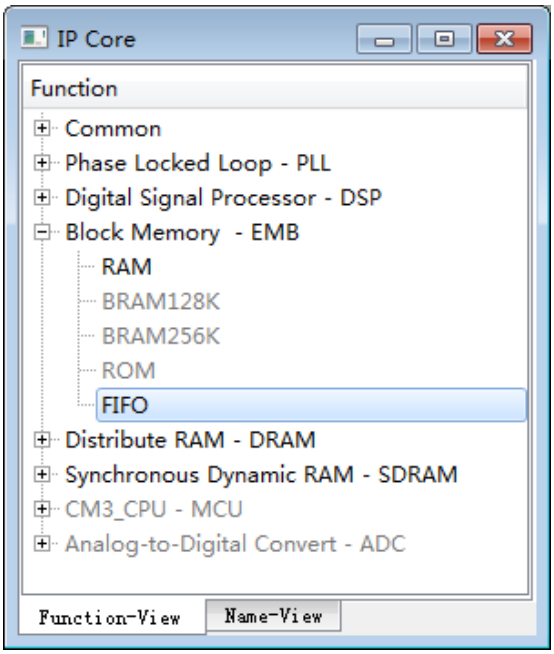
1. 选择 **Tools ⇒ IP Generator**，选择“ **Create a new IP core**”



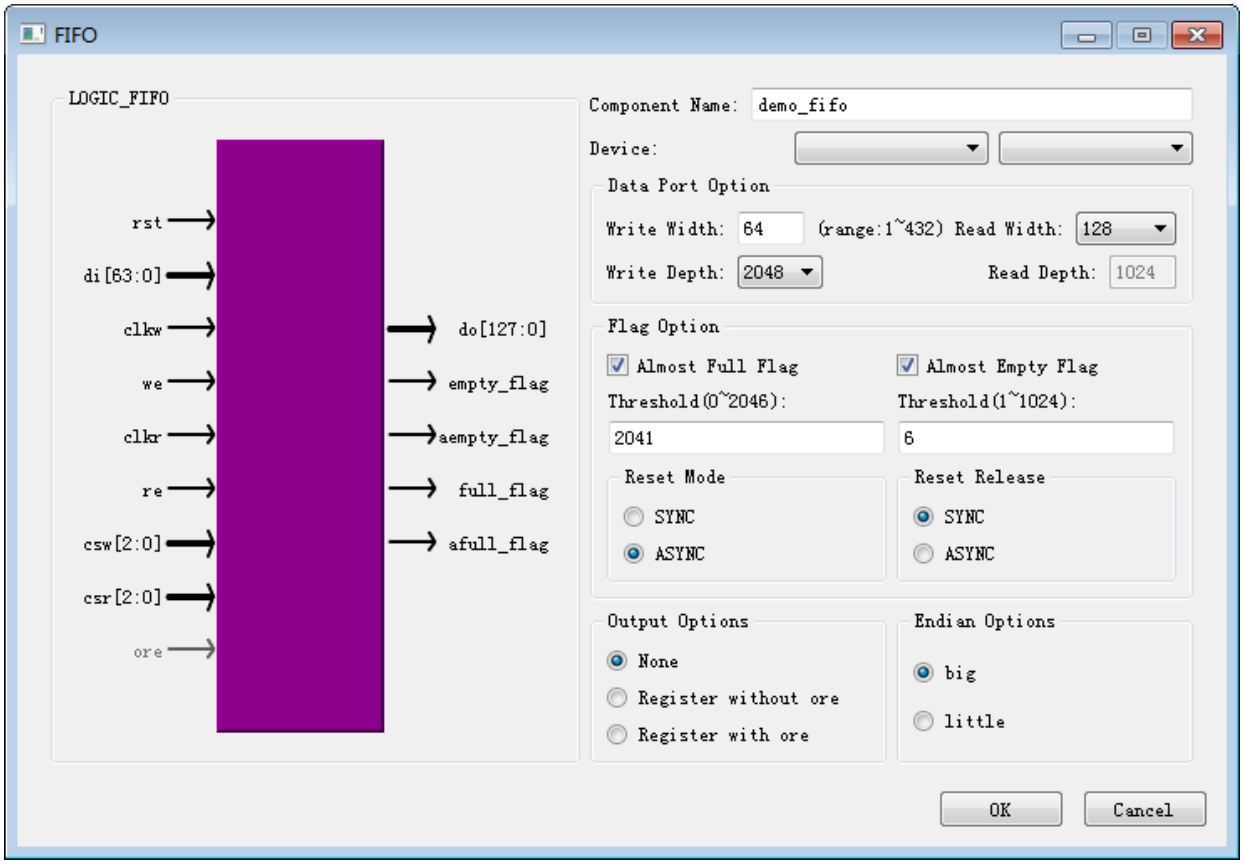
2. Enter the module name and select the storage path. Here, if a FIFO module is created on an engineering basis, the storage path and device name will be consistent with the project.
   If the FIFO module is created without engineering, the user must manually set the save path and device name. If "Simultaneously create VHDL file" is checked, the TD will generate the corresponding VHDL file.



3. Expand Block Memory in the Function window, double-click FIFO to open the configuration interface.



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



Endian Options: The size of the output data is as follows:

Endian Options : In big mode:

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

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

Endian Options: In little mode

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

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

The empty full flag in FIFO mode indicates:

FIFO Flag name	direction	Range	Description
Empty_Flag (EF)	Output	0	FIFO read empty flag, synchronized with clkr
Aempty_Flag (AE)	Output	1 to FF-1	The FIFO is almost empty and is synchronized with clkr. The relative read latency 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	The FIFO is almost full and is synchronized with clkw. The almost full capacity of the FIFO is determined by the AF_POINTER parameter.

- Empty full flag set

In FIFO mode, the user can set the FIFO empty flag attribute via software: Empty flag (Empty_Flag), Almost_Empty, Full_Flag, Almost_Full. When the internal counter counts to the flag value, the corresponding port of the empty pointer EF/AE/FF/AF outputs a high level.

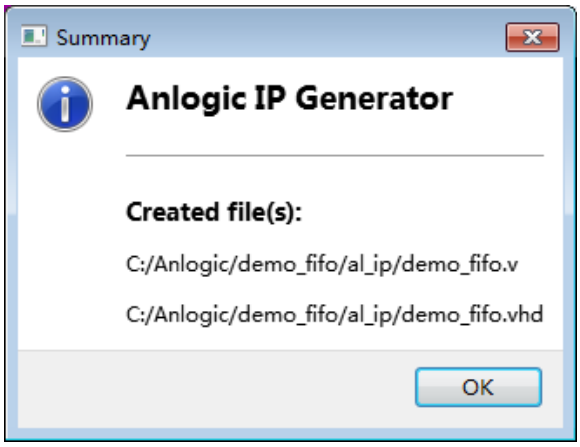
- Description of the empty full pointer

The FIFO mode lower finger (EF) is fixed at 0, the AE/AF/FF pointer is 14-bit binary number "0b0X_XXXX_XXXX_XXXX", the highest bit [13] is always 0, and the valid bit [12:0] represents 8K depth 1 bit width. The minimum 4 bits [3:0] of the pointer is valid depending on the bit width of the read/write.

- Simple FIFO with csw/csr reverse configuration

  csw is the 3-bit chip select signal of the FIFO write port, which can be inverted; csr is the 3-bit chip select signal of the FIFO read port, which can be reversed. To avoid pointer overflow when the FIFO is full or read, the full signal can be reversed to the csw terminal through the interconnect resource, and the null signal is inverted and then connected to the csr terminal. Reverse logic can take advantage of the inverse and logic implementations inside csw/csr.

5. Click "OK" to complete the FIFO setup and TD will give the path to the generated file.



The existing demo_fifo.ipc can also be opened and edited via the "Edit an existing IP core" method.

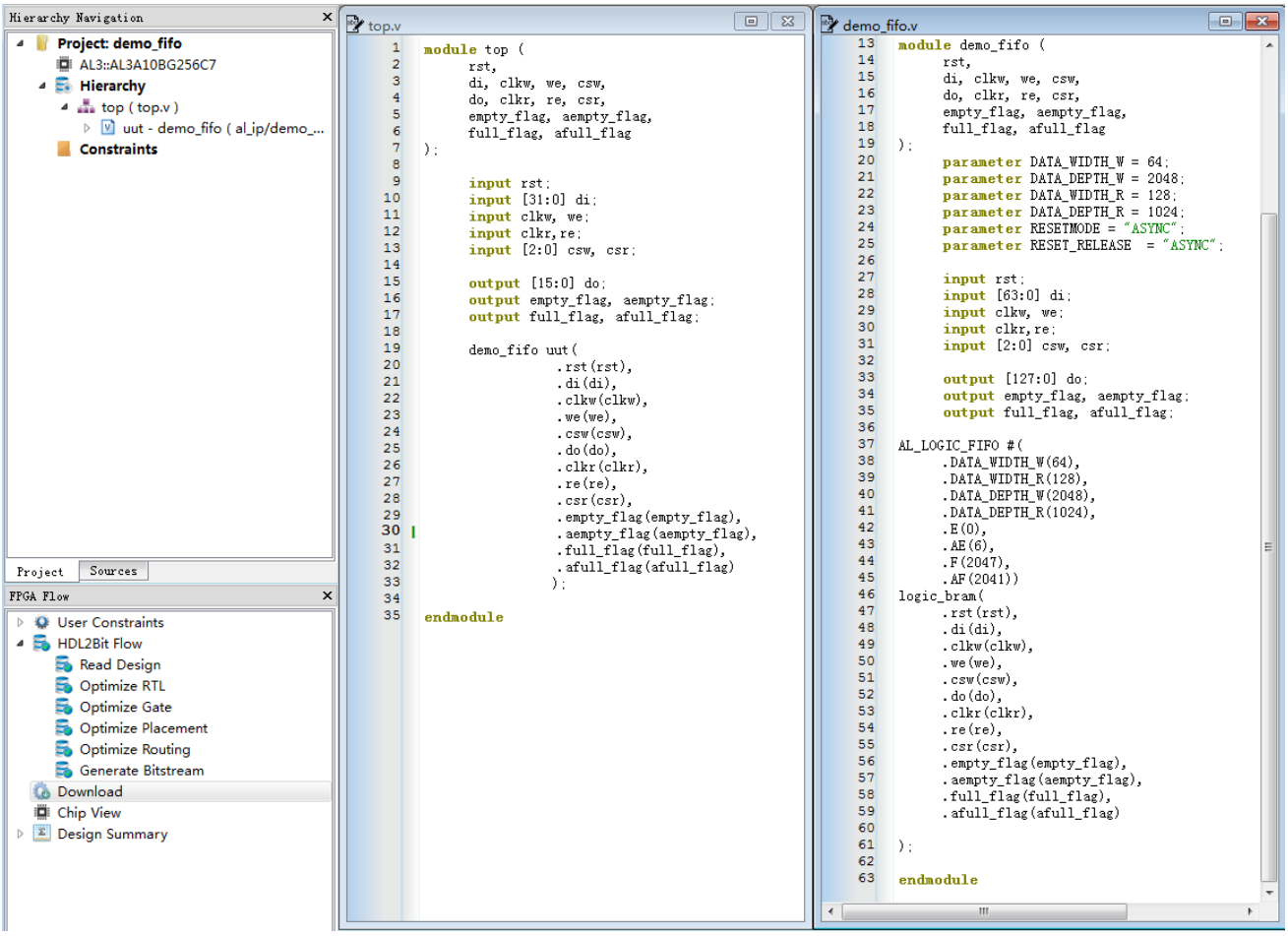
3.6.2 Instantiate the FIFO module

This manual introduces the process of instantiating a FIFO module by taking a new project as an example. Users can also instantiate on the basis of existing projects, and the instantiation process is consistent.

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, modify the inst name and port name, and click the Save button to complete the instantiation of the FIFO module.

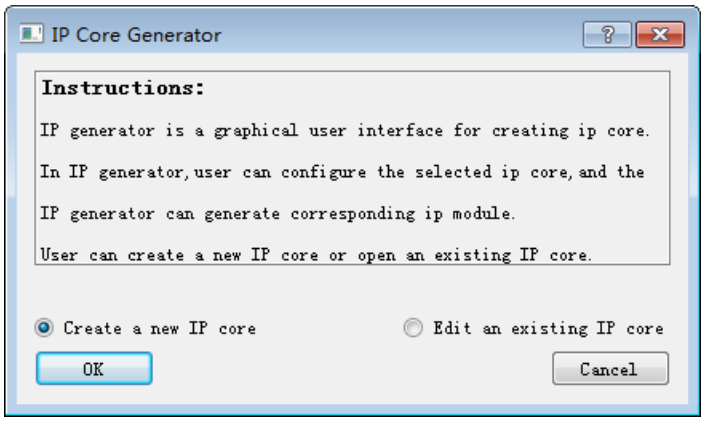


3.7 DRAM Module

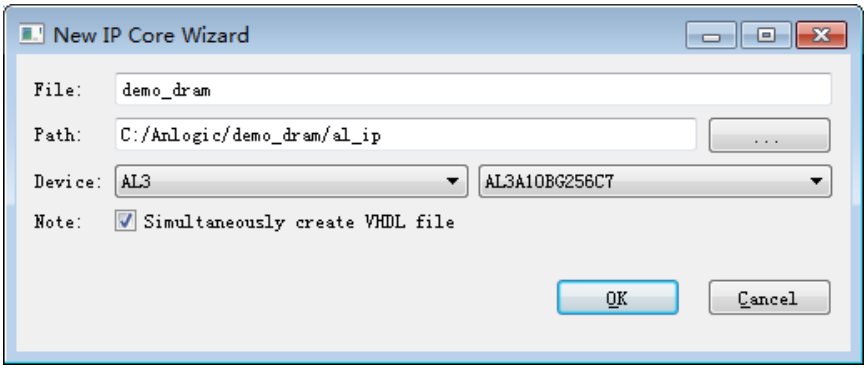
In the AL3 family of devices, each PLB contains 2 MSLICEs (4 LUTs) for 16x4 RAM blocks, each supporting a simple dual-port RAM.

3.7.1 Create a DRAM module

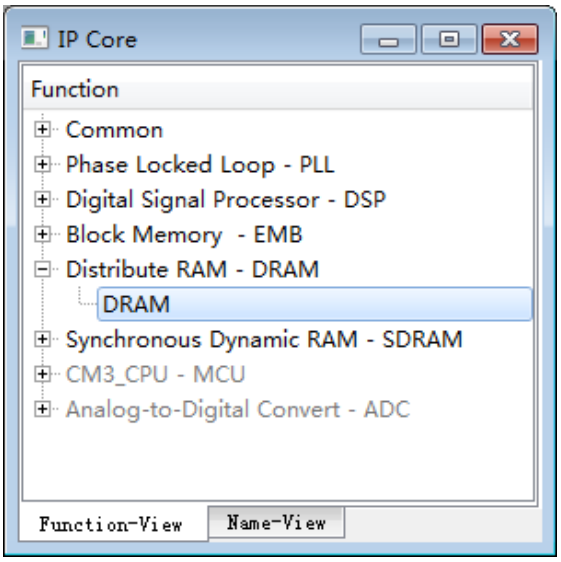
1. 选择 **Tools ⇒ IP Generator**，选择“ **Create a new IP core**”



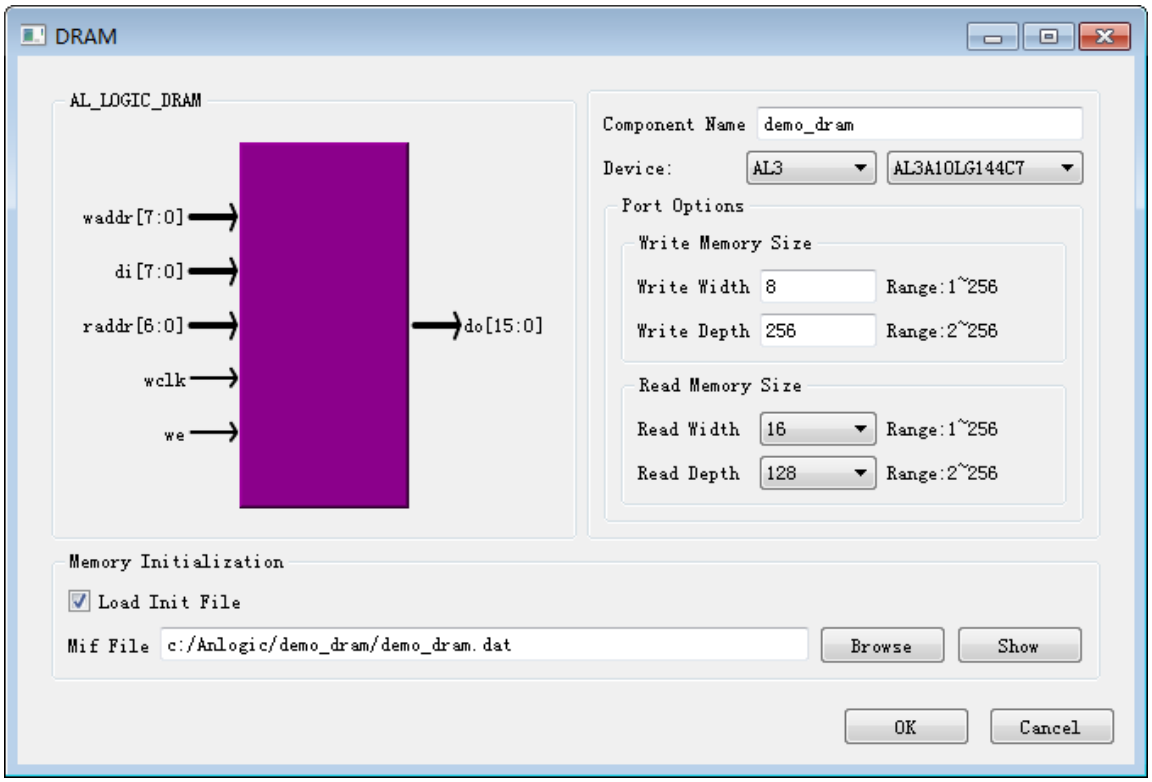
2. Enter the module name and select the storage path. Here, if a DRAM module is created on an engineering basis, the storage path and device name will be consistent with the project. If you create a DRAM module without engineering, you need to manually set the save path and device name. If "Simultaneously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



3. Expand Distribute RAM in the Function window, double-click DRAM to open the configuration interface.



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



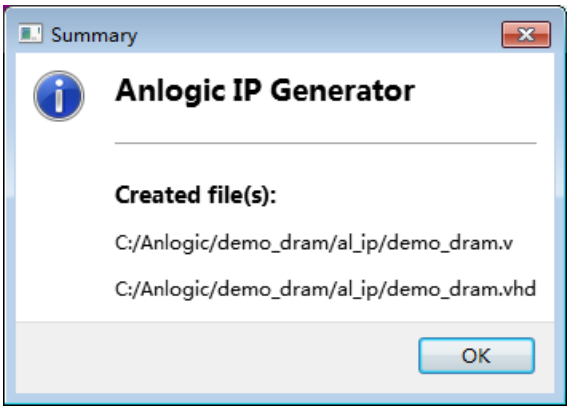
Here, the format of the Init File is the same as in BRAM.

The port list and description are as follows:

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

Port	Features	Description
WCLK	Write clock	Active rising edge (programmable reverse)
WE	Write enable	Embedded WCLK rising edge sync latch
WADDR[3:0]	Write address	Embedded WCLK rising edge sync latch
DI[3:0]	Write data	Embedded WCLK rising edge sync latch
RADDR[3:0]	Read address	asynchronous
DO[3:0]	Reading data	asynchronous

5. Click "OK" to complete the DRAM setup and TD will give the path to the generated file.



The existing demo_dram.ipc can also be opened and edited via the "Edit an existing IP core" method.

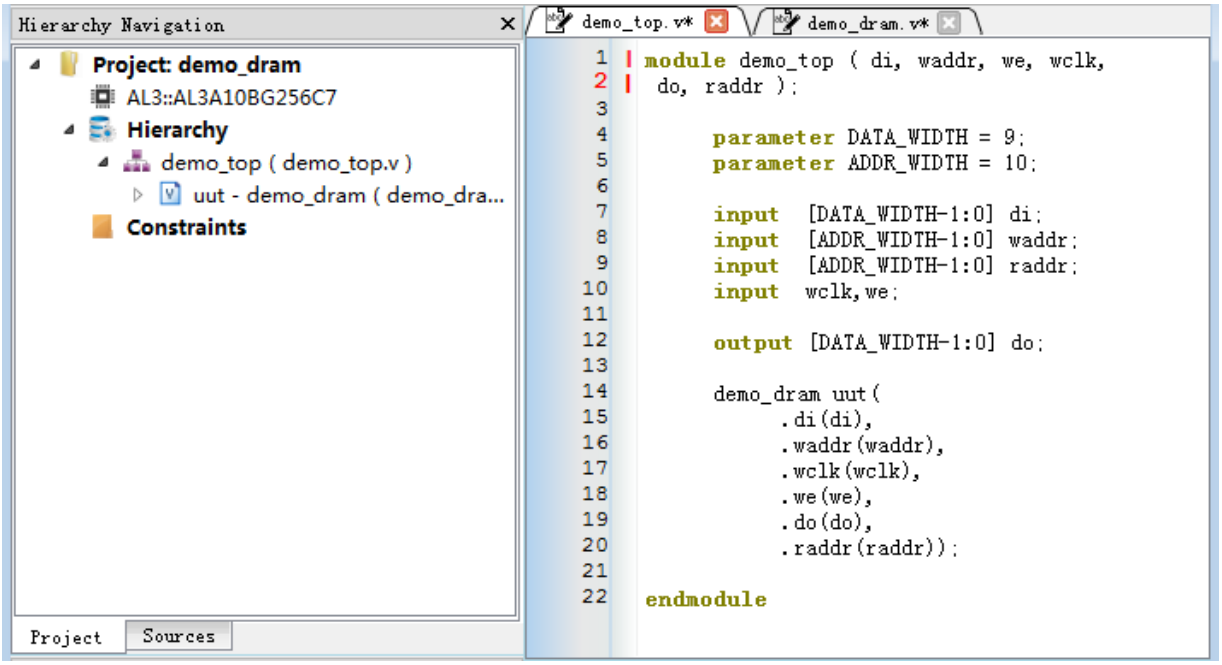
3.7.2 Instantiate DRAM module

This manual introduces the process of instantiating a DRAM module by taking a new project as an example. Users can also instantiate on the basis of existing projects, and 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, modify the inst name and port name, and click the Save button to complete the instantiation of the DRAM module.

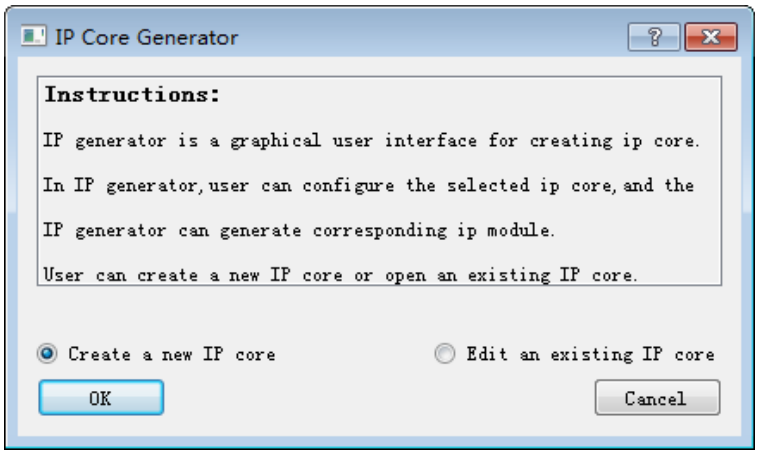


3.8 SDRAM Module

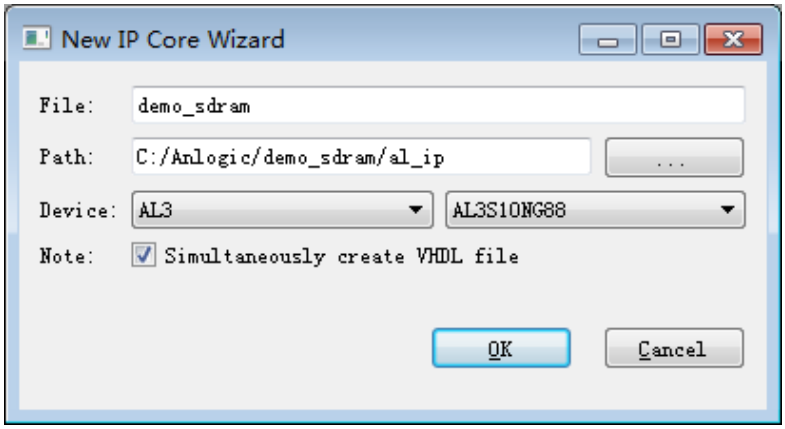
SDRAM is embedded in the latest integrated chip of Anlu. It is deeply integrated with the FPGA through software. When using it, you only need to instantiate the IP module at the top level. For details and use, please refer to Chapter 6 of this manual.

3.8.1 Create SDRAM module

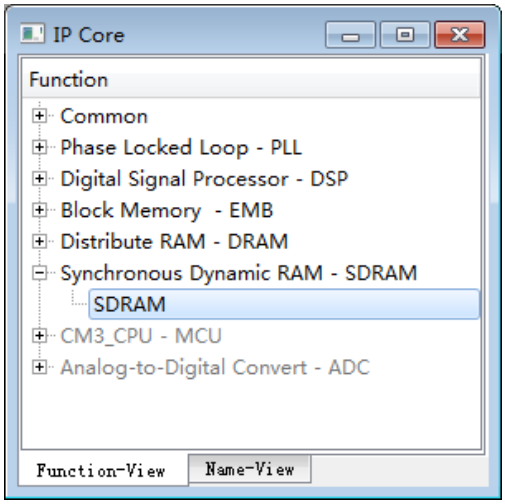
1. 选择 **Tools ⇒ IP Generator**，选择“ **Create a new IP core**”

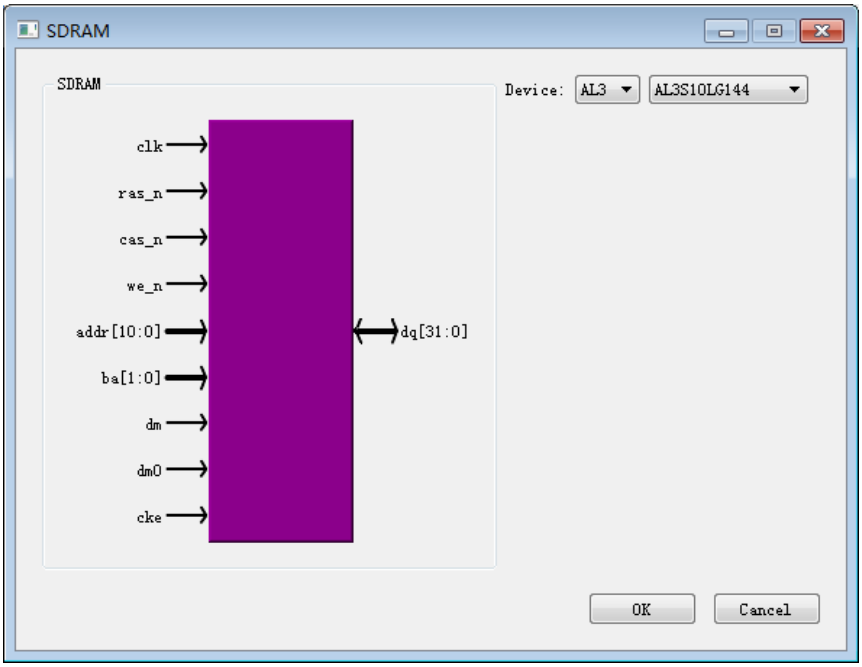


2. Enter the module name and select the storage path. Here, if an SDRAM module is created on an engineering basis, the storage path and device name will be consistent with the project. If you create an SDRAM module without engineering, you need to manually set the save path and device name. If "Simultaneously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

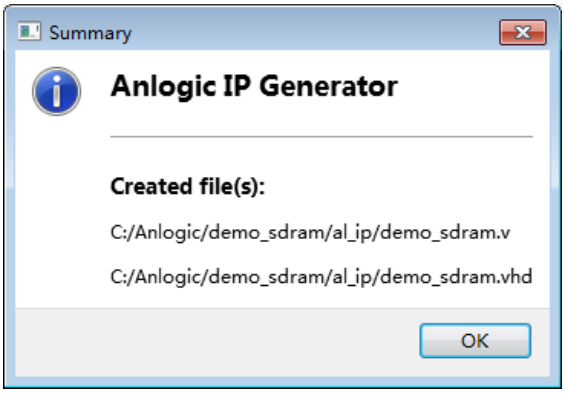


3. Expand Synchronous Dynamic RAM in the Function window, double-click SDRAM to open the configuration interface.





4. Click "OK" to complete the setup and generate the file as follows:

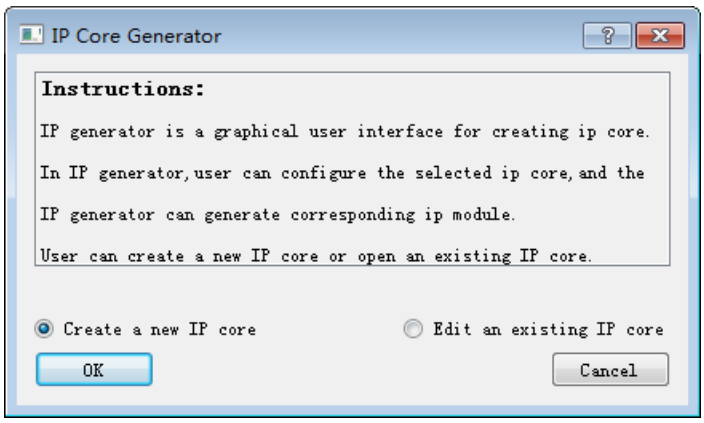


3.9 ADC Module

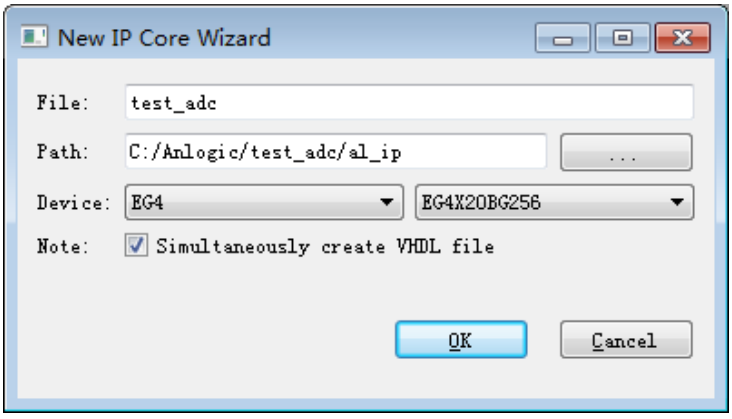
The Eagle series is equipped with an 8-channel 12-bit 1MSPS ADC embedded in the BANK8 chip. The ADC module requires a separate 3.3V analog operating voltage and analog ground as well as a separate VREF voltage input. Eight channel inputs are multiplexed with user IO and can be used as normal IO when the user does not need to use the ADC module. When using an ADC, the VCCIO voltage of BANK8 should not be lower than the ADC analog supply voltage.

3.9.1 Create an ADC module

1. 选择 **Tools ⇒ IP Generator**，选择“ **Create a new IP core**”

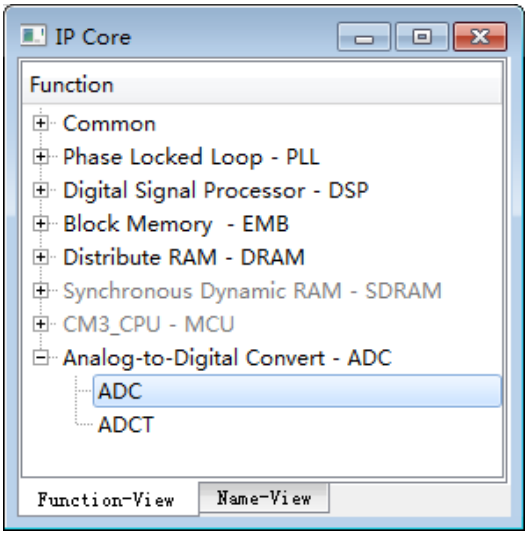


2. Enter the module name and select the storage path. Here, if the ADC module is created on an engineering basis, the storage path and device name will be consistent with the project. If the ADC module is created without engineering, the user needs to manually set the save path and device name. If "Simultaneously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

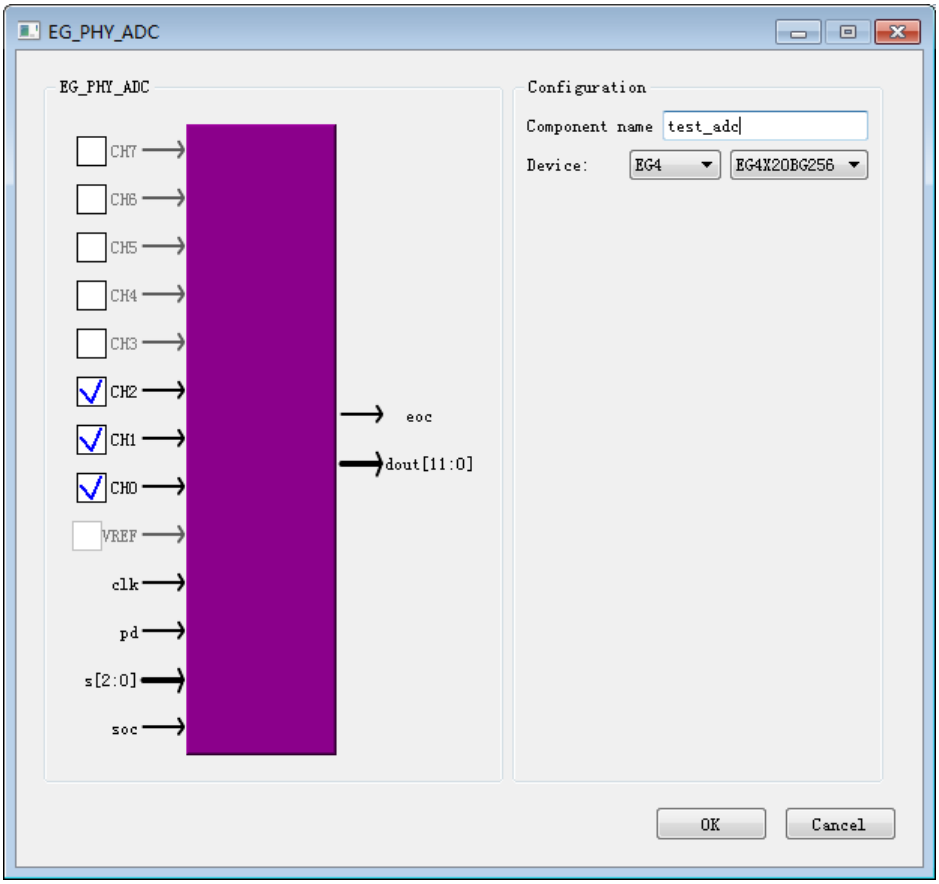


3. Expand Analog-to-Digital Convert in the Function window and double-click ADC to open the configuration interface.

The ADCT module adds a temperature sensor to the ADC module.



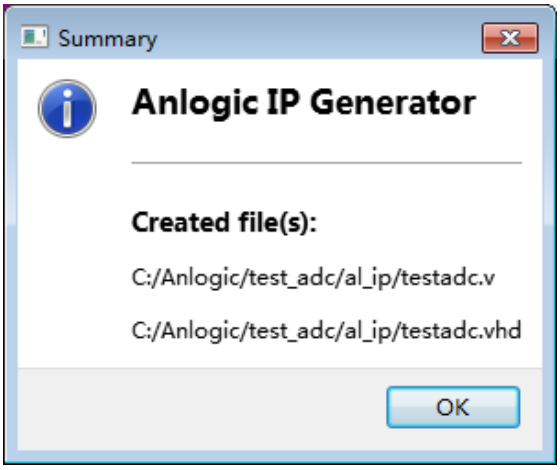
4. Fill in the “Component Name” and select the channel of the ADC. The currently available channel depends on the package type of the current device.



The ADC module external/internal ports are described below:

Chip port name	Port type	Description
clk	External power supply PAD	3.3V Analog power input
pd	External power supply PAD	3.3V Analogue
VREF	External PAD	Independent input, sampling reference analog potential input, input voltage range 2.0V~3.3V, no larger than VDDA
CH<7：0>	External PAD	8 sampling signal inputs, multiplexed with user IO
Internal port name	Port direction	Description
s<2:0>	Input (from FPGA)	ADC channel select signal input
soc	Input (from FPGA)	ADC sample enable signal input, high efficiency
eoc	Output (to FPGA)	ADC conversion complete output, high effective
dout<11:0>	Output (to FPGA)	ADC conversion result for the corresponding channel

5. Click "OK" to complete the setup. The generated file is as follows:



The existing test_adc.ipc can also be opened and edited via the "Edit an existing IP core" method.

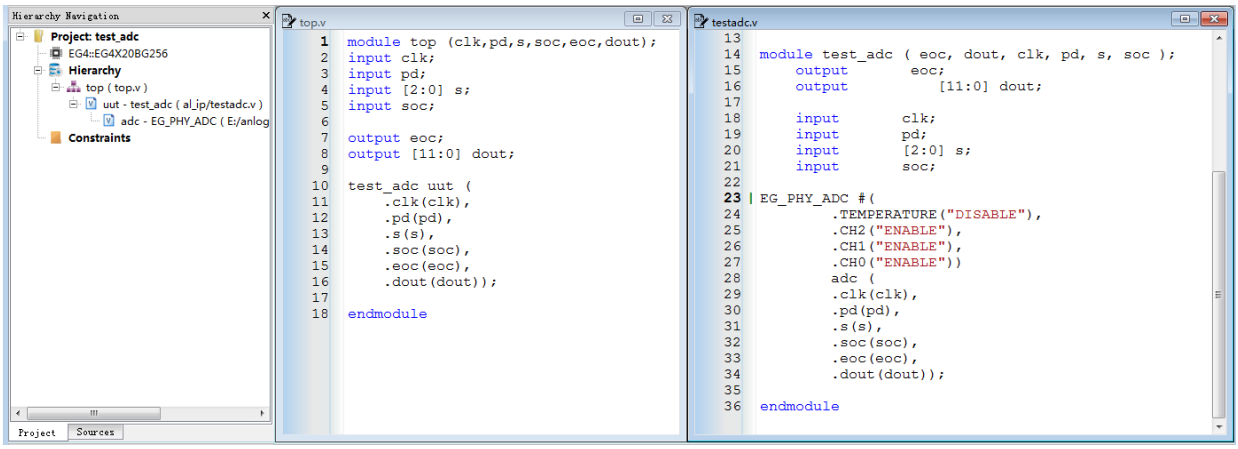
3.9.2 Instantiate the ADC module

This manual introduces the process of instantiating an ADC module with a new project as an example. Users can also instantiate on the basis of existing projects, and the instantiation process is consistent.

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, modify the inst name and port name, and click the Save button to complete the instantiation of the ADC module.



4 User constraint

User constraints consist of two parts: physical constraints and timing constraints. The physical constraints are based on Anlogic's custom ADC (Anlogic Design Constraint) format, which constrains the pin characteristics and internal cell characteristics of the FPGA chip. Pin characteristics include Bank, Location, PullType, IOStandard, SlewRate, Drivestrength, VREF, DiffResistor, and PCIClamp.

The unit feature constraint allows the user to specify the location of the location in the ADC file with set_inst_assignment. Pin feature constraints can be set through the interface.

Refer to Appendix 9.1 for physical constraints.

The timing constraints are based on the industry standard SDC (Synopsys Design Constraint) format, which constrains the external latency information and internal timing requirements of the FPGA chip. The optimization related to the delay in the design flow considers the constraints of the SDC.

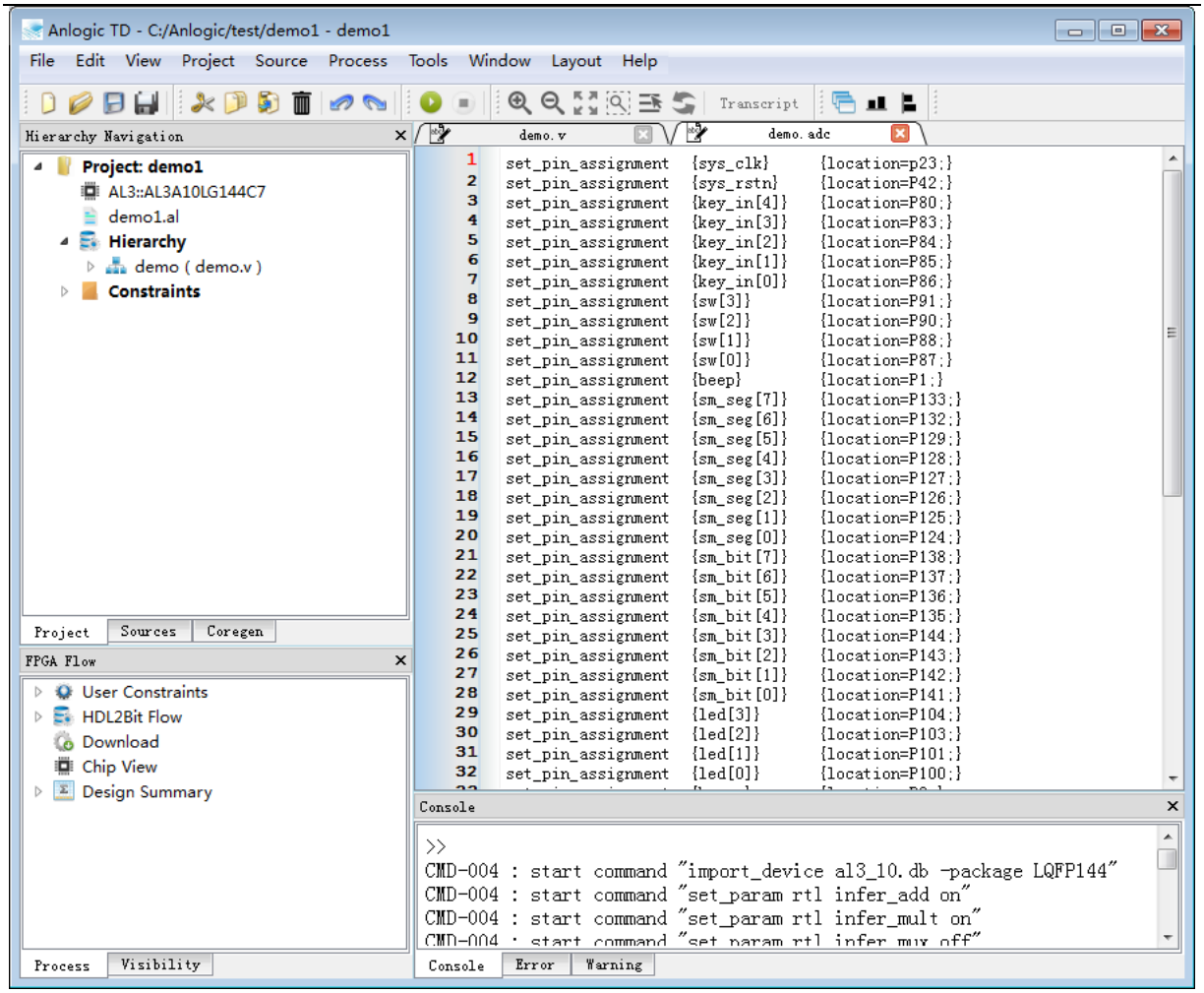
Please refer to Appendix 9.2 for timing constraints.

4.1 Physical constraint

4.1.1 Add IO constraints

New ADC file

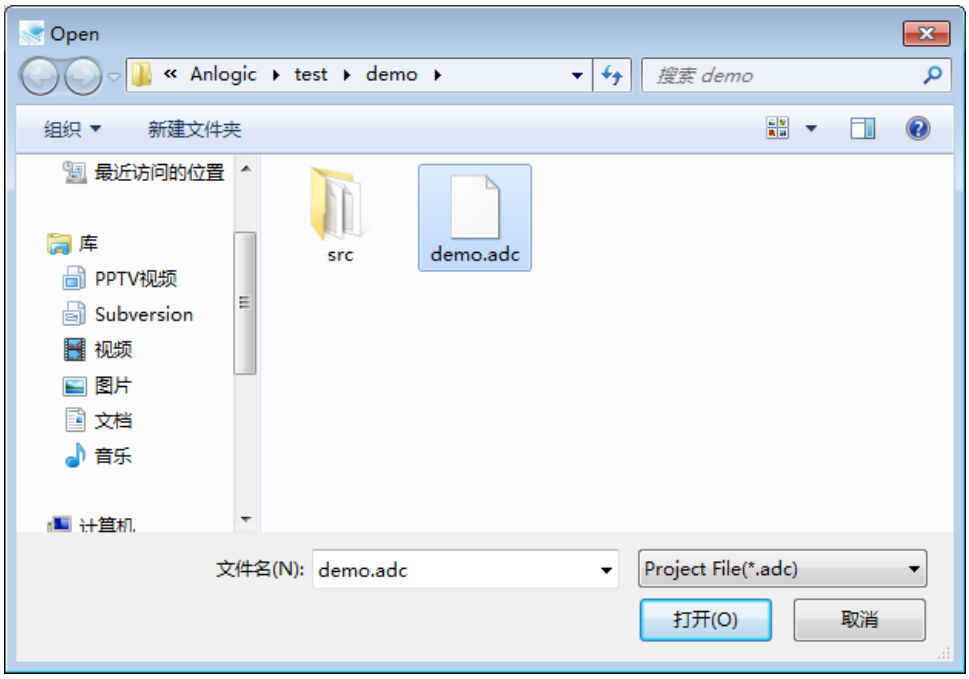
1. Click the New button  and enter the ADC command in the newly opened window.



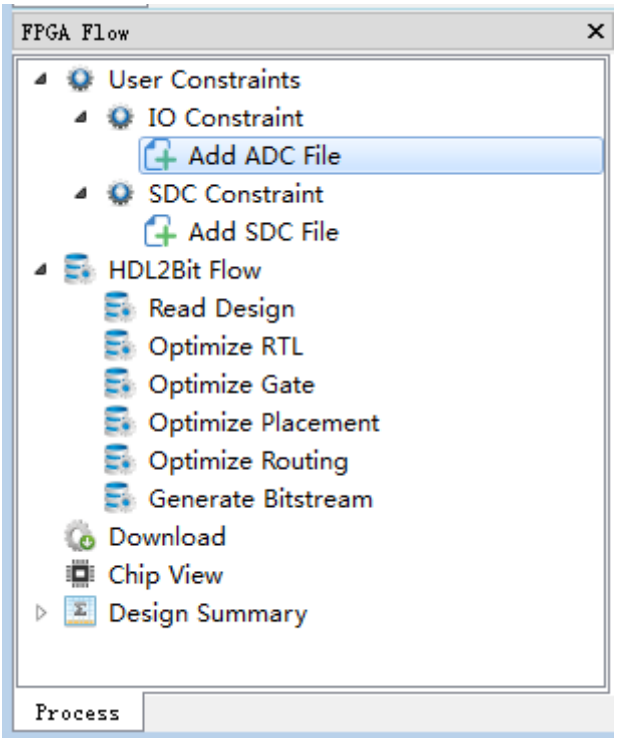
2. Click the Save button to set the file path to the project directory, the file name is demo.adc, and then click Save.

Add ADC file

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



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



4.1.2 Interface settings IO constraints

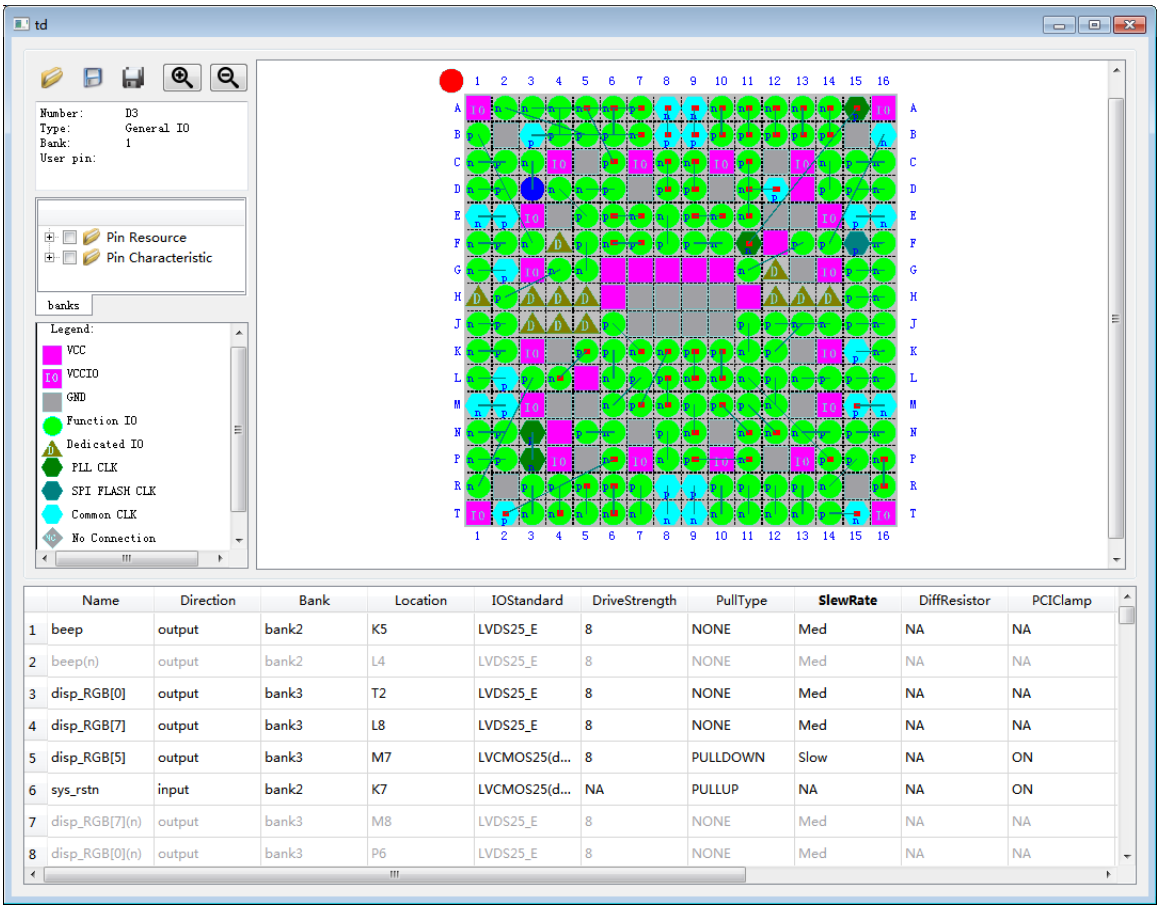
1. Expand User Constraints in the FPGA Flow panel
2. **Double click on IO Constraint**
3. **Set parameters such as Bank, Location, PullType, and IOStandard for each port.**

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

Table 4-1　IO interface configuration properties

category	colour	shape	Features
	Magenta	square	power supply
	gray	square	Ground
	green	Round	Function IO
	Dark yellow	triangle	Dedicated IO
	blue	hexagon	Common CLK
	dark green	hexagon	PLL
	Navy blue	hexagon	Spi flash clock
	brown	triangle	Dedicated IO
	light grey	diamond	No Connection
		Short slash	Differential pair

When the chip package is BGA256, the configuration interface of the IO constraint is as shown in the figure below.

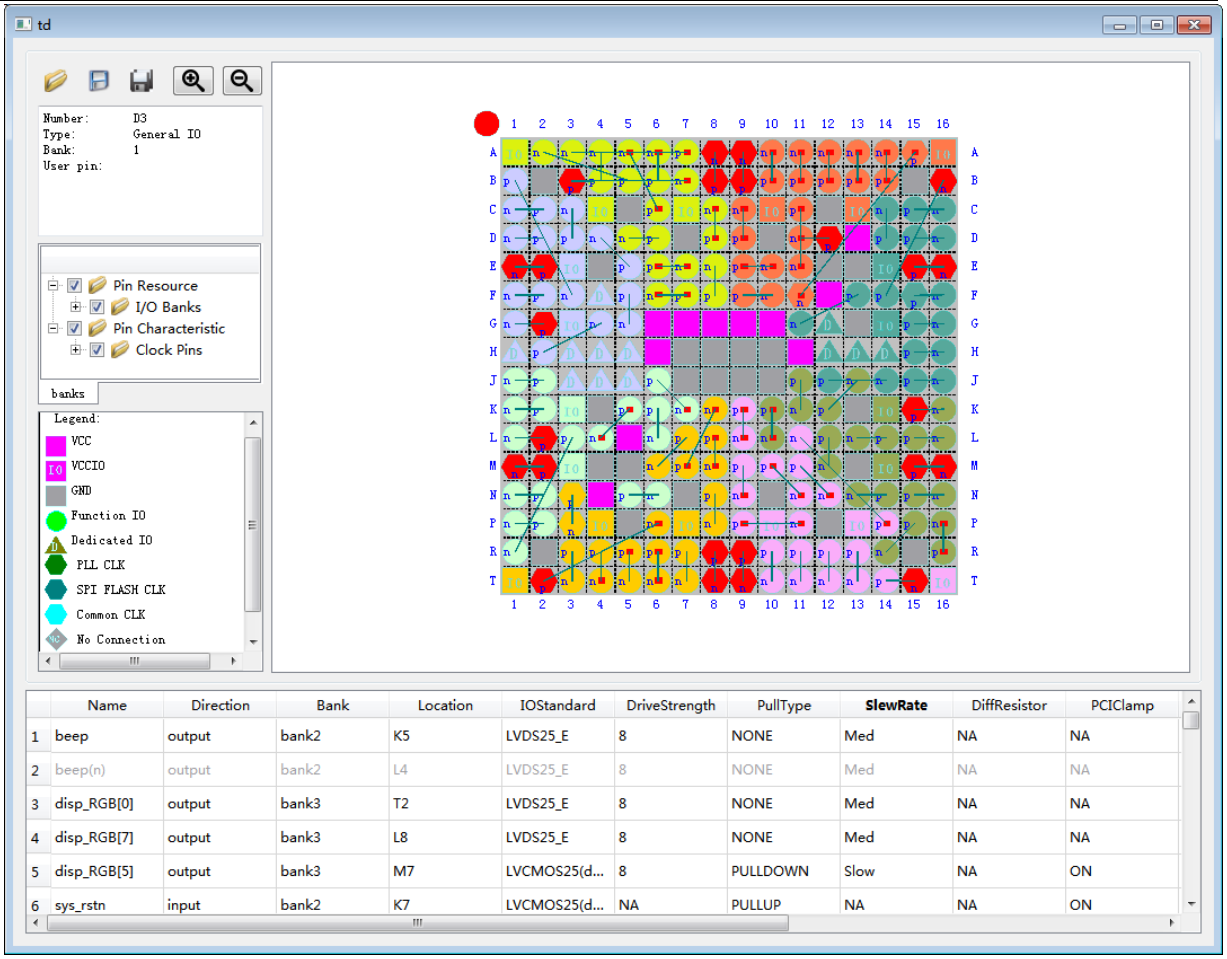


	Name	Direction	Bank	Location	IOStandard	DriveStrength	PullType	SlewRate	DiffResistor	PCIClamp
1	beep	output	bank2	K5	LVDS25_E	8	NONE	Med	NA	NA
2	beep(n)	output	bank2	L4	LVDS25_E	8	NONE	Med	NA	NA
3	disp_RGB[0]	output	bank3	T2	LVDS25_E	8	NONE	Med	NA	NA
4	disp_RGB[7]	output	bank3	L8	LVDS25_E	8	NONE	Med	NA	NA
5	disp_RGB[5]	output	bank3	M7	LVCMOS25(d...	8	PULLDOWN	Slow	NA	ON
6	sys_rstn	input	bank2	K7	LVCMOS25(d...	NA	PULLUP	NA	NA	ON
7	disp_RGB[7](n)	output	bank3	M8	LVDS25_E	8	NONE	Med	NA	NA
8	disp_RGB[0](n)	output	bank3	P6	LVDS25_E	8	NONE	Med	NA	NA

A small red dot in each shape indicates that the IO port is already occupied.

Users can also distinguish different banks by different colors, as shown in the figure below. Among them, red: clock, magenta: power, gray: ground, these three types of ports do not belong to any Bank.

The red origin in the upper left corner corresponds to the small pit on the chip, indicating the starting point of the chip pin.

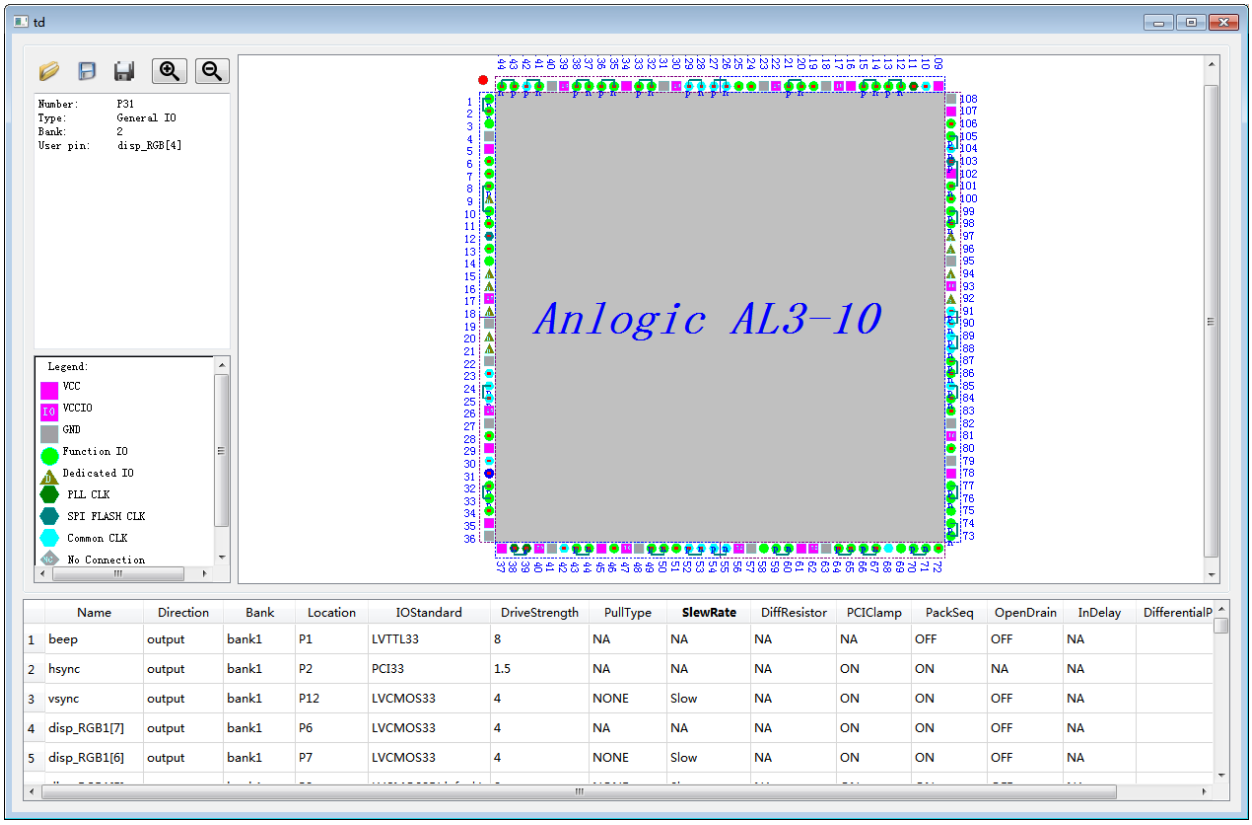


	Name	Direction	Bank	Location	IOStandard	DriveStrength	PullType	SlewRate	DiffResistor	PCIClamp
1	beep	output	bank2	K5	LVDS25_E	8	NONE	Med	NA	NA
2	beep(n)	output	bank2	L4	LVDS25_E	8	NONE	Med	NA	NA
3	disp_RGB[0]	output	bank3	T2	LVDS25_E	8	NONE	Med	NA	NA
4	disp_RGB[7]	output	bank3	L8	LVDS25_E	8	NONE	Med	NA	NA
5	disp_RGB[5]	output	bank3	M7	LVCMOS25(d...	8	PULLDOWN	Slow	NA	ON
6	sys_rstn	input	bank2	K7	LVCMOS25(d...	NA	PULLUP	NA	NA	ON

The short slashes indicate differential pairs, such as B1 (positive) and F3 (negative) in the above figure. If the IOStandard attribute of B1 is selected as LVDS25, the TD automatically assigns the corresponding differential pair pin F3(negative), and the IOStandard attribute of F3 is also LVDS25. 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 attributes are shown in Table 4-1, and the configuration interface is as follows:



	Name	Direction	Bank	Location	IOStandard	DriveStrength	PullType	SlewRate	DiffResistor	PCIClamp	PackSeq	OpenDrain	InDelay	DifferentialP
1	beep	output	bank1	P1	LVTTL33	8	NA	NA	NA	NA	OFF	OFF	NA	
2	hsync	output	bank1	P2	PCI33	1.5	NA	NA	NA	ON	ON	NA	NA	
3	vsync	output	bank1	P12	LVCMOS33	4	NONE	Slow	NA	ON	ON	OFF	NA	
4	disp_RGB1[7]	output	bank1	P6	LVCMOS33	4	NA	NA	NA	ON	ON	OFF	NA	
5	disp_RGB1[6]	output	bank1	P7	LVCMOS33	4	NONE	Slow	NA	ON	ON	OFF	NA	

BANK and LOCATION settings:

When the chip package is different, the number of pins and the pin name are different, and the number of banks is different. Taking the Anlogic AL3-10 series as an example, the IO pins are divided into 8 banks. When setting the pin position, you can specify Bank first, then specify the Location in Bank, or directly specify Location. TD will automatically match the corresponding Bank.

If only Bank is selected and Location is not specified, TD will default to the user selecting the pin with the smallest pin number in this Bank as the location of this pin. When Location is specified as VirtualIO, the port is assigned to a non-physical IO, does not belong to any Bank, and will not be assigned any IO resources during cabling. All LevelIO parameters of VirtualIO cannot be set.

	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	bank1	P1	PCI33	1.5	NONE	Slow	NA
3	vsync	output	bank2	P12	LVCMOS33	4	NONE	Slow	NA
4	disp_RGB1...	output	bank3	P6	LVCMOS33	4	NA	NA	NA
5	disp_RGB1...	output	bank4	P7	LVCMOS33	4	NONE	Slow	NA
6	disp_RGB1...	output	bank5	P8	LVCMOS25(default)	8	NONE	Slow	NA
7	disp_RGB1...	output	bank6	P10	LVCMOS25(default)	8	NA	NA	NA
8	disp_RGB1...	output	bank7	P11	LVCMOS25(default)	8	NA	NA	NA
			bank8						

IOstandard settings:

IOStandard sets the level standard for the IO port. Each bank can be freely set to support the level standard of the device, and different level standards must be the same level in the same bank.
TD provides LVCMOS, LVDS, LVTTL33, PCI33 for users to choose. Among them, LVCMOS has 1.2v, 1.5V, 1.8V, 2.5V, 3.3V voltage options. LVDS is the differential pair input and output.

When the selected IO port is the input signal, only LVDS25, LVDS33, LVPECL33 can be selected.
When the selected IO port is the output signal, LVDS25_E, LVDS33_E, LVPECL33_E can be selected.
The default level standard is LVCMOS25 (default).

Name	Direction	Bank	Location	IOStandard	DriveStrength	PullType	SlewRate	DiffResistor	PCIClamp	PackSeq	OpenDrain	InDelay
hsync2	output	bank4	P72	LVCMOS25(default)	8	NA	NA	NA	NA	ON	NA	NA
clk_vga_25m	output	bank5	P77	LVCMOS33	8	NONE	Slow	NA	ON	ON	OFF	NA
key_in[4]	input	bank5	P80	LVCMOS12	NA	NONE	NA	NA	ON	ON	NA	NONE
key_in[3]	input	bank5	P83	LVCMOS15	NA	NONE	NA	NA	NA	ON	NA	NA
key_in[2]	input	bank5	P84	LVCMOS18	NA	NONE	NA	NA	NA	ON	NA	NA
key_in[1]	input	bank5	P85	LVCMOS25	NA	NONE	NA	NA	NA	ON	NA	NA
key_in[0]	input	bank5	P86	LVCMOS33	NA	NONE	NA	NA	NA	ON	NA	NA
sw[0]	input	bank5	P87	LVTTL33	NA	NONE	NA	NA	NA	ON	NA	NA
				PCI33								
				LVDS25								
				LVDS25_E								
				LVPECL33_E								

DriveStrength settings

DriveStrength sets the drive capability of the IO port. The drive capability of different level standards is different. For example, the drive capability of LVCMOS25 is 4, 8, 12, 16, and the unit is mA.
The smaller the value of DriveStrength, the weaker the drive capability, and the larger the value of DriveStrength, the stronger the drive capability.

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

PullType settings:

PullType Sets the pull-up type of the IO port. There are four options for PullType: PULLUP, PULLDOWN, NONE, KEEPER. Digital circuits have three states: high, low, and high impedance.

When the input is an invalid signal, it can be stabilized by the pull-up (PULLUP) resistor and the pull-down (PULLDOWN) resistor. When KEEPER is selected, the level is held at the last valid value. When the IO port is set to LVDS, the PullType can only be set to None.

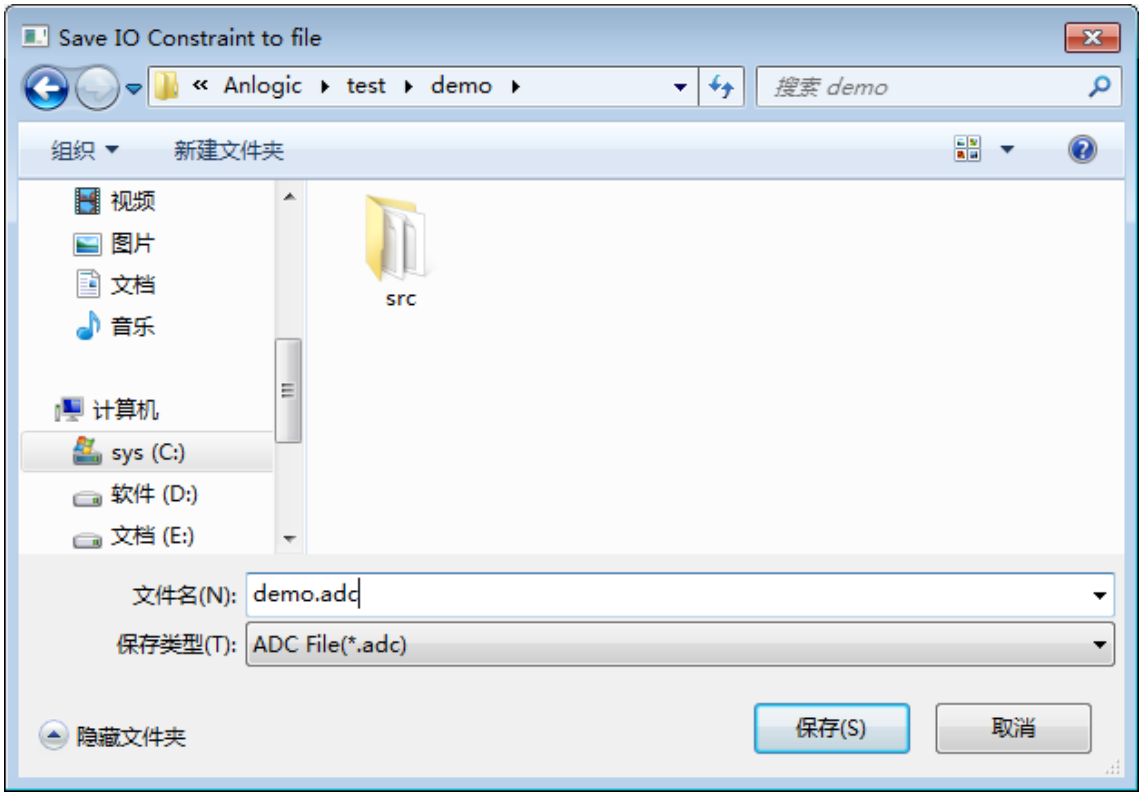
	Name	Direction	Bank	Location	IOStandard	DriveStrength	PullType	SlewRate
1	beep	output	bank4	P54	LVTTL33	8	NONE	Slow
2	hsync	output	bank1	P1	PCI33	1.5	NONE	Slow
3	vsync	output	bank1	P12	LVCMOS33	4	PULLUP	Slow
4	disp_RGB1...	output	bank1	P6	LVCMOS33	4	PULLDOWN	NA
5	disp_RGB1...	output	bank1	P7	LVCMOS33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

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

Parameter	Scope of application	Meaning	Optional value
SlewRate	Single-ended output	Output slew rate	Slow, Med, Fast
DiffResistor	Differential input	Differential Termination Resistors 100ohm	100, None
PCIClamp	Single-ended input and output	PCI level standard requires clamping	ON, OFF
PackSeq	input Output	Pack the registers into the pad Auto means to follow the global settings ON, OFF has a higher priority than the global setting	Auto, ON, OFF
OpenDrain	Single-ended output	Turn off the pull-up P tube of the output, then only output 0, logic 1 depends on external pull-up resistor	ON, OFF
InDelay	Input	Adjust input delay	The actual delay depends on the type of chip and IOB
OutDelay	Output	Adjust output delay	The actual delay depends on the type of chip and IOB

When all settings are complete, the user can click the Save button in the upper left corner, enter the name of the file, and click Save.

If an adc file has been added to the project, the information in the adc file will be read when the IO Constraint interface is opened. After the settings are changed, the contents of adc will be updated directly. If the adc file is not added to the project, the adc file will be added directly to the project after saving.

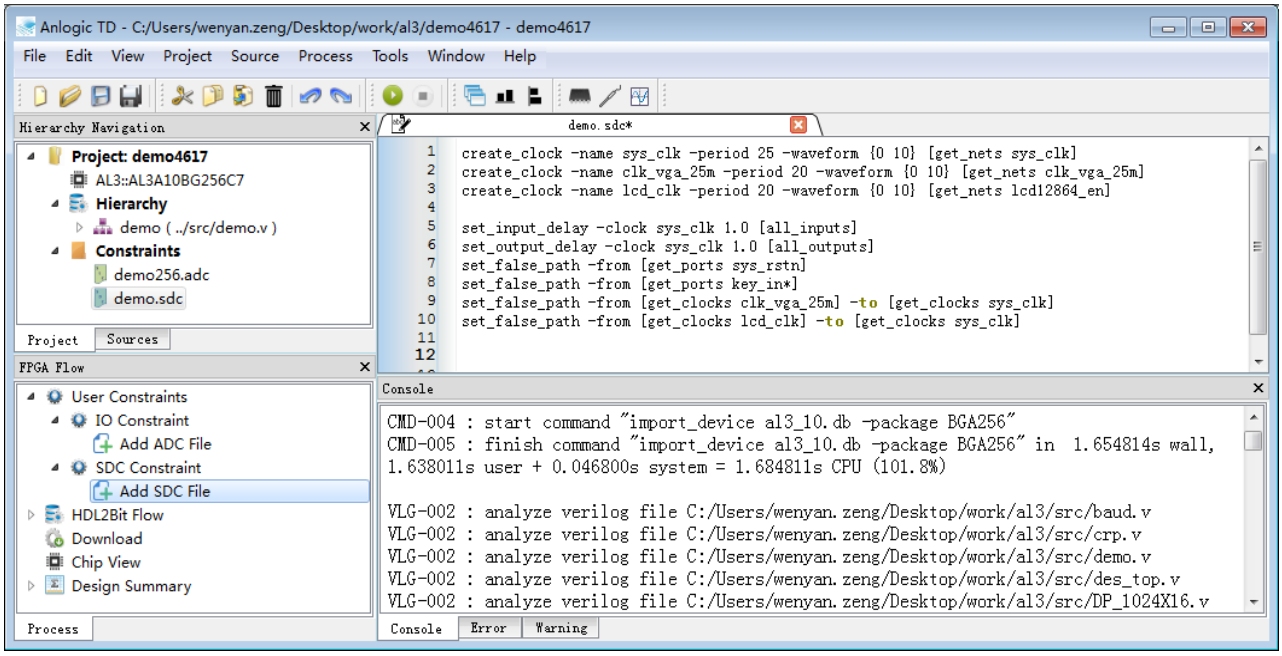


4.2 Timing constraint

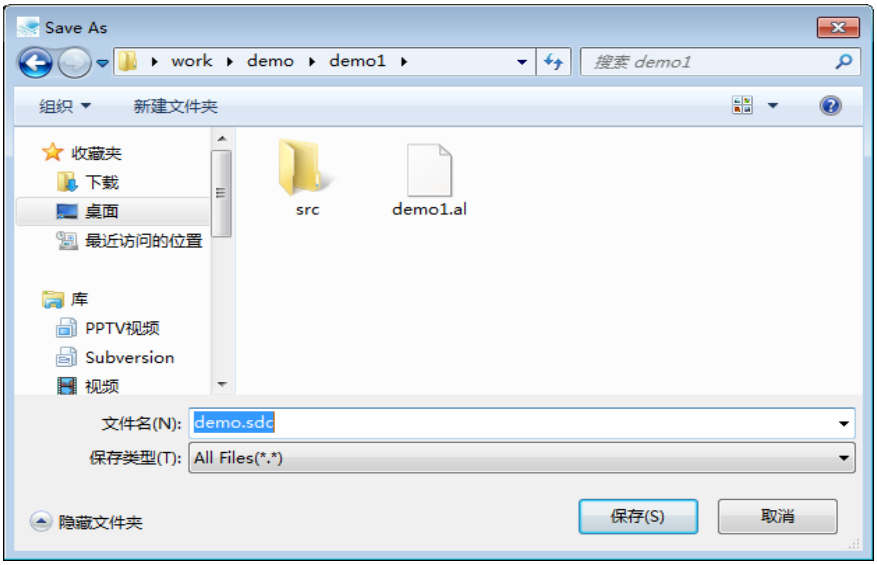
4.2.1 Add timing constraints

Create a new SDC file

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

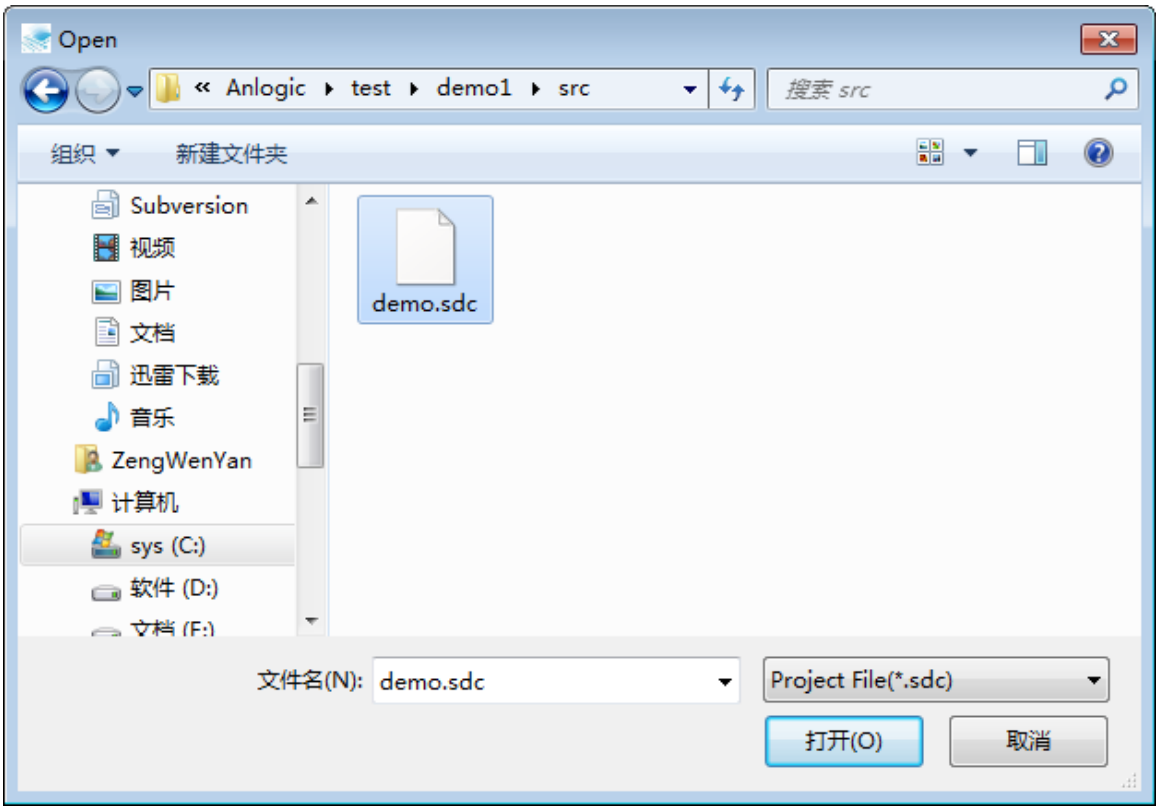


2. Click the Save button to set the file path under the project path, the file name can be set to demo.sdc, and then click Save.

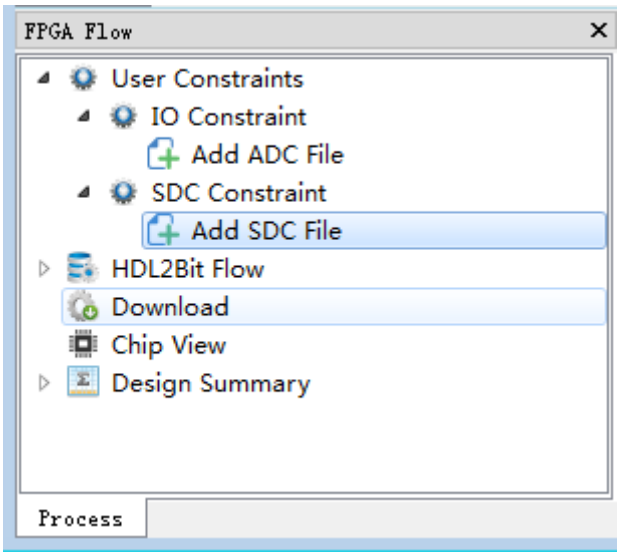


Add an SDC file:

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



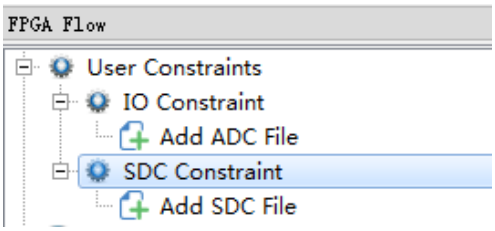
Or expand User Constraints in FPGA Flow, double-click Add SDC File, select demo.sdc as shown above and click to open.



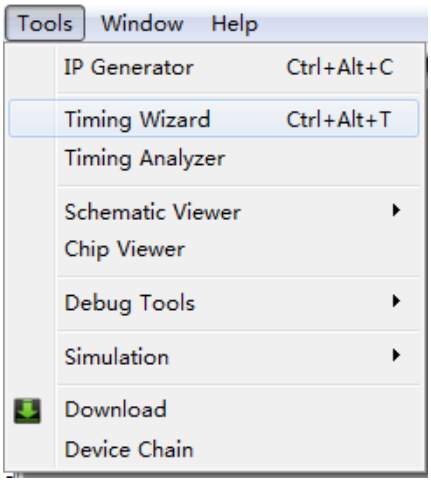
4.2.2 Interface setting timing constraints

There are two ways to open an interface that sets timing constraints:

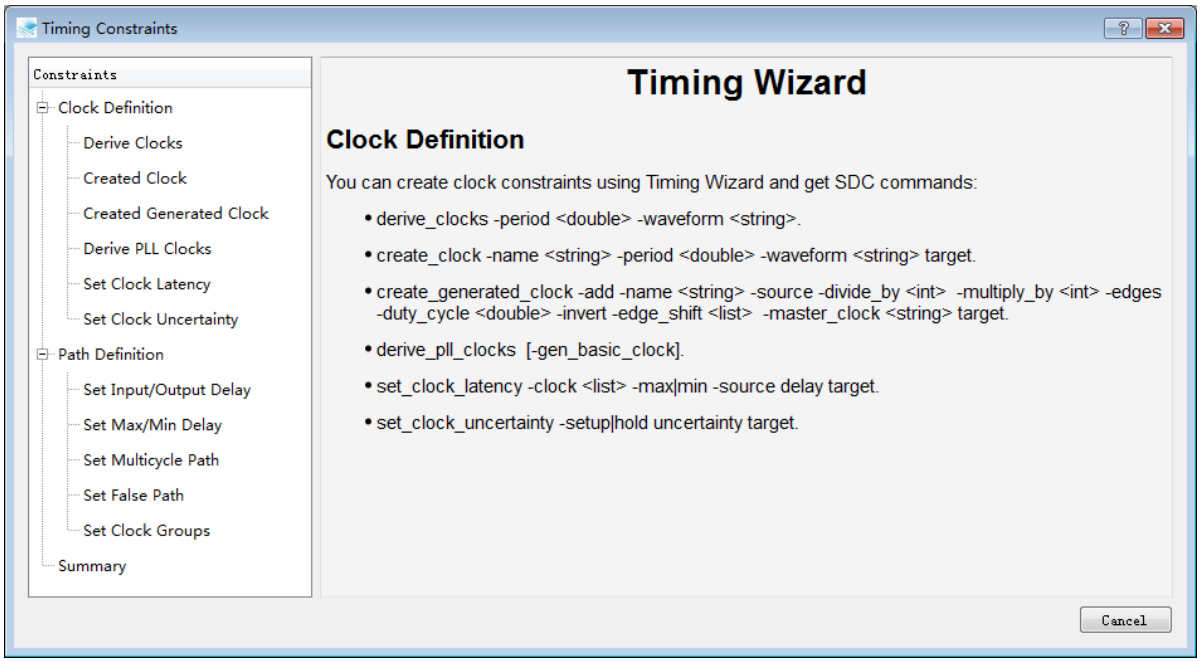
1. In the FPGA Flow panel, expand User Constraints and double-click SDC Constraint;



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



The main interface of Timing Wizard is shown below:



The Timing Wizard consists of three parts:

1. Clock Definition : This section mainly contains SDC commands that create or define clock constraints;
2. Path Definition : This section mainly contains SDC commands that create or define timing path constraints;
3. Summary : This section summarizes all the constraints of the clock and timing paths that have been created.

The following describes in detail how to use each command:

1. Created Clocks

Format: create_clock -add -name <string> -period <double> -waveform <string> target
Definition:
Define a clock: target indicates the clock source, which can be nets or ports, if the target is empty, it means a virtual clock is defined. ;

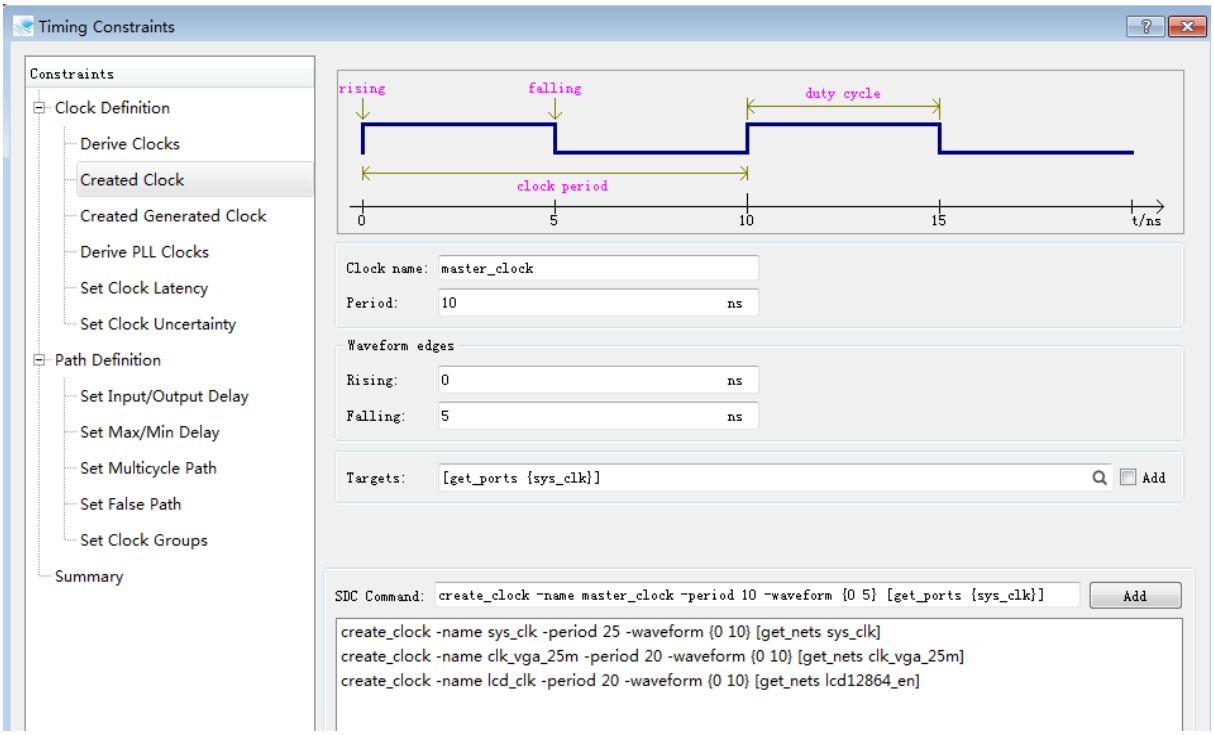
-name specifies the clock name. If the item is empty, the clock name is the first item in the target list;

-period is the period, the option must be specified, and the value needs to be greater than 0;

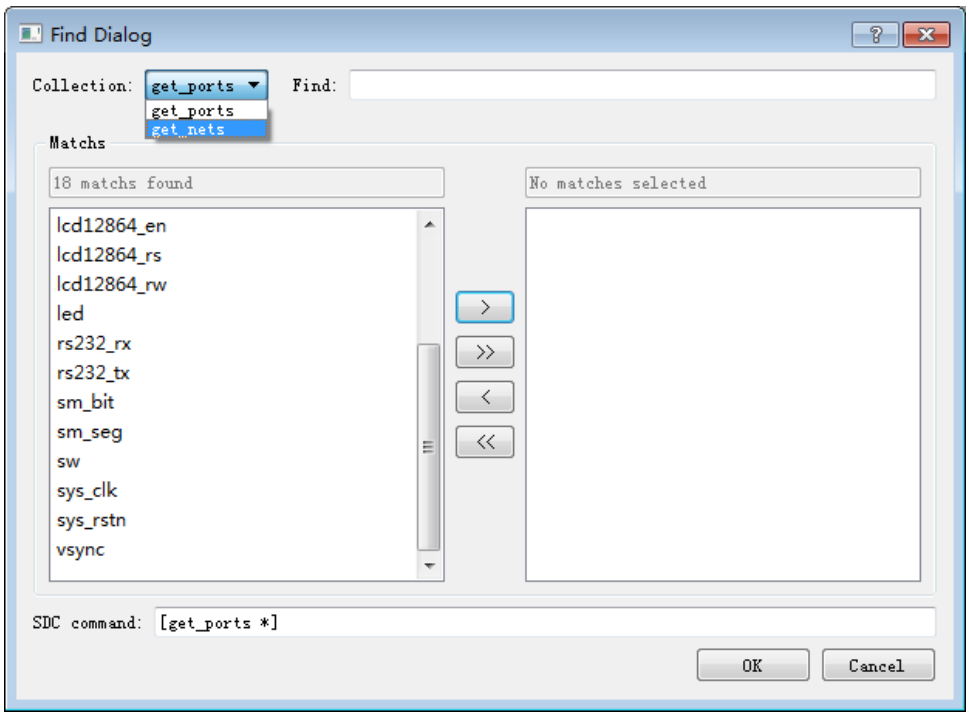
-waveform specifies the first rising edge and The time point of the first falling edge temporarily supports only two clock edges per cycle;

-add indicates that a new clock is added to the pin when the clock has been defined on the target pin, otherwise the overwrite has been There are clocks, mainly for the case of the clock multiplexer.

The parameter settings are as shown below:



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

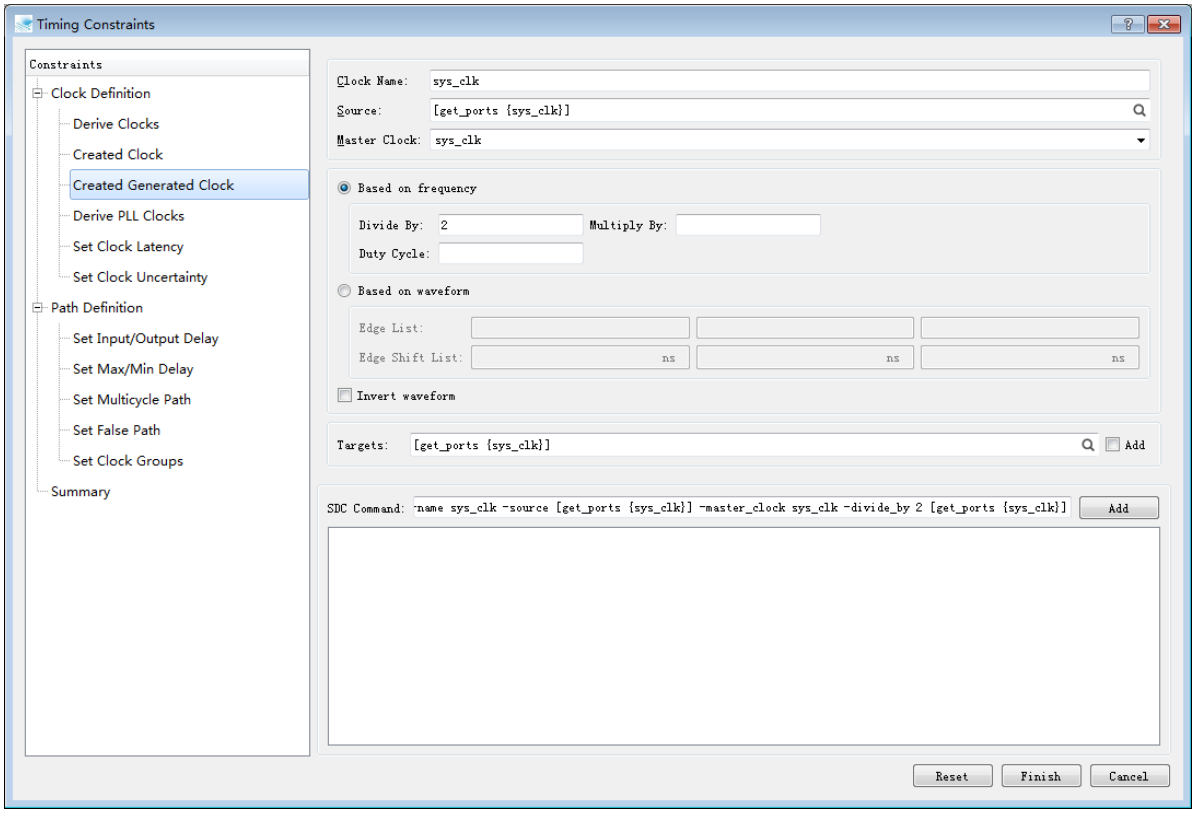


Once the parameters are set, click Add to add the command to the box below, and you can continue to set the command for other parameters, otherwise the command will not be added to the Summary. If the sdc file has been added to the project and the created_clock command already exists in the sdc file, the existing command is also added to the Summary.

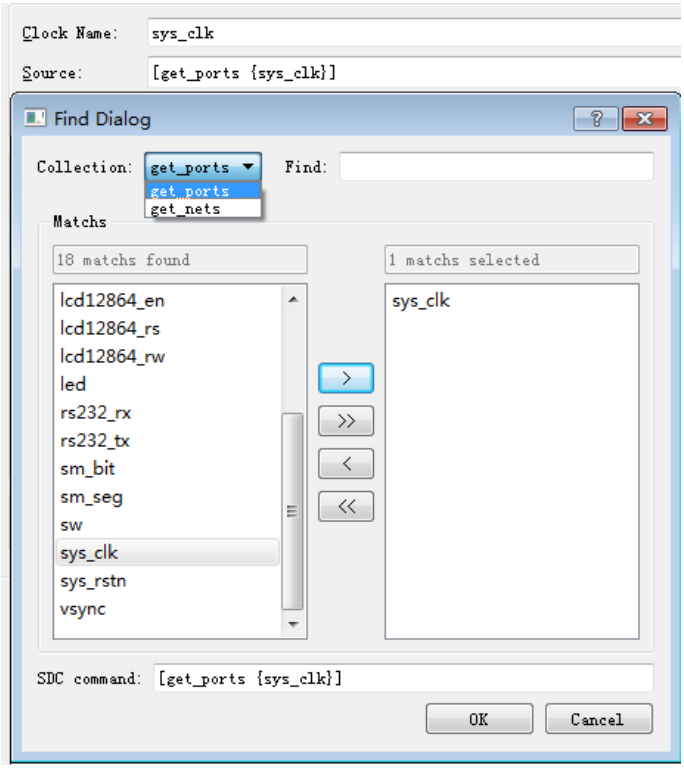
2. Created Generated Clock

Format : create_generated_clock -add -name <string> -source <list> -divide_by <double> -multiply_by <double> -edges <string> -duty_cycle <double> -invert -edge_shift <string> -master_clock <string> target

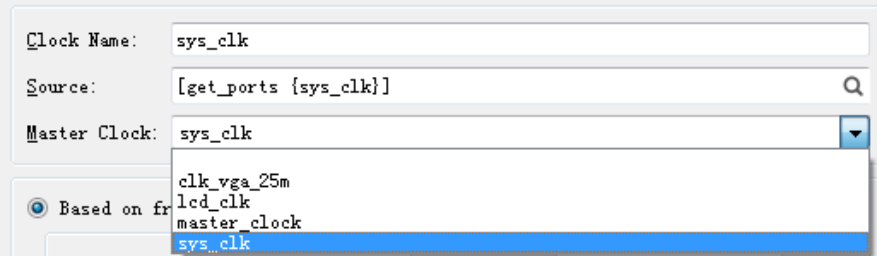
Definition: Define a derived clock that belongs to the clock domain where the master clock is located. It will also have the same start point as the master clock in the timing report.



-source specifies the source point where its master clock is located, which can be a list of nets or ports; click the lookup icon after the Source column to select different types of source:

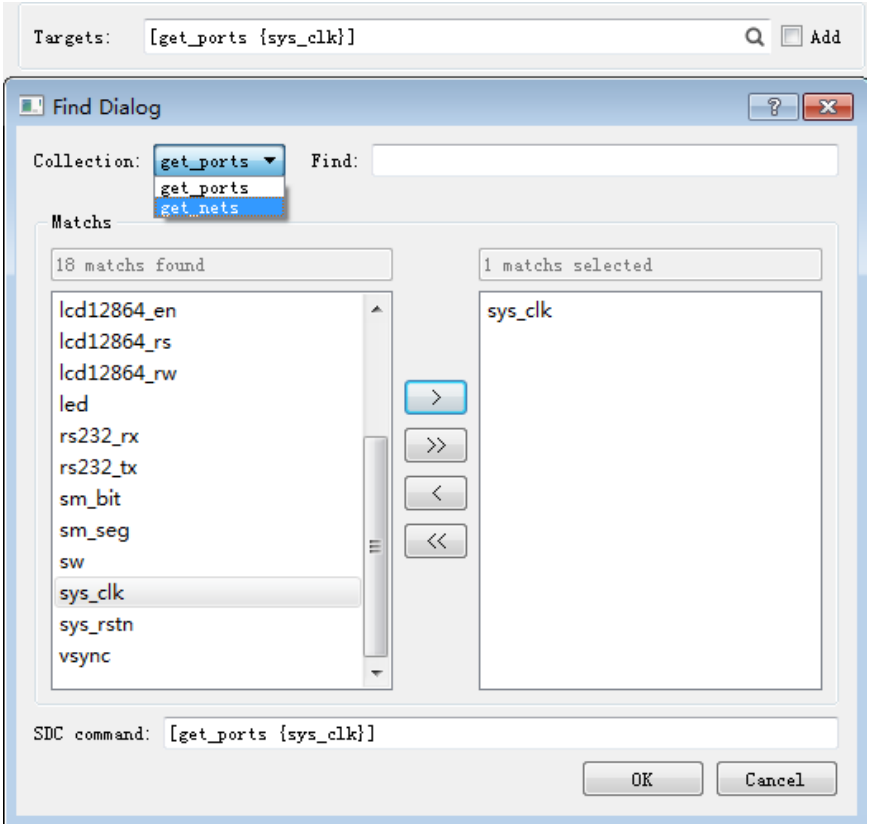


-master_clock specifies the name of the master clock, click the drop-down menu of the Master Clock, and select the created clock;



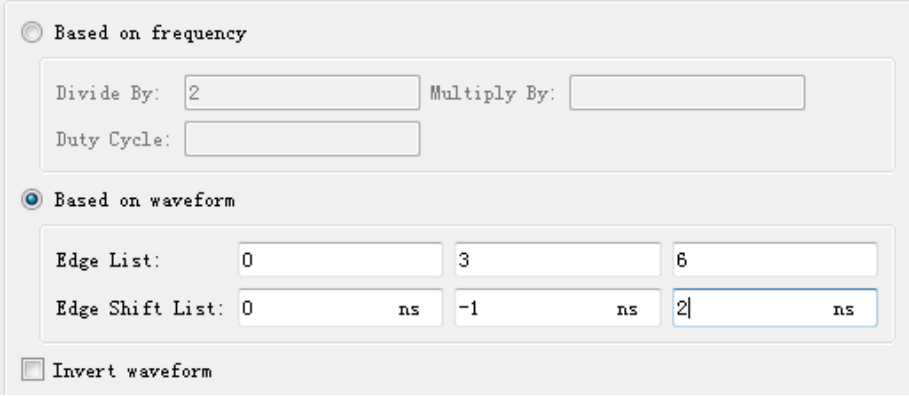
-name specifies the clock name. If the item is empty, the clock is named the first item in the source list; target is the source point of the currently generated clock. -add Description Adds a new clock to this pin if the clock has been defined on the target pin. Otherwise, the current generated clock is ignored, which is different from the master clock definition.

You can also add different types of targets by clicking the Find button in the Targets bar.



The period and waveform of the generated clock are adjusted by the master clock. -divide_by / -multiply_by refers to the frequency divided by / multiplied by the specified multiple. -invert is used with these two options to reverse the clock waveform, and -duty_cycle with -multiply_by Option used to adjust the duty cycle; Based on frequency is selected by default.

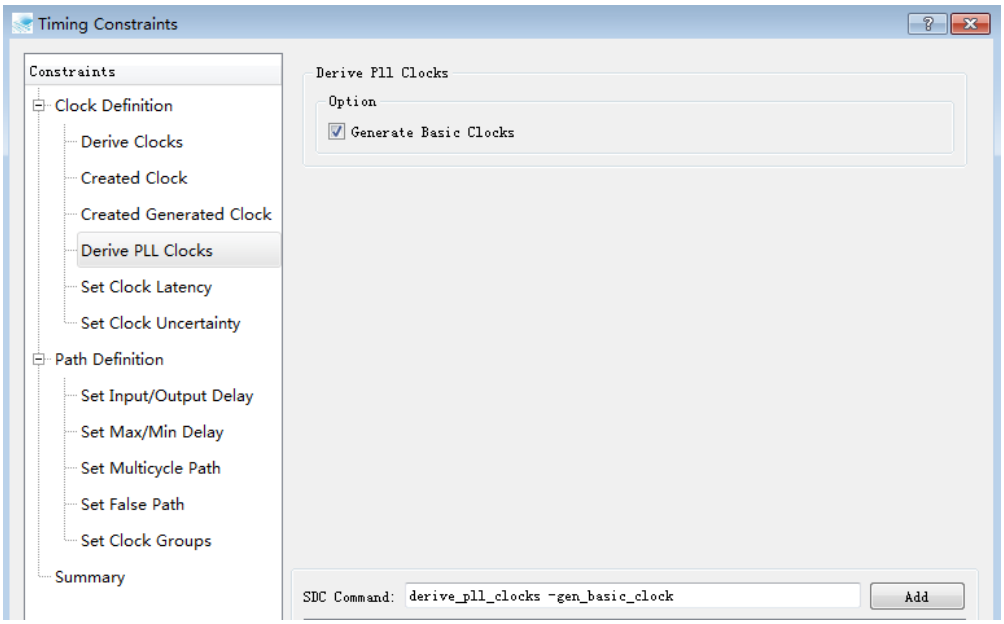
If you select Based on waveform, you can set the edge option. The -edges option contains three integers that specify the first rising edge of the newly generated clock, the first falling edge, and the second rising edge corresponding to the first edge of the source clock; the -edge_shift option specifies the -edges option The offset of the three clock edges, so it will contain 3 positive and negative integers in basic time units (default is ns).



3. Derive PLL Clocks

Format: derive_pll_clocks [-gen_basic_clock]

Definition: Automatically generate clock constraints on all used PLL clkc[x] ports. The frequency and phase of the generated clock will be set exactly according to the parameters inside the PLL.



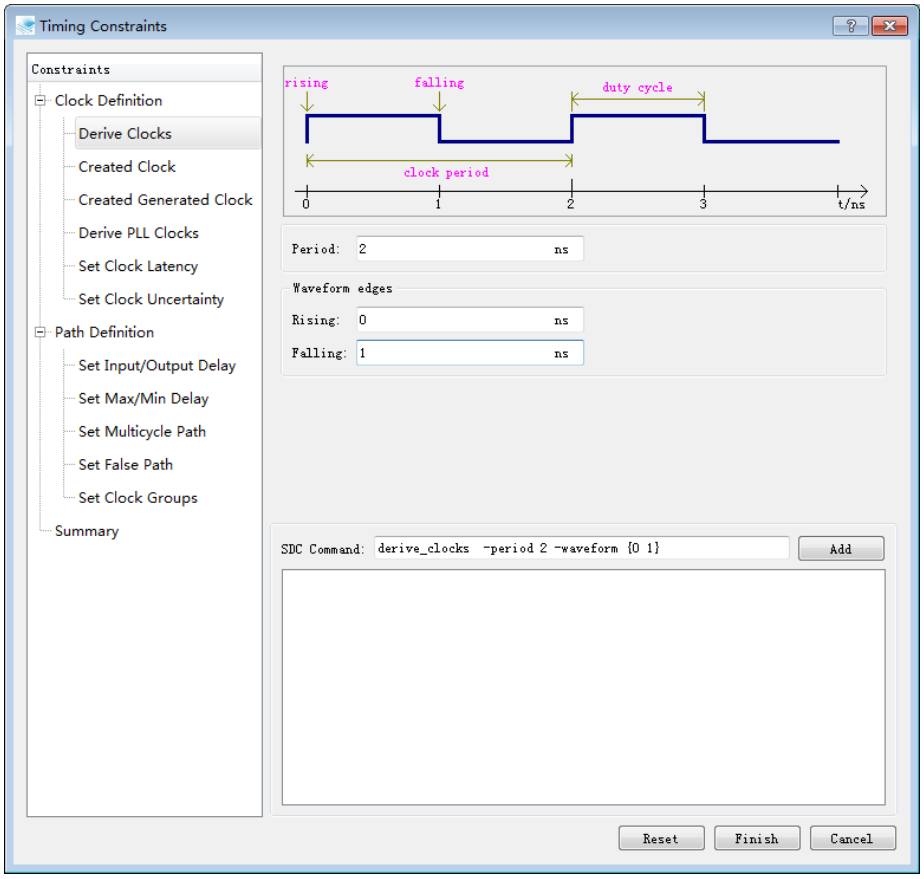
-gen_basic_clock will define the reference clock for the FIN frequency on the corresponding PLL refclk, otherwise it will automatically search for the refclk pin and the clock defined on the connected net. The clock generated by this command will not take effect until flow is running, so it cannot be referenced in subsequent settings of timing wizard.

4. Derive Clocks

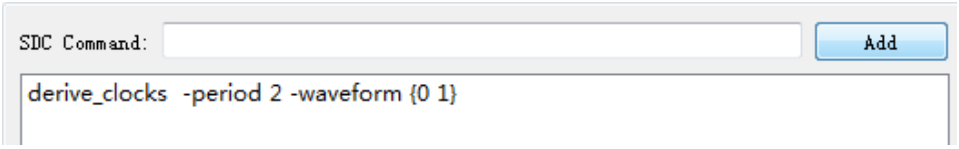
Format: derive_clocks -period <double> -waveform <string>

Definition: Specify a default clock on the clock pin of each undefined clock, -period is the period, this option must be specified, and the value needs to be greater than 0; -waveform specifies the time of the first rising edge and the first falling edge Point, temporarily only supports the case where there are two clock edges per cycle.

The parameter settings are as shown below:



Click Add to add the command to the box below, and you can continue to set the command for other parameters, otherwise the command will not be added to the Summary.



5. Set Clock Latency

Format: set_clock_latency -clock <list> -max -min -source delay

Definition: set the delay of the clock, delay is the value of the delay;

-clock specifies a list of clocks;

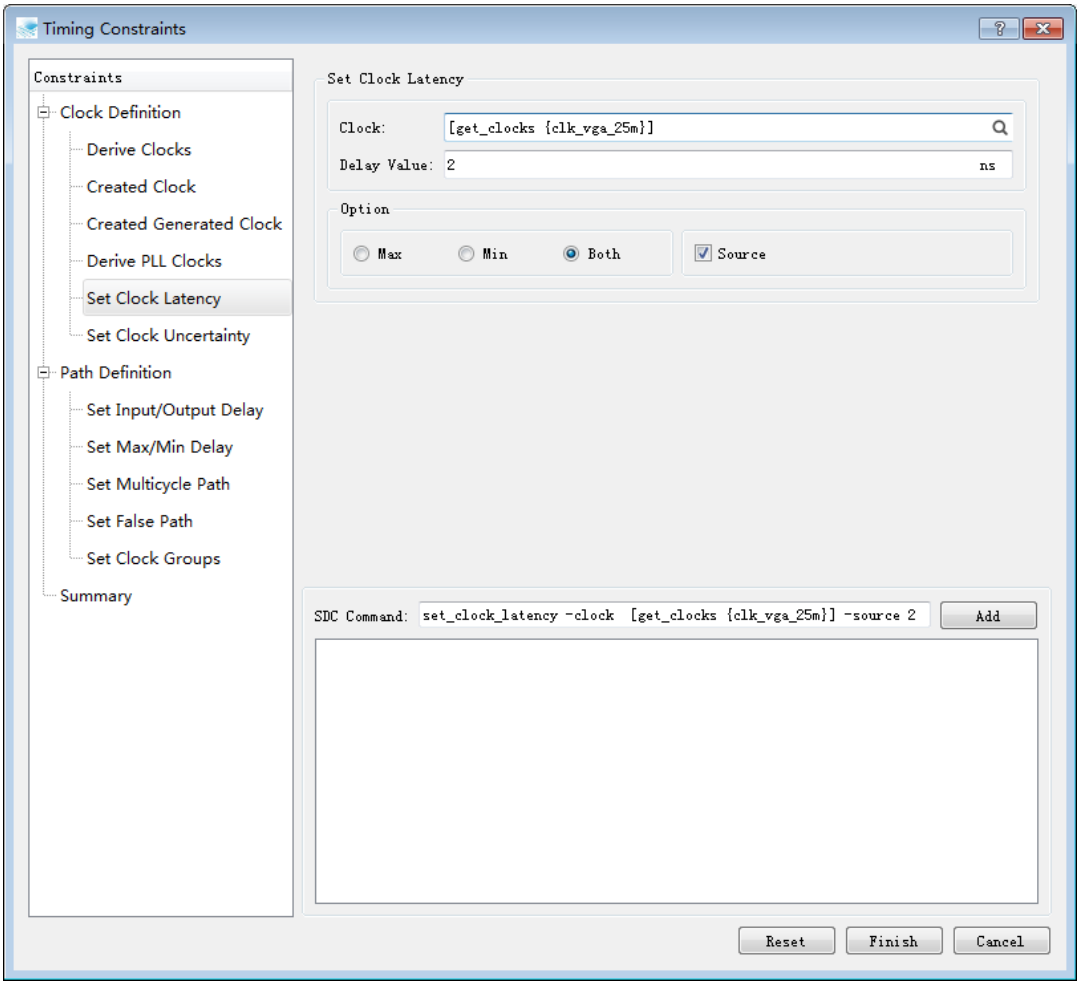
-max/ -min option specifies the maximum/minimum delay specified by this command. The default is the same for both;

-source option indicates that source latency is specified, otherwise it is network latency.

Network latency is replaced by the actual interconnect delay in the timing analysis after routing.

You can specify the clock you have created by clicking the Find button after the Clock column.

Click the Add button to add the command to the Summary.



6. Set Clock Uncertanty

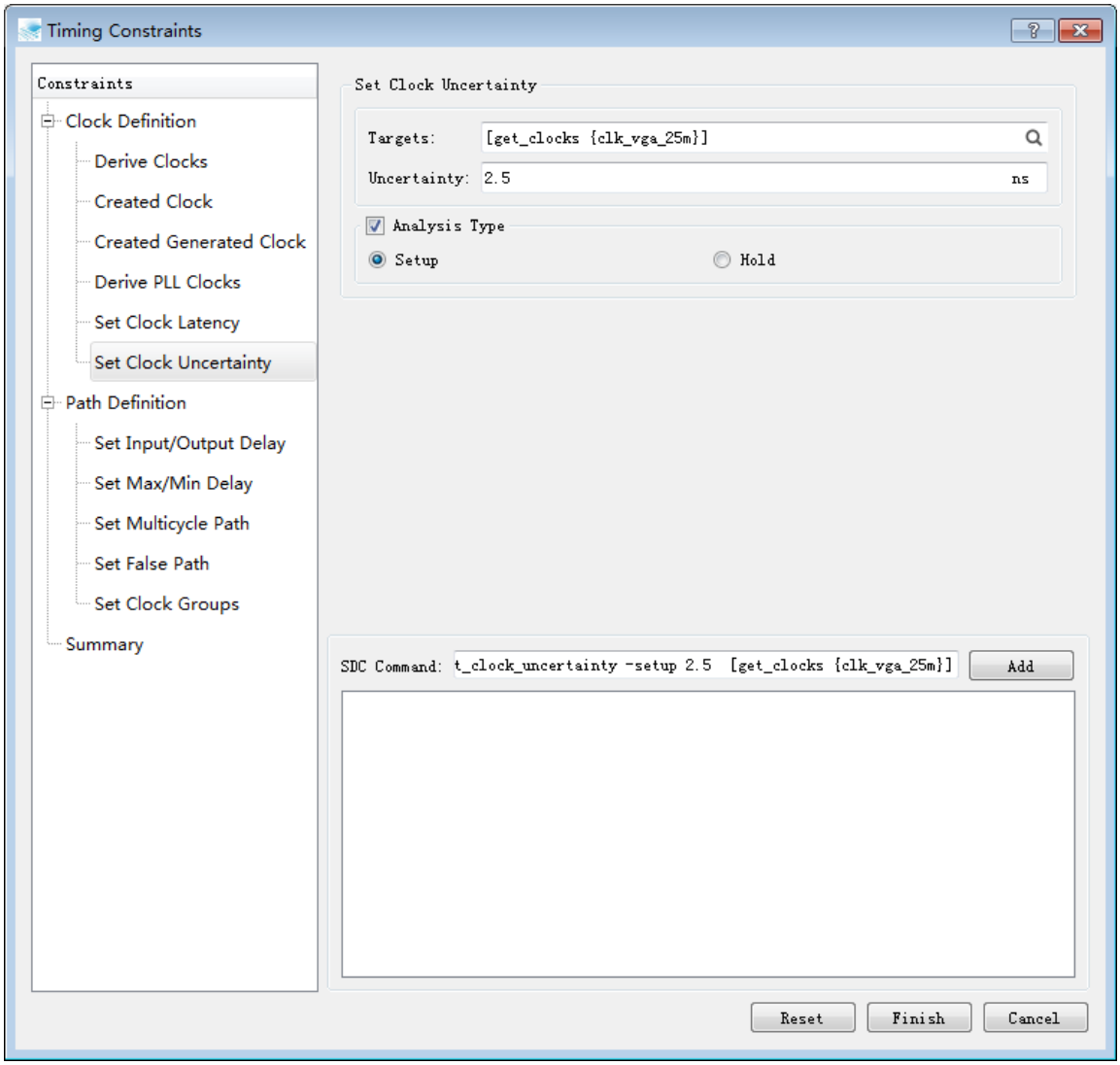
Format: set_clock_uncertainty -setup -hold uncertainty target

Definition: Set the alarm value of the clock. Currently, it only supports the setting of the individual clock itself and does not support the definition of the clock across the clock domain; the target is the list of clock, the uncertainty is the value item;

-setup/-hold option indicates the corresponding Is the value corresponding to the maximum/minimum path timing analysis. If this option is not specified, both checks will take effect at the same time.

Click Add to add the command to the Summary.

The parameter settings are as shown below:

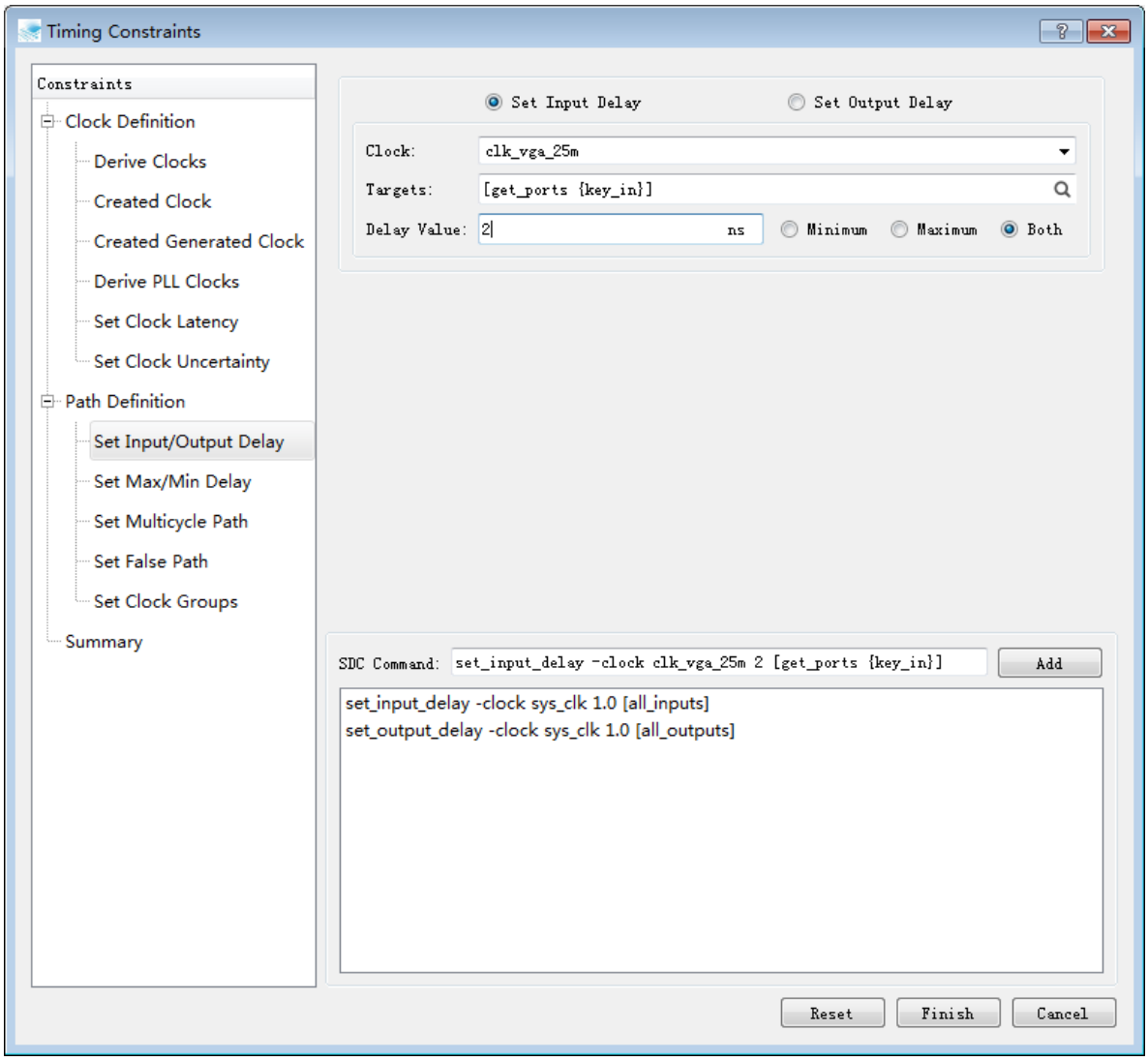


7. Set Input/Output Delay

Format: set_input_delay / set_output_delay -clock <list> -max -min delay target

Definition: Set the delay of the input/output port, the delay item is the delay value, the target can be the object list of pins or ports;
-clock specifies the clock information corresponding to the delay;
-max/-min option specifies the time. The value set by the command is the maximum/minimum delay, and the default is the same.

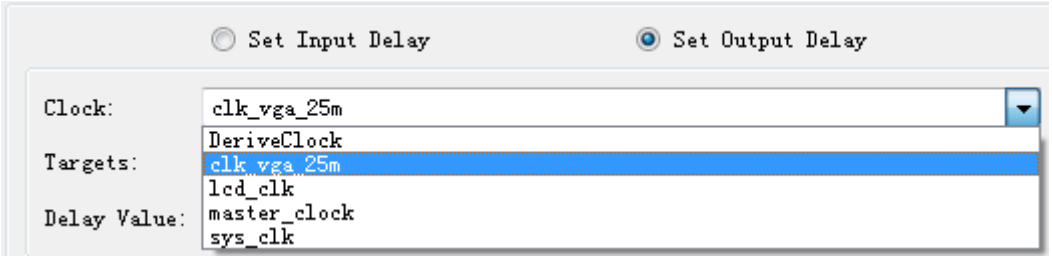
The parameter settings are as shown below:



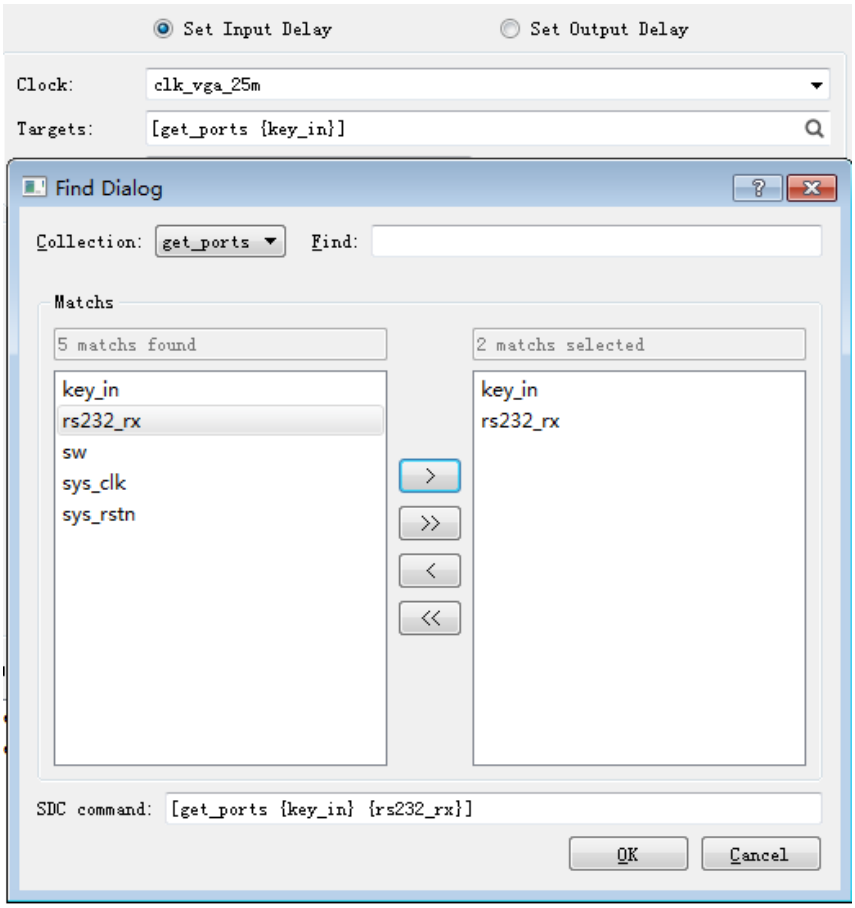
The default is to set the input delay related parameters;

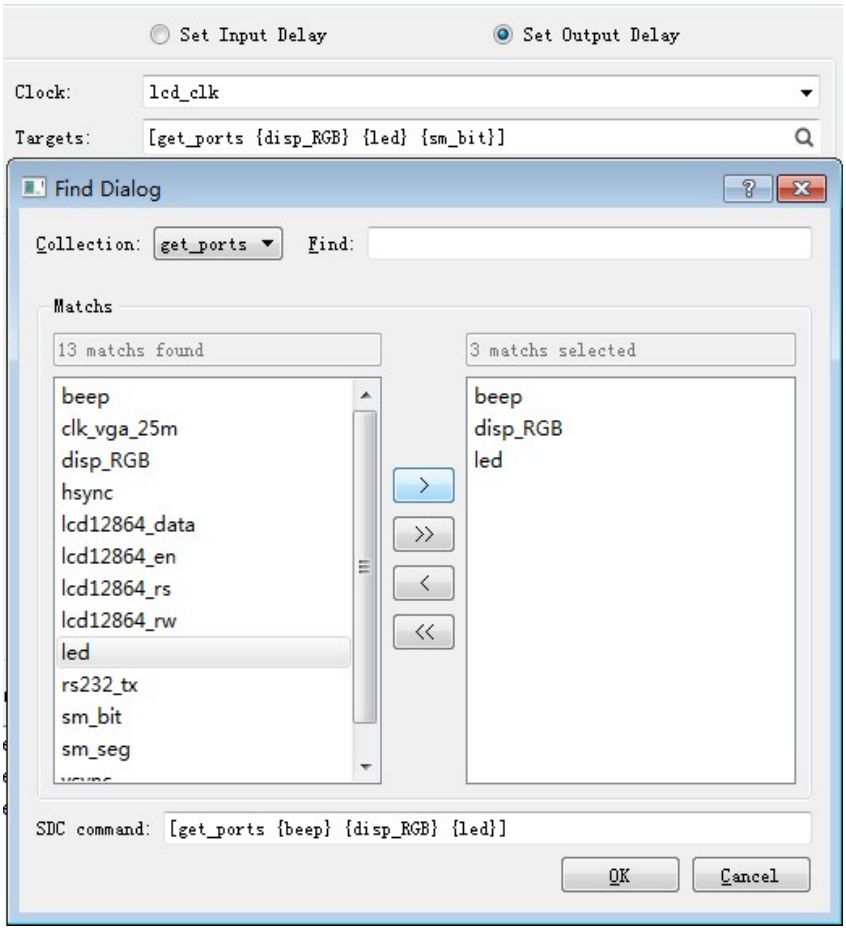
To set the output delay related parameters, you need to select Set Output Delay, and the parameter setting is the same as input delay.

You can specify the clock you have created by clicking the drop-down menu in the Clock column.

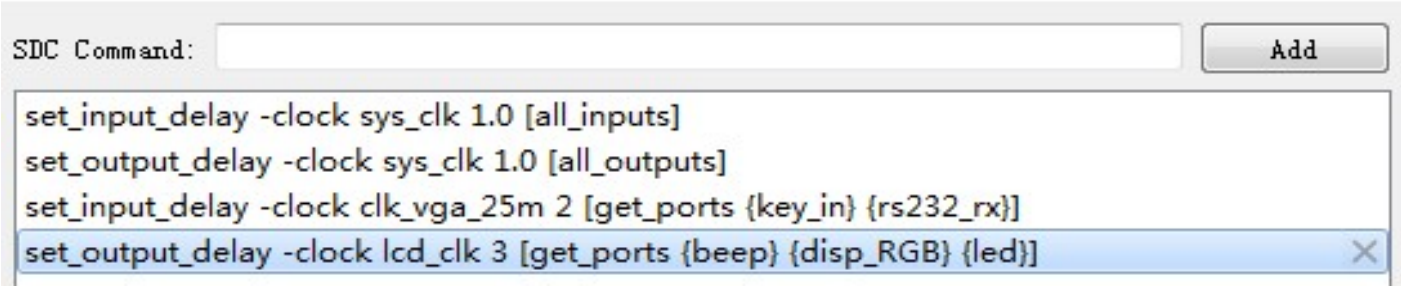


Click the Find button in the Targets bar to specify the corresponding ports. If you select Set Input Delay, only input/inout ports are visible. If you select Set Output Delay, only output/inout ports are visible.





Click Add to add the command to the Summary.



To delete an already generated SDC Command, select it in the Summary and click the delete button "X" at the end of the line.

8. Set Max/Min Delay

Format: set_max_delay / set_min_delay -from <list> -to <list> -through <list> delay

Definition: Set the maximum/minimum delay allowed for the timing path. The delay term is the delay value;

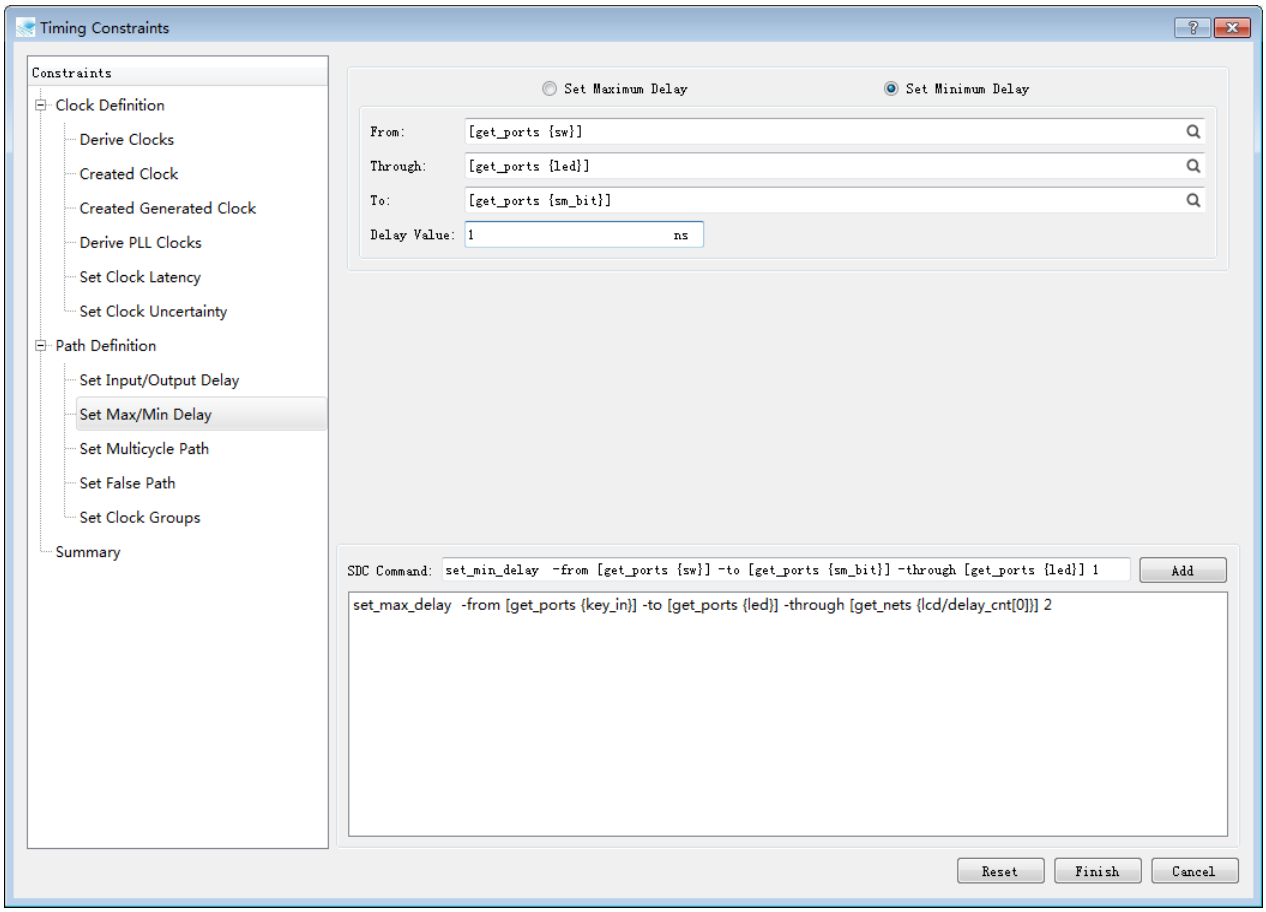
-from must be the starting point of the timing path, ie the list of inputs;

-to must be the end point of the timing path, ie the list of outputs; The through option can be nets, a list of ports that specifies the intermediate point through which the timing path must pass.

When there are multiple through options, the target timing path must pass through each intermediate point in turn.

The parameter settings are as shown below:

The default is Set Maximum Dealy. To set the Minimum Delay, select Set Minmum Delay, and the parameters are the same.



Click Add to add the set commands to the Summary.

9. Set Multicycle Path

Format: set_multicycle_path -setup -hold -start -end -from <list> -to <list> -through <list> multiplier

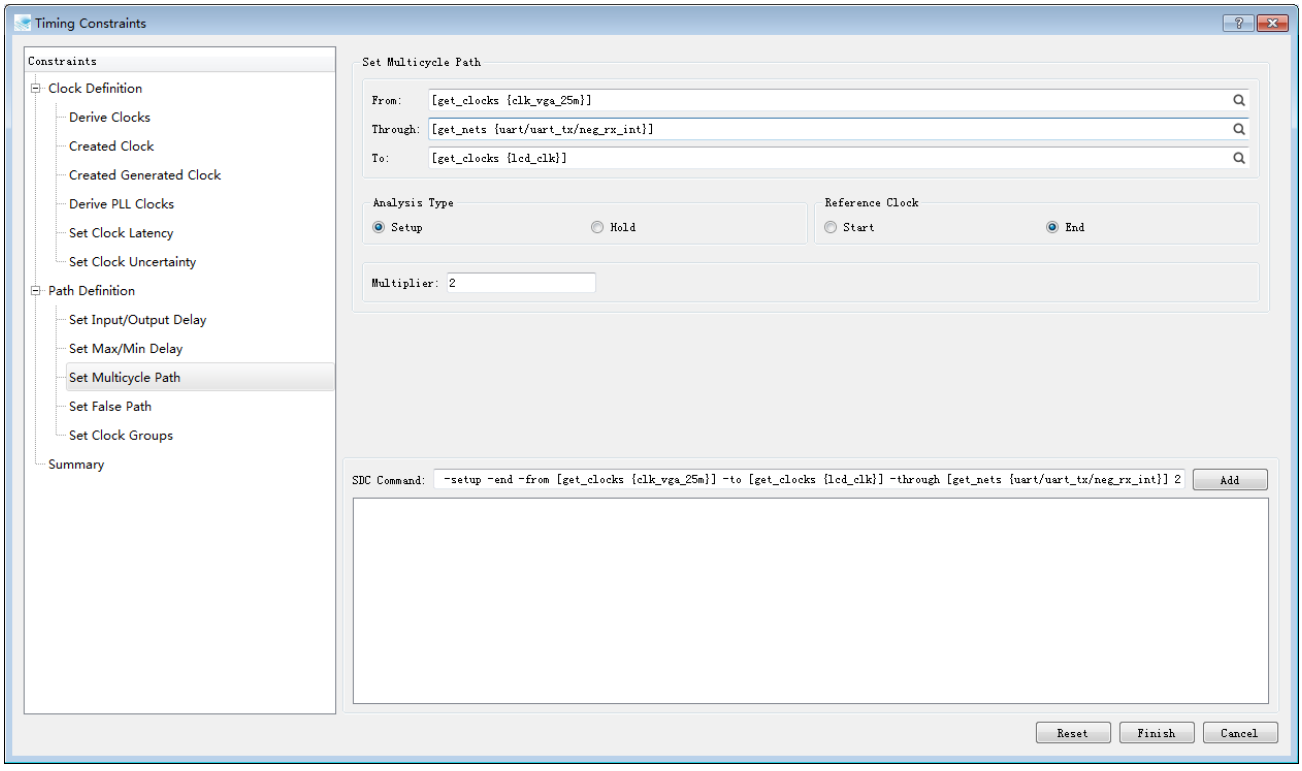
Definition: Set the timing path that allows multiple clock cycle delays, the multiplier term is the number of clock cycles;
-setup/-hold option sets the timing path type that the command constrains, and setup check will allow timing when only the -setup option is used The path uses up to N clock cycles, but at the same time hold check becomes the required timing path for a minimum of N-1 clock cycles. When only the -hold option is used, the hold check constraint is set to N without affecting the constraints of the setup check.

In the normal usage scenario, in order to allow setup check to use multiple clock cycles without affecting the hold check, two commands are required to be used, that is, set a set constraint of N cycles, and then match a hold of N-1 cycles. constraint.

-start/-end is the option used across clock domains, specifying the delay to the period corresponding to the launch/capture clock. By default, setup check uses the capture clock and hold check uses launch clock.

-from is the starting point of the time series path, which can be a list of clocks or inputs,
-to is the end point of the time series path, can be a list of clocks or outputs, and
-through specifies the intermediate point through which the time series path must pass, but is ports or nets list of.

The parameter settings are as shown below:



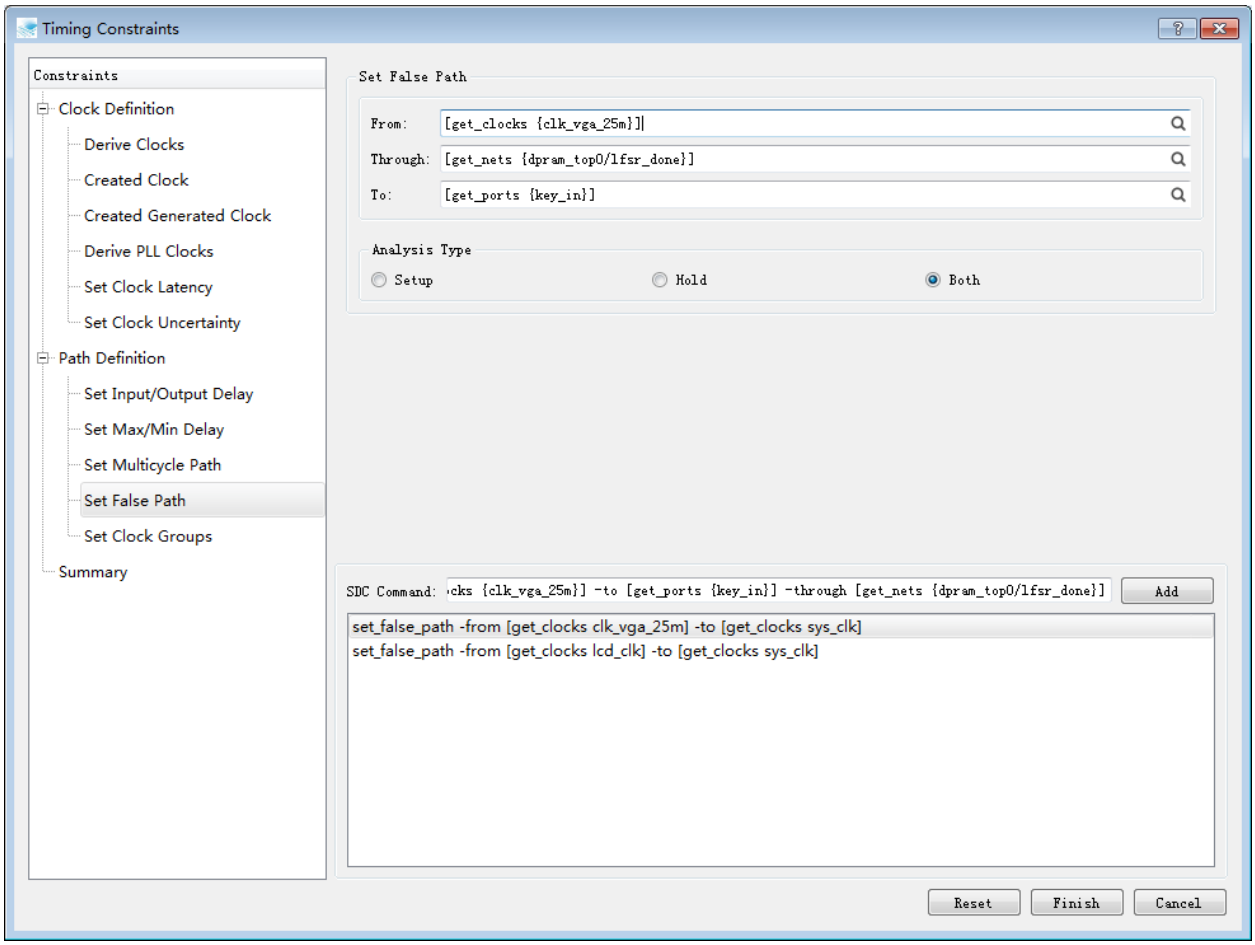
Click Add to add the generated command to the Summary.

10. Set False Path

Format: set_false_path -setup -hold -from <list> -to <list> -through <list>

Definition: Set the timing path to be a fake path and therefore not perform timing analysis;
-setup/-hold option specifies that the target path is not analyzed during setup/hold check.
-from is the starting point of the time series path, which can be a list of clocks or inputs,
-to is the end point of the time series path, can be a list of clocks or outputs,
-through specifies the intermediate point through which the time series path must pass, but is ports or nets list of.

The parameter settings are as shown below:



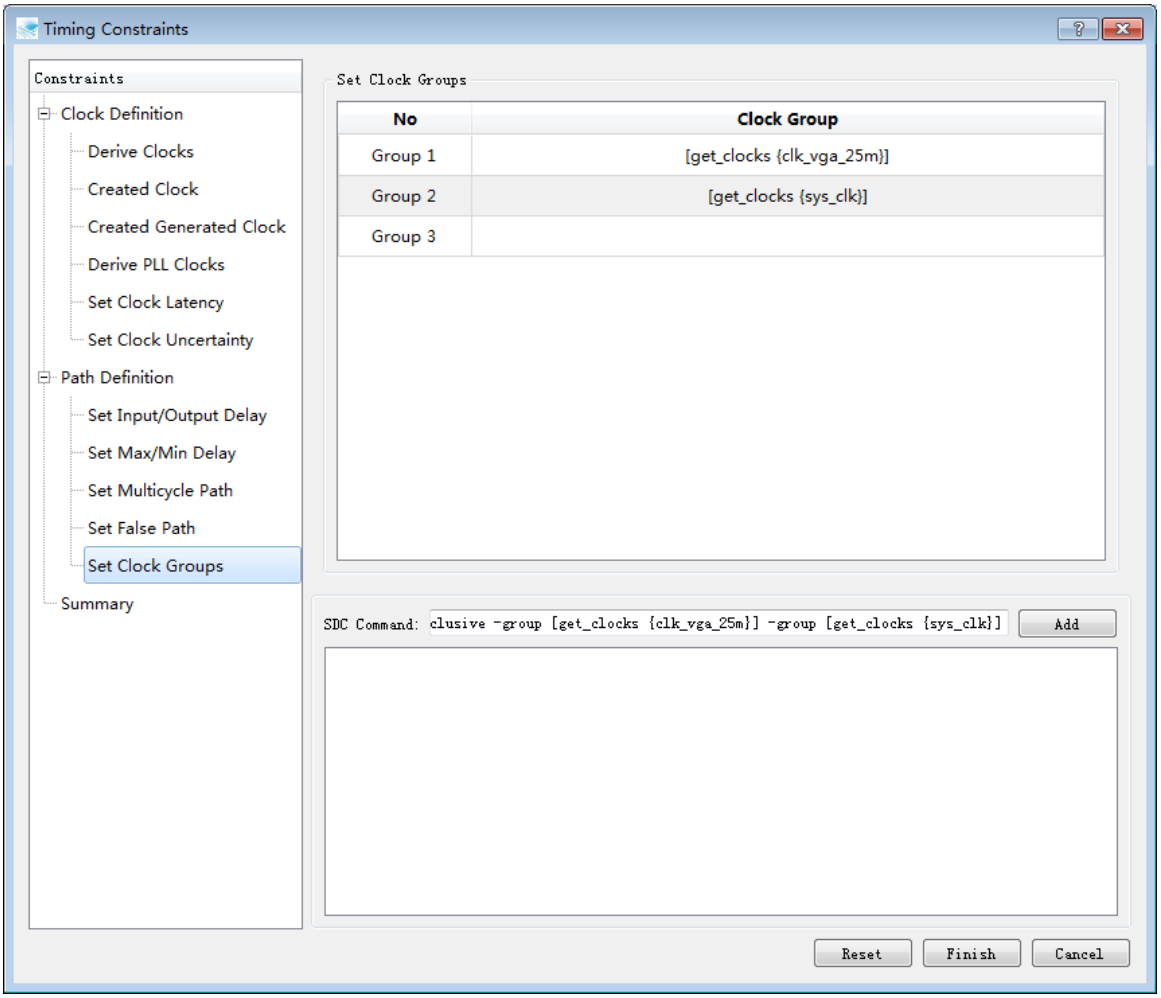
Click Add to add the generated command to the Summary.

11. Set Clock Groups

Format: set_clock_groups -exclusive -asynchronous -group <list>

Definition: Sets the grouping for the clock domain and does not analyze the timing path across the clock group. In general,
-exclusive option means that these clock groups do not logically appear at the same time, and
-asynchronous means a completely irrelevant clock, but in terms of implementation and effect, the two options are the same, the command will be at all
-group lists the set of false path constraints between the clocks.

The parameter settings are as shown below:

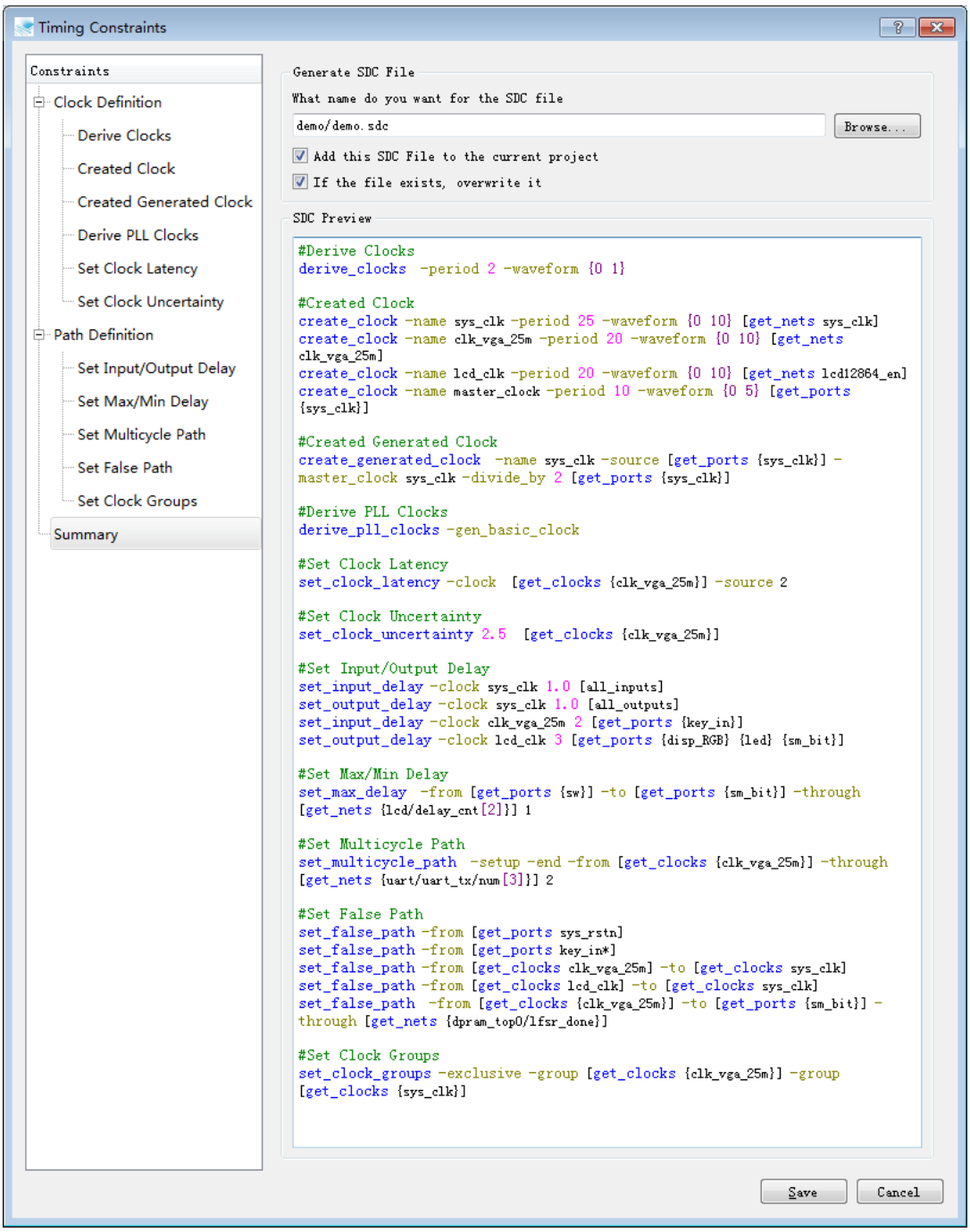


Click Add to add the generated command to the Summary.

12. Summary

Click Add to add the generated command to the Summary.

Specify the path to save the file and choose whether to add it to the current project. Click Save to complete the settings for each command. Note that if the sdc file has been added to the original project, please carefully select "If the file exist, overwrite it". When this option is checked, the original file will be replaced.



5 HDL2Bit Process

After entering the design source file and constraint file, the next step is to enter the HDL2Bit design implementation process. The HDL2Bit process includes Design Read, RTL Level Optimizer, Optimize Gate, Optimize Placement, Optimize Routing, and Generate Bitstream. step.

In most cases, the user simply double-clicks HDL2Bit and the software automatically runs the entire process. Users can also use the Run, Rerun, Stop in the Process drop-down menu to control. Run and Stop in the Process menu have corresponding buttons (and) in the navigation bar, and the user can click directly. There is also a Properties column in the Process menu. Properties provides detailed parameter control options for the main steps in the HDL2Bit process. The user can fine-tune the entire running process. The Global Option parameters are set as follows:

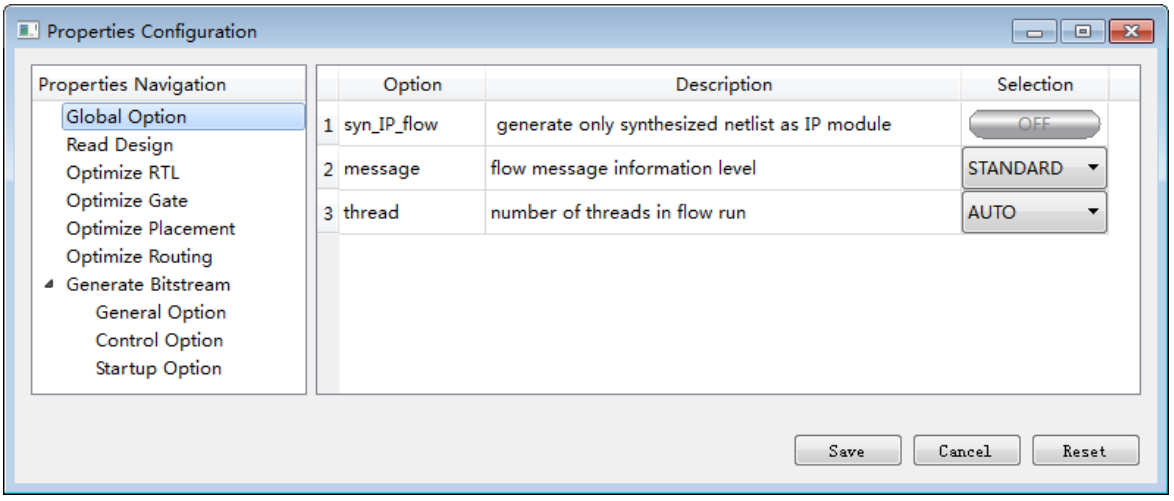


表 5-1 Global Option

Property	Comments	Default
syn_IP_flow	Generate synthesizable IP modules	OFF
message	Redundancy level of output information	STANDARD
thread	Number of threads when running HDL2Bit	AUTO

5.1 Read in file

This step analyzes the correctness of the syntax semantics of the user source file and produces the original behavioral level circuit structure.

1. Expand HDL2Bit in the FPGA Flow panel
2. Double-click Read Design, or right-click on Read Design and select run
3. Parameter configuration

In the menu bar, expand Process→Properties, the Properties Configuration window pops up, select Read Design, and the user can customize the desired function as needed.

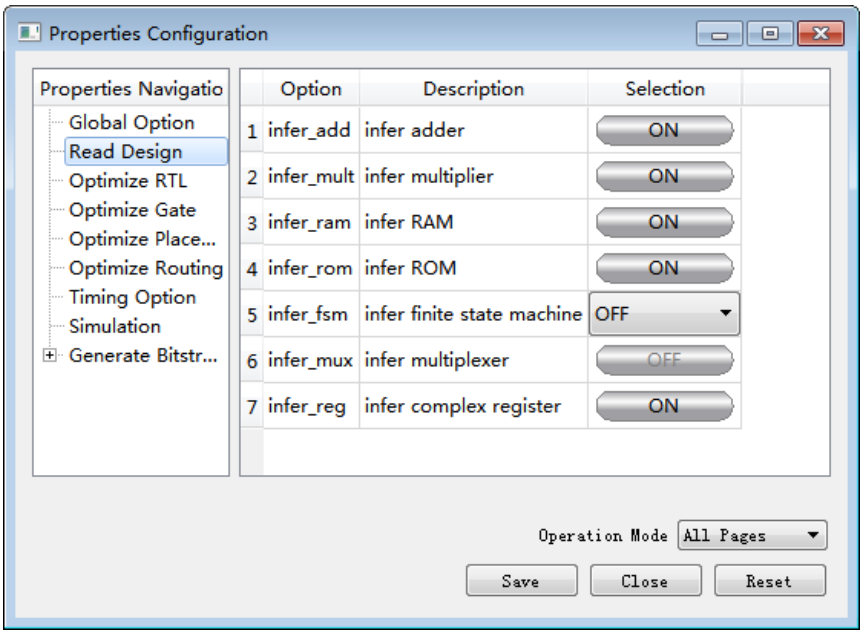


表 5-1　Read Design Properties

Property	Comments	Default
infer_add	Automatic recognition behavioral level addition description	ON
infer_mult	Automatic recognition of behavioral multiplication description	ON
infer_ram	Automatic recognition of behavioral level RAM Description	ON
infer_rom	Automatic recognition of behavioral level ROM description	ON
infer_fsm	Automatic identification of behavioral finite state machine description	OFF
infer_mux	Automatic identification of behavioral multiplex selector description	OFF
infer_reg	Automatic identification of behavioral level complex register descriptions	ON

5.2 RTL level optimization

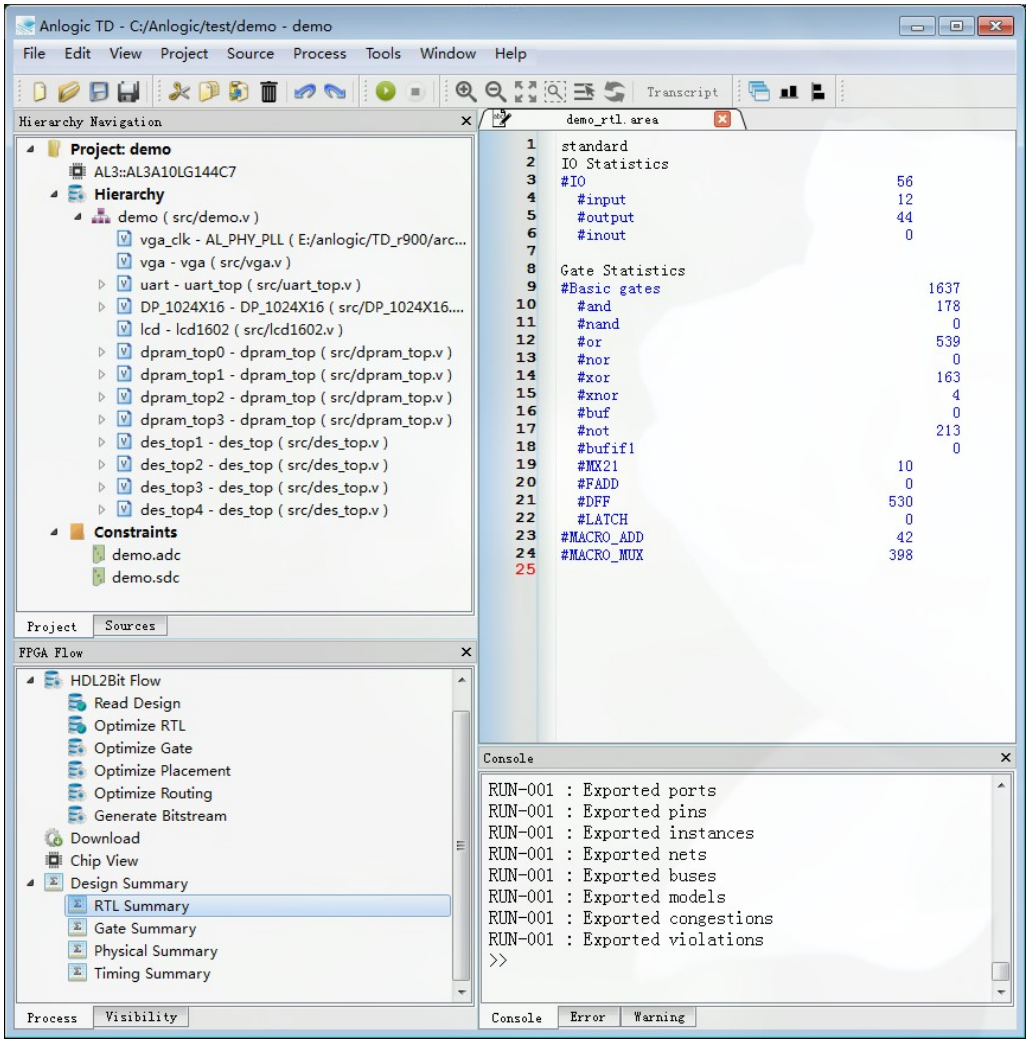
After reading in the design file, TD will optimize the design for RTL. This step optimization includes multiplexer optimization, data path optimization, and automatic identification of special function modules.

RTL-level optimization will produce circuits that contain basic gates (AND, OR, FF/Latch) and special function blocks.

1. Expand HDL2Bit in the FPGA Flow panel

2. Double-click Optimize RTL, or right-click on Optimize RTL, select Run, and the area report file rtl.area will be generated. This file contains all the input and output ports in the source file and the usage of each logic gate.

3. **Expand Design Summary, double click to view RTL Summary**



4. Parameter configuration

In the menu bar, expand Process→Properties, the Properties Configuration window pops up, and select Optimize RTL to set it up.

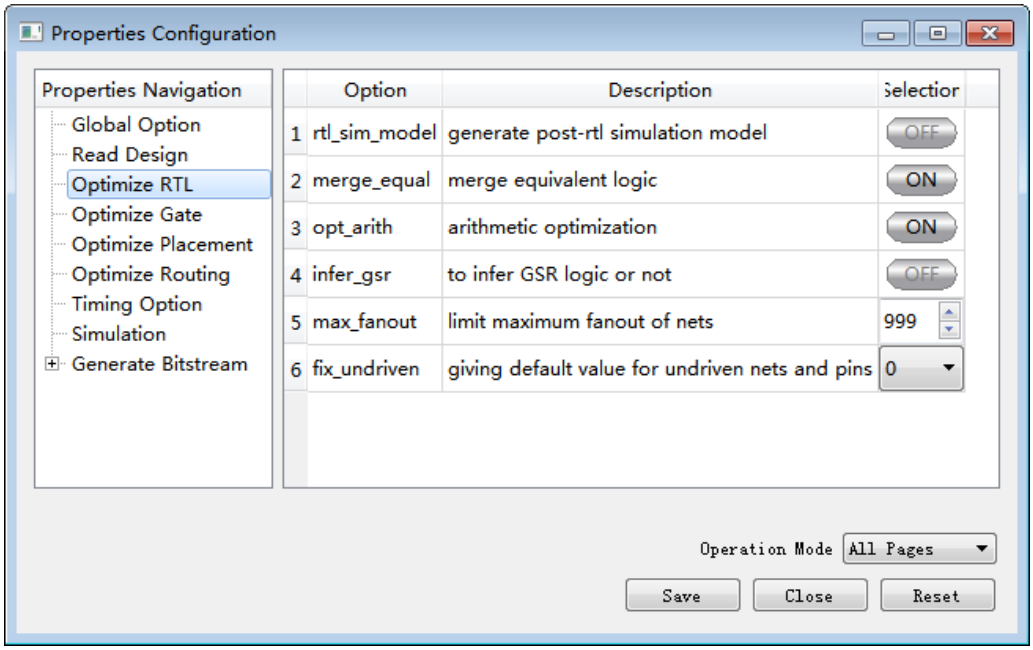


表 5-2　Optimize RTL Properties

Property	Comment	Default
rtl_sim_model	Generate RTL level circuit simulation model	OFF
merge_equal	Merging functionally equivalent logic modules	ON
opt_arith	Arithmetic optimization	ON
infer_gsr	Global gsr optimization	OFF
max_fanout	Limit the maximum number of fanouts of the logical network	999
fix_undriven	For a network or port to which the user is not assigned, give a constant value of 0/1.	0

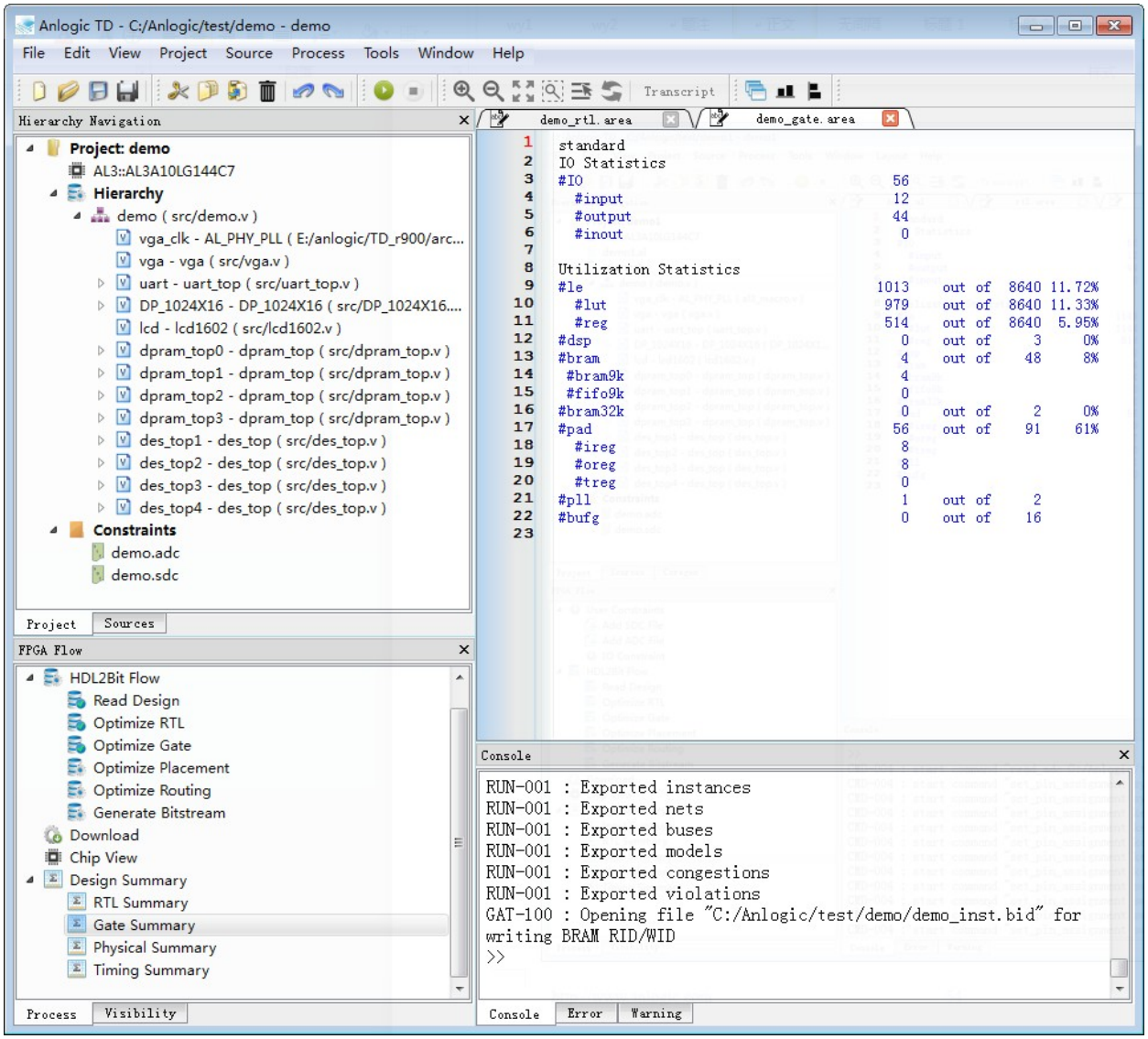
5.3 Gate level optimization

Gate-level optimization includes optimization of general logic and optimization of mapping, special logic optimization, and mapping. Gate level optimization will result in circuits containing logic cells and dedicated functional units.

1. Expand HDL2Bit in the FPGA Flow panel

2. Double-click Optimize Gate, or right-click on Optimize Gate, select Run, and the file will be generated: gate.area, which lists the usage of IO ports and logical units.

3. **Expand Design Summary, double click to view Gate Summary**



4. Parameter configuration

In the menu bar, expand Process→Properties, the Properties Configuration window pops up, and select Optimize Gate to set it up.

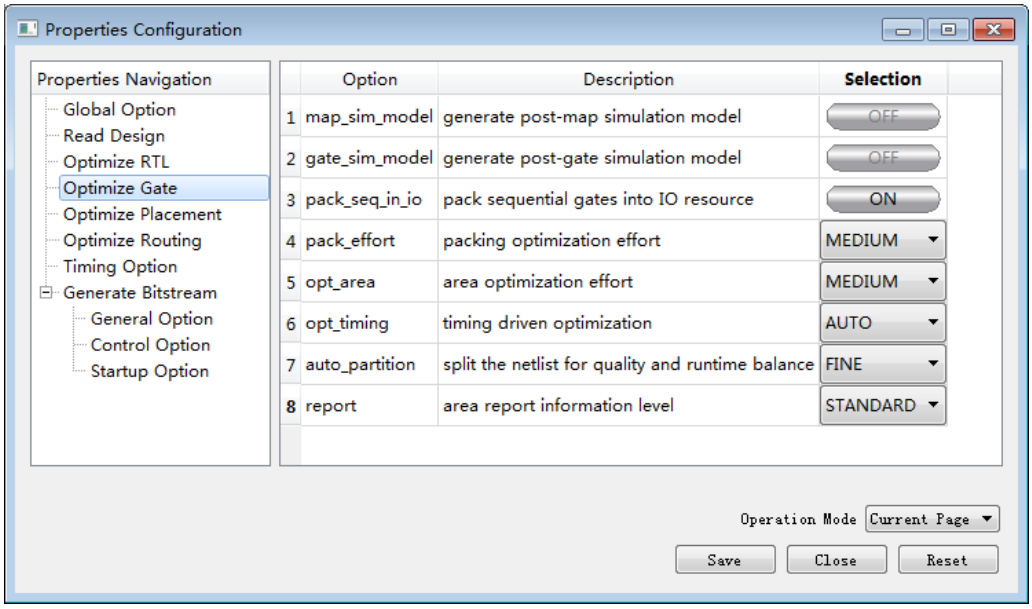


表 5-3 Optimize Gate Properties

Property	Comments	Default
map_sim_model	Generating a mapped circuit simulation model	OFF
gate_sim_model	Generating gate level circuit simulation model	OFF
pack_seq_in_io	Absorption register logic into IO module	ON
pack_effort	Logical packaging optimization level	MEDIUM
opt_area	Combinatorial logic optimization level	MEDIUM
opt_timing	Timing optimization level	AUTO
auto_partition	Partitioning the netlist at runtime	OFF
report	Report information level	STANDARD

5.4 Layout optimization

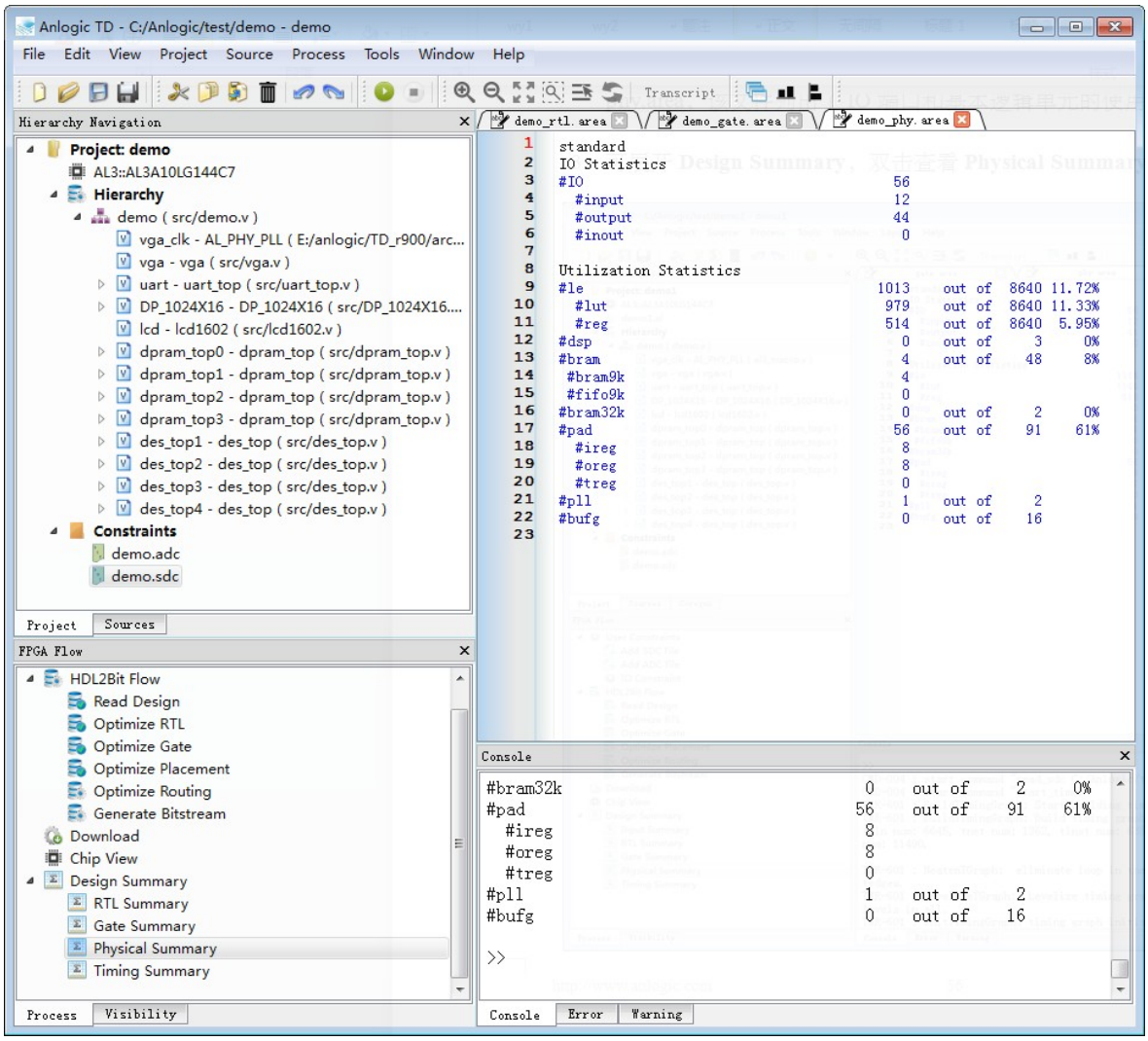
After getting the correct physical unit netlist, you need to optimize the physical layout of the design, IO unit layout, physical unit layout and physical level logic optimization. Layout optimization will generate and process circuits that contain only physical function blocks (IOPAD, SLICE, RAM, DSP, etc.).

1. Expand HDL2Bit in the FPGA Flow panel

2. Double-click Optimize Placement, or right-click on Optimize Placement and select Run. A file will be generated:

   phy.area, which lists the usage of IO ports and basic logical units

3. **Expand Design Summary, double click to view Physical Summary**



4. Parameter configuration

In the menu bar, expand Process→Properties, the Properties Configuration window pops up, and select Optimize Placement to set it up.

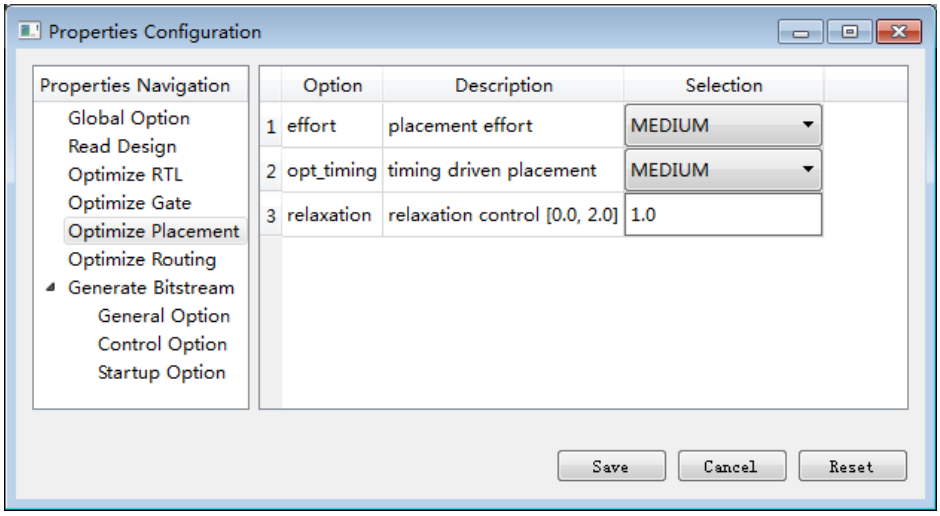


表 5-4 Optimize Placement Properties

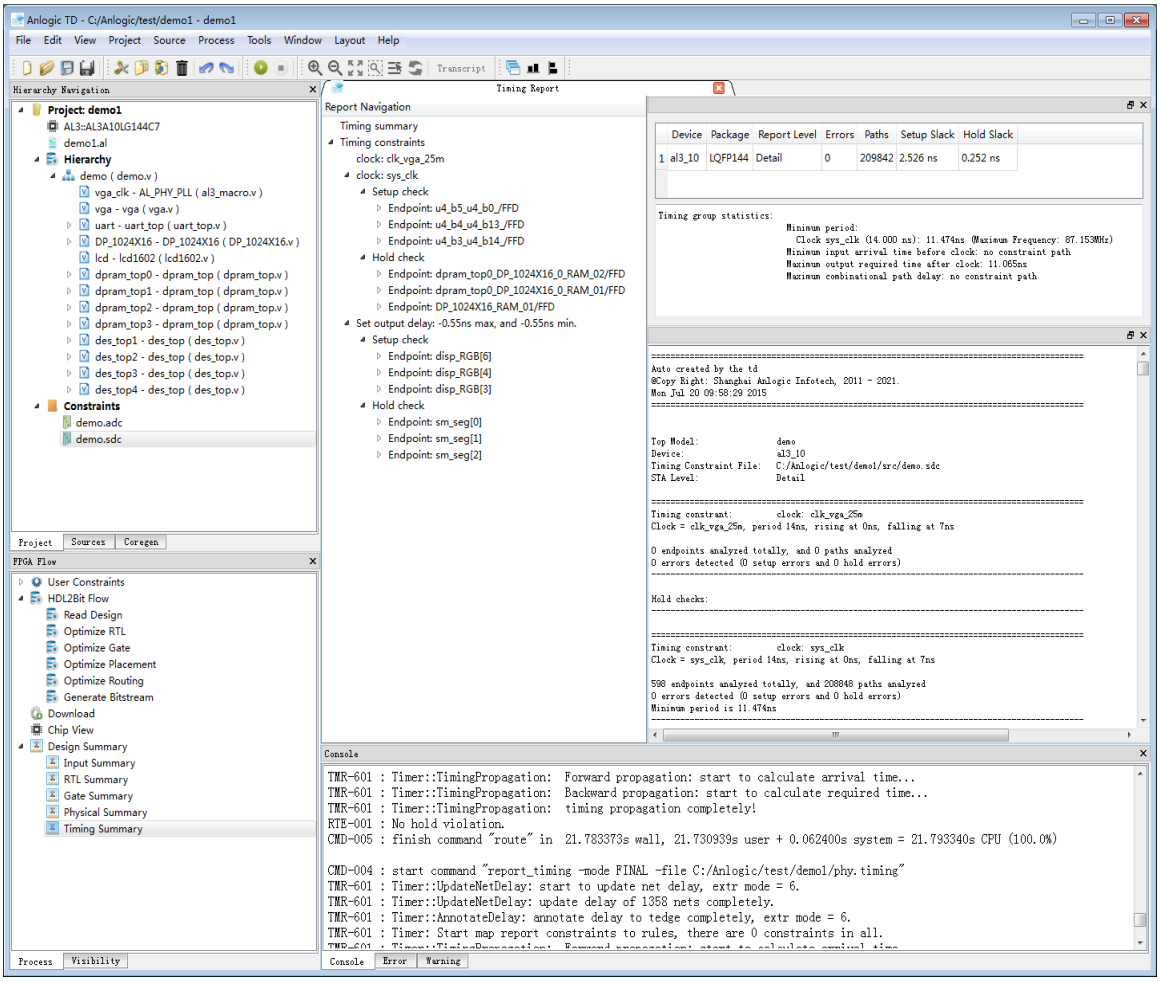
Property	**Comments**	**Default**
effort	Cabling optimization level	MEDIUM
timing	Delay optimization level	MEDIUM
relaxation	Degree of relaxation control [0.0, 2.0]	1.0

5.5 Cabling optimization

After the layout is optimized, routing optimization is performed. Wiring optimization will complete the physical connection of all module interconnect signals. This step is also the final step in user design implementation. After this step is completed, all physical information is determined. After routing optimization, you can view the details of the design and get accurate circuit timing information.

1. Expand HDL2Bit in the FPGA Flow panel

2. Double-click Optimize Routing, or right-click Optimize Routing and select Run. If the user has added an SDC constraint for the project, the TD will generate a timing analysis report for the user by default after the routing is completed. Expand the Design Summary in the FPGA Flow panel and select Timing Summary to view the final timing report. If the user does not add an SDC file for the project, when viewing the Timing Summary, the TD will prompt the user for no timing constraints.



3. Parameter configuration

In the menu bar, expand Process→Properties, the Properties Configuration window pops up, and select Optimize Routing to set it up.

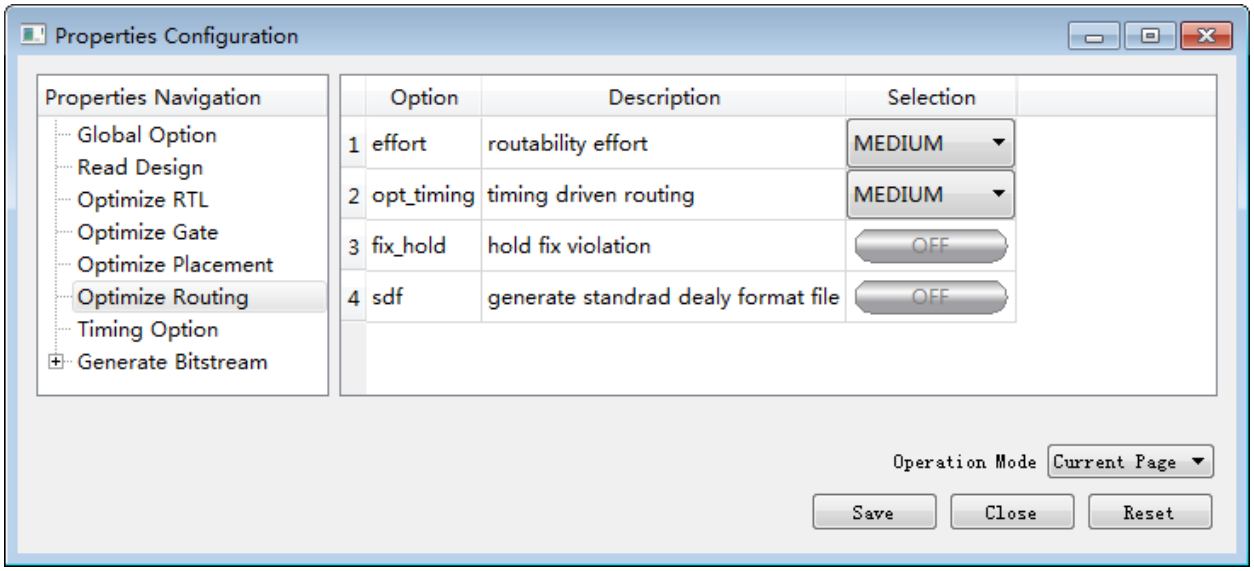


表 5-5 Optimize Routing Properties

Property	Comments	Default
effort	Step rate optimization level	MEDIUM
opt_timing	Delay optimization level	MEDIUM
fix_hold	Fix hold violation	OFF
sdf	Generate standard timing backoff files	OFF

5.6 Generate a bitstream file

Generate Bitstream is used to display the configuration information of the programmable switch in the FPGA chip into bitstream data in binary 0 and 1 format for program download. The bitstream generator generates a bitstream file for device programming, and the download tool loads the bitstream file into an external SPI Flash memory chip or directly into the configuration memory inside the FPGA.

1. Expand HDL2Bit in the FPGA Flow panel

2. Double-click Generate Bitstream, or right-click on Generate Bitstream and select Run. A file will be generated:

   Your_Project_Name.bit, this file is the configuration information of the programmable switch in binary 0, 1 format

3. Parameter configuration

   In the menu bar, expand Process→Properties, the Properties Configuration window pops up, select Generate Bitstream to set it up.

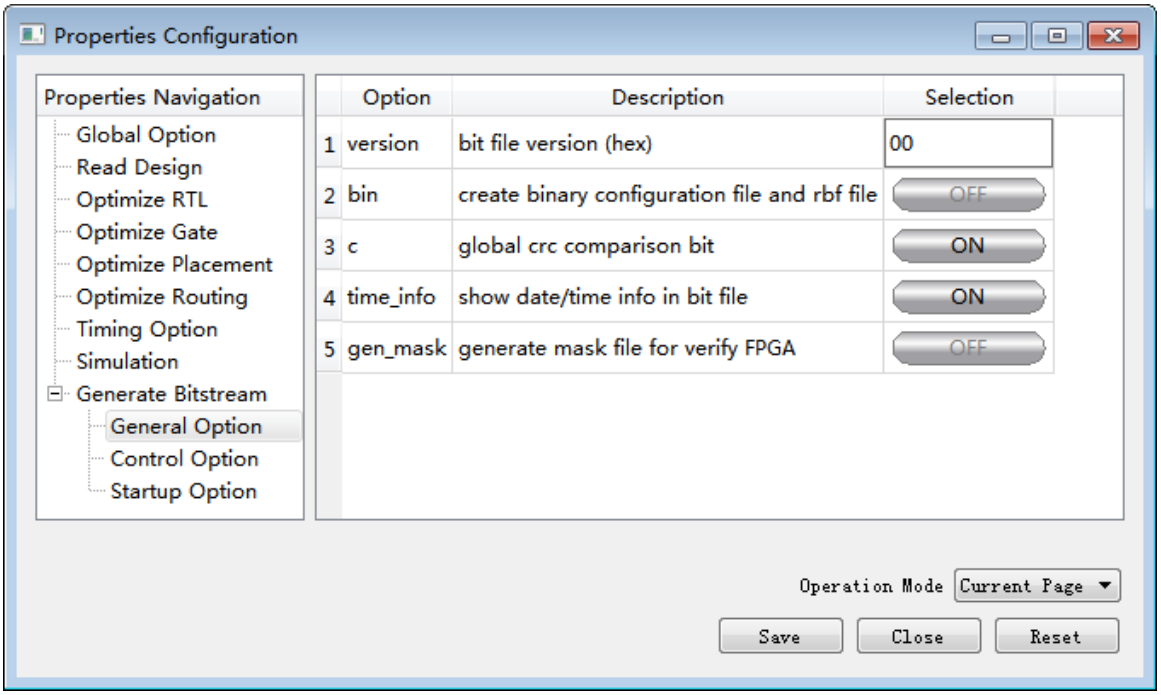


表 5-6　General Option

Property	Comments	Default
version	Define the number of the bitstream file	00
bin	Generate a pure binary bit stream file, that is, a file header that does not contain a regular bit file	OFF
c	Generate a global CRC check bit	ON
time_info	Display date/time information in a bitstream file	ON
gen_mask	Generate a mask file for the verify FPGA	OFF

Control Option is a 32-bit control register that is divided into several control options to change the value of each item to achieve different control effects.

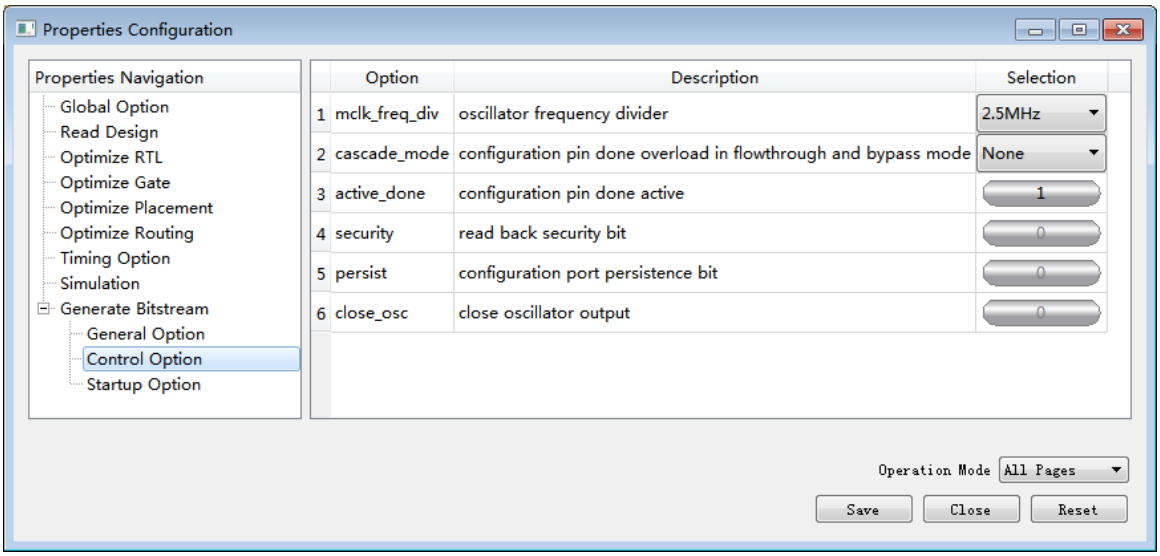


表 5-7　Control Option

Property	Comments	Default
mclk_freq_div	spi flash clock division factor is used in cascade, done pin as input	2.5MHz
cascade_mode	(2'b11: flowthrougth mode 2'b10: bypass mode)	None
active_done	Done pin is active and is determined internally by cfg	0
security	For 1 time, forbidden to read sram	0
persist	Whether the spi pin continues to be used as the cfg pin in user mode	0
close_osc	Turn off the oscillator	0

The Startup Option is a 32-bit control register associated with the startup item. The control principle is the same as the Control Option.

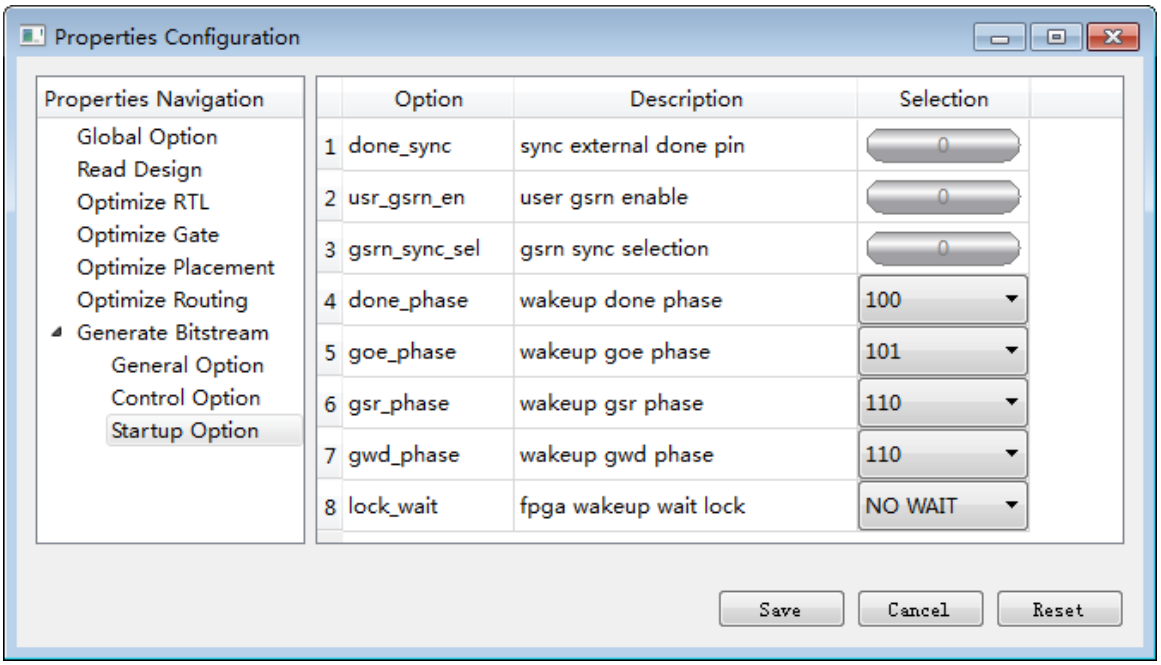
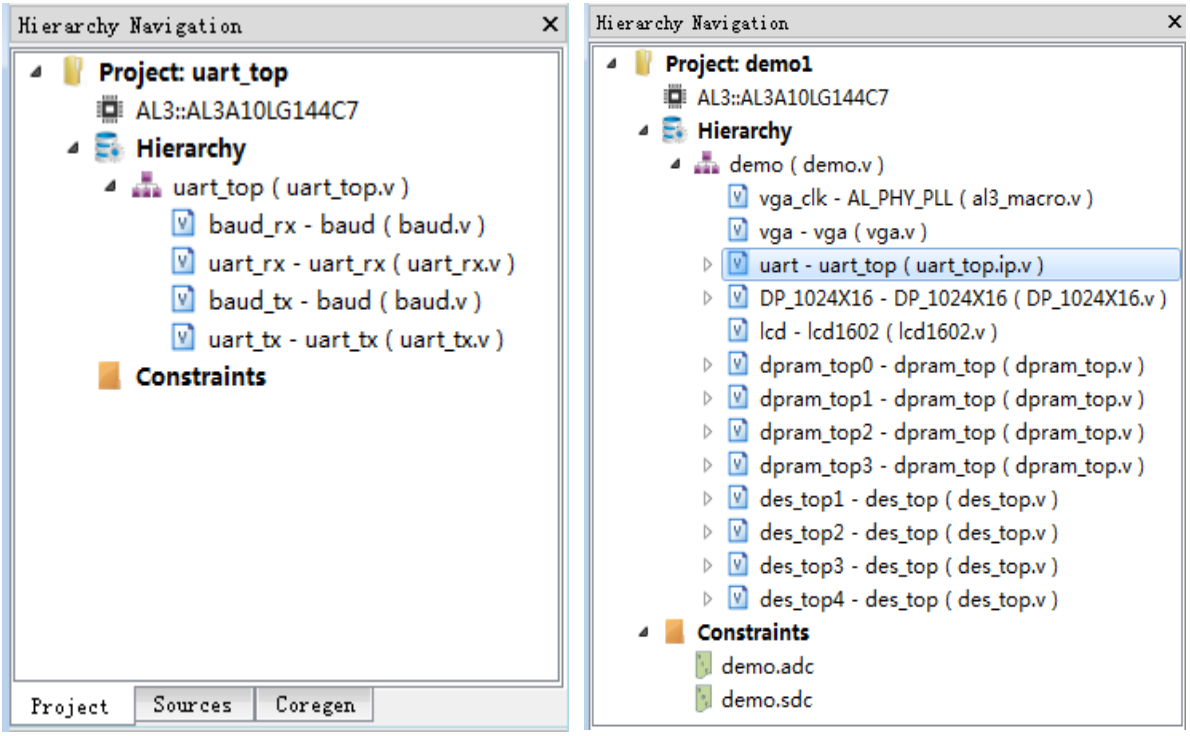


表 5-8 Startup Option

Property	Comments	Default
done_sync	Whether to synchronize the value of done pin	0
usr_gsrn_en	Whether to enable the user's gsrn signal	0
gsrn_sync_sel	Whether to synchronize gsrn	0
done_phase	Decide which phase to release done	3'b100
goe_phase	Decide which phase to release goe	3'b101
gsr_phase	Decide which phase to release gsr	3'b110
gwd_phase	Decide which phase to release gwd	3'b110
lock_wait	fpga wake up waiting clock (NO WAIT: 0000 PLL0： 0001 PLL2： 0100 PLL0&PLL2：0101)	NO WAIT

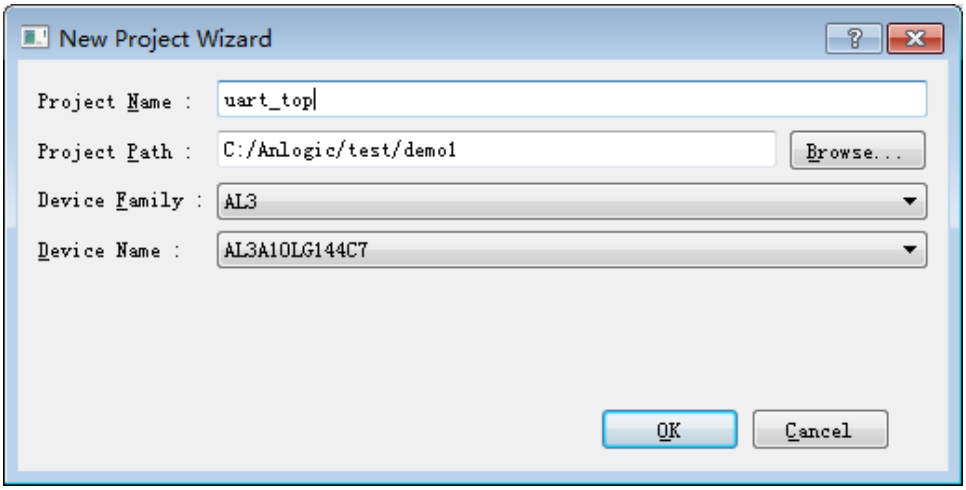
5.7 Syn_ip_flow

In the FPGA design process, if the user does not want the third party to see their own source code, they can implement protection measures for the source code of the design, that is, the source code is separately made into an IP module for third-party calling. To this end, TD provides users with the function syn_ip_flow, which synthesizes the user's high-level circuit description into a gate-level netlist, which protects the source code to a certain extent. When using this function, users need to create two projects, add the source code to be protected in the first project and run syn_ip_flow, add the peripheral circuit in the second project and call the netlist file generated by syn_ip_flow. For example, in the demo module used in this manual, if the submodule that the user wants to protect is uart_top, the following two projects need to be established:

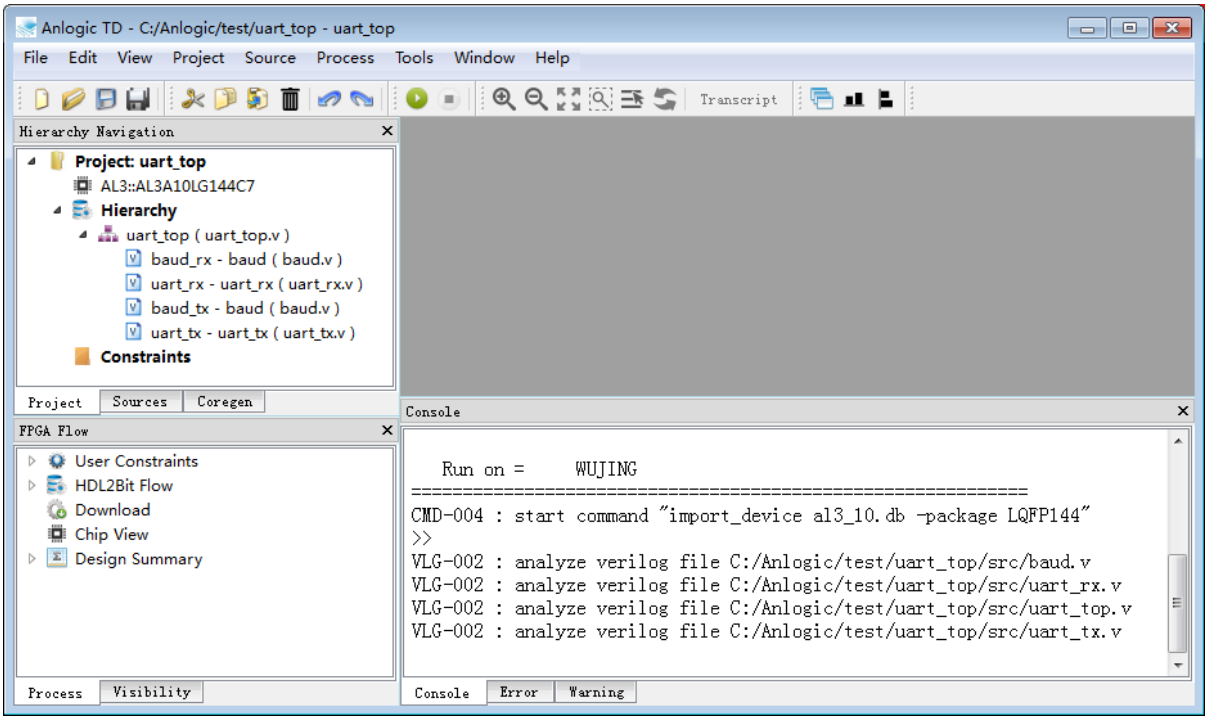


The specific steps are as follows:

1. The submodules that need to be protected are built separately. In this example, the uart module is used as an example to create a separate project.

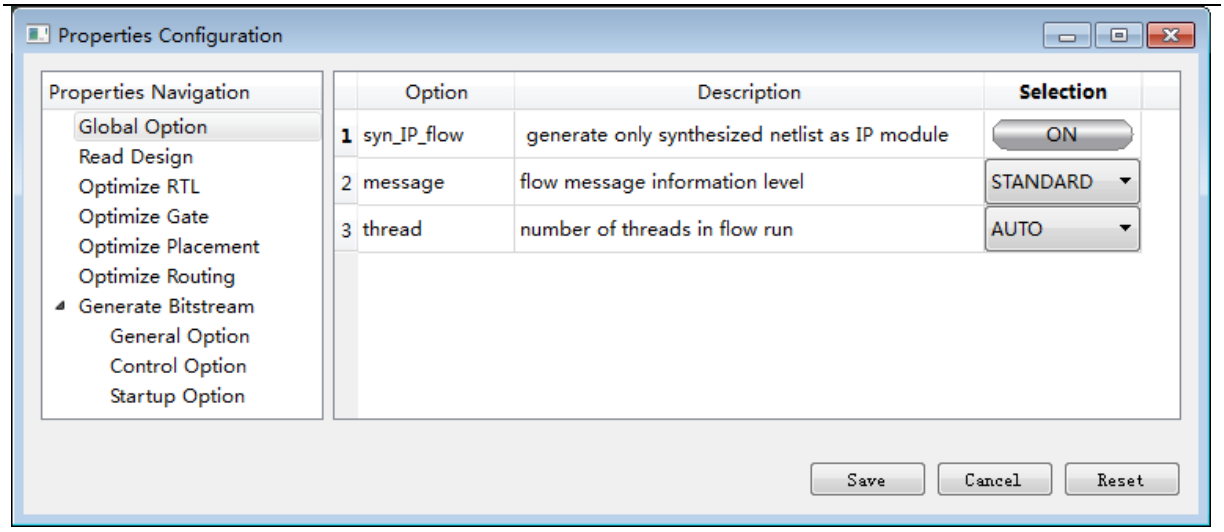


13. Add source files to the project



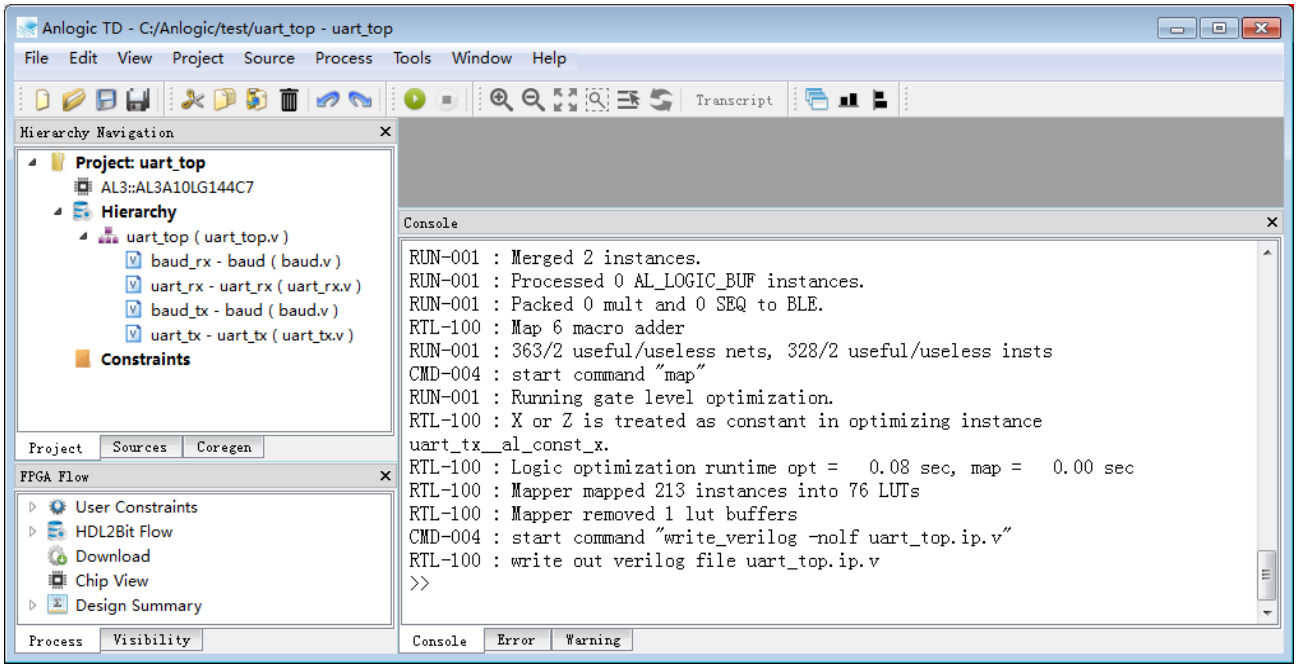
14. When you run syn_IP_flow, you need to set the parameters first:

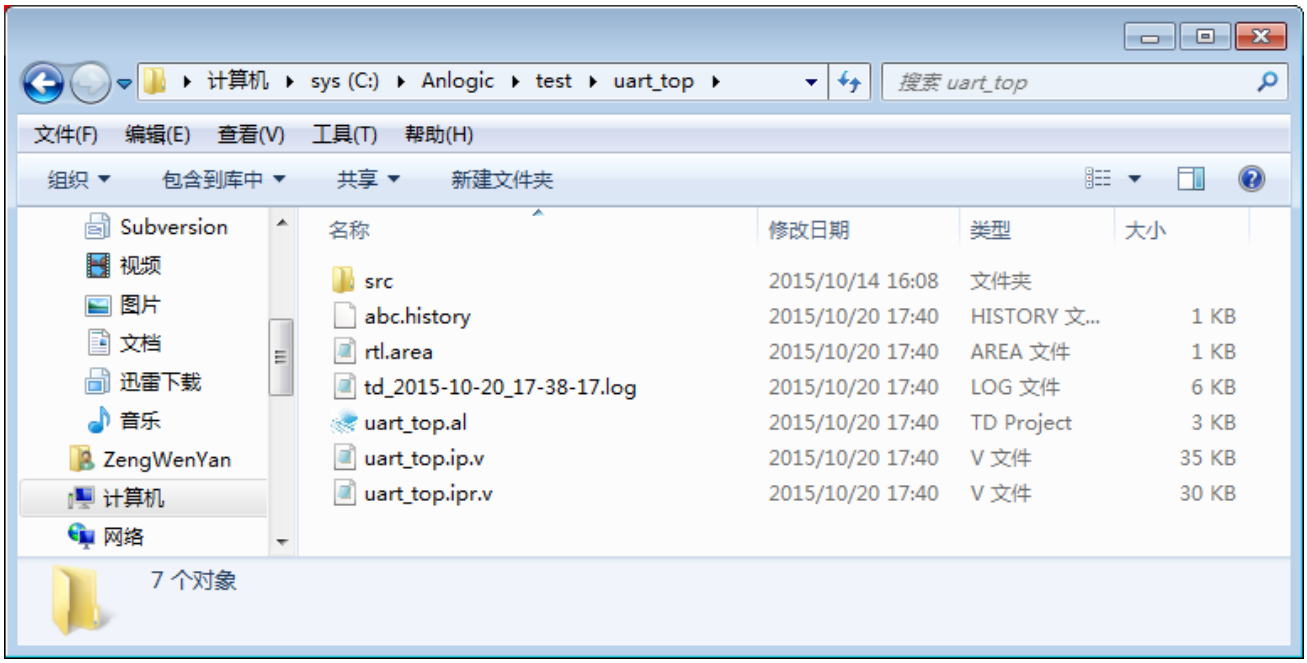
Process →Properties →Global Option →syn_ip_flow, set this parameter to ON and click “Save” to save.



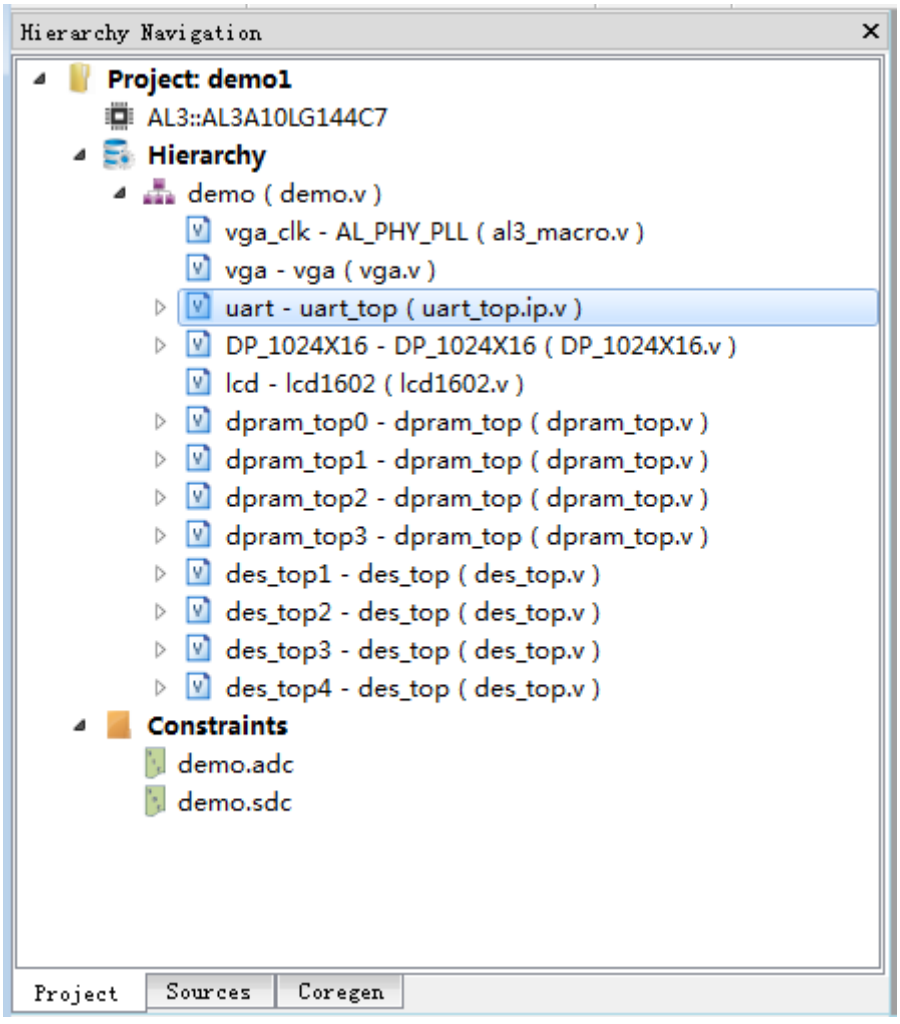
15. After the parameter setting is completed, click the icon to run the project. After the project runs, TD will generate two files: project_name.ip.v and project_name.ipr.v, and store them in the project directory. These two files are completely functional. Equivalent and can be used for simulation.

    However, project_name.ip.v contains information such as naming and declarations in the source code for easy debugging. Project_name.ipr.v hides the naming and declaration information in the source code to better protect the source code.





5. Create a new project, and call the project_name.ip.v or project_name.ipr.v generated in the previous step, and add the corresponding constraint file for the project, run the entire project, and complete the whole process of syn_ip_flow.



5.8 Synthesis Keep

Using Synthesis keep ensures that the signal will not be optimized by subsequent processes, making it easy for users to debug later.

The specific practices are as follows:

1. In the verilog file, add the appropriate comment for the signal you want to keep. The comment is written as:

   //synthesis keep = 1;　　/*synthesis keep=1*/;
   //synthesis keep = true;　　/*synthesis keep=true*/
   //synthesis keep;　　/*synthesis keep*/

   Such as :

```
module test_keep (clk, a, b, c, out);

input clk;

input [3:0] a;
input [3:0] b;
input [3:0] c;
output reg [3:0] out;

wire [3:0] sig; //synthesis keep

assign sig  = a & b;

always@(posedge clk)
begin
   out <= sig | c;
end

endmodule
```

2. Save the file and compile it;

3. This internal signal sig can be viewed when using the debug tool.

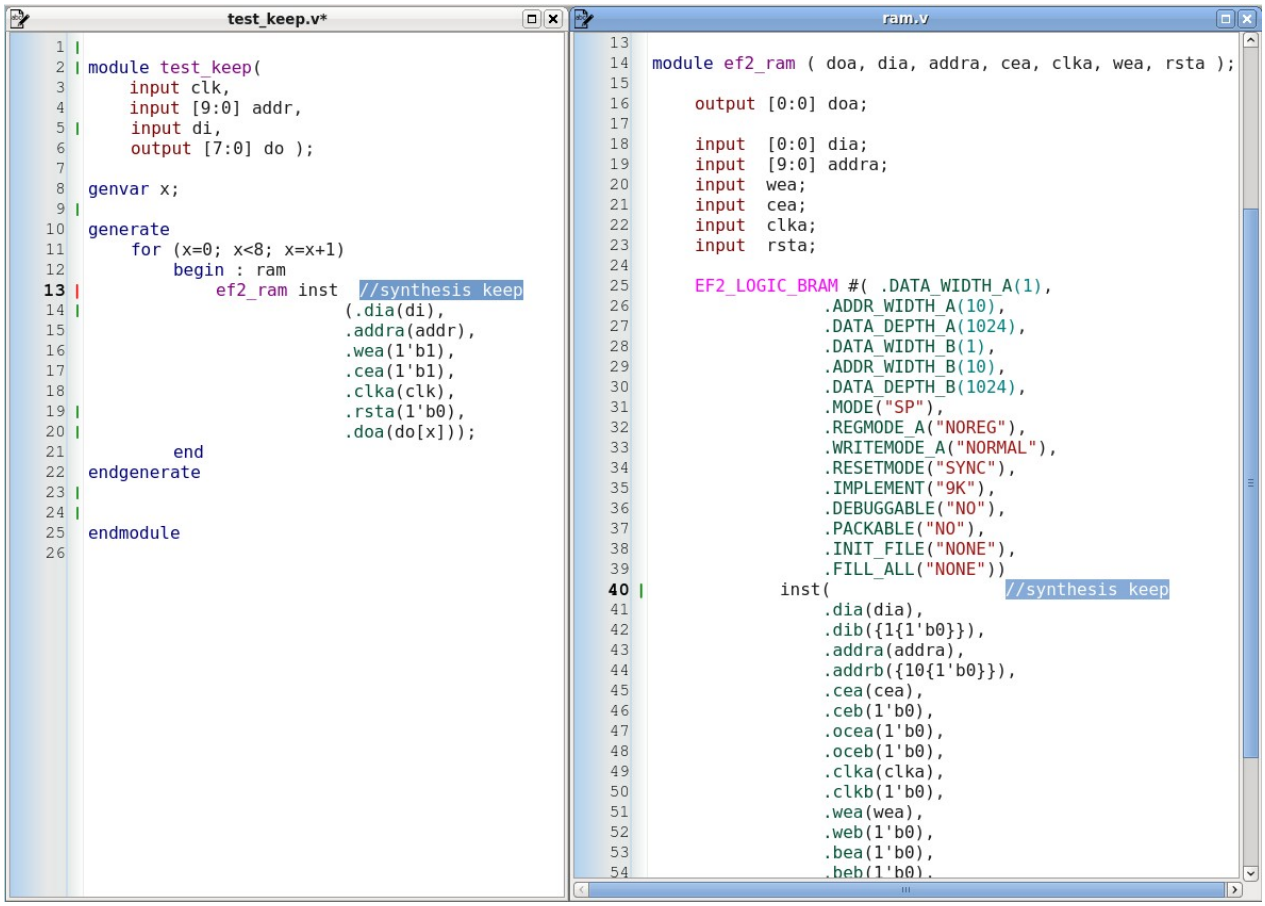
4. The wording in vhdl is as follows:

```
attribute keep : BOOLEAN;
attribute keep of clkc_wire : signal is TRUE;
```

Among them, clkc_wire is the net / bus that needs to keep.

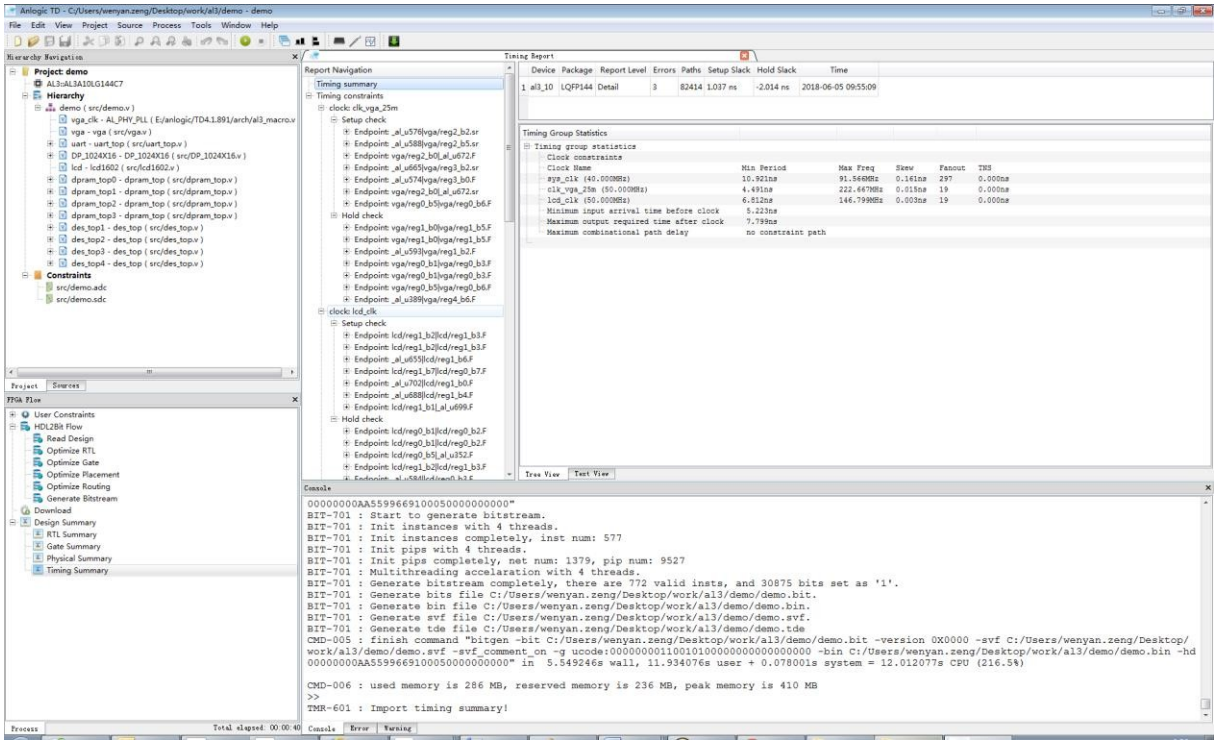
If you want to keep multiple identical modules in design from being optimized, TD provides a way to keep instance.

The module ef2_ram in the following figure is completely equivalently instantiated multiple times in the top module. If the combine keep function is not used at this time, all instances will be merged into one BRAM. To keep all instances from being optimized, you need to add the comment "//synthesis keep" to each level of the instance.

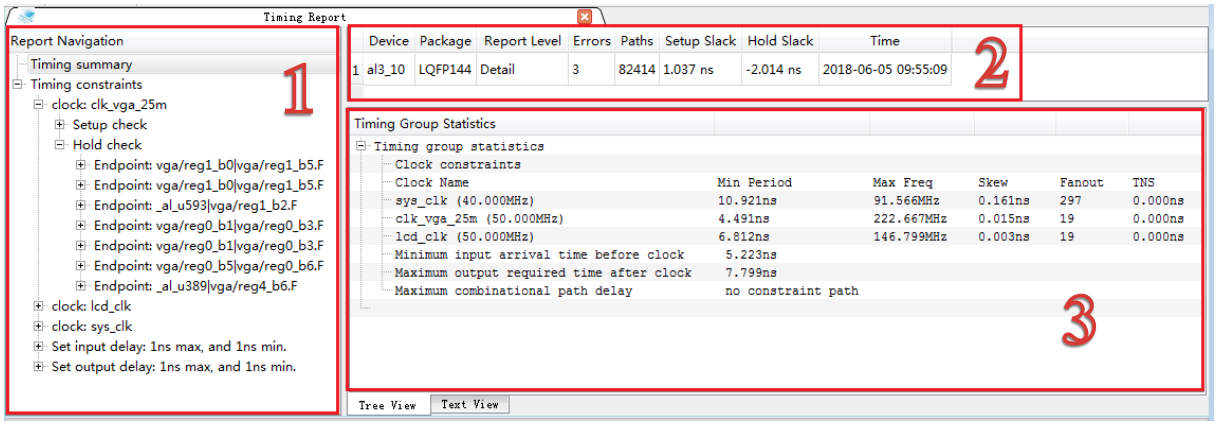


5.9 Introduction to Timing Report

When HDL2Bit Flow is running, double-click Design Summary -> Timing Summary to open the Timing Report. The main interface is as follows:



Timing Report interface consists of the following sections:



1. Report Navigation

Here is the directory tree for the timing report, which is listed hierarchically in order: timing constraint type, timing check type, EndPoint name, timing path name, and so on.

2. Timing Summary

The chip, package, and abbreviated timing statistics for the timing report are listed here.

3. Timing Group Statistics

Clock constraints: lists the minimum period, maximum frequency, skew of the clock tree, fanouts, and TNS data that each clock defined in the SDC can reach;

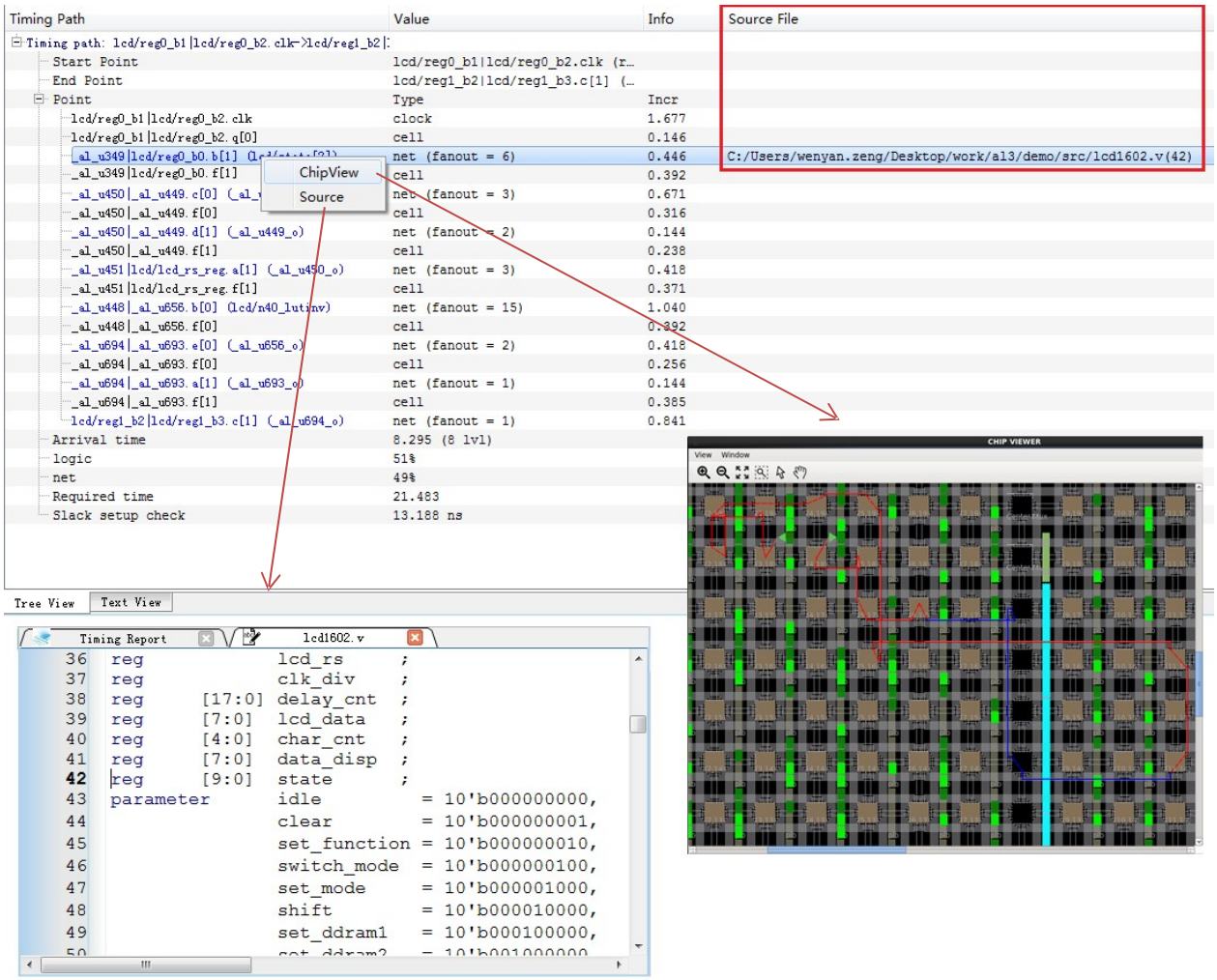
Minimum input arrival time before clock：The maximum delay of the input->reg path inside the FPGA chip.

Maximum output required time after clock：The maximum delay of the reg->output path inside the FPGA chip.

Maximum combinational path delay：Input->output The maximum delay of the combined logical path of the passthrough.

Expanding the Setup Check or Hold Check under clock in Timing constraints will display the corresponding number of timing paths according to the proerties setting. Double-clicking on a certain timing path will display the details of the timing path in the right column of the Timing Report, where the main entries are displayed. The meaning is as follows:

Start Point	Refers to the starting point of the timing path, typically the clock side of the register or the original input
End Point	Refers to the end of the timing path, typically the input data pin or the original output of the timing unit
Point	List the node names passed by the timing path in turn
Type	Indicate whether the time series path of the segment passes through a cell or a net
Incr	Delay increment of the segment timing path
Source File	Indicate where the net is defined in the Source File (requires the net_info option)
Arrival time	Indicates the arrival time of this timing path: Arrival time = $T_{launch\ edge}$ + $T_{launch\ clock\ delay}$ + path delay In the case of setup check, the maximum possible delay, while in the case of hold check, the minimum possible delay, and the arrival time of the trigger clock is also included.
Logic	Indicates the proportion of delay that a logical resource occupies in the entire timing path
Net	Indicates the proportion of delay that the interconnect resource occupies in the entire timing path
Required time	Indicates the required time for this timing constraint: Maximum path constraint（setup check）Required time = $T_{capture\ edge}$ + $T_{capture\ clock\ delay}$ − T_{setup} − $T_{clock\ uncertainty}$ Minimum path constraint（hold check）:Required time = $T_{capture\ edge}$ + T_{hold} + $T_{clock\ uncertainty}$
Slack	Indicates the slack margin of the timing path. A value less than 0 represents a timing risk: Maximum path constraint（setup check）:Slack = Required time – Arrival time Minimum path constraint（hold check）:Slack = Arrival time - Required time



The naming rules for Timing path are:

"/" indicates a hierarchical relationship;

“|” indicates a parallel relationship;

"." indicates affiliation.

The following table describes the meaning of each node through which the timing path passes:

Point	Type	Incr	Comment
lcd/reg0_b1|lcd/reg0_b2.clk	clock	1.677	Indicates that reg0_b1 and reg0_b2 in the lcd module are merged into the same slice, starting at the clk end of the slice. Clock represents the delay of the clock tree is 1.677ns
lcd/reg0_b1|lcd/reg0_b2.q[0]	cell	0.146	The delay inside the cell, that is, the delay from the clk end of the slice to the output q[0] is 0.146 ns.
_al_u349|lcd/reg0_b0.b[1] (lcd/state[2])	Net fanout=6	0.446	For the net delay, the line delay between the q end of the previous slice and the b end of the next slice is 0.446 ns; in parentheses is the net name: net lcd/state[2]. Fanout=6, which means that the signal drives a total of 6 pins.
…	…	…	

For the net that gives the file path, you can right click on the net, click Source to jump to the corresponding place of the source file, click ChipView to display the specific trace of the path in ChipView, and the blue line is the net. The part of the entire path.

6 AL3S10 Device

6.1 Introduction to the AL3S10 device

Anlogic's latest AL3S10 FPGA is based on Anlogic's proven low-cost, low-power programmable FPGA AL3A10, which is sealed with a 2M x 32bit SDRAM using the latest 3D sealing technology. The AL3S10 FPGA has a smaller, simpler and more reliable device package and a larger embedded memory capacity, making it ideal for high-capacity, high-speed data acquisition, transmission and conversion applications.

Special advantages:

Multi-variety, large capacity built-in storage space

- Built-in 64Mb SDRAM memory space, 32-bit data bus width, up to 200Mhz operating frequency, maximum read/write bandwidth up to 800MB/s
- Built-in 48 EMB9K random read/write RAM, which can be configured as true dual port, simple dual port, single port RAM and FIFO working mode. The bit width can be configured as 512×18, 1K×9, 2K×4, 4K×2, 8K× 1, the highest frequency is 250Mhz
- Built-in 2 32Kb RAM, configurable as single-port RAM, dual-port RAM, independently configurable as 2Kx16 or 4Kx8

Smaller package, more IO, better for PCB layout

- HLQFP144 package, EPAD ground, up to 111 general purpose IO, 4 reusable IO
- Supports up to 16 pairs of True LVDS with a maximum frequency of 600Mbps
- 0.4mm pitch, 18mm x 18mm ultra small package
- Only need 1.2V, 3.3V two sets of voltage supply
- Optimized pinouts make it easy to use all of the device's IO with just two layers of PCB

- Supports simple and low-cost SPI FLASH configuration; after power-on configuration, FLASH can be used as a user.

6.2 Use internal SDRAM

The AL3S10 embeds a 2M x 32bit SDRAM with a maximum operating frequency of 200Mhz and a maximum read/write bandwidth of up to 800MB/s. SDRAM and FPGA are deeply integrated by software, so if you want to use SDRAM, you only need to instantiate the following IP module at the top level. The prototype of this IP is as follows:

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

The pin assignments for SDRAM are as follows:

SDRAM pin name	SDRAM pin description	Pin connection
DQ0 ~ DQ31	Data pin 0 ~ data pin 31	Connected to IP
SA0 ~ SA10	Address pin 0 ~ address pin 10	Connected to IP
BA0	BANK address foot 0	Connected to IP
BA1	BANK address foot 1	Connected to IP
WE_N	Write enable	Connected to IP
RAS_N	Line strobe	Connected to IP
CAS_N	Column strobe	Connected to IP
CLK	Chip clock	Connected to IP
CS_N	Chip Select	Internal pull down

DM0	Data 0-7 Shielding	Internal pull down
DM1	Data 8-15 Shielding	Connected to IP
DM2	Data 16-23 Shielding	Internal pull down
DM3	Data 24-31 Shielding	Internal pull down
CKE	Clock enable	Internal pull high

When using SDRAM, please note the following:

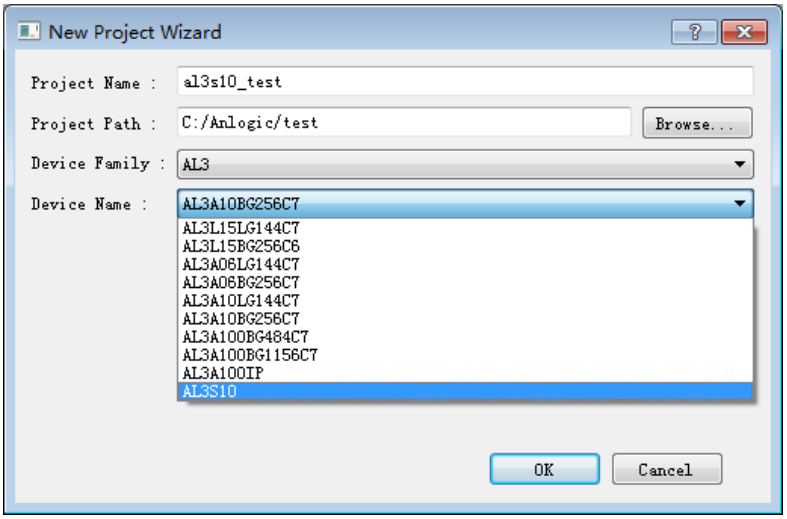
1. DQ0~DQ31 are two-way data feet;

2. The SDRAM operating clock CLK requires a 90 degree phase difference from the FPGA internal clock;
3. When DM1 is assigned a value of 0, 32-bit data is available; assigning a value to DM1 and masking 8-15 bits of data reduces power consumption.

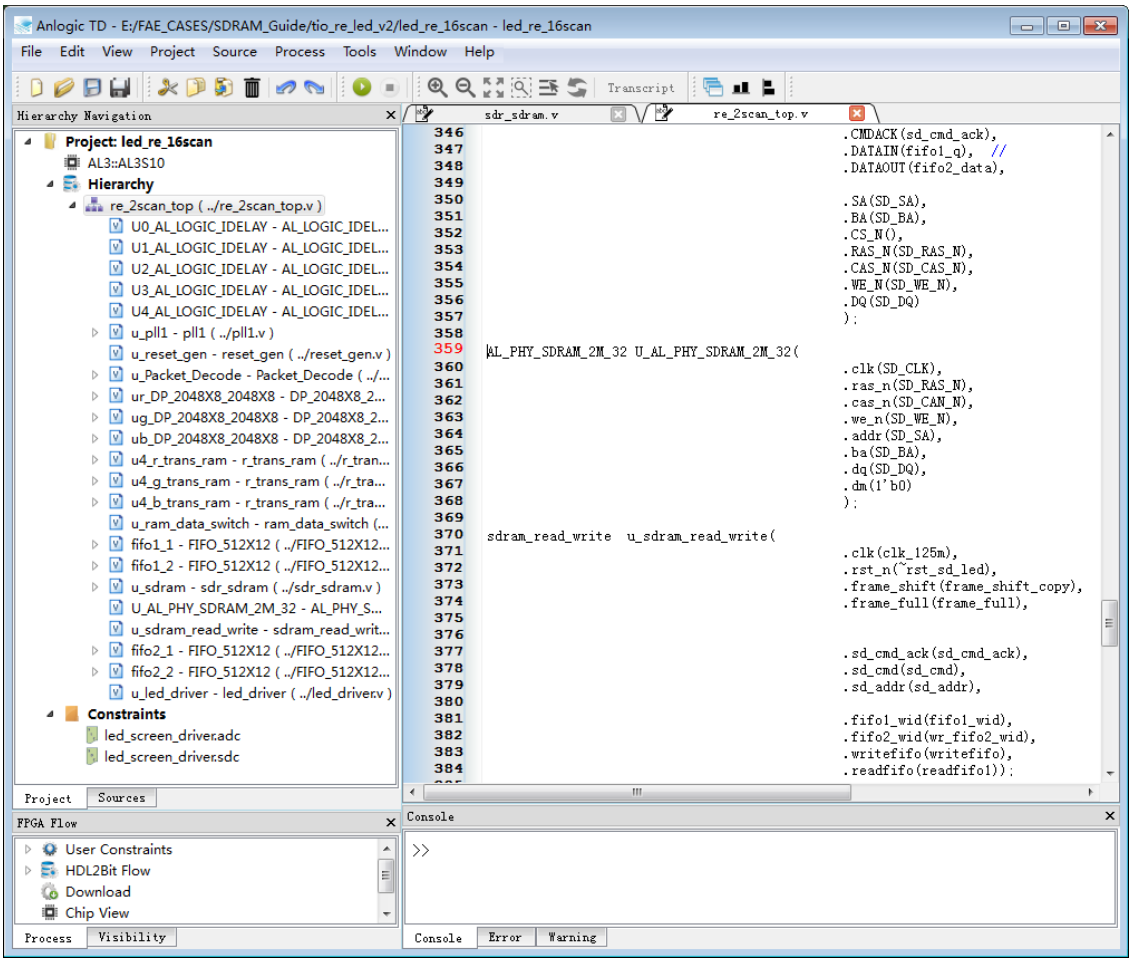
6.3 AL3S10 software usage process

6.3.1 Build project

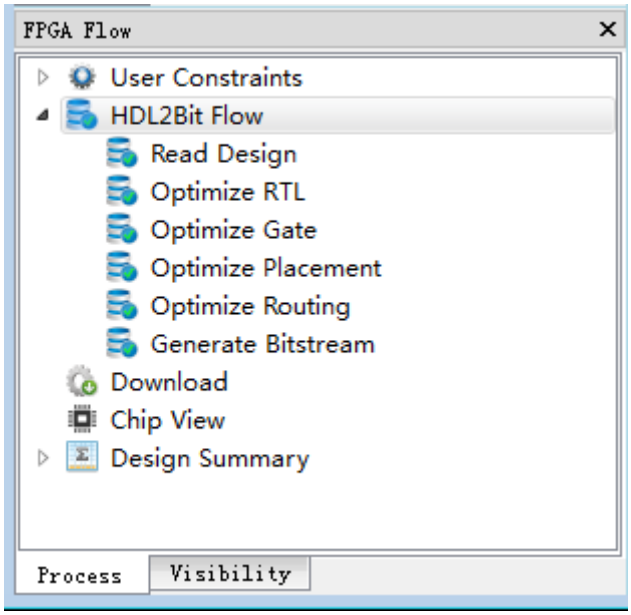
1. **Project → New Project，选择器件 AL3S10**



2. Add a source file to the project and add the sdc and adc constraints.



3. The entire process of running HDL2Bit



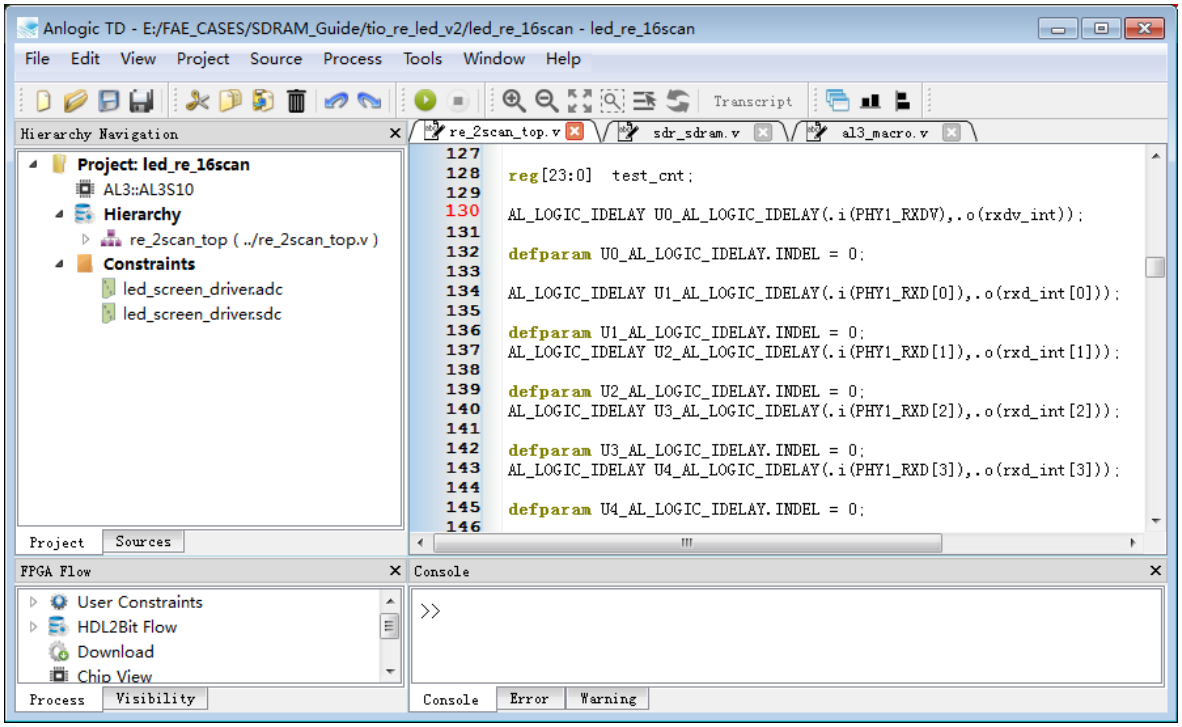
6.3.2 Use of special IP

1. An IO delay unit that can be used to adjust the input delay of the RGMII signal.

   AL_LOGIC_IDELAY U0_AL_LOGIC_IDELAY(.i(PHY1_RXDV),.o(rxdv_int));

   defparam U0_AL_LOGIC_IDELAY.INDEL = 1;

   After use, there is an initial delay of 0.2ns. The parameter is used to set the delay length. For each additional 1, the delay is increased by 0.1ns. For example, when the parameter is set to 1, the delay is 0.3ns.



2. IO input double edge sampling unit IDDR for double edge sampling of RGMII input signal

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

3. IO output double-edge drive unit ODDR for double-edge drive 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));

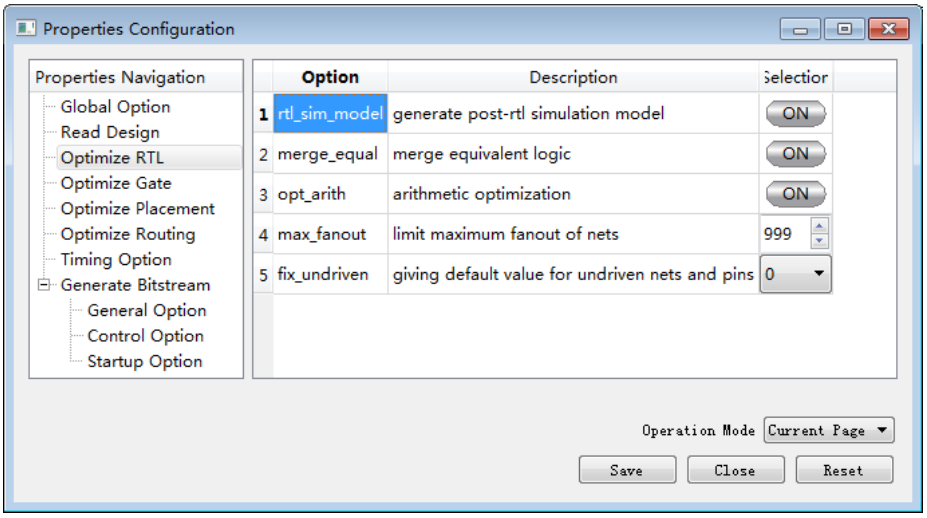
7 Functional simulation

TD supports users to use third-party tools (such as Synopsys VCS, Mentor Graphics Modelsim, etc.) for functional verification and timing verification. The TD provides the functional and timing models required for simulation.

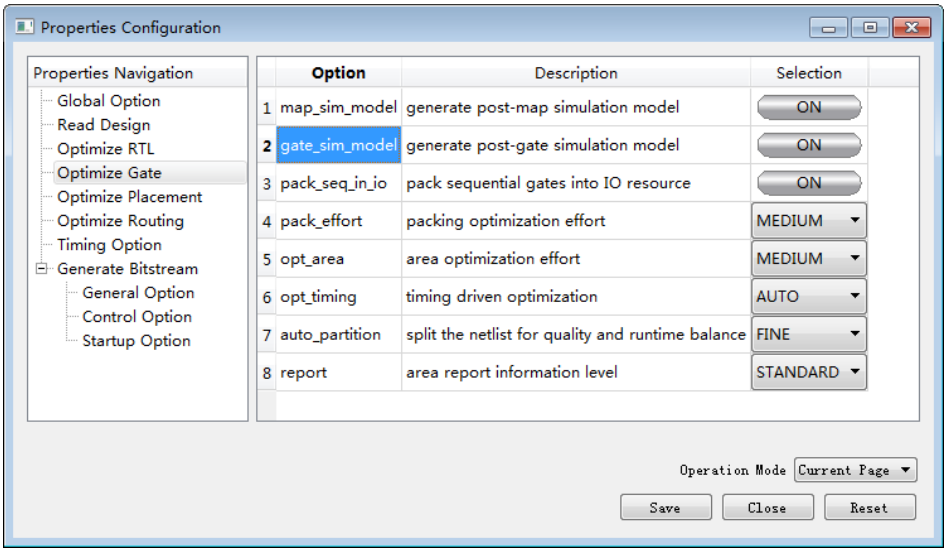
This chapter focuses on the process of generating the files required for Modelsim simulation in the TD software.

1. Set the relevant parameters before running HDL2Bit Flow.

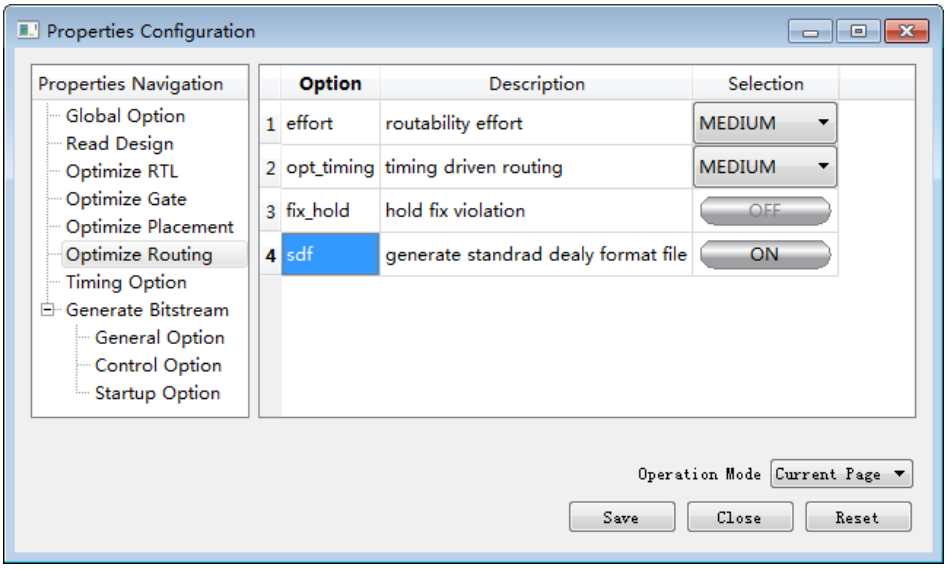
Process → Properties → Optimize RTL：set rtl_sim_model ON。



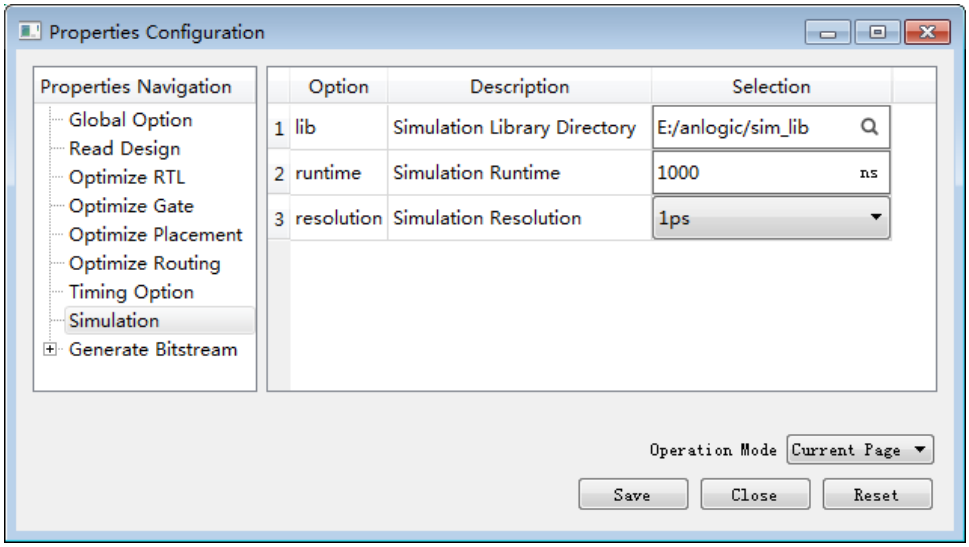
Process → Properties → Optimize Gate： set gate_sim_model ON。



Process → Properties → Optimize Routing：set sdf ON。



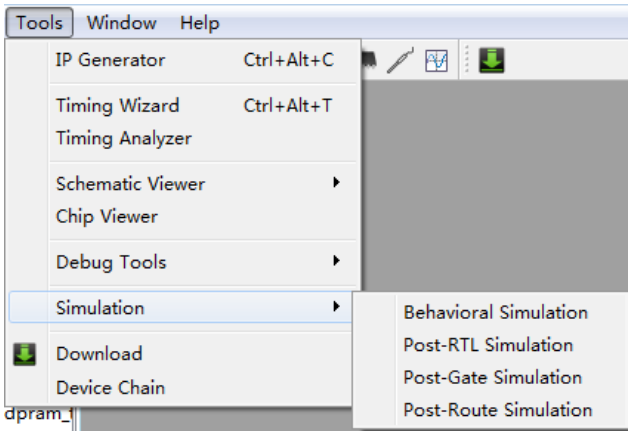
2. Set Modelsim simulation related parameters



Property	Comments	Default
lib	Specify the library file for the simulation	No default value, you need to specify manually
runtime	Specify when the simulation runs	1000 ns
resolution	Specify the time precision of the simulation	1 ps

3. Run HDL2Bit Flow

4. **Run Tools → Simulation**



Perform Behavioral Simulation when HDL2Bit Flow runs to the Read Design step;

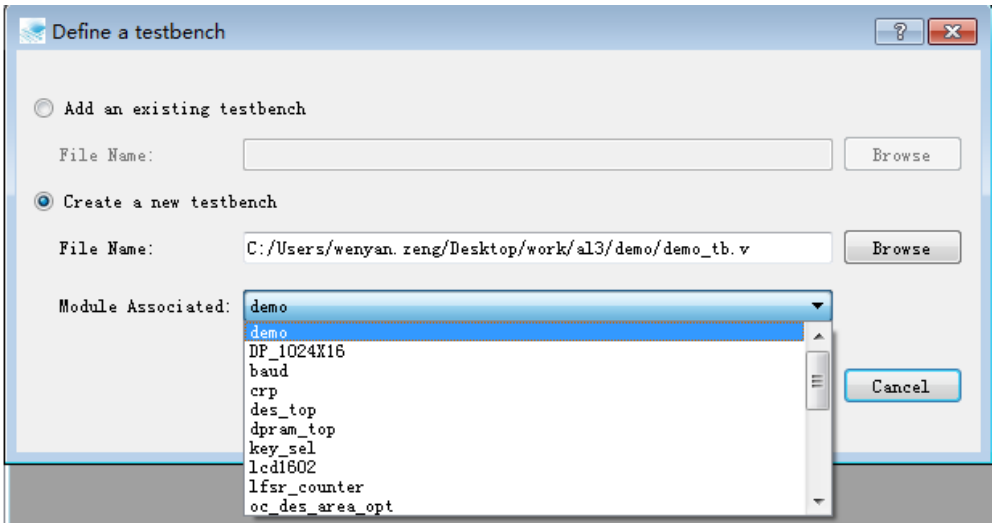
Post-RTL Simulation can be performed when HDL2Bit Flow runs to the Optimize RTL step;

Post-Gate Simulation can be executed when HDL2Bit Flow runs to the Optimize Gate step;

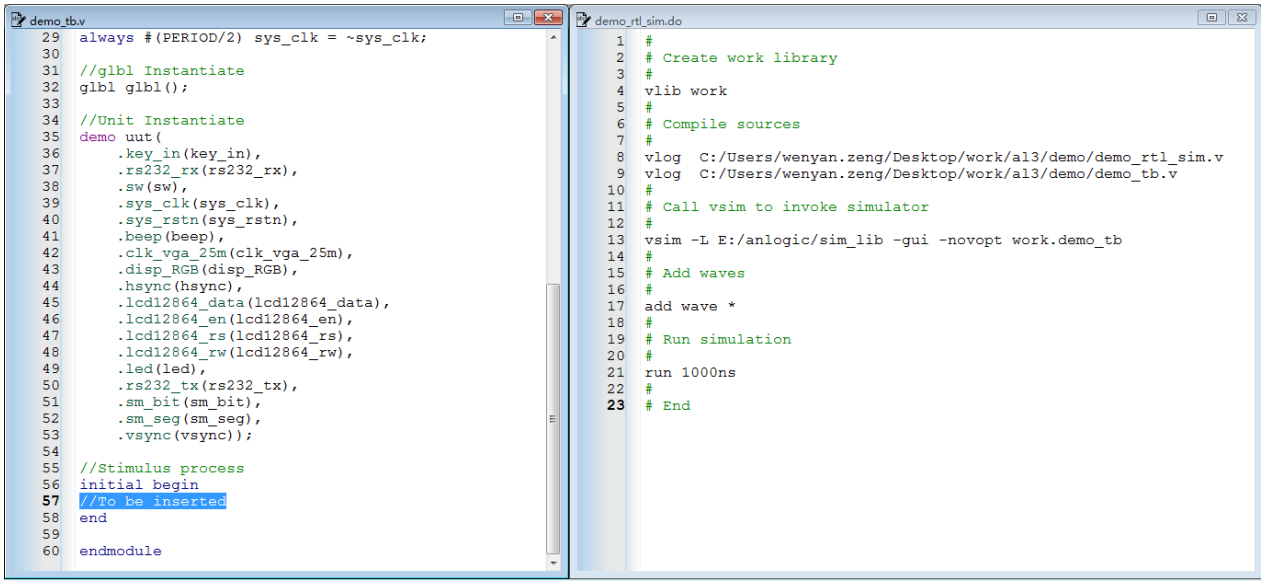
Post-Route Simulation can be performed when HDL2Bit Flow runs to the Optimize Routing step.

5. Define the testbench file

If you click Post-RTL Simulation, the following dialog box will pop up, you can add an existing testbench file, or you can create a new testbench file. When creating a new one, you need to specify the corresponding module.



After clicking OK, prj_tb.v and prj_name_rtl_sim.do will be generated in the project directory and these files will be opened in the TD interface. Note that there is no incentive in prj_tb.v, you need to fill it in manually before doing the simulation.



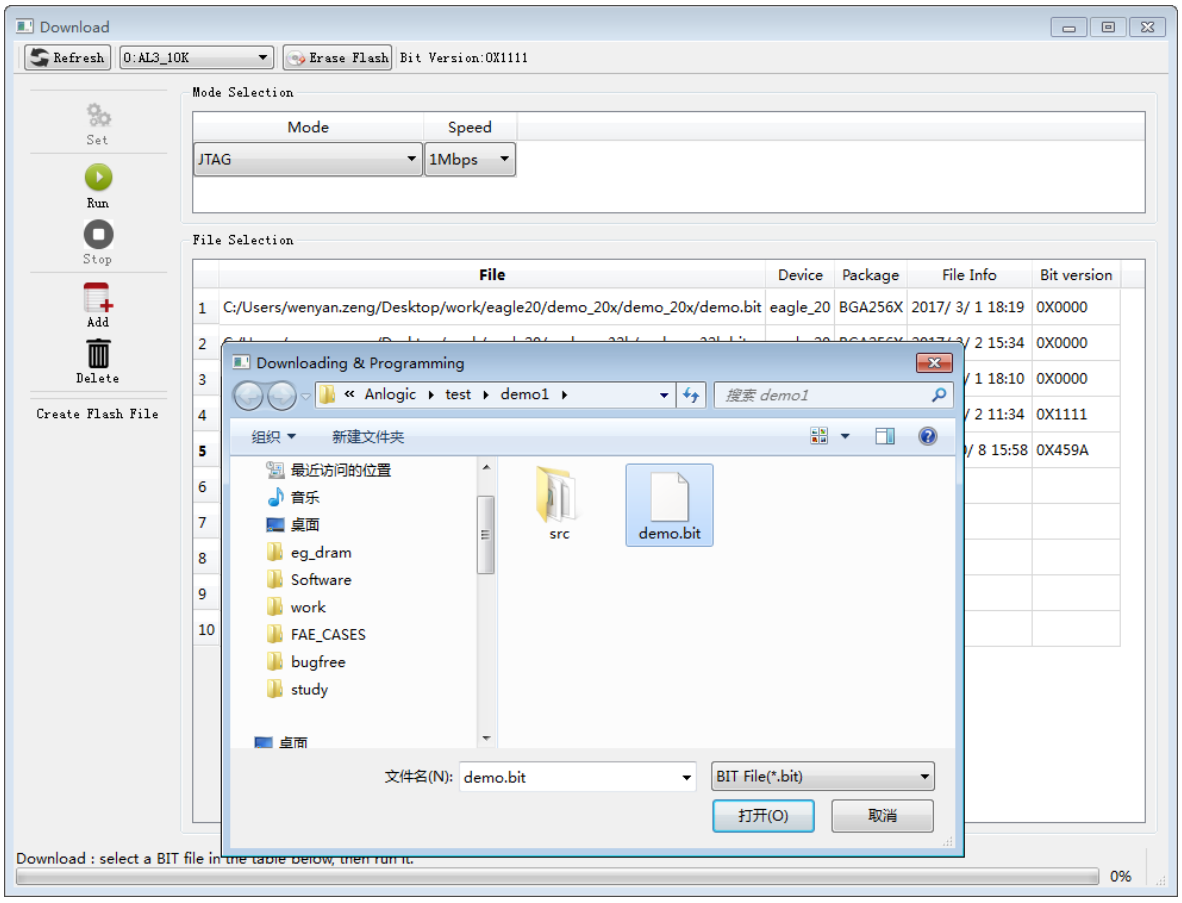
The specific simulation process in Modelsim can be found in the 10.3 Modelsim simulation process in this manual.

8 Download

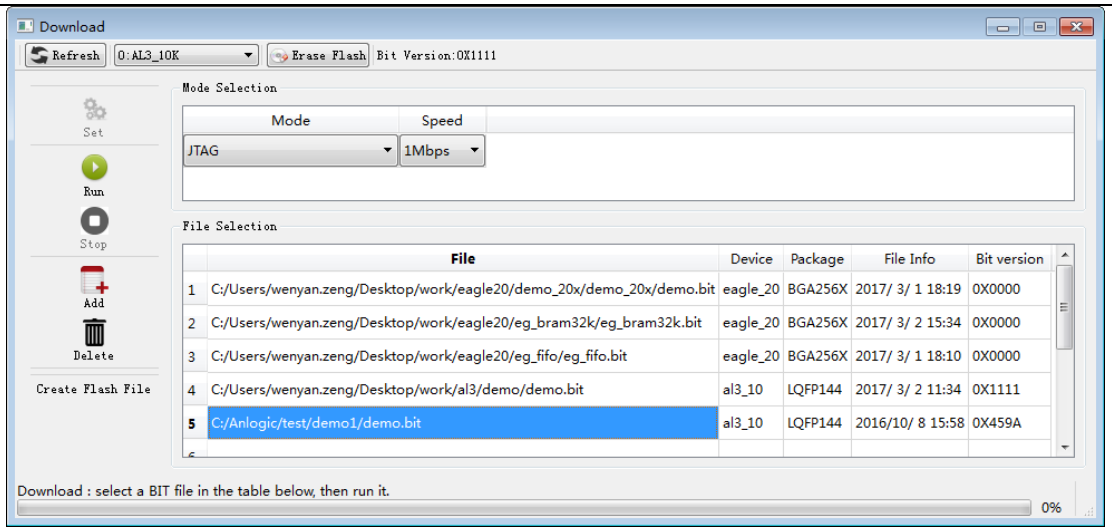
8.1 Introduction to the download process

Once the bitstream files have been successfully generated, they can be loaded into the configuration memory of the FPGA chip or SPI Flash memory.

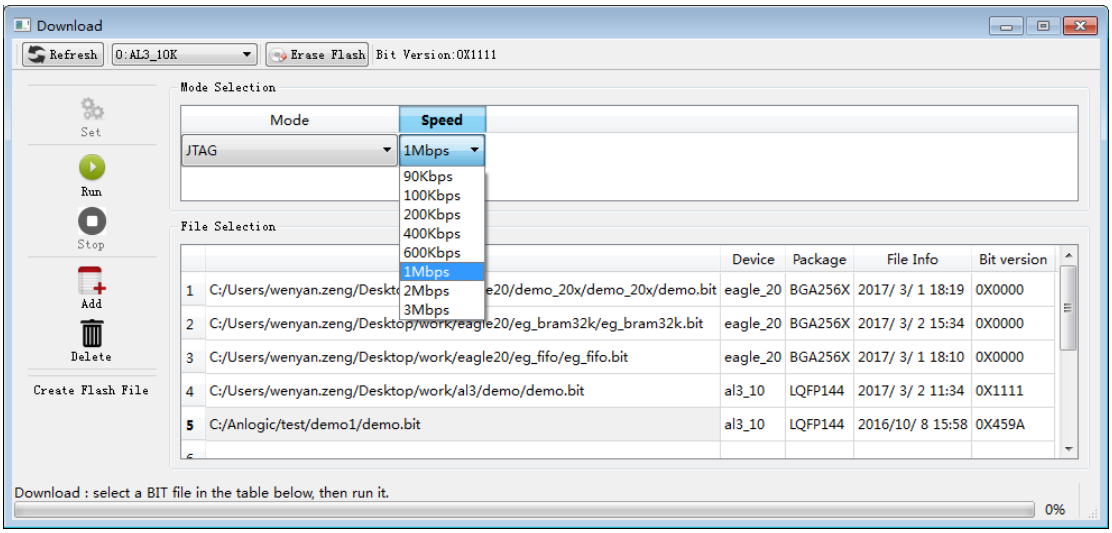
1. In the FPGA Flow panel, double-click Download
2. Add a bitstream file to download via Add.



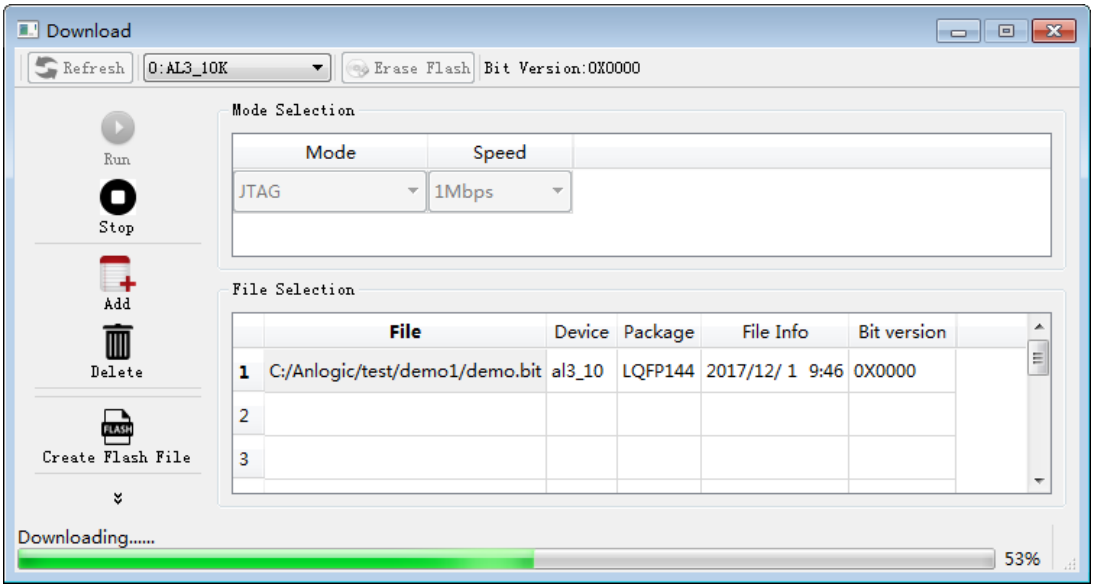
3. Select the appropriate bitstream file and click Run to download it.



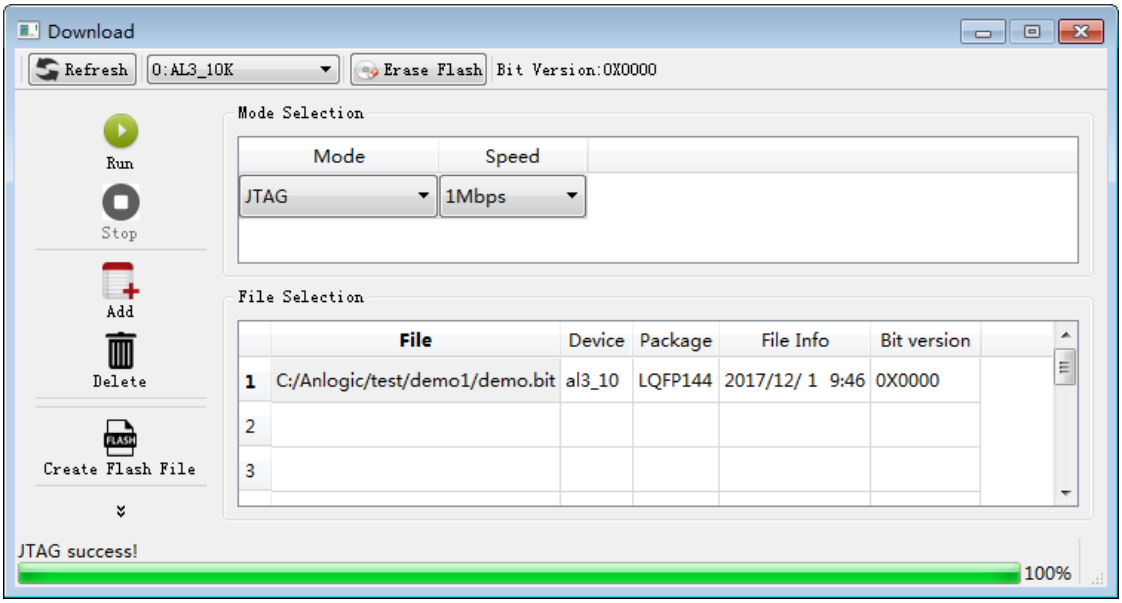
The download speed is divided into nine levels, 90Kbps is the slowest, 3Mbps is the fastest, and the default is 1Mbps.



The progress of the download can be viewed through the progress bar during the download process.



After the download is complete, a prompt to download success will be returned.



TD does not recognize the chip:

1. “No hardware”: The user does not install the USB driver correctly before downloading. For the USB download driver installation instructions, please refer to Appendix 10.4. When downloading, each interface is not connected correctly. Please check if there is any looseness at each interface, then click the Refresh button to refresh.

2. “USB Cable is connected”: When downloading, the FPGA chip or Flash chip is not recognized. Please check if the power of the board is turned on, and then click the Refresh button to refresh.

8.2 Bitstream file type

The bitstream files supported, downloaded, and downloaded in the TD software are as follows:

1. bit: bit file contains the complete chip configuration and bitstream information.

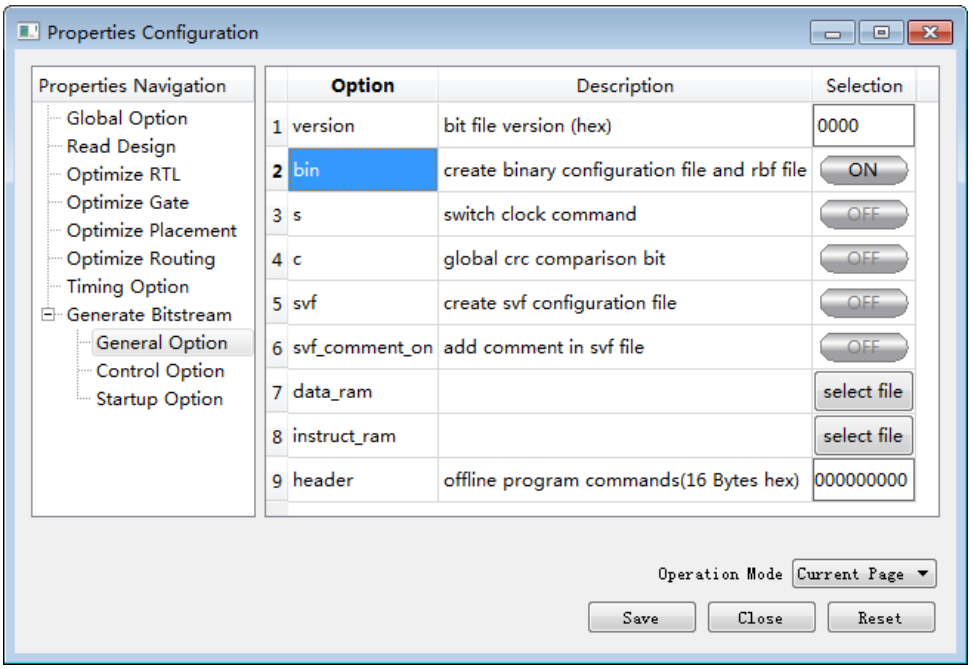
   Running Bitstream in the TD interface generates a bit file by default.

   The bit file can be used in any of the download modes supported by the TD.

2. bin: A pure binary file containing only bitstream information.

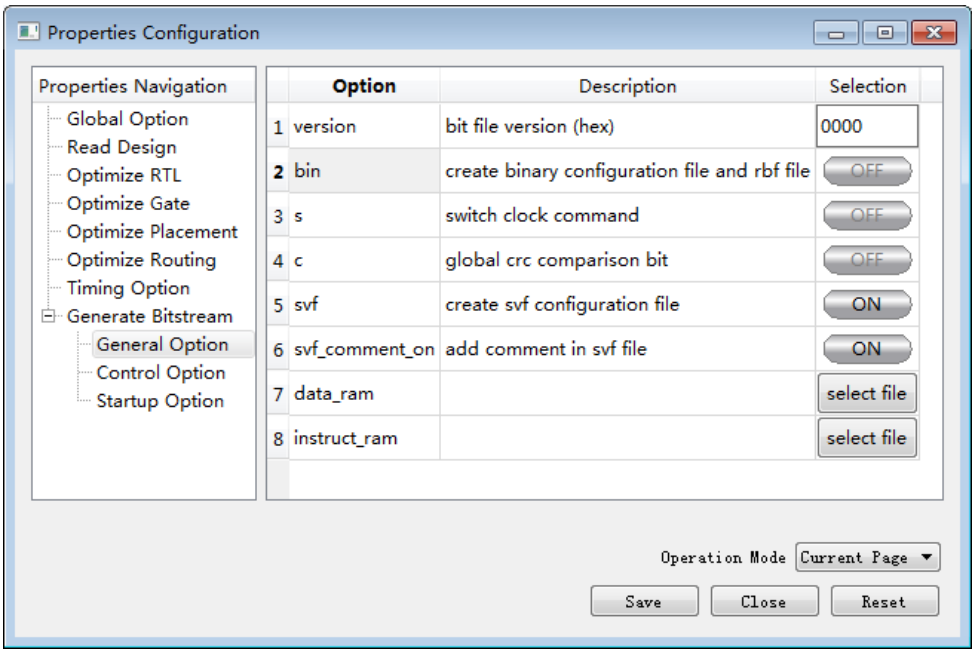
   In the Process→Properties→Generate Bitstream→General Option, set the value of the bin option to ON and save it. After running Generate Bitstream, the corresponding bin file will be generated in the project directory.

   The bin file is only available for offline downloader downloads, ie only the download mode is Direct Flash Write.



3. svf: Serial vector file. A uniform standard structure for shielding internal details.

In Process→Properties→Generate Bitstream→General Option, set the value of the svf option to ON, and svf_comment_on will be set to ON together. This option is used to set whether to print the comment in the svf file and save it.



The following six files are generated in the project directory after running Generate Bitstream:

1) *_sram.svf: Used to configure the SRAM inside the FPGA/CPLD. It will take effect immediately after configuration, the power loss will be lost, and the flash will not be updated.

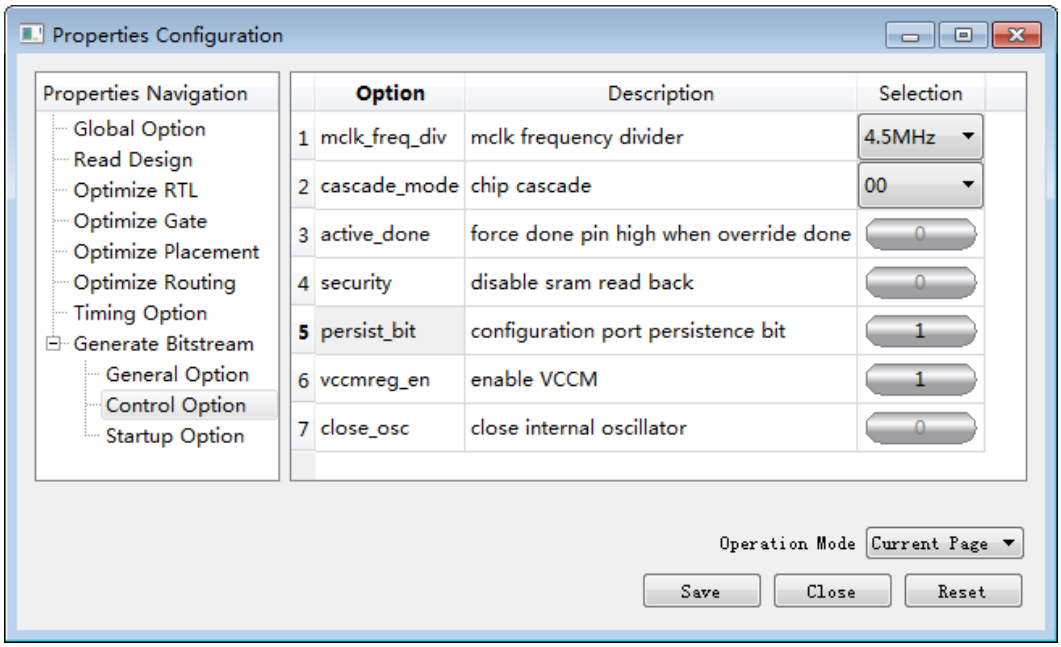
2) *_spi_bk.svf: Used to update the FPGA internal FLASH in the background mode. After the update is completed, it will be automatically loaded into the SRAM to start running. This function needs to be used with persist_bit in Process→Properties→Generate Bitstream →Control Option. The generated svf file will work only if persist_bit=1 in the current working mode.

3) *_spi_norefresh_bk.svf: Used to update the FPGA internal FLASH in the background mode. After the update is completed, it will not be loaded into the SRAM, which will not affect the current state of the FPGA/CPLD. This function needs to be used with persist_bit in Process→Properties→Generate Bitstream→Control Option. The generated svf file will work only if persist_bit=1 in the current working mode.

4) *_refresh.svf: Activates the instruction to reload the bitstream from flash. This svf file is used in conjunction with the *_spi_norefresh_bk.svf file, and when the background update is complete, use this svf to restart the new load process.

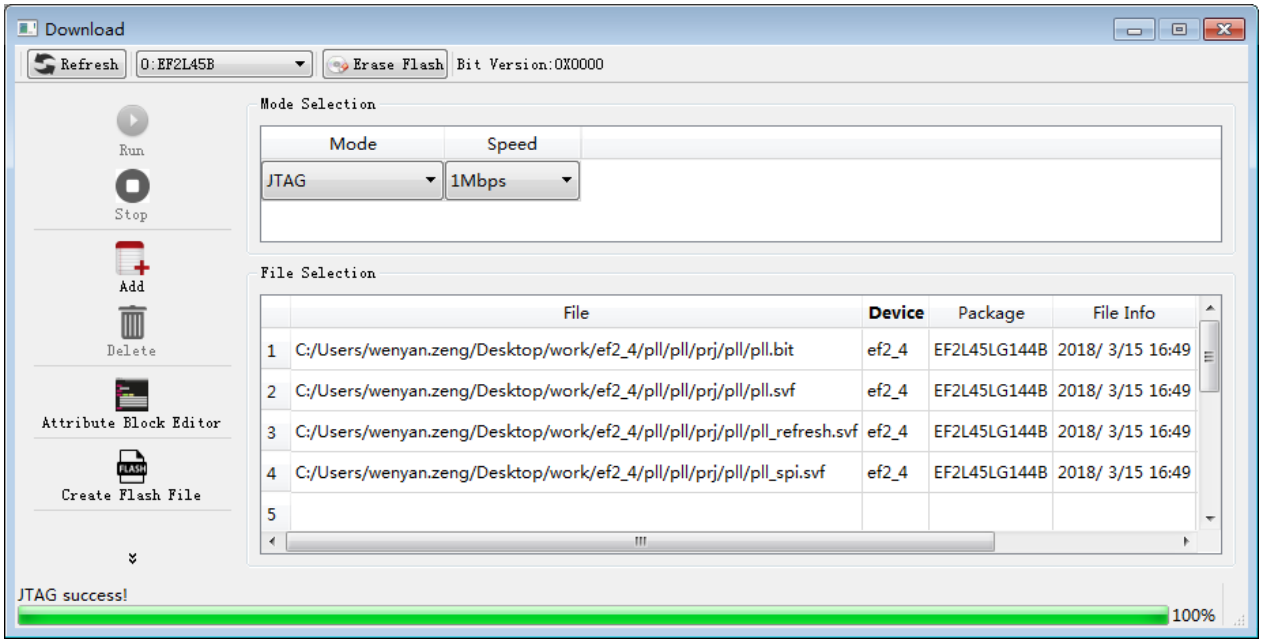
5) *_erase_spi.svf : Erases the spi instruction to erase the entire spi content.

6) *_readstatus.svf : Status check instruction to detect the value of the FPGA/CPLD internal status done register.



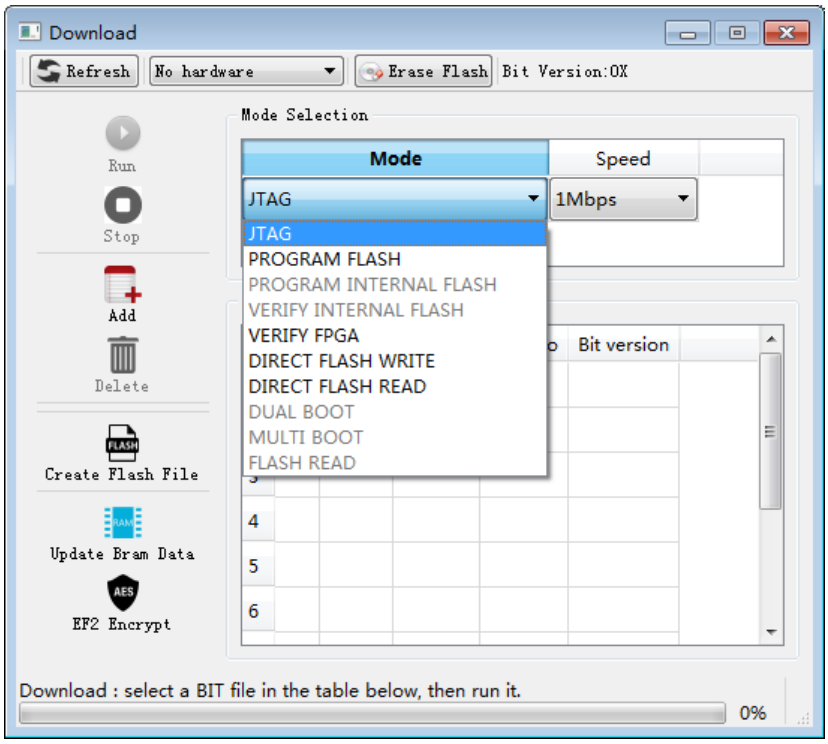
svf file is only used for JTAG downloads. The download process is as follows:

*_sram.svf file is downloaded with JTAG and the contents of the FPGA are updated immediately. If you don't want to update your current work, you can download *_spi_norefresh_bk.svf first and wait for it to be needed. *_refresh.svf Start the on-chip preloading function.



8.3 Download mode

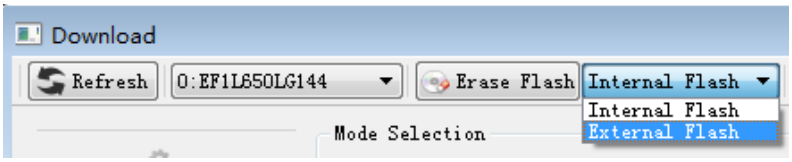
TD provides the following download modes for users to choose from: JTAG, PROGRAM FLASH, PROGRAM INTERNAL FLASH, VERIFY INTERNAL FLASH, VERIFY FPGA, DIRECT FLASH WRITE, DIRECT FLASH READ, DUAL BOOT, MULTI BOOT, FLASH READ.



1. JTAG mode: The downloaded bit file will not be saved to the flash. The configuration bit information is directly stored in the control chip of the FPGA chip. After the board is powered off, the configuration bit information is completely lost.

2. PROGRAM FLASH mode: The bit file will be saved to the external Flash chip. After the board is powered off and restarted, the FPGA chip automatically reads the bit stream information stored in the Flash chip. If you want to erase the bit stream information in the flash, click the Erase Flash button. The erasure time depends on the device parameters of the flash chip.

   For ELF series devices, this feature is used for external FLASH downloads. When erasing the external FLASH, you need to select Erase External Flash.



3. PROGRAM INTERNAL FLASH mode: Only ELF series devices are supported for internal flash download. When erasing the internal FLASH, you need to select Erase Internal Flash.

4. VERIFY INTERNAL FLASH mode: Only ELF series devices are supported, which are used to compare the configuration file in the internal flash with the information in the bit file currently selected by the user.

5. VERIFY FPGA mode: It is used to compare the configuration bit information in the FPGA chip with the information in the currently selected bit file of the user. It is best to use it together with the mask file (.bmk) to ensure that the bit stream file and the mask file are in the same Inside a folder.

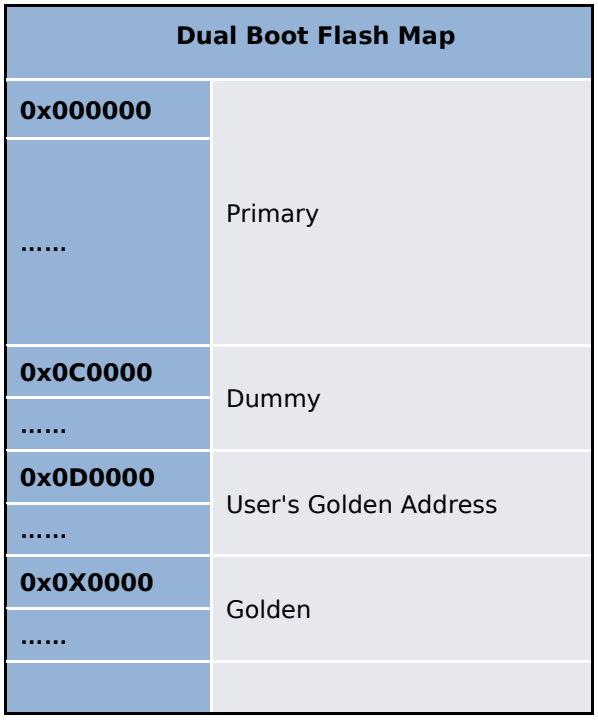
6. DIRECT FLASH WRITE mode: directly write data into the specified address area of FLASH without going through the FPGA. The hardware needs to be connected directly to the FLASH signal line. For offline downloads, only the bin file is supported for download.

7. DIRECT FLASH READ mode: The data of the specified area is read directly from the FLASH without going through the FPGA, and exists in the specified file. This mode also needs the downloader to directly connect with the signal line of the FLASH.

8.3.1 Dual Boot

For Eagle and EF2 series FPGAs, Dual Boot (Double Boot Mode) and Multi Boot (Multi Boot Mode) are supported when program loading external flash.

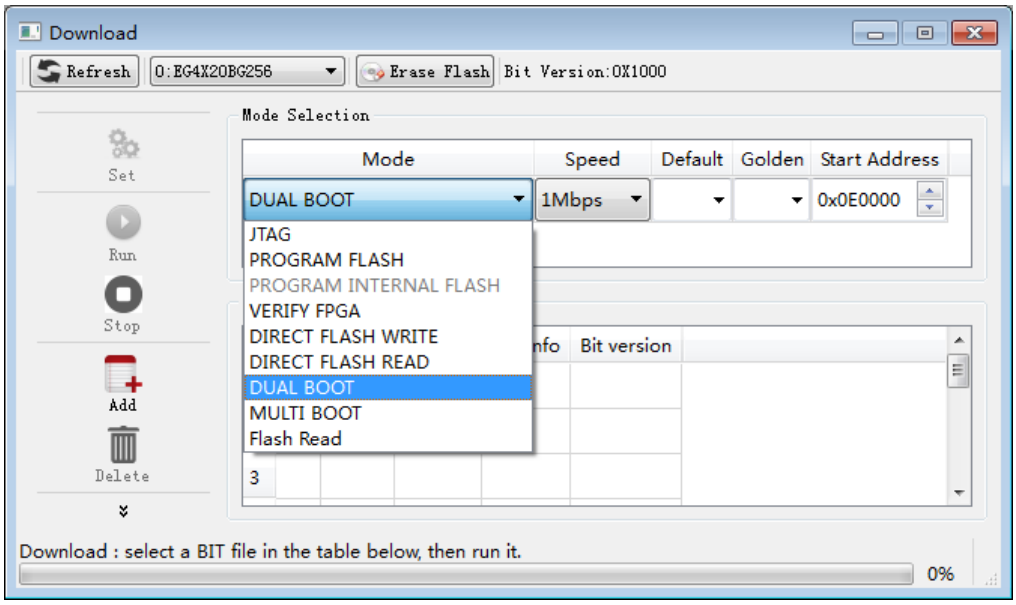
Dual boot mode means that two sets of FPGA bitstreams are stored in SPI FLASH. After power-on, the FPGA first loads the Primary bitstream. If the primary bitstream error causes the load to fail, the Golden bit is loaded according to the Golden Address. flow. In dual boot mode, the data space distribution is shown in the following figure:



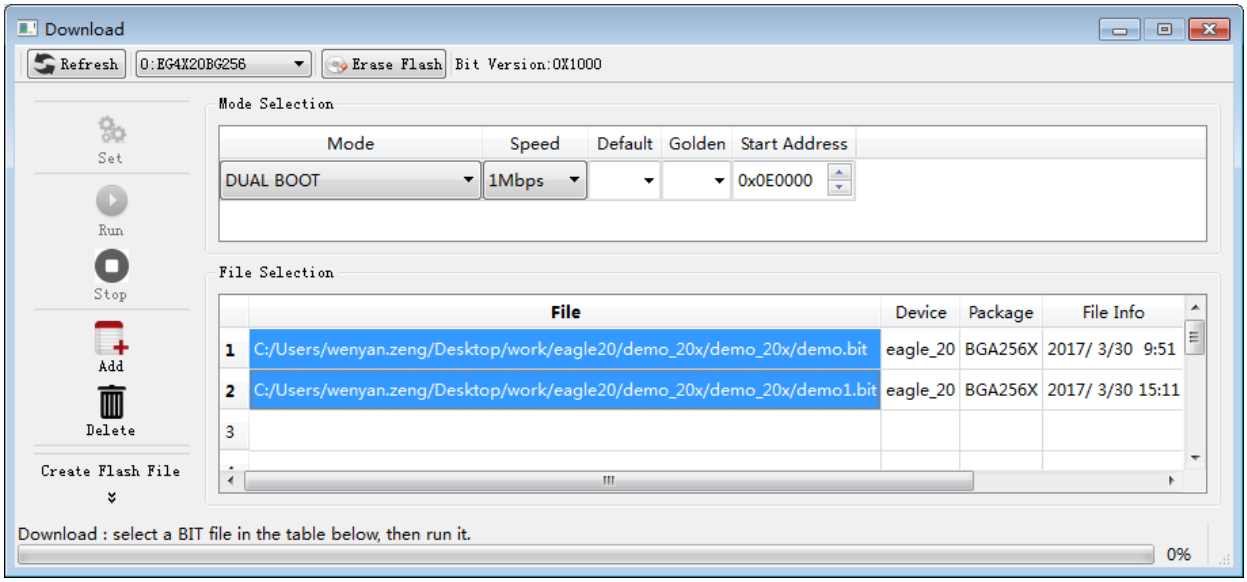
Eagle series FPGAs support dual boot mode by default, that is, the program will be loaded from the 0 address of FLASH by default after power-on. If the bit stream starting at address 0 is destroyed and the FPGA program fails to load, the FPGA will read to address 0X0D0000. Take the jump address and then load the bit stream from the specified jump address.

The steps to use dual boot mode are:

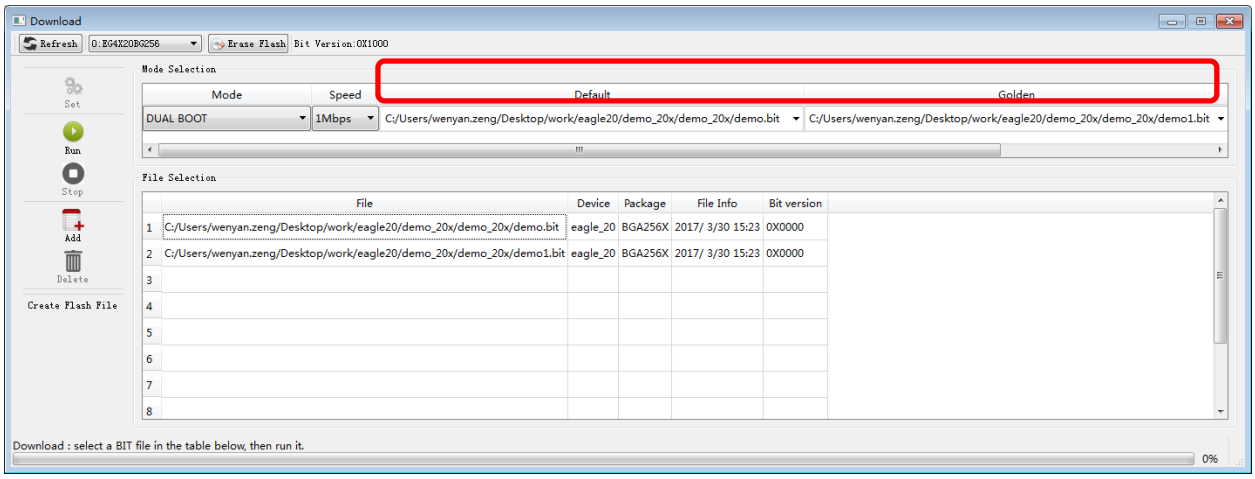
1. **In the Download interface, select the download mode as Dual Boot;**



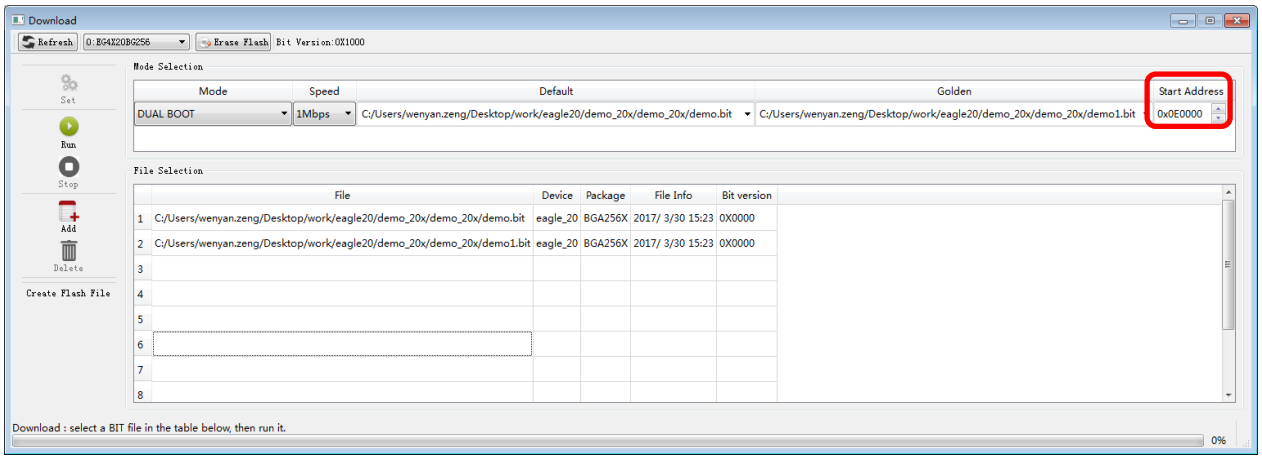
2. Add two bitstream files that need to be downloaded to FLASH via the Add button;



3. Set the Default bit stream (Primary address segment) and the Golden bit stream (Golden address segment);



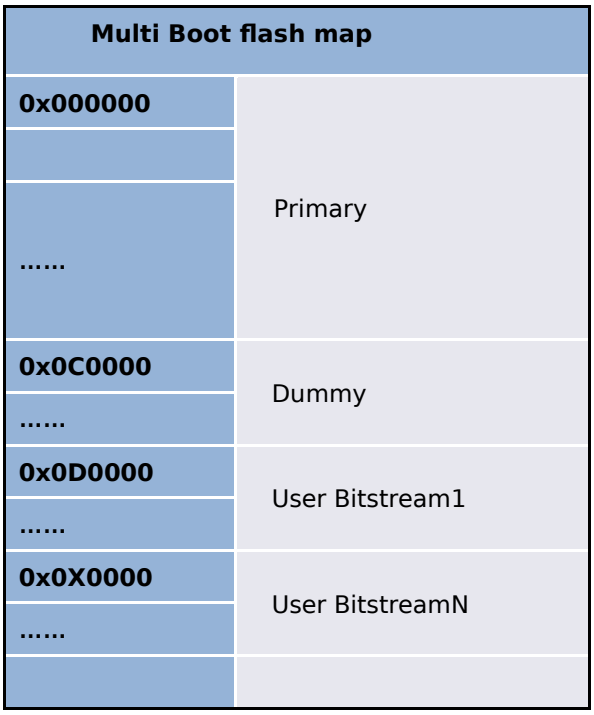
4. Set the starting address where the Golden bit stream is stored. Note that the Golden storage address can only be greater than 0X0D0000, the default is: 0X0E0000;



5. Click Run to download the bitstream.

8.3.2 Multi Boot

Multi-boot mode means that the user can store two or more FPGA bitstreams in the SPI FLASH. After power-on, the FPGA first loads the Primary bitstream, and then in the FPGA code of the Primary bitstream, the FPGA can be controlled to load the bitstream from the specified address. In multi-boot mode, the data space distribution is as shown below:

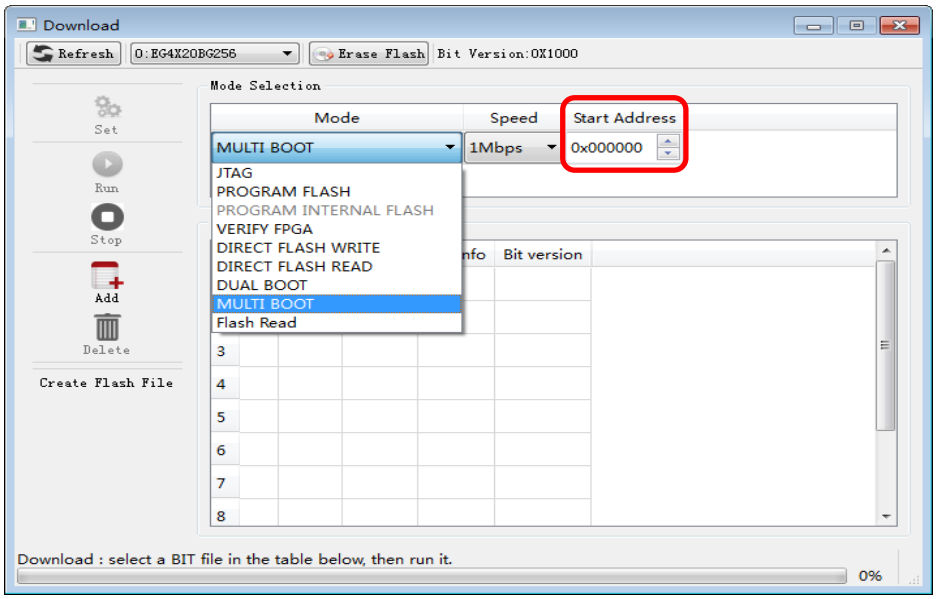


Before using multi-boot mode, the user needs to call the following IP unit in the code:

EG_LOGIC_MBOOT　#(“DYNAMIC”,8’h00) mboot(rebootn, dynamic_addr);

Among them, dynamic_addr is an 8-bit FLASH address, which is the upper 8 bits of the 24-bit FLASH address. After setting the address, the FPGA program can be reloaded from the specified address of dynamic_addr by giving rebootn a low pulse.

In the TD Download interface, you need to set the download mode to MULTI BOOT and specify the Start Address of the bit file used for the jump. This address needs to be the same as dynamic_addr.

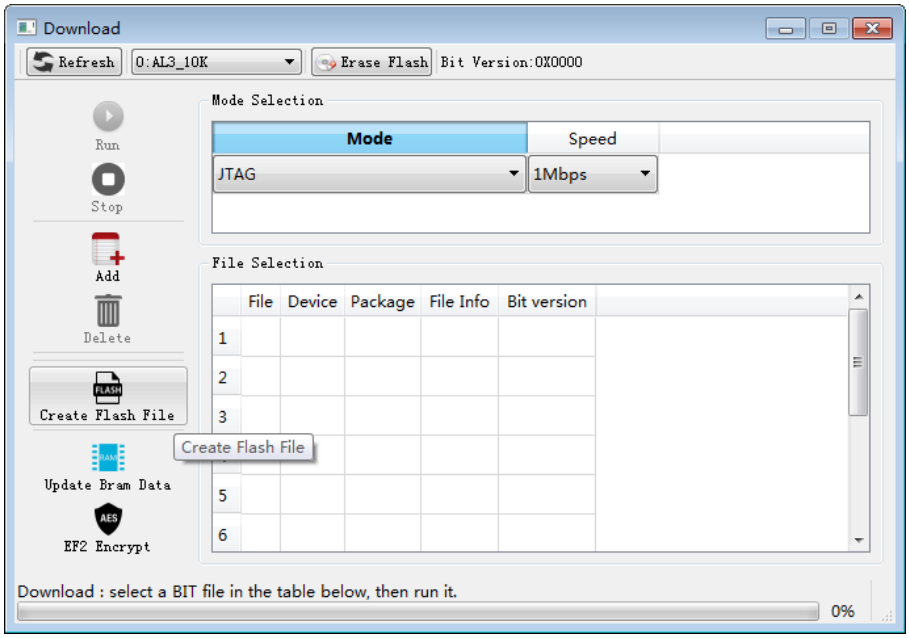

Download
Refresh
0:EG4X20BG256
Erase Flash
Bit Version:0X1000
Mode Selection
Set
Run
Stop
Add
Delete
Create Flash File
Mode
Speed
Start Address
MULTI BOOT
1Mbps
0x000000
JTAG
PROGRAM FLASH
PROGRAM INTERNAL FLASH
VERIFY FPGA
DIRECT FLASH WRITE
DIRECT FLASH READ
DUAL BOOT
MULTI BOOT
Flash Read
Bit version
3
4
5
6
7
8
Download : select a BIT file in the table below, then run it.
0%

8.4 Extensions

8.4.1 Create Flash File

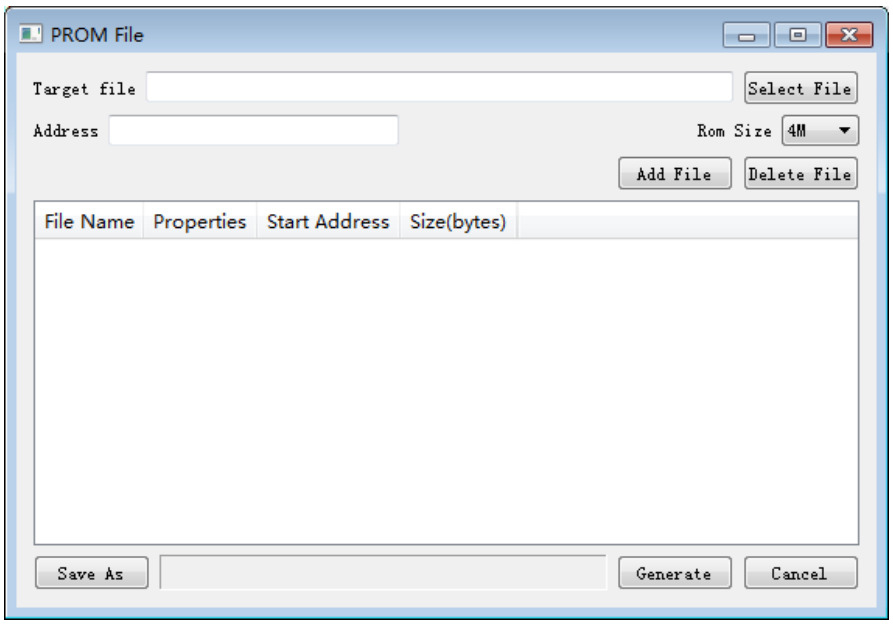
The TD software provides the Create Flash File function, which allows users to add custom content to the target bitstream file, and expand the function of the existing bitstream file. It needs to re-edit the source code and save a lot of time. The specific operations are as follows:

1. **Open the Download interface and click "Create Flash File";**

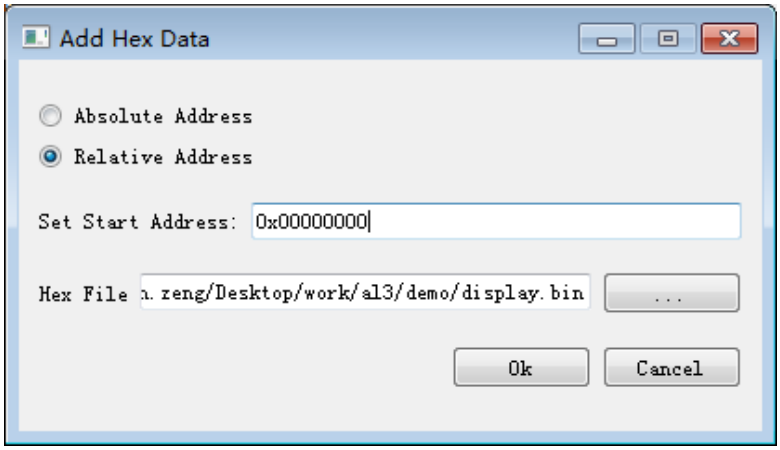


2. Click “Select File” to add the Target File. The target file can be a bit file or a bin file. When the target file is added, the size of the file, that is, Address, is displayed accordingly.

   Rom Size refers to the capacity of the target Flash, that is, the last generated file cannot exceed this capacity, otherwise it will fail when Generate.

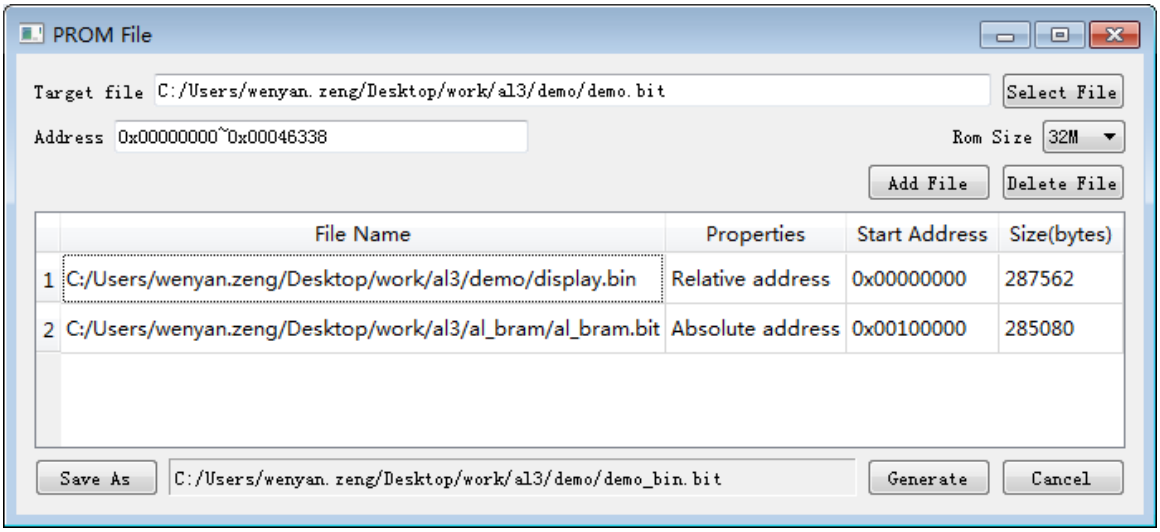


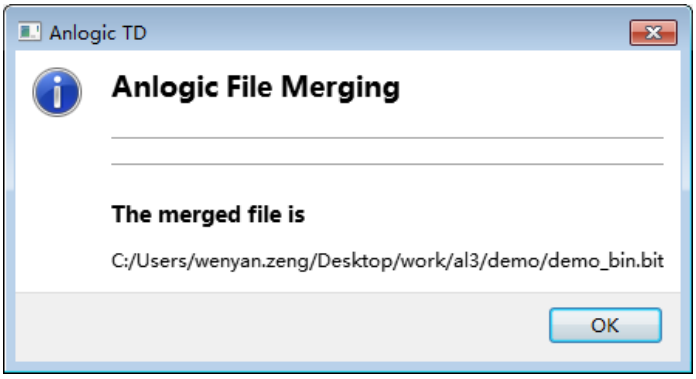
3. Click "Add File" to add a file, which is called a merge file. The merge file can be a bin file, a hex file or a bit file. If you select Absolute Address, you need to set the starting address to be larger than the Target File size. Otherwise, if you add a file, the content of the Target File will be overwritten. If you select Relative Address, it refers to the size of the Target File, and then add it, such as setting. The starting address is 0x00, and the file is added immediately after the Target File.



4. Multiple merge files can be added at the same time, just set the start address reasonably according to the size of each file.

5. Select the path to generate the file, click Generate, the following prompt will be given, otherwise the corresponding error will be given.


PROM File
Target file
C:/Users/wenyan.zeng/Desktop/work/al3/demo/demo.bit
Select File
Address
0x00000000~0x00046338
Rom Size
32M
Add File
Delete File
File Name
Properties
Start Address
Size(bytes)
1
C:/Users/wenyan.zeng/Desktop/work/al3/demo/display.bin
Relative address
0x00000000
287562
2
C:/Users/wenyan.zeng/Desktop/work/al3/al_bram/al_bram.bit
Absolute address
0x00100000
285080
Save As
C:/Users/wenyan.zeng/Desktop/work/al3/demo/demo_bin.bit
Generate
Cancel

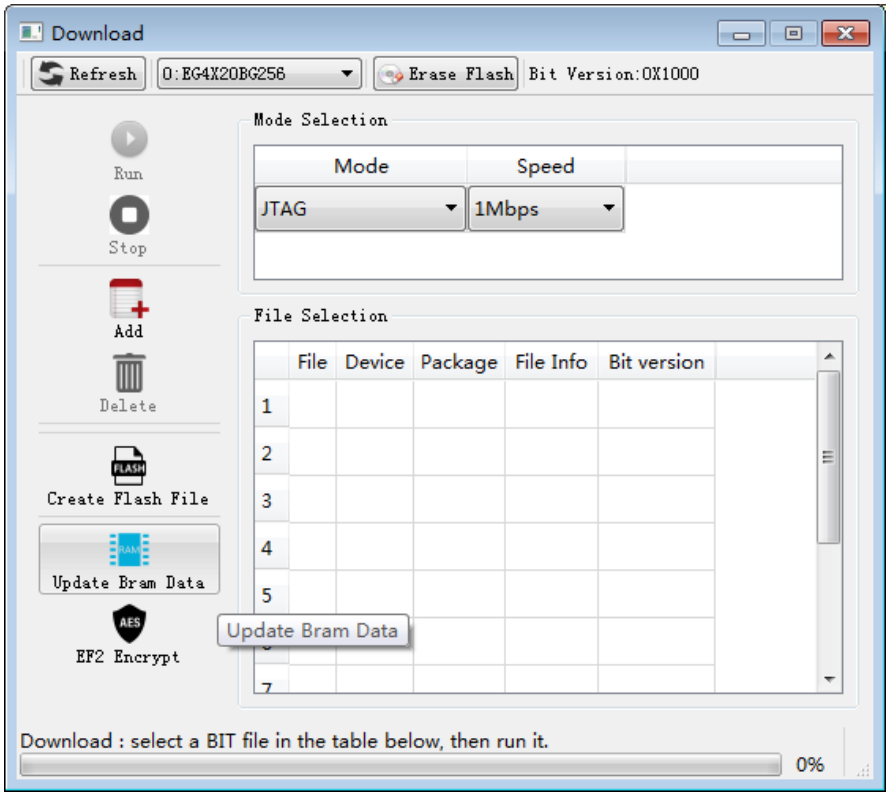

Anlogic TD
Anlogic File Merging
The merged file is
C:/Users/wenyan.zeng/Desktop/work/al3/demo/demo_bin.bit
OK

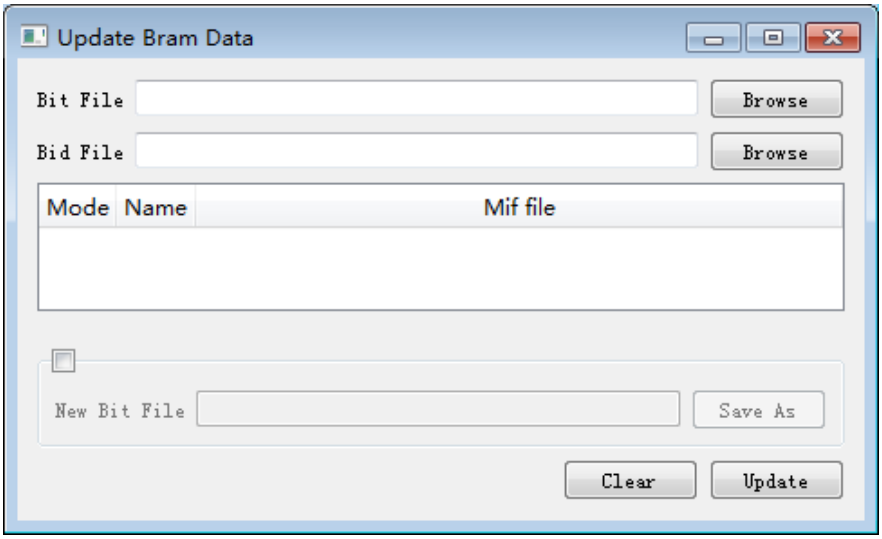
8.4.2 Update BRAM Data

TD provides the Update BRAM Data tool. If you only need to update the initial value of BRAM in the design, you can directly generate a new bitstream file by modifying the data segment of the BRAM in the bitstream file without recompiling the project, saving a lot of time.

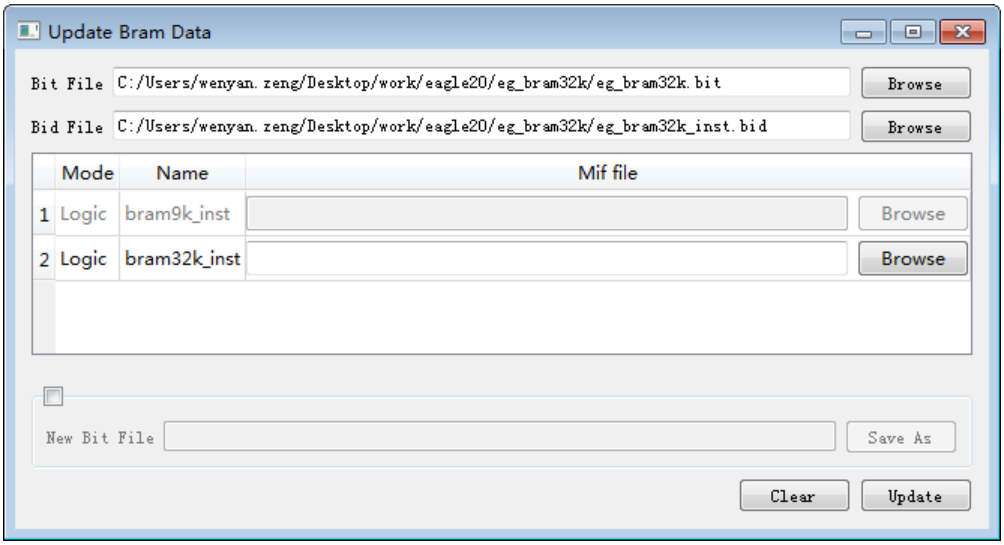
The specific operations are as follows:

1. Open the Download interface and click on "Update Bram Data";

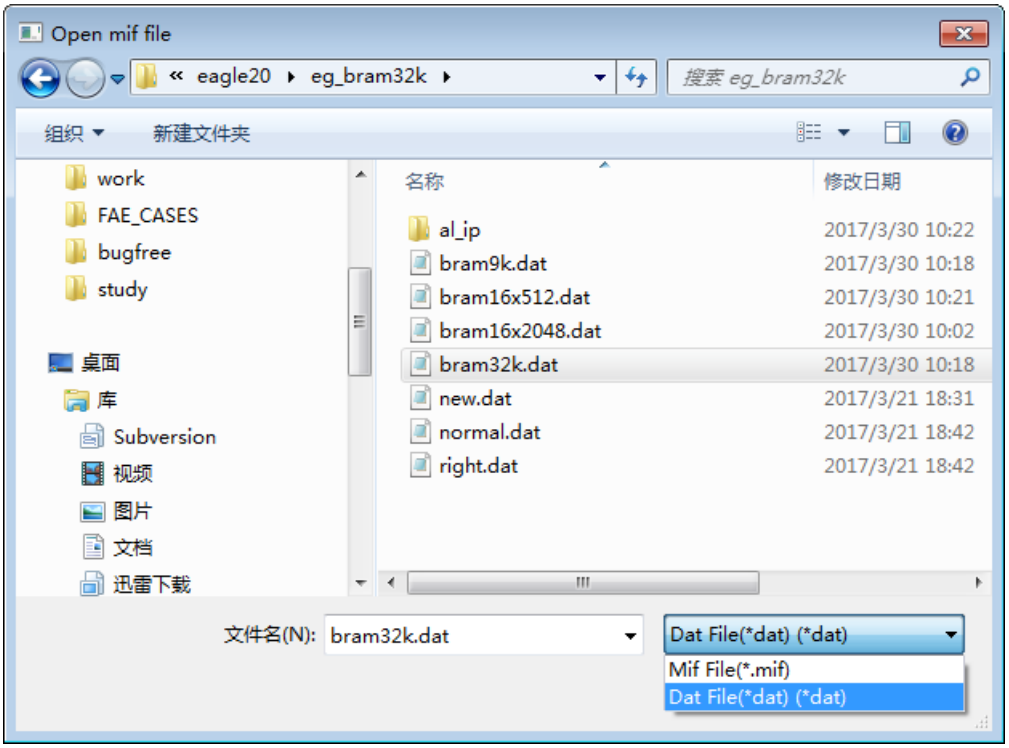




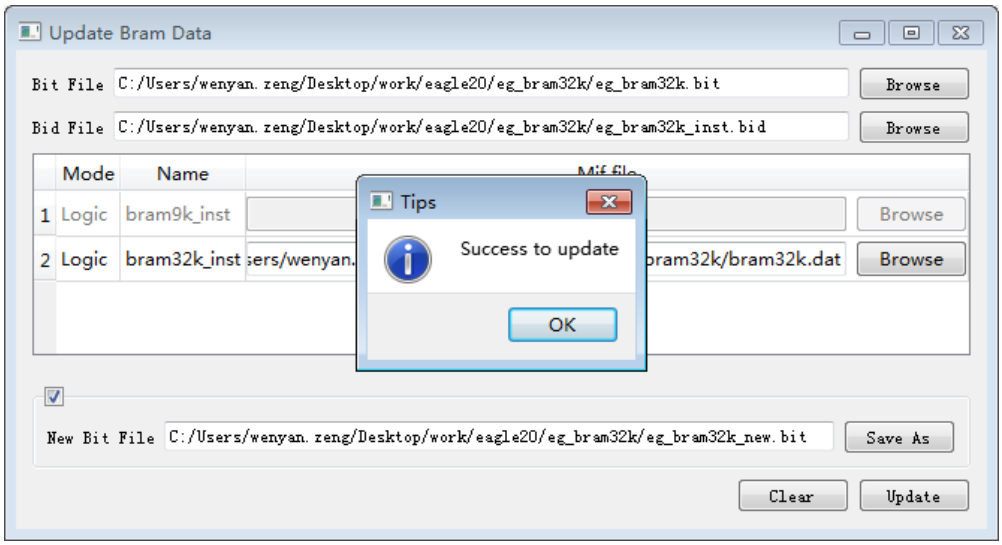
2. Selecting the Bit File to be updated and the Bid File describing the BRAM will display the Logic BRAM in that bitstream accordingly. Only the BRAM with the initialization file added at the beginning of the design can be updated here; the BRAM without the initialization file is grayed out, as shown in the following figure, bram9k_inst;



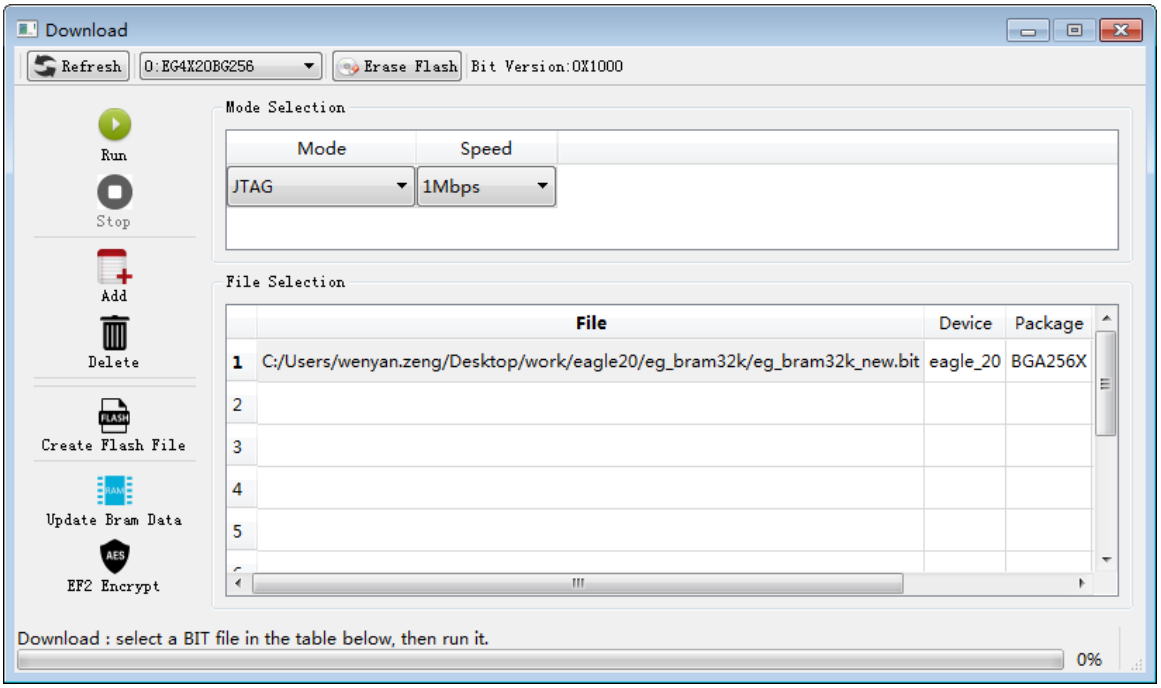
3. Click "Browse" to add a new BRAM Data. The files that can be added are .dat files and .mif files. Here, the size of the new BRAM Data needs to be the same as the size of the BRAM in the design, otherwise it will give a warning and fail to achieve the expected function;



4. You can click Update to directly update the current bit file, or you can use Save As to generate a new bit file.



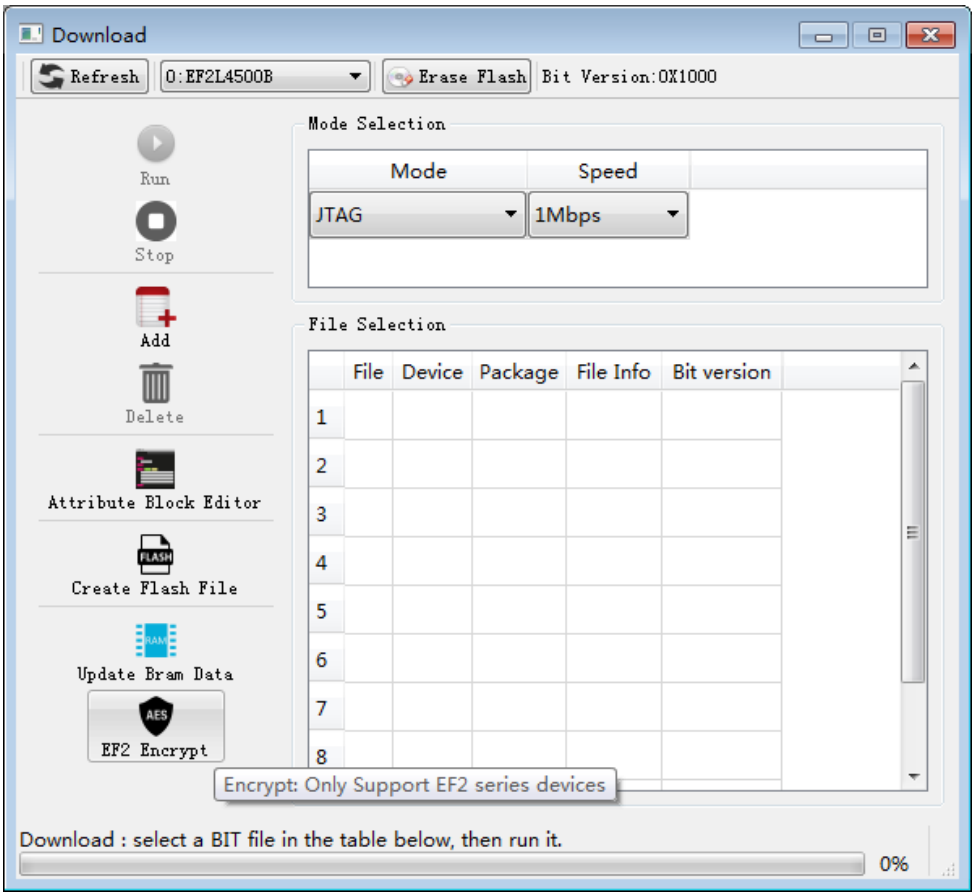
5. Click OK to complete the update and add a new bit file to the Download interface for download.

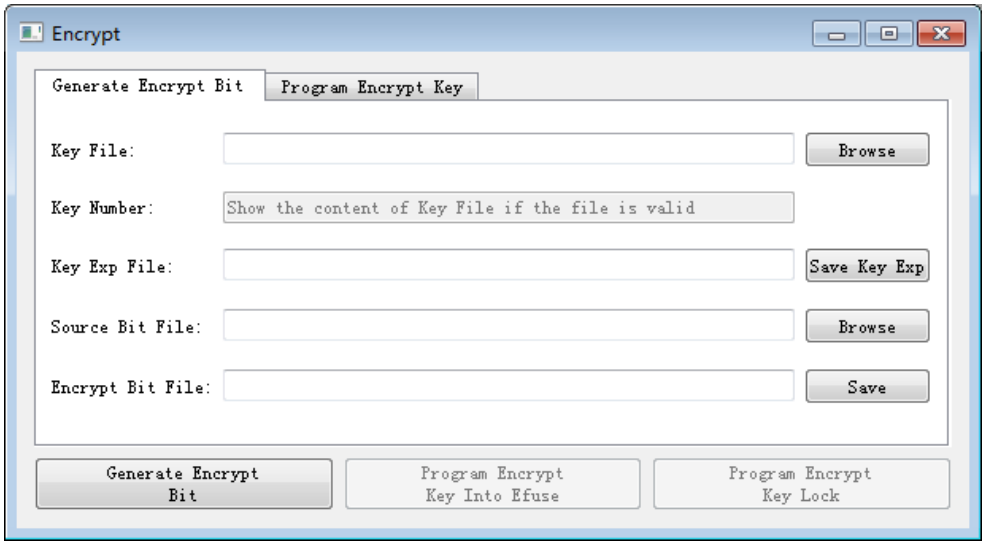


8.4.3 EF2 Encrypt

EF2 series devices, TD software supports 128Bit AES encryption for bitstream files. The encrypted file must be decrypted by the user-provided key.

Open the Download interface, click the EF2 Encrypt button, the following interface appears:





The encryption steps are as follows:

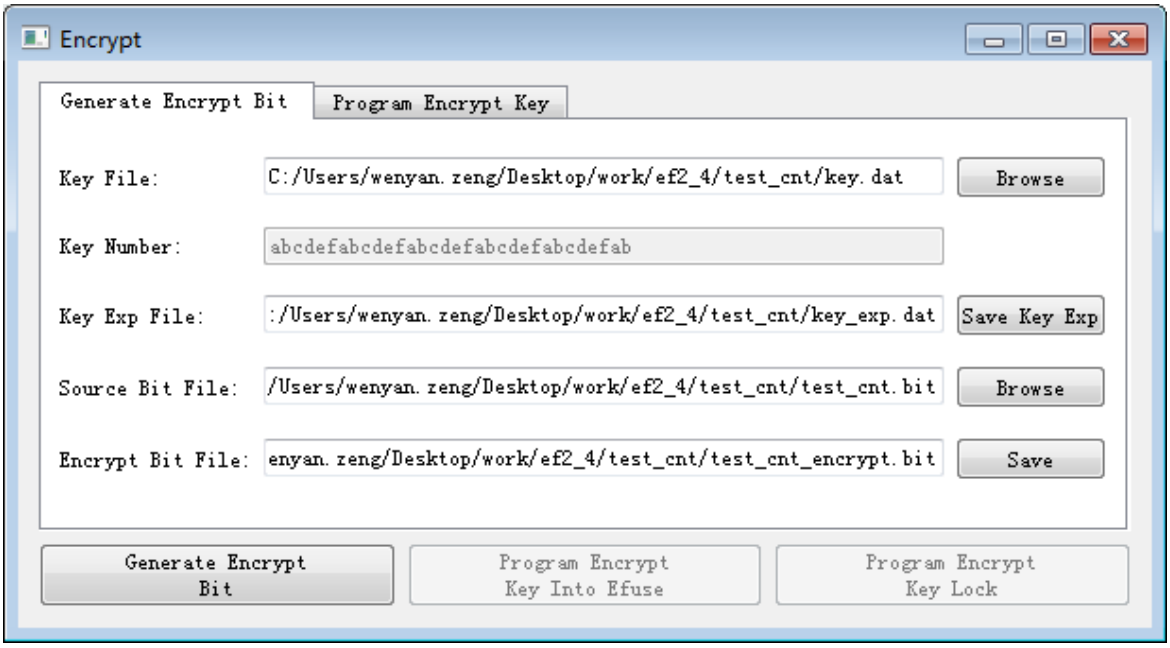
1. Select the user secret key file in Key File. The user key is 32 hexadecimal numbers. Once the key is selected, it will be displayed in the Key Number.

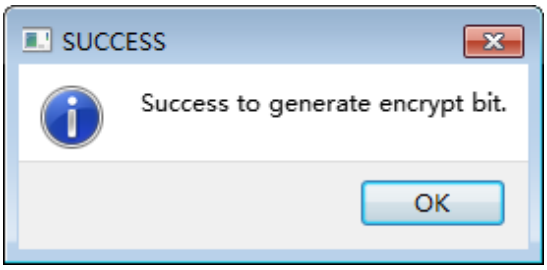
2. Select the location where the encrypted key file is saved in Key Exp File.

3. Select the EF2 bitstream configuration file (*.bit file) to be encrypted in the Source Bit File.

4. Select the location where the encrypted bitstream configuration file is saved in Encrypt Bit File.

5. Click the Generate Encrypt Bit button to generate an encrypted file. If the encryption is successful, it will give a prompt, otherwise the encryption failure will give the corresponding error in the TD interface.





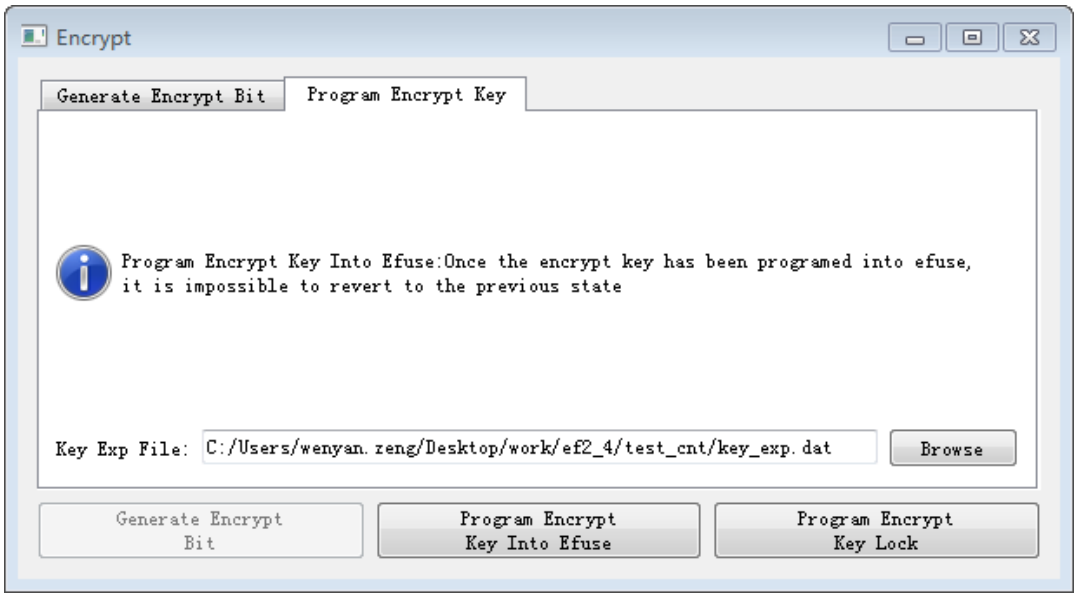
After the encrypted file is generated, the content of the encrypted key file needs to be written into the EF2 chip. The programming process is as follows:

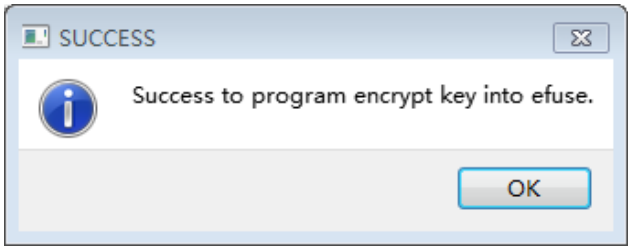
1. In the Encrypt interface, select the Program Encrypt Key interface.

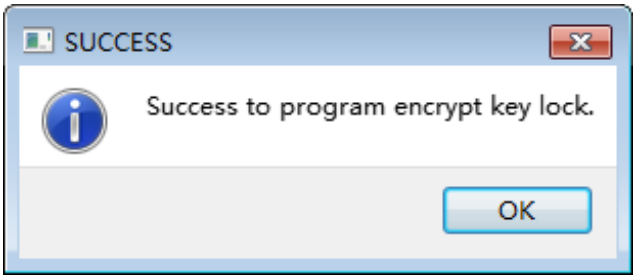
2. Select the key key file exp_exp.dat that you just generated in Key Exp File.

3. Click the Program Encrypt Key Inot Efuse button to burn the encryption key.

4. After burning the encryption key, click the Program Encrypt Key Lock button, the encryption key will be locked by the hardware, and the encryption key will not be read from the EF2 device.







8.5 Offline downloader

8.5.1 Introduction to the offline downloader

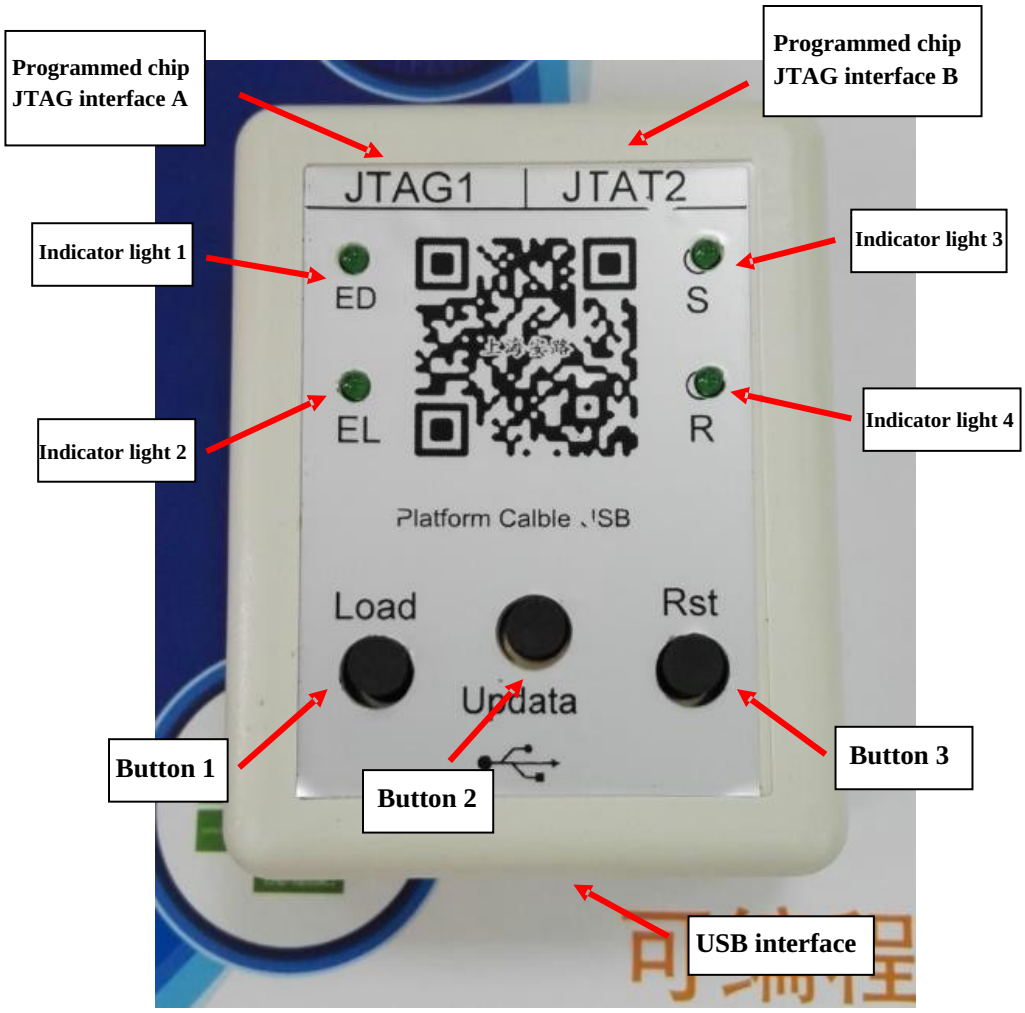
The offline downloader supports both online JTAG program download (compatible with traditional downloader), online FLASH direct reading and writing, and offline FLASH program programming. Among them, the offline FLASH program programming mode is divided into the following three modes:

1. support burning SPI FLASH through JTAG

2. support direct burning SPI FLASH

3. support user-defined programming protocol

The hardware of the offline downloader is as follows:



The working status of the offline downloader is as follows:

After the power is turned on, the offline downloader is in the online JTAG download mode. At this time, the downloader is like a normal downloader, and the program download of the target FPGA can be realized by connecting the JTAG interface of the chip to the JTAG of the FPGA on the target board. Includes online JTAG debugging and target board FLASH online download. In this state, the indicator 1, indicator 2, and indicator 3 are off, and indicator 4 is blinking.

When button 2 is pressed, the offline downloader enters the FLASH direct read/write mode, and the PC software can read and write the content of the source FLASH used in the offline mode on the offline downloader. In this mode, the indicator 3 flashes and the indicator 4 remains flashing.

When button 1 is pressed, the offline downloader enters the offline download mode, and will no longer be controlled by the PC software. The offline downloader will automatically read the 16-byte header of the source FLASH, and then enter three offline downloads according to the byte header. One of the modes, then FLASH copy. In this mode, the indicator light 4 is always on. When the indicator light 1 is on, it means that the ID of the target FLASH is detected incorrectly, and the target FLASH may not be detected. When the indicator light 2 is on, it means that the error is detected when the FLASH data is detected. Bright means that the FLASH copy is completed correctly. At this time, you can exchange the next chip, and press button 1 again to continue the next programming.

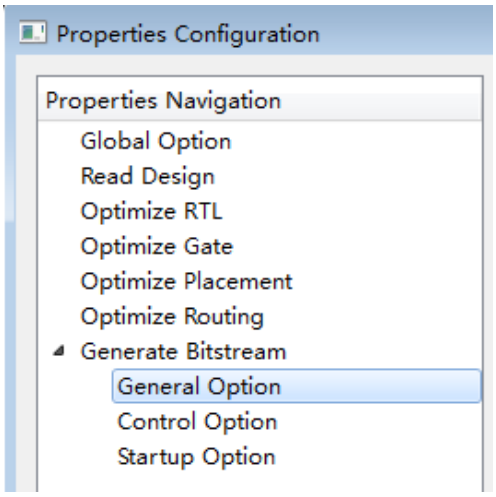
In any mode, the offline downloader can be returned to the power-on initial mode by pressing button 3, which is the online JTAG download mode.

8.5.2 Offline downloader steps

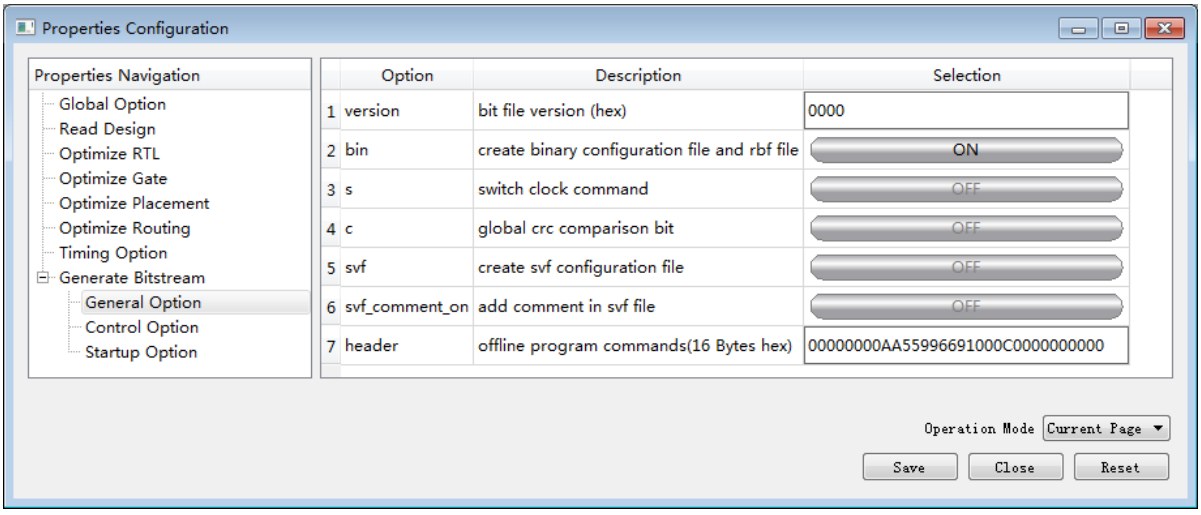
1. Generate a BIN file that can be downloaded to the source FLASH

The BIN file is characterized by the addition of a 16-byte header based on the bit file run by the target FPGA. These 16-byte headers can be added by setting the software before compiling the project. The specific steps are as follows:

Step one, open the software property settings window Properties→General Option



Step two, set the value of the bin option to ON, and add a 16-byte header to the header column.



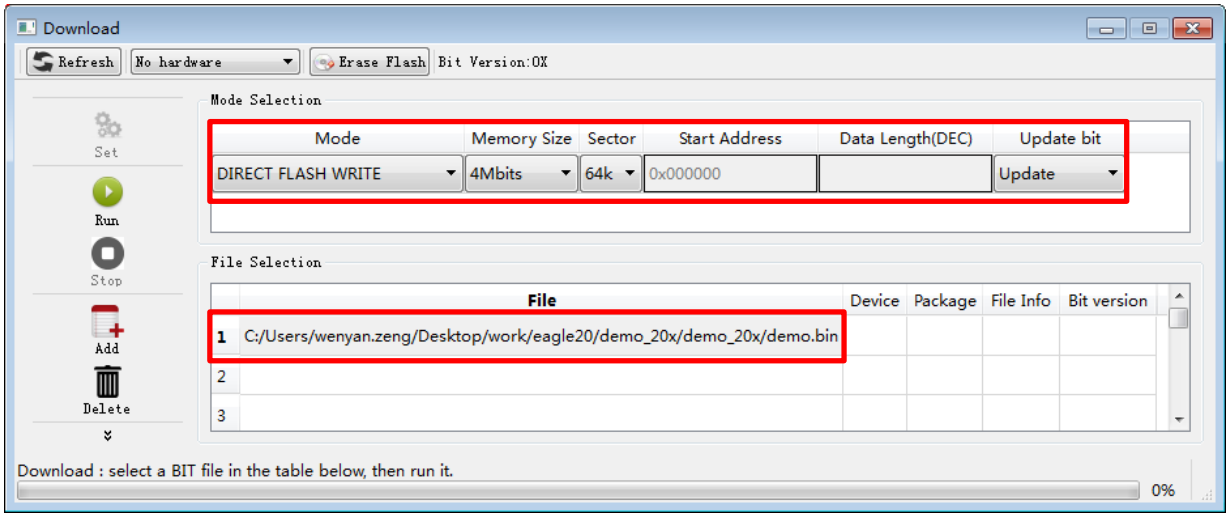
The definition of the header format is as follows:

Byte	Example (hexadecimal)	meaning
0~3	0000_0000	Fill bit
4~7	AA55_9966	Identifier
8	11	Command word: Bit7~6 Select the erase command: 01： sector erase 4K ~50ms 10: block erase 64K~0.2s 11: chip erase full chip ~1.5s Bit5~4 Select whether to read back the check: 01：Read back 10：Not reading Bit3 selection is an automatic connection test: 0：Not automatically tested 1：Idle state automatic test connection Bit2:0 select programming mode: 001： Burn SPI FLASH with AL3 010： Direct burning SPI FLASH 011： Custom
9~10	800	Burning length, number of pages, minimum 1 page, low byte first
11~12	400	Burn start address, number of pages, 0 address corresponds to 0, low byte first
13~15	00_0000	Fill bit

After filling in the header according to the form, save the settings, recompile the target project, and generate a BIN download file containing the setup header in the project path.

2. Download the generated BIN file to the source of the offline downloader FLASH

After the compilation is complete, open the Download interface of the software and set the contents in the red box below. Note that Start Address is set to 0; Data Length is the size of the BIN file, expressed in decimal, in Bytes; Sector is the Sector size of the chip being programmed, usually 64K. Select the BIN file you just generated, press the middle button of the offline downloader (button 2), and then click Run to start burning the source FLASH on the offline downloader.

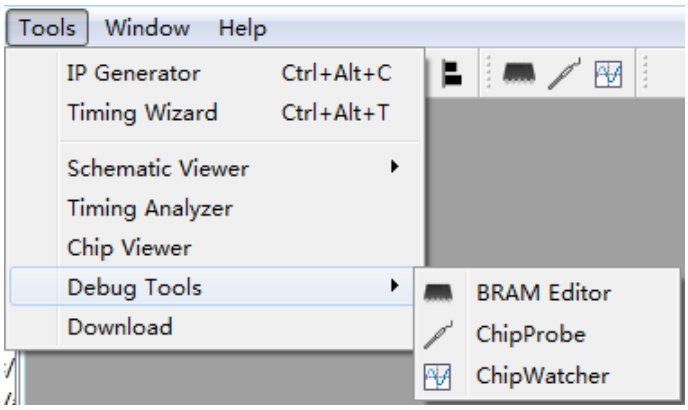


3. Download the FLASH of the target FPGA

Connect the offline downloader to the JTAG (or the corresponding port) of the target FPGA, and press the button 1 on the offline downloader to start the program download of the FLASH of the target FPGA. After successful download, the indicator 3 lights up and the buzzer sounds 2s. After downloading, press button 1 repeatedly to download the FLASH of the target FPGA repeatedly. If the indicator 1 is on during the download, the target FLASH ID is incorrect; the indicator 2 is on, indicating that the data comparison error. If there is an error in the download process, the buzzer will continue to sound until the reset button (key 3) is pressed.

9 Toolset

There are many tools in the TD software to help users better analyze the project, mainly: Schematic Viewer, ChipViewer, and three tools in the debugging toolset: BramEditor, ChipProbe, ChipWatcher.



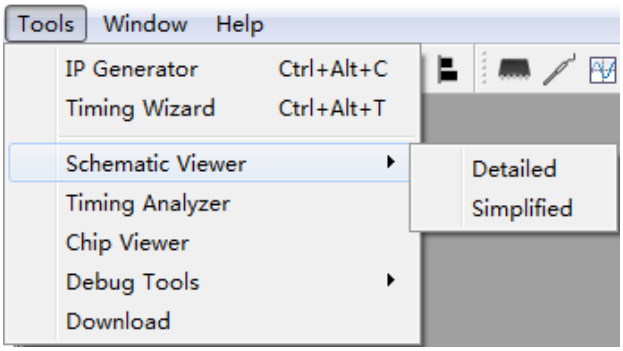
Schematic Viewer generates corresponding logic diagrams for each intermediate process of comprehensive optimization, improves interactivity in the design process, helps designers understand circuits, and shortens the development cycle of circuit design. Chip Viewer provides detailed information after physical implementation, including resource usage, detailed layout, global routing, and critical timing paths.

Without changing the design, ChipWatcher is an in-circuit debugging tool that allows you to see how internal signals change under specified conditions without the need for external devices. ChipProbe uses an external tool such as an oscilloscope to assign signals that need to be monitored internally to an idle IO port. Bram Editor reads the data from the RAM in the chip, and can modify the data, modify it and write it into the chip to see the effect.

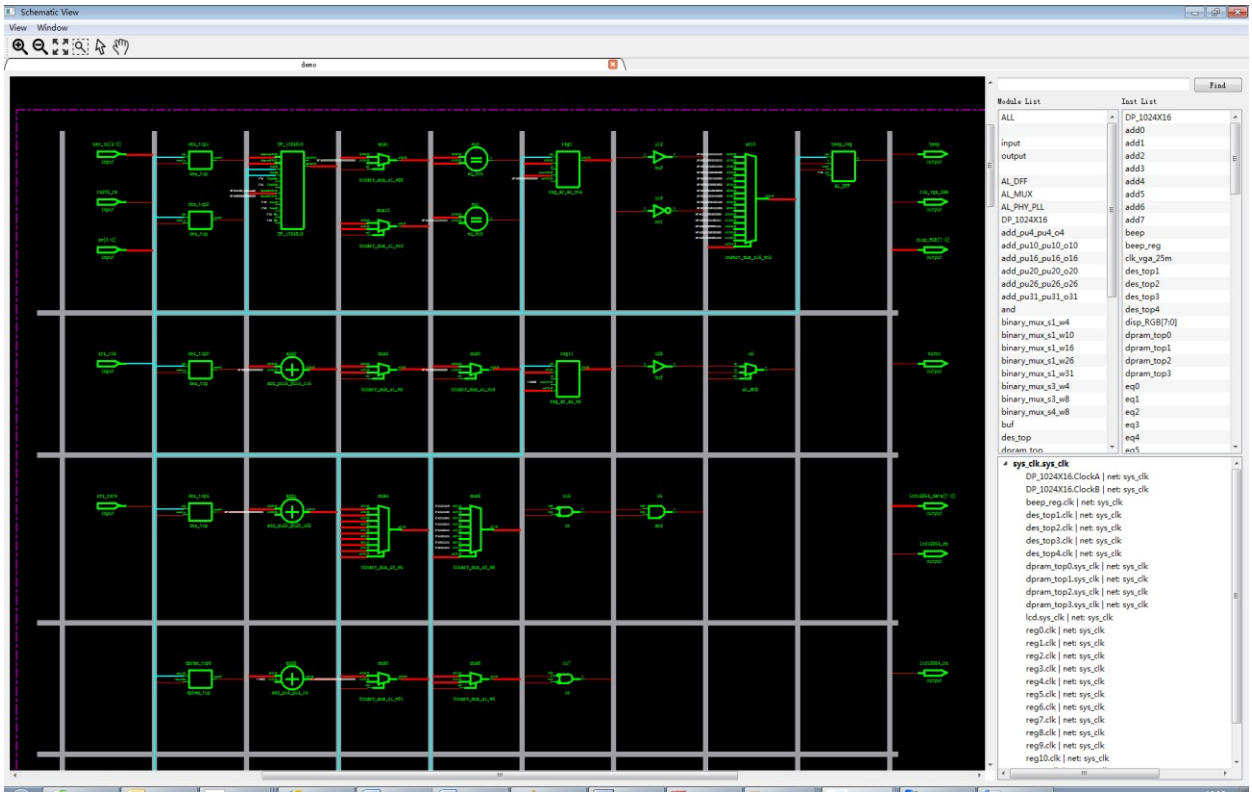
9.1 Schematic Viewer

After completing each process in the TD software, you can view the corresponding logic diagram in the Schematic Viewer. After the "Read Design" and "Optimize RTL" are completed, the viewed logic circuit diagram is the effect of syntax analysis and logic optimization. The circuit structure and the engineering device are independent at this time, and the adder is seen in the circuit diagram. , multipliers, comparators, AND gates, or gates and other components. After the "Optimize Gate", "Optimize Placement", and "Optimize Routing" are completed, the logic circuit viewed is based on the optimized structure of the selected FPGA device structure. In the circuit diagram, the lookup table, register, and BRAM are seen. , PLL and other components.

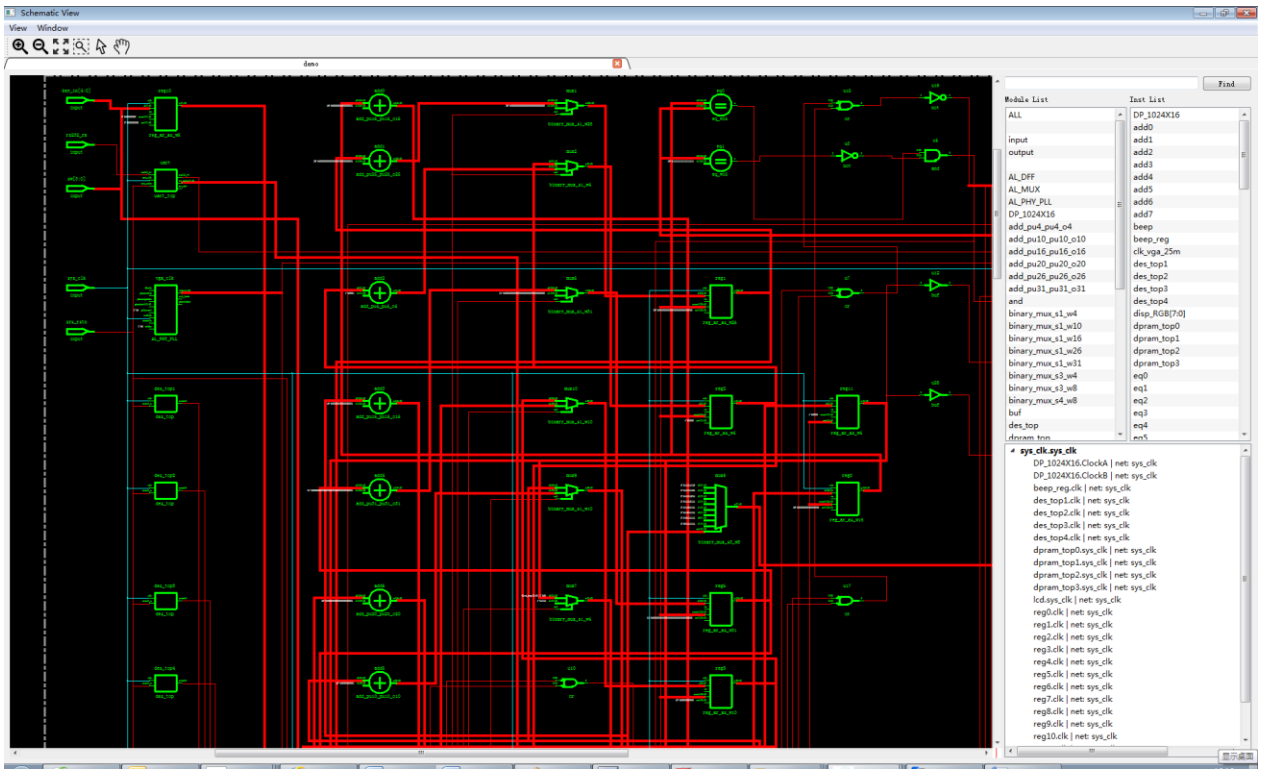
展开 **Tools** ⇒ **Schematic Viewer**，可看到有两种模式可以选择： **Detailed** 和 **Simplified**。



In Simplified mode, the displayed circuit diagram will be more streamlined, there is no complicated circuit connection, and all the lines are transmitted via the bus. This mode is suitable for projects with more resources, as shown in the following figure.



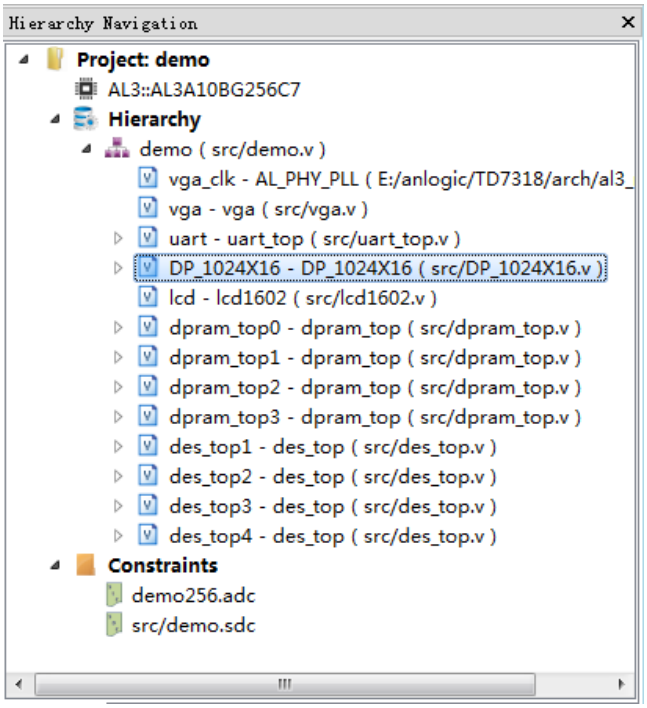
In the Detailed mode, you can see the detailed circuit information after TD comprehensive optimization. This mode is suitable for resources with less circuit, which can better understand the circuit design, as shown in the figure below.



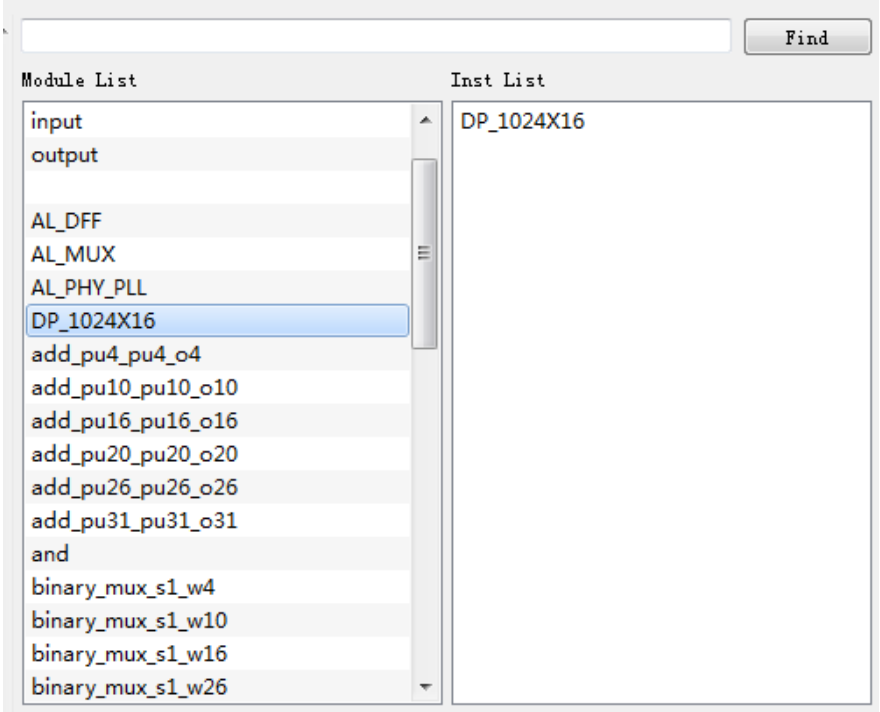
How to operate in the graphical interface:

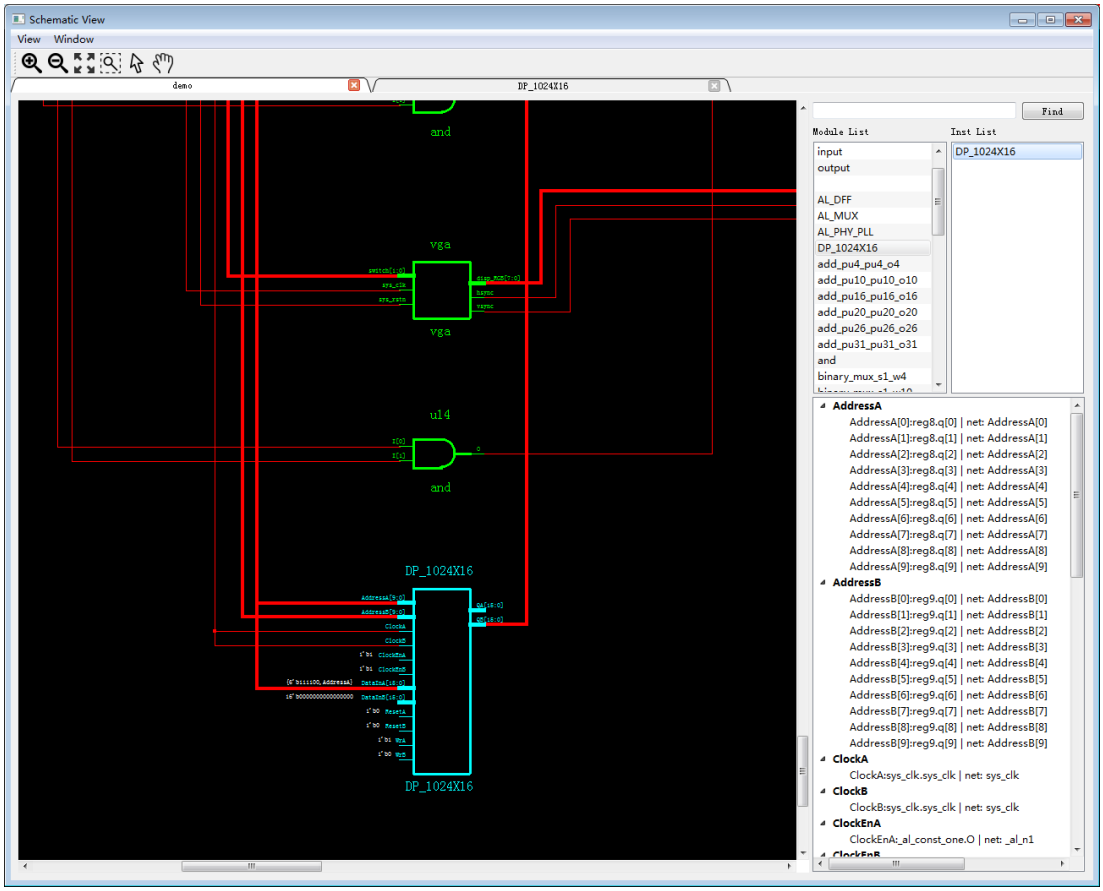
1. Highlight: mouse click on the connection, circuit components or by double-clicking Inst, Net;
2. Zoom: Click the button in the toolbar to zoom in, click the button to zoom out, or zoom by Ctrl+Scroll;
3. Overall view: Click the button in the toolbar to display the entire circuit diagram in the main window;
4. Drag: Click the button in the toolbar to drag the main window up and down and left and right;
5. Partial zoom in/out: Click the button in the toolbar, press the left mouse button in the main window, drag to the lower right to zoom in on the part, and drag to the left to zoom out.
6. Select mode: Click the button in the toolbar to switch back to the normal mouse function.

After "Read Design", the user-designed hierarchy is retained, so in Schematic Viewer you can double-click to view the submodules in the design, as in the submodule DP_1024x16 in the figure below.

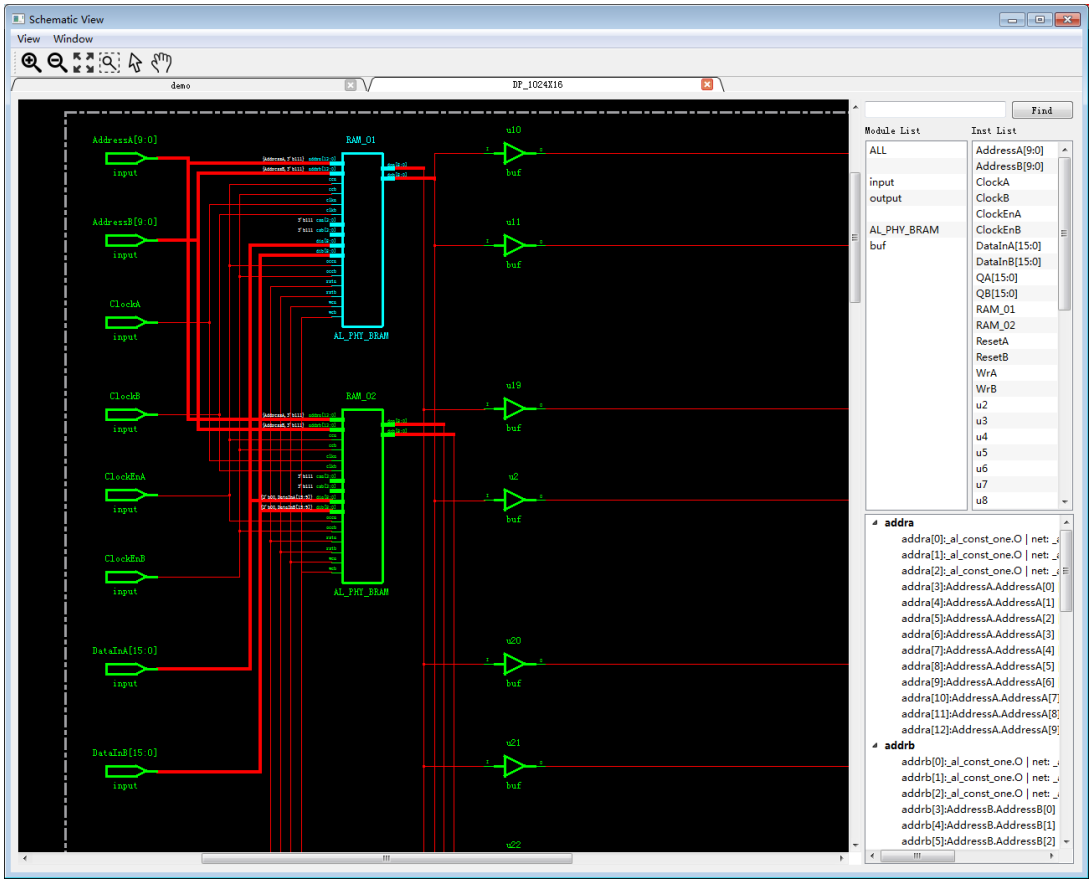


Find DP_1024x16 in the Module List and double-click it. The Instance corresponding to the module will be listed in the Inst List. Double-click the Instance to jump to its location in the main window.

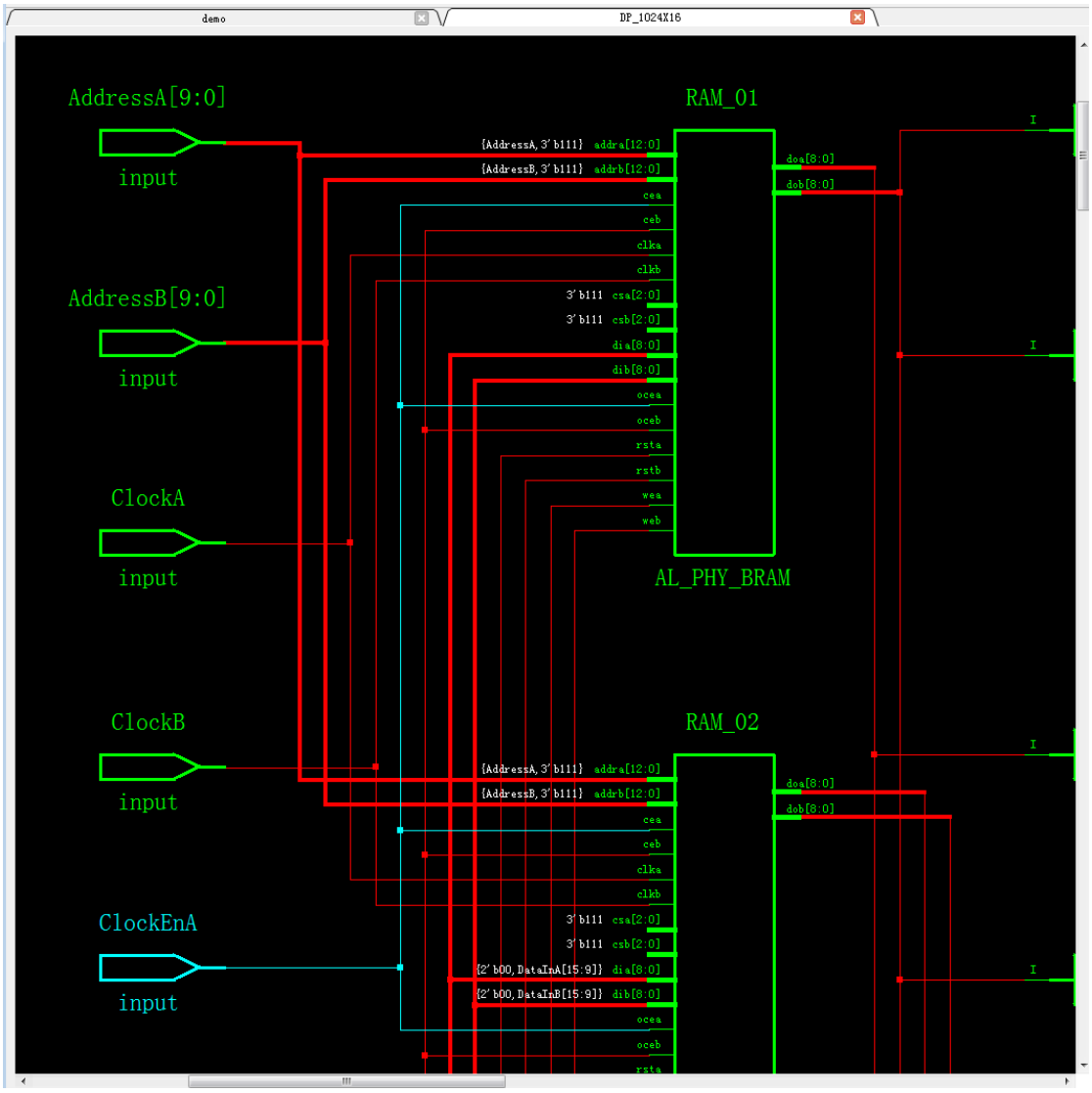




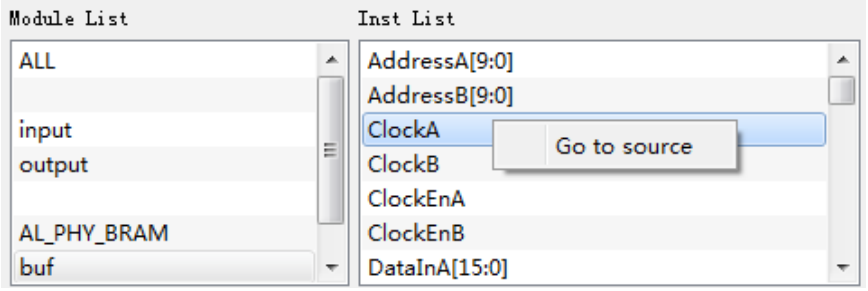
Double-click the DP_1024x16 graphic in the main window to open the corresponding circuit diagram of the module.



Selecting a module in the main view will display all its related nets in the lower right window. Double-clicking a net will highlight all the connection relationships of the net. In the main window, the bold connection is bus, the non-bold is net, the port without connection is constant, and the value of the port can be viewed.



In the Inst List, you can also right-click on an Instance and select “Go to Souce” to jump to the location in the source code of the Instance.

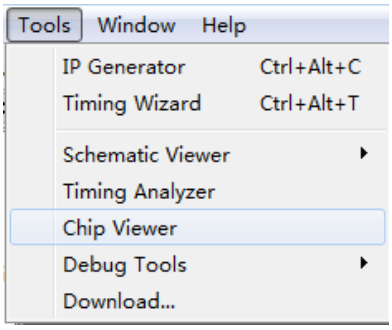


9.2 Chip Viewer

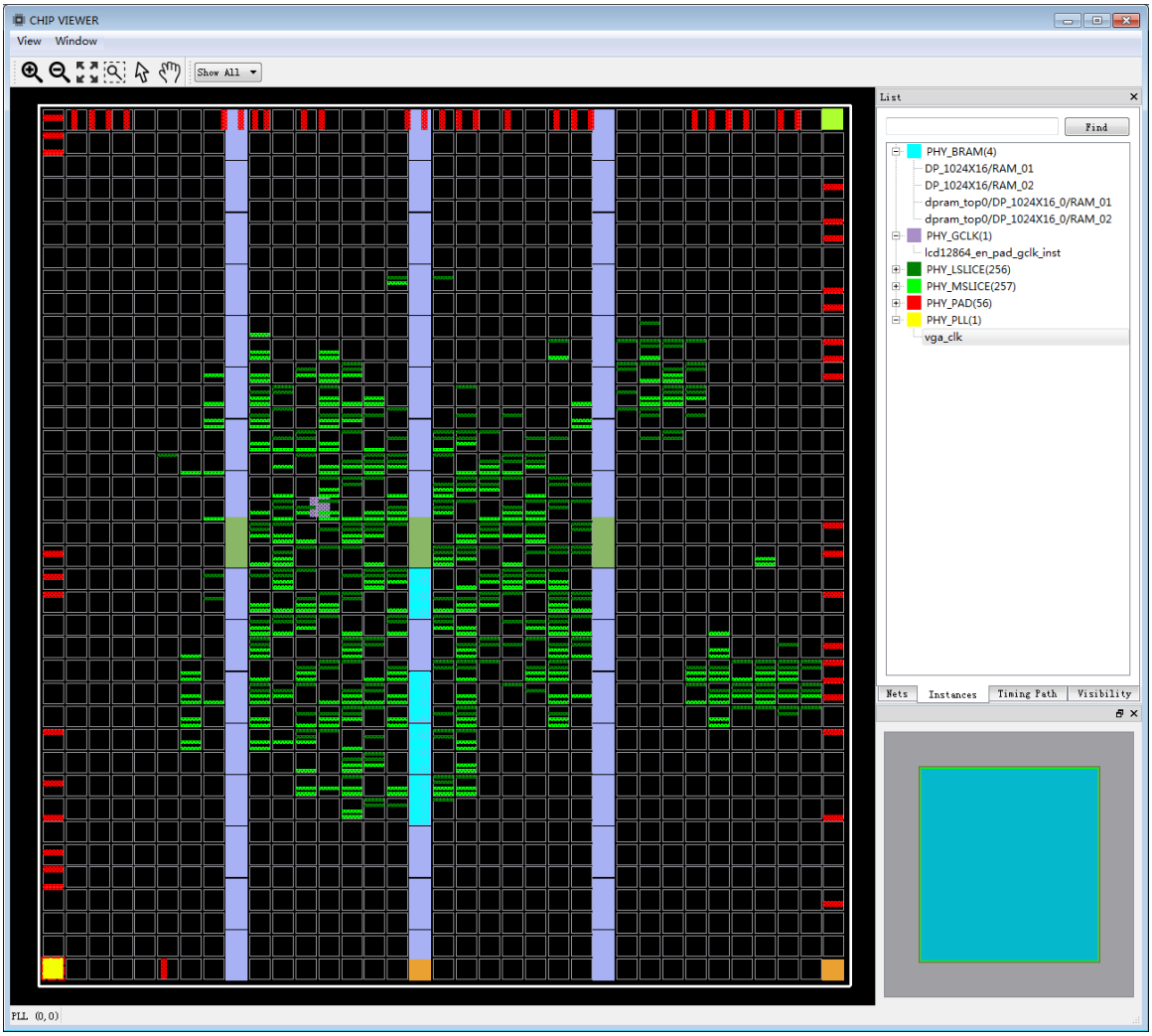
After the placement and routing is completed, the user can view the detailed information after the physical implementation is completed through ChipViewer: including Cell Utilization, Global Routing, Detail Routing, All Nets, Timing Path.

、

单击 **Tools ⇒ Chip Viewer** 打开，

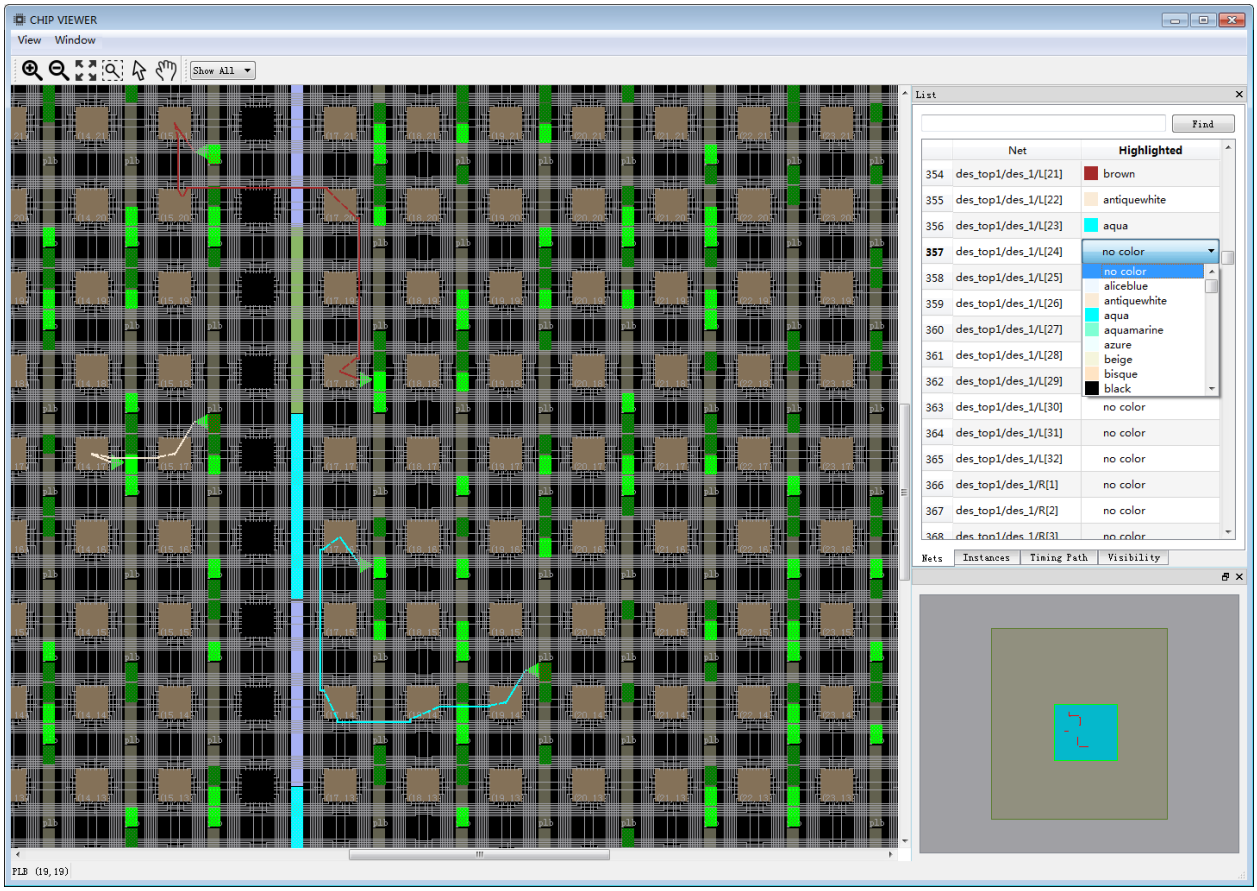


The top view of the Chip Viewer is shown below:

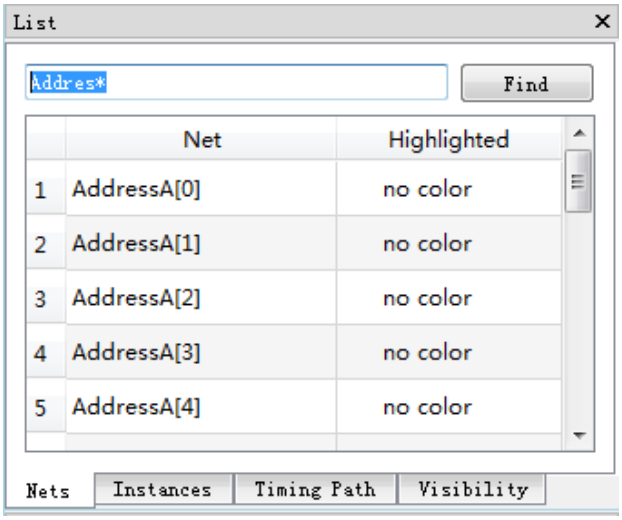


The graphical interface operates in the same way as the Schematic Viewer.

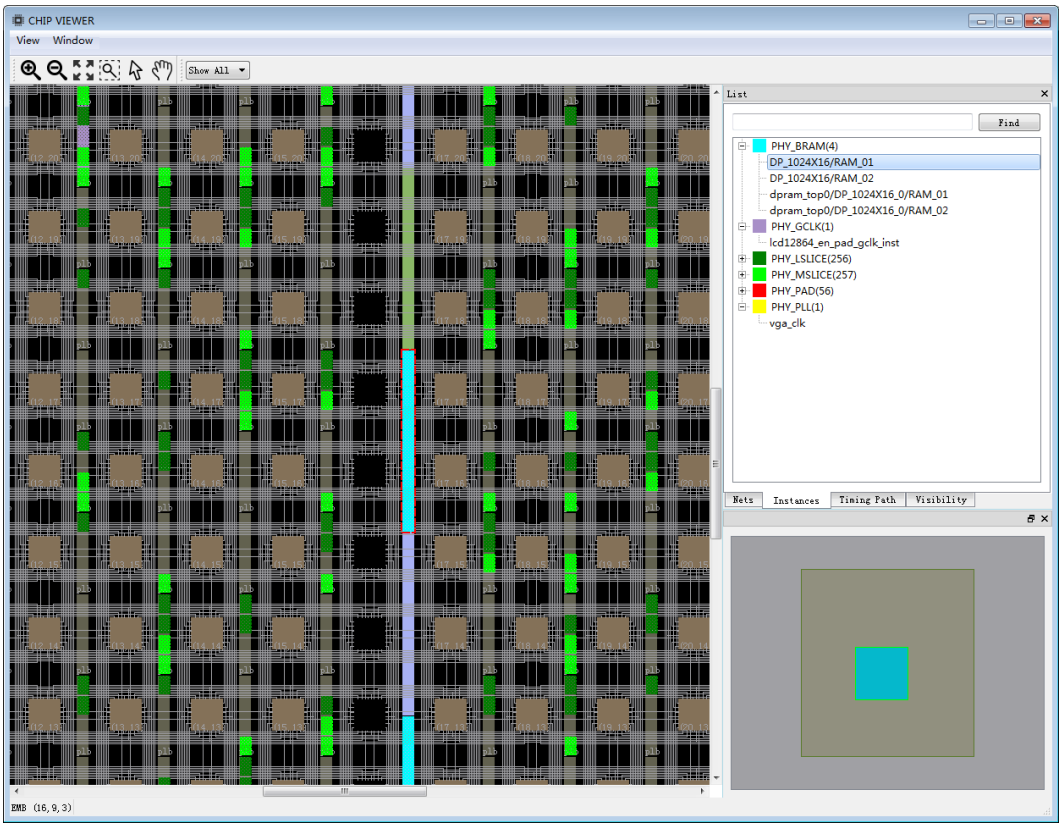
You can view all the nets in the design in the Nets column. Double-click the net to display the specific routing in the left window. You can select different colors in the Highlighted column to keep the net highlighted. For a net, the triangle arrow is sourced from plb outward, and the triangle arrow is sink to plb.

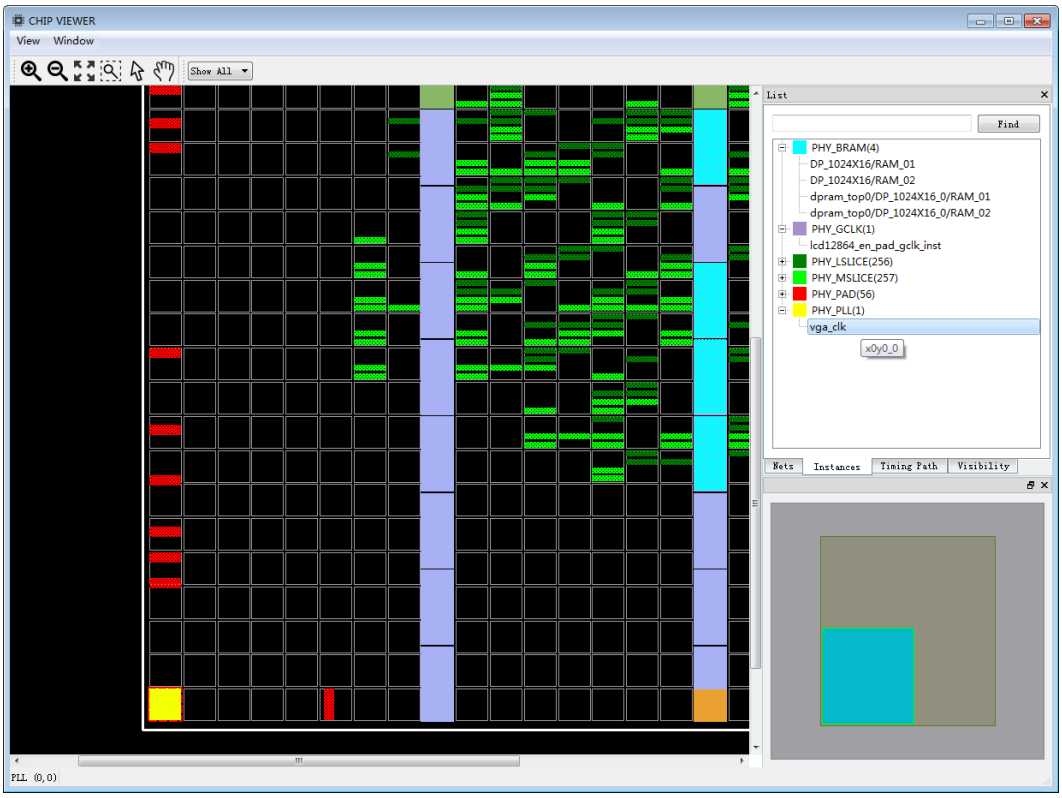


Search for net in the Find box.

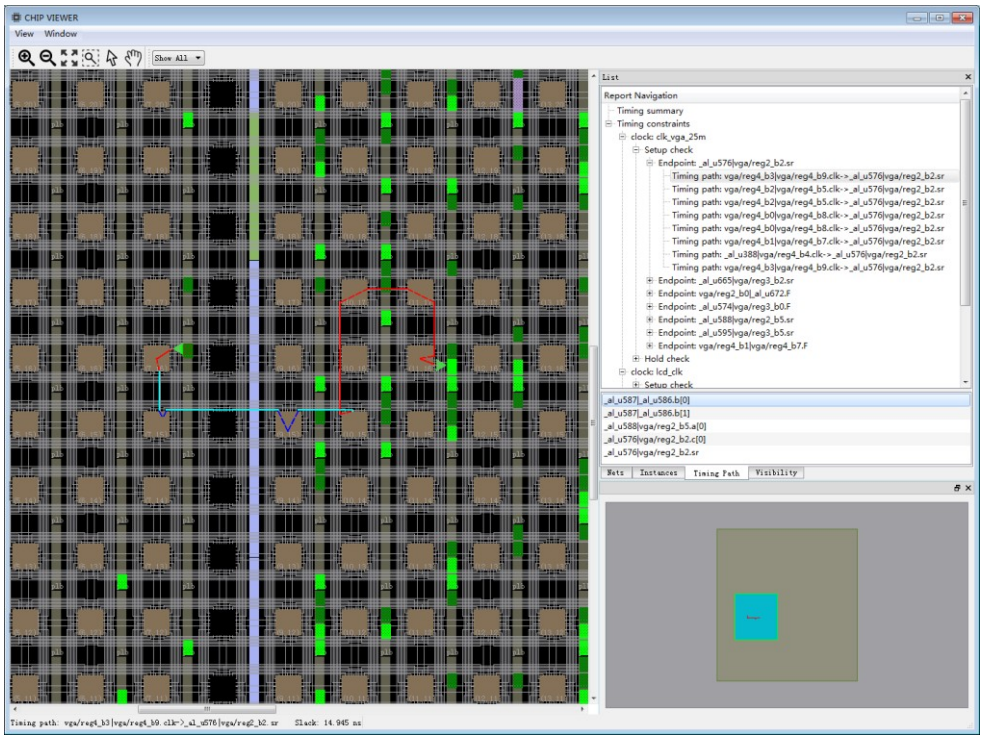


In the Instance column, you can view all the physical modules in design. Double-click the instance to display its layout position in the left window. Click on the module and its physical coordinates will be displayed in the lower left corner.

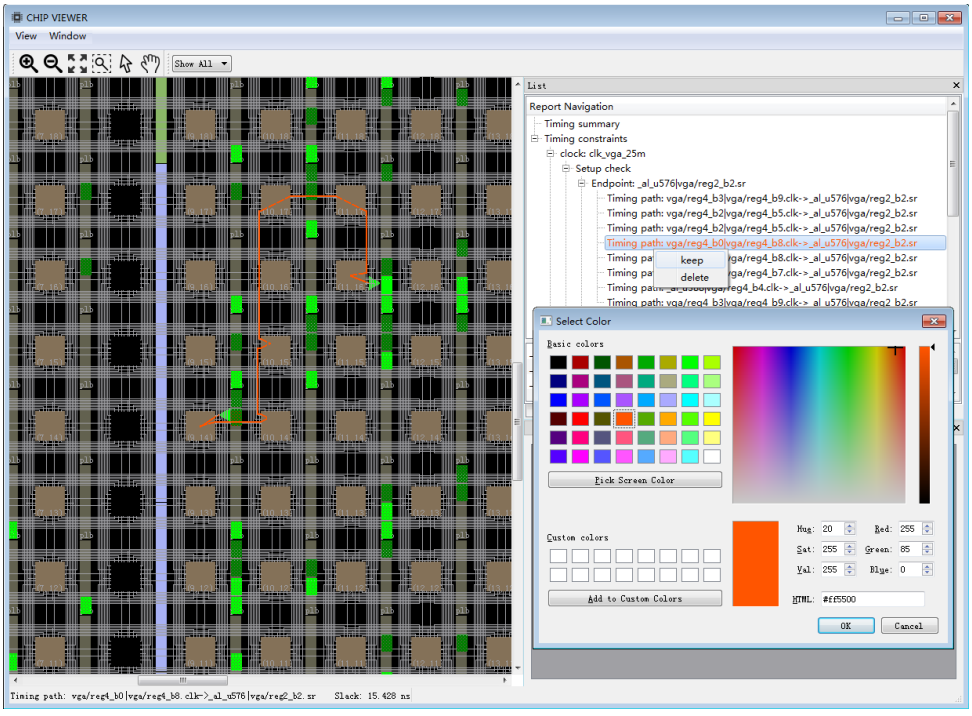




In the Timing Path column, you can view all the timing paths listed in the timing report. Double-click the timing path to display the detailed routing in the left window. Double-click on a net in path to highlight it.

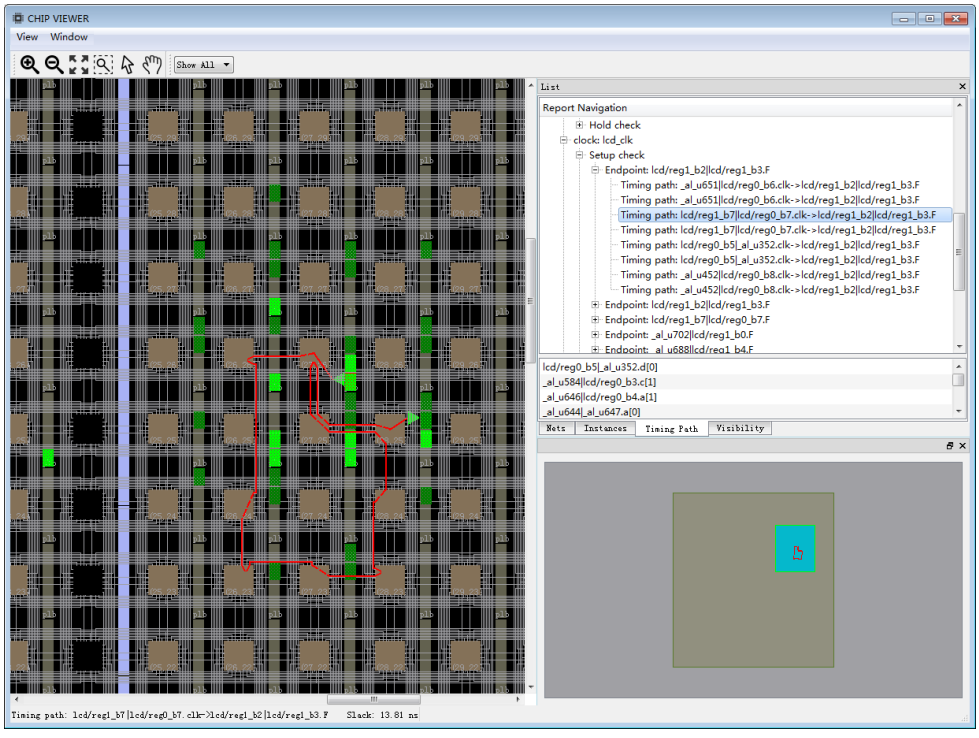


Similarly, select a timing path, right-click, select keep, and select a color to highlight the path and keep the path continuously displayed. That is, multiple timing paths or nets can be displayed simultaneously in the Chip Viewer. .

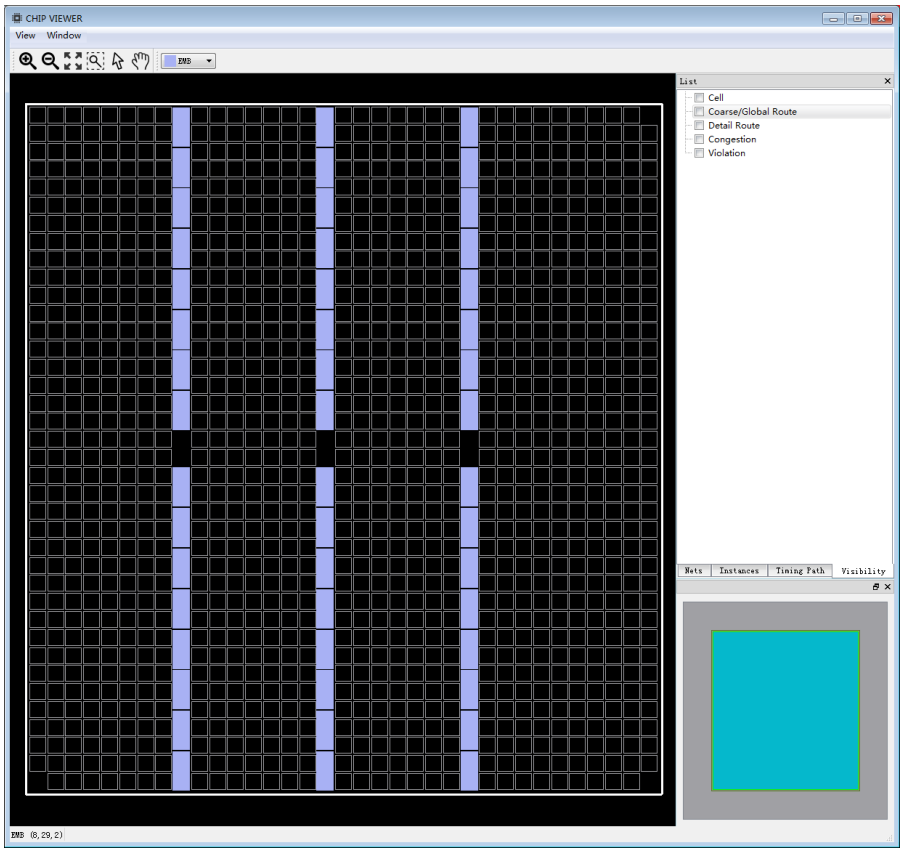


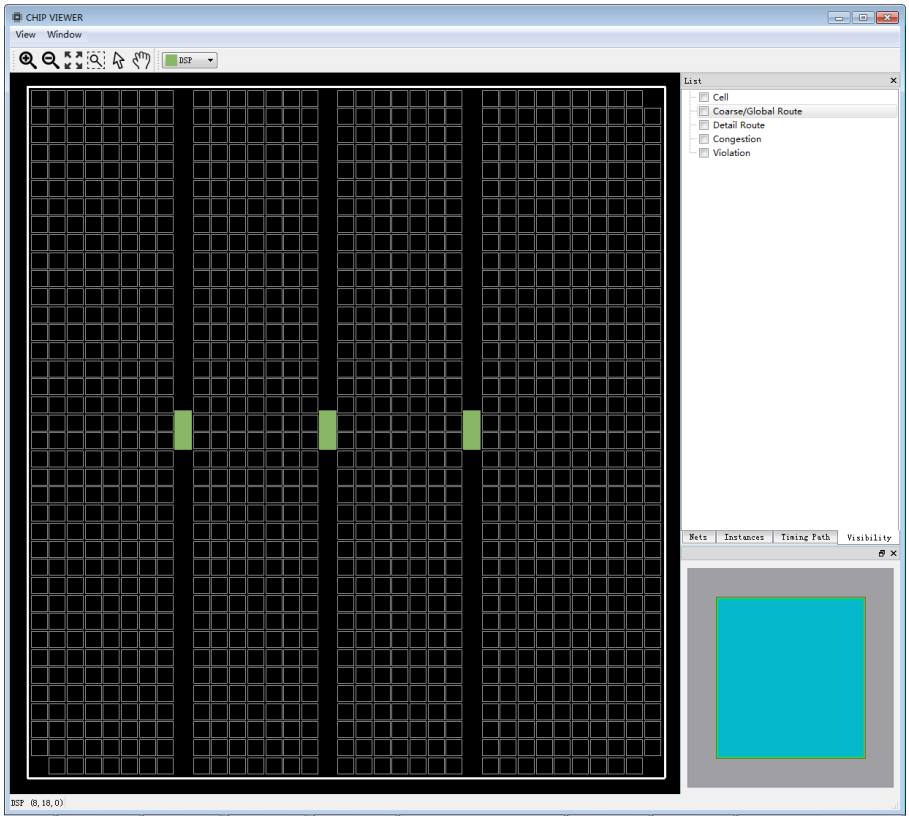
In the Visibility field, you can choose to display basic units, global routing or detailed routing.

The bottom right corner shows the bird's eye view in the Chip Viewer.

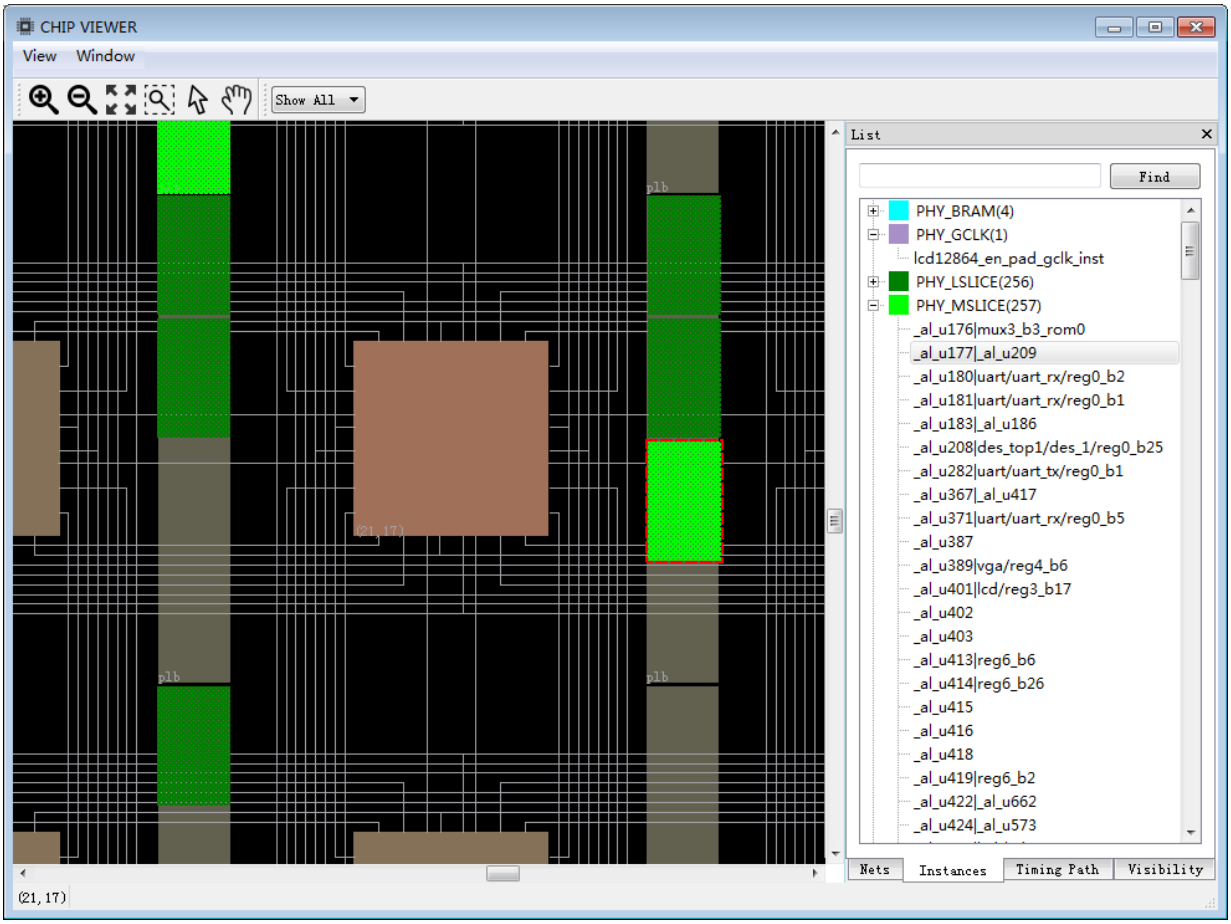


In the menu bar of Chip Viewer, you can choose to display the location of all physical resources in the device, such as: PLL, EMB, EMB32K, DSP. Click on a module to display its physical coordinates in the lower left corner.





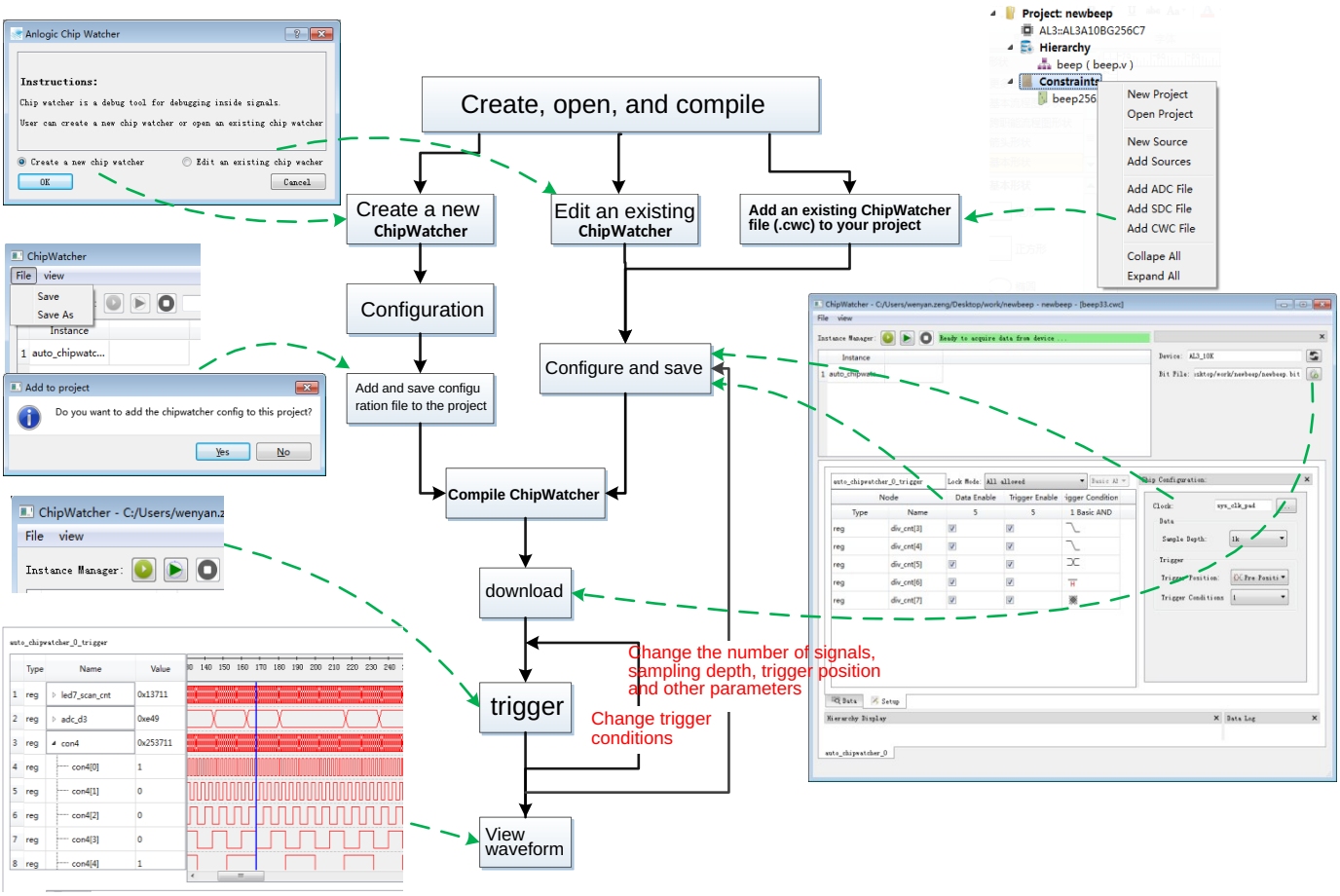
For the coordinates of mslice and lslice, you need to check the coordinates of plb first. A plb contains 2 mslice and 2 lslice. If the coordinates of a plb are (21,17), then from bottom to top, the coordinates of mslice are (21,17,0), (21,17,1); The coordinates of lslice are (21, 17, 2), (21, 17, 3).



9.3 ChipWatcher

With ChipWatcher, users can monitor the internal signal changes on-line without the need for external devices. In ChipWatcher, users can add multiple signals at the same time.

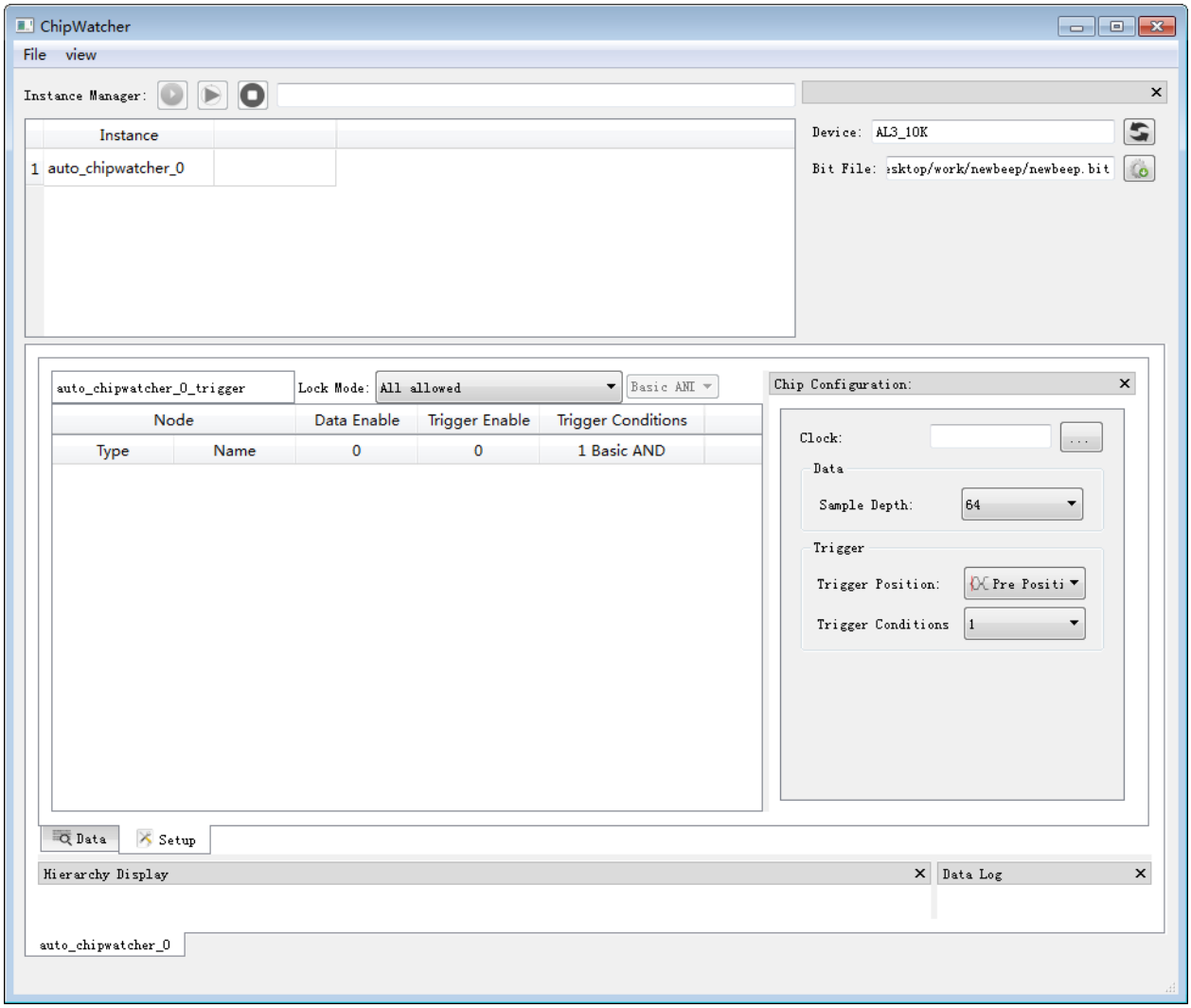
After setting the sampling clock, sampling depth, trigger condition and trigger position of the signal, after recompiling, downloading and triggering, the signal changes under the specified conditions can be viewed. The workflow of ChipWatcher is shown below.



1. After running the HDL2Bit process, there are three ways to start ChipWatcher:

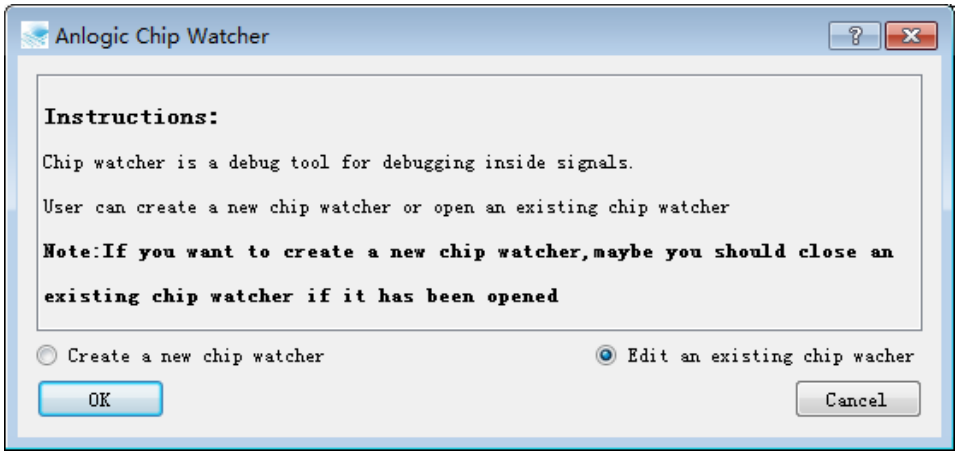
   ①. Create a new ChipWatcher

   Expand Tools → Debug Tools, select ChipWatcher, select “Create a new chip watcher” according to the prompt, click “OK”, the following main interface of ChipWatcher appears:



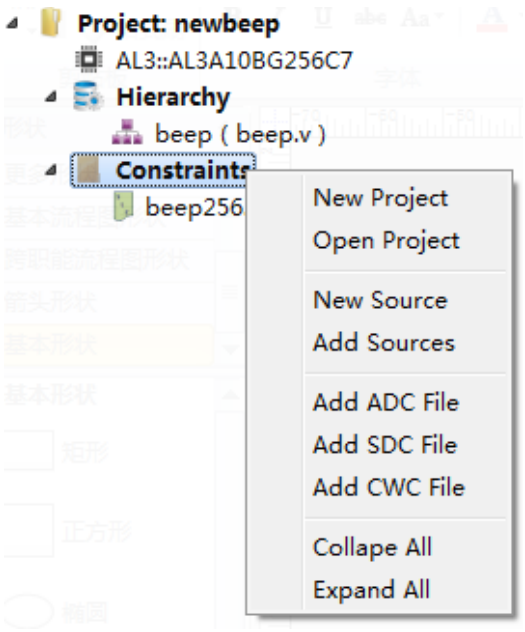
②. Edit an existing ChipWatcher

Expand Tools → Debug Tools, select ChipWatcher, select “Edit an existing chip watcher” according to the prompt, click “OK” to enter the main interface of ChipWatcher.

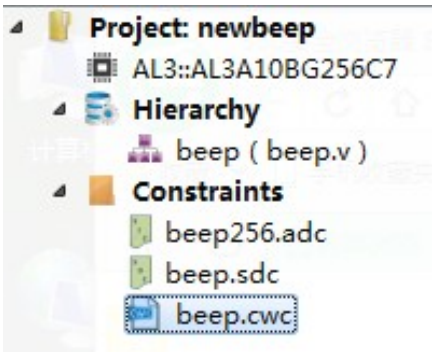


③. Add an existing ChipWatcher configuration file (.cwc) to your project and double-click it to open it.

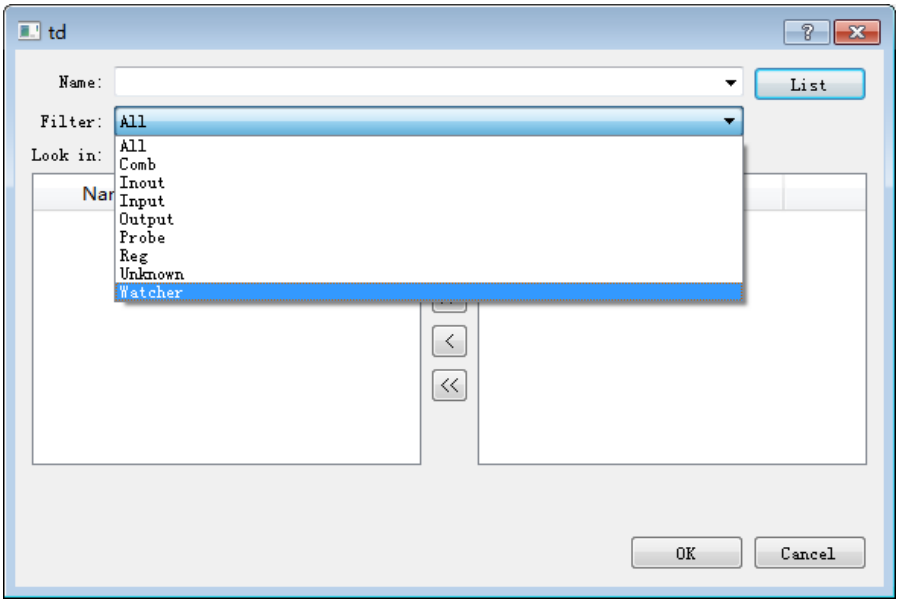
In the Project section, right-click Constraints and select Add CWC File to add the generated configuration file (.cwc) to the project.



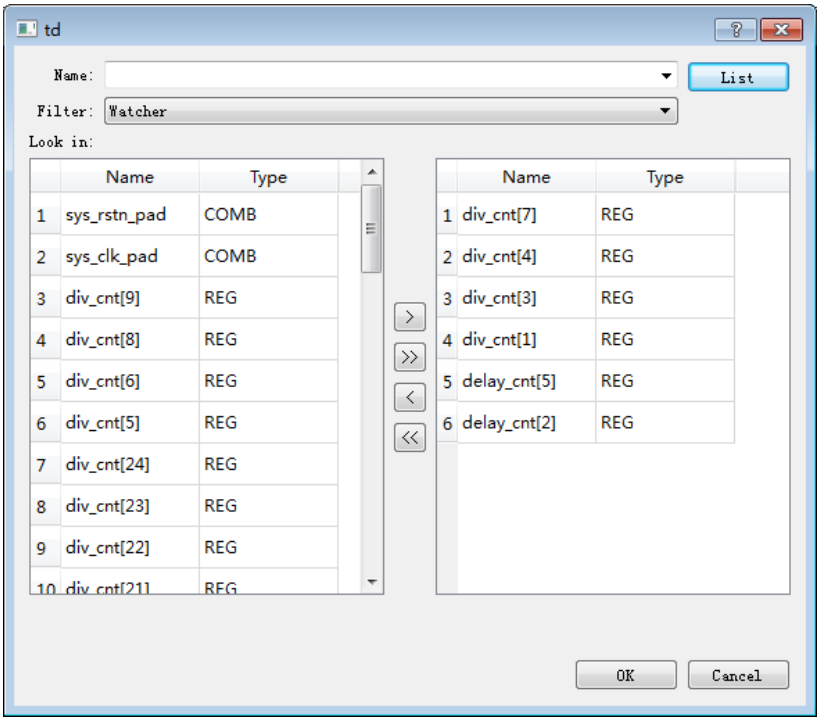
Double-click the file to open the main interface of ChipWatcher.

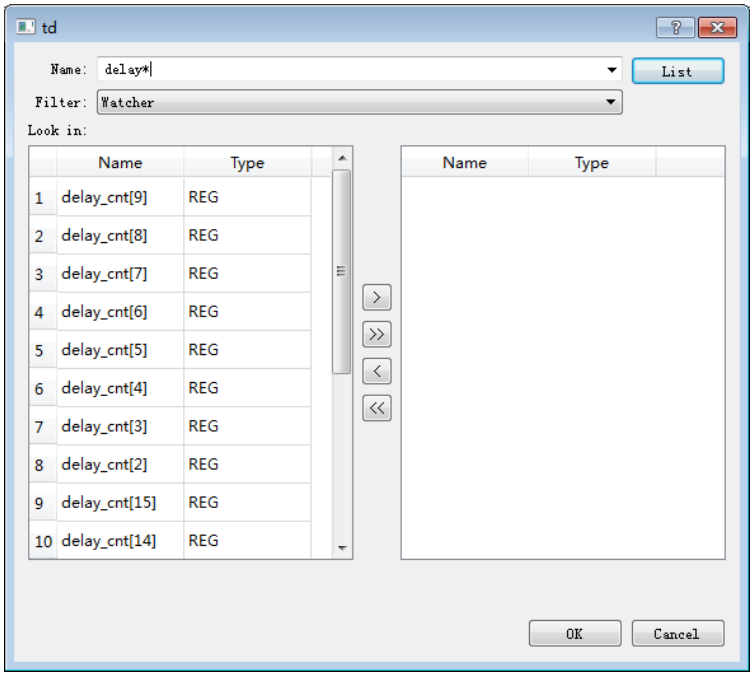


2. Right-click in the blank space of the setup interface and select “Add Node...”. The following Node Filter interface appears. Expand the Filter drop-down menu. The default type is Watcher. If the user selects a signal whose filter type is other than Watcher, it will not be guaranteed to be correctly sampled;

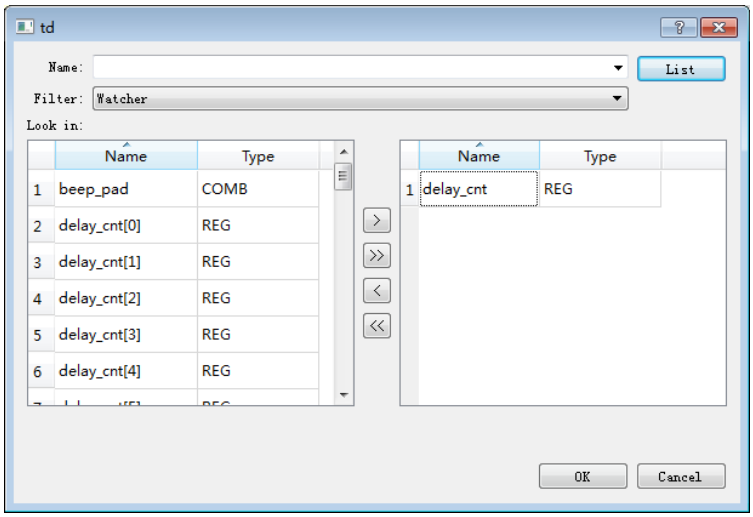


3. You can also search for signals in the Name column by selecting or deleting the signal with the left and right arrows (support wildcards*);

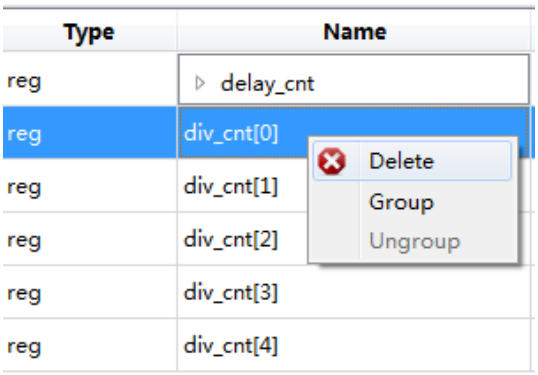




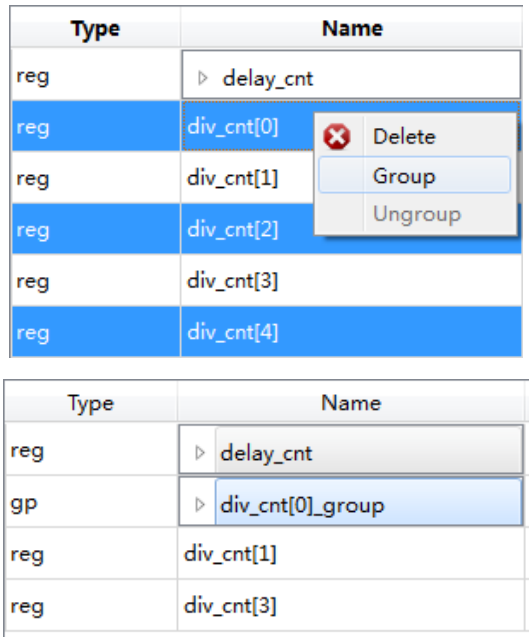
Add net or bus signals as needed, click the OK button to complete the signal addition.



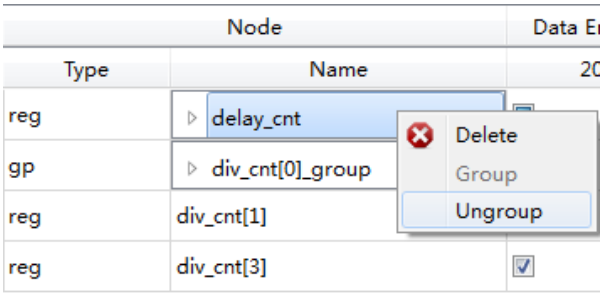
4. For the added net or bus, right click on the signal name to perform Delete/group/ungroup operation; single signal or whole bus can be deleted directly;

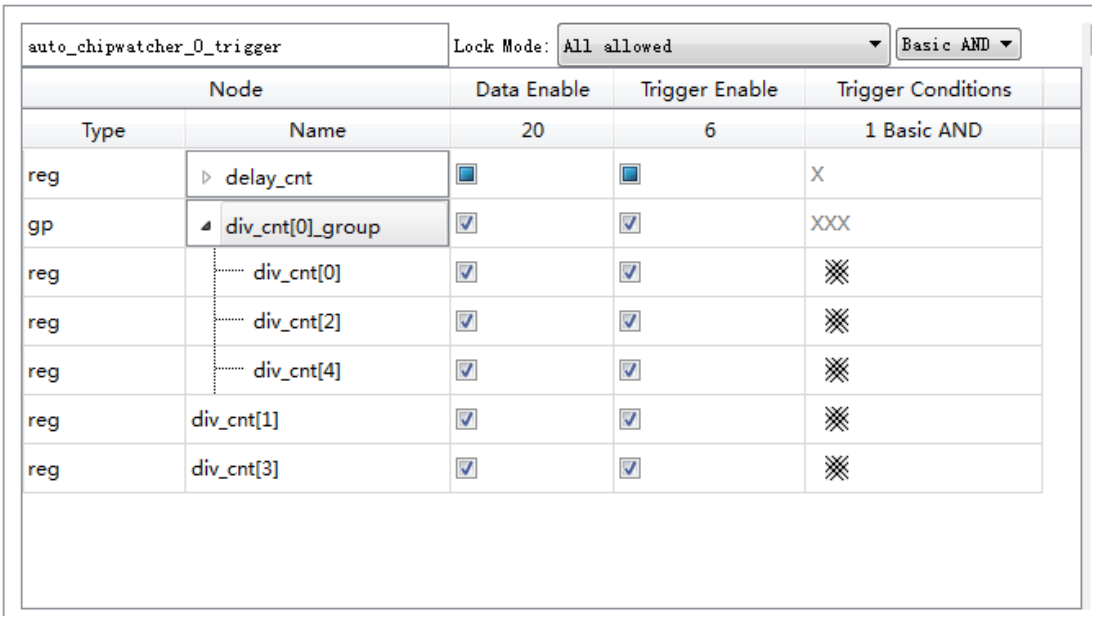


At the same time, in order to facilitate the observation of the signal, Group operation can be performed on multiple network signals;



Upgroup operations can be performed for bus or group. At the same time, the mouse can be dragged by selecting net, bus or group to adjust the current Node order.



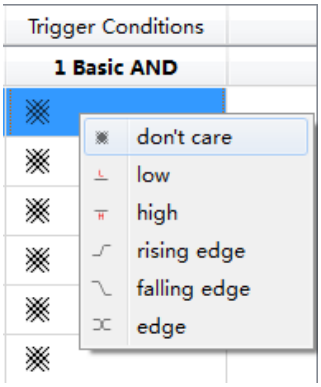


5. Data Enable refers to selecting the signal that needs to acquire and display the waveform. Tick the check box to enable the signal; Trigger Enable refers to the state of the signal as the trigger condition; Trigger Conditions refers to when the condition is met. The signal can be triggered.

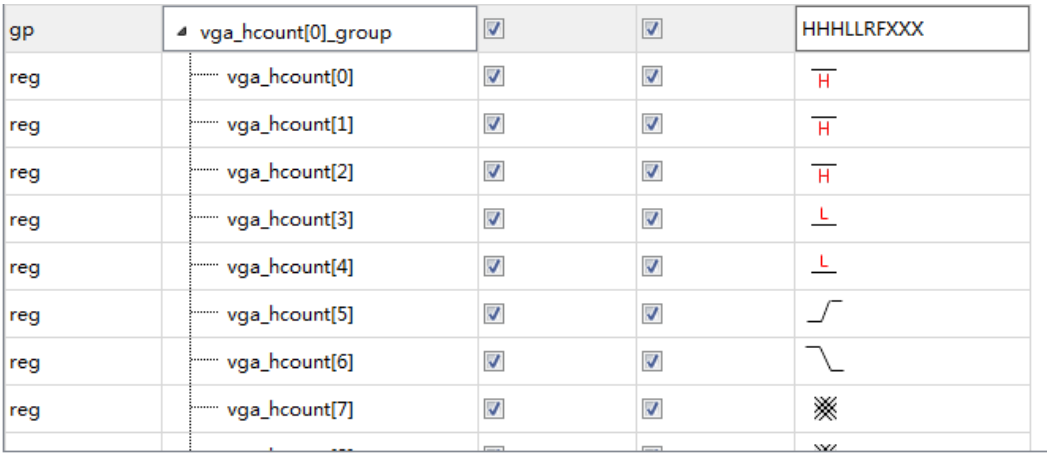
   Basic AND means that the signal can be triggered when all the following trigger conditions are met.

   Basic OR means that the signal can be triggered as long as any of the following trigger conditions are met.

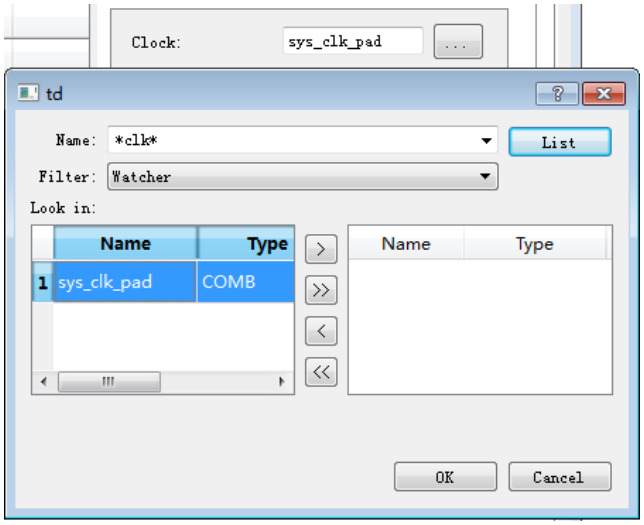
   Right-click on the trigger condition column to change the trigger condition.
   The trigger conditions are as follows: arbitrary position, low level, high level, rising edge, falling edge and double edge (rising edge or falling edge);



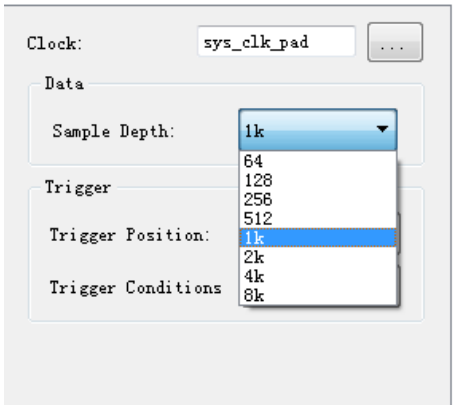
At the same time, you can double-click the trigger conditions of bus or group to edit. The characters you can enter are: x/X, l/L, h/H, r/R, f/F, e/E. The unfilled net is filled with X. After inputting, the trigger condition of the corresponding net is changed.

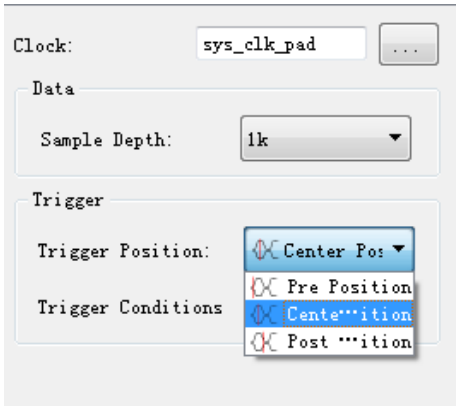


6. Click the ... button after the Clock column to add the sampling clock. This signal will be used as the working clock of the entire ChipWatcher module. It is necessary to ensure that a valid and appropriate clock signal is selected. Otherwise, it may cause incorrect triggering or sampling accuracy deviation.

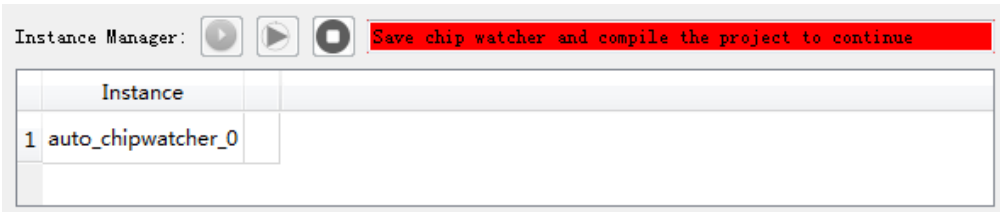


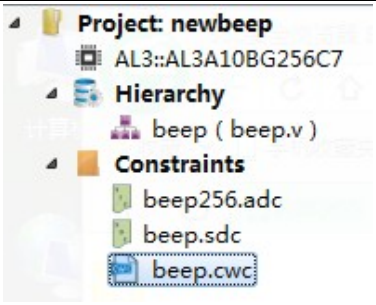
7. Select the sampling depth and set the location of the trigger. Pre Position indicates that the trigger position will be in the first third of the entire sampled data; Center Position indicates that the trigger position will be at one-half of the entire sampled data; Post Position indicates that the trigger position will be in the last third of the entire sampled data. one place;



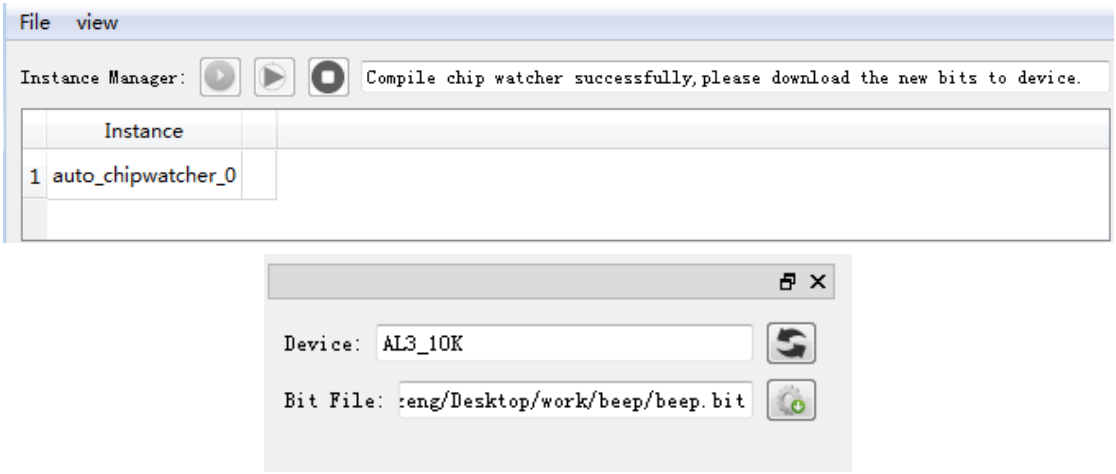


8. After setting all the parameters, according to the prompt given by the interface, click File → Save in the upper left corner, or use the shortcut ctrl+s to save the ChipWatcher configuration file (.cwc), add the configuration file to the project, and recompile the project. ;

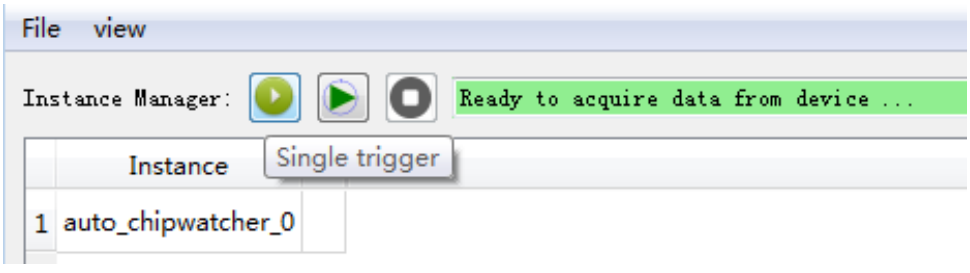


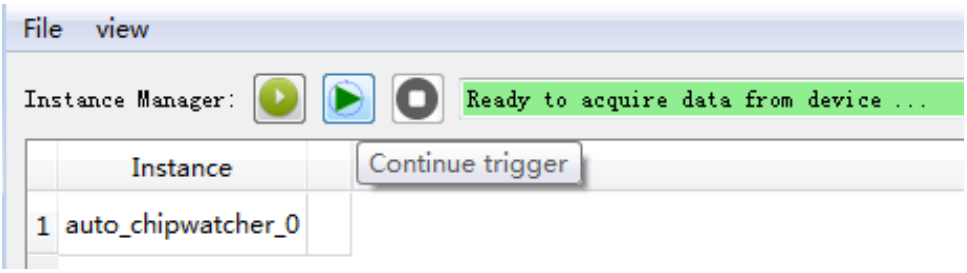


9. After the compilation is completed, download the newly generated bit file according to the prompts;

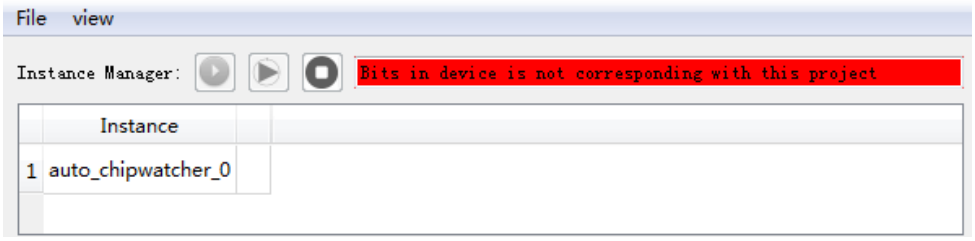


10. After downloading, the trigger button in the upper left corner of ChipWatcher will light up and give the following prompt. At this point, click the trigger button, ChipWatcher will start monitoring the specified signal, and once the preset trigger condition is met, the data in the chip will be returned. The single trigger is a single trigger, that is, the data under the current condition of the current time in the chip is acquired; the Continue trigger is a continuous trigger, and the data under the condition in the chip can be obtained in real time;

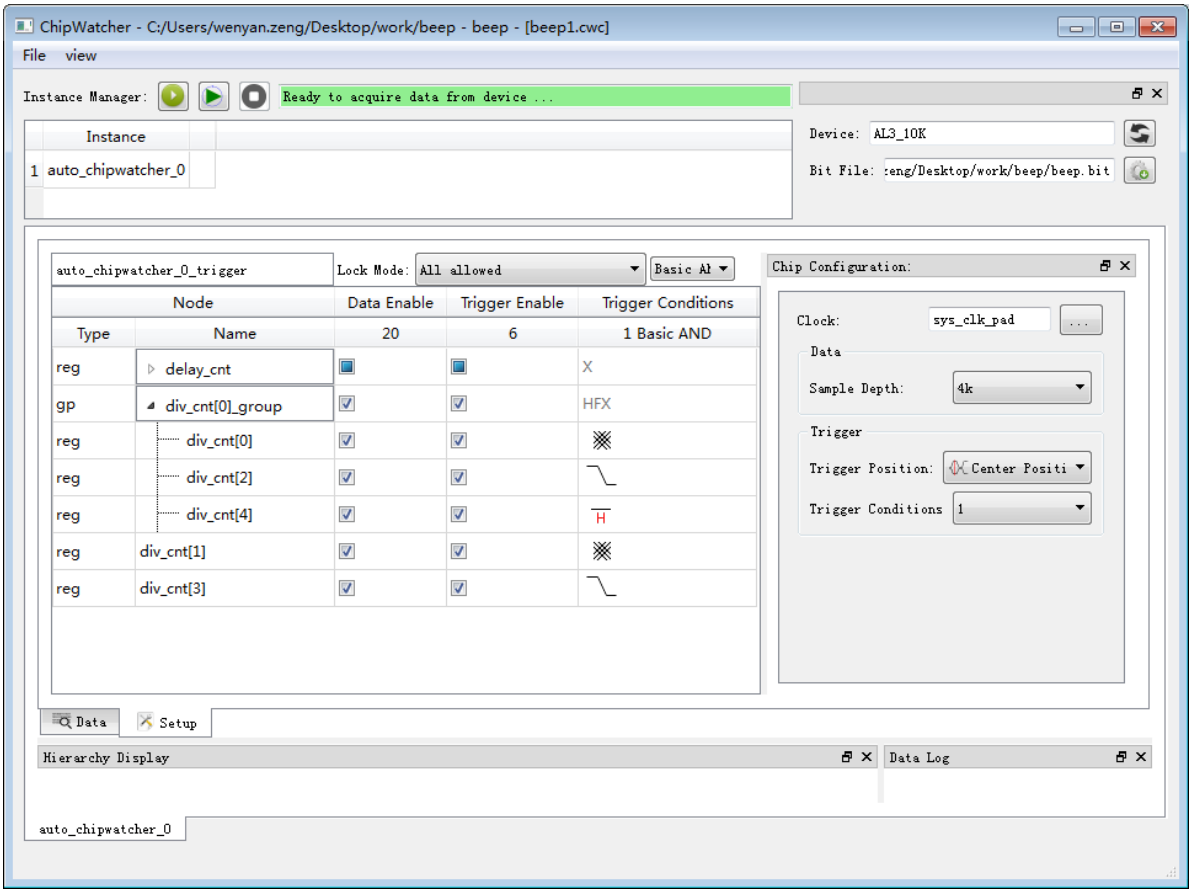


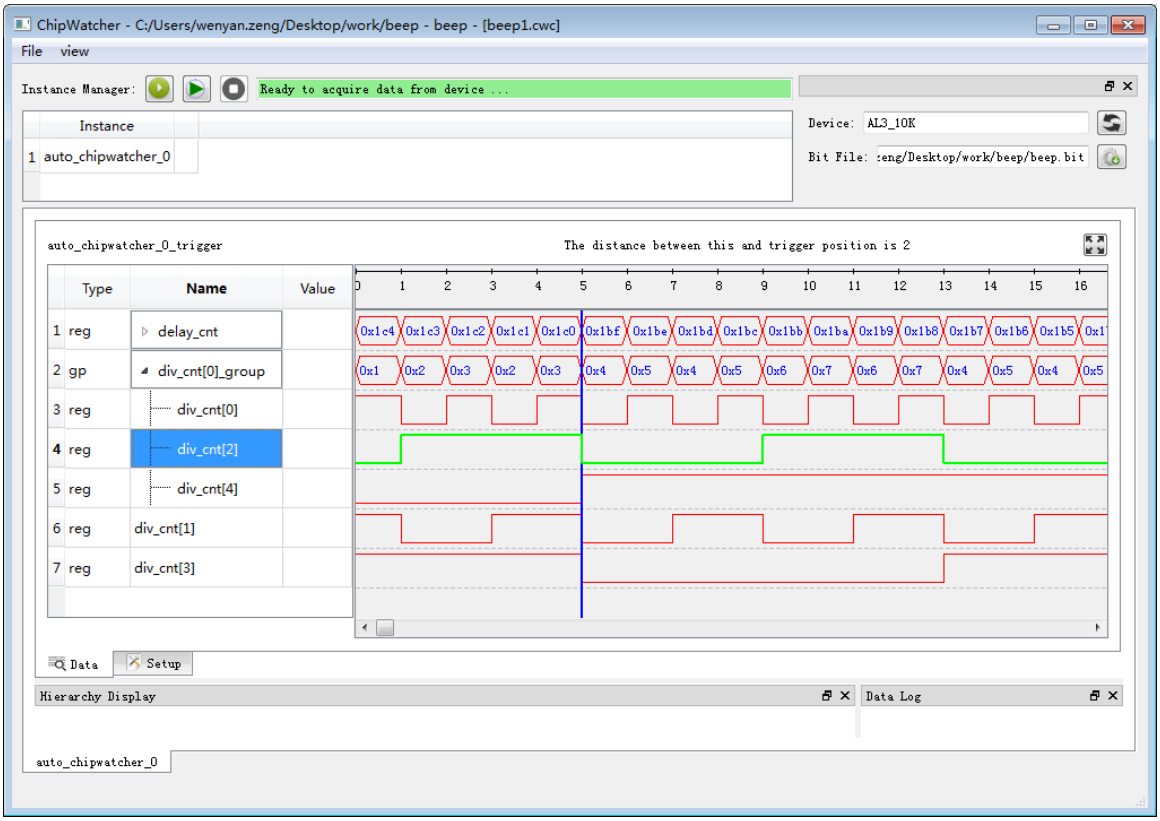


If the downloaded bit file does not match the current ChipWatcher object, the following prompt will be given.

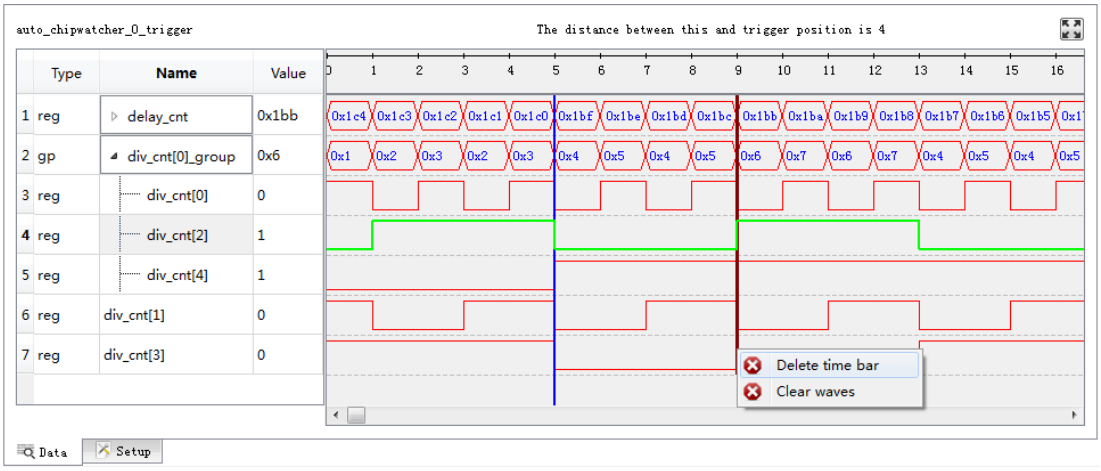


11. Once the signal is triggered, the ChipWatcher page will be switched from the Setup interface to the Data page and will display the waveform of the readback signal. The blue vertical line indicates the trigger position and cannot be deleted.

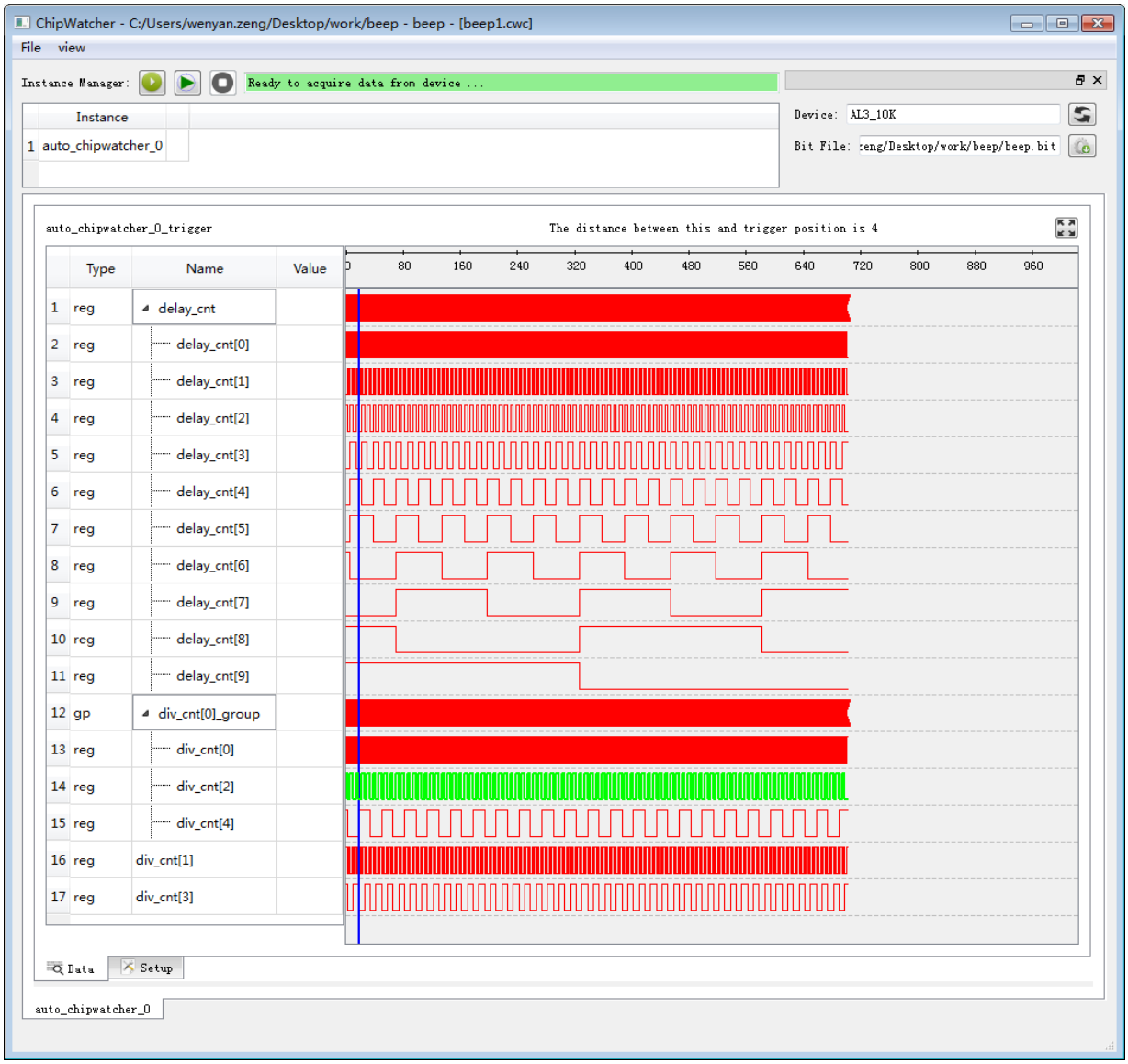




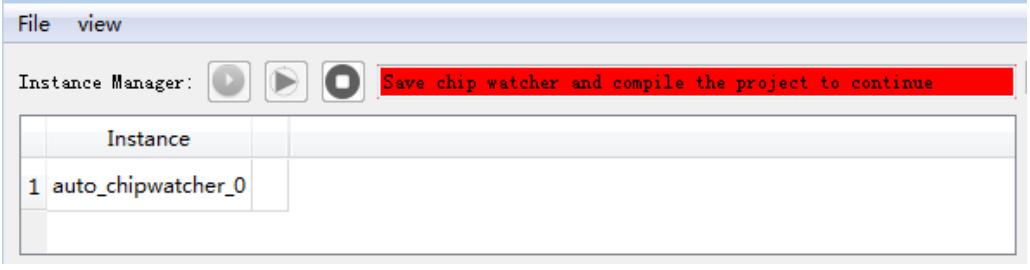
The user can double-click the waveform to add a time bar, as shown in the following figure, the time bar can be moved by mouse or right click to delete. When you move the time bar, the Value column on the left shows the value of each signal at that location and gives the distance from that position to the trigger position. You can also right click on the waveform and select “Clear waves” to clear the current waveform and re-trigger.



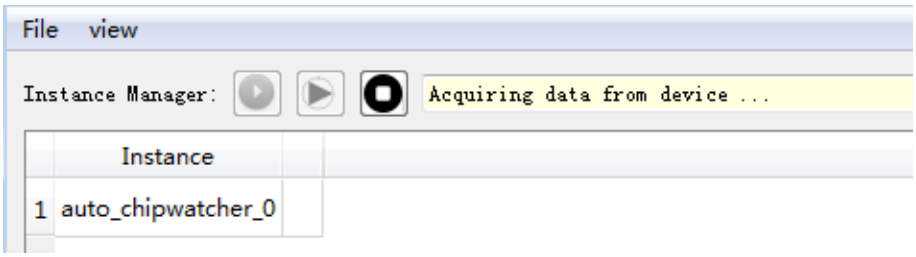
12. When the mouse is in the waveform area, the mouse wheel up indicates the enlarged waveform, and the scroll wheel indicates the reduced waveform. At the same time, you can also click the “Fit for view” in the upper right corner to display the entire waveform status.

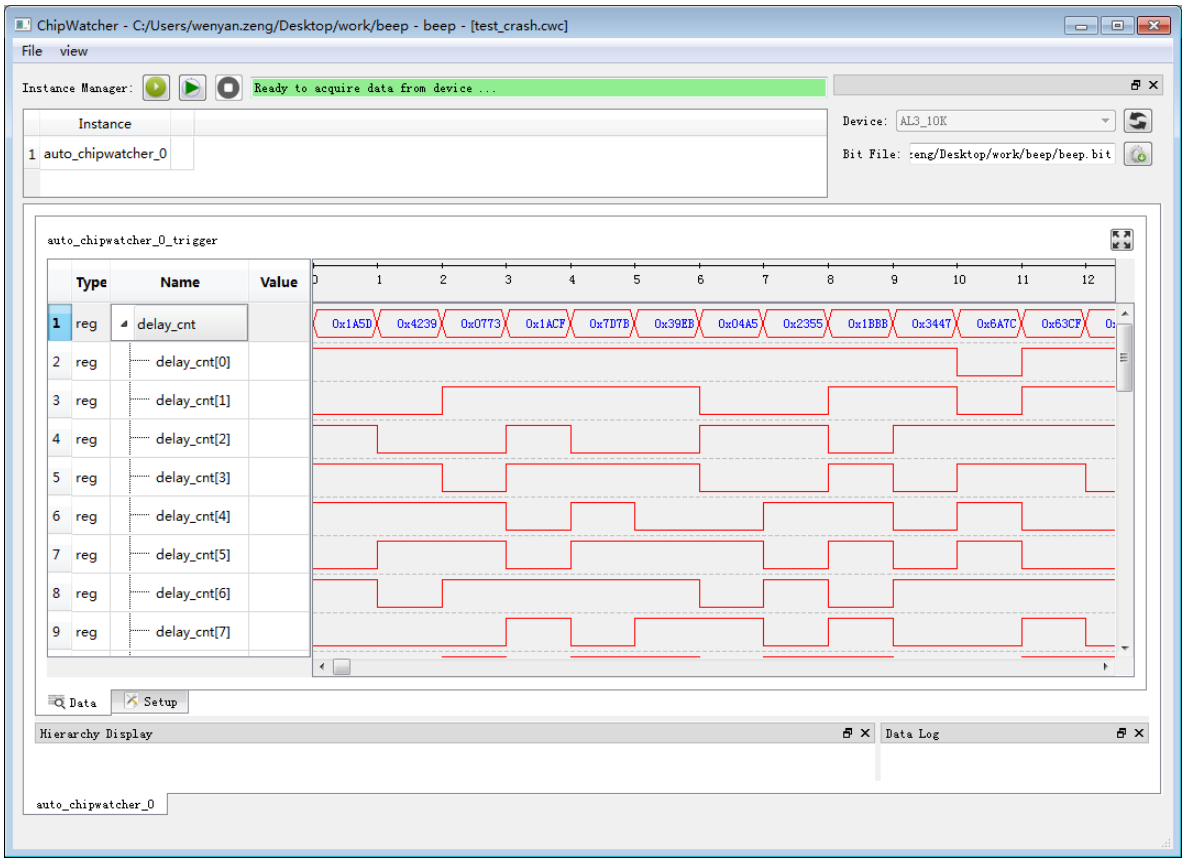


13. If the user only changes the trigger condition (such as changing the rising edge to the falling edge, changing the low level to high level, etc.), it can be triggered again without recompiling. When the user changes the number of signals, sampling depth, sampling position and other conditions, it needs to be saved, compiled and downloaded again to trigger again. At this time, Chipwatcher will also give a corresponding prompt:



14. If the trigger is clicked, it is always in the following state, indicating that the data in the chip cannot meet the trigger condition. Please change the trigger condition and re-trigger. At this time, click the stop button, and the data interface will export the waveform of the current attention signal.





With no engineering, you can still use ChipWatcher to open an existing cwc file for waveform viewing.

Need to pay attention to the following issues:

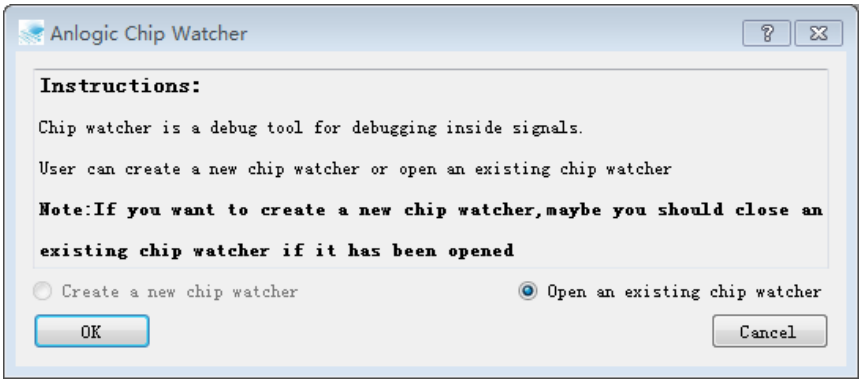
1. Need to ensure that the files required by ChipWatcher .cwc, .bid, .bit files are in the same folder;

2. The ChipWatcher interface from the project cannot add, delete, and change signals;

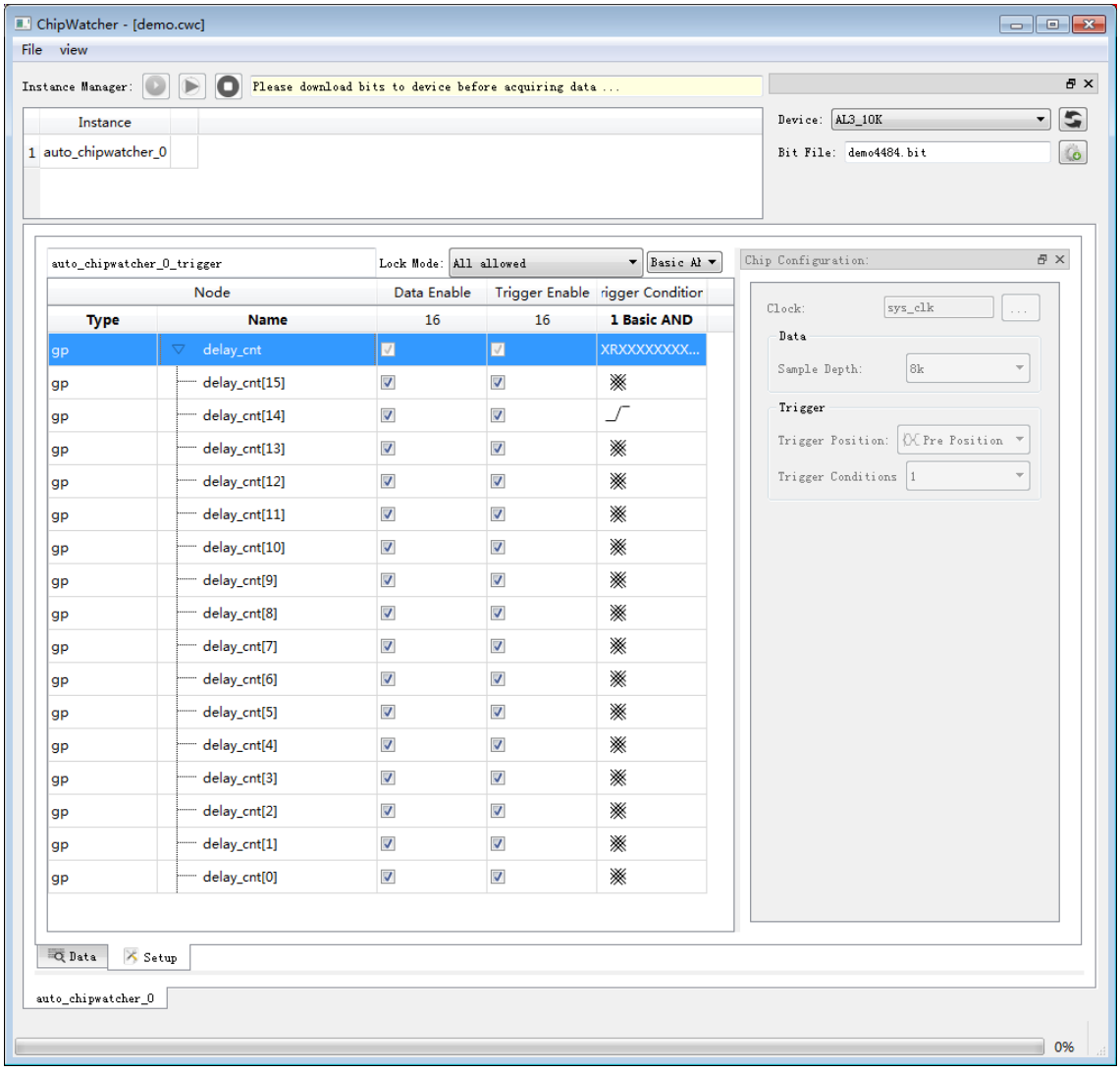
3. The sample depth and trigger position cannot be changed from the project's ChipWatcher interface.

The specific use is as follows:

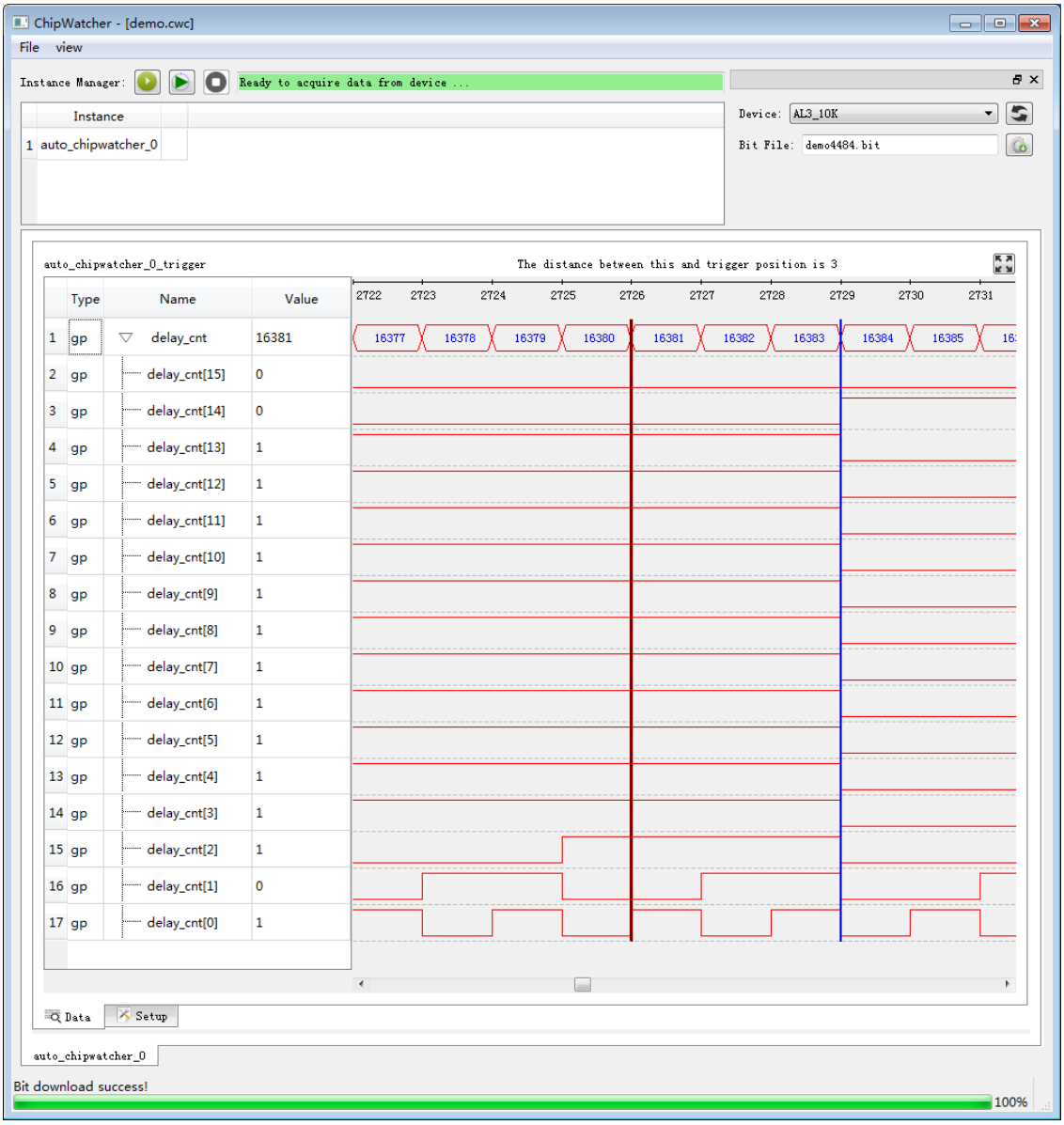
1. Open the ChipWatcher interface, you can only select "Open an existing chip watcher" and click "OK";



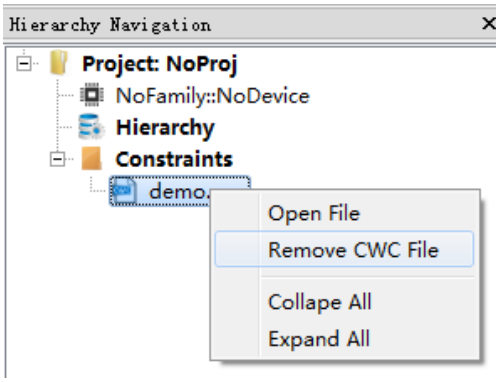
2. Open an existing .cwc file;



3. The trigger condition can be modified to download and trigger;



4. When you need to open another .cwc file, you need to close and remove the current file first.

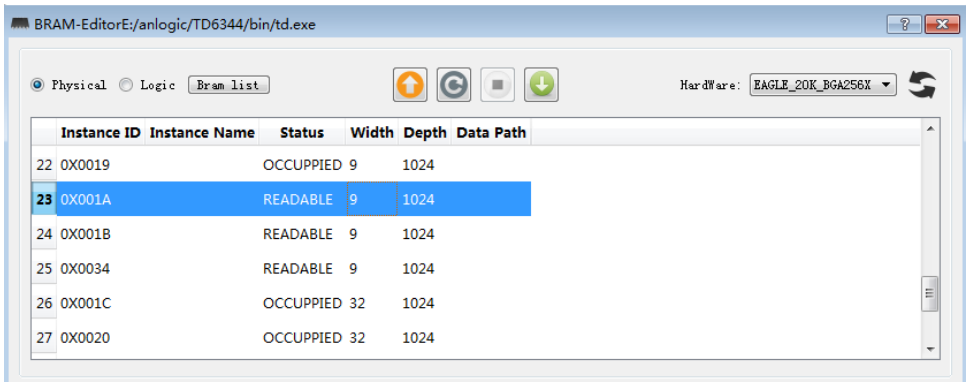


9.4 BramEditor

Users can use BramEditor to read data from the RAM in the chip, and modify the data, modify it and write it into the chip to see the effect.

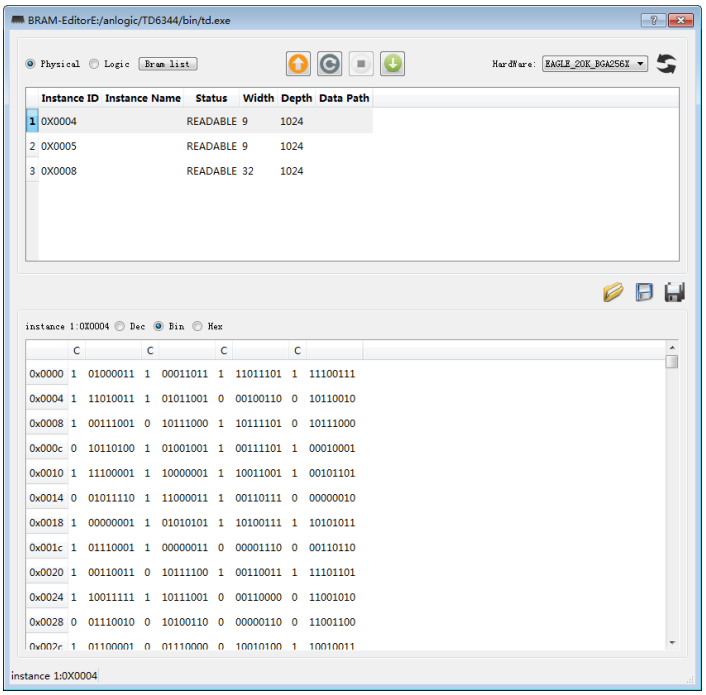
1. Expand Tools → Debug Tools and select BramEditor;

2. If “No Hardware” is displayed in the Hardware column, check whether the hardware interfaces are connected correctly, and whether the chip is powered on. Finally, click the refresh button next to refresh. Select an Instance in the pop-up BramEditor dialog box, and then read and write the Bram information. Only Instance with Status READABLE can read and write;

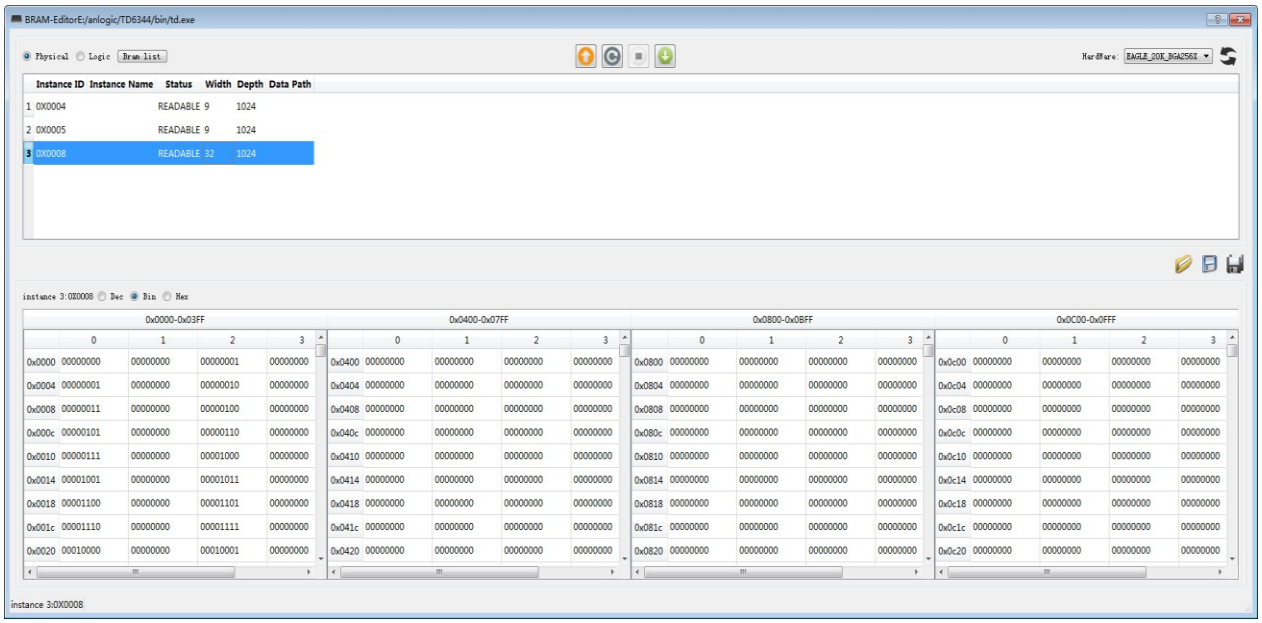


3. Click the button to read data from the chip. The user can choose to display the read back data in decimal (Dec), binary (Bin) or Hex (Hex). The default is binary. For Physical BRAM9K, the depth is 1024, the width is 9 bits, the highest bit is the parity bit (ninth bit); for Physical BRAM32K, the depth is 1024, the width is 32 bits, and there is no parity bit. For Logic BRAM, the depth and width are consistent with the user design;

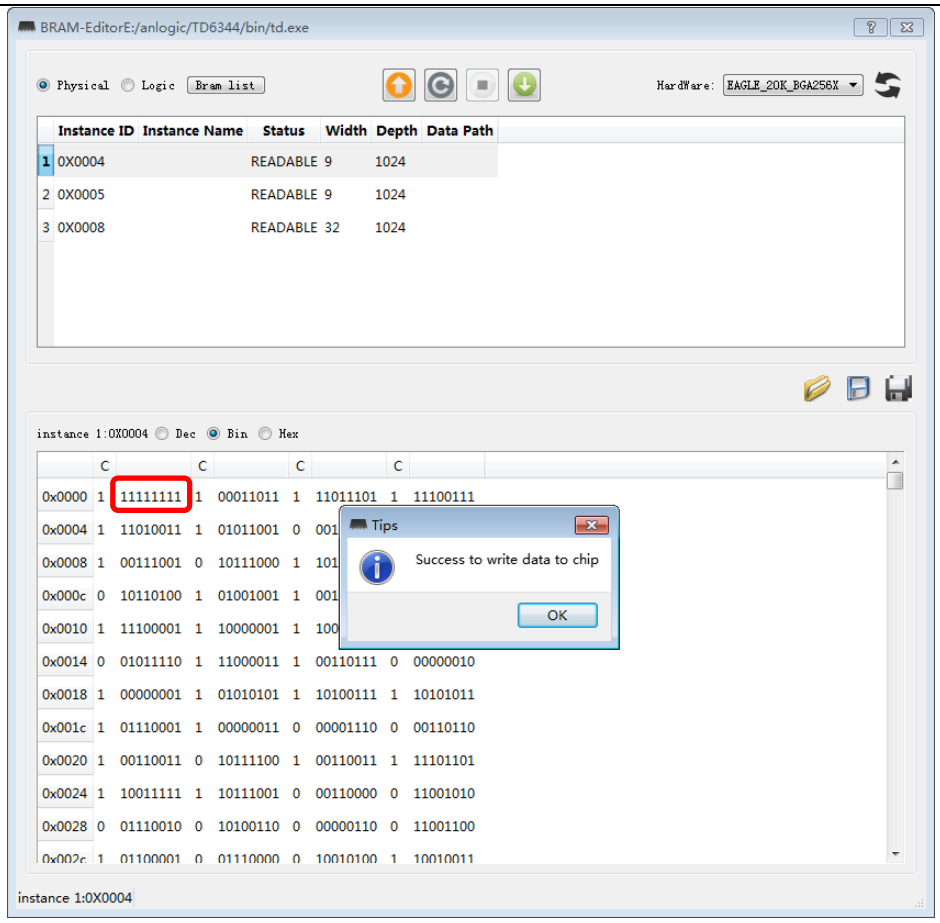
The data of BRAM9K is shown as follows:



The data of BRAM32K is shown as follows:

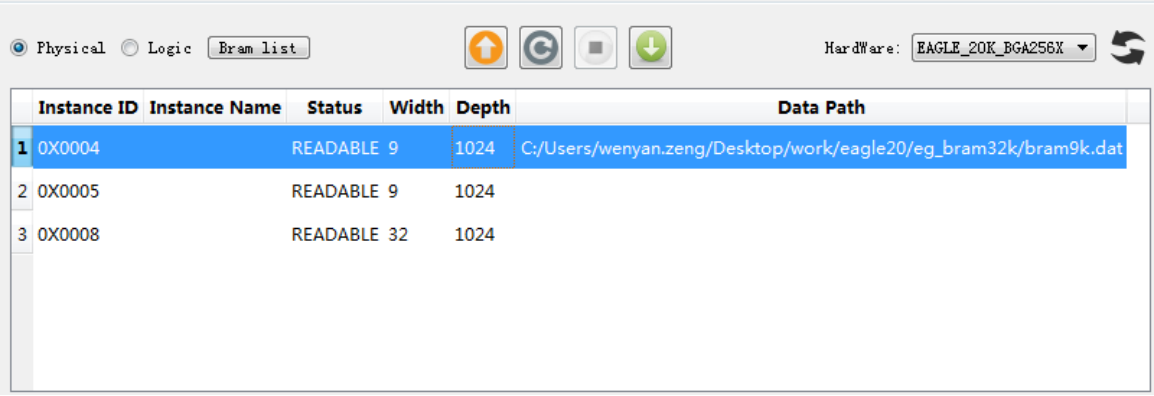


4. Double-click on a certain data to modify it. After modifying it, click the button to write the data back to the chip. You can use the button to cycle through the data and the button to stop the loop;

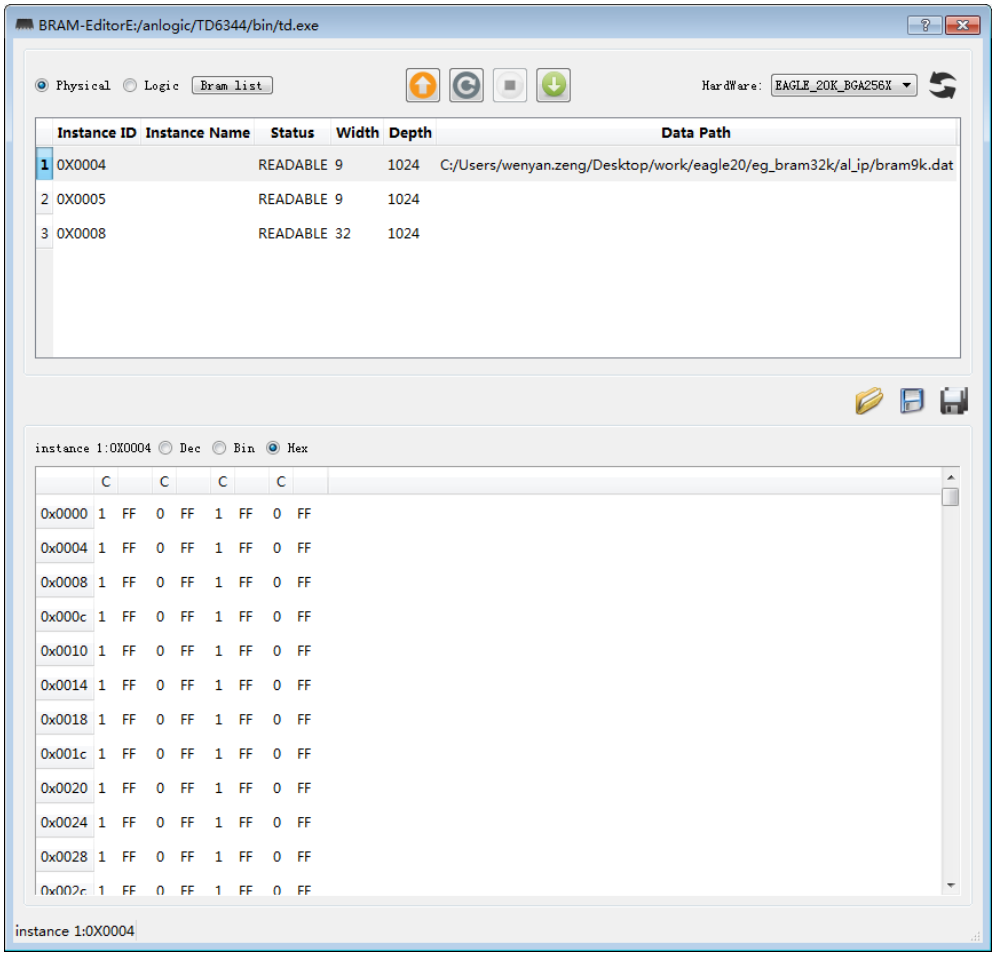


Bulk writes can also be made to the data in RAM.

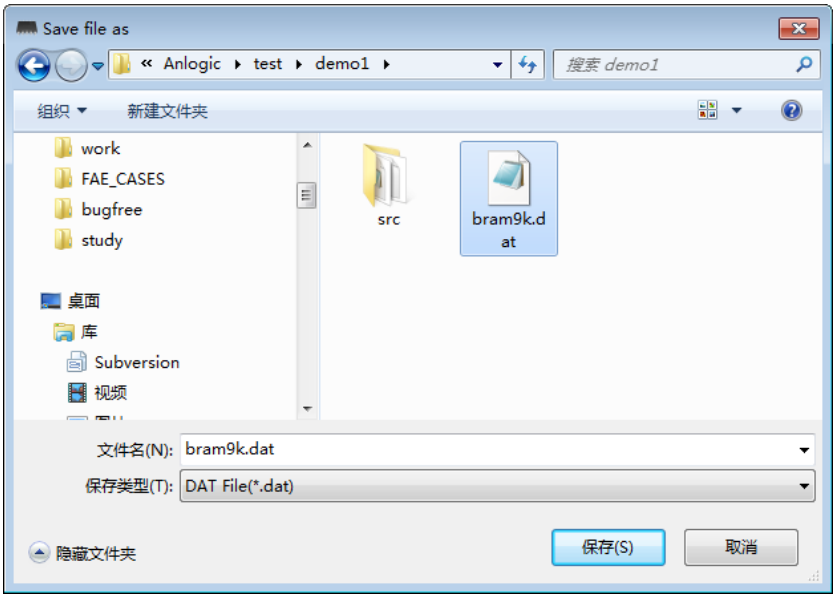
Select an Instance, click to open a .dat file to be written, click to write data to the chip, and the data will be successfully written. A warning is given if the data written does not match the size of the RAM. Finally click to view the data after writing.



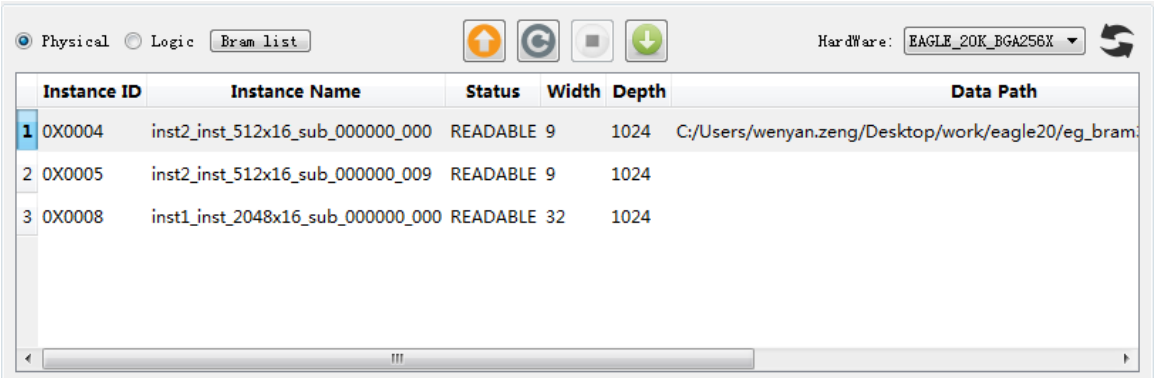
After writing to RAM, the data read back is as follows



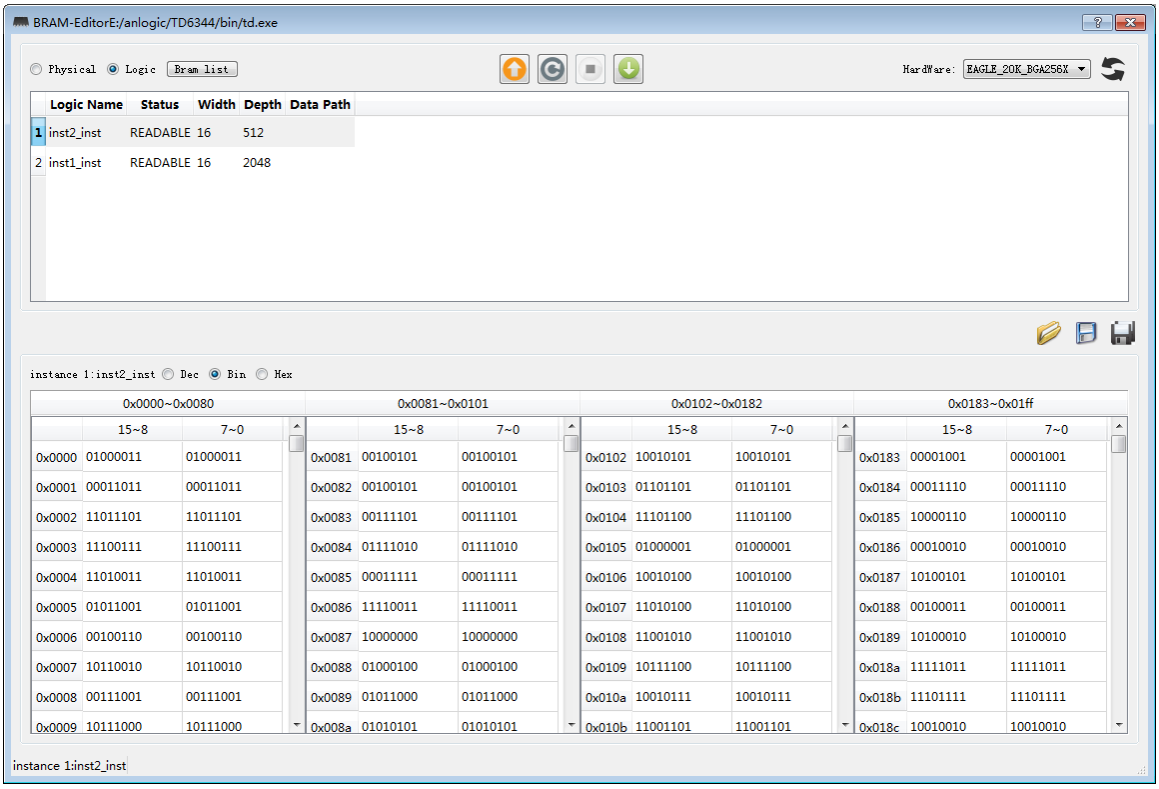
5. The data read back can be saved as a dat file with the click of button.



6. In the process of Flow, a .bid file containing BRAM Instance Name will be generated. In BramEditor, click the Bram list button in the upper left corner of the interface to match the internal Instance Name with the Instance ID, which is convenient for users to debug.



7. Click the Logic button to switch to the interface displayed by Logic BRAM. Other operations are consistent with Physical BRAM.

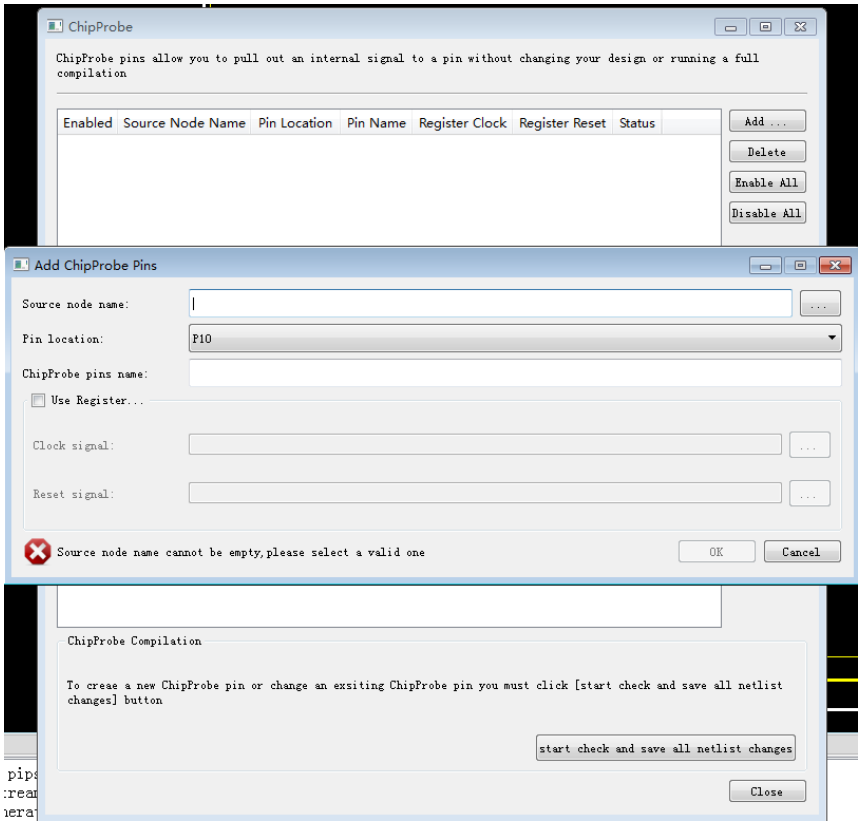


9.5 ChipProbe

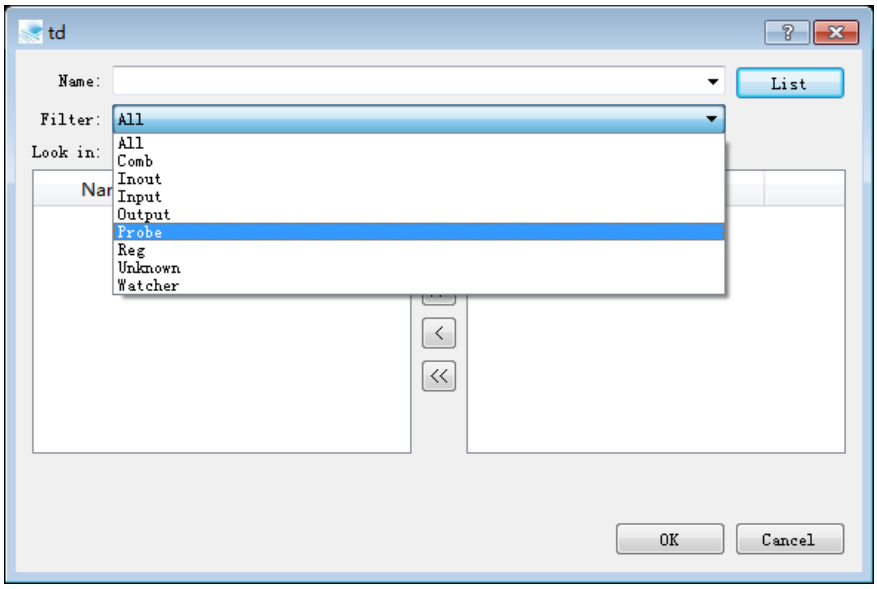
With ChipProbe, the user can pull some internal signals out to the IO port without changing the design, allowing the user to check the internal signal changes in real time with an external device.

1. After running the HDL2Bit process, expand Tools → Debug Tools and select ChipProbe

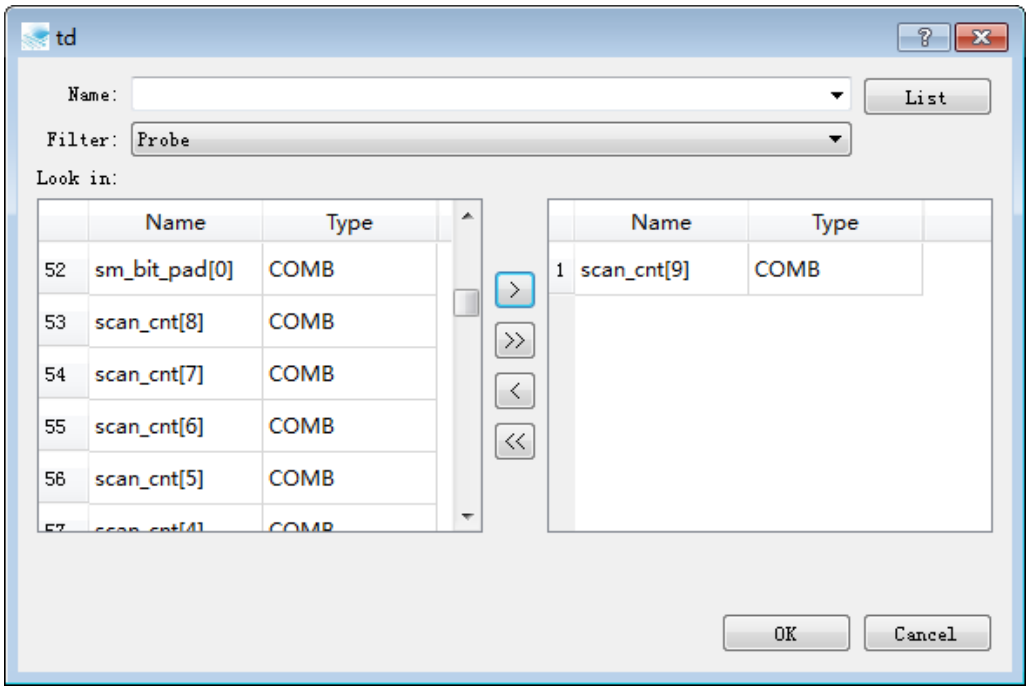
2. In the pop-up ChipProbe dialog, click Add to add the internal variables you want to view.



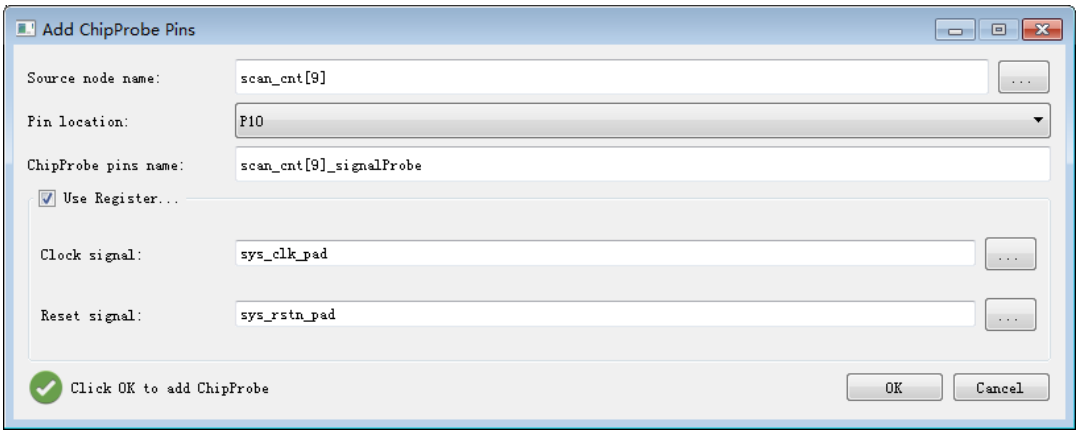
3. Users can manually enter the source node name or click ... to add, in the new pop-up dialog box, select the filter type as Probe. When using ChipProbe, debugging can only be initiated when the filtering type of the selected internal signal belongs to Probe.



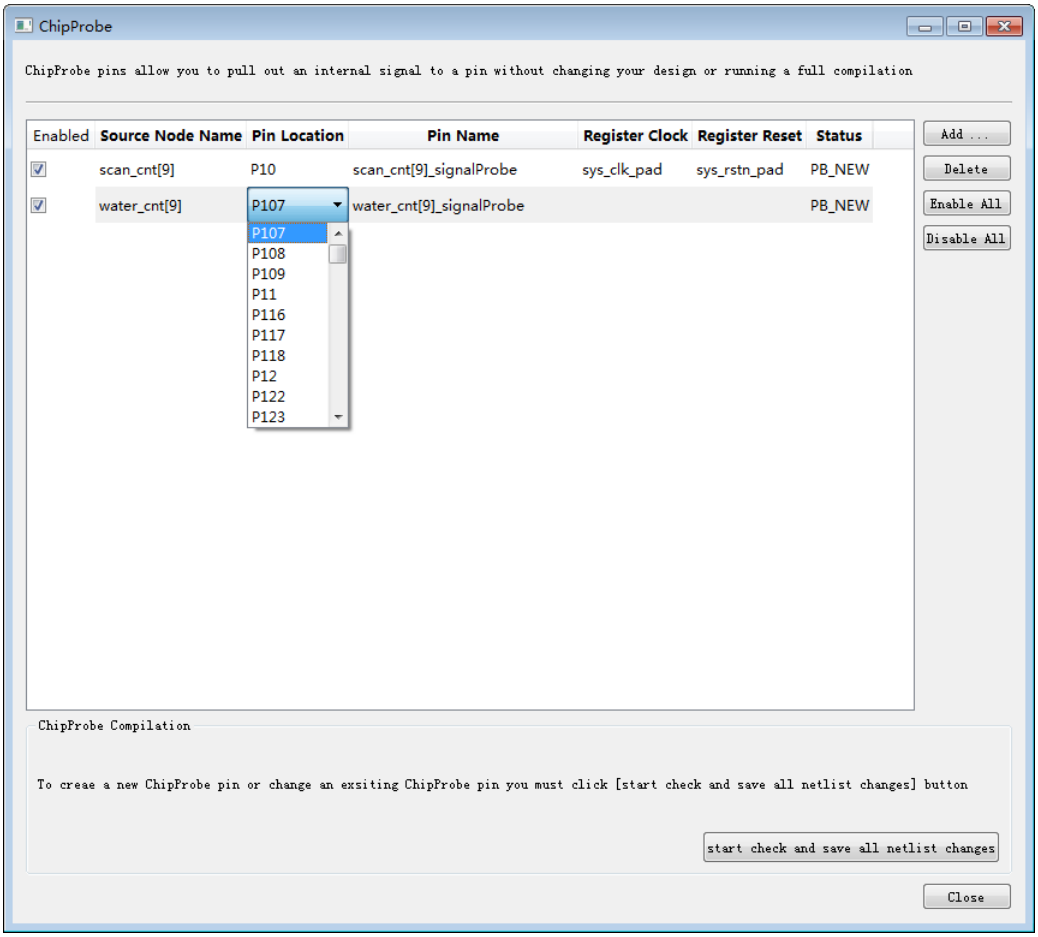
4. Click on List software manual, select an internal variable, click to > add it, click OK when you are done.



5. After the addition is complete, select an output pin for the internal signal. If you need to use register to latch the incoming internal signal, you need to select a Clock signal or add a Reset signal to it, as shown in the figure below. If you don't need to use register, just click OK.



6. It can be added by Add or deleted by Delete. The user can also change the output pin by right clicking on Pin Location.



7. You can check the small box below Enabled to activate an internal signal. You can also activate all signals by Enable all. After activation, select start check and save all netlist changes in the lower right corner to recompile the generated bit stream file and download it to the chip. Connect the external device to the pre-selected output pin for debugging.

10 Appendix

10.1 ADC constraint description

The ADC (Anlogic Design Constraint) file is used as the user physical constraint file of the TD software, and contains the constraints related to the various types of pins and unit physical information specified by the user.

1. Pin characteristic constraint

The definition is as follows:

set_pin_assignment　{pin_name}　{attributes}

Pin_name: the pin name of the circuit

Attributes: There are 9 kinds of physical related attributes, such as BANK, LOCATION, PULLTYPE, IOSTANDARD, PREEMPHASIS, SLEWRATE, DRIVESTRENGTH, VREF, DIFFRESISTOR, and PCICLAMP. Among them, for position constraint (LOCATION), it is checked whether the target position is legal, and whether multiple pins are assigned to the same position. Once the ADC file is added, it is required to lock the position for all pins.

Note: PreEmphasis pre-emphasis technology is a signal processing method that compensates the high-frequency components of the input signal at the transmitting end, which can effectively improve the output signal-to-noise ratio. Only LVDS output needs to be added, LVDS_E is the analog LVDS output, there is no pre-emphasis setting.

Format example:

```
> set_pin_assignment {sys_clk} {LOCATION=P23; PULLTYPE=PULLUP; VREF=
  NONE; SLEWRATE=MED; DRIVESTRENGTH=8; IOSTANDARD=LVCMOS25;
  VCM=0.8; VOD=350m}
```

表 10-1 IO 特性设置参数

PAMERATER	VALUE
BANK	BANK1, BANK 2, BANK 3, BANK 4, BANK 5, BANK 6, BANK 7, BANK 8
LOCATION	Different chip packages, the number of pins and pin names are different
PULLTYPE	PULLUP, PULLDOWN, NONE, KEEPER
IOSTANDARD	LVCMOS12,LVCMOS15,LVCMOS18,LVCMOS25,LVCMOS33,LVDS25, SSTL15_I,SSTL15_II,SSTL18_I,SSTL18_II,SSTL33_I,SSTL33_II, HSTL15_I,HSTL15_II,HSTL18_I,HSTL18_II, VREF1_DRIVER,VREF2_DRIVER
SLEWRATE	FAST, MED, SLOW
DRIVESTRENGTH	4, 8, 12, 16, 20, NA
VREF	VREF1, VREF2, NONE
DIFFRESISTOR	100, NONE
PCICLAMP	ON, OFF
VCM	0.8, 0.9, 1.2
VOD	150, 200, 250, 350, (add 480 for eagle_20 & eagle_s20)，单位为 mv

Note: The VCM, VOD option can only be set if the user has set up LVDS25.

2. Unit characteristic constraint

The definition is as follows:

set_inst_assignment {inst_name} {attributes}

Inst_name: The cell instance name of the circuit. The name of the instance can be found in the generated netlist file (prj_gate_sim.v).

Attributes: Currently only location constraints are supported. Location can be found in Chip Viewer. For a detailed manual of Chip Viewer, please refer to the 9.2 Chip Viewer chapter of this document.

Format example:

> set_inst_assignment {PLL_INST1} {location = x34y37z0;}
> set_inst_assignment {PLL_INST2} {location = x0y0z0;}
> set_inst_assignment { _al_u451|lcd/lcd_rs_reg} {location = x21y17z1;}

10.2 SDC constraint description

The Timer module of TD supports the user-entered timing constraint format as a common subset of commands in the SDC standard format. The SDC commands supported by the TD and related parameter options are listed below.

1. Refers to the object related:

   a) **all_clocks / all_inputs / all_outputs / all_registers**：Refers to all objects of this category;

   b) **get_cells / get_pins / get_nets** [-hierarchical] [-nocase] [-nowarn] <filter>

   Returns a collection of cells / pins / nets in the design, -hierarchical for hierarchical search, matching the corresponding name in each level of the module, -nocase means case insensitive, and -nowarn means no warning when no match is successful .

   c) **get_clocks / get_ports** [-nocase] [-nowarn] <filter>

   Returns a collection of clocks/input and output ports with the same meaning as above.

2. Clock setting related (clock information is the most basic timing constraint):

   a) **create_clock** -add -name <string> -period <double> -waveform <string> target

   Define a clock: target indicates the clock source, which can be nets or ports. If target is empty, it means a virtual clock is defined; -name specifies the clock name. If the item is empty, the clock is named the first item in the target list. ;-period is the period, this option must be specified, and the value needs to be greater than 0; -waveform specifies the time point of the first rising edge and the first falling edge, and temporarily supports only two clock edges per cycle; Add indicates that a new clock is added to the pin when the clock has been defined on the target pin. Otherwise, the existing clock is overwritten, mainly for the clock multiplexer.

   Format example:

   > create_clock –name sys_clk -period 20 -waveform {0 10} [get_ports sys_clk]

b) **create_generated_clock** -add -name <string> -source <list> -divide_by <double> -multiply_by <double> -edges <string> -duty_cycle <double> -invert -edge_shift <string> -master_clock <string> target

Define a derived clock that belongs to the clock domain where the master clock is located. It will also have the same start point as the master clock in the timing report.

-source specifies the source point where its master clock is located, which can be a list of nets or ports.

-master_clock specifies the name of the master clock; -name specifies the clock name, if the item is empty, the clock is named the first item in the source list; target is the source point of the currently generated clock.

-add Description Adds a new clock to this pin if the clock has been defined on the target pin. Otherwise, the current generated clock is ignored, which is different from the master clock definition. The period and waveform of the generated clock are adjusted by the master clock.

-divide_by / -multiply_by refers to the frequency divided by / multiplied by the specified multiple,

-invert works with these two options to invert the clock waveform, and -duty_cycle is used with the -multiply_by option to adjust the duty cycle;

-edges option consists of three integers that specify the first rising edge of the newly generated clock, the first falling edge, and the second rising edge corresponding to the first edge of the source c

-edge_shift option specifies the offset of the three clock edges in the -edges option, so it will contain 3 positive and negative integers in base time units (default is ns).

Format example:

> create_generated_clock -divide_by 1.25 -source [get_ports clk] -name sys_clk [get_ports sys_clk]

c) **derive_pll_clocks** [-gen_basic_clock]

Automatically generate clock constraints on all used PLL clkc[x] ports, the frequency and phase of the generated clock will be strictly in accordance with the internal parameters of the PLL.

-gen_basic_clock will define the reference clock for the FIN frequency on the corresponding PLL refclk, otherwise it will automatically search for the refclk pin and the clock defined on the connected net.

The clock generated by this command will not take effect until flow is running, so it cannot be referenced in subsequent settings of timing wizard.

> derive_pll_clocks -gen_basic_clock

d) **derive_clocks** -period <double> -waveform <string>

Specify a default clock on the clock pin of each undefined clock,

-period is the period, this option must be specified, and the value needs to be greater than 0;

-waveform specifies the time point of the first rising edge and the first falling edge. Temporarily only supports the case where there are two clock edges per cycle.

Format example:

> derive_clocks -period 10 -waveform {0 5}

e) **set_clock_latency** -clock <list> -max -min -source delay

Set the delay of the clock, delay is the value of the delay; -clock specifies the list of clocks;

-max/ -min option specifies the maximum/minimum delay specified by this command, the default is the same;

-source option description Specified as source latency, otherwise network latency. Network latency is replaced by the actual interconnect delay in the timing analysis after routing.

> set_clock_latency -clock [get_clocks {clk_vga_25m}] -source 2

f) **remove_clock_latency** -clock <list> -source target

Cancel the previously set clock delay, each option has the same meaning as above.

Format example:

> remove_clock_latency -source [get_clocks clk_vga_25m]

g) **set_clock_uncertainty** -setup -hold uncertainty target

Set the clock's uncertainty value, currently only supports the setting of the individual clock itself and does not support the definition of the clock across the clock domain; the target is the list of clock, the uncertainty is the value item;

-setup/-hold option indicates that the command corresponds to the maximum / The value corresponding to the minimum path timing analysis. If this option is not specified, both checks will take effect at the same time.

Format example:

```
> set_clock_uncertainty -setup -hold 1.00 [get_ports clk]
```

h) **remove_clock_uncertainty** -setup -hold target

Remove the previously set clock uncertainty value, each option has the same meaning as above.

Format example:

```
> remove_clock_uncertainty -setup -hold [get_ports clk]
```

3. Timing path constraint settings:

a) **set_input_delay / set_output_delay** -clock <list> -max -min delay target

Set the delay of the input/output port. The delay item is the delay value. The target can be the object list of pins or ports. The -clock specifies the clock information corresponding to the delay.

-max/-min option specifies the command. The value set is the maximum/minimum delay and the default is the same.

Format example:

```
> set_input_delay -clock sys_clk 1.0 [all_inputs]
> set_output_delay -clock lcd_clk 3 [get_ports {disp_RGB} {led} {sm_bit}]
```

b) **remove_input_delay / remove_output_delay** -max -min target

Remove the previously set input/output delay. The parameters have the same meaning as above.

Format example:

> remove_input_delay -clock sys_clk 2 [get_ports {data[0]}]

> remove_output_delay -clock sys_clk -0.55 [all_outputs]

c) **set_max_delay / set_min_delay** -from <list> -to <list> -through <list> delay

Set the allowable maximum/minimum delay of the timing path, the delay term is the delay value;

-from must be the starting point of the timing path, ie the list of inputs;
-to must be the end point of the timing path, ie the list of outputs;
-through option It can be nets, a list of ports that specifies the intermediate point that the timing path must pass. When there are multiple through options, the target timing path must pass through each intermediate point in turn.

Format example:

> set_max_delay -from [get_ports {sw}] -to [get_ports {sm_bit}] -through [get_nets {lcd/delay_cnt[2]}] 1

d) **set_false_path** -setup -hold -from <list> -to <list> -through <list>

The timing path is set to a spurious path, so it is not time-stamped;
-setup/-hold option specifies that the target path is not analyzed during setup/hold check.
-from is the starting point of the time series path, which can be a list of clocks or inputs,
-to is the end point of the time series path, can be a list of clocks or outputs,
-through specifies the intermediate point through which the time series path must pass, but is ports or nets list of.

Format example:

> set_false_path -from [get_clocks clk] -to [get_clocks sys_clk]

> set_false_path -from [get_clocks {clk_vga_25m}] -to [get_ports {sm_bit}] -through [get_nets {dpram_top0/lfsr_done}]

e) **set_multicycle_path** -setup -hold -start -end -from <list> -to <list> -through <list> multiplier

Set the timing path that allows multiple clock cycle delays. The multiplier entry is the number of clock cycles.

-setup/-hold option sets the timing path type that the command constrains. When only the -setup option is used, setup check will allow the most sequential path. Use N clock cycles, but at the same time hold check will become the required timing path for the shortest N-1 clock cycles. When only the -hold option is used, the hold check constraint is set to N without affecting the setup check constraint.

In the normal usage scenario, in order to enable setup check to use multiple clock cycles without affecting hold check, two commands are required to be used together, that is, set a N period of setup constraint, and then with a N-1 period hold constraint. .

-start/-end is the option used across clock domains, specifying the delay to the period corresponding to the launch/capture clock. By default, setup check uses the capture clock and hold check uses launch clock. -from is the starting point of the time series path, which can be a list of clocks or inputs, -to is the end point of the time series path, can be a list of clocks or outputs, and -through specifies the intermediate point through which the time series path must pass, but is ports or nets list of.

Format example:

> set_multicycle_path -setup -end -from [get_clocks {clk_vga_25m}] -through [get_nets {uart/uart_tx/num[3]}] 2

f) **set_clock_group** -exclusive -asynchronous -group <list>

Set the packet for the clock domain without analyzing the timing path across the clock group. In general, the -exclusive option means that these clock groups do not logically appear at the same time, and -asynchronous means a completely irrelevant clock, but in terms of implementation and effect, the two options are the same, the command will be at all A set of false path constraints is defined between the clocks listed by -group.

> set_clock_groups -exclusive -group [get_clocks {clk_vga_25m}] -group [get_clocks {sys_clk}]

g) **set_clock_route** <net_name>

Used to specify a specific clock line network without a dedicated clock interconnect. When the number of clocks in design exceeds the limit of TD, you can manually specify which clock lines do not go through the clock network. If the specified clock does not go through the clock network, it will cause the layout and routing to fail, and the TD will give an error in time.

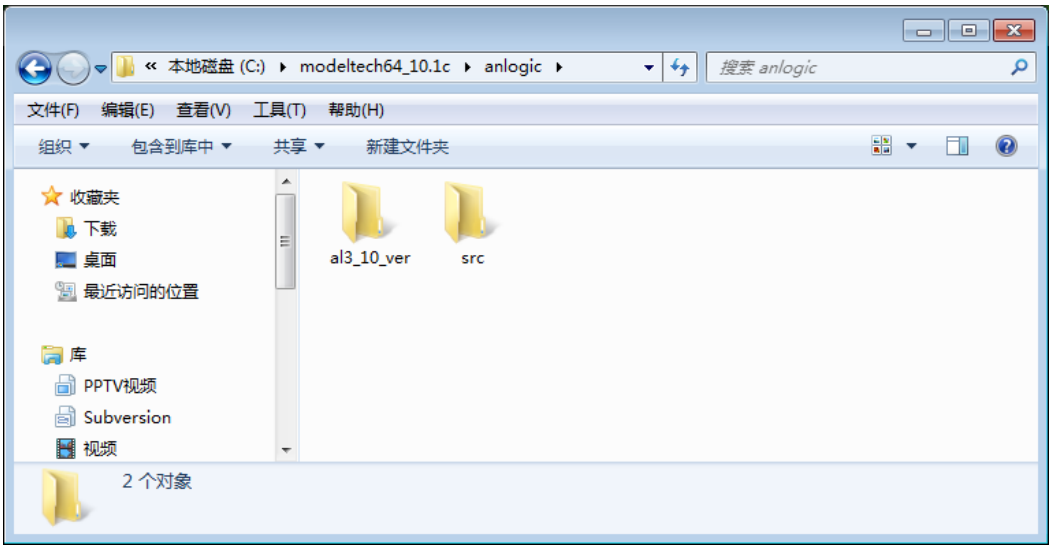
> set_clock_route lcd12864_en_pad

10.3 ModelSim simulation process

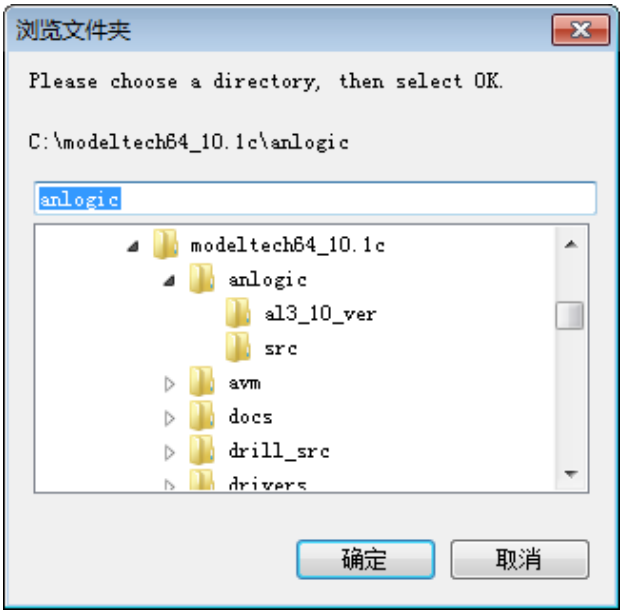
10.3.1 Add simulation library

Taking the AL3_10 device as an example, the TD software comes with a simulation model and can be compiled in modelsim as follows:

1. In the installation directory of modelsim, create a new folder, such as: Anlogic,

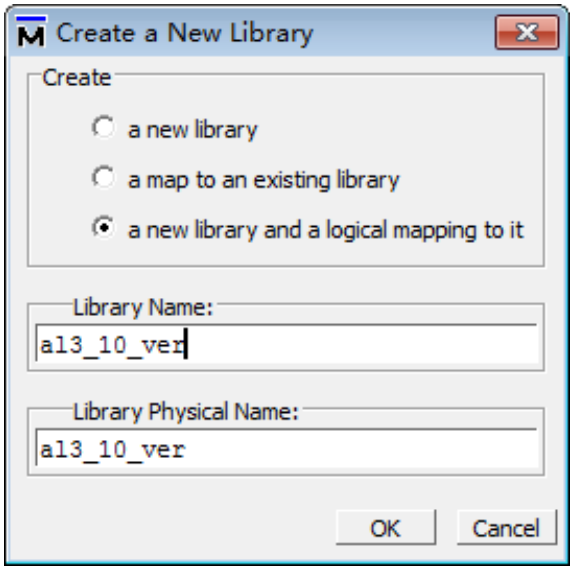


2. Start modelsim and select file → change directory to go to the anlogic folder

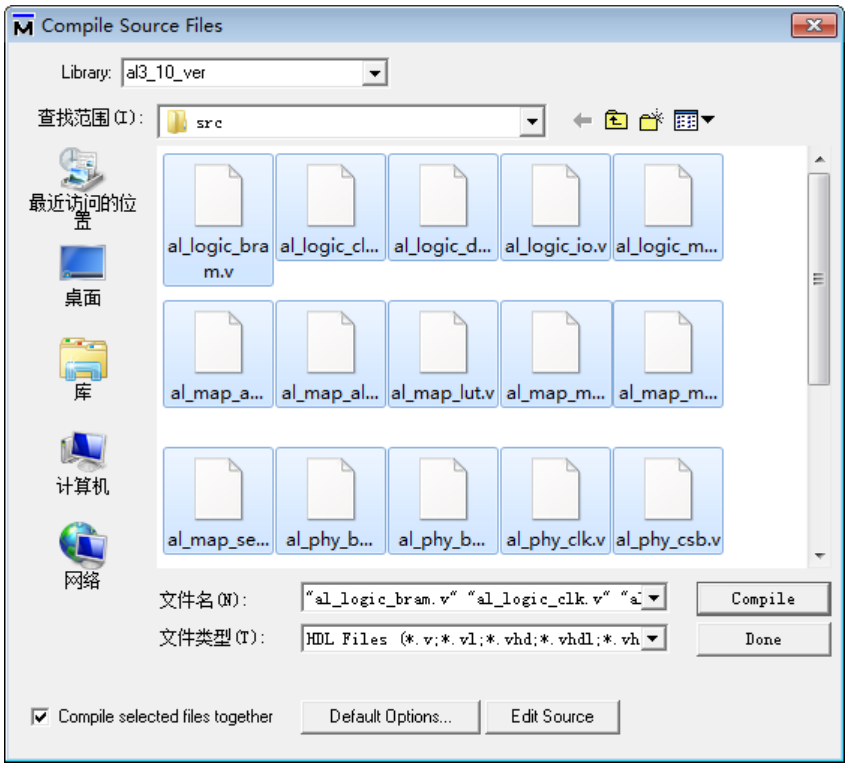


3. Create a new folder under the anlogic folder, such as src, to store the TD's simulation model source files, and copy all the files in the sim directory under the TD installation path.

4. Create a new library named al3_10_ver under modelsim's file → new → library

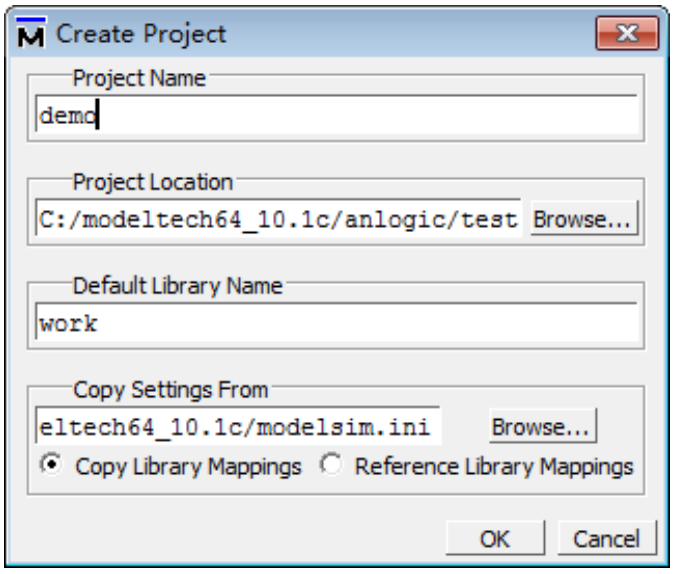


5. Open compile → compile, pop up the compile souce files dialog box, select the newly created al3_10_ver in the library, find all the files under the src search range, check compile selected files together, execute the compile command

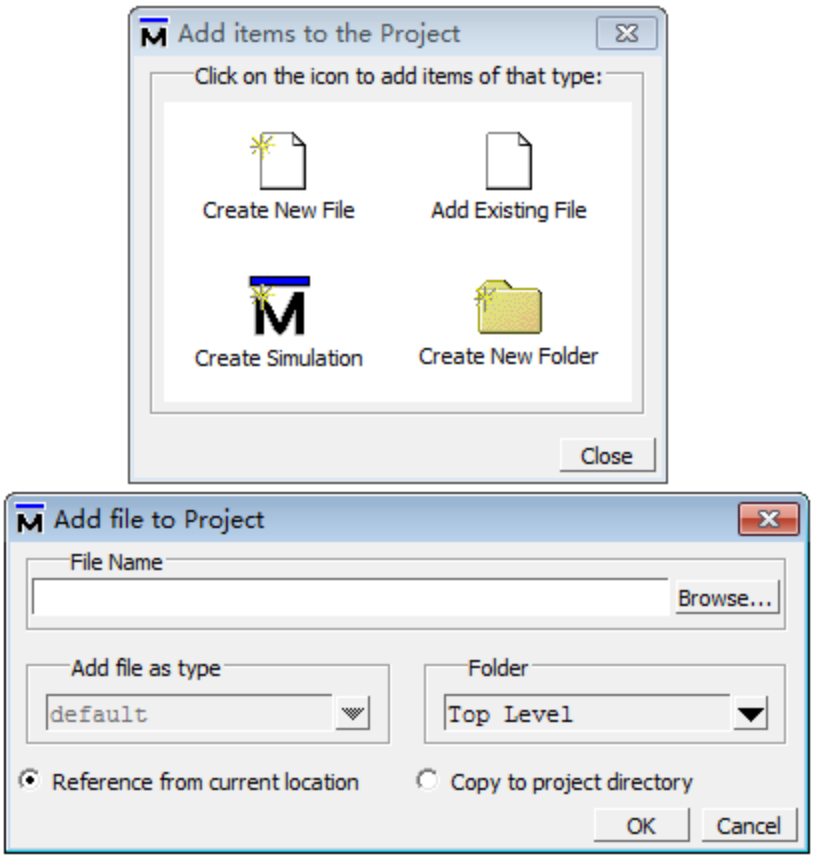


10.3.2 Simulation

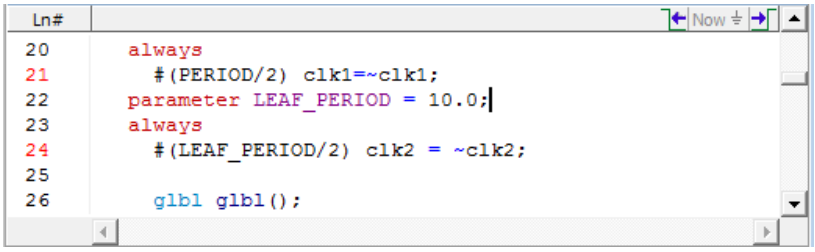
1. In modelsim, click file → new → project to create a new project, such as: demo

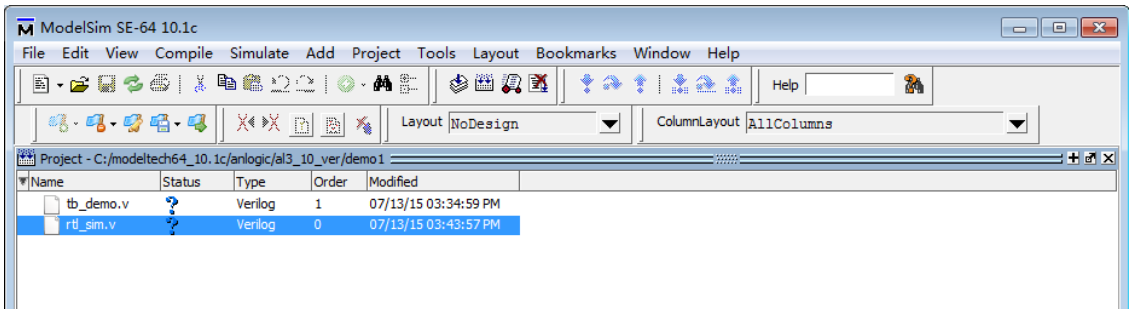


2. You can add a design file by clicking add existing file, or you can create a new design file by clicking Create New File and add it to the project.

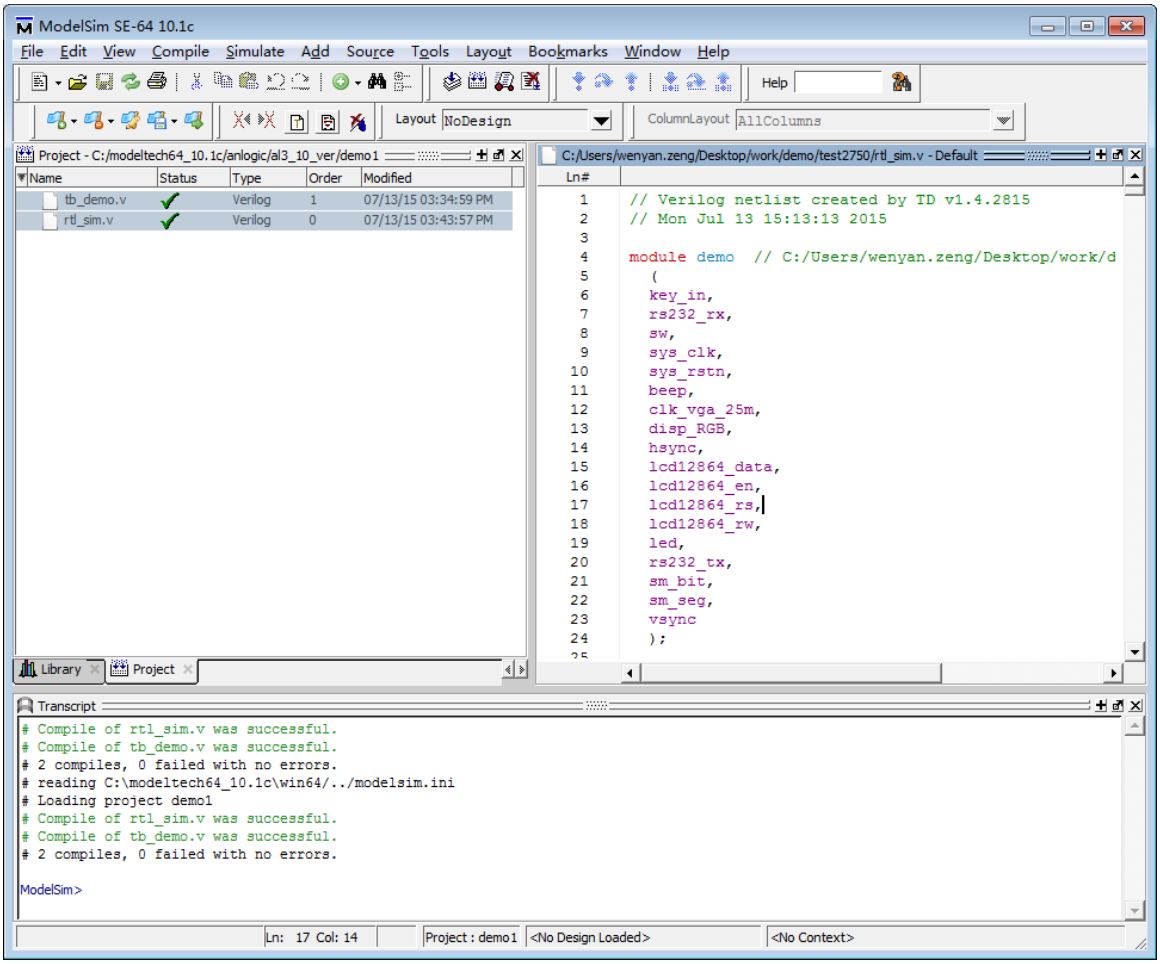


3. Select an existing design source file and its testbench file. Here we use TD-generated RTL-level circuit simulation model rtl_sim.v as an example to simulate. If you encounter problems with glbl during simulation, please refer to Anlogic's glbl module in testbench as follows:

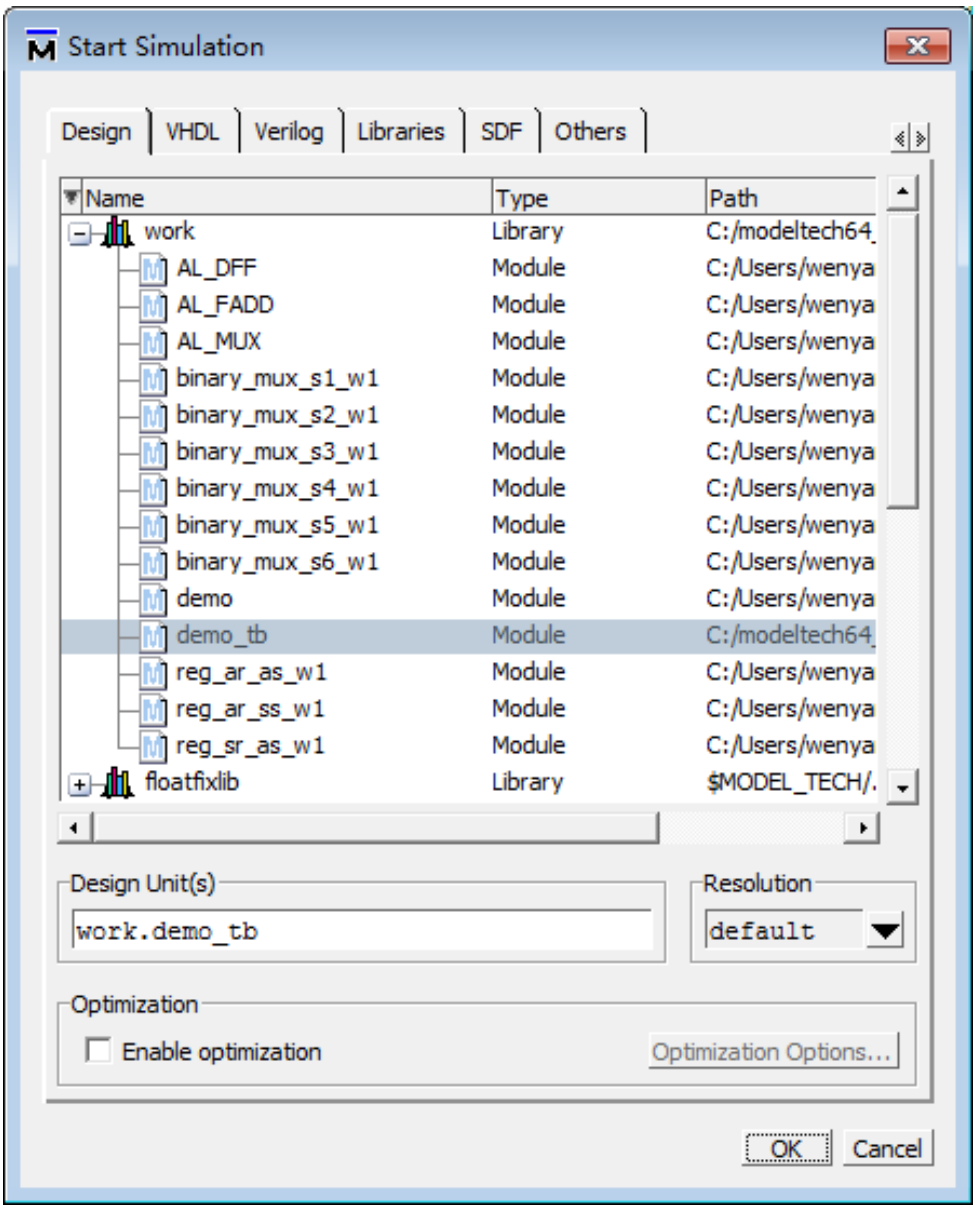




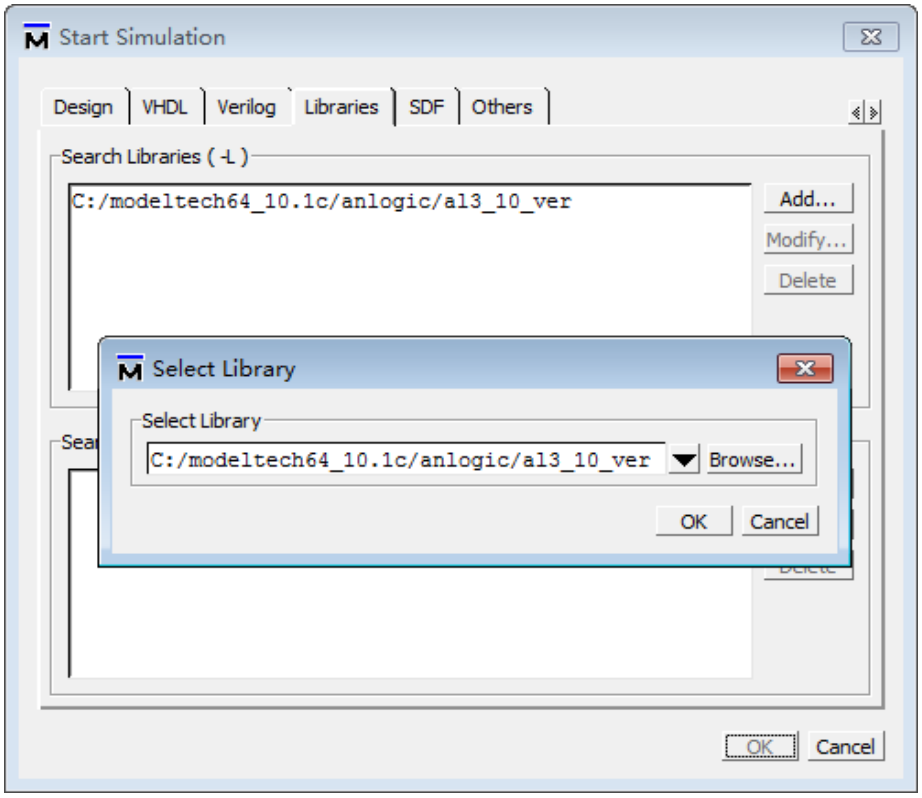
4. Click to compile, after the compilation is successful, the status of the source file will change from "?" to "✔"



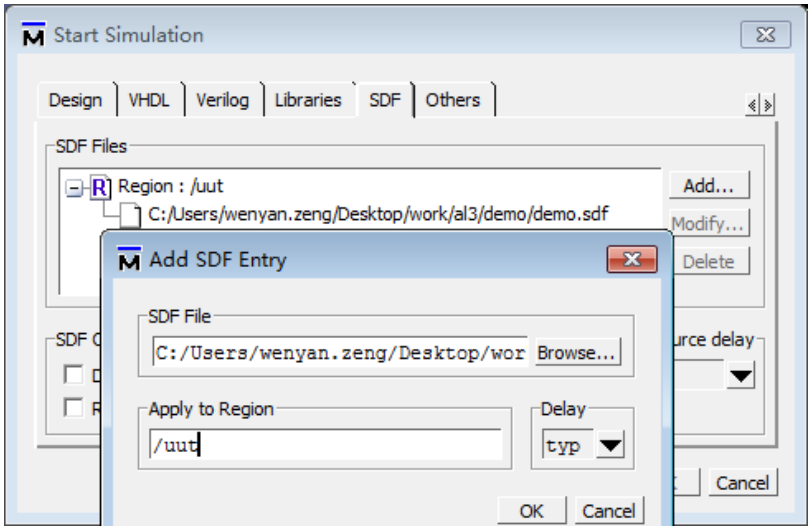
5. Click simulate → start simulate, select the testbench file in the work library to simulate. If you want to view the changes of each signal parameter or waveform in the module list after simulation, you can remove the check mark in front of “Enable optimization”. Otherwise, Modelsim will Optimize the signal parameters, causing the signal list to be empty.



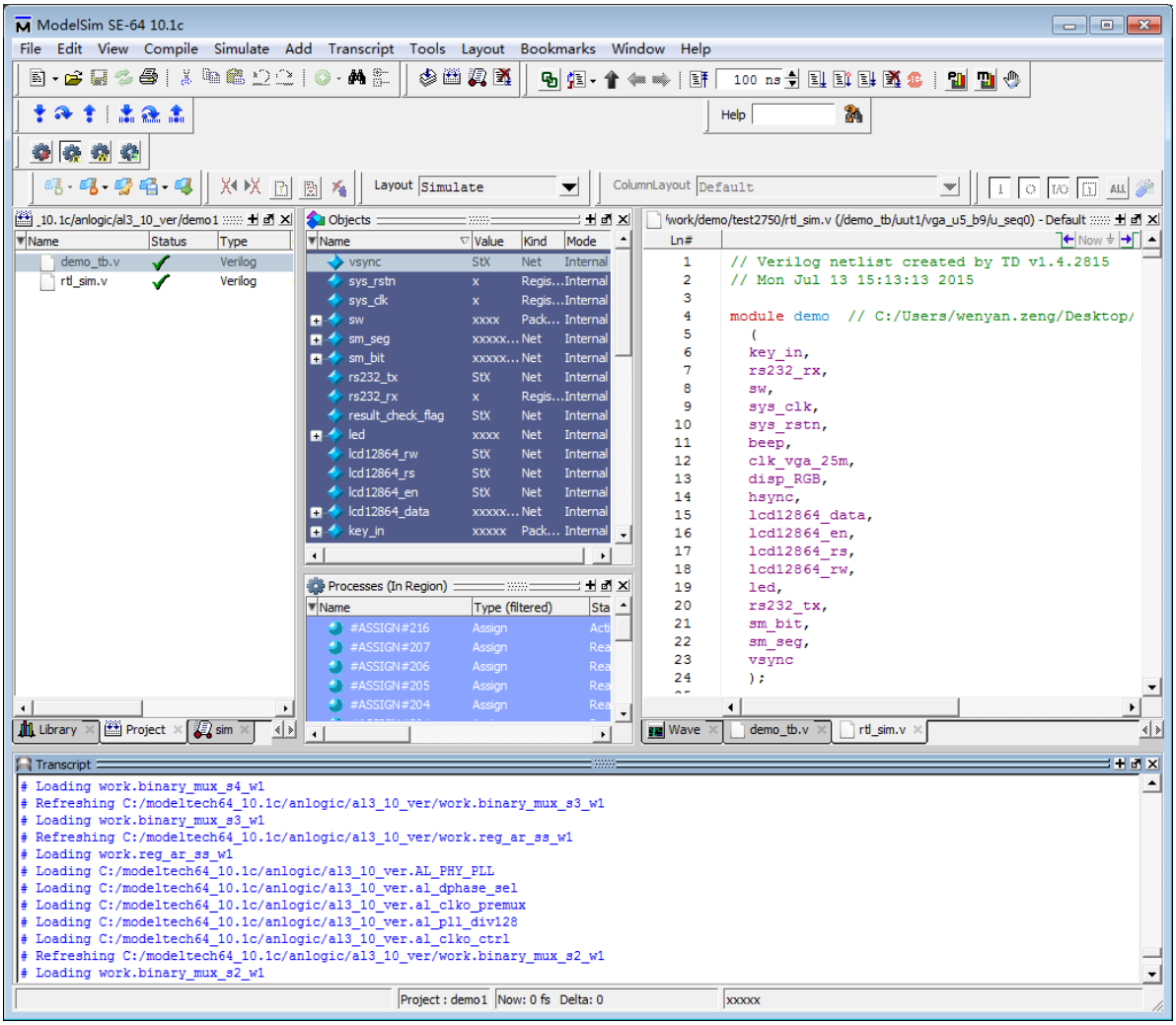
6. Then select libraries, click add, select al3_10_ver, and click OK to simulate.



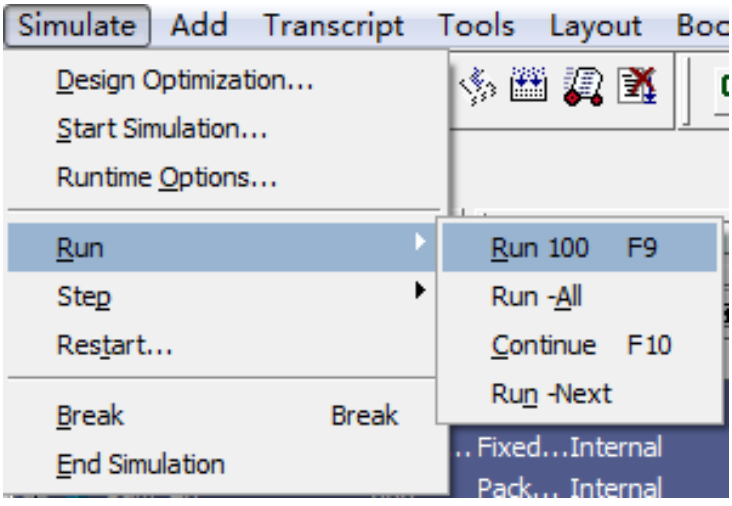
Note: For timing simulation, specify the project's sdf file during simulation, where Apply to Region refers to the instantiated Instance name in the tb file. The netlist file for timing simulation is: prj_name_phy_sim.v



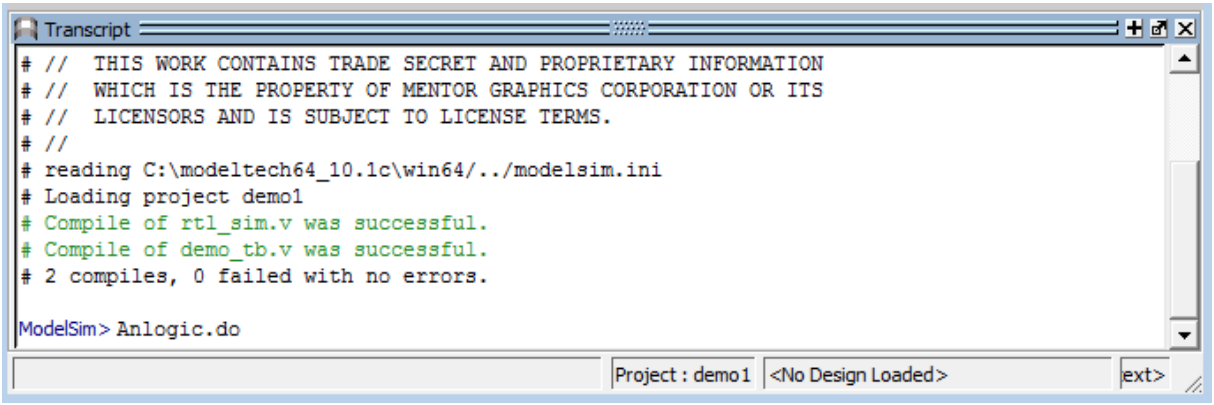
7. After the simulation is complete, you can view the list of signals under Objects. You can view the waveform changes by right-clicking on a signal and selecting “Add wave”.



8. Click Simulate → Run → Run 100, or click in the navigation bar to run the simulation 100ns. You can also enter the simulation time manually.



The above process can be done directly by running a script, such as a script: Anlogic.do



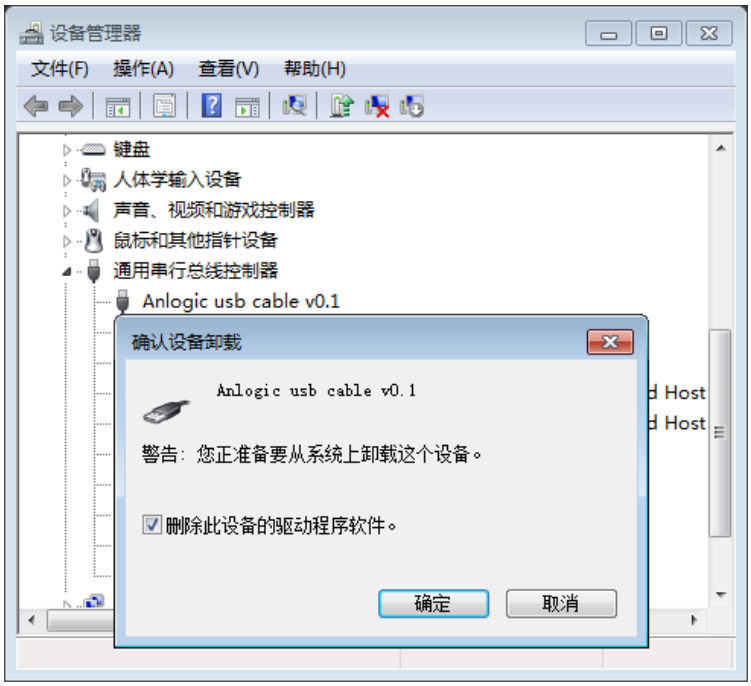
Anlogic.do is as follows:

```
#
# Create work library
#
if {[file exists work]} {
   vdel -lib work –all }
vlib work
#
# Compile sources
#
vlog    "C:/Anlogic/test/work/test.v"
vlog    "C:/ Anlogic/test/work/TestBench. v"
vlog    "C:/ Anlogic/test/work/Addbit.v"
#
# Call vsim to invoke simulator
#
vsim -voptargs="+acc" -L al3_10_ver -gui work.TestBench
#
# Source the wave do file
#
add wave *
#
# Set the window types
#
view wave
view structure
view signals
run 100ns
```

10.4 USB download driver installation

USB driver installation

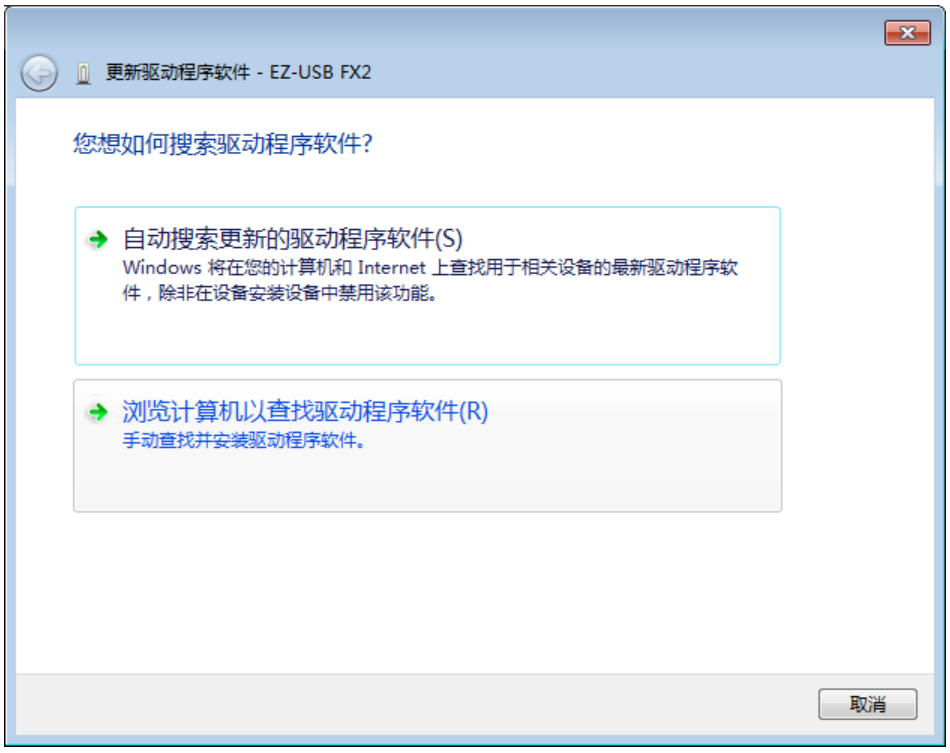
The TD software has been updated with the USB driver. If the user has previously installed the ReadyDriverPlus software and the Anlogic usb cable driver, please uninstall the software and driver and reinstall the USB driver.

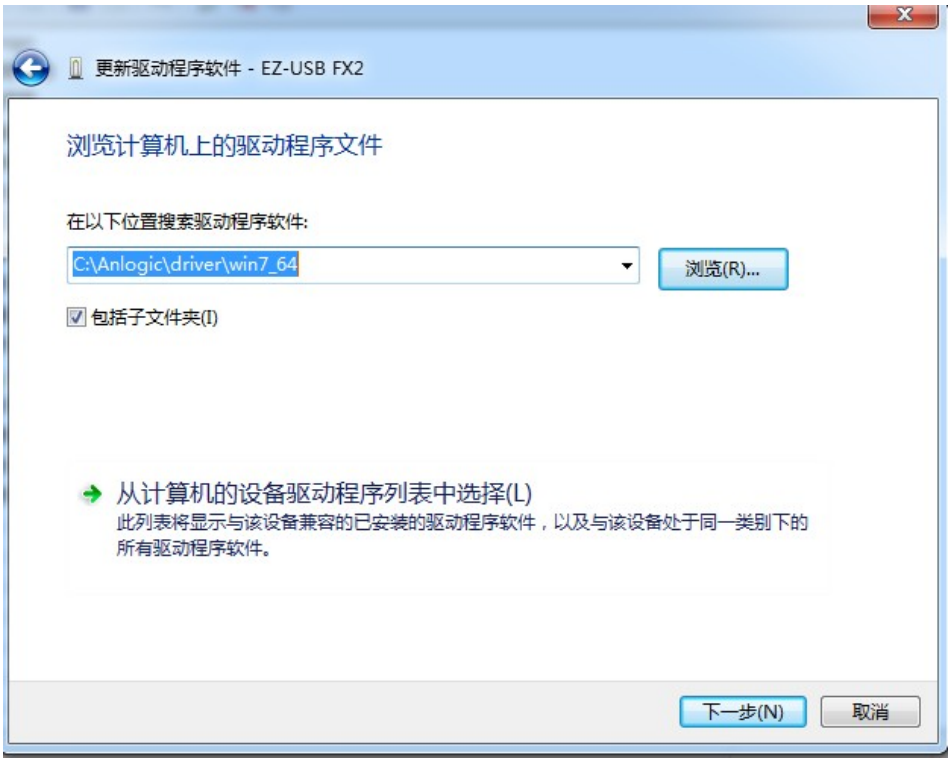


The specific installation steps are as follows:

1. Insert the download cable into your computer and open the device manager. You can see the unknown device in "Other devices" EZ-USB FX2

2. Right click on EZ-USB FX2 and select “Update Driver Software”. The following prompt will pop up, select “Browse my computer to find driver software” and select driver under TD installation path.





3. According to the system, choose 64/32-bit drivers of win7, win8 or win10, such as:

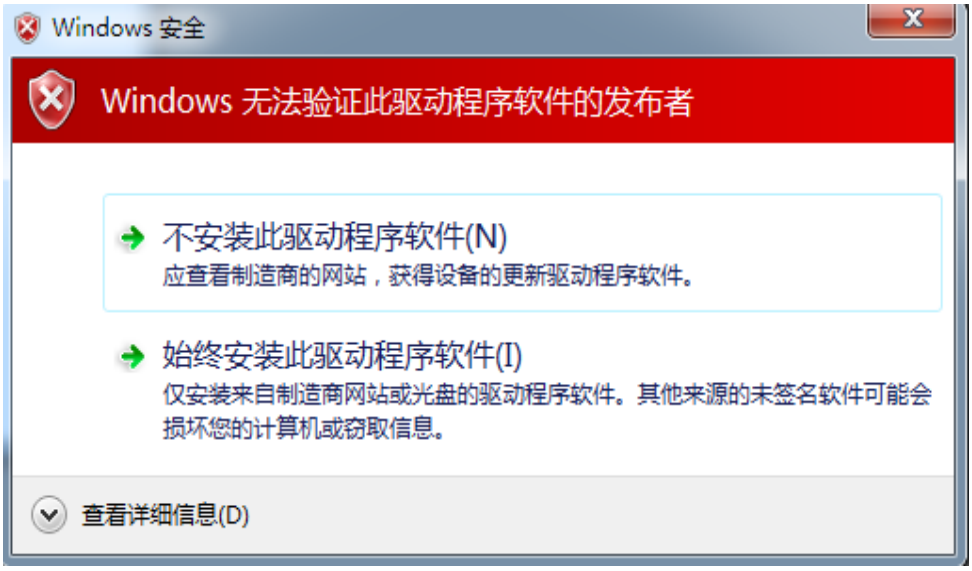
   Win7 32-bit system, choose win7_32;

Win7 64-bit system, select win7_64;

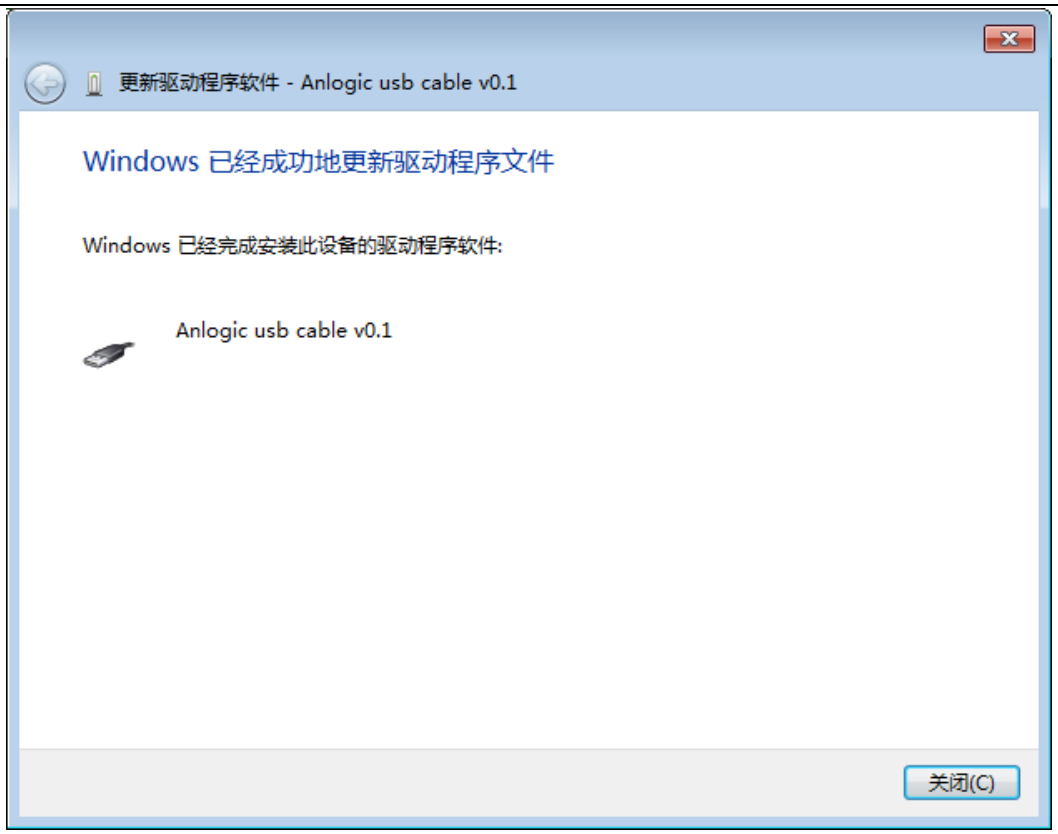
For win8 or win10 32-bit systems, select win8_10_32;

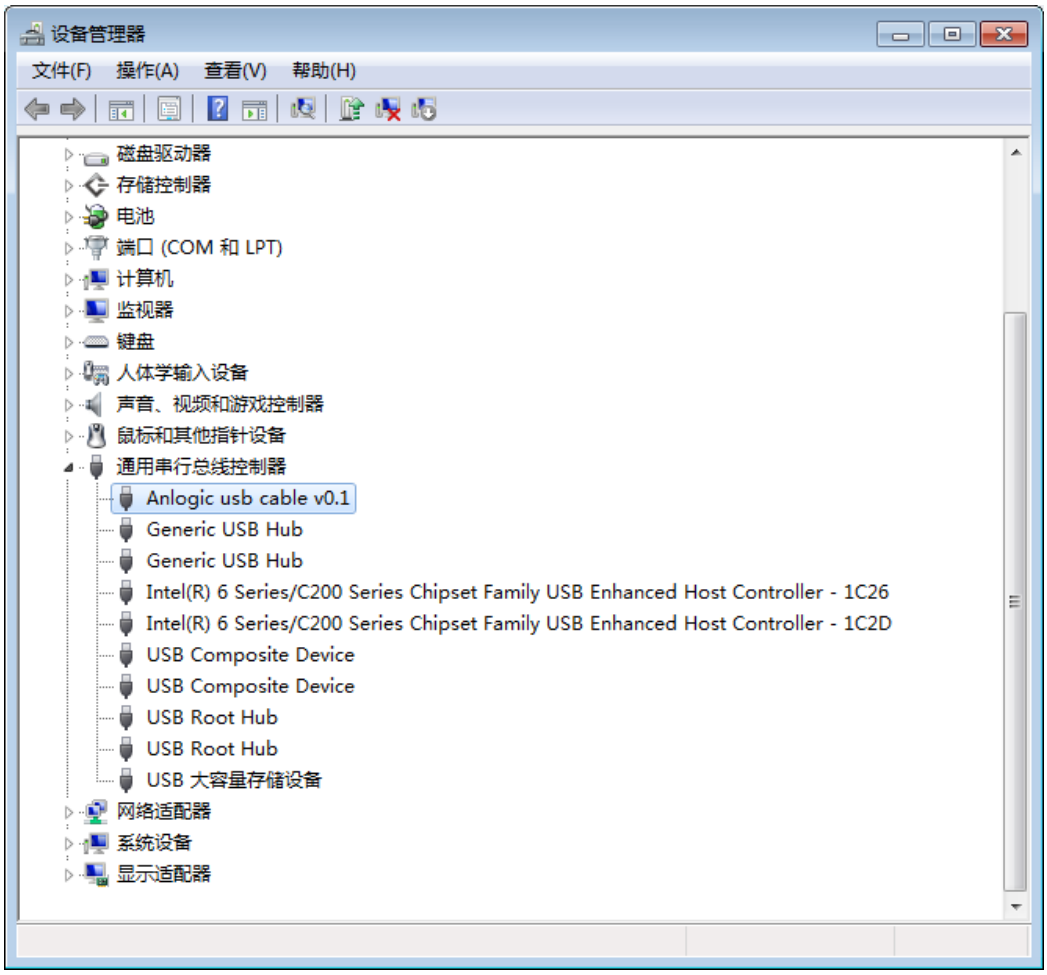
For win8 or win10 64-bit systems, choose win8_10_64.

There will be a warning during installation, the driver is not certified, choose to continue to install



4. After the installation is successful, open the device manager, expand the "Universal Serial Bus Controller", you can see "Anlogic usb cable v0.1", and there is no exclamation point, the USB driver is successfully installed.





5. After the driver is installed, restart your computer.