




Gowin 设计时序约束 用户指南

SUG940-1.8.2, 2024-10-25

版权所有 © 2024 广东高云半导体科技股份有限公司

GOWIN高云、、Gowin、云源以及高云均为广东高云半导体科技股份有限公司注册商标，本手册中提到的其他任何商标，其所有权利属其拥有者所有。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，或以禁止反言或其它方式授予任何知识产权许可。除高云半导体在其产品的销售条款和条件中声明的责任之外，高云半导体概不承担任何法律或非法律责任。高云半导体对高云半导体产品的销售和 / 或使用不作任何明示或暗示的担保，包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知识产权的侵权责任等，均不作担保。高云半导体对文档中包含的文字、图片及其它内容的准确性和完整性不承担任何法律或非法律责任，高云半导体保留修改文档中任何内容的权利，恕不另行通知。高云半导体不承诺对这些文档进行适时的更新。

版本信息

日期	版本	说明
2020/06/09	1.0	初始版本。
2020/09/01	1.1	<ul style="list-style-type: none">● 工作条件设定增加车规级-grade a;● 增加基础时钟与衍生时钟联动功能。
2021/06/16	1.2	<ul style="list-style-type: none">● 增加通配符描述;● 更新图例。
2021/11/02	1.3	<ul style="list-style-type: none">● 更新软件版本号;● 更新图例以及相应说明;● 更新附录 A 时序约束语法规范。
2022/05/20	1.3.1	更新一些语言描述。
2022/07/28	1.4	<ul style="list-style-type: none">● Create Clock 增加-Add 选项描述;● 增加了延迟数据模型的温度电压描述;● 增加 tUnc 以及 tSu 含义描述;● 增加通配符跨层级匹配功能描述。
2022/12/16	1.4.1	<ul style="list-style-type: none">● 更新 4.7.2 I/O 延迟约束中选项 Add delay 的描述。
2023/03/31	1.4.2	<ul style="list-style-type: none">● 更新 4.7.2 I/O 延迟约束中 set_input_delay、set_output_delay 描述;● 更新 5.1.4 Total Negative Slack Summary 描述。
2023/04/20	1.4.3	新增图 4-8 新建窗口图 4-9 打开窗口以及相关描述。
2023/05/25	1.5	<ul style="list-style-type: none">● 更新例外约束描述;● 更新虚拟时钟 DEFAULT_CLK 描述。
2023/08/18	1.5.1	<ul style="list-style-type: none">● 更正 set_operation_conditions 为 set_operating_conditions;● 更新报告最大时钟频率约束描述。
2023/11/30	1.6	<ul style="list-style-type: none">● 移除时钟周期与频率转换描述;● 移除 Process 窗口截图及相关描述;● 更新 A.2.2 set_output_delay 描述。
2024/02/02	1.7	增加 derive_clocks 约束。
2024/06/28	1.8	create_clock 增加-add 选项的 DCS 使用场景。
2024/08/09	1.8.1	更新图 5-1 静态时序分析报告和图 5-2 Timing Summaries。
2024/10/25	1.8.2	完善 5.1.2 Clock Summary 中 Clock Name 默认显示规则描述。

目录

目录	i
图目录	iv
表目录	vi
1 关于本手册	1
1.1 手册内容	1
1.2 相关文档	1
1.3 术语、缩略语	1
1.4 技术支持与反馈	2
2 简介	3
3 STA 概述	4
3.1 概述	4
3.2 时序分析基本模型	4
3.3 时序分析术语	5
3.4 时序分析路径	5
3.5 常见的时序检查	6
3.5.1 建立时间 (setup time) 和保持时间 (hold time) 检查	6
3.5.2 恢复时间 (recovery time) 和移除时间 (removal time) 检查	6
3.5.3 最小时钟脉冲 (MPW) 检查	6
4 时序约束编辑器	7
4.1 概述	7
4.2 启动时序约束编辑器	7
4.3 新建、打开及添加约束文件	7
4.3.1 新建约束文件	7
4.3.2 打开约束文件	9
4.3.3 添加约束文件	9
4.4 时序约束编辑器界面	10
4.5 打开时序约束窗口	12
4.6 编辑 SDC 文件	13
4.7 创建时序约束	13

4.7.1 时钟约束.....	14
4.7.2 I/O 延迟约束.....	22
4.7.3 时序例外约束	23
4.7.4 工作条件约束	26
4.7.5 时序报告内容约束	27
4.7.6 其它约束.....	35
4.7.7 保存与导出	36
4.8 时序约束的优先级	36
5 时序报告	37
5.1 Timing Summaries	38
5.1.1 STA Tool Run Summary.....	38
5.1.2 Clock Summary	39
5.1.3 Max Frequency Summary	40
5.1.4 Total Negative Slack Summary.....	40
5.2 Timing Details	40
5.2.1 Path Slacks Table	40
5.2.2 Minimum Pulse Width Table	42
5.2.3 Timing Report By Analysis Type	42
5.2.4 Minimum Pulse Width Report	47
5.2.5 High Fanout Nets Report	48
5.2.6 Route Congestions Report	49
5.2.7 Timing Exceptions Report.....	49
5.2.8 Timing Constraints Report	52
附录 A 时序约束语法规范	54
A.1 时钟约束	54
A.1.1 create_clock	54
A.1.2 create_generated_clock.....	56
A.1.3 set_clock_latency	58
A.1.4 set_clock_uncertainty.....	59
A.1.5 set_clock_groups	60
A.2 I/O 延迟约束	61
A.2.1 set_input_delay	61
A.2.2 set_output_delay	62
A.3 时序路径约束	64
A.3.1 set_max_delay / set_min_delay.....	64
A.3.2 set_false_path	65
A.3.3 set_multicycle_path.....	67

A.4 工作条件约束	68
A.5 时序报告内容约束	69
A.5.1 report_timing	69
A.5.2 report_high_fanout_nets	71
A.5.3 report_route_congestion	71
A.5.4 report_min_pulse_width	72
A.5.5 report_max_frequency	73
A.5.6 report_exceptions	73
A.6 其它约束	74
A.6.1 derive_clocks	74

图目录

图 3-1 时序分析基本模型.....	4
图 3-2 STA 四类时序路径	5
图 4-1 打开新建时序约束文件对话框.....	8
图 4-2 新建时序约束文件.....	8
图 4-3 打开时序约束文件.....	9
图 4-4 添加时序约束文件.....	10
图 4-5 时序约束编辑器界面	10
图 4-6 Netlist Tree 窗口	11
图 4-7 约束编辑窗口	11
图 4-8 新建窗口	12
图 4-9 打开窗口	12
图 4-10 菜单栏打开时序约束窗口.....	12
图 4-11 右键打开时序约束窗口.....	13
图 4-12 编辑 SDC 文件.....	13
图 4-13 创建基础时钟.....	14
图 4-14 选择作用目标.....	15
图 4-15 添加时钟	15
图 4-16 时钟列表	16
图 4-17 时钟列表右键内容.....	16
图 4-18 创建衍生时钟约束.....	17
图 4-19 选择 Create Generated Clock	18
图 4-20 设置时钟延迟.....	19
图 4-21 设置时钟不确定量.....	20
图 4-22 设置时钟组	21
图 4-23 创建 I/O Delay 约束	23
图 4-24 创建 False Path 约束.....	24
图 4-25 创建 Max/Min Delay 约束	25
图 4-26 创建 Multicycle Path 约束.....	26
图 4-27 创建 Operating Conditions 约束	27
图 4-28 创建 Report Timing	28

图 4-29 Report Timing 对话框	29
图 4-30 创建 Report High Fanout Nets	30
图 4-31 Report High Fanout Nets 对话框.....	30
图 4-32 创建 Report Route Congestion.....	31
图 4-33 Report Route Congestion 对话框.....	31
图 4-34 创建 Report Min Pulse Width	32
图 4-35 Report Min Pulse Width 对话框.....	32
图 4-36 创建 Report Max Frequency.....	33
图 4-37 Report Max Frequency 对话框.....	33
图 4-38 创建 Report Exception.....	34
图 4-39 Report Exception 对话框.....	34
图 4-40 创建 Derive Clocks	35
图 4-41 选择 Create Derive Clocks	35
图 4-42 Derive Clocks 列表	36
图 5-1 静态时序分析报告.....	37
图 5-2 Timing Summaries.....	38
图 5-3 Path & Endpoints.....	39
图 5-4 路径余量表.....	41
图 5-5 最小脉冲宽度表	42
图 5-6 路径信息综述.....	43
图 5-7 数据到达路径.....	44
图 5-8 数据请求路径.....	44
图 5-9 路径统计信息.....	45
图 5-10 保持时间分析报告.....	45
图 5-11 恢复时间分析报告	46
图 5-12 移除时间分析报告.....	47
图 5-13 最小脉冲宽度.....	48
图 5-14 高扇出报告	48
图 5-15 绕线拥塞报告.....	49
图 5-16 测试案例	49
图 5-17 Timing Exceptions 约束.....	50
图 5-18 时序例外报告	51
图 5-19 report_exception 语句	52
图 5-20 report_exception 报告	52
图 5-21 时序约束报告.....	53

表目录

表 1-1 术语、缩略语	1
--------------------	---

1 关于本手册

1.1 手册内容

本手册主要描述高云半导体时序约束的相关内容，包含时序约束编辑器（Timing Constraints Editor）的使用、约束语法规范以及静态时序分析报告（以下简称时序报告）说明，旨在帮助用户快速实现时序约束以及如何阅读静态时序分析（Static Timing Analysis, STA）报告。

1.2 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：[SUG918, Gowin 云源软件快速入门指南](#)。

1.3 术语、缩略语

本手册中的相关术语、缩略语及相关释义请参见表 1-1。

表 1-1 术语、缩略语

术语、缩略语	全称	含义
GUI	Graphical User Interface	图形用户界面
MPW	Minimum Pulse Width	最小脉冲宽度
OSC	Oscillator	振荡器
REG	Register	寄存器
SDC	Synopsys Design Constraint	Synopsys 设计约束
STA	Static Timing Analysis	静态时序分析

1.4 技术支持与反馈

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

网址：www.gowinsemi.com.cn

E-mail：support@gowinsemi.com

Tel: +86 755 8262 0391

2 简介

本手册主要包含三部分内容，分别为 **STA** 概述、时序约束编辑器以及时序报告。

STA 概述讲解静态时序分析的基本概念，意在帮助用户了解时序分析的基本原理，旨在快速掌握时序约束编辑器的使用及读懂时序报告。时序约束编辑器是一款能够创建和修改 **SDC** 文件的 **GUI** 工具。在高云半导体云源布局布线流程执行成功后，会根据用户时序约束产生相应的 **STA** 报告。

主要功能

- 支持时钟约束，如基础时钟及衍生时钟约束、源延迟及不确定值约束及组约束；
- 支持数据端口的输入输出延迟约束；
- 支持例外约束，如多循环周期、路径最大最小延迟约束、伪路径约束；
- 支持时序报告内容约束，如 **module** 最大频率约束、**Grid** 拥塞度约束等；
- 提供高效的网表单元查找功能，支持规则表达式匹配；
- 时序约束编辑器采用扁平化设计，界面简约，视觉层次清晰。

主要特点

- 报告内容严格遵循标准的 **W3C XHTML 1.0** 格式规范；
- 报告支持使用外部浏览器工具打开；
- 时序报告支持 **TXT** 文本格式打印；
- 支持导航栏功能，可实现快速的具体内容的定位；
- 完整报告时序约束编辑器生成的所有约束语句；
- 报告的内容表示清晰、层次结构分明，易于阅读。

3 STA 概述

3.1 概述

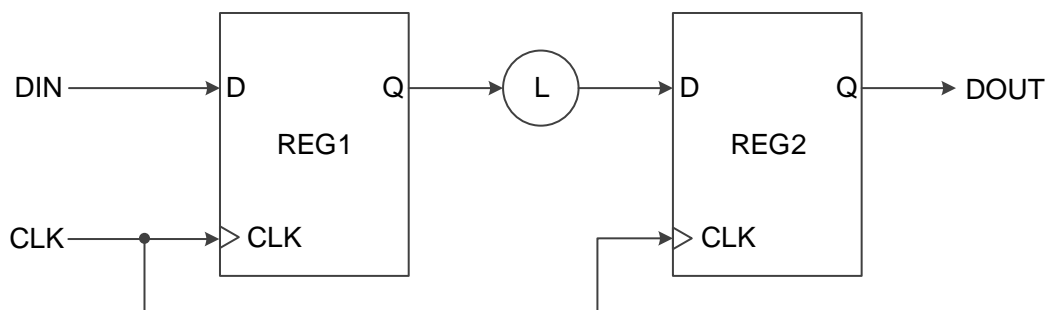
静态时序分析对电路网表中的时序模型进行全面的分析，自动计算电路中时序路径延迟，并判断其是否满足时序要求。云源可自动分析设计中的时序模型电路，同时设计者也可以添加约束约束，计算分析过程由云源自动完成。

在进行静态时序分析前，需对其基本概念进行了解，下面将对分析过程中涉及的基本模型、术语和概念进行简介。

3.2 时序分析基本模型

静态时序分析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，基本模型参考图 3-1 所示，寄存器 REG1 在时钟有效沿将 D 端数据同步到 Q 端，经过逻辑电路到达寄存器 REG2，寄存器 REG2 在时钟有效沿时采集从寄存器 REG1 送出的数据，静态时序分析即是检查 REG2 能否正确采集 REG1 传输的数据。

图 3-1 时序分析基本模型



REG1 的有效时钟沿称为发起沿 (launch edge)，REG2 的有效时钟沿称为锁存沿 (latch edge)。

3.3 时序分析术语

时序模型基本时序单元构成如下：

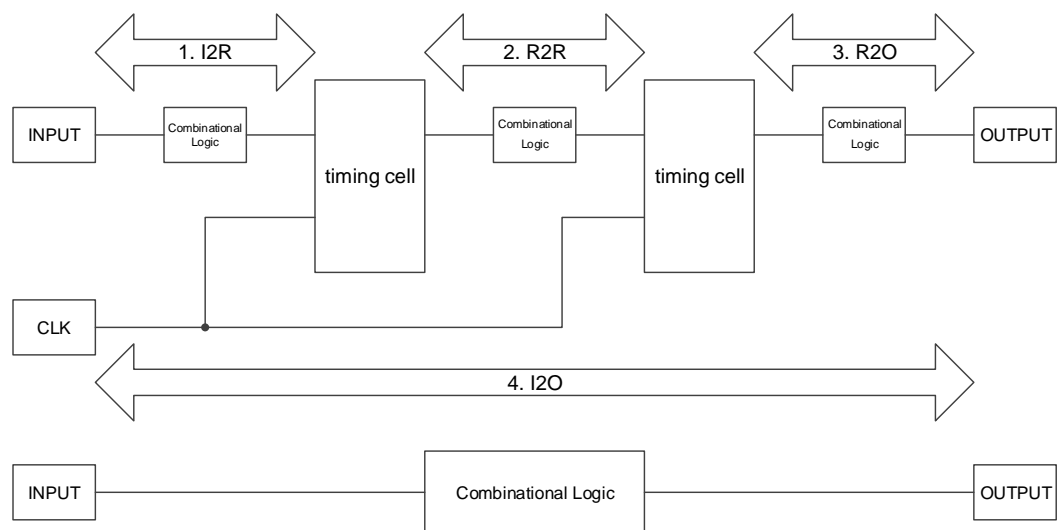
- Cells: 包括 LUT、DFF、MUX 等基本逻辑单元；
- Pins: Cells 的输入输出端口；
- Ports: 顶层模块的输入输出端口；
- Nets: 连线。

3.4 时序分析路径

通常静态时序分析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 3-2 所示：

- I2R: 输入端口到时序元件；
- R2R: 时序元件到时序元件；
- R2O: 时序元件到输出端口；
- I2O: 输入端口到输出端口。

图 3-2 STA 四类时序路径



云源通过该四类路径计算各类路径的数据到达时间（data arrival time）和数据请求时间（data required time）。

数据到达时间是指从数据路径的起点到终点所需的时间，数据请求时间是指时序路径中时钟从起点到终点的时间。在计算数据到达时间时，时钟路径存在时钟偏斜（clock skew），时钟偏斜是指时钟到达不同时序元件时钟端口的时间差。

3.5 常见的时序检查

静态时序分析通常对以下三类进行检查，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。

3.5.1 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需稳定的最短时间，如不满足该时间，则数据不能被时钟有效采集；
- 保持时间：数据在时钟有效沿后需稳定的最短时间，如不满足该时间，则数据不能被时钟有效采集。

3.5.2 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，异步清零、置位、复位信号需保持稳定的最短时间，如不满足该时间，则触发器可能无法进入正常工作状态；
- 移除时间：在时钟有效沿后，异步清零、置位、复位信号需保持稳定的最短时间，如不满足该时间，则触发器可能无法进入正常工作状态。

3.5.3 最小时钟脉冲（MPW）检查

最小时钟脉冲（MPW）：器件内部触发器（如 DFF）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。

4 时序约束编辑器

4.1 概述

时序约束编辑器（Timing Constraints Editor）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，用户可通过该工具提供的图形用户界面添加时序约束。时序约束编辑器简单使用示例可参考 [SUG918, Gowin 云源快速入门指南](#) 时序约束一节。

4.2 启动时序约束编辑器

时序约束编辑器可单独启动使用也可在打开工程综合后使用。

单独使用时单击“Tools > Timing Constraints Editor”启动。打开工程使用时需在云源 Process 窗口中运行“Synthesize”成功后，双击“Process > Timing Constraints Editor”启动。网表文件与工程中的时序约束文件会自动加载到时序约束编辑器中，若工程中不存在时序约束文件则会自动创建。

4.3 新建、打开及添加约束文件

4.3.1 新建约束文件

新建约束文件的步骤如下：

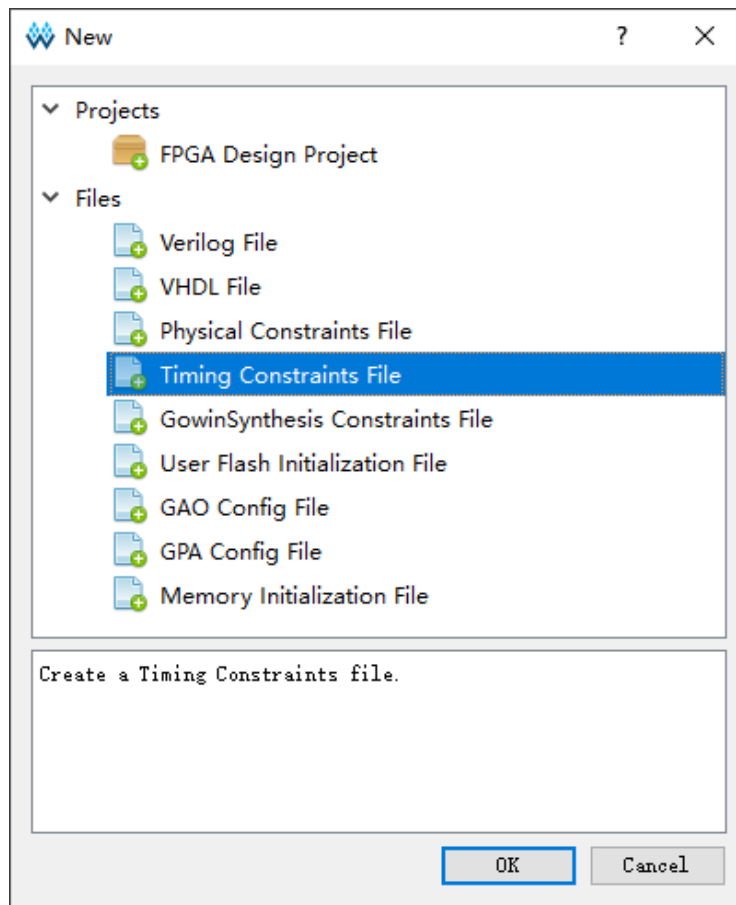
1. 单击“File > New”菜单项，弹出新建文件对话框；
2. 选择“Timing Constraints File”选项，如图 4-1 所示。

注！

亦可通过以下方式打开新建时序约束文件对话框：

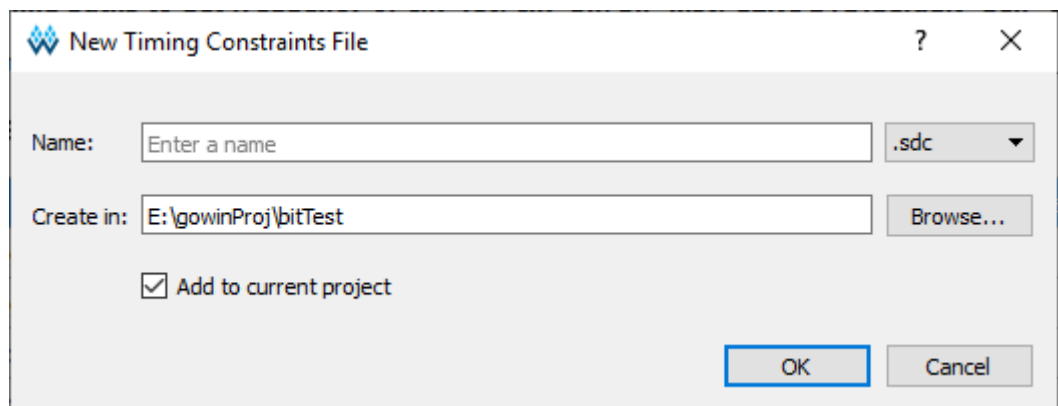
- 单击工具栏上的“New”图标；
- 使用快捷键 Ctrl + N。

图 4-1 打开新建时序约束文件对话框



3. 单击“OK”确认，弹出新建时序约束文件的对话框，如图 4-2 所示；

图 4-2 新建时序约束文件



4. 键入文件名及选定创建目录后单击“OK”，创建时序约束文件且自动将文件加载进工程中。

- **Name:** 新建时序约束文件的名称，文件类型支持.sdc，文件名建议使用字母或下划线开头的具有工程相关意义的标识符；
- **Create in:** 单击“Browse”按钮选择新建约束文件的存放位置，需

是存在的目录，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，默认勾选。

4.3.2 打开约束文件

打开约束文件的步骤如下：

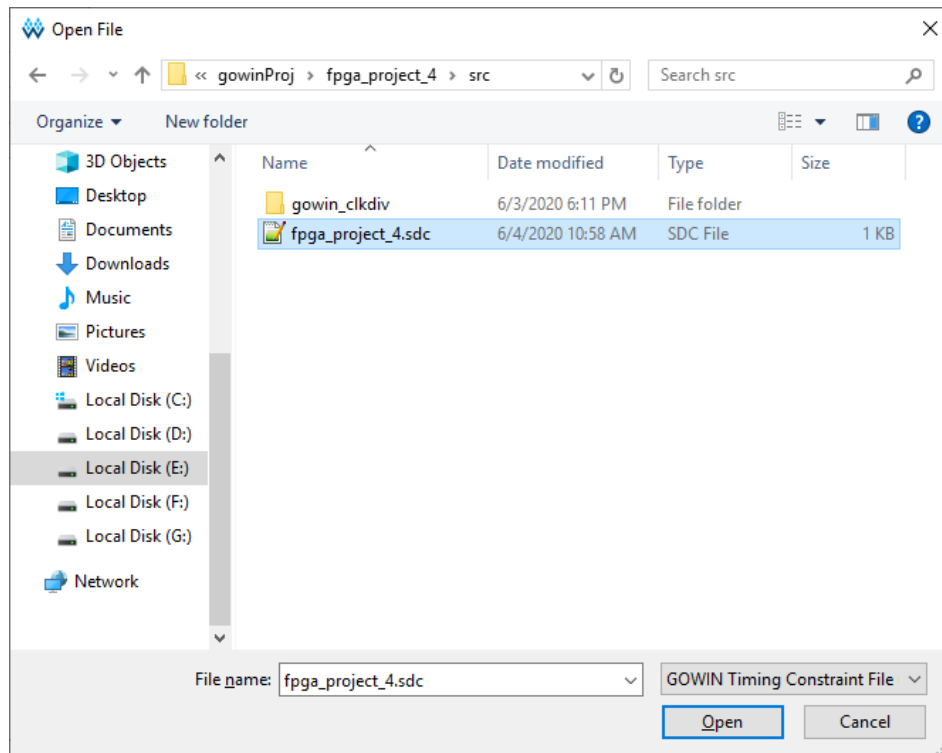
1. 在 IDE 界面中，单击“**File > Open**”菜单项；
2. 打开“**Open File**”对话框，如图 4-3 所示；

注！

亦可通过以下方式打开时序约束文件对话框：

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

图 4-3 打开时序约束文件



3. 选择时序约束文件所在的目录，选中文件单击“**Open**”后打开文件，支持 **sdc** 文件类型。

注！

打开文件操作并不会将文件自动加载进工程中。

4.3.3 添加约束文件

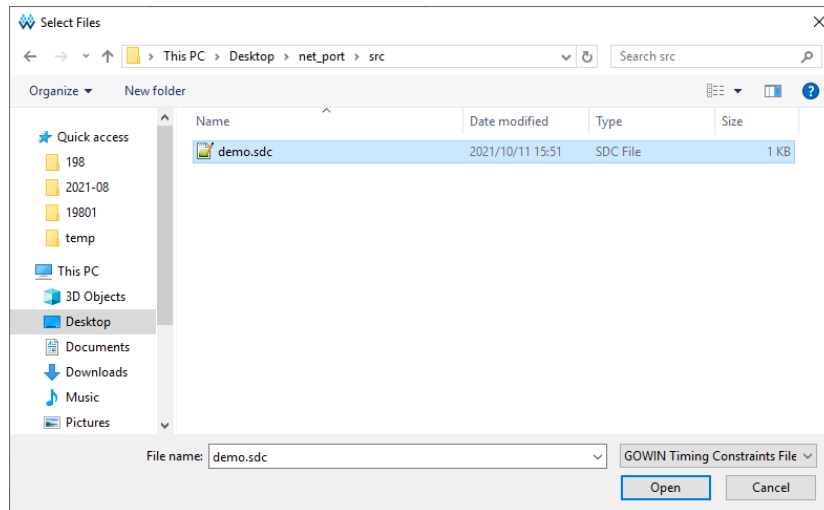
添加时序约束文件步骤如下：

1. 在 IDE 界面 Design 窗口内右击选择“Add Files”；
2. 弹出“Select Files”对话框，文件类型选择“.sdc”，如图 4-4 所示；
3. 选定一个或多个约束文件后单击“Open”即可添加进工程。

注！

添加多个文件时仅一个有效。

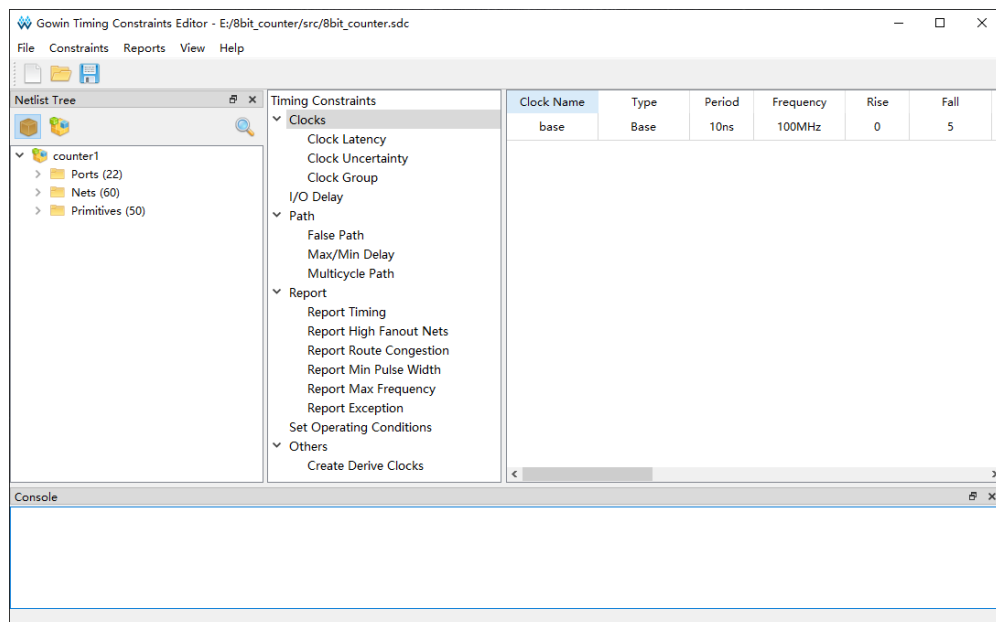
图 4-4 添加时序约束文件



4.4 时序约束编辑器界面

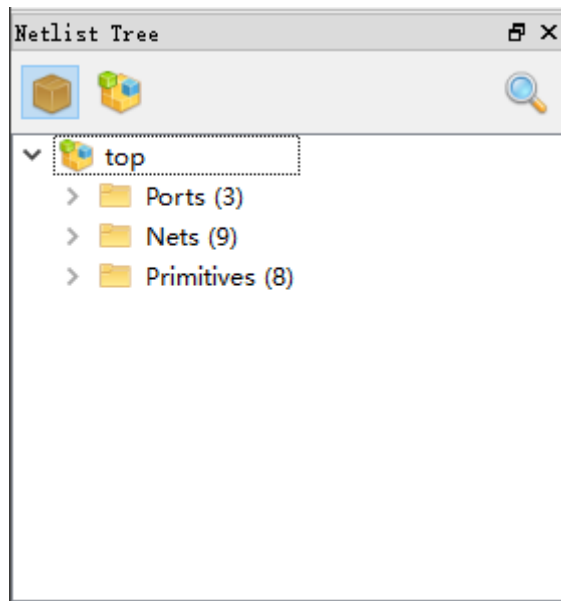
打开约束文件后，时序约束编辑器界面如图 4-5 所示。

图 4-5 时序约束编辑器界面




主窗口左上角为 Netlist Tree 窗口，如图 4-6 所示。

图 4-6 Netlist Tree 窗口



Netlist Tree 窗口中包括当前网表文件中的 Top Module、I/O Ports、Nets 和 Primitives。

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

主窗口中间及右侧区域为约束编辑区，如图 4-7 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 4-7 约束编辑窗口

Timing Constraints						
▼ Clocks						
Clock Name	Type	Period	Frequency	Rise	Fall	
base	Base	10ns	100MHz	0	5	
Clock Latency Clock Uncertainty Clock Group I/O Delay ▼ Path False Path Max/Min Delay Multicycle Path ▼ Report Report Timing Report High Fanout Nets Report Route Congestion Report Min Pulse Width Report Max Frequency Report Exception Set Operating Conditions ▼ Others Create Derive Clocks						




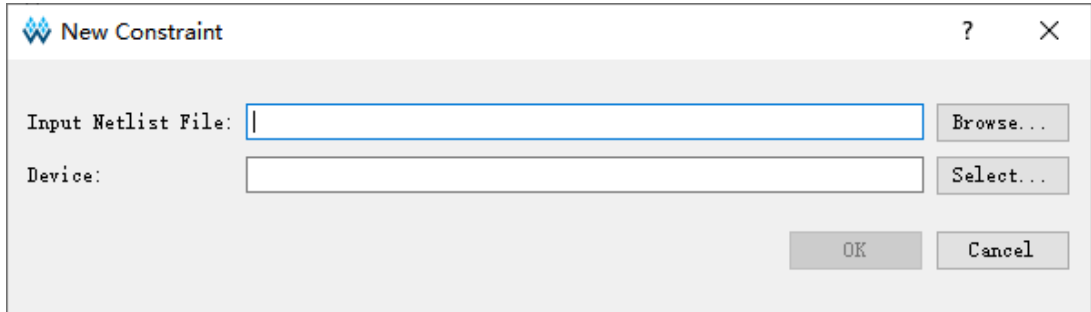
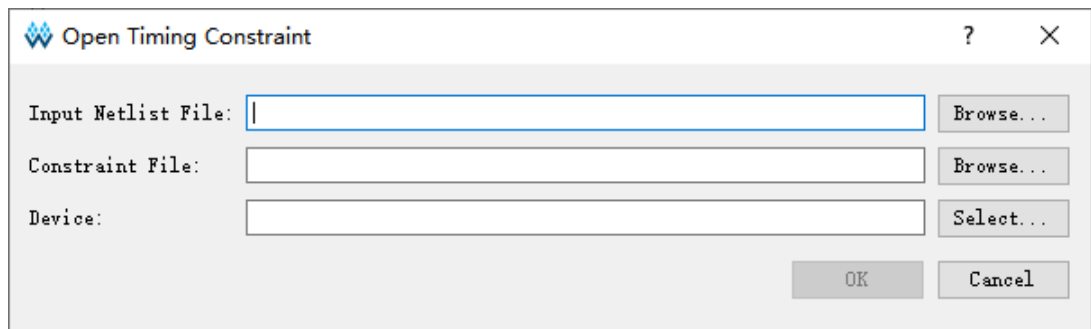
主窗口上部工具栏包含新建“”、打开“”、保存“”。新建窗口包含选定网表文件“Input Netlist File”与选定器件信息“Device”。

图 4-8 新建窗口



打开窗口包含选定网表文件“Input Netlist File”、约束文件“Constraint File”与选定器件信息“Device”。

图 4-9 打开窗口

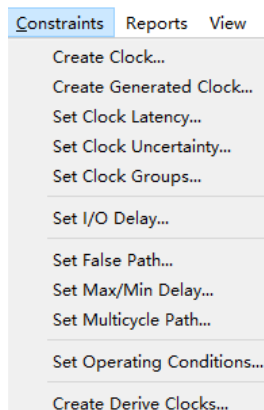


4.5 打开时序约束窗口

提供两种时序约束窗口打开方式。

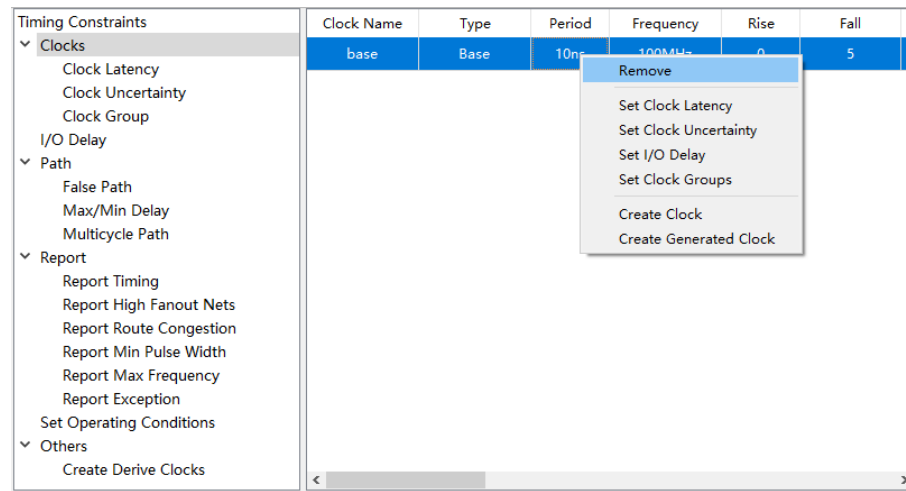
1. 在菜单栏中，单击“Constraints”，在其下拉菜单中，选择时序约束命令，通过选取相应的约束命令打开时序约束窗口，如图 4-10 所示；

图 4-10 菜单栏打开时序约束窗口



- 在时序约束编辑器右侧的表格窗口右击，根据右键菜单列表中不同的选项，选取不同的时序约束命令可打开相应的约束窗口，如图 4-11 所示。

图 4-11 右键打开时序约束窗口



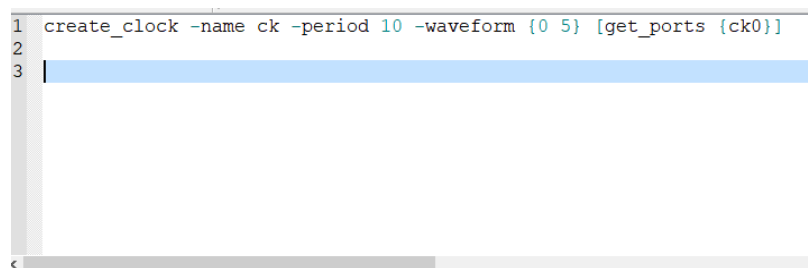
4.6 编辑 SDC 文件

云源支持读取工程的 SDC 文件，并可在文本编辑器中手动修改约束，操作便捷，如图 4-12 所示。

SDC 文件的解析支持通配符功能，目前支持两种通配符“*”和“?”。“*”实现零个或多个字符的匹配，而“?”实现对一个字符的匹配。

SDC 文件支持单行注释和多行注释。单行注释使用“//”或“#”，多行注释使用“/* */”。

图 4-12 编辑 SDC 文件



4.7 创建时序约束

本小节介绍使用图形化时序约束编辑器创建时序约束，创建的时序约束会写入工程中的 SDC 文件中，详细的时序约束语法介绍可参考[附录 A](#)。

4.7.1 时钟约束

Create Clock

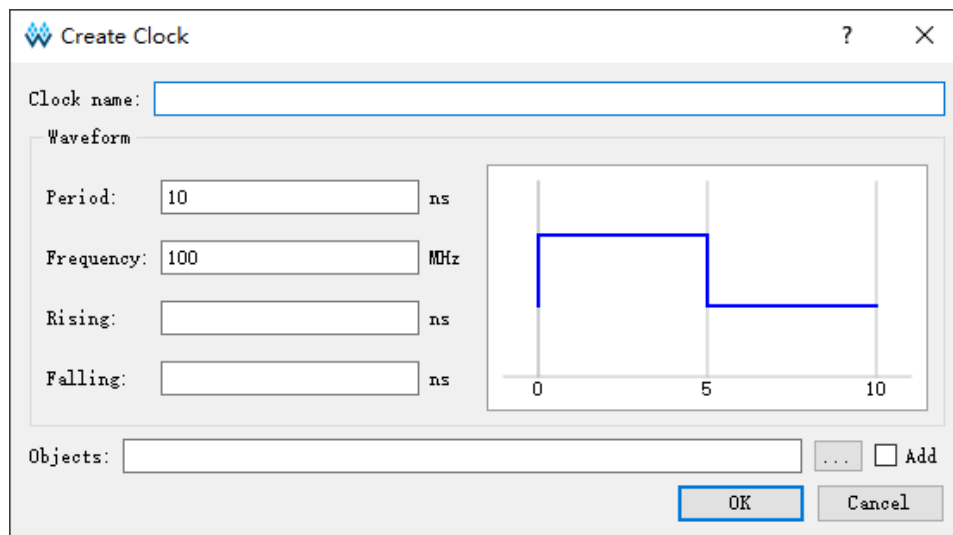
- 可指定时钟名称、周期、频率、上升沿、下降沿，以及该时钟作用的目标等参数；
- 云源支持建立多个时钟，形成多个时钟域，支持跨时钟域分析。

`create_clock` 可为用户设计创建一个基础时钟。

可通过以下两种方式新增 Clock 约束：

1. 通过 Constraints 菜单新增 Clock 约束；
 - a). 选择“Constraints > Create Clock...”，弹出“Create Clock”对话框，如图 4-13 所示；

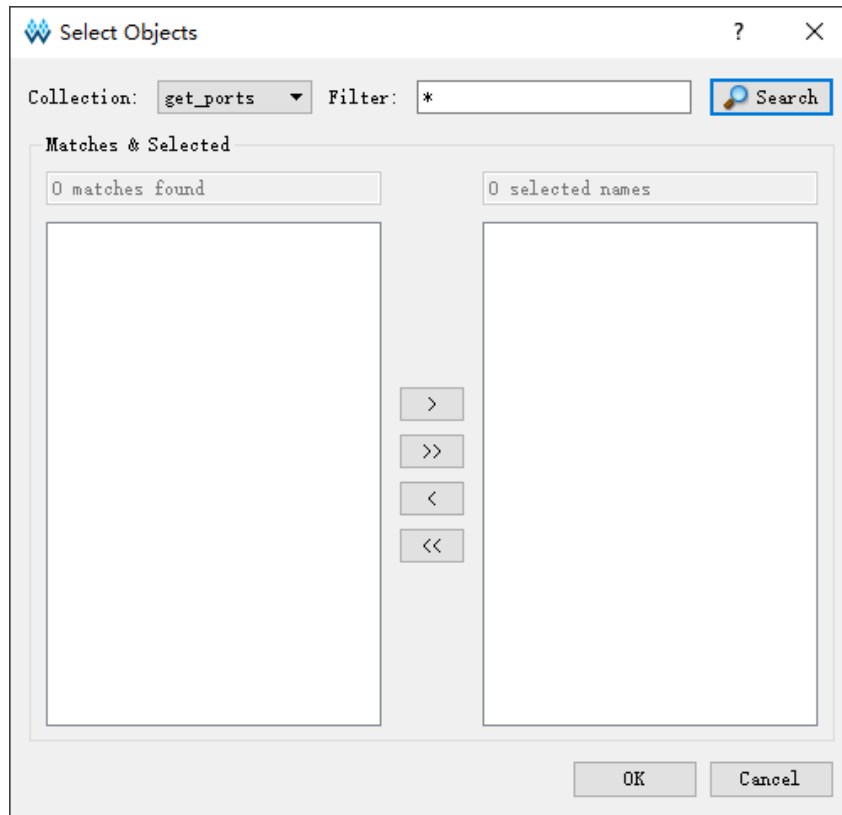
图 4-13 创建基础时钟



- b). 填写 Clock 信息，包括“Clock Name”、“Waveform”、“Objects”；
 - **Clock Name:** 时钟名，支持字母或下划线开头的标识符；
 - **Period:** 周期，默认 10，大于 0 的浮点型，精确到千分位，单位 ns；
 - **Frequency:** 频率，默认 100，大于 0 的浮点型，精确到千分位，单位 MHz；
 - **Rising:** 上升沿时刻，大于 0 的浮点型，精确到千分位，单位 ns；
 - **Falling:** 下降沿时刻，大于 0 的浮点型，精确到千分位，单位 ns；
 - **Objects:** 指定作用目标，通过“...”选择进行内容的自动填充；
 - **Add:** 在同一个源上添加多个时钟时需要勾选。

- c). 单击 Objects 右侧的 “...” 按钮，会弹出 “Select Objects” 对话框，如图 4-14 所示；

图 4-14 选择作用目标

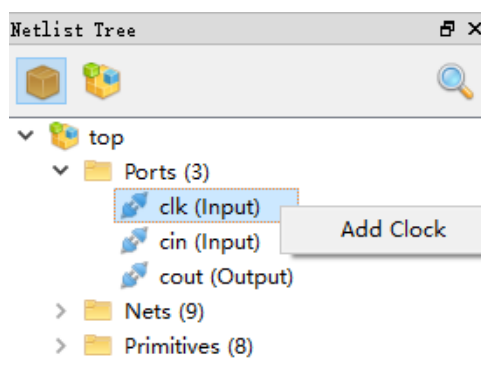


- d). 在图 4-14 中，“Collection”指定搜索的集合类型，“Filter”为过滤器，单击“Search”后左侧为匹配后结果，右侧为已选中列表，“>”按钮将左边列表中的选中项添加到右边列表，“>>”按钮添加左边所有项，“<”按钮移除右边选中项，“<<”按钮移除右边所有项；
- e). 单击“OK”，完成 Objects 的添加。

2. 通过 Netlist Tree 新增 Clock 约束。

- a). 在 Netlist Tree 中，选中 I/O Port 或 Net；
- b). 右击，选择 “Add Clock”，添加一个时钟，如图 4-15 所示。

图 4-15 添加时钟



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 时钟列表

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：

- 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；
- 删除 Clock，在列表中选择该条 Clock，右击，选择“Remove”；
- 右击 Clock，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 时钟列表右键内容

Clock Name	Type	Period	Frequency	Rise	Fall	Divide by	Multiply by	Duty cy
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

注！

- 当约束与 PLL 配置不一致时以 Create Clock 创建的约束为准，PnR 时将提示警告信息；
- Create Clock 不支持创建一个虚拟时钟。

Create Generated Clock

- 用于创建一个基于基础时钟的衍生时钟；
- 通过该约束可基于基础时钟进行分频、倍频、相移和调整占空比等操作，进而完成衍生时钟的创建。

衍生时钟的创建需基于基础时钟，可创建在用户设计中任何一个节点上。实际应用中通常作用于 PLL、CLKDIV 等硬核的输出端口上。如用户在设计中使用了 PLL，创建基础时钟后，即可创建 Objects 为 PLL.CLKOUT、Source 为基础时钟的衍生时钟。创建后的衍生时钟自动与基础时钟进行联动，当基础时钟的属性发生变化时衍生时钟会自动修正以适配基础时钟。

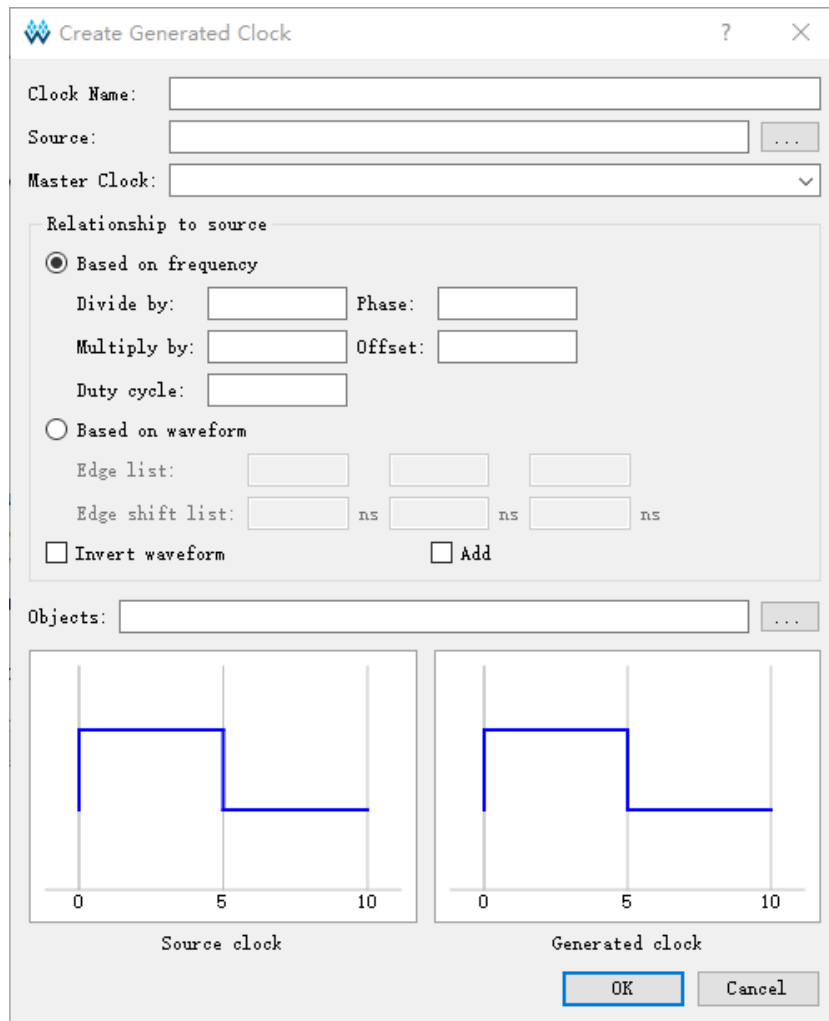
可通过以下两种方式创建衍生时钟：

1. 通过 Constraints 菜单创建；
 - a). 在“Constraints”菜单中，选择“Create Generated Clock”，弹出

“Create Generated Clock”对话框，如图 4-18 所示；

- Clock Name: 时钟名，支持字母或下划线开头的标识符；
- Source: 衍生时钟的源，通过右侧“...”选择；
- Master Clock: 作用在 Source 上的时钟，通过右侧“v”进行选择。

图 4-18 创建衍生时钟约束



- b). 通过单击 Source 右侧的“...”选择时钟的源，与“Source”关联的时钟会自动添加到“Master Clock”列表中，选择“Master Clock”，当“Master Clock”存在多个时钟时，仅支持任选其一；
- c). “Relationship to source”中，基于频率（Base on frequency）可对当前创建的衍生时钟进行分频、倍频、偏置、占空比及相位等调整，基于波形（Base on waveform）使用边沿列表（Edge list）并配合边沿偏移列表（Edge shift list）可实现对衍生时钟进行边沿调整；
 - Divide by: 分频数，正整数；
 - Phase: 相位，浮点型，精确到千分位，负数左移，正数右移；

- **Multiply by:** 倍频数，正整数；
 - **Offset:** 偏置量，浮点型，精确到千分位，负数左移，正数右移；
 - **Duty cycle:** 占空比，浮点型，精确到千分位，数值不大于 100；
 - **Edge list:** 依次递增的正整数；
 - **Edge shift list:** 浮点型，精确到千分位。
- d). “Invert waveform” 实现对时钟的反相，“Add” 可在已有时钟的目标上实现添加，STA 分析时同时有效；
- e). “Objects” 栏用以选定时钟作用的对象，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 无 Clock，则 Master Clock 无选项，需重新选择 Source；
 - 当约束与 PLL 配置不一致时以 Create Generated Clock 创建的约束为准，PnR 时提示警告信息。
2. 通过 Clocks 列表创建 Generated Clock。在 Clocks 列表中，在空白处右击选择“Create Generated Clock”新建 Generated Clock，如图 4-19 所示。

图 4-19 选择 Create Generated Clock

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：

- 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；
- 删除 Generated Clock，在表格编辑区选中该 Clock，右键选择“Remove”。

Set Clock Latency

- 用于设置时钟信号到达器件时钟端口之前的延时，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；
- 时钟延时分为两种：网络（NETWORK）延时和源（SOURCE）延时；
 - 网络（NETWORK）延时是器件内部时钟路径的延时；
 - 源（SOURCE）延时是器件外部时钟路径的延时；

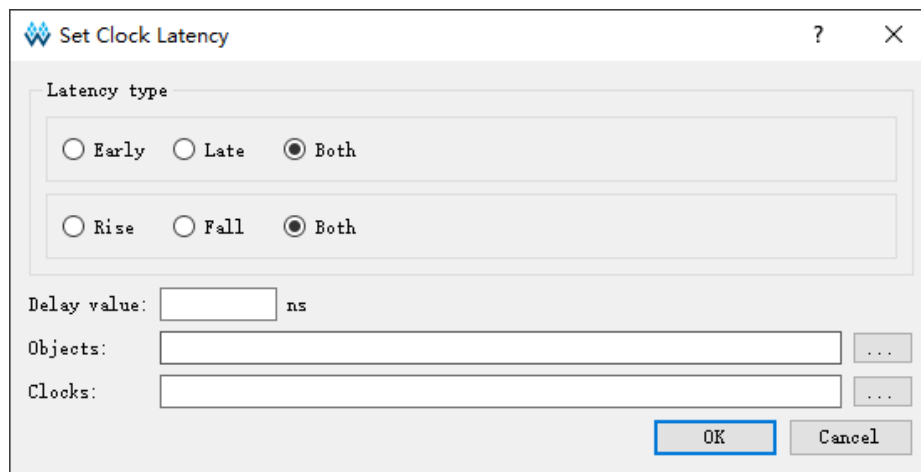
- 云源自动计算时钟的网络（NETWORK）延时，所以用户只需设定源（SOURCE）延时。

时钟信号从时钟源（比如外部晶振）到达器件时钟端口上的延迟称为时钟的源延迟，该延迟值是云源无法自动获知的，默认为 0ns。若用户已知源延迟 2ns，则可配置 Delay Value 为 2ns 的延迟值，云源在进行时序分析时会自动计算添加的 2ns 值，并在时序报告中的 Setup、Hold 报告中的 Tcd 数据体现。

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，填写 Latency 信息，单击“OK”保存约束。
 - Early、Late：表示设置的时钟最早、最晚延迟值；
 - Rise、Fall：分别表示对上升沿下降沿有效，Both 表示对两者均有效；
 - Delay value：时钟延迟值，浮点型，精确到千分位，单位 ns；
 - Objects：通过右侧“...”选择，指明时钟的输入端口或时钟；
 - Clocks：通过右侧“...”选择，指明作用的时钟。

图 4-20 设置时钟延迟



2. 通过 Clocks 列表新建 Clock Latency 约束。

在 Clocks 列表中选中 Clock，右击选择 Set Clock Latency 为该 Clock 设置 Latency 信息，Objects 将自动指定为该时钟目标。

Set Clock Uncertainty

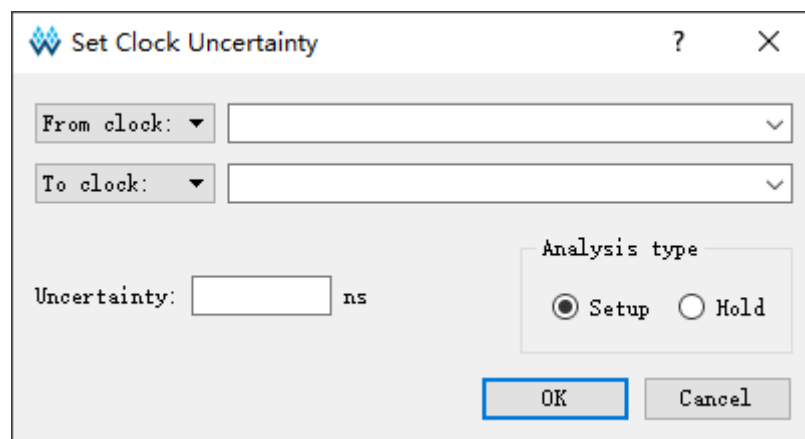
- 设置时钟的不确定量或偏移量，用于时钟传递的分析；
- 可分别对 setup 和 hold 设置不确定量，也可对时钟上升沿和下降沿的传输分别设置不确定量；
- 允许用户将时钟抖动（jitter）、（悲观度 pessimism）通过该约束通知云源，进而影响时序计算。

理想的时钟信号不随时间推移产生超前或滞后的不确定因素，然而时钟的不确定因素通常都是存在的，云源默认会计算不确定因素导致的不确定值。用户也可以根据实际的硬件使用环境设定一个更贴合实际的不确定值供云源计算分析。假设器件工作在强磁环境，用户已知不确定值 0.2ns 时，则 **Uncertainty** 可设定为 0.2ns 。产生的效果请在 **Setup**、**hold** 分析报告中查看 **tUnc**。

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；
 - **From clock**: 指明起始时钟，通过右侧“ \vee ”进行选择；
 - **To clock**: 指明结束时钟，通过右侧“ \vee ”进行选择；
 - **Uncertainty**: 浮点型，精确到千分位，设置时钟的不确定值，单位 ns；
 - **Analysis type**: 指明分析的类型。

图 4-21 设置时钟不确定量



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。

Set Clock Group

- 用于指定不同时钟之间的关系；
- 云源默认提供组内成员之间相关，组与组之间不相关；
- 云源默认认为设计中所有时钟同属一组，且都相关。

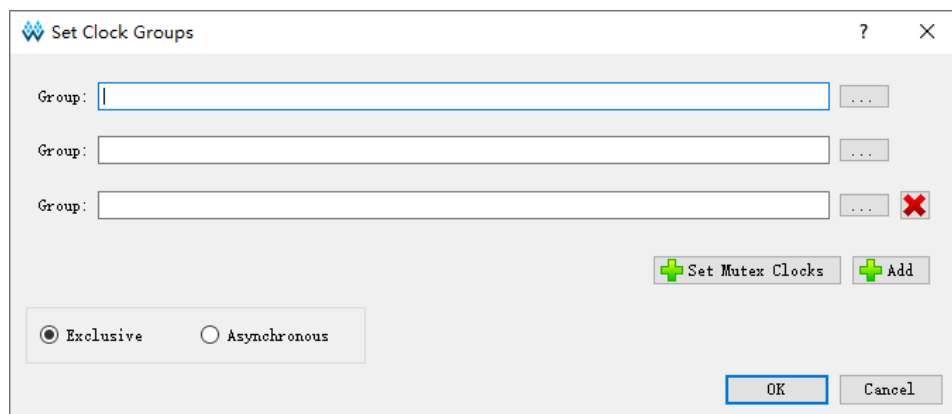
该约束通常用于互斥或异步时钟的约束，如设计中存在两个不同频率的时钟 **CLK1**、**CLK2**，两时钟通过一个多路复选进行二选一驱动时序逻辑，同一时刻两时钟仅有一个有效，表现为互斥，则用户可使用该约束对 **CLK1**、**CLK2** 约束为两个不同的组进行不相关的时序分析。

建议用户使用该约束语句严格指明时钟间的关系，对于异步或互斥的时钟建立时钟组约束。

新建 Clock Group 方式如下：

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：
 - **Group:** 通过右侧“...”进行选择，指明组内的时钟，至少指定一个时钟；
 - **Set Mutex Clocks:** 用以实现一次设置多个不同时钟组；
 - **Add:** 再添加一条 Group 条目；
 - **Exclusive:** 指明时钟为互斥关系，同一时刻时钟不会同时有效，例如 Clock0、Clock1 经过一个 MUX2(两路复选)后输出的时钟 Clock3 作用于一个时序模型，同时刻 Clock3 只能取 Clock0 或 Clock1，则可使用该选项；
 - **Asynchronous:** 指明时钟异步不相关，时钟有不同的时钟源，例如一个时序模型的发送、采样分别由 Clock0、Clock1 驱动，Clock0、Clock1 来自不同的外部端口则可指定该选项。

图 4-22 设置时钟组



2. 单击“...”按钮，为 Group 选择 Clock，如需删除添加的 Group，单击对应条目右侧的“X”按钮；
3. 单击“OK”，保存约束。

注！

选项“Exclusive”与“Asynchronous”实现的作用相同。

4.7.2 I/O 延迟约束

set_input_delay

设定数据输入的延迟值，分析数据到达与时钟到达的时间关系，用户须设定一个合适的输入延迟值，软件会根据给定的延迟值进行余量的分析。

注！

云源产生的时序报告中输入延时类型为“tIn”。

set_output_delay

设定数据输出的延迟值，分析数据输出与时钟输出的时间关系。用户须设定一个合适的输出延迟值，软件会根据给定的延迟值进行余量的分析。

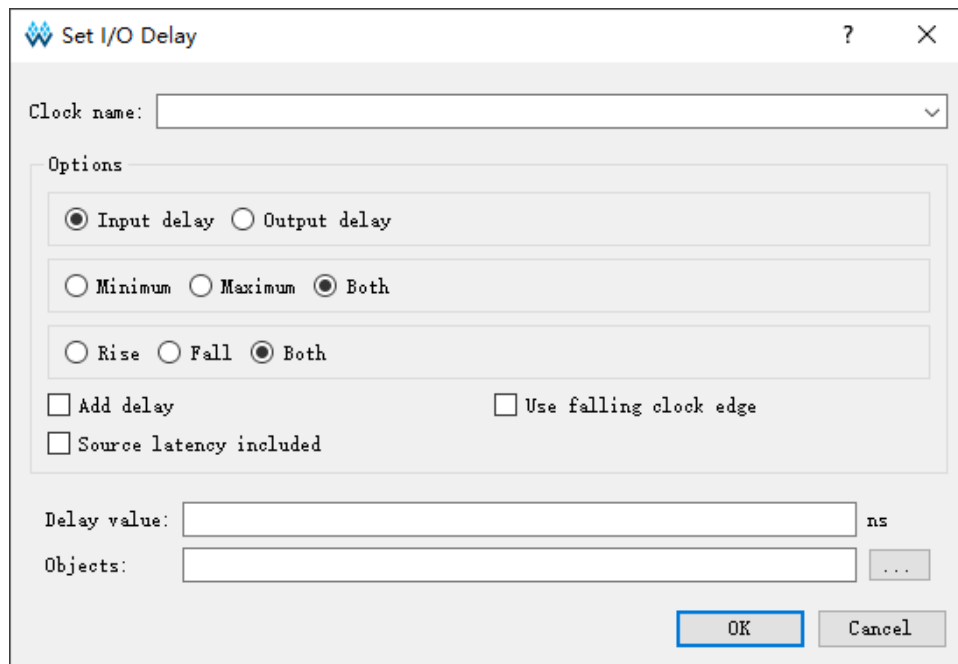
注！

云源的时序报告中，输出延时的类型为“tOut”。

新建 I/O Delay 约束操作如下：

1. 在“Constraints”菜单中，选择“Set I/O Delay”，弹出“Set I/O Delay”对话框，如图 4-23 所示；
 - **Clock name** 指明 I/O 关联时钟的名称，需是存在的时钟，请使用右侧“∨”进行选择；
 - **Options** 用来配置延迟类型、最大最小延迟、作用时钟边沿等；
 - **Input delay**、**Output delay** 指明输入或输出延迟类型，两者互斥；
 - **Minimum**、**Maximum** 指明 I/O 的最小或最大延迟值，**Both** 表示两者延迟值一致；
 - **Rise**、**Fall** 指明对上升沿或下降沿有效，**Both** 表示同时都有效；
 - **Delay value** 设置 I/O 的延迟值，正负浮点型，精确到千分位，当为负数时表示提前到达，当为正数时表示推迟到达，单位 ns；
 - **Objects** 指明输入输出端口，请使用右侧“☐”按钮进行选择；
 - **Add delay** 用以对同一个端口附加一个延迟值，当同一个端口存在多个延迟值时软件会选取最大的进行 **Setup** 分析选取最小的进行 **Hold** 分析，若不指定该选项则在同一端口上的相同约束会被覆盖；
 - **Use falling clock edge** 被勾选后则指明与关联时钟的下降沿相关，默认上升沿相关；
 - **Source Latency include** 被勾选后表示设置的延迟值中已经包含了时钟的延迟值，若不勾选则在计算时云源会算入时钟的延迟值。

图 4-23 创建 I/O Delay 约束



2. 配置填写完成后，单击“OK”保存约束。

4.7.3 时序例外约束

时序例外允许用户修改特定路径的默认静态时序分析规则，时序例外约束命令包含 `set_false_path`、`set_multicycle_path`、`set_max_delay`、`set_min_delay` 四种。

Set False Path

云源默认会分析所有的时序路径，通过该约束语句指定设计中的不需要分析的路径，即非关键路径。建议用户使用该语句指定无需分析的路径。

通常有两类时序路径不需要分析：

- 与设计正常工作不相关的时序电路如测试电路；
- 为跨异步时钟域的路径。假定设计中存在触发器 A 与触发器 B，A 输出数据至 B，A、B 分别被异步时钟 CLK1、CLK2 驱动，则可配置 From 为 CLK1，To 为 CLK2，云源不会分析 CLK1 launch 至 CLK2 latch 的路径。

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 4-24 所示；
 - Analysis type 指明对 Setup 或 Hold 进行检查，Both 表示两者均检查；
 - From 指明路径的起点；

- To 指明路径的终点；
- Through 用于指明路径经过的点或线。

注！

From、To 及 Through 可单独也可相互配合使用。

图 4-24 创建 False Path 约束



2. 单击右侧按钮“...”选择 From、To 以及 Through 对应的 Object，参考图 4-14，单击“OK”保存约束。

Set Max/Min Delay

用以指定一条路径上的最大、最小延迟值。

通常用在端到端延迟分析，如设计中存在输入端口 A 经组合逻辑后输出到端口 B，云源默认并不会分析并报告端口 A 到端口 B 的路径，用户可使用该约束自行指定一个合适的从 A 到 B 的延迟值，云源会自动计算、分析、报告用户指定的路径。当指定最大延迟时会在 Setup 分析报告中进行报告，当指定最小延迟时则在 HOLD 分析报告中进行报告。

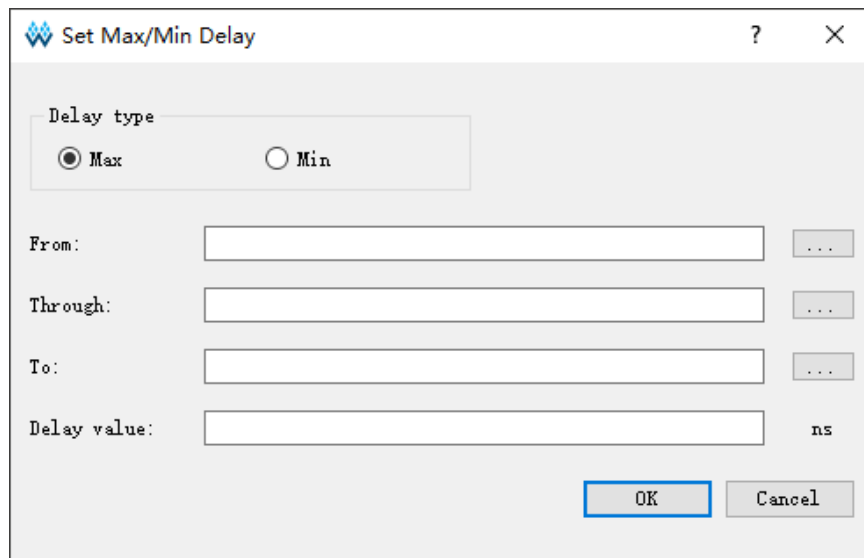
新建 Max/Min Delay 约束操作如下：

1. 选择“Constraints > Set Max/Min Delay”，弹出“Set Max/Min Delay”对话框，如图 4-25 所示；
 - From 参数用于规定路径的起点，请使用右侧“...”进行选择；
 - To 参数用于指定路径的终点，请使用右侧“...”进行选择；
 - Through 参数用于指定路径经过的点或线，请使用右侧“...”进行选择；
 - Delay value 由用户指定的延迟值，浮点型，精确到千分位，单位 ns。

注！

From、To 及 Through 可单独也可相互配合使用。

图 4-25 创建 Max/Min Delay 约束



2. Delay Type 选择 Delay 的类型 (Max 或 Min), From 和 To 选择对应的 Object。填写完成 Delay 信息后, 单击“OK”完成创建。

Set MultiCycle Path

默认情况下, 云源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。

如设计中时序路径 Path_A 上的数据需要 2 个周期才能稳定, 而云源默认为单周期分析与实际不符, 则用户需设置 Value 为 2, 云源可根据设定的值进行分析。产生的报告请在 Setup、Hold 分析报告中进行查看。

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, 则云源默认为 -setup。如已设置 -setup 值, 则 hold 值不会受其影响;
- 云源默认提供自动修复 hold 的功能。如用户指定 hold 值, 云源会优先考虑用户设置的约束。

新建 Multicycle Path 约束操作如下:

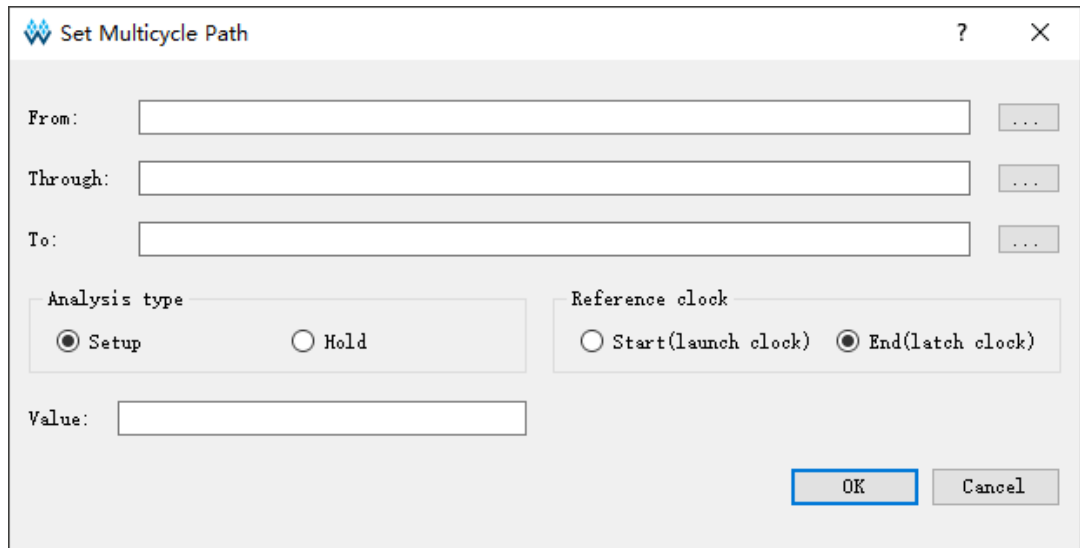
1. 选择“Constraints > Set Multicycle Path”, 弹出“Set Multicycle Path”对话框, 如图 4-26 所示;
 - Reference clock 指明参考时钟是发起时钟, 还是锁存时钟;
 - Analysis type 指明约束对 Setup 或 hold 检查;
 - From 用于指明路径的起点, 请使用右侧“...”进行选择;
 - Through 参数用于指定路径经过的点, 请使用右侧“...”进行选择;

- To 用于指明路径的终点，请使用右侧“...”进行选择；
- Value 指定多循环周期个数，正负整数，当为负数时表示提前，当为正数时表示推迟。

注！

From、To 及 Through 可单独也可相互配合使用。

图 4-26 创建 Multicycle Path 约束



2. 填写对话框中相关信息，单击“OK”保存约束。

4.7.4 工作条件约束

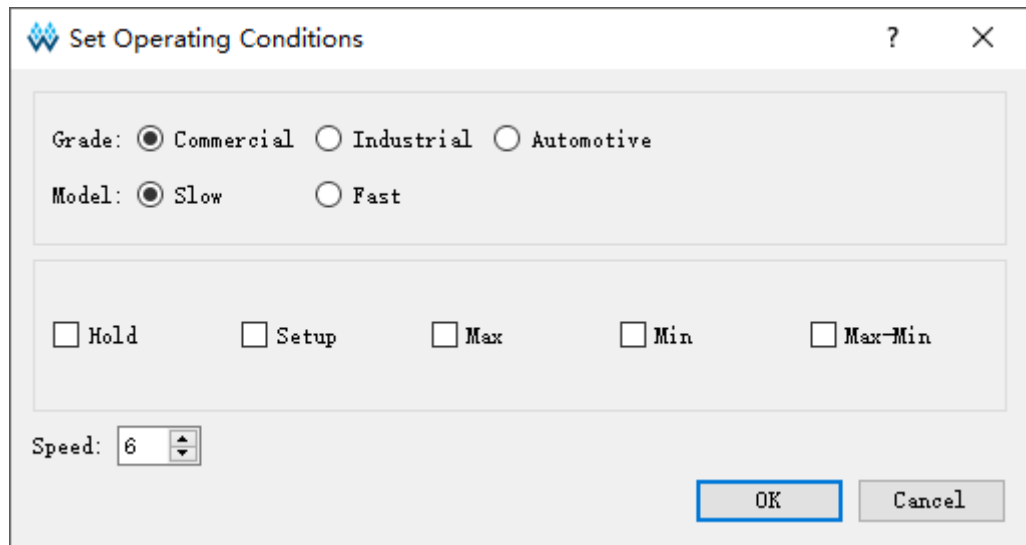
约束时序分析时使用的延迟模型，可指定速度等级、模型类型等。默认云源在进行 Setup 分析时使用 Slow Model（慢速延迟模型），Hold 分析时使用 Fast Model（快速延迟模型）。

用户也可自定义云源分析的时序模型，如在较热且电源不稳定的情况下可指定慢速延迟模型使时序的分析更加贴合实际应用。完成后可在 STA Tool Run Summary 中查看使用的延迟模型。

选择“Constraints > Set Operating Conditions”，弹出“Set Operating Conditions”对话框，新建 Operating Conditions 约束，如图 4-27 所示。

- Grade 分为商业级、工业级以及车规级；
- Model 分为慢速、快速；
- Hold、Setup 指明对保持时间或建立时间有效；
- Max 功能与 Setup 一致，Min 功能与 Hold 一致；
- Max-Min 功能等同于同时选定 Max、Min。

图 4-27 创建 Operating Conditions 约束



注！

- 当设置的 Grade Speed 与芯片型号不匹配时以实际约束为准；
- 若实际约束的 Grade Speed 不支持当前工程则云源将提示警告信息；
- 若 Grade-Speed 仅设置了 Setup，则 Hold 按照 Setup 设置的 Grade-Speed 进行分析；
- 若 Grade-Speed 仅设置了 Hold，则 Setup 按照 Hold 设置的 Grade-Speed 进行分析；
- 工程样片（ES）默认使用最慢速度等级进行时序分析，建议用户自行设定相应值。

4.7.5 时序报告内容约束

Report Timing

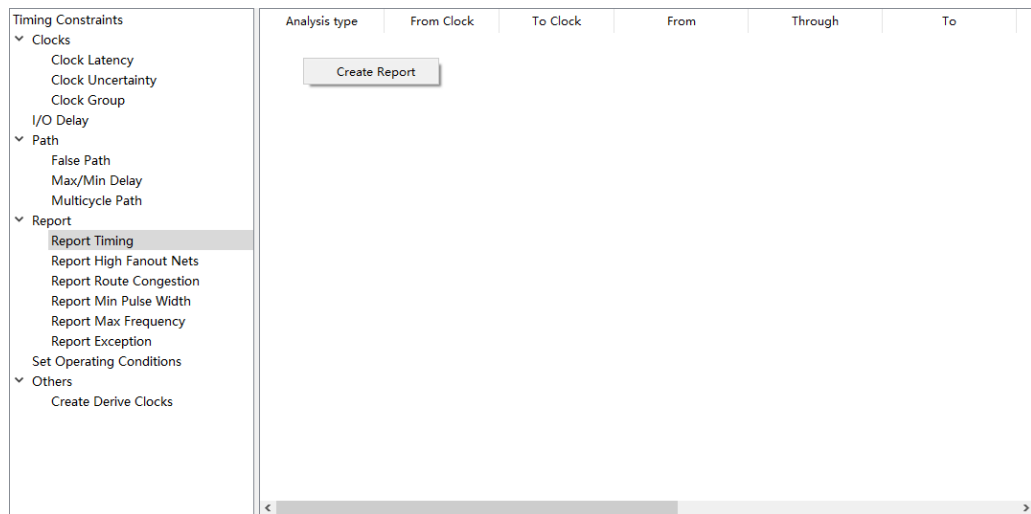
根据设置的参数，输出相应的报告内容，可实现更具体的时序报告与分析。

例如，默认云源报告 25 条 Setup 分析路径，当用户需要查看 35 条最差的 Setup 路径分析信息时可直接填入图 4-29 中的“Max Paths”值 35 即可。产生的报告在 Setup、Hold 分析报告中进行查看。

操作步骤如下：

1. 在主窗口，选择“Timing Constraints > Report Timing”，空白处右击，出现“Create Report”，如图 4-28 所示；

图 4-28 创建 Report Timing



2. 选择“Create Report”弹出如图 4-29 所示的对话框；

- **Path** 指定时序报告的最大路径数（Max Paths）、最大共同路径（Max Common Paths）、最大最小逻辑级数（Max/Min Logic Level）均为正整数；
- **Clocks** 指明时序报告路径的关联时钟，From/To Clock 分别指明发送时钟、采样时钟，使用右侧“∨”按钮进行选择；
- **Objects** 指明分析的起始和结束目标，请使用右侧“...”按钮进行选择；
- **Analysis Type** 指定时序报告检查的类型分别为建立时间（Setup）、保持时间（Hold）、恢复时间（Recovery）及移除时间（Removal）；
- **Module Instance** 指明报告的 Module 的实例化名称，请使用右侧“...”按钮进行选择。

图 4-29 Report Timing 对话框

The image shows the 'Report Timing' dialog box with the following fields and options:

- Clocks:** 'From clock:' and 'To clock:' dropdown menus.
- Objects:** 'From:', 'Through:', and 'To:' dropdown menus, each followed by a text input field and an ellipsis button.
- Analysis Type:** Radio buttons for 'Setup' (selected), 'Hold', 'Recovery', and 'Removal'.
- Path:** 'Max Paths:', 'Max Common Paths:', 'Min Logic Level:', and 'Max Logic Level:' text input fields.
- Module Instance:** A text input field with an ellipsis button.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom right.

3. 填写对话框中相关信息，单击“OK”，保存时序报告设置。

Report High Fanout Nets

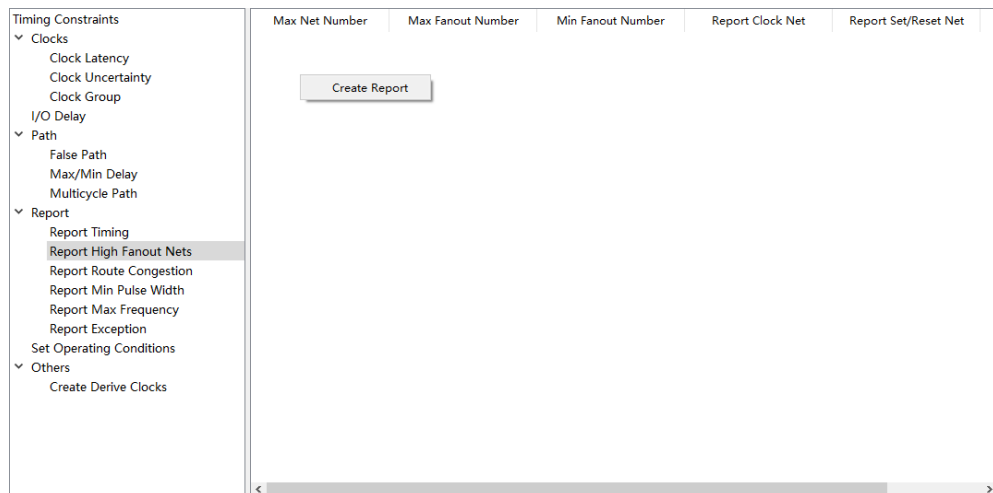
报告 Net 的扇出数目，默认报告 10 条最大的。

如用户需查看扇出在 5 到 7 之间的 Net 时可指定 Min Fanout 为 5, Max Fanout 为 7，产生的报告可在 High Fanout Nets Report 中进行查看。

操作步骤如下：

1. 在主界面中，选择“Timing Constraints > Report High Fanout Nets”；
2. 在右侧空白处右击弹出“Create Report”，如图 4-30 所示；

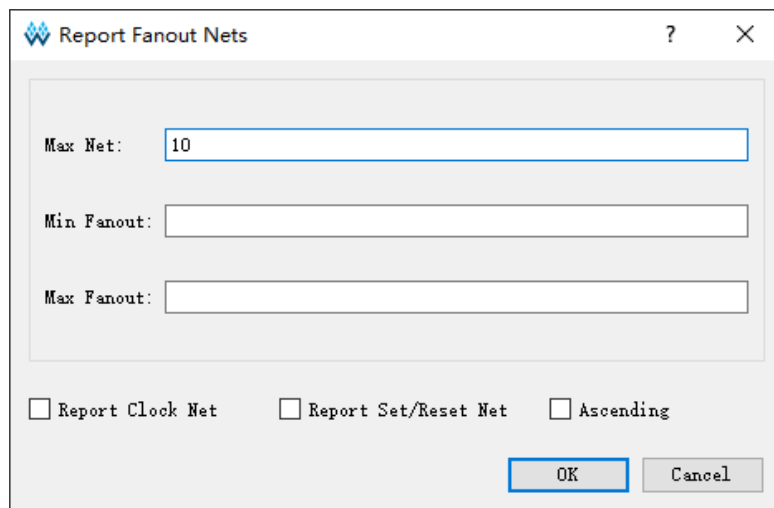
图 4-30 创建 Report High Fanout Nets



3. 选择“Create Report”，弹出如图 4-31 所示的对话框：

- Max Net 指明最大报告的个数，正整数；
- Min、Max Fanout 分别指明报告扇出的下限、上限，正整数；
- Report Clock Net 报告连接时序元件时钟输入端的 Net；
- Report Set/Reset Net 报告连接时序元件复位输入端 Net；
- Ascending 指 Net 的排列顺序，默认采用升序。

图 4-31 Report High Fanout Nets 对话框



4. 填写对话框中相关信息，单击“OK”，保存时序报告设置。

Report Route Congestion

报告拥塞度情况，默认报告 10 个最差的 Grid。

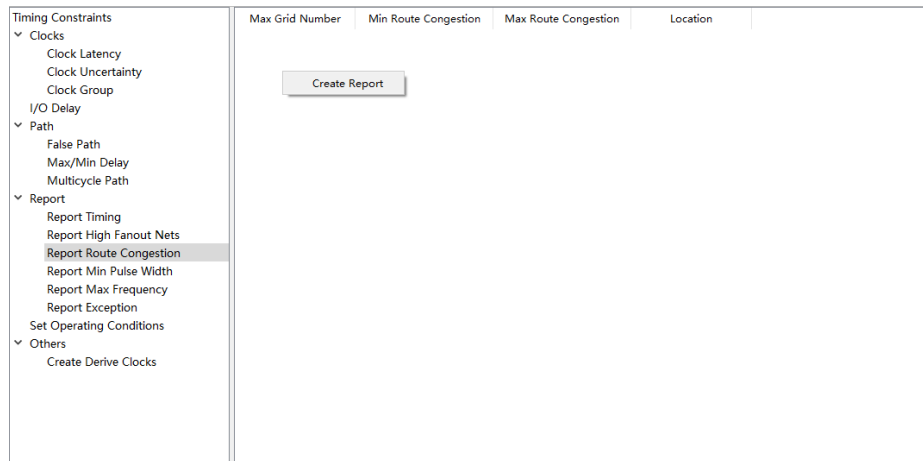
通常用在一个特定 Grid 上的绕线拥塞度报告，如用户需要报告 Grid R4C4 上的拥塞度，指定 Grid Location 为 R4C4 即可，产生的报告请在 Route

Congestions Report 中进行查看。

相关操作步骤如下：

1. 在主界面中，选择“Timing Constraints > Report Route Congestion”；
2. 在右侧空白处右击，出现“Create Report”，如图 4-32 所示；

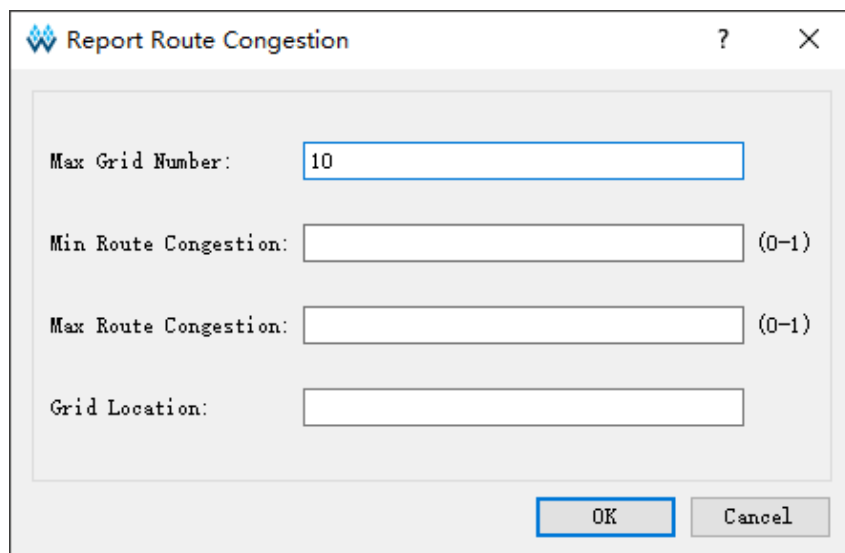
图 4-32 创建 Report Route Congestion



3. 选择“Create Report”，弹出如图 4-33 所示的对话框；

- Max Grid Number 指明报告的个数；
- Min、Max Route Congestion 分别指明绕线拥塞度的下限、上限，浮点型，精确到千分位；
- Grid Location 指定报告的 Grid，如 R4C4。

图 4-33 Report Route Congestion 对话框



4. 填写对话框中相关信息，单击“OK”，保存时序报告设置。

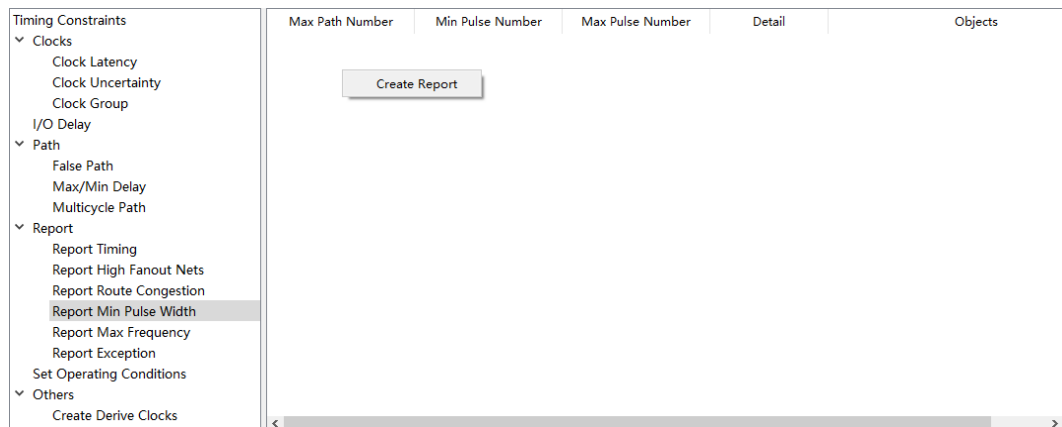
Report Min Pulse Width

报告最小脉冲宽度，默认报告 10 条。用户可使用该约束语句报告特定范围内的脉冲宽度或特定目标上的脉冲宽度。如设计中存在触发器实例化名 Reg11_Z，则用户可指定 Objects 为 Reg11_Z 进行报告，产生的报告请在 Minimum Pulse Width Report 查看。

操作参考如下：

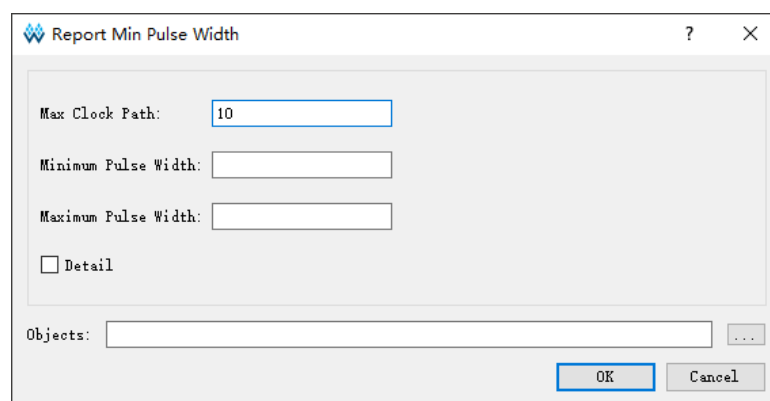
1. 在主界面中，选择“Timing Constraints > Report Min Pulse Width”；
2. 在右侧空白处右击，出现“Create Report”，如图 4-34 所示；

图 4-34 创建 Report Min Pulse Width



3. 选择“Create Report”出现如图 4-35 所示的对话框；
 - Max Clock Path 指明最大报告数,正整数；
 - Minimum、Maximum Pulse Width 指明报告的实际脉冲宽度的下限、上限，浮点型，精确到千分位；
 - Detail 指明是否报告详细路径；
 - Objects 指明需要报告的时序元件，仅支持触发器如 DFF 等，请使用右侧“...”按钮进行选择。

图 4-35 Report Min Pulse Width 对话框



4. 填写对话框中相关信息，单击“OK”，保存时序报告设置。

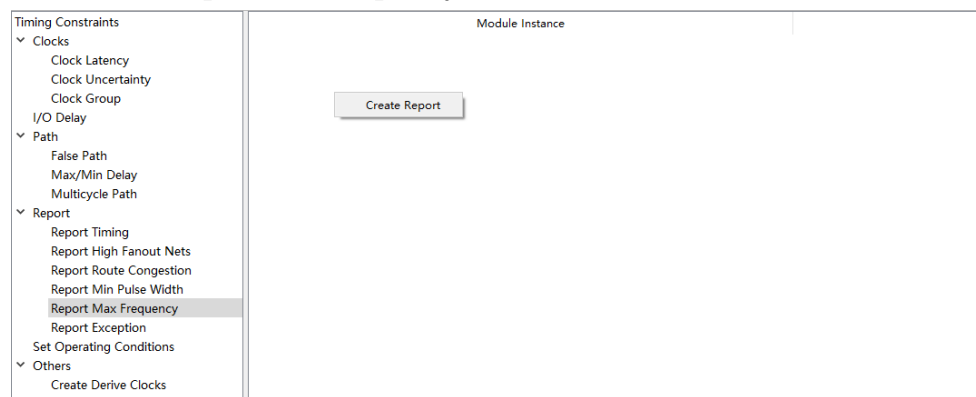
Report Max Frequency

报告最大工作频率，默认云源仅报告设计 Top 层的时钟最大频率。用户可指定报告一个特定的 module 的最大工作时钟频率，该 module 的最大工作时钟频率时序关键路径不限于本 module 内部但与本 module 同步相关。

操作步骤如下：

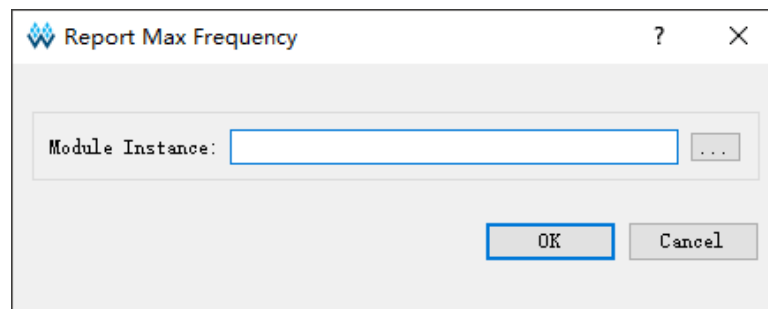
1. 在主界面中选择“Timing Constraints > Report > Report Max Frequency”；
2. 在右侧空白处右击，出现“Create Report”，如图 4-36 所示；

图 4-36 创建 Report Max Frequency



3. 选择“Create Report”，弹出如图 4-37 所示的对话框。“Module Instance”中为模块的实例化名称，使用右侧“...”按钮进行选择；

图 4-37 Report Max Frequency 对话框



4. 单击“OK”，保存时序报告设置。

Report Exception

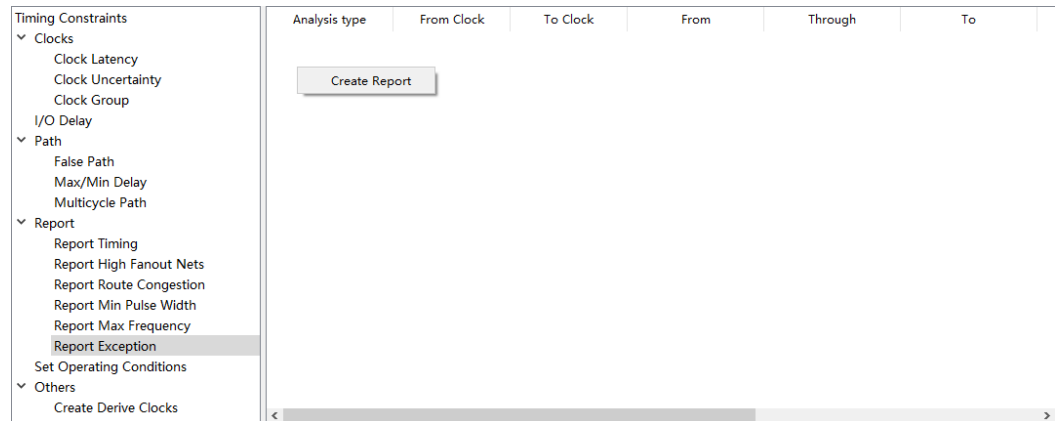
对例外约束语句作用的时序路径进行进一步约束，以报告用户关心的时序路径。

相关操作步骤如下：

1. 在主界面中选择“Timing Constraints > Report > Report Exception”；

2. 在右侧空白处右击，出现“Create Report”，如图 4-38 所示；

图 4-38 创建 Report Exception

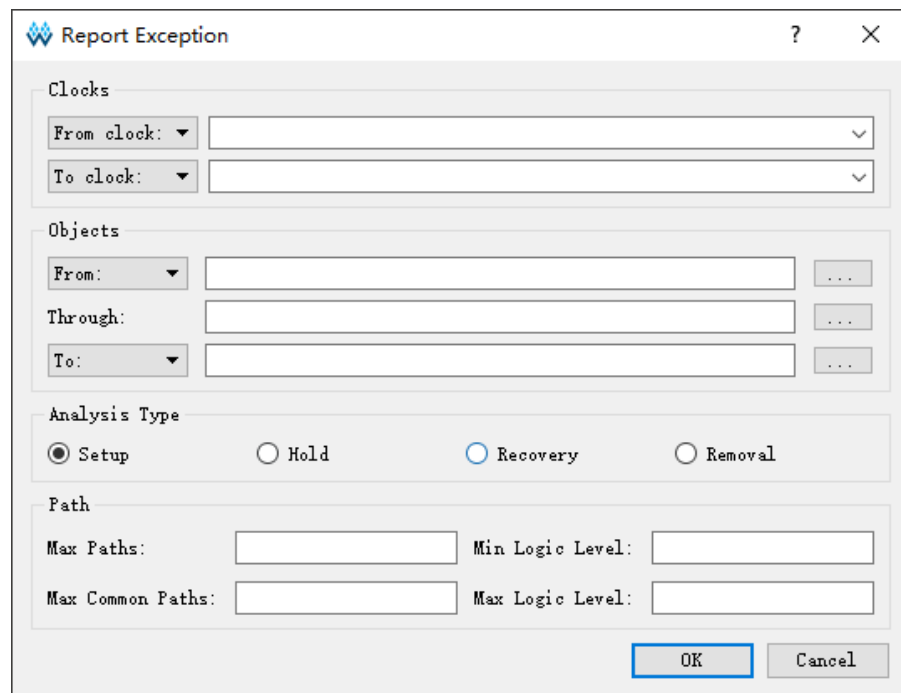


3. 选择“Create Report”，弹出如图 4-39 所示的对话框。

注！

窗口选项简介请参考 Report Timing。

图 4-39 Report Exception 对话框



4. 填写对话框中相关信息，单击“OK”，保存时序报告设置。

4.7.6 其它约束

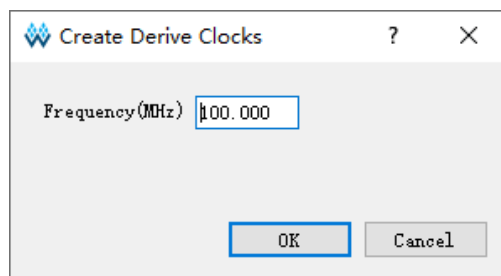
Create Derive Clocks

- 为设计创建一个全局目标的时钟频率；
- 支持频率设置，最大支持 1200MHz。

可通过以下两种方式新增 Derive Clocks 约束：

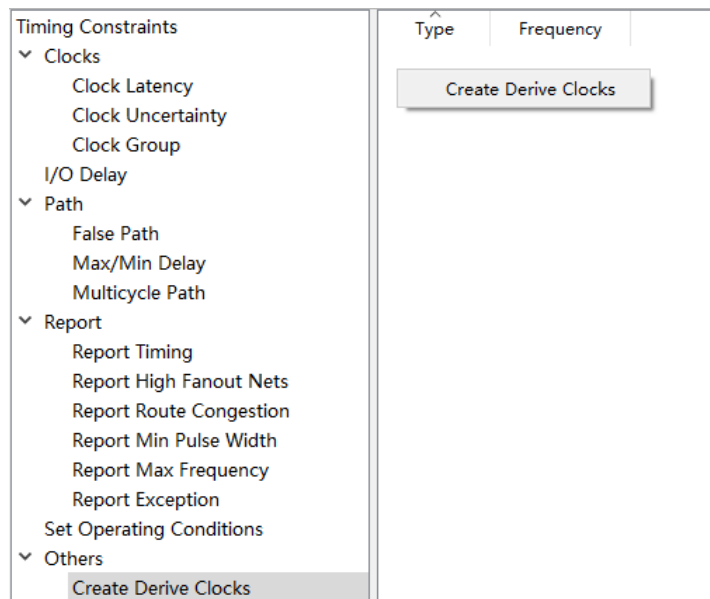
1. 通过 Constraints 菜单新增 Derive Clocks 约束。
 - a). 选择“Constraints > Create Derive Clocks...”，弹出“Create Derive Clocks”对话框，如图 4-40 所示。

图 4-40 创建 Derive Clocks



- b). Frequency(MHz): 全局目标频率，小于等于 1200 的正浮点数，精确到千分位。
2. 通过“Others > Create Derive Clocks”创建 Derive Clocks。在空白处右击选择“Create Derive Clocks”新建 Derive Clocks，如图 4-41 所示。

图 4-41 选择 Create Derive Clocks



时钟创建完成后，Derive Clocks 列表中会增加对应的约束，如图 4-42

所示。

图 4-42 Derive Clocks 列表

Timing Constraints	Type	Frequency
<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 ▼ Others <ul style="list-style-type: none"> Create Derive Clocks 	Derive	100MHz

4.7.7 保存与导出

所有约束编辑完成后，单击“File > Save”或“File > Save As”，可将当前时序约束编辑器中的约束信息保存至工程中的 SDC 文件中，时序约束文件内容格式请参考附录 A 时序约束语法规范。

4.8 时序约束的优先级

云源提供多种类型的时序约束，按照优先级由低到高依次如下所示：

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 时序报告

本章节将对高云半导体静态时序分析结果报告进行描述，方便用户快速了解时序报告内容。如图 5-1 所示，报告分为左侧导航栏和右侧内容栏，当存在不满足时序分析情况时左侧的导航栏对应的标题会显示红色。

图 5-1 静态时序分析报告

Timing Messages

- ▶ **Timing Summaries**
 - STA Tool Run Summary
 - Clock Summary
 - Max Frequency Summary
 - Total Negative Slack Summary
- ▶ **Timing Details**
 - ▶ **Path Slacks Table**
 - Setup Paths Table
 - Hold Paths Table
 - Recovery Paths Table
 - Removal Paths Table
 - Minimum Pulse Width Table
 - ▶ **Timing Report By Analysis Type**
 - Setup Analysis Report
 - Hold Analysis Report
 - Recovery Analysis Report
 - Removal Analysis Report
 - Minimum Pulse Width Report
 - High Fanout Nets Report
 - Route Congestions Report
 - ▶ **Timing Exceptions Report**
 - Setup Analysis Report
 - Hold Analysis Report
 - Recovery Analysis Report
 - Removal Analysis Report
 - Timing Constraints Report

Timing Summaries

STA Tool Run Summary:

Setup Delay Model	Slow 1.14V 85C C5/I4
Hold Delay Model	Fast 1.26V 0C C5/I4
Numbers of Paths Analyzed	42
Numbers of Endpoints Analyzed	17
Numbers of Falling Endpoints	0
Numbers of Setup Violated Endpoints	0
Numbers of Hold Violated Endpoints	0

Clock Summary:

NO.	Clock Name	Type	Period	Frequency(MHz)	Rise	Fall	Source	Master	Objects
1	clk0	Base	10.000	100.000	0.000	5.000			clk

Max Frequency Summary:

NO.	Clock Name	Constraint	Actual Fmax	Logic Level	Entity
1	clk0	100.000(MHz)	147.195(MHz)	4	TOP

Total Negative Slack Summary:

Clock Name	Analysis Type	Endpoints TNS	Number of Endpoints
clk0	Setup	0.000	0
clk0	Hold	0.000	0

Timing Details

Path Slacks Table:

5.1 Timing Summaries

时序综述（Timing Summaries）由四部分组成，分别是运行信息综述（STA Tool Run Summary）、时钟综述（Clock Summary）、最大频率综述（Max Frequency Summary）及终点余量为负值综述（Total Negative Slack Summary），如下图 5-2 所示。

图 5-2 Timing Summaries

Timing Summaries

STA Tool Run Summary:

Setup Delay Model	Slow 1.14V 85C C5/I4
Hold Delay Model	Fast 1.26V 0C C5/I4
Numbers of Paths Analyzed	42
Numbers of Endpoints Analyzed	17
Numbers of Falling Endpoints	0
Numbers of Setup Violated Endpoints	0
Numbers of Hold Violated Endpoints	0

Clock Summary:

NO.	Clock Name	Type	Period	Frequency(MHz)	Rise	Fall	Source	Master	Objects
1	clk0	Base	10.000	100.000	0.000	5.000			clk

Max Frequency Summary:

NO.	Clock Name	Constraint	Actual Fmax	Logic Level	Entity
1	clk0	100.000(MHz)	147.195(MHz)	4	TOP

Total Negative Slack Summary:

Clock Name	Analysis Type	Endpoints TNS	Number of Endpoints
clk0	Setup	0.000	0
clk0	Hold	0.000	0

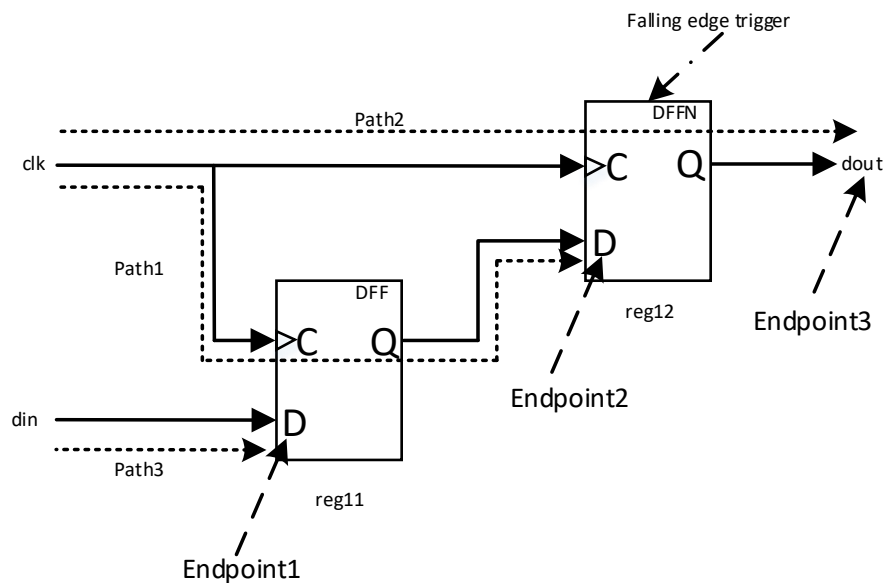
5.1.1 STA Tool Run Summary

- **Setup Delay Model:** 云源进行建立时间分析时使用的数据模型，默认采用 Slow 模型即高温低压；
- **Hold Delay Model:** 云源进行保持时间分析时使用的数据模型，默认采用 Fast 模型即低温高压；
- **Numbers of Paths Analyzed:** 静态时序分析路径的数量，如图 5-3 所示，共分析了 3 条时序路径，标记为 Path1、Path2 及 Path3；
- **Numbers of Endpoints Analyzed:** 分析的时序路径的终点，如图 5-3 所

示，共分析了 3 个终点，标记为 Endpoint1、Endpoint2 及 Endpoint3；

- **Numbers of Falling Endpoints:** 终点分析时触发方式为下降沿的数量，如图 5-3 所示，reg12 类型为 DFFN，触发方式为下降沿，则终点 D 即是下降沿触发终点；
- **Numbers of Setup Violated Endpoints:** 经时序分析后不满足建立时间的终点量；
- **Numbers of Hold Violated Endpoints:** 经时序分析后不满足保持时间的终点量。

图 5-3 Path & Endpoints



5.1.2 Clock Summary

报告用户设计中的所有时钟。若设计中的时钟未进行约束则软件会默认为其创建时钟，晨熙家族默认时钟为 100MHz，小蜜蜂家族默认时钟为 50MHz。此外含有 GAO 的设计，TCK 时钟频率为 20MHz；

- **NO.:** 序号；
- **Clock Name:** 时钟的名称。软件默认创建时钟时若时钟重名则会自动添加“_gowin”后缀。若为 PLL 类、OSC 类及 CLKDIV 类，软件会默认为时钟名添加“.default_gen_clk”后缀；
- **Type:** 有 Base、Generated 两种值。Base 表示基础时钟，Generated 表示衍生时钟；
- **Period:** 时钟的周期；
- **Frequency (MHz) :** 时钟频率；
- **Rise:** 时钟的上升沿时间；
- **Fall:** 时钟的下降沿时间；

- **Source:** 指时钟的获取源，可从 PORT、PIN、NET、REG 进行时钟获取；
- **Master:** 衍生出时钟的时钟即为主时钟；
- **Objects:** 时钟作用对象如 PORT、PIN、NET、REG。

5.1.3 Max Frequency Summary

- **NO.:** 序号；
- **Clock Name:** 驱动时序模型的时钟的名称；
- **Constraint:** SDC 约束的时钟频率或无 SDC 约束时默认的时钟频率；
- **Actual Fmax:** 云源 PnR 后经云源分析后得出的最大实际频率；
- **Logic Level:** 时钟驱动的最差时序路径的逻辑级数；
- **Entity:** 报告设计中模块的最大频率，默认为顶层模块即 TOP。

注！

- 当 PnR 后时钟没有驱动时序模型时则为 “No timing paths to get frequency of *”；
- 最大时钟频率仅报告被相同时钟驱动的时序模型（包含衍生时钟）上的时钟；
- 建议用户为设计添加完整的时序约束使云源能够更加精准的分析。

5.1.4 Total Negative Slack Summary

- **Clock Name:** 时钟名称；
- **Analysis Type:** 分析类型为 Setup 或 Hold 两种；
- **Endpoints TNS:** 统计时钟（对应 ClockName）驱动的时序路径上的终点余量为负值的时间总量，其中共同终点的路径只统计最差路径；
- **Number of Endpoints:** 统计时钟（对应 ClockName）驱动的时序路径上的终点余量为负值的终点个数总量，其中共同终点的路径只统计最差路径。

注！

仅报告被相同时钟驱动的时序模型。

5.2 Timing Details

5.2.1 Path Slacks Table

时序路径的静态分析余量表，分为 **Setup Paths Table**（建立时间路径分析表）、**Hold Paths Table**（保持时间路径分析表）、**Recovery Paths Table**（恢复时间路径分析表）、**Removal Paths Table**（移除时间路径分析表）。上述四类表头信息表达信息相同，图 5-4 表头说明如下：

- **Path Number:** 路径编号，默认最大报告 25 条；
- **Path Slack:** 其值等于数据请求时间减去数据到达时间，当为负值时时序不满足；
- **From Node:** 前级时序元件的时序分析开始节点；
- **To Node:** 后级时序元件的时序分析终止节点；
- **From Clock:** 前级时序元件的数据发送时钟以及发送边沿类型。其中发送边沿类型指的是上升沿或下降沿；
- **To Clock:** 后级时序元件的数据锁存时钟以及锁存边沿类型；
- **Relation:** 描述发送时钟和采样时钟之间的时间关系；
- **Clock Skew:** 时钟偏斜。发送时钟和锁存时钟到达前级和后级时序元件的时间差；
- **Data Delay:** 数据到达路径中的数据延迟，其数值是整个数据到达路径中延迟值的一部分。

注！

- 当没有可供分析的时序路径时则报告为 “Nothing to report!”；
- Path Slacks Table 默认分析最差的 25 条路径，如用户需要查看的路径不在 25 条范围内则可通过 SDC 约束命令 `report_timing` 进行报告，命令语法请参见 [Report Timing](#)；
- Path Slacks Table 默认分析包含跨时钟域时序路径，如用户不关心跨时钟域分析则可通过 `set_clk_group` 或 `set_false_path` 进行配置，命令语法请参见 [Set Clock Group](#) 或 [Set False Path](#)。

图 5-4 路径余量表

Path Slacks Table:

Setup Paths Table

Report Command: `report_timing -setup -max_paths 25 -max_common_paths 1`

Path Number	Path Slack	From Node	To Node	From Clock	To Clock	Relation	Clock Skew	Data Delay
1	8.806	synS_r_s0/Q	synE_r_s0/D	ck0:[R]	ck0:[R]	10.000	0.000	0.794

Hold Paths Table

Report Command: `report_timing -hold -max_paths 25 -max_common_paths 1`

Path Number	Path Slack	From Node	To Node	From Clock	To Clock	Relation	Clock Skew	Data Delay
1	0.570	synS_r_s0/Q	synE_r_s0/D	ck0:[R]	ck0:[R]	0.000	0.000	0.570

Recovery Paths Table

Report Command: `report_timing -recovery -max_paths 25 -max_common_paths 1`

Path Number	Path Slack	From Node	To Node	From Clock	To Clock	Relation	Clock Skew	Data Delay
1	8.649	rstSrc_r_s0/Q	rstObj_r_s0/CLEAR	ck0:[R]	ck1:[R]	10.000	0.000	1.278

Removal Paths Table

Report Command: `report_timing -removal -max_paths 25 -max_common_paths 1`

Path Number	Path Slack	From Node	To Node	From Clock	To Clock	Relation	Clock Skew	Data Delay
1	0.788	rstSrc_r_s0/Q	rstObj_r_s0/CLEAR	ck0:[R]	ck1:[R]	0.000	0.000	0.833

5.2.2 Minimum Pulse Width Table

时序元件可识别的最小脉冲宽度静态时序分析表。脉冲宽度指的是有效高/低电平信号持续的时间长度。默认报告最差的 10 条。图 5-5 表头信息说明如下：

- **Number:** 从小到大顺序序列号，默认 10 条；
- **Slack:** 元件可识别的最小脉冲宽度的余量值；
- **Actual Width:** 实际脉冲宽度，云源 PnR 后进行静态时序分析后得出的元件可识别的实际脉冲宽度；
- **Required Width:** 要求脉冲宽度，元件要求的可正常识别的最小脉冲宽度；
- **Type:** 脉冲宽度的类型，仅 Low Pulse Width 和 High Pulse Width 两种类型，分别为逻辑低电平脉冲宽度和逻辑高电平脉冲宽度；
- **Clock:** 进行最小脉冲宽度分析的时钟；
- **Objects:** 进行最小脉冲宽度分析的时序元件实例化对象。

注！

当无最小脉冲宽度分析报告时显示“Nothing to report!”。

图 5-5 最小脉冲宽度表

Minimum Pulse Width Table:

Report Command: report_min_pulse_width -nworst 10 -detail

Number	Slack	Actual Width	Required Width	Type	Clock	Objects
1	2.738	4.238	1.500	Low Pulse Width	DEFAULT_CLK	reg12
2	2.738	4.238	1.500	Low Pulse Width	DEFAULT_CLK	reg11_Z
3	2.813	4.313	1.500	High Pulse Width	DEFAULT_CLK	reg12
4	2.813	4.313	1.500	High Pulse Width	DEFAULT_CLK	reg11_Z

5.2.3 Timing Report By Analysis Type

该部分包含 Setup Analysis Report、Hold Analysis Report、Recovery Analysis Report、Removal Analysis Report 四类静态时序分析类型。其中，Setup Analysis Report 包含 Recovery Analysis Report，Hold Analysis Report 包含 Removal Analysis Report。分析计算方法一致。下面将对这四类分析类型进行介绍。

Setup Analysis Report

建立时间分析报告，用来分析设计中时序元件的时钟信号上升沿到达前，数据稳定不变的时间，如时间不够，数据将不能在时钟上升沿被稳定的送入时序元件。

云源对时序路径上的数据到达时间、数据请求时间、采样时钟、发送时钟等进行了详细的计算、分析并最终打印在建立时间分析报告中供用户参考。

该报告由命令 `report_timing -setup` 生成，云源默认分析并报告 25 条余量最差的时序路径，内容包含 **Path Summary**、**Data Arrival Path**、**Path Statistics**，释义如下所示。

1. **Path Summary**。图 5-6 为静态时序分析的路径信息综述，图中信息说明如下：

- **Slack**: 数据允许最迟到达时间减去数据实际到达时间。正值表示时序收敛，负值表示时序不收敛；
- **Data Arrival Time**: **Launch edge** 到达后级时序元件数据端口消耗的时间；
- **Data Required Time**: **Latch edge** 到达后级时序元件时钟端口消耗的时间；
- **From**: 前级时序元件；
- **To**: 后级时序元件；
- **Launch Clock**: 提供 **Launch edge** 的时钟以及作用边沿。作用边沿分为 **R** (**Rise**, 上升沿) 和 **F** (**Fall**, 下降沿) 两种；
- **Latch Clock**: 提供 **Latch edge** 的时钟以及作用边沿，作用边沿分为 **R** 和 **F**。

图 5-6 路径信息综述

Path Summary:

Slack	5.789
Data Arrival Time	6.767
Data Required Time	12.556
From	reg11_Z
To	reg12_Z
Launch Clk	sysclk1:[R]
Latch Clk	sysclk1:[R]

2. **Data Arrival Path**。图 5-7 为一条数据到达路径，图中信息说明如下：

- **AT**: 指某一时刻，是时序路径上的一个时间节点；
- **DELAY**: 指延时值，其值表示一段时间间隔；
- **TYPE**: 指时序分析路径上 **NODE** 的类型，当为空值表示不可用。

注！

图 5-7 中，**TYPE** 包含多种类型，含义如下：

- **tCL**: time of clock latency, 时钟源延迟；
- **tINS**: time of module instance, 实例化的元器件延迟；
- **tNET**: time of net, net 的延迟；
- **tC2Q**: time of clock to quit, 时序元件内部延迟。
- **RF**: 指的是当前被分析元件的翻转类型。**RR** 表示正脉冲不翻转，**FF** 表示负脉冲不翻转，**RF** 表示正脉冲向负脉冲翻转，**FR** 表示负脉

冲向正脉冲翻转；

- **FANOUT**: 扇出；
- **LOC**: 当前分析的元件在器件中的物理位置，没有位置信息的使用 UNPLACE 标记，如 DHCEN；
- **NODE**: 静态时序分析路径上的节点。包括实例化的名称加端口、时钟、时钟边沿激活时间（active clock edge time）。

图 5-7 数据到达路径

Data Arrival Path:

AT	DELAY	TYPE	RF	FANOUT	LOC	NODE
0.000	0.000					active clock edge time
0.000	0.000					sysclk1
0.000	0.000	tCL	RR	1	IOL7[A]	clk1_ibuf/I
0.943	0.943	tINS	RR	2	IOL7[A]	clk1_ibuf/O
3.236	2.293	tNET	RR	1	IOL2[B]	reg11_Z/CLK
3.786	0.550	tC2Q	RF	1	IOL2[B]	reg11_Z/Q
6.767	2.981	tNET	FF	1	R5C9[1][A]	reg12_Z/D

3. **Data Required Path**。如图 5-8 所示，数据请求路径是指时钟从有效沿开始到达时序元件时钟端口时所经过的路径。

注！

图 5-8 中，TYPE 含义如下：

- **tUnc**: time of clock uncertainty, 时钟的不确定值；
- **tSu**: time of setup, 建立时间。

图 5-8 数据请求路径

Data Required Path:

AT	DELAY	TYPE	RF	FANOUT	LOC	NODE
10.000	10.000					active clock edge time
10.000	0.000					sysclk1
10.000	0.000	tCL	RR	1	IOL7[A]	clk1_ibuf/I
10.943	0.943	tINS	RR	2	IOL7[A]	clk1_ibuf/O
13.236	2.293	tNET	RR	1	R5C9[1][A]	reg12_Z/CLK
13.036	-0.200	tUnc				reg12_Z
12.556	-0.480	tSu		1	R5C9[1][A]	reg12_Z

4. **Path Statistics**。图 5-9 为路径统计信息，图中信息说明如下。

- **Clock Skew**: 时钟倾斜；
- **Setup Relationship**: 前级时序元件发送数据，后级时序元件锁存数据的时间关系；
- **Logic Level**: 两个时序元件之间的逻辑元件数量，0 代表直接相连；
- **Arrival Clock Path Delay**: 统计了 Data Arrival Path 上时钟延时的情况，cell 表示逻辑元件延迟，route 表示绕线延迟，tC2Q 表示时序元件内部延迟；
- **Arrival Data Path Delay**: 统计了 Data Arrival Path 上数据的延时情况；

- **Required Clock Path Delay:** 统计了 Data Required Path 上时钟的延时情况。

图 5-9 路径统计信息

Path Statistics:

Clock Skew	0.000
Setup Relationship	10.000
Logic Level	1
Arrival Clock Path Delay	cell: 0.943, 29.131%; route: 2.293, 70.869%
Arrival Data Path Delay	cell: 0.000, 0.000%; route: 2.981, 84.423%; tC2Q: 0.550, 15.577%
Required Clock Path Delay	cell: 0.943, 29.131%; route: 2.293, 70.869%

Hold Analysis Report

图 5-10 为保持时间分析报告,分析在时序元件的时钟信号上升沿到达之后,数据稳定不变的时间,如时间不足,数据不能被稳定的送入时序元件。云源对设计中的时序路径上的数据到达时间、数据请求时间、采样时钟、发送时钟等进行了详细的计算、分析并最终生成报告。该报告由命令 `report_timing -hold` 生成,默认报告 25 条余量最差的时序路径。报告表头信息解释请参考 Setup Analysis Report。

图 5-10 保持时间分析报告

Hold Analysis Report						
Report Command:report_timing -hold -max_paths 25 -max_common_paths 1						
Path1						
Path Summary:						
Slack	1.003					
Data Arrival Time	3.554					
Data Required Time	2.551					
From	reg11_s0					
To	reg12_s0					
Launch Clk	sysclk:[R]					
Latch Clk	sysclk:[R]					
Data Arrival Path:						
AT	DELAY	TYPE	RF	FANOUT	LOC	NODE
0.000	0.000					active clock edge time
0.000	0.000					sysclk
0.000	0.000	tCL	RR	1	IOL11[A]	clk_ibuf/I
0.811	0.811	tINS	RR	2	IOL11[A]	clk_ibuf/O
2.533	1.723	tNET	RR	1	R2C9[0][A]	reg11_s0/CLK
2.933	0.400	tC2Q	RR	1	R2C9[0][A]	reg11_s0/Q
3.554	0.621	tNET	RR	1	R2C9[1][A]	reg12_s0/CLEAR
Data Required Path:						
AT	DELAY	TYPE	RF	FANOUT	LOC	NODE
0.000	0.000					active clock edge time
0.000	0.000					sysclk
0.000	0.000	tCL	RR	1	IOL11[A]	clk_ibuf/I
0.811	0.811	tINS	RR	2	IOL11[A]	clk_ibuf/O
2.533	1.723	tNET	RR	1	R2C9[1][A]	reg12_s0/CLK
2.533	0.000	tUnc				reg12_s0
2.551	0.018	tHld		1	R2C9[1][A]	reg12_s0
Path Statistics:						
Clock Skew	0.000					
Hold Relationship	0.000					
Logic Level	1					
Arrival Clock Path Delay	cell: 0.811, 31.998%; route: 1.723, 68.002%					
Arrival Data Path Delay	cell: 0.000, 0.000%; route: 0.621, 60.818%; tC2Q: 0.400, 39.182%					
Required Clock Path Delay	cell: 0.811, 31.998%; route: 1.723, 68.002%					

Recovery Analysis Report

图 5-11 为恢复时间分析报告，指分析时序元件在时钟有效沿前，异步清零、置位、复位信号需保持稳定的最短时间，如不满足该时间，则触发器可能无法进入正常工作状态。恢复时间的分析、计算方法与建立时间一致，该报告由命令 `report_timing -recovery` 生成，云源默认分析并报告 25 条余量最差的时序路径，表头信息请参考 [Setup Analysis Report](#)。

图 5-11 恢复时间分析报告

Recovery Analysis Report							
Report Command: report_timing -recovery -max_paths 25 -max_common_paths 1							
Path1							
Path Summary:							
Slack	8.355						
Data Arrival Time	4.629						
Data Required Time	12.984						
From	reg11_s0						
To	reg12_s0						
Launch Clk	sysclk:[R]						
Latch Clk	sysclk:[R]						
Data Arrival Path:							
AT	DELAY	TYPE	RF	FANOUT	LOC	NODE	
0.000	0.000					active clock edge time	
0.000	0.000					sysclk	
0.000	0.000	tCL	RR	1	IOL11[A]	clk_ibuf/I	
0.943	0.943	tINS	RR	2	IOL11[A]	clk_ibuf/O	
3.236	2.293	tNET	RR	1	R2C9[0][A]	reg11_s0/CLK	
3.786	0.550	tC2Q	RF	1	R2C9[0][A]	reg11_s0/Q	
4.629	0.843	tNET	FF	1	R2C9[1][A]	reg12_s0/CLEAR	
Data Required Path:							
AT	DELAY	TYPE	RF	FANOUT	LOC	NODE	
10.000	10.000					active clock edge time	
10.000	0.000					sysclk	
10.000	0.000	tCL	RR	1	IOL11[A]	clk_ibuf/I	
10.943	0.943	tINS	RR	2	IOL11[A]	clk_ibuf/O	
13.236	2.293	tNET	RR	1	R2C9[1][A]	reg12_s0/CLK	
13.036	-0.200	tUnc				reg12_s0	
12.984	-0.052	tSu		1	R2C9[1][A]	reg12_s0	
Path Statistics:							
Clock Skew	0.000						
Setup Relationship	10.000						
Logic Level	1						
Arrival Clock Path Delay	cell: 0.943, 29.131%; route: 2.293, 70.869%						
Arrival Data Path Delay	cell: 0.000, 0.000%; route: 0.843, 60.531%; tC2Q: 0.550, 39.469%						
Required Clock Path Delay	cell: 0.943, 29.131%; route: 2.293, 70.869%						

Removal Analysis Report

图 5-12 为移除时间分析报告，分析时序元件在时钟有效沿后，异步清零、置位、复位信号需保持稳定的最短时间，如不满足该时间，则触发器可能无法进入正常工作状态。移除时间的分析、计算方法与保持时间一致，该报告由命令 `report_timing -removal` 生成，云源默认分析并报告 25 条余量最差的时序路径，表头信息请参考 [Hold Analysis Report](#)。

图 5-12 移除时间分析报告

Removal Analysis Report

Report Command: report_timing -removal -max_paths 25 -max_common_paths 1

Path1

Path Summary:

Slack	1.003
Data Arrival Time	3.554
Data Required Time	2.551
From	reg11_s0
To	reg12_s0
Launch Clk	sysclk:[R]
Latch Clk	sysclk:[R]

Data Arrival Path:

AT	DELAY	TYPE	RF	FANOUT	LOC	NODE
0.000	0.000					active clock edge time
0.000	0.000					sysclk
0.000	0.000	tCL	RR	1	IOL11[A]	clk_ibuf/I
0.811	0.811	tINS	RR	2	IOL11[A]	clk_ibuf/O
2.533	1.723	tNET	RR	1	R2C9[0][A]	reg11_s0/CLK
2.933	0.400	tC2Q	RR	1	R2C9[0][A]	reg11_s0/Q
3.554	0.621	tNET	RR	1	R2C9[1][A]	reg12_s0/CLEAR

Data Required Path:

AT	DELAY	TYPE	RF	FANOUT	LOC	NODE
0.000	0.000					active clock edge time
0.000	0.000					sysclk
0.000	0.000	tCL	RR	1	IOL11[A]	clk_ibuf/I
0.811	0.811	tINS	RR	2	IOL11[A]	clk_ibuf/O
2.533	1.723	tNET	RR	1	R2C9[1][A]	reg12_s0/CLK
2.533	0.000	tUnc				reg12_s0
2.551	0.018	tHld		1	R2C9[1][A]	reg12_s0

Path Statistics:

Clock Skew	0.000
Hold Relationship	0.000
Logic Level	1
Arrival Clock Path Delay	cell: 0.811, 31.998%; route: 1.723, 68.002%
Arrival Data Path Delay	cell: 0.000, 0.000%; route: 0.621, 60.818%; tC2Q: 0.400, 39.182%
Required Clock Path Delay	cell: 0.811, 31.998%; route: 1.723, 68.002%

5.2.4 Minimum Pulse Width Report

最小脉冲宽度报告分析所有参与时序分析的路径上时序元件的最小脉冲宽度，包括高电平最小脉冲和低电平最小脉冲两种。如图 5-13 所示，图中信息说明如下：

- **Actual Width:** 实际脉冲宽度，其值为被分析目标上脉冲宽度实际维持的时间长度即是 Early clock Path 减去 Late clock Path 的值；
- **Required Width:** 元件要求的最小识别宽度即是脉冲信号维持的最短时间，小于这个宽度，那么这个电平脉冲将不能被识别；
- **Slack:** 脉冲宽度余量，其值为实际脉冲宽度减去请求脉冲宽度；
- **Type:** 指明脉冲类型。有两种值 Low Pulse Width 与 High Pulse Width，分别为低脉冲宽度和高脉冲宽度；
- **Clock:** 进行静态时序分析的时钟；
- **Objects:** 当前分析的时序元件；
- **Late clock Path:** 脉冲起始时刻开始分析的路径，对于高脉冲宽度是逻辑

高信号起始时刻开始分析的路径，对于低脉冲宽度是逻辑低信号起始时刻开始分析的路径；

- **Early clock Path:** 脉冲结束时刻开始分析的路径，对于高脉冲宽度是逻辑高信号结束时刻开始分析的路径，对于低脉冲宽度是逻辑低信号结束时刻开始分析的路径。

图 5-13 最小脉冲宽度

MPW Summary:

Slack:	2.738
Actual Width:	4.238
Required Width:	1.500
Type:	Low Pulse Width
Clock:	sysclk1
Objects:	reg12_Z

Late clock Path:

AT	DELAY	TYPE	RF	NODE
5.000	0.000			active clock edge time
5.000	0.000			sysclk1
5.000	0.000	tCL	FF	clk1_ibuf/I
5.945	0.945	tINS	FF	clk1_ibuf/O
8.295	2.350	tNET	FF	reg12_Z/CLK

Early clock Path:

AT	DELAY	TYPE	RF	NODE
10.000	0.000			active clock edge time
10.000	0.000			sysclk1
10.000	0.000	tCL	RR	clk1_ibuf/I
10.811	0.811	tINS	RR	clk1_ibuf/O
12.533	1.723	tNET	RR	reg12_Z/CLK

5.2.5 High Fanout Nets Report

高扇出报告分析所有参与时序分析的路径中 net 的扇出情况，同时还会分析这个 net 的最差 Slack，最大延时。默认分析 10 条。依照 FANOUT 值由大到小顺序排序，如图 5-14 所示，图中信息说明如下：

- **FANOUT:** 指明分析的 net 其扇出是多少；
- **NET NAME:** 指明当前分析的 net 名称；
- **WORST SLACK:** 指明当前分析的 net 上所存在的最差 Slack，一条 net 上可能存在不止一个 Slack；
- **MAX DELAY:** 指明当前分析 net 上最大延时。

图 5-14 高扇出报告

High Fanout Nets Report:

Report Command: report_high_fanout_nets -max_nets 10

FANOUT	NET NAME	WORST SLACK	MAX DELAY
2	clk1_c	5.789	2.350
2	clk2_c	17.616	2.350
1	reg21_i	17.616	0.000
1	reg11	5.789	2.981
1	reg21	17.616	0.403

5.2.6 Route Congestions Report

图 5-15 为绕线拥塞报告，图中信息说明如下：

- GRID LOC: 分析的 Grid 位置；
- ROUTE CONGESTIONS: Grid 上绕线的拥塞度，如 0.056 表示该 Grid 上的拥塞度为 5.6%；
- 默认报告 10 条最差的，按照 ROUTE CONGESTIONS 值的大小由大到小顺序排列。

图 5-15 绕线拥塞报告

Route Congestions Report:

Report Command: report_route_congestion -max_grids 10

GRID LOC	ROUTE CONGESTIONS
R5C9	0.056
R2C1	0.028
R3C1	0.028
R3C9	0.028
R1C1	0.014
R5C1	0.014

5.2.7 Timing Exceptions Report

下面通过一个实际案例进行时序例外报告说明。

针对图 5-16 案例，设计一个特定的 SDC 文件，文件内容如图 5-17 所示。

图 5-16 测试案例

```

1 module timing(
2   output dout,
3   input din, clk1, clk2
4 );
5
6   reg reg11, reg12;
7   reg reg21, reg22;
8
9
10
11   always @(posedge clk1)
12   begin
13     reg11 <= din;
14     reg12 <= reg11;
15   end
16
17   always @(posedge clk2)
18   begin
19     reg21 <= din;
20     reg22 <= ~reg21;
21   end
22
23   assign dout = reg22 & reg12;
24
25 endmodule

```

图 5-17 Timing Exceptions 约束

```
create_clock -name sysclk1 -period 10 -waveform {0 5} [get_ports {clk1}]
create_clock -name sysclk2 -period 10 -waveform {0 5} [get_ports {clk2}]
set_max_delay -from [get_clocks {sysclk1}] -to [get_clocks {sysclk1}] 5
set_max_delay -from [get_clocks {sysclk2}] -to [get_clocks {sysclk2}] 4
```

图 5-17 中的时序例外约束语句 `set_max_delay`，是将 `sysclk1`、`sysclk2` 影响的时序路径上的最大绝对延迟值分别设定为 5ns、4ns。`set_max_delay` 会对 `setup` 分析产生影响且受影响的路径会默认显示在时序例外报告下，默认产生的报告如下图 5-18 所示。

图 5-18 时序例外报告

Timing Exceptions Report:

Setup Analysis Report

Report Command:report_exceptions -setup -max_paths 5 -max_common_paths 1

Timing Path Constraint[1]: set_max_delay -from [get_clocks {sysclk1}] -to [get_clocks {sysclk1}] 5

Path1

Path Summary:

Slack	0.789
Data Arrival Time	6.767
Data Required Time	7.556
From	reg11_Z
To	reg12_Z
Launch Clk	sysclk1:[R]
Latch Clk	sysclk1:[R]

Data Arrival Path:

AT	DELAY	TYPE	RF	FANOUT	LOC	NODE
0.000	0.000					active clock edge time
0.000	0.000					sysclk1
0.000	0.000	tCL	RR	1	IOL7[A]	clk1_ibuf/I
0.943	0.943	tINS	RR	2	IOL7[A]	clk1_ibuf/O
3.236	2.293	tNET	RR	1	IOL2[B]	reg11_Z/CLK
3.786	0.550	tC2Q	RF	1	IOL2[B]	reg11_Z/Q
6.767	2.981	tNET	FF	1	R5C9[1][A]	reg12_Z/D

Data Required Path:

AT	DELAY	TYPE	RF	FANOUT	LOC	NODE
5.000	5.000					active clock edge time
5.000	0.000					sysclk1
5.000	0.000	tCL	RR	1	IOL7[A]	clk1_ibuf/I
5.943	0.943	tINS	RR	2	IOL7[A]	clk1_ibuf/O
8.236	2.293	tNET	RR	1	R5C9[1][A]	reg12_Z/CLK
8.036	-0.200	tUnc				reg12_Z
7.556	-0.480	tSu		1	R5C9[1][A]	reg12_Z

Path Statistics:

Clock Skew	0.000
Setup Relationship	5.000
Logic Level	1
Arrival Clock Path Delay	cell: 0.943, 29.131%; route: 2.293, 70.869%
Arrival Data Path Delay	cell: 0.000, 0.000%; route: 2.981, 84.423%; tC2Q: 0.550, 15.577%
Required Clock Path Delay	cell: 0.943, 29.131%; route: 2.293, 70.869%

Timing Path Constraint[14]: set_max_delay -from [get_clocks {sysclk2}] -to [get_clocks {sysclk2}] 4

Path1

Path Summary:

Slack	1.616
Data Arrival Time	4.940
Data Required Time	6.556
From	reg21_Z
To	reg22_Z
Launch Clk	sysclk2:[R]
Latch Clk	sysclk2:[R]

Data Arrival Path:

AT	DELAY	TYPE	RF	FANOUT	LOC	NODE
0.000	0.000					active clock edge time
0.000	0.000					sysclk2
0.000	0.000	tCL	RR	1	IOL5[A]	clk2_ibuf/I
0.943	0.943	tINS	RR	2	IOL5[A]	clk2_ibuf/O
3.236	2.293	tNET	RR	1	R5C9[0][B]	reg21_Z/CLK
3.786	0.550	tC2Q	RR	1	R5C9[0][B]	reg21_Z/Q
4.189	0.403	tNET	RR	1	R5C9[0][A]	reg21_i_c2/I0
4.940	0.751	tINS	RF	1	R5C9[0][A]	reg21_i_c2/F
4.940	0.000	tNET	FF	1	R5C9[0][A]	reg22_Z/D

Data Required Path:

AT	DELAY	TYPE	RF	FANOUT	LOC	NODE
4.000	4.000					active clock edge time
4.000	0.000					sysclk2
4.000	0.000	tCL	RR	1	IOL5[A]	clk2_ibuf/I
4.943	0.943	tINS	RR	2	IOL5[A]	clk2_ibuf/O
7.236	2.293	tNET	RR	1	R5C9[0][A]	reg22_Z/CLK

时序例外报告默认报告所有的受时序例外约束语句影响的路径，云源提供 report_exception 约束命令，允许用户配置和显示关心的部分报告内容，将不关心的报告路径进行过滤。如图 5-19 所示是在图 5-17 基础上再添加 report_exception 语句，红框里第一行表示受 sysclk1 影响的路径报告一条

setup 分析，第二行表示受 sysclk2 影响的路径不进行 setup 分析报告。

图 5-19 report_exception 语句

```
create_clock -name sysclk1 -period 10 -waveform {0 5} [get_ports {clk1}]
create_clock -name sysclk2 -period 10 -waveform {0 5} [get_ports {clk2}]
set_max_delay -from [get_clocks {sysclk1}] -to [get_clocks {sysclk1}] 5
set_max_delay -from [get_clocks {sysclk2}] -to [get_clocks {sysclk2}] 4
report_exceptions -setup -from_clock [get_clocks {sysclk1}] -to_clock [get_clocks {sysclk1}] -max_paths 1 -max_common_paths 1
report_exceptions -setup -from_clock [get_clocks {sysclk2}] -to_clock [get_clocks {sysclk2}] -max_paths 0 -max_common_paths 0
```

图 5-19 约束后的时序例外报告如图 5-20 所示。

图 5-20 report_exception 报告

Timing Exceptions Report:

Setup Analysis Report

Setup Analysis Report[1]:

Report Command: report_exceptions -setup -from_clock [get_clocks {sysclk1}] -to_clock [get_clocks {sysclk1}] -max_paths 1 -max_common_paths 1

Timing Path Constraint[1]: set_max_delay -from [get_clocks {sysclk1}] -to [get_clocks {sysclk1}] 5

Path1

Path Summary:

Slack	-0.654
Data Arrival Time	7.947
Data Required Time	7.293
From	reg11_ins23
To	reg12_ins20
Launch Clk	sysclk1:[R]
Latch Clk	sysclk1:[R]

Data Arrival Path:

AT	DELAY	TYPE	RF	FANOUT	LOC	NODE
0.000	0.000					active clock edge time
0.000	0.000					sysclk1
0.000	0.000	tCL	RR	1	IOL15[A]	clk1_ibuf13/I
0.982	0.982	tINS	RR	2	IOL15[A]	clk1_ibuf13/O
2.893	1.911	tNET	RR	1	IOL2[B]	reg11_ins23/CLK
3.351	0.458	tC2Q	RF	1	IOL2[B]	reg11_ins23/Q
7.947	4.596	tNET	FF	1	R15C23[1][A]	reg12_ins20/D

Data Required Path:

AT	DELAY	TYPE	RF	FANOUT	LOC	NODE
5.000	5.000					active clock edge time
5.000	0.000					sysclk1
5.000	0.000	tCL	RR	1	IOL15[A]	clk1_ibuf13/I
5.982	0.982	tINS	RR	2	IOL15[A]	clk1_ibuf13/O
7.893	1.911	tNET	RR	1	R15C23[1][A]	reg12_ins20/CLK
7.693	-0.200	tUnc				reg12_ins20
7.293	-0.400	tSu		1	R15C23[1][A]	reg12_ins20

Path Statistics:

Clock Skew	0.000
Setup Relationship	5.000
Logic Level	1
Arrival Clock Path Delay	cell: 0.982, 33.942%; route: 1.911, 66.058%
Arrival Data Path Delay	cell: 0.000, 0.000%; route: 4.596, 90.932%; tC2Q: 0.458, 9.068%
Required Clock Path Delay	cell: 0.982, 33.942%; route: 1.911, 66.058%

5.2.8 Timing Constraints Report

图 5-21 为时序约束报告，图中信息说明如下：

- **SDC Command Type:** 静态时序约束命令的类型；
- **State:** 包含 Invalid、Actived 两个值，Actived 表示命令生效，Invalid 表示命令无效；
- **Detail Command:** 其值等于 SDC 文件中对应的时序约束语句。

图 5-21 时序约束报告

Timing Constraints Report:

SDC Command Type	State	Detail Command
TC_CLOCK	Activated	create_clock -name main -period 18.182 -waveform {0 9.091} [get_ports {clk}]
TC_GENERATED_CLOCK	Activated	create_generated_clock -name main_gen -source [get_ports {clk}] -master_clock main -divide_by 5 -duty_cycle 40 -phase 22 -offset 50 [get_ports {in}]
TC_INPUT_DELAY	Activated	set_input_delay -clock main_gen 0.2 -clock_fall -add_delay -source_latency_included [get_ports {in}]
TC_CLOCK_LATENCY	Activated	set_clock_latency -source 1.2 [get_clocks {main}]
TC_CLOCK_UNCERTAINTY	Activated	set_clock_uncertainty 2.3 -setup -from [get_clocks {main}] -to [get_clocks {main}]
TC_FALSE_PATH	Activated	set_false_path -from [get_clocks {main_gen}] -to [get_clocks {main_gen}]
TC_MULTICYCLE	Activated	set_multicycle_path -from [get_clocks {main_gen}] -to [get_clocks {main_gen}] -setup -end 3
TC_MAX_DELAY	Activated	set_max_delay -from [get_clocks {main}] -to [get_clocks {main}] 1.11
TC_CLOCK_GROUP	Activated	set_clock_groups -exclusive -group [get_clocks {main}] -group [get_clocks {main_gen}]

附录 A 时序约束语法规范

高云时序约束语法规范参考标准的 SDC (Synopsys Design Constraint) 语法格式，可有效的对设计进行时序约束以满足特定的时序要求。

支持通配符“?”和“*”的使用，“?”匹配一个字符，“*”可匹配零个或多个字符且支持跨 hierarchy 层级匹配，同时支持时序约束分多行。

A.1 时钟约束

A.1.1 create_clock

语法

命令: create_clock

参数: -period <period_value>

[-name <clock_name>]

[-waveform <edge_list>]

<objects>

[-add]

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

<objects>: 用于指定创建时钟的目标，支持集合 `get_ports`、`get_pins`、`get_nets` 及 `get_regs`。如果用户选择的目标上已经创建了时钟，用户可使用 **-add** 命令来创建新的时钟，如用户未使用 **-add** 命令，则云源会忽略该条命令，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目标，则云源将会忽略这条命令，不会正确创建时钟。

应用示例

#通配符?使用示例，匹配一个字符，如 `clk`、`cck`。

```
create_clock -name ck -period 100 -waveform {0 50} [get_ports {c?k}]
```

#通配符*使用示例，匹配零个或多个单个字符，如 `clk`、`clock`。

```
create_clock -name ck -period 100 -waveform {0 50} [get_ports {c*k}]
```

#通配符*使用示例，匹配如 `uut/rpll_inst/CLKOUT`。

```
create_clock -name cck0 -period 25 -waveform {0 12.5} [get_pins {uut/r*_inst/CLKOUT}]
```

#使用转义字符示例，避免语法错误。

```
create_clock -name cck -period 25 -waveform {0 12.5} [get_pins {uut\*_inst/CLKOUT}]
```

```
create_clock -name cck -period 25 -waveform {0 12.5} [get_pins {?ut/rpll_inst/CLKOUT}]
```

#创建一个周期 10ns，下降沿先达的时钟，时钟名默认为 `clk`。

```
create_clock -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。

```
create_clock -period 10 -name clk [get_ports {clk}] #成功创建时钟 clk
```

```
create_clock -period 10 -name clk1 [get_ports {clk}] #由于缺少-add参数，命令 2 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
```

#具有四输入的 DCS，使用 **-add** 在时钟输出端口上创建四个时钟。

```
create_clock -name clk0 -period 10 -waveform {0 5} [get_pins
```



```
{dcs_inst/CLKOUT}]
    create_clock -name clk1 -period 10 -waveform {0 4} [get_pins
{dcs_inst/CLKOUT}] -add
    create_clock -name clk2 -period 10 -waveform {0 3} [get_pins
{dcs_inst/CLKOUT}] -add
    create_clock -name clk3 -period 10 -waveform {0 2} [get_pins
{dcs_inst/CLKOUT}] -add
```

A.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>]

<objects>

-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, 衍生时钟名称需唯一, 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, 如果来源存在有多个时钟, 则需通过 “-master_clock” 指定具体的主时钟, 支持集合 get_ports、get_pins、get_nets 及 get_regs;

-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；

-multiply_by: 设置衍生时钟相对于主时钟的倍频数；

-duty_cycle: 设置衍生时钟的占空比；

-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，云源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）；

-offset: 设置衍生时钟沿偏移量；

<objects>: 用来指定时钟的入口，支持集合 get_ports、get_pins、get_nets 及 get_regs。

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}]

```

A.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>

-source: 必选，表示时钟源延时；

-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；

-late | -early: 表示设置的是最大延时还是最小延时。对于 setup 分析，late 作用于 launch clock，early 作用于 latch clock。对于 hold 分析则与 setup 相反；

<delay>: 设置时钟源延时值，默认设置值为 0。

注！

late 的值应当大于等于 early 的值否则 late 将按照 early 的值计算。

-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，支持集合 get_clocks；

<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，支持集合 get_clocks、get_ports、get_pins、get_nets 及 get_regs。

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}]
```

#设置 clk0 上的时钟 cck 的上升沿时钟源延迟，并指定最晚最早值分别为 0.111、0.011。

```
set_clock_latency -source -rise -late 0.111 [get_ports {clk0}] -clock [get_clocks {cck}]
```

```
set_clock_latency -source -rise -early 0.011 [get_ports {clk0}] -clock [get_clocks {cck}]
```

#设置 clk0 上的时钟 cck 的下降沿时钟源延迟，并指定最晚最早值分别为 0.222、0.022。

```
set_clock_latency -source -fall -late 0.222 [get_ports {clk0}] -clock [get_clocks {cck}]
```

```
set_clock_latency -source -fall -early 0.022 [get_ports {clk0}] -clock [get_clocks {cck}]
```

#通配符使用示例，如匹配 uut/rpll_inst/CLKOUT。

```
set_clock_latency -source 0.123 [get_pins {u?t/r*_inst/^*OUT}] -clock [get_clocks {cck0}]
```

A.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>

-from/-rise_from/-fall_from: 指定该不确定性的起始时钟，其中“-rise_from”和“-fall_from”指定该不确定性的有效起始时钟沿，支持集合 `get_clocks`;

-to/-rise_to/-fall_to: 指定该不确定性的终点时钟，其中“-rise_to”和“-fall_to”指定该不确定性的终点有效时钟沿，支持集合 `get_clocks`;

-setup/-hold: 指定该不确定性是对建立时间还是保持时间产生影响，同一个约束语句互斥，若都不指定，则对这二种检查均有效;

<uncertainty value>: 不确定性值。

注!

需指明至少一个 `launch` 时钟或 `latch` 时钟，否则约束无效。

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
```

#设置 `launch` 为 `clk0` 的保持时间、建立时间不确定性为 0.111、0.222

```
set_clock_uncertainty 0.222 -setup -from [get_clocks {clk0}]
```

```
set_clock_uncertainty 0.111 -hold -from [get_clocks {clk0}]
```

#设置 `latch` 为 `clk1` 的保持时间、建立时间不确定性为 0.111、0.222

```
set_clock_uncertainty 0.222 -setup -to [get_clocks {clk1}]
```

```
set_clock_uncertainty 0.111 -hold -to [get_clocks {clk1}]
```

#设置 `launch` 为 `clk` 的保持时间、建立时间不确定性为 0.111

```
set_clock_uncertainty 0.111 -from [get_clocks {clk}]
```

A.1.5 set_clock_groups

语法

命令: `set_clock_groups`

参数: `[-asynchronous | -Exclusive]`

`[-group <clock name>] ...`

`-asynchronous | -Exclusive:` 指定时钟间关系为异步或互斥;

-group: 指定时钟为同一个组，支持使用集合 `get_clocks` 搜集一个或多个时钟。

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}]
```

A.2 I/O 延迟约束

A.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>`

-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，若无此参数，则默认与时钟上升沿关联;

-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，分别影响 `setup`、`hold`，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

输入延期内，若不指定则外部时钟延时不包含在输入延期内；

<delay_value>: 指定的输入延时值，默认为 0ns；

<port_list>: 指定受约束的输入端口（PORT），支持集合 get_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 进行覆盖
set_input_delay -clock clk1 -max 1.5 [get_ports {a}]
set_input_delay -clock clk1 -max 2.5 -add_delay [get_ports {a}]
#通配符示例，匹配 d0、d1 等
set_input_delay -clock cck0 -max 1.4 [get_ports {d*}]
```

A.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>

```

-clock: 参数“-clock”指定与输出延时相关的时钟；

-clock_fall: 指定输出延时与时钟下降沿相关，若不指定则默认与上升沿相关；

-rise/-fall: 指定上升沿或下降沿数据的输出延时，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出延时，分别影响 setup、hold，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出延时时内；

<delay_value>: 指定的输出延时值，默认为 0；

<port_list>: 指定受约束的输出端口（PORT），支持集合 get_ports。

应用举例

```

#设置端口 b 的外部输出延时为 0.5ns
set_output_delay -clock clk 0.5 [get_ports {b}]
#设置所有输出端口的的外部输出延时为 0.5ns
set_output_delay -clock clk 0.5 [all_outputs]
#设置端口 b 基于时钟下降沿的外部输出延时为 0.5ns
set_output_delay -clock clk -clock_fall 0.5 [get_ports {b}]
#设置端口 b 基于时钟上升沿的外部输出延时
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”使基于不同时钟沿外部输出延时同时有效
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}]
```

A.3 时序路径约束

A.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>

-from: 参数用于规定路径的起点, 支持的集合有 get_clocks、get_ports、get_regs、get_pins;

-to: 参数用于指定路径的终点, 支持的集合有 get_clocks、get_ports、get_regs、get_pins;

-through: 此参数用于指定路径经过的点或线, 支持集合 get_pins、get_nets, 当该参数搜集引脚 (PIN) 时, 只能是非时序元件的引脚 (PIN), 同一条约束中不允许使用多个“-through”参数;

<delay value>: 指定的输出延时值。

注!

- 约束 set_max_delay 影响 setup 时钟关系, set_min_delay 影响 hold 时钟关系;
- 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, 云源将忽略此约束, 不会对时序计算中产生影响。

应用举例

#设置 clk0 驱动的元素到 clk1 驱动的元素时序路径的最大延迟值为 5ns。

```
set_max_delay -from [get_clocks {clk0}] -to [get_clocks {clk1}] 5
```

```

#通配的示例，如端口 d00、d10 到触发器 r0、r1 的时钟关系为 2ns。
set_max_delay -from [get_ports {d*}] -to [get_regs {r?}] 2
#约束输入端口到 pin，影响 setup 分析，pin 到输出端口影响 hold 分析。
set_max_delay -from [all_inputs] -to [get_pins {r*/D}] 1.234
set_max_delay -from [get_pins {r?_s0/CLK}] -to [all_outputs] 0.989
#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 驱动的元素到 clock 驱动的元素的路径的最小延时为
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_regs {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_regs {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
#设置端口 a 到时钟 clk 以及所有数据端口到相关时钟的 setup 延迟关系。
set_max_delay -from [get_ports {a}] -to [get_clocks {clk}] 0.5
set_max_delay -from [all_inputs] -to [all_clocks] 0.111

```

A.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]

-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，如都没有指定则默认都对 setup、hold 均有效；

-from: 用于规定路径的起点，可通过集合 `get_ports`、`get_regs`、`get_pins` 或 `get_clocks` 来搜集起点，可以单独使用，云源自动获取相关终点；

-to: 用于指定路径的终点，可通过集合 `get_ports`、`get_regs`、`get_pins` 或 `get_clocks` 来搜集起点，可以单独使用，云源自动获取相关的起点；

-through: 此参数用于规定路径经过的点或线，可通过集合 `get_pins` 或者 `get_nets` 来搜集经过的点或线，该参数列表中可指定多个引脚(PIN)或者多个连线 (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！

如果使用集合 `get_pins` 则 `-from` 的值需是时钟 pin，`-to` 的值需是非时钟 pin，`-through` 的值需是数据路径上的输出 pin 如 `DFF.Q` 或接收 pin 如 `DFF.D`、`DFF.CE`。

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}]
```

#设置触发器 `reg0` 到触发器 `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}]
```

#单独使用 `-from`，并对 setup、hold 均有效。

```
set_false_path -from [get_pins {reg0_s0/CLK}]
```

```
set_false_path -from [get_regs {reg0_s0}]
```

```
set_false_path -from [get_clocks {cck}]
```

#单独使用 `-to`，并对 setup 有效。

```
set_false_path -from [get_regs {reg0_s0}] -setup
```

#单独使用 `-to`，并对 hold 有效。

```
set_false_path -from [get_regs {reg0_s0}] -hold
```

#单独使用 `-through`，经过 `reg0_s0.Q` 的时序路径不再分析。

```
set_false_path -through [get_pins {reg0_s0/Q}]
```

#单独使用 `-through`，经过 `reg0_c` 的时序路径不再分析。

```
set_false_path -through [get_nets {reg0_c}]
```

#"*"匹配多个字符，如 `mi/reg0`。

```
set_false_path -from [get_regs {mi/r*0}] -to [get_regs {spi/R*}]
```

#”?”匹配一个字符，如 reg0、reg1。

```
set_false_path -from [get_pins {mi/r?g0/CLK}] -to [get_pins {spi/DI}]
```

A.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>

-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）。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。默认对建立时间检查产生影响；

-from: 用于规定路径的起点，可通过集合 get_pins、get_ports、get_regs 或 get_clocks 来搜集起点；

-to: 用于指定路径的终点，可通过集合 get_pins、get_ports、get_regs 或 get_clocks 来搜集终点；

-through: 此参数用于规定路径经过的点或线，可通过集合 get_pins 或者 get_nets 来搜集经过的点或线；该参数列表中可指定多个引脚或者多个线，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；

<path multiplier>: 指定周期个数。

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，云源将忽略此条约束，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：参考时钟为触发器 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
```

#通配符“?”和“*”参考示例，“?”可匹配 addr0、addr1 等，“*”可匹配 Data_s0、D0_s0 等

```
set_multicycle_path -from [get_regs {SD/addr?}] -to [get_regs {RSG/D*_s0}]
```

A.4 工作条件约束

语法

命令：set_operating_conditions

参数：[-grade <c|l|a>]

[-model <slow|fast>]

[-speed <speed>]

[-setup]

[-hold]

[-max]

[-min]

[-max_min]

-grade: 指定器件的温度等级，目前支持商业级（commercial）、工业级（industrial）以及车规级（automotive）；

-model: 指定时序分析的时序模型；

-speed: 指定器件的速度等级；

-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。

应用举例

```
#设定工业级速度等级 6,模型快速, 影响 setup、hold 分析
set_operating_conditions -grade i -model fast -speed 6 -setup -hold
#设定商业级速度等级 7,模型慢速, 影响 setup、hold 分析
set_operating_conditions -grade c -model slow -speed 7 -max_min
```

A.5 时序报告内容约束

A.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 < value >]`

`[-max_logic_level < value >]`

`[-mod_ins {mod_ins1 mod_ins2 ...}]`

`-setup|-hold|-recovery|-removal`: 指定时序报告检查的类型, 互斥;

`-max_paths`: 指定时序报告的最大路径数, 默认是 25 条。如果指定路径的条数没有达到指定数目时, 会把未指定路径的最差路径来补全满足指定的数目。

`-max_common_paths`: 指定时序报告共享同一结束点路径的最大条数;

`-from/-rise_from/-fall_from`: 指定时序报告路径的起点, 其中

`-rise/fall_from` 需是时钟, 集合支持 `get_clocks`。单独使用时云源自动获取起点;

`-to /-rise_to /-fall_to`: 指定时序报告路径的终点, 其中 `-rise/fall_to` 需是时钟, 集合支持 `get_clocks`。单独使用时云源自动获取终点;

`-through`: 指定时序报告路径经过的点, 集合支持 `get_nets`、`get_pins`;

`-from_clock /-fall_from_clock /-rise_from_clock`: 指明时序报告路径的起点关联时钟, 集合支持 `get_clocks`。单独使用时云源自动获取终点;

`-to_clock /-rise_to_clock /-fall_to_clock`: 指明时序报告路径的终点关联时钟, 集合支持 `get_clocks`。单独使用时云源自动获取起点;

`-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
```

#报告的起点 `launch` 是 `ck`, 终点采用通配符可匹配 `r0`、`r1` 等。

```
report_timing -hold -rise_from [get_clocks {ck}] -to [get_pins {r*/D}]
```

```
report_timing -setup -fall_from [get_clocks {ck}] -to [get_regs {r*}]
```

#指定路径上的逻辑级数为 2, 最多报告 2 条路径且共同终点路径最多报告 1 条。

```
report_timing -recovery -from_clock [get_clocks {cck0}] -to_clock
[get_clocks {cck1}] -max_paths 2 -max_common_paths 1 -max_logic_level
2 -min_logic_level 2
```

#hold 分析只报告实例化 `uut` 模块内部的时序。

```
report_timing -hold -mod_ins { uut }
```

A.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>]

-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。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;

-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。

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。

```
report_high_fanout_Nets -max_nets 10
```

A.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>]`

`-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;

`-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;

`-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, 该参数的数值需不小于 `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]
```

A.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}]`

`-nworst`: 规定了报告多少条最差的路径;

`-min_pulse_width`: 规定报告时序元件上实际最小脉冲宽度;

`-max_pulse_width`: 规定了报告的时序元件上实际最大脉冲宽度;

`-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, 不指定该选项时, 默认对所有触发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 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 20 条时钟始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
```

A.5.5 report_max_frequency

语法

命令: `report_max_frequency`

参数: `-mod_ins {mod_ins1 mod_ins2 ...}`

`-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。

应用举例

报告 bsram0 的最大工作频率

```
report_max_frequency -mod_ins {bsram0}
```

A.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>]

```

其关键字的名称、含义以及使用与 `report_timing` 的关键字相同，对例外约束产生的路径进行报告。

应用举例

```

#显示 recovery 报告路径一条
create_clock -name mm -period 10 -waveform {0 5} [get_ports {clk}]
set_max_delay -from [get_clocks {mm}] -to [get_clocks {mm}] 0.22
report_exceptions -recovery -from_clock [get_clocks {mm}] -to_clock
[get_clocks {mm}] -max_paths 1 -max_common_paths 1

```

A.6 其它约束

A.6.1 derive_clocks

语法

命令: `derive_clocks`

参数: `-freq <value>`

`-freq`: 全局目标频率，小于等于 1200 的正浮点数，精确到千分位，单位为 MHz。

应用示例

```

#为全局创建一个频率为 100MHz 的时钟

```

derive_clocks -freq 100

