



Arora V Design Physical Constraints

User Guide

SUG1018-1.2E, 08/18/2023

Copyright © 2023 Guangdong Gowin Semiconductor Corporation. All Rights Reserved.

GOWIN and GOWIN are trademarks of Guangdong Gowin Semiconductor Corporation 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. No part of this document may be reproduced or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written consent of GOWINSEMI.

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
04/20/2023	1.0E	Initial version published.
06/30/2023	1.1E	JTAGSEL_N Net, JTAG Net, SSPI Net, MODE Net, and I2C Net constraints added.
08/18/2023	1.2E	<ul style="list-style-type: none">● The descriptions of JTAG Net, SSPI Net, MODE Net, and I2C Net constraints updated.● The MIPI_DPHY_RX primitive constraint example added.● Timing Paths function removed.

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	2
1.4 Support and Feedback	2
2 Physical Constraints Syntax Definition	3
2.1 I/O Location Constraints	3
2.2 I/O Attribute Constraints	4
2.3 Primitive Location Constraints	5
2.4 Group Constraints	8
2.4.1 Primitive Group Constraints	8
2.4.2 Relative Group Constraints	10
2.5 Resource Reservation Constraints	11
2.6 Vref Constraints	11
2.7 Clock Net Constraints	12
2.8 GCLK Primitive Constraints	14
2.9 HCLK Primitive Constraints	15
2.10 Other Constraints	15
2.10.1 ADC Input Voltage Source Constraint	15
2.10.2 JTAGSEL_N Net Constraints	16
2.10.3 JTAG Net Constraints	17
2.10.4 SSPI Net Constraints	17
2.10.5 MODE Net Constraints	19
2.10.6 I2C Net Constraints	20
3 FloorPlanner	21
3.1 Introduction	21

3.2 Start FloorPlanner	22
3.3 FloorPlanner Interface	23
3.3.1 Menu Bar	24
3.3.2 Summary and Netlist Windows	35
3.3.3 Package View	39
3.3.4 Chip Array Window	43
3.3.5 Constraints Editing Window	49
3.3.6 Message Window	54
4 Using FloorPlanner	55
4.1 Create Constraint File	55
4.2 Edit Constraints File	57
4.2.1 Constraints Examples	57
4.2.2 Edit I/O Constraints	59
4.2.3 Edit Primitive Constraints	60
4.2.4 Edit Group Constraints	61
4.2.5 Edit Resource Reservation Constraints	65
4.2.6 Edit Clock Net Constraints	66
4.2.7 Edit GCLK Primitive Constraints	67
4.2.8 Edit HCLK Primitive Constraints	68
4.2.9 Edit Vref Constraints	69

List of Figures

Figure 3-1 Start FloorPlanner via Menu Bar	22
Figure 3-2 Start FloorPlanner in Process View.....	22
Figure 3-3 Start FloorPlanner via Start Page.....	23
Figure 3-4 FloorPlanner Interface	24
Figure 3-5 File	25
Figure 3-6 Open Physical Constraints	25
Figure 3-7 Constraints	26
Figure 3-8 Select Primitive	26
Figure 3-9 New Primitive Group.....	27
Figure 3-10 Right Primitive Group	28
Figure 3-11 Invalid Locations	28
Figure 3-12 Invalid Locations	28
Figure 3-13 New Relative Group	29
Figure 3-14 Right Relative Group	29
Figure 3-15 Resource Reservation	30
Figure 3-16 Clock Assignment	31
Figure 3-17 GCLK Primitive Constraints.....	32
Figure 3-18 HCLK Primitive Constraints.....	33
Figure 3-19 Vref Constraints	33
Figure 3-20 Tools	33
Figure 3-21 Back-annotate Physical Constraints.....	34
Figure 3-22 Back-annotate Port.....	34
Figure 3-23 View Menu	35
Figure 3-24 Windows Menu	35
Figure 3-25 Summary Window	36
Figure 3-26 Netlist View	37
Figure 3-27 BUS and Non-Bus Display	38
Figure 3-28 Hierarchy Display	38
Figure 3-29 Netlist Right-clicking	39
Figure 3-30 Package View (GW5AT-138-FPBGA676A).....	40

Figure 3-31 Package View Right-clicking	41
Figure 3-32 Differential Pair Display	41
Figure 3-33 Top View	42
Figure 3-34 Bottom View	42
Figure 3-35 Chip Array Window	43
Figure 3-36 Constraints in Grid	44
Figure 3-37 Constraints in Macrocell	44
Figure 3-38 Constraints in Primitive	45
Figure 3-39 Chip Array Right-clicking	47
Figure 3-40 Show Place View	47
Figure 3-41 Mouse Hovering Display	48
Figure 3-42 Right-click to Select Highlight	49
Figure 3-43 I/O Constraints View	50
Figure 3-44 Primitive Constraints View	51
Figure 3-45 Group Constraints View	51
Figure 3-46 Resource Reservation View	52
Figure 3-47 Clock Assignment View	52
Figure 3-48 GCLK Primitive Constraints	53
Figure 3-49 HCLK Primitive Constraints	53
Figure 3-50 Vref Constraints View	54
Figure 3-51 Message Window	54
Figure 4-1 New Physical Constraints	55
Figure 4-2 Select Device	56
Figure 4-3 Save Output File	57
Figure 4-4 Drag to Chip Array to Create I/O Constraints	59
Figure 4-5 Drag to Package View to Create I/O Constraints	60
Figure 4-6 Drag to Chip Array to Create Primitive Constraints	61
Figure 4-7 Group Constraints Right-clicking	61
Figure 4-8 Create Primitive Group Constraints	62
Figure 4-9 Primitive Group Constraints	63
Figure 4-10 Create Relative Group Constraints	64
Figure 4-11 Relative Group Constraints	65
Figure 4-12 Create Resource Reservation	65
Figure 4-13 Resource Reservation	66
Figure 4-14 Create Clock Assignment Constraints	66
Figure 4-15 Clock Assignment Constraints	67
Figure 4-16 Create GCLK Primitive Constraints	67
Figure 4-17 GCLK Primitive Constraints	68

Figure 4-18 Create HCLK Primitive Constraints	68
Figure 4-19 HCLK Primitive Constraints	69
Figure 4-20 Create Vref Constraints	69
Figure 4-21 Drag to Chip Array to Generate Vref Constraints Location	70
Figure 4-22 Drag to Package View to Generate Vref Constraints Location	71
Figure 4-23 Duplicate Names Prompt	72

List of Tables

Table 1-1 Terminology and Abbreviations	2
Table 2-1 Constraint Positions for DCS /DCE.....	14

1 About This Guide

1.1 Purpose

This manual describes the interface and use of FloorPlanner and the syntax definition of physical constraint for Arora V FPGA products 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](#)
- [DS981, GW5AT series of FPGA Products Data Sheet](#)
- [DS1103, GW5A series of FPGA Products Data Sheet](#)
- [DS1104, GW5AST series of FPGA Products Data Sheet](#)
- [UG982, GW5AT-138 Pinout](#)
- [UG985, GW5A-25 Pinout](#)
- [UG986, GW5AST-138 Pinout](#)
- [UG306, Arora V Clock User Guide](#)
- [UG704, Arora V FPGA Product Programming and Configuration 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
BSRAM	Block SRAM
CFU	Configurable Function Unit
CLKDIV	Clock Divider
CLS	Configurable Logic Section
DCS	Dynamic Clock Selector
DDRDLL	Double Data Rate Delay-locked Loop
DLLDLY	DLL Delay
DQS	Bidirectional Data Strobe Circuit for DDR Memory
FloorPlanner	Physical Constraint Editor
FPGA	Field Programmable Gate Array
GCLK	Global Clock
I/O	Input/Output
IDE	Integrated Development Environment
LUT	Look-up Table
LW	Long Wire
PLL	Phase-locked Loop
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 Physical Constraints Syntax Definition

2.1 I/O Location Constraints

The IO constraints can constrain ports to the specified IO locations. For the details, you can see following manuals.

- [UG982, GW5AT-138 Pinout](#)
- [UG985, GW5A-25 Pinout](#)
- [UG986, GW5AST-138 Pinout](#)

Syntax

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

Constraint Elements

obj_name

Obj_name can be the name of port.

obj_location

obj_location is the IO 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 is constrained to the A1.

Example 2

```
IO_LOC "io_1" A1, B14, A15;
```

// io_1 is constrained to A1, B14, A15, first reasonable location of the three will be taken for the placement.

Example 3

```
IO_LOC "io_2" A1 exclusive;
```

// io_2 is constrained to the A1, and A1 can only be used by io_2.

Example 4

```
IO_LOC "io_2" A1, B14, A15 exclusive;
```

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

2.2 I/O Attribute Constraints

You can set attribute values for I/O, such as `IO_TYPE`, `PULL_MODE` and `DRIVE`, etc. You can see [DS981, GW5AT series of FPGA Products Data Sheet](#) for the details.

Syntax

```
IO_PORT " obj_name " attribute = attribute_value;
```

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

Constraint Elements

obj_name

Obj_name can be the name of port.

Attribute and Attribute Value

For the details of attribute and attribute value, you can see [UG304,](#)

Arora V Programmable IO (GPIO) User Guide.**Examples****Example 1**

```
IO_PORT "port_1" IO_TYPE = LVTTL33;
// Set port_1 IO_TYPE as "LVTTL33".
```

Example 2

```
IO_PORT "port_2" IO_TYPE = LVTTL33 PULL_MODE =KEEPER;
// Set port_2 IO_TYPE as the LVTTL33, PULL_MODE value is
"KEEPER".
```

2.3 Primitive Location Constraints

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

Syntax

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

Constraint Elements**obj_name**

The instance name

obj_location

obj_location includes:

1. LUT Constraint 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 CLSs or LUTs: "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!

- x, m in the LUT location refers to the row information of the GRID.
- y, n in the LUT location refers to the column information of the GRID.
- R in the LUT location refers to the row of the GRID and C refers to the column GRID.
- 0-3 in the LUT location refers to the number of a CLS in a GRID.
- A-B in the LUT location refers to the number of a specific LUT location in a CLS.

2. PLL Constraint Location

For the "PLL_L" or "PLL_R" of the PLL constraint 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]" ...

3. BSRAM Constraint Location

The BSRAM constraint location is "BSRAM_R10[0]" (the first BSRAM at row 10), "BSRAM_R10[1]"

4. DSP Constraints Location

The DSP constraint location is "DSP_R37[0]" (the first DSP Block at row 37), "DSP_R37[1]" ... If MULT12X12 is specified, it can be marked as: DSP_R37[0][A] or DSP_R37[0][B].

Note!

A DSP Block location includes two macros; a macro refers to a location where the MULT12X12 can be placed.

5. DDRDLL Constraint Location

DDRDLL constraint location is "DDRDLLM_TL", "DDRDLLM_TR".

6. MIPI_DPHY_RX Constraint Location

MIPI_DPHY_RX constraint location is "QUAD[0]", "QUAD [1]".

Note!

MIPI_DPHY_RX constraints support device: GW5A(S)(T)-138, but does not support exclusive

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.

Note!

Multiple obj_locations in a single constraint statement can be separated by ",".

Examples

Example 1

```
INS_LOC "lut_1" R5C10[0][A];
```

// lut_1 is constrained at the 1st LUT of the 1st CLS of R5C10.

Example 2

```
INS_LOC "ins_2 " R5C6[2] exclusive;
```

// ins_2 is constrained at the 3rd CLS of R5C6, and only this primitive can be placed in this location.

Example 3

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

// ins_3 is constrained at the range between 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 at the range between the second to fourth rows and the second to sixth columns, and this range can only be used by this primitive.

Example 5

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

// ins_5 is constrained at 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;
```

// reg_name is constrained to the IO B14.

Example 7

```
INS_LOC "pll_name" PLL_L[0];
```

// Constrain pll_name to the first position of the left PLL by means of an INS_LOC constraint on the PLL.

Example 8

```
INS_LOC "bsram_name" BSRAM_R10[2];
```


// Constrain bsram_name to the third BSRAM in row 10 by means of an INS_LOC constraint on the BSRAM.

Example 9

```
INS_LOC "dsp_name" DSP_R19[2];
```

// Constrain dsp_name to the 3rd DSP Block in row 19 by means of an INS_LOC constraint on the DSP.

Example 10

```
INS_LOC "ddrdll_name" DDRDLLM_TL;
```

// Constrain ddrdll_name to the top left DDRDLL by means of an INS_LOC constraint on the DDRDLL.

Example 11

```
INS_LOC "mipi_dphy_rx_name" QUAD[0];
```

// Constrain mipi_dphy_rx_name to the first MIPI_DPHY_RX location by means of an INS_LOC constraint on the MIPI_DPHY_RX.

2.4 Group Constraints

The group constraints include Primitive Group Constraints and Relative Group Constraints.

2.4.1 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, DDRDLL, 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

GROUP Definition:

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

Add the instance to the group:

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

The location of the constrained group:

```
GRP_LOC group_name group_location[exclusive];
```

Note!

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

Constraint 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 constraint location of the group, and the group_location can be constrained at the IOB, GRID, BSRAM, DSP, PLL, DDRDLL.

exclusive

The keyword "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 used 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 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 this range.
```

Example 6

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

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

2.4.2 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;
```

Constraint 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.

2.5 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.

2.6 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];"
```

Constraint Elements

vref_name

Customized VREF pin name

location

Any IO location with IOLOGIC in the GRID 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 = SSTL18_I VREF=vref_pin;
IO_PORT "port_2" IO_TYPE = SSTL18_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 = SSTL18_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 constrain it to C16; the constraint location
of port_1 needs to be in the same bank as E16.
```

2.7 Clock Net Constraints

The Clock Net Constraints is used to constrain a specific net to the global clock wire or the non-clock wire in the design.

- BUFG[0-15] means this net is constrained to the global clock wire.
- 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;
```

Constraint Elements

net_name

The net name

global_clocks

BUFG[0-15] means this net is routed to a specific global clock wire.

BUFG means this net is routed to the global clock wire.

LOCAL_CLOCK means this net is not routed to the global 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, CLEAR, PRESET pins.

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

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

Note!

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

Examples

Example 1

```
CLOCK_LOC "net" BUFG = CLK|CE;
```

```
NET_LOC "net" BUFG = CLK|CE;
```

// Constrain the clock net whose signal_type is a clock pin or a clock enable pin to the global clock wire.

Example 2

```
CLOCK_LOC "net" LOCAL_CLOCK;
```

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

Example 3

```
CLOCK_LOC "net" BUFG[0] = CLK;
```

```
// Constrain the clock net whose signal_type is a clock pin to the first
global clock wire.
```

2.8 GCLK Primitive Constraints

GCLK Primitive Constraints are used to constrain the DCS, DCE to the specified locations.

Syntax

```
INS_LOC "obj_name" position;
```

Constraint Element

obj_name

The name of a constraint object.

position

Table 2-1 Constraint Positions for DCS/DCE

Device	Position	
	DCE	DCS
GW5AT-138	PTR0, PTR1, PTR2 PTR3, PTR0[0~5], PTR1[0~5], PTR20~5], PTR3[0~5], PBR0, PBR1, PBR2, PBR3, PBR0[0~5], PBR1[0~5], PBR2[0~5], PBR3[0~5], PG, SG, PG[0~11], SG[0~15]	PTR0, PTR1, PTR2, PTR3, PTR0[0~1], PTR1[0~1], PTR2[0~1], PTR3[0~1], PBR0, PBR1, PBR2, PBR3, PBR0[0~1], PBR1[0~1], PBR2[0~1], PBR3[0~1], PG, PG[0~3]
GW5A-25	TOPLEFT, TOPRIGHT, BOTTOMLEFT, BOTTOMRIGHT, STOP, SBOTTOM	TOPLEFT, TOPRIGHT, BOTTOMLEFT, BOTTOMRIGHT

Note!

The specific location of the constraint position for DCS/DCE, see [UG306, Arora V Clock User Guide](#).

Example

```
INS_LOC "dcs_name" PTR0[1].
```

```
// Constrain the DCS object dcs_name to the PTR0[1] position.
```

2.9 HCLK Primitive Constraints

The CLKDIV/DLLDLY can be constrained to the Hclk locations using the HCLK Primitive Constraints. The constraints locations of CLKDIV/DLLDLY are different from those of other general instances. The "BOTTOMSIDE", "LEFTSIDE", and "RIGHTSIDE" indicate the sides of the constraints location.

Syntax

```
INS_LOC "obj_name" position;
```

Constraints Elements

obj_name

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

location

Constraint positions for CLKDIV (GW5AT-138): BOTTOMSIDE[0~7], LEFTSIDE[0~7], RIGHTSIDE[0~7]

Constraint positions for CLKDIV (GW5A-25): LEFTSIDE[0~3], RIGHTSIDE[0~3], BOTTOMSIDE[0~3], TOPSIDE[0~3]

Constraint positions for DLLDLY (GW5AT-138): BOTTOMSIDE[0~3], LEFTSIDE[0~3], RIGHTSIDE[0~3]

Constraint positions for DLLDLY (GW5A-25): BOTTOMSIDE[0~1], LEFTSIDE[0~1], RIGHTSIDE[0~1], TOPSIDE[0~1]

Example

```
INS_LOC "clkdiv_name" LEFTSIDE[0];  
  
// Place the clkdiv_name to LEFTSIDE[0].
```

2.10 Other Constraints

2.10.1 ADC Input Voltage Source Constraint

The ADC input voltage source can come from external IO, and the IO location in bank0/2/3/4/5/6/7 can be used to specify the ADC input voltage source through the constraints.

The ADC input voltage constraint syntax includes bus0 and bus1, and the bus0 corresponds to the IO location in bank0/6/7 and bus1 corresponds to the IO location in bank2/3/4/5.

Note!

GW5A-25 devices support this constraint.

Syntax

USE_ADC_SRC bus0 location

USE_ADC_SRC bus1 location

Constraint Element**location**

location is the IO location, which is only supported in the form of IOB constraints, such as IOT48.

Note!

IOL25/IOL31/IOB45/IOR31/IOR5/IOT43/IOT3 IO locations cannot be used.

Example

USE_ADC_SRC bus0 IOT48

// Use IOT48 as the external input for the ADC input voltage source.

USE_ADC_SRC bus1 IOR24

//Use IOR24 as the external input for the ADC input voltage source.

2.10.2 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_JTAGSELN;

Constraints Element**obj_name**

The net 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.

2.10.3 JTAG Net Constraints

The JTAG function pins include TCK, TMS, TDI, TDO, which can be realized through IO configuration or FPGA internal logic control, and these two methods are mutually exclusive; these function pins can only be realized in the same method, i.e., selecting the IO configuration to realize these pin functions or selecting the FPGA internal logic control to realize them; in addition, TDO can only be realized through IO configuration and is not affected by the implementation mode of TCK, TMS and TDI.

When the internal logic of the FPGA is used to control TCK, TMS, and TDI of JTAG, the net physical constraints of JTAG need to be added, For the details, you can refer to [UG704, Arora V FPGA Product Programming and Configuration Guide](#).

Syntax

```
NET_LOC " obj_Name" V_TCK;
```

```
NET_LOC " obj_Name" V_TMS;
```

```
NET_LOC " obj_Name" V_TDI;
```

Constraints Element

obj_name

The net can be wired in the internal logic can be used as obj_name

Example

```
NET_LOC "netname" V_TCK;
```

```
// The netname net is used to control TCK functions.
```

```
NET_LOC " netname " V_TMS;
```

```
// The netname net is used to control TMS functions.
```

```
NET_LOC " netname " V_TDI;
```

```
// The netname net is used to control TDI functions.
```

2.10.4 SSPI Net Constraints

SSPI pins include SI, SO, SSPI_WPN, CLKHOLD_N, SSPI_CLK, and SSPI_CS_N, which can be realized through IO configuration or FPGA internal logic control, and these two methods are mutually exclusive; these

function pins can only be realized in the same method, i.e., selecting the IO configuration to realize these pin functions or selecting the FPGA internal logic control to realize them.

When the internal logic of the FPGA is used to control SSPI functions, the net physical constraints of SSPI need to be added, For the details, you can see [UG704, Arora V FPGA Product Programming and Configuration Guide](#).

Syntax

```
NET_LOC " obj_Name" V_SSPI SI;
NET_LOC " obj_Name" V_SSPI SO;
NET_LOC " obj_Name" V_SSPI WPN;
NET_LOC " obj_Name" V_SSPI CLKHOLDN;
NET_LOC " obj_Name" V_SSPI CLK;
NET_LOC " obj_Name" V_SSPI CSN;
```

Constraints Element

obj_name

The net can be wired in internal logic can be used as obj_name

Examples

```
NET_LOC "netname" V_SSPI SI;
// The netname net is used to control SSPI SI functions.
NET_LOC " netname " V_SSPI SO;
// The netname net is used to control SSPI SO functions.
NET_LOC " netname " V_SSPI WPN;
// The netname net is used to control SSPI WPN functions.
NET_LOC "netname" V_SSPI CLKHOLDN;
// The netname net is used to control CLKHOLD_N functions.
NET_LOC " netname " V_SSPI CLK;
// The netname net is used to control SSPI CLK functions.
NET_LOC " netname " V_SSPI CSN;
// The netname net is used to control SSPI_CS_N functions.
```

2.10.5 MODE Net Constraints

MODE pins include MODE0, MODE1, MODE2, which can be realized through IO configuration or FPGA internal logic control, and these two methods are mutually exclusive; these function pins can only be realized in the same method, i.e., selecting the IO configuration to realize these pin functions or selecting the FPGA internal logic control to realize them.

When the FPGA internal logic is used to control the function of MODE, in addition to MODE0, MODE1 and MODE2, the MODE_LD signal is added, which is a loading signal that loads the value of MODE0~MODE2 controlled by the internal logic on the rising edge, i.e., switching the value of MODE0~MODE2 when MODE_LD is at a rising edge; prior to the first rising edge of MODE_LD, the values set for MODE0 to MODE2 during bitstream download are used. When the internal logic of the FPGA is used to control MODE functions, the net physical constraints of MODE need to be added, For the details, you can refer to [UG704, Arora V FPGA Product Programming and Configuration Guide](#).

Syntax

```
NET_LOC " obj_Name" V_MODE0;
NET_LOC " obj_Name" V_MODE1;
NET_LOC " obj_Name" V_MODE2;
NET_LOC "obj_Name" V_MODE_LD;
```

Constraints Element

obj_name

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

Examples

```
NET_LOC " netname" V_MODE0;
// The netname net is used to control MODE0.

NET_LOC " netname" V_MODE1;
// The netname net is used to control MODE1.

NET_LOC " netname" V_MODE2;
// The netname net is used to control MODE2.
```

```
NET_LOC " netname" V_MODE_LD;  
// The netname net is used to control MODE_LD.
```

2.10.6 I2C Net Constraints

The I2C pins include SCL and SDA, which can be realized through IO configuration or FPGA internal logic control, and these two methods are mutually exclusive; these function pins can only be realized in the same method, i.e., selecting the IO configuration to realize these pin functions or selecting the FPGA internal logic control to realize them.

When the internal logic of the FPGA is used to control I2C functions, the net physical constraints of I2C need to be added.

Syntax

```
NET_LOC " obj_Name" V_SCL;  
NET_LOC " obj_Name" V_SDA;
```

Constraints Element

obj_name

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

Examples

```
NET_LOC " netname" V_SCL;  
// The netname net is used to control SCL functions.  
NET_LOC " netname" V_SDA;  
// The netname net is used to control SDA functions.
```

Note!

I2C net constraint is supported by the device GW5A-25.

3 FloorPlanner

3.1 Introduction

FloorPlanner is a physical constraint 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.

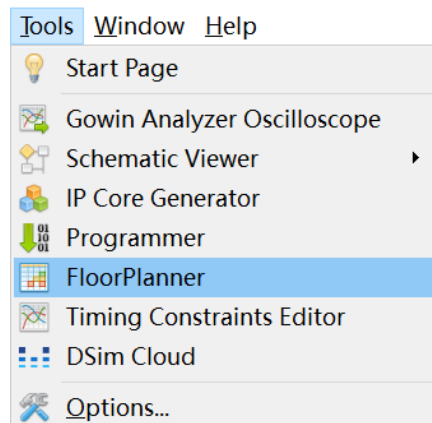
- Supports the input of user design files & constraint files, editing constraint files, and the output of constraint files
- Supports the display of I/O Port, Primitive and Group constraints in user design files
- Can create, edit and modify constraints files
- Supports grid mode, macro cell mode and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

3.2 Start FloorPlanner

There are three methods to start FloorPlanner:

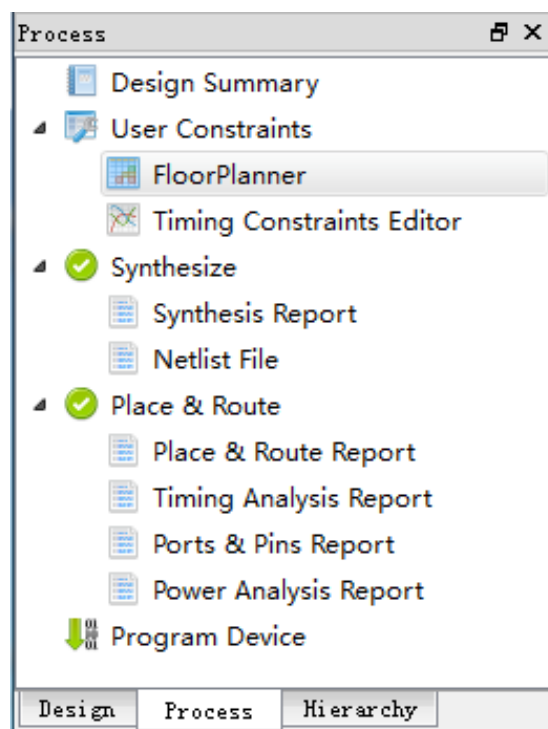
1. Click "IDE > Tools" to open "FloorPlanner", as shown in Figure 3-1.

Figure 3-1 Start FloorPlanner via Menu Bar



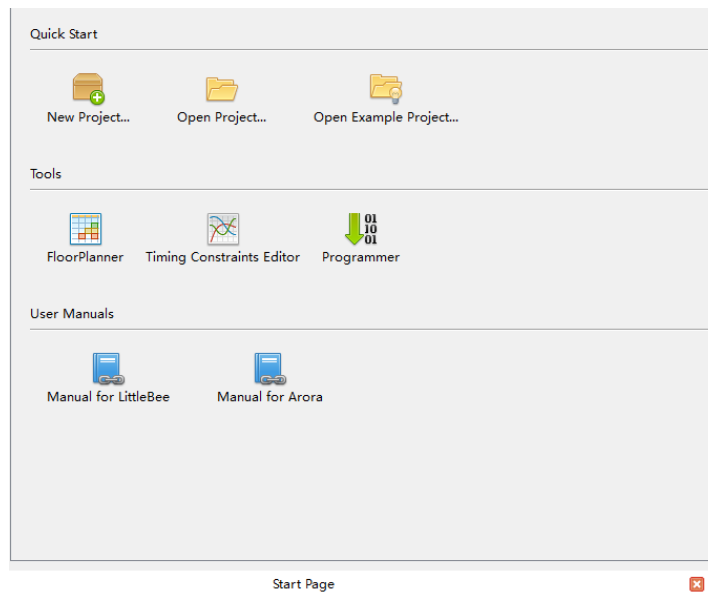
2. After synthesis, double-click "FloorPlanner" in Process view, as shown in Figure 3-2.

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!

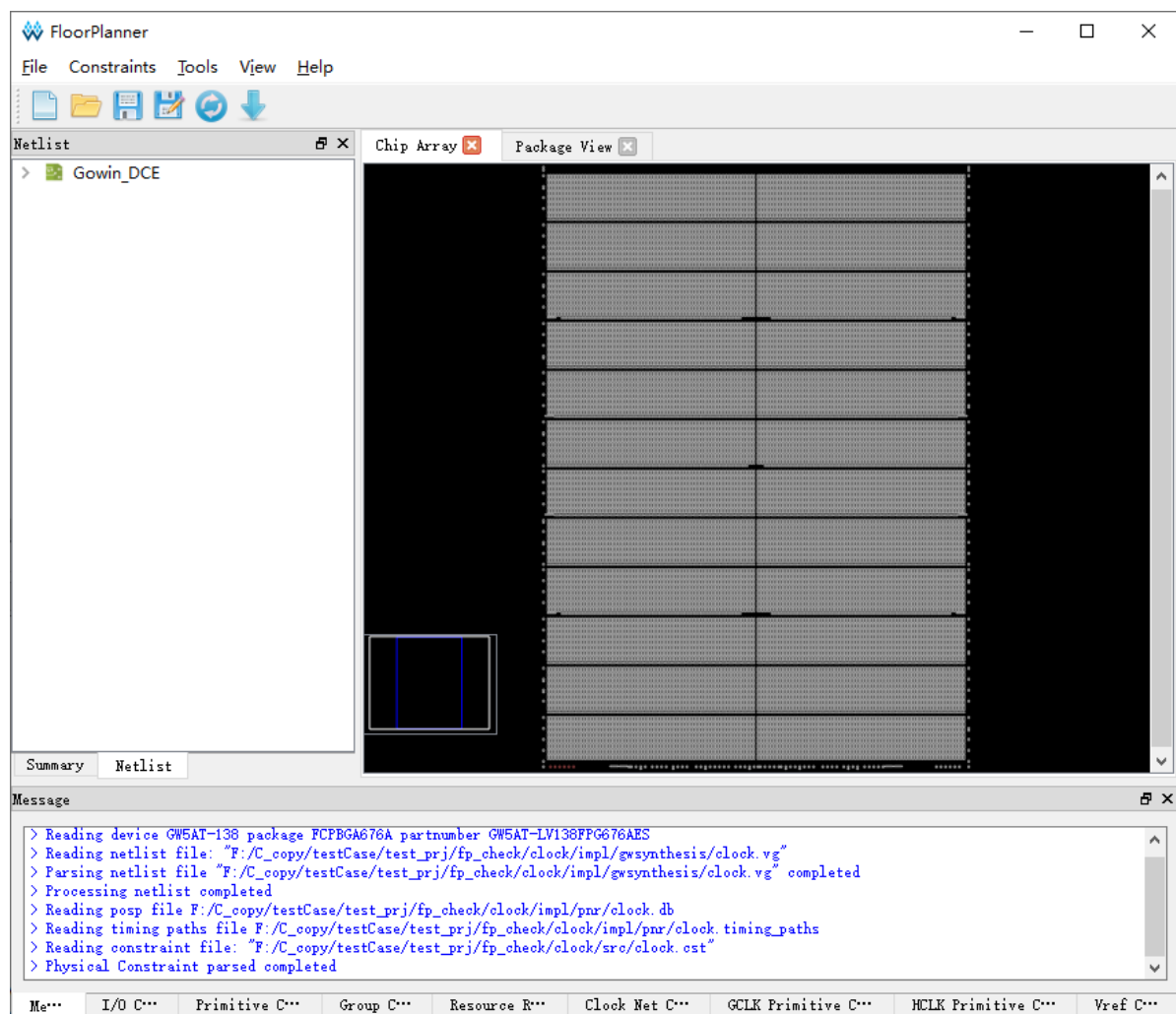
- If you use Gowin FloorPlanner for constraints, the netlist file should be added first.
- When you choose the first or the second method to start Gowin FloorPlanner, the netlist file will be loaded automatically.
- When you choose the third method to start Gowin FloorPlanner, the netlist file is needed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

3.3 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



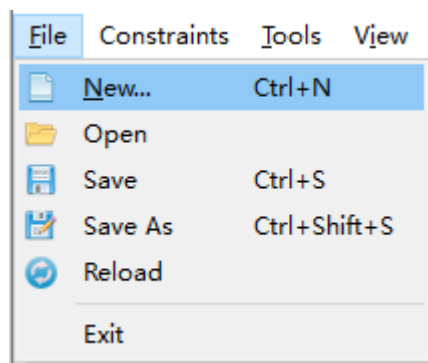
3.3.1 Menu Bar

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

File Menu

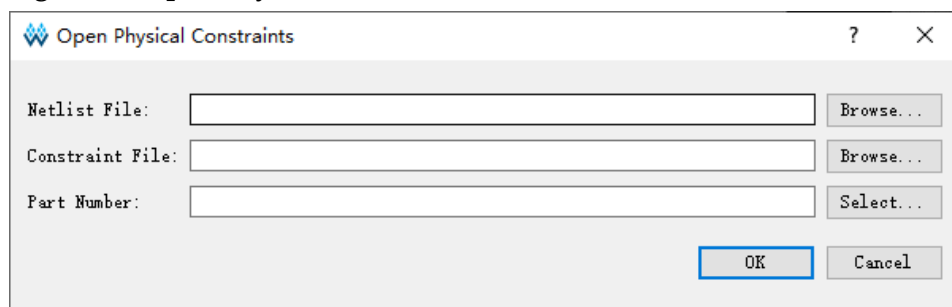
File menu is shown in Figure 3-5.

Figure 3-5 File



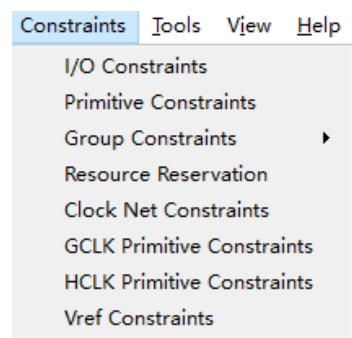
- New: Create constraints, add user design and select device, etc.
- Open: Open to add constraints, select part number, as shown in Figure 3-6.
- Reload: Physical constraint files, and place files can be reloaded after modifying.
- Save: Save the modified files.
- Save As: Save the modified file to a specified file and use the netlist name as the name of constraint file, which can be modified.
- Exit: Exit FloorPlanner.

Figure 3-6 Open Physical Constraints



Constraints Menu

Constraints menu is as shown in Figure 3-7.

Figure 3-7 Constraints

Primitive Constraints

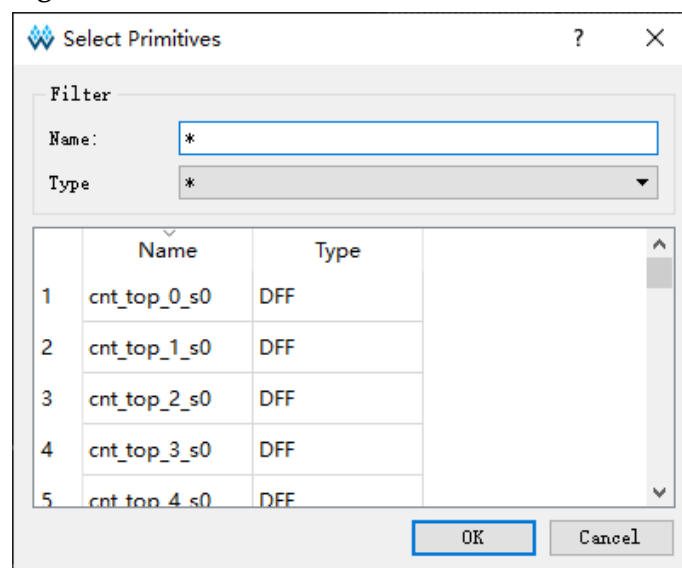
Right-click to select "Select Primitives" and a dialog box pops up, as shown in Figure 3-8.

You can select primitives by name or type.

1. Click "OK" to generate the constraints and the constraints are displayed in "Primitive Constraints" at the bottom of main interface.
2. 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 Select Primitive

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

The screenshot shows a dialog box titled "New Primitive Group". It has a standard Windows-style title bar with a question mark icon and a close button (X). The main content area is divided into several sections. At the top is a text input field labeled "Group Name:". Below this is a section titled "Members" which contains a table with two columns, "Name" and "Type". Underneath the table are two buttons, a green "+" and a red "-", and a checkbox labeled "Exclusive". Below the "Members" section is another section titled "Locations" which contains a large, empty text area for input. To the right of this text area is another checkbox labeled "Exclusive". At the bottom of the dialog are two buttons: "OK" and "Cancel".

Figure 3-10 Right Primitive Group

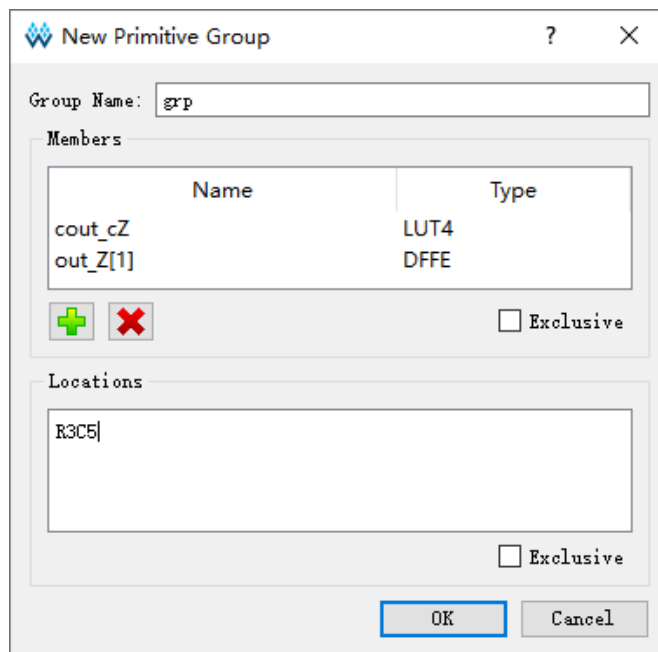


Figure 3-11 Invalid Locations

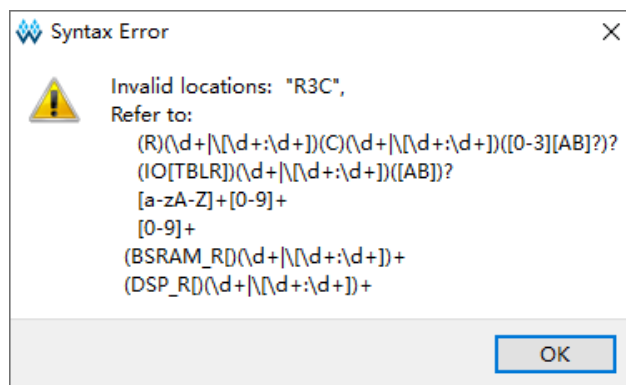
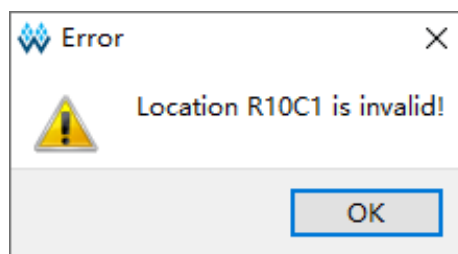


Figure 3-12 Invalid Locations



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 primitives, 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

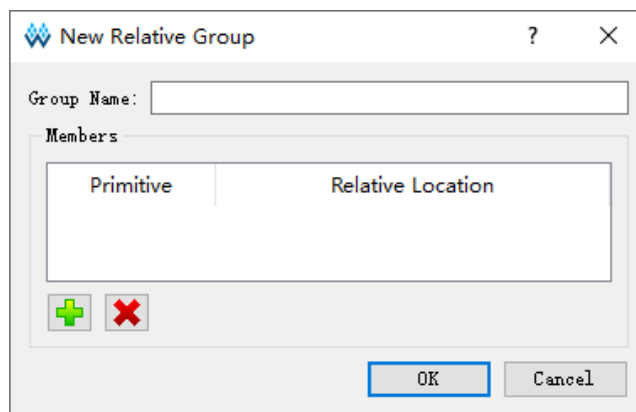
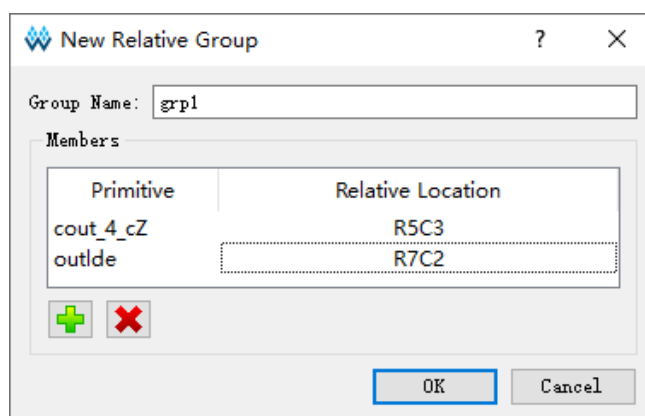


Figure 3-14 Right Relative Group



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 set the locations by typing or 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

	Name	Locations	Attribute
1	reserve_0	drag or type t...	ALL
			ALL
			LUT
			REG

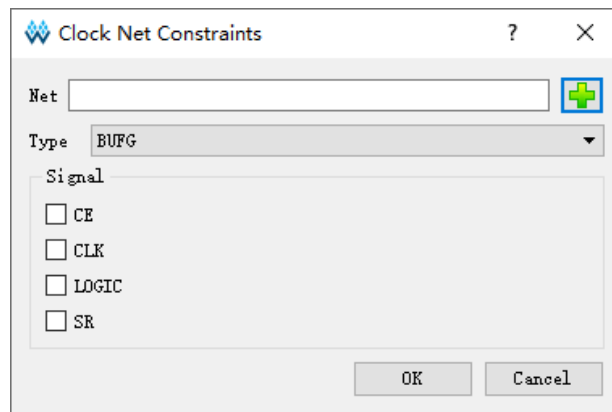
Clock Net Constraints

Create global clock constraints and the number of constraints is limited; and 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]~[15]", and "LOCAL_CLOCK" in "Type" drop-down list.
3. Configure Signal by "CE" and "CLK" check box. After finished, click "OK" to generate constraints in "Clock Net Constraints". Double-click to open the dialog box for editing.

Note!

When LOCAL_CLOCK selected, the signal check box is grayed.

Figure 3-16 Clock Assignment

GCLK Primitive Constraints

GCLK Primitive Constraints is used to create global clock constraints for the global clock primitives, which constrains the specified Instance to a specific global clock according to the global clock distribution of the device; use GW5AT-138 device as an example, right-click on the "GCLK Primitive Constraints" constraint editing window at the bottom of the main interface and select the "GCLK Primitive Constraints", the dialog box shown in Figure 3-17 pops up. The related operations are shown as follows:


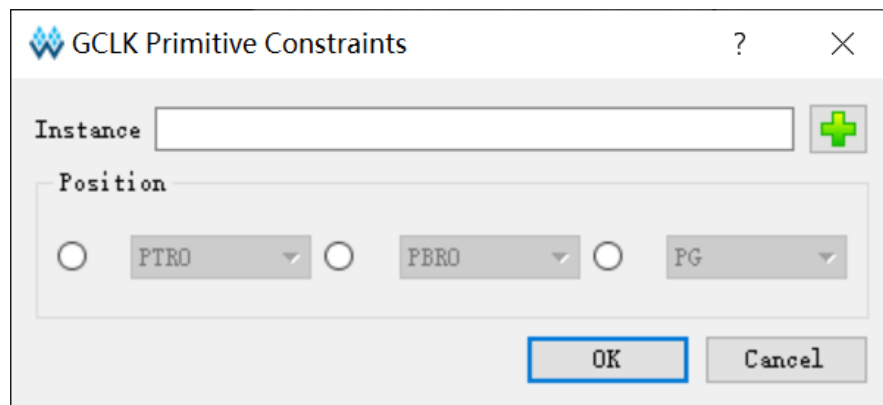
1. Select the corresponding GCLK primitive by clicking the " " button, if there is no GCLK primitive in the design, you cannot add.
2. Configure the global clock position using "Position" and the drop-down list.
3. Click "OK" to generate the constraints, which are displayed in the "GCLK Primitive Constraints"; double click the editing window to reopen the constraint dialog box, and you can edit and modify the constraints.

Figure 3-17 GCLK Primitive Constraints

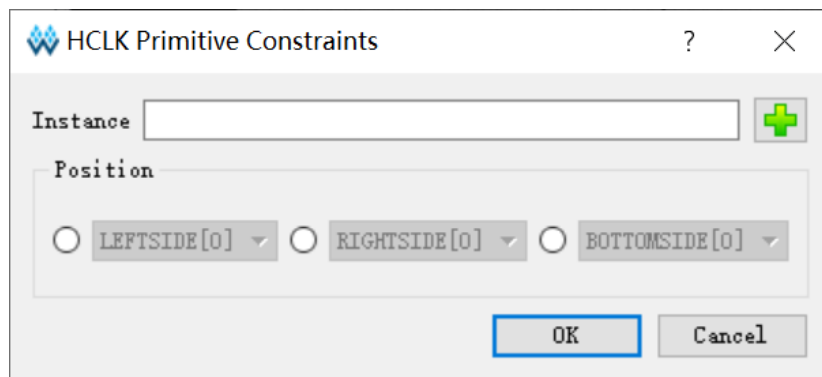
**Note!**

- When "Instance" is selected, "Position" is highlighted.
- Position varies depending on the device, and the position available for different global clock primitives will also vary.

HCLK Primitive Constraints

HCLK Primitive Constraints is used to create constraints for the HCLK primitives, and specify the constraints to high-speed clock locations; use GW5AT-138 device as an example, right-click on the "HCLK Primitive Constraints" constraint editing window at the bottom of the main interface and select "Select HCLK Primitive", the dialog box shown in Figure 3-18 pops up. The related operations are shown as follows:

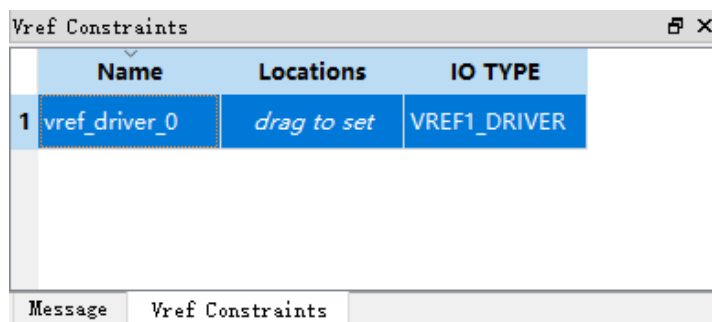
1. Select the corresponding HCLK primitive by clicking the "+" button, if there is no GCLK primitive in the design, you cannot add.
2. Configure the high-speed clock position using "Position" and the drop-down list.
3. Click "OK" to generate the constraints, which are displayed in the "HCLK Primitive Constraints"; double click the editing window to reopen the constraint dialog box, and you can edit and modify the constraints.

Figure 3-18 HCLK Primitive Constraints**Note!**

- When "Instance" is selected, "Position" is highlighted.
- Position varies depending on the device, and the position available for different global clock primitives will also vary.

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 3-19.

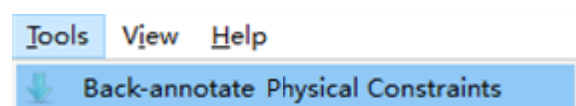
Figure 3-19 Vref Constraints**Note!**

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

Tools Menu

Tools menu is shown in Figure 3-20.

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

Figure 3-20 Tools

1. Click "Tools > Back-annotate Physical Constraints" to open a dialog box, as shown in Figure 3-21. 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 3-22, it is the generated physical constraints file when Port and Port Attribute selected in Back-annotate Physical Constraints.

Figure 3-21 Back-annotate Physical Constraints

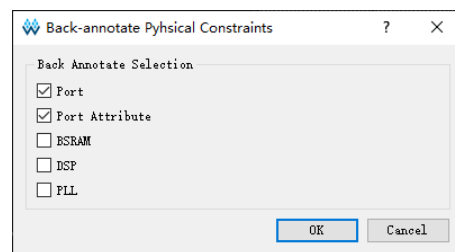
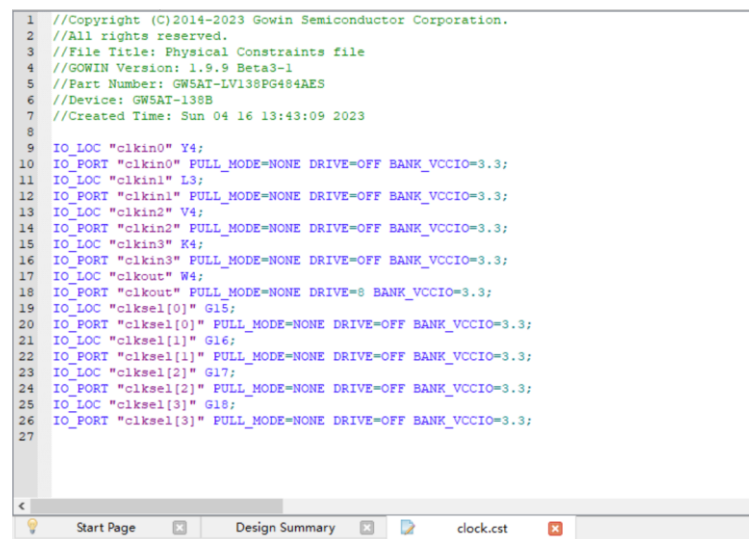


Figure 3-22 Back-annotate Port



View Menu

As shown in Figure 3-23, View includes Toolbars, Windows, Zoom In, Zoom Out and Zoom Fit. The descriptions of these sub-menus are as follows.

- Toolbars: Display shortcuts

- Windows: Display different windows, as shown in Figure 3-24
- 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 3-23 View Menu

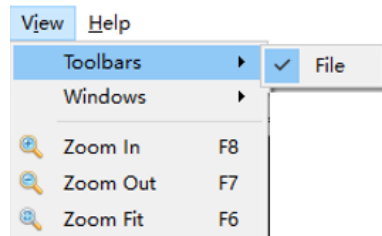
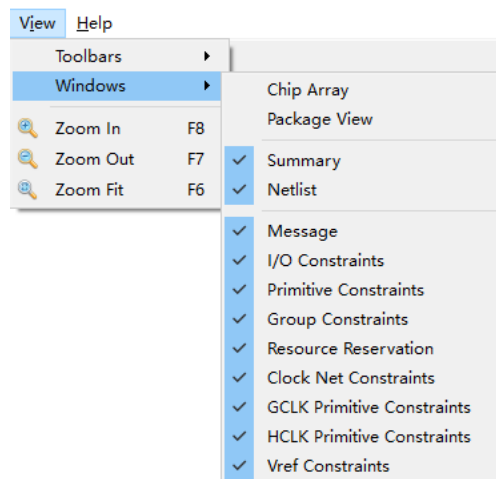


Figure 3-24 Windows Menu



Help Menu

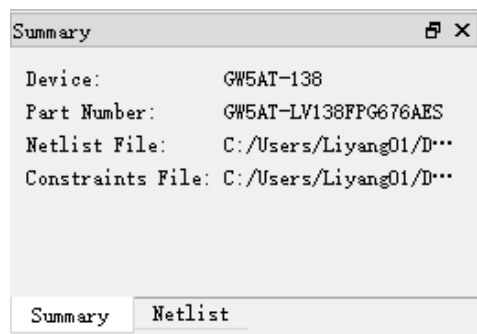
Help is used to provide software version and copyright information.

3.3.2 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 3-25. It displays the information of device, part number, design files and constraints files.

Figure 3-25 Summary Window

Netlist Window

As shown in Figure 3-26, 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 3-27.
- Modules are displayed in hierarchy and the number of instances in each module is also displayed, as shown in Figure 3-28.

Figure 3-26 Netlist View

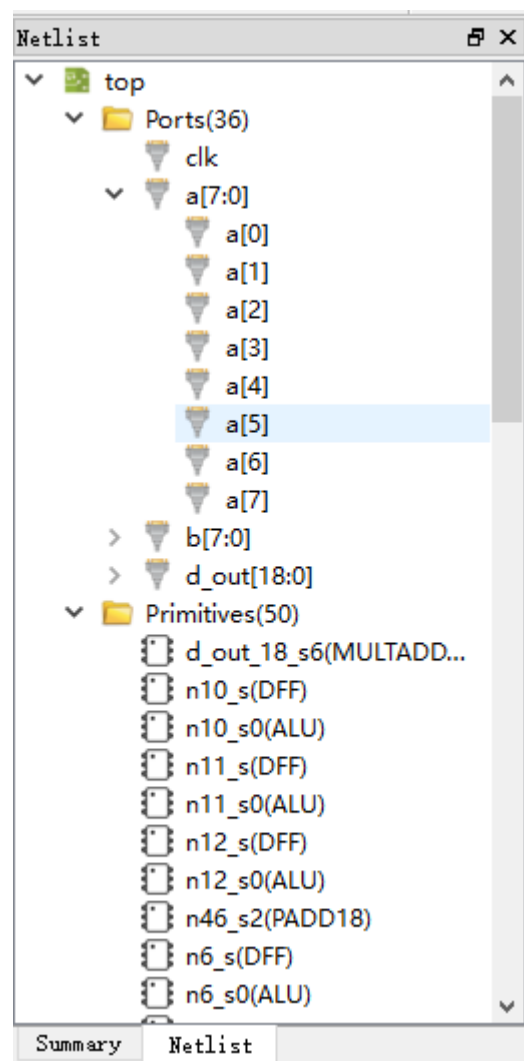


Figure 3-27 BUS and Non-Bus Display

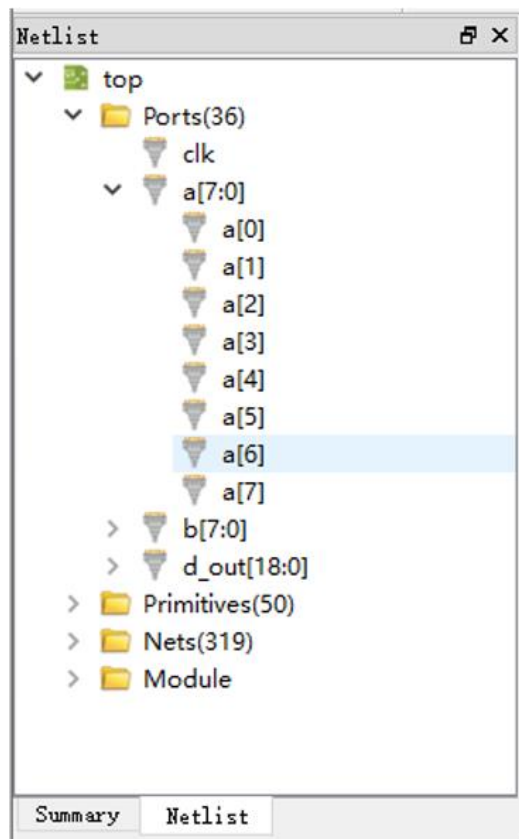
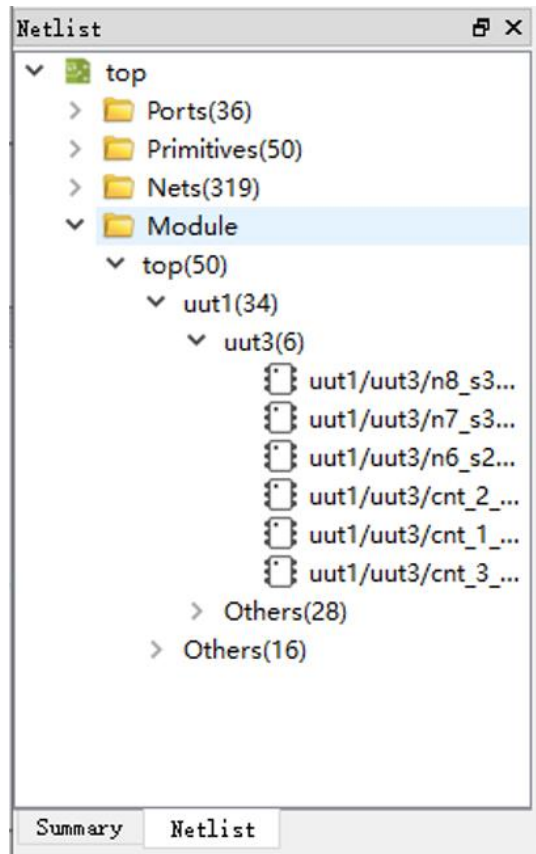


Figure 3-28 Hierarchy Display



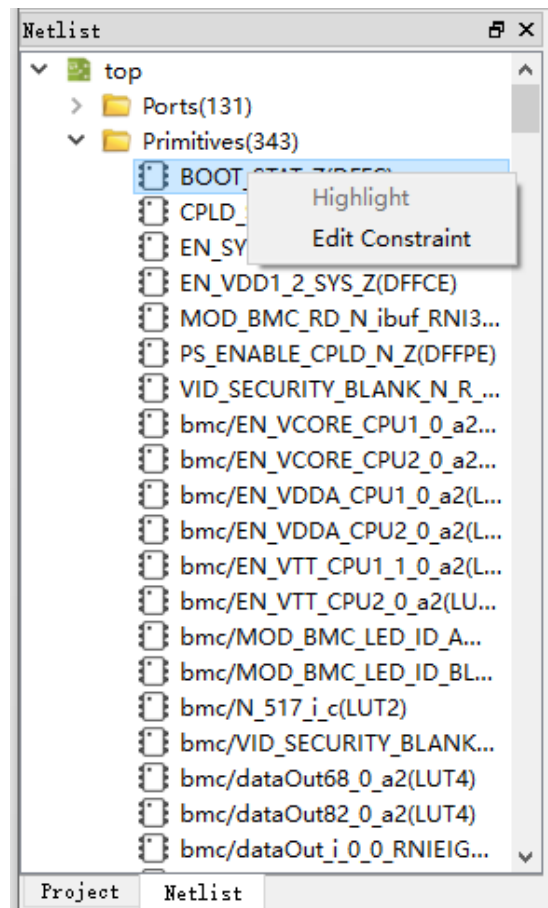
Netlist provides right-click menu as shown below.

- 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 3-29.

Figure 3-29 Netlist Right-clicking



3.3.3 Package View

As shown in Figure 3-30 , taking GW5AT-138-FCPBGA676A as an example, Package View displays I/O, power, and ground pins based on chip package. When the mouse is placed on a location, the I/O type, bank, and LVDS will display.

Figure 3-30 Package View (GW5AT-138-FPBGA676A)



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

- "⤴": User I/O
- "⤴": VCCIO
- "⤴": VSS

The right-click menu supported by the Package View is shown in Figure 3-31. 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 3-32, 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 GW5AT-138-FCPBGA676A with coordinate origin at the

top left corner is as shown in Figure 3-33.

- Bottom View: The bottom view of GW5AT-138-FCPBGA676 with coordinate origin at the top right corner is as shown in Figure 3-34.

Figure 3-31 Package View Right-clicking

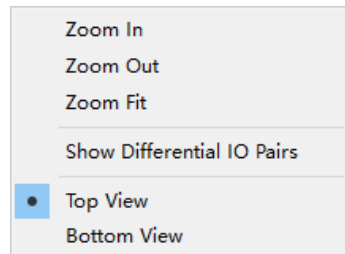


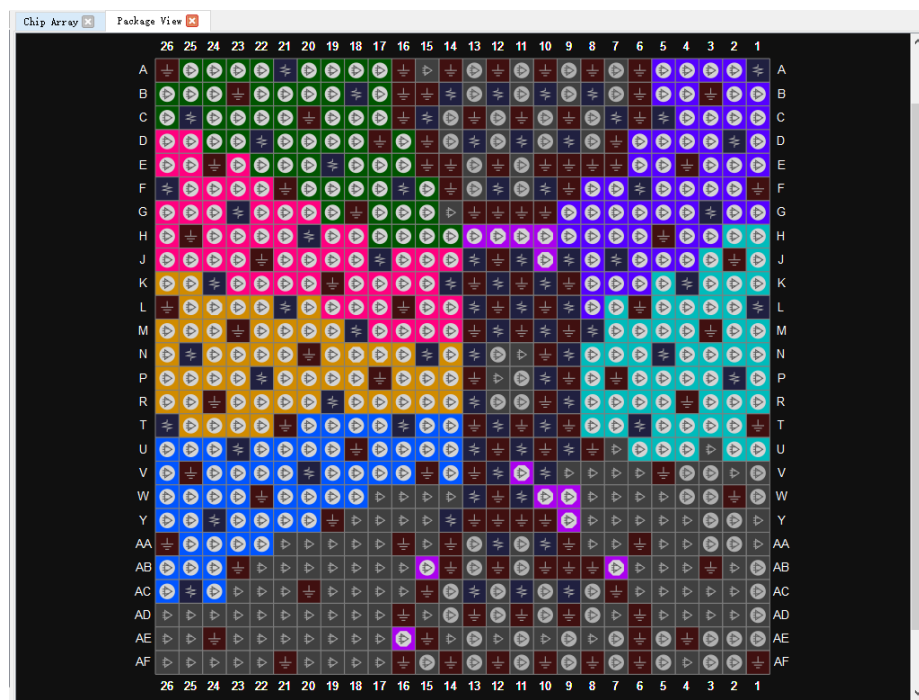
Figure 3-32 Differential Pair Display



Figure 3-33 Top View



Figure 3-34 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.

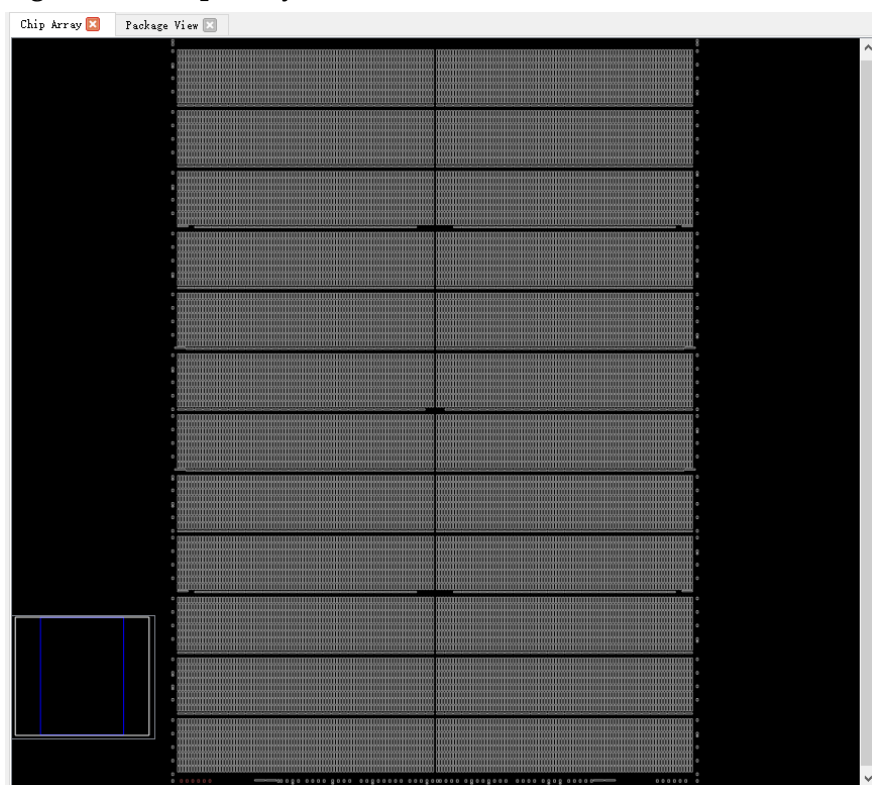
3.3.4 Chip Array Window

The Chip Array window of FloorPlanner is shown in Figure 3-35. 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 I/O location
- Red: Unbonded I/O location

Figure 3-35 Chip Array Window



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

- Grid mode: Display constraint locations in grid, as shown in Figure 3-36.
- Macrocell mode: Display constraint locations in CLS, IOBLK, as shown in Figure 3-37.
- Primitive mode: Display constraint locations in REG, LUT etc., as shown in Figure 3-38.

Figure 3-36 Constraints in Grid

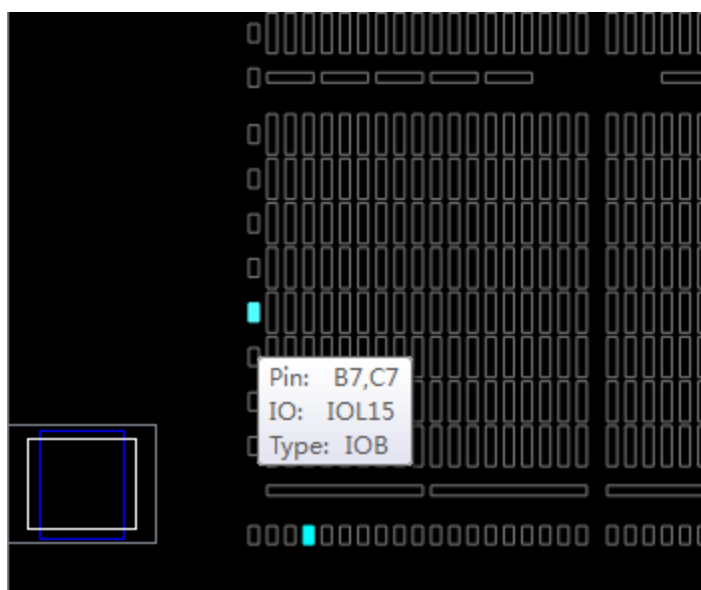


Figure 3-37 Constraints in Macrocell

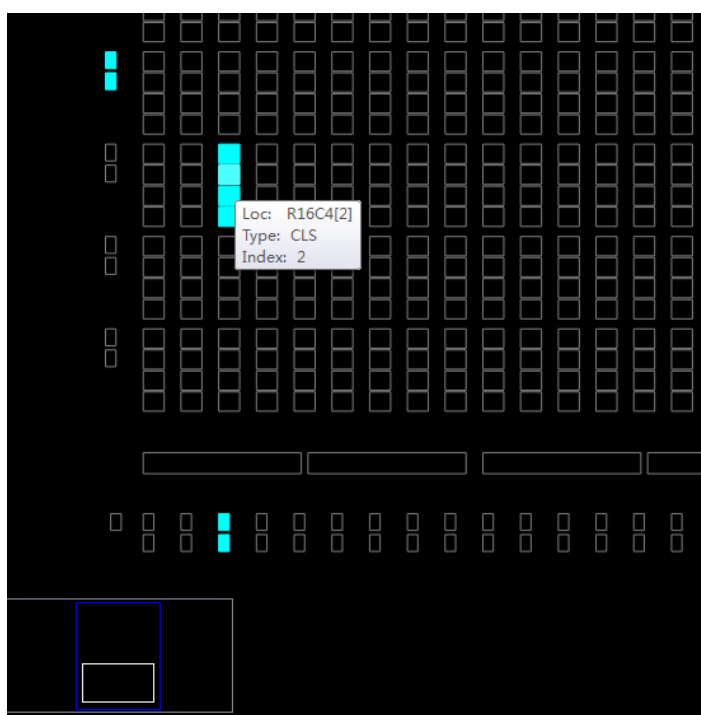
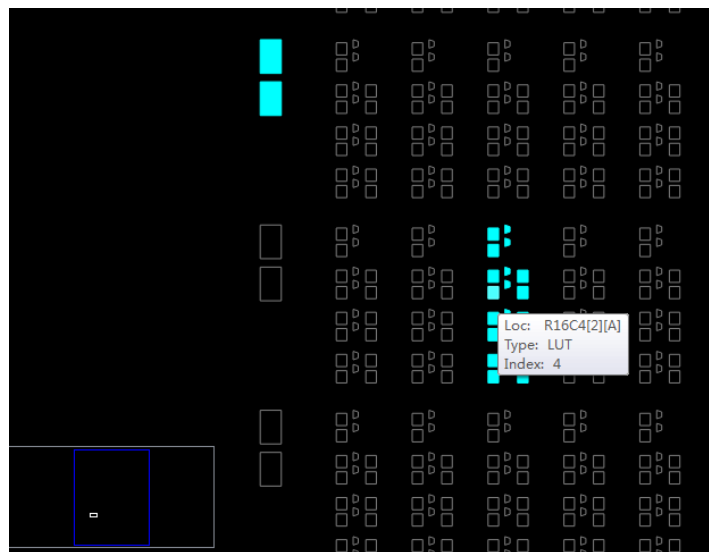


Figure 3-38 Constraints in Primitive



Chip Array supports following dragging functions.

- Drag from Netlist to Array to generate constraints and specify constraint location.
- Drag from Editing to Array to specify constraint 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 3-39.

Show Place View also shows Lut and Reg density, as shown in Figure 3-40.

- 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 3-41.
- 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 3-42.

Note!

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

Figure 3-39 Chip Array Right-clicking

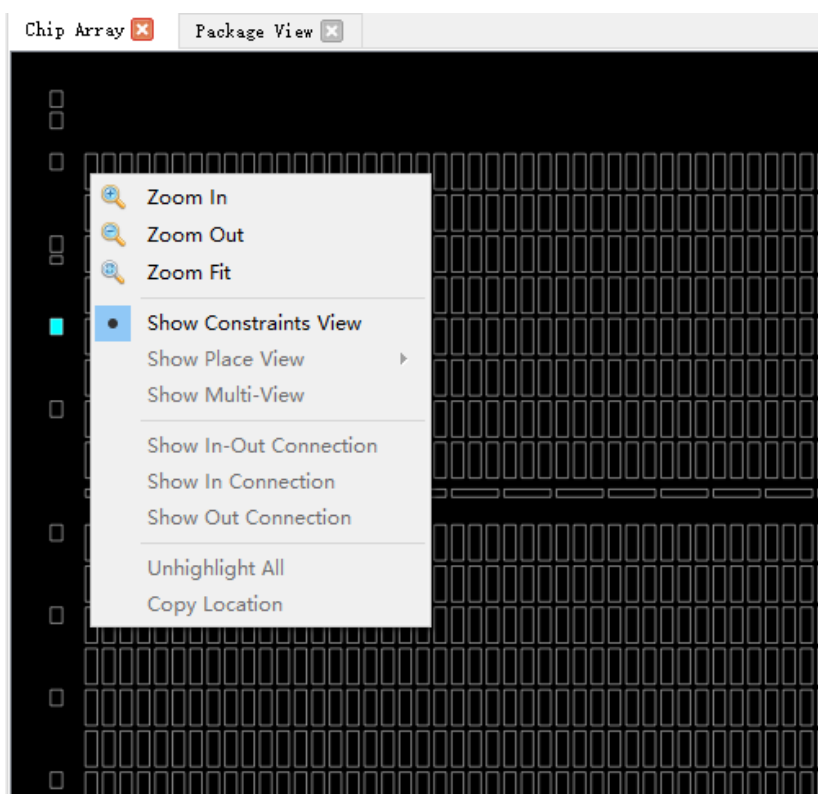


Figure 3-40 Show Place View

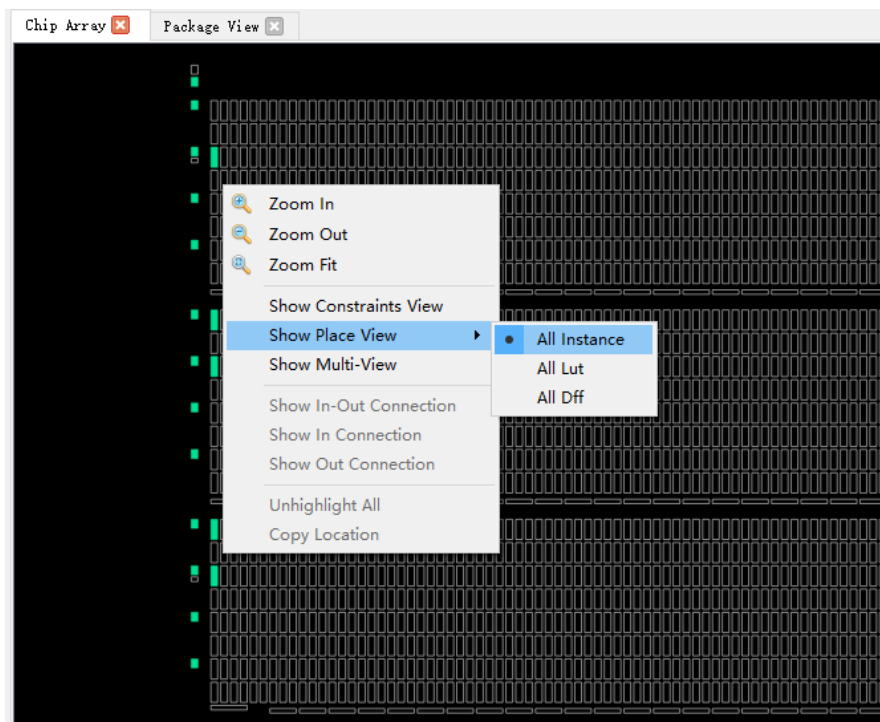


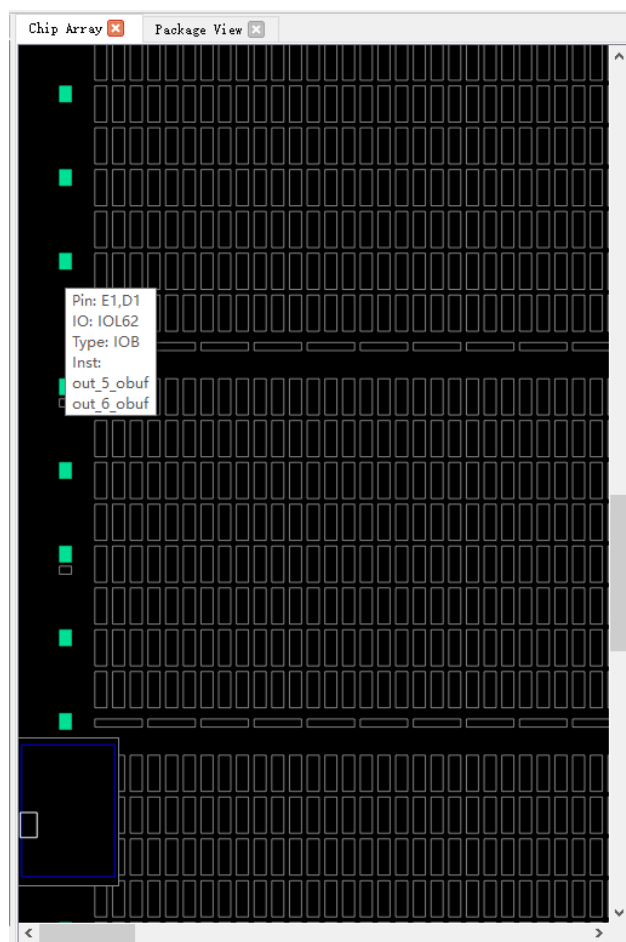
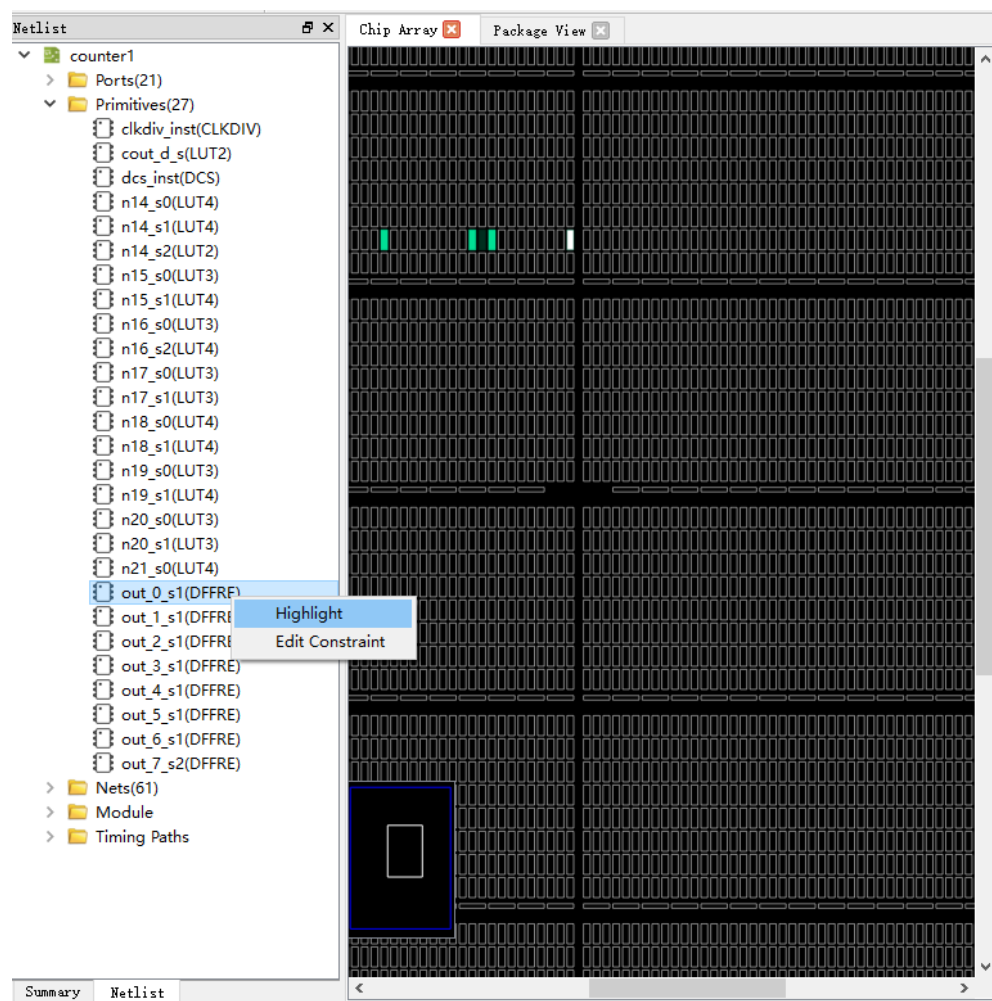
Figure 3-41 Mouse Hovering Display

Figure 3-42 Right-click to Select Highlight



3.3.5 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 3-43, 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 current
- Pull Mode: Set pull-up mode
- PCI Clamp: Set the switch of PCI protocol
- Hysteresis: Set hysteresis
- Open Drain: Turn on/off open drain
- Vref: Set reference voltage
- Single Resistor: Turn on/off single-ended resistor
- Diff Resistor: Turn on/off differential resistor
- Bank Vccio: 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 [DS981, GW5AT series of FPGA Products Data Sheet](#).

Figure 3-43 I/O Constraints View

I/O Constraints											
	Port	Direction	Diff Pair	Location	Bank	Exclusive	IO Type	Drive	Pull Mode	PCI Clamp	Hys ^
2	cin	input				False	LVC MOS18	N/A	UP	N/A	N
3	clk	input				False	LVC MOS18	N/A	UP	N/A	N
4	clk0	output				False	LVC MOS18	8	UP	N/A	I
5	cout	output				False	LVC MOS18	8	UP	N/A	I
6	data[0]	input				False	LVC MOS18	N/A	UP	N/A	N
7	data[1]	input				False	LVC MOS18	N/A	UP	N/A	N

Primitive Constraints

Primitive Constraint is used to constrain primitive location, as shown in Figure 3-44, 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 3-44 Primitive Constraints View

Primitive	Type	Locations	Exclusive
1	out_Z[7]	DFFE	False

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 3-45; 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 3-45 Group Constraints View

Group	Type	Members Number	Members Exclusive	Locations	Locations Exclusive
1	grp1	Primitive	False	R3C4	False

Resource Reservation

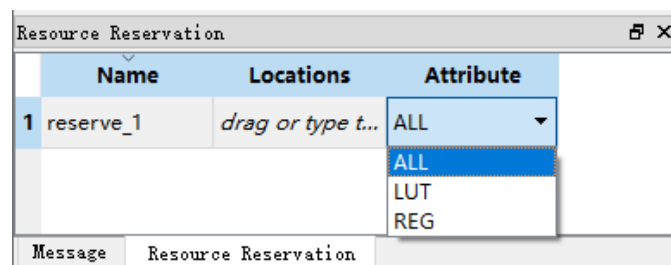
Resource Reservation is used for reservation constraints on the resources available in the current package, as shown in Figure 3-46; and the functions are as follows.

- 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 3-46 Resource Reservation View



Clock Net Constraints

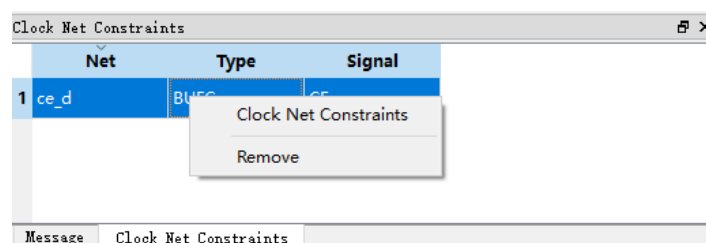
Clock Net Constraints is used for global clock assignment constraints on the net in the design, as shown in Figure 3-47, and the functions are as follows.

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

Note!

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

Figure 3-47 Clock Assignment View



GCLK Primitive Constraints

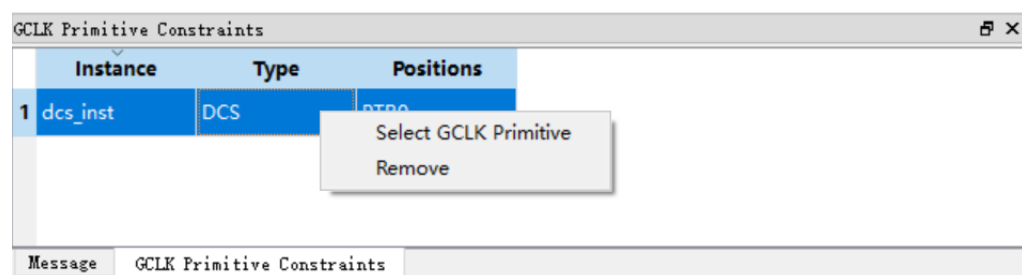
GCLK Primitive Constraints is used to constrain global clock primitives, and its window is as shown in Figure 3-48, with the following functions:

- Display all global clock constraints, including Instance name, type, and quadrant location.
- Support a right-click function for adding new global clock constraints and deleting existing constraints.

Note!

The window for creating a new global clock constraint is shown in Figure 3-17.

Figure 3-48 GCLK Primitive Constraints



HCLK Primitive Constraints

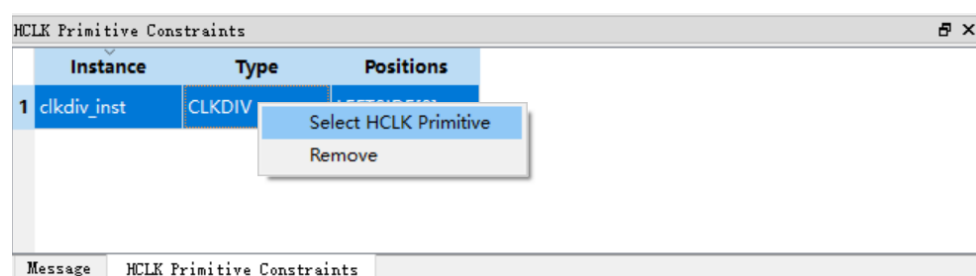
HCLK Primitive Constraints is used to constrain high-speed clock primitives, and its window is as shown in Figure 3-49, with the following functions:

- Display all location constraints for Instance related with high-speed clock, including Instance name, type, and high-speed clock location.
- Support a right-click function for adding new high-speed clock constraints and deleting existing constraints.

Note!

The window for creating a new high-speed clock constraint is shown in Figure 3-18.

Figure 3-49 HCLK Primitive Constraints



Vref Constraints

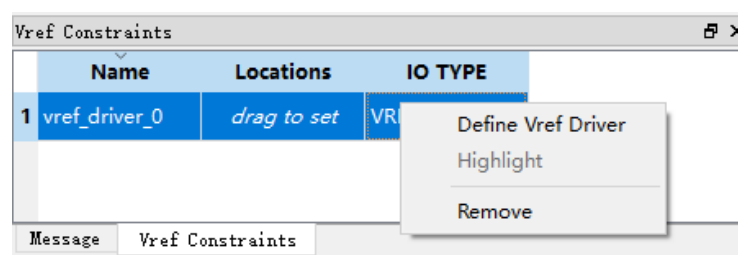
Vref Constraints is used for the external reference voltage of the bank where the constraint is located, as shown in Figure 3-50, and the functions are as follows.

- 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 3-50 Vref Constraints View



3.3.6 Message Window

The message window is as shown in Figure 3-51, and it displays the output.

Figure 3-51 Message Window



4 Using FloorPlanner

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

4.1 Create Constraint File

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

1. Start FloorPlanner as described in 3.2 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

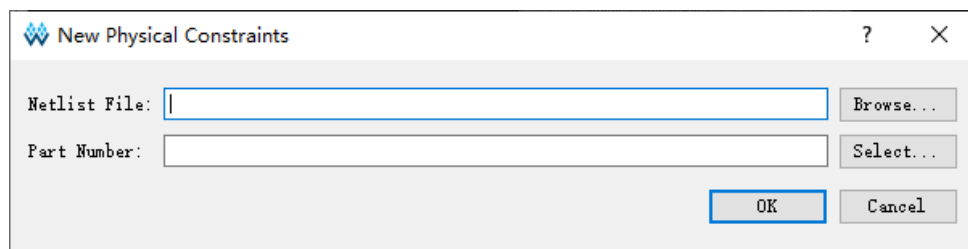


Figure 4-2 Select Device

Select Device

Filter

Series: Package:

Device: Speed:

Device Version:
*no version number is initial version

Part Number	Device	Device Version	Package	Speed	Voltage	IO	LUT	
GW5AT-LV138PG484AES	GW5AT-138	B	PBGA484A	ES	LV	297	138240	1
GW5AT-LV138GW391AES	GW5AT-138	B	GW391A	ES	LV	324	138240	1
GW5AT-LV138GW391ES	GW5AT-138	B	GW391	ES	LV	324	138240	1
GW5AT-LV138FPG676AES	GW5AT-138	B	FCPBGA676A	ES	LV	312	138240	1
GW5AT-LV138GW391ES	GW5AT-138		GW391	ES	LV	324	138240	1
GW5AT-LV138FPG676AES	GW5AT-138		FCPBGA676A	ES	LV	312	138240	1

OK Cancel

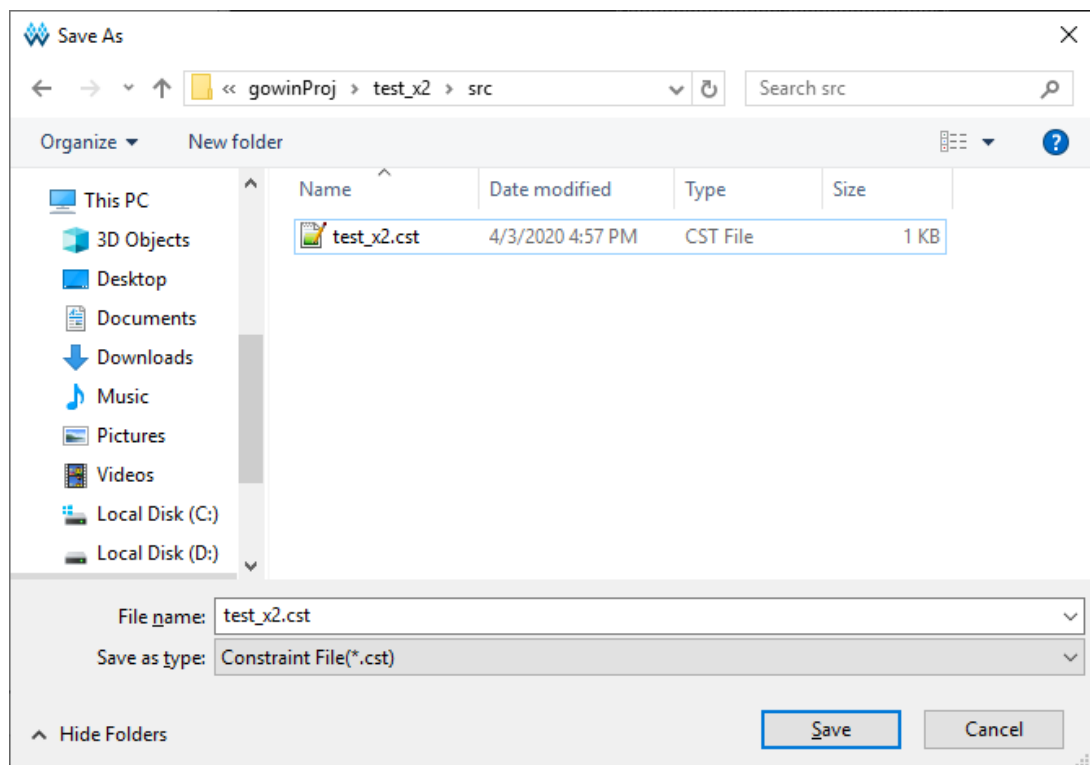
Note!

- You can select the device, package, and all Gowin FPGA devices, as shown in Figure 4-2.
- Start FloorPlanner using the third way in 3.2 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, and reference voltage, etc. Constraints can be generated via Constraints menu, see [3.3.1 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, clko);
output [7:0] out;
output cout;
output clko;
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";

DCS dcs_inst (
    .CLKOUT(clko),
    .CLKSEL(4'b0000),
    .CLKIN0(clkout),
    .CLKIN1(clkout),
    .CLKIN2(clkout),
    .CLKIN3(clkout),
    .SELFORCE(1'b0)
);
```

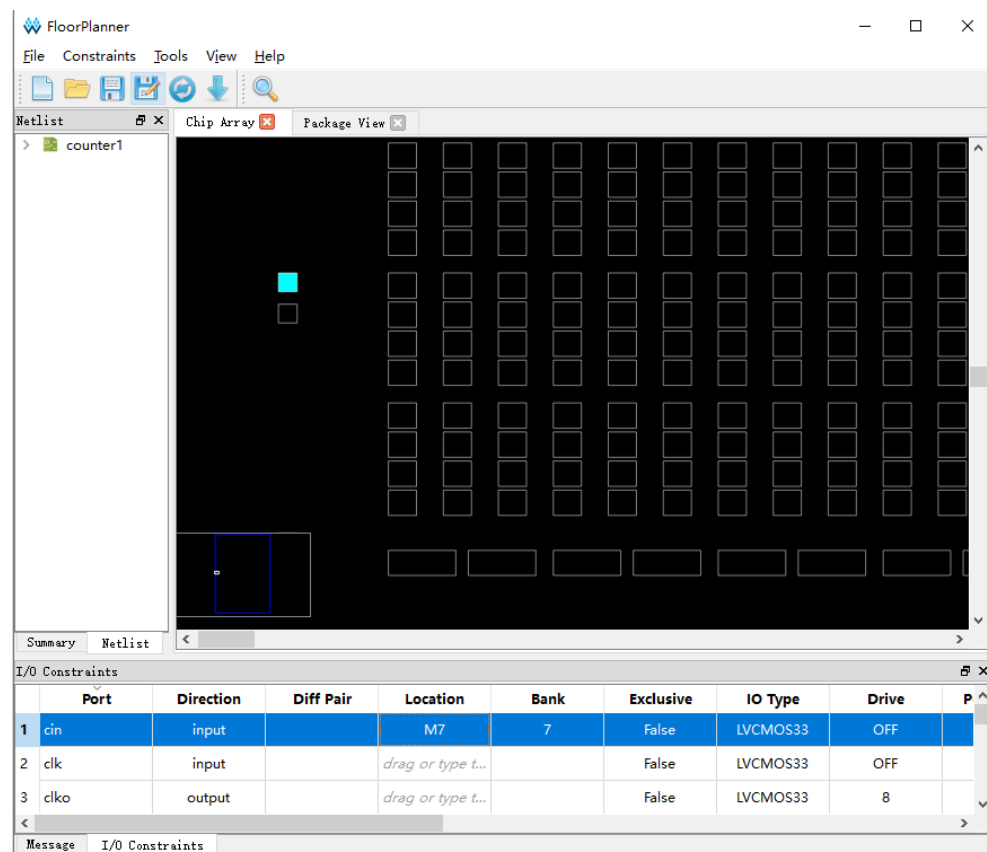
```
defparam dcs_inst.DCS_MODE = "RISING";
endmodule
```

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 "cin" and drag it to "M7" in Chip Array, as shown in Figure 4-4.
3. Port "cin" location is displayed as M7.

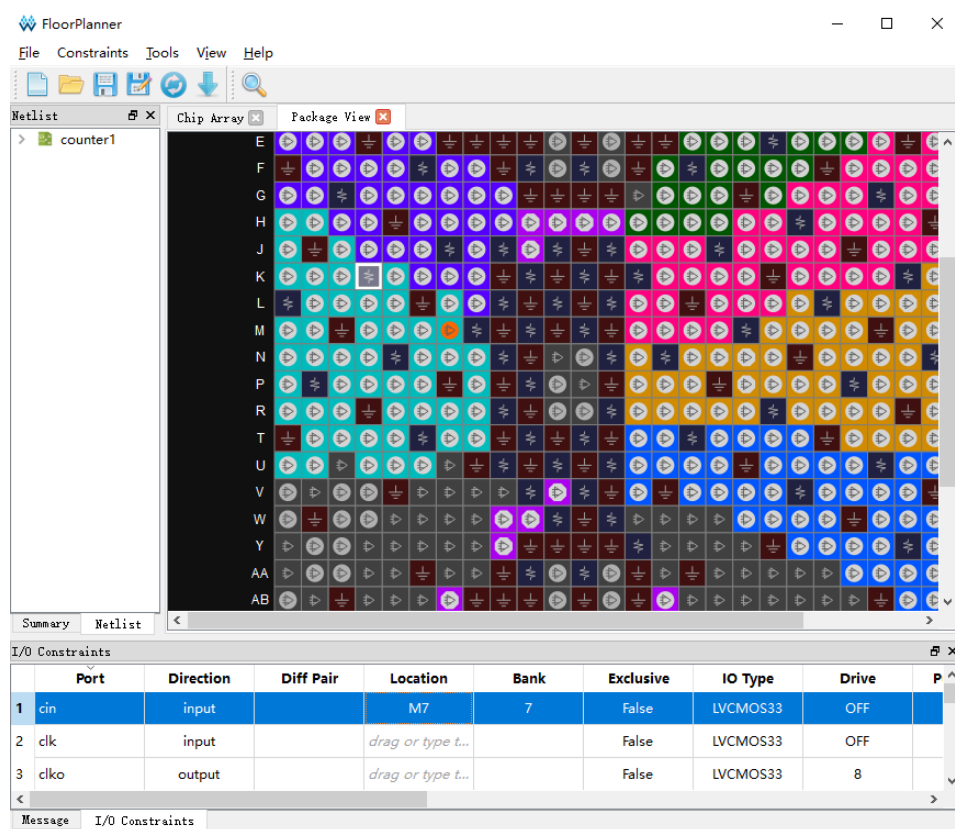
Figure 4-4 Drag to Chip Array to Create I/O Constraints



Drag to Package View to create I/O constraints.

1. Click IO Constraints.
2. Select Port "cin" and drag it to "M7" in Package View, as shown in Figure 4-5.
3. Location for Port "cie" is displayed as M7.

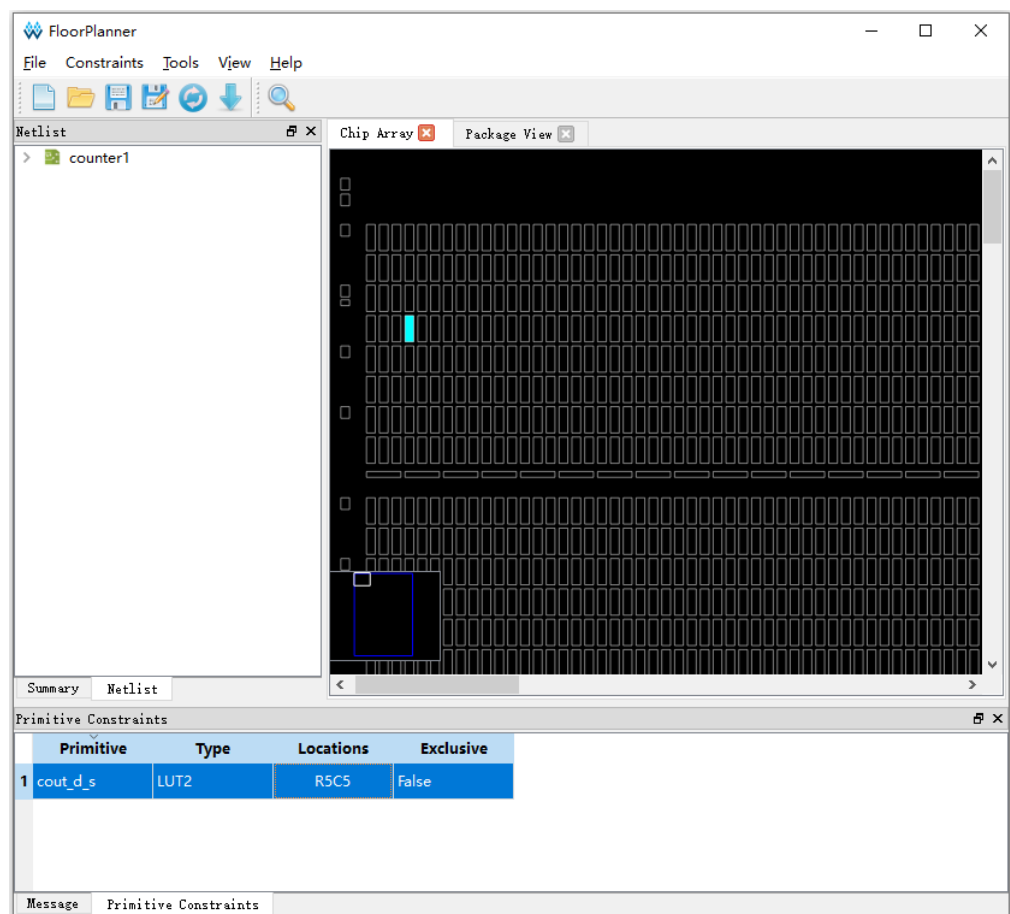
Figure 4-5 Drag to Package View to Create I/O Constraints



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 "New Primitive Group" pops up.
2. Enter "grp1" and click "+", then "Primitive Finder" dialog box pops up.
3. Select "n17_s0" and "cout_d_s"; click "OK", then add them to Members list.

4. Enter "R5C10" in "Locations", as shown in Figure 4-8.
5. Click "OK" in "New Primitive Group" dialog box to create primitive group constraints, as shown in Figure 4-9.

Figure 4-8 Create Primitive Group Constraints

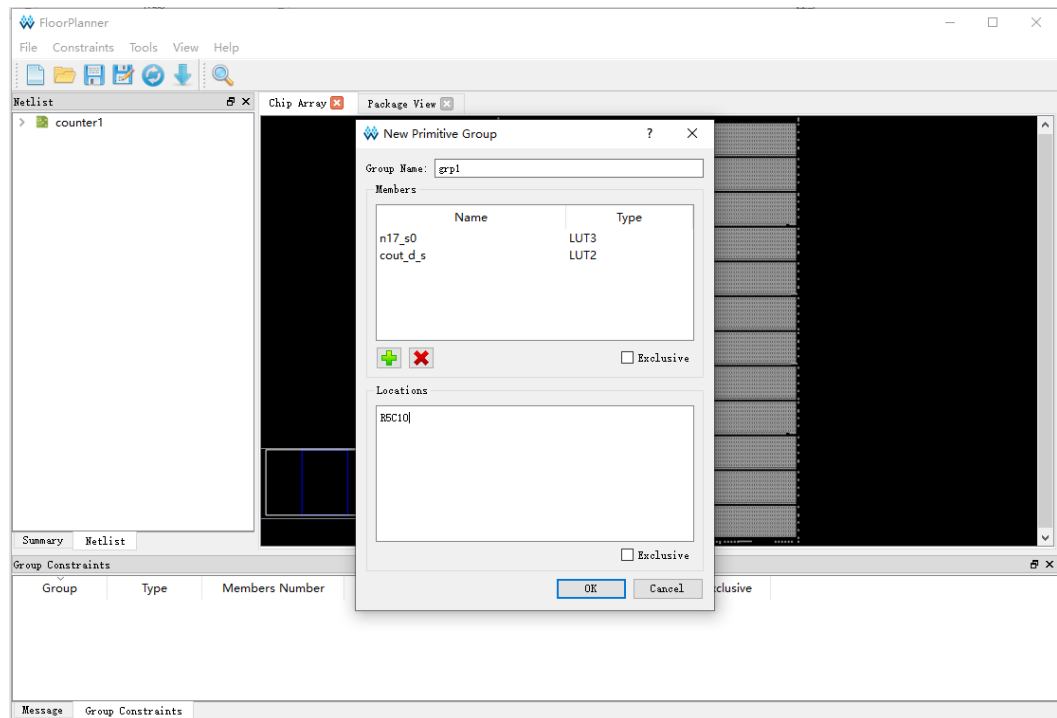
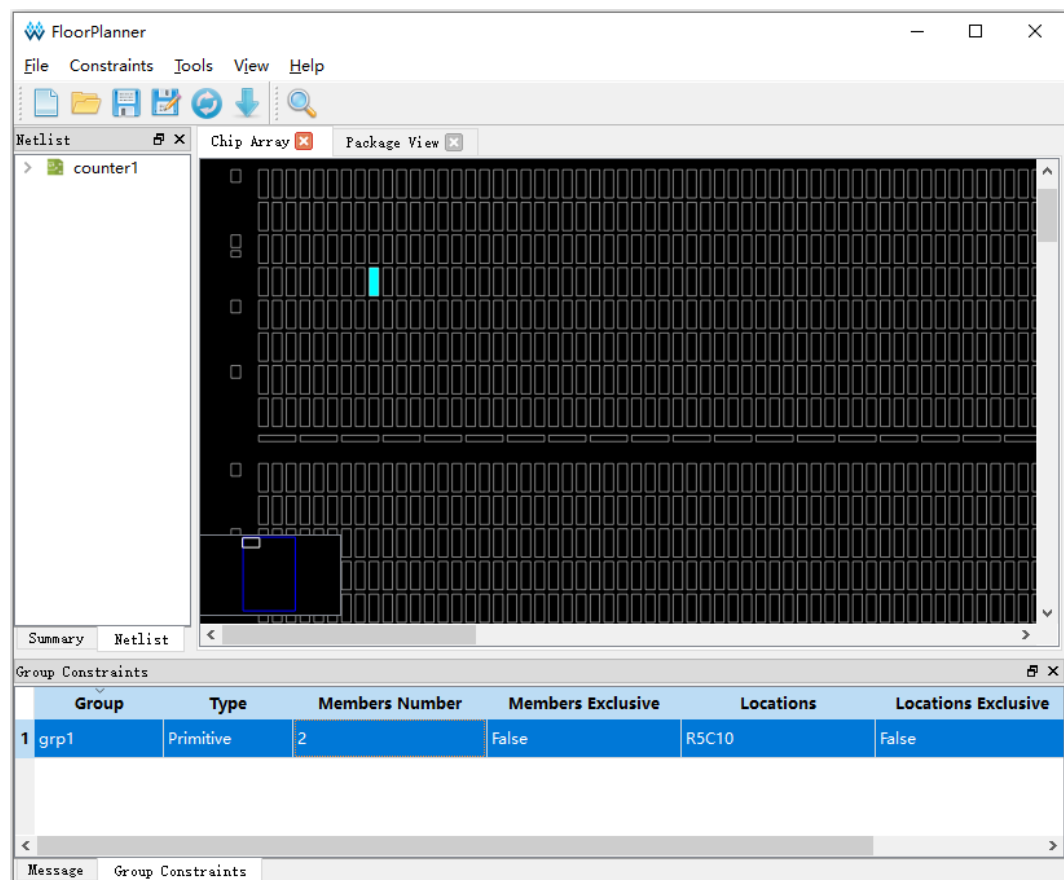


Figure 4-9 Primitive Group Constraints

**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 "Primitive Finder" dialog box pops up.
3. Select the primitives "cout_0_s1" and "cout_1_s1" 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

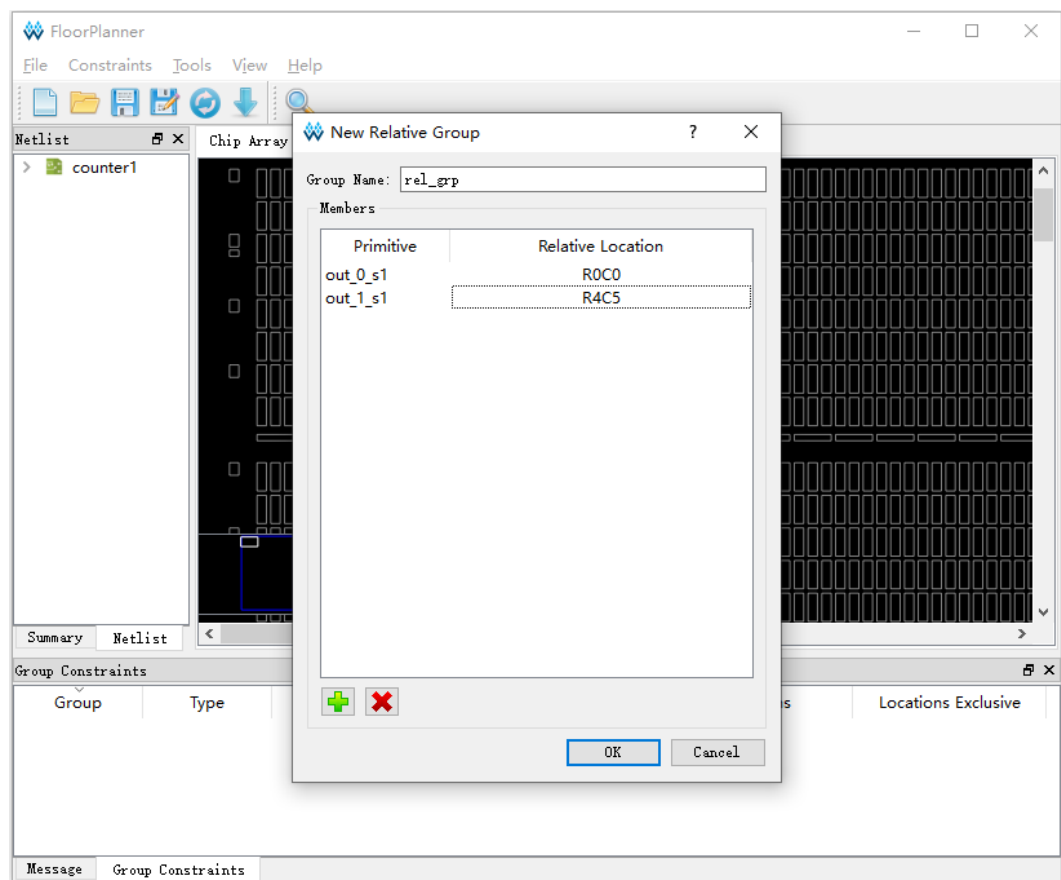
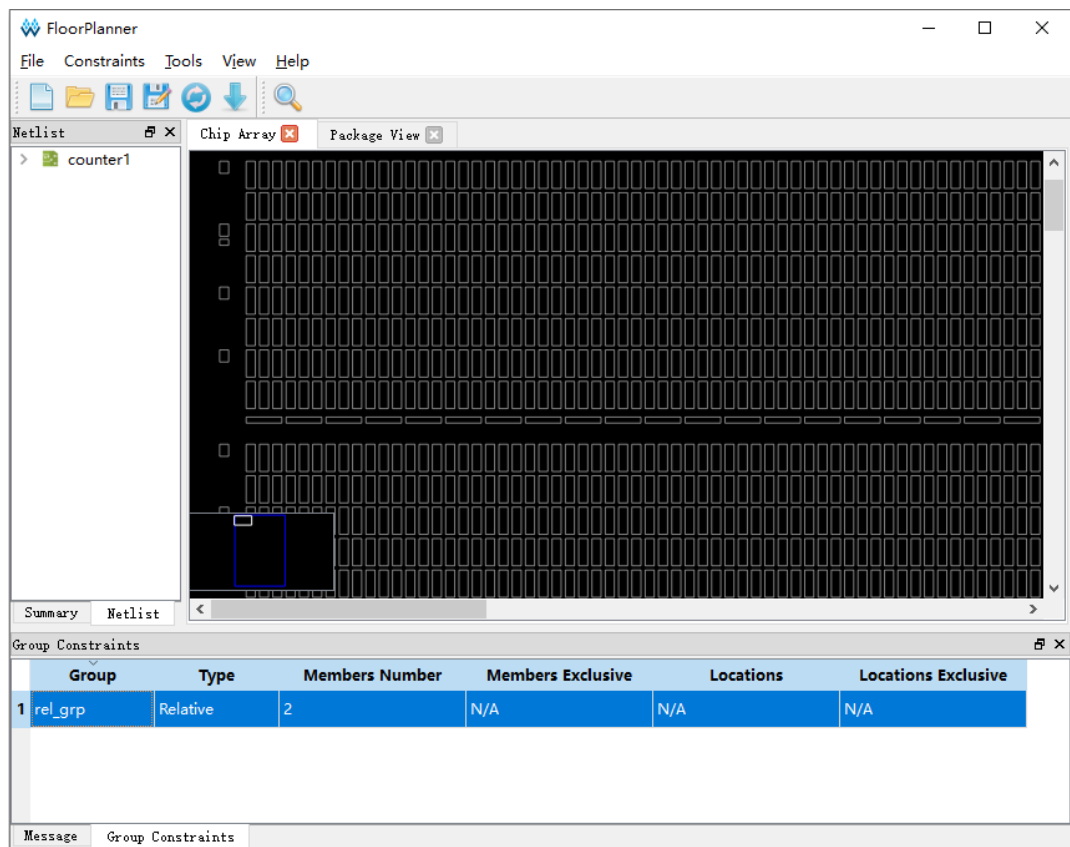


Figure 4-11 Relative Group Constraints



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

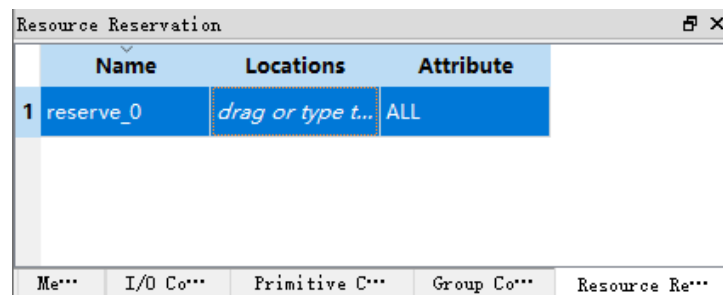


Figure 4-13 Resource Reservation

Resource Reservation		
	Name	Locations
1	reserve_0	BSRAM_R10[1]
		Attribute
		ALL
Me*** I/O Co*** Primitive C*** Group Co*** Resource Re***		

4.2.6 Edit Clock Net Constraints

1. Right-click "Clock Net Constraints" and select "Clock Net Constraints", then "Clock Net Constraints" dialog box pops up.
2. Click "+" and "Select Net" dialog box pops up. Select a Net and click "OK".
3. Select clock type and set signal, 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 Assignment Constraints

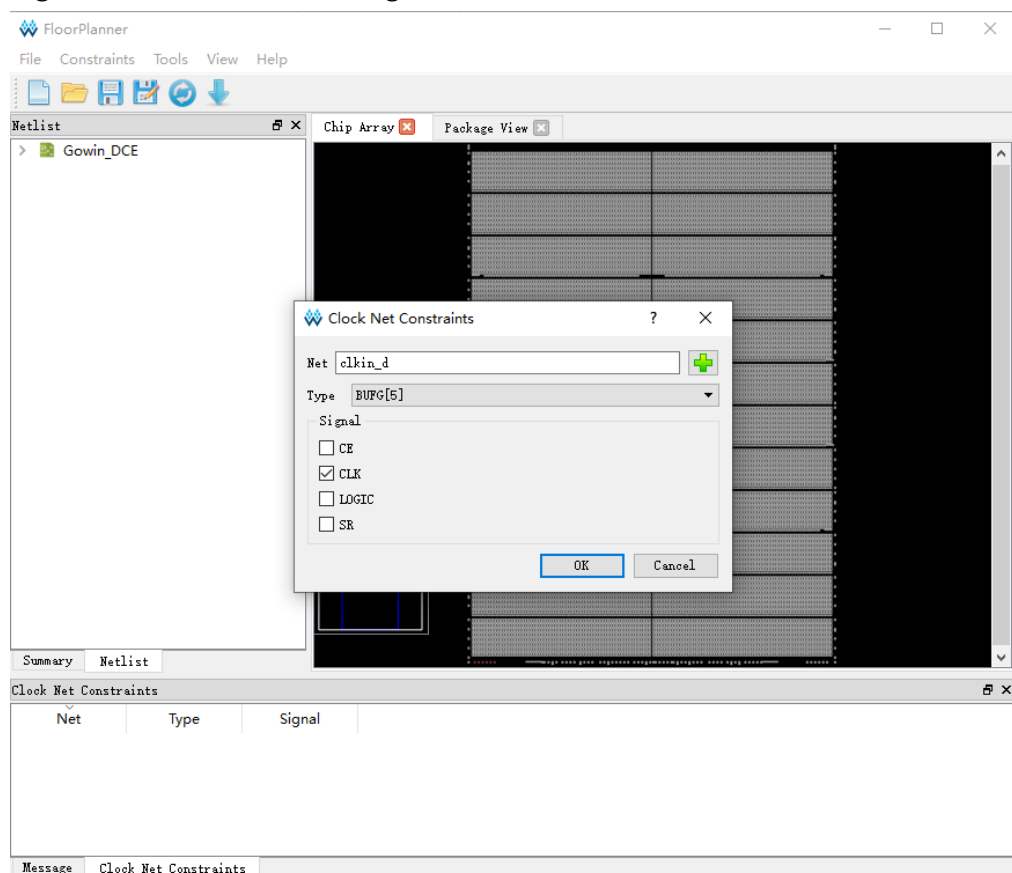
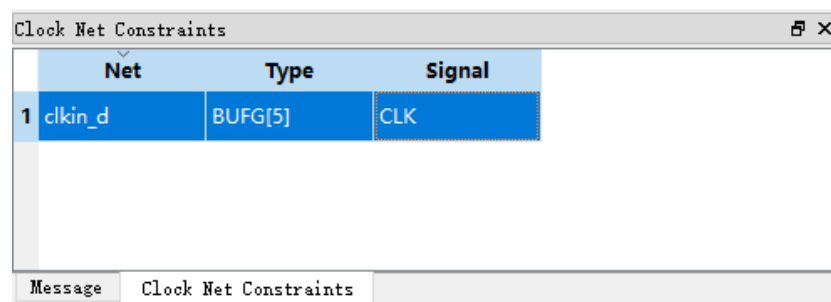


Figure 4-15 Clock Assignment Constraints



4.2.7 Edit GCLK Primitive Constraints

GCLK Primitive Constraints only supports constraints for DCS and DCE.

The steps to create GCLK Primitive Constraints are as follows:

1. In the GCLK Primitive Constraints editing window, right-click and select "Select GCLK Primitive", and the GCLK Primitive Constraints dialog box pops up.
2. Click "+" to open GCLK selection dialog box; select Instance, and click "OK".
3. Select the global clock constraint location using "Position", as shown in Figure 4-16.
4. Click "OK" in the GCLK Primitive Constraints dialog box, and the constraint will be added, as shown in Figure 4-17.

Figure 4-16 Create GCLK Primitive Constraints

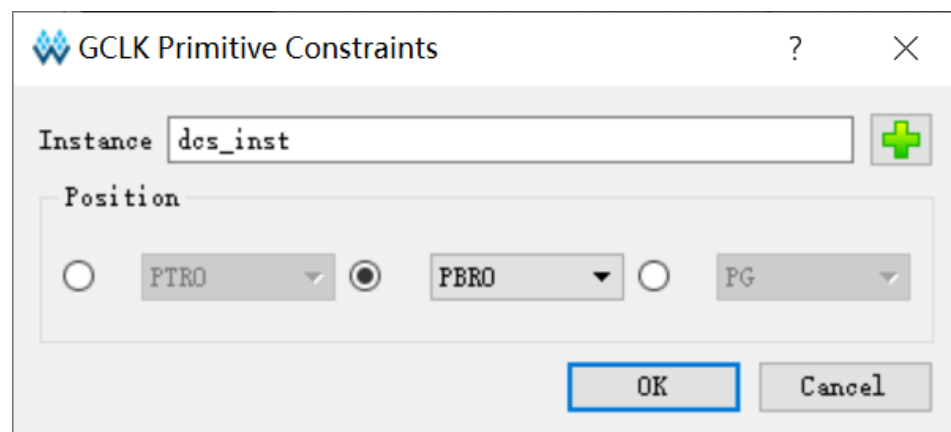


Figure 4-17 GCLK Primitive Constraints

	Instance	Type	Positions
1	dcs_inst	DCS	PBR0

Message: GCLK Primitive Constraints

4.2.8 Edit HCLK Primitive Constraints

HCLK Primitive Constraints only supports Instance of CLKDIV and DLLDLY.

The steps to create HCLK Primitive Constraints are as follows:

1. In the HCLK Primitive Constraints editing window, right-click and select "Select HCLK Primitive", and the HCLK Primitive Constraints dialog box pops up.
2. Click "+" to open HCLK selection dialog box; select Instance, and click "OK".
3. Select the high-speed clock constraint location using "Position", as shown in Figure 4-18.
4. Click "OK" in the HCLK Primitive Constraints dialog box, and the constraint will be added, as shown in Figure 4-19.

Figure 4-18 Create HCLK Primitive Constraints

HCLK Primitive Constraints

Instance:

Position:

☐ LEFTSIDE[0]
 ☐ RIGHTSIDE[0]
 ☒ BOTTOMSIDE[0]

OK Cancel

Figure 4-19 HCLK Primitive Constraints

HCLK Primitive Constraints			
	Instance	Type	Positions
1	clkdiv_inst	CLKDIV	BOTTOMSIDE[0]

Message 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 B5 in Chip Array. The location of the Vref Constraints is displayed as "B5", as shown in Figure 4-21.

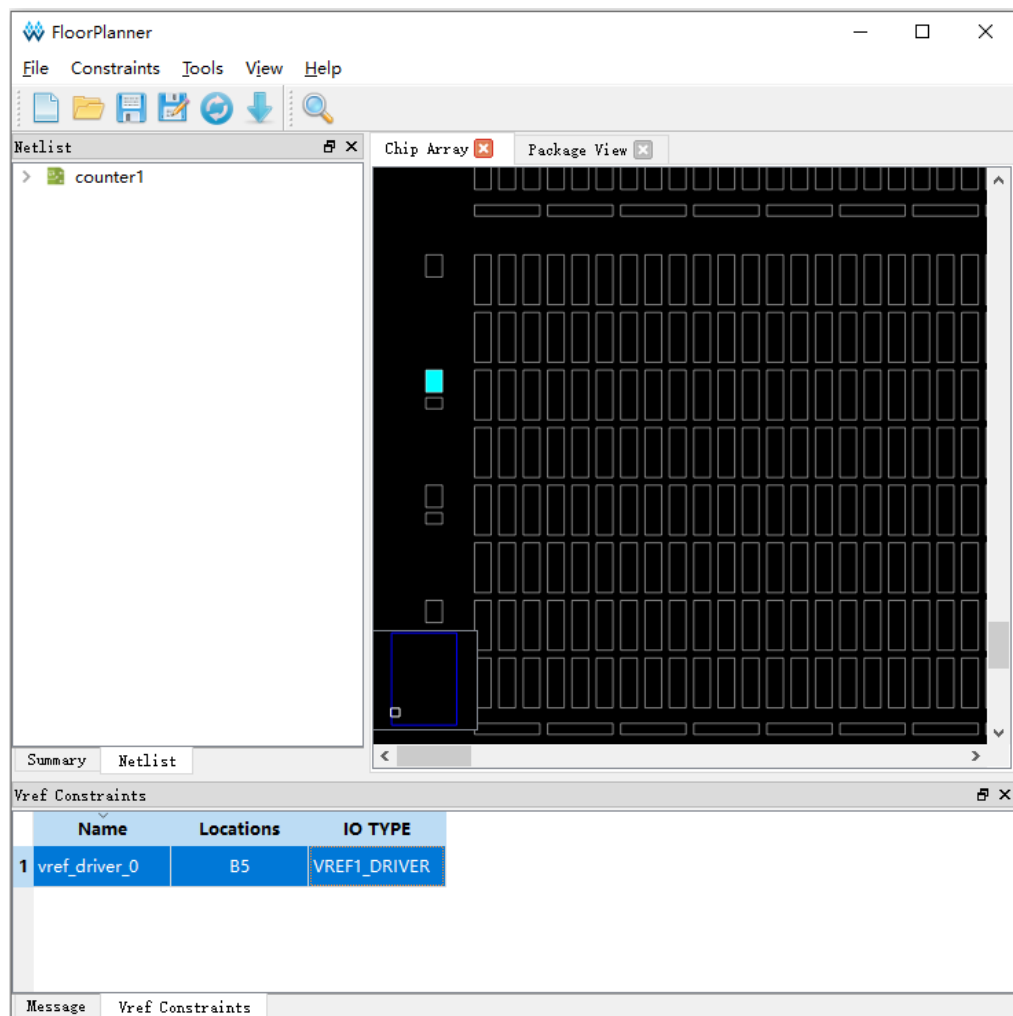
Figure 4-20 Create Vref Constraints

Vref Constraints			
	Name	Locations	IO TYPE
1	vref_driver_0	drag to set	VREF1_DRIVER

Message 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 4-23.

Figure 4-21 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 "B5" in Package View. The location of the Vref Constraints is displayed as "B5", as shown in Figure 4-22.

Figure 4-22 Drag to Package View to Generate Vref Constraints Location

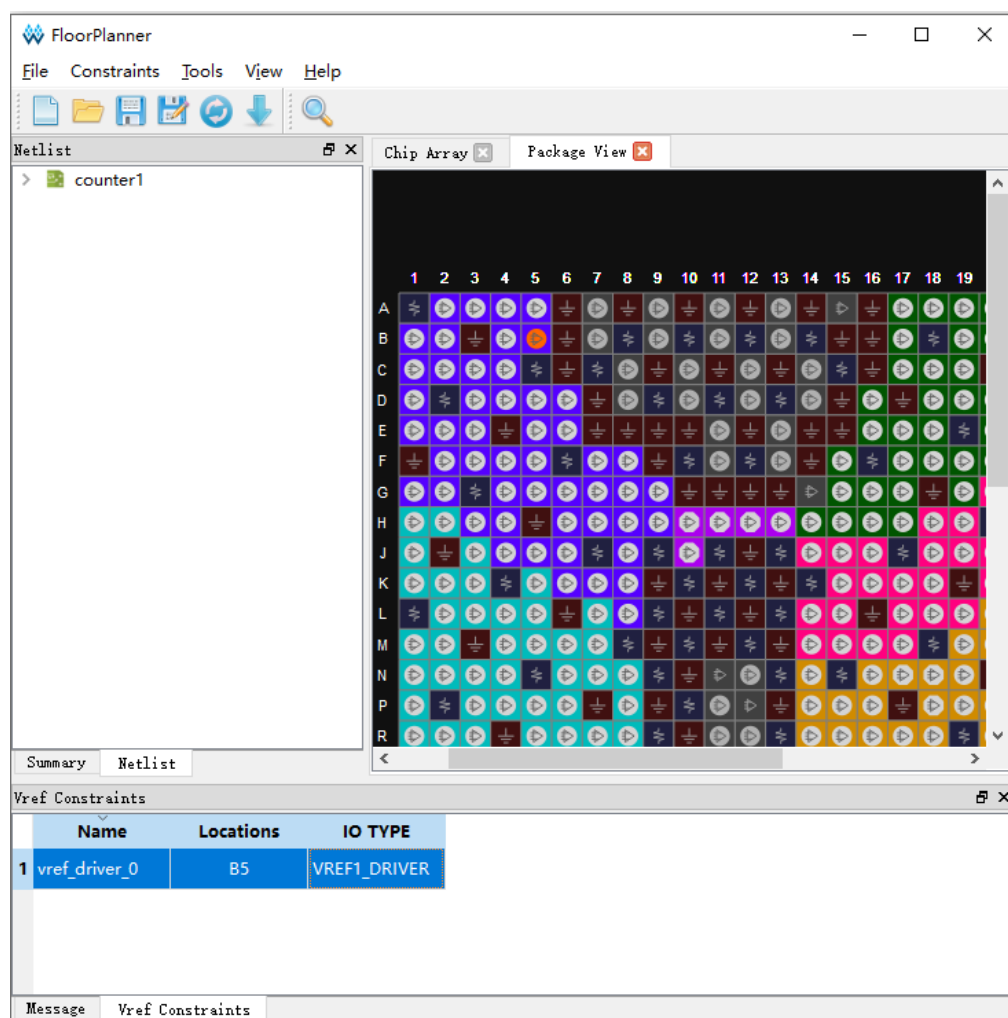


Figure 4-23 Duplicate Names Prompt

