in the Summary.

2. Created Generated Clock

Format: **create_generated_clock** -add -name <string> -source <list> -divide_by

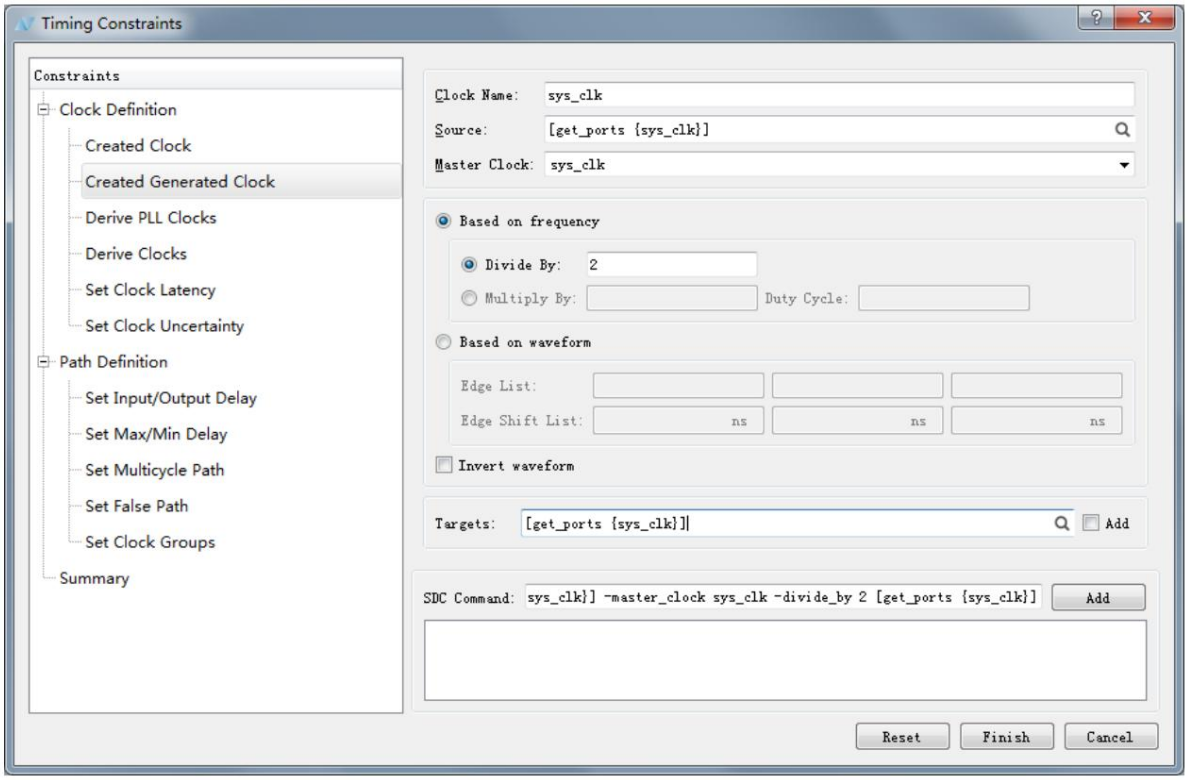
<double> -multiply_by <double> -edges <string> -duty_cycle <double> -invert -

edge_shift <string> -master_clock <string> target

Definition: Define a derived clock that belongs to the clock domain where the master clock is located, and will also be included in the timing report

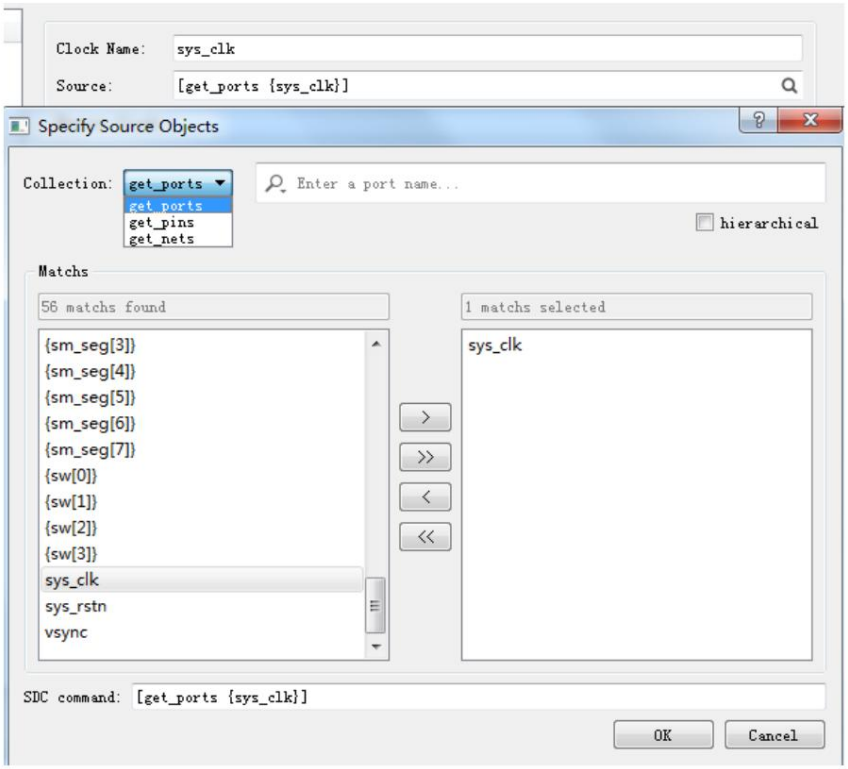
Has the same start point as the master clock.

The parameter settings are shown in the following figure:



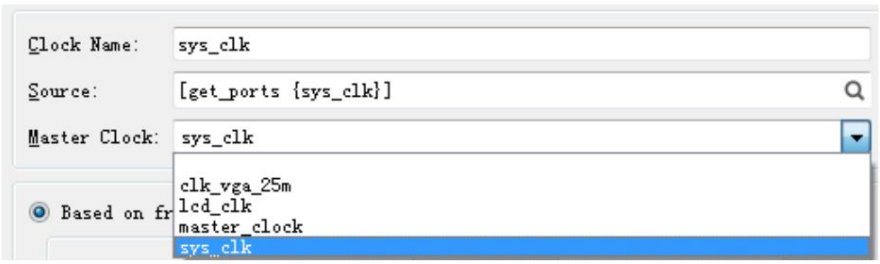

-source specifies the source point where its master clock is located, which can be a list of nets, pins or ports;

Click the Find icon behind the Source column to select different types of sources:




-master_clock specifies the name of the master clock, click the drop-down menu of Master Clock, optional

Select the created clock;

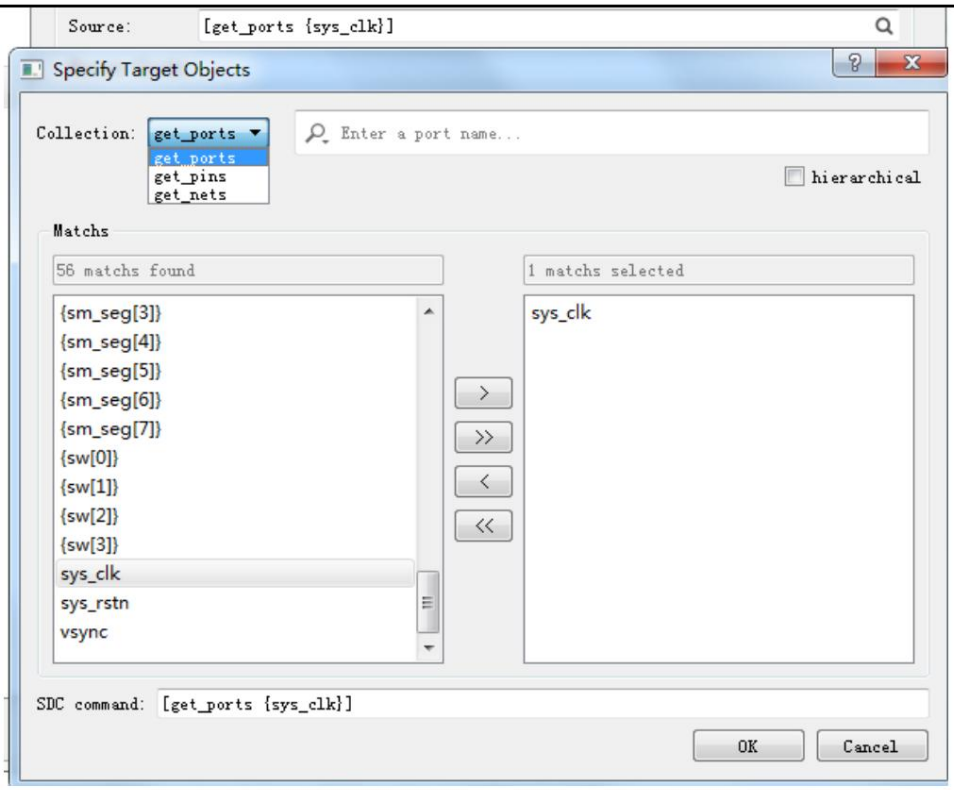



-name specifies the clock name, if the item is empty, the clock name is the first item in the source list; target is

The source point of the currently generated clock. -add indicates that if the clock has been defined on the target pin, add the new

The clock is added to this 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 column.




The cycle and waveform of the generated clock are adjusted by the master clock, -divide_by / -multiply_by refers to the frequency

Divide/multiply by the specified multiple, -invert works with these two options to invert the clock waveform, while

-duty_cycle is used with the -multiply_by option to adjust the duty cycle; Based on is selected by default

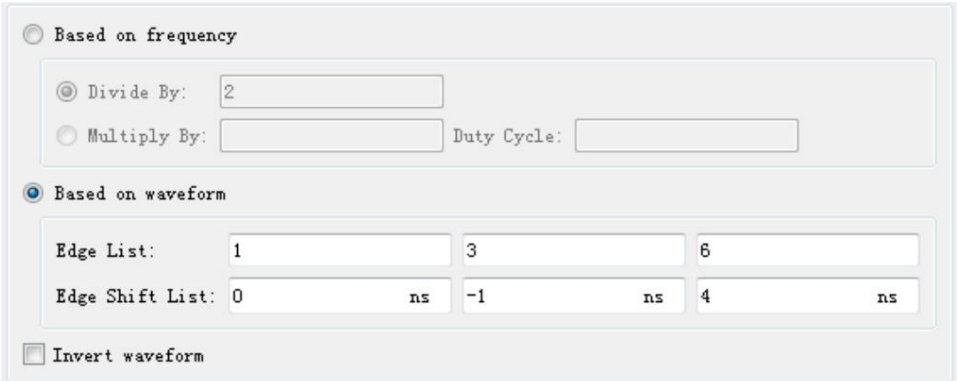
frequency.

If you select Based on waveform, you can set the edge option. The -edges option contains three integers, cents

Do not specify the first rising edge of the newly generated clock, the first falling edge, and the second rising edge corresponding to the source clock

the first edge of the clock; the -edge_shift option will specify the offset of the three clock edges in the -edges option, so

Will contain 3 positive or negative integers in base time units (defaults to ns).

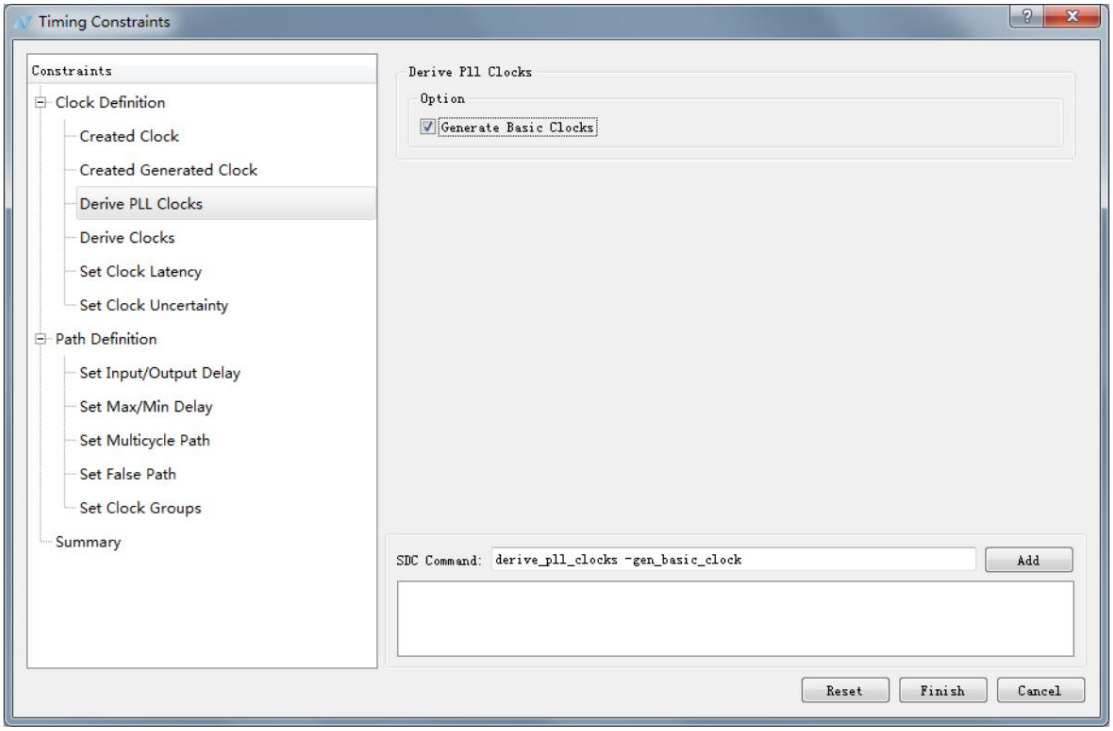


3. Derive PLL Clocks

**Format: derive_pll_clocks** [-gen_basic_clock]

Definition: Automatically generate clock constraints on all used PLL clkc[x] ports, generate clock frequency, phase

The bits will be set strictly according to the parameters inside the PLL.




-gen_basic_clock will define the base clock of the FIN frequency on the corresponding PLL refclk, otherwise it will automatically

Automatically searches for the refclk pin and the clock defined on the connected net. The clock generated by this command will run in the flow

It will take effect only when it takes effect, so it cannot be referenced in the subsequent settings of the timing wizard.

4. Derive Clocks

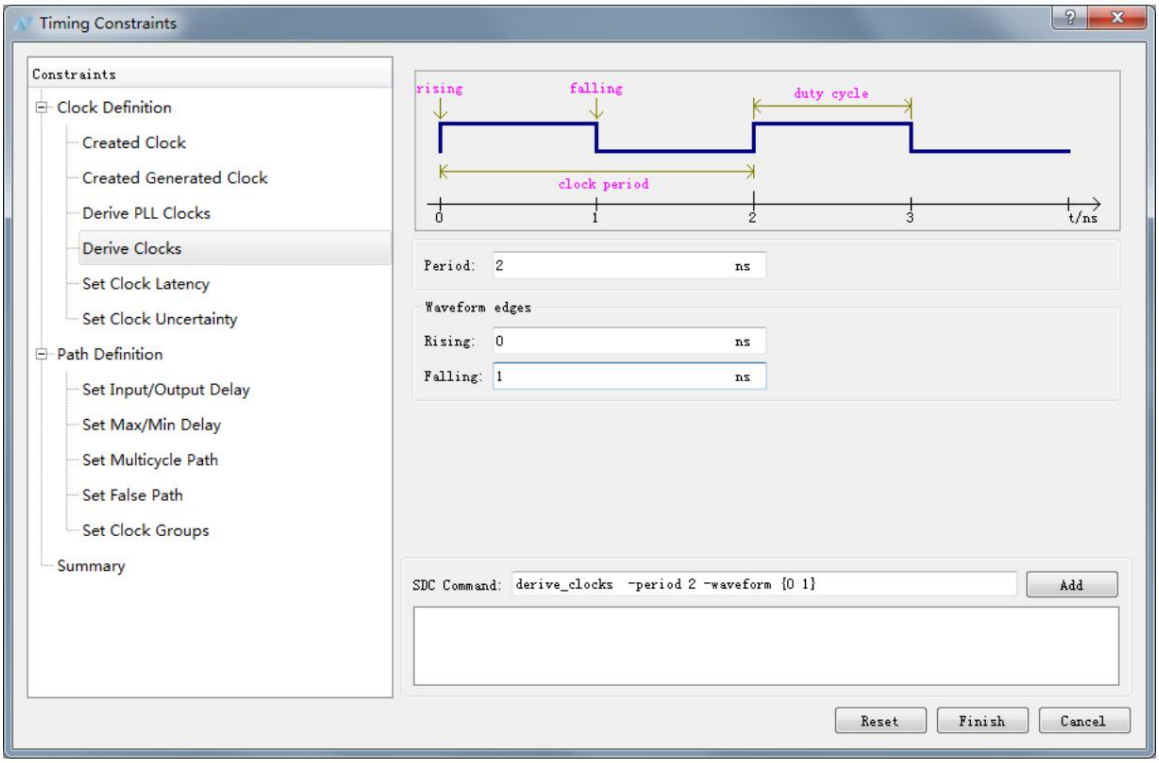
Format: derive_clocks -period <double> -waveform <string>

Definition: Specify a default clock on the clock pins of each undefined clock, -period is the period, the option

The item must be specified, and the value must be greater than 0; -waveform specifies the first rising edge and the first falling edge

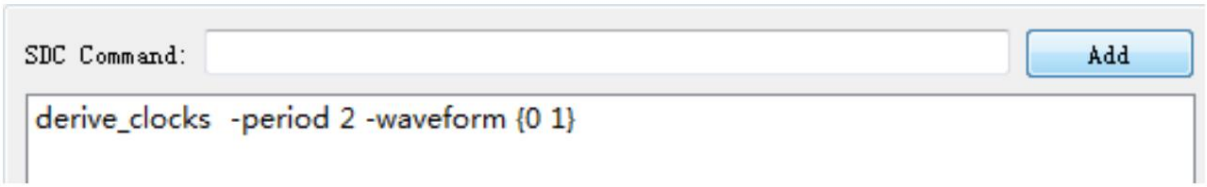
At the time point, only the case where each cycle contains two clock edges is temporarily supported.

The parameter settings are shown in the following figure:




Click Add to add the command to the box below, you can continue to set other parameters of the command, otherwise

This 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;

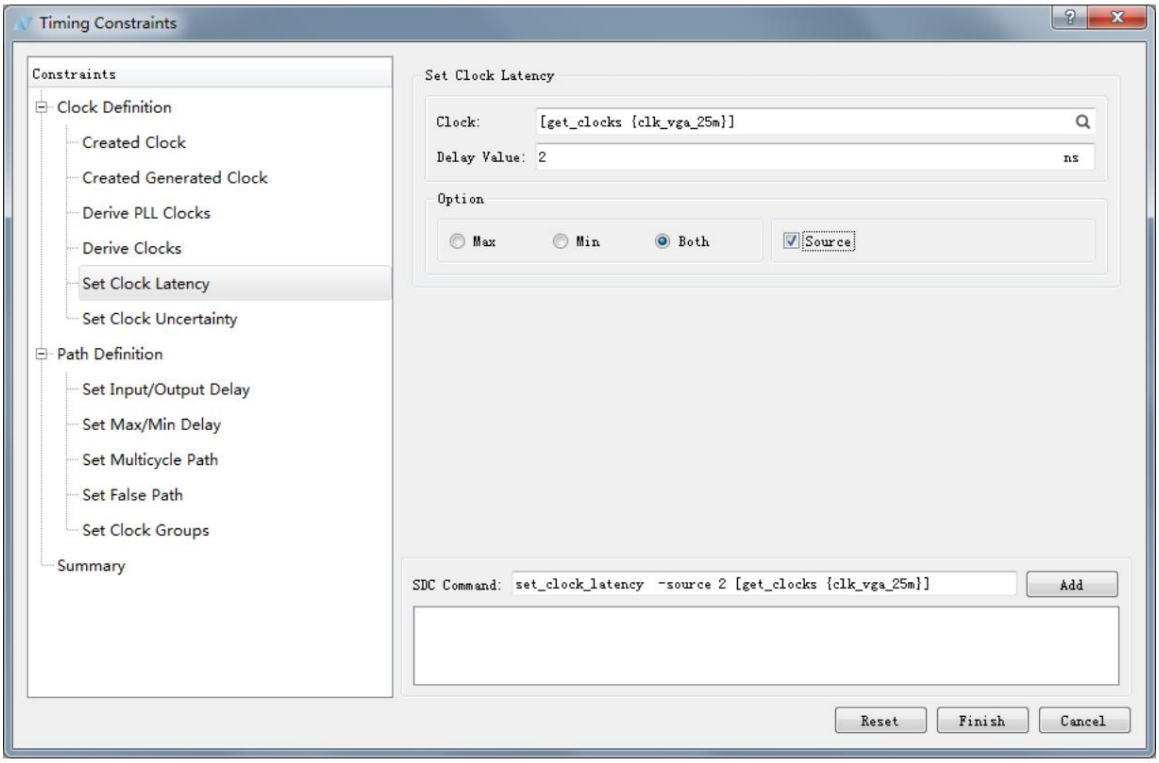
The -max/ -min option specifies the maximum/minimum delay specified by this command, and the default is the same for both;

The -source option specifies that the source latency is specified, otherwise it is the network latency.

The network latency is replaced by the actual interconnect latency in post-routing timing analysis.

Click the Find button behind the Clock column to specify a clock that has been created.

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 uncertainty value of the clock, currently only supports setting for a single clock itself but not

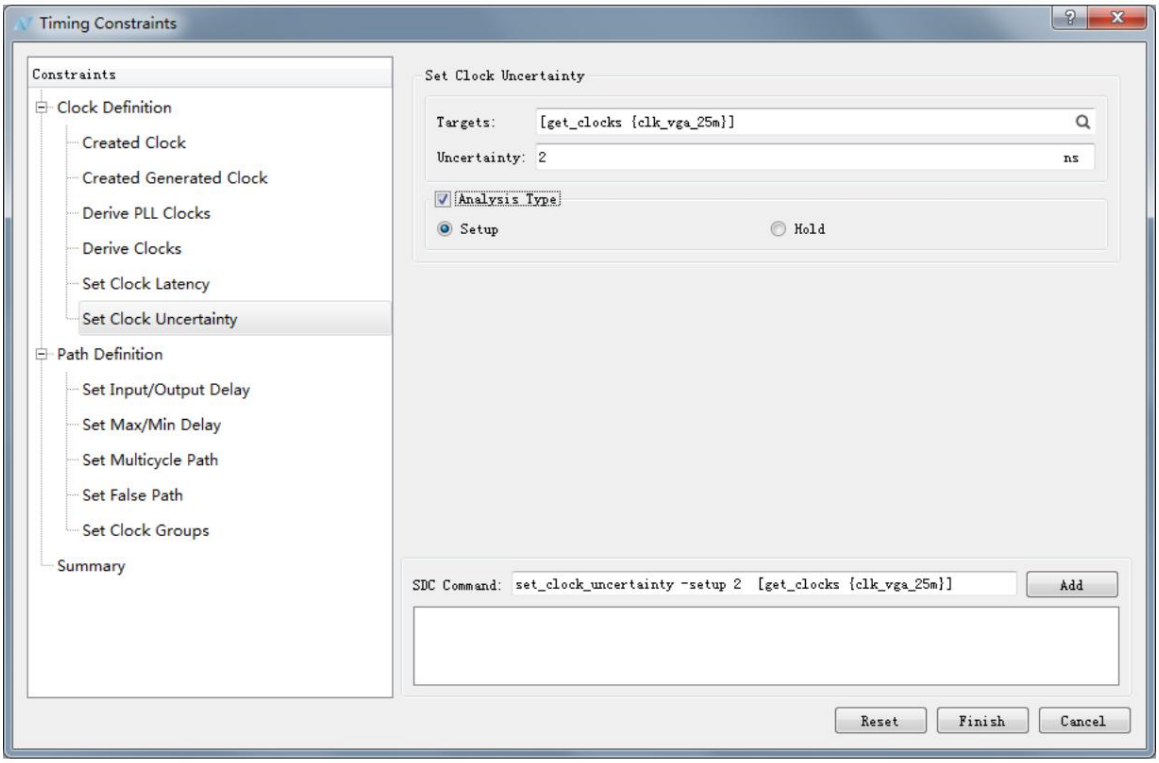
Holds the definition of uncertainty across clock domains; target is a list of clocks, and uncertainty is a numeric item; -

The setup/-hold option indicates that this command corresponds to the value corresponding to the maximum/minimum path timing analysis.

If this option is not specified, both checks are in effect.

Click Add to add the command to the Summary.

The parameter settings are shown in the following figure:




7. Set Input/Output Delay

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

target

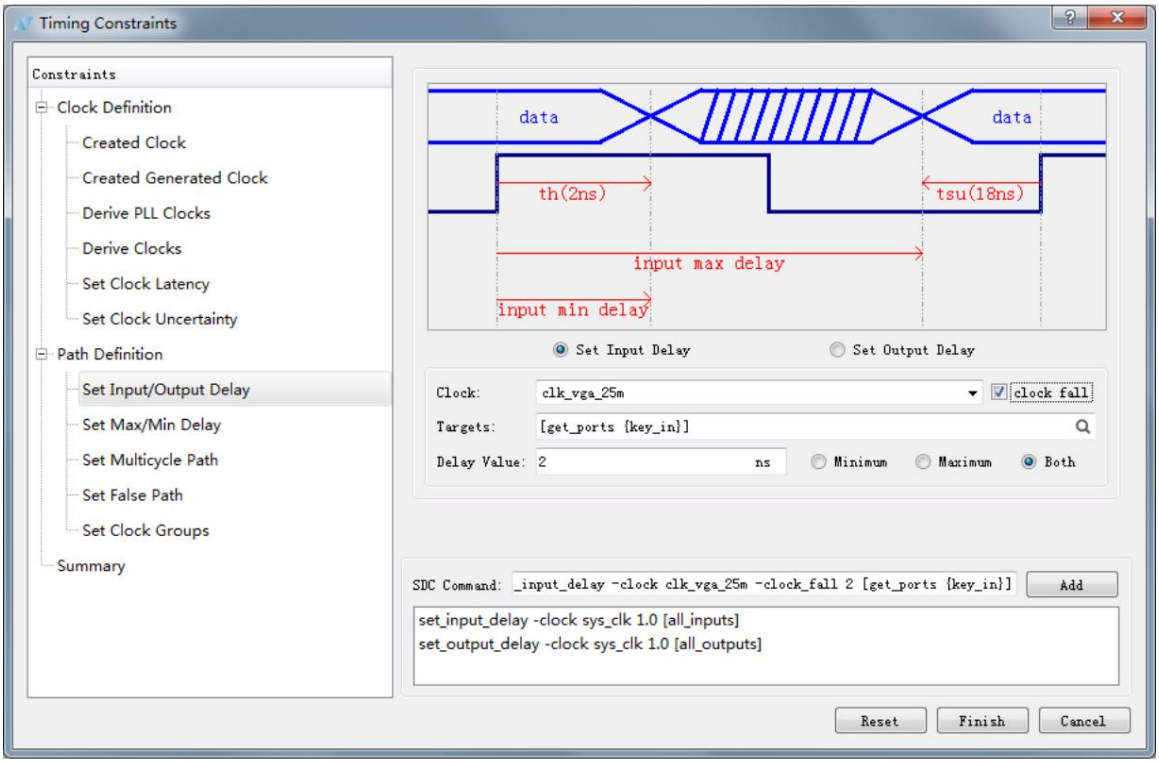
Definition: Set the delay of the input/output ports, the delay item is the delay value, and the target can only be the pair of ports

Like list; -clock specifies the clock information corresponding to the delay; -clock_fall specifies that the reference clock is in

Falling edge sampling; the -max/-min option specifies the maximum/minimum delay value set by this command, the default

for the same.

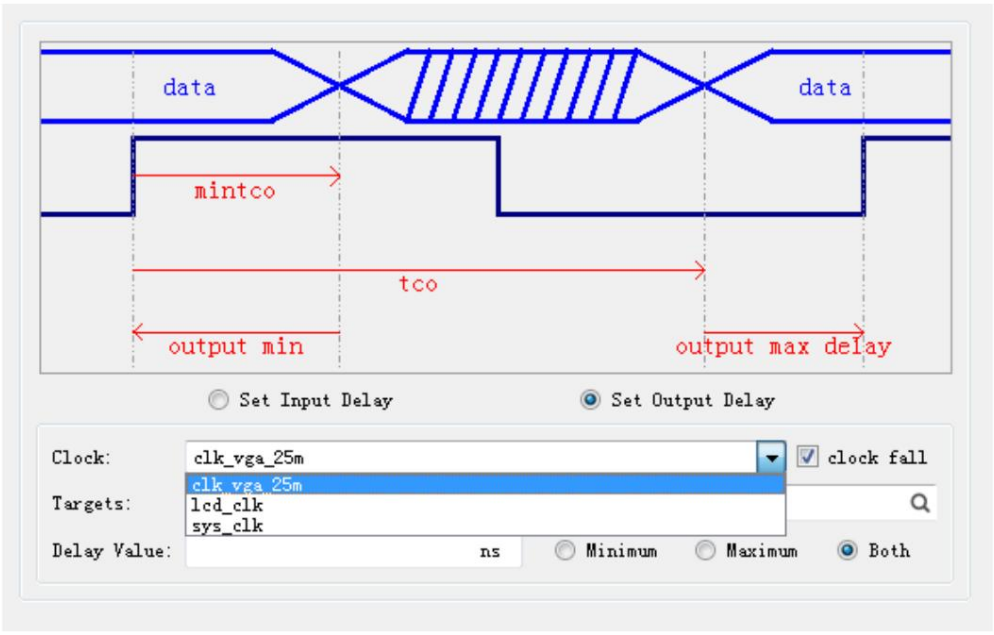
The parameter settings are shown in the following figure:




The default is to set input delay related parameters;

If you need to set the output delay related parameters, you need to select Set Output Delay, the parameter settings are the same as the input delay.

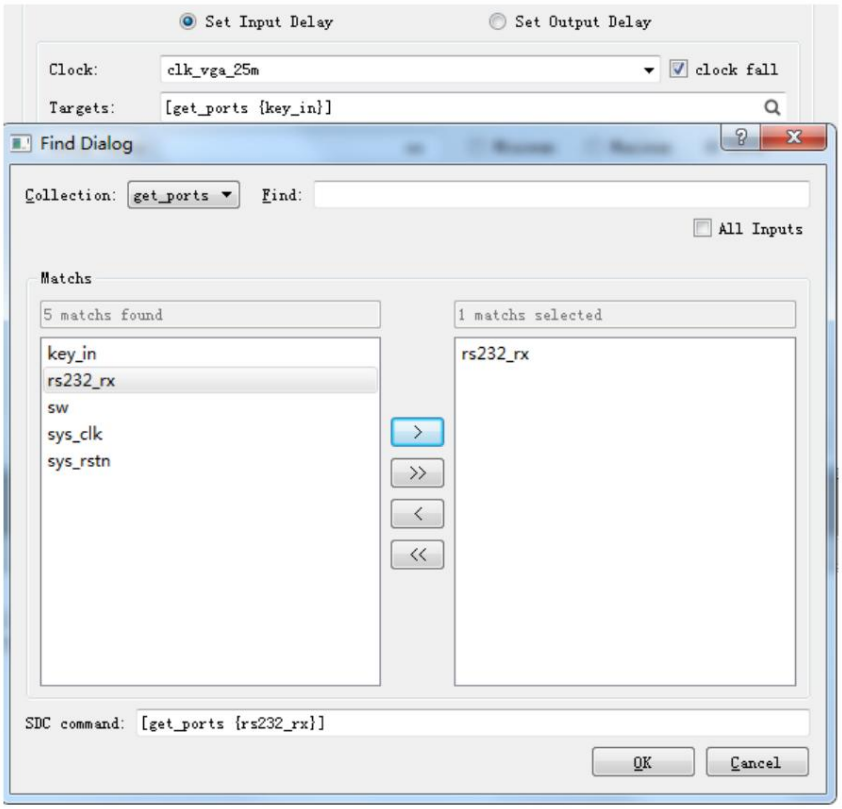
Click the drop-down menu in the Clock column to specify the clock that has been created.

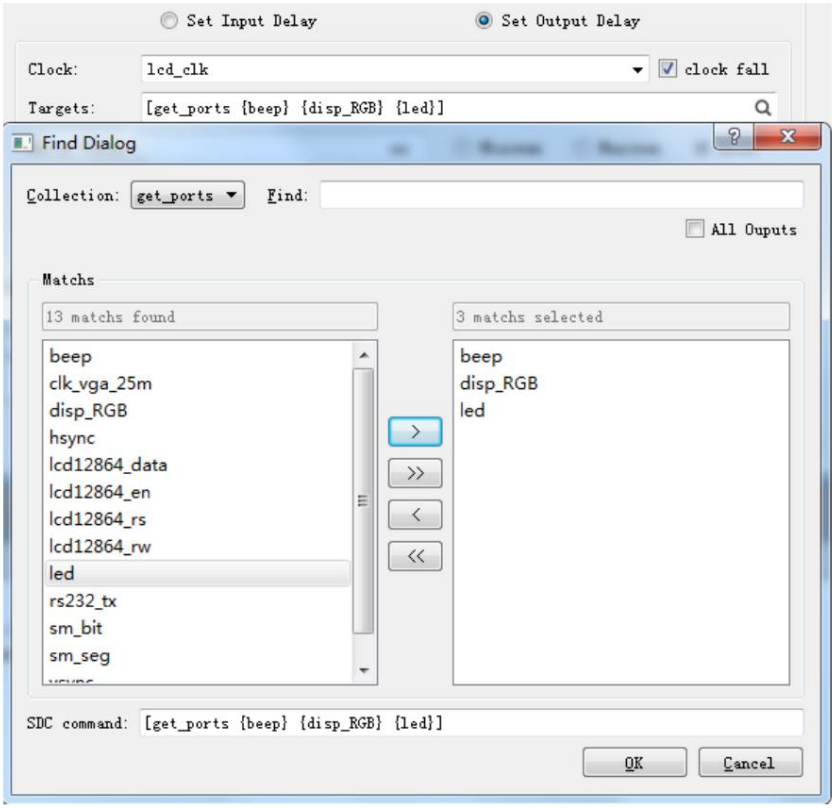



Click the Find button in the Targets column to specify the corresponding ports. If Set Input Delay is selected,

Only input/inout ports are visible; if Set Output Delay is selected, only output/inout

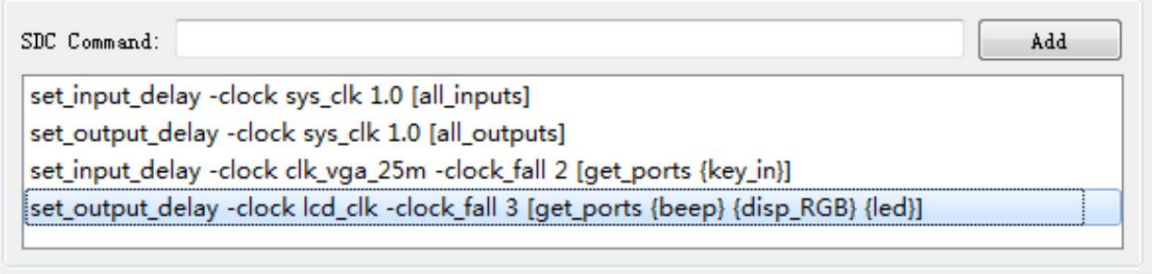
ports are visible.




If clock_fall is checked, it is sampled on the falling edge of the selected clock.

Click Add to add the command to the Summary.




To delete the generated SDC Command, select the command in the Summary, click the end of the line

the delete button "X".

8. Set Max/Min Delay

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

Definition: Set the allowable maximum/minimum delay of the timing path, the delay item is the delay value; -from must be

The starting point of the timing path, that is, the input list, or regs, or clocks, or the corresponding position defined by clock

The pins; -to must be the end point of the timing path, that is, a list of outputs, or regs, or clocks, or

The pins corresponding to the position defined by the clock; the -through option can be nets, a list of pins, specifying the time

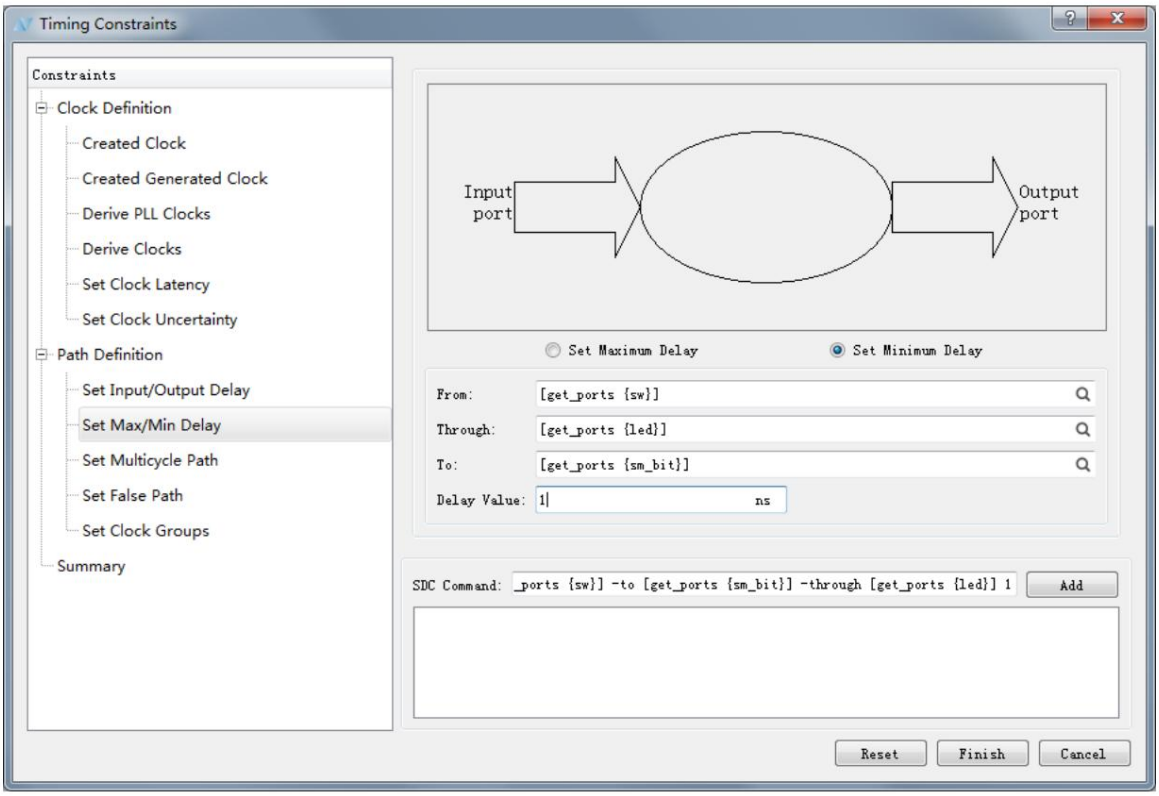
The intermediate point that the sequence path must pass through. When there are multiple through options, the target sequence path must pass through in sequence.

through every intermediate point.

The parameter settings are shown in the following figure:

The default is Set Maximum Dealy. To set Minimum Delay, select Set Minmum

Delay, the parameter settings of the two are the same.




Click Add to add the configured 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 cycles to delay, and the multiplier item is the number of clock cycles;-

The setup/-hold option sets the timing path type constrained by this command. When only the -setup option is used, setup

check will allow timing paths to use up to N clock cycles, but at the same time hold check will become required

The sequence path is the shortest through N-1 clock cycles. When only the -hold option is used, the constraints of the hold check are set to

Set to N without affecting the constraints of setup check.

In normal usage scenarios, in order to allow the setup check to use multiple clock cycles without affecting the hold check,

Two commands are required to be used together, that is, to set a setup constraint of N cycles, and then cooperate with a N-1

Period hold constraint.

-start/-end is the option used when crossing the clock domain, and the specified delay is the week corresponding to the launch/capture clock

Period, by default, the setup check uses the capture clock and the hold check uses the launch clock.

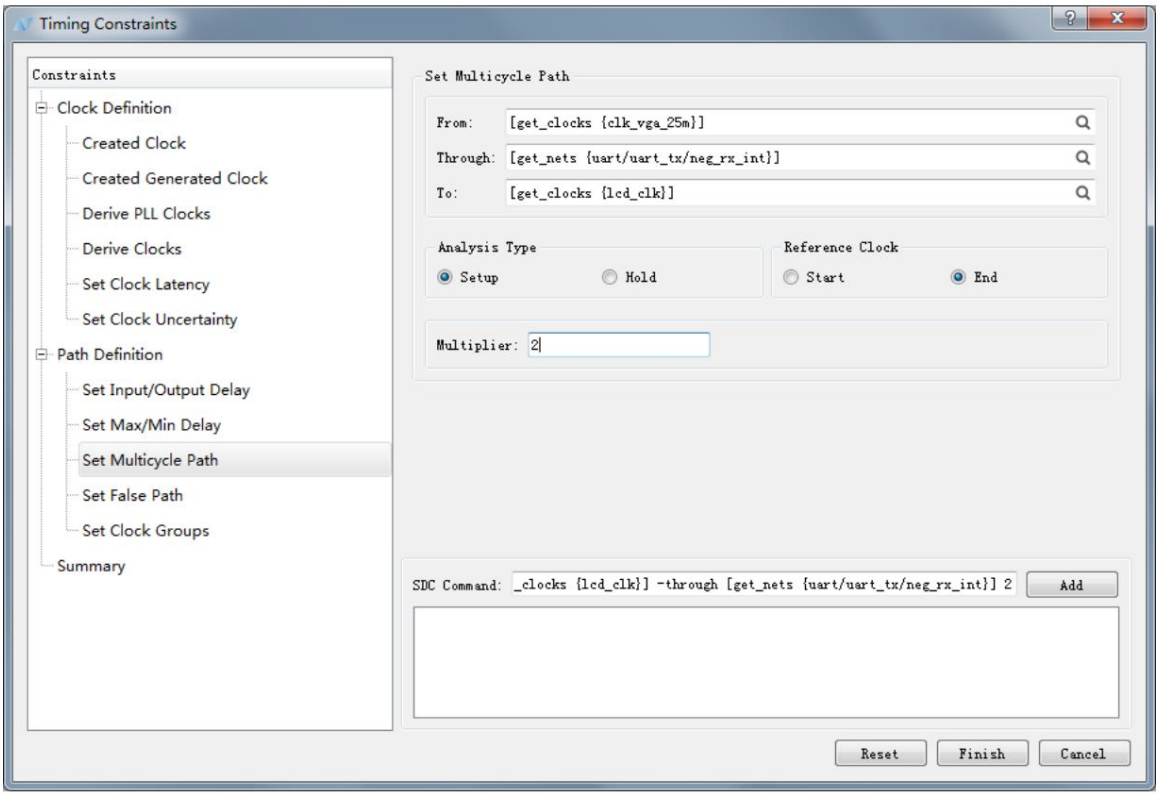
-from specifies the starting point of the constraint, which can be a list of clocks or inputs, regs, pins, etc., -to specifies

The endpoint of the constraint, which can be a list of clocks or outputs, regs, pins, etc., -through specified

The intermediate point that this timing path must pass through, but is a list of pins or nets. After specifying, the system will automatically

Trace back to the beginning and end of the timing path.

The parameter settings are shown in the following figure:




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: Sets the timing path as a false path, so it is not subject to timing analysis; -setup/-hold option

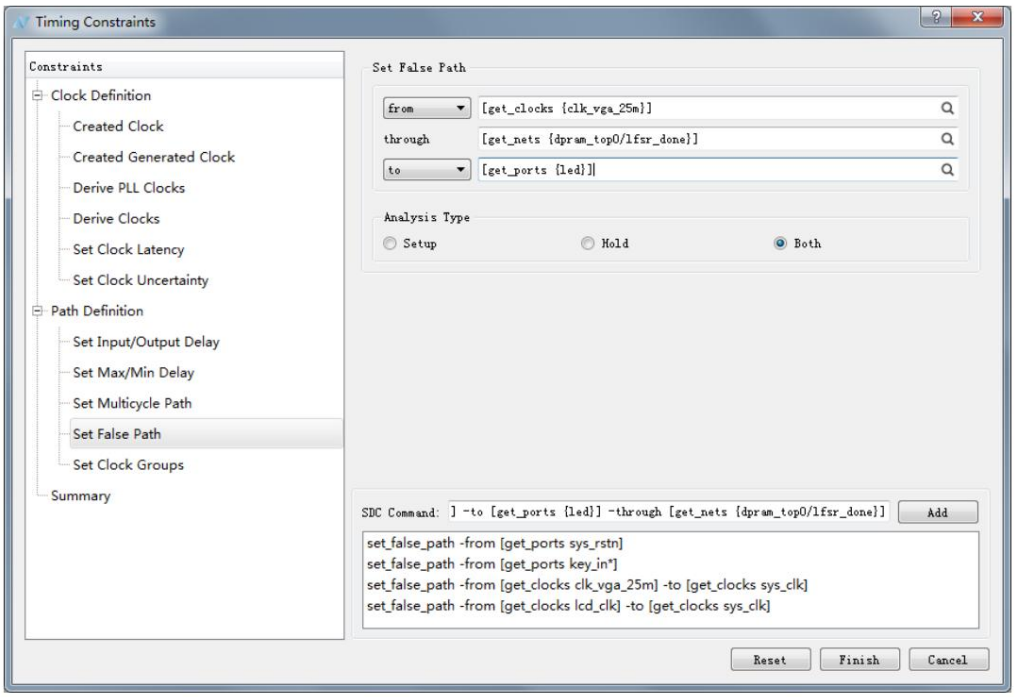
Specifies not to analyze target paths during setup/hold check. -from specifies the starting point of the constraint, which can be

List of clocks or inputs, regs, pins, etc., -to specifies the endpoint of the constraint, which can be clocks or

A list of outputs, regs, pins, etc., -through specifies the intermediate point that the timing path must pass through,

But it is a list of pins or nets. After specifying, the system will automatically trace back to the start and end points of the timing path.

The parameter settings are shown in the following figure:




Click Add to add the generated command to the Summary.

11. Set Clock Groups

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

Definition: Sets groups for clock domains and does not analyze timing paths across clock groups. In general, -exclusive

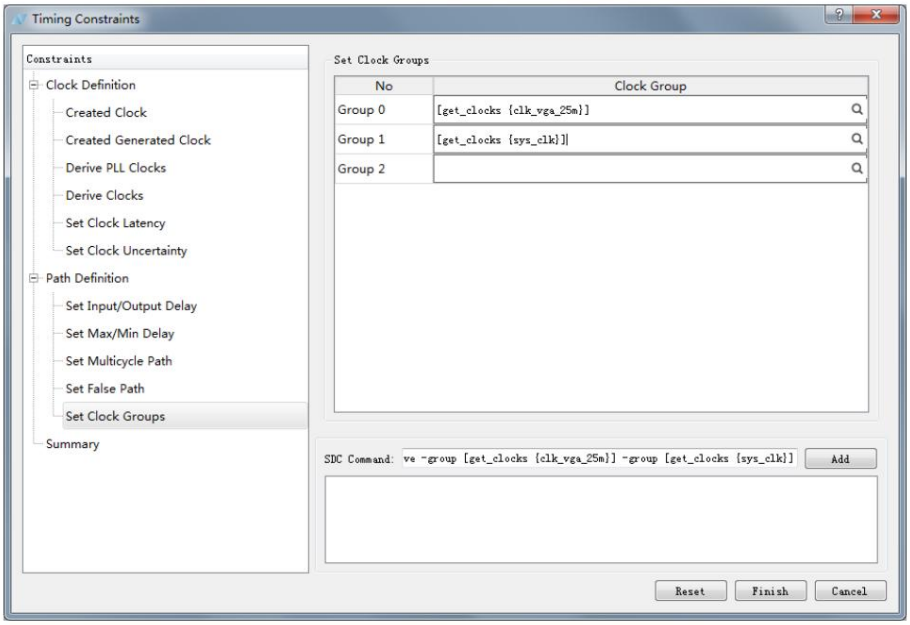
The option means that these groups of clocks are not logically concurrent, while -asynchronous means completely unrelated

clock, but in terms of specific implementation and effect, these two options are the same, this command will

A set of false path constraints are defined between clocks listed in group.

The parameter settings are shown in the following figure:

Click Add to add the generated command to the Summary.



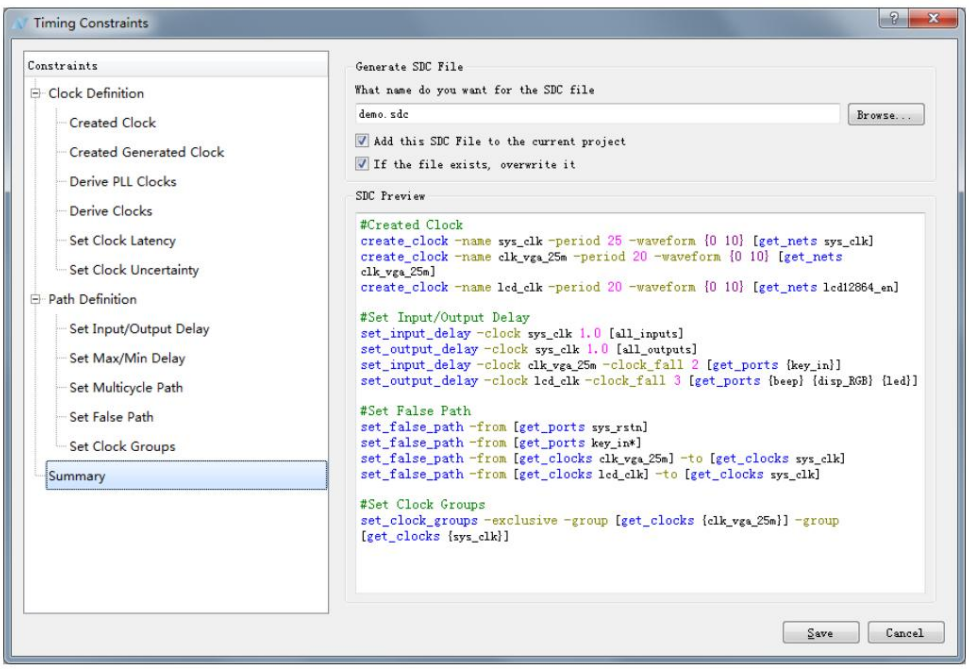

12. Summary

All generated commands are recorded in the Summary.

Specify the path to save the file, and choose whether to add it to the current project. Click Save to complete the operation of each command.

set up. Note that if the sdc file has been added to the original project, please carefully check the "If the file exist, overwrite

it". After this option is checked, the original file will be replaced.



# 5 HDL2Bit process

After inputting the design source file and constraint file, the next step is to enter the design implementation flow of HDL2Bit. HDL2Bit

The process includes design read-in (Read **Design), RTL -level optimization (Optimize** RTL), gate-level optimization (Optimize

Gate), Optimize Placement, Optimize **Routing, and Generate Bitstream**

**(Generate Bitstream) Six steps.**

In most cases, users only need to double-click HDL2Bit, and the software will automatically run the entire process. Users can also use

**Run, Rerun, Stop** in the Process drop-down menu to control. **Run** and **Stop** in the Process menu are navigating

There are corresponding buttons ( and ) in the bar, and the user can directly click to operate. There is also a **Properties** column in the Process menu

For example , **Properties** provides detailed parameter control options for the main steps in the **HDL2Bit** process, and users can control the entire operation.

The process is fine-tuned and controlled, and the parameters of Global Option are set as follows:

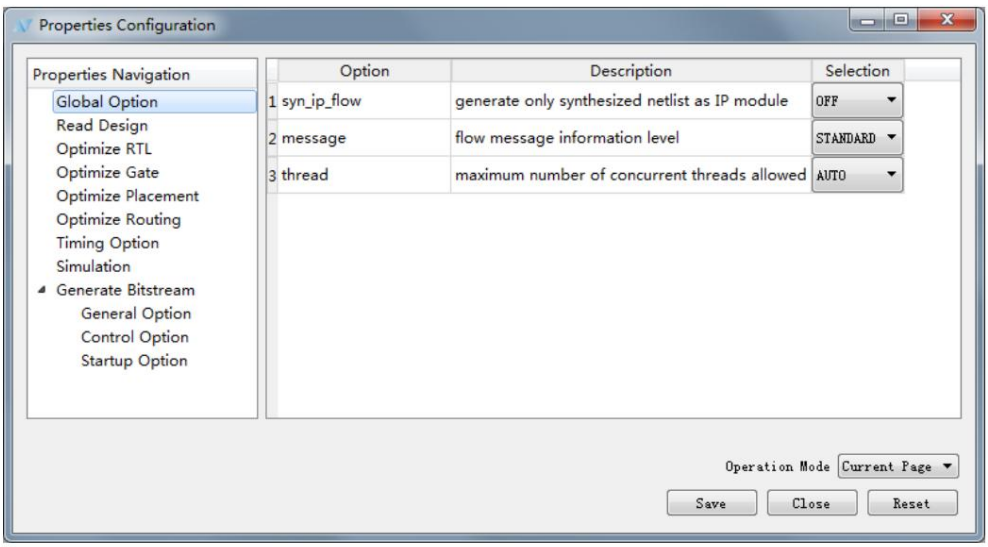



Table 5-1 Global Option

| Property | Comments | Default |
|---|---|---|
| syn_ip_flow | Generate synthesizable IP blocks | 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 and semantics of the user source file, and generates the original behavior-level circuit structure.

1. Expand **HDL2Bit** in the FPGA Flow panel

2. Double-click **Read** Design, or right-click **Read Design and select run**

3. Parameter configuration

In the menu bar, expand **ProcessÿProperties, the** Properties Configuration window pops up, select

**Read** Design, users can customize the functions they want according to their needs.

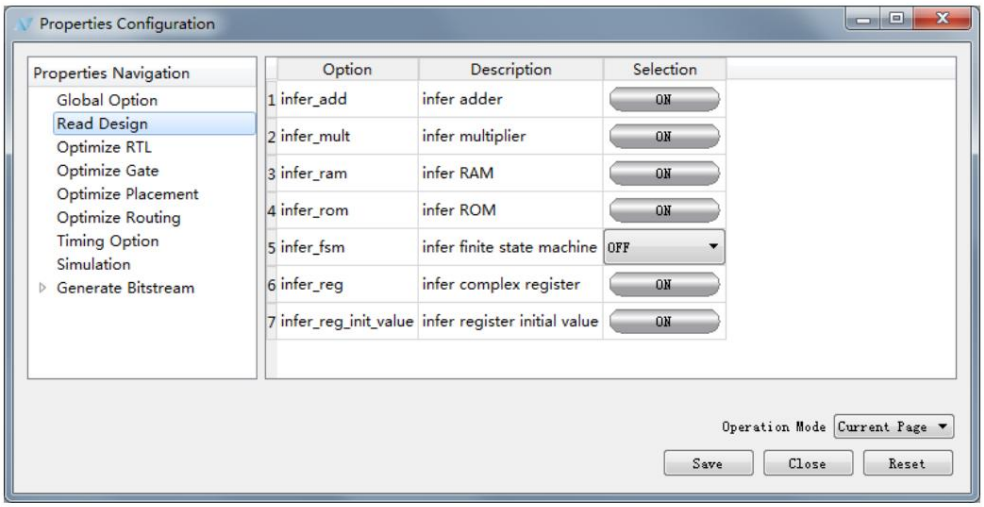



Table 5-1 Read Design Properties

| Property | Comments | Default |
|---|---|---|
| infer_add | Automatic recognition of behavior-level additive descriptions | ON |
| infer_mult | Automatic identification of behavioral-level multiplication descriptions | ON |
| infer_ram | Automatic recognition of behavioral RAM descriptions | ON |
| infer_rom | Automatically recognize behavior-level ROM descriptions | ON |
| infer_fsm | Automatically identify behavioral-level finite state machine descriptions | OFF |
| infer_reg | Automatic recognition of behavior-level complex register descriptions | ON |
| infer_reg_init_value | Automatically identify behavior-level register initial value | ON |

## 5.2 RTL -level optimization

After reading in the design file, TD will optimize the design at the RTL level. This stage will perform multiplexer optimization, Data path optimization, automatic identification of special function modules, etc. RTL-level optimization will result in inclusion of elementary gates (AND, OR, FF/Latch) and special function modules.

1. Expand **HDL2Bit** in the FPGA Flow panel

2. Double-click **Optimize RTL,** or right-click **Optimize RTL,** select Run, and the area will be generated

   Report the file rtl.area, the content of this file will be related to the setting of the keep_hierachy parameter.

3. Parameter configuration

   In the menu bar, expand **ProcessÿProperties, the** Properties Configuration window pops up, select

   Select **Optimize RTL** to set.

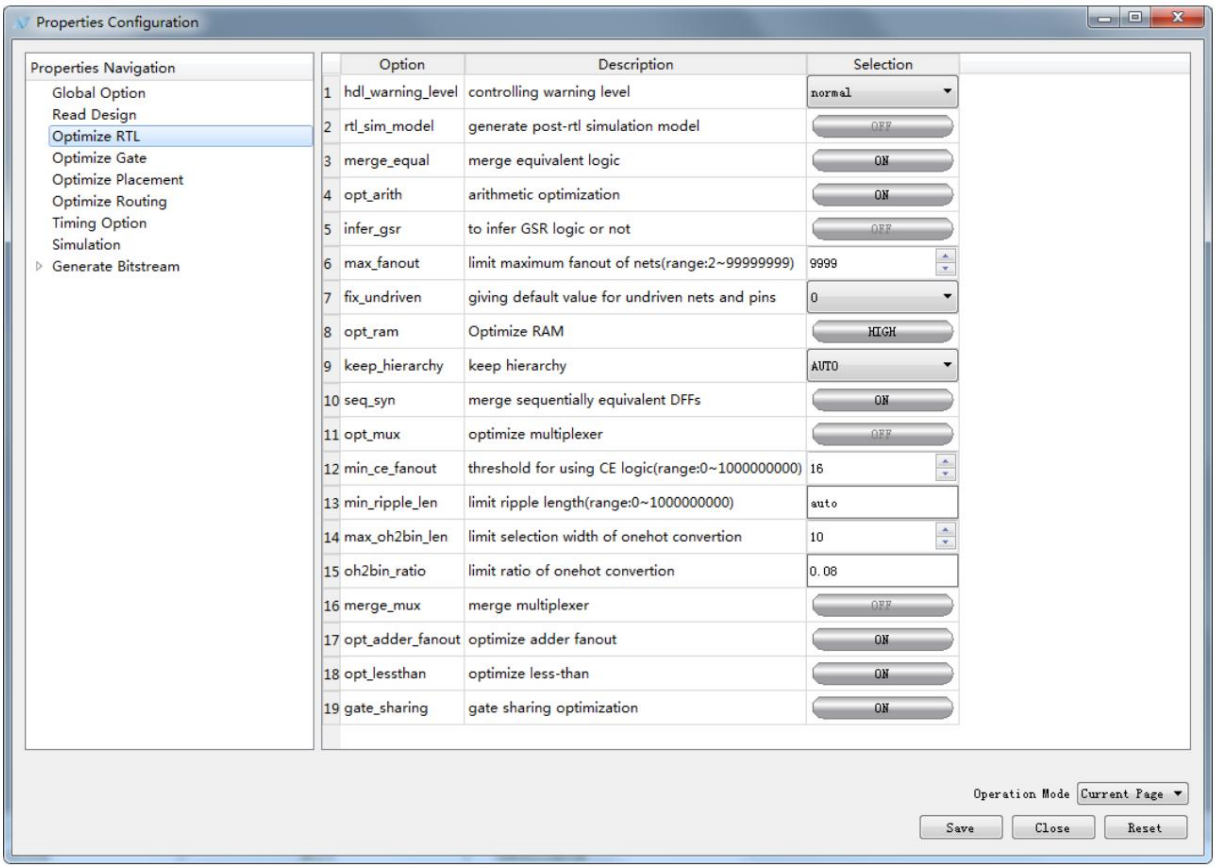



Table 5-2 Optimize RTL Properties

| Property | Comment | Default |
|---|---|---|
| hdl_warning_level | Control warning level | normal |
| rtl_sim_model | Generate RTL-level circuit simulation models | OFF |
| merge_equal | Merge functionally equivalent logic modules | ON |
| opt_arith | Arithmetic optimization | ON |
| infer_gsr | Global gsr optimization | OFF |
| max_fanout | Limit the maximum number of fanouts for logic nets | 9999 |
| fix_undriven | For nets or ports that are not assigned by the user, give a constant value of 0/1 | 0 |
| opt_ram | Loop-initialized multi-array optimization into DRAM | HIGH |
| keep_hierarchy | Preserve level AUTO for modules selected by user via synthesis directive | |
| seq_syn | Merge Equivalent DFFs | ON |
| opt_mux | Optimized multiplexer | OFF |
| min_ce_fanout | Thresholds using CE logic | 16 |
| min_ripple_len | Minimum length implemented with ripple | auto |
| max_oh2bin | One-hot conversion to BinaryMux's maximum selection port width 10 | |
| oh2bin_ratio | One-hot conversion to data-side sparsity of BinaryMux | 0.08 |
| merge_mux | Combine cascaded Muxes of the same selected port | OFF |
| opt_adder_fanout | Optimizing the fan-out of addition and subtraction | ON |
| opt_lessthan | Optimized Comparator | ON |
| gate_sharing | Extract common gate logic | ON |

*keep_hierarchy option description: when flatten is selected, td will flatten the user design; when manual is selected

, td will retain the hierarchy for the module selected by the user through the synthesis directive; when auto is selected, td will automatically

Automatically select a larger module to retain the level, and the rest are leveled.

## 5.2.1 Synthesis Keep

Using the Synthesis keep command can ensure that the signal will not be optimized by the subsequent process, which is convenient for users to adjust later.

try. The specific methods are as follows:

1. In the verilog file, add a corresponding 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 can be viewed when using the debug tool.

4. The writing 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 keep.

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

Law.

The module ef2_ram in the figure below is completely equivalently instantiated many times in the top module.

Using the function of synthesis keep, all instances will be merged into one BRAM. to keep all

The instance is not optimized, and the annotation "//synthesis keep" needs to be added to each level of the instance.

test_keep.v*

```
1
2  module test_keep(
3      input clk,
4      input [9:0] addr,
5      input di,
6      output [7:0] do );
7
8  genvar x;
9
10 generate
11     for (x=0; x<8; x=x+1)
12         begin : ram
13             ef2_ram inst  //synthesis keep
14                         (.dia(di),
15                          .addra(addr),
16                          .wea(1'b1),
17                          .cea(1'b1),
18                          .clka(clk),
19                          .rsta(1'b0),
20                          .doa(do[x]));
21         end
22 endgenerate
23
24
25 endmodule
26
```

ram.v

```
13
14 module ef2_ram ( doa, dia, addra, cea, clka, wea, rsta );
15
16     output [0:0] doa;
17
18     input  [0:0] dia;
19     input  [9:0] addra;
20     input  wea;
21     input  cea;
22     input  clka;
23     input  rsta;
24
25     EF2_LOGIC_BRAM #( .DATA_WIDTH_A(1),
26                  .ADDR_WIDTH_A(10),
27                  .DATA_DEPTH_A(1024),
28                  .DATA_WIDTH_B(1),
29                  .ADDR_WIDTH_B(10),
30                  .DATA_DEPTH_B(1024),
31                  .MODE("SP"),
32                  .REGMODE_A("NOREG"),
33                  .WRITEMODE_A("NORMAL"),
34                  .RESETMODE("SYNC"),
35                  .IMPLEMENT("9K"),
36                  .DEBUGGABLE("NO"),
37                  .PACKABLE("NO"),
38                  .INIT_FILE("NONE"),
39                  .FILL_ALL("NONE"))
40             inst(                    //synthesis keep
41                  .dia(dia),
42                  .dib({1{1'b0}}),
43                  .addra(addra),
44                  .addrb({10{1'b0}}),
45                  .cea(cea),
46                  .ceb(1'b0),
47                  .ocea(1'b0),
48                  .oceb(1'b0),
49                  .clka(clka),
50                  .clkb(1'b0),
51                  .wea(wea),
52                  .web(1'b0),
53                  .bea(1'b0),
54                  .beb(1'b0),
```

## 5.2.2 Synthesis Directive

### 1. Synthesis keep_hierarchy

In the keep hierarchy auto/manual mode, the synthesis directive can be added to the HDL

to specify whether the hierarchy for a particular module is to be preserved. In Flatten mode, the synthesis directive is invalid.

The specific methods are as follows:

1) For the modules you want to keep, add corresponding comments to the source files. The comments are written as:

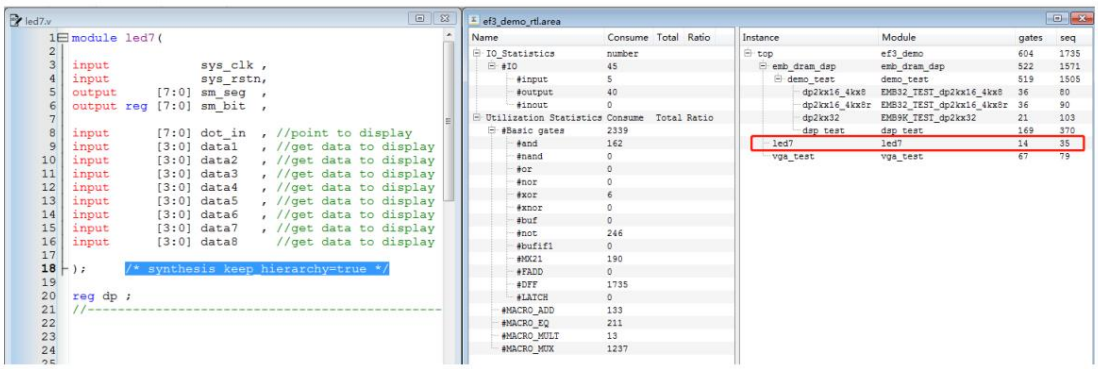
a) keep the entire module

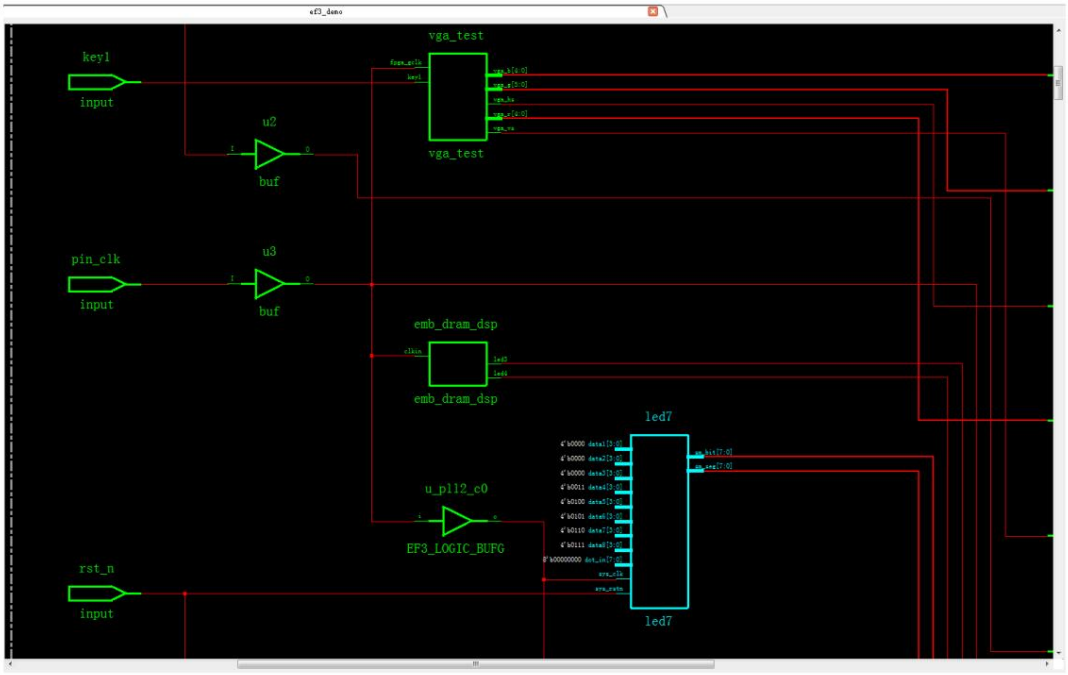
Verilog:

```
module SomeModule(
...
…
);                /* synthesis keep_hierarchy=true */
....
endmodule
```

VHDL:

```
architecture rtl of SomeModule is
    attribute keep_hierarchy : string;
    attribute keep_hierarchy of rtl : architecture is "true";
    ....
end rtl;
```





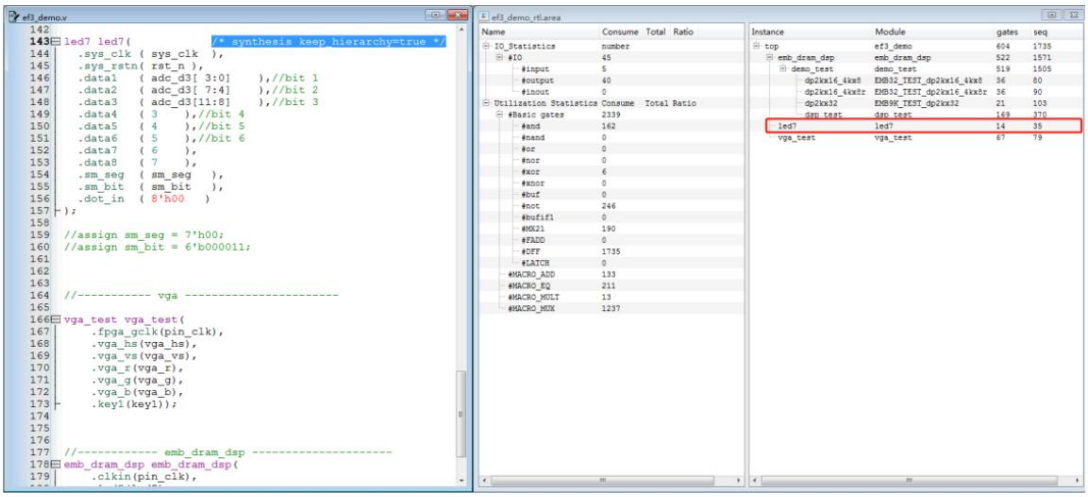

b) keep instantiated modules

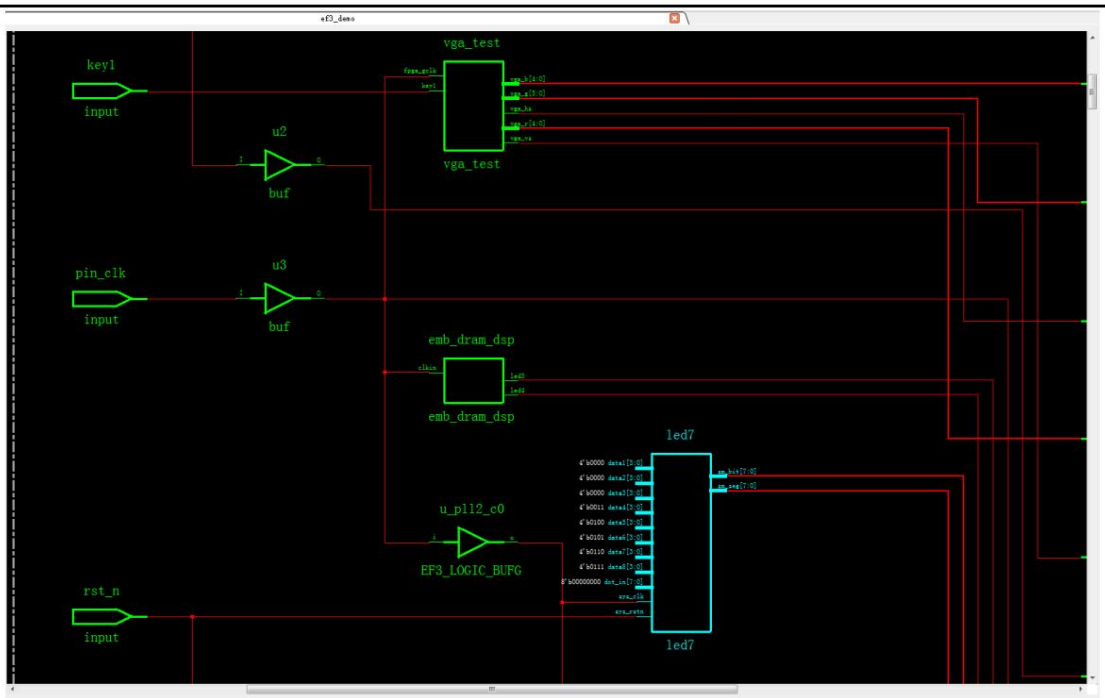
Verilog:

```
SomeModule u1(                    /* synthesis keep_hierarchy=true */
...
…
…
);
```

VHDL:

```
attribute keep_hierarchy : string;
attribute keep_hierarchy of u1 : label is "true";
```





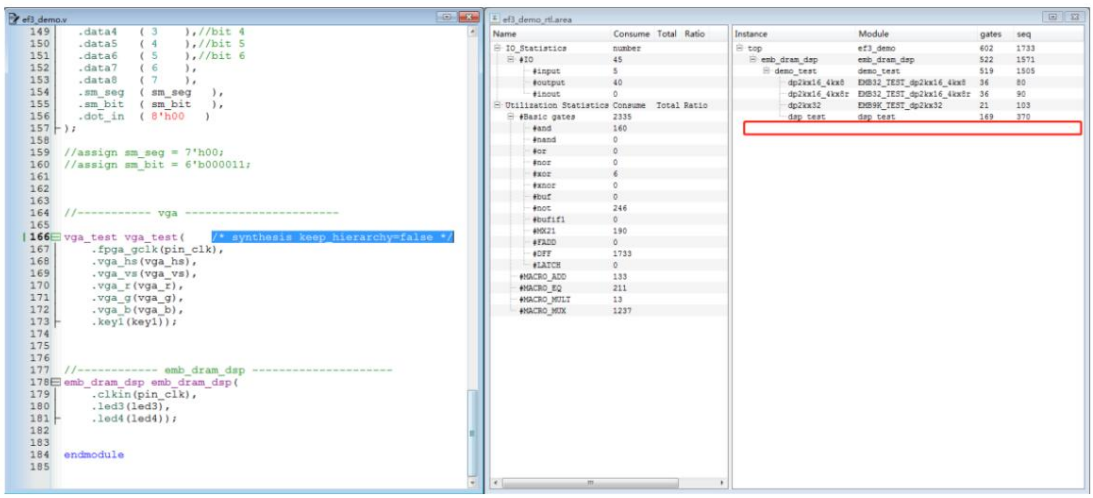

c) do not preserve instantiated modules

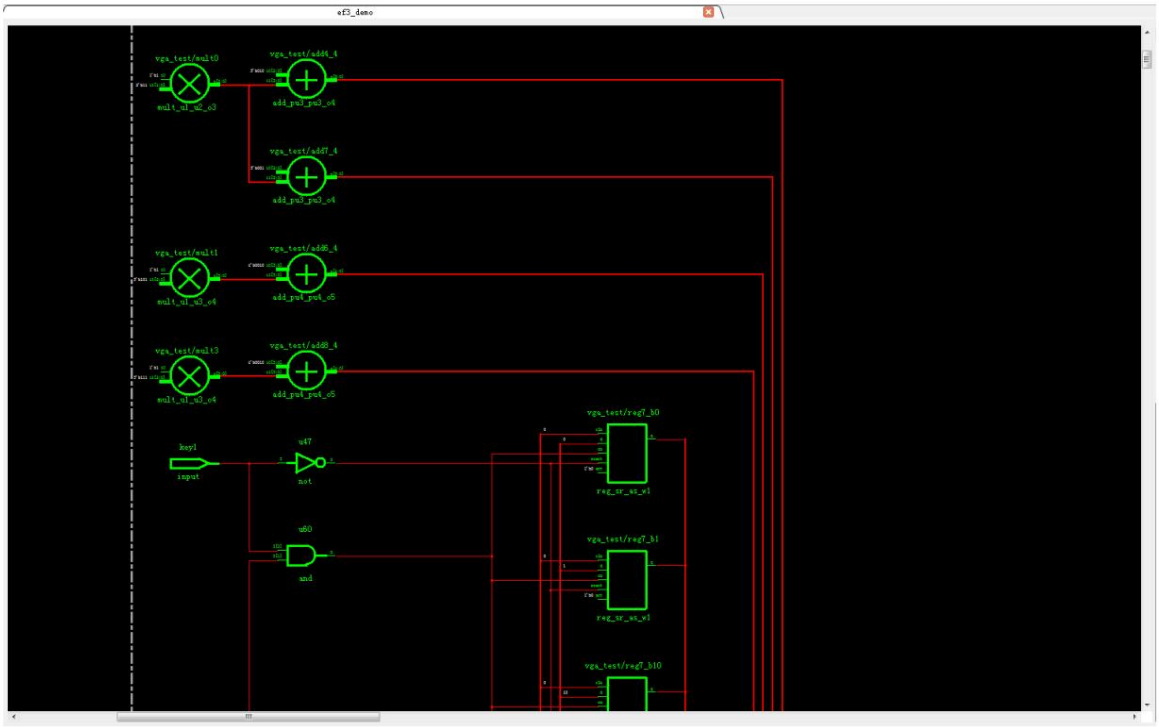
Verilog:

```
SomeModule u2(                    /* synthesis keep_hierarchy=false */
...
…
…
);
```

VHDL:

```
attribute keep_hierarchy : string;
attribute keep_hierarchy of u2 : label is "false";
```





2) Save the file and compile;

3) You can see the resource usage of the reserved modules in the RTL/Gate Summary.

The RTL Schematic/Gate Schematic of Schematic View can see the reserved modules

connection situation.

**2. Synthesis max_fanout**

Currently, the fanout settings in the TD software are global. If you want to specify the fanout of certain networks,

Reduce fanout and help timing optimization by automatically copying instance during synthesis, which can be used in HDL

Add the synthesis directive command in .

The specific methods are as follows:

1. In the verilog file, add a corresponding comment for the signal you want to set fanout. The comment is written as:

//synthesis max_fanout = value; /*synthesis max_fanout = value*/;

*Note: The value in this command is required to be a positive integer and greater than 1.

Such as:

```
reg [7:0] data; wire data_en;   //synthesis max_fanout=10
                                //synthesis max_fanout=20
```

2. Save the file and compile it;

3. When running the flow, you can see the effect of the synthesis directive command in the log.

4. The writing in vhdl is as follows:

```
SIGNAL  SYNTHESIZED_WIRE_79 :  STD_LOGIC;

SIGNAL  GDFX_TEMP_SIGNAL_1 :  STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL  GDFX_TEMP_SIGNAL_0 :  STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL  GDFX_TEMP_SIGNAL_2 :  STD_LOGIC_VECTOR(3 DOWNTO 0);

attribute max_fanout : integer;
attribute max_fanout of SYNTHESIZED_WIRE_79: signal is 20;
```

Among them, SYNTHESIZED_WIRE_79 is the signal that wants to set fanout.

**3. Synthesis ram_style**

The synthesis directive 'ram_style' is used in the same way as the synthesis directive 'keep_hierarchy',

Written in the user's source code in the form of comments. ram_style has the following values that can be set:

bram: instructs the TD software to use BRAM to implement user-specified objects.

requirements, it will be implemented using SLICE;

dram: Instruct the TD software to use DRAM to implement user-specified objects.

requirements, it will be implemented using SLICE;

none: Instruct the TD software to use SLICE to implement user-specified objects (the front end is REG + Gates

Implementation, the backend is implemented with SLICE).

The specific methods are as follows:

1. In the verilog file, add a corresponding comment for the object you want to set ram_style, and write the comment

The law is:

//synthesis ram_style = bram //synthesis

ram_style = dram //synthesis ram_style =

none

*Note: This command cannot be guaranteed to be implemented according to the set implementation method, and the corresponding set requirements must be met.

Such as:

reg[7:0] mem[127:0]; // synthesis ram_style=dram

```
reg [7:0] mem1 [15:0];   // synthesis ram_style=dram
reg [7:0] mem2 [15:0];   // synthesis ram_style=none
```

2. The writing in vhdl is as follows:

attribute ram_style : string; attribute

ram_style of mem1 : signal is "dram";

3. Save the file and compile;

4. In the sim.v file generated after RTL/Gate/Place_Routing, see whether the TD is implemented according to the settings

way to achieve.

```
AL_LOGIC_DRAM #(
  .ADDR_WIDTH_R(4),
  .ADDR_WIDTH_W(4),
  .DATA_DEPTH_R(16),
  .DATA_DEPTH_W(16),
  .DATA_WIDTH_R(8),
  .DATA_WIDTH_W(8))
  al_ram_mem1 (
  .di(din[7:0]),
  .raddr(ra),
  .waddr(wa),
  .wclk(clk),
  .we(we[0]),
  .do(n1));
```

5. If the implementation of BRAM is set, you can see BRAM in Gate/Physical Summary

The resource usage of the Schematic View can be seen in the RTL Schematic/Gate Schematic of the Schematic View

is implemented as a BRAM.

## **5.3** Gate-level optimization

Gate-level optimization includes general logic optimization and mapping, special logic optimization and mapping and other comprehensive optimizations. door level ization will result in a circuit containing logic cells and special-purpose functional units.

1. Expand **HDL2Bit** in the FPGA Flow panel

2. Double-click **Optimize** Gate, or right-click **Optimize Gate,** select Run, and the file will be generated:

   gate.area, the content of this file will be related to the setting of the keep_hierachy parameter.

3. Parameter configuration

In the menu bar, expand **ProcessÿProperties, the** Properties Configuration window pops up, select **Optimize Gate** to set.

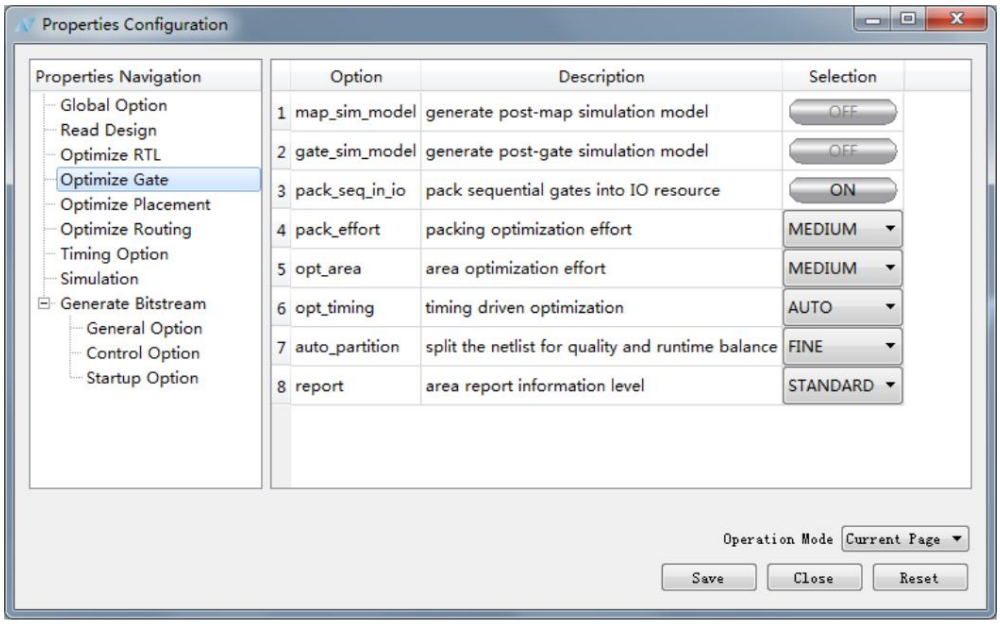


Table 5-3 Optimize Gate Properties

| Property | Comments | Default |
|---|---|---|
| map_sim_model | Generate a post-mapped circuit simulation model | OFF |
| gate_sim_model | Generate gate-level circuit simulation models | OFF |
| pack_seq_in_io | Absorb register logic into IO module | ON |
| pack_effort | Logic wrapping optimization level | MEDIUM |
| opt_area | Combinatorial logic optimization level | MEDIUM |
| opt_timing | Timing optimization level | AUTO |
| auto_partition | The netlist can be partitioned at runtime | FINE |
| report | report information level | STANDARD |

## 5.4 Layout optimization

After getting the correct physical cell netlist, the design needs to be optimized for physical layout, IO cell layout, physical

Logic cell layout and physical-level logic optimization, etc. Layout optimization will generate and process blocks containing only physical functions (IOPAD,

SLICE, RAM, DSP, etc.).

1. Expand **HDL2Bit** in the FPGA Flow panel

2. Double-click **Optimize Placement, or right-click Optimize Placement and** select **Run**

3. Parameter configuration

In the menu bar, expand **ProcessÿProperties, the** Properties Configuration window pops up, select

**Optimize Placement** to set.

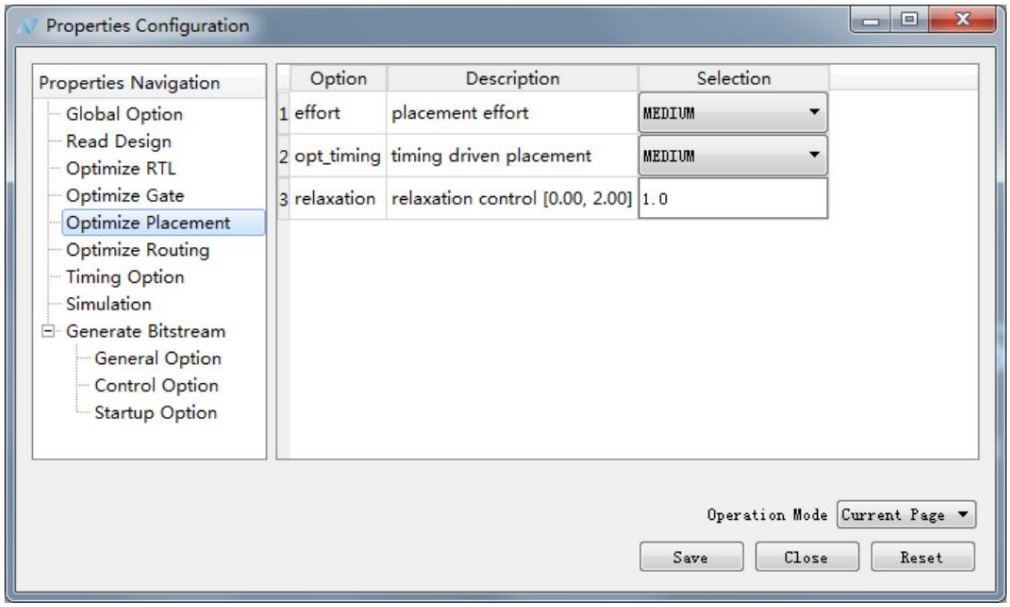



Table 5-4 Optimize Placement Properties

| Property | Comments | Default |
|---|---|---|
| effort | Layout optimization level | MEDIUM |
| timing | Latency optimization level | MEDIUM |
| relaxation | Relaxation control range [0.00, 2.00] | 1.0 |

## **5.5** Routing Optimization

After layout optimization, routing optimization is performed. Cabling optimization will complete the physical connection of all module interconnect signals. this

Step is also the last step in the implementation of user design. After this step, all physical information is determined. Routing optimization

Later, you can view the details of the design and obtain accurate circuit timing information.

1. Expand **HDL2Bit** in the FPGA Flow panel

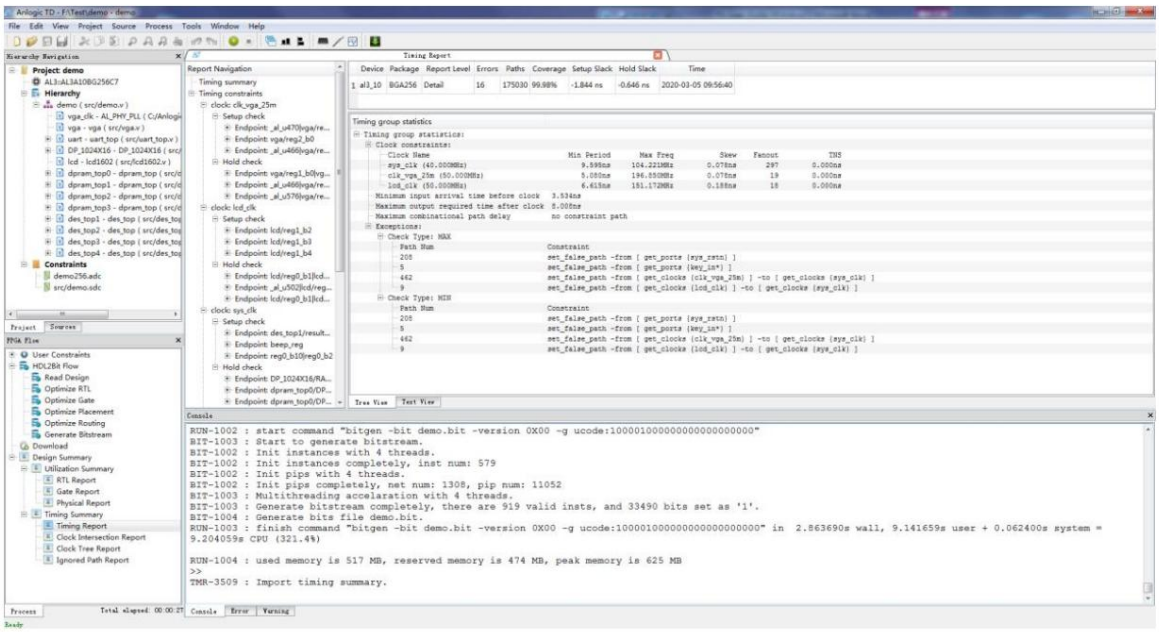
2. Double-click **Optimize Routing, or right-click Optimize Routing** and select **Run .**

   Raw file: phy.area. If the user has added SDC constraints to the project, after the wiring is completed, the TD will default to

   Generate timing analysis reports for users. Expand the **Design Summary** in the FPGA Flow panel , select

   Timing **Report** in **Timing Summary** to view the final timing report. If the user does not

   When adding the SDC file and viewing the **Timing Report ,** TD will prompt the user that there is no timing constraint.




3. Parameter configuration

In the menu bar, expand **ProcessÿProperties, the** Properties Configuration window pops up, select

**Optimize Routing** to set.

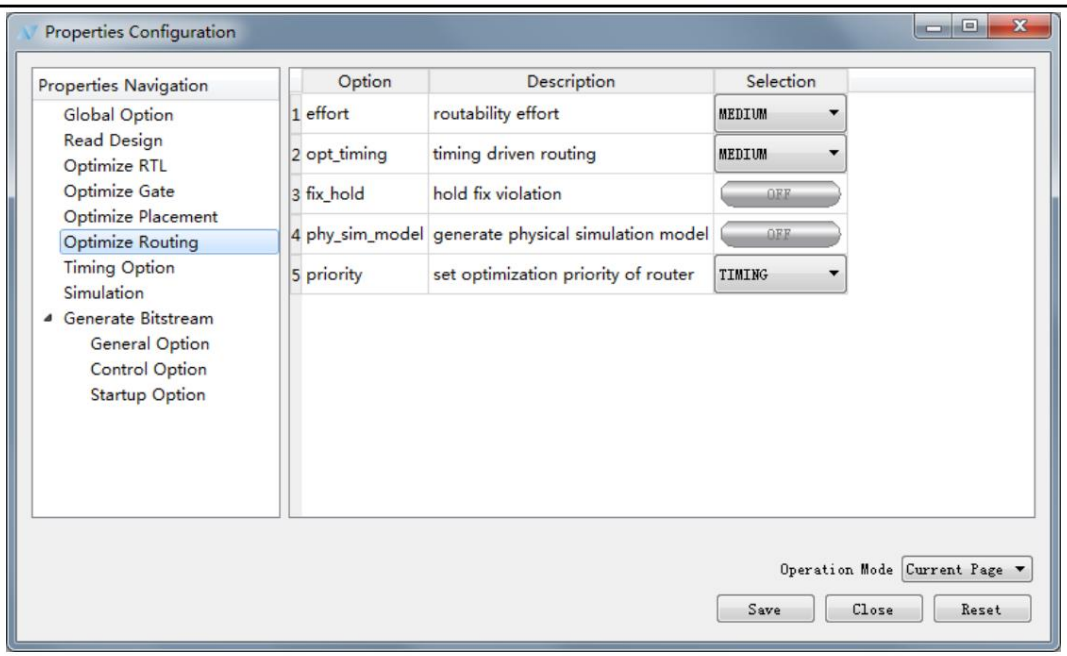



Table 5-5 Optimize Routing Properties

| Property | Comments | Default |
|---|---|---|
| effort | Routine optimization level | MEDIUM |
| opt_timing | Latency optimization level | MEDIUM |
| fix_hold | fix hold violation | OFF |
| phy_sim_model | Generate physical-level circuit simulation models | OFF |
| priority | Prioritize routing optimizations | TIMING |

## 5.6 Generate Bitstream File

**Generate Bitstream** is to use the binary 0, 1 format for the configuration information of the programmable switches in the FPGA chip

Represented as bitstream data for programming download. The bitstream generator generates a bitstream file for device programming and downloads the

It can load the bit stream file into the 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 Generate Bitstream,** select **Run,** then the

   Generated file: Your_Project_Name.bit, this file is programmable in binary 0, 1 format

   switch configuration information

3. Parameter configuration

In the menu bar, expand **ProcessÿProperties, the** Properties Configuration window pops up, select

**Generate Bitstream** to set.

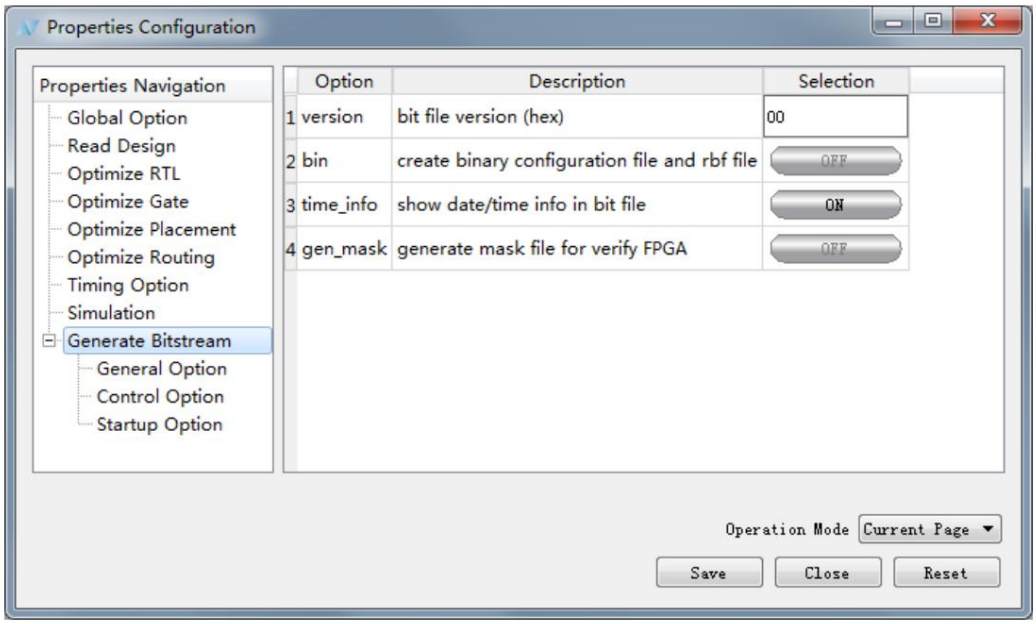


Table 5-6 General Option

| Property | Comments | Default |
|---|---|---|
| version | Define the number of the bitstream | 00 |
| bin | file to generate a pure binary bitstream file, that is, does not contain regular bits The file header of the | OFF |
| time_info | file displays date/time information in the bitstream file | ON |
| gen_mask | Generate a mask file for verifying FPGA | OFF |

Control Option is a 32-bit control register, divided into several control options, changing the value of each item,

Different control effects can be achieved. For different devices, the control option options are different.

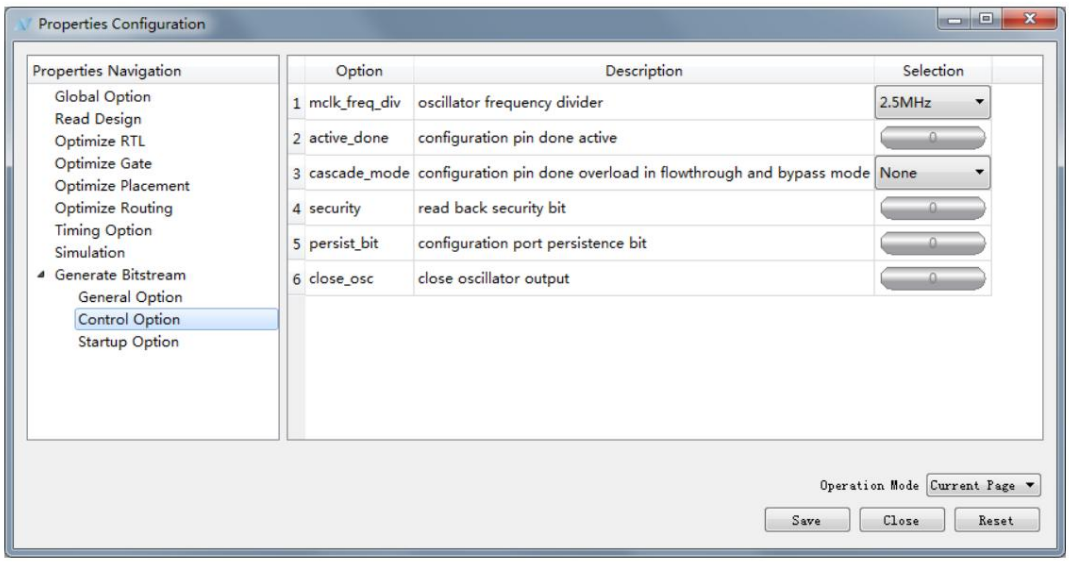



Table 5-7 Control Option

| Property | Comments | Default |
|---|---|---|
| mclk_freq_div | Clock division factor of spi flash | 2.5MHz |
| active_done | done pin is in active state, determined internally by cfg | 1 |
| cascade_mode | Used when cascading, done pin as input ( 2'b11: flowthrougth mode 2'b10: bypass mode) | None |
| security | When it is 1, read sram is forbidden | 0 |
| persist | Whether the pin of spi will continue to be used as cfg pin in user mode | 0 |
| close_osc | Turn off the oscillator | 0 |

Startup Option is a 32-bit control register related to the startup item, and its control principle is the same as that of Control Option.

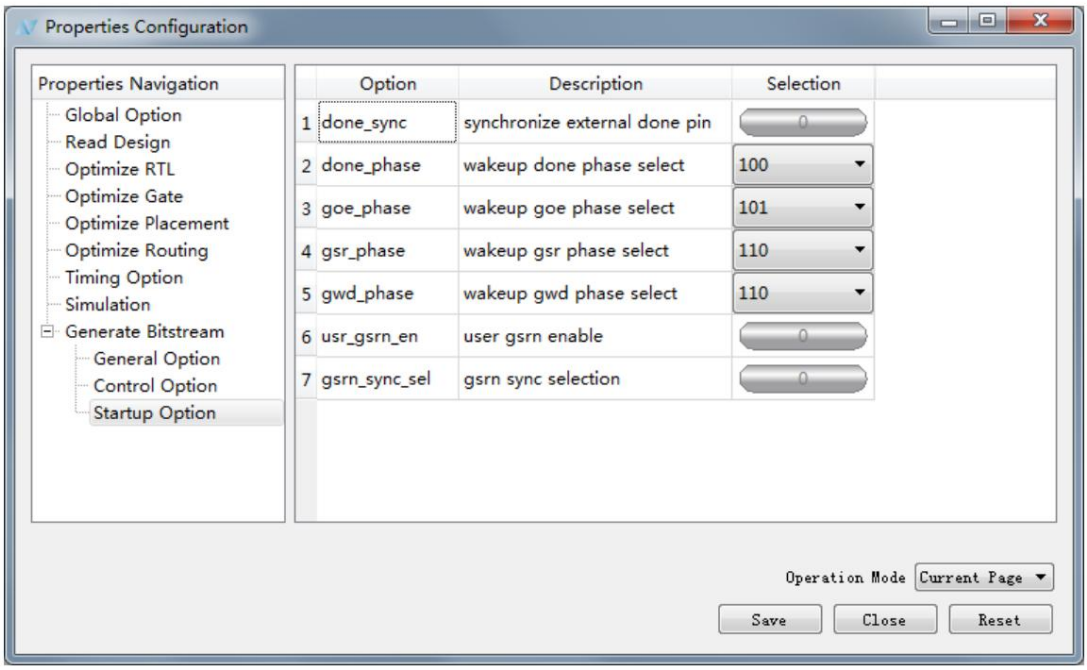



Table 5-8 Startup Option

| Property | Comments | Default |
|---|---|---|
| done_sync | Whether to synchronize the value of | 0 |
| done_phase | done pin determines in which phase to release | 3'b100 |
| go_phase | done Determine in which phase to release | 3'b101 |
| gsr_phase | goe to determine in which phase to release | 3'b110 |
| gwd_phase | gsr Determine in which phase to release gwd | 3'b110 |
| usr_gsrn_en | whether to enable the user's gsrn signal | 0 |
| gsrn_sync_sel | whether to synchronize gsrn | 0 |

# 5.7 syn_ip_flow

In the design process of FPGA, if users do not want to let third parties see their own source code, they can check the source code of the design.

The code implements protection measures, that is, the source code is separately made into an IP module for the third party to call. To this end, TD gives users

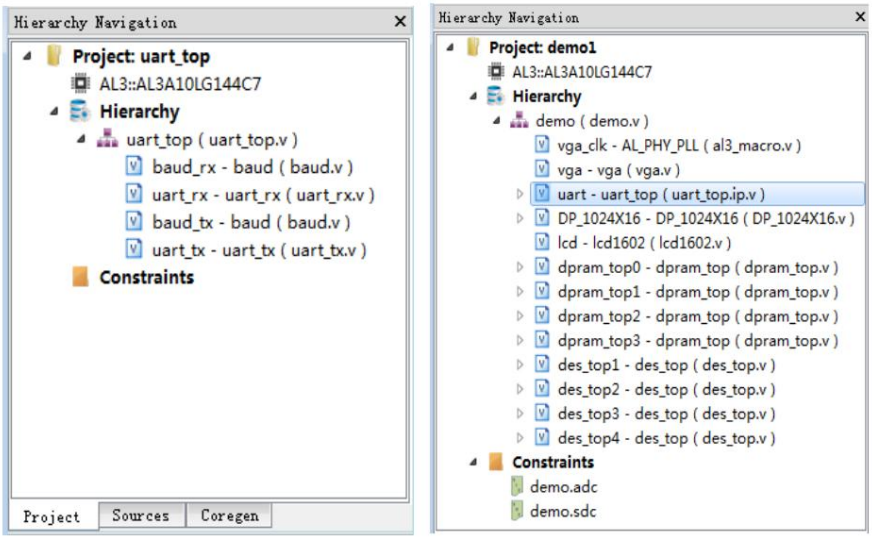
Provides the function of syn_ip_flow, syn_ip_flow synthesizes the user's high-level circuit description into a gate-level netlist, which can be found in

The source code is protected to some extent. When users use this function, they need to create two projects and add them in the first project.

Need to protect the source code and run syn_ip_flow, add peripheral circuits in the second project and call syn_ip_flow

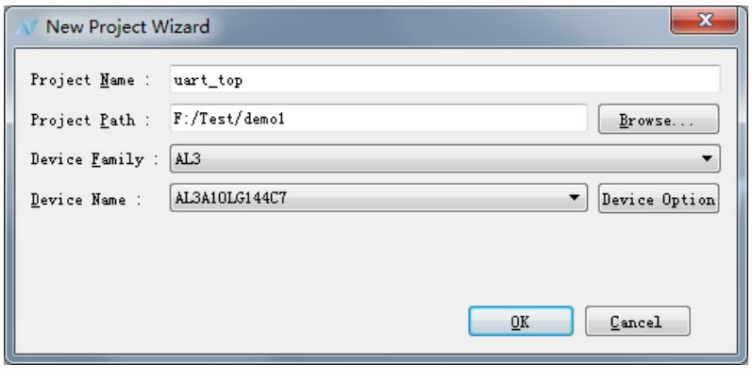
Generated netlist file. For example: in the demo module used in this manual, the submodule the user wants to protect is uart_top,

Then you need to create the following two projects:

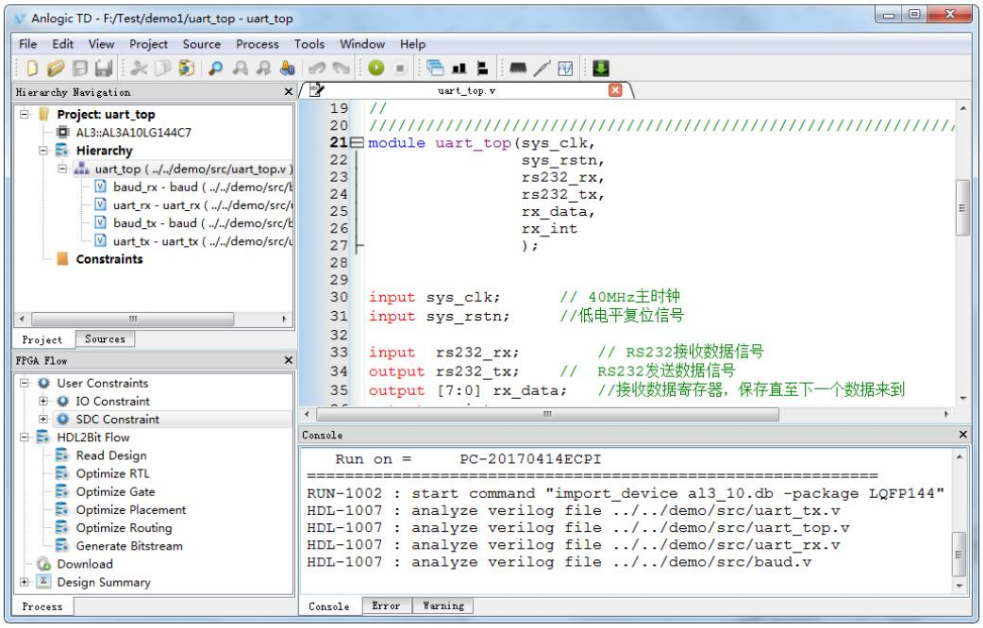



The specific operation steps are as follows:

1. Create a separate project for the submodule to be protected. In this example, take the uart module as an example to create a separate project.



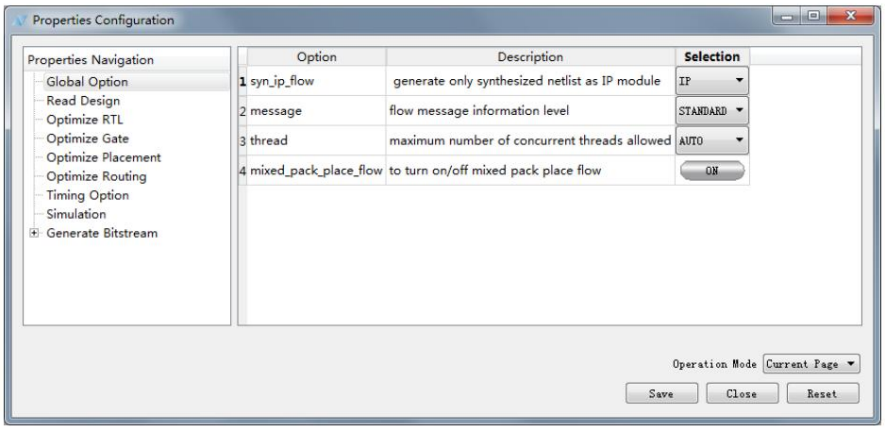
2. Add source files to the project

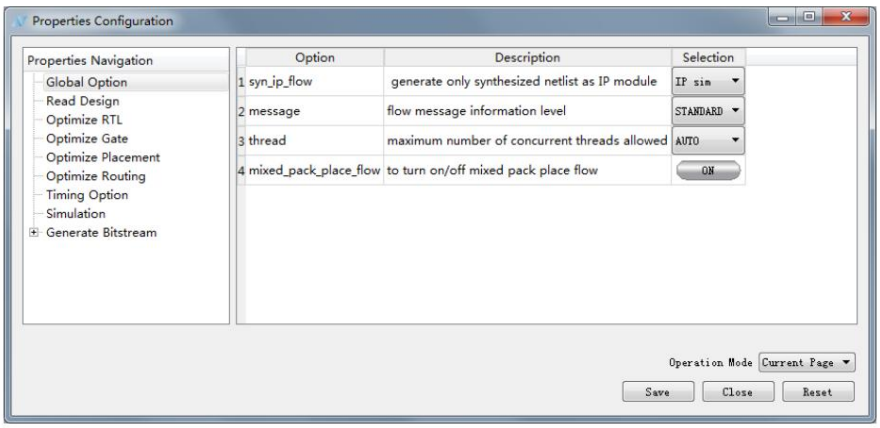



3. When running syn_IP_flow, you need to set the parameters first:

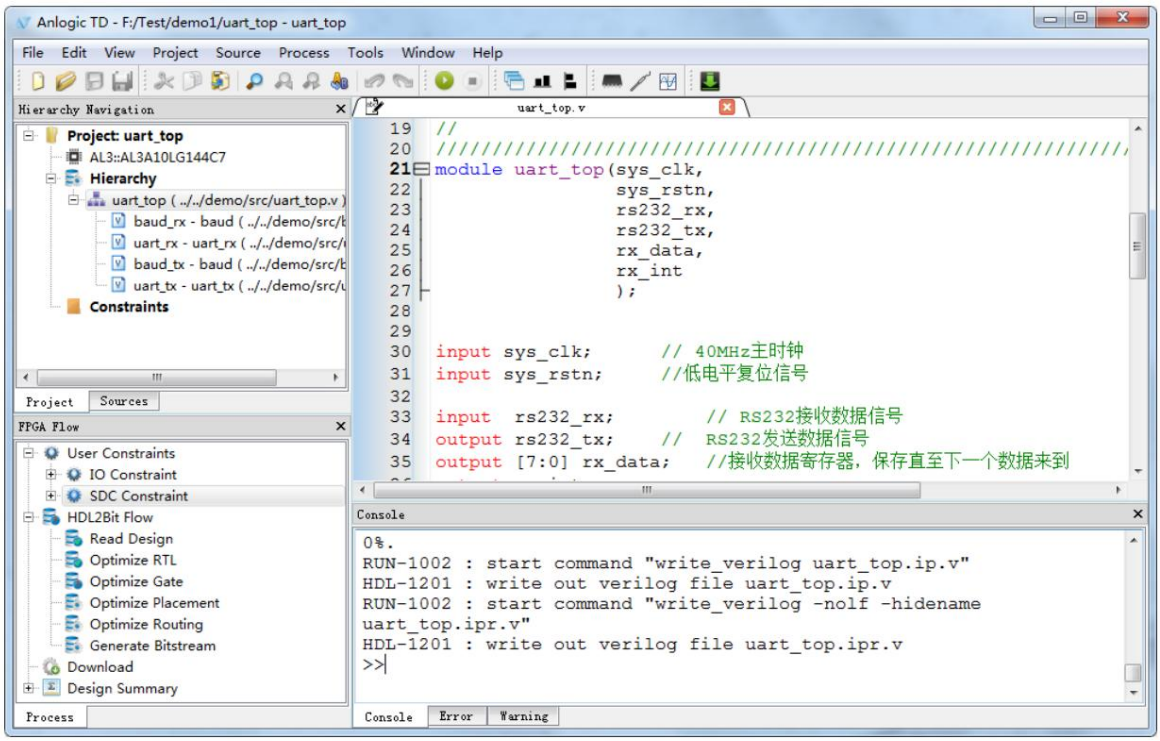
**Process ÿ Properties ÿ Global Option ÿ syn_ip_flow,** set this parameter to **IP/IP**

**sim, and click "Save" to save.**



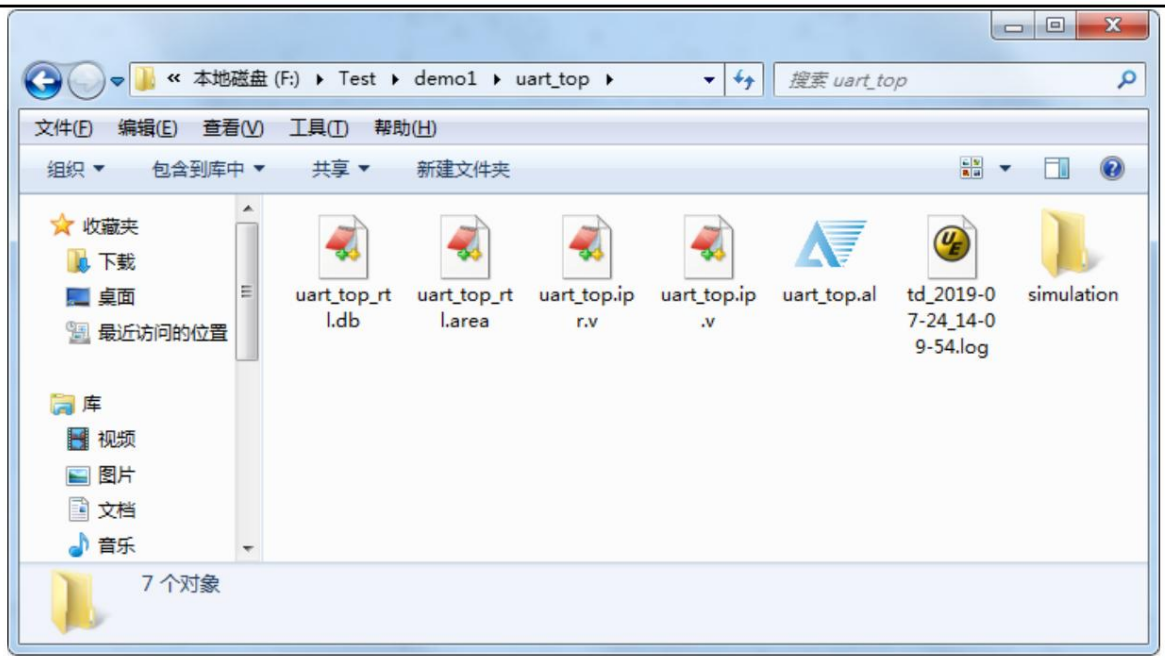



4. After the parameter setting is completed, click the icon to run the project. After the project is completed, the TD will generate two files: top_name.ip.v and top_name.ipr.v, and stored in the project directory, these two files Functionally equivalent. However, top_name.ip.v contains information such as names and declarations in the source code. information for easy debugging. Top_name.ipr.v hides the naming and declaration information in the source code, which can be updated Well protected source code.



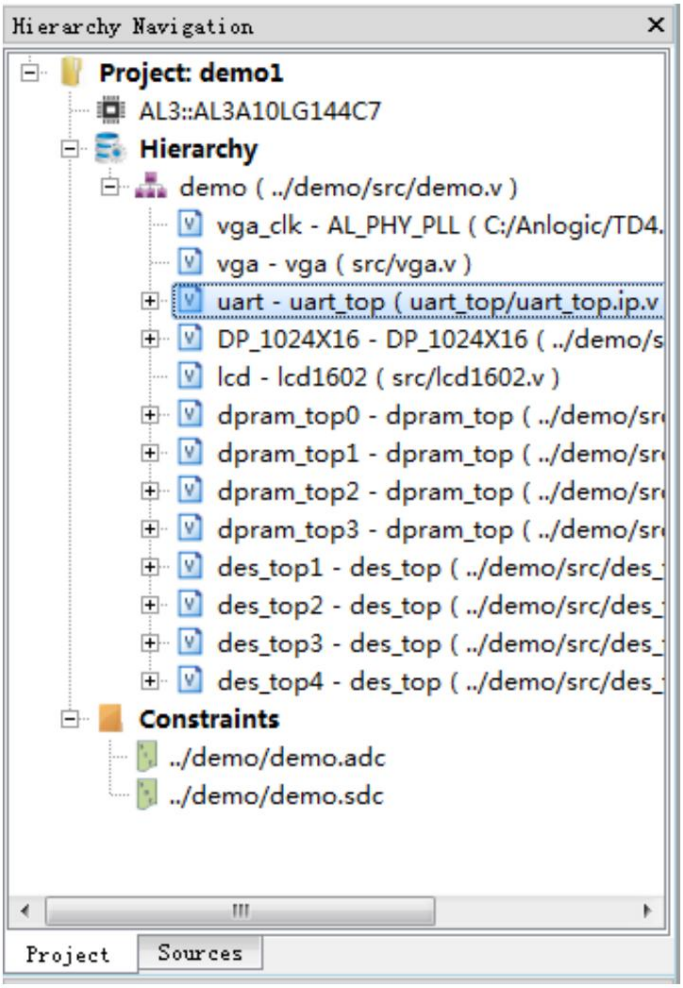

ÿ When the parameter is set to **IP ,** the generated file can be directly added to the required project, but cannot be used for Simulation, otherwise the simulation will report an error;

ÿ When the parameter is set to **IP sim ,** the generated file can be used for simulation. If it is to be added to the project, the Need to add al_map_basic.v additionally, The file path is: the installation directory where td is located /sim/al/al_map_basic.v.




5. Create a new project, call the top_name.ip.v or top_name.ipr.v generated in the previous step, and add

Add the corresponding constraint file and run the whole project, then the whole process of syn_ip_flow is completed.



# 5.8 Design Summary

## 5.8.1 RTL Report

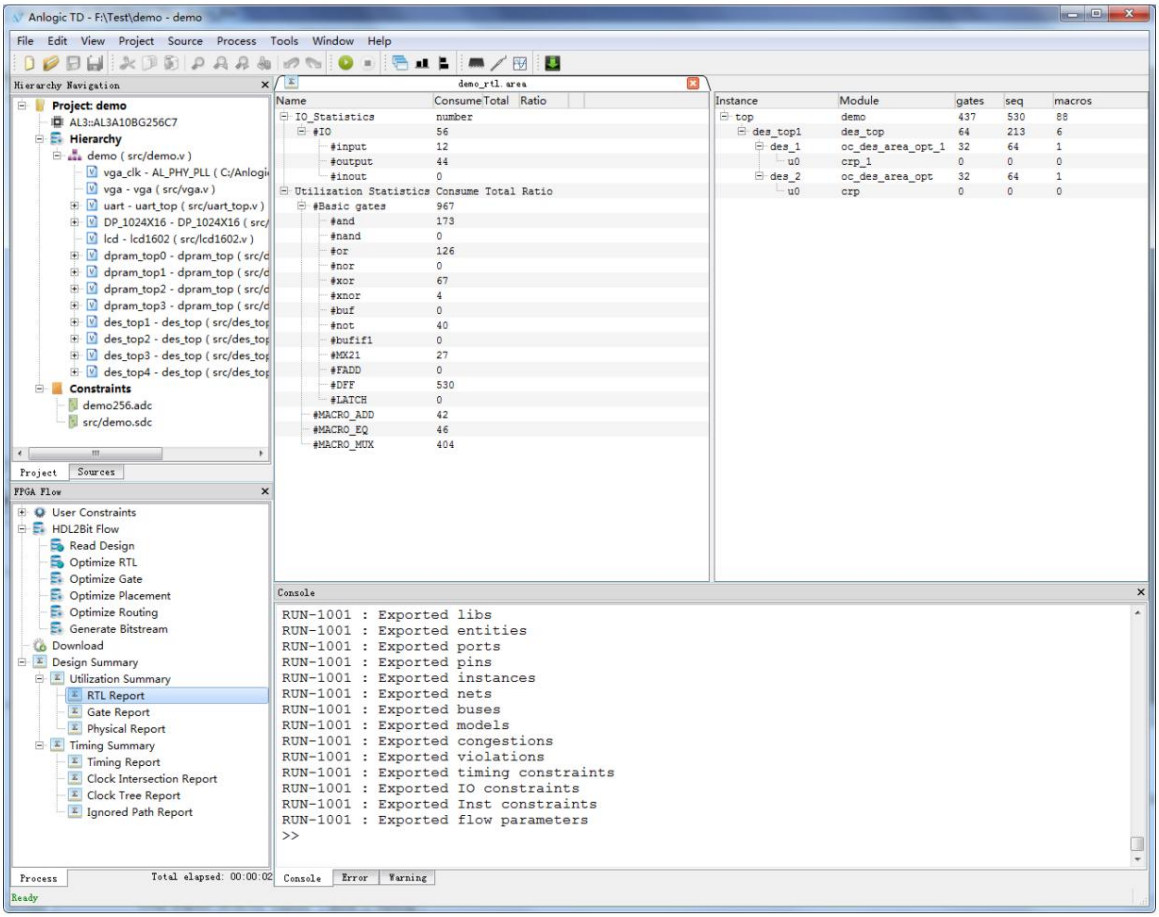
When HDL2Bit Flow finishes running Optimize RTL, you can expand **Design Summary ÿ Utilization Summary, double-click to view the RTL Report.**

When keep_hierarchy is selected as manual, td

Module retention hierarchy, this file will hierarchically give the resource usage of sub-modules; when keep_hierarchy is auto,

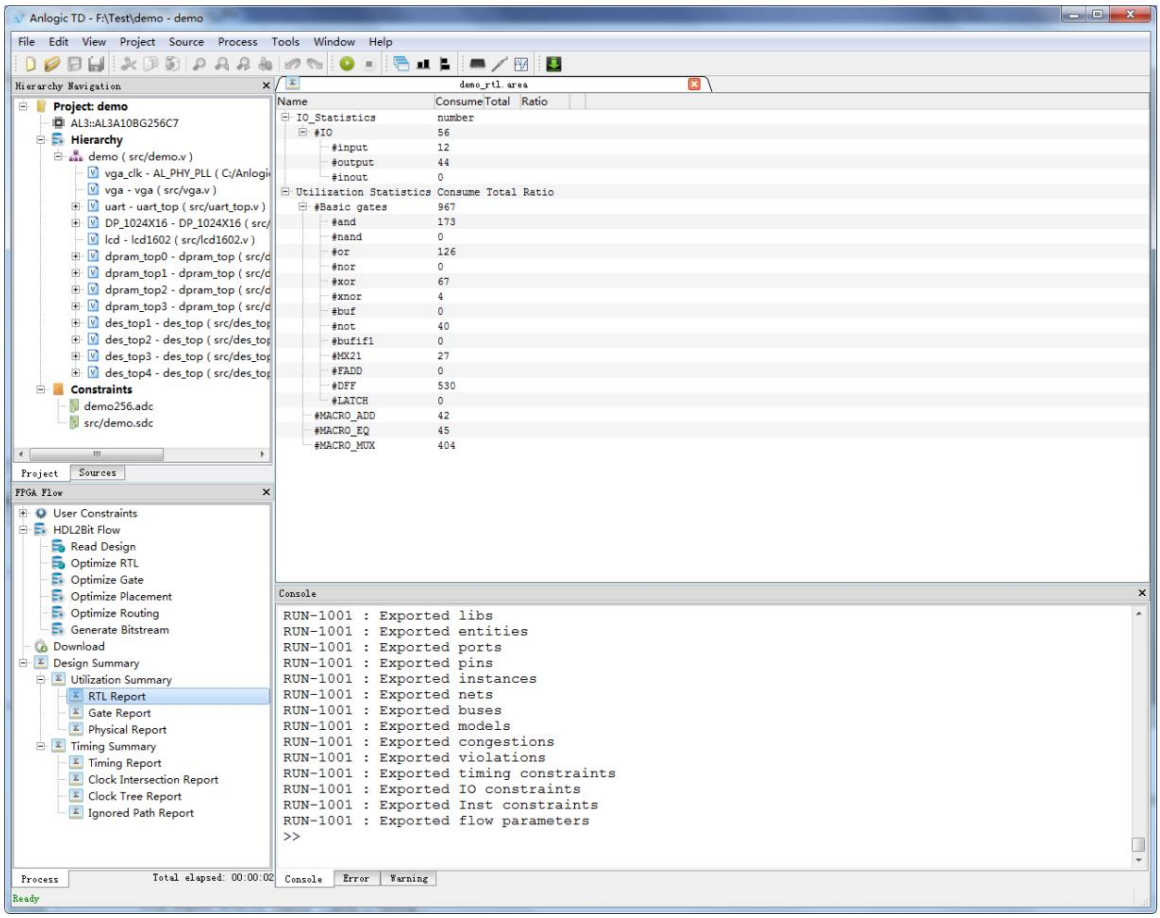
td will automatically select the larger module retention level, and this file will hierarchically give the resource usage of submodules.

Hierachy RTL Report:



When keep_hierarchy is flatten, the file contains all input and output ports in the source file and each Logic gate usage.

Flatten RTL Report:



# 5.8.2 Gate Report

When HDL2Bit Flow finishes running Optimize Gate, you can expand **Design Summary ÿ Utilization**

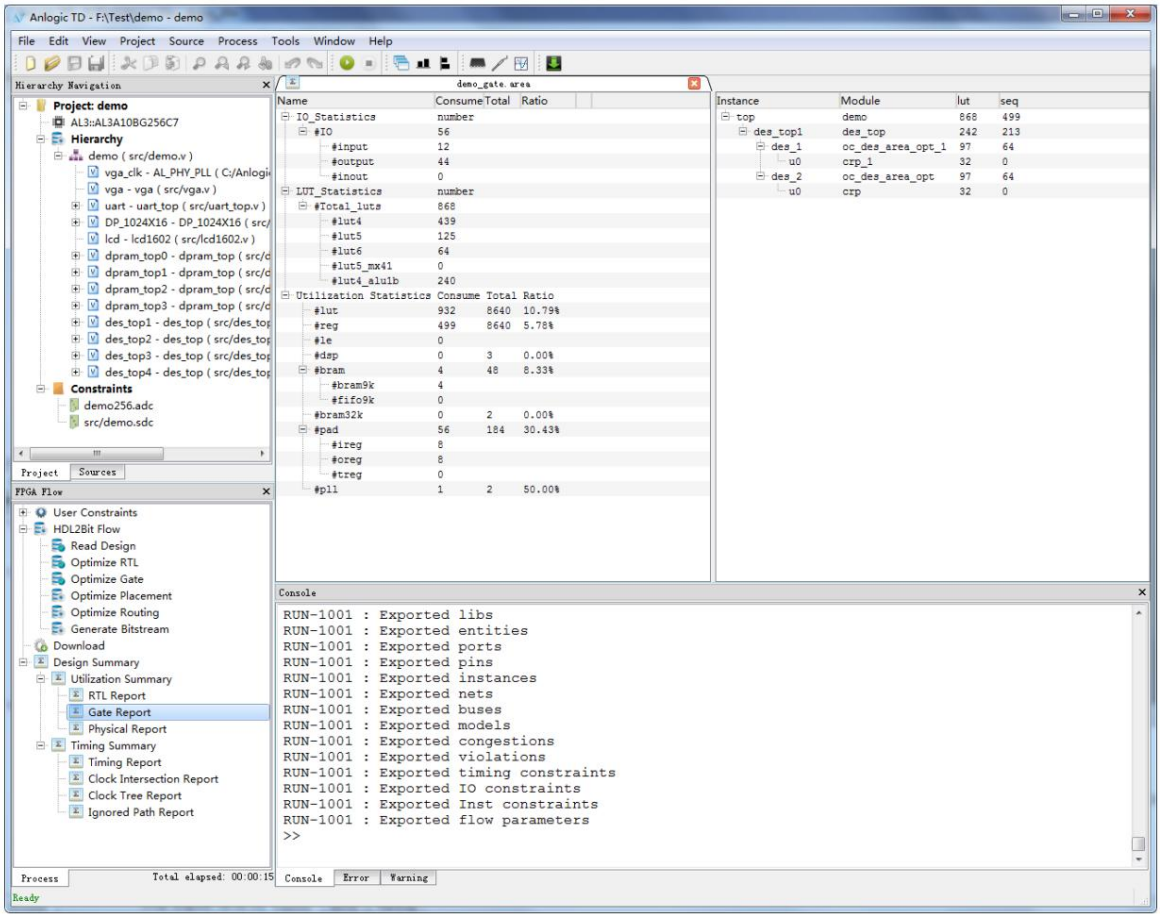
**Summary, double-click to view the Gate Report.**

When keep_hierarchy is selected as manual, td is only selected by the user through the synthesis directive

The module retention hierarchy of , this file will hierarchically give the resource usage of sub-modules; when keep_hierarchy is auto

, td will automatically select a larger module retention level, and this file will hierarchically give the resource usage of sub-modules.

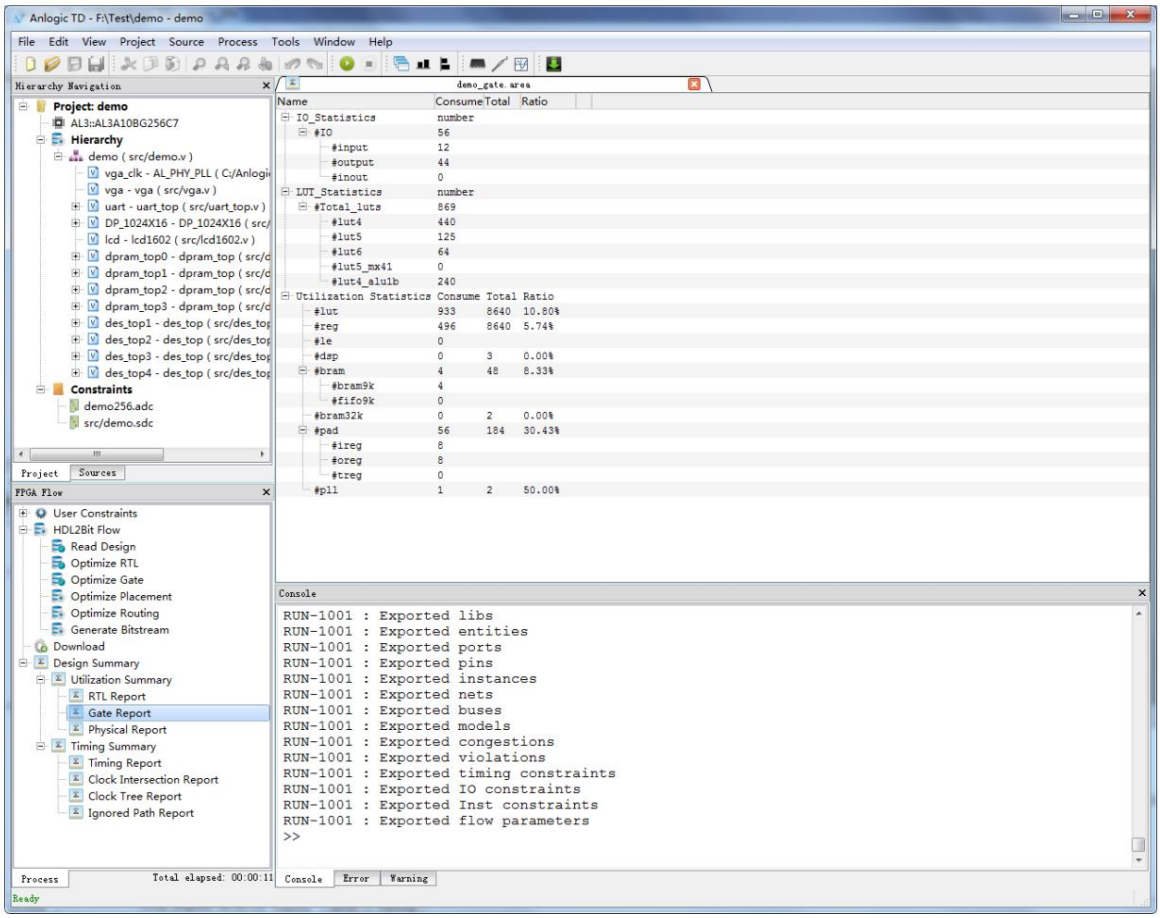
Hierarchy Gate Report:



When keep_hierarchy is flatten, the file contains all input and output ports in the source file and

The usage of each logic gate.

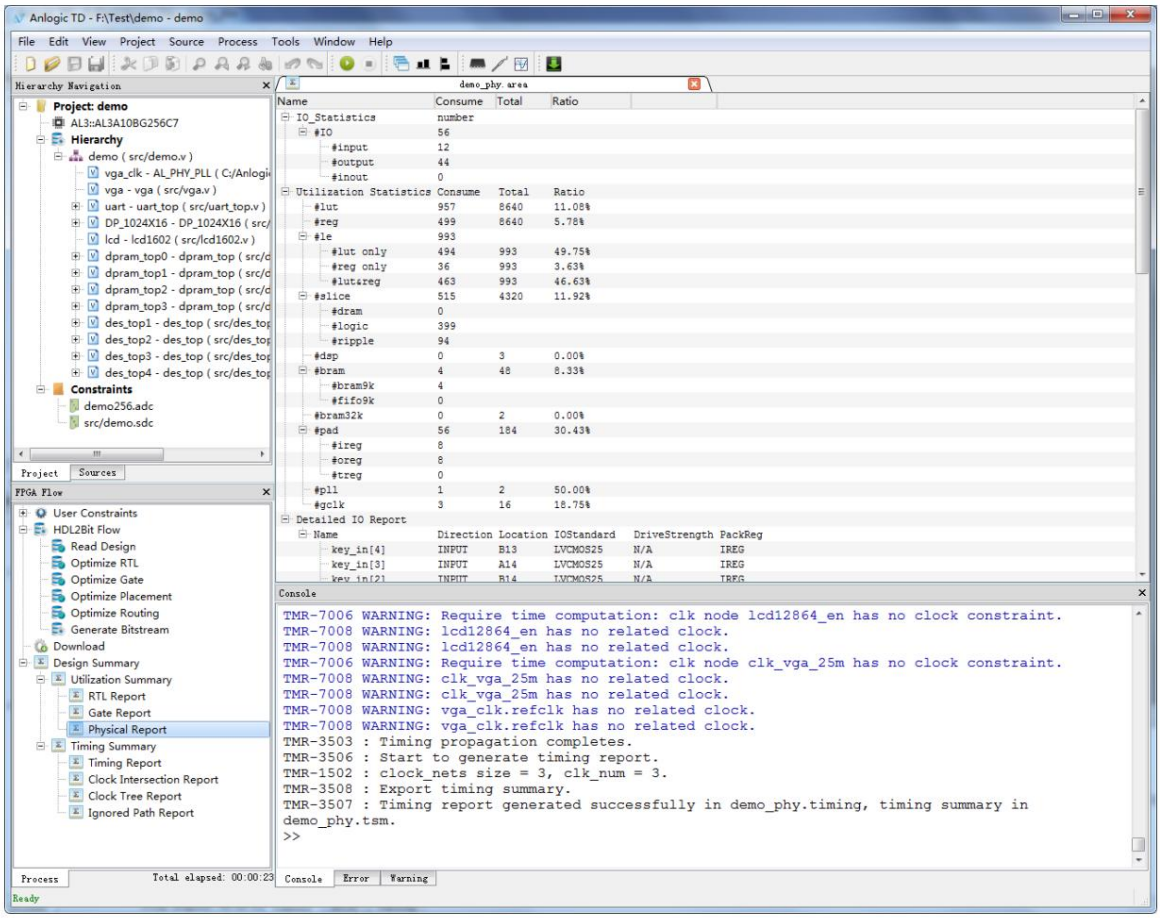
Flatten Gate Report:



## 5.8.3 Physical Report

When HDL2Bit Flow finishes running Optimize Routing, you can expand **Design Summary ÿ Utilization Summary, double-click to view the Physical Report.**

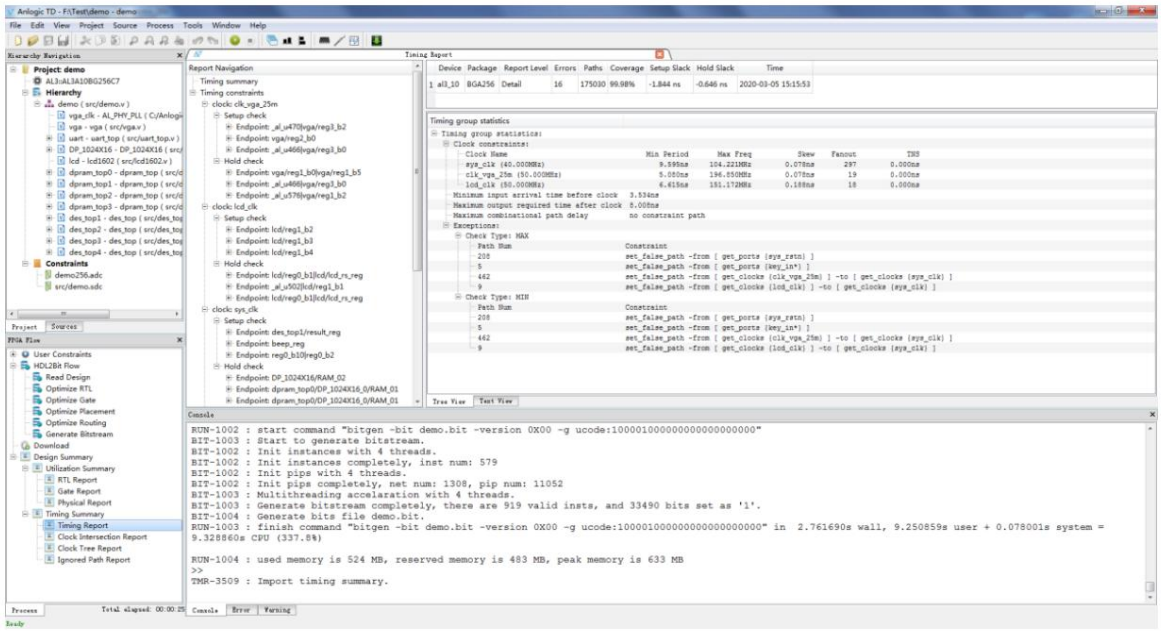
This file lists IO port and basic logic cell usage and IO pin assignments.



# 5.8.4 Timing Report

When the HDL2Bit Flow is completed, expand **Design Summary ÿ Timing Summary, double-click to check**

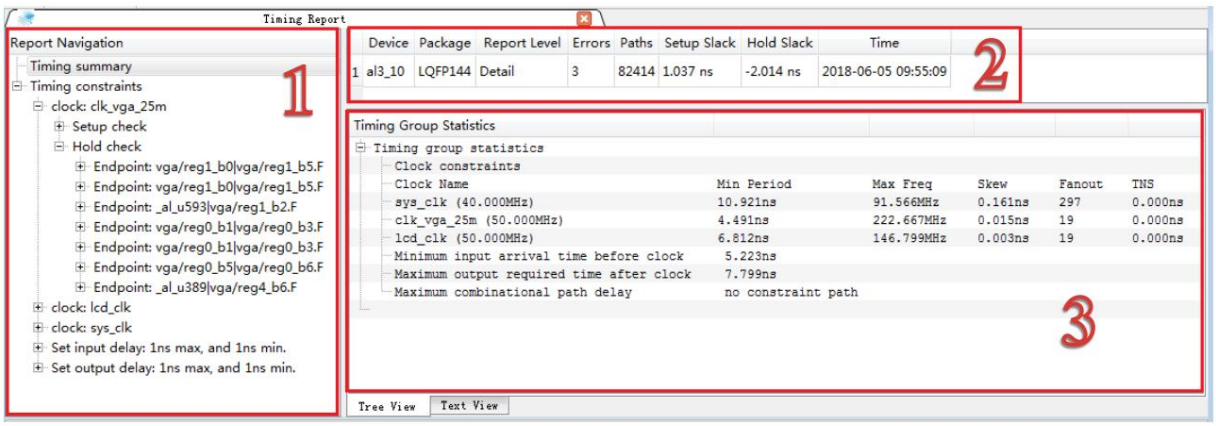
Looking at the Timing Report, the main interface is as follows:



If there is a timing unsatisfied condition in the Timing Report, there will be a timing report in the lower left corner of the interface.

warning Flag to remind.

The interface of Timing Report mainly includes the following parts:

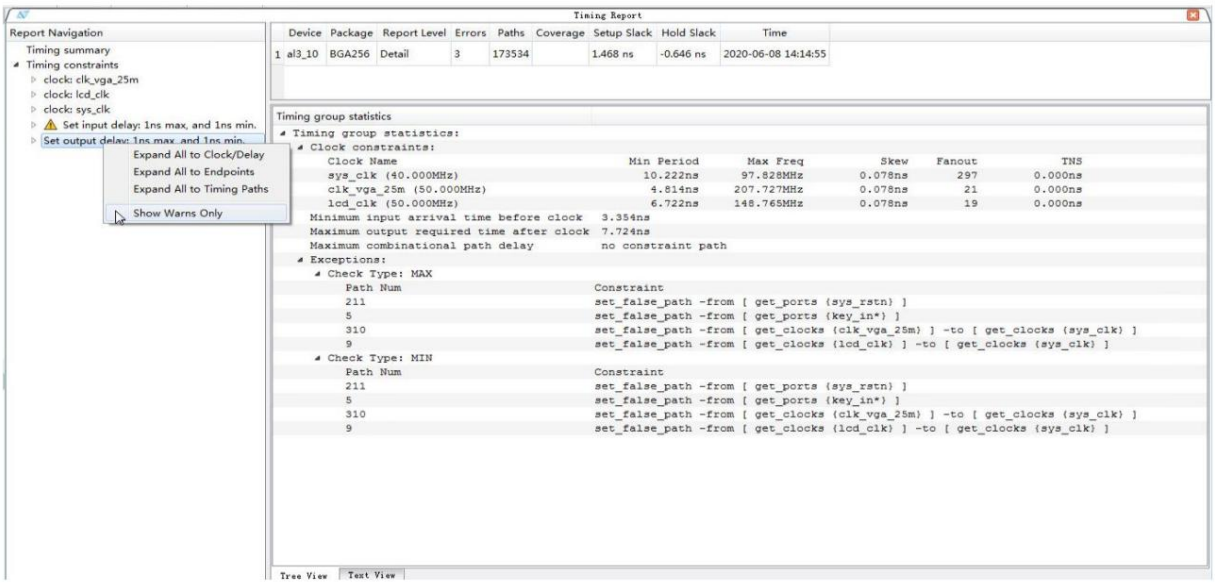



1. Report Navigation

Here is the directory tree of the timing report, which will be listed in order: timing constraint type, timing check type,

EndPoint name, timing path name and other information.

For clocks that do not meet the timing, there will be a warning sign on the left side of the clock. Right click and select Show Warns Only, will only display clocks that do not meet timing and related timing paths that do not meet timing. Right click to Select Expand All to display in Clock/Delay or EndPoints or Timing Paths.




2. Timing Summary

Chips, packages, and brief timing statistics for timing reports are listed here.

3. Timing Group Statistics

Clock constraints: lists the minimum period, maximum frequency that each clock defined in SDC can reach,

The skew of the clock tree, the number of fanouts, and the TNS data; for clocks that do not meet the timing, there will be a clock on the left side of the clock name.

warning Flag to remind.

Minimum input arrival time before clock: the maximum delay of input ÿ reg path inside the FPGA chip

Time.

Maximum output required time after clock: the maximum value of reg ÿ output path inside the FPGA chip

delay.

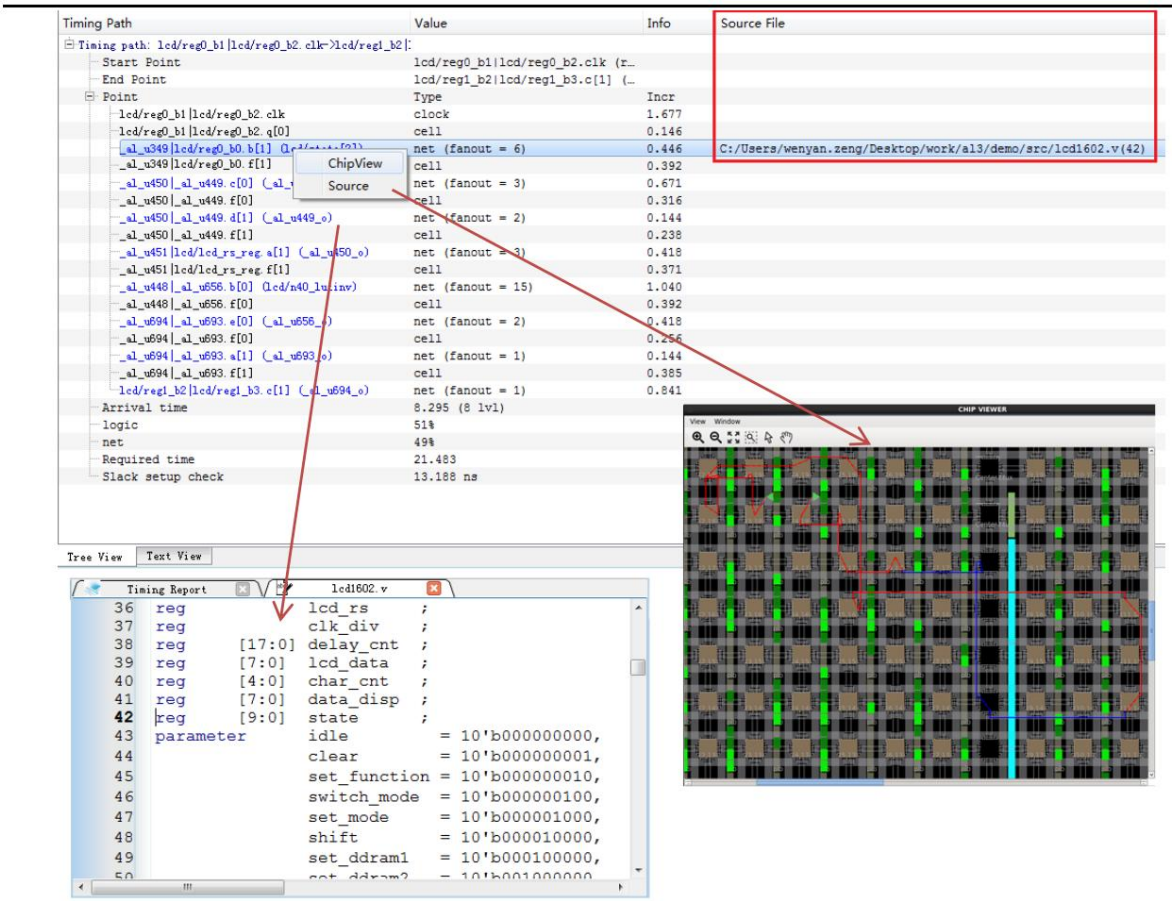
Maximum combinational path delay: The maximum delay of the combinational logic path of input ÿ output straight-through.

Expand Setup Check or Hold Check under clock in Timing constraints,

Proerties set to display the corresponding number of timing paths, double-click a timing path, it will be displayed in the Timing path

The right column of the Report shows the details of the timing path, where the main entries have the following meanings:

| | |
|---|---|
| Start Point refers to the starting point of the timing path, generally the clock end of the register or the original input end | |
| End Point refers to the end point of the timing path, which is generally the input data pin or the original output terminal of the timing unit. | |
| Point | The node names that the timing path passes through are listed in turn |
| Type | Indicates whether the timing path of this segment passes through a cell or a net |
| Incr | Delay increment of this segment of timing path |
| Source File indicates where the net is defined in Source File (need to open the net_info option) | |
| Arrival time represents the arrival time of this timing path:<br>Arrival time = Tlaunch edge + Tlaunch clock delay + path delay<br>The maximum possible delay in the case of setup check, and the minimum possible delay in the case of hold check<br>, and the arrival time of the trigger clock is also included | |
| Logic | Indicates the proportion of delay occupied by logic resources in the entire timing path |
| Net | Indicates the proportion of delay occupied by interconnect resources in the entire timing path |
| Required time indicates the required time of the timing constraint:<br>Maximum path constraint (setup check): Required time = Tcapture edge + Tcapture clock delay – Tsetup<br>– Tclock uncertainty<br>Minimum path constraint (hold check): Required time = Tcapture edge + Thold + Tclock uncertainty | |
| Slack | Indicates the slack margin of the timing path, and if it is less than 0, it means there is timing risk:<br>Maximum path constraint (setup check): Slack = Required time – Arrival time<br>Minimum path constraint (hold check): Slack = Arrival time - Required time |




Timing path naming rules are:

"/" indicates a hierarchical relationship; "|" indicates a parallel relationship; "." indicates a subordinate relationship.

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

| Point | Type | Incr | Comment |
|---|---|---|---|
| lcd/reg0_b1‖lcd/reg0_b2.clk clock | | 1.677 means reg0_b1 and reg0_b2 in the lcd module are merged | When reaching the same slice, the starting point of the path is the clk end of the slice.<br>clock represents the delay of the clock tree is 1.677ns |
| lcd/reg0_b1‖lcd/reg0_b2.q[0] cell | | 0.146 is the delay inside the cell, that is, the clk end of the slice to the output | The delay of q[0] is 0.146ns. |
| _al_u349‖lcd/reg0_b0.b[1]<br>(lcd/state[2]) | Net<br>fanout=6 | 0.446 is the net delay, that is, the q end of the previous slice to the next | The network delay between the b-ends of the slice is 0.446ns;<br>In parentheses is the net name: net lcd/state[2].<br>fanout=6, indicating that the signal drives a total of 6 pins. |
| … | … | … | |

For the net given the file path, you can right-click the net, and click Source to jump to the corresponding source file.

, click ChipView to display the specific routing of the path in ChipView, and the blue line segment is the net

part in the entire path.

4. Parameter setting

In the menu bar, expand **ProcessÿProperties, the** Properties Configuration window pops up, select

**Timing Option, you can** set parameters for Timing Report.

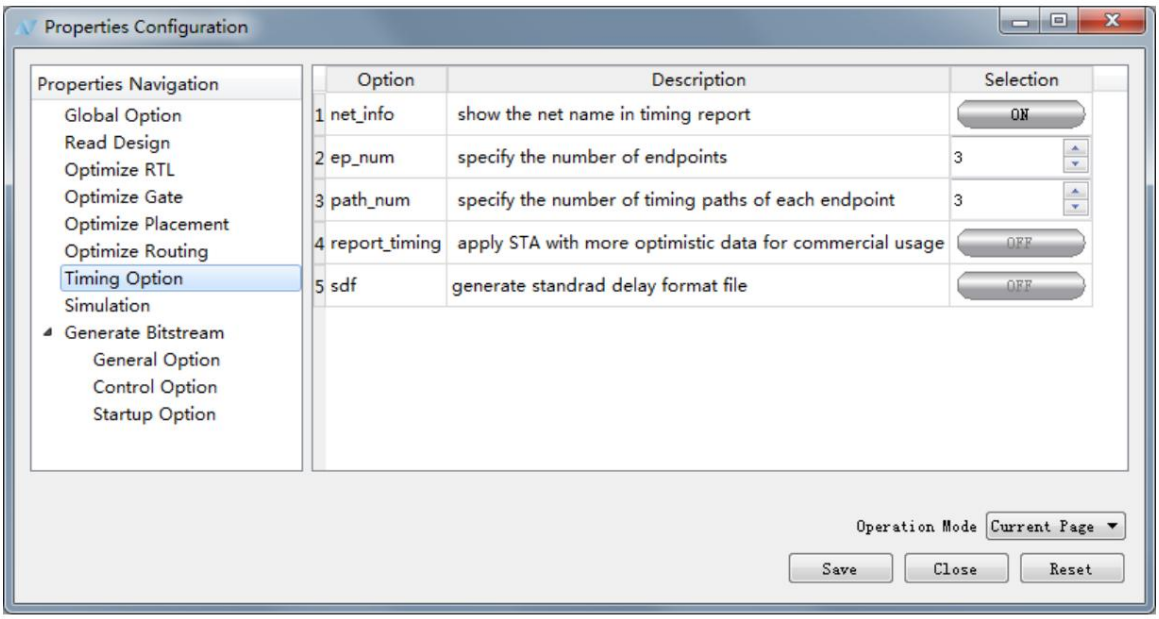


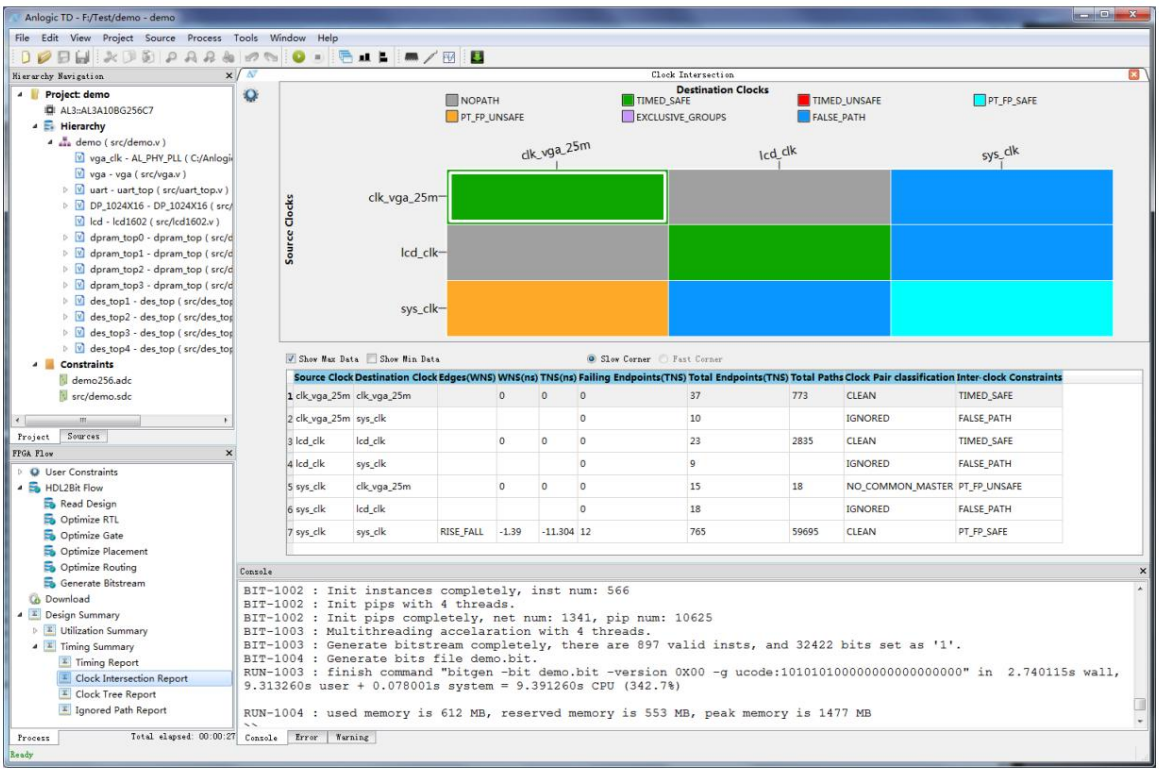

Table 5-9 Timing Option Properties

| Property | Comments | Default |
|---|---|---|
| net_info | Display net information in timing report | ON |
| ep_num | Display the number of endpoints | 3 |
| path_num | Displays the number of timing paths per endpoint3 | |
| report_timing | STA uses more optimistic data | OFF |
| sdf | Generate standard time series backscale files | OFF |

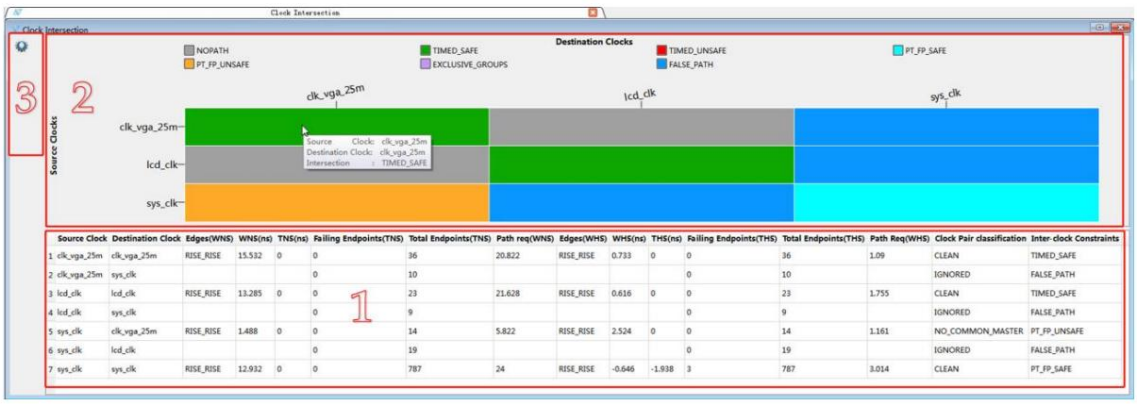
# 5.8.5 Clock Intersection Report

When the HDL2Bit Flow is completed, expand **Design Summary ÿ Timing Summary, double-click to check**

Looking at the **Clock Intersection** Report, the main interface is as follows:




The interface of Clock Intersection mainly includes the following parts:




1. Clock Intersection Data Sheet

Here is the data table of clock interaction. The data in each row of the table is "one-way" by the clock, that is, from a clock

to another clock. If two different clocks (such as clk_vga_25m and sys_clk) mutually signal and capture

Obtained, this table needs to be described in two lines, clk_vga_25m ÿ sys_clk and sys_clk ÿ

clk_vga_25m.

| | Source Clock | Destination Clock |
|---|---|---|
| 1 | clk_vga_25m | clk_vga_25m |
| 2 | clk_vga_25m | sys_clk |
| 3 | lcd_clk | lcd_clk |
| 4 | lcd_clk | sys_clk |
| 5 | sys_clk | clk_vga_25m |
| 6 | sys_clk | lcd_clk |
| 7 | sys_clk | sys_clk |

Among them, source clock and destination clock are defined as follows:

Source Clock The control clock of the starting point DFF of the data path

Destination Clock data path The control clock of the destination DFF

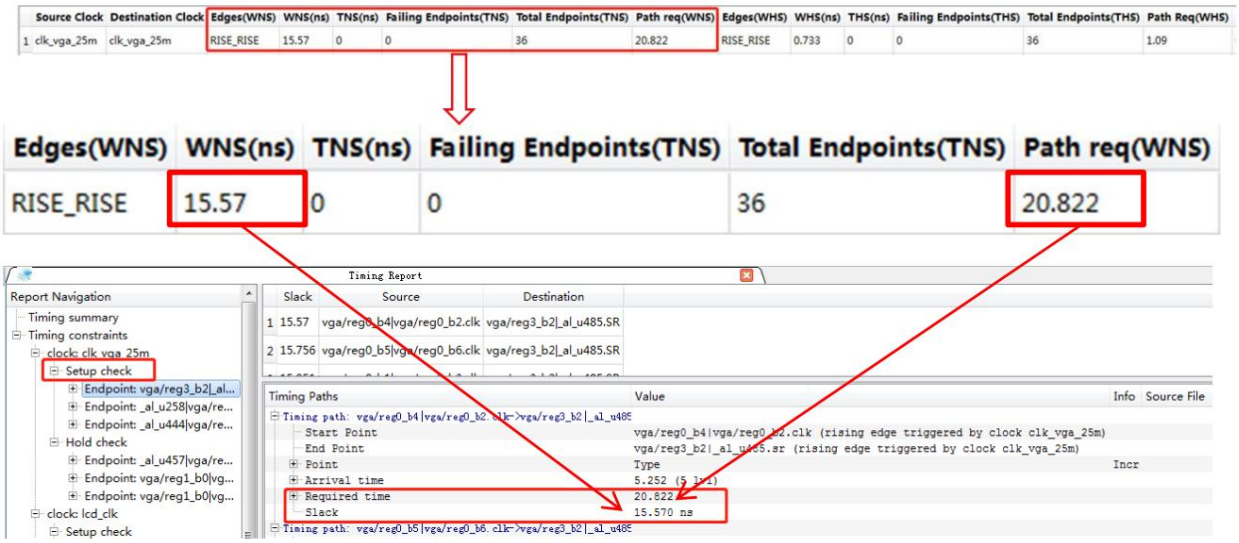
The timing path details are shown in the table and correspond to the information in the Timing Report, with the main

The meaning of the purpose is as follows:

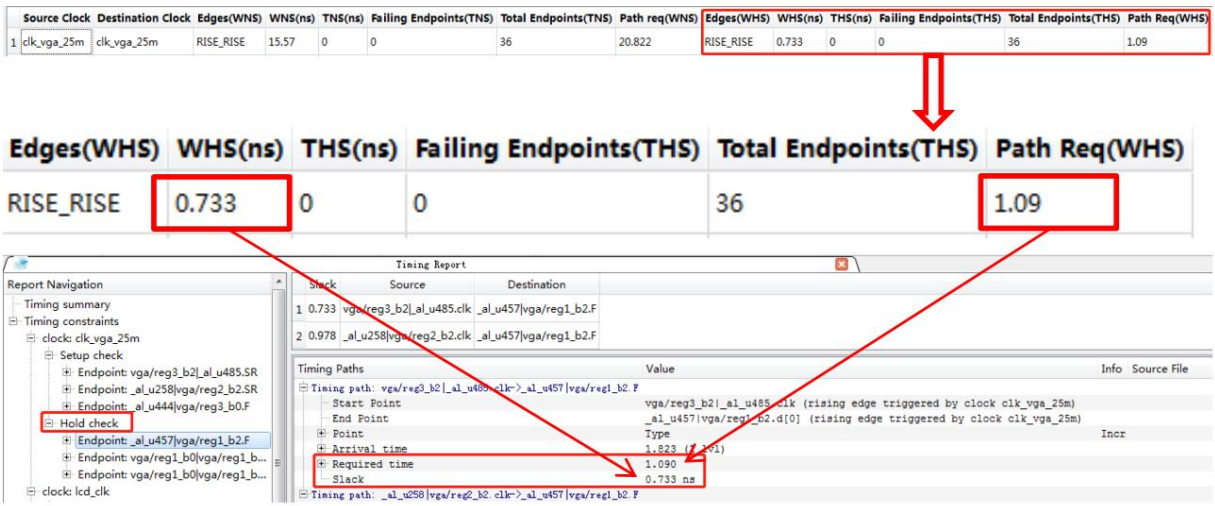
| | |
|---|---|
| Edges(WNS) | The launch clock and capture clock of the most critical path in this one-way clock relationship trigger edge. RISE_RISE indicates that the launch/capture clock is a rising edge trigger hair, similar to RISE_FALL, FALL_RISE, FALL_FALL |
| WNS (ns) | Worst Negative Slack, in all timing paths in this one-way clock relationship, The slack of the most critical path, in ns, not necessarily negative |
| TNS (ns) | Total Negative Slack, the slack in all timing paths for this one-way clock relationship is a negative sum, in ns, which is definitely not positive. If there is no timing violation path, this value is 0 |
| Failing Endpoints(TNS) Endpoints | with negative slack in all timing paths of this unidirectional clock relationship total |
| Total Endpoints(TNS) | The total number of all endpoints for this one-way clock relationship that exist in design |
| Path req(WNS) | Among all timing paths in this one-way clock relationship, the most critical path has require time, in ns |


Open the Clock Intersection Report to display Max Data by default; if you check Show Min Data, it will be hidden

Hide Max Data and display Min Data.




The types of analysis described here are setup or recovery, and the corresponding information in the timing report.




The analysis types described here are hold or remove, and the corresponding information in the timing report.

| Clock Pair classification | Inter-clock Constraints |
|---|---|
| CLEAN | TIMED_SAFE |

Clock Pair classification: Information on static synthesis timing constraints to classify two clocks. may appear

The tags are:

| | |
|---|---|
| IGNORED | Specify this one-way clock using set_clock_groups or set_false_path<br>All paths in the relationship are not considered |
| VIRTUAL | Indicates that at least one of the source or destination clock is a virtual clock, which is mostly reflected in the timing path of the input/output path |
| DERIVED | Indicates that at least one of source or destination clock is derived clock, the clock generated by derived_clocks |
| NO_COMMON_MASTER indicates that the two clocks come from two primary clocks, including the comparison between different primary clocks | |
| NO_EXPAND | Indicates that the two clock cycles cannot find the coincidence of the clock edges within the range of 1000ns point, completely asynchronous clock |
| CLEAN | Indicates that the two clocks belong to the same clock domain defined by the primary clock and have Sync within a limited time span |

Static analysis checks qualitatively in the order described above, and terminates when a definition is satisfied.

Inter-clock Constraints: Dynamically record one-way clock relationships during timing analysis and end the analysis

After completing the qualitative summary. Possible labels are:

| | |
|---|---|
| NOPATH | Timing paths for which this unidirectional clock relationship does not exist |
| FALSE_PATH | All timing paths in this unidirectional clock relationship are false paths |
| EXCLUSIVE_GROUPS All timing paths in this unidirectional clock relationship are grouped by set_clock_groups<br>Constraints require ignoring | |
| TIMED_SAFE | The source and destination clocks belong to the same clock domain and the periods are simply synchronized |
| TIMED_UNSAFE | The source and destination clocks belong to different clock domains or belong to the same clock Domain but period out of sync |
| PT_FP_SAFE | The source and destination clocks belong to the same clock domain and have the same period. step, some timing paths are false paths |
| PT_FP_UNSAFE | The source and destination clocks belong to different clock domains or belong to the same clock Domain but the cycle is not synchronized, and some timing paths are false paths |

2. N*N grid

Here is an N*N grid based on the contents of the Clock Intersection data table:

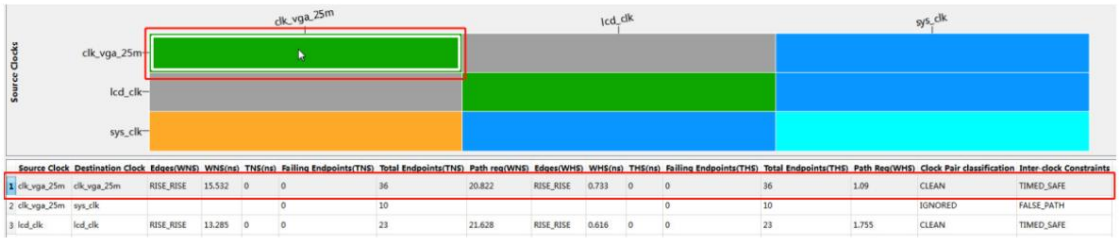
The direction is Source clocks, the horizontal direction is Destination Clocks, and different Inter-clock Constraints correspond to different

color, the default default color is as follows:

NOPATH
TIMED_SAFE
TIMED_UNSAFE
PT_FP_SAFE
PT_FP_UNSAFE
EXCLUSIVE_GROUPS
FALSE_PATH

Selecting a cell in the grid will display the corresponding row in the data table; selecting a row in the data table will display the corresponding row in the data table.

The corresponding squares are marked with white borders.



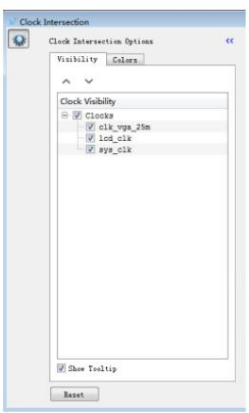

| | Source Clock | Destination Clock | Edges(WNS) | WNS(ns) | TNS(ns) | Failing Endpoints(TNS) | Total Endpoints(TNS) | Path req(WNS) | Edges(WHS) | WHS(ns) | THS(ns) | Failing Endpoints(THS) | Total Endpoints(THS) | Path Req(WHS) | Clock Pair classification | Inter clock Constraints |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | clk_vga_25m | clk_vga_25m | RISE_RISE | 15.532 | 0 | 0 | 36 | 20.822 | RISE_RISE | 0.733 | 0 | 0 | 36 | 1.09 | CLEAN | TIMED_SAFE |
| 2 | clk_vga_25m | sys_clk | | | | 0 | 10 | | | | | 0 | 10 | | IGNORED | FALSE_PATH |
| 3 | lcd_clk | lcd_clk | RISE_RISE | 13.285 | 0 | 0 | 23 | 21.628 | RISE_RISE | 0.616 | 0 | 0 | 23 | 1.755 | CLEAN | TIMED_SAFE |

3. Clock Intersection Options

Click the button in the figure to open the Clock Intersecton Options: The Visibility column can be used for any

Check or uncheck the box before clk to decide to show or hide the clk; the Colors column can be used for any Inter

The color of clock Constraints is modified.




# 5.8.6 Clock Tree Report

The Clock **Tree** Report is mainly used for delay analysis and auxiliary debugging of the clock tree. report meeting

Give the detailed clock network topology and the maximum clock skew for the entire clock. Among them, the clock network topology

Including the connection relationship and physical properties of any pins (position, network delay, etc.), the properties of terminal pins according to the precursor path

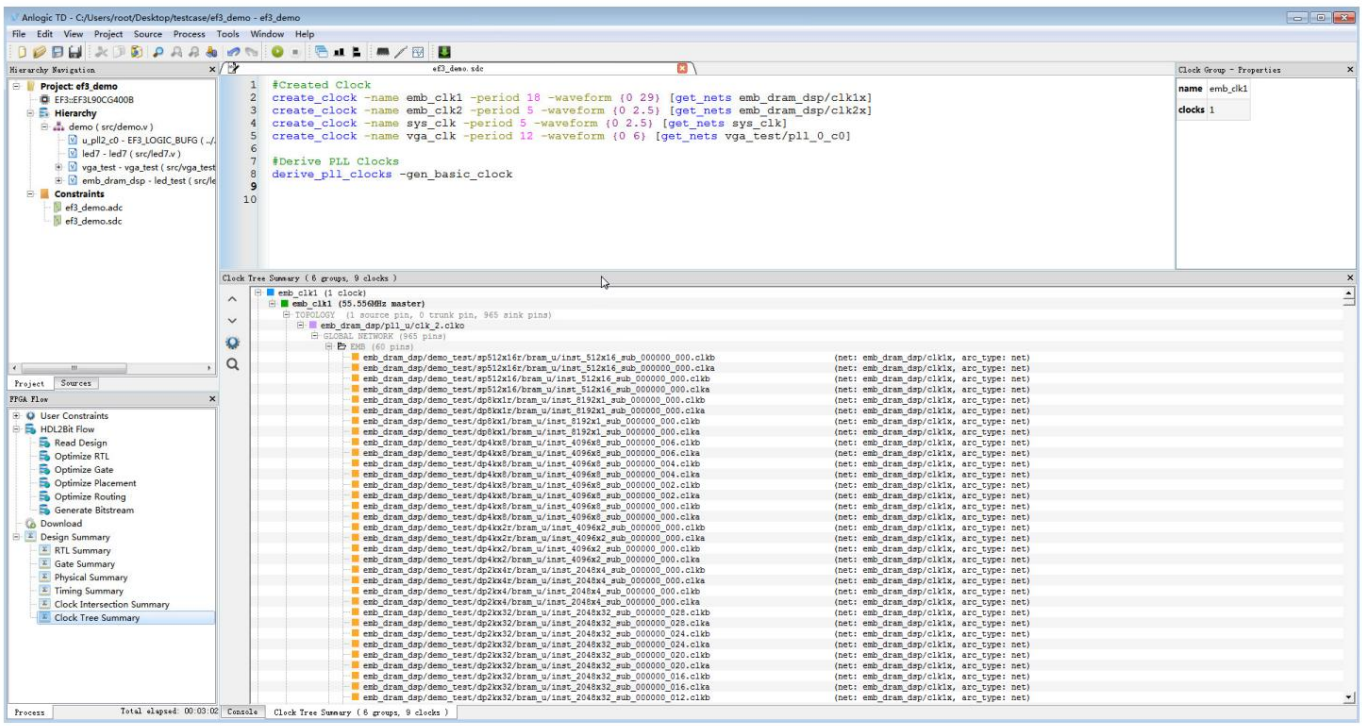
Classification (going global clock resources and common interconnection resources) and clock terminals are classified according to cell attributes (such as slice, bram,

dsp, etc.); the maximum clock deviation accurately shows the absolute value of the deviation and the corresponding pin set, where the calculation of the deviation value

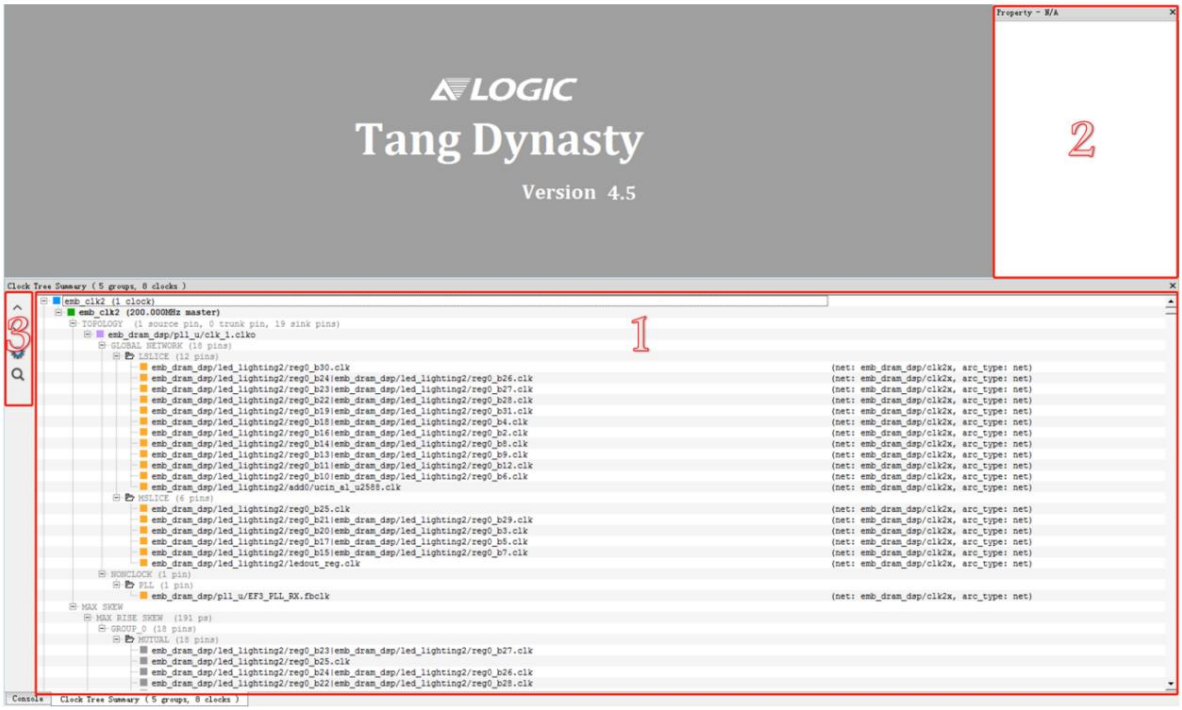
The process removes common path pessimism.

When the HDL2Bit Flow is completed, expand **Design Summary ÿ Timing Summary, double-click to**

Open the **Clock Tree** Report to open and view the Clock Tree. The main interface is as follows:



The interface of Clock Tree Summary mainly includes the following parts:




1. Clock Tree Summary

The Clock Tree Summary is listed in order with the clock domain report as the basic unit, and the clock domain report is based on the clock report as

The basic units are listed in order. The number of clocks and clock domains is consistent with the clock constraints in sdc.

a) Clock Tree Summary

The top of the Clock Tree Summary shows the total number of clock domains and clocks included in the entire timing constraint.

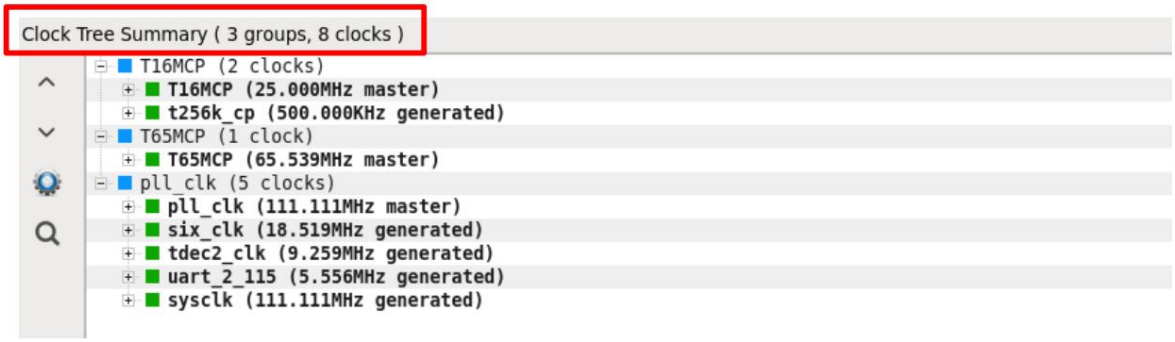
```
create_clock -name T65MCP -period 15.258  [get_ports {T65MCPI*}]


create_clock -name pll_clk -period 9 [get_nets {sys_clock/pll0/clkc[0]}]

create_generated_clock -name six_clk -source [get_nets {sys_clock/pll0/clkc[0]}] -divide_by 6 [get_nets {sub_uart/uart_nxbk/six_clk}]
create_generated_clock -name tdec2_clk -source [get_nets {sys_clock/pll0/clkc[0]}] -divide_by 12 [get_nets {sub_uart/uart_nxbk/tdec2_clk}]
create_generated_clock -name uart_2_115 -source [get_nets {sys_clock/pll0/clkc[0]}] -divide_by 20 [get_nets {sub_uart/uart_nxbk/uart_2_115}]
create_generated_clock -name sysclk -source [get_nets {sys_clock/pll0/clkc[0]}] -divide_by 1 [get_nets {sysclk}]

create_clock -name T16MCP -period 40 [get_nets {sys_clock/T16MCP}]
create_generated_clock -name t256k_cp -source [get_nets {sys_clock/T16MCP}] -divide_by 50 [get_nets {t256k_cp}]

set_false_path  -from [get_clocks {sysclk}] -to [get_clocks {t256k_cp}]
set_false_path  -from [get_clocks {sysclk}] -to [get_clocks {T16MCP}]
```



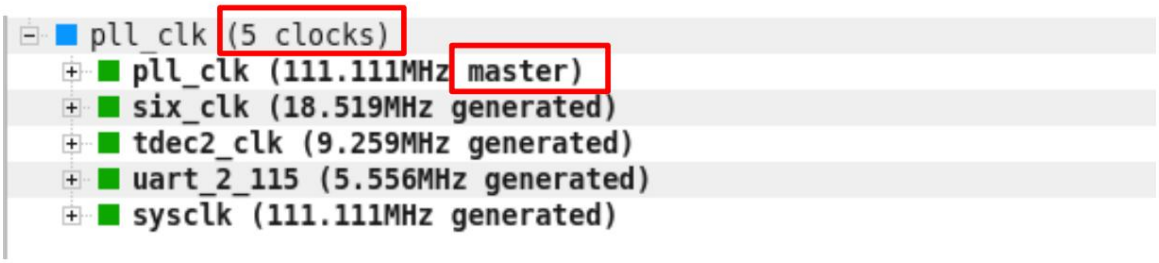
There are three master clocks defined in the sdc file in the figure: T65MCP, pll_clk, T16MCP. Clock Tree

The Summary is divided into 3 groups to display each clock domain and its internal clock information in turn.

b) Clock Group Summary

Each clock domain is divided into a primary clock and a number of derived clocks. The master clock is listed first in the clock domain report, after

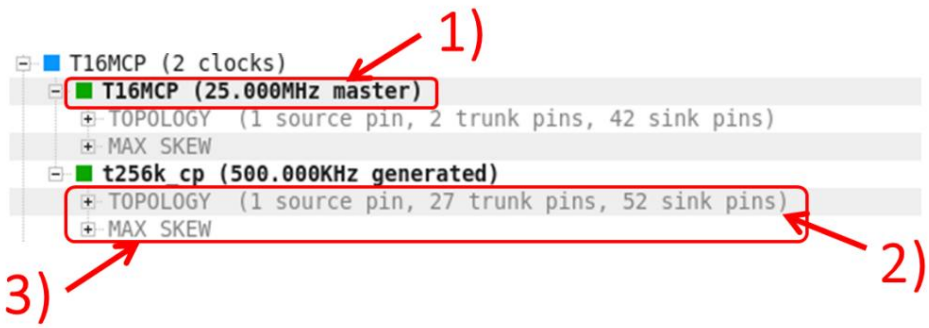
Continued in order to derive the clock. The top of each clock domain shows how many clocks are defined in the current clock domain.




c) Clock Summary

The single clock report is mainly divided into 3 parts:

1) Clock name, clock frequency, type




Among them, the light gray font in the single clock report is auxiliary information, which does not belong to the clock network itself. It is used for

Interpret and express some type of attribute or statistical information.

2) Clock network topology

The clock network topology is described with pins as the basic unit. Pin names are expressed as top model

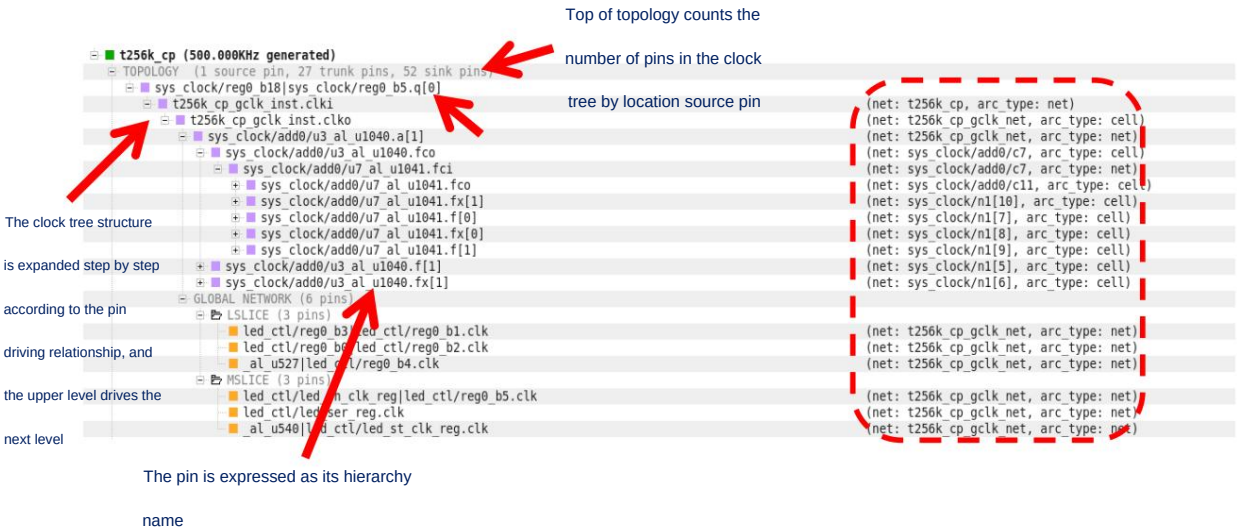
to the Hierarchy Name of the current pin.

The top summarizes the total number of pins in this clock tree. The pins are expanded level by level according to their level in the clock tree, and each pin is

The pin occupies a separate line, with two descriptions: the first paragraph: the name of the pin, the second paragraph: the simple attribute of the pin, that is, the location of the pin.

The timing arc type from the net, the superior pin to the current pin. If two pins belong to the same instance, then

The sequence arc type is 'cell', otherwise it is 'net'; lock the pin, right-click to get more detailed property information.




The clock terminal pins in the clock network topology are folded according to the classification method: there are two levels of folding, each level of folding

contains the number of pins in the properties.

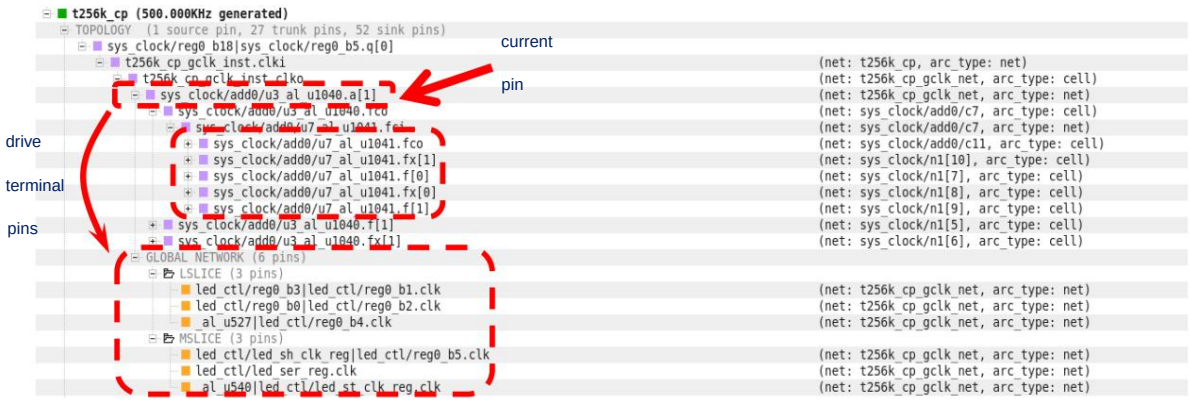
The first-level attributes include:

GLOBAL_NETWORK, indicating that the clock path takes the global clock interconnection;

LOCAL_NETWORK, indicating that the clock path is connected by ordinary interconnection, which is more common in clocks with larger skew;

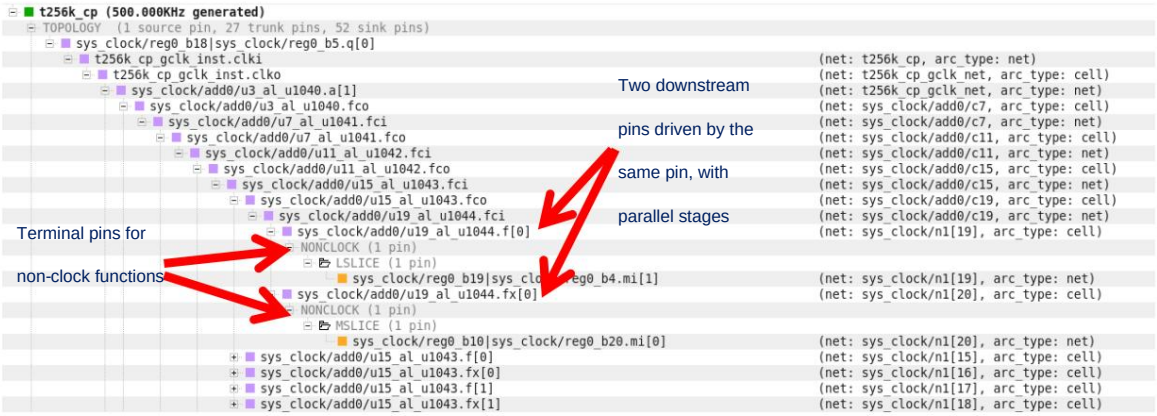
NONCLOCK, indicating that the terminal pin is not clocked.

The second-level attribute is the name of the logic unit, such as MSLICE/LSLICE/IOL/DSP and so on.




Multiple pins driven by the current pin may contain clock tree trunk pins and terminal pins: not grayed out

Pins folded in font are torso pins; pins folded in fonts explained in gray are terminal pins.



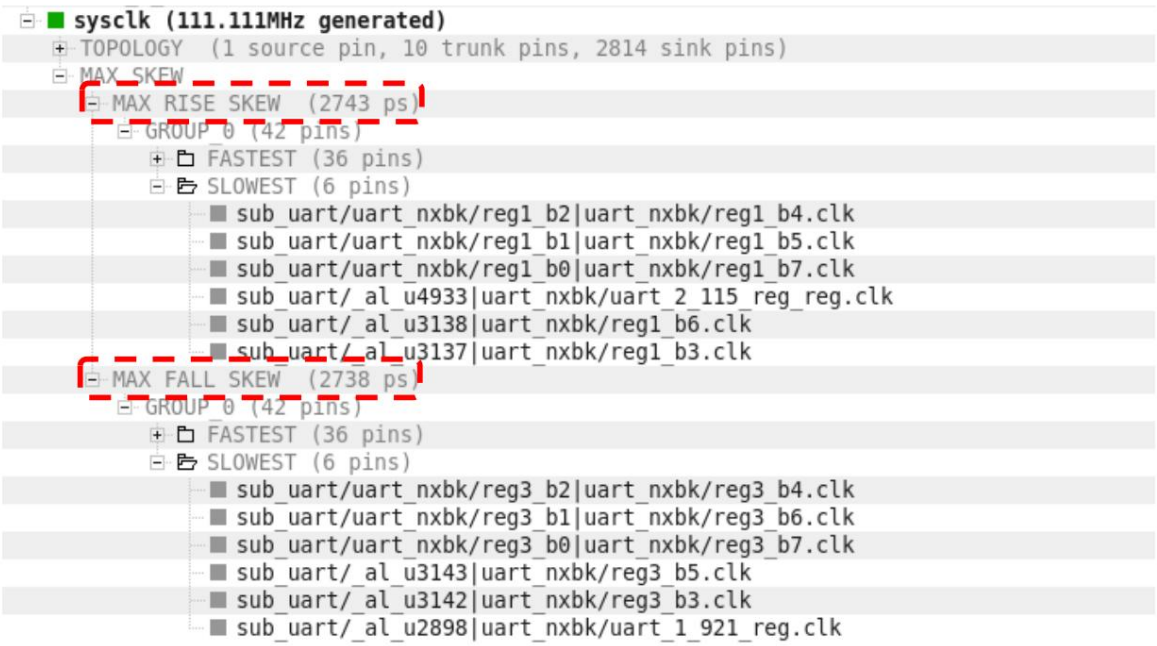

3) Maximum clock skew

The delay from the clock source to any terminal pin has a range: the best delay and the worst delay. compare the whole

The best delay pin and the worst delay pin of the clock network, the difference is the maximum clock skew.

Clock skew is divided into: rising edge clock skew and falling edge clock skew. Timing library delay data is divided into signal rise

Edge and falling edge delays, so the clock skew values of the rising and falling edges of the clock and the corresponding pin sets are calculated respectively.




Clock skew is described in groups: the largest skew must occur within a subtree of the entire clock tree. if the clock tree

There are two subtrees with the same deviation and the maximum deviation, so they need to be displayed in two groups. However, the maximum

Clock skew usually occurs within the 'subtree' at the root of the entire clock tree, so the maximum skew for most clocks is only

There is a group.

Pins within a set of deviations are described by class, the classes are:

ÿ FASTEST: The set of pins with the shortest delay starting from the common ancestor;

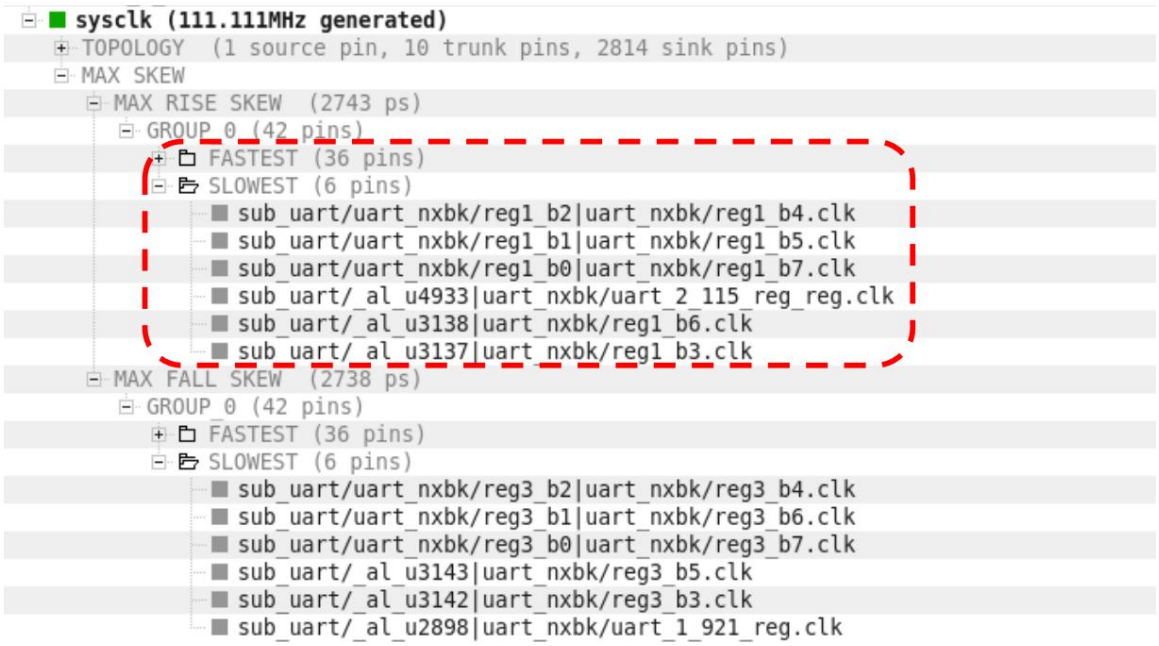
ÿ SLOWEST: The set of pins with the longest delay from the common ancestor;

ÿ MUTUAL: The longest and shortest extension from the common ancestor to all leaf pins of this subtree

time is the same. Some terminal pins have the longest delay and some terminal pins have a delay when the maximum clock skew occurs

Shortest, usually occurs with very balanced clock trees.

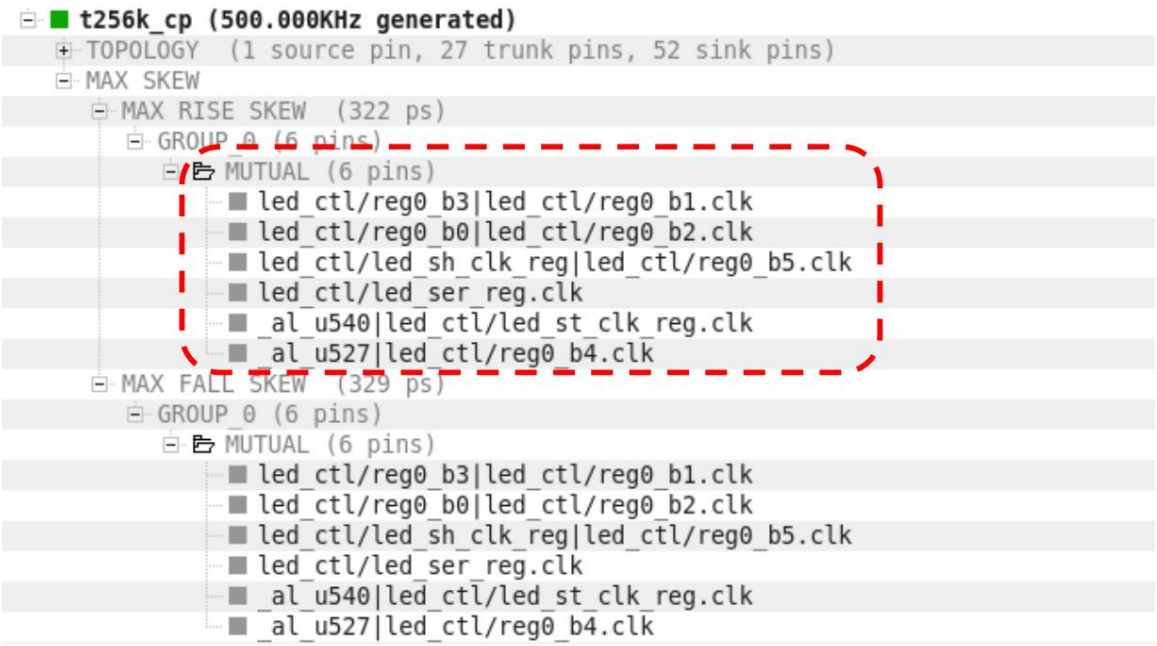
FASTEST/SLOWEST:




The maximum and minimum delay pin sets are shown separately. The pins here are only represented by the hier name, and the related attributes can be

Find in TOPOLOGY.

MUTUAL:




The calculation of the maximum deviation is to compare the difference between the min delay and max delay of each pin, and the set of pins with the largest difference

The combination is the maximum deviation pin set. When the set of pins with the largest difference is the same set, it indicates that the maximum deviation occurs.

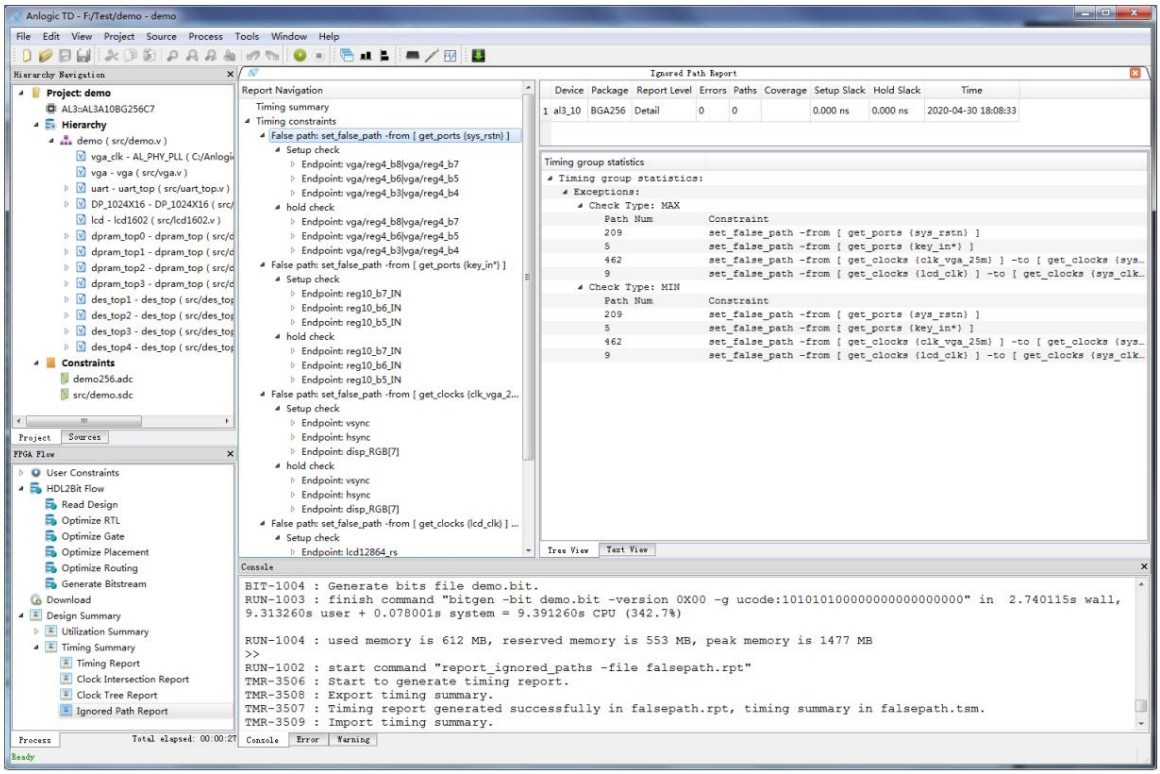
Between any two pins in this set (more common in fully balanced clock trees). Therefore, FASTEST and SLOWEST

The collection is the same collection, which only needs to be described once, using the MUTUAL keyword.

# 5.8.7 Ignored Path Report

When the HDL2Bit Flow is completed, expand **Design Summary ÿ Timing Summary, double-click to**

Open **Ignored Path Report** to view, the main interface is as follows:




The interface composition of Ignored Path Report is basically the same as that of Timing Report. For details, please refer to 5.8.4

Timing Report, the difference is that the Timing Group Statistics of the Ignored Path Report shows

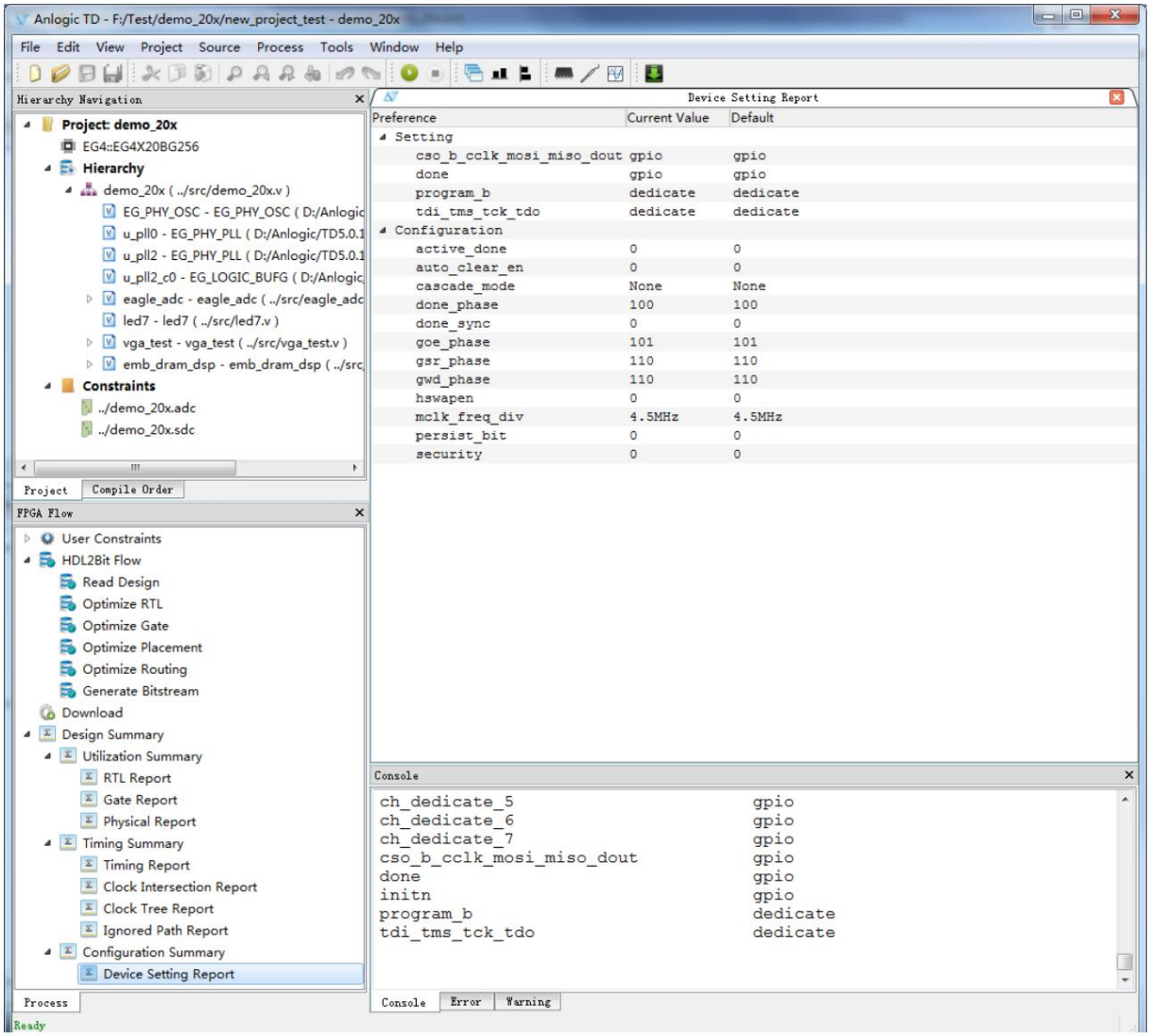
Information about set_false_path. Ignored Path Report if set_false_path is not set in SDC file

Display is empty.

# 5.8.8 Device Setting Report

Expand **Design Summary ÿ Configuration Summary, double-click to open Device Setting Report**

You can view it, the main interface is as follows:

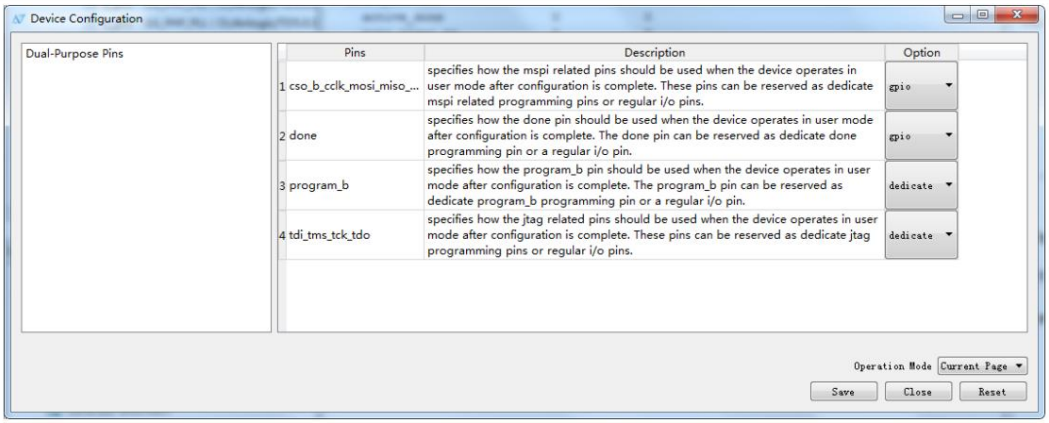



The interface of Device Setting Report is mainly divided into two parts: Setting and Configuration.

Setting displays the relevant information in the device option, and the column displays the Current corresponding to the option

Value and Default. When one or some of the options are changed and the settings are saved, the Current Value

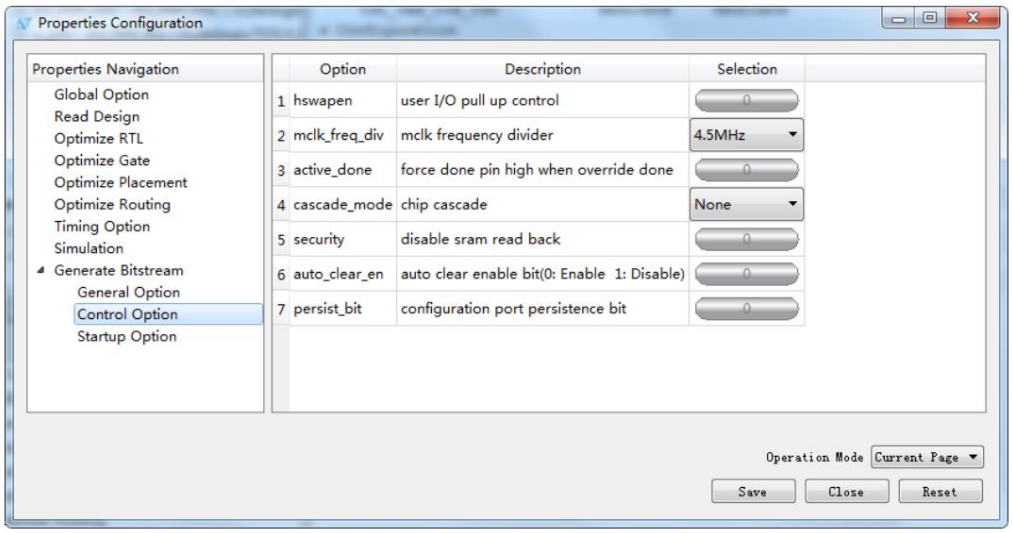
The option value of the corresponding setting will be displayed.

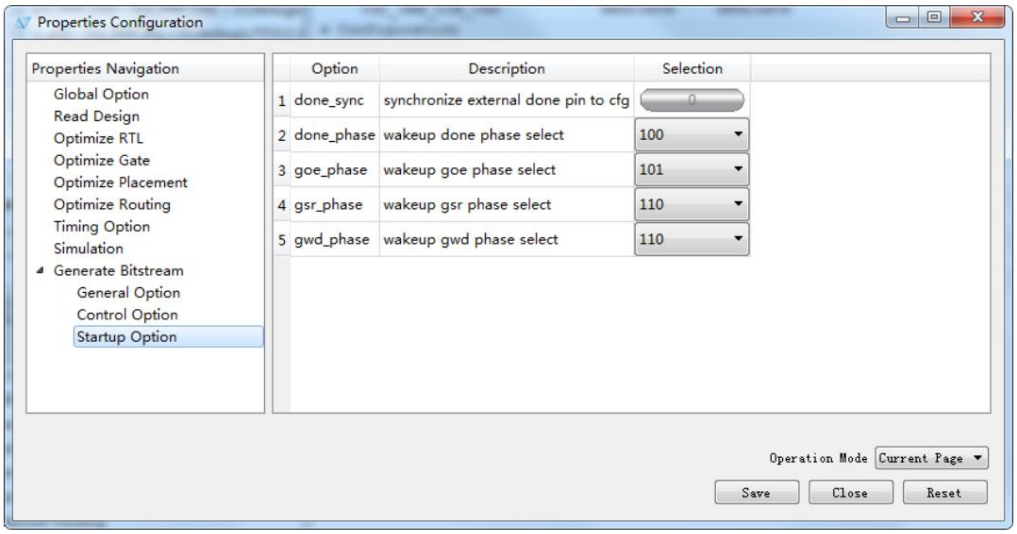



Configuration shows Control Option and Startup in Properties ÿ Generate Bitstream

For the relevant information of Option, the columns display the Current Value and Default of the corresponding option. when one of

After one or some options have been changed and the settings are saved, Current Value will display the option value of the corresponding setting.



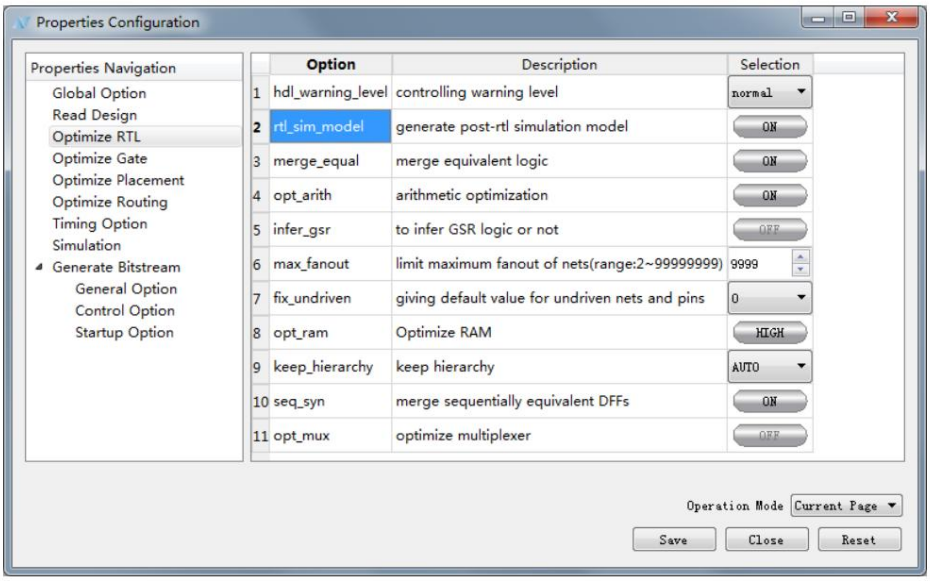
# 6 Functional Simulation

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

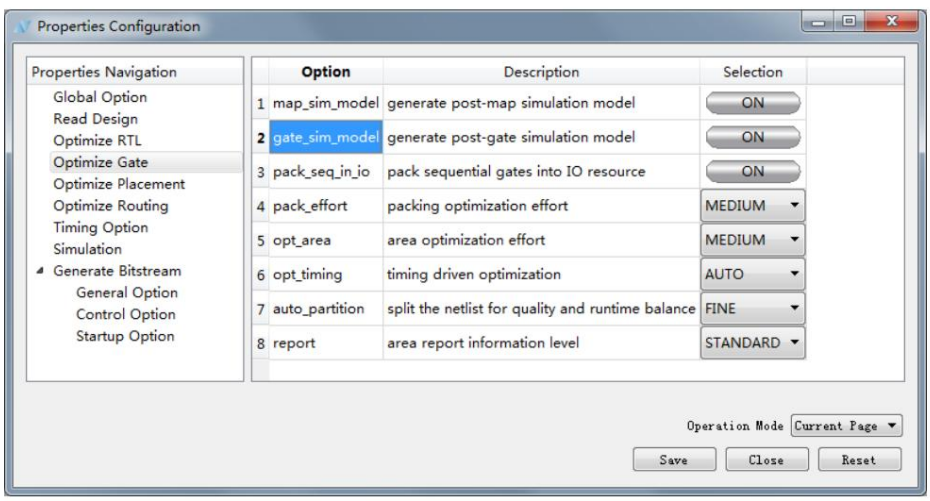
This chapter mainly introduces the process of generating files required for Modelsim simulation in TD software.

1. Before running HDL2Bit Flow, set relevant parameters.

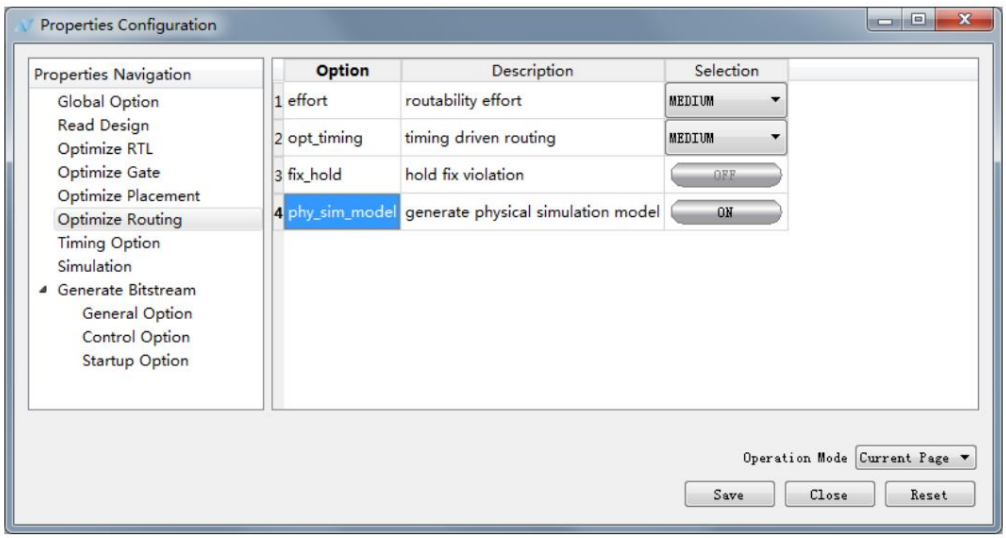
**Process ÿ Properties ÿ Optimize** RTL: set rtl_sim_model ON.



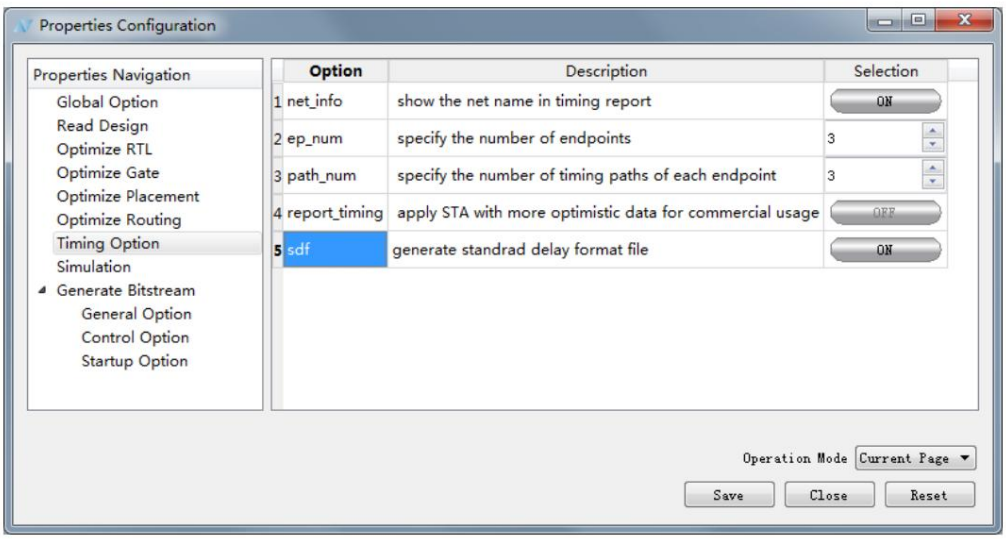

**Process ÿ Properties ÿ Optimize Gate:** set gate_sim_model ON.



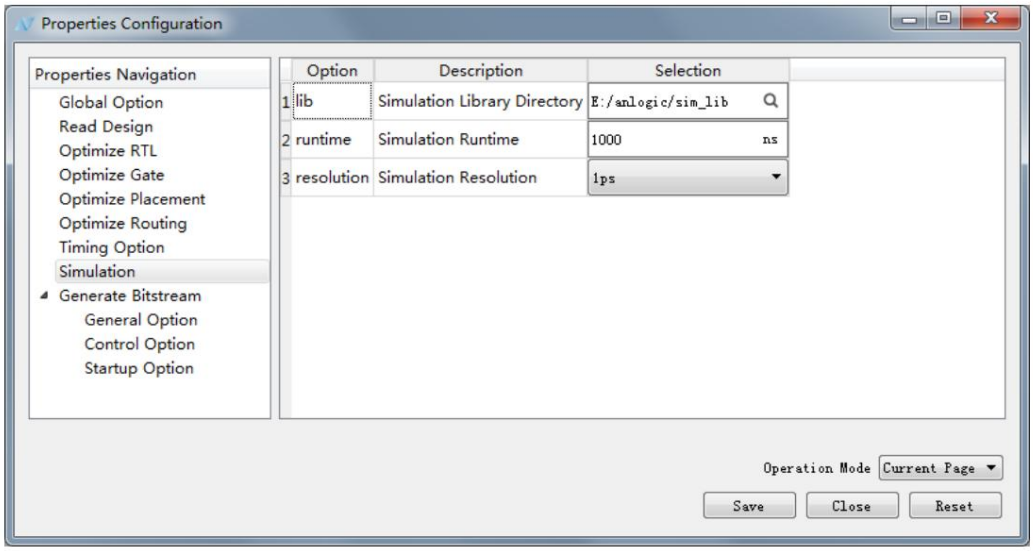
**Process ÿ Properties ÿ Optimize Routing: set** phy_sim_model ON.




**Process ÿ Properties ÿ Timing Option: set** sdf ON.



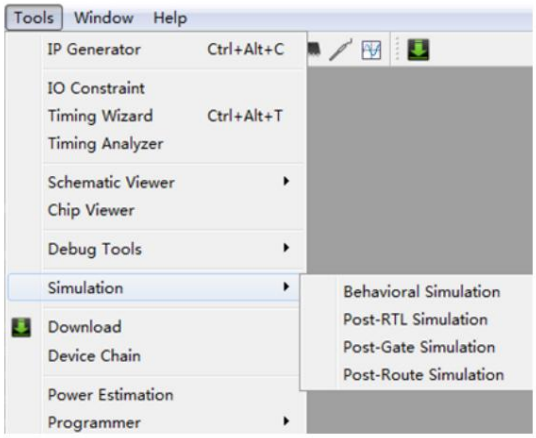

2. Set Modelsim simulation related parameters




| Property | Comments | Default |
|---|---|---|
| lib | Specifies the library file for simulation | There is no default value, you need to specify the path manually |
| runtime | Specify when the simulation runs | 1000ns |
| resolution resolution | Specifies the time precision of the simulation | 1 ps |

3. Run HDL2Bit Flow

4. Run **Tools ÿ Simulation**




When HDL2Bit Flow runs to the Read Design step, Behavioral Simulation can be executed;

When HDL2Bit Flow runs to the Optimize RTL step, Post-RTL Simulation can be executed;

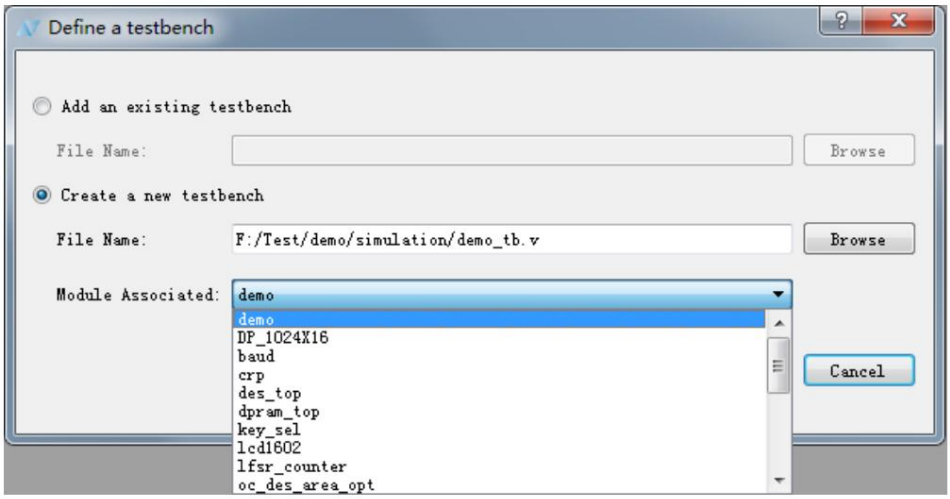
When HDL2Bit Flow runs to the Optimize Gate step, Post-Gate Simulation can be executed;

When HDL2Bit Flow runs to the Optimize Routing step, Post-Route Simulation can be executed.

5. Define the testbench file

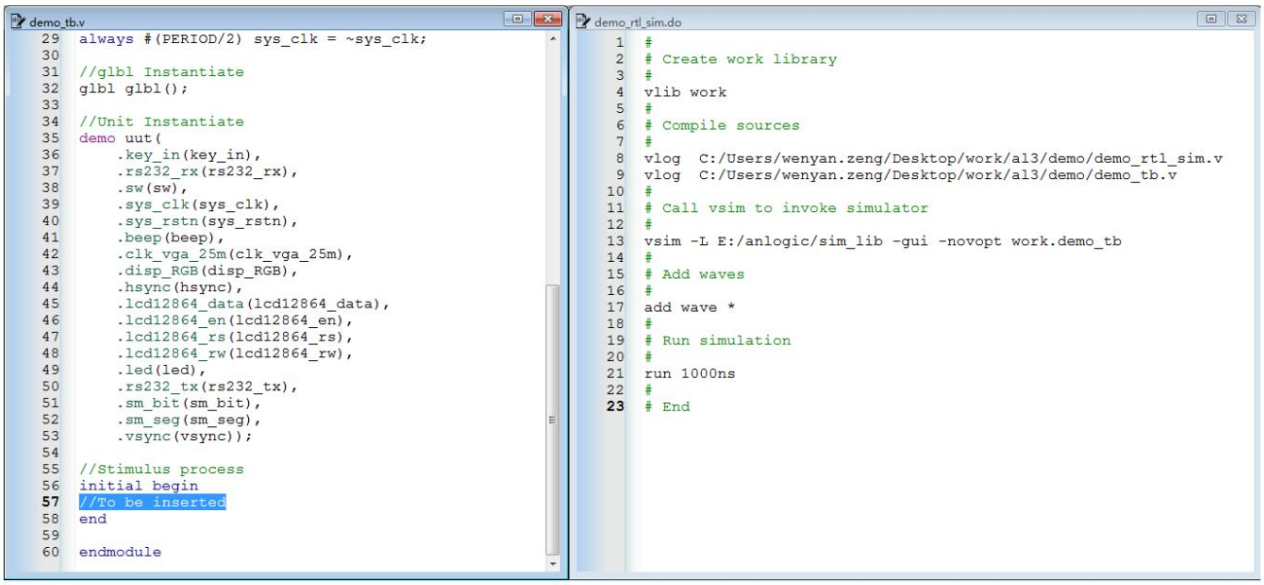
If you click Post-RTL Simulation, the following dialog box will pop up, you can add an existing one

testbench file, you can also 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 displayed on the TD interface

Open these files. Note that there is no incentive in prj_tb.v, it needs to be filled in manually before simulation.




For the specific simulation process in Modelsim, please refer to 9.4 Modelsim Simulation Process in this manual.

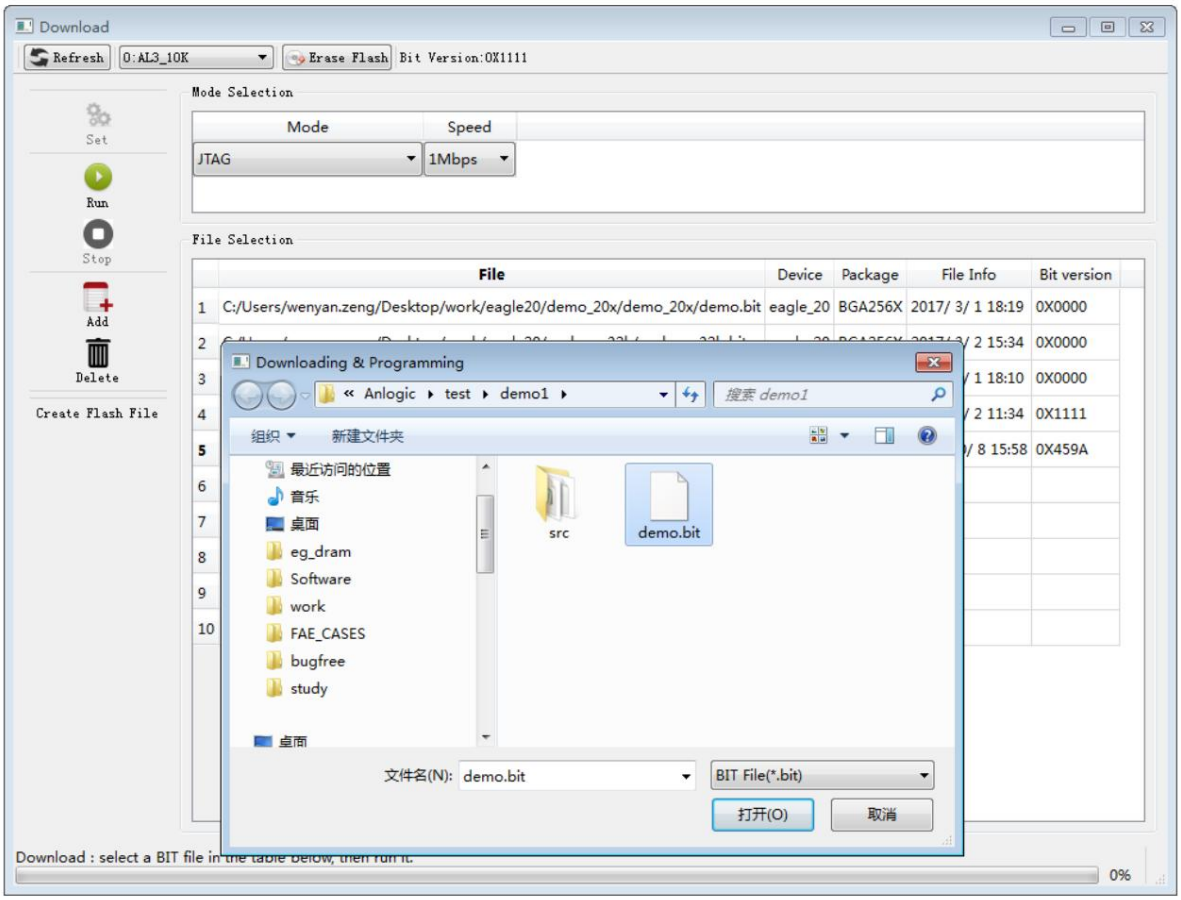
# 7 downloads

## 7.1 Introduction to the download process

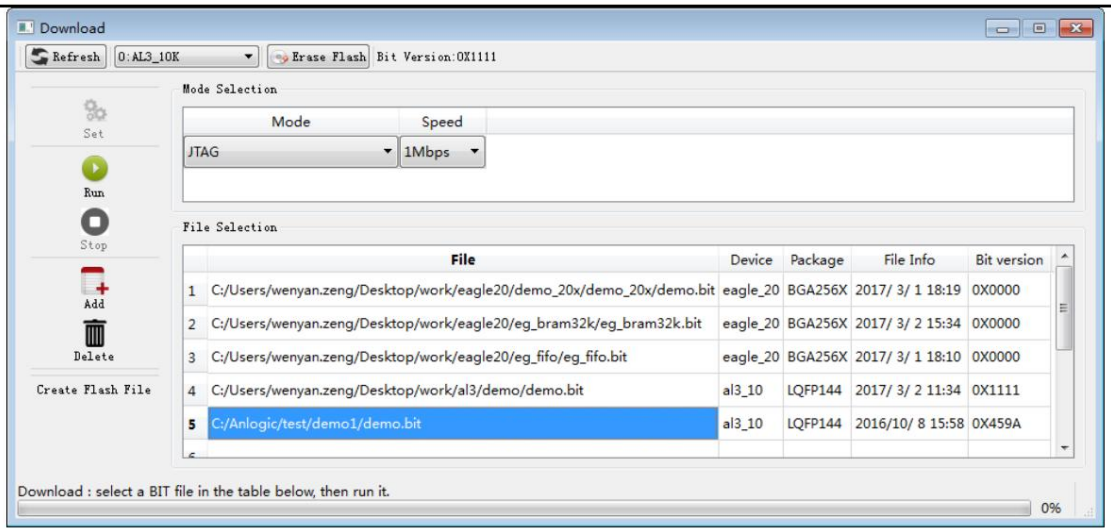
After the bitstream files are successfully generated, they can be loaded into the FPGA chip's configuration memory or SPI Flash in memory.

1. In the FPGA Flow panel, double-click **Download**

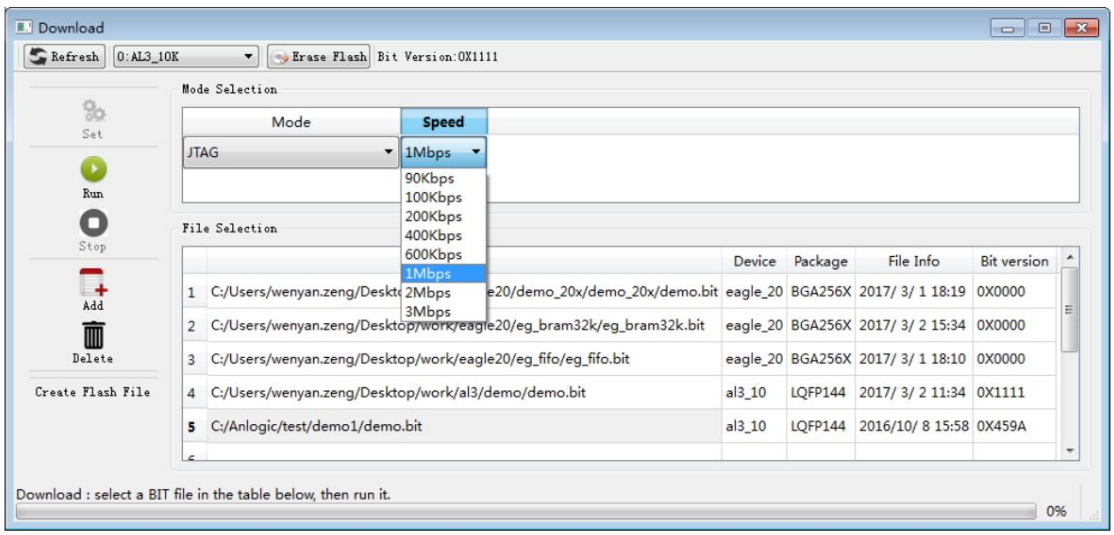
2. Add the bitstream file to be downloaded through **Add .**



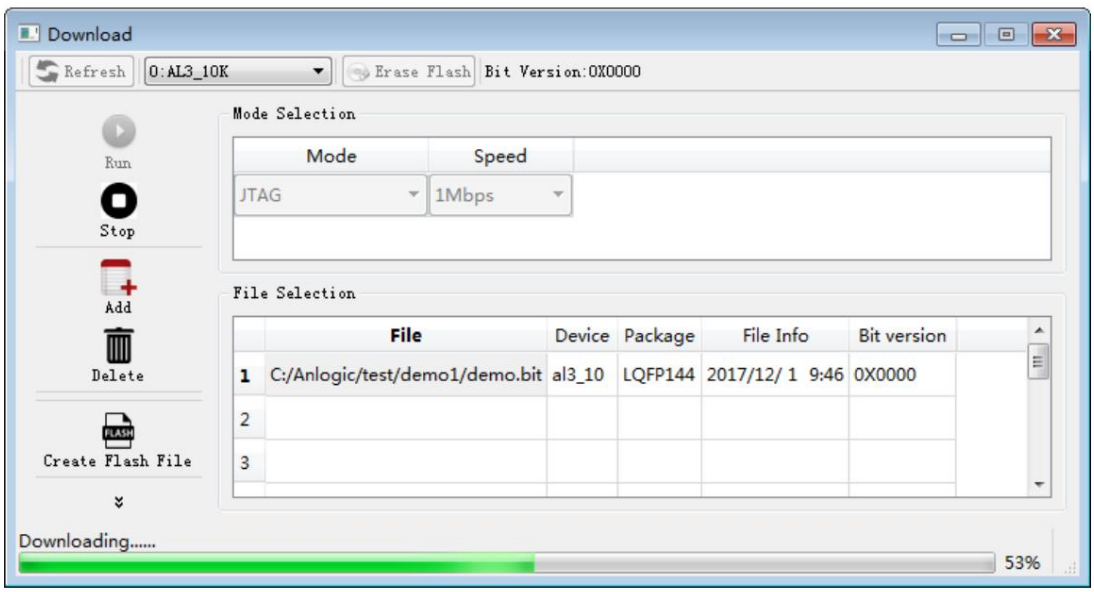

3. Select the corresponding bitstream file and click **Run** to download.



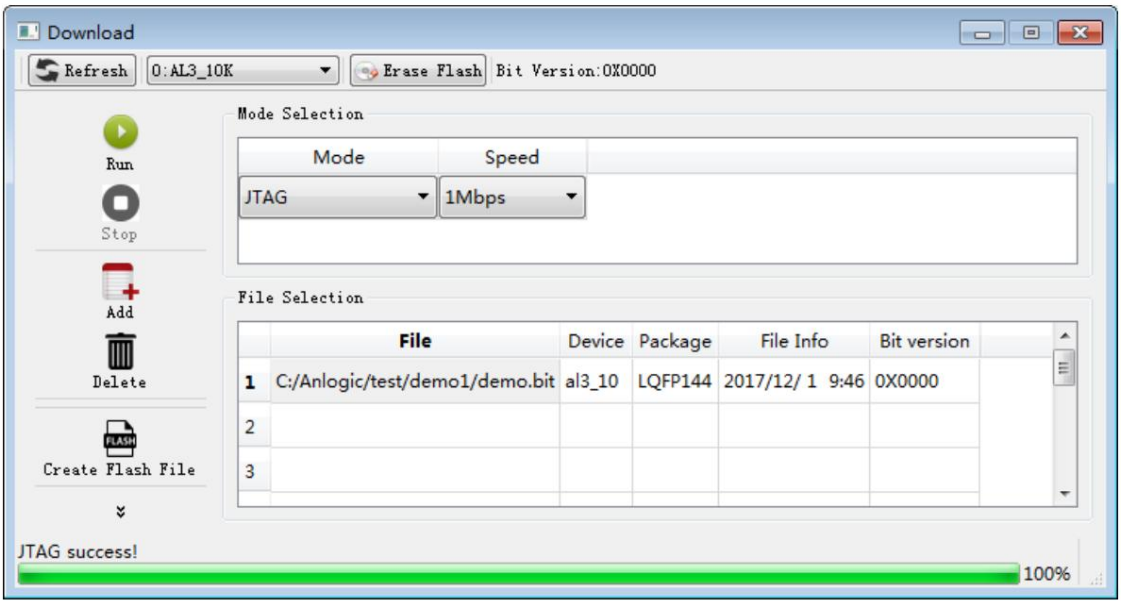

The download speed is divided into nine levels, 90Kbps is the slowest, 3Mbps is the fastest, and the default is 1Mbps.




During the download process, you can view the download progress through the progress bar.



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




**TD** can't recognize the chip:

1. "No hardware": The user did not install the USB driver correctly before downloading, and the USB download driver installation said

   Please refer to Appendix 9.5 for details. When downloading, each interface is not connected correctly, please check whether each interface is loose,

   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 the

   Check whether the power of the circuit board is turned on, and then click the **Refresh** button to refresh.

Bitstream file check:

In order to ensure the safety of customer files and equipment, the TD software will check the chip ID matching during the download process.

And a CRC check for bitstream file integrity and correctness:

1. When the chip ID recorded in the bitstream file does not match the download target chip, the TD software will report an error and terminate the download.

   load;

2. When the bit stream file is incomplete or the content has been tampered with, the CRC check will not pass, and the download cannot be continued.

   load.

## 7.2 Bitstream File Type

The bit stream files, generation methods and download operations supported for downloading in the TD software are as follows:

1. bit: bit file contains complete chip configuration and bit stream information.

   Run Generate Bitstream in the TD interface to generate a bit file by default.

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

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

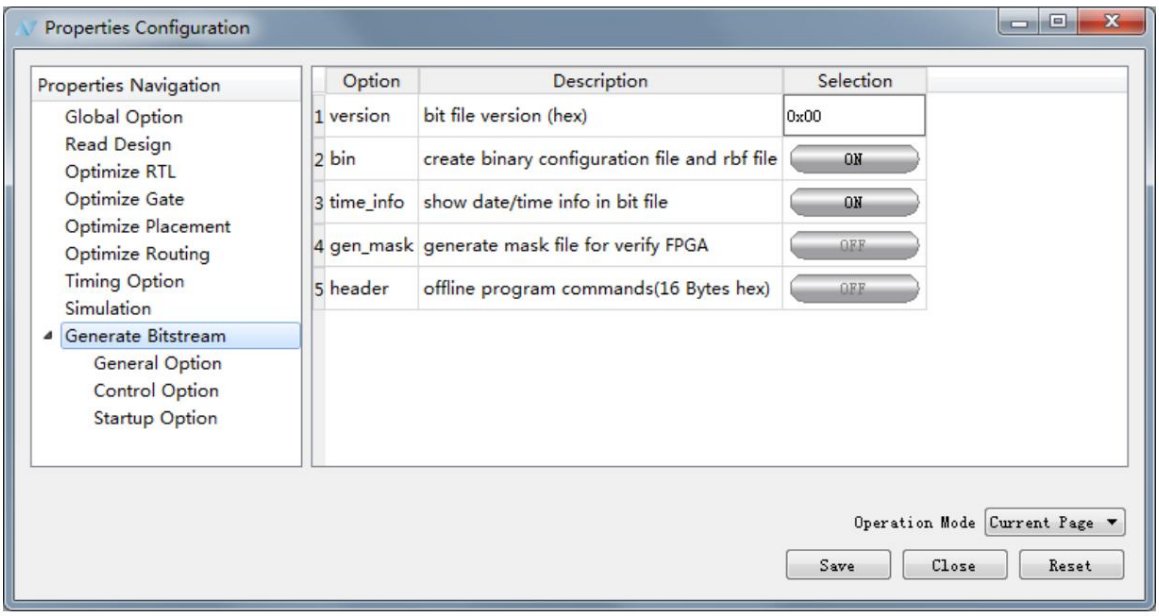
   Set the value of the bin option in **ProcessÿPropertiesÿGenerate BitstreamÿGeneral Option**

   Set it to ON and save it. After running Generate Bitstream, the corresponding bin will be generated in the project directory

   document.

   The bin file can be downloaded by the offline downloader, and the supported download modes are Direct Flash Write, Program

   SPIBIN.




3. svf: Serial vector file. Uniform standard structure provided for shielding internal details.

   The generation and download of svf files will be described in detail in 7.6 Device Chain.

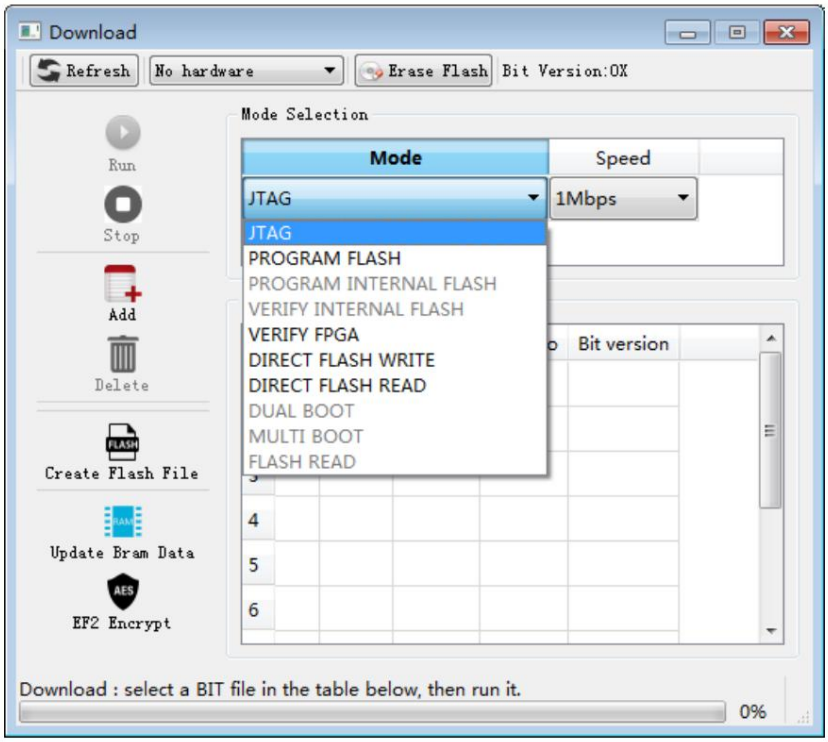
## 7.3 Download Mode

TD provides the following download modes for users to choose: JTAG, PROGRAM **FLASH, PROGRAM**

**INTERNAL FLASH, VERIFY INTERNAL FLASH, VERIFY FPGA, DIRECT FLASH**

**WRITE, DIRECT FLASH** READ, DUAL **BOOT, MULTI BOOT, FLASH READ,**

**PROGRAM SPIBIN.**




1. **JTAG** mode: The downloaded bit file will not be saved in the flash, and the configuration bit information will be stored directly

The programming switch is controlled in the FPGA chip, and the configuration bit information is completely lost when the circuit board is powered off.

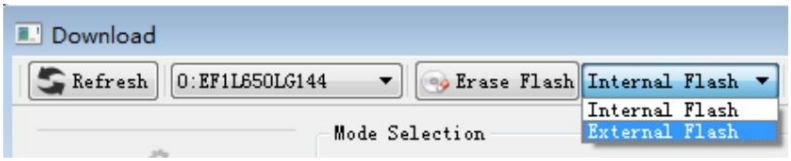
2. **PROGRAM FLASH** mode: The bit file will be saved to the external Flash chip, and the circuit board will be damaged.

After the power is restarted, the FPGA chip automatically reads the bit stream information stored in the Flash chip. To erase flash

Click the **Erase Flash** button for the bit stream information in the Flash chip, and the erasing time depends on the device parameters of the Flash chip.

For ELF series devices, this function is used for external FLASH download. When erasing external FLASH,

**Erase External Flash** needs to be selected .




3. **PROGRAM INTERNAL FLASH** mode: only supports ELF series devices, used for internal

download of flash. When erasing the internal FLASH, select **Erase Internal Flash.**

4. **VERIFY INTERNAL FLASH** mode: only supports ELF series devices, used to compare internal

Whether the configuration file in flash is consistent with the information in the bit file currently selected by the user.

5. **VERIFY FPGA** mode: used to compare the configuration bit information in the FPGA chip with the one currently selected by the user

Whether the information in the bit file is consistent, it is best to use it together with the mask file (.bmk) to ensure that the bit stream file and the

The mask files are in the same folder.

6. **DIRECT FLASH WRITE** mode: directly write data to FLASH without going through FPGA

In the address area, the downloader needs to be directly connected to the FLASH signal line on the hardware. for offline downloader

, only supports downloading bin files.

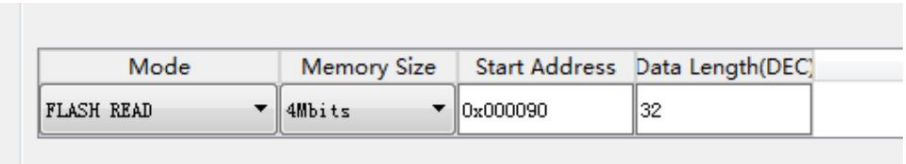
7. **DIRECT FLASH READ** mode: directly read the specified area from FLASH without going through FPGA

The data is stored in the specified file. This mode also requires the downloader to be directly connected to the FLASH signal line.

even.

8. **FLASH READ** mode: read back FLASH content, output in the specified file, can specify the start of readback

address and readback content length.




9. **PROGRAM SPIBIN** mode: Download bin file to FLASH. Only supports downloading bin files, you can

Specify the download start address.

## 7.3.1 Dual Boot

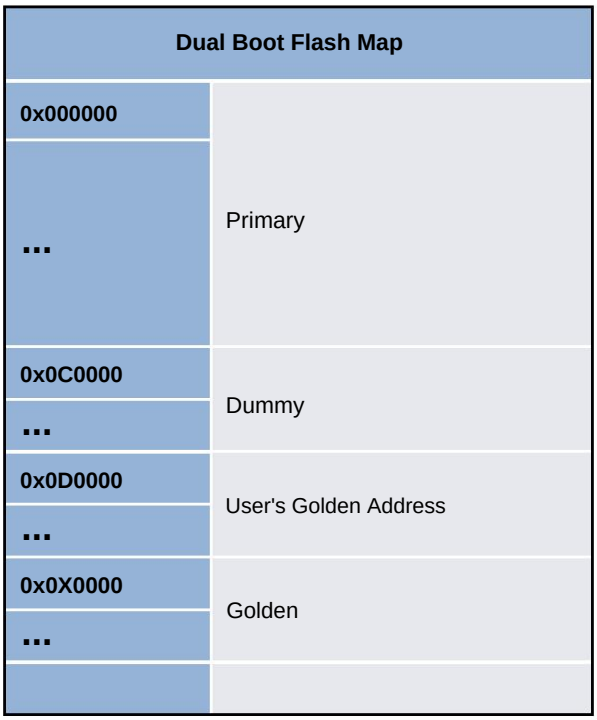
For Eagle and EF2 series FPGAs, it supports Dual Boot when loading programs from external flash.

(dual-boot mode) and Multi Boot (multi-boot mode).

The dual-boot mode means that two sets of FPGA bit streams are stored in the SPI FLASH. After power-on, the FPGA is loaded first.

Primary bit stream, if the primary bit stream error causes loading failure, it will be based on the Golden Address (jump location

address) to load the Golden bitstream. In dual-boot mode, the data space distribution is shown in the following figure:



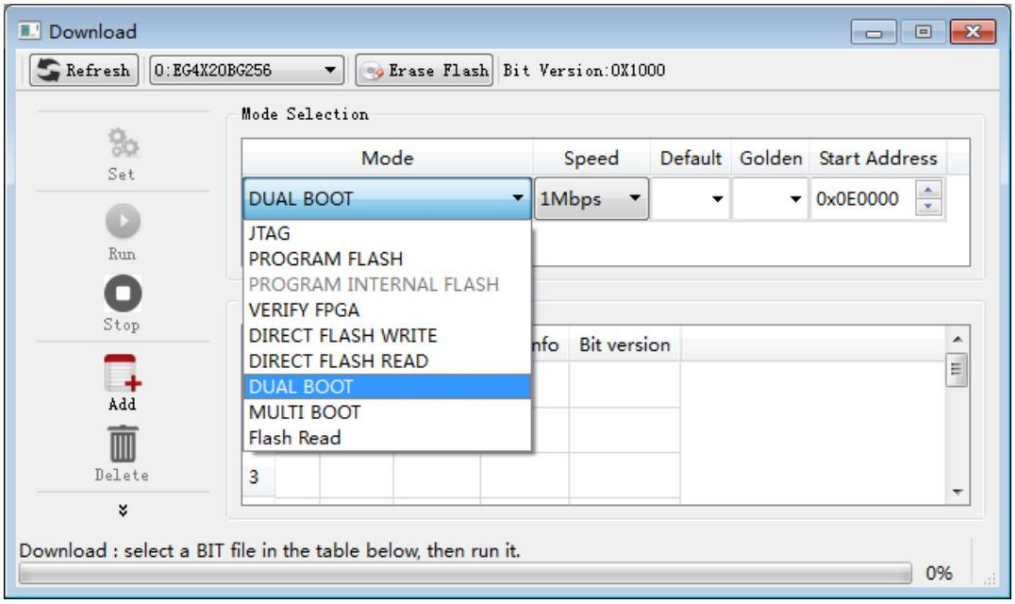

Eagle series FPGA supports dual-boot mode by default, that is, it will default from FLASH after power-on.

0 address loader, if the bit stream starting at address 0 is corrupted and the FPGA program fails to load, the FPGA will

Read the jump address to 0X0D0000 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 the two bitstream files that need to be downloaded to FLASH through the **Add** button;