Gowin Design Physical Constraints

User Guide

SUG935-1.4.2E, 02/02/2024

Disclaimer

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. 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.
06/10/2021	1.2E	The description of the combined use of Port Constraints and Vref Constraints added.
11/02/2021	1.3E	● Some descriptions updated. ● Document structure adjusted.
10/28/2022	1.3.1E	GW1NS-2 removed.
12/16/2022	1.3.2E	● A.10 Other Constraints added. ● VCC modified to VCCIO
03/31/2023	1.3.3E	● Slew Rate setting shielded. ● Find function removed.
05/25/2023	1.3.4E	● BUFS description updated. ● Clock Assignment updated to Clock Net Constraints. ● Quadrant Constraints updated to GCLK Primitive Constraints. ● Hclk Constraints updated to HCLK Primitive Constraints.
06/30/2023	1.3.5E	The descriptions of Primitive Group Constraints/Vref Constraints/Clock Net Constraints in Appendix A Physical Constraints Syntax Definition updated.
08/18/2023	1.4E	Timing Paths function removed.
11/30/2023	1.4.1E	● Some screenshots in Chapter 3 FloorPlanner GUI updated. ● The title description for the dialog box that pops up when "Instance" selected in "HCLK/GCLK Primitive Constraints" updated.
02/02/2024	1.4.2E	Figure 3-23 Back-annotate Port updated.

Contents

Contents i
List of Figures iii
List of Tables vi
1 About This Guide 1
1.1 Purpose 1
1.2 Related Documents 1
1.3 Terminology and Abbreviations 1
1.4 Support and Feedback 2
2 Introduction 3
3 FloorPlanner GUI 4
3.1 Start FloorPlanner 4
3.2 FloorPlanner Interface 6
3.2.1 Menu Bar 6
3.2.2 Summary and Netlist Windows 17
3.2.3 Package View 20
3.2.4 Chip Array Window 25
3.2.5 Constraints Editing Window 30
3.2.6 Message Window 35
4 FloorPlanner Usage 36
4.1 Create Constraints File 36
4.2 Edit Constraints File 38
4.2.1 Constraints Examples 38
4.2.2 Edit I/O Constraints 40
4.2.3 Edit Primitive Constraints 41
4.2.4 Edit Group Constraints 42
4.2.5 Edit Resource Reservation Constraints 46
4.2.6 Edit Clock Net Constraints 47
4.2.7 Edit GCLK Primitive Constraints 48
4.2.8 Edit HCLK Primitive Constraints 49

4.2.9 Edit Vref Constraints 49

Appendix A Physical Constraints Syntax Definition 52

A.1 I/O Location Constraints 52

A.2 I/O Attributes Constraints 53

A.3 Primitive Constraints 54

A.4 Group Constraints 57

A.4.1 Primitive Group Constraints 58

A.4.2 Relative Group Constraints 59

A.5 Resource Reservation Constraints 60

A.6 Vref Constraints 61

A.7 GCLK Primitive Constraints 62

A.8 Clock Net Constraints 62

A.9 HCLK Primitive Constraints 64

A.10 Other Constraints 65

A.10.1 JTAGSEL_N Net Constraints 65

A.10.2 RECONFIG_N Net Constraints 65

List of Figures

Figure 3-1 Start FloorPlanner via Menu Bar 4
Figure 3-2 Start FloorPlanner in Process View 5
Figure 3-3 Start FloorPlanner via Start Page 5
Figure 3-4 FloorPlanner Interface 6
Figure 3-5 File 7
Figure 3-6 Open Physical Constraints 7
Figure 3-7 Constraints 8
Figure 3-8 Primitive Finder 8
Figure 3-9 New Primitive Group 9
Figure 3-10 Right Primitive Group 10
Figure 3-11 Invalid Locations 10
Figure 3-12 Invalid Locations 10
Figure 3-13 New Relative Group 11
Figure 3-14 Right Relative Group 11
Figure 3-15 Resource Reservation 12
Figure 3-16 Clock Net Constraints 13
Figure 3-17 GCLK Primitive Constraints (GW1N-1) 13
Figure 3-18 GCLK Primitive Constraints (GW2A-18) 14
Figure 3-19 HCLK Primitive Constraints 14
Figure 3-20 Vref Constraints 15
Figure 3-21 Tools 15
Figure 3-22 Back-annotate Physical Constraints 16
Figure 3-23 Back-annotate Port 16
Figure 3-24 View 16
Figure 3-25 Windows 17
Figure 3-26 Summary Window 17
Figure 3-27 Netlist Window 18
Figure 3-28 BUS and Non-Bus Display 19
Figure 3-29 Hierarchy Display 19
Figure 3-30 Netlist Right-clicking 20

Figure 3-31 Package View (GW1NRF-4B-QFN48) 21
Figure 3-32 Package View Right-clicking 22
Figure 3-33 Differential Pair Display 22
Figure 3-34 Top View 23
Figure 3-35 Bottom View 23
Figure 3-36 GW1N-9-WLCSP81M Top View 24
Figure 3-37 GW1N-9-WLCSP81M Bottom View 24
Figure 3-38 Chip Array Window 25
Figure 3-39 Constraints in Grid 26
Figure 3-40 Constraints in Macrocell 26
Figure 3-41 Constraints in Primitive 27
Figure 3-42 Chip Array Right-clicking 29
Figure 3-43 Show Place View 29
Figure 3-44 Mouse Hovering Display 30
Figure 3-45 Right-click to Select Hightlight 30
Figure 3-46 I/O Constraints View 32
Figure 3-47 Primitive Constraints View 32
Figure 3-48 Group Constraints View 33
Figure 3-49 Resource Reservation View 33
Figure 3-50 Clock Net Constraints View 34
Figure 3-51 GCLK Primitive Constraints View 34
Figure 3-52 HCLK Primitive Constraints 35
Figure 3-53 Vref Constraints View 35
Figure 3-54 Message View 35
Figure 4-1 New Physical Constraints 36
Figure 4-2 Select Device 37
Figure 4-3 Save Output File 38
Figure 4-4 Drag to Chip Array to Create I/O Constraints 40
Figure 4-5 Drag to Package View to Create I/O Constraints 41
Figure 4-6 Drag to Chip Array to Create Primitive Constraints 42
Figure 4-7 Group Constraints Right-clicking 42
Figure 4-8 Create Primitive Group Constraints 43
Figure 4-9 Primitive Group Constraints 44
Figure 4-10 Create Relative Group Constraints 45
Figure 4-11 Relative Group Constraints 46
Figure 4-12 Create Resource Reservation 46
Figure 4-13 Resource Reservation 47
Figure 4-14 Create Clock Net Constraints 47

Figure 4-15 Clock Net Constraints 48
Figure 4-16 Create GCLK Primitive Constraints 48
Figure 4-17 GCLK Primitive Constraints 48
Figure 4-18 Create HCLK Primitive Constraints 49
Figure 4-19 HCLK Primitive Constraints 49
Figure 4-20 Create Vref Constraints 50
Figure 4-21 Prompt 50
Figure 4-22 Drag to Chip Array to Generate Vref Constraints Location 50
Figure 4-23 Drag to Package View to Generate Vref Constraints Location 51

List of Tables

Table 1-1 Terminology and Abbreviations 1

1 About This Guide

1.1 Purpose

This manual describes Gowin FloorPlanner. It introduces the GUI and syntax of FloorPlanner in order to help you add physical constraints to your design. 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 documents:

- SUG100, Gowin Software User Guide
- UG290, Gowin FPGA Products Programming and Configuration User Guide
- DS102, GW2A series of FPGA Products Data Sheet

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
BSRAM	Block SRAM
CFU	Configurable Function Unit
CLKDIV	Clock Divider

Terminology and Abbreviations	Meaning
CLS	Configurable Logic Section
DCS	Dynamic Clock Selector
DLLDLY	DLL Delay
DQS	Bidirectional Data Strobe Circuit for DDR Memory
FloorPlanner	Physical Constraints Editor
FPGA	Field Programmable Gate Array
GCLK	Global Clock
I/O	Input/Output
IDE	Integrated Development Environment
LUT	Look-up Table
PCLK	Primary Clock
PLL	Phase-locked Loop
SCLK	Segmented Clock
SSRAM	Shadow SRAM
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

FloorPlanner is a physical constraints editor and designed in-house by Gowin. It supports reading and editing the attributes and locations of I/O, Primitive, and Group, etc. It also supports the generation of place constraints files according to your configuration. These files define I/O attributes, as well as locations of primitives and modules. FloorPlanner provides easy and fast placement and constraint editing functions to improve the efficiency of writing physical constraint files.

The functions of FloorPlanner are as follows.

- Supports the input of user design files & constraints files, editing constraints files, and the output of constraints files
- Supports the display of I/O Port, Primitive and Group constraints in user design files
- Can create, edit and modify constraints files
- Supports grid mode, macro cell mode and primitive mode of chip array
- Supports Package View
- Can display Chip Array and Package View synchronously
- Supports real-time display and differences display of constraints locations
- Can generate locations by dragging
- Supports I/O port configuration and batch configuration
- Supports display and editing function of Clock Net Constraints
- Supports constraints legality check
- Supports Back-annotate Physical Constraints

3 FloorPlanner GUI

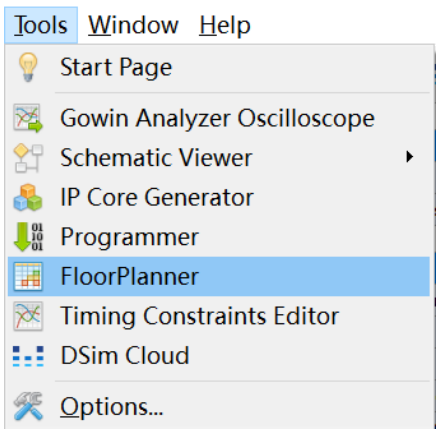
FloorPlanner can create and edit physical constraint files. It supports tabulated constraints editing and efficient netlist lookup so as to improve the efficiency of writing physical constraints files.

3.1 Start FloorPlanner

There are three methods to start FloorPlanner:

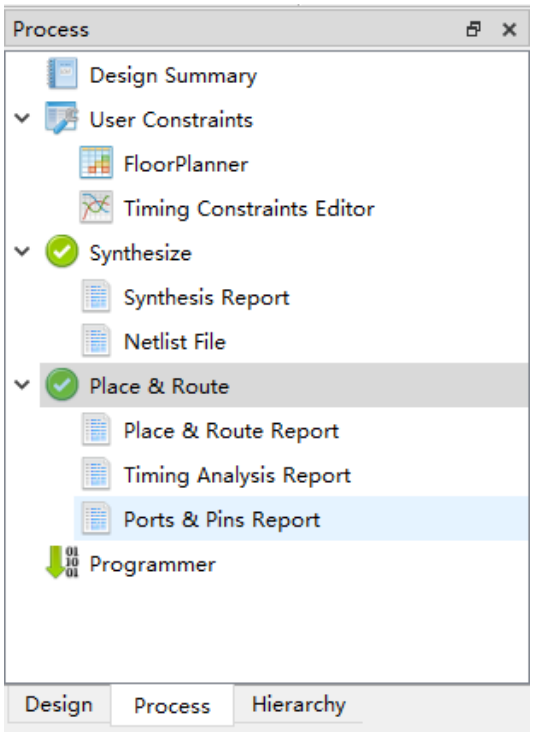
1. Click "IDE > Tools" to open "FloorPlanner", as shown in Figure 3-1.

Figure 3-1 Start FloorPlanner via Menu Bar



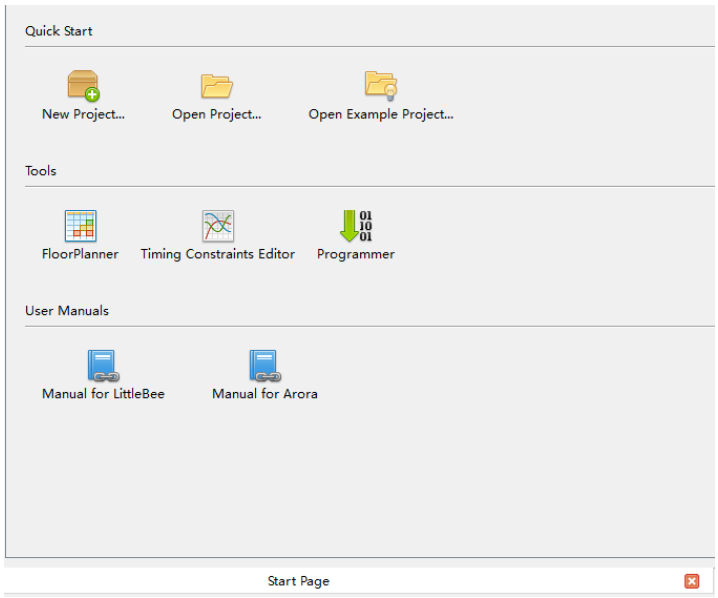
2. After synthesis, Double-click "FloorPlanner" in Process view, as shown in Figure 3-2.

Figure 3-2 Start FloorPlanner in Process View



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

Figure 3-3 Start FloorPlanner via Start Page



Note!

- If you use Gowin FloorPlanner for constraints, the netlist file should be added first.
- When you choose the first or the second method to start Gowin FloorPlanner, the netlist file will be loaded automatically.

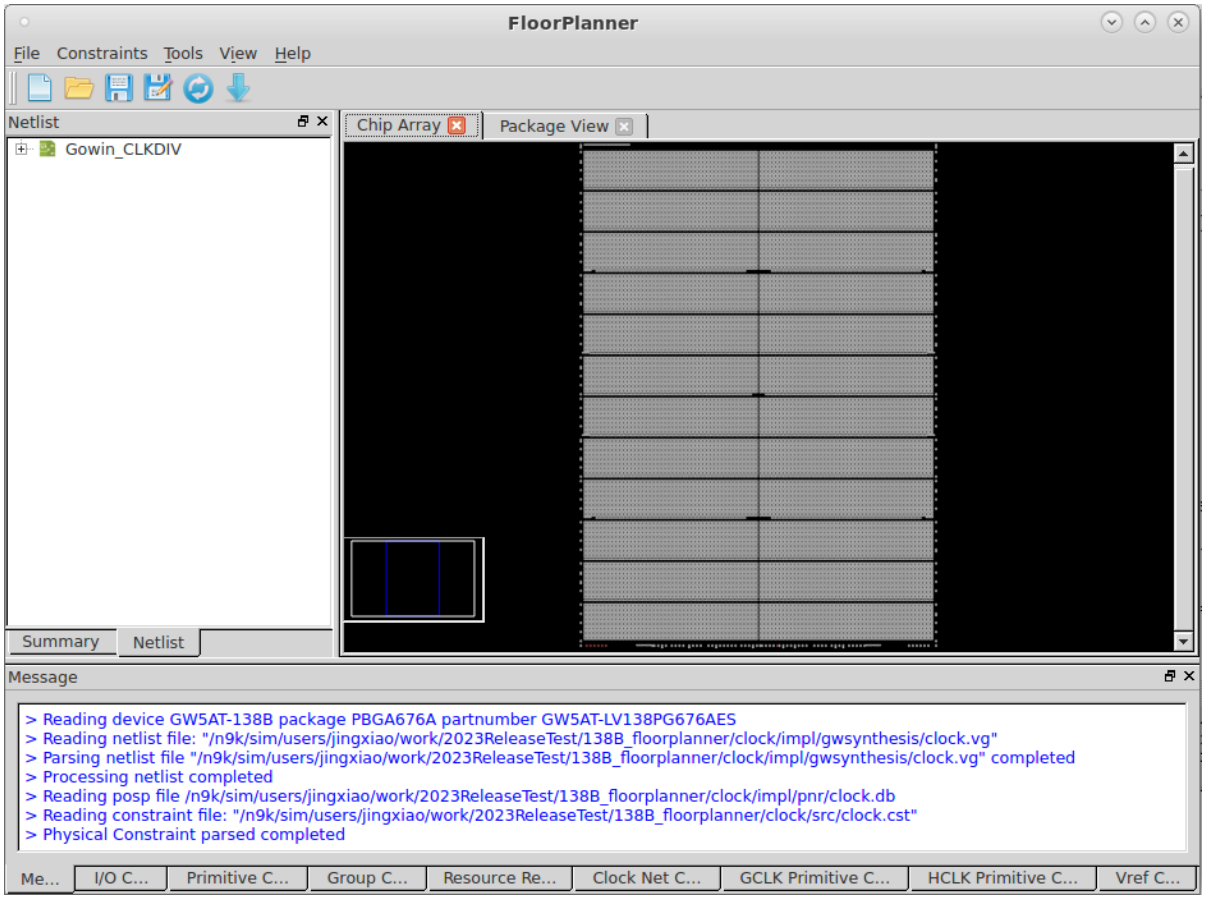
- When you choose the third method to start Gowin FloorPlanner, the netlist file is needed to be loaded via "File > New".

3.2 FloorPlanner Interface

Create or open FloorPlanner interface (including the netlist file), as shown in Figure 3-4.

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

Figure 3-4 FloorPlanner Interface



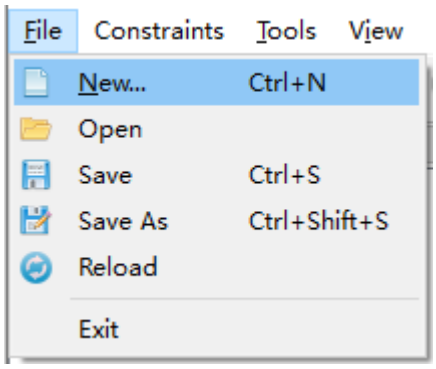
3.2.1 Menu Bar

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

File Menu

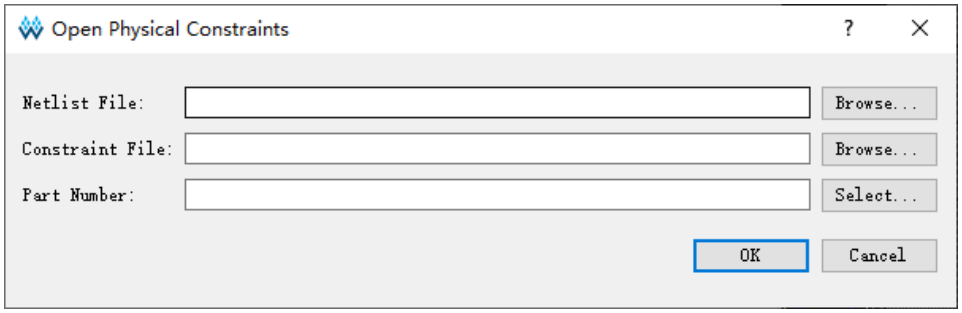
File view is shown in Figure 3-5.

Figure 3-5 File



- New: Create constraints, add user design and select device, etc.
- Open: Open to add constraints, select part number, as shown in Figure 3-6.
- Reload: Physical constraints files and place 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.
- Exit: Exit FloorPlanner.

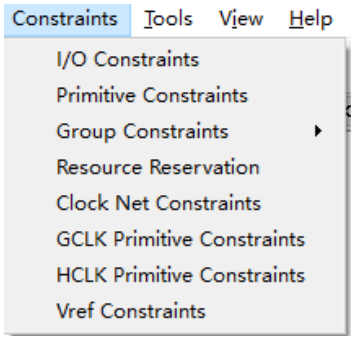
Figure 3-6 Open Physical Constraints



Constraints Menu Bar

Constraints menu bar is as shown in Figure 3-7.

Figure 3-7 Constraints



Primitive Constraints

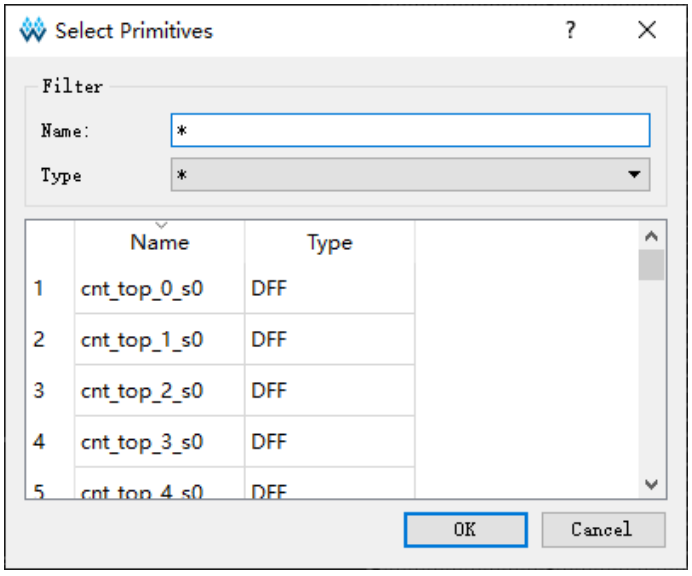
Right-click to select "Select Primitives" and a dialog box pops up, as shown in Figure 3-8.

1. You can select primitives by name or type.
2. Click "OK" to generate the constraints and the constraints are displayed in "Primitive Constraints" at the bottom of main interface.
3. You can set the location by typing or dragging in editing view.

Note!

The location is highlighted in light blue in Chip Array.

Figure 3-8 Primitive Finder



Group Constraints

Group constraints include New Primitive Group and New Relative Group.

Create Primitive Group.

1. Create primitive group. Right-click to select "New Primitive Group" and a dialog box pops up, as shown in Figure 3-9.
2. You can set Group name, Primitives locations and Exclusive. you can add and remove Primitives by clicking "  " and "  " buttons to create a right Primitive Group, as shown in Figure 3-10.

Note!

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

3. After finishing configuration, click "OK", and the syntax of the locations will be checked by the tool.
   - If the location is invalid, a prompt dialog box as Figure 3-11 and Figure 3-12 will pop up. You need to change the location.
   - If there is no error, click "OK", and the available location will be displayed in Chip Array.
4. You can see the created group constraints in "Group Constraints". Double-click the group constraint, and Figure 3-10 pops up; you can edit the constraints.

Figure 3-9 New Primitive Group

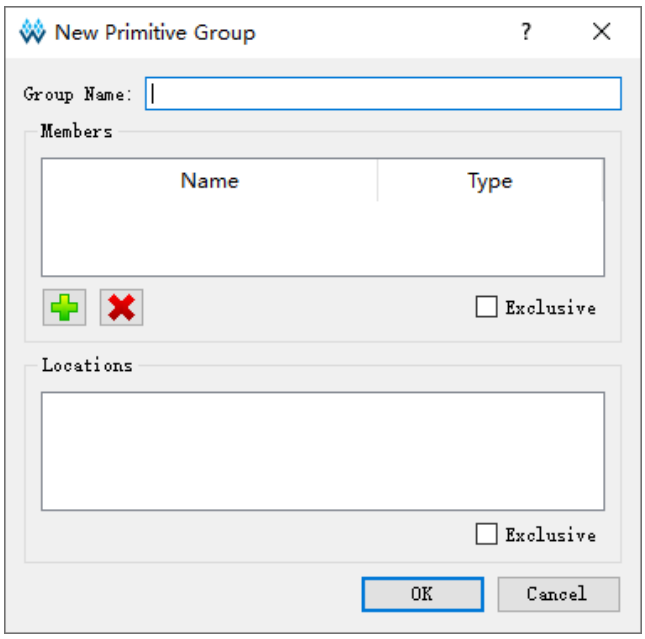


Figure 3-10 Right Primitive Group

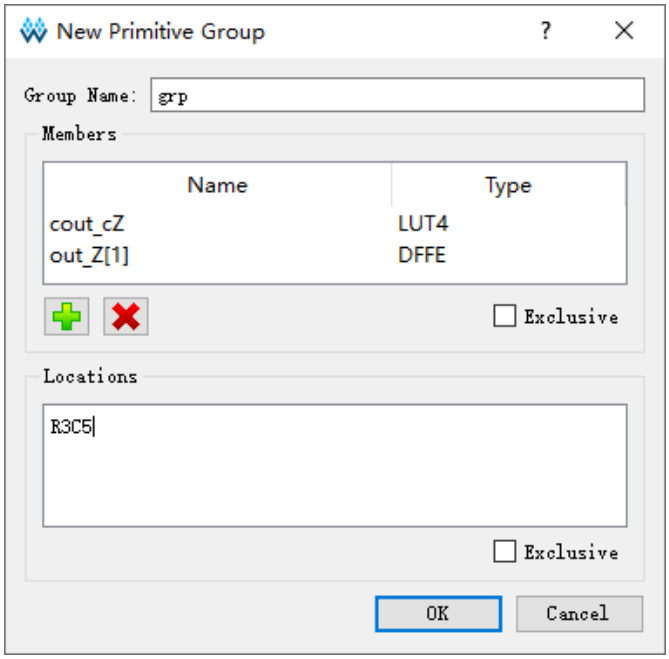


Figure 3-11 Invalid Locations

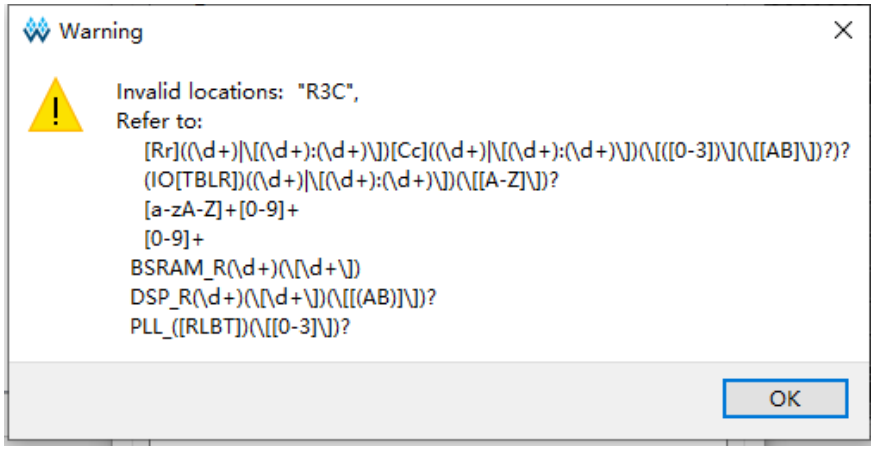
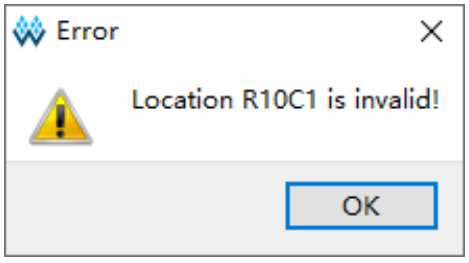


Figure 3-12 Invalid Locations



Create Relative Group.

1. Create Relative Group constraints. Right-click to select "New Relative

Group" and a dialog box pops up, as shown in Figure 3-13.

2. You can set the group name, group members, and their relative locations. You can add and remove primitives by "  " and "  " buttons. The created relative group constraints are shown in Figure 3-14.

Note!

- Group name, Primitive, and Relative Location are required.
- The locations can be inputted 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 window.

3. Click "OK" to generate the constraints.
4. See the created constraints in "Group Constraints". Double-click the constraints, and a dialog box pops up, as shown in Figure 3-14; you can edit the constraints.

Figure 3-13 New Relative Group

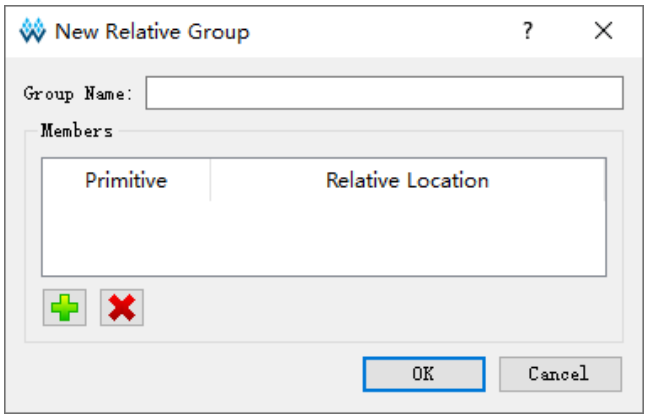
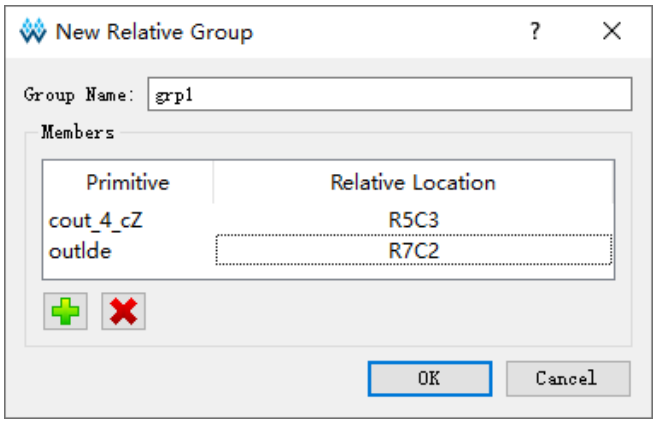


Figure 3-14 Right Relative Group



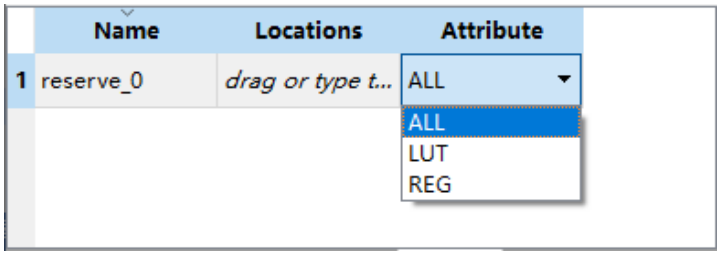
Resource Reservation

1. Create Resource Reservation constraints. Click Reserve Resources to create a new constraint in "Resource Reservation" window at the bottom of the interface.
2. You can type the locations or by dragging.
3. Double-click "Attribute" or click the "Attribute" column drop-down box to set the attribute of the reserved locations, as shown in Figure 3-15.

Note!

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

Figure 3-15 Resource Reservation



Clock Net Constraints

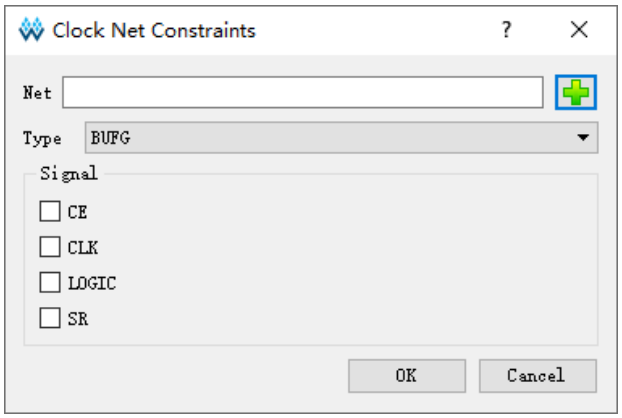
Create Clock Net Constraints and the number of constraints is limited; the constraints validity will be checked. Right-click to select "Clock Net Constraints" and a dialog box pops up, as shown in Figure 3-16. You can perform the following operations.

1. Click "  " to select the corresponding Net.
2. Select "BUFG", "BUFG[0]~[7]", "BUFS" and "LOCAL_CLOCK" from "Type" drop-down list.
3. Configure Signal type through "CE" and "CLK" check box. After the configuration is complete, click "OK" to generate constraints, which are displayed in "CLOCK Net Constraints" window. Double-click to open the dialog box for editing.

Note!

When LOCAL_CLOCK is selected in "Type", the signal check box is grayed.

Figure 3-16 Clock Net Constraints



GCLK Primitive Constraints

Create global clock constraints for DCS and DQCE. Constrain the specified instance to the specific GCLK according to GCLK distribution. Right-click to select "Select GCLK Primitive" in "GCLK Primitive Constraints" window and a dialog box pops up, as shown in Figure 3-17. You can perform the following operations

1. Select a corresponding GCLK primitive by clicking "  ". If there are no GCLK primitives in the design, you can not add.
2. Configure global clock positions through "Position" check box.
3. Click "OK" to generate constraints, which are displayed in the "GCLK Primitive Constraints" window; you can double-click to open the dialog box for editing.

Figure 3-17 GCLK Primitive Constraints (GW1N-1)

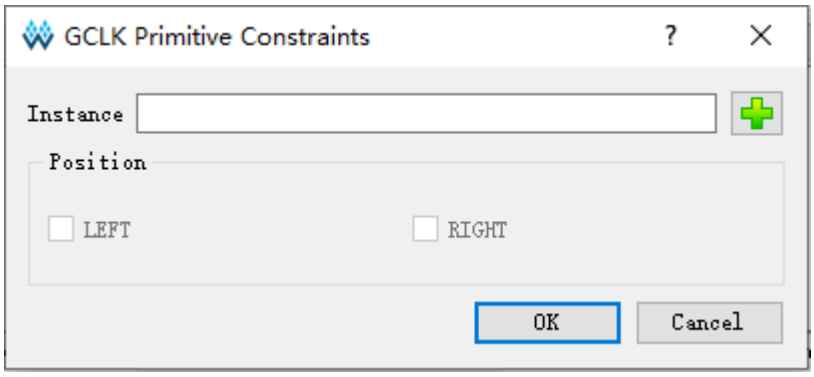
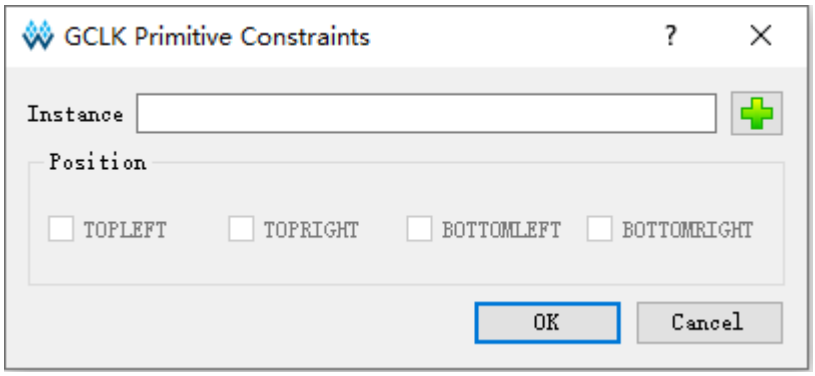


Figure 3-18 GCLK Primitive Constraints (GW2A-18)



Note!

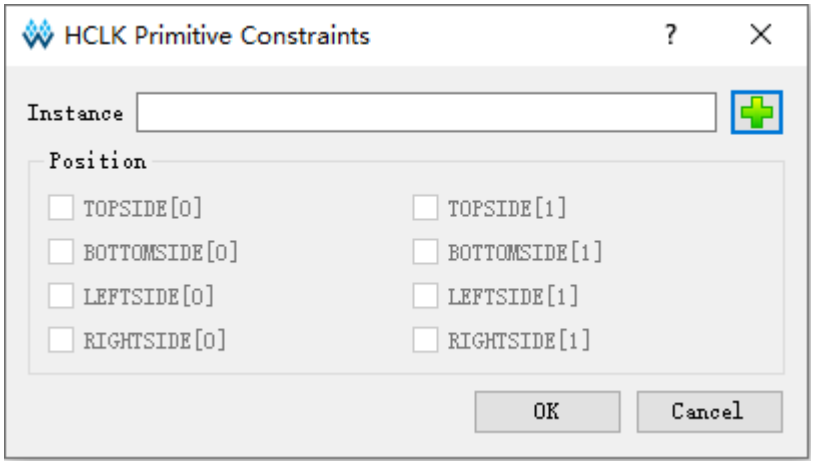
- When "Instance" is selected, "Position" is highlighted.
- The available positions vary depending on the device in the project, and the available positions for different global clock primitives also vary.

HCLK Primitive Constraints

Create HCLK primitive constraints and specify the constraint to HCLK locations. Right-click to select "Select HCLK Primitive" in "HCLK Primitive Constraints" window and a dialog box pops up, as shown in Figure 3-19. The related operations are as follows.

1. You can select a corresponding HCLK primitive by clicking "". If there are no HCLK primitives in the design, you can not add.
2. Configure HCLK positions through "Position" check box.
3. Click "OK" to generate constraints, which are displayed in "HCLK Primitive Constraints" window, you can double-click to open the dialog box for editing.

Figure 3-19 HCLK Primitive Constraints



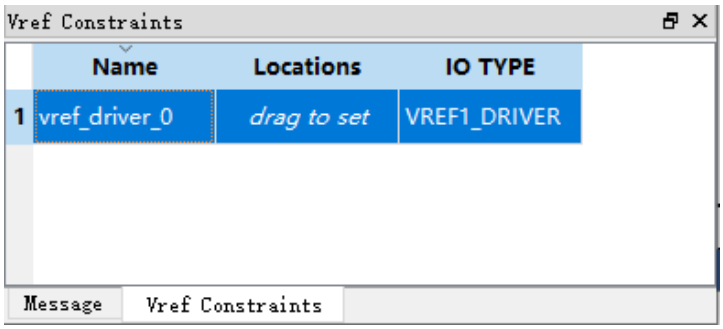
Note!

- When "Instance" is selected, "Position" is highlighted.
- The available positions are different depending on the device in the project, and the unavailable positions are greyed.

Vref Constraints

Create Vref Driver to configure IO Port Vref; right-click and select "Define Vref Driver" to create a new constraint in the "Vref Constraints" window, as shown in Figure 3-20.

Figure 3-20 Vref Constraints



Note!

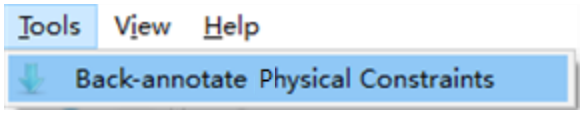
- Specify the location of Vref by dragging.
- Modify Vref name by double-clicking.

Tools Menu

Tools view is shown in Figure 3-21.

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

Figure 3-21 Tools



1. Click "Tools > Back-annotate Physical Constraints" to open a dialog box, as shown in Figure 3-22. Back-Annotate Physical Constraints is effective only when FloorPlanner is started in the project after Place & Route runs successfully.
2. You can select one or more objects in the Back-annotate Physical Constraints dialog box. Click "OK" to open the "Save as" dialog box and print the place information to the physical constraint file.
3. As shown in Figure 3-23, it is the generated physical constraints file when Port and Port Atrribute selected in Back-annotate Physical

Constraints.

Figure 3-22 Back-annotate Physical Constraints

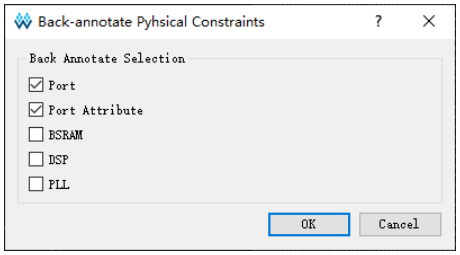


Figure 3-23 Back-annotate Port

```
1
2  IO_LOC "in1" N13;
3  IO_PORT "in1" IO_TYPE=LVCMOS33 PULL_MODE=NONE BANK_VCCIO=3.3;
4  IO_LOC "in2" G18;
5  IO_PORT "in2" IO_TYPE=LVCMOS33 PULL_MODE=NONE BANK_VCCIO=3.3;
6  IO_LOC "in3" H14;
7  IO_PORT "in3" IO_TYPE=LVCMOS33 PULL_MODE=NONE BANK_VCCIO=3.3;
8  IO_LOC "in4" J16;
9  IO_PORT "in4" IO_TYPE=LVCMOS33 PULL_MODE=NONE BANK_VCCIO=3.3;
10 IO_LOC "in5" J15;
11 IO_PORT "in5" IO_TYPE=LVCMOS33 PULL_MODE=NONE BANK_VCCIO=3.3;
```

View Menu

As shown in Figure 3-24, 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 3-25
- Zoom In: Zoom in Chip Array or Package View
- Zoom Out: Zoom out Chip Array or Package View
- Zoom Fit: Zoom in/out Chip Array or Package View according to the window size.

Figure 3-24 View

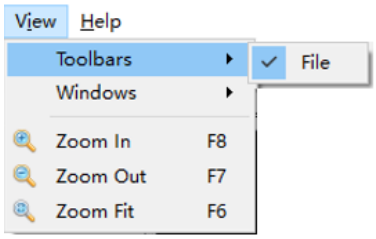
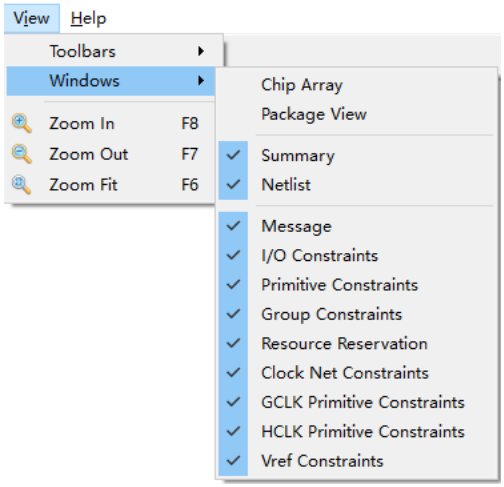


Figure 3-25 Windows



Help Menu

Help is used to provide software version and copyright information.

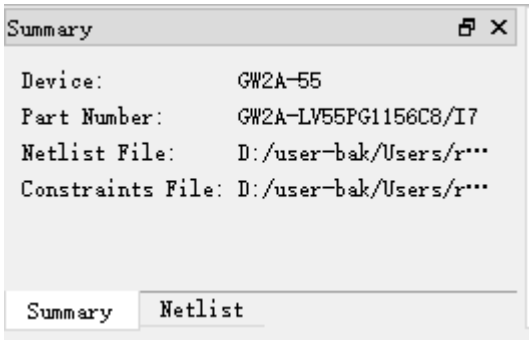
3.2.2 Summary and Netlist Windows

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

Summary Window

The summary window is shown in Figure 3-26. It displays the information of device, part number, design files and constraints files.

Figure 3-26 Summary Window



Netlist Window

As shown in Figure 3-27, Netlist window displays Ports, Primitives, Nets, and Module.

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 3-28.
- Modules are displayed in hierarchy and the number of instances in each module is also displayed, as shown in Figure 3-29.

Figure 3-27 Netlist Window

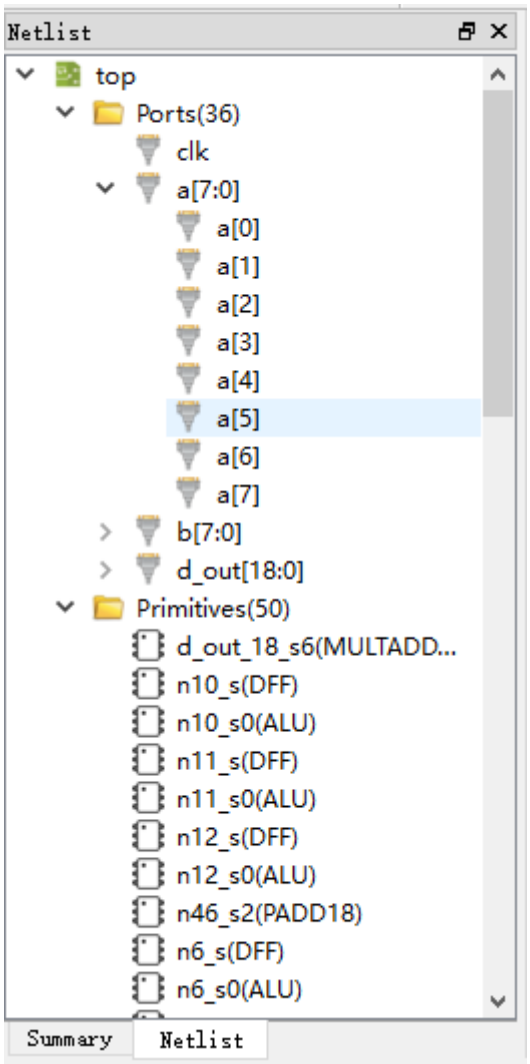


Figure 3-28 BUS and Non-Bus Display

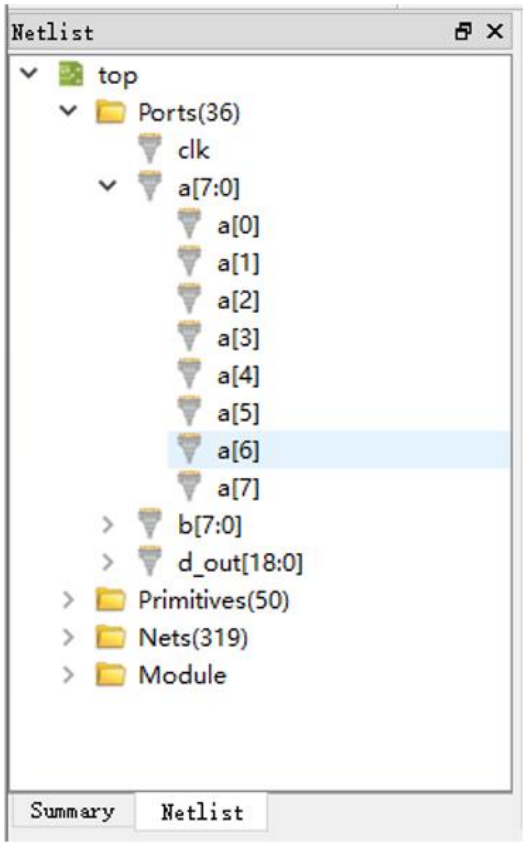
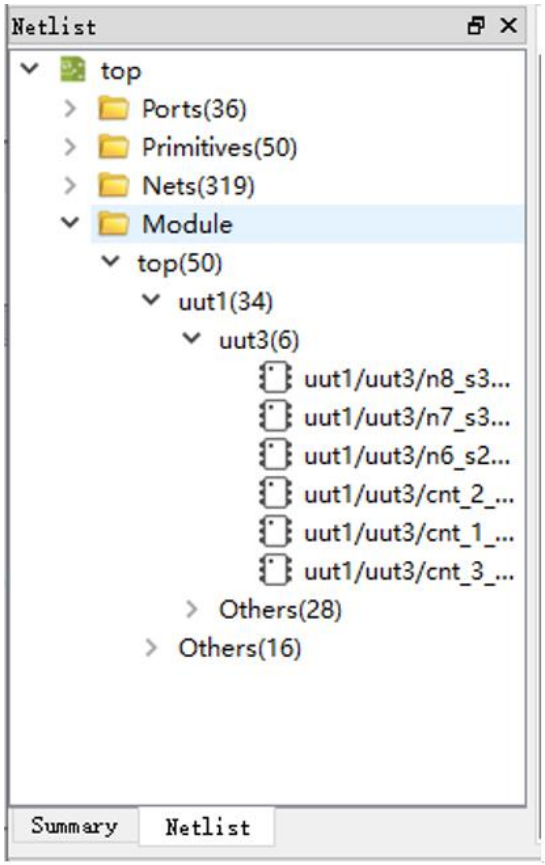


Figure 3-29 Hierarchy Display



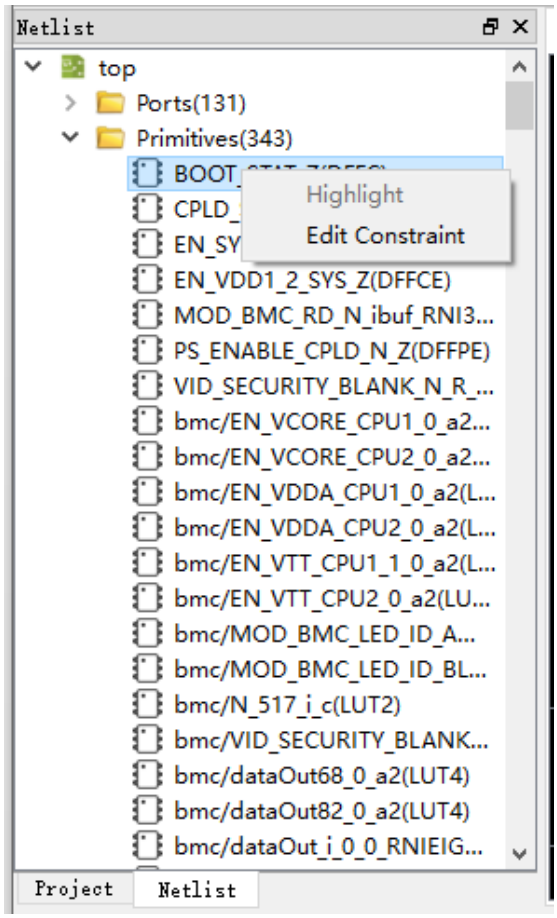
Netlist provides right-click menu as shown below.

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

Note!

If the current Primitive or Port has no constraints locations, the highlight is not available, as shown in Figure 3-30.

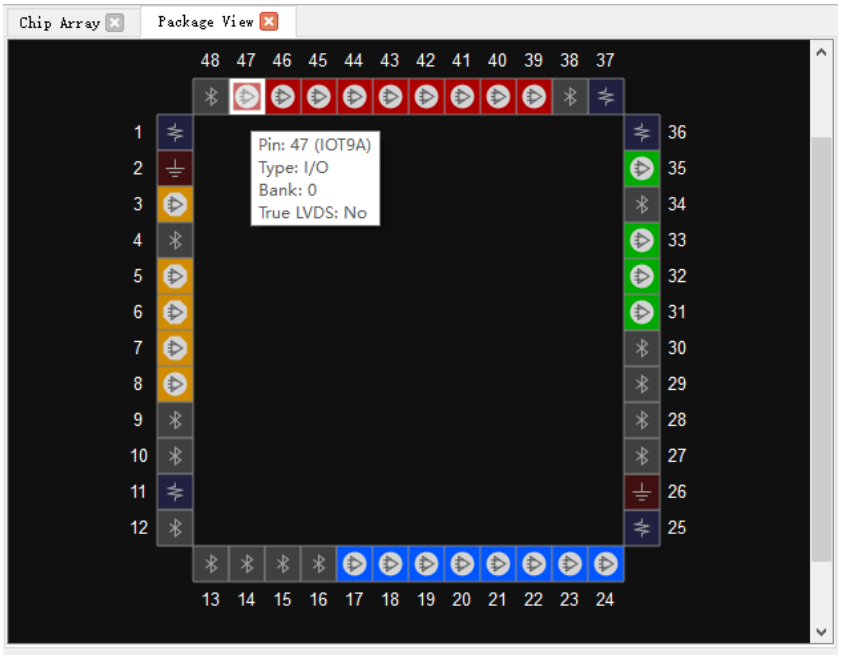
Figure 3-30 Netlist Right-clicking



3.2.3 Package View

As shown in Figure 3-31 , 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 3-31 Package View (GW1NRF-4B-QFN48)



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

- "  ": User I/O
- "  ": VCCIO
- "  ": VSS
- "  ": Bluetooth interface

The right-click menu supported by the Package View is shown in Figure 3-32. The descriptions are as follows.

- Zoom In: Zoom in Package View
- Zoom Out: Zoom out Package View
- Zoom Fit: Zoom in/out Package View to fit window size
- Show Differential IO Pairs: Display differential pair. As shown in Figure 3-33, 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 with coordinate origin at the top left corner is as shown in Figure 3-34; and the top view of

GW1N-9-WLCSP81M package with coordinate origin at the top right corner as shown in Figure 3-36.

- Bottom View: The bottom view of GW1N-9-WLCSP64 with coordinate original at the bottom right corner is as shown in Figure 3-35; and the bottom view of GW1N-9-WLCSP81M with coordinate original at the bottom left is as shown in Figure 3-37.

Figure 3-32 Package View Right-clicking

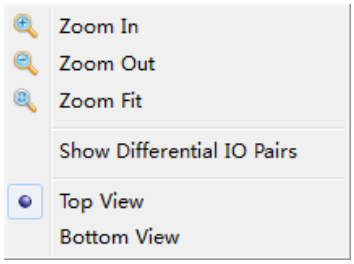


Figure 3-33 Differential Pair Display

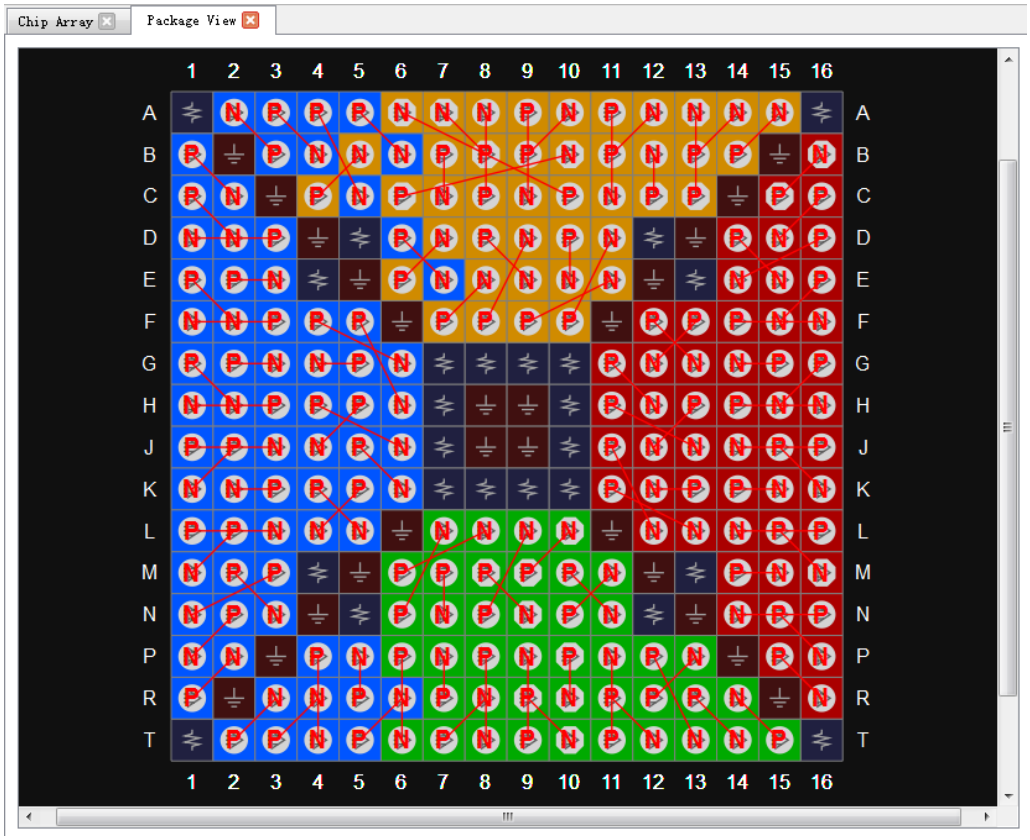


Figure 3-34 Top View

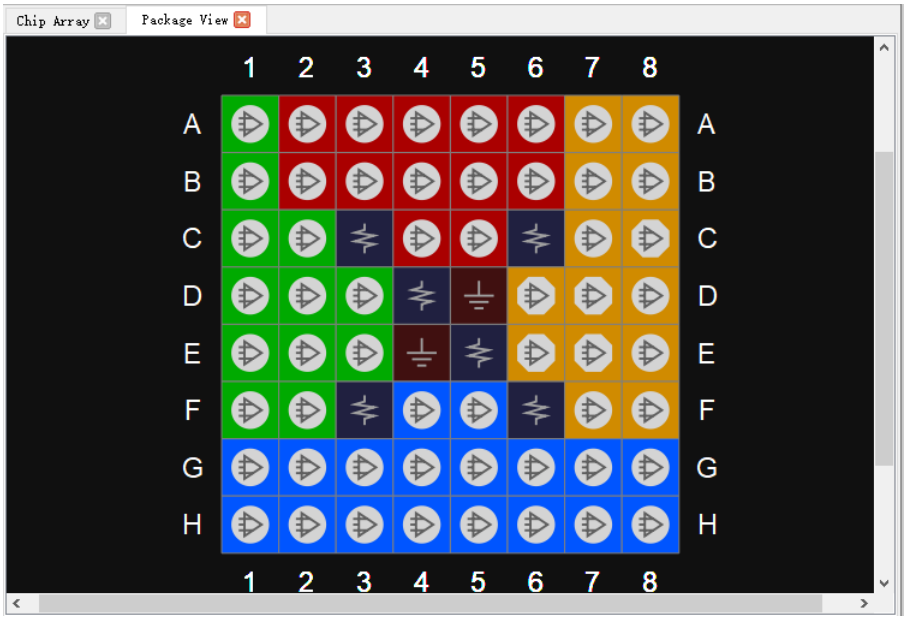


Figure 3-35 Bottom View

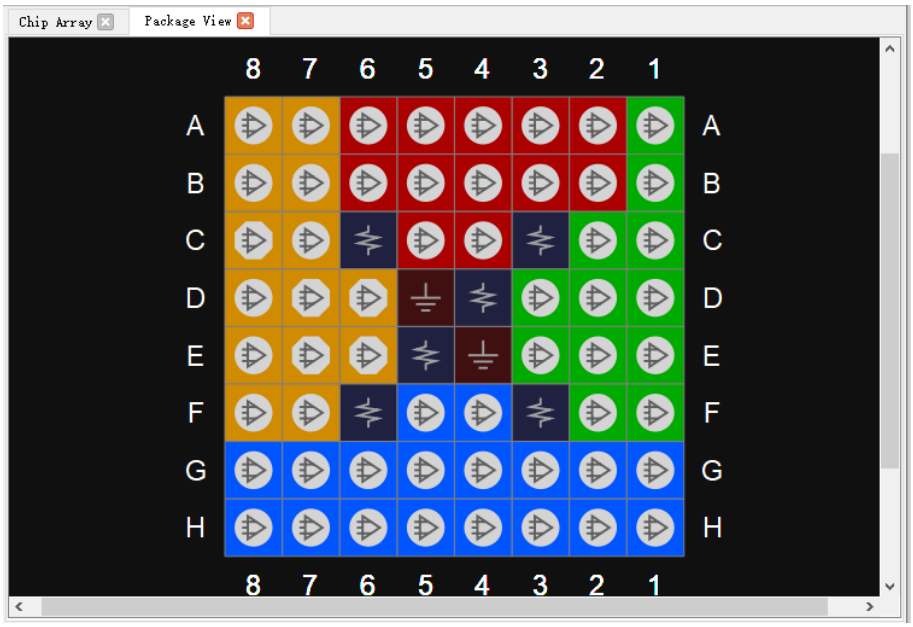


Figure 3-36 GW1N-9-WLCSP81M Top View

Figure 3-37 GW1N-9-WLCSP81M Bottom View

Package View supports the display of IO Port constraint locations, and the IO Port can be constrained by dragging from the Netlist or I/O Constraints to the Package View window. When dragged, the port name will be displayed; the unconstrained pins are grayed out and not dragged.

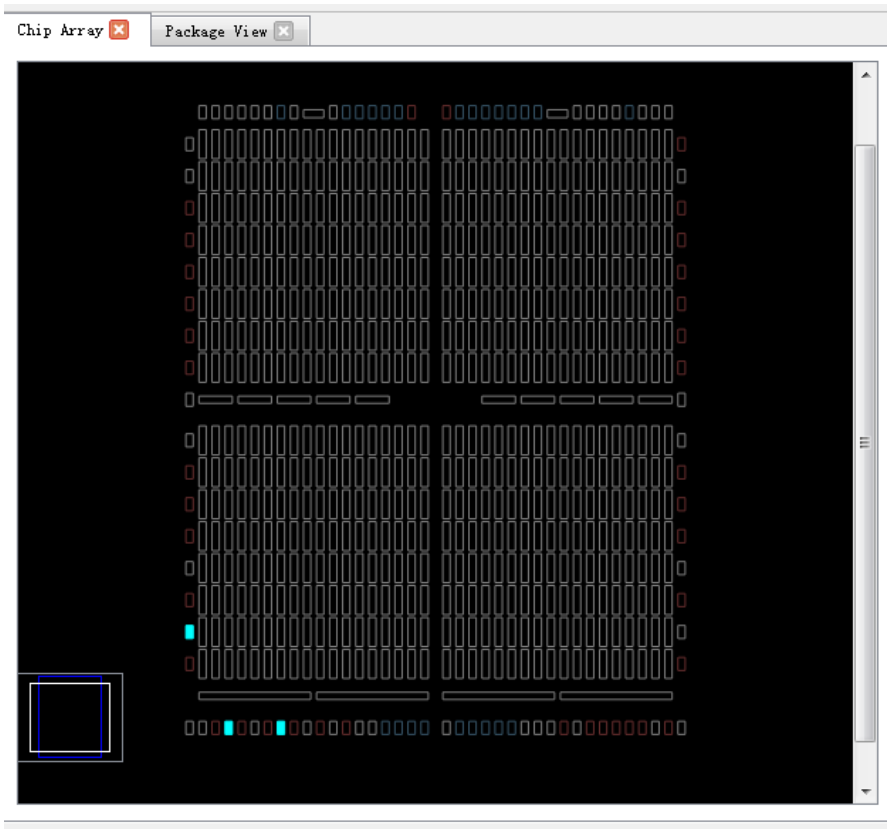
3.2.4 Chip Array Window

The Chip Array window of FloorPlanner is shown in Figure 3-38. Chip Array displays I/O, CFU, CLU, DSP, PLL, BSRAM, and DQS according to the row and column information of the chip, supports real-time display of all constraints locations, and also supports the functions of zoom in/out and dragging, etc.

I/O denotes all I/O locations of die and is distinguished with different colors.

- White: Bonded out I/O location
- Red: Unbonded I/O location
- Blue: If it is a SIP device, such as GW2AR-18, GW1NR-4, and GW1NR-9, there will be blue marked I/O.

Figure 3-38 Chip Array Window



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

- Grid mode: Display constraints locations in grid, as shown in Figure 3-39.
- Macrocell mode: Display constraints locations in CLS, IOBLK, as

shown in Figure 3-40.

- Primitive mode: Display constraints locations in REG, LUT etc., as shown in Figure 3-41.

Figure 3-39 Constraints in Grid

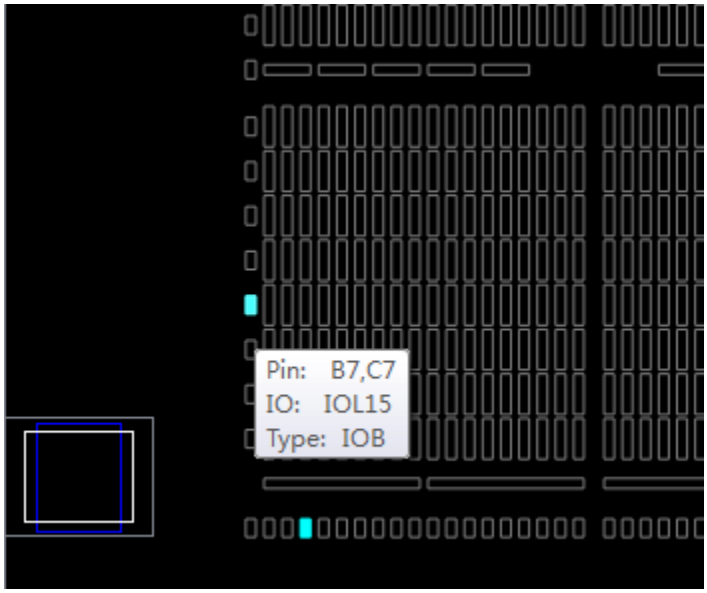


Figure 3-40 Constraints in Macrocell

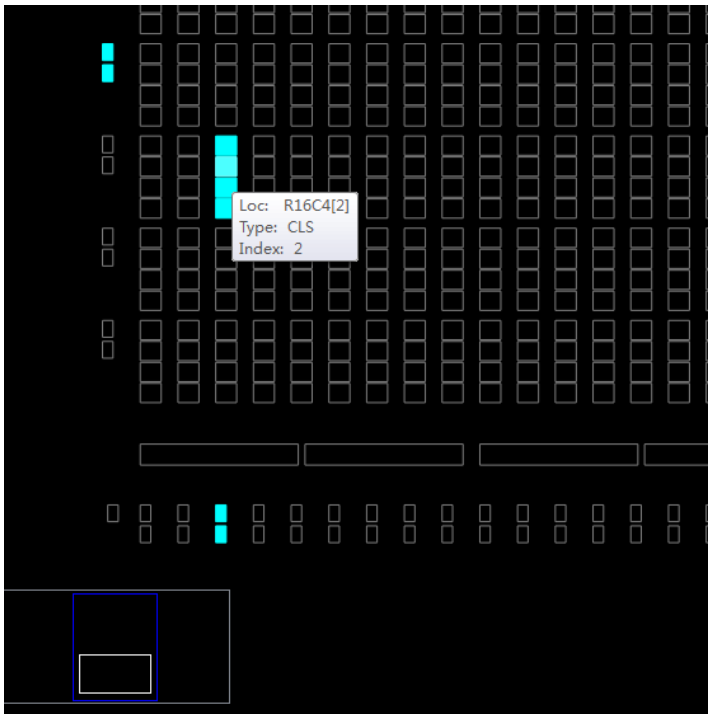
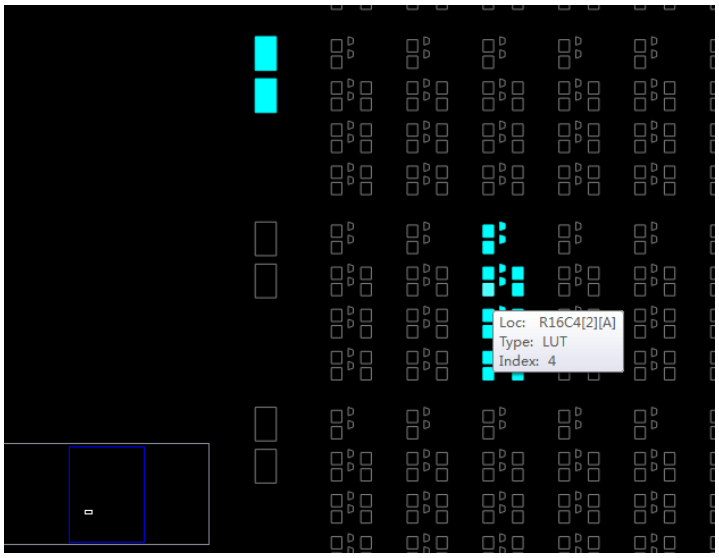


Figure 3-41 Constraints in Primitive



Chip Array supports following dragging functions.

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

There is a chip sub-window in Chip Array for real-time display of the current view relative to the device. 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 being selected or highlighted.
- Dark 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 are as follows.

- Zoom In: Zoom in Chip Array
- Zoom Out: Zoom out Chip Array
- Zoom Fit: Zoom in/out Chip Array to fit the window size
- Show Constraints View: Show instances constraints view of Chip Array
- Show Place View: Show instances place view of Chip Array; it is only effective when FloorPlanner is started after running Place & Route. Otherwise, it is greyed out.

- Show Multi-View: Show instances constraints and place views of Chip Array; it is only effective when FloorPlanner is started after running Place & Route. Otherwise, it is greyed out.
- Show In-Out Connection: Show and select the input and output connection of the instance in the Place View; it can only be used if an instance is selected when Show Place View > All Instance view opens. Otherwise, it is greyed out.
- Show In Connection: Show and select the input connection of the instance in the Place View; it can only be used if an instance is selected when Show Place View > All Instance view opens. Otherwise, it is greyed out.
- Show Out Connection: Show and select the output connection of the instance in the Place View; it can only be used if an instance is selected when Show Place View > All Instance view opens. Otherwise, it is greyed out.
- Unhighlight All: Remove highlight.
- Copy Location: Copy the selected location or area; If GRID and Block are selected, "Copy Location" in right-click menu is available. Otherwise, it is unavailable, as shown in Figure 3-42.

Show Place View also shows Lut and Reg density, as shown in Figure 3-43.

- ALL Instance: Show 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.

You can view the place of all instances in the design by clicking Show Place View > ALL Instance.

- Hover the mouse over the instance place location in the Chip Array window to display the name of the instance, as shown in Figure 3-44.
- Select a specific instance in the Netlist window; right-click and select "Highlight", the place location of that instance will be highlighted, as shown in Figure 3-45.

Note!

You can select an area by "Ctrl" + left mouse button; right click and select "Copy Location", you can copy the location and paste it to any constraint editing window.

Figure 3-42 Chip Array Right-clicking

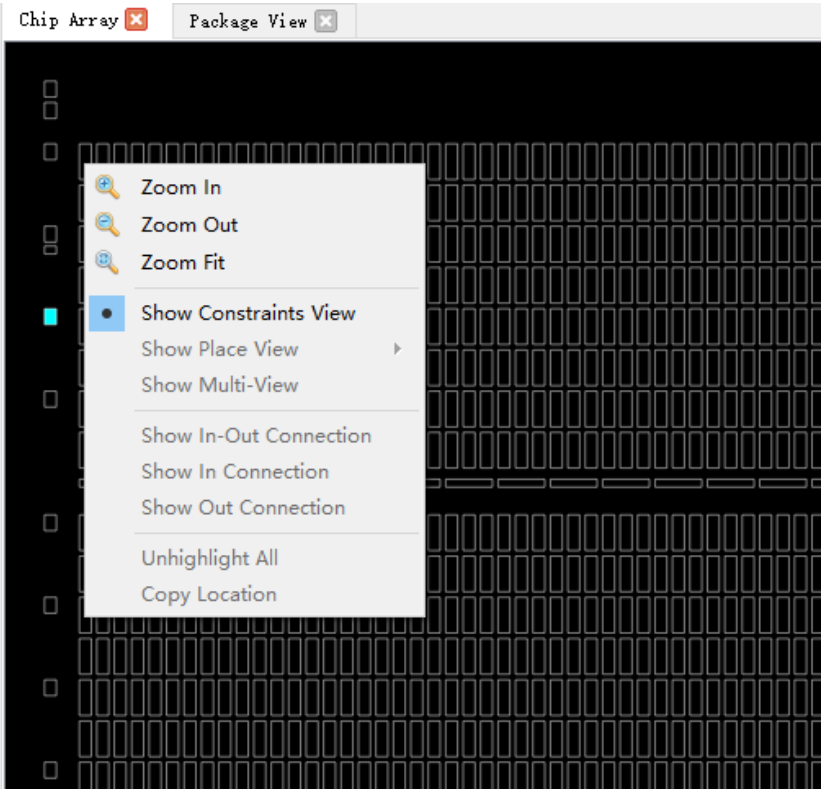


Figure 3-43 Show Place View

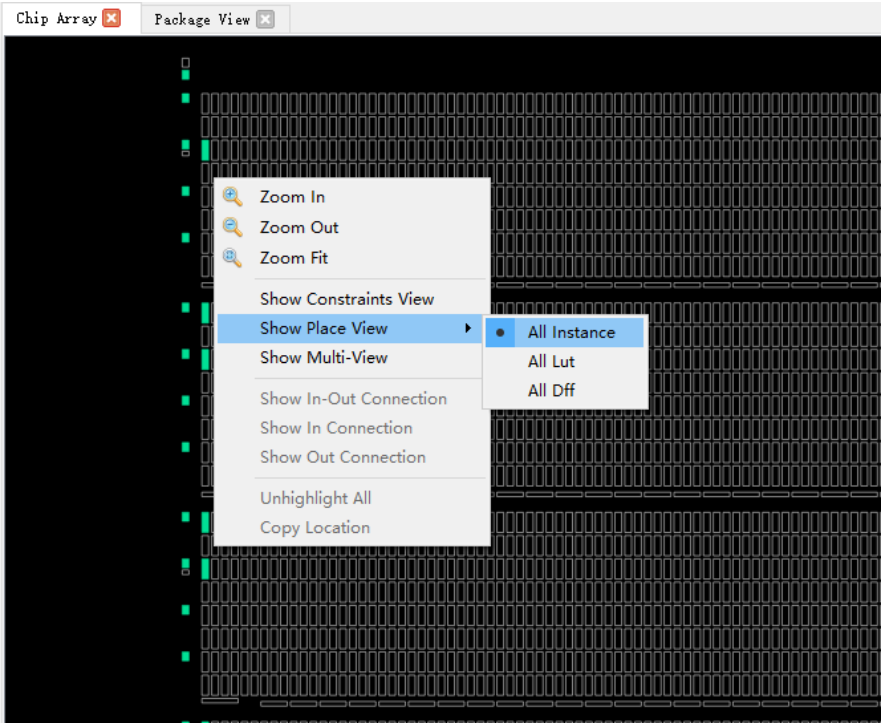


Figure 3-44 Mouse Hovering Display

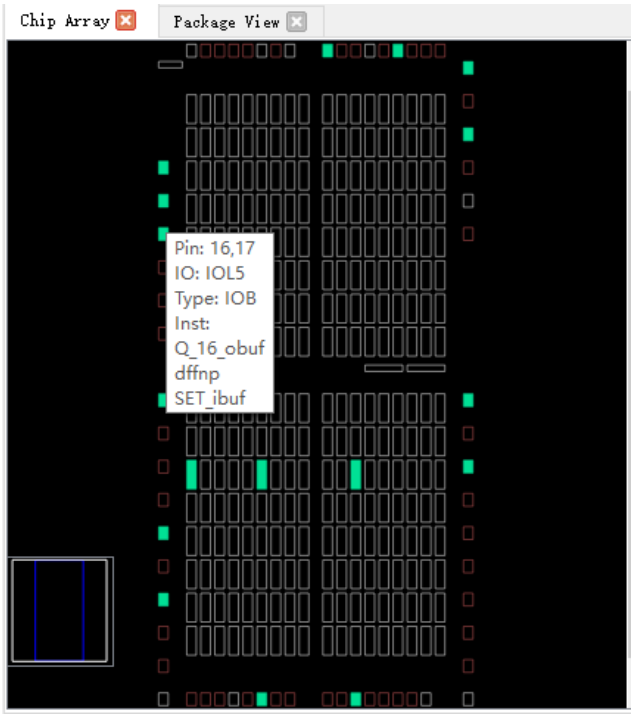
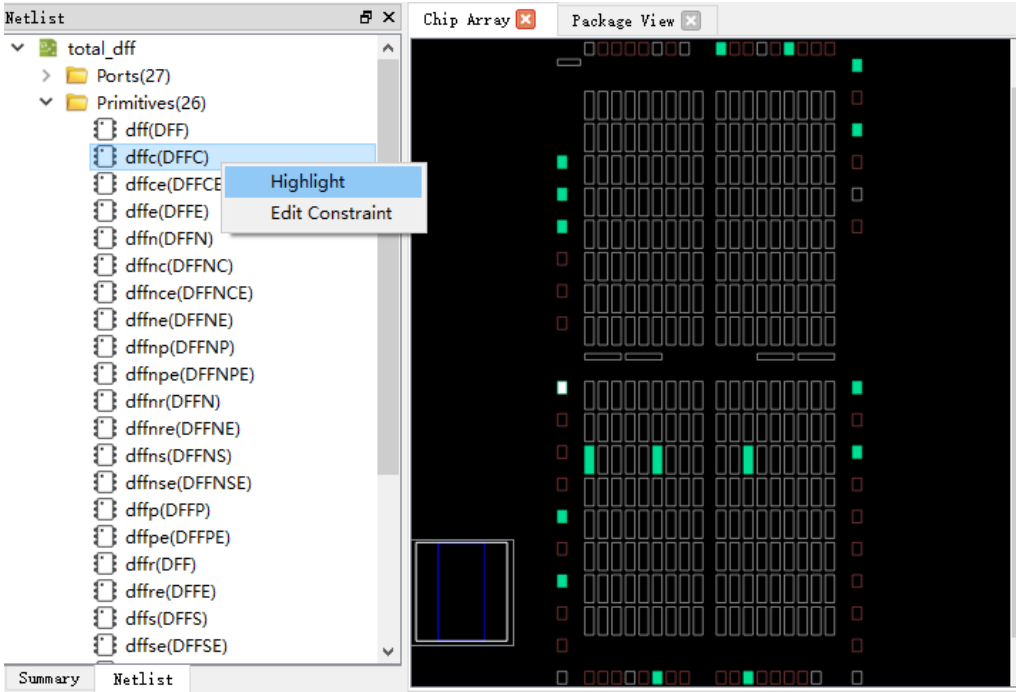


Figure 3-45 Right-click to Select Hightlight



3.2.5 Constraints Editing Window

Constraints editing 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. The brief introduction of each view is as follows.

I/O Constraints

I/O Constraints is used to constrain ports. I/O constraint view is as shown in Figure 3-46, and the functions are as follows.

- Display IO Port attributes and constraints in user design, such as Direction, Bank, IO Type, Pull Mode, 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 dragged.
- 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 remains 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 cannot be placed becomes darker.
- After setting, the constraints location in Chip Array is highlighted in light blue, and the constraints location in Package View is highlighted in orange.

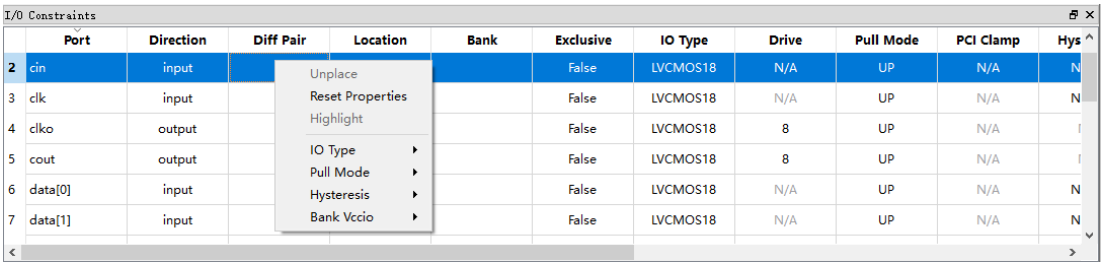
The details of the right-click menu are as follows.

- Unplace: Cancel placement
- Reset Properties: Reset Port properties
- Highlight: Highlight constraints location
- IO Type: Set the level standard
- 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
- Vref: Set reference voltage
- Single Resistor: Set the switch of single-ended resistor
- Diff Resistor: Set the switch of differential resistor
- Bank V_{CCIO}: Set BANK voltage

Note!

You can modify port attributes in batches by right-clicking; if you select multiple ports, and these ports have the same attribute values to be configured, they can be configured in batches. For the details, you can see *DS102, GW2A series of FPGA Products Data Sheet*.

Figure 3-46 I/O Constraints View



Primitive Constraints

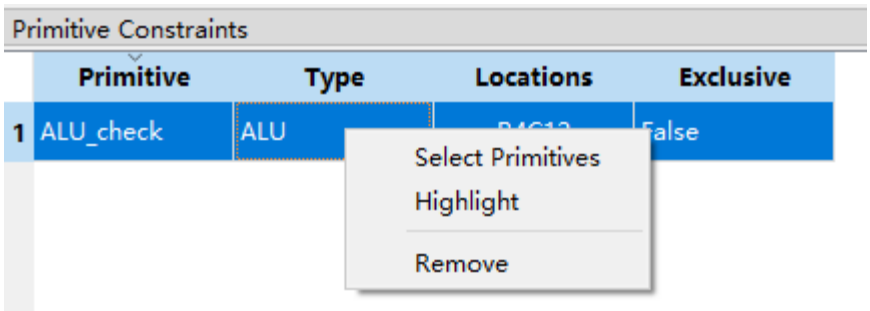
Primitive Constraint is used to constrain primitive location, as shown in Figure 3-47, and the functions are as follows:

- Display the name, type, location, and Exclusive of all Primitive constraints;
- Support editing; you can highlight, remove, add and update constraints by right-clicking.

Note!

- Modify the locations by dragging or double-clicking
- Set Exclusive by double-clicking
- Syntax and legality will be checked for the locations when manually writing primitive constraints, and error message dialog boxes are as shown in Figure 3-11 and Figure 3-12.

Figure 3-47 Primitive Constraints View



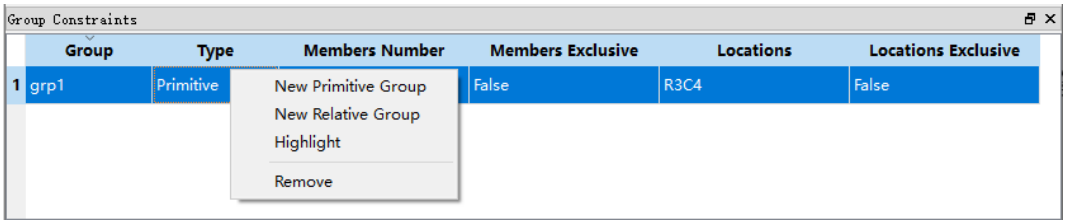
Group Constraints

Group Constraints is used for group constraints on the I/O and some

primitives in the design, the group constraint view is shown in Figure 3-48; and the functions are as follows.

- The view displays the name, type, number of primitive, location, and Exclusive of all group constraints, which includes primitive and relative group constraints. As shown in Figure 3-10 and Figure 3-14, double-click the group to edit constraints.
- You can highlight, remove, add and update constraints by right-clicking.

Figure 3-48 Group Constraints View



Resource Reservation

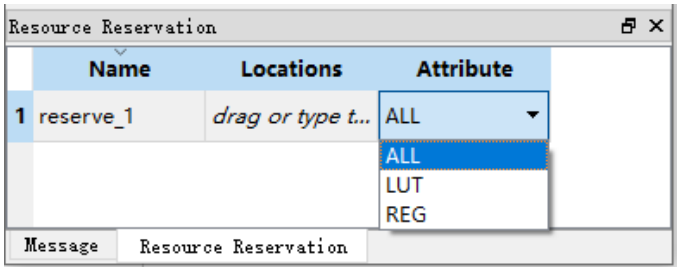
Resource Reservation is used for reservation constraints on the resources available in the current package, as shown in Figure 3-49; and the functions are as follows.

- The view can display reserved constraints locations.
- You can highlight, remove, add and update constraints by right-clicking.
- "Name" is used to distinguish utilization resource of each reservation constraint; and you cannot modify the name.

Note!

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

Figure 3-49 Resource Reservation View



Clock Net Constraints

Clock Net Constraints is used for clock constraints on the net in the

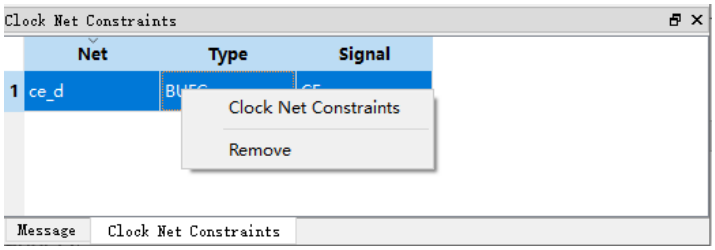
design, as shown in Figure 3-50, and the functions are as follows.

- The view displays all clock constraints.
- You can add and remove clock constraints by right-clicking.

Note!

- Double-click to edit.
- Dragging is not supported if there is no location.
- Global clock constraint creation is as shown in Figure 3-16.

Figure 3-50 Clock Net Constraints View



GCLK Primitive Constraints

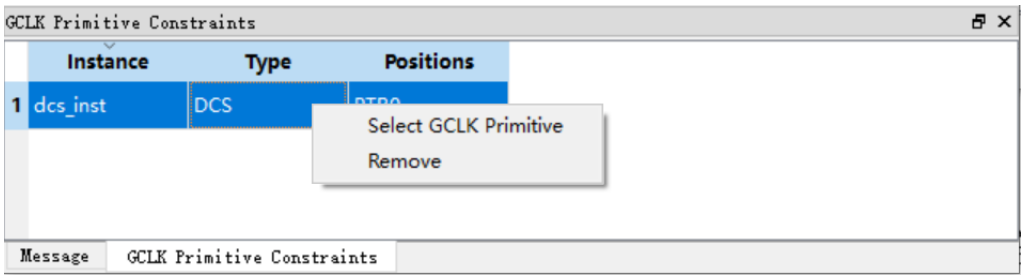
GCLK Primitive Constraints is used to constrain global clock primitives, as shown in Figure 3-51, and the functions are as follows.

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

Note!

Global clock constraint creation is as shown in Figure 3-17.

Figure 3-51 GCLK Primitive Constraints View



HCLK Primitive Constraints

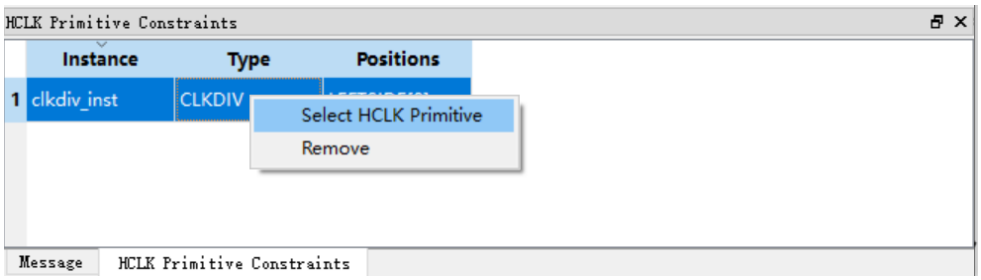
HCLK Primitive Constraints is used to constrain HCLK primitives, as shown in Figure 3-52, and the functions are as follows.

- 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 3-19.

Figure 3-52 HCLK Primitive Constraints



Vref Constraints

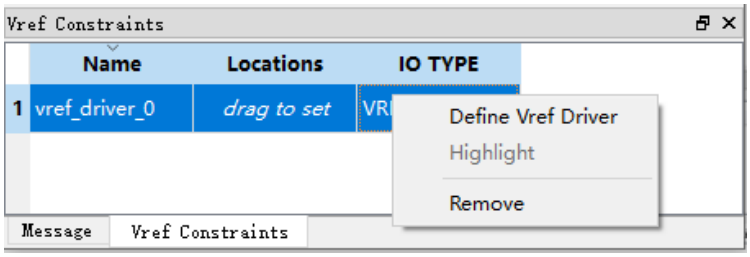
Vref Constrains is used for the external reference voltage of the bank where the constraint is located, as shown in Figure 3-53, and the functions are as follows.

- 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.

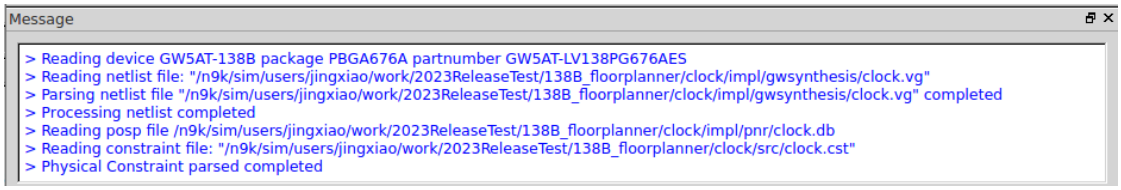
Figure 3-53 Vref Constraints View



3.2.6 Message Window

The message view is as shown in Figure 3-54, and it displays the output.

Figure 3-54 Message View



4 FloorPlanner Usage

FloorPlanner can create and edit constraints, generate physical constraint files used in Place & Route.

4.1 Create Constraints File

FloorPlanner can output newly created or modified physical constraint files, and the steps are shown below.

1. Start FloorPlanner as described in 3.1 Start FloorPlanner.
2. Click "File > New" to open the "New" dialog box.

   Note!

   You can also open "New" dialog box in the following two ways.

   - Use the "Ctrl+N" shortcut
   - Click the "New" icon in the toolbar

3. Select netlist file and part number, as shown in Figure 4-1.

Figure 4-1 New Physical Constraints

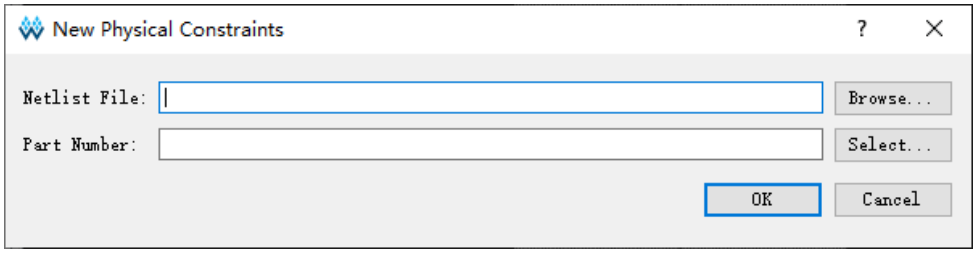
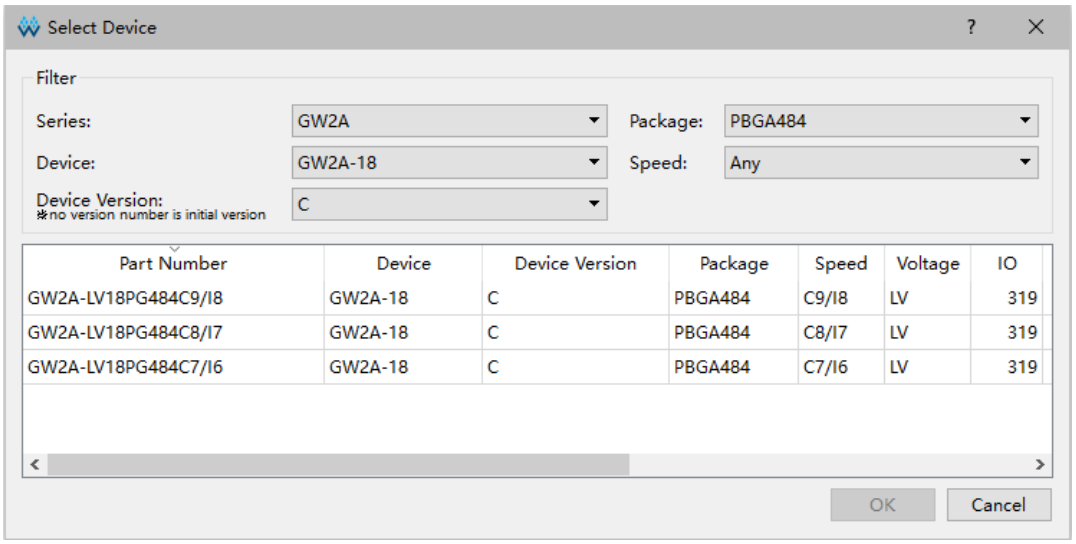


Figure 4-2 Select Device



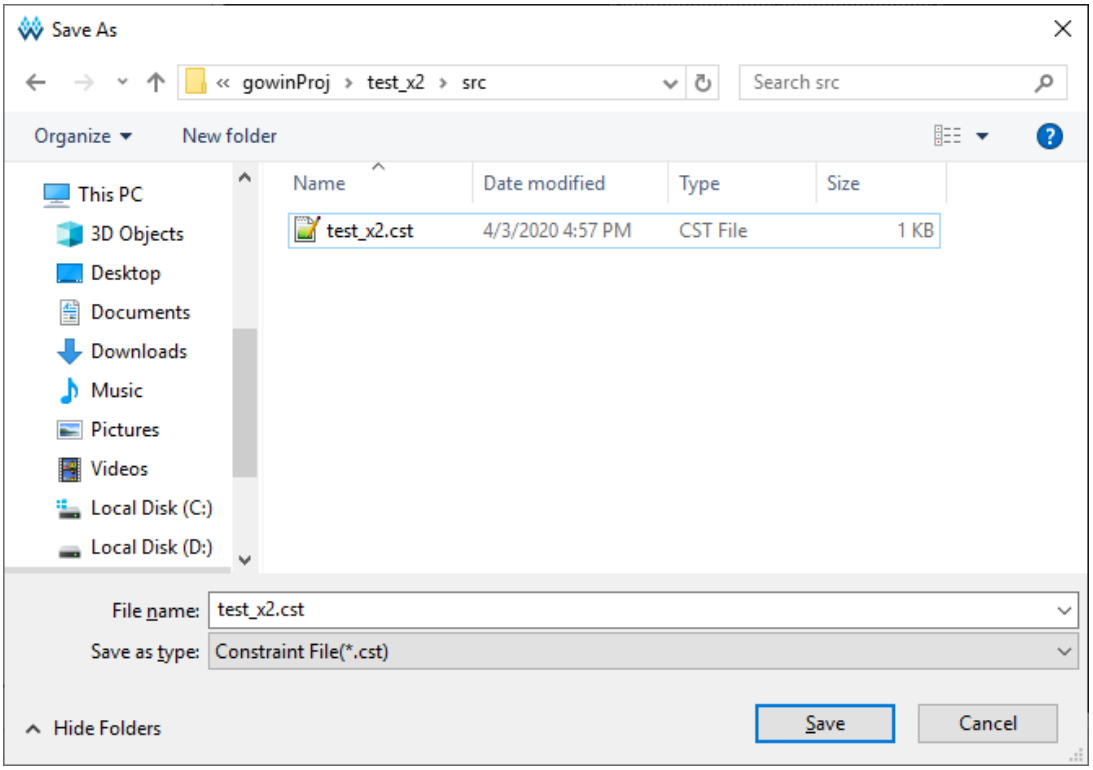
Note!

- You can select the device, package, and all Gowin FPGA devices, as shown in Figure 4-2.
- Start FloorPlanner using the first way in 3.1 Start FloorPlanner.

You can perform the 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 4-3.

Figure 4-3 Save Output File



4.2 Edit Constraints File

FloorPlanner supports constraints creation of I/O, primitive, group, resource reservation, global clock assignment, global clock primitive, high-speed clock primitive, and reference voltage, etc. Constraints can be generated via Constraints menu, see 3.2.1 Menu Bar for details.

Note!

Constraints can also be created by other ways; the following section mainly introduces how to generate constraints by dragging.

4.2.1 Constraints Examples

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

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

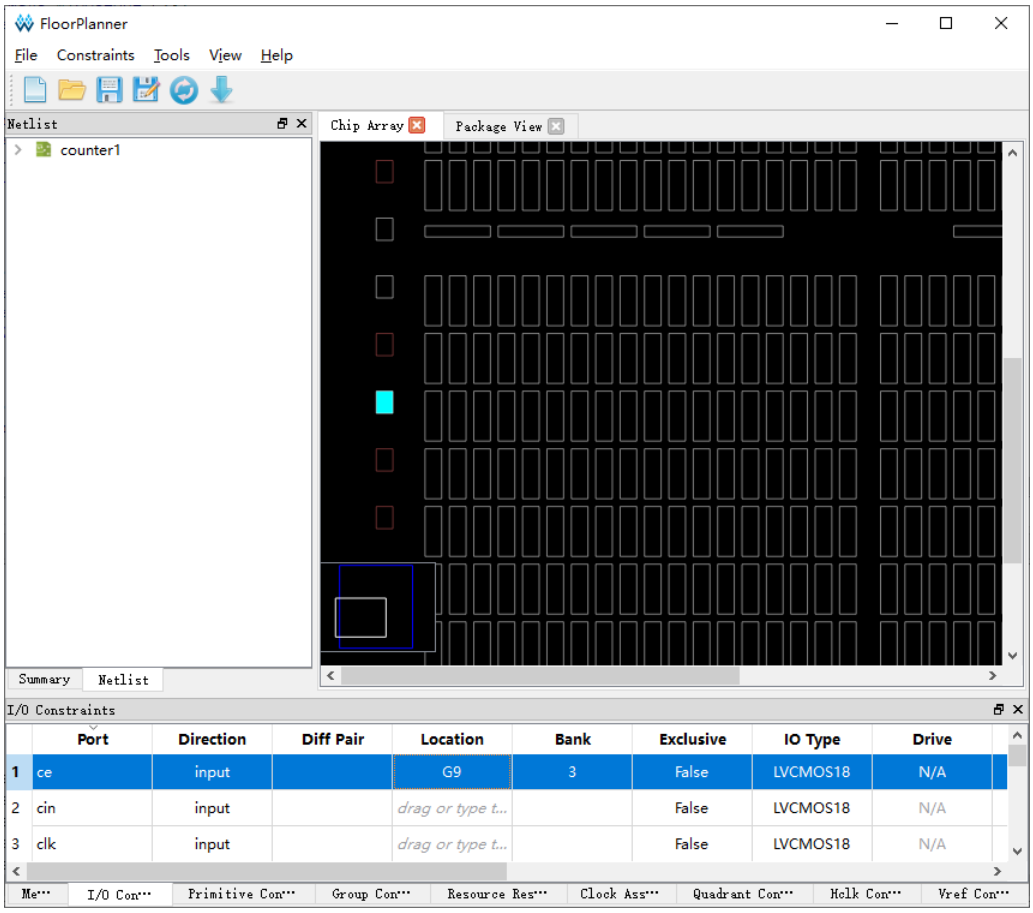
```
input [7:0] data;
input load, cin, clk;
reg [7:0] out;
always @(posedge clk)
begin
    if (load)
        out = data;
    else
        out = out + cin;
end
assign cout = &out & cin;
wire clkout;
CLKDIV clkdiv_inst (
      .CLKOUT(clkout),
      .HCLKIN(clk),
      .RESETN(1'b1),
      .CALIB(1'b0)
);
defparam clkdiv_inst.DIV_MODE = "2";
defparam clkdiv_inst.GSREN = "false";
DQCE dqce_inst (
      .CLKOUT(clko),
      .CLKIN(clkout),
      .CE(ce)
);
endmodule
```

4.2.2 Edit I/O Constraints

Drag to Chip Array to create I/O constraints.

1. Click I/O Constraints to zoom in Chip Array to macrocell mode.
2. Select Port "ce" and drag it to "G9" in Chip Array, as shown in Figure 4-4.
3. Port "ce" location is displayed as G9.

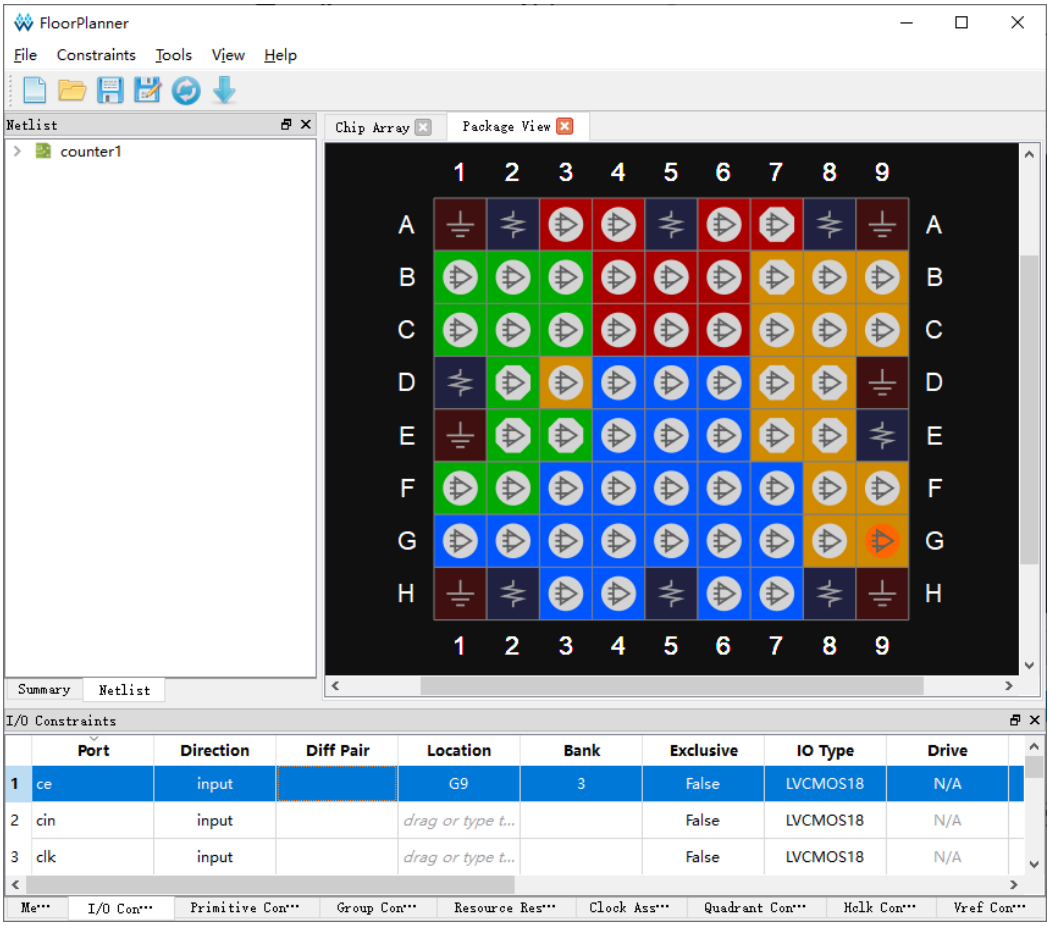
Figure 4-4 Drag to Chip Array to Create I/O Constraints



Drag to Package View to create I/O constraints.

1. Click IO Constraints.
2. Select Port "ce" and drag it to "G9" in Package View, as shown in Figure 4-5.
3. Location for Port "ce" is displayed as G9.

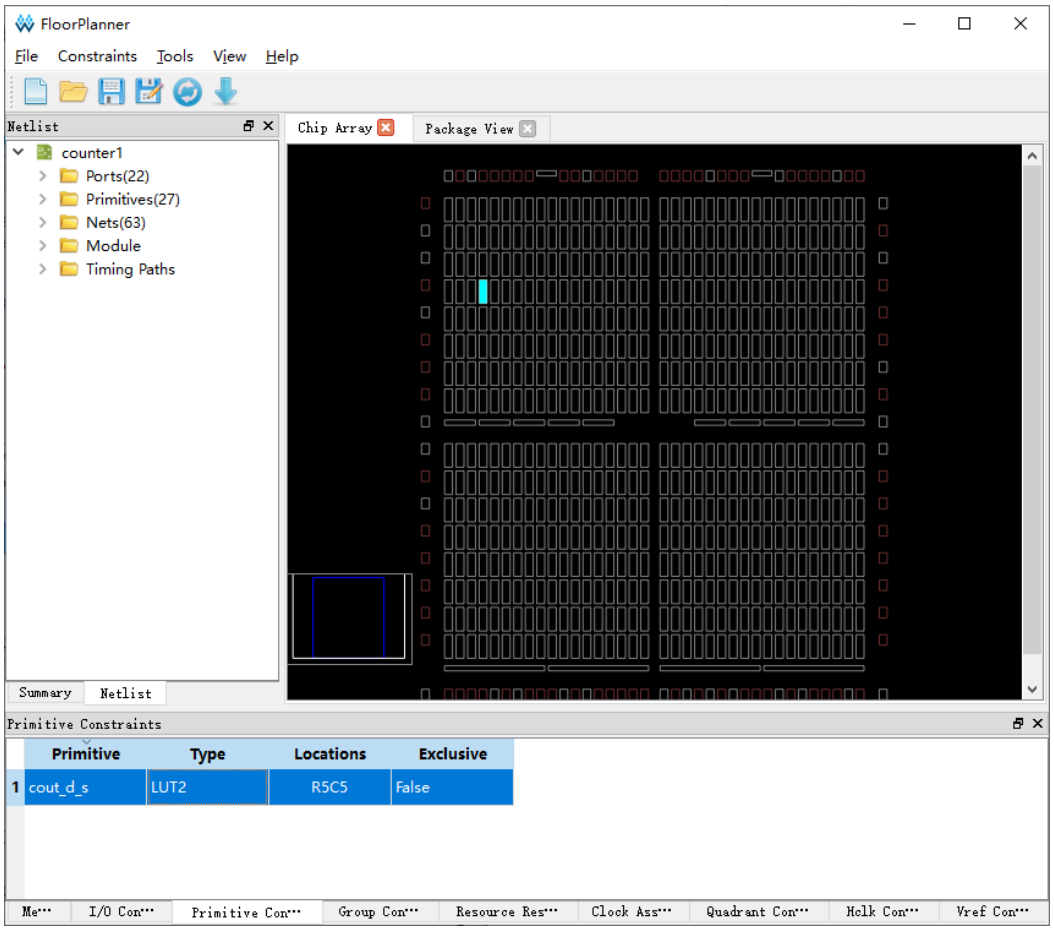
Figure 4-5 Drag to Package View to Create I/O Constraints



4.2.3 Edit Primitive Constraints

1. You can right-click the menu in "Primitive Constraints" and select "Select Primitives", then "Select Primitives" dialog box pops up. You can select Primitive "cout_d_s" and click "OK".
2. Select the created primitive constraints and drag it to "R5C5" in Chip Array, as shown in Figure 4-6.
3. Location for primitive "cout_d_s" is displayed as R5C5.

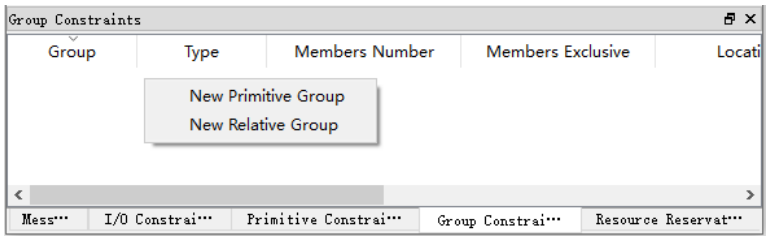
Figure 4-6 Drag to Chip Array to Create Primitive Constraints



4.2.4 Edit Group Constraints

As shown in Figure 4-7, you can create Primitive Group and Relative Group by right-clicking in Group Constraints.

Figure 4-7 Group Constraints Right-clicking



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 "Select Primitives" pops up.
3. Select "n14_s0" and "n14_s"; click "OK", then add them to Members

list.

4. Enter "R9C7" in "Locations", as shown in Figure 4-8.
5. Click "OK" in "New Primitive Group" diaog box to create primitive group constraints, as shown in Figure 4-9.

Figure 4-8 Create Primitive Group Constraints

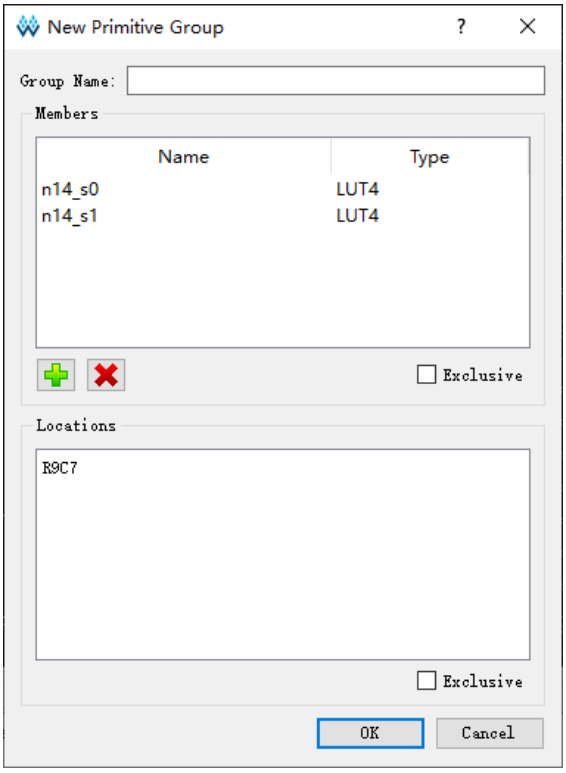
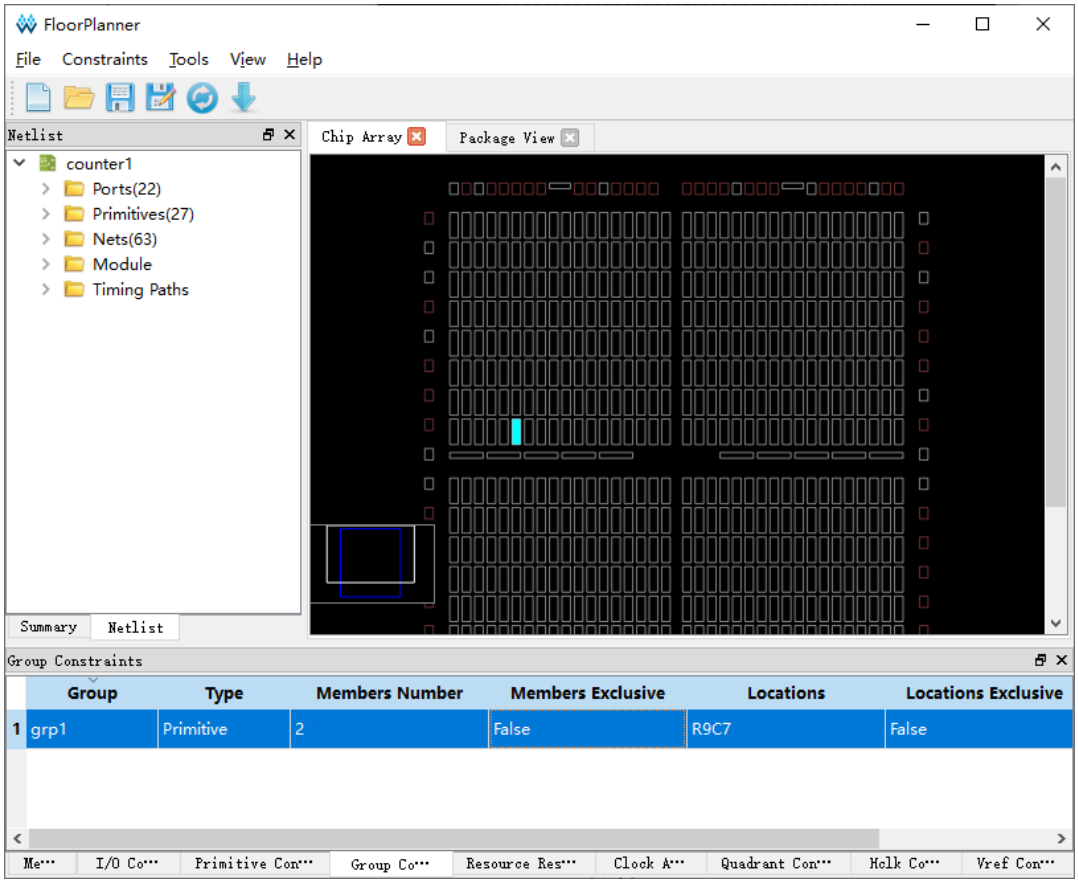


Figure 4-9 Primitive Group Constraints



Note!

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

Create Relative Group Constraints

1. Right-click "Group Constraints" and click "New Relative Group", and "New Relative Group" pops up.
2. Enter "rel_grp" and click "", and "Select Primitive" pops up.
3. Select the primitives "cout_d_s" and "n14_s0" in "Select Primitive" and click "OK", then add them to the Member list.
4. Add "R0C0" and "R4C5" to the Primitives, as shown in Figure 4-10.
5. Click "OK" in "New Relative Group" to create relative primitive group constraints, as shown in Figure 4-11.

Figure 4-10 Create Relative Group Constraints

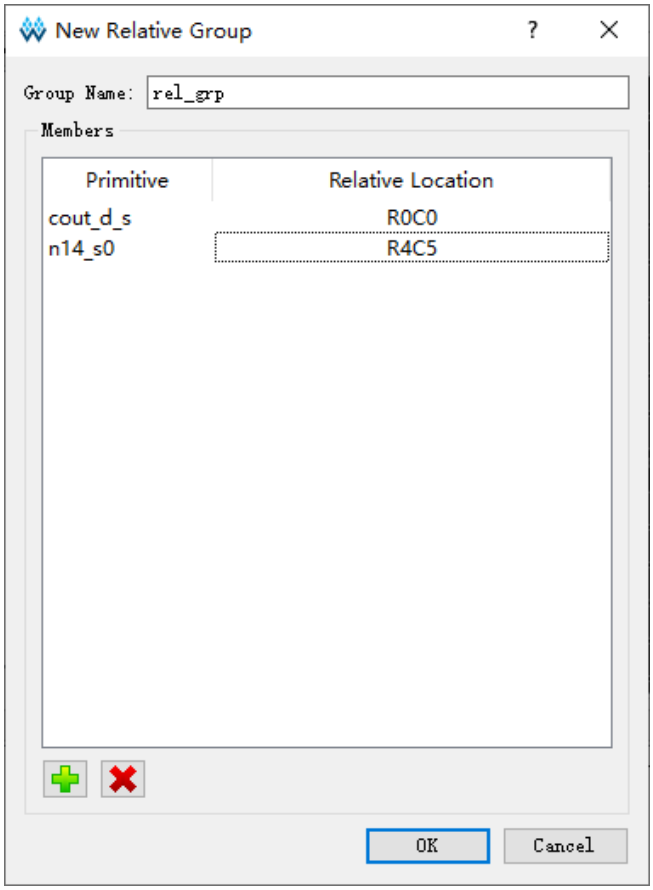
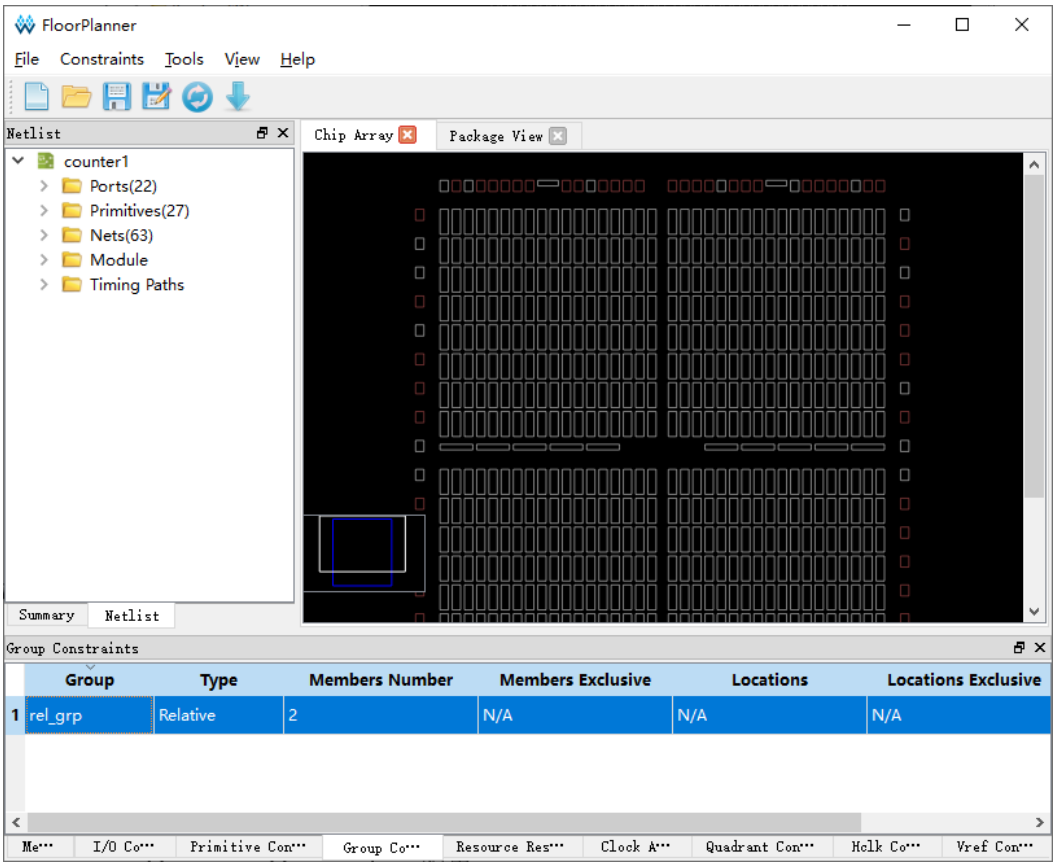


Figure 4-11 Relative Group Constraints



4.2.5 Edit Resource Reservation Constraints

1. Right-click "Resource Reservation" and click "Reserve Resources" to add resource reservation constraints, as shown in Figure 4-12.
2. Select the created resource reservation constraints and drag it to a location in Chip Array. As shown in Figure 4-13, drag to BSRAM_R10[1] to generate constraints.

Figure 4-12 Create Resource Reservation

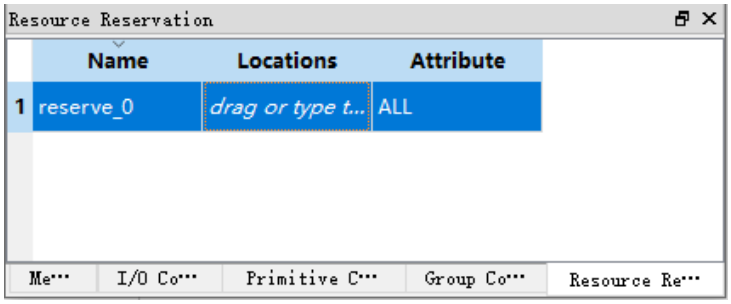
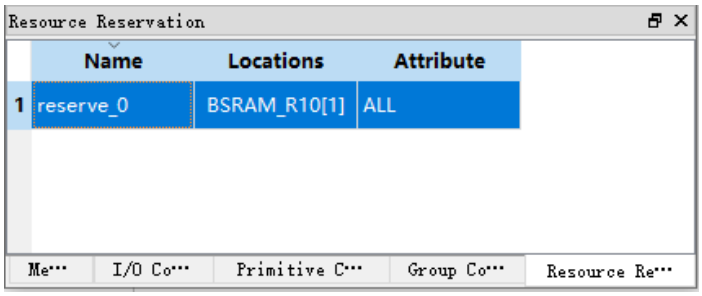


Figure 4-13 Resource Reservation



4.2.6 Edit Clock Net Constraints

1. Right-click to select "Clock Net Constraints" in Clock Net Constraints window, then "Clock Net Constraints" dialog box pops up.
2. Click "  " and "Select Net" dialog box pops up. Select a Net and click "OK" to add a net.
3. Select clock type and set signal type, as shown in Figure 4-14.
4. Click "OK" to add constraints to Clock Net Constraints, as shown in Figure 4-15.

Figure 4-14 Create Clock Net Constraints

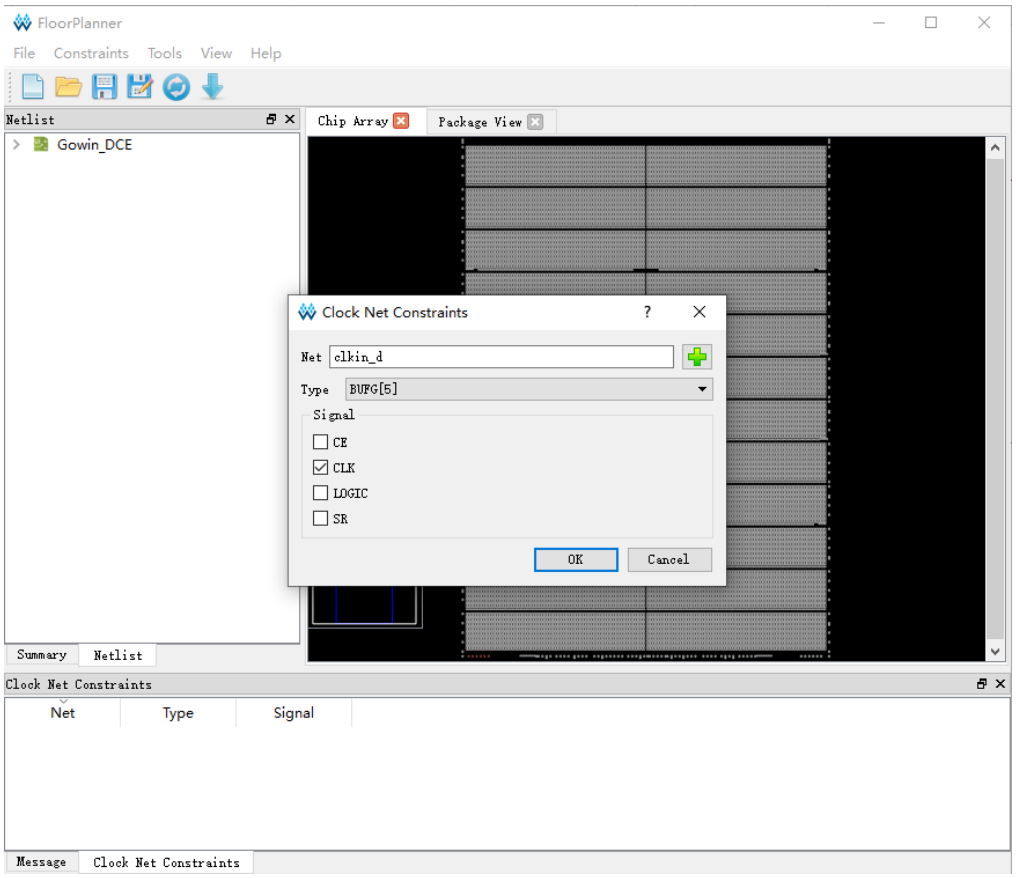
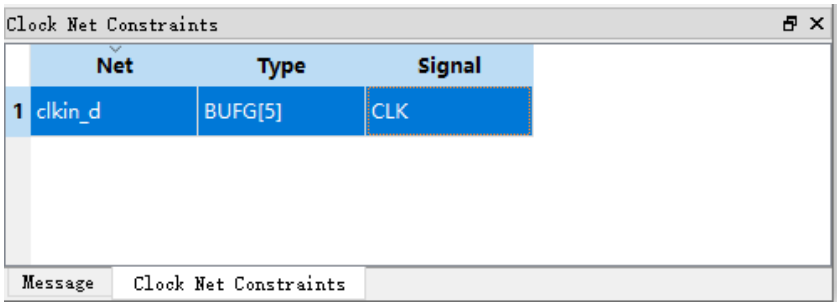


Figure 4-15 Clock Net Constraints



4.2.7 Edit GCLK Primitive Constraints

GCLK Primitive Constraints only support DCS and DQCE constraints.

1. Right-click to select "Select GCLK Primitive" in GCLK Primitive Constraints window, then "GCLK Primitive Constraints" dialog box pops up.
2. Click "  " and "GCLK Primitive" dialog box pops up. Select a "Instance" and click "OK".
3. Select a GCLK position in "Position", as shown in Figure 4-16.
4. Click "OK" to add this constraint, as shown in Figure 4-17.

Figure 4-16 Create GCLK Primitive Constraints

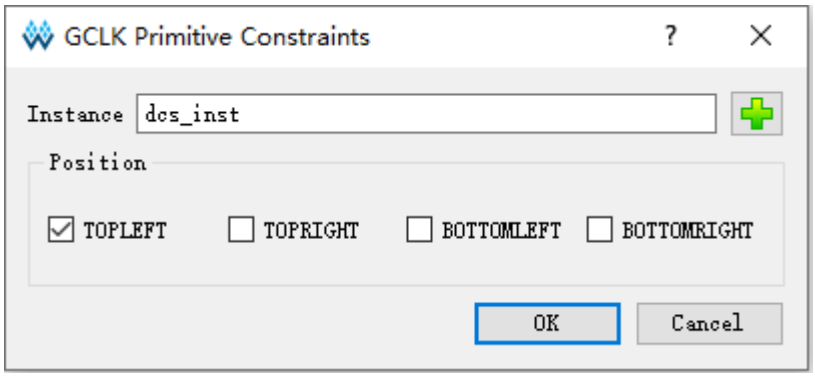
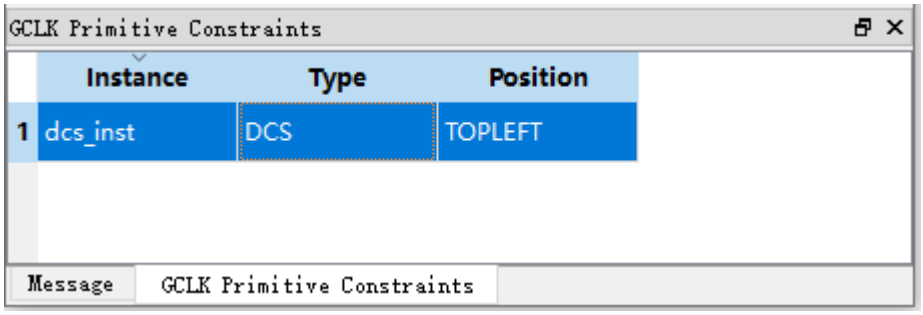


Figure 4-17 GCLK Primitive Constraints



4.2.8 Edit HCLK Primitive Constraints

HCLK Primitive Constraints only support CLKDIV and DLLDLY constrains.

The steps are as follows.

1. Right-click to select "Select HCLK Primitive" in HCLK Primitive Constraints window, then "HCLK Primitive Constraints" dialog box pops up.
2. Click "  " and "HCLK Primitive" dialog box pops up. Select a "Instance" and click "OK".
3. Select a HCLK position in "Position", as shown in Figure 4-18.
4. Click "OK" to add the constraints, as shown in Figure 4-19.

Figure 4-18 Create HCLK Primitive Constraints

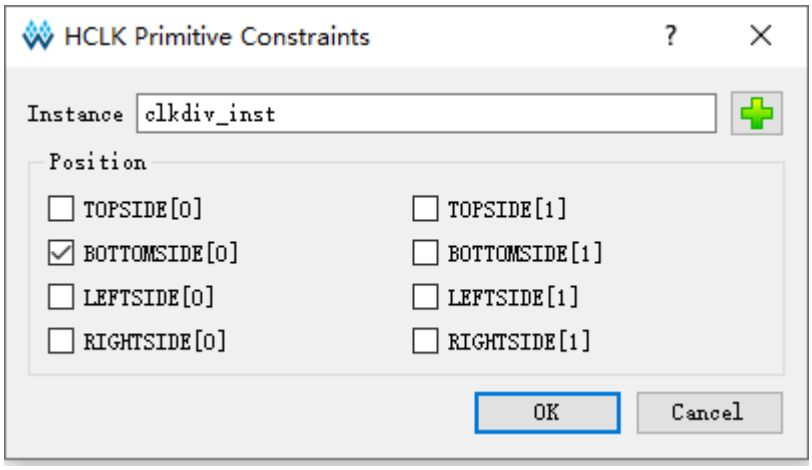
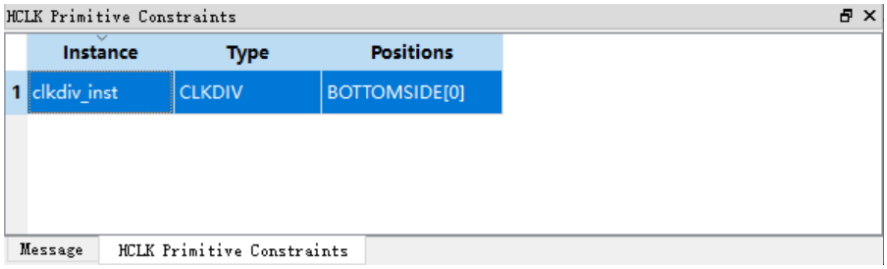


Figure 4-19 HCLK Primitive Constraints



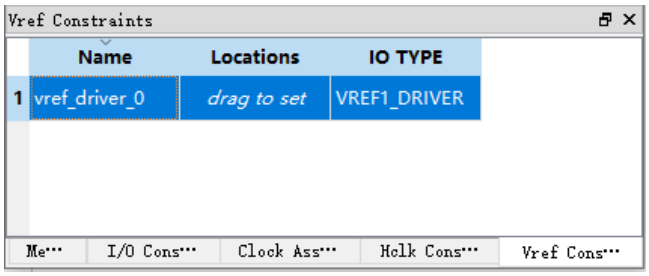
4.2.9 Edit Vref Constraints

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 4-20.

2. Zoom in Chip Array to macrocell mode. Select the created Vref Constraints and drag it to B7 in Chip Array. The location of the Vref Constraints is displayed as "B7", as shown in Figure 4-22.

Figure 4-20 Create Vref Constraints



You can customize Vref constraint name, but duplicate names are not allowed; if there are duplicate names, you will be prompted, as shown in Figure 4-21.

Figure 4-21 Prompt

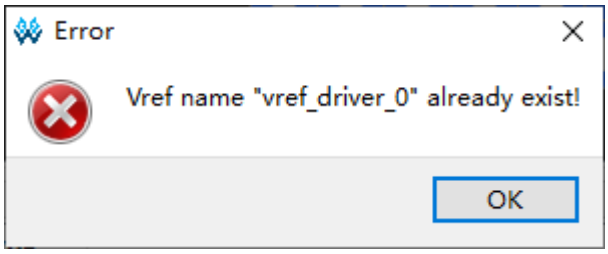
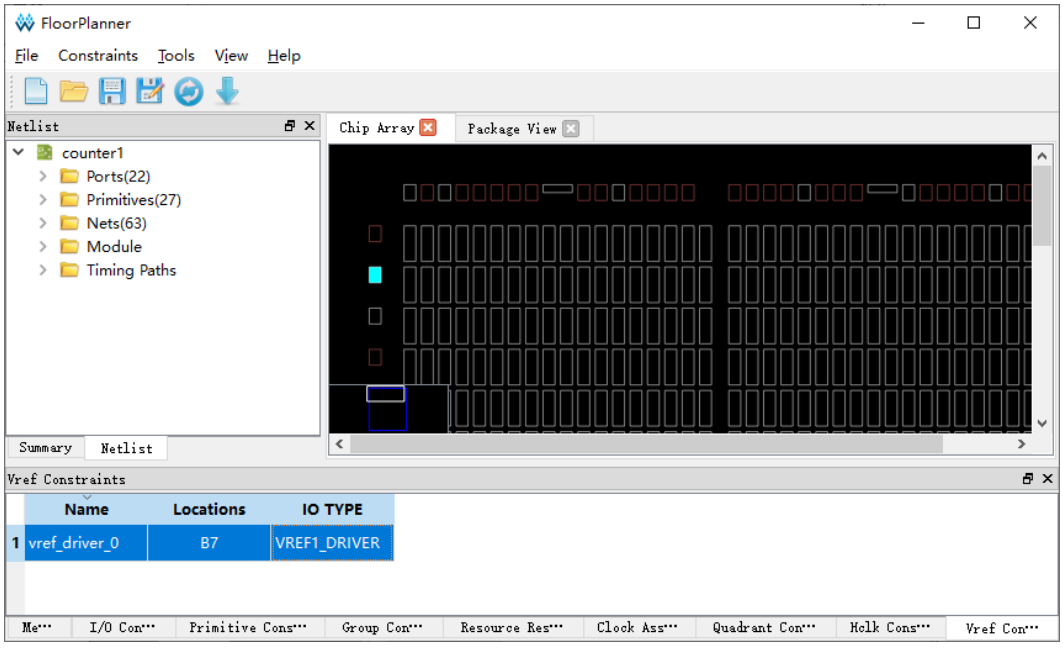


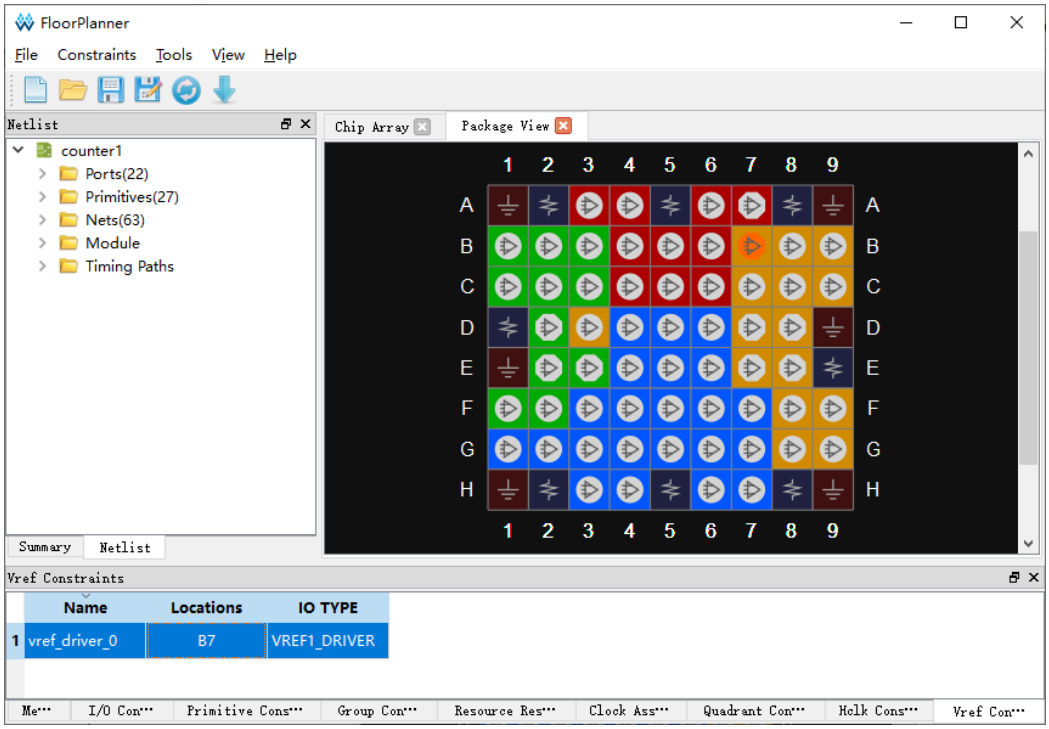
Figure 4-22 Drag to Chip Array to Generate Vref Constraints Location



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 4-20.
2. Select the newly created Vref Constraints and drag it to B7 in Package View. The location of the Vref Constraints is displayed as "B7", as shown in Figure 4-23.

Figure 4-23 Drag to Package View to Generate Vref Constraints Location



Appendix A Physical Constraints Syntax Definition

A.1 I/O Location Constraints

The IO location constraints can be used to constrain port and Buffer to the specified IOB location.

Syntax

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

Constraints Elements

obj_name

Obj_name can be the name of port or Buffer.

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 and follows the constraint location, indicating that only the primitive specified by obj_name can be placed in the obj_location of the constraint statement.

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 constrained to the pin A1.

Example 2

IO_LOC "io_1" A1, B14, A15;

// io_1 should be constrained to 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 constrained 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 constrained to pin A1, B14, or A15, and all these locations can only be used by io_2.

A.2 I/O Attributes Constraints

I/O attributes constraints can used to set attribute values for ports, such as IO_TYPE, PULL_MODE and DRIVE, etc. You can see *DS102, GW2A series of FPGA Products Data Sheet* 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 PULL_MODE =KEEPER;

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

Example 3

IO_PORT "port_3" IO_TYPE = LVDS25;

// Buffer 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/DQS 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:

LUT Constraints Location

- 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 CLS 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]"

Note!

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

The keyword "exclusive" is optional and follows the constraint location, indicating that only the primitive specified by obj_name can be placed in the obj_location of the constraint statement.

Examples

Example 1

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

// lut_1 is constrained to the R2C3 and the first CLS of the R5C10 in the first LUT.

Example 2

INS_LOC "ins_2 " R5C6[2] exclusive;

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

Example 3

INS_LOC "ins_3" R[2:6]C2;

// ins_3 is constrained to the range between the second to sixth rows and the second column.

Example 4

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

// ins_4 is constrained to the range between second row to fourth rows and the second to sixth columns, and the location can only be used by this instance.

Example 5

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

// ins_5 is constrained to the 2nd CLS of any GRID in the range between the second to fourth rows and the second to sixth columns.

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 by INS_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 row 19 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 CLSs). 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 example is as follows.

Example 10

LUT4 Constraints

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

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

Example 11

CLS Constraints

INS_LOC "lut5_name" R5C15[3];

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

Example 12

CLS Constraints

INS_LOC "lut6_name" R5C15[0];

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

Example 13

GRID Constraints

INS_LOC "lut7_name" R5C15;

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

Example 14

GRID Constraints

INS_LOC "lut8_name" R5C15；

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

Example 15

DSP MACRO Constraints

INS_LOC "mult_name" DSP_R19[1][A];

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

A.4 Group Constraints

The group constraints include 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, BSRAM, SSRAM, DSP, PLL, DQS, or Buffer, IOLOGIC 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 be at IOB, GRID, BSRAM, DSP, PLL.

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 only the ins_5, ins_6, ins_7 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[2:3]C[2:4] exclusive;

// The Instance in group_2 can be placed in the range of R[2:3]C[2:4], and the instances in group_2 can only be placed in the range.

Example 6

GRP_LOC group_3 PLL_L, BSRAM_R10[0], DSP_R19[0];

The Instance in group_3 can be placed at PLL_L, BSRAM_R10[0], DSP_R19[0].

A.4.2 Relative Group Constraints

The instance such as LUT, REG, MUX relative location constraints 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.

Example

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 Constraints

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 during placement.

A.6 Vref Constraints

The chip supports 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.

Note!

- The input pin location where the external reference voltage can be set must have IOLOGIC resource.
- Vref Constraints and PORT constraints are valid when used together. When a single-ended input or inout port with IO Type SSTL/HSTL, the Vref attribute can be set to the created Vref Constraints, indicating that the reference voltage of this port uses the external reference voltage input from the Vref Constraints location.

Syntax

"USE_VREF_DRIVER" vref_name [location]";"

Constraints Elements

vref_name

Customized VREF pin name

location

Any IO location with IOLOGIC in the device 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_I VREF=vref_pin;

IO_PORT “port_2” IO_TYPE = SSTL25_I 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 E16;

IO_LOC "port_1" C16;

IO_PORT "port_1" IO_TYPE = SSTL25_I VREF=vref_pin;

// Define a VREF pin named "vref_pin" and constrain it to E16. Set the VREF of port_1 to vref_pin and place it at C16, and port_1 location needs to be in the same bank as E16.

A.7 GCLK Primitive Constraints

GCLK Primitive Constraints is used to constrain objects such as DCS/DQCE to a specified position.

Syntax

INS_LOC "obj_name" position;

Constraints Elements

obj_name

The name of the constraint object.

position

LittleBee® family: GW1N-9, GW1NR-9, GW1N-9C, GW1NR-9C can constrain 4 positions of "TOPLEFT", "TOPRIGHT", "BOTTOMLEFT", "BOTTOMRIGHT"; other devices can only constrain 2 positions of "LEFT", "RIGHT".

Arora family can constrain 4 positions of "TOPLEFT", "TOPRIGHT", "BOTTOMLEFT", "BOTTOMRIGHT".

Example

INS_LOC "dcs_name" LEFT;

// Constrain the DCS dcs_name to the LEFT position.

A.8 Clock Net Constraints

Clock Net Constraints is used to constrain a specific net to the global clock wire or non-clock wire in the design; it can implement the global clock constraints for the net of specific signal_type (CLK/CE/SR/LOGIC).

- BUFG[0-7] means the net is routed to PCLK.

- BUFS means the net is routed to SCLK.
- LOCAL_CLOCK means this net is not routed to the global clock wire.

The CLK signal is the signal connected to the clock pin. The CE signal is the signal connected to the clock enable pin. The SR signal is the signal connected to the SET/RESET/CLEAR/PRESET pins, and the LOGIC is the signal connected to logic input pins.

Syntax

CLOCK_LOC "net_name" global_clocks = signal_type ;

Constraints Elements

net_name

The net name

global_clocks

BUFG[0-7] means the net is routed to a specific PCLK.

BUFG means the net is routed to PCLK.

BUFS means the net is routed to SCLK.

LOCAL_CLOCK means this net is not routed to the clock wire.

signal_type

CLK: signal_type is the net of the clock pin.

CE: signal_type is the net of the clock enable pin.

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

LOGIC: signal_type is the net other than the signal_type above.

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

Note!

If LOCAL_CLOCK is selected for LOCAL_CLOCK, signal_type is not available.

Examples

Example 1

CLOCK_LOC "net" BUFG[0] = CLK;

// Constrain the net whose signal_type is the clock pin to the first PCLK.

Example 2

CLOCK_LOC "net" BUFG = CLK|CE;

NET_LOC "net" BUFG = CLK|CE;

// Constrain the net whose signal_type is the clock pin/clock enable pin to the PCLK.

Example 3

CLOCK_LOC "net" BUFS = CE;

NET_LOC "net" BUFS = CE;

// Constrain the net whose signal_type is the clock enable pin to SCLK.

Example 4

CLOCK_LOC "net" LOCAL_CLOCK;

// Constrain the net to the non-clock wire.

A.9 HCLK Primitive Constraints

HCLK Primitive Constraints is used to constrain CLKDIV and DLLDLY to the HCLK positions. The constraints positions of CLKDIV/DLLDLY are different from the ones of other general instances. The "TOPSIDE", "BOTTOMSIDE", "LEFTSIDE", and "RIGHTSIDE" indicate the four sides of the constraint positions.

Syntax

INS_LOC "obj_name" position;

Constraints Elements

obj_name

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

position

"TOPSIDE[0-1]"

"BOTTOMSIDE[0-1]"

"LEFTSIDE[0-1]"

"RIGHTSIDE[0-1]"

Example

INS_LOC "clkdiv_name" TOPSIDE[0];

// Constrain the clkdiv_name to TOPSIDE[0].

A.10 Other Constraints

A.10.1 JTAGSEL_N Net Constraints

When the internal logic of the FPGA is used to control JTAGSEL_N functions, i.e., during the second download without power off, pull down JTAGSEL_N so that JTAG can be switched to the configuration function, and net physical constraints of JTAGSEL_N need to be added. For the details, you can refer to *UG290, Gowin FPGA Products Programming and Configuration User Guide*.

Syntax

NET_LOC "obj_name" V_JTAGSEL_N;

Constraint Element

obj_name

Any net that can be wired in the internal logic can be used as obj_name

Example

NET_LOC "netname" V_JTAGSELN;

// The netname net is used to control the functions of JTAGSEL_N.

A.10.2 RECONFIG_N Net Constraints

When the internal logic of the FPGA is used to control RECONFIG_N functions, i.e., during the second download without power off, pull down RECONFIG_N so that JTAG can be switched to the configuration reset function, and net's physical constraints of RECONFIG_N need to be added. For the details, you can refer to *UG290, Gowin FPGA Products Programming and Configuration User Guide*.

Syntax

NET_LOC "obj_name" V_RECONFIGN;

Constraint Element

obj_name

Any net that can be wired in the internal logic can be used as obj_name.

Application examples

Example

NET_LOC "netname" V_RECONFIGN;

// Use this netname to control the RECONFIG_N function.


GOWIN
PROGRAMMING FOR THE FUTURE