Gowin Design Physical Constraints

User Guide

SUG935-1.1E, 09/04/2020

Disclaimer

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

Revision History

Date	Version	Description
05/09/2020	1.0E	Initial version published.
09/04/2020	1,1E	● FloorPlanner menu bar optimized; ● Back-annotate Physical Constraints supported.

Contents

List of Figures

List of Tables

1 About This Guide

1.1 Purpose

This guide describes Gowin FloorPlanner. It introduces the interface and syntax definition of FloorPlanner in order to help users add physical constraints to their design. The software screenshots and the supported products list in this guide are based on 1.9.7Beta. As the software is subject to change without notice, some information may not remain relevant and may need to be adjusted according to the software that is in use.

1.2 Related Documents

The latest user guides are available on GOWINSEMI Website: www.gowinsemi.com. You can find the related document: SUG100, Gowin Software User Guide.

1.3 Terminology and Abbreviations

Table 1-1 shows the abbreviations and terminology that are used in this manual.

Table 1-1 Terminology and Abbreviations

Terminology and Abbreviations	Meaning
FPGA	Field Programmable Gate Array
I/O	Input/Output
SIP	System in Package
Chip Array	Chip Array
Package View	Package View
VREF	Voltage Reference

1.4 Support and Feedback

Gowin Semiconductor provides customers with comprehensive technical support. If you have any questions, comments, or suggestions, please feel free to contact us directly by the following ways.

Website: www.gowinsemi.com

E-mail:support@gowinsemi.com

2 Introduction

Gowin FloorPlanner is designed in-house by Gowin and used for place, physical constraints and timing optimization. It supports the read and write of I/O, Primitive, block (B-SRAM and DSP), and Group, etc. It also supports the generation of place and constraints files according to users configuration. The files define I/O attributes, primitives and locations, etc.

The features of Gowin FloorPlanner are as follows:

- Supports the input of user design files & constraints files, and the output of constraints files;
- Supports the display of IO Port, Primitive and Group constraints in user design files;
- Supports to create, edit and modify constraints files;
- Supports grid mode, macro cell mode and primitive mode of chip array;
- Supports Package View;
- Supports to display Chip Array and Package View synchronously;
- Supports real-time display and differences display of constraints locations;
- Supports to generate locations by dragging;
- Supports to generate constraints by One Hit Drag;
- Supports IO port configuration and batch configuration;
- Supports the display and editing function of Clock Assignment;
- Supports the check of constraints legality;
- Supports Back-annotate Physical Constraints;
- Supports timing optimization.

3 Edit Physical Constraints

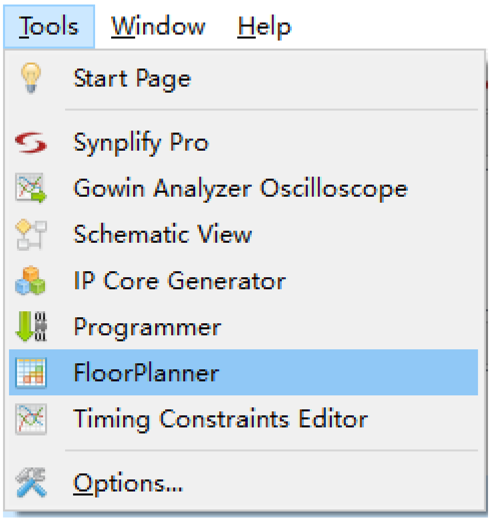
Gowin FloorPlanner can create and edit physical constraint files. It supports tabulated constraints editing and efficient netlist lookup so as to improve the efficiency of writing physical constraints files.

3.1 Start FloorPlanner

There are two ways to start FloorPlanner:

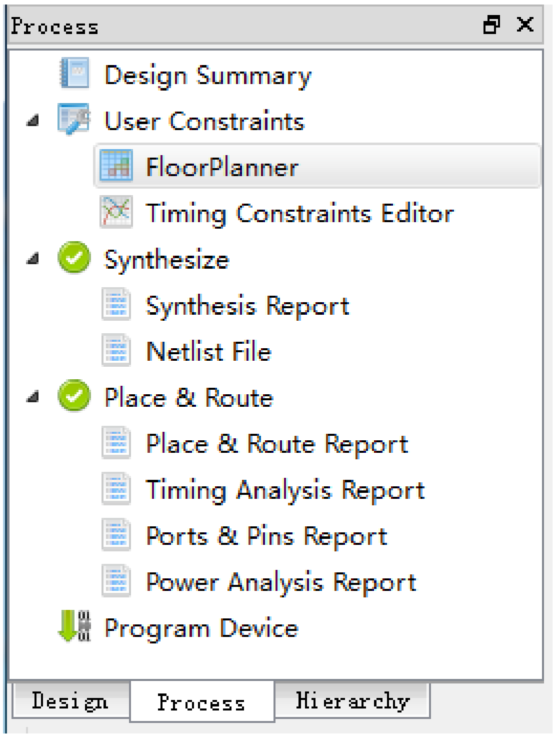
1. Click "IDE > Tools" to open "FloorPlanner", as shown in Figure 3-1.

Figure 3-1 Start FloorPlanner



2. After synthesis, Double-click "FloorPlanner" in Process view, as shown in Figure 3-2.

Figure 3-2 Start FloorPlanner in Process View



Note!

- If you use Gowin FloorPlanner for constraints, the netlist file should be added first;
- When you choose the first way to start Gowin FloorPlanner, the netlist file is needed to be loaded via "File > new".
- When you choose the second way to start Gowin FloorPlanner, the netlist file will be loaded automatically.

3.2 Create and Open Constraints File

If the project requires IO constraints and primitive locations constraints, you can create physical constraints files in the following two ways:

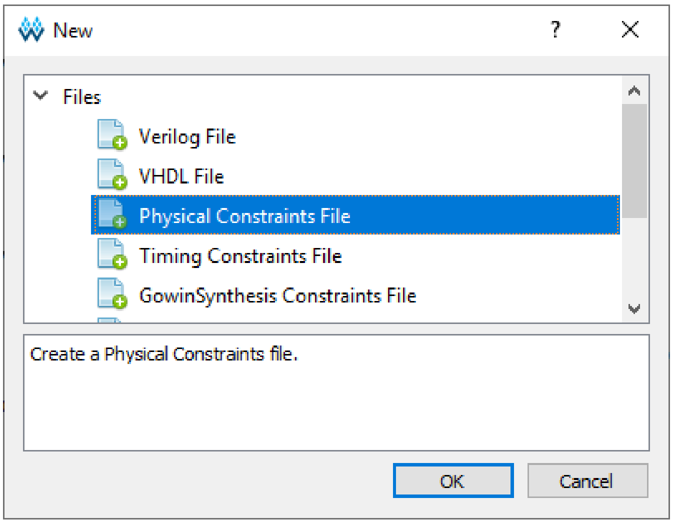
- Manually write;
- Output the constraints file by Gowin FloorPlanner.

3.2.1 Create Constraints File

Steps of creating constraint files are as follows:

1. Click "File > New" to open the "New" dialog box;
2. Select "Physical Constraints File", as shown in Figure 3-3.

Figure 3-3 New Physical Constraints



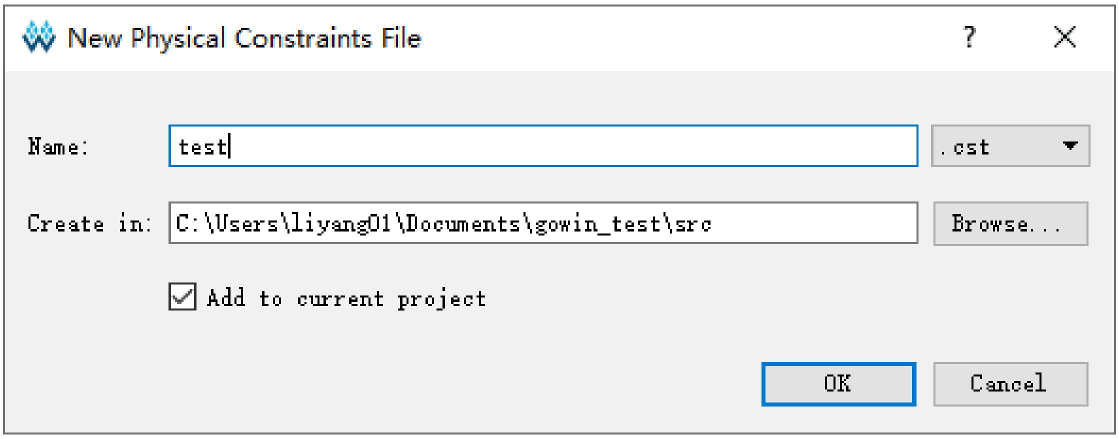
Create a Physical Constraints file.

Note!

You can also open "New" in the following two ways:

- Using the "Ctrl+N" short cut;
- Click the "New" icon in the toolbar;
- Click "OK", as shown in Figure 3-4.

Figure 3-4 Create Physical Constraints File



Name: test .cst

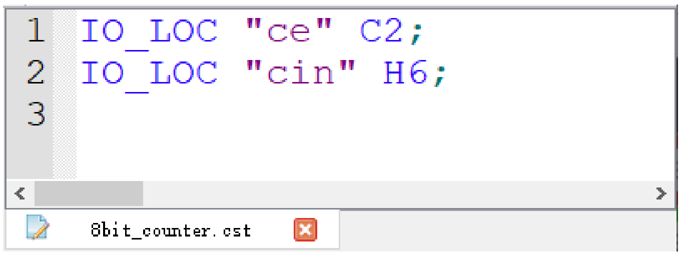
Create in: C:\Users\liyang01\Documents\gowin_test\src

☑ Add to current project

- Name: The physical constraints file name with the suffix of .cst and .ucf;
- Create in: Select the constraints file path by "Browse" and it is saved in src folder by default;
- Add to current project: Add the constraints file to the project automatically if this option is checked.

After finished, users can write constraints files according to the syntax rule of Gowin FloorPlanner as shown in Figure 3-5.

Figure 3-5 Manually Write Physical Constraints File



```
IO_LOC "ce" C2;
IO_LOC "cin" H6;
```

3.2.2 FloorPlanner Output Constraint Files

Gowin FloorPlanner can output the new physical constraints file and the modified physical constraints file. The steps are as follows:

Start FloorPlanner according to the description in 3.1 Start FloorPlanner.

1. Click "File> New…" to open "New" dialog, as shown in Figure 3-6.

Note!

You can also open "New" in the following two ways:

- Using the "Ctrl+N" short cut;
- Click the "New" icon in the toolbar;

2. Enter the netlist file and select device type, then click "OK".

Figure 3-6 New FloorPlanner

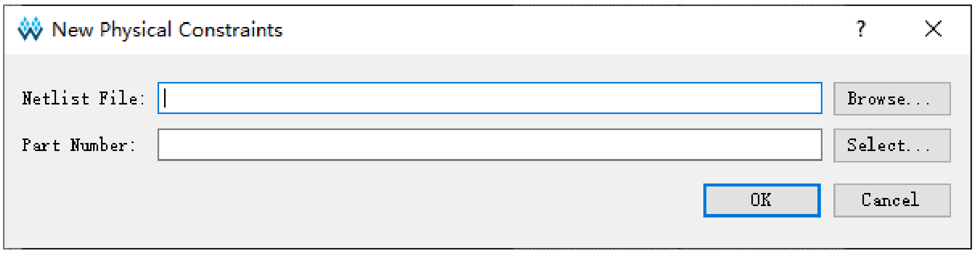
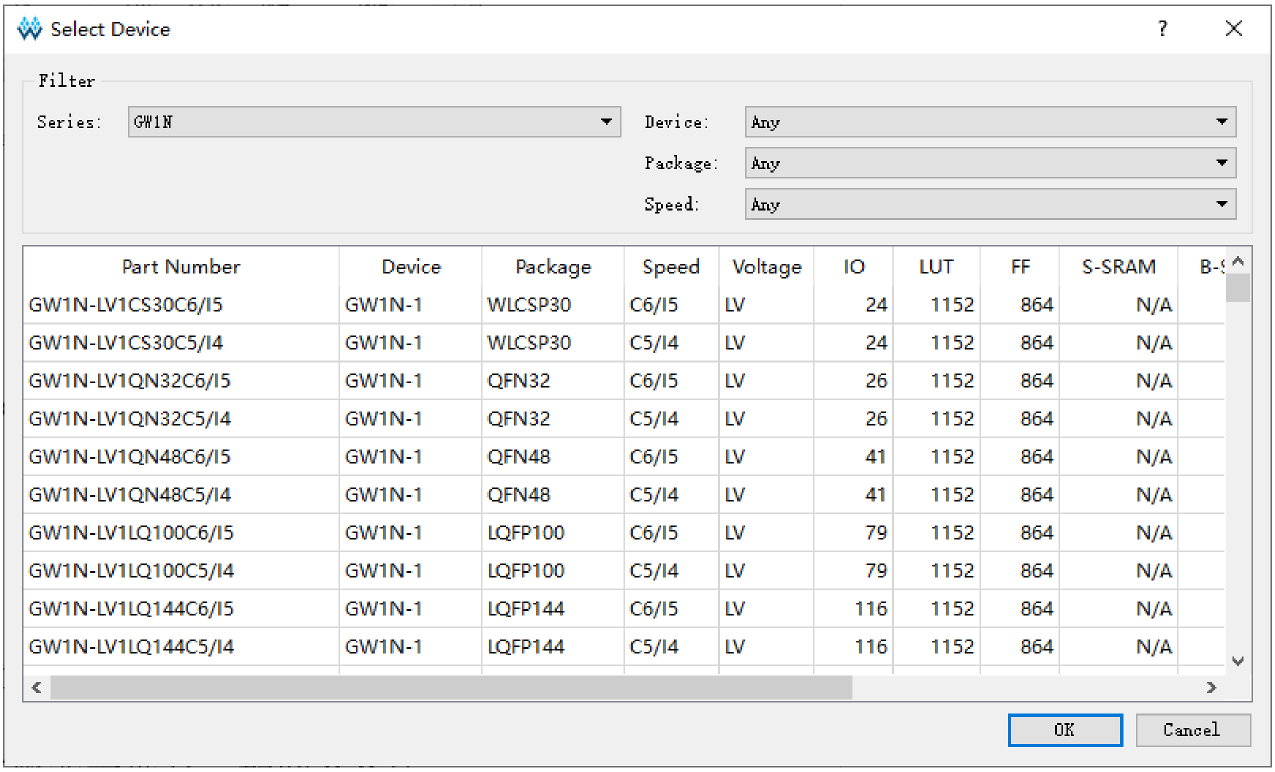


Figure 3-7 Select Device



Series: GW1N

Device: Any

Package: Any

Speed: Any

Part Number	Device	Package	Speed	Voltage	IO	LUT	FF	S-SRAM	B-S
GW1N-LV1CS30C6/I5	GW1N-1	WLCSP30	C6/I5	LV	24	1152	864	N/A	
GW1N-LV1CS30C5/I4	GW1N-1	WLCSP30	C5/I4	LV	24	1152	864	N/A	
GW1N-LV1QN32C6/I5	GW1N-1	QFN32	C6/I5	LV	26	1152	864	N/A	
GW1N-LV1QN32C5/I4	GW1N-1	QFN32	C5/I4	LV	26	1152	864	N/A	
GW1N-LV1QN48C6/I5	GW1N-1	QFN48	C6/I5	LV	41	1152	864	N/A	
GW1N-LV1QN48C5/I4	GW1N-1	QFN48	C5/I4	LV	41	1152	864	N/A	
GW1N-LV1LQ100C6/I5	GW1N-1	LQFP100	C6/I5	LV	79	1152	864	N/A	
GW1N-LV1LQ100C5/I4	GW1N-1	LQFP100	C5/I4	LV	79	1152	864	N/A	
GW1N-LV1LQ144C6/I5	GW1N-1	LQFP144	C6/I5	LV	116	1152	864	N/A	
GW1N-LV1LQ144C5/I4	GW1N-1	LQFP144	C5/I4	LV	116	1152	864	N/A	

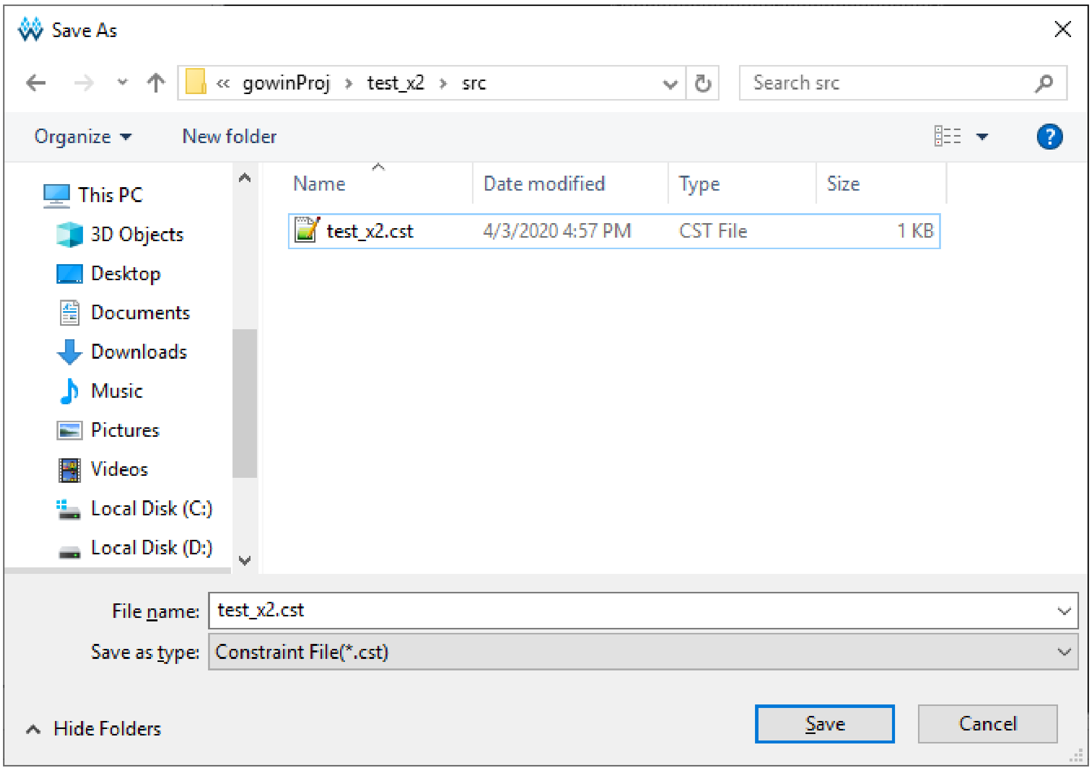
Note!

- You can select the chip, package and all Gowin FPGA devices, as shown in Figure 3-7.
- Start FloorPlanner using the first way in 3.1 Start FloorPlanner.

You can perform following operations in FloorPlanner:

1. Distribute the pins location by dragging;
2. Click "Save" to output constraint files.
3. You can modify the file name in "Save" dialog box, as shown in Figure 3-8.

Figure 3-8 Save Output File



Name	Date modified	Type	Size
test_x2.cst	4/3/2020 4:57 PM	CST File	1 KB

File name: test_x2.cst

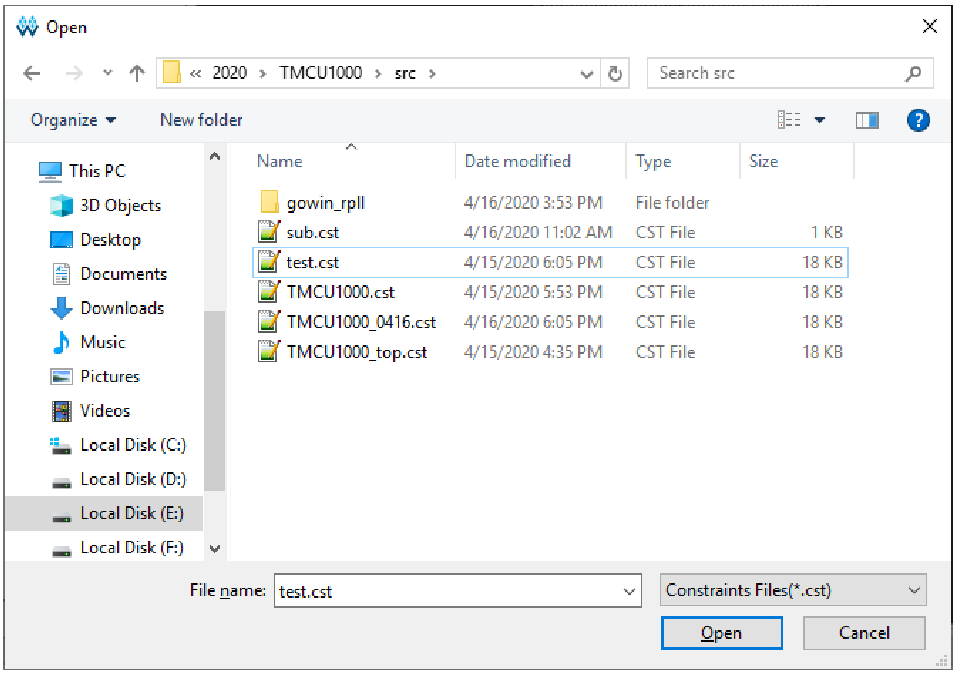
Save as type: Constraint File(*.cst)

3.2.3 Open Constraints File

The steps are as follows:

1. Click "File > Open" in IDE;
2. Open "Open File" dialog box, as shown in Figure 3-9;

Figure 3-9 Open Physical Constraints File



Name	Date modified	Type	Size
gowin_rpll	4/16/2020 3:53 PM	File folder	
sub.cst	4/16/2020 11:02 AM	CST File	1 KB
test.cst	4/15/2020 6:05 PM	CST File	18 KB
TMCU1000.cst	4/15/2020 5:53 PM	CST File	18 KB
TMCU1000_0416.cst	4/16/2020 6:05 PM	CST File	18 KB
TMCU1000_top.cst	4/15/2020 4:35 PM	CST File	18 KB

File name: test.cst

Constraints Files(*.cst)

Note!

You can also open the "Open File" dialog box via the following two ways:

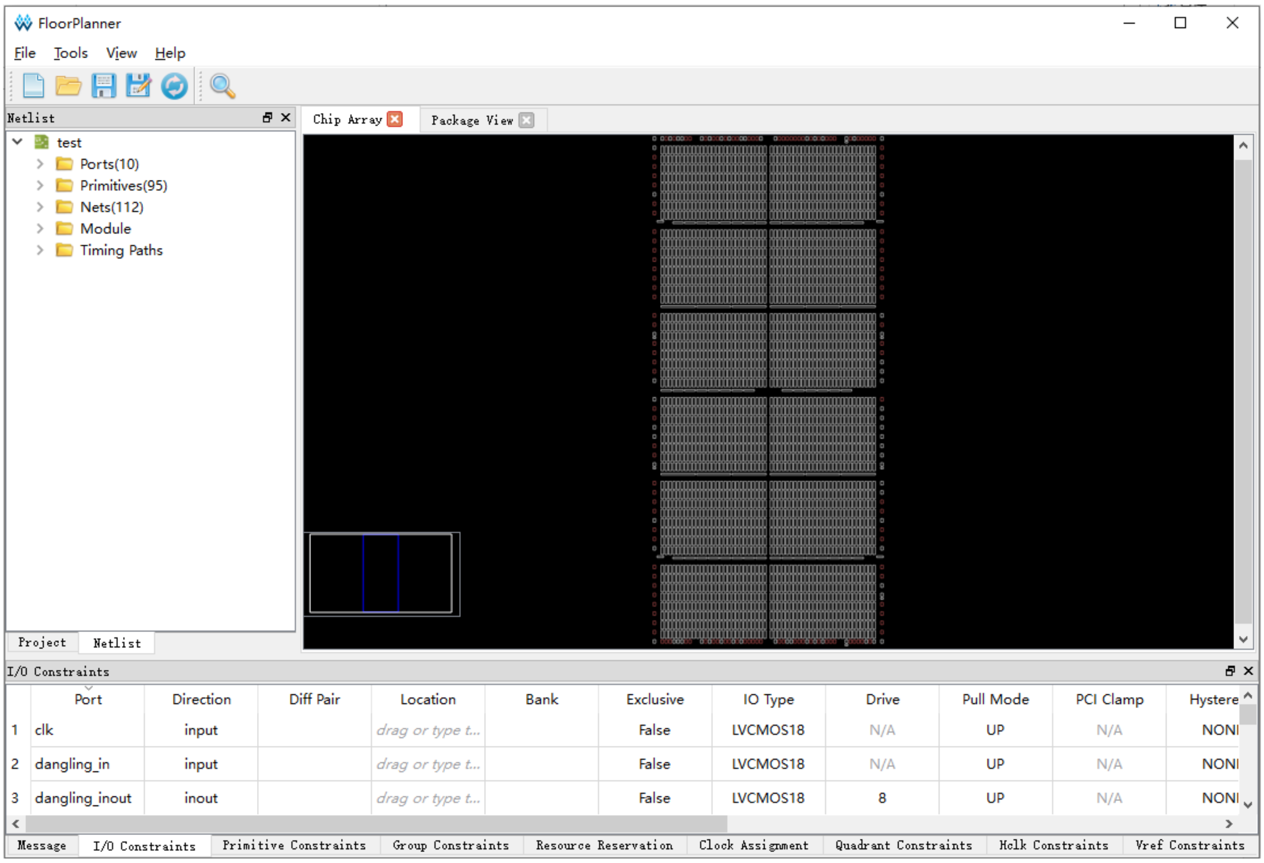
- Use shortcut: Ctrl + O;
- Click "Open" icon in the toolbar; Select the physical constraints file directory and open the selected file.

3.3 FloorPlanner Interface

Create or open FloorPlanner interface (including the netlist file), as shown in Figure 3-10.

The interface displays menu, toolbar, Netlist, Project, Chip Array, Package View and Message, etc.

Figure 3-10 FloorPlanner Interface



	Port	Direction	Diff Pair	Location	Bank	Exclusive	IO Type	Drive	Pull Mode	PCI Clamp	Hystere
1	clk	input		drag or type t...		False	LVCMOS18	N/A	UP	N/A	NONI
2	dangling_in	input		drag or type t...		False	LVCMOS18	N/A	UP	N/A	NONI
3	dangling_inout	inout		drag or type t...		False	LVCMOS18	8	UP	N/A	NONI

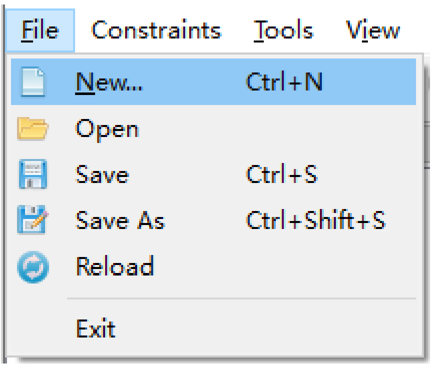
3.3.1 Menu Bar

The menu bar includes "File", "Constraints", "Tools", "View", and "Help".

File Menu Bar

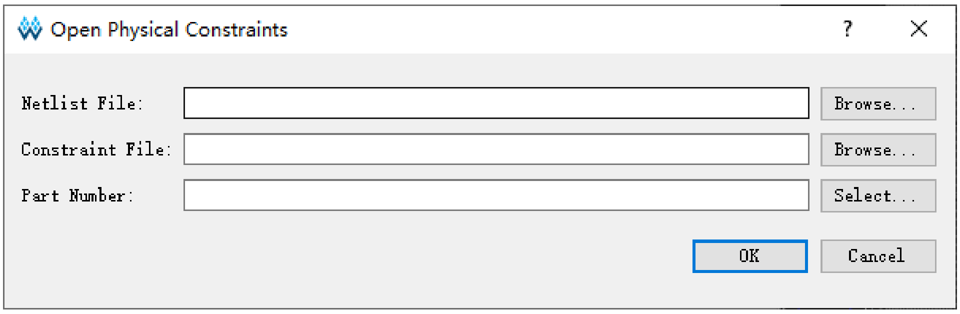
File view is shown in Figure 3-11.

Figure 3-11 File



- New: Create constraints, add user design and select chip, etc., as shown in Figure 3-6;
- Open: Open and add constraints, select part number as shown in Figure 3-12;
- Reload: cst, posp and timing path files can be reloaded after modifying;
- Save: Save the modified files;
- Save As: Save the modified file to a specified file and use the netlist name as the name of constraints file, which can be modified by users;
- Exit: Exit Gowin FloorPlanner.

Figure 3-12 Open Physical Constraints

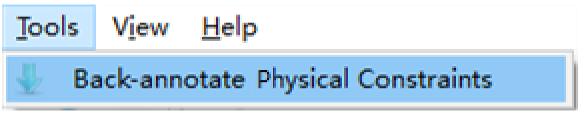


Tools Menu Bar

Tools view is shown in Figure 3-13.

Back-annotate Physical Constraints: Back annotate each primitive and IO place information to physical constraints file.

Figure 3-13 Tools

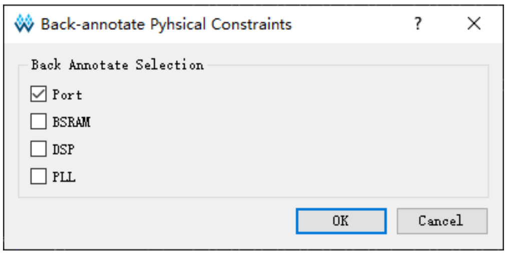


1. Click "Tools > Back-annotate Physical Constraints" to open it as shown

in Figure 3-18.

2. You can select one or more objects in the Back-annotate Physical Constraints window. Click "OK" to open the "Save as" window and print the place information to the physical constraint file.
3. As shown in Figure 3-15, it is the generated physical constraints file when selecting Port in Back-annotate Physical Constraints

Figure 3-14 Back-annotate Physical Constraints



Back-annotate Pyhsical Constraints

Back Annotate Selection

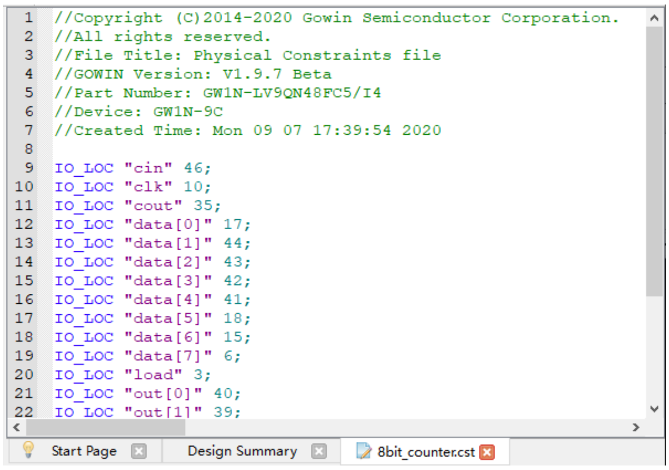
☑ Port

□ BSRAM

□ DSP

□ PLL

Figure 3-15 Back-annotate Port



```
//Copyright (C)2014-2020 Gowin Semiconductor Corporation.
//All rights reserved.
//File Title: Physical Constraints file
//GOWIN Version: V1.9.7 Beta
//Part Number: GW1N-LV9QN48FC5/I4
//Device: GW1N-9C
//Created Time: Mon 09 07 17:39:54 2020

IO_LOC "cin" 46;
IO_LOC "clk" 10;
IO_LOC "cout" 35;
IO_LOC "data[0]" 17;
IO_LOC "data[1]" 44;
IO_LOC "data[2]" 43;
IO_LOC "data[3]" 42;
IO_LOC "data[4]" 41;
IO_LOC "data[5]" 18;
IO_LOC "data[6]" 15;
IO_LOC "data[7]" 6;
IO_LOC "load" 3;
IO_LOC "out[0]" 40;
IO_LOC "out[1]" 39;
```

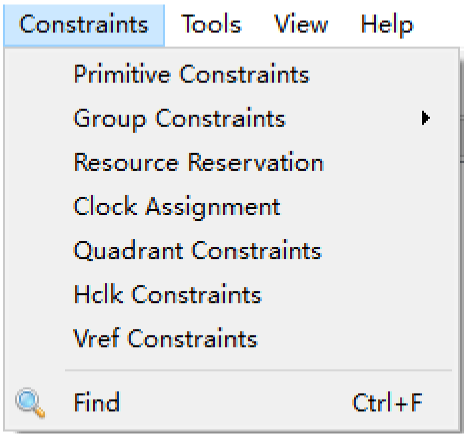
Note!

Back-Annotate Physical Constraints is effective only when FloorPlanner is started in the project after Place & Route runs successfully.

Constraints Menu Bar

Constraints view is as shown in Figure 3-16.

Figure 3-16 Constraints



- Primitive Constraints

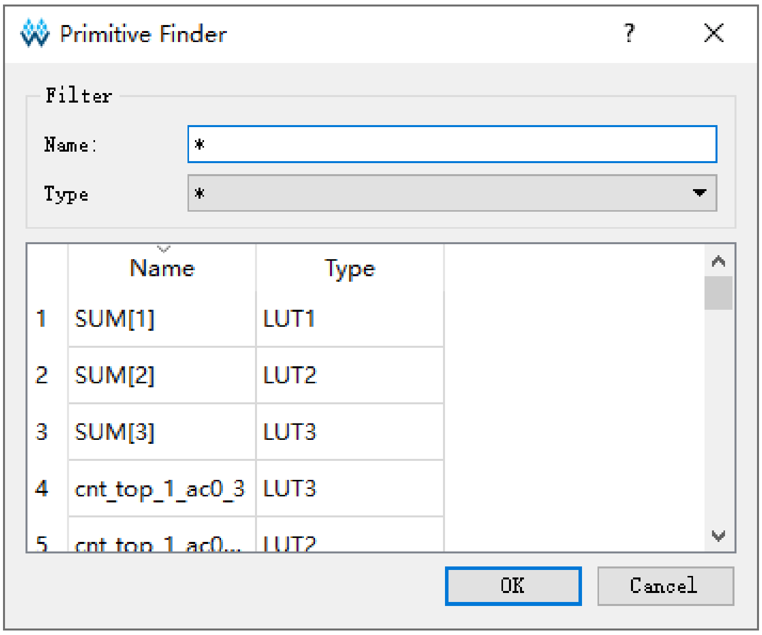
Click "Select Primitives" and Figure 3-17 pops up;

1. Find primitives by "Name" or "type", and select the corresponding primitive;
2. Click "OK" to generate the constraints.

Note!

- The constraints are displayed in "Primitive Constraints" at the bottom of main interface;
- The user can set the location by entering or dragging in editing view;
- The location is highlighted in light blue in Chip Array.

Figure 3-17 Primitive Finder



Name: *

Type *

	Name	Type
1	SUM[1]	LUT1
2	SUM[2]	LUT2
3	SUM[3]	LUT3
4	cnt_top_1_ac0_3	LUT3
5	cnt_top_1_ac0...	LUT2

- Group Constraints

Group constraints include Create Primitive Group and Create Relative Group:

Create Primitive Group

1. Create primitive group. Click "Create Primitive Group" and Figure 3-18 pops up.
2. Users can set group name, primitives, locations and exclusive; Users can add and remove primitives by "✚" and "✖" buttons to create right primitive group as shown in Figure 3-19.

Note!

- Group name, Primitive and Locations are required;
- Locations can be inputed in the following ways:
  - Manually input;
  - Before creating group constraints, copy the location and paste it into "New Primitive Group > Location" in "Chip Array".

3. After finished, click "OK", and the syntax of locations will be checked by tool.
   - If the location is invalid, a prompt dialog box as Figure 3-20 and Figure 3-21 will pop up. User needs to change the location.
   - If there is no error, click "OK", and the available location will be displayed in Chip Array.

Note!

- See the created group constraints in "Group Constraints".
- Double-click the group constraint, and Figure 3-19 will pop up. You can edit the constraints.

Figure 3-18 New Primitive Group

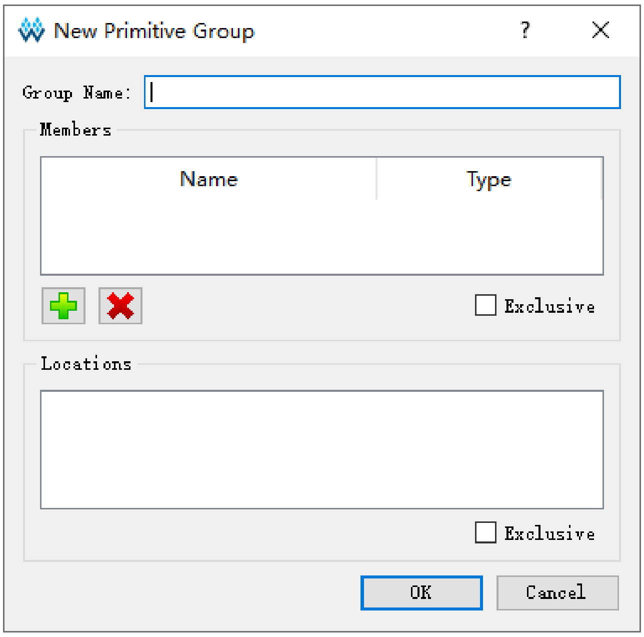
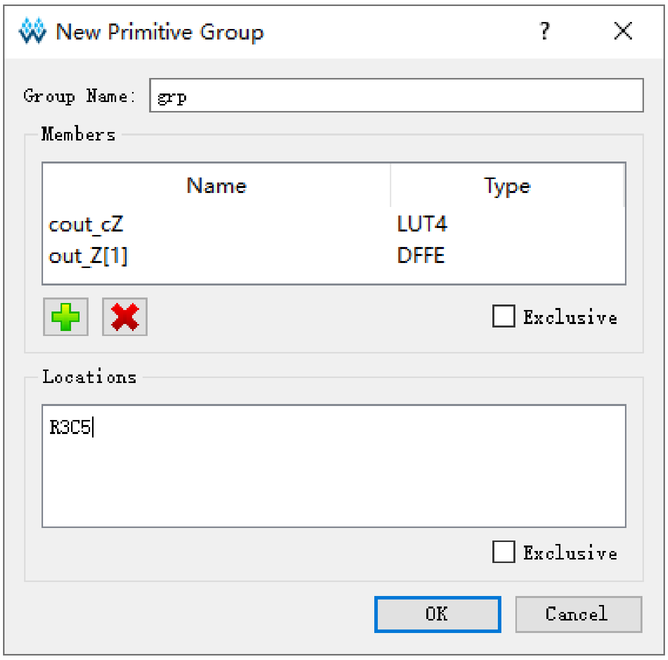


Figure 3-19 Right Primitive Group

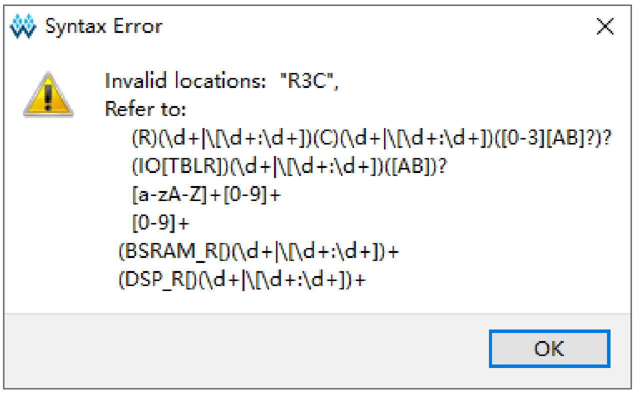


Group Name: grp

Name	Type
cout_cZ	LUT4
out_Z[1]	DFFE

R3C5

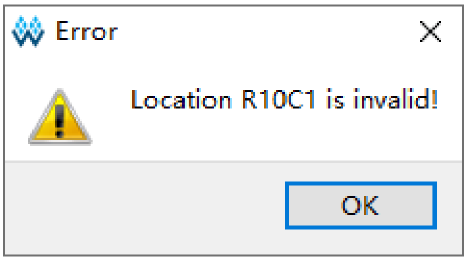
Figure 3-20 Invalid Locations



Syntax Error

```
Invalid locations:  "R3C",
Refer to:
    (R)(\d+|\[\d+:\d+])(C)(\d+|\[\d+:\d+])([0-3][AB]?)?
    (IO[TBLR])(\d+|\[\d+:\d+])([AB])?
    [a-zA-Z]+[0-9]+
    [0-9]+
  (BSRAM_R[)(\d+|\[\d+:\d+])+
  (DSP_R[)(\d+|\[\d+:\d+])+
```

Figure 3-21 Invalid Locations



Error

Location R10C1 is invalid!

Create Relative Group

1. Create relative group constraints. Click Create Relative Group and pops up.
2. Users can set group name, primitive and locations. Users can add and remove primitive by "➕" and "✖" buttons. The created relative group constraints are shown in Figure 3-22.

Note!

- Group name, Primitive and Relative Location are required;
- The locations can be inputed in the following ways:
  - Manually input;
  - Before creating group constraints, copy the location and paste it to "New Relative Group > Relative Location" in "Chip Array".

3. Click "OK" to generate the constraints.

Note!

- See the created constraints in "Group Constraints".
- Double-click the constraints, and Figure 3-23 will pop up. You can edit the constraints.

Figure 3-22 New Relative Group

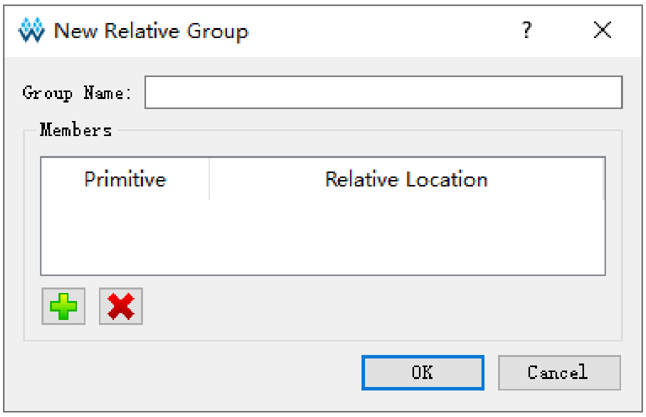
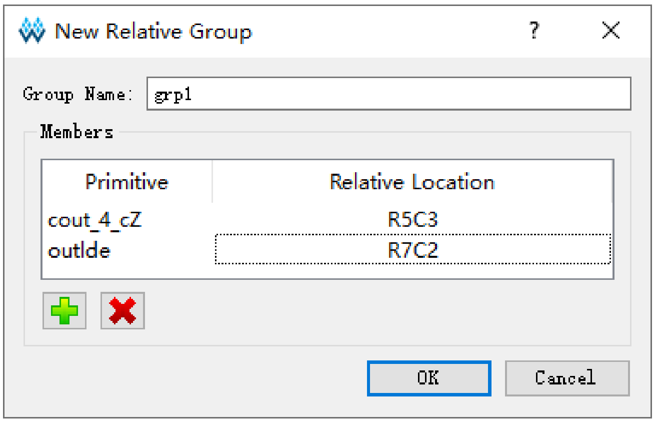


Figure 3-23 Right New Relative Group



Group Name: grp1

Primitive	Relative Location
cout_4_cZ	R5C3
outlde	R7C2

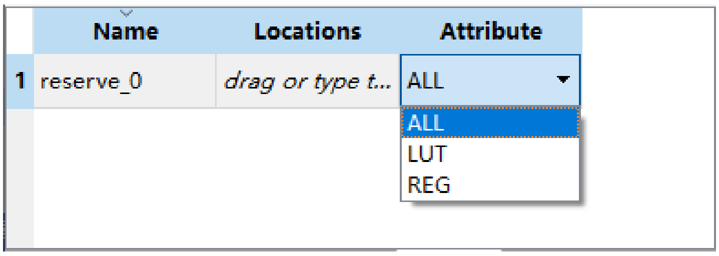
- Resource Reservation

1. Create reserve resources constraints. Click Reserve Resources to create a new constraint in "Resource Reservation" at the bottom of the interface;
2. You can enter the locations or by dragging;
3. Double-click "Attribute" to set the attribute for the reserved locations, as shown in Figure 3-24.

Note!

Name is used to distinguish reserved constraints. The name cannot be modified.

Figure 3-24 Resource Reservation



	Name	Locations	Attribute
1	reserve_0	*drag or type t...*	ALL

- Clock Assignment

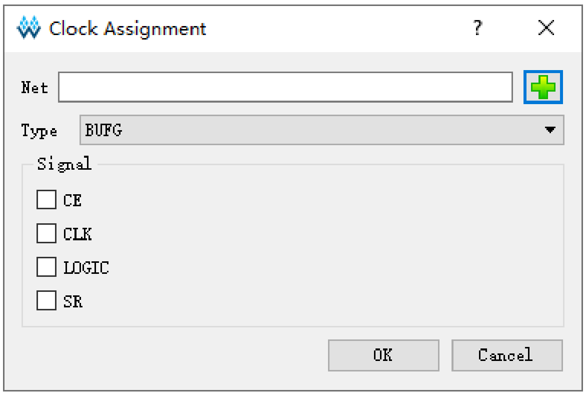
Create clock constraints and the number of constraints is limited. It will be checked when checking constraints validity. Click Clock/Control Assignment and Figure 3-25 pops up. You can perform the following operations:

1. Click "+" to select the corresponding Net;
2. Select "BUFG", "BUFG[0]~[7]", "BUFS" and "LOCAL_CLOCK" via "Type" drop-down list;
3. Configure Signal via "CE" and "CLK". After finished, click "OK" to generate constraints in "Clock the Assignment". Double click to open the dialog box for editing.

Note!

When selecting LOCAL_CLOCK, the signal check box is grayed.

Figure 3-25 Clock Assignment



Clock Assignment

Net

Type BUFG

Signal

□ CE

□ CLK

□ LOGIC

□ SR

- Quadrant Constraints

Create DCS and DQCE quadrant constraints. Constrain the specified instance to the specific quadrant according to the chip quadrants distribution, as shown in Figure 3-26 and Figure 3-27. The related operations are as follows:

1. Select the corresponding DCS/DQCE primitive by clicking "+";
2. Configure quadrant positions via the check boxes under "Position".
3. Click "OK" to generate constraints, which will display in "Quadrant Constraints". You can double click to open the dialog box for editing.

Note!

It cannot add constraints if there is no DCS/DQCE primitive.

Figure 3-26 Quadrant Constraints (GW1N)

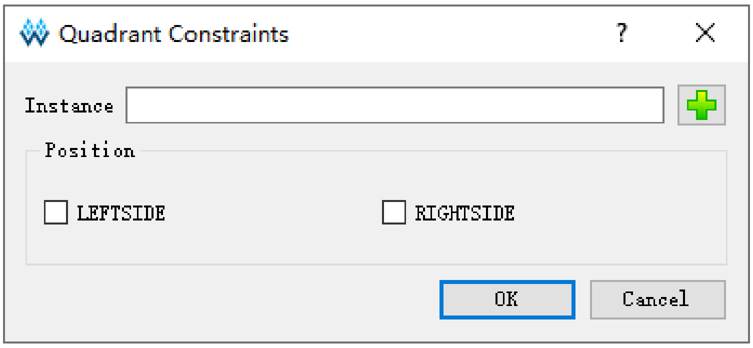
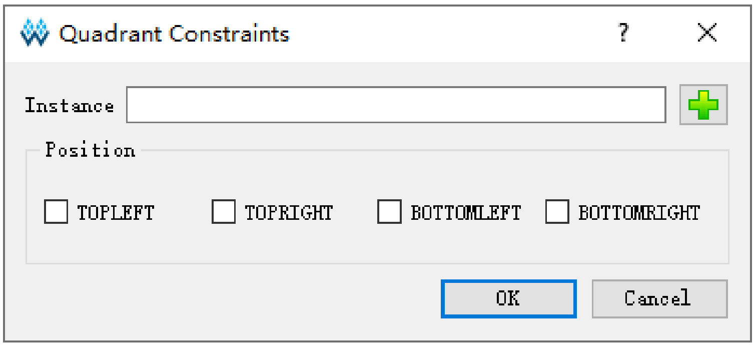


Figure 3-27 Quadrant Constraints (GW2A)



- Hclk Constraints

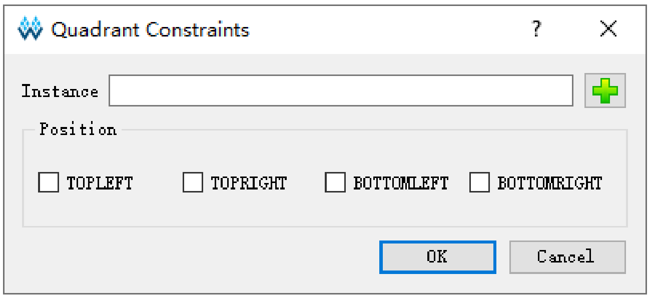
Create HCLK primitives constraints and specify the constraint locations on chip, as shown in Figure 3-28. The related operations are as follows:

1. You can select primitive by clicking "✚";
2. Configure quadrant positions via the check boxes under "Position；
3. Click "OK" to generate constraints, which will display in "Hclk Constraints". You can double click to open the dialog box for editing.

Note!

- It cannot add constraints if there is no primitive;
- The positions are different due to different devices in project.

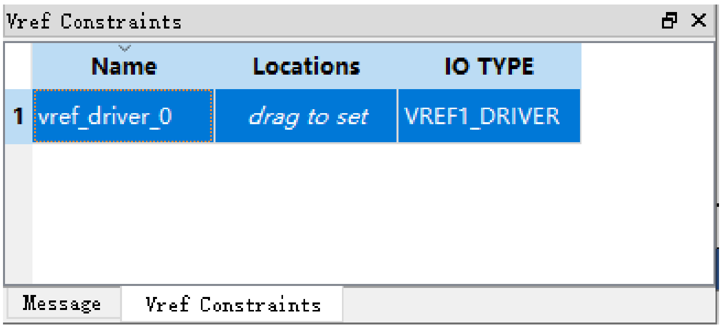
Figure 3-28 Hclk Constraint



- Vref Constraints

Create Vref Driver to configure IO Port Vref, and click the menu to create a new constraint in "Vref Constraints", as shown in Figure 3-29 .

Figure 3-29 Vref Constraints



	Name	Locations	IO TYPE
1	vref_driver_0	*drag to set*	VREF1_DRIVER

Note!

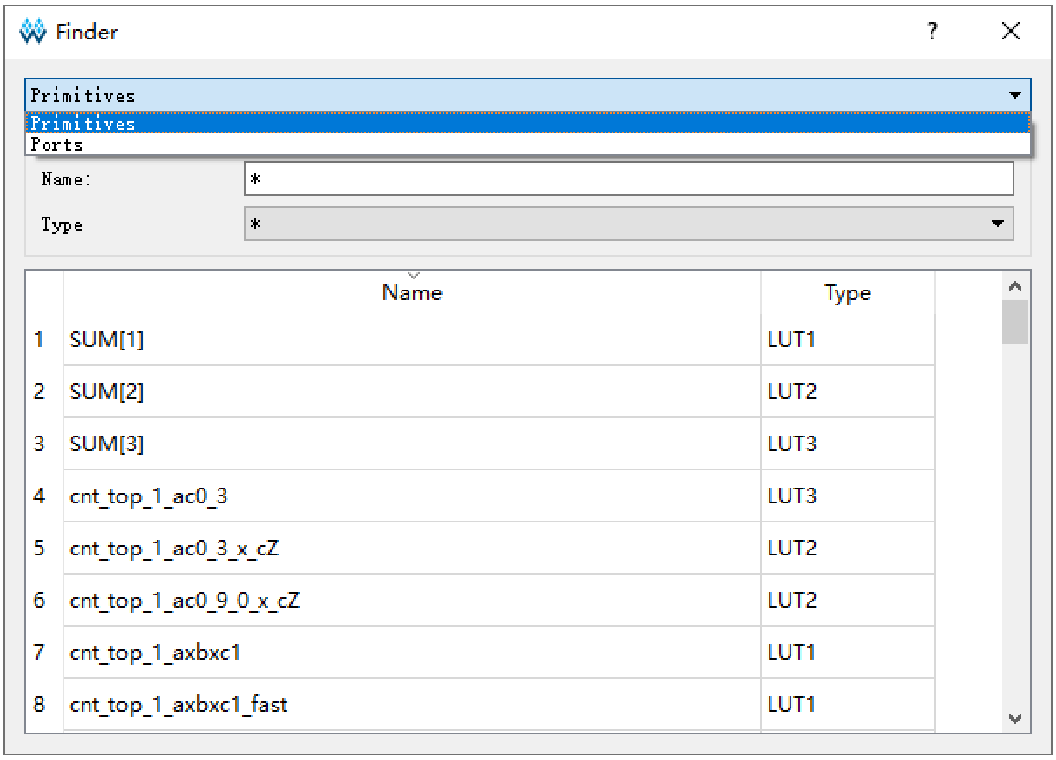
- Specify Vref constraints position by dragging.
- Modify Vref name by double-clicking.

Find

You can find quickly find Primitive, IO Port, and edit the corresponding constraints by right-clicking. Figure 3-30 will pop up by clicking. The related operations are as follows:

- You can find primitives or ports by selecting "All Primitive" or "Ports";
- You can select the corresponding item and right-click "Edit Primitive Constraint" to create constraints.

Figure 3-30 Find



Name: *

Type: *

	Name	Type
1	SUM[1]	LUT1
2	SUM[2]	LUT2
3	SUM[3]	LUT3
4	cnt_top_1_ac0_3	LUT3
5	cnt_top_1_ac0_3_x_cZ	LUT2
6	cnt_top_1_ac0_9_0_x_cZ	LUT2
7	cnt_top_1_axbxc1	LUT1
8	cnt_top_1_axbxc1_fast	LUT1

View Menu Bar

As shown in Figure 3-31, View includes Toolbars, Windows, Zoom In, Zoom Out and Zoom Fit. The description of these sub-menus is as follows:

- Toolbars: Display shortcuts;
- Windows: Display different windows, as shown in Figure 3-32;
- Zoom In: Zoom in Chip Array or Package View;
- Zoom Out: Aoom out Chip Array or Package View;
- Zoom Fit: Zoom in/out Chip Array or Package View according to the window size.

Figure 3-31 View

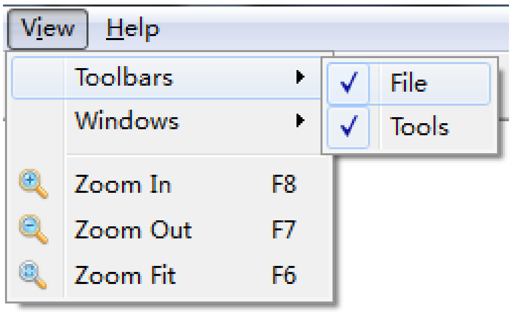
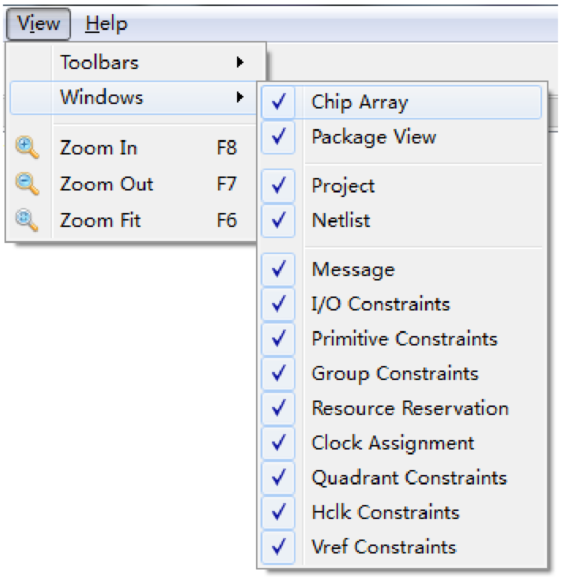


Figure 3-32 Windows



Help Menu Bar

Help is used to provide software version and copyright information.

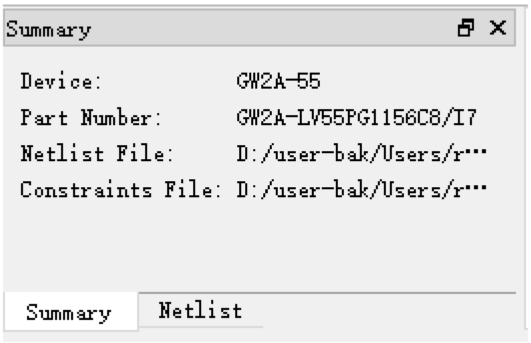
3.3.2 Summary and Netlist Window

Summary and Netlist windows display device, part number, user design, constraints path and netlist, etc.

Summary Window

Project interface is shown in Figure 3-33. It can display the chip information in current project, such as part number, design files, constraint fille, posp file and timing paths file.

Figure 3-33 Summary Window



Device: GW2A-55
Part Number: GW2A-LV55PG1156C8/I7
Netlist File: D:/user-bak/Users/r…
Constraints File: D:/user-bak/Users/r…

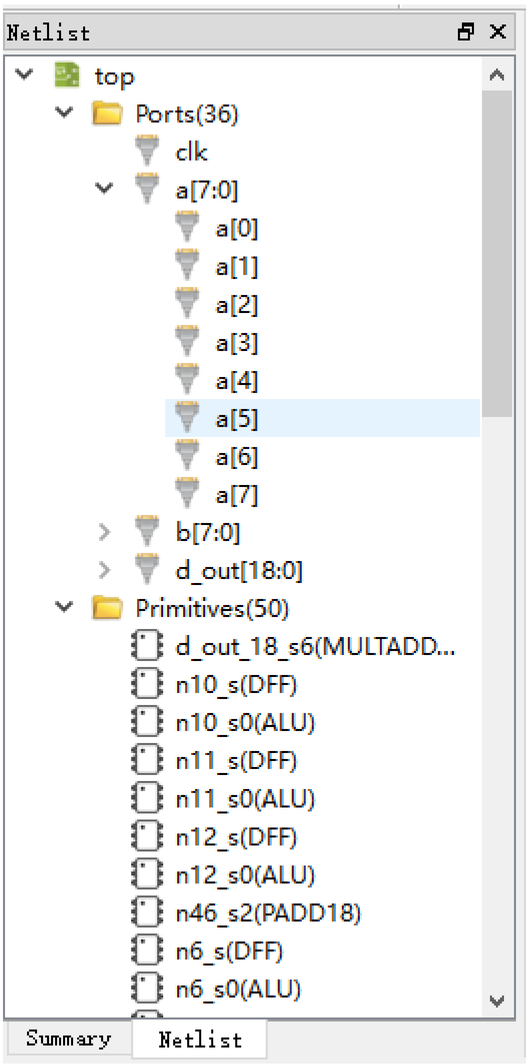
Netlist Window

As shown in Figure 3-34, Netlist window displays Port, Primitives, Nets, Module and timing paths.

Note!

- The Ports and Primitives names are displayed in full path and sorted in alphabetical ascending order by default.
- Port and Net display via Bus and non-Bus, as shown in Figure 3-35;
- Module displays in hierarchy and the number of instances in each module is also displayed, as shown in Figure 3-36;
- The timing path is listed in slack order from small to large, as shown in Figure 3-37.

Figure 3-34 Netlist Window



```
Netlist
top
  Ports(36)
    clk
    a[7:0]
      a[0]
      a[1]
      a[2]
      a[3]
      a[4]
      a[5]
      a[6]
      a[7]
    b[7:0]
    d_out[18:0]
  Primitives(50)
    d_out_18_s6(MULTADD...
    n10_s(DFF)
    n10_s0(ALU)
    n11_s(DFF)
    n11_s0(ALU)
    n12_s(DFF)
    n12_s0(ALU)
    n46_s2(PADD18)
    n6_s(DFF)
    n6_s0(ALU)
```

Figure 3-35 BUS and Non-Bus Display

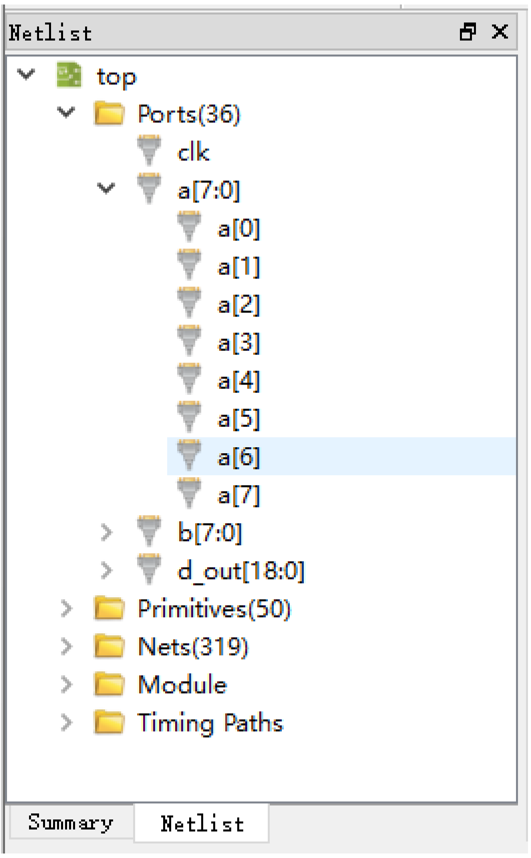
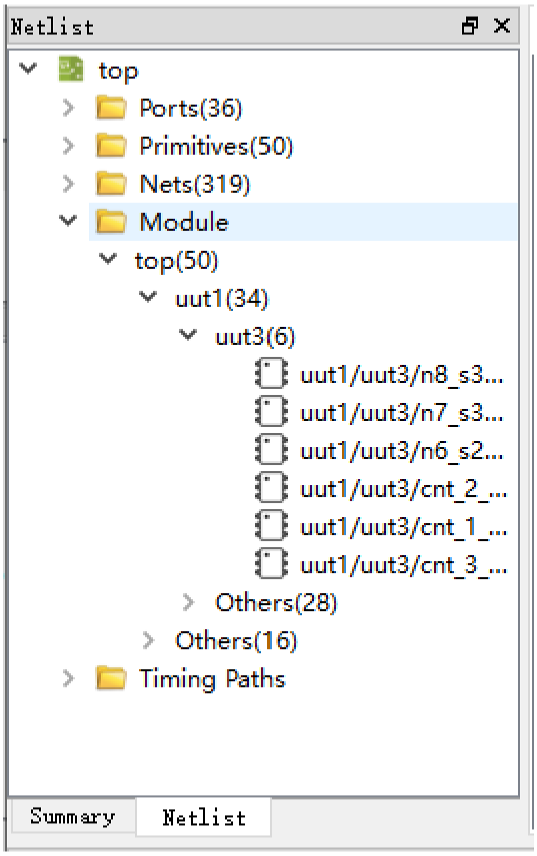
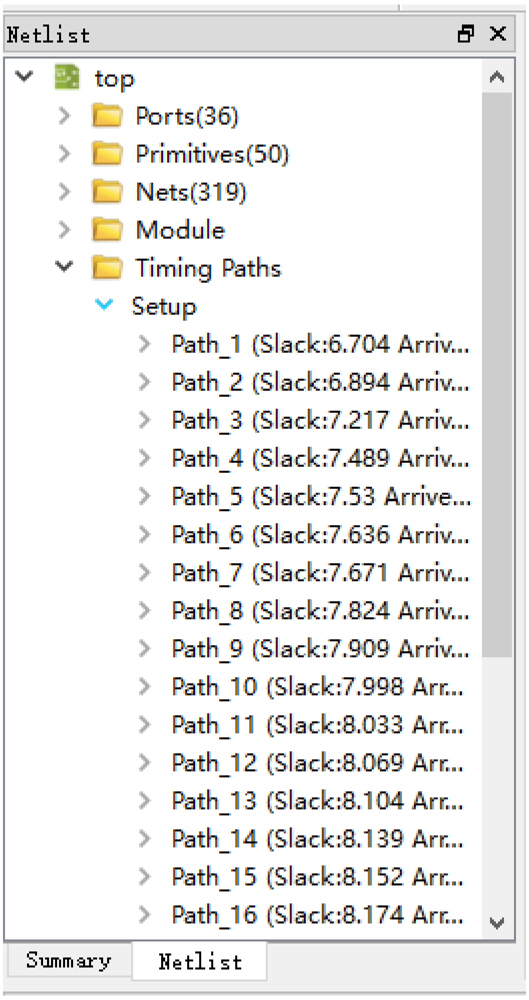


Figure 3-36 Hierarchy Display



```
top
    Ports(36)
    Primitives(50)
    Nets(319)
    Module
        top(50)
            uut1(34)
                uut3(6)
                    uut1/uut3/n8_s3...
                    uut1/uut3/n7_s3...
                    uut1/uut3/n6_s2...
                    uut1/uut3/cnt_2_...
                    uut1/uut3/cnt_1_...
                    uut1/uut3/cnt_3_...
                Others(28)
            Others(16)
    Timing Paths
```

Figure 3-37 Timing Paths



Netlist

- top
  - Ports(36)
  - Primitives(50)
  - Nets(319)
  - Module
  - Timing Paths
    - Setup
      - Path_1 (Slack:6.704 Arriv...
      - Path_2 (Slack:6.894 Arriv...
      - Path_3 (Slack:7.217 Arriv...
      - Path_4 (Slack:7.489 Arriv...
      - Path_5 (Slack:7.53 Arrive...
      - Path_6 (Slack:7.636 Arriv...
      - Path_7 (Slack:7.671 Arriv...
      - Path_8 (Slack:7.824 Arriv...
      - Path_9 (Slack:7.909 Arriv...
      - Path_10 (Slack:7.998 Arr...
      - Path_11 (Slack:8.033 Arr...
      - Path_12 (Slack:8.069 Arr...
      - Path_13 (Slack:8.104 Arr...
      - Path_14 (Slack:8.139 Arr...
      - Path_15 (Slack:8.152 Arr...
      - Path_16 (Slack:8.174 Arr...

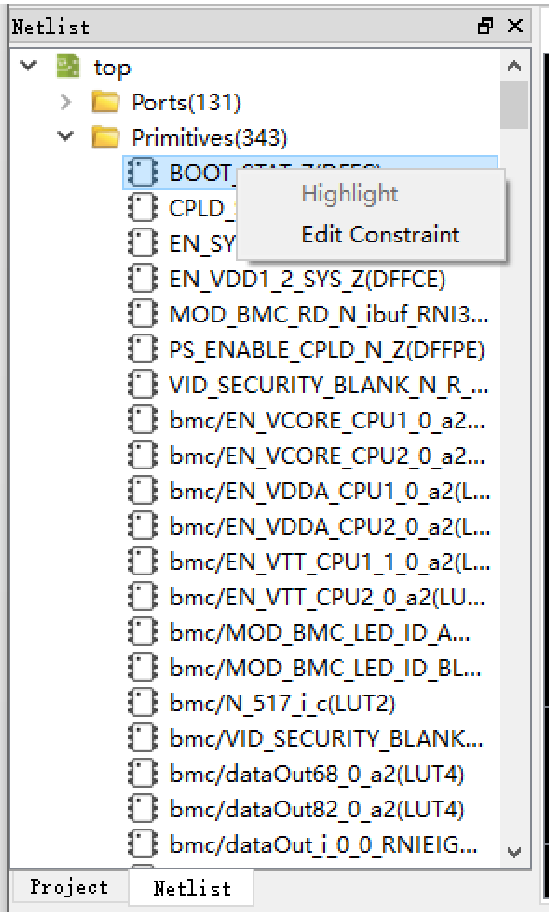
Netlist provides right-click menu as shown below:

- Highlight: Highlight the corresponding constraints location in Chip Array.
- Edit Constraintt: Edit the corresponding constraints.

Note!

If the current Primitive or Port has no constraints locations, the highlight is not available, as shown in Figure 3-38.

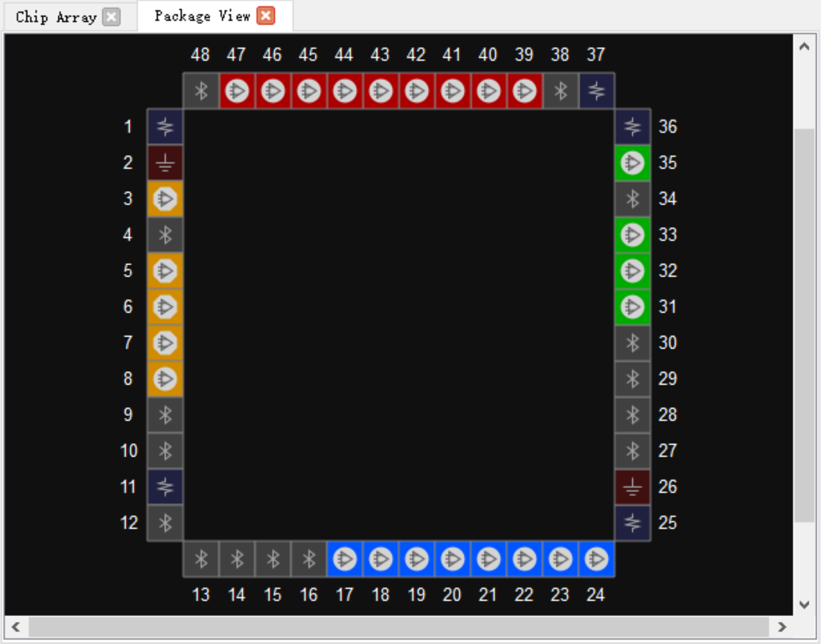
Figure 3-38 Netlist Right-clicking


Netlist
top
Ports(131)
Primitives(343)
BOOT_
CPLD_
EN_SY
Highlight
Edit Constraint
EN_VDD1_2_SYS_Z(DFFCE)
MOD_BMC_RD_N_ibuf_RNI3...
PS_ENABLE_CPLD_N_Z(DFFPE)
VID_SECURITY_BLANK_N_R_...
bmc/EN_VCORE_CPU1_0_a2...
bmc/EN_VCORE_CPU2_0_a2...
bmc/EN_VDDA_CPU1_0_a2(L...
bmc/EN_VDDA_CPU2_0_a2(L...
bmc/EN_VTT_CPU1_1_0_a2(L...
bmc/EN_VTT_CPU2_0_a2(LU...
bmc/MOD_BMC_LED_ID_A...
bmc/MOD_BMC_LED_ID_BL...
bmc/N_517_i_c(LUT2)
bmc/VID_SECURITY_BLANK...
bmc/dataOut68_0_a2(LUT4)
bmc/dataOut82_0_a2(LUT4)
bmc/dataOut_i_0_0_RNIEIG...
Project
Netlist

3.3.3 Package View

As shown in Figure 3-39 , taking GW1NRF-4B-QFN48 as an example, Package View displays I/O, supply pin, and ground pin based on chip package. When the mouse is placed on a location, the I/O type, bank, and LVDS will display.

Figure 3-39 Package View (GW1NRF-4B-QFN48)



Various symbols and colors are used to distinguish I/O, supply pin and ground pin. The colors of IO pins of different BNAKS are different, as shown below:

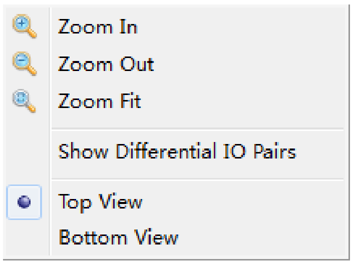
- "▷" denotes I/O. The color changes with the BANK;
- "≑" denotes VCC. The color does not change;
- ⏚ denotes VSS. The color does not change;
- ᛒ denotes bluetooth interface. The color does not change;

As shown in Figure 3-40 , Package View supports right-clicking as described below:

- Zoom In: Zoom in Package View;
- Zoom Out: Zoom out Package View;
- Zoom Fit: Zoom in/out Package View according to window size.
- Show Differential IO Pairs: Display differential pair. As shown in Figure 3-41, a differential pair is connected by a red line.
- Top View: Package View is displayed in the top view by default. The top view of GW1N-9-WLCSP64 is as shown in Figure 3-42;
- Bottom View: The bottom view of GW1N-9-WLCSP64 is as shown in

Figure 3-43;

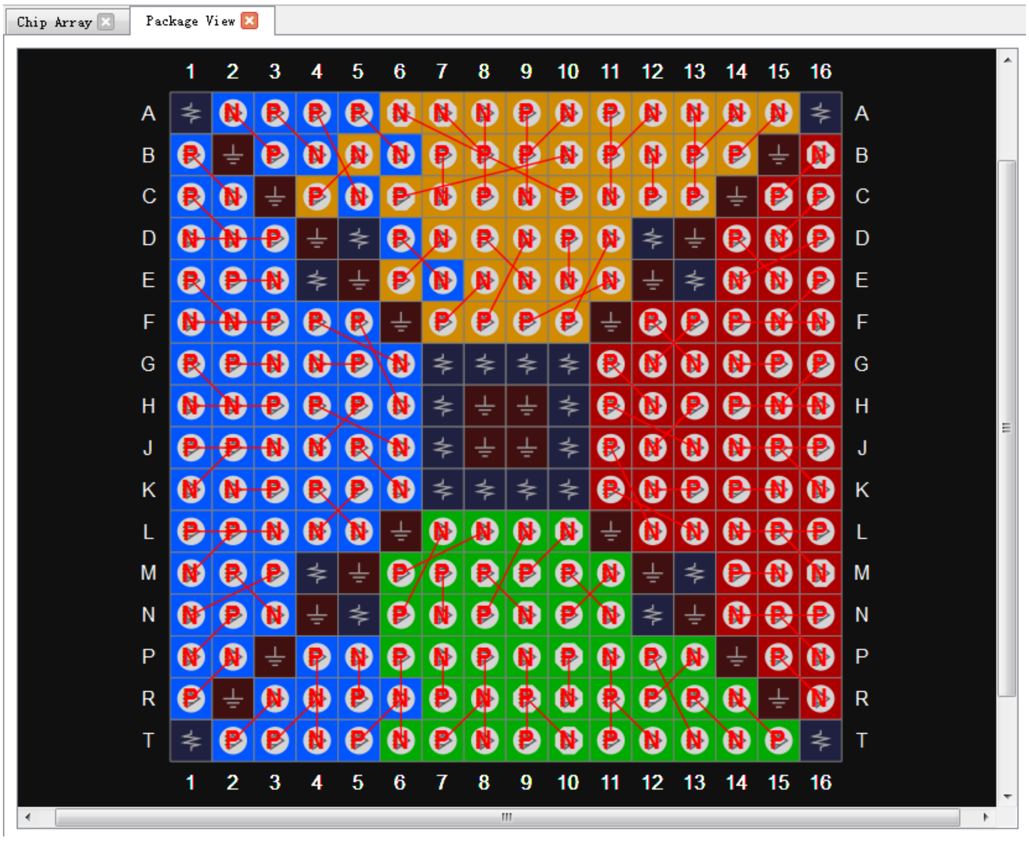
Figure 3-40 Package View Right-clicking



Note!

The top view and bottom view of GW1N-9-WLCSP81M are opposite compared with the general ones, as shown in Figure 3-44 and Figure 3-45.

Figure 3-41 Differential Pair Display



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
A	⚡	N	P	P	P	N	N	N	P	N	P	N	N	N	N	⚡	A
B	P	⏚	P	N	N	N	P	P	P	N	P	N	P	P	⏚	N	B
C	P	N	⏚	P	N	P	N	P	N	P	N	P	P	⏚	P	P	C
D	N	N	P	⏚	⚡	P	N	P	N	P	N	⚡	⏚	P	N	P	D
E	P	P	N	⚡	⏚	P	N	N	N	N	N	⏚	⚡	N	N	P	E
F	N	N	P	P	P	⏚	P	P	P	P	⏚	P	P	P	N	N	F
G	P	P	N	N	P	N	⚡	⚡	⚡	⚡	P	N	N	N	P	P	G
H	N	N	P	P	P	N	⚡	⏚	⏚	⚡	P	N	P	P	N	N	H
J	P	P	N	N	P	N	⚡	⏚	⏚	⚡	P	N	N	N	P	P	J
K	N	N	P	P	P	N	⚡	⚡	⚡	⚡	P	N	P	P	N	N	K
L	P	P	N	N	N	⏚	N	N	N	N	⏚	N	N	N	P	P	L
M	N	P	P	⚡	⏚	P	P	P	P	P	N	⏚	⚡	P	N	N	M
N	N	P	N	⏚	⚡	P	N	P	N	P	N	⚡	⏚	N	P	P	N
P	N	N	⏚	P	N	P	N	P	N	P	N	P	N	⏚	P	N	P
R	P	⏚	N	N	P	N	P	N	P	N	P	P	P	N	⏚	N	R
T	⚡	P	P	N	P	N	P	N	P	N	P	N	N	N	P	⚡	T
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	

Figure 3-42 Top View

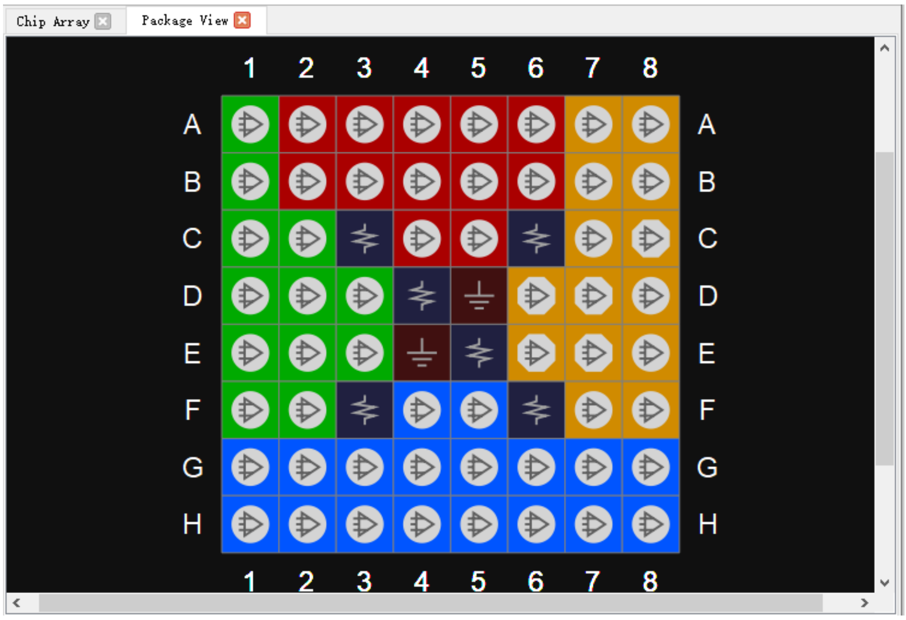


Figure 3-43 Bottom View

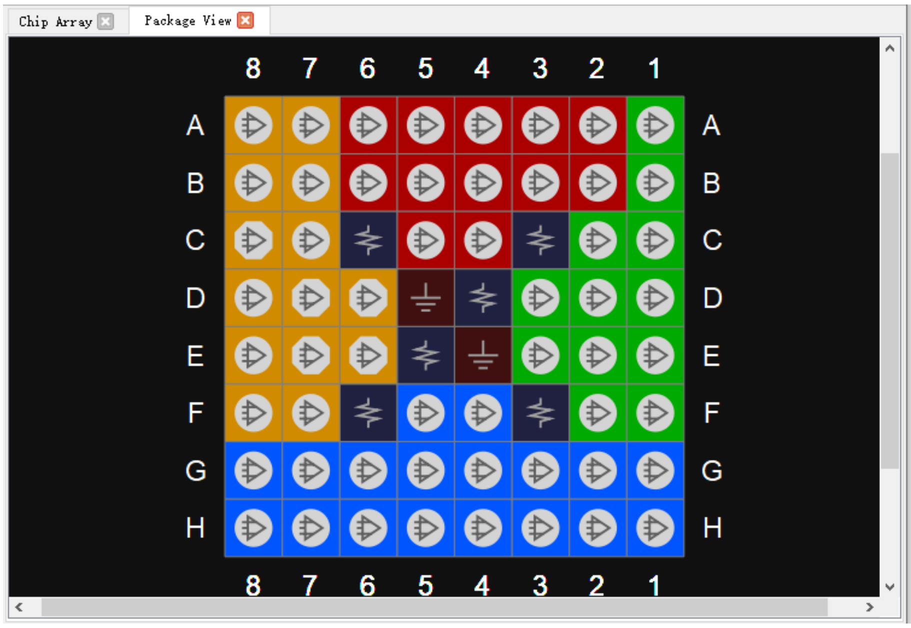


Figure 3-44 GW1N-9-WLCSP81M Top View

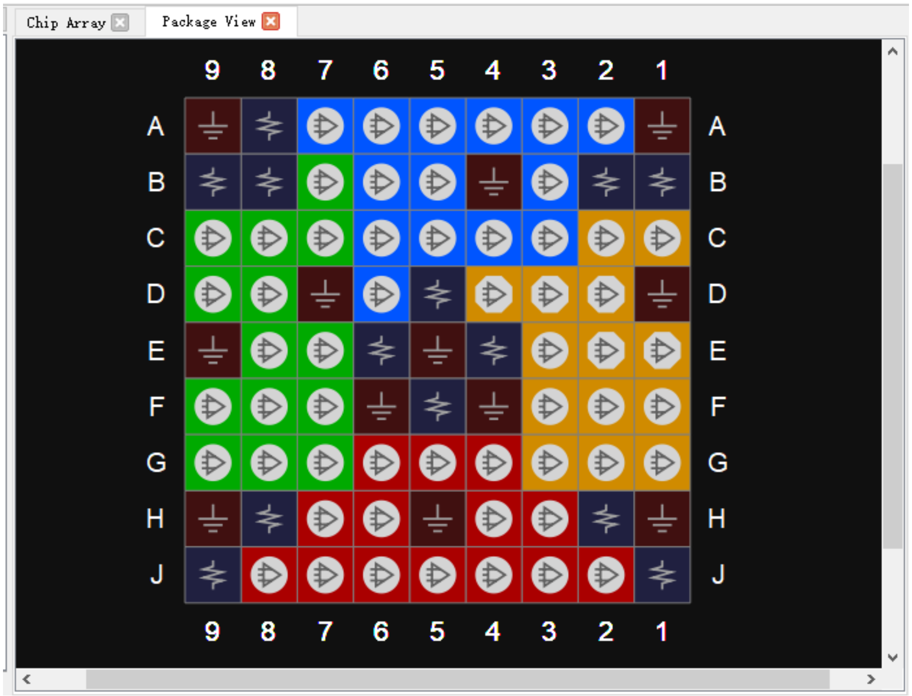
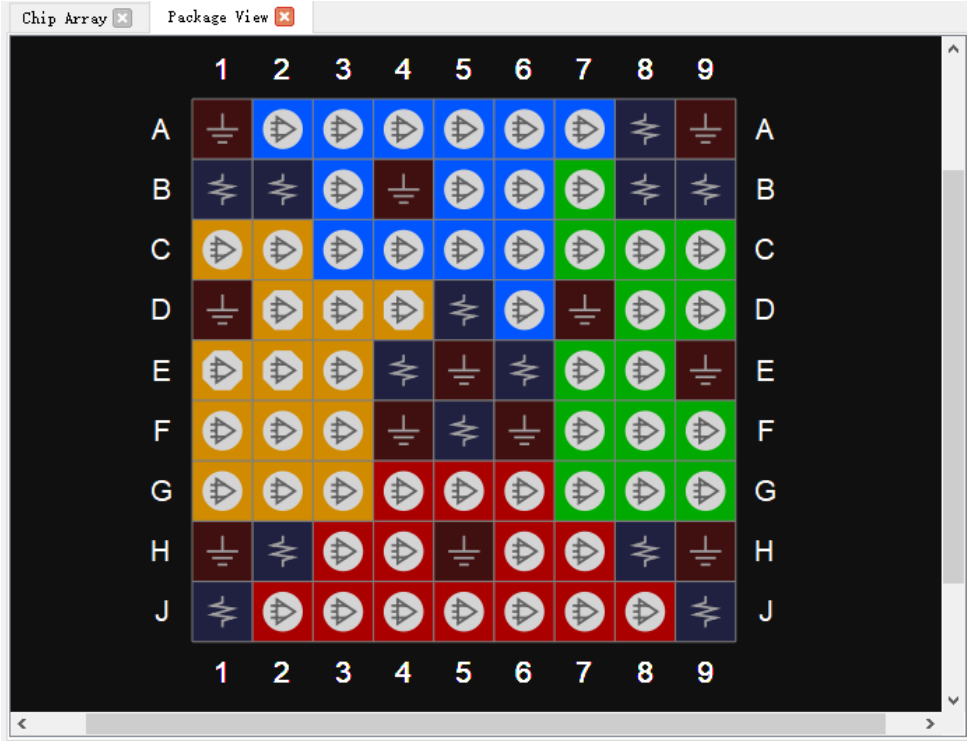


Figure 3-45 GW1N-9-WLCSP81M Bottom View



Package View supports the display of IO Port constraints locations, and the IO port can be constrained by dragging from the Netlist or I/O Constraints to the Package View window.

Note!

- When dragging, the port name will display;
- "⊘" indicates that the location cannot place the dragged port.

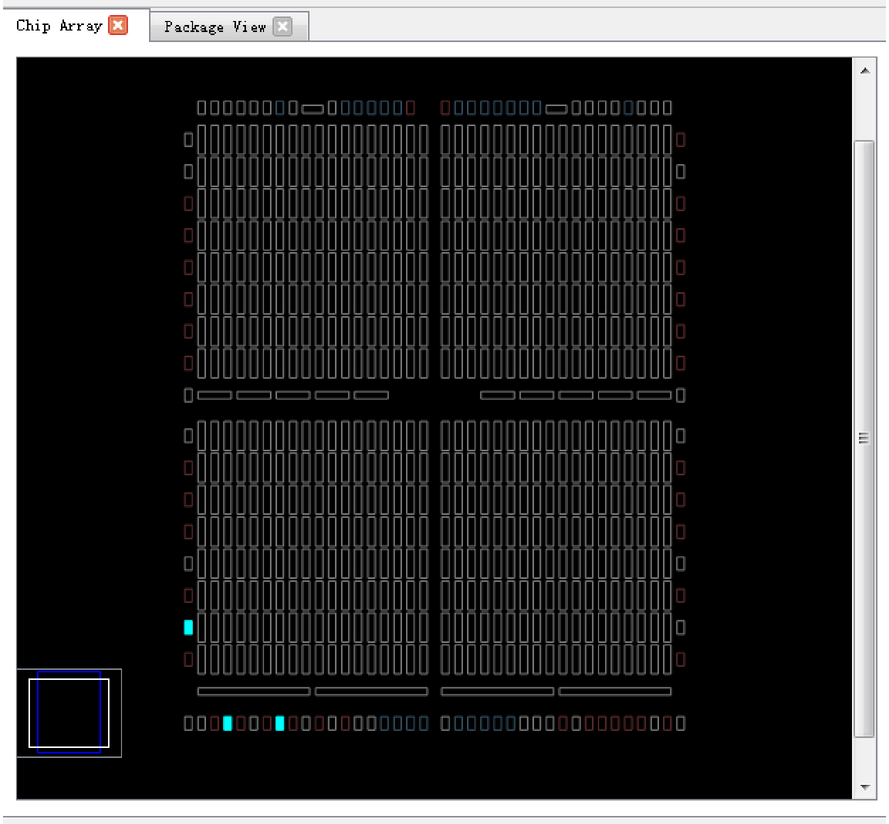
3.3.4 Chip Array Window

Chip Array interface of FloorPlanner is as shown in Figure 3-46. Chip Array displays IOB, CFU, DSP, PLL and BSRAM according to chip row and column, supports the real-time display of all constraints locations, and also supports the functions of zoom in/out and dragging, etc.

IOB denotes all IOB locations of die and is distinguished with different colors:

- White: Packaged IO location;
- Red: Unpackaged IO location;
- Blue: If the device is with SIP, such as GW2AR-18, GW1NR-4, GW1NR-9, GW1NSR-2C, the device will have blue marked IOB, which means the embedded IO locations.

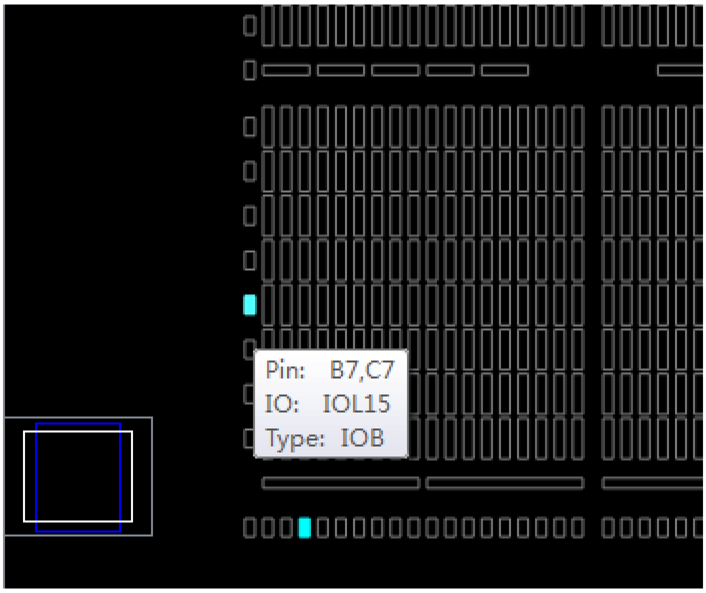
Figure 3-46 Chip Array Interface



Chip Array provides grid mode, macrocell mode and primitive mode.

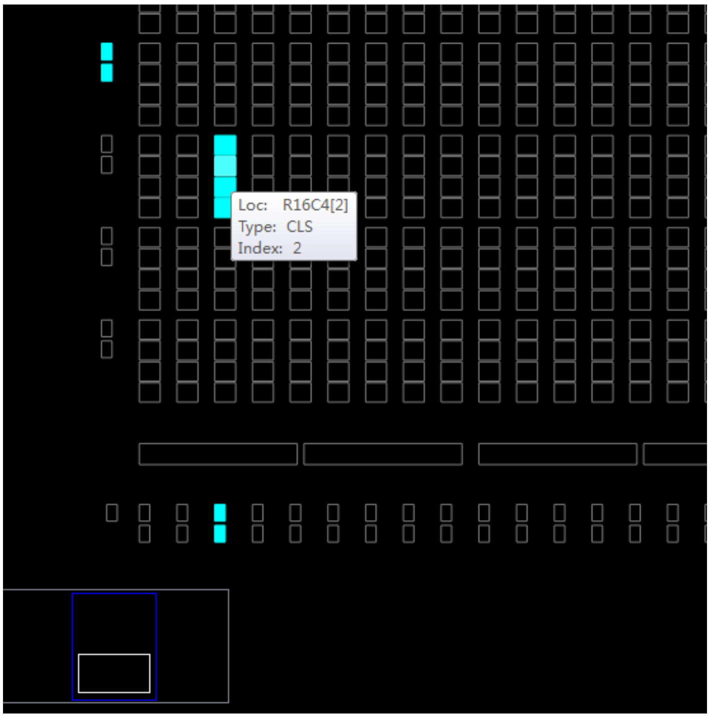
- Grid mode: Display constraints locations, as shown in Figure 3-47;
- Macrocell mode: Display constraints locations in CLS, block etc., as shown in ;
- Primitive mode: Display constraints locations in REG, LUT etc., as shown in Figure 3-49.

Figure 3-47 Constraints in Grid



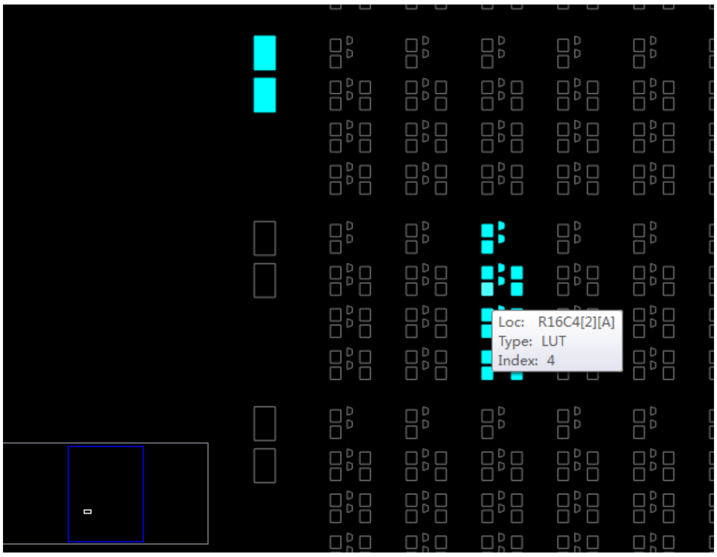
Pin: B7,C7
IO: IOL15
Type: IOB

Figure 3-48 Constraints in Macrocell



Loc: R16C4[2]
Type: CLS
Index: 2

Figure 3-49 Constraints in Primitive



Loc: R16C4[2][A]
Type: LUT
Index: 4

Chip Array supports dragging functions:

- Drag from Netlist to Array to generate constraints and specify constraints location;
- Drag from Editing area to Array to specify constraints location;

There is chip sub-window in Chip Array for real-time display of the current window in the chip. Drag the white frame in the chip sub-window to move Chip Array. Chip Array distinguishes constraints type and displays constraints locations in different colors. The color meaning is as follows:

- White: Display constraints location in selected or highlighted;
- Blue: Display reserved constraints, indicating that the location cannot be occupied again;
- Light blue: Display IO and Primitive constrained in a grid or range.

Chip Array supports right-clicking functions:

- Zoom In: Zoom in Chip Array
- Zoom Out: Zoom out Chip Array;
- Zoom Fit: Zoom in/out Chip Array.
- Show Constraints View: Show instances constraints view of Chip Array;
- Show Place View: Show instances place view of Chip Array;
- Show Multi-View: Show instances constraints view and and place view of Chip Array;
- Show In-Out Connection: Show and select instances in-out connection

in Place View;

- Show In Connection: Show and select instances in connection in Place View;
- Show Out Connection: Show and select instances out connection in Place View;
- Convert Place To Constraints: Convert the place to constraints of the selected instance in Place View;
- Unhighlight All: Remove highlight;
- Copy Location: Copy the selected location or area.

Show Place View also shows Lut and Reg density, as shown in Figure 3-51:

- ALL Instance: Shows place of all instances. Light green indicates less than five, green indicates six to ten, and dark green indicates more than ten;
- Only Lut: Shows place of all Lut. Light green indicates less than two, green indicates three to four, and dark green indicates more than four;
- Only Dff: Shows place of all Reg. Light green indicates less than two, green indicates three to four, and dark green indicates more than four;

Note!

- Show Place View and Show Multi-View are only effective when FloorPlanner is started after running Place & Route. Otherwise, they are greyed out.
- Show In-Out Connection, Show In Connection, Show Out Connection and Convert Place To Constraints can only be used if an instance is selected when Show Place View->All Instance view opens.
- If grid, block, reg, or lut etc. in view are selected, "Copy Location" in right-click menu is available;
- If no grid is selected, "Copy Location" is not available, as shown in Figure 3-50;
- Select area via "Ctrl" + left mouse button. Right-click to select "Copy Location" to copy and paste.

Figure 3-50 Chip Array Right-clicking

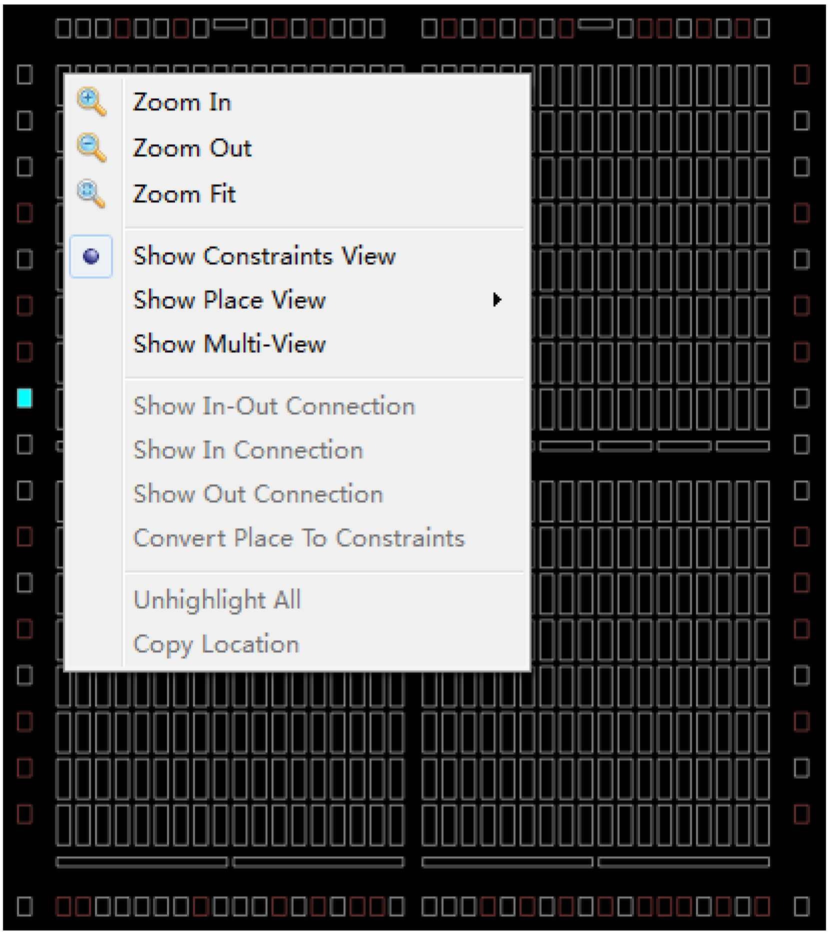
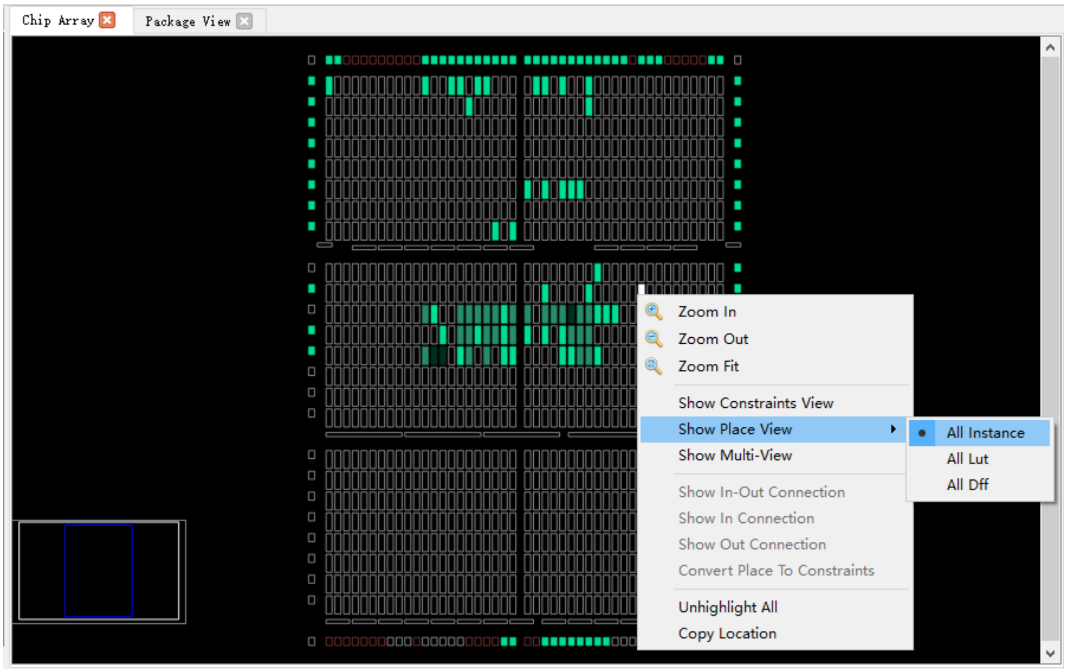
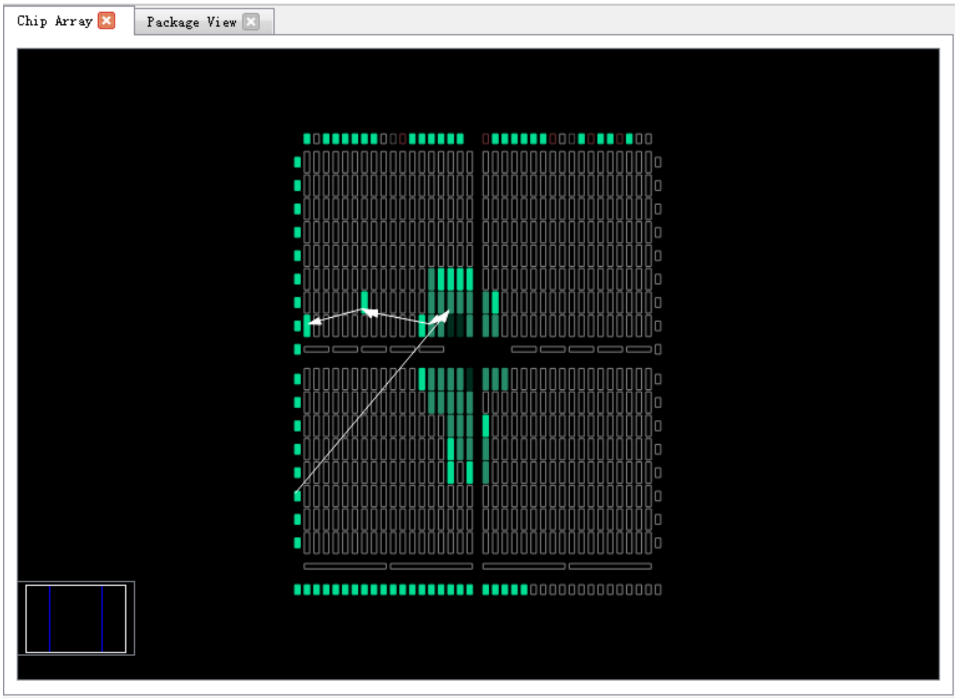


Figure 3-51 Show Place View



Chip Array also can highlight timing paths, as shown in Figure 3-52.

Figure 3-52 Timing Path Highlighted



3.3.5 Constraints Window

Constraints window includes eight constraints views, such as, "I/O Constraints" , "Primitive Constraints", "Group" Constraints, etc., which are used to display constraints and provide constraints editor and drag function. Brief introduction of each view is as follows:

I/O Constraints

I/O Constraints is to constrain ports. I/O constraint view is as shown in Figure 3-53, and the functions are as follows:

- Display IO Port attributes and constraints in user design, such as Direction, Bank, IOType, PullMode, etc.
- Support to edit constraints locations and attribute, etc.;
- Change constraints by dragging, double-clicking, etc.

Note!

- Set I/O location by dragging or double-clicking;
- Display IO name when dragging;
- When I/O is dragged into Chip Array window, the location where I/O can be placed is brightened, and the color of the location where I/O cannot be placed is unchanged.

- When I/O is dragged into Package View window, the color of the location where I/O can be placed remains unchanged and the location where I/O can not be placed becomes dark.
- After setting, the constraints location in Chip Array is highlighted in yellow, and the constraints location in Package View is highlighted in orange.

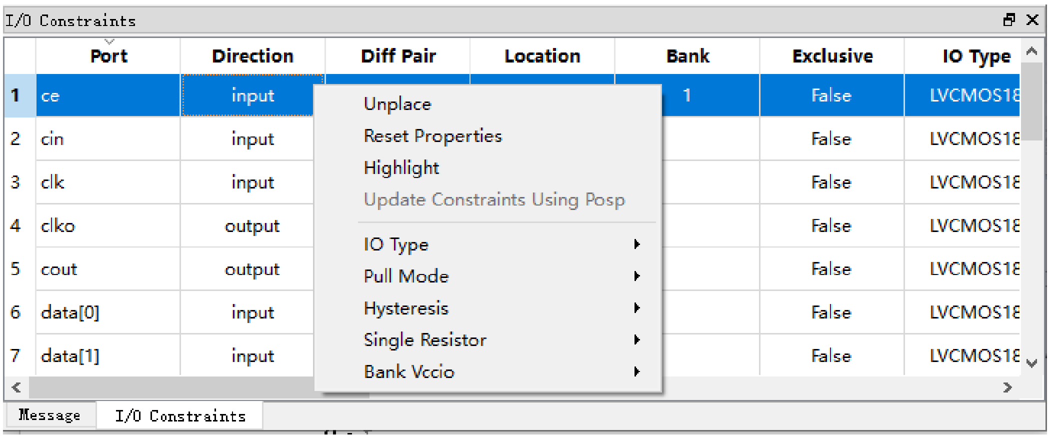
The details of the right-clicking menu are as follows:

- Unplace: Cancel place;
- Reset Properities: Reset Port properities;
- Highlight: Highlight constraints location;
- Update Constraints Using Posp: Update constraints according to posp file;
- IO Type: Set the level;
- Drive: Set drive voltage;
- Pull Mode: Set pull-up mode;
- PCI Clamp: Set the switch of PCI protocol;
- Hysteresis: Set hysteresis;
- Open Drain: Set the switch of open-drain circuit;
- Slew Rate: Set the voltage slew rate;
- Vref: Set reference voltage;
- Single Resistor: Set the switch of signal resistor;
- Diff Resistor: Set the switch of differential resistor;
- Bank Vccio: Set BANK voltage.

Note!

- Users can modify port properities in batches by right-clicking.
- Update Constraints Using Posp can only be used after posp file is loaded. Otherwise, it is greyed out.

Figure 3-53 I/O Constraints View



I/O Constraints

	Port	Direction	Diff Pair	Location	Bank	Exclusive	IO Type
1	ce	input			1	False	LVCMOS18
2	cin	input				False	LVCMOS18
3	clk	input				False	LVCMOS18
4	clko	output				False	LVCMOS18
5	cout	output				False	LVCMOS18
6	data[0]	input				False	LVCMOS18
7	data[1]	input				False	LVCMOS18

Primitive Constraints

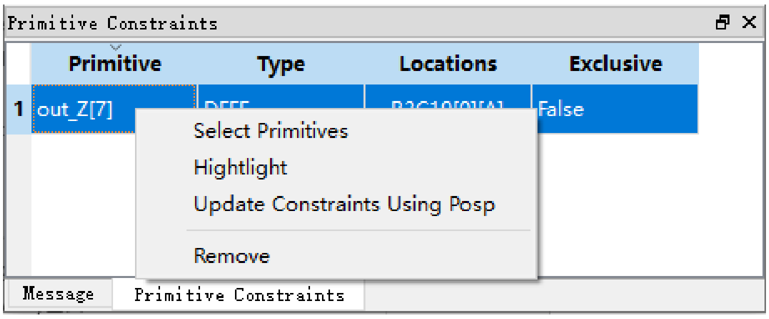
Primitive Constraint View is as shown in Figure 3-54, and the functions are as follows:

- Display the name, type, location, and Exclusive of all Primitive constraints;
- Support to edit.

Note!

- Modify the locations by dragging or double-clicking;
- Set Exclusive by double-clicking;
- You can highlight, remove, add and update constraints by right-clicking.
- Syntax and legality will be checked for the locations when manually writing primitive constraints, and error message dialog box are as shown in Figure 3-20 and Figure 3-21.

Figure 3-54 Primitive Constriaints View



Primitive Constraints

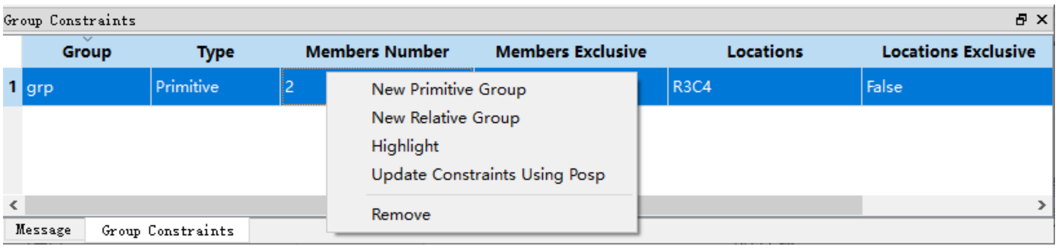
	Primitive	Type	Locations	Exclusive
1	out_Z[7]	DFFF	R2C10[0][A]	False

Group Constraints

Primitive Constraints view is as shown in Figure 3-55, and the functions are as follows:

- The view displays the name, type, number of primitive, location, and Exclusive, includes primitive and relative groups. As shown in Figure 3-19 and Figure 3-23, double-click the group to edit constraints.
- You can highlight, remove, add and update constraints by right-clicking.

Figure 3-55 Group Constraints View



Group Constraints

	Group	Type	Members Number	Members Exclusive	Locations	Locations Exclusive
1	grp	Primitive	2		R3C4	False

Resource Reservation

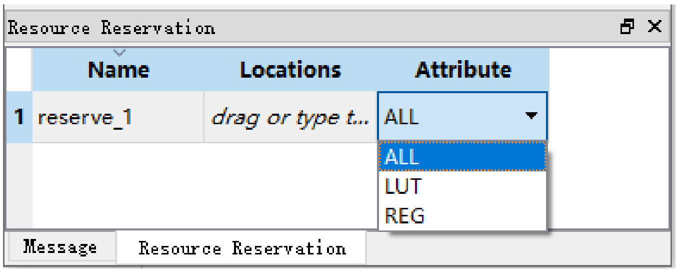
Resource Reservation view is as shown in Figure 3-56, and the functions are as follows:

- The view can display reserved constraints locations;
- You can highlight, remove, add and update constraints by right-clicking.
- Users cannot modify name.

Note!

You can change the locations by dragging or double-clicking;

Figure 3-56 Resource Reservation View



Resource Reservation

	Name	Locations	Attribute
1	reserve_1	*drag or type t...*	ALL

Clock Assignment

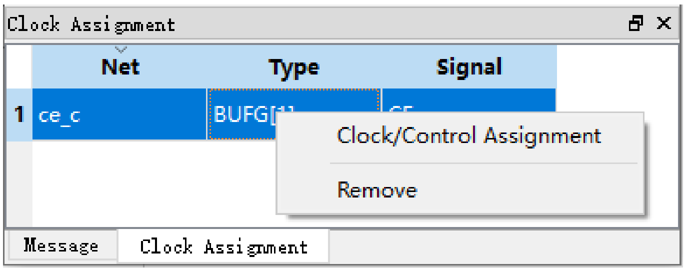
Clock Assignement view is as shown in Figure 3-57, and the functions are as follows:

- The view display all clock constraints;
- You can highlight, remove, add and update constraints by right-clicking.

Note!

- Double-click to edit;
- Dragging is not supported if there is no location.
- Clock constraint creation is as shown in Figure 3-25.

Figure 3-57 Clock Assignment View



Clock Assignment

	Net	Type	Signal
1	ce_c	BUFG[1	CE

Quadrant Constraints

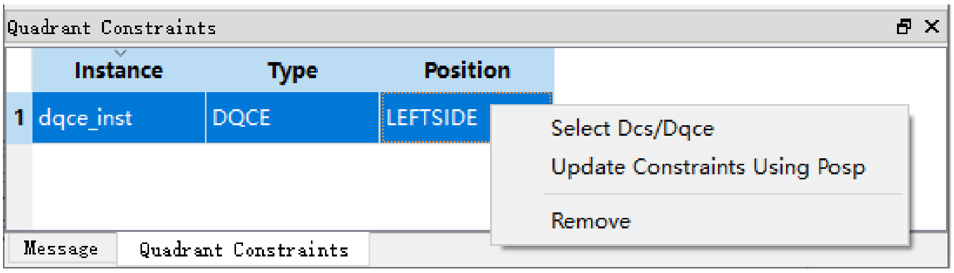
Quadrant Constraints view is as shown in Figure 3-58, and the functions are as follows:

- Display all quadrant constraints, including Instance name, type, and locations;
- You can remove and add constraints by right-clicking.

Note!

- Quadrant constraints are only valid for DCS and DQCE.
- Quadrant constraint creation is as shown in Figure 3-26 and Figure 3-27.

Figure 3-58 Quadrant Constraints View



Quadrant Constraints

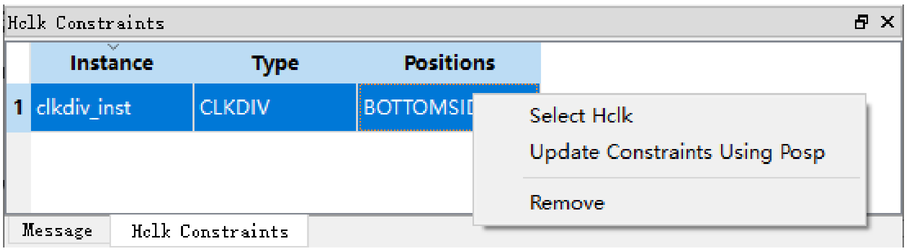
	Instance	Type	Position
1	dqce_inst	DQCE	LEFTSIDE

Hclk Constraints

Hclk Constraints view is as shown in Figure 3-59, and the functions are as follows:

- The view can display the instance location constraints of Hclk, including Instance name, type, and quadrant location;
- You can remove and add constraints by right-clicking. Hclk constraints creation is shown in Figure 3-28.

Figure 3-59 Hclk Constraints View



Hclk Constraints

	Instance	Type	Positions
1	clkdiv_inst	CLKDIV	BOTTOMSID

Note!

CLKDIV constraints are only valid for CLKDIV and CLKDIV primitives.

Vref Constraints

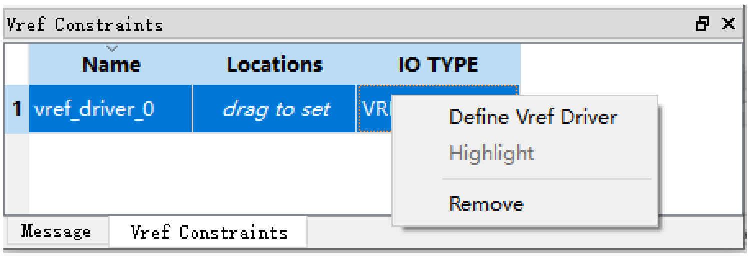
Vref Constraints view is as shown in Figure 3-60, and the functions are as follows:

- The view can display Vref Driver defined by users, such as, Vref name and location;
- You can highlight, remove, and add constraints by right-clicking.

Note!

Locations can only be set by dragging.

Figure 3-60 Vref Constraints View

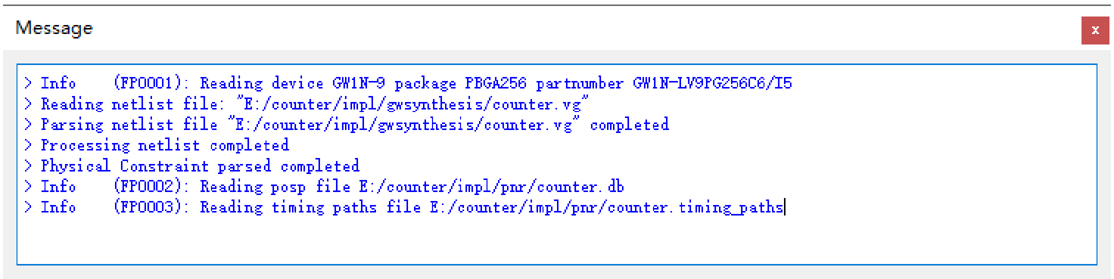


	Name	Locations	IO TYPE
1	vref_driver_0	drag to set	VR

3.4 Message Window

The message window is as shown in Figure 3-61, and it displays the output.

Figure 3-61 Message Window



```
> Info    (FP0001): Reading device GW1N-9 package PBGA256 partnumber GW1N-LV9PG256C6/I5
> Reading netlist file: "E:/counter/impl/gwsynthesis/counter.vg"
> Parsing netlist file "E:/counter/impl/gwsynthesis/counter.vg" completed
> Processing netlist completed
> Physical Constraint parsed completed
> Info    (FP0002): Reading posp file E:/counter/impl/pnr/counter.db
> Info    (FP0003): Reading timing paths file E:/counter/impl/pnr/counter.timing_paths
```

4 Create Constraints

The Constraints editor supports the constraints such as IO, Primitive, Group, Resource, Clock, Quadrant, Hclk, and Vref. You can create constraints in Constraints menu bar. See 3.3.1 Menu Bar for details.

Note!

You can also create constraints in other ways. This section describes how to generate constraints by dragging.

4.1 Constraints Examples

Take the user design counter.v for an instance to introduce how to create constraints.

```
// Eight bit counter example 1


module counter1(out, cout, data, load, cin, clk, ce, clko);
output [7:0] out;
output cout;
output clko;
input ce;
input [7:0] data;
input load, cin, clk;


reg [7:0] out;
```

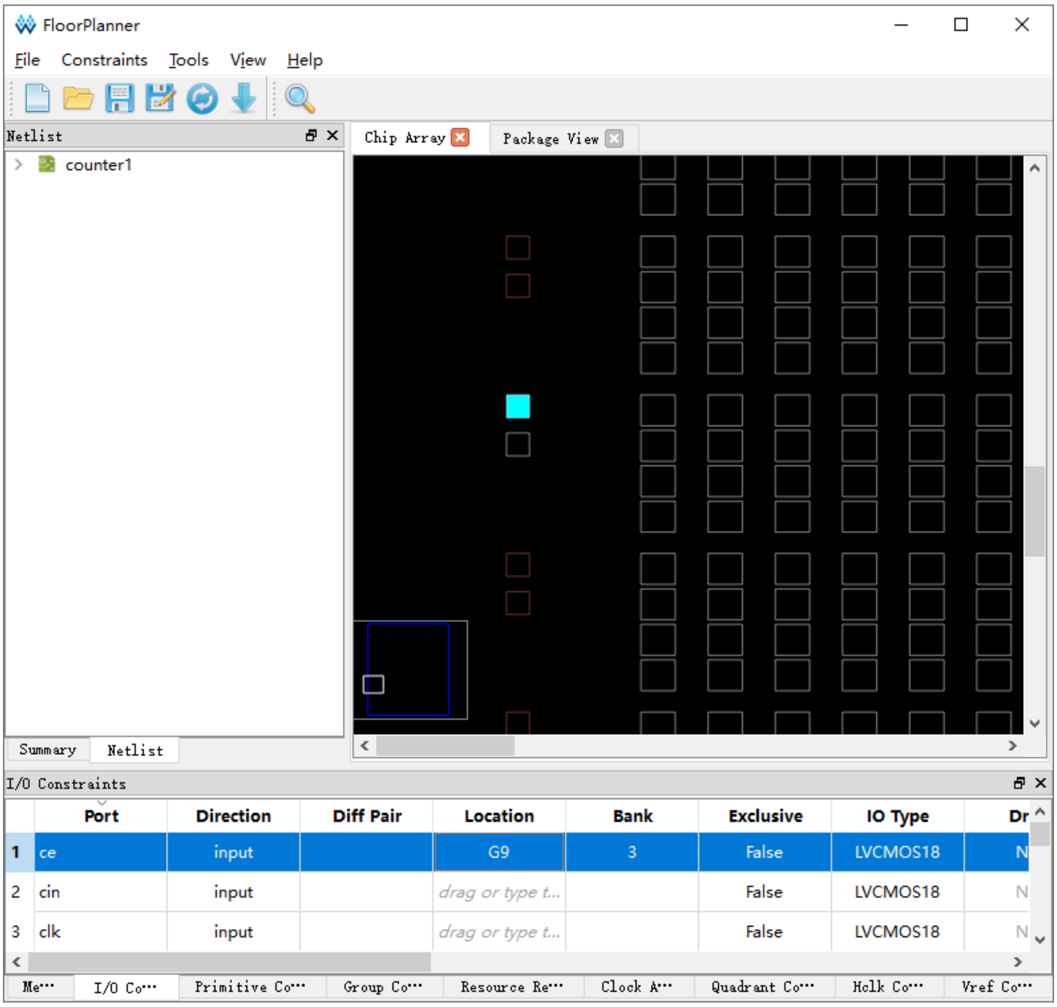
```
always@(posedge clk)
begin
    if (load)
        out = data;
    else
        out = out + cin;

end

// all bits of out must be one and the
// carry in must be on to generate a
// carry out
assign cout = &out & cin;

wire clkout;
CLKDIV clkdiv_inst(
     .CLKOUT(clkout),
     .HCLKIN(clk),
     .RESETN(1'b1),
     .CALIB(1'b0)
);

defparam clkdiv_inst.DIV_MODE = "2";
defparam clkdiv_inst.GSREN = "false";

DQCE dqce_inst (
     .CLKOUT(clko),
     .CLKIN(clkout),
     .CE(ce)
);


Endmodule
```

4.2 Create I/O Constraints

Drag to Chip Array to create I/O constraints.

1. Click IO Constraints to zoom in Chip Array to macrocell mode.
2. Select Port "ce" and drag to the G9 in Chip Array, as shown in Figure 4-1.
3. Location for Port "ce" is displayed as G9.

Figure 4-1 Drag to Chip Array to Create I/O Constraints

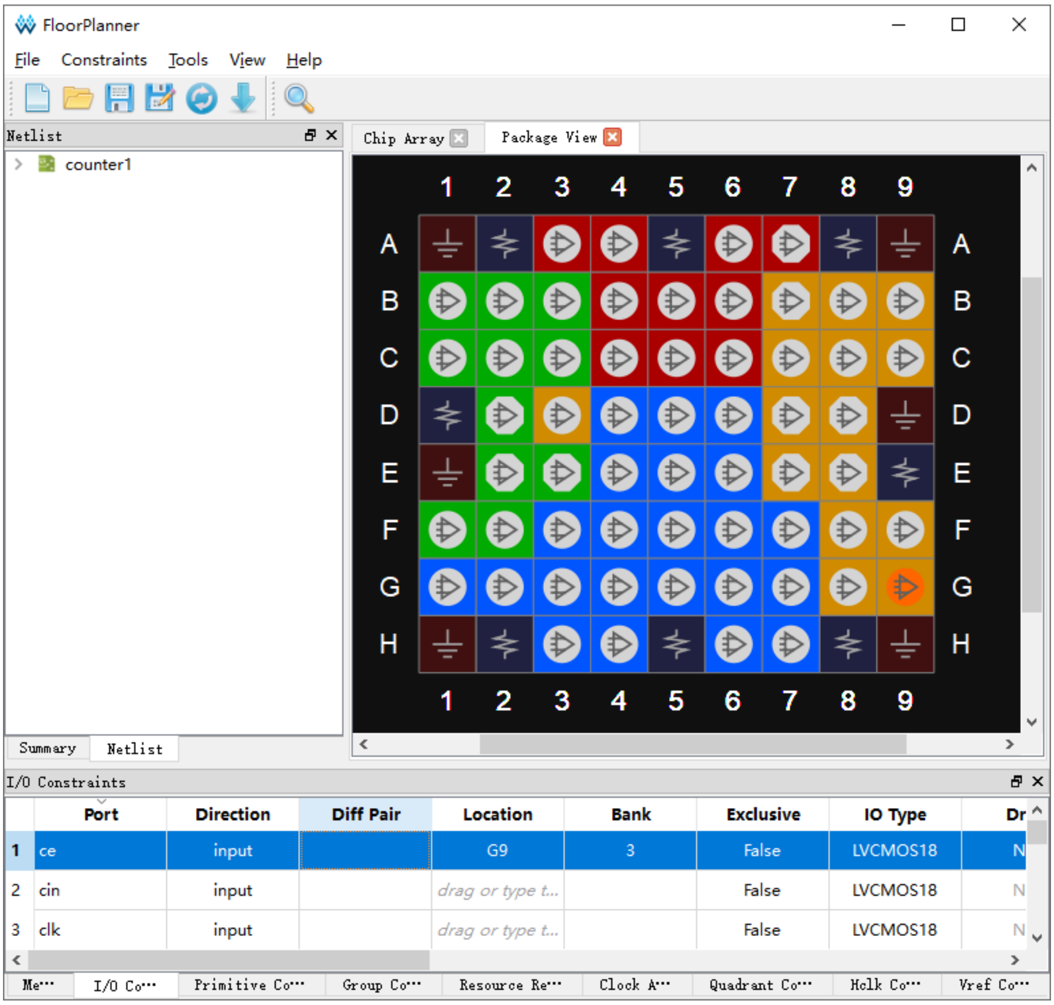


	Port	Direction	Diff Pair	Location	Bank	Exclusive	IO Type	Dr
1	ce	input		G9	3	False	LVCMOS18	N
2	cin	input		drag or type t…		False	LVCMOS18	N
3	clk	input		drag or type t…		False	LVCMOS18	N

Drag to Package View to create I/O constraints.

1. Click IO Constraints;
2. Select Port "ce" and drag to the G9 in Package View, as shown in Figure 4-2.
3. Location for Port "ce" is displayed as G9.

Figure 4-2 Drag to Package View to Create I/O Constraints

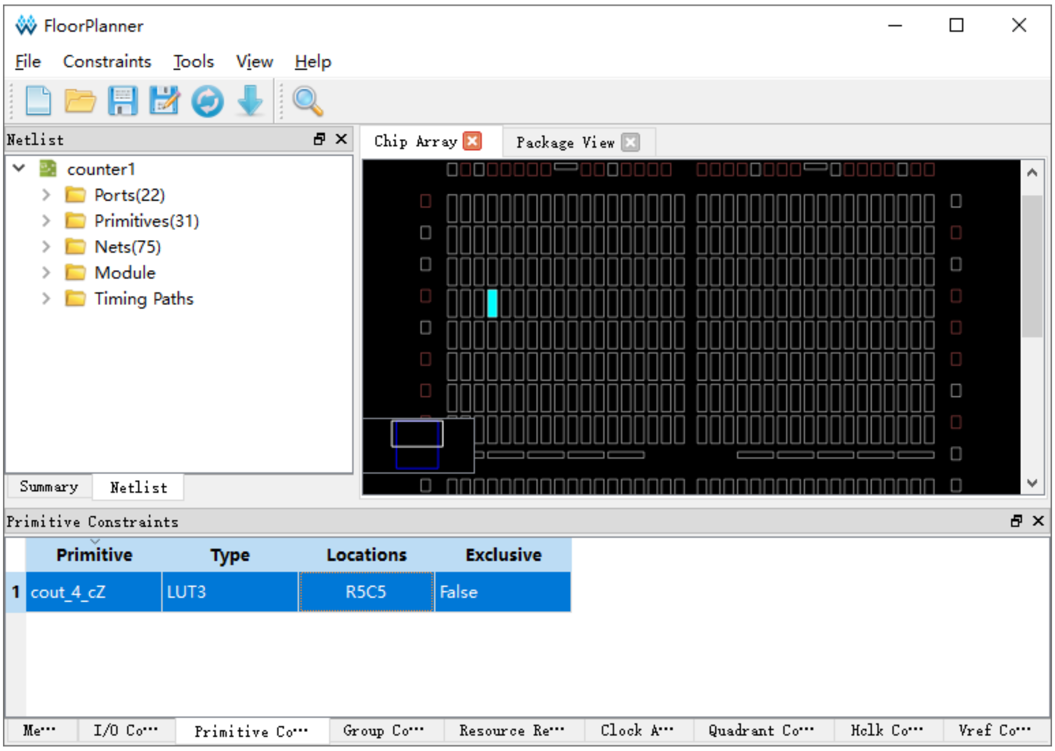


	Port	Direction	Diff Pair	Location	Bank	Exclusive	IO Type	Dr
1	ce	input		G9	3	False	LVCMOS18	N
2	cin	input		drag or type t...		False	LVCMOS18	N
3	clk	input		drag or type t...		False	LVCMOS18	N

4.3 Create Primitive Constraints

1. You can right-click the menu in Primitive Constraints to select "Select Primitives". Primitive Finder pops up. You can select Primitive "cout_4_cZ" and click "OK".
2. Select the created primitive constraints and drag to the R5C5 in Chip Array, as shown in Figure 4-3.
3. Location for primitive "cout_4_cZ" is displayed as R5C5.

Figure 4-3 Drag to Chip Array to Create Primitive Constraints

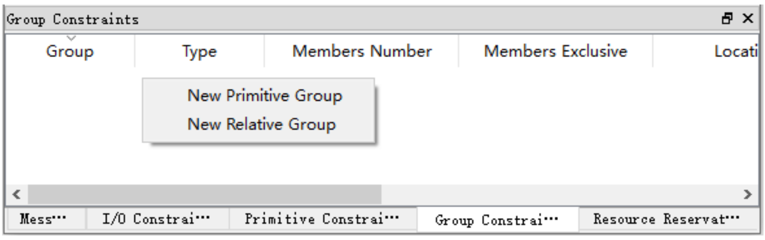


	Primitive	Type	Locations	Exclusive
1	cout_4_cZ	LUT3	R5C5	False

4.4 Create Group Constraints

As shown in Figure 4-4, you can create Primitive Group and Relative Group by right-clicking in Group Constraints.

Figure 4-4 Group Constraints Right-clicking

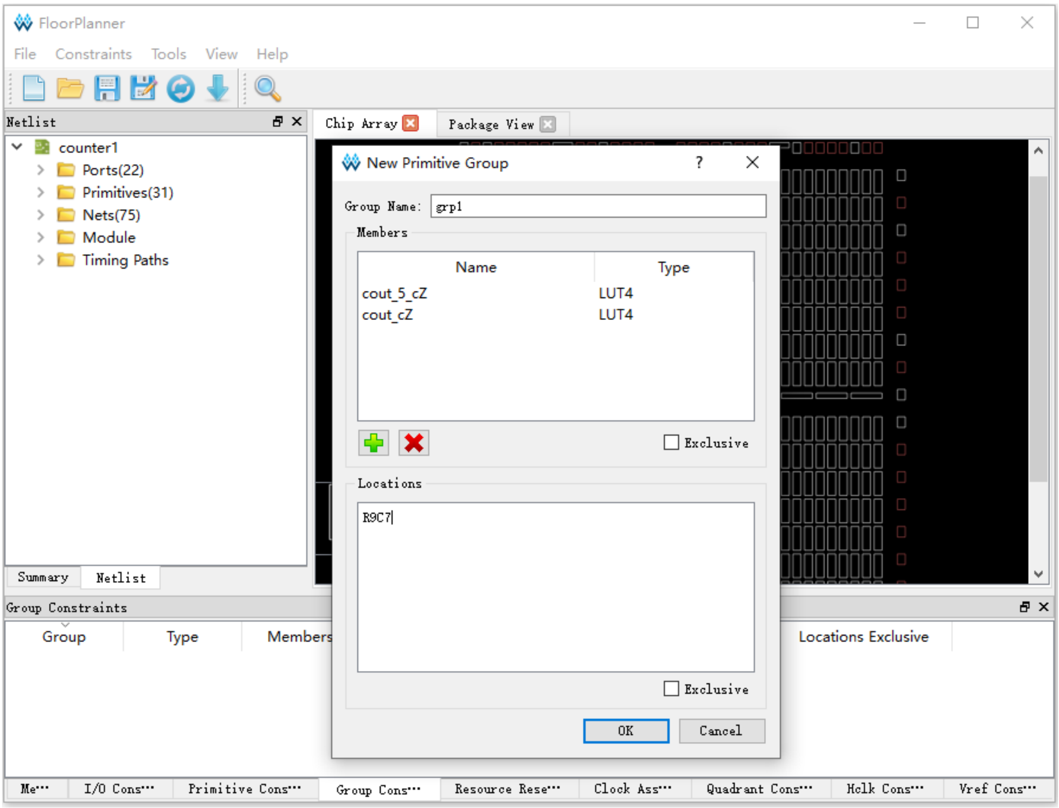


4.4.1 Create Primitive Group Constraints

1. Right-click Group Constraints and click "New Primitive Group", then "Edit Primitive Group" pops up;
2. Enter "grp1" and click "+", then Primitive Finder pops up;
3. Select "cout_5_cZ" and "cout_cZ and click"OK", then add them to Members list.
4. Enter R9C7 in Locations as shown in Figure 4-5;

5. Click "OK" in "Edit Primitive Group" to create primitive group constraints, as shown in Figure 4-6.

Figure 4-5 Create Primitive Group Constraints

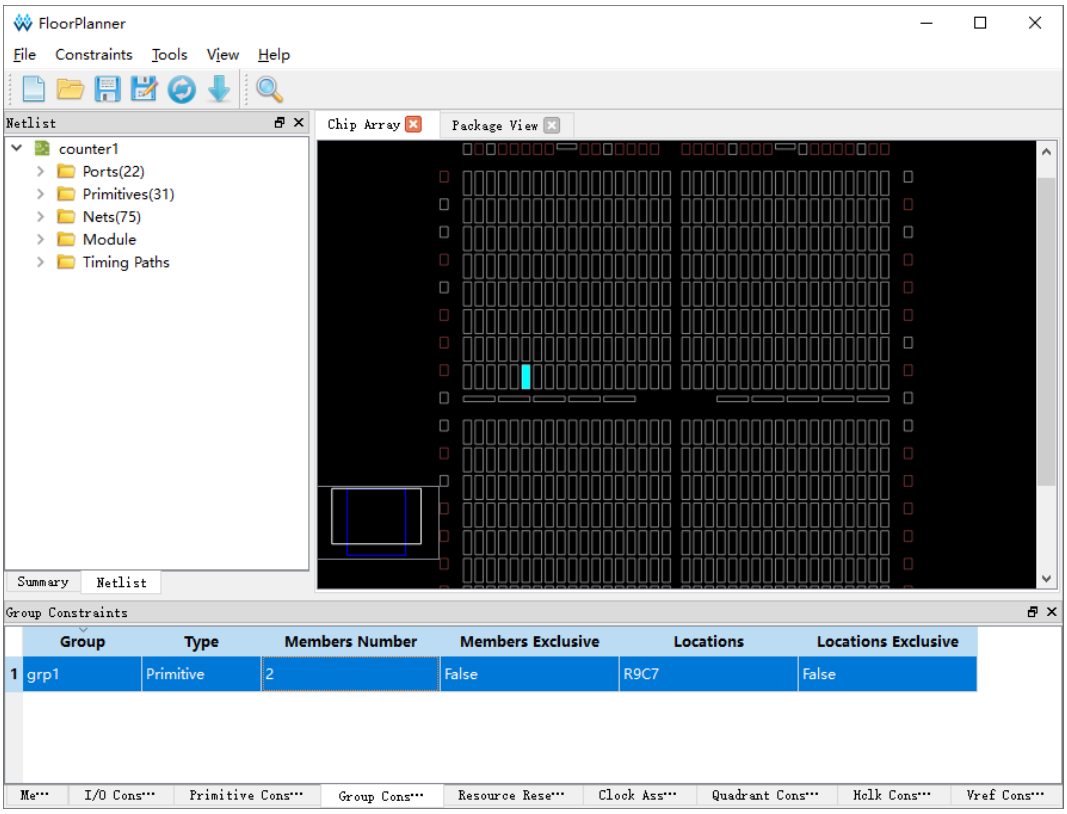


Group Name: grp1

Name	Type
cout_5_cZ	LUT4
cout_cZ	LUT4

R9C7

Figure 4-6 Primitive Group Constraints



	Group	Type	Members Number	Members Exclusive	Locations	Locations Exclusive
1	grp1	Primitive	2	False	R9C7	False

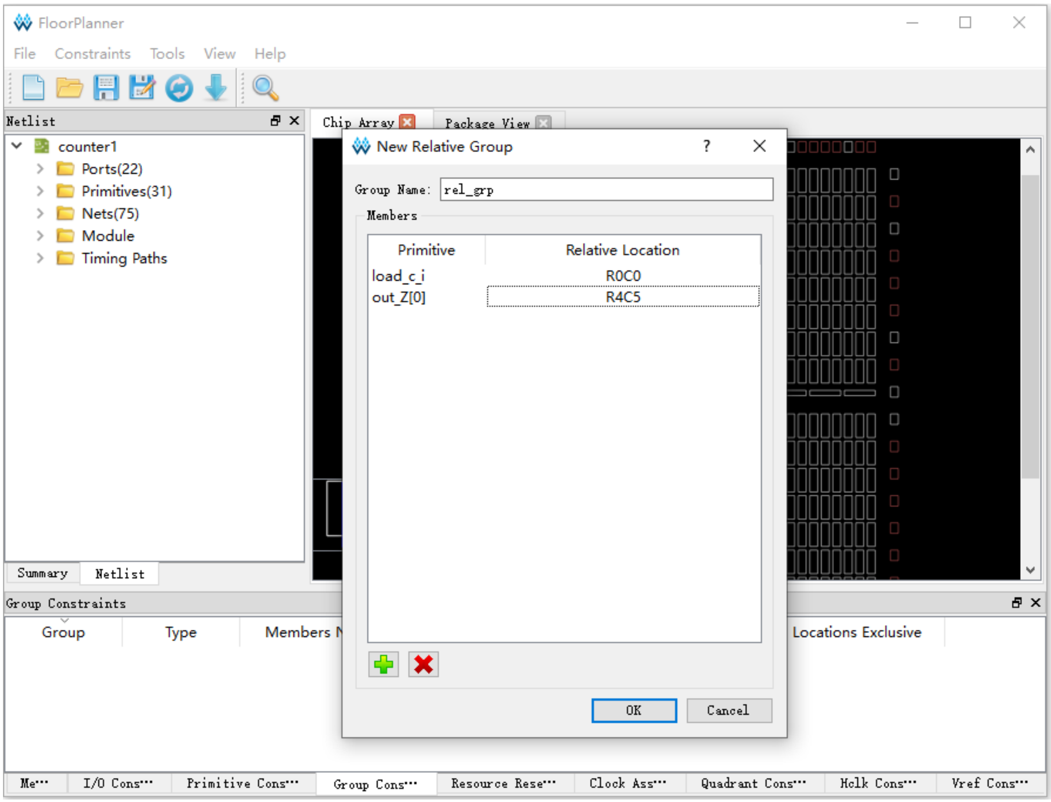
Note!

The location in Primitive Group Constraints can only be entered manually or copied from Chip Array, but can not be generated by dragging

4.4.2 Create Relative Group Constraints

1. Right-click Group Constraints and click "New Relative Group", and "Edit Primitive Group" pops up.
2. Enter "rel_grp" and click "+", and "Primitive Finder" pops up;
3. Select the primitives "load_c_i" and "out_Z[0]" in Primitive Finder and click "OK", then add them to the Member list.
4. Add "R0C0" and "R4C5" to the Primitives, as shown in Figure 4-7;
5. Click "OK" in "Edit Relative Group" to create relative primitive group constraints, as shown in Figure 4-8.

Figure 4-7 Create Relative Group Constraints

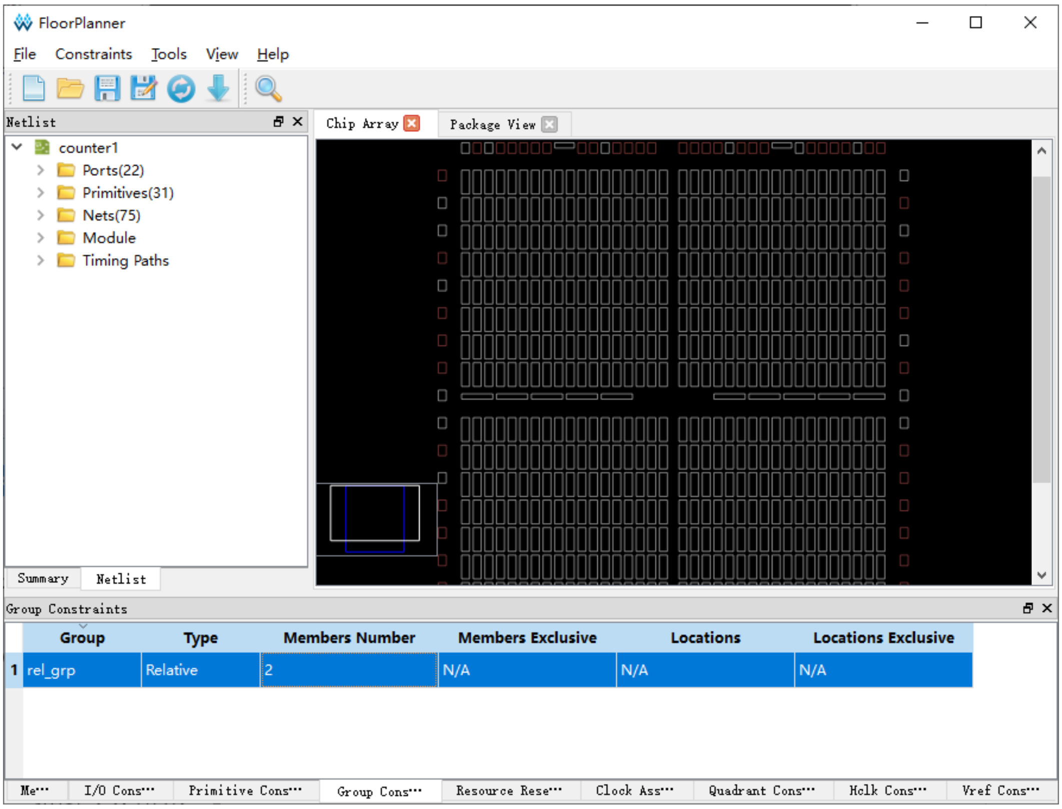

FloorPlanner
File Constraints Tools View Help
Netlist
counter1
Ports(22)
Primitives(31)
Nets(75)
Module
Timing Paths
Summary Netlist
Chip Array
Package View
New Relative Group
Group Constraints
Group Type Members N Locations Exclusive
OK Cancel
Me··· I/O Cons··· Primitive Cons··· Group Cons··· Resource Rese··· Clock Ass··· Quadrant Cons··· Hclk Cons··· Vref Cons···

Group Name: rel_grp

Members

Primitive	Relative Location
load_c_i	R0C0
out_Z[0]	R4C5

Figure 4-8 Relative Group Constraints

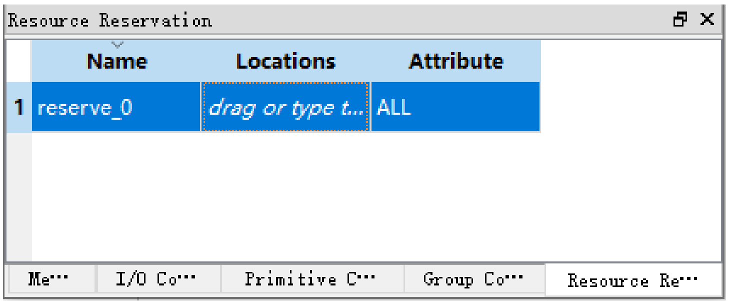

FloorPlanner
File Constraints Tools View Help
Netlist
counter1
Ports(22)
Primitives(31)
Nets(75)
Module
Timing Paths
Summary Netlist
Chip Array
Package View
Group Constraints
Me··· I/O Cons··· Primitive Cons··· Group Cons··· Resource Rese··· Clock Ass··· Quadrant Cons··· Hclk Cons··· Vref Cons···

	Group	Type	Members Number	Members Exclusive	Locations	Locations Exclusive
1	rel_grp	Relative	2	N/A	N/A	N/A

4.5 Create Resource Reservation

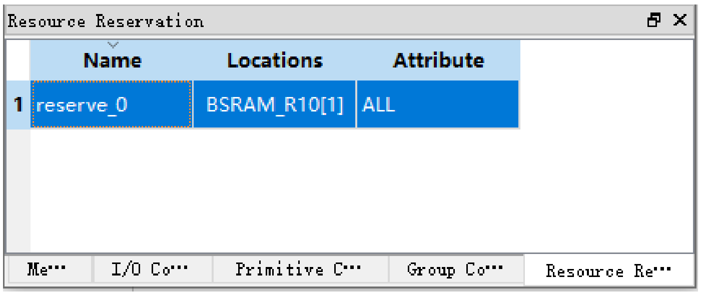
1. Right-click "Reserve Resources" and click "Reserve Resources" to add resource reservation constraints, as shown in Figure 4-9;
2. Select the created resource reservation constraints and drag it to a location in Chip Array. As shown in Figure 4-10, drag to BSRAM_R10[1] to generate constraints.

Figure 4-9 Create Resource Reservation



	Name	Locations	Attribute
1	reserve_0	*drag or type t...*	ALL

Figure 4-10 Resource Reservation

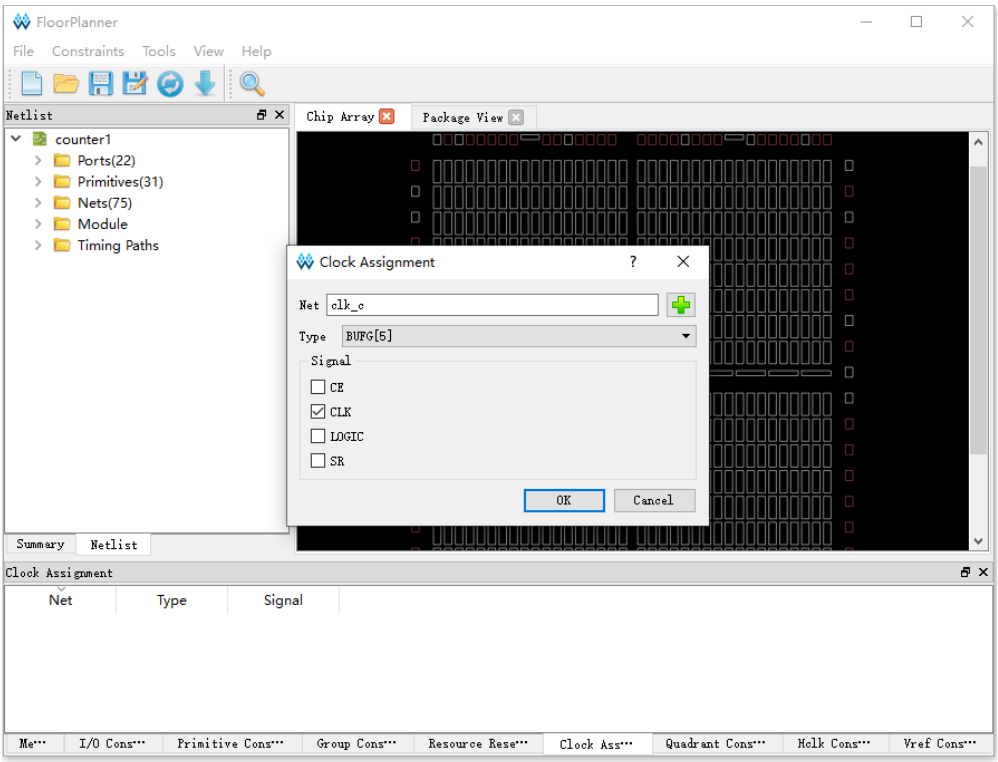


	Name	Locations	Attribute
1	reserve_0	BSRAM_R10[1]	ALL

4.6 Create Clock Assignment

1. Right-click Clock Assignment and select "Clock/Control Assignment", then "Clock Assignment" dialog box pops up.
2. Click "➕" and "Net Finder" dialog box pops up. Select a Net and click "OK" to add.
3. Select clock type and set signal type, as shown in Figure 4-11.
4. Click "OK" to add constraints to Clock Assignment, as shown in Figure 4-12.

Figure 4-11 Modify Clock Assignment Constraints Create


FloorPlanner
File Constraints Tools View Help
Netlist
counter1
Ports(22)
Primitives(31)
Nets(75)
Module
Timing Paths
Chip Array
Package View
Summary Netlist
Clock Assignment
Net Type Signal
Me··· I/O Cons··· Primitive Cons··· Group Cons··· Resource Rese··· Clock Ass··· Quadrant Cons··· Hclk Cons··· Vref Cons···

Clock Assignment

Net: clk_c

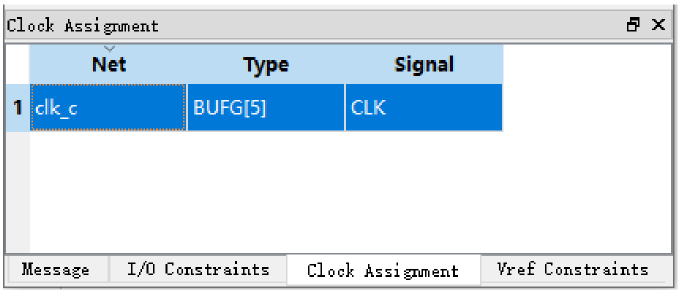
Type: BUFG[5]

Signal

- ☐ CE
- ☑ CLK
- ☐ LOGIC
- ☐ SR

OK Cancel

Figure 4-12 Clock Assignment Constraints



	Net	Type	Signal
1	clk_c	BUFG[5]	CLK

4.7 Create Quadrant Constraints

Quadrant Constraints only constrain the following two types of instance:

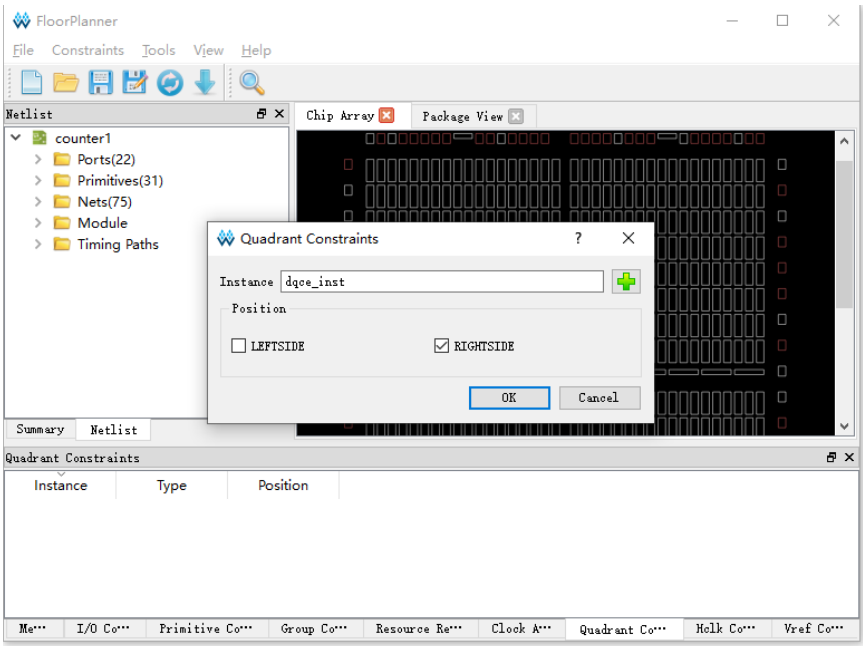
- Dcs
- Dqce

1. Right-click Quadrant Constraints and select "Select Dcs/Dqce", then Quadrant Constraints dialog box pops up;

2. Click "✚" and Dcs/Dqce pops up. Select Instance and click ″OK″ in Dcs/Dqce.

3. Select the quadrant in "Position", as shown in Figure 4-13.
4. click "OK" to add this constraints to Quadrant Constraints, as shown in Figure 4-14.

Figure 4-13 Create Quadrant Constraints



Quadrant Constraints

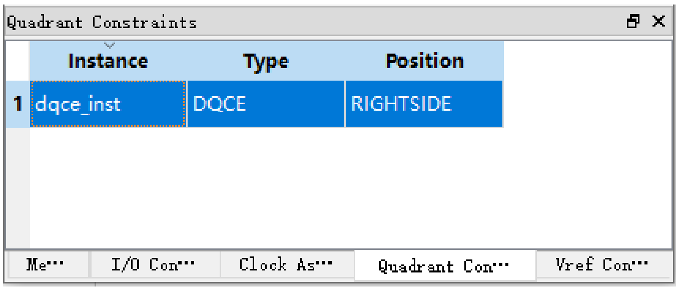
Instance dqce_inst

Position

☐ LEFTSIDE ☑ RIGHTSIDE

OK Cancel

Figure 4-14 Quadrant Constraints



	Instance	Type	Position
1	dqce_inst	DQCE	RIGHTSIDE

4.8 Create Hclk Constraints

Hclk Constraints only constrain the following types of instance:

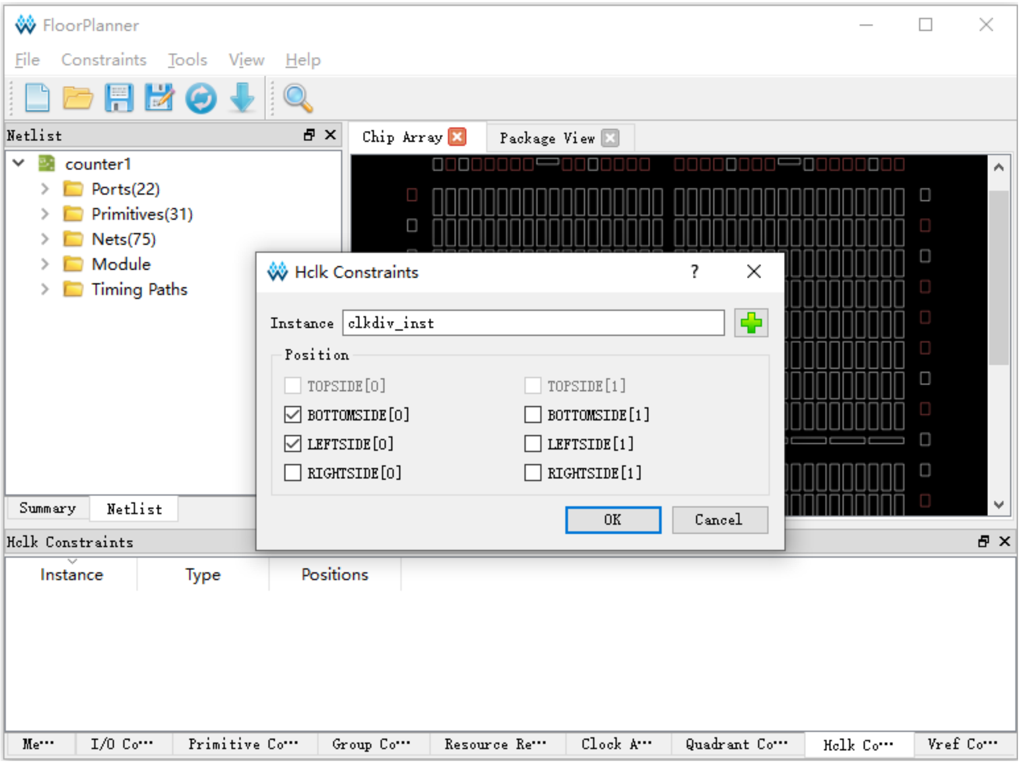
- Clkdiv
- Dlldly

The steps are as follows:

1. Right-click Hclk Constraints and select "Select Hclk", then Hclk Constraints pops up;
2. Click "✚" and Hclk Constraints dialog box pops up, then click "OK".
3. Select a position in "Position", as shown in Figure 4-15.

4. Click "OK" to add the constraints to Hclk Constraints, as shown in Figure 4-16.

Figure 4-15 Create Hclk Constraints


FloorPlanner
File Constraints Tools View Help
Netlist
counter1
Ports(22)
Primitives(31)
Nets(75)
Module
Timing Paths
Summary Netlist
Chip Array
Package View
Hclk Constraints
Me··· I/O Co··· Primitive Co··· Group Co··· Resource Re··· Clock A··· Quadrant Co··· Hclk Co··· Vref Co···

Hclk Constraints

Instance clkdiv_inst

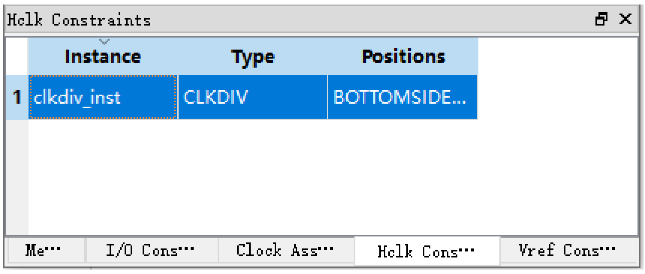
Position

□ TOPSIDE[0]	□ TOPSIDE[1]
☑ BOTTOMSIDE[0]	□ BOTTOMSIDE[1]
☑ LEFTSIDE[0]	□ LEFTSIDE[1]
□ RIGHTSIDE[0]	□ RIGHTSIDE[1]

OK Cancel

Instance	Type	Positions

Figure 4-16 Hclk Constraints



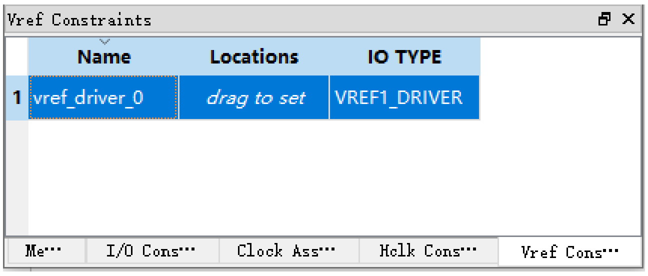
	Instance	Type	Positions
1	clkdiv_inst	CLKDIV	BOTTOMSIDE...

4.9 Create Vref Constraints

Drag to Chip Array to create Vref Constraints:

1. Right-click Vref Constraints and select "Define Vref Driver" to add the constraints to Vref Constraints, as shown in Figure 4-17;
2. Zoom in Chip Array to macrocell mode. Select the created Vref Constraints and drag it to the B7 in Chip Array. The location of the Vref Constraints is displayed as B7, as shown in Figure 4-18.

Figure 4-17 Create Vref Constraints

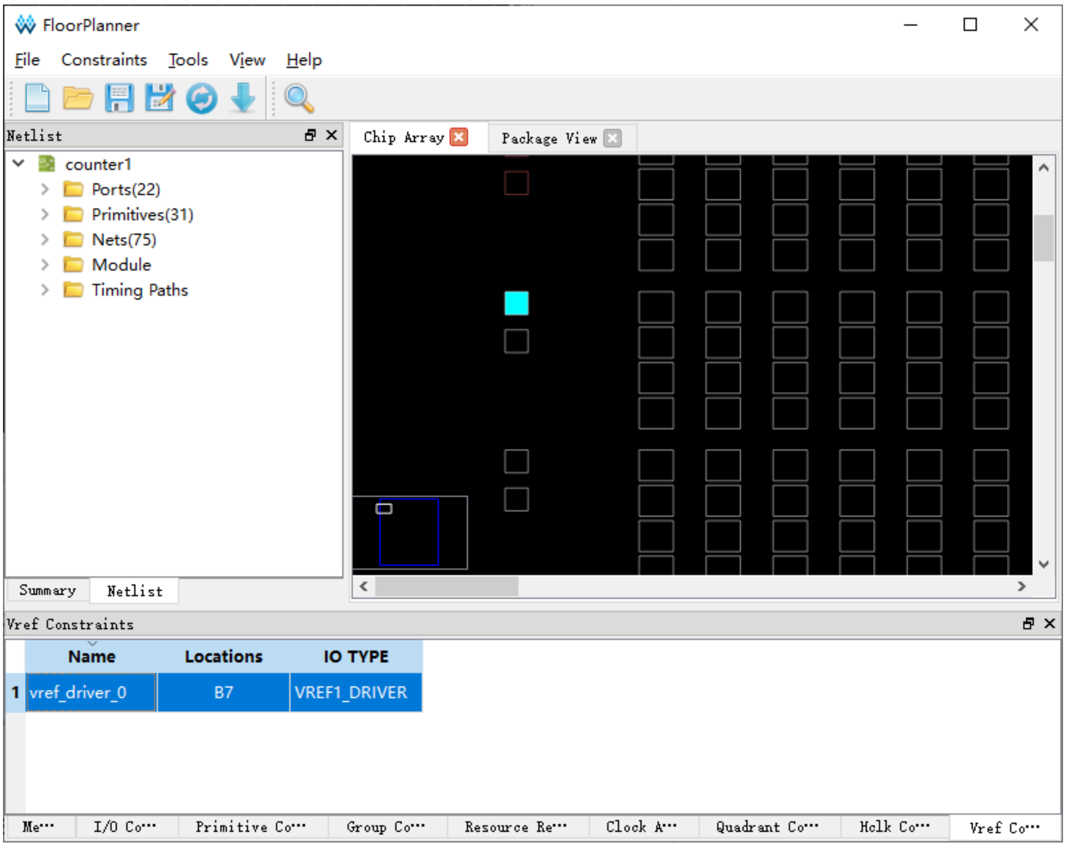


Vref Constraints

	Name	Locations	IO TYPE
1	vref_driver_0	*drag to set*	VREF1_DRIVER

You can customize Vref constraint name, but Vref name is not allowed to repeat; The system will check, if name repeats, prompt will pop up, as shown in Figure 4-20.

Figure 4-18 Drag to Chip Array to Generate Vref Constraints Location



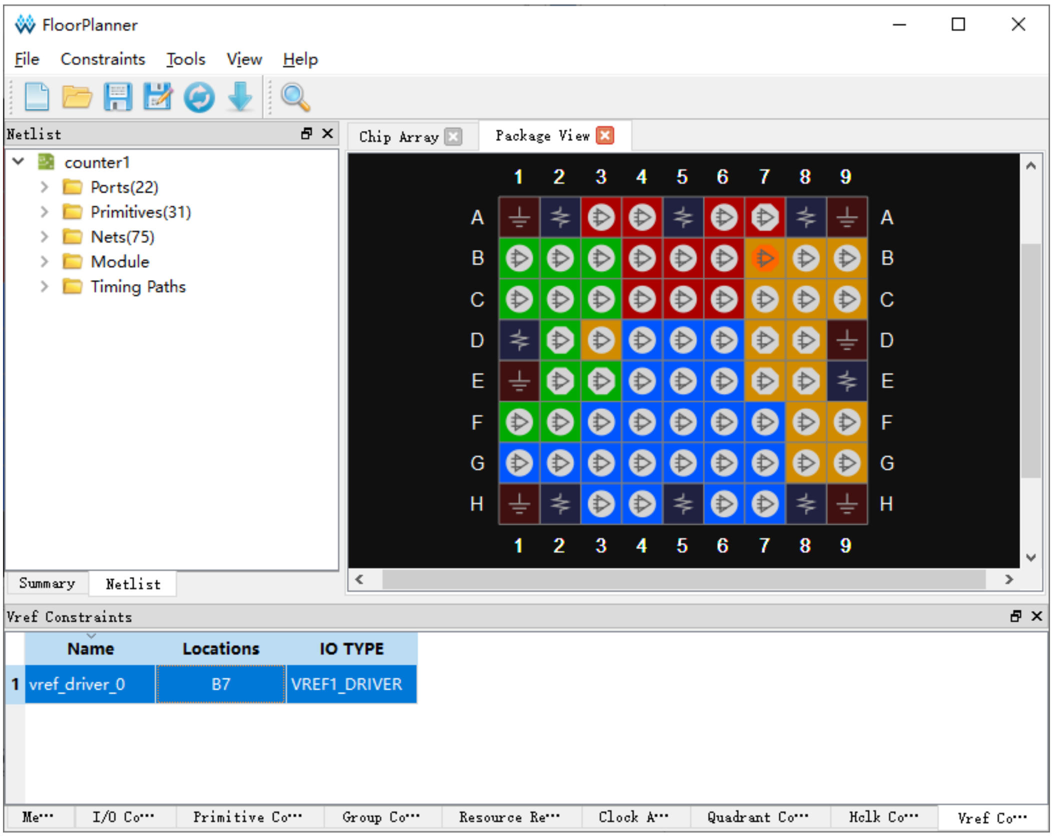
	Name	Locations	IO TYPE
1	vref_driver_0	B7	VREF1_DRIVER

Drag to Package View to create Vref constraints:

1. Right-click Vref Constraints and select "Define Vref Driver" to add the constraints to Vref Constraints, as shown in Figure 4-17;

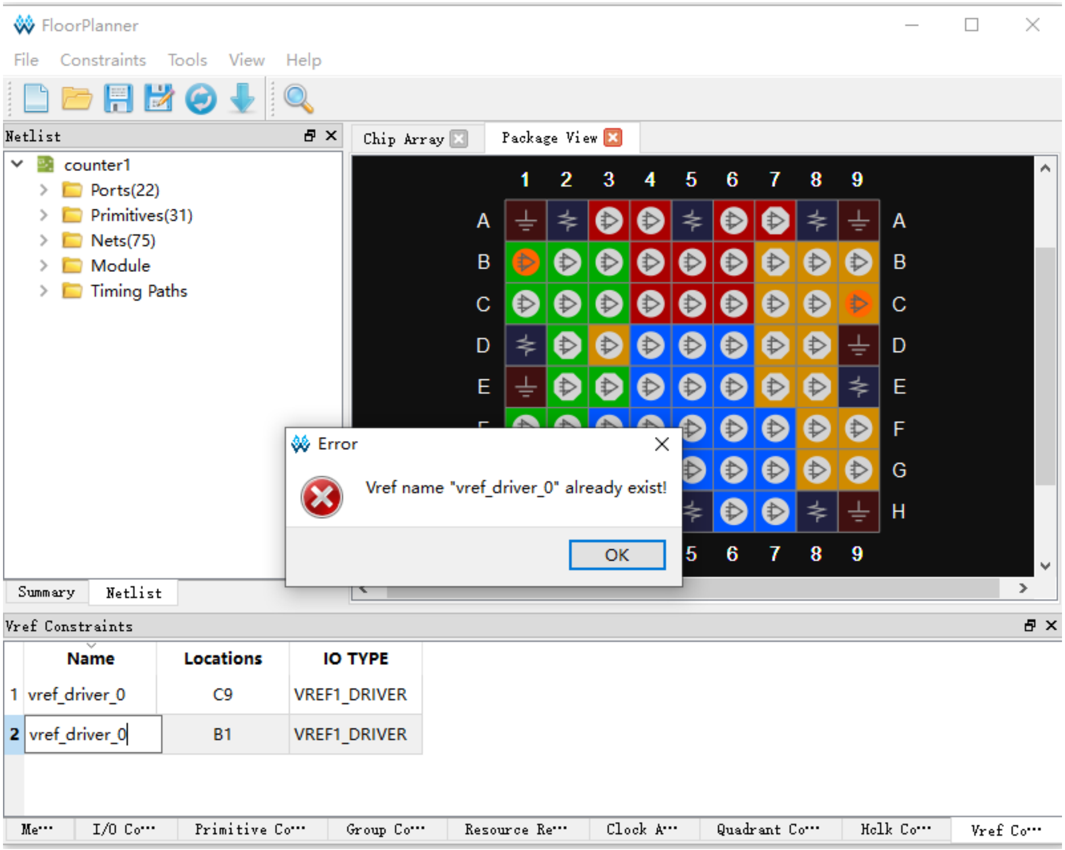
2. Select the created Vref Constraints and drag to the B7 in Package View. The location of the Vref Constraints is displayed as B7, as shown in Figure 4-19.

Figure 4-19 Drag to Package View to Generate Vref Constraints Location



	Name	Locations	IO TYPE
1	vref_driver_0	B7	VREF1_DRIVER

Figure 4-20 Error Prompt



Vref name "vref_driver_0" already exist!

	Name	Locations	IO TYPE
1	vref_driver_0	C9	VREF1_DRIVER
2	vref_driver_0	B1	VREF1_DRIVER

5 Timing Optimization

FloorPlanner supports timing optimization and helps users to realize timing closure by physical location constraints and key path modification, etc.

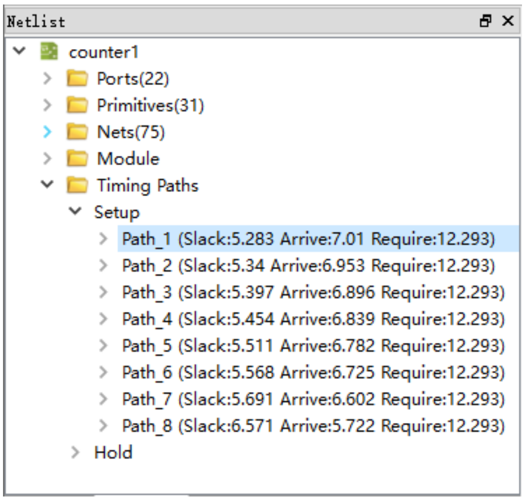
The steps to optimize timing using FloorPlanner are as follows:

1. Create a New Project;
2. Run "Synthesize" to generate netlist file with the .vm suffix after synthesis;
3. Add physical constraints file and timing constraints file. Physical constraints and timing constraints are not a must, but it's suggested to add for better project implementation.
4. Run "Place & Route" to generate posp and timing path files;
5. Read the timing report to check whether the max. frequency meets the requirements or not. If it is not, you need to use FloorPlanner to generate multi-constraints and multi-iterations to realize timing closure;
6. Run "Place & Route" to start FloorPlanner. The timing path is listed in "Netlist > Timing Paths", including slack, arrival time, require time, etc., as shown in Figure 5-1.

Note!

In debugging, it does not need to start FloorPlanner again and again. You can reload by clicking "Reload".

Figure 5-1 Read Timing Path

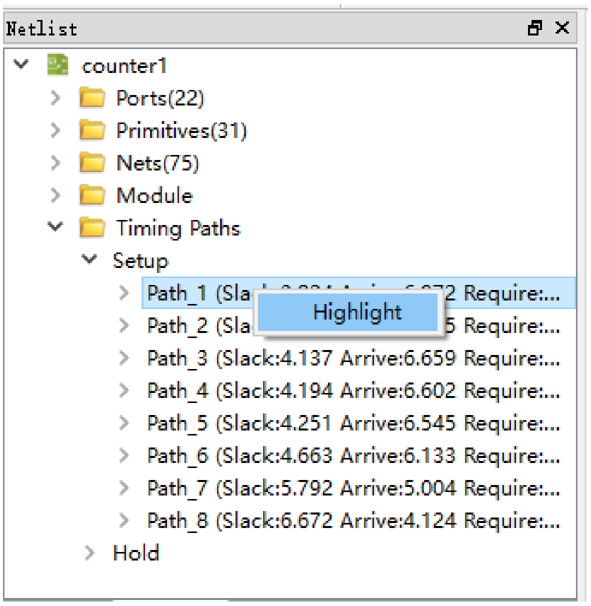


Netlist

- counter1
  - Ports(22)
  - Primitives(31)
  - Nets(75)
  - Module
  - Timing Paths
    - Setup
      - Path_1 (Slack:5.283 Arrive:7.01 Require:12.293)
      - Path_2 (Slack:5.34 Arrive:6.953 Require:12.293)
      - Path_3 (Slack:5.397 Arrive:6.896 Require:12.293)
      - Path_4 (Slack:5.454 Arrive:6.839 Require:12.293)
      - Path_5 (Slack:5.511 Arrive:6.782 Require:12.293)
      - Path_6 (Slack:5.568 Arrive:6.725 Require:12.293)
      - Path_7 (Slack:5.691 Arrive:6.602 Require:12.293)
      - Path_8 (Slack:6.571 Arrive:5.722 Require:12.293)
    - Hold

7. In debugging, you need to find the key path by modifying the design or device locations to realize timing closure. In FloorPlanner, you can adjust locations to realize timing closure. The steps are as follows:
   a). Check the key path signal flow.
   In timing closure debugging, key path signal flow is an important factor. Right-click Place View > All Instance in Chip Array to show the project palce view. Select a key path in the Netlist, right-click and select "Highlight", as shown in Figure 5-2. The path signal flow can be observed in Chip Array, as shown in Figure 5-3.
   b). Adjust improper locations.
   As shown in Figure 5-3, the locations are centralized, and only one locates relatively far. The winding path with large span can influence timing. You can adjust the improper location by dragging to optimize the winding path, as shown in Figure 5-4.
8. Rerun "Place & Route" to view timing result. If the frequency can not meet the requirements, repeat from step 5 to step 7.

Figure 5-2 Highlight Key Path



Netlist

- counter1
  - Ports(22)
  - Primitives(31)
  - Nets(75)
  - Module
  - Timing Paths
    - Setup
      - Path_1 (Sla 2 Require:...
      - Path_2 (Sla 5 Require:...
      - Path_3 (Slack:4.137 Arrive:6.659 Require:...
      - Path_4 (Slack:4.194 Arrive:6.602 Require:...
      - Path_5 (Slack:4.251 Arrive:6.545 Require:...
      - Path_6 (Slack:4.663 Arrive:6.133 Require:...
      - Path_7 (Slack:5.792 Arrive:5.004 Require:...
      - Path_8 (Slack:6.672 Arrive:4.124 Require:...
    - Hold

Figure 5-3 Key Path Signal Flow

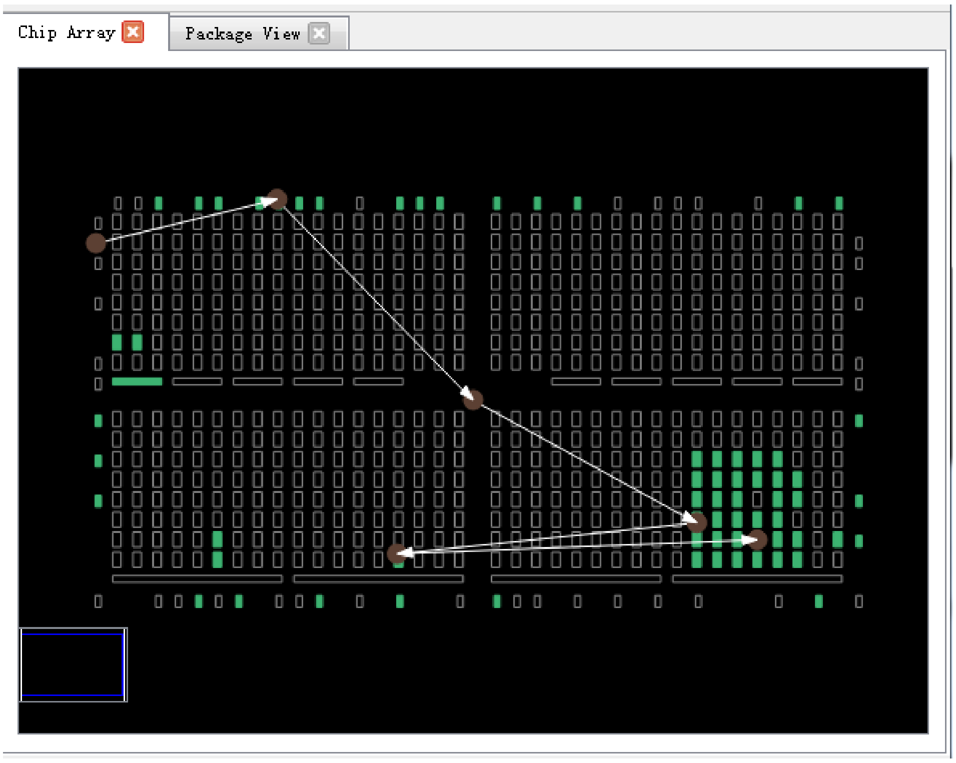
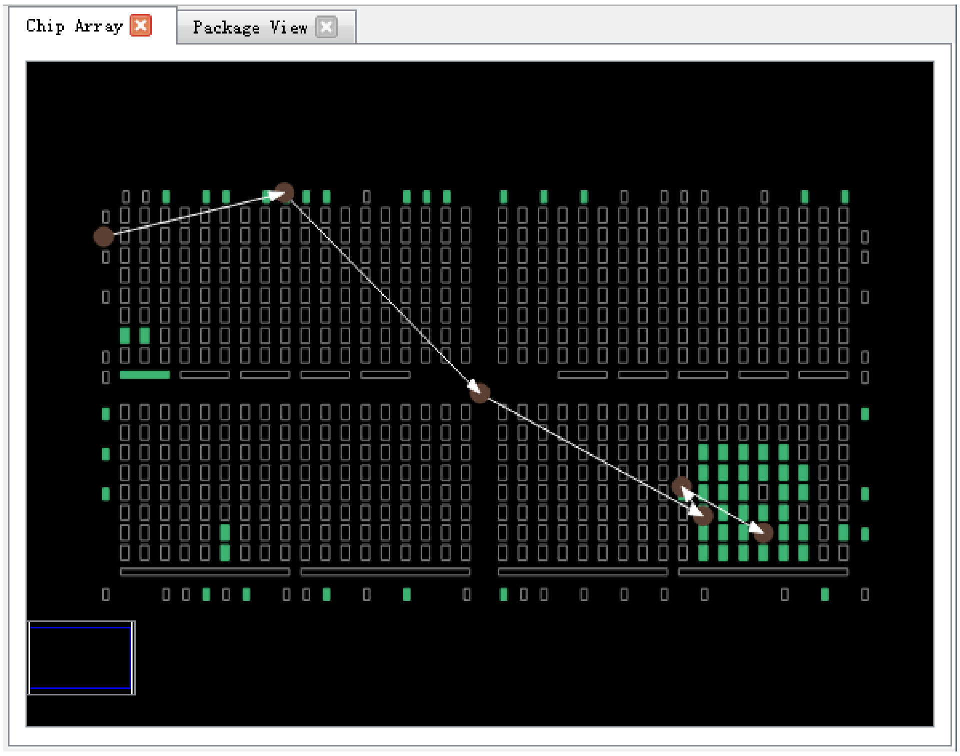


Figure 5-4 Location after Adjustment



Appendix A Physical Constraints Syntax Definition

A.1 I/O Constraints

The IO constraints can constrain port and buffer to the specified IOB location.

Syntax

"IO_LOC" """obj_name""" obj_location ["exclusive"] ";"

Constraints Elements

obj_name

Obj_name can take name of port and I/O buffer as the name.

obj_location

Obj_location is the IOB location, such as "A11", "B12", etc. If multiple locations are specified, they need to be separated by commas, such as "A11, B2".

exclusive

Exclusive is optional, which indicates that the obj_location in the constraints can only place the primitives specified by obj_name after locations constraints.

Note!

If obj_name is the escaped name format (begin with backslash and end with space), the obj_name must be quoted on both sides.

Examples

Example 1

IO_LOC "io_1" A1;

// io_1 should be located to the pin A1.

Example 2

IO_LOC "io_1" A1, B14, A15;

// io_1 should be located in the pin A1, pin B14, or pin A15, one of the three locations will be taken for the placement.

Example 3

IO_LOC "io_2" A1 exclusive;

// io_2 should be located to the pin A1, and pin A1 can only be used by io_2.

Example 4

IO_LOC "io_2" A1, B14, A15 exclusive;

// io_2 should be located to pin A1, B14, or A15, and all these locations can only be used by io_2.

A.2 PORT Attributes Constraints

Port attributes constraints can set attribute values for ports, such as IO_TYPE, PULL_MODE and DRIVE, etc. You can see datasheets for the details.

Syntax

"IO_PORT" """port_name """ attribute "=" attribute_value ";"

Multiple attributes can be set in a constraint statement. Each attribute can be separated by spaces.

Constraints Elements

It needs to constrain the port name, attribute and attribute value.

Examples

Example 1

IO_PORT "port_1" IO_TYPE = LVTTL33;

// Set the IO_TYPE as "LVTTL33".

Example 2

IO_PORT "port_2" IO_TYPE = LVTTL33 SLEW_RATE = FAST PULL_MODE =KEEPER;

// Set the IO_TYPE as the "LVTTL33’, SLEW_RATE value is "FAST", PULL_MODE value is "KEEPER".

Example 3

IO_PORT "port_3" IO_TYPE = LVDS25;

// BUF at port_3 is IBUF, and convert the IBUF to TLVDS_IBUF via the constraints.

A.3 Primitive Constraints

Primitive Constraints are used to place the instances to the specified GRIDs. LUT/BSRAM/SSRAM/DSP/PLL instances can be constrained using Primitive Constraints.

Syntax

"INS_LOC" """ obj_name""" obj_location ["exclusive"]";"

Constraints Elements

obj_name

The instance name

obj_location

obj_location includes:

- A signal location is specified to LUT, such as, RxCy[0-3][A-B];
- A range of the locations are specified to the multiple rows or columns:

Include multiple PLS or LUT: "RxCy", "RxCy[0-3]"

Specify multiple rows: "R[x:y]Cm", "R[x:y]Cm[0-3]", "R[x:y]Cm[0-3][A-B]"

Specify multiple columns: "RxC[m:n]", "RxC[m:n][0-3]", "RxC[m:n][0-3][A-B]"

Specify multiple rows and columns: "R[x:y]C[m:n]", "R[x:y]C[m:n][0-3]", "R[x:y]C[m:n][0-3][A-B]"

Multiple ins_locations can be included in a constraint statement, and they are separated by ",".

- PLL Constraints Location
  For the "PLL_L" or "PLL_R" of the PLL constraints locations, if more than one PLL are placed on the left side, it can be set to "PLL_L[0]", "PLL_L[1]" ...; If more than one PLL are placed on the right side, it can be set to "PLL_R[0]", "PLL_R[1]" ...
- BSRAM Constraints Location
  The BSRAM constraints location is "BSRAM_R10[0]" (the first BSRAM at row 10), "BSRAM_R10[1]"….
- DSP Constraints Location
  The DSP constraints location is "DSP_R19[0]" (the first DSP Block at row 19), "DSP_R19[1]"… If it specifies a macro, it can be marked as: DSP_R19[0][A] or DSP_R19[0][B].

exclusive

Exclusive is optional, which indicates that the obj_location in the constraints can only place the instance specified by obj_name after locations constraints.

Examples

Example 1

INS_LOC "lut_1" R2C3, R5C10[0][A];

// lut_1 is constrained at the R2C3 and the 0th CLS of the R5C10 in the first LUT.

Example 2

INS_LOC "ins_2 " R5C6[2] exclusive;

// ins_2 is constrained at the 2nd CLS of the R5C6, and only the instance can be placed at this location.

Example 3

INS_LOC "ins_3" R[2:6]C1;

// ins_1 is constrained between the row 2 and the row 6 and in the column 1.

Example 4

INS_LOC "ins_4" R[1:4]C[2:6] exclusive;

// ins_3 is constrained between row 1 and row 4, between column 2 and column 6. This location can only be occupied by this instance.

Example 5

INS_LOC "ins_5" R[1:4]C[2:6][1];

// ins_4 is constrained between row 1 and row 4, and in the first CLS of a GRID between column 2 and column 6.

Example 6

INS_LOC "reg_name" B14;

// It is constrained to the IOB B14 by REGISTER/IOLOGIC INS_LOC constraint.

Example 7

INS_LOC "pll_name" PLL_L;

// It is constrained to the PLL left byINS_LOC constraint.

Example 8

INS_LOC "bsram_name" BSRAM_R10[2];

// It is constrained to the third BSRAM in row 10 by INS_LOC constraint.

Example 9

INS_LOC "dsp_name" DSP_R19[2];

// It is constrained to the third DSP in row19 by INS_LOC constraint.

A lut1/lut2/lut3/lut4 can be placed in LUT4. A lut5 needs to occupy two LUT4s (one CLS). A lut6 needs to occupy four LUT4s (two CLS). A lut7 needs to occupy four CLSs (one GRID). A lut8 needs to occupy eight CLSs (two GRIDs). Therefore, for different instance constraints, the minimum unit of the constraint location is also different. For BSRAM/SSRAM/DSP (one DSP includes two MACROs and one MACRO includes two UNITs), the examples are as follows:

Example 11

LUT4 Constraints

INS_LOC "lut4_name" R5C15[1][A];

// lut4_name is constrained to the first LUT in the first CLS of the R5C15.

Example 12

CLS Constraints

INS_LOC "lut5_name" R5C15[3];

// lut5_name is constrained to the third CLS of the R5C15.

Example 13

CLS Constraints

INS_LOC "lut6_name" R5C15[0];

// lut6_name is constrained to the zeroth CLS of the R5C15 (the CLS[0] and CLS[1] will be occupied).

Example 14

GRID Constraints

INS_LOC "lut7_name" R5C15;

BSRAM type: INS_LOC "bsram_name" R10C5; // for GW2A55K

// lut7_name is constrained to the R5C15, and the LUT7 will occupy one GRID。

Example 15

GRID Constraints

INS_LOC "lut8_name" R5C15；

// lut8_name is constrained to the R5C15; lut8_name will occupy R5C15 and the R5C16.

Example 16

DSP MACRO Constraints

INS_LOC "mult_name" DSP_R19[1][A]; // for GW2A55K

// mult_name is constrained to the the first macro of the second DSP in row 19.

A4. Group Constraints

The group constraints includes primitive group constraints and relative group constraints.

A.4.1 Primitive Group Constraints

Primitive Group Constraint is used to define a group constraint. A group is a collection of various instance objects. The instances such as LUT, DFF, etc., or BUF, IOLOGIC, etc. can be added to a group using the Primitive Group constraints. And the location constraints of all objects in the group can be achieved by constraining the location of the group.

Syntax

Definition of GROUP:

"GROUP" group_name "=" "{" """obj_names """ "}" ["exclusive"]";"

Add the instance to the group:

"GROUP" group_name "+=" "{" """obj_names """ "}" ["exclusive"]";"

The location of the group is constrained:

"GRP_LOC" group_name group_location["exclusive"]";"

Note!

If group_name is the escaped name format (begin with backslash and end with space), the quotes at two sides of group_name are necessary.

Constraints Elements

group_name

Define a name as the name of the group.

obj_name

Obj_name is used to add the specified instance to the group.

group_location

Specify the constraints location of the group, and the group_location can at the IOB and the GRID.

exclusive

The "exclusive" is optional, which is at the end of the group definition or the location constraints;

An object can be included in multiple groups, but the object can only be included in the group that the "exclusive" is added;

The "exclusive" indicates that the constraints location can only be occupied by the objects in the group.

Examples

Example 1

GROUP group_1 = { "ins_1" "ins_2" "ins_3" "ins_4" };

// Create a group named group_1 and add the ins_1, ins_2, ins_3, ins_4 to the group.

Example 2

GROUP group_2 = { "ins_5" "ins_6" "ins_7" } exclusive;

// Create a group named group_2 and the ins_5, ins_6, ins_7, ins_4 only can be added to this group.

Example 3

GROUP group_1 += { "io_1" "io_2"};

// Add io_1, io_2 to group_1.

Example 4

GRP_LOC group_1 R3C4, A14, B4;

// The objects in group_1 can be placed at R3C4, A14, B4.

Example 5

GRP_LOC group_2 R[1:3]C[1:4] exclusive;

// The Instance in group_2 can be placed in the range of R[1:3]C[1:4], and the instances in group_2 can only be placed in the range.

A.4.2 Relative Group Constraints

The instance relative location constraints on can be realized using the Relative Group Constraints.

Syntax

Define relative group constraints:

"REL_GROUP" group_name "=" "{" """obj_names """ "}"";"

Add the instance to the defined group:

"REL_GROUP" group_name "+=" "{" """obj_names """ "}"";"

The instance relative location is constrained in the group:

"INS_RLOC" """ obj_name""" relative_location ";"

Constraints Elements

obj_name

The name of the constraint object.

relative_location

The description of the relative locations in row and column.

Examples

Example 1

```
REL_GROUP grp_1 = { "ins_1" "ins_2" "ins_3" "ins_4" };
INS_RLOC "ins_1" R0C0;
INS_RLOC "ins_2" R2C3;
INS_RLOC "ins_3" R3C5;
```

// Define a group constraint named grp_1 and add the ins_1, ins_2, ins_3, ins_4 to grp_1. The ins_1 is the relative location origin R0C0, the ins_2 is constrained to the R2C3 relative to the ins_1, and the ins_3 is constrained to the R3C5 relative to ins_1.

A.5 Resource Reservation

The specified location or range can be reserved using the Resource Reservation constraints.

Syntax

```
"LOC_RESERVE" location [ res_obj ] ";"
```

Examples

Example 1

```
LOC_RESERVE R2C3[0][A] -LUT;
LOC_RESERVE R2C3[0][A] -REG;
```

Example 2

```
LOC_RESERVE IOR3, IOR6, R2C3, R3C4;
```

Example 3

```
LOC_RESERVE R[2:5]C[3:6], R3C[8:9];
```

// The locations constraints in the above examples will be reserved in place.

A.6 Vref Constraints

The chip supports the external reference voltage. Each PAD of the chip (include IOLOGIC) can be used as an input PAD for the external reference voltage, which is valid for the BANK. The Vref Constraints can be used to constrain the name and location of the input pin of the external reference voltage.

Syntax

"USE_VREF_DRIVER" vref_name [location]";"

Constraints Elements

vref_name

Customized VREF pin name

location

Any PAD (include IOLOGIC) location in the chip can be used as a location for the VREF pin constraints.

Examples

Example 1

USE_VREF_DRIVER vref_pin;

IO_PORT "port_1" IO_TYPE = SSTL25 VREF=vref_pin;

IO_PORT "port_2" IO_TYPE = SSTL25 VREF=vref_pin;

// Define a VREF pin named "vref_pin" and set the port_1 and port_2 to vref_pin.

Example 2

USE_VREF_DRIVER vref_pin C7;

IO_PORT "port_1" IO_TYPE = SSTL25 VREF=vref_pin;

IO_PORT "port_2" IO_TYPE = SSTL25 VREF=vref_pin;

// Define a VREF pin named "vref_pin" and constrain it to PAD C7 (bank 3, GW1N-4, WLCSP72). Set the VREF value of port_1 and port_1 as vref_pin, and port_1 and port_1 will be placed in the bank where C7 locates.

A.7 Quadrant Constraints

The Quadrant is used to constrain objects such as DCS/DQCE to a specified quadrant (GW1N family has LEFT and RIGHT quadrants, and the GW1N-9/GW1NR-9/GW1N-9C/GW1NR-9C and GW2A family have four quadrants: TOPLEFT, TOPRIGHT, BOTTOMLEFT, BOTTOMRIGHT. See the relevant datasheets for the details).

Syntax

"INS_LOC" """obj_name""" quadrant ";"

Constraints Elements

obj_name

The name of the constraint object.

quadrant

GW1N series: "LEFT" ("L"), "RIGHT" ("R")

GW1N-9/GW1NR-9/GW1N-9C/GW1NR-9C and GW2A series: "TOPLEFT"("TL"), "TOPRIGHT"("TR"), "BOTTOMLEFT"("BL"), "BOTTOMRIGHT"("BR").

Note!

Abbreviations are in parentheses.

Examples

Example 1

INS_LOC "dcs_name" LEFT;

// Constrain the DCS dcs_name to the LEFT quadrant (GW1N family).

A.8 Clock Assignment

The clock assignment constrain a specific net to the global clock line or not route clock line in the design. There are eight master clocks and eight long nets in each quadrant of the chip resources. It can constrain the global clock line for specific fanout (CLK/CE/SR/LOGIC) of the net.

BUFG[0-7] represents the eight master clocks.

BUFS represents eight long lines.

LOCAL_CLOCK means this net is not to route clock line.

The CLK signal is the signal connected to the CLK pin. The CE signal is the signal connected to the CE pin. The SR signal is the signal connected to the SET/RESET/CLEAR/PRESET pins, and the LOGIC is the signal connected to other logics pins.

Syntax

"CLOCK_LOC" """net_name """ global_clocks "=" fanout [quadrant]";"

Note!

The "NET_LOC" is updated to "CLOCK_LOC", but "NET_LOC" used to constrain BUFG and BUFS still supports.

Constraints Elements

net_name

The net name

global_clocks

BUFG[0-7] represents the eight master clocks.

BUFS represents the eight long lines.

LOCAL_CLOCK means this net is not to route clock line.

fanout

CLK: Fanout is the net of the CLK.

CE: Fanout is the net of the CE.

SR: Fanout is the net of SET/RESET (synchronous reset signal), CLEAR/PRESET (asynchronous reset signal).

LOGIC: Fanout is the net other than the fanout above.

The sign "|" can be used to separate multiple specified fanout.

Note!

If global clocks select LOCAL CLOCK, fanout can not be selected.

Examples

Example 1

CLOCK_LOC "net" BUFG[0] = CLK;

// Constrain the net fanout as the CLK net routing to the 0th master clock.

Example 2

CLOCK_LOC "net" BUFG = CLK|CE;

NET_LOC "net" BUFG = CLK|CE;

// Constrain the net fanout as the CLK/CE net routing to the master clock.

Example 3

CLOCK_LOC "net" BUFS = CE;

NET_LOC "net" BUFS = CE;

// Constrain the net fanout as the CE net routing to the long line.

Example 4

CLOCK_LOC "net" LOCAL_CLOCK;

// Constrain the net not to route the clock line.

A.9 Hclk Constraints

The CLKDIV/DLLDLY can be constrained to the relevant locations via the CLKDIV/DLLDLY constraints. The constraints locations of CLKDIV/DLLDLY are different from the ones of other general instances. The "TOPSIDE", "BOTTOMSIDE", "LEFTSIDE", and "RIGHTSIDE" indicates the four sides of the constraints location.

Syntax

"INS_LOC" """ obj_name """ location";"

Constraints Elements

obj_name

The instance name of the CLKDIV/DLLDLY is the obj_name.

location

"TOPSIDE[0-1]" ("TS[0-1]")

"BOTTOMSIDE[0-1]" ("BS[0-1]")

"LEFTSIDE[0-1]" ("LS[0-1]")

"RIGHTSIDE[0-1]" ("RS [0-1]")

Note! Abbreviations are in parentheses.

Examples

Example 1

INS_LOC "clkdiv_name" TS[0];

// Place the clkdiv_name to TOPSIDE[0].