



# Gowin 设计物理约束 用户指南

SUG935-1.0, 2020-05-09

## **版权所有© 2020 广东高云半导体科技股份有限公司**

未经本公司书面许可，任何单位和个人都不得擅自摘抄、复制、翻译本文档内容的部分或全部，并不得以任何形式传播。

### **免责声明**

本文档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止发言或其它方式授予任何知识产权许可。除高云半导体在其产品的销售条款和条件中声明的责任之外，高云半导体概不承担任何法律或非法律责任。高云半导体对高云半导体产品的销售和 / 或使用不作任何明示或暗示的担保，包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知识产权的侵权责任等，均不作担保。高云半导体对文档中包含的文字、图片及其它内容的准确性和完整性不承担任何法律或非法律责任，高云半导体保留修改文档中任何内容的权利，恕不另行通知。高云半导体不承诺对这些文档进行适时的更新。

## 版本信息

日期	版本	说明
2020/05/09	1.0	初始版本。

# 目录

目录 .....	i
图目录 .....	iii
表目录 .....	vii
<b>1 关于本手册 .....</b>	<b>1</b>
1.1 手册内容 .....	1
1.2 相关文档 .....	1
1.3 术语、缩略语 .....	1
1.4 技术支持与反馈 .....	2
<b>2 简介 .....</b>	<b>3</b>
<b>3 物理约束编辑 .....</b>	<b>4</b>
3.1 启动 FloorPlanner .....	4
3.2 新建和打开约束文件 .....	5
3.2.1 新建约束文件 .....	5
3.2.2 FloorPlanner 输出约束文件 .....	7
3.2.3 打开约束文件 .....	9
3.3 FloorPlanner 界面 .....	10
3.3.1 菜单栏 .....	11
3.3.2 Project 和 Netlist 窗口 .....	22
3.3.3 Package View 窗口 .....	27
3.3.4 Chip Array 窗口 .....	32
3.3.5 Constraint 编辑窗口 .....	37
3.4 Message 窗口 .....	42

---

<b>4 创建 Constraints .....</b>	<b>43</b>
4.1 创建 Constraints 示例.....	43
4.2 I/O Constraints 创建.....	45
4.3 Primitive Constraints 创建.....	46
4.4 Group Constraints 创建 .....	47
4.4.1 Primitive Group Constraints 创建.....	47
4.4.2 Relative Group Constraints 创建 .....	49
4.5 Resource Reservation 创建 .....	51
4.6 Clock Assignment 创建 .....	52
4.7 Quadrant Constraints 创建 .....	53
4.8 Hclk Constraints 创建 .....	54
4.9 Vref Constraints 创建 .....	56
<b>5 时序优化 .....</b>	<b>60</b>
<b>附录 A 物理约束语法规范 .....</b>	<b>65</b>
A.1 I/O Constraints .....	65
A.2 PORT 属性约束 .....	66
A.3 Primitive Constraints .....	67
A.4 Group Constraints .....	70
A.4.1 Primitive Group Constraints .....	70
A.4.2 Relative Group Constraints .....	72
A.5 Resource Reservation.....	73
A.6 Vref Constraints.....	73
A.7 Quadrant Constraints .....	74
A.8 Clock Assignment.....	75
A.9 Hclk Constraints .....	76

# 图目录

图 3-1 菜单栏启动 FloorPlanner .....	4
图 3-2 Process 窗口启动.....	5
图 3-3 打开新建物理约束 .....	6
图 3-4 新建物理约束文件 .....	6
图 3-5 手写物理约束文件 .....	7
图 3-6 新建 FloorPlanner .....	7
图 3-7 选择器件.....	8
图 3-8 保存输出文件 .....	9
图 3-9 打开物理约束 .....	10
图 3-10 FloorPlanner 界面 .....	11
图 3-11 File 菜单 .....	11
图 3-12 Tools 菜单 .....	12
图 3-13 原语查找界面 .....	13
图 3-14 新建原语组.....	14
图 3-15 正确原语组界面.....	14
图 3-16 无效位置.....	15
图 3-17 无效位置.....	15
图 3-18 创建相对位置的组 .....	16
图 3-19 正确的相对组界面 .....	16
图 3-20 预留约束.....	16
图 3-21 时钟约束.....	17
图 3-22 象限约束（GW1N 家族） .....	18

图 3-23 象限约束 (GW2A 家族) .....	18
图 3-24 Hclk 约束 .....	19
图 3-25 Vref 约束 .....	19
图 3-26 选择 One Hit drag 原语 .....	20
图 3-27 选择 One Hit Drag 位置 .....	20
图 3-28 One Hit Drag 产生约束 .....	20
图 3-29 查找界面 .....	21
图 3-30 View 菜单 .....	22
图 3-31 Windows 菜单 .....	22
图 3-32 Project 窗口 .....	23
图 3-33 Netlist 窗口 .....	24
图 3-34 BUS 和非 BUS 结合显示 .....	25
图 3-35 层级显示 .....	25
图 3-36 时序路径显示 .....	26
图 3-37 Netlist 右键功能 .....	27
图 3-38 GW1NRF-4B-QFN48 Package view 界面 .....	28
图 3-39 Package View 右键功能 .....	29
图 3-40 差分对显示 .....	29
图 3-41 Top View .....	30
图 3-42 Bottom View .....	30
图 3-43 GW1N-9-WLCSP81M Top View .....	31
图 3-44 GW1N-9-WLCSP81M Bottom View .....	31
图 3-45 Chip Array 界面 .....	32
图 3-46 网格模式约束 .....	33
图 3-47 宏单元模式约束 .....	33
图 3-48 原语模式约束 .....	34
图 3-49 Chip Array 右键功能 .....	36
图 3-50 Show Place View 显示 .....	36

图 3-51 时序路径高亮显示 .....	37
图 3-52 I/O 约束窗口 .....	38
图 3-53 原语约束窗口 .....	39
图 3-54 组约束窗口 .....	39
图 3-55 预留约束窗口 .....	40
图 3-56 时钟约束窗口 .....	40
图 3-57 象限约束窗口 .....	41
图 3-58 Hclk 约束窗口 .....	41
图 3-59 Vref 约束窗口 .....	42
图 3-60 Message 窗口 .....	42
图 4-1 拖拽到 Chip Array 创建 I/O Constraints .....	45
图 4-2 拖至 Package View 创建 I/O Constraints .....	46
图 4-3 拖拽到 Chip Array 创建 Primitive Constraints .....	47
图 4-4 Group Constraints 编辑器右键菜单 .....	47
图 4-5 创建 Primitive Group Constraints .....	48
图 4-6 Primitive Group Constraints .....	49
图 4-7 Relative Group Constraints 创建 .....	50
图 4-8 Relative Group Constraints .....	51
图 4-9 创建 Resource Reservation 约束 .....	51
图 4-10 Resource Reservation .....	52
图 4-11 Clock Assignment 约束创建 .....	53
图 4-12 Clock Assignment 约束 .....	53
图 4-13 Quadrant Constraints 创建 .....	54
图 4-14 Quadrant Constraints .....	54
图 4-15 Hclk Constraints 创建 .....	55
图 4-16 Hclk Constraints .....	55
图 4-17 Vref Constraints 创建 .....	56
图 4-18 拖拽至 Chip Array 窗口生成 Vref Constraints location 信息 .....	57

---

图 4-19 拖拽至 Package View 窗口生成 Vref Constraints location 信息.....	58
图 4-20 Vref Constraints 名字重复.....	59
图 5-1 生成器件布局信息文件设置.....	61
图 5-2 读取时序路径文件.....	62
图 5-3 高亮关键路径操作.....	63
图 5-4 关键路径信号流向示意图.....	63
图 5-5 调整后的位置信息.....	64

# 表目录

表 1-1 术语、缩略语 ..... 1

# 1 关于本手册

## 1.1 手册内容

本手册主要描述高云半导体 FloorPlanner，介绍 Gowin 云源软件 FloorPlanner 的界面使用以及语法规则，旨在帮助用户快速实现物理约束。有关本手册中的软件界面截图和支持的产品列表等信息参考的是 1.9.6 Beta 版本。因软件版本更新，部分信息可能会略有差异，具体以用户软件版本信息为准。

## 1.2 相关文档

通过登录高云半导体网站 [www.gowinsemi.com.cn](http://www.gowinsemi.com.cn) 可下载、查看以下相关文档：[SUG100](#)，Gowin 云源软件用户指南

## 1.3 术语、缩略语

本手册中的相关术语、缩略语及相关释义请参见表 1-1。

表 1-1 术语、缩略语

术语、缩略语	全称	含义
FPGA	Field Programmable Gate Array	现场可编程门阵列
I/O	Input/Output	输入/输出
SIP	System In a Package	系统级封装
Chip Array	Chip Array	芯片阵列
Package View	Package View	封装视图
Posp File	Post-Place File	器件布局信息文件
VREF	Voltage Reference	参考电压

## 1.4 技术支持与反馈

高云半导体提供全方位技术支持，在使用过程中如有任何疑问或建议，可直接与公司联系：

网址：[www.gowinsemi.com.cn](http://www.gowinsemi.com.cn)

E-mail：[support@gowinsemi.com](mailto:support@gowinsemi.com)

Tel: +86 755 8262 0391

# 2 简介

**Gowin FloorPlanner** 是高云半导体面向市场自主研发的布局与物理约束编辑以及时序优化工具，支持对 I/O、Primitive（原语）、block（B-SRAM、DSP）、Group 等属性及位置信息的读取与修改，同时可根据用户的配置生成新的布局与约束文件，文件中规定了 I/O 的属性信息，原语、模块的位置信息等。**Gowin FloorPlanner** 提供了简单快捷的布局与约束编辑功能，提高编写物理约束文件的效率，同时可以根据器件布局和时序路径进行时序优化。

**Gowin FloorPlanner** 功能特点：

- 支持用户设计文件、约束文件的读入，以及约束文件的输出；
- 支持对用户设计文件中 IO Port、Primitive、Group 约束信息等的显示；
- 支持用户新建、编辑、修改约束信息；
- 支持 Chip Array 的网格模式、宏单元模式以及原语模式显示；
- 支持依据 Package 信息的 Package View 显示；
- 支持 Chip Array 和 Package View 的同步显示；
- 支持约束位置信息的实时显示及区别显示；
- 支持拖拽设置位置信息的功能；
- 支持 One Hit Drag 一键产生约束的功能；
- 支持 IO Port 的属性配置功能，支持批量配置；
- 支持 Clock Assignment 的显示、编辑功能；
- 支持约束信息合法性检查的功能；
- 支持文件和时序路径文件的读入；
- 支持时序优化。

# 3 物理约束编辑

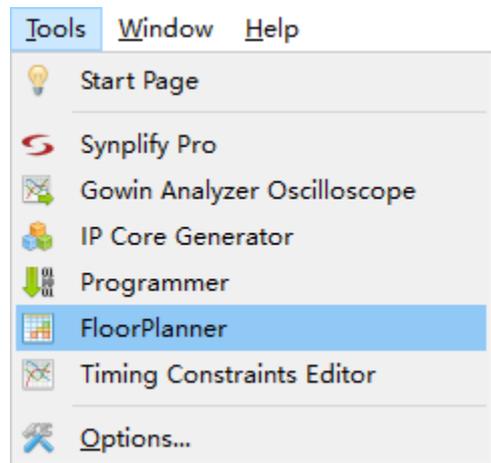
Gowin FloorPlanner 能够创建和修改物理约束，可以提供表格化的约束编辑和高效的网表单元查找功能，提高编写物理约束文件的效率。

## 3.1 启动 FloorPlanner

可通过以下两种方式启动 FloorPlanner：

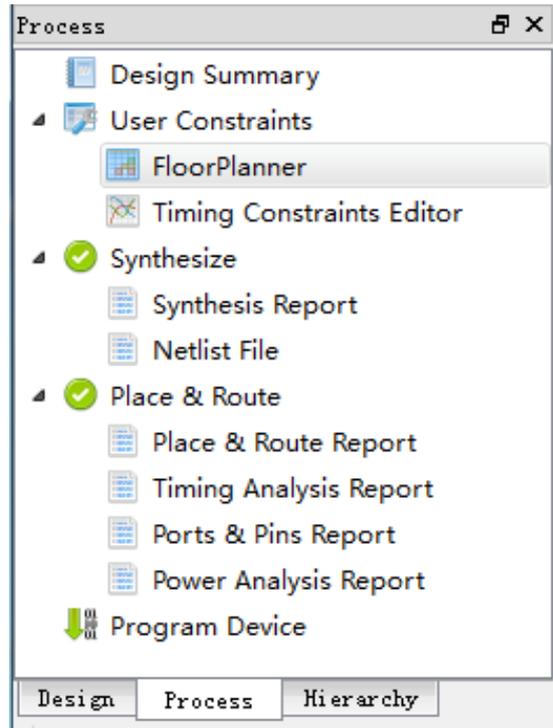
1. 单击“IDE > Tools”，打开“FloorPlanner”，如图 3-1 所示。

图 3-1 菜单栏启动 FloorPlanner



2. 建立工程在 Process 窗口运行 Synthesize 后，双击“FloorPlanner”，如图 3-2 所示。

图 3-2 Process 窗口启动



注!

- 如需 Gowin FloorPlanner 进行约束，应先加载网表文件；
- 通过第一种方式启动 Gowin FloorPlanner 时，需要通过“File>new”加载网表文件；
- 通过第二种方式启动 Gowin FloorPlanner 时，工程中的网表文件会自动加载。

## 3.2 新建和打开约束文件

工程中需物理约束文件约束 I/O 的位置、属性，Primitive 的位置信息等，可通过以下两种方式完成物理约束文件：

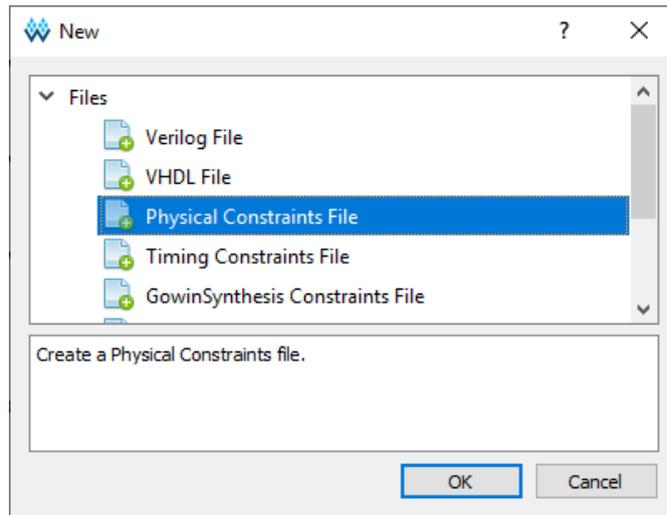
- 手动书写；
- 通过 Gowin FloorPlanner 工具输出约束文件。

### 3.2.1 新建约束文件

新建约束文件步骤如下：

1. 单击“File > New”，打开“New”对话框；
2. 选择“Physical Constraints File”，如图 3-3 所示。

图 3-3 打开新建物理约束

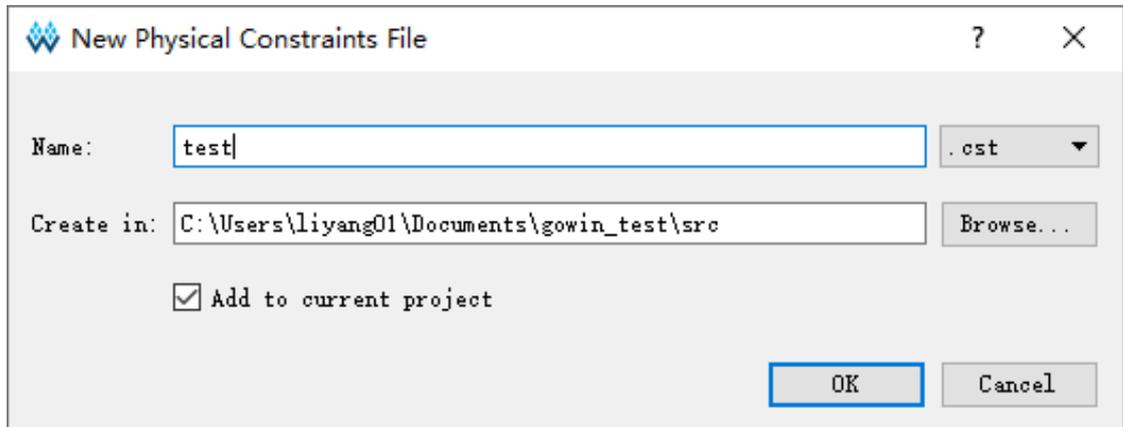


注！

亦可通过以下两种方式打开“New”对话框：

- 使用快捷键 **Ctrl + N**；
- 单击工具栏上的“New”图标。
- 单击“OK”，出现如图 3-4 所示的对话框。

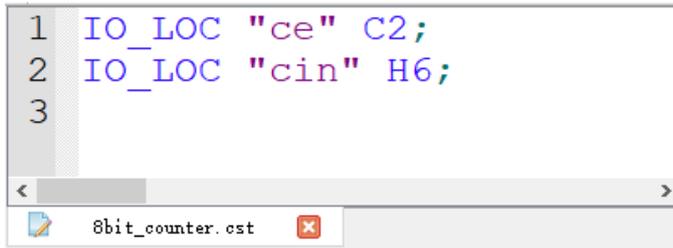
图 3-4 新建物理约束文件



- **Name**：新建物理约束文件的名称，后缀支持.cst 和.ucf；
- **Create in**：通过“Browse”对话框选择约束文件的存放位置，默认存于工程目录的 src 文件夹下；
- **Add to current project**：选择该选项约束文件自动添加到工程中。

设置完成后，出现新建的约束文件，用户可根据高云半导体物理约束语法规则书写约束文件，如图 3-5 所示。

图 3-5 手写物理约束文件



```

1 IO_LOC "ce" C2;
2 IO_LOC "cin" H6;
3

```

### 3.2.2 FloorPlanner 输出约束文件

Gowin FloorPlanner 可输出新建的物理约束文件，亦可输出修改后的物理约束文件，操作步骤如下所示：

根据 [3.1](#) 启动 FloorPlanner 所述，启动 FloorPlanner；

1. 单击“File > New”，打开“New”对话框，如图 3-6 所示；

注！

亦可通过以下两种方式打开“New”对话框：

- 使用快捷键 Ctrl + N；
- 单击工具栏上的“New”图标。

2. 输入工程的网表文件，选择器件类型，单击“OK”。

图 3-6 新建 FloorPlanner

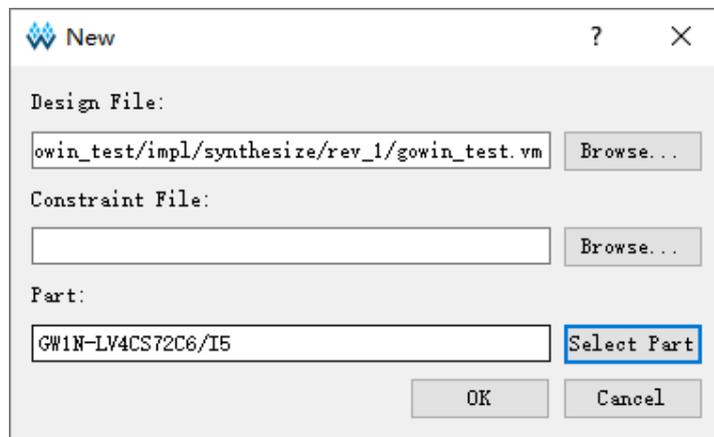
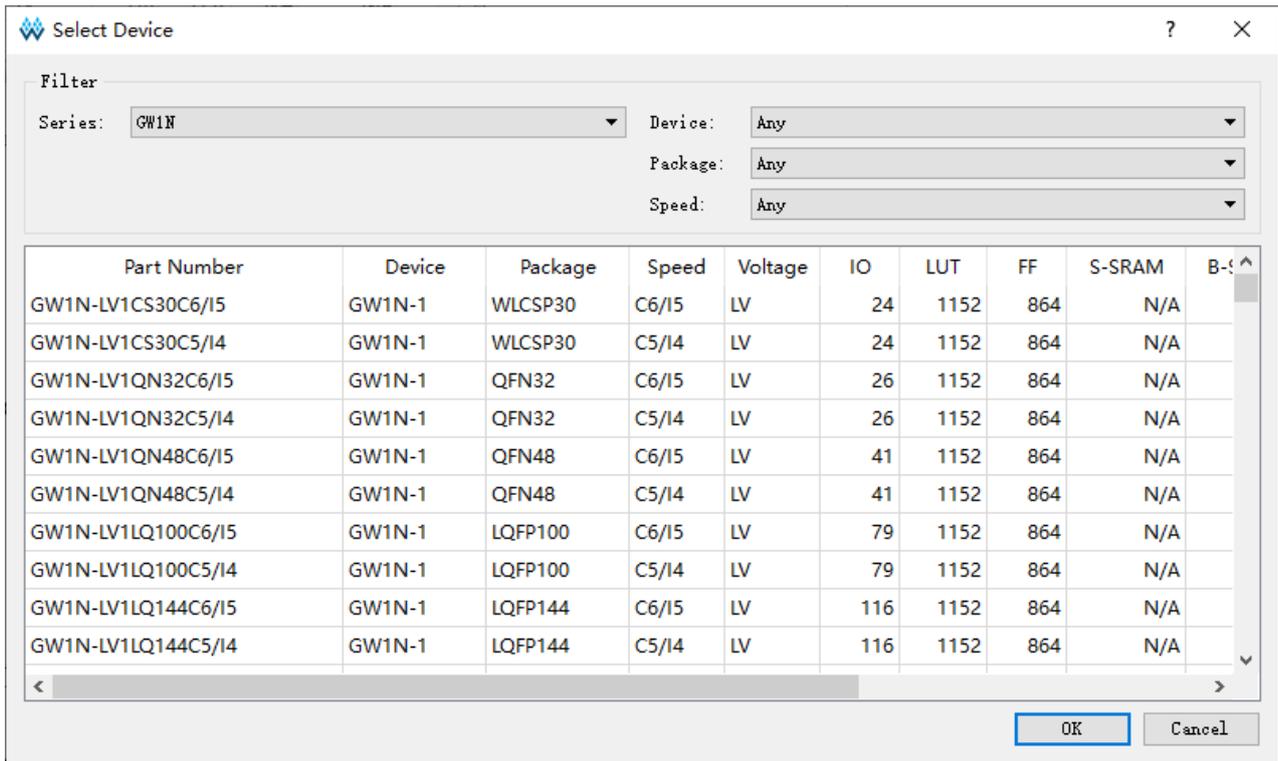


图 3-7 选择器件



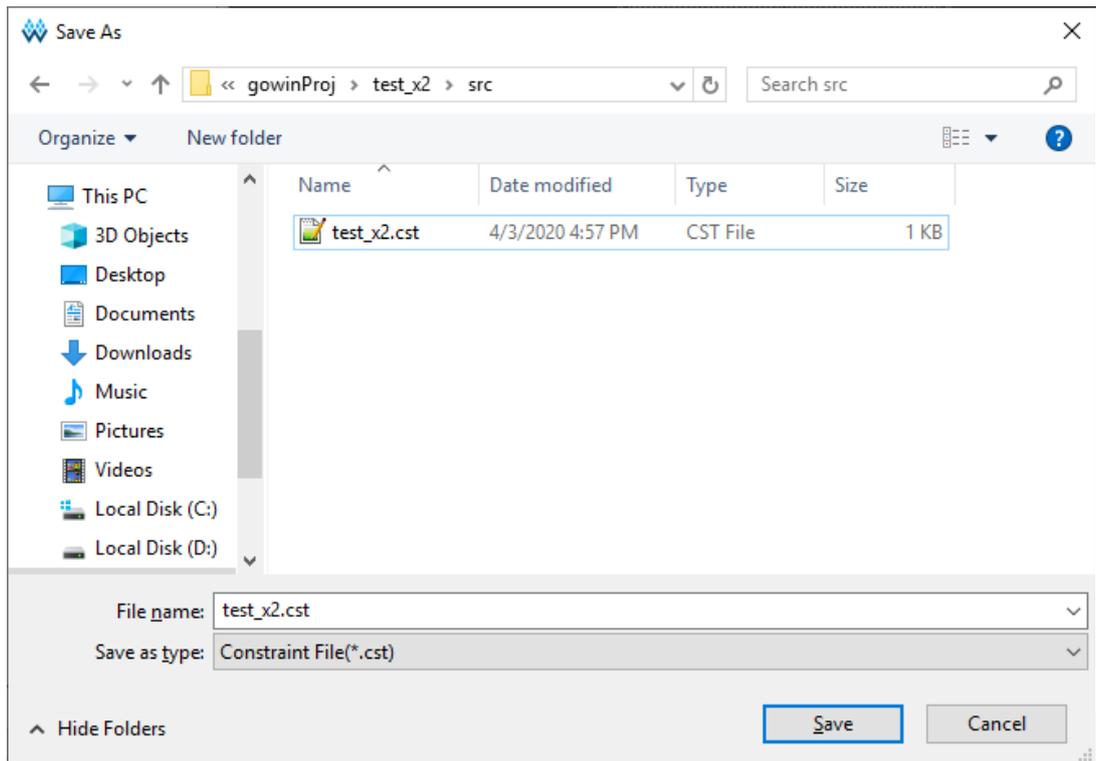
注！

- **Select Part** 按钮用于选取芯片、封装信息，支持高云半导体所有的 FPGA 器件，如图 3-7 所示。
- 启动 FloorPlanner 采用 3.1 启动 FloorPlanner 介绍中的第一种方式：

新建物理约束，在 FloorPlanner 主界面中可进行如下操作：

1. 用户通过拖拽等方式分配管脚位置；
2. 点击工具栏中的“Save”图标，即可输出约束文件；
3. 在弹出的“Save”对话框中，可修改文件名，如图 3-8 所示。

图 3-8 保存输出文件

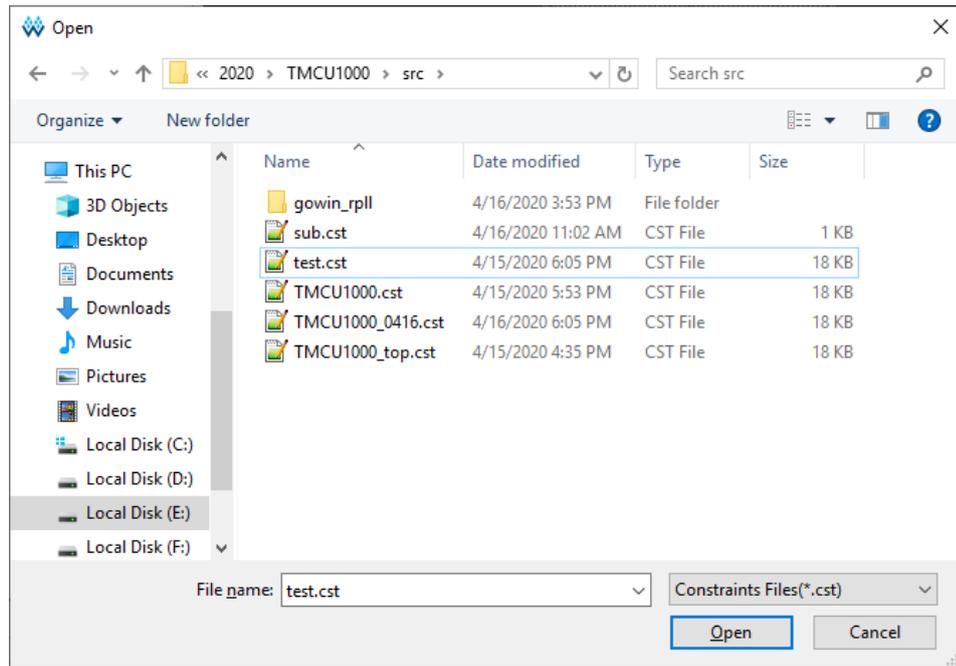


### 3.2.3 打开约束文件

打开约束文件步骤如下：

1. 在 IDE 界面中，单击“File > Open”菜单项；
2. 打开“Open File”对话框，如图 3-9 所示。

图 3-9 打开物理约束



注！

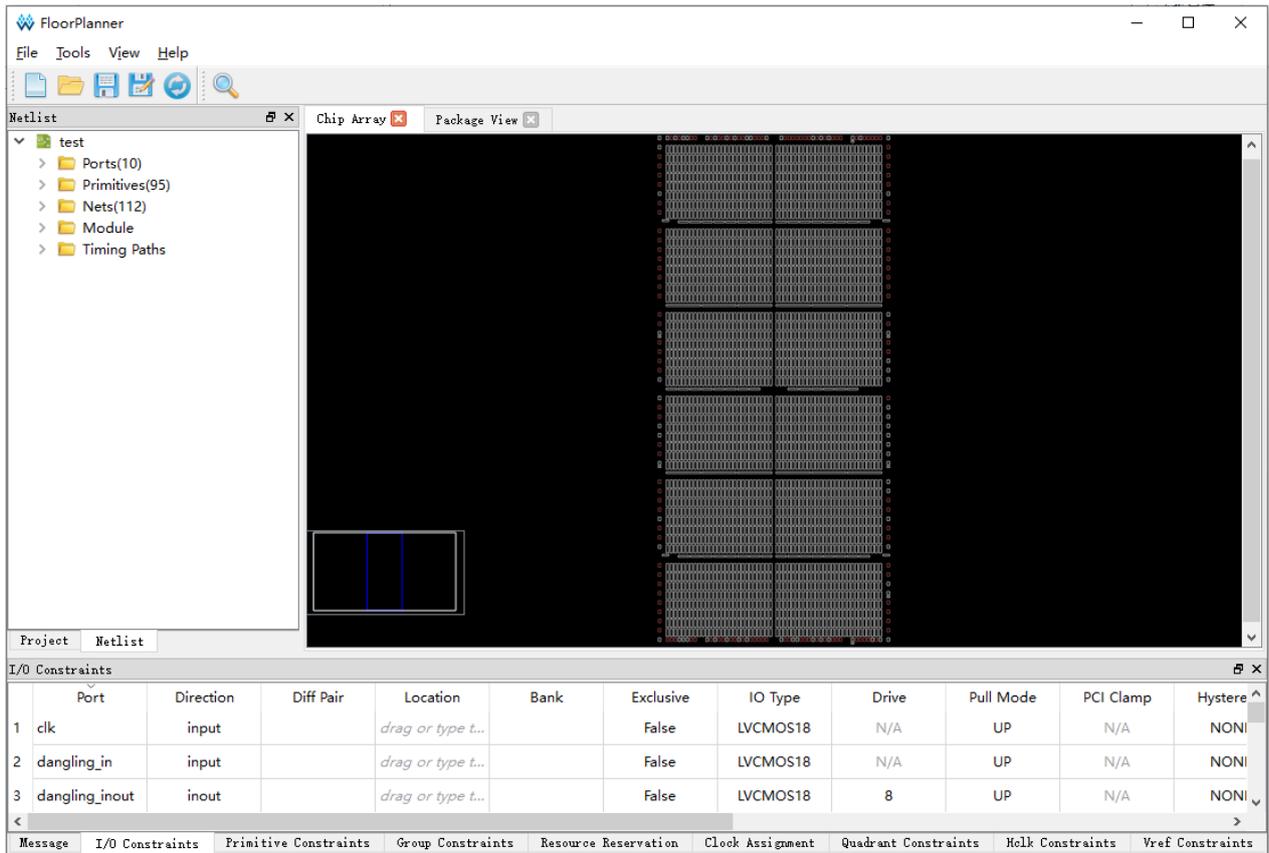
- 亦可通过以下两种方式打开“Open File”对话框：
- 使用快捷键 **Ctrl + O**；
- 单击工具栏上的“Open”图标。选择物理约束文件所在的目录，打开选中文件。

## 3.3 FloorPlanner 界面

新建或打开 FloorPlanner 界面（包含网表文件），如图 3-10 所示。

界面包括菜单栏、Netlist 窗口、Project 窗口、Chip Array 窗口、Package View 窗口、Message 窗口以及各类约束编辑窗口等。

图 3-10 FloorPlanner 界面

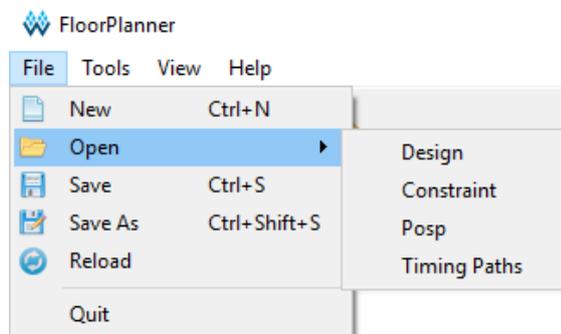


### 3.3.1 菜单栏

FloorPlanner 的菜单栏分为“File”、“Tools”、“View”及“Help”4个子菜单。

File 菜单界面如图 3-11 所示。

图 3-11 File 菜单



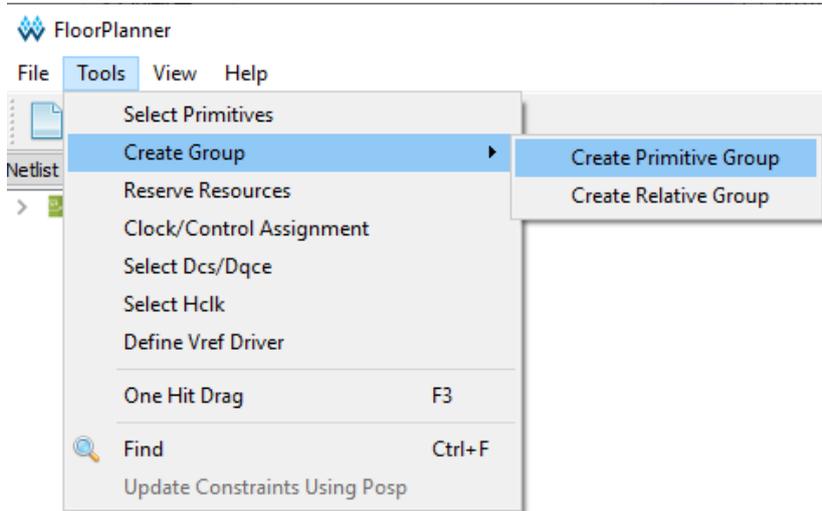
- **New:** 新建工程，添加用户设计、约束，设置芯片信息等，如图 3-6 所示；
- **Open:** 加载用户网表文件、约束文件、器件布局信息文件（posp 文件）或时序路径文件（timing paths 文件）；

- **Reload:** 当在磁盘或工程中对 **cst** 文件进行修改保存后，可以重新加载 **cst** 约束文件；
- **Save:** 当前约束信息的修改信息覆盖原约束文件；
- **Save As:** 将当前约束信息的修改信息输出到用户指定的文件中，默认采用网表文件名作为约束文件名称，用户可修改；
- **Quit:** 退出 **Gowin FloorPlanner** 软件。

**Tools** 菜单界面如图 3-12 所示。

- 创建各种约束信息；
- 提供特有的 **One Hit Drag** 功能、查找功能，以及约束信息合法性检查。
- 创建的各种约束信息实时显示在约束编辑窗口中，其对应的位置信息显示在 **Chip Array** 窗口和 **Package View** 窗口中。各功能菜单介绍如下：

图 3-12 **Tools** 菜单



### Select Primitives

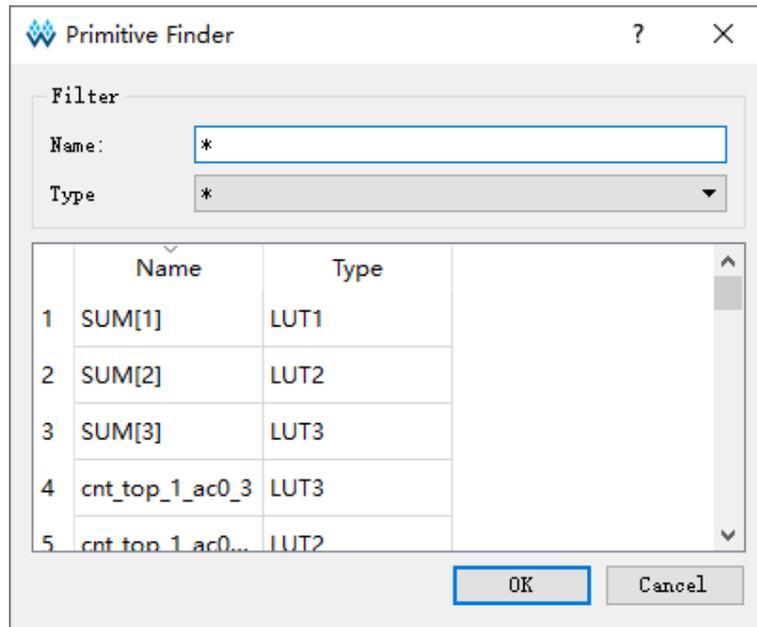
创建 **Primitive** 约束，选择 **Primitive** 创建对应的约束，单击该菜单弹出如图 3-13 所示对话框；

1. 可通过 **Primitives** 名称或类型进行查找，选择对应的 **Primitive**；
2. 单击“**OK**”，产生约束信息。

注！

- 约束信息显示在主界面底部的“**Primitive Constraints**”约束编辑窗口中；
- 用户可在编辑窗口中通过输入或拖拽的方式设置位置信息；
- 所设置的位置在 **Chip Array** 窗口中呈浅蓝色高亮显示。

图 3-13 原语查找界面



Create Group, 包括 Create Primitive Group 和 Create Relative Group, 各功能介绍如下:

### Create Primitive Group

1. 创建 Primitive Group 约束, 单击该菜单, 弹出如图 3-14 所示对话框;
2. 用户可设置 Group 的名称、包含的 Primitive、位置信息, 以及 Group 的 Exclusive 信息; 通过 “+” 和 “-” 两个按钮实现 Primitive 的添加和删除, 正确创建的 Primitive Group 如图 3-15 所示。

注!

- Group 的名称、包含的 Primitive、Group 的位置为必填项;
- 可通过以下方式输入 Group 的位置信息:
  - 通过手动方式输入;
  - 建立 Group 约束前, 在“Chip Array”窗口中, 复制位置, 粘贴到“New Primitive Group > Location”中。
- 3. Primitive Group 创建配置完成后, 单击 “OK”, 工具此时会对 Group 的位置信息进行语法检查。
  - 若位置信息不合理或不合法, 会弹出如图 3-16 和如图 3-17 所示的提示框, 用户需重新修改位置信息;
  - 若无错误提示, 单击 “OK”, 在 Chip Array 中会显示可用的位置。

注!

- 新产生的组约束显示在主界面底部的 “Group Constraints” 约束编辑窗口中。

- 在“Group Constraints”约束编辑窗口中，双击 Primitive Group 约束可打开如图 3-15 所示的对话框，重新进行编辑修改。

图 3-14 新建原语组

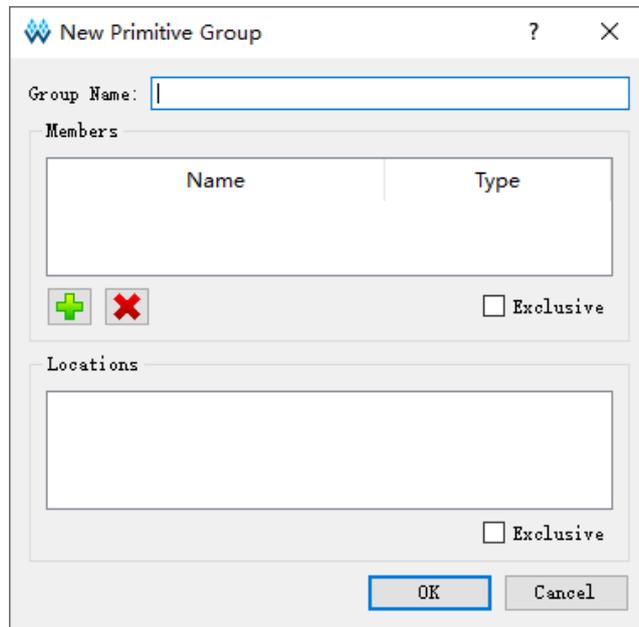


图 3-15 正确原语组界面

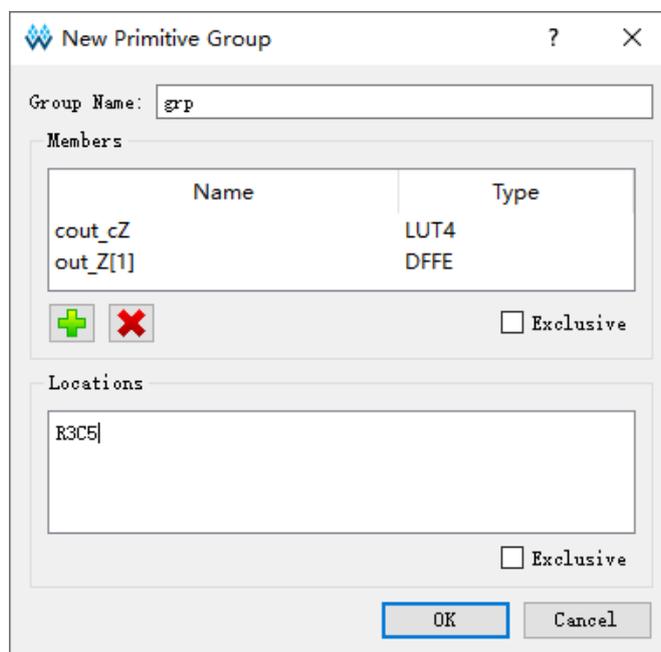


图 3-16 无效位置

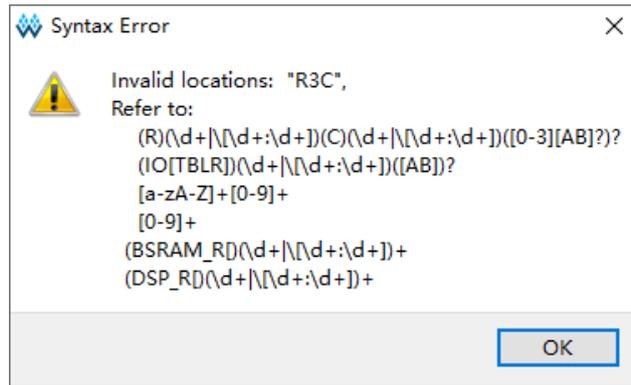
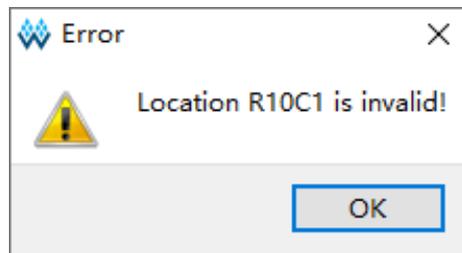


图 3-17 无效位置



### Create Relative Group

1. 创建具有相对位置的组约束，单击该菜单，弹出如图 3-18 所示对话框；
2. 用户可设置 Group 的名称、包含的 Primitive 以及各 Primitive 对应的相对位置信息；可通过“+”和“-”实现 Primitive 的添加和删除，创建成功的相对位置的组约束如图 3-19 所示。

注！

- Group 的名称、包含的 Primitive 及 Relative Location 为必填项；
- 可通过以下方式输入 Group 的位置信息：
  - 通过手动方式输入；
  - 在建立 Group 约束前，在“Chip Array”窗口中，复制位置，粘贴到“New Relative Group > Relative Location”中。

3. 配置完成后单击“OK”，产生约束信息，

注！

- 产生的约束信息显示在主界面底部的“Group Constraints”约束编辑窗口中。
- 在编辑窗口中，双击约束，重新打开如图 3-19 所示的对话框，可进行编辑修改。

图 3-18 创建相对位置的组

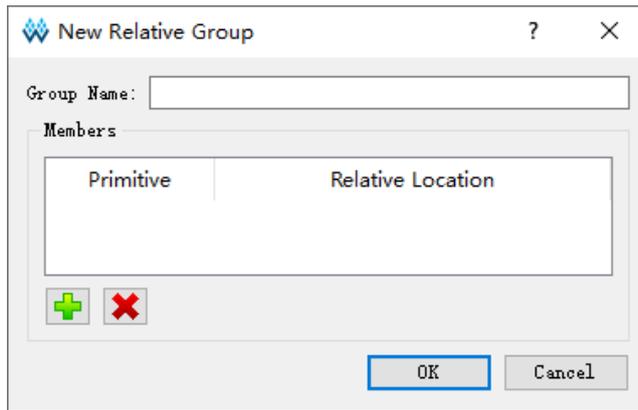
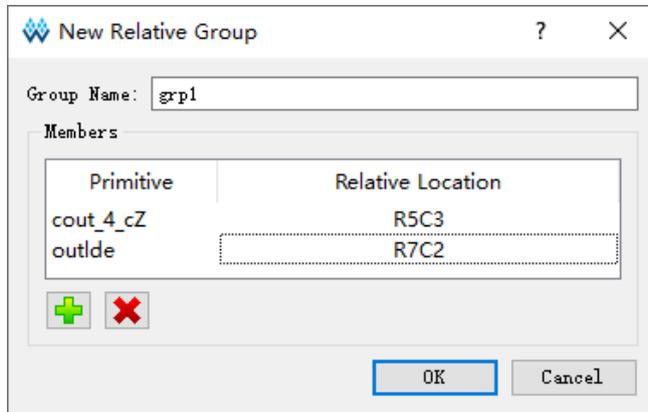


图 3-19 正确的相对组界面



### Reserve Resources

1. 创建预留约束，单击该菜单，将直接在主界面底部的“Resource Reservation”约束编辑窗口中新建一条约束；
2. 通过输入或拖拽的方式设置位置信息；
3. 双击“Attribute”栏可设置预留位置的属性，如图 3-20 所示。

注！

Name 属性用于区分不同的预留约束，不可修改该名称。

图 3-20 预留约束

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

## Clock/Control Assignment

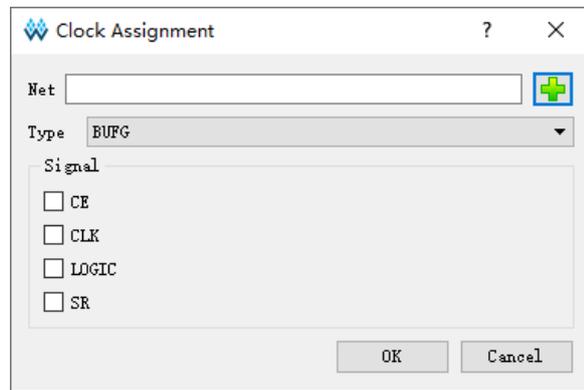
创建时钟约束，该约束的数目有限制，在检查约束合法性时会进行相应检查。单击该菜单，弹出如图 3-21 所示对话框，可进行如下操作：

- 单击 “” 按钮，选择对应 Net；
- 通过 “Type” 下拉列表，选择 “BUFG” “BUFG[0]~[7]” “BUFS” “LOCAL\_CLOCK”；
- 通过 “CE”、“CLK” 等复选框配置 Signal 类型，配置完成后，单击 “OK”，产生约束信息，显示在主界面底部的 “Clock Assignment” 约束编辑窗口中，在编辑窗口中，双击，重新打开约束对话框，进行编辑。

注！

当 Type 选择 LOCAL\_CLOCK 时 Signal 复选框为置灰不可配置状态。

图 3-21 时钟约束



## Select Dcs/Dqce

用于创建针对 DCS 和 DQCE 的象限约束，根据芯片的象限分布约束指定的 Instance 到具体的象限，如图 3-22 或图 3-23 所示。相关操作如下所示：

- 通过单击 “” 按钮，选择相应的 DCS/DQCE 原语；
- 通过 Position 下的复选框配置象限位置。
- 单击 “OK”，产生约束信息，显示在主界面底部的 “Quadrant Constraints” 约束编辑窗口中，在编辑窗口中，双击，重新打开约束对话框，可进行编辑修改。

注！

若设计无 DCS/DQCE 原语则无法添加。

图 3-22 象限约束 (GW1N 家族)

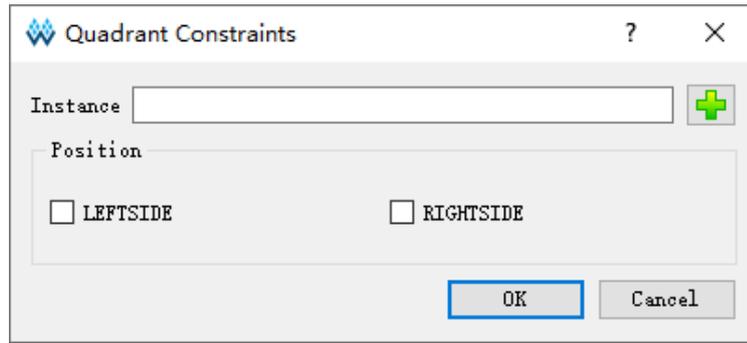
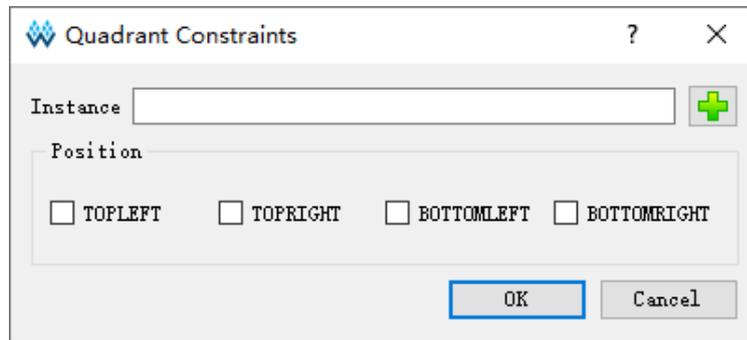


图 3-23 象限约束 (GW2A 家族)



### Select Hclk

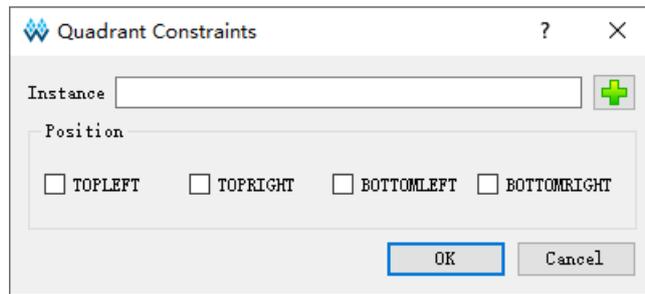
创建针对 HCLK 相关原语进行的约束，指定其约束在芯片上下左右的某些位置上，如图 3-24 所示。相关操作如下所示：

- 用户可通过单击“”按钮选择相应的原语；
- 通过 Position 下的复选框配置象限位置；
- 单击“OK”，产生约束信息，显示在主界面底部的“Hclk Constraints”约束编辑窗口中，在编辑窗口中，双击，重新打开约束对话框，可进行编辑修改。

#### 注！

- 若设计中无符合的原语则无法添加；
- Position 根据工程中 device 不同可用 Position 不同。

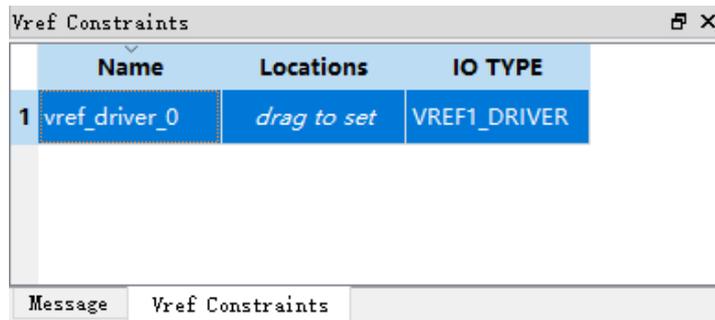
图 3-24 Hclk 约束



### Define Vref Driver

创建新的 Vref Driver，用于配置 IO Port 的 Vref 属性，单击该菜单，将直接在主界面底部的“Vref Constraints”约束编辑窗口中新建约束。如图 3-25 所示。

图 3-25 Vref 约束



注！

- 可通过拖拽方式指定 Vref 约束位置；
- 可通过双击修改 Vref 的名称。

### One Hit Drag

用于快速创建 Primitive、IO 等约束信息，创建步骤如下所示：

1. 在 Netlist 窗口中，选中 Primitive 或 port，如图 3-26 所示；
2. 在 Chip Array 窗口中，选中一个或多个位置信息如图 3-27 所示；
3. 单击“Tools > One Hit Drag”或按 F3 键直接产生约束信息，如图 3-28 所示。

注！

可通过“Ctrl”+鼠标左键选择矩形区域位置。

图 3-26 选择 One Hit drag 原语

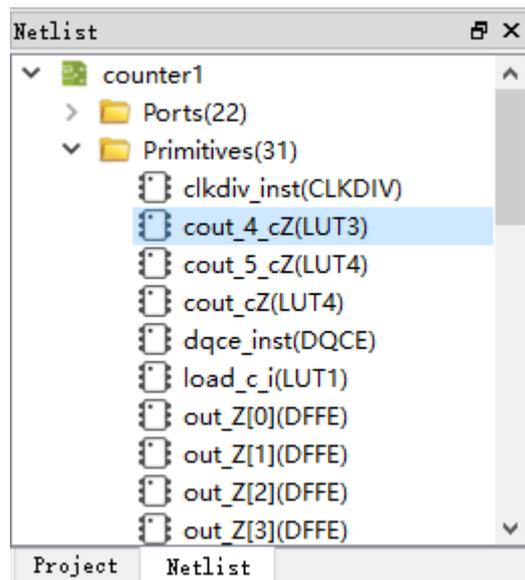


图 3-27 选择 One Hit Drag 位置

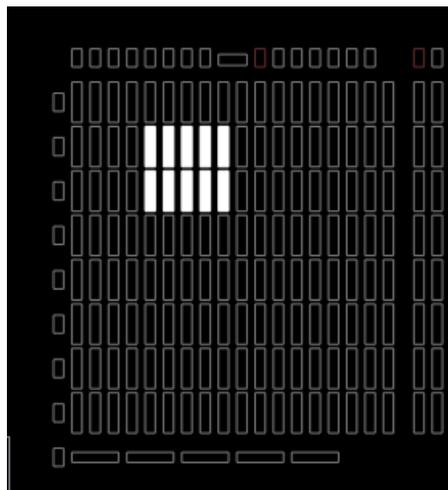


图 3-28 One Hit Drag 产生约束

	Primitive	Type	Locations	Exclusive
1	cout_4_cZ	LUT3	R[3:4]C[5:9]	False

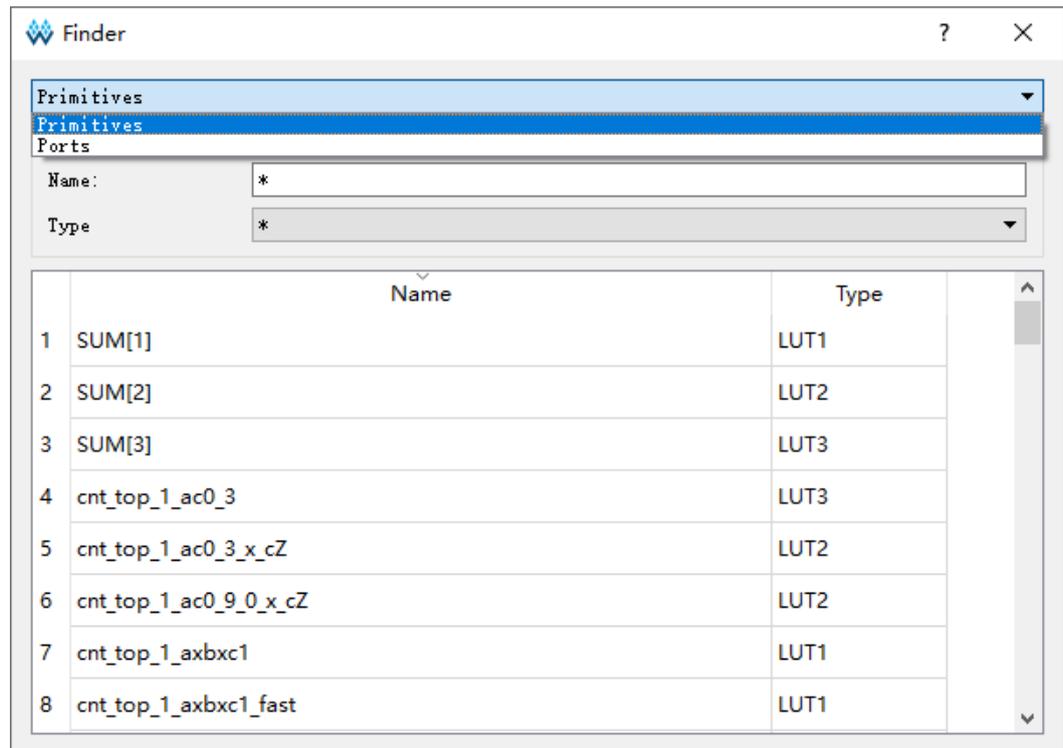
Message Primitive Constraints

## Finder

快速查找 Primitive、IO Port 信息，并可在相应的 primitive 或 port 上通过右键菜单编辑对应的约束信息，单击该菜单，弹出如图 3-29 所示对话框。相关操作如下所示：

- 通过选择“Primitives”或“Ports”，进行相应查找；
- 在对应的条目上，右键单击“Edit Primitive Constraint”，可在主界面底部的窗口中创建约束信息。

图 3-29 查找界面



## Update Constraints Using Posp

将约束位置信息更新为器件布局信息文件中的位置信息。

注：

在加载器件布局信息文件后 Update Constraints Using Posp 功能才能使用。

## View

View 菜单界面如图 3-30 所示，主要用于控制工具条、窗口的显示以及 Chip Array 和 Package View 两个窗口的放大、缩小等。各子菜单介绍如下：

- Toolbars: 用于控制工具栏快捷按钮的显示；
- Windows: 用于控制各个窗口的显示，如图 3-31 所示；
- Zoom In: 用于放大 Chip Array 视图或 Package View 视图；

- Zoom Out: 用于缩小 Chip Array 视图或 Package View 视图;
- Zoom Fit: 按照窗口大小缩放 Chip Array 视图或 Package View 视图。

图 3-30 View 菜单

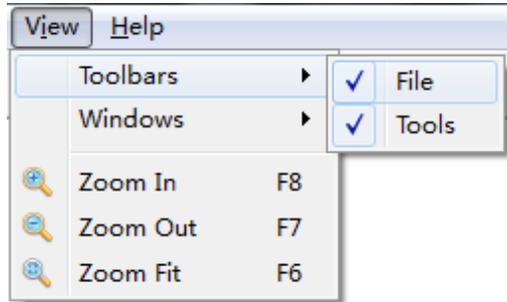
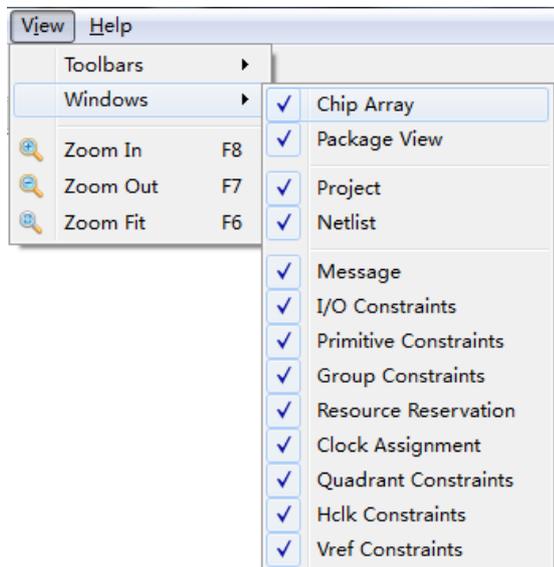


图 3-31 Windows 菜单



## Help

Help 菜单用于提示软件的版本号及版权信息。

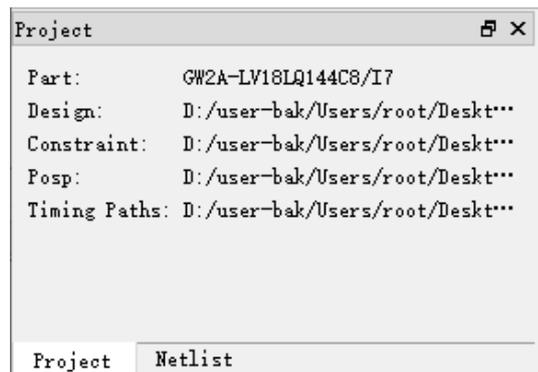
### 3.3.2 Project 和 Netlist 窗口

Project 和 Netlist 两个窗口可显示当前工程的 Device 信息、用户设计及约束文件的路径信息、Netlist 信息等。

#### Project 窗口

Project 窗口界面如图 3-32 所示。用于显示当前工程中所用的芯片信息，如 Part Number，以及用户输入的设计文件、约束文件、器件布局信息文件和时序路径文件信息。

图 3-32 Project 窗口



### Netlist 窗口

Netlist 窗口界面如图 3-33 所示，以树形结构显示用户设计中的 Ports、Primitives、Nets、Module 和 Timing paths 以及对应的数量信息。

注！

- Port、Primitive 等名称采用全路径方式进行显示，默认按字母升序排序；
- Port 和 Net 的显示采用 Bus 和非 Bus 相结合的显示方式，如图 3-34 所示；
- Module 采用层级的方式显示，各 Module 后可显示各 Module 中各类型的 Instance 数目，如图 3-35 所示；
- 时序路径则按照 Slack 时间从小到大的顺序显示，如图 3-36 所示。

图 3-33 Netlist 窗口

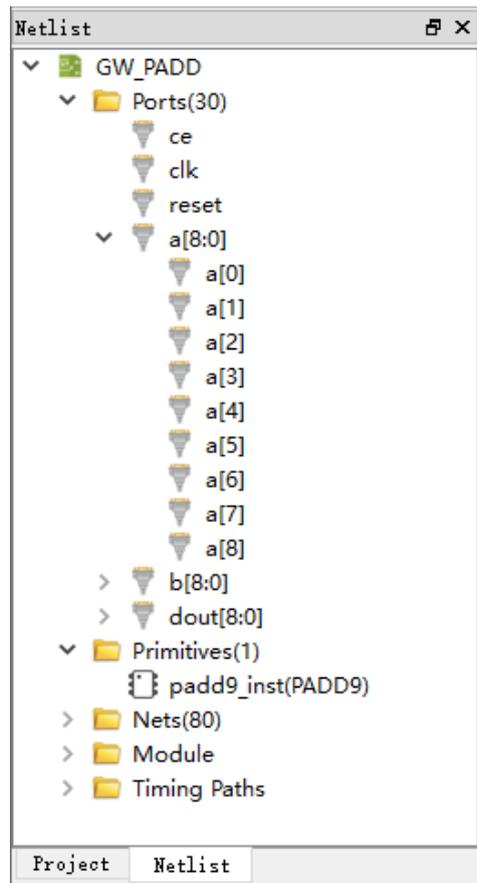


图 3-34 BUS 和非 BUS 结合显示

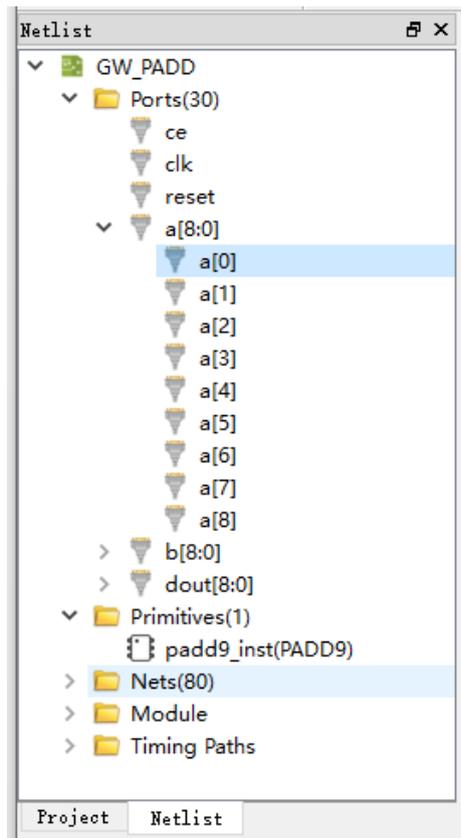


图 3-35 层级显示

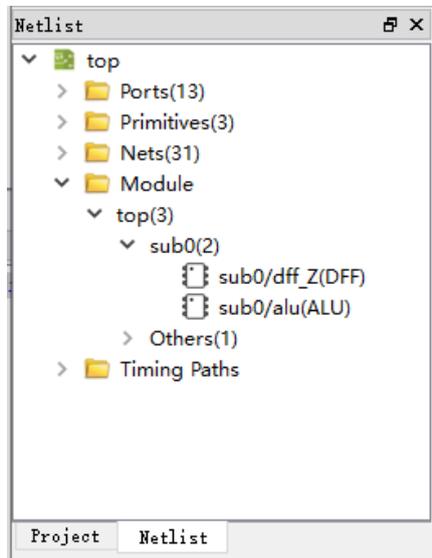
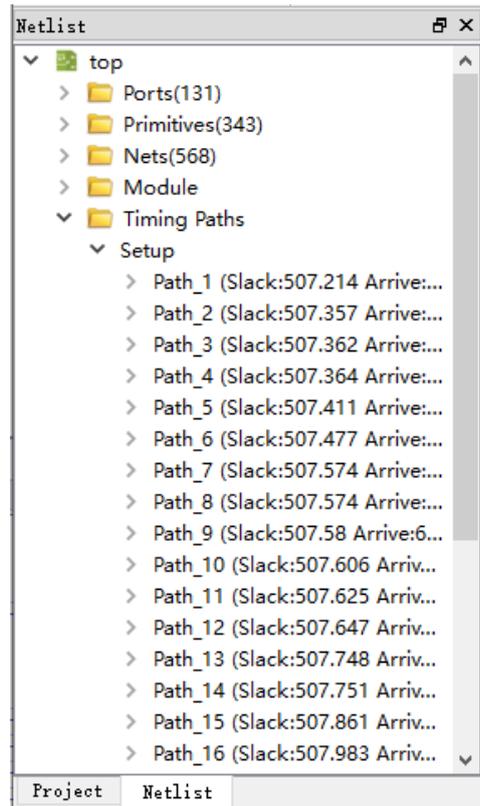


图 3-36 时序路径显示



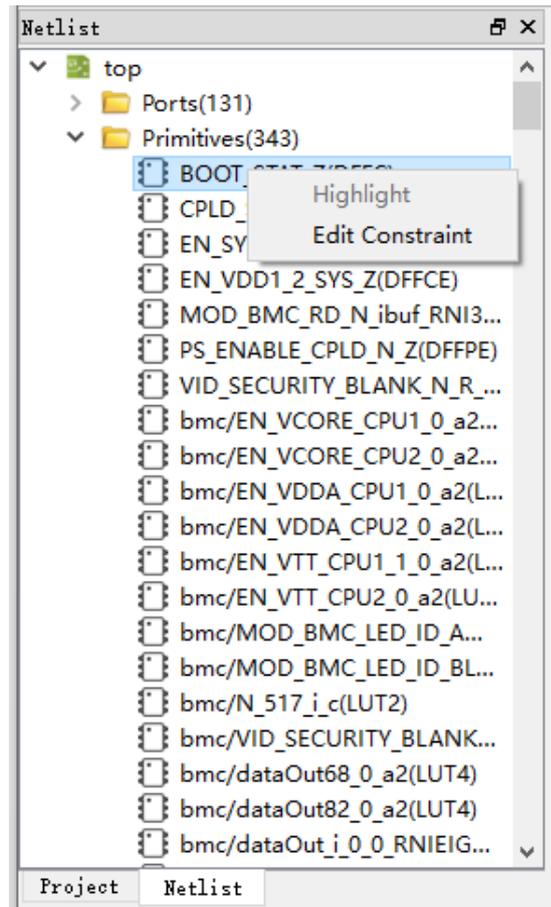
Netlist 窗口提供右键菜单功能，具有如下功能：

- **Highlight:** 可实现在 Chip Array 中高亮显示对应的约束位置；
- **Edit Constraint:** 编辑对应约束信息的功能。

注！

如当前 Primitive 或 Port 无位置约束，则高亮显示功能不可用，如图 3-37 所示。

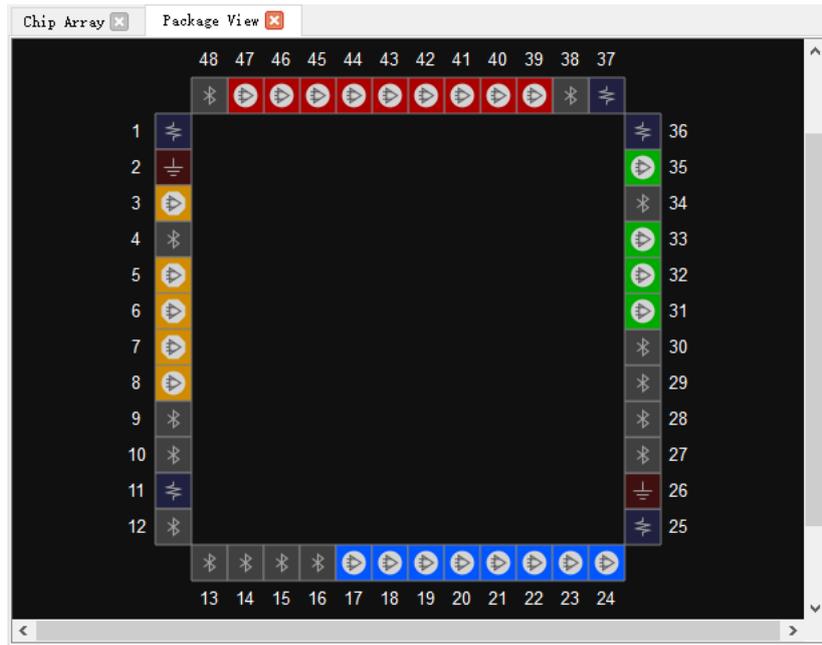
图 3-37 Netlist 右键功能



### 3.3.3 Package View 窗口

以 GW1NRF-4B-QFN48 为例, Package View 窗口界面如图 3-38 所示, 该窗口以芯片 package 信息为基础显示芯片的封装信息, 显示用户 I/O、电源、地等管脚。将鼠标放置某个位置上时, 会悬浮显示该位置的 I/O 信息, 包括 I/O 的 Type、Bank 以及 LVDS 信息等。

图 3-38 GW1NRF-4B-QFN48 Package view 界面



用户 I/O、电源、地管脚使用不同的符号和颜色来区分。不同 BANK 的 IO 引脚的颜色不同，图中管脚定义如下所示：

- “” 表示用户 I/O，填充颜色随 BANK 变化；
- “” 表示 VCC，填充颜色不变；
- “” 表示 VSS，填充颜色不变；
- “” 表示蓝牙接口，填充颜色不变。

Package View 支持右键菜单如图 3-39 所示，相关功能如下：

- Zoom In: 放大 Package View 视图；
- Zoom Out: 缩小 Package View 视图；
- Zoom Fit: 按照窗口大小缩放 Package View 视图；
- Show Differential IO Pairs: 显示差分对，如图 3-40 所示，红线相连的为 一对差分对；
- Top View: Package View 以顶部视图进行显示，默认以顶部视图进行显示，如图 3-41 所示为 GW1N-9-WLCSP64 封装的顶部视图；
- Bottom View: Package View 以底部视图进行显示，如图 3-42 所示为 GW1N-9-WLCSP64 封装的底部视图。

图 3-39 Package View 右键功能



注:

GW1N-9-WLCSP81M 的 Top View 和 Bottom View 和一般的 Top View、Bottom View 是相反的, 如图 3-43 和图 3-44 所示。

图 3-40 差分对显示

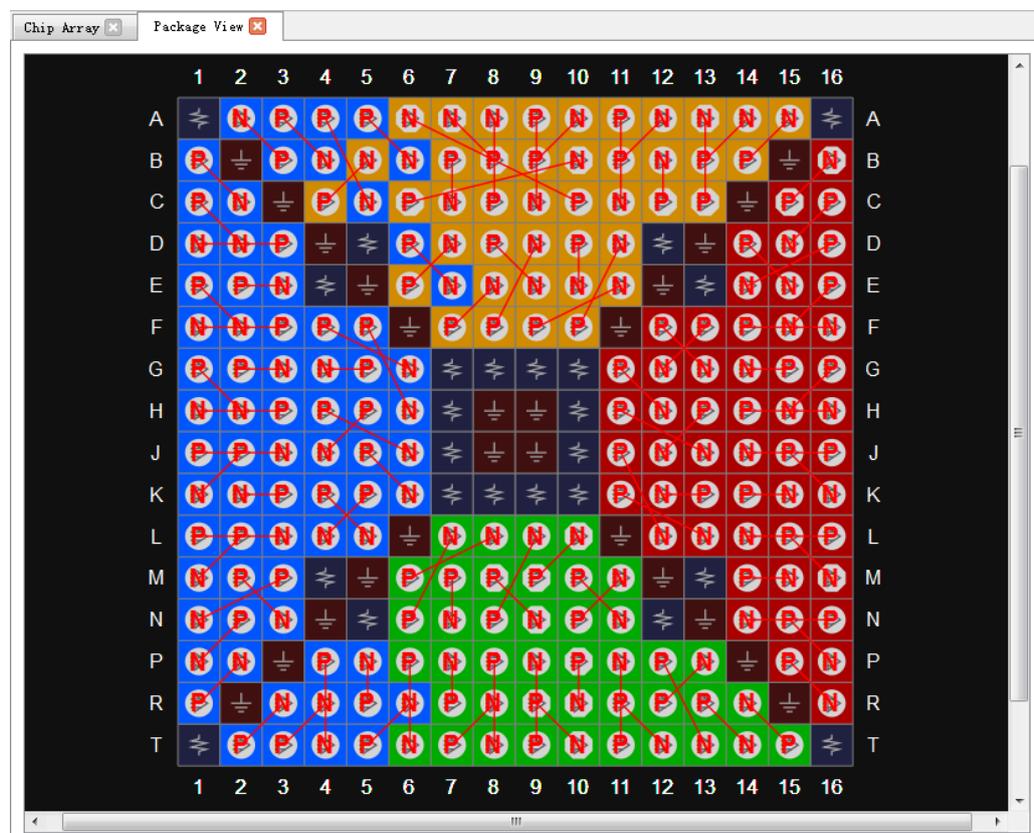


图 3-41 Top View

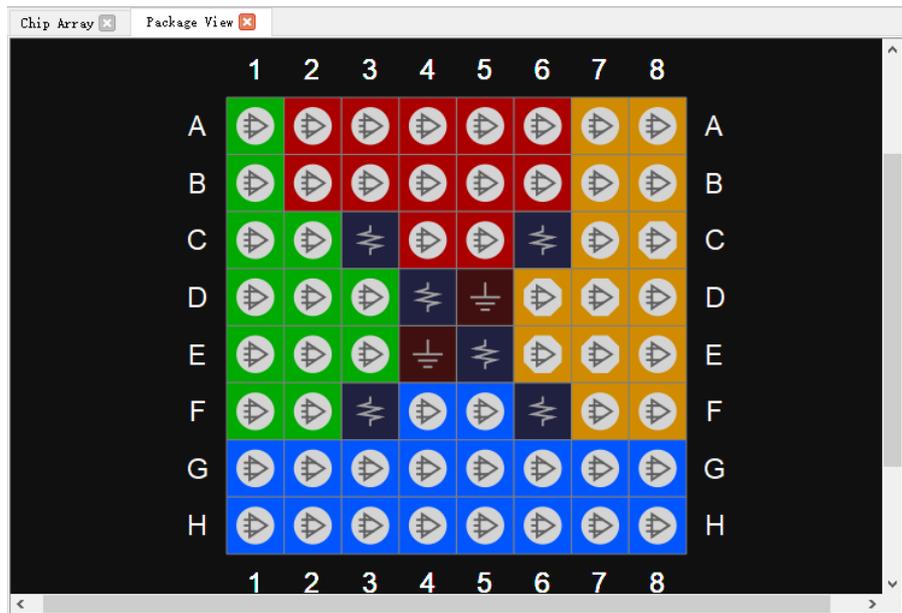


图 3-42 Bottom View

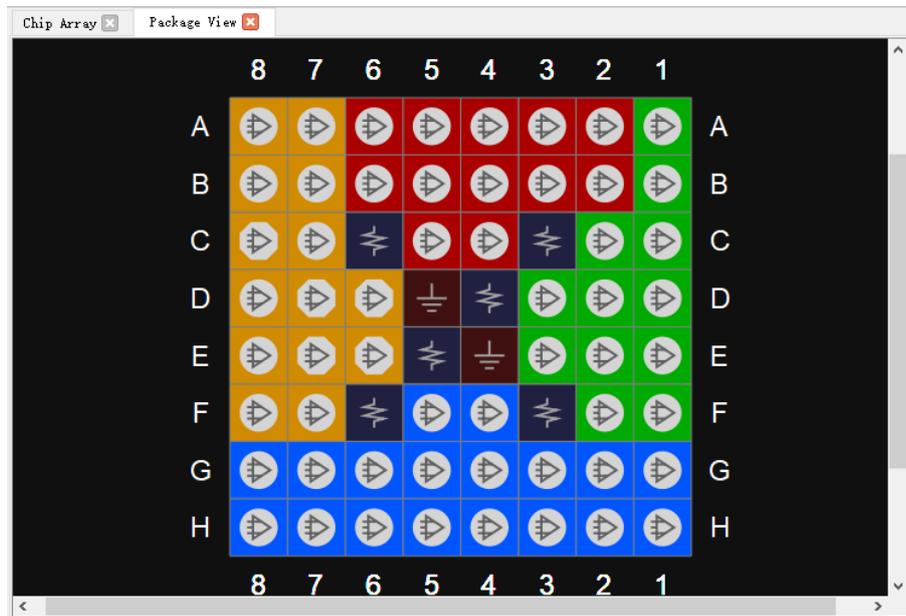


图 3-43 GW1N-9-WLCSP81M Top View

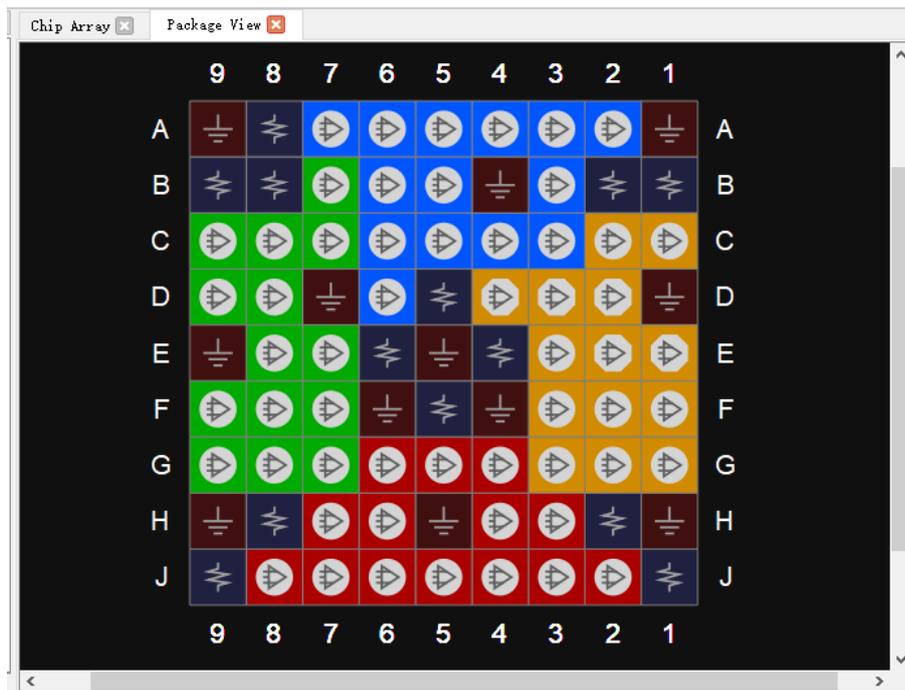
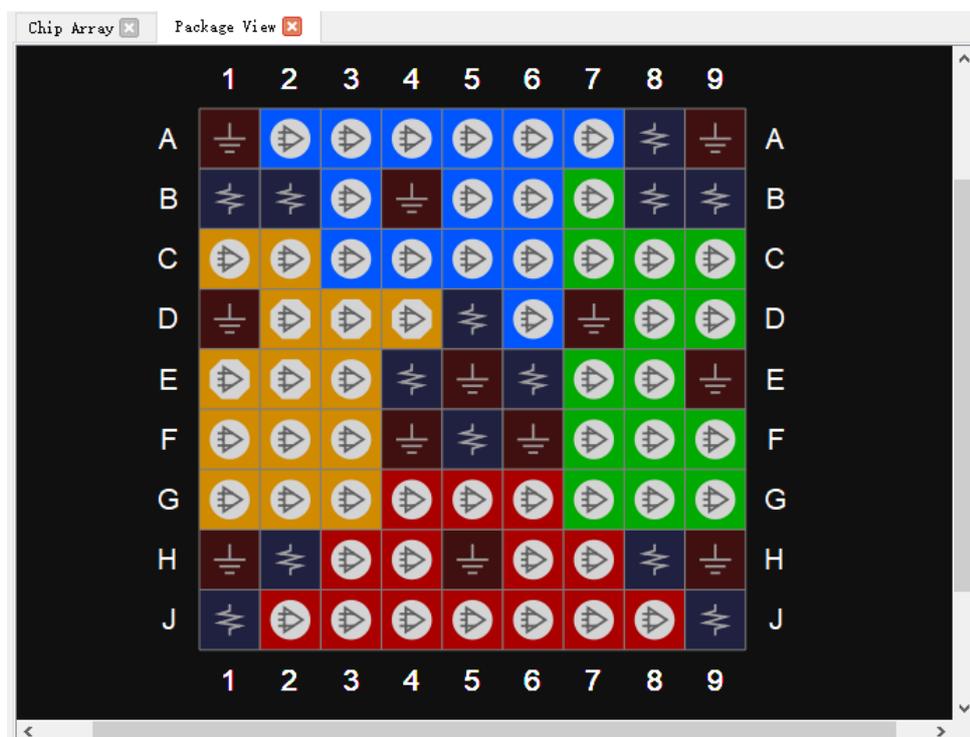


图 3-44 GW1N-9-WLCSP81M Bottom View



Package View 支持 IO Port 约束位置的显示, 可以通过从 Netlist 窗口或底部 I/O Constraints 窗口中将 IO Port 拖拽到 Package View 窗口来约束 IO Port 的位置。

注!

- 拖动时，鼠标会相应显示经拖动的 Port 名称;
- “”表示该位置不能放置拖动的 Port。

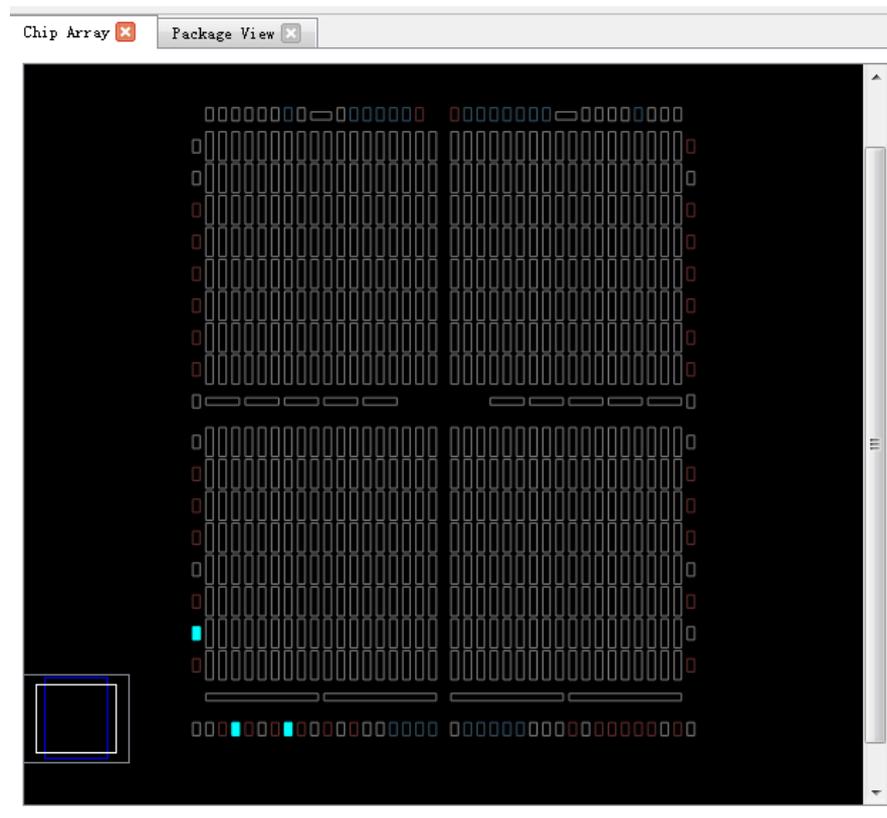
### 3.3.4 Chip Array 窗口

FloorPlanner 的 Chip Array 窗口界面如图 3-45 所示, Chip Array 窗口根据芯片的行列信息显示芯片的 IOB、CFU、DSP、PLL、BSRAM 等资源的分布, 实现对所有约束位置的实时显示, 支持放大、缩小、复制位置、悬停显示、拖拽等功能。

其中, IOB 是器件裸片的所有 IOB 位置, 且会以不同颜色区分 IOB:

- 白色: 该封装对应封装出的 IO 位置;
- 红色: 该封装下未封装的 IO 位置;
- 蓝色: 如果是 GW2AR-18、GW1NR-4、GW1NR-9、GW1NSR-2C 等 SIP 封装的器件, 则会有蓝色标记 IOB, 表示内嵌配置 IO 位置。

图 3-45 Chip Array 界面



Chip Array 分为网格模式、宏单元模式、原语模式三种显示模式。

- 网格模式: 以 GRID 为单位宏观显示约束位置, 如图 3-46 所示;

- 宏单元模式：以 CLS、BLOCK 等为单位显示约束位置，如图 3-47 所示；
- 原语模式：以 REG、LUT 等为单位显示约束位置，如图 3-48 所示。

图 3-46 网格模式约束

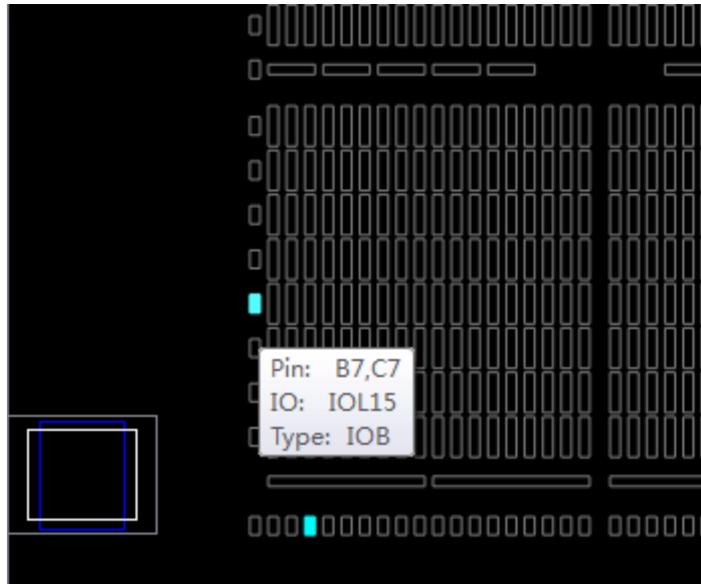


图 3-47 宏单元模式约束

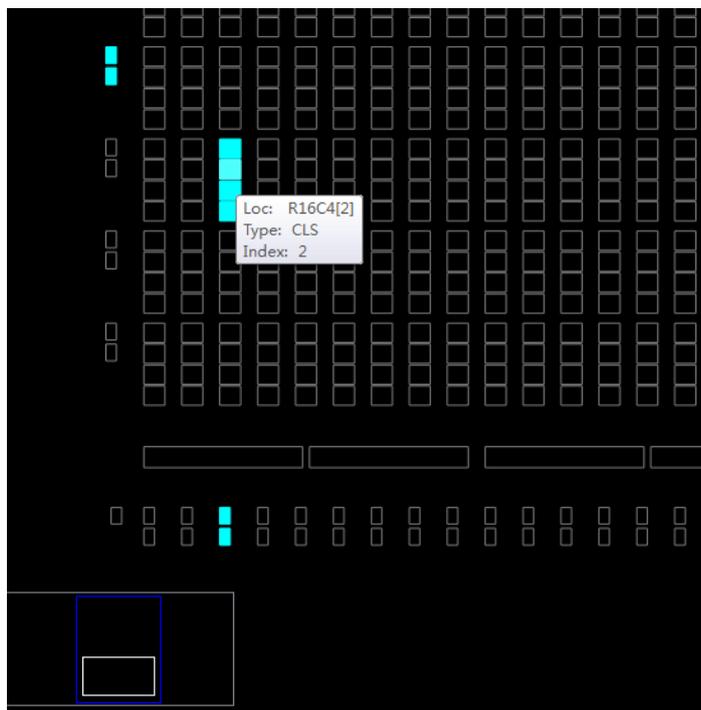
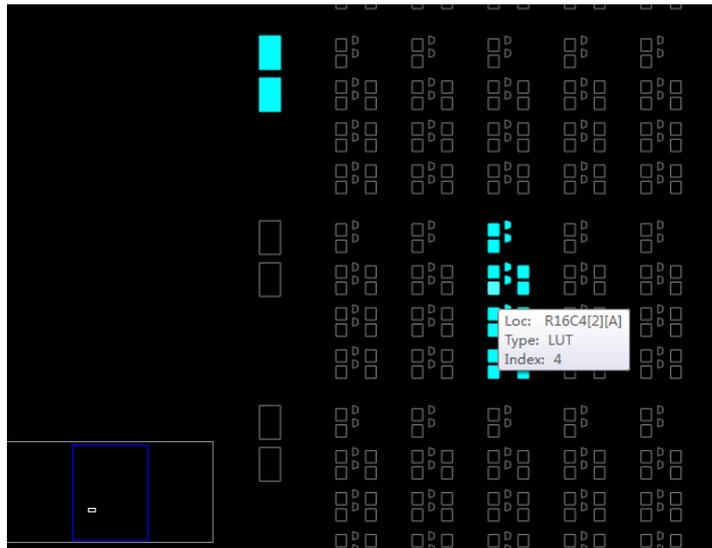


图 3-48 原语模式约束



Chip Array 支持以下拖拽功能：

- 从 Netlist 窗口拖拽到 Array 窗口，用于产生约束、指定约束位置；
- 从约束编辑窗口拖拽到 Array 窗口，用于指定约束位置。

Chip Array 窗口内置 chip 子窗口，用于实时显示当前视窗相对于整个芯片的位置，拖动 chip 子窗口中的白色框，Chip Array 的视窗将跟随移动。同时，Chip Array 窗口采用不同颜色区分约束类型、显示约束位置，各颜色的含义分别介绍如下：

- 白色：用于显示处于选择状态或正在高亮显示的约束位置；
- 深蓝色：用于显示预留约束的位置信息，表示该位置不能再被占用；
- 浅蓝色：用于显示 IO 和 Primitive 约束在某个 grid 或 range 内，界面上显示为浅蓝色。

Chip Array 窗口支持右键菜单，具有如下功能：

- Zoom In: 放大 Chip Array 视图；
- Zoom Out: 缩小 Chip Array 视图；
- Zoom Fit: 按照窗口大小缩放 Chip Array 视图；
- Show Constraints View: 显示 Chip Array 的 instance 约束视图；
- Show Place View: 显示 Chip Array 的 instance 放置视图；
- Show Multi-View: 同时显示 Chip Array 的 instance 约束和放置的复合视图；
- Show In-Out Connection: 在 Place View 中显示和选中 instance 输入输出连接的 instance 位置；
- Show In Connection: 在 Place View 中显示和选中 instance 输入连接的

instance 位置;

- **Show Out Connection:** 在 Place View 中显示和选中 instance 输出连接的 instance 位置
- **Convert Place To Constraints:** 在 Place View 中把选中 instance 的放置信息转化成约束信息;
- **Unhighlight All:** 对于选中变亮的位置或者区域消除高亮;
- **Copy Location:** 复制选中的位置或者区域。

在 Show Place View 中, 可对 Lut、Reg 的密度进行显示, 如图 3-50 所示, 详情如下:

- **ALL Instance,** 显示所有 Instance 的 place 情况, 5 个以内呈淡绿色、6-10 个呈绿色、10 个以上呈深绿色;
- **Only Lut,** 只显示所有 Lut 的 place 情况, 2 个以内呈淡绿色、3-4 个呈绿色、4 个以上呈深绿色;
- **Only Dff,** 只显示所有 Reg 的 place 情况, 2 个以内呈淡绿色、3-4 个呈绿色、4 个以上呈深绿色。

注!

- Show Place View 和 Show Multi-View 只有在加载了器件布局信息文件后才能使用, 否则置灰;
- Show In-Out Connection、Show In Connection、Show Out Connection、Convert Place To Constraints 只有在 Chip Array 窗口是 Show Place View->All Instance 视图时选中某个 instance 才能够使用, 否则置灰;
- 若视窗中有 grid、block、reg、lut 等处于选中状态, 则右键菜单中的“Copy Location”功能可用;
- 若无 grid 等被选中, “Copy Location”功能不可用, 如图 3-49 所示;
- 通过“Ctrl”键+鼠标左键拖动, 可选取区域, 单击鼠标右键, 选择“Copy Location”, 可复制所选区域的位置信息, 复制的位置可直接粘贴到任意约束编辑窗口中。

图 3-49 Chip Array 右键功能

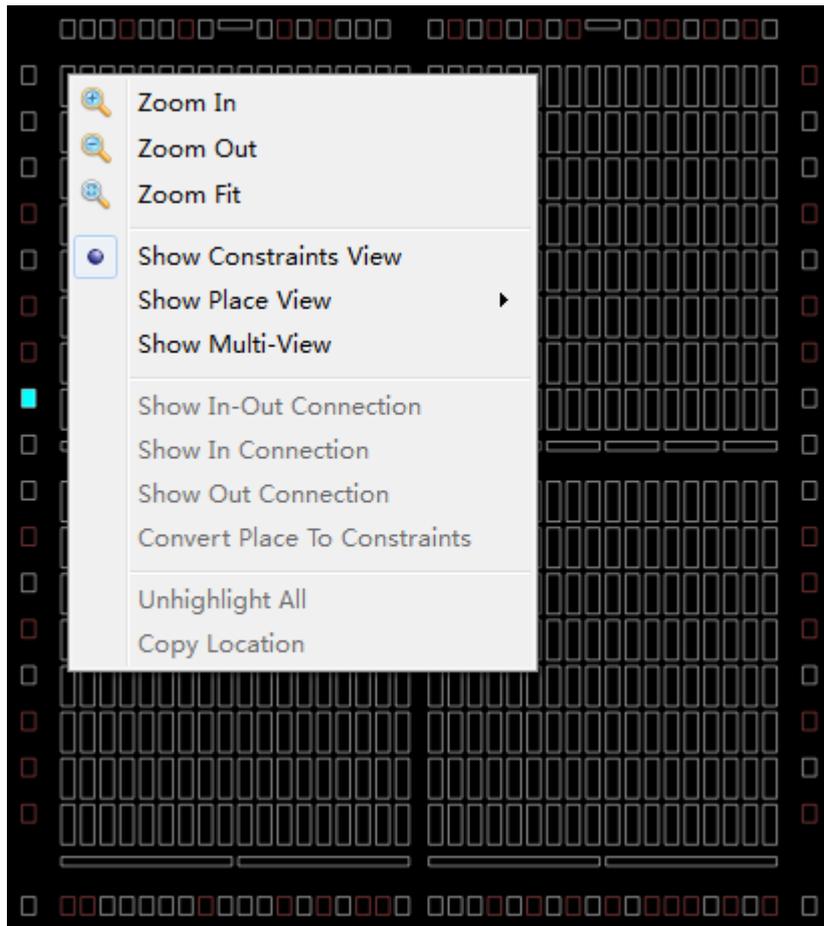
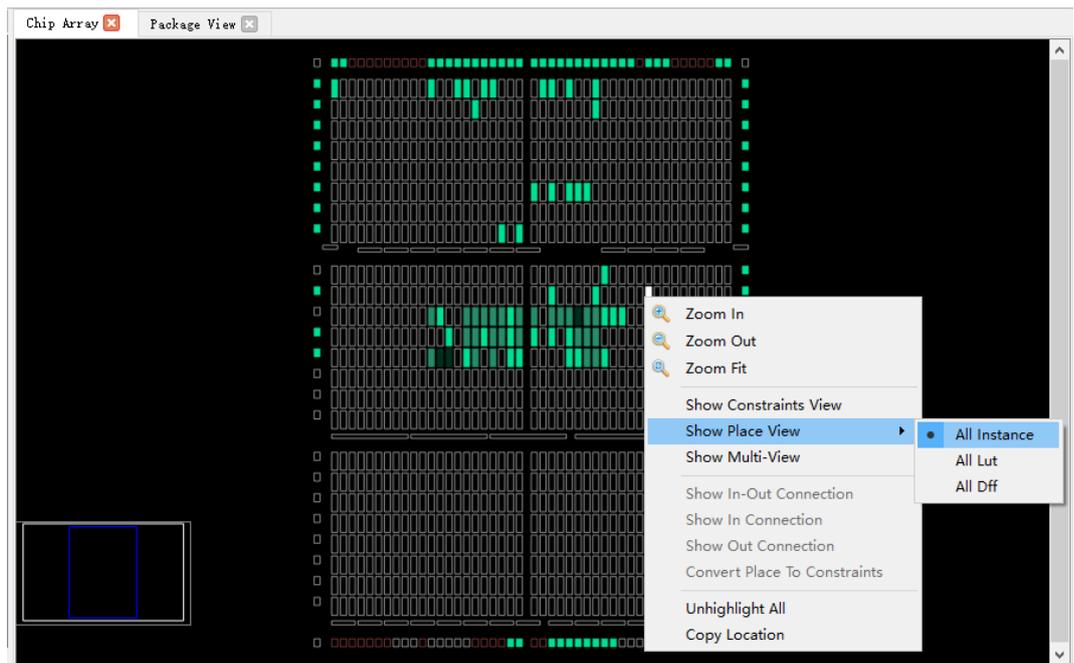
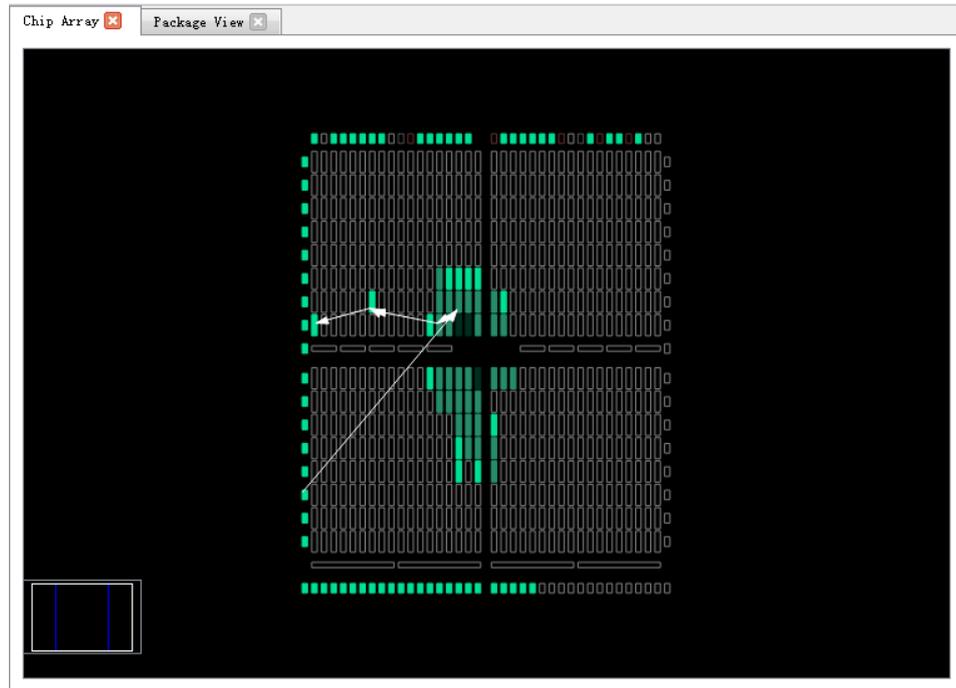


图 3-50 Show Place View 显示



另外，Chip Array 窗口还可对时序路径进行高亮显示，如图 3-51 所示。

图 3-51 时序路径高亮显示



### 3.3.5 Constraint 编辑窗口

Constraint 编辑窗口包含“I/O Constraints”、“Primitive Constraints”、“Group Constraints”等 8 个编辑窗口，用于显示各约束的详细信息，并提供约束编辑功能和位置拖拽功能，各窗口分别介绍如下：

#### I/O Constraints

I/O Constraints 是对设计的 port 进行管脚约束。I/O 约束窗口如图 3-52 所示，各功能如下：

- 显示用户设计中所有 IO Port 的属性及约束信息，如 Port 的 Direction、Bank、IOType、PullMode 等；
- 提供约束位置、属性等编辑功能；
- 可通过拖拽、双击等方式改变约束信息。

注！

- I/O 的位置可以通过拖拽的方式进行设置，也可以双击输入；
- 在拖拽 IO 过程中会显示所拖拽的 IO 名称；
- 将 IO 拖拽至 Chip Array 窗口中时，可放置的位置变亮，不可放置的位置颜色亮度不变；
- 将 IO 拖拽至 Package View 窗口中时，可放置位置亮度不变，不可放置位置变暗；
- 设置完成后，在 Chip Array 窗口中约束的位置变为浅蓝色高亮，在 Package View 窗口中约束的位置变为橙色高亮。

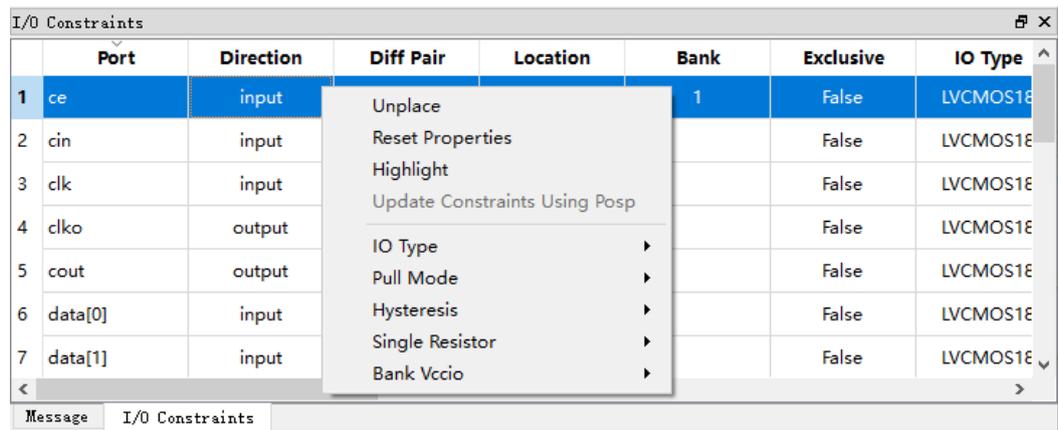
窗口提供右键菜单功能，详情如下：

- Uplace: 取消放置；
- Reset Properties: 复位 Port 属性设置；
- Highlight: 高亮显示约束位置；
- Update Constraints Using Posp: 按照器件布局信息更新约束；
- IO Type: 设置电平标准；
- Drive: 设置驱动电压；
- Pull Mode: 设置上拉模式；
- PCI Clamp: 设置 PCI 协议的开关；
- Hysteresis: 设置迟滞量；
- Open Drain: 设置开漏电路的开关；
- Slew Rate: 设置电压转换速率；
- Vref: 设置外部参考电压；
- Single Resistor: 设置单端电阻的开关；
- Diff Resistor: 设置差分电阻的开关；
- Bank Vccio: 设置 BANK 电压。

注！

- 右键菜单支持用户批量修改 Port 属性的功能，用户可选择多个 Port，若多个 Port 有相同的属性值可配置，则通过右键菜单可统一进行配置；
- Update Constraints Using Posp 只有在加载了器件布局信息文件后才能使用，否则置灰。

图 3-52 I/O 约束窗口



## Primitive Constraints

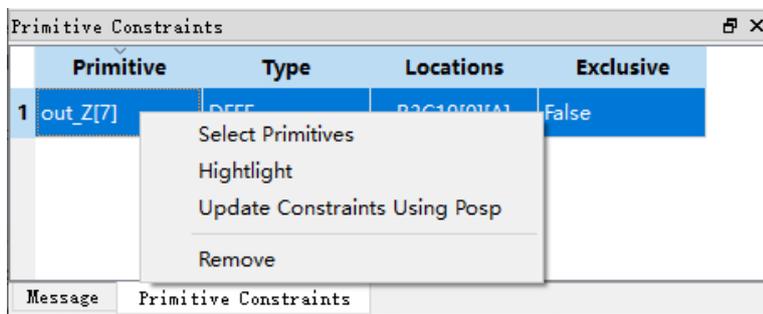
Primitive Constraints 是约束设计中原语的位置，原语约束窗口如图 3-53 所示，功能如下：

- 用于显示当前所有 **Primitive** 约束的名称、类型、位置以及 **Exclusive** 信息；
- 提供编辑功能。

注！

- 可通过拖拽的方式或双击输入的方式修改位置信息；
- 可通过双击设置 **Exclusive** 属性；
- 该窗口提供右键菜单功能，用于提供高亮显示约束位置、删除、添加、按照器件布局信息更新约束。
- 在 **Primitive** 约束位置进行手动输入时，会对位置进行语法检查及合法性检查，错误提示对话框如图 3-16 和图 3-17 所示。

图 3-53 原语约束窗口

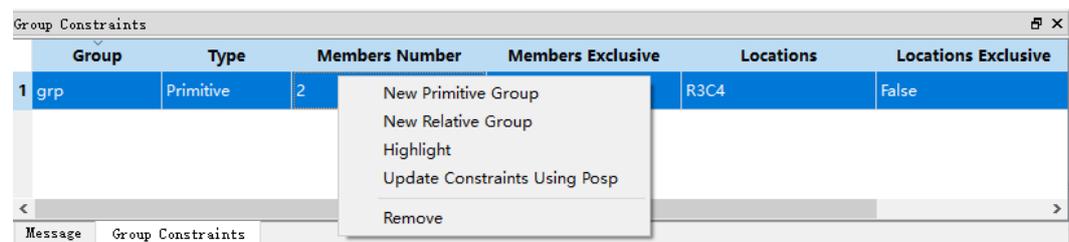


## Group Constraints

**Group Constraints** 是对设计中的 **BUF** 和原语进行组约束，组约束窗口如图 3-54 所示，功能如下：

- 用于显示当前所有 **Group** 约束的名称、类型、包含的 **Primitive** 个数、位置以及 **Exclusive** 信息，包含 **Primitive** 和 **Relative** 两种 **Group** 的显示；如图 3-15 和图 3-19 所示，双击对应的 **Group** 条目，打开对话框，可实现约束信息的编辑修改功能；
- 该窗口提供右键菜单功能，用于提供高亮显示约束位置、删除、添加、按照器件布局信息更新约束的功能。

图 3-54 组约束窗口



## Resource Reservation

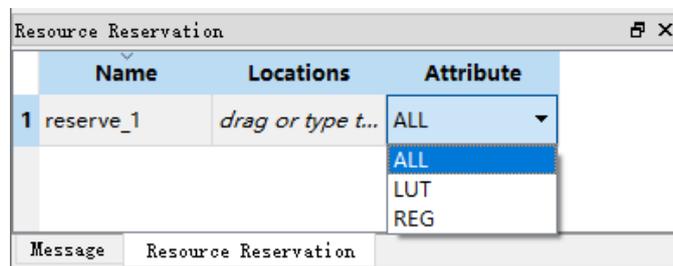
Resource Reservation 是对当前封装中的可用资源进行预留约束，预留约束窗口如图 3-55 所示，功能如下：

- 用于显示当前所有预留约束的位置信息；
- 该窗口提供右键菜单功能，用于提供高亮显示约束位置、删除、添加约束的功能；
- Name 属性用于区分各使用率约束，用户不能进行修改。

注！

可通过拖拽或双击输入修改位置信息。

图 3-55 预留约束窗口



## Clock Assignment

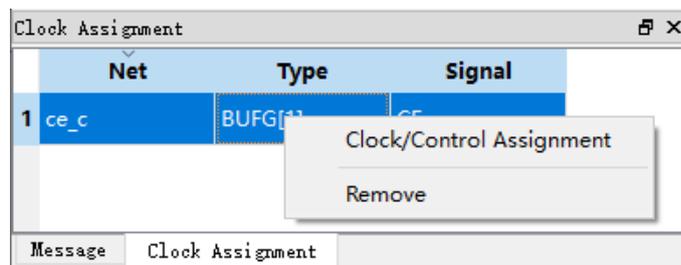
Clock Assignment 是对设计中的 net 进行时钟约束，时钟约束窗口如图 3-56 所示，功能如下：

- 用于显示当前所有 Clock 约束的相关信息；
- 该窗口提供右键菜单功能，用于提供添加、删除 Clock 约束的功能。

注！

- 双击约束可进行编辑修改；
- Clock 约束无位置信息，不支持拖拽功能。
- 新建时钟约束的窗口如图 3-21 所示。

图 3-56 时钟约束窗口



## Quadrant Constraints

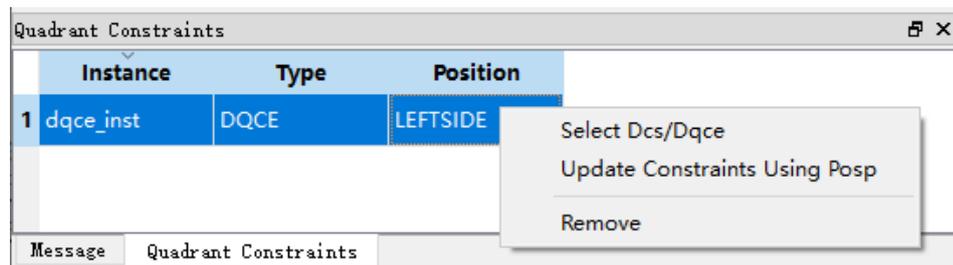
Quadrant Constraints 是对设计中的 DCE/DQCE 进行象限约束，象限约束窗口如图 3-57 所示，功能如下：

- 用于显示所有的象限约束，包括 Instance 名称、类型以及象限位置；
- 窗口支持右键菜单功能，用于添加新的象限约束和删除已有约束。

注！

- 象限约束只针对 DCS 和 DQCE 两种原语有效。
- 新建象限约束的窗口如图 3-22 和图 3-23 所示。

图 3-57 象限约束窗口

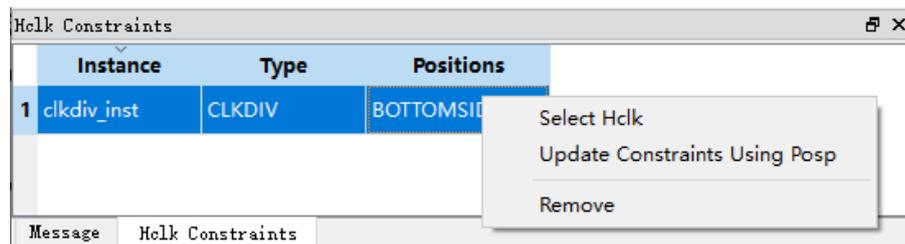


## Hclk Constraints

Hclk Constraints 是对设计中的 CLKDIV/DLLDLY 进行 Hclk 约束，Hclk 约束窗口如图 3-58 所示，功能如下：

- 用于显示针对 Hclk 相关的 Instance 的位置约束，包括 Instance 名称、类型以及象限位置；
- 窗口支持右键菜单功能，用于添加新的象限约束和删除已有约束。新建 Hclk 约束的窗口如图 3-24 所示。

图 3-58 Hclk 约束窗口



注！

Hclk 约束只针对 CLKDIV 和 DLLDLY 两种原语有效。

## Vref Constraints

Vref Constrains 是约束所在 bank 的外部参考电压，Vref 约束窗口如图 3-59 所示，功能如下：

- 用于显示用户自定义的 Vref Driver 信息，用户可自定义 Vref 的名称、位

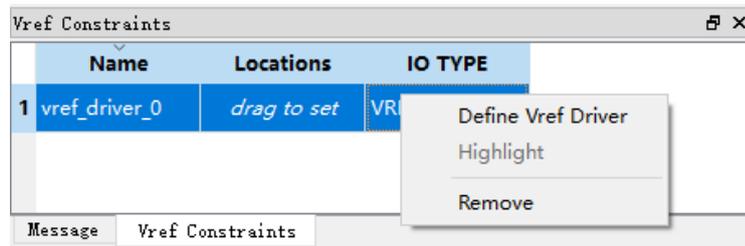
置信息；

- 窗口支持右键菜单功能，用于高亮显示约束位置、添加、删除约束信息。

注！

位置信息只能通过拖拽的方式进行设置。

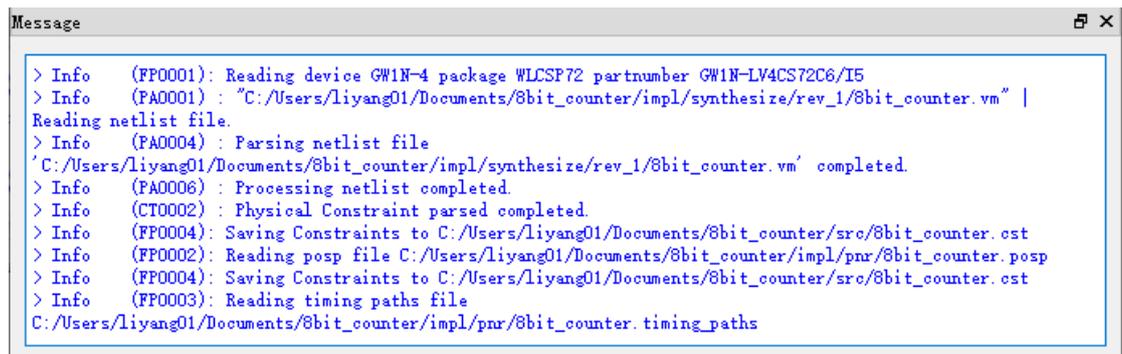
图 3-59 Vref 约束窗口



## 3.4 Message 窗口

Message 窗口如图 3-60 所示，窗口提供输出结果的显示。

图 3-60 Message 窗口



# 4 创建 Constraints

Constraints 编辑器支持 IO、Primitive、Group、Resource、Clock、Quadrant、Hclk、Vref 等 Constraints 的创建。可通过 Tools 菜单创建 Constraints，详情请参考 [3.3.1](#) 菜单栏。

注！

亦可通过其他方式创建 Constraints，本节主要以拖拽的方式为例，介绍如何通过拖拽生成 Constraint。

## 4.1 创建 Constraints 示例

以用户设计 counter.v 为例，演示如何创建 Constraints：

```
// Eight bit counter example 1

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

reg [7:0] out;
```

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

end

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

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

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

DQCE dqce_inst (
    .CLKOUT(clko),
    .CLKIN(clkout),
```

```
.CE(ce)
);

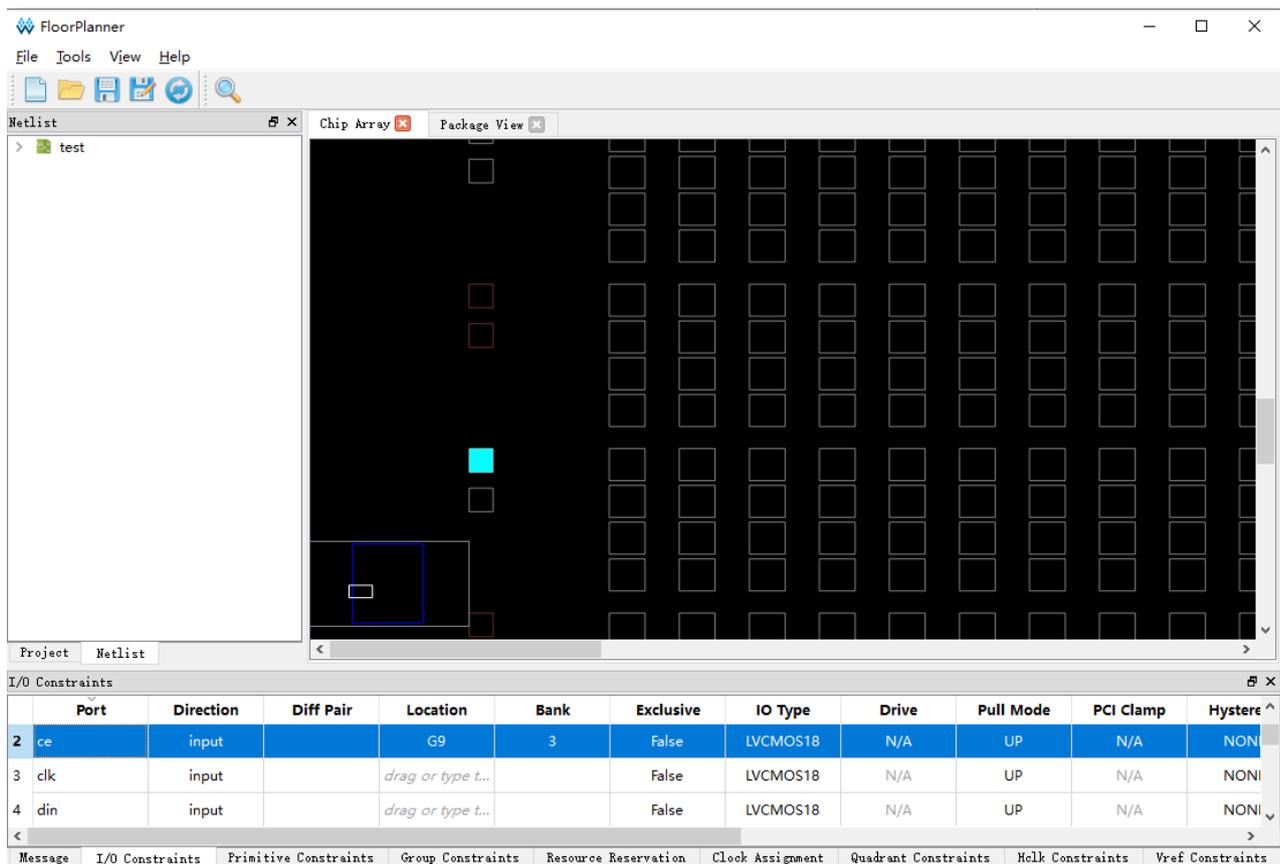
Endmodule
```

## 4.2 I/O Constraints 创建

拖至 Chip Array 创建 I/O Constraints，步骤如下：

1. 点击至 IO Constraints 编辑窗口，放大 Chip Array 窗口至宏单元模式；
2. 选中 Port “ce” 拖拽至 Chip Array 窗口中的 G9 位置，如图 4-1 所示；
3. Port “ce” 的 Location 信息显示为 G9。

图 4-1 拖拽到 Chip Array 创建 I/O Constraints

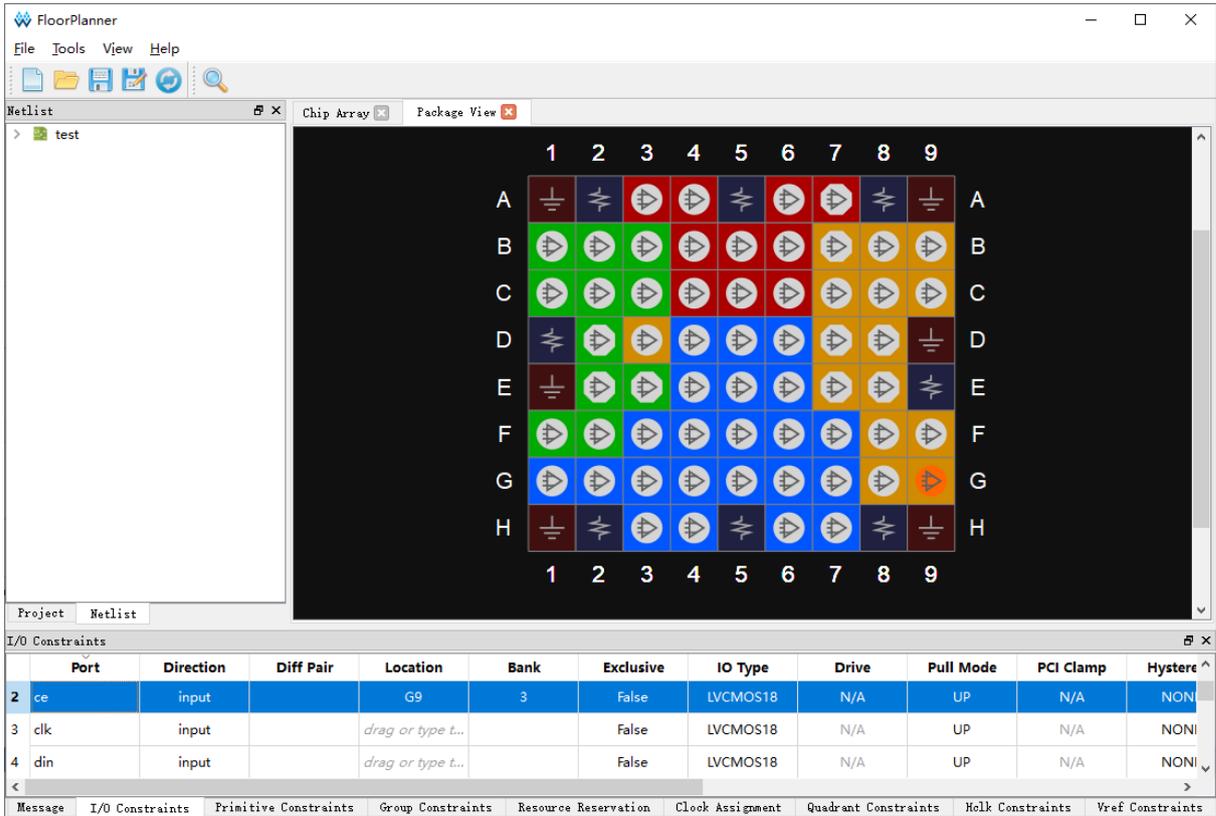


拖至 Package View 创建 I/O Constraints，步骤如下：

1. 点击至 IO Constraints 编辑窗口；
2. 选中 Port “ce” 拖拽至 Package View 窗口中的 G9 位置，如图 4-2 所示；

3. Port “ce” 的 Location 信息显示为 G9。

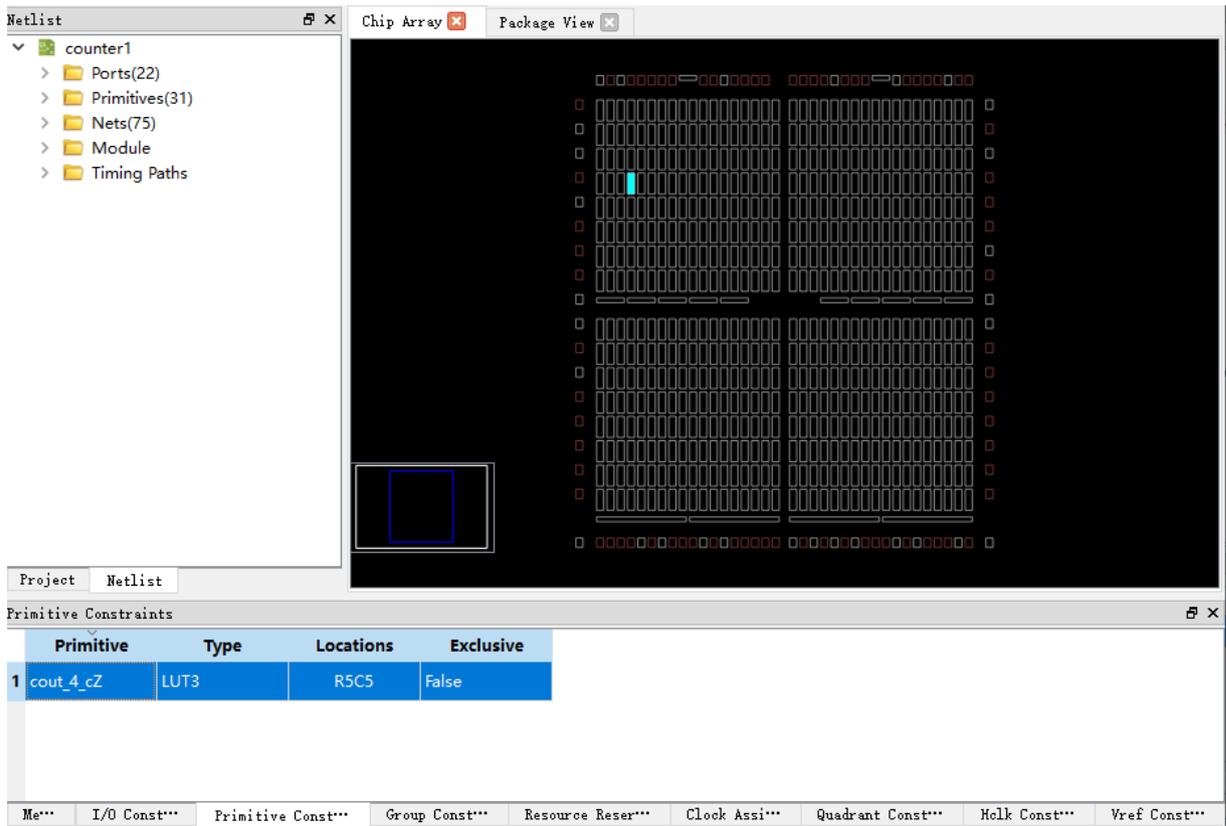
图 4-2 拖至 Package View 创建 I/O Constraints



## 4.3 Primitive Constraints 创建

1. 在 Primitive Constraints 编辑窗口，右键单击菜单栏，选择 “Select Primitives”，弹出 Primitive Finder 对话框后，选择 Primitive “cout\_4\_cZ”，点击 “OK”；
2. 选中新创建的 primitive 约束，拖拽至 Chip Array 窗口中的 R5C5 位置，如图 4-3 所示；
3. Primitive “cout\_4\_cZ” 的 Location 信息显示为 R5C5。

图 4-3 拖拽到 Chip Array 创建 Primitive Constraints



## 4.4 Group Constraints 创建

如图 4-4 所示，在 Group Constraints 编辑器中，可通过右键单击菜单，创建 Primitive Group 和 Relative Group。

图 4-4 Group Constraints 编辑器右键菜单



### 4.4.1 Primitive Group Constraints 创建

1. 在 Group Constraints 编辑窗口中，右键单击菜单，点击“New Primitive Group”，弹出 Edit Primitive Group 对话框；
2. 输入 Group Name “grp1”，点击“+” 弹出 Primitive Finder 对话框；
3. 选取所要设置的 Primitive “cout\_5\_cZ”、“cout\_cZ”，点击“OK”，加入 Members 列表；

4. 在 Locations 输入所要约束的位置 R9C7，如图 4-5 所示；
5. 在 Edit Primitive Group 对话框中点击“OK”，可创建一条 Primitive Group Constraints，如图 4-6 所示。

图 4-5 创建 Primitive Group Constraints

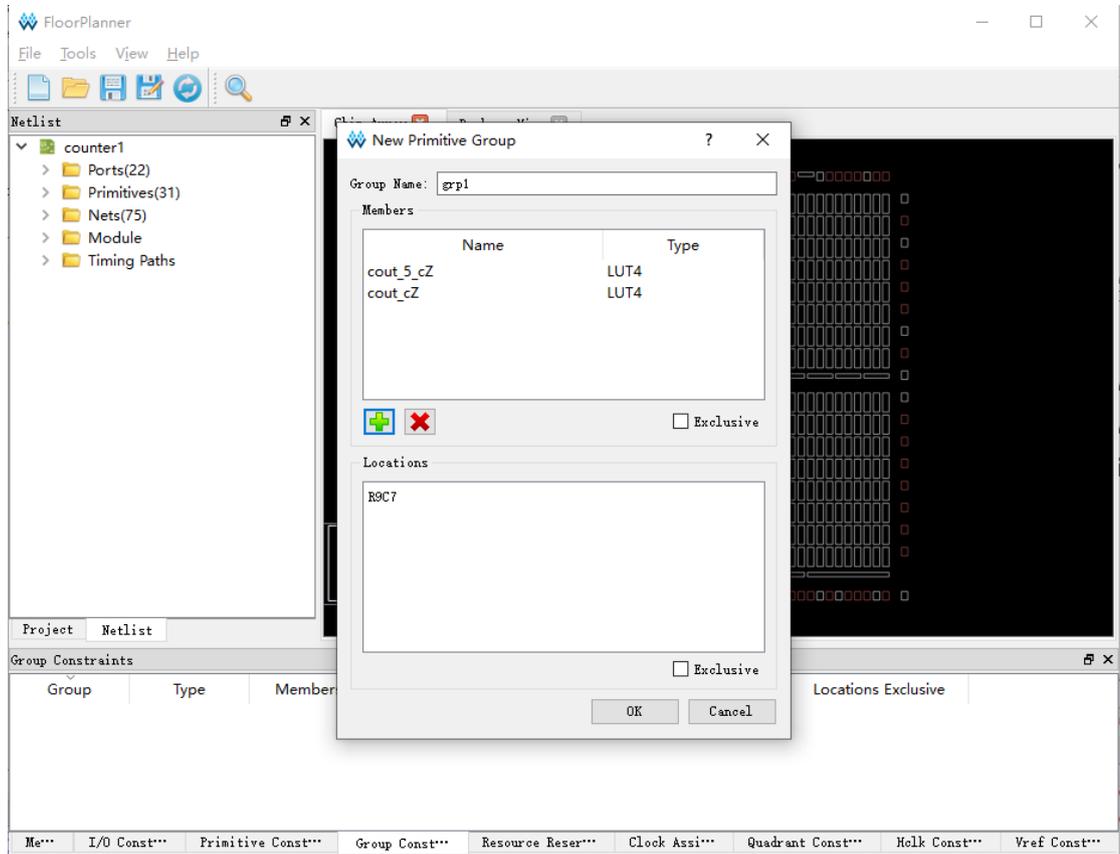
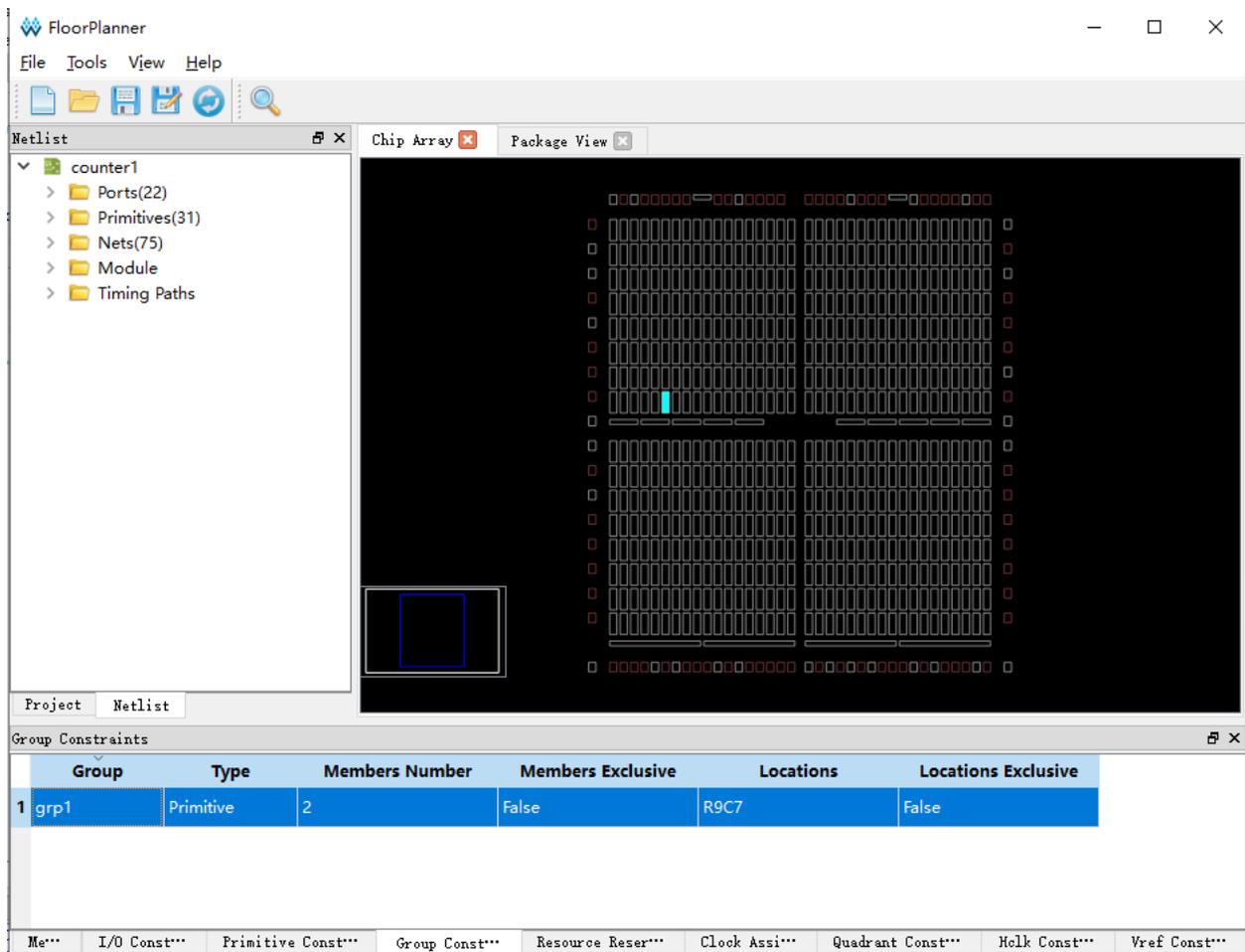


图 4-6 Primitive Group Constraints



注！

Primitive Group Constraints 的 Location 信息只能通过手动输入或者从 Chip Array 窗口中复制，不能通过拖拽实现。

## 4.4.2 Relative Group Constraints 创建

1. 在 Group Constraints 编辑窗口中，右键单击菜单，点击“New Relative Group”，弹出 Edit Relative Group 对话框；
2. 输入 Group 的名字“rel\_grp”，点击“+”，弹出 Primitive Finder 对话框；
3. 在 Primitive Finder 对话框中选择所要设置的 Primitives “load\_c\_i”、“out\_Z[0]”，选择“OK”，添至 Member 列表中；
4. 为每个 Primitive 添加相对位置“R0C0”、“R4C5”，如图 4-7 所示；
5. 在 Edit Relative Group 对话框中，选择“OK”创建 Relative Group Constraints，如图 4-8 所示。

图 4-7 Relative Group Constraints 创建

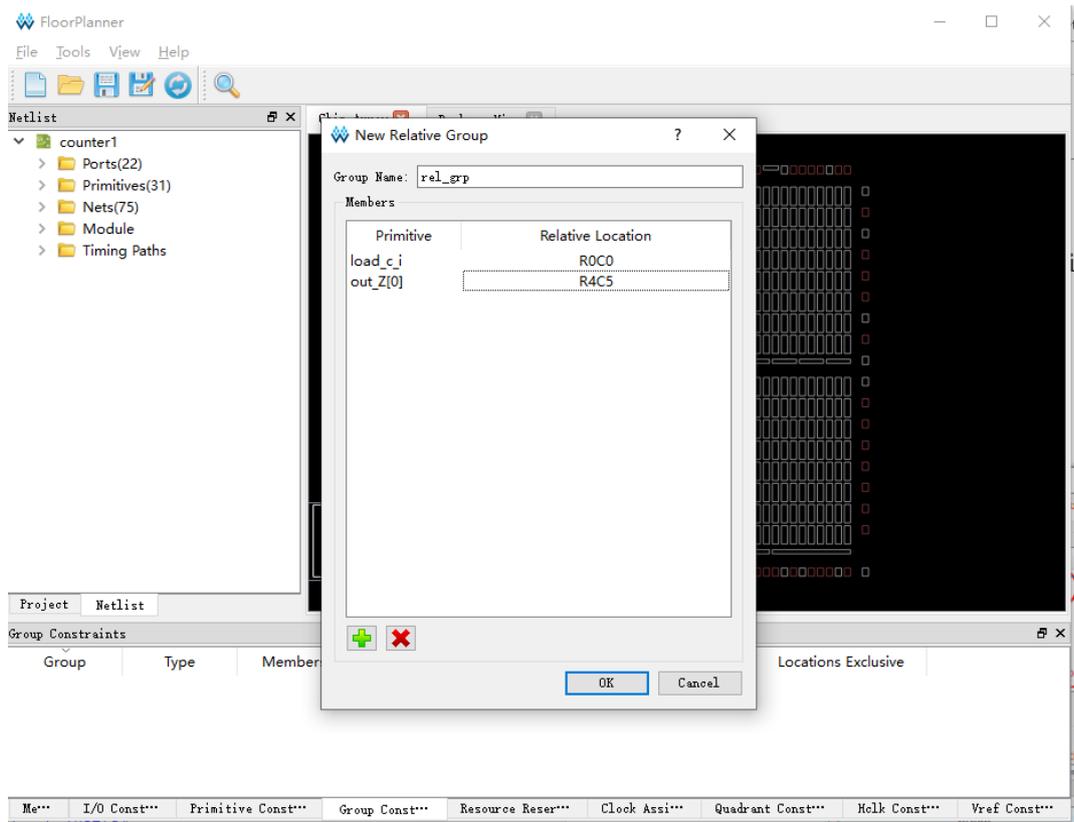
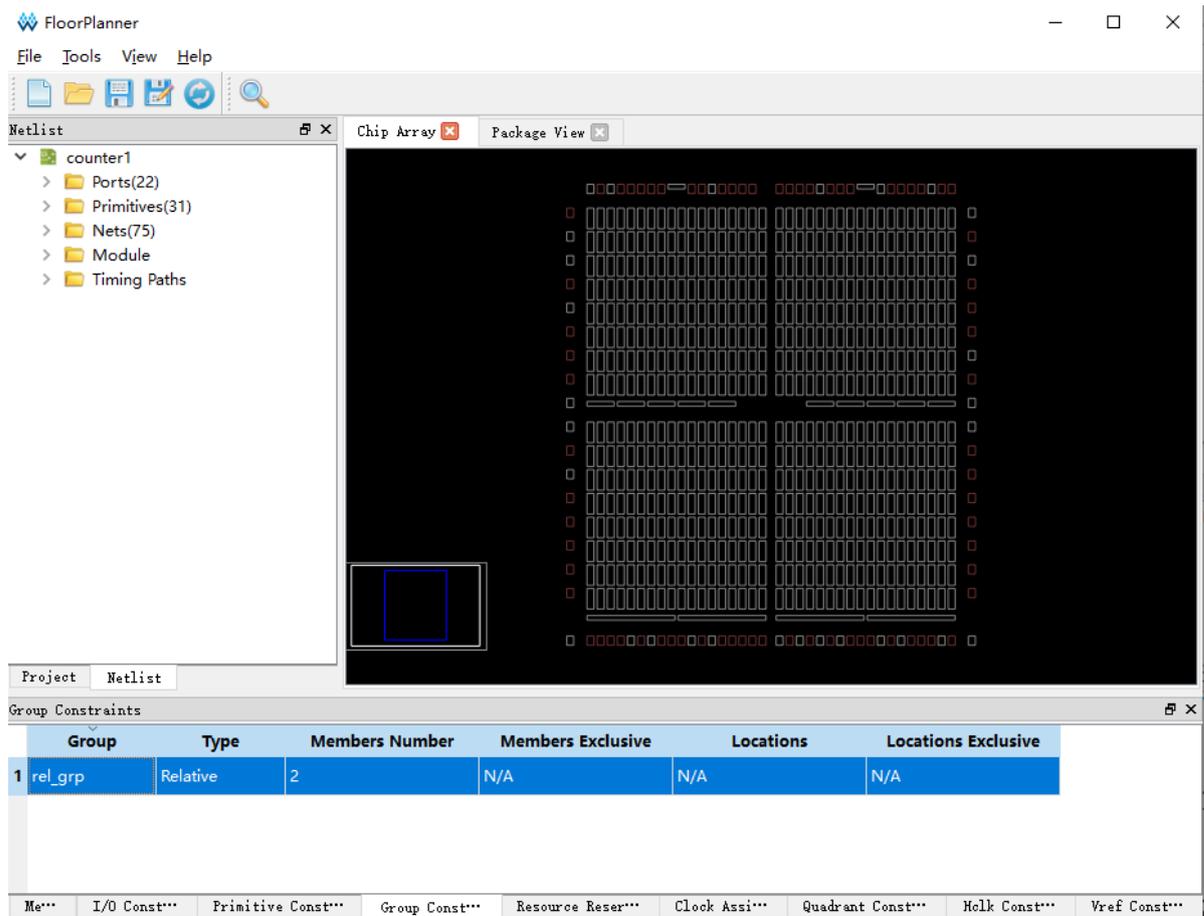


图 4-8 Relative Group Constraints



## 4.5 Resource Reservation 创建

1. 在 Resource Reservation 编辑窗口中，单击鼠标右键，弹出菜单，点击“Reserve Resources”，在编辑器中添加 Resource Reservation 约束，如图 4-9 所示；
2. 选中新建的 Resource Reservation 约束拖拽到 Chip Array 窗口中想要进行预留约束的位置，图 4-10 所示拖拽至 BSRAM\_R10[1]位置完成 Resource Reservation 约束。

图 4-9 创建 Resource Reservation 约束

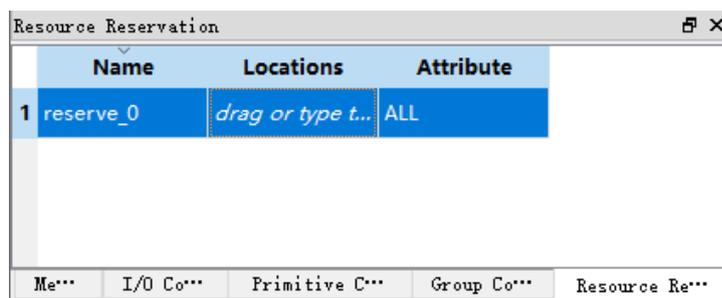
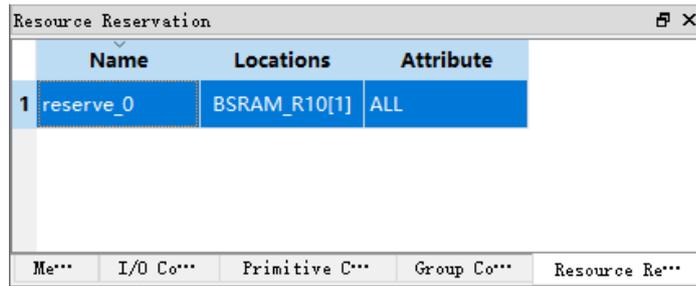


图 4-10 Resource Reservation



	Name	Locations	Attribute
1	reserve_0	BSRAM_R10[1]	ALL

Me... I/O Co... Primitive C... Group Co... Resource Re...

## 4.6 Clock Assignment 创建

1. 在 Clock Assignment 编辑窗口中，右键单击菜单，选择“Clock/Control Assignment”，弹出 Clock Assignment 设置对话框；
2. 单击“”，弹出 Net Finder 对话框，选择需要约束的 Net，在 Net Finder 对话框中，单击“OK”添加 Net；
3. 设置时钟类型，在 Type 下拉列表中选择 Type 类型，设置 Signal 类型，如图 4-11 所示；
4. 设置完成后，单击“OK”，即将该条约束添加至 Clock Assignment 编辑窗口中，如图 4-12 所示。

图 4-11 Clock Assignment 约束创建

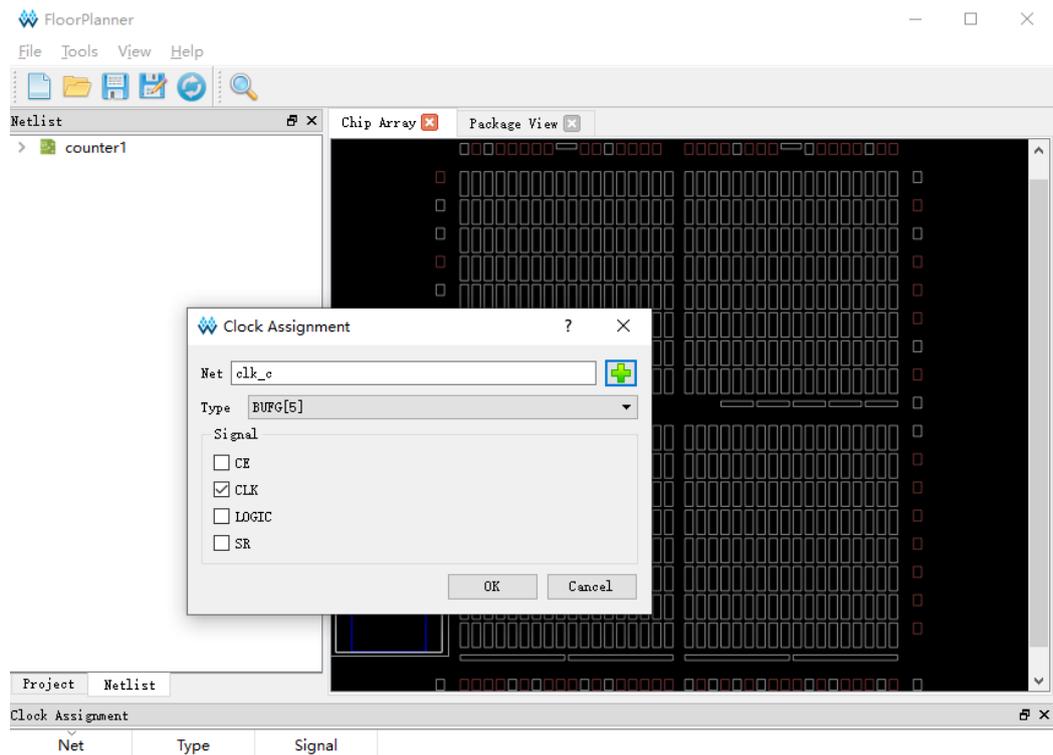
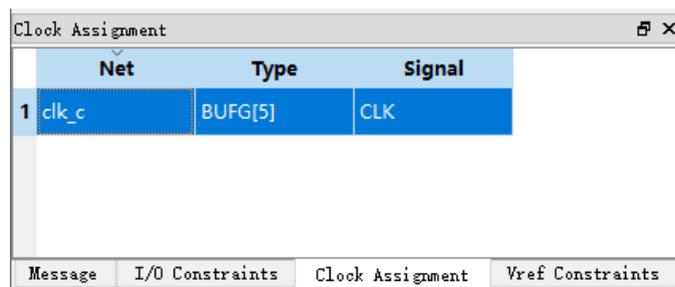


图 4-12 Clock Assignment 约束



## 4.7 Quadrant Constraints 创建

Quadrant Constraints 仅对以下两种类型的 Instance 进行约束：

- Dcs
- Dqce

1. 在 Quadrant Constraints 编辑窗口中，右键单击菜单，选择“Select Dcs/Dqce”，弹出 Quadrant Constraints 对话框；
2. 单击“”，弹出 Dcs/Dqce 选择对话框，选取 Instance，在 Dcs/Dqce

选择对话框中，单击“OK”，Instance 完成设置；

3. 在 Quadrant Constraints 对话框的 Position 下方选取所要约束的象限，如图 4-13 所示；
4. 在 Quadrant Constraints 对话框中单击“OK”，即会将该条约束添至 Quadrant Constraints 编辑窗口，如图 4-14 所示。

图 4-13 Quadrant Constraints 创建

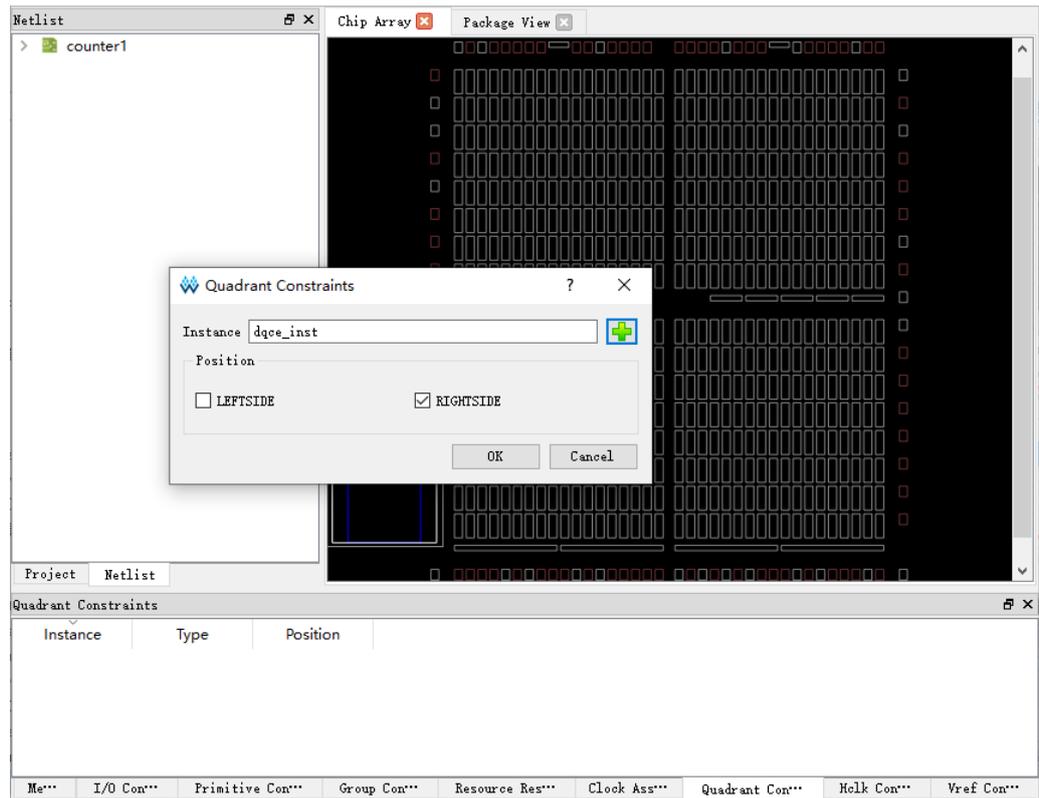


图 4-14 Quadrant Constraints

Instance	Type	Position
1 dqce_inst	DQCE	RIGHTSIDE

## 4.8 Hclk Constraints 创建

Hclk Constraints 仅对以下两种类型的 Instance 进行约束：

- Clkdiv
- Dlldly

Hclk Constraints 的创建步骤如下：

1. 在 Hclk Constraints 编辑窗口中，右键单击菜单，选择“Select Hclk”，弹出 Hclk Constraints 对话框；
2. 单击“+”按钮，弹出 Hclk 对话框，选取 Instance，在 Hclk 对话框中，单击“OK”，设置 Instance；
3. 在 Hclk Constraints 对话框中 Position 下方选取所要约束的位置，如图 4-15 所示；
4. 单击 Hclk Constraints 对话框的“OK”，即可将该约束添至 Hclk Constraints 编辑器中，如图 4-16 所示。

图 4-15 Hclk Constraints 创建

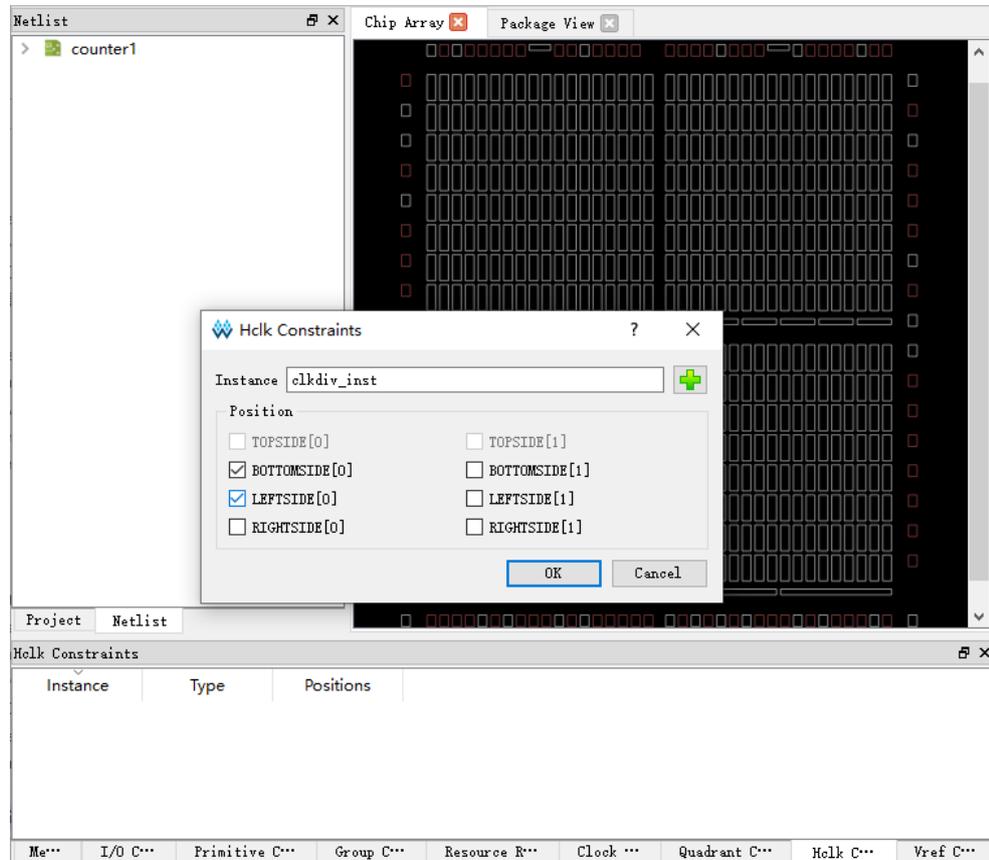
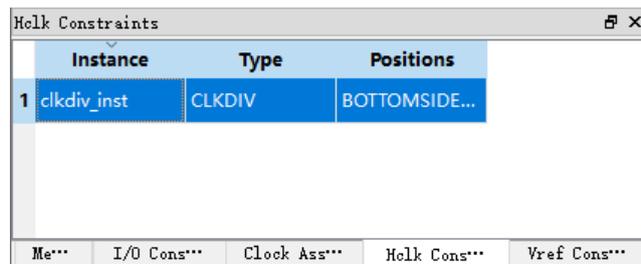


图 4-16 Hclk Constraints

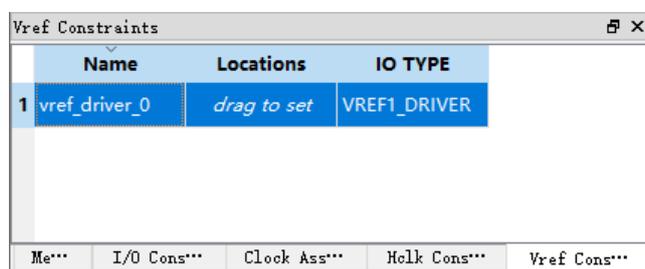


## 4.9 Vref Constraints 创建

拖至 Chip Array 窗口创建 Vref Constraints，步骤如下：

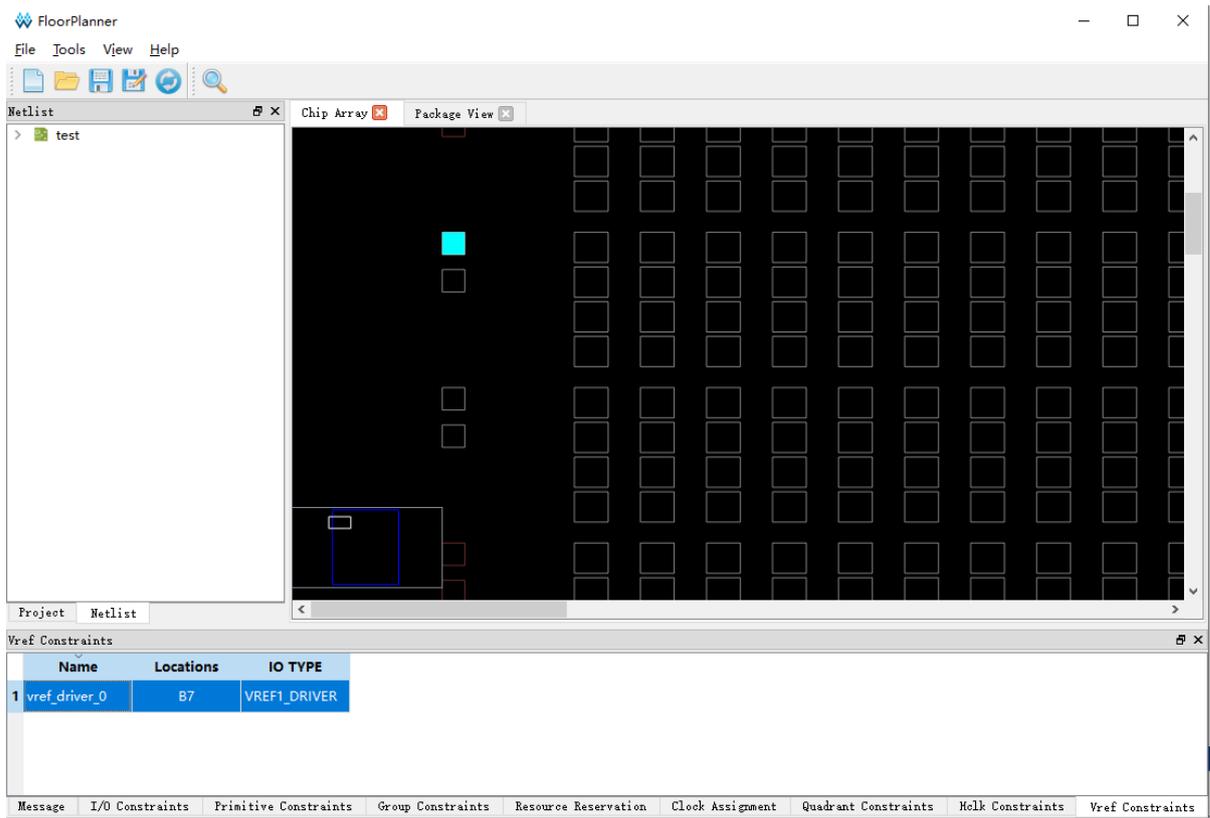
1. 在 Vref Constraints 编辑窗口中，右键单击菜单，选择“Define Vref Driver”，即可将该条 Vref Constraints 约束添至 Vref Constraints 编辑器中，如图 4-17 所示；
2. 放大 Chip Array 窗口至宏单元模式，选中 Vref Constraints 编辑窗口中新创建的 Vref Constraints，拖拽至 Chip Array 窗口中的 B7 位置，Vref Constraints 的 Location 信息显示为 B7，如图 4-18 所示。

图 4-17 Vref Constraints 创建



可自定义 Vref 约束名，Vref 名字不允许重名，设置期间，系统会进行检查，如名字重复，则提示用户，如图 4-20 所示。

图 4-18 拖拽至 Chip Array 窗口生成 Vref Constraints location 信息



拖至 Package View 创建 Vref Constraints，步骤如下：

1. 在 Vref Constraints 编辑窗口中，右键单击菜单，选择“Define Vref Driver”，即可将该条 Vref Constraints 约束添至 Vref Constraints 编辑器中，如图 4-17 所示；
2. 选中 Vref Constraints 编辑窗口中新创建的 Vref Constraints，拖拽至 Package View 窗口中的 B7 位置，Vref Constraints 的 Location 信息显示为 B7，如图 4-19 所示。

图 4-19 拖拽至 Package View 窗口生成 Vref Constraints location 信息

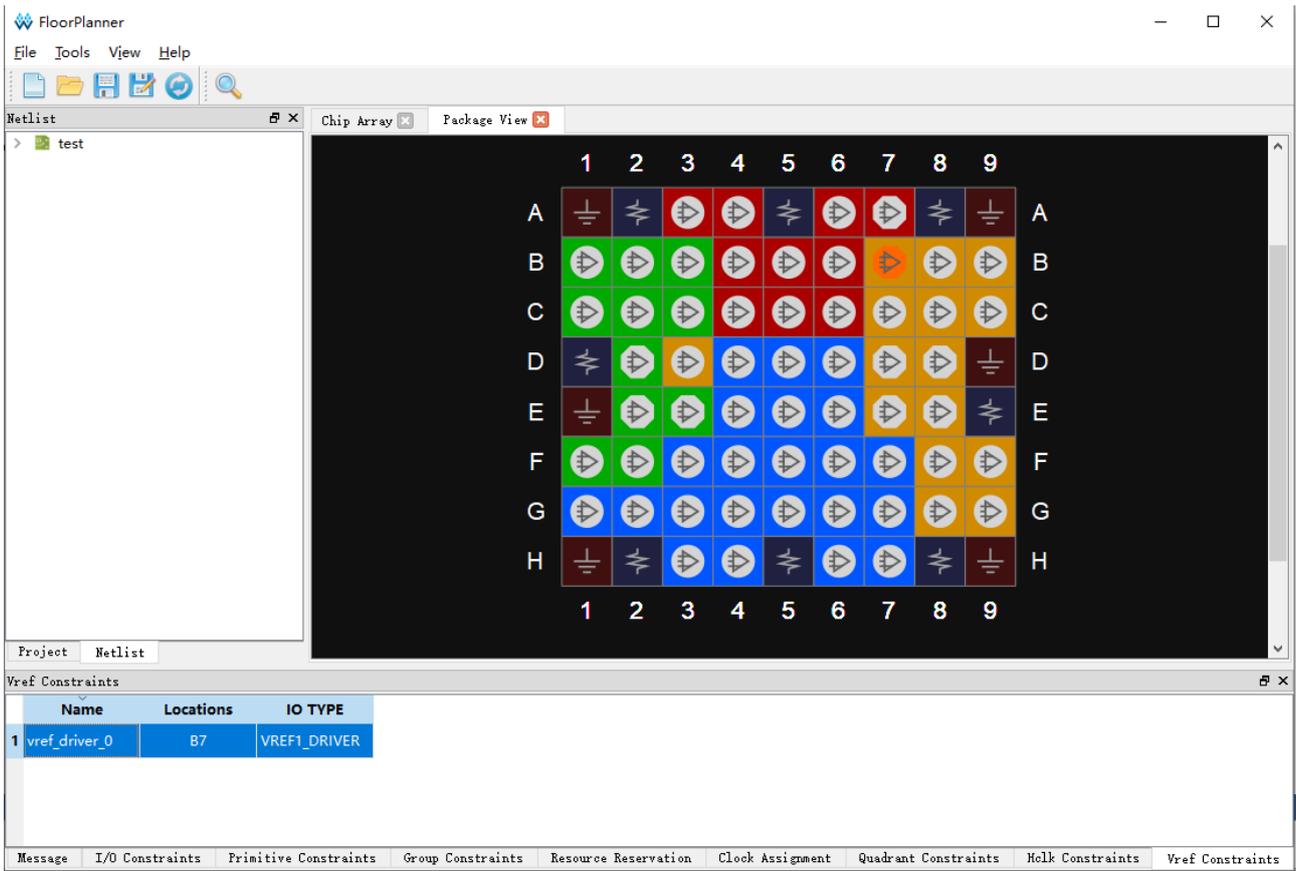
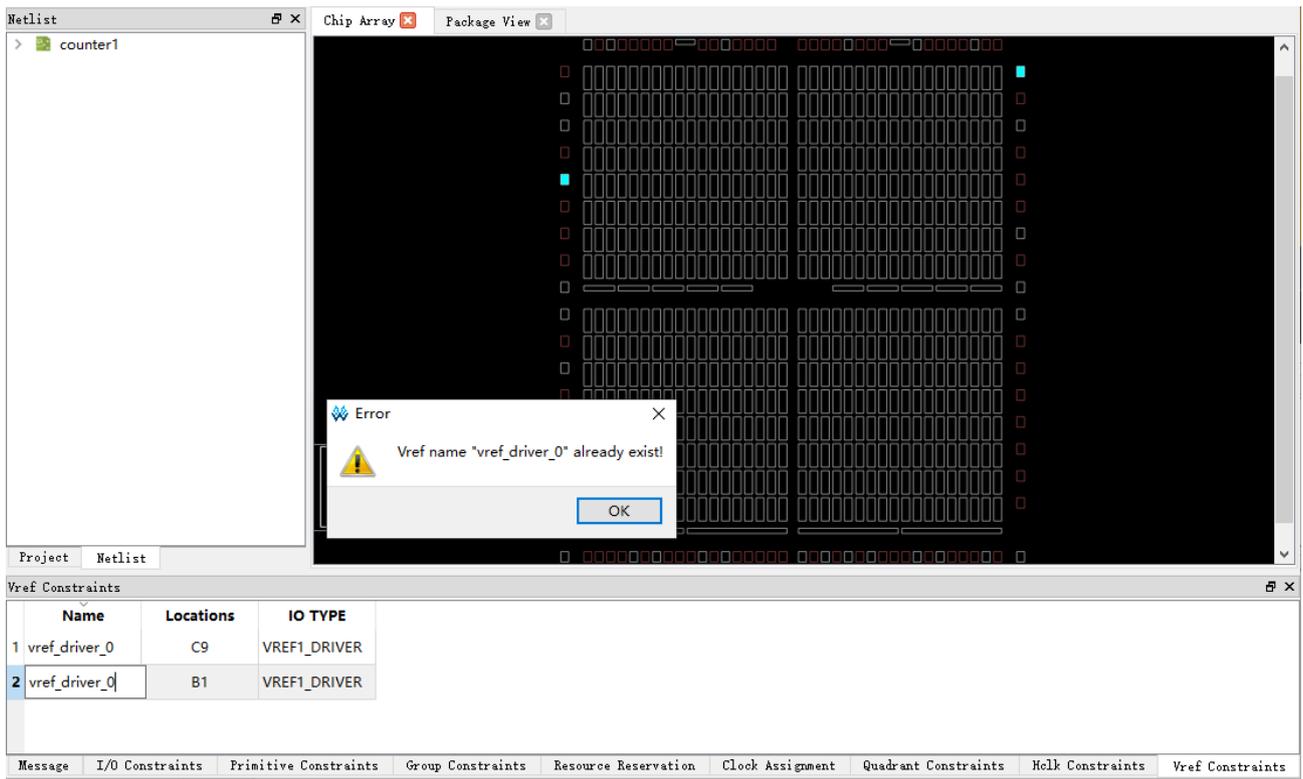


图 4-20 Vref Constraints 名字重复



# 5 时序优化

Gowin 软件 FloorPlanner 支持工程的时序优化，通过物理位置的约束或关键路径的修改等方式帮助用户实现时序收敛。

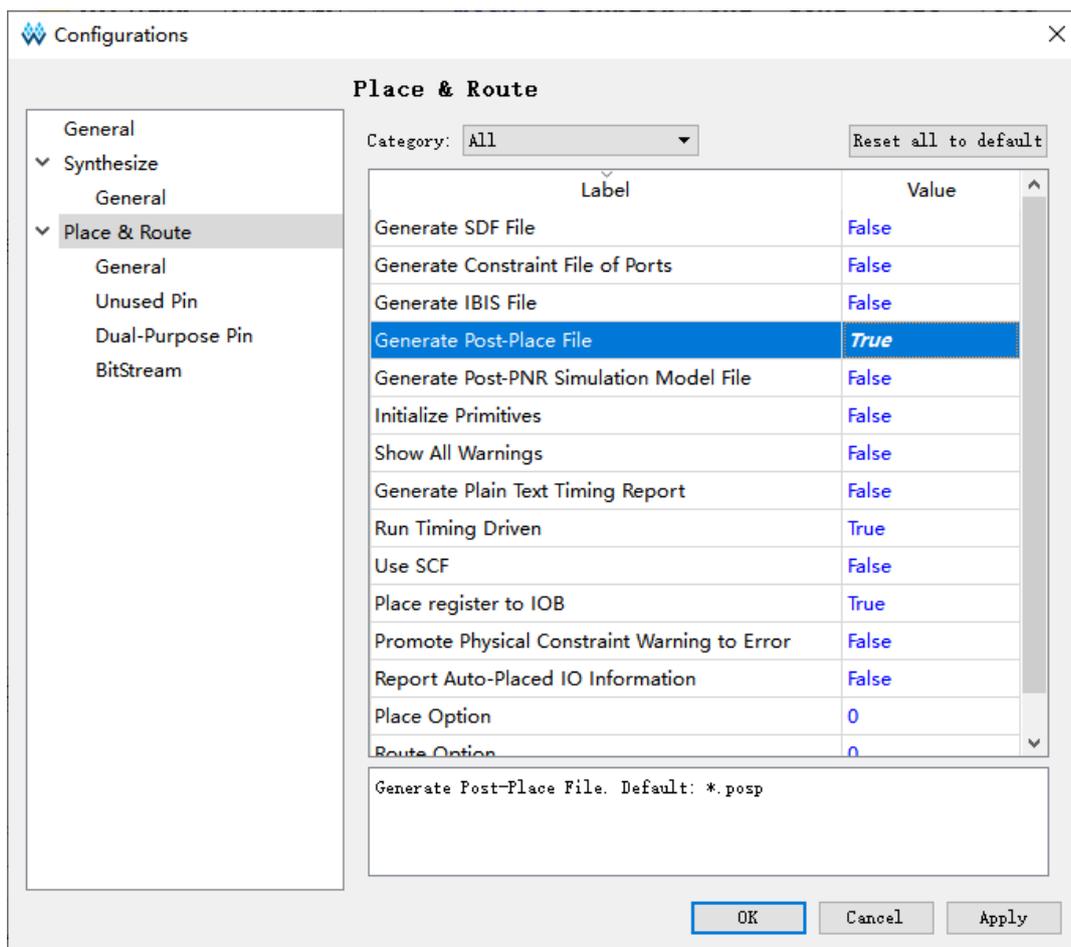
采用 FloorPlanner 进行时序优化的操作流程如下：

1. 新建工程；
2. 运行 “Synthesize”实现综合，生成网表文件，后缀为.vm；
3. 添加物理约束文件和时序约束文件。物理约束和时序约束非必须选项，但可帮助用户更好的实现功能，建议添加；
4. 运行 “Place & Route”进行布局布线，同时生成数据流文件；

**注！**

运行布局布线前，需设置 “Place & Route”的配置对话框，设置“Generate Post-Place File”为“True”生成器件布局信息文件如图 5-1 所示，用于 FloorPlanner 读取布局布线后的位置信息。

图 5-1 生成器件布局信息文件设置

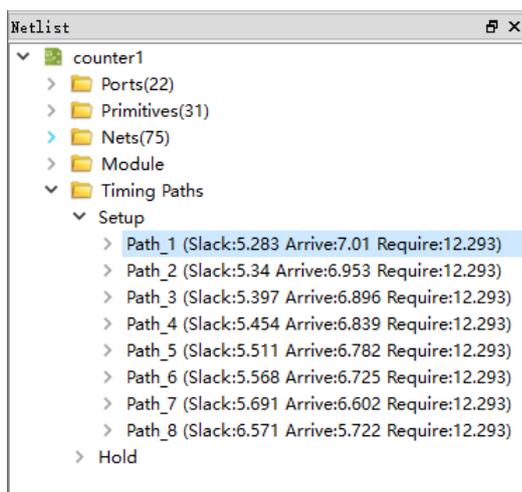


5. 查看时序报告，如最大频率满足设计需求可不再调试。如果时序不满足，需使用 FloorPlanner 产生多种组合的约束，多次迭代实现时序收敛；
6. 读取物理约束文件和时序约束文件；
7. 启动 FloorPlanner 工具，加载布局布线后生成的文件.posp 文件；
8. 加载时序路径文件，在“Netlist > Timing Paths”窗口中显示读入的关键路径信息，显示每条 path 的 slack、arrive time、require time 等信息，如图 5-2 所示。

#### 注！

在反复调试过程中，.posp 文件和.timing\_paths 文件不用每次加载，用户使用“Reload”重新加载即可。

图 5-2 读取时序路径文件



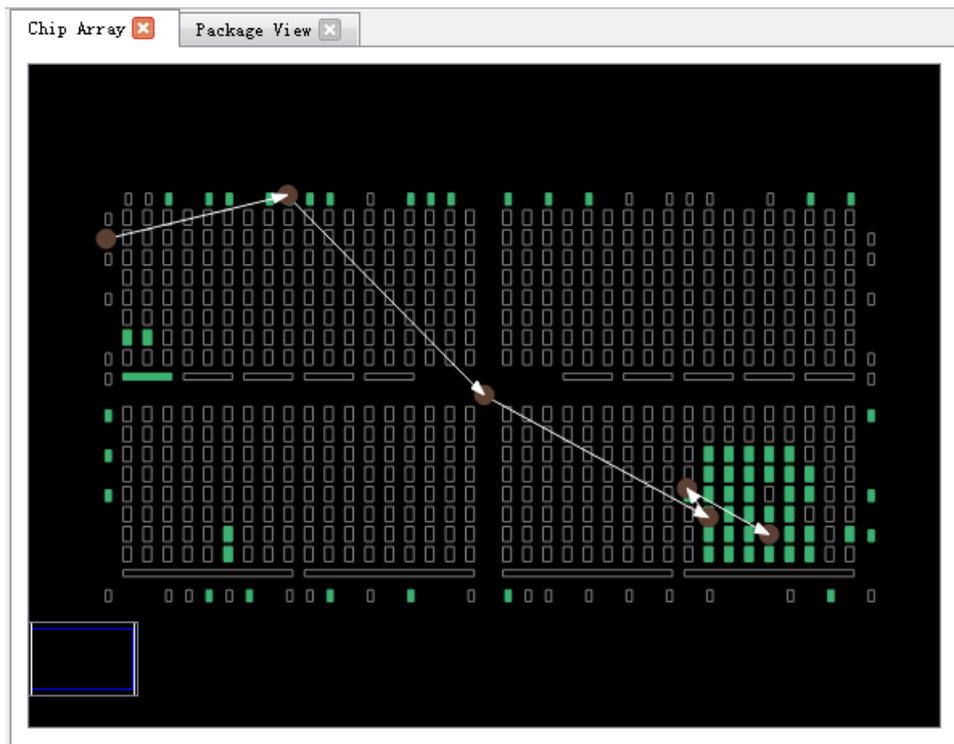
9. 在时序收敛的调试过程中，需找出工程中的关键路径，通过修改设计或调整器件位置实现时序收敛。在 FloorPlanner 中，可通过调整位置信息实现时序收敛。步骤如下：
  - a). 查看关键路径信号流向。

在时序收敛的调试中，关键路径的信号流向是一个因素。在 FloorPlanner 的 Netlist 窗口中选择一条关键路径，右键单击，选择“Highlight”如图 5-3 所示，在“Chip Array”窗口中可观察到该路径的信号流向，如图 5-4 所示，此时 Chip Array 窗口必须通过右键菜单 Show Place View > All Instance 调整到所有 instance 的 place 视图才能够显示该路径的信号流行。
  - b). 调整不合理的位置信息。

如图 5-4 所示，模块的分布相对集中，只有一个分布的位置相对较远。再观察关键路径的信号流向，路径比较曲折，跨度较大，是影响时序的一个因素。以拖拽的方式调整位置不合理的那个模块，减少信号的曲折路径，如图 5-5 所示。
10. 重新运行“Place & Route”，查看时序结果。如果频率满足用户需求可不再做调试，如果不满足重复上述 5，6，7，8，9 步骤。



图 5-5 调整后的位置信息



# 附录 A 物理约束语法规范

## A.1 I/O Constraints

IO 约束可将 port、buffer 约束到指定 IOB 位置处。

### 语法

```
“IO_LOC” “”obj_name“” obj_location [“exclusive”] “;”
```

### 约束元素

obj\_name

obj\_name 可取 port、I/O buffer 的 name 作为 obj\_name。

obj\_location

obj\_location 为 IOB 位置，如“A11”、“B12”等，若指定多个位置，位置之间需要用逗号分隔，如“A11,B2”。

exclusive

exclusive 为可选项，在约束位置之后，表明该约束语句中的 obj\_location 仅可以放置 obj\_name 指定的原语。

### 注！

当 obj\_name 为 escaped name 格式（以反斜线开头，空格结尾）时，obj\_name 两边需加上引号。

### 应用举例

示例 1.

```
IO_LOC “io_1” A1;
```

```
// 对象 io_1 被约束在 pin A1 的位置。
```

示例 2.

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

// 对象 io\_1 被约束在 pin A1、B14、A15 的位置，布局时将取三个位置之一进行布局。

示例 3.

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

// 对象 io\_2 被约束在 pin A1 处，且 A1 位置仅可以被 io\_2 所占用。

示例 4.

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

// 对象 io\_2 被约束在 pin A1、B14、A15 处，且 A1、B14、A15 三个位置仅可以被 io\_2 占用。

## A.2 PORT 属性约束

Port 属性约束，用于设定 port 的各种属性值。如 port 的电平标准 IO\_TYPE，上拉/下拉模式 PULL\_MODE，驱动能力 DRIVE 等，详细属性设置标准请参考相应数据手册。

### 语法

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

一个约束语句中可设定多个属性，各个属性之间使用空格分隔。

### 约束元素

需要属性约束的 Port 的 name，attribute 和 attribute value。

### 应用举例

示例 1.

```
IO_PORT "port_1" IO_TYPE = LVTTTL33;
```

// 设置 port\_1 的 IO\_TYPE 为“LVTTTL33”。

示例 2.

```
IO_PORT "port_2" IO_TYPE = LVTTTL33 SLEW_RATE = FAST  
PULL_MODE =KEEPER;
```

// 设置 port\_2 的 IO\_TYPE 为“LVTTTL33”，SLEW\_RATE 属性值为“FAST”，PULL\_MODE 属性值为“KEEPER”。

示例 3.

```
IO_PORT "port_3" IO_TYPE=LVDS25;
```

// port\_3 连接的 BUF 为普通 IBUF，通过该约束，可将该 IBUF 转化为 TLVDS\_IBUF。

## A.3 Primitive Constraints

Primitive Constraints 用于将 instance 布局到指定的 GRID 处，可以通过 Primitive Constraints 对 LUT/BSRAM/SSRAM/DSP/PLL 等 instance 进行约束。

### 语法

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

### 约束元素

obj\_name

约束对象的 instance 的 name。

obj\_location

obj\_location 包含如下几类：

- 单一位置信息，指定到 LUT，如：RxCy[0-3][A-B]；
- 位置信息为一个范围，指定多行或多列：
  - 包含多个 CLS 或 LUT：“RxCy”，“RxCy[0-3]”
  - 指定多行：“R[x:y]Cm”，“R[x:y]Cm[0-3]”，“R[x:y]Cm[0-3][A-B]”
  - 指定多列：“RxC[m:n]”，“RxC[m:n][0-3]”，“RxC[m:n][0-3][A-B]”
  - 指定多行多列：“R[x:y]C[m:n]”，“R[x:y]C[m:n][0-3]”，“R[x:y]C[m:n][0-3][A-B]”

### 注！

在一条约束语句中，可包含多个 ins\_location，使用“,”分隔。

### PLL 约束位置

对于 PLL 约束位置信息书写格式为“PLL\_L”或“PLL\_R”，若左边可放置多个 PLL，可设为“PLL\_L[0]”、“PLL\_L[1]”...，若右边可放置多个 PLL，可设为“PLL\_R[0]”、“PLL\_R[1]”...

### BSRAM 约束位置

BSRAM 约束位置信息为“BSRAM\_R10[0]”（第 10 行第一个 BSRAM），“BSRAM\_R10[1]”...

### DSP 约束位置

DSP 约束位置格式为 “DSP\_R19[0]” (第 19 行第一个 DSP Block), “DSP\_R19[1]”... 若需指定具体 macro, 可标记为: DSP\_R19[0][A]或 DSP\_R19[0][B]。

**exclusive**

关键字“exclusive”为可选项, 在约束位置之后, 表明该约束语句中的 obj\_location 仅可以放置 obj\_name 指定的 instance。

### 应用举例

示例 1.

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

// lut\_1 被约束在 R2C3 位置和 R5C10 的第 0 个 CLS 的第 1 个 LUT 的位置。

示例 2.

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

// ins\_2 被约束在 R5C6 的第 2 个 CLS 的位置, 且该位置仅可以放置该 instance。

示例 3.

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

// ins\_3 被约束在行坐标第二行到第六行, 列坐标第一列的区域位置。

示例 4.

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

// ins\_4 被约束在行坐标为第一行到第四行, 列坐标为第二列到第六列之间的区域位置, 且该区域位置仅能被该 instance 所占用。

示例 5.

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

// ins\_5 被约束在行坐标第一行到第四行, 列坐标第二列到第六列之间的区域位置的任意一个 GRID 的第 1 个 CLS 中。

示例 6.

```
INS_LOC "reg_name" B14;
```

// 通过对 REGISTER/IOLOGIC 的 INS\_LOC 约束, 约束其到 IOB 的位置 B14。

示例 7.

```
INS_LOC "pll_name" PLL_L;
// 通过对 PLL 的 INS_LOC 约束，约束其位置 PLL left.
```

示例 8.

```
INS_LOC "bsram_name" BSRAM_R10[2];
// 通过对 BSRAM 的 INS_LOC 约束，约束其位置第 10 行的第 3 个
BSRAM 位置处。.
```

示例 9.

```
INS_LOC "dsp_name" DSP_R19[2];
// 通过对 DSP 的 INS_LOC 约束，约束其位置第 19 行第 3 个 DSP
Block.
```

一个 LUT4 的位置可以放置一个 lut1/lut2/lut3/lut4，lut5 需要占用两个 LUT4 的位置（一个 CLS），lut6 需要占用 4 个 LUT4 的位置（两个 CLS），lut7 需要占用 4 个 CLS 的位置（一个 GRID），lut8 需要占用 8 个 CLS（两个 GRID）。故对于不同 Instance 类型的约束，其约束位置的最小单元也不相同，对于 BSRAM/SSRAM/DSP（每个 DSP 单元有两个 MICRO，每个 MICRO 有两个 UNIT）等也是如此，如下示例：

示例 10.

LUT4 单元约束：

```
INS_LOC "lut4_name" R5C15[1][A];
// 将 lut4_name 约束到 R5C15 的第 1 个 CLS 的第 1 个 LUT 处。
```

示例 11.

CLS 单元约束：

```
INS_LOC "lut5_name" R5C15[3];
// 将 lut5_name 约束到 R5C15 的第 3 个 CLS 处。
```

示例 12.

CLS 单元约束：

```
INS_LOC "lut6_name" R5C15[0];
// 将 lut6_name 约束到 R5C15 的第 0 个 CLS 处（将占用 CLS[0]和
CLS[1]）。
```

示例 13.

GRID 单元约束：

```
INS_LOC "lut7_name" R5C15;
BSRAM type: INS_LOC "bsram_name" R10C5; // for GW2A55K
// 将 lut7_name 约束到 R5C15 处，LUT7 占用一个 GRID。
```

示例 14.

GRID 单元约束:

```
INS_LOC "lut8_name" R5C15;
// 将 lut8_name 约束到 R5C15 处，lut8_name 将占用 R5C15
和 R5C16 两个 GRID。
```

示例 15.

DSP MICRO 单元约束:

```
INS_LOC "mult_name" DSP_R19[1][A]; // for GW2A55K
// 将 mult_name 约束到第 19 行第 2 个 DSP 的第一个 macro 中。
```

## A.4 Group Constraints

Group Constraints 包括 Primitive Group Constraints 和 Relative Group Constraints，如下所述。

### A.4.1 Primitive Group Constraints

Primitive Group 约束用于定义一个组约束，组是包含各类 Instance 对象的集合。通过 Primitive Group 约束，可将普通 Instance 如 LUT、DFF 等，或 BUF、IOLOGIC 等添加到一个组中，并可通过约束该组的位置实现对该组中所有的对象的位置约束。

#### 语法

GROUP 的定义:

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

添加 Instance 到组中:

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

约束组的位置:

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

注!

当 `group name` 为 `escaped name` 格式（以反斜线开头，空格结尾）时，`group_name` 两边须加上引号。

### 约束元素

`group_name`

定义一个 `name` 作为该组的 `name`

`obj_name`

`obj_name` 用于将指定的 `Instance` 对象添加到组中

`group_location`

指定该 `group` 的约束位置，`group_location` 可取 `IOB` 和 `GRID` 的位置

`exclusive`

关键字“`exclusive`”为可选项，在组定义语句或位置约束语句之后；

一个对象可以被多个组包含，但在组定义语句后添加“`exclusive`”关键字，表示该组内的对象仅可被该组所包含；

在位置约束语句之后使用“`exclusive`”，表示该约束位置仅可被该组内的对象所占用。

### 应用举例

示例 1.

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

// 创建一个名为 `group_1` 的组，添加对象 `ins_1`, `ins_2`, `ins_3`, `ins_4` 到该组中。

示例 2.

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

// 创建一个名为 `group_2` 的组，对象 `ins_5`, `ins_6`, `ins_7` 属于且仅可属于该组。

示例 3.

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

// 添加 `io_1`, `io_2` 到组 `group_1` 中。

示例 4.

```
GRP_LOC group_1 R3C4, A14, B4;
```

// 组 `group_1` 中的对象可布局在 `R3C4`, `A14`, `B4` 位置处。

示例 5.

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

// 组 group\_2 中的 Instance 对象可布局在区域 R[1:3]C[1:4]的范围内，且该范围仅可布局 group\_2 中的 Instance 对象。

## A.4.2 Relative Group Constraints

通过 Relative Group Constraints, 可实现对 instance 对象的相对位置约束。

### 语法

定义 Relative 约束的组:

```
“REL_GROUP” group_name “=” “{” “” obj_names “” “}”“;”
```

添加 instance 对象到已定义的组中:

```
“REL_GROUP” group_name “+” “{” “” obj_names “” “}”“;”
```

对组中的 instance 进行相对位置约束:

```
“INS_RLOC” “” obj_name “” relative_location “;”
```

### 约束元素

obj\_name

约束对象的名称。

relative\_location

行列相对位置信息描述。

### 应用举例

示例 1.

```
REL_GROUP grp_1 = { “ins_1” “ins_2” “ins_3” “ins_4” };
```

```
INS_RLOC “ins_1” R0C0;
```

```
INS_RLOC “ins_2” R2C3;
```

```
INS_RLOC “ins_3” R3C5;
```

// 定义一个名为 grp\_1 的组约束, 并添加 ins\_1, ins\_2, ins\_3, ins\_4 到 grp\_1 中。以 ins\_1 为相对位置原点 R0C0, ins\_2 约束到相对 ins\_1 的 R2C3 处, ins\_3 约束到相对 ins\_1 的 R3C5 处。

## A.5 Resource Reservation

通过 Resource Reservation 约束，可保留指定的位置或区域以避免在布局中使用。

### 语法

```
“LOC_RESERVE” location [ res_obj ] “;”
```

### 应用举例

示例 1.

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

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

示例 2.

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

示例 3.

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

// 以上示例中约束的位置信息将会在布局阶段被保留。

## A.6 Vref Constraints

芯片支持外部参考电压输入，芯片的每个 PAD（有 IOLOGIC）均可作为外部参考电压的输入 PAD，对整个 BANK 有效。Vref Constraints 约束可用于对外部参考电压的输入 pin 的名称和位置进行约束。

### 语法

```
“USE_VREF_DRIVER” vref_name [location]“;”
```

### 约束元素

vref\_name

自定义的 VREF pin name

location

芯片中任意 PAD(有 IOLOGIC)位置可作为 VREF pin 约束的 location。

### 应用举例

示例 1.

```
USE_VREF_DRIVER vref_pin;
```

```
IO_PORT "port_1" IO_TYPE = SSTL25 VREF=vref_pin;
IO_PORT "port_2" IO_TYPE = SSTL25 VREF=vref_pin;
// 定义一个名为"vref_pin"的 VREF pin, 设置 port_1 与 port_2 的 VREF
属性为 vref_pin。
```

示例 2.

```
USE_VREF_DRIVER vref_pin C7;
IO_PORT "port_1" IO_TYPE = SSTL25 VREF=vref_pin;
IO_PORT "port_2" IO_TYPE = SSTL25 VREF=vref_pin;
// 定义一个名为"vref_pin"的 VREF pin, 将其约束到 PAD C7(bank 3,
GW1N-4, WLCSP72), 设置 port_1 与 port_2 的 VREF 值为 vref_pin,
port_1 与 port_2 将布局到 C7 所在的 bank 上。
```

## A.7 Quadrant Constraints

Quadrant (象限) 用于将 DCS/DQCE 等需要象限布局的对象约束到指定的象限中 (GW1N 家族有 LEFT 和 RIGHT 两个象限, GW1N-9/GW1NR-9/GW1N-9C/GW1NR-9C 及 GW2A 家族有 TOPLEFT、TOPRIGHT、BOTTOMLEFT、BOTTOMRIGHT 四个象限, 具体信息见相关数据手册)。

### 语法

```
"INS_LOC" ""obj_name"" quadrant ","
```

### 约束元素

obj\_name

约束对象的名称。

quadrant

GW1N 系列: "LEFT"("L"), "RIGHT"("R")

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

注!

括号内为缩写形式。

### 应用举例

示例 1.

```
INS_LOC "dcs_name" LEFT;
// 约束 DCS 对象 dcs_name 到 LEFT 象限中 (GW1N 家族)。
```

## A.8 Clock Assignment

Clock Assignment 约束是对于设计中特定 net 到全局时钟线或不绕时钟线的约束。芯片资源中每个象限存在 8 个主时钟和 8 个长线资源，可通过该约束，实现对特定 fanout (CLK/CE/SR/LOGIC) 的 net 进行全局时钟线布线约束。

BUFG[0-7]表示 8 个主时钟的资源。

BUFS 表示 8 个长线资源。

LOCAL\_CLOCK 表示该条 net 不绕时钟线。

CLK 信号为连接 CLK 引脚的信号，CE 信号为连接 CE 引脚的信号，SR 信号为连接 SET/RESET/CLEAR/PRESET 引脚的信号，LOGIC 为连接其它类型输入引脚的信号。

### 语法

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

注!

NET\_LOC 关键字更新为 CLOCK\_LOC，但是之前的 NET\_LOC 用来约束 BUFG、BUFS 仍然支持。

### 约束元素

net\_name

net 的名字

global\_clocks

BUFG[0-7]: 8 个主时钟资源

BUFS: 8 个长线资源

LOCAL\_CLOCK: 不绕时钟线

fanout

CLK: fanout 为 CLK 的 net

CE: fanout 为 CE 的 net

SR: fanout 为 SET/RESET/CLEAR/PRESET 的 net

LOGIC: fanout 为以上 fanout 之外的 net

指定多个 fanout，可使用“|”符号进行分隔。

注！

若 global\_clocks 选择的是 LOCAL\_CLOCK，则 fanout[quadrant]不可选。

quadrant

GW1N 系列：“LEFT”(“L”), “RIGHT”(“R”)

GW1N-9/GW1NR-9/GW1N-9C/GW1NR-9C 及 GW2A 系列：  
“TOPLEFT”(“TL”),“TOPRIGHT”(“TR”), “BOTTOMLEFT”(“BL”),  
“BOTTOMRIGHT”(“BR”)

注！

括号内为缩写形式，仅在指定主时钟 BUFG[0-7]资源时使用象限约束关键字有效。

### 应用举例

示例 1.

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

// 约束 CLOCK 对象“net”的 fanout 为 CLK 的 net 绕到芯片 LEFT 象限的第 0 条主时钟资源（GW1N 系列）上。

示例 2.

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

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

// 约束 CLOCK 对象“net”的 fanout 为 CLK 或 CE 的 net 绕到芯片的主时钟资源上。

示例 3.

```
CLOCK_LOC "net" BUFS = CE;
```

```
NET_LOC "net" BUFS = CE;
```

// 约束 CLOCK 对象“net”的 fanout 为 CE 的 net 绕到芯片的长线资源上。

示例 4.

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

// 约束 CLOCK 对象“net”不绕时钟线。

## A.9 Hclk Constraints

通过 CLKDIV/DLLDLY 约束，可将 CLKDIV/DLLDLY 约束到相关位置。CLKDIV/DLLDLY 约束位置与普通 instance 对象约束位置不同，使用

“TOPSIDE”, “BOTTOMSIDE”, “LEFTSIDE”, “RIGHTSIDE”表示约束位置的四边。

### 语法

```
“INS_LOC” “”ins_name “” location“;
```

### 约束元素

obj\_name

取 CLKDIV/DLLDLY 的 instance name 作为 obj\_name。

location

“TOPSIDE[0-1]” (“TS[0-1]”)

“BOTTOMSIDE[0-1]” (“BS[0-1]”)

“LEFTSIDE[0-1]” (“LS[0-1]”)

“RIGHTSIDE[0-1]” (“RS [0-1]”)

注!

括号内为缩写形式。

### 应用举例

示例

```
INS_LOC “clkdiv_name” TS[0];
```

```
// 将 clkdiv_name 布局到 TOPSIDE[0]处。
```

