



Gowin 设计约束指南

SUG101-1.5, 2018-11-15

版权所有©2018 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、翻译本档内容的部分或全部，并不得以任何形式传播。

免责声明

本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
2016/03/03	1.05	初始版本
2016/07/14	1.06	物理约束编辑器更新为 FloorPlanner
2016/11/16	1.07	<ol style="list-style-type: none"> 更新 Tools 菜单下的约束描述，增加 Primitive 约束、Net 约束一键拖拽功能的描述，增加 Primitive Group/Relative Group 约束的 copy location 的描述，增加其他约束的 Constraint 编辑窗口功能描述； 增加 Constraint 编辑窗口的模块约束描述。
2017/02/28	1.08	<ol style="list-style-type: none"> Reserve constraint 中增加 LUT/WLUT/REG 的语法描述； Place View 中增加 Lut、Reg 密度显示的描述； 增加按照模块打印时序报告的功能。
2018/01/30	1.1	<ol style="list-style-type: none"> FloorPlanner 增加 reload 功能； IO 约束中增加 I3C Mode 属性； FloorPlanner 中 Package View 界面中取消多种 VCC 合并成一种； 时钟约束中取消 position 的约束。
2018/08/05	1.2	<ol style="list-style-type: none"> 增加支持 GW1N-2B/GW1N-4B/GW1N-6ES/GW1N-9ES/GW1NR-9ES/GW1NS-2/GW1NS-2C； FloorPlanner 中 Package View 界面中取消差分 IO 正负极的区分以及 IO 的颜色显示； FloorPlanner 中 Chip Array 界面中更改了约束位置颜色的显示； FloorPlanner 中 Reserve constraint 中 Arribute 一栏更改为 ALL/LUT/REG； FloorPlanner 删除命令窗口 TCL Console, 更改为 Message 窗口； FloorPlanner 删除 check 工具； set_false_path 、 set_min_delay 、 set_max_delay 、 set_multicycle_path 这几个约束中，from 的类型只保留一个"from"，而"rise_from""fall_from"不再提供给用户；to 的类型只保留一个"to"，而"rise_to""fall_to"不再提供给用户。
2018/09/21	1.3	<ol style="list-style-type: none"> 更新附录 A 的描述。
2018/10/26	1.4	<ol style="list-style-type: none"> 增加支持 GW1NZ-1/GW1NSR-2C； 更新附录 A 的部分描述。
2018/11/15	1.5	<ol style="list-style-type: none"> 增加支持 GW1NSR-2； 删除器件 GW1N-6ES、GW1N-9ES、GW1NR-9ES。

目录

目录	i
图目录.....	v
表目录.....	x
1 关于本手册	1
1.1 手册内容.....	1
1.2 适用产品.....	1
1.3 相关文档.....	1
1.4 术语、缩略语	2
1.5 技术支持与反馈.....	2
2 简介.....	3
3 物理约束	4
3.1 功能简介.....	4
3.2 启动 FloorPlanner	4
3.3 新建和打开约束文件.....	5
3.3.1 新建约束文件	6
3.3.2 FloorPlanner 输出约束文件.....	7
3.3.3 打开约束文件	8
3.4 FloorPlanner 界面	9
3.4.1 菜单栏	10
3.4.2 List 窗口	22
3.4.3 Package View 窗口	27
3.4.4 Chip Array 窗口	30
3.4.5 Constraint 编辑窗口	35
3.5 Message 窗口	39

3.6 创建 Constraints—拖曳方式	39
3.6.1 设置 I/O Constraints 约束位置.....	39
3.6.2 Primitive Constraints 创建.....	42
3.6.3 Group Constraints 的创建	44
3.6.4 Resource Reservation 的创建	45
3.6.5 Clock Assignment 的创建	46
3.6.6 Quadrant Constraints 的创建.....	47
3.6.7 Hclk Constraints 的创建.....	49
3.6.8 Vref Constraints 的创建	51
4 时序约束	53
4.1 静态时序分析(STA)概述.....	53
4.1.1 时序分析基本模型	53
4.1.2 时序分析术语	54
4.1.3 时序分析路径	54
4.1.4 常见的时序检查.....	55
4.2 时序约束编辑器.....	55
4.2.1 概述.....	55
4.2.2 打开编辑器.....	55
4.2.3 新建和打开约束文件.....	56
4.2.4 编辑器界面	58
4.2.5 时序约束界面	59
4.3 编辑 SDC 文件	60
4.4 时序约束编辑	61
4.4.1 Clock 约束.....	61
4.4.2 I/O Delay 约束.....	69
4.4.3 Path 约束	70
4.4.4 工作条件约束	72
4.4.5 时序报告.....	73
4.4.6 保存与导出.....	80
4.5 时序约束的优先级	80
5 时序优化	81

附录 A.物理约束语法规范	86
A.1 I/O Constraints	86
A.2 PORT 属性约束	87
A.3 primitive Constraints	88
A.4 Primitive Group Constraints	90
A.5 Resource Reservation	92
A.6 Relative Group Constraints	92
A.7 Vref Constraints	93
A.8 Quadrant Constraints	94
A.9 Clock Assignment	94
A.10 Hclk Constraints	96
附录 B.时序约束语法规范	97
B.1 时钟约束	97
B.1.1 create_clock	97
B.1.2 create_generated_clock	98
B.1.3 set_clock_latency	100
B.1.4 set_clock_uncertainty	101
B.1.5 set_clock_groups	102
B.2 IO 约束	103
B.2.1 set_input_delay	103
B.2.2 set_output_delay	104
B.3 路径约束	106
B.3.1 set_max_delay set_min_delay	106
B.3.2 set_false_path	107
B.3.3 set_multicycle_path	108
B.4 工作条件约束	110
B.5 时序报告	111
B.5.1 report_timing	111
B.5.2 report_high_fanout_nets	112
B.5.3 report_route_congestion	113
B.5.4 report_min_pulse_width	113

B.5.5 report_max_frequency 114

B.5.6 report_exceptions..... 114

图目录

图 3-1 菜单栏启动 FloorPlanner	5
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 保存输出文件	8
图 3-8 打开物理约束	9
图 3-9 FloorPlanner 界面	10
图 3-10 File 菜单	10
图 3-11 选择器件	11
图 3-12 Tools 菜单	11
图 3-13 原语查找界面	12
图 3-14 新建原语组	13
图 3-15 正确原语组界面	14
图 3-16 无效位置	14
图 3-17 无效位置	14
图 3-18 创建相对位置的组	15
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)	17
图 3-23 象限约束 (GW2A 家族)	18
图 3-24 Hclk 约束	18
图 3-25 Vref 约束	19

图 3-26 选择 One hit drag 原语	19
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.....	21
图 3-31 Windows 菜单.....	22
图 3-32 Help 提示框	22
图 3-33 Project 窗口	23
图 3-34 Netlist 窗口	24
图 3-35 BUS 和非 BUS 结合显示	25
图 3-36 层级显示	25
图 3-37 时序路径显示.....	26
图 3-38 Netlist 右键功能	27
图 3-39 Package view 界面	28
图 3-40 Package View 右键功能	29
图 3-41 差分对显示	29
图 3-42 Chip Array 界面	30
图 3-43 网格模式约束	31
图 3-44 宏单元模式约束	31
图 3-45 原语模式约束.....	32
图 3-46 Chip Array 右键功能	33
图 3-47 Show Place View 显示	34
图 3-48 时序路径高亮显示.....	34
图 3-49 I/O 约束窗口	35
图 3-50 原语约束窗口	36
图 3-51 组约束窗口	36
图 3-52 预留约束窗口	37
图 3-53 时钟约束窗口	37
图 3-54 象限约束窗口	38
图 3-55 Hclk 约束窗口	38
图 3-56 Vref 约束窗口.....	39

图 3-57 Message 窗口.....	39
图 3-58 拖拽到 Chip Array 设置 I/O Constraints.....	40
图 3-59 拖至 Package View 设置 I/O Constraints	41
图 3-60 F3 一键设置 I/O Constraints 位置约束	42
图 3-61 拖拽到 Chip Array 设置 Primitive Constraints.....	43
图 3-62 F3 一键设置 Primitive Constraints 的位置.....	43
图 3-63 Group Constraints 编辑器右键菜单.....	44
图 3-64 创建 Primitive Group Constraints	44
图 3-65 创建 Relative Group Constraints.....	45
图 3-66 创建 Resource Reservation 约束	45
图 3-67 复制 Chip Array 中的位置设置 Resource Reservation 约束位	46
图 3-68 创建 Clock Assignment 约束.....	47
图 3-69 修改 Clock Assignment 约束.....	47
图 3-70 创建 Quadrant Constraints	48
图 3-71 修改 Quadrant Constraints	49
图 3-72 创建 Hclk Constraints 约束	50
图 3-73 修改 Hclk Constraints 约束	50
图 3-74 创建 Vref Constraints.....	51
图 3-75 Vref 重命名检查	51
图 3-76 拖动到 Chip Array 中修改 Location.....	52
图 3-77 拖动到 Package View 中设置 Vref 约束 Location.....	52
图 4-1 时序分析基本模型.....	53
图 4-2 STA 四类时序路径	54
图 4-3 Process 窗口	56
图 4-4 打开新建时序约束文件	57
图 4-5 新建时序约束文件.....	57
图 4-6 打开时序约束文件.....	58
图 4-7 时序约束编辑器界面	58
图 4-8 Netlist Tree 窗口	59
图 4-9 约束编辑界面	59
图 4-10 菜单打开时序约束界面	60

图 4-11 右键打开时序约束界面.....	60
图 4-12 编辑 SDC 文件.....	61
图 4-13 创建 Clock 约束	62
图 4-14 Objects 列表	63
图 4-15 创建 clock 约束.....	64
图 4-16 Clock 约束列表	64
图 4-17 Clock 列表右键内容.....	64
图 4-18 创建 Generated Clock 约束	65
图 4-19 Clock 列表右键内容.....	66
图 4-20 创建 Clock Latency 约束.....	67
图 4-21 创建 Clock Uncertainty 约束	68
图 4-22 创建 Clock Group 约束	68
图 4-23 Clock Group 成员列表	69
图 4-24 创建 I/O Delay 约束	70
图 4-25 创建 False Path 约束	71
图 4-26 创建 Max/Min Delay 约束	71
图 4-27 创建 Multicycle Path 约束	72
图 4-28 创建 Operating Conditions 约束	73
图 4-29 Report Timing 创建界面.....	73
图 4-30 Report Timing 界面	74
图 4-31 Report High Fanout Nets 创建界面	74
图 4-32 Report High Fanout Nets 界面	75
图 4-33 Report Route Congestion 创建界面	75
图 4-34 Report Route Congestion 界面	76
图 4-35 Report Min Pulse Width 创建界面	76
图 4-36 Report Min Pulse Width 界面	77
图 4-37 Report Exception 创建界面	77
图 4-38 Report Max Frequency 界面.....	78
图 4-39 Report Exception 创建界面	78
图 4-40 Report Exception 界面.....	79
图 4-41 查看时序报告界面.....	79

图 5-1 生成.posp 文件设置	82
图 5-2 读取.posp 文件	83
图 5-3 读取时序约束文件.....	83
图 5-4 高亮关键路径的 Module 操作	84
图 5-5 关键路径的 Module 显示红色	84
图 5-6 关键路径信号流向示意图.....	85
图 5-7 调整后的位置信息.....	85

表目录

表 1-1 术语、缩略语.....	2
-------------------	---

1 关于本手册

1.1 手册内容

本手册主要描述高云半导体物理约束和时序约束的内容，介绍 Gowin 云源软件物理约束和时序约束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和时序约束。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 1.8.1Beta 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 适用产品

本手册中描述的信息适用于以下产品：

- GW2A 系列 FPGA 产品：GW2A-18, GW2A-55
- GW2AR 系列 FPGA 产品：GW2AR-18
- GW1N 系列 FPGA 产品：GW1N-1, GW1N-2, GW1N-2B, GW1N-4, GW1N-4B, GW1N-6, GW1N-9
- GW1NR 系列 FPGA 产品：GW1NR-4, GW1NR-4B, GW1NR-9
- GW1NS 系列 FPGA 产品：GW1NS-2, GW1NS-2C
- GW1NZ 系列 FPGA 产品：GW1NZ-1
- GW1NSR 系列 FPGA 产品：GW1NSR-2C、GW1NSR-2

1.3 相关文档

通过登录高云半导体网站 www.gowinsemi.com.cn 可下载、查看以下相关文档：

1. GW2A 系列 FPGA 产品数据手册
2. GW2AR 系列 FPGA 产品数据手册
3. GW1N 系列 FPGA 产品数据手册
4. GW1NR 系列 FPGA 产品数据手册
5. GW1NS 系列 FPGA 产品数据手册
6. GW1NZ 系列 FPGA 产品数据手册
7. GW1NSR 系列 FPGA 产品数据手册
8. Gowin 云源软件用户指南

9. GW1N 系列 FPGA 产品封装与管脚手册
10. GW1NR 系列 FPGA 产品封装与管脚手册
11. GW2A 系列 FPGA 产品封装与管脚手册
12. GW2AR 系列 FPGA 产品封装与管脚手册
13. GW1NS 系列 FPGA 产品封装与管脚手册
14. GW1NZ 系列 FPGA 产品封装与管脚手册
15. GW1NSR 系列 FPGA 产品封装与管脚手册

1.4 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

1.5 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

E-mail：support@gowinsemi.com

2 简介

Gowin 云源软件约束编辑器分为物理约束编辑器（**FloorPlanner**）和时序约束编辑器（**Timing Constraints Editor**）。用户可通过编辑器进行物理约束和时序约束。

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工具，支持对 **I/O**、**Primitive**（原语）、**block**（**B-SRAM**、**DSP**）、**Net**、**Group** 等的属性及位置信息的读取与修改功能，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，有效地提高编写物理约束文件的效率。

时序约束编辑器，是一种能够创建和修改时序约束文件的工具。可提供高效的网表单元查找功能，能够有效地提高编写时序约束文件的效率。

时序约束编辑器有如下特点：

根据特定的时序模型通过信号传输路径计算延时值，再将其与预计值进行比较，判断用户设计是否满足时序要求，通过进行时序约束提高设计的工作速率和稳定性；

提供默认的基础时钟，以及对跨时钟域的时序分析，提供两种格式的时序报告：基于网页（**HTML**）和基于文本格式，默认为网页格式；

支持业界通用的时序约束，包括时钟相关约束、输入输出端口约束和时序路径约束等，并支持业界通用的时序报告指令；

对所支持约束规定业界通用的优先级；

默认对建立时间（**setup time**）、保持时间（**hold time**）、恢复时间（**recovery time**）、移除时间（**removal time**）和最小时钟脉冲（**MPW**）等进行检查，可通过时序报告指令定制时序报告的具体内容。

3 物理约束

Gowin FloorPlanner 是一种能够创建和修改物理约束文件的工具。用于提供表格化的约束编辑和高效的网表单元查找功能，能够有效地提高编写物理约束文件的效率。

3.1 功能简介

Gowin FloorPlanner 功能特点：

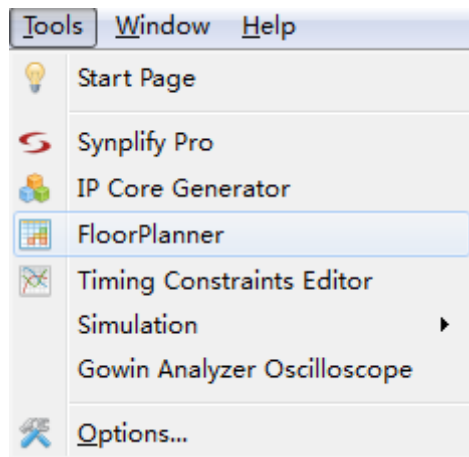
- 支持用户设计文件、约束文件的读入，以及约束文件的输出；
- 支持对用户设计文件中 IO Port、Primitive、Net 约束信息等的显示；
- 支持用户新建、编辑、修改约束信息；
- 支持 Chip Array 的网格模式、宏单元模式以及原语模式显示；
- 支持依据 Package 信息的 Package View 显示；
- 支持 Chip Array 和 Package View 的同步显示；
- 支持约束位置信息的实时显示及区别显示；
- 支持拖拽设置位置信息的功能；
- 支持 One Hit Drag 一键产生约束的功能；
- 支持 IO Port 的属性配置功能，支持批量配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

3.2 启动 FloorPlanner

可通过以下两种方式启动 FloorPlanner：

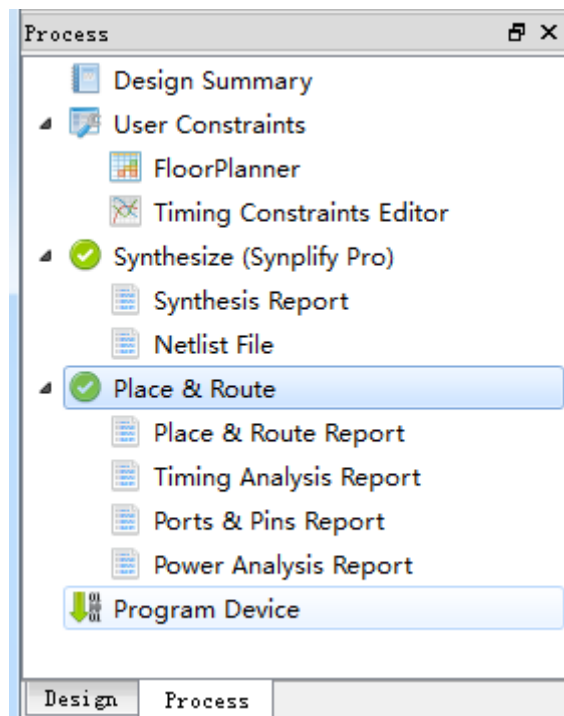
单击“IDE > Tools”，打开“FloorPlanner”，如图 3-1 所示。

图 3-1 菜单栏启动 FloorPlanner



建立 RTL 软件工程后，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。

3.3 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/Net 的位置信息等，可通过以下两种方式完成物理约束文件：

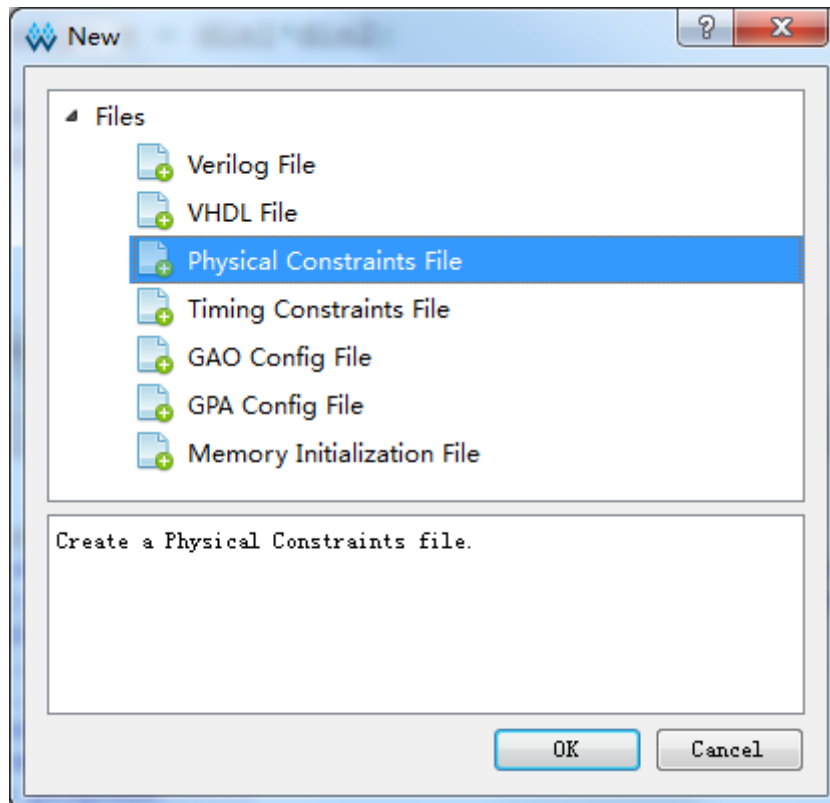
1. 手动书写；
2. 通过 Gowin FloorPlanner 工具输出约束文件。

3.3.1 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。

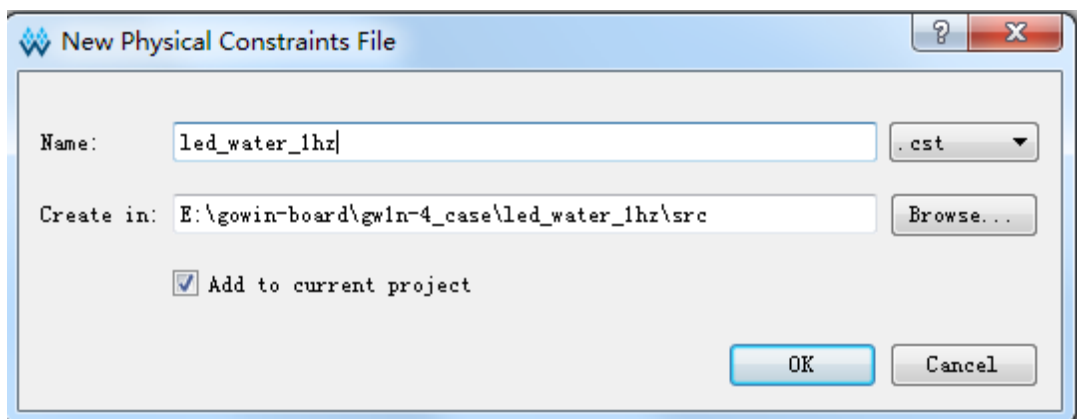
图 3-3 打开新建物理约束



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。

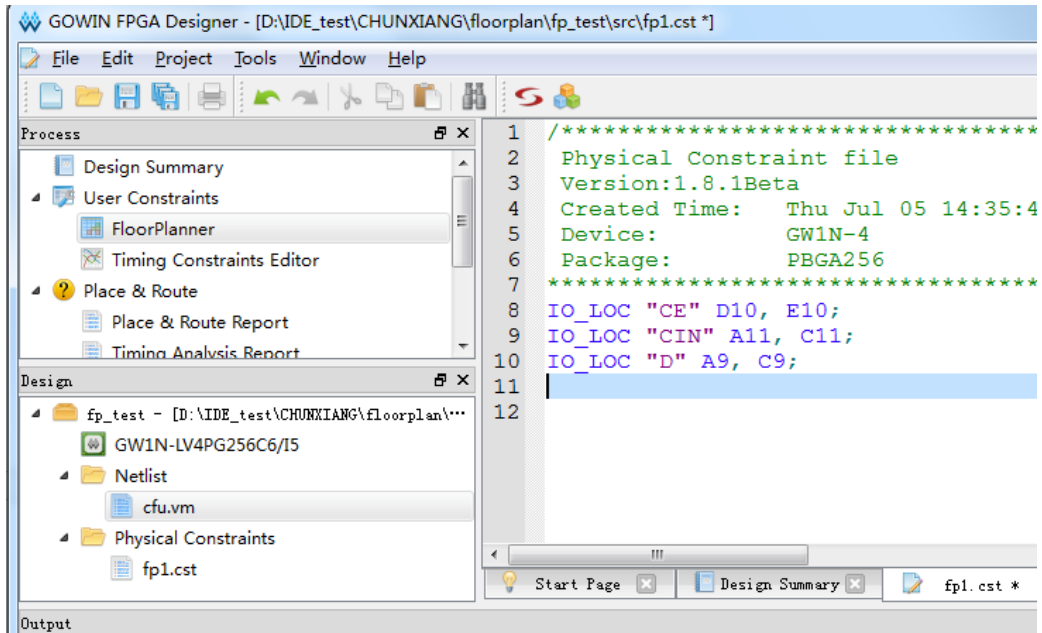
图 3-4 新建物理约束文件



- **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。手写物理约束文件，如图 3-5 所示。

图 3-5 手写物理约束文件



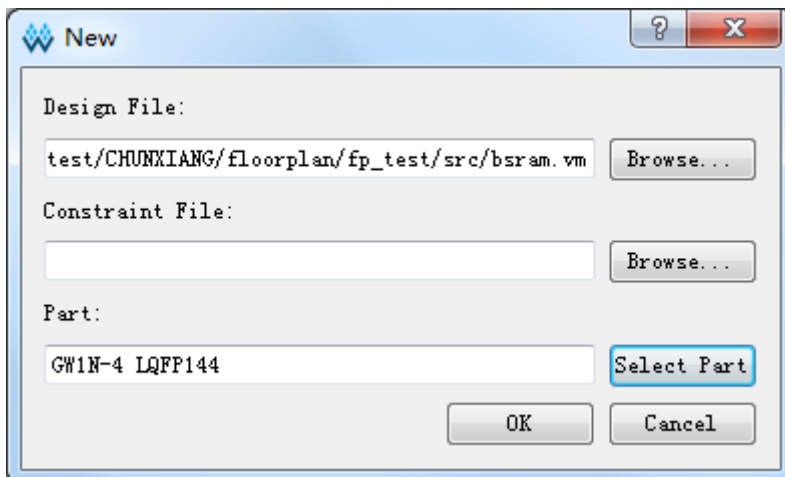
3.3.2 FloorPlanner 输出约束文件

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 3.2 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；
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



注！

启动 FloorPlanner 采用 3.2 启动 FloorPlanner 介绍中的第一种方式；
亦可通过以下两种方式打开“New”对话框：

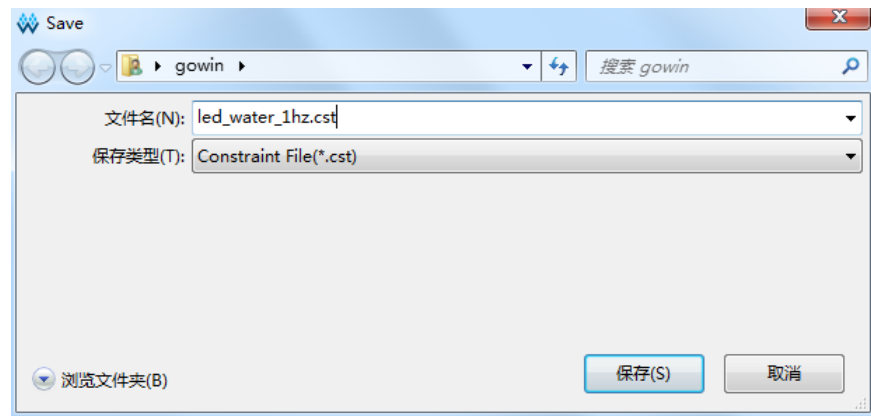
- 使用快捷键 **Ctrl + N**；
- 单击工具栏上的“New”图标。

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. 单击工具栏中的“Save”图标，即可输出约束文件；

在弹出的“Save”对话框中，可修改文件名，如图 3-7 所示。

图 3-7 保存输出文件

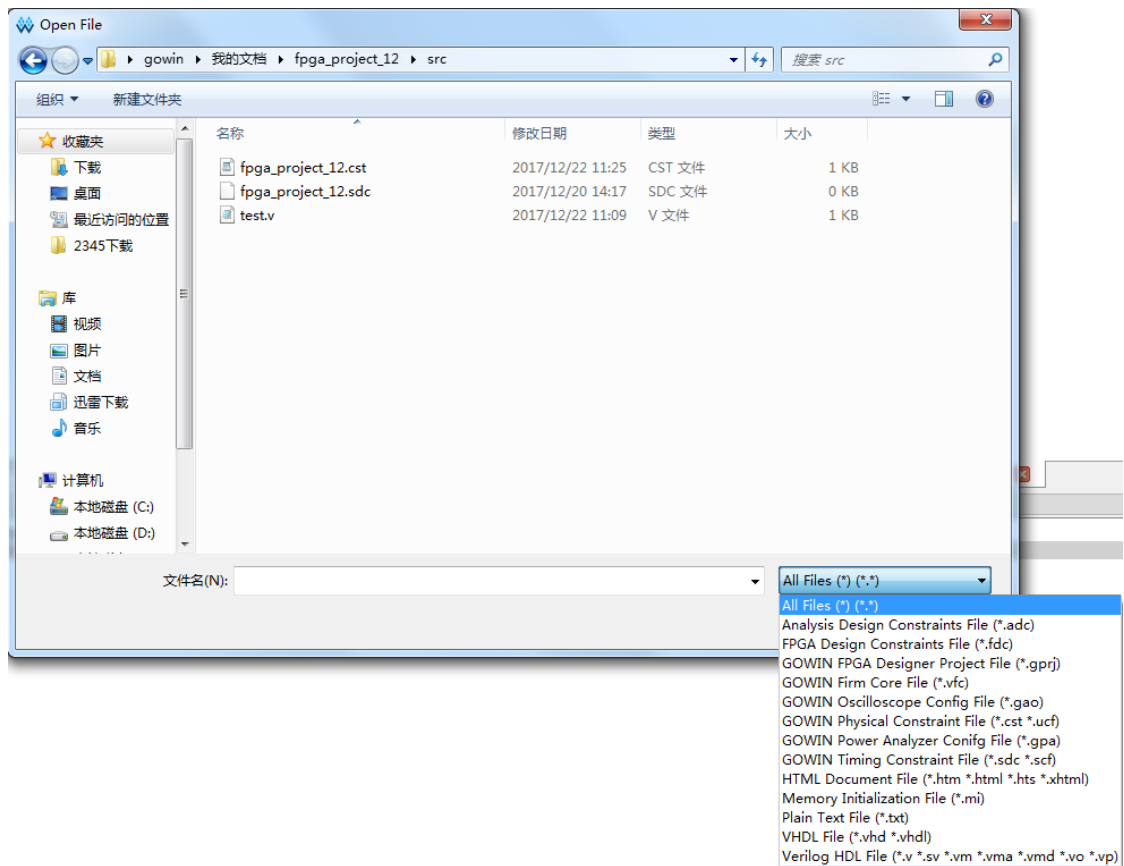


3.3.3 打开约束文件

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 3-8 所示。

图 3-8 打开物理约束



注！

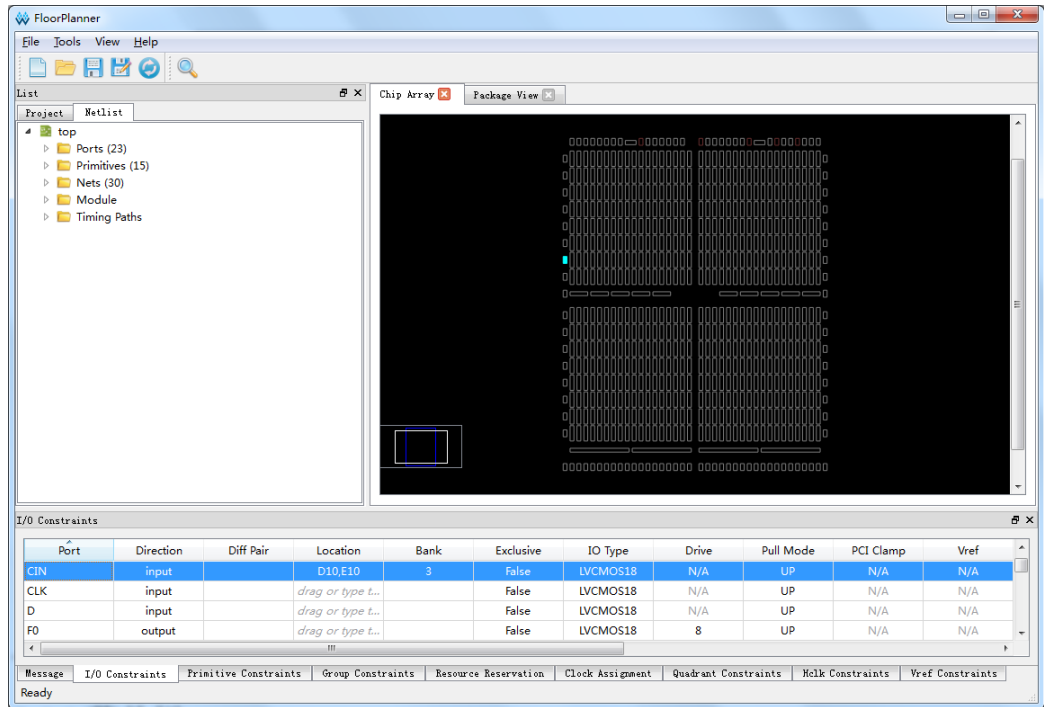
- 亦可通过以下两种方式打开“Open File”对话框：
- 使用快捷键 **Ctrl + O**；
- 单击工具栏上的“Open”图标。
- 选择物理约束文件所在的目录，打开选中文件。

3.4FloorPlanner 界面

新建或打开 FloorPlanner 界面（包含网表文件），如图 3-9 所示。

界面包括菜单栏、List 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 3-9 FloorPlanner 界面



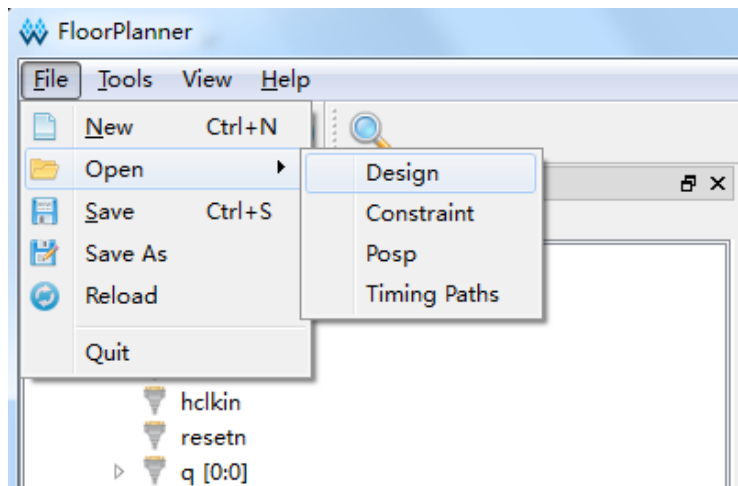
3.4.1 菜单栏

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

File 菜单

File 菜单界面如图 3-10 所示，各子部分介绍如下：

图 3-10 File 菜单



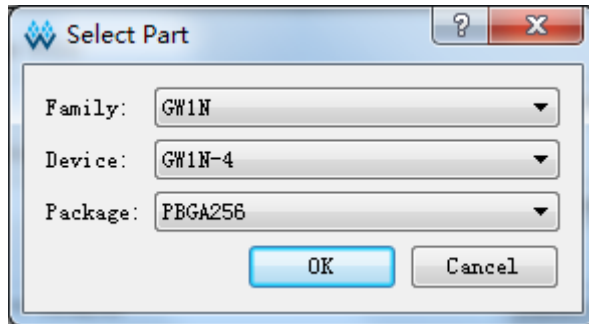
- **New:** 新建工程，添加用户设计、约束，设置芯片信息等，如图 3-6 所示；
- **Open:** 打开用户网表文件、约束文件、器件布局信息文件或时序路径文件；
- **Reload:** 当在磁盘或工程中对 **cst** 文件进行修改保存后，可以重新加载 **cst** 约束文件；

- **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。

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 3-11 所示。

图 3-11 选择器件



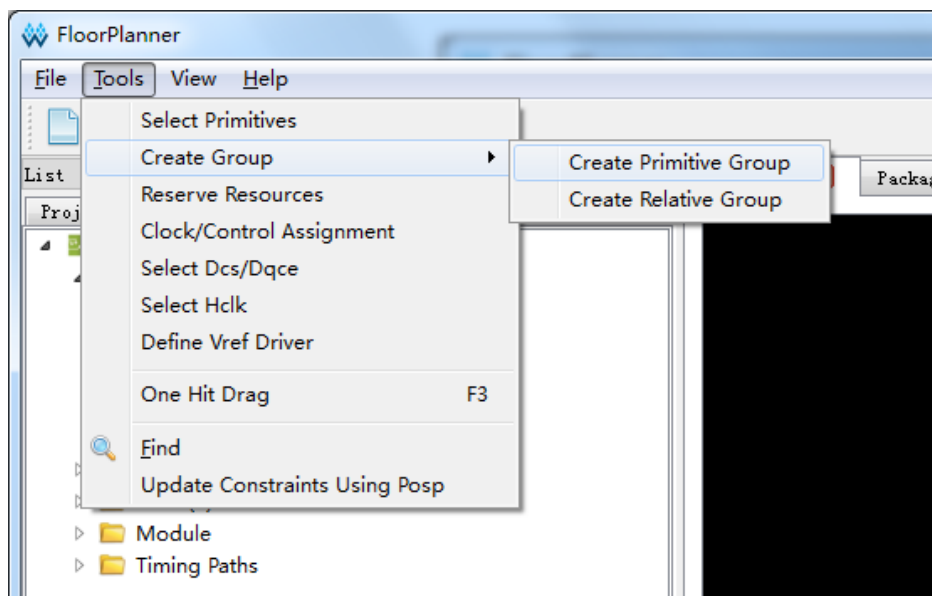
Tools 菜单

Tools 菜单界面如图 3-12 所示。Tools 菜单功能如下:

- 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:**

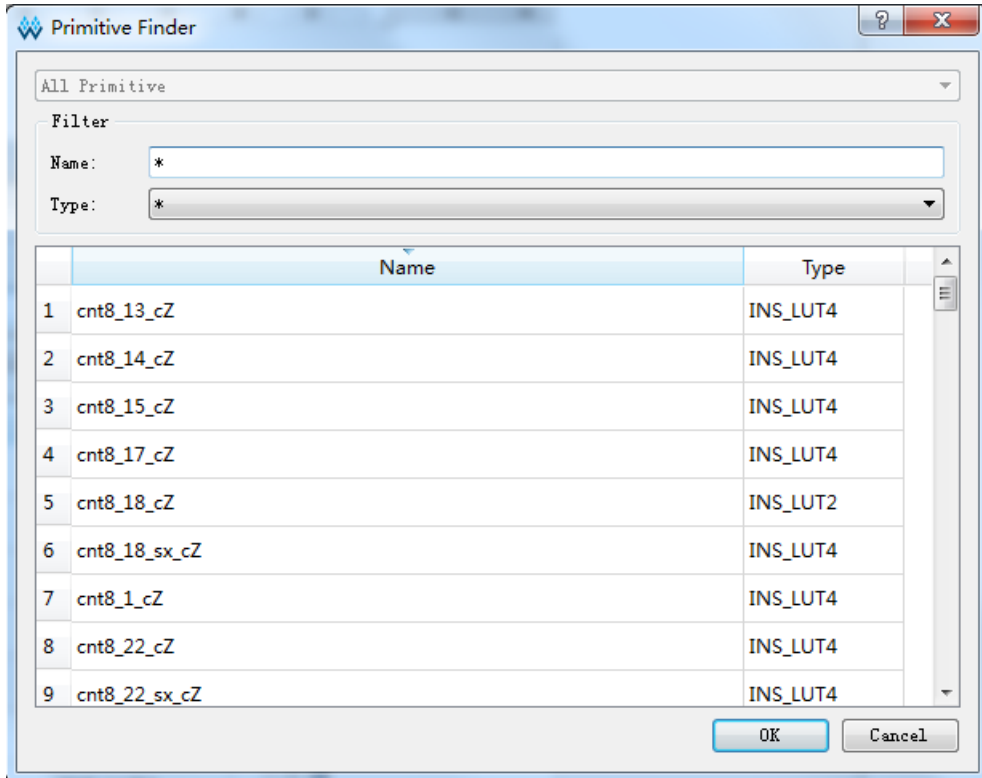
选择 **Primitive** 创建对应的约束，单击该菜单弹出如图 3-13 所示对话框；

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** 窗口中呈黄色高亮显示。

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** 组约束，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；
3. 通过“+”和“-”两个按钮实现 **Primitive** 的添加和删除，正确创建的原语组如图 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。
4. 原语组创建配置完成后，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。
- 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。

在编辑窗口中，双击，重新打开如图 3-15 所示的对话框，进行编辑。

图 3-14 新建原语组

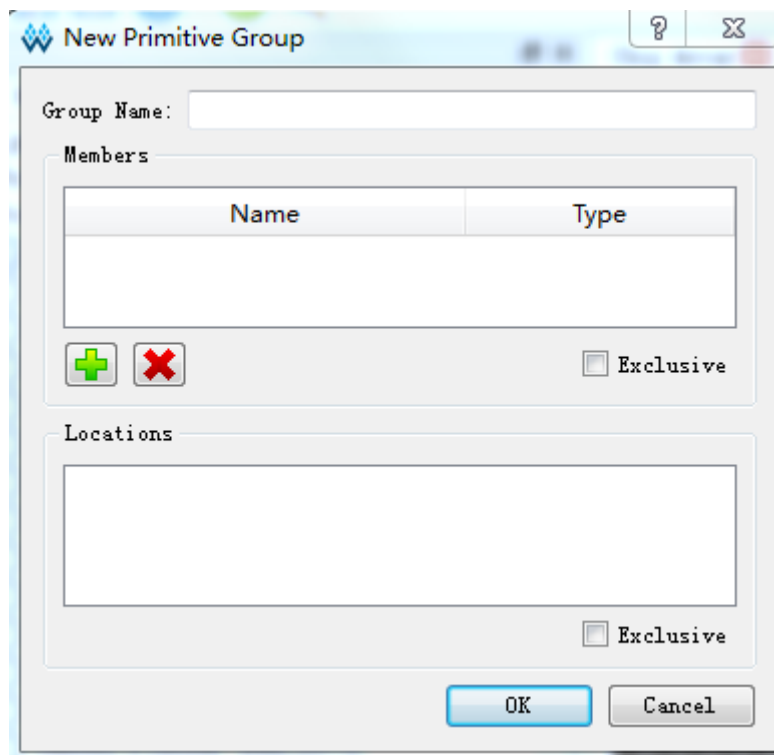


图 3-15 正确原语组界面

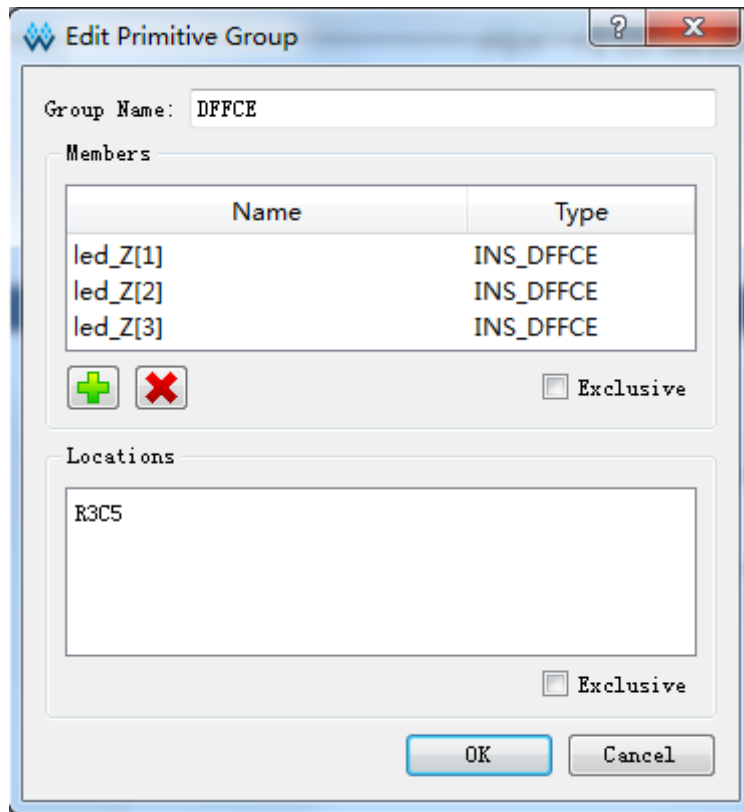


图 3-16 无效位置

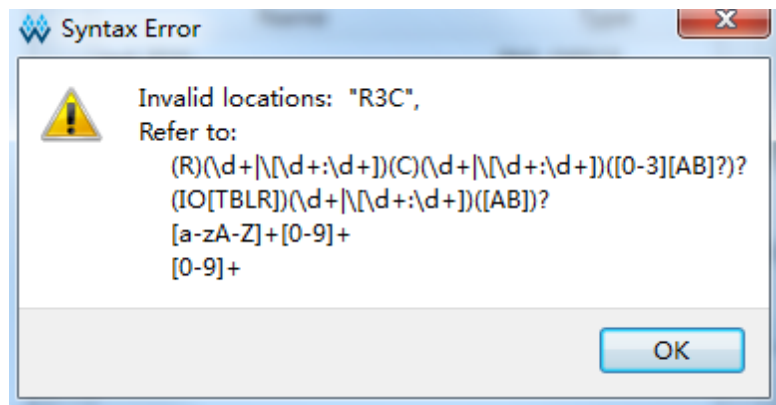
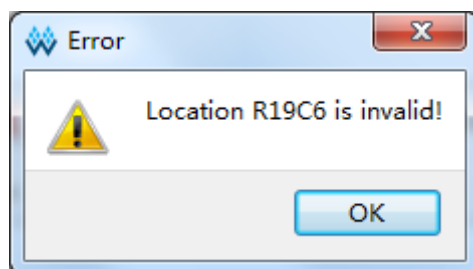


图 3-17 无效位置



- **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；
3. 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。
- 4. 配置完成后单击“OK”，产生约束信息，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，重新打开如图 3-19 所示的对话框，进行编辑。

图 3-18 创建相对位置的组

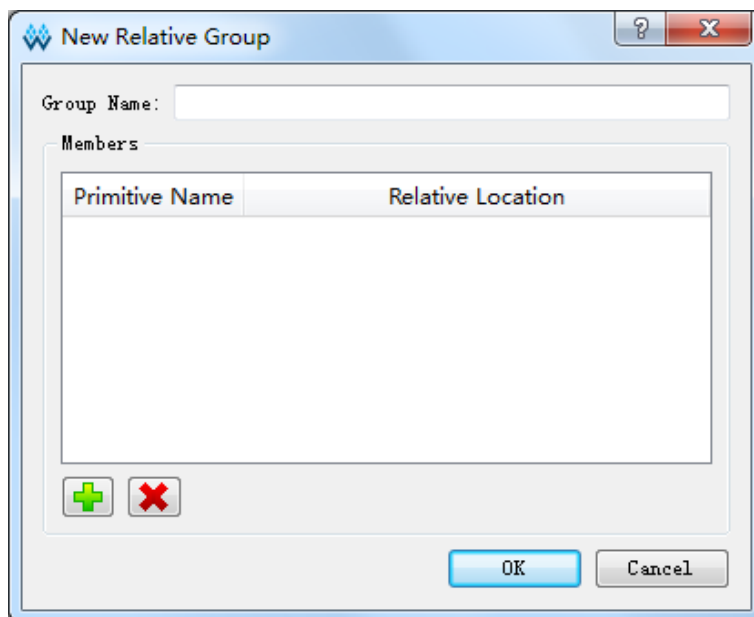
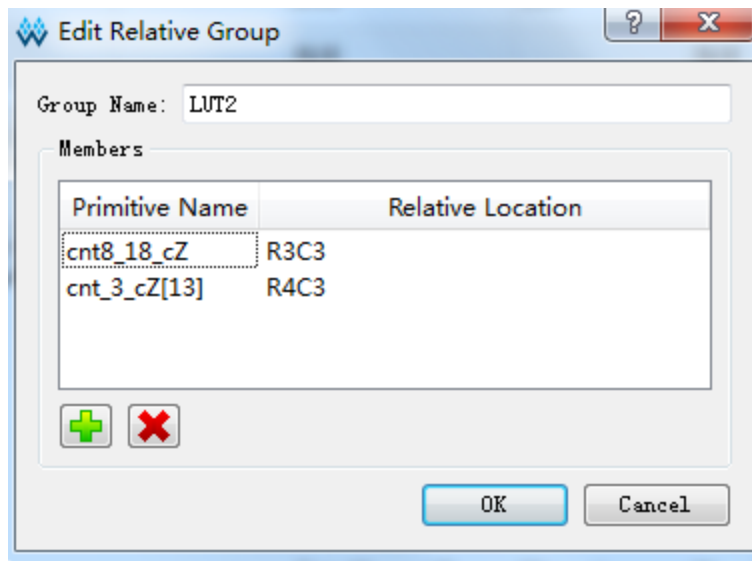


图 3-19 正确的相对组界面



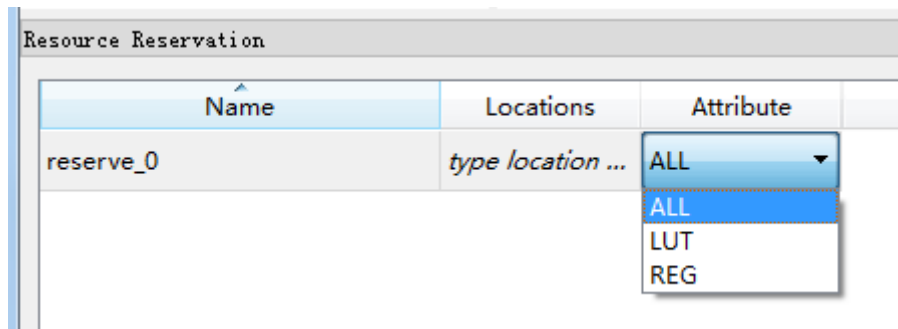
Reserve Resources:

- 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.双击位置栏，可输入约束位置；
- 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使用率约束，不可修改该名称。

图 3-20 预留约束

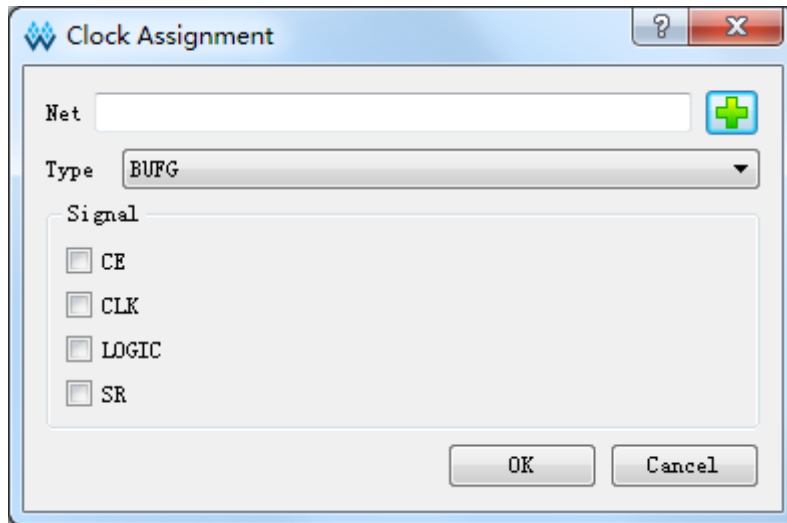


Clock/Control Assignment

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”；
- 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。

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 和图 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 器件；
- 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，进行编辑。

注！

若设计无 DCS/DQCE 器件则无法添加。

GW2A-18, GW2AR-18, GW2A-55 系列打开的“Quadrant Constraints”约束编辑窗口图 3-23 所示。

图 3-22 象限约束 (GW1N 家族)

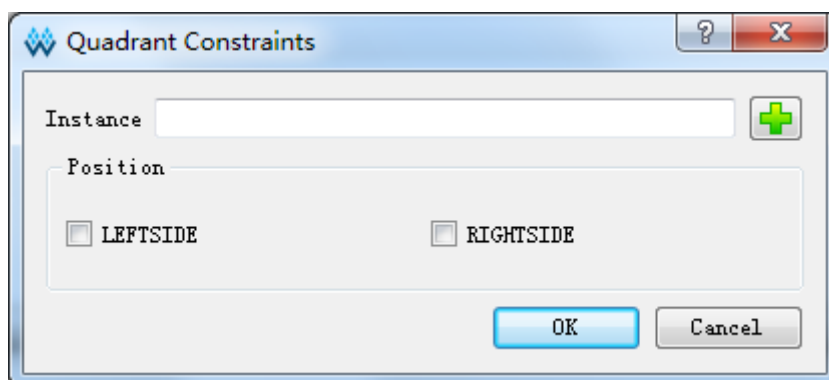
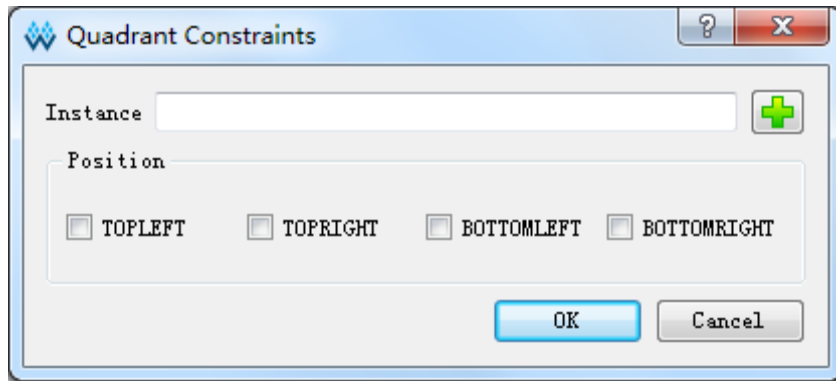


图 3-23 象限约束 (GW2A 家族)



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  按钮选择相应的器件；

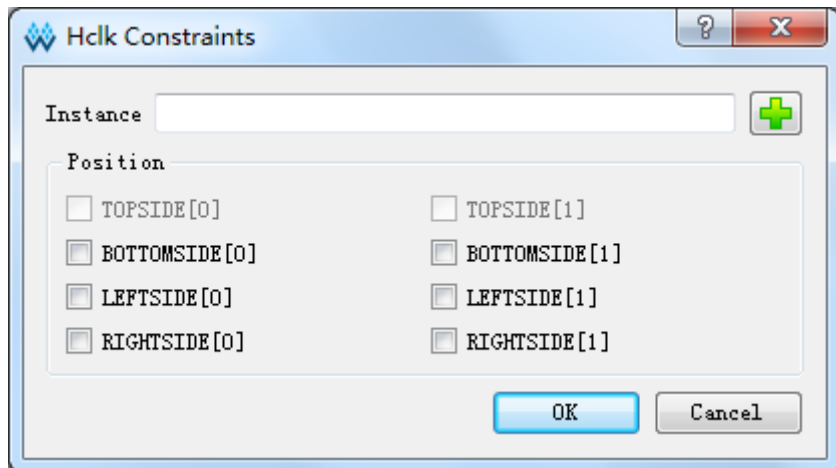
通过 Position 下的复选框配置象限位置。

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，进行编辑。

注！

- 若设计中无符合的器件则无法添加；
- Position 根据工程中 device 不同可用 Position 不同。

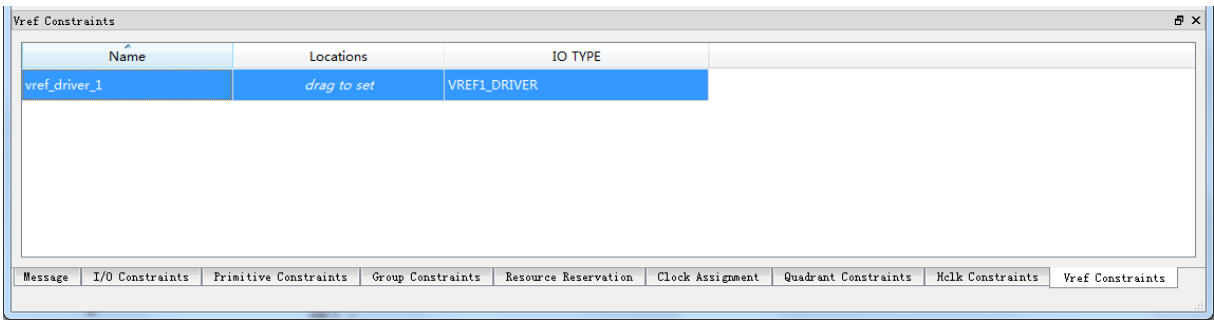
图 3-24 Hclk 约束



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

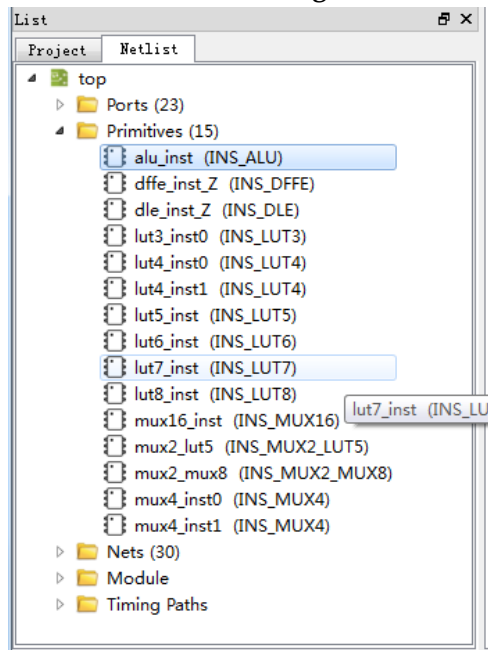


图 3-27 选择 One Hit Drag 位置

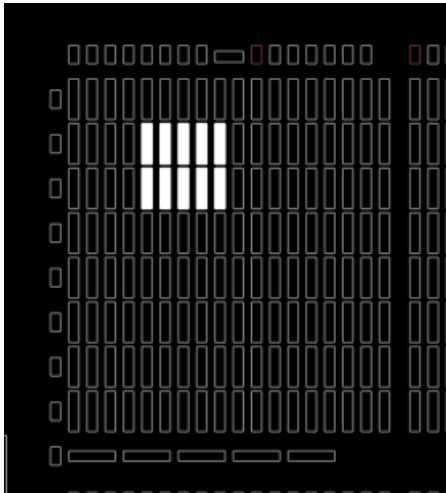


图 3-28 One Hit Drag 产生约束

Primitive Constraints			
Primitive	Type	Locations	Exclusive
alu_inst	INS_ALU	R[3:4]C[6:10]	False

Mes... I/O Constr... Primitive Constr... Group Constr... Resource Reserva... Clock Assign... Quadrant Constr...

Find

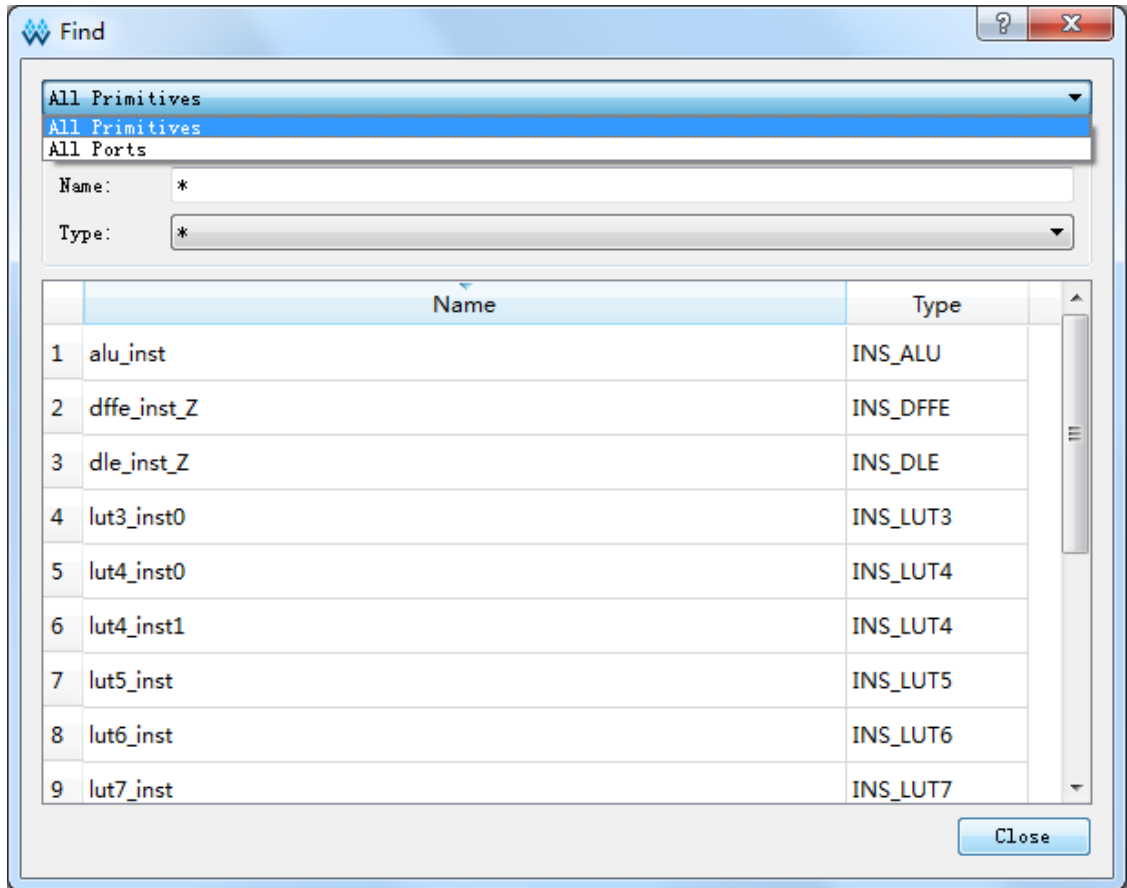
快速查找 Primitive、IO Port 信息，并可在相应的 primitive 或 port 上通过右键菜单编辑对应的约束信息，单击该菜单，弹出如图 3-29 所示对话框。

相关操作如下所示：

通过选择 “All Primitive” 或 “All Port”，进行相应查找；

在对应的条目上，右键单击 “Edit *** Constraint”，可在主界面底部的窗口中编辑约束信息。

图 3-29 查找界面



Update Constraints Using Posp

将约束信息更新为器件布局信息文件中的位置信息。

View 菜单

- 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 Out: 用于缩小 Chip Array 窗口或 Package View 窗口;
- Zoom In: 用于放大 Chip Array 窗口或 Package View 窗口;
- Zoom Fit: 按照窗口大小缩放 Chip Array 窗口或 Package View 窗口。

图 3-30 View 菜单

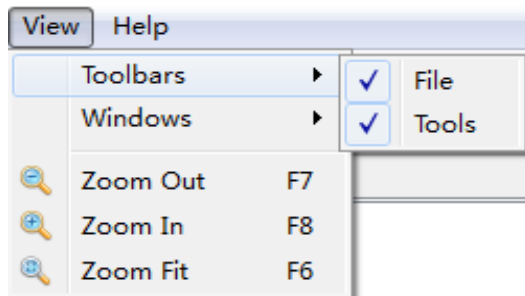
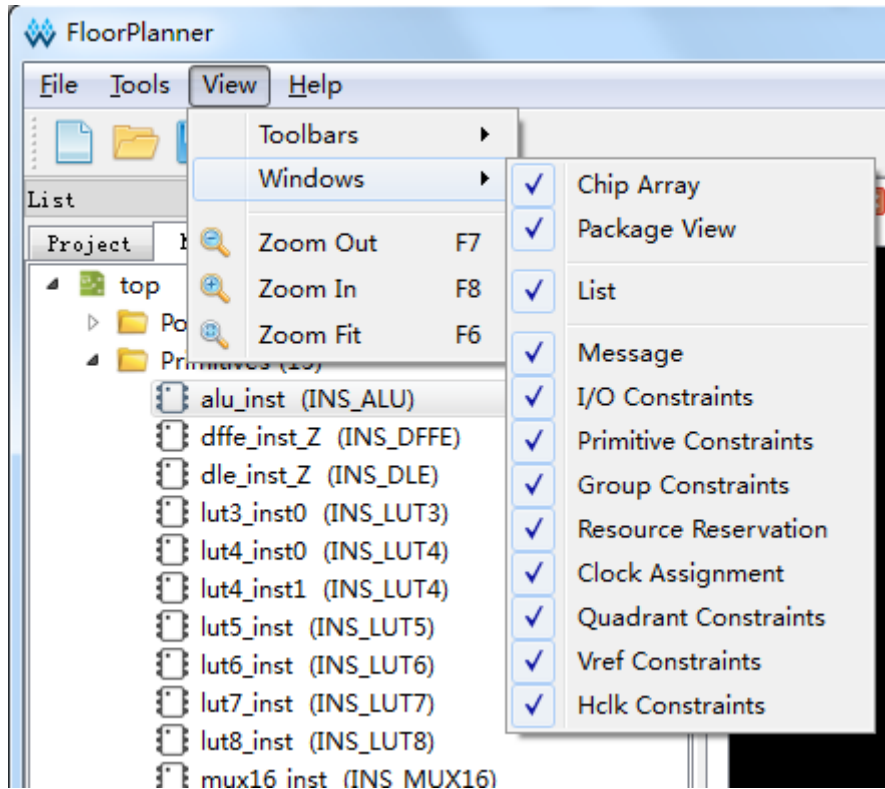


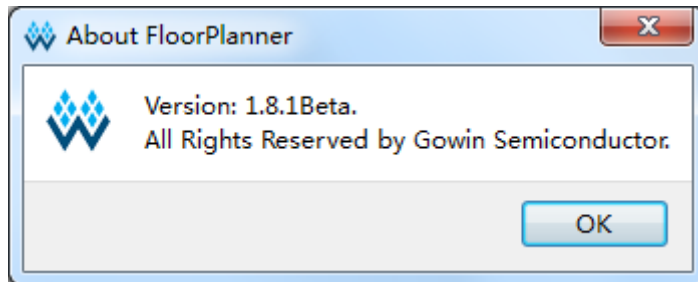
图 3-31 Windows 菜单



Help 菜单

Help 菜单用于提示软件的版本号及版权信息。单击“About”弹出如图 3-32 所示的提示框。

图 3-32 Help 提示框



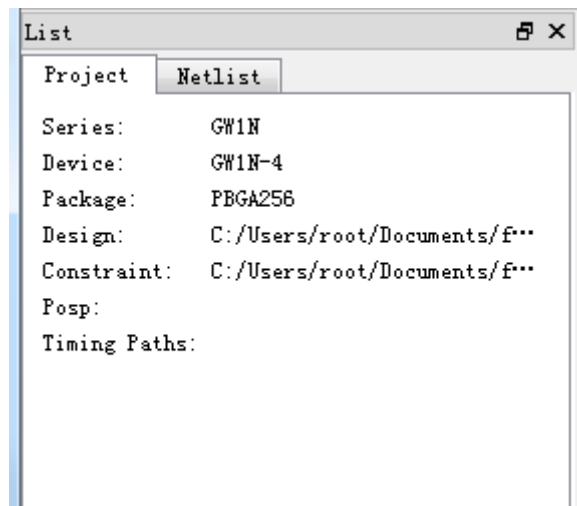
3.4.2List 窗口

List 窗口包括 Project 和 Netlist 两个子窗口。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 3-33 所示。用于显示当前工程中所用的芯片信息，如 Series、Device、Package，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 3-33 Project 窗口



Netlist 窗口

Netlist 窗口界面如图 3-34 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 3-35 所示；
- Module 采用层级的方式显示，当鼠标悬留在 Module 列表的某个 module 上时，可显示各 Module 中各类型的 Instance 数目，如图 3-36 所示；
- 时序路径则按照 Slack 时间从小到大的顺序显示，如图 3-37 所示。

图 3-34 Netlist 窗口

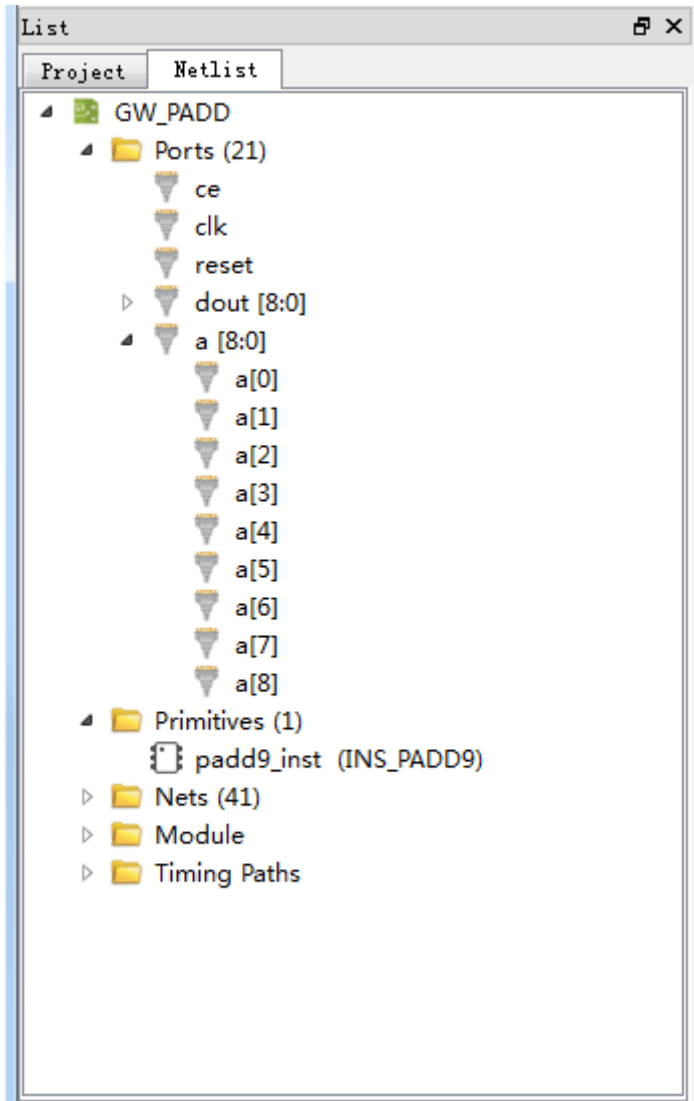


图 3-35 BUS 和非 BUS 结合显示

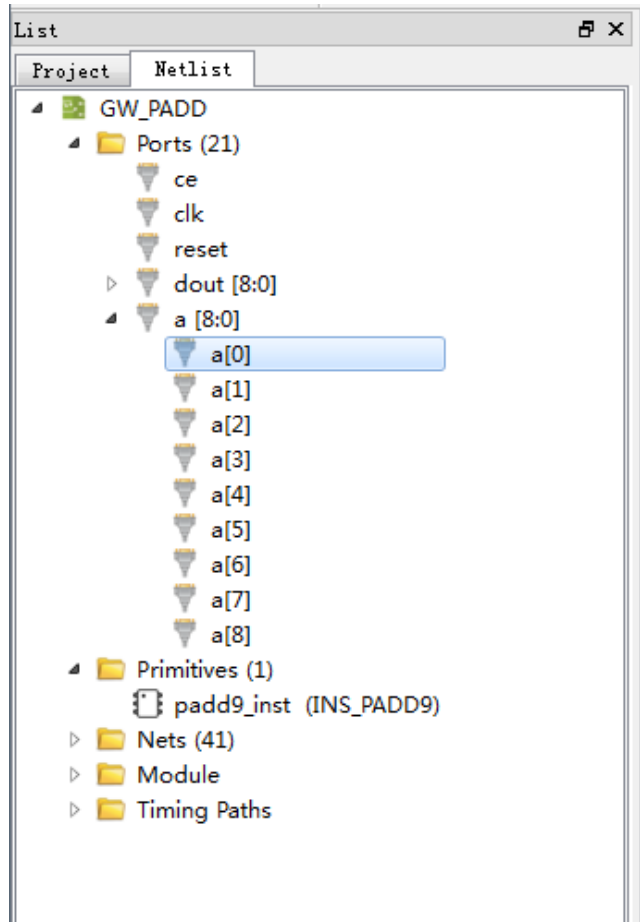


图 3-36 层级显示

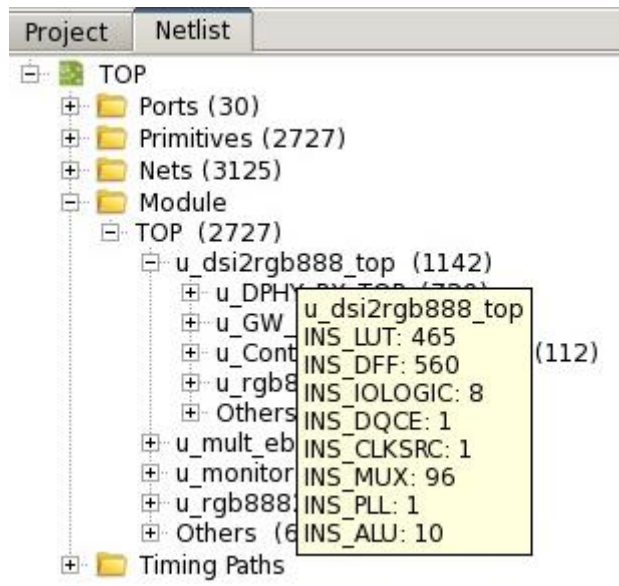
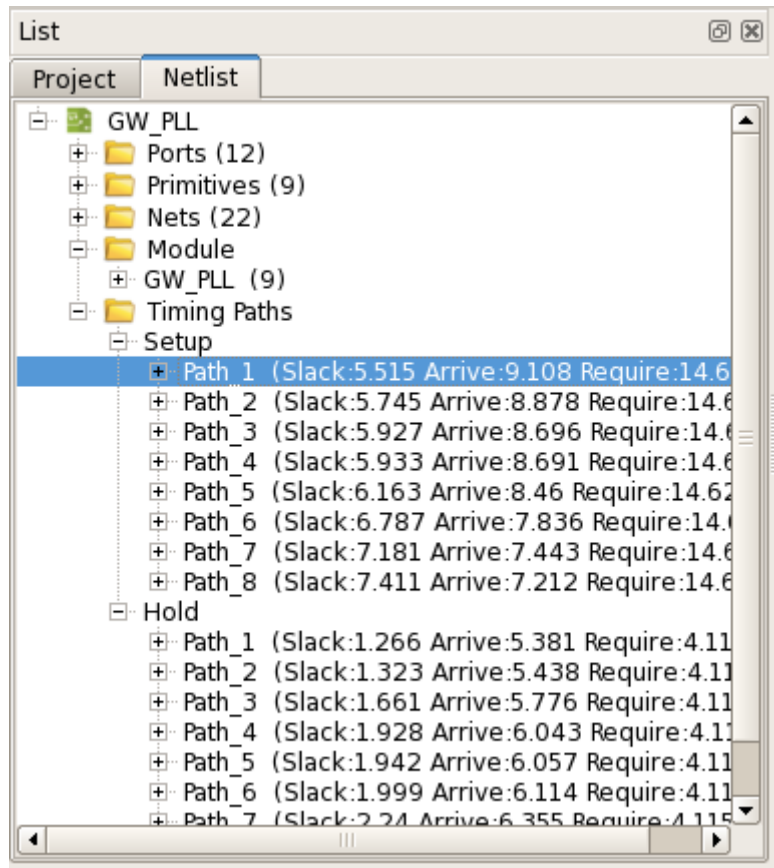


图 3-37 时序路径显示



Netlist 窗口

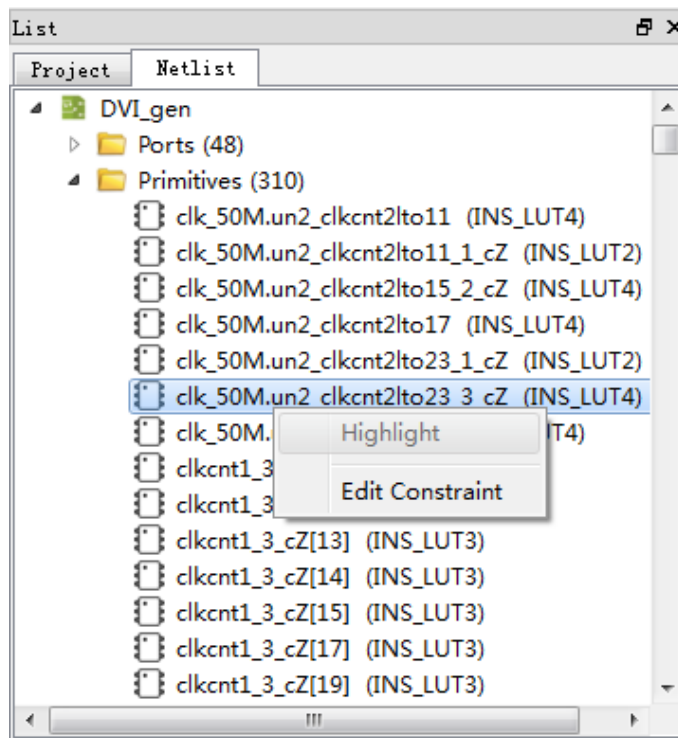
Netlist 窗口提供右键菜单功能，具有如下功能：

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

注！

如当前 Primitive、Net 或 Port 无位置约束，则高亮显示功能不可用，如图 3-38 所示。

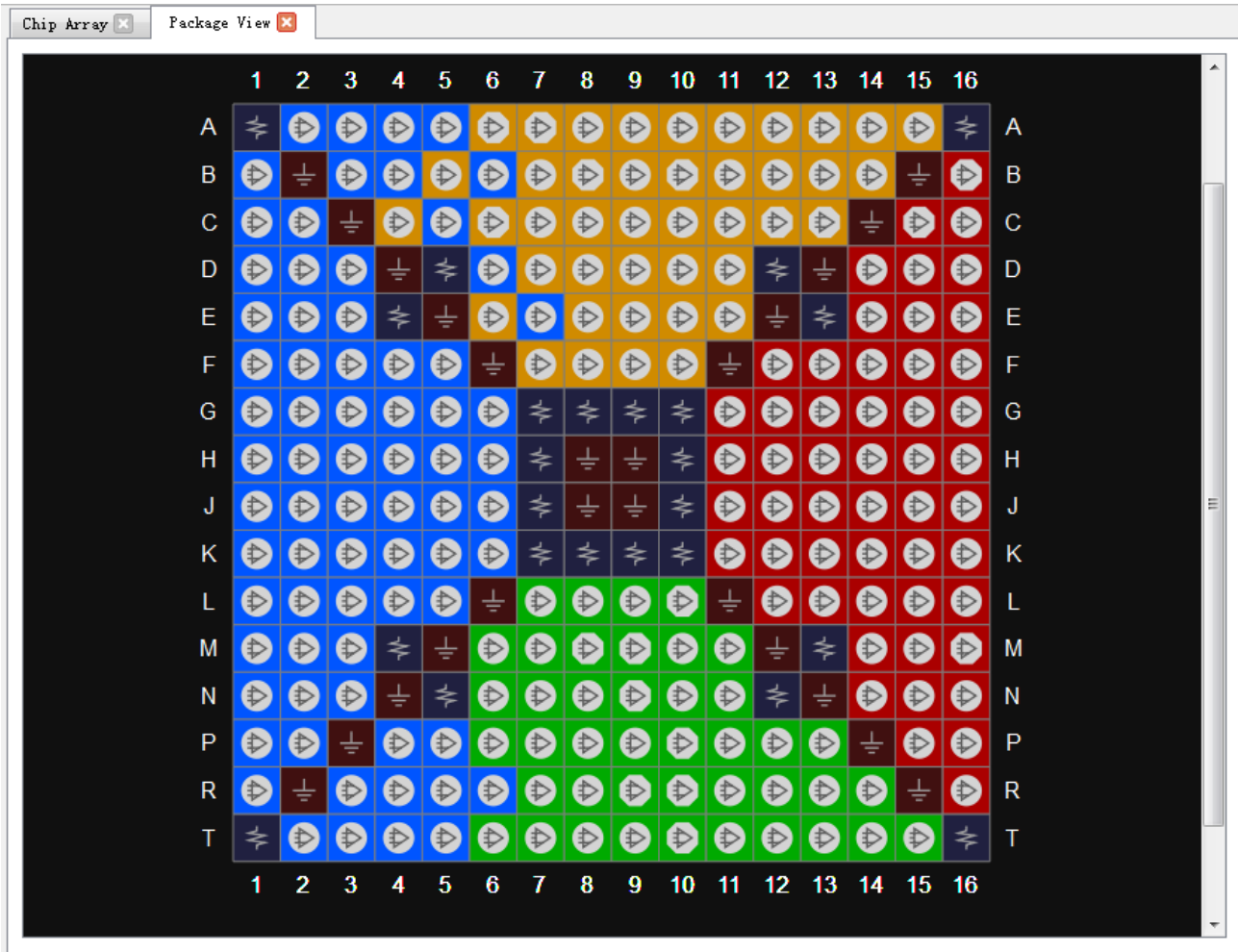
图 3-38 Netlist 右键功能









3.4.3 Package View 窗口

Package View 窗口界面如图 3-39 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 3-39 Package view 界面



用户 I/O 和有复用功能的 I/O、电源、地和专用管脚使用不同的符号和颜色来区分。GW1N 系列 FPGA 产品管脚示意图中管脚定义如下所示：

- 红色填充表示 BANK0 的管脚，绿色填充表示 BANK1 的管脚；
- 蓝色填充表示 BANK2 的管脚，黄色填充表示 BANK3 的管脚；
-  表示 BANK0 中的单端和差分 I/O，填充颜色随 BANK 变化；
-  表示 BANK1 中和下载配置相关的复用 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；
-  表示专用管脚；
-  表示 NC。

Package View 支持右键菜单如图 3-40 所示，相关功能如下：

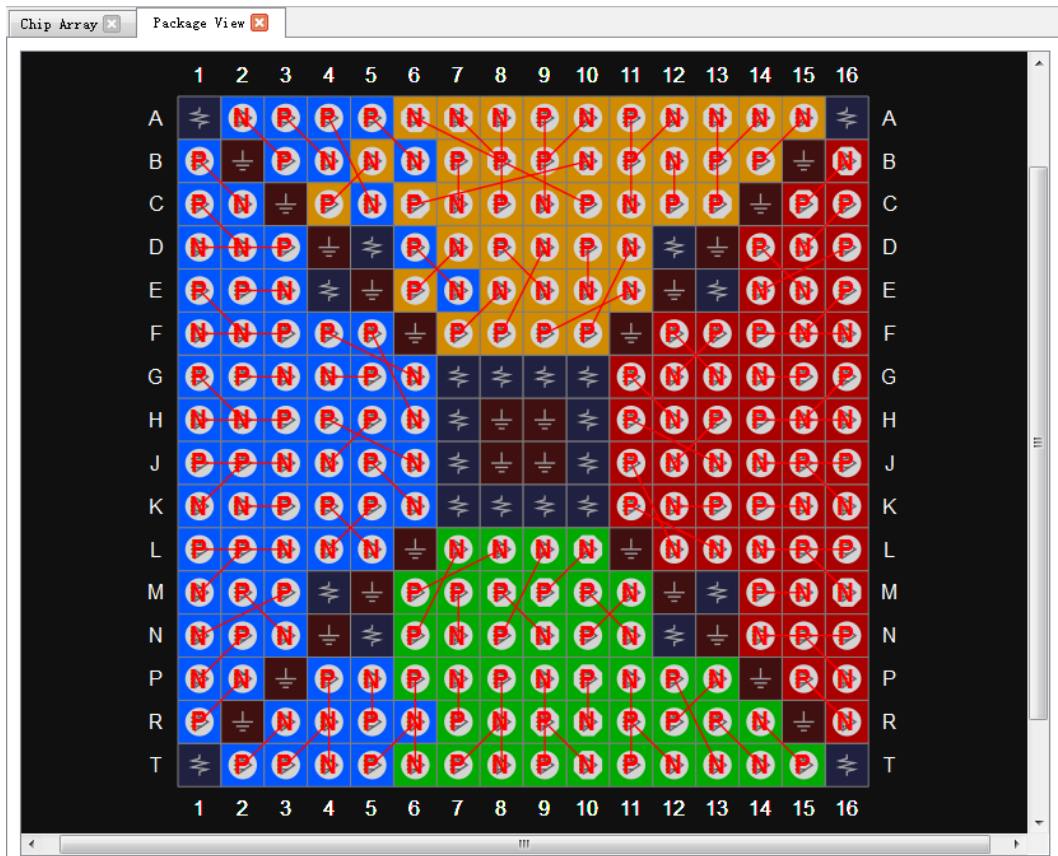
- 支持视图的放大、缩小，差分对的显示；
- Top 和 Bottom View 的切换显示，默认显示 Top View。

图 3-40 Package View 右键功能



Package View 中，可通过右键菜单选择“Show Differential IO Pairs”显示差分对，如图 3-41 所示，红线相连的为一对差分对。

图 3-41 差分对显示




Package View 支持 IO Port 约束位置的显示，可通过以下两种方式设置约束位置：

在 Package View 内部任意拖拽 IO Port 的约束位置；

从“Netlist 窗口”或底部“I/O Constraints”窗口中，将 IO Port 拖动到 Package View 窗口。

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。

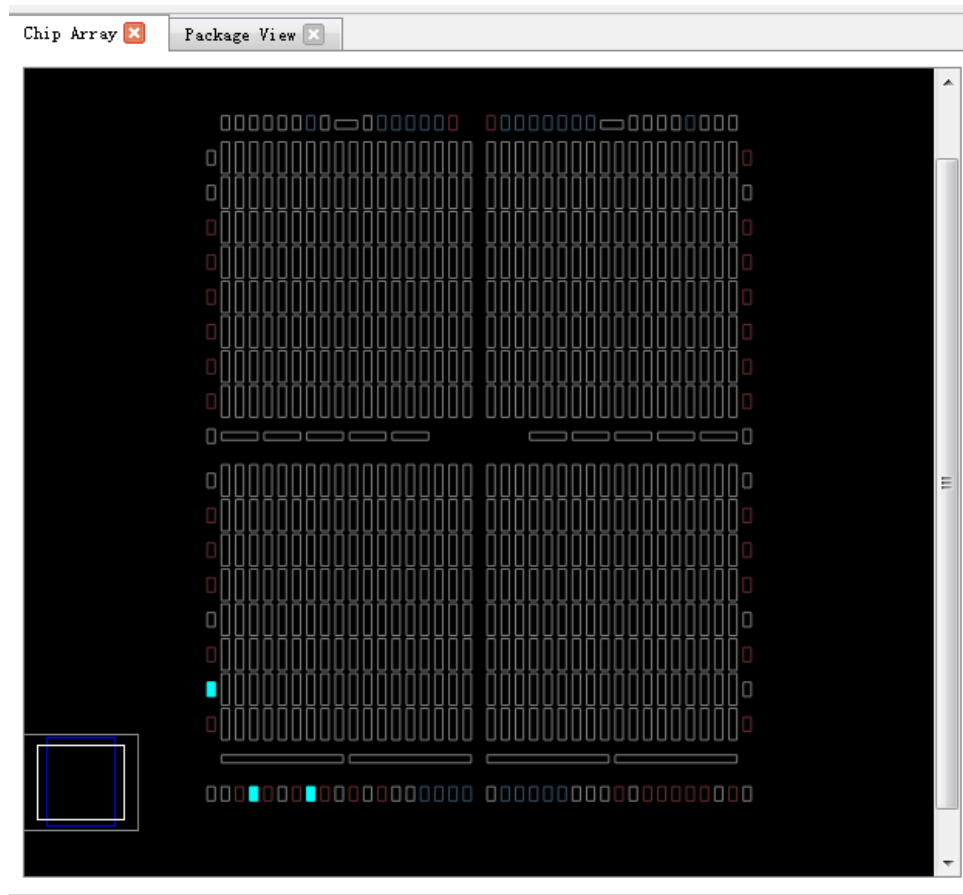
3.4.4 Chip Array 窗口

FloorPlanner 的 Chip Array 窗口界面如图 3-42 所示, Chip Array 窗口根据芯片的行列信息显示芯片的 IOB、CFU、DSP 等的分布, 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: 该封装对应的封装 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-4B、GW1NR-9、GW1NR-9ES、GW1NSR-2C 器件, 则会有蓝色标记 IOB, 表示内嵌 SDRAM 配置 IO 位置。

图 3-42 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 3-43 所示;
- 宏单元模式: 以 cls、block 等单位显示约束位置, 如图 3-44 所示;
- 原语模式: 以 reg、lut 等单位显示约束位置, 如图 3-45 所示。

图 3-43 网格模式约束

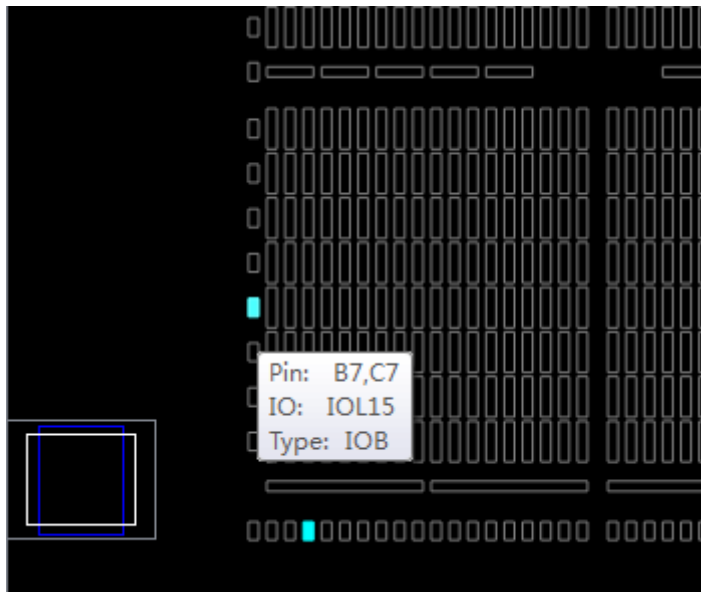


图 3-44 宏单元模式约束

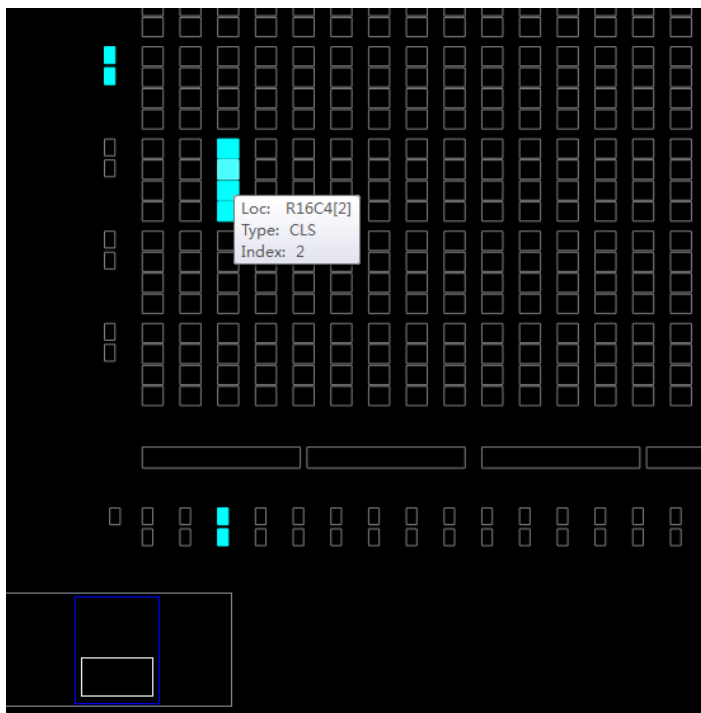
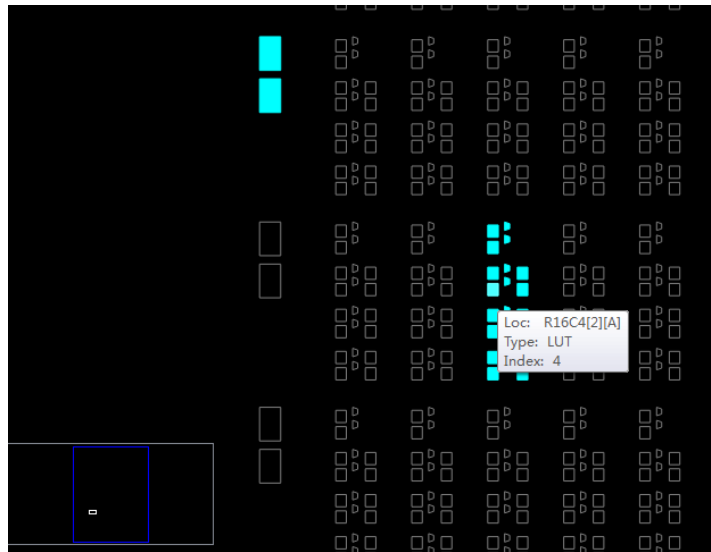


图 3-45 原语模式约束



Chip Array 支持以下拖拽功能：

- 从 Array 内部拖拽：适用于由特定 Primitive 占用的约束位置，可在 Array 中随意拖动该位置；
- 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；
- 黄色：用于显示 IO 和 Primitive 约束在 fix 位置时占用的约束位置；
- 浅蓝色：用于显示 IO 和 Primitive 约束在 range 范围时用到的位置。

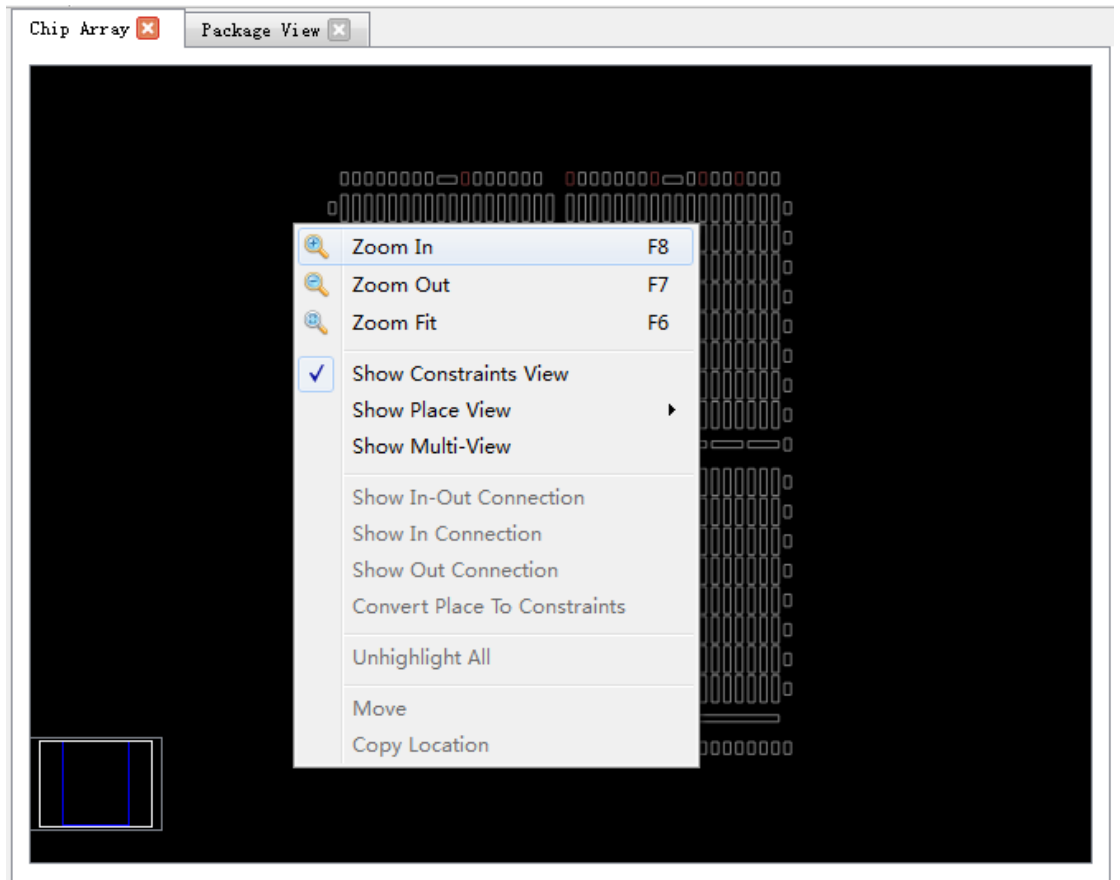
Chip Array 窗口支持右键菜单，具有如下功能：

- 用于放大、缩小视窗；
- 显示约束、放置以及二者复合的视图；
- 显示输入、输出的连接关系；
- 将 Place 的位置信息转化为约束信息；
- 消除高亮显示；
- 删除以及复制位置信息。

注！

- 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，该功能不可用，如图 3-46 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。

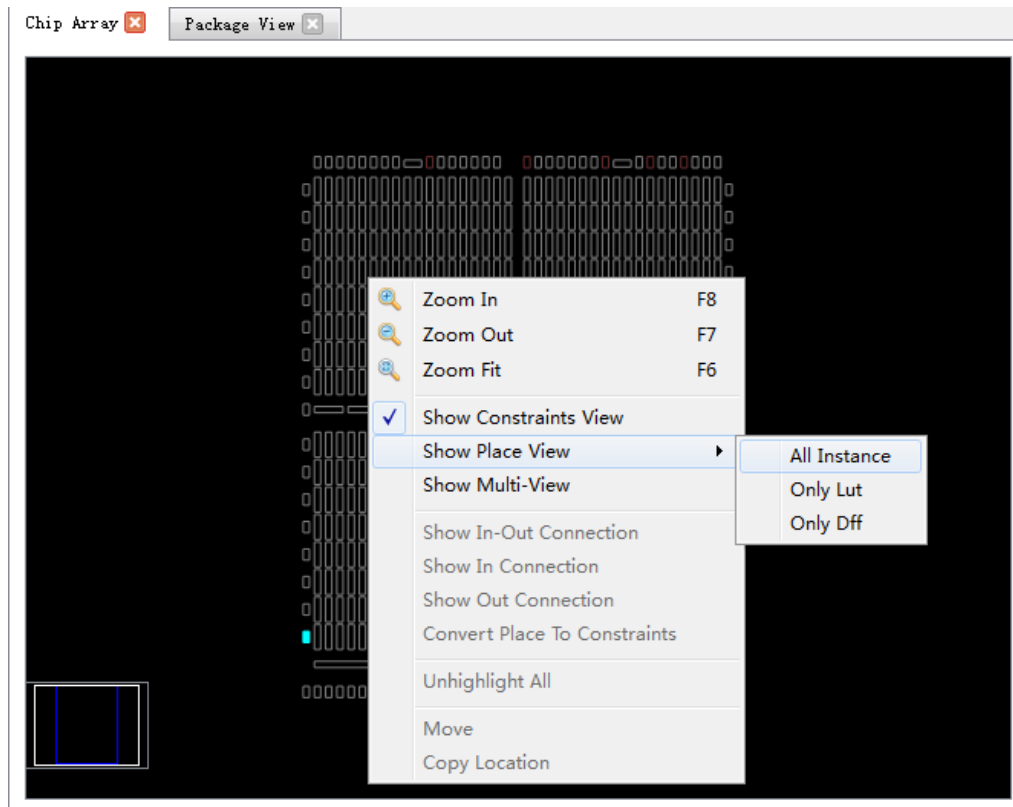
图 3-46 Chip Array 右键功能



在 Show Place View 中，可对 Lut、Reg 的密度进行显示，如图 3-47 所示，详情如下：

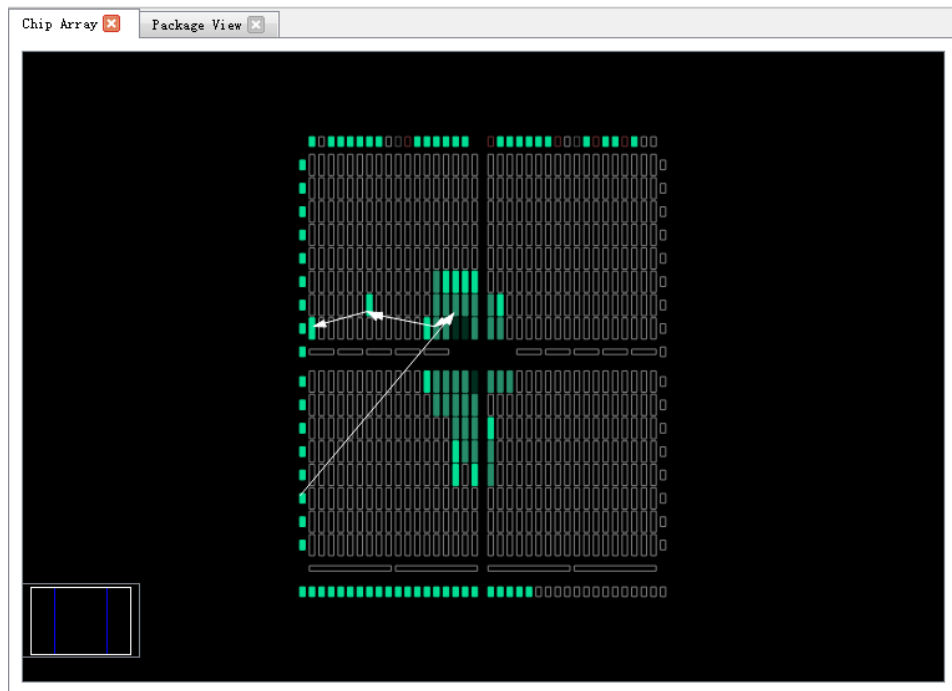
- ALL Instance，显示所有 Instance 的 place 情况，5 个以内呈淡绿色、6-10 个呈绿色、10 个以上呈深绿色；
- Only Lut，只显示所有 Lut 的 place 情况，2 个以内呈淡绿色、3-4 个呈绿色、4 个以上呈深绿色；
- Only Dff，只显示所有 Reg 的 place 情况，2 个以内呈淡绿色、3-4 个呈绿色、4 个以上呈深绿色。

图 3-47 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 3-49 所示。

图 3-48 时序路径高亮显示



3.4.5 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 约束窗口如图 3-49 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、属性等的编辑功能；
- 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；

设置完成后，在 Chip Array 窗口中约束的位置变为黄色或浅蓝色高亮，在 Package View 窗口中约束的位置变为橙色高亮。

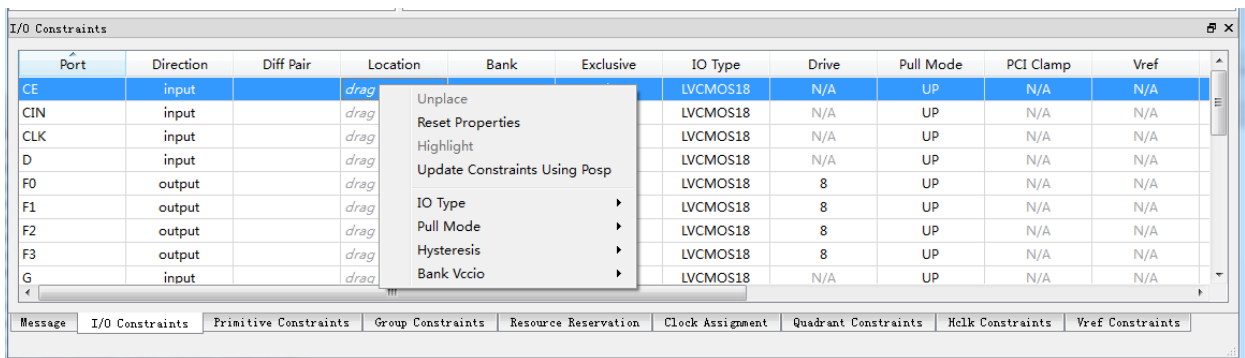
窗口提供右键菜单功能，详情如下：

- 提供取消放置；
- 复位 Port 属性；
- 高亮显示约束位置；
- 按照器件布局信息文件更新约束；
- 设置 I/O 类型、翻转率、上拉模式、驱动方式、BANK 电压等；
- 同时，右键菜单支持用户批量修改 Port 属性的功能。

注！

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。

图 3-49 I/O 约束窗口



Primitive Constraints

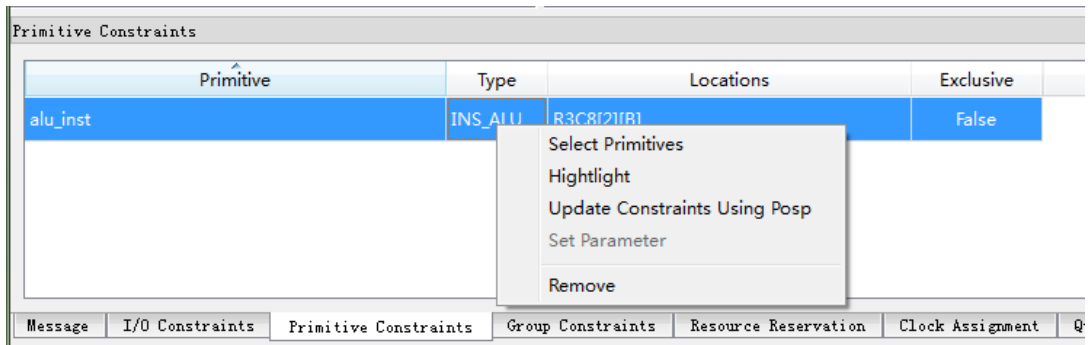
原语约束窗口如图 3-50 所示，功能如下：

- 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；
- 可通过双击选择 **Exclusive** 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、更新约束、置 **parameter** 值的功能。
- 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 3-50 原语约束窗口

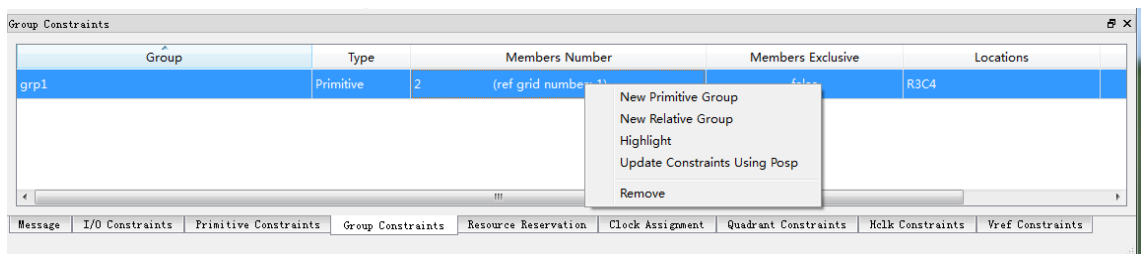


Group Constraints

组约束窗口如图 3-51 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、更新约束的功能。

图 3-51 组约束窗口



Resource Reservation

预留约束窗口如图 3-52 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

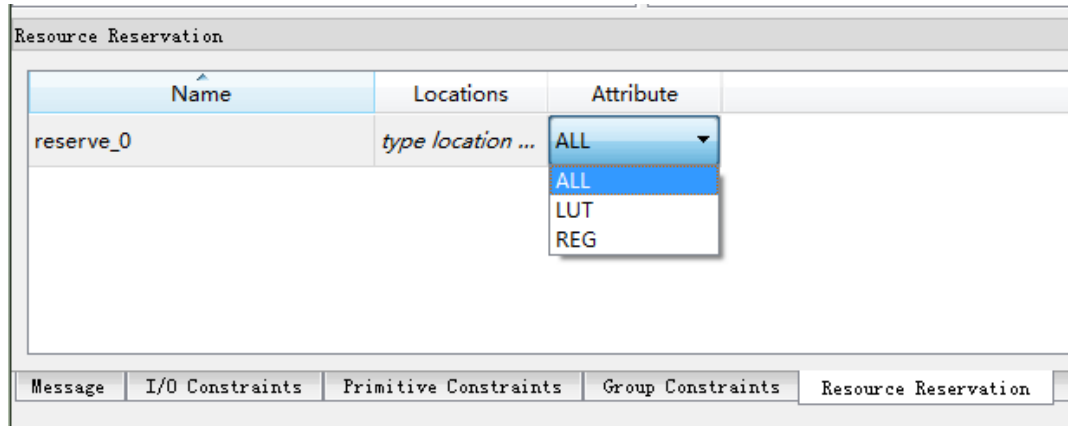
的功能：

- **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 3-52 预留约束窗口



Clock Assignment

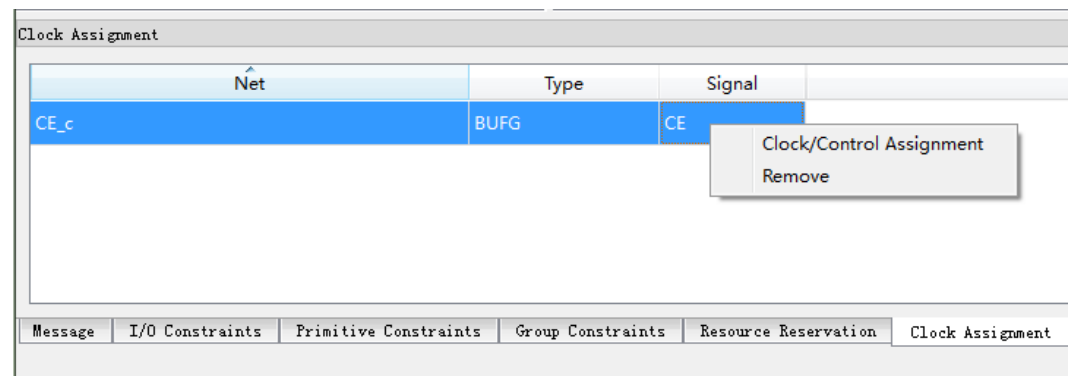
时钟约束窗口如图 3-53 所示，功能如下：

- 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！

- 双击可进行编辑；
- 如 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 3-53 时钟约束窗口



Quadrant Constraints

象限约束窗口如图 3-54 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 两种器件有效。
- 新建象限约束的窗口如图 3-22 和图 3-23 所示。

图 3-54 象限约束窗口



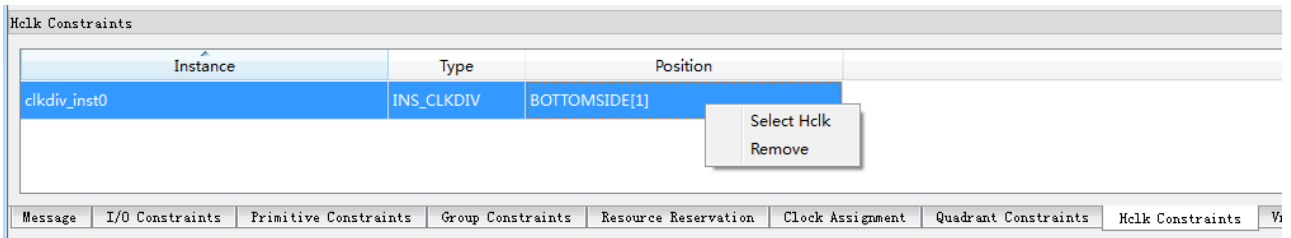
Hclk Constraints

Hclk 约束窗口如图 3-55 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 3-55 Hclk 约束窗口



Vref Constraints

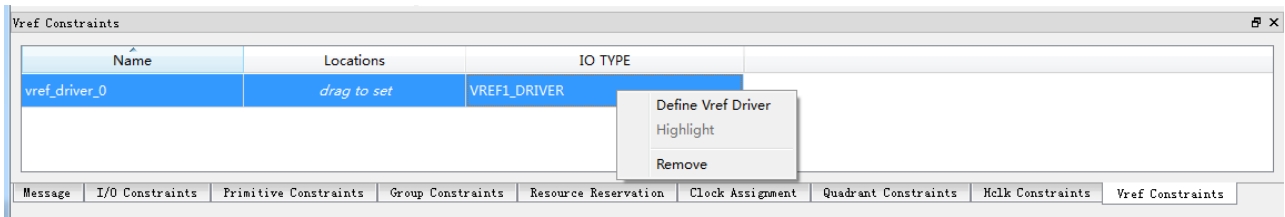
Vref 约束窗口如图 3-56 所示，功能如下：

- 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，用于添加、删除约束信息。

注！

位置信息通过拖拽的方式进行设置。

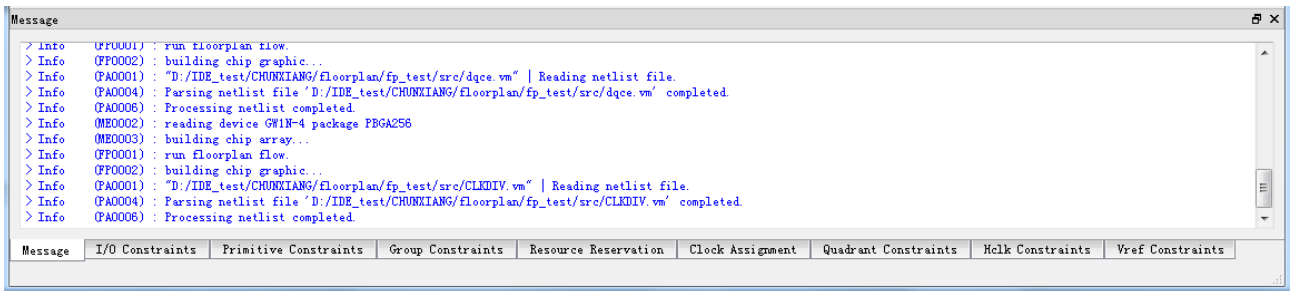
图 3-56 Vref 约束窗口



3.5Message 窗口

Message 窗口如图 3-57 所示，窗口提供输出结果的显示。

图 3-57 Message 窗口



3.6创建 Constraints - 拖拽方式

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 3.4.1 菜单栏。

注！

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

3.6.1设置 I/O Constraints 约束位置

可通过以下方式对 I/O Constraints 约束位置进行设置：

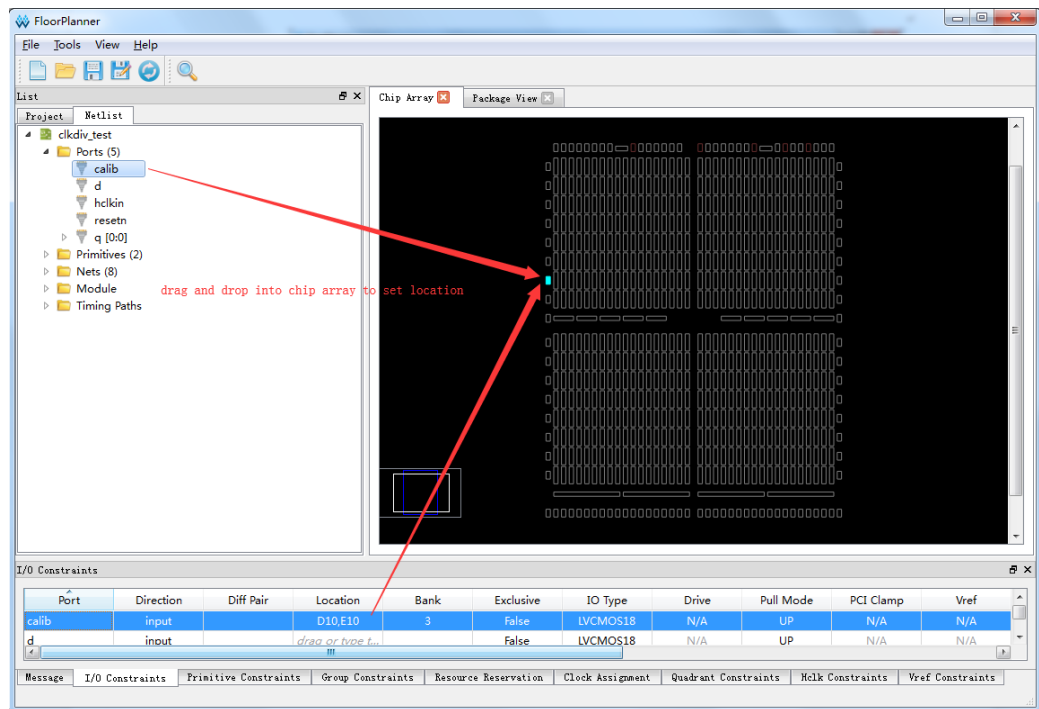
启动 Floor Planner 读取 design 文件后，Netlist 中所有 Ports 对应的 Constraints 已经自动加载到 I/O Constraints 编辑器中。

可通过以下两种方式进行位置拖放：

在 Netlist 窗口（位于左方）中，选择 Port，将其拖至 Chip Array 中；

在 I/O Constraints 编辑器（位于下方）中，选择 Constraints，将其拖至 Chip Array 中，如提示该位置允许放置，则可放置。完成后，Constraints 的 Location 信息变为拖放到的 IOB 的位置，如图 3-58 所示。

图 3-58 拖拽到 Chip Array 设置 I/O Constraints



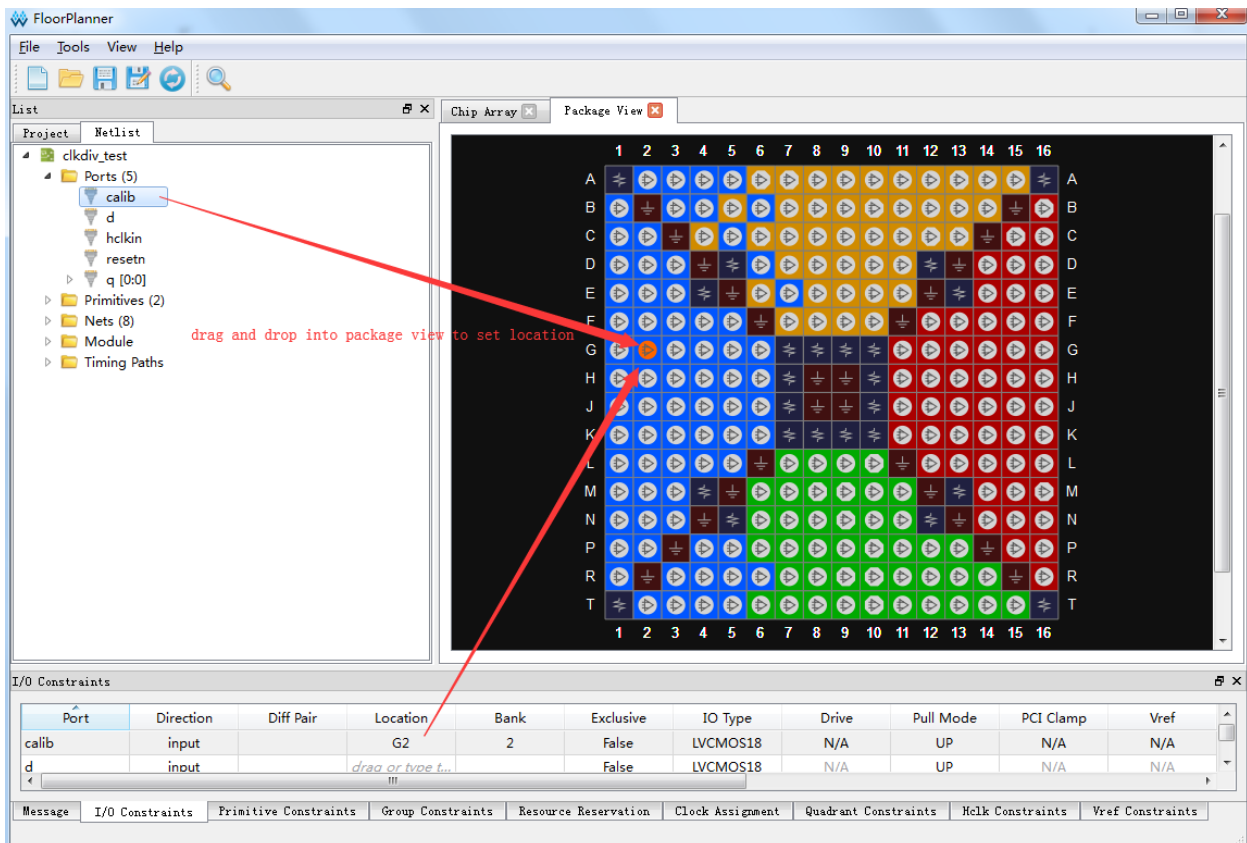
拖至 Package View 设置 I/O Constraints。步骤如下：

1. 在 Netlist 中，将 Port 拖至 Package View 中，置于可放置位置处；
2. Constraints 的 Location 变为拖放到 Package View 中 Pin 的位置，如图 3-59 所示。

注！

亦可通过将 I/O Constraints 编辑器中的 Constraints 拖放至 Package View 中对其进行设置。

图 3-59 拖至 Package View 设置 I/O Constraints



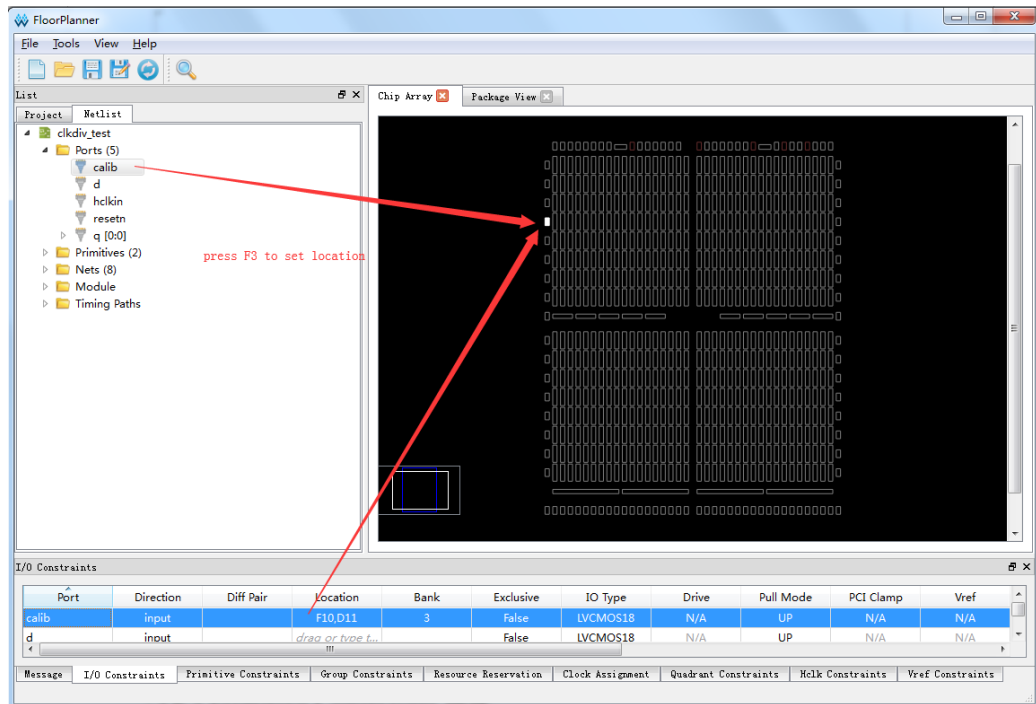
如图 3-60 所示, Constraints 编辑器中可设置对应 Port 的约束 Location。步骤如下:

1. 在 Netlist 中选择 Port;
2. 在 Chip Array 中选择 IOB;
3. 按“F3”键。

注!

亦可通过将 Chip Array 放大后选择 IOBLOCK。

图 3-60 F3 一键设置 I/O Constraints 位置约束



除了以上三种方式外，还可通过直接双击所要设置的 I/O Constraints 的 Location，输入需约束位置。

3.6.2 Primitive Constraints 创建

可通过以下两种方法创建 Primitive Constraints。步骤如下：

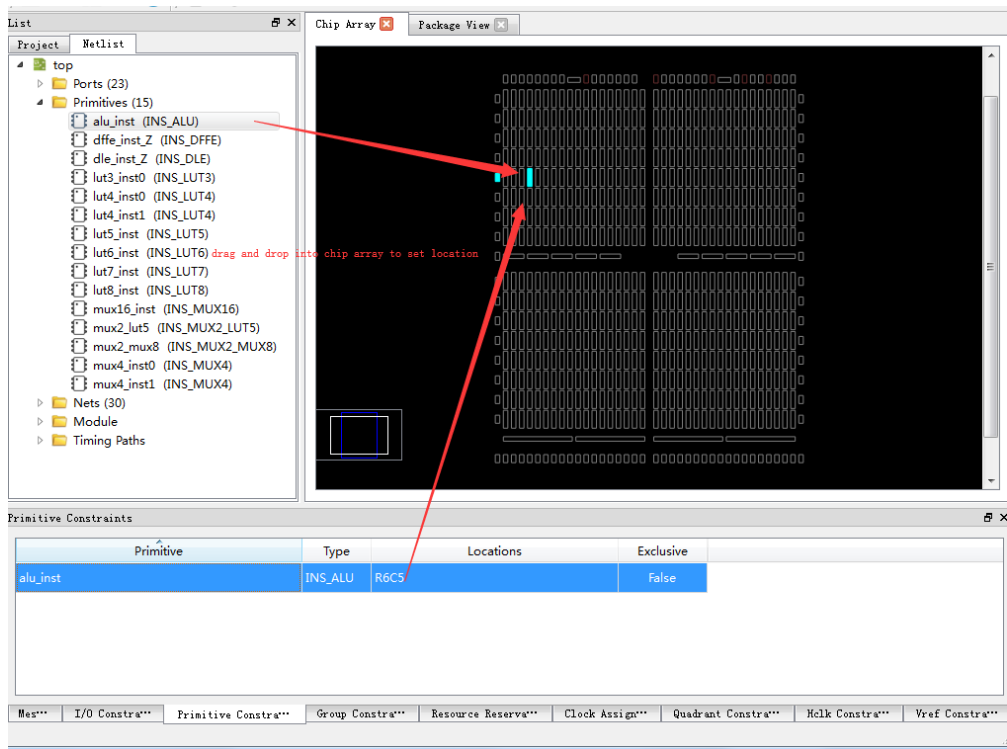
拖拽至 Chip Array 设置 Primitive Constraints

1. 在 Netlist 中，双击或者右键单击菜单，选择 “Edit Constraint”，生成 Primitive Constraints，对应的 Primitive 的约束加入到 Primitive Constraints 编辑器中；
2. 产生 Constraints 后，设置约束位置，在 Primitive Constraints 编辑器中，选择对应的 Constraints，将其拖至 Chip Array 中如图 3-61 所示。

注！

- 亦可在 Primitive Constraints 编辑器中，右键单击菜单，选择 “Select Primitives”，弹出 Primitive Finder 对话框后，选择 Primitive，点击 “OK”，将对应的 Primitive 的约束加入到 Primitive Constraints 编辑器中；
- 或者选中 Netlist 中对应的 Primitive，将其拖至 Chip Array 中，设置对应 Constraints 的位置。

图 3-61 拖拽到 Chip Array 设置 Primitive Constraints



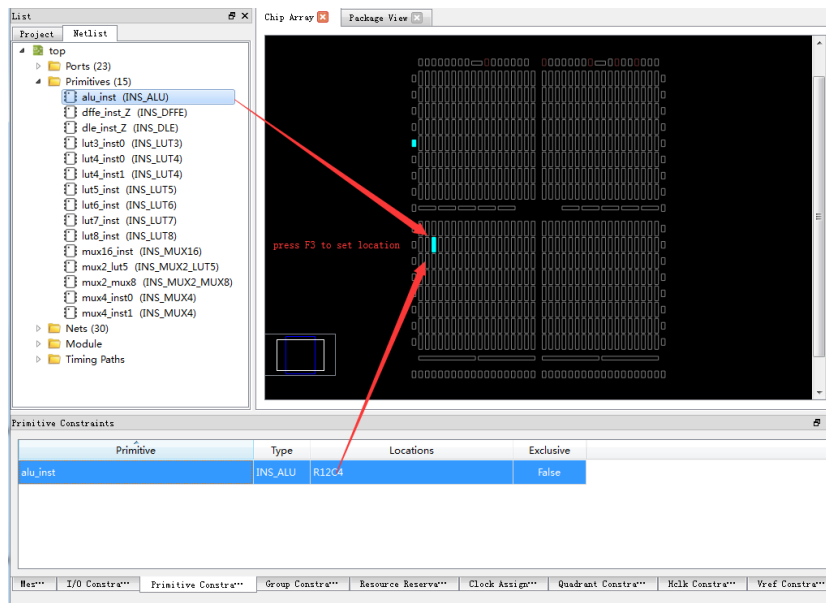
F3 一键设置 Primitive Constraints 的位置

1. 在 Netlist 列表中，选择 Primitive;
2. 在 Chip Array 中，选择 grid，按“F3”键，设置对应 Primitive Constraints 的位置，如图 3-62 所示：

注！

如对应 Primitive 的 Constraints 不存在，则 Floor Planner 会创建该 Primitive 的 Constraints，并约束到选取的 Grid 的位置。

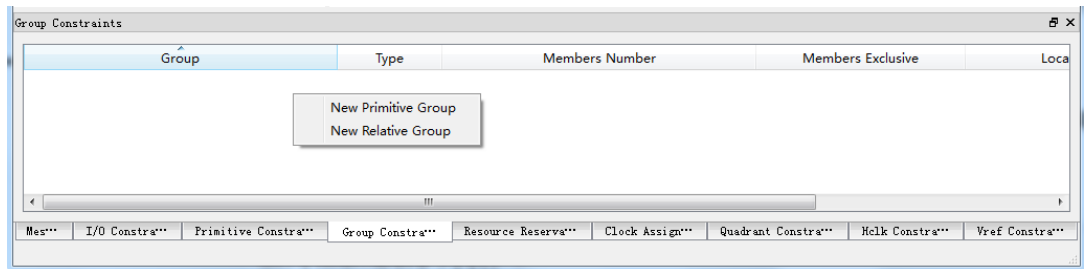
图 3-62 F3 一键设置 Primitive Constraints 的位置



3.6.3 Group Constraints 的创建

如图 3-63 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。

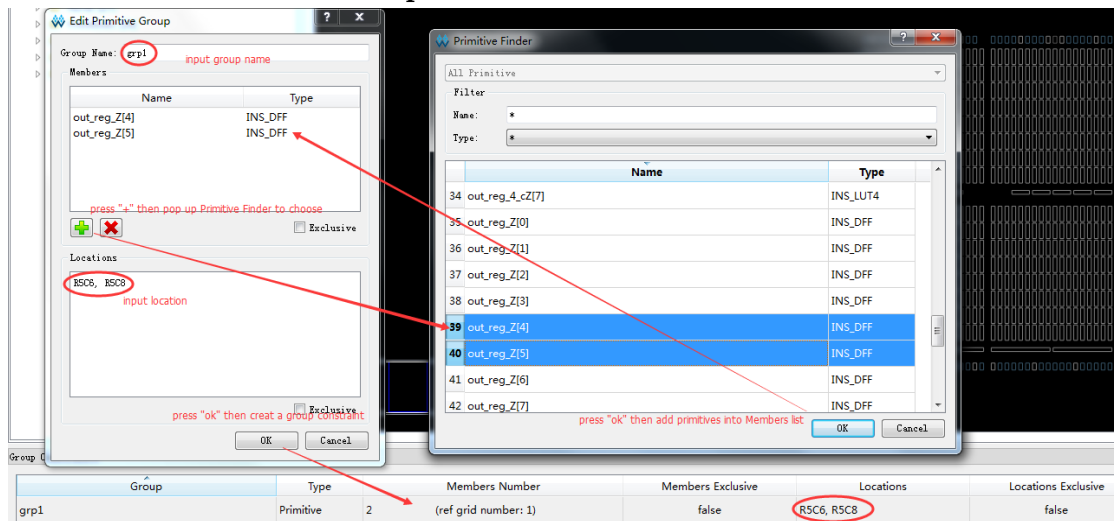
图 3-63 Group Constraints 编辑器右键菜单



创建 Primitive Group Constraints

1. 右键单击菜单，点击“New Primitive Group”，弹出 Edit Primitive Group 对话框；
2. 输入 Group Name，点击“+”弹出 Primitive Finder 对话框；
3. 选取所要设置的 Primitive，点击 Primitive Finder 对话框，选择“OK”，加入 Members 列表；
4. 在 Locations 输入所要约束的位置；
5. 选择 Edit Primitive Group 的“OK”，在 Group Constraints 中加入一条 Group Constraints，如图 3-64 所示；
6. 在 Netlist 中，右键“Timing Paths > Path”；
7. 弹出菜单，点击“Edit Constraints”，即可将该 Path 对应的 Group Constraint 加入到 Group Constraints 编辑器中。

图 3-64 创建 Primitive Group Constraints

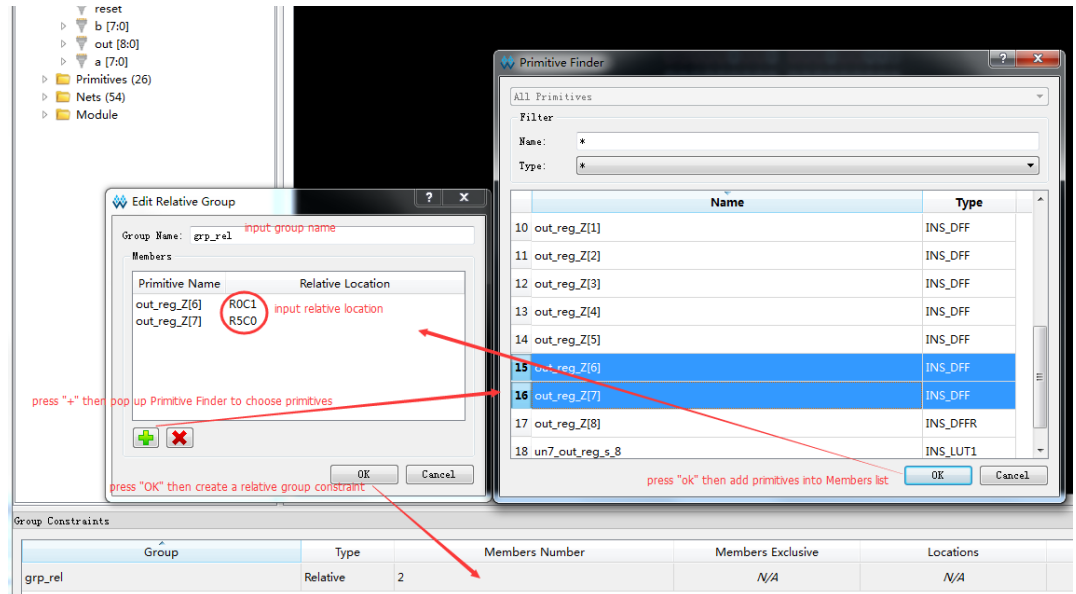


创建 Relative Group Constraints

右键单击菜单，点击“New Relative Group”，弹出 Edit Relative Group 对话框；

- 输入 Group 的名字，点击“+”，弹出 Primitive Finder 对话框；
- 选择所要设置的 Primitives；
- 选择“OK”，添至 Member 列表中；
- 为每个 Primitive 添加相对位置；
- 在 Edit Primitive Group 中，选择“OK”，如图 3-65 所示。

图 3-65 创建 Relative Group Constraints

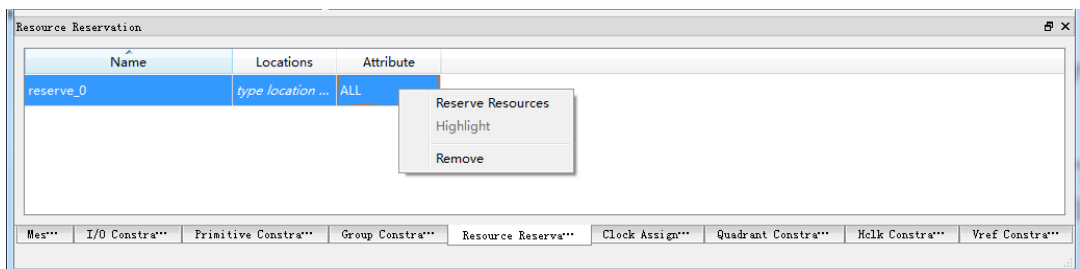


3.6.4 Resource Reservation 的创建

创建 Resource Reservation

1. 在 Resource Reservation 编辑器中，单击鼠标右键，弹出菜单；
2. 点击“Reserve Resources”，在编辑器中添加 Resource Reservation 约束，如图 3-66 所示。

图 3-66 创建 Resource Reservation 约束



设置 Resource Reservation 位置

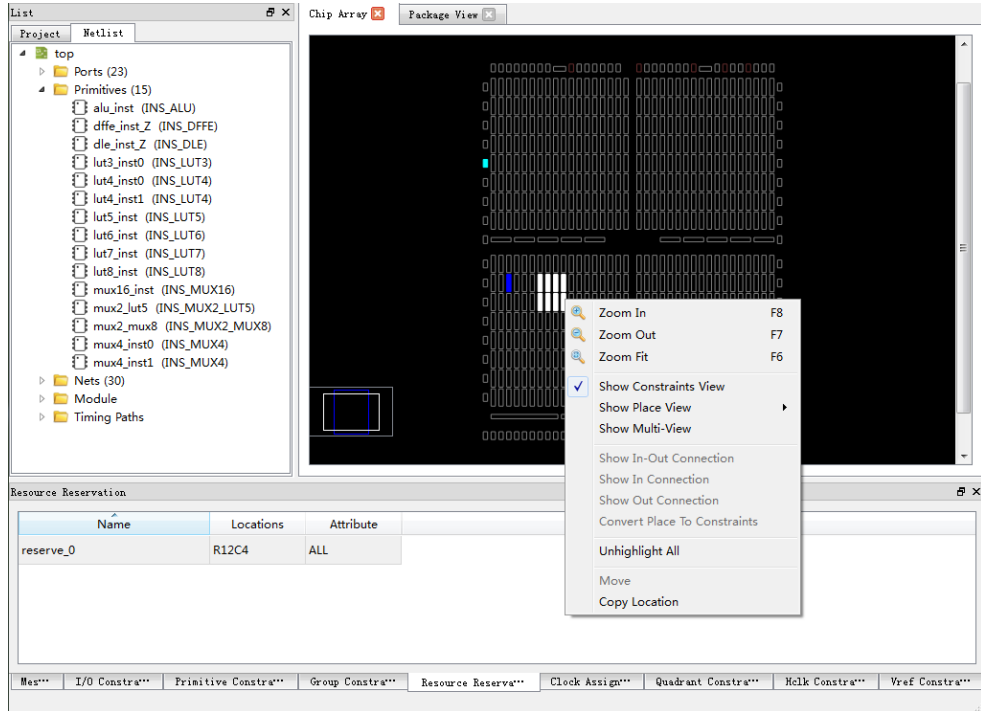
1. 从 Resource Reservation 中选中一条 Constraint，将其拖至 Chip Array 中的目标位置处；
2. 对该 Constraints 位置进行修改，如图 3-67 所示。
在 Chip Array 中，按“Ctrl”键同时，选取区域，右键单击“Copy Location”，将该位置复制到需修改的“Resource Reservation > location”处，设置

Resource Reservation 约束的位置。

注！

- 亦可通过以下方式修改位置：
- 亦可双击该条 Constraints 的 Location，手动输入想要约束的位置。

图 3-67 复制 Chip Array 中的位置设置 Resource Reservation 约束位

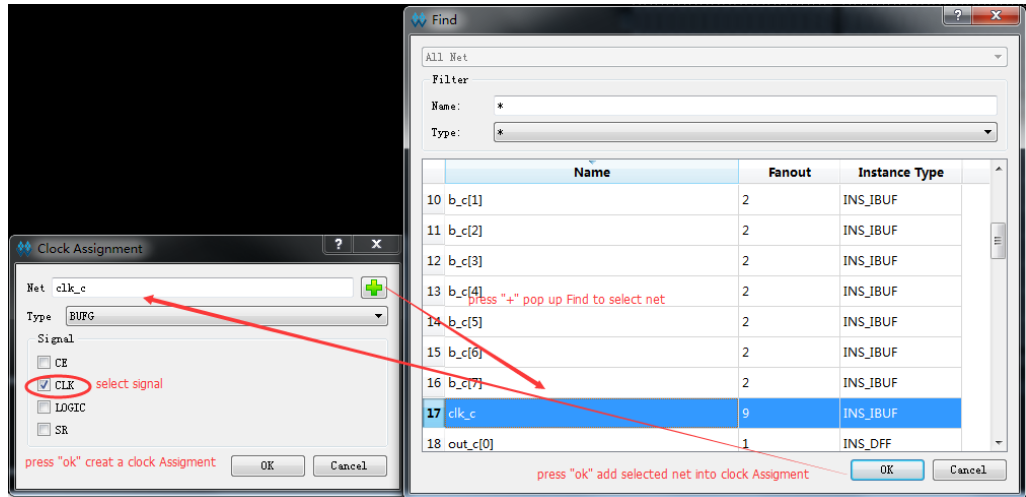


3.6.5 Clock Assignment 的创建

创建 Clock Assignment 约束

1. 在 Clock Assignment 编辑器中，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 Find 对话框，选择 Net，在 Find 对话框中，单击“OK”；
3. 设置 Net，在下拉列表中选择 Type 类型，设置 Signal，设置完成后，单击“OK”，即会将该约束添加至 Clock Assignment 编辑器中，如图 3-68 所示。

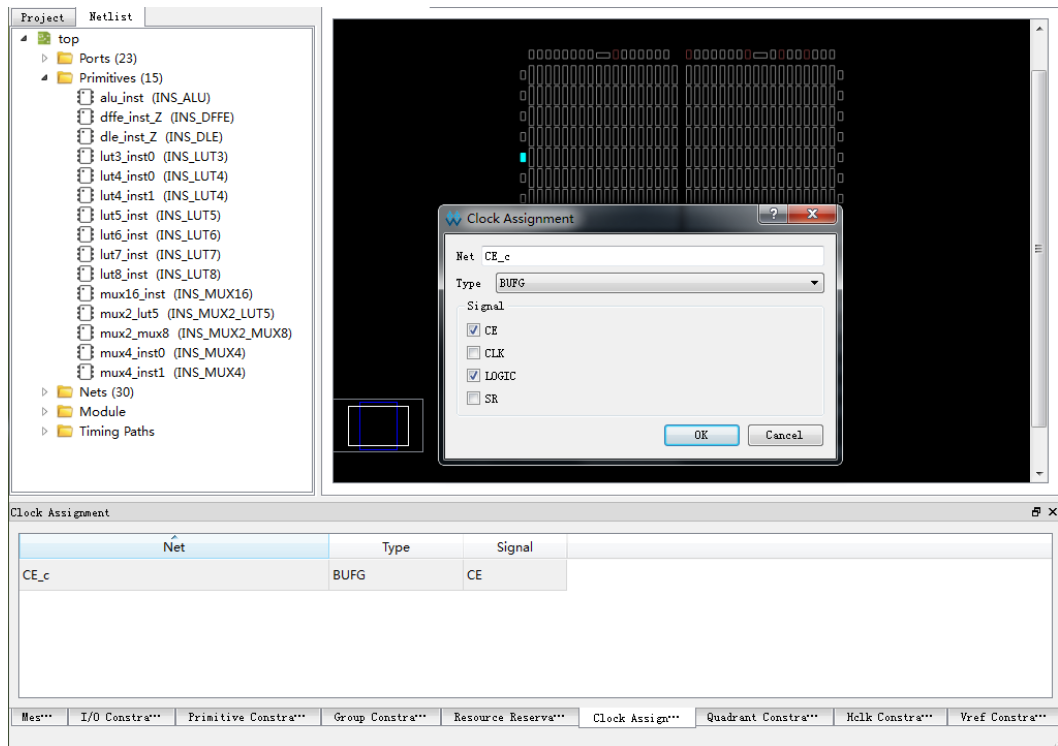
图 3-68 创建 Clock Assignment 约束



修改 Clock Assignment 约束

1. 在 Clock Assignment 约束编辑器中，双击所要修改的约束，弹出 Clock Assignment 对话框；
2. 除 Net 不可更改外，可修改 Type 以及 Signal 属性，如图 3-69 所示。

图 3-69 修改 Clock Assignment 约束



3.6.6 Quadrant Constraints 的创建

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

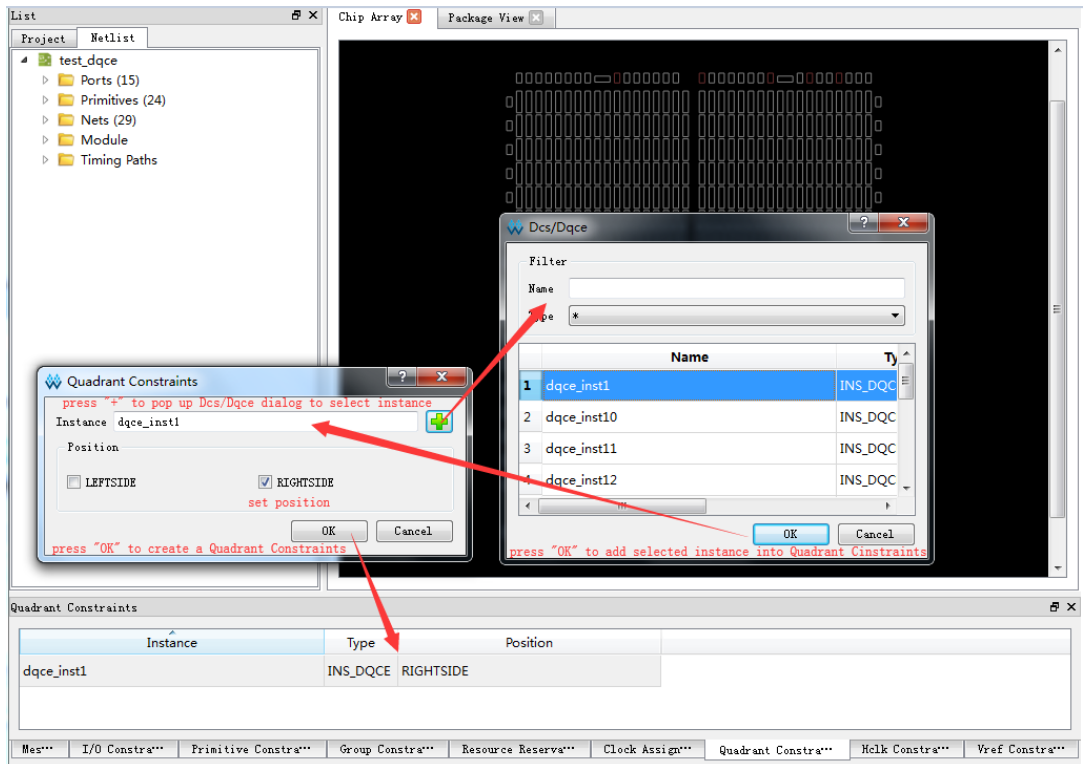
- Dcs

● Dqce

创建 Quadrant Constraints 约束

1. 在 Quadrant Constraints 编辑器中，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 完成设置后，在 Position 下方选取所要约束的位置，单击 “OK”，即会将该条约束添至 Quadrant Constraints 编辑窗口，如图 3-70 所示。

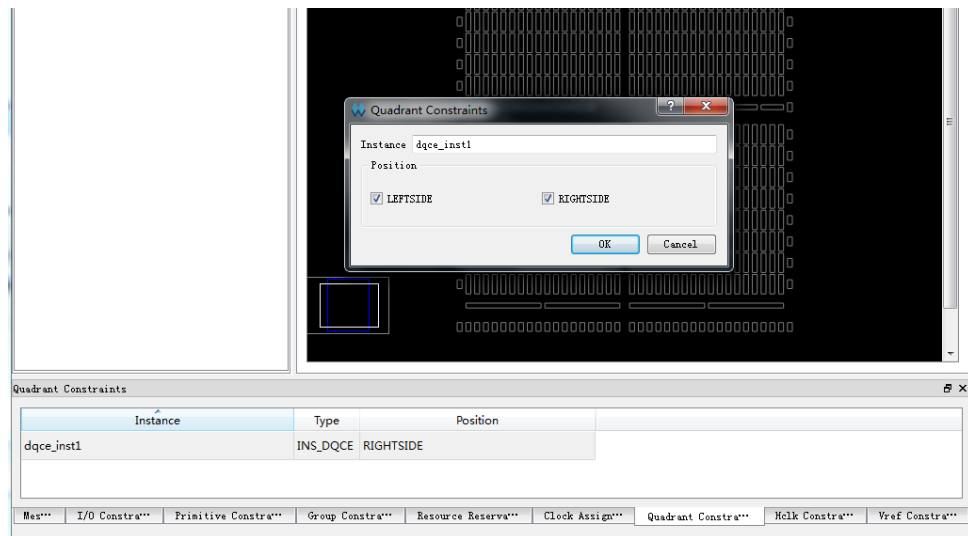
图 3-70 创建 Quadrant Constraints



修改 Quadrant Constraints 约束

在 Quadrant Constraints 编辑窗口中双击所要修改的约束，会弹出 Quadrant Constraints 对话框，Instance 不可更改，只可修改 Position 属性，如图 3-71 所示。

图 3-71 修改 Quadrant Constraints



3.6.7 Hclk Constraints 的创建

创建 Hclk Constraints 约束

Hclk Constraints 的创建步骤如下：

在 Hclk Constraints 编辑器中，右键单击菜单，选择“Select Hclk”，弹出 Hclk Constraints 设置对话框；

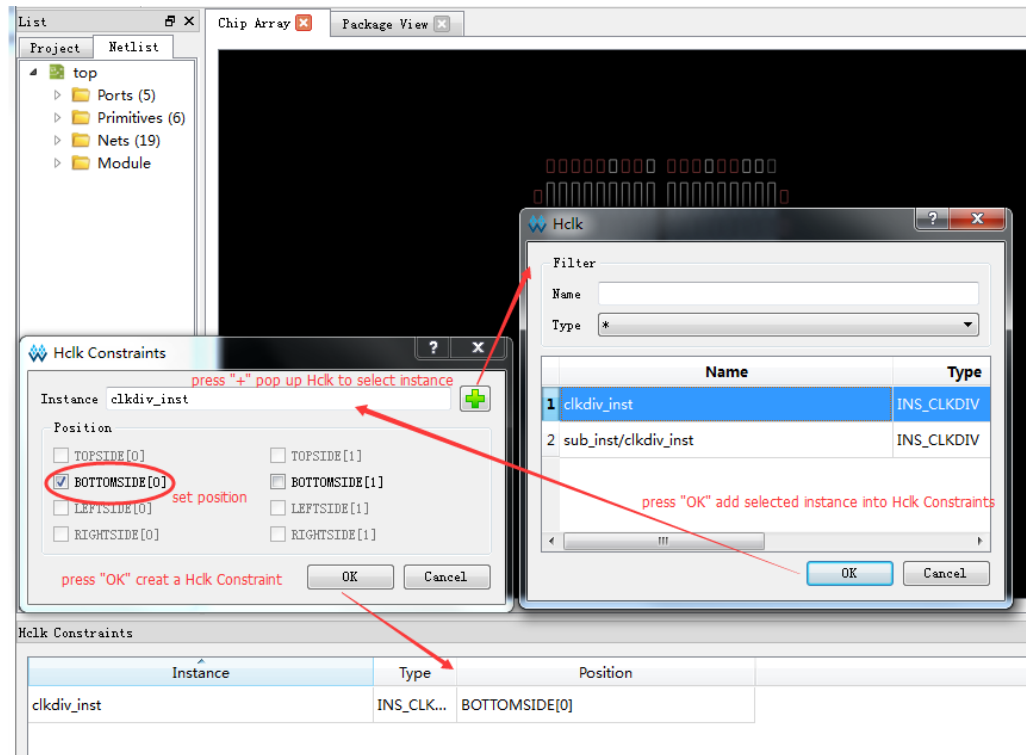
单击“+”按钮，弹出 Hclk 对话框；

选取 Instance，在 Hclk 对话框中，单击“OK”，设置 Instance，在 Position 下方选取所要约束的位置，单击“OK”，即可将该约束添至 Hclk Constraints 编辑器中，如图 3-72 所示。

注！

Hclk Constraints 只对 Clkdiv 和 Dldly 两种类型的 Instance 进行约束。

图 3-72 创建 Hclk Constraints 约束



修改 Hclk Constraints 约束

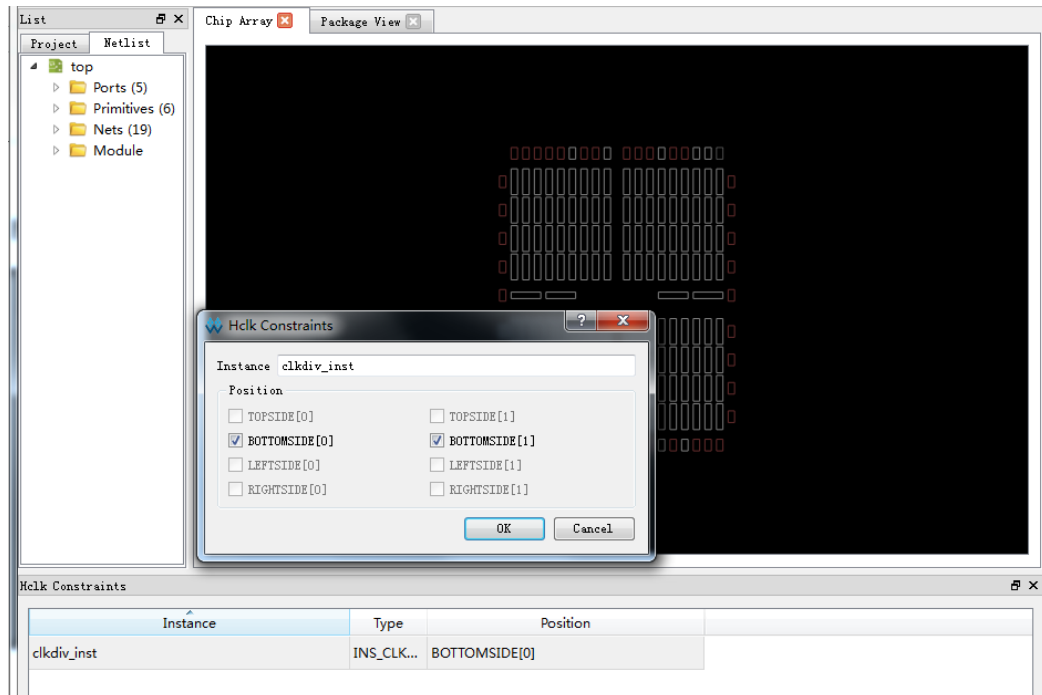
修改 Hclk Constraints 约束步骤如下:

在 Hclk Constraints 约束编辑器中, 双击需修改的约束, 弹出 Hclk Constraints 对话框, 如图 3-73 所示。

注!

Instance 不可更改, 只可修改 Position 属性。

图 3-73 修改 Hclk Constraints 约束

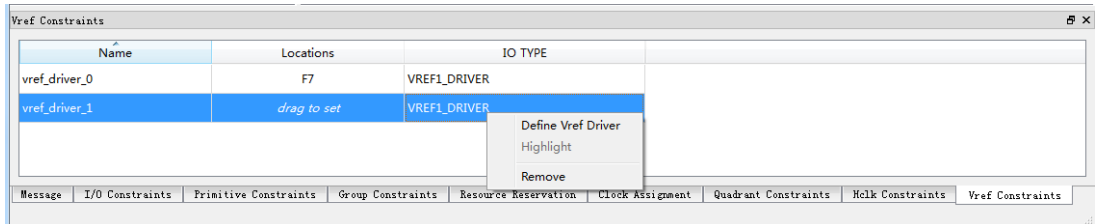


3.6.8 Vref Constraints 的创建

创建 Vref Constraints

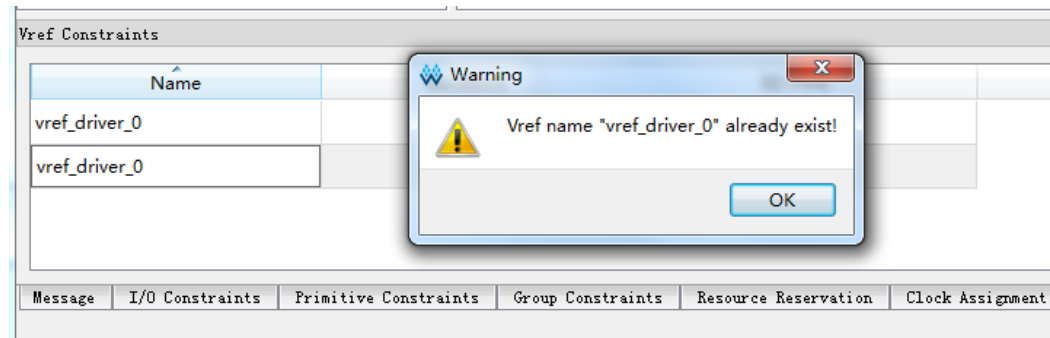
在 Vref Constraints 编辑器中，右键单击菜单，选择“Define Vref Driver”，即可将该条 Vref Constraints 约束添至 Vref Constraints 编辑器中，如图 3-74 所示。

图 3-74 创建 Vref Constraints



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 3-75 所示。

图 3-75 Vref 重命名检查



设置 Vref Constraints 约束位置

在 Vref Constraints 编辑器中选取一条约束，将其拖至 Chip Array 中，如提示可放置，则将其置于可放置的 grid 处，如图 3-76 所示。

注！

- 亦可通过以下方式设置 Vref Constraints 约束的位置：
- 在 Vref Constraints 编辑器中，选取一条约束，将其拖至 Package View 中，如提示可放置，则将其置于可放置处，如图 3-77 所示。

图 3-76 拖动到 Chip Array 中修改 Location

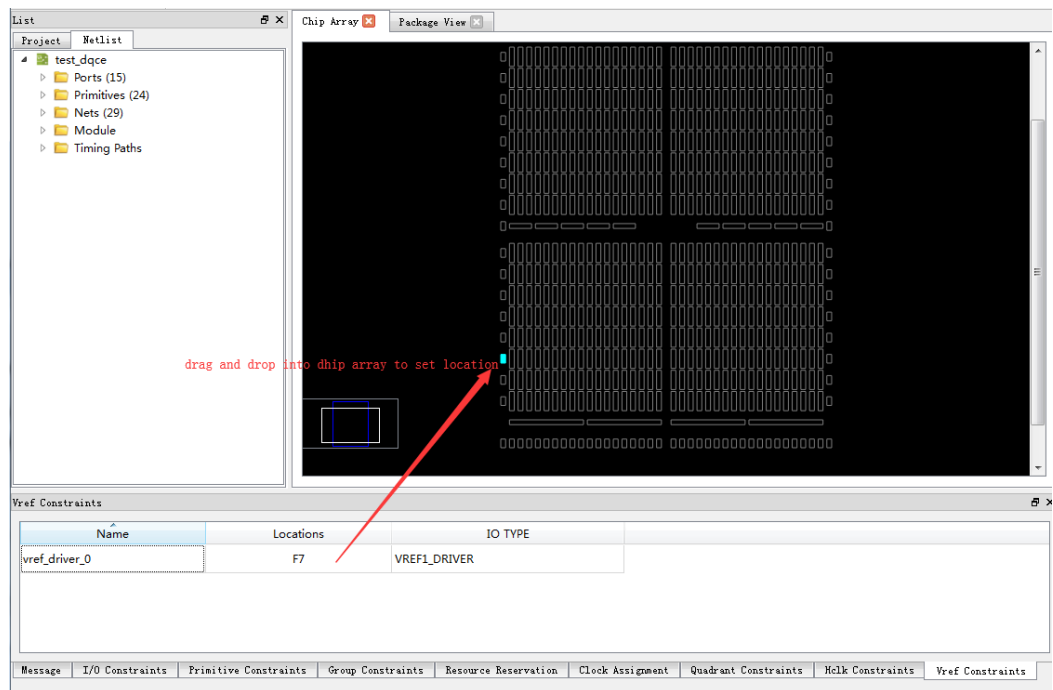
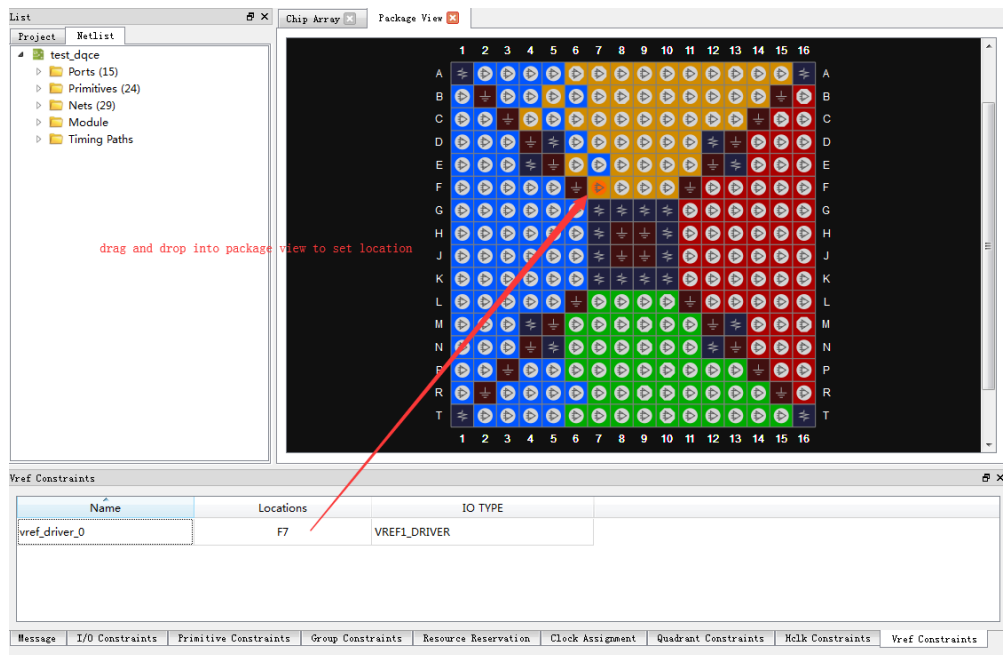


图 3-77 拖动到 Package View 中设置 Vref 约束 Location



4 时序约束

4.1 静态时序分析(STA)概述

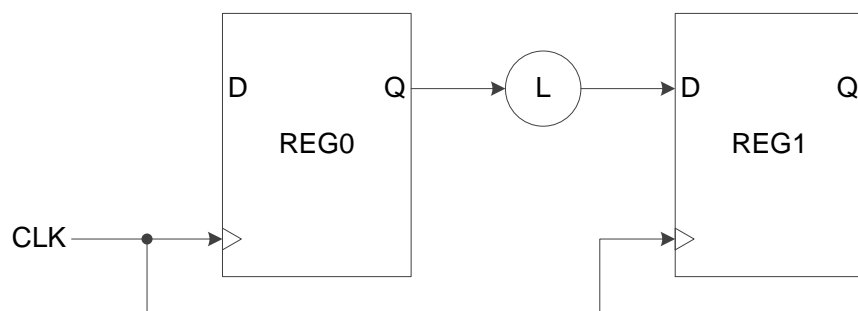
静态时序分析 (STA) 对电路网表进行全面的分析, 计算电路中时序元件的建立和保持时间, 并判断其是否满足要求。设计者仅需提供约束激励, 计算分析过程由软件自动完成, 与传统的分析方法相比, 其验证时间以及覆盖率均占有极大的优势。静态时序分析 (STA) 从用户约束入手, 对特定的时序模型进行分析。高云半导体软件通过分析数据需求时间 (data required time)、数据到达时间 (data arrival time) 和时钟到达时间 (clock arrival time) 来验证电路的性能, 发现可能的时序违规现象, 生成时序分析报告反馈给用户。用户可根据是否满足自身需求, 进一步调整电路设计, 从而更好地提高系统工作速率和稳定性。

在进行静态时序分析 (STA) 前, 须对其基本概念进行了解。以下对分析过程中涉及的基本模型、术语和概念进行介绍。

4.1.1 时序分析基本模型

静态时序分析 (STA) 是对从时序元件发起到时序元件结束的模型进行时序分析, 基本模型如图 4-1 所示, 触发器 REG0 在时钟有效沿将 D 端数据同步到 Q 端, 经过逻辑电路到达触发器 REG1, 触发器 REG1 在时钟有效沿时采集从触发器 REG0 送出的数据, 静态时序分析 (STA) 即是检查触发器 REG1 能否正确采集触发器 REG0 传输的数据。

图 4-1 时序分析基本模型



触发器 REG0 的有效时钟沿称为发起沿 (launch edge)，触发器 REG1 的有效时钟沿称为锁存沿 (latch edge)。如对路径约束的影响不予考虑，两个边沿通常差一个或者半个时钟周期。

4.1.2 时序分析术语

时序模型基本时序单元构成如下：

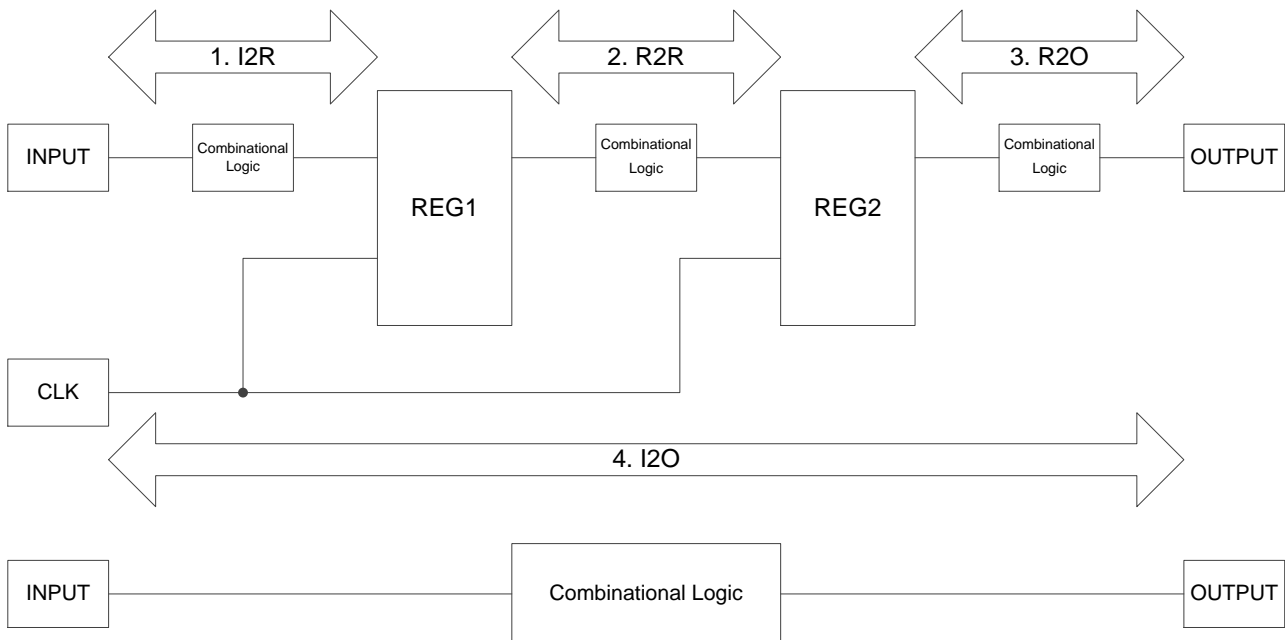
- Cells: 包括 LUT、DFF、MUX、DL、DDR 等基本逻辑单元；
- Pins: Cells 的输入输出端口；
- Ports: 顶层模块的输入输出端口，通常是指器件外部管脚；
- Nets: pin 与 pin 之间的连线；
- Clocks: 时序约束中设定的时钟。

4.1.3 时序分析路径

通常静态时序分析 (STA) 对四种类型的路径进行分析，根据起点和终点的不同对其进行分类。如图 4-2 所示：

- I2R: 输入端口到寄存器；
- R2R: 寄存器到寄存器；
- R2O: 寄存器到输出端口；
- I2O: 输入到输出。

图 4-2 STA 四类时序路径



静态时序分析 (STA) 工具通过该四类路径计算各类路径的数据到达时间 (data arrival time) 和数据要求时间 (data required time)。

数据到达时间 (data arrival time) 是指从时序路径的起点到终点所需的时间；数据要求时间 (data required time) 是指要求数据到达的时间。在计算数

据到达时间 (data arrival time) 时, 时钟路径存在时钟偏斜 (clock skew), 时钟偏斜 (clock skew) 是指时钟到达不同时序元件时钟端口的时间差。

4.1.4 常见的时序检查

建立时间 (setup time) 和保持时间 (hold time) 检查

建立时间: 数据在时钟有效沿前须稳定的最短时间, 如不满足该时间, 则下一级寄存器不能对该数据进行正常采集;

保持时间: 数据在时钟有效沿后须稳定的最短时间, 如不满足该时间, 则数据会被上级寄存器传输的新数据覆盖。

恢复时间 (recovery time) 和移除时间 (removal time) 检查

恢复时间: 在时钟有效沿前, 解除异步置复位的信号须保持稳定的最短时间, 如不满足该时间, 则寄存器可能无法进入正常工作状态;

移除时间: 在时钟有效沿后, 解除异步置复位的信号须保持稳定的最短时间, 如不满足该时间, 则寄存器可能无法进入正常工作状态。

最小时钟脉冲 (MPW) 检查

最小时钟脉冲 (MPW): 芯片内部元器件可识别的高低电平的最小宽度, 低于该宽度, 则时钟不能被正常识别。

静态时序分析 (STA) 通常对上述三类进行检查, 并对布局布线过程提供建议, 旨在更好地满足用户对时序的要求。

4.2 时序约束编辑器

4.2.1 概述

高云半导体静态时序分析 (STA) 工具 (以下简称 STA 工具) 支持多种时序命令, 包括时钟、输入输出、路径等约束和时钟报告等命令, 用户可通过 STA 工具提供的图形界面 (GUI) 添加时序约束。

STA 工具支持默认的时序分析, 默认时钟的上升沿为 0ns, 下降沿为 5ns, 周期为 10ns。

STA 工具默认对所有时钟进行跨时钟域时序分析, 如不选择进行跨时钟域分析, 可通过具体时序约束设置时钟间的关系。

STA 工具提供两种时序报告: 网页 (HTML) 格式和文本格式, 默认为网页 (HTML) 格式。时序报告的内容默认包括建立时间检查、保持时间检查、最大频率计算和最小时钟脉冲检查, 并支持用户根据具体需求提供定制的时序报告。

4.2.2 打开编辑器

启动时序约束编辑器方式:

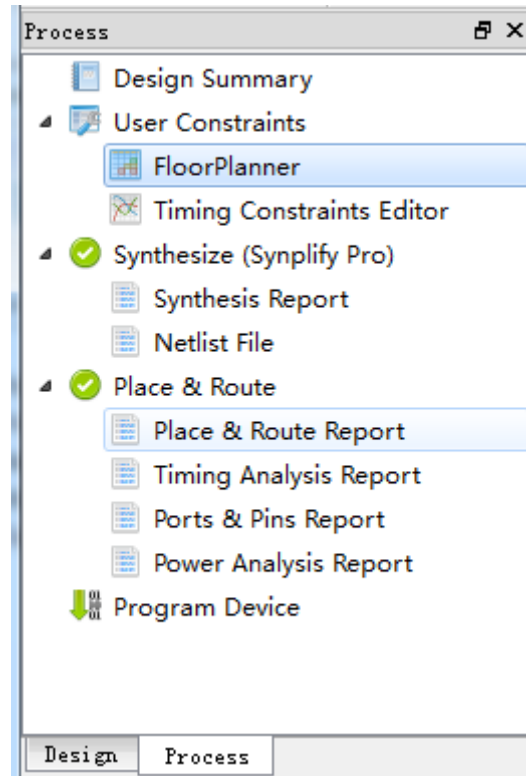
RTL 软件工程建立完成后, 在云源软件菜单栏中, 双击 “Process >

Timing Constraints Editor”，如图 4-3 所示，即可打开时序约束编辑器。

注！

工程中的网表文件会自动加载到时序约束编辑器中。

图 4-3 Process 窗口



4.2.3 新建和打开约束文件

如下所述，通过实例对时序约束编辑器的使用方法进行介绍。

新建约束文件

步骤如下：

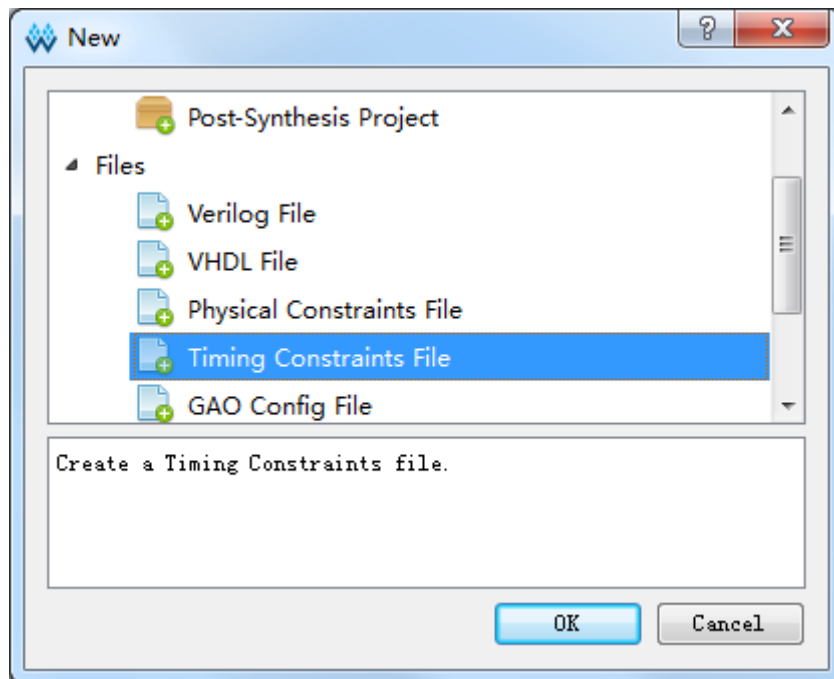
1. 单击“File > New”菜单项，弹出打开文件窗口；
2. 选择“Timing Constraints File”选项，如图 4-4 所示。

注！

亦可通过以下方式打开新建时序约束文件窗口：

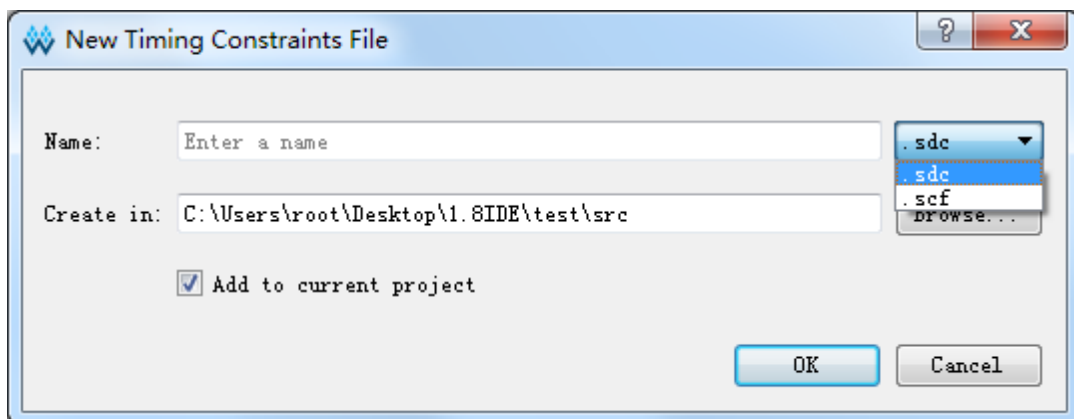
- 单击工具栏上的“New”图标；
- 使用快捷键 Ctrl + N。

图 4-4 打开新建时序约束文件



单击“OK”确认，弹出新建时序约束文件的对话框，如图 4-5 所示。

图 4-5 新建时序约束文件



Name: 新建时序约束文件的名称，后缀支持.sdc 和.scf；

Create in: 通过“Browse”对话框选择新建约束文件的存放位置，默认路径为工程目录下的 src 文件夹下；

Add to current project: 选择该选项后，会自动将约束文件添加到工程中。

打开约束文件

步骤如下：

- 1.单击“File > Open”菜单项；
- 2.打开“Open Timing Constraint”对话框，如图 4-6 所示。

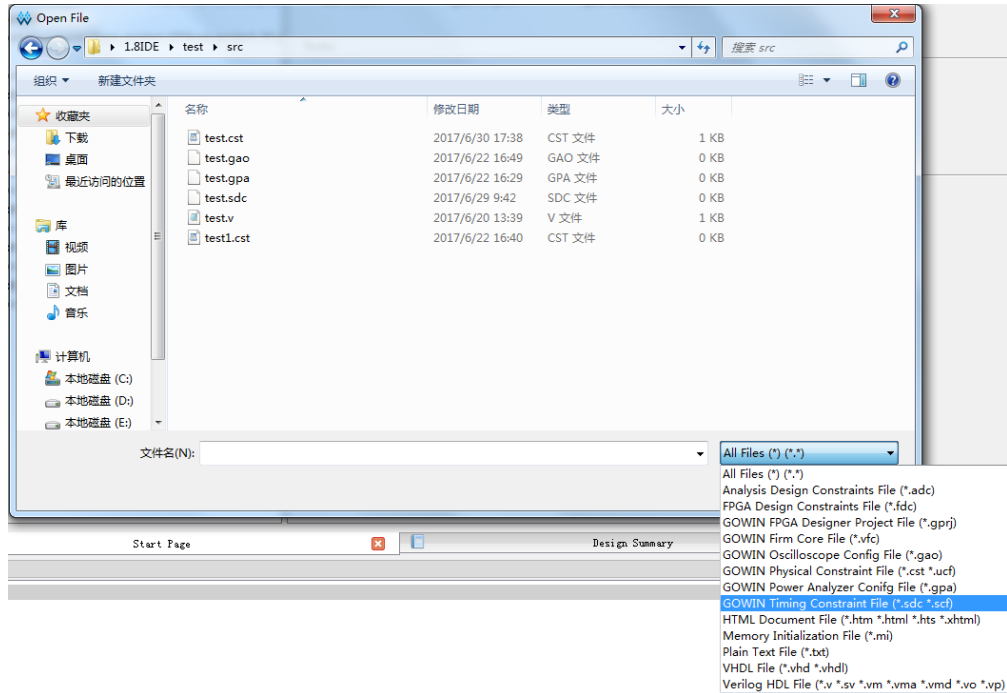
注！

亦可通过以下方式打开时序约束文件窗口：

- 单击工具栏上的“Open”图标；

- 使用快捷键 Ctrl + O。

图 4-6 打开时序约束文件

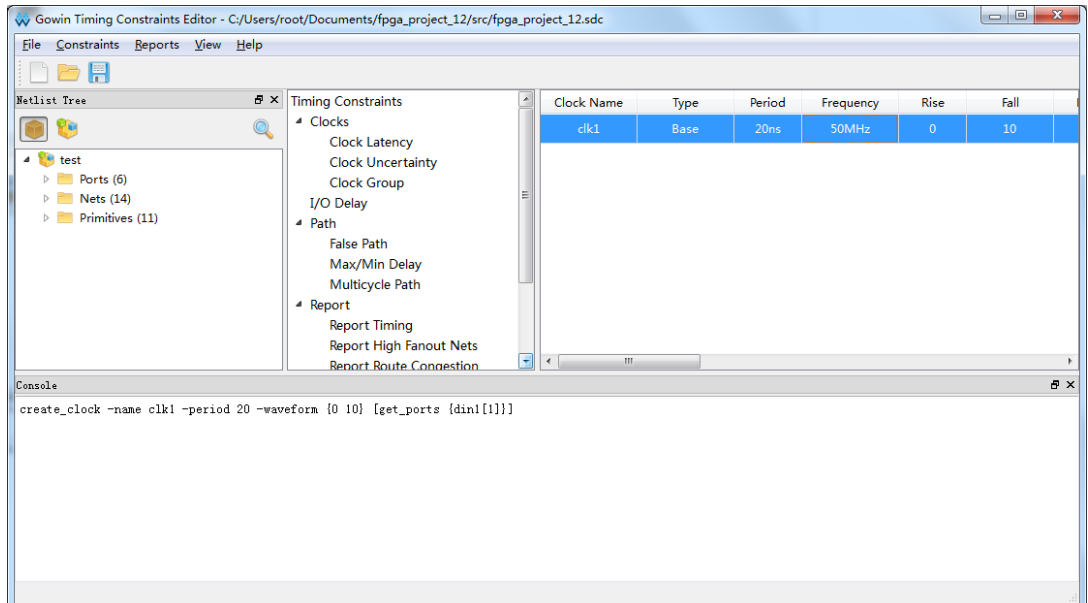


选择时序约束文件所在的目录，选中文件打开。

4.2.4 编辑器界面

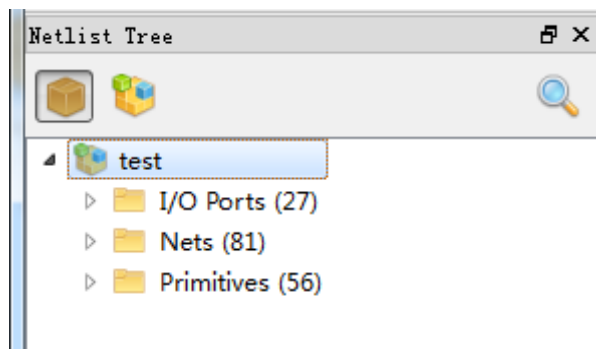
新建或打开约束文件后，编辑器界面如图 4-7 所示。

图 4-7 时序约束编辑器界面



主界面左上角为 Netlist Tree 窗口，如图 4-8 所示。

图 4-8 Netlist Tree 窗口



Netlist Tree 窗口中含列当前网表文件中的各种元素，包括 Top Module、I/O Ports、Nets 和 Primitives。

- “”：查看 flatten 列表；
- “”：查看 hierarchy 列表。

主界面中间及右侧区域即为约束编辑区，如图 4-9 所示。其中，左侧列表为时序约束类型目录，右侧为表格编辑区。在类型目录上单击选中某一约束类型，表格编辑区中会显示已设置的约束编辑列表。

图 4-9 约束编辑界面

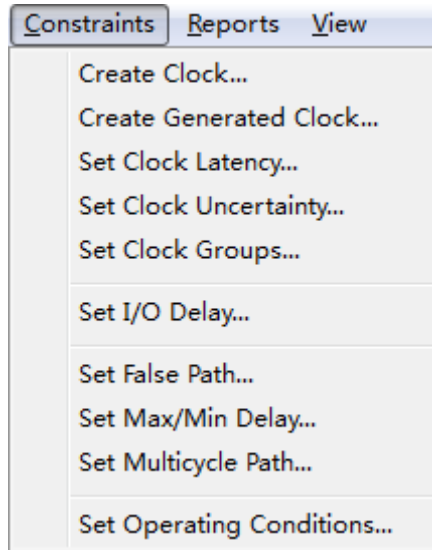
Timing Constraints	Clock Name	Type	Period	Frequency	Rise	Fall	Divide by	Multi
<ul style="list-style-type: none"> ▲ Clocks <ul style="list-style-type: none"> Clock Latency Clock Uncertainty Clock Group I/O Delay ▲ Path <ul style="list-style-type: none"> False Path Max/Min Delay Multicycle Path ▲ Report <ul style="list-style-type: none"> Report Timing Report High Fanout Nets Report Route Congestion Report Min Pulse Width Report Max Frequency Report Exception Set Operating Conditions 	clk1	Base	10ns	100MHz	0	5	N/A	N

4.2.5 时序约束界面

提供两种 GUI 界面时序约束方式。

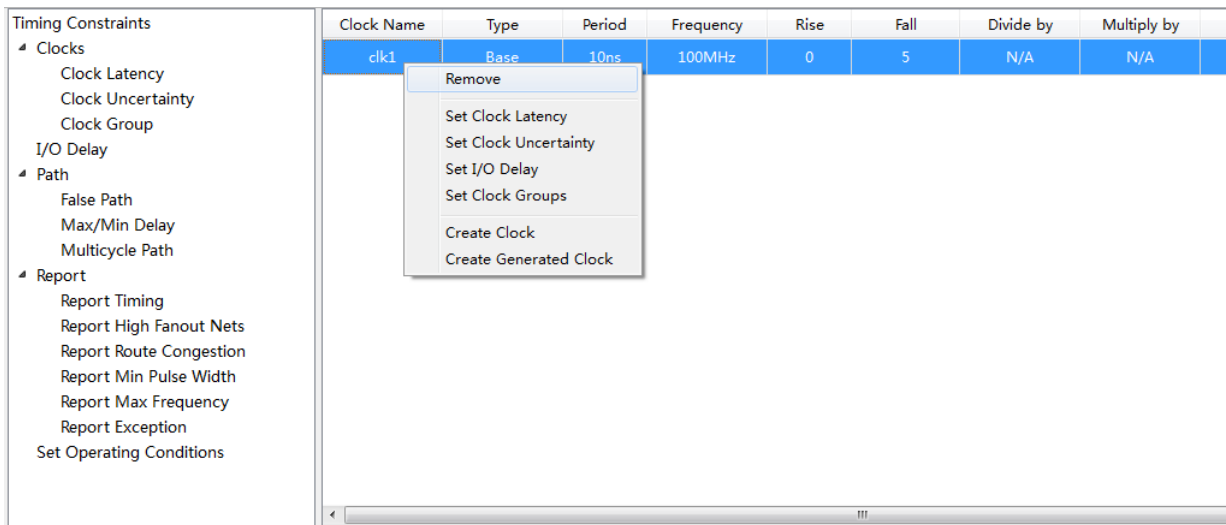
1. 在菜单栏中，单击“Constraints”，在其下拉菜单中，选择时序约束命令，通过选取相应的约束命令打开约束命令的图形界面(GUI)，如图 4-10 所示；

图 4-10 菜单打开时序约束界面



2. 在 STA 软件工具右侧的表格界面中，单击鼠标右键，根据中间栏不同的选项，选取不同的时序约束命令，如图 4-11 所示。

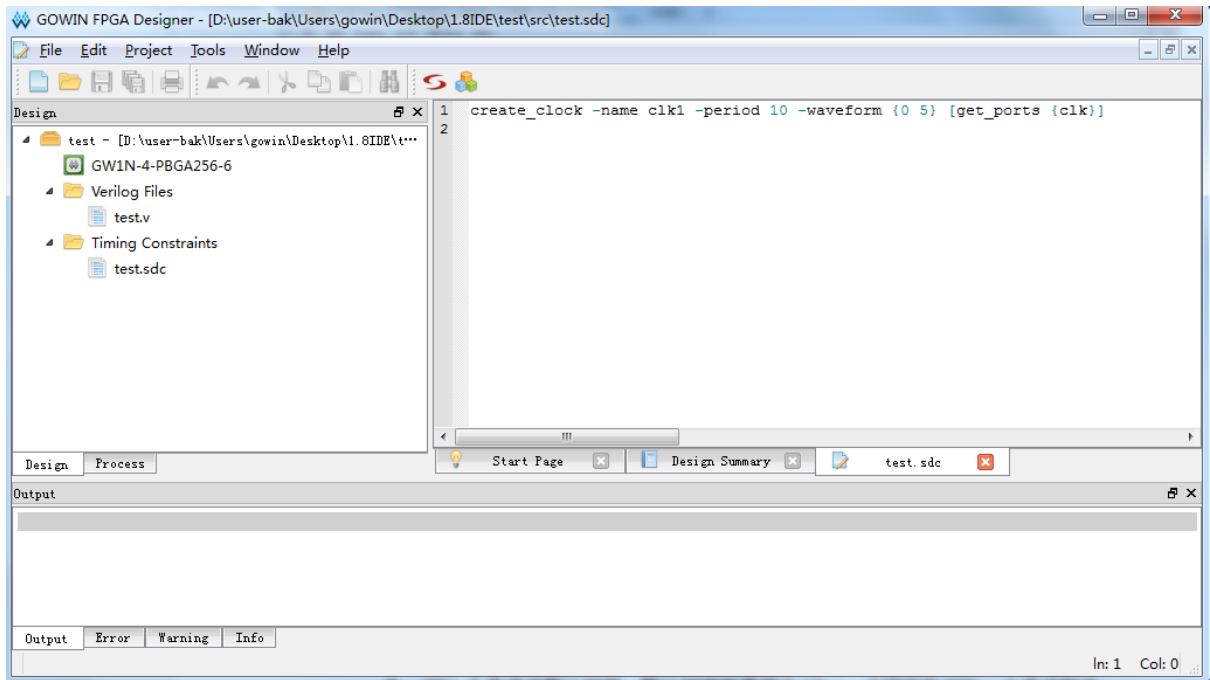
图 4-11 右键打开时序约束界面



4.3编辑 SDC 文件

支持读取工程的 SDC 文件，并可在工具编辑器中手动修改约束，操作便捷。使用方式如图 4-12 所示。

图 4-12 编辑 SDC 文件



4.4 时序约束编辑

4.4.1 Clock 约束

Create Clock

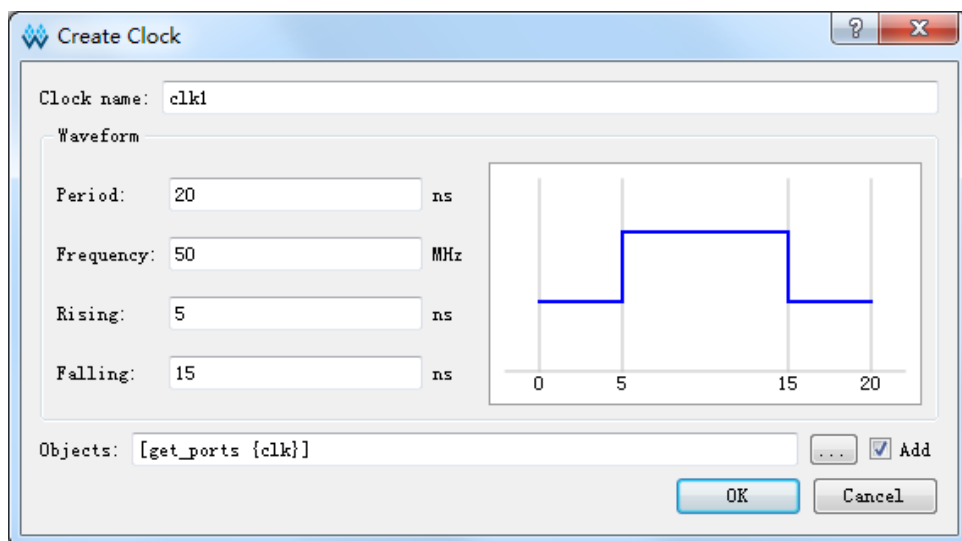
- 通常用于创建系统主时钟，可指定时钟名称、周期、波形、占空比，以及该时钟源；
- STA 工具提供默认时钟，默认时钟的周期为 10ns，占空比为 50%，上升沿到达；
- STA 工具可建立多个时钟，形成多个时钟域，支持跨时钟域的分析。

可通过以下两种方式新增 Clock 约束。操作如下：

1. 通过“Constraints”菜单新增 Clock 约束：

- a). 在“Constraints”菜单中，选择“Create Clock...”，弹出“Create Clock”对话框，如图 4-13 所示；

图 4-13 创建 Clock 约束

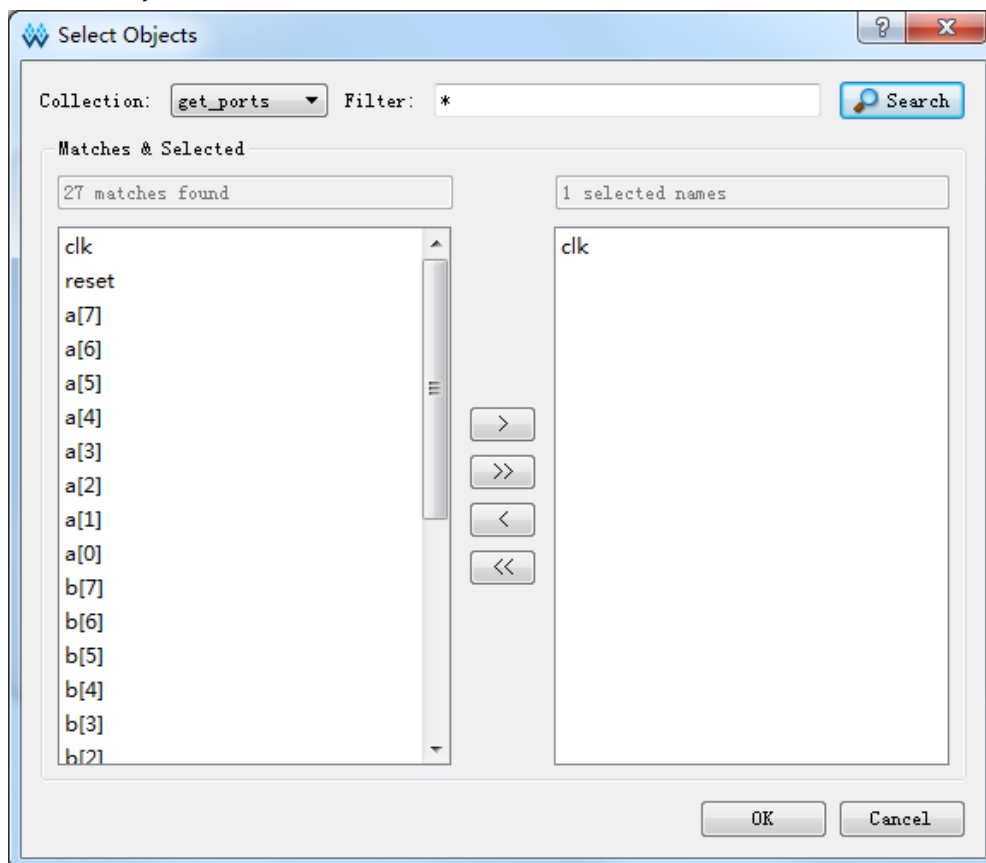


b). 填写 Clock 信息，包括 Clock Name、Waveform、Objects。

注！

- Waveform 右边的图形是根据填写的时钟信息显示的 clock 波形；
- 单击 Objects 右侧的“...”按钮，会弹出“Select Objects”对话框，如图 4-14 所示；
- 当目标对象上已有时钟时，勾选 Add 可将时钟添加到目标对象上，否则新建的时钟将被忽略；
- 默认的 Waveform 周期为 10ns，0ns 处为上升沿，半周期处为下降沿。

图 4-14 Objects 列表

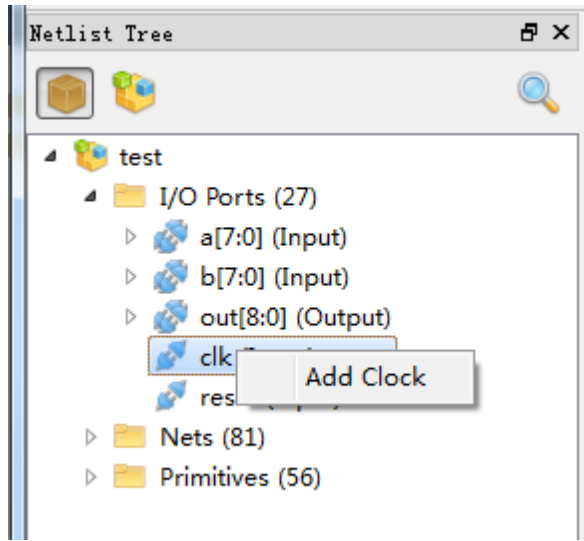


- 在图 4-14 中，各项操作及功能如下：
- “Collection”指定搜索的对象类型；
- “Filter”处为通配符筛选；
 - 搜索后选择 **Objects**，右侧列表为已选中列表；
 - “>” 按钮将左边列表中的选中项添加到右边列表；
 - “>>” 按钮添加左边所有项；
 - “<” 按钮移除右边选中项；
 - “<<” 按钮移除右边所有项；
 - 单击“OK”，完成 **Objects** 的添加。

2. 通过 Netlist Tree 新增 Clock 约束：

- 在 Netlist Tree 中，选中 I/O Port 或 Net；
- 单击鼠标右键，选择“Add Clock”，创建新的 clock，如图 4-15 所示。

图 4-15 创建 clock 约束



3. 时钟创建完成后，Clock 列表中会增加对应的约束，如图 4-16 所示。

图 4-16 Clock 约束列表

Clock Name	Type	Period	Frequency	Rise	Fall	Divide by	Multiply by	Duty cycle	Phase	Offset
clk1	Base	10ns	100MHz	0	5	N/A	N/A	N/A	N/A	N/A
clk2	Base	20ns	50MHz	0	10	N/A	N/A	N/A	N/A	N/A

在该列表中，可进行如下操作：

1. 编辑 Clock，双击“Clocks”列表中对应的约束，打开 Clock 的编辑对话框，可在对话框中编辑修改 Clock 信息；
2. 删除 Clock，在列表中选择该条 Clock，单击鼠标右键，选择“Remove”；
3. 选中某个 Clock，右键菜单，可快速为该条 Clock 设置 Clock Latency、Clock Uncertainty 或 I/O Delay 信息，如图 4-17 所示。

图 4-17 Clock 列表右键内容

Clock Name	Type	Period	Frequency	Rise	Fall	Divide by	Multiply by	Duty cycle
clk1	Base	10ns	100MHz	0	5	N/A	N/A	N/A
clk2	Base	20ns	50MHz	0	10	N/A	N/A	N/A

Remove
Set Clock Latency
Set Clock Uncertainty
Set I/O Delay
Set Clock Groups
Create Clock
Create Generated Clock

Generated Clock

- 用于创建一个基于主时钟的衍生时钟，主时钟和衍生时钟是同一个时钟域；
- 通过该约束可基于主时钟进行分频、倍频、相移和调整占空比等操作，进而完成衍生时钟的创建。

可通过以下两种方式新建 Generated Clock:

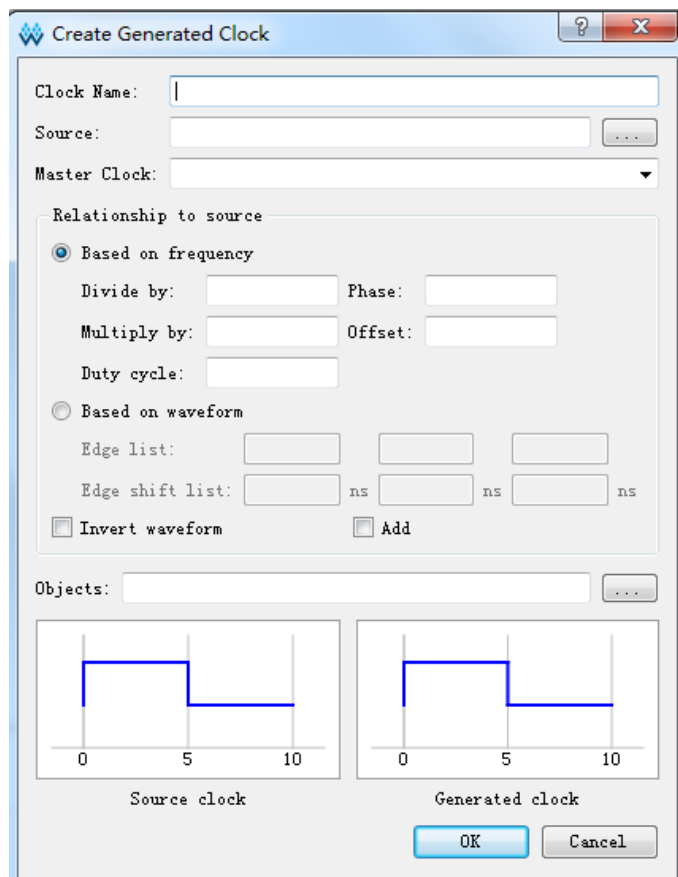
1. 通过“Constraints”菜单创建

- 在“Constraints”菜单中，选择“Create Generated Clock”，弹出“Create Generated Clock”对话框，如图 4-18 所示；
- 选择“Source”，将与 Source 关联的 Clock 添加到“Master Clock”列表中，选择 master clock；
- Objects 栏，单击 Objects 右边的“...”按钮，会弹出“Select Objects”对话框，选择目标对象。

注！

- 如选择的 Source 上无 Clock，则 Master Clock 无选项，需重新选择 Source；
- 在下方的 Waveform 波形图中，左侧图形是 Master Clock 的波形图，右侧是依据 Generated 选项（如倍频、分频）生成的 Generated Clock 的波形图，图中显示 clock 的上升沿、下降沿及周期信息。

图 4-18 创建 Generated Clock 约束



2. 通过 Clocks 列表创建 Generated Clock:

- 在 **Clocks** 列表中，在空白处右键单击菜单；
- 选择“**Create Generated Clock**”新建 **Generated Clock**，如图 4-19 所示。

图 4-19 Clock 列表右键内容

Clock Name	Type	Period	Frequency	Rise	Fall	Divide by	Multiply by	Duty cycle
clk1	Base	10ns	100MHz	0	5	N/A	N/A	N/A
clk2	Base	20ns	50MHz	0	10	N/A	N/A	N/A

Create Clock
Create Generated Clock

添加后表格编辑区会增加新建的约束。

在该列表中，可进行如下操作：

1. 编辑 **Generated Clock** 约束，双击“**Clocks**”列表中对应的约束，打开 **Generated Clock** 的编辑对话框，在对话框中编辑修改 **Generated Clock** 信息；
2. 删除 **Generated Clock**，在表格编辑区选中该 **Clock**，右键选择“**Remove**”。

Clock Latency

约束说明

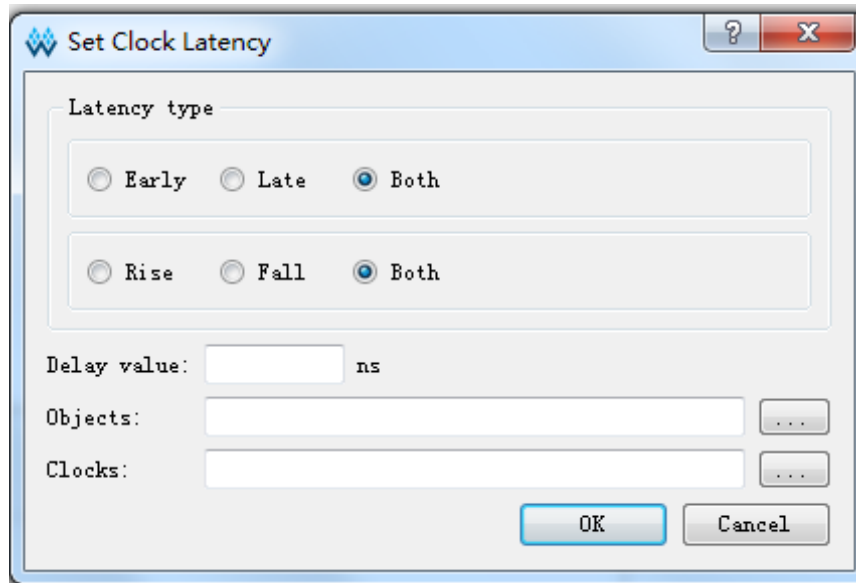
1. 用于设置时钟信号到达接入点之前的延时，通过参数的选择可对时钟的上升沿/下降沿到达接入点的最大/最小延时分别进行精确的设置；
2. 时钟延时分为两种：网络 (**network**) 延时和源 (**source**) 延时。
 - 网络 (**network**) 延时是芯片内部时钟路径的延时；
 - 源 (**source**) 延时是芯片外部时钟路径的延时；
 - **STA** 工具自动计算时钟的网络 (**network**) 延时，所以用户只需设定源 (**source**) 延时。

界面操作

可通过以下两种方式新建 **Clock Latency** 约束：

1. 通过“**Constraints**”菜单新建 **Clock Latency** 约束：
在“**Constraints**”菜单中，选择“**Set Clock Latency**”，弹出“**Set Clock Latency**”对话框，如图 4-20 所示；

图 4-20 创建 Clock Latency 约束



填写 Latency 信息，单击“OK”保存约束。

2. 通过 Clocks 列表新建 Clock Latency 约束。

在 Clocks 列表中选中某个 Clock，右键选择 Set Clock Latency 为该 Clock 设置 Latency 信息。

注！

根据此方式创建 Latency 信息，图 4-20 中的 Objects 将自动指定为该 Clock。

Clock Uncertainty

约束说明

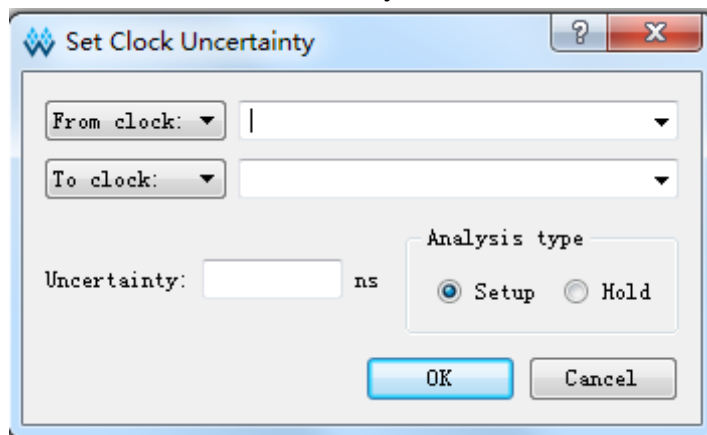
1. 设置时钟的不确定量或偏移量，用于时钟传递的分析；
2. 可分别对 setup 和 hold 设置不确定量，也可对时钟上升沿和下降沿的传输分别设置不确定量；
3. 允许用户将时钟抖动(jitter)、悲观度(pessimism)等通过该约束通知高云 STA 工具，进而影响时序计算。

界面操作

新建 Clock Uncertainty，操作如下：

1. 在“Constraints”菜单中，选择“Set Clock Uncertainty”，弹出“Set Clock Uncertainty”对话框，如图 4-21 所示；

图 4-21 创建 Clock Uncertainty 约束



2. 通过左侧的下拉框选择 From 的类型（From clock、Rise from、Fall from）和 To 的类型（To clock、Rise to、Fall to），通过右侧的下拉框从当前所有已创建的 Clock 中选择目标的 Clock；
3. 填写信息完成后，单击“OK”保存约束，完成 Uncertainty 的添加。

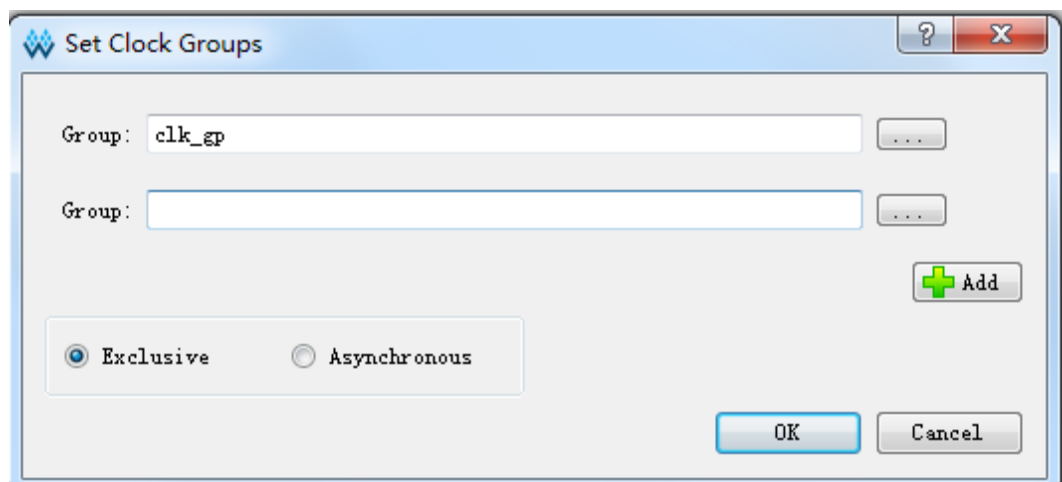
Clock Group

- 用于指定不同时钟之间的关系；
- STA 工具默认提供组内成员之间相关，组与组之间不相关。

可通过以下方式新建 Clock Group：

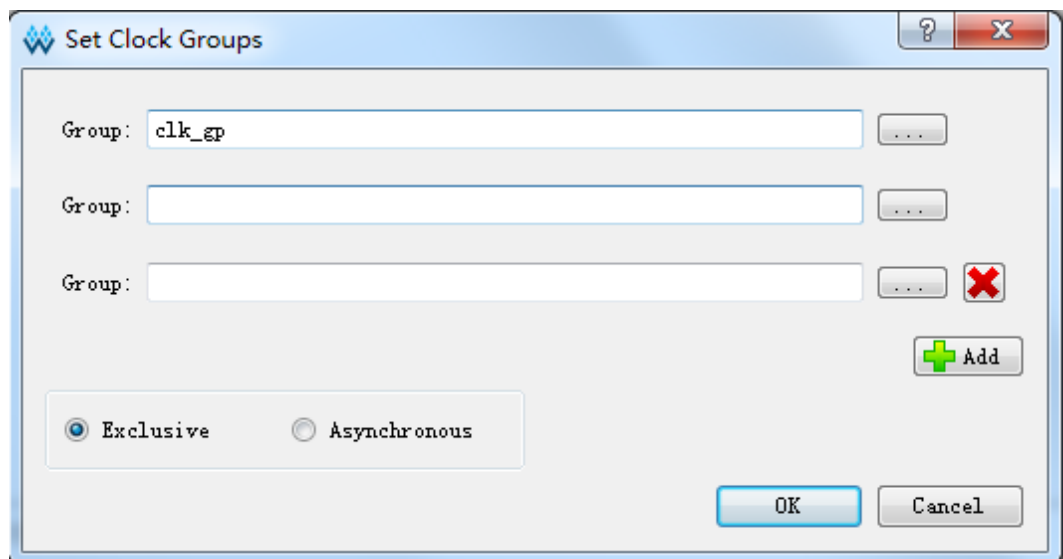
1. 在“Constraints”菜单中，选择“Set Clock Groups”，弹出“Set Clock Groups”对话框，如图 4-22 所示；

图 4-22 创建 Clock Group 约束



2. 单击“...”按钮，为 Group 选择 Clock。如图 4-23 所示；

图 4-23 Clock Group 成员列表



3.如需删除添加的 Group，单击对应条目右侧的“**X**”按钮；

4.单击“OK”，保存约束。

注！

如需设置多个“Group”，单击“Add”按钮，即会新增一行。

4.4.2 I/O Delay 约束

set_input_delay

- 规定数据到达某指定输入端口(PORT)的时间，与输入(PORT)相关的时钟通过“-clock”参数指定，该时钟须为设计中存在的时钟；
- 输入延时可规定与时钟的上升沿相关（默认）或下降沿相关（由参数“-clock_fall”指定）。

注！

- 输入延时中可包含外部时钟延时，默认的情况下，计算到达延时时应加上外部时钟延时，当指定参数“-source_latency_included”时，则计算到达延时不加入外部时钟延时；
- 默认情况下，对一端口(PORT)添加基于相同时钟的该类约束，若第一条指定参数“-clock_fall”，而第二条未指定“-clock_fall”参数，则第二条会覆盖第一条约束，除非指定-add_delay 参数；
- STA 工具产生的时序报告中输入延时类型为“tln”。

set_output_delay

指定数据经端口(PORT)的输出延时，同时须指定该输出延时的参考时钟，默认情况下，输出延时与时钟的上升沿相关，可通过使用参数“-clock_fall”指定该输出延时与时钟的下降沿相关。

注！

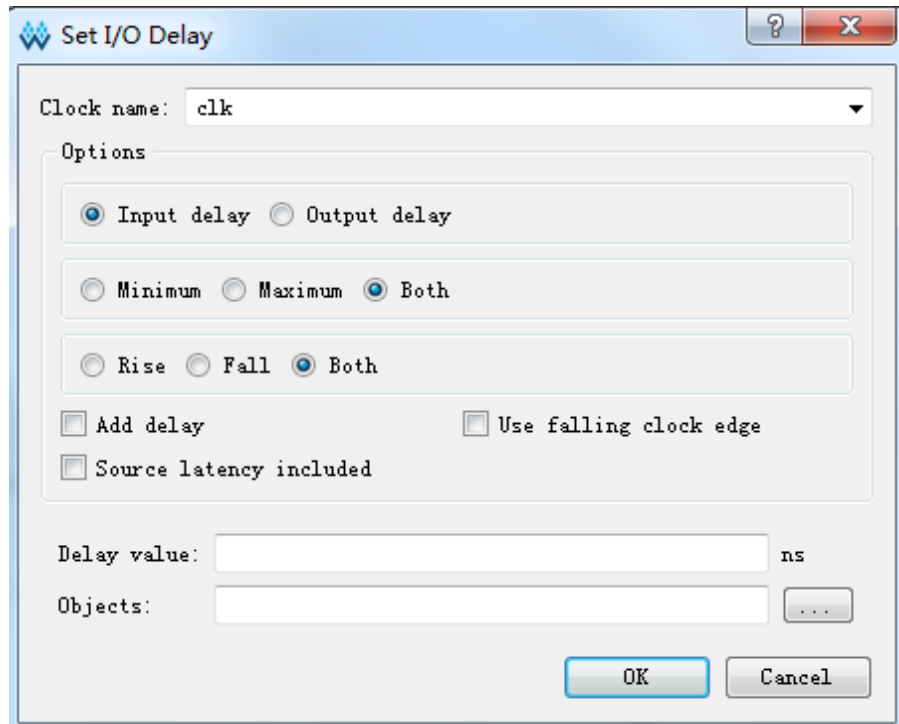
- 默认情况下，外部时钟延时不包含在输出延时中，当使用参数“-source_latency_included”时，表示输出延时中已包含外部时钟延时；

- 默认情况下，该类约束会覆盖添加在同一个端口(PORT)上且具有相同时钟、不同时钟参考沿的约束，使用参数“-add_delay”可避免发生覆盖；
- STA 工具的时序报告中，输出延时的类型为“tOut”。

可通过以下方式新建 I/O Delay 约束：

1. 在“Constraints”菜单中，选择“Set I/O Delay”，弹出“Set I/O Delay”对话框，如图 4-24 所示；

图 4-24 创建 I/O Delay 约束



2. I/O Delay 有 Input Delay 和 Output Delay 类型，首先需选择 Delay 类型；
3. Delay 信息填写完成后，单击“OK”保存约束。

4.4.3 Path 约束

False Path

时序分析时，通过该约束设置不需进行时序分析的路径。

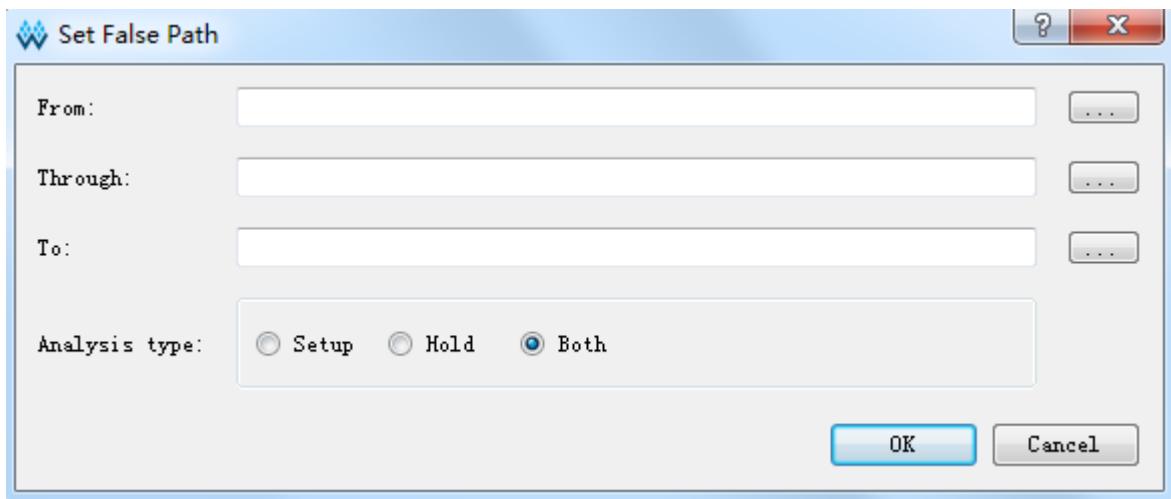
注！

STA 工具提供默认的所有的时钟路径都是相关的，默认支持跨时钟域处理。

新建 False Path 约束。操作如下：

1. 选择“Constraints > Set False Path”，弹出“Set False Path”对话框，如图 4-25 所示；

图 4-25 创建 False Path 约束



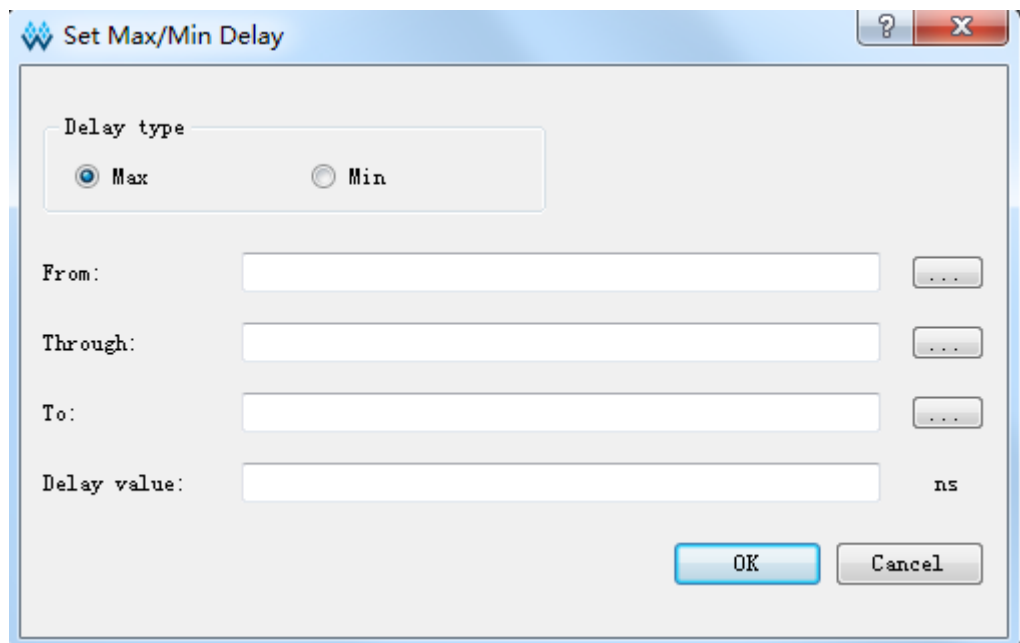
2. 单击右侧按钮 “...” 选择 From 和 To 对应的 Object。单击 “OK” 保存约束。

Max/Min Delay

新建 Max/Min Delay 约束，操作如下：

选择 “Constraints > Set Max/Min Delay”，弹出 “Set Max/Min Delay” 对话框，如图 4-26 所示；

图 4-26 创建 Max/Min Delay 约束



Delay Type 选择 Delay 的类型 (Max or Min)，From 和 To 的选择同 set false path 约束。填写完成 Delay 信息后，单击 “OK” 完成创建。

MultiCycle Path

默认情况下，STA 工具执行的是单周期时钟分析，即建立时间的检查是

在源时钟边沿的下一个时钟周期的有效时钟沿。但此方式对某些特定的时序路径不适用。逻辑设计电路分析是最典型实例，一条逻辑电路计算较复杂或路径较长，需多于一个时钟周期的时间数据方能稳定。STA 工具提供多周期路径设置命令 `set_multicycle_path`，允许用户设置 N 个时钟周期的路径检查，参数中的 `path multiplier` 定义了传播路径大于一个时钟周期的信号的总的时钟周期数，即从信号的起始源点到目的点之间信号路径传播的时间周期总数。

多周期路径设置命令 `set_multicycle_path` 可用于规定多周期路径的建立时间分析和保持时间分析、上升沿/下降沿的分析、发起时钟/锁存时钟的分析等，具体内容可参考该命令的参数。

用来放松发起时钟 (launch clock) 和锁存时钟 (latch clock) 的相对关系，也是时序例外的一种，应用场景包括非关键路径和关联时钟路径等。

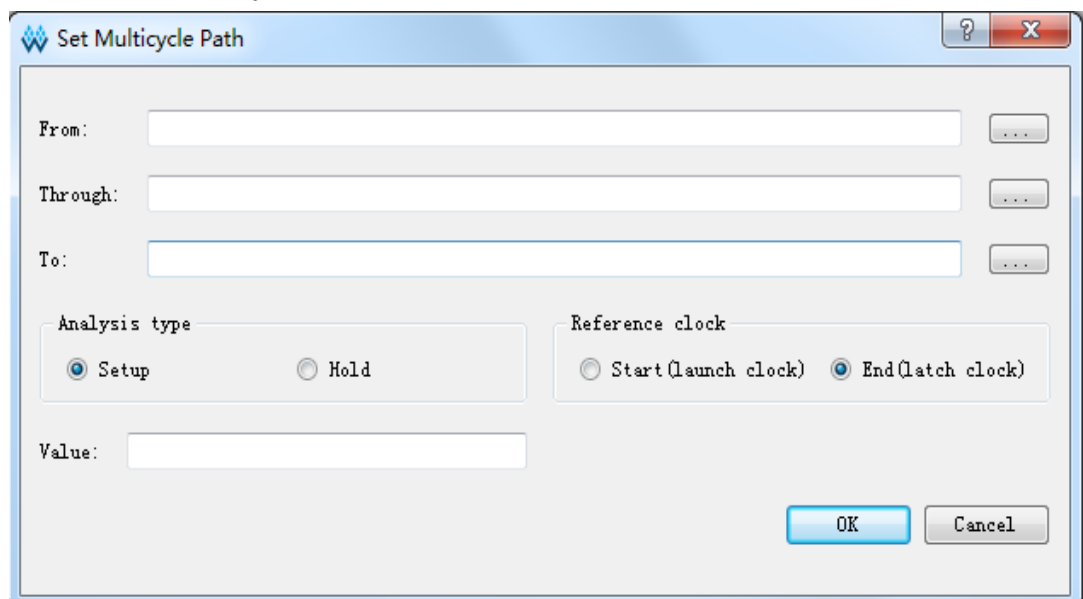
注！

设置多周期路径命令会对建立时间 (setup) 和保持时间 (hold) 造成一定影响，如未指明 `-setup` 或者 `-hold` 选项，则 STA 工具默认为 `-setup`。如已设置 `-setup` 值，则 `hold` 值不会受其影响。STA 工具默认提供自动修复 hold 的功能。如用户指定 `hold` 值，STA 工具会优先考虑用户设置的约束。

新建 Multicycle Path 约束，操作如下：

1. 选择 “Constraints > Set Multicycle Path”，弹出 “Set Multicycle Path” 对话框，如图 4-27 所示。

图 4-27 创建 Multicycle Path 约束



2. 填写对话框中相关信息，单击 “OK” 保存约束。

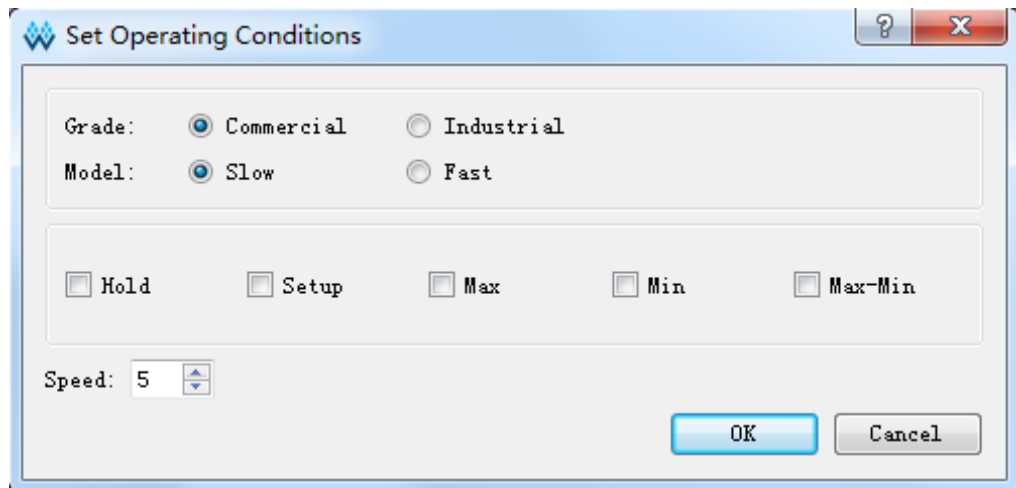
4.4.4 工作条件约束

设置 FPGA 芯片的温度等级、速度等级、时序分析的工艺角等约束。

新建 Operating Conditions 约束，操作如下：

- 选择 “Constraints > Set Operating Conditions”，弹出 “Set Operating Conditions” 对话框，如图 4-28 所示。

图 4-28 创建 Operating Conditions 约束



4.4.5 时序报告

使用时序约束编辑器界面的操作如下所示：

Report Timing

- 报告内容

时序报告，根据设置的时序报告的参数，文件输出报告内容。

- 界面操作

相关操作步骤如下：

1. 在主界面中，选择“Timing Constraints Report Timing”；
2. 在右侧空白处右键，出现“Create Report”，如图 4-29 所示；
3. 选择“Create Report”弹出如图 4-30 所示的界面；
4. 填写对话框中相关信息，单击“OK”，保存时序报告设置。

图 4-29 Report Timing 创建界面

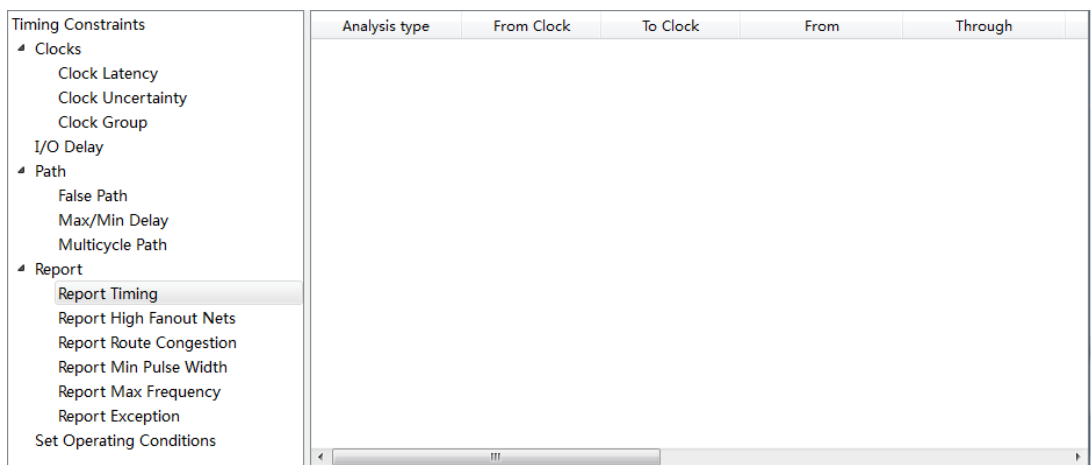
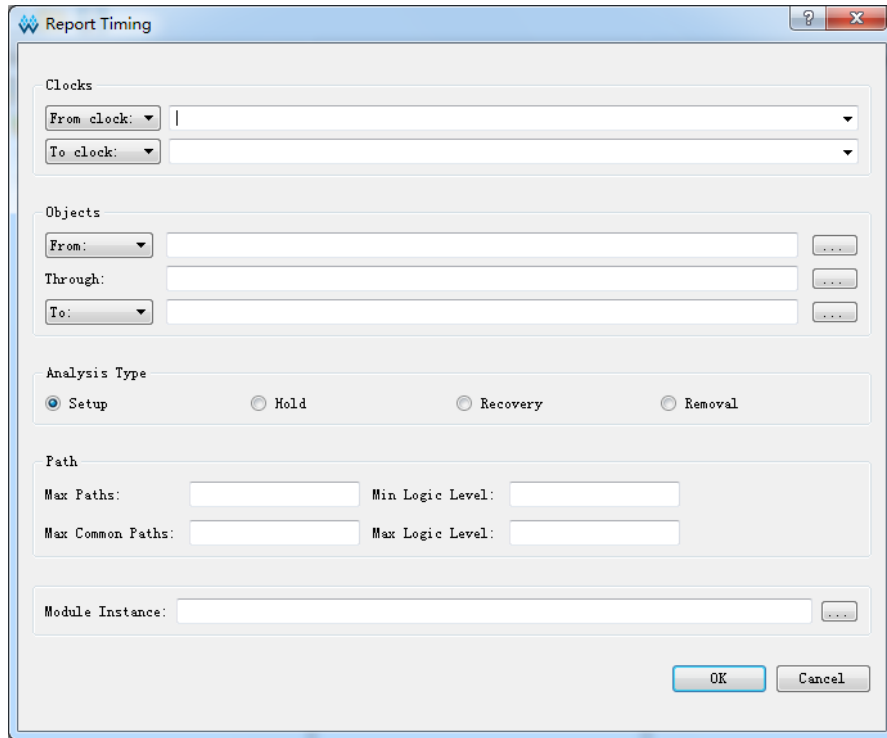


图 4-30 Report Timing 界面



Report High Fanout Nets

- 报告内容

时序报告最大扇出的 Net 数目。

- 界面操作

1. 在主界面中，选择“Timing Constraints > Report High Fanout Nets”；
2. 在右侧空白处右键，出现“Create Report”，如图 4-31 所示；
3. 选择“Create Report”，弹出如图 4-32 所示的界面；
4. 填写对话框中相关信息，单击“OK”，保存时序报告设置。

图 4-31 Report High Fanout Nets 创建界面

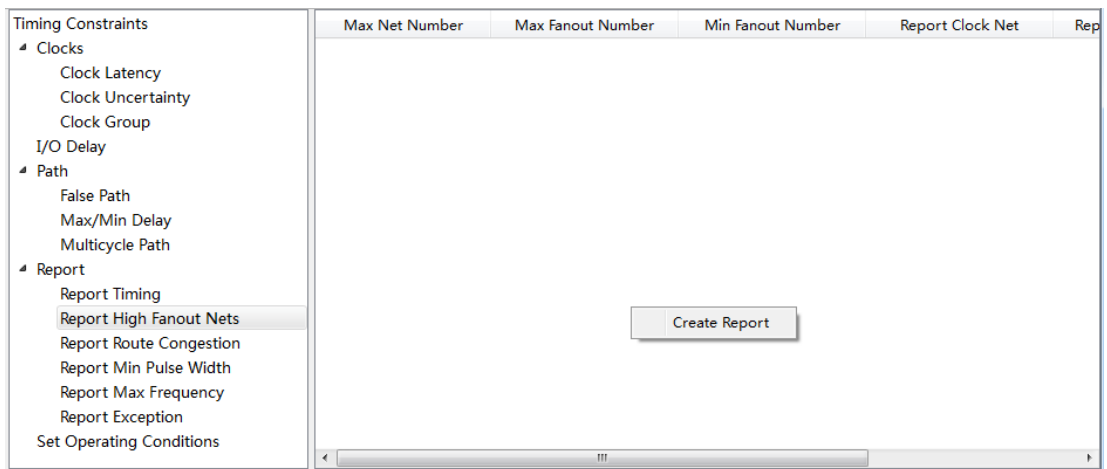
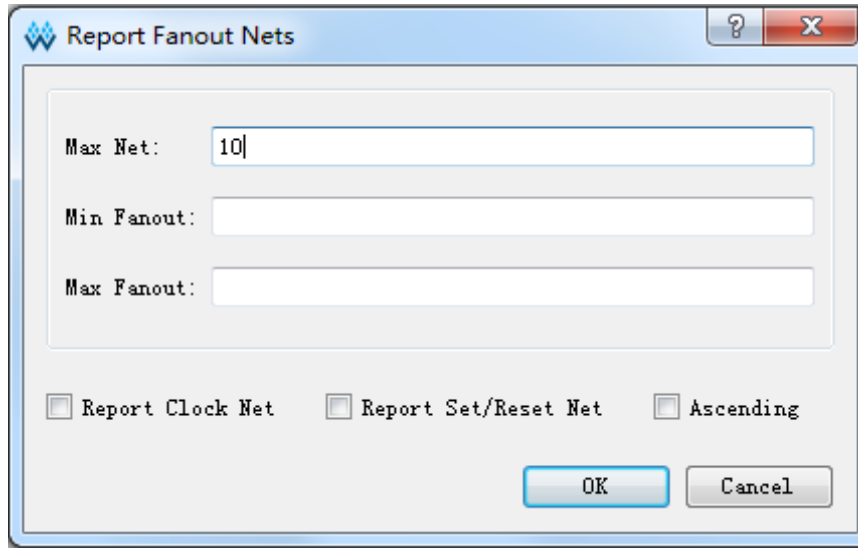


图 4-32 Report High Fanout Nets 界面



Report Route Congestion

- 报告内容

时序报告拥塞度情况。

- 界面操作

相关操作步骤如下：

1. 在主界面中，选择“Timing Constraints > Report Route Congestion”；
2. 在右侧空白处右键，出现“Create Report”，如图 4-33 所示；
3. 选择“Create Report”，弹出如图 4-34 所示的界面；
4. 填写对话框中相关信息，单击“OK”，保存时序报告设置。

图 4-33 Report Route Congestion 创建界面

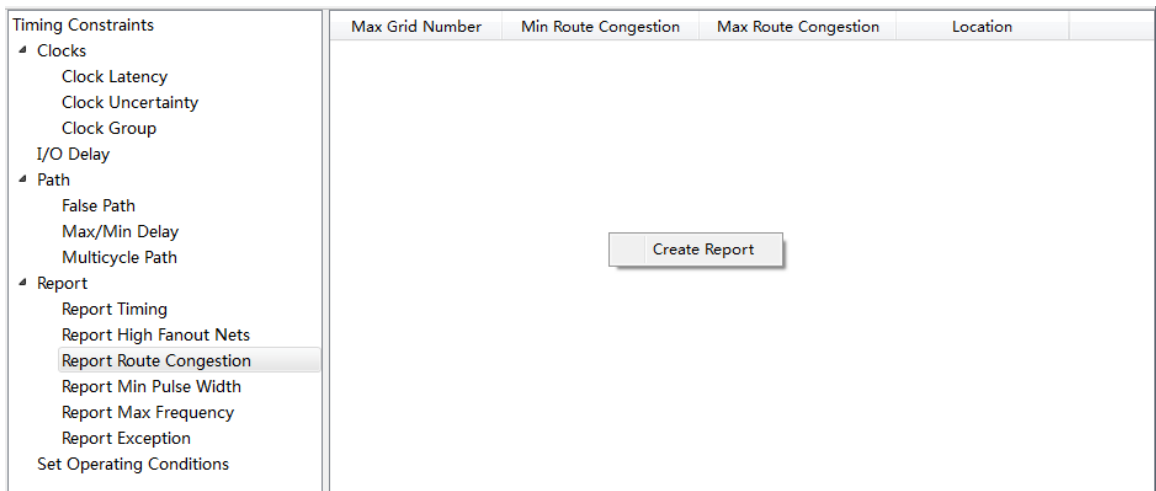
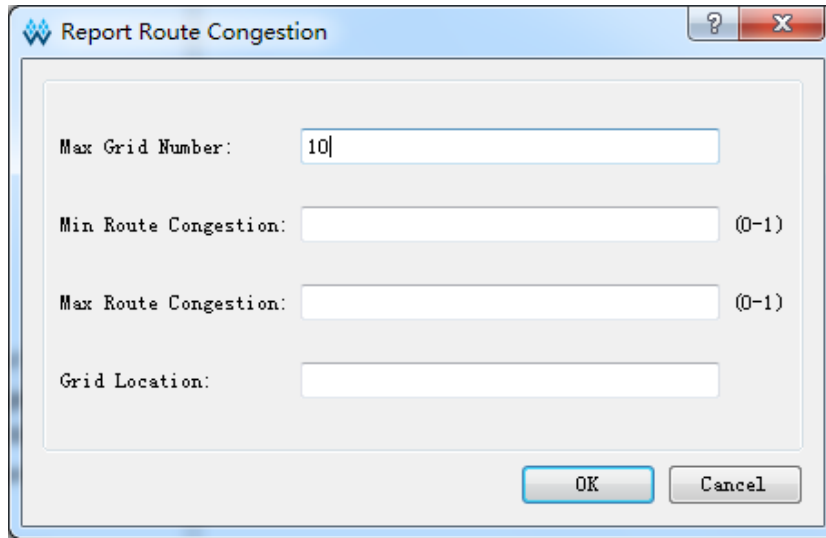


图 4-34 Report Route Congestion 界面



Report Min Pulse Width

- 报告内容
 - 时序报告最小脉冲宽度。
- 界面操作
 1. 在主界面中，选择“Timing Constraints > Report Min Pulse Width”；
 2. 在右侧空白处右键，出现“Create Report”，如图 4-35 所示；
 3. 选择“Create Report”出现如图 4-36 所示的界面；
 4. 填写对话框中相关信息，单击“OK”，保存时序报告设置。

图 4-35 Report Min Pulse Width 创建界面

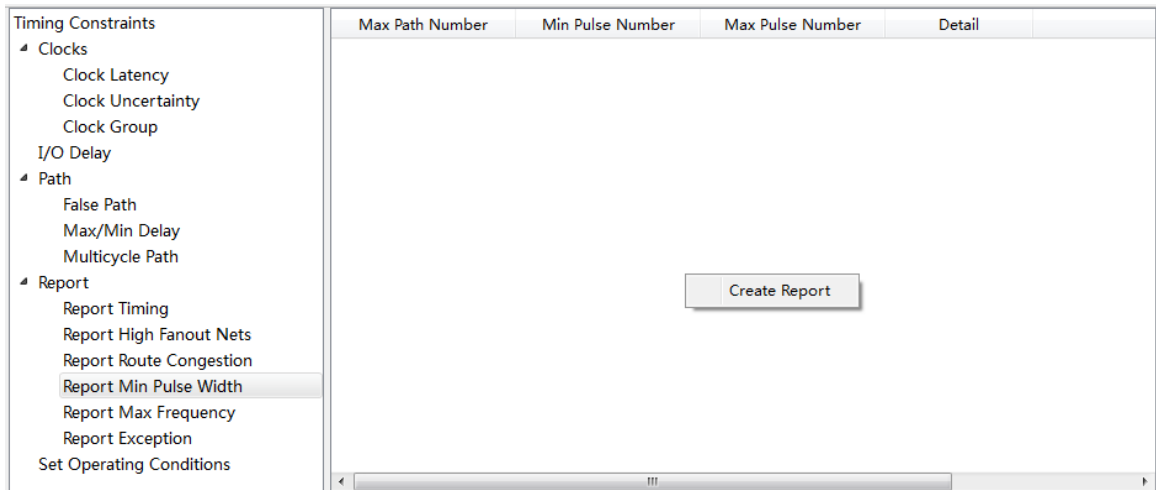
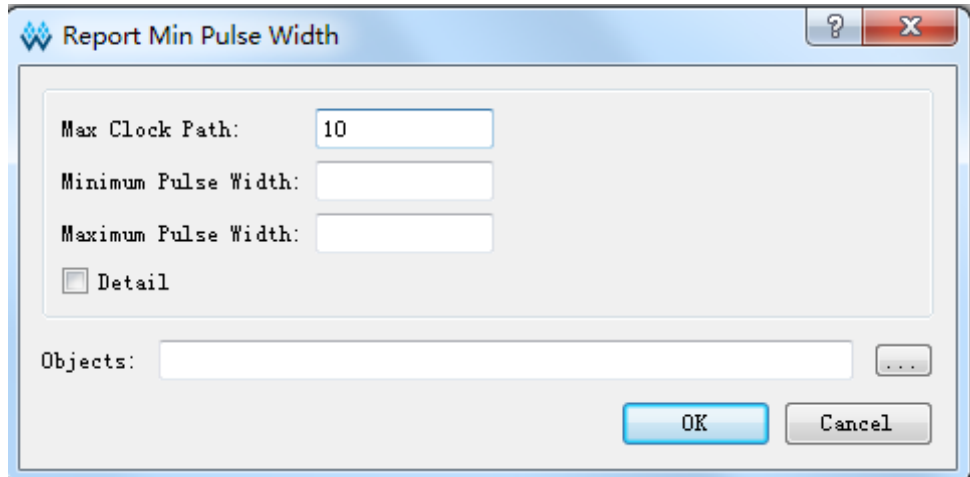


图 4-36 Report Min Pulse Width 界面



Report Max Frequency

- 报告内容

最大频率报告。

- 界面操作

相关操作步骤如下：

1. 在主界面中选择 “Timing Constraints > Report > Report Max Frequency” ；
2. 在右侧空白处右键，出现 “Create Report” ，如图 4-37 所示；
3. 选择 “Create Report” ，弹出如图 4-38 所示的界面；
4. 填写对话框中相关信息，单击 “OK” ，保存时序报告设置。

图 4-37 Report Exception 创建界面

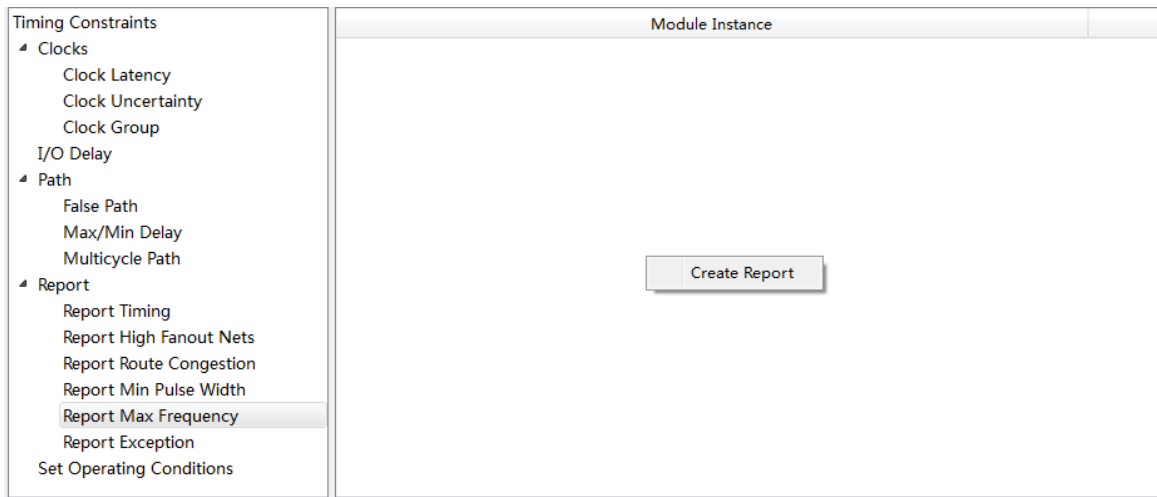
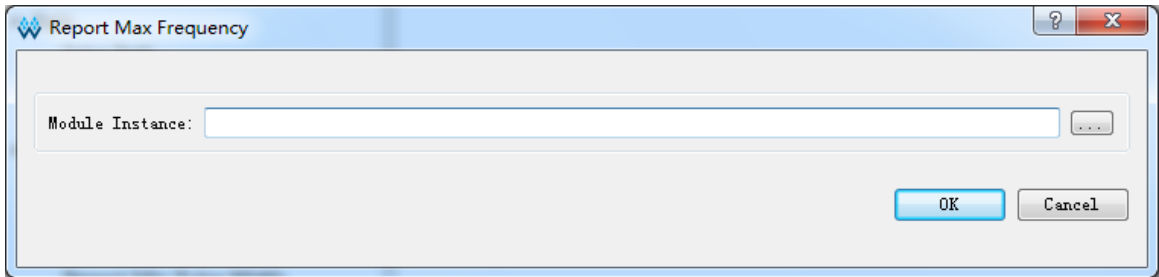


图 4-38 Report Max Frequency 界面



Report Exception

- 报告内容
 时序例外报告。
- 界面操作
 相关操作步骤如下：

1. 在主界面中选择“Timing Constraints > Report > Report Exception”；
2. 然后在右侧空白处右键，出现“Create Report”，如图 4-39 所示；
3. 选择“Create Report”，弹出如图 4-40 所示的界面；
4. 填写对话框中相关信息，单击“OK”，保存时序报告设置。

图 4-39 Report Exception 创建界面

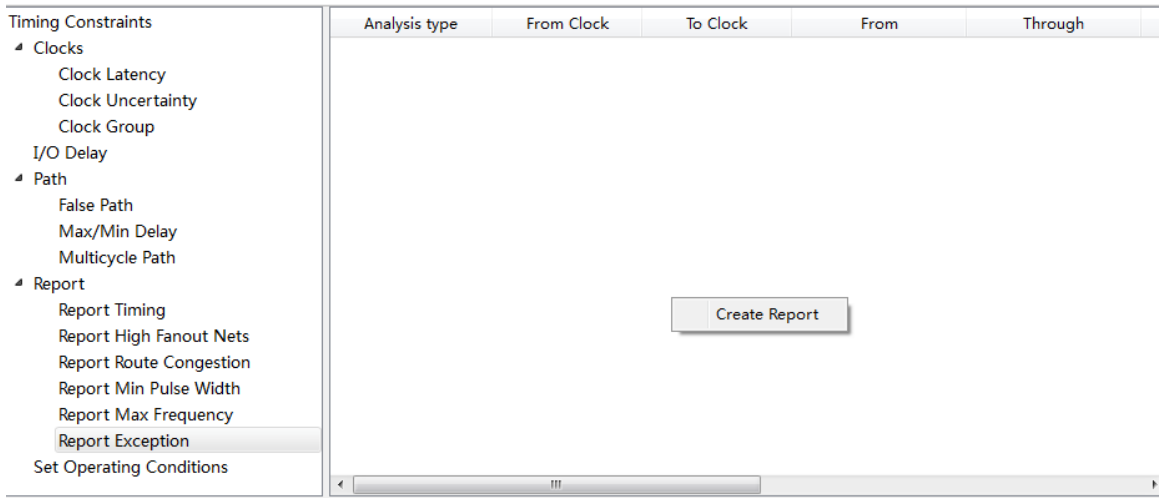
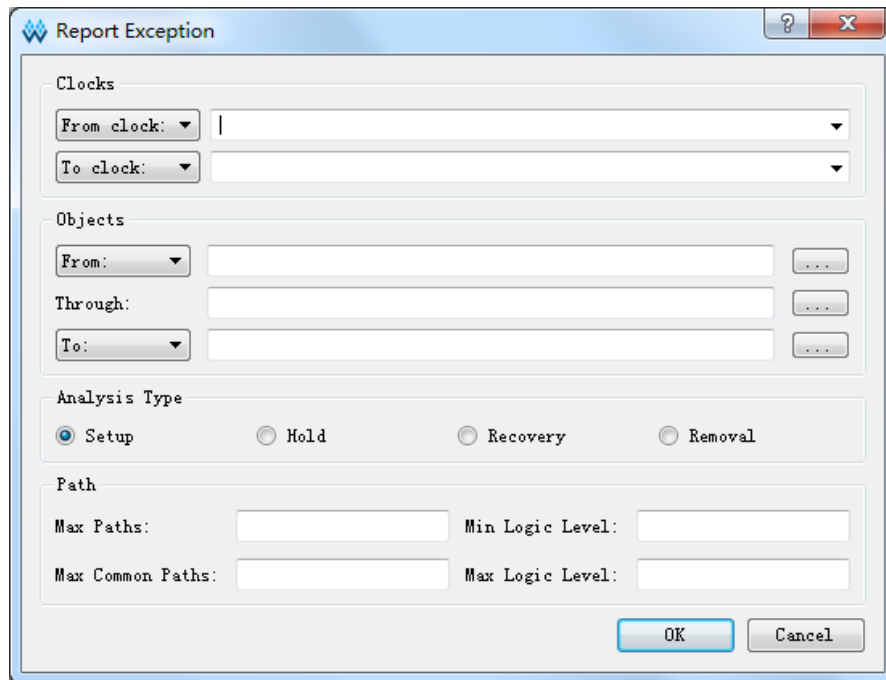
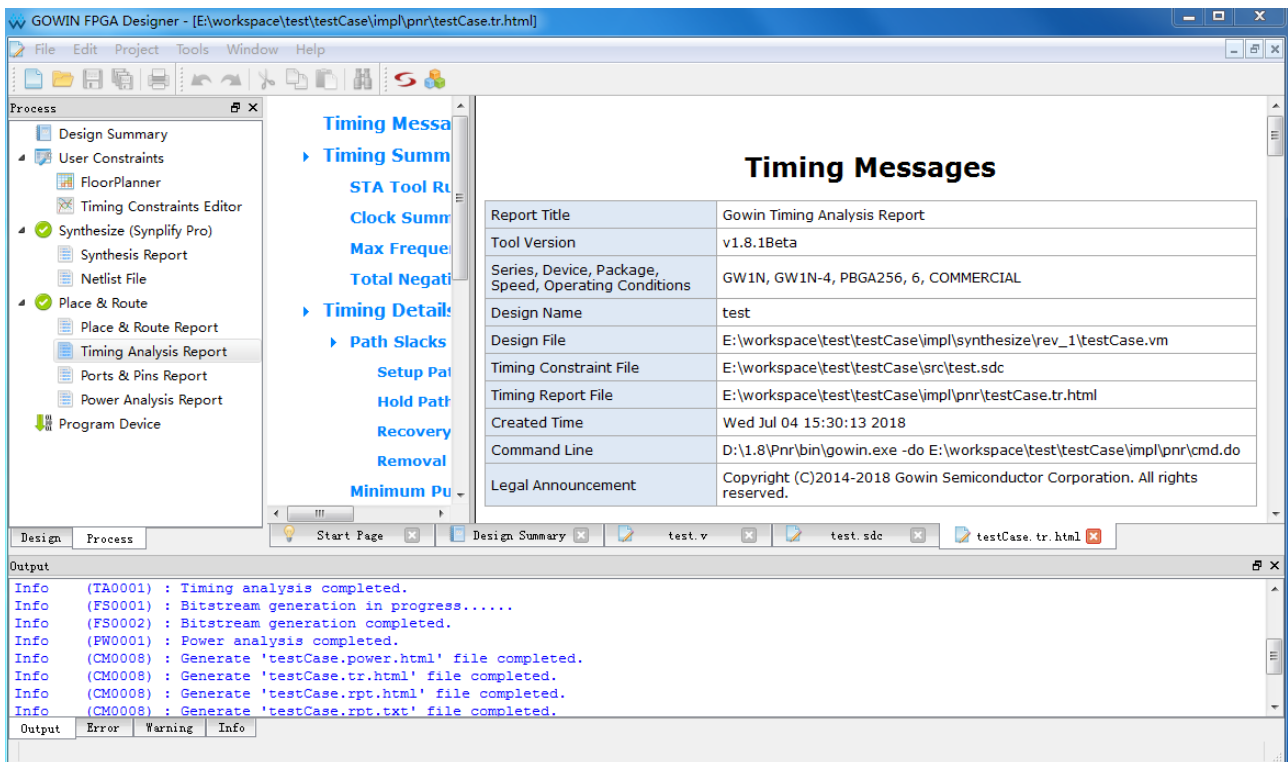


图 4-40 Report Exception 界面



对整个工程执行 Timing Analysis 时序分析后，软件会自动生成时序报告，切换至 Process 界面，双击“Timing Analysis Report”进行查看，如图 4-41 所示。单击中间窗口的命令，报告显示在右侧的窗口。

图 4-41 查看时序报告界面



4.4.6 保存与导出

所有约束编辑完成后，单击“File > Save”或“File > Save As”，可将当前编辑器中的约束信息保存至时序约束文件（.sdc）中，时序约束文件内容格式请参考附录 B.时序约束语法规范。

4.5 时序约束的优先级

STA 工具提供多种类型的时序约束，按照优先级由低到高依次是：

- 1.create_clock 和 create_generated_clock
- 2.set_multicycle_path
- 3.set_max_delay 和 set_min_delay
- 4.set_false_path
- 5.set_clock_groups

注！

只对在同一条时序路径上可能产生竞争的时序约束进行排序，其它未提及约束不会产生不同类型约束间的竞争。

5 时序优化

高云 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。包括:

- 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.posp 文件如图 5-1 所示,用于 FloorPlanner 读取布局布线后的位置信息。

- 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.运行 FloorPlanner 工具,打开布局布线后生成的文件.posp 文件,Ports、Primitive 等的位置约束信息见“Netlist”窗口,如图 5-2 所示;
- 8.打开时序约束文件.timing_paths 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 5-3 所示。

注!

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

分析并调整约束的位置。

- 9.在时序收敛的调试过程中,需找出工程中的关键路径,通过修改代码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:

c).查找关键路径对应的 Module 信息:

- 查看关键路径经过的 Instance 信息,从 Instance 名称中分辨出所属

Module;

- 右键单击关键路径，选择“**Highlight Corresponding Module**”，如图 5-4 所示。对应的 Module 将在 Module 列表中以红色名称显示，如图 5-5 所示。
 - d). 查看 Module 的位置分布信息和关键路径信号流向。
 - 选择一个 Module，右键单击选择“**Highlight Group Constraints**”，该模块组的位置信息白色高亮显示在“**Chip Array**”窗口中，可看到位置信息是相对集中还是分散；
 - 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 中选择一条关键路径，右键单击，选择“**Highlight**”，在“**Chip Array**”窗口中可观察到该路径的信号流向，如图 5-6 所示。
 - e). 调整不合理的位置信息。
如图 5-6 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。拖拽的方式调整位置不合理的那个模块的，减少信号的曲折路径，如图 5-7 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 5, 6, 7, 8, 9 过程。

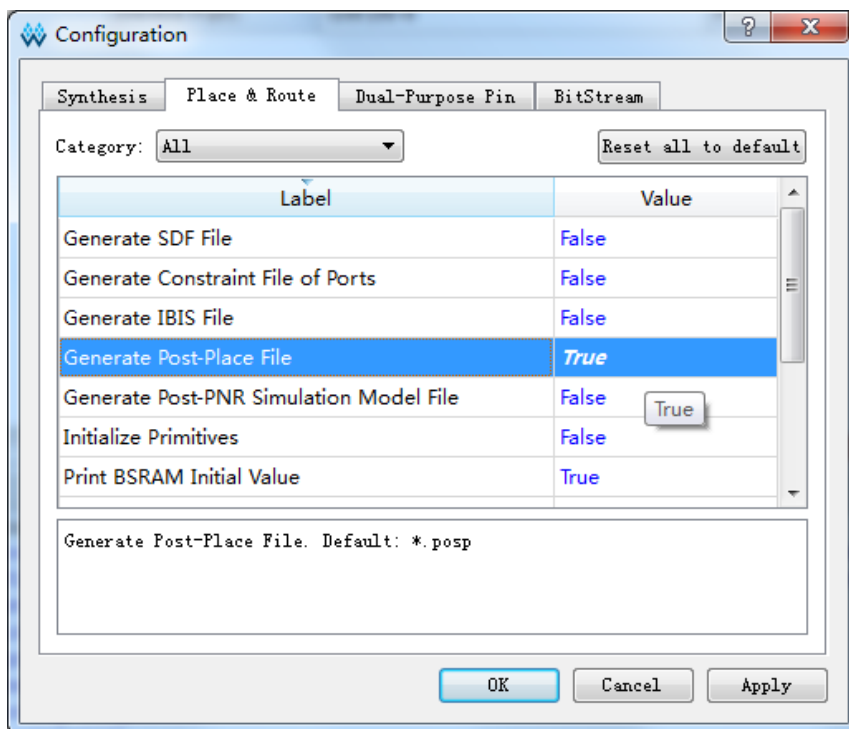
图 5-1 生成.posp 文件设置

图 5-2 读取.posp 文件

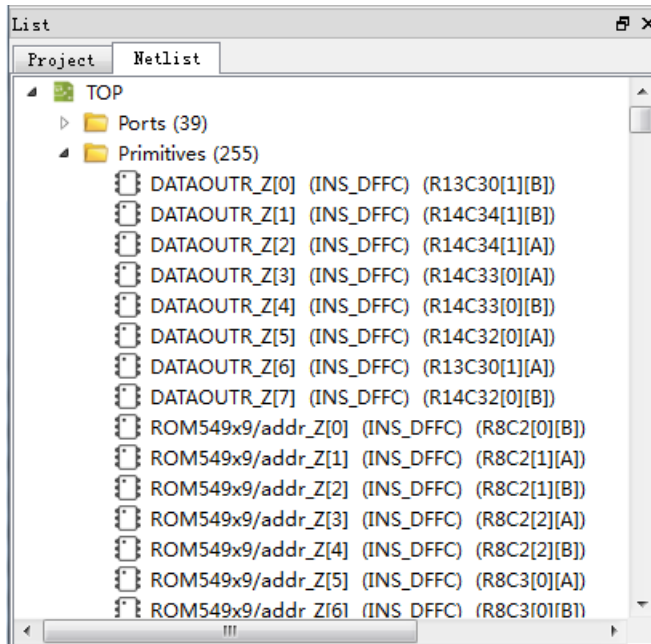


图 5-3 读取时序约束文件

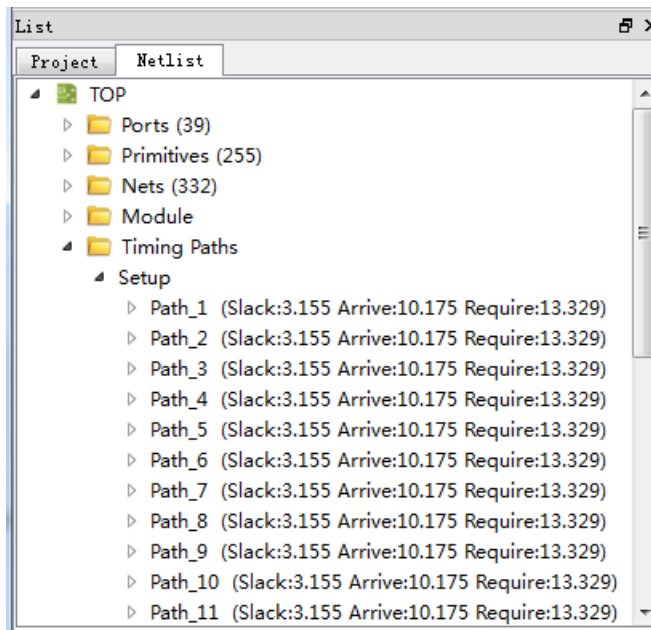


图 5-4 高亮关键路径的 Module 操作

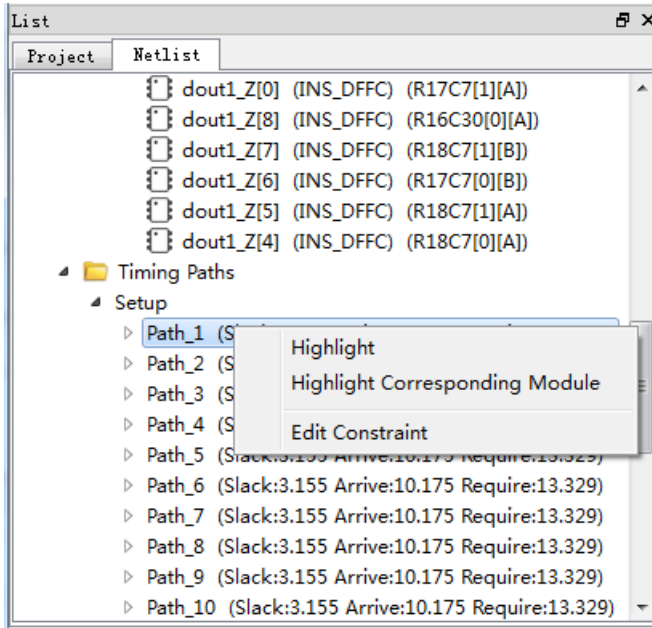


图 5-5 关键路径的 Module 显示红色

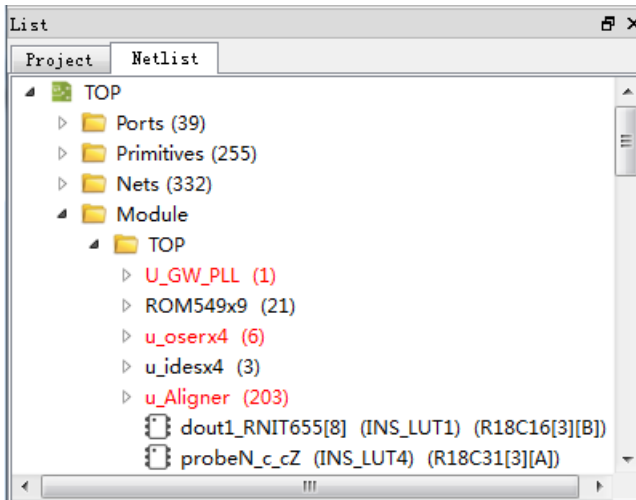


图 5-6 关键路径信号流向示意图

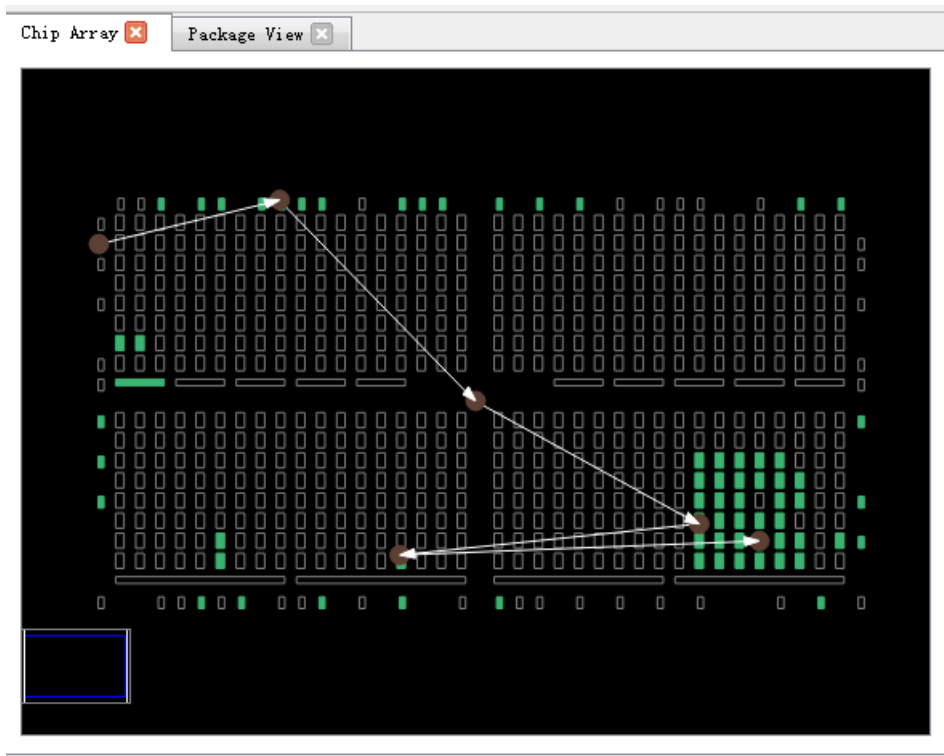
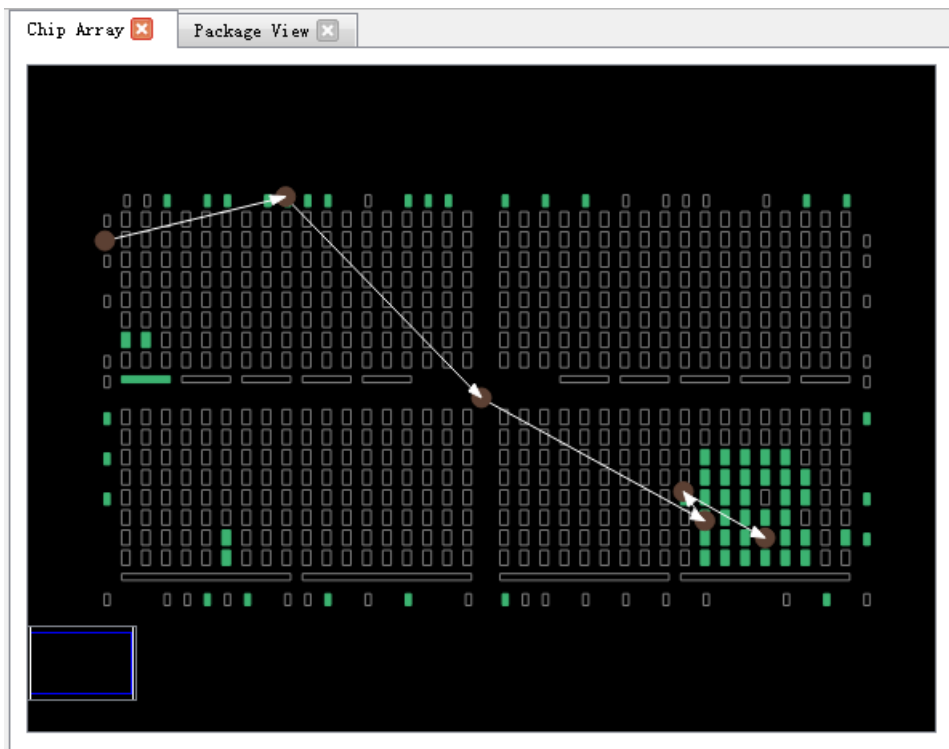


图 5-7 调整后的位置信息



附录 A.物理约束语法规范

A.1 I/O Constraints

IO 约束可将 port、buffer、I/O 寄存器以及 I/O 逻辑器件约束到指定 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、I/O 寄存器、I/O 逻辑器件的 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 指定的器件。

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。
```

示例 4.

IO_PORT “port_4” I3C_MODE=ON OPEN_DRAIN=OFF;
 // I3C_MODE/MIPI_INPUT/MIPI_OUTPUT 属性只有在
 GW1N6K/GW1N9K/GW1NR9K/ GW1NS2K/ GW1NSR2K 中可用，且
 I3C_MODE 属性和 OPEN_DRAIN 属性不能同时置为“ON”。

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/DLL 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：

- 1) 单一位置信息，指定到 LUT，如：RxCy[0-3][A-B]
- 2) 位置信息为一个范围，指定多行或多列，如：

包含多个 PLS 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]”

在一条约束语句中，可包含多个 ins_location，使用“,”分隔。

- 3) 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]”...

- 4) DLL 约束位置

DLL 约束位置为“DLL_TL”，“DLL_BL”，“DLL_TR”，

“DLL_BR”。

- 5) 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]”....

- 6) DSP 约束位置

DSP 约束位置格式为 “DSP_R19[0]”（第 19 行第一个 DSP Block），

或 “DSP_R19[1]”... 若需指定具体 macro, 可标记为: DSP_R19[0][A]
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 PLS 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 PLS 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_1 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_3 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_4 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 PLS 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。
```

示例 7.

```
INS_LOC "dll_name" DLL_TL;
// 通过对 DLL 的 INS_LOC 约束, 约束其位置 DLL top left corner.
```

示例 8.

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

示例 9.

```
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 10.

```
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.
```

注：

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 PLS），lut6 需要占用 4 个 LUT4 的位置（两个 PLS），lut7 需要占用 4 个 PLS 的位置（一个 GRID），lut8 需要占用 8 个 PLS（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 11. LUT4 单元约束：

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

示例 12. PLS 单元约束：

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

示例 13. PLS 单元约束：

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

示例 14. 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 15. 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 16. 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 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”];”

追加 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”};
```

// 追加 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.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 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 “” “}” “;”

追加 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.7 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.8 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，GW1N6K/GW1N9K/GW1NR9K 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”)

GW1N6K/GW1N9K/GW1NR9K 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.9 Clock Assignment

Clock Assignment 约束是对于设计中特定 wire 到全局时钟线的约束。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 net 的特定 fanout（CLK/CE/SR/LOGIC）的 wire 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。

CLK 信号为连接 CLK 引脚的 wire 信号，CE 信号为连接 CE 引脚的 wire 信号，SR 信号为连接 SET/RESET/CLEAR/PRESET 引脚的 wire 信号，LOGIC

为连接其它逻辑器件输出引脚的 wire 信号。

语法

- “NET_LOC” “”net_name “” global_clocks “=” fanout [quadrant]“;”

约束元素

- net_name

net 的名字

- global_clocks

BUFG[0-7]: 8 个主时钟资源

BUFS[0-7]: 8 个长线资源

- fanout

CLK: fanout 为 CLK 的 wire

CE: fanout 为 CE 的 wire

SR: fanout 为 SET/RESET（同步复位信号）、CLEAR/PRESET（异步复位信号）的 wire

LOGIC: fanout 为以上 fanout 之外的 wire

ALL: 所有 fanout 的 wire

指定多个 fanout, 可使用“|”符号进行分隔。

- quadrant

GW1N 系列: “LEFT”(“L”), “RIGHT”(“R”)

GW1N6K/GW1N9K/GW1NR9K 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.

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

// 约束 NET 对象 net 的 CLK fanout 的 wire 绕到芯片 LEFT 象限的第 0 条主时钟资源（GW1N 家族）上。

示例 2.

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

// 约束 NET 对象 net 的 CLK 和 CE fanout 的 wire 绕到芯片的主时钟资源上。

示例 3.

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

// 约束 NET 对象 net 的 CE fanout 的 wire 绕到芯片的长线资源上。

A.10 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 1.**

```
INS_LOC “clkdiv_name” TS[0];
```

// 将 clkdiv_name 布局到 TOPSIDE[0]处。

附录 B.时序约束语法规范

B.1 时钟约束

B.1.1 create_clock

语法

```
命令: create_clock  
参数: -period <period_value>  
  
[-name <clock_name>]  
  
[-waveform <edge_list>]  
  
<source_objects>  
  
[-add]
```

注!

- []内为可选项。一般来说，使用的可选项越多，约束的越详细。可选项没有被指定时约束更通用并适合更多目标；
- 支持的约束类型：SDC 标准。

-period: 用于指定时钟的周期，参数值应设置为大于 0 的数，周期的单位为 ns。

-name: 用于指定时钟的名称，该参数是时钟的唯一识别标志，因此不能创建重名时钟，否则后创建的时钟会覆盖先创建的时钟。若没有规定该参数，则时钟默认命名为 **source objects** 中第一个元素的名称。

-waveform: 用于指定时钟的上升沿和下降沿的时间，这两个时间是递增的非负数，且二者之差小于一个时钟周期。通常情况下，若规定上升沿先达，则设置上升沿和下降沿时间均小于一个时钟周期，如“{0 5}”表示该时钟为上升沿在 0ns 时刻先达，下降沿在 5ns 时刻到达；若时钟下降沿先达，则可通过设置上升沿时间小于一个时钟周期，下降沿时间大于等于一个时钟

周期即可，如周期设置为 10ns，“-waveform {5 10}”表示该时钟下降沿在 0ns 时刻到达，上升沿在 5ns 时刻到达。

-add: 在同一个源（source object）上添加多个时钟时，应在第二条及以后创建时钟（create_clock）语句中使用-add 参数，否则第二条及以后创建时钟（create_clock）语句会被忽略（时钟不会被创建成功）。

-source_objects: 用于指定创建时钟的时钟源，可以是 PORT、PIN、NET 等元素。如果用户选择的时钟源上已经创建了时钟，用户可使用-add 命令来创建新的时钟，如用户未使用-add 命令，则 STA 工具会忽略该条命令，不会创建新的时钟。如用户在使用 create_clock 命令创建时钟时并未指定时钟源，则 STA 工具将会忽略这条命令，不会正确创建时钟。

应用示例

```
# 创建一个周期 10ns，下降沿先达的时钟
create_clock -name clk -period 10.000 -waveform {5 10} [get_ports
{clk}]
# 创建一个占空比为 40%的时钟
create_clock -name clk -period 10.000 -waveform {6 10} [get_ports
{clk}]
#在同一个端口上添加两个有效时钟
1.create_clock -period 10 -name clk #命令被忽略，不会创建时钟 clk
2.create_clock -period 10 -name clk [get_ports {clk}] #成功创建时钟 clk
3.create_clock -period 10 -name clk1 [get_ports {clk}] #由于缺少-add 参
数，命令 3 被忽略，不会创建时钟 clk1
create_clock -period 20 -name clk1 -add [get_ports {clk}]
#成功创建时钟 clk1
```

B.1.2 create_generated_clock

语法

命令: create_generated_clock

参数: [-name <clock name>]

-source <master pin>

[-edges <edge list>]

[-edge_shift <shift list>]

[-divide_by <factor>]

[-multiply_by<factor>]

[-duty_cycle <percent>]

[-add]

[-invert]

[-master_clock <clock>]

[-phase <phase>]

[-offset <offset>]

<source objects>

注!

- []内为可选项。一般来说，使用的可选项越多，约束的越详细。可选项没有被指定时约束更通用并适合更多目标。
- 支持的约束类型：SDC 标准。

-name: 指定衍生时钟的名称，如果该参数未指定，则用第一个“source object”作为衍生时钟的名称，衍生时钟名称须唯一，如果衍生时钟名称已存在，则先前创建的同名时钟被覆盖。

-source: 指定衍生时钟的来源，如果来源（object）上存在有多个时钟，则需通过“-master_clock”指定具体的主时钟。

-master_clock: 指定衍生时钟所对应的主时钟。

-edges: 指定衍生时钟的时钟沿时间，该参数列表由三个递增正整数组成，表示衍生时钟的第一个上升沿、第一个下降沿、第二个上升沿与主时钟边沿的关系。例如，以主时钟的第一个上升沿为 1，第一个下降沿为 2，第二个上升沿为 3，依次计数，则利用该参数创建一个二分频的衍生时钟的方法是“-edge {1 3 5}”。

-edge_shift: 此参数应与“-edges”参数一起使用，用来在-edges 参数设置的边沿上增加偏移，可取值为任意数，但不能使某边沿超出它的相邻边沿。

注!

“-edge”和“-edge_shift”不能与其他除“-invert”外调整波形的参数同时使用。

-divide_by: 设置衍生时钟相对于主时钟的分频数，该参数应为正整数。

-multiply_by: 设置衍生时钟相对于主时钟的倍频数，该参数应为正整数。

-duty_cycle: 设置衍生时钟的占空比，该参数应为小于 100 的正整数。

-add: 用于添加到同一源上的时钟同时生效。

-invert: 使用该参数可使衍生时钟反相，STA 工具采用平移半个周期的方式实现反相的操作。

-phase: 设置主时钟时钟沿的偏移量（单位：度，正数右移，负数左移）。

-offset: 设置衍生时钟沿偏移量（正数右移，负数左移）。

-source object: 用来指定时钟的入口，可是 PORT、PIN、NET 等元素。

应用举例

#用“-divide_by”在端口 a 上创建一个二分频衍生时钟

```
create_clock -period 10 [get_ports clk]
```

```
create_generated_clock -name genClk -source [get_ports {clk}]
```

```
-divide_by 2 [get_ports {a}]
```

#用“-edges”在端口 a 上创建一个二分频衍生时钟

```
create_generated_clock -name genClk -source [get_ports {clk}] -edges
```

```
{1 3 5} [get_ports {a}]
```

#创建一个占空比为 40%的二倍频衍生时钟

```
create_generated_clock -name genClk0 -source [get_ports {clk}]
```

```
-multiply_by 2 -duty_cycle 40 [get_pins {pll_out}]
```

#创建一个主时钟的二分频反向衍生时钟

```
create_generated_clock -name genClk1 -source [get_ports {clk}]
```

```
-divide_by 2 -invert [get_pins {pll_out}]
```

#创建一个二倍频且相移 90 度的衍生时钟

```
create_generated_clock -name genClk2 -source [get_ports {clk}]
```

```
-multiply_by 2 -phase 90 [get_pins {pll_out}]
```

#创建一个二分频衍生时钟

```
create_generated_clock -name genClk3 -source [get_ports {clk}]
```

```
-edges {2 4 6} [get_pins {pll_out}]
```

#创建一对基于同一源不同主时钟的衍生时钟

```
create_clock -period 10 -name clk [get_ports {clk}]
```

```
create_clock -period 20 -name clk1 -add [get_ports {clk}]
```

```
create_generated_clock -name genClk -source [get_ports {clk}]
```

```
-divide_by 2 -master_clock clk -add [get_pins {pll_out}]
```

```
create_generated_clock -name genClk1 -source [get_ports {clk}]
```

```
-master_clock clk1 -divide_by 2 -add [get_pins {pll_out}]
```

B.1.3 set_clock_latency

语法

命令: set_clock_latency

参数: -source [-rise | -fall]

[-late | -early]

<delay>

[-clock <clock list>]

<object list>

注!

[]内为可选项。一般来说，使用的可选项越多，约束的越详细。可选项没有被指定时约束更通用并适合更多目标。

支持的约束类型：**SDC 标准**

-source: 参数表示设置的是时钟的 **source** 延时。

-rise | -fall: 表示设置的是上升沿还是下降沿的延时，这两个参数不能同时出现在同一条语句中，当这两个参数都没有时，对上升沿和下降沿的延时都做相同的设置，设置为该语句所规定的值。

注！

由于该设置值是时钟源端的数值，需用户来确定此数值。**GOWIN** 软件默认情况下的设置值为 0 ns。

-late | -early: 表示设置的是最大延时还是最小延时，**-late** 用于常规的 **setup** 分析，**-early** 用于常规的 **hold** 分析。

<delay>: 设置时钟的延时值。

注！

STA 提供的默认设置值为 0ns。

-clock: 当创建了多个时钟时，应当使用该参数来确定对哪个时钟设置延时，当没有设置该参数时，对所有的时钟都设置相同的延时。

<source objects>: 用来表示对哪个时钟接入点或者哪个时钟进行延时设置。

应用举例

```
create_clock -period 10 -name clk [get_ports {clk}]
create_clock -period 10 -name clk0 [get_ports {clk}] -add
#为 clk 指定 2ns 时钟延时
set_clock_latency -source 2 [get_clocks {clk}]
#为时钟端口上的 clk0 指定时钟延时
set_clock_latency -source 2 -clock [get_clocks {clk0}] [get_ports {clk}]
```

B.1.4 set_clock_uncertainty

语法

命令：**set_clock_uncertainty**

参数：**[-from <from clock>]**

[-rise_from <rise from clock>]

[-fall_from <-fall from clock>]

[-to <to clock>]

[-rise_to <rise to clock>]

[-fall_to <fall to clock>]

[-setup | -hold]

<uncertainty value>

注！

- []内为可选项。一般来说，使用的可选项越多，约束的越详细。可选项没有被指定时约束更通用并适合更多目标。
- 支持的约束类型：SDC 标准。

-from/-rise_from/-fall_from: 指定该不确定性的源时钟，可通过“-rise_from”和“-fall_from”指定该不确定性的有效源时钟沿。

-to/-rise_to/-fall_to: 指定该不确定性的终点时钟，可通过“-rise_to”和“-fall_to”指定该不确定性的终点有效时钟沿。

-setup/-hold: 指定该不确定性是对建立时间（setup time）还是保持时间（hold time）产生影响，若不指定该参数，则对这二种检查均生效。

<uncertainty value>: 不确定性值，由用户提供。

注！

STA 提供的默认设置值为 0.02ns。

应用举例

#设置从 clk 到 clk 的建立时间不确定性为 0.5

```
set_clock_uncertainty -setup -from clk -to clk 0.5
```

#设置从 clk0 到 clk 的保持时间不确定性为 0.0

```
set_clock_uncertainty -hold -from clk0 -to clk 0.0
```

B.1.5 set_clock_groups

语法

命令：set_clock_groups

参数：[-asynchronous | -Exclusive]

-group <clock name>

-group <clock name>

[-group <clock name>] ...

注！

[]内为可选项。一般来说，使用的可选项越多，约束的越详细。可选项没有被指定时约束更通用并适合更多目标。

支持的约束类型：SDC 标准。

-asynchronous | -Exclusive: 指定时钟间关系为异步或者互斥，二者为互斥关系；

-group: 指定时钟为同一个组；

应用举例

#设置时钟 clk 与时钟 clk0 关系为互斥

```
set_clock_groups -Exclusive -group [get_clocks {clk}] -group
[get_clocks {clk0}]
```

B.2 IO 约束

B.2.1 set_input_delay

语法

命令: set_input_delay

参数: -clock clock_name

[-clock_fall]

[-rise]

[-fall]

[-max]

[-min]

[-add_delay]

[-source_latency_included]

<delay_value>

<port_list>

注!

- []内为可选项。一般来说，使用的可选项越多，约束的越详细。可选项没有被指定时约束更通用并适合更多目标；
- 支持的约束类型：SDC 标准。

-clock: 指定该输入端口与哪个时钟关联；

-clock_fall: 表示该输入端口与时钟的下降沿关联。

注!

若没有指定此参数，则默认为与时钟的上升沿关联。

-rise/-fall: 指定上升沿或下降沿数据的输入延时，若只规定了一个，则另一个自动赋值为相同的值。

-max/-min: 指定数据的最大或最小输入延时，若只规定了一个，则另一个自动赋值为相同的值。

-add_delay: 使得多个此类约束同时生效。

-source_latency_included: 指定该参数，表示外部时钟延时已经包含在输入延时时内。

注！

默认的情况下外部时钟延时不包含在输入延时时内；

<delay_value>: 指定的输入延时值；

注！

STA 工具提供的默认的输入延迟值为 0ns。

<port_list>: 指定受约束的输入端口（PORTS）；

应用举例

```
#设置端口 a 基于 clk 上升沿的输入延时为 0.8ns
set_input_delay -clock clk 0.8 [get_ports {a}]
# 为所有的输入端口设置基于 clk 上升沿的延时为 0.8
set_input_delay -clock clk 0.8 [all_inputs]
#设置端口 a 基于 clk 下降沿的输入延时为 0.8ns
set_input_delay -clock clk -clock_fall 0.8 [get_ports {a}]
#设置端口 a 基于 clk 上升沿的四类延时
set_input_delay -clock clk -max -rise 1.4 [get_ports {a}]
set_input_delay -clock clk -max -fall 1.5 [get_ports {a}]
set_input_delay -clock clk -min -rise 0.7 [get_ports {a}]
set_input_delay -clock clk -min -fall 0.8 [get_ports {a}]
#通过-add_delay 使得基于不同时钟沿的输入延时同时有效
set_input_delay -clock clk0 -min 1.2 [get_ports {a}]
set_input_delay -clock clk0 -max 1.8 [get_ports {a}]
set_input_delay -clock clk0 -clock_fall 1.6 -add_delay [get_ports a]
set_input_delay -clock clk1 -min 2.1 -add_delay [get_ports {a}]
set_input_delay -clock clk1 -max 2.5 -add_delay [get_ports {a}]
```

B.2.2 set_output_delay

语法

命令：set_output_delay
参数：-clock clock_name

[-clock_fall]

[-rise]

[-fall]

```

[-max]
[-min]
[-add_delay]
[-source_latency_included]
<delay_value>
<port_list>

```

注！

- []内为可选项。一般来说，使用的可选项越多，约束的越详细。可选项没有被指定时约束更通用并适合更多目标。
- 支持的约束类型：SDC 标准。

-clock: 参数“-clock”指定与输出延时相关的时钟；

-clock_fall: 指定输出延时的时钟参考沿；

注！

默认参考时钟上升沿。

-rise/fall: 指定上升沿或下降沿数据的输入延时，若只规定了一个，则另一个自动赋值为相同的值；

-max/-min: 指定数据的最大或最小输入延时，若只规定了一个，则另一个自动赋值为相同的值；

-add_delay: 使得多个此类约束同时生效；

-source_latency_included: 指定该参数，表示外部时钟延时已经包含在输入延时效内；

<delay_value>: 指定的输出延时值；

注！

STA 工具提供默认的输出延迟值为 0ns。

<port_list>: 指定受约束的输入端口（PORTS）；

应用举例

#设置端口 b 的输出延时为 0.5ns

```
set_output_delay -clock clk 0.5 [get_ports {b}]
```

#设置所有输出端口的输出延时为 0.5ns

```
set_output_delay -clock clk 0.5 [all_outputs]
```

#设置端口 b 基于时钟下降沿的延时为 0.5ns

```
set_output_delay -clock clk -clock_fall 0.5 [get_ports {b}]
```

#设置端口 b 基于时钟上升沿的延时

```

set_output_delay -clock clk -max -rise 0.3 [get_ports {b}]
set_output_delay -clock clk -max -fall 0.5 [get_ports {b}]
set_output_delay -clock clk -min -rise 0.8 [get_ports {b}]
set_output_delay -clock clk -min -fall 0.7 [get_ports {b}]
#通过参数“-add_delay”使得基于不同时钟沿的输出延时同时有效
set_output_delay -clock clk0 -min 0.5 [get_ports {b}]
set_output_delay -clock clk0 -max 0.6 [get_ports {b}]
set_output_delay -clock clk0 -clock_fall 0.7 -add_delay [get_ports {b}]
set_output_delay -clock clk1 -min 0.8 -add_delay [get_ports {b}]
set_output_delay -clock clk1 -max 0.9 -add_delay [get_ports {b}]

```

B.3 路径约束

B.3.1 set_max_delay | set_min_delay

语法

命令: set_max_delay

参数: [-from <from list>

[-to <to list>

[-through <through_list>

<delay value>

命令: set_min_delay

参数: [-from <from list>

[-to <to list>

[-through <through_list>

<delay value>

注!

- []内为可选项。一般来说，使用的可选项越多，约束的越详细。可选项没有被指定时约束更通用并适合更多目标。
- 支持的约束类型：SDC 标准。

-from 参数用于规定路径的起点，可搜集端口（PORTS）、网络（NETS）、触发器（REGS）、时钟（CLOCKS）和引脚（PINS）等基本单元；

-to 参数用于指定路径的终点，可搜集端口（PORTS）、网络（NETS）、触发器（REGS）、时钟（CLOCKS）和引脚（PINS）等基本单元。

-through:此参数用于指定路径经过的点，它可搜集网络（NETS）和引脚（PINS）等基本单元，当该参数搜集引脚（PINS）时，只能是非时序元件的引脚（PINS），同一条约束中不允许使用多个“-through”参数。

注!

以上三类参数可结合起来使用，也可单独使用，当这三个参数指定的基本单元不在同一条路径上时，STA 工具将忽略此约束，不会对时序计算中产生影响。

应用举例

#设置 clock 驱动的元件到 clock 驱动的元件的时序路径的最大延时为 5ns

```
set_max_delay -from [get_clocks {clk}] -to [get_clocks {clk}] 5
```

```
#设置从端口 a 到触发器 reg0 的最大延时为 2ns
```

```
set_max_delay -from [get_ports {a}] -to [get_registers {reg0}] 2
```

```
#设置从触发器 reg0 到端口 b 的最大延时为 2ns
```

```
set_max_delay -from [get_registers {reg0}] -to [get_ports {b}] 2
```

```
#设置所有受时钟驱动的时序元件的最大延迟为 5ns
```

```
set_max_delay -from [all_clocks] 5 -to [get_ports {out*}]
```

```
#设置从端口 a 到端口 b 的最大延时为 2ns
```

```
set_max_delay -from [get_ports {a}] -to [get_ports {b}] 2
```

```
#设置从触发器 reg0 到 clk 下降沿激励的时序元件的最大延时是 2ns
```

```
set_max_delay -from [get_regs {reg0}] -to [get_clocks {clk}] 2
```

#设置 clock 驱动的元件到 clock 驱动的元件的时序路径的最小延时为 0.5ns

```
set_min_delay -from [get_clocks {clk}] -to [get_clocks {clk}] 0.5
```

```
#设置从端口 a 到触发器 reg0 的最小延时为 0.5ns
```

```
set_min_delay -from [get_ports {a}] -to [get_registers {reg0}] 0.5
```

```
#设置从触发器 reg0 到端口 b 的最小延时为 0.5
```

```
set_min_delay -from [get_registers {reg0}] -to [get_ports {b}] 0.5
```

```
#设置从端口 a 到端口 b 的最小延时是 0.5ns
```

```
set_min_delay -from [get_ports {a}] -to [get_ports {b}] 0.5
```

```
#设置从端口 a 到时钟 clk 的下降沿激励的时序元件的最小延时为 0.5ns
```

```
set_max_delay -from [get_ports {a}] -to [get_clocks {clk}] 0.5
```

B.3.2 set_false_path**语法**

命令: set_false_path

参数: [-from <from list>

[-to <to list>]

[-through <through list>]

[-setup]

[-hold]

注!

- []内为可选项。一般来说，使用的可选项越多，约束的越详细。可选项没有被指定时约束

更通用并适合更多目标。

- 支持的约束类型：SDC 标准。

-setup/-hold: 用于指定当前约束是对建立时间检查还是保持时间检查产生影响，这两个参数互斥，默认对建立时间检查产生影响。

-from: 用于规定路径的起点，可通过端口（`get_ports`）、寄存器（`get_regs`）或时钟（`get_clocks`）来搜集起点。

-to: 用于指定路径的终点，可通过端口（`get_ports`）、寄存器（`get_regs`）或时钟（`get_clocks`）来搜集起点。

-through: 此参数用于规定路径经过的点，可通过引脚（`get_pins`）或者网络（`get_Nets`）来搜集经过的点；该参数列表中可指定多个引脚(PIN)或者多个网络（NET），它们可在同一条路径上，也可在不同的路径上，在同一条约束中不可使用多个“-through”参数。

注！

“-from”、“-to”和“-through”这三类参数可结合起来使用，也可单独使用；当这三个参数规定的点不在同一条路径上时，STA 工具将忽略此条约束，不会对时序分析产生影响。

应用举例

#设置时钟 `clk0` 与时钟 `clk1` 激励的路径不进行时序分析

```
set_false_path -from [get_clocks {clk0}] -to [get_clocks {clk1}]
```

#设置寄存器 `reg0` 到寄存器 `reg1` 的路径不进行时序分析

```
set_false_path -from [get_regs {reg0}] -to [get_regs {reg1}]
```

#设置时钟 `clk` 的上升沿到时钟 `clk1` 下降沿激励的路径不进行时序分析

```
set_false_path -from [get_clocks {clk}] -to [get_clocks {clk1}]
```

#指定端口 `a` 到端口 `b` 的路径不进行时序分析

```
set_false_path -from [get_ports {a}] to [get_ports {b}]
```

B.3.3 set_multicycle_path

语法

命令：set_multicycle_path

参数：[-setup|-hold]

[-start|-end]

[-from <from_list>]

[-to <to list>]

[-through <through_list>]

<path multiplier>

注!

- []内为可选项。一般来说，使用的可选项越多，约束的越详细。可选项没有被指定时约束更通用并适合更多目标。
- 支持的约束类型：SDC 标准。

-start/-end: 指定该约束参考时钟是发起时钟（launch clock），还是锁存时钟（latch clock），参数“-start”指定的参考时钟是发起时钟（launch clock），参数“-end”的参考时钟的是锁存时钟（latch clock）。

注!

STA 工具默认为锁存时钟（latch clock）。

-setup/-hold: 用于指定当前约束是对建立时间检查还是保持时间检查产生影响，这两个参数互斥。

注!

STA 工具默认对建立时间检查产生影响。

-from: 用于规定路径的起点，可通过端口（get_ports）、寄存器（get_regs）或时钟（get_clocks）来搜集起点。

-to: 用于指定路径的终点，可通过端口（get_ports）、寄存器（get_regs）或时钟（get_clocks）来搜集起点。

-through: 此参数用于规定路径经过的点，可通过引脚（get_pins）或者网络（get_Nets）来搜集经过的点；该参数列表中可指定多个引脚(PIN)或者多个网络（NET），它们可在同一条路径上，也可在不同的路径上，在同一条约束中不可使用多个“-through”参数。

注！

“-from”、“-to”和“-through”这三类参数可结合起来使用，也可单独使用；当这三个参数规定的点不在同一条路径上时，STA 工具将忽略此条约束，不会对时序分析产生影响。

应用举例

```
create_clock -name clk -period 10 [get_ports {clk}]
create_generated_clock -name genClk -multiply_by 2 -source
[get_ports {clk}] [get_pins {pll_out}]
#设置多周期路径：参考时钟为 genClk，对建立时间检查产生影响
set_multicycle_path -end -setup -from [get_clocks {clk}] -to [get_clocks
{genClk}] 2
#设置多周期路径：参考时钟为寄存器 reg0 的时钟，对建立时间和保持
时间检查产生影响
set_multicycle_path -start -setup -from [get_regs {reg0}] -to [get_regs
{reg1}] 3
set_multicycle_path -start -hold -from [get_regs {reg0}] -to [get_regs
{reg1}] 1
#设置多周期路径：参考时钟是 clk0，只对源时钟是 clk 上升沿到 clk0 下
降沿激励的路径产生影响
set_multicycle_path -end -setup -from [get_clocks {clk}] -to [get_clocks
{clk0}] 3
```

B.4 工作条件约束

语法

命令：set_operation_conditions

参数：[-grade <c|i>]

[-model <slow|fast>]

[-speed <speed>]

[-setup]

[-hold]

[-max]

[-min]

[-max_min]

注！

- []内为可选项。一般来说，使用的可选项越多，约束的越详细。可选项未被指定时约束更通用并适合更多目标；
- -grade: 指定芯片的温度等级，目前支持商业级（commercial）和工业级（industrial），默认为商业级；
- -model: 指定时序分析的工艺角，默认为 slow；

- -speed: 指定芯片的速度等级;
- -setup: 指定当前工艺角下进行建立时间检查, 与-max 功能一致;
- -hold: 指定当前工艺角下进行保持时间检查, 与-min 功能一致;
- -max: 指定当前工艺角下进行建立时间检查, 与-setup 功能一致;
- -min: 指定当前工艺角下进行保持时间检查, 与-hold 功能一致;
- -max_min: 指定当前工艺角下进行建立、保持时间检查, 与同时指定-setup 和-hold 功能一致。

B.5 时序报告

B.5.1 report_timing

语法

命令: report_timing

参数: [-setup|-hold|-recovery|-removal]

[-max_paths <value>]

[-max_common_paths < value >]

[-rise_from <rise_from_list>]

[-fall_from <fall_from_list>]

[-to <to list>]

[-rise_to <rise_to_list>]

[-fall_to <fall_to_list>]

[-through <through list>]

[-from_clock<from clock>]

[-fall_from_clock <from clock>]

[-rise_from_clock <from clock>]

[-to_clock <to clock>]

[-rise_to_clock <to clock>]

[-fall_to_clock <to clock>]

[-min_logic_level]

[-max_logic_level]

[-mod_ins {mod_ins1 mod_ins2 ...}]

注!

- []内为可选项。一般来说, 使用的可选项越多, 约束的越详细。可选项未被指定时约束更

通用并适合更多目标；

- 支持的约束类型：SDC 标准；
- -setup|-hold|-recovery|-removal：指定时序报告检查的类型；
- -max_paths：指定时序报告的最大路径数；
- -max_common_paths：指定时序报告共享同一结束点路径的最大条数；
- -rise_from/-fall_from：指定时序报告路径的起点；
- -to /-rise_to /-fall_to：指定时序报告路径的终点；
- -through：指定时序报告路径经过的点；
- -from_clock /-fall_from_clock /-rise_from_clock /-to_clock /-rise_to_clock /-fall_to_clock：指明时序报告路径的关联时钟；
- -min_logic_level/-max_logic_level：对报告路径的 logic level 进行限制；
- -mod_ins {mod_ins1 mod_ins2 ...}：可指定多个实例化的 module instance，用空格间隔，若不加该参数则默认报告整个设计的时序。

应用举例

```
#指定对建立时间检查进行报告，报告条数为 100 条
report_timing -setup -max_paths 100 -max_common_paths 5
```

B.5.2 report_high_fanout_nets

语法

命令：report_high_fanout_nets

参数：[-clock_regions]

[-slr]

[-ascending]

[-max_nets <max_net_value>]

[-min_fanout <min_fanout_value>]

[-max_fanout <max_fanout_value>]

注！

- -clock_regions：可选参数，当规定了此参数时，将报告范围限制为连接时序元件时钟输入端的 NET；
- -slr：可选参数，当规定了此参数时，将报告范围限制为连接时序元件的复位/置位输入端（可是同步，也可是异步）的 NET；
- -ascending：可选参数，当规定了此参数时，将指定报告 nets 的扇出值按照降序进行排列，如果不指定该项，默认采用升序进行排列；
- -max_net：可选参数，该参数规定了报告的最大 NET 数量，参数的值应当是非负整数。当未规定该参数的时候，默认的报告最大 NET 数量为 10；
- -min_fanout：可选参数，该参数规定了只报告扇出数不小于该参数值的 NET 的扇出情况，参数的值应当是非负整数；
- -max_fanout：可选参数，该参数规定了只报告扇出数不大于该参数值的 NET 的扇出情况，参数的值应当是非负整数。

应用举例

连接时序元件的复位/置位输入端的 NET，扇出数在 [1,15]的区间，最多报告 10 条：

```
report_high_fanout_Nets -slr -max_nets 10 -min_fanout 1 -max_fanout 15
```

所有 NET 中，报告 NET 的扇出情况，最多报告 10 条：

```
report_high_fanout_Nets -max_nets 10
```

B.5.3 report_route_congestion

语法

命令：report_route_congestion

参数：[-max_grids <max grids value>]

[-min_route_congestion <min route congestion value>]

[-max_route_congestion <max route congestion>]

[-LOC <position>]

注！

- -max_grids: 可选参数，规定了报告的最大 grid 数目，当未规定这个参数时，默认报告 10 个 grid 的拥塞度情况。该参数须是非负整数，否则报告警告信息，该语句被忽略；
- -min_route_congestion: 可选参数，规定了报告 grid 拥塞度的最小值，当未规定这个参数时，默认值为 0。该参数须是[0,1]区间内的浮点数，否则报告警告信息，该语句被忽略；
- -max_route_congestion: 可选参数，规定了报告 grid 拥塞度的最大值，当未规定这个参数时，默认值为 1。该参数须是[0,1]区间内的浮点数，否则报告警告信息，该参数使用默认值 1。该参数的数值须不小于 min_route_congestion 的参数值，否则报告警告信息，该语句被忽略；
- -LOC: 可选参数，规定了报告 grid 的物理位置，可规定单个 grid，如 R1C3，表示报告第 1 行，第 3 列的 grid。也可规定一个范围，如 R[1:3]C3，表示报告第 1 至 3 行第 3 列的 grid；R[1:3]C[1:3]，表示报告 1 至 3 行第 1 至 3 列的 grid；R1C[1:3]，表示报告第 1 行第 1 至 3 列的 grid。

应用举例

报告物理地址为第 1 至 5 行第 1 至 5 列上拥塞度在 0 到 0.5 之间的 grid 的拥塞度情况，只报告拥塞度最高的 5 个。

```
report_route_congestion -max_grids 5 -min_route_congestion 0
-max_route_congestion 0.5 -LOC R[1:5]C[1:5]
```

B.5.4 report_min_pulse_width

语法

命令：report_min_pulse_width

参数：[-nworst <nworst value>]

[-min_pulse_width <min pulse width value>]

```
[-max_pulse_width <max pulse width value>]
```

```
[-detail]
```

```
[get_regs {regIns name}]
```

注！

- []内为可选项。一般来说，使用的可选项越多，约束的越详细。可选项未被指定时约束更通用并适合更多目标。
- -nworst: 规定了报告多少条最差的路径；
- -min_pulse_width: 规定了报告的最小的脉冲宽度，该参数须是大于零的浮点数；
- -max_pulse_width: 规定了报告的最大的脉冲宽度，该参数须是大于零的浮点数；
- -detail: 若规定了这个参数，则进行详细的报告，报告中包含时钟路径；否则进行简略的报告；
- get_regs {regIns name}: 用于指定报告对象，不指定盖选项时，默认对所有寄存器进行脉冲宽度时序分析，可指定一项或多项 reg。

应用举例

详细报告脉冲宽度在 0.1 到 4 之间的最差的路径的 3 条时钟路径的最小脉冲宽度情况：

```
report_min_pulse_width -nworst 3 -min_pulse_width 0.1
-max_pulse_width 4 -detail
```

简略报告脉冲宽度在 0.001 到 4 之间最差的路径的 20 条始终路径的最小脉冲宽度情况：

```
report_min_pulse_width -nworst 20 -min_pulse_width 0.001
-max_pulse_width 4
```

B.5.5 report_max_frequency

语法

命令：report_max_frequency

参数：-mod_ins {mod_ins1 mod_ins2 ...}

注！

-mod_ins {mod_ins1 mod_ins2 ...}: 可指定多个实例化的 module instance，用空格间隔，不管用户是否指定该参数，整个设计的最大频率均会报告；

B.5.6 report_exceptions

语法

命令：report_exceptions

参数：-setup|-hold | -recovery | removal

```
[-max_paths<number>]
```

```
[-max_common_paths< number >]
```

```
[-max_logic_level <number>]
```

[-min_logic_level <number>]
[-rise_from <rise_from_list>]
[-fall_from <fall_from_list>]
[-to <to list>]
[-rise_to <rise_to_list>]
[-fall_to <fall_to_list>]
[-through <through list>]
[-rise_through <rise_through_list>]
[-fall_through <fall_through_list>]
[-from_clock<from clock>]
[-fall_from_clock<from clock>]
[-rise_from_clock<from clock>]
[-to_clock<to clock>]
[-rise_to_clock<to clock>]
[-fall_to_clock<to clock>]

注！

其关键字的名称及含义与 report_timing 的关键字相同。

应用举例

```
set_input_delay -clock sysclk 1 all_inputs
set_output_delay -clock virtual_clock 1 all_outputs
set_max_delay -from [get_clocks {sysclk}] 5 -to [get_ports{out*}]
set_min_delay -from [get_clocks{sysclk}] 3 -to [get_ports {out*}]
set_multicycle_path -end -setup -from [get_clocks {sysclk}] -to
[get_clocks {sysclk}] 2
set_multicycle_path -end -hold -from [get_clocks {sysclk}] -to
[get_clocks {sysclk}] 2
report_exceptions -setup
```